TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JAVIER PÉREZ DE FRUTOS

LOW-COST STANDALONE COMMUNICATION INTERFACE FOR DATA ACQUISITION SYSTEM IN MICROROBOTIC PLATFORM

Master of Science Thesis

Examiner: Prof. PASI J. KALLIO
Supervisor: MATHIAS VON ESSEN
Examiners and topic approved by the Council of the Faculty of Engineering Sciences on 9th of April 2014

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
JAVIER PÉREZ DE FRUTOS : Low-Cost Standalone Communication Interface For Data Acquisition System In Microrobotic Platform
Master of Science Thesis, 73 pages, 12 appendix pages
July, 2014
Major: Automation science and industrial electronics
Examiner: Professor Pasi J. Kallio
Supervisor: Mathias Von Essen
Keywords: BeagleBone Black, interface, communication, data acquisition system, Inter-Integrated Circuit, analogue to digital covnersion

A development platform is an standalone electronic system, suited with open hardware and software, governed by a microprocessor or a microcontroller. The flexibility and adaptability encourages the spread of these systems in a wide range of applications, by different level designers. The platform is aimed to provide the enough software and hardware features for these projects. Though, additional circuitry and subsystems can be connected enhancing the capability of the electronic prototyping platform.

The core objective of this thesis work is the design and implementation of a communication interface over a BeagleBone Black (BBB) development platform, named as BeagleBone Black Communication Interface (CI-BBB). The scope of the resulted device, is the data acquisition system (DAQ) of the Microrobotic Platform (MP) of FIBAM project, developed by the Micro- and Nanosystems Research Group, of Tampere University of Technology. For that purpose the work is divided in two main areas. In first place, the necessary electronic circuits for signal adapting, which is built on a printed circuit board (PCB) and connected to the BBB. And the control software, which includes a function library and a remote control program. The final device results in an standalone system that comprises the hardware and software in a small area.

Performance measurements are made to characterize the features of the CI-BBB and the integration within the MP. The platform presents the capacity to acquire data, from analogue sources and I^{2}C bus communication system, and provide the fetched samples in a file. The remote control provides a communication line with the system, ensuring the integration of the CI-BBB in the MP and avoiding the introduction of additional elements in the overall system.

PREFACE

The present thesis has been handled at the Micro- and Nanosystems Research Group, at the Department of Automation Science and Engineering of Tampere University of Technology.

In first place, I would like to express my gratitude to Professor Pasi Kallio, for giving me the opportunity to work on this thesis within the Microrobotic Platform project, and learn how real engineering projects are carried out. To my supervisor, Mathias von Essen, for all the invaluable help, support and patience when programming the BeagleBone Black and designing the circuits. To Pooya Saketi, for lending me the tracker NSE 5310 to test the I^{2}C program. To Kourosh Latifi, for assisting me make the tests with the microforce sensor. To Jarmo Verho, for his advices and guidelines to design the printed circuit board of the cape. Furthermore, thank you to my director of the thesis at my home university in Spain Jorge Portilla, for his help and advices despite the distance.

To my beloved family, with which I have an eternal debt of gratitude. I would have never reach this far without them. Their support and love, encouraged and will inspire me through all my life. Muchas gracias de todo corazón, sin vuestro apoyo y amor este día nunca habría sido posible.

Finally I want to thank all my friends, the family we chose to life with, formed by the special people we meet along the path. Thank you to the ones I left at home, with which I discover the life beyond textbooks and learned what is not taught in engineering schools. And thank you, to all the people I meet during my stay in Finland. They filled my days with joy, great experiences and memories I could hardly forget.

Javier Pérez de Frutos
Tampere, 21st July 2014

TABLE OF CONTENTS

1. Introduction . . . 1
   1.1 Motivation . . . 3
   1.2 Objective and scope . . . 3
   1.3 Outline . . . 4
2. Principles of Data Acquisition Systems . . . 5
   2.1 Definition of Data Acquisition Systems . . . 5
   2.2 Elements of the voltage adapter stage . . . 6
      2.2.1 Voltage divider . . . 7
      2.2.2 Operational Amplifier . . . 8
   2.3 Theoretical framework for the communication channels . . . 19
      2.3.1 Analogue to digital conversion . . . 19
      2.3.2 Inter-Integrated Circuit communication protocol . . . 23
3. BeagleBone Black and Scope of the thesis . . . 26
   3.1 Microrobotic platform . . . 26
      3.1.1 Force sensor FT-S10000 . . . 27
      3.1.2 Tracker NSE-5310 . . . 29
   3.2 BeagleBone Black . . . 30
      3.2.1 Hardware description . . . 30
      3.2.2 Software support . . . 39
4. Communication Interface . . . 43
   4.1 Voltage Attenuator circuit . . . 44
      4.1.1 Voltage Attenuator based on voltage divider circuit . . . 44
      4.1.2 Voltage Attenuator based on operational amplifiers . . . 45
      4.1.3 Measurements and circuit selection . . . 47
      4.1.4 Components of the Voltage Attenuator prototypes . . . 50
   4.2 The cape implemented for the BeagleBone Black . . . 52
   4.3 Software: library and remote control . . . 55
      4.3.1 The Communication Interface Library . . . 56
      4.3.2 Remote Control software . . . 61
   4.4 Performance measurements . . . 62
      4.4.1 A/D converter analysis . . . 62
      4.4.2 Long-term test . . . 64
      4.4.3 Final implementation of the system . . . 65
5. Conclusions and future work . . . 67
References . . . 70

Appendices **74**
Appendix A: Schematic of the cape for the BeagleBone Black in EAGLE 6.6.0 74
Appendix B: Flowchart diagrams 76
A/D converter function flowchart - Final version 76
I^{2}C function flowchart 77
Remote control program flowchart 78
Appendix C: User Manual for the Remote Control of CI-BBB 81
Start up the CI-BBB 81
A/D conversion of analogue signals 82
I^{2}C communication bus 84
How to find the I^{2}C device address 85

LIST OF FIGURES

1.1 Concept of the CI-BBB 4

2.1 Block diagram of a DAQ system 5
2.2 Operation of a potentiometer 7
2.3 Connection of energy sources to an OA 9
2.4 Block diagram of an Operational Amplifier 9
2.5 Internal structure of the general-purpose OA μA741 [19] 10
2.6 Schematic of a current mirror 11
2.7 Schematic of a differential amplifier implemented in an OA 11
2.8 Schematic of a common collector amplifier implemented in an OA . . 12
2.9 Schematic of the push-pull AB of the output stage of an OA 13
2.10 Inverting amplifier 15
2.11 Non-inverting amplifier 16
2.12 Voltage follower 16
2.13 Inverting adder 17
2.14 Ideal integrator 17
2.15 Ideal differentiator 18
2.16 Instrumentation amplifier 18
2.17 Relation between the input and the output signals of a A/D converter 20
2.18 Quantification and gain errors of a A/D converter 21
2.19 Successive Approximation Register A/D converter block diagram [4; 24] 23
2.20 Successive Approximation Register search process [4; 24] 23
2.21 Basic structure of a I^2C bus communication system 24
2.22 Data sequence of a basic I^2C communication process 25

3.1 Microrobotic platform of the FIBAM project: (a) Old version, (b) New version 27
3.2 Detail of the force sensor [43] 28
3.3 Force sensor FT-S10000 of FemtoTools [43] 28
3.4 3D model of the Tracker NSE-5310 [41] 29
3.5 Tracker NSE-5310 29
3.6 Block diagram of the microprocessor with the most relevant OCP . . 30
3.7 BBB. Revision A5C 31
3.8 Block diagram of the TSC_ADC subsystem [22] 32
3.9 Block diagram of the PRU-ICSS [21] 34
3.10 Block diagram of the PRU-ICSS [21] 35
3.11 Block diagrams: (a) TPCC, (b) TPTC. 36

3.12 A-Synchronization [21] 36
3.13 AB-Synchronization [21] 37
3.14 Integration of the I^{2}C bus [21] 38
3.15 Examples of capes: (a) WL1835 W/ Chip Antenna, (b) LCD7 cape [11]. 39

4.1 Block diagram of the CI-BBB 43
4.2 Voltage divider prototype circuit based on a potentiometer 44
4.3 Simulations of the PSPICE model of the voltage divider circuit 45
4.4 Effect of the output diode over the load voltage waveform 46
4.5 Circuit of the VA based on OA 46
4.6 Simulations of the PSPICE model of the operational amplifier 47
4.7 Measurement equipment: (a) oscilloscope, (b) function generator, (c) power source. 47
4.8 Prototype circuits of the VA: (a) potentiometer, (b) operational amplifiers approach. 48
4.9 Measurements of the potentiometer circuit 48
4.10 Measurements of the OA circuit 49
4.11 FFT of the input and output signals of the VA prototypes. The FFT of the input sine waveform is depicted in blue color and the output voltage signal in red. The upper plot corresponds to the potentiometer prototype, and the lower to the OA prototype circuit. 49
4.12 Power gain of the proposed voltage attenuators against frequency . . 52
4.13 Schematic circuits of the CI-BBB-cape: (a) VA, (b) buttons, (c) status LED 53
4.14 PCB design of the cape in EAGLE 6.6.0 54
4.15 PCB of the cape: (a) upper layer, (b) lower layer 54
4.16 CI-BBB-cape: (a) upper layer, (b) lower layer 55
4.17 Architecture of the library 56
4.18 Sampled sinusoidal waveform of 1 kHz using buffers 58
4.19 Sampled sinusoidal waveform of 1 kHz without using buffers 58
4.20 TCP socket architecture 61
4.21 FFT of the input signal (red) and the sampled waveform by the A/D converter (blue) 63
4.22 Sampled sine waveform of 100 Hz 64
4.23 Test with the force sensor 65
4.24 Test with the magnetic encoder 65
4.25 CI-BBB device 66

A.1 Schematic of the circuit of the cape designed for the BeagleBone Black in EAGLE 6.6.0 74
A.2 Top layer of the PCB in EAGLE 6.6.0 75
A.3 Bottom layer of the PCB in EAGLE 6.6.0 75
B.1 Flowchart of the A/D converter read function for TXT file 76
B.2 Flowchart of the A/D converter read function for CSV file 76
B.3 Flowchart of the I^{2}C function 77
B.4 Flowchart of the client program 78
B.5 Flowchart of the server program 79
B.6 Flowchart of the *executeCI()* function 80
C.1 Hardware Selection screen 81
C.2 Backup configuration of the A/D converter 82
C.3 A/D converter new parameters screen 83
C.4 New measurement screen 83
C.5 Sample of backup values for the I^{2}C bus 84
C.6 Execution of the commands `i2cdetect -l` and `i2cdetect -r 1` . . . 85

LIST OF TABLES

4.1 Required values for the parameters of the OA for the VA prototype . 50
4.2 Selection of OA available in *Farnell*, for the prototype of the VA [39] [17] [20] [19] [33] 51
4.3 Components of the VA prototype circuit based on OA 51
4.4 Components of the VA prototype circuit based on a potentiometer . . 51
4.5 Components of the cape for the BBB 53

SYMBOLS AND ABBREVIATIONS

Symbols

β	Ratio between the collector and base current of a Bipolar junction transistor
ϵ_0	Vacuum permittivity
ϵ_0	Relative permittivity
A	Area
d	Distance
$f_{ClosedLoop}$	Cutoff frequency in closed loop of an Operational Amplifier
$f_{(G_{vClosedLoop}=1)}$	Frequency at which the Operational Amplifier has unitary gain
G_d	Differential voltage gain of an operational amplifier
G_cm	Common mode voltage gain of an operational amplifier
G_v^{OL}	Open loop voltage gain of an operational amplifier
G_v^{CL}	Closed loop voltage gain of an operational amplifier
G_{sensor}	Gain of the microforce sensor
I	Current
I_c	Collector current
I_{in}, i_{in}	Input current
I_{in-}	Current through the negative terminal of an operational amplifier
I_{in+}	Current through the positive terminal of an operational amplifier
$I_{in(bias)}$	Bias current. Average of the currents applied to the terminals of an operational amplifier
$I_{in(offset)}$	Offset current. Ratio of the mismatch between the currents applied to the terminals of an operational amplifier
$in-$	Negative input terminal
$in+$	Positive input terminal
out	Output terminal

t	Time
t_0	Initial time
V	Voltage
Vcc	Voltage level of DC voltage source
V_{error1}	Voltage error due to the bias current on operational amplifiers
V_{error2}	Voltage error due to the offset current on operational amplifiers
V_{error3}	Voltage error due to the offset voltage on operational amplifiers
V_{in}, v_{in}	Input voltage
V_{in-}	Voltage applied to the negative terminal of an operational amplifier
V_{in+}	Voltage applied to the positive terminal of an operational amplifier
V_{out}, v_{out}	Output voltage
$V_{out,0}$	Output voltage in zero load condition

Abbreviations

A/D	Analogue to Digital
AFE	Analogue Front End
AIDX	Number of bytes of an array of data
AIN	Analogue Input pin
API	Application Programming Interface
ARM	Advanced RISC Machine
BBB	BeagleBone Black
CI-BBB	BBB Communication Interface system
CI-BBB-cape	CI-BBB cape
BIDX	Number of arrays of AIDX bytes wide of data
C	Capacitor

CIDX	Number of frames of BIDX number of arrays
CMRR	Common Mode Rejection Ratio
CSV	Comma separated values file format
D/A	Digital to Analogue
DAQ	Data Acquisition System
DMA	Direct memory Access channels
EDMA3	Enhanced Direct memory Access 3
eMMC	Multimedia Card
ESR	Event Set Register of the EDMA
FFT	Fast Fourier Transform
FIFO	First In First Out memory
GPIO	General Purpose Input/Output
GUI	Graphic User Interface
I^2C	Inter-Integrated Circuit
IDX	Index
INTC	Interrupt Controller
IP	Internet Protocol
LED	Lighting-emitting diode
LibCI_BBB	Library of the Communication Interface for the BeagleBone Black
LSB	Least Significant Bit
MAC	Multiplier with optional accumulator
MP	Microrobotic Platform
MPU	Microprocessor Unit
NoC	Network on Chip
OA	Operational amplifier

OCP	On Chip Peripherals
OS	Operating System
P	Potentiometer
PaRAM	Parameter RAM
PCB	Printed Circuit Board
PID	Proportional Integral Derivative controller
PRU-ICSS	Programmable Realtime Unit and Industrial Communication Subsystem
Q_n	Transistor
QDMA	Quick Direct Memory Access
R_B	Base resistor
R_C	Collector resistor
R_{loop}	Resistor of the feedback loop
R_{in}	Resistor connected to the input terminal
R_{in-}	Resistor connected to the negative terminal of an OA
R_{in+}	Resistor connected to the positive terminal of an OA
R_n	Resistor
$R_{pull-down}$	Pull-down resistor
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
rms	Root Mean Square
Reg_n	Register number n
ROM	Read Only Memory
SCTP	Stream Control Transmission Protocol
SD	Secure Data
SoC	System on Chip

SPAD	Scratchpad memory
SSH	Secure Shell
TCP	Transmission Communication Protocol
TSC_ADC	Touchscreen Controller and Analogue to Digital Converter
TXT	Text file format
T.I.	Texas Instruments
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VA	Voltage Adapter
Z	High impedance logic state

1. INTRODUCTION

There is no doubt that the versatility, low-cost and ease to be used, have granted the prototyping platforms a privileged position in the world of electronic design. Conceived to be suitable for a broad range of different applications, these systems are equipped with a large variety of hardware devices. Nevertheless, this purpose to be general purpose, derives in a lack of functionality when more specific and complex applications arise. Consequently, manufacturers design these electronic systems so additional hardware can be connected to them, enhancing their potential and meeting the requirements for the task. The most well-known electronic platforms are Arduino, pioneer of this market sector, Rapsberry Pi, one of the first single-board computers, and BeagleBone, one of the most powerful electronic protoryping platforms. Though the differences in terms of software and hardware, these platforms have in common certain aspects as being low-priced, open-hardware and open-software, community supported and accessible to any technical designer or fan of the electronics and computers.[11; 2; 12]

An electronic prototyping platform is a normally small, standalone electronic system. Composed of a control unit, hardware devices e.g., Analogue to Digital (A/D) converter or General Purpose Input/Output pins (GPIO); and software, to program and control the system. As said before, these electronic platforms are cost-effective, open-hardware and open-software. This means, the hardware schematics and the software source files are available to be studied and modified by the users.

The control unit is typically implemented with a microprocessor or a microcontroller. Depending on which is used, the processing capacity of the platform is limited, and so its suitability for certain applications. While Raspberry Pi and BBB use a microprocessor, Arduino boards are run with a microcontrollers developed by Atmel. This allows the user to configure and control, in a very precise way, the flow of the routines, and so the hardware. However, the programming process is restricted to either the software provided by the manufacturer or the assembly syntax of the controller. [2]

Furthermore, the use of microprocessors leads to more powerful platforms, with capacity for carrying out large computing-load tasks. But, in order to exploit all the potential of the microprocessor-based electronic platforms the user should be computer-literate, compared to the platforms built with microcontrollers. Raspberry Pi and BeagleBone, specially its newest version BeagleBone Black (BBB), are the best known electronic prototyping platforms based on a microprocessor. The main difference between these two electronic platforms is the micrprocessor. While the Raspberry Pi is build with an ARM1176JZS micrprocessor, the BBB features a more powerful Sitara AM3359. [15; 16]

The Sitara AM3359 of Texas Instruments is a microprocessor that contains a 32-bits ARMv7 Cortex-A8 microprocessor core. Along with this, the chip contains additional hardware that supports the main processor e.g., a Programmable Real Time Unit (PRU-ICSS), an 12-bits A/D converter or an Universal Asynchronous Receiver Transmitter (UART). Regarding the software features of the platform, it is shipped with an Ångström Operating System (OS) Linux distribution, designed for embedded systems. Hence, the BBB is a portable mini-computer with additional services provided by the supplementary hardware. Regarding the programming possibilities of the BBB, the user is able to use any language as long as it has a compiler and an assembler adapted for the ARM architecture. This aspect improves the customization capabilities of the platform compared to microcontroller-based electronic prototyping platforms. Additionally, due to the use of Linux distributions, the OS can be modified by the user to fit its expectations. Also the support web page, provides additional OS to load to the BBB, such as Android or Ubuntu. [22; 7]

Actually, there is an upward trend in the use of electronic prototyping platforms, among designers of electronic and computer systems. The On Chip Peripherals (OCP) and the software that support the system, results in a powerful tool to develop and build a wide range of applications. From a radio control vehicle to a multimedia center, throughout robotic applications, the electronic protorypings platforms can be adapted to meet the requirements of the application. Nonetheless, the great versatility combined with a cheap product, makes the portable computers an alternative to be considered in larger engineering projects, even more during the prototyping phase, but not restricted to this. Hence, the present thesis work aims to demonstrate this capability, by implementing a communication interface for the DAQ of the MP, scope of the FIBAM project, developed by the Micro- and Nanosystems Research Group at Tampere University of Technology, in a single BeagleBone Black.

1.1 Motivation

The aim is to support the acquisition and treatment process of the signals generated by the sensors of the autonomous measurement platform for fibrous materials, scope of the FIBAM project of the Micro- and Nanosystems Research Group. The communication interface developed in this thesis work, enhances the functionality and performance of the overall system, by taking care of the outputs of the sensors of the platform, and providing human-readable files based on these inputs. The final device needs to support different types of input signals, and thus specific hardware and software is required i.e., analogue signals should be discretised through a A/D converter to be analyzed by a computer. Also, control means should be provided, so the measurement procedure can be customized and adapted to the expectations of the user. To ensure the portability of the system, it should be build on a single standalone system, which comprises the necessary hardware and software.

According to these requirements, the BeagleBone Black seems to be a suitable solution as support for the communication interface due to the high level of customization and its low price, avoiding high investments during the prototyping phases of the project.

1.2 Objective and scope

The final goal of the present thesis project is to design and implement a system that brings together the necessary hardware and software to support data acquisition and signal processing of the outcome of different sensors, on a BBB low-cost development platform. The resulting device is named BeagleBone Black Communication Interface (CI-BBB). Figure 1.1 illustrates the concept of the objective system of the present thesis work. To reach the final communication platform, the thesis work is divided is two main parts:

- **The signal conditioning stage:** provide the necessary circuitry to process the output signals of the sensors, so they meet the requirements imposed by the BBB to process the data e.g., isolation, filtering or amplification/attenuation of the signal [4].

- **The control software:** produce a software which controls the hardware of the platform, and serves as interface for the user. In addition, it generates understandable files based on the information provided by the sensors.

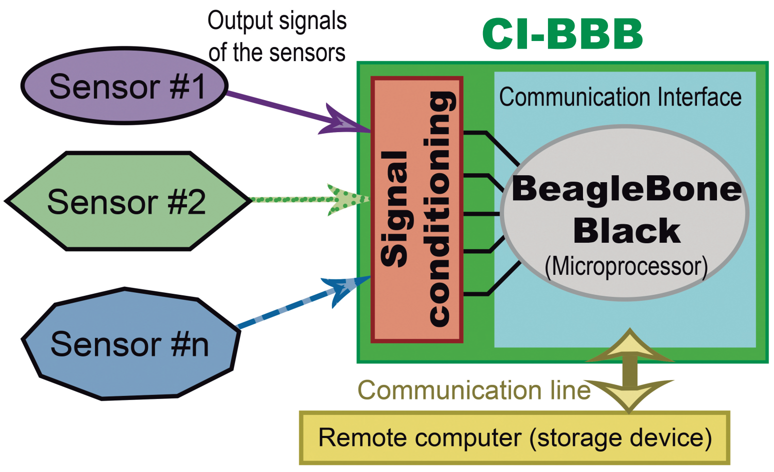


Figure 1.1: Concept of the CI-BBB

1.3 Outline

This thesis is organized as follows:

Chapter 1 gives a brief introduction to the topic of the thesis and the motivation behind this work.

Chapter 2 provides theoretical background related to concepts of DAQ, communication interfaces and signal processing, focusing on the most relevant aspects concerning the present work.

Chapter 3 introduces the BBB development platform and the aim of this component within the scope of the MP.

Chapter 4 describes the physical implementation of the platform as well as gives a description about who this works. Discuss the software development and the features is provides to the system.

Chapter 5 summary, conclusions and discussion of future work to improve the features of the CI-BBB device.

Appendix A depicts the Printed Circuit Board (PCB) schematics of the cape designed for the BBB (CI-BBB-cape) 4.2.

Appendix B includes the flow charts of the functions implemented in the software of the interface.

Appendix C contains an user manual for the Remote Control software of the CI-BBB platform.

2. PRINCIPLES OF DATA ACQUISITION SYSTEMS

The reality we see and in which we live is a continuous time-space existence, and thus unsuitable for being directly analyzed by computer based systems. This need of measuring and manipulating the world can be fulfilled by the wide range of transducers, commonly known as sensors and actuators [36; 26], fundamental part of the DAQ systems, that provides the necessary hardware and software support for the correct measurement of the desired phenomena.

In this Chapter, the reader will be introduced to the principles of these multidisciplinary systems, leading to a further description of the theoretical background concerning in the present thesis.

2.1 Definition of Data Acquisition Systems

These devices are designed to measure the reality and convert it into electric signals comprehensible for computers and analysis software, or capture digital inputs to be transferred to the host computer. DAQ comprises different types of hardware devices and software to implement the demanded features, ranging from A/D converters, Digital to Analogue (D/A) converters, other digital communication protocol interfaces, wiring, transducers (sensors and actuators), etc.

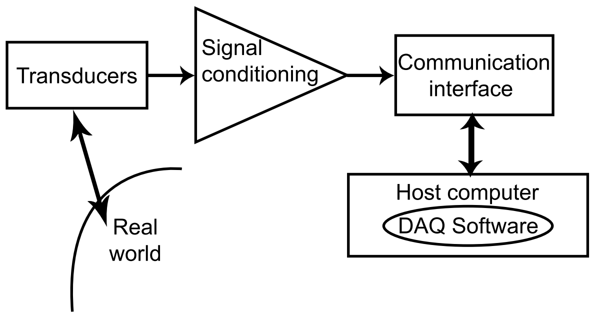


Figure 2.1: Block diagram of a DAQ system

In Figure 2.1 a block diagram of a basic DAQ system is depicted where the following blocks can be identified [26; 4]:

- **Transducers:** devices capable of transform energy to different domains such as electrical to mechanical, or chemical to electrical [36]. This interaction with the real world can be manifested as an action applied towards the phenomena, or as a response of the physical system measured by the sensors.

- **Signal conditioning:** due to the design of the internal structure of the transducers, the output signal is not suitable to be supplied directly to the host computer e.g., because of the content of harmonics or the voltage level of these. So the signal should be adapted to the input requirements of the next stage, to ensure the phenomena is correctly analyzed. Filtering and amplification are two of the tasks carried out by the signal conditioning stage, to which must be added linearization, isolation of the next stage from the source, and excitation of transducers to work.

- **Communication interface:** main component of the system. Serves as interface between the computer or digital control system, and the transducers of the DAQ system. Comprises communication modules and necessary hardware to interact with either the sensors and actuators, and the control system. The A/D and D/A converters play a major role in this stage, because they allow the communication between the analogue and digital components of the DAQ system.

- **Host computer:** controls and interacts with the platform by means of the control software installed in it.

- **DAQ software:** program within the host computer that allows the user to interact with the sensors and actuators available in the DAQ system. Control the performance of the system, calculates the control signals of the actuators and analyses the information provided by the sensor.

Attending to these definitions, the scope of this thesis project covers the signal conditioning stage, the communication interface, the host computer and the control software (see Section 4.3). While signal conditioning stage needs to be implemented with an additional circuit, the communication interface and the host computer are already integrated in the BBB. Finally the control software is programmed and installed in the microprocessor of the BBB.

2.2 Elements of the voltage adapter stage

As said before, the objective of the VA is to handle the signals provided by the sensors and process them, so they meet the requirements of the communication interface input terminals. Two different approaches are proposed for the implementation of the circuit. The first option is based on the use of a voltage divider to establish a proportional relation between the input and output voltage, implemented by fixed value registers. The alternative is a circuit build with operational amplifiers, exploiting the features of the inverting amplifier topology explained in Section 2.2.2.

Thus, this Section aims to describe the theoretical basis for developing these two VA circuits.

2.2.1 Voltage divider

A resistor is a passive element used to limit the current flow through the branch in which it is connected, and so reduce the voltage level. The relation between the current flow and the voltage drop between the terminals of a resistor, can be explained by the Law of Ohm presented in Equation 2.1. [32]

$$V = I \cdot R \tag{2.1}$$

where V represents the voltage drop between the terminals of the device, I is the current that flows through the resistor and R is the value of the resistance of the component. Figure 2.2 illustrates the circuit of a voltage divider.

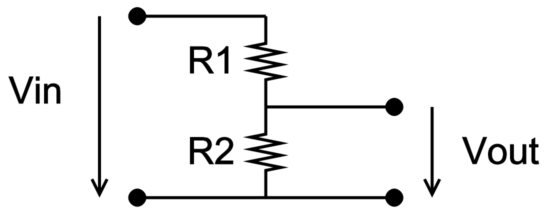


Figure 2.2: Operation of a potentiometer

The output and input voltages can be related by Equation 2.3, which can be derived from the Second Law of Kirchhoff or Voltage Law of Kirchhoff (see Equation 2.2) to the previous circuit [32].

$$\sum_{k=1}^{N} \vec{V_k} = 0 \tag{2.2}$$

Equation 2.2 establishes that, within a closed network, the sum of all voltages results to be zero, based on the conservation of energy within closed systems [32]. The vector notation remarks the fact that the magnitudes might be complex numbers, when using Alternate Current.

$$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2} \tag{2.3}$$

A potentiometer is used for the prototype of this circuit. This device has two electric terminals and an internal moving contact that slides over the surface of a resistive element. This contact is connected to the output terminal, being the length of the resistive element between the input point and this contact defined as the resistor named R_1 in Figure 2.2. These devices can be found in linear and rotatory configuration, typically used to define a reference voltage i.e., as a position sensor using the output voltage as a measure of the linear or angular displacement of the sliding contact attached to the moving system. [4]

2.2.2 Operational Amplifier

An Operational Amplifier (OA) is considered the basic unit of the signal conditioning systems [4]. The standard design presents two input ports in phase opposition and a unique asymmetric output port. The inverting and non-inverting inputs form a differential input, and the connection made between these produces modify the behavior of the OA [34]. Additionally, the OA features two extra pins to make corrections on the output voltage level, apart from the positive and negative voltage supply terminals. [4]

Overview of the Operational Amplifier

From a general point of view, the relation between the input voltages and the output voltage of an OA can be described as in equation 2.4.

$$v_{out} = G_d \cdot (v_{in+} - v_{in-}) + G_{cm} \cdot v_{cm} \tag{2.4}$$

where G_D represents the differential gain and G_{CM} is the common mode gain. Ideally, the OA works like a differential amplifier of infinite G_D and no common mode signal amplification. But real OA usually has a G_D of 100000 to 1000000, in open loop configuration, and the common mode signals, denoted as v_{cm}, must be taken into consideration. These signals include noise and interferences which are applied to both inputs of the OA, and thus, amplified at the output terminal. The capability of the device to avoid this propagation is called Common Mode Rejection Ratio (CMRR) (see Equation 2.5). This parameter determines how much the common mode signals are attenuated, compared to the amplification of the differential input signal. The larger the CMRR is the smaller the common mode component of the output signal. [32]

$$CMRR = \frac{|G_D|}{|G_{CM}|} \tag{2.5}$$

The energy source must be symmetric i.e., two identical voltage sources connected in series, with the ground connection common to the negative terminal of the first source and the positive port of the second (see Figure 2.3).

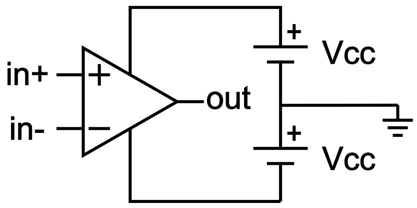


Figure 2.3: Connection of energy sources to an OA

An important characteristic of the OA which should be taken in consideration when selecting components is the *slew rate*, defined as the maximum variation of output voltage against time, when varying the input signal. A low slew rate value results in the inability of the output signal to follow the changes of the input signal, and so the input waveform is distorted. The slew rate determines the bandwidth of the OA, setting the maximum frequency at which the device can operate without distorting the input signal waveform [32]. In case the input signal has small amplitude and frequency values, the amplification will not be affected by the slew rate. However, with high output voltages and high frequencies the slew rate has to be taken in consideration [32].

Internal structure of an Operational Amplifier

The internal structure of the device is composed of three main stages: an amplifier phase followed by an adapter circuit of direct voltage level, and an output amplifier [34]. However, the design can vary depending on the manufacturer. Regardless of the internal design, all OA share the same external layout: the differential input and the asymmetric output [32]. This structure is depicted in Figure 2.4.

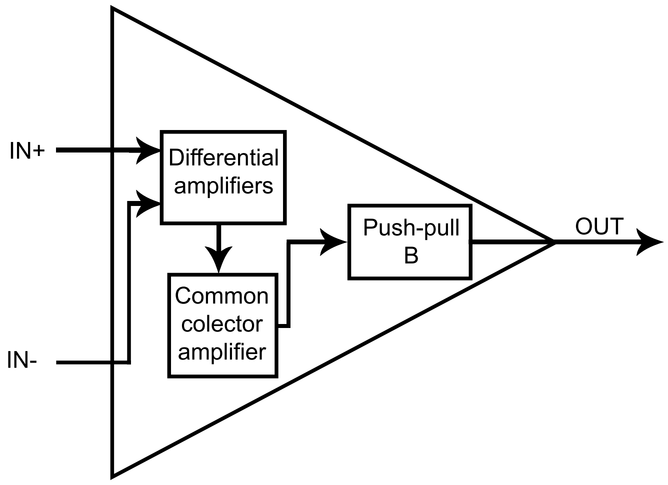
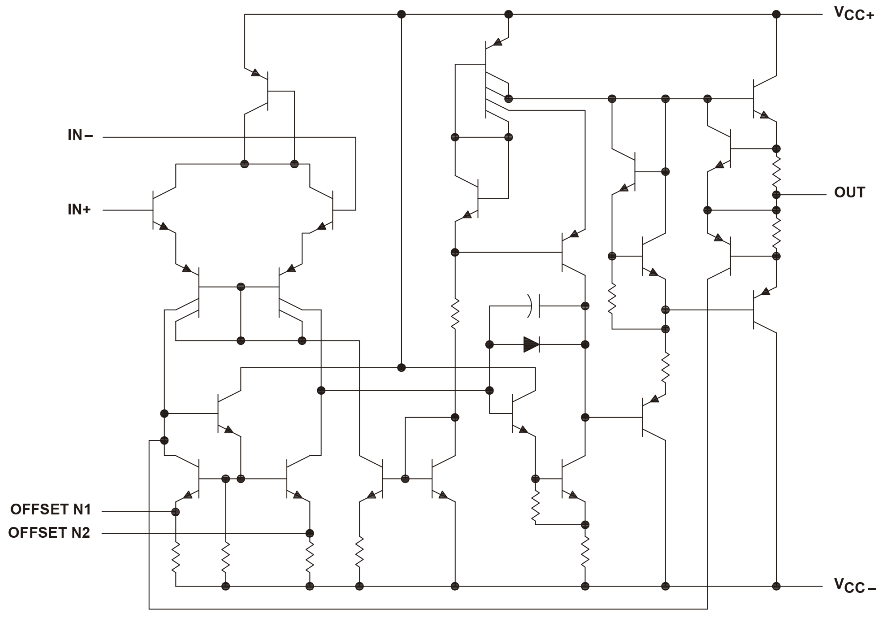


Figure 2.4: Block diagram of an Operational Amplifier

As shown in Figure 2.4, there are three main blocks to consider within the OA. An amplification stage comprised of two differential amplifiers, a common collector amplifier stage and a push-pull of type B as output step [32; 34]. Though it might look simple, real OA circuits are quite complicated as it can be seen in Figure 2.5, where it can be seen the internal structure of the general-purpose OA µA741 designed by Texas Instruments.



Component Count	
Transistors	22
Resistors	11
Diode	1
Capacitor	1

Figure 2.5: Internal structure of the general-purpose OA μA741 [19]

Yet an additional component not depicted in Figure 2.4, completes the internal circuitry of the OA. This circuit is known as current mirror, whose topology is depicted in Figure 2.6. The current mirror behaves as a current source, providing a constant current at the collector terminal of the transistor (I_c), which is directly related to the current that flows through opposite branch (I) i.e., is equal to the current that flows through the resistor and diode depicted in Figure 2.6 [1]. Current mirrors are normally used in integrated circuits as active loads on differential amplifiers, fixing the emitter current to a constant value and increasing the gain of the circuit; and as tail resistors connected to the emitter of the transistors of the differential amplifier circuits, to improve the Common Mode Rejection Ratio (CMRR) of the amplifier.[32]

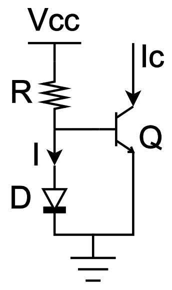


Figure 2.6: Schematic of a current mirror

Large resistive loads are required in the differential amplifiers, to improve the gain of the circuit., The size of traditional resistors, with this resistance value, are unsuitable to be integrated in the circuit of the OA [32; 34]. Because current mirrors behave as current sources, their output impedance can reach very high values, without increasing the required area of silicon. Figure 2.7 shows a general schematic of a differential amplifier with three current mirrors: two of these current mirrors are constituted of transistors Q_4 and Q_5, sharing the diode Q_3 (implemented with a transistor with the base connected to the collector terminal), and the resistor R_1; and the third mirror is formed by transistor Q_6, diode Q_7 and resistor R_2.

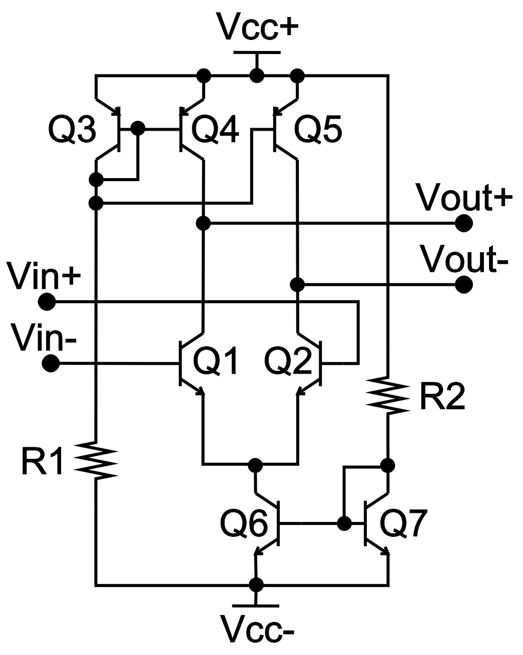


Figure 2.7: Schematic of a differential amplifier implemented in an OA

The input stage of the OA, is composed of two differential amplifiers in cascade configuration. The output is an asymmetric signal i.e., one of the two output terminals is connected to the ground, so the voltage level is referred to ground. The first amplifier, in common emitter configuration, produces a proportional voltage as explained in Equation 2.6, where G_v is the voltage gain of the amplifier and $v_(in+)$ and $v_(in-)$ are the input voltages at the non-inverting and inverting ports, respectively.

Due to the high input impedance provided by the transistors of the differential amplifier stage, the current through the input terminals of the OA can be neglected. In the case that this current has a significant value, the use of the Offset and Null pins can be used to compensate this effect, by connecting a potentiometer to these pins, and the energy rail. [32; 34]

$$v_{out} = G_v(v_{in+} - v_{in-}) \tag{2.6}$$

Unlike the previous circuit, the second differential amplifier is a single-ended amplifier, and does not present Offset Null terminals. The differential output voltage of the first stage is amplified through this circuit, resulting in an asymmetric output signal proportional to this voltage difference. [32; 1]

The next stage, depicted in Figure 2.8, consists of an amplifier circuit in emitter follower or common collector configuration, with a current mirror as load connected to the emitter port [1]. Instead of increasing the voltage level, as in the previous amplifiers, the emitter follower produces an output current related to the input current level i.e., a small increase of the current supplied to the base of the transistor results in a huge increment of the output current at the emitter terminal, with almost negligible variation of the voltage level [32]. Therefore, the power content of the signal is increased in this stage. Finally this stage matches the impedance of the differential amplifier and the output stage of the OA at the same time the bandwidth in open loop is limited [13].

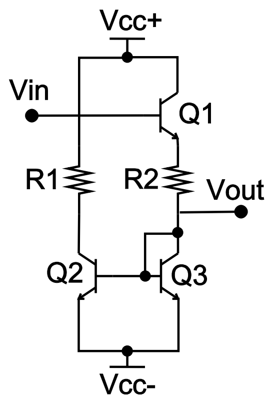


Figure 2.8: Schematic of a common collector amplifier implemented in an OA

The last stage of the OA typically consists of a push-pull of type B, as depicted in Figure 2.9. In an ideal circuit, the upper NPN transistor conducts while the input signal has a positive voltage level, and the lower PNP transistor switches on during the negative period. The transistors must be polarized to opposite operation points, in order to reproduce the complete input waveform. The resulting signal is almost equal to the input voltage waveform, and additionally, because the topology of the circuit, the output impedance of the OA is almost negligible. [32; 13; 34]

Yet, due to the use of real components, the transistors are not polarized at the desired point, so the switching is not simultaneous. Actually there is a small period of time in which both devices are turned on, resulting in a overlapping of the output waveforms of each device known as cross-over distortion. This problem can be magnified because the operation temperature, that can change the polarization point of the transistors (thermal runaway), and the unequal set point of these devices, due to physical differences between them. [32; 34]

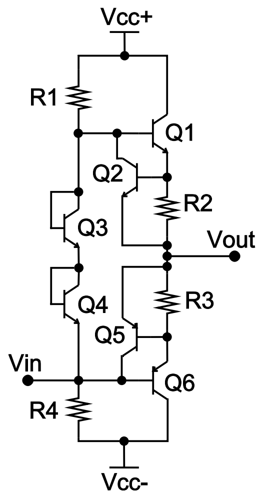


Figure 2.9: Schematic of the push-pull AB of the output stage of an OA

Biasing the bases of the transistors using a voltage divider, can prevent the cross-over distortion by fixing the voltage level between the bases of the transistors, and thus avoiding the polarization point from changing during operation [32]. However this solution does not address the thermal runaway problem. In this case the use of compensating diodes is highly recommended, adjusting the bias emitter voltage whenever the temperature increases. These diodes can be found in Figure 2.9 named as Q_2 and Q_3. Additional protection devices can be found in this last stage of the OA i.e., to prevent from shortcuts.[1; 32; 34; 1]

Non-ideality errors

Since the OA is composed of non-ideal devices, errors can be introduced in the output signal, which might be taken in consideration. The scope of this project considers only frequency signals up to 1 kHz (see Section 3.1.1), thus in the next paragraphs the inaccuracies that may appear when working at such frequencies are presented.

The differential amplifier stage is the main cause of the errors of the OA, due to the non-symmetry of its structure and components i.e., the opposite transistors are not equal, and so there is a mismatch between the current gains (β), leading to a different current level drained by the base of the transistor [32]. This mismatch results in the input bias current (equation 2.7) and input offset current (Equation 2.8) [32; 1]. These currents appear at the inverting and non-inverting inputs, connected to the base of the transistors of the differential amplifier stage (see Figure 2.7).

$$I_{in(bias)} = \frac{I_{in+} + I_{in-}}{2} \tag{2.7}$$

$$I_{in(offset)} = I_{in+} + I_{in-} \tag{2.8}$$

When a resistance is connected to one of the input terminals, due to the base current a voltage difference is produced, and can be amplified by the OA resulting an output voltage, even though no input signal is being deliberately supplied to the device.

Due to the asymmetry of the differential amplifier, an output voltage can be detected even though no signal is supplied to the input terminals. This voltage error is known as offset voltage. [32].

Therefore, the output error signal has three different sources: the bias current, the offset current and the offset voltage [32; 1]. These error sources are expressed in Equations 2.9, 2.10 and 2.11, and can be derived from the previous equations 2.7 and 2.8, and the circuit depicted in Figure 2.7.

$$V_{error1} = (R_{in+} - R_{in-})I_{bias} \tag{2.9}$$

$$V_{error2} = (R_{in+} - R_{in-})\frac{I_{bias}}{2} \tag{2.10}$$

$$V_{error3} = V_{offset} \tag{2.11}$$

$$V_{erroroutput} = G_v^{OL}(V_{error1} + V_{error2} + V_{error3}) \tag{2.12}$$

where R_{in+} and R_{in-} are the base resistors and G_v^{OL} is the open loop voltage gain of the OA. The bias current error can be eliminated just by connecting the same impedance to the input terminals of the OA. Additionally, offset current error are also attenuated. But, if the application requires to eliminate all the DC imperfections, the nullify circuit recommended by the manufacturer in the datasheet of the device, should be implemented. The Offset Null pins are used to compensate the non-symmetry of the differential amplifiers by modifying the tail resistors, typically implemented with current mirrors. An alternative approach is to apply an input voltage which opposes to the voltage error, output and nullifies it.

Typical application circuits

Some of the most typical topologies in which the operational amplifier is the main component will be presented in this Section.

- **Inverting amplifier:** in this circuit, the output voltage is proportional to the input but the polarity is changed i.e., a positive input signal will result in a negative output voltage. This circuit uses a negative feedback to stabilize the immense and unstable open loop voltage gain of the OA as depicted in Figure 2.10. The opposite amplified output voltage is driven back to the input port and neutralize any sudden variation of the open loop gain of the OA by decreasing the input voltage, stabilizing the system [32; 4; 1]. Equation 2.13 corresponds to the working principle of this topology.

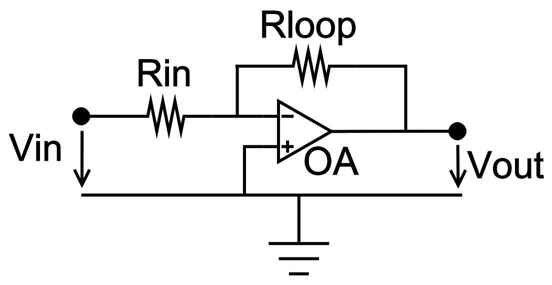


Figure 2.10: Inverting amplifier

$$\frac{V_{out}}{V_{in}} = -\frac{R_{loop}}{R_{in}} \tag{2.13}$$

Not only the voltage gain is improved, also the bandwidth of the OA increases with the decrease of the closed loop gain, according to Equation 2.14 presented in [32; 4].

$$f_{ClosedLoop} = \frac{f_{(G_{vClosedLoop}=1)}}{G_{vClosedLoop}+1} \tag{2.14}$$

- **Non-inverting amplifier:** unlike in the previous circuit, the non-inverting amplifier produces an output with the same polarity as the input signal (see Equation 2.15). This is done by driving the voltage through the non-inverting input pin of the OA as exposed in Figure 2.11. But here the system is also stabilized by means of a negative feedback loop which conducts the output voltage into the inverting port of the amplifier. Any variation of the open loop gain of the OA will be reflected in the output voltage, and due to the feedback the differential input voltage will vary accordingly damping this deviation and stabilizing the circuit.

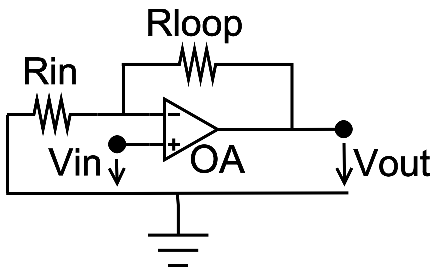


Figure 2.11: Non-inverting amplifier

Additionally the bandwidth of the OA is improved in a similar way as in the previous circuit as shown in Equation 2.16 [32; 4; 1].

$$\frac{V_{out}}{V_{in}} = \frac{R_{loop} + R_{in}}{R_{in}} \tag{2.15}$$

$$f_{ClosedLoop} = \frac{f_{(G_{vClosedLoop}=1)}}{G_{vClosedLoop}} \tag{2.16}$$

- **Voltage follower:** this circuit can be considered as the limit case of the non-inverting amplifier. Assuming a zero feedback loop resistor, all the output voltage is driven back to the input of the OA as explained in Figure 2.12. This results in an output voltage level equal to the input voltage ($G_{vClosedLoop} = 1$). Because the input and output impedance are increased and decreased respectively due to the feedback loop, the voltage follower seems to be the perfect choice to implement a buffer or impedance matching i.e., connect a high/low impedance device to another with low/high impedance. regarding the bandwidth improvement, according to Equation 2.16 this will be maximum cause $f_{ClosedLoop} = f_{(G_{vClosedLoop}=1)}$ [32; 4; 1].

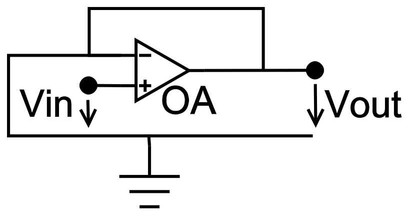


Figure 2.12: Voltage follower

- **Inverting adder:** Figure 2.13 depicts this circuit which can be understood as a generalization of the inverting amplifier described previously, but with several voltage levels at the input port weighted by resistors. As it can be deduceds the main purpose of this topology is to produce an averaged voltage related to the inputs. From Equation 2.13 can be deduced the expression to estimate the output signal as a sum of the contributions of each of the input voltages [32; 4; 1].

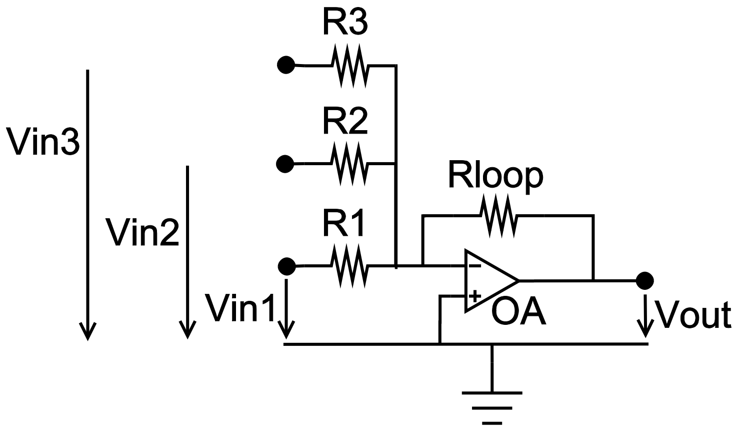


Figure 2.13: Inverting adder

$$V_{out} = -R_{loop} \sum_{k=1}^{N} \frac{V_{in,k}}{R_{in,k}} \tag{2.17}$$

- **Ideal integrator:** using a capacitor in the feedback loop or an inverting amplifier circuit will result in an output voltage proportional to the integral of the input voltage over time. The following expression can be deduced from applying the Current Law of Kirchhoff to the connection point between the input and the feedback and taking into account that the current that flows through a capacitor is proportional to the variation rate of the voltage between its terminals [4; 1].

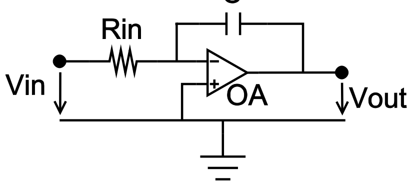


Figure 2.14: Ideal integrator

$$\Delta V_{out} = -\frac{1}{R_{in}C} \int_{\Delta t} V_{int} dt \tag{2.18}$$

- **Ideal differentiator:** if the input resistor in the inverting amplifier is replaced by a capacitor, the resulting output voltage will be proportional to the variation rate of the input signal. Thus the output will be related to the derivative of the input voltage.

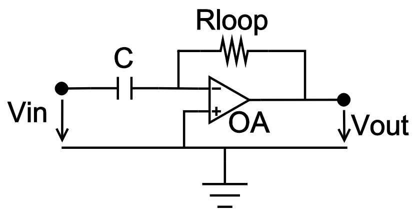


Figure 2.15: Ideal differentiator

$$V_{out} = -R_{input}C\frac{dV_{int}}{dt} \tag{2.19}$$

The combination of this circuit and the integrator can be used to implement a Proportional Integral Derivator controller (PID). Setting the resistors will modify the proportional gain of the controller. The capacitor affect the characteristic time of the proportional and integral components.

- **Instrumentation amplifier:** the circuit depicted in Figure 2.16 displays a differential amplifier tuned to have unity gain, composed by an OA and resistors R_3 and R_4, with a pre-stage formed by two non-inverting amplifiers (Figure 2.11) with the feedback loops connected through resistors of value R_1 [32]. This configuration improves the characteristics of the normal differential amplifier increasing the CMRR, the input impedance and the voltage gain from 1 V/V to 1 kV/V [13], and decreasing the output impedance and the offset voltage.

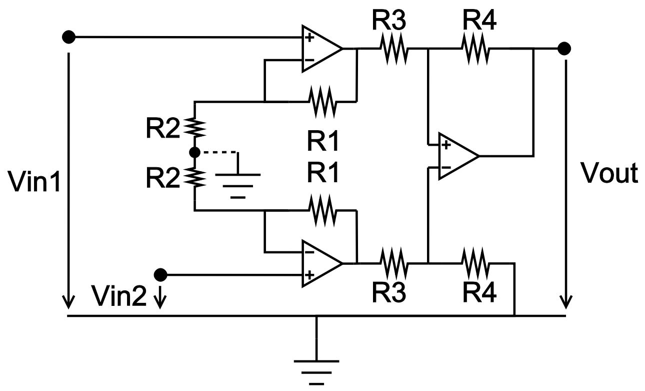


Figure 2.16: Instrumentation amplifier

Due to the shared point by the non-inverting amplifiers, this configuration is capable of amplify the differential input without enhancing the common-mode signal. This can be explained because of the fact that this common point behaves as a floating point for the common-mode signal, and as a virtual ground for the differential input. When the same voltage level is applied to input ports the output of the inverting amplifiers have the same level and hence the feedback loops present the same potentials, making these circuits to behave as voltage followers (Figure 2.12 and thus not amplifying the input signal. Whereas if the input voltages are opposite one from the other the common point will be at zero potential, behaving as a virtual ground and hence the amplifiers will act as non-inverting amplifiers [32; 13].

Equation 2.20 shows the transfer function of this circuit where V_{in1} and V_{in2} are the input voltages to each temrinal of the amplifier, and V_{out} the output voltage level [4; 32; 13].

$$V_{out} = (1 + \frac{2 \cdot R_1}{R_2})(\frac{R_4}{R_3})(V_{in2} - V_{in1}) \tag{2.20}$$

2.3 Theoretical framework for the communication channels

Of all communication channels available in the BBB, this thesis covers the A/D converter and the Inter-Integrated Communication (I^2C) bus. This Section gives a theoretical introduction of these two communication schemes. First the A/D conversion is presented,

2.3.1 Analogue to digital conversion

The sampling of continuous analogue signals into a set of discrete digital values manageable for digital systems, is accomplished by A/D converters. The basic procedure is based on a succession of hold-and-read operations, where a digital code is generated according to the voltage level read at the input port. The evolution of the DAQ systems has lead to an important improvement of the A/D converters, resulting in wide range of different strategies to digitize analogue signals e.g., ramp, dual ramp, successive approximation register or the flash A/D converter. Among them this Section will focus on the Successive Approximation register A/D conversion scheme, algorithm implemented in the BBB A/D converter described in Section 3.2.1.

The maximum resolution of a particular A/D converter is determined by the number of bits of the digital word. Figure 2.17 illustrates the relation between the input and the output values of a 3-bit A/D converter. In this example, the input voltage is sampled in eight (2^3) different levels or quantization steps of 1 Least Significant Bit (LSB) wide. The LSB represents the range of voltage values that are represented by one bit of the digital code, as explained in Equation 2.21. In this Equation, V_{max} represents the maximum input analogue voltage the A/D converter can withstand, and N is the number of bits of the digital word.[4; 30; 47]

$$1LSB = \frac{V_{max}}{2^N} \tag{2.21}$$

The difference between two consecutive digital samples is called quantification level, and it can be seen that due to the nature of the discrete signals the digital output does not change in value even though the input does. This fact leads to quantification errors, defined as the difference between the voltage value and the nearest discrete voltage level. This difference can be as large as $\pm\frac{1}{2}$LSB.[4; 30; 47]

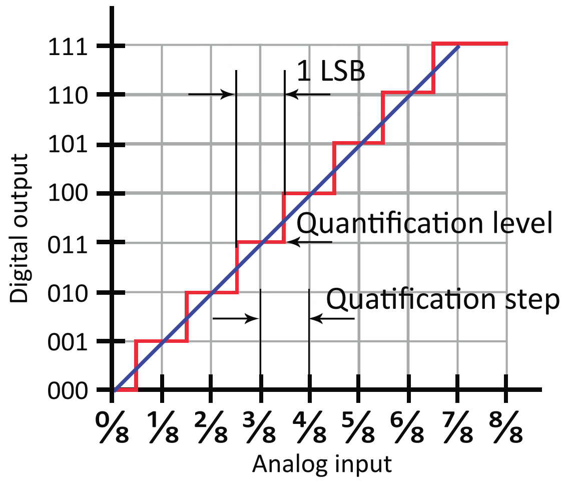


Figure 2.17: Relation between the input and the output signals of a A/D converter

The capability of an A/D converter is measured through a series of parameters that quantify the static and dynamic characteristics of the device in terms of the data conversion. In the following the most important figures of merit are explained and illustrated with the previous example of a 3-bit A/D converter, extending the ones introduced in the previous paragraph.

- **Resolution:** determined by the number of bits N of the digital word. This parameter is directly related to the size of the LSB by Equation 2.21. The word length determines the number of levels of tha analogue signal the converter can differentiate i.e., the quatification step depicted in Figure 2.17.[4; 30; 47]

- **Sampling rate:** the Nyquist-Shannon sampling theorem establishes the minimum sampling frequency to produce a reasonable sampling of a given analogue continuous signal, so it can be reconstructed again from the discrete digital samples using a Digital to Analogue converter. As shown in Equation 2.22, the lower bound for the sampling rate, known as Nyquist rate, is at least twice of the highest frequency of the input signal. Below this point aliasing effect takes place when the reconstructed signal differs from the original.[4; 30; 47]

$$f_s = 2f_{max} \tag{2.22}$$

- **Quantification error:** the difference between a given input voltage and the nearest voltage level, dictated by the digital word length. (see Figure 2.18). This error can be characterized by its root mean square value (rms) assuming an uniform distribution along each digital code and ranged between $\pm\frac{1}{2}$LSB. Using the definition of rms value and equation 2.21, the following equation can be derived.[4; 30; 47]

$$V_{Q,rms} = \sqrt{\frac{1}{T}\int_0^T LSB\left(\frac{x}{T}-\frac{1}{2}\right)^2 dx} = \frac{V_{max}}{2^N}\frac{1}{\sqrt{12}} \tag{2.23}$$

where $V_{Q,rms}$ is the rms voltage value of the quantification error.

- **Gain error:** the equation 2.24 depicted as a ramp in Figure 2.17, represents the transfer function of an ideal A/D converter. Gain error quantifies the deviation of the slope of the ramp from the ideal case, leading to loose of resolution and/or quantification levels as shown in Figure 2.18.[47; 30]

$$TF = \frac{V_{max}}{2^N}\sum_{0}^{N-1} b_k 2^k \tag{2.24}$$

where $b_0, b_1, ..., b_{N-1}$ represents the bits of the digital sample.

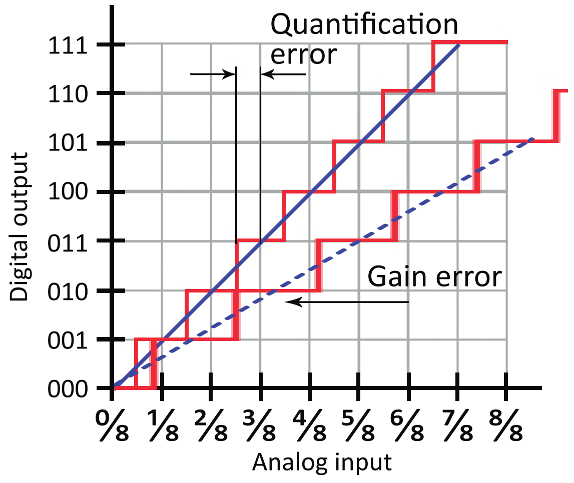


Figure 2.18: Quantification and gain errors of a A/D converter

- **Offset error:** deviation of the digital output signal at zero input voltage. This error is easily removable by subtracting this value from the output.[4; 30]

- **Differential Linearity Error:** difference between the ideal quantification step width (1 LSB) and the real, e.g., the ideal A/D converter present a zero value for this type of inaccuracy.[47; 30]

- **Integral Linearity Error:** if the differential linearity error was referred to the quantification step, the integral is based on the distance between the center point of each quantification level. Depending on the manufacturer this error might be expressed on the basis of the ideal transfer function line or the best fit line that join the center points of each interval.[47; 30]

- **Missing code:** when the A/D converter does not generate a digital code for any voltage level it is said that there is a missing code. This happens when the differential linearity error has a negative value which means a quantification level is being overlapped by the next level, thus the voltage level that was supposed to generate that digital code will result in the immediate next digital word.[47; 30]

- **Signal-to-Noise ratio:** relation between the full scale analogue voltage input and the *rms* quantification error value as expressed in Equation 2.25. Ideally this measure is only affected by the quantification error, however additional errors should be taken in consideration such as thermal noise or reference noise. [47; 30]

$$SN = 20 \log \frac{V_{max,rms}}{V_{Q,rms}} = 6.02N + 1.76(dB) \tag{2.25}$$

The AM3359 microprocessor, used by the BBB electronic prototyping platform, includes a Successive Approximation Register (SAR) A/D converter. In general terms, these SAR A/D converters are characterized by high resolution, accuracy and low power consumption; which makes these type of converters suitable for integrated circuits systems. However, the main drawback is, as it will be seen in the next paragraph, the requirement of additional components which might decrease the performance of the overall system; and reduce the data conversion rate compared to other A/D converters of same resolution such as Pipelined of Sigma-Delta schemes. The block diagram of the SAR A/D converter is depicted in Figure 2.19.[24]

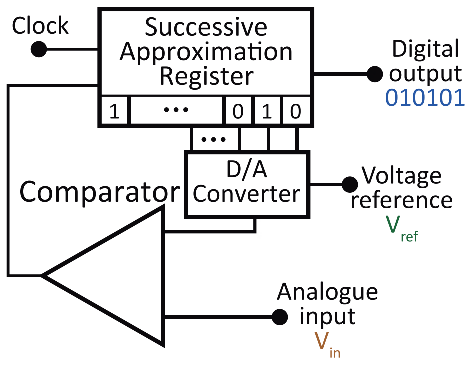


Figure 2.19: Successive Approximation Register A/D converter block diagram [4; 24]

Using a search algorithm, the A/D converter is able to generate an N-bit binary code equivalent to the analog signal held in the input port. In the first iteration, a digital code with the most significant bit (MSB) set to logical one, is converted into an analogue voltage value by a D/A converter. Then it is compared with the input signal, if the input voltage is greater than the converted value, the MSB remains with unchanged, otherwise the MSB is cleared. Then the SAR moves to the next bit and repeats the process. This routine continues till the LSB is determined, resulting in a N-bit code that represents the analogue voltage level. This process is illustrated in Figure 2.20.[4; 24]

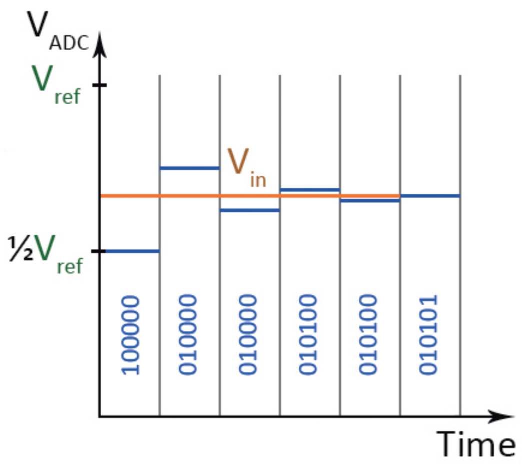


Figure 2.20: Successive Approximation Register search process [4; 24]

2.3.2 Inter-Integrated Circuit communication protocol

The need of sharing information has been a basic need since the appearance of the first computers. Even within a single microcontroller, the data flow between the different system on chip (SoC) devices, such as memories or the microprocessors, has to follow certain structured rules so the information is comprehensible by all the parts involved in the communication. This amount of rules or algorithms defines a communication protocol. Syntax, semantics or timing are some of the aspects stipulated by these rules, which can be implemented by hardware e.g., I^2C; or software e.g., TCP/IP. The portability is one of the most important aspects of the communication components, being independent of the platform where is going to be used so it can be integrated in many different systems. [5]

The Inter-Integrated Circuit (I^2C) is a hardware serial bus interface developed by the company Philips Semiconductors in the early 1980 [42]. This structure connects different devices which can be part of the same microcontroller, resulting in a Network on Chip (NoC); or independent components e.g., a sensor sending information to a computer. Due to the standardization of this protocol, it is easier to connect different systems using I^2C interface.

The bus is composed of two main lines: a bidirectional data channel (SDA) and the clock signal (SCL). Both lines are AND-wired to all and devices, and open drained i.e., it can be set to a logic 0 level or *high-impedance* (Z). This high impedance state means that the logic level of the line, cannot be determined at any time [40]. That is the reason why external pull-up resistors are used to keep a logic high level on these lines, as sketched in Figure 2.21, when no device is using them. [25; 40]

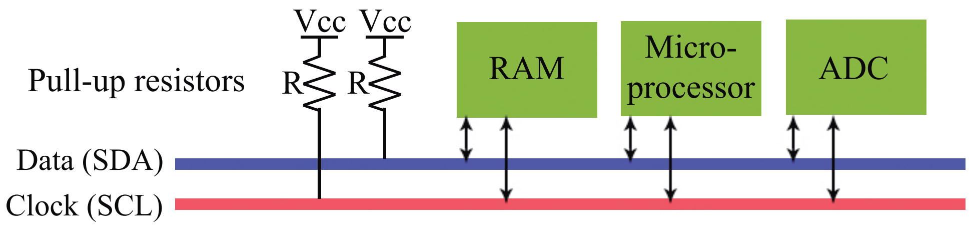


Figure 2.21: Basic structure of a I^2C bus communication system

The devices connected to the bus can play two different roles: *master* or *slave*, and regardless of which of them is chosen, the devices can be data *transmitters* or *receivers*. Though, this last two terms are self-explanatory, the master and slave should be explained in more detail within the context of the I^2C bus. A master device is capable of initiating the communication with any other hardware connected to the bus. Unlike the slave, which waits till a communication request arrives [5; 25].

To send data through the bus, the device must set the corresponding value on the SDA line while the clock signal is low. Because of the possibility of having several master devices at the same time and on the same bus, there is a high risk of having several components requesting the bus for its own communication [42]. Thereby, *synchronization* and *arbitration* are very important features, implemented in the I^{2}C serial bus communication protocol. [5; 40]

- **Synchronization:** describes how the I^{2}C bus deals with the situation in which more that one device is willing to impose its clock signal on the SCL line. Due to the fact that it is open drained, its logic value will be 0 as long as one of the master device sets this value. Whenever this signal is set to zero, the internal counter of each device is being reset, and so all devices are synchronized. Hence, the device with the largest clock period will impose its clock signal to the rest. [5; 42]

- **Arbitration:** attains to the case in which more than one device is sending data through the SDA line. If a master detects that the data line is pulled down when it should be high, at the moment of putting its own data, it assumes another master is controlling the SDA line, so it stops the current communication and waits for the other device to finish. [5; 42]

The I^{2}C can reach data transfer speeds of 100 kHz in the case of Industrial and SMBus, up to 3.4 MHz when configured in *High Speed mode* [25].

The different devices connected to the bus are identified by a 7 to 10 bits digital code (see Figure 2.21) used to identify an specific target for each communication process. This identifier is specified at the beginning of the communication, after the start condition of the communication process. Then data is transferred as soon as the target device has acknowledged the communication request, before the master has specified the type of operation to perform using one control bit: read or write. Finally the communication is terminated with the stop condition (see Figure 2.22).

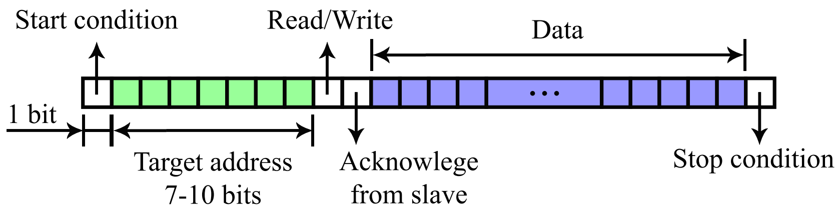


Figure 2.22: Data sequence of a basic I^{2}C communication process

3. BEAGLEBONE BLACK AND SCOPE OF THE THESIS

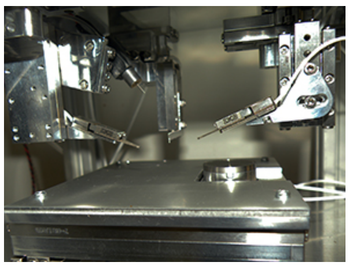
Arduino, Rapsberry Pi or BeagleBone are some of the most well-known electronic development platforms. As said in Chapter 1, the flexibility and wide range of hardware and software included, make these devices suitable to be programmed for a broad range of tasks, avoiding large investments in the early stages of the projects.

This Chapter introduces the BBB development platform used in this thesis work. In the first Section an overview of the MP is included as background, to define the necessities the BBB platform should cover. Later the prototyping platform is explained emphasizing the hardware and features used in the scope of the present work.

3.1 Microrobotic platform

Since September of 2011, FIBAM project accomplished by the Micro- and Nanosystems Rsearch Group of the University of Technology of Tampere, Finland, aims to develop an autonomous system capable of providing statistically significant measurements of mechanical properties of fibrous materials i.e., fiber to fiber bonds or flexibility of individual fibers [27]. For such purpose an automated MP with a stacked gantry crane structure, is build under the supervision of Professor Pasi Kallio. This device is able to treat and manipulate fiber samples to be used for measurements carried out by the same platform, or to be used by other devices. The implementation includes three micromanipulators, two of four degrees of freedom and one of five degrees of freedom which also incorporates the sample dispenser, to which microgrippers are attached to hold the fibers. The hardware is completed with a rotary table, a fiber bank, an XYZ positioner, an XY table and a microforce sensor. [37; 35]

Figure 3.1 shows the two versions of the MP. The first version was presented on 2009, as part of the SMARTFIBER project, done by the Micro- and Nano System Research Group [37]. The new version is being developed within the FIBAM project, by the same research team as the previous MP [27]. In the actual implementation, the microforce sensor is handled by a National Instruments PCI-6229 DAQ board, which supports the A/D conversion of the signal and the communica-


(a)

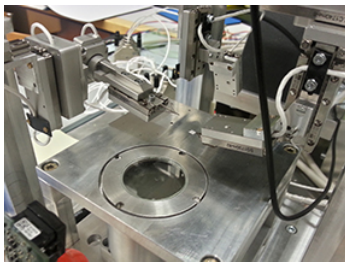

(b)

Figure 3.1: Microrobotic platform of the FIBAM project: (a) Old version, (b) New version

tion between the sensor and the control computer [46]. The magnetic encoder (see Section 3.1.2) is handled directly by the control computer, using the software MATLAB of Mathworks®. CoSMic is the control software designed specifically for the platform, which features three different control modes for the MP: manual control mode, enhanced manual control mode and automatic control mode [46].

In this first version of the CI-BBB, the BBB will manage the output signals provided by the microforce sensor and the magnetic encoder. So, instead of having different devices to cope with each sensor, the CI-BBB will bring together the actual DAQ systems into one platform. This will simplify the management of the sensors, and release the control computer from this task.

3.1.1 Force sensor FT-S10000

The sensor installed in the MP is a capacitive microforce sensing probe manufactured by FemtoTools. The force is measured by means of the voltage difference across a capacitor, which is directly related to the capacitance of the component (see Equation 3.1) [32]. Hence having one of the parallel plates attached to the probe and the second one fixed, the movement of the end effector results in a variation of the capacitance (see Equation 3.2, capacitance of a two parallel plates capacitor),and thus, of the voltage measured to calculate the force [4].

$$V(t) = V(t_0) + C \cdot \int_{t_0}^{t} i(\tau) d\tau \tag{3.1}$$

$$C = \frac{\epsilon_0 \epsilon_r A}{d} \tag{3.2}$$

In real implementations, these sensors are build with several capacitors connected in parallel, improving the linearity and stability of the measures[37], and the sensitivity of the sensor. Figure 3.2 shows the structural configuration of the tip of the force sensor, where the two rows of capacitors can be appreciated at both sides of the probe.

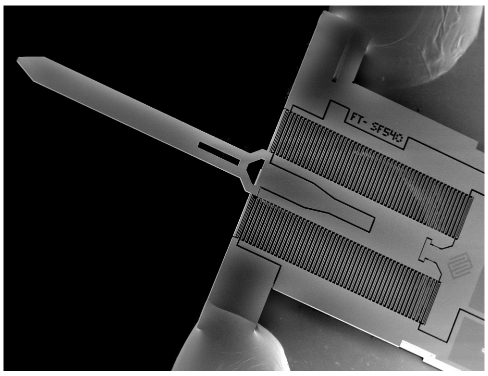


Figure 3.2: Detail of the force sensor [43]

The range of this sensor is 10 000 µN, depending if the probe is under tensile or compression stress. The output signal is an analogue voltage ranging 0 V to 5 V, related to the measured force by Equation 3.3. Where F is the force sensed by the probe in the axial direction, G_{sensor} is the sensor gain, V_{out} is the output voltage when the load is applied, and $V_{out,0}$ is the zero load output voltage [44].

$$F = G_{sensor} \cdot (V_{out} - V_{out,0}) \tag{3.3}$$

For this sensor, the gain has a value of 5000 µN/V and the zero load voltage is 2.25 V. The resolution of the sensor depends on the operating frequency, for 10 Hz the resolution is 0.5 µN, while for 1000 Hz this parameter increases up to 5 µN [44].

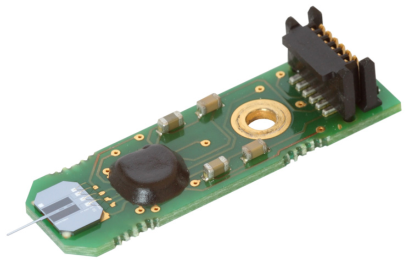


Figure 3.3: Force sensor FT-S10000 of FemtoTools [43]

3.1.2 Tracker NSE-5310

The Tracker NSE-5310 is a linear magnetic encoder that measures displacements, by means of an array of Hall sensors and a magnetic strip attached to the monitored object. It is manufactured by New Scale Technologies. The magnetic strip is made of successive parallel magnetic bands (see Figure 3.4), with opposite magnetic field. So the Hall sensors see a fluctuating magnetic field when the strip moves over it. The frequency of this magnetic field is proportional to the speed of the object at which the magnetic strip is attached.

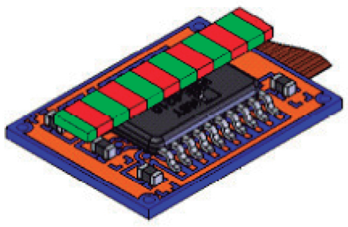


Figure 3.4: 3D model of the Tracker NSE-5310 [41]

According to the datasheet [41], the sensor has a resolution of 488 pm and an absolute error lower than 10 µm in both directions. Concerning the communication, which is scope of this thesis work, the device is equipped with an I^2C based on the version 2.1 released on January 2000 [40]. The address of the sensor is 7 bit wide, being the left-most bit established by the connection of a pin. Bits 1 to 6 can be configured to generate a specific address, though the default direction is 1000000. The encoder generates output data structured in 5 bytes, in order of priority. So the first two bytes contain the linear position and information related to this measurement, and the remaining refers to parameters of the sensor at the time of measuring [41].

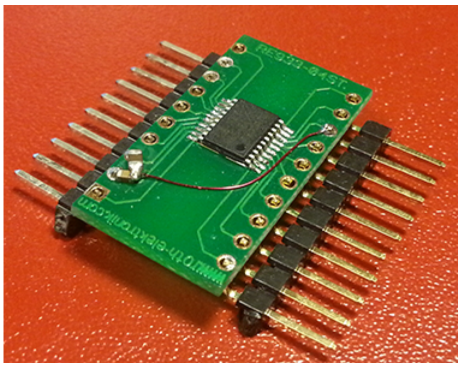


Figure 3.5: Tracker NSE-5310

3.2 BeagleBone Black

This Section will introduce the BBB electronic prototyping platform, and the hardware and software used for the CI-BBB platform. The first Section describes the features of the board and the meaningful components for the present project. Then, the relevant software tools used for the devlopment of the control software are presented.

3.2.1 Hardware description

BBB is a low-cost development platform designed for prototyping and to be used by developers. Operates with a Sitara^{TM}AM3359AZCZ100 microprocessor, featuring an ARMv7 Cortex-A8 microprocessor core. The AM3359 microprocessor is developed by *Texas Instruments*, and includes a wide variety of additional hardware e.g., Inter-Integrated Circuit (I^{2}C) bus, a Universal Asynchronous Receiver/Transmitter (UART) modules, a 12 bits Analog to Digital Converter (A/D converter) or a Graphic Accelerator.

The most relevant On Chip Peripherals, within the scope of this thesis work, are the Analog to Digital Converter, the Programmable Realtime Unit and Industrial Communication Subsystem (PRU-ICSS), the Enhanced Direct Memory Access system (EDMA3), the Inter-Integrated Circuit module and the General Purpose Input/Output (GPIO). Figure 3.6 depicts a block diagram of the microprocessor with the just mentioned OCP.

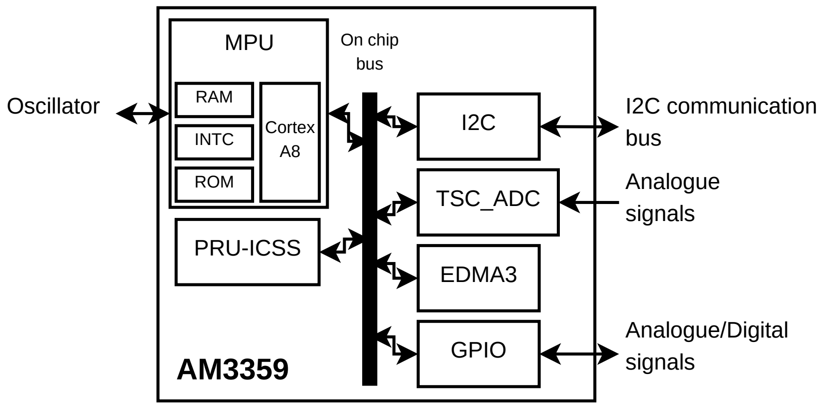


Figure 3.6: Block diagram of the microprocessor with the most relevant OCP

The platform is shipped with a Ångström distribution of Linux OS. However, the prototyping platform is aimed to work with different OS, which can be found in the Support Web page of the platform[11]. For this project, the Ubuntu 13.04 distribution is used, due to the lack of resources of the Ångström distribution.

The Sitara ARM3359 is a 32-bit Reduced Instruction Set Computer (RISC) microprocessor based on the ^{TM}ARMv7 Cortex-A8 processor [23]. This ARM microprocessor core contains two levels of cache memory: level one includes two memories, the instruction and data cache memories, each of 32 k, while the second consists on a single 256 k cache memory. Along with the Cortex-A8, the microprocessor unit (MPU), includes the Read Only Memory (ROM) of 176 k, the Interrupt Controller (INTC), with capacity for 128 interrupt lines, a 64 k on chip RAM, and additional hardware for emulation and debug procedures [23]. An external 2 G Multi-media Card (eMMC) which includes a Secure Digital (SD) slot for SD cards, provides to the BBB a non-volatile data storage. It also allows to store the OS, so there is no need of using an external SD card to boot the system as with the Raspberry Pi. [7; 12]

The GPIO pins can be accessed through the expansion headers of 46 pins, P8 and P9, located at both sides of the platform. Figure 3.7 shows the BBB used in this thesis work.

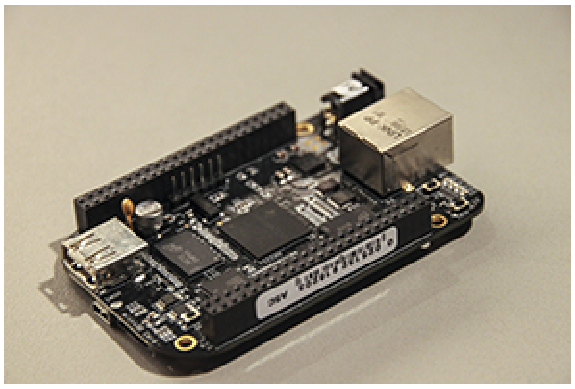


Figure 3.7: BBB. Revision A5C

Analog to Digital Converter

The Touchscreen Controller and Analog to Digital Converter Subsystem (TSC_ADC Subsystem) comprise a 12-bits SAR A/D converter (see Section 2.3), which provides the platform with the capability of reading analogue signals, and to manage a resistive touchscreen (see Figure 3.8). As it can be seen in Figure 3.8, both features are implemented within the same module. Nevertheless, the programmer can active any of these options separately i.e., the TSC_ADC Subsystem can be used as normal A/D converter, or as touchscreen controller. From now on this Section will focus on the A/D converter, being the touchscreen controller out of the scope of the present project.

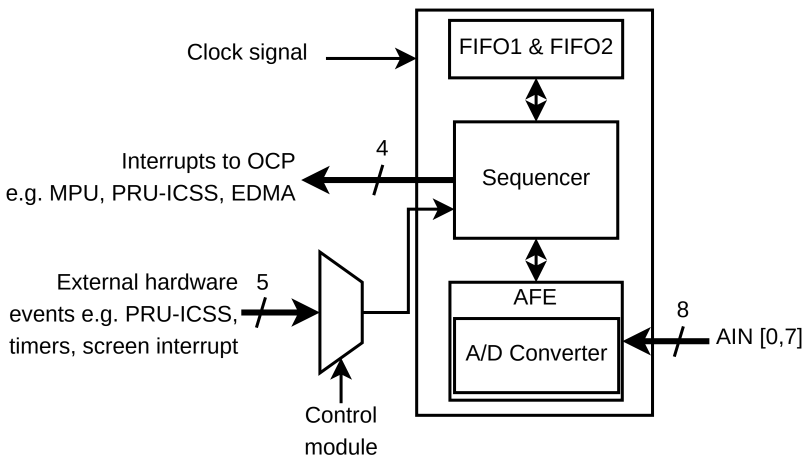


Figure 3.8: Block diagram of the TSC_ADC subsystem [22]

As shown in Figure 3.8, the module has two First In First Out (FIFO) memories of 16-bit wide and 64 k deep, five multiplexed triggering input signals coming from other neighbor hardware modules, three interrupts heading towards the MPU, the PRU-ICSS and the wake signal, two output events to the EDMA3 module and two physical addresses to access neighboring peripherals memory space; to support the conversion process and control of the A/D converter. The analogue input signals are transmitted from the eight GPIO Analogue Inputs (AIN) terminals [22], however only seven of them (AIN0 to AIN6) are physically connected to the expansion header P9 [7]. [23; 7]

There are two different clock domains, the first one is governed by the OCP clock which has a maximum frequency of 100 MHz and affects the FIFOs, the bus interfaces and all the other hardware on the module, except for the proper A/D converter. The second clock domain belongs to the A/D converter clock, which controls the Analogue Front End (AFE) block. The AFE includes the SAR A/D converter, thus the sampling rate is determined by the frequency of the A/D converter clock. This clock can be configured up to 24 MHz frequency, with which the maximum sampling rate can be achieved. According to [23] the conversion process can take up to 15 A/D converter clock cycles. Additionally, the open and sample delays affect the overall performance of the A/D converter by introducing waiting times between sample conversions. Also the master clock, which controls all the microprocessor, can be used with the AFE functional block, only if the signal is adjusted to be under 24 MHz, by means of the clock divider register included in the subsystem. [23]

The open delay refers to the time, in number of the A/D converter clock cycles, that can pass before the conversion procedure starts, while the sample delay relates to the number of cycles the start conversion signal is set to high level. Until the start conversion order is not lowered, the process does not start. The sample delay has a minimum value of one clock cycle, unlike the open delay that can be set to zero. These two properties allows the user to precisely control the sampling rate of the A/D converter. [23]

Regarding the control of the system, the conversion process can be triggered by means of different sources. The interrupt controller logic allows the system to capture software and hardware events. Software interrupts are triggered by other OCP, while hardware interrupts can be set from outside of the BBB through the GPIO input ports i.e., when some object touches the touchscreen, or lifts from it. Also the ADC_TSC subsystem is capable of triggering interrupt signals to other peripherals i.e., when the conversion process is finished or the FIFO memories are full. [23].

The conversion process is determined by the configuration loaded into the *Step Configuration registers*. There are 16 sets of configuration registers and include information about the channel to sample, reference voltages, the FIFO memory used or the interrupt type to start the sampling process. Before each loop of the A/D converter, the configuration of the current *Step* is loaded. If more that one *Step*, is configured, these are loaded into the converter in order before each sampling process. Thus, the A/D converter will sample only one analogue channel in each iteration.

Programmable Real Time Unit and Industrial Communication Subsystem

Two independent Programmable Realtime Units (PRU), designed for real time applications, form this subsystem of the microprocessor designed by Texas Instruments. A real time application is characterized by having tight time boundaries. This means, the application is meant to perform the programmed tasks in a fixed human-time, resulting in failure if this time is not meet.

Each of these units are a 32-bit RISC processor, with additional support hardware such as memory, interrupt controller, a multiplier with optional accumulator (MAC), and communication modules e.g., UART or Industrial Ethernet Peripheral. Additionally, it has access to the GPIO channels of the BBB (see Figure 3.9). The PRUs are designed to execute instructions in 5 ns, with a 200 MHz timer that drives these processors.

Each PRU processor is a Harvard architecture with three different RAM memories for store/load data (DRAM): DRAM0, DRAM1 and a shared RAM. The first

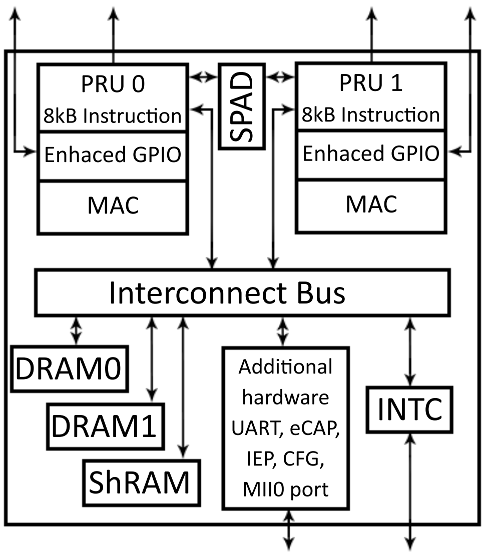


Figure 3.9: Block diagram of the PRU-ICSS [21]

two are meant to be the principal memories of the PRU0 and PRU1 respectively, while the third memory is common to both processor. The shared RAM is 12 k deep, while the DRAM memories are 8 k, being all of them 32 wide. The DRAM memories are integrated in each of PRU processor units, and are used to store information and programs to execute by the PRU units. The instructions are loaded by the host processor at the start up time. The Scratchpad Memory (SPAD) provides a fast communication path between the two processors. To grant the share of information between the subsystem and the rest of the SoC hardware, the memories of the PRU_ICSS subsystem are mapped with global memory addresses, accessible by all the SoC peripherals [21].

The Interrupt Controller (INTC) supports the device with asynchronous communication using external and internal event signals. These interrupts can be generated by other peripherals on the platform, every time the interrupt line between the PRU-ICSS and such hardware exist. The subsystem has 64 different interrupt events, among them 31 interrupts are generated from within the subsystem from modules such as the industrial Ethernet Peripheral or any of the processors PRU, and the 33 remaining are associated to external signals connected to other SoC peripherals e.g., the A/D Converter, the EDMA3 module or the MPU.[23]

The INTC is organized in 20 registers: 10 host and 10 channel registers (see Figure 3.10). This way the designer is able to enalbe and assign priority levels to the different system events. The interrupts are first mapped to the channel registers, being possible to assign more than one interrupt line to the same channel. The channels are then assigned to the host registers, or hosts, which lead to the PRU0/1 (Hosts 0 and 1) or the microprocessor INTC (Hosts 2 to 9). Incoming events are driven to bits 30 and 31 of the register number 31 of the PRU processors. Before the signal reaches the target device, the INTC carries on a set of functions upon the arrival of this event. This routine checks the configured status of the event in the control registers, and therefore the interrupt is asserted or ignored. For the sake of simplicity, Texas Instruments designed the PRU-ICSS in such way that no interrupt routines can be programmed, being necessary a procedure of periodically poll the value of R31 to capture system events. [23]

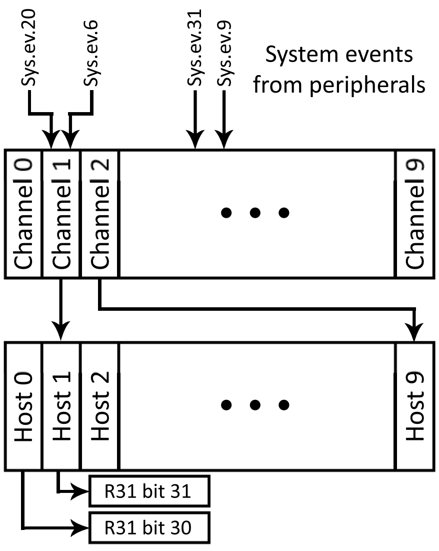
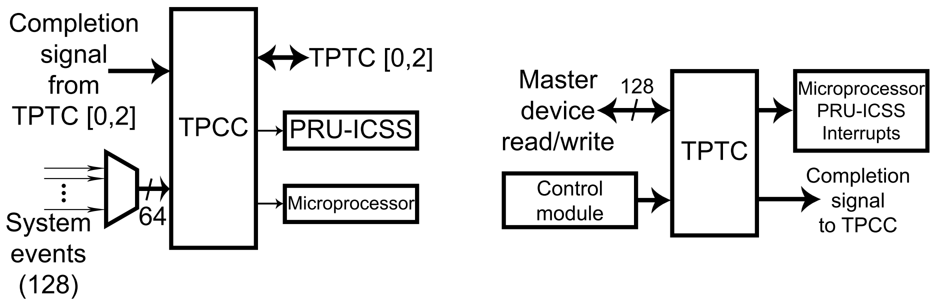


Figure 3.10: Block diagram of the PRU-ICSS [21]

R31 is also used to send pulses through the INTC from the PRUs to different target devices, such as external modules or other internal hardware of the PRU-ICSS, by configuring the first 6 bits as described in [21]. While register 31 is used to send signals outside the subsystem, register number 30 is used to manage input signals captured through the GPIO ports of the AM3359.

Enhanced Direct Memory Access 3

Enhanced Direct Memory Access 3 (EDMA) is a device designed to release the microprocessor for taking care of data transfer between memories, improving the overall performance of the system and allowing the CPU to attend other processes. The system includes two main blocks: the Third-Party Transfer Controller (TPTC) and the Third-Party Channel Controller (TPCC). While the TPTC is in charge of the data transfer between two fixed memories, the TPCC controller allows the user to program and manage the different communication lines, or channels, available for the microprocessor. [23]



(a) (b)

Figure 3.11: Block diagrams: (a) TPCC, (b) TPTC.

Each channel has a set of dedicated registers to configure the data transfer known as Parameter RAM (PaRAM). The addresses of the source and target memories, as well as the amount of data, and how this has to be read and stored is defined in the fields by the user before initiating the transfer. How the data is manipuladed, is defined through six registers. These are the A, B and C indexes (IDX) of the source (SRC) and destination (DST) memories. AIDX defines the number of bytes of each individual data, while BIDX defines the number of data values stored in a frame according to the notation used in [23]. Finally CIDX allows to define BIDX-wide packages. These definitions are applied to the destination and the source memories, allowing the user to easily and precisely select and manipulate the format of the information to be transferred. Regarding the data transmission procedure, the EDMA allows two types of methods depending on the amount of information to be moved.

- **A-Synchronized Transfer**: for each request one A-bytes wide array of data is transferred. Requiring BIDX*CIDX number of requests to deliver all the information to the destination memory. As illustrated in Figure 3.12.

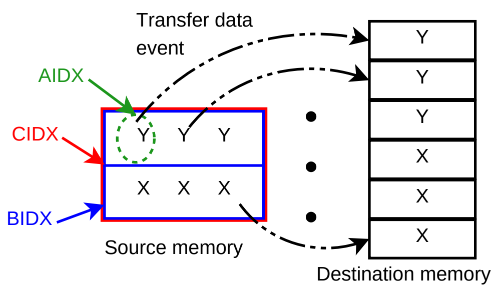


Figure 3.12: A-Synchronization [21]

- **AB-Synchronized Transfer**: as depicted in Figure 3.13, each transfer request triggers the movement of a set of BIDX AIDX-bytes wide arrays. Therefore CIDX transfers are needed to retrieve all the data from the source.

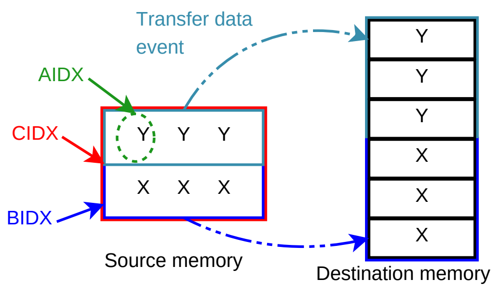


Figure 3.13: AB-Synchronization [21]

The TPCC has 64 Direct Memory Access channels (DMA channels) and eight quick DMA channels (QDMA channels). The main difference between these two types of communication lines, is how they are triggered and how they behave after the transfer is completed. The QDMA channels can be triggered by two different procedures both of them based on writing the trigger word on the parameter set.

- **Auto-triggered** the transfer is initiated as soon as the trigger word is written.
- **Link-triggered** used within a set of successive transfers. The process starts whenever the PaRAM set is updated at the completion of the previous data transmission.

On the other hand the DMA channels have three different triggering methods to initiate a data transfer: *manually-triggered*, *event-triggered* or *chain-triggered*. The data transfer requests are captured through the Event Set Register (ESR) of the EDMA. So the difference between these procedures is based on how this register is set.

- **Manually-triggered** the register is set directly from the CPU or other peripheral by writing to the appropriate bits.
- **Event-triggered** modifies the event set register whenever an interrupt asserted by a peripheral arrives to the module.
- **Chain-triggered** the completion of a transfer, triggers the next one.

Inter-Integrated Communication bus

The details of this communication protocol can be found in Section 2.3.2. This Section refers to the I^{2}C module implemented in the microprocessor AM3359 of

Texas Instruments. It is based on the I^{2}C protocol version 2.1 of Philips, released on January 2000 [40]. As any standard I^{2}C bus, it supports several master devices (multimaster) working as data transmitters or receivers, communicating with the slaves attached to the bus. All these devices can be accessed with a 7 or 10-bits wide address. The data speed can be configured in two modes: standard mode (100 kbits/s) and fast mode (400 kbits/s). The microprocessor modules feature two DMA channels, one interrupt line and a 32-bytes wide internal FIFO memory for buffering. Being three the number of I^{2}C modules, the microprocessor has (I^{2}C 0,I^{2}C 1 and I^{2}C 2). The first module additionally has an interrupt line dedicated to the PRU-ICSS, but it cannot be accessed from the GPIO pins on the expansion headers of the BBB, unlike I^{2}C 1 and I^{2}C 2.

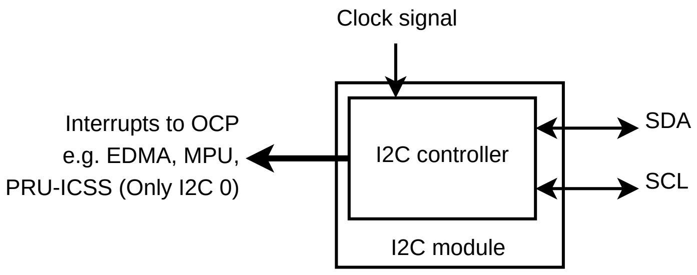


Figure 3.14: Integration of the I^{2}C bus [21]

As illustrated in Figure 3.14, each module includes a bidirectional data (SDA) and a bidirectional clock (SCL) pins, noise filters for each of the pins, data registers for buffering the information, interrupt controller, DMA event generator and a peripheral data bus interface, allowing communication with other OCPs. According to [7], the subsystem I^{2}C0 is used to control the power rails of the power regulators on the BBB i.e., as an interface between the power regulator (model TPS65217C), and the microprocessor. The pins of I^{2}C1 and I^{2}C2 are connected to the expansion header P9, to pins 17, 18 and 19, 20, respectively.

The start and stop conditions are a high to low level and a low to high level transitions, respectively. Data must be stable during the high level of the clock signal, to ensure proper sending of the information. The data scheme is the same as explained in Figure 2.22. But, when using a 10-bits address, the direction is provided in two consecutive bytes. The first has a '11110' binary value, starting from its most left handed but, being the remaining ten bits the desired address. The Read/Write and acknowledge bits are located between the two bytes that compound the address. An additional capability of the I^{2}C modules is the *Draining Feature*, which allows to recover the remaining data in the FIFO whenever the received data is not enough to fill the memory. [22]

Under Linux OS, the I^{2}C modules are mapped to system files *i2c-0*, *i2c-1* and *i2c-2*, located in the directory `/dev`. While *i2c-0* refers to module I^{2}C0 of the AM3359, *i2c-1* and *i2c-2* are related to modules I^{2}C2 and I^{2}C1, respectively. By default, only I^{2}C0 and I^{2}C2 are initialized. While I^{2}C0 is used for communications between different components of the BBB, and thus, not exposed in the expansion headers, I^{2}C2 can be used through pins number 19 (SCL) and 20 (SDA) of the expansion header P9.

I^{2}C1 can be enabled by recompiling the kernel or loading the device tree overlay BB-I2C1 to the cape manager of the BBB, being then mapped to file *i2c-2*. This module can be accessed through pins 17 (SCL) and 18 (SDA) of the expansion header P9. These pins are also exposed in the cape designed for the CI-BBB to ease the connection with this module. [9]

Cape for BeagleBone Black

A cape consist on additional hardware, typically build on a circuit boards, that can be connected to the BBB through the expansion headers P8 and P9. These circuits are normally sold with software designed for the application for which the cape is designed e.g., the capability to control a 3D printer, work as media center or manage robotic manipulators. Some examples can be found in Figure 3.15: the WL1835 W/ Chip Antenna (a) includes a Wi-Fi and Bluetoothantennas, and the LCD7 cape (b) is a touchscreen with five additional buttons [11].

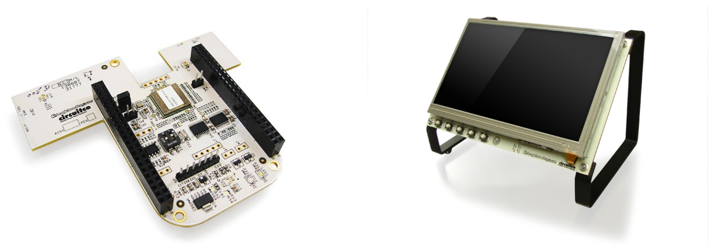

(a) (b)

Figure 3.15: Examples of capes: (a) WL1835 W/ Chip Antenna, (b) LCD7 cape [11].

3.2.2 Software support

Two main libraries were used to support the software developed in this thesis. AM335x PRU Package to program and control the PRU-ICSS subsystem, and LibPruIo, based on the previous, which implements routines to control the TSC_ADC subsystem and the GPIO. Apart from these packages, the device tree files were used to enable the hardware of the BBB. In this Section, these three elements are described.

Device tree

To ease the integration of new hardware in the computer system, Linux OS provides a special binary files with a detailed description of the hardware, called *Device Tree*. These files are upload to the *Cape Manager*, a specific software to load the hardware information to the kernel of the OS. The different connections e.g., interrupts and communication lines; and nodes the new hardware device has, are described in these files. [8]

In this thesis work, the Device Tree files BB-BONE-PRU-01, BB-ADC-00A0 and BBB-I2C1. The first file contains information of the PRU-ICSS subsystem, next file describes the A/D converter, while the second is related to the I^2C module.

AM335x PRU Package

AM335x PRU Package is a software package developed by the BeagleBone community to program and control the PRU-ICSS, based on the software development package released by Texas Instruments on June 2010 for the previous version of the Programmable Realtime Unit Subsystem of the OMAP-L138, C6748 and C6746 microprocessors [18]. It includes libraries with the implementation of the different functions and macros, and an assembler to generate the binary files to be executed by the PRUs from the assembly source file written by the programmer [3].

The tools included allows the programmer to load programs to the processors and execute them. Also, manage the memory and interrupt events of the subsystem, communicate the host processor with the PRU-ICSS and access other OCP from the PRU processors directly. The instruction set to program the realtime units is composed of mnemonic instructions with parameters. These parameters can be registers, immediate values or labels. The written format consists on the name of the instruction followed by the parameters separated by commas. As said in 3.2.1 the execution time of almost all instructions is 5 nsec (one clock cycle when running at 200 MHz), only when accessing an external memory location from the PRU, the number of cycles required to complete the task increases [3; 21].

The instructions are classified in four different groups: six arithmetic, eleven logic, fourteen Input/Output and seventeen program flow control functions. Making a total of forty eight operations. Additionally there are four assembler statements allowed: dot commands, labels, hash commands (#), and instructions/pseudo-instructions. Comments are single line and identified as in C/C++ programming languages, using the double slash (/) [3].

LibPruIo: AM33xx-PRU driver for digital input / output and analog input

Despite the advantages the Programmable Real Time Unit subsystem offer for realtime applications, its main drawback its the obscure programming procedure and management of these devices. The impossibly of debugging programs on these processors without the use of community supported programs, with any warranty, increases the development time of new software for these platforms. The LibPruIo library by Thomas Freiherr was released to provide easy-to-use tools to program and control the PRU-ICSS, the Control Module, the GPIOs, the TSC_ADC subsystems of the BBB, easing the task of the programmer releasing him from writing assembly code or kernel drivers [14].

This library is programmed in FreeBASIC programming language, developed on a BBB under Ubuntu 13.10 and compiled using FreeBASIC compiler. It is possible to use the functions and macros in a C-based program by means of a already programmed wrapper. The software uses the PRUs to control and configure the other devices and as interface with the host process.

LibPruIo implements two operation modes: measurement mode (MM) and input/output mode (IO). The first method configures the A/D converter to sample certain number of samples, indicated by the user. The processor which host the main program waits till the conversion has finished to get the data. On the other side, the IO mode keeps the A/D converter converting the analogue input signals continously. The samples are stored in a ring buffer i.e., when the memory is full, the pointer is moved again to the first memory direction rewriting the old data when new samples are stored [14].

Additionally, the MM mode allows the user to specify the sampling frequency i.e., number of samples obtained form the A/D converter per second, by programming an external timer. In the case of the IO mode, the sampling frequency is determined by the *sample delay* and the *open delay*. The open delay refers to the time between a new analogue signal arrives to the A/D converter, and the start of the conversion. The sample delay is the number of clock cycles the start conversion signal is kept in high level. This time can be used to change the configuration context of the converter [23]. The library can also be used to configure other parameters of the A/D converter such as the clock frequency of the converter or the possibility to perform an average operation of the samples.

Please, notice the difference between the sampling frequency at which the documentation of the library refers [14], and the sampling frequency at which the A/D converter handles the analogue signals. The clock signal that drives the A/D converter is a 24 MHz signal. Because the device is a SAR converter that takes 15 cycles to complete a conversion, resulting in a sampling frequency of 160 kHz i.e., it generates 160000 valid samples every second. Regarding the sampling frequency of the library, this refers to the amount of valid samples that are subtracted from the TSC_ADC subsystem per second. Thus these two frequencies can be different during the same process.

The functions, macros and definitions are classified within three different C++ classes:

- **AdcSteps**: macros and definitions for the configuration of the steps (see Section 3.2.1) and delays of the A/D converter.
- **GpioSet**: macros and definitions to describe the setting of the GPIOs modules.
- **PruIo**: implements functions to create, configure and upload drivers to the PRU-ICSS (see Section 3.2.1), manage the steps of the A/D converter, supervise and manipulate the state of the GPIO pins.

The communication between the main process and the PRU-ICSS is done through the local memory of the PRUs and an additional memory space located in the RAM of the MPU. This additional memory is allocated when uploading the device tree of the PRU-ICSS subsystem to the *cape manager* of the OS. In IO mode only the local data RAM memory of the PRUs 0 and 1 is used, while the additional external memory is used in MM mode. By default the processor allocates 256 kB for the external memory, being possible to enlarge or shrink this space using the following kernel instruction, as explained in [14]:

```
/sbin/modprobe uio_pruss extram_pool_sz= <Size of memory>
```

4. COMMUNICATION INTERFACE

As explained in the Introduction Chapter, the main objective of the CI-BBB is to serve to bridge the sensors of the MP and the control station or remote computer i.e., external computer from where to monitor the CI-BBB and the MP. This Chapter describes the design of the CI-BBB. For that purpose, the content is divided as follows. Section 4.1 concerns the VA circuit, designed for the system, and the measurements done to the prototype circuits. Section 4.2 presents the cape board designed for this thesis, which includes the VA circuits and additional hardware such as control buttons and an state LED. The software library programmed for the CI-BBB and a remote control software is detailed in Section 4.3. The final Section 4.4 collects the results of the measurements and tests made to the prototype of the CI-BBB. Figure 4.1 illustrates a general block diagram of the CI-BBB, in which the contents of this Chapter are displayed.

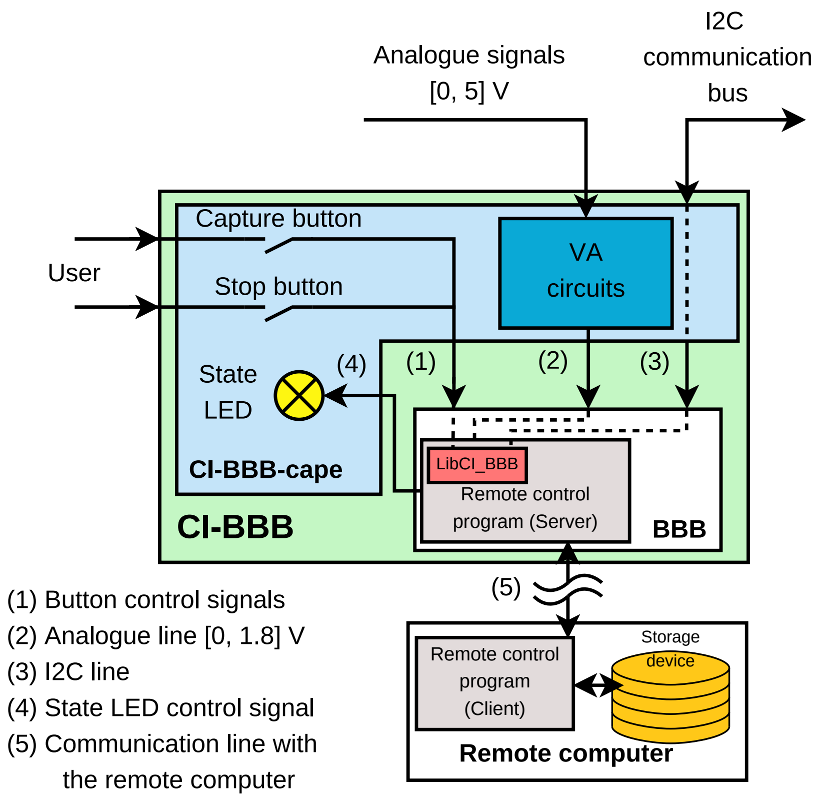


Figure 4.1: Block diagram of the CI-BBB

4.1 Voltage Attenuator circuit

Due to restrictions imposed by the ARM 3359 microprocessor of the BBB, the maximum input voltage the analog input pins (AIN) can withstand is 1.8 V. Because the force sensor provides an output voltage ranging from 0 V to 5 V, a circuit to adapt the voltage level is required to ensure the proper operation of the CI-BBB.

This Section describes the VA circuit designed for the analogue input pins of the CI-BBB. Sections 4.1.1 and 4.1.2 present two approaches for this circuit. First based on a voltage divider, and the second based on the use of OA. Section 4.1.4 includes measurements and results to validate the performance of the circuits. Finally, Section 4.1.3 discusses to the electrical components used in the VA prototype circuits.

4.1.1 Voltage Attenuator based on voltage divider circuit

The voltage divider circuit was already introduced in Section 2.2.1. The main advantages of this circuit, over the OA approach, are the ease of tune, ease of assembly and the price. It is a cheaper solution, as it will be seen in Table 4.4. For the prototype circuit, the two resistors were replaced by a potentiometer as depicted in Figure 4.2, where the schematic circuit model of the prototype can be found.

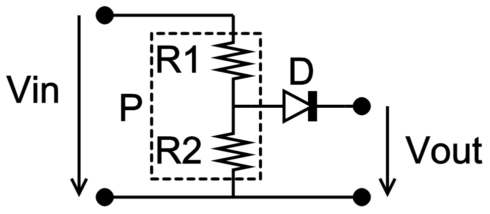


Figure 4.2: Voltage divider prototype circuit based on a potentiometer

Using Equation 2.3 for an output voltage of 1.8 V and an input voltage of 5 V, results in a relation between the total resistance of the potentiometer and the output resistor of $\frac{R_2}{R_{total}} = 0.36$. The trimmer potentiometer used for this circuit has 10 kΩ of total resistance, thus it has to be tuned so the input resistor is 6400 Ω and the output resistor is 3600 Ω. Figure 4.3 depicts the simulation waveforms of the input and output voltage signals generated using the OrCAD PSPICE software. For this simulations, the output diode depicted in Figure 4.2 was not used. The input signals are a constant 5 V signal, and a sine function with 2.5 V of amplitude, 2.5 V of offset and 1 Hz frequency.

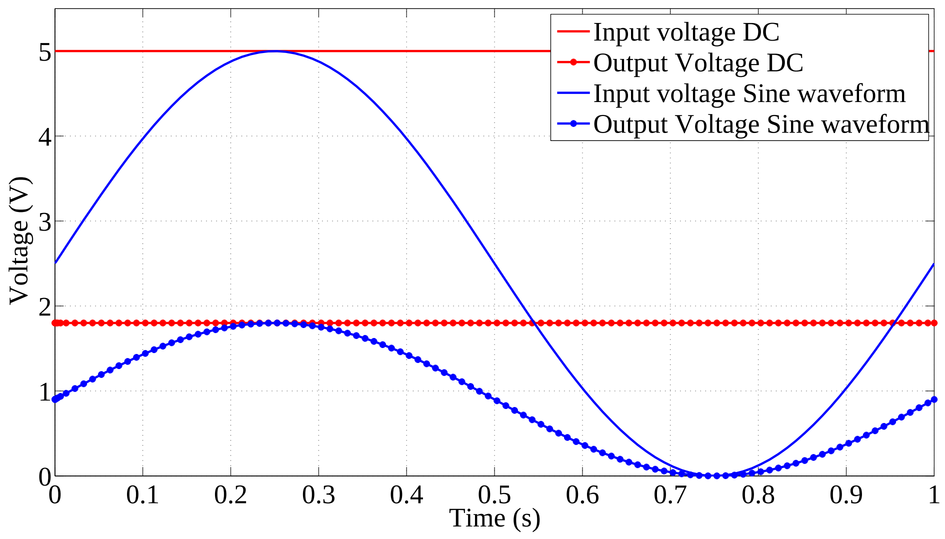


Figure 4.3: Simulations of the PSPICE model of the voltage divider circuit

As can be observed, the calculated value for the set point of the potentiometer is correct. The output voltage signal is proportional to the input voltage, ranging 0 V to 1.8 V.

Because the analogue pins of the BBB are configured as sink, a diode is placed at the output side of the VA circuit to prevent power to flow from the BBB to the sensor. But, the non linearity of the diode near the knee voltage derives in a distortion of the output voltage waveform, and thus, in a loss of accuracy of the measure. The simulation waveform using the output diode can be found in Figure 4.4, it can be seen how near the zero voltage, the output waveform presents a distortion due to the presence of the diode.

4.1.2 Voltage Attenuator based on operational amplifiers

As introduced in Section 2.2.2, the OA can be used to design amplifying circuits. Attending to the topologies presented in Section 2.2.2, the non-inverting amplifier results to be inadequate for the VA circuit owing to the fact that the closed loop gain is higher than the unity, as it can be deduced from equation 2.15. Besides, the closed loop gain of the inverting amplifier can reach values below unity when the input resistor is greater than the loop resistor, feature that makes it suitable for the purpose of the Voltage Adapter.

Since the output voltage of the inverting amplifier is in opposite of phase with the input, a second inverting amplifier is used in order to produce a voltage signal in phase with the original. The amplifiers are connected in a cascade configuration

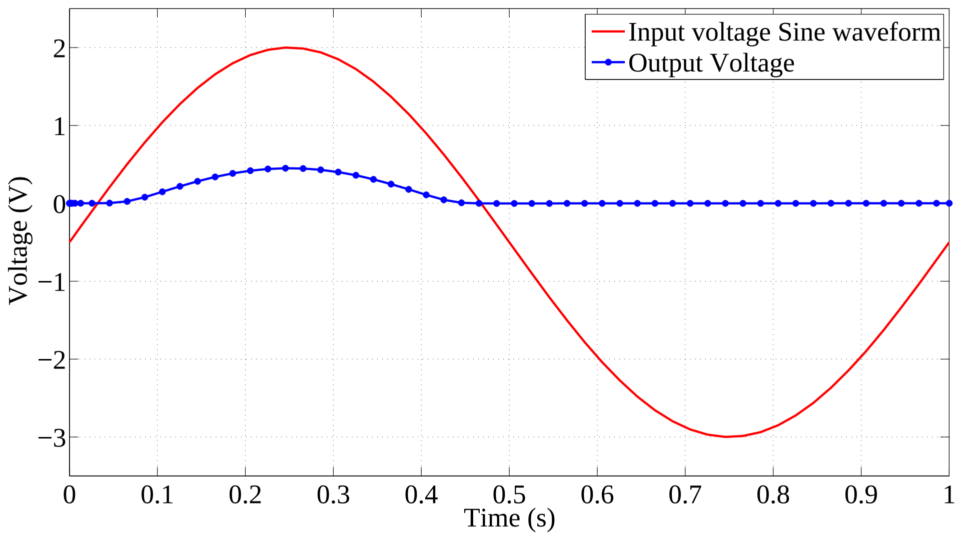


Figure 4.4: Effect of the output diode over the load voltage waveform

so the output of the first stage feeds the input of the second. Thus, the first stage produces the reduction of the voltage level and the second corrects the polarity of the signal. The model of this circuit can be found in Figure 4.5.

From the relation between the input and output voltage derived in Section 4.1.1 ($\frac{V_{out}}{V_{in}} = 0.36$) and Equation 2.13 the values of the input and loop resistors are calculated being 287 Ω and 800 Ω respectively. The resistors of the second stage should be the same so the closed loop gain equals the unity, hence 1 kΩ resistors are used.

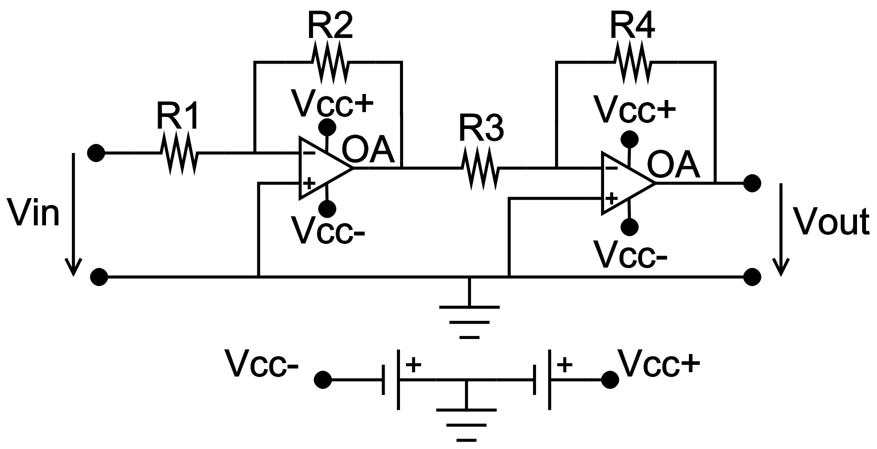


Figure 4.5: Circuit of the VA based on OA

The waveform of the output signals when driving the PSPICE model circuit, with a constant voltage and a sinusoidal signal of 1 Hz, is depicted in Figure 4.6. From this Picture it can be seen that the ratio between the output and the input voltage levels is 0.36 units, with no distortion of the waveform the of output.

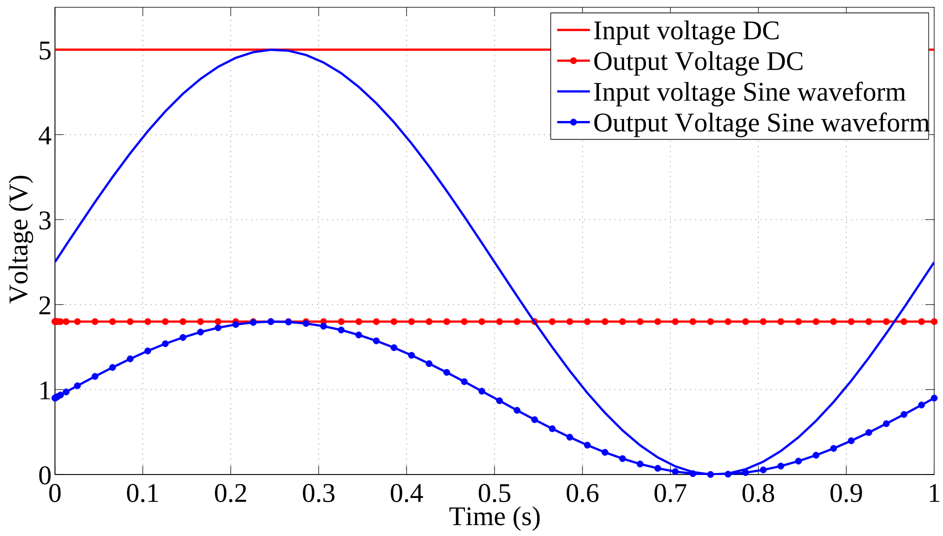
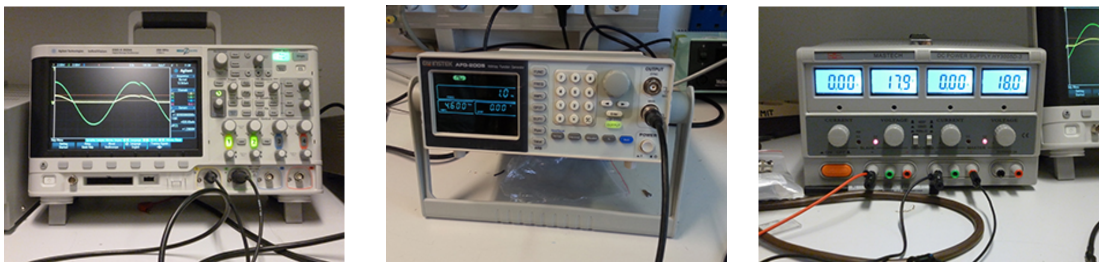


Figure 4.6: Simulations of the PSPICE model of the operational amplifier

4.1.3 Measurements and circuit selection

The measurements obtained to validate the proposed circuits are taken using a DSO-X 2024A Oscilloscope of Agilent Technologies. The input signals are produced by a function generator (AFG-2005 of Gw Instek), while the voltage supply for the operational amplifiers is provided by a regulated power source (HY3005D-3 of Mastech). This equipment is displayed in Figure 4.7. The data obtained from the oscilloscope is analyzed using MATLAB software, developed by Mathworks®.



(a) (b) (c)

Figure 4.7: Measurement equipment: (a) oscilloscope, (b) function generator, (c) power source.

The physical implementation of the already mentioned circuits can be found in Figure 4.8. Special attention deserves the difference of size between both circuits. While the circuit (b) requires a larger area to be implemented, circuit (a) can fit in a smaller surface. This aspect is desirable for the design of the cape for the BBB, where the area is limited (See Section 4.2).

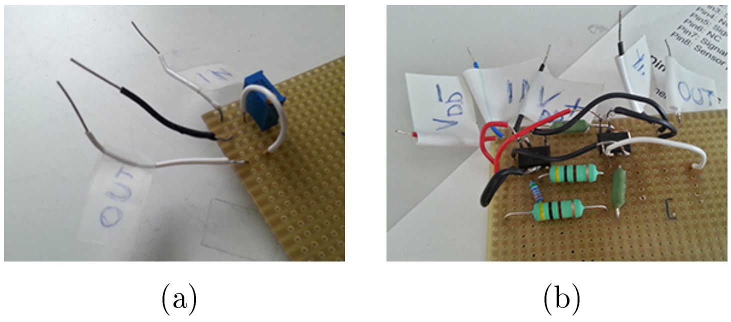


Figure 4.8: Prototype circuits of the VA: (a) potentiometer, (b) operational amplifiers approach.

The input signals for testing the prototypes are the same as the ones used with the PSPICE models. Figure 4.9 shows the output and input voltages measured on the potentiometer prototype. Considering the case of a constant voltage input, the average voltage level of the input and output signals are 4.93 V and 1.74 V, respectively. Thereby, the relation between the voltage levels is 0.353, slightly below 0.36. The behavior under a sinusoidal input reveals that the circuit does not distort the waveform, besides the voltage level attenuation.

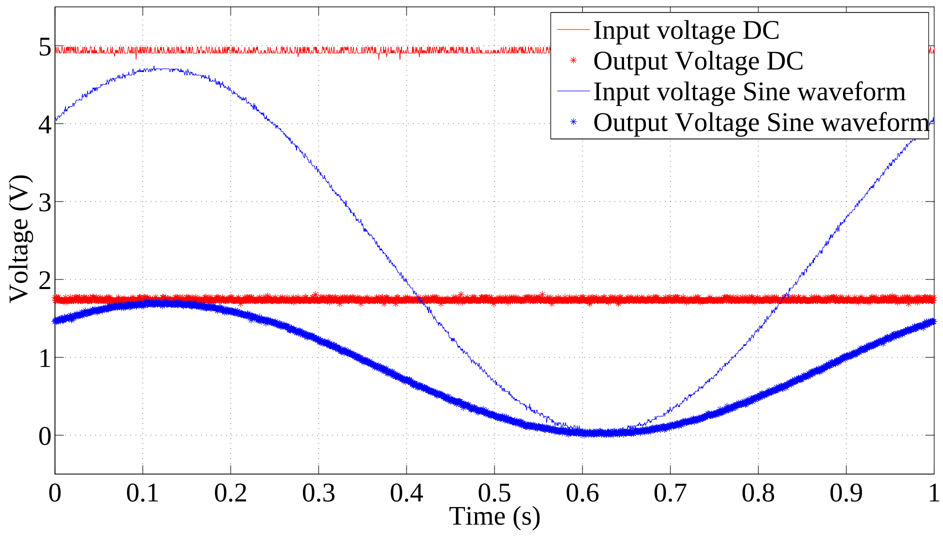


Figure 4.9: Measurements of the potentiometer circuit

In the case of the operational amplifier circuit (see Figure 4.10), the voltage gain is even lower than in the previous circuit, with a approximate value of 0.33.

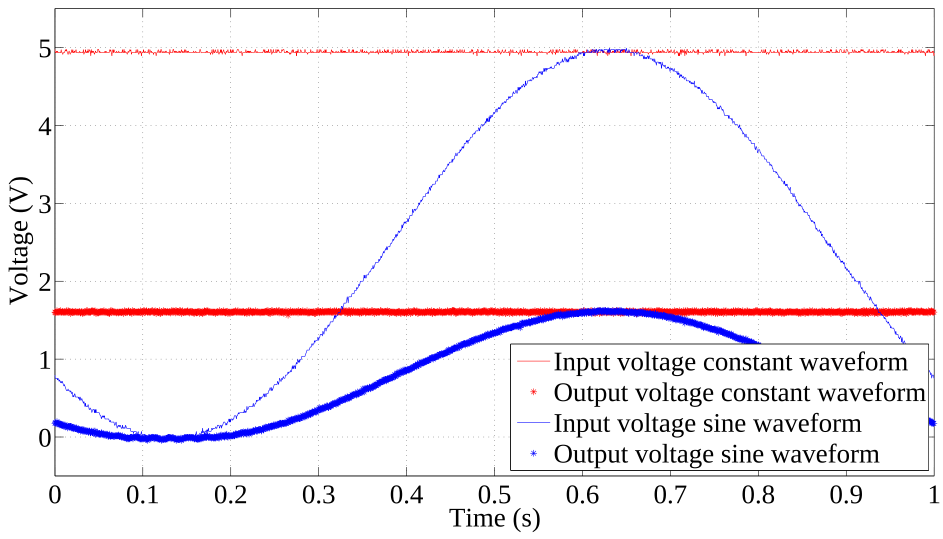


Figure 4.10: Measurements of the OA circuit

Figure 4.11 shows Fast Fourier Transform (FFT) of the output and input voltage level, measured with the oscilloscope shown in Figure 4.7. The sine waveform used for the previous test is supplied to the input of the circuits. The red signal corresponds to the FFT of the input signal, while the blue line represents the FFT of the output signal. The upper plot illustrates the potentiometer prototype, and lo lower refers to the OA VA.

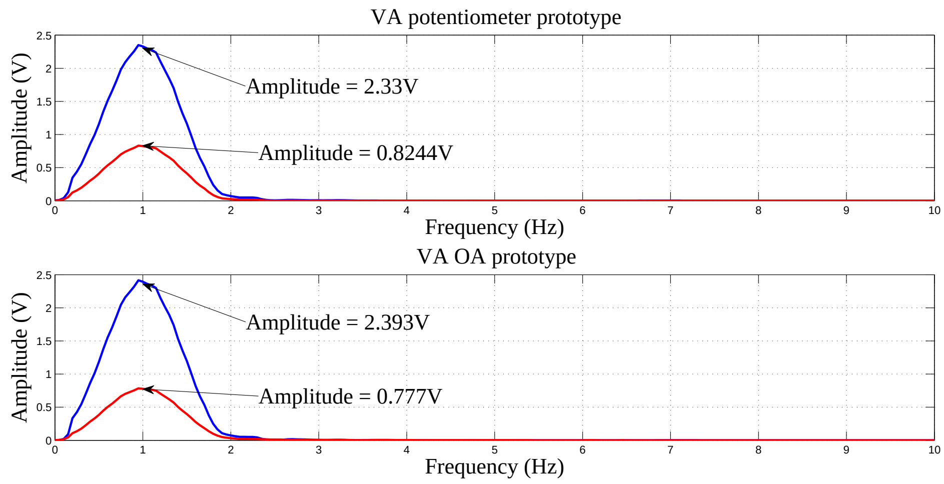


Figure 4.11: FFT of the input and output signals of the VA prototypes. The FFT of the input sine waveform is depicted in blue color and the output voltage signal in red. The upper plot corresponds to the potentiometer prototype, and the lower to the OA prototype circuit.

It can be concluded, that there is no difference in terms of the voltage attenuation or the distortion of the input waveform. The voltage divider solution is selected instead of the AO circuit based on the ease of assembly, the lower number of components, the lower cost of the total circuit and the small size of the circuit, making it suitable to be easily implemented on the cape for the BBB.

4.1.4 Components of the Voltage Attenuator prototypes

The criteria established for the selection of components for the prototype circuits is based on three main statements. First, the component should meet the requirements of the application and the constrains imposed by the BBB. Next, because the aim of this project is to develop a low-cost communication interface, the financial issue is considered an important aspect, to avoid large investments at the early stages of the project. Finally, due to the fact that the prototypes are build manually, the facility to solder the components becomes a desirable feature to take into consideration.

For the implementation of the circuits in the cape board (see Section 4.2), an additional requirement should be considered. To ensure the portability of the CI-BBB, the circuitry of the cape should be powered by the BBB, using the 1.8 V, 3.3 V or 5 V power pins available in the expansion header P9. Besides, for the prototype circuits, this issue is not considered.

The candidate components for the OA of the VA prototype circuit (see Section 4.1.2) are exposed in Table 4.2. General purpose amplifiers were selected to be used for this circuit. The desired range of values for the main parameters of these devices are exposed in Table 4.1. These values are estimated based on the theory presented in Section 2.2.2, and the specifications of the force sensor described in Section 3.1.1.

Table 4.1: Required values for the parameters of the OA for the VA prototype

Differential input voltage	>5 V
Input voltage	Sustain 5 V
Price per unit	<0.5 €
Slew rate	>0.5 V/µs
Bandwidth	>1 kHz
CMMR	>70 dB
Price	Lowest possible

Among the models presented in Table 4.2, the OA chosen for the prototype circuit is the 741CPE4 of Texas Instruments, attending to the requirements presented in Table 4.1. The low price and ease to be mounted are the main factors that favored the selection of this component for the prototype of the VA.

Table 4.2: Selection of OA available in *Farnell*, for the prototype of the VA [39] [17] [20] [19] [33]

	LM741C(N)	TL072	LM741(H)	µA741CPE4
Manufacturer	T.I.	DIODES	T.I.	T.I.
Voltage supply	±18 V	±18 V	±22 V	±18 V
Power Dissipated	500 mW	860 mW	500 mW	500 mW
Differential input voltage	±30 V	±30 V	±30 V	±15 V
Input voltage	±15 V	±15 V	±15 V	±15 V
Price per unit	0.69€	0.35€	7.55€	0.32€
Slew rate	0.5 V/µs	13 V/µs	0.5 V/µs	0.5 V/µs
Bandwidth	1 MHz	3 MHz	1 MHz	1 MHz
CMRR (Min./Typ.)	$[70, 90]$ dB	$[75, 100]$ dB	$[70, 90]$ dB	$[70, 90]$ dB

In Table 4.3 can be found the components used in the VA prototype based on OA, shown in Figure 4.5. All the electronic components listed can be found at *Farnell element 14* online shop [10].

Table 4.3: Components of the VA prototype circuit based on OA

Symbol	Component	Farnell code	Manufacturer	Price
R_1	Resistor 287 Ω	1563161	Multicomp	0.067€
R_2	2xResistor 400 Ω (series)	1903838	Multicomp	0.26€
R_3 and R_4	Resistor 1 kΩ	1735061	Vishay Draloric	0.44€
µA741	Operational amplifier	1210962	Texas Instruments	0.32€
Total cost				1.787€

The potentiometer for the prototype circuit, was already given by the laboratory where this thesis work was done. As a reference, in Table 4.4 a similar trimmer potentiometer is exposed.

Table 4.4: Components of the VA prototype circuit based on a potentiometer

Symbol	Component	Farnell code	Manufacturer	Price
P	Potentiometer	2396041	Vishay Sfernice	1.13€
Total cost				1.13€

4.2 The cape implemented for the BeagleBone Black

The cape board (CI-BBB-cape) contains the VA circuits the analogue input channels of the A/D converter, as well as the input pins of the I^{2}C modules, the control buttons and the state LED. So, all the hardware required for the device is integrated on the same board, fixed to the BBB, improving the portability and usability of the CI-BBB.

Four out of seven analogue inputs are provided with a VA circuit, leaving 3 channels available to use with sensors that can provide up to 1.8 V. The two buttons are aimed to allow the user trigger the capture routines by pressing the *Capture Button*, and stop the manual operation mode process using the *Stop Button*. The status LED lights up when the CI-BBB is capturing new data. These elements are highlighted in Figure 4.12, which shows the final CI-BBB-cape.

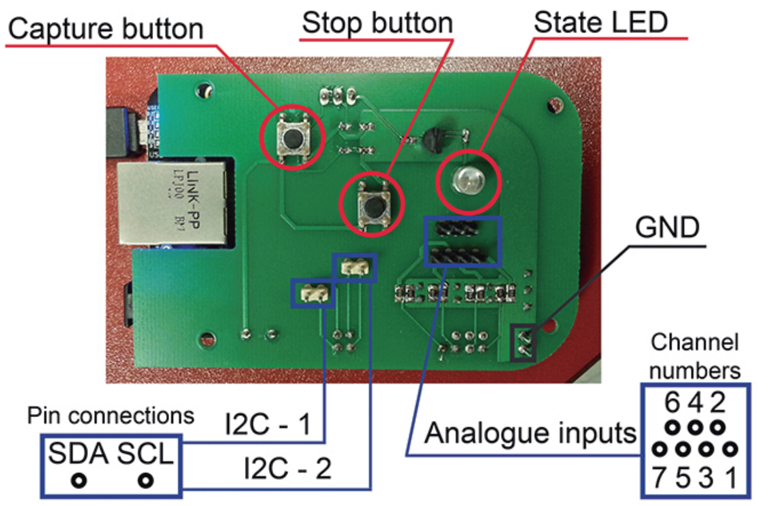


Figure 4.12: Power gain of the proposed voltage attenuators against frequency

The schematic circuits of the VA, the buttons and the LED circuits are exposed in Figure 4.13. The same criteria as in Section 4.1.1 bases the component selection for the cape board. Additionally, for the CI-BBB-cape, the requirement of using low powered components is taken into account. Moreover, surface mounted components are preferred for this circuit, because of the small area of the cape board. A Printed Circuit Board (PCB) is used to implement the CI-BBB-cape. Due to the high number of components used for the CI-BBB-cape, the PCB is a better choice compared to the prototype boards of parallel rails, which could have hindered the design of the circuitry.

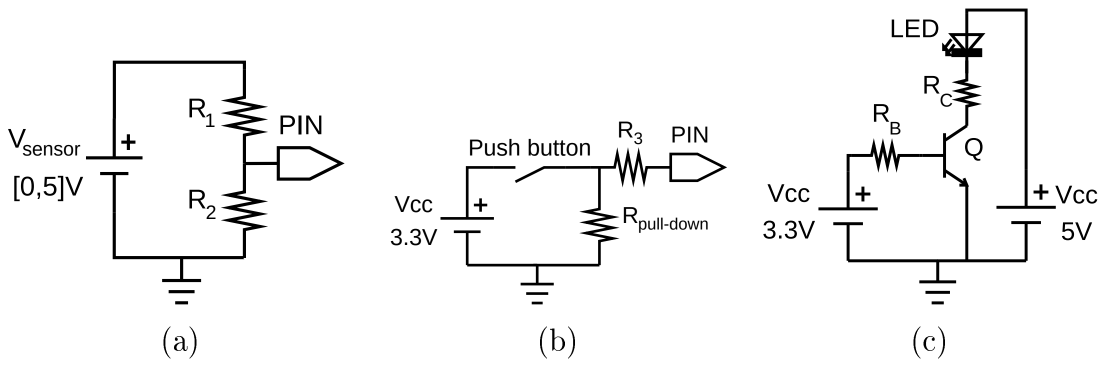


Figure 4.13: Schematic circuits of the CI-BBB-cape: (a) VA, (b) buttons, (c) status LED

The components of the cape are listed in Table 4.5. The resistors values are chosen so the relation between the input and output voltage is 0.36, as explained in Section 4.1.1. For the push-buttons circuits a pull-down resistor of 10 kΩ is used, to ensure a logical 0 input when the contact is open, and a 100 Ω resistor is used to limit the current supplied to the pin. The state LED has a maximum forward current of 25 mA [28], for that amperage the selected transistor has a $\beta \approx 110$ [38], which results in a base current of 0.22 mA. For a base-emitter voltage of 0.8 V and the 3.3 V source, the base resistor is estimated to be 12 kΩ. The collector resistor of 100 Ω is meant to limit the current across the LED, in case of a current peak. The transistor NPN is chosen as the cheapest without bias resistors in the stock of Farnell [10]. The push-buttons, provided by the department, are 6 mm by 6 mm surface mounted.

Table 4.5: Components of the cape for the BBB

Symbol	Component	Farnell code	Manufacturer
R_1	Resistor 825 Ω	2117153	TE Connectivity
R_2	Resistor 464 Ω	2117126RL	TE Connectivity
$R_{pull-down}$	Pull-down resistor 10 kΩ	9330399	Multicomp
R_B	Base resistor 12 kΩ	9238611RL	Multicomp
R_C and R_3	Collector resistor 100 Ω	1358015	Multicomp
LED	LED T-1 5mm bulb	1142572	Kingbrigth
Q	Transistor NPN 2N5551	9846751	Fairchild Semiconductor

EAGLE 6.6.0 software is used for the design of the PCB. First, the schematic of the circuit needs to be defined as depicted in Figure A.1 in Appendix A. Then the design is transferred to the board design tool, where the final location of the components, tracks and borders of the board are defined. The tracks that should support an amperage of 250 mA are 0.6096 mm (0.024") width, while the rest are

0.000 39 mm (0.01"). The final design is shown in Figure 4.14, where the red tracks correspond to the upper layer and the blue ones are located at the bottom layer. Detailed figures of the two layers can be found in Appendix A, Figures A.2 and A.3. [6]

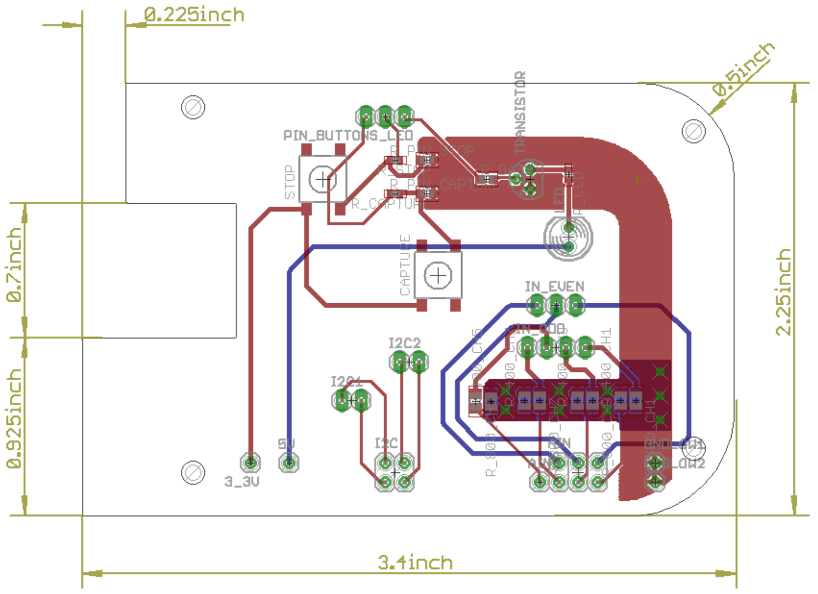
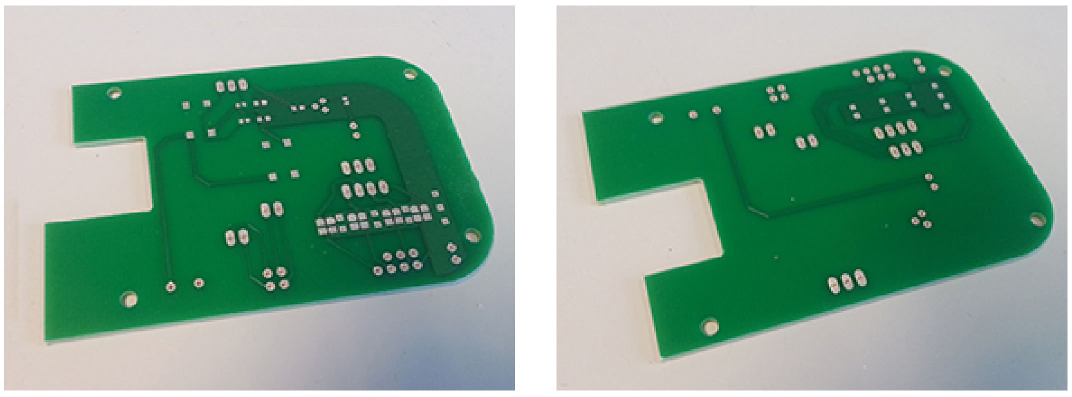


Figure 4.14: PCB design of the cape in EAGLE 6.6.0

In Figure 4.15 it can be seen the PCB manufactured for the cape without components. Figure (a) shows the upper layer, while in Figure (b) the lower layer is displayed.



(a) (b)

Figure 4.15: PCB of the cape: (a) upper layer, (b) lower layer

Figure 4.16 shows the CI-BBB-cape with the components soldered on the upper layer (a) and the lower layer (b).

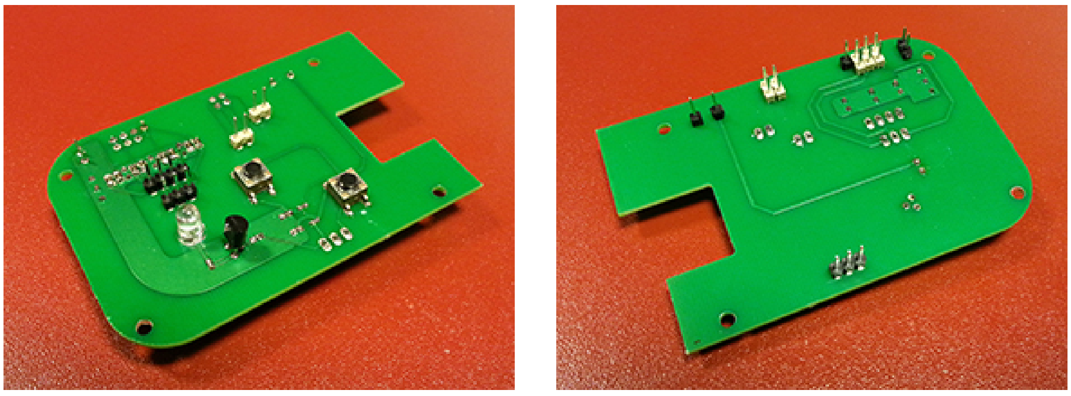

(a) (b)

Figure 4.16: CI-BBB-cape: (a) upper layer, (b) lower layer

4.3 Software: library and remote control

The software library (LibCI-BBB) designed for this thesis work, aims to provide tools to the user to control the CI-BBB. The functions implemented interact with the hardware of the BBB for fetching data coming from the sensors of the MP. The library LibCI_BBB includes all the routines described in this Section, so it can be integrated in the program of the MP. This library features the capability to capture analogue signals through the A/D converter, and communicate using the I^{2}C bus communication interface. Finally, to ease the configuration of the measurement tasks and the communication with the board, a client/server architecture is provided as a remote control of the CI-BBB, which includes the functionality to send the generated file to an external storage.

The PRU-ICSS subsystem was considered from the very best moment to implement the control functions of the A/D converter and the I^{2}C bus, and the EDMA3 subsystem to support the transmission of data and improve the overall performance. The configuration of the A/D converter from the PRU-ICSS was achieved, and the performance of the PRU processors was as expected. Still, this solution was discarded in favor of the library LibPruIo (see Section 3.2.2), due to problems with the interaction between the A/D converter, the EDMA3 subsystem and the PRU processors.

The library described in this Section can be found in the GitHub software repository, in the following URL: `https://github.com/jpdefrutos/CI-BBB`

4.3.1 The Communication Interface Library

Figure 4.17 depicts the architecture of the library designed for this thesis work. The Communication Interface Library (LibCI_BBB), is divided in three categories: the specific A/D converter routines, the I^{2}C functions and the common instructions. The latter functions are used either with A/D converter or the I^{2}C bus. The data type `_CI_parameters`, is a structure shared by all the functions which contains the configuration parameters of the CI. It must be initialized at the beginning, with the required data according to the desired function.

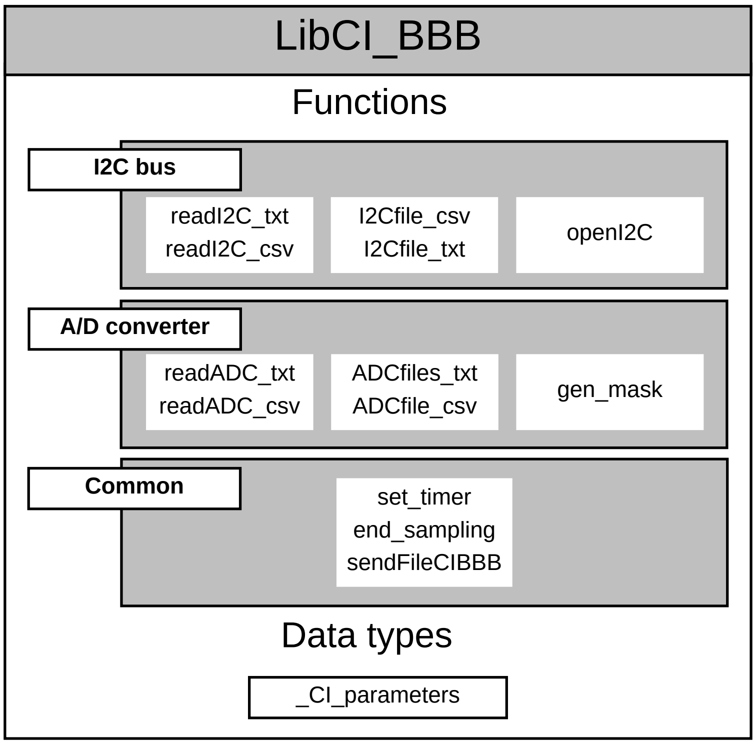


Figure 4.17: Architecture of the library

The data provided by the sensors is compiled in a file, along with additional information e.g., the date and capture time. The implementation of the library gives support for text (TXT) and comma separated value file formats (CSV). All files include a heading with the starting time of the capture process. In the case of the A/D converter, one TXT file is generated per selected input channel. While in the CSV file, all the samples of the enabled input channels are stored in the same row separated by comas. In the case of the I^{2}C bus, the output file includes all the information the magnetic encoder can provide, displayed in the same order as shown in [41]. Either TXT or CSV files present the values arranged in a row separated by commas.

LibCI_BBB supports two different operation modes: automatic and manual.

- **Automatic mode:** the routine fetches new data within a loop, controlled by the handler function `end_sampling` of the alarm interruption. The capture time introduced by the user is used by the instruction `set_timer`, to program an alarm signal to stop the execution of the program.
- **Manual mode:** this mode is designed to sample data only when the *Capture button* is pressed. Following the flowchart in Figure B.6, the program keeps polling the state of the control buttons to determine the next step of the program. By pressing the *Stop button*, the program exits the reading loop.

Analogue to digital converter routines

In the first approach of the read instructions, two alternating buffers were used to store the data from the A/D converter, before being saved in the output file. A thread launched by the main process was used to empty these buffers, and copy the content to the output files. After the first measurement test, done with an input sine signal of 1 kHz, 0.79 V of amplitude and 0.41 V of offset, a large amount of missing data was detected as shown in Figure 4.18. Because the microprocessor can only execute one instruction at the same time, during the execution of the thread some valid samples were not saved in the buffers, and thus, overwritten by new samples.

Two possible solutions were proposed: modify the priority level of the processes in the schedule table of the microprocessor to prioritize the fetch of new samples over the thread execution, or eliminate the buffers and copy the samples directly in the output file. Because there are nearly 400 cycles of the microprocessor clock, between each time a sample of the A/D converter is ready, the second option was finally implemented. Figure 4.19 shows the samples produced with the same input signal as in Figure 4.18, but without the use of the alternating buffers. It can be seen that there is no missing codes as in the second case.

The A/D converter functions can be classified in three groups: routines related to the output files, functions used to read values from the A/D converter and support commands. To reduce the execution time of the two first groups, each file format has its specific routine. The `ADC_file` functions open and configure the output file. The `ADCfiles_txt` function returns as many file descriptors as channels enabled, stored in the array `dstfile[]`. The `ADCfile_csv` returns the variable `dstfilecsv` with a file descriptor of the CSV output file. Both functions also save the file routes in the array `sendFile[]`.

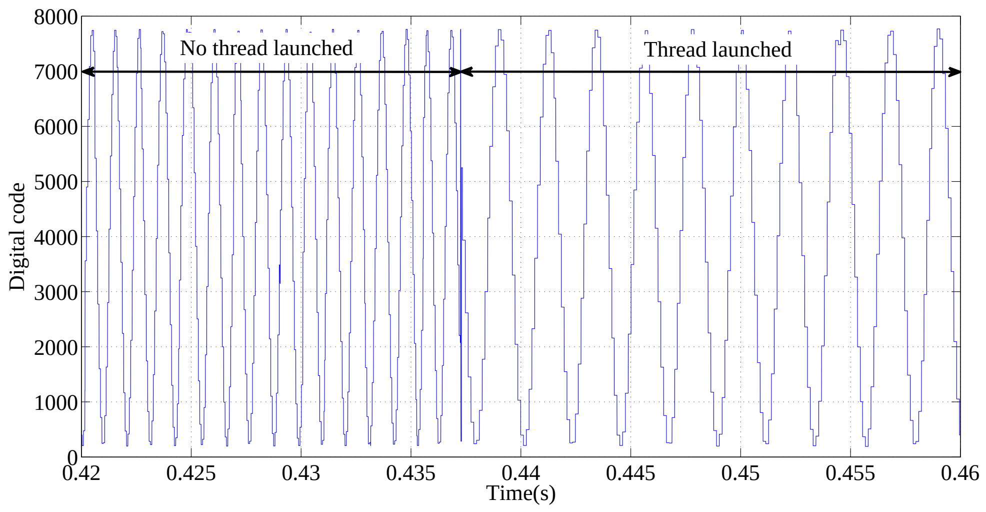


Figure 4.18: Sampled sinusoidal waveform of 1 kHz using buffers

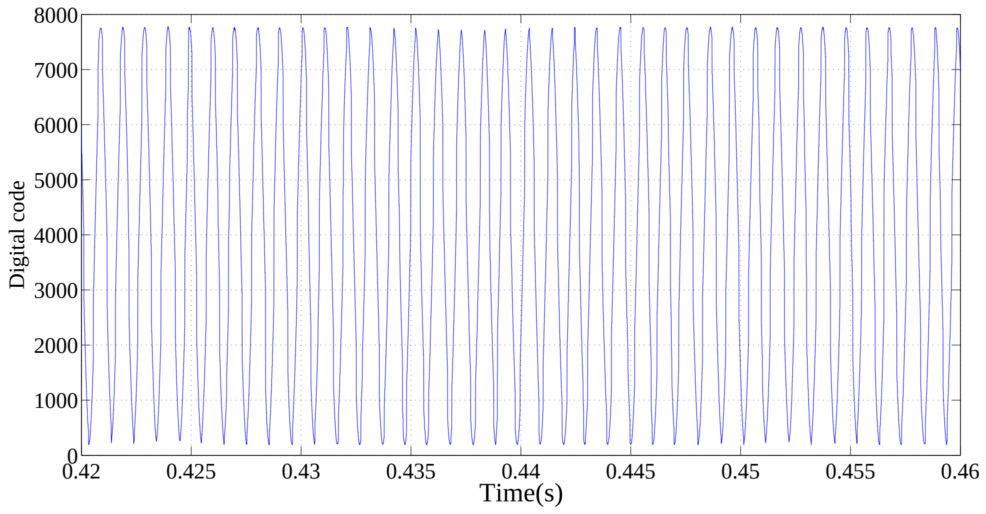


Figure 4.19: Sampled sinusoidal waveform of 1 kHz without using buffers

The `readADC` routines copy data from the array `Values[]`, which maps the DRAM memory of the PRU processor, where the samples of the A/D converter are stored (see Section 3.2.2), into the output file. The flowchart of these routines can be found in Figures B.1 and B.2. By default, the sample rate is, 160 kHz. But, this rate decreases with the increase of input channels, as explained in Section 3.2.2.

A PRU-ICSS driver is assumed to be loaded, before calling the read instructions. The driver is created using the function `pruio_new` and loaded to the kernel with the instruction `pruio_config()` provided by the library LibPruIo. This structure is initialized with parameters to control the sampling rate of the A/D converter, the PRU processor to use and a configuration for the *Step Configuration registers* (see

Section 3.2.1). The read instruction can sample the seven channels during the same program execution. So, `gen_mask` is used to select which channels to sample.

Inter-Integrated Circuit functions

Section 2.3.2 presents the I^{2}C communication bus as a communication network to which different devices can be connected and share information from point to point.

The LibCI_BBB functions within this group are complemented using the commands of the *i2c-tools* package of Linux. The command `i2c-detect` displays the address of the devices connected to a particular I^{2}C module. This information is asked to the user when working with the Remote Control software presented in Section 4.3.2. Regarding the new functions implemented in the library LibCI_BBB, there are two routines available for each file format `readI2C\_` and `I2Cfile\_`, as in the case of the A/D function. Figure B.3, in Appendix B, depicts the flowchart of the read routines for the I^{2}C sensor. The `openI2C` instruction returns a file descriptor to the I^{2}C system file, used to read data sent by the slave I^{2}C device connected to the bus.

Common use instructions

`set_timer`, `end_sampling` and `sendFileCIBBB` are routines that are not bounded to an specific hardware, such as `gen_mask` or `openI2C`. As explained before, `set_timer` and `end_sampling` are use in the automatic mode, configuring an alarm and stopping the capture data process, respectively.

`sendFileCIBBB` sends a file saved in the directory `/root/Documents` of the file system, whose name is specified in the array `sendFile[]` of the _CI_parameters structure shown in Program 4.1. The document is sent through a Transmission Control Protocol (TCP) socket (see Section 4.3.2)using the system call `sendfile()`.

_CI_parameters data type

This structure aims to bring together all the parameters required to execute the different functions of the LibCI_BBB. The definition of the structure variables is as shown in Program 4.1 and Program 4.2. Like the library functions, the variables are divided according to the scope of application. The variables that need to be initialized by the user are marked with the tag `User`. The common variables displayed in Program 4.1 need to be initialized, regardless of the hardware that is going to be used.

```
struct CI_parameters{
//Common variables
FILE *dstfilecsv;
FILE *dstfile[7]; //One TXT file per channel, 7 at the most
char sendFile[7][100]; //Addresses of used files
sigset_t *signalSet;
PruIo *driver;
unsigned int samplingTime;        //<-- User
struct sigaction samplingAlarm;
int mode; //Manual or automatic    <-- User
int fileSelection;                //<-- User
time_t usect0, usect1, sect0, sect1;
duble totalTime;
int firstTime;
```

Program 4.1: Common variables of the _CI_parameters data type

Besides, the specific variables listed in Program 4.2 should be set when the corresponding device is aimed to be used. Values for variables `avg`, `clkDiv`, `mask`, `OpDelay` and `SamDelay` are proposed in the implementation of the Remote control software (Section 4.3.2), to reach the maximum sampling rate of the A/D converter.

```
//ADC specific
int Nchannels;                    //<-- User
int channels[7];                  //<-- User
int avg, clkDiv;                  //<-- User
int mask;                         //<-- User
int OpDelay, SampDelay;           //<-- User

//I2C specific
FILE *dstfileI2C;
int i2cFile;
int selectedDev;                  //<-- User
char devAddress[10];              //<-- User
char *ptrdevAdd;
char *i2cName;
} typedef _CI_parameters;
```

Program 4.2: Specific variables of the _CI_parameters data type

4.3.2 Remote Control software

A remote control software is provided to ensure the portability and integration in the MP system. For that purpose, a one-to-one client/server architecture is used between the remote computer and the BBB. The latter device runs the server program, while the computer executes the client software. The detailed flow chart of the Remote Control can be found in Figures B.4, B.5 and B.6, in Appendix B. Appendix C includes a User Manual the Remote Control software.

There are two main transmission protocols, the TCP, and the User Datagram Protocol (UDP). TCP focuses on the reliability of the communication, ensuring the arrival of data to the destination and preserving the sequence of the information, in behalf of transmission speed. Besides, the UDP protocol favors the transmission speed by packaging the information into datagrams, and sending them to the receiver, without the need of an open connection, unlike TCP. But, there is no guarantee that the transmission order is preserved, nor that all the packages arrive to the destination. [31]

The reliability of the transmission is a priority for the Remote Control program, beyond the time required to complete the transfer of data. That is the reason why TCP protocol is chosen for this program. After connecting the client to the socket created by the server, continuous transmissions of data is done between the server and the client. When the client closes the connection, the server can also close the socket, or wait for a new client. This process is illustrated in Figure 4.20.

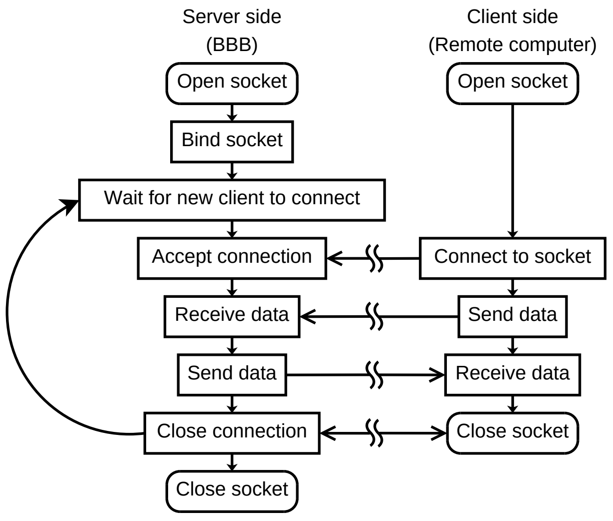


Figure 4.20: TCP socket architecture

The client program includes an user interface through the Linux terminal. The communication is by means of an USB-net, created when the BBB is connected to the remote computer through the USB port. This network behaves as an Ethernet network, but supported through an USB connection. By default, the Internet Protocol (IP) (version 4) of the BBB is 192.168.7.2, and the communication port used is 5002. However, the software allows the user to change this information, when executing the programs. To ensure the proper operation of the Remote Control, the server must be started before the client.

The server program is in charge of configuring the data structure `CI_parameters`, using backup files stored in the local memory of the BBB, or reading new parameters provided by the user through the client program. A pointer to the instantiation of the configuration structure is used to pass the information to the different routines. This results in less memory consumption and avoids redundancy of variables with the same information.

After the structure is initialized, the server opens the corresponding output file with `prepare_files_txt` or `prepare_file_csv` functions and executes either `read_adc` or `read_i2c` routines. Once the capture is completed, the program allows the possibility to add new measurements to the same file or exit the program. Regardless of the decision, the server sends the generated file to the client and saves a copy in the internal storage after finishing the capture process.

4.4 Performance measurements

Before installing the CI-BBB in the MP, it is worth giving results that validate the performance of the device. In the first Section, an analysis of the A/D converter provided by the BBB is presented i.e., a sampling frequency analysis, to make sure the converter can sample the force sensor output. The following Section provides results of the performance of the cape Next Section relates to a robustness test done to the CI-BBB, letting it work for long period of time continuously.

4.4.1 A/D converter analysis

The CI-BBB is going to be used for sampling the information provided by the force sensor (see Section 3.1.1), thus the minimum frequency the A/D converter must be able to sample correctly is 1000 Hz. For this test, the maximum sample rate that can be configured using LibPruIo is used i.e., zero open delay, 1 clock cycle for the sample delay (minimum possible according to [22]) and one input channel enabled. With this configuration the sample rate of the A/D converter is 160 kHz, with the A/D converter running at 2.4 MHz, according to [14].

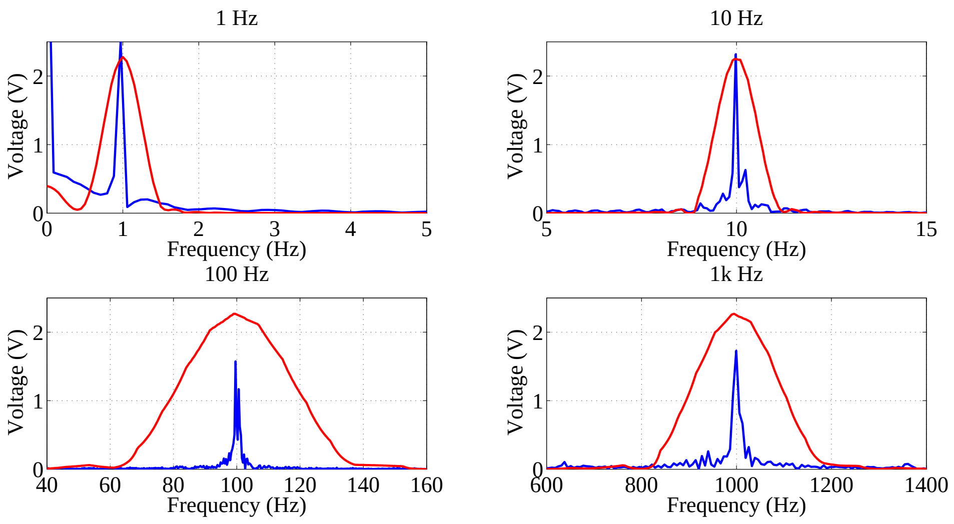


Figure 4.21: FFT of the input signal (red) and the sampled waveform by the A/D converter (blue)

These measurements were made using the channel 1 of the CI-BBB-cape. An input sine waveforms of 2.5 V of amplitude and 1.25 V of offset, so. The sine signal was generated using the function generator presented in Section 4.1.3. To make sure the input voltage level would not exceed the 1.8 V limit, the oscilloscope was used to monitor the input signal and adjust when needed. Figure 4.21 depicts the FFT of the input signal measured with the oscilloscope in red colour, and the FFT obtained by the sampled data provided by the A/D converter. It can be seen that the A/D converter is capable of sampling up to 1 kHz, as required for the force sensor of the MP (see Section 3.1.1).

Data loss was detected in the sampled analogue signals using the A/D converter. Two different types of missing codes were observed, depending on the source of the problem: due to the A/D converter module, or due to the LibCI-BBB software. The first type, named as type I in Figure 4.22, includes occasional missing samples because the A/D converter system is not prepared to proceed with the next sample. This might be because the new *Step* configuration is being loaded at the moment. This error is more often in high frequency input signals.

The missing codes related to the LibCI-BBB software, or type II according to Figure 4.22, is due to the fact that, the actual implementation of the software, runs the reading functions of the A/D converter in the *user space*. This means, the process has a CPU time limit, after which the process is thrown out of the processor so another process of higher priority can be attended. In the actual implementation of the Remote Control program, the priority of the process is configured to be the highest possible (*nice* value of −20). One possible solution could be to program a kernel module to execute the reading instructions of the A/D converter. Because the programs executed in *kernel space* are not subject to the resources and time restrictions of the *user space*, this approach might avoid the type II missing codes. This task is out of scope of the present thesis work, thus it is left as a future improvement for the CI-BBB (see Section 5). Unlike the first type, type II missing codes may appear regardless of the characteristics of the input signal. Figure 4.22 shows an example of a sampled sine signal of 10 kHz which presents the missing codes of types I and II. The red dots represent the sample points generated by the A/D converter, while the blues dashed line is added to ease the interpretation of the waveform.

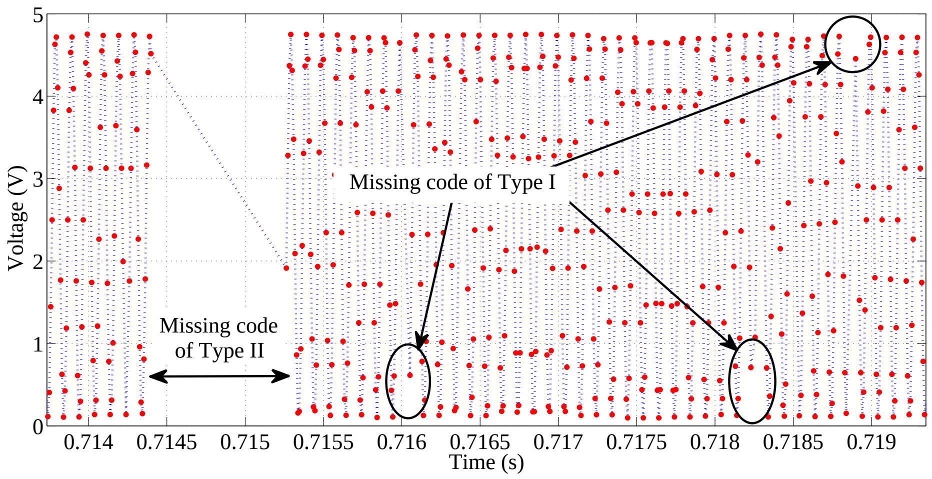


Figure 4.22: Sampled sine waveform of 100 Hz

4.4.2 Long-term test

Due to the lack of necessary software for sending in a continuous way the output file and the free memory space available in the BBB, the files generated with samples of the A/D converter or the I^2C bus are limited to nearly 1.2 GB.

After a working period of 4 h sampling an input signal of 2.5 V of amplitude, 1.25 V of offset and 1 Hz frequency, generated by the function generator AFG-2005 of Gw Instek, the microprocessor reached a very high temperature. To avoid any potential risk due to the lack of thermal protection of the board, a fan was used to cool down the platform. Additional legs were added to ease the air flow across the bottom of the platform, avoiding overheating of the BBB. Apart from this issue, missing codes of type II (see Section 4.4.1) were detected. Some of them reaching almost one second of missing data. No more unforeseen events happened, such as drift or waveform distortion.

4.4.3 Final implementation of the system

Figure 4.23 and 4.24, illustrates the outcome of the CI-BBB when sampling the outputs of the force sensor and the magnetic encoder, respectively. For the force sensor test, the CI-BBB was connected in parallel to the DAQ system PCI-6259 of National Instruments, so the same signal was supplied to both devices. The blue line corresponds to the sampled data from the CI-BBB, while the outcome of the PCI-6259 DAQ is colored in red. The PCI-6259 was configured to a sample frequency of 200 Hz. The I^{2}C test was done with the magnetic encoder (3.1.2), powered directly using the 5 V and GND pins of the BBB. The magnetic strip was moved manually over the sensor, with resulted in a not smooth ramp waveform.

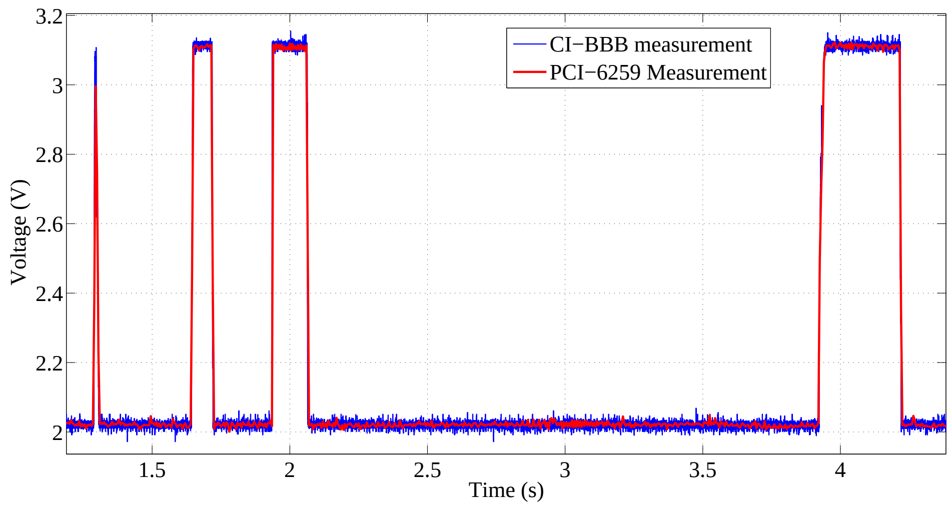


Figure 4.23: Test with the force sensor

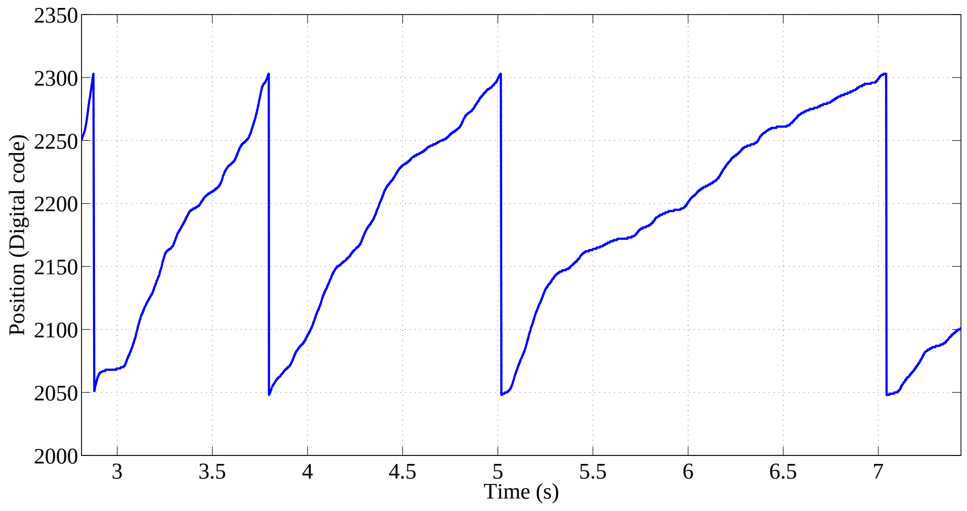


Figure 4.24: Test with the magnetic encoder

As it can be seen, the CI-BBB is able to properly sample the force sensor output signal. And to read the information provided by the magnetic encoder through the I^{2}C bus.

Figure 4.25 shows the final aspect of the CI-BBB, with the cape designed for this thesis work. The connection of the sensors to the boards is easily achieved by means of the pins mounted on the CI-BBB-cape. The two buttons are easily reached for launching the sampling process, and the state LED lights up whenever new data is being captured by the board, either by the A/D converter o the I^{2}C module.

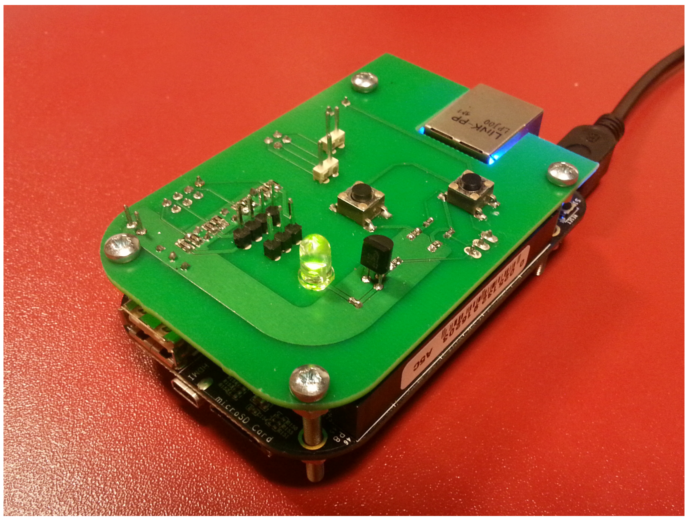


Figure 4.25: CI-BBB device

5. CONCLUSIONS AND FUTURE WORK

The project presented in this document presents a communication interface to handle analogue signals and I^{2}C communication bus, named CI-BBB. The scope of the system is the force sensor and magnetic encoder of the Microrobotic Platform, build on BeagleBone Black electronic prototyping platform. The work is divided in two main sections: the additional hardware to support the features of the device, and the software library which includes all the commands and instructions to manage the CI-BBB.

First, a cape board (CI-BBB-cape) was implemented for the BBB. The additional VA circuit was designed, and tested using prototypes, to guarantee the correct voltage level of the input signals of the BBB. The VA circuit chosen for this application consists on a voltage divider. The ease of assembly and low number of components favored the choice of this circuit beyond the OA proposal. The VA circuit and further components were also implemented in the to support the software of the CI-BBB i.e., the control buttons and the state LED.

Next, a software library was programmed to encompass all the routines used to control the system, and thus, ensure the integration and portability of the CI-BBB, in the control software of the MP. In the first approach, the PRU-ICSS unit was programmed directly by means of the AM335x PRU Package. But, finally the library LibPruIo was used to implement this Section of the CI-BBB, controlling the PRU-ICSS subsystem through the routines implemented in the library.

The functions programmed for the CI-BBB, are divided in three categories: related to the A/D converter, to the I^{2}C and routines of common use. The hardware specific functions take care of writing the data supplied by the sensors, into TXT or CSV files. On the other hand, the common instructions give support to these routines. A client/server architecture as Remote Control of the BBB is presented, to allow the control of the platform from a remote computer and send the generated files to an external storage device

The overall performance of the CI-BBB system meets the proposed objectives. The system can sample analogue voltage signals and communicate through the I^{2}C bus, by using the instructions implemented int the LibCI-BBB library. The Remote Control program enhances the integration in the MP system, without loosing the control over the system. Measurements are provided in Section 4.4.1 to prove that the A/D converter is able to handle the output signal of the force sensor, which can work up to 1 kHz. The I^{2}C communication works as expected. The platform is able to request new data to the slave device, and to copy it into de output file. Only punctual missing code errors were detected when sampling with the A/D converter. Nevertheless, after minor improvements proposed already in the future works, the CI-BBB will be ready to be installed in the MP and substitute the actual DAQ system.

The technical specifications of the CI-BBB are:

- 7-channels 12-bits SAR A/D converter, with a theoretic maximum sample rate of 160 kHz.
- 4 input channels of 5 V signals, and 3 input channels for 1.8 V signals.
- 2 I^{2}C communication bus modules, configurable to 100 kbit/s or 400 kbit/s.
- Automatic and manual operation modes.
- Remote Control software with TCP/IP connection over USB-net.
- LibCI-BBB Library software which includes all the routines.

In the following lines, various proposals are recorded to be considered for future works, towards improving the performance and efficiency of the Communication Interface:

- **Add an external A/D converter:** the word length of the A/D converter provided within the BeagleBone Black, is not enough to achieve the resolution of the actual force sensor. Instead, a 16-bit A/D converter should be sufficient to achieve the maximum resolution and decrease the lose of information.
- **Connection to Ethernet:** The common procedures used in Ubuntu OS to configure an Ethernet net were not sufficient to stablish a connection with the BBB through Ethernet, and so the USB-net was used instead. The possibility of connecting the CI-BBB to an Ethernet net would improve the performance of the platform, eliminating the need of an USB connection to the computer of the user.

- **Improve the library LibCI-BBB:** the present solution is a first approach to what the control software should be able to do. Hence, this code needs to be optimized, and there is plenty of room for improvements e.g., implement the reading instructions with kernel modules. Enhancing the performance of the software will improve the efficiency of the whole system.

- **Improve the file management:** actually, the maximum size of the generated files is limited by the internal memory of the BBB. Thus, a routine to dynamically send the information of the sensors to the external storage, would improve the maximum sampling time the CI-BBB can work. An alternative solution to this could be to add an external storage device, through the USB ports of the platform.

- **New communication protocols:** as a communication interface, one of the main points to consider is the number of communication protocols it can understand. As a first approach, the actual implementation only supports two types. However, a rise of the number of protocols that the system can handle, would derive in an improvement on the features of the CI.

- **Implement Stream Control Transmission Protocol (SCTP) server:** on 2000 the SCTP protocol has become an alternative to the TCP and UDP, combining the most relevant features of both protocols. Within the scope of this thesis, the most important aspects of SCTP protocol to consider are the reliability of the transmission, and the use of data in packages to improve transmission speed. Additionally it support multi-streaming i.e., deliver data through different streams to the same destination; and multi-homing i.e., broadcast information to multiple IP addresses from the same source [29]. Thus, taking into account these features it might be interesting to implement this protocol instead of the actual TCP client/server, improving the performance of the Remote Control software.

REFERENCES

[1] Aparicio, J. L. Lecture notes of the course 1172 - Electrónica y Regulación Automática. Escuela Técnica Superior de Ingenieros Industriales de Madrid, Universidad Politécnica de Madrid, 2012.

[2] Arduino. Arduino website. `http://www.arduino.cc/`. Last checked: July 2014.

[3] BeagleBoard.org. Programmable real-time unit and industrial communication sub-system (pru-icss) overview. `http://github.com/beagleboard/am335x_pru_package/tree/master/Documentation`. Last checked: March 2014.

[4] Bolton, W. *Mechatronics: a multidisciplinary approach.* Pearson Prentice Hall, 2008.

[5] Brian DeMuth, D. E. *Designing Embedded Internet Devices.* Elsevier, 2002.

[6] CadSoft. Cadsoft eagle pcb design software / eagle pcb software. `http://www.cadsoftusa.com/eagle-pcb-design-software/product-overview/`. Last checked: June 2014.

[7] Coley, G. *BeagleBone Black System Reference Manual*, October 2013.

[8] Cooper, J. Introduction to the beaglebone black device tree. `https://learn.adafruit.com/downloads/pdf/introduction-to-the-beaglebone-black-device-tree.pdf`. Last checked: July 2014.

[9] Datko, J. Beaglebone black i2c references. `http://datko.net/2013/11/03/bbb_i2c/`. Last checked: July 2014.

[10] element 14. Farnell element 14. Last checked: June 2014.

[11] Foundation, B. Beagleboard.org. `http://www.beagleboard.org/`. Last checked: July 2014.

[12] Foundation, R. P. Raspberry pi website. `http://www.raspberrypi.org/`. Last checked: July 2014.

[13] Franco, S. *Design With Operational Amplifiers And Analog Integrated Circuits.* McGraw-Hill Education (India) Pvt Limited, 2002.

[14] FREIHERR, T. Libpruio 0.0 - am33xx-pru driver for digital input / output and analog input. `http://users.freebasic-portal.de/tjf/Projekte/libpruio/doc/html/`. Last checked: May 2014.

[15] HOLDINGS, A. Arm1176 processor. `http://www.arm.com/products/processors/classic/arm11/arm1176.php`. Last checked: July 2014.

[16] HOLDINGS, A. Cortex-a8 processor. `http://arm.com/products/processors/cortex-a/cortex-a8.php`. Last checked: July 2014.

[17] INCORPORATED, D. *TL072 Low Noise JFET Input Operational Amplifiers*, May 2009.

[18] INSTRUMENTS, T. Pru software development package. `http://www.ti.com/tool/sprc940`. Last checked: March 2014.

[19] INSTRUMENTS, T. *μA741x General-Purpose Operational Amplifiers*, November 1970.

[20] INSTRUMENTS, T. *LM741 Operational Amplifier*, May 1998.

[21] INSTRUMENTS, T. *AM335x PRU-ICSS Reference Guide*, May 2012.

[22] INSTRUMENTS, T. *AM335x ARM® Cortex^{TM}-A8 Microprocessors (MPUs): Technical Reference Manual*, December 2013.

[23] INSTRUMENTS, T. *Sitara ^{TM}AM335x ARM Cortex-A8 Microprocessors (MPUs)*, April 2013.

[24] INTEGRATED, M. Understanding SAR ADCs: Their architecture and comparison with other ADCs. `http://www.maximintegrated.com/app-notes/index.mvp/id/1080`. Last checked: March 2014.

[25] JEAN-MARC IRAZABAL, S. B. *I^2C manual*, 2003.

[26] JOHN PARK, S. M. *Practical Data Acquisition for Instrumentation and Control Systems*. Newnes, 2003.

[27] KALLIO, P. Fibam autonomous microrobotic system for manipulation, stimulation and characterization of fibrous materials. `http://www.tut.fi/en/about-tut/departments/automation-science-and-engineering/research/projects/fibam/index.htm`. Last checked: June 2014.

[28] KINGBRIGHT. *T-1 3/4 (5mm) SOLID STATE LAMP*, 2006.

[29] L. Ong, J. Y. Rfc 3286 - an introduction to the stream control transmission protocol (sctp). `http://tools.ietf.org/html/rfc3286`. Last checked: June 2014.

[30] Lundberg, K. H. Analog-to-digital converter testing. Available at: `http://www.mit.edu/~klund/papers/UNP_A2Dtest.pdf`, 2002.

[31] M. Mitchell, J. O., and Samuel, A. *Advanced Linux Programming.* New Riders Publishing, 2001.

[32] Malvino, A., and Bates, D. *Electronic Principles.* Revision of a classic. McGraw-Hill/Higher Education, 2007.

[33] Microchip. *MCP6241/1R/1U/2/4 50µA, 550kHz Rail-to-Rail Op Amp*, 2008.

[34] Miguel, P. *Electrónica general: equipos electrónicos de consumo.* Electricidad-Electrónica. Thomson-Paraninfo, 2003.

[35] P. Saketi, A. Treimanis, P. F. P. R., and Kallio, P. Microrobotics platform for manipulation and flexibility measurement of individual paper fibers. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009).

[36] Pons, J. L. *Emerging Actuator Technologies. A Michomechatronic Approach.* John Wiley Sons Ltd, 2005.

[37] Saketi, P. Microrobotics platform for manipulation and flexibility measurement of individual paper fibers. Master's Thesis, Tampere University of Technology. Publication - Tampere University of Technology, 2009.

[38] Semiconductor, F. *2N5551 / MMBT5551 NPN General Purpose Amplifier*, 2009.

[39] Semiconductors, N. *LM741 Operational Amplifier*, August 2000.

[40] Semiconductors, P. *The I^2C-Bus Specification Version 2.1*, 2000.

[41] Technologies, N. S. *NSE-5310 Miniature Position Encoder with Zero Reference and I^2C Output*, 2013.

[42] telos Systementwicklung GmbH. I^{2}c bus. `http://www.i2c-bus.org/`. Last checked: April 2014.

[43] Tools, F. Ft-s microforce sensing probe. `http://www.femtotools.com/index.php?id=products-s`. Last checked: June 2014.

[44] Tools, F. *FT-S10000 Microforce Sensing Probe.*

[45] Unknown. Interfacing with i2c devices. `http://elinux.org/Interfacing_with_I2C_Devices`. Last checked: June 2014.

[46] v. Essen, M. Control software for microrobotic platform. Master's Thesis, Tampere University of Technology. Publication - Tampere University of Technology, 2010.

[47] Walden, R. H. Analog-to-digital converter survey and analysis. *IEEE Journal on Selected Areas in Communication* (1999).

APPENDIX A: SCHEMATIC OF THE CAPE FOR THE BEAGLEBONE BLACK IN EAGLE 6.6.0

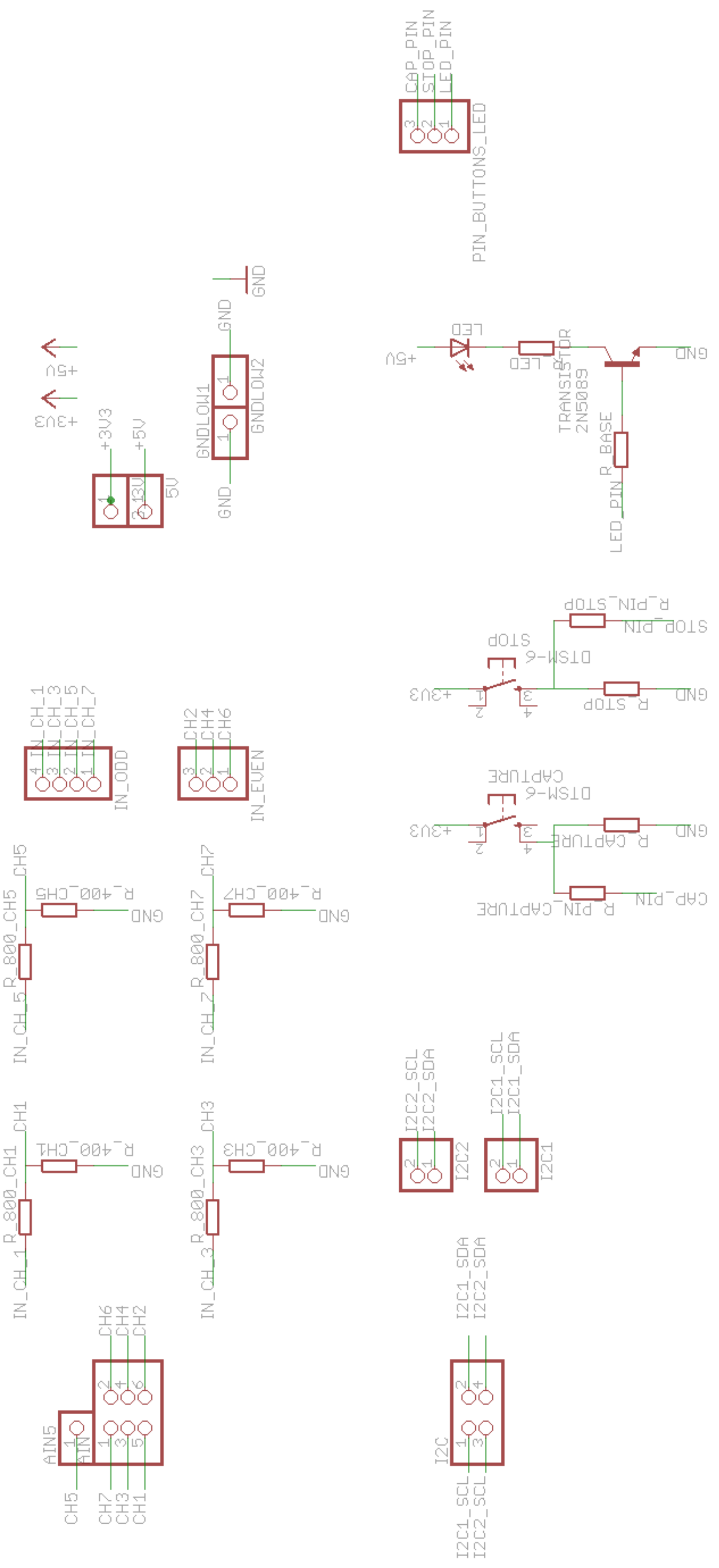


Figure A.1: Schematic of the circuit of the cape designed for the BeagleBone Black in EAGLE 6.6.0

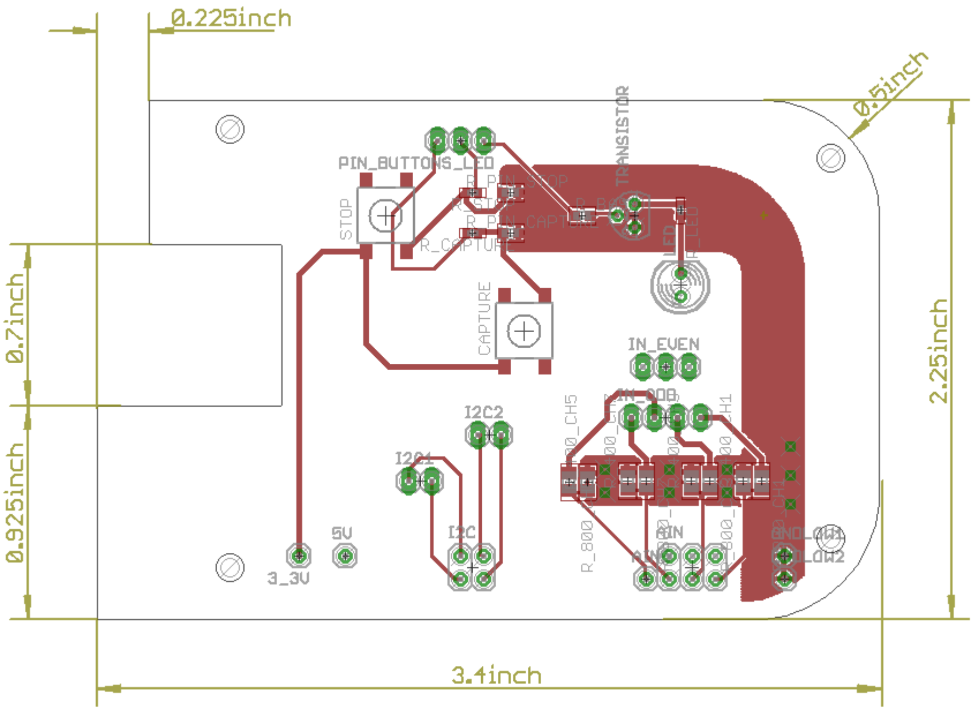


Figure A.2: Top layer of the PCB in EAGLE 6.6.0

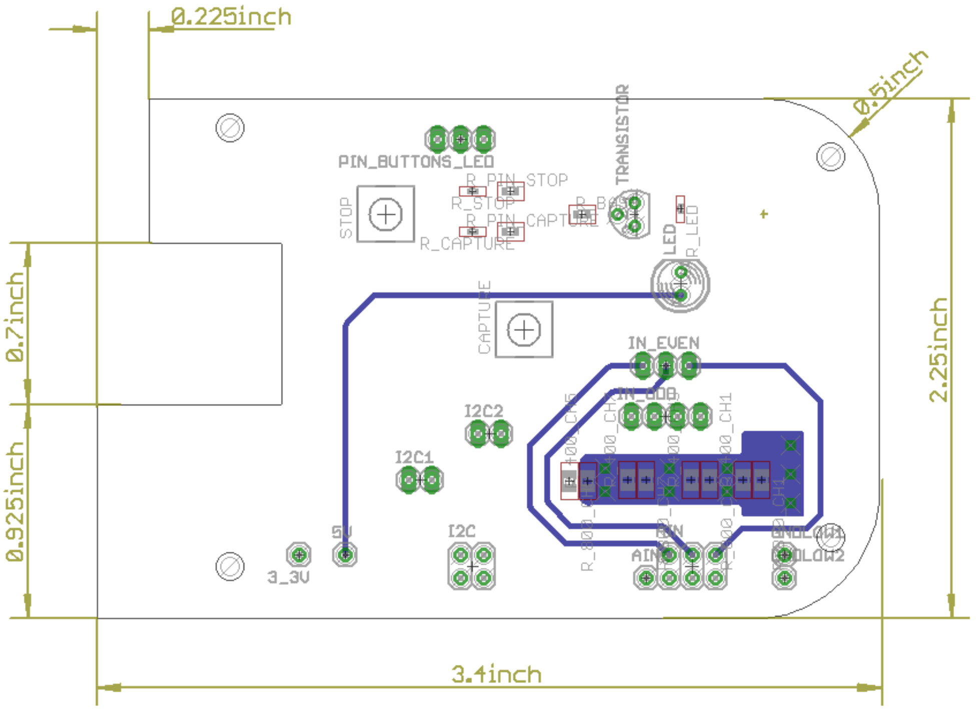


Figure A.3: Bottom layer of the PCB in EAGLE 6.6.0

APPENDIX B: FLOWCHART DIAGRAMS

A/D converter function flowchart

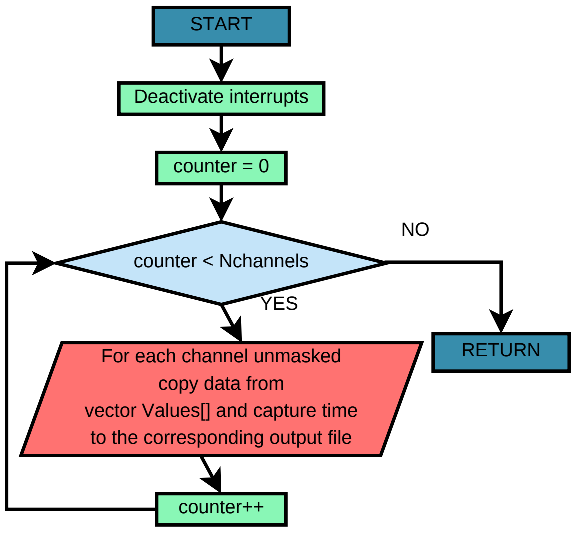

START
Deactivate interrupts
counter = 0
counter < Nchannels
NO
YES
RETURN
For each channel unmasked copy data from vector Values[] and capture time to the corresponding output file
counter++

Figure B.1: Flowchart of the A/D converter read function for TXT file

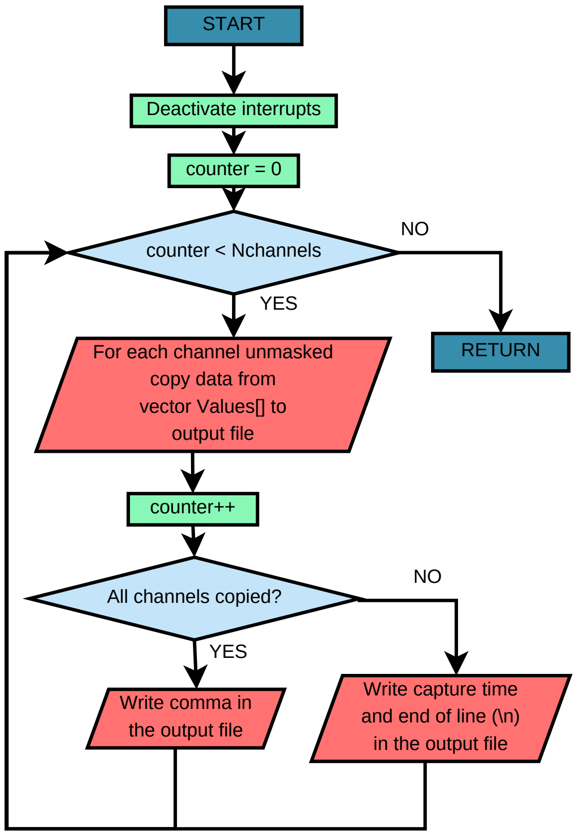

START
Deactivate interrupts
counter = 0
counter < Nchannels
NO
YES
RETURN
For each channel unmasked copy data from vector Values[] to output file
counter++
All channels copied?
NO
YES
Write comma in the output file
Write capture time and end of line (\n) in the output file

Figure B.2: Flowchart of the A/D converter read function for CSV file

I^{2}C function flowchart

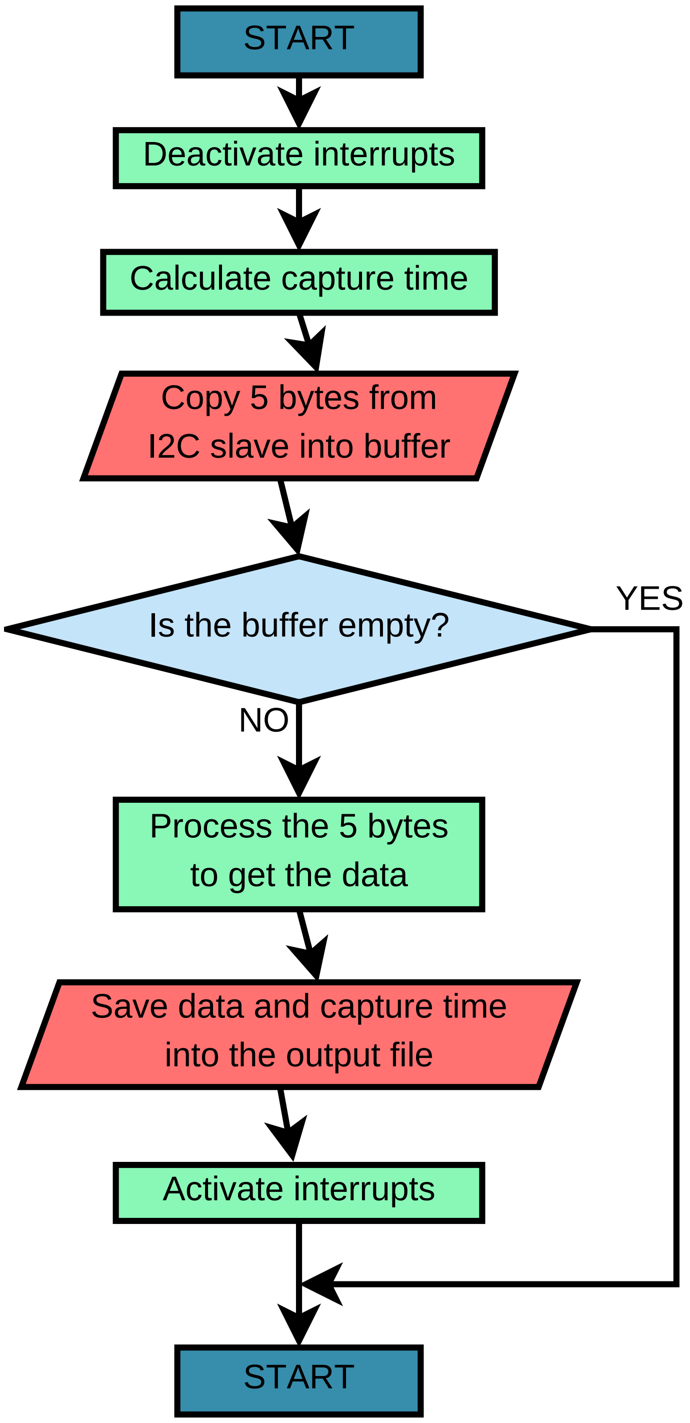

START
Deactivate interrupts
Calculate capture time
Copy 5 bytes from I2C slave into buffer
Is the buffer empty?
YES
NO
Process the 5 bytes to get the data
Save data and capture time into the output file
Activate interrupts
START

Figure B.3: Flowchart of the I^{2}C function

Remote control program flowchart

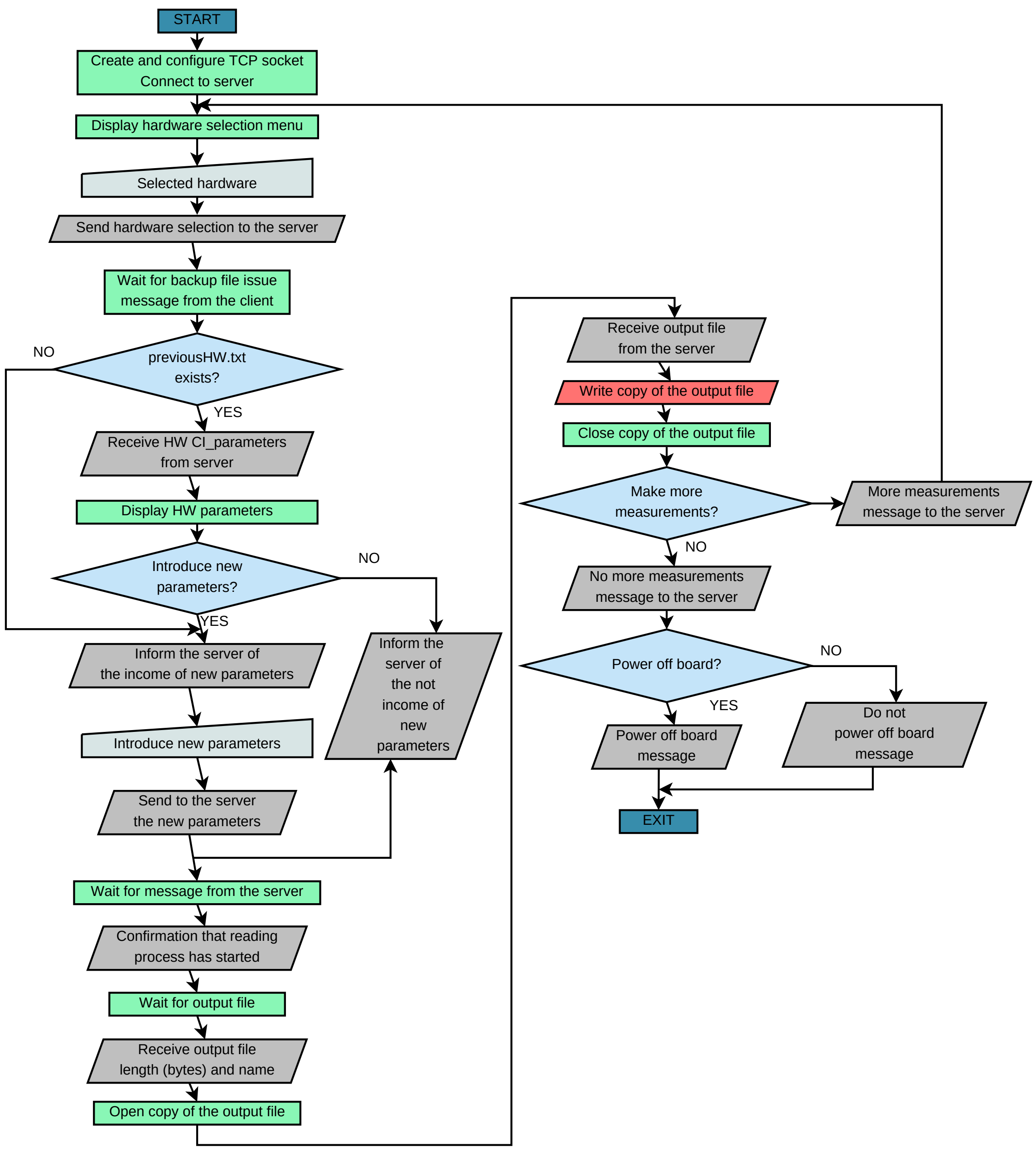

START
Create and configure TCP socket
Connect to server
Display hardware selection menu
Selected hardware
Send hardware selection to the server
Wait for backup file issue message from the client
previousHW.txt exists?
NO
YES
Receive HW CI_parameters from server
Display HW parameters
Introduce new parameters?
NO
YES
Inform the server of the income of new parameters
Inform the server of the not income of new parameters
Introduce new parameters
Send to the server the new parameters
Wait for message from the server
Confirmation that reading process has started
Wait for output file
Receive output file length (bytes) and name
Open copy of the output file
Receive output file from the server
Write copy of the output file
Close copy of the output file
Make more measurements?
More measurements message to the server
NO
No more measurements message to the server
Power off board?
NO
YES
Power off board message
Do not power off board message
EXIT

Figure B.4: Flowchart of the client program

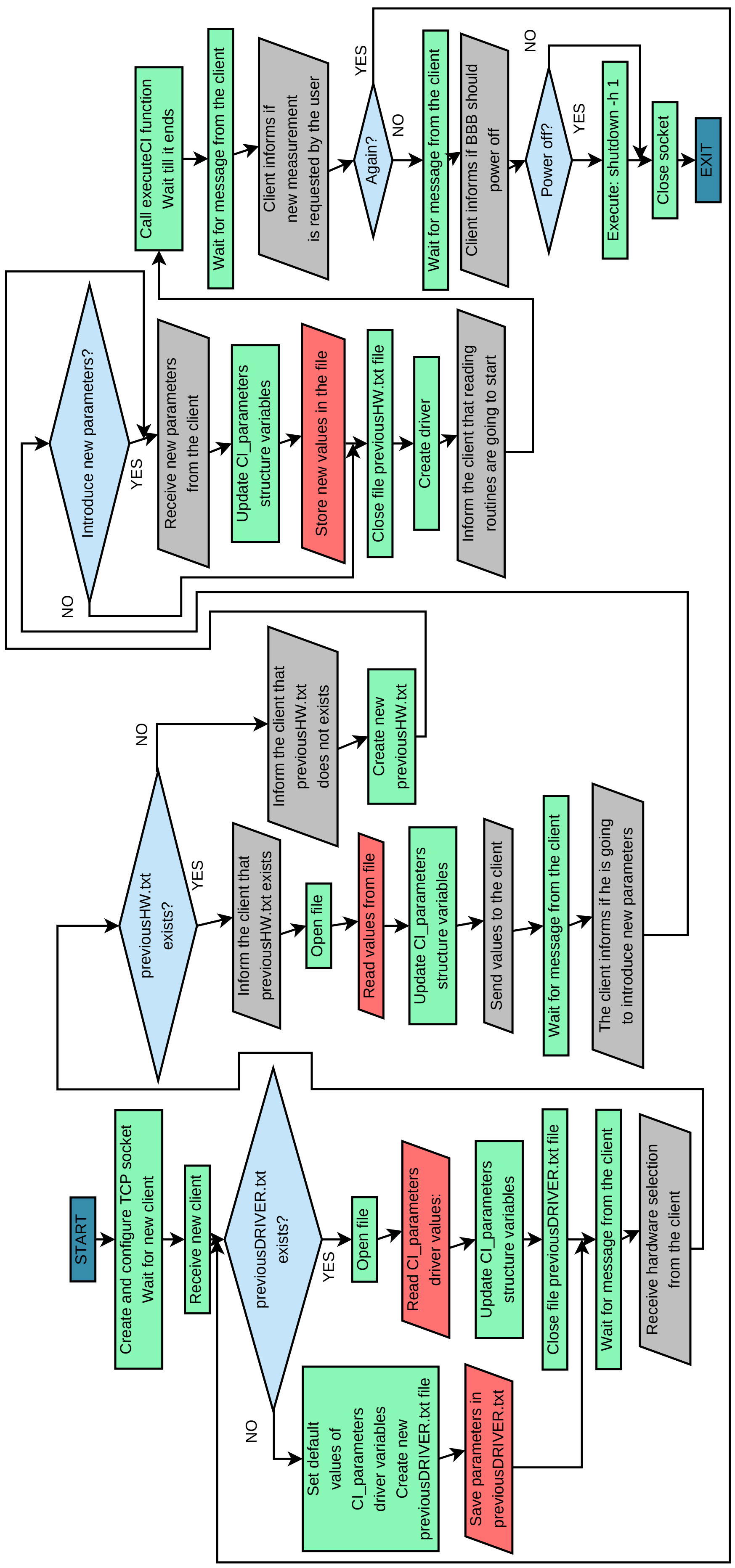


Figure B.5: Flowchart of the server program

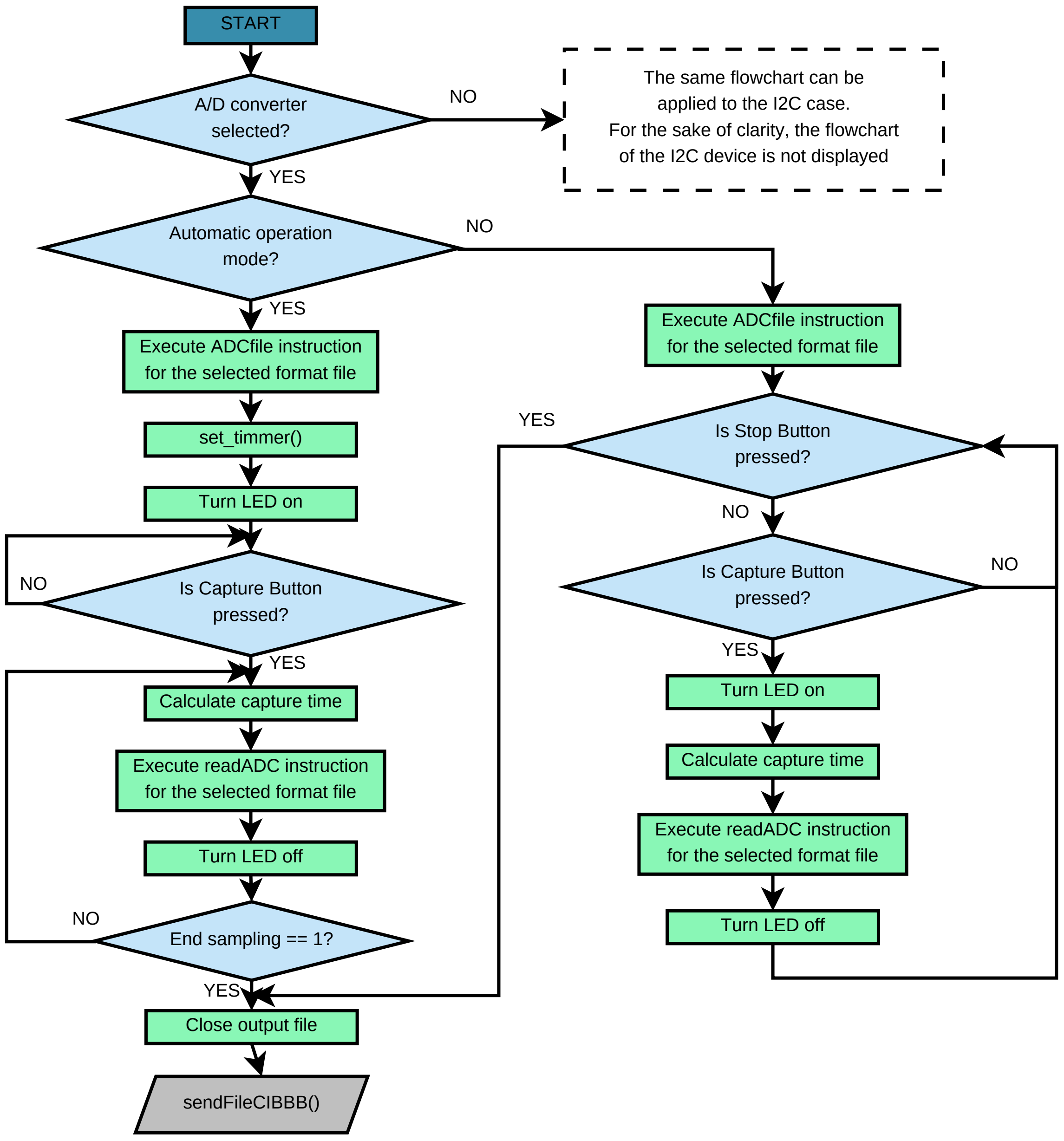

START
A/D converter selected?
NO
The same flowchart can be applied to the I2C case. For the sake of clarity, the flowchart of the I2C device is not displayed
YES
Automatic operation mode?
NO
YES
Execute ADCfile instruction for the selected format file
set_timmer()
Turn LED on
NO
Is Capture Button pressed?
YES
Calculate capture time
Execute readADC instruction for the selected format file
Turn LED off
NO
End sampling == 1?
YES
Close output file
sendFileCIBBB()
Execute ADCfile instruction for the selected format file
YES
Is Stop Button pressed?
NO
Is Capture Button pressed?
NO
YES
Turn LED on
Calculate capture time
Execute readADC instruction for the selected format file
Turn LED off

Figure B.6: Flowchart of the *executeCI()* function

APPENDIX C: USER MANUAL FOR THE REMOTE CONTROL OF CI-BBB

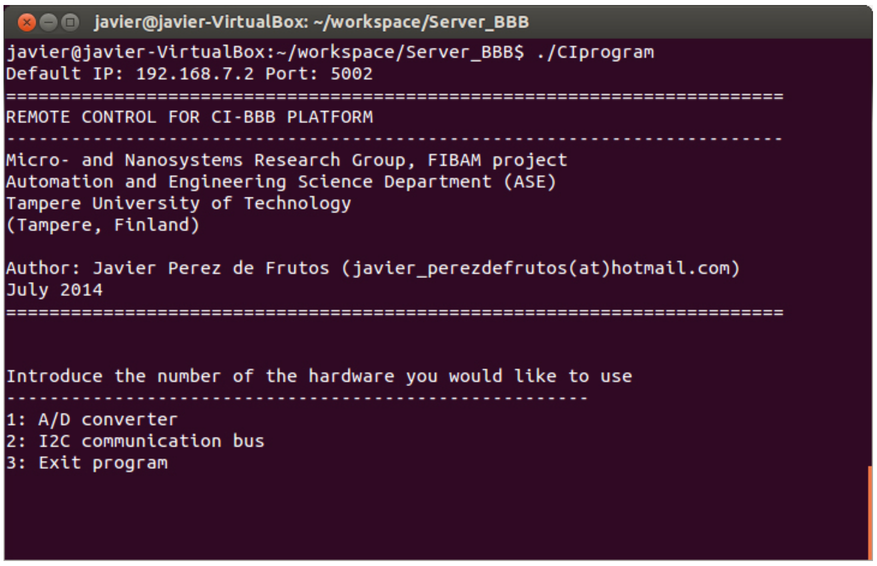
Start up the CI-BBB

1. Plug the USB cable to the miniUSB port of the BBB and the USB connector of the remote computer, to power up the system and connect to the CI-BBB. A Secure Shell (SSH) connection can be used to access the BBB.

2. Execute the program `CIserver`, located in the `/root` directory, to initiate the server program of the Remote Control. The default port (5002) of the server can be changed by introducing the port number when executing the program e.g., to use the port number 5000, type the command `./CIserver 5000`

3. Execute the program `CIclient` to establish a connection between the remote computer and the BBB. Make sure the client is configured with the same port as the server, and the correct IP direction of the BBB. Configure a new IP and port number by adding this information, when executing the program with the command `./CIclient 192.168.7.2 5002`

4. The Hardware Selection screen should be now displayed on the terminal, as illustrated in Figure C.1.



```
javier@javier-VirtualBox: ~/workspace/Server_BBB
javier@javier-VirtualBox:~/workspace/Server_BBB$ ./CIprogram
Default IP: 192.168.7.2 Port: 5002
======================================================================
REMOTE CONTROL FOR CI-BBB PLATFORM
----------------------------------------------------------------------
Micro- and Nanosystems Research Group, FIBAM project
Automation and Engineering Science Department (ASE)
Tampere University of Technology
(Tampere, Finland)

Author: Javier Perez de Frutos (javier_perezdefrutos(at)hotmail.com)
July 2014
======================================================================


Introduce the number of the hardware you would like to use
----------------------------------------------------
1: A/D converter
2: I2C communication bus
3: Exit program
```

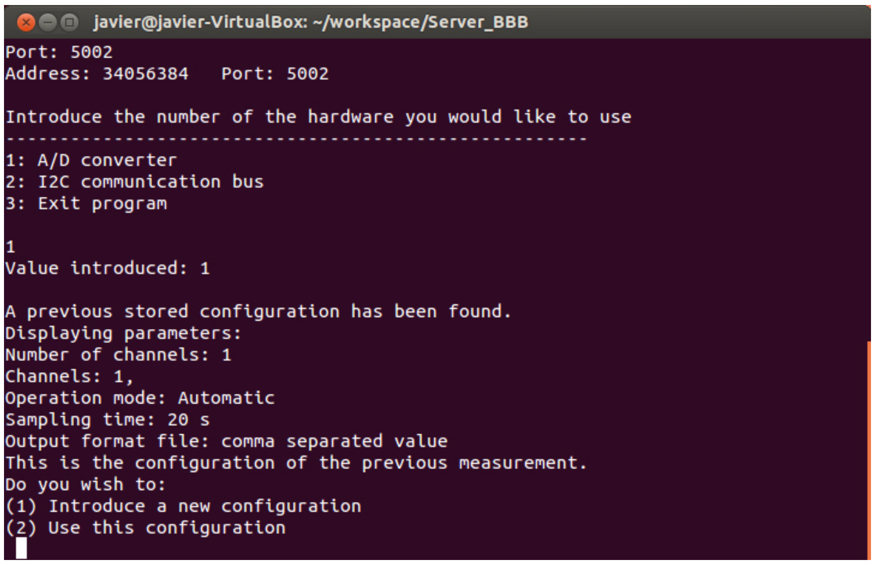
Figure C.1: Hardware Selection screen

5. Type the number according to the desired device to use. Or type `3` to exit the program.

A/D conversion of analogue signals

1. Type `1` in the Hardware Selection screen to use the A/D converter.

2. The client will show the configuration settings used in the last measurement, as illustrated in Figure C.2. If there is no information of a previous measurement, the program will ask the to introduce new values for the different parameters displayed on the terminal. The configuration is stored in the file *backupADC.txt*, in the directory `/root/BACKUP` of the BBB.



```
javier@javier-VirtualBox: ~/workspace/Server_BBB
Port: 5002
Address: 34056384   Port: 5002

Introduce the number of the hardware you would like to use
------------------------------------------------------
1: A/D converter
2: I2C communication bus
3: Exit program

1
Value introduced: 1

A previous stored configuration has been found.
Displaying parameters:
Number of channels: 1
Channels: 1,
Operation mode: Automatic
Sampling time: 20 s
Output format file: comma separated value
This is the configuration of the previous measurement.
Do you wish to:
(1) Introduce a new configuration
(2) Use this configuration
```

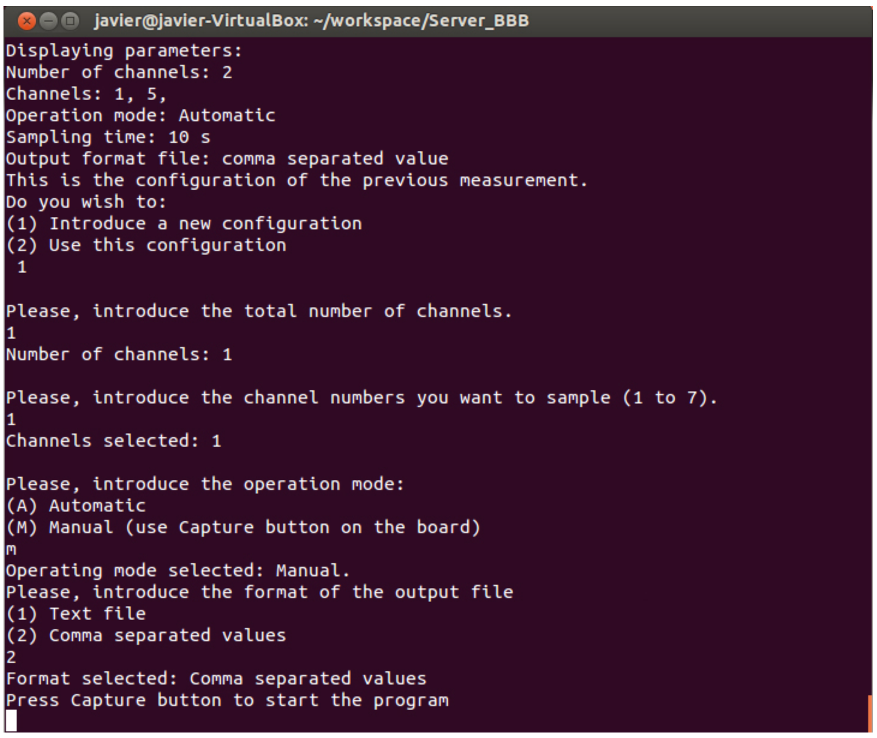
Figure C.2: Backup configuration of the A/D converter

3. Type `1` to select the displayed set up to configure the measurement process. If a new configuration is needed, type `2` to introduce new values for the configuration parameters.

4. If the value introduced is 2, the `CIclient` program will display the different configuration parameters while asking for the new values. Press enter after typing the new configuration. When finished, the CI-BBB will configure the system, and wait for the *Capture Button*. Figure C.3 shows a sample of how the program request the new set up.

5. If the manual mode is selected, press and hold the *Capture Button* whenever the input signal should be sampled. Press the *Stop Button* to finish the process. For the automatic mode, just press the *Capture Button* to start sampling.

6. After the capture data process is finished, the `CIserver` will send the generated file with the sampled values to the remote computer, and will be stored in the same directory as from where the `CIclient` program is executed. A

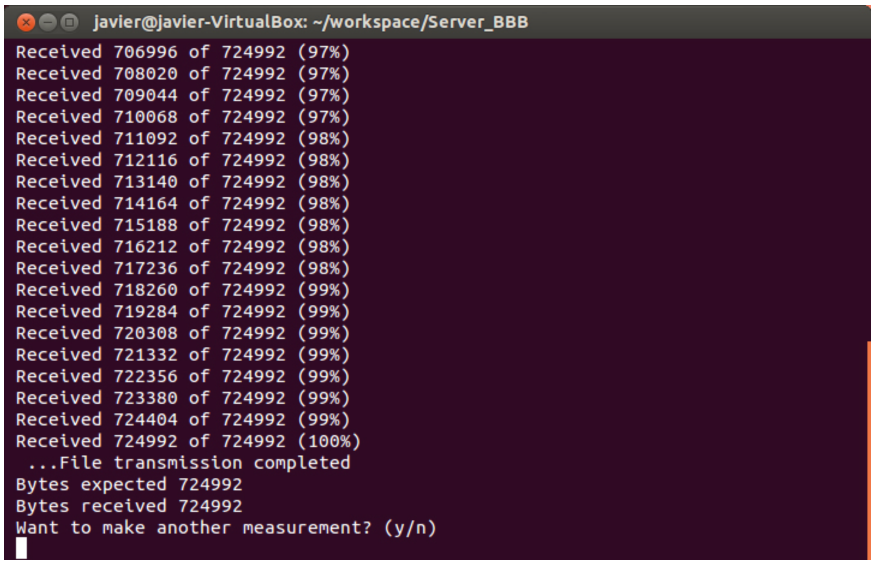


```
javier@javier-VirtualBox: ~/workspace/Server_BBB
Displaying parameters:
Number of channels: 2
Channels: 1, 5,
Operation mode: Automatic
Sampling time: 10 s
Output format file: comma separated value
This is the configuration of the previous measurement.
Do you wish to:
(1) Introduce a new configuration
(2) Use this configuration
 1

Please, introduce the total number of channels.
1
Number of channels: 1

Please, introduce the channel numbers you want to sample (1 to 7).
1
Channels selected: 1

Please, introduce the operation mode:
(A) Automatic
(M) Manual (use Capture button on the board)
m
Operating mode selected: Manual.
Please, introduce the format of the output file
(1) Text file
(2) Comma separated values
2
Format selected: Comma separated values
Press Capture button to start the program
```

Figure C.3: A/D converter new parameters screen

copy of the file is stored in the local memory of the BBB, in the directory `/root/Documents`.

7. After the file transfer is completed, the client program will asked for a new measurement as illustrated in Figure C.4. Type `Y` to proceed with a new data capture, or `N` to exit the program.



```
javier@javier-VirtualBox: ~/workspace/Server_BBB
 Received 706996 of 724992 (97%)
 Received 708020 of 724992 (97%)
 Received 709044 of 724992 (97%)
 Received 710068 of 724992 (97%)
 Received 711092 of 724992 (98%)
 Received 712116 of 724992 (98%)
 Received 713140 of 724992 (98%)
 Received 714164 of 724992 (98%)
 Received 715188 of 724992 (98%)
 Received 716212 of 724992 (98%)
 Received 717236 of 724992 (98%)
 Received 718260 of 724992 (99%)
 Received 719284 of 724992 (99%)
 Received 720308 of 724992 (99%)
 Received 721332 of 724992 (99%)
 Received 722356 of 724992 (99%)
 Received 723380 of 724992 (99%)
 Received 724404 of 724992 (99%)
 Received 724992 of 724992 (100%)
  ...File transmission completed
 Bytes expected 724992
 Bytes received 724992
 Want to make another measurement? (y/n)
```

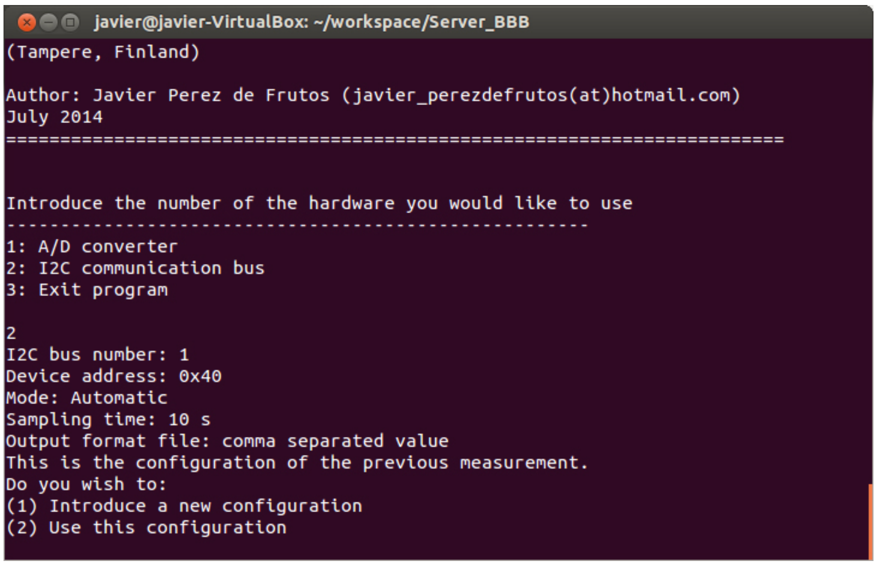
Figure C.4: New measurement screen

8. Before exiting, the Remote Control allows to shutdown the BBB. Type `Y` to power off the system, or `N` to exit the program without turning off the BBB.

I^{2}C communication bus

1. In the Hardware Selection screen, type `2` to use the I^{2}C communication bus.

2. If the file *backupI2C.txt* exists in the directory `/root/BACKUP` of the BBB, the `CIclient` program will display the last configuration used. Type `1` to used the configuration displayed. Type `2` to introduce new values. If no backup file was detected, the program will ask directly for a new configuration. Figure C.5 illustrates a sample of the backup values stored from a previous measurement, done with the I^{2}C.
   Note: *The I^{2}C device number corresponds to the AM3359 module number. Figure 4.12 illustrates the device number associated with each group of I^{2}C pins.* `i2cdetect` *command of the i2c-tools software package displays the address of the devices connected to the I^{2}C bus.*



```
javier@javier-VirtualBox: ~/workspace/Server_BBB
(Tampere, Finland)

Author: Javier Perez de Frutos (javier_perezdefrutos(at)hotmail.com)
July 2014
========================================================================


Introduce the number of the hardware you would like to use
------------------------------------------------------
1: A/D converter
2: I2C communication bus
3: Exit program

2
I2C bus number: 1
Device address: 0x40
Mode: Automatic
Sampling time: 10 s
Output format file: comma separated value
This is the configuration of the previous measurement.
Do you wish to:
(1) Introduce a new configuration
(2) Use this configuration
```

Figure C.5: Sample of backup values for the I^{2}C bus

3. Press the *Capture Button* to start the sampling process. When running in manual mode, hold the *Capture Button* to capture new data and press the *Stop Button* to finish the process.

4. The `CIserver` will send the output file to the remote computer, and store a copy in the local memory of the BBB, in the directory `/root/Documents`.

5. Type `Y` to return to the Hardware Selection screen to make a new measurement, or `N` to exit the program. Introduce again `Y` to turn off the BBB, or `N` to leave it on and exit the program.

How to find the I^{2}C device address

The software package *i2c-tools* provides with functions to detect and find the I^{2}C modules available, and the devices connected to communication bus. The command `i2cdetect -l` displays a list with the mapped I^{2}C devices, connected to the system. Figure C.6 depicts the default I^{2}C mapped modules of the BBB (see Section 3.2.1). `i2cdetect -r` followed by the module number, as displayed in the list of mapped components, scans the addresses associated with the I^{2}C bus selected. Figure C.6 illustrates also the addresses associated with the I^{2}C2 module (*i2c-1*). It can be seen that at direction 0x40 there is a component available, to communicate with, and addresses 0x54 to 0x57 (named as U) are reserved, and thus are not available to connect an external device to them. [45]

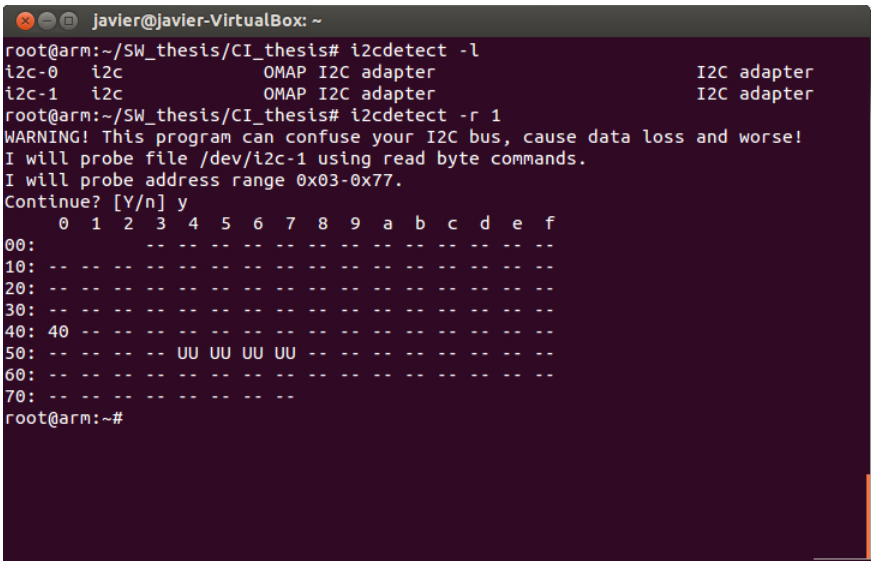

```
javier@javier-VirtualBox: ~
root@arm:~/SW_thesis/CI_thesis# i2cdetect -l
i2c-0   i2c             OMAP I2C adapter                        I2C adapter
i2c-1   i2c             OMAP I2C adapter                        I2C adapter
root@arm:~/SW_thesis/CI_thesis# i2cdetect -r 1
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-1 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@arm:~#
```

Figure C.6: Execution of the commands `i2cdetect -l` and `i2cdetect -r 1`