



TECHNISCHE
UNIVERSITÄT
WIEN

Vienna University of Technology

Dissertation

Motion Creating System Design and Implementation for a Biped Humanoid Robot

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der
technischen Wissenschaften unter der Leitung von

Em.o.Univ.Prof. Dipl.-Ing. Dr.tech. Dr.h.c.mult. Peter Kopacek

E325/A6

Institut für Mechanik und Mechatronik

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

Von

MSc. Siavash Dezfouli

Matrikelnummer 1028312

Wien, November 2013

Kurzfassung

Ziel der vorliegenden Arbeit ist es einen zweibeinigen Humanoiden Roboter, hinsichtlich des Gehens zu verbessern. Bei Beginn der Arbeit war der Unterkörper vorhanden.

Die erste Aufgabe bestand darin diese Hardware Struktur zu verbessern. Es wurden neue Fußgelenke sowie neue Fußplatten entwickelt, gefertigt und implementiert. Seitens der Elektrotechnik wurden neue, bürstenlose Motoren mit zeitgemäßen Zahnriemen implementiert und die Energieversorgung verbessert.

Von der IT Seite fanden neue USB/CAN Konverter Verwendung. Weiters wurde das Datenübertragungsnetzwerk neu gestaltet. Schließlich wurde eine neue Software zur Bewegungssteuerung einschließlich einer grafischen Benutzeroberfläche entwickelt.

Die entwickelte Bewegungssteuerung basiert auf einem Positions-Zeit (PT) System mit dem die Bewegungen der Gelenke koordiniert werden können. Bei dieser Methode werden die entsprechenden PT Daten aus seiner Quelldatei in einen Pufferspeicher geladen, die Motoren zu diesen Positionen bewegt und die Daten entsprechend dem "Control Area Network - CAN" formatiert. Durch Interpolation dieser Daten erfolgt die Berechnung der gewünschten Positionen und Geschwindigkeiten der Gelenke.

Durch diese Neuentwicklungen ist es möglich, verschiedene Arten des Gehens unter Anpassung an die Bodenbeschaffenheit realisiert werden. Mit dieser Methode ist es möglich mit dem derzeit vorhandenen Unterteil Schritte mit einer Länge von 50 cm und einer Höhe von 12 cm zu realisieren. Abschließende Tests ergaben, dass unter diesen Bedingungen der Roboter eine gerade Strecke von 10 m mit einer Geschwindigkeit von ungefähr 0,08 km/h zurücklegen konnte.

Abstract

This work effort, aims to develop the biped humanoid robot Archie, and present a simple but reliable motion creating system for stable bipedal walking. This new system for robot included developing the hardware structure (i.e., new frontal ankle joints, new brushless motors, new timing belts, USB to CAN converter, new power supply and new foot plates), designing a distributed communication network, new motion controller program with new graphical user interface and software implementation approaches to perform the stable walking.

Although various humanoid robots have successfully demonstrated their capabilities, but stable bipedal walking methods are still one of the main technical challenges that robotics researchers are attempting to solve it. If we consider this problem from a different standpoint, the development of a biped humanoid robot can be simplified as long as the bipedal walking method is easily formulated. Therefore, this thesis focuses on design and implementation of a motion control system based on the Position-Time (PT) mode and formulating the constraints of the hip and foot motion parameters.

In this method the PT data pulled from its source table into the buffer, therefore drive gets the PT position points and transmitted data according to the designed CAN (Control Area Network) format message. On the other hand, the motor drive interpolates the motion specification in order to calculate the desired position and speed at the sampling instances, when it needs the information.

In PT motion method the drive manages a read pointer for the position points vectors (QP[N]), when the read pointer is N, the active motion segment starts at position QP[N] and ends at QP[N+1], and after control sampling times (MP[4]), the drive increments the read pointer to N+1, and reads QP[N+2] to calculate the parameters of the next motion segment. PT data transferred to motor drive online, by using the designed distributed CAN communication network.

Therefore in the proposed system by varying the values of the constraints of the hip and foot motion parameters (X_{sd} , X_{ed} : distances along the x-axis from the hip to ankle of support foot at the start and end of single-support phase respectively) we can achieve different types of foot motion to adapt the ground conditions and run the stable walking. To validate the designed system, forward stable walking test performed on an ordinary room floor based on the obtained optimised values of the walking pattern (max. step height=12cm, max. step length=50cm, min. hip height=54cm, max. hip height=57cm). Finally Archie could walk forward for 10 meters and keeping balance along the entire sequence with the speed of ~ 0.08 km/h.

Acknowledgment

My father said, whenever you achieve a success you should not forget, those who supported you along the journey, therefore I give true gratitude to Prof. Peter Kopacek for his supervision, advice and guidance from the starting day of my PhD study in Vienna University of Technology. His passion and broad knowledge in robotics nourished my growth during these years. I'm indebted to him more than he knows.

I would like to say thanks to my amazing colleagues and friends: Paul Zinniel, Prof. Jacky Baltes, Dr. Chi Tai Cheng, Andrew Winton, Dr. Ahmad Byagowi, Peter Unterkreuter and Mohsen Mohamadi Daniali for all their help, and supports.

My deepest thanks to my Mom "Minoo Sodagar" and Dad "Saied Dezfouli" who have always dedicated their support and passion in every stage of my life and special sense of gratitude to my soul mate "Sepideh Mostofi" for her unconditional love.

Table of Contents

Kurzfassung	I
Abstract.....	II
Acknowledgment	III
Table of Contents	IV
List of Figures	IX
List of Tables	XII
Chapter 1 Problem Formulation	1
1.1. Problem Statement.....	1
Chapter 2 Introduction.....	3
2.1 Humanoid Robot Research	3
2.2 Biped Walking (Basic Functions)	6
2.2.1 Bipedal Motion System	7
2.2.2 Static Stability in Humanoid Robot	8
2.3 Thesis Approach.....	9
2.4 Chapter Organization	9
Chapter 3 Hardware Design and Developments of the Biped Humanoid Robot Archie	11
3.1 Introduction	11
3.2 Hardware Description	11
3.2.1 Mechanical Design of Robot	14
3.3 Performance Management in Archie Design	16
3.3.1 Degrees of Freedom (DOF) Selection.....	17
3.3.2 Actuator/Sensor Selection	18
3.3.3 Joint Actuator Mechanism	18
a) Transmissions.....	18
b) Harmonic Drive	18
c) Power Amplifiers.....	19
d) Low Power Hall Effect Switch	19
e) Power Supply	20
f) Motor Performance Management	20
3.4 Motor Drive Controller	21
3.4.1 Digital Servo Drive.....	21
3.5 New Hardware Improvements.....	22
3.5.1 New Ankle Joint Design and Manufacturing.....	22
3.5.2 Motor Type Selection for New Ankle Joint	25

3.5.3 New Timing Belts	27
3.5.4 New Encoder (PCB mounted)	28
a) Encoder Protection	29
3.5.5 Old Hardware Malfunction/Problems	29
Chapter 4 Design and Establishment of a Distributed Communication/Control Network	31
4.1 Introduction	31
4.2 Designed Real-Time Message Frame	31
4.2.1 Designed CAN Message Architecture	32
4.3 Communication Channels Establishment Approach.....	33
4.4 Designed Message Transmit Planning Approach.....	34
4.4.1 Designed Data Source Mechanism	34
4.4.2 Client-Server Interactions in Deigned Network	34
4.4.3 Process Design of Message Transmitting	35
4.4.4 Code Priority Transmission Scheduling.....	35
4.4.5 Inhibit Times.....	36
4.5 Process Data Objects (PDOs) Design.....	37
4.5.1 Receive PDOs Method	37
a) Message Error Detection Approach.....	37
4.5.2 PDOs Transmission Approach	38
4.5.3 Proposed PDO Mapping in Communication Network	38
4.5.4 Synchronous Trigger in Motion Controller Program.....	38
4.6 Network Management Message Structure (NMT)	38
4.6.1 SYNC and Time Stamp	39
4.6.2 Set and Query Commands in Network Controller	40
a) Binary Interpreter Commands	40
b) TPDO2 Structure in Controller Program	41
4.7 I-converter (USB to CAN)	42
4.7.1 Data Communication Bus via CAN-Bus Converter	43
Chapter 5 Motion Creating System Design based on PT and PVT Method.....	47
5.1 Introduction	47
5.2 Biped Robot Walking Problem	47
5.3 Gait Planning Method in Archie.....	49
5.3.1 Trajectory Planning Approach in Archie	49
5.3.2 Calculated Foot Trajectory.....	51
5.3.3 Calculated Hip Trajectory.....	53
5.4 Drive Motion Design Method	57

5.4.1 Servo Drive	57
5.4.2 Drive Reference Signal Generation Method	57
a) Drive Motion Reference Generator Approach	58
5.5 New Designed Drive Motion Command/Program Structure	58
5.5.1 Single Joint Motion Program (Pattern Design)	58
a) Communication Control Command Structure	59
b) Motor Drive Command Structure	59
c) Absolute Position Command Structure	59
d) Relative Position Command Structure	59
e) Begin Motion Command Structure	60
5.5.2 Multiple Joints Motion Program (Pattern Design)	61
5.5.3 Point-to-Point Method	61
5.6 New Motion Command Library	63
5.6.1 Motion Modes Changing Method	64
5.6.2 Position - Time (PT) Method	64
5.6.3 PT Table Design (QP Vector)	65
5.6.4 Position-Time Motion Scope	65
5.7 Stop Command Mechanism	68
5.7.1 Stop Manager Structure (Internal Elements)	68
5.8 Motor Fault Diagnosis and Error Detection Approach	70
5.8.1 Motor Failure Protection Mechanism	71
5.8.2 MF Command/Message Structure	71
Chapter 6 New Motion Control Program and New Software Architecture Design	73
6.1 Introduction	73
6.2 Motion Controller Program/Software (ver. 1.1 & 1.2)	73
6.2.1 Program/Software Architecture	74
a) Creating Program (Project)	74
b) GUI Design	75
c) Code editing	76
d) Building-qmake	76
e) Testing	76
f) Debugging	76
6.2.2 Class Creating (in C++)	77
6.3 Developed Program Structure Description (ver. 1.2)	78
6.3.1 Motion Controller Program GUI (ver. 1.1)	82
6.3.2 Scheme of the Motion Controller Program (ver. 1.2)	82

a) Motion Controller	82
b) Main Menu	82
c) Walking Direction.....	83
d) Communication Network + Motion Start and Stop.....	83
e) Walking Parameters.....	84
6.3.3 GUI Improvements in Motion Controller Program (ver. 1.2).....	87
6.4 New Motion Controller Program (ver. 2.1).....	90
6.4.1 Walking Control System in Program (ver. 2.1).....	91
6.4.2 Program Communication/Graphical User Interface Design	92
6.4.3 PT and PVT Mechanism in Software	94
6.4.4 Motion Management in PVT Method.....	95
6.4.5 PVT Motion Programming Message	98
Chapter 7 New Software Implementation and Forward Walking Test Results	101
7.1 Introduction	101
7.2 Software Implementation and Joint Motion Test.....	101
7.2.1 New Proposed CANbus Viewer Module Structure	102
7.2.2 Proposed Implemented Data Layers.....	103
7.2.3 Error Mechanism Implementation Approach	104
7.2.4 Proposed Message Controller Module	105
7.2.5 Real-Time Message Setting Approach	107
7.3 Joint Motion Test	108
7.3.1 Single Joint Motion Test.....	108
7.3.2 Continuous/Multiple Joints Motion Approach	110
7.3.3 Synchronized Joints Motion Approach	111
7.3 Preliminary Test	115
7.3.1 Half Gait Test.....	115
a) Several Tests (with different input parameters).....	117
7.4 Static Walking Test.....	118
7.4.1 Walking Simulation Results.....	120
7.4.2 Stable Walking Realization Test	121
7.4.3 Forward Walking Test Results (10 m)	125
7.4.4 Optimized W_m Value (Test Result)	127
Chapter 8 Summary, Outlook and Proposed Future Works	130
8.1 Summary and Outlook	130
8.2 Proposed Future Works	132
8.3 New Proposed Approaches and Requirements for Dynamic Walking	133

8.3.1 How to Improve the Stability Problem ? (Proposed Algorithm Method) 133

8.3.2 Absorbing Impact and Adapting to a Surface Problem..... 135

8.3.3 Attitude Problem 135

8.3.4 Proposed Upper Body Motion for Dynamic Walking..... 136

 a) Slope Angel of the Upper Body 136

References 137

 Appendix A 141

 Appendix B 148

List of Figures

Fig. 2.1 WL-1 vs. W-2R robot	3
Fig. 2.2 Developed humanoid robots.....	4
Fig. 2.3 a) Surena II vs. b) ASIMO standing on one leg (Guizzo, 2010)	8
Fig. 3.1 Archie hardware (updated structure from (Byagowi, 2010)).....	13
Fig. 3.2 CAD model of Archie	14
Fig. 3.3 Improved biped humanoid robot Archie.....	16
Fig. 3.4 Designed Archie lower body.....	17
Fig. 3.5 Operating characteristics of hall sensor/switch (www.diods.com, 2013)	19
Fig. 3.6 Drive motion controller connected to modular joint (Dezfouli & Mohamadi Daniali, 2012)	22
Fig. 3.7 New designed ankle joint parts.....	23
Fig. 3.8 New ankle joint parts and components (assembled to new foot plate and leg)	24
Fig. 3.9 Manufactured and assembled parts of new ankle joint + new PCB connections.....	24
Fig. 3.10 Assembled harmonic drive to new joint components	25
Fig. 3.11 Motor type selection criteria (www.maxonmotor.ch, 2013).....	25
Fig. 3.12 Designed PCB for brushless motor and connections description	27
Fig. 3.13 New timing belts for hip transversal joints	27
Fig. 3.14 Encoder pins connections on new designed PCB.....	28
Fig. 3.15 Archie standing on one leg.....	30
Fig. 4.1 Various fields in the designed CAN message frame for Archie	31
Fig. 4.2 Designed CAN bus line system for Archie	32
Fig. 4.3 Priority queue description.....	33
Fig. 4.4 Active/Passive error frame format in Designed CAN messages for Archie.....	37
Fig. 4.5 NMT message mapping	39
Fig. 4.6 Necessary USB to CAN converter features used in motion controller program	42
Fig. 4.7 USB to CAN converter connection establishment.....	44
Fig. 4.8 Code converting-setting mode.....	45
Fig. 4.9 CAN message structure design in test mode	46
Fig. 4.10 CAN message-receive signal format.....	46
Fig. 5.1 Biped walking phases	48
Fig. 5.2 Archie leg parameters	50
Fig. 5.3 Swing foot trajectory phases-time	51
Fig. 5.4 Archie stride	52
Fig. 5.5 Archie foot and hip trajectory calculation parameters	52
Fig. 5.6 Calculated leg and hip motion in x-axis.....	54
Fig. 5.7 Calculated ankle and hip trajectory of Archie	55
Fig. 5.8 Calculated left vs. right ankle frontal joint-angle trajectory	55
Fig. 5.9 Calculated left vs. right ankle lateral joint-angle trajectory	56
Fig. 5.10 Calculated left vs. right knee joint- angle trajectory	56
Fig. 5.11 Calculated left vs. right hip lateral joint-angle trajectory	57
Fig. 5.12 Single joint motion pattern	58
Fig. 5.13 Multiple joints motion commands order structure	61
Fig. 5.14 PTP motion decision flowchart (www.elmomc.com, 2013).....	62
Fig. 5.15 Motor drive command (designed pattern)	63
Fig. 5.16 PT motion flowchart.....	66
Fig. 5.17 PT auto increment mode flowchart	67

Fig. 5.18 Stop manager internal elements.....	69
Fig. 6.1 Necessary QT creator tools for motion controller program	75
Fig. 6.2 Qt creator platform	77
Fig. 6.3 Different created classes in motion controller program.....	78
Fig. 6.4 Data stream between modules in motion controller.....	80
Fig. 6.5 Motion controller GUI design (ver. 1.1)	82
Fig. 6.6 Scheme of the motion controller interface (ver. 1.2)	83
Fig. 6.7 Arm motion window.....	85
Fig. 6.8 Step parameters menu.....	85
Fig. 6.9 New GUI for motion controller program (ver. 1.2)	86
Fig. 6.10 Simulation of the humanoid robot Archie (including upper body motion)	87
Fig. 6.11 Different desired step path	88
Fig. 6.12 How interface in motion controller software works.....	89
Fig. 6.13 Main problems of old motion controller program.....	91
Fig. 6.14 Motion controller program flowchart (Dezfouli & Mohamadi Danial, 2012)	92
Fig. 6.15 Walking parameters (first stage).....	92
Fig. 6.16 Designing the new GUI	93
Fig. 6.17 Designed GUI for new motion controller program (ver. 2.1).....	93
Fig. 6.18 Designed PVT decision flowchart	97
Fig. 6.19 Auto increment PVT mode design.....	99
Fig. 7.1 Functions implemented in proposed CANbus Viewer module	102
Fig. 7.2 Proposed CANbus Viewer module implementation chart	104
Fig. 7.3 Normal transmission with ACK (Marais, 2008)	105
Fig. 7.4 Measured CANL normal transmission with ACK in Archie	105
Fig. 7.5 New joint message controller implementation approach	106
Fig. 7.6 Joint ID and message baudrate defining approach (1 Mbit/s).....	107
Fig. 7.7 Implemented zero position command's structure for lower body joints	108
Fig. 7.8 Single joint motion implementation	109
Fig. 7.9 Proposed implementation method for multiple joints motion	111
Fig. 7.10 Implemented cycle time for synchronized joint motion.....	112
Fig. 7.11 Input parameters for half gait test.....	115
Fig. 7.12 Desired vs. actual toe trajectory	115
Fig. 7.13 Calculated hip, knee and ankle lateral-joints angle trajectories (step length=30 cm).....	116
Fig. 7.14 Half gait simulation model (based on the elliptical trajectory).....	116
Fig. 7.15 Archie half gait performance.....	117
Fig. 7.16 Actual obtained trajectories of Archie (full gait motion test)	118
Fig. 7.17 Right and left leg phases during static walking test.....	118
Fig. 7.18 Proposed software implementation approach for Archie walking	119
Fig. 7.19 New motion controller program implementation method to perform stable walking	120
Fig. 7.20 Simulation of Archie left leg motion in forward static walking cycle.....	120
Fig. 7.21 Simulation of Archie right leg motion in forward static walking cycle.....	121
Fig. 7.22 Front view - sequence of stable forward walking of Archie (Frames A1-9)	123
Fig. 7.23 Side view - sequence of stable forward walking of Archie (Frames B1-9)	124
Fig. 7.24 Archie stable forward walking performance in 10 meters.....	126
Fig. 7.25 Archie posture based on the different W_m (find an optimised value)	127
Fig. 7.26 Ankle frontal joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$	128

Fig. 7.27 Ankle lateral joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$	128
Fig. 7.28 Knee joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$	129
Fig. 7.29 Hip lateral joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$..	129
Fig. 8.1 Proposed dynamic walking algorithm for enhancing the stability margin based on the online ZMP calculation from force/torque sensor	134

List of Tables

Table 3.1 Archie mechanical and electrical components (legend)	12
Table 3.2 Location and arrangements of Archie lower body joints	14
Table 3.3 Main characteristics of Archie	15
Table 3.4 Joint angle (Archie vs. Human)	17
Table 3.5 Harmonic drive characteristics	19
Table 3.6 Selected hall switch characteristics	20
Table 3.7 Brushed servo motors specifications	26
Table 3.8 Brushless servo motor specifications	26
Table 3.9 New Polyurethane transmission belt specifications	27
Table 3.10 Designed encoder PCB mounted pin description	29
Table 3.11 Electrical characteristics of selected encoder (Min-Max rating)	29
Table 4.1 Communication types (services) used in motor drive	36
Table 4.2 Network Management States	39
Table 4.3 Comparison of ASCII vs. Binary interpreter commands	40
Table 4.4 DLC4 values frame	41
Table 4.5 Execute command reply frame (Success/Failure)	41
Table 4.6 General USB to CAN device specifications	43
Table 5.1 Archie parametric model values	50
Table 5.2 Archie trajectory parameters values (with stride=50 cm)	53
Table 5.3 Walking parameters values	53
Table 5.4 Time boundary condition value	54
Table 5.5 NMT command structure	59
Table 5.6 MO command structure	59
Table 5.7 PA/PR command format	60
Table 5.8 BG command specifications	60
Table 5.9 BG command format (for ID no. 3)	60
Table 5.10 PT motion parameters	65
Table 6.1 Legend of the Fig. 6.4	81
Table 6.2 Final optimised value of main parameters in motion controller program (ver. 1.2)	86
Table 6.3 Final optimised joints angle in motion controller program (ver. 1.2)	87
Table 6.4 PVT table	95
Table 6.5 PVT mode definition	96
Table 6.6 PDO mapping for PVT mode before using	98
Table 7.1 Desired step length, height and hip height to perform full gait	117
Table 7.2 Module A and B (selected parameters)	121
Table 7.3 Sequence of stable forward walking test (frames specification)	121
Table 7.4 Calculated QP vector elements for each joint in static walking test (Module D)	125
Table 7.5 Different W_m value	127

Chapter 1 Problem Formulation

1.1. Problem Statement

The biped robot Archie has been constructed in Vienna University of Technology under supervision of Prof. Peter Kopacek (Byagowi, 2010). But Archie could not perform the autonomous stable walking and suffered from the motion control software that could support synchronized motion of all robot joints.

Therefore this PhD thesis presents the new motion creating system and implementation approach for a biped humanoid robot Archie, both in hardware and software levels. Biped robots need systems that control joint motions, cycle the use of legs and generate smooth desired trajectories. It is embedded with flexible communication network to transmit data between main control system and robot joint for stable walking. Additionally the software including simple graphical user interface to adjust the inputs and control parameters for desired stable walking is a must.

In this work the new motion control approach including new motion controller software is presented and tested on the robot to perform the synchronized motion between motor drives (joints) based on the desired walking patterns. By varying the values of the constraint parameters, different types of foot motion are generated to adapt to ground conditions and achieve the smooth stable walking. The walking pattern concept for proposed new motion creating system (implemented on biped humanoid robot Archie) is based on the paper (Huang et al., 2006).

The new tasks and principle goals (in hardware and software) of the proposed motion control system are listed as follows:

- Improving the hardware system based on the new designed mechanical and electrical components to perform autonomous stable walking,
- Design and establishment of the distributed communication/control network to make the real time communication between high level control system and robot hardware. It calls nodes to wait until a bus idle period is detected before attempting to transmit. If two or more nodes start to transmit at the same time, then by monitoring each bit on the bus, each node can determine if it is transmitting the highest priority message (with a numerically lower identifier) and should continue or if it should stop transmitting and wait for the next bus idle period before trying again. In this approach a simple and robust broadcast bus capable of operating at speeds of up to 1 Mbit/s is designed. The duration of each bit must be sufficient for the signal to propagate the length of the network.
- New approach to formulate the torso and ankle motion, in order to achieve the smooth stable walking by adjusting the distances along the x-axis from the hip to the ankle of the support foot at the start and end of the single-support phase, it is necessary for the robot to adapt to the ground conditions with a foot motion and maintain its stability with torso motion. If both hip trajectories and the foot trajectory are known, all joint trajectories of the biped robot will be determined by kinematic

constraints. The walking pattern can therefore be denoted uniquely by both foot trajectories and the hip trajectory.

- Design a motion mechanism based on the Position-Time (PT) approach, and specifying a sequence of absolute position with equal sampling time. In this method the PT data pulled from its source table into the buffer, therefore drive gets the PT position points and transmitted data according to the required CAN (Control Area Network) format message. In PT motion mode the drive manages a read pointer for the position points vectors ($QP[N]$), when the read pointer is N , the active motion segment starts at position $QP[N]$ and ends at $QP[N+1]$, and after control sampling times ($MP[4]$), the drive increments the read pointer to $N+1$, and reads $QP[N+2]$ to calculate the parameters of the next motion segment. PT data transferred to motor drive online, by using the proposed CAN communication network.
- Design and create a new software for proposed motion creating system with a new graphical user interface (GUI), new library motion commands and data/command monitoring system to realize the errors types and location in the robot joint. The new program (ver. 2.1) is released based on the improvement the proposed communication network, CAN data distributing, new walking pattern. The new task in developed motion control software is organised by a hierarchical control structure, in which control loops that require accurate execution cycles are implemented as real-time kernel modules, such as the motor driver loop and the walking control system. Programs and processes that operate over longer control cycles, such as speed control loop and new walking planning method, are implemented as new features in this program.
- New software implementation including the real-time message frame module, controller module and proposed walking pattern algorithm. Accordingly the new C++ source codes are implemented on the robot hardware, and the optimal forward walking trajectory calculated by adjusting the walking parameters of desired trajectory through the dozens of walking tests.

Chapter 2 Introduction

2.1 Humanoid Robot Research

The research in humanoid robots is now tending to diverge into various categories. It incorporates a wide range of areas, across different fields of study. The research on such areas as artificial intelligence, hardware and software development, realization of biped locomotion, implementation methods and interaction with the human are gaining a rapid phase of development across the rapid growth of technology. Hence, there are many theoretical and experimental studies on the biped robots.

It took 10 years of research by Honda Motors to develop P2 and more to produce ASIMO. Also they have almost certainly invested billions of US dollars in these projects (Chen et al., 2011). The research is not finished yet, only a small step has been taken along the road to the creation of a real artificial human.

There are several distinct research areas in the development of the humanoid robot. One of the major research areas today are bipedal locomotion and motion controller system. In order to simplify the control, bipedal walking robots were made without taking into account the whole body dynamics. A bipedal robot has the same bottom part (legs) but does not have body, arms and head. Also a lot of research centres started to design arms and heads separately in order to add them later to a humanoid robot.

The Japan University of Waseda has played a fundamental role in the evolution of humanoid robots. As a result, since 1967, the series of WL robots has appeared. The WL-1 robot as the first one in this series, culminated in the WABIAN-2R (Fig. 2.1) bipedal robot in 2009 (Omer & Ghorbani, 2009).

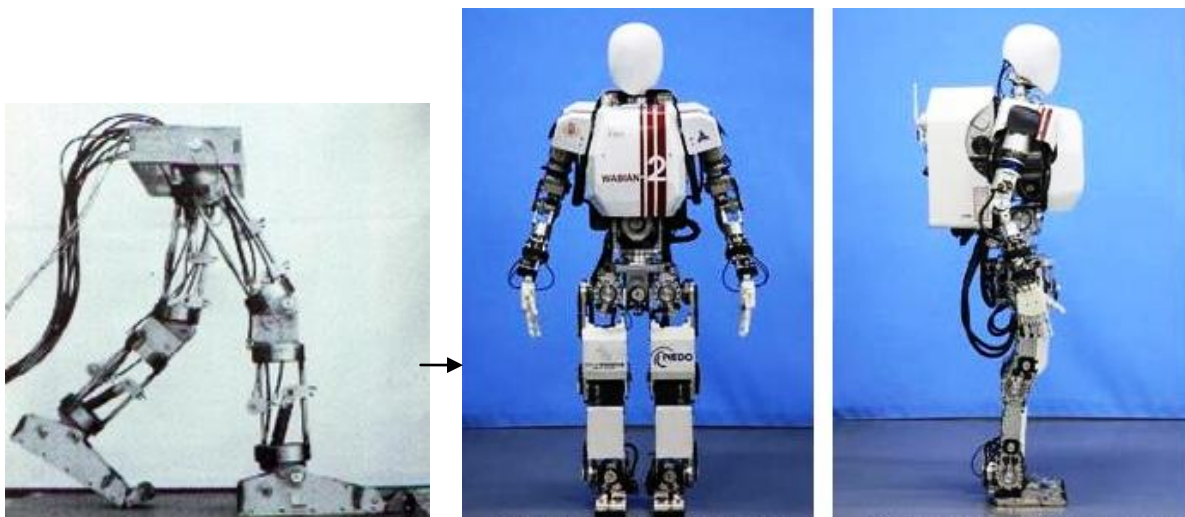


Fig. 2.1 WL-1 vs. W-2R robot

WABIAN-2R is among the most successful bipedal walking humanoid robots. In spite of the extensive research on humanoid robots, the actions of walking, running, jumping and manipulation are still difficult for robots.

In the following some of the developed humanoid robots are introduced (Fig. 2.2). Humanoids don't yet have some features of the human body. They include structures with variable flexibility, which provide safety, and redundancy of movements, i.e. more degrees of freedom and therefore wide task availability. Although these characteristics are desirable to humanoid robots, they will bring more complexity and new problems to planning and control.



Fig. 2.2 Developed humanoid robots

ASIMO

The name is an acronym for "Advanced Step in Innovative MObility". ASIMO is a humanoid robot created by Honda. Standing at 130 centimetres and weighing 54 kilograms, the robot

resembles a small astronaut wearing a backpack and can walk or run on two feet at speeds up to 6 km/h. The robot has 7 DOF (Degrees of freedom) in each arm — two joints of 3 DOF, shoulder and wrist, giving "Six degrees of freedom" and 1 DOF at the elbow; 6 DOF in each leg — 3 DOF at the crotch, 2 DOF at the ankle and 1 DOF at the knee; and 3 DOF in the neck joint. The hands have 2 DOF — 1 DOF in each thumb and 1 in each finger. This gives a total of 34 DOF in all joints (www.hondau3-x.net, 2013).

Reem-B

The 1.47-meter-high robot can walk at a relatively slow speed of 1.5 kilometers per hour, but thanks to powerful actuators in its legs and arms, Reem-B "is probably the one of the most strongest humanoid in the world since the robot can carry a 12-kilogram payload (www.pal-robotics.com, 2013).

JUSTIN

It is currently a four-wheeled robot with a head and two dexterous arms, but researchers have demonstrated a pair of legs [right] that may become its lower body. The legs use powerful yet lightweight motors to explore joint torque-based control concepts for biped balancing and walking (Guizzo, 2010).

CHARLI

Virginia Tech's Robotics & Mechanisms Laboratory- CHARLI (Cognitive Humanoid Autonomous Robot with Learning Intelligence) is the first untethered, autonomous, full-size walking humanoid robot built in the United States, according to Virginia Tech robotic Dennis Hong. He and his team are now upgrading it with custom-made linear actuators that help mimic how human limbs move. In a soccer match against Asimo, Hong's team is confident that CHARLI would prevail (Guizzo, 2010).

Surena 2

It has 45 kilograms weight and is 1.45 meter tall, has a total of 22 degrees of freedom: each leg has 6 DOF, each arm 4 DOF, and the head 2 DOF. An operator uses a remote control to make the robot walk and move its arms and head. The robot can stand on one leg and bow like ASIMO (Guizzo, 2010).

Romeo

France is set to join the select club of countries that have developed advanced adult-size humanoid robots. Paris-based Aldebaran Robotics, famed for its small humanoid robot Nao, is building a larger and more capable humanoid called Romeo. Romeo is designed to assist elderly and disabled individuals in their daily activities. The 1.4 meter-tall robot with 40 Kg weight will be able to walk through a home, fetching food from the kitchen, taking out the garbage, and acting as a loyal companion who helps entertain its owners and keep tabs on their health (Guizzo, 2010).

COMAN

The COMAN robot is 95cm tall, has 31 kg weight and has 25 DOF. Its mechanical components are made from titanium alloy, stainless steel and aluminium alloy, giving it good physical robustness. Its modular joint design uses brushless, frameless DC motors, harmonic drive gears and series elastic elements. Leg, waist and shoulder joints have a peak torque capability of 55 Nm. Custom torque sensors are integrated into every joint to enable active stiffness control and 6-DOF sensors are included at the ankle to measure ground reaction forces. COMAN can walk and balance using inertial sensors in the pelvis and chest, and its series elastic joint design makes it robust against impacts and external disturbances. From the kinematic perspective the new lower body includes the lower torso (housing the 3 DOF waist module), and the two leg assemblies with 6 DOF each. The height of the COMAN lower body, from the foot to the waist, is 671 mm, with a maximum width and depth (at the hips) of 176 mm and 110 mm, respectively. The total lower body weight is 17.3 kg, with each leg weighing approximately 5.9 kg, and the waist section, including the hip flexion motors, weighing 5.5 kg (Dallali et al., 2013).

ATLAS

Atlas is based on Boston Dynamics' earlier PETMAN humanoid robot, and has four hydraulically-actuated limbs. Constructed of aircraft-grade aluminium and titanium, it stands approximately 1.8 m tall and weighs 150 kg, and is illuminated with blue LEDs. Atlas is equipped with two vision systems – a laser rangefinder and stereo cameras, both controlled by an onboard computer and has hands with fine motor skill capabilities. Its limbs possess a total of 28 degrees of freedom. Atlas can navigate rough terrain and climb independently using its arms and legs, although the 2013 prototype version was tethered to an outside power supply to maintain stability. In October 2013 Boston Dynamics uploaded a video showing Atlas could withstand being hit by projectiles and balance on one leg (www.news.cnet.com, 2013).

2.2 Biped Walking (Basic Functions)

In order to provide the basic functions required for walking, each stride involves an ever-changing alignment between the body and the supporting foot during stance and selective advancement of the limb segments in swing. These reactions result in a series of motion patterns performed by the hip, knee and ankle. Early in the development of gait analysis the investigators recognized that each pattern of motion related to a different functional demand and designated them as the phases of gait. Further experience in correlating the data has progressively expanded the number of gait phases identified. It now is evident that each stride contains eight functional patterns. Technically these are sub phases, as the basic divisions of the gait cycle are stance and swing, but common practice also calls the functional intervals phases (Vaughan, 2003).

In the past it has been the custom to use normal events as the critical actions separating the phases. While this practice proved appropriate for the amputee, it often failed to accommodate the gait deviations of patients impaired by paralysis or arthritis. For example,

the onset of stance never contact the ground or do so much later in the gait cycle. Similarly initial floor contact may be by the whole foot (foot flat), rather than having forefoot contact occur later, after a period of heel-only support. To avoid these difficulties and other areas of confusion, the Rancho Los Amigos gait analysis committee developed a generic terminology for the functional phases of gait (Vukobratovic, 1969).

Analysis of a person's walking pattern by phases more directly identifies the functional significance of the different motions occurring at the individual joints. The phases of gait also provide a means for correlating the simultaneous actions of the individual joints into patterns of total limb function. This is a particularly important approach for interpreting the functional effects of disability. The relative significance of one joint's motion compared to the other's varies among the gait phases. Also, a posture that is appropriate in one gait phase would signify dysfunction at another point in the stride, because the functional need has changed. As a result, both timing and joint angle are very significant. This latter fact adds to the complexities of gait analysis (Goswami, 1999).

Each of the eight gait phases has a functional objective and a critical pattern of selective synergistic motion to accomplish this goal. The sequential combination of the phases also enables the limb to accomplish three basic tasks. These are weight acceptance (WA), single limb support (SLS) and limb advancement (LA). Weight acceptance begins the stance period and uses the first two gait phases (initial contact and loading response). Single limb support continues stance with the next two phases of gait (mid stance and terminal stance). Limb advancement begins in the final phase of stance (pre-swing) and then continues through the three phases of swing (initial swing, mid swing and terminal swing).

2.2.1 Bipedal Motion System

Biped robots have potential ability to elevate mobility of robotic system and they attract general attention in the last decade. Due to their form, it is easy to introduce them into our living space without preparing special infrastructure. Recently, several autonomous biped humanoid robots have been developed. In 1996, the first autonomous bipedal humanoid robot with battery power supply was developed by Honda Motor Co., Ltd. (Hirai et al., 1998). Takanishi and his co-workers at Waseda University built a human-size bipedal humanoid robot and proposed a basic control method of whole body cooperative dynamic biped walking (Yamaguchi et al., 1999). Kaneko and his colleagues at the National Institute of Advanced Industrial Science and Technology (AIST) also developed a bipedal humanoid robot of 1.54 m height and 58 kg weight mainly for the purpose of investigating fundamental techniques and applications (Harada et al., 2004). Same group also proposed a method of a biped walking pattern generation by using a preview control of the zero moment point (ZMP) (Kajita et al., 2003). The ZMP and its extension (FRI; foot-rotation indicator) are basic stability criteria for biped robots walking (Vukobratovic & Borovac, 2001). Nishiwaki and his co-researchers at University of Tokyo studied a humanoid walking control system that generates body trajectories to follow a given desired motion online (Xiao et al., 2006). Loffler and his colleagues at Technical University of Munich also designed a biped robot to achieve a dynamically stable gait pattern (Loffler et al., 2003).

Biped robot have better mobility than the conventional wheeled robots, but they tend to tip over easily. To be able to walk stably in various environments, such as on rough terrain, up and down sloped, or in regions containing obstacles, it is necessary for the robot to adapt to the ground conditions with a foot motion and maintain its stability with a torso motion. The Huang and Kajita in 2001 introduced a method for planning walking pattern for a biped robot and in 2006 they confirmed the effectiveness of their proposed method by simulation and experiments with their developed robot BHR-02 with 32 DOF.

2.2.2 Static Stability in Humanoid Robot

A statically stable robot is well balanced and does not fall over when standing. This means that the center of gravity of a robot is within its ground contact base. Let us take an example of a robot with three legs arranged in the form of a triangle. This robot does not require any kind of movement to stand stable (Fig. 2.3) and can stand balanced as long as the center of mass is within the triangle. This triangle is called the "support polygon" which is a horizontal region over which the center of mass lies to achieve static stability.

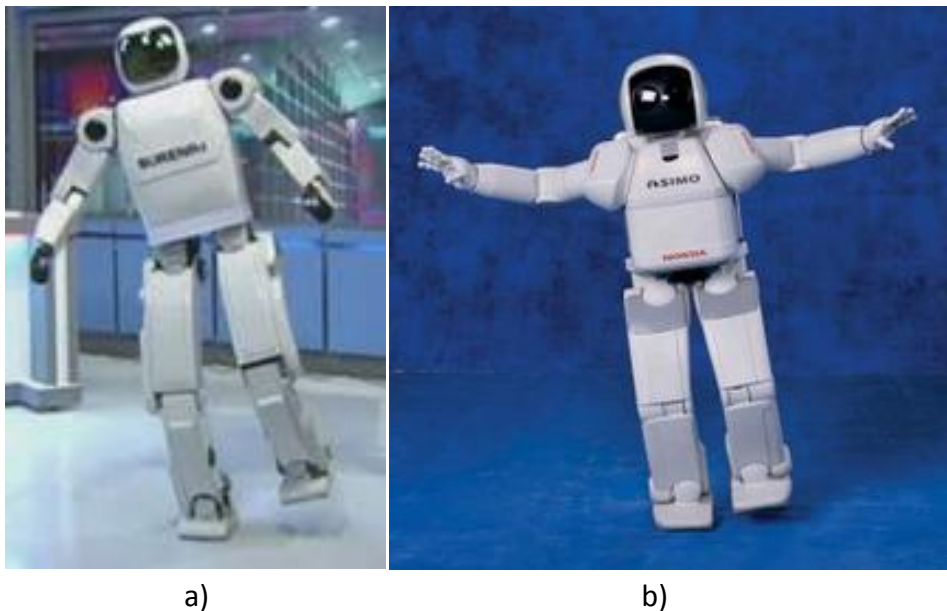


Fig. 2.3 a) Surena II vs. b) ASIMO standing on one leg (Guizzo, 2010)

If the previous statement is confusing, just understand that support polygon is a projection between all the support points of a robot onto the surface it is standing. The minimum number of ground contact points required for a statically stable robot is three. However, achieving static stability with two legs is not an easy task. Humans although may seem, are not statically stable. Our muscles and nerves control our balance and make it seem an effortless stability. This is the same reason why a baby takes a year or two to learn how to stand.

Since bipedal robots took their first steps, the majority has been designed with the same basic joint/actuator configuration in their legs. This design, based on a simplified human leg, uses

just six motors (three for the hip, one for the knee, and two for the ankle), and though it proved successful, it has also shown several limitations over the last 25 years (www.spectrum.ieee.org, 2013).

2.3 Thesis Approach

The work presented in this PhD thesis has been conducted in 4 parts. The first part involved developing the hardware structure that could potentially influence on the stable walking performance of the robot. This step was accomplished by adding 2 new joints, parts and components. The second part involved design a new motion creating system in order to control the robot joint motion by trajectory constraint formulation, and new pattern generation calculation. In the third step the new created motion controller software, programmed the new ideas and proposed strategies which already described in second part. In the last part the new motion controller software, proposed walking method and new motion creating system are implemented and dozens times tested on actual robot to achieve the optimised input values for stable walking.

2.4 Chapter Organization

In **chapter 3** we introduced the hardware structure (mechanical, electrical), and latest developed parts such as new designed and manufactured ankle joint including new joint, connector parts, harmonic drive, ball bearing, magnetic rotary encoder (PCB mounted), new brushless motor, hall switch, Elmo motor drive, new foot plates, new powerful timing belts in hip and new power supply source. In order to let robot move properly and avoid any damages the joint limit angle are calculated and compared to human.

In **chapter 4** the proposed distributed communication/control network for fast calculation, fast communication and real-time data exchange is well described.

Chapter 5 discussed about the new walking pattern generation method for Archie. In this method the new approach to formulate the constraint trajectories for torso and ankle of Archie are presented. On the other hand we designed new motion creating system mechanism based on the Position-Time (PT) and Position-Velocity-Time (PVT) motion. They specifies the PT, PVT tabulated motion and define the starting index in the QP[N] arrays. In PT motion, a new position reference value is selected from the QP[N] array once per MP[4] position controller sampling times. But in PVT motion new position and speed reference values are picked from the QP[N] and QV[N] arrays at the time specified by the elements of the QT[N] array. The QP[N], QV[N] and QT[N] arrays store and define the position, speed and any single time instance.

In **chapter 6** the new motion controller program (software) is introduced. The advantages and capabilities are presented and compared to previous programs. The new software allows to manage the motion for fully synchronized joints movement, containing the new motion command library and new graphical interface.

The new software implementation method and static walking test results of proposed motion creating system are presented in **chapter7**. The implementation structure and relative modules to perform a stable static walking of Archie are carefully described.

Chapter 8 discussed about the conclusion, summary and proposed future works.

Appendix A: a) Hip and Foot trajectory calculation, b) Foot motion in x and z axis, c) Drive commands

Appendix B: Developed and New Software source codes summary

a) Joint creation (Archie.cpp file), b) Inverse kinematic calculation codes (foot-trajectory.cpp), c) optimised value defined in module A and B, d) All joints angle for stable static walking of robot.

Chapter 3 Hardware Design and Developments of the Biped Humanoid Robot Archie

3.1 Introduction

The mode of locomotion for humanoids is biped dynamic walk that is inherently unstable (Shivaraj & Lasitha, 2012). In order to make the biped locomotion stable and perform human-like activities, robot hardware, software (motion controller program) and walking control strategies are fundamentally important. For example, all and more of the following factors need to be considered in designing a humanoid body: limited torque of actuators at joints in the robot, rigidity of hardware such as main frames, hip, arm and legs, impulsive loading at landing of legs that can damage hardware, interference among links, weight and ease of handling.

On the other hand, the motion controller/program must have the following characteristics: stability in the sense of continued smooth walk, practicability with starting, stopping, low-energy consumption, versatility in choosing landing position of the swinging leg which is one of most notable advantages of biped locomotion.

In IHRT we established and developed hardware design concept, new motion creating system (software, hardware), and control strategies for our biped humanoid robots Archie that can solve these problems and could truly perform human-like walking. This chapter presents important physical considerations and hardware improvements in design and manufacturing to finally achieve the smooth walking of a robot.

3.2 Hardware Description

Biped robot design is different from conventional robots since, there are restrictions and differences in the amount, type and size, response time of the actuators, sensors, parts-weight and even configuration, position, and distribution of the biped's robot structure. Thus, the robot's hardware design has a huge influence in order to obtain a reliable biped robot. For that reason, mass distributions, COM (Center of Mass) location and the actuators selection are important stage on the mechanical structure design and have a direct impact on the robot's performance (Bachar, 2004).







A biped robot design also requires to withstand the mechanical stress imposed during experimentation. The electronic design must mainly assure be fast enough to perform the control algorithms, handle encoders and sensors, and high intercommunication to a PC (Kim et al., 2005).

This section describes and highlights hardware structure and the improvements of the biped humanoid robot Archie. The goal of the project is to create a humanoid robot that is able to support human in a broad variety of different tasks alone or in cooperation with human.

In general, the basic design concepts and characteristics of the Archie biped humanoid are considered as follows:

- The robot's hardware design add less weight as possible, therefore high rigidity and lightweight have been realized by aluminium alloy (EN AW 5083), as the major structural material in biped robot Archie.
- Having sufficient joints (Degree of Freedom) and range of joint motion, torque, speed in order to achieve smooth walking.
- Low power consumption, low cost of production.
- Main controller (PC), Power switch, CAN converter, battery and servo controllers/drivers for hip are located in torso and chest (Fig. 3.1, Table 3.1). The mass except actuators is concentrated on torso, since we need to reduce the load which is inflicted to the actuators in frequently moving parts.

Table 3.1 Archie mechanical and electrical components (legend)

	Lower body joint drive (including drive, PCB and harmonic drive)
	Designed upper body joint
	Motion Control Program
	Motor + magnetic encoder
	CAN Cable
	Power supply (36V-28A)

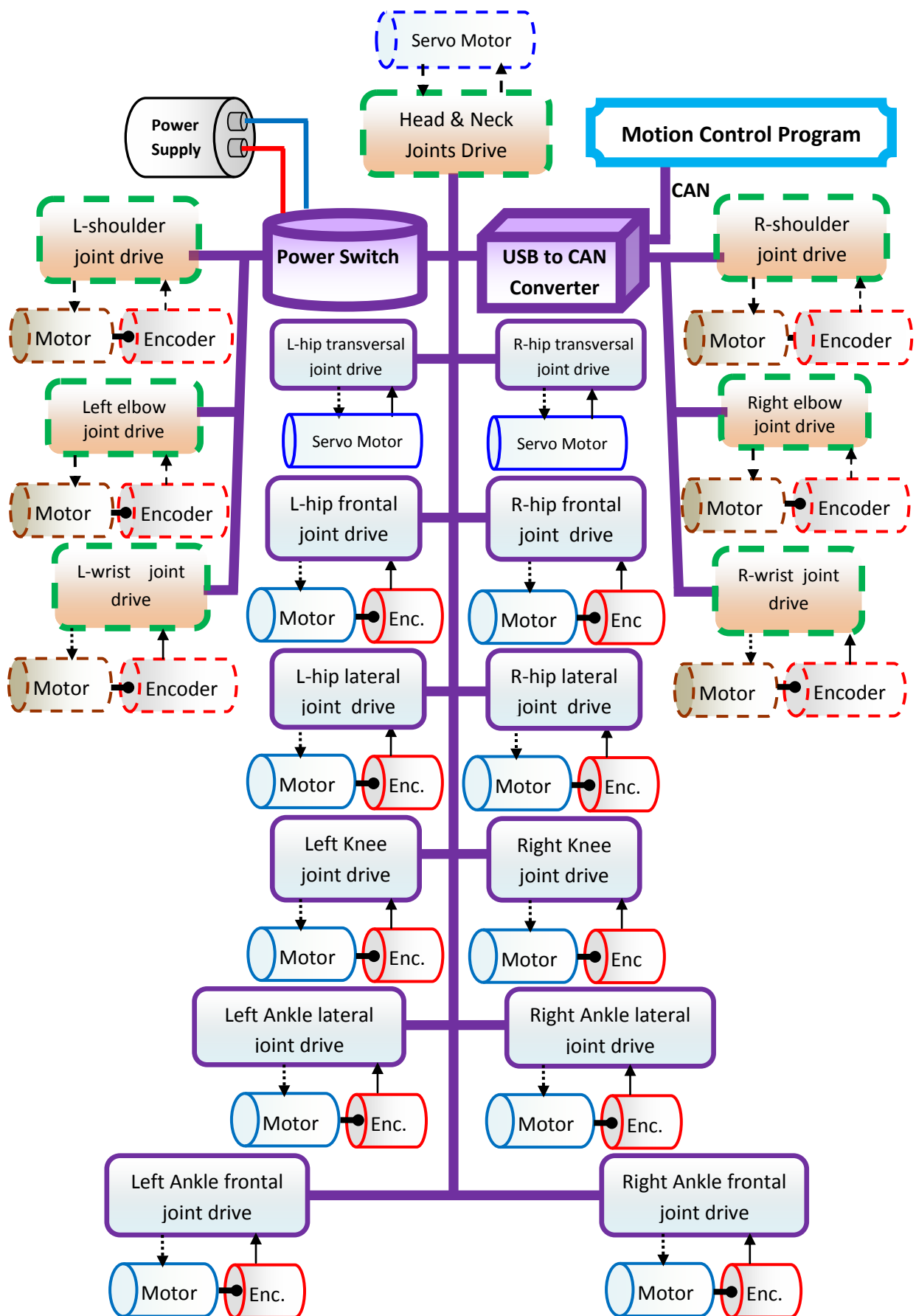


Fig. 3.1 Archie hardware (updated structure from (Byagowi, 2010))

The designed upper body consists of the following modules: head and neck joint, two arms with shoulder, elbow, forearm and wrist, thorax and torso joint. Humanoid robot Archie is designed in CATIA (Fig. 3.2).

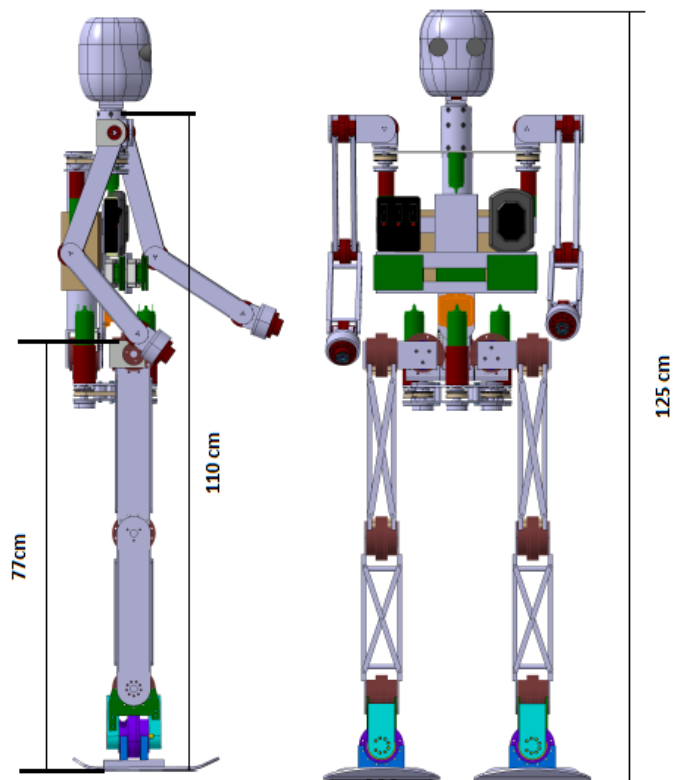


Fig. 3.2 CAD model of Archie

3.2.1 Mechanical Design of Robot

Besides providing the general support to the main passive and active subsystems of the humanoid, imitating the role of the human skeleton, it comprises a set of simple kinematics joints mimicking the most important degrees of freedom of the human body. On the other hand robot must show good motion performance relative to its height to weight ratio or body mass index.

Table 3.2 Location and arrangements of Archie lower body joints

Location	No. of joints (DOF)
Hip	3
Knee	1
Ankle	2
Total/Leg	6

Archie is designed to have 31 degrees of freedom in total but at the moment only the lower body is realized with 12 DOF, it has 6 degrees of freedom for each leg that includes 3 DOF in hip and 1 DOF in knee, and 2 DOF in ankle (Table 3.2). The platform is thought to be of human size, modular and to perform a natural looking straight gait.

Table 3.3 gives the main characteristics of the Archie biped humanoid. The hardware structure of the robot with a careful and well discipline between form, function, power, weight, cost and manufacturability is designed (Dezfouli et al., 2011).

Table 3.3 Main characteristics of Archie

Designed Degree of Freedom	31
Realized Body	Lower Body (12 DOF)
Height	125 cm
Weight	~ 20 kg
Sensor	Magnetic Rotary Encoder
Actuator	Brushless and Brushed DC motor + Harmonic drive gear
Control Program	Motion Controller Program (ver. 2.1)
Power (Battery)	2x 24V (Li-ion) / 6.1 Ah
Main Power Supply (External)	36V/28A (COSEL ADA1000F-36)
Operating Section	Laptop/PC, PC ARTIGO 1000
Operating System (OS)	Linux
Communication Unit	CAN bus protocol, USB to CAN converter

According to above concepts, teen sized humanoid robot Archie is developed so which resembles a human shape and has sufficient joints to imitate a human motion such as walking as illustrated in Fig. 3.3.

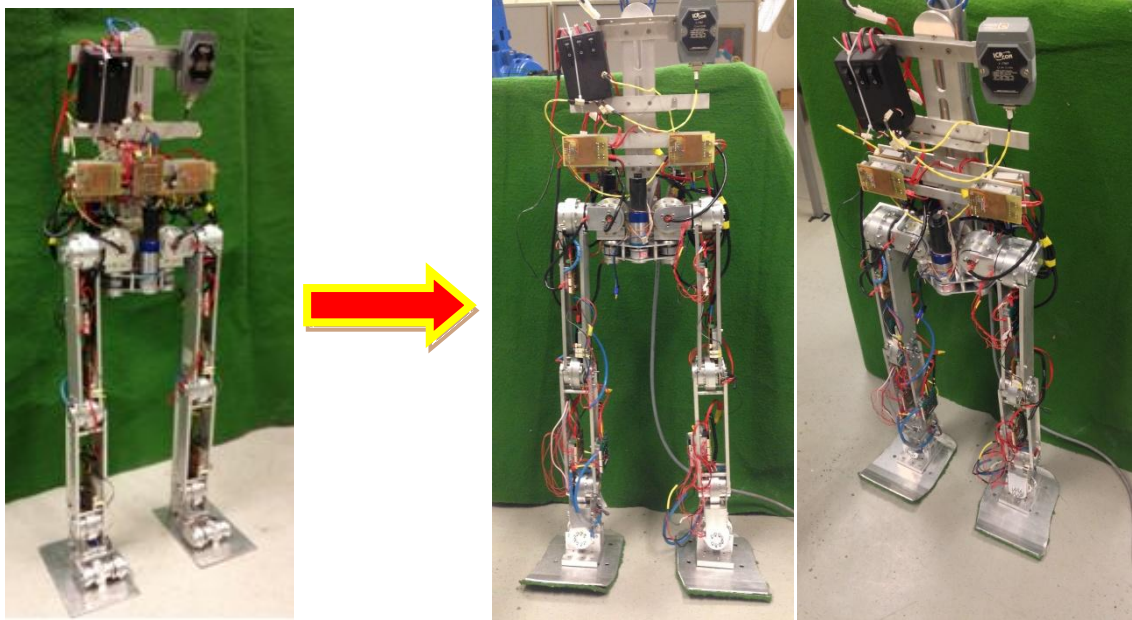


Fig. 3.3 Improved biped humanoid robot Archie

3.3 Performance Management in Archie Design

The actuator's size and weight are restricted in developing a biped humanoid robot. Therefore, the power of actuators, maximum rotation angle because of the hardware design, the pick-torque, and pick-velocity of joints are limited. When the stability constraint and the ground conditions are satisfied, it is additionally desirable to select a walking pattern that requires small torque and velocity of the joint actuators. To reach this goal, it is necessary to clarify the relationship between the actuator specifications and the walking patterns.

This relationship is also important to select suitable actuators and speed reduction devices such as gears and pulley-belts when designing the actual biped robot (Takanishi et al., 2007).

The design of a biped robot is very important for the achievable performance of the robot, especially the weight of the system imposes physical limits.

Additional weight deteriorates the walking performance, as a higher mass has to be accelerated; the maximum achievable joint acceleration decreases with rising weight. Hence motor-harmonic drive (joint) should be chosen as light as possible, thus just about fulfilling the requirements on possible torque, acceleration and velocity. Besides the increased weight of more powerful motors, they usually also consume more power and therefore require larger and more batteries. Furthermore, the electronics to handle higher currents become heavier and bulkier.

Human body mechanism basically comprises bones as rigid links, cartilages as joints, muscles and tendons that actuate each part of the body. It is impossible to replace all of this muscular-skeletal system by current mechanical components. Therefore, the primary goal of the mechanical design in Archie is considered as development of a robot that can imitate equivalent human motion. To generate a joint with more DOF the necessary number of motor/gear combination are combined. There are various design considerations and factors when Archie is designed and developed which briefly are described in the following.

3.3.1 Degrees of Freedom (DOF) Selection

It is specified that the robot be able to move in 3 dimensions (3D). That means that the robot must be physically capable of changing walking directions, walking up stairs, and others similar tasks. To be able to do these tasks, the robot needs to have sufficient DOFs. Assume if the robot should only move legs forwards and backwards (no sideways movements), in this case it would not be able to change walking direction. Fig. 3.4 shows the location and the arrangements of the joints on the robot.

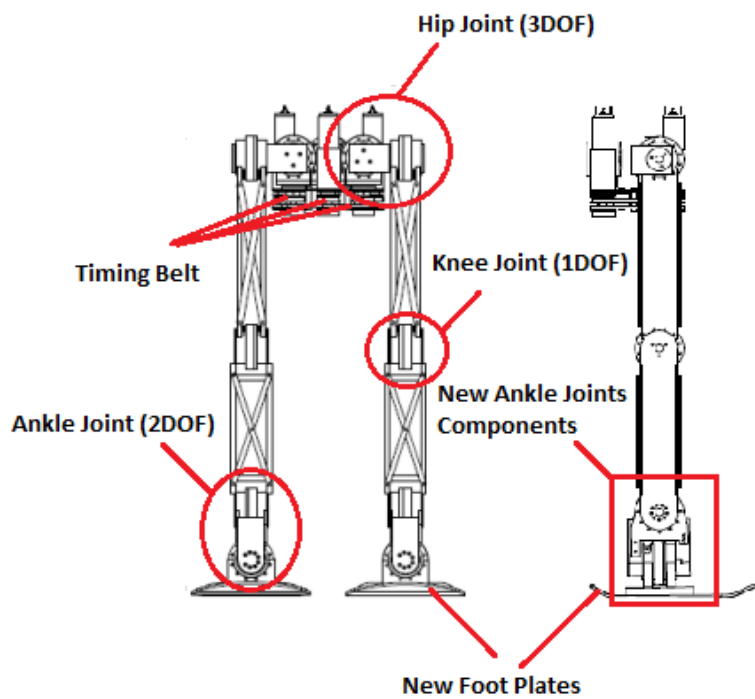


Fig. 3.4 Designed Archie lower body

The range of joint motion of the robot is significantly less than that commonly seen in human motion. Therefore, scaling the motion appropriately to bring it within the joint angle limits of the robot is particularly important for preserving the style of the motion.

Table 3.4 Joint angle (Archie vs. Human)

Joints	Biped Archie		Human	
	Min. Angle (degree)	Max. Angle (degree)	Min. Angle (degree)	Max. Angle (degree)
Hip (Pitch)	-11	45	-10	135
Hip (Roll)	-20	35	-25	40
Hip (Yaw)	-25	70	-20	65
Knee	-70	0	-135	5
Ankle (Pitch)	-50	50	-50	30
Ankle (Roll)	-50	50	-60	30

Each joint trajectory of the motion should lie within the joint limits angle of the robot while performing the simple motion. Each joint motion limitation in Archie versus the human is presented in Table 3.4.

3.3.2 Actuator/Sensor Selection

Perhaps the most significant decision to be made when constructing a biped robot, is the choice of actuator. Giving the robot its freedom of movement, the role of the actuator is to mimic the functionality of muscles and tendons. Many widely varying devices and techniques can be employed to produce controllable motion. The selection of a suitable type, or types, of actuator will depend on a design constraints, including such issues as size, weight, performance, strength, accuracy of control, and of course cost. In this section, two basic robot joint components are discussed: actuators and sensors. In the following, the features of Archie actuating system are presented in terms of the power supply, power amplifier, servomotor (brushed and brushless) and transmission (harmonic drive).

3.3.3 Joint Actuator Mechanism

Joint actuator must be strong enough to carry out the robot's weight and maintain a good relationship between its weight and torque. The motion imposed to Archie joint is realized by an actuating system which in general consists of:

a) Transmissions

The execution of joint motions of a manipulator demands low speeds with high torques. In general, such requirements do not allow an effective use of the mechanical features of servomotors, which typically provide high speeds with low torques in optimal operating conditions. It is then necessary to interpose a transmission (gear) to optimize the transfer of mechanical power from the motor to the joint. During this transfer, the power is dissipated as a result of friction.

b) Harmonic Drive

Gear selection as one of the main part of joint design involved with major importance, since the selected DC motors operate at high speeds and low torques. This suitable gear for Archie posses the following qualities:

- High gear ratio
- High efficiency
- Zero backlash (or low)
- Low weight

The develop gears that fulfils the specific requirements for the robot harmonic drive is a special type of mechanical gear system that can improve certain characteristics compared to traditional gearing systems (such as helical gears or planetary gears). The strain wave gearing

(harmonic drive) is based on elastic dynamics and utilizes the flexibility of metal. The Archie harmonic drive gear is an assembly of three main parts; an outer ring named circular spline, a metal cup named flexspline and an elliptic steel disk named the wave generator, which is nearly a very difficult task to assemble them (Table 3.5).

Table 3.5 Harmonic drive characteristics

Series	Size	Ratio	Version	Flexspline
CSD	20 mm	160	2A-GR	<ul style="list-style-type: none"> •Standard •Flexspline with enlarged central bore diameter

c) Power Amplifiers

The current and voltage of motors supplied with batteries are very limited. The power amplifier has the task of modulating, under the action of a control signal, the power flow which is provided by the primary supply and has to be delivered to the motor drive of Archie for the execution of the desired motion. In other words, the amplifier takes a fraction of the power available at the source which is proportional to the control signal; then it transmits this power to the motor in terms of suitable force and flow quantities. The value of voltage for permanent-magnet DC servomotors or the values of voltage and frequency for brushless DC servomotors are determined by the control signal of the amplifier, so as to make the motor executes the desired motion.

d) Low Power Hall Effect Switch

A three-terminal Hall effect sensor device with an output driver, mainly designed for battery operation hand-held equipment, and total operation power is down to 15μW in the 2.75 V supply. The south pole of sufficient strength will turn the output on in SIP-3L but the north pole of sufficient strength will turn output on. The output will be turned off under no magnetic field.

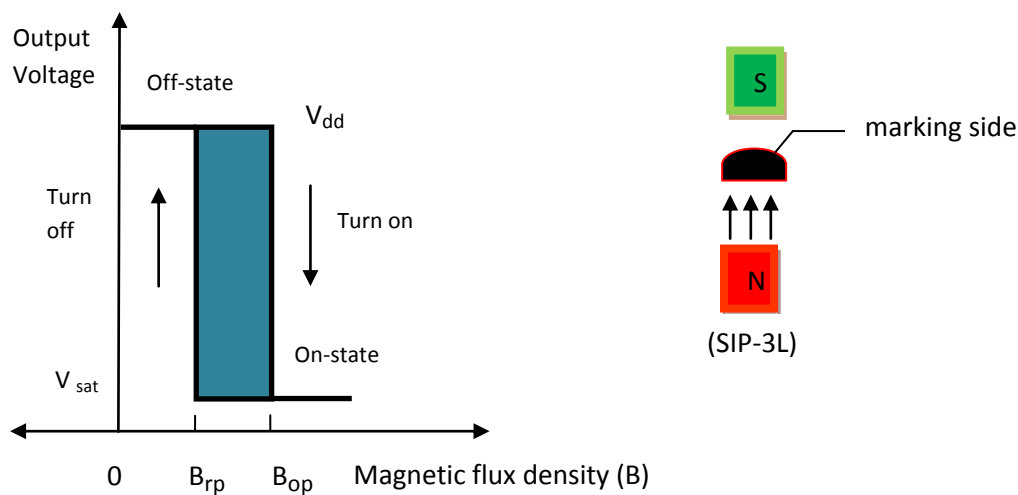


Fig. 3.5 Operating characteristics of hall sensor/switch (www.diodes.com, 2013)

While the magnetic flux density (B) is larger than operation point (B_{op}), the output will be turned on (low), the output is held until B is lower than the release point (B_{rp}), then turned off (Fig. 3.5 and Table 3.6).

Table 3.6 Selected hall switch characteristics

Symbol	Parameter	Rating	Unit
V_{dd}	Supply Voltage	7	V
B	Magnetic Flux Density	Unlimited	
I_{out}	Output current	10	mA
P_D	Power Description	550	mW
$T_{J(MAX)}$	Maximum Junction Temperature	150	°C
T_{ST}	Storage Temperature Range	-65 to 150	°C

e) Power Supply

The task of the power supply is to supply the primary power to the amplifier which is needed for operation of the actuating system. In the case of electric servomotors, the power supply consists of a transformer and a typically uncontrolled bridge rectifier. These allow the alternating voltage available from the distribution to be converted into a direct voltage of suitable magnitude which is required to feed the power amplifier.

f) Motor Performance Management

Actuation of joint motion is entrusted to motors which allow the realization of a desired motion for the mechanical system. A portion of the input power is converted to output as mechanical power, and the rest is dissipated because of mechanical, electric loss. For the typical performance required, Archie motors have the following requirements with respect to those employed in conventional applications:

- low inertia and high power-to-weight ratio,
- possibility of overload and delivery of impulse torques,
- capability to develop high accelerations,
- wide velocity range (from 1 to 1000 revolute/min),
- high positioning accuracy,
- low torque ripple so as to guarantee continuous rotation even at low speed.

These requirements are enhanced by the good trajectory tracking and positioning accuracy demanded for an actuating system for Archie, and thus the motor must play the role of a servomotor. The most employed motors in robotics applications are electric servomotors, therefore brushed and brushless servomotors are considered as actuators in Archie joints.

The main reason for using a brushless DC motor in Archie is to eliminate the problems due to mechanical commutation of the brushes in a permanent-magnet DC motor (Dezfouli & Kopacek, 2011). In fact, the presence of the commutator limits the performance of a

permanent-magnet DC motor, since this provokes electric loss due to voltage drops at the contact between the brushes and plates, and mechanical loss due to friction and arcing during commutation from one plate to the next one caused by the inductance of the winding. The elimination of the causes provoking such inconveniences, i.e., the brushes and plates, allows an improvement of motor performance in terms of higher speeds and less material wear.

The inversion between the functions of stator and rotor leads to further advantages. The presence of a winding on the stator instead of the rotor facilitates heat disposal. The absence of a rotor winding, together with the possibility of using rare-earth permanent magnets, allows construction of more compact rotors which are, in turn, characterized by a low moment of inertia. Therefore, the size of a brushless DC motor is smaller than that of a permanent-magnet DC motor of the same power; an improvement of dynamic performance can also be obtained by using a brushless DC motor. For the choice of the most suitable servomotor for a specific application, the cost factor plays a relevant role. Not uncommon are also stepper motors. These actuators are controlled by suitable excitation sequences and their operating principle does not require measurement of motor shaft angular position. The dynamic behaviour of stepper motors is greatly influenced by payload, though. Also, they induce vibration of the mechanical structure of the manipulator. Such inconveniences confine the use of stepper motors to the field of micromanipulators, for which low-cost implementation prevails over the need for high dynamic performance. In this respect, electric servomotors present the following advantages:

- widespread availability of power supply,
- low cost and wide range of products,
- high power conversion efficiency,
- easy maintenance,
- no pollution of working environment.

3.4 Motor Drive Controller

The designed motion controller is a PCB (printed circuit board) mounted device that enables efficient and cost saving implementation (Fig. 3.6). Each drive controller is mounted on a PCB board which allows the connection of each driver controller with the others elements of the joint motion system. These other elements include: a hall sensor, magnetic rotary encoder, CAN bus connections, power supply and a DC motor (Dezfouli & Mohamadi Daniali, 2012).

3.4.1 Digital Servo Drive

The digital servo driver operates from a DC power source in current, velocity and position modes, and also in conjunction with a permanent-magnet synchronous brushless motor or DC brushed motor, depending the type of joint, therefore all the motor drives include fully digital motion controller (hardware/software) that features current, velocity and position loops and a wide range of commutation types and position feedback (chapter 5).

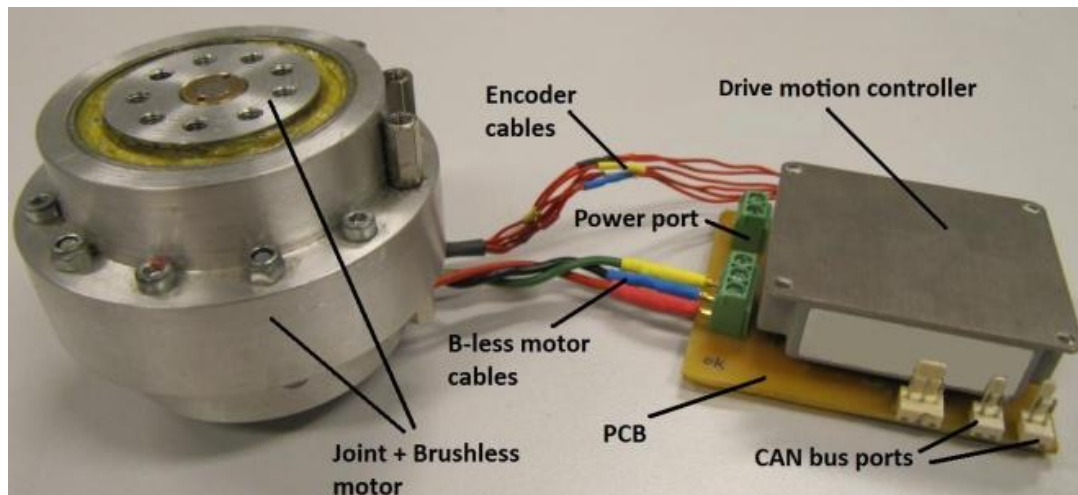


Fig. 3.6 Drive motion controller connected to modular joint (Dezfouli & Mohamadi Daniali, 2012)

The kind of Elmo motion controller which is used in Archie is Whistle. The Whistle is a series of intelligent miniature digital servo drives for the current DC and brushless motors of Archie. The matchbox-sized servo drive weights only 50 g and supports up to 20 amps continuous current. Its high density allows the drive to deliver a peak of 3200 W of power and 1600 W of continuous power (www.elmomc.com, 2013).

3.5 New Hardware Improvements

it is necessary and significant to improve hardware structure to prepare the qualified platform for software implementation and perform the robot walking (chapter 7). Hence, the lack of existing frontal ankle joint which lets robot to stand on one leg and improve the swing phase of walking, and on the other hand weak timing belts in hip, absence of powerful motor in foot joints, deficiency of the high quality PCB and cables caused to design and produce new parts and components to tackle these problems.

3.5.1 New Ankle Joint Design and Manufacturing

One of the main challenges in the development of Archie biped humanoid robot is designing joints for the ankle joint which can produce sufficient torque to handle the required static and dynamic loads (Fig. 3.7). This is due on the one hand to the fact that the ankle is the joint that needs the most torque in the locomotive apparatus and, on the other hand, because of the constraints of size and weight.

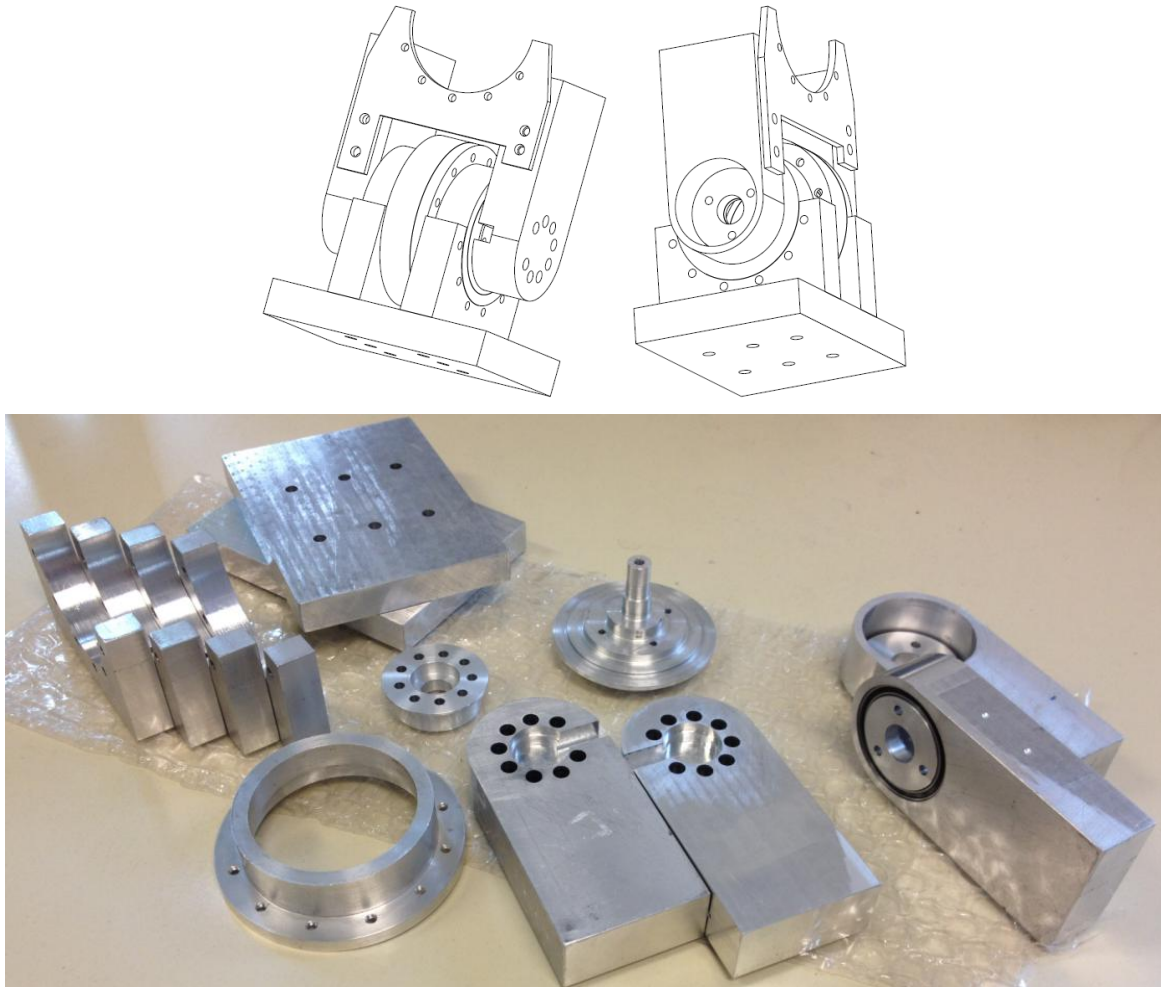


Fig. 3.7 New designed ankle joint parts

The load carrying construction of the ankles consists of 2 plates bracket bolted together and a joint set connected to the foot plate and 2 front and back holder which are attached to the upper part of foot (lateral-ankle joint). The ankles comprise of two actuated axes; roll and pitch, which runs by distance. The front and rear U plates are mounted by screws in the connection to the bottom plate and joint casing holes. In this manner when assembling, it can be positioned to negate potential effects of misalignment relative to the joint holes.

Two front and rear connectors are designed to attach the frontal joint to upper lateral joint. The encoder (PCB mounted) is embedded in frontal connector. Although, mirrored right/left ankles are made from identical components, which is possible since the rear connector is reversible and thereby allows mounting of the motor on both sides of the bracket. All above components are screwed to a foot plate which is shaped similar to a human foot (Fig. 3.8).

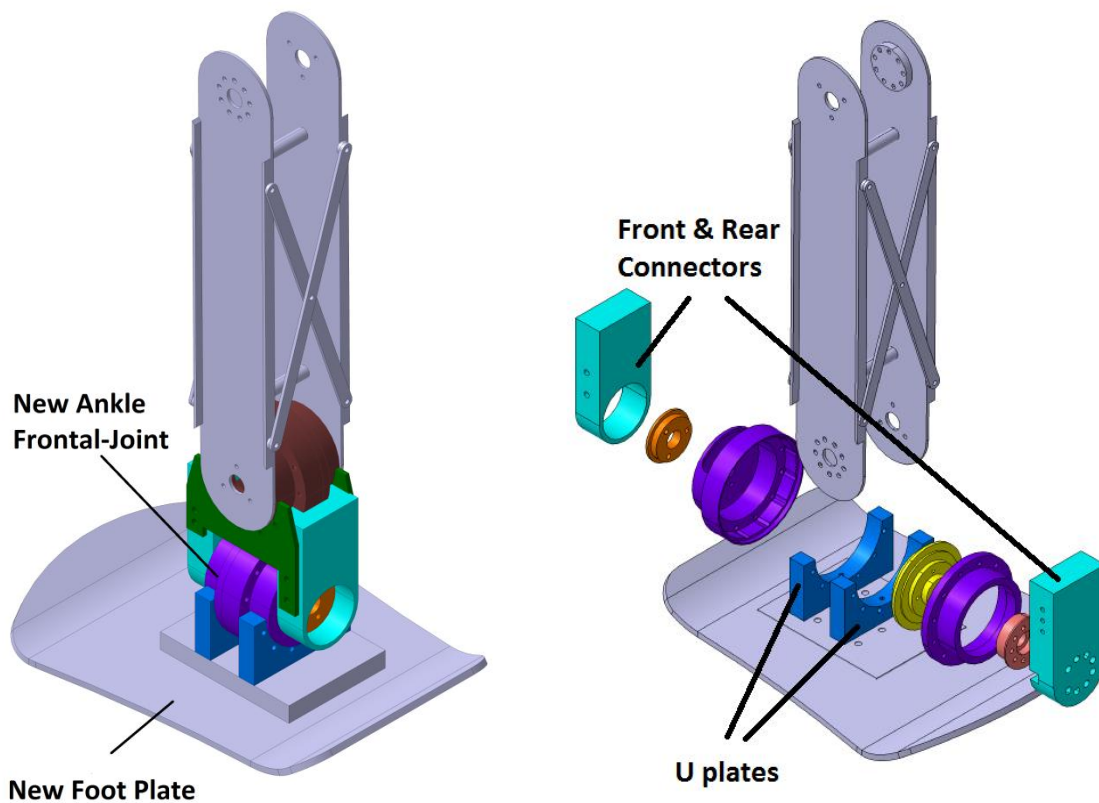


Fig. 3.8 New ankle joint parts and components (assembled to new foot plate and leg)

The manufactured parts of the new ankle joints, assembled component such as harmonic drive and attached new designed PCB are presented in following pictures (Fig. 3.9 and 3.10):

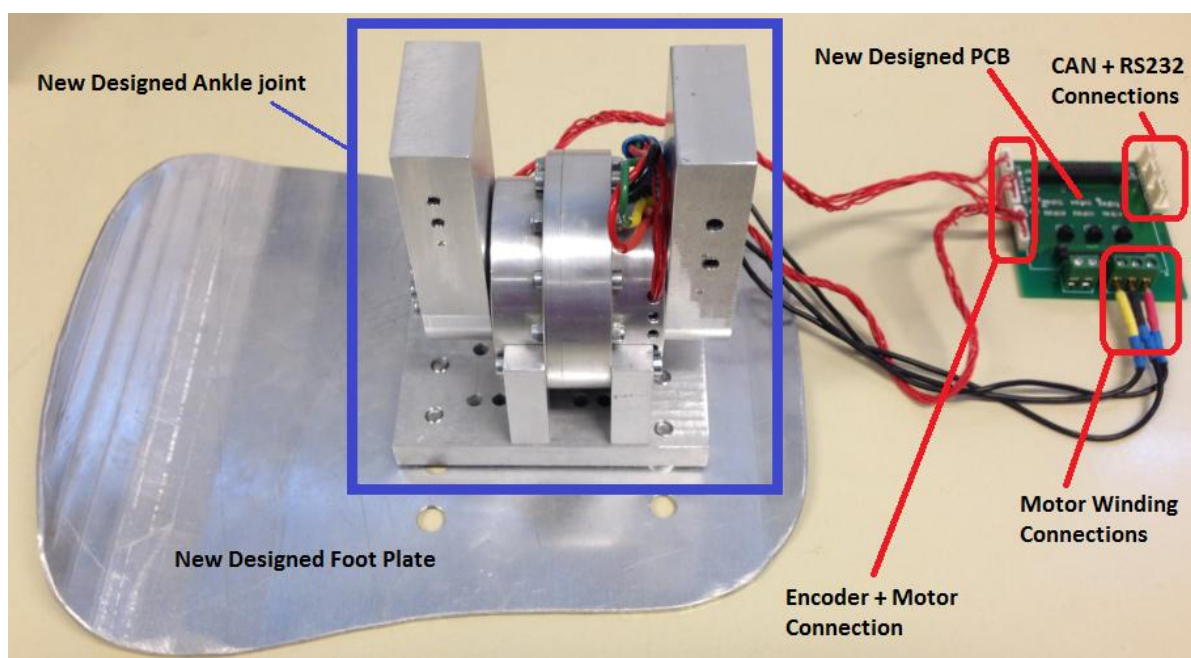


Fig. 3.9 Manufactured and assembled parts of new ankle joint + new PCB connections

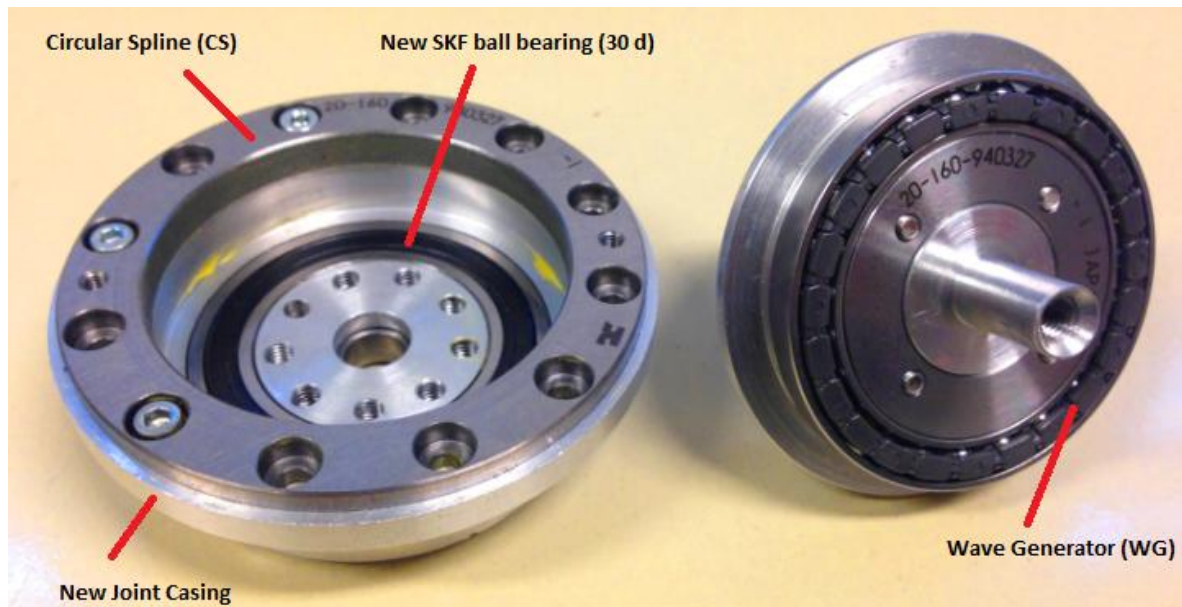


Fig. 3.10 Assembled harmonic drive to new joint components

3.5.2 Motor Type Selection for New Ankle Joint

The main requirements that must be considered before proceeding to Archie motor selection is listed as follows:

- Rated torque (continuous torque)
- Combination with selected gearhead
- max. permissible speed
- maximum torque

Mostly the drive is indirect, this means that there is a mechanical transformation of the motor output power using belts, gears and screws like the servo DC brushed motor selected in hip transversal joint. The drive parameters, therefore, are to be calculated to the motor shaft. Furthermore, the power supply requirements need to be checked.

The possible motor types are selected using the required torque (Fig. 3.11). On the one hand, the peak torque (M_{\max}), is to be taken into consideration and on the other, the effective torque (M_{RMS}). Continuous operation is characterized by a single operating point (M_B, n_B).

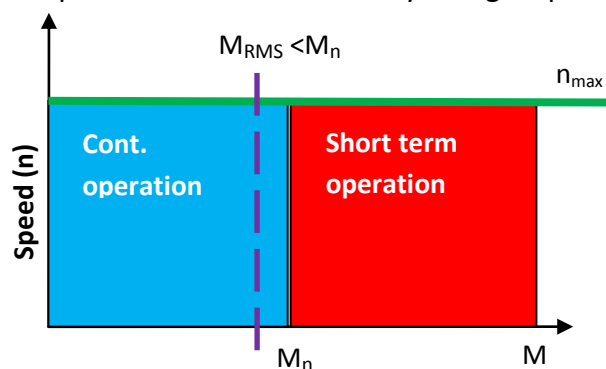


Fig. 3.11 Motor type selection criteria (www.maxonmotor.ch, 2013)

The motor types must have a nominal torque (=max. continuous torque) M_N that is greater than operating torque M_B ($M_B < M_N$). In work cycles, such as start/stop operation, the motor's nominal torque must be greater than the effective load torque (quadratically averaged). This prevents the motor from overheating ($M_{RMS} < M_N$). The stall torque of the selected motor should usually exceed the emerging load peak torque ($M_{max} < M_H$). Therefore accordingly the main characteristics of the selected brushed and brushless DC servomotor for joint motion in Archie are presented in table 3.7 and table 3.8.

Table 3.7 Brushed servo motors specifications

Brushed servo motor	24CR	unit
Nominal voltage	24	V
Efficiency, max.	83	%
No-load speed	5900	rpm
No-load current	129	mA
Stall torque	539	mNm
Speed constant	253	rpm/V
Torque constant	37.7	mNm/A
Current constant	0.027	A/mNm
Speed up to	5000	rpm
Torque up to	70	mNm

Table 3.8 Brushless servo motor specifications

Brushless servo motor (50Watt)	EC45flat	Unit
Nominal voltage	24	V
Max. Efficiency	82	%
No-load speed	6700	rpm
No-load current	201	mA
Nominal speed	5260	rpm
Nominal torque (Max. Cont. Torque)	84.3	mNm
Nominal current	2.36	A
Stall torque	822	mNm
Starting current	24.5	A
Speed up to	5000	rpm
Torque up to	70	mNm

In order to improve the motor connections to drive controller the new PCB is designed and embedded in brushless motor. This new attached PCB connections are described in Fig. 3.12.

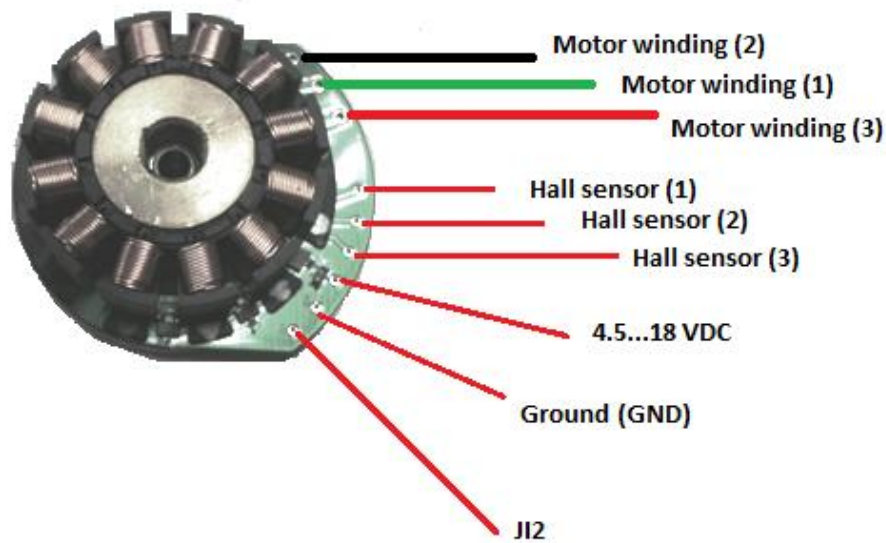


Fig. 3.12 Designed PCB for brushless motor and connections description

3.5.3 New Timing Belts

They are a positive transfer belt that can track relative movement. These belts have teeth that fit into a matching toothed pulley. When correctly tensioned, they have no slippage, run at constant speed, and are often used to transfer direct motion for indexing or timing purposes. Since the previous belts in Archie were not strong enough, hence the stronger belts (Polyurethane material) and thicker toothed pulleys with following specifications are replaced (Fig 3.13 and Table 3.9.).

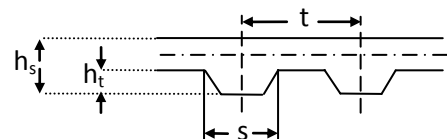


Fig. 3.13 New timing belts for hip transversal joints

Table 3.9 New Polyurethane transmission belt specifications

Profile	t(mm)	ht (mm)	Hs (mm)	s (mm)	Pitch length (mm)	No. of teeth
T2.5	2.5	0.70	1.3	1.5	245	98

3.5.4 New Encoder (PCB mounted)

There are two types of encoders: absolute and incremental. The absolute encoder consists of an optical-glass disk on which concentric circles (tracks) are disposed; each track has an alternating sequence of transparent sectors and matte sectors obtained by deposit of a metallic film. A light beam is emitted in correspondence of each track which is intercepted by a photodiode or a phototransistor located on the opposite side of the disk. With a suitable arrangement of the transparent and matte sectors, it is possible to convert a finite number of angular positions into corresponding digital data. The number of tracks determines the length of the word, and thus the resolution of the encoder.

Like the absolute one, the incremental encoder consists of an optical disk on which two tracks are disposed, whose transparent and matte sectors (in equal number on the two tracks) are mutually in quadrature. The presence of two tracks also allows, besides the number of transitions associated with any angular rotation, the detection of the sign of rotation. Often a third track is presented with one single matte sector which allows the definition of an absolute mechanical zero as a reference for angular position. The use of an incremental encoder for a joint actuating system clearly demands the evaluation of absolute positions.

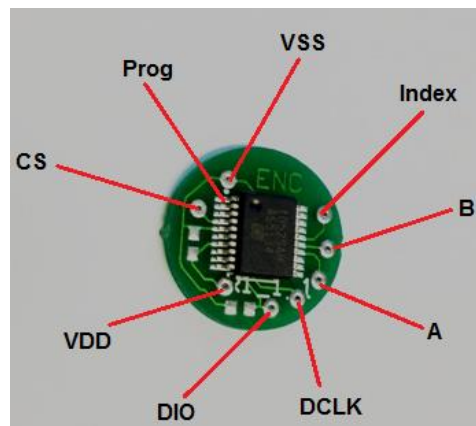


Fig. 3.14 Encoder pins connections on new designed PCB

This is performed by means of suitable counting and storing electronic circuits. To this end, it is worth noticing that the position information is available on volatile memories, and thus it can be corrupted due to the effect of disturbances acting on the electronic circuit, or else fluctuations in the supply voltage. Such limitation obviously does not occur for absolute encoders, since the angular position information is coded directly on the optical disk.

The selected encoder is a contactless magnetic rotary encoder for accurate angular measurement over a full turn of 360°. It is a system-on-chip, combining integrated Hall elements, analog front-end and digital signal processing in a single device.

To measure the angle, only a simple two-pole magnet, rotating over the center of the chip is required. The absolute angle measurement provides instant indication of the magnet's angular position with a resolution of 8.5 bit = 360 positions per revolution (Encoder, 2010). New embedded PCB and encoder pin assignment are presented in Fig. 3.14 and accordingly the pin description is mentioned in Table 3.10.

In order to run the motor smoothly with maximum efficiency the exact position of the rotor should be realized at any time and the encoders do that with high accuracy, high speed and high resolution. The rotor position constantly feed back to the motor drive controller which then calculates the most efficient signals for state of the coils. The encoders allows to move the rotors to where exactly is needed.

Table 3.10 Designed encoder PCB mounted pin description

Pin Name	Description
Prog.	Programming voltage input, must be left open in normal operation. Maximum load = 20pF (except during programming)
VSS	Supply ground
CS	Chip select input for 3-wire mode
VDD	Positive supply voltage (double bond to VDD_A and VDD_D)
DIO	Data I/O for serial interface. Command and data information over one single line. The first bit of the command defines a read or write access.
DCLK	Clock source for the communication over the digital interface. The maximum and minimum frequency depends on the mode.
A	Incremental output
B	Incremental output
Index	Incremental output

a) Encoder Protection

Stresses beyond those listed in Table 3.11 may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those which are indicated in electrical characteristics is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 3.11 Electrical characteristics of selected encoder (Min-Max rating)

Parameter	Min	Max	Unit	Comments
Supply voltage (VDD)	4.5	5.5	V	Except during OTP programming
Input Pin Voltage (VIN)	VSS-0.5	VDD	V	
Input Current (latch up immunity)	-100	100	mA	

3.5.5 Old Hardware Malfunction/Problems

The hardware structure of robot suffered from some mechanical and electrical parts, that are fixed by the following replacements.

- Lack of enough power

One of the major problem was that all joints could not move simultaneously. As a result at higher power, the current could not develop fully during the correspondingly short

commutation intervals. Therefore, the apparent torque produced is lower. Current is also fed back into the controller's power stage. Hence the more power was needed to satisfy this malfunctioning during the synchronized joints motion hence, new power supply source (36V,28A) was replaced with the old batteries.

- CAN communication interruption

The low quality of cables and loose connections in connectors caused interruption in data transferring, which is improved by new cables and the monitoring approach used in new motion controller software (ver. 2.1).

- Foot shape

Flat foot shape of previous biped robots Archie foot is another reason why it could not achieve human-like walk. A new prototype foot shape for adult human-like walk is designed and manufactured to overcome this problem.

Summary

In order to enhance the robot stability, improvements such as design and manufacturing of new ankle joint, new foot plate, using strong and high quality timing belt, new power supply and new designed PCBs are considered: for instance the new ankle joint (frontal) is designed and manufactured to increase DOF and allowing robot to stand on one leg that can cause much more stable robot during the stable walking performance (Fig. 3.15).

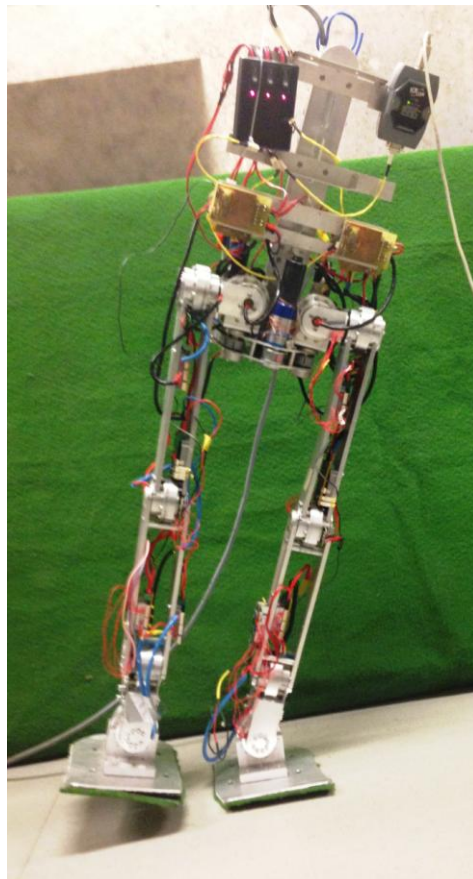


Fig. 3.15 Archie standing on one leg

Chapter 4 Design and Establishment of a Distributed Communication/Control Network

4.1 Introduction

In this chapter a distributed communication control system which is designed to establish and transfer commands/orders via proposed Control Area Network (CAN) is presented. Since the system needs efficient control method, fast calculation, fast communication and real-time data exchange the new real-time communication/control system for a biped robot Archie is designed and implemented.

4.2 Designed Real-Time Message Frame

The proposed control network must carry real-time messages to robot joints, as well as non-real-time messages. Therefore these messages must be properly scheduled on the network so that real-time messages meet their deadlines while coexisting with non real-time messages. The Archie communication network is an asynchronous multi-master serial data bus that uses Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) to determine access. In this protocol, just two lines are needed for data transmission therefore, it is very simple to extend it to other sub controllers.

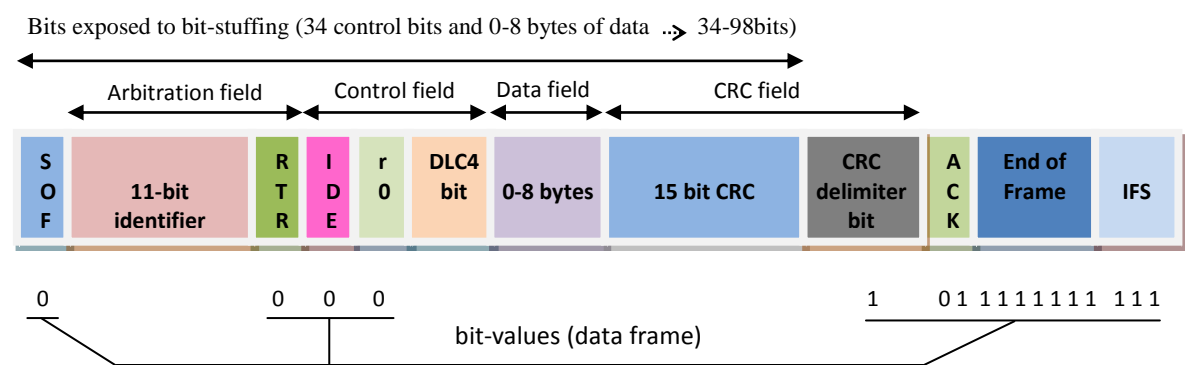


Fig. 4.1 Various fields in the designed CAN message frame for Archie

The designed real-time message transfer is controlled by 4 different types of frame: Data frames, Remote Transmit Request (RTR) frames, Overload frames and Error frames. The layout of each CAN frame has seven fields, as shown in Fig. 4.1, but in a proposed communication bus line for Archie, it is concerned only with the data length (DL) and the identifier (ID) fields. The DL field is 4 bits wide and specifies the number of data bytes in the data field, from 0 to 8. The ID field can be of two lengths. Each data frame is required to have a unique identifier.

The identifier serves two purposes beyond simply identifying the message. First, the identifier is used as a priority to determine which message, among those contending for the bus, will be transmitted next. Second, the identifier may be used by receivers to filter out messages they are not interested in, and so reduce the load on the receiver’s host microprocessor. This is the designed plan which is considered in motion controller program design (chapter 6).

4.2.1 Designed CAN Message Architecture

In this part the priority of transmitting commands (CAN messages) to specific driver/joint in Archie is described carefully, since it is the starting point of designing the fast and reliable communication network. The created messages are queued in the stations before being transmitted to the CAN bus. In the proposed CAN protocol for Archie, stations wait until the bus idle period is detected before attempting to transmit (Fig. 4.2).

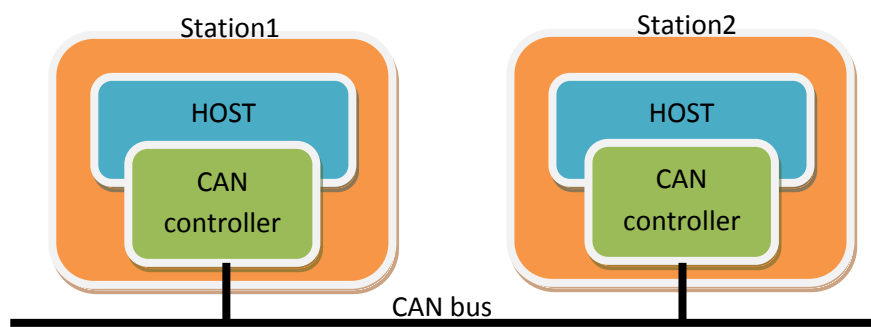


Fig. 4.2 Designed CAN bus line system for Archie

When two or more stations start transmitting at the same instant, they can monitor each bit on the bus to determine who should transmit the message with highest priority to Archie joint. Data is transmitted in messages containing between 0 and 8 bytes of data. An 11 bit number is associated with each message. The identifier is required to be unique, in the sense that two simultaneously active messages originating from different sources must have distinct identifiers. Typically, an identifier corresponds to a particular type of message from a specific source. The identifier serves two purposes: (1) assigning a priority to the message, and (2) enabling receivers to filter messages.

A station, filters messages by only receiving messages with particular bit patterns (typically using comparators and mask registers). Thus designed messages have no explicit destination, since any station with an appropriate filter may receive a message.

The use of the identifier as priority is the most important part of message design with respect to real-time performance (Fig. 4.3). These messages are a collision-detect broadcast bus, but takes much more systematic approach to contention. The identifier field of a desired message is used to control access to the bus after collisions by taking advantage of certain electrical characteristics of a CAN bus.

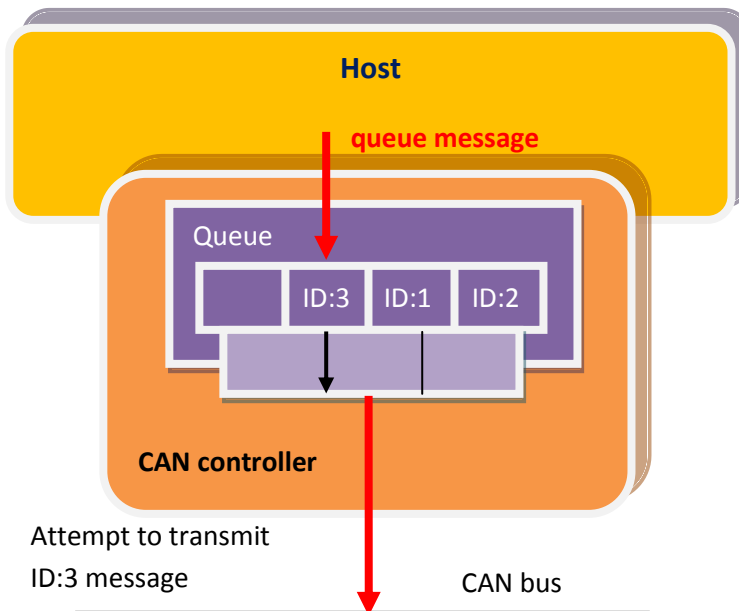


Fig. 4.3 Priority queue description

In fact, the proposed communication bus line acts like a large AND gate, with each station able to see the output of the gate. This behaviour is used to resolve collisions in Archie communication network design, which means that each station waits until seeing bus idle. When silence is detected, each station begins to transmit the highest priority message held in its output queue whilst monitoring the bus. The identifier is the first part of the message to be transmitted; the identifier is transmitted from most-significant to least significant bit.

4.3 Communication Channels Establishment Approach

In general, the purpose of establishing any bus line is to decrease the number of pathways necessary for communication between joints. A bus allows communication between several joints over one data channel and is characterized by how much information it can transmit at once. The amount of data is expressed in bits and corresponds to the number of physical lines over which the information is sent. In proposed designed network approach for Archie, the communication channels are established among all the processes before the application starts to communicate. The advantage of this approach is that, there is just a one-time effort involved in setting up of the communication channels. Since all the processes call the `init()` method, the message passing layer gets control in all the processes to perform any protocol specific activity that needs synchronization between the processes.

The proposed communication protocol, efficiently supports distributed real-time control with a very high level of security. It provides the error process mechanisms and message priority concepts.

4.4 Designed Message Transmit Planning Approach

In this section the data source usage and created CAN message time/transmit plan in the proposed network will be explained. Each node (joint) in robot is able to send and receive messages. A message consists primarily of an id, which represents the priority of the message, and up to eight data bytes. It is transmitted serially in to the bus line. This signal pattern is encoded in non-return-to-zero and is sensed by all nodes. The actuators and sensors are not connected directly to the bus line, but through a host processor and a CAN controller. If the bus is free, any node may begin to transmit. If two or more nodes begin sending messages at the same time, the message with the more dominant id (which has more dominant bits, i.e., zeroes) will overwrite other nodes less dominant id's, so that eventually (after this arbitration on the id.) only the dominant message remains and is received by all nodes. This mechanism is referred to as priority based bus arbitration. Messages with numerically smaller values of id. have higher priority and are transmitted first.

4.4.1 Designed Data Source Mechanism

A data source is transmitted as a message, consisting of between 1 and 8 bytes ('octets'). A data source may be transmitted periodically, randomly, or on demand. The identifier serves two purposes: filtering messages upon reception, and assigning a priority to the message. A station on a CAN bus is able to receive a message based on the message identifier: if a particular host processor needs to obtain the road speed (for example) then it indicates the identifier to the interface processor. In this design only messages with desired identifiers are received and presented to the host processor. Thus in designed command a message has no destination. The use of the identifier as priority is the most important part of CAN regarding real-time performance in motion control program which has been explained in chapter 6. In designed communication network for Archie any station waits for silence and then starts transmitting. If more than one station tries to transmit together then they all detect this, wait for a randomly determined time period. It is an example of a carrier sense broadcast bus, since each station waits until the bus is idle (i.e. no carrier is sensed), and monitors its own traffic for collisions. The identifier field of a designed message is used to control access to the bus after collisions by taking advantage of certain electrical characteristics.

In the motion controller program ver. 2.1 this procedure is implemented to resolve collisions: each station waits until bus idle. When silence is detected each station begins to transmit the highest priority message held in its queue whilst monitoring the bus. The full description of this method implementation is described in chapter 6 and 7.

4.4.2 Client-Server Interactions in Deigned Network

A client in Archie is a controller that makes requests to nodes to respond to its commands. In this design the robot joint is assumed a single-master network arrangement, in which the servo drives are the slaves and the PC/main motion controller is the master. Every servo drive has a unique ID. The network master does not require an ID. As a slave, the servo drive never

sends an unrequested message, other than emergencies. The drive responds only to messages addressed to its ID or to broadcast messages, which have an ID of 0. All messages sent by a servo drive in joints are marked with its own ID.

4.4.3 Process Design of Message Transmitting

The message is coded so that the most significant bit of the identifier field is transmitted first. If a station transmits a recessive bit of the message identifier, but monitors the bus and sees a dominant bus then a collision is detected. The station knows that the message it is transmitting is not the highest priority message in the system, stops transmitting, and waits for the bus to become idle. If the station transmits a recessive bit and sees a recessive bit on the bus then it may be transmitting the highest priority message, and proceeds to transmit the next bit of the identifier field. Because each message requires identifiers to be unique within the system, a station transmitting the last bit (least significant bit) of the identifier without detecting a collision must be transmitting the highest priority queued message, and hence can start transmitting the body of the message (if identifiers were not unique then two stations attempting to transmit different messages with the same identifier would cause a collision after the arbitration process has finished, and an error would occur). There are some general observations to make on this arbitration protocol. Firstly, a message with a smaller identifier value is a higher priority message. Secondly, the highest priority message undergoes the arbitration process without disturbance (since all other stations will have backed-off and ceased transmission until the bus is next idle). The whole message is transmitted without interruption.

4.4.4 Code Priority Transmission Scheduling

If two or more messages are transmitting at the same time and at least one of them transmits a '0' then the value on the bus will be a '0'. This mechanism is used to control access to the bus and also to signal errors. The designed protocol calls for nodes to wait until a bus idle period is detected before attempting to transmit. If two or more nodes start to transmit at the same time, then by monitoring each bit on the bus, each node can determine if it is transmitting the highest priority message (with a numerically lower identifier) and should continue or if it should stop transmitting and wait for the next bus idle period before trying again. As the message identifiers are unique, a node transmitting the last bit of the identifier field, without detecting a '0' bit that it did not transmit, must be transmitting the message with the lowest numerical value and hence the highest priority that was ready at the start of arbitration. This node then continues to transmit the remainder of its message, all other nodes having backed off.

Normally, the network nodes are only allowed to start transmitting when the bus is idle (Davis et al., 2007). Thus, when the bus is idle beyond the 3-bit inter-frame space and a node starts to transmit a message beginning with the dominant start of frame bit, then all the other nodes synchronise on the leading edge of this bit and become receivers – i.e. they are not permitted to transmit until the bus next becomes idle. In this case any message that becomes

ready for transmission after the leading edge of the start of frame bit has to wait for the next bus idle period before it can enter into arbitration.

However, to avoid problems due to clock drift, the CAN protocol also specifies that, if a CAN node has a message ready for transmission and detects a dominant bit at the 3rd bit of the inter-frame space, it will interpret this as a start of frame bit, and, with the next bit, start transmitting its own message with the first bit of the identifier without first transmitting a start of frame bit and without becoming a receiver. Again the leading edge of the start of frame bit causes a synchronisation.

The above high level description is a simplified view of the timing behaviour of proposed control network. Specifically, every node must synchronise to the leading edge of the start of frame bit caused by whichever node starts to transmit first. The proposed mechanism means that messages are sent as if all the nodes on the network shared a single global priority based queue. In fact, messages are sent on the bus according to fixed priority scheduling.

4.4.5 Inhibit Times

The inhibit time for a given message type is the minimum time that must elapse from the time the message is first transmitted until the time that it may be transmitted again. The purpose of inhibit times is to ensure that high-priority messages do not flood the bus and thereby prevent service messages of lower priority from being transmitted. The inhibit times of drives are defined only for asynchronous transmit process data objects (TPDOs).

The resolution of the inhibit time is 100 microseconds, with an accuracy of 2 milliseconds Communication Objects (COB). The data-byte units transported through a control network are called communication objects. Servo drive uses the following COB types (Table 4.1):

Table 4.1 Communication types (services) used in motor drive

COB type	Description
Service data object (SDO)	SDO messages are used to manipulate OD objects according to their IDs. The server receives the SDO, which specifies in its message which object is to be dealt with.SDO messages can be chained to form a "domain transfer," which is useful for sending large data items such as long strings Domain transfers are time-consuming, because the CAN bus is half-duplex. Each time a data segment is downloaded, a full-sized data segment is uploaded for verification, and vice versa.
Process data object (PDO)	PDO messages are used to manipulate OD objects without explicit reference to the object identifier, which is possible if there is an a-priori convention concerning the OD item referenced. Such conventions are called "PDO mapping", these are actually OD objects themselves, and may be defined and manipulated using SDO.
Network Management (NMT)	NMT objects are used by CAN clients to initialize a servo drive s a server.

4.5 Process Data Objects (PDOs) Design

4.5.1 Receive PDOs Method

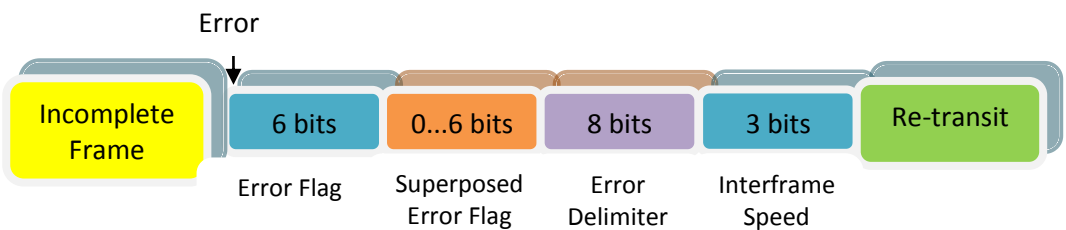
A Receive Process Data Objects (RPDO) is used to receive predefined and unconfirmed messages. An RPDO is received through use of an event, which may be asynchronous (such as “Message Received”) or synchronous with the reception of a SYNC. Four receive PDOs are used with the Elmo drive. In the following, the main characteristics of this method are listed:

- Objects that can be mapped and have write access can be mapped to each RPDO.
- Execution of the mapped objects begins in the lower index of the relevant mapping object.
- High-priority objects, used for high-speed motion modes, can be mapped to an RPDO to avoid the overhead in using the standard interpreter.
- Data of RPDOs is queued and it is passed for interpretation and execution at the next available background loop (Idle Loop) and according to the transmission type.
- RPDOs cannot be retrieved by a remote transmitting (RTR).

a) Message Error Detection Approach

Proposed CAN messages (commands) are designed as a robust and reliable form of communication for short messages. Each data frame carries between 0 and 8 bytes of payload data and has a 15-bit Cyclic Redundancy Check (CRC). The CRC is used by receiving nodes to check for errors in the transmitted message. A failure is considered when the received data could not be interpreted or executed (Fig. 4.4).

Active Error Frame



Passive Error Frame

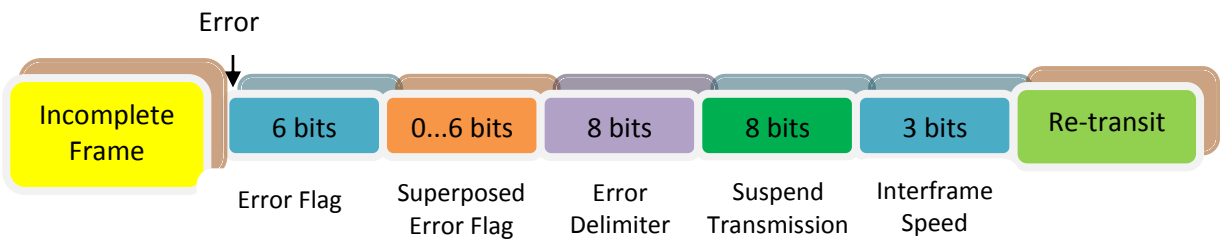


Fig. 4.4 Active/Passive error frame format in Designed CAN messages for Archie

4.5.2 PDOs Transmission Approach

Four transmit PDOs can be used in Elmo drives. TPDOs are used to retrieve an object (data) from the drive. Objects that have read access can be mapped to each one of the TPDOs. The transmitted data inside the TPDO is ordered according to the mapping order. The data starting from the LSB data is mapped first - in the lower index of the relevant mapping object.

4.5.3 Proposed PDO Mapping in Communication Network

PDO mapping is a convention that assigns (maps) an object from the object dictionary (data payload) to a PDO. Once mapped, the PDO can carry the assigned data items without explicit reference to the object dictionary, thereby saving on communication and CPU overhead. Only a subset of the objects in the object dictionary can be mapped to a PDO, which can either receive (RPDO) or transmit (TPDO).

The mapping of an RPDO enables reception of commands and variables, for instance, efficient transmission of high-speed online motion commands to the drive, whereas the mapping of a TPDO enables the drive to send a predefined message in response to an event.

4.5.4 Synchronous Trigger in Motion Controller Program

Synchronous triggers are always related to the previous SYNC reception. The received message is buffered but actually transmitted for execution at the next SYNC message. Only one RPDO can be buffered for synchronous trigger. If another RPDO arrives before the SYNC, it overrides the previous RPDO without any notification.

This method enables the simultaneous synchronization of executing commands in several drives. When a SYNC arrives, the buffered message is performed in the next available background cycle (Idle loop).

4.6 Network Management Message Structure (NMT)

NMT commands are used to control the communication state of the servo drive and to broadcast messages to all other connected servo drives. When the servo drive is powered on, it enters the initialization state.

After completing the boot sequence, it automatically enters the pre-operational state. The transition between pre-operational, operational and prepared states is carried out according to Network Management messages. In the following, the servo drive status in different NMT message states is described (Table 4.2).

Table 4.2 Network Management States

State	Description
Unpowered/Initialization	Servo drive is not ready, or it is booting. Drive will not respond to communication and will not transmit anything.
Pre-operational	Servo drive boot sequence is complete, but no command has been received to enter operational mode. The servo drive will respond to SDO and NMT message, but not to PDO's
Operational	Servo drive is fully operational, responding to PDO, SDO and NMT message
Stopped	Servo drive can be respond only to NMT objects

An NMT message is always two bytes long: the first byte is the command specifier and the second byte is the ID of the units that are to respond to the message. If the ID is 0, the NMT message will be executed by the entire set of connected servo drives. The NMT mapping design and form of the negotiation (from 1 to 14) is depicted in Fig. 4.5.

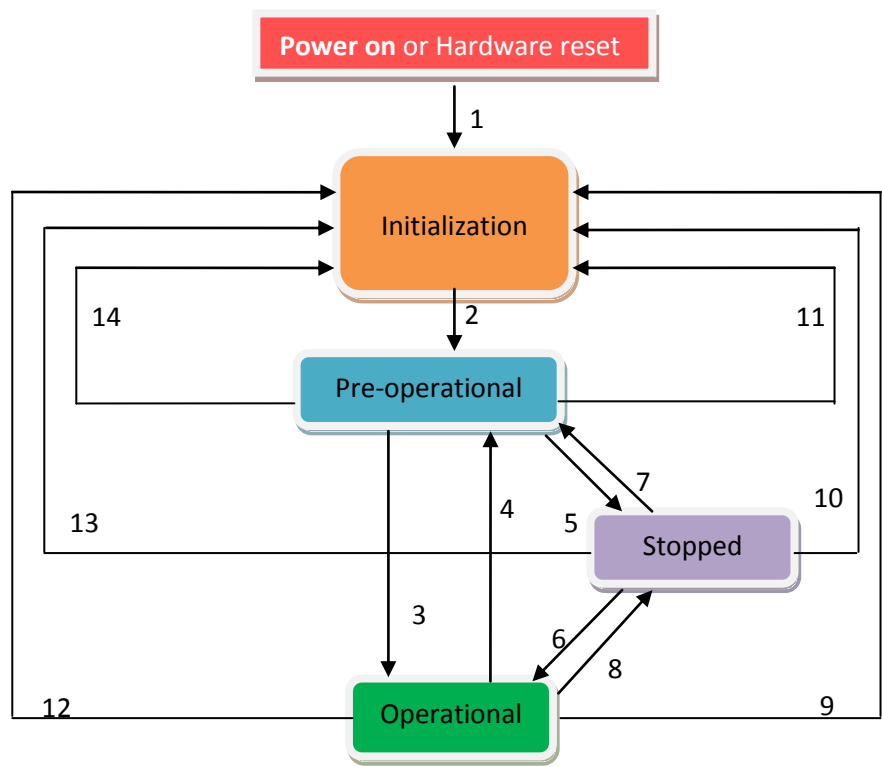


Fig. 4.5 NMT message mapping

4.6.1 SYNC and Time Stamp

The Archie drives are synchronized by the transmission of a SYNC message, whose arrival time is captured by the drive. Upon reception of the SYNC, the drive latches its internal timer. The Time Stamp causes a clock synchronization cycle to be executed. The Elmo drive uses the Time

Stamp as an absolute timer and adjusts its internal time in relation to the time latched in the last SYNC. To synchronize the master and drive clocks to full precision, the synchronization process is filtered in order to ensure that the timing jitter of the time stamping process does not adversely affect motion smoothness (www.elmomc.com, 2013).

4.6.2 Set and Query Commands in Network Controller

The client sends commands (RPDO2) for setting variables in eight bytes (DLC=8). The drive transmits the reply (TPDO2) as an asynchronous event of the received object.

a) Binary Interpreter Commands

With CAN bus cable, the interpreter commands are sent in binary form and are used for setting and retrieving all numerical data of the servo drives setup. The commands used by the binary interpreter for CAN communication are very similar to commands of ASCII interpreter used for RS-232 communication. The following table (Table 4.3) summarizes the main differences between the binary interpreter used for CAN communication and the ASCII interpreter used for RS-232 in Archie motion controller program.

Table 4.3 Comparison of ASCII vs. Binary interpreter commands

Feature	ASCII Interpreter	Binary Interpreter
Command Length	Depends on data	Fixed: 8 Bytes for set commands; 4 Bytes for Get commands.
Delimiter	;or<CR> for commands and servo drive responses.	None
Servo Drive responses to Set commands	Always	Drive does not respond to Set commands. An emergency object is sent if commands execution fails.
Long response strings	Returned by certain commands, such as LS and BH	No support for returned long strings, which are read via SDO's.

TPDO2 is mapped by default to the Transmit binary interpreter object. The binary interpreter supports three types of commands:

- Set value

These commands can be four or eight bytes in length. The transmitted message includes either the reflection of the Set command or an error code, if failure has occurred.

- Get value

These commands can be four or eight bytes in length. An 8 bytes response includes the reflection of the command and the resulting numerical value, and an error if a fault has occurred.

- Execute command

This command can be four or eight bytes in length. An 8-bytes response includes the reflection of the command and the resulting numerical value, and an error if fault has occurred. If an interpreter command cannot be serviced for any reason, bit 6 in byte 3 of TDO2 is set on, and byte 4 of the response contains the Elmo error code.

These commands are used to instruct the drive to perform a sequence. The reply to these commands is only an acknowledgement or an error code; there is no value for executing command. Execute commands are a unique case of RPDO2, which can be used with a DLC of either 4 or 8 (Table 4.4).

Table 4.4 DLC4 values frame

DLC4				
Byte	0	1	2	3
Hex value	42	47	0	0

The reply message (Table 4.5) is always eight bytes long and indicates either success or failure (error).

Table 4.5 Execute command reply frame (Success/Failure)

Success								
Byte	0	1	2	3	4	5	6	7
Hex value	42	47	0	0	0	0	0	0

Failure (error code 58 (3Ah) for a "Motor must be on")								
Byte	0	1	2	3	4	5	6	7
Hex value	42	47	0	40	3A	0	0	0

b) TPDO2 Structure in Controller Program

The drive controller replies (TPDO2) to query and set requests in eight bytes (DLC=8):

- Bytes 0 to 3 are the header, which includes the responding command, command index (when needed) and data type (float or integer). It also indicates whether the response data is true data or an error code.
- Bytes 4 to 7 are data, which is either a reflection of the host Set command or an error code according to the EC command.

4.7 I-converter (USB to CAN)

In this stage the commands are ready to be transmitted to robot joints. The solution is a cost-effective device for integrating the CAN bus to the PC by using the standard USB interface which is called USB to CAN converter. So when the connection between the I-7565 and the PC during the runtime of the computer would be established, then the PC automatically loads the relevant device driver.

By installing the I-7565 converter in robot, the PC can be access/control the CAN devices by the utility tool and be the CAN host, network monitor or CAN-interface. It can handle both an 11-bit and 29-bit ID format according to whether it is a CAN 2.0A or 2.0B. In order to provide high performance when converting data, the converter has built in software FIFO queues, which include 1000 CAN data frames (see chapter 7). In the following most important features of the I-converter which are considered in motion controller program ver. 2.1 are presented (Fig. 4.6). In table 4.6 the general hardware specifications of the I-7565 converter are listed.

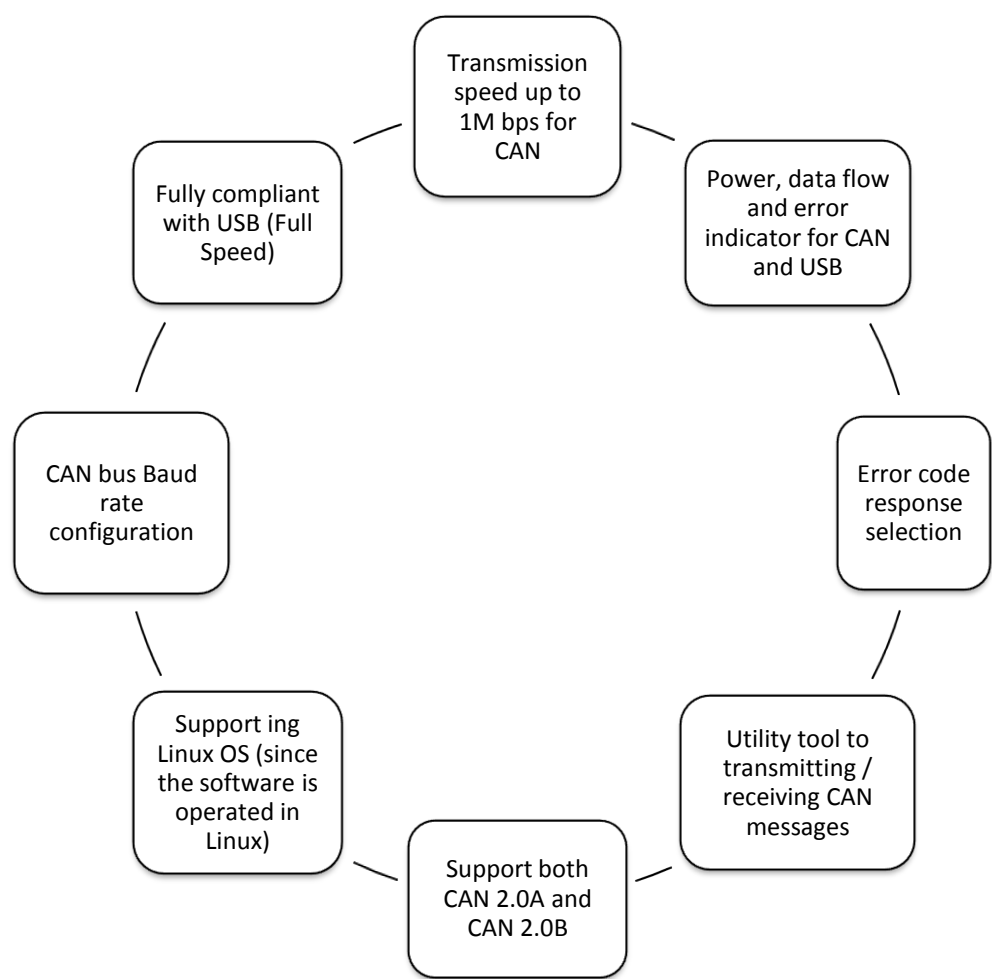


Fig. 4.6 Necessary USB to CAN converter features used in motion controller program

Table 4.6 General USB to CAN device specifications

CAN Interface	
Controller	Microprocessor inside with 20 MHz
Connector	9-pin male D-Sub (CAN_L, CAN_SHLD, CAN_H, N/A for others)
Baud Rate	10 k, 20 k, 50 k, 100 k, 125 k, 250 k, 500 k, 800 k and 1Mbps
Receive Buffer	1000 data frames
Max Data Flow	250 fps
Power	
Power Consumption	1.5W
LED	
Round LED	ON LED: Power and Data Flow; ERR LED: Error
Mechanism	
Installation	DIN-Rail
Dimensions	108mm x 72mm x 33mm
Environment	
Operating Temp.	-25°C to 75°C
Humidity	5~95% non-condensing

4.7.1 Data Communication Bus via CAN-Bus Converter

In this part the methods used to implement the I-7565 module into their applications in a quick and easy way will be described. USB to CAN bus converter has a utility features and software as depicted in Fig. 4.8 which should be installed and connected (step1) for communication from the PC to the converter via the USB cable into the i-7565 module and in the next step send the message to the Elmo controller. PC's USB connection should be set (step2) by choosing the proper port. The green light on the converter will turn on while the USB port is connected to the converter (step3).

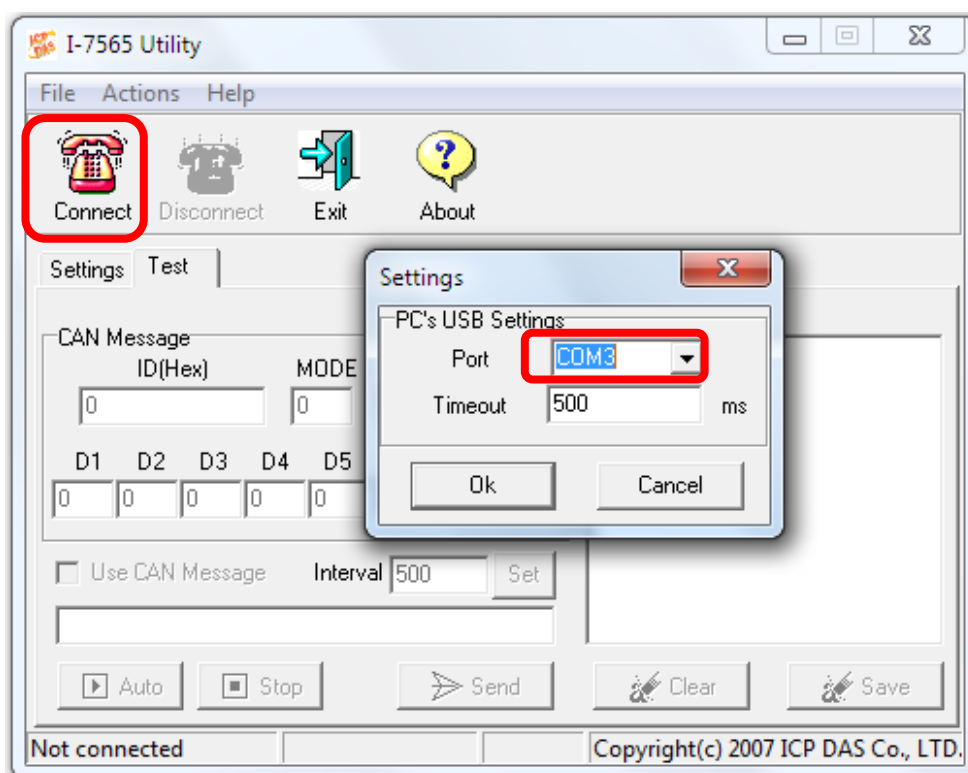


Fig. 4.7 USB to CAN converter connection establishment

In the I-7565 software there are two tabs, one for setting the USB parameters, and one for CAN bus parameters. The setting procedure is as follows: first should set the Init/Normal switches on the back of the I-7565 A to the “Init” position. Then, connect to the USB port of PC. The ON LED of the I-7565 a will be flash approximately once per second. That means that the I-7565 is in the configuration mode. If this process is successful, the I-7565 Utility shows connection settings as presented in Fig. 4.7.

USB parameters in setting mode are included in Add Checksum and Error Response, In order to match these parameters on the PC; they should be configured as follows:

- Add Checksum: No
- Error Response: No

The CAN bus parameters are included in CAN specification and CAN bus Baud rate. These magnitudes should be defined as depicted in the follow:

- CAN specification: 2.0A
- CAN bus baud rate: 1000K bit/sec.

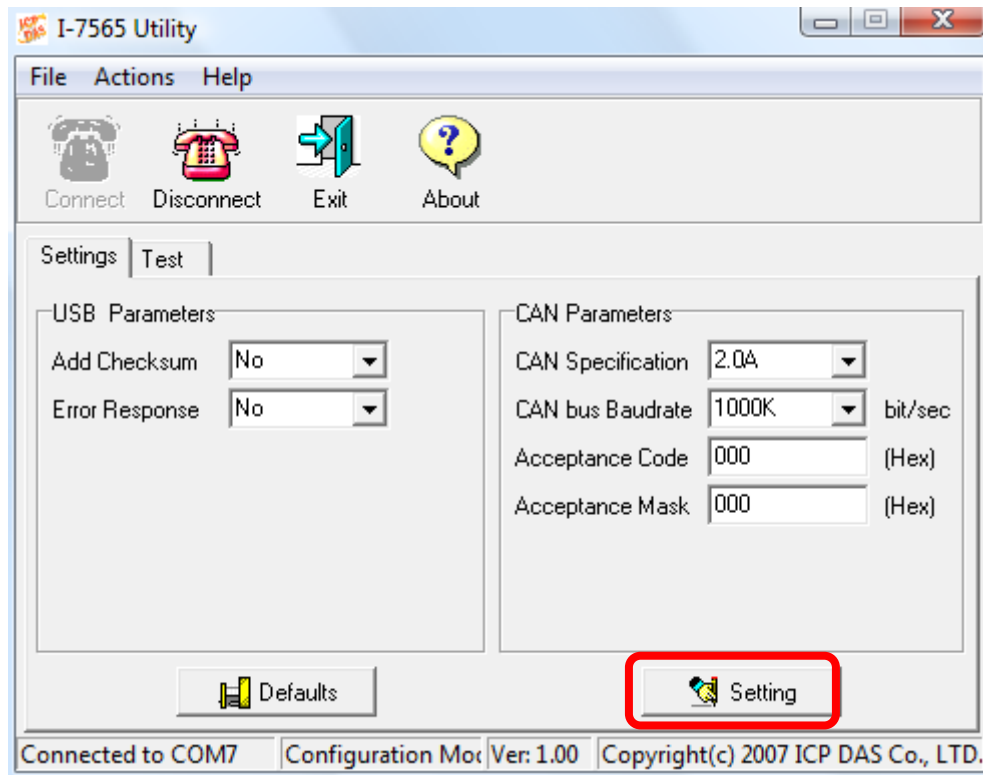


Fig. 4.8 Code converting-setting mode

On the other hand there is another tab so called test (Fig. 4.8). It is used while the series of message send to the Archie for joints movement and monitoring stream of command message transferring. The most important attributes of this software converter is that user can utilize it to check each motor individually for rotation or changing the each joint from initial position to the desired position manually.

In next step "Use CAN Message" is checked and then input value to the "CAN Message" frame on Utility A. can be executed and sent (Fig. 4.9). Thereafter, the utility will automatically transfer these CAN messages to the command string with ASCII 0x0d, and send it out through the USB to CAN cable, meanwhile converter sends back the receive signal command (Fig. 4.10). Afterward the commands can be real-time transferred to join drives via presented communication bus line.

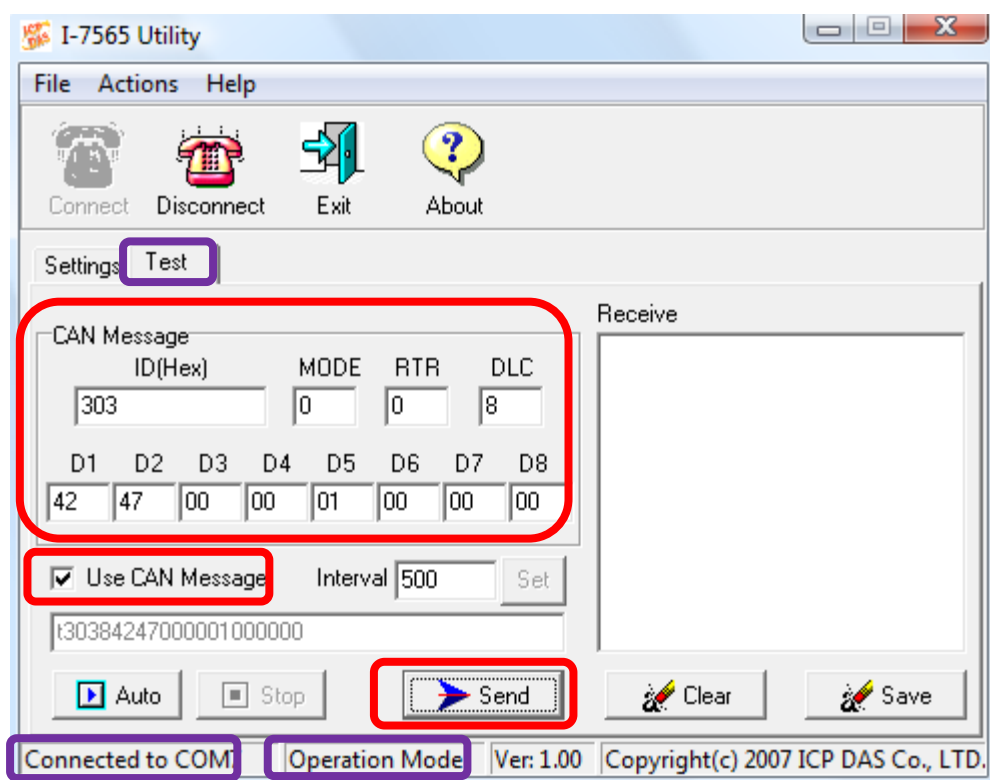


Fig. 4.9 CAN message structure design in test mode

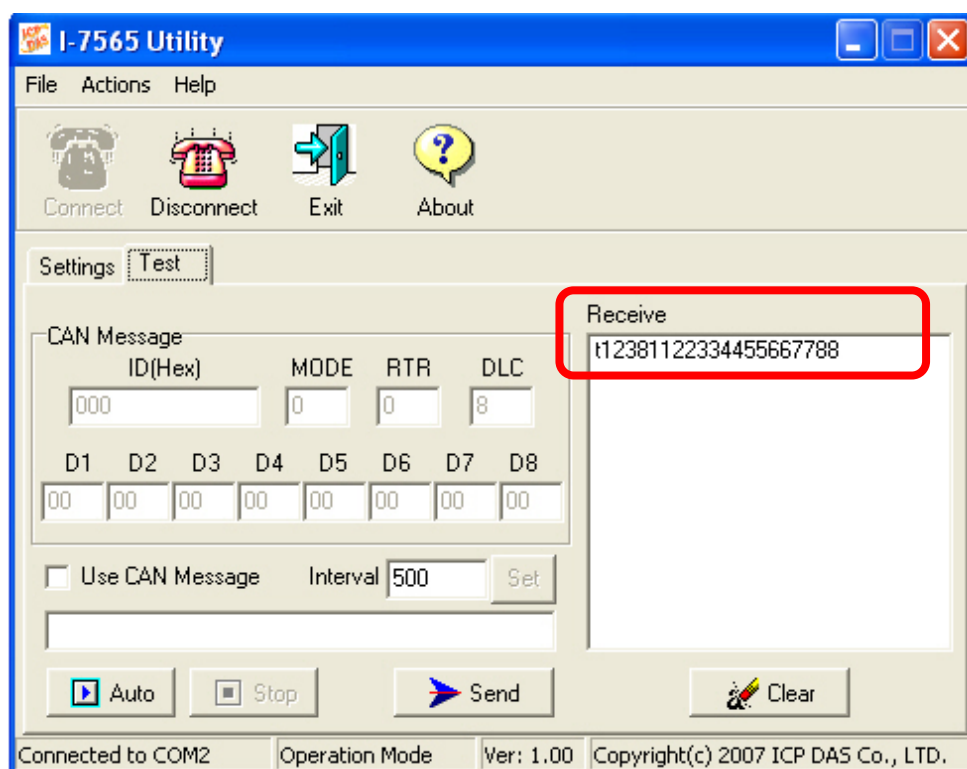


Fig. 4.10 CAN message-receive signal format

Chapter 5 Motion Creating System Design based on PT and PVT Method

5.1 Introduction

Since humanoid robots are designed to interact closely with humans, it is important that their motion look natural. One reason is that we have no clear idea what “natural” actually means. Intuitively, we might want the robot to minimize erratic arm and leg movements or to move with a consistent style, but it is difficult to quantify this intuition.

In first part of this chapter the goal is to generate “natural-looking” motion primitives for a biped humanoid robot Archie. The focus is on creating realistic, adaptable and synchronized walking motion. The robot is moved according to the joint angle trajectories. The trajectories are generated based on the paper "Planning Walking Patterns for a Biped Robot" (Huang et al., 2001). The provided process for joint motions are different. Joint angles are mainly calculated by inverse kinematics and joint angles are mainly generated as patterns from step primitives. To realize smooth walking, one reasonable approach is to create an initial motion and to transform it to keep balance. This process should preserve the characteristics of the original motion particularly for walking. For this, the important factors of the characteristics in the walking motions must be clarified. It is necessary to avoid the problems of the mechanical structure and adding the missing joint drives to enhance walking stability. It would be required to attach the new joint mechanism and actuator because of the lack of stability on ankle frontal joint. This hardware improvement is already presented in chapter 3.

5.2 Biped Robot Walking Problem

Even though there has been much research and certainly some interesting progress in biped locomotion, there is certainly no single solution that provides robots with the same capabilities as humans (Xiao et al., 2006). What seems to be especially lacking is a quantitative method for comparing different solutions. By definition, walking is a form of locomotion in which the body's center of gravity moves alternately on the right side and the left side. All these phenomena observed during the foot-ground contact have to be introduced into the model to make it realistic. A biped locomotion phase can be classified into a contact, a single and a double supporting phase to generate a leg motion (Jo & Jayamani, 2009). During the single supporting phase, one foot is not constrained to the ground but the other foot is on the ground. As soon as the heel of the swinging foot reaches the ground, the single supporting phase is changed to the contact phase. If the toe and heel of the swinging foot contacts on the ground, the phase becomes a double supporting phase. Fig. 5.1 shows a locomotion phase for a leg pattern generation.

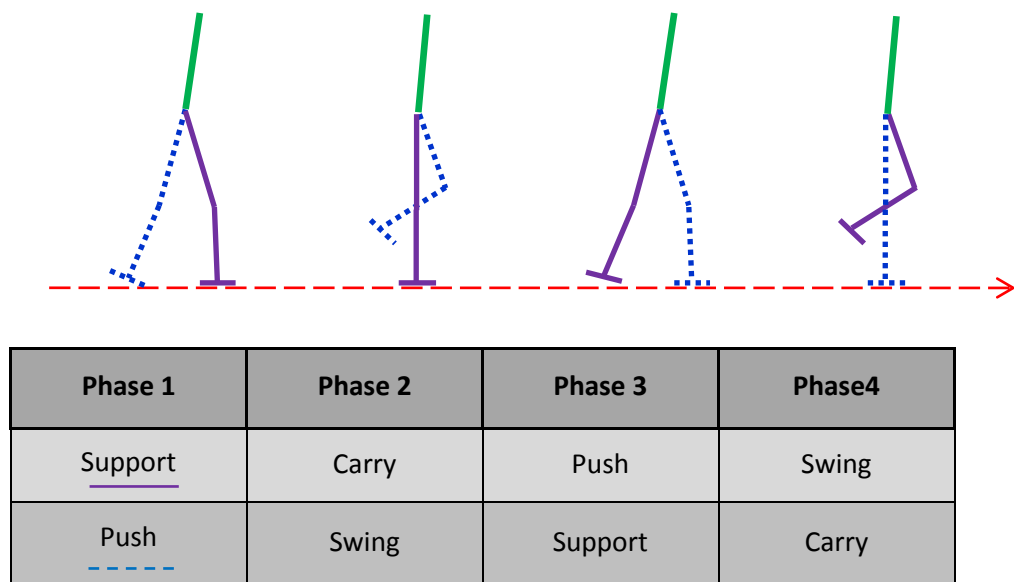


Fig. 5.1 Biped walking phases

Biped walking is a periodic phenomenon. Leg motion is generated under the assume that all the area of a foot sole contacts with the floor when it is supporting the body. In other words, a sole does not rotate during that time. In terms of dynamics, this assume is satisfied when the point at which the moment to the robot body is zero exists in the area of the sole surface. In this time, the sole does not rotate. As already described the point is called ZMP and the area is called supporting area. Therefore if a robot is supported by both feet, supporting area corresponds to the convex area which consists of both soles.

Many studies on gait planning have assumed that the double-support phase is instantaneous. But in such a case, the related hip has to move too fast (Ha & Choi, 2007). In order to maintain its stability, the robot's center of gravity, in the case of static stability or the ZMP in the case of dynamic stability, must be transferred from the rear foot to the front foot during the short double-support phase. On the other hand, if the interval of the double-support phase is too long, it is difficult for the biped robot to walk at high speed. The interval of the double-support phase in human locomotion is about 20% so this value is used as the basis for Archie joint trajectory calculation (Huang et al., 1999). To make the robot walk, a gait trajectory is designed offline.

In the biped robotics research field, the gait trajectory, known as the walking pattern, generates the relative position trajectories of two feet with respect to the pelvis center (Kim and Park, 2007). The stable walking of biped robots can be characterized by some criteria. Static walking is characterized by maintaining the COG inside of the support region (the interior perimeter formed by the foot or the feet in contact with the floor).

This section is focused on two problems: the first one is how to generate a complete foot trajectory; the second one is how to derive a smooth hip motion with high stability. If both foot trajectories and the hip trajectory are known, all joint trajectories of the biped robot will be determined by kinematic constraints. The walking pattern can therefore be denoted uniquely by both foot trajectories and the hip trajectory. When the robot moves straightforward, the lateral positions of both feet are constant. The lateral hip motion can be

obtained similarly as the sagittal hip motion and swing ankle's trajectory in the sagittal plane. Some of the restrictions for designing the swing ankle's trajectory are considered as follows:

- When the swing foot touches the ground and becomes stance foot, its speed should be zero or very small.
- When the stance foot becomes swing foot and leaves the ground, its speed should be zero or very small.
- The swing foot should be above the ground with a small distance when swinging.
- The trajectory should be smooth enough.

In chapter 7 the correlation between the actuator specifications and walking patterns is presented with experimental tests and results on the robot.

5.3 Gait Planning Method in Archie

Foot-support interactions and load-regulating mechanisms play a crucial role during stepping. In order to achieve the natural gait of biped robot like human, human's gait has to be modelled precisely. However, since the human's gait is composed of dynamic motions on the sagittal, frontal and transverse plane, the complete gait of the biped robot can be achieved only if the gait is analyzed on two more planes.

The humanoid motion on the frontal plane aims to move the ZMP of the biped robot from one foot to the other. To control the robot on the frontal plane each leg of the Archie has two DOFs, one at the hip joint and the other at the ankle joint. To simplify the gait planning, it is assumed that the biped robot always keep two legs parallel.

The foundation of Archie walking concept is based on the implementation of planning walking patterns that has following major principles:

- Formulation of the constraints including ground conditions, foot and hip trajectory. Various foot motion can be provided by adjusting the values of the foot constraint parameters.
- A formulated hip motion, makes it possible to derive a highly stable, smooth hip motion without first designing the desired ZMP trajectory.
- A mutual relation between the driver specifications and walking patterns were clarified. Therefore, it is possible to select a walking pattern with small torque and velocity of the joint actuators after the ground conditions and the stability constraint are satisfied.

5.3.1 Trajectory Planning Approach in Archie

Trajectory planning techniques in Archie allow the generation of the reference inputs to the motion creating system. The problem of controlling a joint can be formulated (as that to determine the time history of the generalized torques) to be developed by the joint actuators, so as to guarantee execution of the commanded task while satisfying given transient and steady-state requirements. In order to construct realizable trajectory, limitation of parameters before trajectory generation and validation of trajectory after generation are carried out. In Fig. 5.2 and Table 5.1 the main parametric model values of Archie and their values to generate the smooth walking trajectory are presented.

Table 5.1 Archie parametric model values

a_1	a_2	a_3	a_6	d_3	d_6
8cm	26cm	30cm	5.6cm	7.4cm	4cm

- a_1 : distance between center of frontal-lateral joints in ankle
 a_2 : shin length
 a_3 : thigh length
 a_6 : distance between center of transversal joint in hip and COG
 d_3 : distance between center of lateral & frontal joints in hip
 d_6 : distance between center of frontal & transversal joints in hip

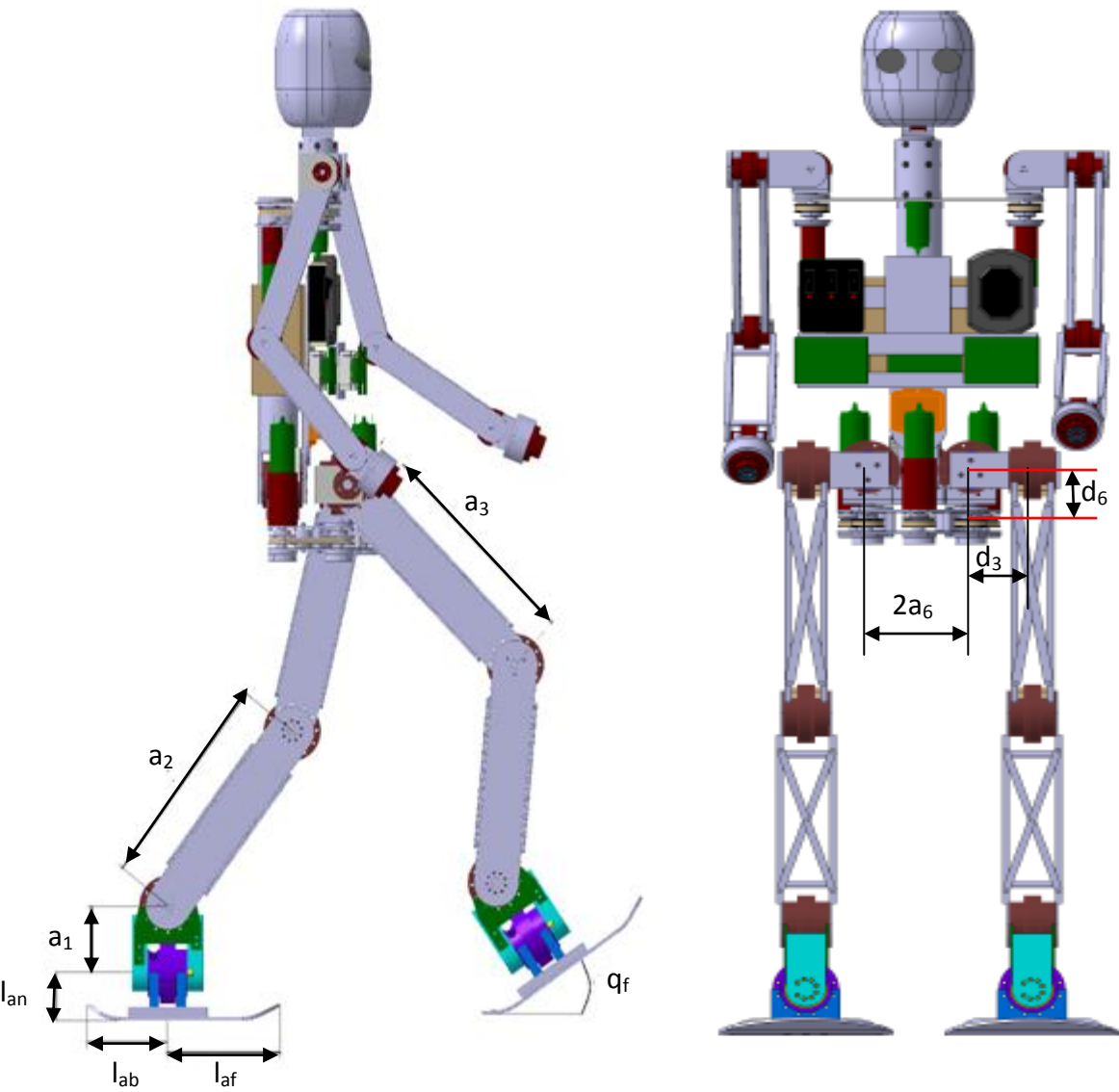


Fig. 5.2 Archie leg parameters

5.3.2 Calculated Foot Trajectory

When there are various constraints such as ground conditions and various foot motions, the order of the polynomial is too high and its computation is difficult, and the trajectory may oscillate. Hence the boundary conditions of the joint are selected based on the designed walking gait. There are three different categories of strides that form a full walking cycle, i.e. starting stride, full stride and ending stride. Furthermore, trajectories of the swing foot can be divided into 3 stages (Fig. 5.3). At the initial stage of swing phase, the heel of the foot is lifted with the toe used as a pivot. When a sufficient rotational motion is done, the foot is to be completely off the ground and swings in the air, which is the second stage. As the swing foot lands on the ground, the heel initially contacts with the ground. As soon as the heel hits the ground, the toe is lowered by the foot rotational motion with the heel as a pivot. The foot rotational motion continues until the entire sole of the foot becomes in contact with the ground. This is the end of the third stage of swing foot motion. During this phase, the other foot, that is the supporting foot, does not change its position and orientation, and the whole part of its sole is in contact with the ground. As soon as the third phase of the swing foot ends, the foot of the supporting leg goes into its own first stage of the swing motion, i.e., heel-lifting motion stage (Huang et al., 2001).

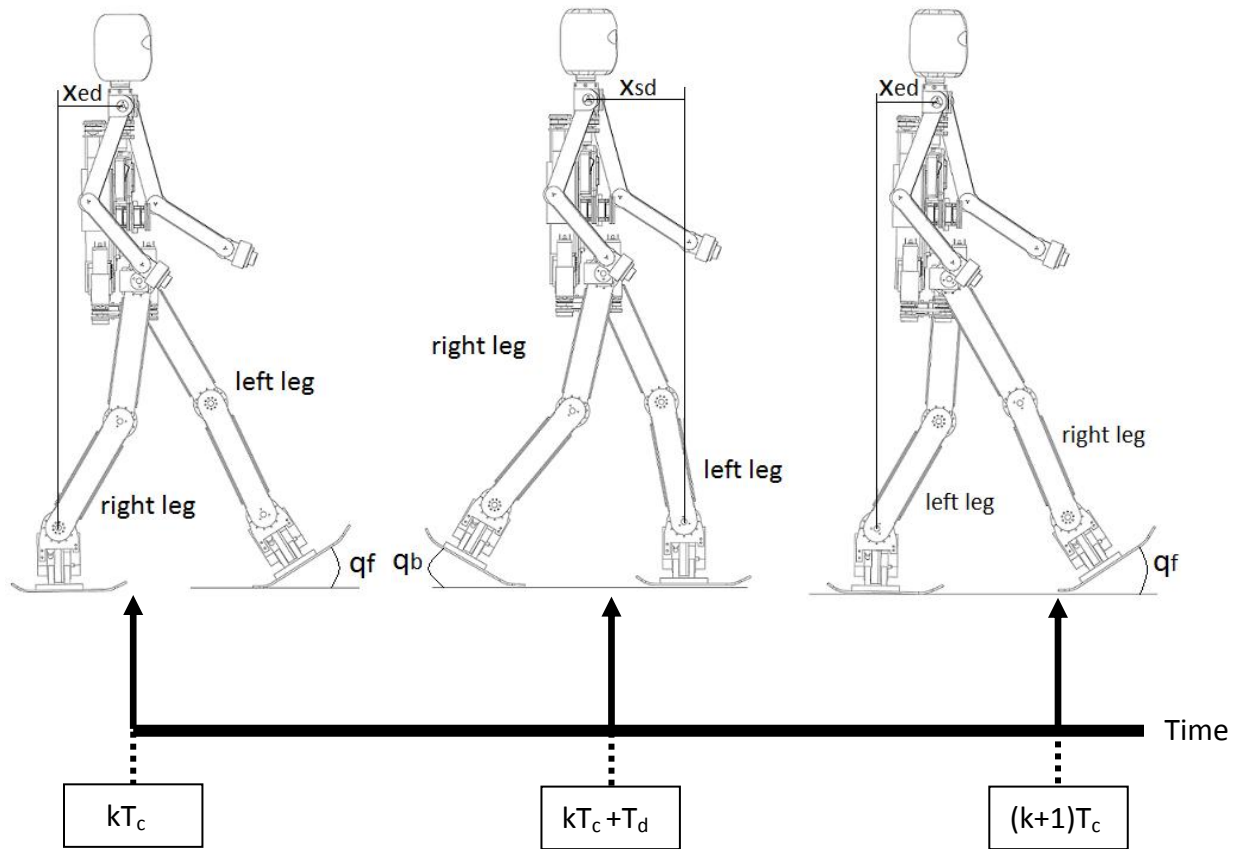


Fig. 5.3 Swing foot trajectory phases-time

One foot is in front of the other by distance 50 cm, which is the stride (Fig. 5.4). After a single step, the posture of the legs becomes identical except that the front and rear feet are reversed. Taking a single step takes time T_c , which is one half of the stride period.

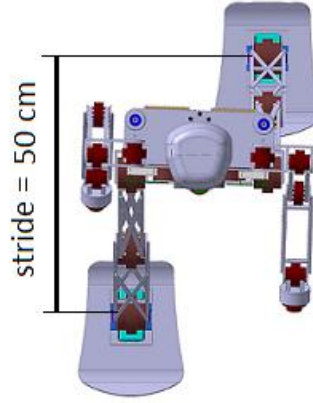


Fig. 5.4 Archie stride

The time of the k^{th} step is from kT_c to $(k+1)T_c$, $k=1,2,3,\dots$ is the number of steps. To simplify the analysis, the k^{th} walking step is defined to begin with the heel of the right foot leaving the ground at $t=kT_c$, and to end with the heel of the right foot making first contact with the ground at $t=(k+1)T_c$ (Fig. 5.3). The left foot trajectory is same as the right foot trajectory except for a T_c delay. q_b and q_f are the designated angles of the right foot as it leaves and lands on the ground, respectively. Assuming that the entire sole surface of the right foot is in contact with the ground at $t=kT_c$ and $t=(k+1)T_c+T_d$. where T_d is the interval of the double-support phase.

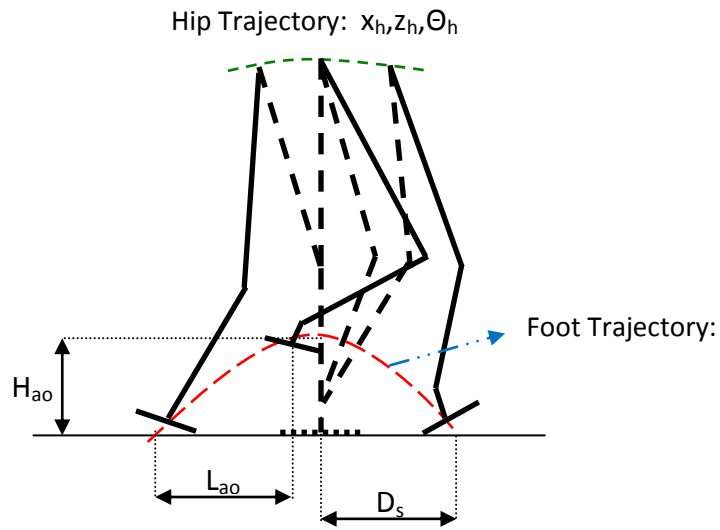


Fig. 5.5 Archie foot and hip trajectory calculation parameters

L_{ao} and H_{ao} are the position of the highest point of the swing foot, and $2D_s$ is the length of one step, kT_c+T_o is the time when the right foot is at its highest point, l_{ao} is the height of the foot, l_{af} is the length from the ankle frontal joint to the toe, l_{ab} is the length from the ankle frontal joint to the heel (Fig. 5.5 and Table 5.2). The foot trajectory calculation is presented in Appendix A.

Table 5.2 Archie trajectory parameters values (with stride=50 cm)

l_{an}	l_{ab}	l_{af}	l_{ao}	H_{ao}	D_s
5.6 cm	6.5 cm	12.5 cm	D_s	12 cm	25 cm

5.3.3 Calculated Hip Trajectory

Joint angles of each leg are calculated from position and orientation of the hip and the feet using inverse kinematics (Mohamadi Daniali, 2013). The design of pelvis movement is accomplished by editing the position and the orientation trajectories of the pelvis according to the time sequence determined by the gait pattern. The trajectory of the leg joints is calculated from the gait pattern and pelvis movement. Hip movement is edited by control points and finding interpolation functions.

Walking cycle that considered for Archie walking is composed of three phases: a starting phase in which the walking speed varies from zero to a desired constant velocity, a steady phase with a desired constant velocity, and an ending phase in which the walking speed varies from a desired constant velocity to zero. First, the hip motion ($x_h(t)$) of the steady phase is obtained as follows:

One-step cycle, can be described by two functions: one for the double-support phase and one for the single support phase. x_{ed} and x_{sd} denote distances along the axis from the hip to the ankle of the support foot at the start and end of the single-support phase, respectively (see Appendix A).

The hip trajectories are slightly modified such that the upper body motion is steered naturally, meaning that it requires practically no actuation. This has the advantage that the upper body actuation hardly influences the position of the ZMP. Validation after trajectory generation is carried out for the following points;

- Range limitation of leg joint angle,
- Limitation of leg joint angle velocity,
- Collision between leg links

Walking parameters values for Archie walking (stride=50 cm) are given in Table 5.3. The optimized chosen values are obtained based on the various experimental tests and actual desired walking data in $x_a(t)$, $x_h(t)$, $z_a(t)$ and $z_h(t)$ graphs (see Appendix A).

Table 5.3 Walking parameters values

q_b	q_f	x_{ed}	x_{sd}	H_{min}	H_{max}
0 deg.	0 deg.	6.5 cm	-3.5 cm	56 cm	59 cm

Considered time boundary condition value for Archie walking (stride= 50 cm) are as given in Table 5.4. Accordingly the leg and hip motion trajectory in x-axis is depicted in Fig. 5.6.

Table 5.4 Time boundary condition value

Double support time interval 40%(T _c)	Step time	Time for the maximum height of ankle joint in Z direction
T _d	T _c	T _o
0.4(T _c)	2 sec.	(T _d +T _c)/2

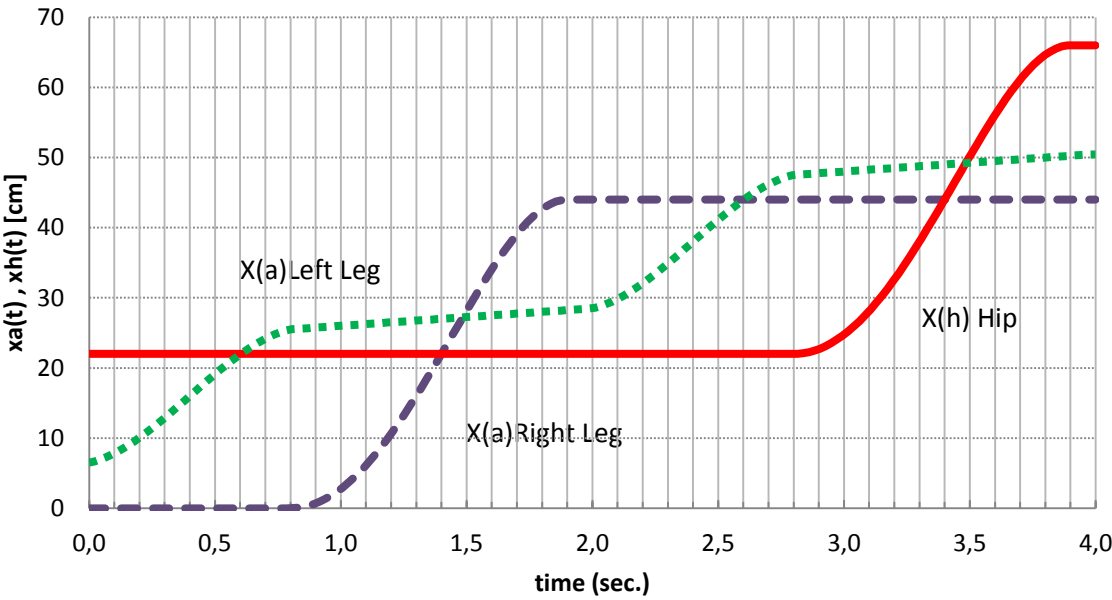


Fig. 5.6 Calculated leg and hip motion in x-axis

During the first interval, the supporting leg rotates slightly to push the robot forward, this motion being made mainly, by the ankle and hip joints. The balance leg starts raising and hence the hip and ankle joints in the supporting leg must counteract the tendency for the robot to tip over. During this period the knee motion is limited to a slight bend.

In this limited motion, the ankle joint moves almost like the hip joint and hence there is similarity between their motion. The raise of the hip by the balance leg is followed by the bending of the balance knee, so that there is enough space for the robot to account for the forward motion due to gravity.

In the air, the balance foot is not influenced by any external forces/torques and therefore the motion of the hip joint is similar to that of the knee joint leading to similar patterns for the joint angle. The main difference in the sign (angle value) being due to the opposite rotations performed by each of them.

Afterward, the supporting leg effectively pushes forward the whole body, whereas the balance leg moves forward, extending its knee (previously bended) and preparing for the impact with the ground. During this period, the hip joint in the balance leg moves forwards and backwards to compensate the gravity motion. In the third interval, the supporting leg extends itself slightly further whereas the hip joint in the balance leg pushes the whole leg

down. Therefore the designed hip and ankle joint trajectory are derived and depicted in Fig. 5.7.

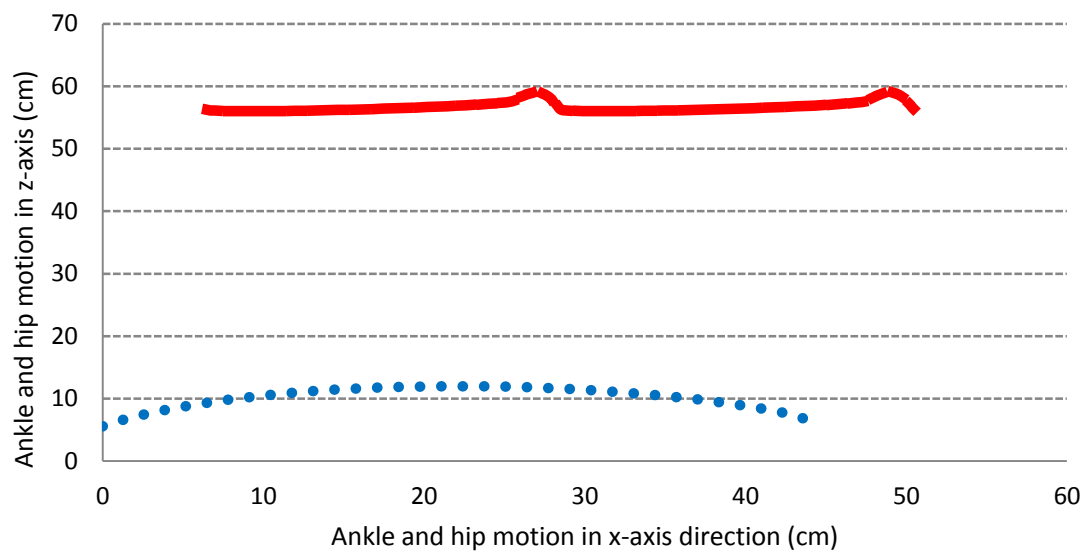


Fig. 5.7 Calculated ankle and hip trajectory of Archie

In addition the above explanation it is important to mention the ground impact leads to the large torques for the supporting leg mainly because: it occurs earlier than expected and at that time the leg was still pushing forward opposing to the ground reaction forces. To help the waist recovering the original height, the knee in the balance leg slightly bends from its landing position while the ankle compensates by slightly extending its foot. Calculated leg joints-angle trajectory (ankle frontal and lateral, knee, hip lateral) based on the above selected values and walking parameters (stride=50 cm) which lead to avoid falling of Archie are depicted in Fig. 5.8, 5.9, 5.10 and 5.11.

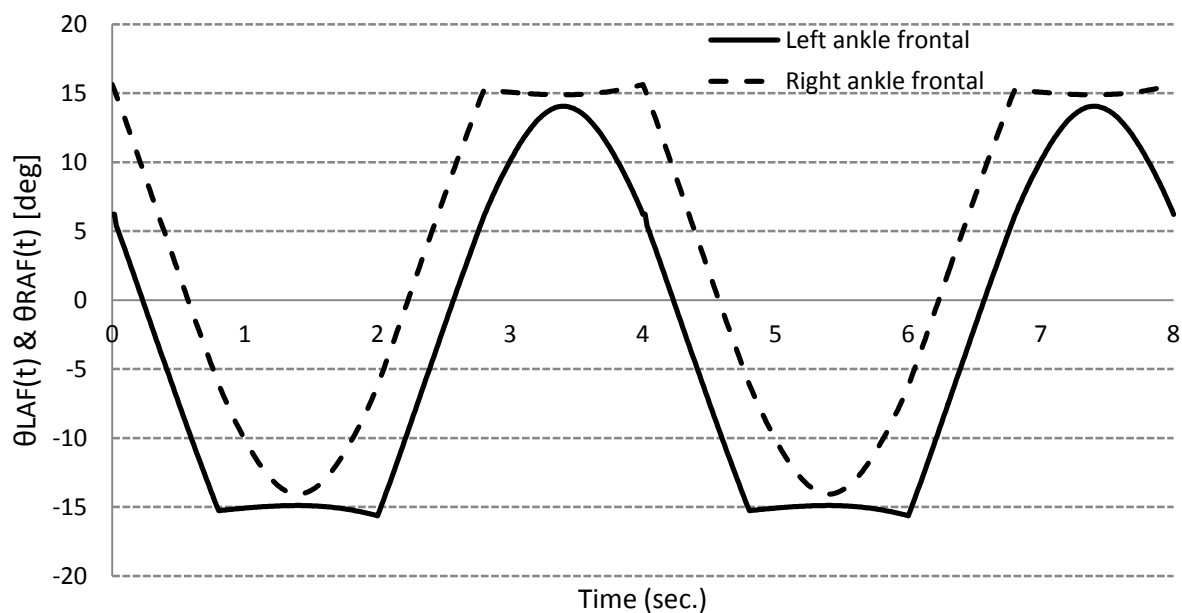


Fig. 5.8 Calculated left vs. right ankle frontal joint-angle trajectory

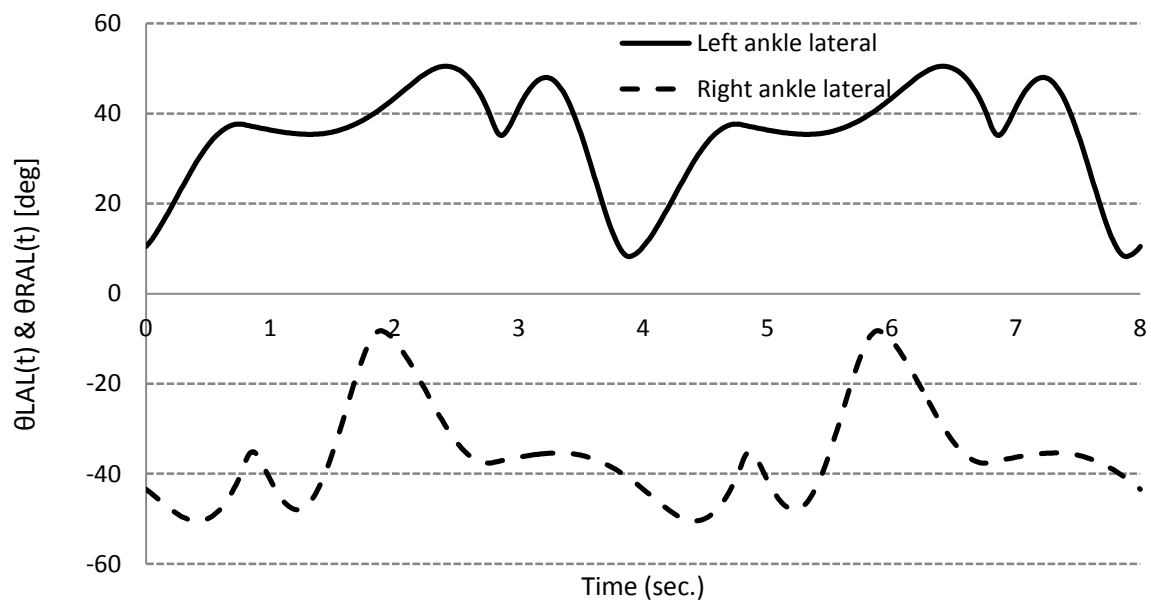


Fig. 5.9 Calculated left vs. right ankle lateral joint-angle trajectory

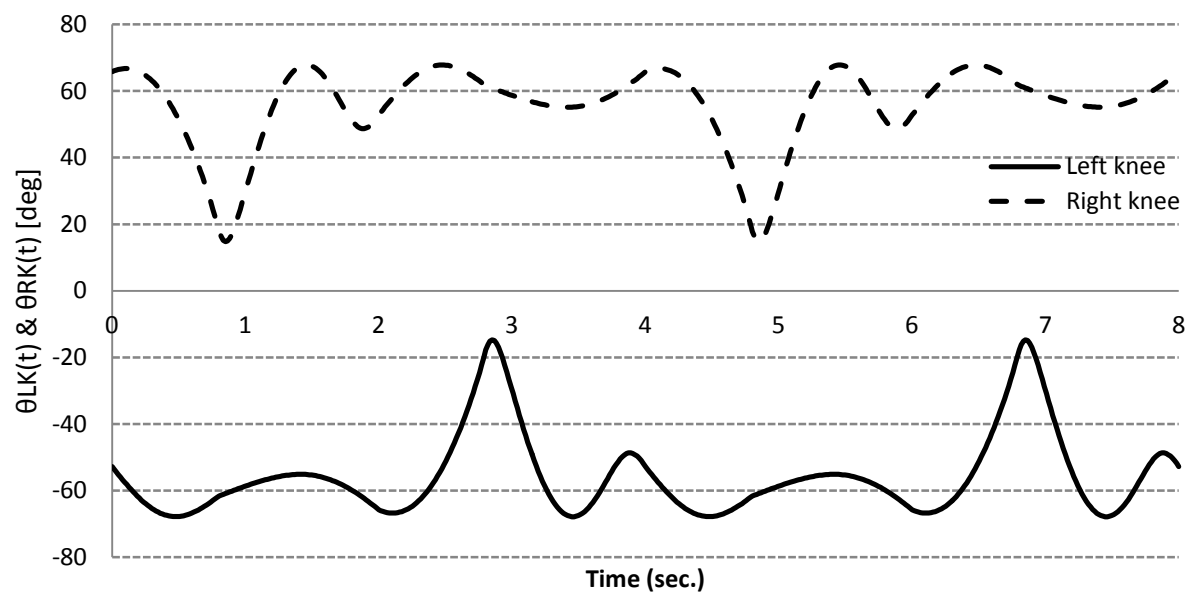


Fig. 5.10 Calculated left vs. right knee joint- angle trajectory

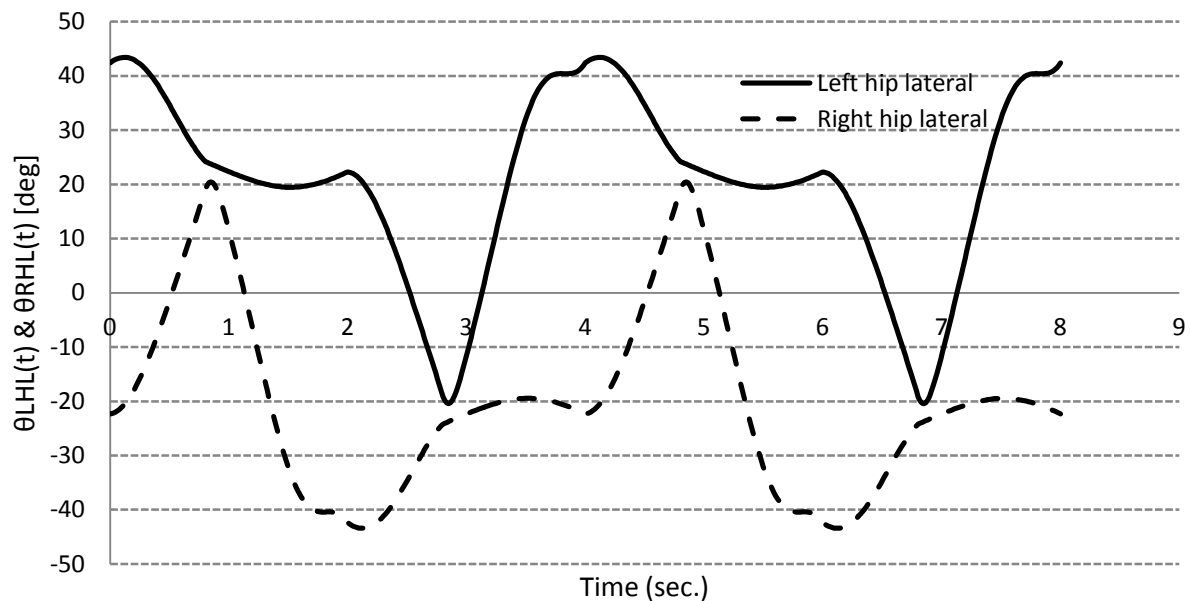


Fig. 5.11 Calculated left vs. right hip lateral joint-angle trajectory

5.4 Drive Motion Design Method

The motion control of biped humanoid robot Archie is combined offline gait programming and online movement modification. The data file of key parameters are generated by offline gait programming and obtain the final trajectory by movement modification. The position control system of Archie includes: central control computer (motion control program), servo motors (brushless motors), drive controller (Elmo), sensor system (encoder).

5.4.1 Servo Drive

The term servo can be applied to systems other than a servo motor; systems that use a feedback mechanism such as an encoder or other feedback device to control the motion parameters. Typically when the term servo is used it applies to a 'Servo Motor' but is also used as a general control term, meaning that a feedback loop is used to position an item.

The type of the servo drive used in Archie's joint is series of digital servo drives that are designed to deliver “the highest magnitude of power and intelligence”. While it is light and highly compact package and the drive operates on DC power. It is a PCB mounted device that enables efficient and cost saving implementation.

5.4.2 Drive Reference Signal Generation Method

The position reference signal is generated by the following components:

- Drive reference generator
- External position reference generator
- Stop manager

a) Drive Motion Reference Generator Approach

The position reference generator are sent in a configurable number of consecutive sub frames, which could be just one sub frame or as many as 12 sub frames. Utilization of more time-frequency resources within a sub frame by the position reference generator can improve the quality of measurements compared to the use of only the basic cell-specific reference signals.

In this part, the set/get commands which are categorized into groups of related orders will be discussed. Each group is presented in table and commands/signal are listed with basic description. The motor driver of Archie joints supports following modes of references:

- PTP (Point-To-Point): This mode specifies the position in which the motor is to stand, and the limits subject to which the trajectory to the target should be designed.
- PT (Position/Time): This mode specifies a set of points to be visited at fixed points of time. The driver interpolates a third-order polynomial between the user's points. The speeds at the user points are selected in order to form a smooth motion trajectory. Using PT, complex motions are easily designed. PT data may be transferred to the drive online using an efficient CAN message network, so that infinite PT motions are possible.
- PVT (Position/Velocity/Time): It specifies a set of motion points, each of them includes a position, speed and time at which the position should be visited. PVT mode allows absolute time specification so that several drives may compose a fully-synchronized motion. PVT data can be transferred to the drive online using an efficient CAN message network, so that infinite PVT motions are possible.

5.5 New Designed Drive Motion Command/Program Structure

The commands are used to instruct the drive to perform a sequence. Each command has its own value to be executed and affect the joint. The Archie's joints (motor, encoder & motor driver) reply to these commands according to their specifications, which is tuned already.

5.5.1 Single Joint Motion Program (Pattern Design)

There is a set of commands that leads to a single joint motion (Fig. 5.12). The primary set of messages for single joint motion are respectively as follows:

- NMT,
- MO (=0/1),
- PA/PR ,
- BG,



Fig. 5.12 Single joint motion pattern

a) Communication Control Command Structure

NMT command is used to control the communication state of the servo drive and to broadcast messages to all other connected joints (Table 5.5).

Table 5.5 NMT command structure

ID (HEX)		Mode		RTR		DLC	
0		0		0		8	
D1	D2	D3	D4	D5	D6	D7	D8
1	0	0	0	0	0	0	0

b) Motor Drive Command Structure

The MO command enable and disable the motor power.

- MO=0

This is the default state of the drive. In this mode, No current flows in the motor and the joint drive can run the several demands that are impossible when the motor is on such as :

- ✓ Download the new firmware and programs
 - ✓ Save the parameters in the memory
 - ✓ Boot on power up
 - ✓ Modify setup data that cannot be modified like unit mode
- MO=1

When the motor is enables, the drive reinitializes the internal parameters and motion drive. Basically the MO=1 is the operative state of the joint drive. Therefore drive is ready to driving the motor and activating and executing the designed programmed motion. In table 5.6 the MO command structure is described.

Table 5.6 MO command structure

ID (HEX)		Mode		RTR		DLC	
303		0		0		8	
D1	D2	D3	D4	D5	D6	D7	D8
4D	4F	0	0	1	0	0	0

c) Absolute Position Command Structure

Absolute position command designate clearly that the next command will be a PTP (point to point) and defines the target position for the next PTP motion (Table 5.7).

d) Relative Position Command Structure

Relative position may be applied in any active motion mode; it is activated and applies changes to the PA setting only after the next BG executed.

- If PTP motion is already active, PA will be increased by PR value and become the new position target.

- If the PTP motion is not active and now is activated by PR, PA will be set to new value, (PA= position command + PR).

Table 5.7 PA/PR command format

ID (HEX)		Mode		RTR		DLC	
303		0		0		8	
D1	D2	D3	D4	D5	D6	D7	D8
50	52	0	0	1	0	0	0

The new PA value causes PR to become 0. In order to make sequential PTP movements of the same size, the PR command is utilized. Subsequent PR settings may cause the PA value to be changed.

e) *Begin Motion Command Structure*

Each single joint motion is initiated by BG. This command immediately starts the next programmed motion.

- In software speed mode, BG activates the latest jogging motion (JV), and also the new smooth factor (SF), acceleration (AC) and deceleration (DC).
- In stepper or position mode (UM=3, 4 or 5), BG starts the latest position mode programmed: a point-to-point motion (PA), a jogging motion or any type of tabulated motion (PVT or PT).

Each motion mode starts with its entire set of parameters. For example, starting a point to-point motion activates the present of acceleration (AC), deceleration (DC), smooth factor (SF) and speed (SP). The BG command may be used to modify the parameters of the present mode, and not only to program new modes (Table 5.8). For example, a BG command in point-to-point mode modifies the active AC parameters (and all other active motion parameters) with its last programmed value.

Table 5.8 BG command specifications

Type	Command, no values
Source	CAN open
Restriction	MO=1
Activation	Immediate
Unit modes	2,3,4,5

BG command format which is used to start joint motion in Archie presented in Table 5.9:

Table 5.9 BG command format (for ID no. 3)

ID (HEX)		Mode		RTR		DLC	
303		0		0		8	
D1	D2	D3	D4	D5	D6	D7	D8
42	47	0	0	1	0	0	0

ID number refers to Joint ID which is already assigned to each joint. D1,D2, and D5 are input values for starting the next program motion.

5.5.2 Multiple Joints Motion Program (Pattern Design)

The most important issue in walking process is how to synchronize motion of all joints such that end-effector (Ankle frontal joint) can track desired trajectory properly, hence the problem of multiple joints motion is formulated (Dezfouli & Mohamadi Danilai, 2012). The set of commands are designed and collected to let the robot moves all joints together. These messages ordered as follows (Fig. 5.13):

- NMT,
- MP[1],
- MP[2],
- MP[3],
- MP[4],
- QP[X],
- PT=1,
- MO=1,
- BG

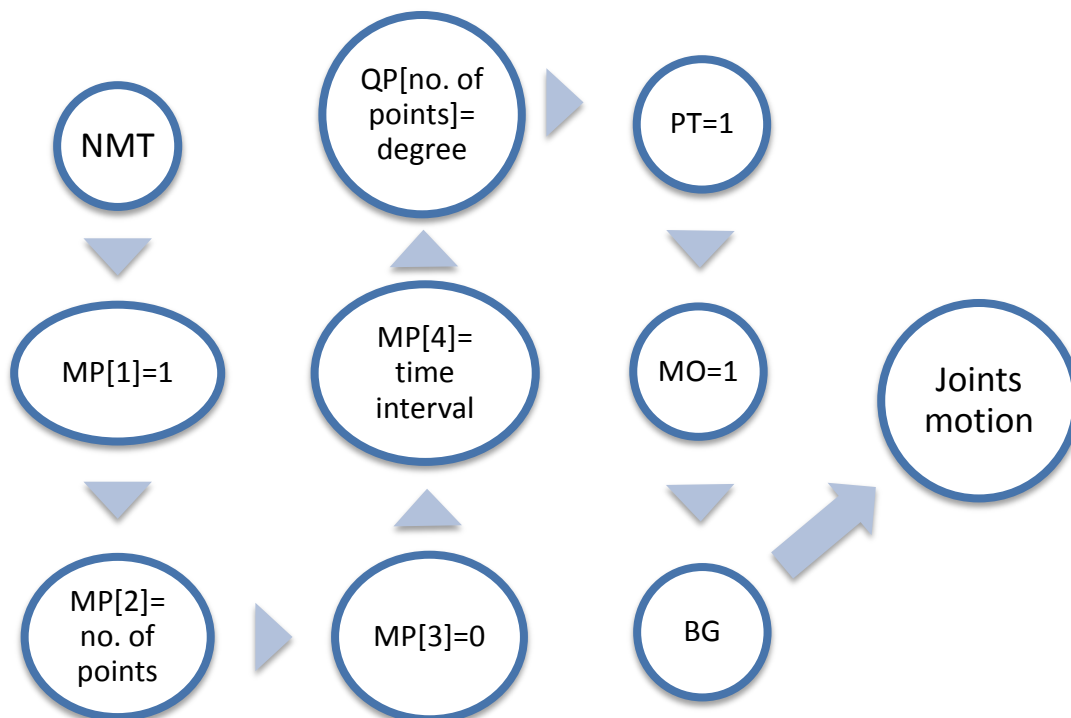


Fig. 5.13 Multiple joints motion commands order structure

5.5.3 Point-to-Point Method

This method is used in motion controller program (ver. 1.1). In PTP motion mode, the motor moves from its present position to a final point at zero speed, and stays there. The trajectory to the final point is calculated based on speed, acceleration and deceleration limits, as set by

the SP, AC, DC and parameters respectively (www.elmomc.com, 2013). The PA command plays more than one role. PA=n specifies that the next BG will start a PTP motion. In addition, it specifies the target of the next PTP move. In PTP mode, a BG command performs the following:

PA=PA+PR,
Go to PA where:

PA is the absolute position target. PR is the relative position target, enabling the specification of an incremental move or a series of incremental moves. PR is reset to zero by the PA=n command. Therefore, PA=n; BG initiates a motion towards n. The value of PA is incremented automatically with PR every time a new motion is initiated. For example:

```
PA=0;  
BG;  
while(MS==0); PR=1000;  
BG;  
while(MS==0);  
BG;  
while(MS==0);  
BG
```

initiates four consecutive PTP motions, targeted at 0, 1000, 2000 and 3000 counts respectively. PTP motions may be initiated any time, using the PA command, but not necessarily from a stationary state. The PTP decision flowchart, which made every position control cycle, is depicted in Fig. 5.14.

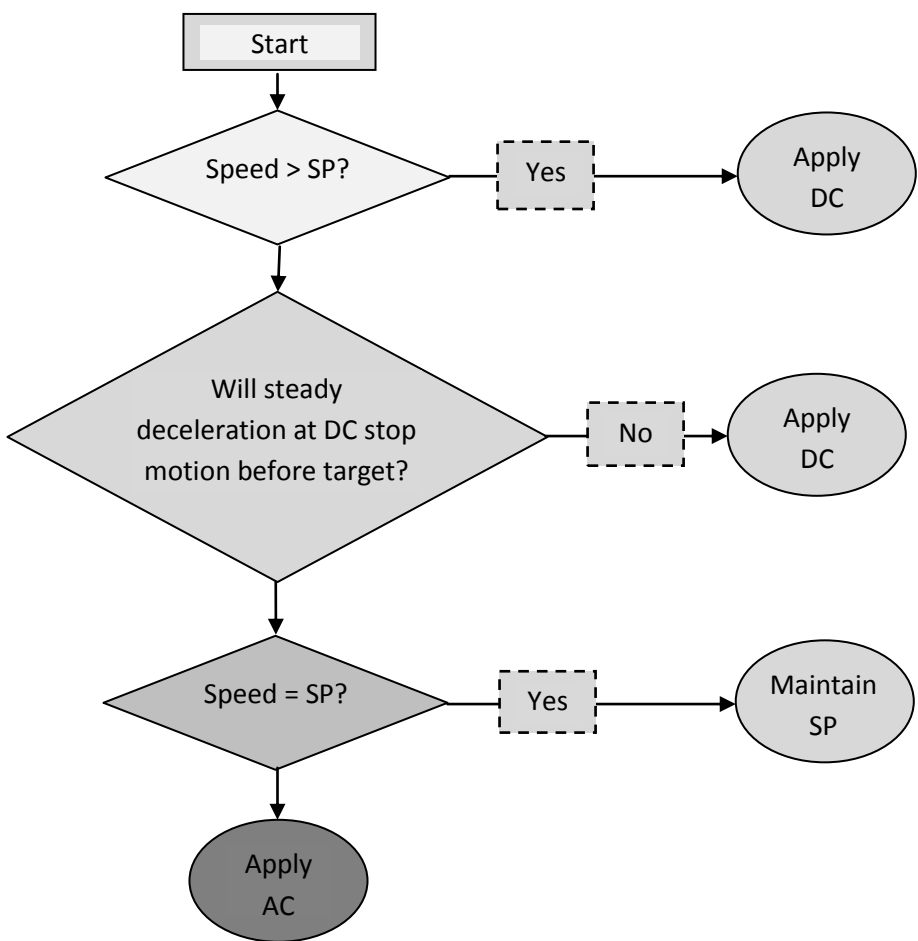


Fig. 5.14 PTP motion decision flowchart (www.elmomc.com, 2013)

5.6 New Motion Command Library

The communication network in the program lets the drive use the object dictionary (OD) method, which is the native CAN method. The motion library produces trajectories based on the PT or PVT mechanism. It implements a set of functions that calculate trajectories as PT or PVT table for vector motion. As a result the output (P,V,T) for requested desired trajectories that supports 2D or 3D vector motion will be generated. The PVT table is presenting 2 or 3 dimensional sequence of PVT points. Each PVT point is including:

- Position value,
- Velocity for this position,
- Time interval that is necessary to arrive from the current position to the position that defined by the next PVT point.

Single shape trajectories can be executed in one of the three modes that predefined by the value of the input parameter:

- max velocity,
- fixed time,
- fixed velocity.

The above designed pattern for single and multiple joints motion are embedded and implemented in new motion control program. The main difference is on the type of motion, which old motion control program (ver. 1.1 & 1.2) is based on the PTP motion and the new version (ver. 2.1) is based on the PT and PVT motion. The classes and/or objects which are participating in this designed pattern are described as follows (Fig. 5.15):

- Command/Signal : declares an interface for executing an operation
- Concrete Command (Calculator Command): defines a binding between a Receiver object and an action and implements Execute by invoking the corresponding operation(s) on Receiver
- Client (Command App.): creates a Concrete Command object and sets its receiver
- Invoker (User): asks the command to carry out the request
- Receiver (Calculator): knows how to perform the operations associated with carrying out the request.

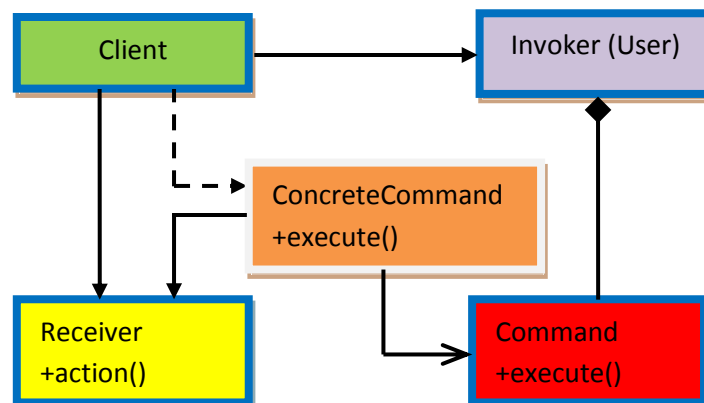


Fig. 5.15 Motor drive command (designed pattern)

Motion drive command in Archie decouples the object that invokes the operation from the one that knows how to perform it. To achieve this separation, the abstract base class is designed that maps a receiver (an object) with an action (a pointer to a member function). The base class contains an execute() method that simply calls the action on the receiver. Encapsulate a request as an object, thereby letting to parameterize clients with different requests, queue or log requests, and support undoable operation.

Sequences of Command objects can be assembled into composite (or macro) commands. A Command class holds some subset of the following: an object, a method to be applied to the object, and the arguments to be passed when the method is applied. The Command's "execute" method then causes the pieces to come together. The proposed motion commands implementation in C++ is presented in chapter 7.

5.6.1 Motion Modes Changing Method

The ST command can be used for stopping at any time. The drive will decelerate the position or velocity reference until it reaches a full stop. The drive will calculate the path to be followed so that the desired speed or position is reached, subject to the acceleration limits. PTP motion can even be initiated during PVT or PT motion. Upon switching from interpolated motion to PTP, the PTP will start unsmoothed, with smoothing gradually building up, until after milliseconds, the motion is fully smoothed.

Care is also required when switching to tabulated (PT and PVT) motion modes. These modes should be started only when the motor is at a complete stop (MS=0 or MS=1), at the exact position specified as the first point for the PT or PVT. If the motor is not totally stopped at the correct start position, the software reference will jump.

5.6.2 Position - Time (PT) Method

In a PT motion, the user specifies a sequence of absolute positions with equal time spaces to be visited by the drive. The time space must be an integer multiple of the drive sampling time. Between the user-specified positions, the drive interpolates smooth motion. The position specifications are absolute. PT implements a third-order interpolation between the position data points provided by the user.

Let $T = m * T_s$

where:

T_s is the sampling time of the position controller. T is the sampling time of the PT trajectory. m (system parameter MP[4]) is the integer parameter that relates T_s and T . For $m=1$, no interpolation is required. For $m>1$, there are certain sampling instances of the position controller for which the path command must be interpolated, using a third order polynomial interpolation. For each motion interval, four requirements must be satisfied:

- Start position
- End position
- Start speed
- End speed

These four requirements exactly suffice for solving the third-order interpolating polynomial.

5.6.3 PT Table Design (QP Vector)

The vector QP[N] defines the position points for PT motion. Each element of the vector defines the position at a given time. The QP vector has 1024 elements, and can therefore specify up to 1023 consecutive PT motion segments, or 1024 PT motion segments in cyclical mode.

5.6.4 Position-Time Motion Scope

In PT mode, the drive manages a “read pointer” for the QP[N] vector. When the read pointer is N, the present motion segment starts at position QP[N] and ends at QP(N+1). After MP[4] control sampling times, the drive increments the read pointer to N+1, and reads QP[N+2] to calculate the parameters of the next motion segment. The entire PT table need not be used for a given motion. The parameters of a PT motion are summarized in table 5.10.

Table 5.10 PT motion parameters

Parameters	Use	Comment
MP[1]	Lowest valid element for QP vector	
MP[2]	Highest valid row of QP vector	
MP[3]	0: motion stops if read pointer reaches MP[2]. 1: motion continues when read pointer reaches MP[2]. The next row of the table is MP[1].	Cyclical behaviour definition
MP[4]	Number of controller sampling times in each PT motion segment.	

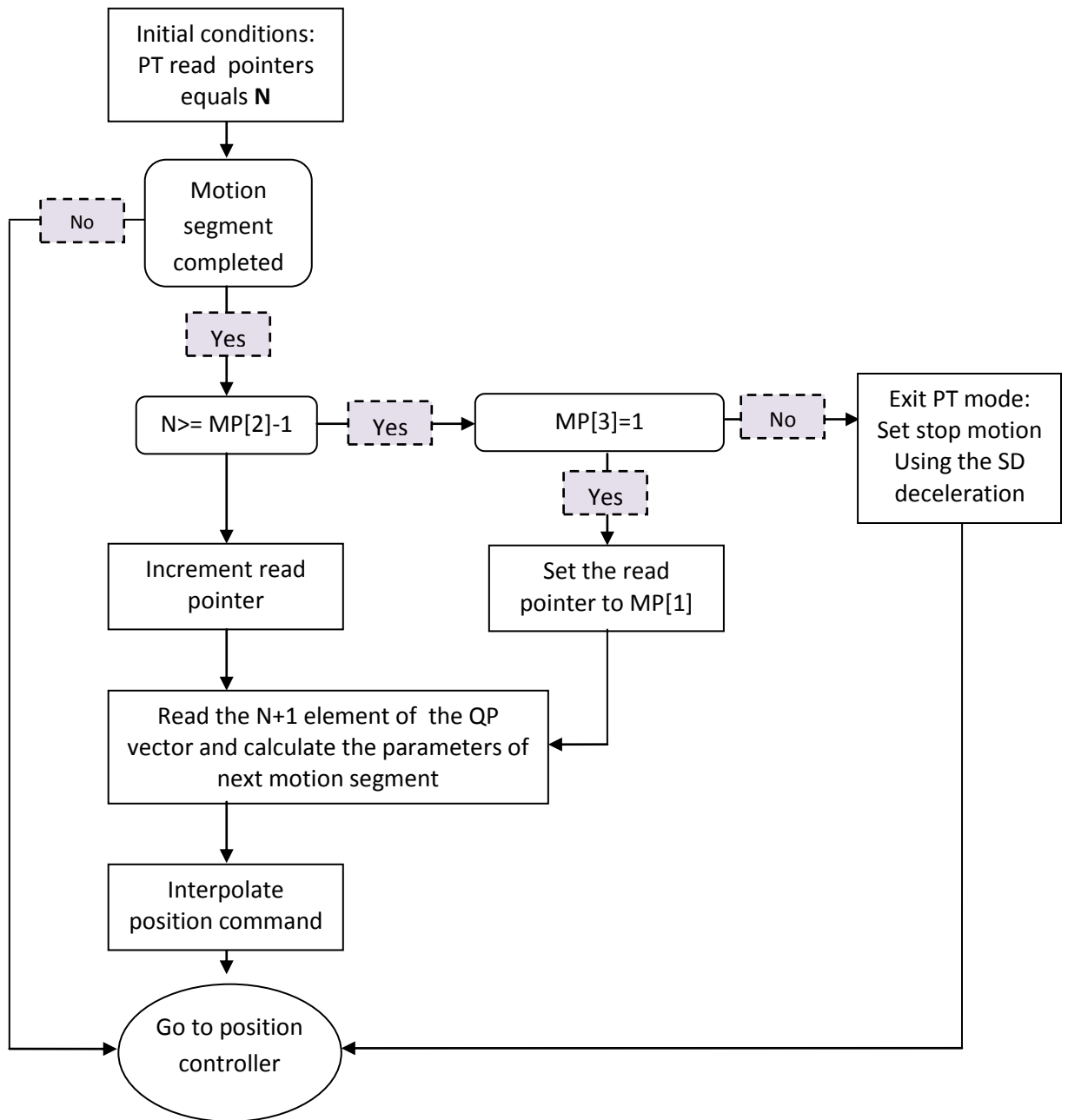


Fig. 5.16 PT motion flowchart

The PT table allows for the performance of both pre-designed and online motion plans by writing the QP vector while PT motion is executing. The online motion design ability is limited by the speed of the communication interface.

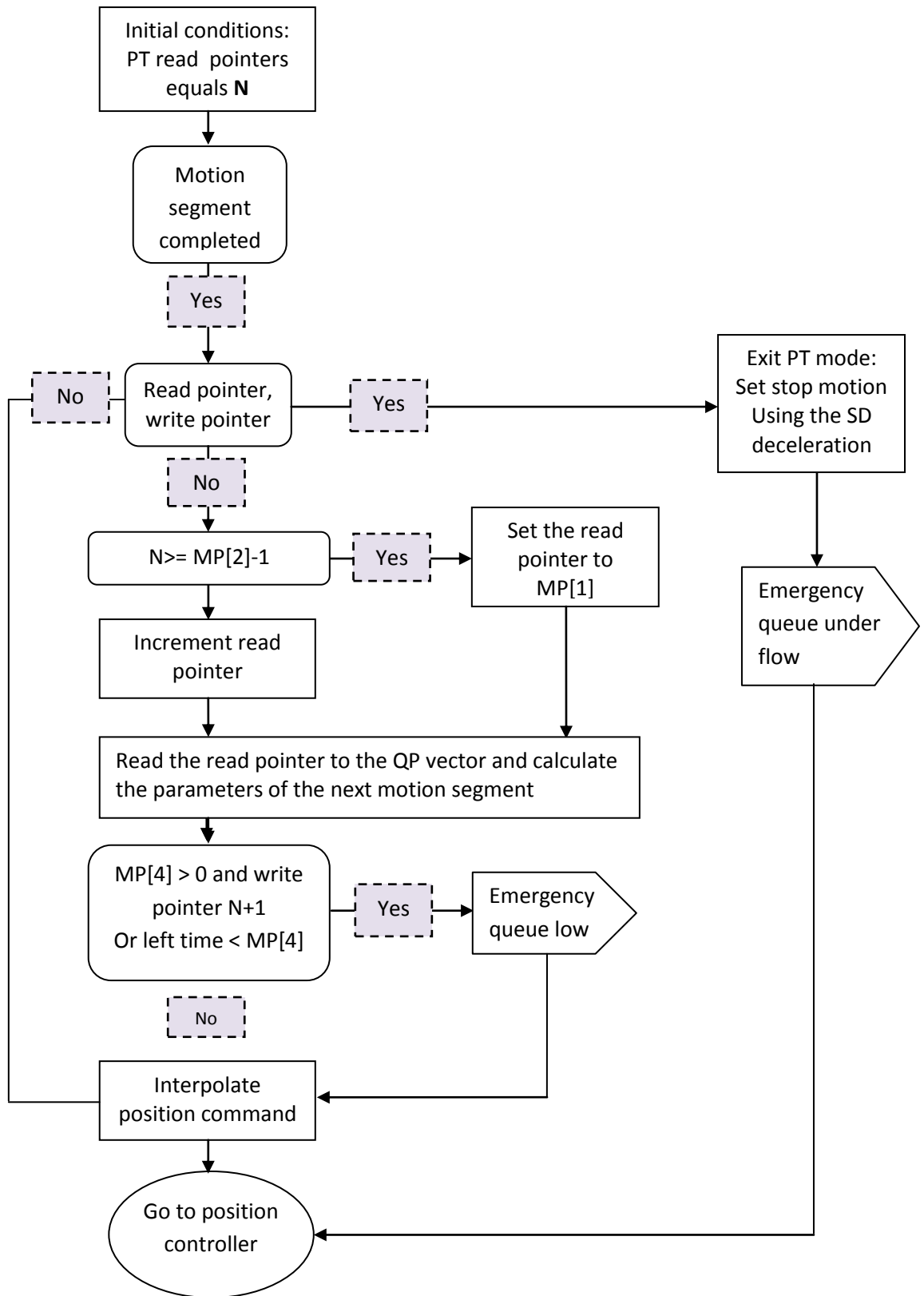


Fig. 5.17 PT auto increment mode flowchart

The proposed implemented communication network allows much faster PT programming, by packing two position points into one PDO communication packet. For easy synchronization with the host, the drive may be programmed to send the PT read and write pointers

continuously to the host as a synchronous PDO, or to send an emergency object whenever the number of yet unexecuted motion segments falls below a given threshold. The PT motion algorithm (basic logic) and PT auto increment mode are depicted in Fig. 5.16, and Fig. 5.17.

The PT motion terminates when one of the following occurs:

- The motor is shut down, either by programming MO=0 or by an exception.
- Another mode of motion is set active; by programming PA=x; BG, for example. In this case, the new motion command executes immediately, without having to explicitly terminate the PT mode.
- The PT motion manager runs out of data. This occurs when the read pointer reaches MP[2] and MP[3] is zero. This may also occur in auto-increment mode (CAN communications only) if the read pointer reaches the write pointer. In that case, the PT motion is stopped immediately, using the SD deceleration. Note that if the last programmed PT speed is zero, the PT motion terminates tidily.

5.7 Stop Command Mechanism

In order to protect the motor under the operation there is need to design the stop manager command. It helps to avoid the damage or malfunction of the motor. The stop command manager performs the following functions:

- Brings the motor to a stop upon a software or hardware ST command
- Brings the motor to a stop when the speed demand is positive and FLS (Forward Limit Switch) is active
- Brings the motor to a stop when the speed demand is negative and RLS (Reverse Limit Switch) is active
- Prevents acceleration or deceleration beyond the motor torque limits

On the other hand stop manager prevents the speed controller command from changing suddenly by limiting the rate of reference change to SD (Maximum motor acceleration/deceleration) counts/second. When the stop manager stops the motor due to a switch action, the reference generator is replaced by a zero command at the input to the stop manager.

The stop manager uses the SD parameter to decelerate the motor command from the output of the reference generator to a complete stop. When the switch action terminates (a Stop switch is released, for example), the stop manager uses the SD acceleration to recover the speed command to the output of the reference generator.

5.7.1 Stop Manager Structure (Internal Elements)

The stop manager plays an important role to avoid any damages, the embedded stop command in designed motion creating system is depicted in Fig. 5.18.

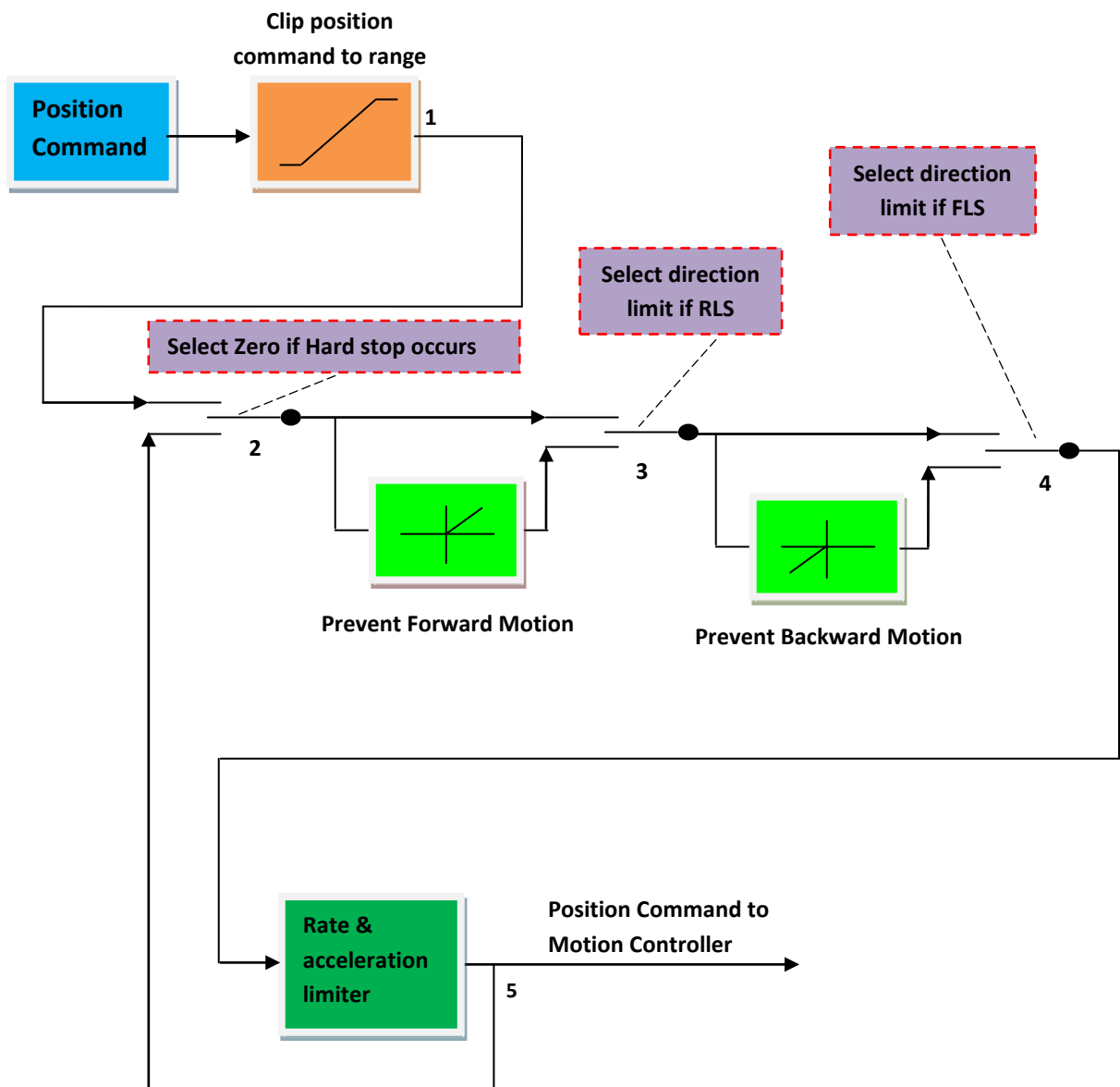


Fig. 5.18 Stop manager internal elements

Each segment which is signed by number in Fig. 5.18 is described in the following:

- Position Command Clipping (No. 1)

The position command is clipped to the following values:

The clipping is necessary because, in some circumstances (explained previously), the sum of the software command and the external command, or even the software command alone, may exceed the command limits.

- Hard Stop (No. 2)

This block stops the desired position reference to its present position if a Hard Stop switch is sensed.

- Reverse Limit Switch (No. 3)

This block stops the desired position reference to its present position if an RLS switch is sensed, and if the output of the position reference generator is less than the controller position command.

- Forward Limit Switch (No. 4)

It stops the desired position reference to its present position if an FLS switch is sensed, and if the output of the position reference generator is greater than the controller position command. Rate and Acceleration (No. 5) This block limits the speed and acceleration of the controller position command. The block limits both the acceleration and the deceleration to the SD parameter, and the speed command (the derivative of the position command) is limited.

The rate and acceleration limiter block intervenes in the following situations:

- The position command to the controller experiences an abrupt change; for example, when a Hard Stop switch is sensed or when a Hard Stop switch is released.
- The reference generator tries to control the motor in the permitted position range, but at too great a speed.
- The position command moves towards its permitted boundary at a speed greater than can be braked until the boundary with SD acceleration.

The stop manager block has the following functions:

- Stops the motion upon sensing a Stop switch, or upon sensing an RLS or FLS limit switch.
- Protects against discontinuity in the controller command. A discontinuity may occur due to:
 - A switch that abruptly stops the motion.
 - An application error (an absolute motion mode such as PVT is started with unacceptable initial conditions).
- Limits the magnitude of the controller command to the maximum allowed range.

This is necessary because even if the software command is generated within the permitted limits and the external command is also within the permitted limits, their total value may exceed the permitted limits. The stop manager prevents the position reference generator from driving the motor to undesired positions. It does not affect the reference generator.

5.8 Motor Fault Diagnosis and Error Detection Approach

The motor failure protection is designed since it happens that Archie motor sometimes cannot rotate well, therefore motor is unable to complete a command to move, and the reasons may be because of:

- The motion sensor is faulty: The motor moves but motion is not detected. In this case, AC motors will generally stop, because the stator field will remain stationary.
- The motor is faulty or another mechanical failure is preventing the motor from moving.

- The controller filter is poorly tuned. In this case, the motor torque may oscillate wildly at high frequency, but the motor will barely move.

5.8.1 Motor Failure Protection Mechanism

According to the facts, it is estimated that most of the electric motor failures occur at the start up. Hence following physical conditions are recommended to be checked before voltage is applied to the Archie motor:

- Power supply under-voltage
- Power supply over-voltage
- Drive temperature too high
- Motor not connected to servo drive
- An active limit switch is programmed to shut down the servo drive

Archie motor drive (motion creating system) is designed in order to avoid starting if the voltage of the power supply is not within range, if the servo drive temperature is too high, or if an active switch prevents motor on. The MO command returns error code or the user can send MF command to detect the problem. If the voltage is in range and the temperature is not excessive, the drive will attempt to start the motor.

5.8.2 MF Command/Message Structure

As mentioned above it can be summarized that error conditions may cause the drive to automatically shut down the motor, at which time:

- The MO variable is set to zero.
- The MF variable is set to reflect the reason for shutdown.
- A flag in the status register (SR command) indicates that the motion has been aborted.
- The next “Motor enable” (MO=1) is permitted after 150 sample times (TS).

The MF variable may reveal the reason for motor shutdown even if the reason no longer exists. For example, if the power supply has too large an impedance, its voltage may drop in full load and the servo drive will be automatically shut down due to under-voltage (www.elmomc.com, 2013).

When the motor is shut down, the under-voltage disappears. On the other hand, if an over-voltage is generated due to an insufficient shunt, the over-voltage will disappear when the motor is shut down.

The over- or under-voltage conditions that caused the fault are captured in the MF variable.

Notes:

- MF is reset automatically every “Motor enable” (MO=1) when the fault is no longer valid.
- When the drive shuts down by exception, the motor continues to run on its own inertia unless brakes are used.

Summary

This chapter is organized in the first part to shortly describes:

- New walking pattern principle of Archie
- New approach to formulate constraints of a foot and torso trajectory and generating the foot and hip trajectory by third spline interpolation.
- Problem of smooth hip motion formulation with the largest stability margin using x_{ed} and x_{sd} , and derive the hip trajectory by iterative computation.

In the second part the unique principle PT and PVT motion design and implementation for Archie locomotion is discussed and compared to the old method point to point (PTP) method which is utilized already.

New drive motion library command is considered for single or multiple joints motion. It is responsible to generate trajectories based on the PT or PVT motion. It implements a set of functions that calculate trajectories as PT or PVT table for vector motion. Hence, Archie new motion creating system is designed in order to perform the PT or PVT mode individually.

Chapter 6 New Motion Control Program and New Software Architecture Design

6.1 Introduction

This chapter aims to provide a comprehensive description of developments and improvements in motion controller program/software and relatively the new architecture design will be discussed. There are two different developed software for Archie locomotion. The version 1.2 is a latest developed form of version 1.1., in the case of graphical simulation which the upper body elements are added. The version 1.1. is designed by Stela H. Seol under the supervision of Prof. J. Baltes in University of Manitoba in 2010. We designed and created the latest version (ver. 2.1.) in order to improve motion control of robot by based on the PT motion by embedding new walking planning (as presented in chapter 5), and minimizing the computational efforts which devoted for generating smooth walking motion. Hence it is adopted a centralized system in order to construct a platform where user can easily develop and debug perception–action coupling control schemes. All of the sensor data, such as actuator encoder values, are directly available to the PC. Control commands are sent from the PC to the motor drivers directly. This requires an operating system in which many different cycle control loops can be executed concurrently (from 1ms servo loop to higher level trajectory generation and motion planning loop, which have cycle times of several seconds). Accordingly first part of this chapter is organized to explain and present the development in program ver. 1.2, and in the second part, the reasons, requirements for designing the new motion control program, basic architecture and the performance mode of the new software (ver. 2.1) are discussed.

The motion controller is the brain of the control system in robot. It is programmed to accomplish specific tasks such as walking. It plays the main control system role and provides the orders (commands) for synchronized motion of motors (joints).

6.2 Motion Controller Program/Software (ver. 1.1 & 1.2)

In this section, features of the modules, data flow and interaction between classes will be provided for motion control program (ver. 1.1 and 2.1). Information needed to configure the shell for an application (walking) and code's structure have been designed modularly, which means that there are several different compartments or “modules” which hold files that perform a specific function for the system. Examples of these files are robot joint, CAN bus communication, position, settings and so on. Each of these modules and their relationships to one another will be explained in the following.

The program architecture of a robot can be defined as the structures of the motion system which are comprised of software components; the externally visible properties of those components and the relationships between them. This structure illuminates the top level design decisions; including such things as how the system's interacting parts are composed, indicating where the main pathways of interaction are, and what the key properties of the

parts are. Moreover an architectural description includes sufficient information to allow high-level analysis and critical appraisal. Accordingly the developed software structure and internal interaction between modules are designed. Software interface will be described in order to answer possible questions of users or system software developers. The goal is to enable future users to develop different and complex simulations to achieve various results using graphs before implementing commands on the robot. Therefore the new embedded parts (upper body components) in GUI are presented.

6.2.1 Program/Software Architecture

Motion controller program is developed with Qt creator, a cross-platform integrated development environment (IDE). This section first justifies the use of the Qt platform for the development of the software and graphical user interface. Additionally a complete description of the different elements that constitute the graphical user interface in motion controller program (which is used to control and simulate Archie humanoid robot walking) will be provided.

Qt Creator is an integrated development environment that provides tools to design and develop applications with the Qt application framework (www.qt.digia.com, 2012). it is designed for developing applications and user interfaces once and deploying them across several desktop and mobile operating systems. Qt Creator is provided with tools for accomplishing new tasks throughout the whole application development life-cycle, from creating a project to deploying the application on the target platforms (Fig. 6.1). The motion controller program ver. 1.2 platform and tools are presented as follows:

a) Creating Program (Project)

The first step in writing motion controller program is to create the project. The capabilities that lead to create a project are:

- ❖ Group files together
- ❖ Add custom build steps
- ❖ Include forms and resource files
- ❖ Specify settings for running applications
- ❖ To generate project with required headers, source files, user interface descriptions and project files, as defined by the wizard.

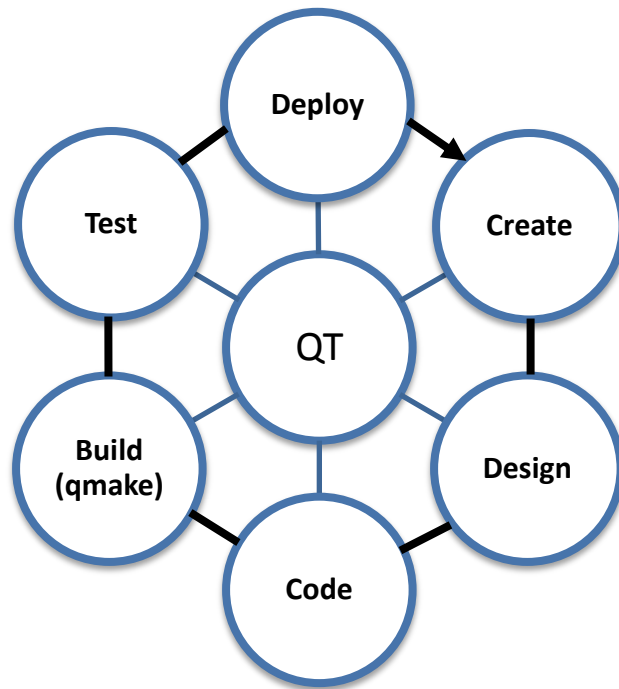


Fig. 6.1 Necessary QT creator tools for motion controller program

b) GUI Design

A motion graphical user interface is developed to provide a communication link and command interface between off-board computations to generate the retargeted joint commands and the on-board real time control. It provides a comprehensive way to give motion commands to the robot, without having the user to care about issues such as delays, synchronization, or on-board control for maintaining balance, and collapse avoidance. Such an interface is desirable since the real-time implementation requires synchronization between critical control processes that may not be satisfied dependably with a network connection. For safe, effective and rapid control of the robot, the robot must operate as an embodiment and extension of the operator. In other words, the operator must cognitively place themselves in the same position as the robot. But two barriers exist in achieving this goal.

The first barrier comes from designing a suitable mapping between what a human considers as intuitive movement and translating it to sensible movements in the robot.

The second barrier is quality level of sensing and perception. The operator is not in the same place as the robot and the sensors on the robot may not match those that a human is used to. In order to overcome this barrier, familiar sensors must be presented in a way that provides the operator with good situational awareness and allows them to form a good mental model of the environment.

Teleoperation remains an important part of interaction with robot. Various degrees of autonomy are starting to become possible however, some form of direct operating interface will be required and even when it becomes possible, alternative forms of user interfaces will

be needed for human-robot cooperation. The design of professional user interfaces for robot awareness and control, are critical to the success of Archie smooth walking.

Designing tool consists of a rich set of user interface elements, a declarative language for describing user interfaces, and a language runtime. Large high-resolution screens, complete input, and significant graphics power are main elements considered in design of the new developed program GUI.

c) Code editing

Code editor in QT (ver. 1.2. platform) is different from text editor, it understands C++ language not just as plain text (www.qt-project.org, 2013), It allows to :

- Write well formatted code
- Anticipate writing and complete the code
- Display inline error and warning messages
- Semantically navigate to classes, functions, and symbols
- Use provided context-sensitive help on classes, functions, and symbols
- Rename symbols in an intelligent way, so that other symbols with the same name that belong to other scopes are not renamed
- Present the locations in code where a function is declared or called

d) Building-qmake

Build settings cause to quickly switch between build targets. It is integrated with cross-platform systems for build automation: qmake. In addition, it is possible to import generic projects that do not use qmake.

e) Testing

One of the advantages is that there are code analysis tools to detect memory leaks, profile cache usage, and profile applications. There is a platform to test applications that are intended for special mobile devices in the simulator, but it is necessary to test the applications on real devices. The mobile device should be connected to PC and then start debugging processes running on the devices.

f) Debugging

In the Debugging mode there is a possibility to inspect the state of application while debugging. Interacting with the debugger is in several ways, including the following processes:

- Going through a program, line-by-line or instruction-by-instruction
- Interrupt running programs, setting breakpoints
- Examine the contents of the call stack
- Examine and modify registers and memory contents of the debugged program
- Examine and modify registers and memory contents of local and global variables
- Examine the list of loaded shared libraries
- Create snapshots of the current state of the debugged program and re-examine them later

It provides high level functions to create and customize a graphical user interface as well as all of the basic components that one could want to use for a graphical interface. Powerful tools to create our own components from low level functions (painting, events) are also provided. The proposed library has the advantage of providing this concept of communication using signals and slots. This library is used to simply connect or disconnect classes together through very clear interfaces. This allows a flexibility in the code and between the classes. Everything can be built independently and then linked at run time. This notion has been particularly used to link the variables to the graphical user interface (Fig. 6.2).

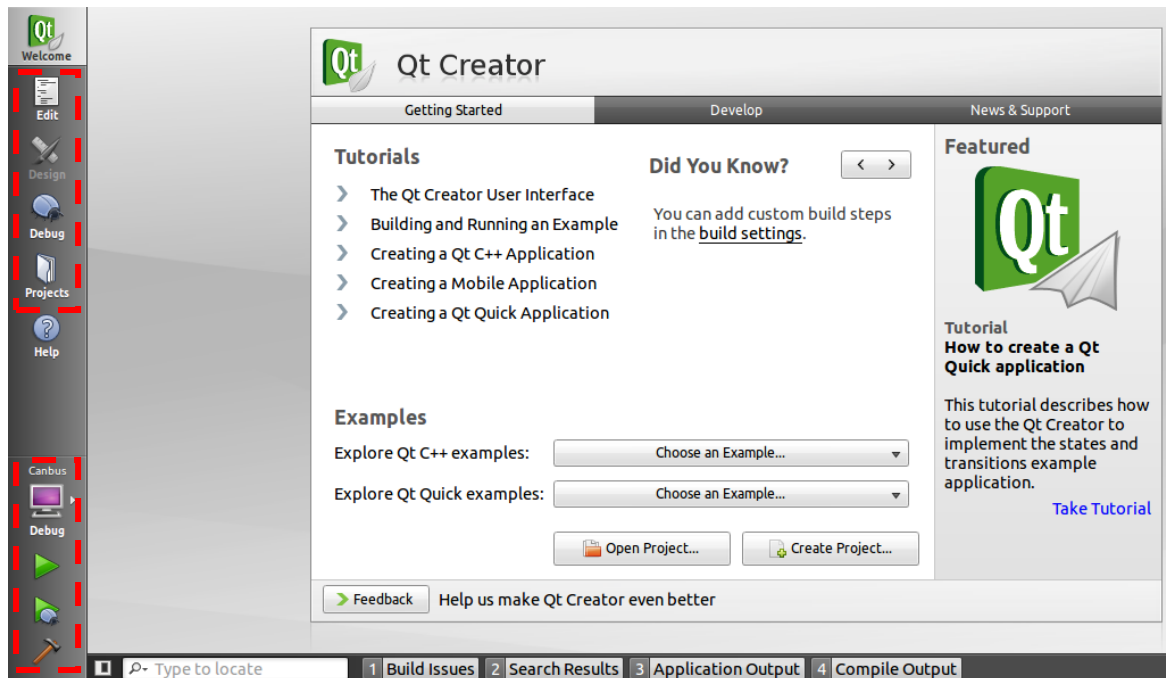


Fig. 6.2 Qt creator platform

The main advantages of this platform which makes it different from others are:

- It has rich libraries for graphical user interfaces.
- It is possible to program in C++ language and the software runs on Linux System operative.
- It uses Object Oriented Programming. It allows flexibility between classes.
- It is based on the concept of Widgets, Signals, Slots and Events.
- The developer can easily add, remove and update the modules (scalability).

6.2.2 Class Creating (in C++)

The C++ class wizard creates a C++ header and source file for a new class that can add to a C++ project. Class name, base class, and header and source files for the class are specified. In Fig. 6.3. created classes in motion controller program are presented. Up to now the main platform and tools to prepare the infrastructure for developing the program are described. From now on the developed version of motion controller program structure will be discussed and compared to realize the new capabilities of the developed program.

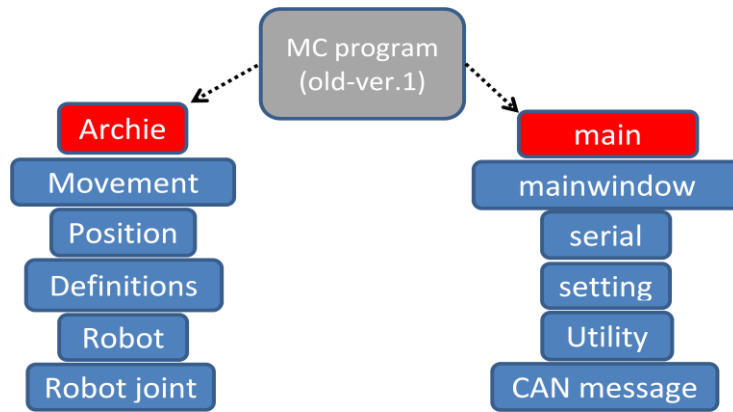


Fig. 6.3 Different created classes in motion controller program

6.3 Developed Program Structure Description (ver. 1.2)

In this part the structure and new interaction forms including GUI improvements and implementations alongside the edited codes are described. The program uses multiple files-.cpp/.h- to group functions and classes into modules. The structure of program has been modified and redesigned to be reasonably competitive with the old version. These modification features are listed as follows:

- The motion controller codes are modified in facile format to be easy understandable and accordingly they can be located, modified or replaced with functions so easy.
- Compilation time sharply reduced because only the changed modules need to get recompiled.
- The modules could work independently but they are related to each other; if one changes the interface of one module, the other modules will need to get recompiled. In this case the error ratio of program compilation is decreased.
- The next module needs to know the functions or classes provided by previous modules (this is included in the headers files). Every classes, functions or objects are only defined once. Redundancy is removed and it is expedited the performance of the program.

The program structure is very important to be well arranged in individual modules. Each module consists of two files: a .cpp file (source code file) and a.h file where functions and classes are declared. Moreover it is strongly needed to define a common interface between all the modules. In order to have scrutiny view of the actual process between improved modules and their functions, the main code features are listed as follows:

- Setting:

This module is used to load and save controller settings in text file (.txt).

- Defs:

This file defines different structures such as kinematic_option or joint_type which will be used in the following modules. Moreover, within the defs. file the different parameters used in the graphical user interface are assigned to predetermined values and names.

- Robot:

One of the most effective module in version 1.2, is robot class, since it collects many functions related to the kinematics of the robot and the motion controller graphs, which will be used in the simulation. The applied parameters and elements are listed as follows:

- ✓ Inverse kinematics. These variables are used to calculate and modify (update) angles for each movement.
- ✓ Collecting robot information for Motion Editor. These provide the robot name and the background image.
- ✓ Drawing components. These display the yellow point on frontal, lateral and step graph.
- ✓ Display the items in Motion Editor.

- Main window:

This module is a modern action-based class including window, toolbar, menu, and docking architecture. It can be called as one of the main changed and edited module for motion controller program.

- Serial:

This class is related to adjusting the baud rate for communication and choosing type of terminal that will be connected to the PC. It gives the permission to open the modem device for reading and writing as well as check the communication in case of an error. When the communication is interrupted, the program advises to change the port. Different range of baud rates are embedded to let the user transfer data in different speed rate.

- Utility:

The functions of this module do not perform specific tasks, but instead support other modules; these functions are used in different modules. For example it makes a variable in bound of range, converts radians to degree and vice versa.

- Robotjoint:

The variables and functions related with the position and angle of the robot humanoid joints are used in this module. They are listed below:

- ✓ D.H. representation: these four variables: length, offset, twist, and angle - are used to build the matrix 4x4 for D.H. Kinematics.

- ✓ Declaring to the actual position of the servo. It declares the variables: initial degree, maximum and minimum degree, one tick of a turn is one degree. It also checks if the motion of each joint occurs within the range allowed. On the other hand the user can limit the rotation angle of the join to avoid any damages on the hardware of the robot via this module.
- ✓ Each joint type. For example name of joint defining, and selecting the Id of driver.
- ✓ The motion editor. X and Y co-ordinates, as well as width and height of steps.

These variables and calculation are applied to:

- ✓ Collect, modify and set joint angles
- ✓ Set exact driver positions
- ✓ Prepare information regarding various joints before starting motion
- ✓ Collect variables for the DH representation matrix
- ✓ Collect motion editor inputs

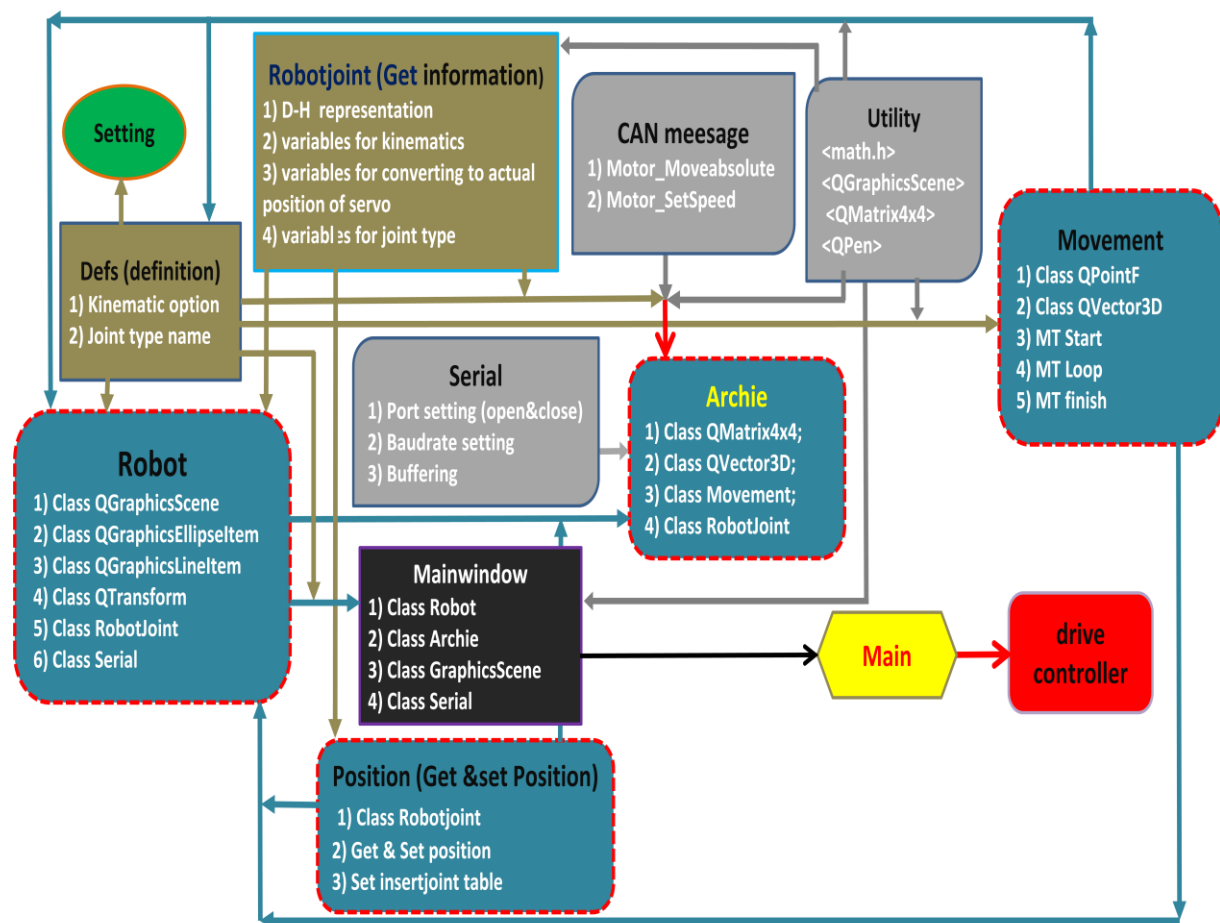


Fig. 6.4 Data stream between modules in motion controller

▪ Canbus:

This module task is to establish the communication network which is specifically devoted to communication between the PC and USB to CAN converter.

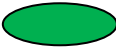



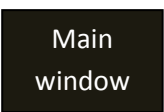


- Position:

This class runs functions and assists to reduce the calculation of the joint position. It can set and reset the initial joints position to let robot standing in the proper form.

- Movement:

In this module, a specific variable for kinematic movement, is defined. Pointers-Qvector3D- are used to locate the position of the end effectors – left/right fingers and left/right toes – and to calculate the motion path. Also pointers – QpointF – are used to get the trajectory of the yellow point in the graphical windows. In this file the new developments such as adding upper body joints and relatively their functions and calculation are embedded.

Table 6.1 Legend of the Fig. 6.4

Symbols & colours	Block description
	Parameters setting for graphical view in motion controller
	Kinematic & joint model option + presentation (Set &Get)
	Calculation & control (Transform. Matrix, Set Parameter)
	Communication network (serial port, CAN message)
	Mainwindow class contains all widgets and all classes where there is an access to all widgets placed in graphical and simulation interface (Update Scenes)
	Main is a specific widget that has things like a menu bar, tool bar and status bar built-in. This class is useful for the main application window to fit around the desired main UI
	Execution commands are situated in this class to let the robot play or stop

- Archie:

This is the main module because all functions from the last modules cooperate together to support and run this module to send final execution commands to robot.

Data streaming between these modules are depicted in Fig. 6.4. Accordingly the blocks definition and their functionality are classified in Table 6.1. It is noticeable that the main focus in development of old-version is on removing the internal errors, expediting the performance and adding the upper body joints simulator in internal structure and GUI.

6.3.1 Motion Controller Program GUI (ver. 1.1)

Graphical user interface in old motion controller program is divided in 3 main parts which can be presented in Fig. 6.5.

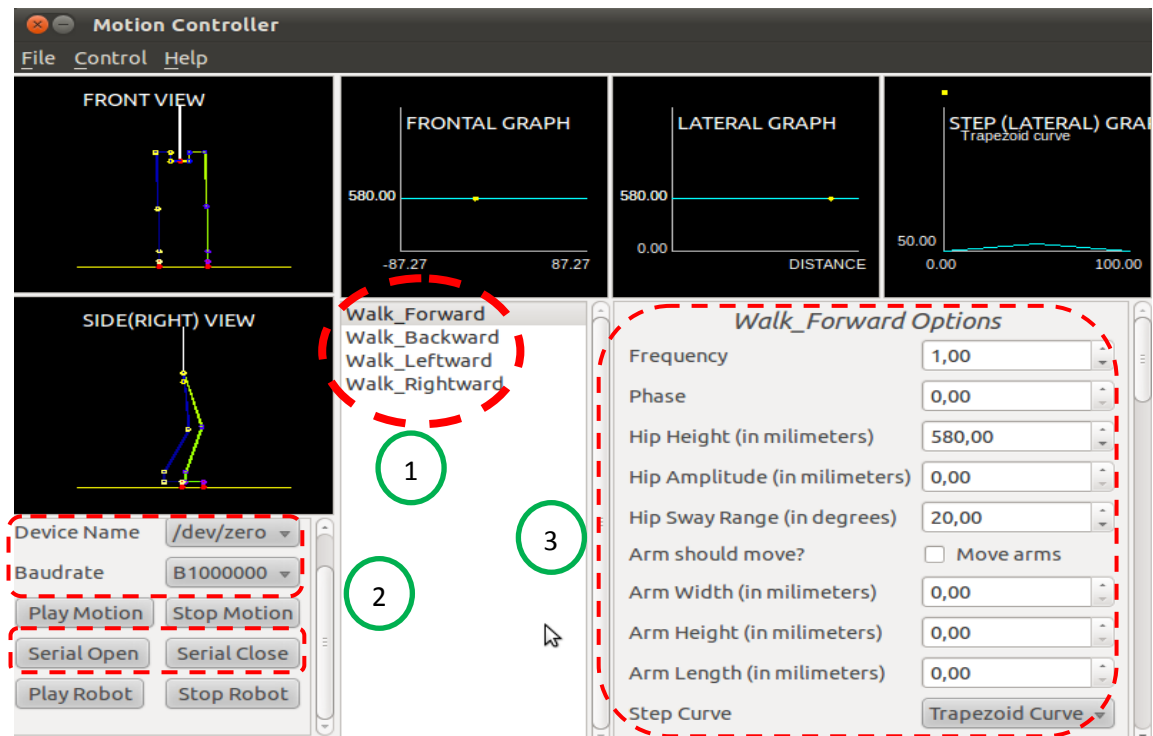


Fig. 6.5 Motion controller GUI design (ver. 1.1)

Incorporated walking types of motion (①) include forward, backward, leftward, and rightward walking. Communication establishment (②) and movement options (③) can be adjusted through the graphical user interface. This version (ver. 1.1) provides facilities for controlling 2 types of humanoid robots (Bioloid and Archie) in combination with graphical simulations.

6.3.2 Scheme of the Motion Controller Program (ver. 1.2)

In the following, different tasks of each frame in graphical user interface are described in more detail (Fig. 6.6).

a) Motion Controller

In this context, the title of software is displayed as well as basic functions such as close and minimizing the main window.

b) Main Menu

The motion controller menu contains three different possible actions: file, control and help menu. The file menu consists of the commands new, open, save, print to XML and exit. The new button opens a new file while the open button allows the user to start the program with predefined parameters which have been saved previously in XML or text file. The save button is used to save the options chosen in frame c and e or any changes made by user. All of the

data can be saved for each type of walking in XML, or text file. The print to XML button saves data about the robots definition, joint type, joint id, joint position as well as the step parameters.

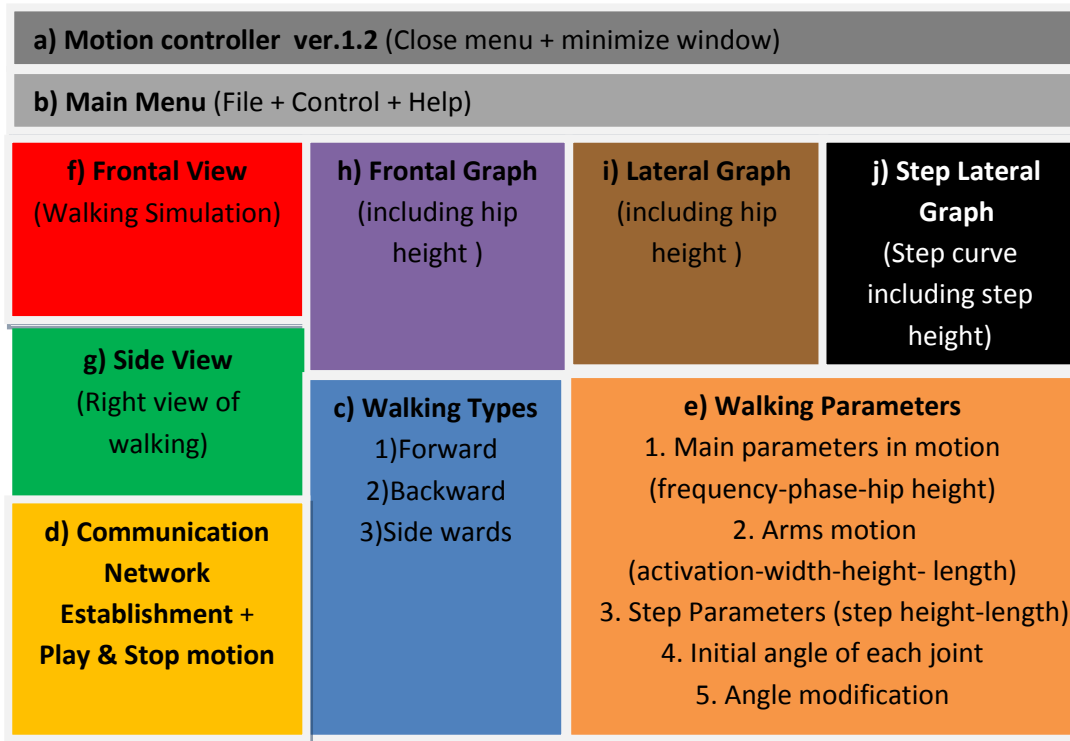


Fig. 6.6 Scheme of the motion controller interface (ver. 1.2)

c) Walking Direction

This frame presents motion experiments for a biped robot that the user can chose for walking. This part presents some basic motions of a humanoid robot, namely walking forward and backward, left or right ward motion.

d) Communication Network + Motion Start and Stop

This window lets the user to check and adjust different features related to the connection between the software (PC) and hardware (robot). These controls are listed below:

✓ Robot Type:

The software architecture allows the user to use the same program with two different robots. Bioloid or Archie robot can be selected.

✓ Device Name:

It enables the user to select the type of terminal to connect the PC with the CAN converter. The Archie robot will use.

✓ Baudrate:

This sets the bit per second rate. The baud rate chosen for the Archie robot communication will be 1 Mbit/s. At 1 Mbit/s one bit will take $1 \cdot 10^{-6}$ sec. and one CAN

message frame takes 64×10^{-6} sec. If the baud rate is changed, then the interval will be longer.

✓ Play Motion/Stop Motion:

Start/Stop the Archie robot simulation in interface.

✓ Serial Open/Serial Close:

this opens the modem device for reading and writing, it is based on the baudrate and terminal that will be connected to the PC.

✓ Play Robot / Stop Robot:

Start/Stop motion of the robot. The robot should perform the same movements as shown in interface within motion controller software.

e) Walking Parameters

Within the Walk-forward Options menu there are five sub-menus: Walk Direction Options, Motion Arms, Step Options, initial angle and modifier angles. These sub-menus are presented as follows:

▪ e.1. Walk Direction Options

Different features of the robot gait can be modified if the parameters within the walk direction option menu are changed. These parameters and how they affect the robot gait are described below:

- ✓ Frequency: This relates to the walking step. If the value is increased the walking of the humanoid robot will be faster.
- ✓ Phase: The walking cycle has different phases. This value depicts the origin where the walking is to begin.
- ✓ Hip Height: the lowest point of the robot's hip on the hip graph. The value is in millimetres.
- ✓ Hip Amplitude: The vertical amplitude that the hip uses to walk. The value is in millimetres.
- ✓ Hip Sway Range: The frontal swing of the centre of gravity. This value is in degrees but the range is measured in millimetres

▪ e.2. Arms motion

The arms motion simulation during the walking is also considered in this window. First of all, the arm width (the distance between end effectors) can be modified. Secondly the height of the end-effectors can be selected. The last and final parameter indicates the lateral length between the end-effectors when the humanoid robot is walking as well as while swinging its arms. The motion arm menu is shown in the Fig. 6.7.

Arm should move?	<input type="checkbox"/> Move arms
Arm Width (in milimeters)	400,00
Arm Height (in milimeters)	340,00
Arm Length (in milimeters)	120,00

Fig. 6.7 Arm motion window

▪ e.3. Step Parameters

The step parameters can be adjusted via this window, it allows the user to select the type of path, either trapezoidal curve or Ellipse curve, which ankle joint should follow as a path or trajectory. All the main parameters considered to control walking step such as height, length, rise and fall angle are depicted in Fig.6.8. Any changes in features of walking are presented on real time graphs and simulation in GUI.

Step Curve	Trapezoid Curve
Step Width (in milimeters)	230,00
Step Height (in milimeters)	50,00
Step Length (in milimeters)	100,00
Step Rise Angle (in degrees)	45,00
Step Fall Angle (in degrees)	45,00
Swing Wating Time (%)	0,25

Fig. 6.8 Step parameters menu

▪ e.4. Initial angle and angle modification

In the last and final frame, the user can modify the initial angle before start walking, and also angle during the walking (in degrees) for each joint of robot.

The motion controller program is developed in order to minimize the communication bugs, errors and miscalculation of smooth walking. Additionally GUI-simulation window for all joints including upper body as depicted in Fig. 6.9 are added. In order to improve Archie walking via motion controller program (ver. 1.2), following items are implemented:

- Internal codes are in necessary classes to improve the running performance.
- Walking parameters modification such as hip height, step length and height and so on (Table 6.2.).
- Joint angle values for smooth synchronised motion are optimised after so many different tests and the final values are presented in Table 6.3.
- Data Communication network and command distribution bugs and errors are removed.

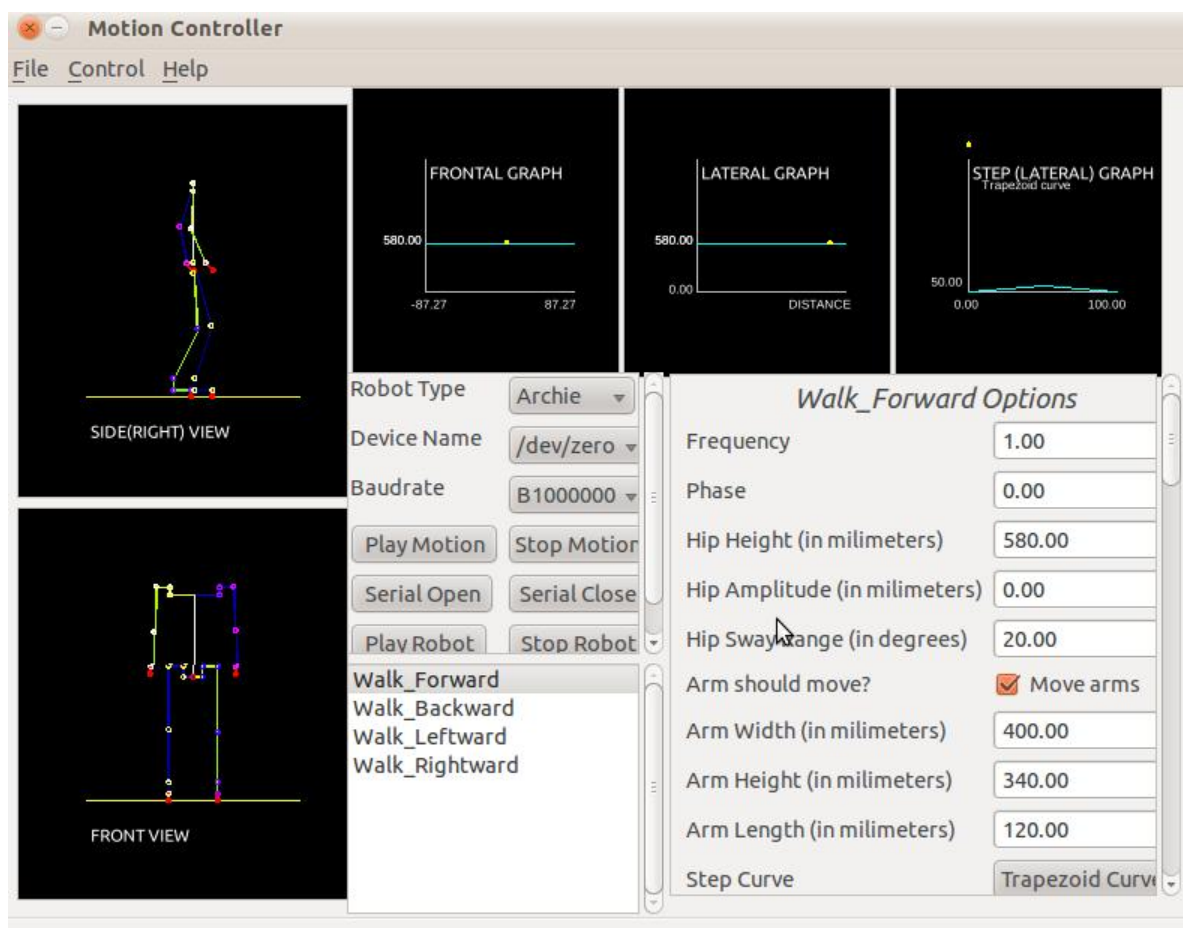


Fig. 6.9 New GUI for motion controller program (ver. 1.2)

Table 6.2 Final optimised value of main parameters in motion controller program (ver. 1.2)

Forward Walking (Parameters)	Optimized value (result)
Hip height (mm)	520
Step Curve	Elliptical
Step length (mm)	60
Step height (mm)	15
Rise angle (deg)	30
Fall angle (deg)	30

Table 6.3 Final optimised joints angle in motion controller program (ver. 1.2)

Initial joint angle	Optimized value (result)
Right Hip lateral (deg)	-30
Right Knee lateral (deg)	28
Right Ankle lateral (deg)	10
Left Hip lateral (deg)	-14
Left Knee lateral (deg)	22
Left Ankle lateral (deg)	10

6.3.3 GUI Improvements in Motion Controller Program (ver. 1.2)

The interface presents two simulations window from two different points of view, side right view and front view. Fig. 6.10 depicts a screen shot of developed Archie simulation window. When the simulation of the Archie robot begins, a yellow point covers the trajectory of each of the three graphs at the same time. This allows the user to gain a comprehensive monitoring of the data flow, in the different phases of the gait cycle.

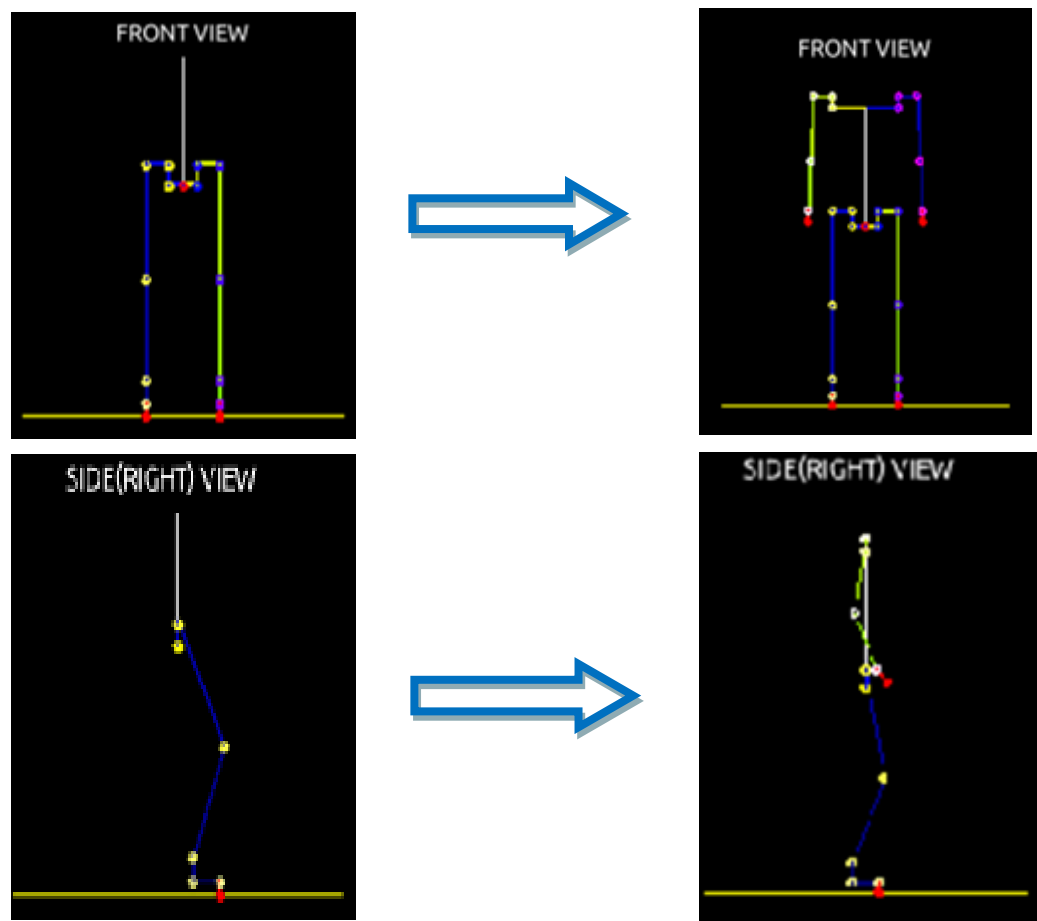


Fig. 6.10 Simulation of the humanoid robot Archie (including upper body motion)

Motion controller simulation has a key role in robot motion creating system. It includes kinematic models of robot, meanwhile being able to simulate a wide range of motion options. Besides the kinematic information in motion controller program of Archie there are graphs and motion simulator for designing a model of an actual and physical behaviour of robot, to execute the movement commands, and test and analyze the execution output.

In this program two different paths such as elliptical and trapezoidal path are defined for end-effector (toe joint). In this window (Fig. 6.11) the robot is in initial position before starts any motion, therefore the yellow point is in the start point of the path (first position point). During robot walking this yellow point follows the desired trajectory which is adjusted already by the main values.

The main goals of using simulation and graph are to provide the platform for the user to play robot in a virtual world and:

- visualize the robot model and movement (walking) in a simulated environment,
- provide a testbed for the development and evaluation of robot control and software behaviours to avoid injuries and damages and necessary or unnecessary changes in robot,
- serve as an interactive, task-level user interface for real time controlling of the robot.

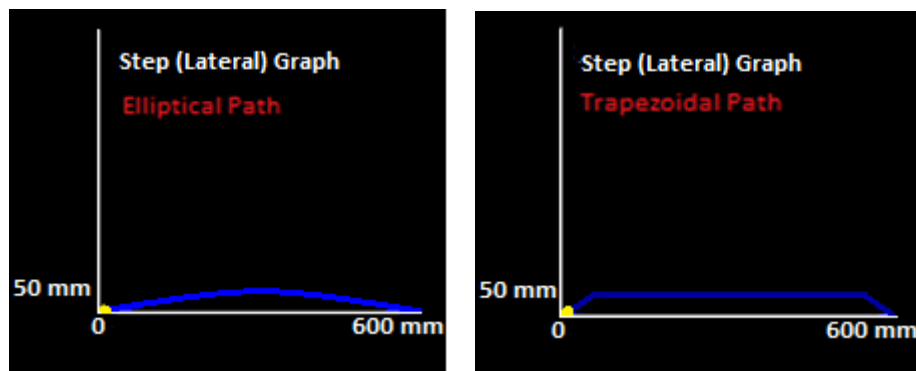


Fig. 6.11 Different desired step path

Different windows (graphs) are considered to simulate and evaluate the robot motion in program. In frontal and lateral view of simulation the robot motion according to parameters which are set already is presented.

It is remarkable that the developed software calculates the angle of each joint using the basic functions supplied by internal modules. As result of this process, the system generates the humanoid robot path. These calculations also provide data to generate the output (Archie simulation and hip motion & step-trajectory graphs). Interface functionality due to classes and codes which already explained is depicted in Fig. 6.12.

From now on the new format of motion controller which is based on the PVT table array for synchronized motion will be explained. This new version is different in path generation algorithm, foot and hip trajectory from the other versions (ver. 1.1 & 1.2). Archie could walk for more than 10 meters for the first time by implementing and running of this version.

It is noticeable that in the following part of this chapter the mechanism and program architecture design of this new software will be discussed, but the actual tests and results are presented in the chapter 7.

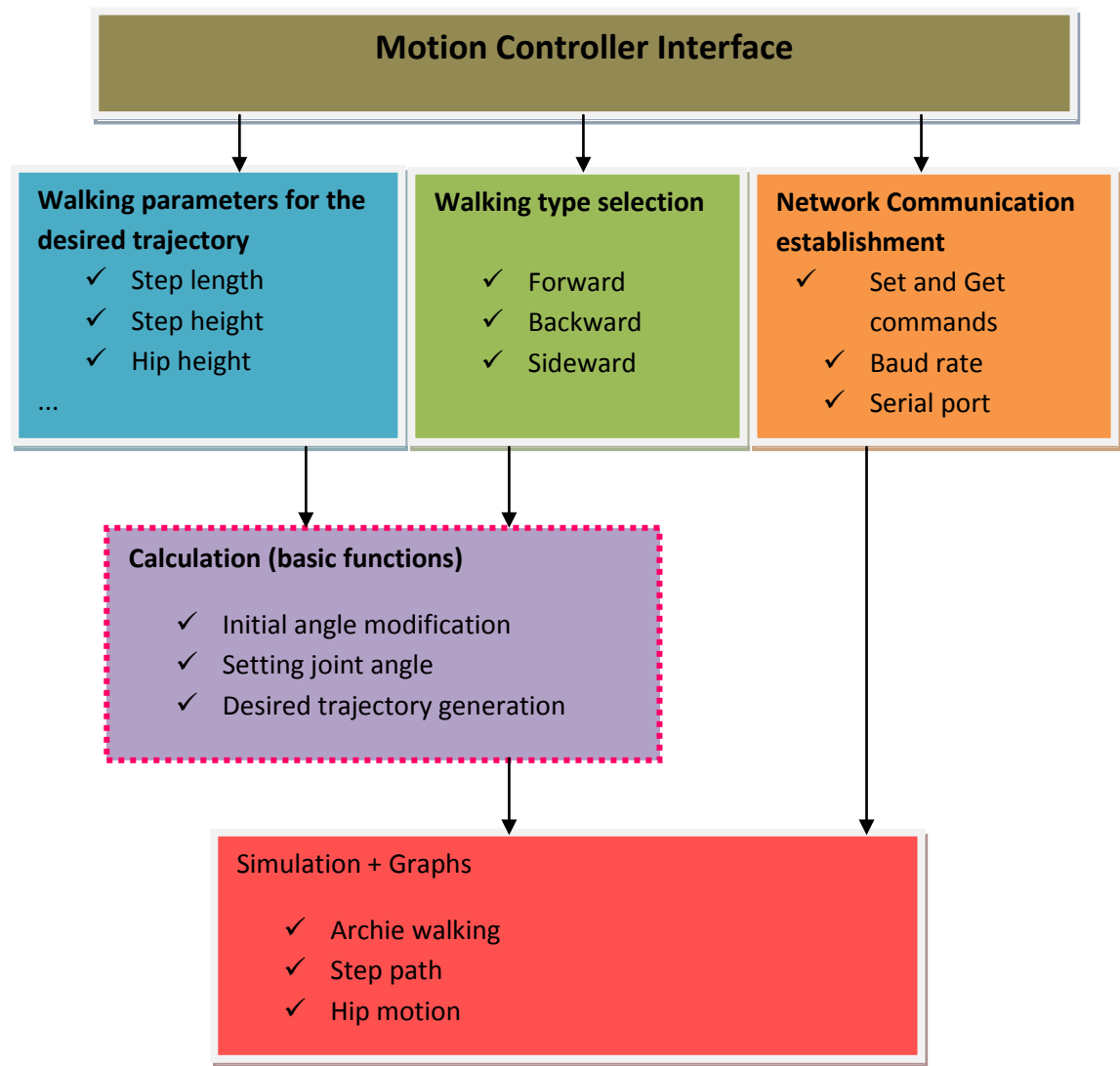


Fig. 6.12 How interface in motion controller software works

6.4 New Motion Controller Program (ver. 2.1)

The latest motion controller program (ver. 2.1.) is created in Linux, and written in C++ codes. Since it is based on a Linux core, it is highly suitable as a research platform owing to the wide availability of high-quality, open-source development tools and libraries. All basic operating system functions (file system, networking, etc.) become standard, and it is relatively easy to add new hardware support by writing a custom device driver.

The system consists of CAN message frame, walking planning and calculation modules. Control loops that require accurate execution cycles are implemented as real-time kernel modules, such as the motor driver loop and the walking control system. Programs and processes that operate over longer control cycles, such as new motion library command, speed control loop and new walking planning method, are embedded.

The main purpose of this part of chapter 5 is to present the design, implementation, and development of a method and principle which differs from those already implemented for walking of Archie. But before starting discussing about any modifications and improvements, the old program/software is analysed based on:

- Interaction style and command generation
- Code design, and expediting the running performance
- Real time communication establishment, fast command distribution
- Walking pattern generation and motion control logic
- Test design, debugging and source code maintenance

The aim of this part is to present the unique way to modify and improve the communication network, data distributing, walking pattern and motion logic that implemented already in older version (1.2). In this manner the new version (2.1) is designed and implemented based on the new hardware improvements (2 new frontal ankle joints and etc.). In this program the new principle is inspired from the paper "Planning Walking Patterns for a Biped Robot" and also adopted from the book called Gait Analysis (Perry, 1992).

It is remarkable that two important specifications which are being user friendly and unify are considered in the interior design. On the other hand, from programming perspective there are several reasons for designing new software and motion controller program (Fig. 6.13).

The new motion controller software is designed based on PT motion for the forward walking of Archie. The command lists and real-time messages are monitored in the graphical interface. It helps the user to find out the location and type of errors easily, and on the other side shows the real time data distribution to joints. This control panel provides the facilities for the user to activate or deactivate any joint just by selecting/unselecting them in GUI, and letting the user to set the walking parameters for the desired path. The initial joint angle value for starting the walking can be loaded and run individually in the command line window.

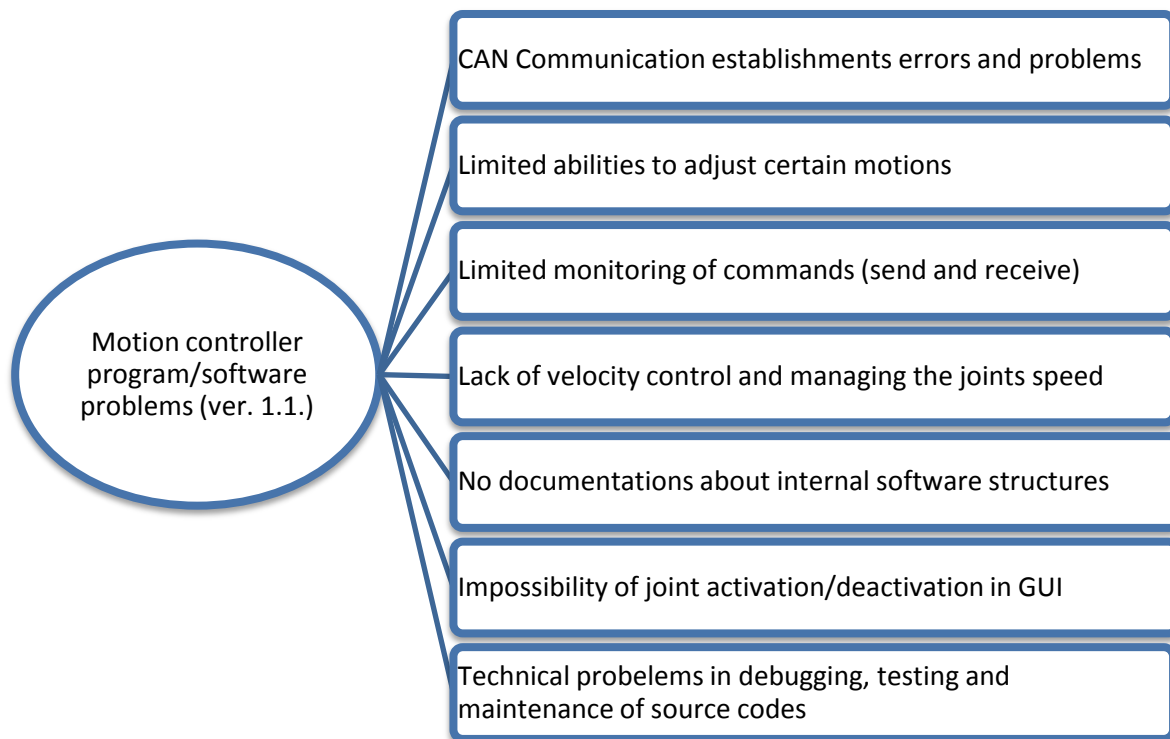


Fig. 6.13 Main problems of old motion controller program

6.4.1 Walking Control System in Program (ver. 2.1)

This section provides an overview of Archie walking control system in motion controller program that generates joints trajectories to follow a given desired motion. A layered software and control architecture is used to aggregate system components and provides a framework for high-level autonomous locomotion behaviours. Walking characteristics such as desired hip movements, upper body posture and step cycles, can be specified and used to generate stable whole-body walking trajectories.

The basic architecture consists of four layers: footstep decision; trajectory generation; trajectory modification; and joint driver control. In order to accommodate these design constraints, a hierarchical architecture that consists of layers of different control cycles is developed. The block diagram of the designed motion controller program for locomotion of the robot is depicted in Fig. 6.14.

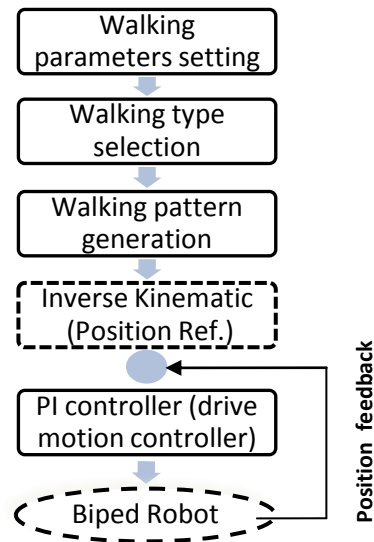


Fig. 6.14 Motion controller program flowchart (Dezfouli & Mohamadi Danial, 2012)

At the first stage, user has to set several walking parameters such as step length (mm), step height (mm), hip height (mm), number of interval, sample time (ms), body center position and USB connection port (Fig. 6.15), and afterwards the desired joints which will be involved for walking and so on. At the second stage, user chooses the walking types such as forward or backward walking, right/left side walking and clockwise/counter clockwise turning of motors.

Pause (μ s)	<input type="text" value="10000"/>	No. of Interval	<input type="text" value="60"/>
Logfile	<input type="text" value="log.txt"/>	Step length	<input type="text" value="60"/>
ST (ms)	<input type="text" value="250"/>	Step height	<input type="text" value="5"/>
USB port	<input type="text" value="/dev/ttyUSB0"/>	Hip height	<input type="text" value="55"/>
		Center position	<input type="text" value="0"/>

Fig. 6.15 Walking parameters (first stage)

In the third stage, proper walking pattern is generated according to the walking parameters and walking types. At the fourth stage, all joints angles are derived by inverse kinematics and control algorithm. Finally, the main computer sends the reference position data to all joint drivers.

6.4.2 Program Communication/Graphical User Interface Design

Since each joint is basically connected to motion controller program via the proposed communication network for an individual motion, the interface should provide appropriate setting ability for position and velocity profile of each joint of robot to manage the proper movement (Fig. 6.16 and 6.17).

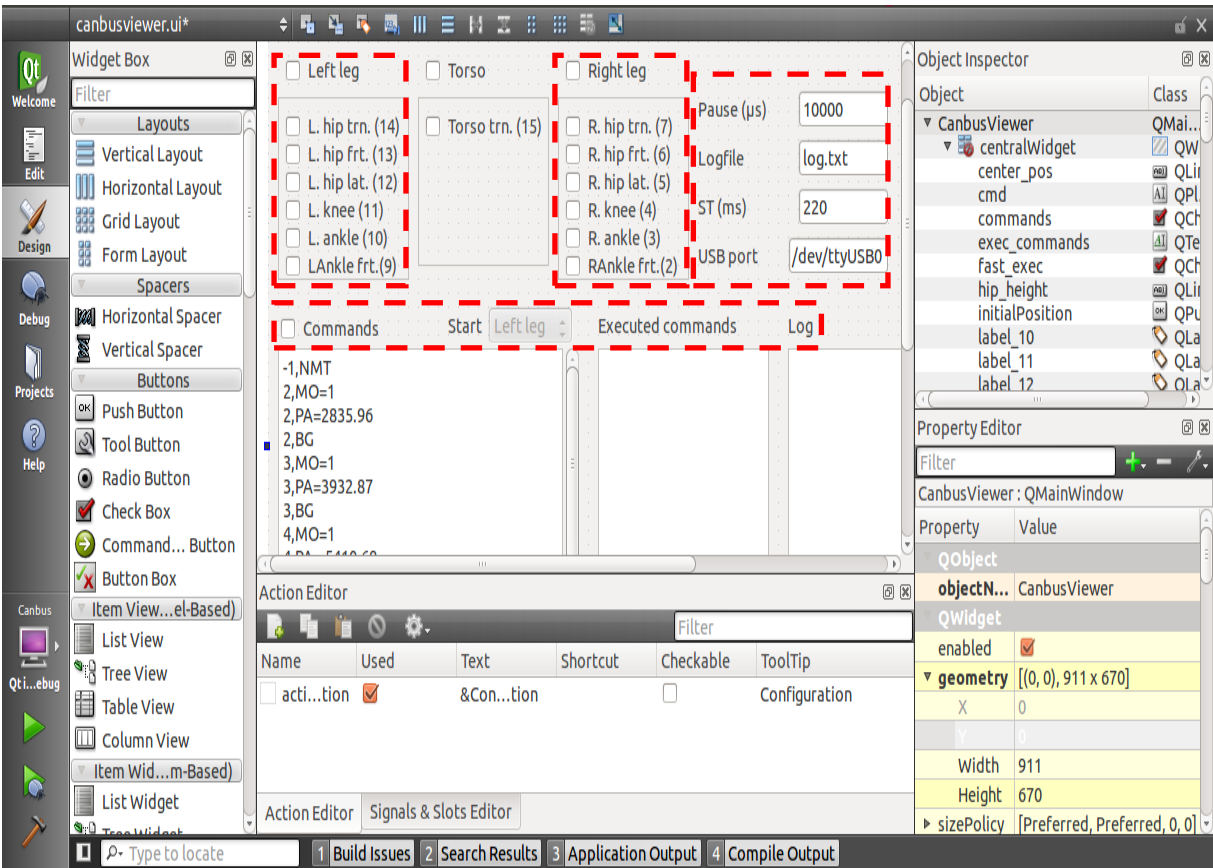


Fig. 6.16 Designing the new GUI

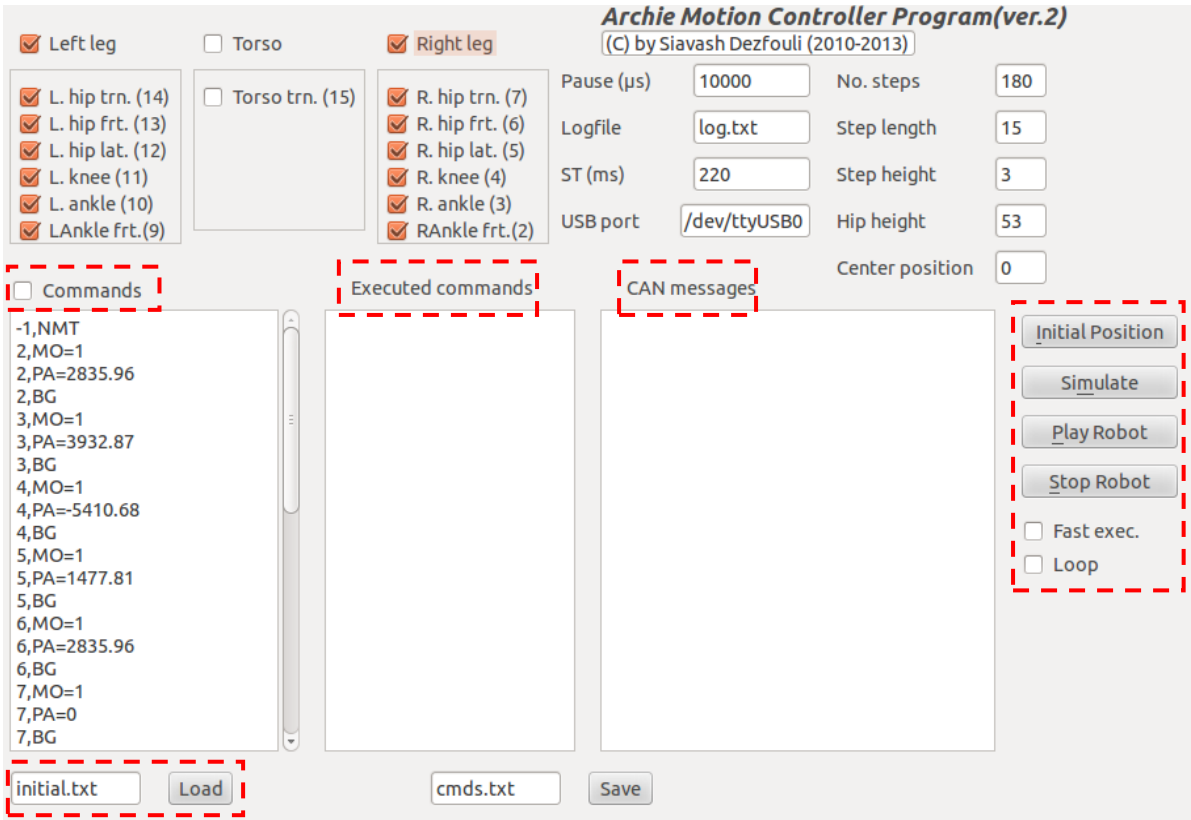


Fig. 6.17 Designed GUI for new motion controller program (ver. 2.1)

6.4.3 PT and PVT Mechanism in Software

One of the main distinction of new designed program (ver. 2.1) is using PT and PVT table in controlling the joints for a stable walking. It is mentioned that the PT motion is embedded in new software but for much more accurate joint control the PVT motion is also embedded. The large majority of motion trajectories are either trapezoidal or S-curve profiles.

In the case of trapezoidal profiles, the velocity is ramped linearly, and in the case of S-curve profiles, the acceleration is ramped linearly. The resulting position profile is 2nd and 3rd order respectively. Although these standard profiles suffice for point to point moves, they are not useful for more complex position profiles or distributed coordinated multiple axis motion. PVT paths are third order (cubic) rather than first order (ex: PT paths). The position and velocity are continuous, the acceleration and jerk are not.

PVT paths tend to be much smoother than PT paths, but the velocity of each axis needs to be provided at each of the supplied points. This can increase the complexity of the application since the velocity at each point is often difficult to determine. The time is defined in milliseconds, not in drive sampling times.

The motor drive interpolates the motion specification in order to calculate the desired position and speed at the sampling instances, when it needs the information. PVT implements a third-order interpolation between the position, speed data provided by the user. For each motion interval, the user specifies the boundary of positions and speeds. Mathematically, the user provides the following data:

- Starting position and speed, denoted by P0 and V0, respectively
- End position and speed, denoted by PT and VT, respectively

The PVT method is an easy to implement algorithm for arbitrary position trajectory generation. Segment times can be relatively large, without much loss of precision. The PVT points can either be stored in a table locally or can be parsed over a communication network to various axes. This allows implementation of a distributed multi axis system without the need for a very high speed deterministic network infrastructure.

A three-column table (Table 6.4) is used to define PVT motion. Each row of the table defines the position and speed at a single time instance. The table has 64 rows, enabling the specification of up to 63 consecutive PVT motion segments (64 segments if the table is used cyclically). The cells of the table may be accessed using the QP, QV and QT commands:

- The QP[N] command sets/reads the nth row of the P column.
- The QV[N] command sets/reads the nth row of the V column.
- The QT[N] command sets/reads the nth row of the T column.

Table 6.4 PVT table

Index	P (32 Bits)	V (24 Bits)	T (8 Bits)
1	QP[1]	QV[1]	QT[1]
2	QP[2]	QV[2]	QT[2]
3	QP[3]	QV[3]	QT[3]
...	QP...	QV...	QT...
64	QP[64]	QV[64]	QV[64]

The first PVT point must be within the range $XM[1] \dots XM[2]$. the remaining PVT points need not be within modulo range; but the difference between consecutive PVT position points must be less than $(XM[2] - XM[1])/2$. For example, suppose that $XM[1] = 0$ and $XM[2] = 1000$. If the PVT describes a trajectory beginning at 0 and ending at 10,000, the motor will travel 10,000 counts, fully completing its position range 10 times. In a PVT motion, the desired position and speed are provided at selected time instances. Between these specified times, the motion controller interpolates to obtain smooth motion. The position and speed specifications are absolute, while the time specification is relative.

A PVT motion can be referenced in absolute time by requiring it to start at a specified time. The BT (Begin on Time) command is used to start a PVT motion exactly at a given time, using a microsecond resolution. The ability to relate PVT motions to absolute time makes them ideal for tightly-synchronized multiple axis motions. PT or PVT motion is programmed as a sequence of points that are visited at programmed times. The QP[N], QV[N] and QT[N] arrays may be referred to as cyclical buffers. In cyclical mode, periodic motions can be set to run forever. The MP[N] array defines how the QP[N], QV[N] and QT[N] tables are used for PVT and PT motions, as mentioned in the Table 6.5 (www.elmomc.com, 2013).

6.4.4 Motion Management in PVT Method

In PVT mode, the drive manages a “read pointer” for the PVT table. When the read pointer is N, the present motion segment starts at the coordinates written on the Nth row of the table, and ends at the coordinates of the (N+1) row². When the time period specified by QT[N] elapses, the N segment is complete, the drive increments the read pointer to N+1, and then reads the N+2 PVT table row in order to calculate the parameters of the next motion segment (Fig. 6.18).

The entire PVT table need not be used for a given motion. The different PVT mode to be used in motion controller program is defined by the following parameters (Table 6.5):

Table 6.5 PVT mode definition

Parameter	Use
MP[1]	Lowest valid row of the PVT table
MP[2]	Highest valid row of the PVT table
MP[3]	A bit field: Bit 0 is:
	0: Stop motion if read pointer reaches MP[2]
	1: Continue motion when read pointer reaches MP[2]. The next row of the table is MP[1].
	Bit 1 is :
	0: PVT motion not expected to terminate. Issue an exception if it does.
	1: PVT motion expected to terminate. When all data in PVT table has been used, exit PVT mode to Idle mode without issuing an emergency object.

The PVT table may be written online while PVT motion is being carried out. An infinite non-periodic motion can be generated in cyclical mode (MP[3]=1) by programming the PVT table on-the-fly. The host must know how much free place is available in the PVT table in order to continue programming and executing PVT motion. This is achieved most efficiently by tracking the table read and write pointers. The host is aware of the write table status, because it controls writing to the table. If there is a doubt, the host can query MP[6]. The PV command is used to query the read pointer.

The read and write pointers can be mapped to a synchronous PDO, so that a CAN master can efficiently and continuously stay informed about the status of multiple drives running PVT in parallel in a network. Rather than polling the status of the PVT motion continuously, the host can use the queue underflow CAN emergency object as a request to refill the PVT table. An unused part of the PVT table may be programmed for the next motion while the present motion is executing. An attempt to modify the data of an executing motion segment generates an error.

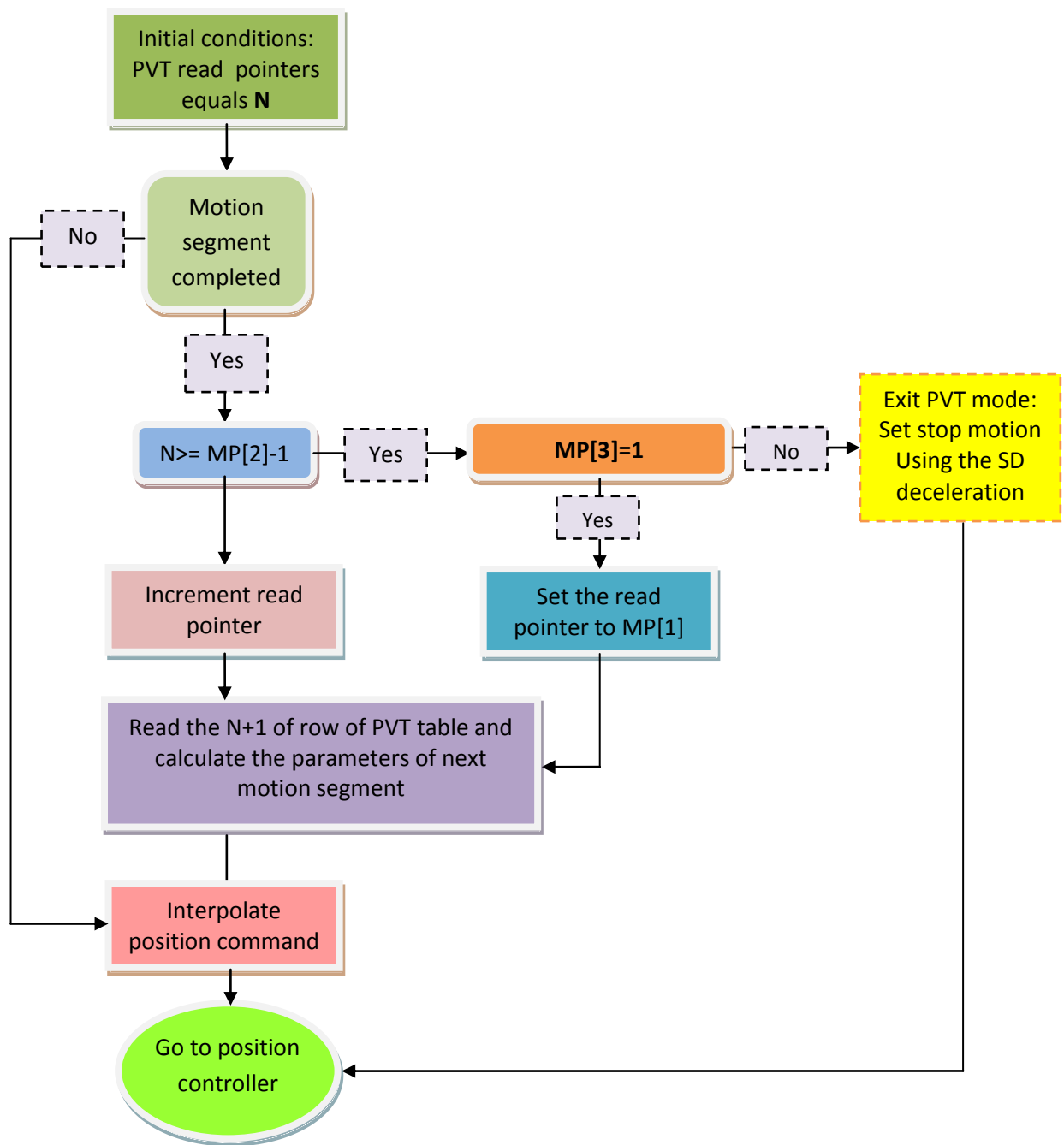


Fig. 6.18 Designed PVT decision flowchart

6.4.5 PVT Motion Programming Message

An entire row of the PVT table may be programmed by a single PDO — 0x200+ID — where ID is the node ID of the drive. Note that before using this PDO, it must be mapped to the object 0x2001. The PDO is mapped for the PVT mode in mentioned in Table 6.6.

Table 6.6 PDO mapping for PVT mode before using

PDO mapping for PVT mode before using	
Object dictionary index	0x2001
Type	Record, three elements
Access	Write only
Structure	Signed 32 position, signed 24 speed, unsigned 8 time
PDO mapping	Yes
Value limits	No
Default value	Not applicable

The PDO does not specify the PVT table row to be programmed; instead, a write pointer specifies the row. The parameter MP[6] initially sets the write pointer. A new PVT CANopen message (object 0x2001) write the data to the table row indicated by MP[6] and then automatically increments MP[6].

The PVT command auto-increment mode is described in the following flowchart (Fig. 6.19). To stay informed about how the PVT motion is advancing, the read and write PVT table pointers can be received continuously, mapped to a synchronous PDO, or the “Queue low” emergency signal can be used to indicate the need for more data. The “Queue low” emergency message includes the present location of the read pointer and the write pointer. The host is well aware of the location of the write pointer, because it can count its own messages.

However, a data message may be rejected because the queue is full, or because a message has been lost. In such a case, the drive issues an emergency object to the host. Accurate timing with respect to the host is the essence of multiple-axis synchronized motion. Such timing can be achieved by using the CAN SYNC signal and the CAN synchronized BG service, as described already. Finally the desired joint trajectories are generated to let robot smoothly walk.

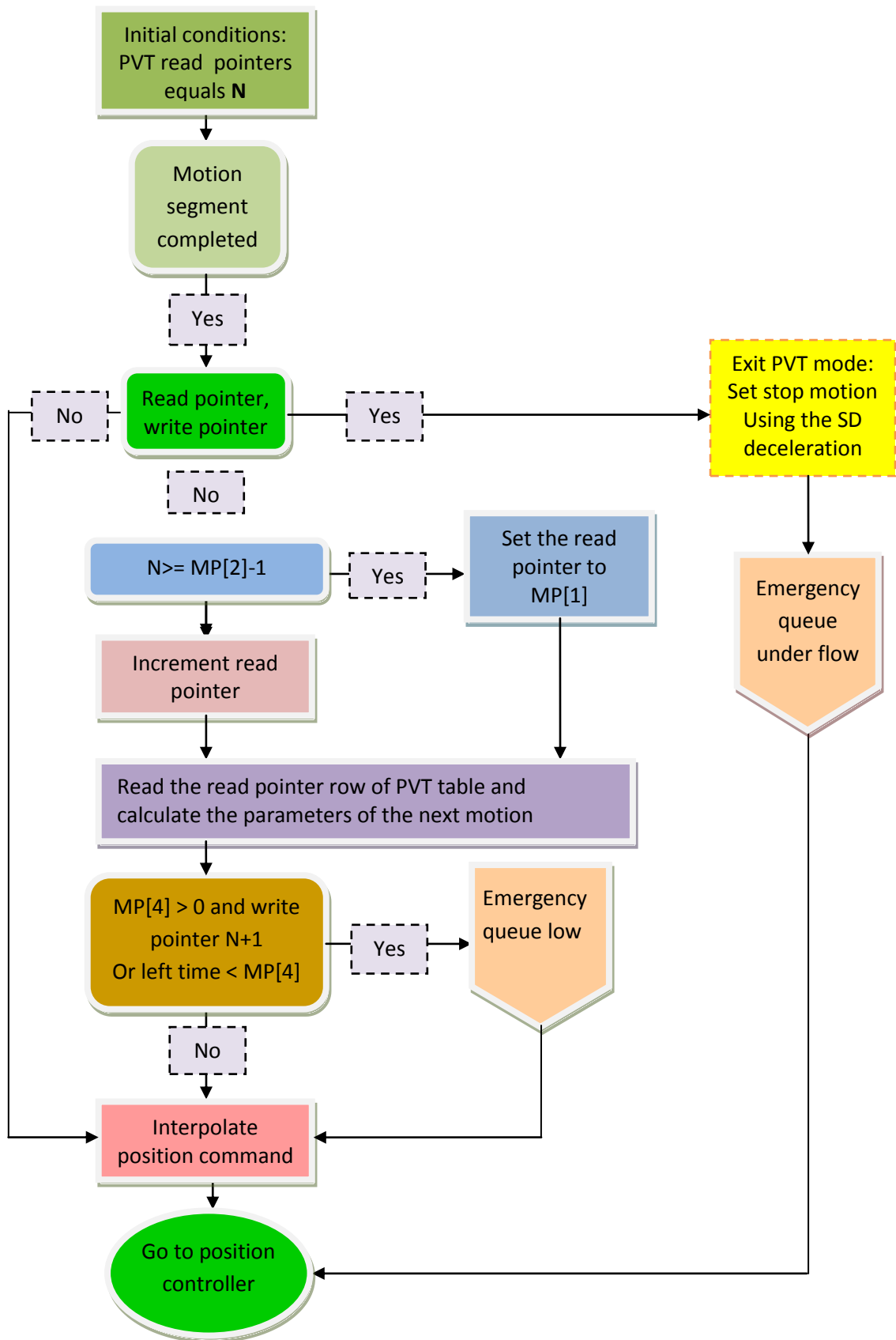


Fig. 6.19 Auto increment PVT mode design

Summary

When a new software project begins, it is necessary to define the tools to be used all along the process. How will a design be described? How will the development and old version of the project be maintained? A motion planning software is not a small application; it really is a big software project where many libraries are working together. For this reason, it is important to begin with a good task plan covering all the development steps. Hence in this chapter the different motion controller program (software), old version (ver. 1.1), the developed one (ver. 1.2) and the latest new version (2.1) for Archie locomotion is described.

The latest version (ver. 2.1) is designed to improve motion planning, stable walking and minimize the computational efforts which devoted for generating smooth walking motion. It is adopted a centralized system in order to construct a platform where user can easily develop and debug perception–action coupling control schemes. All of the sensor data, such as actuator encoder values, are directly available to the PC. Control commands are sent from the PC to the motor drivers directly. This requires an operating system in which many different cycle control loops can be executed concurrently (from 1ms servo loop to higher level trajectory generation and motion planning loop, which have cycle times of several seconds).

In order to achieve a smooth walking of biped humanoid robot Archie, the new motion controller is designed including :

- New joint trajectory generation to follow a given desired motion
- New motion library command to be used for joints motion based on the PT,PVT motion method
- New motion management mechanism (easy switching between modes)
- Developed synchronization mechanism
- Code priority message transmission based on the arbitration
- PDO mapping and planning mechanism
- New simple GUI including data flow monitoring to detect the errors in transmitted or received commands, easy joint activation/selection for single, multiple or synchronized motion

Chapter 7 New Software Implementation and Forward Walking Test Results

7.1 Introduction

In previous chapters the hardware and software development including the new structure, new creating motion design and program for a biped humanoid robot Archie are presented. The main contribution of this PhD thesis is implementation approach and testing the new proposed motion creating system (chapter 3,4,5, and 6) which is including:

- Hardware design and new developments (chapter3),
- New approach to design a CAN network protocol for Archie (chapter4),
- New approach of data transmit planning (new Mechanism) including error detection system (chapter 4),
- New PDO mapping approach for a synchronized joint motion (chapter4),
- Management approach of data transmission via proposed network to send/receive CAN commands (chapter4),
- New trajectory planning principle for Archie walking (chapter5),
- New drive motion management approach (chapter5),
- New drive motion reference generator (chapter5),
- New approach of motion command structure based on the PT and PVT modes (chapter5),
- Fault diagnose and motor protection mechanism (chapter5),
- New motion controller software including real time motion control system, data/command monitoring, CANbus module, synch module, data link and control modules (chapter6),
- New GUI with easy joint activation/deactivation (chapter6).

This chapter describes carefully the implementation stages based on the above improvements, including software implementation (CANbus Viewer module, message controller module, message setting module) and joint motion tests (single joint, multiple joints and synchronized joints motion). At the end the walking performance (half gait test, full gait test, static walking test) is presented to proof the effectiveness of the new implemented motion creating system.

7.2 Software Implementation and Joint Motion Test

In early test stage, it is significant to investigate the performance of the designed communication network (chapter 4) and new designed motion control software (chapter 6). The software implementation test consists of single, multiple and synchronized joint motion tests and examine the monitoring method which is described in chapter 6. The designed CAN

monitoring system assists to find the errors (trouble shooting) in designed communication network and check the accuracy of the transferred designed command signal in the system.

7.2.1 New Proposed CANbus Viewer Module Structure

It is the unique designed module, embedded in new motion control software to configure and control the proposed CAN network implemented with the following functions. (Fig. 7.1).

- CAN_Init: function for CAN setting such CAN clock, CAN baud rate (1 Mbit/s) and CAN mode (Basic CAN),
- CAN_GetMessage: when a new message is completely received it will be copied using this function from the CAN internal buffer.
- CAN_SendMessage: function to send a message to the master via the CAN bus
- CAN_Task: one of the cyclic processes. This function checks whether new messages exists in the receive buffer and if so - they are removed (consumed) from the buffer and processed further.

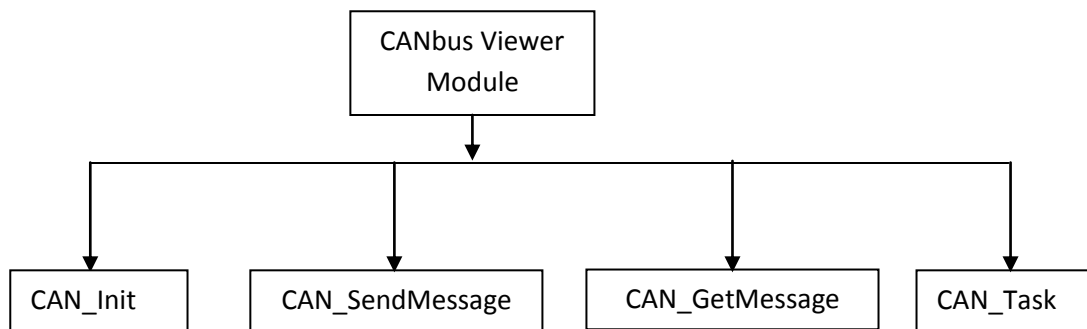


Fig. 7.1 Functions implemented in proposed CANbus Viewer module

The CANbus Viewer module ((*CanbusViewer::sendCommand(std::string command, int id)*) is responsible for constructing the commands which should be sent to the CANbus controller module. There are two options when invoking this method: simulation only (when button Simulate is pushed; or actually sending the commands (button Play robot was pressed).

The called function *CanbusViewer::sendCommandToController (c, id, msg, objectid)* implements the real command sending functionality. Every simulated or send command is displayed in the text field shown in Listing. 7.1 (lines 20-23). Finally, if the fast execution mode is disabled, the GUI is made responsive after every send command (lines 24-26).

```

1 int
2 CanbusViewer::sendCommand(std::string command, int id) {
3   std::string c = command;
4   if (c == "NMT") {
5     c = hCommand[0].message.toString(); //t0008010000000000000000
6     objectid = 0;
7   } else {
  
```

```

8 objectid = 6;
9 }
10 struct CanbusMessage msg;
11 initCanbusMessage( & msg );
12 int ret;
13 if (simulate == false) {
14 //sends the actual command to the controller
15 ret = CanbusViewer::sendCommandToController(c, id, msg, objectid);
16 } else {
17 //only simulation, no command is sent to controller
18 std::cout << "Sending command " << c << " to " << id << "/" << objectid << std::endl;
19 }
20 //append to textedit
21 std::stringstream ec;
22 ec << id << "," << command;
23 ui> exec_commands> append(ec.str().c_str());
24 //make GUI responsive
25 if (ui> fast_exec> isChecked() == false)
26 QApplication::processEvents();
27 return ret;
28 }

```

Listing. 7.1 CANbus Viewer module structure

7.2.2 Proposed Implemented Data Layers

The implementation block diagram of the CANbus Viewer module is depicted in Fig. 7.2. CAN transceivers provide the differential physical layer interface between the data link layer, the CAN controller module and the physical wiring of the CAN bus.

Implementation of main layers are based on the:

- 1) Physical layer transceiver to translate the CAN messages to/from differential signals across a physical medium such as a twisted pair cable,
- 2) CAN controller module embedded in drive motion controller that implements the data link layer and finally
- 3) CANbus Viewer application, implementing the application layer protocol (translating the software application data to/from CAN messages).

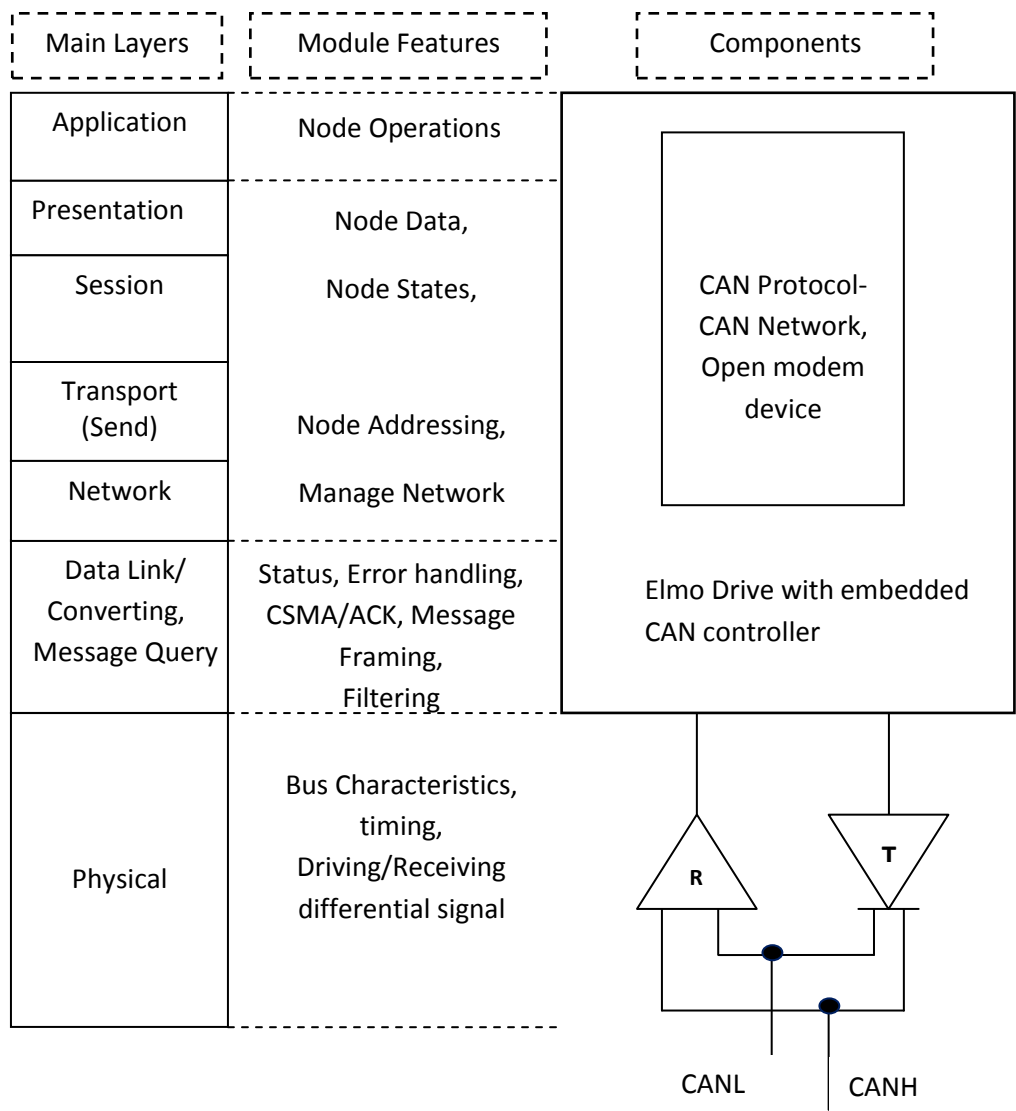


Fig. 7.2 Proposed CANbus Viewer module implementation chart

7.2.3 Error Mechanism Implementation Approach

Designed CAN node implements two error counters; a transmit error counter and a receive error counter. These are incremented on the basis of transmit or receive errors and decremented on the basis of successful transmission or receiving of messages. An error frame is distinguished by having six consecutive bits. This sequence is dominant or recessive depending on the state of the node transmitting the error. This sequence violates the normal transmission rules and so is detectable by other nodes. Any node transmits error frames immediately when it detects an error. As an error frame itself highlights an error, other nodes transmit their own error frames, resulting in a superposition of multiple error frames.

The sequence of six consecutive bits is the error flag. The proposed error frame also comprises an error delimiter to allow for the error flags from other nodes overrunning the initial six bit periods. An example of transmission implementation of a created CAN message

with an error and a subsequent CAN active error frame is shown in Fig. 7.3. compared with obtained CAN message frame in Archie from oscillator (Fig. 7.4).

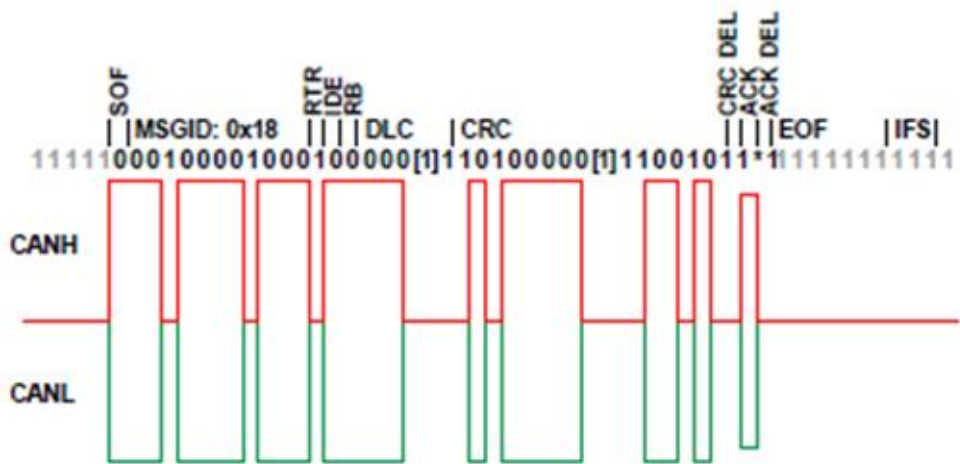


Fig. 7.3 Normal transmission with ACK (Marais, 2008)

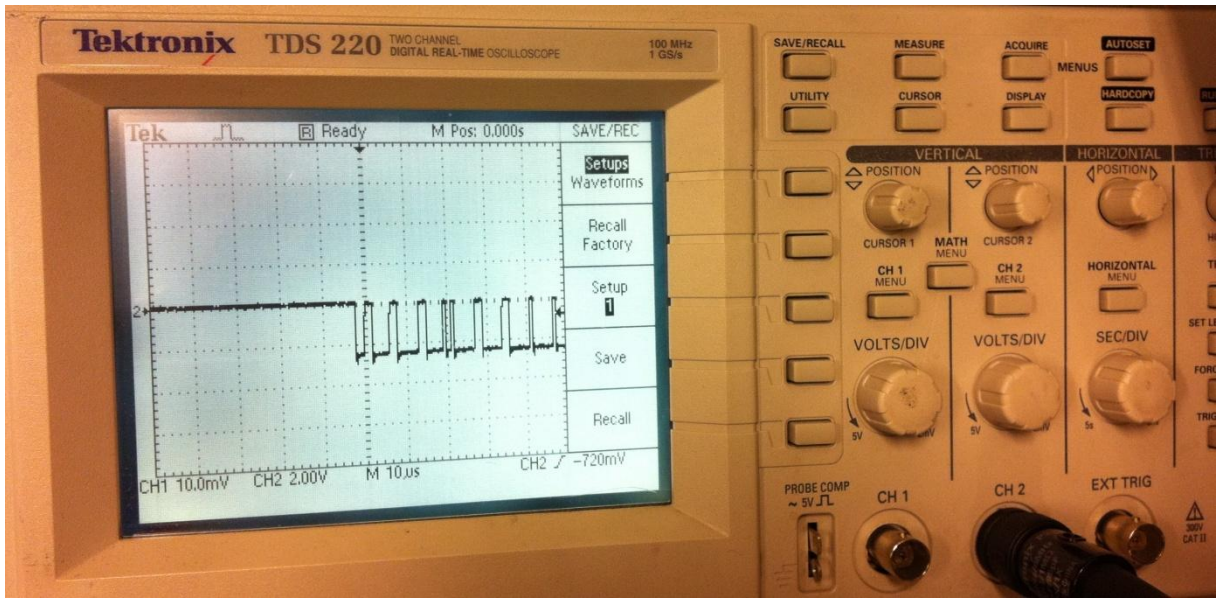


Fig. 7.4 Measured CANL normal transmission with ACK in Archie

Any node transmitting an error flag subsequently sends recessive bits until the bus is detected as being in the recessive state, after which, an additional seven recessive bits are transmitted. The node can then attempt transmission of regular real-time frames.

7.2.4 Proposed Message Controller Module

The structure of the embedded CAN message controller module in motion controller software is fundamental to the schemes used for achieving robust communication through error

detection, as well as inherent prioritization of message and multiple drivers capability based on bit-wise arbitration. Hence the CAN controller module detects and handles errors and supports the error detection by framing real-time messages.

As presented in chapter 4. signalling for real-time message at the physical layer comprises leaving the bus in a high impedance recessive state for a transmission of 1 and transmitting a different high/low dominant state for a transmission of 0. On the other hand, the message identifier form part of the designed message frame, in conjunction of flag bits (Corrigan, 2002). This section of the real-time message frame is termed the arbitration field, with the ID and flags denoting the message type, dictating arbitration and as a result, message priority.

Accordingly message controller unit with configurable data FIFO is implemented (Fig. 7.5). In this new proposed approach the transmitted CAN frames are filtered and stored in the FIFO, and will be received with a different identifier. The implemented module is running with 1Mbit/s baudrate, the message controller implemented to independent bus timing logic. To avoid data corruption or loss of data a FIFO structure is a reliable solution.

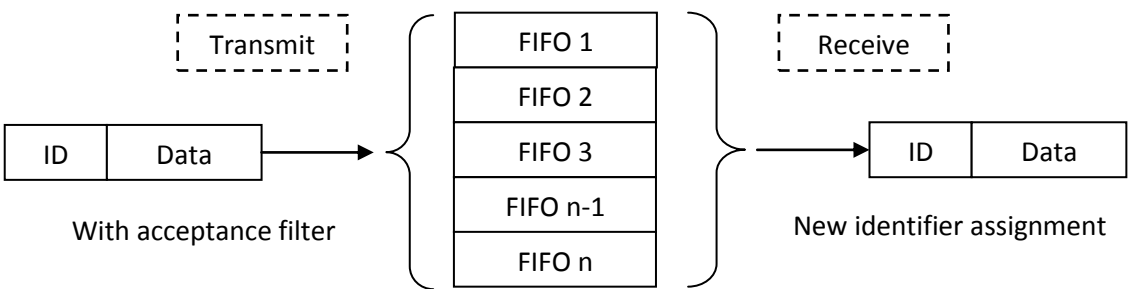


Fig. 7.5 New joint message controller implementation approach

Modules spend a lot of time to poll each other and some will have to wait until its partner is ready for communication. Therefore the more complicated approach based on message passing and queues was chosen. Every single described module in above has a queue behind its input. A module can put a message in an input queue at any time, without having to wait for the receiver to be ready for it.

The receiving module reads the message when it is ready with processing the previous message from the queue. Each module implements a so called "message loop" which will get a message from the queue, process it, get the next message, process it and so on. Processing can take as long as it needs. When the input queue is empty, the execution of the message loop thread will be suspended until a message becomes available. Each proposed module should have only one message loop, so all messages are guaranteed to be processed in the same way.

The single message loop requirement makes it difficult to handle answers as normal messages (orders and allocation). The second example is a message loop which handles all messages in a

single thread. This is convenient for operations which take little time, or for operations which have to be executed sequentially anyway. The program uses functions to process the messages, so the difference with the multi-threaded implementation will be more clear. All handler functions have the same parameter, the address of the pointer to the buffer. All typecasting, parameter extraction etc. is done in the handler function and the message loop stays clean and short. The pointer is also used to return the answer buffer for question handlers, so the message loop can reuse the buffer.

7.2.5 Real-Time Message Setting Approach

In order to perform a synchronized motion, real-time messages should be assigned to specific joint, since each message has its own command and value. One of the most important thing before starting walking or any joint motion is to initialize the primary joints position or posture of the robot. In the following the proposed setting to get the joint zero position for Archie is described:

- ID and baudrate setting (Fig. 7.6): the first step is to assign an ID and baudrate (based on the designed communication network) to each single joint, to prepare joints for receiving their own messages. A joint will only receive a command if its corresponding check box is activated. Therefore, it is possible to activate or deactivate only the joints needed for the specific motion.

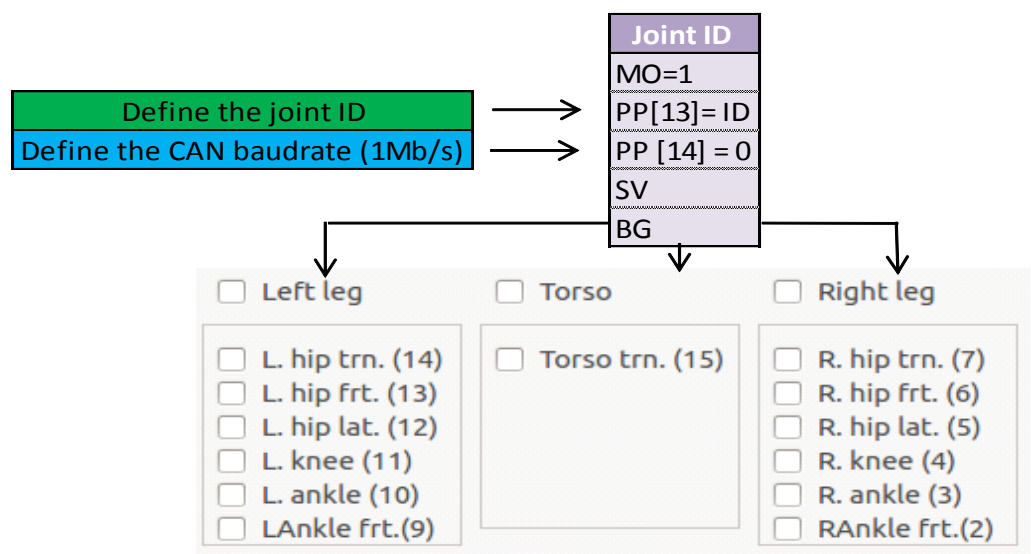


Fig. 7.6 Joint ID and message baudrate defining approach (1 Mbit/s)

- PX command: Defining the fixed initial angle for lower body joints is a must. In this manner there is no need to set initial joint angle after motor operation. It means that upon power on, the main position is set to zero. PX accumulates the main feedback pulses, and it can count cyclically. When the motor is off, PX may be used to set a value for the position counter by typing PX=n. The implemented code structure for setting the joint position to zero is mentioned in the following (Fig. 7.7).

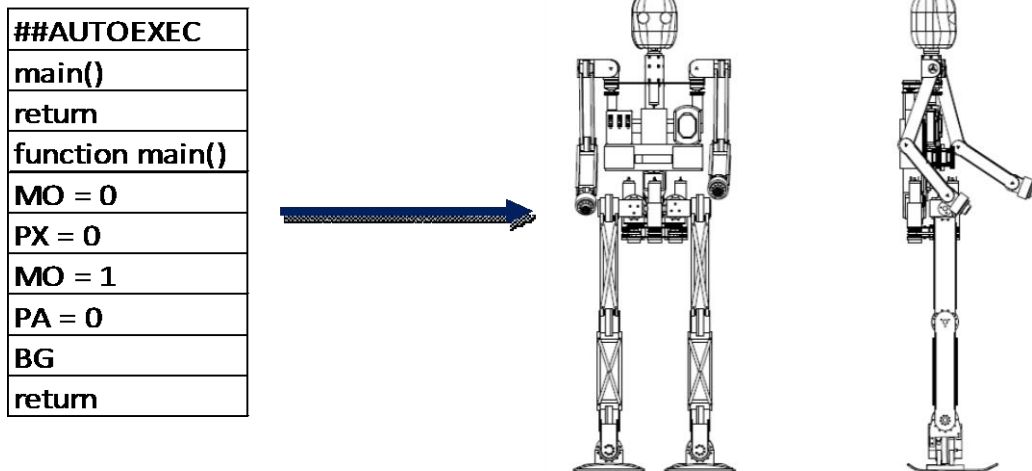


Fig. 7.7 Implemented zero position command's structure for lower body joints

7.3 Joint Motion Test

7.3.1 Single Joint Motion Test

The simplest task is to move one single joint to a certain degree. The single joint motion test assists to realize effectiveness of the software implementation and on the other hand lets to find out the joint angle limitation. The designed codes structure in C++, implementation process and test result are presented in the following.

In the first step the desired joint for motion is selected and afterward the commands (initial position + motion) are executed. Finally the designed CAN messages are generated and transferred to joint. At the end the motion is created and the joint starts to move according to different angle values. The number of the transferred CAN messages depends on the number of interval which is considered already in the desired trajectory of motion. In order to have much more smooth motion it is necessary to increase this number.

To read and write messages from and to the CANbus controller, the methods *readI7565* (*int fd, uint8_t * buffer, unsigned int msgLength, struct timeval * timeOut*) and *writel7565*(*int fd, uint8_t *buffer, int ilength*) in file *i7565.cpp* are used. Of course, a connection must be established first, to write a command, such as MO=1, to the CAN-Bus controller, it must be translated in a hexadecimal representation. The transformation to hex code is done in method *convertStrToCanbusMessage* (*char const * out, struct CanbusMessage * msg*). Translating a hex code back to a string is done by calling *convertCanbusMessageToStr*(*struct CanbusMessage const * msg, int checkSum, char * out, int length*).

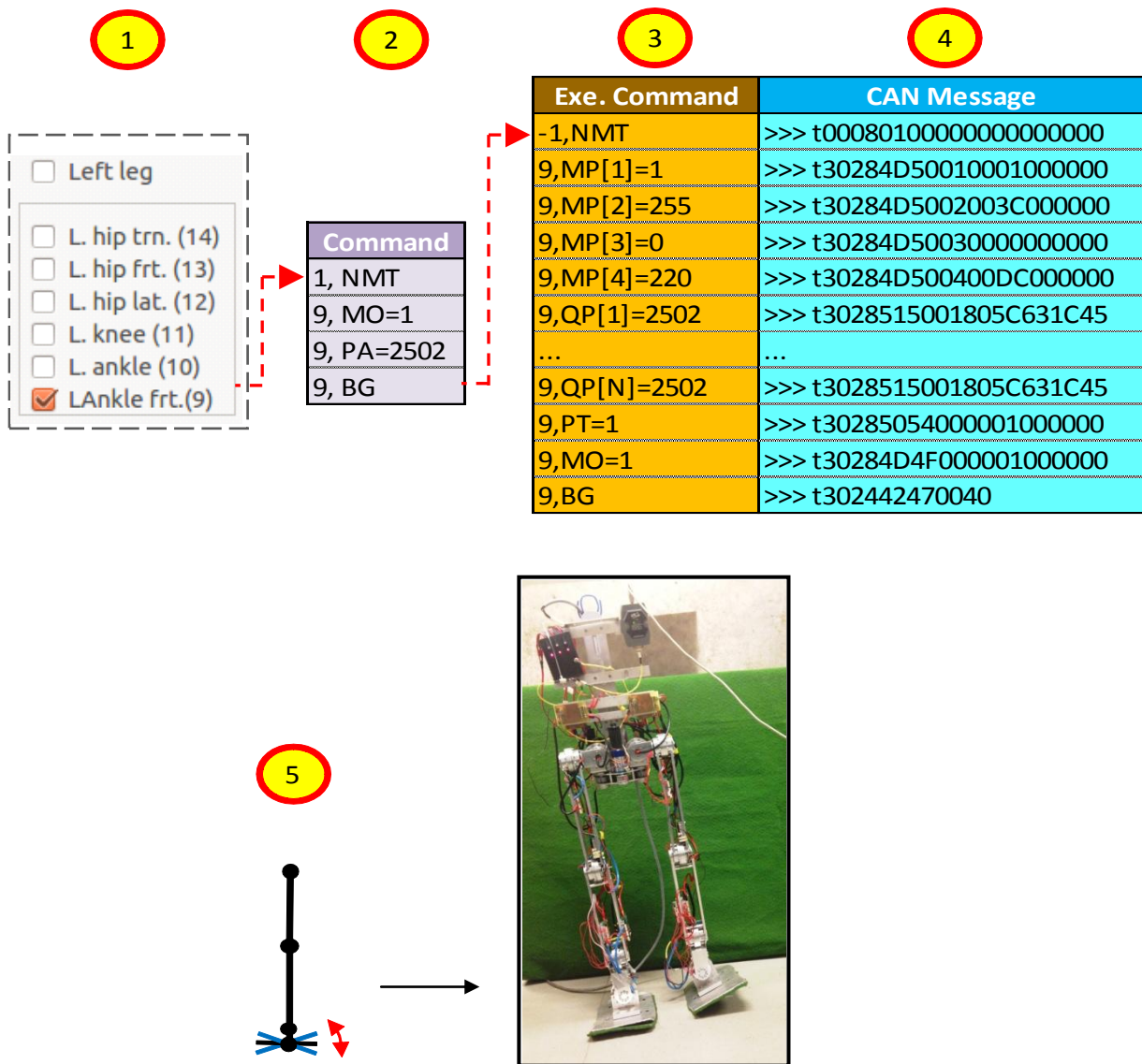


Fig. 7.8 Single joint motion implementation

While CANbus controller module is connected to network an initial NMT command must be sent for establishing the network management, hence the NMT command is translated to the hex code message t00080100000000000000. The CANbus Viewer module (Listing 7.2) is invoked for every position per joint. Therein, the servo motor is switched on (line 4), the angle degrees are calculated to radian (line 6), the absolute position command is constructed (line 6) and send (line 7), as well as the begin motion command (line 8).

If a motion of a joint is still in progress while the next BG command is received from the CAN-Bus controller, the former movement is aborted and the joint tries to reach the new position issued with the newest PA command. Therefore, line 12-14 define a pause in which the execution of the program (i.e., sending of new commands) is halted for a defined time period. In order to get feedback while a joint reached its position, the motion status of the servo motor can be realized (lines 18-22). If the MS command returns 2, the joint is still moving. Return value 1 indicates that the joint reached its position or the motor is off (either way, the

servo is stationary). The servo status per joint, query every 500 microseconds and wait with the execution of the next command until the joint reached its desired position. Therefore, the returned hex message of the CAN-Bus controller module can be checked, while it matches a specific pattern it means that the joint is still in motion (line 19). This implemented method helped to improve the correctness of the intended (desired) motion (Fig. 7.8).

```

1 void
2 CanbusViewer::move(int id, int j, int i) {
3   if (joint_nr[id]) {
4     CanbusViewer::sendCommand("MO=1", id);
5     stringstream pa;
6     pa << "PA=" << CanbusViewer::deg2rad(test_pos[j][i]);
7     CanbusViewer::sendCommand(pa.str(), id);
8     CanbusViewer::sendCommand("BG", id);
9     // sleep necessary, next BG overrides former command!
10    // caution: usleep(microsec); 1000000 = 1sec
11    // caution: sleep/usleep not platform independent!
12    if (ui> pause> text() != "" && ui>pause> text().toInt() > 0) {
13      usleep(ui>pause>text().toInt());
14    }
15  }
16  /*
17  // MS = 2 > moving; MS = 1 > stationary
18  CanbusViewer::sendCommand("MS", id);
19  while (reply_msg.compare("M(0) Id:28a DLC:8 4d530000 00000002") == 0) {
20    qDebug() << ":::::" << reply_msg.c_str() << reply_msg.compare("M(0) Id:28a DLC:8
21    4d530000 00000002");
22    usleep(500);
23    CanbusViewer::sendCommand("MS", id);
24  }*/

```

Listing 7.2 CANbus controller module code structure

7.3.2 Continuous/Multiple Joints Motion Approach

The main difference between the single joint motion and multiple joints motion is to directing and delivering different commands to different joints at once (Fig. 7.9). This capability assists to build a platform for synchronizing the different joints motion at the same time.

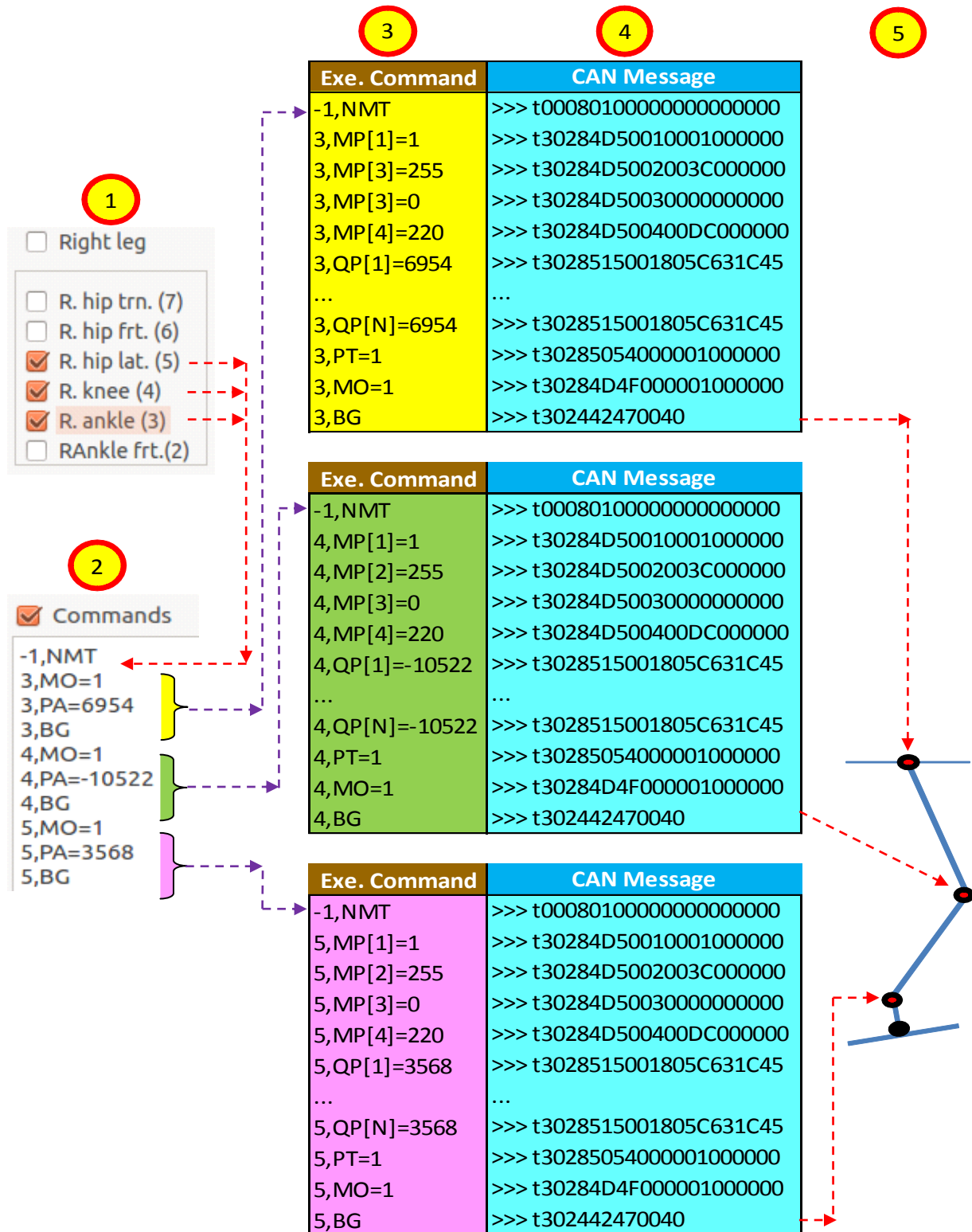


Fig. 7.9 Proposed implementation method for multiple joints motion

7.3.3 Synchronized Joints Motion Approach

Producing the coordinated movement of the Archie legs is the elementary step of the robot walking implementation. In order to synchronise the activities of all CAN nodes (joints) within a proposed network, a common time reference is needed. Time allocation and motion

scheduling are the main difference with the other above implementation (single and multiple). Each node has its own local time, which is a counter that is incremented each network time unit (NTU). The system wide NTU is derived from the node's local clock and local time unit ratio (TUR). This reference message (transmitted by a time master) restarts the cycle time in each node. The cycle time is derived from the node's local time.

Fig. 7.10 describes the synchronisation of the cycle time, performed in the same manner by all CAN nodes, including the time master (Milushev, 2010).

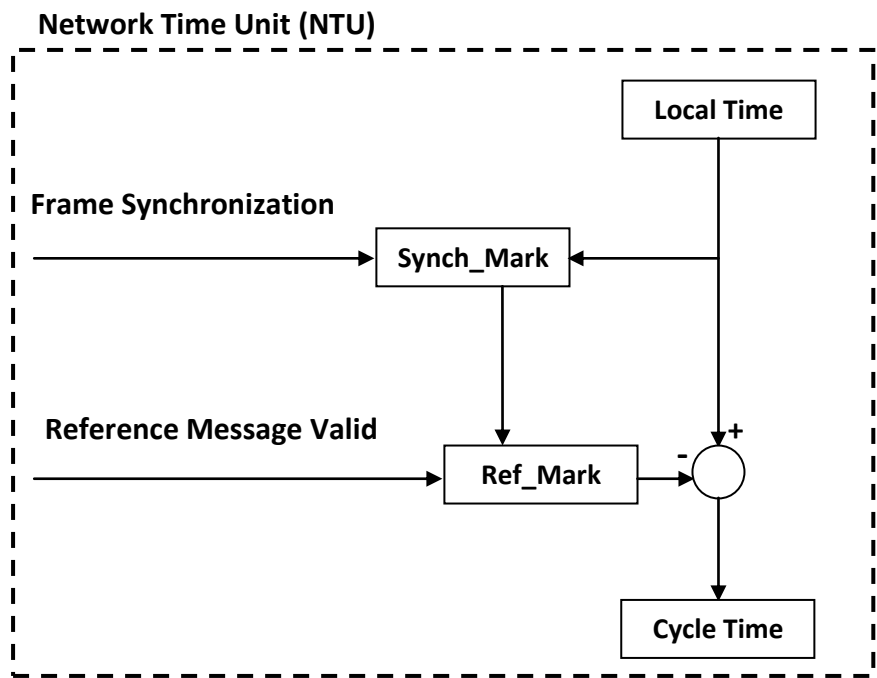


Fig. 7.10 Implemented cycle time for synchronized joint motion

Any received or transmitted message invokes a capture of the local time taken at the message's frame synchronisation (Hartwich et al., 2000). This frame synchronisation event occurs at the sample point of each start of frame (SOF) bit and causes the local time to be loaded into the Sync_Mark register. Whenever a valid reference message is transmitted or received, the contents of the Sync_Mark register is loaded into the Ref_Mark register. The difference between the actual value of the Ref_Mark and the local time is the cycle time (Cycle Time = Local Time – Ref_Mark).

The IDs of all messages have to be updated at every start of epoch (SOE). In previous synchronization method, priority inversion occurred so often since joint ID updated on different nodes and could not coincide exactly, and the ID of a low priority message is updated before that of a high-priority one. Then, for a small window of time, the low-priority message had a higher priority ID than the high-priority message. To overcome this problem an agreement protocol is used in new version to trigger the ID update on all nodes. The CAN clock synchronization algorithm synchronizes clocks to within 20µs. A simple agreement protocol can be that one node is designated to broadcast a message on the CAN bus. This

message will be received by all nodes at the same time and, upon receiving this special message, all nodes will update the IDs of their local messages. Instead of the older method, the following test protocol, which is not only robust, but also consumes less bandwidth is implemented. Each node has a periodic timer which fires every x seconds, at which time the node takes the following actions:

1. Set a flag to inform the CAN device driver that the ID update protocol has begun.
2. Configure the CAN network adapter (NA) to receive all messages
3. Increment the data length (DL) field of the highest-priority ready message on that node.

The first incremented-DL message to be sent on the CANbus will serve as a signal to all nodes to update the IDs of their messages. If the original DL of the message is less than 8, then incrementing the DL will result in transmission of one extra data byte. If the DL is already 8, CAN adapters allow the 4-bit DL field to be set to 9 (or higher), but only 8 data bytes are transmitted. Now, each node starts receiving all messages transmitted on the bus line.

The CAN device driver on each node has a table listing the IDs of all message streams in the system along with their data lengths. As messages arrive, the CAN device driver compares their DL field to the values in this table until it finds a message with an incremented DL field. All nodes receive this message at the same time and they all take the following actions:

1. Restore the receive filter to reenables message filtering in the NA.
2. If the local message whose DL field was incremented by the periodic timer has not been transmitted yet, then decrement the DL field back to its original value.
3. Update message IDs to reflect the new SOE.

Each node receives the incremented-DL message at the same time, so the ID update on each node starts at the same time. After the first incremented-DL message completes, the next-highest priority message begins transmission. As long as all nodes complete their ID updates before this message completes, all messages will have updated IDs by the time the next bus arbitration round begins and no priority inversion will occur. In case one or more nodes are slow and cannot complete the ID update within this time, all nodes can be configured to do the update while the n^{th} message after the first incremented-DL message is in transmission, where n is a small number, large enough to allow the slowest node to calculate all new IDs, and then just write these to the NA while the n^{th} message is in transmission. Reception of the first incremented-DL message causes the CAN device drivers to set the DL fields of their local messages back to their original values, but, before this can complete, the next transmission (also with an incremented DL field) has already started.

In this case each basic data transmitting cycle starts with a reference message that is used for synchronization purposes (Zuberi and Shin, 2000). When the nodes are synchronised, any message can be transmitted at a specific time slot, without competing with other messages for the bus. Thus the loss of arbitration is avoided, the latency time becomes predictable.

Synchronized joints motion structure, embedded in Archie motor drives:

```
CanbusViewer::checkForCheckedCB();
CanbusViewer::sendCommand("NMT");
stringstream ss;
for (unsigned int id=1; id<=sizeof(joint_nr)/sizeof(*joint_nr); id++) {
    if (joint_nr[id]) {
        CanbusViewer::sendCommand("MP[1]=1", id);
        ss.str("");
        ss << "MP[2]=" << numPoints_ ;
        CanbusViewer::sendCommand(ss.str(), id);
        if (ui->loop->isChecked()) {
            CanbusViewer::sendCommand("MP[3]=1", id);
        } else {
            CanbusViewer::sendCommand("MP[3]=0", id);
        }
        ss.str("");
        double st = ui->sample_time->text().toDouble();
        ss << "MP[4]=" << st;

        CanbusViewer::sendCommand(ss.str(), id);
    }
    for (unsigned int j=0; j<numPoints_; j++){
        ss.str("");
        ss << "QP[" << j+1 << "]= " << joint[id][j][0];
        CanbusViewer::sendCommand(ss.str(), id);
    }
    CanbusViewer::sendCommand("PT=1", id);
    CanbusViewer::sendCommand("MO=1", id);
}
}
for (unsigned int id=1; id<=sizeof(joint_nr)/sizeof(*joint_nr); id++) {
    if (joint_nr[id]) {
        CanbusViewer::sendCommand("BG", id);
    }
}
}
```

7.3 Preliminary Test

7.3.1 Half Gait Test

Based on the motion control structure which is already described and presented in chapter 6. (Fig. 6.14), this section presents the preliminary experiment performed on Archie. In this test the desired elliptical path (half gait) is generated according to defined step height and length. The desired values are presented in Fig. 7.11.

Step Curve	Ellipse Curve
Step Width (in millimeters)	230.00
Step Height (in millimeters)	100.00
Step Length (in millimeters)	300.00

Fig. 7.11 Input parameters for half gait test

The PT parameters are provided by reference command generator using this desired toe trajectory. Then these parameters are inputs to each drive controller. In order to show the performance of motion controller, actual trajectory of toe (end-effector) is needed. Therefore in each sample time, the position of toe is calculated using angle measured by encoder and forward kinematics of robot. The good performance of the designed creating motion system to track the desired elliptical trajectory of toe is depicted in Fig. 7.12. The lateral joint angle-trajectories are compared in Fig. 7.13.

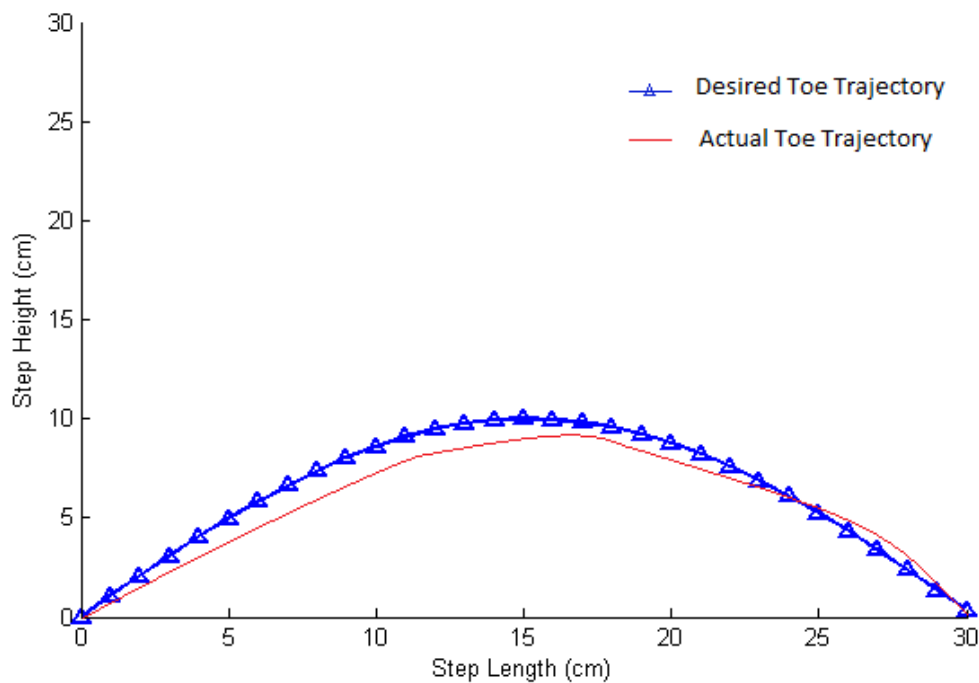


Fig. 7.12 Desired vs. actual toe trajectory

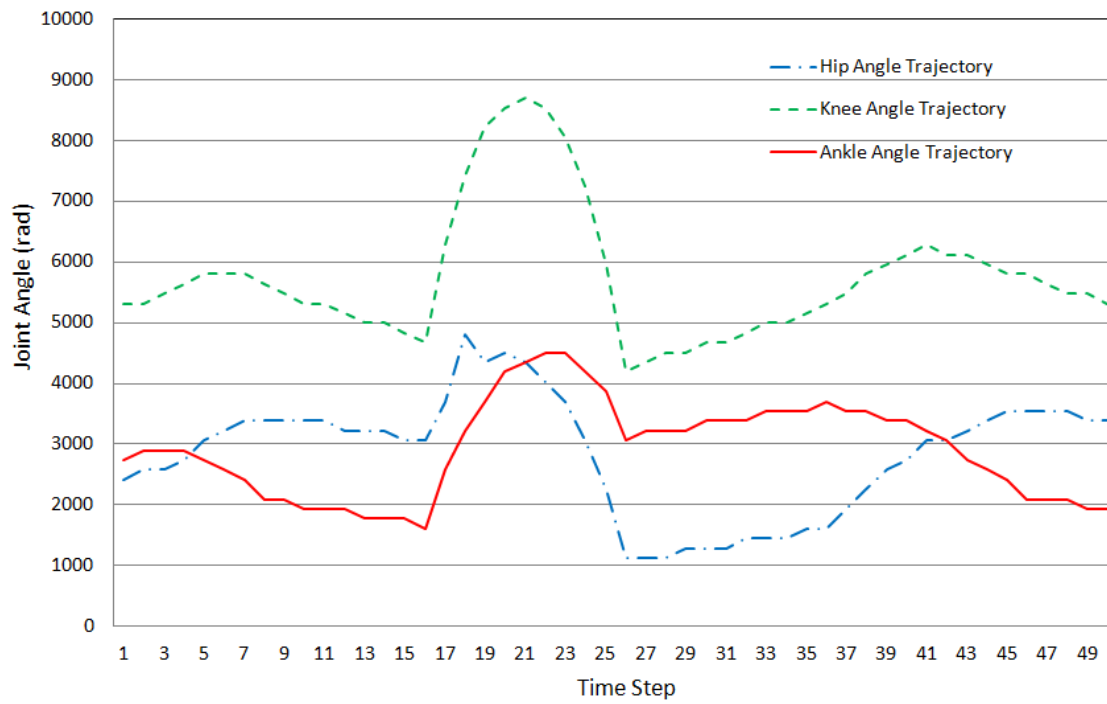


Fig. 7.13 Calculated hip, knee and ankle lateral-joints angle trajectories (step length=30 cm)

Accordingly in the following the simulation model, and half gait of this preliminary test on Archie are presented (Fig. 7.14)

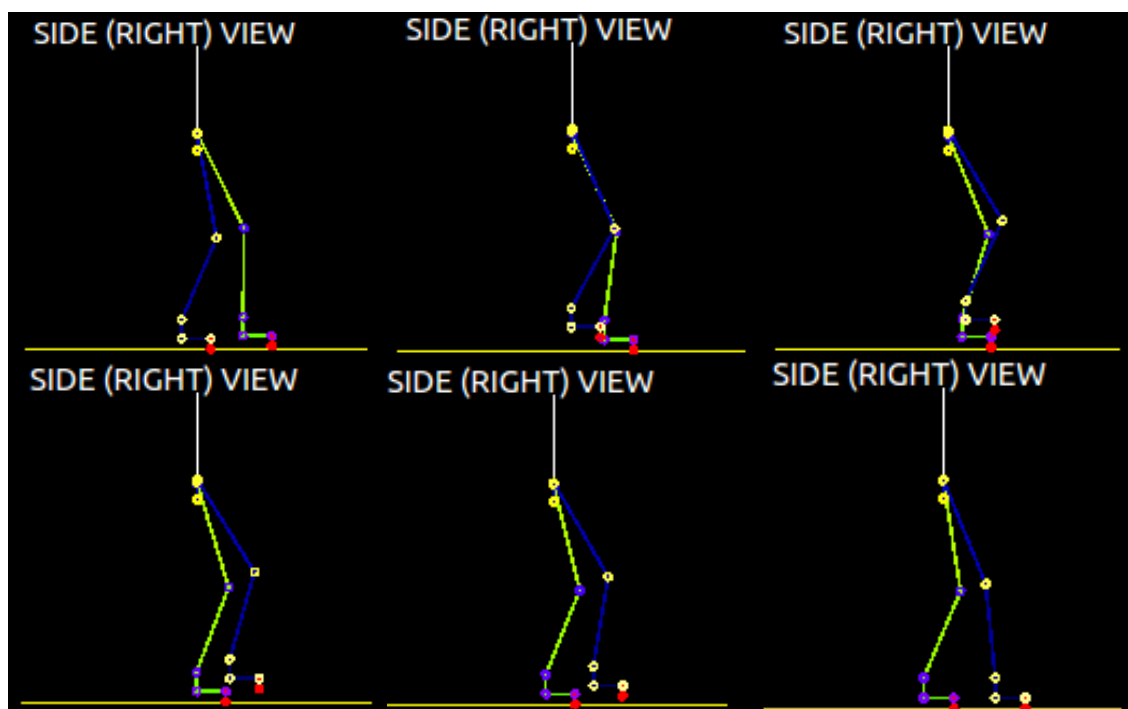


Fig. 7.14 Half gait simulation model (based on the elliptical trajectory)

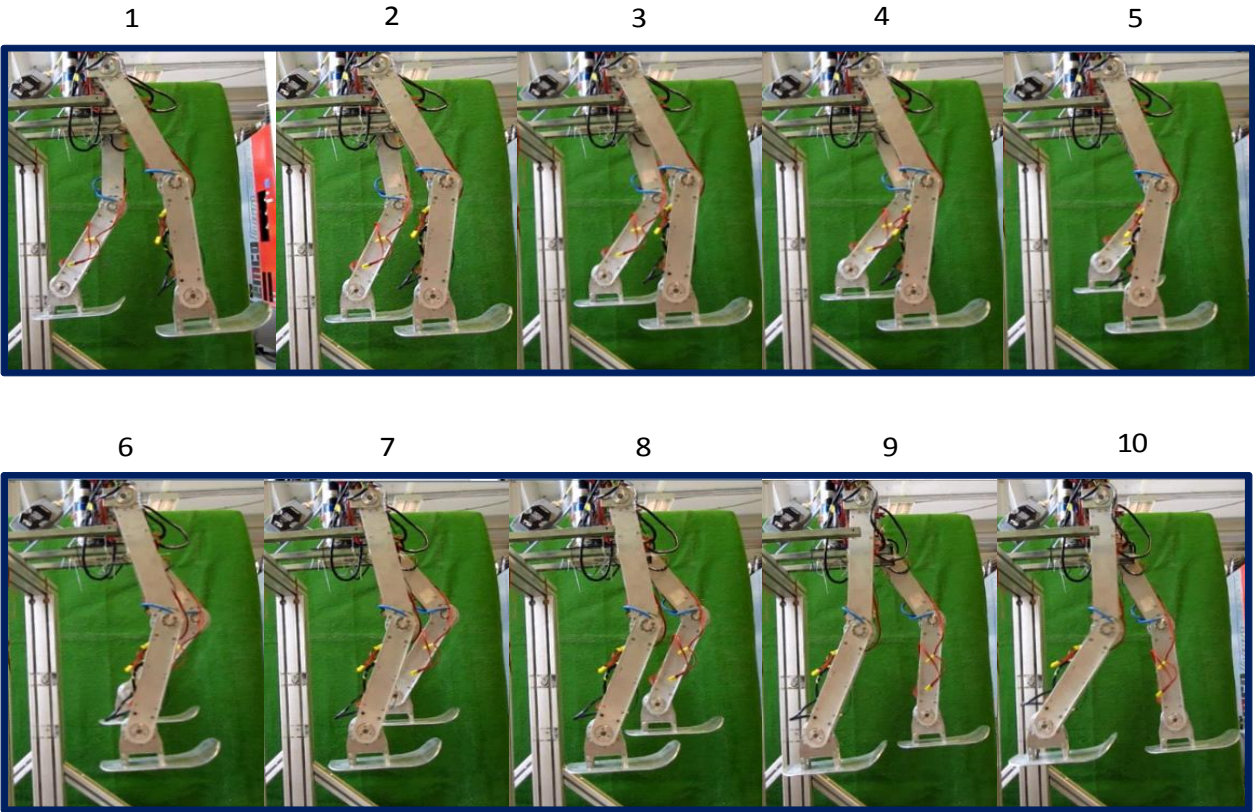


Fig. 7.15 Archie half gait performance

Frames 1 to 5 in Fig. 7.15 depict the left leg rising up, to max. step height, frame 6 presents the max. step height position (6 cm), and frames 7 till 10 present the moving down of the left leg to complete the half gait test.

a) Several Tests (with different input parameters)

The actual end-effector motion to provide different trajectories is tested (By attaching the marker to end-effector joint of Archie to provide trajectories on the board, and using different input parameters, Table 7.1) to realize the performance of the proposed motion creating system (Fig. 7.16).

Table 7.1 Desired step length, height and hip height to perform full gait

No. of trial	Step length	Step height	Hip height
1	30 cm	3 cm	53 cm
2	24 cm	3 cm	53 cm
3	20 cm	3 cm	53 cm
4	20cm	4 cm	51 cm

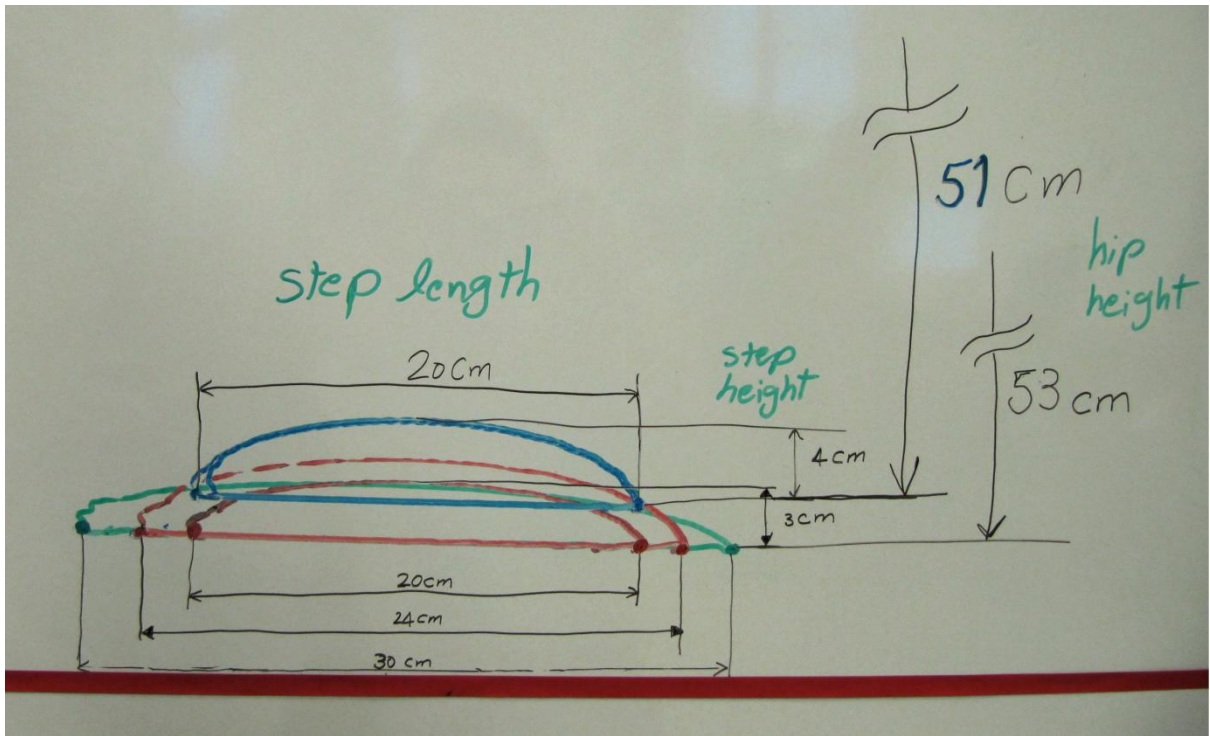


Fig. 7.16 Actual obtained trajectories of Archie (full gait motion test)

7.4 Static Walking Test

When the walking motion is done at a slow speed, the gait is called "static gait". In this approach Archie walking is involved static walking with a very low walking speed. From the definition of walking, at least one foot is always on the ground. While the center of mass in Archie is above the foot area, then if the robot is moving slowly enough (hence the name static), it is always stable. If the joints receives their own messages to move periodically such that the rear foot is lifted, moved forward, and put down, then a stable walking motion is obtainable: the robot walks without falling over (Fig. 7.17).

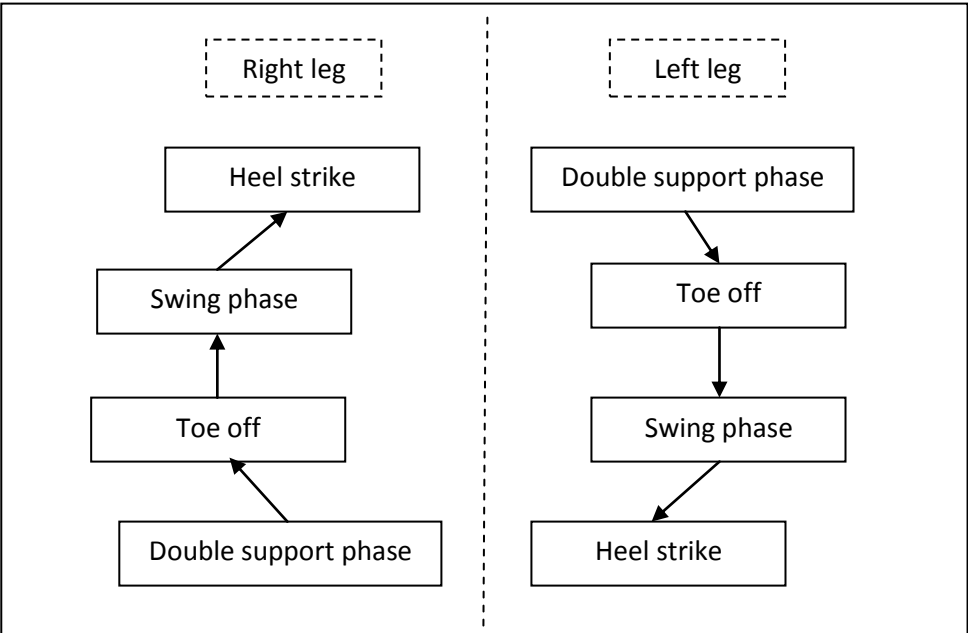


Fig. 7.17 Right and left leg phases during static walking test

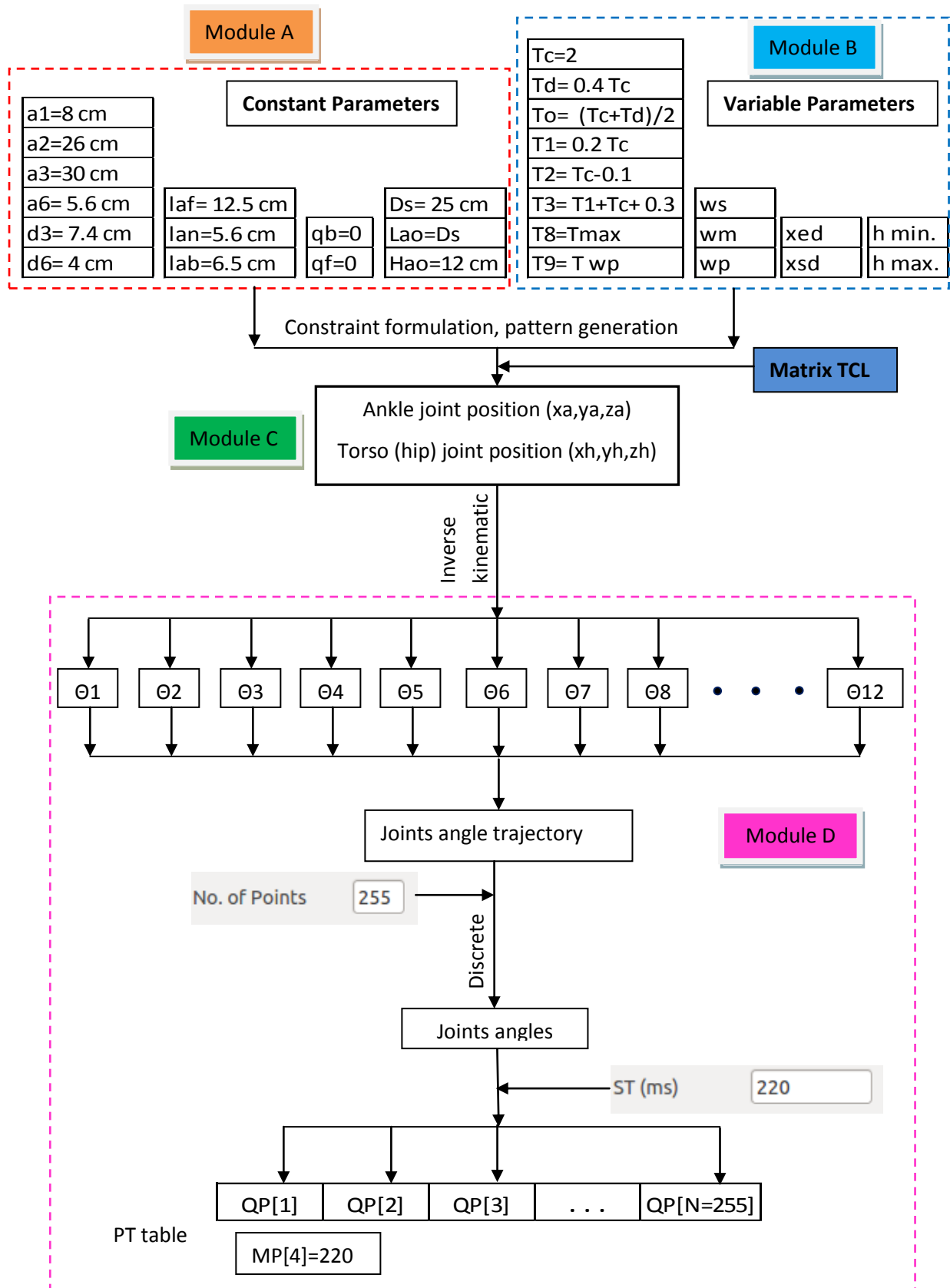


Fig. 7.18 Proposed software implementation approach for Archie walking

In this experiment approach, gait development is not directly based upon dynamic principles, rather it is more a method of building one step of the gait at a time. This is achieved by experimental modification of the gait by examining the performance of the gait on the robot. This approach, has no previously known data to compare the performance of the robot

against, and therefore we cannot generate a performance measure for the gait. Without such a quantitative measure we can only observe the robot while it is enacting the gait to determine the stability of the gait. Through many trials and examinations of the behaviour of the robot, the final optimised values are obtained. The MATLAB simulation results and robot posture based on the following parameters in module A, module B for stable forward walking are presented in Mohamadi Daniali M., 2013. The different stages which are implemented in motion controller software to perform synchronised motion and walking are described carefully in Fig. 7.18. The proposed implemented method are embedded in motion controller software (see Appendix B for source codes). The summarized description is depicted in Fig. 7.19.

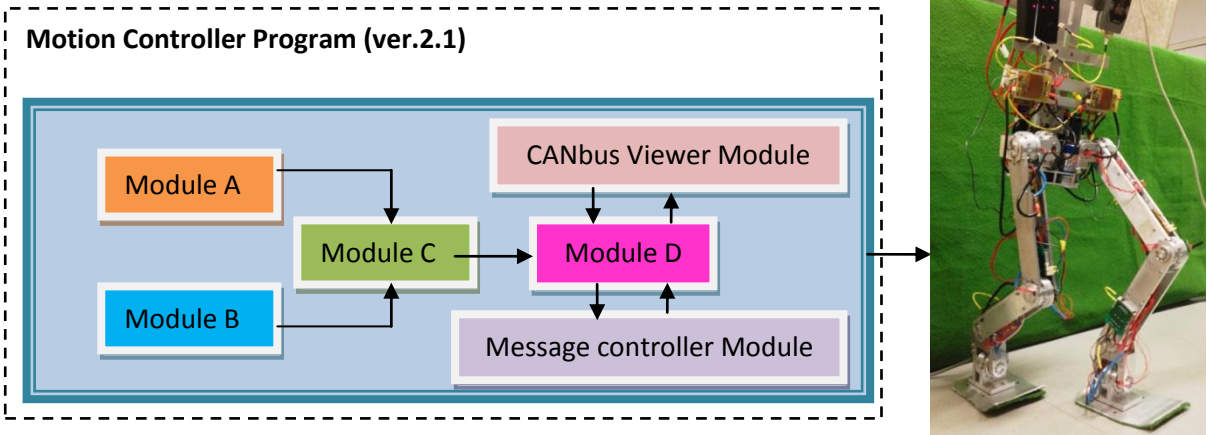


Fig. 7.19 New motion controller program implementation method to perform stable walking

7.4.1 Walking Simulation Results

In order to achieve the better understanding of the Archie walking phases and behavior, the simulation test is done. Each phase of the walking consists of a group of motions that need to be planned individually for each specific joint. For instance, within the single support phase, the planning of the hip, floating leg and supporting leg are required to allow the robot to move forward. The motion controller software allows the robot to perform forward static walking (Fig. 7.20 and 7.21).

- Left Step

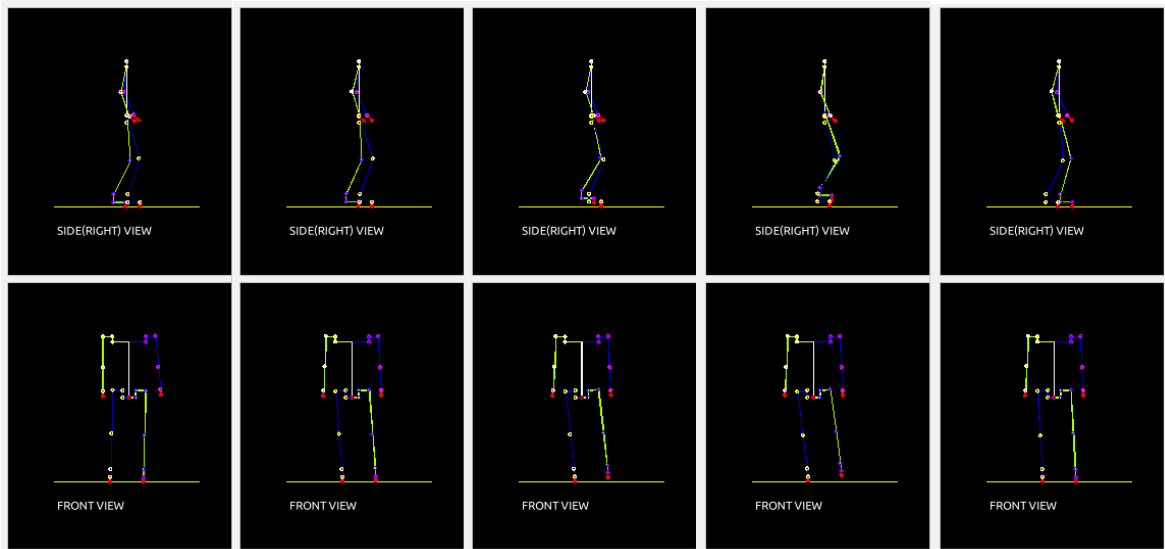


Fig. 7.20 Simulation of Archie left leg motion in forward static walking cycle

- Right Step

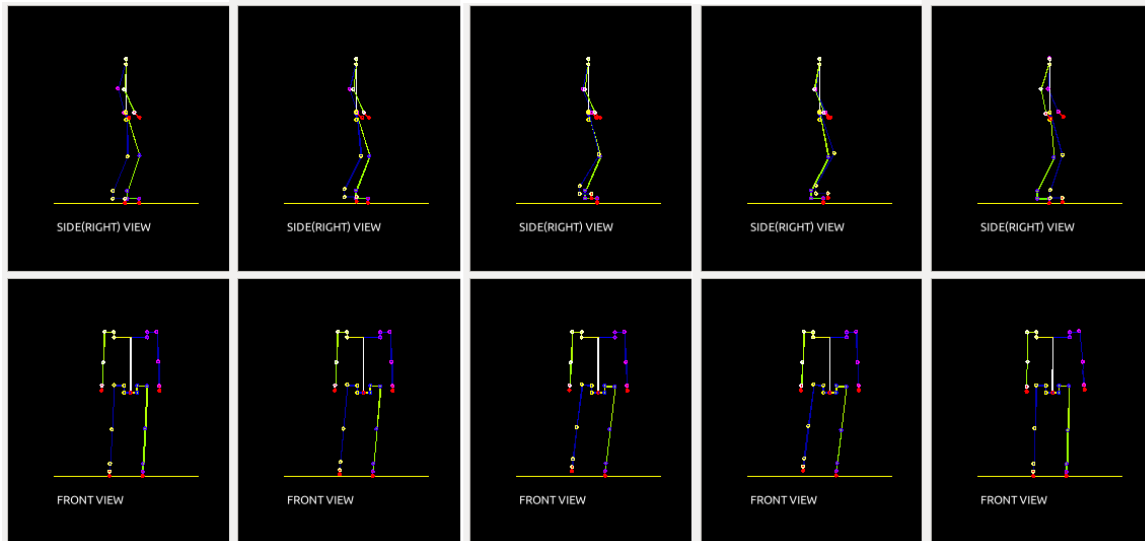


Fig. 7.21 Simulation of Archie right leg motion in forward static walking cycle

7.4.2 Stable Walking Realization Test

In the following the different sequences of stable walking test results with the selected input values (Table 7.2 and 7.3) and obtained QP vectors value (Table 7.4) are presented in Fig. 7.22, 7.23 (frame A1-9, frame B1-9). Based on the described implementation method in above, the modules are tuned and selected. In order to obtain the best performance and optimised values the D_s and H_{ao} parameters from module A and w_s , w_p , w_m , x_{ed} , x_{sd} , $h_{min.}$ and $h_{max.}$ from module B are modified and tuned through the different experimental tests.

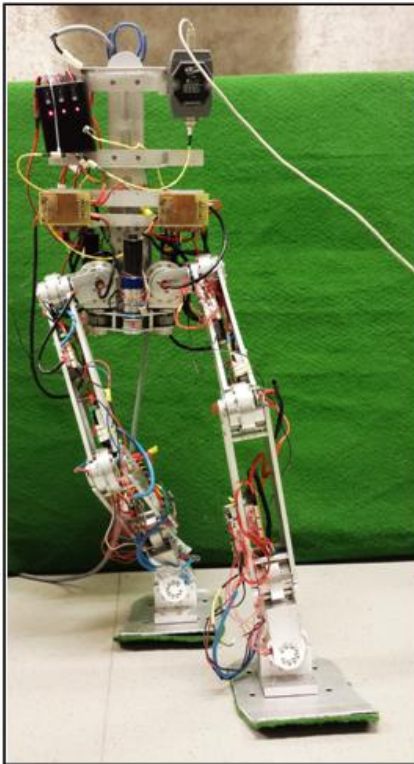
Table 7.2 Module A and B (selected parameters)

Module A			Module B
a1=8 cm	laf= 12.5 cm	Ds= 25 cm	ws= 17 cm
a2=26 cm	lan=5.6 cm	Lao=Ds	wm= 2 cm
a3=30 cm	lab=6.5 cm	Hao=12 cm	wp= 5cm
a6= 5.6 cm	qb=0		xed= 6.5 cm
d3= 7.4 cm	qf=0		xsd= -3.5 cm
d6= 4 cm			h min.= 54 cm
			h max.=57 cm

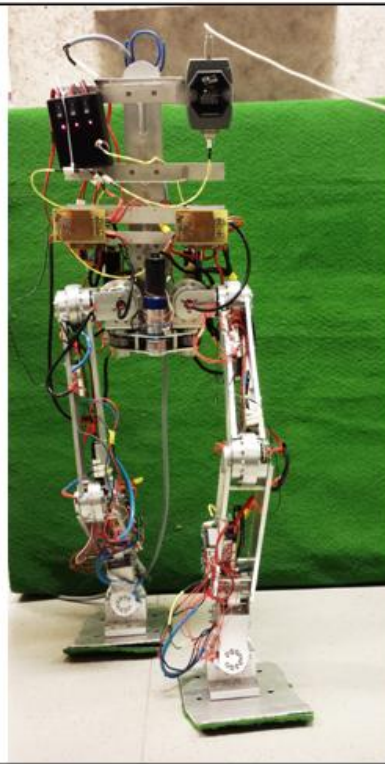
Table 7.3 Sequence of stable forward walking test (frames specification)

Frame No.	Time	Point No.
A1,B1	t=0	0
A2,B2	t= (Td)/2	27
A3,B3	t=Td	52
A4,B4	t=(Td+Tc)/2	90
A5,B5	t=Tc	128
A6,B6	t=Tc+(Td)/2	154
A7,B7	t= Tc+Td	179
A8,B8	t= Tc+(Tc+Td)/2	217
A9,B9	t=2Tc	256

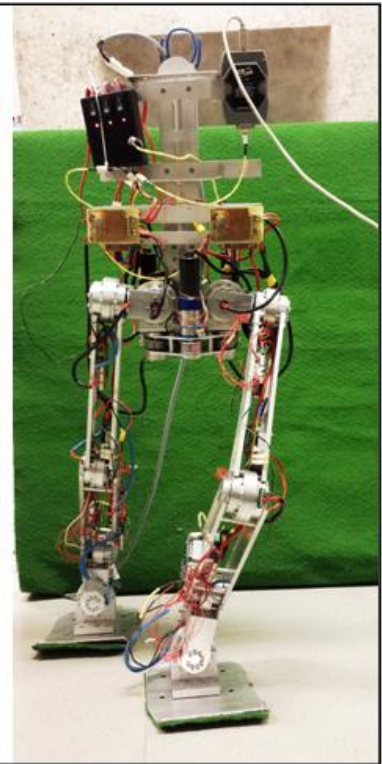
Frame A1



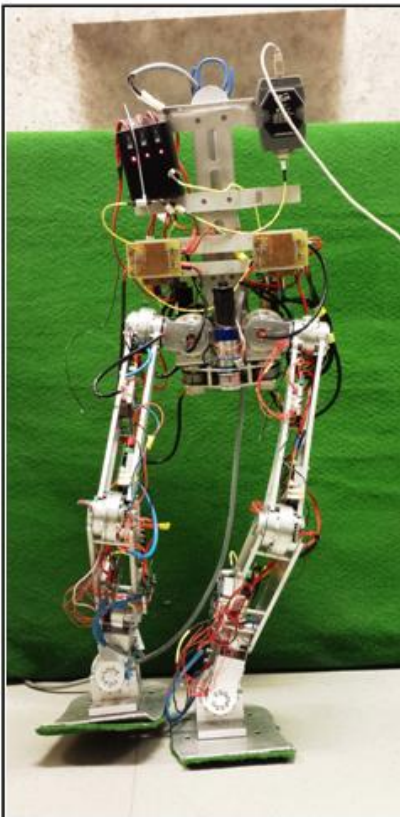
Frame A2



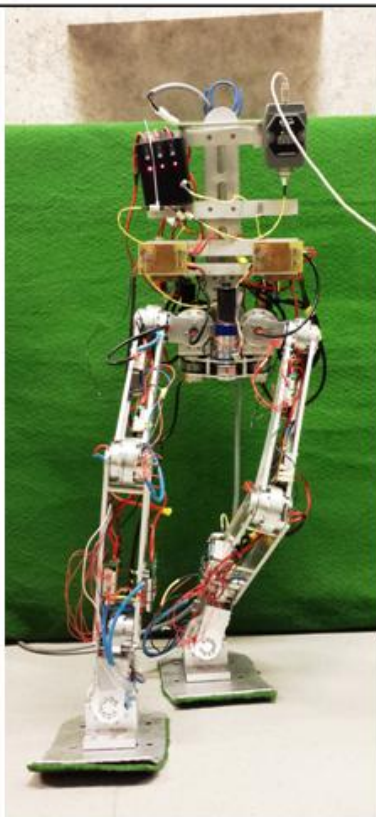
Frame A3



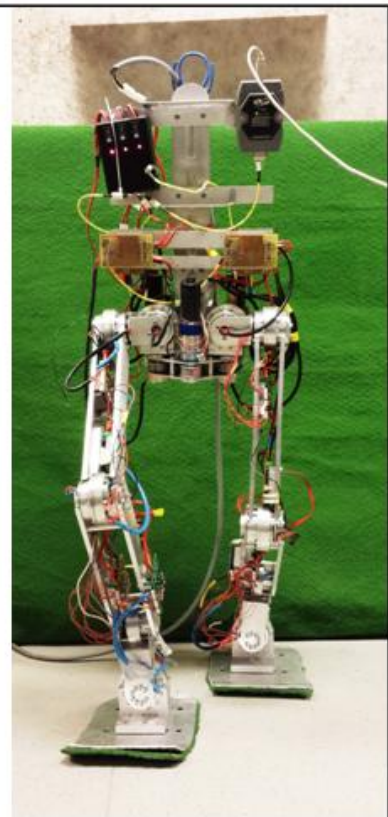
Frame A4



Frame A5



Frame A6



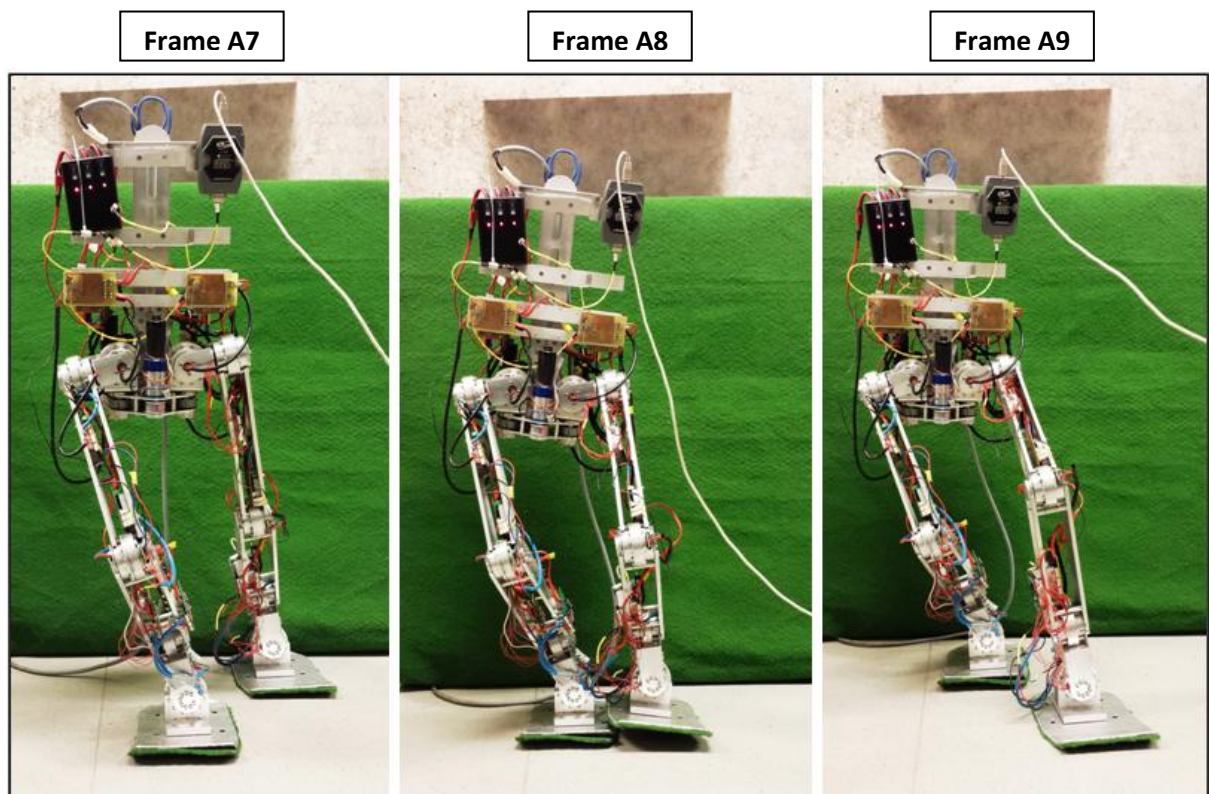
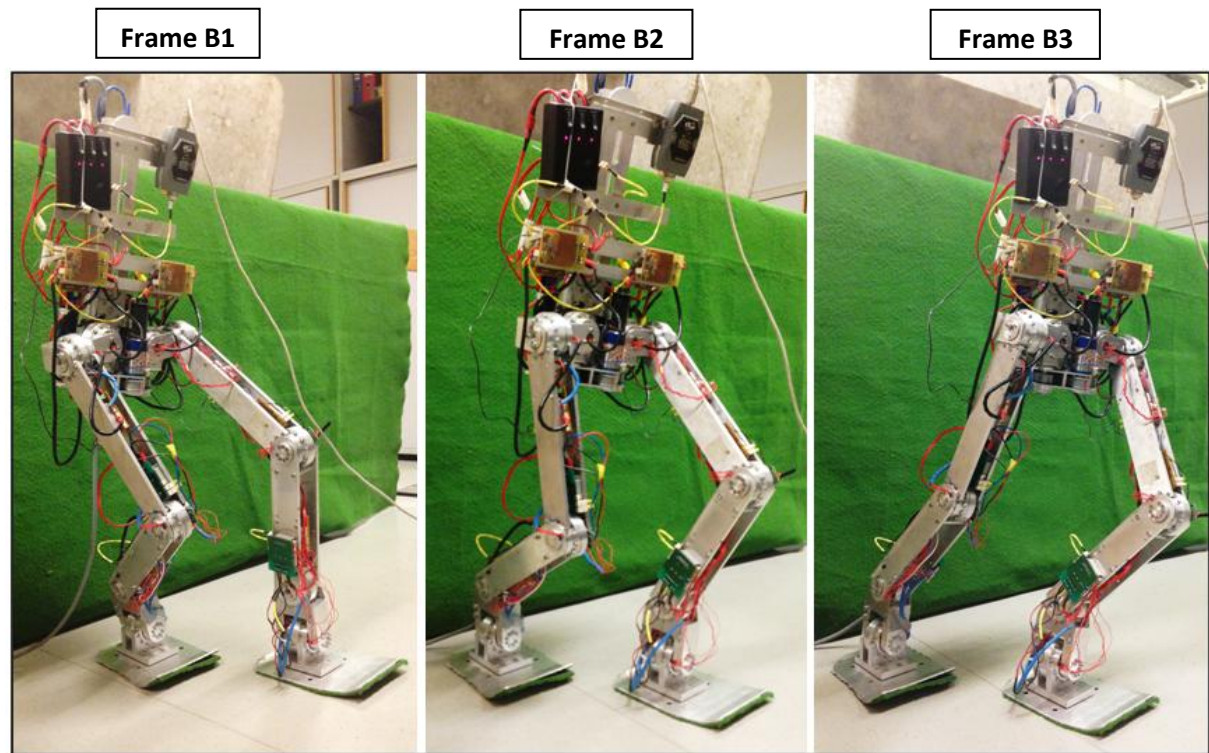


Fig. 7.22 Front view - sequence of stable forward walking of Archie (Frames A1-9)



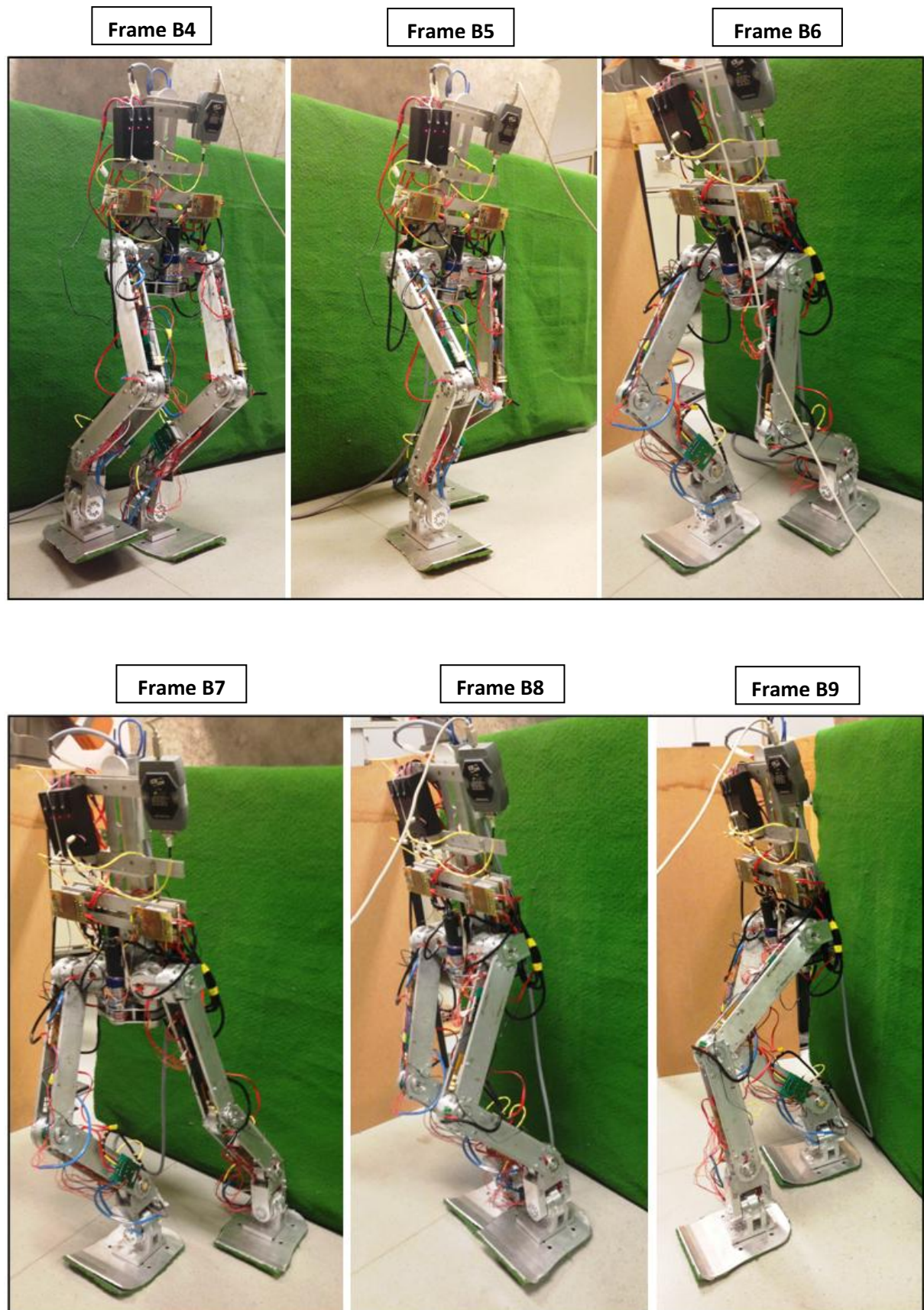


Fig. 7.23 Side view - sequence of stable forward walking of Archie (Frames B1-9)

Table 7.4 Calculated QP vector elements for each joint in static walking test (Module D)

Module D						
Joint	Frame A,B-1		Frame A,B-2		Frame A,B-3	
R-Ankle frontal	2,QP[0]=	2596	2,QP[0]=	742	2,QP[0]=	-1016
R-Ankle lateral	3,QP[0]=	7644	3,QP[0]=	8968	3,QP[0]=	6766
R-Knee lateral	4,QP[0]=	-11669	4,QP[0]=	-10108	4,QP[0]=	-3320
R-Hip lateral	5,QP[0]=	4025	5,QP[0]=	1140	5,QP[0]=	-3446
R-Hip frontal	6,QP[0]=	2596	6,QP[0]=	742	6,QP[0]=	-1016
R-Jip transversal	7,QP[0]=	0	7,QP[0]=	0	7,QP[0]=	0
L-Ankle frontal	9,QP[0]=	1033	9,QP[0]=	-829	9,QP[0]=	-2530
L-Ankle lateral	10,QP[0]=	1349	10,QP[0]=	4930	10,QP[0]=	6687
L-Knee lateral	11,QP[0]=	-9037	11,QP[0]=	-11814	11,QP[0]=	-11076
L-Hip lateral	12,QP[0]=	-7688	12,QP[0]=	-6885	12,QP[0]=	-4389
L-Hip frontal	13,QP[0]=	1033	13,QP[0]=	-829	13,QP[0]=	-2530
L-Hip transversal	14,QP[0]=	0	14,QP[0]=	0	14,QP[0]=	0
Joint	Frame A,B-4		Frame A,B-5		Frame A,B-6	
R-Ankle frontal	2,QP[0]=	-1993	2,QP[0]=	-1033	2,QP[0]=	829
R-Ankle lateral	3,QP[0]=	7643	3,QP[0]=	1349	3,QP[0]=	4930
R-Knee lateral	4,QP[0]=	-12219	4,QP[0]=	-9037	4,QP[0]=	-11814
R-Hip lateral	5,QP[0]=	4575	5,QP[0]=	7688	5,QP[0]=	6885
R-Hip frontal	6,QP[0]=	-1993	6,QP[0]=	-1033	6,QP[0]=	829
R-Jip transversal	7,QP[0]=	0	7,QP[0]=	0	7,QP[0]=	0
L-Ankle frontal	9,QP[0]=	-2467	9,QP[0]=	-2596	9,QP[0]=	-742
L-Ankle lateral	10,QP[0]=	6440	10,QP[0]=	7644	10,QP[0]=	8968
L-Knee lateral	11,QP[0]=	-10145	11,QP[0]=	-11669	11,QP[0]=	-10108
L-Hip lateral	12,QP[0]=	-3705	12,QP[0]=	-4025	12,QP[0]=	-1140
L-Hip frontal	13,QP[0]=	-2467	13,QP[0]=	-2596	13,QP[0]=	-742
L-Hip transversal	14,QP[0]=	0	14,QP[0]=	0	14,QP[0]=	0
Joint	Frame A,B-7		Frame A,B-8		Frame A,B-9	
R-Ankle frontal	2,QP[0]=	2530	2,QP[0]=	2467	2,QP[0]=	2596
R-Ankle lateral	3,QP[0]=	6687	3,QP[0]=	6440	3,QP[0]=	7644
R-Knee lateral	4,QP[0]=	-11076	4,QP[0]=	-10145	4,QP[0]=	-11669
R-Hip lateral	5,QP[0]=	4389	5,QP[0]=	3705	5,QP[0]=	4025
R-Hip frontal	6,QP[0]=	2530	6,QP[0]=	2467	6,QP[0]=	2596
R-Jip transversal	7,QP[0]=	0	7,QP[0]=	0	7,QP[0]=	0
L-Ankle frontal	9,QP[0]=	1016	9,QP[0]=	1993	9,QP[0]=	1033
L-Ankle lateral	10,QP[0]=	6766	10,QP[0]=	7643	10,QP[0]=	1349
L-Knee lateral	11,QP[0]=	-3320	11,QP[0]=	-12219	11,QP[0]=	-9037
L-Hip lateral	12,QP[0]=	3446	12,QP[0]=	-4575	12,QP[0]=	-7688
L-Hip frontal	13,QP[0]=	1016	13,QP[0]=	1993	13,QP[0]=	1033
L-Hip transversal	14,QP[0]=	0	14,QP[0]=	0	14,QP[0]=	0

7.4.3 Forward Walking Test Results (10 m)

Based on the selected parameters of the walking pattern (defined in Table 7.4) the forward stable walking experiments were performed on an ordinary room floor, and Archie walked

forward for 10 meters. Robot could perform the motion keeping balance along the entire sequence (Fig. 7.24).



Fig. 7.24 Archie stable forward walking performance in 10 meters

7.4.4 Optimized W_m Value (Test Result)

In order to achieve the proper value of the W_m during stable walking in the proposed motion creating system, several tests with different W_m value are done (Table 7.5). The results are depicted in Fig. 7.25, as it realized in the frame C4, in this position Archie is on the border of the stable feasible region, since it is a threshold point of falling, it means by increasing the last W_m value ($W_m=5\text{cm}$) Archie will fall down during the static walking.

Table 7.5 Different W_m value

Frame No.	Wm
C1	0
C2	1 cm
C3	2 cm
C4	5 cm

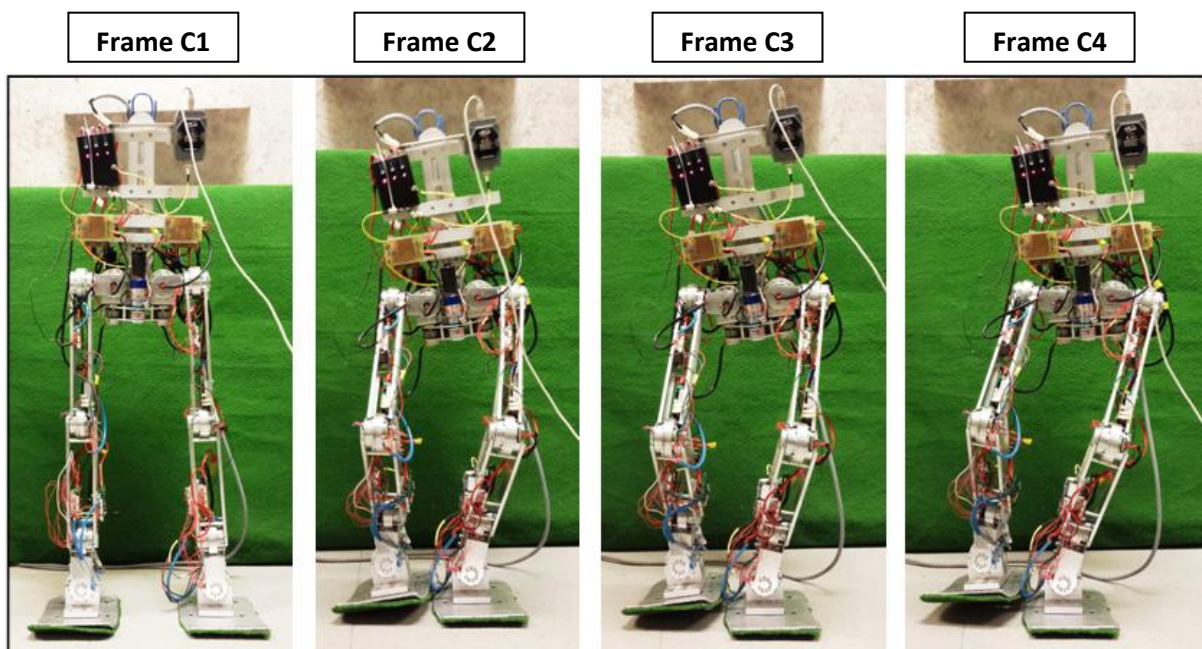


Fig. 7.25 Archie posture based on the different W_m (find an optimised value)

Based on the min., max. and optimised value of W_m the calculated angle for different joints during stable walking of the robot are compared and depicted in Fig. 7.26, 7.27, 7.28 and 7.29).

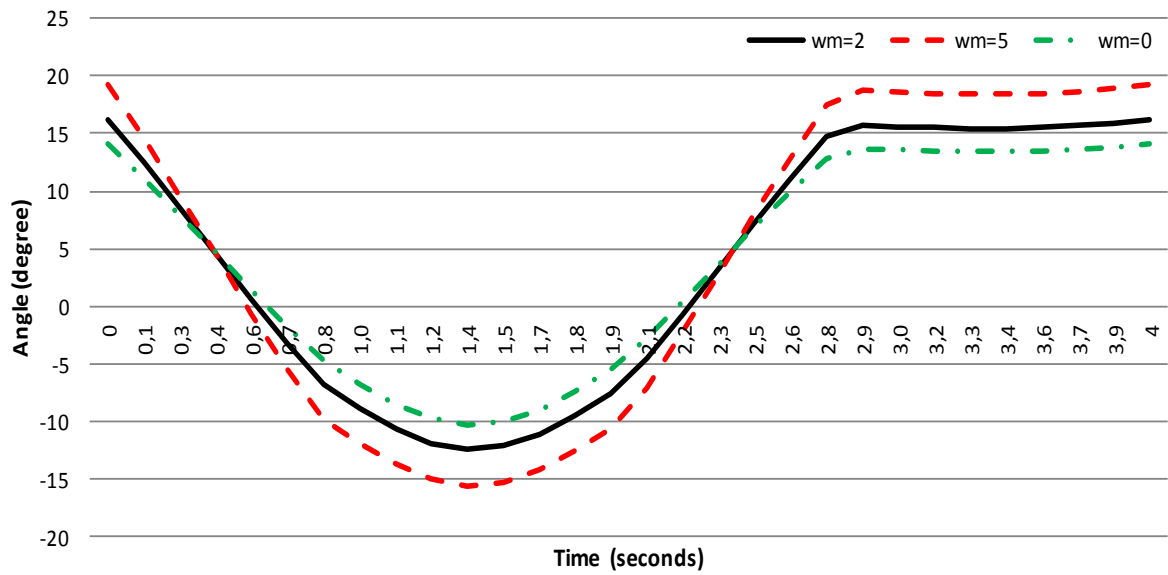


Fig. 7.26 Ankle frontal joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$

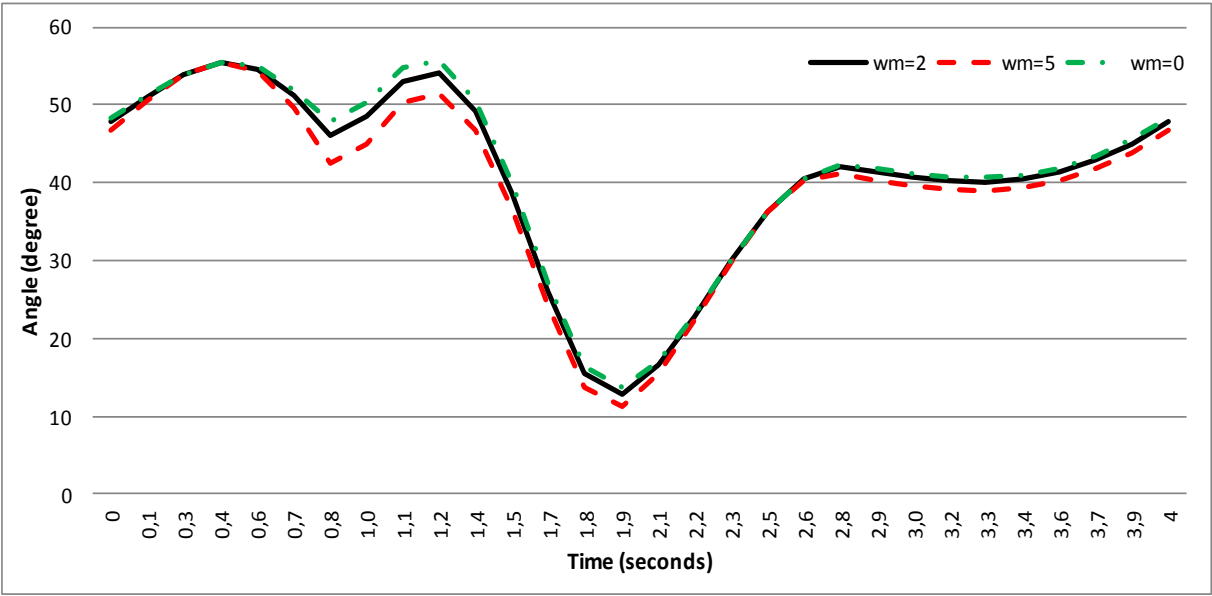


Fig. 7.27 Ankle lateral joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$

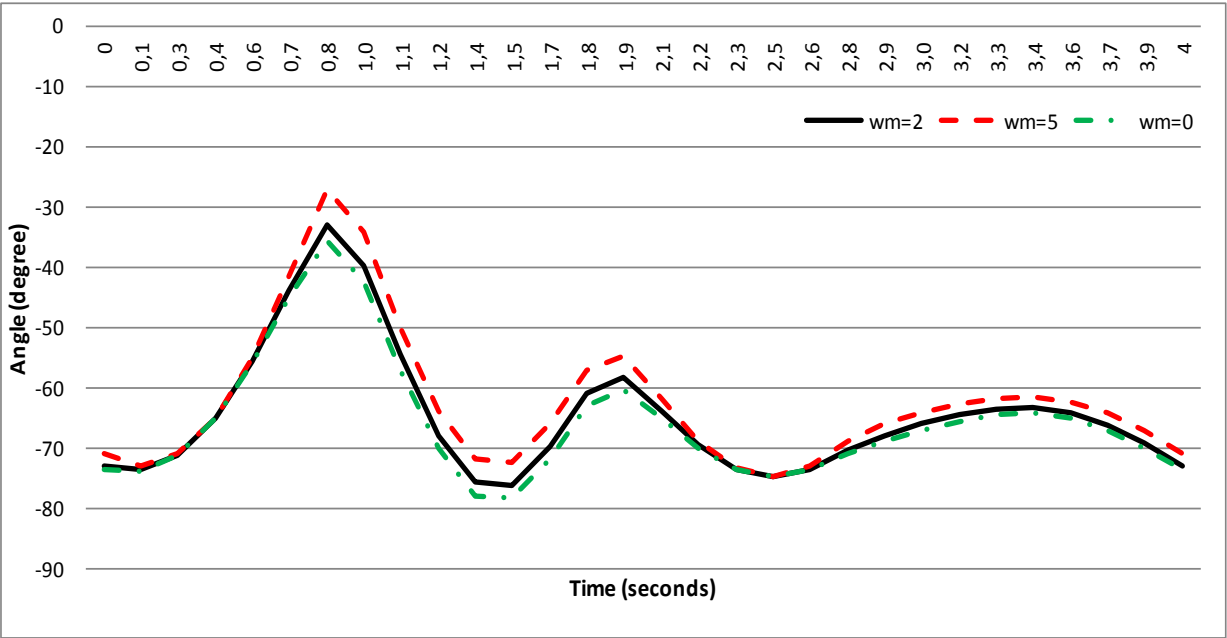


Fig. 7.28 Knee joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$

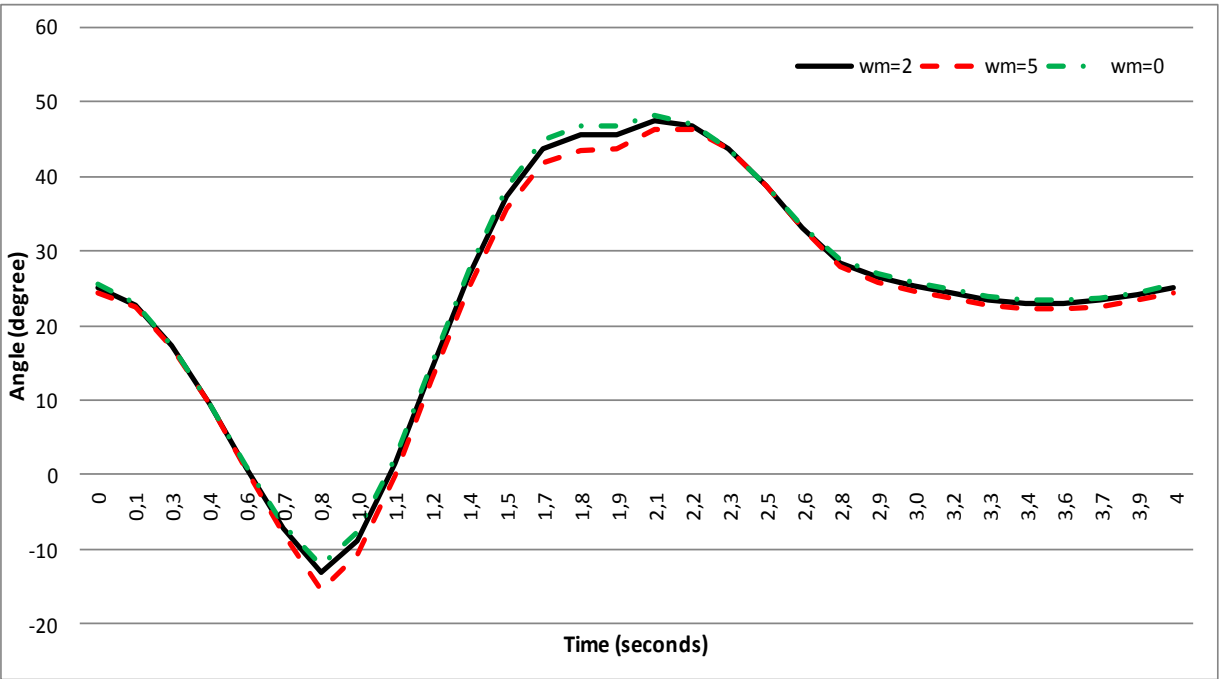


Fig. 7.29 Hip lateral joint-angle trajectory with $W_m(\text{optimised})=2\text{cm}$, $W_m(\text{min.})=0$, $W_m(\text{max.})=5\text{cm}$

Chapter 8 Summary, Outlook and Proposed Future Works

8.1 Summary and Outlook

This thesis presented the new design concept for motion creating system (both in hardware and software) including a method to formulate the torso and ankle motion, a method to manage the robot joint motion based on the Position-Time (PT), and specifying a sequence of absolute position with equal sampling time and fully described implementation approach for static walking of biped humanoid robot Archie. The developed robot comprises a set of simple kinematics joints mimicking the most important degrees of freedom of the human body (with the realized 12 DOF on lower body).

Archie biped robot is designed with 125 cm height, light weight (~20 kg), brushless and brushed motor as actuators, magnetic rotary encoder as sensors, CAN communication protocol as a communication network unit, and COSEL ADA1000F (36V,28A) as a main power supply. Motor drive, harmonic drive and hall sensor/switch, PCB mounted drive controller (Elmo) are the main components of its joint.

Since the autonomous characteristic needs the high performance of the communication network, the proposed CAN communication protocol is utilized as a distributed control network. Therefore fast calculation, fast communication and real-time data exchange via new proposed communication/control system for a biped robot Archie is developed and implemented.

A designed real-time message/command consists primarily of an id, which represents the priority of the message, and up to eight data bytes which is transmitted serially onto the bus. The messages are queued in the stations before being transmitted to the CAN bus. In the implemented proposed CAN protocol, stations wait until the bus idle period is detected before attempting to transmit. The use of the identifier as priority is the most important part of CAN message design with respect to real-time performance. Therefore the proposed communication protocol, efficiently supports distributed real-time control with a very high level of security. It provides the error process mechanisms and message priority concepts.

In implemented real-time system two wires in its cable are used, one for a dominant state and one for a recessive state. With CAN bus cable, the interpreter commands are sent in binary form and are used for setting and retrieving all numerical data of the simple drives motion setup. The commands are used to instruct the drive to perform a sequence and synchronized motion.

Thereafter motion control architecture was explained. In this controller there are different sections; static walking pattern planning, trajectory calculation algorithm (reference command generator) using inverse kinematics, distributed control drive for each joint using three cascaded loops as position, velocity and current to control individual joints precisely and central control program using PT, PVT table and CAN communication capability.

In PT mode a set of points are generated to be visited at certain time. The driver interpolates a third-order polynomial between the defined position points. PT data are transferred to the drive online using an efficient real-time message network, so that infinite PT motions are possible. The PT table allows for the performance of both pre-designed and online motion plans by writing the QP vector while PT motion is executing. The CAN communication protocol allows much faster PT programming, by packing two position points into one PDO communication packet. For easy synchronization with the joints, the drive may be programmed to send the PT read and write pointers continuously to the robot joint as a synchronous PDO.

PVT motion design is independent of controller sampling time. A PVT motion can be referenced in absolute time by requiring it to start at a specified time. It allows absolute time specification so that several drives may compose a fully-synchronized motion.

By embedding the PT and PVT method in motion creating system, single, multiple and synchronized joints motions of robot are achievable. In order to implement these modes, we designed and created a motion library command. Therefore the motion control program (software) is designed (ver. 2.1) based on the proposed controller system to improve motion planning, stable walking and minimize the computational efforts which devoted for generating smooth walking motion. It is a centralized motion program (software) in order to construct a platform where user can easily develop and debug perception–action coupling control schemes. Motion commands are transmitted from the PC/Laptop to the motor drivers directly via distributed CAN network. Motion control program (ver. 2.1) includes new GUI, that the executed commands are online monitored. Execution monitoring is needed in implementation in order to handle problems caused by uncertainties, both in the robot hardware itself and in the generated designed CAN messages. Uncertainties that could be solved in this program can be classified as :

- Missing complete information (lack of enough data for generating the joints trajectory comparing to older software version (1.1),
- Unreliable resources such as broken driver, cable loos connection,
- Fault or error codes based on the Elmo drive

On the other hand constraints of a complete foot trajectory are formulated and the foot trajectory by third spline interpolation are generated in new proposed motion control program (ver. 2.1). In this manner by adjusting the values of the constraint parameters, different types of foot motion are provided in order to adapt to ground conditions. Smooth torso motion with the largest stability margin are formulated. While both foot trajectories and the torso trajectory are known, all joint trajectories of the biped robot will be determined by kinematic constraints. In this case the designed motion creating system is ready to be implemented and tested on the robot. The new developed biped humanoid robot Archie (both in hardware and software) allows testing and evaluation of new ideas and concepts for stable static walking. Based on designed motion creating system dozens time, different

experiments such as single, multiple, synchronized joints motion, right and left leg motion, standing on one leg (to find a feasible region of stability) and finally static walking test are done to support our aim and experimentally confirmed its effectiveness. Currently Archie walks stable for 10 meters with the maximum step length of 50 cm, max. step height of 12 cm and speed of ~ 0.08 km/h based on the proposed motion creating system.

8.2 Proposed Future Works

This section suggests and lists several approaches that should be considered as the future works and ongoing strategies to make Archie as an advanced autonomous biped robot.

In the first primary stage completing the Archie hardware structure (construction of the upper body) is a must to enhance performing of stable walking. On the other hand adding vision sensors that can passively detect wide areas of an environment are necessary for autonomous locomotion.

The robot should be equipped with a stereo camera that helps to build a map reference relative to its surroundings. Since the ability to build real-time maps for autonomous robot navigation is essential.

Stereo vision is currently one of the reasonable solutions for this kind of sensor. Its widespread acceptance motivates effort in a difficult and expensive but compelling discipline: the building of human-like robots. The cognition system (vision based) for Archie could include the following features to complete Archie human-like motion and behaviour:

- Humanoid robot vision
- Imitation learning
- Recognition of human activities (objects recognition)
- Learning new skills and concepts from interaction with an environment and user

Future studies could be aimed at developing the motion control program's GUI. Developing the more accurate simulation model and adding features such as:

- Adding 3D vision, by adding camera in virtual environment, the robot motion could be simulated in 3D space, hence the walking performance could be evaluated from different perspective,
- Improving the real time data communication between PC/operator and robot by installing wireless connection and establishing online data transmission to robot (remote control system via wireless connection),
- Integrating 3 different interfaces for Archie locomotion; robot control interface (including graphical walking trajectory generation), environment interface (in order to remotely control the robot motion) and human interaction interface (including face recognition, voice recognition),

will increase real time performance, reliability, efficiency and safety, but on the other hand these improvements increase the cost and complexity of the system.

8.3 New Proposed Approaches and Requirements for Dynamic Walking

This section presents the proposed suggestions to improve and enhance the dynamic walking performance in Archie based on adding force/torque sensor and upper body construction.

The main control problems with a biped robot control system are the stability to avoid falling down in any posture, the capacity of the mechanism to absorb the impact during foot landing and the adaption to any surface and the attitude for maintaining the reference biped orientation.

Adequate sensors and control algorithm should be employed to obtain real stable walking motion from the reference patterns. The combination of inertial (such accelerometers and gyroscopes) and force sensors should be used to feed back external disturbances, due to terrain irregularities, structure imperfections, inertial and gravity effects by measuring the actual attitude and forces on each foot (and designed upper body) during the double support phase, walking motion or cooperation tasks. Otherwise, structural compliances should be mounted in order to mechanically reduce the impact effects (due to working precision, design and terrain imperfections), normally in the soles of the feet.

Furthermore, suitable reference walking patterns in order to distribute the biped mass during walking motion to maintain stability (by taking into account the dynamics effects) and reduce the impacts are required. These developed walking patterns based on dynamic behaviour of Archie are the starting point of the improvements based on the current proposed (designed) hardware, software and the control system of Archie.

8.3.1 How to Improve the Stability Problem ? (Proposed Algorithm Method)

The ZMP criteria affirms that the biped robot does not fall down if the ZMP is maintained inside of the convex hull during the walking motion. Therefore, it is necessary to add a control loop of ZMP to obtain real stable walking motion. To compute the ZMP online, force/torque sensors should be used and this information feedback to the respective control loop (Yang, 2006). The ZMP calculation equation for Archie based on the dynamic walking pattern is proposed as follows:

$$x_{zmp} = \frac{\sum_{i=1}^n m_i(\ddot{z}+g)x_i - \sum_{i=1}^n m_i \ddot{x}_i z_i - \sum_{i=1}^n I_{iy} \ddot{\Omega}_{iy}}{\sum_{i=1}^n m_i(\ddot{z}+g)} \quad (8.1)$$

$$y_{zmp} = \frac{\sum_{i=1}^n m_i(\ddot{z}+g)y_i - \sum_{i=1}^n m_i \ddot{y}_i z_i - \sum_{i=1}^n I_{ix} \ddot{\Omega}_{ix}}{\sum_{i=1}^n m_i(\ddot{z}+g)} \quad (8.2)$$

where the m_i is the mass of link i , and I_{ix} and I_{iy} are the inertial components, $\ddot{\theta}_{ix}$ and $\ddot{\theta}_{iy}$ are the absolute angular velocity components around x-axis and y-axis at the center of the gravity of link i , g is the gravitational acceleration, $(x_{zmp}, y_{zmp}, 0)$ is the coordinate of the ZMP, x_i, y_i, z_i is the coordinate of the link i on an absolute Cartesian coordinate system.

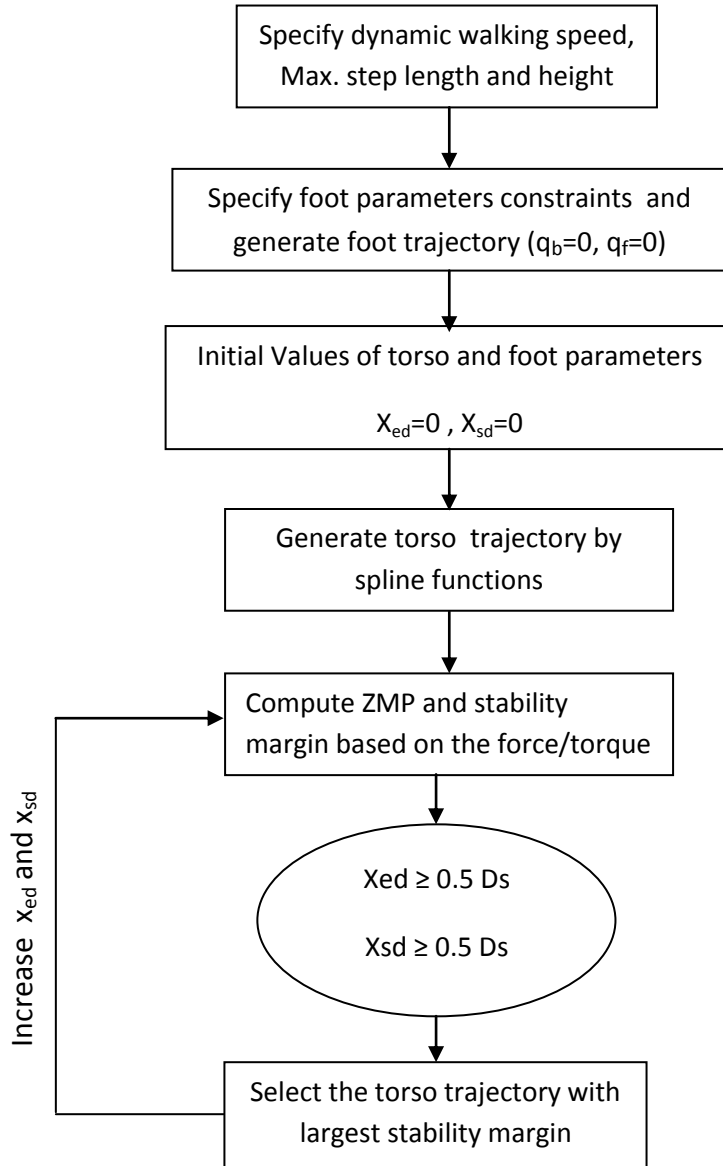


Fig. 8.1 Proposed dynamic walking algorithm for enhancing the stability margin based on the online ZMP calculation from force/torque sensor

The proposed dynamic walking algorithm for enhancing the stability is presented in Fig. 8.1. If the ZMP is within the convex hull of all contact points (the stable region), the biped robot is able to walk. If the minimum distance between the ZMP and the boundary of the stable region is large, the moment preventing the biped robot from tipping over is large. The

minimum distance d_{zmp} between the ZMP and the boundary of the stable region is called the stability margin. The advantage of this method is that the stability margin can be large enough if the desired ZMP is designed near the center of the stable region. However, since the change of the ZMP due to torso motion is limited, not all desired ZMP trajectories can be achieved. Furthermore, to achieve a desired ZMP trajectory, the torso acceleration may need to be large. In this case, since the torso is relatively massive, energy consumption increases, and control for task execution of the upper limbs becomes much more difficult.

In order to maintain its stability, the robot's center of gravity (in the case of static stability) or the ZMP (in the case of dynamic stability), must be transferred from the rear foot to the front foot during the short double-support phase. On the other hand, if the interval of the double-support phase is too long, it is difficult for the biped robot to walk at high speed. It is concluded that the dynamic stable "biped gait" should take into account the whole body dynamics to make any motion on any surface.

8.3.2 Absorbing Impact and Adapting to a Surface Problem

Imperfections in the walking surface and changing the support foot while walking drastically cause force variations on the landing foot of Archie. The compliance control loop should be implemented to adapt Archie to these changes. Furthermore, the compliance structural design should be developed in order to cushion the impacts. The force/torque sensor should be embedded to feedback the external reaction on lower (or designed upper body).

8.3.3 Attitude Problem

During fast walking motion, the dynamics and gravitational effects cause tipping torques which cause the Archie fall down. Furthermore, structural imperfections cause important flexion on some joints such as ankle and hip (during fast walking test). As the reference patterns include the attitude because the biped should walk straight forward, all the time the control attitude loop should be implemented to obtain real stable human walking.

The suitable Inertia Measurement Unit-IMU (to allow the biped robot to return to its desired posture (position)), gyros and accelerometers should be used to compute the actual robot attitude and it is feedback to the biped current control system. Real waking requires high stability to provide antigravity support of body weight, mobility of body segments and motor control to sequence multiple segments while transferring body weight from one limb to another.

8.3.4 Proposed Upper Body Motion for Dynamic Walking

Since upper body is massive, the changes of the upper body motions greatly affect the stability. In this section, the upper body motion affects and its relationship with hip motions both is discussed.

a) Slope Angel of the Upper Body

While the Archie upper body construction is completed, as the upper body sways during human walking, the upper body can also swings for Archie walking. In the current proposed static walking pattern generation for Archie (chapter 5) the torso motion parameter $\Theta_h(t)$ is constant ($\Theta_h(t)=0$). In that case, since the massive size of upper body, at the beginning or the end of the double-support phase, the center of gravity will move to the front foot in a very short period particularly for high speed walking (Yang et al., 2006). The impact force between the robot and the ground may become very large, and the biped robot easily becomes unstable. From the viewpoint of harmonious real dynamic human locomotion, it is undesirable that the torso slope always be constant. Therefore, it is suggested that $\Theta_h(t)$ should be variable for dynamic walking. In this case, it is possible for the center of gravity to move smoothly, and the impact force should become less. Hence based on the proposed gait planning principle in Archie (presented chapter 5), the proposed slope angle of the constructed upper body could be calculated based on the following equations:

$$\text{New } \Theta_h(t) = (\text{Current } \Theta_h(t)) - A_{\text{slope}} \cos((t - kT_c)/T_d) \quad kT_c < t < kT_c + T_d \quad (8.3)$$

$$\text{New } \Theta_h(t) = (\text{Current } \Theta_h(t)) - A_{\text{slope}} \cos((t - (k+1)T_c)/(T_c - T_d)) \quad kT_c + T_d < t < (k+1)T_c \quad (8.4)$$

where A_{slope} is a constant value and denotes the slope amplitude in sagittal plane. The position of upper body, both the changes of the hip motions $x_h(t)$ and $y_h(t)$ which are the main factors that affect the stability, because of the massive size of the upper body.

References

- Bachar, Y. (2004). *Developing controllers for biped humanoid locomotion*. School of Informatics, University of Edinburg.
- Byagowi, A. (2010). *Control System for a Humanoid Robot*. TU Wien.
- Chen, X., Huang, Q., Yu, Z., & Xu, W. (2011). Biped walking planning using Extended Linear Inverted Pendulum Mode with a continuous moving ZMP. In *IEEE International Conference on Mechatronics and Automation (ICMA)* (pp. 1280–1285), Beijing, China.
- Corrigan, S. (2002). Introduction to the Controller Area Network (CAN), 1–17.
- Dallali, H., Mosadeghzad, M., Medrano-Cerda, G. a., Docquier, N., Kormushev, P., Tsagarakis, N., & Caldwell, D. (2013). Development of a dynamic simulator for a compliant humanoid robot based on a symbolic multibody approach. In *IEEE International Conference on Mechatronics (ICM)* (pp. 598–603). Takamatsu, Japan.
- Davis, R., Burns, A., Bril, R., & Lukkien, J. (2007). Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *The International Journal of Real-Time Systems*, 1–18.
- Davis, R. I., Burns, A., Bril, R. J., & Lukkien, J. J. (2007). Controller Area Network (CAN) schedulability analysis. *The International Journal of Real-Time Systems*, 35(3), 239–272.
- Dezfouli, S., & Kopacek, P. (2011). Mechatronic Design of a Humanoid Robot. In *Proceedings of the RAAD, 20th International Workshop on Robotics in Alpe-Adria-Danube Region* (pp. 1–6). Brno, Czech Republic.
- Dezfouli, S., Kopacek, P., & Mohamadi Daniali, M., (2011). cost oriented humanoid robot archie. *International journal automation austria*, 19(2), 62-70.
- Dezfouli, S., & Mohamadi Daniali, M. (2012). motion controller design for a biped humanoid robot. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE* (pp. 1–7). Chicago, USA.
- Encoder. (2010). Datasheet 360 Step Programmable High Speed Magnetic Rotary Encoder. *austriamicrosystem* (pp. 1-31).
- Goswami, A. (1999). Foot rotation indicator (FRI) point: A new gait planning tool to evaluate postural stability of biped robots. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 47–52). Detroit, USA.
- Guizzo, E., (2010). Huamnaoid Robot Rise, Now Can they Walk?. *IEEE Spectrum Magazine*, 47(10), pp. 20-22.
- Ha, T., & Choi, C.-H. (2007). An effective trajectory generation method for bipedal walking. *Jounral of Robotics and Autonomous Systems*, 55(10), 795–810.

- Harada, K., Kajita, S., Kaneko, K., & Hirukawa, H. (2004). An analytical method on real-time gait planning for a humanoid robot. In *Proceedings of 4th IEEE/RAS International Conference on Humanoid Robots* (pp. 640–655). Santa Monica, California.
- Hartwich, F., Müller, B., Führer, T., & Hugel, R. (2000). CAN network with time triggered communication. In the *7th international conference on CAN Network System* (pp. 1–7). Amsterdam, Netherlands.
- Hirai, K., Hirose, M., & Haikawa, Y. (1998). The development of Honda humanoid robot. In *Proceedings of the IEEE-International Conference on Robotics & Automation* (pp. 1321–1326). Leuven, Belgium.
- Huang, Q., Kajita, S., & Koyachi, N. (1999). A high stability, smooth walking pattern for a biped robot. In *Proceedings of the IEEE- International Conference on Robotics & Automation* (pp. 65–71). Detroit, USA.
- Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., & Tanie, K. (2001). Planning walking patterns for a biped robot. In *Proceedings of the IEEE Transactions on Robotics and Automation*, 17(3), 280–289.
- Jo, H. S., & Jayamani, E. (2009). Design and Trajectory Planning of Bipedal Walking Robot with Minimum Sufficient Actuation System. *World Academy of Science, Engineering and Technology*, (35) 160–166.
- Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1620–1626). Taipei, Taiwan.
- Kim, J., Park, I., Lee, J., & Kim, M. (2005). System design and dynamic walking of humanoid robot KHR-2. In *Proceedings of the IEEE-International Conference on Robotics & Automation* (pp. 1431–1436). Barcelona, Spain.
- Kim, J. Y., Park, I. W., & Oh, J. H. (2007). Walking control algorithm of biped humanoid robot on uneven and inclined floor. *The International Journal of Intelligent & Robotic Systems*, 48(4), 457–484.
- Loffler, K., Gienger, M., & Pfeiffer, F. (2003). Sensors and Control Concept of Walking “Johnnie.” *The International Journal of Robotics Research*, 22(3-4), 229–239.
- Milushev, M. (2010). Design of an Open Software Architecture for Leg Control of a Walking Robot. *Electronics*, 14(2), 110-120.
- Mohamadi Daniali, M. (2013). Walking Control of a Humanoid Robot. Vienna University of Technology.
- Omer, A. M. M., & Ghorbani, R. (2009). Semi-passive dynamic walking for biped walking robot using controllable joint stiffness based on dynamic simulation. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (pp. 1600–1605). Singapore.

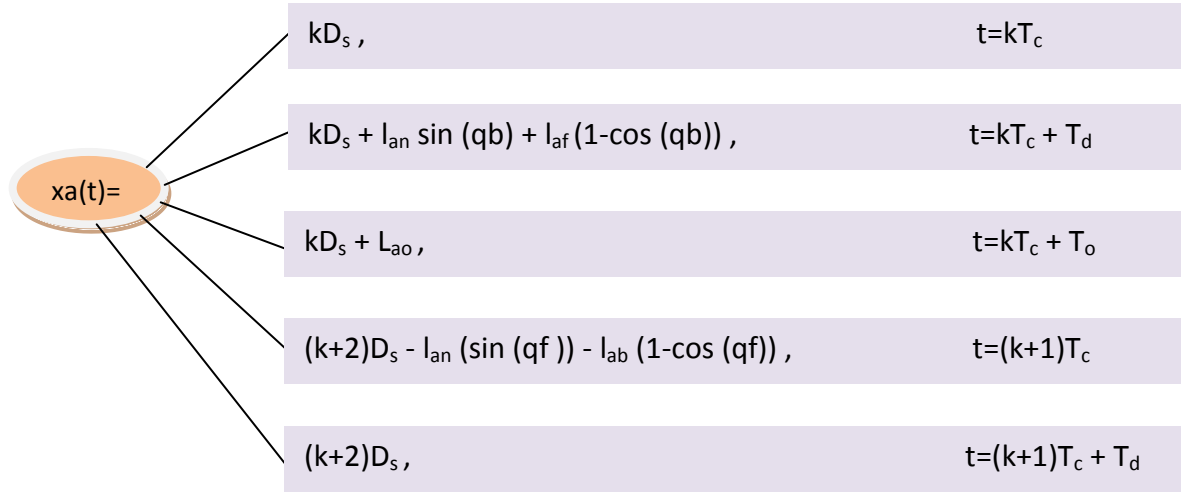
- Perry, J. (1992). *Gait Analysis Normal and Pathological Function*. SLACK Incorporated.
- Shivaraj, D., P, C. P. R., & Lasitha, M. (2012). Design and Development of a Biped Humanoid Robot, In *6th IEEE International Conference on humanoid Robots*, (pp. 67–78).
- Takanishi, A., Ogura, Y., & Itoh, K. (2007). Some Issues in Humanoid Bipedal Humanoid Robot WABIAN-2, *Robotics Research in Springer tracts in Advanced Robotics*, (28) 357-372.
- Tindell, K., Burns, a., & Wellings, a. J. (1995). Calculating controller area network (CAN) message response times. *The International Journal of Control Engineering*, 3(8), 1163–1169.
- Tindell, KW. (1994). Analysing real-time communications: controller area network (CAN). *Real-Time Systems Symposium*.
- Vaughan, C. (2003). Theories of bipedal walking: an odyssey. *The International Journal of Biomechanics*, 36(4), 513–523.
- Vukobratovic, M. (1969). Contribution to the synthesis of biped gait. *IEEE Transactions on Bio-medical Engineering*, (1), 2–7.
- Vukobratovic, M., & Borovac, B. (2001). zero-moment point - proper interpretation and new application. In *Proceedings of the 2nd IEEE/RAS International Conference on Humanoid Robots* (pp. 237-243). Tokyo, Japan.
- Www.diods.com, last visit May 2013.
- Www.elmomc.com, last visit June 2013.
- Www.hondau3-x.net/asimo, last visit October 2013.
- Www.maxonmotor.ch, last visit May 2012.
- Www.news.cnet.com/8301-17938_105-57593396-1/be-afraid-darpa-unveils-terminator-like-atlas-robot/, last visit October 2013.
- Www.pal-robotics.com/robots/reem-b, last visit October 2013.
- Www.qt.digia.com/Product/Developer-Tools, last visit August 2012.
- Www.spectrum.ieee.org/, last visit October 2013.
- Xiao, T., Huang, Q., Li, J., Zhang, W., & Li, K. (2006). Trajectory Calculation and Gait Change On-line for Humanoid Teleoperation. In *Proceedings of the International Conference on Mechatronics and Automation* (pp. 1614–1619). Chengdu, China.
- Yamaguchi, J., Soga, E., Inoue, S., & Takanishi, A. (1999). Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking. *IEEE Robotics and Automation*, (Vol. 1, 368–374).

- Yang, J., Huang, Q., Li, J., Li, C., & Li, K. (2006). Walking pattern generation for humanoid robot considering upper body motion. In *Proceedings of the IEEE/RAS International Conference on Humanoid Robots* (pp. 41–46). Genova, Italy.
- Zuberi, K., & Shin, K. (2000). Design and implementation of efficient message scheduling for controller area network. *IEEE Transactions on Computers*, 49(2), 182–188.

Appendix A

a) Foot and Hip trajectory Calculation

- Foot trajectory equation ($x_a(t)$)

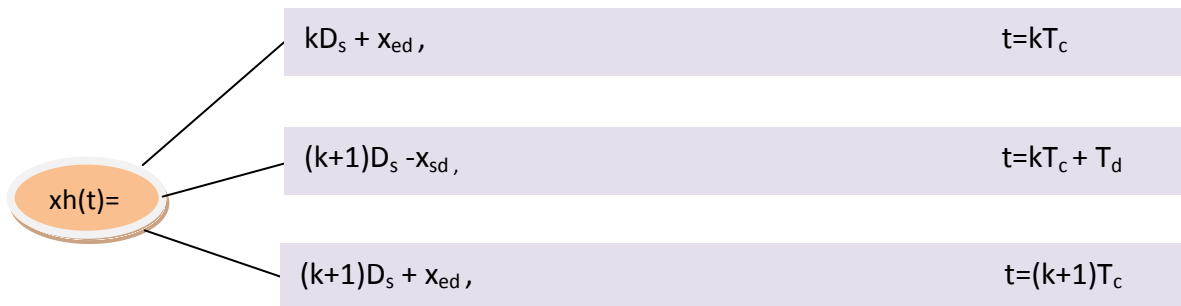


- Hip motion (Upward- $z_h(t)$)

Hip trajectory ($z_h(t)$) constraint at its highest position at the middle of the single-support phase (h_{max}), and at its lowest position (h_{min}) at the middle of the double-support phase during one walking step are:



- Hip motion (Forward- $x_h(t)$)

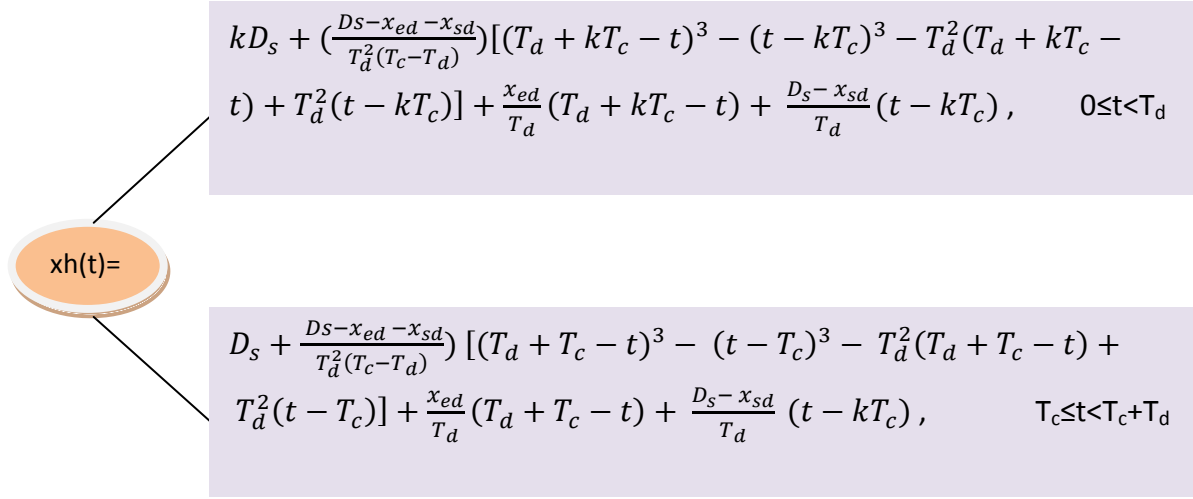


The change of $x_h(t)$ is the main factor that affects the stability of a biped robot walking in a sagittal plane. To obtain a smooth $x_h(t)$, the following derivative constraint must be satisfied:

$$\dot{x}_h(kT_c) = \dot{x}_h(kT_c + T_c)$$

$$\ddot{x}_h(kT_c) = \ddot{x}_h(kT_c + T_c)$$

and the second derivative continuity conditions is as follows:



The diagram shows a central orange oval labeled $x_h(t) =$ with two lines pointing to two separate light purple rectangular boxes containing piecewise equations for $x_h(t)$.

Top box (for $0 \leq t < T_d$):

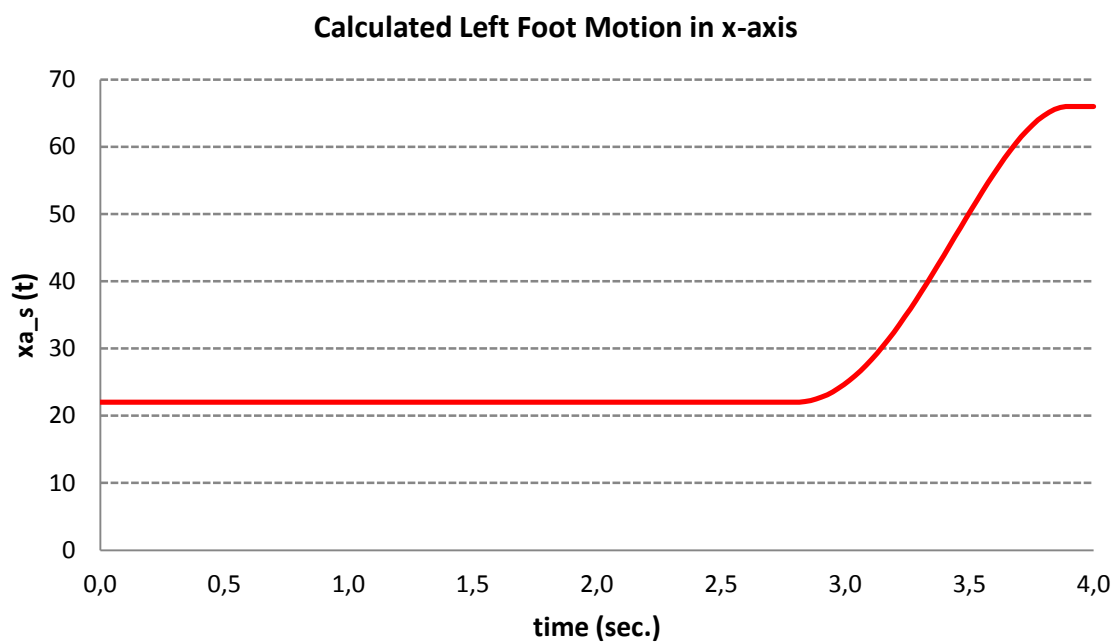
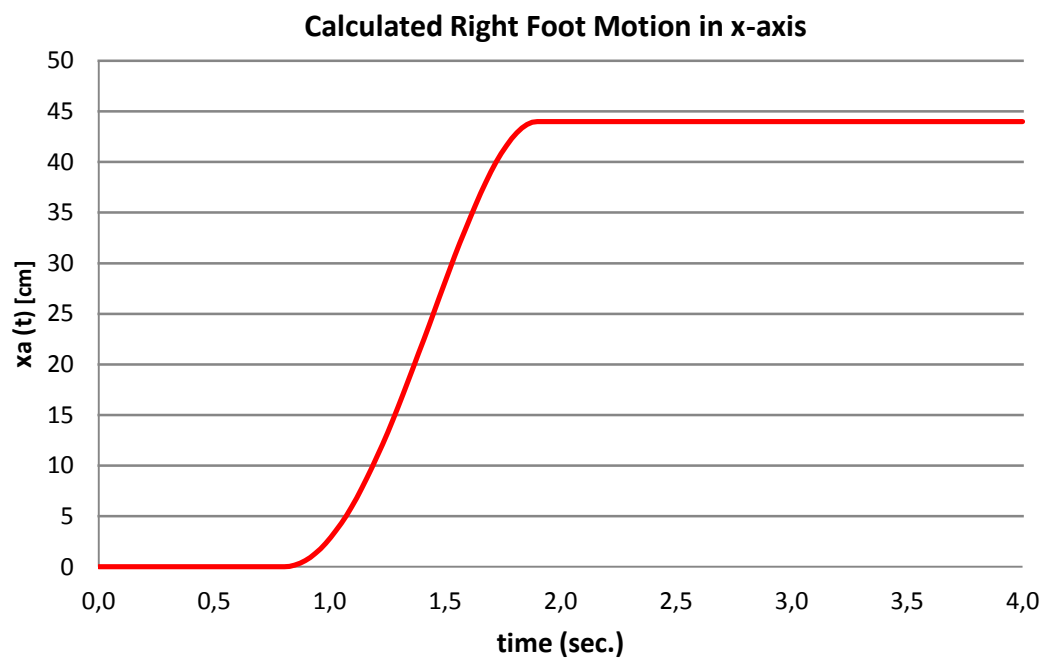
$$kD_s + \left(\frac{D_s - x_{ed} - x_{sd}}{T_d^2(T_c - T_d)} \right) [(T_d + kT_c - t)^3 - (t - kT_c)^3 - T_d^2(T_d + kT_c - t) + T_d^2(t - kT_c)] + \frac{x_{ed}}{T_d} (T_d + kT_c - t) + \frac{D_s - x_{sd}}{T_d} (t - kT_c), \quad 0 \leq t < T_d$$

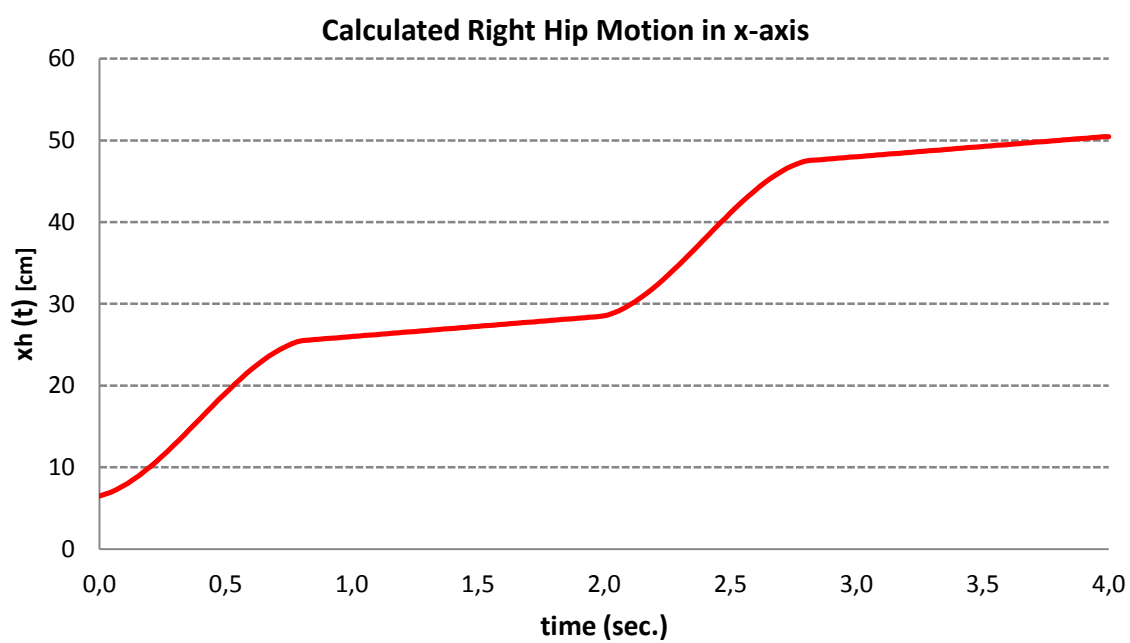
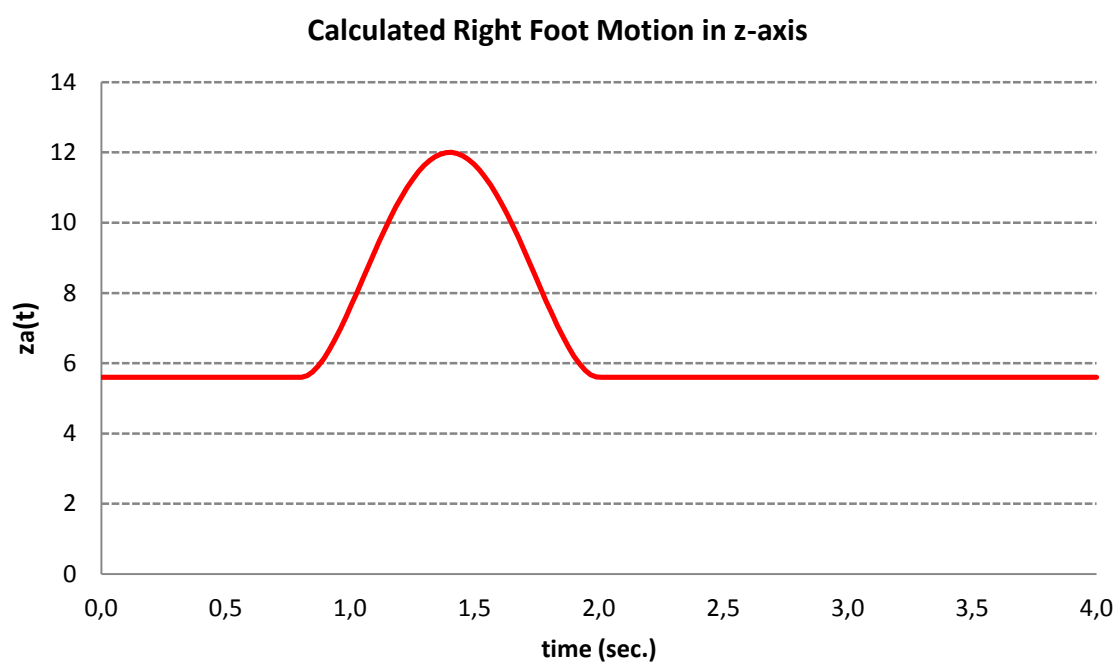
Bottom box (for $T_c \leq t < T_c + T_d$):

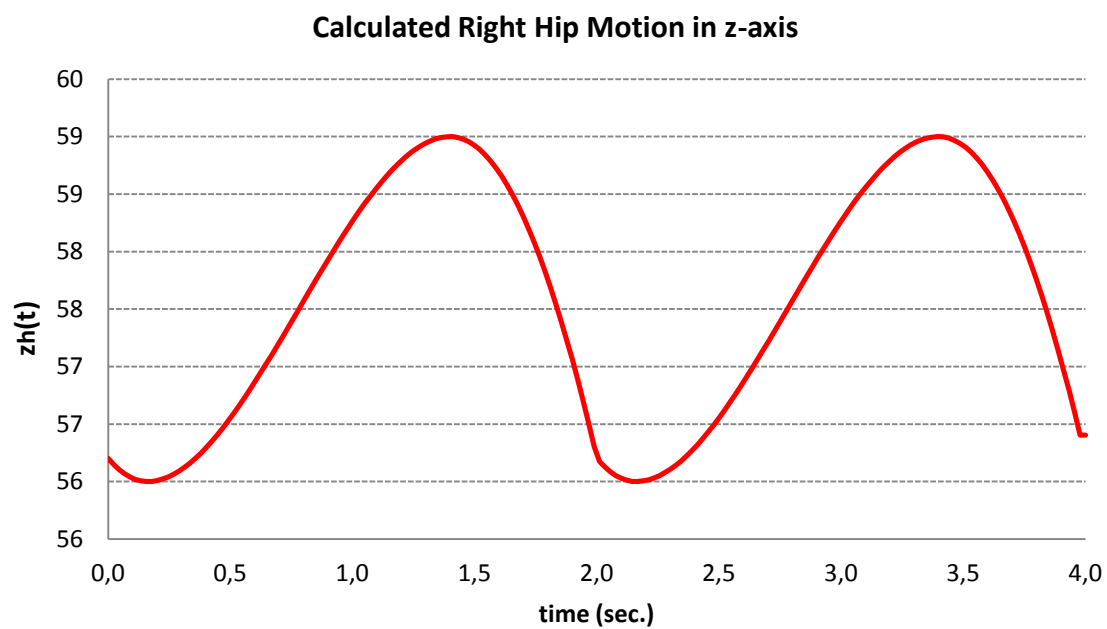
$$D_s + \frac{D_s - x_{ed} - x_{sd}}{T_d^2(T_c - T_d)} [(T_d + T_c - t)^3 - (t - T_c)^3 - T_d^2(T_d + T_c - t) + T_d^2(t - T_c)] + \frac{x_{ed}}{T_d} (T_d + T_c - t) + \frac{D_s - x_{sd}}{T_d} (t - kT_c), \quad T_c \leq t < T_c + T_d$$

By setting different values for x_{sd} and x_{ed} in above condition the various series of smooth $x_h(t)$ are obtained. Up to here the general characteristics of pattern generation and trajectory calculation are presented, but the complete process description, calculation and different robot posture pictures are discussed in Mohamadi Daniali , 2013.

b) Foot Motion in x & z axis:







c) Drive Commands:

The summarize of the driver functional commands are listed as follows:

- **I/O commands description:**

AN[N] read analogue inputs

IB[N] Bit-wise digital input

IF[N] Digital input filter

IP Read all digital inputs

OB[N] Bit-wise digital output

OC[N] Output compare

OL[N] Output logic

OP Set all digital outputs

- **Communication commands:**

PP[N] Define the parameters of CAN or RS-232

- **Motion Commands:**

AC Acceleration

BG Begin motion

IL[N] defining how dedicated inputs behave

MO Motor on/off

PA Absolute position reference for point to point motion

PR Relative position reference for point to point motion

SD Stop deceleration

SF Smooth factor for motion command

- **Protection commands**

HL[N] Over speed and position range limit

LL[N] Low actual feedback limit

PL[N] Peak current

- **Configuration commands**

MP[N] Motion (PV/PVT) parameters

PT position-time

PV position-velocity

QP Position

QV Velocity

UM Unit mode

- **Feedback commands**

PX Main encoder position

XM Specifies the counting range for the main feedback

- **CAN message frame fields**

- **SOF:** the single dominant start of frame (SOF) bit marks the start of a message, and is used to synchronize the nodes on a bus after being idle.
- **Identifier:** the Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.
- **IDE:** a dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.
- **RTR:** the single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way all data being used in a system is uniform.
- **r0:** reserved bit (for possible use by future standard amendment).
- **DLC:** the 4-bit data length code (DLC) contains the number of bytes of data being transmitted.
- **Data:** up to 64 bits of application data may be transmitted.
- **CRC:** the 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.
- **ACK:** every node receiving an accurate message overwrites this recessive bit in the original message with a dominant bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message and the sending node repeats the message after re-arbitration. In this way each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgement bit and the second is a delimiter.
- **EOF:** this end-of-frame (EOF) 7-bit field marks the end of a CAN frame (message) and disables bit-stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.
- **IFS:** this 7-bit inter-frame space (IFS) contains the amount of time required by the controller to move a correctly received frame to its proper position in a message buffer area.

Appendix B

In the file archie.cpp, all available joints of Archie are created and set to initial values (see Listing 7.3). The four parameters are: distance of z-axis (length), distance of x-axis (offset), rotation around x-axis (twist), and rotation around z-axis (angle) (motion controller ver.1.2). The degrees are transformed to radian values because the absolute position movement is done in radians (PA command).

```
1 _joints[CENTERAL_POINT] = new RobotJoint( 0.0, 0.0, toRadians( 0.0 ), toRadians( 0.0 ) );
2 _joints[LEFT_HIP_TRANSVERSAL] = new RobotJoint( 45.0, 0.0, toRadians( 0.0 ), toRadians(0.0 ) );
3 _joints[LEFT_HIP_FRONTAL] = new RobotJoint( 0.0, 50.0, toRadians( 90.0 ), toRadians( 0.0 ) );
4 _joints[LEFT_HIP_LATERAL] = new RobotJoint( 70.0, 0.0, toRadians( 0.0 ), toRadians( 0.0 ) );
5 _joints[LEFT_KNEE_LATERAL] = new RobotJoint( 310.0, 0.0, toRadians( 0.0 ), toRadians(0.0 ) );
6 _joints[LEFT_ANKLE_LATERAL] = new RobotJoint( 260.0, 0.0, toRadians( 0.0 ), toRadians(0.0 ) );
7 _joints[LEFT_ANKLE_FRONTAL] = new RobotJoint( 55.0, 0.0, toRadians( 90.0 ), toRadians(0.0 ) );
8 _joints[LEFT_FOOT_LATERAL] = new RobotJoint( 0.0, 84.0, toRadians( 90.0 ), toRadians( 0.0 ) );
9 _joints[LEFT_TOE] = new RobotJoint( 30.0, 0.0, toRadians( 0.0 ), toRadians( 0.0 ) );
10 _joints[RIGHT_HIP_TRANSVERSAL] = new RobotJoint( 45.0,0.0, toRadians( 0.0 ), toRadians( 0.0 ) );
11 _joints[RIGHT_HIP_FRONTAL] = new RobotJoint( 0.0, 50.0, toRadians( 90.0 ), toRadians( 0.0 ) );
12 _joints[RIGHT_HIP_LATERAL] = new RobotJoint( 70.0,0.0, toRadians( 0.0 ), toRadians( 0.0 ) );
13 _joints[RIGHT_KNEE_LATERAL] = new RobotJoint( 310.0,0.0, toRadians( 0.0 ), toRadians(0.0 ) );
14 _joints[RIGHT_ANKLE_LATERAL] = new RobotJoint( 260.0,0.0, toRadians( 0.0 ), toRadians(0.0 ) );
15 _joints[RIGHT_ANKLE_FRONTAL] = new RobotJoint( 55.0,0.0, toRadians( 90.0 ), toRadians(0.0 ) );
16 _joints[RIGHT_FOOT_LATERAL] = new RobotJoint( 0.0, 84.0, toRadians( 90.0 ), toRadians(0.0 ) );
17 _joints[RIGHT_TOE] = new RobotJoint( 30.0,0.0, toRadians( 0.0 ), toRadians( 0.0 ) );
```

Listing 7.3 Archie creation of joints

additional joint and servo information is initialized as well as GUI parameters (see Listing 7.4). The third parameter of the method `setServoInfo` represents the joint id. If it is set to -1 the joint is deactivated. Listing 4 shows only joint, servo, and GUI information for Archie's left side. The right side information is defined equally and we omit the source code due to space limitations.

```
1 _joints[_typeTable[LEFT_HIP_TRANSVERSAL]]> setServoInfo( "I7565 Canbus SimplIq Motor",
"LeftHipTransversal", /*14*/1, 0, 0, 0, 0, true, DESIRED_SPEED );
2 _joints[_typeTable[LEFT_HIP_TRANSVERSAL]]> setUIInfo( 338, 242, 68, 22, "ServoSpinner" );
3 _joints[_typeTable[LEFT_HIP_FRONTAL]]> setServoInfo( "I7565 Canbus SimplIq Motor", "LeftHipFrontal",
/*13*/1,0, 20, 20,161, true, DESIRED_SPEED );
4 _joints[_typeTable[LEFT_HIP_FRONTAL]]> setUIInfo( 338, 220, 68, 22, "ServoSpinner" );
5 _joints[_typeTable[LEFT_HIP_LATERAL]]> setServoInfo( "I7565 Canbus SimplIq Motor", "LeftHipLateral", 12 , 0,
20, 75, 161, true, DESIRED_SPEED + 100000);
6 _joints[_typeTable[LEFT_HIP_LATERAL]]> setUIInfo( 406, 220, 68, 22, "ServoSpinner" );
7 _joints[_typeTable[LEFT_KNEE_LATERAL]]> setServoInfo( "I7565 Canbus SimplIq Motor", "LeftKneeLateral",
11, 0, 37, 84, 161, false, DESIRED_SPEED 200000);
8 _joints[_typeTable[LEFT_KNEE_LATERAL]]>setUIInfo( 406, 290, 68, 22, "ServoSpinner" );
9 _joints[_typeTable[LEFT_ANKLE_LATERAL]]>setServoInfo( "I7565 Canbus SimplIq Motor", "LeftAnkleLateral",
10, 0, 45, 30,161, false, DESIRED_SPEED 400000); 10 _joints[_typeTable[LEFT_ANKLE_LATERAL]]> setUIInfo(
406, 356, 68, 22, "ServoSpinner" );
11 _joints[_typeTable[LEFT_ANKLE_FRONTAL]]>setServoInfo( "I7565 Canbus SimplIq Motor", "LeftAnkleFrontal",
9, 0, 27, 27, 500, true, DESIRED_SPEED );
12 _joints[_typeTable[LEFT_ANKLE_FRONTAL]]> setServoInfo( "I7565 Canbus SimplIq Motor", "LeftAnkleFrontal",
/*9*/1, 0, 0, 0, 0, true, DESIRED_SPEED );
13 _joints[_typeTable[LEFT_ANKLE_FRONTAL]]>setUIInfo( 396, 378, 68, 22, "ServoSpinner" );
14 _joints[_typeTable[LEFT_FOOT_LATERAL]]>setServoInfo( "I7565 Canbus SimplIq Motor", "LeftToeLateral",
/*8*/1,0, 0, 0, 0, true, DESIRED_SPEED );
15 _joints[_typeTable[LEFT_FOOT_LATERAL]]>setUIInfo( 464, 378, 68, 22, "ServoSpinner" );
```

Listing 7.4. SetservoInfo method

In the following inverse kinematic calculation for the joints of Archie in new motion controller software (ver.2.1) is presented.

```
//-----Inverse Kinematic calculation-----
//-----swing leg-----
//----new calculation (6DOF)----
double matrixRR0Elements[] = {-sin(theta_), 0, cos(theta_),
                                0, -1, 0,
                                cos(theta_), 0, sin(theta_)};

double matrixRR0TElements[] = {-sin(theta_), 0, cos(theta_),
                                0, -1, 0,
                                cos(theta_), 0, sin(theta_)};

double matrixRR6Elements[] = {-sin(alpha_), -cos(alpha_), 0,
                                -cos(alpha_), sin(alpha_), 0,
                                0, 0, -1};

double matrixpR6Elements[] = {xh_,yh_,zh_};
double matrixpR0Elements[] = {xa_,ya_,za_};
double matrixpRcElements[] = {a6_,0,-d6_};

RR0_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRR0Elements));
RR0T_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRR0TElements));
RR05_ = QSharedPointer<Matrix>(new Matrix(3,1));

RR6_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRR6Elements));
RR62_ = QSharedPointer<Matrix>(new Matrix(3,1));
RR63_ = QSharedPointer<Matrix>(new Matrix(3,1));

pR6_ = QSharedPointer<Matrix>(new Matrix(3,1,matrixpR6Elements));
pR62_ = QSharedPointer<Matrix>(new Matrix(3,1));

pR0_ = QSharedPointer<Matrix>(new Matrix(3,1,matrixpR0Elements));
pRc_ = QSharedPointer<Matrix>(new Matrix(3,1,matrixpRcElements));
pR5_ = QSharedPointer<Matrix>(new Matrix(3,1));

//pR5 calculation;

(*pR6_.data())-=(*pR0_.data());
(*pR62_.data())=(*pR6_.data());
(*RR6_.data())*=(*pRc_.data());
(*RR62_.data())=(*RR6_.data());

(*RR62_.data())+=(*pR62_.data());
(*RR63_.data())=(*RR62_.data());
(*RR0T_.data())*=(*RR63_.data());
```

```

(*RR05_.data())=(*RR0T_.data());

//pR5
(*pR5_.data())=(*RR05_.data());

//pR5(1,2,3)
pR5_1 = (*RR05_.data())[0][0];
pR5_2 = (*RR05_.data())[1][0];
pR5_3 = (*RR05_.data())[2][0];

th1_ = atan2 (pR5_2,pR5_1) + asin (d3_/sqrt(pow(pR5_1,2) + pow(pR5_2,2))) ;

c3_ = ( pow(pR5_3,2) + pow (pR5_1 - d3_ * sin(th1_) - a1_ *cos(th1_),2) + pow (pR5_2 + d3_ *
cos(th1_) - a1_ * sin(th1_),2)
- pow(a2_,2) - pow (a3_,2))/ (2*a2_*a3_);

th3_ = atan2 (sqrt(1-pow(c3_,2)),c3_);
th2_ = atan2 (-pR5_3,sqrt(pow(pR5_1 - d3_ * sin (th1_) - a1_ * cos (th1_),2) + pow (pR5_2 + d3_ * cos
(th1_) - a1_ * sin (th1_),2)))
- atan2 (a3_ * sin (th3_), a3_ * cos (th3_) + a2_);

RR6_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRR6Elements));
RR0_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRR0Elements));
RR_ = QSharedPointer<Matrix>(new Matrix(3,3));

(*RR0_.data())*=(*RR6_.data());
(*RR_.data())=(*RR0_.data());

double matrixRRP0TElements[] = {cos (th1_) * cos (th2_+ th3_), sin (th1_) * cos (th2_+th3_), -sin
(th2_ + th3_),
-cos( th1_) * sin (th2_+th3_), -sin (th1_) * sin (th2_ + th3_), -cos (th2_+ th3_),
-sin (th1_), cos (th1_), 0 };

RRP_ = QSharedPointer<Matrix>(new Matrix(3,3));
RRPOT_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRRP0TElements));

(*RRPOT_.data())*=(*RR_.data());

(*RRP_.data())=(*RRPOT_.data());

RRP_11 = (*RRP_.data())[0][0];
RRP_13 = (*RRP_.data())[0][2];
RRP_21 = (*RRP_.data())[1][0];
RRP_23 = (*RRP_.data())[1][2];
RRP_31 = (*RRP_.data())[2][0];
RRP_32 = (*RRP_.data())[2][1];

```

```

RRP_33 = (*RRP_.data())[2][2];

if ((RRP_13 == 0) &&
    (RRP_23 == 0)) {

th5_ = 0 ;
th6_ = 0 ;
th4_ = atan2(RRP_21,RRP_11) - th6_ ;
    }
else
    {
th5_ = atan2 (sqrt(1-(pow(-RRP_33,2))),RRP_33);
th4_ = atan2 (-RRP_23,-RRP_13);
th6_ = atan2 (-RRP_32,RRP_31);
    }

tRight_Ankle_frontal_ = th1_ * 160 * 180/M_PI;
tRight_Ankle_lateral_ = th2_ * 160 * 180/M_PI;
tRight_Knee_ = th3_ * 160 * 180/M_PI;
tRight_Hip_lateral_ = th4_ * 160 * 180/M_PI;
tRight_Hip_frontal_ = (th5_ - ((M_PI)/2)) * 160 * 180/M_PI;
tRight_Hip_transversal_ = th6_ * 2360 * 180/M_PI;

tRAfList_.append(tRight_Ankle_frontal_);
tRAList_.append(tRight_Ankle_lateral_);
tRKList_.append(tRight_Knee_);
tRHList_.append(tRight_Hip_lateral_);
tRHfList_.append(tRight_Hip_frontal_);
tRHtList_.append(tRight_Hip_transversal_);

//-----stance leg-----LEFT leg -----

// ----new calculation (6DOF)
double matrixRL0Elements[] = {-sin(theta_s), 0, cos(theta_s),
                                0, -1, 0,
                                cos(theta_s), 0, sin(theta_s)};

double matrixRL0TElements[] = {-sin(theta_s), 0, cos(theta_s),
                                0, -1, 0,
                                cos(theta_s), 0, sin(theta_s)};

double matrixRL6Elements[] = {-sin(alpha_), -cos(alpha_), 0,
                                -cos(alpha_), sin(alpha_), 0,
                                0, 0, -1};

double matrixpL6Elements[] = {xh_,yh_,zh_};

```

```

double matrixpL0Elements[] = {xa_s,ya_s,za_s};
double matrixpLcElements[] = {-a6_,0,-d6_};

RL0_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRL0Elements));
RL0T_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRL0TElements));
RL05_ = QSharedPointer<Matrix>(new Matrix(3,1));

RL6_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRL6Elements));
RL62_ = QSharedPointer<Matrix>(new Matrix(3,1));
RL63_ = QSharedPointer<Matrix>(new Matrix(3,1));

pL6_ = QSharedPointer<Matrix>(new Matrix(3,1,matrixpL6Elements));
pL62_ = QSharedPointer<Matrix>(new Matrix(3,1));

pL0_ = QSharedPointer<Matrix>(new Matrix(3,1,matrixpL0Elements));
pLc_ = QSharedPointer<Matrix>(new Matrix(3,1,matrixpLcElements));
pL5_ = QSharedPointer<Matrix>(new Matrix(3,1));

//pL5 calculation;

(*pL6_.data())-=(*pL0_.data());
(*pL62_.data())=(*pL6_.data());
(*RL6_.data())*=(*pLc_.data());
(*RL62_.data())=(*RL6_.data());

(*RL62_.data())+=(*pL62_.data());
(*RL63_.data())=(*RL62_.data());
(*RL0T_.data())*=(*RL63_.data());
(*RL05_.data())=(*RL0T_.data());

//pL5
(*pL5_.data())=(*RL05_.data());

//pL5(1,2,3)
pL5_1 = (*RL05_.data())[0][0];
pL5_2 = (*RL05_.data())[1][0];
pL5_3 = (*RL05_.data())[2][0];

th1_ = atan2 (pL5_2,pL5_1) - asin (d3_/sqrt(pow(pL5_1,2)+ pow(pL5_2,2))) ;
c3_ = ( pow(pL5_3,2) + pow (pL5_1 + d3_ * sin(th1_) - a1_ *cos(th1_),2) + pow (pL5_2 - d3_ *
cos(th1_)- a1_ * sin(th1_),2)
- pow(a2_,2) - pow (a3_,2))/ (2*a2_*a3_);
th3_ = atan2 (-sqrt(1-pow(c3_,2)),c3_);
th2_ = atan2 (pL5_3,sqrt(pow(pL5_1 + d3_ * sin (th1_)- a1_ * cos (th1_),2) + pow (pL5_2 - d3_ *
cos (th1_) - a1_ * sin (th1_),2)))
- atan2 (a3_ * sin (th3_), a3_ * cos (th3_) + a2_ ) ;

```

```

RL6_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRL6Elements));
RL0_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRL0Elements));
RL_ = QSharedPointer<Matrix>(new Matrix(3,3));

(*RL0_.data())*=(*RL6_.data());
(*RL_.data())=(*RL0_.data());

double matrixRLPOTElements[] = {cos (th1_) * cos (th2_+ th3_),    sin (th1_) * cos (th2_+th3_),
sin (th2_ + th3_),  -cos( th1_) * sin (th2_+th3_),  -sin (th1_) * sin (th2_ + th3_),  cos (th2_+ th3_), sin
(th1_), -cos (th1_), 0 };

RLP_ = QSharedPointer<Matrix>(new Matrix(3,3));
RLPOT_ = QSharedPointer<Matrix>(new Matrix(3,3,matrixRLPOTElements));

(*RLPOT_.data())*=(*RL_.data());
(*RLP_.data())=(*RLPOT_.data());

RLP_11 = (*RLP_.data())[0][0];
RLP_13 = (*RLP_.data())[0][2];
RLP_21 = (*RLP_.data())[1][0];
RLP_23 = (*RLP_.data())[1][2];
RLP_31 = (*RLP_.data())[2][0];
RLP_32 = (*RLP_.data())[2][1];
RLP_33 = (*RLP_.data())[2][2];

if  ((RLP_13 == 0) &&
(RLP_23 == 0 )) {

    th5_ = 0 ;
    th6_ = 0 ;
    th4_ = atan2(RLP_21,RLP_11)+ th6_ ;
}
else
{
    th5_ = atan2 (sqrt(1-(pow(-RLP_33,2))),-RLP_33);
    th4_ = atan2 (-RLP_23,-RLP_13);
    th6_ = atan2 (RLP_32,-RLP_31);
}

tLeft_Ankle_frontal_ = th1_ * 160 * 180/M_PI;
tLeft_Ankle_lateral_ = th2_ * 160 * 180/M_PI;
tLeft_Knee_ = th3_ * 160 * 180/M_PI;
tLeft_Hip_lateral_ = th4_ * 160 * 180/M_PI;
tLeft_Hip_frontal_ = (th5_ - ((M_PI)/2)) * 160 * 180/M_PI;
tLeft_Hip_transversal_ = th6_ * 2360* 180/M_PI;

```

```

        tLAfList_.append(tLeft_Ankle_frontal_);
        tLAlList_.append(tLeft_Ankle_lateral_);
        tLKList_.append(tLeft_Knee_);
        tLHlList_.append(tLeft_Hip_lateral_);
        tLHfList_.append(tLeft_Hip_frontal_);
        tLHtList_.append(tLeft_Hip_transversal_);
    }
}

```

Listing 7.5 Inverse Kinematic calculation (source code)

The method shown in Listing 7.6 (excerpt) from robot.cpp invokes the defined motion path for a robot. It specifies an infinite loop (line 1) that the defined motion will be executed until the stop button is pressed. The moveRobot method call (line 18) passes the position of the joint to be moved to. The joint position is retrieved at line 16. The implementation of the method moveRobot is then specialized for the different robot types in their corresponding classes.

```

1 while( !_thrun )
2 {
3     _mutex.unlock();
4     _mutex.lock();
5     if( !_playing )
6     {
7         _moves[_ktype].increment();
8         setJointAngles( _moves[_ktype].angles(), _modAngles[_ktype] );
9         _mutex.unlock();
10        _mutex.lock();
11        _playing = _moves[_ktype].isPlaying();
12        if( !_sending )
13        {
14            _sending = _playing;
15            _mutex.unlock();
16            _pos.setPosition( _joints );
17            _mutex.lock();
18            moveRobot( _pos, motionDuration() );
19            if( !_sending )
20            {
21                time = motionDuration();
22            }
23            else
24            {
25                time = UPDATE_INTERVAL;
26            }
27        }
28        else
29        {
30            updateGraphics();
31        }
32        _mutex.unlock();
33    }
34    _mutex.unlock();
35    msleep( time );
36    _mutex.lock();
37 }

```

Listing 7.6: Excerpt of method void Robot::run(void)

Parameters used in new motion controller program (ver.2.1-source code)

- `a1_ = 8;` //distance between frontal-lateral joint in ankle
- `a2_ = 26;` //leg length
- `a3_ = 30;` //thigh length
- `a6_ = 5.6;` //distance between transversal joint in hip and CoG
- `d3_ = 7.4;` //distance between lateral-frontal joint in hip
- `d6_ = 4;` //distance between frontal-transversal joint in hip
- `double laf = 12.5;` //distance between ankle frontal to toe
- `double lan = 5.6;` //distance between ankle lateral to ground
- `double lab = 6.5;` //distance between ankle frontal to heel

`theta, theta_s`

- `double qb = 0 ;` //ankle angle when $t=T_d$ (pre-swing angle)
- `double qf = 0 ;` //ankle angle when $t=T_c$ (post-swing angle)

`xa, za, xa_s, za_s`

- `double Ds = 25;` //Step Length
- `double lao = Ds;` //x position of maximum height of ankle
- `double Hao = 12;` //(y direction)ankle maximum height to pass obstacle

`ya, ya_s, yh`

- `ws_ = 17;` //Step width
- `wm_ = 2 ;` //Max. side movement of CoG from ankle position
- `wp_ = 5 ;` //Max. side movement of ankle

Hip Parameters

`xh`

- `double xed = 6.5 ;` //forward position of CG from back foot at $T=K*T_c$
- `double xsd = -3.5 ;` //forward position of CG from front foot at $T=K*T_c+T_d$

`zh`

- `double hmin = 56;` //min height of CG
- `double hmax = 59;` //max height of CG (max=65.6)
- `h0_ = hmin + 0.2;`

Zero Setting

- `alpha_ = 0;` //angle rotation difference between main coordinate and CoG in Z direction
- `xa_ = 0.0;`
- `xh_ = 0.0;`
- `ya_ = 0.0;`
- `za_ = 0.0;`
- `zh_ = 0.0;`
- `x_ = 0.0;`
- `y_ = 0.0;`
- `x_s = 0.0;`

- $y_s = 0.0;$
- $xa_s = 0.0;$
- $ya_s = 0.0;$
- $za_s = 0.0;$

Time Inputs

Basic time parameters

- $Tc_ = 2$; //step time if time : $t=kTc+Tc+Td$ then $xa = (k+2)*Ds$
- $Td_ = 0.4 * Tc_;$ //Td: double support time interval 40%Tc

xa,za

- $double To = (Td_+Tc_)/2;$

//time for maximum height of ankle in z direction (time of Lao,Hao)

theta, theta_s

- $T1_ = 0.2 * Td_;$ //time of leaving stance foot from ground
- $T4_ = 0.4;$
- $T3_ = Tc_ + T1_ + T4_;$

// time of landing swing leg completely in ground $T1+T4 < Td$

theta, theta_s, xa, xa_s, za, za_s

- $T2_ = Tc_ - 0.1;$ //only for curve fitting

zh

- $T7_ = T1_;$ //time of hmin
- $T8_ = (Tc_+Td_)/2;$ //time of hmax

ya,ya_s

- $T9_ = (Tc_+Td_)/2;$ //time of wp (wp=max side movement)

- **Joint angles (QP)**

All joints angle value for stable forward walking test (10m) based on the new designed motion creating system (chapter 5) and new motion controller software (chapter 6):

Right Ankle frontal joint angle value (2,QP[256])			
2,QP[1]=2596.06	2,QP[21]=1184.84	2,QP[41]=-252.116	2,QP[61]=-1362.96
2,QP[2]=2528.85	2,QP[22]=1112.28	2,QP[42]=-322.162	2,QP[62]=-1399.49
2,QP[3]=2461.15	2,QP[23]=1039.69	2,QP[43]=-391.953	2,QP[63]=-1435.35
2,QP[4]=2392.98	2,QP[24]=967.099	2,QP[44]=-461.479	2,QP[64]=-1470.53
2,QP[5]=2324.37	2,QP[25]=-894.514	2,QP[45]=-530.734	2,QP[65]=-1504.99
2,QP[6]=2255.32	2,QP[26]=-821.958	2,QP[46]=-599.711	2,QP[66]=-1538.68
2,QP[7]=2185.87	2,QP[27]=749.448	2,QP[47]=-668.403	2,QP[67]=-1571.56
2,QP[8]=2116.03	2,QP[28]=677.002	2,QP[48]=-736.804	2,QP[68]=-1603.6
2,QP[9]=2045.83	2,QP[29]=604.637	2,QP[49]=-804.91	2,QP[69]=-1634.74
2,QP[10]=1975.29	2,QP[30]=-532.368	2,QP[50]=-872.715	2,QP[70]=-1664.93
2,QP[11]=1904.42	2,QP[31]=-460.212	2,QP[51]=-940.215	2,QP[71]=-1694.12
2,QP[12]=1833.28	2,QP[32]=-388.183	2,QP[52]=-1007.41	2,QP[72]=-1722.26
2,QP[13]=1761.89	2,QP[33]=-316.297	2,QP[53]=-1049.27	2,QP[73]=-1749.3
2,QP[14]=1690.29	2,QP[34]=-244.566	2,QP[54]=-1090.52	2,QP[74]=-1775.18
2,QP[15]=1618.49	2,QP[35]=-173.006	2,QP[55]=-1131.18	2,QP[75]=-1799.84
2,QP[16]=1546.52	2,QP[36]=-101.628	2,QP[56]=-1171.26	2,QP[76]=-1823.24
2,QP[17]=1474.4	2,QP[37]=-30.4446	2,QP[57]=-1210.76	2,QP[77]=-1845.31
2,QP[18]=1402.15	2,QP[38]=-40.5322	2,QP[58]=-1249.69	2,QP[78]=-1866.02
2,QP[19]=1329.8	2,QP[39]=-111.291	2,QP[59]=-1288.05	2,QP[79]=-1885.31
2,QP[20]=1257.35	2,QP[40]=-181.823	2,QP[60]=-1325.81	2,QP[80]=-1903.13

2,QP[81]=-1919.43	2,QP[101]=-1896.91	2,QP[121]=-1326.17	2,QP[141]=-149.016
2,QP[82]=-1934.19	2,QP[102]=-1878.86	2,QP[122]=-1289.07	2,QP[142]=-77.1133
2,QP[83]=-1947.35	2,QP[103]=-1859.42	2,QP[123]=-1251.37	2,QP[143]=-5.14283
2,QP[84]=-1958.88	2,QP[104]=-1838.63	2,QP[124]=-1213.07	2,QP[144]=66.8762
2,QP[85]=-1968.76	2,QP[105]=-1816.54	2,QP[125]=-1174.17	2,QP[145]=138.925
2,QP[86]=-1976.96	2,QP[106]=-1793.19	2,QP[126]=-1134.66	2,QP[146]=210.983
2,QP[87]=-1983.45	2,QP[107]=-1768.65	2,QP[127]=-1094.52	2,QP[147]=283.034
2,QP[88]=-1988.22	2,QP[108]=-1742.96	2,QP[128]=-1053.73	2,QP[148]=355.059
2,QP[89]=-1991.27	2,QP[109]=-1716.18	2,QP[129]=-998.529	2,QP[149]=427.04
2,QP[90]=-1992.57	2,QP[110]=-1688.35	2,QP[130]=-929.174	2,QP[150]=498.96
2,QP[91]=-1992.14	2,QP[111]=-1659.52	2,QP[131]=-859.486	2,QP[151]=570.802
2,QP[92]=-1990.02	2,QP[112]=-1629.75	2,QP[132]=-789.485	2,QP[152]=642.549
2,QP[93]=-1986.2	2,QP[113]=-1599.08	2,QP[133]=-719.191	2,QP[153]=714.186
2,QP[94]=-1980.69	2,QP[114]=-1567.56	2,QP[134]=-648.627	2,QP[154]=785.696
2,QP[95]=-1973.52	2,QP[115]=-1535.22	2,QP[135]=-577.815	2,QP[155]=857.066
2,QP[96]=-1964.7	2,QP[116]=-1502.12	2,QP[136]=-506.777	2,QP[156]=928.281
2,QP[97]=-1954.26	2,QP[117]=-1468.28	2,QP[137]=-435.536	2,QP[157]=999.327
2,QP[98]=-1942.23	2,QP[118]=-1433.73	2,QP[138]=-364.115	2,QP[158]=1070.19
2,QP[99]=-1928.63	2,QP[119]=-1398.52	2,QP[139]=-292.537	2,QP[159]=1140.86
2,QP[100]=-1913.51	2,QP[120]=-1362.66	2,QP[140]=-220.83	2,QP[160]=1211.32

2,QP[161]=1281.56	2,QP[181]=2526.98	2,QP[201]=2481.88	2,QP[221]=2467.42
2,QP[162]=1351.57	2,QP[182]=2524.38	2,QP[202]=2480.23	2,QP[222]=2468.05
2,QP[163]=1421.34	2,QP[183]=2521.79	2,QP[203]=2478.65	2,QP[223]=2468.85
2,QP[164]=1490.86	2,QP[184]=2519.23	2,QP[204]=2477.16	2,QP[224]=2469.82
2,QP[165]=1560.12	2,QP[185]=2516.68	2,QP[205]=2475.75	2,QP[225]=2470.96
2,QP[166]=1629.11	2,QP[186]=2514.17	2,QP[206]=2474.43	2,QP[226]=2472.27
2,QP[167]=1697.82	2,QP[187]=2511.69	2,QP[207]=2473.21	2,QP[227]=2473.75
2,QP[168]=1766.25	2,QP[188]=2509.24	2,QP[208]=2472.07	2,QP[228]=2475.4
2,QP[169]=1834.39	2,QP[189]=2506.83	2,QP[209]=2471.03	2,QP[229]=2477.23
2,QP[170]=1902.22	2,QP[190]=2504.45	2,QP[210]=2470.09	2,QP[230]=2479.22
2,QP[171]=1969.76	2,QP[191]=2502.12	2,QP[211]=2469.26	2,QP[231]=2481.39
2,QP[172]=2036.98	2,QP[192]=2499.84	2,QP[212]=2468.53	2,QP[232]=2483.74
2,QP[173]=2103.89	2,QP[193]=2497.61	2,QP[213]=2467.91	2,QP[233]=2486.26
2,QP[174]=2170.49	2,QP[194]=2495.43	2,QP[214]=2467.4	2,QP[234]=2488.95
2,QP[175]=2236.76	2,QP[195]=2493.3	2,QP[215]=2467	2,QP[235]=2491.83
2,QP[176]=2302.7	2,QP[196]=2491.24	2,QP[216]=2466.72	2,QP[236]=2494.88
2,QP[177]=2368.32	2,QP[197]=2489.23	2,QP[217]=2466.57	2,QP[237]=2498.11
2,QP[178]=2433.61	2,QP[198]=2487.29	2,QP[218]=2466.54	2,QP[238]=2501.52
2,QP[179]=2498.57	2,QP[199]=2485.42	2,QP[219]=2466.66	2,QP[239]=2505.12
2,QP[180]=2529.6	2,QP[200]=2483.61	2,QP[220]=2466.96	2,QP[240]=2508.9

2,QP[241]=2512.87	2,QP[249]=2551.51
2,QP[242]=2517.02	2,QP[250]=2557.24
2,QP[243]=2521.36	2,QP[251]=2563.18
2,QP[244]=2525.9	2,QP[252]=2569.32
2,QP[245]=2530.63	2,QP[253]=2575.68
2,QP[246]=2535.55	2,QP[254]=2582.25
2,QP[247]=2540.67	2,QP[255]=2589.04
2,QP[248]=2545.99	2,QP[256]=2596.06

Right Ankle lateral joint angle value (3,QP[256])			
3,QP[1]=7643.63	3,QP[21]=8735.03	3,QP[41]=8483.9	3,QP[61]=7611.8
3,QP[2]=7697.08	3,QP[22]=8769.59	3,QP[42]=8414.27	3,QP[62]=7698.27
3,QP[3]=7752.47	3,QP[23]=8800.14	3,QP[43]=8339.4	3,QP[63]=7789.41
3,QP[4]=7809.52	3,QP[24]=8826.44	3,QP[44]=8259.43	3,QP[64]=7883.07
3,QP[5]=7867.96	3,QP[25]=8848.28	3,QP[45]=8174.52	3,QP[65]=7977.34
3,QP[6]=7927.48	3,QP[26]=8865.46	3,QP[46]=8084.86	3,QP[66]=8070.51
3,QP[7]=7987.75	3,QP[27]=8877.79	3,QP[47]=7990.68	3,QP[67]=8161.09
3,QP[8]=8048.45	3,QP[28]=8885.11	3,QP[48]=7892.24	3,QP[68]=8247.76
3,QP[9]=8109.24	3,QP[29]=8887.28	3,QP[49]=7789.84	3,QP[69]=8329.34
3,QP[10]=8169.75	3,QP[30]=8884.17	3,QP[50]=7683.82	3,QP[70]=8404.78
3,QP[11]=8229.62	3,QP[31]=8875.68	3,QP[51]=7574.58	3,QP[71]=8473.13
3,QP[12]=8288.62	3,QP[32]=8861.71	3,QP[52]=7462.57	3,QP[72]=8533.56
3,QP[13]=8346.59	3,QP[33]=8842.2	3,QP[53]=7395.9	3,QP[73]=8585.3
3,QP[14]=8403.23	3,QP[34]=8817.09	3,QP[54]=7353.63	3,QP[74]=8627.65
3,QP[15]=8458.19	3,QP[35]=8786.34	3,QP[55]=7335.09	3,QP[75]=8659.99
3,QP[16]=8511.15	3,QP[36]=8749.95	3,QP[56]=7338.91	3,QP[76]=8681.76
3,QP[17]=8561.78	3,QP[37]=8707.9	3,QP[57]=7363.12	3,QP[77]=8692.43
3,QP[18]=8609.79	3,QP[38]=8660.23	3,QP[58]=7405.26	3,QP[78]=8691.56
3,QP[19]=8654.85	3,QP[39]=8606.96	3,QP[59]=7462.61	3,QP[79]=8678.76
3,QP[20]=8696.69	3,QP[40]=8548.16	3,QP[60]=7532.38	3,QP[80]=8653.69

2,QP[81]=-1919.43	3,QP[101]=5456.98	3,QP[121]=2002.36	3,QP[141]=3573.18
2,QP[82]=-1934.19	3,QP[102]=5222.22	3,QP[122]=2000.25	3,QP[142]=3699.17
2,QP[83]=-1947.35	3,QP[103]=4986.59	3,QP[123]=2018.01	3,QP[143]=3826.59
2,QP[84]=-1958.88	3,QP[104]=4751.19	3,QP[124]=2043.19	3,QP[144]=3955.07
2,QP[85]=-1968.76	3,QP[105]=4517.11	3,QP[125]=2076.2	3,QP[145]=4084.23
2,QP[86]=-1976.96	3,QP[106]=4285.48	3,QP[126]=2117.59	3,QP[146]=4213.71
2,QP[87]=-1983.45	3,QP[107]=4057.45	3,QP[127]=2167.84	3,QP[147]=4343.13
2,QP[88]=-1988.22	3,QP[108]=3834.21	3,QP[128]=2227.38	3,QP[148]=4472.13
2,QP[89]=-1991.27	3,QP[109]=3616.98	3,QP[129]=2299.27	3,QP[149]=4600.35
2,QP[90]=-1992.57	3,QP[110]=3407.05	3,QP[130]=2380.8	3,QP[150]=4727.43
2,QP[91]=-1992.14	3,QP[111]=3205.72	3,QP[131]=2468.04	3,QP[151]=4853.02
2,QP[92]=-1990.02	3,QP[112]=3014.38	3,QP[132]=2560.62	3,QP[152]=4976.77
2,QP[93]=-1986.2	3,QP[113]=2834.48	3,QP[133]=2658.18	3,QP[153]=5098.35
2,QP[94]=-1980.69	3,QP[114]=2667.54	3,QP[134]=2760.34	3,QP[154]=5217.43
2,QP[95]=-1973.52	3,QP[115]=2515.15	3,QP[135]=2866.74	3,QP[155]=5333.7
2,QP[96]=-1964.7	3,QP[116]=2378.97	3,QP[136]=2976.99	3,QP[156]=5446.84
2,QP[97]=-1954.26	3,QP[117]=2260.7	3,QP[137]=3090.7	3,QP[157]=5556.57
2,QP[98]=-1942.23	3,QP[118]=2162.12	3,QP[138]=3207.5	3,QP[158]=5662.6
2,QP[99]=-1928.63	3,QP[119]=2085.02	3,QP[139]=3327	3,QP[159]=5764.67
2,QP[100]=-1913.51	3,QP[120]=2031.18	3,QP[140]=3449	3,QP[160]=5862.51

3,QP[161]=5955.9	3,QP[181]=6671.05	3,QP[201]=6459.71	3,QP[221]=6466.74
3,QP[162]=6044.62	3,QP[182]=6658.82	3,QP[202]=6452.9	3,QP[222]=6477.98
3,QP[163]=6128.45	3,QP[183]=6646.6	3,QP[203]=6446.66	3,QP[223]=6490.6
3,QP[164]=6207.22	3,QP[184]=6634.43	3,QP[204]=6441.01	3,QP[224]=6504.6
3,QP[165]=6280.74	3,QP[185]=6622.33	3,QP[205]=6435.99	3,QP[225]=6519.97
3,QP[166]=6348.86	3,QP[186]=6610.32	3,QP[206]=6431.62	3,QP[226]=6536.72
3,QP[167]=6411.44	3,QP[187]=6598.44	3,QP[207]=6427.94	3,QP[227]=6554.83
3,QP[168]=6468.36	3,QP[188]=6586.7	3,QP[208]=6424.98	3,QP[228]=6574.31
3,QP[169]=6519.51	3,QP[189]=6575.13	3,QP[209]=6422.76	3,QP[229]=6595.14
3,QP[170]=6564.78	3,QP[190]=6563.76	3,QP[210]=6421.32	3,QP[230]=6617.34
3,QP[171]=6604.09	3,QP[191]=6552.62	3,QP[211]=6420.68	3,QP[231]=6640.88
3,QP[172]=6637.38	3,QP[192]=6541.73	3,QP[212]=6420.88	3,QP[232]=6665.76
3,QP[173]=6664.57	3,QP[193]=6531.12	3,QP[213]=6421.94	3,QP[233]=6691.99
3,QP[174]=6685.61	3,QP[194]=6520.82	3,QP[214]=6423.9	3,QP[234]=6719.54
3,QP[175]=6700.46	3,QP[195]=6510.85	3,QP[215]=6426.78	3,QP[235]=6748.41
3,QP[176]=6709.06	3,QP[196]=6501.25	3,QP[216]=6430.62	3,QP[236]=6778.6
3,QP[177]=6711.38	3,QP[197]=6492.04	3,QP[217]=6435.43	3,QP[237]=6810.09
3,QP[178]=6707.37	3,QP[198]=6483.26	3,QP[218]=6441.27	3,QP[238]=6842.88
3,QP[179]=6696.98	3,QP[199]=6474.92	3,QP[219]=6448.38	3,QP[239]=6876.95
3,QP[180]=6683.28	3,QP[200]=6467.06	3,QP[220]=6456.87	3,QP[240]=6912.3

3,QP[241]=6948.93	3,QP[249]=7286.36
3,QP[242]=6986.81	3,QP[250]=7333.95
3,QP[243]=7025.94	3,QP[251]=7382.7
3,QP[244]=7066.3	3,QP[252]=7432.61
3,QP[245]=7107.9	3,QP[253]=7483.66
3,QP[246]=7150.71	3,QP[254]=7535.86
3,QP[247]=7194.74	3,QP[255]=7589.18
3,QP[248]=7239.96	3,QP[256]=7643.63

Right Knee lateral joint angle value (4,QP[256])			
4,QP[1]=-11668.6	4,QP[21]=-11175.4	4,QP[41]=-7918.46	4,QP[61]=-5989.28
4,QP[2]=-11709.6	4,QP[22]=-11075.3	4,QP[42]=-7699.61	4,QP[62]=-6214.57
4,QP[3]=-11744	4,QP[23]=-10968.1	4,QP[43]=-7477.61	4,QP[63]=-6456.58
4,QP[4]=-11771.9	4,QP[24]=-10853.5	4,QP[44]=-7252.91	4,QP[64]=-6711.68
4,QP[5]=-11793.1	4,QP[25]=-10731.8	4,QP[45]=-7026	4,QP[65]=-6976.67
4,QP[6]=-11807.6	4,QP[26]=-10603	4,QP[46]=-6797.45	4,QP[66]=-7248.76
4,QP[7]=-11815.4	4,QP[27]=-10467.2	4,QP[47]=-6567.84	4,QP[67]=-7525.53
4,QP[8]=-11816.1	4,QP[28]=-10324.5	4,QP[48]=-6337.84	4,QP[68]=-7804.86
4,QP[9]=-11809.9	4,QP[29]=-10175	4,QP[49]=-6108.21	4,QP[69]=-8084.9
4,QP[10]=-11796.5	4,QP[30]=-10018.8	4,QP[50]=-5879.77	4,QP[70]=-8364.02
4,QP[11]=-11775.8	4,QP[31]=-9856.14	4,QP[51]=-5653.44	4,QP[71]=-8640.77
4,QP[12]=-11747.9	4,QP[32]=-9687.16	4,QP[52]=-5430.29	4,QP[72]=-8913.87
4,QP[13]=-11713	4,QP[33]=-9512.03	4,QP[53]=-5315.81	4,QP[73]=-9182.15
4,QP[14]=-11671.1	4,QP[34]=-9330.97	4,QP[54]=-5251.16	4,QP[74]=-9444.54
4,QP[15]=-11622	4,QP[35]=-9144.19	4,QP[55]=-5235.54	4,QP[75]=-9700.07
4,QP[16]=-11565.7	4,QP[36]=-8951.95	4,QP[56]=-5266.84	4,QP[76]=-9947.82
4,QP[17]=-11502.3	4,QP[37]=-8754.49	4,QP[57]=-5341.78	4,QP[77]=-10186.9
4,QP[18]=-11431.5	4,QP[38]=-8552.09	4,QP[58]=-5456.23	4,QP[78]=-10416.6
4,QP[19]=-11353.4	4,QP[39]=-8345.07	4,QP[59]=-5605.51	4,QP[79]=-10636.1
4,QP[20]=-11268.1	4,QP[40]=-8133.74	4,QP[60]=-5784.77	4,QP[80]=-10844.7

4,QP[81]=-11041.7	4,QP[101]=-11914.6	4,QP[121]=-9275.75	4,QP[141]=-11082.8
4,QP[82]=-11226.4	4,QP[102]=-11802.8	4,QP[122]=-9273.69	4,QP[142]=-11174.1
4,QP[83]=-11398.3	4,QP[103]=-11679.5	4,QP[123]=-9296.04	4,QP[143]=-11261.7
4,QP[84]=-11556.7	4,QP[104]=-11545.6	4,QP[124]=-9334.5	4,QP[144]=-11345.3
4,QP[85]=-11701.1	4,QP[105]=-11402.2	4,QP[125]=-9390.27	4,QP[145]=-11424.6
4,QP[86]=-11831.1	4,QP[106]=-11250.8	4,QP[126]=-9464.58	4,QP[146]=-11499.3
4,QP[87]=-11946.1	4,QP[107]=-11092.5	4,QP[127]=-9558.61	4,QP[147]=-11569.3
4,QP[88]=-12045.7	4,QP[108]=-10929	4,QP[128]=-9673.42	4,QP[148]=-11634.4
4,QP[89]=-12129.7	4,QP[109]=-10761.9	4,QP[129]=-9795.37	4,QP[149]=-11694.3
4,QP[90]=-12197.8	4,QP[110]=-10593	4,QP[130]=-9908.68	4,QP[150]=-11748.9
4,QP[91]=-12249.8	4,QP[111]=-10424.3	4,QP[131]=-10022.2	4,QP[151]=-11798
4,QP[92]=-12286	4,QP[112]=-10257.8	4,QP[132]=-10135.6	4,QP[152]=-11841.6
4,QP[93]=-12306.3	4,QP[113]=-10095.8	4,QP[133]=-10248.3	4,QP[153]=-11879.6
4,QP[94]=-12310.6	4,QP[114]=-9940.78	4,QP[134]=-10360.1	4,QP[154]=-11911.8
4,QP[95]=-12299.2	4,QP[115]=-9795.24	4,QP[135]=-10470.5	4,QP[155]=-11938.2
4,QP[96]=-12272	4,QP[116]=-9661.88	4,QP[136]=-10579.1	4,QP[156]=-11958.9
4,QP[97]=-12229.5	4,QP[117]=-9543.47	4,QP[137]=-10685.5	4,QP[157]=-11973.8
4,QP[98]=-12171.9	4,QP[118]=-9442.85	4,QP[138]=-10789.3	4,QP[158]=-11983
4,QP[99]=-12099.9	4,QP[119]=-9362.86	4,QP[139]=-10890.1	4,QP[159]=-11986.5
4,QP[100]=-12013.9	4,QP[120]=-9306.28	4,QP[140]=-10988	4,QP[160]=-11984.3

4,QP[161]=-11976.7	4,QP[181]=-11030.7	4,QP[201]=-10399.2	4,QP[221]=-10149.3
4,QP[162]=-11963.7	4,QP[182]=-10996.1	4,QP[202]=-10374.2	4,QP[222]=-10155.6
4,QP[163]=-11945.4	4,QP[183]=-10961.5	4,QP[203]=-10350.3	4,QP[223]=-10164.2
4,QP[164]=-11922	4,QP[184]=-10927.1	4,QP[204]=-10327.4	4,QP[224]=-10175.2
4,QP[165]=-11893.7	4,QP[185]=-10892.7	4,QP[205]=-10305.6	4,QP[225]=-10188.5
4,QP[166]=-11860.8	4,QP[186]=-10858.6	4,QP[206]=-10284.9	4,QP[226]=-10204.2
4,QP[167]=-11823.3	4,QP[187]=-10824.6	4,QP[207]=-10265.5	4,QP[227]=-10222.1
4,QP[168]=-11781.6	4,QP[188]=-10791	4,QP[208]=-10247.2	4,QP[228]=-10242.4
4,QP[169]=-11735.9	4,QP[189]=-10757.7	4,QP[209]=-10230.3	4,QP[229]=-10265
4,QP[170]=-11686.5	4,QP[190]=-10724.7	4,QP[210]=-10214.7	4,QP[230]=-10289.9
4,QP[171]=-11633.6	4,QP[191]=-10692.1	4,QP[211]=-10200.5	4,QP[231]=-10317
4,QP[172]=-11577.4	4,QP[192]=-10660	4,QP[212]=-10187.8	4,QP[232]=-10346.4
4,QP[173]=-11518.4	4,QP[193]=-10628.4	4,QP[213]=-10176.6	4,QP[233]=-10378.1
4,QP[174]=-11456.7	4,QP[194]=-10597.3	4,QP[214]=-10166.9	4,QP[234]=-10411.9
4,QP[175]=-11392.7	4,QP[195]=-10566.8	4,QP[215]=-10158.8	4,QP[235]=-10447.9
4,QP[176]=-11326.5	4,QP[196]=-10537	4,QP[216]=-10152.4	4,QP[236]=-10486.1
4,QP[177]=-11258.6	4,QP[197]=-10507.8	4,QP[217]=-10147.6	4,QP[237]=-10526.5
4,QP[178]=-11189.1	4,QP[198]=-10479.4	4,QP[218]=-10144.6	4,QP[238]=-10569
4,QP[179]=-11118.4	4,QP[199]=-10451.8	4,QP[219]=-10143.8	4,QP[239]=-10613.6
4,QP[180]=-11065.3	4,QP[200]=-10425.1	4,QP[220]=-10145.4	4,QP[240]=-10660.3

4,QP[241]=-10709	4,QP[249]=-11169.7
4,QP[242]=-10759.8	4,QP[250]=-11235.8
4,QP[243]=-10812.5	4,QP[251]=-11303.6
4,QP[244]=-10867.3	4,QP[252]=-11373.2
4,QP[245]=-10924	4,QP[253]=-11444.5
4,QP[246]=-10982.6	4,QP[254]=-11517.6
4,QP[247]=-11043.1	4,QP[255]=-11592.3
4,QP[248]=-11105.5	4,QP[256]=-11668.6

Right Hip lateral joint angle value (5,QP[256])			
5,QP[1]=4024.98	5,QP[21]=2440.33	5,QP[41]=-565.439	5,QP[61]=-1622.52
5,QP[2]=4012.48	5,QP[22]=2305.76	5,QP[42]=-714.665	5,QP[62]=-1483.7
5,QP[3]=3991.53	5,QP[23]=2167.92	5,QP[43]=-861.797	5,QP[63]=-1332.83
5,QP[4]=3962.34	5,QP[24]=2027.1	5,QP[44]=-1006.52	5,QP[64]=-1171.4
5,QP[5]=3925.15	5,QP[25]=1883.56	5,QP[45]=-1148.51	5,QP[65]=-1000.67
5,QP[6]=3880.16	5,QP[26]=1737.58	5,QP[46]=-1287.41	5,QP[66]=-821.745
5,QP[7]=3827.6	5,QP[27]=1589.43	5,QP[47]=-1422.84	5,QP[67]=-635.557
5,QP[8]=3767.69	5,QP[28]=1439.38	5,QP[48]=-1554.39	5,QP[68]=-442.896
5,QP[9]=3700.67	5,QP[29]=1287.69	5,QP[49]=-1681.63	5,QP[69]=-244.436
5,QP[10]=3626.76	5,QP[30]=1134.63	5,QP[50]=-1804.05	5,QP[70]=-40.7568
5,QP[11]=3546.2	5,QP[31]=980.464	5,QP[51]=-1921.13	5,QP[71]=167.639
5,QP[12]=3459.29	5,QP[32]=825.443	5,QP[52]=-2032.27	5,QP[72]=380.308
5,QP[13]=3366.42	5,QP[33]=669.83	5,QP[53]=-2080.09	5,QP[73]=596.847
5,QP[14]=3267.83	5,QP[34]=513.881	5,QP[54]=-2102.48	5,QP[74]=816.885
5,QP[15]=3163.8	5,QP[35]=357.853	5,QP[55]=-2099.56	5,QP[75]=1040.07
5,QP[16]=3054.59	5,QP[36]=202.002	5,QP[56]=-2072.08	5,QP[76]=1266.06
5,QP[17]=2940.47	5,QP[37]=46.5862	5,QP[57]=-2021.33	5,QP[77]=1494.51
5,QP[18]=2821.71	5,QP[38]=-108.133	5,QP[58]=-1949.03	5,QP[78]=1725.07
5,QP[19]=2698.58	5,QP[39]=-261.892	5,QP[59]=-1857.11	5,QP[79]=1957.37
5,QP[20]=2571.36	5,QP[40]=-414.419	5,QP[60]=-1747.61	5,QP[80]=2191.03

5,QP[81]=2425.63	5,QP[101]=6457.58	5,QP[121]=7273.39	5,QP[141]=7509.6
5,QP[82]=2660.74	5,QP[102]=6580.57	5,QP[122]=7273.44	5,QP[142]=7474.95
5,QP[83]=2895.88	5,QP[103]=6692.86	5,QP[123]=7278.02	5,QP[143]=7435.13
5,QP[84]=3130.55	5,QP[104]=6794.37	5,QP[124]=7291.31	5,QP[144]=7390.23
5,QP[85]=3364.22	5,QP[105]=6885.14	5,QP[125]=7314.06	5,QP[145]=7340.35
5,QP[86]=3596.3	5,QP[106]=6965.3	5,QP[126]=7346.99	5,QP[146]=7285.64
5,QP[87]=3826.19	5,QP[107]=7035.08	5,QP[127]=7390.77	5,QP[147]=7226.22
5,QP[88]=4053.27	5,QP[108]=7094.82	5,QP[128]=7446.04	5,QP[148]=7162.26
5,QP[89]=4276.87	5,QP[109]=7144.94	5,QP[129]=7496.11	5,QP[149]=7093.94
5,QP[90]=4496.31	5,QP[110]=7185.97	5,QP[130]=7527.88	5,QP[150]=7021.46
5,QP[91]=4710.97	5,QP[111]=7218.55	5,QP[131]=7554.17	5,QP[151]=6945.02
5,QP[92]=4920.26	5,QP[112]=7243.4	5,QP[132]=7574.95	5,QP[152]=6864.85
5,QP[93]=5123.52	5,QP[113]=7261.33	5,QP[133]=7590.16	5,QP[153]=6781.21
5,QP[94]=5320.05	5,QP[114]=7273.24	5,QP[134]=7599.78	5,QP[154]=6694.35
5,QP[95]=5509.19	5,QP[115]=7280.09	5,QP[135]=7603.77	5,QP[155]=6604.55
5,QP[96]=5690.34	5,QP[116]=7282.91	5,QP[136]=7602.11	5,QP[156]=6512.09
5,QP[97]=5862.89	5,QP[117]=7282.77	5,QP[137]=7594.78	5,QP[157]=6417.27
5,QP[98]=6026.33	5,QP[118]=7280.73	5,QP[138]=7581.77	5,QP[158]=6320.41
5,QP[99]=6180.17	5,QP[119]=7277.84	5,QP[139]=7563.1	5,QP[159]=6221.82
5,QP[100]=6324.03	5,QP[120]=7275.1	5,QP[140]=7539.01	5,QP[160]=6121.83

5,QP[161]=6020.79	5,QP[181]=4359.63	5,QP[201]=3939.46	5,QP[221]=3682.54
5,QP[162]=5919.04	5,QP[182]=4337.26	5,QP[202]=3921.34	5,QP[222]=3677.58
5,QP[163]=5816.92	5,QP[183]=4314.92	5,QP[203]=3903.64	5,QP[223]=3673.59
5,QP[164]=5714.78	5,QP[184]=4292.62	5,QP[204]=3886.39	5,QP[224]=3670.58
5,QP[165]=5612.99	5,QP[185]=4270.39	5,QP[205]=3869.6	5,QP[225]=3668.53
5,QP[166]=5511.9	5,QP[186]=4248.25	5,QP[206]=3853.31	5,QP[226]=3667.45
5,QP[167]=5411.87	5,QP[187]=4226.21	5,QP[207]=3837.52	5,QP[227]=3667.32
5,QP[168]=5313.25	5,QP[188]=4204.3	5,QP[208]=3822.26	5,QP[228]=3668.13
5,QP[169]=5216.41	5,QP[189]=4182.52	5,QP[209]=3807.55	5,QP[229]=3669.89
5,QP[170]=5121.7	5,QP[190]=4160.91	5,QP[210]=3793.41	5,QP[230]=3672.57
5,QP[171]=5029.46	5,QP[191]=4139.48	5,QP[211]=3779.87	5,QP[231]=3676.16
5,QP[172]=4940.05	5,QP[192]=4118.25	5,QP[212]=3766.94	5,QP[232]=3680.67
5,QP[173]=4853.81	5,QP[193]=4097.24	5,QP[213]=3754.64	5,QP[233]=3686.07
5,QP[174]=4771.08	5,QP[194]=4076.46	5,QP[214]=3742.99	5,QP[234]=3692.36
5,QP[175]=4692.19	5,QP[195]=4055.95	5,QP[215]=3732.02	5,QP[235]=3699.52
5,QP[176]=4617.47	5,QP[196]=4035.72	5,QP[216]=3721.73	5,QP[236]=3707.54
5,QP[177]=4547.22	5,QP[197]=4015.78	5,QP[217]=3712.16	5,QP[237]=3716.41
5,QP[178]=4481.77	5,QP[198]=3996.17	5,QP[218]=3703.33	5,QP[238]=3726.11
5,QP[179]=4421.4	5,QP[199]=3976.9	5,QP[219]=3695.41	5,QP[239]=3736.63
5,QP[180]=4382.02	5,QP[200]=3957.99	5,QP[220]=3688.48	5,QP[240]=3747.95

5,QP[241]=3760.06	5,QP[249]=3883.39
5,QP[242]=3772.95	5,QP[250]=3901.85
5,QP[243]=3786.59	5,QP[251]=3920.93
5,QP[244]=3800.97	5,QP[252]=3940.62
5,QP[245]=3816.07	5,QP[253]=3960.88
5,QP[246]=3831.88	5,QP[254]=3981.71
5,QP[247]=3848.38	5,QP[255]=4003.08
5,QP[248]=3865.56	5,QP[256]=4024.98

Right Hip frontal joint angle value (6,QP[256])			
6,QP[1]=2596.06	6,QP[21]=1184.84	6,QP[41]=-252.116	6,QP[61]=-1362.96
6,QP[2]=2528.85	6,QP[22]=1112.28	6,QP[42]=-322.162	6,QP[62]=-1399.49
6,QP[3]=2461.15	6,QP[23]=1039.69	6,QP[43]=-391.953	6,QP[63]=-1435.35
6,QP[4]=2392.98	6,QP[24]=967.099	6,QP[44]=-461.479	6,QP[64]=-1470.53
6,QP[5]=2324.37	6,QP[25]=894.514	6,QP[45]=-530.734	6,QP[65]=-1504.99
6,QP[6]=2255.32	6,QP[26]=821.958	6,QP[46]=-599.711	6,QP[66]=-1538.68
6,QP[7]=2185.87	6,QP[27]=749.448	6,QP[47]=-668.403	6,QP[67]=-1571.56
6,QP[8]=2116.03	6,QP[28]=677.002	6,QP[48]=-736.804	6,QP[68]=-1603.6
6,QP[9]=2045.83	6,QP[29]=604.637	6,QP[49]=-804.91	6,QP[69]=-1634.74
6,QP[10]=1975.29	6,QP[30]=532.368	6,QP[50]=-872.715	6,QP[70]=-1664.93
6,QP[11]=1904.42	6,QP[31]=460.212	6,QP[51]=-940.215	6,QP[71]=-1694.12
6,QP[12]=1833.28	6,QP[32]=388.183	6,QP[52]=-1007.41	6,QP[72]=-1722.26
6,QP[13]=1761.89	6,QP[33]=316.297	6,QP[53]=-1049.27	6,QP[73]=-1749.3
6,QP[14]=1690.29	6,QP[34]=244.566	6,QP[54]=-1090.52	6,QP[74]=-1775.18
6,QP[15]=1618.49	6,QP[35]=173.006	6,QP[55]=-1131.18	6,QP[75]=-1799.84
6,QP[16]=1546.52	6,QP[36]=101.628	6,QP[56]=-1171.26	6,QP[76]=-1823.24
6,QP[17]=1474.4	6,QP[37]=30.4446	6,QP[57]=-1210.76	6,QP[77]=-1845.31
6,QP[18]=1402.15	6,QP[38]=-40.5322	6,QP[58]=-1249.69	6,QP[78]=-1866.02
6,QP[19]=1329.8	6,QP[39]=-111.291	6,QP[59]=-1288.05	6,QP[79]=-1885.31
6,QP[20]=1257.35	6,QP[40]=-181.823	6,QP[60]=-1325.81	6,QP[80]=-1903.13

6,QP[81]=-1919.43	6,QP[101]=-1896.91	6,QP[121]=-1326.17	6,QP[141]=-149.016
6,QP[82]=-1934.19	6,QP[102]=-1878.86	6,QP[122]=-1289.07	6,QP[142]=-77.1133
6,QP[83]=-1947.35	6,QP[103]=-1859.42	6,QP[123]=-1251.37	6,QP[143]=-5.14283
6,QP[84]=-1958.88	6,QP[104]=-1838.63	6,QP[124]=-1213.07	6,QP[144]=66.8762
6,QP[85]=-1968.76	6,QP[105]=-1816.54	6,QP[125]=-1174.17	6,QP[145]=138.925
6,QP[86]=-1976.96	6,QP[106]=-1793.19	6,QP[126]=-1134.66	6,QP[146]=210.983
6,QP[87]=-1983.45	6,QP[107]=-1768.65	6,QP[127]=-1094.52	6,QP[147]=283.034
6,QP[88]=-1988.22	6,QP[108]=-1742.96	6,QP[128]=-1053.73	6,QP[148]=355.059
6,QP[89]=-1991.27	6,QP[109]=-1716.18	6,QP[129]=-998.529	6,QP[149]=427.04
6,QP[90]=-1992.57	6,QP[110]=-1688.35	6,QP[130]=-929.174	6,QP[150]=498.96
6,QP[91]=-1992.14	6,QP[111]=-1659.52	6,QP[131]=-859.486	6,QP[151]=570.802
6,QP[92]=-1990.02	6,QP[112]=-1629.75	6,QP[132]=-789.485	6,QP[152]=642.549
6,QP[93]=-1986.2	6,QP[113]=-1599.08	6,QP[133]=-719.191	6,QP[153]=714.186
6,QP[94]=-1980.69	6,QP[114]=-1567.56	6,QP[134]=-648.627	6,QP[154]=785.696
6,QP[95]=-1973.52	6,QP[115]=-1535.22	6,QP[135]=-577.815	6,QP[155]=857.066
6,QP[96]=-1964.7	6,QP[116]=-1502.12	6,QP[136]=-506.777	6,QP[156]=928.281
6,QP[97]=-1954.26	6,QP[117]=-1468.28	6,QP[137]=-435.536	6,QP[157]=999.327
6,QP[98]=-1942.23	6,QP[118]=-1433.73	6,QP[138]=-364.115	6,QP[158]=1070.19
6,QP[99]=-1928.63	6,QP[119]=-1398.52	6,QP[139]=-292.537	6,QP[159]=1140.86
6,QP[100]=-1913.51	6,QP[120]=-1362.66	6,QP[140]=-220.83	6,QP[160]=1211.32

6,QP[161]=1281.56	6,QP[181]=2526.98	6,QP[201]=2481.88	6,QP[221]=2467.42
6,QP[162]=1351.57	6,QP[182]=2524.38	6,QP[202]=2480.23	6,QP[222]=2468.05
6,QP[163]=1421.34	6,QP[183]=2521.79	6,QP[203]=2478.65	6,QP[223]=2468.85
6,QP[164]=1490.86	6,QP[184]=2519.23	6,QP[204]=2477.16	6,QP[224]=2469.82
6,QP[165]=1560.12	6,QP[185]=2516.68	6,QP[205]=2475.75	6,QP[225]=2470.96
6,QP[166]=1629.11	6,QP[186]=2514.17	6,QP[206]=2474.43	6,QP[226]=2472.27
6,QP[167]=1697.82	6,QP[187]=2511.69	6,QP[207]=2473.21	6,QP[227]=2473.75
6,QP[168]=1766.25	6,QP[188]=2509.24	6,QP[208]=2472.07	6,QP[228]=2475.4
6,QP[169]=1834.39	6,QP[189]=2506.83	6,QP[209]=2471.03	6,QP[229]=2477.23
6,QP[170]=1902.22	6,QP[190]=2504.45	6,QP[210]=2470.09	6,QP[230]=2479.22
6,QP[171]=1969.76	6,QP[191]=2502.12	6,QP[211]=2469.26	6,QP[231]=2481.39
6,QP[172]=2036.98	6,QP[192]=2499.84	6,QP[212]=2468.53	6,QP[232]=2483.74
6,QP[173]=2103.89	6,QP[193]=2497.61	6,QP[213]=2467.91	6,QP[233]=2486.26
6,QP[174]=2170.49	6,QP[194]=2495.43	6,QP[214]=2467.4	6,QP[234]=2488.95
6,QP[175]=2236.76	6,QP[195]=2493.3	6,QP[215]=2467	6,QP[235]=2491.83
6,QP[176]=2302.7	6,QP[196]=2491.24	6,QP[216]=2466.72	6,QP[236]=2494.88
6,QP[177]=2368.32	6,QP[197]=2489.23	6,QP[217]=2466.57	6,QP[237]=2498.11
6,QP[178]=2433.61	6,QP[198]=2487.29	6,QP[218]=2466.54	6,QP[238]=2501.52
6,QP[179]=2498.57	6,QP[199]=2485.42	6,QP[219]=2466.66	6,QP[239]=2505.12
6,QP[180]=2529.6	6,QP[200]=2483.61	6,QP[220]=2466.96	6,QP[240]=2508.9

6,QP[241]=2512.87	6,QP[249]=2551.51
6,QP[242]=2517.02	6,QP[250]=2557.24
6,QP[243]=2521.36	6,QP[251]=2563.18
6,QP[244]=2525.9	6,QP[252]=2569.32
6,QP[245]=2530.63	6,QP[253]=2575.68
6,QP[246]=2535.55	6,QP[254]=2582.25
6,QP[247]=2540.67	6,QP[255]=2589.04
6,QP[248]=2545.99	6,QP[256]=2596.06

Left Ankle frontal joint angle value (9,QP[256])			
9,QP[1]=1033.07	9,QP[21]=-391.056	9,QP[41]=-1800.35	9,QP[61]=-2508.03
9,QP[2]=963.895	9,QP[22]=-463.009	9,QP[42]=-1868.34	9,QP[62]=-2505.63
9,QP[3]=894.371	9,QP[23]=-534.892	9,QP[43]=-1936.03	9,QP[63]=-2503.28
9,QP[4]=824.523	9,QP[24]=-606.688	9,QP[44]=-2003.41	9,QP[64]=-2500.98
9,QP[5]=754.373	9,QP[25]=-678.382	9,QP[45]=-2070.48	9,QP[65]=-2498.72
9,QP[6]=683.942	9,QP[26]=-749.958	9,QP[46]=-2137.23	9,QP[66]=-2496.51
9,QP[7]=613.251	9,QP[27]=-821.4	9,QP[47]=-2203.66	9,QP[67]=-2494.36
9,QP[8]=542.323	9,QP[28]=-892.694	9,QP[48]=-2269.77	9,QP[68]=-2492.26
9,QP[9]=471.18	9,QP[29]=-963.826	9,QP[49]=-2335.55	9,QP[69]=-2490.23
9,QP[10]=399.846	9,QP[30]=-1034.78	9,QP[50]=-2401.01	9,QP[70]=-2488.25
9,QP[11]=328.344	9,QP[31]=-1105.55	9,QP[51]=-2466.13	9,QP[71]=-2486.35
9,QP[12]=256.699	9,QP[32]=-1176.11	9,QP[52]=-2530.92	9,QP[72]=-2484.51
9,QP[13]=184.935	9,QP[33]=-1246.47	9,QP[53]=-2528.29	9,QP[73]=-2482.74
9,QP[14]=113.074	9,QP[34]=-1316.59	9,QP[54]=-2525.68	9,QP[74]=-2481.05
9,QP[15]=41.1353	9,QP[35]=-1386.49	9,QP[55]=-2523.08	9,QP[75]=-2479.43
9,QP[16]=-30.8618	9,QP[36]=-1456.13	9,QP[56]=-2520.51	9,QP[76]=-2477.9
9,QP[17]=-102.898	9,QP[37]=-1525.52	9,QP[57]=-2517.95	9,QP[77]=-2476.45
9,QP[18]=-174.954	9,QP[38]=-1594.65	9,QP[58]=-2515.42	9,QP[78]=-2475.08
9,QP[19]=-247.011	9,QP[39]=-1663.5	9,QP[59]=-2512.92	9,QP[79]=-2473.81
9,QP[20]=-319.051	9,QP[40]=-1732.07	9,QP[60]=-2510.46	9,QP[80]=-2472.63

9,QP[81]=-2471.54	9,QP[101]=-2476.29	9,QP[121]=-2548.73	9,QP[141]=-1726.11
9,QP[82]=-2470.55	9,QP[102]=-2478.2	9,QP[122]=-2554.35	9,QP[142]=-1654.41
9,QP[83]=-2469.66	9,QP[103]=-2480.29	9,QP[123]=-2560.18	9,QP[143]=-1582.53
9,QP[84]=-2468.88	9,QP[104]=-2482.54	9,QP[124]=-2566.22	9,QP[144]=-1510.48
9,QP[85]=-2468.2	9,QP[105]=-2484.98	9,QP[125]=-2572.47	9,QP[145]=-1438.29
9,QP[86]=-2467.64	9,QP[106]=-2487.58	9,QP[126]=-2578.94	9,QP[146]=-1365.99
9,QP[87]=-2467.18	9,QP[107]=-2490.37	9,QP[127]=-2585.62	9,QP[147]=-1293.58
9,QP[88]=-2466.85	9,QP[108]=-2493.33	9,QP[128]=-2592.52	9,QP[148]=-1221.1
9,QP[89]=-2466.63	9,QP[109]=-2496.47	9,QP[129]=-2562.52	9,QP[149]=-1148.57
9,QP[90]=-2466.54	9,QP[110]=-2499.79	9,QP[130]=-2495.06	9,QP[150]=-1075.99
9,QP[91]=-2466.58	9,QP[111]=-2503.3	9,QP[131]=-2427.12	9,QP[151]=-1003.4
9,QP[92]=-2466.79	9,QP[112]=-2506.99	9,QP[132]=-2358.73	9,QP[152]=-930.804
9,QP[93]=-2467.17	9,QP[113]=-2510.86	9,QP[133]=-2289.9	9,QP[153]=-858.232
9,QP[94]=-2467.72	9,QP[114]=-2514.92	9,QP[134]=-2220.65	9,QP[154]=-785.696
9,QP[95]=-2468.43	9,QP[115]=-2519.17	9,QP[135]=-2151	9,QP[155]=-713.216
9,QP[96]=-2469.32	9,QP[116]=-2523.61	9,QP[136]=-2080.98	9,QP[156]=-640.809
9,QP[97]=-2470.37	9,QP[117]=-2528.24	9,QP[137]=-2010.6	9,QP[157]=-568.489
9,QP[98]=-2471.6	9,QP[118]=-2533.06	9,QP[138]=-1939.89	9,QP[158]=-496.275
9,QP[99]=-2472.99	9,QP[119]=-2538.09	9,QP[139]=-1868.88	9,QP[159]=-424.181
9,QP[100]=-2474.56	9,QP[120]=-2543.31	9,QP[140]=-1797.61	9,QP[160]=-352.221

9,QP[161]=-280.411	9,QP[181]=1069.97	9,QP[201]=1762.39	9,QP[221]=1983.66
9,QP[162]=-208.764	9,QP[182]=1110.93	9,QP[202]=1787.66	9,QP[222]=1977.32
9,QP[163]=-137.293	9,QP[183]=1151.29	9,QP[203]=1811.7	9,QP[223]=1969.32
9,QP[164]=-66.0111	9,QP[184]=1191.08	9,QP[204]=1834.44	9,QP[224]=1959.68
9,QP[165]=5.07034	9,QP[185]=1230.3	9,QP[205]=1855.84	9,QP[225]=1948.44
9,QP[166]=75.9397	9,QP[186]=1268.94	9,QP[206]=1875.84	9,QP[226]=1935.62
9,QP[167]=146.586	9,QP[187]=1307	9,QP[207]=1894.4	9,QP[227]=1921.26
9,QP[168]=217	9,QP[188]=1344.46	9,QP[208]=1911.47	9,QP[228]=1905.39
9,QP[169]=287.171	9,QP[189]=1381.3	9,QP[209]=1927.01	9,QP[229]=1888.06
9,QP[170]=357.09	9,QP[190]=1417.5	9,QP[210]=1940.97	9,QP[230]=1869.31
9,QP[171]=426.75	9,QP[191]=1453.03	9,QP[211]=1953.32	9,QP[231]=1849.19
9,QP[172]=496.141	9,QP[192]=1487.85	9,QP[212]=1964.03	9,QP[232]=1827.74
9,QP[173]=565.258	9,QP[193]=1521.93	9,QP[213]=1973.07	9,QP[233]=1805.02
9,QP[174]=634.093	9,QP[194]=1555.22	9,QP[214]=1980.42	9,QP[234]=1781.07
9,QP[175]=702.64	9,QP[195]=1587.69	9,QP[215]=1986.05	9,QP[235]=1755.95
9,QP[176]=770.894	9,QP[196]=1619.28	9,QP[216]=1989.96	9,QP[236]=1729.7
9,QP[177]=838.85	9,QP[197]=1649.96	9,QP[217]=1992.13	9,QP[237]=1702.39
9,QP[178]=906.503	9,QP[198]=1679.65	9,QP[218]=1992.57	9,QP[238]=1674.06
9,QP[179]=973.85	9,QP[199]=1708.33	9,QP[219]=1991.29	9,QP[239]=1644.75
9,QP[180]=1028.42	9,QP[200]=1735.92	9,QP[220]=1988.32	9,QP[240]=1614.52

9,QP[241]=1583.42	9,QP[249]=1307.7
9,QP[242]=1551.49	9,QP[250]=1270.3
9,QP[243]=1518.76	9,QP[251]=1232.3
9,QP[244]=1485.29	9,QP[252]=1193.7
9,QP[245]=1451.09	9,QP[253]=1154.49
9,QP[246]=1416.21	9,QP[254]=1114.67
9,QP[247]=1380.67	9,QP[255]=1074.21
9,QP[248]=1344.49	9,QP[256]=1033.07

Left Ankle lateral joint angle value (10,QP[256])			
10,QP[1]=2260.74	10,QP[21]=4536.36	10,QP[41]=6494.66	10,QP[61]=6580.89
10,QP[2]=2339.3	10,QP[22]=4664.05	10,QP[42]=6542.88	10,QP[62]=6569.42
10,QP[3]=2423.73	10,QP[23]=4790.43	10,QP[43]=6585.18	10,QP[63]=6558.16
10,QP[4]=2513.69	10,QP[24]=4915.14	10,QP[44]=6621.49	10,QP[64]=6547.14
10,QP[5]=2608.8	10,QP[25]=5037.85	10,QP[45]=6651.74	10,QP[65]=6536.39
10,QP[6]=2708.71	10,QP[26]=5158.23	10,QP[46]=6675.86	10,QP[66]=6525.93
10,QP[7]=2813.04	10,QP[27]=5275.94	10,QP[47]=6693.81	10,QP[67]=6515.79
10,QP[8]=2921.4	10,QP[28]=5390.68	10,QP[48]=6705.54	10,QP[68]=6506.01
10,QP[9]=3033.43	10,QP[29]=5502.15	10,QP[49]=6711.01	10,QP[69]=6496.6
10,QP[10]=3148.74	10,QP[30]=5610.07	10,QP[50]=6710.17	10,QP[70]=6487.6
10,QP[11]=3266.93	10,QP[31]=5714.15	10,QP[51]=6702.97	10,QP[71]=6479.03
10,QP[12]=3387.7	10,QP[32]=5814.13	10,QP[52]=6689.38	10,QP[72]=6470.93
10,QP[13]=3510.84	10,QP[33]=5909.78	10,QP[53]=6677.16	10,QP[73]=6463.32
10,QP[14]=3635.97	10,QP[34]=6000.86	10,QP[54]=6664.93	10,QP[74]=6456.24
10,QP[15]=3762.72	10,QP[35]=6087.16	10,QP[55]=6652.71	10,QP[75]=6449.71
10,QP[16]=3890.72	10,QP[36]=6168.48	10,QP[56]=6640.51	10,QP[76]=6443.76
10,QP[17]=4019.59	10,QP[37]=6244.64	10,QP[57]=6628.37	10,QP[77]=6438.42
10,QP[18]=4148.95	10,QP[38]=6315.48	10,QP[58]=6616.31	10,QP[78]=6433.72
10,QP[19]=4278.45	10,QP[39]=6380.85	10,QP[59]=6604.36	10,QP[79]=6429.69
10,QP[20]=4407.71	10,QP[40]=6440.62	10,QP[60]=6592.55	10,QP[80]=6426.37

10,QP[81]=6423.77	10,QP[101]=6584.56	10,QP[121]=7263.01	10,QP[141]=8375.1
10,QP[82]=6421.94	10,QP[102]=6606.07	10,QP[122]=7310.01	10,QP[142]=8430.94
10,QP[83]=6420.9	10,QP[103]=6628.94	10,QP[123]=7358.18	10,QP[143]=8484.94
10,QP[84]=6420.67	10,QP[104]=6653.15	10,QP[124]=7407.51	10,QP[144]=8536.78
10,QP[85]=6421.3	10,QP[105]=6678.71	10,QP[125]=7457.99	10,QP[145]=8586.13
10,QP[86]=6422.81	10,QP[106]=6705.6	10,QP[126]=7509.62	10,QP[146]=8632.7
10,QP[87]=6425.22	10,QP[107]=6733.81	10,QP[127]=7562.38	10,QP[147]=8676.19
10,QP[88]=6428.58	10,QP[108]=6763.34	10,QP[128]=7616.27	10,QP[148]=8716.31
10,QP[89]=6432.9	10,QP[109]=6794.18	10,QP[129]=7670.1	10,QP[149]=8752.8
10,QP[90]=6438.22	10,QP[110]=6826.32	10,QP[130]=7724.55	10,QP[150]=8785.38
10,QP[91]=6444.65	10,QP[111]=6859.75	10,QP[131]=7780.81	10,QP[151]=8813.84
10,QP[92]=6452.45	10,QP[112]=6894.47	10,QP[132]=7838.59	10,QP[152]=8837.93
10,QP[93]=6461.63	10,QP[113]=6930.46	10,QP[133]=7897.6	10,QP[153]=8857.46
10,QP[94]=6472.19	10,QP[114]=6967.71	10,QP[134]=7957.54	10,QP[154]=8872.24
10,QP[95]=6484.12	10,QP[115]=7006.22	10,QP[135]=8018.07	10,QP[155]=8882.08
10,QP[96]=6497.43	10,QP[116]=7045.97	10,QP[136]=8078.86	10,QP[156]=8886.85
10,QP[97]=6512.12	10,QP[117]=7086.95	10,QP[137]=8139.55	10,QP[157]=8886.39
10,QP[98]=6528.17	10,QP[118]=7129.16	10,QP[138]=8199.79	10,QP[158]=8880.61
10,QP[99]=6545.6	10,QP[119]=7172.57	10,QP[139]=8259.23	10,QP[159]=8869.39
10,QP[100]=6564.4	10,QP[120]=7217.2	10,QP[140]=8317.75	10,QP[160]=8852.65

10,QP[161]=8830.35	10,QP[181]=7371.74	10,QP[201]=8607.69	10,QP[221]=7087.8
10,QP[162]=8802.42	10,QP[182]=7341.47	10,QP[202]=8645.11	10,QP[222]=6891.29
10,QP[163]=8768.85	10,QP[183]=7334.32	10,QP[203]=8672.23	10,QP[223]=6686.73
10,QP[164]=8729.63	10,QP[184]=7348.61	10,QP[204]=8688.51	10,QP[224]=6474.93
10,QP[165]=8684.77	10,QP[185]=7382.11	10,QP[205]=8693.47	10,QP[225]=6256.79
10,QP[166]=8634.29	10,QP[186]=7432.21	10,QP[206]=8686.68	10,QP[226]=6033.22
10,QP[167]=8578.25	10,QP[187]=7496.12	10,QP[207]=8667.78	10,QP[227]=5805.19
10,QP[168]=8516.7	10,QP[188]=7571.05	10,QP[208]=8636.47	10,QP[228]=5573.7
10,QP[169]=8449.75	10,QP[189]=7654.31	10,QP[209]=8592.49	10,QP[229]=5339.77
10,QP[170]=8377.49	10,QP[190]=7743.4	10,QP[210]=8535.67	10,QP[230]=5104.45
10,QP[171]=8300.04	10,QP[191]=7836.05	10,QP[211]=8465.9	10,QP[231]=4868.8
10,QP[172]=8217.58	10,QP[192]=7930.24	10,QP[212]=8383.15	10,QP[232]=4633.92
10,QP[173]=8130.27	10,QP[193]=8024.16	10,QP[213]=8287.44	10,QP[233]=4400.92
10,QP[174]=8038.32	10,QP[194]=8116.21	10,QP[214]=8178.89	10,QP[234]=4170.94
10,QP[175]=7941.97	10,QP[195]=8204.99	10,QP[215]=8057.71	10,QP[235]=3945.15
10,QP[176]=7841.51	10,QP[196]=8289.25	10,QP[216]=7924.16	10,QP[236]=3724.77
10,QP[177]=7737.26	10,QP[197]=8367.89	10,QP[217]=7778.61	10,QP[237]=3511.02
10,QP[178]=7629.57	10,QP[198]=8439.89	10,QP[218]=7621.52	10,QP[238]=3305.22
10,QP[179]=7518.89	10,QP[199]=8504.39	10,QP[219]=7453.59	10,QP[239]=3108.71
10,QP[180]=7426.17	10,QP[200]=8560.56	10,QP[220]=7275.47	10,QP[240]=2922.91

10,QP[241]=2749.29	10,QP[249]=1997.87
10,QP[242]=2589.43	10,QP[250]=2008.18
10,QP[243]=2444.93	10,QP[251]=2029.66
10,QP[244]=2317.48	10,QP[252]=2058.68
10,QP[245]=2208.84	10,QP[253]=2095.82
10,QP[246]=2120.77	10,QP[254]=2141.58
10,QP[247]=2055.08	10,QP[255]=2196.43
10,QP[248]=2013.53	10,QP[256]=2260.74

Left Knee lateral joint angle value (11,QP[256])			
11,QP[1]=-9738.92	11,QP[21]=-11665	11,QP[41]=-11759.3	11,QP[61]=-10774.3
11,QP[2]=-9851.98	11,QP[22]=-11722.3	11,QP[42]=-11711.6	11,QP[62]=-10741.1
11,QP[3]=-9965.45	11,QP[23]=-11774.1	11,QP[43]=-11660.4	11,QP[63]=-10708.3
11,QP[4]=-10078.9	11,QP[24]=-11820.5	11,QP[44]=-11605.9	11,QP[64]=-10676
11,QP[5]=-10192.1	11,QP[25]=-11861.3	11,QP[45]=-11548.3	11,QP[65]=-10644.1
11,QP[6]=-10304.4	11,QP[26]=-11896.4	11,QP[46]=-11487.9	11,QP[66]=-10612.7
11,QP[7]=-10415.5	11,QP[27]=-11925.7	11,QP[47]=-11425	11,QP[67]=-10582
11,QP[8]=-10525.1	11,QP[28]=-11949.3	11,QP[48]=-11359.8	11,QP[68]=-10551.8
11,QP[9]=-10632.6	11,QP[29]=-11967.1	11,QP[49]=-11292.8	11,QP[69]=-10522.3
11,QP[10]=-10737.7	11,QP[30]=-11979.1	11,QP[50]=-11224	11,QP[70]=-10493.5
11,QP[11]=-10840.1	11,QP[31]=-11985.5	11,QP[51]=-11153.9	11,QP[71]=-10465.5
11,QP[12]=-10939.4	11,QP[32]=-11986.1	11,QP[52]=-11082.6	11,QP[72]=-10438.3
11,QP[13]=-11035.8	11,QP[33]=-11981.2	11,QP[53]=-11048	11,QP[73]=-10412
11,QP[14]=-11128.9	11,QP[34]=-11970.8	11,QP[54]=-11013.4	11,QP[74]=-10386.6
11,QP[15]=-11218.4	11,QP[35]=-11955.2	11,QP[55]=-10978.8	11,QP[75]=-10362.1
11,QP[16]=-11304	11,QP[36]=-11934.3	11,QP[56]=-10944.3	11,QP[76]=-10338.7
11,QP[17]=-11385.5	11,QP[37]=-11908.5	11,QP[57]=-10909.9	11,QP[77]=-10316.4
11,QP[18]=-11462.5	11,QP[38]=-11877.8	11,QP[58]=-10875.6	11,QP[78]=-10295.1
11,QP[19]=-11535	11,QP[39]=-11842.6	11,QP[59]=-10841.6	11,QP[79]=-10275
11,QP[20]=-11602.5	11,QP[40]=-11803	11,QP[60]=-10807.8	11,QP[80]=-10256.2

11, QP[81]=-10238.6	11, QP[101]=-10253.4	11, QP[121]=-11137.4	11, QP[141]=-11692.9
11, QP[82]=-10222.3	11, QP[102]=-10277.2	11, QP[122]=-11202.5	11, QP[142]=-11647.4
11, QP[83]=-10207.5	11, QP[103]=-10303.2	11, QP[123]=-11269.5	11, QP[143]=-11594.8
11, QP[84]=-10194	11, QP[104]=-10331.5	11, QP[124]=-11338.2	11, QP[144]=-11534.9
11, QP[85]=-10182	11, QP[105]=-10362	11, QP[125]=-11408.7	11, QP[145]=-11467.8
11, QP[86]=-10171.5	11, QP[106]=-10394.7	11, QP[126]=-11480.8	11, QP[146]=-11393.4
11, QP[87]=-10162.6	11, QP[107]=-10429.6	11, QP[127]=-11554.7	11, QP[147]=-11311.7
11, QP[88]=-10155.4	11, QP[108]=-10466.8	11, QP[128]=-11630.2	11, QP[148]=-11222.6
11, QP[89]=-10149.8	11, QP[109]=-10506.1	11, QP[129]=-11689.9	11, QP[149]=-11126.3
11, QP[90]=-10145.9	11, QP[110]=-10547.5	11, QP[130]=-11727.6	11, QP[150]=-11022.6
11, QP[91]=-10143.9	11, QP[111]=-10591	11, QP[131]=-11758.8	11, QP[151]=-10911.7
11, QP[92]=-10144.3	11, QP[112]=-10636.7	11, QP[132]=-11783.3	11, QP[152]=-10793.6
11, QP[93]=-10147	11, QP[113]=-10684.4	11, QP[133]=-11801.2	11, QP[153]=-10668.3
11, QP[94]=-10152.1	11, QP[114]=-10734.1	11, QP[134]=-11812.4	11, QP[154]=-10536
11, QP[95]=-10159.6	11, QP[115]=-10785.9	11, QP[135]=-11816.6	11, QP[155]=-10396.7
11, QP[96]=-10169.4	11, QP[116]=-10839.7	11, QP[136]=-11813.9	11, QP[156]=-10250.6
11, QP[97]=-10181.6	11, QP[117]=-10895.4	11, QP[137]=-11804.1	11, QP[157]=-10097.7
11, QP[98]=-10196	11, QP[118]=-10953	11, QP[138]=-11787.1	11, QP[158]=-9938.28
11, QP[99]=-10212.9	11, QP[119]=-11012.6	11, QP[139]=-11762.7	11, QP[159]=-9772.43
11, QP[100]=-10232	11, QP[120]=-11074.1	11, QP[140]=-11731.3	11, QP[160]=-9600.35

11, QP[161]=-9422.23	11, QP[181]=-5277.27	11, QP[201]=-9314.14	11, QP[221]=-12310.5
11, QP[162]=-9238.28	11, QP[182]=-5237.32	11, QP[202]=-9573.22	11, QP[222]=-12306.9
11, QP[163]=-9048.74	11, QP[183]=-5245.5	11, QP[203]=-9824.97	11, QP[223]=-12287.5
11, QP[164]=-8853.85	11, QP[184]=-5299.1	11, QP[204]=-10068.5	11, QP[224]=-12252.7
11, QP[165]=-8653.89	11, QP[185]=-5394.35	11, QP[205]=-10303	11, QP[225]=-12202.6
11, QP[166]=-8449.14	11, QP[186]=-5526.81	11, QP[206]=-10527.7	11, QP[226]=-12137.7
11, QP[167]=-8239.92	11, QP[187]=-5691.69	11, QP[207]=-10741.8	11, QP[227]=-12058.6
11, QP[168]=-8026.57	11, QP[188]=-5884.16	11, QP[208]=-10944.7	11, QP[228]=-11965.8
11, QP[169]=-7809.45	11, QP[189]=-6099.59	11, QP[209]=-11135.6	11, QP[229]=-11860.2
11, QP[170]=-7588.97	11, QP[190]=-6333.72	11, QP[210]=-11314	11, QP[230]=-11742.5
11, QP[171]=-7365.56	11, QP[191]=-6582.7	11, QP[211]=-11479.2	11, QP[231]=-11613.8
11, QP[172]=-7139.7	11, QP[192]=-6843.12	11, QP[212]=-11630.7	11, QP[232]=-11475
11, QP[173]=-6911.9	11, QP[193]=-7111.99	11, QP[213]=-11767.9	11, QP[233]=-11327.4
11, QP[174]=-6682.73	11, QP[194]=-7386.7	11, QP[214]=-11890.4	11, QP[234]=-11172.4
11, QP[175]=-6452.84	11, QP[195]=-7665	11, QP[215]=-11997.8	11, QP[235]=-11011.3
11, QP[176]=-6222.93	11, QP[196]=-7944.9	11, QP[216]=-12089.7	11, QP[236]=-10845.8
11, QP[177]=-5993.78	11, QP[197]=-8224.67	11, QP[217]=-12165.8	11, QP[237]=-10677.6
11, QP[178]=-5766.28	11, QP[198]=-8502.77	11, QP[218]=-12225.8	11, QP[238]=-10508.5
11, QP[179]=-5541.4	11, QP[199]=-8777.85	11, QP[219]=-12269.9	11, QP[239]=-10340.6
11, QP[180]=-5366.83	11, QP[200]=-9048.68	11, QP[220]=-12298.2	11, QP[240]=-10176.1

11, QP[241]=-10017.3	11, QP[249]=-9271.02
11, QP[242]=-9866.66	11, QP[250]=-9282.89
11, QP[243]=-9726.87	11, QP[251]=-9313.19
11, QP[244]=-9600.63	11, QP[252]=-9360.14
11, QP[245]=-9490.76	11, QP[253]=-9425.03
11, QP[246]=-9400.1	11, QP[254]=-9509.07
11, QP[247]=-9331.47	11, QP[255]=-9613.36
11, QP[248]=-9287.6	11, QP[256]=-9738.92

Left Hip lateral joint angle value (12,QP[256])			
12,QP[1]=-7478.18	12,QP[21]=-7128.63	12,QP[41]=-5264.59	12,QP[61]=-4193.39
12,QP[2]=-7512.68	12,QP[22]=-7058.21	12,QP[42]=-5168.77	12,QP[62]=-4171.7
12,QP[3]=-7541.71	12,QP[23]=-6983.72	12,QP[43]=-5075.25	12,QP[63]=-4150.17
12,QP[4]=-7565.25	12,QP[24]=-6905.39	12,QP[44]=-4984.38	12,QP[64]=-4128.84
12,QP[5]=-7583.25	12,QP[25]=-6823.45	12,QP[45]=-4896.52	12,QP[65]=-4107.71
12,QP[6]=-7595.67	12,QP[26]=-6738.17	12,QP[46]=-4811.99	12,QP[66]=-4086.82
12,QP[7]=-7602.48	12,QP[27]=-6649.8	12,QP[47]=-4731.14	12,QP[67]=-4066.17
12,QP[8]=-7603.65	12,QP[28]=-6558.63	12,QP[48]=-4654.29	12,QP[68]=-4045.8
12,QP[9]=-7599.15	12,QP[29]=-6464.95	12,QP[49]=-4581.77	12,QP[69]=-4025.71
12,QP[10]=-7588.98	12,QP[30]=-6369.07	12,QP[50]=-4513.88	12,QP[70]=-4005.93
12,QP[11]=-7573.14	12,QP[31]=-6271.31	12,QP[51]=-4450.93	12,QP[71]=-3986.49
12,QP[12]=-7551.72	12,QP[32]=-6171.98	12,QP[52]=-4393.21	12,QP[72]=-3967.4
12,QP[13]=-7524.97	12,QP[33]=-6071.42	12,QP[53]=-4370.83	12,QP[73]=-3948.68
12,QP[14]=-7492.93	12,QP[34]=-5969.98	12,QP[54]=-4348.44	12,QP[74]=-3930.35
12,QP[15]=-7455.68	12,QP[35]=-5868	12,QP[55]=-4326.08	12,QP[75]=-3912.43
12,QP[16]=-7413.31	12,QP[36]=-5765.83	12,QP[56]=-4303.76	12,QP[76]=-3894.96
12,QP[17]=-7365.9	12,QP[37]=-5663.82	12,QP[57]=-4281.5	12,QP[77]=-3877.94
12,QP[18]=-7313.59	12,QP[38]=-5562.34	12,QP[58]=-4259.31	12,QP[78]=-3861.39
12,QP[19]=-7256.51	12,QP[39]=-5461.73	12,QP[59]=-4237.22	12,QP[79]=-3845.35
12,QP[20]=-7194.8	12,QP[40]=-5362.36	12,QP[60]=-4215.24	12,QP[80]=-3829.82

12,QP[81]=-3814.83	12,QP[101]=-3668.89	12,QP[121]=-3874.39	12,QP[141]=-3317.82
12,QP[82]=-3800.41	12,QP[102]=-3671.11	12,QP[122]=-3892.54	12,QP[142]=-3216.48
12,QP[83]=-3786.57	12,QP[103]=-3674.25	12,QP[123]=-3911.32	12,QP[143]=-3109.82
12,QP[84]=-3773.33	12,QP[104]=-3678.31	12,QP[124]=-3930.7	12,QP[144]=-2998.13
12,QP[85]=-3760.71	12,QP[105]=-3683.26	12,QP[125]=-3950.68	12,QP[145]=-2881.65
12,QP[86]=-3748.73	12,QP[106]=-3689.11	12,QP[126]=-3971.23	12,QP[146]=-2760.67
12,QP[87]=-3737.42	12,QP[107]=-3695.84	12,QP[127]=-3992.33	12,QP[147]=-2635.47
12,QP[88]=-3726.79	12,QP[108]=-3703.43	12,QP[128]=-4013.97	12,QP[148]=-2506.3
12,QP[89]=-3716.86	12,QP[109]=-3711.87	12,QP[129]=-4019.8	12,QP[149]=-2373.47
12,QP[90]=-3707.65	12,QP[110]=-3721.16	12,QP[130]=-4003.05	12,QP[150]=-2237.23
12,QP[91]=-3699.24	12,QP[111]=-3731.27	12,QP[131]=-3977.95	12,QP[151]=-2097.86
12,QP[92]=-3691.82	12,QP[112]=-3742.19	12,QP[132]=-3944.73	12,QP[152]=-1955.65
12,QP[93]=-3685.39	12,QP[113]=-3753.91	12,QP[133]=-3903.61	12,QP[153]=-1810.86
12,QP[94]=-3679.93	12,QP[114]=-3766.41	12,QP[134]=-3854.81	12,QP[154]=-1663.76
12,QP[95]=-3675.46	12,QP[115]=-3779.67	12,QP[135]=-3798.55	12,QP[155]=-1514.63
12,QP[96]=-3671.96	12,QP[116]=-3793.68	12,QP[136]=-3735.06	12,QP[156]=-1363.72
12,QP[97]=-3669.44	12,QP[117]=-3808.43	12,QP[137]=-3664.56	12,QP[157]=-1211.32
12,QP[98]=-3667.87	12,QP[118]=-3823.89	12,QP[138]=-3587.3	12,QP[158]=-1057.67
12,QP[99]=-3667.26	12,QP[119]=-3840.05	12,QP[139]=-3503.51	12,QP[159]=-903.044
12,QP[100]=-3667.61	12,QP[120]=-3856.89	12,QP[140]=-3413.58	12,QP[160]=-747.695

12, QP[161]=-591.882	12, QP[181]=2094.47	12, QP[201]=-706.451	12, QP[221]=-5222.66
12, QP[162]=-435.861	12, QP[182]=2104.15	12, QP[202]=-928.107	12, QP[222]=-5415.58
12, QP[163]=-279.889	12, QP[183]=2088.82	12, QP[203]=-1152.74	12, QP[223]=-5600.8
12, QP[164]=-124.223	12, QP[184]=2049.52	12, QP[204]=-1380	12, QP[224]=-5777.72
12, QP[165]=30.8769	12, QP[185]=1987.76	12, QP[205]=-1609.55	12, QP[225]=-5945.78
12, QP[166]=185.149	12, QP[186]=1905.39	12, QP[206]=-1841.03	12, QP[226]=-6104.48
12, QP[167]=338.327	12, QP[187]=1804.43	12, QP[207]=-2074.06	12, QP[227]=-6253.38
12, QP[168]=490.136	12, QP[188]=1686.89	12, QP[208]=-2308.24	12, QP[228]=-6392.11
12, QP[169]=640.295	12, QP[189]=1554.72	12, QP[209]=-2543.15	12, QP[229]=-6520.41
12, QP[170]=788.511	12, QP[190]=1409.68	12, QP[210]=-2778.34	12, QP[230]=-6638.06
12, QP[171]=934.48	12, QP[191]=1253.35	12, QP[211]=-3013.31	12, QP[231]=-6744.96
12, QP[172]=1077.88	12, QP[192]=1087.12	12, QP[212]=-3247.55	12, QP[232]=-6841.09
12, QP[173]=1218.37	12, QP[193]=912.17	12, QP[213]=-3480.49	12, QP[233]=-6926.53
12, QP[174]=1355.58	12, QP[194]=729.507	12, QP[214]=-3711.56	12, QP[234]=-7001.47
12, QP[175]=1489.13	12, QP[195]=539.991	12, QP[215]=-3940.12	12, QP[235]=-7066.18
12, QP[176]=1618.58	12, QP[196]=344.353	12, QP[216]=-4165.54	12, QP[236]=-7121.05
12, QP[177]=1743.47	12, QP[197]=143.216	12, QP[217]=-4387.15	12, QP[237]=-7166.55
12, QP[178]=1863.3	12, QP[198]=-62.8807	12, QP[218]=-4604.27	12, QP[238]=-7203.28
12, QP[179]=1977.49	12, QP[199]=-273.465	12, QP[219]=-4816.33	12, QP[239]=-7231.89
12, QP[180]=2059.34	12, QP[200]=-488.117	12, QP[220]=-5022.69	12, QP[240]=-7253.18

12, QP[241]=-7267.98	12, QP[249]=-7273.16
12, QP[242]=-7277.23	12, QP[250]=-7274.71
12, QP[243]=-7281.94	12, QP[251]=-7283.53
12, QP[244]=-7283.15	12, QP[252]=-7301.46
12, QP[245]=-7281.92	12, QP[253]=-7329.21
12, QP[246]=-7279.33	12, QP[254]=-7367.48
12, QP[247]=-7276.39	12, QP[255]=-7416.93
12, QP[248]=-7274.07	12, QP[256]=-7478.18

Left Hip frontal joint angle value (13, QP[256])			
13, QP[1]=1033.07	13, QP[21]=-391.056	13, QP[41]=-1800.35	13, QP[61]=-2508.03
13, QP[2]=963.895	13, QP[22]=-463.009	13, QP[42]=-1868.34	13, QP[62]=-2505.63
13, QP[3]=894.371	13, QP[23]=-534.892	13, QP[43]=-1936.03	13, QP[63]=-2503.28
13, QP[4]=824.523	13, QP[24]=-606.688	13, QP[44]=-2003.41	13, QP[64]=-2500.98
13, QP[5]=754.373	13, QP[25]=-678.382	13, QP[45]=-2070.48	13, QP[65]=-2498.72
13, QP[6]=683.942	13, QP[26]=-749.958	13, QP[46]=-2137.23	13, QP[66]=-2496.51
13, QP[7]=613.251	13, QP[27]=-821.4	13, QP[47]=-2203.66	13, QP[67]=-2494.36
13, QP[8]=542.323	13, QP[28]=-892.694	13, QP[48]=-2269.77	13, QP[68]=-2492.26
13, QP[9]=471.18	13, QP[29]=-963.826	13, QP[49]=-2335.55	13, QP[69]=-2490.23
13, QP[10]=399.846	13, QP[30]=-1034.78	13, QP[50]=-2401.01	13, QP[70]=-2488.25
13, QP[11]=328.344	13, QP[31]=-1105.55	13, QP[51]=-2466.13	13, QP[71]=-2486.35
13, QP[12]=256.699	13, QP[32]=-1176.11	13, QP[52]=-2530.92	13, QP[72]=-2484.51
13, QP[13]=184.935	13, QP[33]=-1246.47	13, QP[53]=-2528.29	13, QP[73]=-2482.74
13, QP[14]=113.074	13, QP[34]=-1316.59	13, QP[54]=-2525.68	13, QP[74]=-2481.05
13, QP[15]=41.1353	13, QP[35]=-1386.49	13, QP[55]=-2523.08	13, QP[75]=-2479.43
13, QP[16]=-30.8618	13, QP[36]=-1456.13	13, QP[56]=-2520.51	13, QP[76]=-2477.9
13, QP[17]=-102.898	13, QP[37]=-1525.52	13, QP[57]=-2517.95	13, QP[77]=-2476.45
13, QP[18]=-174.954	13, QP[38]=-1594.65	13, QP[58]=-2515.42	13, QP[78]=-2475.08
13, QP[19]=-247.011	13, QP[39]=-1663.5	13, QP[59]=-2512.92	13, QP[79]=-2473.81
13, QP[20]=-319.051	13, QP[40]=-1732.07	13, QP[60]=-2510.46	13, QP[80]=-2472.63

13, QP[81]=-2471.54	13, QP[101]=-2476.29	13, QP[121]=-2548.73	13, QP[141]=-1726.11
13, QP[82]=-2470.55	13, QP[102]=-2478.2	13, QP[122]=-2554.35	13, QP[142]=-1654.41
13, QP[83]=-2469.66	13, QP[103]=-2480.29	13, QP[123]=-2560.18	13, QP[143]=-1582.53
13, QP[84]=-2468.88	13, QP[104]=-2482.54	13, QP[124]=-2566.22	13, QP[144]=-1510.48
13, QP[85]=-2468.2	13, QP[105]=-2484.98	13, QP[125]=-2572.47	13, QP[145]=-1438.29
13, QP[86]=-2467.64	13, QP[106]=-2487.58	13, QP[126]=-2578.94	13, QP[146]=-1365.99
13, QP[87]=-2467.18	13, QP[107]=-2490.37	13, QP[127]=-2585.62	13, QP[147]=-1293.58
13, QP[88]=-2466.85	13, QP[108]=-2493.33	13, QP[128]=-2592.52	13, QP[148]=-1221.1
13, QP[89]=-2466.63	13, QP[109]=-2496.47	13, QP[129]=-2562.52	13, QP[149]=-1148.57
13, QP[90]=-2466.54	13, QP[110]=-2499.79	13, QP[130]=-2495.06	13, QP[150]=-1075.99
13, QP[91]=-2466.58	13, QP[111]=-2503.3	13, QP[131]=-2427.12	13, QP[151]=-1003.4
13, QP[92]=-2466.79	13, QP[112]=-2506.99	13, QP[132]=-2358.73	13, QP[152]=-930.804
13, QP[93]=-2467.17	13, QP[113]=-2510.86	13, QP[133]=-2289.9	13, QP[153]=-858.232
13, QP[94]=-2467.72	13, QP[114]=-2514.92	13, QP[134]=-2220.65	13, QP[154]=-785.696
13, QP[95]=-2468.43	13, QP[115]=-2519.17	13, QP[135]=-2151	13, QP[155]=-713.216
13, QP[96]=-2469.32	13, QP[116]=-2523.61	13, QP[136]=-2080.98	13, QP[156]=-640.809
13, QP[97]=-2470.37	13, QP[117]=-2528.24	13, QP[137]=-2010.6	13, QP[157]=-568.489
13, QP[98]=-2471.6	13, QP[118]=-2533.06	13, QP[138]=-1939.89	13, QP[158]=-496.275
13, QP[99]=-2472.99	13, QP[119]=-2538.09	13, QP[139]=-1868.88	13, QP[159]=-424.181
13, QP[100]=-2474.56	13, QP[120]=-2543.31	13, QP[140]=-1797.61	13, QP[160]=-352.221

13, QP[161]=-280.411	13, QP[181]=1069.97	13, QP[201]=1762.39	13, QP[221]=1983.66
13, QP[162]=-208.764	13, QP[182]=1110.93	13, QP[202]=1787.66	13, QP[222]=1977.32
13, QP[163]=-137.293	13, QP[183]=1151.29	13, QP[203]=1811.7	13, QP[223]=1969.32
13, QP[164]=-66.0111	13, QP[184]=1191.08	13, QP[204]=1834.44	13, QP[224]=1959.68
13, QP[165]=5.07034	13, QP[185]=1230.3	13, QP[205]=1855.84	13, QP[225]=1948.44
13, QP[166]=75.9397	13, QP[186]=1268.94	13, QP[206]=1875.84	13, QP[226]=1935.62
13, QP[167]=146.586	13, QP[187]=1307	13, QP[207]=1894.4	13, QP[227]=1921.26
13, QP[168]=217	13, QP[188]=1344.46	13, QP[208]=1911.47	13, QP[228]=1905.39
13, QP[169]=287.171	13, QP[189]=1381.3	13, QP[209]=1927.01	13, QP[229]=1888.06
13, QP[170]=357.09	13, QP[190]=1417.5	13, QP[210]=1940.97	13, QP[230]=1869.31
13, QP[171]=426.75	13, QP[191]=1453.03	13, QP[211]=1953.32	13, QP[231]=1849.19
13, QP[172]=496.141	13, QP[192]=1487.85	13, QP[212]=1964.03	13, QP[232]=1827.74
13, QP[173]=565.258	13, QP[193]=1521.93	13, QP[213]=1973.07	13, QP[233]=1805.02
13, QP[174]=634.093	13, QP[194]=1555.22	13, QP[214]=1980.42	13, QP[234]=1781.07
13, QP[175]=702.64	13, QP[195]=1587.69	13, QP[215]=1986.05	13, QP[235]=1755.95
13, QP[176]=770.894	13, QP[196]=1619.28	13, QP[216]=1989.96	13, QP[236]=1729.7
13, QP[177]=838.85	13, QP[197]=1649.96	13, QP[217]=1992.13	13, QP[237]=1702.39
13, QP[178]=906.503	13, QP[198]=1679.65	13, QP[218]=1992.57	13, QP[238]=1674.06
13, QP[179]=973.85	13, QP[199]=1708.33	13, QP[219]=1991.29	13, QP[239]=1644.75
13, QP[180]=1028.42	13, QP[200]=1735.92	13, QP[220]=1988.32	13, QP[240]=1614.52

13, QP[241]=1583.42	13, QP[249]=1307.7
13, QP[242]=1551.49	13, QP[250]=1270.3
13, QP[243]=1518.76	13, QP[251]=1232.3
13, QP[244]=1485.29	13, QP[252]=1193.7
13, QP[245]=1451.09	13, QP[253]=1154.49
13, QP[246]=1416.21	13, QP[254]=1114.67
13, QP[247]=1380.67	13, QP[255]=1074.21
13, QP[248]=1344.49	13, QP[256]=1033.07

Curriculum Vitae



Personal data :

Name	Surname	Date of Birth	Nationality	Gender	Marital status
Siavash	Dezfouli	21/02/1983	Iranian	Male	Married

Home Address
Sechshauser Straße 31, 1150 Wien, Top 548

Telephon	Mobile:	E-mail
+43- 1 2313714	+43-6764287020	siavash.dezfouli@gmail.com

Educational Background: (Last one first)

Certificate Degree	Field of Specialization	Name of Institution attended	Graduated date
PhD. candidate	Mechanical Engineering/ Mechatronics	<i>Vienna University of Technology (Institute of Mechanics and Mechatronics, Intelligent Handling Robotics and Devices, IHRT)</i>	05.12.2013
M.Sc.	Engineering Management	<i>Vienna University of Technology (CEC)</i>	05.12.2011
M.Sc.	Mechanical Engineering/ Mechatronics	<i>Sharif University of Technology (International Campus Kish Island)</i>	29.09.2009
B.Sc.	Mechanical Engineering (Heat & Fluid)	<i>Karaj Azad University</i>	22.07.2006

Title of –PhD. Thesis:

Motion Creating System Design and Implementation for a Biped Humanoid Robot

Title of –Graduate Thesis:

Mechatronics M.Sc.:

Structural Health Monitoring of Buried Pipelines using Piezoelectric Sensors

Job Experiences:

Place of Work	City	Date		Activity
		From	To	
Damavand Power Plant	Tehran	Summer 2004		co-supervisor in phase#2 in installation & commissioning of gas turbine Siemens V-94
FARAZARAB Co. (www.farazarab.co)	Tehran/Kish Island	May. 2008	June. 2010	Cooperation in design, manufacturing, installation and commissioning 4000 m3/day Multi-Effect Desalination plant
FARAZARAB Co.	Tehran/Kish Island			Cooperation in design and manufacturing 2x2400 m3/day Multi-Effect Desalination plant
FARAZARAB Co.	Tehran/Kish Island			Cooperation as documentation responsible engineer
FARAZARAB Co.	Tehran/Kish Island			Cooperation in project control management

Technical Skills:

- CATIA V5R20,
- ANSYS,
- MATLAB,
- SolidWorks,
- Auto CAD
- MS Project
- MS Office

Area of Interests:

- Gas/Oil industry (Process/Control Engineering)
- Water/Power industry (Process/Control Engineering)
- Smart Systems & Structures-Advanced Materials (Structural Health Monitoring)
- Advance Control-Mathematics Modeling & Finite Element Modeling
- MEMS-micro mechanic & processing
- Project Planning/Management

Language skills:

Title	Grade/level	Date	Name of Institution
IELTS	6	2006	English Language Testing Center Tehran
German	ZD (Zertifikat Deutsch)	2006	Teheran Goethe Institute
German	A1.1-A1.2-B1.1-B1.2	2012	WIFI (Vienna)

Persoanl Characterisitic:

- Strong team collaboration skills. Work closely with team members to achieve engineering goals.
- Willingness of travelling

Publications:

Dezfouli, S., & Mohamadi Daniali, M. (2012). motion controller design for a biped humanoid robot. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE* (pp. 1–7). Chicago, USA.

Dezfouli, S., & Kopacek, P. (2011). Mechatronic Design of a Humanoid Robot. In *Proceedings of the RAAD, 20th International Workshop on Robotics in Alpe-Adria-Danube Region* (pp. 1–6). Brno, Czech Republic.

Dezfouli, S., Kopacek, P., & Mohamadi Daniali, M., (2011). cost oriented humanoid robot archie. *International journal automation austria*, 19(2), 62-70.

Dezfouli S., Zabiholah A., (2010). Structural health monitoring of buried pipelines under static dislocation and vibration, In *6th ASME/IEEE International Conference on Mechatronics & Embedded Systems & Applications* (pp. 325-329). QingDao, China.

PhD Graduated Courses:

- Mechatronic system
- Autonomous, Mobile Robots
- Robot systems and handling devices
- Robot systems and handling devices (seminar course)
- Advanced Business Management and Culture
- Resource Efficiency

Mechatronics Master Graduated Courses:

- Optimum Design
- Micro-Nano Fluids Flow
- Neural Network Control
- Mechatronic Systems
- Advanced Mathematics
- Fuzzy Logic Control
- Robotics
- Electronics
- Advanced Modern Control
- Seminar Presentation
- Technical Writing

Reference:

Em. o.Univ.Prof. Dr.techn.Dr.hc.mult. Peter Kopacek

Technische Universität Wien, IHRT

Favoritenstrasse 9-11/E325A6

A - 1040 Wien

Tel: +43 1 58801 31800

FAX: +43 1 58801 31899

E-mail: peter.kopacek@tuwien.ac.at