TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

M A S T E R A R B E I T

MAMoRo

A Biologically Inspired **M**odular **A**utonomous **Mo**bile **Ro**bot Platform

Ausgeführt am
Institut für Handhabungsgeräte und Robotertechnik (E318)
der Technischen Universität Wien

unter der Anleitung von
o.Univ.Prof. Dipl.-Ing. Dr. Dr.h.c.mult. Peter Kopacek
und
Dipl.-Ing. Dr. Bernhard Putz

durch

Bakk.techn. Jidan Al-Eryani
Matr.Nr. 0025361
Brigittenauer Lände 224/1740, 1200 Wien

Wien, Juni 2007 ______________________

To My Parents

To Grendizer

To The Iron Man (Dinosaur War Aizenborg)

Abstract

This thesis deals with the design and implementation from scratch of a biologically inspired modular autonomous mobile robot platform or *MAMoRo* (acronym for: **M**odular **A**utonomous **Mo**bile **Ro**bot) and demonstrating its functionality with some practical applications. The emphasis is on the hardware of the robot platform.

MAMoRo possess the processing power to be used in intensive-processing applications such as graphic processing, and is at the same time low-cost. The main functions of MAMoRo are distributed into three modules: *Power & Motion module*, *Control module*, and *Intelligence module*. The decision-making functions of MAMoRo are distributed according to their complexity into two modules: the Control module, which is equipped with a low-cost microcontroller, and is responsible for handling low-level hardware functions; and the Intelligence module, which is equipped with a low-cost Field Programmable Gate Array (FPGA), and handles high-level and processing-intensive functions. This model of distribution of functions according to their complexity, and particularly using an FPGA – a volatile programmable hardware unit – for handling high-level functions was inspired from the anatomy of the human brain, and brings with it many advantages. In addition, the combination of a microcontroller and an FPGA in the same system enhances considerably the flexibility and makes hardware re-configuration at run-time possible. Furthermore, the whole system is programmable through a single USB interface.

After implementing MAMoRo, its hardware basic functions were tested. Afterwards, the overall functionality of MAMoRo was demonstrated with some practical applications. The project was successfully completed, and all objectives stated at the beginning of the thesis were accomplished.

Keywords: Robot Platforms, Evaluation Platforms, Autonomous Robots, Mobile Robots, Robots in Education, Robots in Entertainment, Reconfigurable Hardware, Biologically Inspired Robots

Kurzfassung

Diese Arbeit beschäftigt sich mit dem Entwurf und der Implementierung einer biologisch inspirierten, modularen, autonomen, mobilen Roboterplattform oder MAMoRo (engl. Abk.: **M**odular **A**utonomous **Mo**bile **Ro**bot) und der Demonstration seiner Funktionalität anhand einiger praktischer Anwendungen, wobei der Schwerpunkt auf der Hardware der Roboterplattform liegt.

MAMoRo besitzt die Rechenleistung um in rechenintensiven Anwendungen genutzt zu werden, wie zum Beispiel in der Bildverarbeitung, und ist gleichzeitig kostengünstig. Die Hauptfunktionen von MAMoRo werden in drei Module unterteilt: *Power & Motion Modul*, *Control Modul*, und *Intelligence Modul*. Die Entscheidungsfunktionen von MAMoRo wurden abhängig von ihrer Komplexität in zwei Module unterteilt: das Control Modul, ausgestattet mit einem kostengünstigen Mikrocontroller, ist für die Abhandlungen systemnaher Funktionen verantwortlich; und das Intelligence Modul, ausgestattet mit einem kostengünstigen Field Programmable Gate Array (FPGA), ist für die Abhandlungen von höhere und rechenintensiven Funktionen verantwortlich. Das Modell der Aufteilung der Funktionen anhand ihrer Komplexität und die Benutzung eines FPGA – ein flüchtiger programmierbarer Baustein – für die Abhandlungen von höhere Funktionen, ähnelt die Anatomie des menschlichen Gehirnes nach und bringt viele Vorteile mit sich. Des Weiteren steigert die Zusammenführung von einem Mikrocontroller und eines FPGA in einem System die Flexibilität erheblich und ermöglicht die Rekonfiguration von Hardware während der Laufzeit. Das ganze System ist über eine einzelne USB Schnittstelle programmierbar.

Nach der Implementierung von MAMoRo wurden seine Hardware Hauptfunktionen überprüft. Anschließend wurde die allgemeine Funktionalität von MAMoRo anhand einiger praktischer Anwendungen demonstriert. Das Projekt wurde erfolgreich abgeschlossen und alle am Anfang dieses Projekt genannten Ziele wurden erreicht.

Stichwörter: Roboterplattformen, Testplattformen, autonome Roboter, mobile Roboter, Roboter in der Lehre, Roboter in der Unterhaltung, Reconfigurable Hardware, biologisch inspirierte Roboter.

Acknowledgments

Robots have fascinated me since my childhood when I used to watch Japanese mecha anime series. It was because of my admiration of robots that I choose my current field study electrical engineering. Robotics is a challenging field; it is an interdisciplinary field involving electrical engineering, computer science, mechanics, physics, and even biology. After attending a laboratory course, I wanted to buy a robot platform for myself to try some experiments at home. Only then did I notice that the robot market lacks low-cost (below € 400) robots platforms with sufficient processing power. Afterwards, I decided to make solving this problem the theme of my master thesis, and came with the idea of developing a biologically inspired robot platform, MAMoRo.

First of all, I want to thank Dr. Bernhard Putz and Prof. Peter Kopacek for recognizing the potential of the idea and allowing me to pursuit it; I also want to thank them for the immense flexibility they showed me during the course of this thesis.

I want to thank sincerely my supervisor Dr. Bernhard Putz for the endless support and advice I got from him during the course of this thesis, and for having confidence in my abilities.
I want to thank Edmund Schierer for sharing his expertise with me and helping me out in many cases where it seemed hopeless. I also want to thank Dr. Man-Wook Han for his help in many things, and the talented technician Peter Unterkreuter for his help in the construction of the mechanical chassis for MAMoRo.

Finally, I want to thank in general all staff of the Institute of Handling Devices and Robotics (IHRT) for creating a friendly and inspiring working environment.

Thank you all !

Contents

Abbreviations

#	
µC	Microcontroller
A	
A/D	Analogue/Digital
ADC	Analogue-Digital Converter
AI	Artificial intelligence
AMR	Autonomous Mobile Robot
B	
BOD	Brown-out Detection
C	
CLB	Configurable Logic Blocks
D	
DCI	Digitally Controlled Impedance
DCM	Digital Clock Manager
DSP	Digital Signal Processor
E	
EEPROM	Electrically Erasable Programmable Read Only Memory
EMI	Electromagnetic Interference
F	
FIRA	Federation of International Robot-soccer Association
FPGA	Field-programmable Gate Array
G	
GCC	GNU Compiler Collection

I

IC	Integrated Circuit
IHRT	Institute of Handling Devices and Robotics, Vienna University of Technology
IOB	Input/Output Blocks
ISP	In-System Programmable
IFR	International Federation of Robotics
ISR	Interrupt Service Routine

K

Kibit	Kilo Binary bit
KiByte	Kilo Binary Byte

L

LSB	Least Significant Bit
LUT	Look-Up Tables

M

MCU	Microcontroller Unit
MIPS	Million Instructions per Second

N

NC	Not Connected

O

Opcode	Operation Code

P

PCB	Printed Circuit Board
PWM	Pulse-Width Modulation
PWR	Power

R

RAM	Random Access Memory
S	
SRAM	Static Random Access Memory
SPI	Serial Peripheral Interface bus
T	
TQ144	144-pin Thin Quad Flat Pack
TU-Vienna	Vienna University of Technology
TWI	Two-wire Serial Interface
U	
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal Synchronous Asynchronous Receiver/Transmitter

List of Figures

List of Tables

1 Introduction

Robots have been slowly conquering our daily life since they were first invented. Their use has increased over the last years due to the fast progress of technology, and will even increase more in the future [IFR06]. Robots are used in a wide variety of fields.

They are widely used in the industry sector, e.g. automotive industry, due their accuracy, productivity, and endurance. In medicine, medical robots are being developed that can perform very accurate surgery where the hand shaking of surgeons is a problem, or to allow a surgeon to perform an operation on a patient who is remotely located. Robots can also be used to do too dangerous work for humans such as removing landmines and defusing bombs, or accomplish work in environments too dangerous for humans such as in space, or even got to unreachable or far distant places to humans such as planet Mars. Domestically, robots can be used for simple but unwanted tasks such as vacuum and floor cleaning, or lawn mowing. There are also robot enthusiasts who have fun building and programming robots and using them for entertainment purposes.

A hot field where there is currently a lot of research going on is the development of biologically inspired robots. This is a field in which the nature is studied and tried to be copied to solve or improve solutions to problems, and develop efficient robots. Another field where there is there currently a lot of research going on is the development of humanoid robots. Besides wanting them to do what we humans can do and even better, they are also used in many research fields such as understanding human body movements, artificial intelligence, or for developing models for various topics in cognitive science (the science of acquisition and use of knowledge).

Due to the wide range of applications where robots can be used and their potential for the future, many universities and research institutions are investing a lot of money and time on research in the field of robotics. Most of the universities have recognized the importance of teaching and preparing their students for this field, and are trying to attract many students to this field through interesting courses in their curriculum.

For teaching and research, general-purpose robot platforms, commercially purchased or developed internally, are generally used for conducting a wide range of experiments on robotics, and for rapid prototyping. They are also used for conducting experiments in many other fields such as in electrical engineering, computer science. Therefore, having an optimal robot platform that is flexible, cost effective, and most importantly has sufficient processing power is very essential. The objective of this thesis is to develop such a robot platform.

1.1 Problem Description

Most universities and other academic institutions use low-cost and commercially available robot platforms to introduce their students to robotics and to perform on them simple robot applications [ZBZ06] [GHW03]. These robot platforms are also the type usually purchased by private robot enthusiasts due to their low cost, ease of use, and availability. The main problem with these robot platforms is that they cannot be used for advanced robot applications, because they do not fulfil requirements such as [GHW03] [BHL04]:

- Processing power – needed for processing-intensive applications such as running algorithms in the field of machine learning, and image recognition.
- Flexibility
- Extendibility
- Power consumption – important for mobile robots, since they usually get their power supply from batteries.

Because of these restrictions, low-cost robot platforms are mostly only used for robotic introductory courses and in simple robotic applications by the academia, and by robotic enthusiasts who can afford them because of their low cost. A popular low-cost and commercially available robot platform that is widely used in the academia, including TU-Vienna (Vienna University of Technology) is the LEGO™ Mindstorms [ZBZ06] [GHW03]. For more advanced robotics applications, a different class of robot platforms are used. There are not many types of these platforms available commercially, or at least they are not easily available. This is probably due to the high cost connected with purchasing them, they often do not fulfil the specific requirements wanted, and because universities and other academic institutions usually want to gain new experiences and knowledge by designing and building their own robot platform. However, developing a new robot platform costs a lot of time and money, and besides the actual designing and implementing, the documentation has to be created , bugs and upgrades have to be continuously be made to it. Therefore, developing your own robot platform is an interesting choice when it is going to be reproduced in a large quantity, or if new experiences and know-how should be gained.

1.2 Thesis Objectives

The aim of this thesis is the design and implementation from scratch of a biologically inspired modular autonomous mobile robot platform or *MAMoRo* (acronym for: Modular Autonomous Mobile Robot), and then evaluating its hardware and demonstrating its functionality with some practical applications. The following requirements are to be fulfilled by MAMoRo:

- Low-cost – in the same price range as other low-cost robot platforms, and should not exceed a lot the price of the popular robot platform LEGO Mindstorms. This will make it affordable by private robot enthusiasts, while in the academia it would be more affordable to provide it in large numbers to their students.
- Processing power – enough for running algorithms in the field of machine learning, and image processing.
- Ease of use – makes it also ideal for newcomers to robotics and robotics introductory courses in the academia.
- Flexibility and versatility – use in a wide range of experiments in robotics, computer science, and in electrical engineering in general.
- Modularity – the robot platform main functions should be distributed into more than one module, which are then placed vertically over each other in a stacked-up form. These individual modules should also be usable as stand-alone.
- Expandability – it should be easy to swap parts or expand the platform functionality by adding additional modules
- Low power consumption – important for mobile robots that run on batteries.

Achieving all these goals might seem far-fetched, but with some innovative measures, and with the right system design and architecture they are achievable. The sytem design and architecture of MAMoRo are described in chapter *3 System Design and Architecture*.

Most commercial robot platforms (e.g. LEGO Mindstorms[1], Khepera[2]) were actually developed at universities then marketed commercially. One of the far-fetched objectives of this thesis is to commercially market MAMoRo.

1.3 Overview of Thesis

Chapter 1 – **Introduction**

Describes the problem and the objectives of this thesis.

Chapter 2 – State of the Art

Looks at related robot platforms.

Chapter 3 – System Design and Architecture

Introduces and describes the overall design concept and architecture of MAMoRo without going down to the technical hardware information.

Chapter 4 – Power & Motion Module

Describes in detail the hardware design of the Power & Motion module.

Chapter 5 – Control Module

Describes in detail the hardware design of the Control module.

Chapter 6 – Intelligence Module

Describes in detail the hardware design of the Intelligence module.

Chapter 7 – Hardware Implementation of MAMoRo

Deals with the hardware implementation of MAMoRo

Chapter 8 – Programming MAMoRo

Describes the programming steps of MAMoRo.

Chapter 9 – Application Examples

Demonstrates the functionality of MAMoRo with two application examples

Chapter 10 – Conclusion and Future Work

The results and future work are discussed here.

[1] LEGO Mindstorms homepage: http://mindstorms.lego.com
[2] Khepera's homepage: http://www.k-team.com

Appendix A - Schematics, Board Layouts, and Parts List

Schematics, board layouts, and parts lists for all boards used in this project.

Appendix B - Headers Pinouts

Pinouts description for all headers on MAMoRo's modules.

Appendix C - CD-ROM

Lists the contents of the CD-ROM attached with this thesis.

2 State of the Art

In this chapter, four different robot platforms are going to be objectively reviewed and compared, then their main advantages and disadvantages will be listed. The chapter then ends with a conclusion of all robot platforms reviewed. The robot platforms picked for this review have similar or overlapping goals as the robot platform MAMoRo.

2.1 LEGO Mindstorms NXT

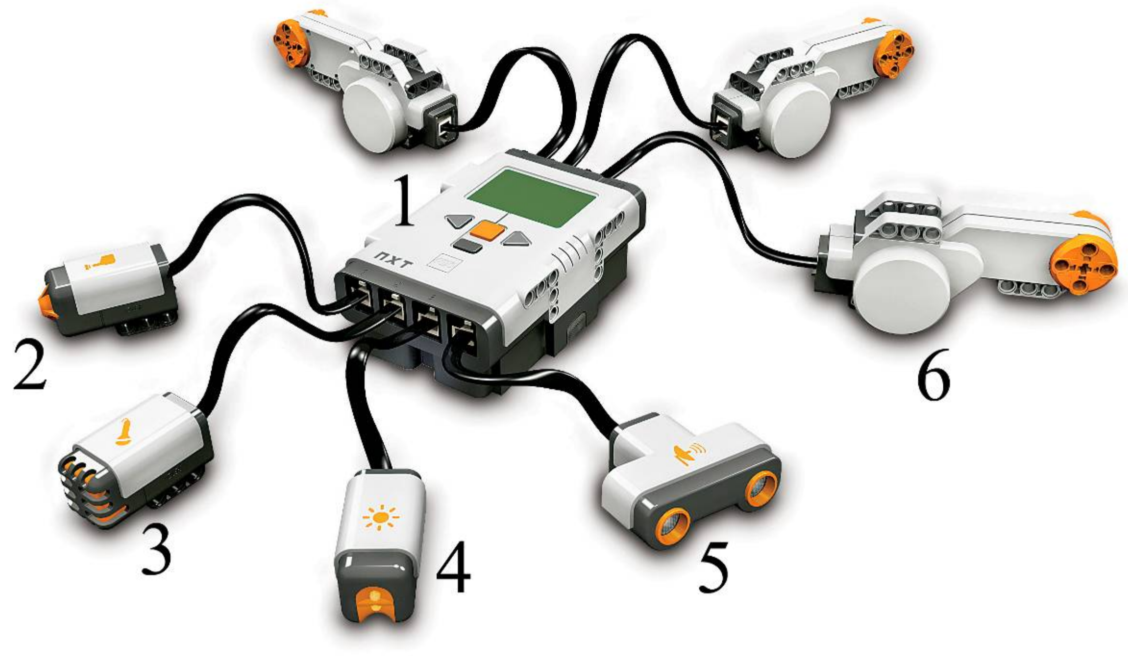


Figure 2.1: Mindstorms NXT kit [LEG06a]

2.1.1 Overview

LEGO Mindstorms NXT is a robotic kit released by LEGO™ in the late of 2006. It is a continuation of the previous LEGO Mindstorms series that first came out in 1998, which were originally developed through a partnership with the Massachusetts Institute of Technology.
The LEGO Mindstorms NXT kit comes with theses parts: (see Figure 2.1):

- NXT brick: the programmable processing unit **(1)**
- Sensors, consisting of:
    - Touch sensor **(2)**
    - Sound sensor **(3)**
    - Light sensor **(4)**
    - Ultrasonic sensor **(5)**
- 3 servo motors; each with a built-in rotation sensor. **(6)**

The sensors, actuators, and the mechanical parts are not the focus of this review, but rather the NXT brick – the programmable unit of this kit.

NXT brick hardware specifications**:**

- Main processor: AT91SAM7S256; Atmel™ 32-bit ARM™
    - 256 KiByte Flash, 64 KiByte RAM
    - 48 MHz
- Co-processor: ATmega48; Atmel™ 8-bit AVR processor
    - 4 KiByte FLASH, 512 Byte RAM
    - 8 MHz
- Communication interfaces:
    - USB 2.0
    - wireless Bluetooth
- User I/O
    - 4 input ports
    - 3 output ports
- Peripherals:
    - 26 x 40.6 mm LCD
    - Loudspeaker
    - 4 button user interface
- Powered by 6 AA batteries (6 x 1.5 V = 9 V)

2.1.2 System Architecture

The hardware architecture of the NXT brick is depicted in Figure 2.2.

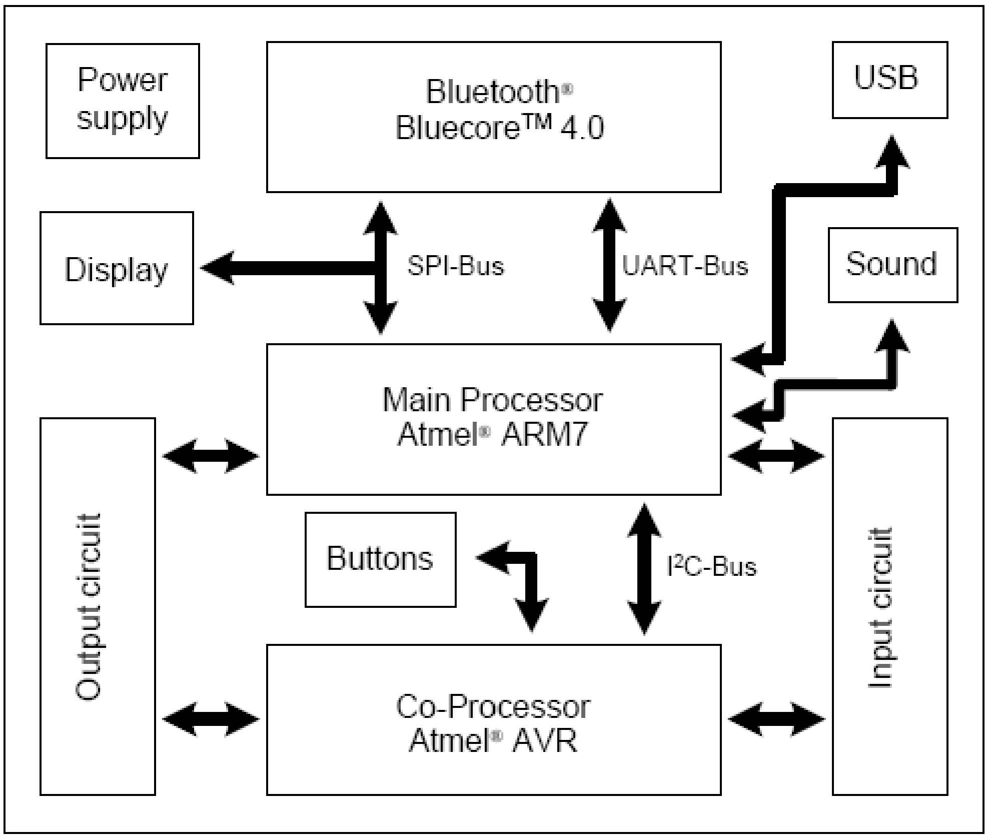


Figure 2.2: NXT brick architecture [LEG06b]

Two µCs are implemented in the NXT brick:

- *AT91SAM7S256*: this µC is the main processor of the system, and operates at 48 MHz (43 MIPS). Its main task is handling user-specific tasks.
- *ATmega4)*: this µC acts as a helper for the main processor. It has a lower performance then the main processor and operates at 8 MHz (8 MIPS). Its main task is to handle lower-level tasks such as power management, A/D conversions, and generating the PWM signals for the motors.

The main and co-processor communicate with each other through an I²C bus. The I²C is a master/slave bus, and the main processor is the Master. The maximum data transfer rate between the two processors is 380 kbit/s. The electronics of the NXT brick is enclosed in a chase, and is accessed externally through three output ports and four input ports.

Output Ports

The NXT brick has three output ports for controlling the actuators. Each Port has six pins:

- 2 x PWM output pins (max 700 mA for both pins)

- 2 x Power pins (GND; +4.3 V; max. deliverable current by this supply is 180 mA)
- 2 x Input pins

The PWM output pins are connected to a motor driver that can supply max. 700 mA. These pins are used to control an actuator. The power supply connected to the power pin can supply max. 180 mA in total. In this output port, two pins are used as input; they are connected to the ARM μC through a Schmitt-trigger to filter out noise. The input pins were added to the output ports to allow, for example, a DC motor with two inputs for speed control and 2 outputs from a quadrature encoder (integrated in the motor itself) to be directly connected to this port.

Input Ports

The NXT brick has four input ports; each port has six pins:

- 1 x ADC input or Power pin (for backward compatibility with some sensors)
- 3 x Power pins (2 x GND, +4.3 V)
- 2 x I/O pins

The two I/O pins are connected to an I^2C bus; one of the two I/O pins can be used as an input to an ADC.

The NXT brick relies on the I^2C to connect external digital sensors. LEGO offers three types of sensors:

- *Passive sensors*: do not need special power/timing requirements, e.g. touch and light sensor.
- *Active sensors*: need special power/timing requirements, e.g. rotation sensor.
- *Digital sensors*: include an internal microcontroller and communicate with I^2C, e.g. ultrasonic sensor.

More can be utilized from the NXT brick (e.g. access to the JTAG) if the case was screwed open. However, if this is done all warranties to this product become invalid. A detailed review of the LEGO Mindstorms NXT can be read in [FO06]. The schematics of the NXT brick are open for everyone to see in [LEG06b].

2.1.3 Conclusion

Advantages

- Easy to use
- Commercially widely available
- Flexible mechanical construction through the LEGO-Technic blocks.
- Wide support through the large community using it, spanning from academia to robot enthusiasts.
- Availability of documentation and software libraries (e.g. for 3D simulation, or control plug-ins for popular control programs such as LabView and Matlab).
- Relative low price: € 233 (tax free, by the toy supplier *Toys "R" Us* ; [24/5/2007])

Disadvantages

- The Processing power (~52 MIPS [ATM06c] [ATM05c]) of the NXT is not sufficient for robotic applications requiring intensive processing such as vision processing.
- Too few user I/O pins
- Hardware is inflexible and too difficult to expand
- The 4.3 V power supply provided to the output and input ports is not a common voltage level. Most electronic devices require 5 V or 3.3 V. Also, the maximum allowable 180 mA current draw is very small.
- The I^2C bus used for the communication between the main and co-processor is very slow.
- Instability of the LEGO-Technic blocks.

LEGO Mindstroms is the most widely used robot platform for academia and entertainment [ZBZ06]. This platform is ideal for simple robotic applications and is cheap, easy to use, and has a wide support of users spanning from academia to entertainment. The LEGO bricks allow the mechanical part of the robot to be modelled freely and easily. Nevertheless, it simply lacks the processing power, flexibility, and expandability to be used in advanced robotic experiments.

2.2 RCUBE

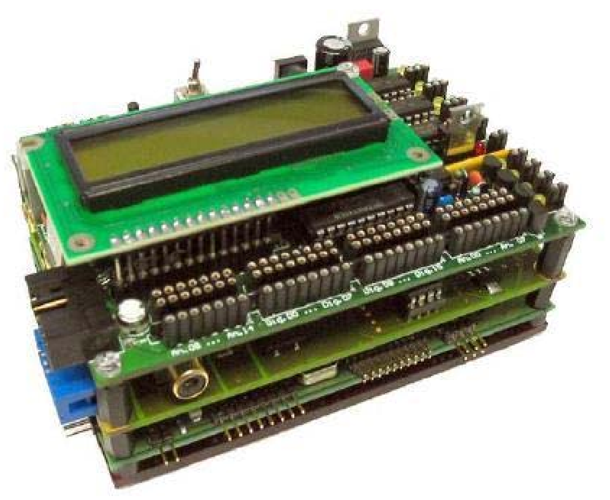


Figure 2.3: RCUBE modules [RCU05]

2.2.1 Overview

RCUBE is a platform for autonomous intelligent systems, developed by the University of Applied Sciences in Brandenburg, Germany. It is commercially available and is intended to be used as a research and education platform for industrial applications and private developers. RCUBE is a hardware platform, and does not come with a mechanical chassis.
The three basic modules of the platform are:

- CPU board
- VIO board – video I/O module
- AKSEN board – actuator/sensor module

All modules are designed to work as standalone and can be connected with each other via a CAN bus. They all have the same size and are stackable.

2.2.2 System Architecture

The three basic modules of the platform are going to be described.

CPU Board

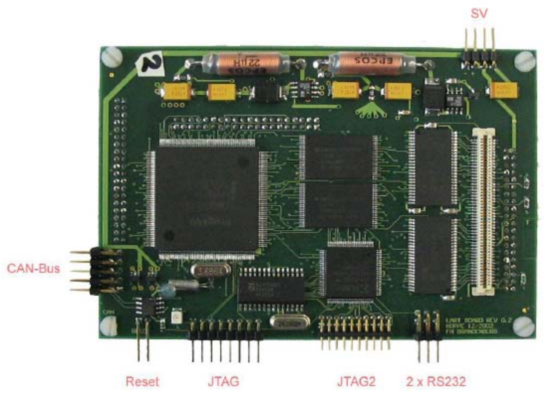


Figure 2.4: CPU board in RCUBE [RCU06]

The CPU board is a slightly varied LART[1] Board. The board contains an embedded system capable of running Linux, and is programmable with GCC. This module contains the processing unit and is the brain of the system.

Hardware specifications:

- 220 MHz Digital SA-1100 StrongARM CPU
- 32 MiByte EDO RAM
- 8 MiByte Intel Fast boot block Flash memory
- CAN Bus interface
- JTAG interface
- 2 x RS232 serial ports

VIO Board

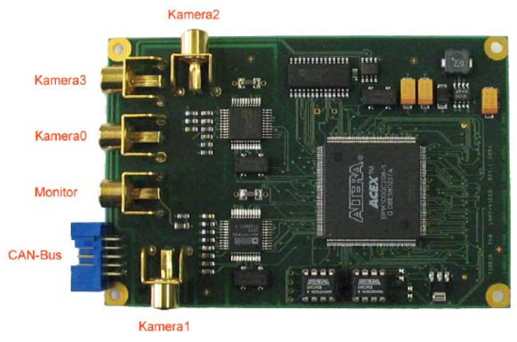


Figure 2.5: VIO board in RCUBE [RCU06]

[1] The LART (Linux Advanced Radio Terminal) was developed by Delft University of Technology (Netherlands). Its Schematics and documentation can be found in [LAR06].

The VIA board is a standalone module for image digitizing and recognition based on the StrongARM processor and Linux.

Hardware specifications:

- CAN bus connector for connecting it with other modules.
- 4 x Video input for connecting PAL video sources, e.g. CMOS cameras.
- Video output for connecting a display device, e.g. TV or a PC graphic with TV-IN.

AKSEN Board

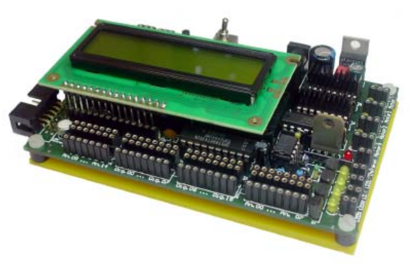


Figure 2.6: AKSEN board in RCUBE [BHL04]

The AKSEN board provides connections for sensors and actuators. It acts as standalone module and is directly programmable by GCC.

Hardware specifications:

- 15 analogue inputs and 16 digital I/O
- 4 motor drivers for DC-motors up to 1A
- 4 power drivers (e.g. for bulbs infrared senders)
- 3 servo outputs
- 1 output for modulated infrared
- 3 encoder inputs for odometry
- 4 dipswitches
- RS232 interface
- CAN interface (optional)
- LCD display (optional)
- Bluetooth interface (optional)

More about RCUBE can be read in the homepage of RCUBE [RCU06].

2.2.3 Conclusion

Advantages

- High processing power
- Integrated operating system allows easy programming and debugging of complex algorithms

Disadvantages

- System is too complex
- Not power efficient
- Cost[1]: VIO: €1848; CPU: €1428; AKSEN: €200–250 (all prices are tax free; 8/1/2007)

This platform has the processing power and expandability to be used in many advanced applications requiring intensive processing. The Linux operating environment makes the programming of high-level abstract algorithms and debugging more comfortable.
Downside is the complexity of the system that makes it not suitable for simple robotic applications. The overall system is not power efficient, which makes it not suitable to be used in mobile robots. The major drawback, however, is its high cost.

[1] These prices were sent to me through E-Mail by Ingo Boersch, one of the developers of RCUBE.

2.3 Tinyphoon

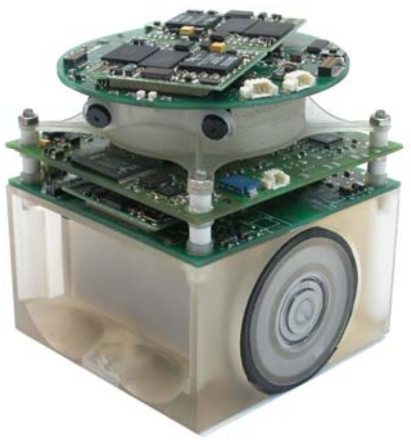


Figure 2.7: Tinyphoon [NM05]

2.3.1 Overview

Tinyphoon is an autonomous mobile robot, developed (and still being developed) by the Institute for Computer Technology (ICT) at TU-Vienna [NM05]. The robot is used currently for teaching and research in ICT. The whole robot fits into a 7.5 cm³ cube and was designed to play in the FIRA (Federation of International Robot-soccer Association) MiroSot[1] league. The developers of Tinyphoon claim that Tinyphoon is: "*The worlds most powerful and smallest soccer playing robot*" [TiP06]. Tinyphoon is a complete robot platform consisting of a mechanical chassis and its electronics. The electronic part consists of a *motion board*, and a *vision board*

.

2.3.2 System Architecture

Motion Board

The main components of the motion board are a power supply, microcontroller module, DSP module; the module is also equipped with various sensors.

[1] Micro Robot World Cup Soccer Tournament (MiroSot): is a category in robot soccer, where two teams, each consisting of three robots, play against each other. Each robot must fit into a 7.5 cm³ cube, and shall be able to communicate wirelessly to a host computer for vision and location processing. More details can be found in FIRA's website: http://www.fira.net.

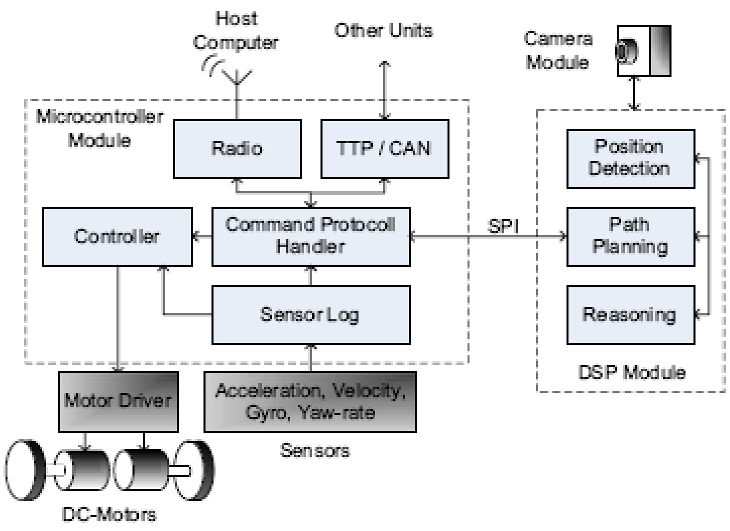


Figure 2.8: Motion board architecture in Tinyphoon [NM05]

The microcontroller module is based on a XC167 microcontroller from Infineon™. Its main function is to control the motor drivers, read sensors, and communicate with other modules including Bluetooth wireless communication with the host computer or other robots. The DSP module is based on a BF533 or BF561 Blackfin family DSP from Analog Devices™, and functions as a coprocessor. This module contains the main computational power of the robot. Its basic functions are path planning, vision processing, and reasoning. An SPI bus connects the microcontroller module with the DSP module.

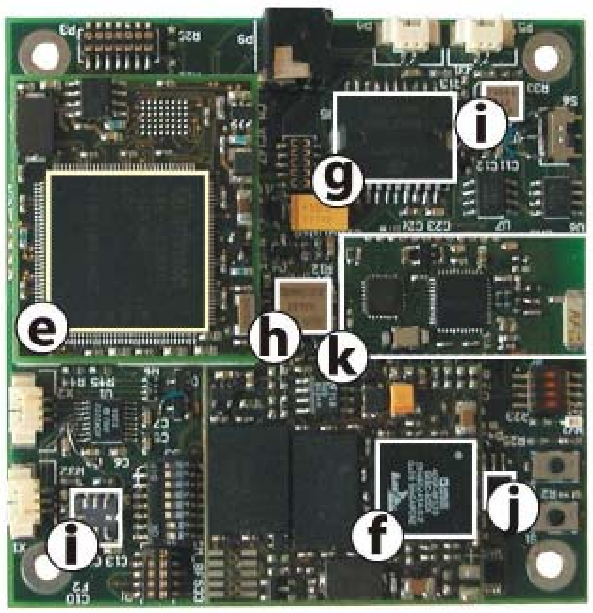


Figure 2.9: Motion board: (e) Infineon XC, (f) Analog Devices Blackfin DSP, (g) Motor Driver Unit, (h) Gyro Sensor, (i) 2x Acceleration Sensors, (i) Magnetic Field Sensor, (k) 2.4 GHz Radio Module [NRB06]

Vision Unit

The vision board enhances the robot with stereo vision capabilities. The board is equipped two Blackfin DSPs modules. Two 320 x 240 pixels CMOS cameras are connected to the board. All modules are can be connected by a CAN or TTP bus to support real-time communications.

2.3.3 Conclusion

Advantages

- Sufficient processing power
- Small size

Disadvantages

- Not general-purpose robot platform; it is more specialized for robot soccer
- Cost: not available[1]; alone the DSP module with BF533 core costs 165 €, while with the better BF561 core it costs € 195 (all prices are tax free).
- Expensive and difficult to upgrade and expand

Tinyphoon is more a specialized robot for soccer than a general-purpose one. Its small size makes it expensive, and very difficult to expand and upgrade.

2.4 Khepera-III

[1] After many E-Mails and weeks of waiting, no response was received regarding the price from [TiP06].

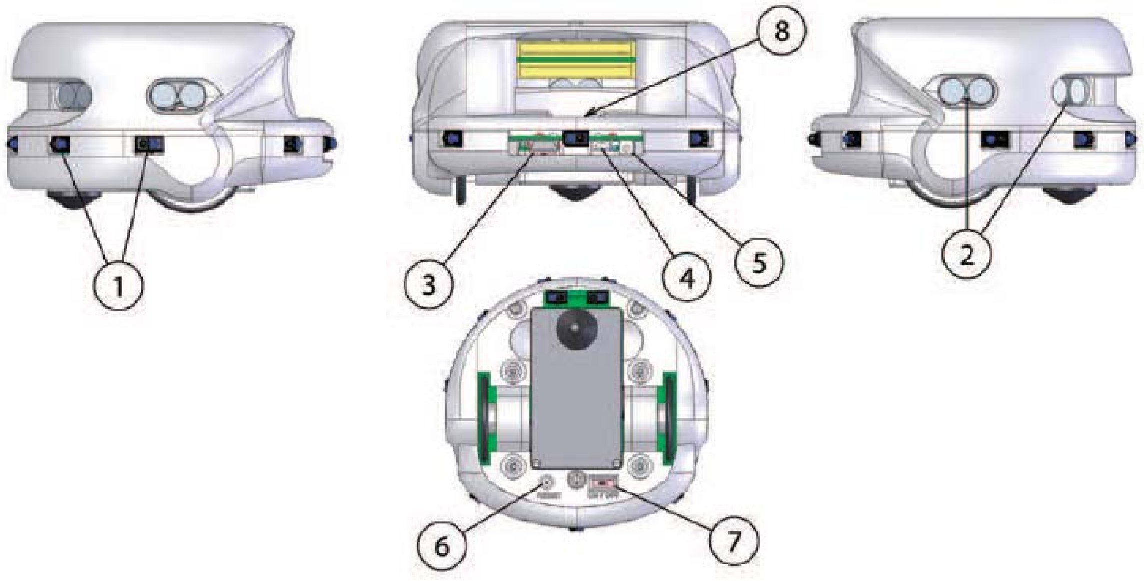


Figure 2.10: Khepera-III as seen from different view angles [KTm07a]

2.4.1 Overview

Khepera III is a robot platform used in many universities for teaching and research, including the Institute of Handling Devices and Robotics (IHRT). The first version was developed in 1991 at the Swiss Federal Institute of Technology of Lausanne (EPFL). The newest version in this series, which will review here, is the Khepera-III, which is developed and produced commercially by K-Team™.

2.4.2 System Architecture

Unfortunately, the circuit schematics for Khepera-III and all its extension modules are not open. Considering the high cost of this robot, this is a big drawback.

Khepera-III hardware specification:

- Processor: DsPIC 30F5011, 60MHz; 4 KiByte RAM, 66 KiByte Flash
- 2 x DC brushed servo motors with incremental encoders
- Sensors:
  - 8 x Infra-red sensors

    - 2 x Infra-red sensors placed at the bottom of the chassis for line following applications
    - 5 x Ultrasonic sensors
- Battery pack: Lithium-Polymer battery pack (1400 mAh)
- Communication: standard serial port
- Size: diameter: 130 mm, height: 70 mm
- Weight: 690 g

Figure 2.10 illustrates the Khepera-III robot platform from different view angles. The robot with its current hardware specifications is not sufficient for applications requiring intensive processing. K-Team recognized this and provided an expansion module: *KoreBot*. There is also a light version for this expansion module: *KoreBot LE* (Figure 2.11), which is cheaper and differs little from the full version.

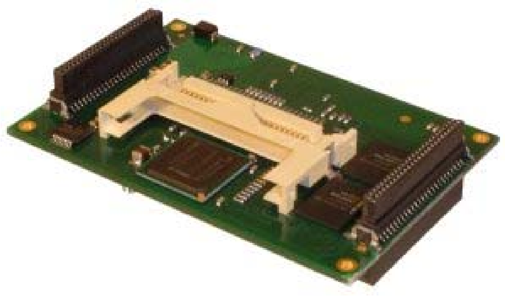


Figure 2.11: KoreBot LE expansion module for Khepera-III [KTm07b]

KoreBot LE hardware specification:

- Processor: Intel XSCALE PXA-255 400MHz
- RAM: 64 MiByte
- Flash: 32 MiByte
- Interfaces:
    - 3x Serial RS332 (including Bluetooth compatible port)
    - 1x USB Client port
    - 3x USB Master port (to connect USB camera)
    - 1x MMC controller (MutiMedia Card)
    - 1x LCD controller
    - 1x I2C bus (400kb/s)
    - 1x SSP/SPI bus (1.8Mb/s)

    - 1x AC97 sound controller
    - 2 x PWM pins
    - 53 x user-I/O pins (if not used for above features)
- OS support: Linux

The KoreBot is plugged into the Khepera-III as illustrated in Figure 2.12.

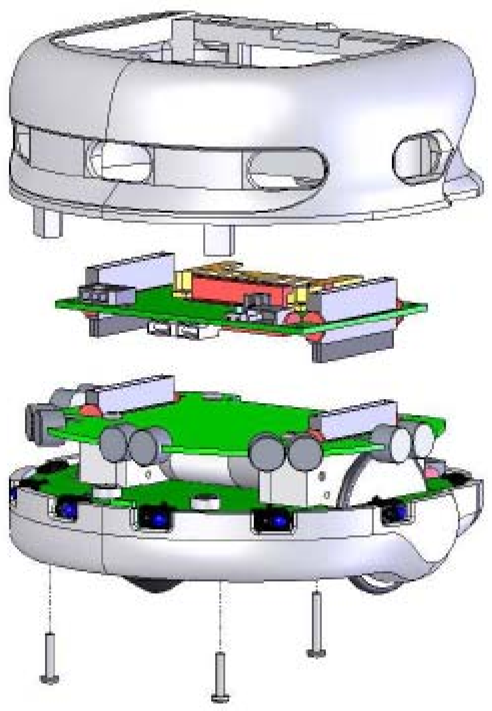


Figure 2.12: Khepera-III and KoreBot (on top) placed in the chassis [KTm07a]

Optionally, the following expansion modules can also be purchased:

- KoreMotor (drive up 4 DC motors)
- KoreConnect (easier connection to RS232 ports and USB Client).
- KoreJTAG (JTAG to USB interface)
- KoreSound (audio input and output).
- KoreIO (Analog I/O, digtal out, servo control)

The robot can be simulated in a PC with the 3D simulator *Webots*™[1]. Furthermore, K-Team offers control plug-ins for popular programs such as Matlab and LabView

[1] Webots homepage: http://www.cyberbotics.com/

2.4.3 Conclusion

Khepera-III is worth considering if it is going to be used with the expansion module KoreBot. The hardware performance of Khepera-III alone can be matched by many other commercial robots with much lower cost. Furthmore, Khepera-III is programmable through the obsolete standard serial interface not found in many of today's PC's and especially Notebooks.
The listed advantages and disadvantages below are true when Khepera-III and KoreBot LE are used.

Advantages: (Khepera-III + KoreBot LE)

- Sufficient processing power for image processing
- Wide community in teaching and research
- Availability of optional expansion modules
- Embedded Linux
- Stable chassis
- Availability of software library to allow 3D simulation of this robot, and control plug-ins for popular control programs such as LabView and Matlab.

Disadvantages: (Khepera-III + KoreBot LE)

- Schematics are not open
- Cost[1]: Khepera-III: ~ €1820 (2450 USD); KorBotLE: ~ €220 (299 USD) (all prices are tax free,)

Khepera-III combined with KoreBot LE make a nice robot platform for teaching and research. Nevertheless, its price is too high compared to what it offers.

2.5 Final Conclusion

Although, only four robot platforms were reviewed in this chapter, in reality many other robot platforms were reviewed from the internet and research papers before coming with the idea to

[1] These prices were sent to me through E-Mail by Laurence Schneider from K-Team

develop MAMoRo. None of the available robot platforms could fulfil the major requirements such as having sufficient processing power, low-cost, and be easy to handle.

From all the robot platforms reviewed, LEGO Mindstorms and Khepera-III+KoreBot were the nearest to achieving the needed goals. LEGO Mindstorms has the best price/performance ratio, and thus, it is not a surprise that it is the most widely used robot in academia and entertainment [ZBZ06]. Nevertheless, its insufficient processor power, inflexibility and difficult expandability makes it not good enough for advanced robotics applications. Khepera-III+KoreBot, on the other hand, although with a much higher processing power than the LEGO Mindstorms, the high cost and worse price/performance ratio are a major drawback.

3 System Design and Architecture

In this chapter, the design, specification, and architecture of MAMoRo are going to be laid down, and how it will manages to solve the problems mentioned in *1.1 Problem Description*, and meets the requirements listed in *1.2 Thesis Objectives*. Since MAMoRo is not going be restricted to certain components from certain vendors, only the requirements for the hardware components and the functions they are supposed to do are described, but not their hardware realization. The detailed hardware realization and implementation of MAMoRo are described in later chapters.

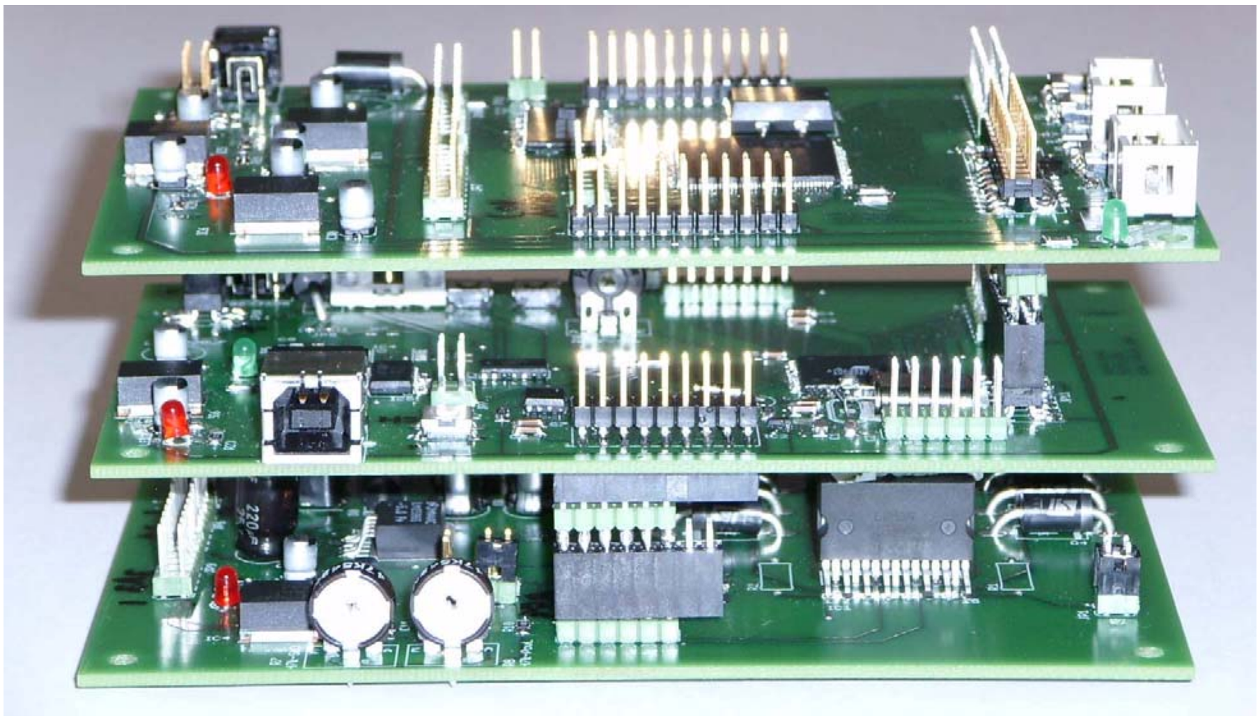


Figure 3.1: *MAMoRo* - The modular autonomous mobile robot platform after its implementation

3.1 System Overview

An autonomous mobile robot (AMR) consists principally of four basic function units (Figure 3.2):

- Power unit: provides power to the robot.
- Motion unit: drives the actuators of the robot.

- Sensor unit: pre-processes the raw signals from sensors before they are send to the decision unit.
- Decision unit: is the "brain" of the robot; it makes decisions from the data read from the sensor unit, and accordingly sends control signals to the motion unit to drive the actuators.

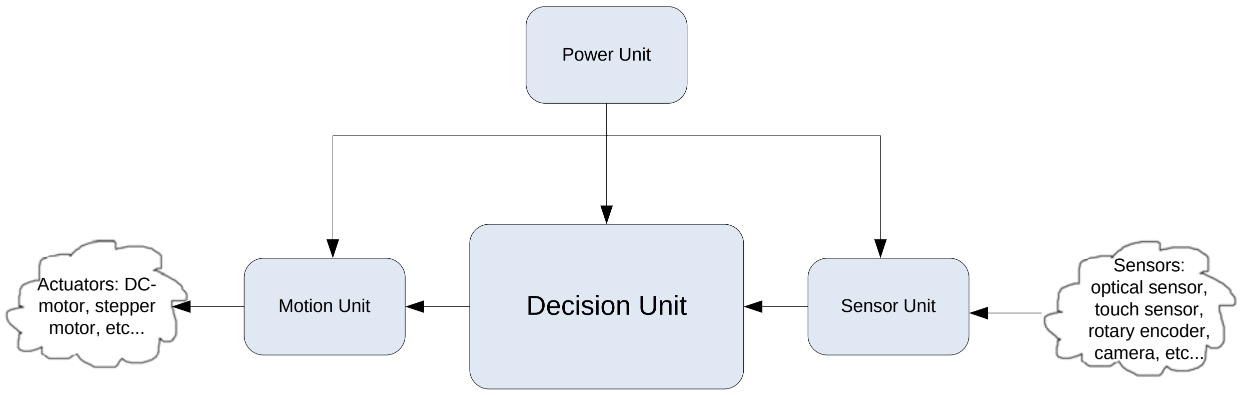

Power Unit
Actuators: DC-motor, stepper motor, etc...
Motion Unit
Decision Unit
Sensor Unit
Sensors: optical sensor, touch sensor, rotary encoder, camera, etc...

Figure 3.2: Basic functions of an autonomous mobile robot

To enhance flexibility and expandability, many robot platforms implement each of the units listed above in one or more hardware modules. This, however, increases the cost and complexity of the whole platform system, and makes it harder to manage. Some of the requirements stated at the beginning of this project for the robot platform MAMoRo were low-cost and ease-of-use, while at the same time be flexible, expandable, and possessing enough processing power to be used in processing-intensive applications. These requirements may sound far-fetched, but with the right design and architecture, they are achievable.

In MAMoRo, the AMR four basic functional units have been distributed into three hardware modules: (The reasons for these distributions of functions are described in detail in section *3.2 The Architecture of MAMoRo*)

- *Power & Motion module* – containing the power unit and motion unit
- *Control module* – containing the sensor unit and part of the decision unit; this module will handle all low-level hardware tasks
- *Intelligence module* – containing part of the decision unit; this module will handle high-level processing-intensive tasks

The three modules altogether make the basic skeleton of the robot platform. The three modules are placed in a stacked-up form (Figure 3.1). The platform can also be optionally expanded to more than those modules to perform additional functions, or even increase the processing power of MAMoRo. Figure 3.10 illustrates clearly the three modules of MAMoRo and their connections.

3.2 The Architecture of MAMoRo

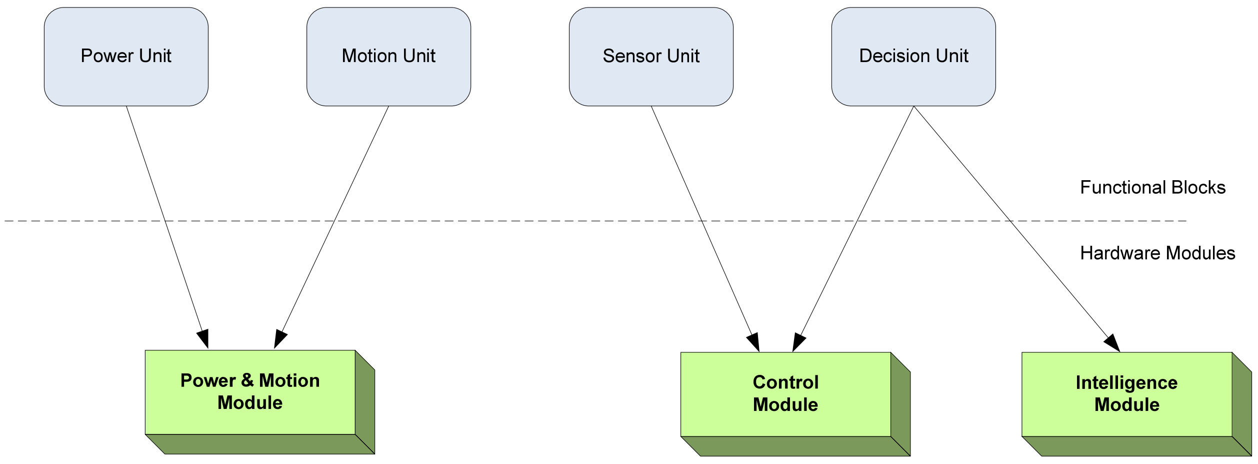


Figure 3.3: The AMR basic functional units in MAMoRo are distributed into three hardware modules

To fulfil the requirements stated at the beginning of this project for MAMoRo, the distribution of the AMR functions into MAMoRo's hardware modules were done as depicted in Figure 3.3. As can be seen from Figure 3.3, some of the AMR functional units have been merged, while others have been further partitioned. The merging of units reduces the number of modules in the system, consequently this reduces cost and complexity of the system, and makes the system easer to manage and handle. In the following sub-sections, the architecture of MAMoRo and general functions of the modules in MAMoRo are going to be described. The functions of the modules are described in more detail in section *3.3 The Modules of MAMoRo*.

3.2.1 Power and Motion Unit

The power and motion units were merged into a single module: *Power & Motion module*. The power and motion units are the least complex units, and contain few components; therefore, they were merged to reduce the number of modules in the system.

The power unit is responsible for providing a stabile power supply to the whole system and monitoring its voltage level. The motion unit drives the robot actuators according to the signals it reads from the Control module, and sends back to it odometry information, which are used for the navigation and path planning of the robot.

3.2.2 Decision unit

The decision unit can be considered the "brain" of the robot. All actions that the robot makes are decided here. Some decisions require more processing time than others, e.g. calculating the square root of a number requires more processing time than sending a command to the motion unit to tell the motor to move forward. The decisions a robot makes can be classified into two categories:

- Low-level functions: such as motor control, A/D conversion, read/write memory.
- High-level functions: such as path planning, artificial intelligence.

In MAMoRo, the decision unit functions are distributed into two modules according to this schema. The modules are:

- *Control module*: handles hardware low-level functions and provides these functions to the intelligence module through an interface independent from the underlying hardware. The main component is a low-cost, low power µC.
- *Intelligence module*: handles high-level, processing-intensive functions. The main component is a low-cost FPGA. The advantages gained by using a µC and an FPGA in a single system are explained further in this chapter.

This model of distribution of functions according to their complexity was inspired from the anatomy of the human brain. A part of our brain is responsible for critical low-level functions such as digestion, heart beating, respiration, which are done involuntary and without any of us "thinking" about it. Another part of the brain is responsible for the higher and more complex functions such as learning, reasoning, and self-consciousness. When a baby is born, the critical low-level functions are already “implemented” in his brain. That is, he does not have to learn such functions as breathing, or controlling heart rate. On the other hand, the part of the brain which is responsible for high-level functions such as taste, emotions, or language, are at birth “empty” and will be programmed during the life time of the baby. This resembles the FPGA – a volatile programmable hardware –, which is at first empty and is programmed at run-time during the operation of the system.

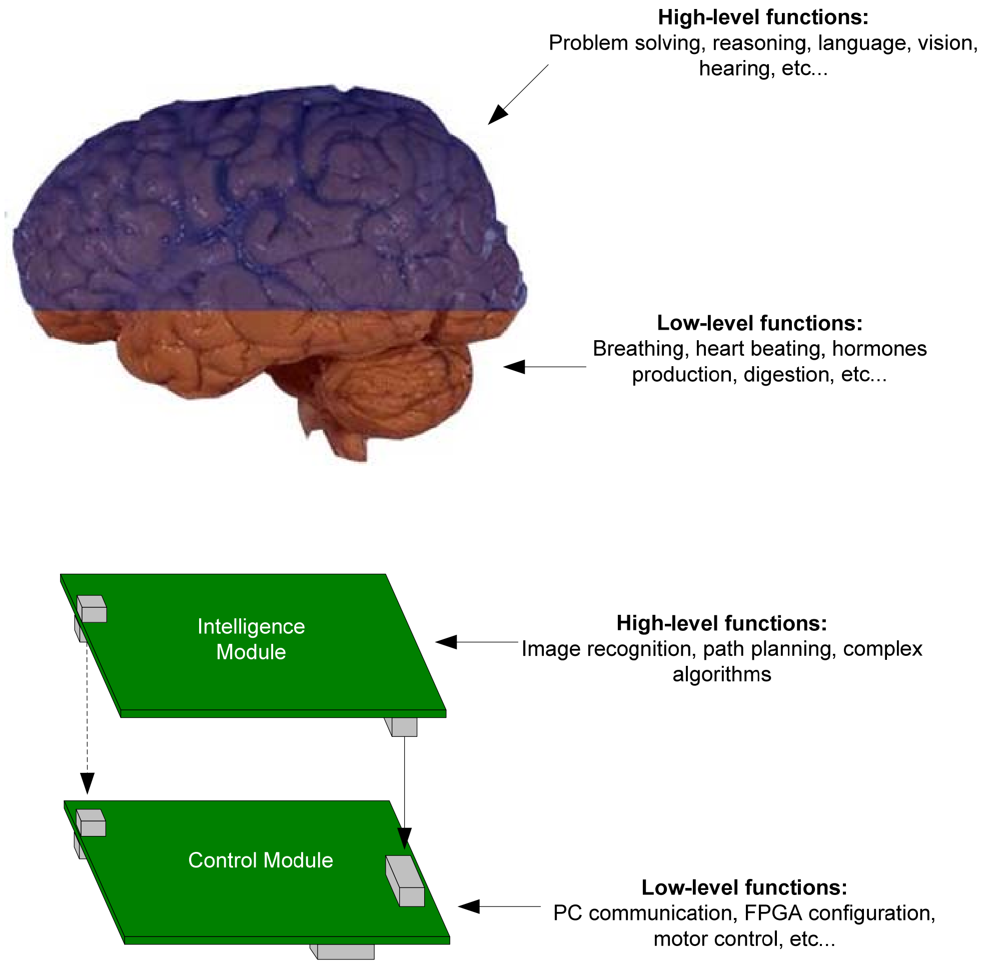

High-level functions:
Problem solving, reasoning, language, vision, hearing, etc...
Low-level functions:
Breathing, heart beating, hormones production, digestion, etc...
Intelligence Module
High-level functions:
Image recognition, path planning, complex algorithms
Control Module
Low-level functions:
PC communication, FPGA configuration, motor control, etc...

Figure 3.4: Comparison of high- and low-level functions in the human brain and the Control and Intelligence module. *Note*: the positions of the functions in the human brain depicted above are not anatomically correct.

The advantages gained in MAMoRo by this type of distribution are listed in the following page.

Advantages gained by using a Control module that handles low-level hardware task, and an Intelligence module that handles high-level and processing-intensive functions:

- A simple robot, which does not need intensive processing, can be implemented with just the Power & Motion module and the Control module, i.e. no need to use the Intelligence module. This reduces the complexity of the system and especially the power consumption; because when it comes to power saving, µCs (used in the Control module) have the edge over FPGAs. If later on it was decided that the robot should indeed do some intensive processing, e.g. graphic processing, complex algorithms, then the available robot platform can be upgraded by just plugging the Intelligence module to the Control module.
- The Control module can handle hardware low level tasks like driving actuators, reading raw data from sensors, converting analogue signals to digital, connecting to a PC for data downloading or uploading, etc.. The Control module can then provide these functions to the Intelligence module through an interface. Hence, the Control module can hide the complexity of the underlying hardware from the Intelligence module. This has great advantages:
    - The Intelligence module is revealed from handling low-level tasks and can use high-level language and abstract functions.
    - Programs written for the Intelligence module are independent from the underlying hardware and remains unchanged if an electronic component is swapped, e.g. a bigger Flash memory, a different motor driver; therefore, increasing flexibility and extendibility.
- For learning purposes, e.g. in introductory courses, the students can first be introduced to hardware programming by letting them first use the Control module. When they master this module, they can move on to the Intelligence module with the on-board FPGA, since using and programming an FPGA is more demanding then using and programming a µC.
- Both modules can be modified, upgraded and replaced independently from each other, as long as the interface connection between both modules stay the same.

The reason for implementing two different types of programmable units in MAMoRo, i.e. a µC in the Control module and an FPGA in the Intelligence module, is to exploit the advantages of each type of programmable unit; these advantages are listed below.

The advantages of microcontrollers over FPGAs:

- Lower power consumption
- Cheaper to purchase and cheaper developing tools
- Easier to program, modify, and handle
- On-chip peripherals, e.g. ADC, Timers, USART, SPI, etc.
- Predictable time behaviour – The microcontroller circuit is integrated in silicon and is unchangeable; unlike the FPGA which has variable timing performances.

The advantages of FPGAs over microcontrollers:

- An FPGA is a programmable hardware device; a specific application implemented in hardware will run a lot more efficient than if it was written in software.
- Flexibility – any logic circuit can be programmed on it
- Faster real-time response – required for interactive Robots
- Parallelism – FPGAs can execute commands in parallel; microcontrollers do this in sequence. Digital signal functions implemented in FPGAs can run orders of magnitude faster then most DSP (Digital signal processing) processors.

Apart from the advantages listed above, the combination of a µC and an FPGA in a system allows that a part of MAMoRo's hardware to be reconfigured at run-time. So, for example, multiple hardware codes can be stored in a flash memory and then uploaded to the FPGA whenever needed. An example of an application where hardware re-configurability might be useful is, for example, in an application where MAMoRo must achieve different tasks efficiently and fast. The algorithms code for each of these tasks can be stored in a flash memory and uploaded to the FPGA when they are needed. Implementing these various tasks in software would not be as much efficient.

3.2.3 Sensor Unit

The primary function of the sensor unit in the AMR model (Figure 3.2) is to pre-process the raw data received from sensors, e.g. convert analogue signals to digital, filter noises, etc. In MAMoRo, these functions are done by the µC in the Control module. The µC implemented in the Control module must have an on-chip A/D converter and sufficient processing power to handle these tasks. By merging the sensor unit to the Control module, we have further reduced the number of modules in MAMoRo

3.3 The Modules of MAMoRo

As have been previously mentioned, MAMoRo's modules are placed in a stacked-up form (Figure 3.1). The modules are connected with each other using headers; sockets on the bottom-side of the upper module plug into pin headers on module beneath it. Therefore, there should be no need for any wires for the connection of modules. All modules are designed to be usable in stand-alone mode. That is, each module can be used separately and perform its main functions without the presence of other modules. This feature is one of the requirements stated at the beginning of this project.

After giving an overview of MAMoRo's architecture in the previous sections, the individual modules are going to be described in more detail in the following sub-sections. The hardware design of the modules is dealt with in *Chapters 4 to 6*, while the implementation of these modules are dealt with in *Chapter 7*.

3.3.1 Power & Motion Module

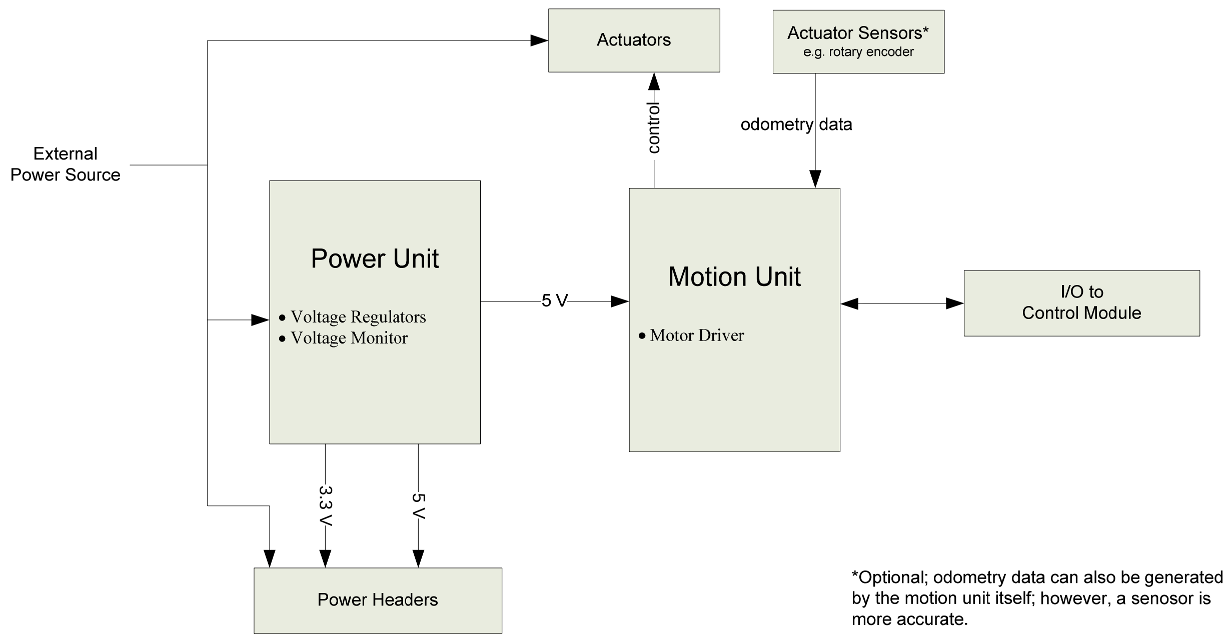


Figure 3.5: Power & Motion module block diagram

This Power & Motion module, as the name already tells, contains the power and motion unit. They are placed on the same module to reduce the overall number of modules in MAMoRo.

The power unit provides all modules with a regulated voltage supply and has a voltage monitor that monitors the external power source if it drops low. The power unit should provide through headers three voltage supplies:

- 3.3 V
- 5 V
- External power source (e.g. from battery)

Particularly the 5 V and 3.3 V voltage supplies because these are the voltage levels used by most of today's electronic components. The external power source should also be made accessible through headers, so that when needed instead of connecting wires directly to this external power source, it can be connected to the header on the Power & Motion module instead.

The low voltage monitor is especially important in battery-powered robots. If the power source drops below a certain voltage, it should warn the user by turning on a LED; the user then knows its time to recharge or change the batteries. If it drops further below a second certain voltage, the voltage monitor shuts the power source to the whole system; this is done to keep the system's behaviour predictable and avoid damages happening to the system itself. The voltage level at which the LED is turned on to warn the user that the battery is low and at which the system is turned off, should be separately adjustable with two potentiometers.

The motion unit reads the control signals from the Control module and drives accordingly the actuators, e.g. DC motors. It also sends back odometry data to the Control module from either a sensor integrated in the actuator itself such as a rotary encoder, or from the circuitry of the motion unit itself. The motion unit circuitry can detect changes in the current usage and transforms these changes into voltage changes, which are then send to the ADC of the µC in the Control module. However, a rotary sensor integrated in the motors will give more accurate results. From the odometry data, information such as speed and direction can be calculated, which are essential for navigation, and path planning in mobile robots.

The hardware design for the Power & Motion module is dealt with in *chapter 4* Power & Motion Module.

3.3.2 Control Module

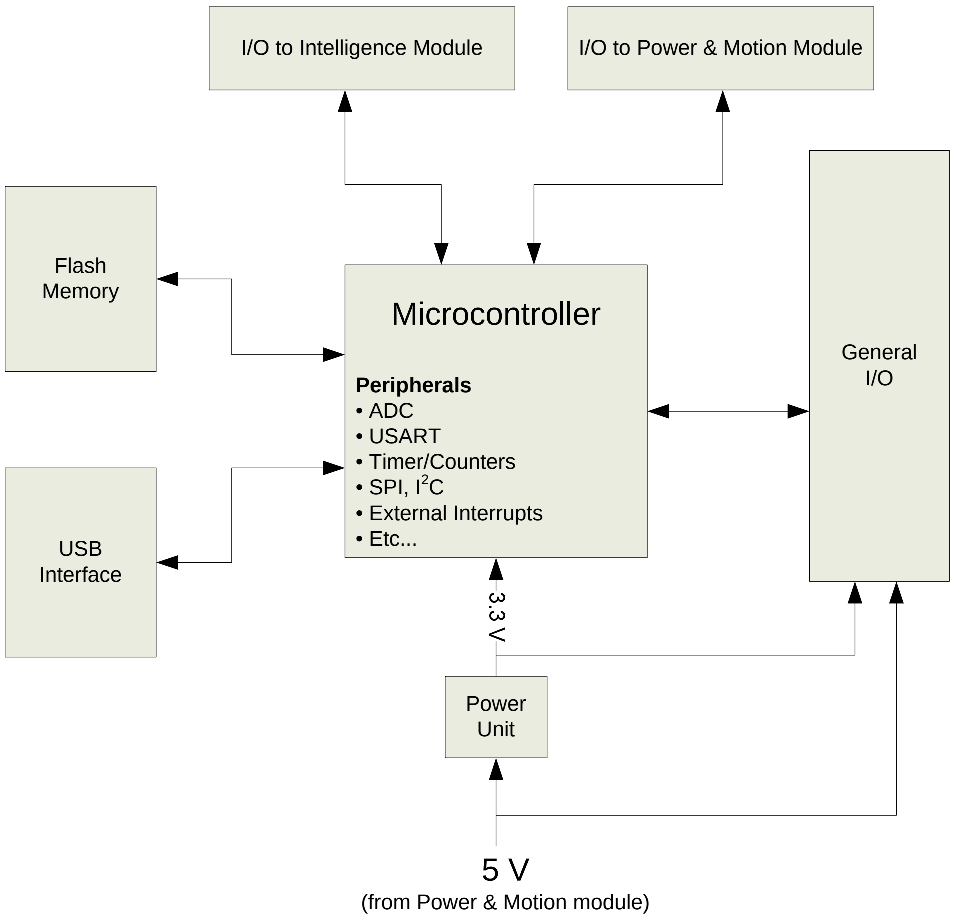


Figure 3.6: Control module block diagram

The Control module handles all low-level hardware tasks, and hides the complexity of the underlying hardware from the Intelligence module; it then provides the latter with these functions to through an interface. The Control and Power & Motion module make a complete robot, which can be used to do simple tasks (e.g. Line follower), but it is still unsuitable for applications requiring high processing power. The main units of the Control module are (Figure 3.6):

- Microcontroller (μC)
- USB interface
- Flash memory – used to save non-volatile data. The FPGA configuration file is stored here, and is uploaded to the FPGA whenever needed.

- Simple power unit – although already implemented in the Power & Motion module, this simple power unit makes the Control module usable in stand-alone mode.

The µC is the main component, and the only processing and programmable unit in the Control module. It handles many system tasks such as:

- Communication between MAMoRo and the PC for downloading and uploading of data.
- Interaction with the motion unit in the Power & Motion module, e.g. generation of PWM signals for motor speed control.
- Handles data transfer to and from the flash memory.
- Configuration of the FPGA in the Intelligence module; afterwards, handling the communication with it.
- Converting analogue signals from the external world to digital for later processing.

In general, the µC handles all low-level tasks and relieves the Intelligence module from the underlying hardware. The µC will be responsible for all system critical tasks; to perform them efficiently, the µC has to fulfil some requirements. The requirements for the µC are:

- Low-cost
- Low-power consumption
- A rich variety of integrated peripherals; peripherals that are particularly essential for the overall operation of MAMoRo:
    - *Serial Peripheral Interface* (SPI): a simple and fast serial communication bus integrated in many of today's µCs; it needs only four wires. It is also used for data transfer with SPI-enabled flash memories, and is the preferred communication method between the µC and the FPGA in the Intelligence module.
    - *Analogue/Digital converters* (ADCs): needed for conversion of analogue signals from the external world to digital for further processing.
    - Timers/Counters: needed for time measuring and generation of PWM signals, which are used, for example, to control the robot actuators connected to the Power & Motion module.
    - External interrupts: used to interrupt the execution of the main program when the voltage level on certain pins changes. External interrupts are important in

robots, because they decrease the response time to an external event without taking much processing time from the µC.

- I^2C (Inter-Integrated Circuit) bus: is a widely used bus for connecting low-speed devices; needs only two lines.
- USB (*Universal Serial Bus*) **or** UART (*Universal Asynchronous Receiver / Transmitter*) port: the preferred communication method between the µC and PC is the USB bus, since it is widely available in most of today's PCs and notebooks. If the µC used does not have a USB port (the cheapest µC's do not), an additional USART-USB converter IC can be used to convert the USART signals into USB and vice versa (see Figure 3.7). Nevertheless, in the end, the method with the USB communication through the UART-USB converter was chosen for this project; the reasons for this choice are listed in *5.8 USB Interface*.

- Sufficient processing power: 8 MHz operating frequency (at 1 MIPS per MHZ) would be adequate for hardware control functions and simple robot applications.
- Availability of open source development tools

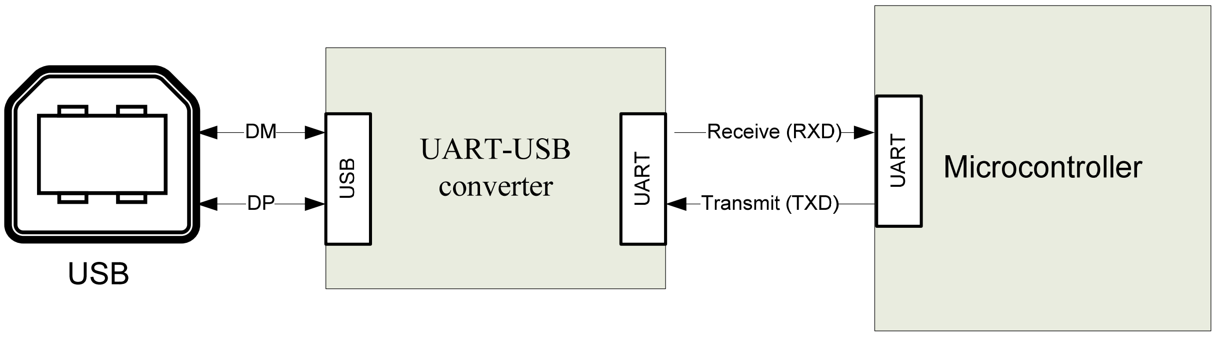


Figure 3.7: USB-µC communication through an UART-USB converter

The hardware design for the Control module is dealt with in *chapter 5 Control Module*.

3.3.3 Intelligence Module

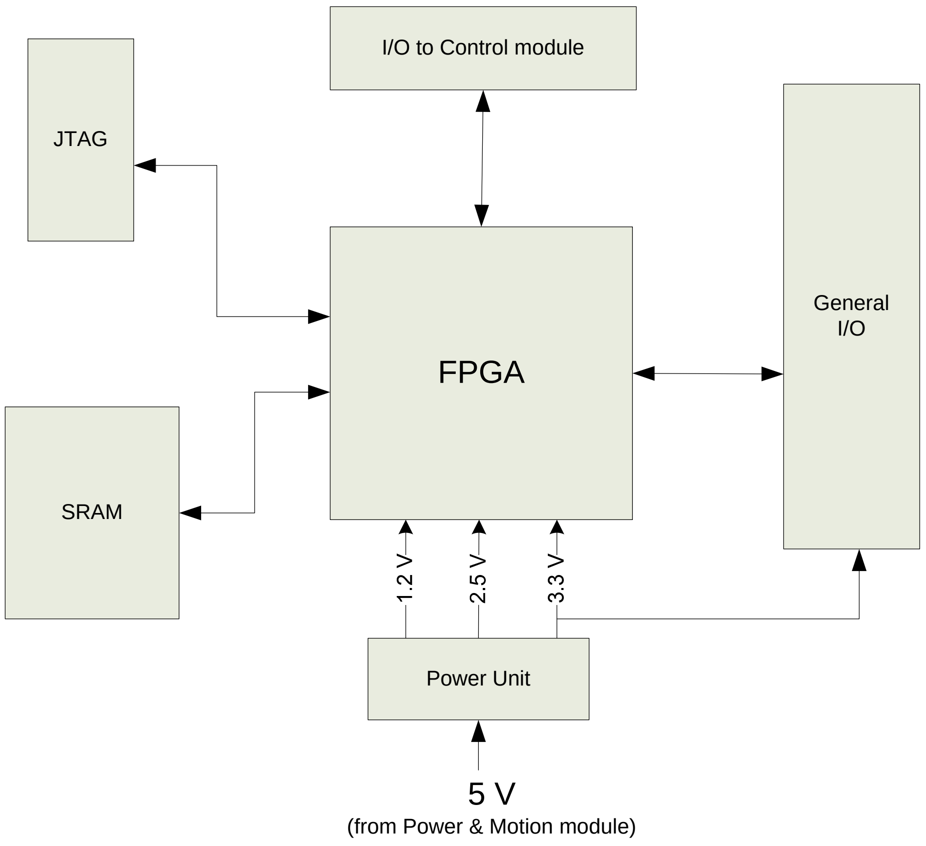


Figure 3.8: Intelligence module block diagram

The Intelligence module is the seat of the most powerful processing unit in MAMoRo. While the Control module is busy with low-level hardware tasks, this module does the real "thinking". With this module, advanced robotic applications requiring processing power such as image recognition and artificial intelligence are possible. The main components of the intelligence module are:

- FPGA
- SRAM
- Power unit – although already implemented in the Power & Motion module, this power unit makes the Intelligence module usable in stand-alone mode.

The FPGA is a programmable hardware unit. It is programmed through a hardware description language such as VHDL or Verilog, unlike the µC is programmed through a software programming language such as C. The FPGA can be seen as a programmable RAM

(*Random access memory*), because just like RAM, before it is programmed it is empty and cannot do anything. An FPGA is also volatile, meaning that each time the power is shut-off, all data in it is lost. The hardware inside the FPGA can be modelled to do anything, unlike the µC with its fixed hardware. For example, a soft core processor can be implemented in the FPGA for processing-intensive applications, and the SRAM (*Static Random Access Memory*) in the Intelligence module can be used then with this processor as a fast data buffer.
Since the FPGA is volatile, the configuration has to be done each time the power is turned on. The configuration of the FPGA is done by the Control module. The µC in the Control module takes the configuration file from the flash memory and uploads it to the FPGA, but before that, the FPGA configuration file has to be downloaded from the PC to the flash memory (see Figure 3.9); this is also the responsibility of the Control module. The FPGA can also be configured directly from the PC through the JTAG interface. After the configuration of the FPGA is done, the FPGA can communicate with the µC in the Control module through general I/O pins, or through the serial bus SPI (Figure 3.9). The SPI Bus is the preferred way for the µC-FPGA communication, because:

- The SPI is already integrated on the µC's hardware, so it does not have to be extra programmed ,and hence processing time wasted.
- SPI is simple and fast
- It is a serial bus and requires only four pins.

The FPGA used in MAMoRo has to fulfil some requirements such as:

- Low cost
- A minimum of 100 MHz operating frequency should be possible
- Big enough to store popular property soft processors, such MicroBlaze[1] by Xilinx™, and open source ones like JOP[2], OpenRISC[3], or LEON[4], while still having space for more hardware logic.

The hardware design for the Intelligence module is dealt with in chapter *6 Intelligence Module.*

[1] http://www.xilinx.com
[2] http://www.jopdesign.com/
[3] http://www.opencores.org
[4] http://www.gaisler.com

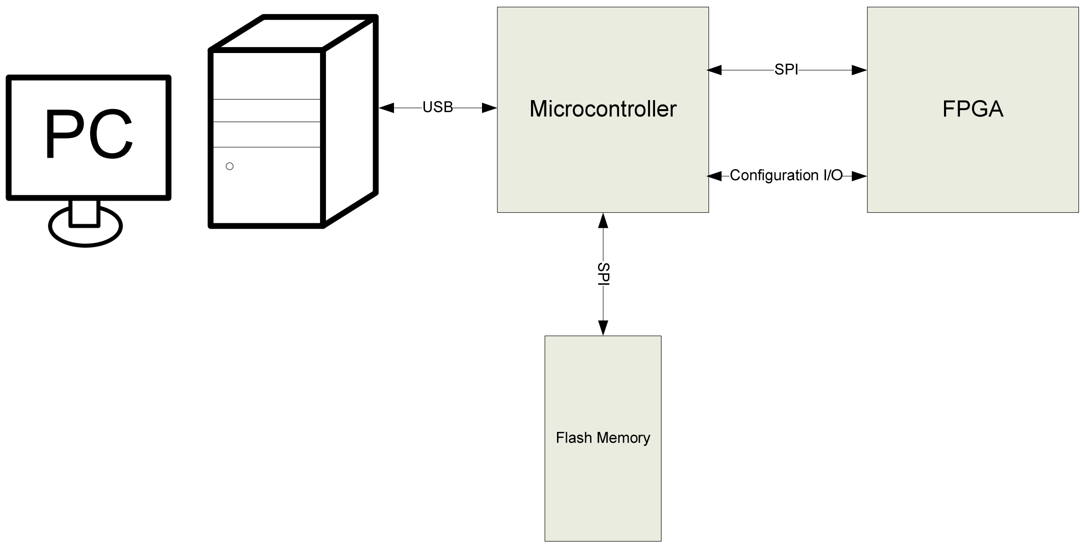


Figure 3.9: Connections between microcontroller, FPGA, and PC

3.4 Stand-Alone Mode

One of the requirements stated at the beginning of this project was that the modules of MAMoRo should also be usable as stand-alone, i.e. the modules can be operated without the presence of other modules

3.4.1 Power & Motion module

The power unit in the Power & Motion module can be used to provide stable 5 V and 3.3 V voltage supplies, and monitor the input power source voltage level. The motion unit, however, can not control the actuators without the control signals, which it usually gets from the Control module.

3.4.2 Control Module

The needed 3.3 V power supply for all components in the Control module is generated by a simple power unit in the module itself. Although, the 3.3 V supply from the Power & Motion module could have been used directly, the reason for implementing an additional power unit in the Control module is so that this module can be used in stand-alone mode without the presence of the Power & Motion module. The Control module has also no dependency from the Intelligence module.

The Control module with its on-board µC and peripherals can be used as an evaluation platform for various electrical engineering experiments.

3.4.3 Intelligence Module

The Intelligence module requires three different power supplies: 3.3 V, 2.5 V, and 1.2 V. Instead of implementing these power supplies here, they could have been implemented in the Power & Motion module, and then provided to the whole system whenever needed. However, that would have made the Intelligence module not operable in stand-alone mode, therefore, it was not done.

When the Intelligence module is used with the Control module, the latter handles the configuration of the FPGA. When the Intelligence module is used alone, the FPGA is configured by connecting the JTAG header on this module to a PC through a programmer cable (see section *8.1.1 The Parallel Programmer Cable*). The main problem with using the Intelligence module in stand-alone mode is that the FPGA configuration data is lost whenever the power supply is turned off; if that happens, the module has to be reprogrammed again from the PC. This problem does not exist if the Control module is attached, because the Control module will always configure the FPGA automatically when the system is powering on, or when the FPGA needs to be reconfigured again by signalling it to the Control module through one of its pins.

The Intelligence module with its on-board FPGA and other peripherals can be used as an evaluation platform for various electrical engineering experiments.

3.5 The Physical Implementation

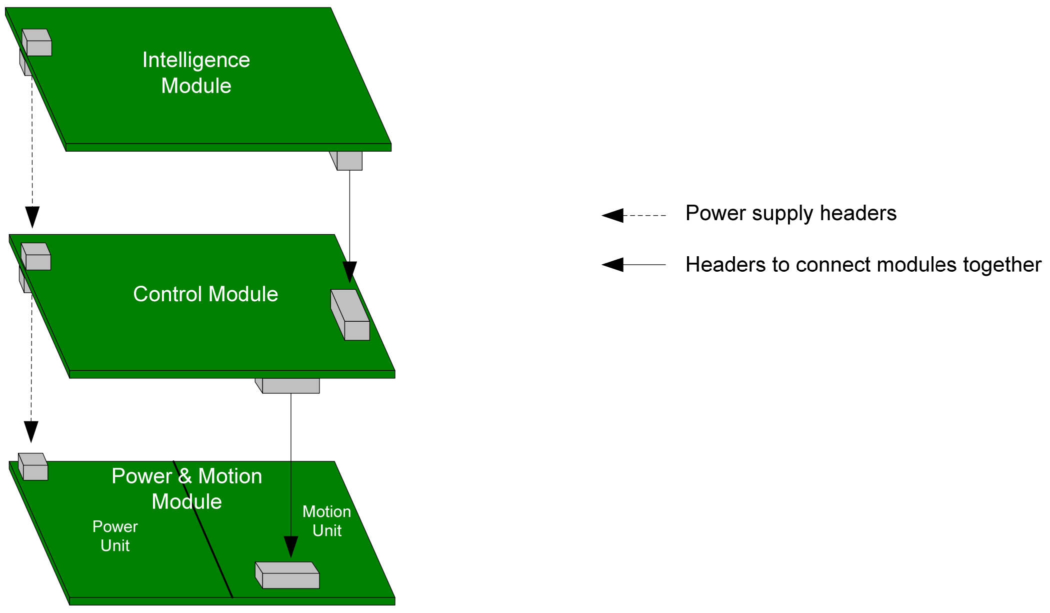


Figure 3.10: MAMoRo's modules. The arrows represent the electrical connections between the modules. The grey blocks are headers used to connect the modules together.

All modules are going to be placed vertically over each other in a stacked-up form as depicted in Figure 3.10, to save area. All modules should have the same area size. The electrical connection between each module is achieved by plugging a female header placed on the bottom-side of the upper module, with a male header placed on the top-side of the module below it (as in Figure 3.10); because of this, there is no need for additional cables to connect the modules together, therefore, avoiding a cable salad. The male headers (pin connectors) on all MAMoRo modules should be placed on the top-side, while the female headers (sockets) are placed on the bottom-side.

The order of the modules starting from the bottom should be (Figure 3.10):

1. Power & Motion module – placed at the bottom of the stack to make the I/O pins of the Control and Intelligence module easier to access.
2. Control module – placed here because it needs a direct connection to the motion unit in the Power & Motion module.

3. Intelligence module – placed at the top, because it has the highest number of user-I/O pins, hence they need to be easily accessible, and because it does not need a direct connection to the Power & Motion module.

Expandability was also stated as one of the requirements that MAMoRo should fulfil, that is, it should be easy to expand the functionality (or processing performance) of MAMoRo by adding new modules to the stack. To make the addition of new modules easier and more flexible, the headers on the Control and Intelligence module (the Power & Motion module does not have any user-I/O pins) have to be so placed, that as many as possible of the headers on the Control module lay exactly below the headers of the Intelligence module (Figure 3.11). The advantage of this placement of headers is that a new added module can be fitted above either the Control module or the Intelligence module with no physical change on the module itself; Figure 3.11 illustrated this.

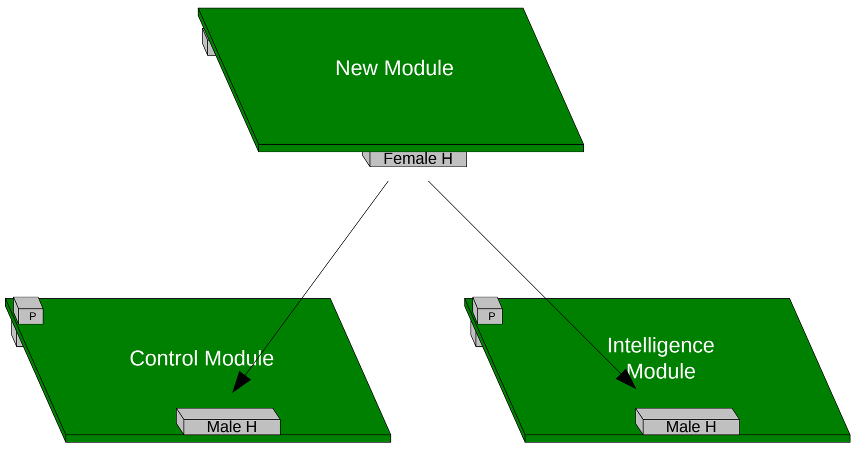


Figure 3.11: The new added expansion module can be fitted above the Control or Intelligence module, since the headers on both modules lay on the same position.

One could also use cables to connect the new added module with the other MAMoRo modules, instead of using headers as depicted in Figure 3.11. However, if the cables get too many we may get then a cable salad, and besides it looking messy and loosing the oversight of the connections, some EMI issues might pop up. This cable salad can be avoided if instead headers were used to connect the modules together.

Another consideration for the physical implementation is the area size of MAMoRo's modules. Although in many applications having modules with small area size is wanted, in some applications it is not; therefore, there will be no requirements regarding the area size of the modules, and the area can be varied according to where MAMoRo is going to be used. For example, if MAMoRo is going to be used by students in introductory courses, then probably a relative large module area size is favourably, because then the students will get a feeling of what components are used and perhaps even be allowed to solder the components on the modules themselves. This will be difficult if the modules are small. The modules of MAMoRo allow the implementation of very small-sized modules, because: the Power & Motion module is simple and contains few components; and most of the functions of the Control and Intelligence module are packed on only one IC in each module, i.e. the µC and FPGA, respectively.

Finally, headers with specialized functions, e.g. JTAG or ISP, should be shrouded to avoid plugging connectors the wrong way, and hence reducing the probability of causing damage.

3.6 Programming MAMoRo

A major disadvantage with distributed systems composed of many modules is the manageability of the overall system. If a system consists of multiple modules with on-board programmable units, then obviously each one of these programmable units have to be programmed sooner or later.
MAMoRo has two programmable units: the µC in the Control module, and the FPGA in the Intelligence module. There could be even more if expansion modules with on-board programmable units are added. To make the programming of the system as simple as possible, and reduce the time needed for it, only one single programming unit has to be programmed, and that is the µC in the Control module. The µC should then take care of programming all other programmable units in the system including the FPGA in the Intelligence module. It should be possible to program, debug, and hence manage the whole system through a single popular interface: the USB interface.

The drawback in using two different types of programmable units in a single system like a µC and an FPGA is that they have to be programmed using two different families of programming languages. While the µC is programmed using a software programming language such as C, the FPGA is programmed through a hardware description language such as VHDL. However, there has been progress recently in developing programming languages that can be used for both programming software and hardware. An example of such a programming language is SystemC[1].

Finally, the software tools used for programming and interacting with MAMoRo should be open source whenever possible, because of transparency, flexibility, and cost.

[1] SystemC homepage: http://www.systemc.org/

4 Power & Motion Module

The specifications for the Power & Motion module were laid down in *3.2.1 Power and Motion Unit* and *3.3.1 Power & Motion Module*. In this chapter, the actual hardware realization is described.

The Power & Motion module contains the power and motion unit. The power unit supplies and monitors the power supply for the whole system. The motion unit drives the actuators according to the control signals it gets from the Control module, and sends back odometry information. Each unit will be described in detail in the following sections.

The complete schematics, board layout, and parts list for this module are in *Appendix A - Schematics, Board Layouts, and Parts List*. While the description for the header pinouts of this module are in *Appendix B - Headers Pinouts.*

4.1 Power Unit

For the electronic components to function properly in a system, a stabile and regulated voltage supply must be provided. This is basically the function of the power unit. Another task of this unit is monitoring the external voltage source if it gets low.

4.1.1 Power Supplies

The power unit will supply two regulated voltage supplies: 3.3 V and 5 V. The first stage of stepping down the external power source to 5V is done by a switching regulator. A switching regulator is used in this stage because of its high power efficiency in stepping down the input voltage. The next stage in stepping down the 5V voltage supply to 3.3 V is done by a cheaper, but less power efficient linear type voltage regulator. The linear type voltage regulator has the advantage that its output voltage is cleaner (i.e. less ripples) than the switching regulator. This is because the switching regulator regulates the output voltage by switching it periodically on and off at a specified rate, and although there is a capacitor placed at the output to filter out the ripples generated by this process, not all ripples are filtered out. These ripples generate electromagnetic interference (EMI) and might cause problems with low voltage devices because of their sensitivity to voltage instability. The main advantages and disadvantages of linear regulators compared to switching regulators are listed below.

Advantages

- Low noise: Little or no electrical noise is generated at their outputs, whereas switching regulators generate a considerable undesirable electromagnetic interference due to the continues switching of the output current [RAS01].
- Linear regulators are more suitable for low power application because of their clean output. Low-voltage devices are usually more sensitive to voltage instability than devices operating at a higher voltage.
- Compact and simple. They need only decoupling capacitors to operate optimally.
- Cheaper

Disadvantages

- Poor efficiency. Their conversion power efficiency ranges from 20 to 60%, whereas switching regulators can have efficiency from 70 to 95% [RAS01].

5 V supply

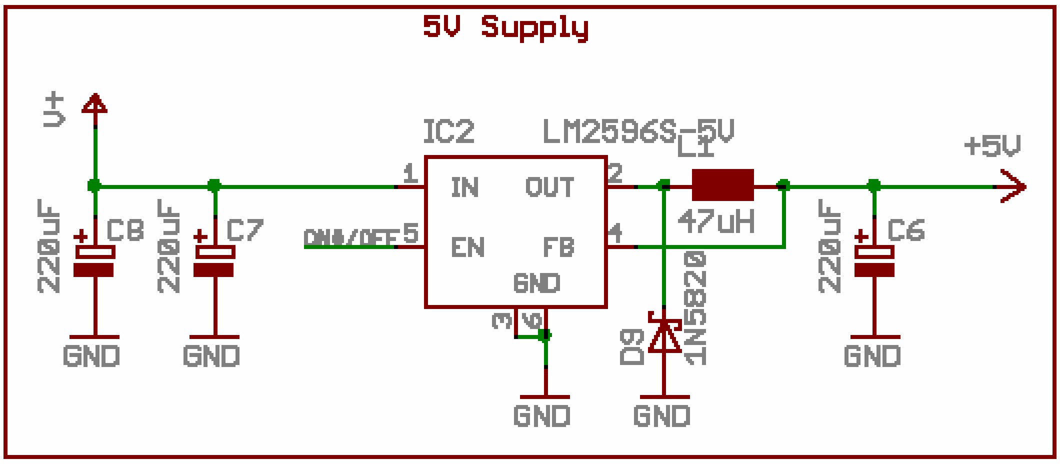


Figure 4.1: 5 V supply: generated by a step-down switching regulator

The 5V supply voltage is generated by a step-down (buck) switching regulator. The IC used in this project is a *LM2596S-5.0* from National Semiconductor™. The LM2596S-5.0 main features are:

- Voltage regulation (at $7V \leq V_{IN} \leq 40V$, $0.2A \leq I_{OUT} \leq 3A$): $4.8V \leq V_{OUT} \leq 5.2V$
- Efficiency (at $V_{IN} = 12V$, $I_{LOAD} = 3A$): 80 %
- Thermal shutdown and current limit protection
- Shut-down capability
- 150 kHz fixed frequency internal oscillator

The Thermal shutdown and current limit protection feature comes useful in hand when, for example, a short-circuit occurs, or if the system needs more current than the regulator can provide. The shut-down capability of this voltage regulator is used to shut off the power supply to the whole system if the external voltage supply drops below a certain level to keep the system's behaviour predictable and avoid damages happening to the system itself. To calculate the values of external components needed for the optimal operation of this regulator, the manufacturer offers on its website an online simulation tool WEBENCH (http://webench.national.com) for most switching regulators.

3.3 V supply

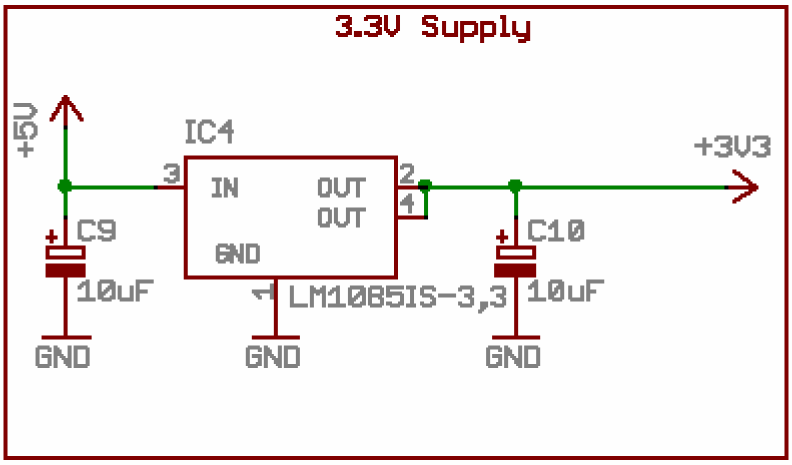


Figure 4.2: 3.3V supply: generated by a linear regulator

The 3.3V linear regulator used in this project is a *LM1085IS-3.3* by National Semiconductor™. It needs one bypass capacitor each at the input and output to operate optimally. The LM1085IS-3.3 main features are:

- Voltage regulation (at $4.8V \leq V_{IN} \leq 15V$, $0 \leq I_{OUT} \leq 3A$): $3.235 \leq V_{OUT} \leq 3.365$
- Thermal shutdown and current limit protection

Power Input

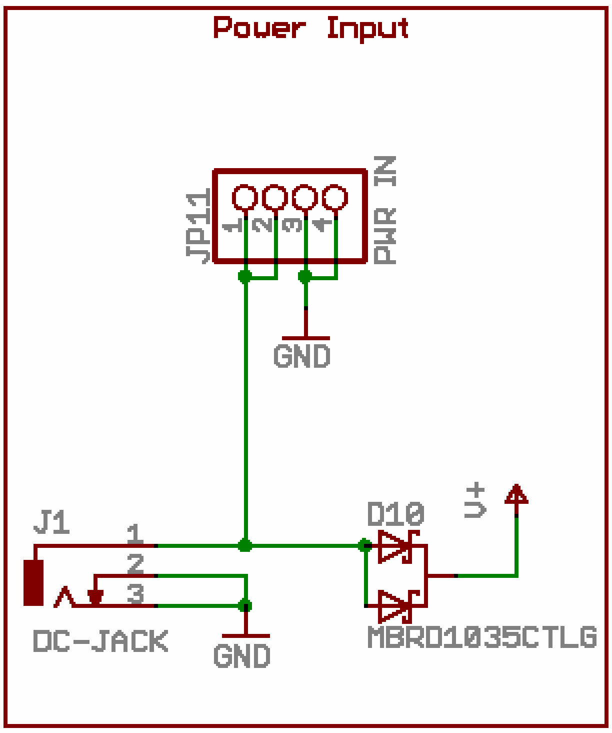


Figure 4.3: Power input of the Power & Motion module

The power can be applied to the system either through the DC-Jack or PWR IN Header (see Figure 4.3). The input is protected against accidental reverse-polarity by a low forward-voltage drop Schottky power diode.

Voltage Headers and Power Indicator

The power supplies from this module are provided through four headers. Each power source has its own header with the exception of the *Modules PWR* header. The Modules PWR contains all three voltage supplies: the external power source (V+, e.g. from a battery), and the both regulated voltage supplies 3.3 V (+3V3) and 5 V (+5V). All modules of MAMoRo get their power supply from this header. The power indicator is a simple LED which signals that an external power source is connected to this module.

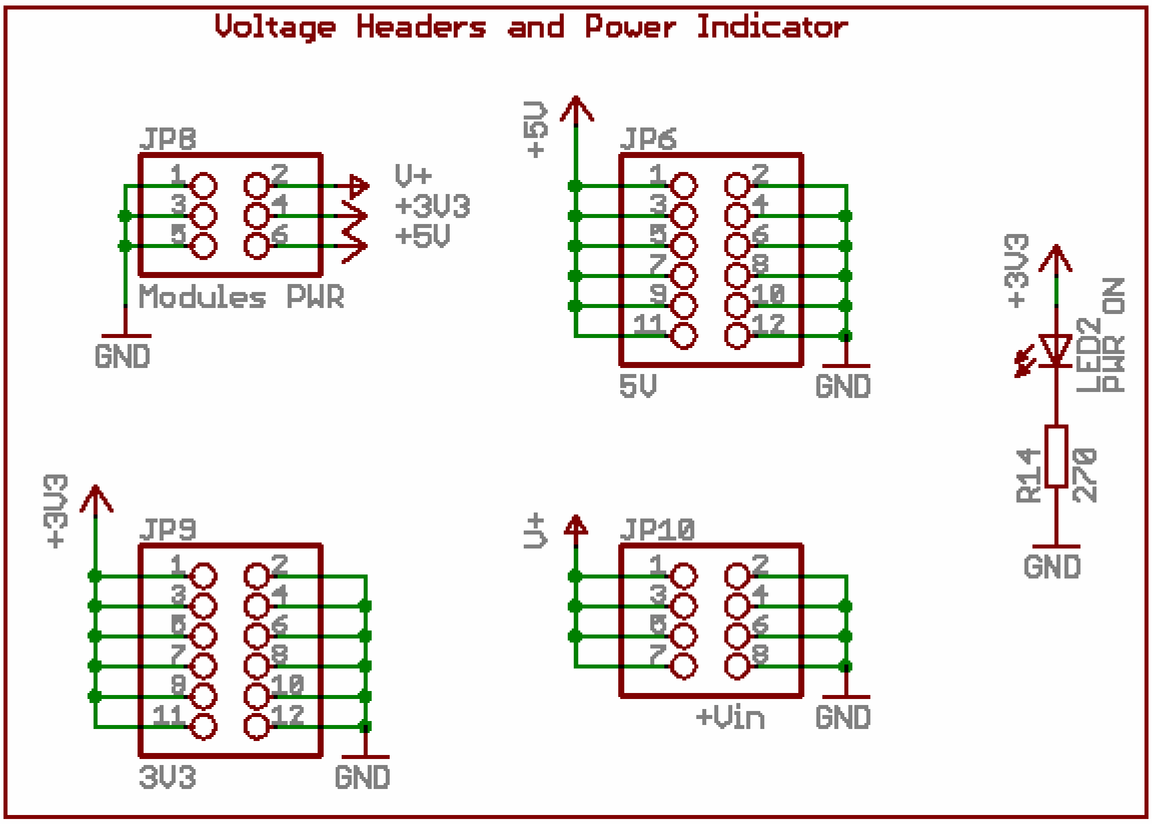


Figure 4.4: Voltage headers and the power indicator (far right)

4.1.2 Low Voltage Monitor

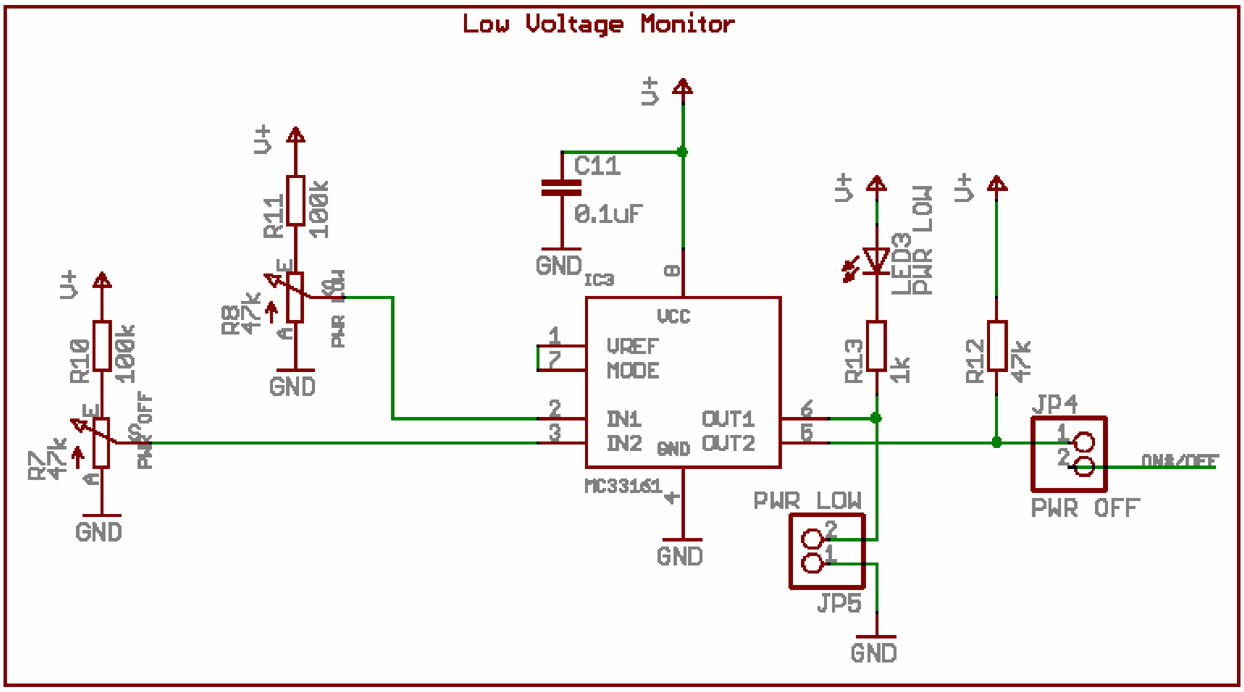


Figure 4.5: Low voltage monitor

In addition to providing a regulated voltage supply, the power unit has also a low voltage monitor that monitors the external power supply when it drops below specified voltage levels. The voltage monitor used in this project is a *MC33161* Universal Voltage Monito*r* from ON Semiconductor™. The voltage monitor consists of two comparators, each with a hysteresis, a mode select input for configuring it as non-inverting or inverting, and an internal 2.54 V reference voltage independent from the voltage supply applied externally to the IC. The outputs of the comparators consists each of an open collector circuit capable of sinking up to 10 mA. The voltage monitor is configured with two voltage thresholds: V_{LOW} and V_{OFF}. They are each individually adjustable by a potentiometer (R7 and R8 in Figure 4.5). If the voltage gets below V_{LOW}, OUT1 on the IC goes low, turning the LED on. This can be especially useful if the external voltage supply is a battery, thus when the LED goes on, it means its time to charge or change the battery. Additionally, the OUT1 signal is made available through the header PWR LOW (JP5). This could be used, for example, to connect OUT1 to the µC in the Control module, so that when the voltage goes low, the µC recognizes this and changes to power-saving mode. If the external power source drops further lower than V_{OFF}, OUT2 goes high, and if a jumper is placed on the header PWR OFF (JP4), this will shut off the 5 V voltage regulator, which is the main power supply to the whole system, hence the system will be shut down. This action is very important, because a state can occur where the input voltage drops low, but not low enough to shut off some of the IC's. The system will continue to work, but might be unpredictable and may corrupt data stored in non-volatile memory. If this option is not wanted, header JP4 can be left unconnected.

4.2 Motion Unit

The motion unit main reads control signals from the Control module and drives accordingly inductive actuators such as DC motors and stepping motors. It also sends back odometry data to the Control module.

4.2.1 Motor Driver

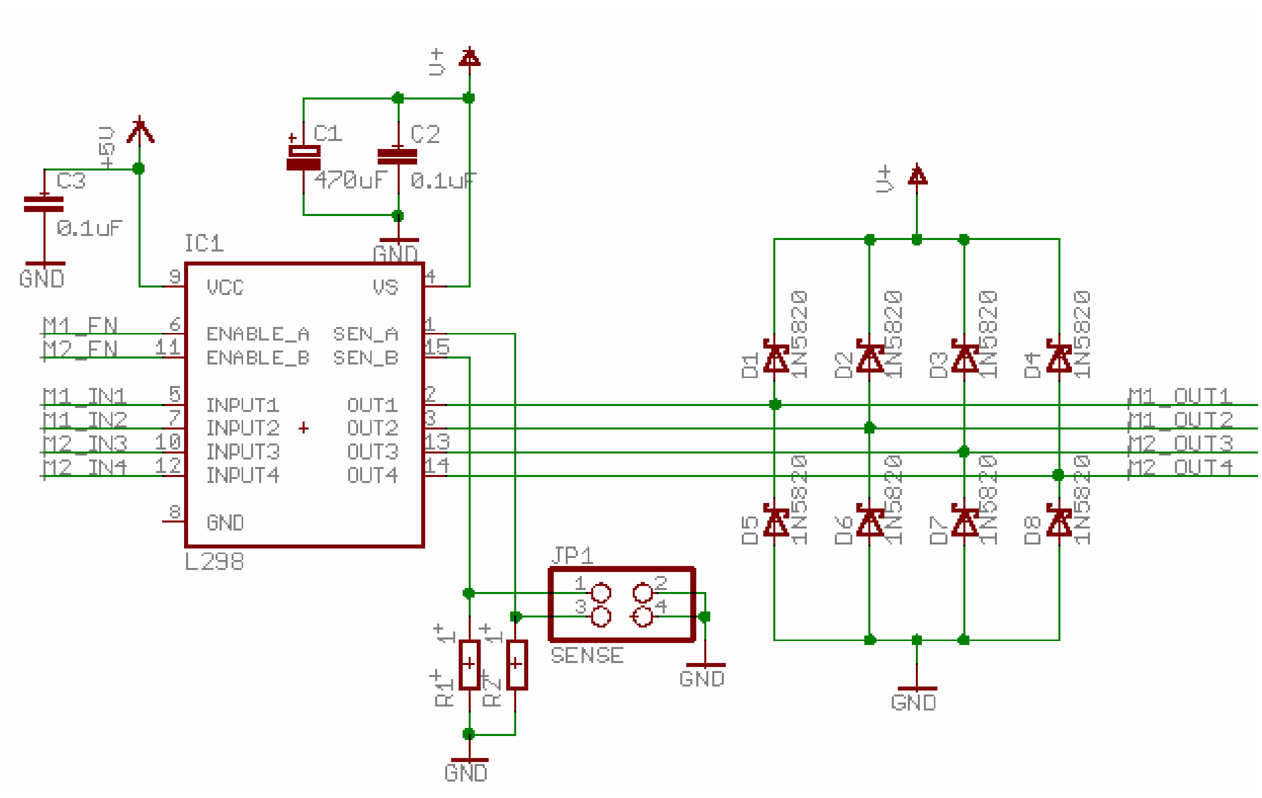


Figure 4.6: Dual full-bridge motor driver

The main component of the motion unit is the motor driver *L298* by STMicroelectronics™. The L298 is a high voltage, high current dual full-bridge driver which can drive up to two motors in both directions. The L298 main features are:

- Dual full-bridge driver
- Max. current per channel: 2 A (DC)
- Internal thermal protection circuit: Shuts down the IC when it gets too hot, e.g. when drawing too much current
- Max. load voltage V_s: 46V
- Logic supply voltage V_{ss}: 5V
- Low voltage logic circuit is isolated from the high voltage motor circuit. This has the advantage that the high power noise caused by the motors is not transferred back to the digital circuit.
- Total Power Dissipation: 25 W

Each of the two bridge drivers, bridge A and B, are independent from each other. Each bridge driver has three inputs: Enable A, Input 1, and Input 2 (Enable B, Input 3, and Input 4 for bridge B), which drive two output pins separately (see Table 4.1).

Table 4.1: L298 control signals

Input			**Output**
Enable A(B*)	**Input 1(3)**	**Input 2(4)**	**Function**
0	0	0	Fast motor stop
0	1	0	Forward
0	0	1	Reverse
0	1	1	Fast motor stop
1	X	X	Free running motor stop

* The Letters and numbers in brackets correspond to the second bridge-driver control signals.

The L298 has also for each bridge a sense pin (Sense A and Sense B). The current flowing through one load has to flow back to GND through this pin. By connecting a low ohm resistor (R1 and R2 in Figure 4.6) from the sense pin to GND, the current flowing through the load flows also through these resistors. By measuring the voltage drop in these resistors (e.g. through the on-chip µC ADC in the Control module) one can calculate the velocity of the motors rotation, and hence the distance travelled by the robot. If this is not needed because, for example, there is already an encoder integrated in the motors, then jumpers can be placed on header JP1(Figure 4.6) to conduct the sense lines directly to GND. The bridge of suppression diodes are there to protect the IC from voltage spikes. High current Schottky diodes were used here.

4.2.2 Motion Unit Headers

Motor Headers

The motors are connected to the motion unit through two 2x3 pin headers: Motor1-2 (JP3 and JP2; see Figure 4.7). Two lines (OUT1-2 and OUT3-4) drive the motors, while the other four pins are reserved for a quadrature encoder. Quadrature encoders are widely used in mobile robots to provide precise odometry data. In this project, two DC motors were used with each an integrated quadrature magnetic encoder.

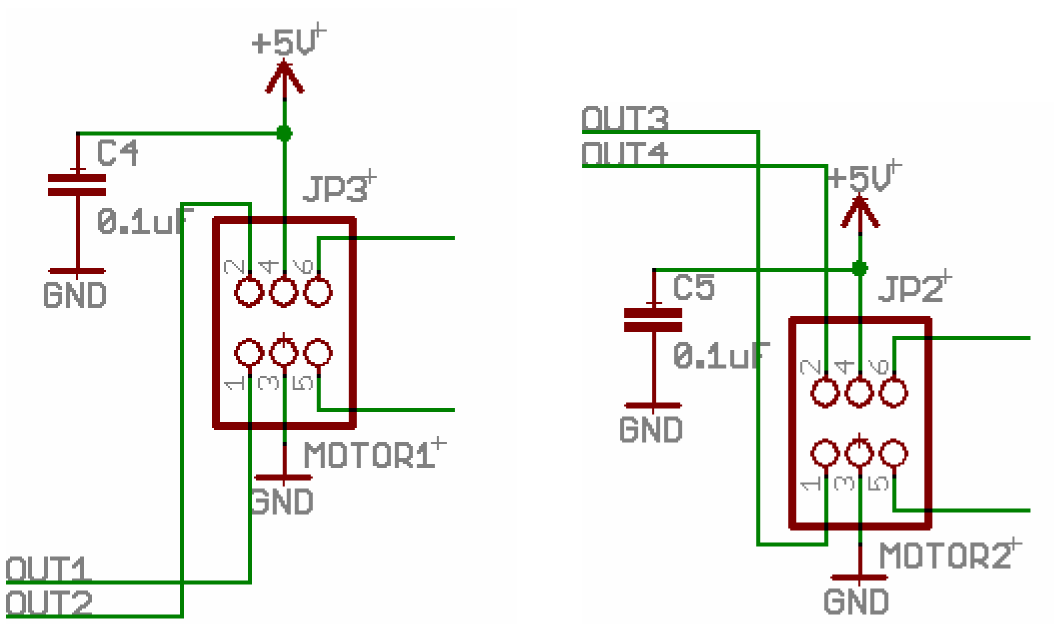


Figure 4.7: Motor Headers

Motion Module I/O Header

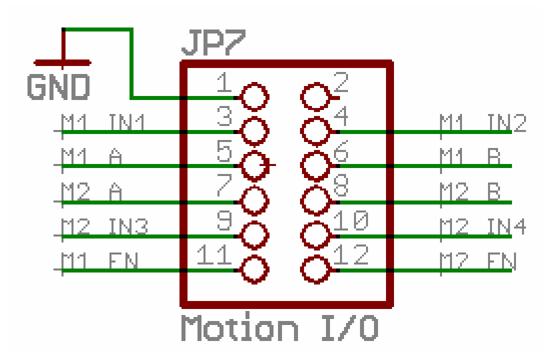


Figure 4.8: Motion module I/O header

The motion unit is connected to the Control module through a 2x6 pin header: Motion I/O (JP7; see Figure 4.8). The motion unit drives the motors according to the input it gets from the Control module, while the signals generated by the motor encoders are sent to the Control module for odometry calculations.

4.3 Specifications

Power & Motion module main hardware specifications:

- Provides three different power supplies:
  - 3.3 V
  - 5 V
  - External power source
- Voltage Monitor: monitors the external powers source for two voltage thresholds, which can be set individually by a potentiometer.
- Motor driver:
  - Drives up to two inductive actuators (e.g. DC motor)
  - Max. current 4 A; Max. voltage: 46 V
  - Sense pins to measure rotation speed of motors

5 Control Module

The specifications for the Control module were laid down in *3.2.2 Decision unit* and *3.3.2 Control Module*. In this chapter, the actual hardware realization is described.

The Control module has following basic functions (see *3.3.2 Control Module*):

- Handling the communication between the system and PC for downloading and uploading of data
- Controlling the motion unit
- Handling the data transfer to and from the flash memory
- Configuring the FPGA in the Intelligence module
- In general, handling all low level tasks and relieving the Intelligence module from the underlying hardware

When used as stand-alone, the Control module can be used as an evaluation platform to conduct a wide variety of experiments in electrical engineering and computer science. The Control module is functionally divided into:

- Processing Unit
- USB Interface
- Flash Memory
- Power Unit

The complete schematics, board layout, and parts list for this module are in *Appendix A - Schematics, Board Layouts, and Parts List*. While the description for the header pinouts of this module are in *Appendix B - Headers Pinouts.*

5.1 Power Unit

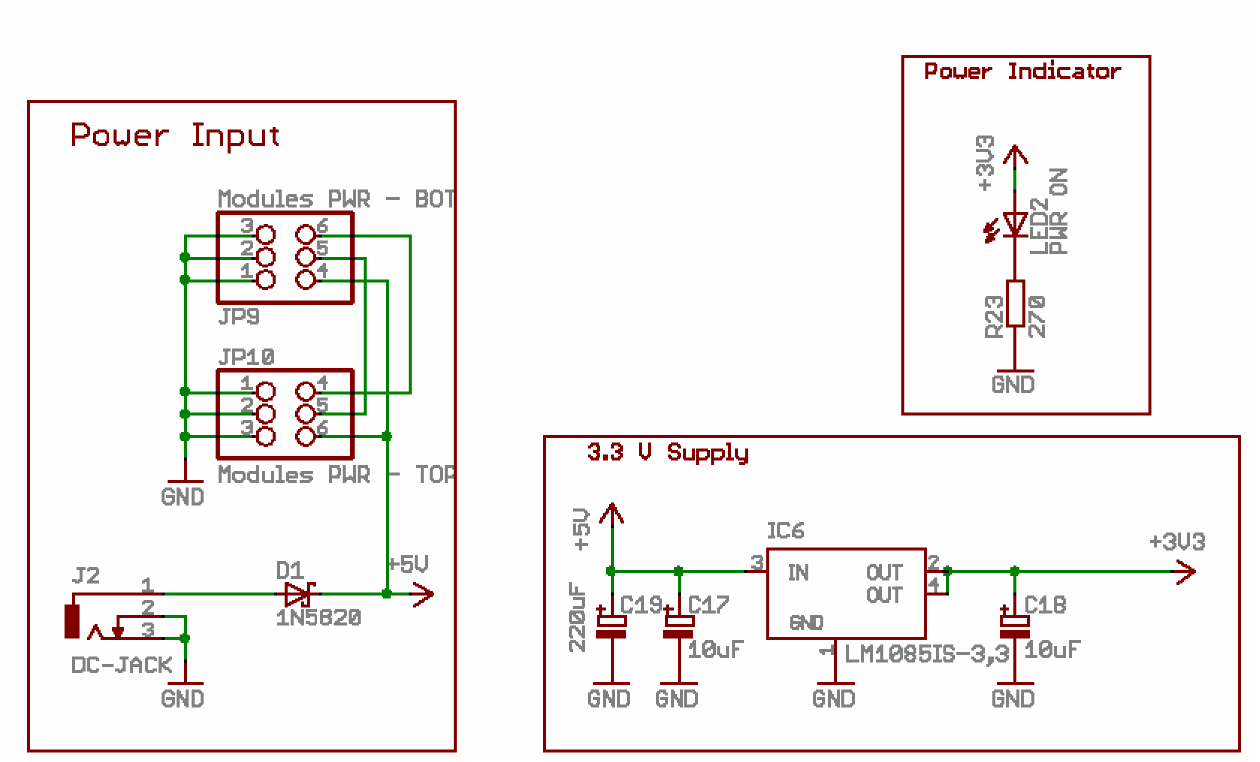


Figure 5.1: The power unit in the Control module

The 5 V input voltage of this module comes from the Power & Motion module through the header *Modules PWR* (Figure 5.1). The 3.3 V supply needed by all IC's in this module is generated on-board by a linear regulator: *LM1085IS-3.3* by National Semiconductor™. Although, the Control module could have drawn the 3.3 V supply directly from the Power & Motion module, it was not done, so that the Control module can also function as stand-alone. A simple LED connected to the 3.3 V supply indicates that a power source is plugged in.

When the Control module is used without the Power & Motion module, the input power source should be applied through the DC jack. The DC jack input is protected from reveres-polarity by a diode. The input voltage at the DC jack can be anything from 4.5 V to 27 V. However, if there are devices that need 5 V while the Control module is used in stand-alone mode, the input voltage applied at the DC jack has to be approximately 5 V.

5.2 Microcontroller Unit

The microcontroller unit, as the name already tells, contains the µC – the programmable processing unit of this module.

5.2.1 Microcontroller

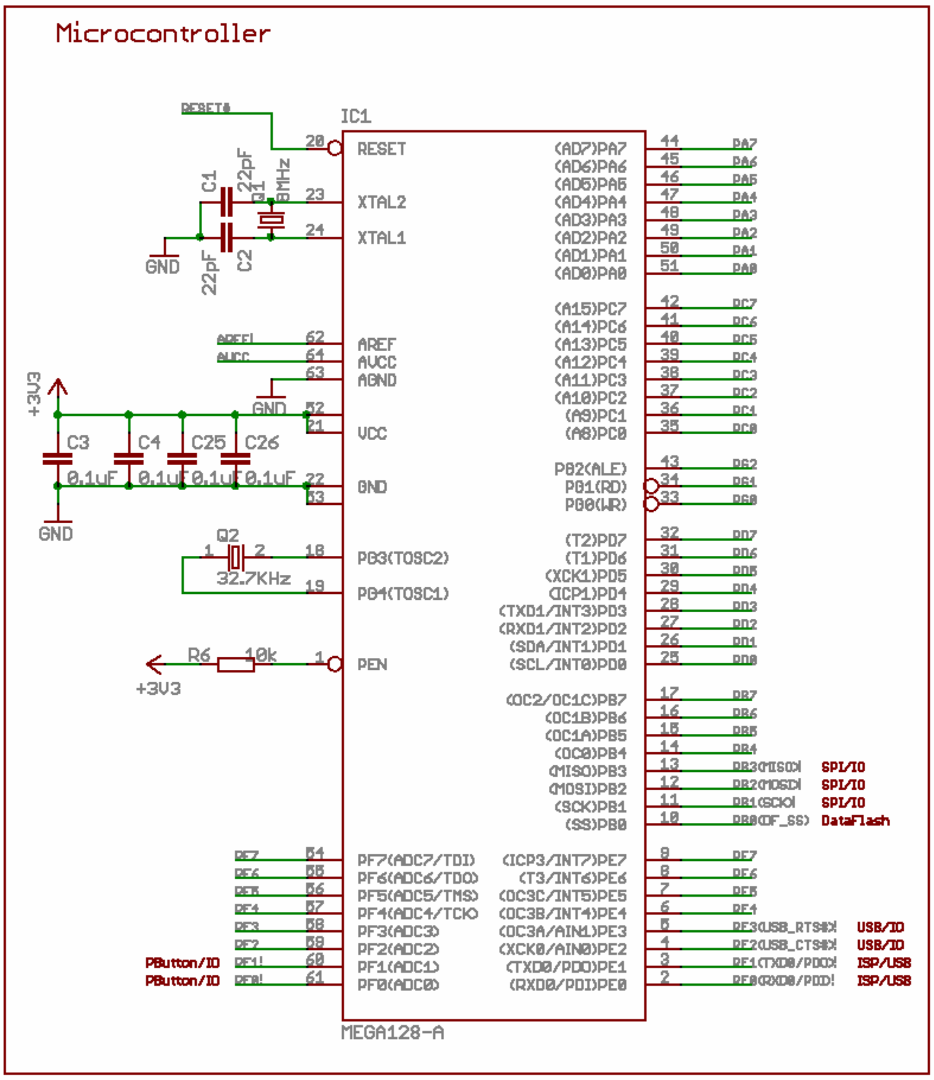


Figure 5.2: The Microcontroller

The µC is the main component of the Control module. Choosing a suitable µC was one of the hardest decisions during this project, because the alternatives are simply too many. In the end, the decision falls at an 8-bit RISC *ATmega128L* by Atmel™.

The main features of the ATmega128L are:

- 8-bit RISC architecture
- Operating Voltages: 2.7 - 5.5V
- Operating frequencies: 0 - 8 MHz
- 133 instructions - most of them need one cycle to finish executing
- Program and Data Memories
    - 128 KiByte flash memory,
    - 4 KiByte EEPROM
    - 4 KiByte SRAM
- Peripherals
    - 2 x 8-bit Timer/Counters; 2 x 16-bit Timer/Counters
    - 2 x 8-bit PWM channels and 6 x PWM channels with programmable resolution from 2 to 16 bits
    - Real time counter (RTC), by using a 32.768 kHz quartz crystal
    - 8 x 10-bit ADC with optional differential input stage with programmable gain
    - Two-wire Serial Interface (TWI), equivalent to I^2C
    - 2 x USART
    - SPI
    - JTAG
- 53 programmable I/O pins

The reasons for choosing the ATmega128L as the μC in the Control module:

- It fulfils the requirements listed in *3.3.2.Control Module,* and more, but not by too much.
- It has eight external interrupts; this is more than most low-cost μCs. External interrupts are important in robots, because they decrease the response time to an external event without taking much processing time from the μC.
- It has 48 free programmable I/O pins; more than enough.
- Easy to program and use.
- Many documentations, and support by the large community using it

The ATmega128L is operated at the maximal allowable operating frequency – 8 MHz.

5.2.2 Supervisory Circuit

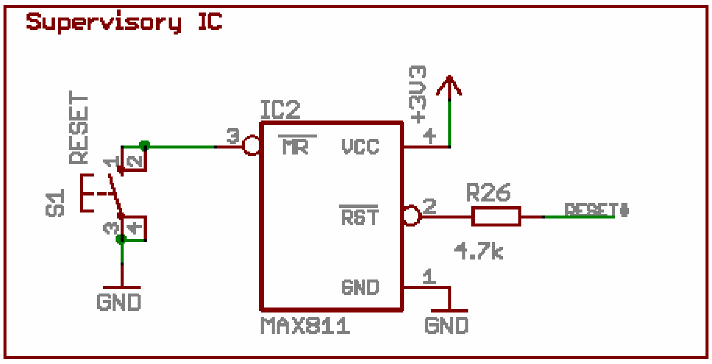


Figure 5.3: Supervisory IC

The function of the supervisory circuit is to reset the µC when the main supply voltage drops below a specific voltage. This is to make sure, that the µC stops executing code when the voltage drops low, as this may cause errors. Although, the µC has an on-chip internal power brown-out detection, it is not as reliable as adding an external supervisory IC. The supervisory IC used here is a *MAX812* by Maxim-IC™. The reset pin goes low when the µC supply power goes below 2.93 V. The µC can also be manually reset by the RESET push button. The resistor R26 (Figure 5.3: **Supervisory IC**) was added to avoid line contention if more than one device tries to drive the reset line.

5.2.3 ADC

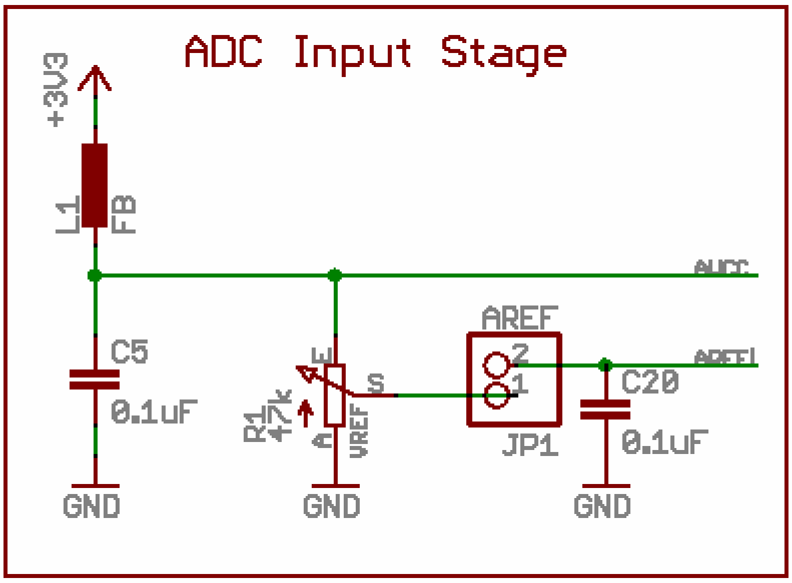


Figure 5.4: ADC input stage

The µC’s on-chip ADC requires a stable supply voltage, AVCC (3.3 V), for the precise conversion of analogue signals to digital. To reduce the noise generated by the digital circuit from passing to the analogue circuit, a low-pass filter consisting of a ferrite bead and a capacitor was added (Figure 5.4). The µC converts the analogue signals by comparing them to a reference voltage, V_{Ref}. To increase the resolution of the A/D conversion, the µC allows V_{Ref} to be set by the user, though V_{Ref} should not exceed AVCC. V_{Ref} can be selected through software as:

- AVCC
- On-chip 2.56V reference voltage
- Through the AREF header (JP1 in Figure 5.4)
    - By applying an external voltage to this header. The VREF potentiometer’s resistance between the slider and GND has to be set to zero, so that the full voltage is applied.
    - By placing a jumper on the AREF header and then adjusting the V_{Ref} using the VREF potentiometer. V_{Ref} can then be set between AVCC(3.3 V) and GND.

5.2.4 Microcontroller Headers

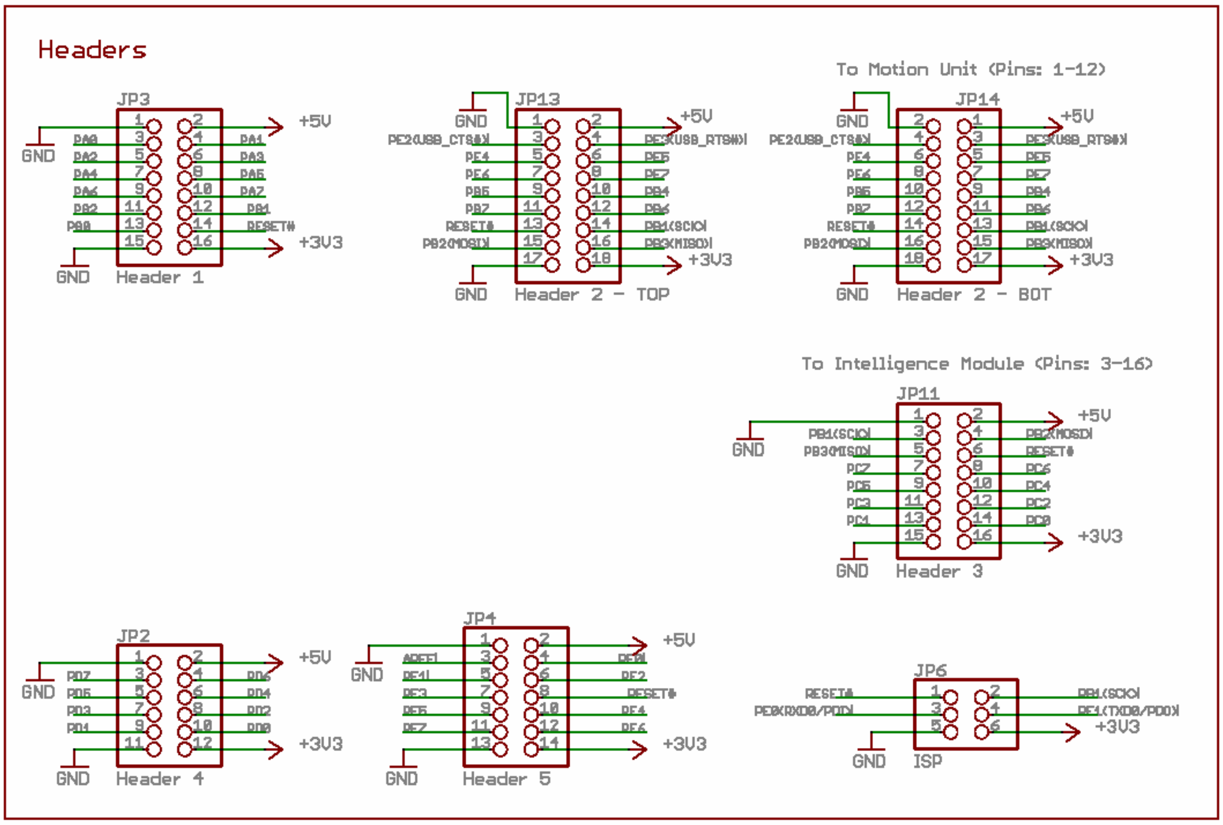


Figure 5.5: The Control module headers

The 48 free programmable I/O pins (from 53 I/O pins in total) of the µC are accessible through the headers in this module. The peripheral I/O pins of the µC are shared with the general-purpose I/O pins, and can be activated/deactivated through software. The pins of the µC are grouped into Ports. The ATmega128L has seven Ports: PortA..G.

All headers have four power pins (3.3 V, 5 V, and two GND), so that expansion modules sensors, etc., are provided with power. All headers have a reset pin with the exception of Header 4 and ISP. Headers 2 and 3 are used for connecting this module with the other modules of MAMoRo; header 2 is used for connecting this module with the motion unit in the Power & Motion module, while header 3 is used for connecting it with the Intelligence module. Header 2 consists of a male header on the top-side and a female header on the bottom-side of the module to allow the Control module to be plugged on the top of the Power & Motion module (see Figure 3.9).

A brief description of the headers on the Control module will follow. A more detailed description of the header pinouts is found in *Appendix B - Headers Pinouts*. An even more detailed description can be found on the µC's datasheet ([ATM06a]).

Header 1

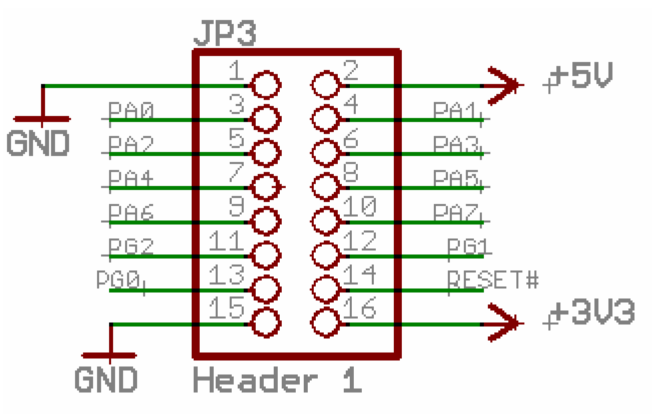


Figure 5.6: Header 1 in the Control module

This header contains all PortA pins and three of PortG pins (PG0..2). The other two pins of PortG (PG3..4) are connected to a 32.768 quartz crystal used for the real-time clock.

Header 2

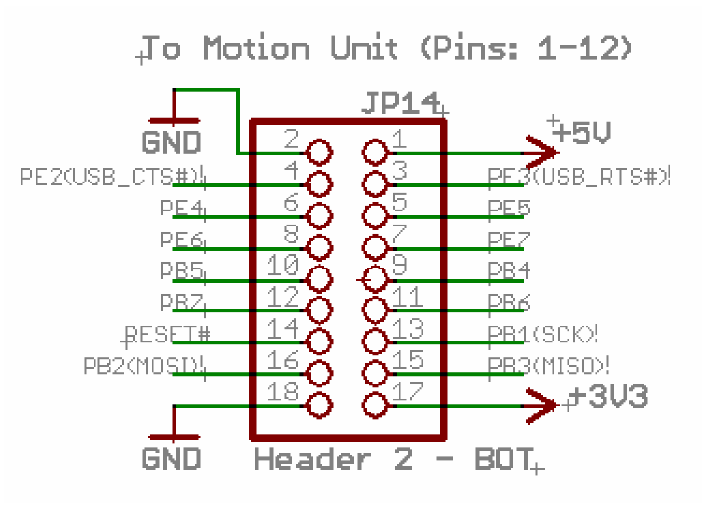


Figure 5.7: Header 2 in the Control module

This header contains all PortB pins with the exception of one pin (PB0), which is used for the flash memory chip select line; and all Port E pins with the exception of two pins (PE0 and PE1), which are used for USB and ISP. Pins 1 to 12 in this header make the interface to the Power & Motion module.

Table 5.1: Interface between the Motion unit and Control module (see also *4.2.1 Motor Driver*)

Motion Unit	**Control Module**
GND	GND
NC*	5 V
Motor 1, Input 1 (M1_IN1)	PE2
Motor 1, Input 2 (M1_IN2)	PE3
Motor 1, Encoder Channel A (M1_A)	PE4 (INT4) - Interrupt on level change
Motor 1, Encoder Channel B (M1_B)	PE5 (INT5) - Interrupt on level change
Motor 2, Encoder Channel A (M2_A)	PE6 (INT6) - Interrupt on level change
Motor 2, Encoder Channel B (M2_B)	PE7 (INT7) - Interrupt on level change
Motor 2, Input 3 (M2_IN3)	PB5
Motor 2, Input 4 (M2_IN4)	PB4
Motor 1, Enable (M1_EN)	PB7 (OC1C/OC2) - PWM
Motor 2, Enable (M2_EN)	PB6 (OC1B) - PWM

*NC: Not Connected

These pins were specifically picked and placed in their actual arrangement to create an optimal interface to the motion unit in the Power & Motion module (see Table 5.1). So, for example, external interrupts and timer/counter pins were placed in certain positions in the header, while pins with special functions which are not needed in this interface were moved away to be used for something else.

In this interface, PE2–3 and PB4–5 are used as general I/O pins. The pins PE4..7 are used as external level-edged interrupts to detect the logic level changes of the signals coming from the quadrature encoder in the motion unit, which are used to calculate speed and direction of the robot. PB6–7 are used for the generation of PWM pulses to control the speed of the motors. By using the dedicated hardware in the µC to generate the interrupts and PWM pulses instead of implementing it in software, processing time is saved, and the µC has more time to do other tasks.

The pins PE2 and PE3 are also used for USB handshaking if a jumper is placed on the header USB HS (Figure 5.14). If the USB handshaking is enabled, controlling the motion unit will not be possible. However, this is not a problem. First, because the USB-USART converter IC (Figure 5.14) is only enabled if a USB cable is plugged and connected to another device, and that usually means the robot has to remain still, hence, no need for the motion unit. Second, the handshaking signals are rarely needed and were never used in this project. Lastly, handshaking can also be implemented in software, therefore sparing these two pins.
There are also three SPI bus pins connected to this header.

Header 3

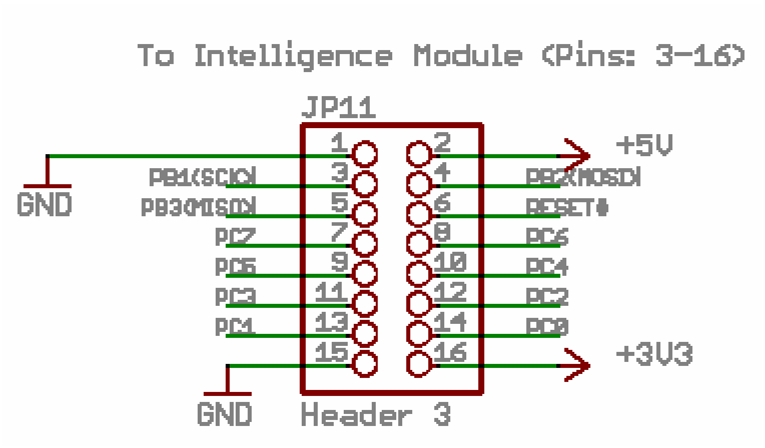


Figure 5.8: Header 3 in the control module

Header 3 contains all PortC pins, the SPI bus pins, and a reset pin. This header is also used as the interface to the Intelligence module (see Table 4.1).

Table 5.2: Interface between the Control and Intelligence module (see also *6.2.3* ***Fehler! Verweisquelle konnte nicht gefunden werden.***).

Control Module	**Intelligence Module**
GND	NC
5 V	NC
PB1/SPI (SCK)	B3_IO6
PB2/SPI (MOSI)	B3_IO7
PB3/SPI (MISO)	B3_IO4
Reset	B3_IO5
PC7	B3_IO2
PC6	B3_IO3
PC5	IO/ CONFIG_INIT_B
PC4	B3_IO1
PC3	CONFIG_PROG_B
PC2	CONFIG_CCLK
PC1	CONFIG_DIN
PC0	CONFIG_DONE
GND	GND
3.3 V	NC

Although, any header can be used as the interface to the intelligence module, header 3 is best suited for this, because PortC pins are not shared by any important peripheral in the µC.

In this interface, pins PC0..3 and PC5 are used for the configuration of the FPGA in the Intelligence module, while the other pins are user-I/O pins. The SPI bus is used for the communication between the µC in the Control module and the FPGA in the Intelligence module. The SPI bus is an optimal and effective choice for the communication between both modules; the reasons are:

- SPI is already integrated in the µC's hardware, so the µC is not burdened with it.

- SPI is fast – it can transfer data with up to the half operating frequency of the µC, i.e. 4 MHz.
- It is a serial bus and needs only 4 lines (MOSI, MISO, SCK, and SS).

Header 4

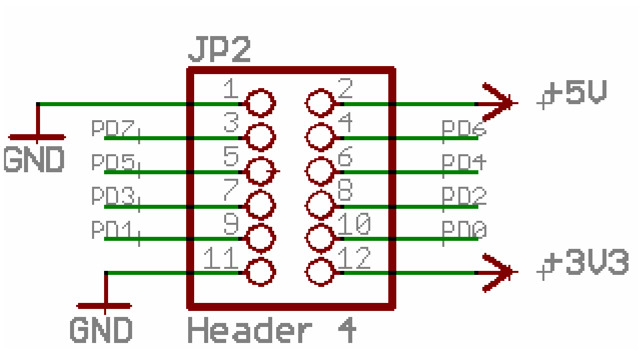


Figure 5.9: Header 4 in the Control module

Header 4 contains all PortD pins (PD0..7). PortD pins are shared with some useful peripherals such as I^{2}C bus and USART.

Header 5

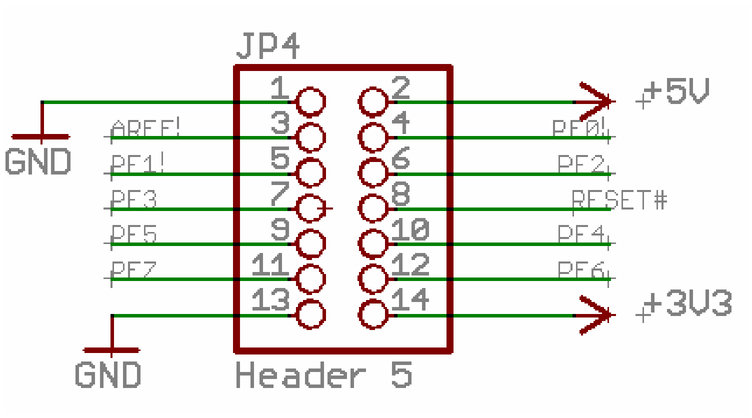


Figure 5.10: Header 5 in the control module

Header 5 contains all PortF pins (PF0..7), the ADC voltage reference pin AREF, and a reset pin. The eight ADC channels are accessed through the pins of PortF. The pins PF0 and PF1 are shared with two user push buttons; they are protected from accidental short-cuts by a resistor (Figure 5.11).

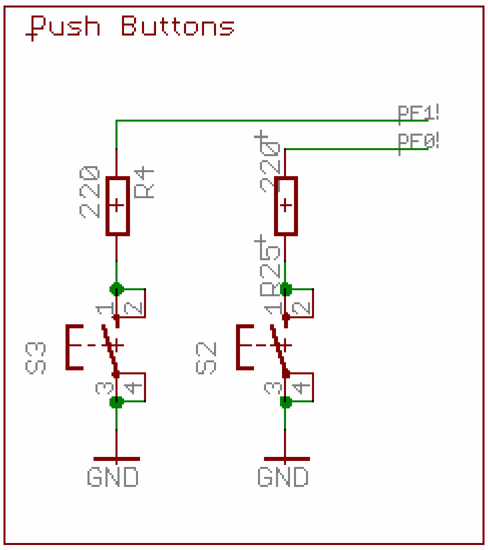


Figure 5.11: User push buttons in the Control Module

A voltage can be applied at the AREF pin to act as the reference voltage, V_{Ref}, for the ADC (see *5.2.3ADC)*

Besides the ADC, pins PF4-7 are also shared with the JTAG pins (TDI, TDO, TMS, TCK).

ISP

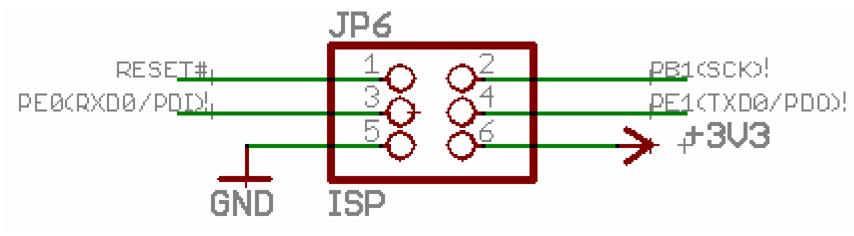


Figure 5.12: ISP header in the Control module

The In-System Programmable (ISP) header is used for the first programming of the µC. Through the ISP, basic configuration fuses of µC are set and a bootloader software is downloaded to the µC (see *Chapter 8*), which allows the µC to be programmed later through USB.

5.3 Flash Memory

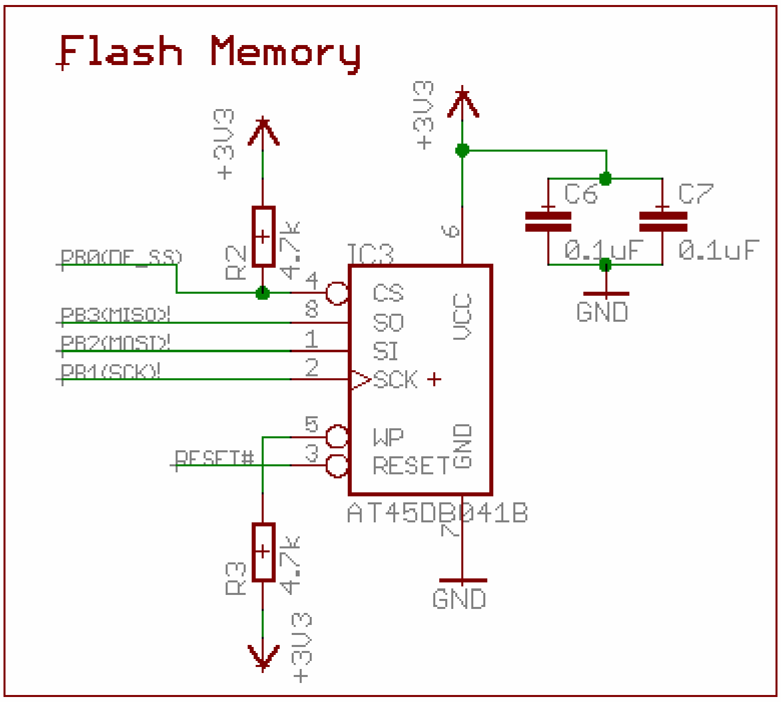


Figure 5.13: Flash memory in the Control module

The flash memory used in this project is a 512 KiByte *AT45DB041B* DataFlash™ by Atmel™. The flash memory is used for saving non-volatile data such as the FPGA configuration file. 512 KiByte is big enough to store two configurations files for the FPGA implemented in the Intelligence module, thus making reconfigurable hardware experiments possible, while still having 100 KiByte to store other things. The µC communicates with the flash memory through the SPI bus. This is very convenient, because:

- Only one pin is wasted for the flash memory—the chip select line.
- The SPI is already integrated on the µC's hardware, so the µC is not burdened with it
- Expandability – The flash memory interface and pins always stays the same no matter what the memory size is. So, if in the future a bigger flash memory is needed (e.g. to store more than two FPGA configuration files), then the current one can be de-soldered (or unplugged if an IC socket is used) and simply be replaced by a bigger flash memory.

5.4 USB Interface

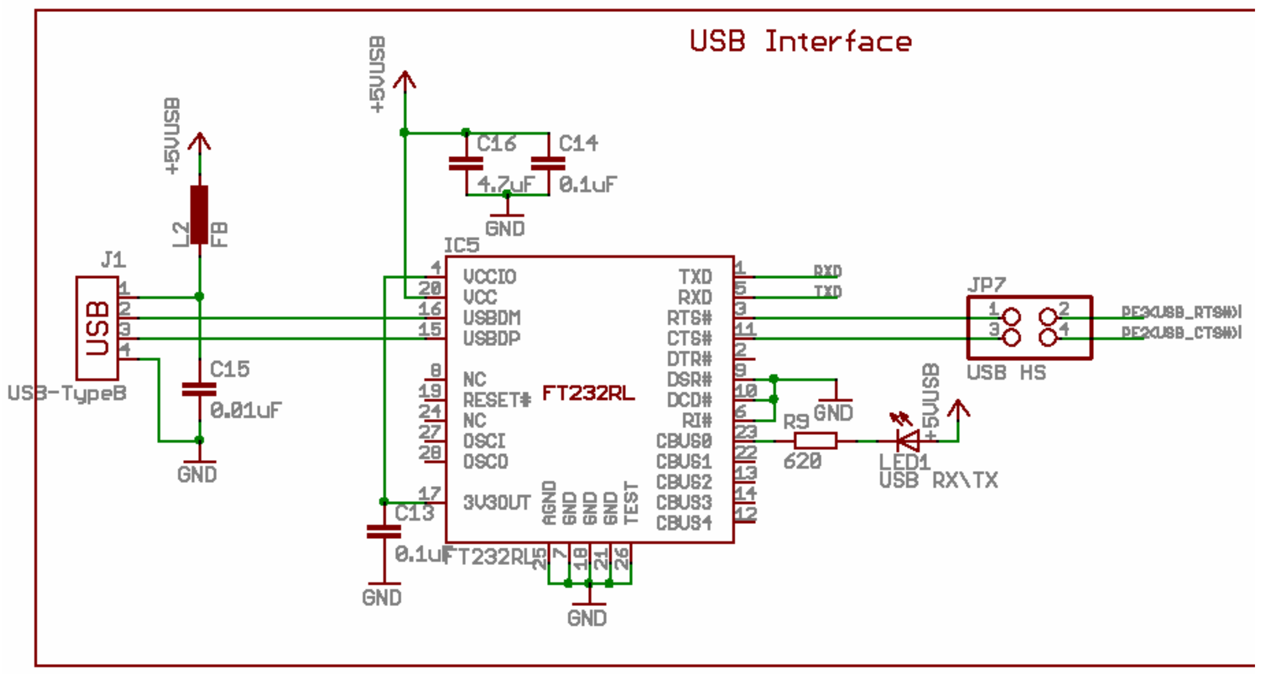


Figure 5.14: USB interface in the Control module

The ATmega128L has an USART port (same thing as UART but can communicate also synchronously), and by adding a RS-232 driver IC (e.g. MAX3221 from Maxim™), the µC will be able to communicate with the PC through the standard serial port (RS-232 standard complaint). However, the problem is that the serial port is not found on most of today's PC's, and especially not on notebooks. The USB interface, on the other hand, is widely used. The µC we picked, the ATmega128L, does not have an on-chip USB port. This was overcome by using a USART-USB converter IC which converts USART signals to USB and vice versa; hence, the communication of the µC through the widely used USB interface is possible now. The USART-USB converter IC used in this project is *FT232RL* by FTDI™.

A question would arise: why not use instead a µC with an on-chip USB port instead of using an additional UART-USB converter IC? This would save cost and PCB area. The reason is that programming the USB interface is too complex compared to the standard serial port. The UART-USB converter FT232RL comes with device drivers that can emulate the USB interface in the PC as a serial port. So, although the USB bus is physically used, logically the communication is done as though the µC is connected to a serial port, and the simpler serial port commands can be used.

The FT232RL acts basically as a gateway between the µC and the USB bus (see Figure 3.7 and Figure 5.14). The USART unit in the µC sends the data to the FT232RL through the TXD (transmit data) pin, and receives the converted data through the RXD (Receive Data) pin.

The header USB HS (Figure 5.14) sets whether handshaking should be used or not. When a jumper is placed on USB HS, handshaking is activated. Handshaking is rarely need and was never used during this project. The lines PE2 and PE3 are also shared by header 2 (see *5.2.4 Microcontroller Headers*). Therefore, before using it for handshaking, it must be made sure that it is not used somewhere else.

The µC lines that are used for the USB interface are PE0 and PE1. These pins are also used for ISP, which is also used for the first programming of the µC (see *Chapter 8*). To avoid line driving conflicts between USB and ISP, a 3-state buffer was used (Figure 5.15).

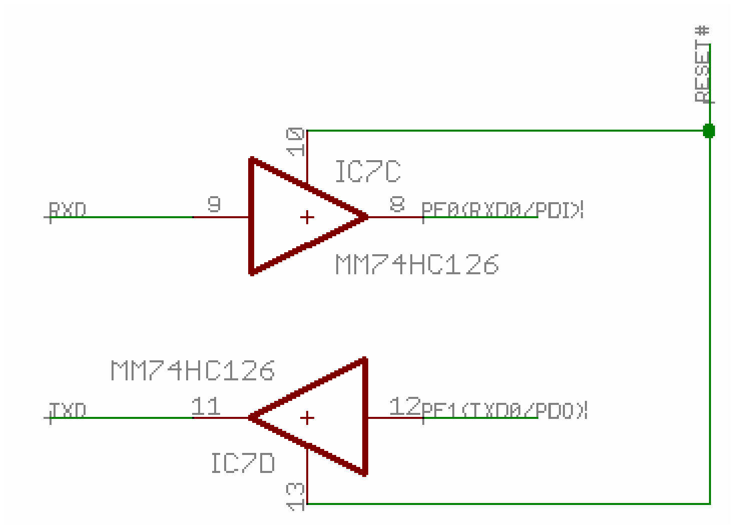


Figure 5.15: A 3-state buffer used between the µC and the FT232RL

It works because the reset line goes low (active-low) when the µC enters the programming mode in ISP. The reset line is also connected to the high-active enable pin for both buffers. Therefore, when the µC enters the programming mode in ISP, the reset goes low, making the buffers behave like a high-impedance, thus the FT232RL is effectively disconnected from the µC.

To communicate with the Control module from the PC, the device drivers found on the manufactures website[1] have to be downloaded, installed, and configured. These drivers

[1] www.ftdichip.com

emulate the USB interface as a serial port on the PC, and any software can access it as if it was a serial port.

The LED USB RX\TX in Figure 5.14 blinks when the FT232RL receives or sends data. But before it does that, this function has to be programmed in the IC using the device drivers installed in the PC.

5.5 Specifications

Control module main hardware specifications:

- Microcontroller: low-power, 8-bit RISC ATmega128L by Atmel™; 8 MHz with 1 MIPS/MHz. Main features:
  - Program and Data Memories
    - 128 KiByte In-System Programmable (ISP) flash memory
    - 4 KiByte EEPROM
    - 4 KiByte SRAM
  - 4 x Timer/Counters
  - Real time counter (RTC)
  - 2 x 8-bit PWM channels and 6 x PWM channels with programmable resolution from 2 to 16 bits
  - 8 x 10-bit ADC with optional differential input stage with programmable gain
  - Two-wire Serial Interface (TWI), equivalent to I^2C
  - 2 x USART
  - SPI
  - JTAG
- 48 free user-I/O pins; accessible through 6 headers.
- On-board power supply; enables the Control module to operate in stand-alone as an evaluation platform to conduct a wide variety of experiments in electrical engineering and computer science.
- External 512 KiByte flash memory
- USB interface
- 2 x user push buttons

6 Intelligence Module

The specifications for the Intelligence module were laid down in *3.2.2 Decision unit* and *3.3.3 Intelligence Module*. In this chapter, the actual hardware realization is described.

The Intelligence module is the seat of the most powerful processing unit in MAMoRo. While the Control module is busy with low-level hardware tasks, this module does the "real" thinking. With this module, advanced robotic applications requiring intensive processing such as image recognition, and machine learning are possible. The programmable processing unit in this module is an FPGA.

When used as stand-alone, the Intelligence module can be used as an evaluation platform to conduct a wide variety of experiments in electrical engineering and computer science.

The complete schematics, board layout, and parts list for this module are in *Appendix A - Schematics, Board Layouts, and Parts List*. While description for the header pinouts of this module are in *Appendix B - Headers Pinouts.*

6.1 Power Unit

The 5 V input voltage of this module comes from the Power & Motion module through the header *Modules PWR* (Figure 6.1). If the Intelligence module is used without the Power & Motion module, the input power source should be applied through the DC jack (see Figure 6.1). The DC jack input is protected from reveres-polarity by a diode. The input voltage at the DC jack can be anything from 4.5 V to 27 V. However, if there are devices that need 5 V while the Intelligence module is used in stand-alone mode, the input voltage applied at the DC jack has to be approximately 5 V. A power indicator LED was implemented that glows when ever a power supply is applied at the input.

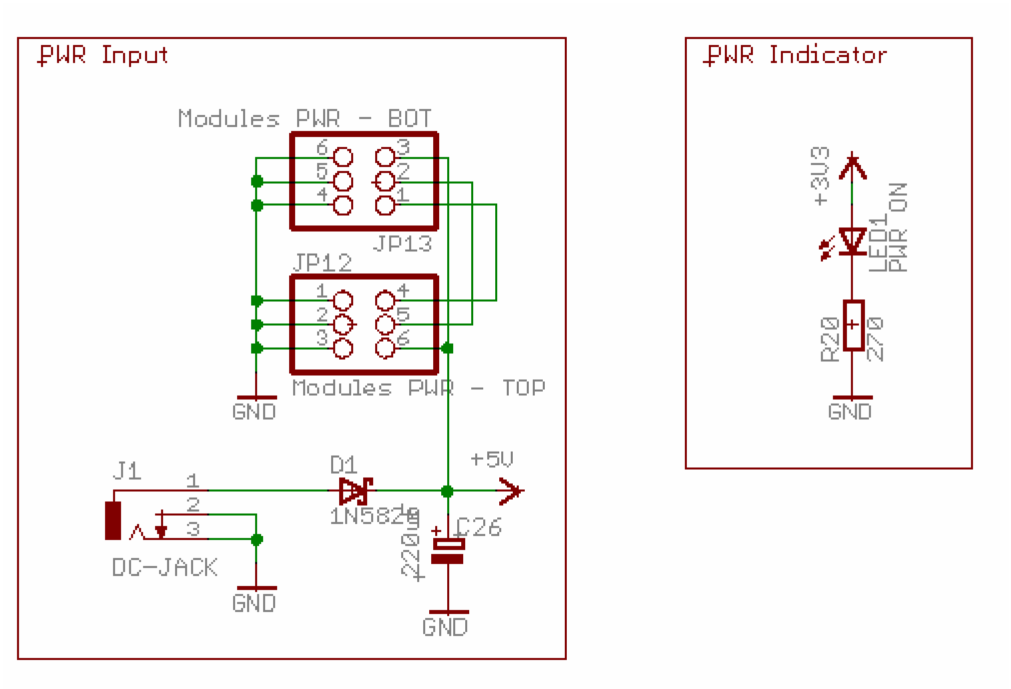


Figure 6.1: Power input in the Intelligence module

The FPGA needs three different power supplies: 3.3 V, 2.5 V, and a 1.2 V. These power supplies are provided by three linear regulators (Figure 6.2).

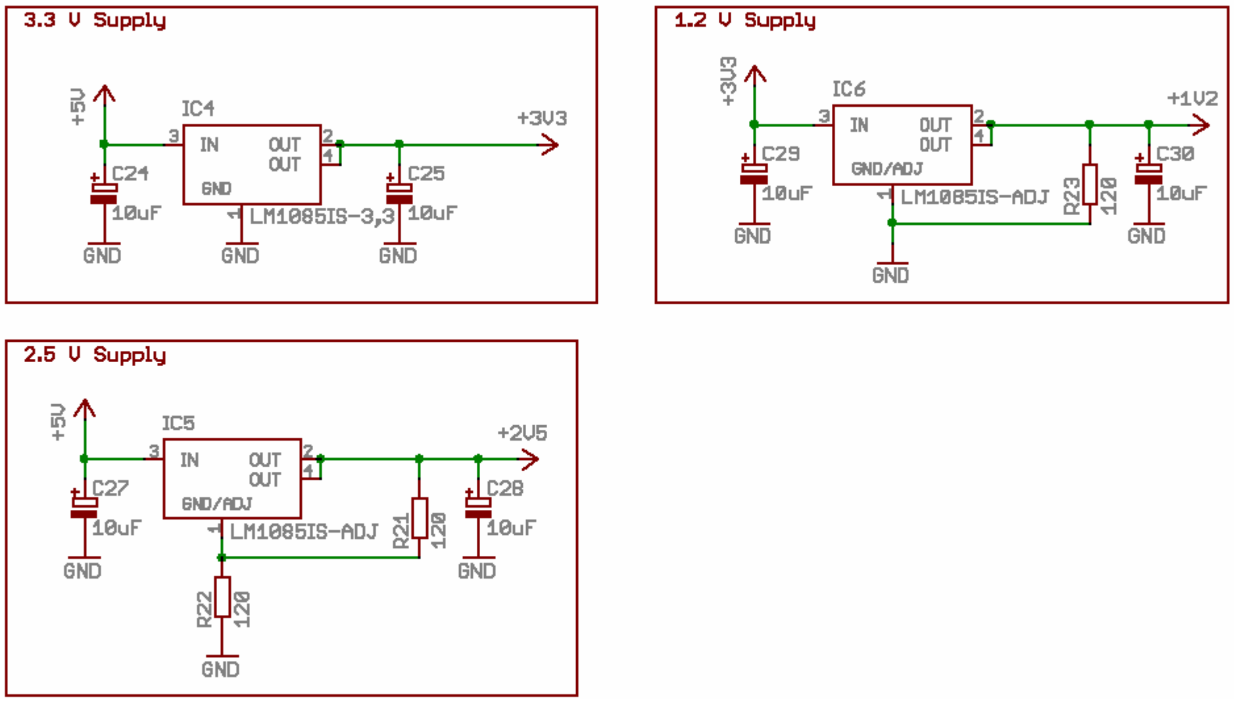


Figure 6.2: 3.3 V, 2.5 V, and 1.2 V power supplies in the Intelligence module

These power supplies were implemented in the Intelligence module rather than the Power & Motion module, so that the intelligence module can be used as stand-alone. In stand-alone mode, the Intelligence module can be used as an evaluation platform to conduct a wide variety of experiments in electrical engineering and computer science.

6.2 FPGA Unit

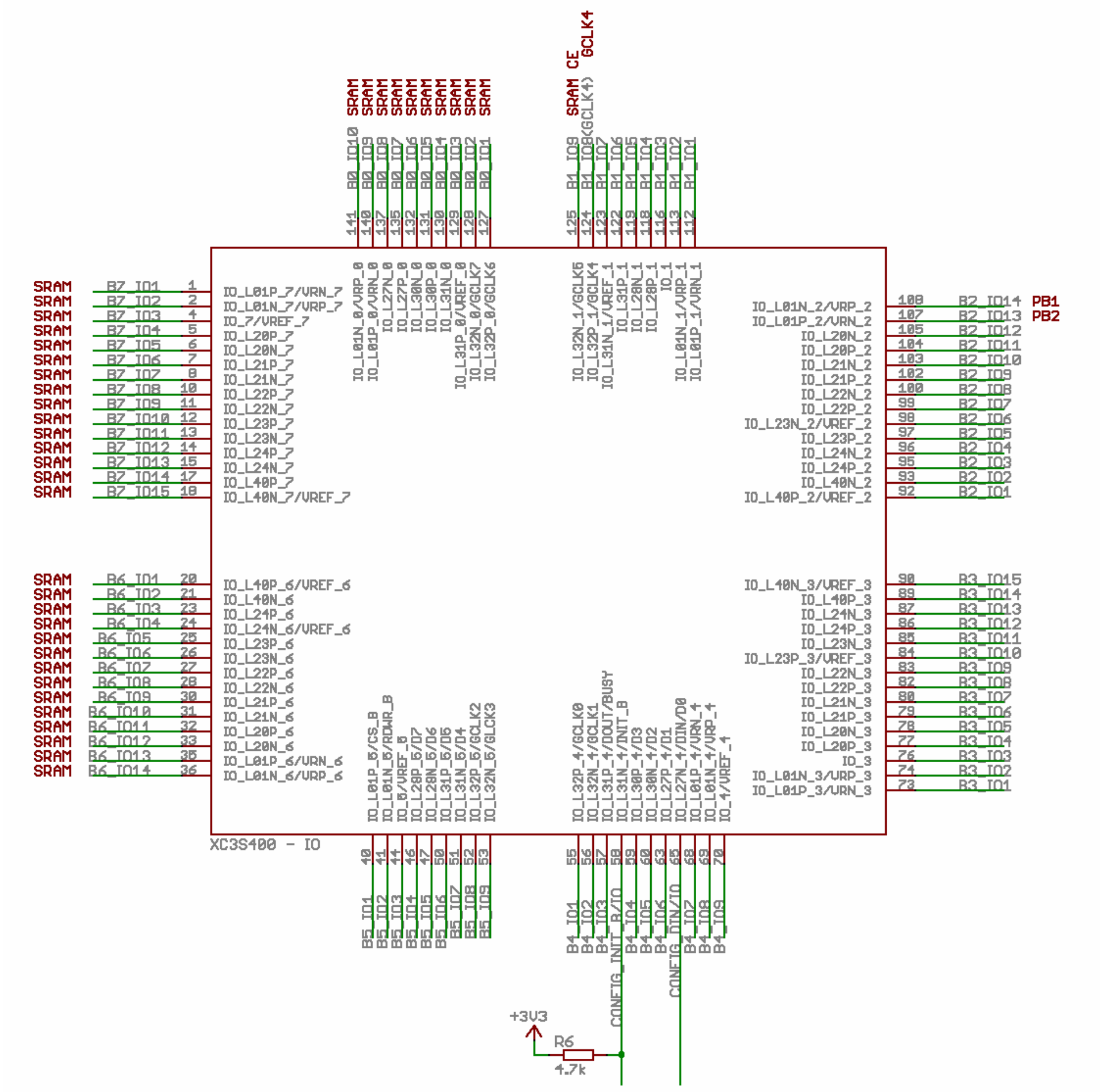


Figure 6.3: Spartan-3 XC3S400-TQ144

The FPGA is the main component of the Intelligence module and the programmable unit with the highest processing performance in MAMoRo. The FPGA chosen in this project is a low-cost *Spartan-3 XC3S400* by FPGA Xilinx^{TM}. This FPGA has 8064 logic cells. That is enough to store five times (area optimized) MicroBlaze™ 32-bit soft processor by Xilinx™ [XIL07c] or two times (area optimized) the open source soft processors OpenRISC or LEON2 [MAC04]. There are ten different packages for this FPGA. The package TQ144 was particularly chosen, because it can be soldered by hand and has sufficient I/O pins.

6.2.1 FPGA Overview

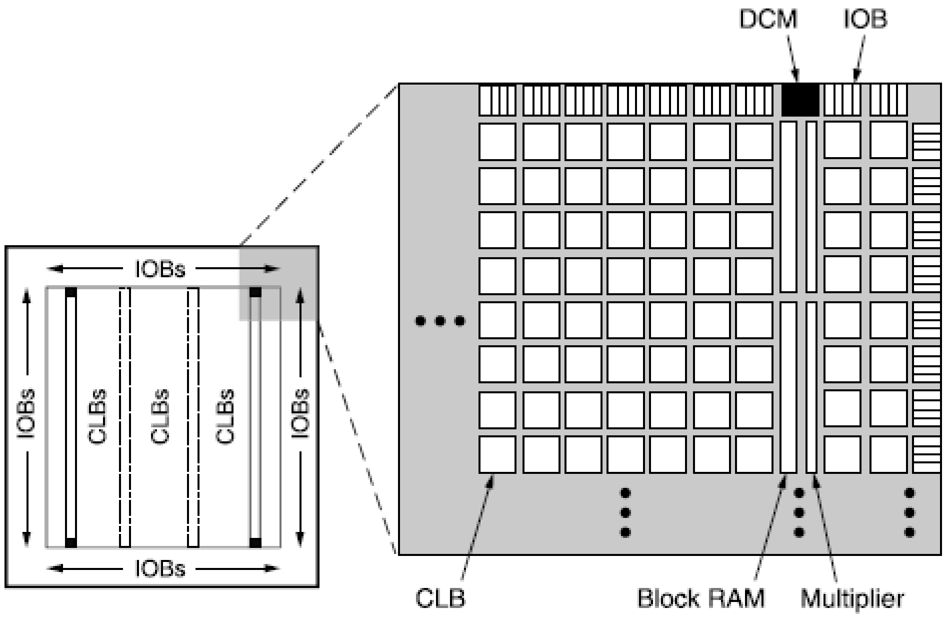


Figure 6.4: The five basic elements of the FPGA: CLB, Block RAM, Multiplier, DCM, and IOB [XIL06a].

The FPGA’s architecture consists basically of five programmable elements (Figure 6.4) [XIL06a]:

- *Configurable Logic Blocks* (CLB): contains *RAM-based Look-Up Tables* (LUT) to implement logic and storage elements that can be used as flip-flops or latches. CLBs can be programmed to perform a wide variety of logical functions as well as to store data.
- *Input/Output Blocks* (IOB): control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. The *Digitally Controlled Impedance* (DCI) block, allows the on-chip addition of termination resistors to prevent reflections on the signal.
- *Block RAM*: provides data storage in the form of 18-kbit dual-port blocks.
- *Multiplier blocks*: can calculate the product of two 18-bit binary numbers.

- *Digital Clock Manager* (DCM): provides self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase shifting of clock signals.

The IOBs of the FPGA are distributed into eight I/O banks (Figure 6.5); two banks on each side of the FPGA. Each I/O bank has it own power supply pins, VCCIO. In the TQ144 package, the two banks on each side of the FPGA share the same VCCIO power line.

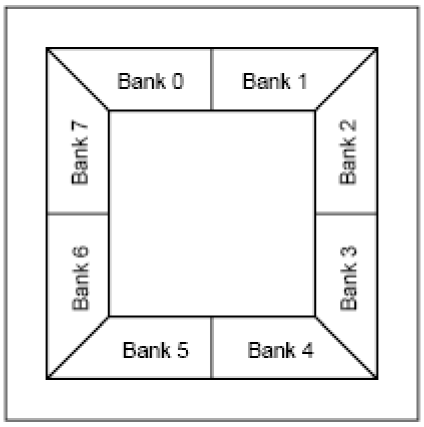


Figure 6.5: Spartan-3 I/O banks (top view) [XIL06a]

The FPGA has different types of pins. Some are specialized and have a specific function, therefore, can not be used as user I/O pins. Nevertheless, most of the pins are general purpose user I/O pins, though some have a dual purpose.

The specialized pins are:

- CONFIG: these pins are used for the configuration of the FPGA in some configuration modes (see *6.2.3 FPGA Configuration*).
- JTAG: used for the JTAG interface. JTAG can be used for configuration and debugging of the FPGA.
- Power pins:
  - VCCO
  - CCAUX
  - VCCINT
  - GND

User I/O pins, most of them are dual purpose:

- General-purpose user-I/O: these pins have no dual purpose and are solely used as user-I/O.
- DCI: dual-purpose pins that can either be user-I/O pins, or be used to calibrate the output buffer impedance using the on-chip digital controlled impedance (DCI) block.
- VREF: dual-purpose pins that can either be user-I/O pin, or voltage reference input pins used as a reference voltage for a specific I/O standard in that particular bank. If this alternate function of the pin was used, all VREF banks within a bank must be connected together.
- GCLK: dual-purpose pins that can either be user-I/O pin, or an input to a specific global clock buffer input.
- DUAL: dual-purpose pins that are used in some configuration modes for the configuration of the FPGA. After the configuration is finished, they are become user-I/O pins.

6.2.2 FPGA Power

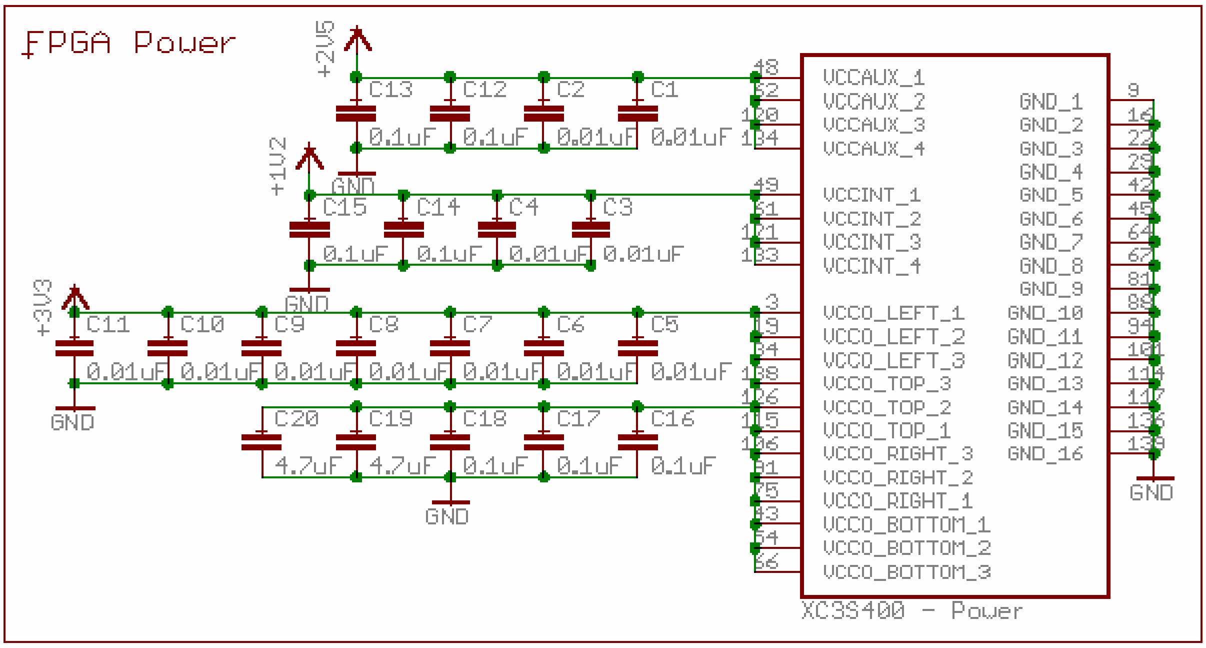


Figure 6.6: FPGA power pins

The FPGA needs three different powers supplies (1.2 V, 2.5 V, and 3.3 V) and has 36 power pins. The number and value of bypass/decoupling capacitors added to the power pins was according to the application note [XIL05]. The function of these capacitors is to offer transient current demands, which the main power supply is to slow for, and filter noise in the power supply line.

6.2.3 FPGA Configuration

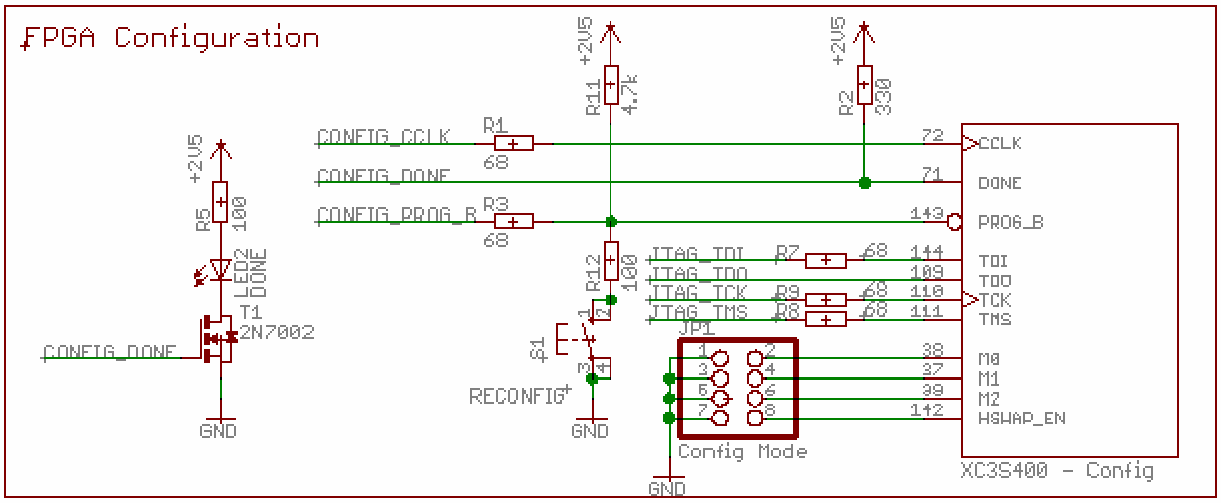


Figure 6.7: FPGA configuration pins

The FPGA can be configured through five different configuration modes:

- Master/Slave Serial
- Master/Slave Parallel
- JTAG

The pins responsible for setting the configuration mode are M0, M1, and M2. The configuration mode is set by placing jumpers on the header *Config Mode* (see Figure 6.7 and Table 6.1). The HSWAP_EN pin in the Config Mode header enables/disables the internal pull-up resistors on all unused I/O pins during the configuration process. If HSWAP_EN is connected to GND by a jumper, the pull-up resistors are enabled. After the configuration of the FPGA is done, this pin has no function.

Table 6.1: FPGA configuration mode settings

Configuration Mode	M2	M1	M0
Master Serial	0	0	0
Slave Serial	1	1	1
Master Parallel	0	1	1
Slave Parallel	1	1	0
JTAG	1	0	1
X	X	X	X

When the Intelligence module is used as stand-alone, using the JTAG interface for configuration is a good choice, because it can then be used for configuration and debugging at the same time.

In MAMoRo, the FPGA in the Intelligence module is configured by the Control module. First, the configuration file is downloaded from the PC and stored in the flash memory located in the Control module. Then, the µC the Control module reads the data from the flash memory and configures the FPGA. Thus, the µC acts as the Master and generates the configuration clock (CCLK), and controls the configuration process. The Control module interface with the Intelligence module is the header *Bank 3 (Config)-BOT*, placed on the bottom-side of the Intelligence module (Figure 6.8).

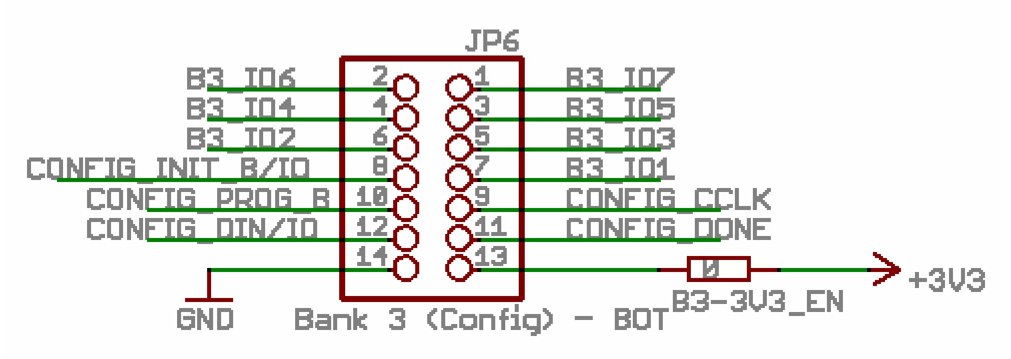


Figure 6.8: The header: *Bank 3(Config)* – the Intelligence module interface to the Control module.

The resistor B3-3V3_EN in Figure 6.8 is in fact not a resistor, and pin 13 is not connected to anything. The resistor was drawn, so that if the 3.3 V supply needed to be connected to pin 13, a wire can be soldered between the pads of the resistor.

The connections between the Intelligence module and the Control module are shown in Table 6.2.

Table 6.2: The interface between the Control and Intelligence module (see also *5.2.4 Microcontroller Headers*)

Control Module	**Intelligence Module**
GND	NC
5 V	NC
PB1/SPI (SCK)	B3_IO6
PB2/SPI (MOSI)	B3_IO7
PB3/SPI (MISO)	B3_IO4
Reset	B3_IO5
PC7	B3_IO2
PC6	B3_IO3
PC5	IO/ CONFIG_INIT_B
PC4	B3_IO1
PC3	CONFIG_PROG_B
PC2	CONFIG_CCLK
PC1	CONFIG_DIN
PC0	CONFIG_DONE
GND	GND
3.3 V	NC

The FPGA is configured in slave serial mode, although other configuration modes are also possible. Slave serial mode was chosen because it is simple and requires only two user-I/O pins (INIT_B and DIN; see Figure 6.3) in addition to the control configuration pins (PROG_B, DONE, and INIT_B). Slave parallel mode, on the other hand, requires 11 pins (Din[0:7], Busy, CS_B, RDWR_B). However, the advantage of parallel mode is that it is faster, since the data is shifted out byte-wise, rather than bit-wise in serial mode. Particularly in reconfigurable hardware applications, where the time it takes for an FPGA to get configured is crucial, parallel mode can be used instead in MAMoRo.

After the configuration of the FPGA is successfully completed, this will be indicated visually by turning the DONE LED on (Figure 6.7). The configuration process of the FPGA can be restarted all over again by pushing the *ReConfig* push button (Figure 6.7). After the configuration process is done, the µC in the Control module could communicate with the FPGA through the serial bus SPI (see *5.2.4 Microcontroller Headers*). But before that, the FPGA has to be loaded with an SPI core, which can be downloaded, for example, from the popular hardware open source website: www.opencores.org.

Detailed information regarding the configuration of Spartan-3 FPGA's can be read in [XIL07a].

6.3 Oscillator

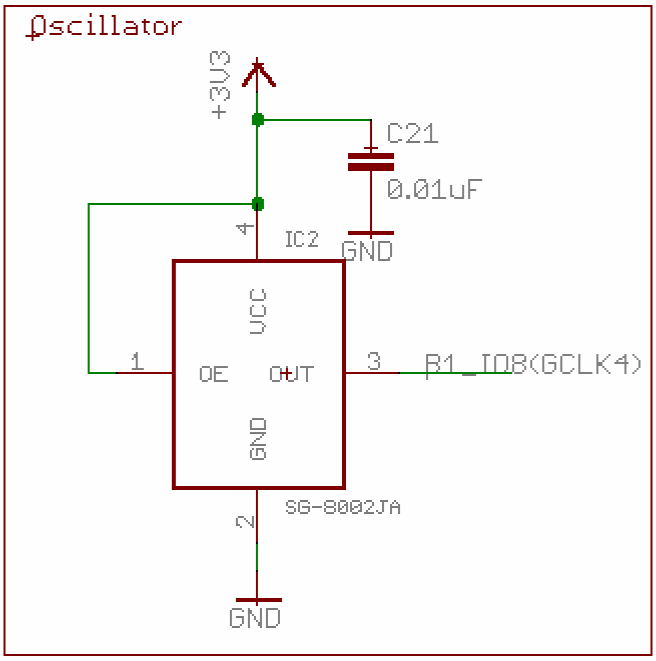


Figure 6.9: Oscillator IC in the Intelligence module

The Oscillator IC has the function of generating the input clock signal for the FPGA. The Oscillator output is connected to a global clock dedicated pin (GCLK4), which is connected to a Digital Clock Manager (DCM) integrated in the FPGA. The on-chip DCM has three main functions: Clock-skew elimination, phase shifting, and frequency synthesis. Provided with a clock signal on any of the eight global clock dedicated pins (GCLK0-7), the DCM can generate a wide range of clock frequencies by multiplying or dividing the input clock signal by definite factors. The possible clock frequencies range are 18 MHz to 307 MHz.

So, if operated at the maximal allowable clock frequency (i.e. 307 MHz), the FPGA runs approximately 38 times faster than the µC in the Control module.

6.4 SRAM

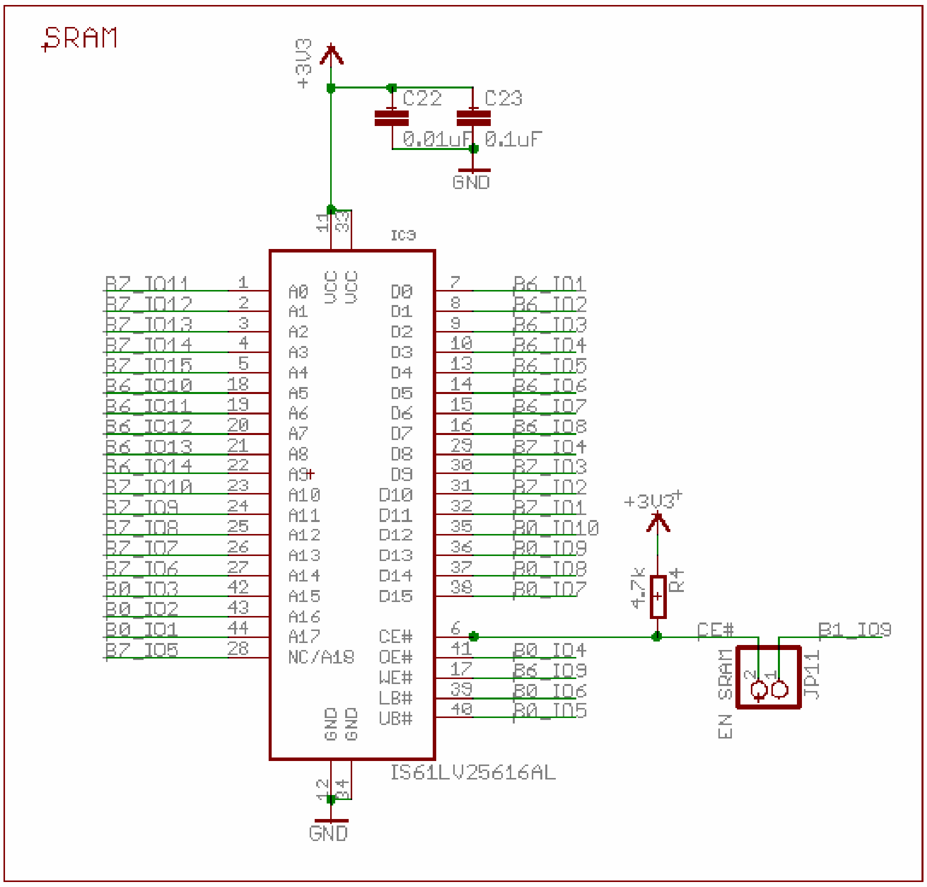


Figure 6.10: 256K x 16 SRAM in the Intelligence module

The Intelligence module has a 256K x 16 SRAM with a high speed access time of 10ns. The SRAM used here is *IS61LV25616AL* by ISSI™. The SRAM data and address pins are connected to the FPGA. The SRAM is used in combination with the FPGA for data storage in various applications. For example, a soft core processor can be implemented on the FPGA, and this accesses the SRAM for data storage. Another application is in graphic processing, where a lot of data is processed and a fast and large data buffer is crucial.

The FPGA has 39 I/O pins shared between the SRAM and three headers (Bank 7,6, and 0/1; see *Appendix A - Schematics, Board Layouts, and Parts List*). If the SRAM is not needed in an application, the I/O pins of the SRAM can be floated by connecting a jumper on the

SRAM EN header (Figure 6.10). Thus, the 39 I/O pins can then be freely used as user-I/O pins.

6.5 FPGA Headers

The TQ144 package has 144 pins. Among these pins, 36 pins are for power, 7 pins for the FPGA configuration, and 4 pins for JTAG. The left 97 pins are general-purpose user-I/O pins. Further, one pin is used as clock input for the FPGA; this leaves 96 user-I/O pins that are accessible through the pin headers. Some of the user-I/O pins are shared with something else. For example, two pins on the header *Bank 2* are shared with two push buttons.
The description of the header pinouts can be read in *Appendix B - Headers Pinouts*.

6.6 Specifications

Intelligence module main hardware specifications:

- Spartan-3 XC3S400-TQ144 low-cost FPGA:
    - 8064 logic cells: capable of storing five times (area optimized) the 32-bit soft processor MicroBlaze™ by Xilinx™ [XIL07c]
    - Possible clock frequencies range: 18 MHz to 307 MHz
- 96 free user-I/O pins, accessible through 7 headers .
- On-board power supply; enables the Intelligence module to operate in stand-alone as an evaluation platform to conduct a wide variety of experiments in electrical engineering and computer science.
- External 256 KiByte SRAM
- 2 x user push buttons

7 Hardware Implementation of MAMoRo

After laying the specifications for MAMoRo's modules in *Chapter 3*, and the hardware design in *chapters 4 to 6*, finally, MAMoRo prototype can be built. Using this prototype, the proof of concept and functionality of the architecture and hardware design of MAMoRo can be proven.

The complete schematics and board layout for this module are in *Appendix A - Schematics, Board Layouts, and Parts List*, and in the CD attached with this thesis.

7.1 PCB Layout

The Board or PCB (Printed Circuit Board) layouts for the modules were made out of the schematics found in *Appendix A - Schematics, Board Layouts, and Parts List.* Since this is a prototype, making the PCB area small for this project not important. In fact a small PCB was not favoured, so that if there were any mistakes in the hardware design, there would be enough space in the board to allow for corrections, for example, adding an capacitor, or swapping an IC without damaging the components around it.

The Schematics and PCBs were designed using the software *EAGLE Layout Editor*[1] (the EAGLE files for the modules are included in the CD attached with this thesis). Afterwards, the PCB data files were sent to a PCB manufacturer for production. The boards worked on the first try as intended with no problems.

The requirements listed in section *3.5 The Physical Implementation* were considered in the PCB layout design.

The following Figure 7.1 to Figure 7.6, depict the modules of MAMoRo after their implementation. The area size of all modules are the same: 8.5 cm x 13.5 cm; the height of the modules stack is 5 cm (see Figure 7.7).

[1] EAGLE Layout Editor: http://www.cadsoft.de/

Notice that four headers on the Control module lay exactly beneath four headers on the Intelligence module: header 2 on the Control module lies beneath Bank 5/4 header on the Intelligence module; header 3 beneath Bank 3(Config) header; header 1 beneath Bank 2 header; and header 5 beneath Bank 0 header. This was done according to the requirements laid in section *3.5 The Physical Implementation*.

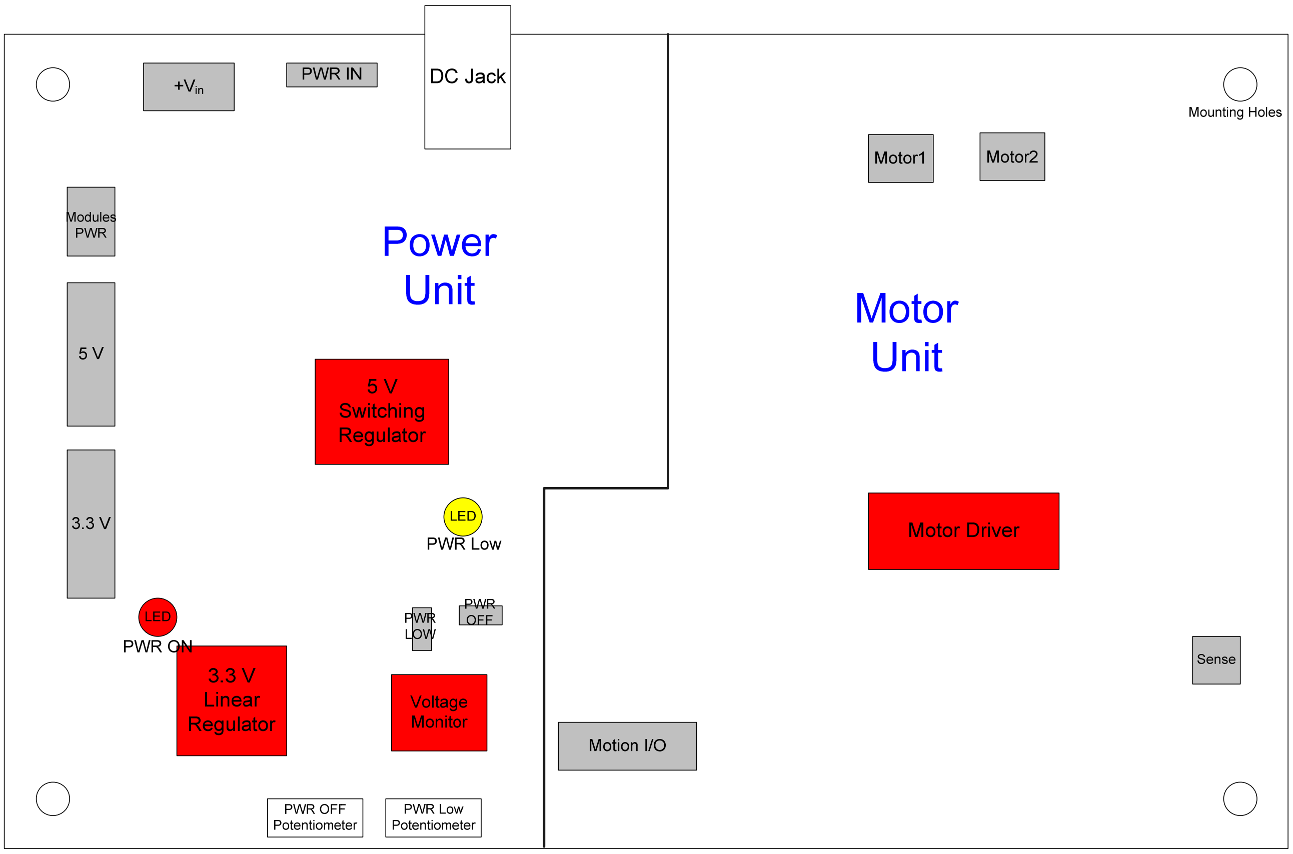


Figure 7.1: Power & Motion module board layout. The rectangles according to their colour are: red for ICs; grey for headers; white for anything else.

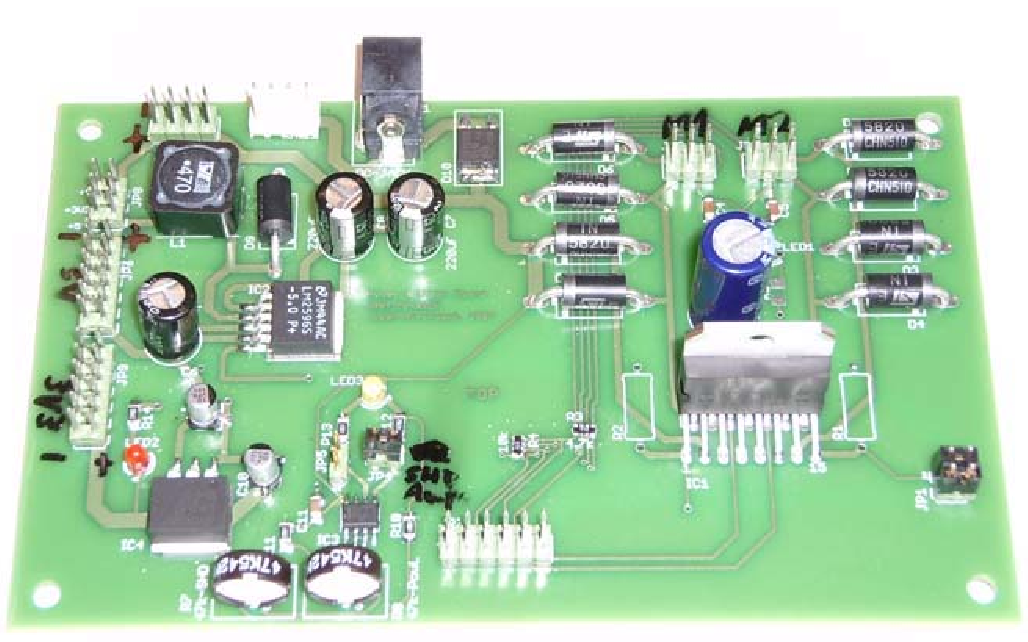


Figure 7.2: Power & Motion module

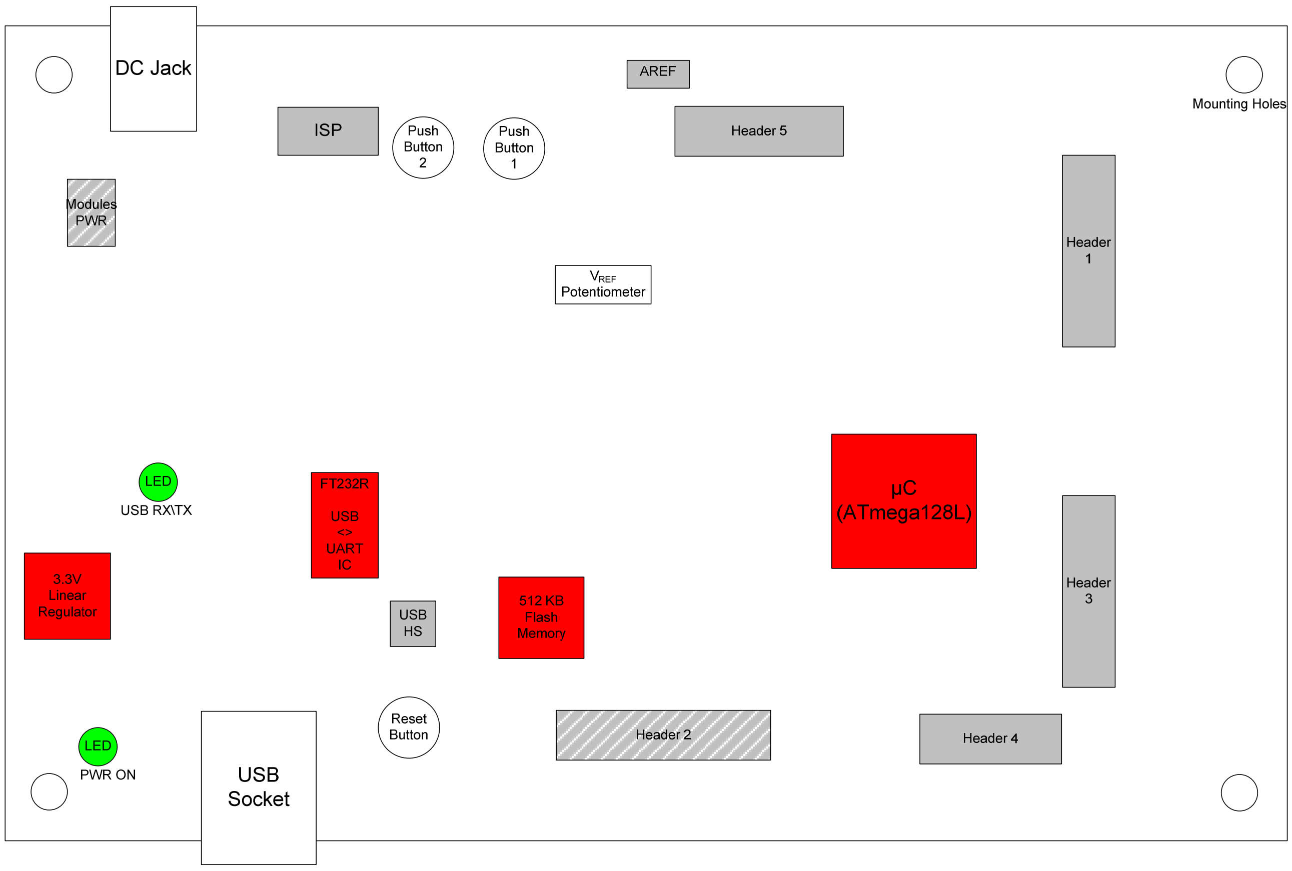


Figure 7.3: Control module board layout. The rectangles according to their colour are: red for ICs; grey for pin headers; grey with white strips for headers that are placed on both sides of the board (male header on the top, female header on the bottom); white for anything else.

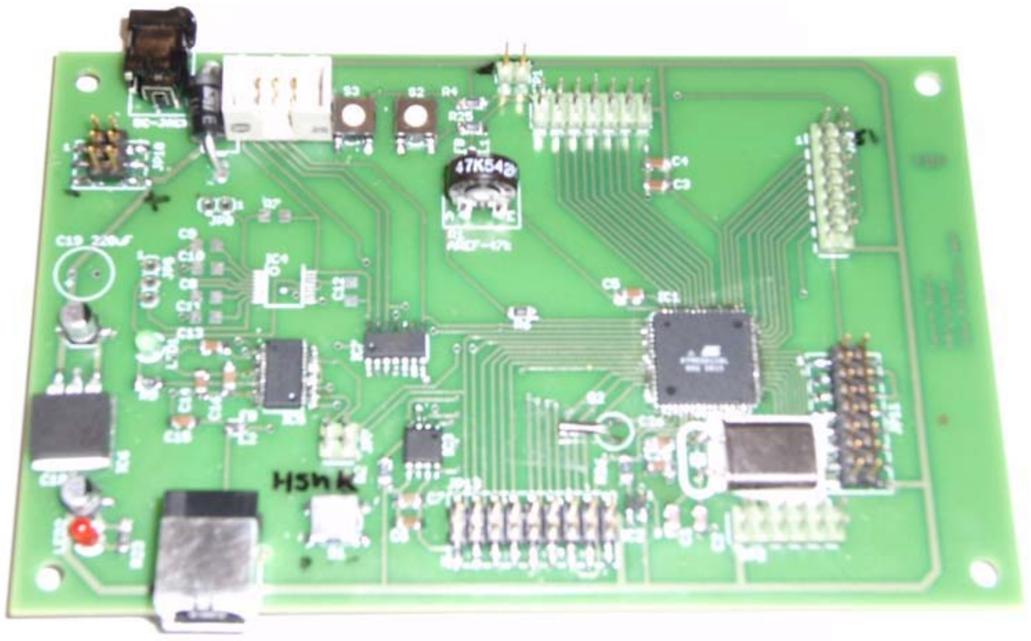


Figure 7.4: Control module

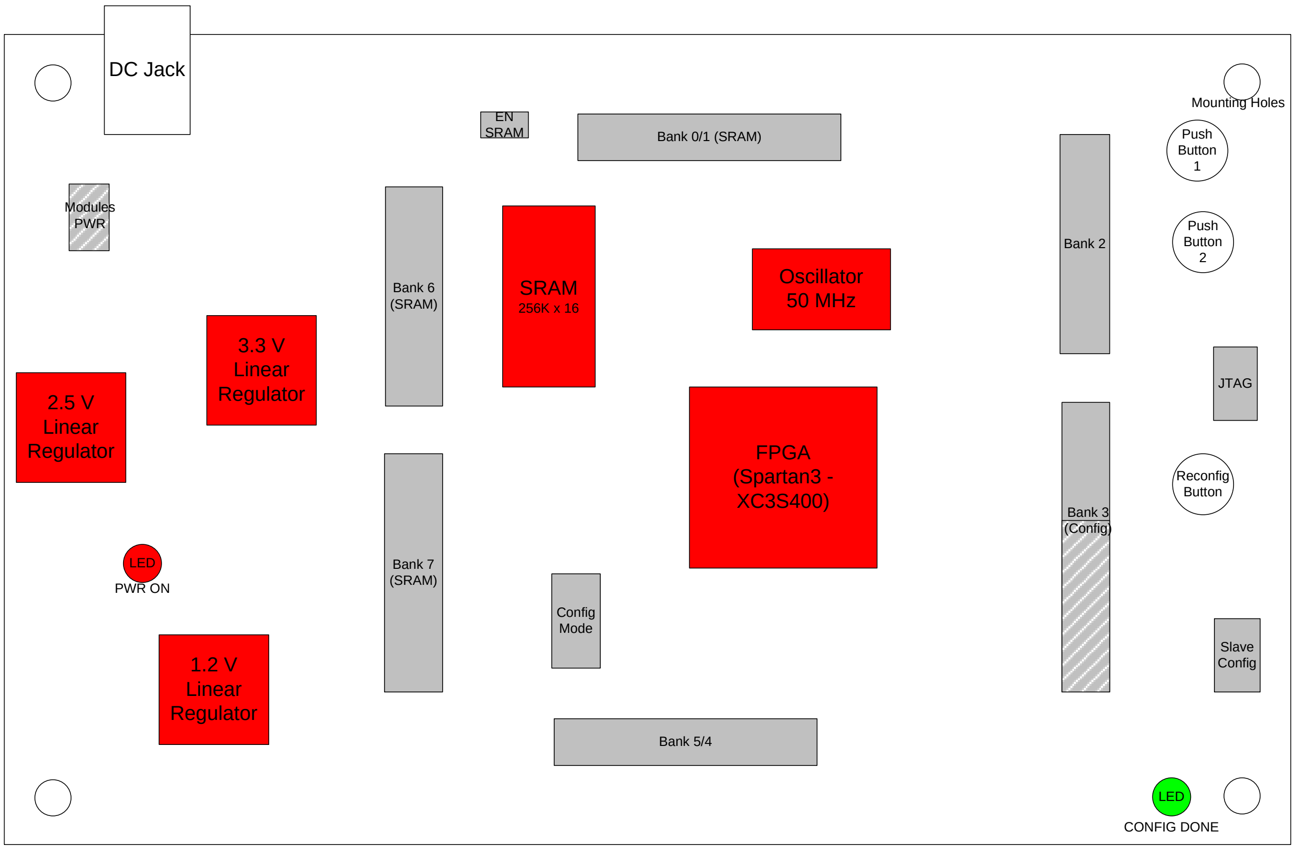


Figure 7.5: Intelligence module board layout. The rectangles according to their colour are: red for ICs; grey for pin headers; grey with white strips for headers that are placed on both sides of the board (male header on the top, female header on the bottom); white for anything else.

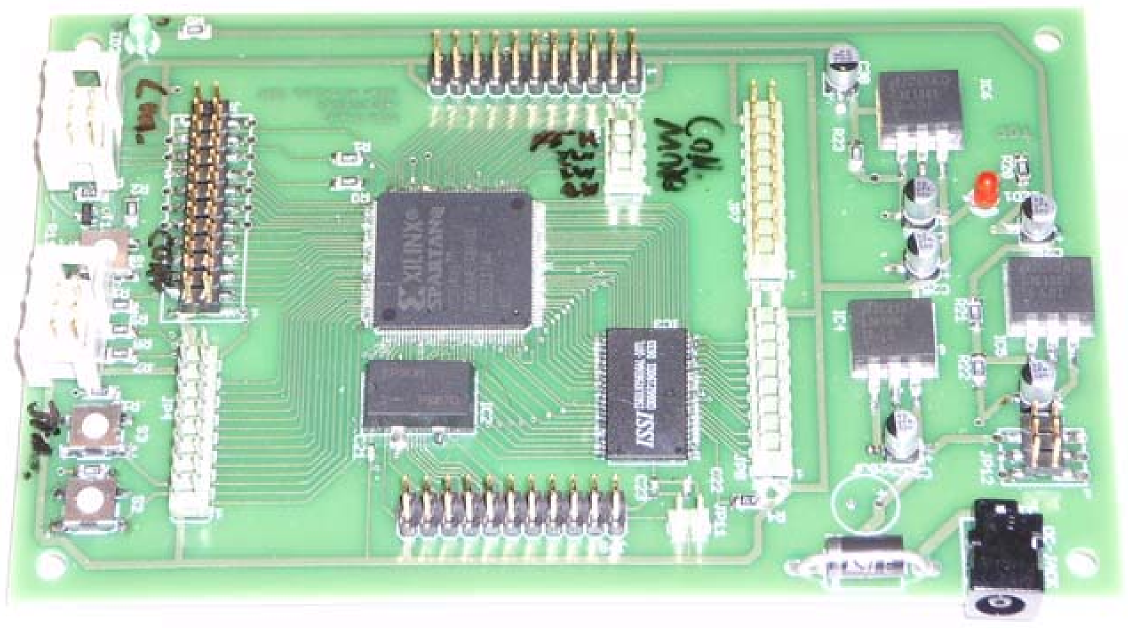


Figure 7.6: Intelligence module PCB

7.2 The Platform

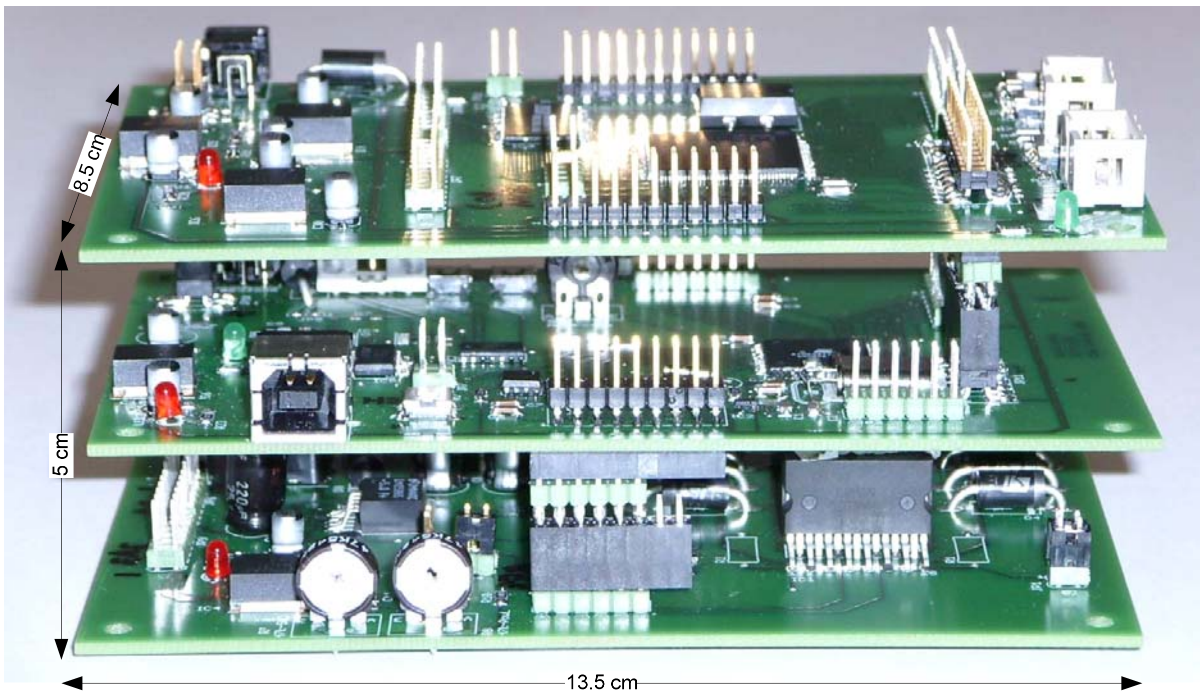


Figure 7.7: MAMoRo - The modular autonomous mobile robot platform after its implementation

Figure 7.7 depicts the implemented robot platform, MAMoRo, with its three modules connected together in a stacked-up form. The order of the module, starting from the bottom, is: Power & Motion module, Control module, then the Intelligence module (see *3.5 The Physical Implementation*).

The electrical connections between each module is achieved by plugging a female header placed on the bottom-side of the upper module with a male header placed on the top-side of the module below it, as illustrated in Figure 7.7. Thus, there is no need for additional cables to connect the modules together, avoiding the cable salad and the platform looks neat.

To increase the height between the modules, so that a module does not touch the components of the module beneath it, *header height-extenders* were used. These extenders were created during the project by soldering a male and female header together (see Figure 7.8), because for some reason they are very rare to find by electronic suppliers.

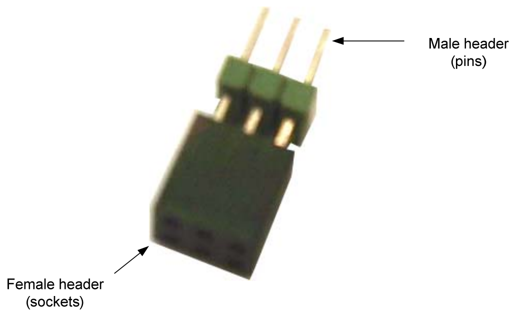
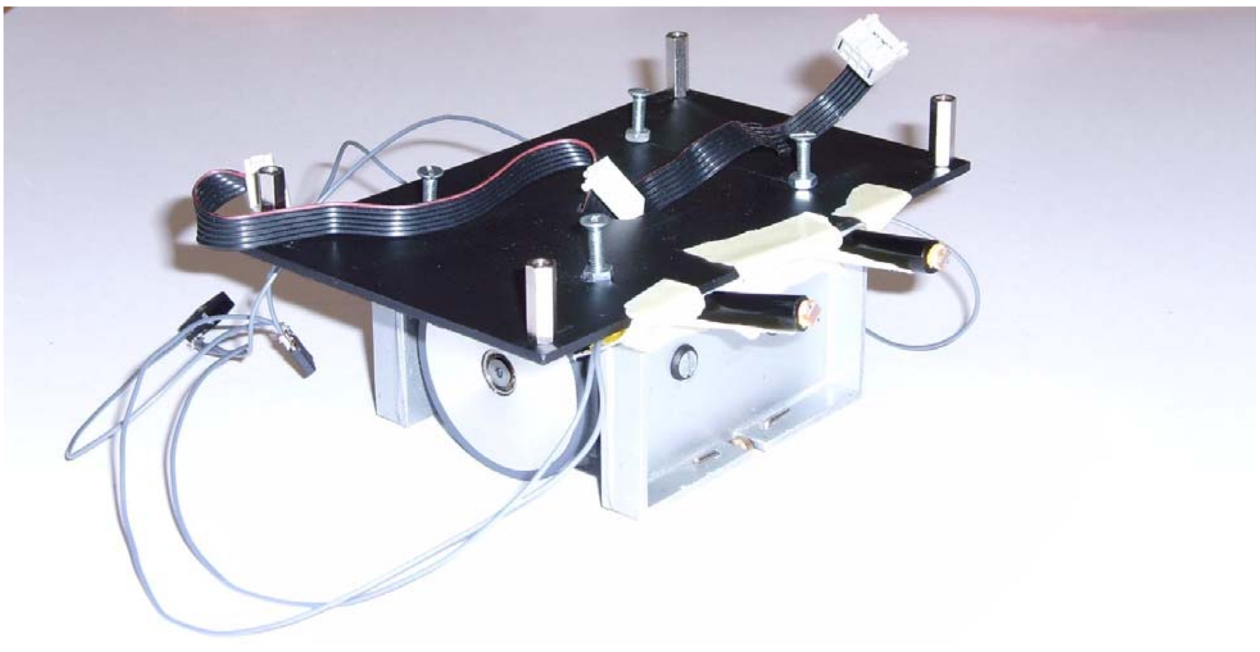


Figure 7.8: Header height-extenders: pins on one side, on the other sockets

7.3 The Mechanical Chassis

Building a mechanical chassis for MAMoRo was not one of the objectives of this thesis. The mechanical chassis where the modules of MAMoRo are placed can be variously chosen. A flexible and cheap option would be to use LEGO™ Technic[1] construction bricks. With these bricks, the mechanical chassis can be modelled with great flexibility and ease of use to many different forms. In this project, however, a mechanical chassis developed by IHRT for a soccer robot was used to hold the modules, the two motors with integrated motors, and the battery (see Figure 7.9). Figure 7.10 depicts MAMoRo when used in the application "Go Ahead" (see *9.2 Go Ahead*).



[1] LEGO Technic: http://technic.lego.com

Figure 7.9: The Mechanical Chassis used for MAMoRo

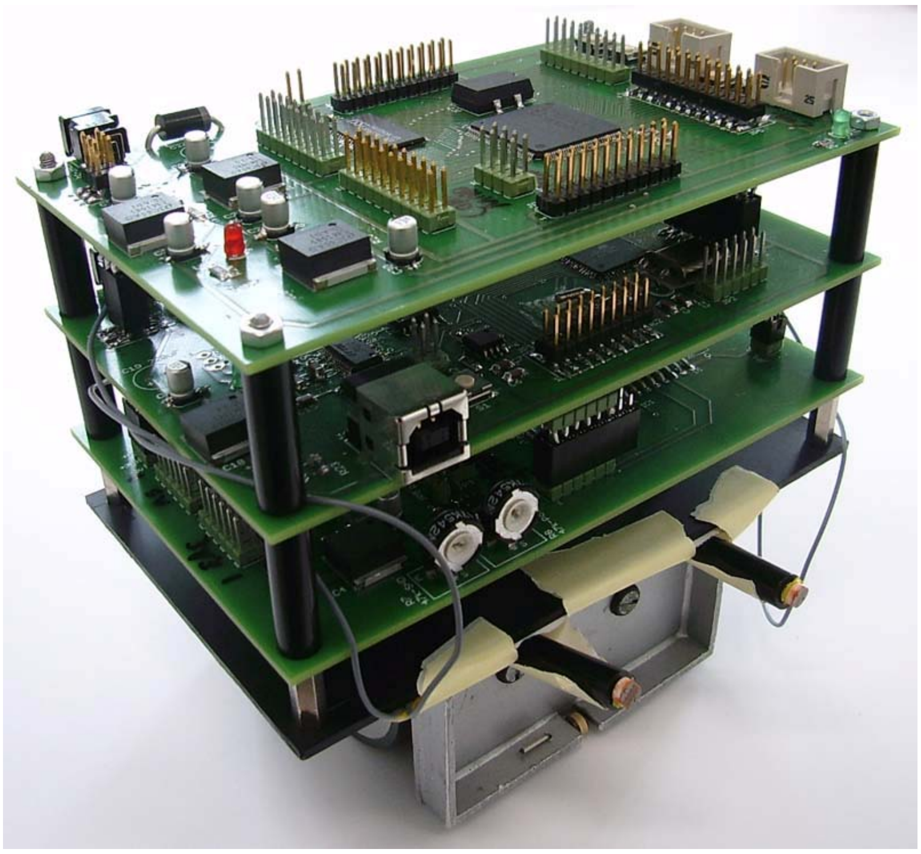


Figure 7.10: MAMoRo in the "Go Ahead" application (see *9.2 Go Ahead*)

8 Programming MAMoRo

After completing the implementation of MAMoRo and discussing it in Chapter 7, in this chapter, the steps for programming MAMoRo are described.

The robot platform MAMoRo with its three modules has two programmable units: the µC in the Control module, and the FPGA in the Intelligence module. However, as have been stated in *3.6 Programming MAMoRo*, the overall function of the system should be programmable by just programming the µC in the Control module. This process is explained in section *8.3 Programming the Overall System*. Section 8.1 and 8.2 deals with programming of the Control and Intelligence module, respectively, when used in stand-alone.

All software tools used for programming and interacting with MAMoRo are free and can be downloaded from the internet.

8.1 Programming the Control Module

The programmable unit in the Control module is the µC ATmega128L. The µC is programmed by connecting it to a PC and downloading the application code to its program flash memory. The connection from the PC to the Control module is done via USB cable.
However, when the µC is used for the first time, it can not communicate through the USB interface. A bootloader has to be downloaded first to make the programming of the µC through the USB possible. In addition, when the µC is used for the first time, system fuse-bits that configure the functionality of the µC (e.g. what type of clock source is used or in what mode will the µC operate) have to be set. There is more than one interface to accomplish the initial programming of the µC. However, the cheapest and easiest is through the ISP (In-System Programmable) interface (see Table 8.1). The pins of the ISP are accessible through the ISP header in the Control module itself (Figure 8.1).

A parallel programmer cable is used for connecting the Control module with the PC for the initial programming of the µC in the Control module. This programmer cable can also used for programming the Intelligence module when used in stand-alone. The programmer cable is described in section *8.1.1 Programmer Cable*.

Table 8.1: The ATmega128L serial SPI interface

Symbol	**Pins**	**Direction**	**Description**
MOSI (PDI)	PE0	Input	Serial data in
MISO (PDO)	PE1	Output	Serial data out
SCK	PB1	Input	Serial clock

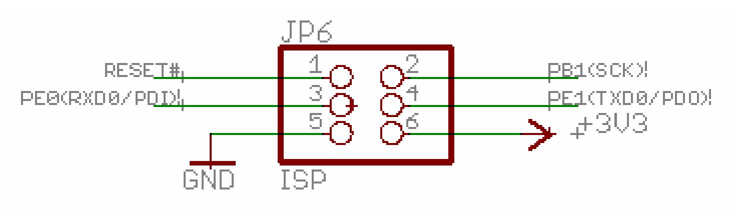


Figure 8.1: ISP header in the control module

The software environment used for programming the µC is *WinAVR™*, which is a set of open source development tools used for Atmel AVR µCs. WinAVR is described in section *8.1.2 WinAVR – Development Tools for Atmel AVR Microcontrollers*. The initial programming of the Control module is described in section 8.1.3 *Initial Programming of the Control Module*, while the programming via USB is described in section 8.1.4 *Programming the Control Module via USB*.

8.1.1 Programmer Cable

For the initial programming of the µC in the Control module, and for programming the FPGA in the Intelligence module when used in as stand-alone, the programmer cable depicted in Figure 8.2 is used. The programmer cable used in this project is a replica of Xilinx's Parallel Download Cable [XIL00]. The schematics and board layout for it can be found in *Appendix A - Schematics, Board Layouts, and Parts List.*

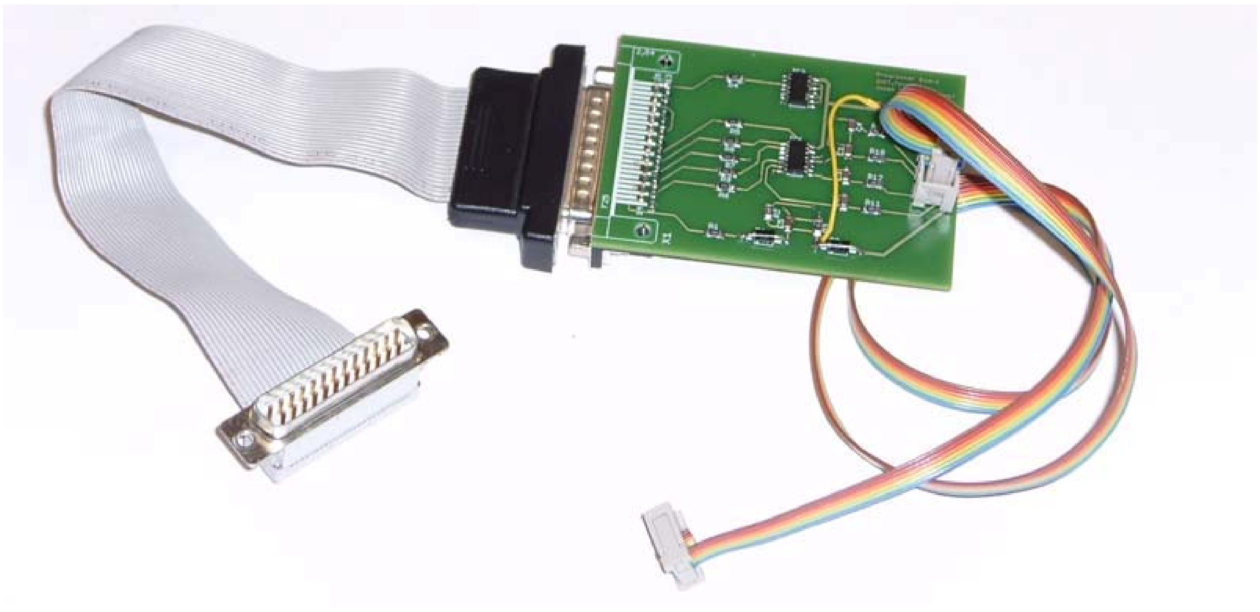


Figure 8.2: The parallel programmer cable used for the initial programming of the µC in the Control module.

The reasons for using particularly this programmer are:

- It can be used for programming both the µC in the Control module and the FPGA in the Intelligence module.
- Cheap and easy to Assemble
- Supported by open source programming software tools

The Programmer cable is plugged at the PC side to a standard parallel DB-25 port, and at the other end to either the SPI header in the Control module for programming the on-board µC, or the JTAG header in Intelligence module for programming the on-board FPGA; both the SPI and JTAG header have six pins.

The programmer board contains buffers that isolate the PC parallel port internal logic from the modules that will be programmed. This avoids damage happing to PC parallel port if the modules are defect, and vice versa.

The data transmission rate supported by this programmer is not an issue, because this connection is only used for programming the modules, and particularly the initial programming of the Control module; later programming of MAMoRo are going to be done via the faster and popular USB interface.

During the project, it was found out that this programmer cable, which is based on the Xilinx Parallel Download Cable, exhibits EMI problems due to its design. Many times did the programming of the µC failed because of verifications errors, and only after multiple tries, did it finally work. These verification errors increases when the size of the program code to be downloaded increases. An improvement to the original Xilinx programmer would be if the buffers were replaced by Schmitt triggers. This issue was no further followed since the programmer is used only for the initial programming of the Control module.

8.1.2 WinAVR – Development Tools for Atmel AVR Microcontrollers

WinAVR[1] is a suite of open source software development tools for the Atmel AVR series microcontrollers. WinAVR runs at the Windows environment and includes a GNU GCC compiler for C and C++. Most of the tools are used through the DOS command-line. An installer file can be downloaded from the homepage of WinAVR [WIN07] and installed on the host PC. After installation, the tools are ready to be used.

Although any programming language can be used to program the µC, in this project the programming language C was used because of its flexibility (as compared to BASIC) and of its high-level code (as compared to Assembly).

The most important tool in the WinAVR suite is the AVR-GCC complier. This complier converts our C code to machine code which can be downloaded directly to the µC. But before that, a *makefile* has to be created, which tells the AVR-GCC how to compile the files, i.e. what µC is used, what libraries and files to link, etc. The makefile can be created comfortably with the tool *MFile* through a graphical interface. This tool is also included in the WinAVR suite.

The last step in the programming process is to download the machine code to the µC. The machine code file downloaded to the µC is stored in a Hex format. The tool used for downloading this file from the PC to the µC is called *avrdude*. To start the downloading process, this tool has to know what kind of µC it is programming, type of programmer cable, the PC port, the name of file to be downloaded, and other parameters.

[1] WinAVR™ homepage: http://winavr.sourceforge.net/

The user manuals for some of the tools are found in the WinAVR installation directory. To summarize everything, the following steps have to be done to successfully program the µC:

1. Write the source code with a text editor (WinAVR suite comes also with one)
2. Create the makefile using MFile
3. Compile the source code by typing "make" in the command line
4. Download the hex file to the µC using avrdude

8.1.3 Initial Programming of the Control Module

The initial programming of the µC in the Control module consists of the following steps:

- Setting the fuse-bits of the µC
- Downloading the bootloader to the boot program section of the flash memory in the µC to enable later programming to be done through the USB interface.

These initial programming steps are done through the programmer cable described in *8.1.1 Programmer Cable*, and are described more in detail in the following sub sections.

Setting the Fuse-bits for the Microcontroller

The ATmega128L microcontroller has fuse-bits to configure some of its operational functions. These fuse-bits are stored in the µC internal flash memory. ATmega128L has three fuse bytes (i.e. 24 fuse-bits):

- Extended Fuse byte
- Fuse High Byte
- Fuse Low Byte.

What each bit does and its default mode when delivered from the manufacturer can be read in the µC's own datasheet ([ATM06a]) and will be no further discussed. Only the fuse-bits altered for this project will be mentioned. Table 8.2 shows these changes.

Table 8.2: Fuse-bits that have to be changed in the ATmega128L from their default value

Fuse Byte	**Fuse-bit(s) altered**	**Action**
Extended Fuse byte	M103C = 1	Disable ATmega103 compatibility mode

Fuse High Byte	JTAGEN = 1
	
	BOOTSZ1=1
	BOOTSZ0=0
	
	BOOTRST=0
		Disable JTAG interface.
		
		Sets the size of the Bootloader flash section to 1024 words (2 KiByte).
		
		Select Reset vector

Fuse Low Byte	SUT1=1
	SUT0=1
	
	CKSEL3 = 1
	CKSEL2 = 1
	CKSEL1 = 1
	CKSEL0 = 1
		Crystal oscillator; slowly rising power (65 ms)
		
		The fuse bits CKSEL3-1 set the frequency range of the µC. In our case it is 8 MHz. CKSEL0 sets the start-up time for the crystal oscillator.

There is no need for the restricted ATmega103 compatibility mode, so it was disabled.
The JTAG interface uses four pins (PF7..4), and to enable these pins to be used as general-purpose I/O, the JTAG is disabled. The Bootloader code requires 1024 words in the bootloader flash section; this is set by the fuse-bits BOOTSZ1-0. In addition, for the bootloader to work, the reset vector has to be moved to the start of the boot section in the flash memory by setting the fuse-bit BOOTRST. The default clock source of the ATmega128L when shipped from the manufacturer is the internal RC oscillator running at 1 MHz. To use instead the 8 MHz quartz crystal, the fuse-bits CKSEL3..0 are changed accordingly. The fuse-bits SUT1-0 were changed to tell the µC that the clock source is a crystal, and that the power source may rise slowly.

The fuse-bits are changed using the open source tool avrdude, which is included in WinAVR (see the previous section). To alter the fuse-bits of the µC according Table 8.2, connect the Control module with the PC using the programmer cable, and enter following three commands at the command-line (assuming port LPT1 is used):

```
> avrdude -p atmega128 -c xil -P lpt1 -U lfuse:w:0xff:m

> avrdude -p atmega128 -c xil -P lpt1 -U hfuse:w:0xdc:m

> avrdude -p atmega128 -c xil -P lpt1 -U efuse:w:0xff:m
```

Downloading the Bootloader

The bootloader allows the programming of the µC through the USB interface, so there will be no need for the programmer cable mentioned previously. The source code for the bootloader is based on the application note AVR109 by Atmel [ATM06b], and was modified and extended for the µC used in this project. The source code for the bootloader is included in the CD attached with this thesis. Before the bootloader is downloaded to the µC, the source code has to be first compiled. The makefile is included with the source code. So, to start the compilation, the following command is entered at the command-line in the directory where the source code is:

```
> make
```

Afterwards, the bootloader is downloaded to the µC using the tool avrdude by entering the following command:

```
> avrdude -p atmega128 -c xil -P lpt1 -U flash:w:bootloader.hex
```

Avrdude will display the success message if the downloading was done without any problems. Now, we can program the µC using a simple USB cable. To protect the boot section in the flash memory (where the bootloader resides) from accidental future overwriting, the Lock fuse-bits are changed by the following command:

```
> avrdude -p atmega128 -c xil -P lpt1 -U lock:w:0xef:m
```

The following sub section deals with programming the Control module via USB.

8.1.4 Programming the Control Module via USB

After completing the initial programming of the Control module (see previous section), and before starting to program the Control module via USB, the device drivers for the FT232R (USB-UART converter IC, see also *5.4 USB Interface*), which are found on the manufactures website[1], have to be downloaded, installed, and configured. These drivers emulate the USB interface as a serial port on the PC, thus any software can access it as if it was a serial port.

Finally, to start programming the Control module via USB, the µC has to enter the programming mode by activating the bootloader in the µC. To enter the programming mode, the Reset and PB1 button (another button can also be chosen in the source code) on the Control module have to be pressed at the same time. After this is done, the following command starts the downloading the application program code to the µC (assuming *myapp.hex* is the compiled hex program code):

```
> avrdude -p atmega128 –b 57600 -c avr109 -P com5 -U flash:w:myapp.hex
```

Note that although we are using physically the USB interface, logically it is accessed as if it were a serial port (see *5.4 USB Interface*). The number 57600 stands for the speed of the communication, i.e. 57600 bit/s. This is the highest possible transfer rate with no errors by the ATmega128L running at 8 MHz, and is more than sufficient for our programming purposes, e.g. downloading the application for “Go Ahead” in *9.2 Go Ahead* takes less than 2 s. The option “-c avr109” in the command line tells avrdude what kind of bootloader is installed on the µC.

[1] www.ftdichip.com

8.2 Programming the Intelligence Module

The Intelligence module can be used as an evaluation platform to conduct a wide variety of experiments in electrical engineering and computer science. This module has an FPGA which because of its volatile nature needs to be configured[1] (programmed) every time after power-on or pressing the ReConfig button.

When the Intelligence module is used with the Control module, the latter takes care of configuring the FPGA whenever it is signalled by the FPGA. But before that, the flash memory and the µC on the Control module have to be programmed to do that (see *8.3 Programming the Overall System*). However, if the Intelligence module is used as stand-alone, the FPGA can be configured using the JTAG header on the module. The programmer cable (see *8.1.1 Programmer Cable*) used for the initial programming of the Control module can also be used to connect the PC parallel port with the JTAG interface of the Intelligence module for configuration. The Drawback in using the Intelligence module without the Control module is that always when the power is turned off or the ReConfig button is pressed, the FPGA has to be re-configured again by the PC.

The development software suit used for writing and compiling the source code and for configuration of the FPGA is ISE WebPACK, which is provided by Xilinx, the manufacturer of the FPGA itself. The software suit is available for free from Xilinx's website[2].
Unlike the µC in the Control module where its source code is written in the C programming language, the FPGA has to be programmed using a hardware description language such as VHDL or Verilog.

The JTAG configuration of the Intelligence module in stand-alone was successfully tested with a simple application.

8.3 Programming the Overall System

MAMoRo with its three modules has two programmable units, the µC in the Control and the FPGA in the Intelligence module which need to be programmed. Furthermore, the volatile

[1] When talking about FPGA, the words "configured" and "programmed" mean the same thing
[2] Xilinx homepage: http://www.xilinx.com/

FPGA has to be configured each time the power is turned on. As have been already stated earlier in *3.6 Programming MAMoRo*, the overall system should be programmable by just programming the µC in the Control module. The µC should afterwards take care of configuring the FPGA; but before that, two things must be done:

1. The FPGA configuration file has to be downloaded from the PC to the flash memory on-board the Control module.
2. The µC must be programmed how to use the data bits stored in the flash memory to configure the FPGA.

For the first problem, regarding the downloading of the configuration file from the PC to the flash memory, there are many ways to do this. For this project, the communication protocol *Xmodem-CRC*[1] was identified as an effective way to let the PC communicate with the µC in the Control module for downloading the FPGA configuration file to the flash memory. Xmodem-CRC was chosen because it is simple and reliable enough for our task. So, the Xmodem-CRC protocol has to be programmed into the µC. Afterwards, on the PC side, a terminal program capable of understanding the Xmodem-CRC protocol is opened and used to communicate with the µC and download the FPGA configuration file to the flash memory. A terminal program capable of using the Xmodem-CRC protocol to send and receive files is, for example, *HyperTerminal*, a tool already integrated in the Windows environment.

After succeeding in downloading the FPGA configuration file to the flash memory on the Control module, the µC has to be programmed how to handle this stored data to configure the FPGA. The FPGA is configured in the slave serial configuration mode. The sequence of steps to configure the FPGA in slave serial mode is depicted in Figure 8.3.

[1] Xmodem-CRC protocol reference: http://pauillac.inria.fr/~doligez/zmodem/ymodem.txt

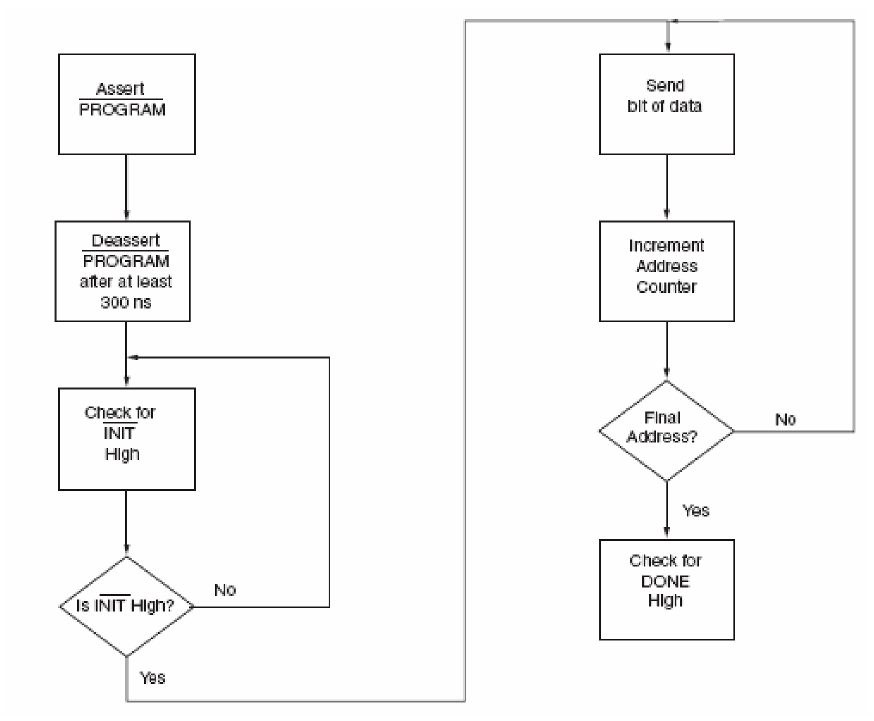


Figure 8.3: The FPGA configuration flow diagram in slave serial mode [XIL02]. PROGRAM, INT, and DONE are pins in the FPGA used for the controlling the configuration.

The algorithm illustrated in Figure 8.3 was implemented in the µC and succeeded in configuring the FPGA from the flash memory. The FPGA configuration file which will be downloaded to the flash memory is a binary file generated by ISE WebPACK. More details in regard to the configuration process of the FPGA can be read in Xilinx application note [XIL02].

Finally, we have now all parts needed to make an overall bootloader for MAMoRo:

- The bootloader that enables us to program the Control module via USB.
- The Xmodem-CRC protocol for downloading or uploading data between the PC and the flash memory on the Control module.
- The µC program code for configuring the FPGA from the flash memory.

These single programs all combined make up the bootloader for MAMoRo. The algorithm for the MAMoRo bootloader is depicted in Figure 8.4.

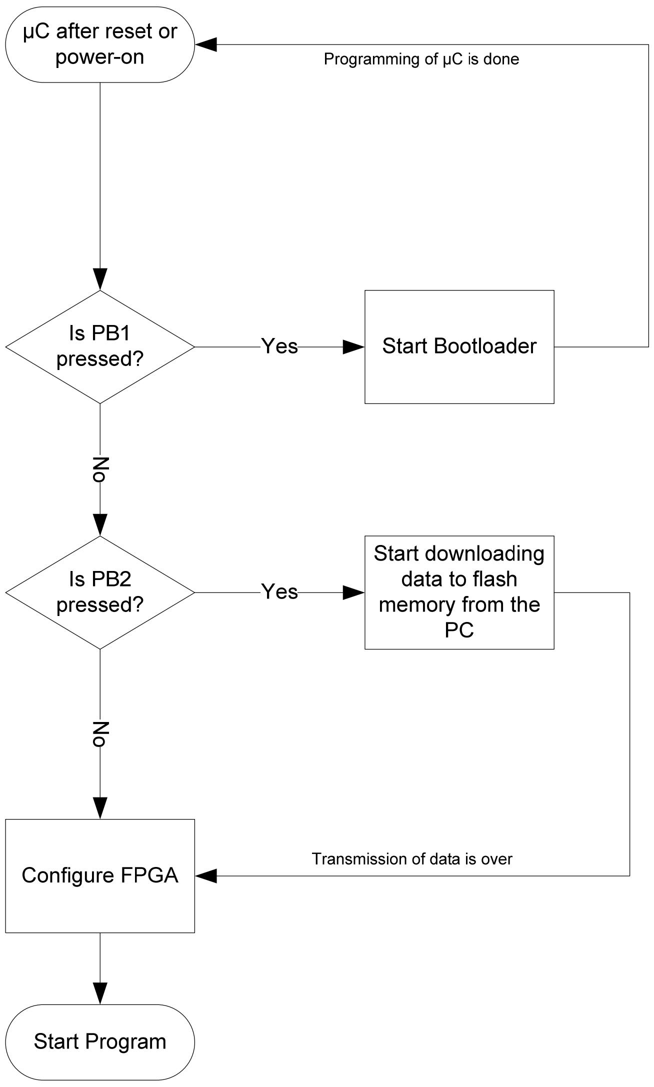

µC after reset or power-on
Programming of µC is done
Is PB1 pressed?
Yes
Start Bootloader
No
Is PB2 pressed?
Yes
Start downloading data to flash memory from the PC
No
Configure FPGA
Transmission of data is over
Start Program

Figure 8.4: MAMoRo-Bootloader

After the power is turned on, the µC checks if push button PB1 (another button can also be chosen) is pressed; if this is the case, the bootloader downloaded during the initial programming of the Control module (see *8.1.3 Initial Programming of the Control Module*) is activated, and the µC enters the programming mode, in which then the µC can be programmed. Next, the µC checks if button PB2 (another button can also be chosen) is pressed; if this is the case, the µC enters the Xmodem-CRC receive mode, and waits for the signals from the PC terminal to start downloading the data to the flash memory. After checking for buttons PB1 and PB2, the µC starts configuring the FPGA. The LED DONE in the Intelligence module glows when the FPGA is configured. Finally, when all this is done, the µC runs the user application program. If the ReConfig button in the Intelligence module is going to be used to re-configure the FPGA, then two lines of code have to be added to the application program code that tells the µC to poll the PROGRAM pin, otherwise pressing the Reset will also do it.

The source codes for the MAMoRo Bootloader are found on the CD attached with this thesis.

9 Application Examples

The final phase of this project is to actually test MAMoRo in real life with example applications. Two classic applications for robots were picked to demonstrate the functionality of MAMoRo:

- Line follower: MAMoRo should follow a line on the ground using two light sensors. Only the Control module and Power & Motion module are used to accomplish this task.
- Go ahead: MAMoRo is programmed to go forward, but when it detects an obstacle, it goes backward, then turns randomly either right or left, then goes forward again, and so on. Whereby, an obstacle is detected using the light sensors and the quadrature encoders integrated in the motors.

9.1 The Line Follower

The line follower is an application in which MAMoRo is programmed to follow a black line on a white ground. In this application the Power & Motion module and the Control module are used without the Intelligence module. This application should demonstrate how MAMoRo can be used for simple applications that do not require the high processing power of the Intelligence module. The interactions between the modules for this application are illustrated in Figure 9.1.

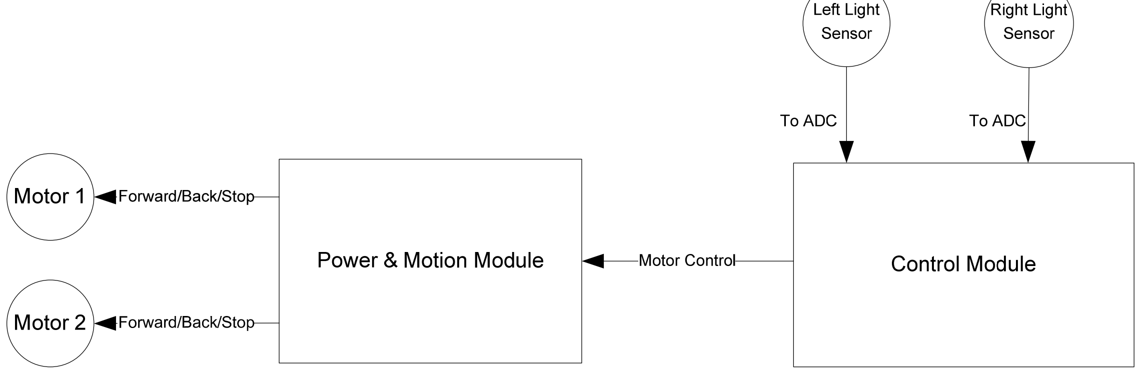


Figure 9.1: Modules interactions in the line follower application

A blank line is drawn in a white surface (see Figure 9.3), and MAMoRo should move along this line using two light sensors placed near to each other and facing the ground (see Figure 9.2). The light sensors used are cheap light dependent resistors (LDR). These sensors allow MAMoRo to detect when it has deviated from its specific course.

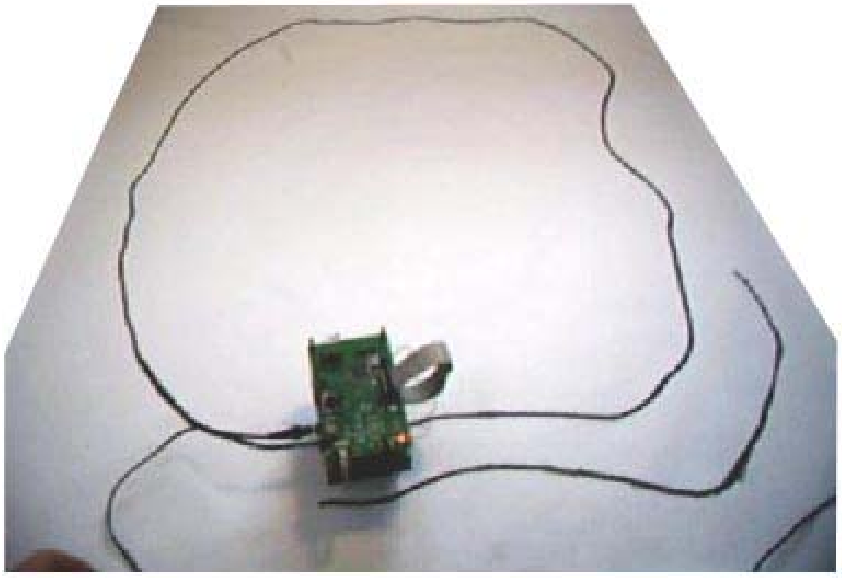


Figure 9.2: The course used for the line following application

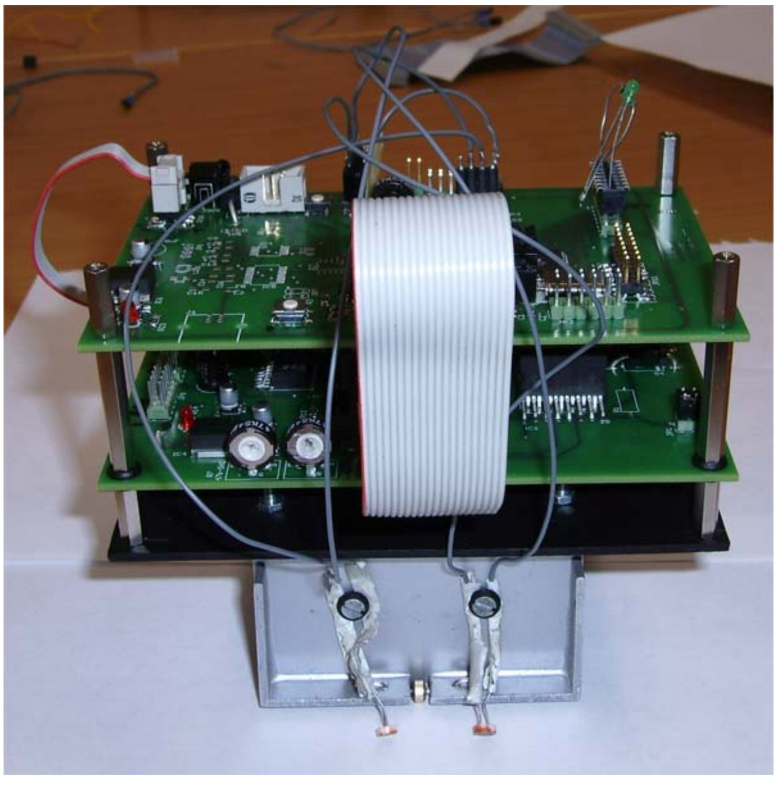


Figure 9.3: MAMoRo used as a line follower. The light sensors (LDRs) are placed near to each other facing the ground.

The light sensors are connected to the µC ADC's in the Control module; their schematic is depicted in Figure 9.4.

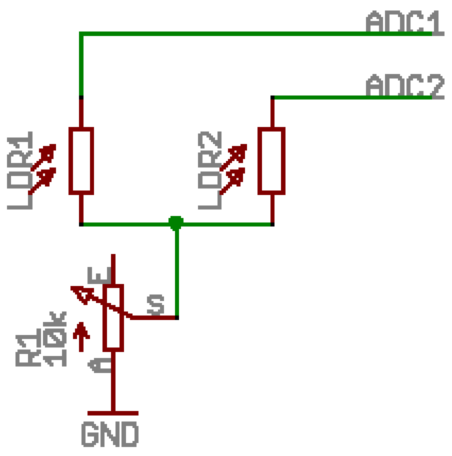


Figure 9.4: LDR circuit

The algorithm used for the line follower application can be read from the flow diagram depicted in Figure 9.5. The µC reads in loops the values from these sensors and calculates the difference between them. If the absolute difference between the values read from each sensor is equal or greater than a specific constant, then that means that MAMoRo has stepped over the line and must turn right or left according to the sign of the difference value. So, basically MAMoRo tries continuously to keep the line between both sensors. By using the difference value between each sensor, rather than the absolute values, background noises such as light interferences or shadows from nearby objects are minimized.

The source code for the line follower is included in the CD attached with this thesis.

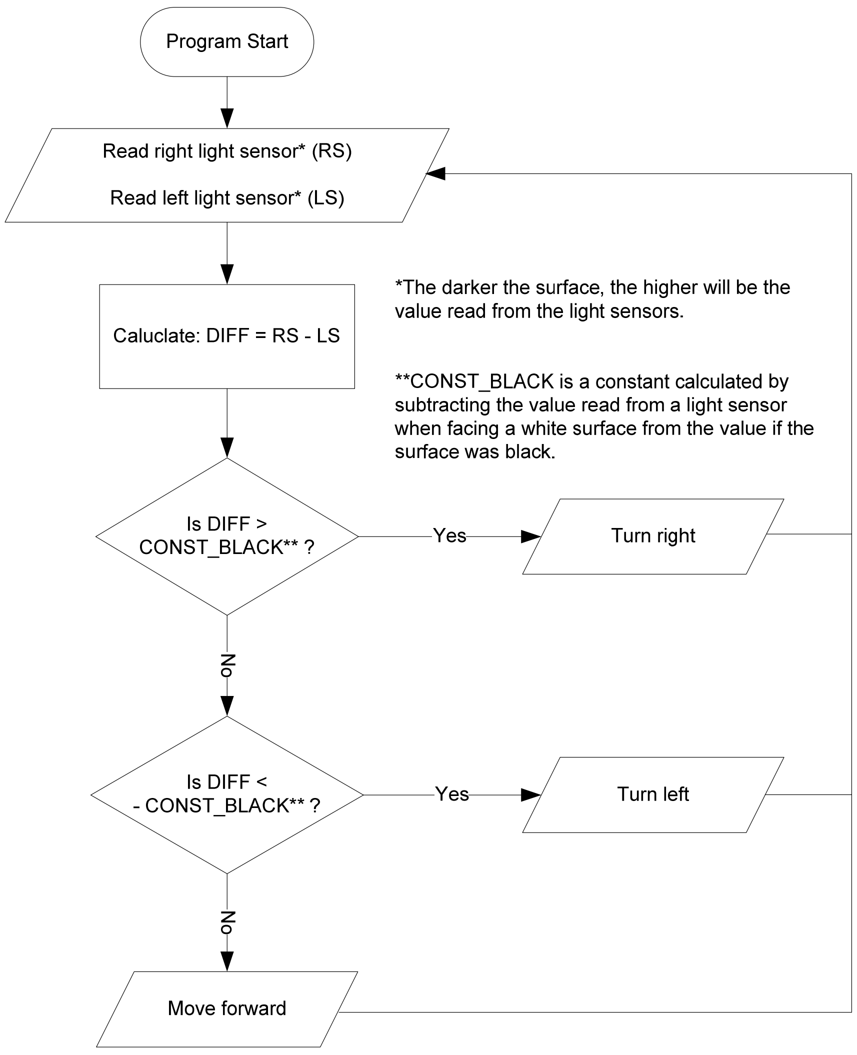

Program Start
Read right light sensor* (RS)
Read left light sensor* (LS)
Caluclate: DIFF = RS - LS
*The darker the surface, the higher will be the value read from the light sensors.
**CONST_BLACK is a constant calculated by subtracting the value read from a light sensor when facing a white surface from the value if the surface was black.
Is DIFF > CONST_BLACK** ?
Yes
Turn right
No
Is DIFF < - CONST_BLACK** ?
Yes
Turn left
No
Move forward

Figure 9.5: Line follower flow diagram

9.2 Go Ahead

Go Ahead is an application in which all of MAMoRo's modules are utilized for a specific task. Therefore, it is a perfect demonstration of the interaction of all of MAMoRo's three modules (see Figure 9.5). In this application, MAMoRo is programmed to go forward (ahead) and stops if it detects an obstacle. It uses two kinds of sensors: two LDRs as light sensors, and quadrature encoders integrated in the motors. If the obstacle is detected, MAMoRo stops and goes backwards, then turns randomly right or left, then goes forward again, and so on.

There are two ways for MAMoRo to detect an obstacle in this application. First, is if this obstacle comes near to the light sensors and blocks some of the light reaching it. Second, is by counting the pulses from the quadrature encoder. The quadrature encoder integrated in each motor generates pulses according to the speed of the motor, i.e. the faster the motor spins, the more pulses per second are generated. So, if the motion unit in the Power & Motion module is telling the motors to move, but is reading from the encoders no or few pulses, then MAMoRo understands that an obstacle is hindering it from moving.

After an obstacle has been detected, it moves backwards and then turns randomly either right or left, then continues moving forward again; this procedure is repeated all the time. The algorithm for the Go Ahead application can be read from the flow diagram in Figure 9.6.

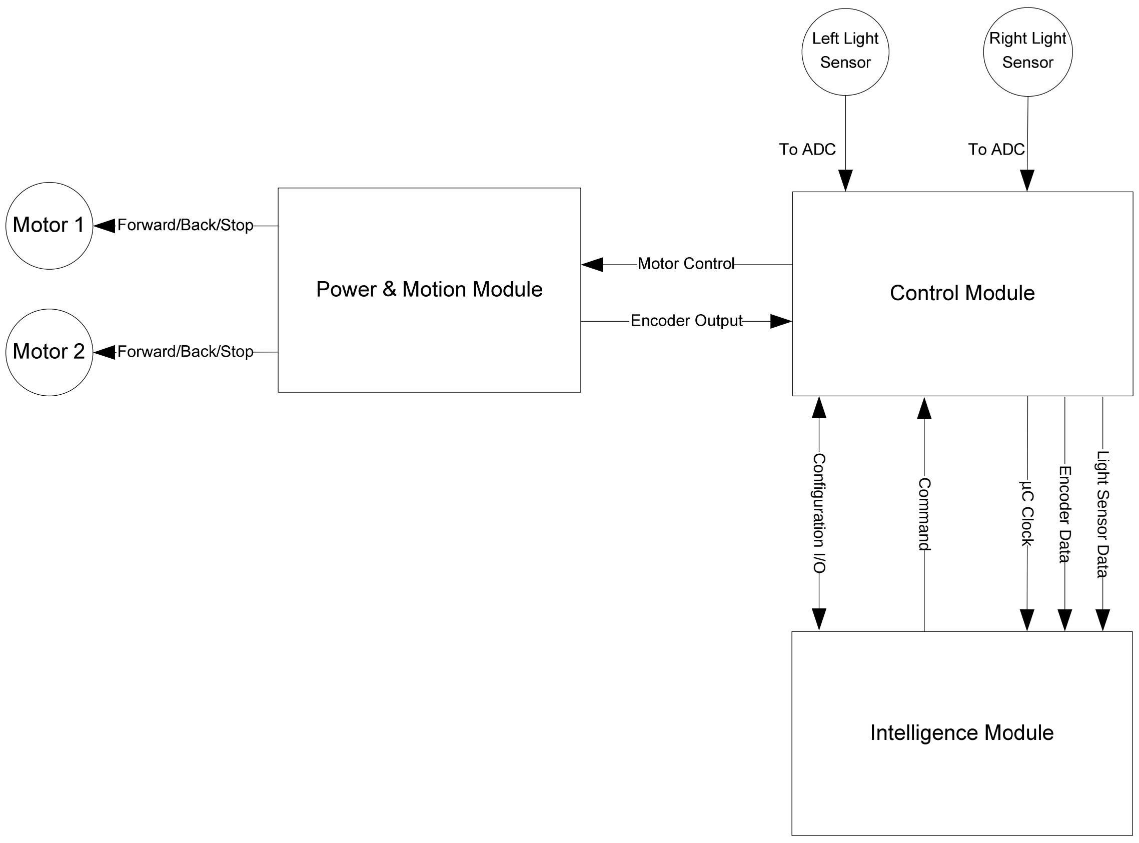


Figure 9.6: Interaction of MAMoRo's three modules: Power & Motion module, Control module, and intelligence module.

The main functions of each module in this application are summarized below.

Power & Motion Module:

- Drives the motors
- Sends the encoders output to the Control module

Control Module:

- Sends motor control signals to the motion unit in the Power & Motion module
- Configures the FPGA after power-on
- Reads the analogue signals from the light sensors, converts them to digital, and sends the results to the intelligence module.
- Reads the encoders pulses and calculates the pulses generated per unit time, then sends the results to the Intelligence module

Intelligence Module:

- Reads the sensor results from the Control module, processes them, and then tells the Control module what action to do next.

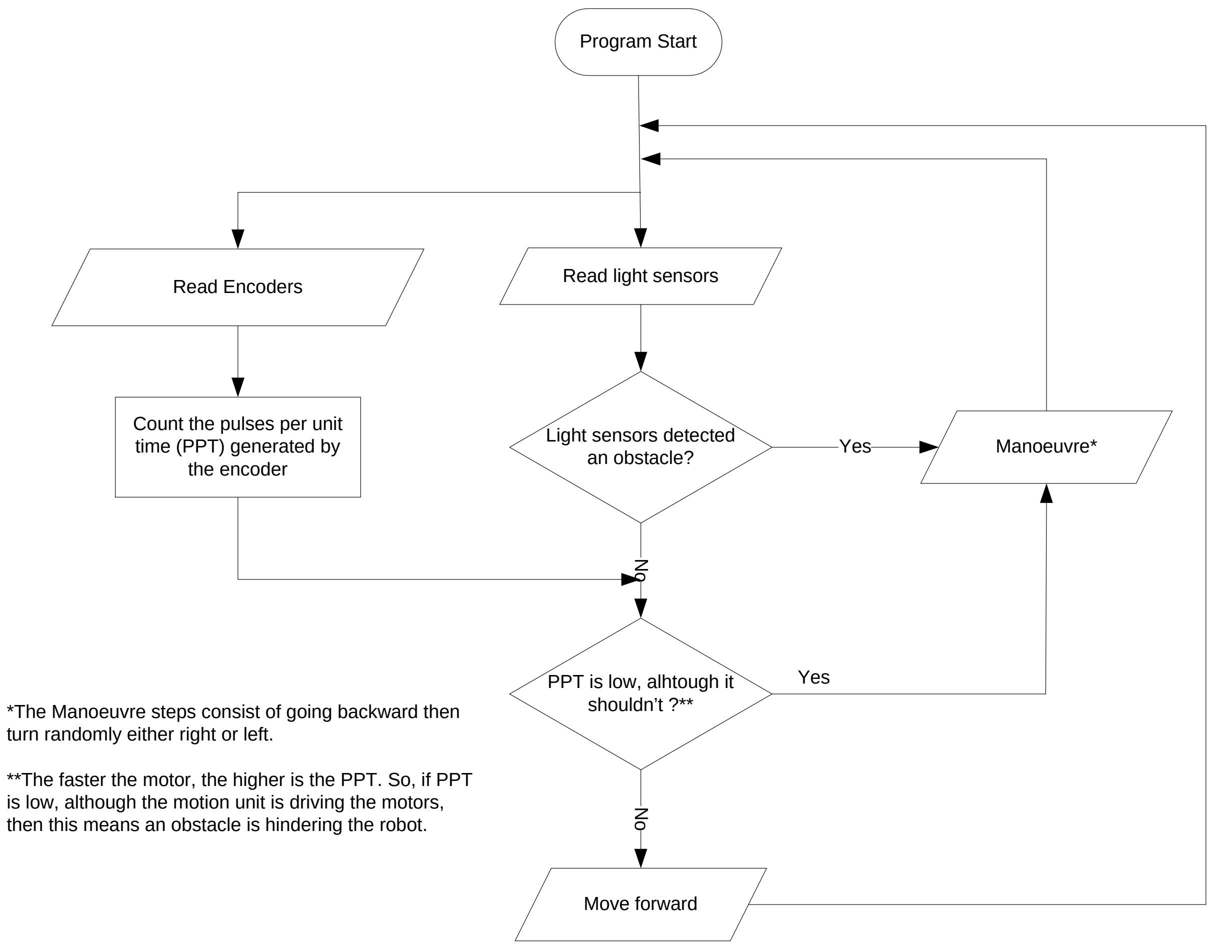

Program Start
Read Encoders
Read light sensors
Count the pulses per unit time (PPT) generated by the encoder
Light sensors detected an obstacle?
Yes
Manoeuvre*
No
PPT is low, alhtough it shouldn't ?**
Yes
No
Move forward
*The Manoeuvre steps consist of going backward then turn randomly either right or left.
**The faster the motor, the higher is the PPT. So, if PPT is low, although the motion unit is driving the motors, then this means an obstacle is hindering the robot.

Figure 9.6: Go Ahead flow diagram

The source code for the Go Ahead application is included in the CD attached with this thesis.

10 Conclusion and Future Work

10.1 Conclusion

The project was successfully completed and all objectives stated at the beginning of the thesis (see *1.2 Thesis Objectives*) were accomplished.

The combination of a µC and an FPGA in MAMoRo enhanced immensely the flexibility and adoptability of the system, and made it possible for MAMoRo to reconfigure its hardware at run-time.

The distribution of the decision making functions into two modules according to their complexity, as it happens in the human brain, achieved that the whole system is more easy to manage, the processing power is more efficiently utilized, and future expandability is made a lot easier.

The total cost[1] for the components used to build MAMoRo was about €200. The PCB fabrication cost was about €120 (excluding the soldering of the components, which was done manually). This makes the total cost of the project about € 320. Note, that this is the price for the production of only one sample of MAMoRo. This price will drop immensely if MAMoRo is produced in larger numbers. For the mechanical part of MAMoRo, LEGO™ Technic[2] bricks can be used, which will allow the mechanical parts to be modelled with great flexibility and ease of use; the price of these kits vary from €20 and above. The software tools used for programming MAMoRo are open source or free, and can be downloaded from the internet; hence, no extra cost falls on software development tools. Consequently, one of the requirements of this project which stated that the robot platform should be low-cost was achieved.

Nevertheless, there are some improvements that could be made to future MAMoRo samples. Since the prototype implemented in this project is the first one, making the PCB area small for this project was not important. In fact, a very small PCB was not favoured, so that if there were any mistakes in the hardware design, there would be enough space in the board to allow

[1] All prices mentioned here are tax free
[2] LEGO Technic: http://technic.lego.com

for corrections; for example, adding an extra capacitor, or swapping an IC without damaging the components besides it, etc. In future samples of MAMoRo, the PCB areas will be reduced; however, there might be still cases, e.g. in introductory courses (see *3.5 The Physical Implementation*) where this is not wanted.

Another concern, is that the time needed for the FPGA in the Intelligence module to get configured by the µC in the Control module after power-on takes about 12 seconds; this is for some reconfigurable hardware at run-time applications too long. This time can be reduced by using more effective algorithms, and by writing the code in the low-level language Assembly. However, the most effective solution is by using a µC that runs at higher operating frequency than the relatively low 8 MHz of the ATmega128L µC on-board the Control module. The low operating frequency of the ATmega128L limits also the maximum error-free speed of the communication between the PC and MAMoRo to 57600 bit/s. These two restrictions can be easily solved by using a µC that operates with a higher operating frequency. An optimal choice would be the ARM™ AT91 Thumb-based series µCs such as the AT91SAM7S256, which can operate up to 55 MHz, has many peripherals on-chip, and costs a little bit more than the ATmega128L.

10.2 Future Work

The focus of this thesis was the design, implementation, and test of the robot platform MAMoRo. Further experimentations with MAMoRo, which would have exceeded considerably the length of a typical master thesis, are still needed to be done. Ideas for future work are listed below:

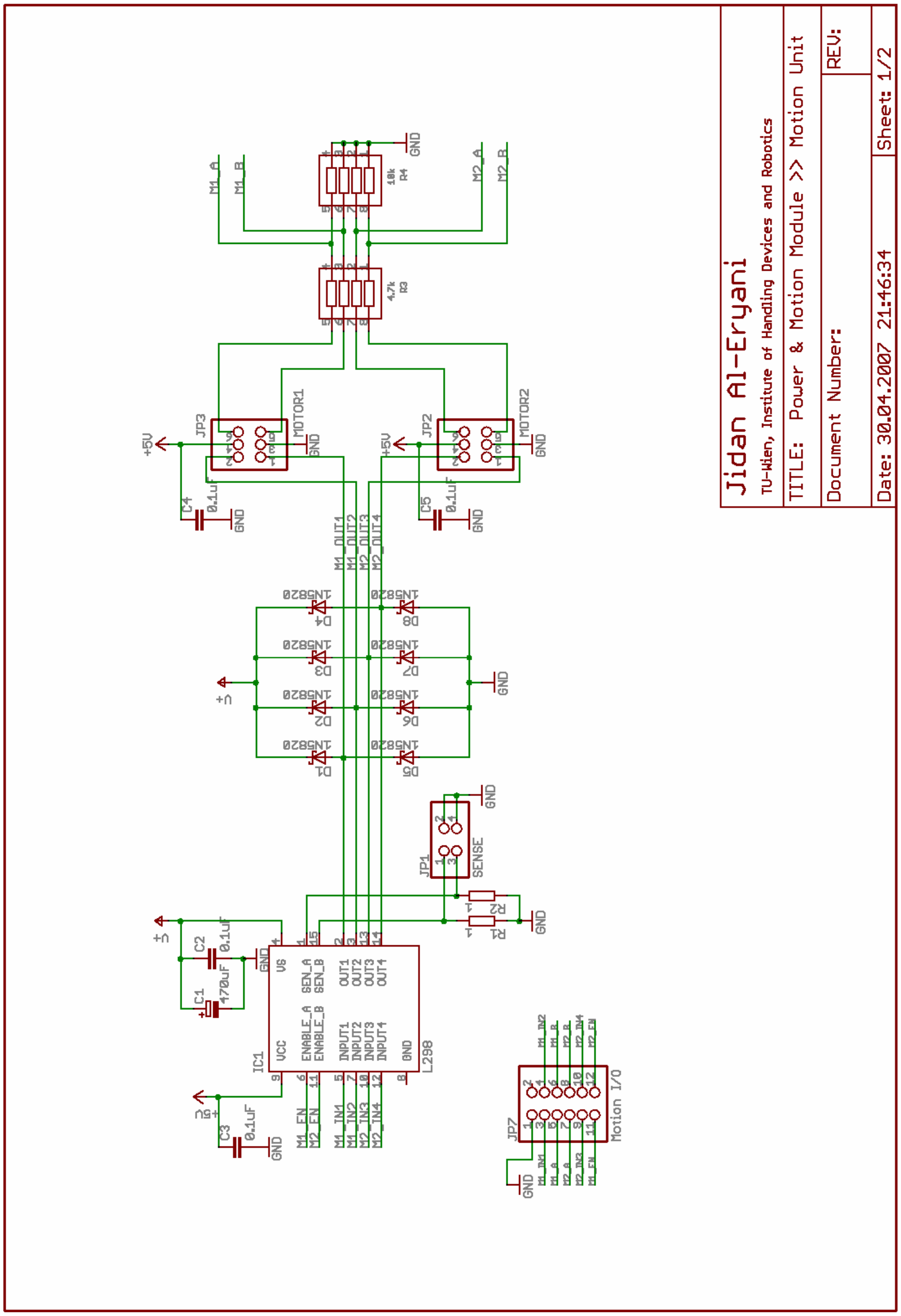
- Implement a soft core processor in the FPGA on-board the Intelligence module and test MAMoRo with some intensive-processing applications.
- Try practical hardware re-configuration applications at run-time with MAMoRo.
- Develop a software development environment with an easy to use graphical interface for programming all aspects of MAMoRo like the µC, the FPGA, the fuse-bits of the µC, downloading and uploading data to MAMoRo, etc.
- Try LEGO Technic™ construction bricks to create mechanical parts for MAMoRo.
- Develop a simulation software environment for modelling and rapid prototyping of MAMoRo.

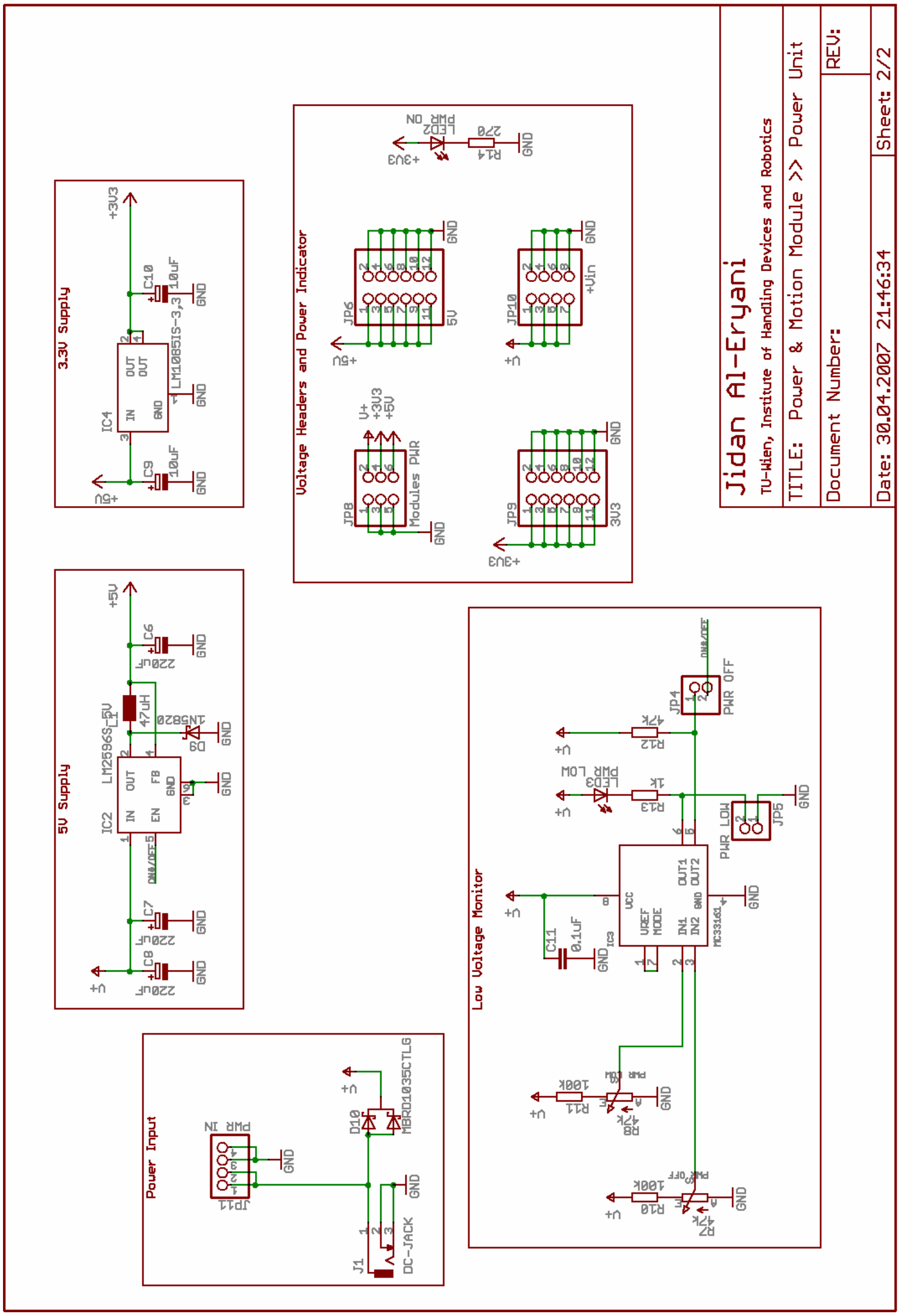
Appendix A - Schematics, Board Layouts, and Parts List

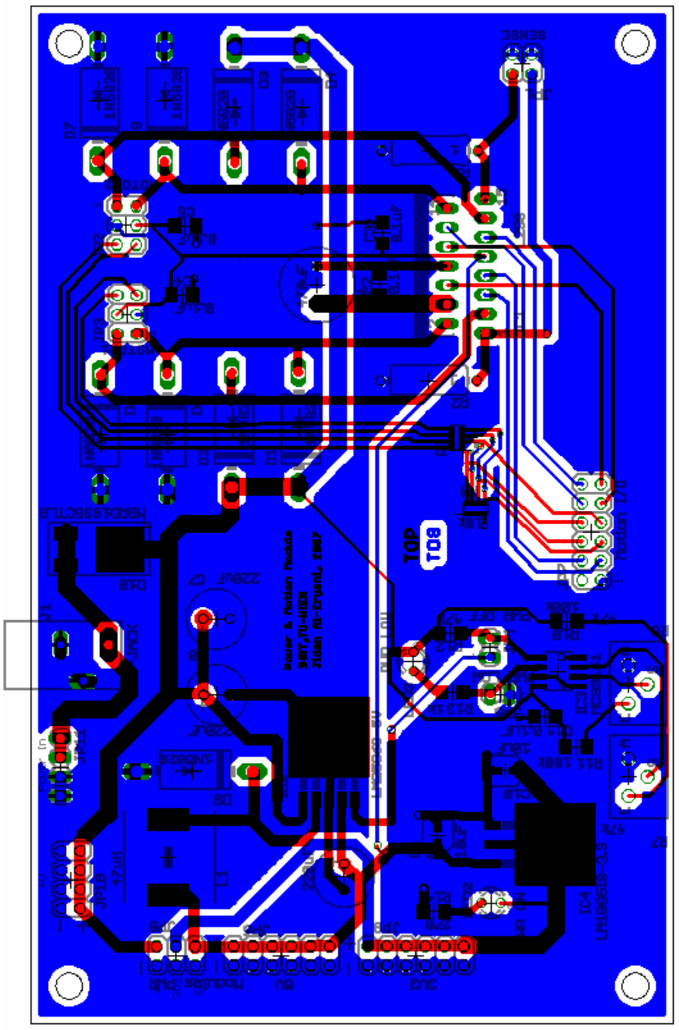
A.1 Power & Motion Module

Power & Motion module parts list

Quantity	Part	Value	Package
1	C1	470uF	C-10
5	C2, C3, C4, C5, C11	0.1uF	C1206
3	C6, C7, C8	220uF	C-8
2	C9, C10	10uF	C-PANS_B
9	D1, D2, D3, D4, D5, D6, D7, D8, D9	1N5820	DO-201AD
1	D10	MBRD1035CTLG	DPAK
1	IC1	L298	MULTIWATT-15
1	IC2	LM2596S-5V	TO263-5
1	IC3	MC33161	SOIC8-0.157
1	IC4	LM1085IS-3.3	TO-263-3
1	J1	DC-JACK	DC-JACK
1	JP1	SENSE	2X02
1	JP2	MOTOR2	2X03
1	JP3	MOTOR1	2X03
1	JP4	PWR OFF	1X02
1	JP5	PWR LOW	1X02
1	JP6	5V	2X06
1	JP7	Motion I/O	2X06
1	JP8	Modules PWR	2X03
1	JP9	3V3	2X06
1	JP10	Vin	2X04
1	JP11	PWR IN	1X04
1	L1	47uH	WE-PD_L/XL/XXL
1	LED2	PWR ON	LED3MM
1	LED3	PWR LOW	LED3MM
2	R1, R2	1	R-12X4
1	R3	4.7k	RX4-0603
1	R4	10k	RX4-0603
2	R7, R8	47k	PT10-5X2.5
2	R10, R11	100k	R1206
1	R12	47k	R1206
1	R13	1k	R1206
1	R14	270	R1206


Jidan Al-Eryani
TU-Wien, Institute of Handling Devices and Robotics
TITLE: Power & Motion Module >> Motion Unit
Document Number:
REV:
Date: 30.04.2007 21:46:34
Sheet: 1/2
IC1
L298
JP1
SENSE
JP2
MOTOR2
JP3
MOTOR1
JP7
Motion I/O
R1
R2
R3
4.7k
R4
10k
C1
470uF
C2
0.1uF
C3
0.1uF
C4
0.1uF
C5
0.1uF
1N5820
+5V
GND
M1_OUT1
M1_OUT2
M2_OUT3
M2_OUT4
M1_A
M1_B
M2_A
M2_B
M1_EN
M2_EN
M1_IN1
M1_IN2
M2_IN3
M2_IN4

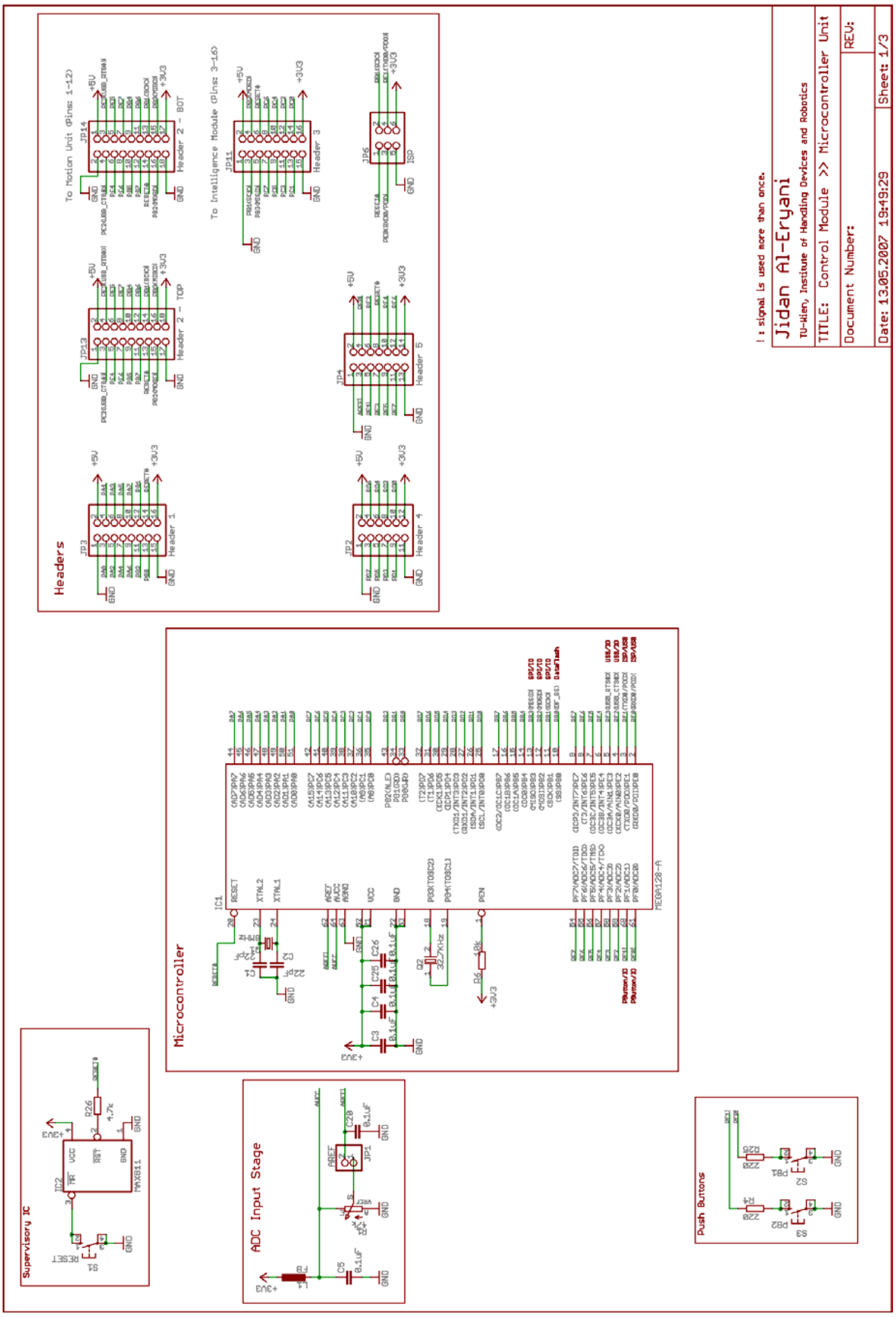

Jidan Al-Eryani
TU-Wien, Institute of Handling Devices and Robotics
TITLE: Power & Motion Module >> Power Unit
REV:
Document Number:
Date: 30.04.2007 21:46:34
Sheet: 2/2
3.3V Supply
IC4
LM1085IS-3,3
C9 10uF
C10 10uF
+3V3
+5V
GND
5V Supply
IC2
LM2596S-5V
L1 47uH
D9 1N5820
C6 220uF
C7 220uF
C8 220uF
ON#/OFF
V+
Power Input
JP11
PWR IN
J1
DC-JACK
D10
MBRD1035CTLG
Voltage Headers and Power Indicator
JP6
5V
JP8
Modules PWR
JP9
3V3
JP10
+Vin
LED2
PWR ON
R14 270
Low Voltage Monitor
IC3
MC33161
C11 0.1uF
R10 100K
R11 100K
R7 47K
R8 47K
PWR OFF
PWR LOW
R12 47K
R13 1K
LED3
PWR LOW
JP4
PWR OFF
JP5
PWR LOW

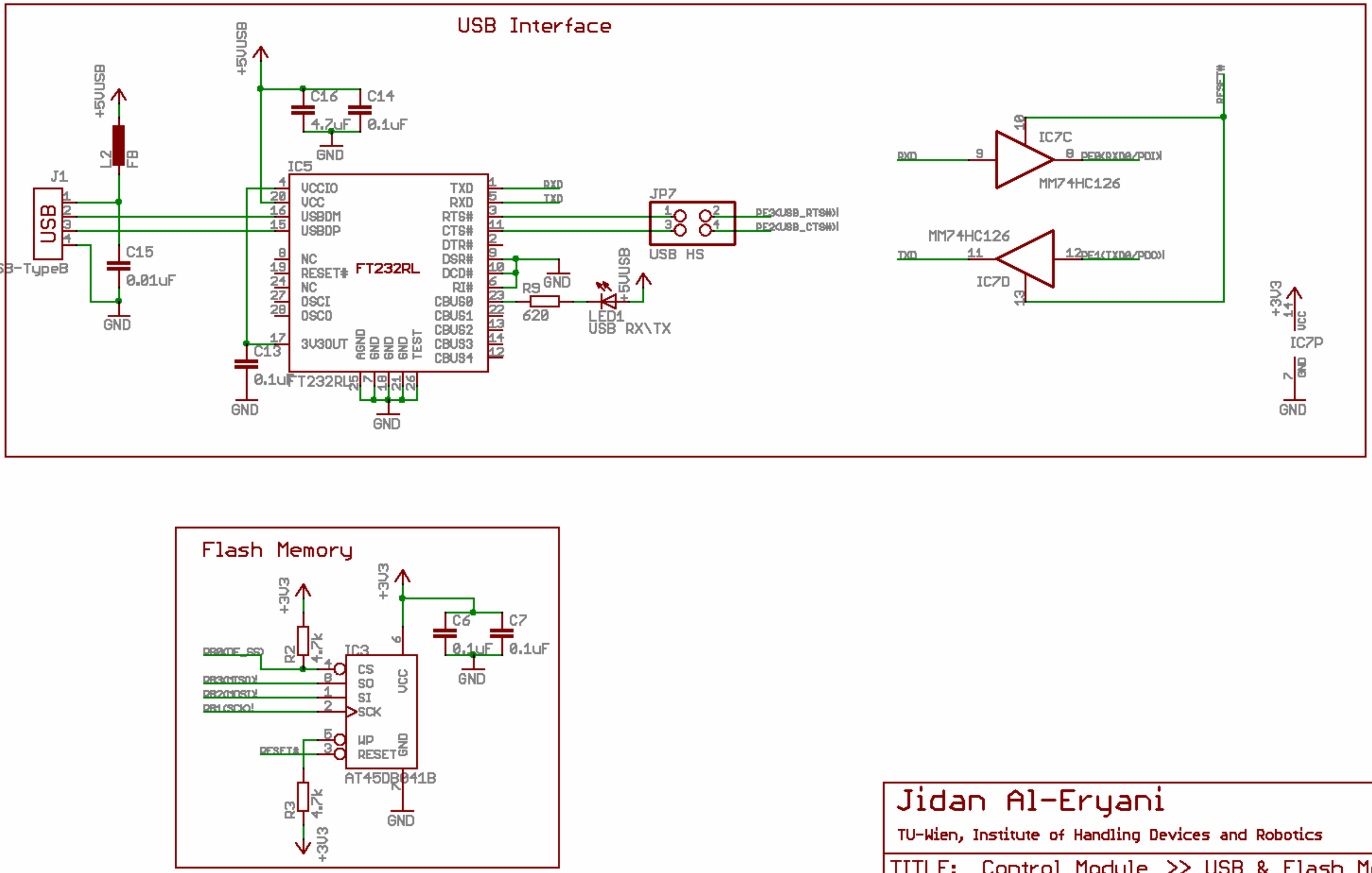


A.2 Control Module

Control module parts list

Quantity	Part	Value	Package
10	C3, C4, C5, C6, C7, C13, C14, C20, C25, C26	0.1uF	C1206
1	C15	0.01uF	C1206
1	C16	4.7uF	C1206
2	C17, C18	10uF	C-PANS_B
1	C19	220uF	C-8
1	D1	1N5820	DO-201AD
1	IC1	MEGA128-A	TQFP64
1	IC2	MAX811	SOT-143
1	IC3	AT45DB041B	SOIC8-0.209
1	IC5	FT232RL	SSOP-28
1	IC6	LM1085IS-3.3	TO-263-3
1	IC7	MM74HC126	SO14
1	J1	USB-TypeB	USB-TYPB
1	J2	DC-JACK	DC-JACK(JSBJ4)
1	JP1	AREF	1X02
1	JP2	Header 4	2X06
1	JP3	Header 1	2X08
1	JP4	Header 5	2X07
1	JP6	ISP	2X03
1	JP7	USB HS	2X02
1	JP9	Modules PWR - BOT	2X3-SMT
1	JP10	Modules PWR - TOP	2X3-SMT
1	JP11	Header 3	2X8-SMT
1	JP13	Header 2 - TOP	2X9-SMT
1	JP14	Header 2 - BOT	2X9-SMT
2	L1, L2	FB	L-805
1	LED1	USB RX\TX	LED3MM
1	LED2	PWR ON	LED3MM
1	Q1	8MHz	HC49U-V
1	Q2	32.7KHz	CFPX-56
1	R1	47k	PT10-5X2.5
3	R2, R3, R26	4.7k	R1206
2	R4, R25	220	R1206
1	R6	10k	R1206
1	R9	620	R1206
1	R23	270	R1206
1	S1	RESET	B3F-10XX
1	S2	PB1	B3F-10XX
1	S3	PB2	B3F-10XX


Headers
Microcontroller
Supervisory IC
ADC Input Stage
Push Buttons
MEGA128-A
MAX811
! : signal is used more than once.
Jidan Al-Eryani
TU-Wien, Institute of Handling Devices and Robotics
TITLE: Control Module >> Microcontroller Unit
Document Number:
REV:
Date: 13.05.2007 19:49:29
Sheet: 1/3



Jidan Al-Eryani

TU-Wien, Institute of Handling Devices and Robotics

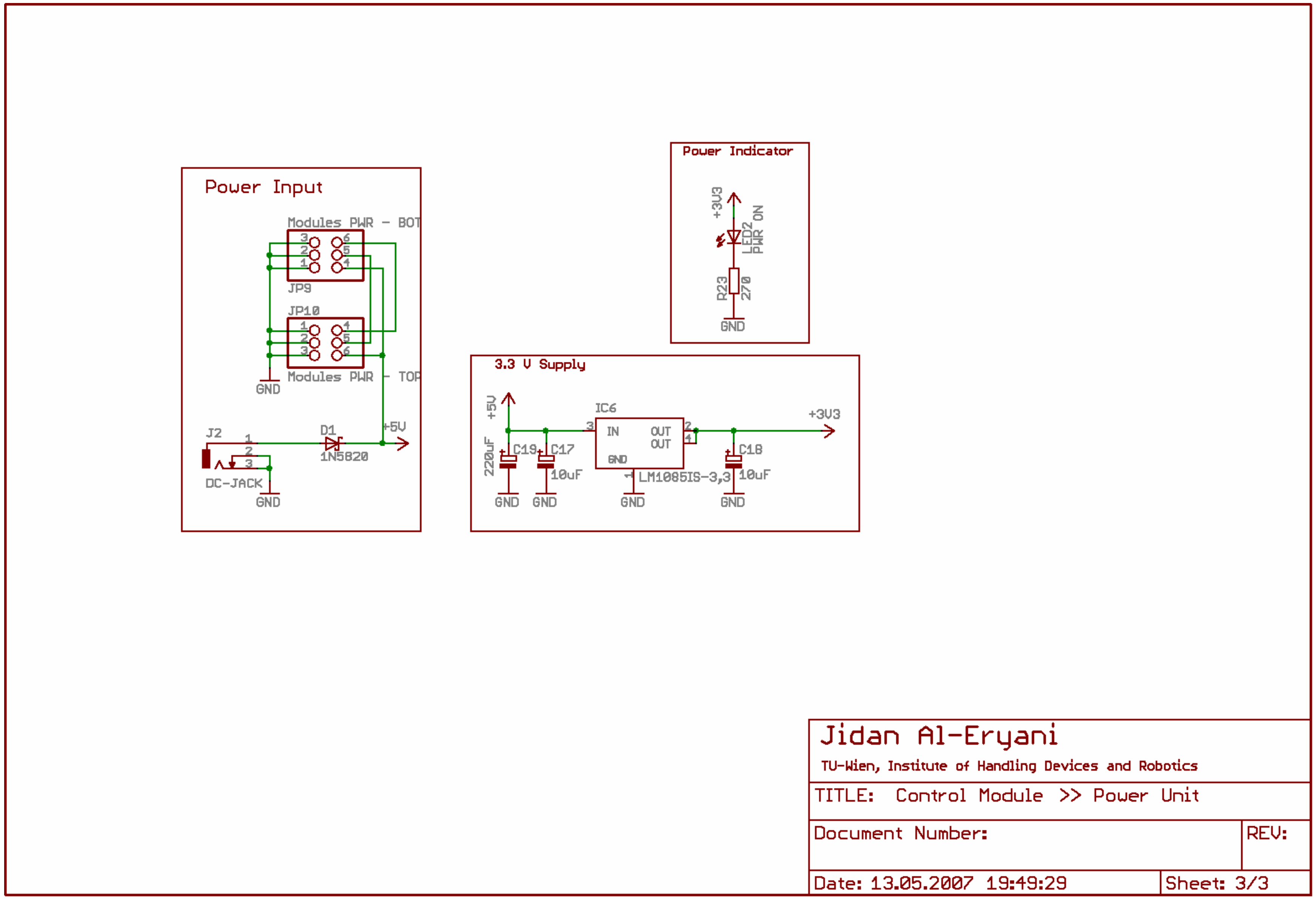
TITLE: Control Module >> USB & Flash Memory

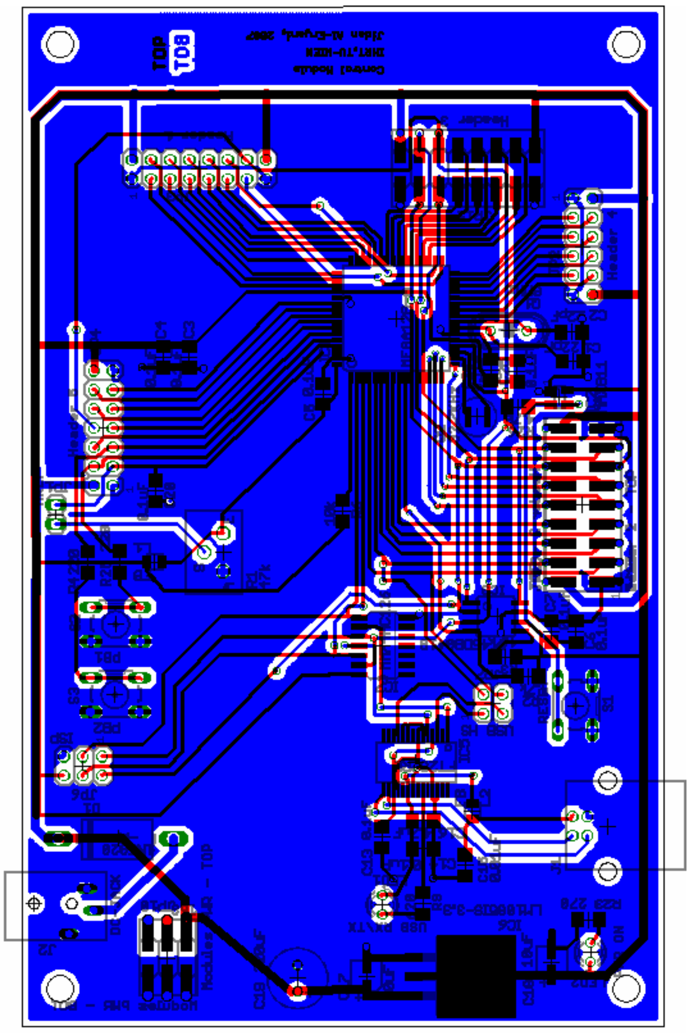
Document Number:

REV:

Date: 13.05.2007 19:49:29

Sheet: 2/3

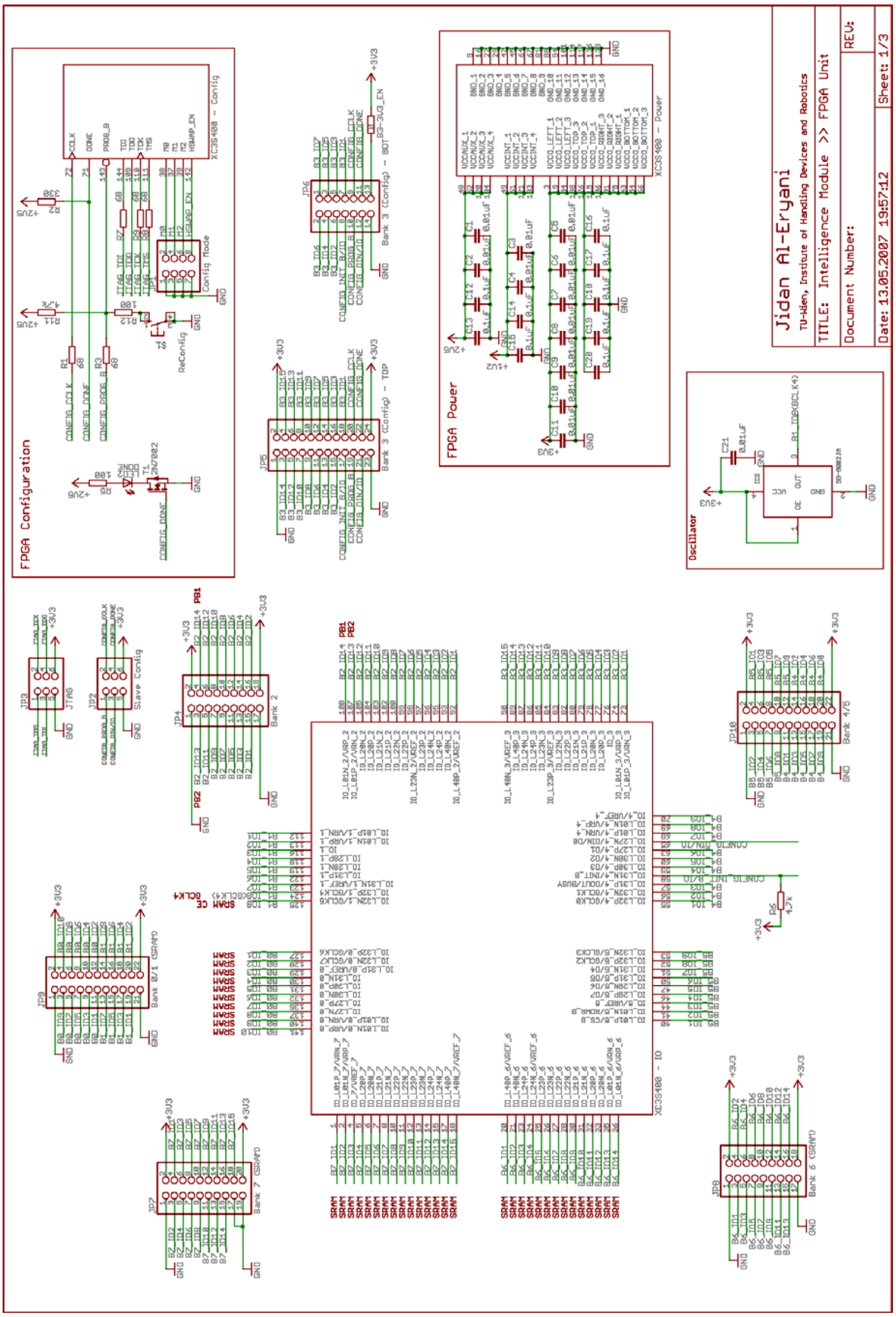

Power Input
Modules PWR - BOT
JP9
JP10
Modules PWR - TOP
GND
J2
DC-JACK
D1
1N5820
+5V
GND
Power Indicator
+3V3
LED2
PWR ON
R23
270
GND
3.3 V Supply
+5V
220uF
C19
C17
10uF
IC6
IN
OUT
OUT
GND
LM1085IS-3,3
C18
10uF
+3V3
GND
Jidan Al-Eryani
TU-Wien, Institute of Handling Devices and Robotics
TITLE: Control Module >> Power Unit
Document Number:
REV:
Date: 13.05.2007 19:49:29
Sheet: 3/3

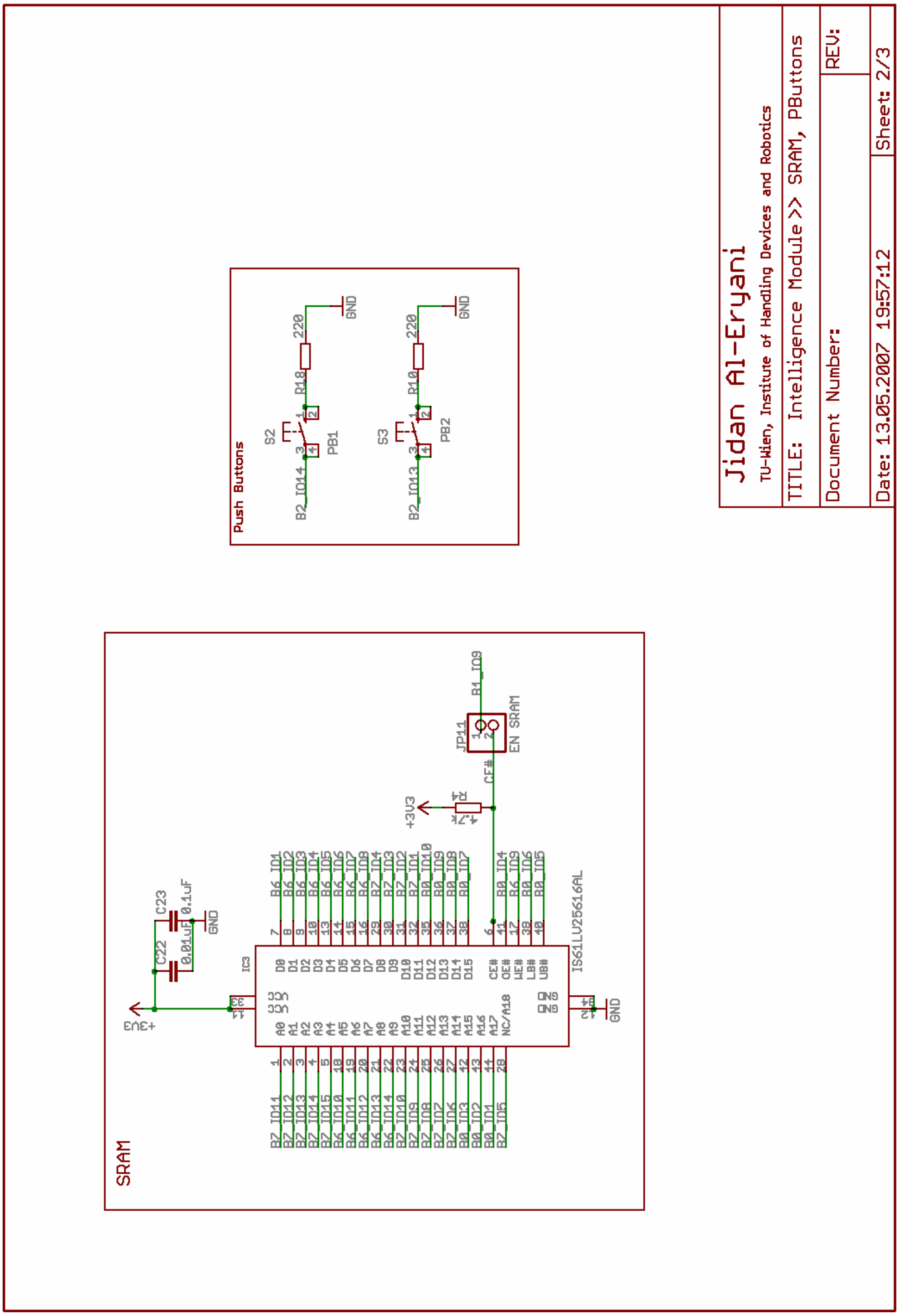


A.3 Intelligence Module

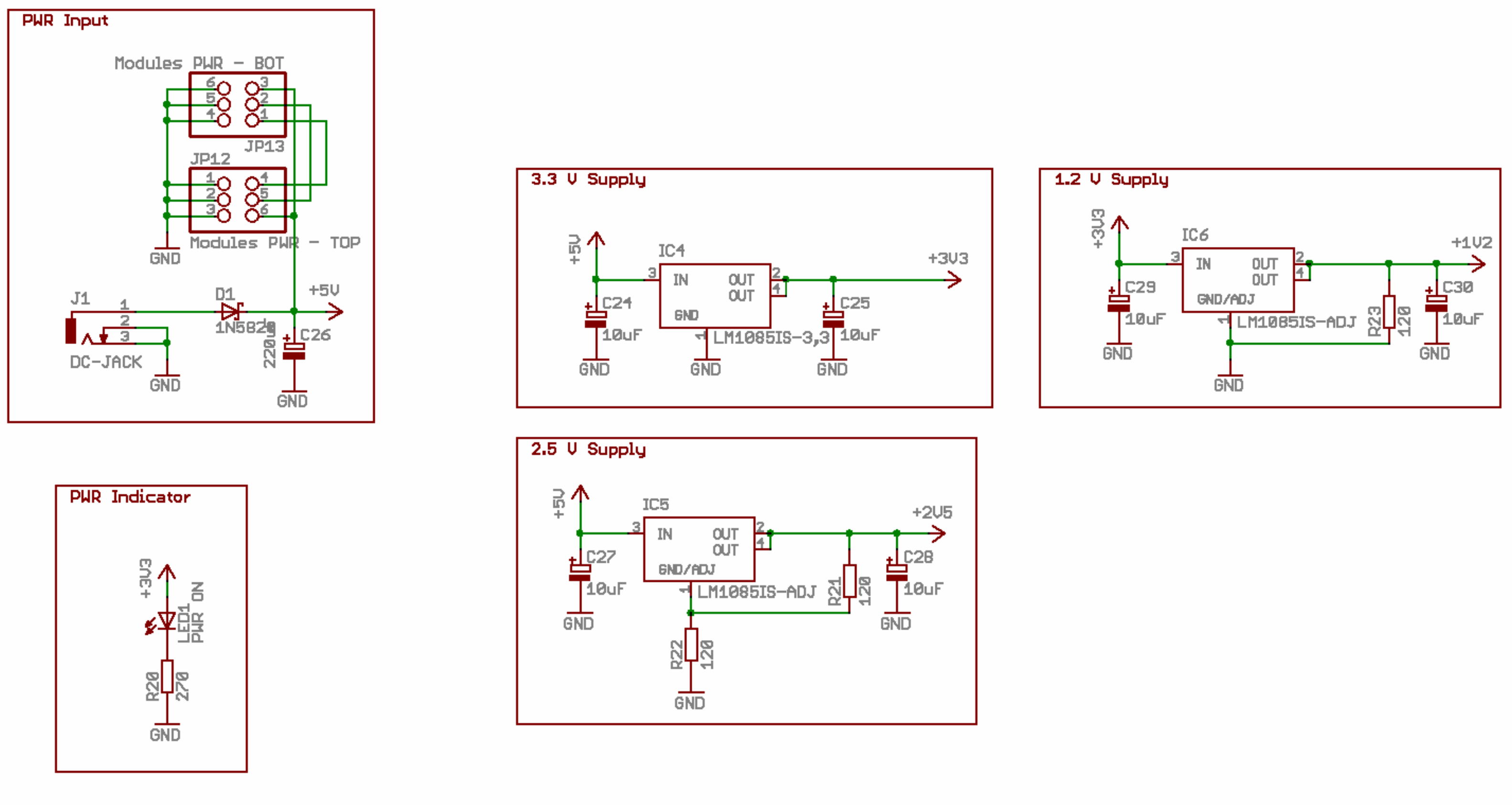
Intelligence module parts list

Quantity	Part	Value	Package
1	B3-3V3_EN	0	R1206
13	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C21, C22	0.01uF	C0603
8	C12, C13, C14, C15, C16, C17, C18, C23	0.1uF	C0603
2	C19, C20	0.1uF	C0805
6	C24, C25, C27, C28, C29, C30	10uF	C-PANS_B
1	C26	220uF	C-8
1	D1	1N5820	DO-201AD
1	IC1	XC3S400-TQ144	TQ144
1	IC2	SG-8002JA	SG-8002JA
1	IC3	IS61LV25616AL	TSOP-44
1	IC4	LM1085IS-3.3	TO-263-3
2	IC5, IC6	LM1085IS-ADJ	TO-263-3
1	J1	DC-JACK	DC-JACK(JSBJ4)
1	JP1	Config Mode	2X04
1	JP2	Slave Config	2X03
1	JP3	JTAG	2X03
1	JP4	Bank 2	2X09
1	JP5	Bank 3 (Config) - TOP	2X12-SMT
1	JP6	Bank 3 (Config) - BOT	2X7-SMT
1	JP7	Bank 7 (SRAM)	2X10
1	JP8	Bank 6 (SRAM)	2X09
1	JP9	Bank 0/1 (SRAM)	2X11
1	JP10	Bank 4/5	2X11
1	JP11	EN SRAM	1X02
1	JP12	Modules PWR - TOP	2X3-SMT
1	JP13	Modules PWR - BOT	2X3-SMT
1	LED1	PWR ON	LED3MM
1	LED2	DONE	LED3MM
5	R1, R3, R7, R8, R9	68	R1206
1	R2	330	R1206
3	R4, R6, R11	4.7k	R1206
2	R5, R12	100	R1206
2	R10, R18	220	R1206
1	R20	270	R1206
3	R21, R22, R23	120	R1206
1	S1	ReConfig	B3F-10XX
1	S2	PB1	B3F-10XX
1	S3	PB2	B3F-10XX
1	T1	2N7002	SOT-23


FPGA Configuration
FPGA Power
Oscillator
XC3S400 - Config
XC3S400 - Power
XC3S400 - IO
Jidan Al-Eryani
TU-Wien, Institute of Handling Devices and Robotics
TITLE: Intelligence Module >> FPGA Unit
Document Number:
REV:
Date: 13.05.2007 19:57:12
Sheet: 1/3



Jidan Al-Eryani
TU-Wien, Institute of Handling Devices and Robotics
TITLE: Intelligence Module >> SRAM, PButtons
Document Number:
REV:
Date: 13.05.2007 19:57:12
Sheet: 2/3
Push Buttons
S2
PB1
S3
PB2
R18
R10
220
GND
B2_IO14
B2_IO13
SRAM
IC3
IS61LV25616AL
C22
0.01uF
C23
0.1uF
+3U3
R1
4.7K
CE#
JP11
EN SRAM
B1_IO9



Jidan Al-Eryani

TU-Wien, Institute of Handling Devices and Robotics

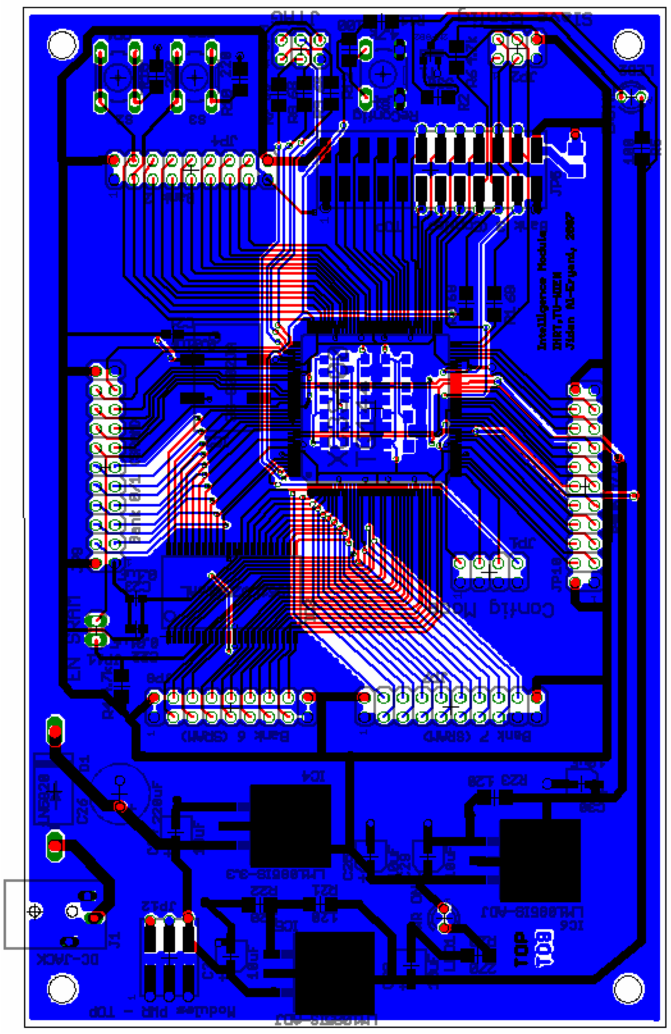
TITLE: Intelligence Module >> Power Unit

Document Number:

REV:

Date: 13.05.2007 19:57:12

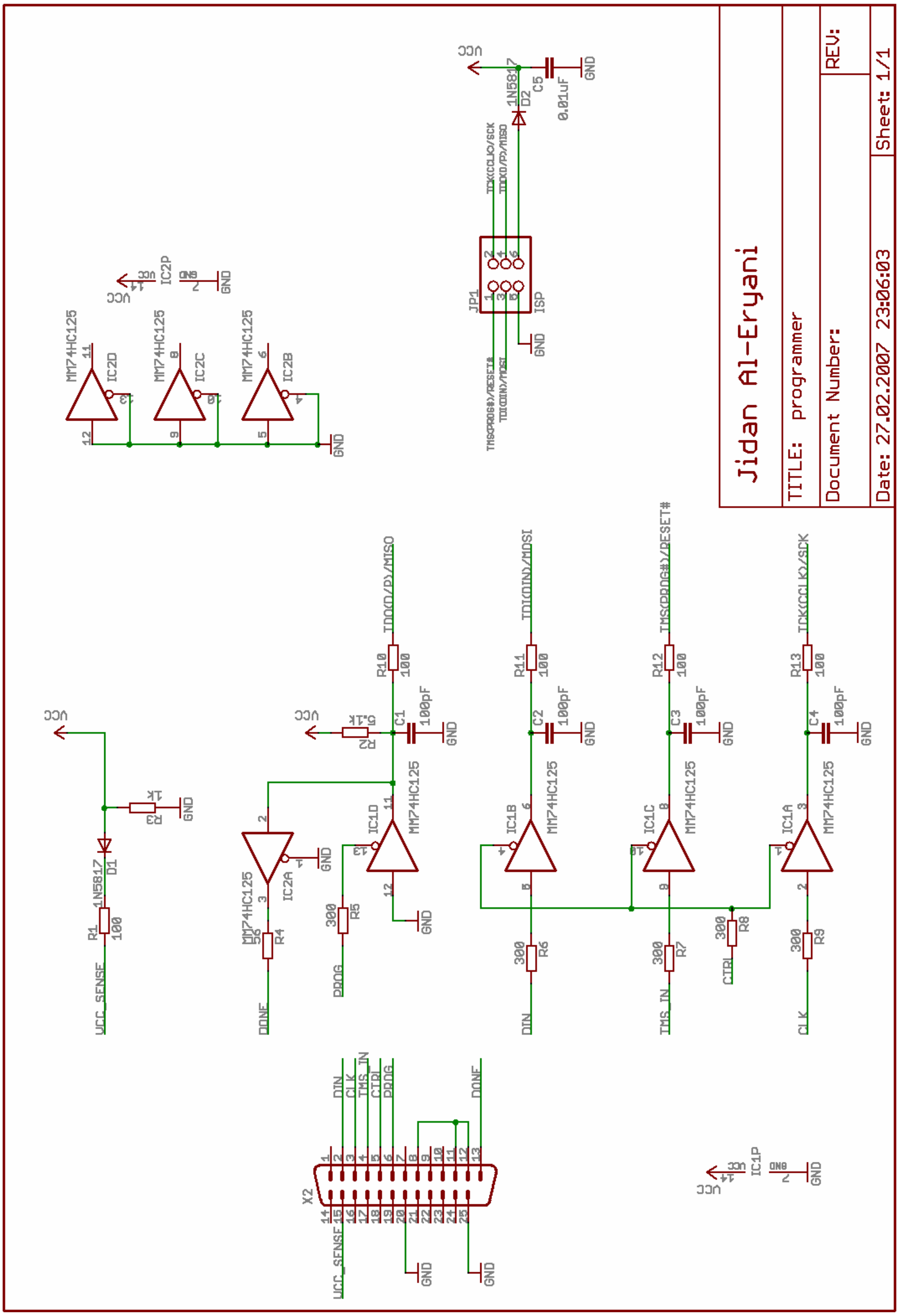
Sheet: 3/3

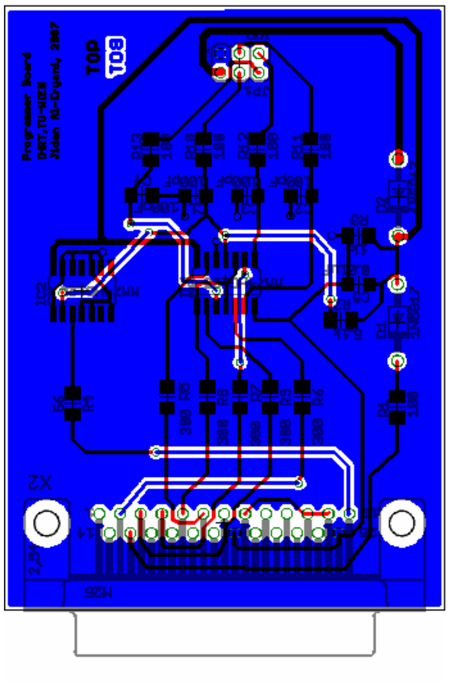


A.4 Programmer Board

Programmer board parts list

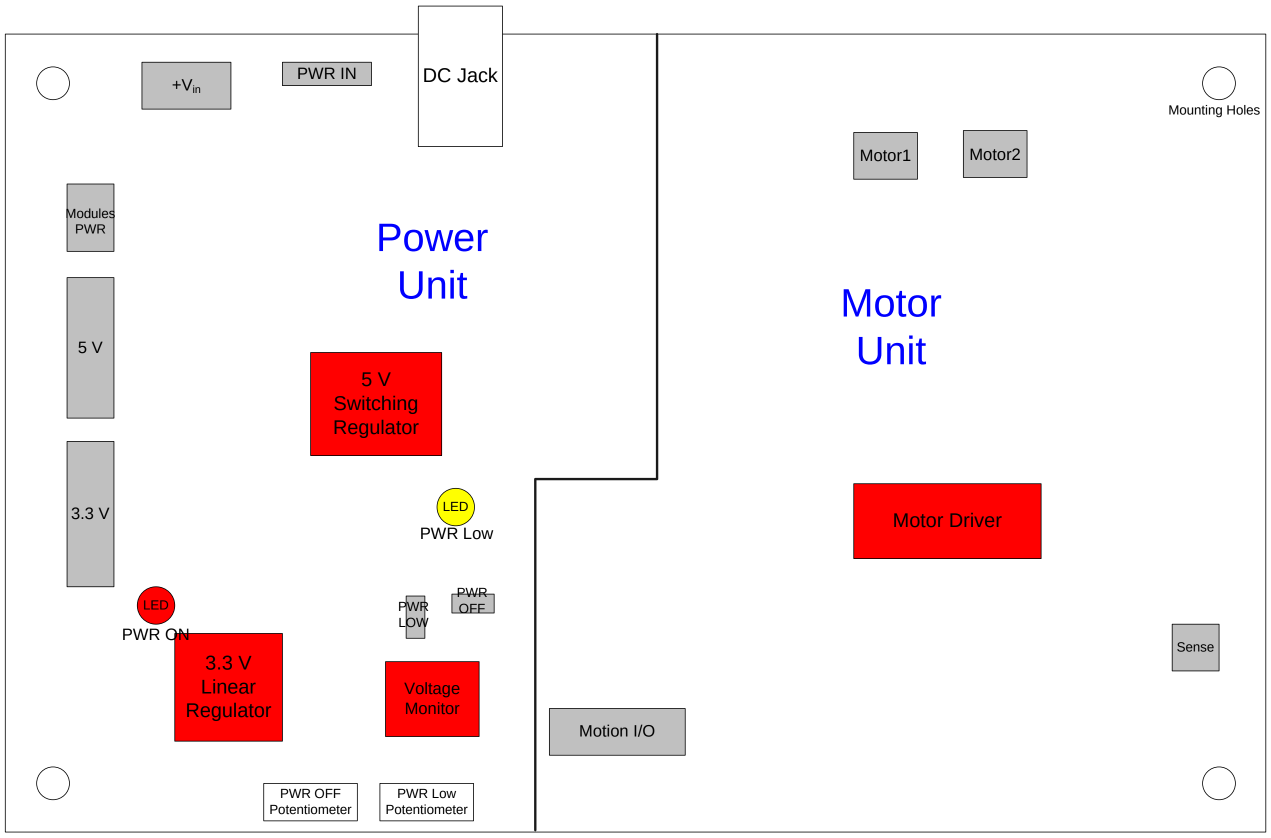
Quantity	Part	Value	Package
4	C1, C2, C3, C4	100pF	C1206
1	C5	0.01uF	C1206
2	D1, D2	1N5817	DO41-10
2	IC1, IC2	MM74HC125	SO14
1	JP1	ISP	2X03
5	R1, R10, R11, R12, R13	100	M1206
1	R2	5.1k	M1206
1	R3	1k	M1206
1	R4	56	M1206
5	R5, R6, R7, R8, R9	300	M1206
1	X2	DB25-M	M25H


Jidan Al-Eryani
TITLE: programmer
Document Number:
REV:
Date: 27.02.2007 23:06:03
Sheet: 1/1
VCC
GND
VCC_SENSE
R1 100
D1 1N5817
R3 1k
DONE
R4 56
IC2A MM74HC125
PROG
R5 300
R2 5.1k
IC1D MM74HC125
C1 100pF
R10 100
TDO(D/P)/MISO
DIN
R6 300
IC1B MM74HC125
C2 100pF
R11 100
TDI(DIN)/MOSI
TMS_IN
R7 300
CTRL
R8 300
IC1C MM74HC125
C3 100pF
R12 100
TMS(PROG#)/RESET#
CLK
R9 300
IC1A MM74HC125
C4 100pF
R13 100
TCK(CCLK)/SCK
IC2D MM74HC125
IC2C MM74HC125
IC2B MM74HC125
IC2P
IC1P
JP1
ISP
TMS(PROG#)/RESET#
TDI(DIN)/MOSI
TCK(CCLK)/SCK
TDO(D/P)/MISO
D2 1N5817
C5 0.01uF
X2



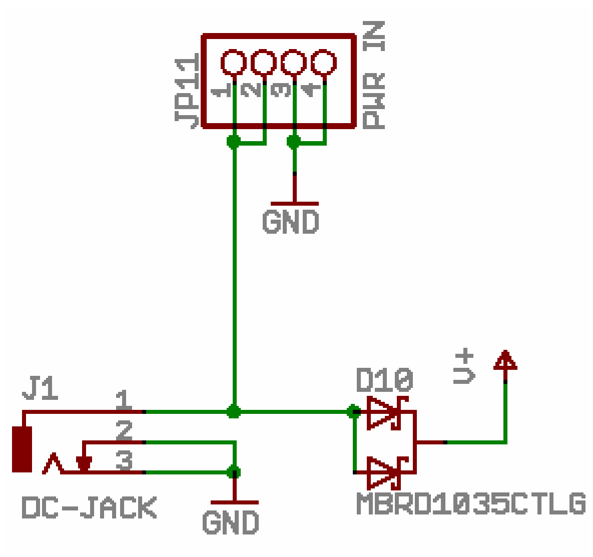
Appendix B - Headers Pinouts

B.1 Power & Motion Module



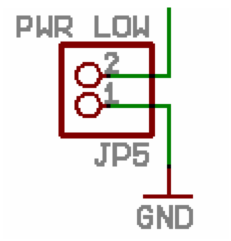
B.1.1 Power Unit

Power Input



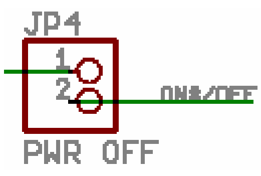
PWR IN (1x4)	
Pin	**Function(s)**
1	+ External power source
2	+ External power source
3	GND
4	GND

PWR LOW



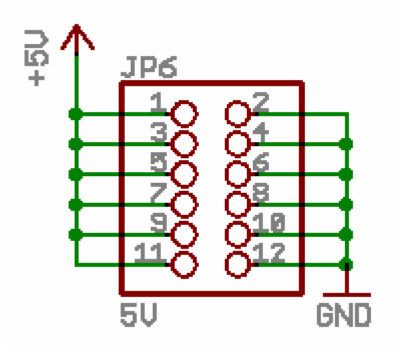
PWR LOW (1x2)	
Pin	**Function(s)**
1	GND
2	Output 1 from the voltage monitor (see *4.1.2 Low Voltage Monitor*)

PWR OFF



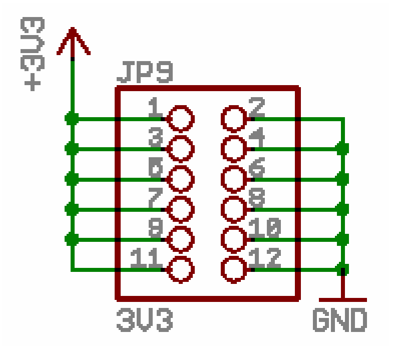
PWR OFF (1x2)	
Pin	**Function(s)**
1	Output 2 from the voltage monitor (see *4.4.2. Low Voltage Monitor*)
2	To ON#/OFF pin on the 5 V switching regulator (see *4.1.1 Power Supplies*)

5 V Header



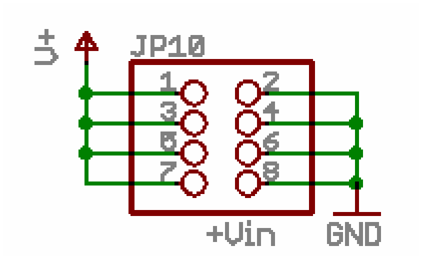
5 V header (2x6)	
Pin	**Function(s)**
1	+5 V
2	GND
3	+5 V
4	GND
5	+5 V
6	GND
7	+5 V
8	GND
9	+5 V
10	GND
11	+5 V
12	GND

3.3 V Header



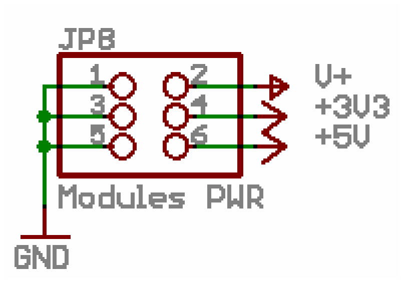
3.3 V header (2x6)	
Pin	**Function(s)**
1	+3.3 V
2	GND
3	+3.3 V
4	GND
5	+3.3 V
6	GND
7	+3.3 V
8	GND
9	+3.3 V
10	GND
11	+3.3 V
12	GND

V_{in} Header



V_{in} header (2x4)	
Pin	**Function(s)**
1	V+ (=External power source – Schottky diode forward voltage)
2	GND
3	V+
4	GND
5	V+
6	GND
7	V+
8	GND

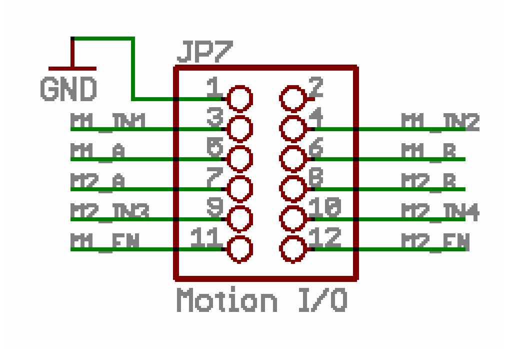
Modules PWR



Modules PWR (2x3)	
Pin	**Function(s)**
1	GND
2	V+
3	GND
4	3.3 V
5	GND
6	5 V

B.1.2 Motion Unit

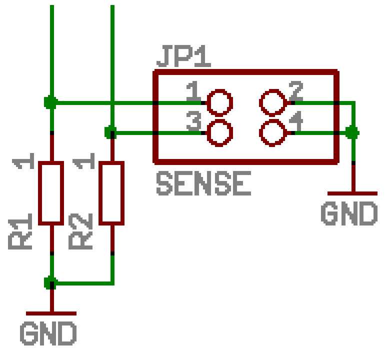
Motion I/O



Motion I/O (2x6)		
Pin	**Name**	**Function(s)**
1	GND	Power
2	NC	-
3	M1_IN1	Motor 1, Input 1 (M1_IN1)
4	M1_IN2	Motor 1, Input 2 (M1_IN2)
5	M1_A	Motor 1, Encoder Channel A (M1_A)
6	M2_B	Motor 2, Encoder Channel B (M2_B)
7	M2_A	Motor 2, Encoder Channel A (M2_A)
8	M2_B	Motor 2, Encoder Channel B (M2_B)

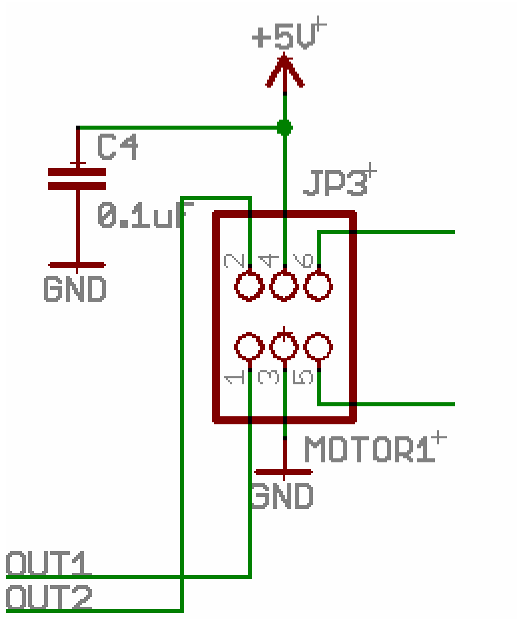
9	M2_IN3	Motor 2, Input 3 (M2_IN3)
10	M2_IN4	Motor 2, Input 4 (M2_IN4)
11	M1_EN	Motor 1, Enable (M1_EN)
12	M2_EN	Motor 2, Enable (M2_EN)

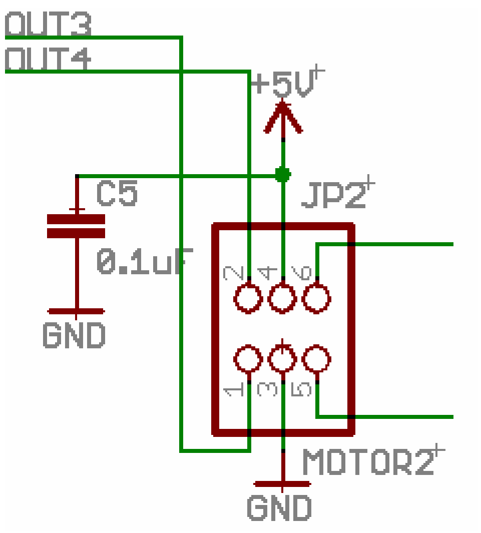
SENSE



SENSE (2x2)	
Pin	**Function(s)**
1	SEN_B from Motor driver (see *4.2.1 Motor Driver*)
2	GND
3	SEN_A from Motor driver (see *4.2.1 Motor Driver*)
4	GND

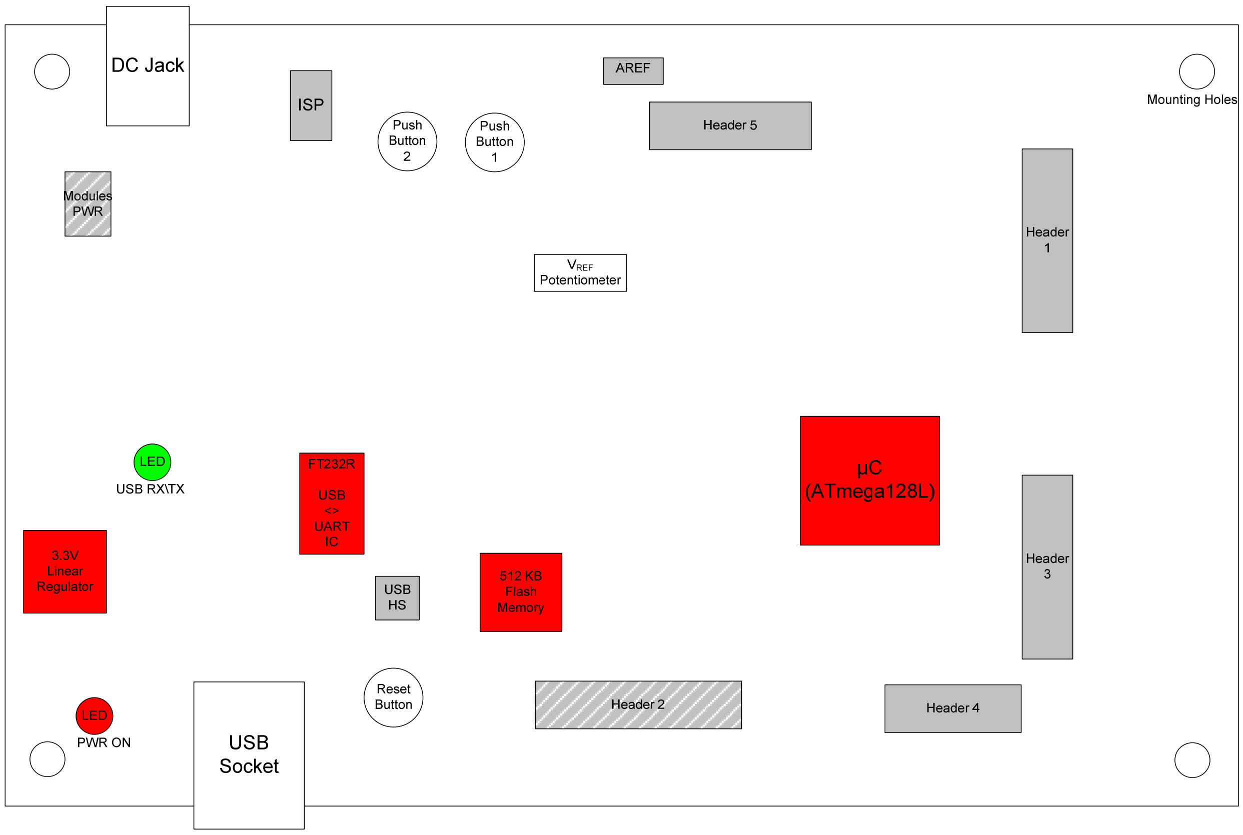
MOTOR1 & 2





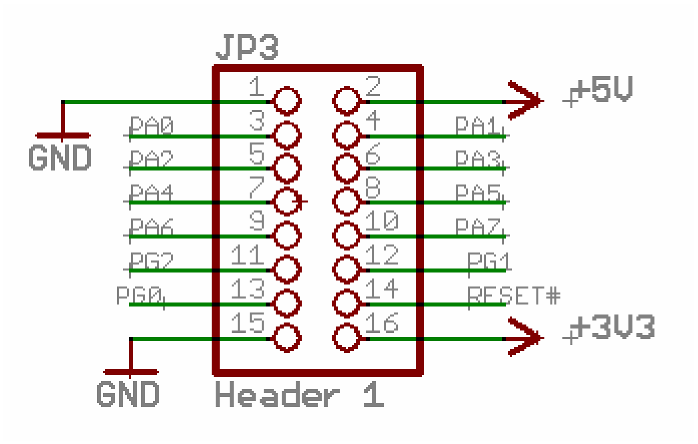
MOTOR1&2 (3x2)	
Pin	**Function(s)**
1	OUT1(3) from Motor driver (see *4.2.1 Motor Driver*)
2	OUT2(4) from Motor driver (see *4.2.1 Motor Driver*)
3	GND
4	5 V
5	To Actuator (e.g. DC Motor)
6	To Actuator

B.2 Control Module



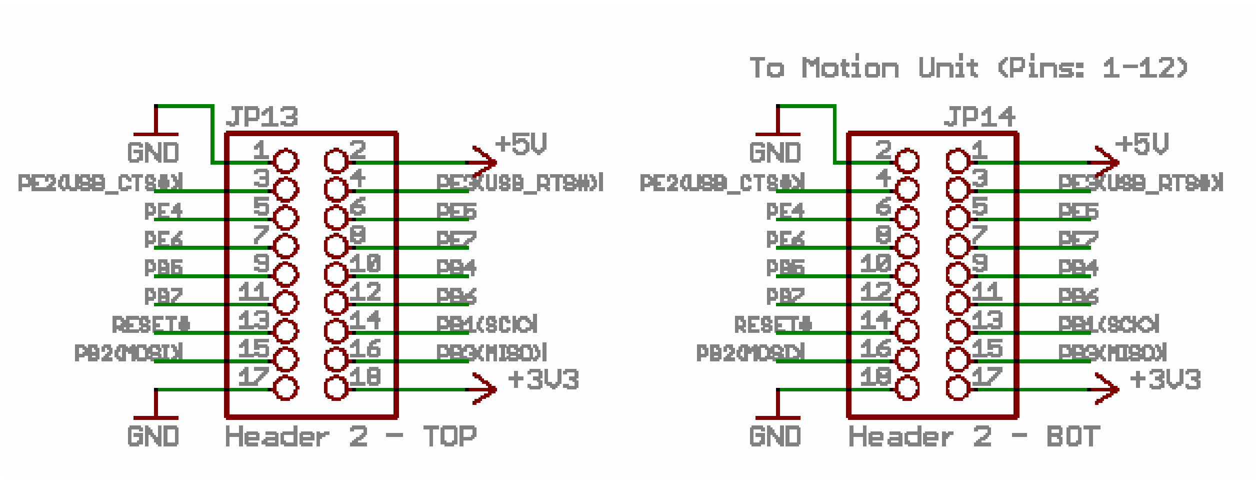
For more detailed description of the µC pins, please see the ATmega128L datasheet ([ATM06]).

Header 1



Header 1 (2x8)		
Pin	**Port**	**Function(s)**
1	-	GND
2	-	5 V
3	PA0	User-I/O, AD0 (External memory interface address and data bit 0)
4	PA1	User-I/O, AD1 (External memory interface address and data bit 1)
5	PA2	User-I/O, AD2 (External memory interface address and data bit 2)
6	PA3	User-I/O, AD3 (External memory interface address and data bit 3)
7	PA4	User-I/O, AD4 (External memory interface address and data bit 4)
8	PA5	User-I/O, AD5 (External memory interface address and data bit 5)
9	PA6	User-I/O, AD6 (External memory interface address and data bit 6)
10	PA7	User-I/O, AD7 (External memory interface address and data bit 7)
11	PG2	User-I/O, ALE (Address Latch Enable to external memory)
12	PG1	User-I/O, RD (Read strobe to external memory)
13	PG0	User-I/O, WR (Write strobe to external memory)
14	-	Reset
15	-	GND
16	-	3.3 V

Header 2



Header 2 (2x9)		
Pin(Top)	**Port**	**Function(s)**
1	-*	GND
2	-*	5 V
3	PE2* / ***	User-I/O, AIN0/XCK0 (Analogue Comparator Positive Input or USART0 external clock I/O)
4	PE3* / ***	User-I/O, AIN1/OC3A (Analogue Comparator Negative Input or Output Compare and PWM Output A for Timer3)
5	PE4*	User-I/O, INT4/OC3B (External Interrupt4 Input or Output Compare and PWM Output B for Timer3)
6	PE5*	User-I/O, INT5/OC3C (External Interrupt 5 Input or Output Compare and PWM Output C for Timer3)
7	PE6*	User-I/O, INT6/ T3 (External Interrupt 6 Input or Timer3 Clock Input)
8	PE7*	User-I/O, INT7/ICP3 (External Interrupt 7 Input or Timer3 Input Capture Pin)
9	PB5*	User-I/O, OC1A (Output Compare and PWM Output A for Timer/Counter1)
10	PB4*	User-I/O, OC0 (Output Compare and PWM Output for Timer/Counter0)
11	PB7*	User-I/O, OC2/OC1C(1) (Output Compare and PWM Output for Timer/Counter2 or Output

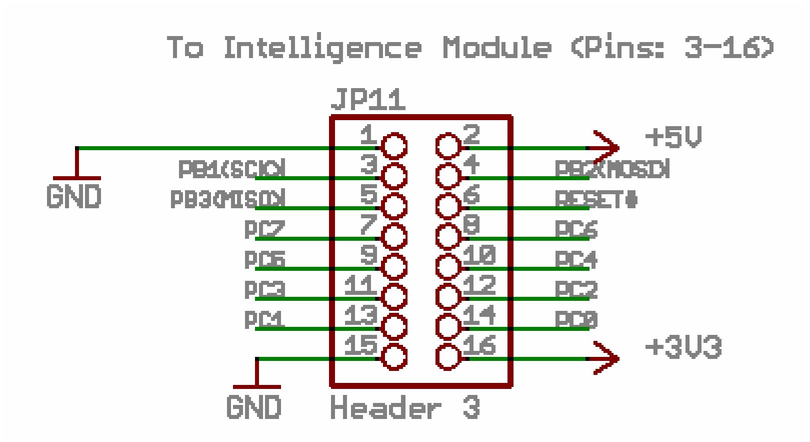
		Compare and PWM Output C for Timer/Counter1)
12	PB6*	User-I/O, OC1B (Output Compare and PWM Output B for Timer/Counter1)
13	-	Reset
14	PB1**	SCK (SPI Bus Serial Clock)
15	PB2**	MOSI (SPI Bus Master Output/Slave Input)
16	PB3**	MISO (SPI Bus Master Input/Slave Output)
17	-	GND
18	-	3.3 V

* Interface pins to the motion unit in the Power & Motion module

** SPI bus pins.

*** Used for USB handshaking, if a jumper is placed on the header USB HS.

Header 3



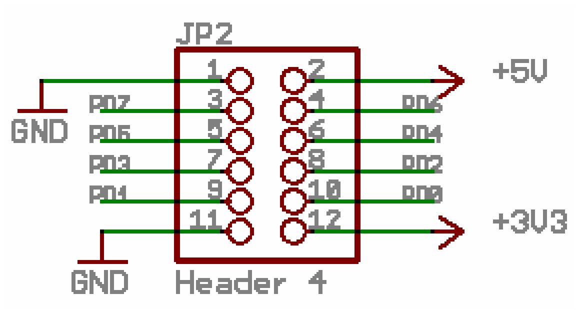
Header 3 (2x8)		
Pin	**Port**	**Function(s)**
1	-*	GND
2	-*	5 V
3	PB1* / **	SCK (SPI Bus Serial Clock)
4	PB2* / **	MOSI (SPI Bus Master Output/Slave Input)
5	PB3* / **	MISO (SPI Bus Master Input/Slave Output)
6	-*	Reset
7	PC7*	User-I/O, A15 (Address high byte for the External Memory Interface)
8	PC6*	User-I/O, A14 (Address high byte for the External Memory Interface)
9	PC5*	User-I/O, A13 (Address high byte for the External Memory Interface)

10	PC4*	User-I/O, A12 (Address high byte for the External Memory Interface)
11	PC3*	User-I/O, A11 (Address high byte for the External Memory Interface)
12	PC2*	User-I/O, A10 (Address high byte for the External Memory Interface)
13	PC1*	User-I/O, A9 (Address high byte for the External Memory Interface)
14	PC0*	User-I/O, A8 (Address high byte for the External Memory Interface)
15	-*	GND
16	-*	3.3 V

* Interface pins to the Intelligence module.

** SPI bus pins.

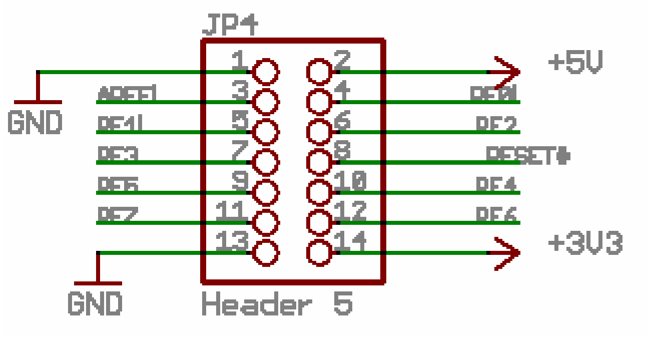
Header 4



Header 4 (2x6)		
Pin	**Port**	**Function(s)**
1	-	GND
2	-	5 V
3	PD7	User-I/O, T2 (Timer/Counter2 Clock Input)
4	PD6	User-I/O, T1 (Timer/Counter1 Clock Input)
5	PD5	User-I/O, XCK1 (USART1 External Clock Input/Output)
6	PD4	User-I/O, ICP1 (Timer/Counter1 Input Capture Pin)
7	PD3	User-I/O, INT3/TXD1 (External Interrupt3 Input or UART1 Transmit Pin)
8	PD2	User-I/O, INT2/RXD1 (External Interrupt2 Input or UART1 Receive Pin)
9	PD1*	User-I/O, INT1/SDA (External Interrupt1 Input or TWI(I^{2}C) Serial Data)
10	PD0*	User-I/O, INT0/SCL (External Interrupt0 Input or TWI(I^{2}C) Serial Clock)
11	-	GND
12	-	3.3 V

*I^{2}C bus pins: this bus can be used to connect slow devices such as an LCD, thereby saving pins.

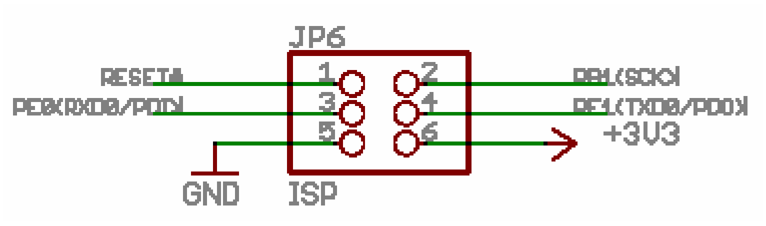
Header 5



Header 5 (2x7)		
Pin	**Port**	**Function(s)**
1	-	GND
2	-	5 V
3	-	AREF
4	PF0*	User-I/O, ADC0 (ADC input channel 0)
5	PF1*	User-I/O, ADC1 (ADC input channel 1)
6	PF2	User-I/O, ADC2 (ADC input channel 2)
7	PF3	User-I/O, ADC3 (ADC input channel 3)
8	-	Reset
9	PF5	User-I/O, ADC5/TMS (ADC input channel 5 or JTAG Test Mode Select)
10	PF4	User-I/O, ADC4/TCK (ADC input channel 4 or JTAG Test Clock)
11	PF7	User-I/O, ADC7/TDI (ADC input channel 7 or JTAG Test Data Input)
12	PF6	User-I/O, ADC6/TDO (ADC input channel 6 or JTAG Test Data Output)
13	-	GND
14	-	3.3 V

*Connected to push buttons

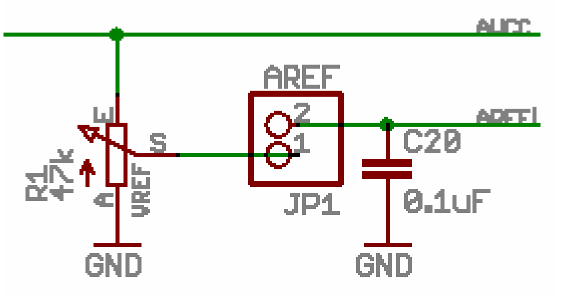
ISP



ISP (2x3)		
Pin	**Port**	**Function(s)**
1	-*	Reset
2	-*	SCK
3	PE0*	PDI
4	PE1*	PDO
5	-*	GND
6	-*	3.3 V

*ISP pins: this header is used for programming the µC

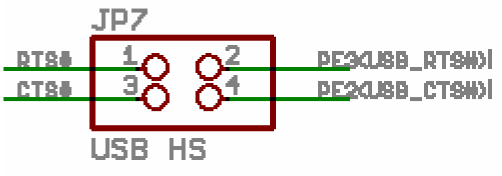
ADC



ADC (2x1)*	
Pin	**Function(s)**
1	VREF
2	AREF

See section *5.4 USB Interface*.

USB HS

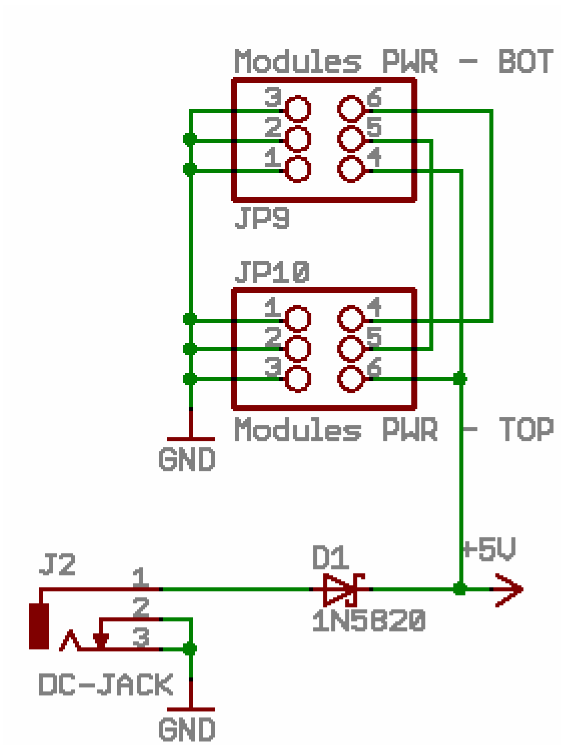


USB HS (2x2)*	
Pin	**Function(s)**
1	RTS# (from USB-UART converter IC)

2	PE3
3	CTS# (from USB-UART converter IC)
4	PE2

See section *5.4 USB Interface*.

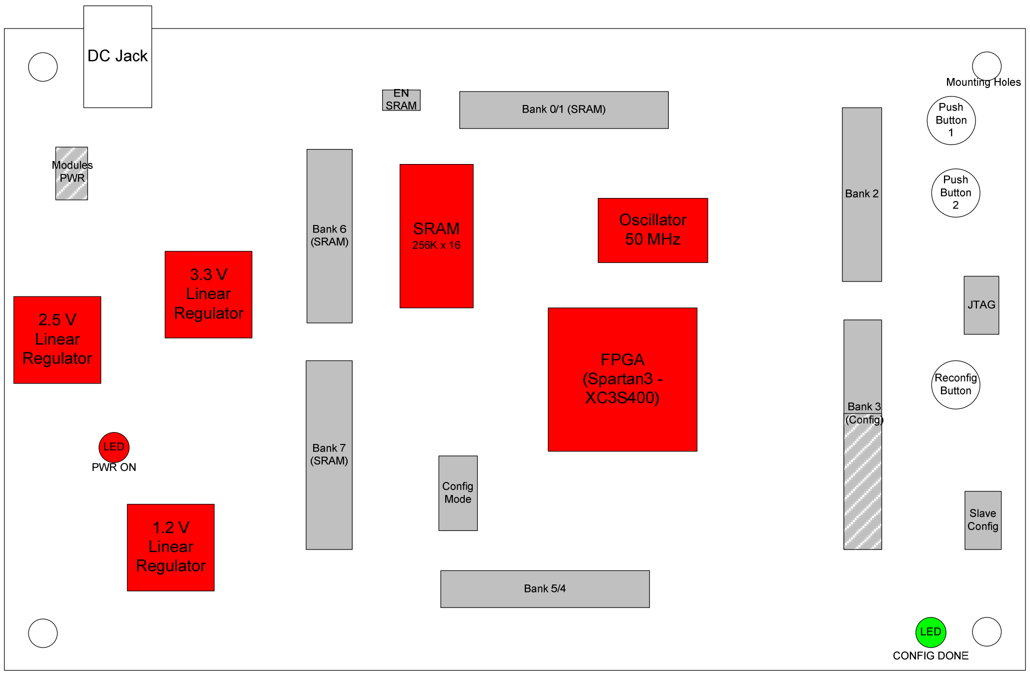
Modules PWR



Modules PWR (2x3)		
Pin(Top)	**Port**	**Function(s)**
1	-*	GND
2	-*	GND
3	-*	GND
4	-*	+V_{in}
5	-*	3.3 V
6	-*	5 V

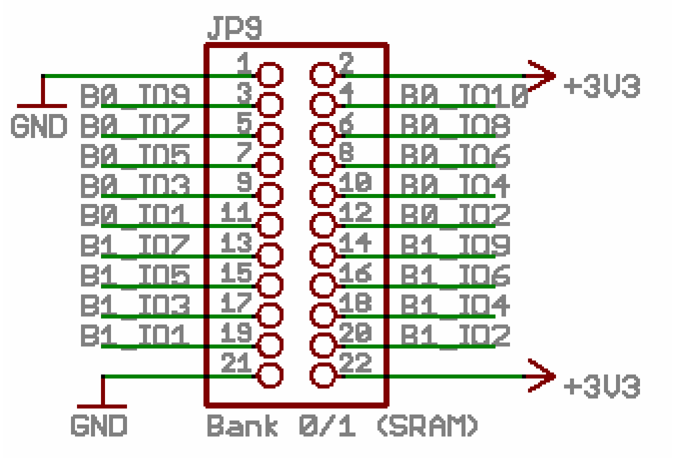
*Power pins from the Power & Motion module; if the module is used in stand-alone, apply the power through the DC-Jack.

B.3 Intelligence Module



For more detailed description of the FPGA pins, please see the Spartan-3 XC3S400 datasheet ([XIL06a]).

Bank 0/1 Header



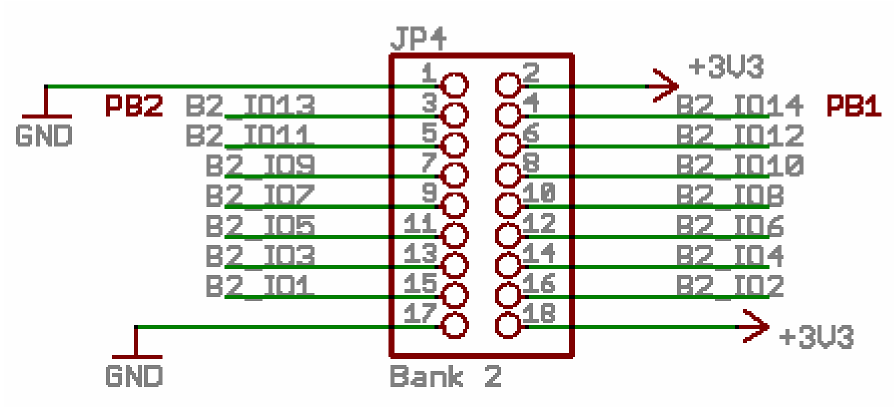
Bank 0/1 Header (2x11)			
Pin	**Name**	**FPGA Pin**	**Function**
1	-	-	GND

2	-	-	3.3 V
3	B0_IO9*	140	User-I/O, DCI
4	B0_IO10*	141	User-I/O, DCI
5	B0_IO7*	135	User-I/O
6	B0_IO8*	137	User-I/O
7	B0_IO5*	131	User-I/O
8	B0_IO6*	132	User-I/O
9	B0_IO3*	129	User-I/O, VREF
10	B0_IO4*	130	User-I/O
11	B0_IO1*	127	User-I/O, GCLK
12	B0_IO2*	128	User-I/O, GCLK
13	B1_IO7	123	User-I/O, VREF
14	B1_IO9**	125	User-I/O, GCLK
15	B1_IO5	119	User-I/O
16	B1_IO6	122	User-I/O
17	B1_IO3	116	User-I/O
18	B1_IO4	118	User-I/O
19	B1_IO1	112	User-I/O, DCI
20	B1_IO2	113	User-I/O, DCI
21	-	-	GND
22	-	-	3.3 V

*Used by SRAM when a jumper is placed on the header EN SRAM.

**Connected to the header EN SRAM.

Bank 2 Header

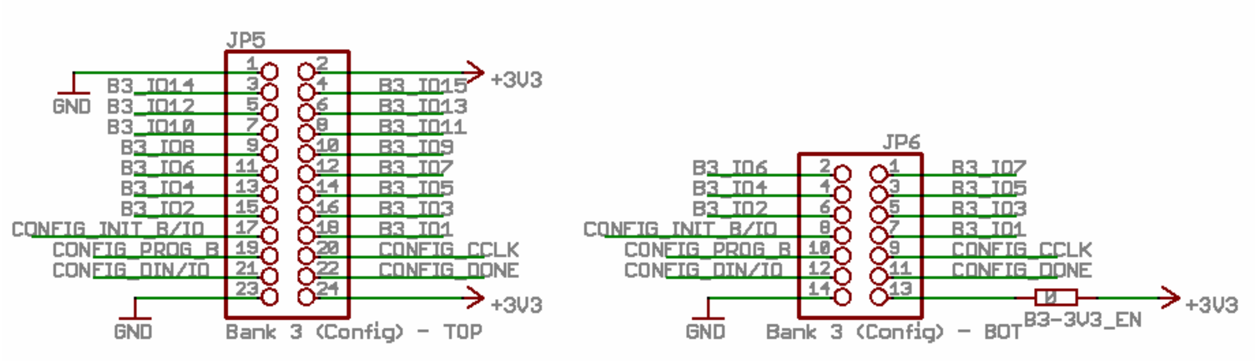


Bank 2 Header (2x9)			
Pin	**Name**	**FPGA Pin**	**Function**

1	-	-	GND
2	-	-	3.3 V
3	B2_IO13*	107	User-I/O, DCI
4	B2_IO14*	108	User-I/O, DCI
5	B2_IO11	104	User-I/O
6	B2_IO12	105	User-I/O
7	B2_IO9	102	User-I/O
8	B2_IO10	103	User-I/O
9	B2_IO7	99	User-I/O
10	B2_IO8	100	User-I/O
11	B2_IO5	97	User-I/O
12	B2_IO6	98	User-I/O, VREF
13	B2_IO3	95	User-I/O
14	B2_IO4	96	User-I/O
15	B2_IO1	92	User-I/O, VREF
16	B2_IO2	93	User-I/O
17	-	-	GND
18	-	-	3.3 V

*Connected to push buttons.

Bank 3 Header

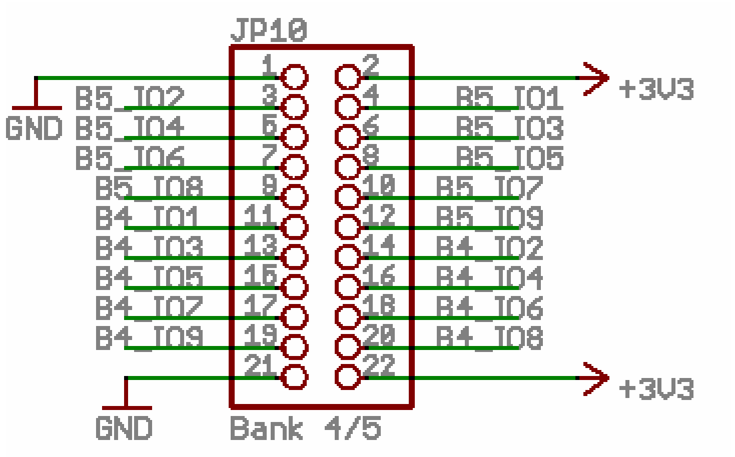


Bank 3 Header (Top:2x12, Bottom:2x7)			
Pin(Top)	**Name**	**FPGA Pin**	**Function**
1	-	-	GND
2	-	-	3.3 V
3	B3_IO14	89	User-I/O
4	B3_IO15	90	User-I/O, VREF
5	B3_IO12	86	User-I/O
6	B3_IO13	87	User-I/O

7	B3_IO10	84	User-I/O, VREF
8	B3_IO11	85	User-I/O
9	B3_IO8	82	User-I/O
10	B3_IO9	83	User-I/O
11	B3_IO6*	79	User-I/O
12	B3_IO7*	80	User-I/O
13	B3_IO4*	77	User-I/O
14	B3_IO5*	78	User-I/O
15	B3_IO2*	74	User-I/O, DCI
16	B3_IO3*	76	User-I/O
17	CONFIG_INIT_B/ IO*	58	User-I/O, DUAL
18	B3_IO1	73	User-I/O, DCI
19	CONFIG_PROG_ B*	143	CONFIG
20	CONFIG_CCLK*	72	CONFIG
21	CONFIG_DIN/IO*	65	User-I/O, DUAL
22	CONFIG_DONE*	71	CONFIG
23	-*	-	GND
24	-*	-	3.3 V

*These pins make up the interface to the control module.

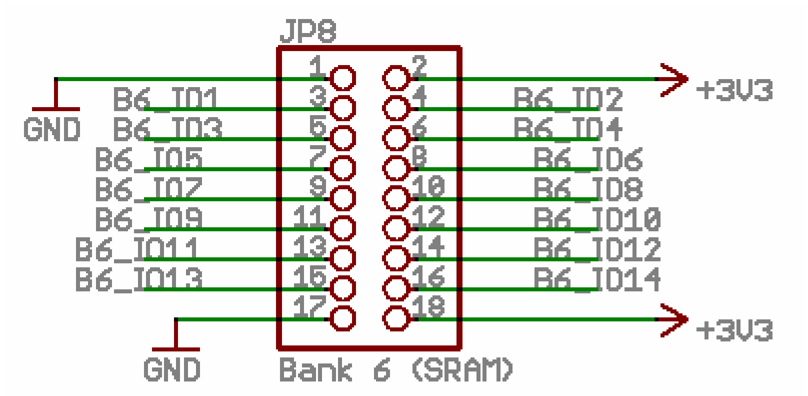
Bank 4/5 Header



Bank 4/5 Header (2x11)			
Pin	**Name**	**FPGA Pin**	**Function**
1	-	-	GND
2	-	-	3.3 V
3	B5_IO2	41	User-I/O, DUAL

4	B5_IO1	40	User-I/O, DUAL
5	B5_IO4	46	User-I/O, DUAL
6	B5_IO3	44	User-I/O, VREF
7	B5_IO6	50	User-I/O, DUAL
8	B5_IO5	47	User-I/O, DUAL
9	B5_IO8	52	User-I/O, GCLK
10	B5_IO7	51	User-I/O, DUAL
11	B4_IO1	55	User-I/O, GCLK
12	B5_IO9	53	User-I/O, GCLK
13	B4_IO3	57	User-I/O, DUAL
14	B4_IO2	56	User-I/O, GCLK
15	B4_IO5	60	User-I/O, DUAL
16	B4_IO4	59	User-I/O, DUAL
17	B4_IO7	68	User-I/O, DCI
18	B4_IO6	63	User-I/O, DUAL
19	B4_IO9	70	User-I/O, VREF
20	B4_IO8	69	User-I/O, DCI
21	-	-	GND
22	-	-	3.3 V

Bank 6 Header

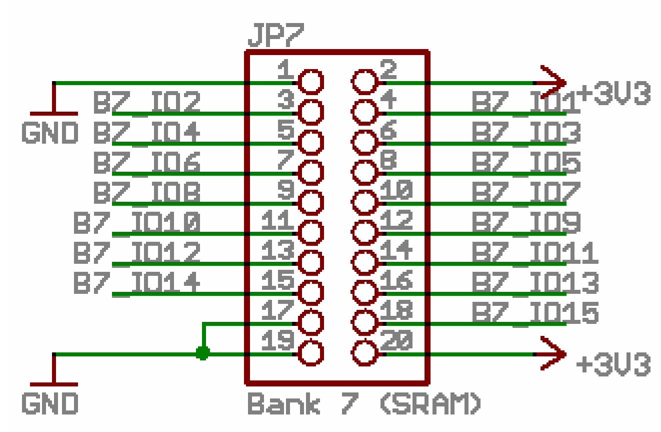


Bank 6 Header (2x9)			
Pin	**Name**	**FPGA Pin**	**Function**
1	-	-	GND

2	-	-	3.3 V
3	B6_IO1*	20	User-I/O, VREF
4	B6_IO2*	21	User-I/O
5	B6_IO3*	23	User-I/O
6	B6_IO4*	24	User-I/O, VREF
7	B6_IO5*	25	User-I/O
8	B6_IO6*	26	User-I/O
9	B6_IO7*	27	User-I/O
10	B6_IO8*	28	User-I/O
11	B6_IO9*	30	User-I/O
12	B6_IO10*	31	User-I/O
13	B6_IO11*	32	User-I/O
14	B6_IO12*	33	User-I/O
15	B6_IO13*	35	User-I/O, DCI
16	B6_IO14*	36	User-I/O, DCI
17	-	-	GND
18	-	-	3.3 V

*Used by SRAM when a jumper is placed on the header EN SRAM.

Bank 7 Header

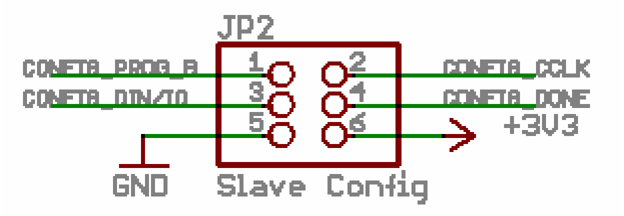


Bank 7 Header (2x10)			
Pin	**Name**	**FPGA Pin**	**Function**
1	-	-	GND
2	-	-	3.3 V
3	B7_IO2*	2	DCI
4	B7_IO1*	1	DCI
5	B7_IO4*	5	I/O

6	B7_IO3*	4	VREF
7	B7_IO6*	7	I/O
8	B7_IO5*	6	I/O
9	B7_IO8*	10	I/O
10	B7_IO7*	8	I/O
11	B7_IO10*	12	I/O
12	B7_IO9*	11	I/O
13	B7_IO12*	14	I/O
14	B7_IO11*	13	I/O
15	B7_IO14*	17	I/O
16	B7_IO13*	15	I/O
17	-	-	GND
18	B7_IO15*	18	VREF
19	-	-	GND
20	-	-	3.3 V

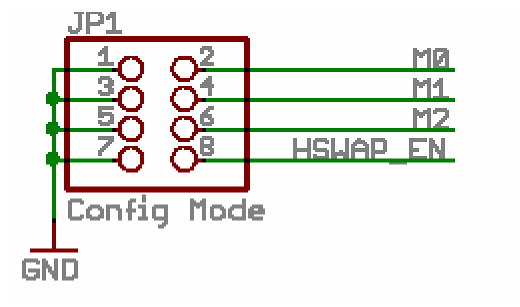
*Used by SRAM when a jumper is placed on the header EN SRAM.

Slave Config



Slave Config (2x3)			
Pin	**Name**	**FPGA Pin**	**Function**
1	CONFIG_PROG_B	143	CONFIG
2	CONFIG_CCLK	72	CONFIG
3	CONFIG_DIN/IO	65	DUAL
4	CONFIG_DONE	71	CONFIG
5	-	-	GND
6	-	-	3.3 V

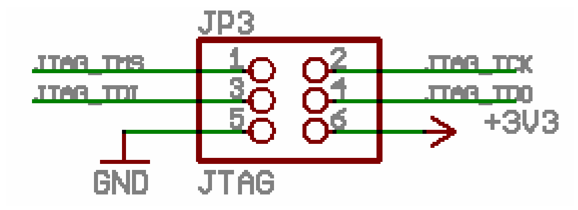
Config Mode



Config Mode* (2x4)			
Pin	**Name**	**FPGA Pin**	**Function**
1	-	-	GND
2	M0	38	CONFIG
3	-	-	GND
4	M1	37	CONFIG
5	-	-	GND
6	M2	39	CONFIG
7	-	-	GND
8	HSWAP_EN	142	CONFIG

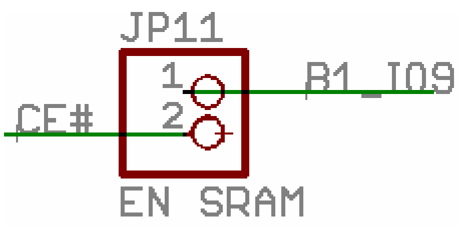
*For the different configuration modes, see section *6.2.3* ***Fehler! Verweisquelle konnte nicht gefunden werden.***.

JTAG



JTAG (2x3)			
Pin	**Name**	**FPGA Pin**	**Function**
1	JTAG_TMS	111	JTAG
2	JTAG_TCK	110	JTAG
3	JTAG_TDI	144	JTAG
4	JTAG_TDO	109	JTAG
5	-	-	GND
6	-	-	3.3 V

EN SRAM

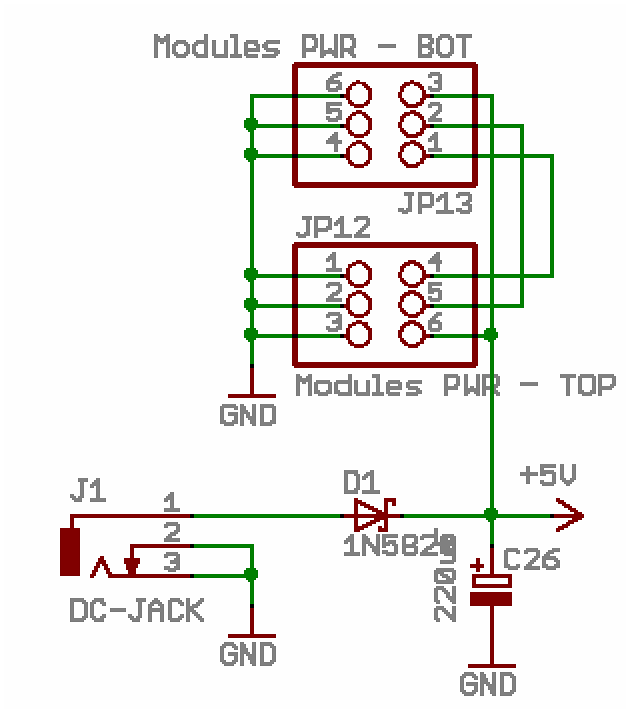


EN SRAM (2x1)*			
Pin	**Name**	**FPGA Pin**	**Function**
1	B1_IO9**	125	User-I/O, GCLK
2	CE#	-	SRAM enable (low-active)

*If the SRAM is needed, place a jumper on this header.

**Shared with the header Bank 0/1

Modules PWR



Modules PWR (2x3)	
Pin(Top)	**Function(s)**
1	GND
2	GND
3	GND

4	+V_{in}
5	3.3 V
6	5 V

*Power pins from the Power & Motion module; if the module is used in stand-alone, apply the power through the DC-Jack.

Appendix C - CD-ROM

The CD-ROM attached with this thesis contains data files related to this project.

Directory structure of the CD-ROM

- **Thesis** – this thesis in Microsoft Word and PDF format
- **Video Clips**
  - MAMoRo - Line Follower.AVI
  - MAMoRo - Go Ahead.AVI
- **Pics –** some pictures of MAMoRo
- **Code**
  - uC Bootloader – source files for the bootloader in *8.1.3 Initial Programming of the Control Module*
  - initprog.bat – this batch file executes the steps needed for the initial programming of the Control Module
  - MAMoRo Bootloader – source files for the MAMoRo bootloader in *8.3 Programming the Overall System*
  - Line Follower – source files for the application in *9.1 The Line Follower*
  - Go Ahead – source files for the application in *9.2 Go Ahead*
  - Misc – contains miscellaneous source files
- **Software**
  - FT232R Drivers – the drivers for the UART-USB converter IC
  - WinAVR – Open Source Development Tools for Atmel AVR Microcontrollers
- **Schematics & Board Layouts**

References

A

[ATM06a] Atmel™: *ATmega128(L) datasheet*. March 2006.

[ATM06b] Atmel™: *AVR109: Self Programming*. Application note, April 2006.

[ATM06c] Atmel™: *AT91 ARM Thumb-based Microcontroller*s. Datasheet, November 2006.

[ATM05a] Atmel™: *AVR042: AVR Hardware Design Considerations*. Application note, September 2005.

[ATM05b] Atmel™: AT45DB041B DataFlash™. Datasheet, May 2005.

[ATM05c] Atmel™: *ATmega48/88/168*. Datasheet, February 2005.

[ATM04] Atmel™: *AVR040: EMC Design Considerations*. Application note,, January 2004.

[ATM00] Atmel™: *AVR910: In-System Programming*. Application note,, November 2000.

B

[BHL04] I. Boersch, J. Heinsohn, H. Loose, and K.-U. Mrkor: *RCUBE - a Multipurpose Platform for Intelligent Autonomous Systems*. In Proceedings of the IEEE International Conference on Mechatronics, pp. 182–187, June 2004.

F

[FO06] C. Frischknecht, and T. Other: *LEGO Mindstorms NXT*. Semester thesis, Swiss federal institute of technology, July 2006.

[FAL06] FAULHABER Group™: *Encoders - Magnetic Encoders*. Datasheet, February 2006.

[FTD05] FTDI™: *FT232R USB UART I.C.*. Datasheet, ver. 1.04, 2005.

G

[GHW03] U. Gerecke, P. Hohmann, and B. Wagner: *Educational robotics in a systems design masters program*. In The 3rd IEEE International Conference on Advanced Learning Technologies, pp. 175–179, July 200.

I

[IFR06] International Federation of Robotics: *Executive Summary of the Study World Robotics 2006*. 2006.

[ISS05] ISSI™: *IS61LV25616AL - 256k x 16 High Speed Asynchronous CMOS Static RAM with 3.3v Supply*. Datasheet, March 2005.

K

[KTm07a] K-Team: *Khepera-III user guide*. Mai 2007.

[KTm07b] *K-Team homepage*. Online at: http://www.k-team.com [2007/5/26]

L

[LAR06] *LART homepage*. Delft University of Technology. Online at: http://www.lartmaker.nl [27/12/2006].

[LEG06a] *LEGO™ Mindstorms homepage*. Online at: http://mindstorms.lego.com [25.12.2006].

[LEG06b] LEGO™: *LEGO MINDSTORMS NXT Hardware Developer Kit*. Manual, ver. 1.00, 2006.

M

[MAX05] Maxim™: *4-Pin µP Voltage Monitors with Manual Reset Input*. Datasheet, December 2005.

[MAC04] D. Mattsson, M. Christensson: *Evaluation of synthesizable CPU cores*. Master thesis, Chalmers University of Technology (Sweden), 2004.

N

[NRB06] G. Novak, C. Roesener, M. Bader, T. Deutsch, S. Jakubek, S. Krywult, and M. Seyr: *The Tinyphoon's Control Concept*. In IEEE International Conference on Mechatronics, pp. 625–630, July 2006.

[NM05] G. Novak and S. Mahlknecht: *Tinyphoon - A Tiny Autonomous Mobile Robot*. In Proceedings of the IEEE International Symposium on Industrial Electronics, Volume 4, pp. 1533–1538, June 2005.

[NAT05] National Semiconductor™: *LM1085 3A Low Dropout Positive Regulators*. Datasheet, June 2005.

[NAT02] National Semiconductor™: *LM2596 SIMPLE SWITCHER™ Power Converter 150 kHz 3A Step-Down Voltage Regulator*. Datasheet, May 2002.

O

[ONS06] On Semiconductor™: *MC34161, MC33161, NCV33161*. Datasheet, June, 2006.

R

[RCU06] *RCUBE homepage*. University of applied sciences in Brandenburg. Online at: http://ots.fh-brandenburg.de/rcube [27.12.2006].

[RAS01] M. Rashid: *Power Electronics Handbook*. Academic Press, 2001. ISBN: 0-12-581650-2.

S

[STM00] STMicroelectronics™: *L298 - Dual Full-Bridge Driver*. Datasheet, January, 2000.

[STM95] STMicroelectronics™: *Applications of Monolithic Bridge Drivers*. Application note, 1995.

T

[TiP06] *Tinyphoon Homepage*. Online at: http://www.tinyphoon.com [6/6/2006].

W

[WIN07] *WinAVR™ Homepage*. Online at: http://winavr.sourceforge.net/ [10/5/2007].

X

[XIL07a] Xilinx™: *Spartan-3 Generation Configuration User Guide*. February 2007.

[XIL07b] *Xilinx™ Homepage*. Online at: http://www.xilinx.com/ [11/5/2007}.

[XIL07c] Xilinx™: *MicroBlaze™ Performance*. Online at: http://www.xilinx.com/ipcenter/processor_central/microblaze/performance.htm [25/4/2007].

[XIL06a] Xilinx™: *Spartan™-3 FPGA Family*. Datasheet, April 2006.

[XIL06b] Xilinx™: *The 3.3 V Configuration of Spartan-4 FPGAs*. Application note, April 2006.

[XIL05] Xilinx™: *Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors*. Application note, February 2005.

[XIL02] Xilinx™: *Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode*. Application note, November 2002.

[XIL00] Xilinx™. *JTAG Programmer Guide*. Application note, 2000.

Z

[ZBZ06] H. Zhang, T. Baier, J.Zhang, W. Wang, R. Liu, D. Li, G. Zong: *Building and Understanding Robotics—a Practical Course for Different Levels Education*. In Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics, December 2006.