Beijing University of Posts and Telecommunications

Undergraduate Science and Technology Innovation Camp

Final project

CONTROL BOARD FOR CNC SYSTEMS BASED IN LINUX

Degree in Electronic and Communications Engineering

Rodolfo Boris Oporto Quisbert

April 2014

CONTROL BOARD FOR CNC SYSTEMS BASED IN LINUX

AUTHOR: Rodolfo Boris Oporto Quisbert
SUPERVISOR: Guo Li

Undergraduate Science and Technology Innovation Camp
Beijing University of Posts and Telecommunication
April 2014

Abstract

Abstract

The objective of this work is to implement a control card for CNC systems. This board will use different Open Source tools and will be entirely integrated into LinuxCNC running on a Raspberry-Pi.

After introducing the technologies involved, this work will focus on the different parts that compose the card.

Subsequently, a series of tests of each of the parts that compose the card designed will be made.

Finally, this card will be integrated into a real system consisting of a RepRap 3D printer, which has been adapted to test the functionality of the card designed.

Key words

LinuxCNC, Open Source, CNC, RepRap, Rapsberry-Pi.

Acknowledgements

I would like to thanks my supervisor, Professor Li Guo from the BUPT, for providing me the tools, support and equipment needed to develop this project.

Also, I would like to thanks the wonderful people from the Undergraduate Science and Technology Innovation Camp of the BUPT, to help me to love the exceptional Chinese culture.

Finally, I would like to thanks my family, especially my mother, for their constant trust in me, even in the most uncertain moments of my life.

Gracias por todo mamá, sin tí, nada hubiese sido posible.

Contents

List of Figures

List of Tables

List of acronyms

- **OS**: Open source
- **RPi**: Raspberry Pi
- **3D**: 3-Dimensions
- **RT**: Real Time
- **CNC**: Computer Numerical Control
- **ADC**: Analog Digital Converter
- **OSHw**: Open Source Hardware
- **CAD**: Computer Aid Design
- **SOA**: Safe Operation Area
- **RTOS**: Real Time Operative Systems
- **PWM**: Pulse Width Modulation
- **GCC**: GNU Compiler Collection
- **DIY**: Do It Yourself
- **SoC: System on Chip**
- **USB**: Universal Serial Bus
- **EMC**: Enhanced Machine Control
- **HAL: Hardware Abstraction Layer**
- **FFF**: Fused Filament Fabrication
- **SPI**: Serial Peripheral Interface
- **MOSI**: Master Output Slave Input
- **MISO**: Master Input Slave Output
- **CLK**: Clock
- **SDI**: System Data Input
- **SDO**: System Data Output
- **CSV**: Comma Separated Values
- **DMA**: Direct Memory Access

1 Introduction

1.1 Project motivation

The market has different solutions for Computer Numerical Control (CNC) control. Most of these controllers are intellectual property of theirs makers, which in general means, to depend on a little number of companies in the lifetime of the system. This circumstance can increases the price of the final system, because of the economic dependence due the prices fixed by maker, and also have to add the maintenance costs that probably will increase the final price even more.

In the last years, this systems have attracted the attention of many groups, interested especially in 3D printers.

The problematic to control 3D printers are the same than in CNC machines, because the variables to control are the same:

- Both required to produce movement in three or more axis.
- Both required to acquire analog variables such speed, acceleration, temperature.
- Both required the implementation of control algorithms, in order to control the system response.
- Both uses G-Code as standard to describe the movements of the axis.

For this reason, during the development of this project, there is no distinction between CNC machines and 3D printers, at least from the engineering point of view.

1.2 Objectives and Approach

The aim of this work was to create a fully functional control-board for controlling CNC systems. To do this, was used several Open Source (OS) projects such as Raspberry-Pi, Raspbian, LinuxCNC, Xenomai, RepRap and GNU.

All these parts solve some problematic related with CNC machines control. These parts working together, make the complete CNC system:

Figure 1.1: Industrial CNC machine.

- Raspberry-Pi is works as the main part of the system. All the software written by this work is running in Raspberry-Pi, and no component works independently of Raspberry-Pi.
- Raspbian is the most used operative system for Raspberry-Pi. It provides the same functionality than Debian Linux-Distribution.
- LinuxCNC is a collection of OS software to control CNC machines. It control everything related with the control system (e.g., speeds, accelerations, temperatures), implements the control algorithms, provide communication solutions and provide the user interface.
- Xenomai provides the support to work with Real-Time (RT) applications under Linux. Linux as a multi-programmed OS, that does not support real-time by default. For this reason, was needed to patch the Linux Kernel to get this functionality.
- RepRap is the most popular OS 3D printers Project. In this work, is used RepRap to verify the whole system.
- In this work was used GNU compiler (GCC) to compile all the code required.

The Figure 1.2 shows a diagram that represent the complete system, used to verify the design. The parts that compose the full system will be in deep in further chapters of this report.

In general, the functionality of a conventional CNC machine was achieved. The implementation process and the testing of each part will be presented in additional chapters.

1.3 Structure of this report

This report is separated in different chapters:

- **Chapter 1 Introduction:** In this chapter will be introduced the problematic to solve, will be fixed the objectives and the organization of the project.
- **Chapter 2 Background:** In this chapter is all the relevant information about the technologies used during the development of this work. Then will be explained the different parts of the system and the peculiarities of each one.
- **Chapter 3 Design and implementation:** In this chapter will be explained the design process, the implementation of the control-board and the development of the software required.

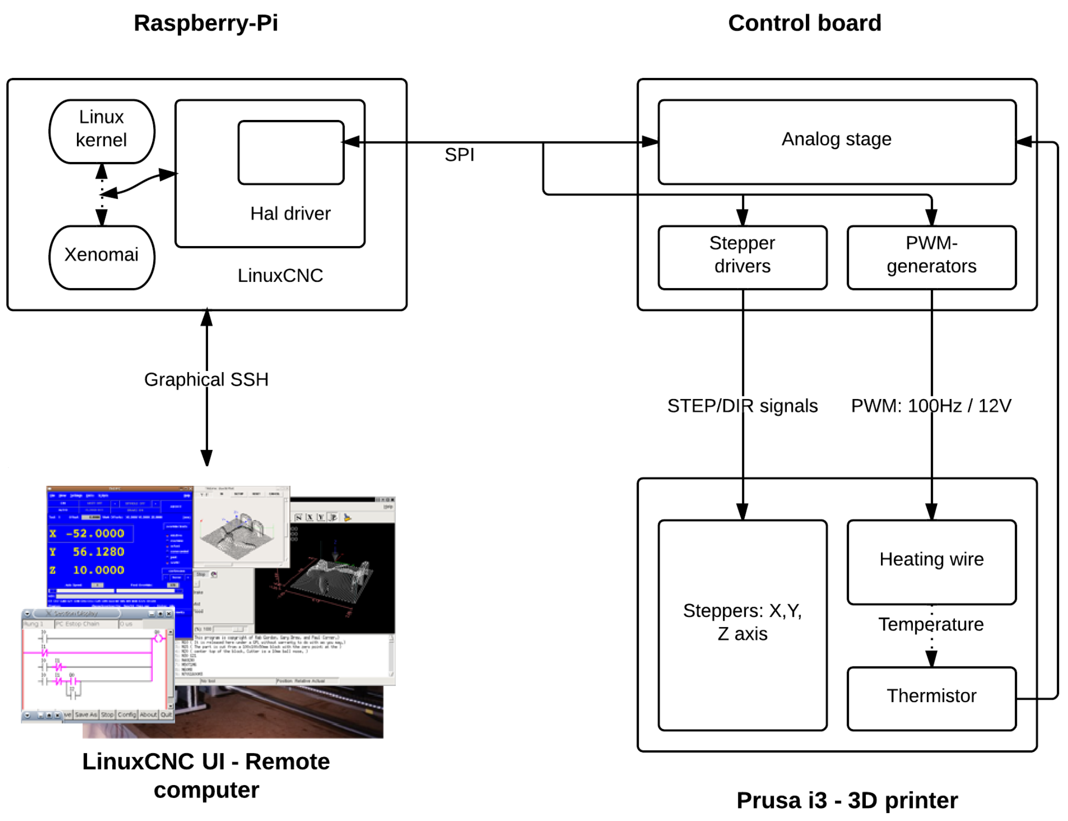


Figure 1.2: Full system overview

- **Chapter 4 Results and discussion:** In this chapter will be explained all the tests, designed to verify each part of the complete system and its results. Also, will be tested the full system, to validate the functionality achieved.

- **Chapter 5 Conclusions and future work:** In this chapter will be discussed the conclusions and will be introduced ideas for future work .

  Another auxiliary documentation is attached at the end of this report, such as schematics and user manuals.

1.4 Methodology and working plan

The work was distributed in many milestones in order to achieve the purposes.

1. **Research of the factors that compound the system:** As was already mentioned, Open Source Hardware (OSHw) projects, usually depends on large communities, this situation sometimes made the information hard to find because of the many sources. That is the reason because is needed a big effort, to identify the reliable sources of information from those that are not.

2. **Selection of the hardware that composed the control board:** Having identifying the required functionality of the board, the components that make the full system will be

selected. In further chapters is described the reasons to select each part and the function in the system.

3. **Design and making of the control board:** Once was selected the main components, the entire system will be designed. For this purpose, was used Eagle CAD software. Then was prototyped, assembled and applied manual testing with a multimeter, and laboratory standard equipment.

4. **Experimentation and tests:** In this stage, was written software libraries to control the board. Here were the first experiences with RT under Linux.

5. **Development of HAL driver:** To integrate the board with LinuxCNC was needed to write an HAL driver. This driver is the interface between LinuxCNC and the hardware.

6. **Integration with LinuxCNC:** Once is finished the driver, was needed to write configuration scripts to use the board.

7. **Final test:** The complete system was tested and validated in a real CNC environment.

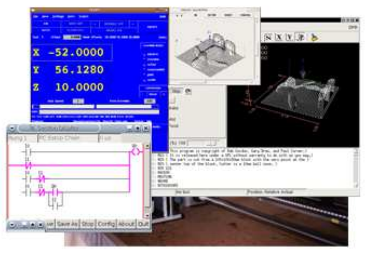


Figure 1.3: LinuxCNC platform user interface

2
Background

In this chapter will be introduced all the technologies used during the development of this work, the interactions among them and the special features used.

2.1 Open Source Projects

2.1.1 Brief Introduction to Open Source

The rise of the internet has brought with it a new paradigm of project development. During nineties, the quick development of GNU-Linux based operative systems has changed the way that people understand the software development. Internet is the house of hundred of new OS projects in which are involved many people.

The most remarkable example of OS project is GNU-Linux. The GNU-Linux project started in the early 80's, it spreads rapidly among technology-lover's communities. The universal access to the internet has just increased the number of people whom every day use, even without knowing, GNU-Linux based-systems.

However, this phenomenon is not only limited to the software. Even before GNU-Linux and internet, in the United States, many people have been sharing information on what might be called primitive open source systems. *Do it yourself* (DIY), have been very popular among a huge community that shared information in hobbyist-clubs, by regular mail or magazines. Information related with various disciplines, such as electronics, mechanics and carpentry.

With the spread of internet, those groups had changed the way to share information, and started communities following the example of GNU-Linux, to develop their interests. Is in this context where OSHw raised. Names such as RepRap, Raspberry-Pi, and Arduino are more popular among people with development concerns.

2.1.2 Common characteristics of Open Source

There are many definitions of Open Source, because of the variety of these kinds of projects. It is a matter of discussion among the Open Source community, but in general, these definitions share the following features:

- Any person can access the information, independently of its purpose. It includes schematics, software produced and documentation generated.
- Any person can study the information released to understand in deep the functionality.
- Any person can redistribute copies of the information released in any support.

  Any person can modify partially, or completely, the information published. These modifications can be released to the public.

The lack of restrictions to access the information released, also have interesting effects:

- Reduce the development time. The developer does not have to waste time redoing parts already implemented.
- The community can support the developer during all the stages of the project. Sharing information and solving difficulties, during the development. Eventually, it optimized the final result adding improvements.
- Reduce the final cost, because in general, do not require the payment of any royalty.

2.2 Raspberry Pi

2.2.1 Introduction and brief description

Raspberry Pi is a low cost fully-functional computer, based in BCM2835 SoC from Broadcom. This board was developed to be used as part of an educational program for high schools, promoted by the Raspberry Pi Foundation, in the United Kingdom.

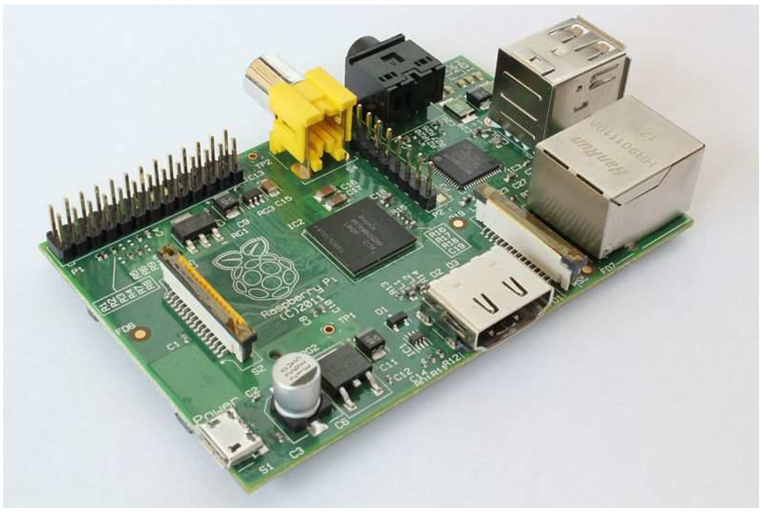


Figure 2.1: Raspberry-Pi Model B board

This board have been widely used by developers around the world and have several spin-up hardware and software projects.

The hardware characteristics are summarized in Table 2.1, according to the Raspberry Foundation Website(3).

SOc	Broadcom BCM2835.
CPU	700 MHz ARM1176JZF-S core (in SOc).
GPU	Broadcom VideoCore IV @ 250 MHz (in SOc).
SDRAM	512 MB (in SOc).
USB Ports	2 (USB hub).
Video Input	CSI connector.
Video Outputs	HDMI, RCA-Composite, DSI.
Audio Outputs	3.5mm jack, I2S audio and HDMI.
Storage	SD card slot.
Network	10/100 Mbit/s Ethernet.
Low-level peripherals	8 X GPIO, UART, I2C, SPI-2 chip selects, I2S audio.
Power source	5 V via MicroUSB connector.
Size	85.60 mm × 56 mm.
Weight	45 g.

Table 2.1: Raspberry-Pi Specifications

2.2.2 Raspbian

To be used, Raspberry-Pi require an Operative System running from the SD card, the majority of the Raspberry-Pi users use the Linux based Operative System Raspbian. Raspbian is based in Debian Wheezy, optimized to be used in Raspberry-Pi hardware.

Raspbian allows the Raspberry-Pi users, to have the fully-potential of Debian, running in a credit card size computer with the same performance than the older Pentium III computer.

In (4) is all the information related with Raspbian including download links.

2.2.3 Patching Linux Kernel

To achieved the functionality required by LinuxCNC, in this work, was used the 3.8.13 kernel version optimized by Raspberry-Pi, available from the Raspberry-Pi Project Git Repository(5).

This kernel was patched with the official Xenomai patch for Raspberry-Pi, available in the Xenomai Project Git Repository(6), in order to obtain the real-time execution required.

The compilations of all the software developed during this work was done using the GCC compiler, optimized by Raspberry-Pi, available from the Raspberry-Pi Project Git Repository(5).

Is not the purpose of this work to get in deep about the patching and compilation procedures. In(7) can be consulted a complete documentation about the procedures followed.

2.2.4 Why Raspberry-Pi?

The reason because was selected Raspberry-Pi as a central board for this design, is because, among all the Open Source ARM-Based boards, Raspberry-Pi has the following characteristics:

- **Raspberry-Pi community is huge:** In practice it means, that the developer has much documentation and examples available to consult when necessary.
- **Raspberry-Pi is low cost:** Raspberry-Pi have a very low cost compared with other computers. To have a low price is one of the main objectives of the Raspberry-Pi project.
- **Raspberry-Pi works with LinuxCNC and Xenomai:** There are experiences running LinuxCNC and Xenomai. It is necessary because LinuxCNC and Xenomai are complex tools, which means that any little modification would delay the finalization of the project for an uncertain time.

All these characteristic make Raspberry-Pi the perfect option for this work.

2.3 Real time under Linux systems

2.3.1 Linux and the Real-time problematic

Linux is a multi-programmed operative system, which means that is designed to perform multiple programs at the same time. These programs will perform in a time-shared environment, it means that, for users, these programs have simultaneous execution.

The truth is that Linux executes one process each time (when the processor has a simple architecture with a single processor), so fast, (at least from the user point of view), that the user can feel that the processes are executing simultaneously.

It works perfectly for general applications where the difference between 100μs or 200μ is not relevant. However in industries where the time of execution is critical, such as robotics and telecommunications, is required what is called real-time execution.

From the beginning of the Linux development, some groups have been working in providing Linux real-time execution solutions. There are well-know RT options summarized in following subsections.

Scheduling

The mechanism to decide which process use the system resources is known as scheduling. Scheduling is the method that the operative system uses to give access to systems resources, such as processor time, access to peripherals and communications, to any thread or process. In general, this mechanism is complicated and exist several algorithms that depend on the operative system or even the kernel version.

It is not the aim of this work to get in deep about the scheduling problematic, but it is necessary at least to know about this process, because most of the solutions to provide real-time in Linux modify, in some way, the scheduling.

2.3.2 PREEMPT_RT patch

Preemption

Before to explain how the PREEMPT_RT patch works. It is necessary to have a brief explanation about preemption.

Preemption is the interruption of a computer process without its cooperation, in order to perform another task. In general the Linux kernel allows the preemption just under certain circumstances:

- When the CPU is running user-mode code.
- When kernel code returns from a system call or an interrupt back to user space.
- When kernel code blocks a mutex, or explicitly yields control to another process.

It means that a standard user-mode code can not be sure about the time of execution of a particular task implemented, even more, any kernel code can take ownership of the CPU when is required.

The consequence is that the user can not be sure about the time to execute any program.

PREEMPT_RT patch summarized

The PREMPT_RT patch transforms Linux in a fully preemptible kernel. It means, that a user-mode code can take control of the system resources under certain conditions.

In the official wiki of the group(8) is explained the functionality, as follows:

- Making in-kernel locking-primitives (using Spinlocks, which is a mechanism that repeatedly try to take ownership of a mutex until it is done.) preemptible though reimplementation with rtmutexes.
- Critical sections protected by i.e. spinlock_t and rwlock_t are now preemptible. The creation of non-preemptible sections (in kernel) is still possible with raw _spinlock_t (same APIs like spinlock_t).
- Implementing priority inheritance for in-kernel mutexes, spinlocks and rw_semaphores.
- Converting interrupt handlers into preemptible kernel threads: The RT-Preempt patch treats soft interrupt handlers in kernel thread context, which is represented by a task_struct like a common userspace process. However it is also possible to register an IRQ in kernel context.
- Converting the old Linux timer API into separate infrastructures for high resolution kernel timers plus one for timeouts, leading to userspace POSIX timers with high resolution.

All these changes are applied to the kernel, using the PREEMP_RT patch, and recompiling the kernel.

2.3.3 Xenomai

Acordign to(9) , Xenomai, is a real-time development framework cooperating with the Linux kernel, in order to provide a pervasive, interface-agnostic, hard real-time support to user-space applications, seamlessly integrated into the GNU/Linux environment.

It means that Xenomai provides a real-time development framework. Xenomai provides two spaces as development frameworks: user-space and kernel-space, both can be used when required, and, in general, share most of the Xenomai API.

For more information about Xenomai API and functionality, consult the official web-site (9).

2.3.4 Why Xenomai?

An important part of the decision making process is to chose among all the alternatives in order to provide real-time execution. The challenge is to achieve real-time execution without adding complexity in the final design of the software.

In (10) , the authors define criteria to choose between all the possibles options.

In this work is used Xenomai, because the strict requirements of the real-time executions and because during the experimentations, Xenomai achieve better performance in latency test.

2.4 LinuxCNC

2.4.1 Introduction and Architecture Overview

LinuxCNC, former know as EMC(11), is a group of software made for Computer Numerical Control machines (CNC). This software allows to control different CNC machines such as mills, plasma cutters, laser cutters and can be adapted easily to 3D printers.

Figure 2.2, shows the LinuxCNC architecture overview.

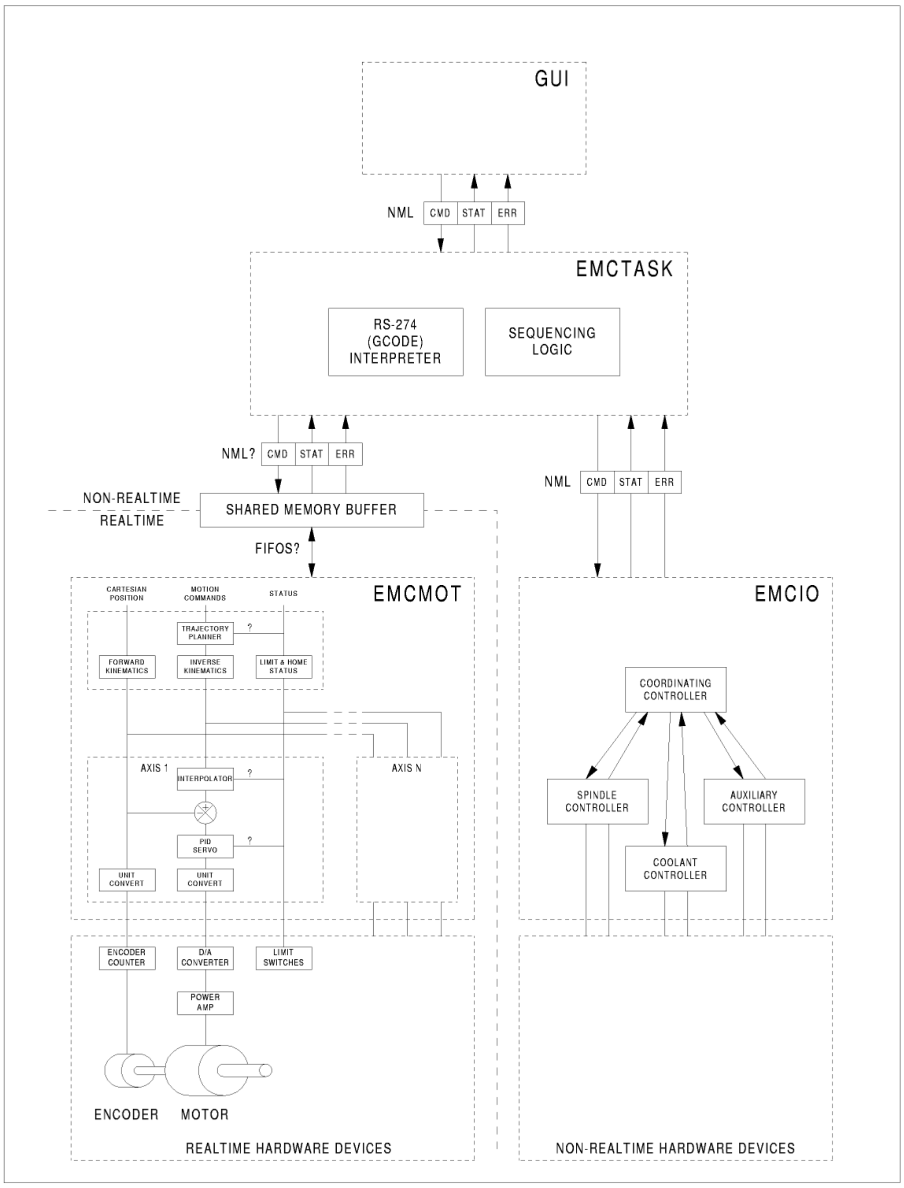


Figure 2.2: LinuxCNC architecture overview

It is not the aim of this work to get in deep about LinuxCNC functionality, but is necessary at least to keep in mind the principal blocks of its architecture and its functionality.

According to the LinuxCNC official web-site(11) the main blocks of its architecture are as follows:

- **Motion controller (EMCMOT):** The motion controller receives commands from user space modules via a shared memory buffer, and executes those commands in real-time (Xenomai). The status of the controller is made available to the user space modules through the same shared memory area. The motion controller interacts with the motors and other hardware using the HAL (Hardware Abstraction Layer).

- **Discrete IO controller (EMCIO):** The discrete I/O controller provide an interface with the physical world providing a common interface, this is the reason because is easy to integrate hardware in LinuxCNC.
- **Task executor (EMCTASK):** It receives the commands from the GUI, monitors the status of subordinate modules (EMCMOT and EMCIO), analyzes the commands based on the current situations, and calls functions either within EMCTask itself or dispatches commands down to its subordinate modules(EMCMOT or EMCIO).
- **User Interfaces :** LinuxCNC have different user interfaces packages, this work uses Axis, which is the most used among LinuxCNC users. LinuxCNC user can manipulate these interfaces to adapt them to any purpose.

2.4.2 LinuxCNC and RepRap differences

RepRap 3D printers and CNC machines are intimately related because both interpret G-Code in order to create motion and other actions. The motion usually is made by some mechanical device such as steppers-motors.

Despite the similarities, there are come differences related with the complexity of the parts that composed the whole system.

The standard controllers for RepRap machines usually works, as follows:

- The main computer sends G-Code to some control-board.
- The control-board receive and interpret the G-Code and send motion signals to the steppers through the energy-drivers.
- At the same time, the control-board acquire any analog's signals such as temperature, acceleration and speed.
- Control-board uses this information to create a response. In the case of heaters, produce a PWM signal proportional to the desired temperature.

On the other hand, the LinuxCNC machines, in general works as follows:

- The main computer interpret the G-Code and send motion signals to some control-board through parallel-port.
- The control-board receive these signals, that just require to provided enough energy, and produce the motion.
- If any analog signal measuring is required, this obtained by ADC.
- This information is processed by the main computer and generates a response.

The Figure 2.3 illustrate these differences.

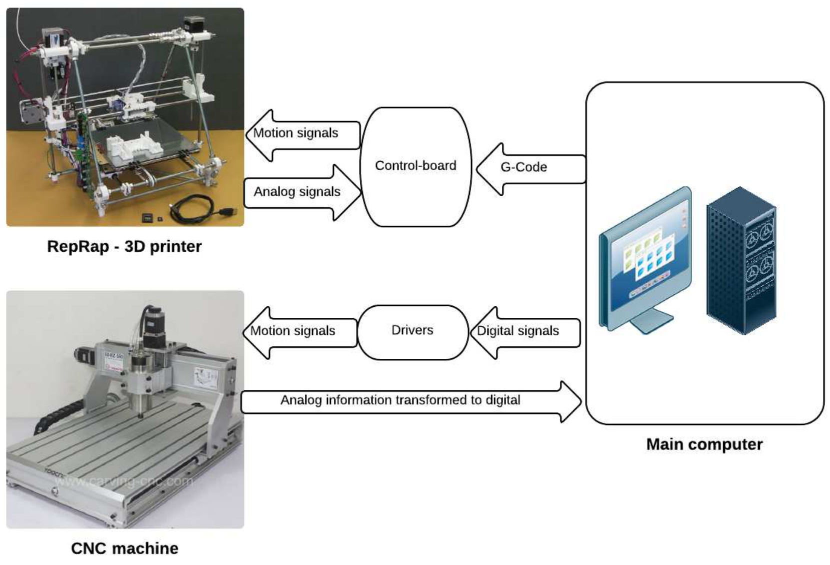


Figure 2.3: Differences between LinuxCNC and RepRap

2.4.3 Hardware Abstraction Layer Introduction

To describe the functionality required to control CNC machines, LinuxCNC use software components know as Hardware Abstraction Layer (HAL). Despite the name, maintained because in the beginnings of LinuxCNC this component represents some hardware (e.g. control boards, encoders), HAL components are not just made to describe hardware, it also describes blocks with some functionality.

So, the developer can abstract from the details inside these blocks and use it like an integrated circuit in a particular circuit.

LinuxCNC provides a complete library of HAL components, in real-time and user spaces. These components are blocks such as real-time threads, Pulse With Modulations (PWM) generators, Proportional Integral Derivative (PID) controllers, stepper-motor's controllers, speed encoders, communications ports.

To control a CNC machine is required the integration of several HAL components, in order to achieve the desired functionality.

Important HAL components

There is a huge collection of HAL components, most of them have complex functionality and the full information related with them can be seen in the corresponding manual page. But because of the importance in this work and in LinuxCNC is necessary tho describe briefly, the most important HAL components:

- **Threads:** This part create a hard real time thread which can be used to perform HAL functions at specific time. Threads are passed as parameters in the execution time of some function. The distribution installed in the Raspbian SD image, adapt the LinuxCNC software to execute threads using Xenomai.
- **Stepgen:** This component allow to control stepper-motors, generating all the signals

needed con control various kinds of steppers. In this work was used the simplest control, which es generating the STEP and DIR signals.

- **PID:** This component describe a PID control block. In this work, this component as the main component to control the temperature.

- **PWMgen:** This component generate a PWM signal. In conventional 3D printers, PWM is used to control heating elements such as power resistances, heating wire. Another important device to be controlled by PWM generators is servo-motors.

- **hal_rpirboq:** This is the HAL component created to control the board. In further chapters are described in deep the characteristics of this driver.

There are another HAL components to be used if necessary. To more information about the components described and other components, consult the manual pages of LinuxCNC components (12) or the HAL manual (13).

2.5 RepRap - 3D printer

2.5.1 Introduction

RepRap is an Open Source project which starts in 2005, in the University of Bath, in UK by Adrian Bowyer. The aim of RepRap project was to create a 3D printer cheaper than the commercial ones, and the most important, a self-replicant 3D printer.

When this project started, the newer version knows as Prusa i3, was released few months ago. The Figure 2.4 shows the machine appearance(14).

Since the beginning of the project, several versions have been released to be used by the general public. These versions have suffered numerous improvements and the performance of this machine has been increased considerably.

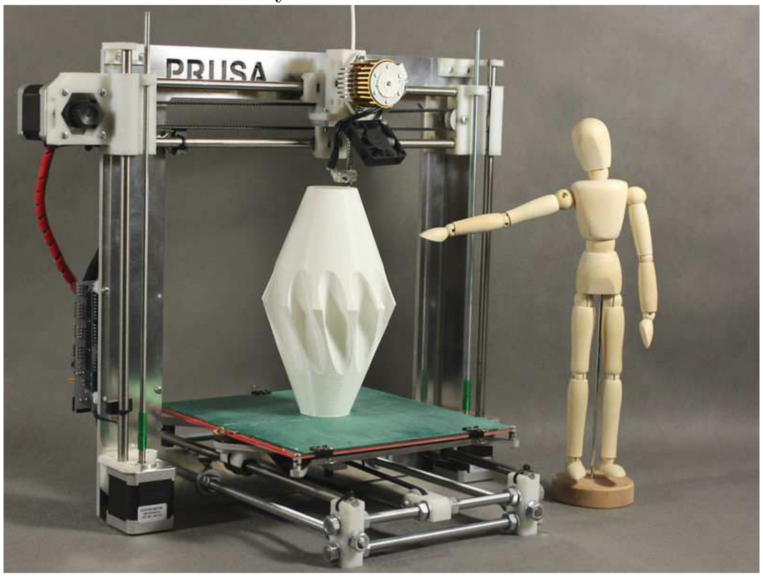


Figure 2.4: RepRap - Prusa Iteration 3

2.5.2 Problematic to be solved in 3D printing

The method used by RepRap to create 3D objects is called additive manufacturing. Specifically, the technic is called Fused Filament Fabrication (FFF). FFF is the method that create a 3D model by making subsequent layers, made by threads of melted material. The most used materials for this 3D printers are plastics such as ABS or PLA.

Some experimental versions work with other materials such as ceramics, melted sugar, chocolate. Moreover, even bigger 3D printers have been used to working in the production of houses.

In summary, the problematic to be solved by 3D printers are the following:

- Some mechanical components produce the movement of the axis. The wide used are stepper-motors, controlled by some control board that provides enough power.
- The material is deposed (in threads of melted material) to make a layer of solid material. To control this, is required some algorithm that controls variables such as temperature, amount of material. Again, this work is usually made by a control-board.
- The movements required (and other variables) are described in G-Code. The control-board interprets this G-code.

In this work, a Prusa i3 was used to validate the control system designed because the problematics to solve in CNC machines are the same than in 3D printers, primarily both require to control movements and other variables with PID controllers.

2.6 ADS8556

2.6.1 General description

The core of the Analog to Digital Conversion (ADC) system is the ADS8556 from Texas Instruments(15). This device has six channels, 16 bit's resolution, fully bipolar inputs, with serial or parallel communication.

The ADS8556 use an SAR as conversion method. For this reason, is required a sample and hold external circuit, to have a good quality in the conversion. The details of the design of sample and hold circuit are explained in Chapter 3.

The most important specifications of the ADS8556 are summarized in the Table 2.2, taken by the ADS8556 manual(15).

Parameter	**Minimum**	**Typical**	**Maximum**
Logic family		CMOS	
Resolution		16 Bits	
Analog Supply Voltage, AVDD to AGND	4.5V	5V	5.5V
Digital Interface Supply Voltage, BVDD to BGND	2.7V	3V	3.6V
Analog Input Positive Supply Voltage, HVDD to AGND	2 × Vref		16.5V
Analog Input Negative Supply Voltage, HVSS to AGND	-16.5V		2 × Vref
Reference Voltage VREF	0.5V	2.5V	3V
Power-supply rejection ratio PSRR		16 Bits	
Serial clock input frequency	0.1 MHz		36 MHz
High-level input voltage	0.7×BVDD		BVDD +0.3
Low-level input voltage	BGND-0.3		0.3×BVDD
Power dissipation (Maximum frequency)		251.7 mW	298.5mW

Table 2.2: ADS8556 main specifications

2.6.2 Communication

In this work was used the SPI bus from the Raspberry-Pi to communicate with the ADS8556 in serial mode. The communication is fixed to 15.625MHz (SPI-CLK) approximately, this frequency was selected, because of the good response during the tests.

The ADS8556 also required other control signals, apart from MOSI(SDI), MISO(SDO) and CLK(SCLK). The Table 2.3 summarized the signals required and its function.

SDI	Data-input serial signal.
SDO	Data-output serial signal.
SCLK	Serial communication input clock.
FS	Frame synchronization. The falling edge controls the transfer.
RESET	Reset of the ADS8556 high-level activated.
STBY	Standby signal of ADS8556. This signal disconnects the analog stage when low.
CONV_A	This signal start a conversion procedure with raising edge. This signal has to be high while converting process.
BUSY	This signal is high while converting process. When the communication ends turn low, it means that the conversion process have finished and Raspberry-Pi can read the outputs in SDO line.

Table 2.3: ADS8556 signals required for serial communication with Raspberry-Pi

2.6.3 Control Register

To control the device, the ADS8556 have a Control Register of 32 bits, which controls the operations inside the device. The Table 2.4 shows the values of these bits and its function.

The ADS8556 require the constant upgrade of the Control Register during the sample operation. The writing of the Control Register in serial-mode software-control is made through the MOSI line from the Raspberry-Pi to the SDI line in the ADS8556.

The Figure 2.5 shows the constant upgrading of control register. The line D2 is the SDI of the ADS8556, that constantly receive the code FF0103FF(hex). The other lines D0, D4 and D3, represents FS, FSCKL and SDO.

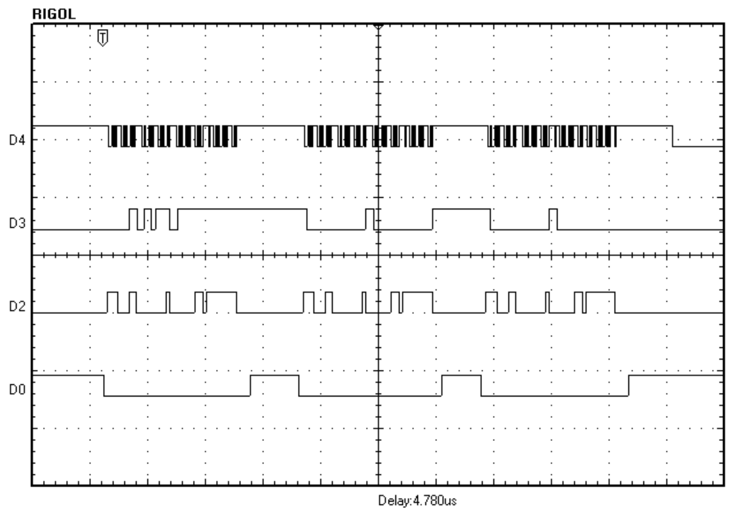


Figure 2.5: SPI communication between Raspberry-Pi and ADS8556

2.7 MCP23S17

2.7.1 Brief Introduction to MCP23S17

The Raspberry-Pi is a powerful computer system, but it has some limitations due to the lack of enough numbers of IO digital pins.

To control the stepper-motors drivers is required at least two digital pins for each motor. One for the STEP signal, and the other one for the DIRECTION signal. Also is required a pin to enable the drivers, that can be a common signal for all the drivers (as in this work). The PWM required one IO digital-pin for each output. Also, are required control signals for the ADS8556.

It means that the Raspberry-Pi board do not have enough digital signals (15) required to control: five steppers drivers, four PWM outputs and the control signals for ADS8556.

To solve this problem, in this work was used an IO expander MCP23S17(16) from Microchip.

2.7.2 Description

The MCP23S17 is a 16-bit IO expander with serial interface, I2C or SPI interface. MCP23S17 is a complex device and have several configuration registers divided into two banks.

These registers allows the user to configure these pins as:

- Use pins as inputs/outputs.
- Use pins in interruption mode.
- Select the polarity of input/output.
- Output/input as open-drain.
- Output/input weak pull-up with a 100kΩ internal resistor.

This 16-bits are divided into two ports(PORTA and PORTB) of 8-bits each one.

In this work the configuration of the PORTA and PORTB are as follows:

- PORTA and PORTB are configured as outputs.
- PORTA and PORTB have activated the weak pull-up 100kΩ enable.

The principal characteristics are summarized in the Table 2.5

2.7.3 Communication

To communicate with the Raspberry-Pi is used the SPI bus. This bus has some advantages, the most important is that in this work MCP23S17 share the same bus with ADS8556 because Raspberry-Pi have two chip-select pins. The second one, is that this bus can works faster than I2C or other bus. The speed is important in this work, because components like steppers and PWM controllers will reduce their working speed if MCP23S17 can not change the digital outputs with enough frequency.

In this work, the communication is fixed at 7.8125MHz because this is the highest frequency that MCP23S17 allows among the Raspberry-Pi frequency ranges.

More details about the communication with the MCP23S17 , see the code attached.

Bit	Name	Description	Value
C31	CH_C	0 = Channel pair C disabled for next conversion (default); 1 = Channel pair C enabled	1
C30	CH_B	0 = Channel pair B disabled for next conversion (default); 1 = Channel pair B enabled	1
C29	CH_B	0 = Channel pair A disabled for next conversion (default); 1 = Channel pair A enabled	1
C28	RANGE_C	0 = Input voltage range selection for channel pair C: 4×VREF (default); 1 = Input voltage range selection for channel pair C: 2×VREF	1
C27	RANGE_B	0 = Input voltage range selection for channel pair B: 4×VREF (default); 1 = Input voltage range selection for channel pair B: 2×VREF	1
C26	RANGE_A	0 = Input voltage range selection for channel pair A: 4×VREF (default); 1 = Input voltage range selection for channel pair A: 2×VREF	1
C25	REF_EN	0 = Internal reference source disabled (default); 1 = Internal reference source enabled	1
C24	REFBUF	0 = Internal reference buffers enabled (default); 1 = Internal reference buffers disabled	1
C23	SEQ	0 = Sequential convert start mode disabled (default); 1 = Sequential convert start mode enabled (bit 11 must be '1' in this case)	0
C22	A-NAP	0 = Normal operation (default); 1 = Auto-NAP feature enabled	0
C21	BUSY/INT	0 = BUSY/INT pin in normal mode (BUSY) (default);1 = BUSY/INT pin in interrupt mode (INT)	0
C20	BUSY L/H	0 = BUSY active high while INT active low (default); 1 = BUSY active low while INT active high	0
C19	Don't use		0
C18	BUSY L/H	0 = BUSY active high while INT active low (default); 1 = BUSY active low while INT active high	0
C17	READ_EN	0 = Normal operation (conversion results available on SDO_x) (default); 1 = Control register contents output on SDO_x with next access	0
C16	C23:0_EN	0 = Control register bits C[31:24] update only (serial mode only) (default); 1 = Entire control register update enabled (serial mode only)	1
C15	PD_C	0 = Normal operation (default); 1= Power-down for channel pair C enabled (bit 31 must be '0' in this case)	0
C14	PD_B	0 = Normal operation (default); 1 = Power-down for channel pair B enabled (bit 30 must be '0' in this case)	0
C13	PD_A	0 = Normal operation (default); 1 = Power-down for channel pair A enabled (bit 29 must be '0' in this case)	0
C12	Don't use		0
C11	CLKSEL	0 = Normal operation with internal conversion clock (mandatory in hardware mode) (default); 1 = External conversion clock (applied through pin 27) used	0
C10	CLKOUT_EN	0 = Normal operation (default); 1 = Internal conversion clock available at pin 27	0
C9	REFDAC[9]	0 = BUSY active high while INT active low (default); 1 = BUSY active low while INT active high	1
C8	REFDAC[8]	Bit 8 of reference DAC value; default = 1	1
C7	REFDAC[7]	Bit 7 of reference DAC value; default = 1	1
C6	REFDAC[6]	Bit 6 of reference DAC value; default = 1	1
C5	REFDAC[5]	Bit 5 of reference DAC value; default = 1	1
C4	REFDAC[4]	Bit 4 of reference DAC value; default = 1	1
C3	REFDAC[3]	Bit 3 of reference DAC value; default = 1	1
C2	REFDAC[2]	Bit 2 of reference DAC value; default = 1	1
C1	REFDAC[1]	Bit 1 of reference DAC value; default = 1	1
C0	REFDAC[0]	Bit 0 of reference DAC value; default = 1	1

Table 2.4: ADS8556 Config Register values

Parameter	Minimum	Typical	Maximum
Bits available		16 bidirectional	
Clock Frequency FSCLK			10MHz
Supply Voltage, VDD to VSS	1.8V		5.5V
High-level input voltage	0.25×VDD +0.8		VDD +0.3
Low-level input voltage	VSS		0.15×VDD

Table 2.5: MCP23S17 main specifications

3 Design and implementation

The solution proposed, required the independent design of each part that composed the full system. In this chapter is introduced the design steps of each section.

3.1 Control board

3.1.1 Hardware design

As was already mentioned, the main objective of this work was to design a control-board fully integrated with LinuxCNC. In Chapter 2 was introduced all the technologies required to get the objectives.

In this section, was related the design process of the hardware. The hardware is divided in several parts:

- **Power supply** : The voltage levels required for the device that compose the board is given by a standard ATX computer power supply. In this design were added some capacitors to reduce the rip produced by long routes in PCB's.
- **Analog stage** : This part is the hardware needed to analog to digital conversion. The main part is the ADS8556 and the rest of the hardware is designed to get a good performance of this device.
- **Steppers drivers** : The steppers-motors require the use of the A4983(2) driver. This driver does not need complex circuits and just need a socket to connect the digital signals required. The digital signals used to controls these drivers, are produced by MCP23S17.
- **PWM outputs** : To produce PWM 12V level outputs was required to use N-MOS transistors, in order to get enough current. These transistors are controlled by digital signals produced by MCP23S17.

3.1.2 ADC stage design

The core of the ADC stage is the ADS8556(15) from Texas Instruments. This chip have six ADC channels

The ADC stage was designed to achieve the requirements specified in Table 3.1 .

These features were selected having in mind that this board can be used in different applications. The voltage range was selected in order to have the best resolution possible for the ADC chip selected that works whit ranges of signals below the intern programmable reference.

Voltage range	0V - 2.5V
Maximum Frequency	25kHz

Table 3.1: ADC input signal characteristics

According to the data-sheet, the ADS8556(15) is an SAR C-ADC. It means that to have a good performance ADS8556 required sample and hold input circuit to achieved a good performance.

The Figure 3.1 shows the equivalent circuit at the input.

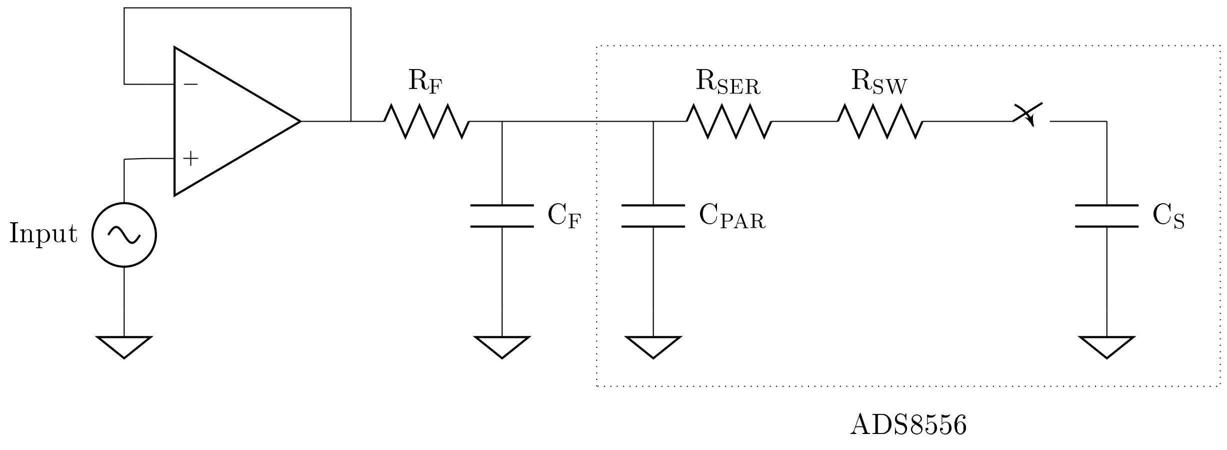


Figure 3.1: Equivalent circuit at the input of the ADS8556.

It is required some data from the ADS8556 data-sheet, summarized in Table 3.2.

Acquisition time tAQ	280ns
SER Resistor RSER	200Ω
SW Resistor RSW	130Ω
PAR Capacitor CPAR	5pF
CS Capacitor CS	20pF

Table 3.2: Information obtained from AS8556 data-sheet.

With this information, we can calculate the values of the external components, given by the following formulas:

$$t_{AQ} \geq k_1 \times \tau$$

$$k_1 = (N + 1) \times ln(2)$$

$$\tau = R_F \times C_F$$

Simplifying:

$$R_F \times C_F \leq \frac{t_{AQ}}{(N+1) \times ln(2)}$$

It is necessary to consider that the circuit introduces one pole and one zero. That can cause instability in the closed loop circuit. The OPAMP was selected to avoid any instability in the circuit.

The equations that describe the pole and the zero added are the following:

$$f_{PX} = \frac{1}{2\pi \times (R_0 + R_F) \times C_F}$$

$$f_{ZX} = \frac{1}{2\pi \times R_F \times C_F}$$

The gains in them are:

$$G_{PX} = -20 \times log(\frac{f_{PX}}{f_U})$$

$$G_{ZX} = G_{PX} - 40 \times log(\frac{f_{ZX}}{f_{PX}})$$

In summary, the design equations are as follows:

- Calculate time-constant multiplier

$$k_1 = (N+1) \times ln(2)$$

- Determine minimum time-constant

$$\tau \leq \frac{t_{AQ}}{k_1}$$

- Calculate frequency of added zero

$$f_{ZX} = \frac{1}{2\pi \times \tau}$$

- Find UGWB

$$GBW = 4 \times f_{ZX}$$

With this information, was selected the OPA365(17) with the characteristics summarized in the Table 3.3

Gain Band Width GBW	50MHz
Slew Rate SR	25V/μs
RSW	130Ω
CPAR	5pF
CS	20pF

Table 3.3: OPA365 specifications

After the selection of the ADC and the OPAMP, we can choose the external components, using the following equations:

- Determine the capacitor

$$20 \times C_{SH} \leq C_F \leq 60 \times C_{SH}$$

- Determine the resistor

$$R_F = \frac{1}{2\pi \times f_{ZX}}$$

- Verify the value of the resistor

$$R_F \geq \frac{R_0}{9}$$

- Calculate frequency of added pole

$$f_{PX} = \frac{1}{2\pi \times (R_0 + R_F) \times C_F}$$

- Keep added pole and zero less then decade a part

$$f_{PX} \geq \frac{1}{10} f_{ZX}$$

In the design, discrete components were selected as follows:

- $$k_1 = (N+1) \times ln(2) = (16+1) \times ln(2) \approx 11.78$$

- $$\tau \leq \frac{t_{AQ}}{k_1} \approx 23.77ns$$

- $$f_{ZX} = \frac{1}{2\pi \times \tau} \approx 6.69MHz$$

- $$GBW = 4 \times f_{ZX} \approx 26.78MHz$$

- $$20 \times C_{SH} \leq C_F \leq 60 \times C_{SH} \Rightarrow 400pF \leq C_F \leq 1200pF \Rightarrow C_F = 470pF$$

- $$R_F = \frac{1}{2\pi \times C_F \times f_{ZX}} \Rightarrow R_F = 43\Omega$$

- $$R_F \geq \frac{R_0}{9} \Rightarrow 50 \geq 3.33$$

- $$f_{PX} = \frac{1}{2\pi \times (R_0 + R_F) \times C_F} \approx 4.64MHz$$

- $$f_{PX} \geq \frac{1}{10} f_{ZX} \Rightarrow 4.64MHz \geq \frac{1}{10} f_{ZX}$$

3.1.3 PWM stage design

To use devices that require power PWM such as DC-motors, heaters, light generators, in this work was designed a driver circuit that can work with 12V levels. This circuit is showed in the Figure 3.2.

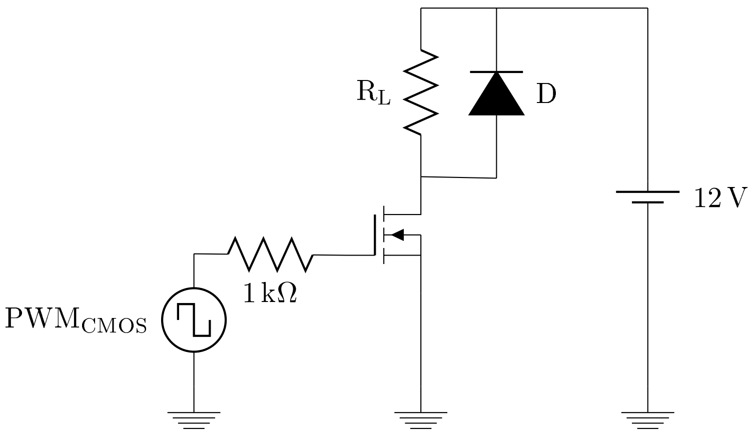


Figure 3.2: PWM driver circuit.

In this circuit, RL represents the load and D is the typical clamp diode. Into the circuit implemented in the PCB, was added a LED to have a visual reference, this sub-circuit is not explained in deep because it does not interfere in the final functionality.

The principal part of this circuit is an N-Channel MOSFET. The selection of the MOSFET, was made keeping in mind the characteristics summarized in the Table 3.4

Min frequency	1kHz
Max voltage in RL	12V
Max power dissipation for a 1Ω	24W

Table 3.4: Target features of the PWM stage

Whit these parameters were selected the IRF630B(1) N-Channel MOSFET. The principal features of IRF630B are showed in Table 3.5

The circuit designed respects the SOA (Safe Operating Area) as shows Figure 3.3. So there is no problem with the component safety.

From the data-sheet, it was obtained the Figure B.1, that shows the switching time response that have to bear take into account to avoid problems working at the target frequency.

Max Drain-Source Voltage V$_{DSS}$	200V
Max Drain Current I$_D$	9A
Min Gate Threshold Voltage V$_{GS_{(th)}}$	2V
Max Turn-On Delay Time t$_{d_{(on)}}$	30ns
Max Turn-On Rise Time t$_r$	150ns
Max Turn-Off Delay Time t$_{d_{(off)}}$	130ns
Max Turn-Off Fall Time t$_f$	140ns

Table 3.5: IRF630B characteristics

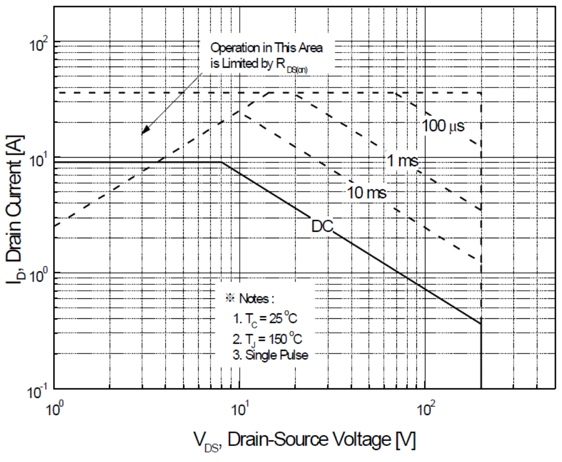


Figure 3.3: Safe Operating Area of IRF630(1)

With the information in IRF630(1) data-sheet, the time parameters are as follows:

$$t_{on} = t_{d_{(on)}} + t_r = 180ns$$

$$t_{off} = t_{d_{(off)}} + t_f = 270ns$$

$$t_{off} + t_{on} = 450ns$$

To achieve the target frequency, and use a 1% resolution for the PWM, the total time must be 1% of the period, ergo have to be bigger than

$$t_{off} + t_{on}$$

. Then :

$$t_{off} + t_{on} < 10\mu s$$

The transistors do not introduce any timing restriction to achieve the target frequency of 1KHz with 1% resolution in duty-cycle.

3.1.4 Steppers control stage

As was already mentioned, the A4983 do not require complex circuits to integrate into the design of PCB.

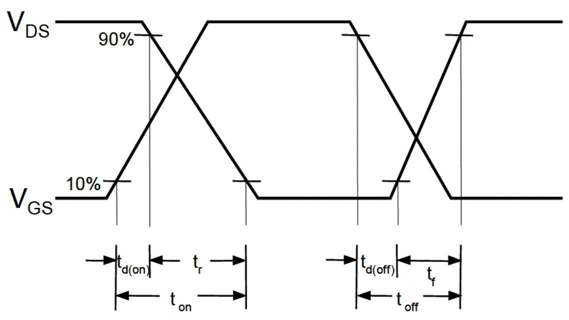
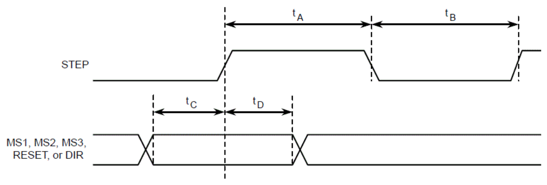
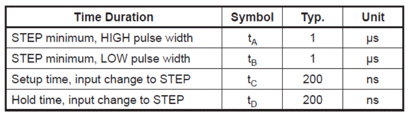


Figure 3.4: Timing characteristics of IRF630(1)





Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t_A	1	μs
STEP minimum, LOW pulse width	t_B	1	μs
Setup time, input change to STEP	t_C	200	ns
Hold time, input change to STEP	t_D	200	ns

Figure 3.5: A4983 timing characteristics(2)

For the steppers control stage, it is necessary to consider, that the MCP23S17(16) have to support the maximum frequency that the A4983(2) can use. The Figure 3.5

In this design, the maximum frequency is given by the LinuxCNC configuration file and can be selected by the user having in mind the timing restrictions due to the latencies in Raspberry-Pi.

3.1.5 PCB design, assembly and test operation of the hardware

Once designed all the sub-circuits required to achieve the objectives, then start to design process of the PCB[1]. It was an important part of this milestone, reading, understanding and implementation all the recommendations for devices used.

It is not the aim of this work, to detail these recommendations, anyhow it does not modify the design criteria set.

The corresponding schematic is attached in following sections.

3.2 HAL Driver

3.2.1 Hardware Abstraction Layer

As was presented in Chapter 2, HAL(18) is the highest level of a component in LinuxCNC. It means that any complex system is composed by several HAL blocks.

In this work was implemented a HAL component that controls the board. This driver, describe the functionality to control the hardware communication and the state of inputs and

[1]To design the schematic and routing, the Eagle CAD software was used.

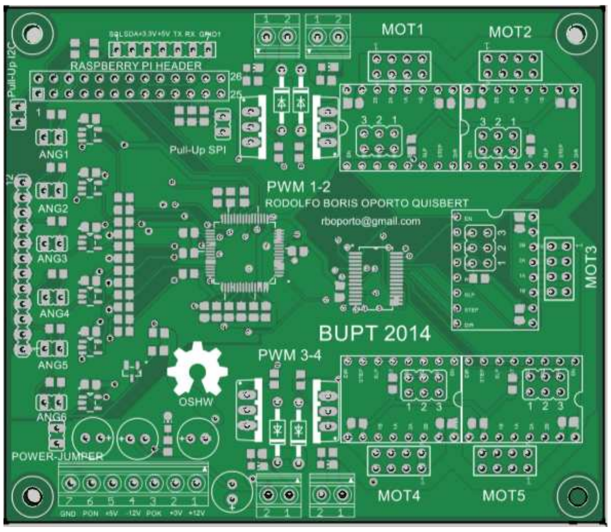


Figure 3.6: 3D simulation of the PCB.

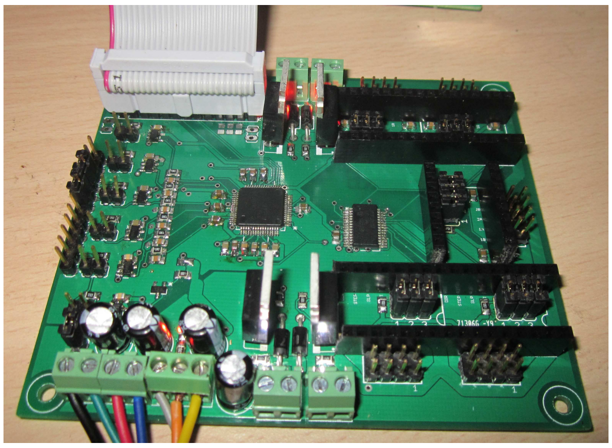


Figure 3.7: Final appearance of the PCB designed.

outputs. It means that in a LinuxCNC environment the integrator can forget about the hardware composition and communications so the developer can focus on composing the different systems with these components, treating HAL components as simple blocks as is showed in Figure 3.9.

3.2.2 Driver description

The Figure 3.9 shows the HAL driver that describe the functionality interacting with hardware at low level. It means that any hardware interactions will be controlled by HAL driver and at the same time this driver will update inputs and outputs that describe the functionality to interact with others HAL components.

The Table 3.6 summarize the inputs and outputs of the block and its functionality.

3.2.3 Driver functions

The driver requires real-time functions to interact with the hardware. As was described in Chapter 2, these functions use a real-time thread component to execute with certain frequency. These functions are not executed until the thread reach the period time.

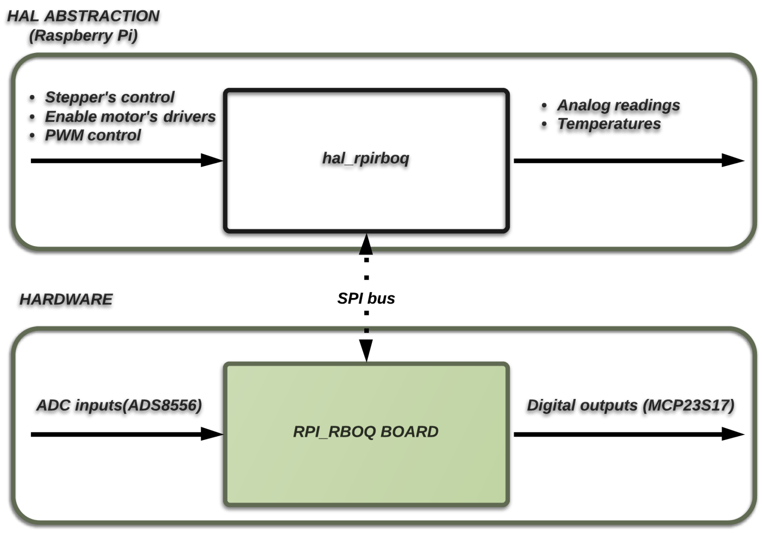


Figure 3.8: Relation between HAL abstraction and hardware

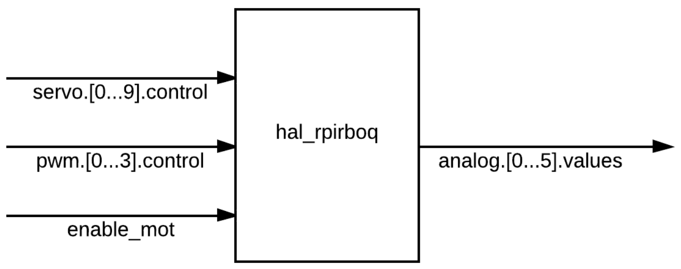


Figure 3.9: Generated HAL block of the driver

The functions implemented by the rpi_rbopq board are described in the Table 3.7.

These functions are not complicated, most of the code is to control the SPI registers in order to get communication with MCP23S17 and ADS8556. Is not the aim of this work to get in deep about the control of BCM2835 in register level. For more information, consult the BCM2835 ARM Peripherals manual(19).

Name	**Length**	**Type**	**Description**
servo.*i*.control	10	bit	Steppers control pins for rpirboq driver. 0:STEP_MOT1, 1:DIR_MOT1, 2:STEP_MOT2, 3:DIR_MOT2, 4:STEP_MOT3, 5:DIR_MOT3, 6:STEP_MOT4, 7:DIR_MOT4, 8:STEP_MOT5, 9:DIR_MOT5.
enable_mot	1	bit	Motors enable signal.
pwm.*i*.control	4	bit	PWM outputs for rpi_rboq board.
analog.*i*.values	6	float	Analog inputs readed by the ADS8556 (in Volts).
temp.*i*.thermistor	6	float	Analogue outputs read by the ADS8556 (in Celsius) with a 10kΩ thermistor connected [2].

Table 3.6: Driver's inputs and outputs description.

Name	**Functionality**
update_rpirboq	Update the signals values of servo.i.control, enable_mot, pwm.i.control. To do this, communicate by SPI with the MCP23S17 and upgrade the GPIO registers to change the outputs in PORTA and PORTB.
read_rpirboq	Read and update analog.i.in inputs. To do this, communicate by SPI with the ADS8556, as we described in Chapter 2, and update the information with the values obtained.

Table 3.7: Driver real time functions

4 Results

4.1 Latency test

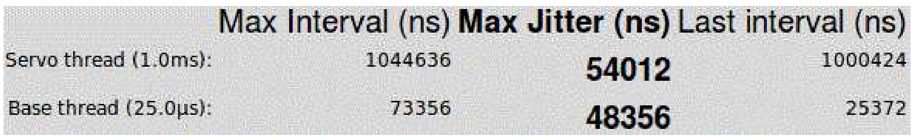
In LinuxCNC integrators manual(18) the authors explain the steps in order to set a CNC machine.

The most important test is the called latency-test that obtain the maximum latency value that a determinate hardware allow to use.

This test execute as follows:

- This test start two real-time threads and perform testing real-time functions.
- The user interface present the average of latency in both, and the most important the maximum jitter value that can be interpreted as the minimum period that can be used to execute real-time functions.

The Figure 4.1 shows the result of latency-test execution.



	Max Interval (ns)	**Max Jitter (ns)**	Last interval (ns)
Servo thread (1.0ms):	1044636	**54012**	1000424
Base thread (25.0µs):	73356	**48356**	25372

Figure 4.1: Latency-test results in Raspberry Pi

In this work, was configure the BASE-THREAD parameter (the fastest frequency) to 150μ to avoid any problem during execution.

4.2 Board tests

Before to integrate all the system, separated parts of the board was evaluated in order to assure that the board can perform the required actions.

4.2.1 ADC test

To test the ADC, was programmed a Xenomai user-space program called `test_adc_csv.c`. This code, use the libraries written to control the ADS8556 device: `ads8556.c` and `ads8556.h`.

This code sample each particular period, the signal in all the analog channels and then storage this information in a CSV file. For more information about the code, check the corresponding manual.

In Figure 4.2 is showed the result of sampling 500Hz, 4Vpp sinusoidal signal, sampled at 200μs period.

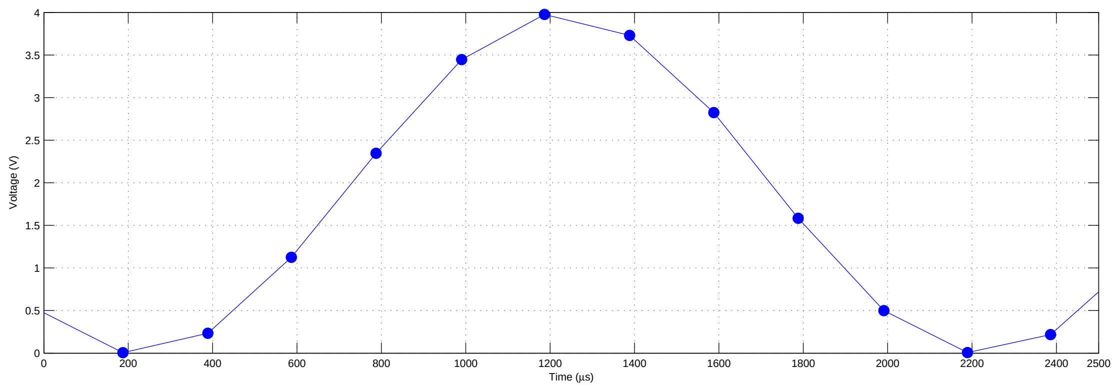


Figure 4.2: Sinusoidal signal: 500Hz, 4Vpp, sampled at 200μs period.

The time in Figure 4.2 was shifted for representation purposes. The values written in CSV are the decimal representation of the code obtained by the ADC.

4.2.2 PWM

To validate the PWM stage, in this work was used the PWMGEN component. This component allows generate a PWM signal in the PWM outputs.

Some testing software was written adapting software(20) from the LinuxCNC project.

This software, generate a PWM of 100Hz of frequency and the normalized duty-cycle can be with the scroll-bar. The signal generated will be showed in the halscope window. The aspect of the application can be seen in Figure 4.3

Figure 4.4 shows the signal with 1% of dutycycle, Figure 4.5 shows the signal with 25% duty-cycle, Figure 4.6 shows the signal with 50% duty-cycle, Figure 4.7 shows the signal with 75% duty-cycle and Figure 4.8 shows the signal with 99% duty-cycle.

4.3 Full System Test

As was already mentioned to test the entire system, was used a Prusa i3 as a machine to be controlled by the rpi_rboq board.

- **Test 1, Motion:** The motion system was tested, using the standard test of LinuxCNC. In this test, the machine uses a pen to draw some letters.

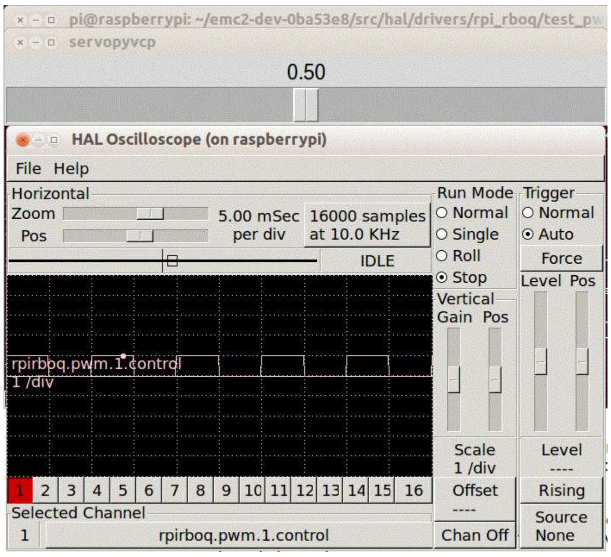


Figure 4.3: LinuxCNC running PWM test

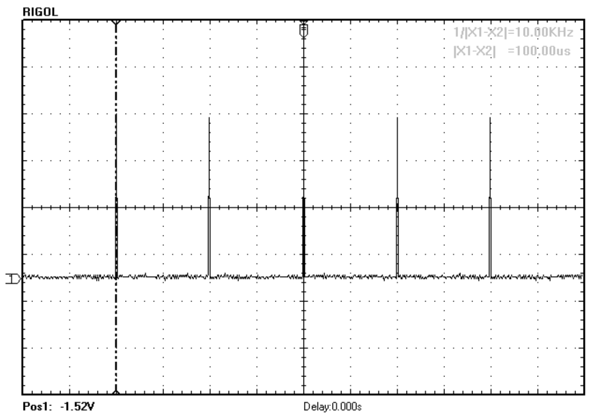


Figure 4.4: 100Hz PWM signal 1% duty-cycle

- **Test 2, Temperature:** In this test the temperature in constantly obtained through one of the analog inputs. With this information, is controlled a PID block that generate a PWM from one of the PWM outputs of the board.

The reason because the validation is into two tests is because the first test, is designed to verify the CNC standard functionality (move and control three axis) and the second is designed to verify the temperature control wide used in 3D printing (e.g. melting materials, heating surfaces).

4.3.1 Test 1: Motion

As was already told, this the was designed to validate the motion of a 3D printer.

The Figure 4.9 shows the diagram that illustrates the blocks that interact in this test.

The functionality of each one are as follows:

- **User Interface:** LinuxCNC offers several user interfaces. In this work is used Axis, which is the most common among LinuxCNC users. From this environment, the user can control

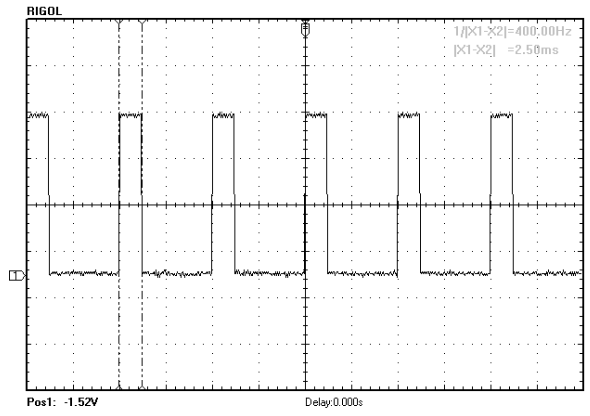


Figure 4.5: 100Hz PWM signal 25% duty-cycle

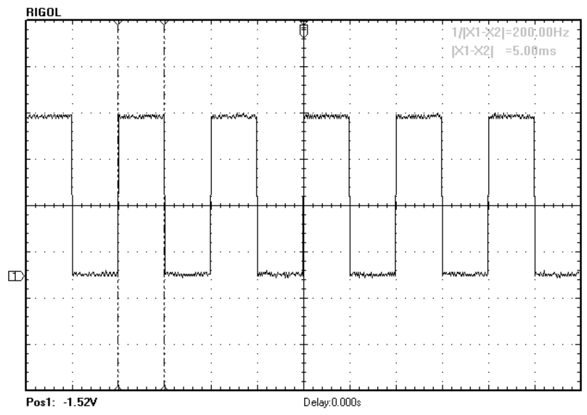


Figure 4.6: 100Hz PWM signal 50% duty-cycle

the XYZ axis, select G-Code, set variables (e.g. homing the axis, set speeds). When one works start, LinuxCNC read a G-Code file, interpret this information and execute the actions through the stepgen components.

- **Step-generator:** Each one of the axis have a stepgen component. Stepgen control and create the STEP and DIR signals to produce motion in steppers-motors. Stepgen is with rpi_rboq block and update the values in the stepper-control signals defined in the driver.
- **rpi_rboq:** This is the block that represents the board in HAL layer. This block executes the read and update functions, with the frequency defined in the configuration files. When the update function is , the information in servo-control variables (previously updated by stepgen block) is transmitted to the board and the board generated the digital signals.

In Figure 4.10 and Figure 4.11 we can see the testing process and final result.

4.3.2 Test 2, Temperature:

To test the temperature controller, was implemented a test that include a PID controller and PWM generator. The Figure 4.12 shows the diagram that illustrates the blocks that interact in this test.

The functionality of each one are as follows:

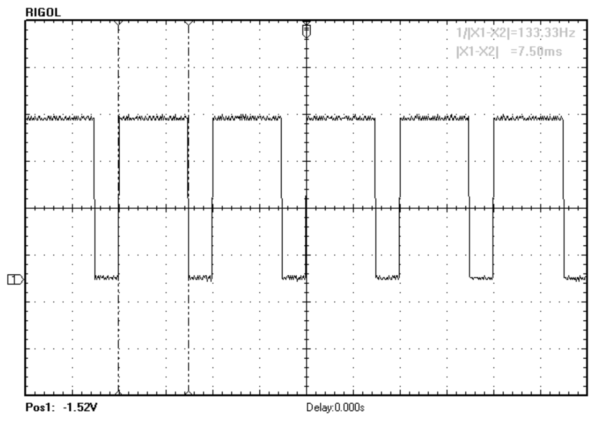


Figure 4.7: 100Hz PWM signal 75% duty-cycle

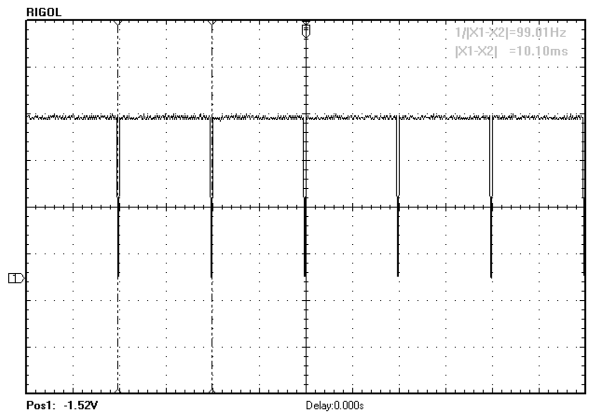


Figure 4.8: 100Hz PWM signal 99% duty-cycle

- **User Interface:** In this test was added a user interface panel to control the PID. This panel has a scroll bar that can be used to fix the voltage obtained from the thermistor. The user can also see the PID gain value, see the voltage in ADC and the PWM generated, the Figure 4.13 shows this panel.
- **PID controller:** This block represent an standard PID controller. This controller uses the analog-value signal as error input, to control the PWM duty-cycle. The output of the PID block is connected to the PWM generator.
- **PWM generator:** The PID generates an output that is proportional with the PWM duty-cycle to generate. Other parameters are in the configuration file. In this test, was used a 100Hz base and 75% duty-cycle at most to avoid any problem with the transistors.
- **rpi_rboq:** As is the other example, the rpi_rboq controls the inputs and outputs through SPI bus.

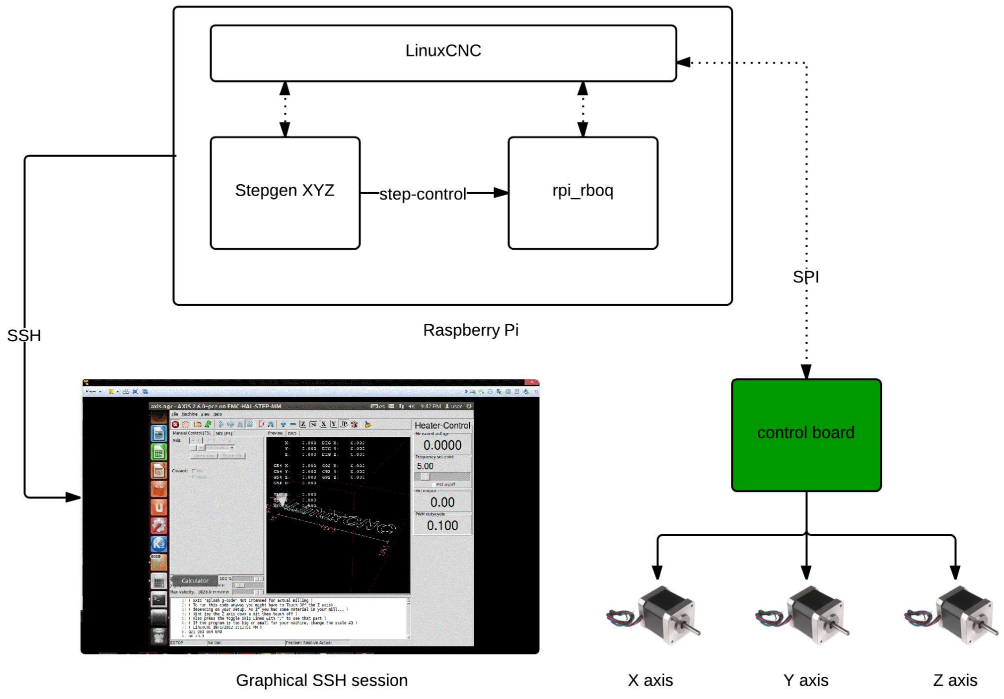


Figure 4.9: Component diagram of the motion test

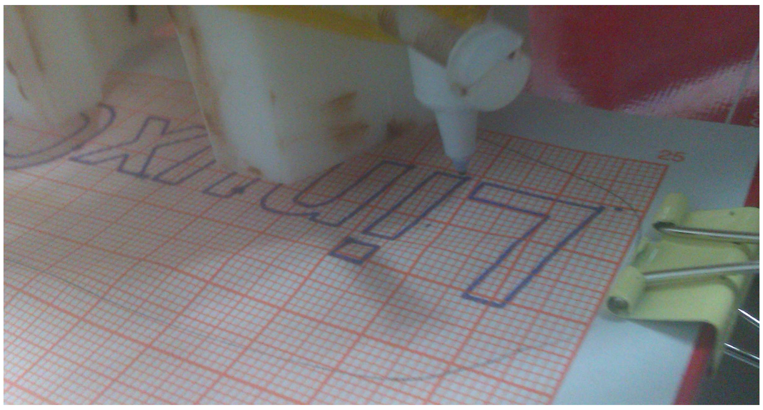


Figure 4.10: Motion testing process

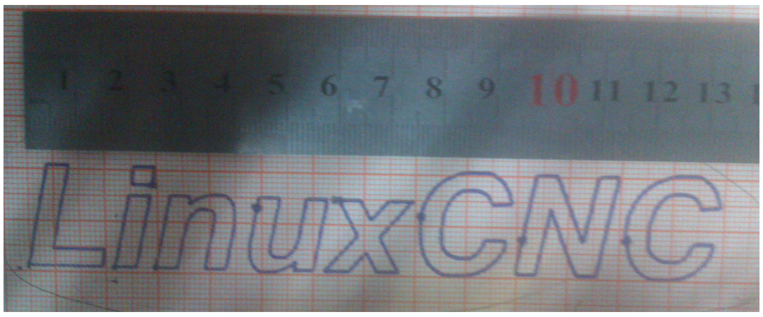


Figure 4.11: Result motion test

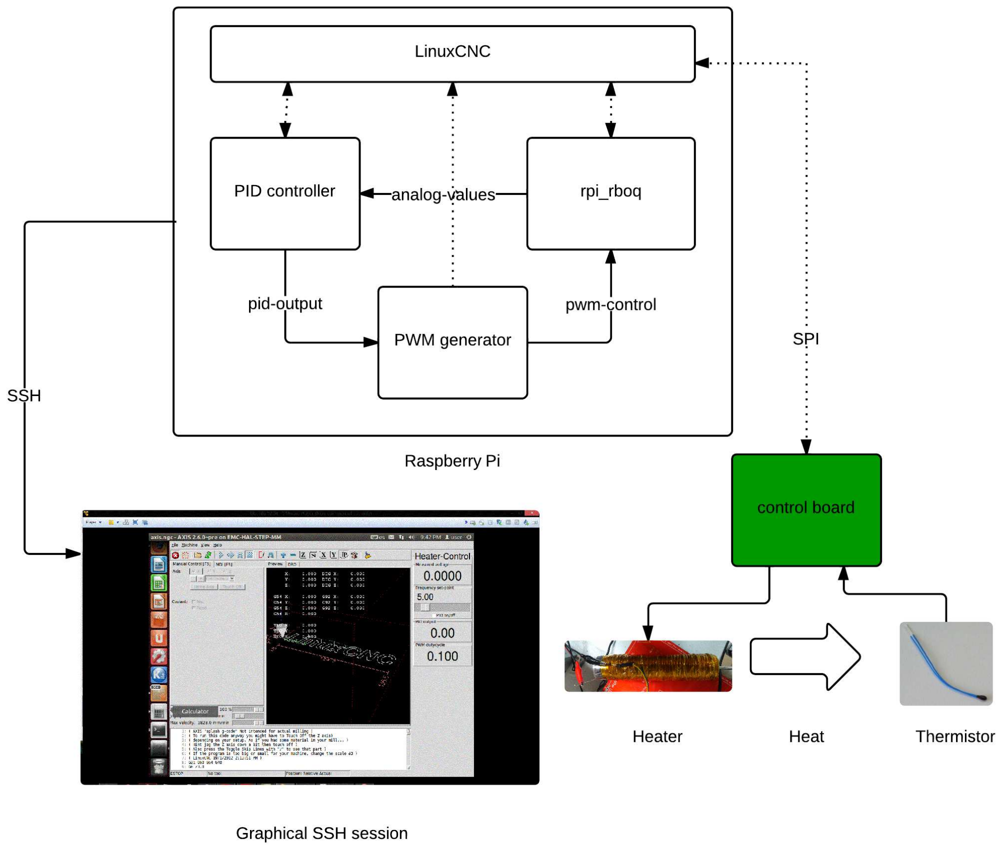
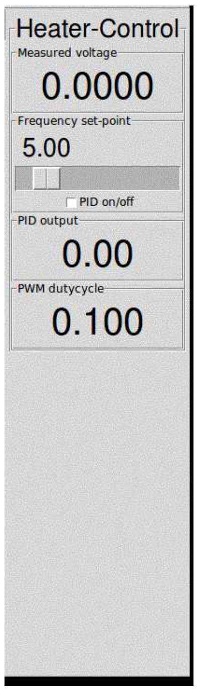


Figure 4.12: Component diagram of the temperature test


Heater-Control

Measured voltage

0.0000

Frequency set-point

5.00

PID on/off

PID output

0.00

PWM dutycycle

0.100

Figure 4.13: Control panel PID/PWM

5 Conclusions and future work

5.1 Conclusions and results discusion

The aim of this work was to create a fully functional control-board for controlling CNC systems based in Open Source solutions. These solutions and its functionality were introduced in Chapter 2.

The design process of all the parts that make the full system was related in Chapter 3.

As was presented in Chapter 4, most of the functionality was achieved validating and experimenting with all the implied parts, first of all with separate tests and after controlling the full integrated system.

In general, the board designed, achieved a good integration between Rapsberry-Pi and Xenoma.

When the system works integrated with LinuxCNC through the HAL driver, was detected some limitations due to the minimum period that Raspberry-Pi can offer without delays. In the latency test, the board shows that could works without problems at 50μs as the minimum period. This circumstance, reduce the objectives that were established for each stage of the board in Chapter 3.

- **PWM stage:** If the application require a 1% of resolution in the duty-cycle, the maximum frequency will be 200Hz(100 $\times$ 50μs). The frequency achieved in practice, is not the 1kHz frequency that was setting as objective to reach, but is enough to control the experimental heating system and standard servomotors that work with 20ms base period.
- **STEPPERS stage:** The same restriction than in PWM are applied to STEPPER drivers. The maximum step rate will be 200 steps/second (200 pulses in STEP signal), this number correspond to one revolution/second in angular unities. Some interesting improvement could try to improve this rate modifying the STEPGEN parameters, which can be done in future improvements of the system.
- **ADC stage:** If the ADC have the minimum sample period of 50μ it means that the board can sample at 20kHz. It means that the Nyquist frequency is 10kHz so the system can sample 10kHz signals which are far from the 20kHz that was established to be able to

sample audio signals. Anyhow, for this application this rate is enough, because temperature does not require to be checked more than each millisecond.

5.1.1 Future work

Is this work we proposed the following improvements in the board, in order to achieve a better performance:

- Review the layout to correct some mistakes that can be the cause of excessive noise when the system works with SCLK(SPI) at 31.MHz or higher, even if the ADS8556 can use higher frequency rates.
- Review of the layout to add more digital signals, in order to control digital controls like end stops or auxiliary devices. The addition of more digital pins need to avoid the increasing of SPI cycles to write/read these digital pins because the communication time can increase the minimum period of operation with SPI working near the device limits.
- Review the software to add Direct Memory Access (DMA) support. Using DMA in the implementation of the reception and sending buffer of the SPI, can increase the maximum transmission rate.

A
Estimate budget

1) Budget	
• Personal computer	2850 RMB
• PCB manufacture	400 RMB
• Integrated electronic components	308 RMB
• Pasive electronic devices	50 RMB
• Total material execution	3608 RMB
2) Project fees	
• 1800 hours at 135/ hour	243000 RMB
3) Consumables	
• Printing costs	240 RMB
• Book binding	20 RMB
4) Total budget	
• Total budget	250260 RMB

Beijing, April 2014

The Chief Project Manager

Signed.: Rodolfo Boris Oporto Quisbert

Graduate in Electronic and Communications Engineering

B
Board schematics

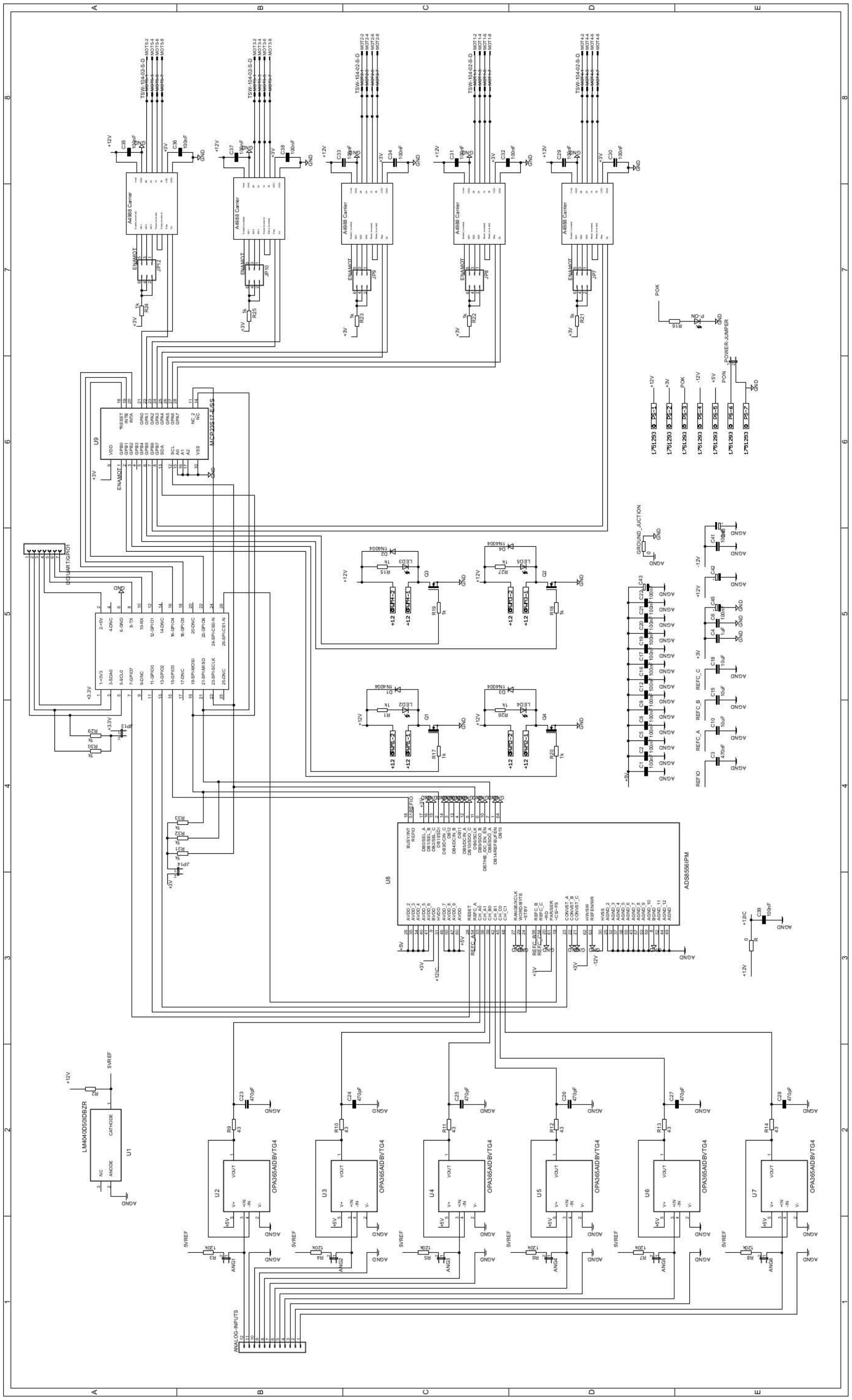


Figure B.1: Designed board schematic.

C
User Manuals

C.1 Rpi_rboq board

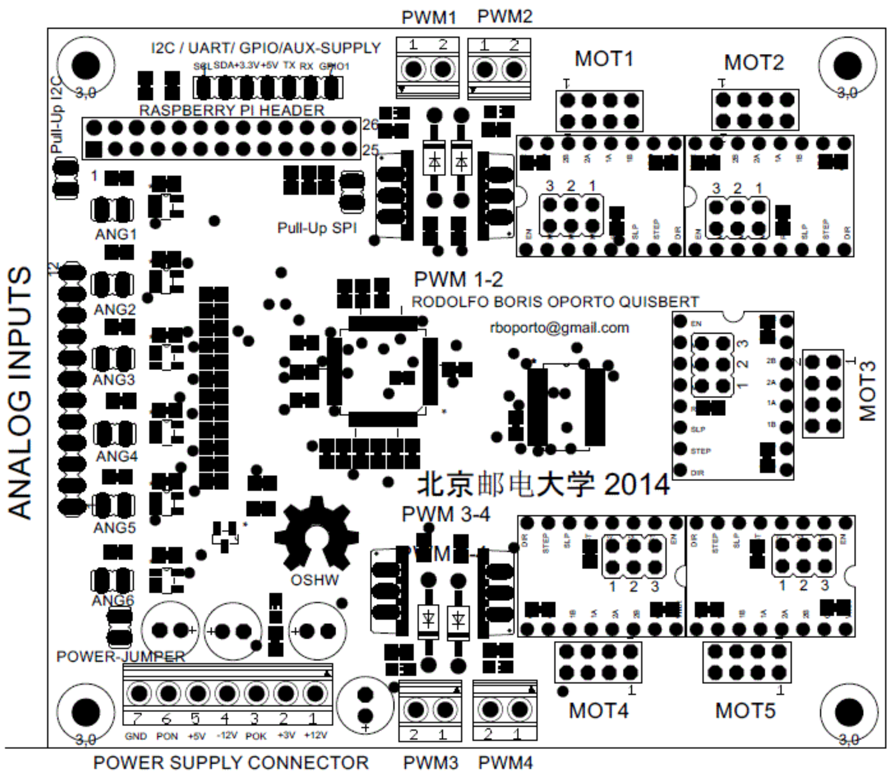


Figure C.1: Rpi_rboq board component placement.

To see the functionality of each block and details of the design process, read the board description chapter.

C.1.1 Preoperative settings

First of all, the user has to connect the rpi_rboq board with the Raspberry-Pi. To do this, the user can use a flat 2X13 connector. The Figure C.2 shows in detail the R-Pi connection and configuration jumpers.The Figure C.3 shows the board connected with Rpi.

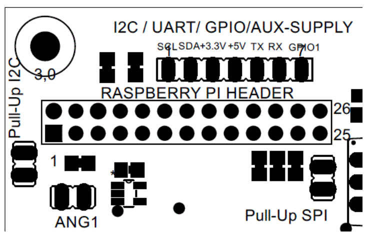


Figure C.2: R-pi connector detail.

The *Pull-up SPI* and *Pull-up I2C* jumpers connect the pull-up resistors of SPI and I2C interfaces if necessary.

To use the board, is needed to connect a [1] power supply as is showed in Figure C.4.

To power on the board, the user just have to connect the *Power-jumper*.

[1]In this work was used an ATX Standard power supply, to get the voltage levels required.

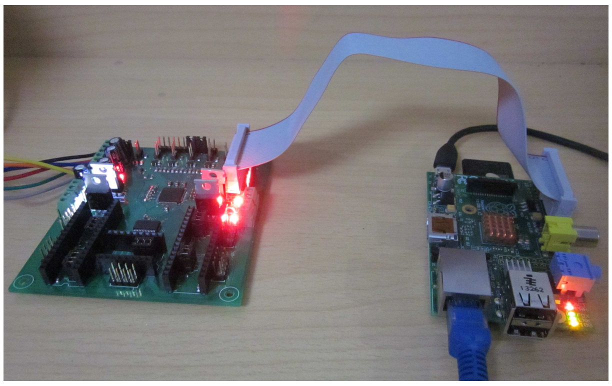


Figure C.3: Connection of the PCB with RPI

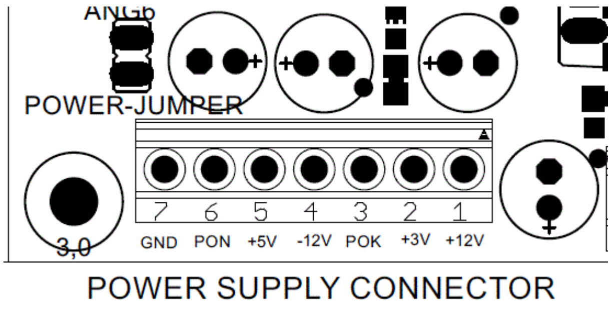


Figure C.4: Connection of power supply

C.1.2 ADC preoperative settings

The ADC block control is a 12X1 connector and 6 jumpers as shows Figure C.5 .

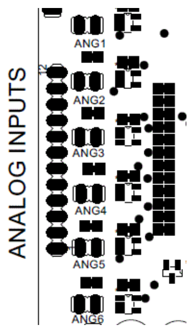


Figure C.5: ADC connectors detail.

The ANALOG INPUTS pins are assigned as showed in Table C.1

The jumpers are used for controlling using thermistor or others resistance dependant transducers. Any jumper connects the voltage reference, that provide a constant and very stabilized 5V level refcap:, which connects with a 120kΩ resistor making a voltage divisor [2].

C.1.3 Stepper-Motors preoperative settings

The Figure C.7 shows the connectors of a single stepper-motor control block.

[2]The 120kΩ was selected to get the best performance with 100kΩ thermistor.

Pin	**Function**
1	AGND: Analog ground.
2	CH_C1 : CH_C1 signal.
3	AGND: Analog ground.
4	CH_C0 : CH_C0 signal.
5	AGND: Analog ground.
6	CH_B1 : CH_B1 signal.
7	AGND: Analog ground.
8	CH_B0 : CH_B0 signal.
9	AGND: Analog ground.
10	CH_A1 : CH_A1 signal.
11	AGND : Analog ground.
12	CH_A0 : CH_A0 signal.

Table C.1: Analog inputs header pinouts.

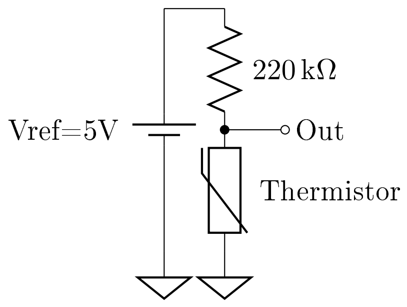


Figure C.6: Equivalent circuit when ANGi jumpers are connected.

It is composed of 8X2 baseboard of an A4988 stepper motor driver[3].

The MOTi connector allows the user to connect two bipolar stepper motors to the same driver if necessary. One of these steppers have to be connected in pair pins and the other in odd.

The most significant preoperative setting is the micro-stepping resolution. It is done by the jumpers 1, 2 and 3. The Table C.2 summarize the stepping depending on the jumper connection [4].

Microstep Resolution	**J1**	**J2**	**J3**
Full step	Not connected	Not connected	Not connected
Half step	Not connected	Not connected	Connected
Quarter step	Not connected	Connected	Not connected
Eighth step	Not connected	Connected	Connected
Sixteenth step	Connected	Connected	Connected

Table C.2: Micro stepping selection jumpers

C.1.4 PWM preoperative settings

The PWM does not require any special configuration. The user just has to connect the device in the connector as is showed in Figure C.8.

C.2 HAL drivers installation and testing

[3]Following the recommendation of the maker, it has to be extremely careful connecting the driver.
[4]This is especially important to keep in mind at the moment to configure the

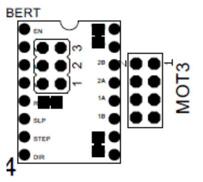


Figure C.7: Stepper motors block detail

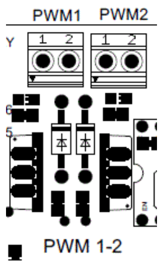


Figure C.8: PWM block detail.

C.2.1 Driver and man page installation

To install the HAL driver and the manual page in the operative system, the user have to invoke the following command .

```
sudo comp --install --install-doc hal_rpirboq.comp
```

This line will automatically install the driver and manual page that can be consulted with the command:

```
man hal_rpirboq
```

C.2.2 Testing programs

Once installed, the user can invoke the hal_rpirboq as a standard HAL LinuxCNC component. In the attached CD is the full source of the driver and the demo configuration files. You can invoke the full environment with the following line[5]:

```
linuxcnc stepper_mm.ini
```

LinuxCNC can open a graphical session from a remote client. To do this use the SSH command as follows[6]:

[5]Invoke from /home/pi/linxcncdemo/.

[6]Where <ip-server> is the IP address of the Raspberry-Pi.

```
ssh -X -l <ip-server>
```

Another testing software can be invoked from the halrun console. This command started a test program that moves one stepper connected to MOT1 See Chapter 2 in increasing and decreasing speed, blinks the PWM1 with different duty cycles and print the value of the ADC1 inputs in Volts. We have to proceed as follows, from the Linux batch:

```
halrun
```

And from the halrun console:

```
source hal_test.hal
```

C.3 MCP23S17 and ADS8556 libraries

To use these libraries, the user have to install the bcm2835 GPIO librariesThe running version of this test libraries use the bcm2835 1.36v. from:

```
http://www.airspayce.com/mikem/bcm2835/index.html
```

An Html manual is available in the CD attached with this work.

Bibliography

[1] Fairchild Semiconductor, "Irf630b/irfs630b: 200v n-channel mosfet,"

[2] Allegro Micro Systems, *A4983: DMOS Microstepping Driver with Translator Â*.

[3] Raspberry Pi Foundation, "Raspberry pi foundation website." http://www.raspberrypi.org, March 13 2014.

[4] Raspbian Project, "Raspbian free operative system website." http://www.raspbian.org/FrontPage, March 13 2014.

[5] Rapsberry-Pi Foundation, "Rapsberry-pi foundation: Git repository." https://github.com/raspberrypi/, March 13 2014.

[6] Xenomai Project, "Xenomai: Git repositories, rapsberry-pi patches." http://git.xenomai.org/xenomai-2.6.git/plain/ksrc/arch/arm/patches/raspberry/, March 13 2014.

[7] eLinux Project, "Raspberry-pi kernel compilation." http://elinux.org/Rpi_kernel_compilation, March 13 2014.

[8] RT Linux Team, "Real-time linux wiki." https://rt.wiki.kernel.org/index.php/Main_Page, March 13 2014.

[9] P. Gerum, "Xenomai-implementing a rtos emulation framework on gnu/linux," *White Paper, Xenomai*, 2004.

[10] J. H. B. . B. Martin, "How fast is fast enough? choosing between xenomai and linux for real-time applications,"

[11] LinuxCNC Team, "Linuxcnc organization webpage." http://www.linuxcnc.org/, March 13 2014.

[12] LinuxCNC Team, "Manual pages of hal components." http://www.linuxcnc.org/docs/html/man/, March 2014.

[13] LinuxCNC Team, *LinuxCNC: HAL Manual V2.5, 2014-05-09.*

[14] RepRap Project, "Prusa i3." http://reprap.org/wiki/Prusa_i3, March 13 2014.

[15] Texas Instruments, "Ads8556: 16-, 14-, 12-bit, six-channel, simultaneous sampling analog-to-digital converters,"

[16] Microchip, "Mcp23s17/mcp23017: 16-bit i/o expander with serial interface,"

[17] Texas Instruments, *OPA365: 50MHz, Low-Distortion, High CMRR, RRI/O, Single-Supply OPERATIONAL AMPLIFIER.* Texas Instruments.

[18] LinuxCNC Team, *LinuxCNC: Integrator Manual V2.5, 2014-04-23.*

[19] Broadcom Corporation, *BCM2835 ARM Peripherals*. Broadcom Corporation.

[20] LinuxCNC Team, "Rc servo test." http://wiki.linuxcnc.org/cgi-bin/wiki.pl?RC_Servo_Test, March 2014.