UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Biofilm growth monitoring

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Josep Izquierdo Moles

In partial fulfilment

of the requirements for the degree in

ELECTRONIC SYSTEMS ENGINEERING

Advisor: Miguel J. Garcia

Barcelona, May 2019

Abstract

The purpose of this project is design and develop a “original” system and low cost as a variation to de own by the “Grupo Sistemas Sensores”. The propose of this design is measure the variation of the resonance frequency and this damping in crystal quartz when grows bacteria in his surface.

The “originality” of this design consist in adding the crystal in a controlled gain loop, to maintain the amplitude oscillation, the crystal frequency and this damping will be measured by the oscillation frequency and the control signal.

This system pretends give a tool to the biologist to characterize the grow of those bacteria’s.

All the measure principles, hardware designs of all the different parts and software are present in this thesis.

Resum

El propòsit d'aquest projecte és dissenyar i desenvolupar un sistema "original" i de baix cost com una variació del que ja disposa el “Grupo Sistemas Sensores”. L'objectiu del disseny és mesurar variacions de la freqüència de ressonància i de l'amortiment en cristalls de quart quan creixen bateries en la superfície.

L'"originalitat" d'aquest disseny consisteix en incluir cristalls de quartz en un llaç de retroalimentació amb guany controlat, per així poder mantenir l'amplitud de l'oscil·lació. La freqüència del cristall i el seu amortiment es mesuren a través de la freqüència d'oscil·lació i el senyal de control del guany de l'amplificació.

Aquest sistema intenta donar una eina més als biòlegs per a poder caracteritzar el creixement dels.

Tot els principis de mesura, dissenys de les diferents parts i software estan presents en aquest treball.

Resumen

El propósito de este proyecto es diseñar y desarrollar un sistema “original” y bajo coste como una variación del que ya dispone el Grupo Sistemas Sensores. El objeto del diseño es medir las variaciones de la frecuencia de resonancia y de la amortiguación en cristales de cuarzo cuando se hacen crecer bacterias en su superficie.

La “originalidad” del diseño consiste en la inclusión del cristal de cuarzo en un lazo de realimentación con ganancia controlada, para así mantener la amplitud de oscilación. La frecuencia del cristal y su amortiguamiento se miden a través de la frecuencia del oscilador y la señal de control de ganancia de la amplificación,

Este sistema pretende dar una herramienta más a los biólogos para la caracterización del crecimiento de bacterias.

Todos los principios de medida, diseño de las distintas partes y software están presentes en este trabajo.

Acknowledgements

Antes de nada, querría agradecer al mi tutor, Miguel García y al GSS, por proponerme este reto, guiarme en su desarrollo y ayudarme a tomar las mejores decisiones posibles tanto a nivel profesional como personal.

La realización de este proyecto, junto a la finalización del grado y el cierre de mi ciclo formativo, es un momento que no olvidare un muchos años y el camino para llegar, no ha sido llano, sino todo lo contrario, por eso me gustaría agradecer a mi familia todos los esfuerzos que han hecho para ayudarme, en especial a Lidia, quien ha sido mi tendón de Aquiles durante todos estos años.

Y para finalizar, pero no menos importante, a todos los compañeros y profesores con los que me he pasado incontables horas en clase, laboratorio, etc

Revision history and approval record

Revision	Date	Purpose
0	02/03/2019	Document creation
1	09/05/2019	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Josep Izquierdo Moles	josepizki@gmail.com
Miguel J. Garcia	miguel.j.garcia@upc.edu

Written by:		Reviewed and approved by:	
Date	02/03/2019	Date	09/05/2019
Name	Josep Izquierdo Moles	Name	Miguel J. Garcia
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract 1
Resum 2
Resumen 3
Acknowledgements 4
Table of contents 6
List of Figures 9
List of Tables: 11
1. Introduction 12
1.1. Objectives 12
1.2. Requirements and specifications 12
1.3. Methods and procedures 12
1.4. Work plan 13
1.4.1. Work breakdown structure 13
1.4.2. Milestones 14
1.4.3. Gantt diagram 16
1.5 Incidences 17
2. State of the art of the technology used or applied in this thesis 18
2.1. Biofilm 18
2.2. Quartz crystal resonators 18
2.2.1. Electrical equivalent 19
2.2.2. Parallel and serial resonance 19
2.2.3. Measure principle 20
3. Methodology / project development: 21
3.1. Project block diagram 21
3.2. Measure circuit block diagram 22
3.3 Schematics, component choose and PCB designs 23
3.3.1 Hardware block diagram 23
3.3.2 PCB design 24
3.3.3 Voltage controlled amplifier 26
3.3.4 Signal rectifier + integrator 27
3.3.5 Microcontroller 29
3.3.6 Buffer 30
3.3.7 Data Storage (EEPROM) 30

3.3.8 Temperature sensor 31
3.3.9 3.3V alimentation 31
3.3.10. PCB to PCB connectors 32
3.3.11. Power supply 32
3.3.12. External connectors 33
3.4 Analog circuit start up 34
3.4.1. Welded PCB 34
3.4.2. Test board 35
3.4.3. Voltage controlled amplifier test 36
3.4.4. Rectifier, integrator and VCA test 36
3.4.5. Test with crystal 37
3.4.6. Buffer “1 bit ADC” test 39
3.4.7. Temperature test 40
3.5 Software 41
3.5.1 Code structure 41
3.5.2. Code flow diagram 42
3.6 Measures 43
4. Results 44
5. Budget 45
5.1. Bill of materials 45
5.2. Design and prototyping cost 46
5.3. Economical/financial viability 46
6. Conclusions and future development: 47
Bibliography: 48
Appendices: 49
A Tools 49
A.1. Simulation Environment 49
A.2. Design environment 49
A.3. Weld tools 49
A.4. Inspection tools 50
A.5. Development tools 51
B Electric schemes 52
C Layout 55
C.1. Measure board top layer 55

C.2. Measure board bottom layer 56
C.3. Power supply and connection board top layer 57
C.4. Power supply and connection board bottom layer 58
D Firmware 59
D.1. main.c 59
D.2. initialization.h 61
D.3. initialization.c 63
D.4. DelayMacro.h 64
D.5. DelayMacro.c 64
D.6. E2PROM.h 64
D.7. E2PROM.c 65
D.8. FrequencyCounter.h 66
D.9. FrequencyCounter.c 67
D.10. I2C.h 67
D.11. I2C.c 68
D.12. MuxControl.h 72
D.13. MuxControl.c 72
D.14. RsCalc.h 74
D.15. RSCalc.c 74
D.16. temperature.h 75
D.17. temperature.c 75
D.18. Timer1us.h 75
D.19. Timer1us.c 75
D.20. Trigger.h 76
D.21. Trigger.c 76
D.22. UART.h 78
D.23. UART.c 79
D.24. GlobalDefines.h 80
Glossary 81

List of Figures

Figure 1.1: Utilized hardware and none 13
Figure 1.2: Work Breakdown Structure 13
Figure 2.1: Biofilm growth stage 18
Figure 2.2: Quartz crystal electrical equivalent 19
Figure 2.3: Quartz crystal resistance in front of frequency 20
Figure 3.1: Project block diagram 21
Figure 3.2: Measure circuit block diagram 22
Figure 3.3: Hardware block diagram 23
Figure 3.4: Measure board top layer 24
Figure 3.5: Measure board bottom layer 24
Figure 3.6: Power supply and interface board top layer 25
Figure 3.7: Power supply and interface board bottom layer 25
Figure 3.8: Voltage controlled generator and properties 26
Figure 3.9: VCA schematic 27
Figure 3.10: Signal rectifier simulation schematic 27
Figure 3.11: Signal rectifier simulation 27
Figure 3.12: Signal rectifier and integrator simulation schematic 28
Figure 3.13: Signal rectifier and integrator simulation 28
Figure 3.14: Signal rectifier and integrator simulation 28
Figure 3.15: Microcontroller schematic 29
Figure 3.16: Buffer schematic 30
Figure 3.17: EEPROM schematic 30
Figure 3.18: Temperature sensor schematic 31
Figure 3.19: 3.3V alimentation schematic 31
Figure 3.20: Spring connector 32
Figure 3.21: Interface and power supply board measurement 32
Figure 3.22: Power supply schematic 32
Figure 3.23: External connector schematic 33
Figure 3.24: Weld top measure board 34
Figure 3.25: Weld bottom measure board 34
Figure 3.26: Weld top interface and power supply board 34
Figure 3.27: Weld bottom interface and power supply board 34
Figure 3.28: Test board schematic 35
Figure 3.29: Weld test board schematic 35
Figure 3.30: VCA test schematic 36
Figure 3.31: Input filter configuration 1 37
Figure 3.32: Input filter configuration 2 37
Figure 3.33: VCA rectified schematic 38
Figure 3.34: VDDS and Vcontrol measure 39

Figure 3.35: Buffer schematic 39
Figure 3.36: “1-bit ADC oscilloscope measure” 40
Figure 3.37: Temperature sensor measure 40
Figure 3.38: Code structure 41
Figure 3.49: Code flow diagram 42
Figure 3.40: Vcontrol in front of resistance 43

List of Tables:

Table 3.1: VCA comparison 26
Table 3.2: Microcontroller comparison 27
Table 3.3: VCA test 36
Table 3.4: Rectifier, integrator and VCA test 37
Table 3.5: Input filter value option 37
Table 3.6: Crystal test 38
Table 3.7: Temperature sensor test 40
Table 5.1: BOM 45
Table 5.2: Design and prototyping cost 46

1. Introduction

1.1. Objectives

The goal of this project is implement a non-destructive test for monitorize the biofilm formation with an array of quartz crystals.

The mainly part of this project will be develop a hardware system which allows a microcontroller measure the resonance frequency and dumping, secondary but not less important will be develop the firmware for the measure of this characteristics and if time allow it build the code for transmission to a computer.

1.2. Requirements and specifications

The current project must comply those specifications:

Measure frequency with a resolution of 0.1Hz

Measure serial resistance with a resolution of 0.1Ω

Resistance range measurement: 200 - 1000 Ω

Monitoring at least 12 crystal

Read all 12 multiplexor channels below 5 min.

Design the system in an embedded board

Other goals are:

Design the power supply circuit

Design the data interface

1.3. Methods and procedures

This thesis will utilize the hardware develop by Luis Belanga Herrera in “Real-Time multichannel system for the monitorization of biofilm formations based on quartz crystal resonators”:

Multiplexor board

12 x Conditioning circuit

Plastic lid

Thermostatic chamber

The next figure extracted by the Luis Berlanga thesis, we can see more graphically the used hardware: in green the utilized area, in red the substituted area.

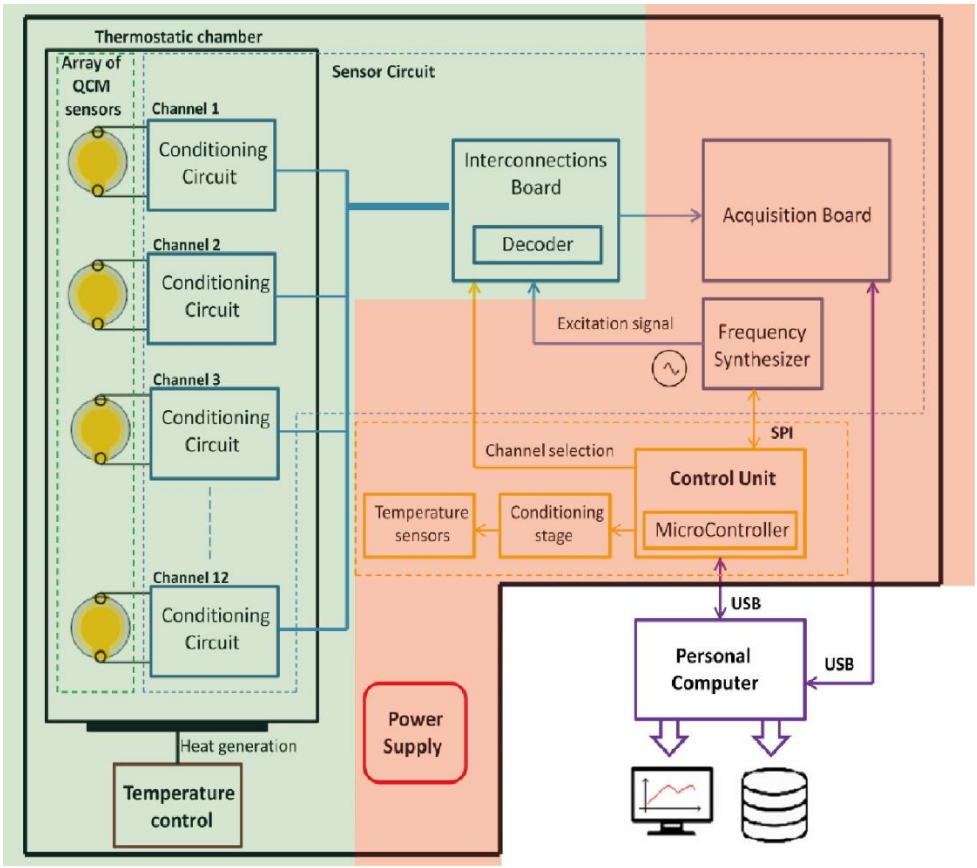


Figure 1.1: Utilized hardware and none

1.4. Work plan

1.4.1 Work Breakdown Structure

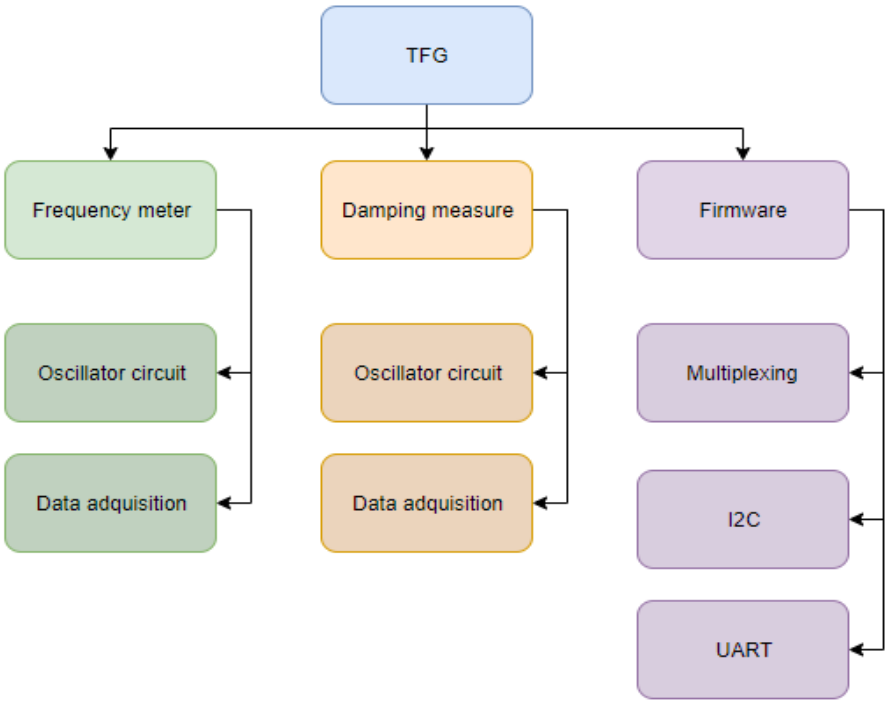


Figure 1.2: Work Breakdown Structure

1.4.2 Milestones

Sub-Project: Oscillator circuit	WP ref: WP1 and WP2	
Major constituent: Hardware prototype and simulation	Sheet 1 of 6	
Short description: Be able to make oscillate the system.	Planned start date: Planned end date:	
	Start event: End event:	
Internal task T1: Design a feedback system to make oscillate the crystal. Internal task T2: Test Internal task T3: Build the PCB	Deliverables: Simulation PCB	Dates:

Sub-Project: Data acquisition (frequency meter)	WP ref: WP1	
Major constituent: Software	Sheet 2 of 6	
Short description: Be able to measure the oscillation frequency.	Planned start date: Planned end date:	
	Start event: End event:	
Internal task T1: Program firmware for the serial resistance acquisition. Internal task T2: Test it.	Deliverables: Code for data acquisition	Dates:

Sub-Project: Data acquisition (damping measure)	WP ref: WP2	
Major constituent: Software	Sheet 3 of 6	
Short description: Be able to measure the QCM resistance in the serial oscillation point through the VCA control signal.	Planned start date: Planned end date:	
	Start event: End event:	
Internal task T1: Program firmware for the serial resistance acquisition Internal task T2: Test it.	Deliverables: Code for data acquisition	Dates:

Sub-Project: Multiplexing	WP ref: WP3	
Major constituent: Software	Sheet 4 of 6	
Short description: Be able to set the multiplexor different channels to get the signal coming from the QCM sensor and the conditioning circuit.	Planned start date: Planned end date:	
	Start event: End event:	
Internal task T1: Program Internal task T2: Test it.	Deliverables: Code for QCM multiplexing	Dates:

Sub-Project: I2C	WP ref: WP3	
Major constituent: Software	Sheet 5 of 6	
Short description: Be able to store the data in to the EEPROM	Planned start date: Planned end date:	
	Start event: End event:	
Internal task T1: Learn how to I2C works Internal task T2: Program and test.	Deliverables: Code for data save	Dates:

Sub-Project: UART	WP ref: WP3	
Major constituent: Software	Sheet 6 of 6	
Short description: Be able to send data to a PC through UART.	Planned start date: Planned end date:	
	Start event: End event:	
Internal task T1: Learn how UART works Internal task T2: Program and test.	Deliverables: Code for data transmission	Dates:

1.4.3 Gantt diagram

TASK NAME	PERCENT COMPLETE	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8
Frequency detector									
Documentation	100%								
Conceptual desing & Simulate	100%								
PCB desing	100%								
Hardware test	100%								
Measure "RS"									
Documentation	100%								
Conceptual desing and Simulate	100%								
PCB desing	100%								
Hardware test	100%								
Power supply and Interface									
Documentation	100%								
PCB desing and Test	100%								
Firmware									
General code	100%								
Multiplexing	100%								
I2C	100%								
UART	100%								
Writing final report and presentation									
Measures	100%								
Final report	100%								

TASK NAME	PERCENT COMPLETE	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16
Frequency detector									
Documentation	100%								
Conceptual desing & Simulate	100%								
PCB desing	100%								
Hardware test	100%								
Measure "RS"									
Documentation	100%								
Conceptual desing and Simulate	100%								
PCB desing	100%								
Hardware test	100%								
Power supply and Interface									
Documentation	100%								
PCB desing and Test	100%								
Firmware									
General code	100%								
Multiplexing	100%								
I2C	100%								
UART	100%								
Writing final report and presentation									
Measures	100%								
Final report	100%								

TASK NAME	PERCENT COMPLETE	WEEK 17	WEEK 18	WEEK 19	WEEK 20	WEEK 21	WEEK 22	WEEK 23	WEEK 24
Frequency detector									
Documentation	100%								
Conceptual desing & Simulate	100%								
PCB desing	100%								
Hardware test	100%								
Measure "RS"									
Documentation	100%								
Conceptual desing and Simulate	100%								
PCB desing	100%								
Hardware test	100%								
Power supply and Interface									
Documentation	100%								
PCB desing and Test	100%								
Firmware									
General code	100%								
Multiplexing	100%								
I2C	100%								
UART	100%								
Writing final report and presentation									
Measures	100%								
Final report	100%								

TASK NAME	PERCENT COMPLETE	WEEK 25	WEEK 26	WEEK 27	WEEK 28	WEEK 29	WEEK 30	WEEK 31
Frequency detector								
Documentation	100%							
Conceptual desing & Simulate	100%							
PCB desing	100%							
Hardware test	100%							
Measure "RS"								
Documentation	100%							
Conceptual desing and Simulate	100%							
PCB desing	100%							
Hardware test	100%							
Power supply and Interface								
Documentation	100%							
PCB desing and Test	100%							
Firmware								
General code	100%							
Multiplexing	100%							
I2C	100%							
UART	100%							
Writing final report and presentation								
Measures	100%							
Final report	100%							

1.5. Incidences

- Difficulties in simulation because the VCA module wasn't able.
- Excess time to design the board, because their compact size and the addition of new components that wasn't predicted in the initial configuration and circuits improvement like E2PROM, temperature sensors, etc.
- Added new board designs to intercommunicate the main board with the PC and the power stage.
- Loss of laboratory (workplace) and some miss components by the works made to improve the "C5".

2. State of the art of the technology used or applied in this thesis:

2.1 Biofilm

Bacterial biofilm is defined as “biological communities formed by microorganisms adhered to a substrate in form of matrix”, those bacteria secrete a substance called extracellular polymeric substances or EPS made with enzymes, proteins, sugar molecules and others, with this settlement they create communities
Biofilm formation is a complex and dynamic process, they got different stages as we can saw in the next figure.

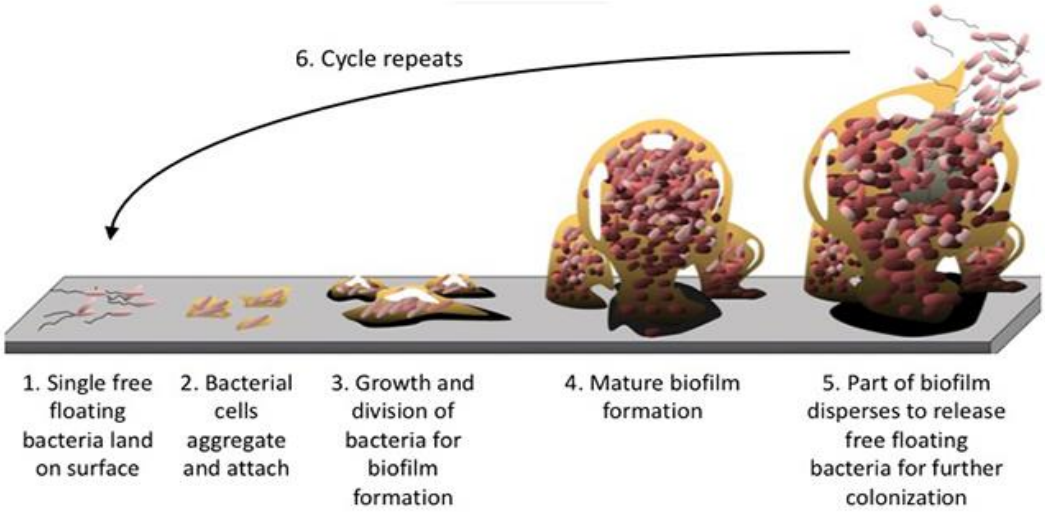


Figure 2.1: Biofilm growth stage

Some examples of biofilm in human body are:

- **Staphylococcus epidermidis**
  Bacteria commonly found in the environment, human and animal skin, mucus, etc. Usually this bacteria don’t cause any disease, but they can create biofilms in plastic devices, because of that they can compromise the life of operated patients.
- **Staphylococcus aureus**
  Frequently found in the nose, respiratory tract and skin, member of the normal flora of the body, when they are found in their natural place, they aren’t pathogenic but it can cause diseases like pneumonias if they are found in other tissue.
- **Staphylococcus mutans**
  This bacteria is usually found in human mouth, creating biofilm with other bacteria and creating dental plaque.

2.2 Quartz crystal resonators

Quartz crystal microbalance or QCM is sensitive device based on the measure of the piezoelectric characteristic, the piezoelectric characteristic makes vary the resonance frequency due the deposit of little amounts of mass on the quartz crystal.

2.2.1. Electrical equivalent

A quartz crystal can be modelled as an electronic circuit near his they resonance points, as it shown in the figure … it composes of two branch, the “motional” and the “electrical”.

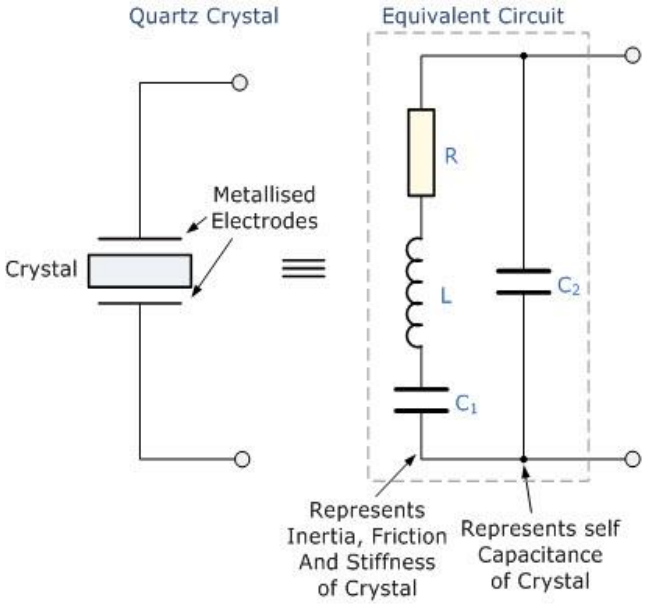


Figure 2.2: Quartz crystal electrical equivalent

The “motional branch” is represented by 3 components, the resistor “R” corresponds to the energy dissipation, the inductor “L” corresponds to the inertial of the oscillation and the capacitance “C1” corresponds to the stored energy in the oscillation.
The “electrical branch” is represented by one capacitance “C2”, this capacitor represents the parasite capacitances of the crystal electrodes and connectors.

2.2.2. Parallel and serial resonance

If we mathematically analyse the electrical model using the Laplace transform, the impedance can be written as:

$$Z(s) = \left(\frac{1}{C2 \ * \ S}\right) || \left(\frac{1}{C1 \ * S} + L * S \ + \ R\right)$$

or

$$Z(s) = \frac{S^2 + S\frac{R}{L} + w_s^2}{(s * C_1)(S^2 + S\frac{R}{L} + w_p^2)}$$

Where w_S is the series resonance and w_p is the parallel resonance, then:

$$w_S = \frac{1}{\sqrt{L*C_1}} \text{ and } w_p = \sqrt{\frac{C_2+C_1}{L*C_2*C_1}}$$

Also in this points we got the minimum (for the series resonance) and the maximum (for the parallel resonance) impedance, graphically:

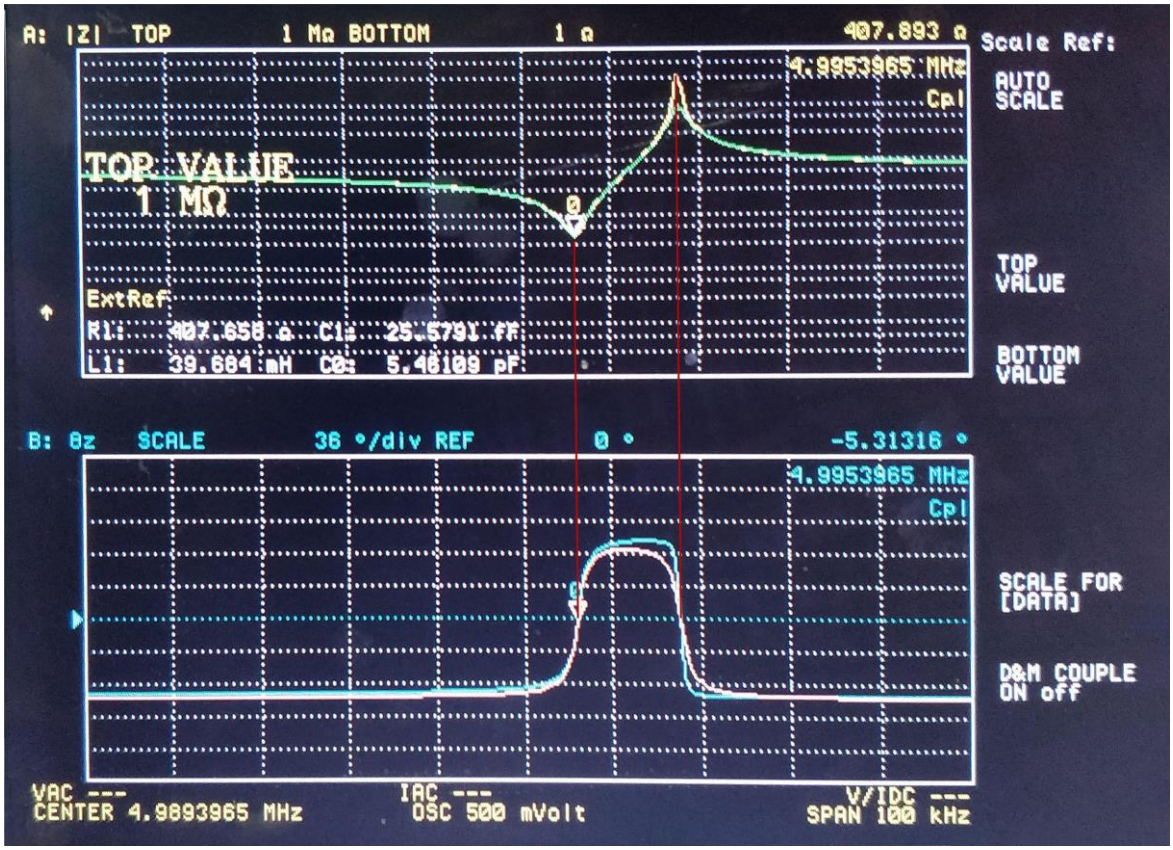


Figure 2.3: Quartz crystal resistance in front of frequency

2.2.3 Measurement principle

The motional inductance L increase when a mass is added to the crystal, this growth of the impedance affects directly to the series resonance, making it shift. Also this mass will increase the value of the motional resistance R, with this variation we can calculate the viscosity of the liquid of film.

The relation between those parameters and the and the material are provide by the Kanazawa's equation, this equation will predict theoretically the dependence of the applied substance and the variation of the mentioned components.

$$\Delta f_s = -f_u^{\frac{3}{2}} \sqrt{\frac{p_L n_L}{\pi p_q u_q}}$$

$$\Delta Rs = n * ws * \frac{Lu}{\pi} \sqrt{\frac{2Wsp_L n_L}{p_q u_q}}$$

where

- f_u : is the frequency of oscillation of the crystal without substances
- p_q : is this quartz density,
- u_q : is the shear modulus of quartz
- p_L : is the density of the substance
- n_L : is the viscosity of the substance
- n : is the number of sides in contact with the substance
- w_s: is the angular frequency of the resonation
- L_u: is the crystal inductance in the air

3. Methodology / project development:

3.1 Project block diagram

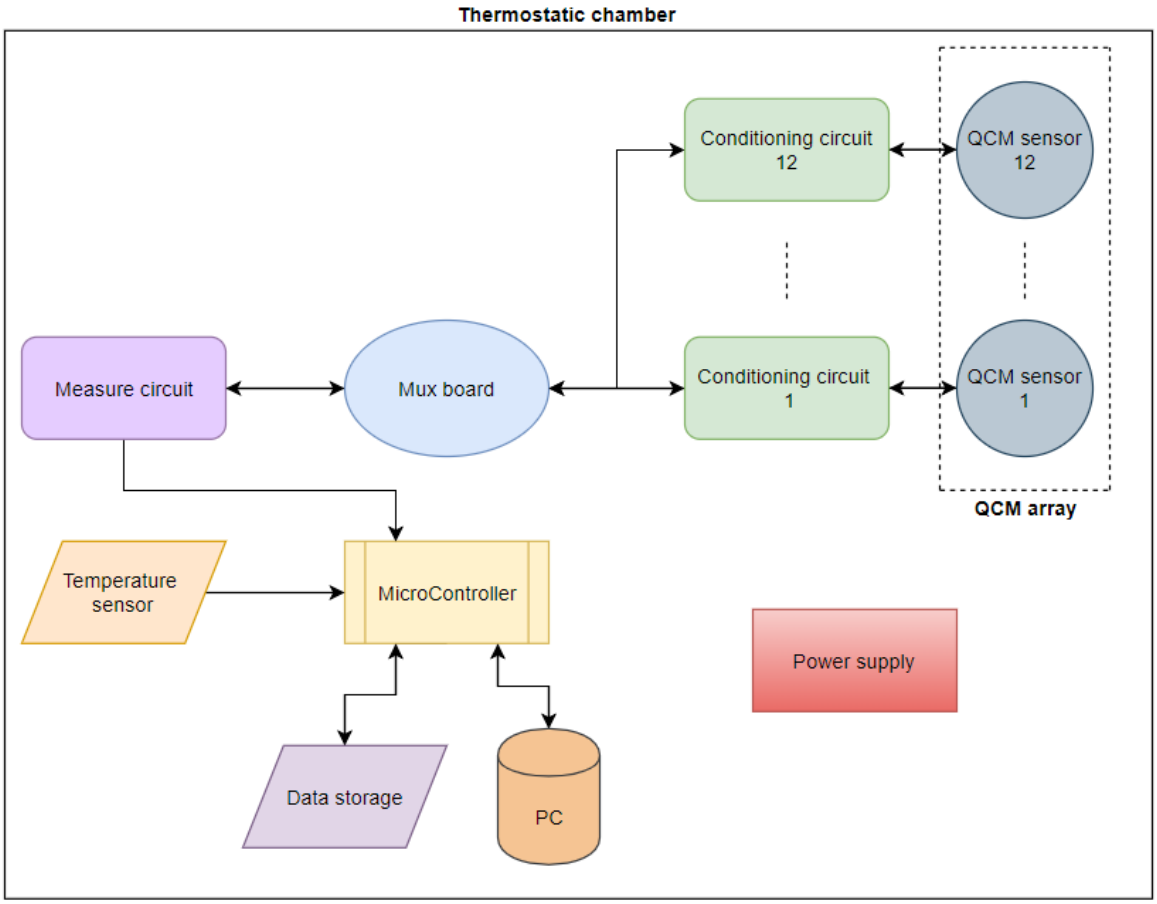


Figure 3.1: Project block diagram

MicroController:

Is the responsible to control the mux board to connect to the different "conditioning circuit" and "QCM sensor", also is the responsible to storage the calculated data, read temperature and calculate the frequency and dumping.

Mux Board:

Interconnect all the conditioning circuits and the measure board, this board also works as mechanical structure holding all the "conditioning circuits" and "QCM sensors".

Conditioning circuit:

Give a conditioned signal to the measure board.

QCM sensor:

Will be in contact with the biofilm, and they are the main responsible to change the resonance frequency and resistance.

Measure board:

Controls the frequency amplitude and allows the microcontroller to measure frequency and dumping.

Temperature sensor:

Allows the microcontroller to calculate the temperature in the “Thermostatic chamber”

Data storage:

Allows the microcontroller to storage data preventing data losses and saving pre-settings.

Power supply:

Supplies the electric energy to all the parts of the project through different DC/DC and linear regulators.

3.2 Measure circuit block diagram

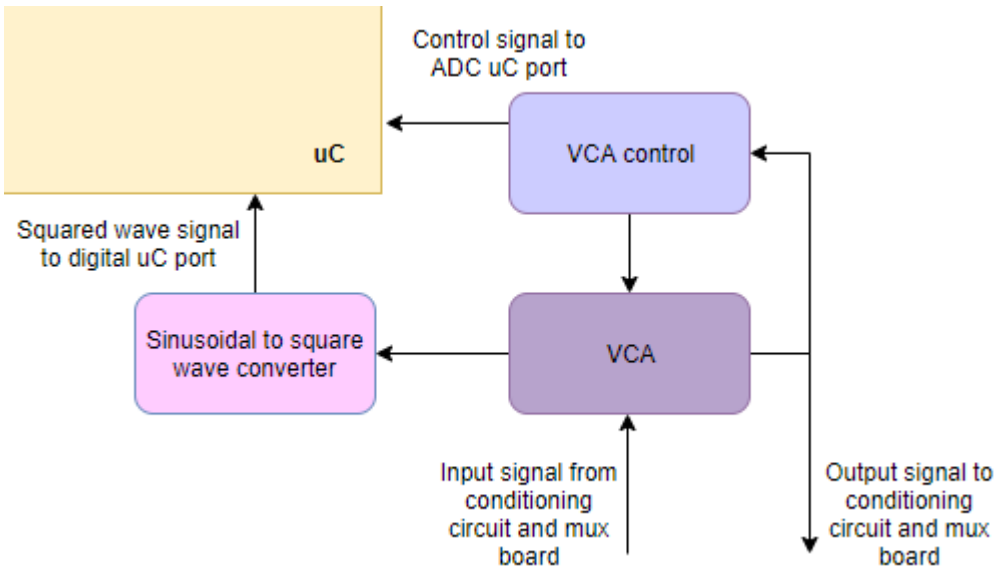


Figure 3.2: Measure circuit block diagram

VCA:

Amplifier in charge to hold the sinusoidal wave amplitude in the different range of frequencies and resistances.

VCA control:

Have to give to the VCA a signal to vary he’s gain in proportion to his output.

Sinusoidal to square wave converter:

Transform the sinusoidal signal in to square one, making the microcontroller capable to calculate the frequency.

3.3 Schematics, component choose and PCB designs

3.3.1 Hardware block diagram

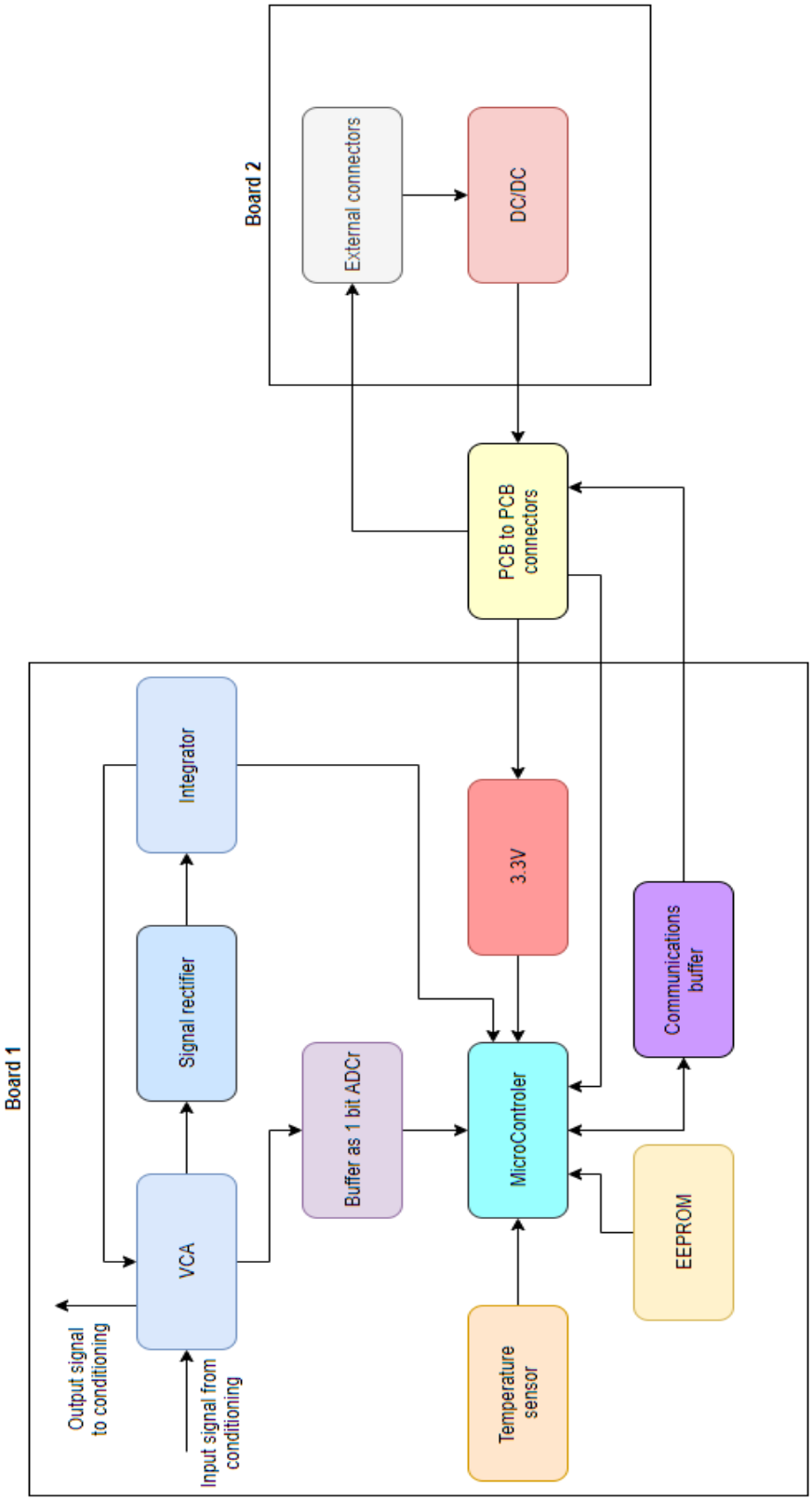


Figure 3.3: Hardware block diagram

3.3.2 PCB design

Measure system top layer

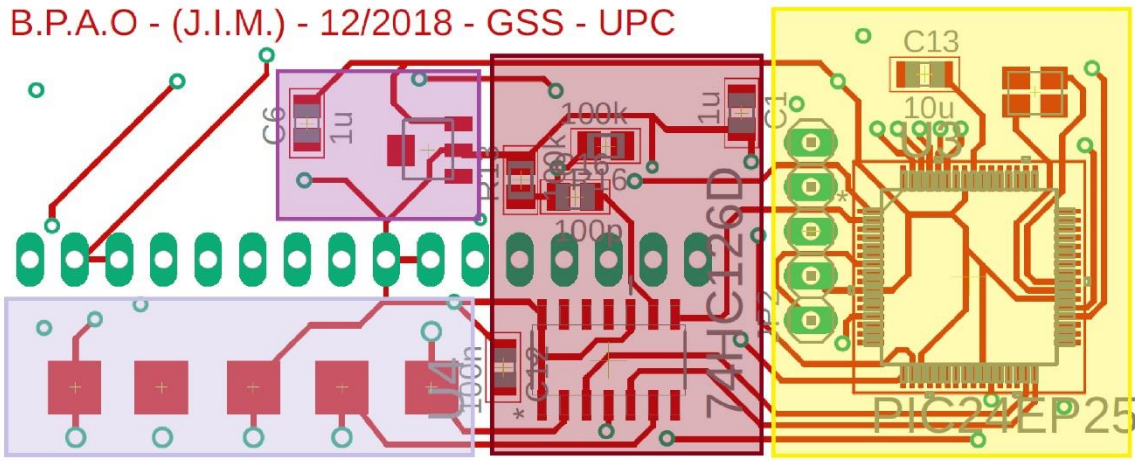


Figure 3.4: Measure board top layer

The squared areas are the following parts:

- Yellow: Microcontroller
- Brown: Buffer
- Purple: 3.3V
- Gray: PCB to PCB spring pads

Measure system bottom layer

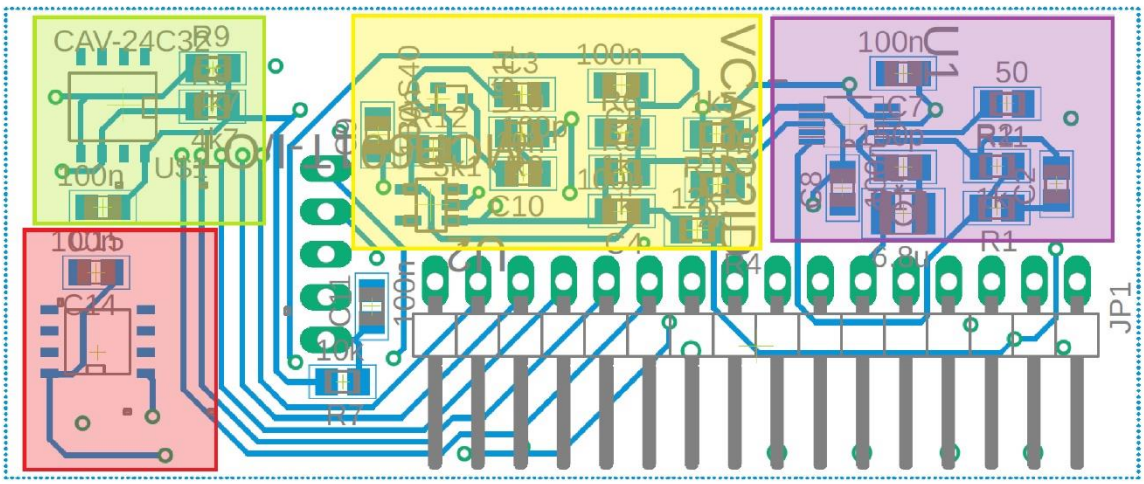


Figure 3.5: Measure board bottom layer

The squared areas are the following parts:

- Purple: VCA
- Yellow: Integrator + Rectifier (VCA control)
- Green: EEPROM
- Red: Temperature sensor

Power supply top layer

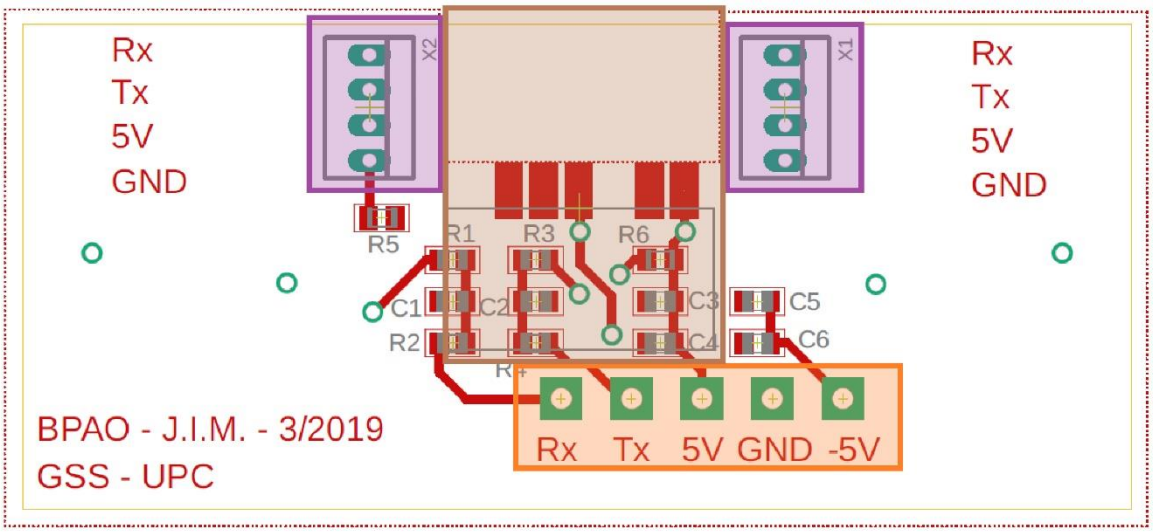


Figure 3.6: Power supply and interface board top layer

The squared areas are the following parts:

Brown: DC/DC converter

Purple: External connectors

Orange: PCB to PCB spring connectors

Power supply bottom layer

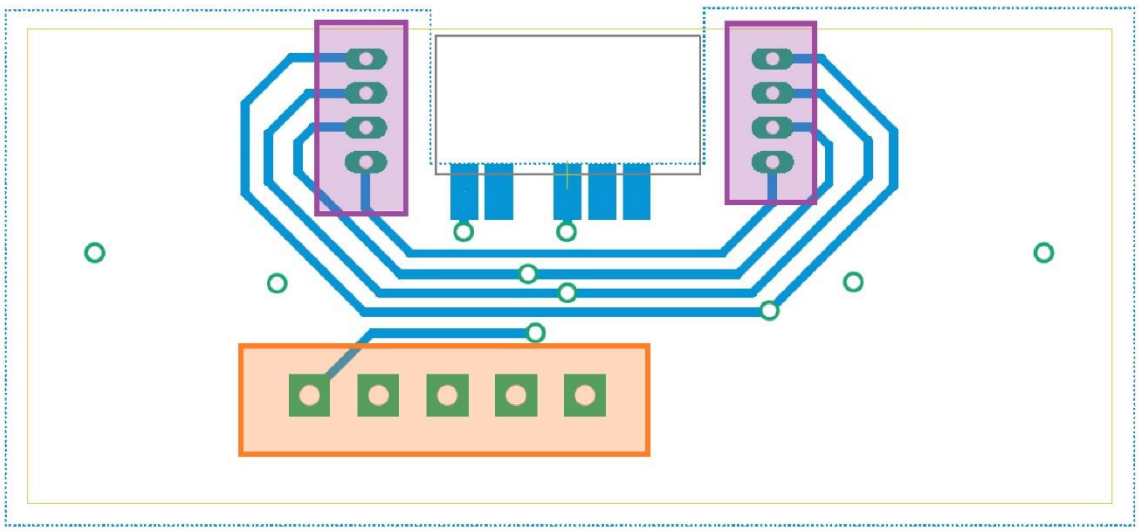


Figure 3.7: Power supply and interface board bottom layer

Purple: External connectors

Orange: PCB to PCB spring connectors

3.3.3 Voltage controlled amplifier

Options and choose

	VCA822	VCA810	LMH6503	AD8338
Bandwidth	150 MHz	30 MHz	135 MHz	18 MHz
Noise	8.2 nV/√Hz	2.4 nV/√Hz	6.6 nV/√Hz	4.5 nV/√Hz
Gain variation	Linear	Logarithmic	Logarithmic	Logarithmic
Price	6.73 €	14.40 €	9.21 €	10.38 €

Table 3.1: VCA comparison

After compare all the VCA, we choose the VCA822, because this linear gain control, low price and big bandwidth. The other are discarded because the gain control is logarithmic, and this will increase the complexity of our design.

Simulation

After some web search and can't found the VCA822 model for simulate with the help of professor Pablo Ortega, we look for an alternative solution and this is the result.
With editing the voltage controller generator characteristic "PSpice Template" with the code "E^@REFDES %3 %4 VALUE={%1,%2}*@GAIN*sin(time*2*3.1415*5e6)" as we can see in the next figures, we build a sinusoidal generator with variable amplitude defined by the voltage in the aux pin.

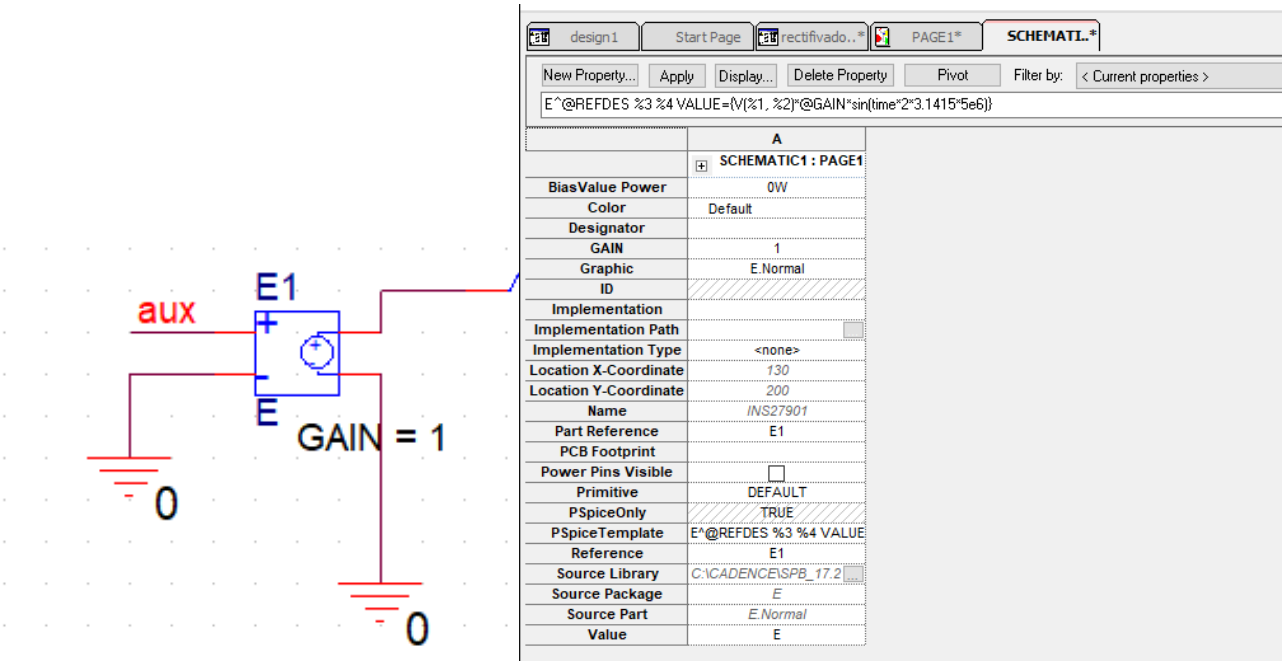


Figure 3.8: Voltage controlled generator and properties

Schematic

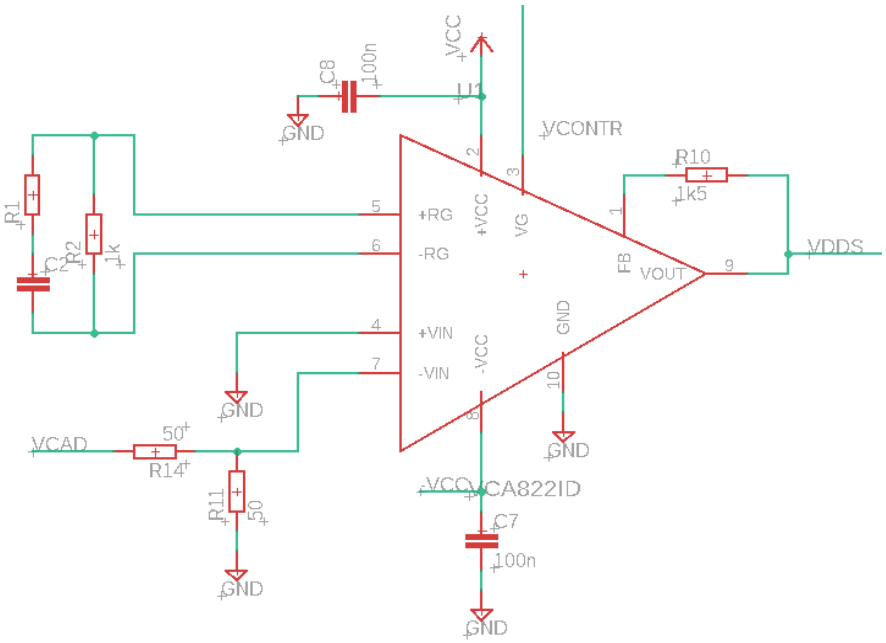


Figure 3.9: VCA schematic

3.3.4 Signal rectifier + integrator

Simulation

The first step was simulating the rectifier and ensure the signal on the blue marker was near to zero, because this will be the input "Vi+" of the amplifier in charge to integrate.

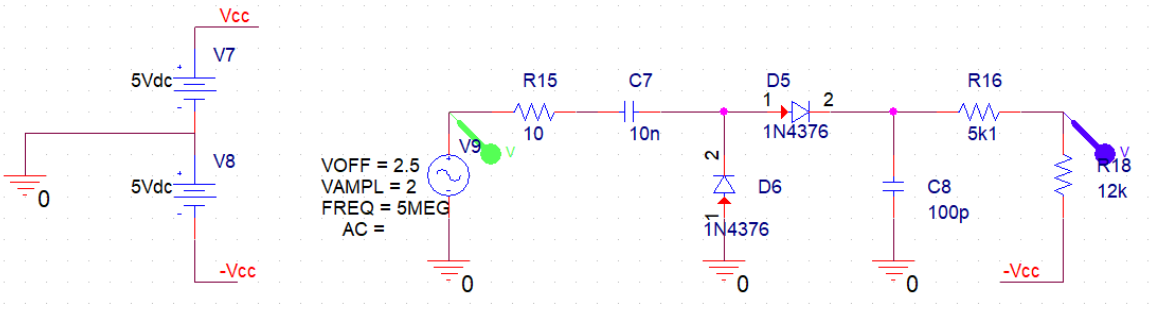


Figure 3.10: Signal rectifier simulation schematic

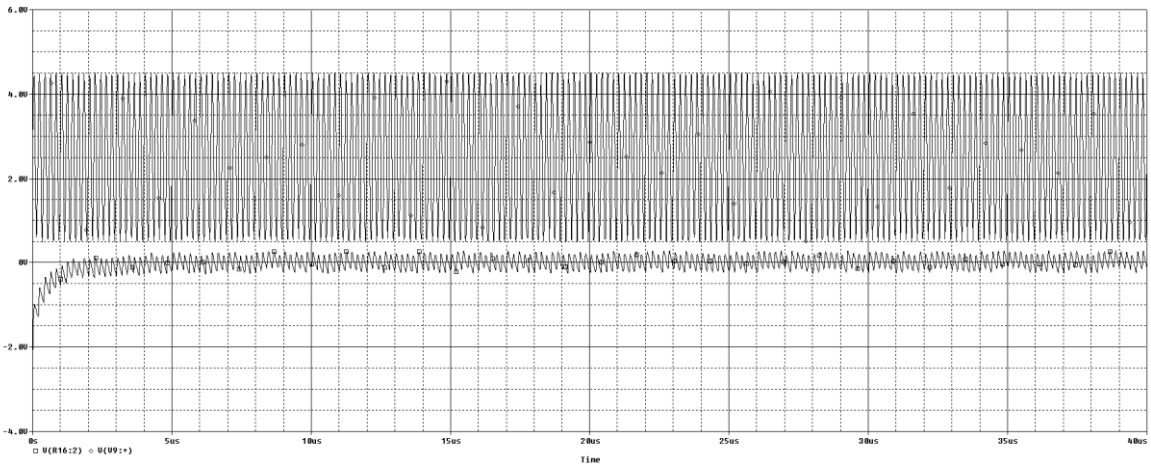


Figure 3.11: Signal rectifier simulation

After simulate the rectifier, we added the VCA hand created block and the integrator, we simulate the circuit many times to prove the integer stability and the integration speed, for this we vary the value of the follow components: C6, C7, R15, R16 and R18.
To ensure the output voltage was between 1V and -1V, the voltage divider composed by R17, R20, R19 and C9 was added.

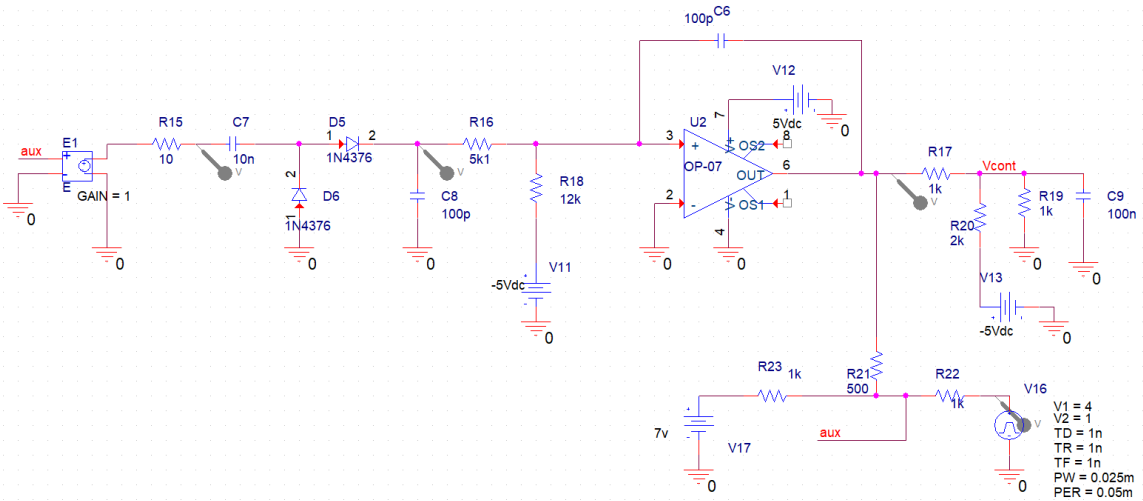


Figure 3.12: Signal rectifier and integrator simulation schematic

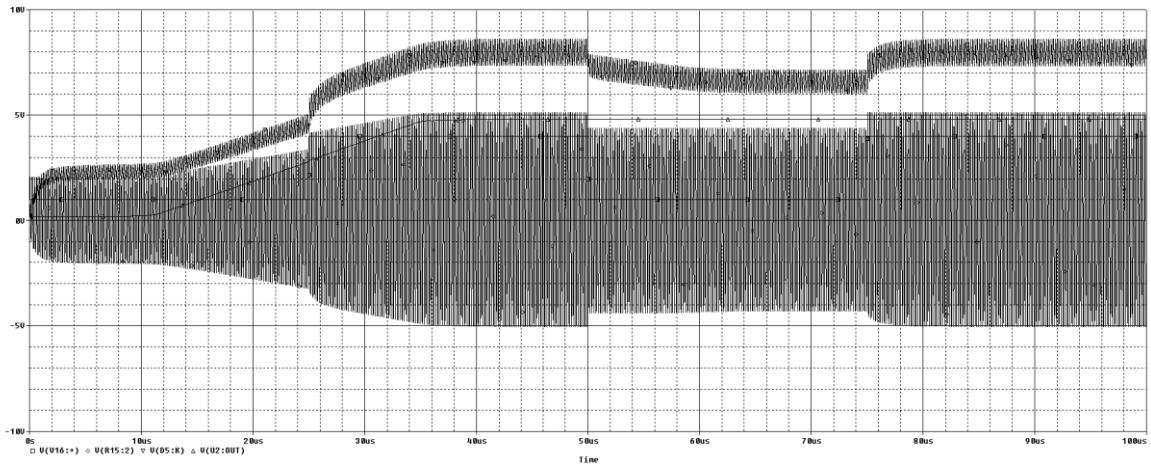


Figure 3.13: Signal rectifier and integrator simulation

Schematic

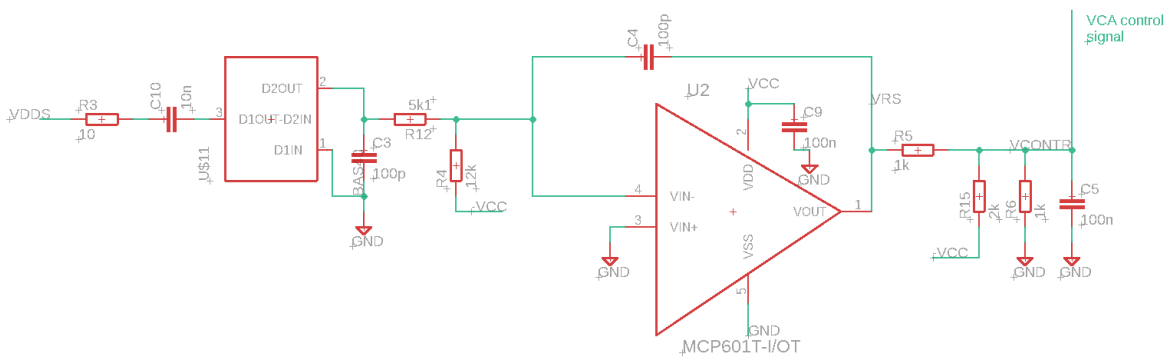


Figure 3.14: Signal rectifier and integrator simulation

3.3.5 Microcontroller

Options and Choose

	PIC24EP256GP206	PIC32MX154F128D	PIC18F66K40
Max CPU Speed	70 MIPS	116 MIPS	16 MIPS
Architecture	16 bit	32 bit	8 bits
UART	2	2	5
I2C	2	2	2
Timers	5x16bits, 2x32bits	5x16bits, 2x32bits	5x8bit, 4x16bits
Price (unit)	3.30 €	3.15 €	2.10 €

Table 3.2: Microcontroller comparison

After compare the 3 uC we decided about the PIC24EP256GP206, this was a hard decision because after discard the PIC18F66K40, because this low speed, the other 2 were pretty similar in characteristics. At the end we decided about it because the student got some knowledge with PIC24E family.

Schematic

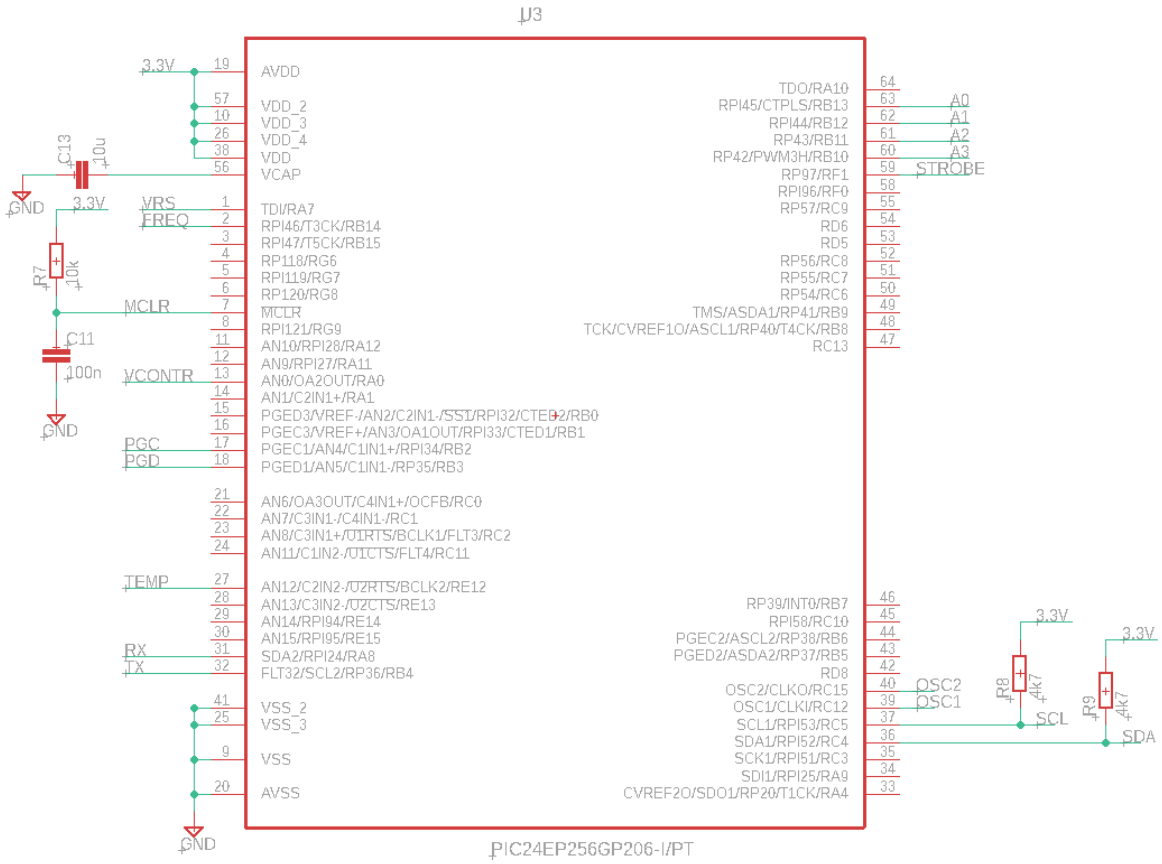


Figure 3.15: Microcontroller schematic

3.3.6 Buffer

Idea of usage as a "1-bit ADC"

As we have to include a buffer to protect the board from the non-communication signals coming from an external source, we got the idea of utilizing this buffer to convert our sinusoidal signal in a squared one.

Schematic

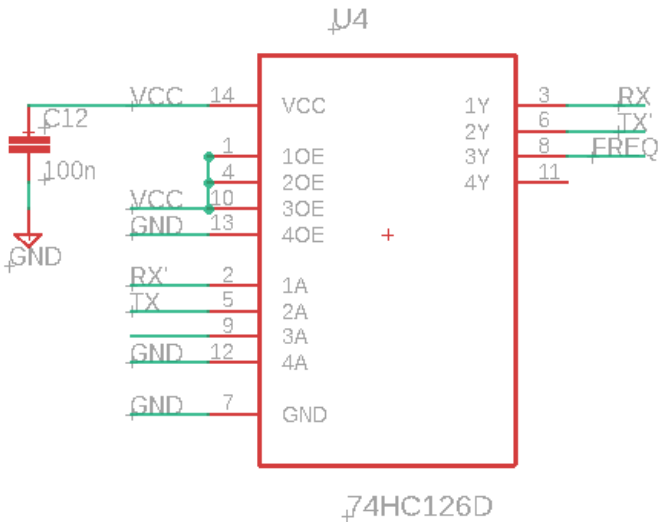


Figure 3.16: Buffer schematic

3.3.7 Data Storage (EEPROM)

As we want to storage the data calculated, we introduce to our design a EEPROM, with the EE2ROM we can save it preventing loss of it for a blackout for example. The EEPROM can be uses for other proposes like factory pre-configurations or save settings. I2C was the choose interface to communicate with the uC.

Schematic

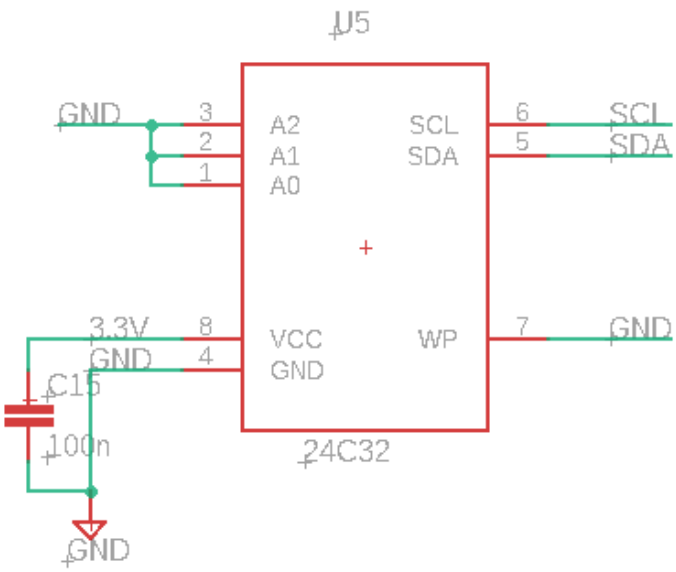


Figure 3.17: EEPROM schematic

3.3.8 Temperature sensor

As our system have to be hermetic and the temperature have to not vary at all, we introduce a temperature sensor to prove this characteristic. The temperature will be read between the measure of the last clock and the start of a new round.

Schematic

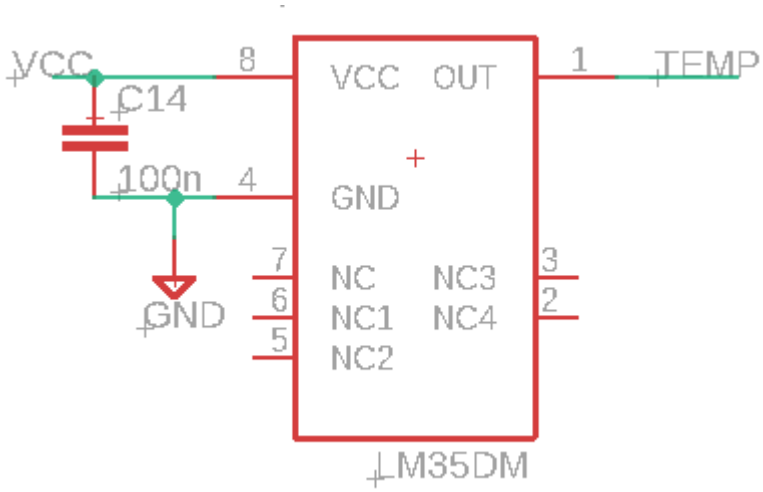


Figure 3.18: Temperature sensor schematic

3.3.9 3.3V alimentation

As all the integrated in the measure board was powered to 5V, was necessary the addition of a lineal regulator to supply the uC.

Schematic

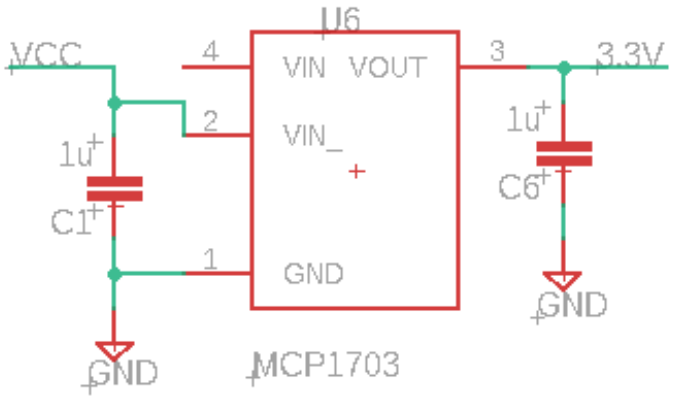


Figure 3.19: 3.3V alimentation schematic

3.3.10 PCB to PCB connectors

As the measure board will be attached to our plastic lid, and the power supply will be attached to the end of the thermostatic chamber we will need to connect both.

Our solution idea was create some big path on the measure board and connect to the power supply though a spring loaded pins

Figure 3.20: Spring connector

After some measurement and calculations, we end with the following PCB size and components place.

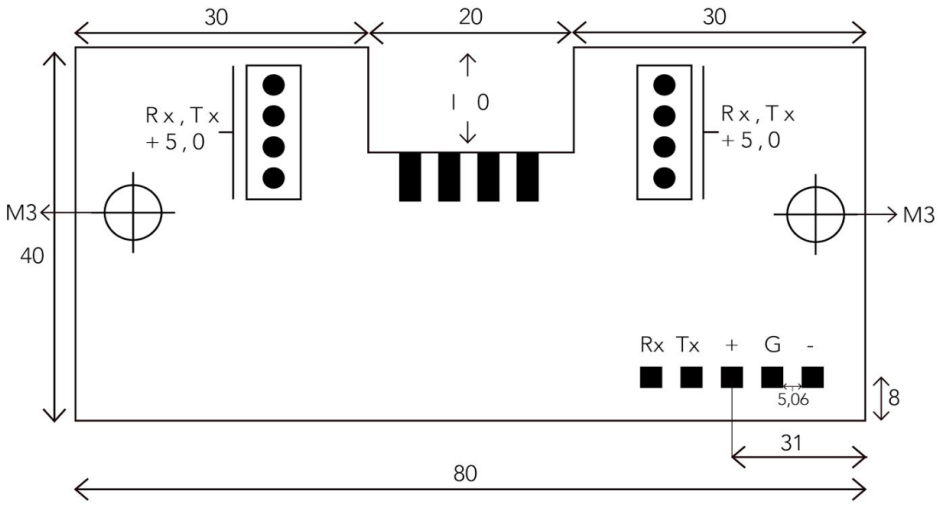


Figure 3.21: Interface and power supply board measurement

3.3.11 Power supply

With the blueprint designed and the last point, we designed the electronics for the board. This auxiliary board receive 5V from an external source and converts it in 5V and -5V using a DC/DC in the below configuration.

Schematic

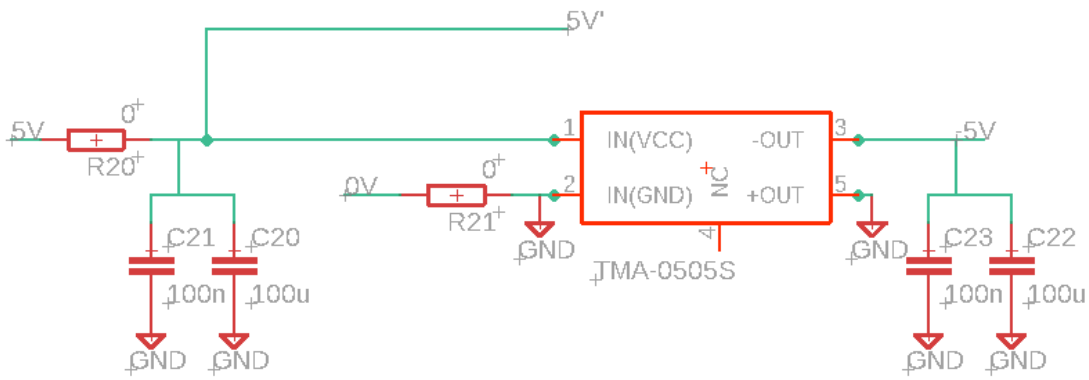


Figure 3.22: Power supply schematic

3.3.12 External connectors

In order to feed the power supply and send the data measured to a pc, those connectors were added to the “Interface and power supply board”.

The 0V and 5V pins will go to a power supply and the Rx and Tx pin will go to a UART to USB adapter like the image before.

Schematic

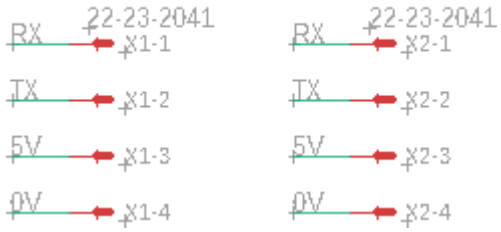


Figure 3.23: External connector schematic

3.4 Analog circuit start up

3.4.1 Welded PCB

Top measure board

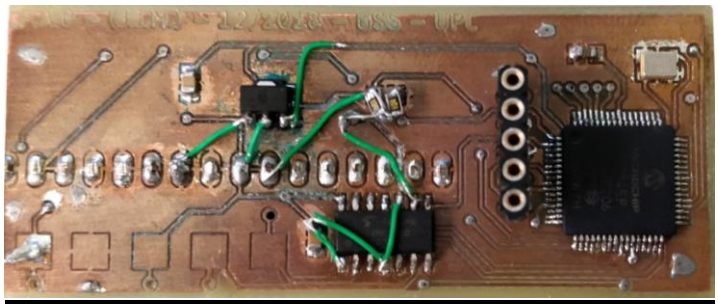


Figure 3.24: Weld top measure board

Botton measure board

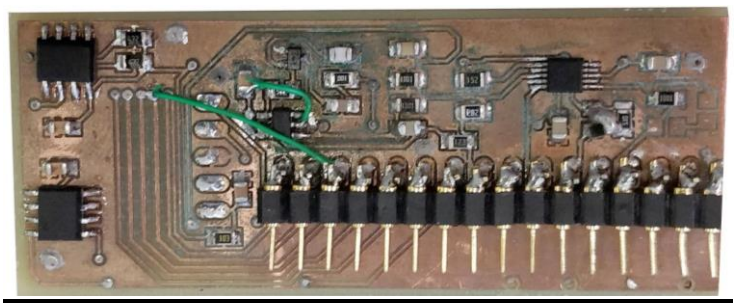


Figure 3.25: Weld bottom measure board

Top power supply board

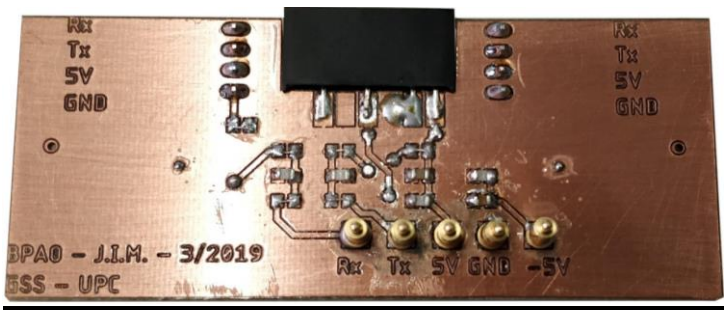


Figure 3.26: Weld top interface and power supply board

Bottom power supply board

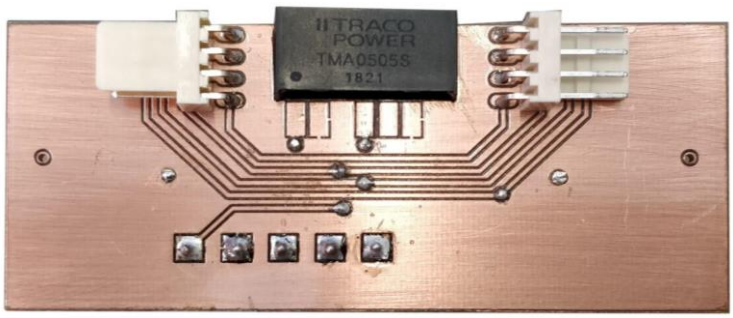


Figure 3.27: Weld bottom interface and power supply board

3.4.2 Test board

Schematic

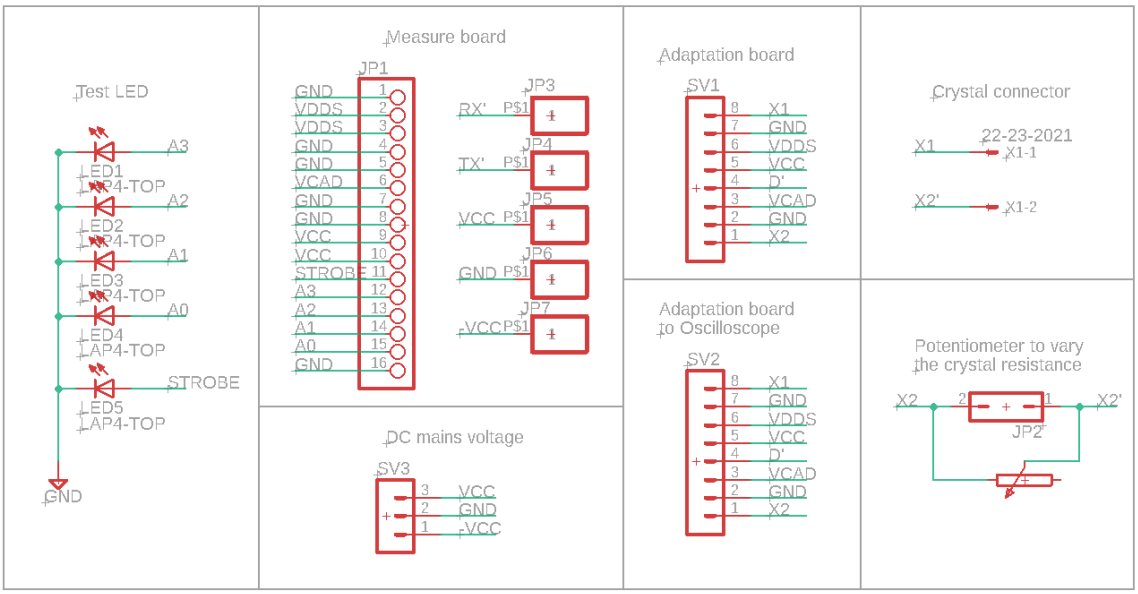


Figure 3.28: Test board schematic

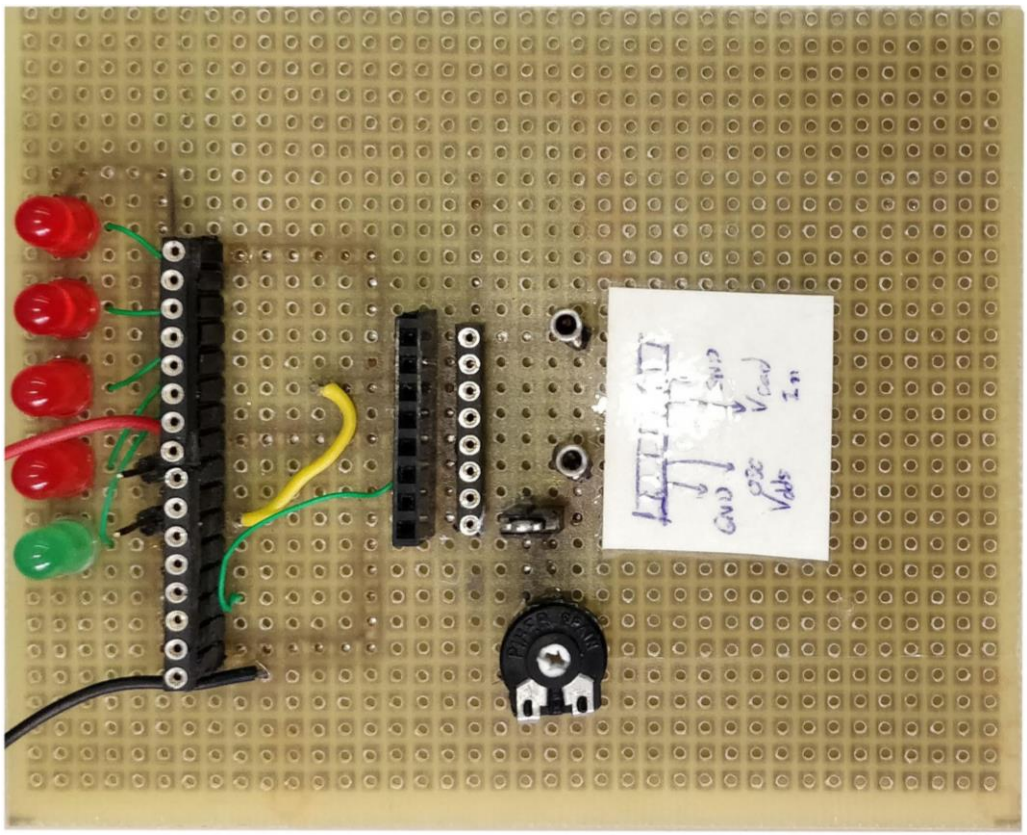


Figure 3.29: Weld test board schematic

In order to test the analog part of the pcb we just create a simple test board. This board connect our pcb, the adaptation board and the crystal, also with this we can change the resistance of the pcb changing potentiometer value.

3.4.3 Voltage controlled amplifier test

The first test was to check the VCA is fully operational. As is shown in the figure we disconnect the control pin of out circuit and we added a potentiometer, with this we can vary the gain, and in Vcad we just input a sinusoidal 5 MHz frequency and 1.5V amplitude in order to get 1V amplitude in Vcad.

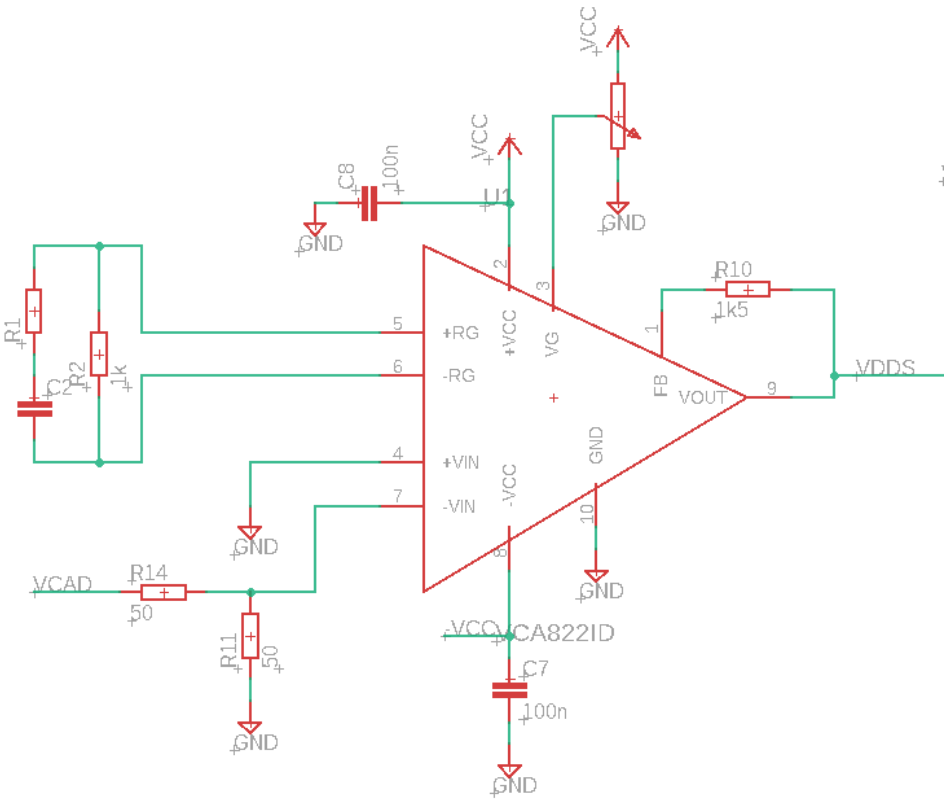


Figure 3.30: VCA test schematic

Vcontrol	Vout
-1 V	-12 mV
-0.5 V	653 mV
0 V	1.452 mV
0.5 V	2.13 V
1 V	2.71 V

Table 3.3: VCA test

3.4.4 Rectifier, integrator and VCA test

After test the integrator and rectifier and see that works as the simulation we decided to test all the circuit with different inputs so we to the VCA the function generator with a signal with frequency of 5 MHz and variable amplitude.

After the test we end with the following values:

Vin	Vout	Vcont	1/A
660mV	700 mV	487 mV	1.05
1 V	710 mV	79 mV	0.76
1.4 V	733 mV	-213 mV	0.55
2.2 V	750 mV	-521 mV	0.33

Table 3.4: Rectifier, integrator and VCA test

3.4.5 Test with crystal

After the individual test, we just proceed to make a full test, in this test we will try with a crystal with the following specs:

fs =

Rs =

After powering up the circuit and measuring the output signal we see that the crystal was oscillating in a 15 MHz frequency. After some test we realize that our circuit is making oscillate the crystal in the point that he got the minimum resistance.
The solution of our problem will be adding to the input port a LC filter.
We ended with two possible filter designs, as it shown in the next figures.

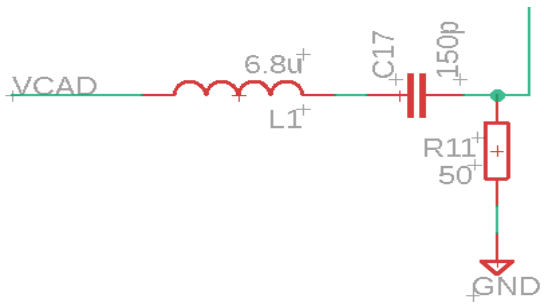


Figure 3.31: Input filter configuration 1

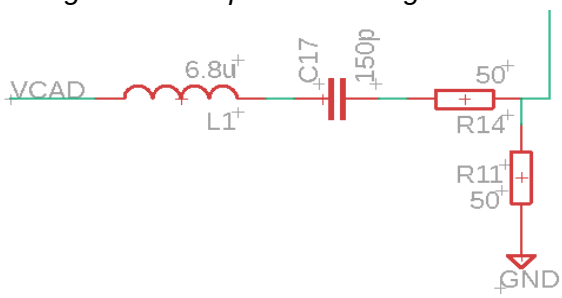


Figure 3.32: Input filter configuration 2

Q	10	5	3	10	5	3
L	32	15	10	7.5	5	7
C	32	68	100	64	130	200
R	100	100	100	50	50	50

Table 3.5: Input filter value option

We ended with:

L = 7.5 uH
C = 130 pF
R = 50 Ohm

This filter we allow as to remove the 15 MHz oscillation and as the input resistance is lower we got more gain due to the lower input resistance and we remove some signal losses.

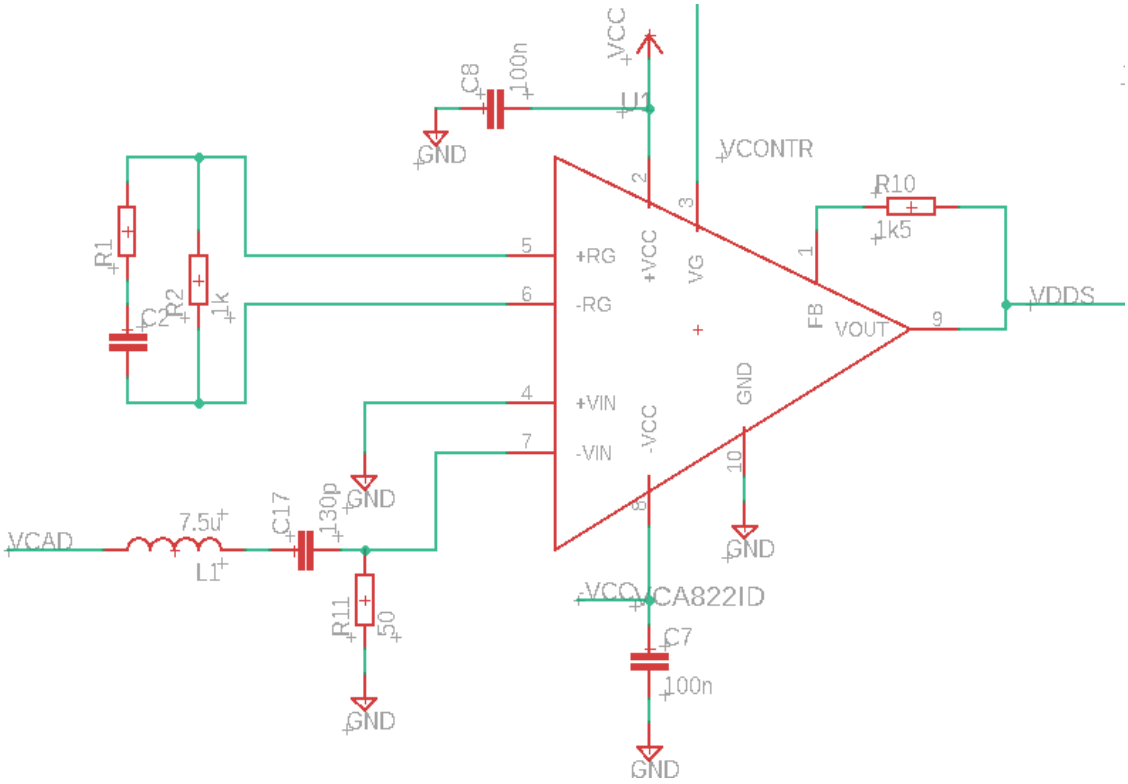


Figure 3.33: VCA rectified schematic

After this we placed the crystal and restart our test, as we expected we see the 15 MHz oscillation was removed and now we see the desired 5 MHz signal.
Next we
Finally, we changed the resistance value or out clock due the potentiometer, this were the results:

Potentiometer	0%	25%	50%	75%	100%
Top value	1.73 V	1.60 V	1.27 V	1.20 V	1.15 V
Bottom value	-1.77 V	-1.80 V	983 mV	-933 V	-983mV

Table 3.6: Crystal test

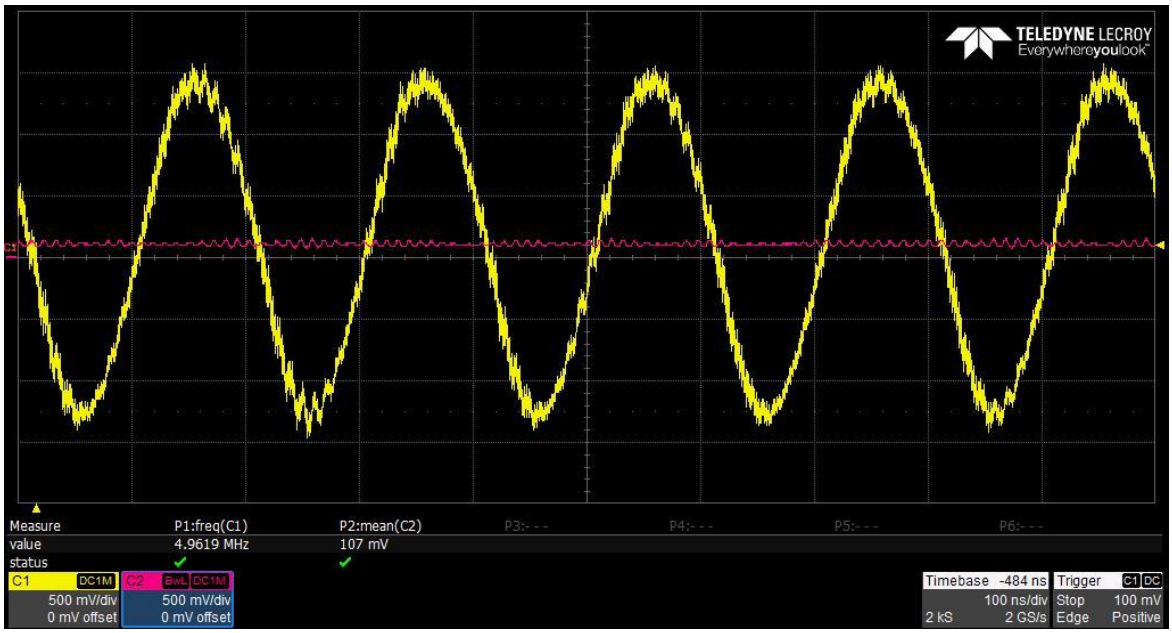


Figure 3.34: VDDS and Vcontrol measure

3.4.6 Buffer "1-bit ADC" test

After the last test we just see we don't satisfy the buffer specs, and we don't got any converted signal from the buffer.
In order to resolve this problem, we just add 2.5 Vdc to our signal, after including to the board a voltage divider in order to get the desired 2.5 Vdc and placing the output at the input of the buffer we got the problem solved and as it shows in the next figure the conversion was working fine.

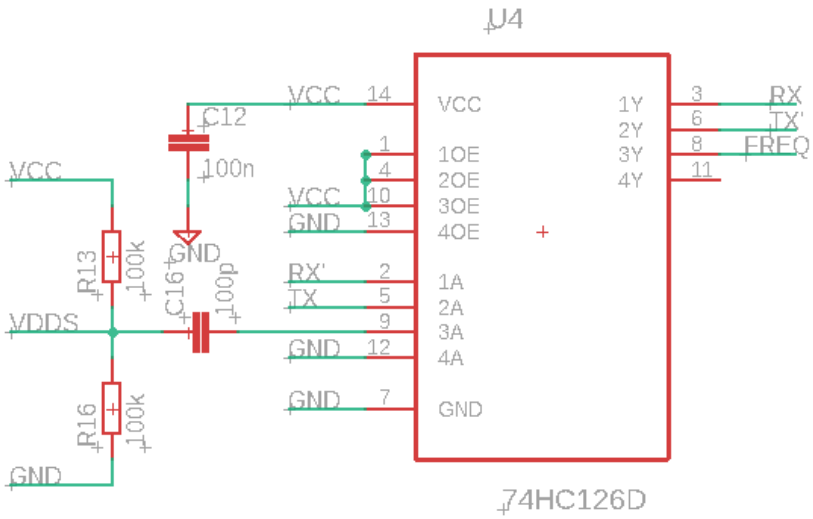


Figure 3.35: Buffer schematic

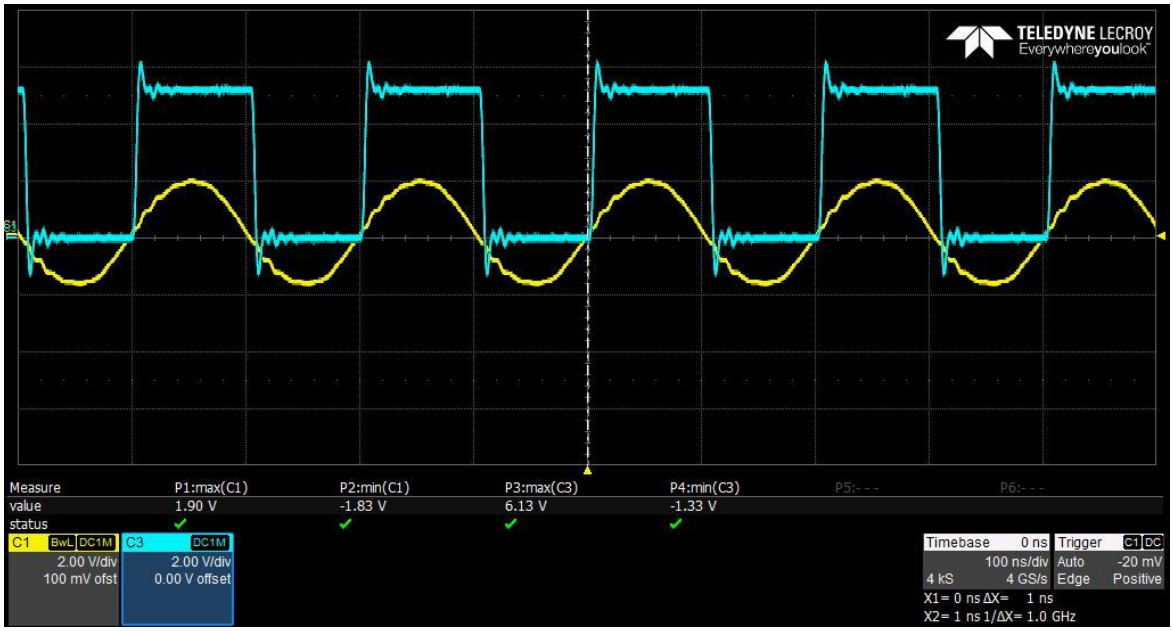


Figure 3.36: "1-bit ADC oscilloscope measure"

3.4.7 Temperature test

After checking all the analog part the analog temperature sensor will be the last and easiest to check, we just only power it up, measure the voltage in the output and convert the voltage in ºC with the formula given by the manufacturer.

Figure 3.37: Temperature sensor measure

$$T = \frac{265.9\ mV}{10\ \frac{mV}{ºC}} = 26.59\ ºC$$

After calculate the temperature measured we check it with a reference thermometer.

Temperature measured	Thermometer temperature
26.59 ºC	26.45ºC

Table 3.7: Temperature sensor test

3.5 Software

3.5.1. Code structure

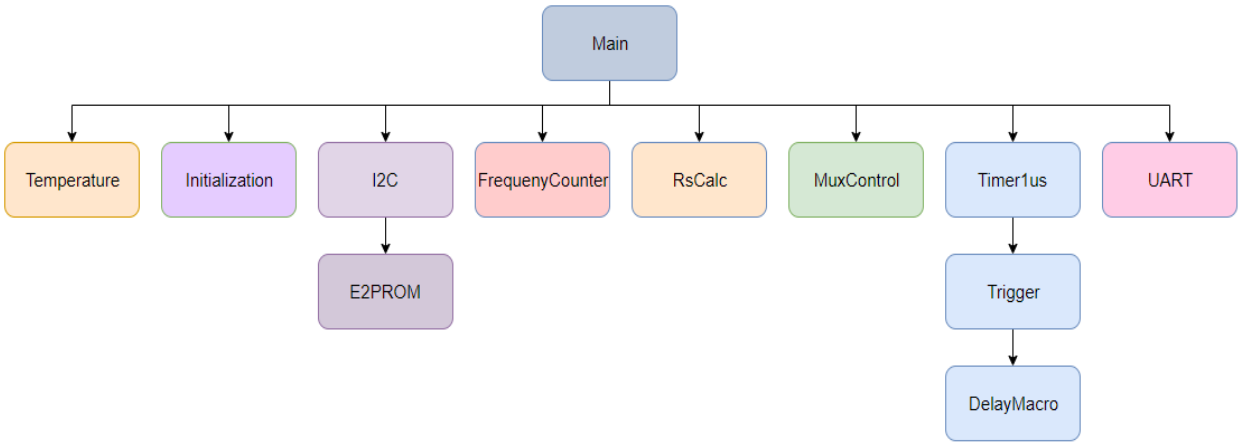


Figure 3.38: Code structure

Main: Principal method, in charge to use the following files and define the program functionality.

Initialization: Code for initialize and configure all the different settings and ports for the uC usage.

I2C: Define the I2C protocol.

E2PROM: Utilize I2C to define functions to send data to the E2PROM

FrequencyCounter: Define functions to calculate the oscillation frequency

RsCalc: Define functions to calculate the damping.

MuxControl: Define functions to control the multiplexor.

Timer1us: Define the interruption.

Trigger: Utilize the Timer1us to create timing functions

DelayMacro: Utilize the Trigger functions to define a Delay function

UART: Define the characteristics of UART port and implement functions to send data.

Temperature: Define functions to measure the temperature

3.5.2 Code flow diagram

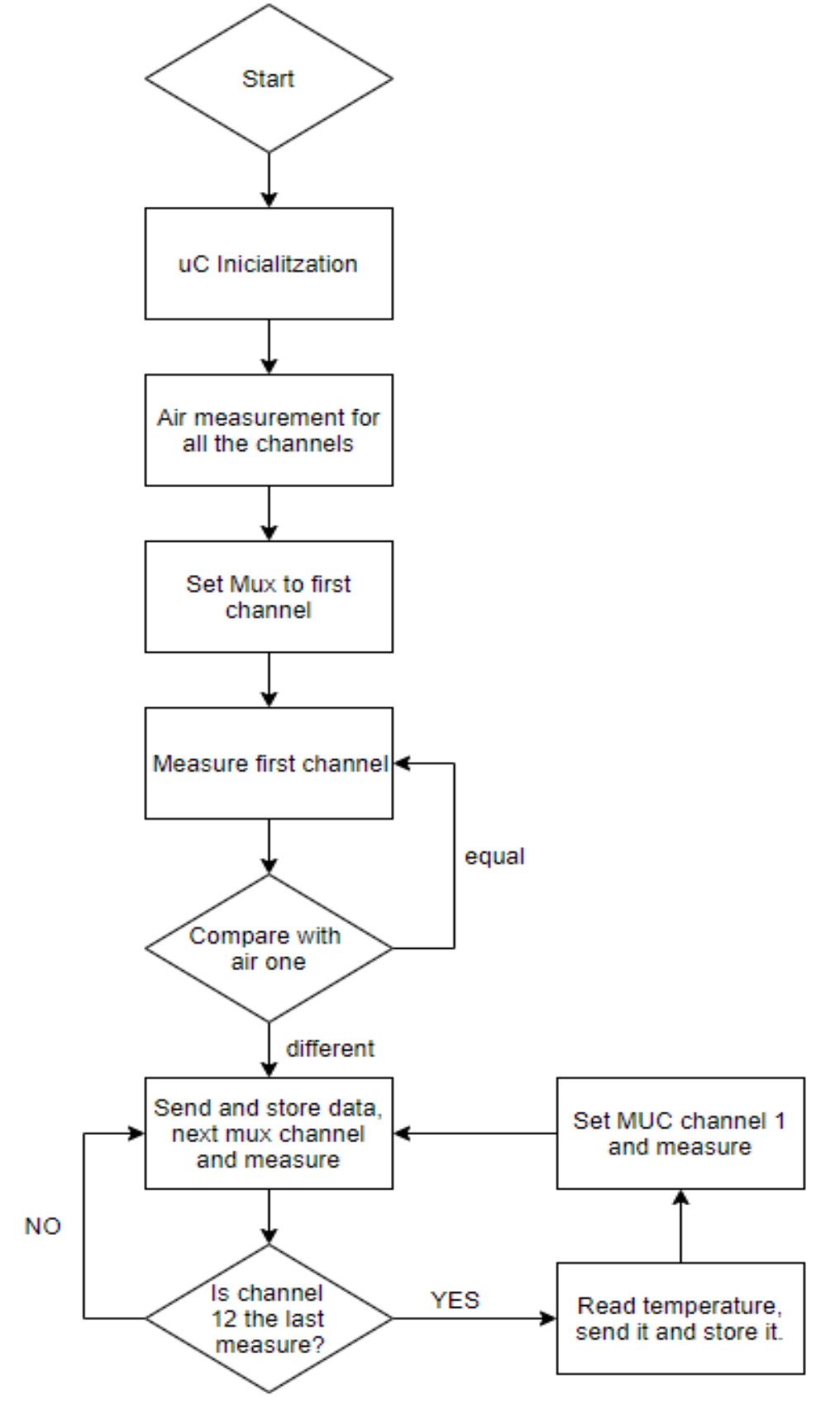


Figure 3.39: Code flow diagram

3.6 Measures

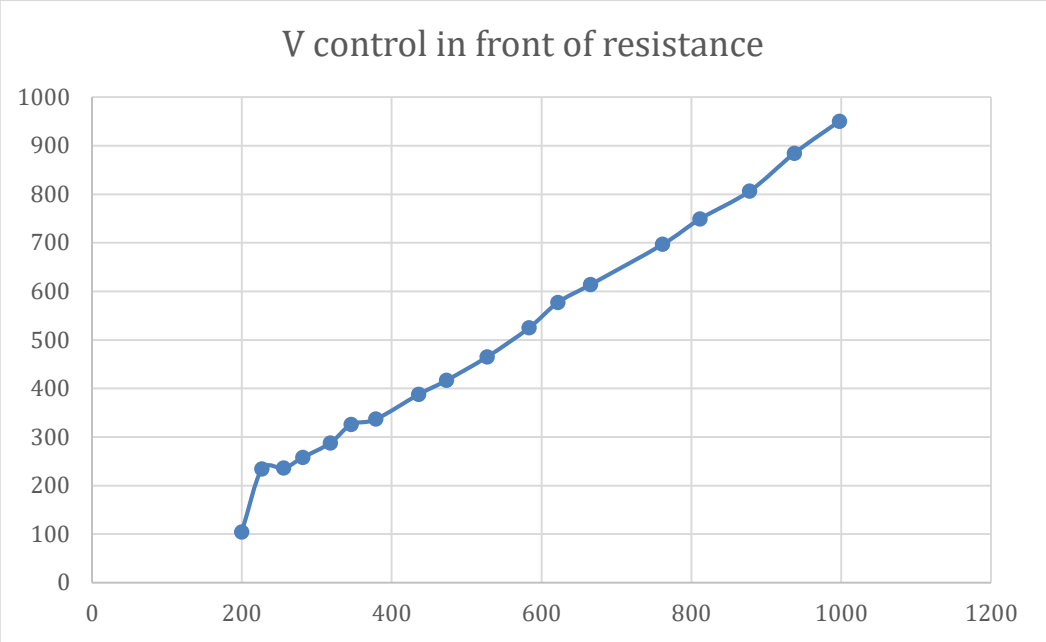


Figure 3.40: Vcontrol in front of resistance

After the measures we found a highly non-linearity response of Vcontrol in front of the different resistance, this result was closer as the expected one but, the addition of high frequency signals to the oscillation, make's it not valuable at all.

4. Results

Due the internal oscillations in the controlled gain loop produced by the nonlinearity of the VCA, thermal deviations, and extern noise/interferences near the oscillation point we got a constant variation of the oscillation so the VCA control signal is affected to this inputs producing no stability at all.

After some days trying to remove the external noise/interferences, characterize the nonlinearity of the VCA and the thermal deviation we realize that was a bigger theme and those can be another thesis itself.

So we ended with a base hardware system, capable to maintain the amplitude oscillation in the theory but in the real world with external noise/interferences it's more hard to make, and a base firmware capable to measure the frequency and the damping, store de data and send it to a pc.

5. Budget

5.1 Bill of materials

Component	Quantity	Unit cost	Subtotal
PIC24EP256GP206	1	3.30€	3.30€
X1E000021016412	1	0.72 €	0.72 €
MCP601	1	0.35 €	0.35 €
VCA822	1	6.73 €	6.73 €
BAS40	1	0.27 €	0.27 €
74HC126D	1	0.35 €	0.35 €
MCP1703	1	0.55 €	0.55 €
LM35DM	1	1.56 €	1.56 €
TMA0505S	1	3.51 €	3.51 €
7.7 kΩ	2	0.05 €	0.10 €
10 kΩ	1	0.05 €	0.05 €
100 kΩ	2	0.05 €	0.10 €
50 Ω	1	0.05 €	0.05 €
1 kΩ	3	0.05 €	0.15 €
1.5 kΩ	1	0.05 €	0.05 €
10 Ω	1	0.05 €	0.05 €
5.1 kΩ	1	0.05 €	0.05 €
12 kΩ	1	0.05 €	0.05 €
2 kΩ	1	0.05 €	0.05 €
100 pF	4	0.05 €	0.7 €
100 uF	4	0.05 €	0.20 €
100 nF	7	0.05 €	0.35 €
10 uF	1	0.05 €	0.05 €
1 uF	2	0.05 €	0.10 €
150 pF	1	0.05 €	0.05 €

10 nF	1	0.05 €	0.05 €
6.8uH	1	0.45€	0.45 €
Spring Connectors	5	0.70€	3.50 €
		Total (Subtotal + 21%IVA)	27.51€

Table 5.1: BOM

5.2 Design and prototyping costs

Task	Price + 21%IVA	Time	Subtotal
Orcad	58 €/month	2 month	116 €
Design and simulate	8 €/h	60 h	480 €
Eagle	70 €/month	4 month	280 €
PCB design	8 €/h	80 h	640 €
PCB manufacturing	140 €	--	140 €
Weld	8 €/h	10 h	80 €
Test	8 €/h	50h	400 €
MPLAB X IDE	0€	--	0 €
PICKIT 3	57 €	--	57 €
Developing software	8 €/h	100 h	800 €
		Total	2993 €

Table 5.2: Design and prototyping cost

5.3 Economical/financial viability

As typically the laboratory equipment is very expensive it can be sold in a reasonable price and recover the initial inversion.

If we produce 150 and we sell 100 at 200 €/unit, we will got the following benefits:

$$Benefits = 200\ \text{"€/unit"} * 100\ \text{"units"} - 2993\ \text{"prototype cost"} - (27.51\ \text{"components"} + 0.15\ \text{"pcb serie manufacture price"}) * 150\text{"produced units"} = 12858\ €$$

6. Conclusions and future development:

As I mentioned in the point 4 “Results” due the external noise/interferences was impossible to get measures in the desired specifications. But we give a good base to work more in this project.

A future development will be:

Reforce the system to those interferences

Characterize the non-linearity of the VCA822

Characterize the thermal deviation of the VCA822

Substitute the VCA822 an analog VCA for a digital one, (when available) and make the VCA control through the microcontroller

Decrease the internal interference loops, in order to improve resistance measurement linearity

Bibliography:

A thorough reference list as in the following examples: Conference paper [1], journal paper [2], book [3], standard-1 [4], standard-2 [5], online reference [6], patent [7], M.S. thesis [8] and Ph.D. dissertation [9].

[1] Luis Berlanga Herrera, Master thesis, “Real-Time multichannel systems for the monitorization of biofilm formation on quartz crystal resonators”, February 2018

[2] PIC24EP256GP206 datasheet, Microchip, August 2013. [Online] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70000657H.pdf [Accessed: 7 May 2019]

[3] VCA822 datasheet, Texas Instruments, October 2015. [Online] Available: https://www.ti.com/lit/ds/symlink/vca822.pdf [Accessed: 7 May 2019]

[4] MCP601 datasheet, Microchip, Desember 2007. [Online] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/21314g.pdf [Accessed: 7 May 2019]

[5] CAV24C32 datasheet, OnSemiconductor, Desember 2007. [Online] Available: https://www.onsemi.com/pub/Collateral/CAV24C32-D.PDF [Accessed: 7 May 2019]

[6] 74HC126D datasheet, Toshiba, Desember 2007. [Online] Available: https://toshiba.semicon-storage.com/info/docget.jsp?did=37028&prodName=74HC126D [Accessed: 7 May 2019]

[7] LM35DM datasheet, Texas Instruments, Desember 2007. [Online] Available: http://www.ti.com/lit/ds/symlink/lm35.pdf [Accessed: 7 May 2019]

[8] BAS40 datasheet, Viashay, Desember 2007. [Online] Available: https://www.vishay.com/docs/85701/bas40v.pdf [Accessed: 7 May 2019]

[9] MCP1703 datasheet, Microchip, Desember 2007. [Online] Available: http://ww1.microchip.com/downloads/en/devicedoc/22049e.pdf [Accessed: 7 May 2019]

Appendices:

A Tools

A.1 Simulation Environment

OrCad is a free licence software for students, with OrCad can design and simulate the circuit schematic.

A.2 Design environment

Eagle is a free licence software for students, with eagle can design the circuit schematic and after validating the correct fintionament, can be linked with LTC Spice, finally we can set the main settings like, clearance, pcb layers, copper width, etc and root our PCB.

A.3 Weld tools

Nae 2B is a welder station, fabricated by JCB. In this project was used mainly to solder all the SMD components and the test boards.

The 2311402 is a air welder station fabricated by Tenma. The usage in this project was mainly solder SMD components imposible to weld by the standar welder like the crystal used in the uC.

The NZ-14B is a microscope fabricated by Iroscope. The usage in this project was give the capability to see what we are welding and optic inspection.

A.4 Inspection tools

The Wavesurfer 3054 is a oscilloscope fabricated by Teledyne. The main usage in this project was for test/validate the correct functioning from the designed circuitry.

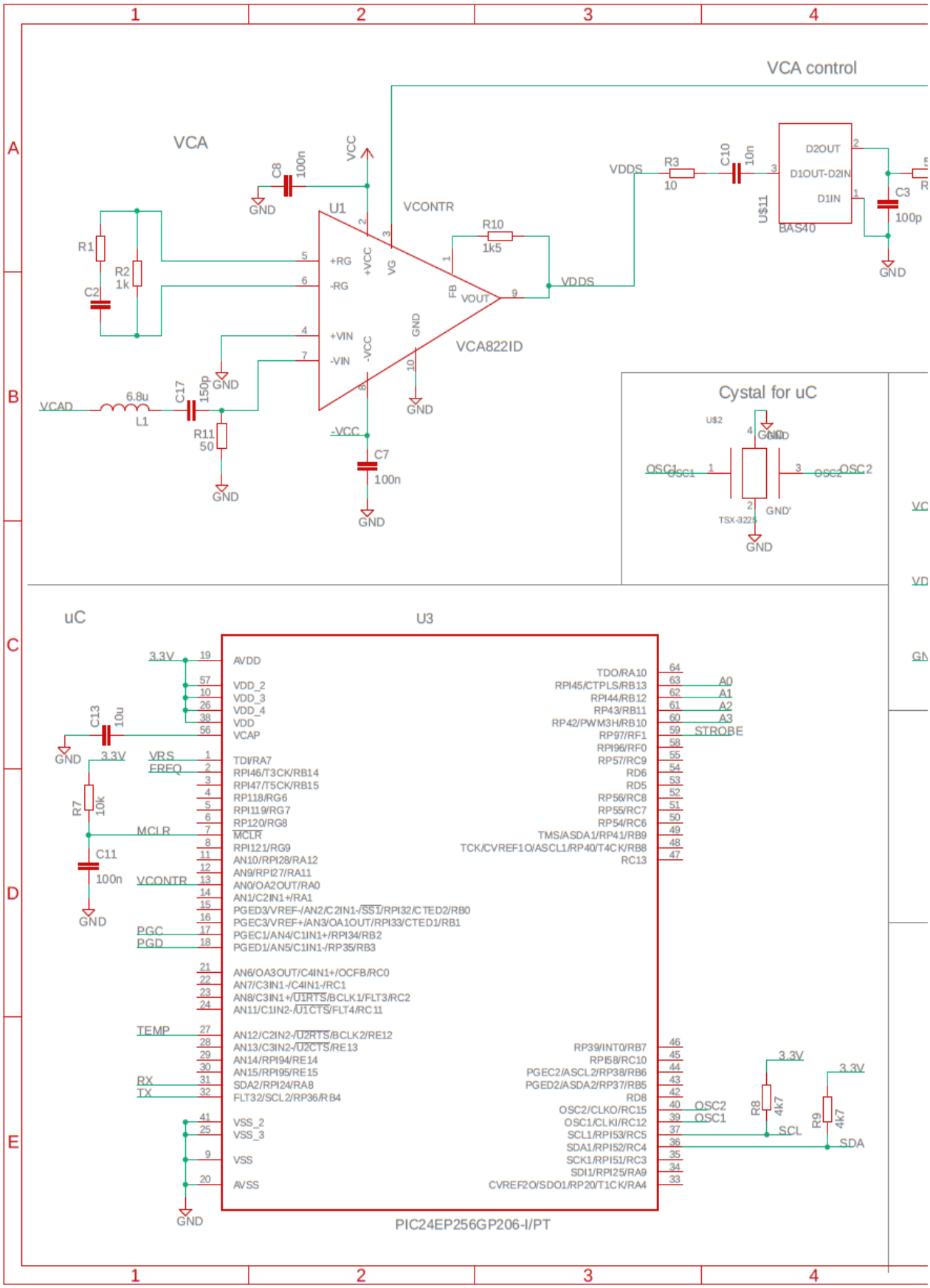
The E3631 is a DC Power Supply fabricated by Agilent. The main usage in this project was giving 5 V and -5V with limitation in current, for all the test.

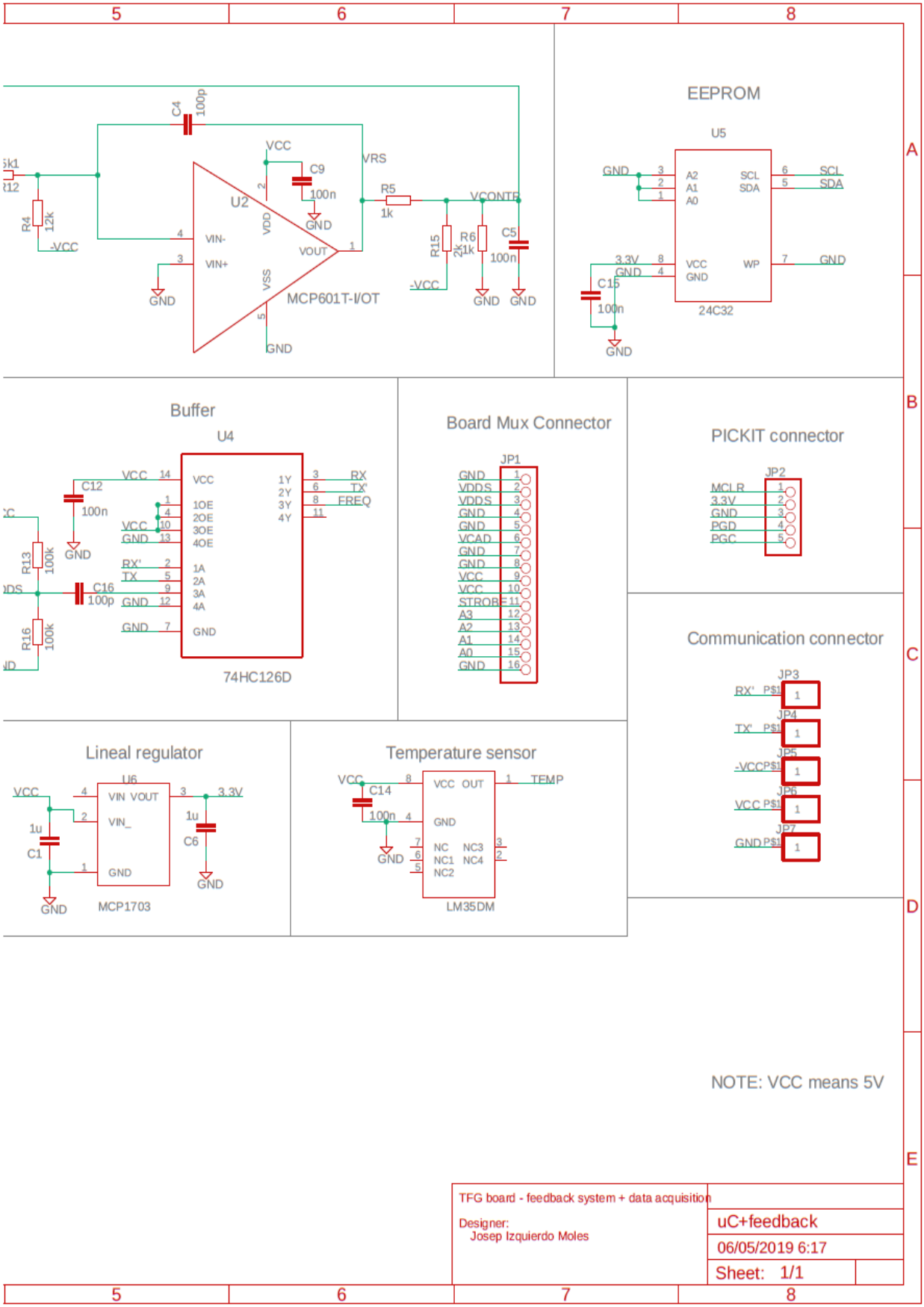
The 33522A is a function generator fabricated by Agilent. The main usage in this project was give a sinusoidal input signal to replace the crystal in some test.

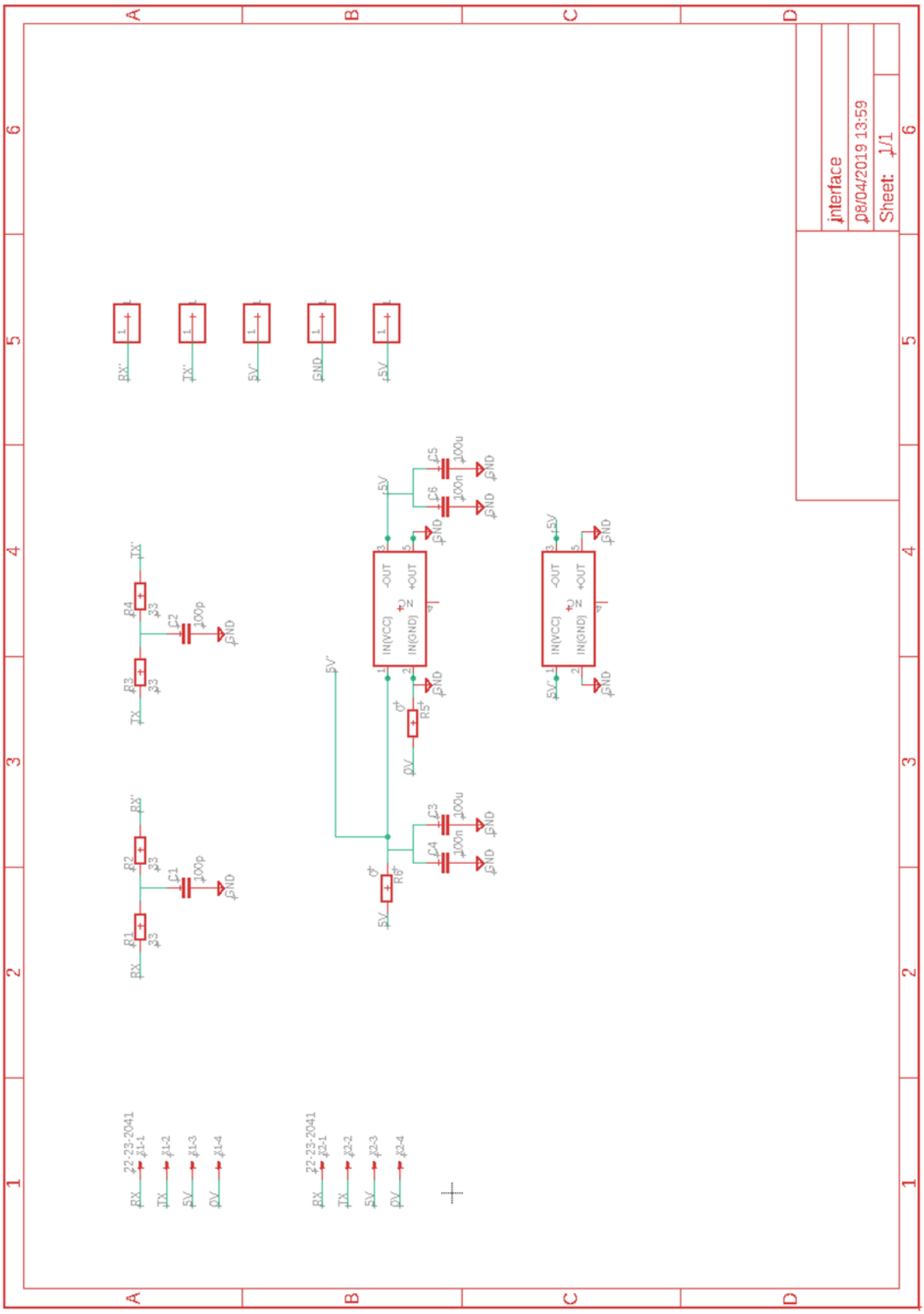
A.5 Development tools

The MPLAB is a free software provided by microchip, it uses PICkit 3 to program the devices manufactured by the same company.

B Electric schemes

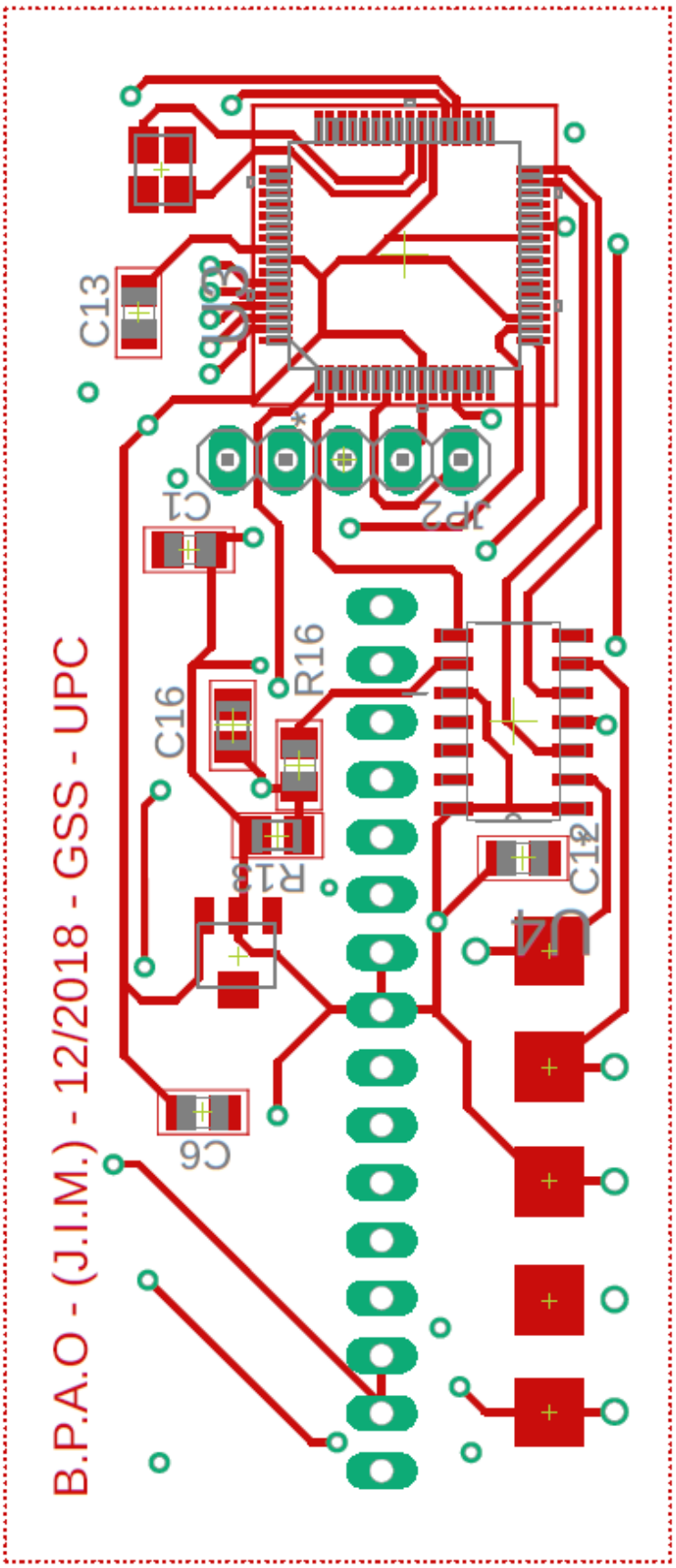



5
6
7
8
A
B
C
D
E
EEPROM
U5
GND
A2
A1
A0
SCL
SDA
VCC
WP
24C32
3.3V
C15
100n
C4
100p
VCC
C9
100n
VRS
R5
1k
VCONTR
U2
VDD
VSS
VIN-
VIN+
VOUT
MCP601T-I/OT
R4
12k
-VCC
R15
R6
21k
C5
100n
Buffer
U4
74HC126D
C12
100n
R13
100k
C16
100p
R16
100k
RX'
TX'
FREQ
Board Mux Connector
JP1
PICKIT connector
JP2
MCLR
3.3V
PGD
PGC
Communication connector
JP3
JP4
JP5
JP6
JP7
Lineal regulator
U6
VIN
VOUT
MCP1703
1u
C1
C6
3.3V
Temperature sensor
C14
100n
TEMP
LM35DM
NOTE: VCC means 5V
TFG board - feedback system + data acquisition
Designer:
Josep Izquierdo Moles
uC+feedback
06/05/2019 6:17
Sheet: 1/1

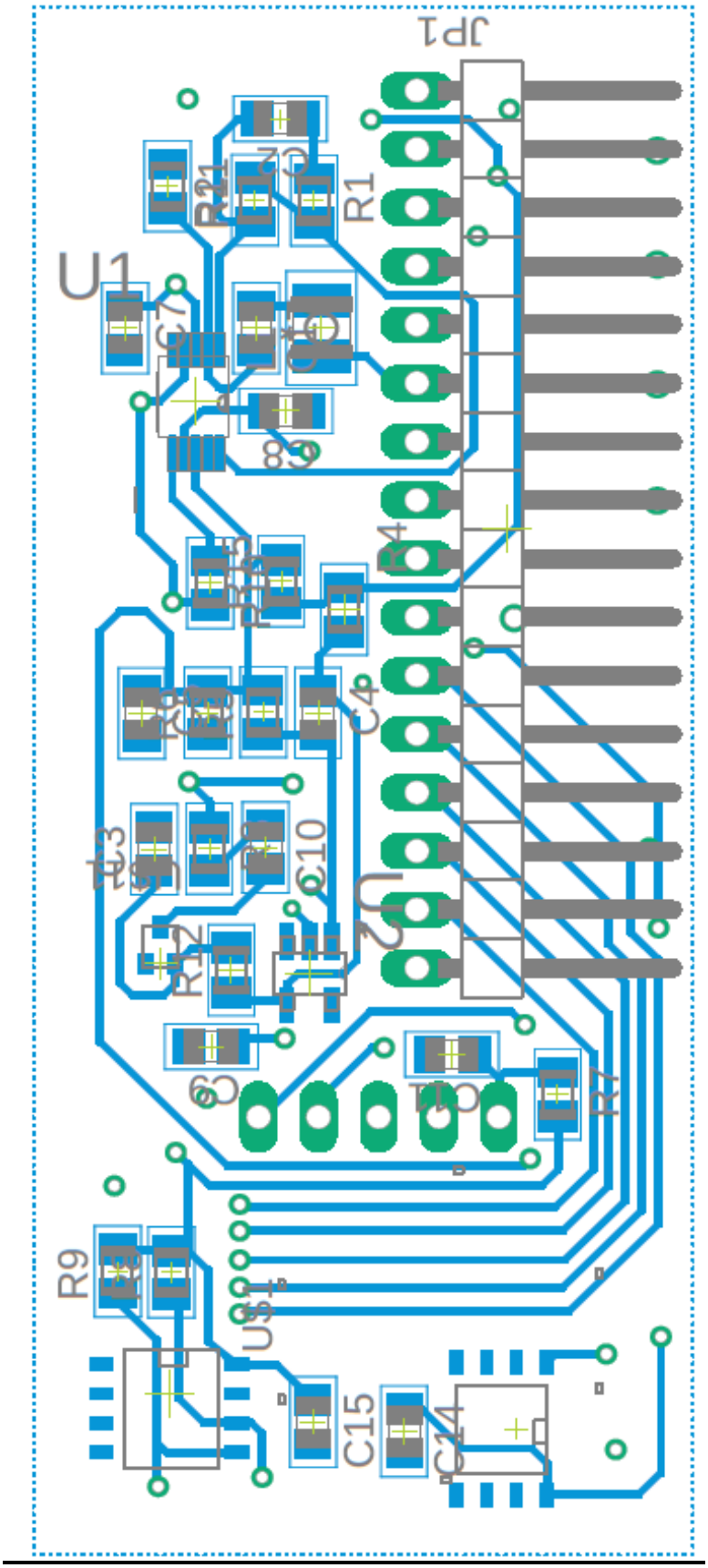

interface
08/04/2019 13:59
Sheet: 1/1
RX'
TX'
5V'
GND
+5V
R4
R3
R2
R1
C2
C1
C5
C6
C3
C4
R5
R6
100p
100u
100n
IN(VCC)
IN(GND)
-OUT
+OUT
22-23-2041
J1-1
J1-2
J1-3
J1-4
J2-1
J2-2
J2-3
J2-4

C Layout

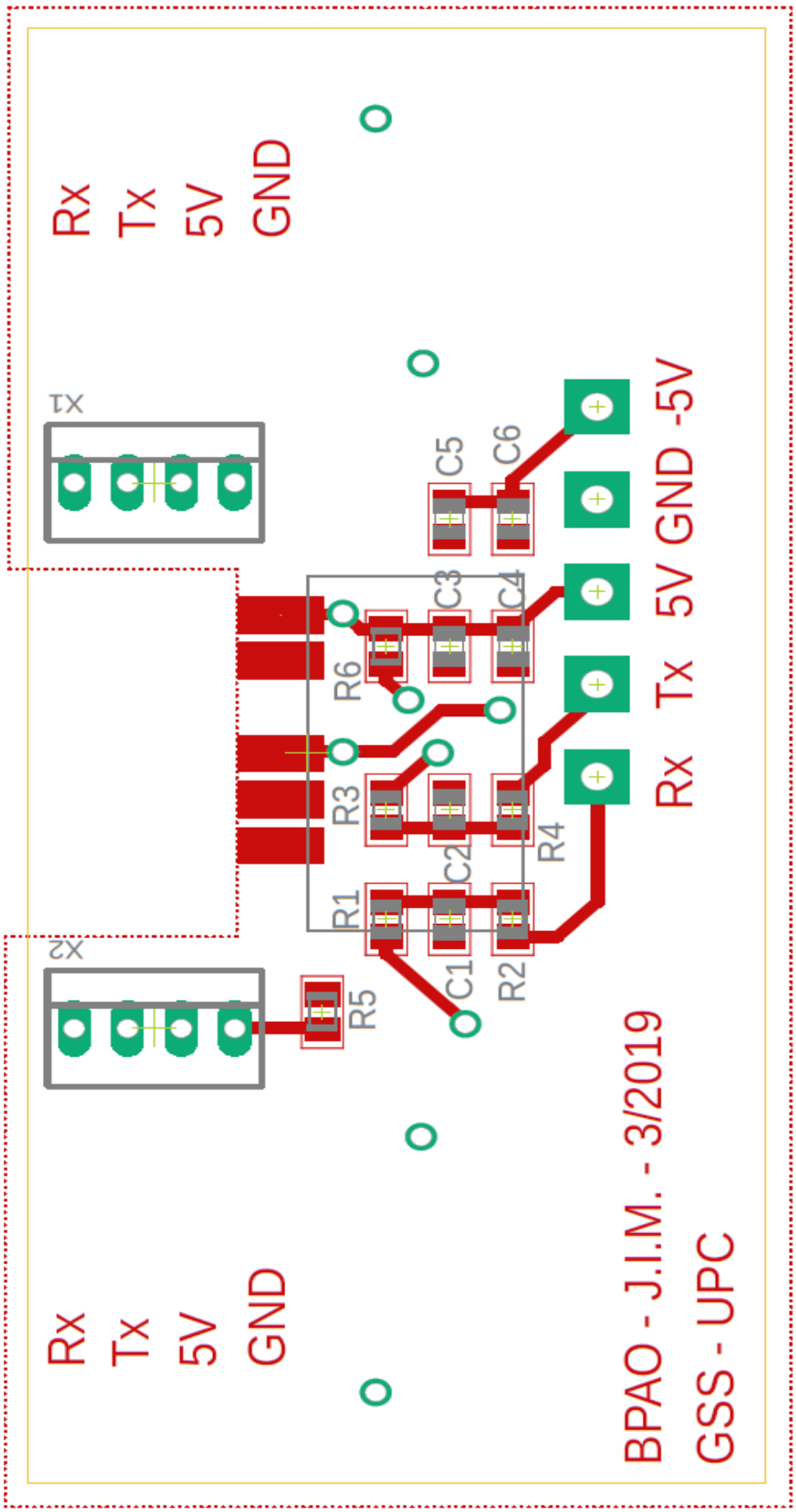
C.1. Measure circuit top layer



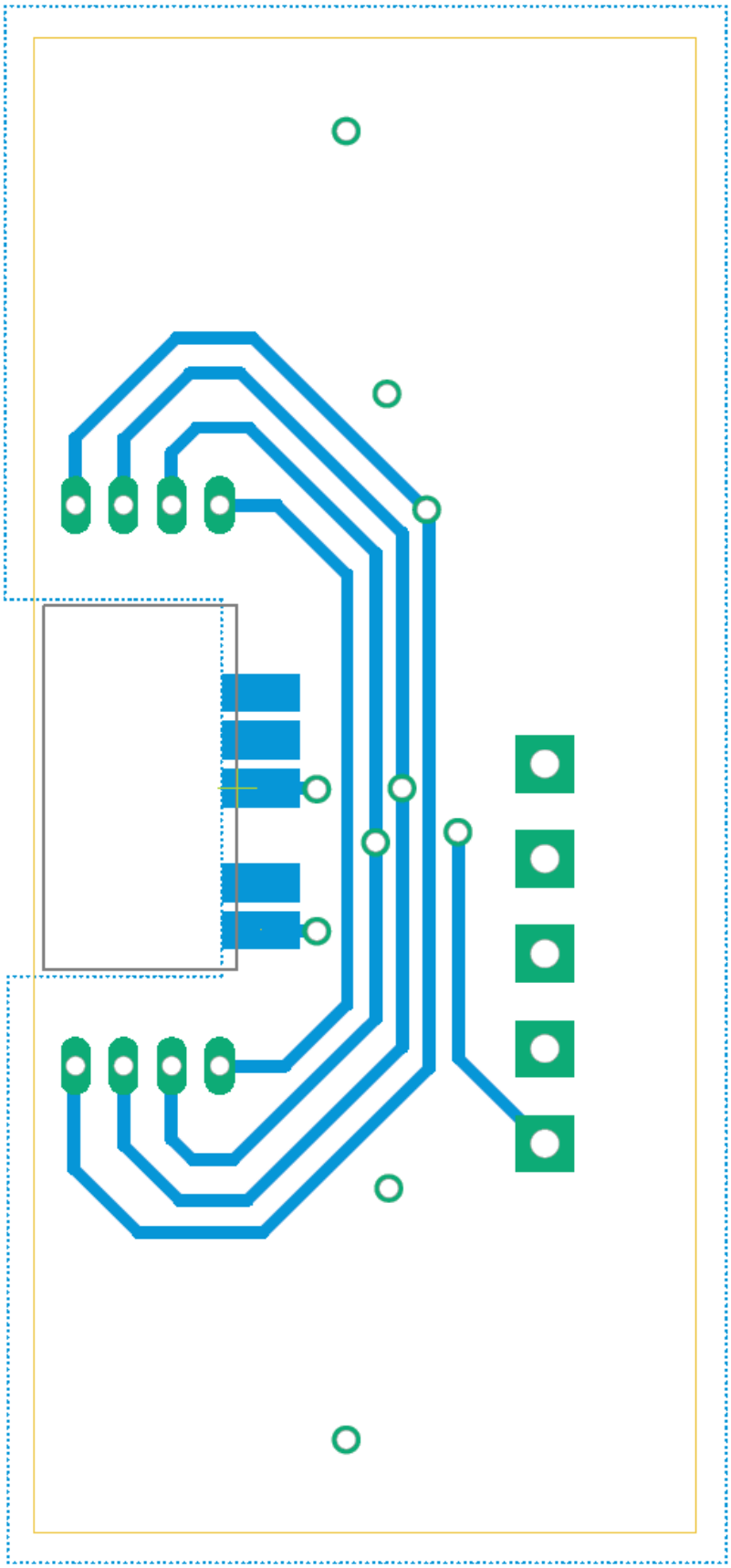
C.2. Measure circuit bottom layer



C.3. Power supply and connection board top layer



C.4. Power supply and connection board bottom layer



D Firmware

D.1. main.c

```
// FICD
#pragma config ICS = PGD1          // ICD Communication Channel Select bits (Communicate on PGEC1 and PGED1)
#pragma config JTAGEN = OFF        // JTAG Enable bit (JTAG is disabled)

// FPOR
#pragma config ALTI2C1 = OFF       // Alternate I2C1 pins (I2C1 mapped to SDA1/SCL1 pins)
#pragma config ALTI2C2 = OFF       // Alternate I2C2 pins (I2C2 mapped to SDA2/SCL2 pins)
#pragma config WDTWIN = WIN25      // Watchdog Window Select bits (WDT Window is 25% of WDT period)

// FWDT
#pragma config WDTPOST = PS32768   // Watchdog Timer Postscaler bits (1:32,768)
#pragma config WDTPRE = PR128      // Watchdog Timer Prescaler bit (1:128)
#pragma config PLLKEN = ON         // PLL Lock Enable bit (Clock switch to PLL source will wait until the PLL lock signal is valid.)
#pragma config WINDIS = OFF        // Watchdog Timer Window Enable bit (Watchdog Timer in Non-Window mode)
#pragma config FWDTEN = ON         // Watchdog Timer Enable bit (Watchdog timer always enabled)

// FOSC
#pragma config POSCMD = HS         // Primary Oscillator Mode Select bits (HS Crystal Oscillator Mode)
#pragma config OSCIOFNC = OFF      // OSC2 Pin Function bit (OSC2 is clock output)
#pragma config IOL1WAY = ON        // Peripheral pin select configuration (Allow only one reconfiguration)
#pragma config FCKSM = CSECMD      // Clock Switching Mode bits (Both Clock switching and Fail-safe Clock Monitor are disabled)

// FOSCSEL
```

```
#pragma config FNOSC = PRI          // Oscillator Source Selection (Primary Oscillator (XT, HS, EC))
#pragma config IESO = OFF           // Two-speed Oscillator Start-up Enable bit (Start up with user-selected oscillator source)

// FGS
#pragma config GWRP = OFF           // General Segment Write-Protect bit (General Segment may be written)
#pragma config GCP = OFF            // General Segment Code-Protect bit (General Segment Code protect is Disabled)

#include "GlobalDefines.h"
#include "main.h"
#include "Initialization.h"
#include "MuxControl.h"
#include "FrequencyCounter.h"
#include "RsCalc.h"
#include "E2PROM.h"
#include "UART.h"
#include "temperature.h"

#define TIMER_SET 10000000
#define E2PROM_ADDRESS  0b10100000

uchar i = 0;
uint auxFrq = 0, e2promCounter = 0;
float auxTmp = 0, auxRs = 0;
float airMeasures[12]= {0};

int main(void) {
    SetQCM(0);
    SendString("AIR measures \r\n\0");
    for(i=0;i<12;i++){
        airMeasures[i] = measureFreq();
        NextQCM();
```

```
        SendData(airMeasures[i]);
    }
    SetQCM(0);
    while(airMeasures[0] == measureFreq())
        continue;
    SetQCM(0);
    SendString("Load measures \r\n\0");
    do{

        auxFrq = measureFreq();
        auxRs = calcRs();
        eepromWrite(e2promCounter, E2PROM_ADDRESS, 2, auxFrq);
        eepromWrite(e2promCounter +1, E2PROM_ADDRESS, 2, auxRs);
        SendData(auxFrq);
        SendData(auxRs);
        NextQCM();
        e2promCounter = e2promCounter + 2;
        if(e2promCounter%12 == 0)
            SetQCM(0);
            auxTmp = readTemp();
            eepromWrite(e2promCounter +1, E2PROM_ADDRESS, 2 , auxTmp);
    }while(true);

    return 1;
}
```

D.2. initialization.h

```
#include "GlobalDefines.h"

void mainInitialization();
void initPLL();
void initTimer1us();
void initInterrups();
void initMux();
void initE2prom();
```

```
void initFreq();
void initADC();
```

D.3. initialization.c

```
#include "Initialization.h"

void mainInitialization(){
    initPLL();
    initInterrups();
    initTimer1us();
    initADC();
    initE2prom();
    initMux();
    initFreq();
}

void initPLL(){
    /*
     * fosc = fin*(M/(N1*N2)) = 140Mhz -> 70MIPS
     */
    PLLFBD=56; // M=40
    CLKDIVbits.PLLPOST=0; // N2=2
    CLKDIVbits.PLLPRE=0b00011; // N1=5

    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(OSCCON | 0x01);

    while (OSCCONbits.COSC!= 0b011); // Wait for Clock switch to occur
    while (OSCCONbits.LOCK!= 1);    // Wait for PLL to lock
}

void initTimer1us(){
    TMR1 = 0x00; //set Timer1 to 0
    PR1 = 140; // interrupt every   1us, fosc = 140Mhz
    IPC0bits.T1IP = 0x01; // Set Timer 1 Interrupt Priority Level
```

```
    IFS0bits.T1IF = 0; // Clear Timer 1 Interrupt Flag
    IEC0bits.T1IE = 1; // Enable Timer1 interrupt
    T1CONbits.TON = 1; // Start Timer
}

void initInterrups(){
    INTCON2bits.GIE = 1; //Enable interrups generaly
    INTCON1bits.NSTDIS = 0; //Interrupt nesting enabled

}

void initMux(){
    TRISBbits.TRISB10 = 0;  //A3
    TRISBbits.TRISB11 = 0;  //A2
    TRISBbits.TRISB12 = 0;  //A1
    TRISBbits.TRISB13 = 0;  //A0
    TRISFbits.TRISF1 = 0;   //Strobe

}

void initE2prom(){
    I2C1BRG = 166; // Baudrate generator for I2C protocol. Given Fclk = 140MHz,
Delay = 100 - 130 ns and Fscl = 400KHz --> I2C1BRG = (((1/Fscl) - Delay)*(Fclk/2))-
2; For 100 ns --> 28
    I2C1CONbits.I2CEN = 1;
}

void initFreq(){
    TRISFbits.TRISF0= 1;
}

void initADC(){
    ANSELA = 0x0000;
    ANSELB = 0x0000;
    ANSELC = 0x0000;
```

```
    ANSELE = 0x0000;

    ANSELEbits.ANSE12 = 1;  // Temperature sensor
    ANSELAbits.ANSA0 = 1;   // Rs calc

    AD1CON1 = 0x000C;   // Enable simultaneous sampling and auto-sample
    AD1CON2 = 0x051D;   // Select 2-channel mode, enable both scanning and
alternate sampling
    AD1CON3 = 0x000F;
    AD1CON4 = 0x0000;
    AD1CHS0 = 0x0005;
    AD1CHS123 = 0x0000;
    AD1CSSH = 0x0000;
    AD1CSSL = 0x1001;

    AD1CHS0bits.CH0SA = 0;  // Select AN0 for CH0 +ve input
    AD1CHS0bits.CH0SA = 12; // Select AN12 for CH0 +ve input

    AD1CON1bits.ADON = 1;
}
```

D.4. DelayMacro.h

```
#include "GlobalDefines.h"

void delay_us(uint timeInUs);
```

D.5. DelayMacro.c

```
#include "DelayMacro.h"
#include "Timer1us.h"

void delay_us(uint timeInUs){
    static FastTrigger wait;
    FastTriggerSet( &wait,timeInUs );
    while(!FastTriggerActivado(&wait));
}
```

D.6. E2PROM.h

```
#define EEPROM_ID    0xA0
```

```
#define EEPROM_READ  0x01
#define EEPROM_WRITE  0x00

bool eepromRead(uchar _block, uchar _address, uchar _size,void* data);
bool eepromWrite(uchar _block, uchar _address, uchar _size,void* data);
```

D.7. E2PROM.c

```
#include "GlobalDefines.h"
#include "Initialization.h"
#include "E2PROM.h"
#include "Trigger.h"
#include "I2C.h"

FastTrigger idleTimeout;

bool eepromRead(uchar _memoryAddress, uchar _address, uchar _sizeData, void*
data)
{
   FastTriggerSet(&idleTimeout,5); // Load Timer with 5 miliseconds
   uint reintentos = 0;
   bool result = false;

   do
   {
      while(!FastTriggerActivado(&idleTimeout)); // Wait timer until it expires
      if(( result = generateStartEvent())== false &&
         ( result |= sendAddresToSlave(_address,EEPROM_WRITE) ) == false &&
         ( result |= sendDataToSlave((uchar*)&_memoryAddress, 1)) == false)
      {
         result |= generateRepeatedStartEvent();
         if( ( result |= sendAddresToSlave(_address,EEPROM_READ)) == false &&
            ( result |=  receivingDataFromSlave(data,_sizeData )) == false )
            return false;
      }
      result |= generateStopEvent();
      FastTriggerSet(&idleTimeout,5); // Load Timer with 5 miliseconds
```

```
        reintentos++;
    }while((result != false) && (reintentos <= 3));

    return result;
}

bool eepromWrite(uchar _memoryAddress, uchar _address, uchar _sizeData, void*
data)
{
    FastTriggerSet(&idleTimeout,1); // Load Timer with 1 miliseconds
    uint reintentos = 0;
    bool result = false;

    do{
        while(!FastTriggerActivado(&idleTimeout)); // Wait timer until it expires
         if( ( result = generateStartEvent() ) == false &&
         ( result |= sendAddresToSlave(_address,EEPROM_WRITE) ) == false &&
         ( result |= sendDataToSlave((uchar*)&_memoryAddress, 1)) == false )
            result |= sendDataToSlave(data, _sizeData);
        result |= generateStopEvent();
        FastTriggerSet(&idleTimeout,5); // Load Timer with 5 miliseconds
        reintentos++;
    }while((result != false) && (reintentos <= 3));

    return result;
}
```

D.8. FrequencyCounter.h

```
#include "GlobalDefines.h"
#include "Trigger.h"

#define FREQ PORTFbits.RF0
#define TIMER_SET 10000000

void readFreq(ulong *value);
uint measureFreq();
```

D.9. FrequencyCounter.c

```
#include "FrequencyCounter.h"

volatile bool last = false;

void readFreq(ulong *value){

    if(FREQ == 1 && last == false){
        value++;
        last = true;
    }
    else if(FREQ == 0 && last == true){
        value++;
        last = false;
    }
}

uint measureFreq(){
    static FastTrigger gate;
    ulong calcFreq = 0;
    FastTriggerSet(&gate,TIMER_SET); //Gate de 10 seg
    while(!FastTriggerActivado(&gate)){
        readFreq(&calcFreq);
    }
    return (readFreq/20);
}}
```

D.10. I2C.h

```
#define BRG 3

bool receivingDataFromSlave(uchar *data, uchar size );
bool generateAcknowledge(bool ack);
bool sendDataToSlave(uchar *data, uchar size );
bool sendAddresToSlave(uchar _address, uchar _readWrite);
bool generateRepeatedStartEvent();
```

```
bool generateStartEvent();
bool generateStopEvent();
```

D.11. I2C.c

```
#include "I2C.h"
#include "DelayMacro.h"

static FastTrigger timeout;
static FastTrigger delay;

bool testRegister(uint *_reg, bool _check )
{
//uint reg = _reg^_check;

uint res;
uint chek = (uint)_check;

FastTriggerSet(&timeout,1000 );
do{
res = (uint)_reg^chek;
if( FastTriggerActivado(&timeout) )
return true;
}while(res);
return false;
}

bool generateStartEvent()
{
//bool isIdle = I2C1STATbits.P; //Check if the bus state is in Idle
//if(!isIdle)
//{
// Start Sequence
I2C1CONbits.SEN = 1; // Initiate Start event
delay_us(2*BRG); //30us
uint S = I2C1STATbits.S;
```

```
if(testRegister((uint*)S, true)) // Slave detects the Start process and sets S = 1
return true;
uint SEN = I2C1CONbits.SEN;
if(testRegister((uint*)SEN, false)) // SEN bit is automatically cleared by hardware at completion of the Start Condition
return true;
return false;
//}
//else
//  return true;
}

bool generateStopEvent()
{
// Stop Sequence
I2C1CONbits.PEN = 1; // Initiate Stop event
delay_us(3*BRG); //40us
uint P = I2C1STATbits.P;
if(testRegister((uint*)P, true)) // Slave detects the Stop process and sets P = 1
return true;
uint PEN = I2C1CONbits.PEN;
if(testRegister((uint*)PEN, false)) // SEN bit is automatically cleared by hardware at completion of the Stop Condition
return true;
return false;
}

bool generateRepeatedStartEvent()
{
I2C1CONbits.RSEN = 1; // Initiate Repeated Start event
delay_us(3*BRG); //40us
uint S = I2C1STATbits.S;
if(testRegister((uint*)S, true))
return true;// Slave detects the Start process and sets S = 1
uint RSEN = I2C1CONbits.RSEN;
```

```
if(testRegister((uint*)RSEN, false))
return true; // SEN bit is automatically cleared by hardware
return false;
}

bool sendAddresToSlave(uchar _address, uchar _readWrite)
{
uchar dataToSend = _address | _readWrite;
I2C1TRN = dataToSend; // Writing to I2C1TRN trigers starts the data sending to slave device

delay_us(9*BRG); //220us // Wait 8x Time BRG ( 1 byte ) + 1 Tbg for Acknowledge
/*****************************************************************************/
uint TRSTAT = I2C1STATbits.TRSTAT;
if(testRegister((uint*)TRSTAT,false)) // Finish wiriting 8 bits to Slave device and collect ACK/NACK
return true;
//WAIT(1);
bool nack = I2C1STATbits.ACKSTAT; // Reading status of ACK received from de slave
if(nack)
return true;
else
return false;
}

bool sendDataToSlave(uchar *data, uchar size )
{
FastTriggerSet( &delay,1 );
uint i = 0;
for(i = 0; i<size;i++)
{
while(!FastTriggerActivado(&delay));
I2C1TRN = data[i]; // Writing to I2C1TRN trigers starts the data sending to slave device
```

```
delay_us(9*BRG); //20us // Wait Time BRG
uint TRSTAT = I2C1STATbits.TRSTAT;
if(testRegister((uint*)TRSTAT,false)) // Finish wiriting 8 bits to Slave device and collect ACK/NACK
return true;
bool nack = I2C1STATbits.ACKSTAT; // Reading status of ACK received from de slave
if(nack)
return true;
FastTriggerSet( &delay,1 );
}
return false;
}

bool generateAcknowledge(bool ack)
{
I2C1CONbits.ACKDT = ack; // Writing Ack(0) or Nack(1)
I2C1CONbits.ACKEN = 1; // Initiate Acknowledge generation
delay_us(2*BRG);
uint ACKEN = I2C1CONbits.ACKEN;
if(testRegister((uint*)ACKEN,false)) // Finish wiriting 8 bits to Slave device and collect ACK/NACK
return true;
return false;
}

bool receivingDataFromSlave(uchar *data, uchar size )
{
FastTriggerSet( &delay,1 );
uint i;
for(i=0; i<size;i++)
{
while(!FastTriggerActivado(&delay));
I2C1CONbits.RCEN = 1; // Start begin receiving sata from slave
delay_us(8*BRG); //100us // Wait 8 Time BRG
```

```
uint RBF = I2C1STATbits.RBF;
if(testRegister((uint*)RBF,true)) // RBF indicates that I2C1RCV register is full
return true;
uint RCEN = I2C1CONbits.RCEN;
if(testRegister((uint*)RCEN,false)) // Hardware cklears the RCEN bit
return true;
data[i] = I2C1RCV; // Reading byte received from the slave device
delay_us(1*BRG); //10us
if(i == size -1){
generateAcknowledge(true);
// Stop
}
else
generateAcknowledge(false);

FastTriggerSet( &delay,1 );
}
return false;
}
```

D.12. MuxControl.h

```
#include <xc.h> // include processor files - each processor file is guarded.
#include "GlobalDefines.h"

#define A3 LATBbits.LATB10
#define A2 LATBbits.LATB11
#define A1 LATBbits.LATB12
#define A0 LATBbits.LATB13

void SetQCM(uchar _mux );
void NextQCM();
void assingMuxChannels(uchar _mux);
```

D.13. MuxControl.c

```
#include "xc.h"
#include "MuxControl.h"
```

```
    volatile uchar mux = 0;

    void SetQCM(uchar _mux ){
        assingMuxChannels(_mux);
    }

    void NextQCM(){
        if(mux<16){
            mux++;
            assingMuxChannels(mux);
        }
        else{
            mux = 0;
        }
    }

    void assingMuxChannels(uchar _mux){
        uchar calA3 = 0, calA2 = 0, calA1 = 0, calA0 = 0, tmp = 0;
        tmp = _mux;

        calA3 = tmp/8;
        tmp = tmp - 8*calA3;
        calA2 = tmp/4;
        tmp = tmp - 4*calA2;
        calA1 = tmp/2;
        calA0 = tmp - 2*calA1;

        A3=calA3;
        A2=calA2;
        A1=calA1;
        A0=calA0;

}
```

D.14. RsCalc.h

```
#include "GlobalDefines.h"
#include "DelayMacro.h"

float calcRs();
uint get14bits();
```

D.15. RSCalc.c

```
#include "RsCalc.h"

float calcRs(){
    float ret = 0;
    uint measure = get14bits();
    ret = (float)measure * 0.91;
    return ret;
}

uint get14bits(){
    uint ADCValue = 0;
    uint cnt = 0;
    uint result = 0;

    while(cnt<256){
        AD1CON1bits.SAMP = 1;          // Start sampling
        delay_us(10);                  // Wait for sampling time (10 us)
        AD1CON1bits.SAMP = 0;          // Start the conversion
        while (!AD1CON1bits.DONE);     // Wait for the conversion to complete
        ADCValue = ADC1BUF0;           // Read the ADC conversion result}

        result = result + ADCValue;
        cnt++;
    }

    return result;
}
```

D.16. temperature.h

```
#include "GlobalDefines.h"
#include "DelayMacro.h"

float readTemp();
```

D.17. temperature.c

```
#include "temperature.h"

float readTemp(){

   uint ADCValue = 0;
   float tmp = 0;
   AD1CON1bits.SAMP = 1;          // Start sampling
   delay_us(10);                  // Wait for sampling time (10 us)
   AD1CON1bits.SAMP = 0;          // Start the conversion
   while (!AD1CON1bits.DONE);     // Wait for the conversion to complete
   ADCValue = ADC1BUF1;           // Read the ADC conversion result}

   tmp = (float)ADCValue * (float)0.03222;

   return tmp;
}
```

D.18. Timer1us.h

```
#include "Trigger.h"
#include "GlobalDefines.h"

void __attribute__((__interrupt__,no_auto_psv)) _T1Interrupt(void);
```

D.19. Timer1us.c

```
#include "Timer1us.h"

extern volatile unsigned long long ullGlobalTime;

void __attribute__((__interrupt__,no_auto_psv)) _T1Interrupt(void)
{
```

```
    ullGlobalTime++;
    IFS0bits.T1IF = 0; //clear interrupt flag
}
```

D.20. Trigger.h

```
typedef struct
{
    unsigned long long ullActivo/*:39*/;
    unsigned long ulPeriodo/*:17*/;
    unsigned char ucPeriodos;
}Trigger;

unsigned long long GetGlobalTime();
void TriggerSet( Trigger* pTrigger, unsigned long ulTiempo );
void TriggerSetPeriodo( Trigger* pTrigger, unsigned long ulPeriodo, unsigned char
ucPeriodos );
unsigned long TriggerCountdown( Trigger* pTrigger );
void TriggerCambioPeriodo( Trigger* pTrigger, unsigned long ulPeriodo );
unsigned char TriggerActivado( Trigger* pTrigger );
void TriggerDisable(Trigger* pTrigger);


typedef unsigned long long FastTrigger;

void FastTriggerSet( FastTrigger* pFastTrigger, unsigned long ushTiempo );
unsigned char FastTriggerActivado( FastTrigger* pFastTrigger );
```

D.21. Trigger.c

```
#include "Trigger.h"

volatile unsigned long long ullGlobalTime = 0;

void TriggerSet( Trigger* pTrigger, unsigned long ulTiempo ){
    TriggerSetPeriodo( pTrigger, ulTiempo, 1 );
}

void TriggerCambioPeriodo( Trigger* pTrigger, unsigned long ulPeriodo ){
```

```
    if( pTrigger->ulPeriodo != ulPeriodo ){
        pTrigger->ullActivo -= pTrigger->ulPeriodo;
        pTrigger->ullActivo += ulPeriodo;
        pTrigger->ulPeriodo = ulPeriodo;
    }
}

void TriggerSetPeriodo( Trigger* pTrigger, unsigned long ulPeriodo, unsigned char
ucPeriodos ){
    if(ucPeriodos==0)
        TriggerDisable(pTrigger);
    else
        pTrigger->ullActivo = ullGlobalTime + ulPeriodo;
    pTrigger->ulPeriodo = ulPeriodo;
    pTrigger->ucPeriodos = ucPeriodos;
}

void TriggerDisable(Trigger* pTrigger){
    pTrigger->ullActivo = (unsigned long long)-1;
}

unsigned long TriggerCountdown( Trigger* pTrigger ){
    volatile unsigned long long tiempo = ullGlobalTime;
    if( tiempo >= pTrigger->ullActivo )
        return 0;
    return ( pTrigger->ullActivo - tiempo );
}

unsigned char TriggerActivado( Trigger* pTrigger ){
    unsigned char ret = ( ullGlobalTime >= pTrigger->ullActivo );
    if( ret )
    {
        if( !pTrigger->ucPeriodos )
            TriggerDisable(pTrigger);
else
```

```
{
        pTrigger->ullActivo = ullGlobalTime + pTrigger->ulPeriodo;
        if( pTrigger->ucPeriodos != 255 )
                                                    -- pTrigger->ucPeriodos;
                                                                    }
   }
   return ret;
}

void FastTriggerSet( FastTrigger* pFastTrigger, unsigned long ushTiempo ){
   *pFastTrigger = ullGlobalTime + ushTiempo;
}

unsigned char FastTriggerActivado( FastTrigger* pFastTrigger ){
   if( *pFastTrigger <= ullGlobalTime )
      return 1;
   else
                                                          return 0;
}

unsigned long long GetGlobalTime(){
   unsigned long long ret;
   ret = ullGlobalTime;
   return ret;
}
```

D.22. UART.h

```
#include "GlobalDefines.h"
#include <plib.h>

void ConfigUART();
void SendCharacter(uchar character);
void SendString(uchar *string);
void SendData(float data);
```

D.23. UART.c

```
#include "UART.h"

void ConfigUART(){
   UARTCONFIGURE(UART1, UART_ENABLE_PINS_TX_RX_ONLY);
   UARTSetFifoMode(UART1, UART_ENABLE_ON_RX_NOT_EMPTY);
   UARTSetLineControl(UART1, UART_DATA_SIZE_8_BITS |
UART_PARITY_EVEN | UART_PARITY_NONE | UART_STOP_BITS_1);
   UARTSetDataRate(UART1, GetPeripherialClock(), 9600);
   UARTEnable(UART1,UART_ENABLE_FLAGS(UART_PERIPHERAL |
UART_RX | UART_TX));
}

void SendCharacter(uchar character){
        while (!UARTTransmitterIsReady(UART1));
        UARTSendDataByte(UART1, character);
        while (!UARTTransmissionHasCompleted(UART1));
}

void SendString(uchar *string){
        unsigned long cont = 0;

        while (string[cont] != '\0' && cont < 1000)
        {
                SendCharacter(string[cont]);
                cont++;
        }
}

void SendData(float data){
   long aux = 0;
   uchar numbers[10] = {'0','1','2','3','4','5','6','7','8','9'};
   uchar value[6] = {0};

   aux = data * 100;
```

```
    SendCharacter(numbers[aux/100000]);
    SendCharacter(numbers[aux/10000]);
    SendCharacter(numbers[aux/1000]);
    SendCharacter(numbers[aux/100]);
    SendCharacter(numbers[aux/10]);
    SendCharacter(numbers[aux%10]);
    SendCharacter('\r');
    SendCharacter('\n');
}
```

D.23 GlobalDefines.h

```
#include <xc.h> // inclinude processor files - each processor file is guarded.
#include <stdbool.h>

#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long

#define FALSE 0
#define TRUE 1
```

Glossary

VCA . Voltage Controlled amplifier

uC: Microcontroller

BOM: Bill of materials