This Bachelor's Thesis focuses on the study of a real product to obtain an improved product through reverse engineering methods and with the help of the GRANASAT team and its documentation.

The project on which the work is based, has two main motivations: on the one hand the design, study and implementation of methods for the creation of a real product; and on the other hand, serving as a learning tool on how to organise a complex project, learning how to cope and solve any problem that might arise during its development and approaching the professional world that every engineer will have to face at the end of his stage as a student.

Although there is still work to be done in relation to the areas of software implementation, the hardware objective was fully achieved and the following steps to take regarding the development of the project are clearly defined.

Gregorio Lamarca García is a Bachelor in Telecommunication Engineering student, specialising in electronics, who culminates his degree with this project. He is from Huesa, Spain and joined the GRANASAT team to elaborate this Bachelor's Thesis.

Andrés María Roldán Aranda is the academic head of the present project, and the student's tutor. He is a professor in the Department of Electronics and Computers Technologies.

BACHELOR'S THESIS

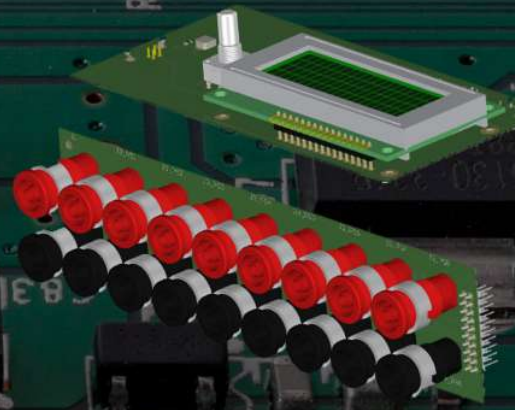
Design of PWM automotive lights tester

Gregorio Lamarca García

TELECOMMUNICATION ENGINEERING

UNIVERSITY OF GRANADA

Bachelor in Telecommunication Engineering



Design of PWM automotive lights tester

Gregorio Lamarca García

2021/2022

Tutor: Andrés María Roldán Aranda

Credits for the cover: **GRANASAT**.
Printed in Granada, July 2022.

“Design of PWM automotive ligths tester”

BACHELOR IN
TELECOMMUNICATION TECHNOLOGY ENGINEERING

Bachelor's Thesis

"Design of PWM automotive lights tester"

ACADEMIC COURSE: 2022

Gregorio Lamarca García

BACHELOR IN TELECOMMUNICATION TECHNOLOGY ENGINEERING

"Design of PWM automotive lights tester"

AUTHOR:

Gregorio Lamarca García

SUPERVISED BY:

Prof. Andrés Roldán Aranda

DEPARTMENT:

Electronics and Computers Technologies

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Grado de D. Gregorio Lamarca García,

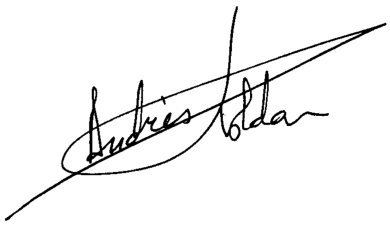
Informa:

Que el presente trabajo, titulado:

"Design of PWM automotive lights tester"

ha sido realizado y redactado por el mencionado alumno bajo mi dirección, y con esta fecha autorizo a su presentación.

Granada, a 17 de junio de 2022

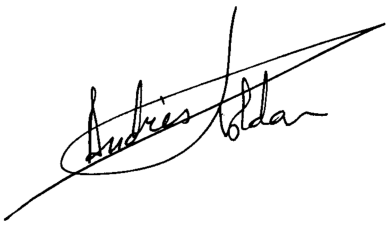


Fdo. Prof. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Grado se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 17 de junio de 2022

Fdo. Gregorio Lamarca García



Fdo. Prof. Andrés María Roldán Aranda

“Design of PWM automotive lights tester”

Gregorio Lamarca García

KEYWORDS:

CH559, Altium Designer® 21, Electronic, luxometer ADC, EPS, PCB Design of PCB, MATLAB.

ABSTRACT:

The main purpose of this project is developing a PWM lights tester cheaper than a previous version with **Arduino** and improvements made. It will be composed of three differentiated blocks, around which the project is structured: previous version, new version and a Pulse-Width Modulation (PWM) lights tester prototype that will be the base for the future **VALEO Pulse-Width Modulation (PWM) lights tester V1.2**.

This thesis is an improvement of Pulse-Width Modulation (PWM) lights tester: on the one hand, replacing **Arduino** micro controller with **CH559**, making it possible to lower the price and on the other hand, enable the device to read consumption so as not to depend on a luxmeter.

During the thesis, information about the methodology for improving the product has been acquired and organised, which will make future improvements easy to understand in order to progress with the product, thus saving time in the preparation of the future project.

“Design of PWM automotive lights tester”

Gregorio Lamarca García

PALABRAS CLAVE:

CH559, Altium Designer® 21, micro controlador, Electrónica, luxómetro, ADC, EPS, Diseño de PCB.

RESUMEN:

El objetivo principal del presente proyecto es desarrollar un testeador de luces Pulse-Width Modulation (PWM) más barato que la versión anterior con **Arduino** y con mejoras introducidas. Estará compuesta de tres bloques diferenciados, en torno a los cuales pivotará el proyecto: una versión anterior, la nueva versión y un prototipo del testeador de luces PWM , que constituirá la base del futuro **VALEO PWM ligths tester V1.2**..

Este trabajo es una mejora del testeador de luces: por un lado, reemplazando el Arduino por CH559 haciendo posible su abaratamiento y por otro lado, conseguir que el dispositivo lea el consumo para no depender de un luxómetro.

Durante el la tesis se ha adquirido y organizado información acerca de la metodología para mejorar el producto, que hará que en futuras mejoras sea fácil de entender para poder progresar con este, ahorrando así tiempo en la preparación del futuro proyecto.

'Perseverance and effort'

Agradecimientos:

Quiero dedicar este trabajo a un grupo de personas especiales que me han dado todo el apoyo que me hizo falta para poder seguir adelante en los momentos más duros de este:

A mi familia, mi madre, Carmen García, y mi hermana, Carmen Lamarca, por el apoyo moral y económico para poder centrar mi esfuerzo en el grado y en este trabajo.

A mis amigos del pueblo,a todos ellos, porque si ahora soy quien soy, es por ellos y por los momentos vividos.

A los amigos que he hecho en el grado, en especial a José Antonio, Osama, Sandra, Javier Merino, Javier Alonso, Víctor y José A. Simón, por el apoyo moral y académico que me dieron, haciendo sentirme acompañado durante este proceso a veces duro y otras veces tan dulce.

Y por último a Sara Díaz Carmona, gran amiga y compañera en etapas importantes de mi vida y apoyo fundamental en el día a día.

También agradecer a todos los compañeros del laboratorio, por tener la paciencia para poder lidiar con mis cuestiones y echarme una mano siempre que podían, y en especial al profesor Don Andrés, por sacarme de la cueva de Platón y cegarme con el mundo real.

Todo el proceso que envuelve este Trabajo Fin de Grado es la culminación de toda una vida en constante aprendizaje, donde he aprendido que este no es el final, sino un nuevo comienzo, y los mencionados anteriormente y otros no mencionados tienen parte de culpa de este logro.

Contents

List of Figures

List of Tables

List of Videos

List of 3D Models

Glossary

0805 Tamaño de dispositivo 0805 mils (2013 métrico): 0.08" × 0.05" (2.0 mm × 1.25 mm). Tamaño para resistencias de potencia típica de 1/10 o 1/8 W.

Altium Designer® 21 software used to design PCB from schematics. It allows 3D Design, as well as electronics simulation.

CH559 Micro controller of the CH55X family, manufactured by WCH..

EDA Plataforma de herramientas software para el diseño electrónico.

Acronyms

ADC Analog to Digital Converter.

EPS Electrical Power System.

ESD Electrostatic Discharge.

EVT EValuation Test.

GND Ground.

GPIO General Purpose Input/Output.

LCD Liquid-Crystal Display.

LED Light-Emitting Diode.

LQPF Low-profile Quad Flat Package.

MATLAB MATrix LABoratory..

PCB Printed Circuit Board.

PWM Pulse-Width Modulation.

RAM Random Access Memory.

ROM Read-Only Memory.

RW Read/Write.

SDCC Small Device C Compiler.

UGR University of Granada.

USB Universal Serial Bus.

Chapter 1

Introduction

The following bachelor's thesis was wrote as a fulfillment of a 4 years Telecommunications Engineering Bachelor's degree, with the help of *GRANASAT*. The goal of this work was to design and implement an update testing system for Pulse-Width Modulation (PWM) Box, which has a previous version that has based a new improvement version.

GRANASAT is an engineering and aerospace development group from the University of Granada (UGR), which is made up entirely for students and under the supervision of the professor Dr. Andrés María Roldán Aranda.

Figure 1.1 – *Granasat logo*

The Granasat project, which logo is shown in the Figure 1.1, began as a student initiative. In 2013, several students who were interested in product and aerospace engineering, study of electronic and wanted to focus their technical studies into the engineering scope.

The development took a half year of working on the design, implementation and test tasks.

Figure 1.2 – *Previous version of Pulse-Width Modulation (PWM) BOX*

1

The author of this document was part of the GranaSAT team, with the role of mainly responsible, among other smaller tasks, for the project management, budget management, test campaign, electronics design, Printed Circuit Board (PCB) design and implementation of a product for VALEO SA.

This thesis has been possible thanks to VALEO company.

Figure 1.3 – *Valeo logo*

1.1 Problem statements

When testing if a headlight is working correctly, it is intended to test the correct luminous intensity with certain conditions such as duration, patterns and other. To test luminous intensity is necessary a luxometer, which is a tool used to measure the light intensity and tilt of pilots.

In the industry is so important to reduce expense in order to be able got higher profits and luminosity sensors or labour cost could be an essential part of expenses.

For this reason it is thought that labour and used tools cost can be critical when creation of a new product is studied.

Figure 1.4 – *Testing of pilot light*

1.2 Motivation and objectives

Since the creation of the car, headlamps and different lights have evolved from simple bulbs to complex systems with Light-Emitting Diode (LED) technology.

This part of the car is as necessary as a seat belt or airbag as it forms part of the active safety of the vehicle and is essential in times of poor visibility.

This work aims to improve testing both economically and in terms of time compared to tests that are done on a driver-by-driver basis with a *luxometer*.

With the completion of this project, two objectives are therefore fulfilled and these are: the improvement of a useful tool for the automotive industry, in this case the manufacturing of luminous devices; and the personal goal of putting into practice what I learned in the bachelor's degree through the process of following a project already started and putting my knowledge to improve it and learning what are the functions of a product engineer and the role he/she would play in a company when it comes to fulfilling those functions.

The following objectives have nothing to do with technical specifications, but with the future personal achievement of competences after completion of the project:

Ref.	Personal goals
Obj.1	Learning how to use tools for hardware creation such as *Altium Designer® 21.*
Obj.2	Learning to study micro controllers in order to take full advantage of their features.
Obj.3	Learn how to organise a large project in which unforeseen events may arise.
Obj.4	Learn the processes involved in the design and manufacture of a product.
Obj.5	Learning to manufacture a product.
Obj.6	Practical application of the knowledge acquired in the bachelor's degree.

Table 1.1 – *Non-technical requirements of the project - Personal goals*

The project will be divided into the following tasks which form part of the following chapters:

(1) To study the use of *Altium Designer® 21.*

(2) To study the use and programming of *CH55X family. CH559L* specifically.

(3) To understand the product idea both hardware and software level.

(4) To design the Printed Circuit Board (PCB) that is able to fulfill the requirements set.

5 To design the CODE functions that is able to fulfill the requirements set.

(6) To provide enough firmware and documentation to make the future easier.

(7) To set a test planning for the future work in order to achieve the product.

1.3 Project structure

Following the objectives listed above , the project was developed after planning the next structure:

- **Chapter 1: Introduction.**
  This chapter describes in general terms what the project is about, the motivation and personal objectives, and ends with a brief description of the structure of the work.
- **Chapter 2: Previous study and system requeriments.**
  This chapter describes previous necessary knowledge to understand the work in following chapters.

A brief Printed Circuit Board (PCB) and *CH559L* programming background will be presented and previous product version. A study of the previous product will also be carried out to check what requirements it needs.

1

- **Chapter 3: System analysis.**
  The analysis of project are exposed after to understand the previous background and product, which is necessary to be able to manage time and project.
- **Chapter 4: System description and design.**
  In this chapter the design of Printed Circuit Board (PCB) and *CH559L* programming are described in detail. From a general description to a specific view are described, both the electronics design and firmware design.
- **Chapter 5: Budget description of product.**
  The budget for the creation of the product will be described in detail.
- **Chapter 6: Conclusions, future work and lessons learned.**
  In last chapter, the conclusions arisen after work, future highlight improvements and learned lessons after bachelor´s thesis are described.
- **Chapter 6.3: Appendix.**
  Appendix A: CH554 programming example

  Appendix B:

Chapter 2

Previous study and system requerimetns

Improvement of a full product takes more time than a time for bachelor´s thesis so hardware design objective has been finished but software objective only has some examples but there is not a complete implementation for the product.

The hardware design objective is referred to Printed Circuit Board (PCB) design where physics components are analyzed through data sheets where electrical and mechanical conditions are available. For it, *Altium Designer® 21* has been utilized with a *Multi Assembly Project*.

The software design objective is referred to CODE functions through a micro controller where hardware design reacts in order to fulfill all functions have been requested. The selected controller has been *CH559L*, which has a 8051 architecture and features with multiple outputs and inputs that are necessary for achieving all product functions.

2.1 Creating a Multi board assembly project in *Altium Designer® 21*

Altium Designer® 21 is a software for mechanical and electrical simulation, used for the creation of Printed Circuit Board (PCB) where, from a schematic, a 2D model is obtained based on layers, which is then modelled in 3D to visualise the final product.

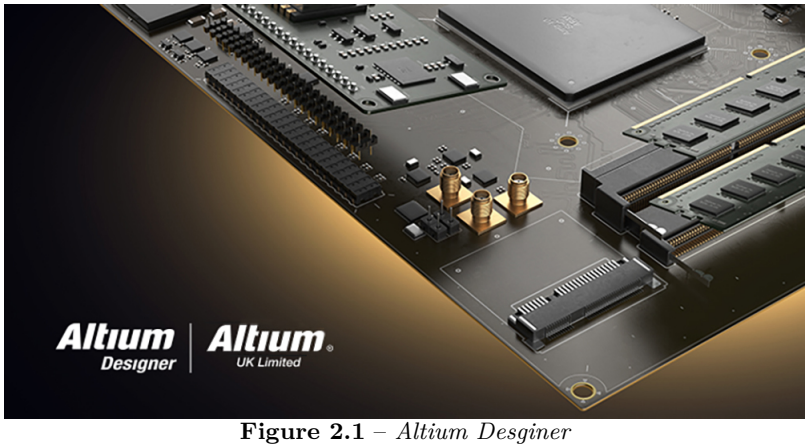


Figure 2.1 – *Altium Desginer*

2.1.1 A single project

For the creation of a single Printed Circuit Board (PCB) it is necessary to create a project with extension (.PrjPcb) in which schematic sheets are created and through a system of libraries, components are loaded to these schematics.

The components need two types of files: the file for the schematic library (.SchLib) and the file for the Printed Circuit Board (PCB) library (.PcbLib). The latter is necessary to move from the schematic model to the PCB as it acts as a footprint.

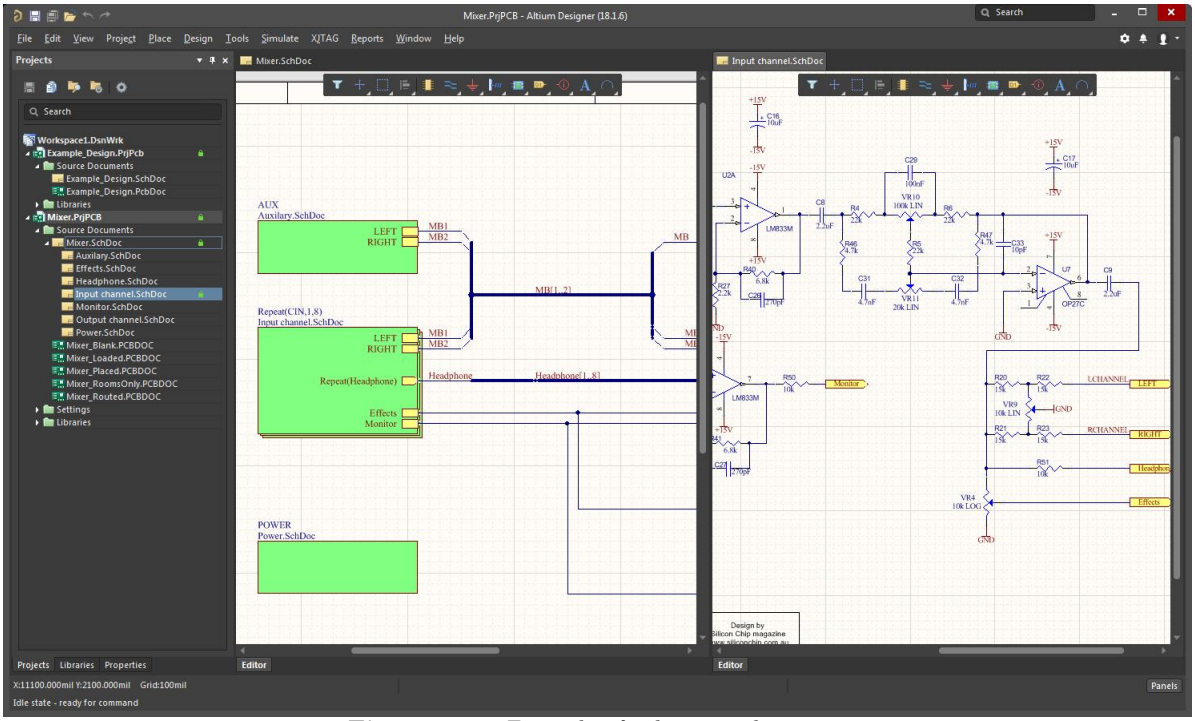


Figure 2.2 – *Example of schematic document*

Once the project is finished and the Printed Circuit Board (PCB) model has been obtained through the schematic sheet, the design rules are checked, which are extremely important in order to be able to manufacture (in the case of this work, it will be in *JLCPCB*) and check the connections of the schematic, the next step is to obtain a document where both the schematic and the PCB are shown.

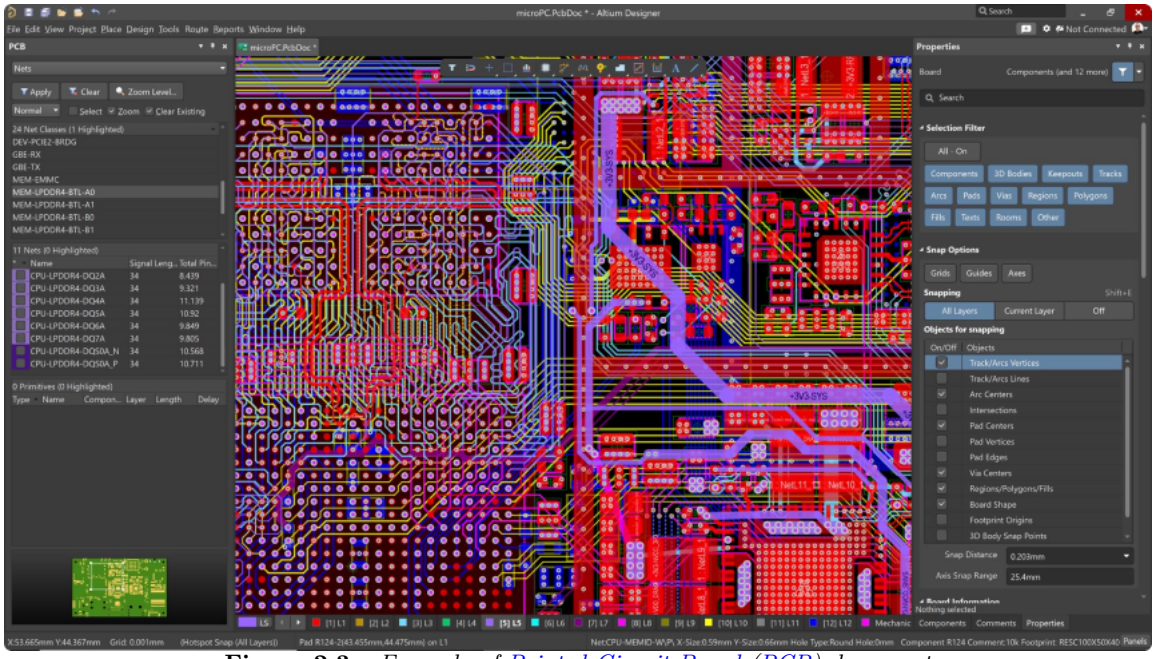


Figure 2.3 – *Example of Printed Circuit Board (PCB) document*

2.1.2 A multi board assembly project

A multi board assembly is a project that hosts several simple projects and has extension (.PrjMbd). By joining the input and output ports in a schematic (.MbsDoc), the electrical connection is created to check the connections between projects and then dump this schematic in a multi-assembly (.MbaDoc), where the 3D models of the Printed Circuit Board (PCB)s can be seen.

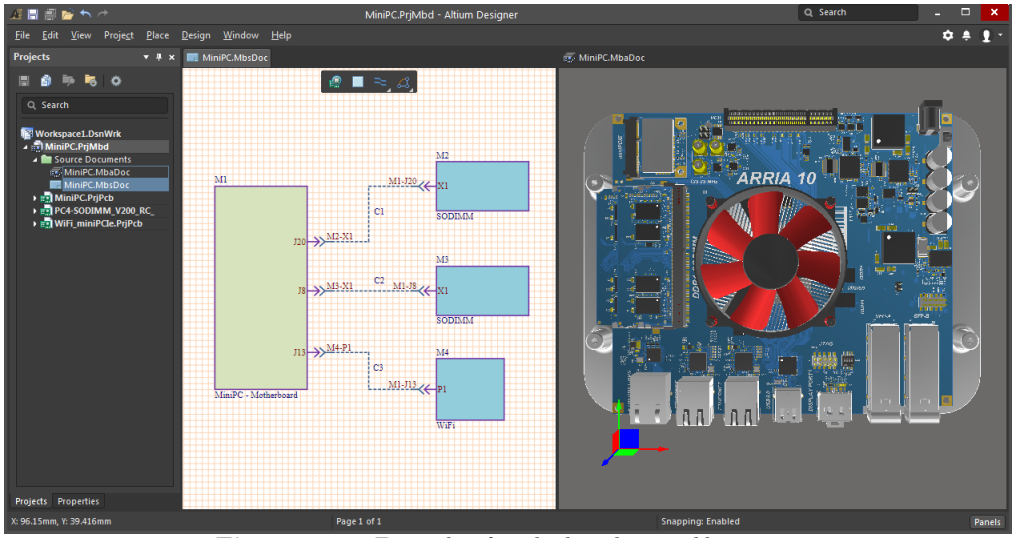


Figure 2.4 – *Example of multi-board assembly project*

2.2 *CH559* programming

2.2.1 Introducing the *CH55X family* of controllers

The *CH55X family* of micro controllers is a family of controllers manufactured by WCH and based on the 8051 architecture, comprising the *CH551*, *CH552*, *CH554*, *CH555*, *CH557*, *CH558* and *CH559*.

They are cheap controllers and in their EValuation Test (EVT) version are very useful as they have small and handy boards for testing with peripherals, before using the controller on a custom board.

For programming, a batch file can be created, allowing this family of micro controllers to be programmed without the need for external software installations, only a programming tool (chflasher.exe) and an external compiler (Small Device C Compiler (SDCC)) are required.

An example of the programming of the *CH554* will be shown in appendix.

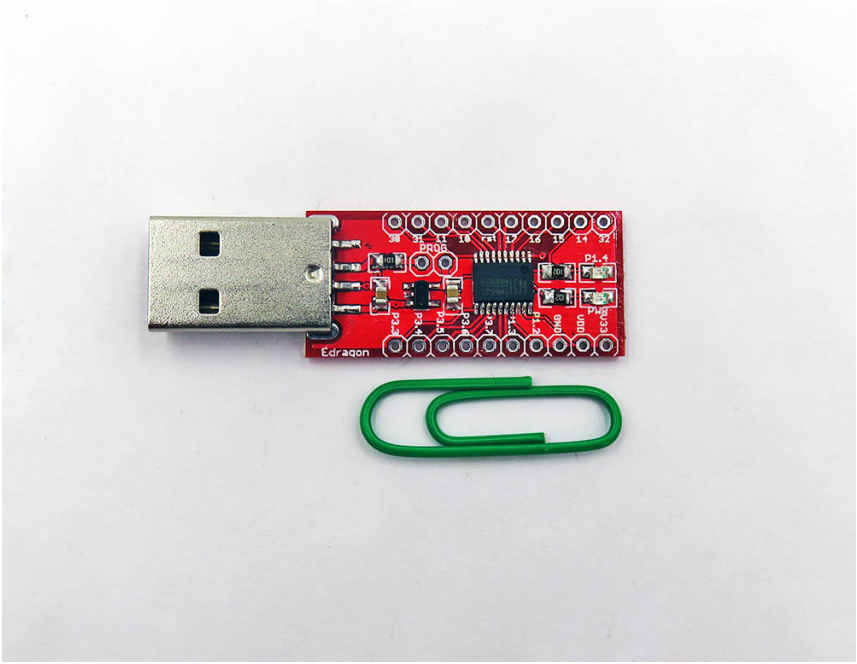


Figure 2.5 – *CH554 mini EValuation Test (EVT)*

2.2.2 Introducing the *CH559*

Figure 2.6 – *CH559 mini EValuation Test (EVT)*

For this work, the *CH559* micro controller with Low-profile Quad Flat Package (LQPF) packaging has been used, so it is called *CH559L*. This controller has 8 11-bit precision Analog to Digital Converter (ADC)s, with 5 ports and contains the following block diagram (taken from the above-mentioned website [5]):

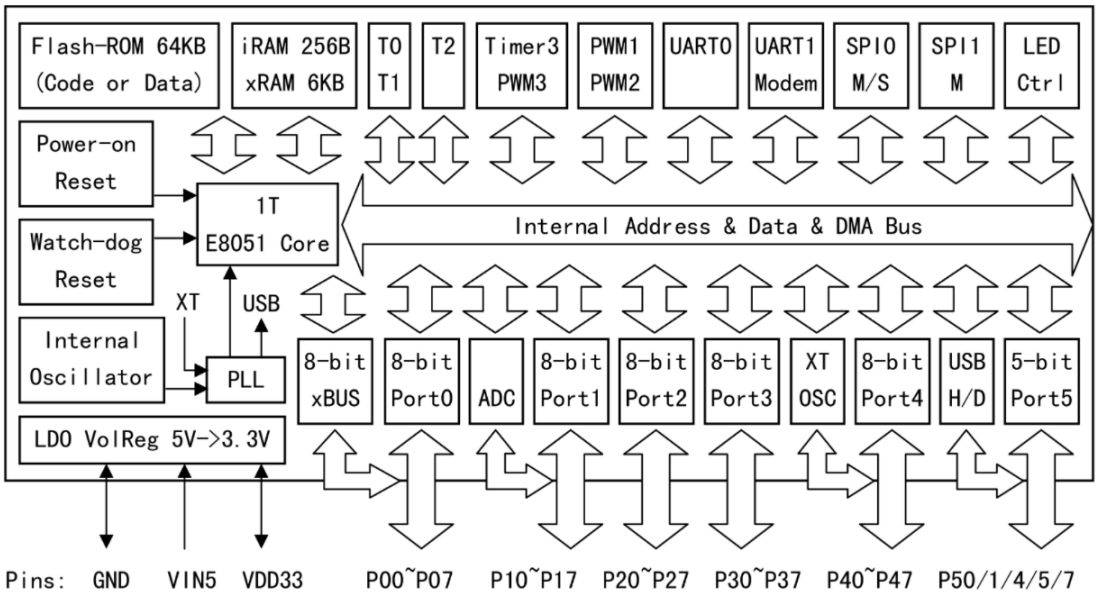
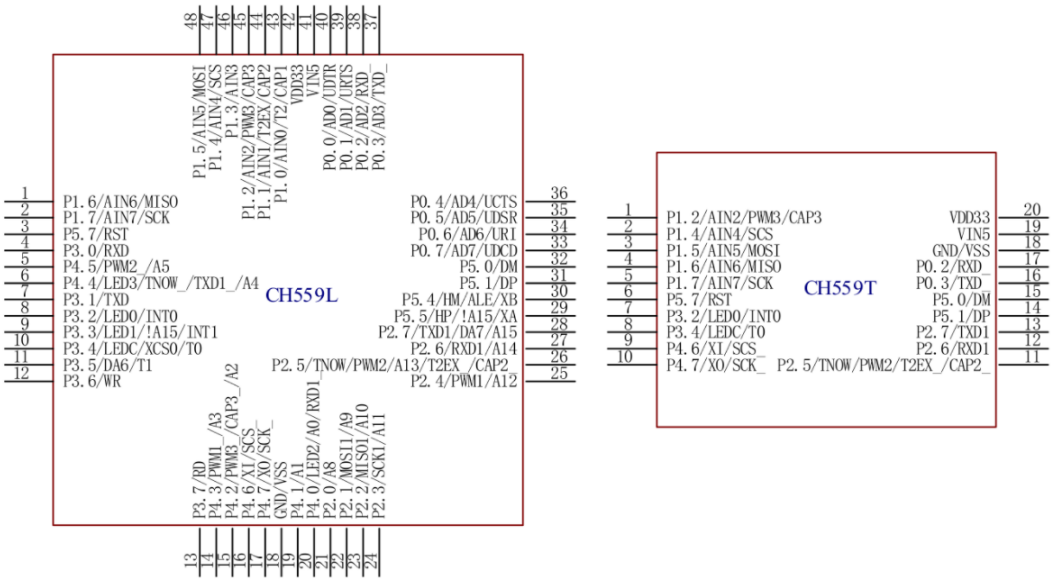


Figure 2.7 – *CH559L block diagram*



Package Description

Package	Package width	Pin spacing	Package description	Part No.
LQFP-48	7x7 mm	0.5mm / 19.7mil	Standard LQFP48 footprint	CH559L
SSOP-20	5.3mm	0.65mm / 25mil	Ultra-small 20-footprint	CH559T

Figure 2.8 – *CH559L package*

An example using an Analog to Digital Converter (ADC) channel and the Light-Emitting Diode (LED)s on the board will be shown in appendix.

2.3 Previous product version

Starting from the previous product of which we have its Hardware design but not the Software, we have to make changes in Hardware to adapt it to the requirements of the product on which this work is based and in the case of the Software, by testing the product at user level, a reverse engineering process is carried out to obtain its diagram that we will later try to implement. Knowing how the previous product works at hardware and software level, the requirements for the new product and its restrictions due to the technology to be used will be established.

The hardware part is discussed first, followed by the software part:

2.3.1 Previous hardware product

In the hardware design of a product, it is important to know that there are guidelines to follow, which are essential to avoid making mistakes that can cost time and/or money.

As the product has a box to which it has to be adapted, there is not much freedom when it comes to placing mechanical components, sizing boards or placing visual components in places where they cannot be seen.

2.3.1.1 Hardware study of the previous product

By means of the following 3D modelling images, which offer more concrete and less confusing information than showing the complete project, an analysis will be made of this Printed Circuit Board (PCB), which is the first of the two that the project consists of.

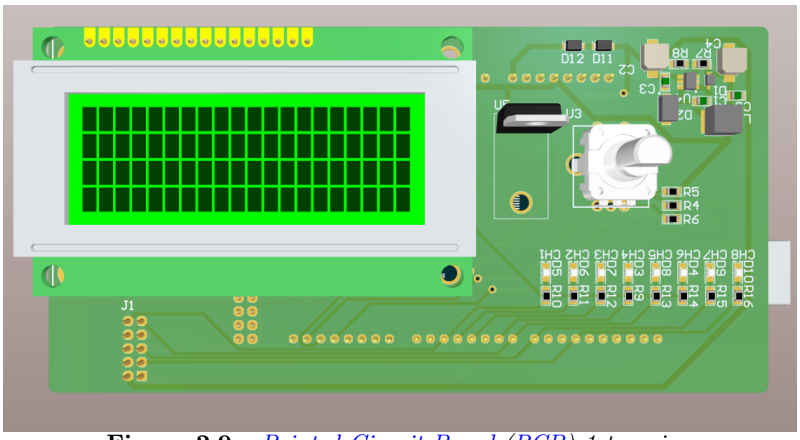


Figure 2.9 – *Printed Circuit Board (PCB) 1 top view*

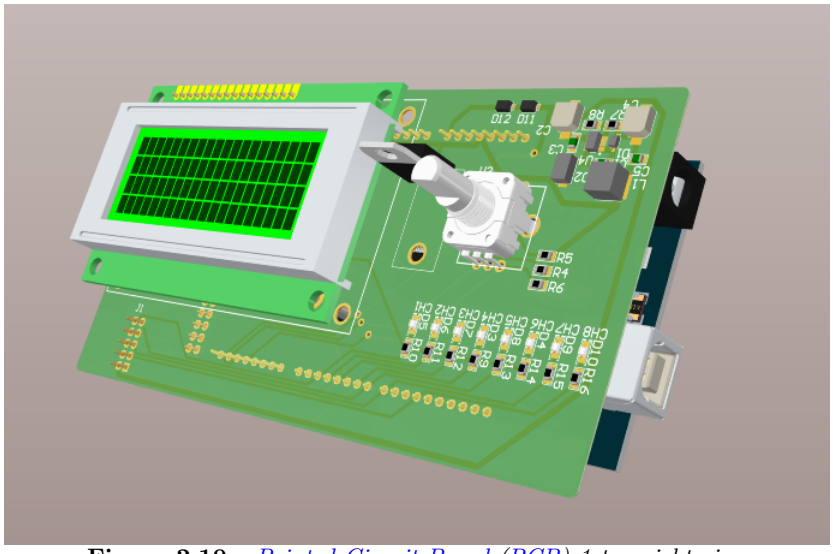


Figure 2.10 – *Printed Circuit Board (PCB) 1 top-right view*

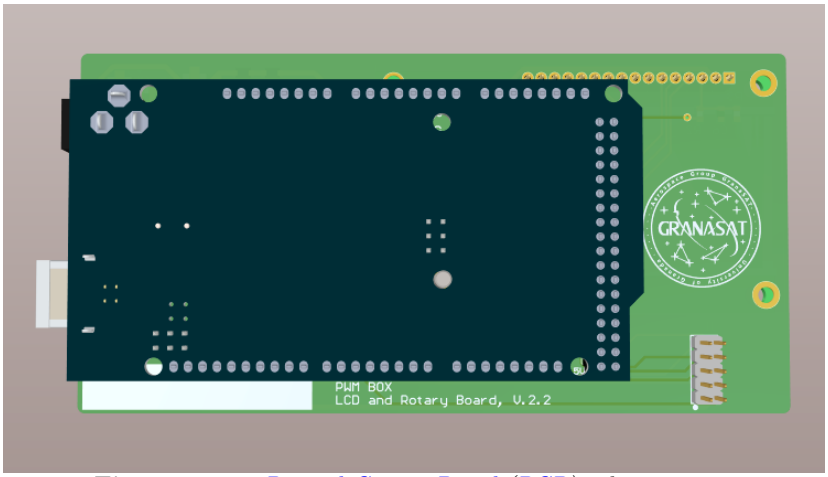


Figure 2.11 – *Printed Circuit Board (PCB) 1 bottom view*

It is now time to list the number of components on this Printed Circuit Board (PCB), which will be divided into TOP and BOTTOM layers to make it easier to find them:

TOP Components	**Quantity**
Capacitor 0805	4
Electrolytic capacitor	2
Diode 1N4148	1
Diode 0805	8
Diode BAS16GWX	2
Iron inductor SRN5040	1
Resistor 0805	16
Voltage regulator MCP16301	1
Voltage regulator LM7805	1
Rotary encoder PEC11R-4120F-S0018	1
LCD Winstar WH2004D-TMI-JT	1
TOTAL COMPONENTS	38

Table 2.1 – *Top components of Printed Circuit Board (PCB) 1*

The top layer contains most of the components that make up this Printed Circuit Board (PCB), leaving space for the Arduino MEGA which will be placed on the bottom layer.

BOTTOM Components	**Quantity**
Pin header 10 in 2	1
Arduino MEGA	1
TOTAL COMPONENTS	2

Table 2.2 – *Bottom components of Printed Circuit Board (PCB) 1*

On the bottom layer, apart from the Arduino MEGA, there is a header pin that will connect to Printed Circuit Board (PCB) 2. It is logical that the connector is on the bottom layer as the mechanical fastening of Printed Circuit Board (PCB) 1 is by the screen and makes it difficult or impossible to move the connector if it were on the top layer.

Total PCB1 Components	**Quantity**
Top components	38
Bottom components	2
TOTAL COMPONENTS	40

Table 2.3 – *Total components of Printed Circuit Board (PCB) 1*

Before explaining the function of this board, we will proceed to do the same procedure with Printed Circuit Board (PCB) 2, in order to have a full knowledge of the hardware and therefore be able to give an explanation from the general to the specific.

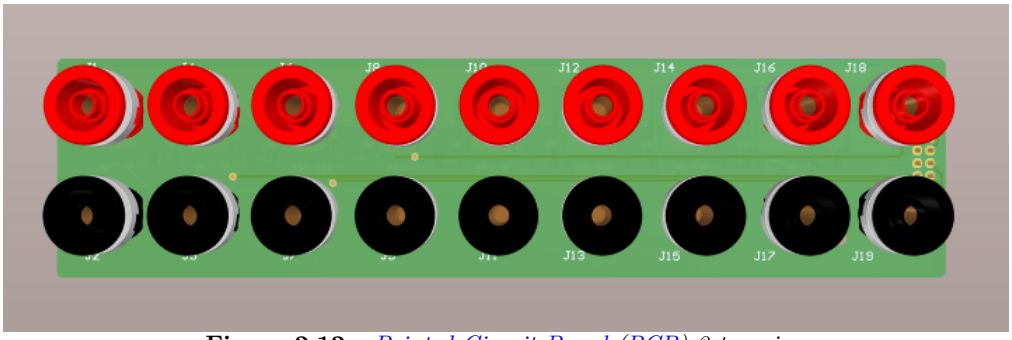


Figure 2.12 – *Printed Circuit Board (PCB) 2 top view*

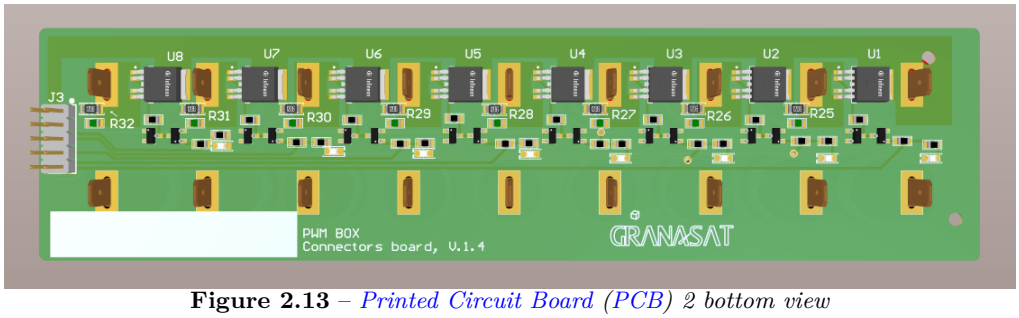


Figure 2.13 – *Printed Circuit Board (PCB) 2 bottom view*

Again, we proceed to list the components of Printed Circuit Board (PCB) 2 separating into top and bottom layers.

TOP Components	**Quantity**
4mm red banana jack connector	9
4mm black banana jack connector	9
TOTAL COMPONENTS	18

Table 2.4 – *Top components of Printed Circuit Board (PCB) 2*

On the top layer of Printed Circuit Board (PCB) 2, there are only connectors which, apart from having an electrical function, have a mechanical clamping function.

BOTTOM Components	Quantity
Capacitor 0805	8
Diode 0805	8
Diode BAS21.215	8
Transistor 2N7002P	8
Resistor 0805	24
Resistor 1206	8
Smart Highside Power Switch BTS6143D	8
Pin header 10 in 2	1
TOTAL COMPONENTS	73

Table 2.5 – *Bottom components of Printed Circuit Board (PCB) 2*

Total PCB 2 Components	Quantity
Top components	18
Bottom components	73
TOTAL COMPONENTS	91

Table 2.6 – *Total components of Printed Circuit Board (PCB) 2*

Having gathered information about the hardware of the previous product, we proceed to study more concretely the function of each board and how they are connected.

2.3.1.2 Function study of the previous product

This subsection will define the function of the components on both Printed Circuit Board (PCB)s, clearly detailing the use that each component has due to its configuration and the software, which has been studied as user level through reverse engineering.

On Printed Circuit Board (PCB) 1, starting from the header pin where both boards will be connected, there are 8 connections that connect to 8 pins of the Arduino MEGA and 8 resistors (()), which are used to hold 8 Light-Emitting Diode (LED)s, to visually check the correct operation.

Continuing with the Arduino MEGA micro controller, there are connected: a LCD screen, a rotary encoder, a LM7805 and a MCP16301. These last two are voltage regulators.

The Liquid-Crystal Display (LCD) has two resistors (()) as voltage divider, a capacitor (()) and another resistor (()) for the anode (pin 15).

The rotary encoder has 3 pull-up resistors (()), one for each output.

The voltage regulator MCP16301, consists of 3 capacitors (()), 2 capacitors (electrolytic), 2 resistors (()), a diode (B140-13-F), a diode (1N4148WS) and the iron inductor (SRN5040).

On Printed Circuit Board (PCB) 2, 8 channels and an input for 12 V power supply are identified. Each channel has the smart highside power switch BTS6143D and different components for configuration according

to the data sheet.

One channel components of PCB 2	Quantity
Smart Highside Power Switch BTS6143D	1
Resistor 0805	3
Transistor 2N7002P	1
Diode 0805	1
Diode BAS21.215	1
Capacitor 0805	1
Resistor 1206	1
4mm red banana jack connector	1
4mm black banana jack connector	1
TOTAL COMPONENTS	11

Table 2.7 – *One channel components of Printed Circuit Board (PCB) 2*

The 2N7002P has one resistor at the gate and one resistor between the drain and pin 2 of the BTS6143D as the default configuration in the data sheet. In turn, this configuration of the BTS6143D serves for the input circuit as Electrostatic Discharge (ESD) protection [7].

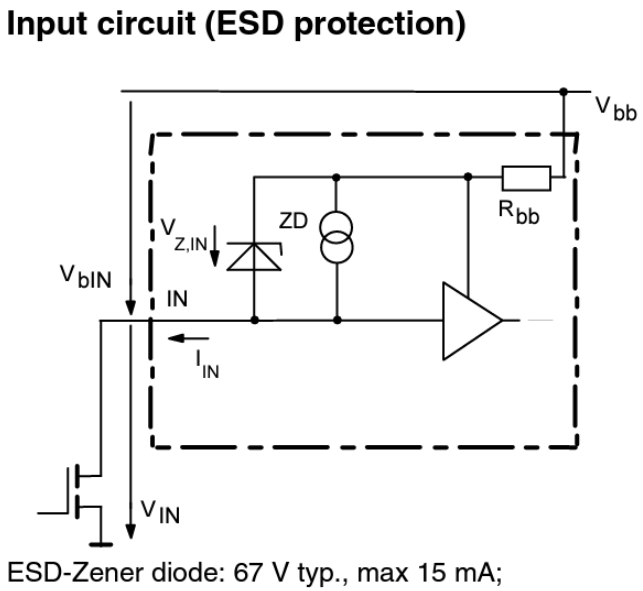


Figure 2.14 – *Input circuit (Electrostatic Discharge (ESD) protection) configuration for BTS6143D*

The transistor will be activated with a signal from the micro controller (Arduino MEGA) and in order to check its correct operation, an Light-Emitting Diode (LED) with its corresponding protection resistor has been installed.

For the input of the BTS6143D, 3 resistors (0805), a diode (0805) and a diode (BAS21.215) and finally the transistor have already been located.

At the output of the BTS6143D are the capacitor (0805) and the resistor (1206). The capacitor improves the transient voltage reaction (pin 5 and 1) and the resistor (pin 5 and 1) acts as a pull-down. 4mm red banana jack connector is connected to output (pin 5 and 1) too and 4mm black banana jack connector is Ground (GND).

Once the 11 components of each channel are described, which results in a total of 88, 3 components are missing to complete the description of Printed Circuit Board (PCB) 2.

Of these 3 components, there is a 4mm red banana jack connector for the 12 V input, which is connected to the BTS6143D of all channels (pin 6), a 4mm black banana jack connector for Ground (GND) and the pin header 10 in 2.

2.3.2 Previous software product

By making use of the previous product, it will be possible to propose the functions that the software encloses in the most reliable attempt to match the behaviour of the software.

A diagram describing the system algorithm is presented below:

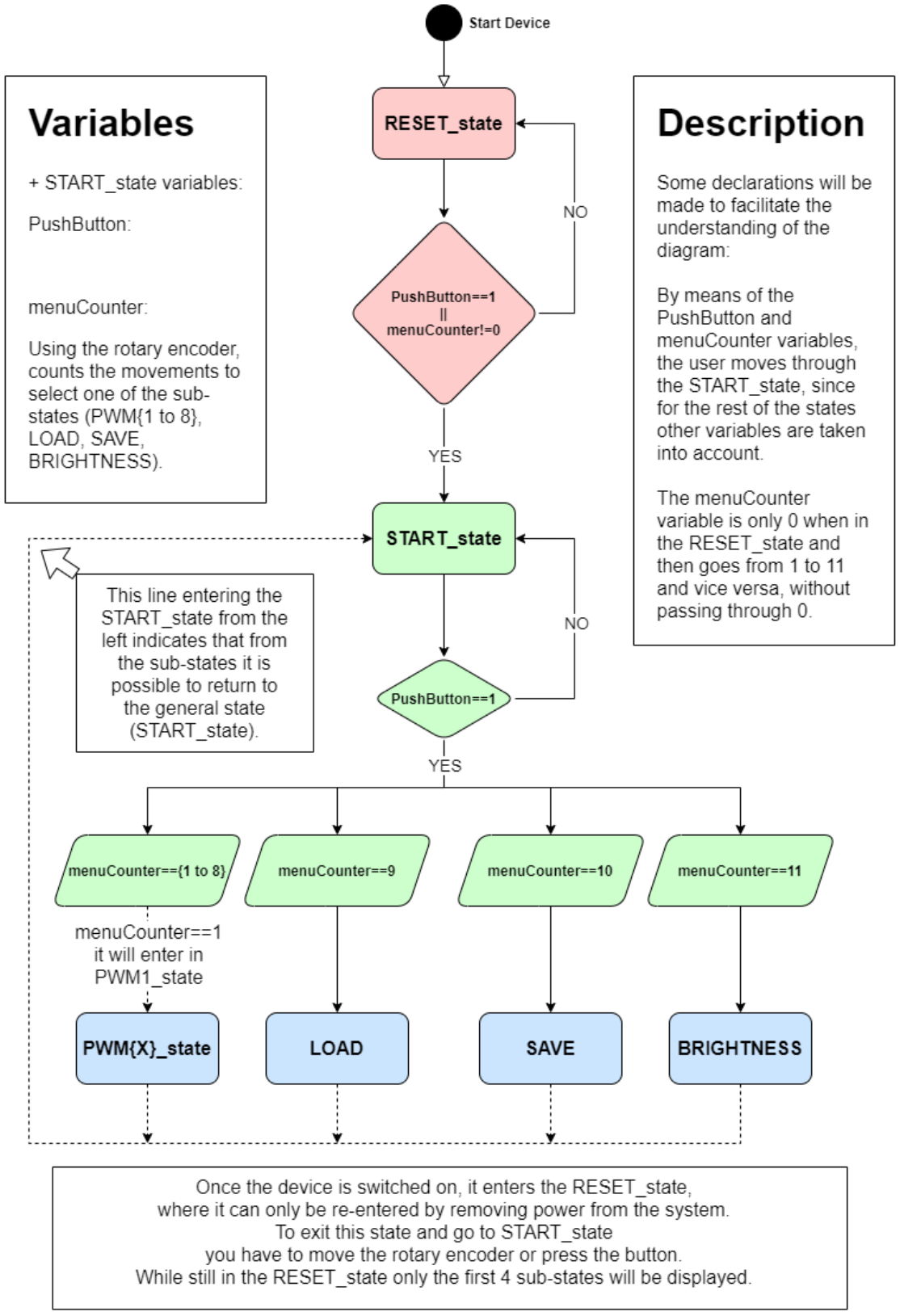


Figure 2.15 – *Flow diagram of previous product software*

2.3.2.1 Sub-states description

We will proceed to explain each sub-state, which variables it uses and what function they have.

2.3.2.2 LOAD, SAVE and BRIGHTNESS states

The LOAD state has a counter (loadCounter) from 0 to 1000, so it would use a 16-bit variable, where the configurations of all channels are loaded (each channel has its own state). To enter this state, the condition must be fulfilled in which the button is pressed when the menuCounter is 9 (the display screen indicates the state to which the counter is pointing). Once the configuration is loaded, it automatically returns to the START state.

In contrast, the SAVE state has the same structure as the LOAD state, but saves the configuration so that it can be loaded later. To enter this option the menuCounter will be 10 and to exit you only need to save.

The BRIGHTNESS state has a counter from 1 to 5, so with the counter it will be an 8-bit variable (brigCounter). To enter this state the menuCounter will be 11 and to exit just press again to choose a new brightness value.

2.3.2.3 PWMx states

To enter the PWMx states, which includes PWM1 state, PWM2 state, PWM3 state, PWM4 state, PWM5 state, PWM6 state, PWM7 state and PWM8 state. This x refers to the specific state (corresponding to a channel) to be entered, which coincides with the menuCounter (if, when the button is pressed, the menuCounter is 3, PWM3 state is accessed).

The following image shows the flow diagram of the PWMx states where the variables used and their functions will also be explained.

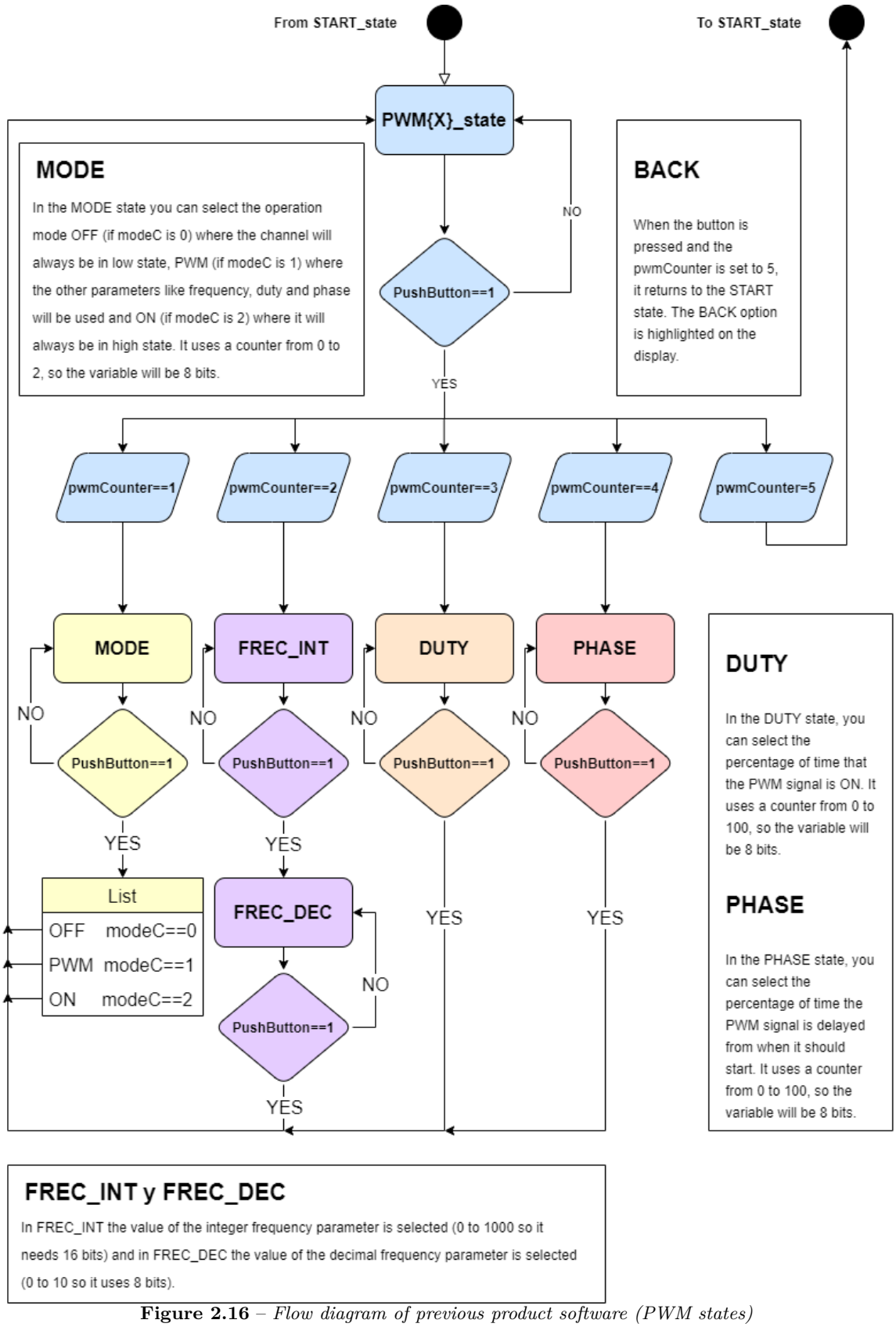


Figure 2.16 – *Flow diagram of previous product software (PWM states)*

Chapter 3

System requirements and analysis

In this chapter, the definition of formal requirements and analysis of the system are presented in this order.

In system requirements, the objectives that need to be known so that once the project has been completed, conclusions can be drawn about their achievement.

In the system analysis, a description of the materials available, why they were chosen and some examples to facilitate the understanding of the functions are given.

3.1 System requirements

Before moving on to the design of the system (4), it must be understood which functions must be fulfilled separately and the previous steps to be able to make a design capable of satisfying all of them.

A table with the definitions of the system requirements will be shown below, which will be shown later in the system analysis (3.2) where the achievement of these requirements will be corroborated.

Ref.	Formal Requirements
PR.FoR.1	Be able to control each function offered by the micro controller separately.
PR.FoR.2	To be able to put together all the software functions in the same code for the implementation of the product.
PR.FoR.3	Be able to create a Printed Circuit Board (PCB) that fulfils the needs for the control of all the software functions.
PR.FoR.4	Be able to create a Printed Circuit Board (PCB) that fulfils the software needs and has the maximum possible advantages for operability, repairability and testability.

Table 3.1 – *Product - Formal Requirements*

It should be noted that one of the premises of this project is to learn to work with the *CH55X family* of micro controllers, so controllers from this family will be compared.

3.2 System analysis

In the first analysis, the first requirement will be carried out, comparing micro controllers and discarding those that cannot perform each function separately.

In the second analysis, we will describe the basic needs of a Printed Circuit Board (PCB) to be able to fulfil all functions once only the micro controller is used without the EValuation Test (EVT) board.

In the third analysis, the needs of a Printed Circuit Board (PCB) to fulfil all necessary functions and requirements for operability, repairability and testability will be described, again with the micro controller without the EValuation Test (EVT) board.

In the fourth and final analysis, it will check that it has been able to compact all the functions into a single code.

3.2.1 Different functions separately

In order to be able to test the functions separately, it is first necessary to know which functions we had in the previous product and which are the new functions for improvement.

These functions will be ordered from least hardware required to most hardware required, as more hardware is required, their difficulty increases, so it is a good requirement to order them.

Ref.	Different functions separately
Function.1	Pin control to activate the switching transistor (BTS6143D).
Function.2	LCD screen control.
Function.3	External peripheral control for menu navigation with button to reprogram the micro controller and select options.
Function.4	Analog to Digital Converter (ADC) reading to be able to read the consumption by means of the configuration offered by the switching transistor (BTS6143D).
Function.5	Creation of a signal generator.

Table 3.2 – *Product - Different functions separately*

Once we have these five main functions separately, we proceed to an analysis according to the micro controller of the *CH55X family*.

During the project, the *CH552*, *CH554* and *CH559* micro controllers have been used, so these will be the ones compared.

Table 3.3 – *CH55X family comparaison*

Characteristics	CH552G	CH554T	CH559L
Enhanced 8051 core	YES	YES	YES
Built-in Flash	16KB program memory Read-Only Memory (ROM)	16KB program memory Read-Only Memory (ROM)	64KB non-volatile memory Flash-Read-Only Memory (ROM)
Built-in iRandom Access Memory (RAM)	256 bytes	256 bytes	256 bytes
Built-in xRandom Access Memory (RAM)	1KB	1KB	6KB
USB	Embedded USB controller and USB transceiver	Embedded USB controller and USB transceiver	Embedded USB controller and dual USB transceiver (2)
UART	2 full-duplex asynchronous serial ports	2 asynchronous serial ports	2 sets of asynchronous serial port
Timer	3 groups of timers/counters	3 groups of timers/counters	4 groups of timers
SPI	1 SPI communication interface	1 SPI communication interface	2 sets of SPI controllers
ADC	4-channel 8-bit	4-channel 8-bit	8-channel 10-bit or 11-bit
PWM	2 PWM output in timers´ pins	2 sets of PWM output	3 sets of PWM outputs
Total pins	16	20	48
Price (€) (lowest found)	2.02	1.03	1.21

3

3.2.1.1 Pin control to activate the switching transistor (BTS6143D)

In the function for pin control as General Purpose Input/Output (GPIO), you should have a minimum of 8 pins for General Purpose Input/Output (GPIO) as there are 8 channels that need to be controlled by the micro controller. The 3 micro controllers compared can fulfil this function.

Below is an example with the CH554 and CH559 micro controller to turn Light-Emitting Diode (LED)s on and off, which although not using 8, gives an idea of how the pins should be declared and by obtaining the knowledge to activate 1, 8 could be controlled in the same way. This example has used the Light-Emitting Diode (LED)s that the EValuation Test (EVT) board already has to obtain a visual result.

Example of LED control on CH554

```
1
2  typedef unsigned char *PUINT8;
3  typedef unsigned char __xdata *PUINT8X;
4  typedef const unsigned char __code *PUINT8C;
5  typedef unsigned char __xdata UINT8X;
6  typedef unsigned char  __data UINT8D;
7
8  #include "lib/CH554.h"
9  #include "lib/debug.h"
10 #include <stdio.h>
11 #include <math.h>
12
13 uint16_t wait = 100;
14 uint8_t i;
15
16 #define LED_PIN1 1
17 SBIT(LED1, 0x90, LED_PIN1);
18 #define LED_PIN2 4
19 SBIT(LED2, 0x80, LED_PIN2);
20 #define LED_PIN3 5
21 SBIT(LED3, 0x90, LED_PIN3);
22 #define LED_PIN4 6
23 SBIT(LED4, 0x90, LED_PIN4);
24 #define LED_PIN5 7
25 SBIT(LED5, 0x90, LED_PIN5);
26
27 #define LED_PIN6 0
28 SBIT(LED6, 0xB0, LED_PIN6);
29 #define LED_PIN7 1
30 SBIT(LED7, 0xB0, LED_PIN7);
31 #define LED_PIN8 2
32 SBIT(LED8, 0xB0, LED_PIN8);
33 #define LED_PIN9 3
34 SBIT(LED9, 0xB0, LED_PIN9);
35 #define LED_PIN10 4
36 SBIT(LED10, 0xB0, LED_PIN10);
37
38
39 void main() {
40     CfgFsys();
41
42     P1_MOD_OC = P1_MOD_OC |(1<<LED_PIN1);
43     P1_DIR_PU = P1_DIR_PU |    (1<<LED_PIN1);
```

```
44
45      P1_MOD_OC = P1_MOD_OC |(1<<LED_PIN2);
46      P1_DIR_PU = P1_DIR_PU |    (1<<LED_PIN2);
47
48      P1_MOD_OC = P1_MOD_OC |(1<<LED_PIN3);
49      P1_DIR_PU = P1_DIR_PU |    (1<<LED_PIN3);
50
51      P1_MOD_OC = P1_MOD_OC |(1<<LED_PIN4);
52      P1_DIR_PU = P1_DIR_PU |    (1<<LED_PIN4);
53
54      P1_MOD_OC = P1_MOD_OC |(1<<LED_PIN5);
55      P1_DIR_PU = P1_DIR_PU |    (1<<LED_PIN5);
56
57
58
59      P3_MOD_OC = P3_MOD_OC |(1<<LED_PIN6);
60      P3_DIR_PU = P3_DIR_PU |    (1<<LED_PIN6);
61
62      P3_MOD_OC = P3_MOD_OC |(1<<LED_PIN7);
63      P3_DIR_PU = P3_DIR_PU |    (1<<LED_PIN7);
64
65      P3_MOD_OC = P3_MOD_OC |(1<<LED_PIN8);
66      P3_DIR_PU = P3_DIR_PU |    (1<<LED_PIN8);
67
68      P3_MOD_OC = P3_MOD_OC |(1<<LED_PIN9);
69      P3_DIR_PU = P3_DIR_PU |    (1<<LED_PIN9);
70
71      P3_MOD_OC = P3_MOD_OC |(1<<LED_PIN10);
72      P3_DIR_PU = P3_DIR_PU |    (1<<LED_PIN10);
73
74  for(i=0;i<7;i++){
75        LED1=0;
76        LED2=0;
77        LED3=0;
78        LED4=0;
79        LED5=0;
80        LED6=0;
81        LED7=0;
82        LED8=0;
83        LED9=0;
84        LED10=0;
85         mDelaymS(100);
86        LED1=1;
87        LED2=1;
88        LED3=1;
89        LED4=1;
90        LED5=1;
91        LED6=1;
92        LED7=1;
93        LED8=1;
94        LED9=1;
95        LED10=1;
96         mDelaymS(100);
97     }
98      while (1) {
99         mDelaymS(wait);
100         LED8 = !LED8;
101        mDelaymS(wait);
102         LED2 = !LED2;
103        mDelaymS(wait);
104         LED3 = !LED3;
105        mDelaymS(wait);
106         LED4 = !LED4;
107        mDelaymS(wait);
108         LED5 = !LED5;
109        mDelaymS(wait);
110         LED7 = !LED7;
111        mDelaymS(wait);
112         LED6 = !LED6;
113        mDelaymS(wait);
114         LED1 = !LED1;
115        mDelaymS(wait);
116         LED9 = !LED9;
117        mDelaymS(wait);
118         LED10 = !LED10;
119     }
120 }
```

Listado 3.1 – *LED control with CH554*

Example of LED control in CH559:

```
1  #include "lib/CH559.H"
2
3
4  #include "lib/debug.h"
5
6
7
8  UINT16 wait = 1000;
9  UINT8 i=0;
10
11
```

```
12
13 #define LED_PIN0 7
14 SBIT(LED_0, 0x90, LED_PIN0);
15 #define LED_PIN1 6
16 SBIT(LED_1, 0x90, LED_PIN1);
17 #define LED_PIN2 5
18 SBIT(LED_2, 0x90, LED_PIN2);
19 #define LED_PIN3 4
20 SBIT(LED_3, 0x90, LED_PIN3);
21
22
23
24 void main() {
25     CfgFsys();
26
27
28
29
30
31 for(i=0;i<7;i++){
32       LED_0=0;
33       LED_1=0;
34       LED_2=0;
35       LED_3=0;
36
37        mDelaymS(500);
38
39       LED_0=1;
40       LED_1=1;
41       LED_2=1;
42       LED_3=1;
43
44        mDelaymS(500);
45    }
46     while (1) {
47        mDelaymS(wait);
48         LED_0 = !LED_0;
49        mDelaymS(2*wait);
50         LED_2 = !LED_2;
51        mDelaymS(3*wait);
52         LED_3 = !LED_3;
53
54
55        mDelaymS(6*wait);
56         LED_1 = !LED_1;
57
58     }
59 }
```

Listado 3.2 – *LED control with CH559*

3.2.1.2 LCD screen control

To control the Liquid-Crystal Display (LCD), apart from the power supply (for pins 1,2,15,16) and Read/Write (RW) pin (pin 5 to Ground (GND)), 6 connections are needed that can be used as General Purpose Input/Output (GPIO), so all three models, again, would work for the control of the Liquid-Crystal Display (LCD).

The 4-bit mode would be used, which, although it has a slightly different configuration, is affordable and requires fewer pins than the 8-bit mode.

In this case, it relies on a specific hardware Liquid-Crystal Display (LCD) WH2004D-TMI-JT using the ST7066U driver. Below you can see the data sheet [6]:

Sitronix

ST7066U

Dot Matrix LCD Controller/Driver

■ Features

- **5 x 8 and 5 x 11 dot matrix possible**
- **Low power operation support:**
  -- 2.7 to 5.5V
- **Wide range of LCD driver power**
  -- 3.0 to 10V
- **Correspond to high speed MPU bus interface**
  -- 2 MHz (when V_{CC} = 5V)
- **4-bit or 8-bit MPU interface enabled**
- **80 x 8-bit display RAM (80 characters max.)**
- **13,200-bit character generator ROM for a total of 240 character fonts(5 x 8 dot or 5 x 11 dot)**
- **64 x 8-bit character generator RAM**
  -- 8 character fonts (5 x 8 dot)
  -- 4 character fonts (5 x 11 dot)
- **16-common x 40-segment liquid crystal display driver**
- **Programmable duty cycles**
  -- 1/8 for one line of 5 x 8 dots with cursor
  -- 1/11 for one line of 5 x 11 dots & cursor
  -- 1/16 for two lines of 5 x 8 dots & cursor
- **Wide range of instruction functions:**
  Display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, display shift
- **Automatic reset circuit that initializes the controller/driver after power on**
- **Internal oscillator with external resistors**
- **Low power consumption**
- **QFP80 and Bare Chip available**

■ Description

The ST7066U dot-matrix liquid crystal display controller and driver LSI displays alphanumeric, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.

The ST7066U character generator ROM is extended to generate 240 5x8(5x11) dot character fonts for a total of 240 different character fonts. The low power supply (2.7V to 5.5V) of the ST7066U is suitable for any portable battery-driven product requiring low power dissipation.

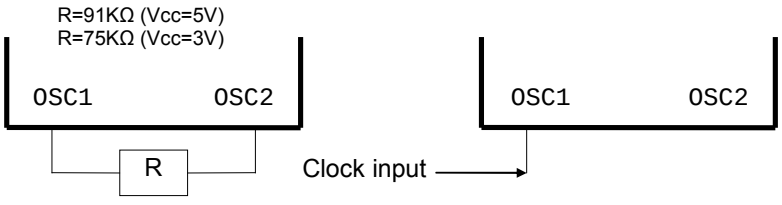
The ST7066U LCD driver consists of 16 common signal drivers and 40 segment signal drivers which can extend display size by cascading segment driver ST7065 or ST7063. The maximum display size can be either 80 characters in 1-line display or 40 characters in 2-line display. A single ST7066U can display up to one 8-character line or two 8-character lines.

Product Name	Support Character
ST7066U-0A	English / Japan
ST7066U-0B	English / European
ST7066U-0E	English / European

■ Pin Function

Name	Number	I/O	Interfaced with	Function
RS	1	I	MPU	Select registers. 0: Instruction register (for write) Busy flag: address counter (for read) 1: Data register (for write and read)
R/W	1	I	MPU	Select read or write. 0: Write 1: Read
E	1	I	MPU	Starts data read/write.
DB4 to DB7	4	I/O	MPU	Four high order bi-directional tristate data bus pins. Used for data transfer and receive between the MPU and the ST7066U. DB7 can be used as a busy flag.
DB0 to DB3	4	I/O	MPU	Four low order bi-directional tristate data bus pins. Used for data transfer and receive between the MPU and the ST7066U. These pins are not used during 4-bit operation.
CL1	1	O	Extension driver	Clock to latch serial data D sent to the extension driver
CL2	1	O	Extension driver	Clock to shift serial data D
M	1	O	Extension driver	Switch signal for converting the liquid crystal drive waveform to AC
D	1	O	Extension driver	Character pattern data corresponding to each segment signal
COM1 to COM16	16	O	LCD	Common signals that are not used are changed to non-selection waveform. COM9 to COM16 are non-selection waveforms at 1/8 duty factor and COM12 to COM16 are non-selection waveforms at 1/11 duty factor.
SEG1 to SEG40	40	O	LCD	Segment signals
V1 to V5	5	-	Power supply	Power supply for LCD drive V_{CC} - V5 = 10 V (Max)
V_{CC}, GND	2	-	Power supply	V_{CC}: 2.7V to 5.5V, GND: 0V
OSC1, OSC2	2		Oscillation resistor clock	When crystal oscillation is performed, a resistor must be connected externally. When the pin input is an external clock, it must be input to OSC1.

Note:
1. Vcc>=V1>=V2>=V3>=V4>=V5 must be maintained
2. Two clock options:



- **4-bit Interface (fosc=270KHz)**

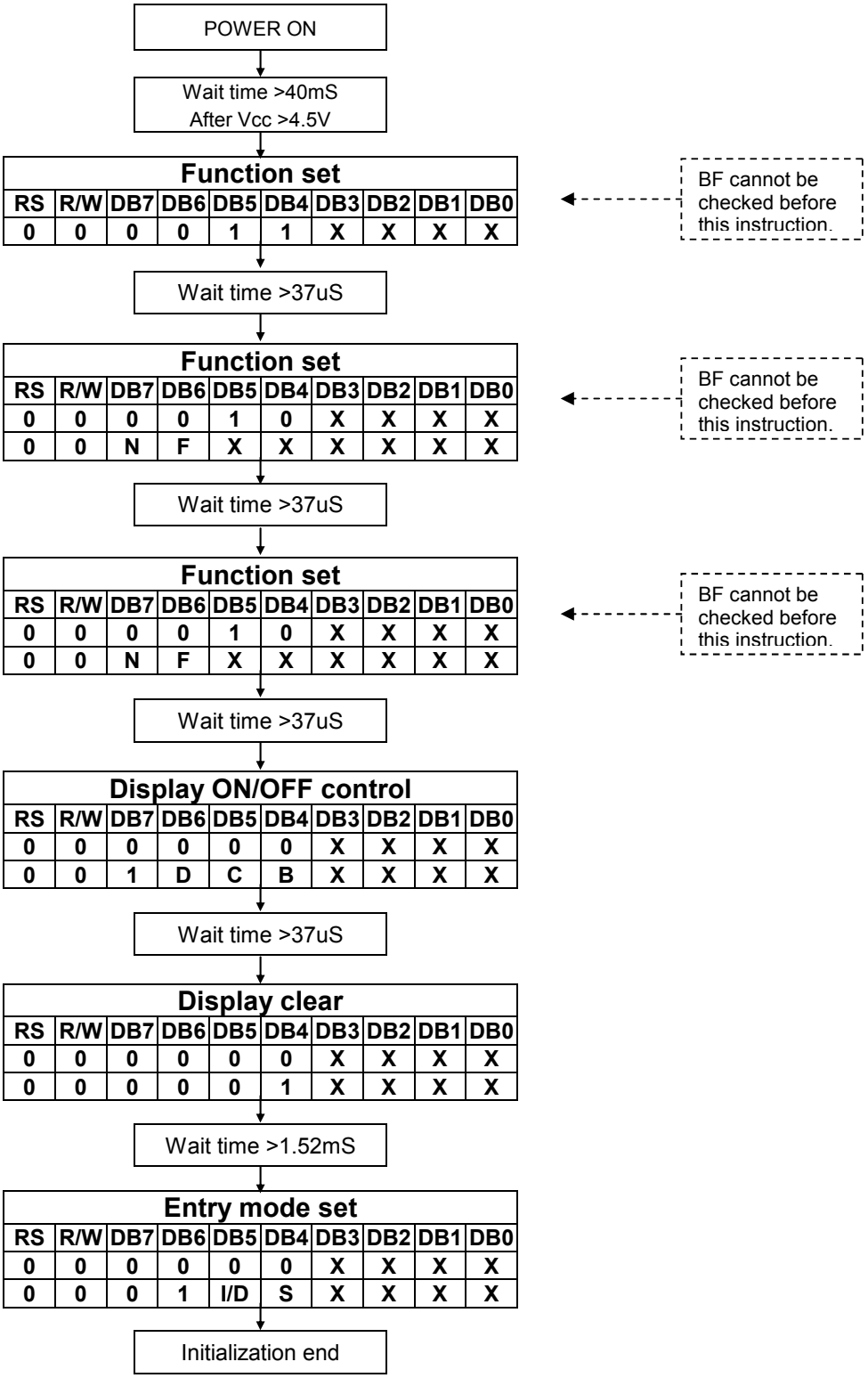


Table 4 Correspondence between Character Codes and Character Patterns (ROM Code: 0A)

NO.7066-0A

- **Display Data RAM (DDRAM)**

Display data RAM (DDRAM) stores display data represented in 8-bit character codes. Its extended capacity is 80 x 8 bits, or 80 characters. The area in display data RAM (DDRAM) that is not used for display can be used as general data RAM. See Figure 1 for the relationships between DDRAM addresses and positions on the liquid crystal display.

The DDRAM address (A_{DD}) is set in the address counter (AC) as hexadecimal.

➢ **1-line display (N = 0) (Figure 2)**

When there are fewer than 80 display characters, the display begins at the head position. For example, if using only the ST7066U, 8 characters are displayed. See Figure 3.
When the display shift operation is performed, the DDRAM address shifts. See Figure 3.

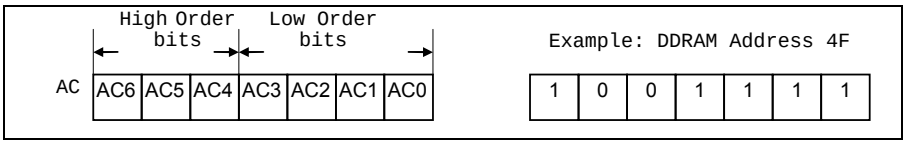


Figure 1 DDRAM Address

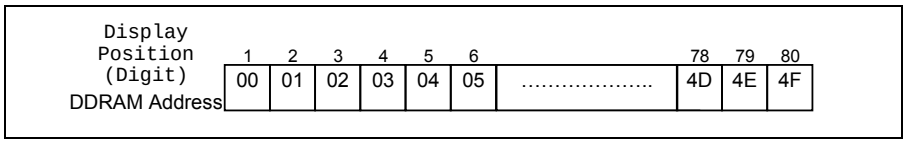


Figure 2 1-Line Display

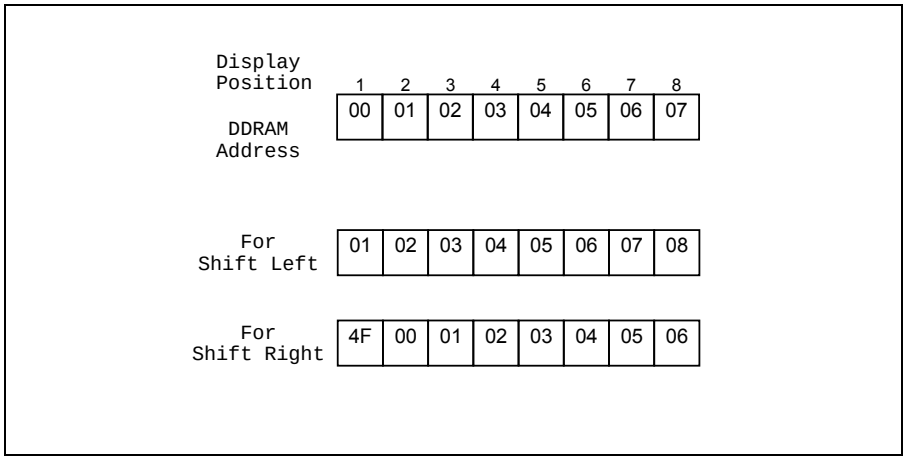


Figure 3 1-Line by 8-Character Display Example

➢ **2-line display (N = 1) (Figure 4)**

Case 1: When the number of display characters is less than 40 × 2 lines, the two lines are displayed from the head. Note that the first line end address and the second line start address are not consecutive. For example, when just the ST7066U is used, 8 characters × 2 lines are displayed. See Figure 5.

When display shift operation is performed, the DDRAM address shifts. See Figure 5.

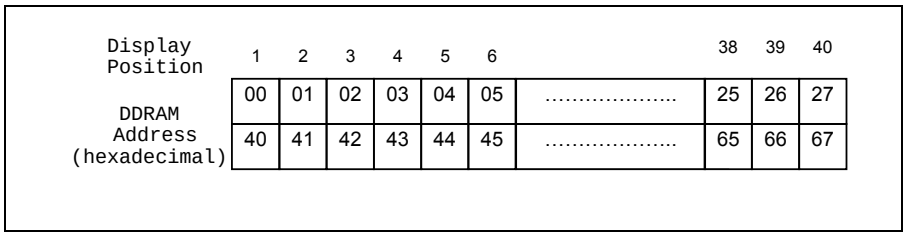


Figure 4 2-Line Display

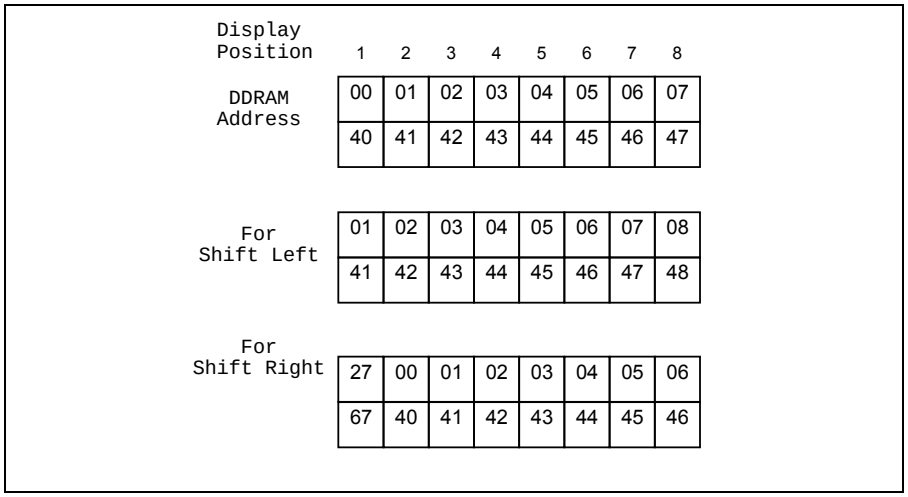


Figure 5 2-Line by 8-Character Display Example

Case 2: For a 16-character × 2-line display, the ST7066U can be extended using one 40-output extension driver. See Figure 6.
When display shift operation is performed, the DDRAM address shifts. See Figure 6.

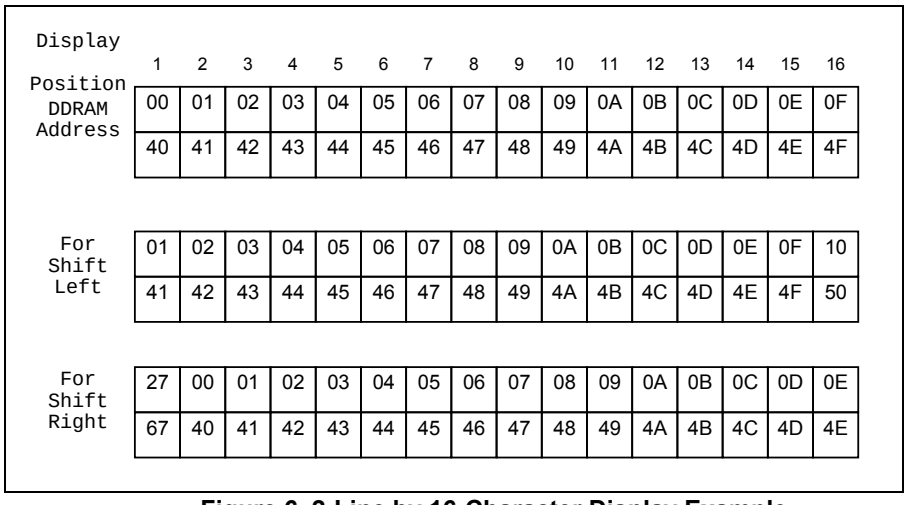


Figure 6 2-Line by 16-Character Display Example

- **Character Generator ROM (CGROM)**

The character generator ROM generates 5 x 8 dot or 5 x 11 dot character patterns from 8-bit character codes. It can generate 240 5 x 8 dot character patterns. User-defined character patterns are also available by mask-programmed ROM.

- **Character Generator RAM (CGRAM)**

In the character generator RAM, the user can rewrite character patterns by program. For 5 x 8 dots, eight character patterns can be written, and for 5 x 11 dots, four character patterns can be written.

Write into DDRAM the character codes at the addresses shown as the left column of Table 4 to show the character patterns stored in CGRAM.

See Table 5 for the relationship between CGRAM addresses and data and display patterns. Areas that are not used for display can be used as general data RAM.

- **Timing Generation Circuit**

The timing generation circuit generates timing signals for the operation of internal circuits such as DDRAM, CGROM and CGRAM. RAM read timing for display and internal operation timing by MPU access are generated separately to avoid interfering with each other. Therefore, when writing data to DDRAM, for example, there will be no undesirable interference, such as flickering, in areas other than the display area.

- **LCD Driver Circuit**

LCD Driver circuit has 16 common and 40 segment signals for LCD driving. Data from CGRAM/CGROM is transferred to 40 bit segment latch serially, and then it is stored to 40 bit shift latch. When each common is selected by 16 bit common register, segment data also output through segment driver from 40 bit segment latch. In case of 1-line display mode, COM1 ~ COM8 have 1/8 duty or COM1 ~ COM11 have 1/11duty , and in 2-line mode, COM1 ~ COM16 have 1/16 duty ratio.

- **Cursor/Blink Control Circuit**

It can generate the cursor or blink in the cursor/blink control circuit. The cursor or the blink appears in the digit at the display data RAM address set in the address counter.

After finding a GitHub library for the ST7066U control and making modifications following the data sheet, the following code is obtained, which although it does not manage to make the Liquid-Crystal Display (LCD) work, it is the one that gives the best results, achieving a blinking of the position where it is necessary to write, but it is not possible to see any character.

The edited header and the main code will be shown below in that order respectively:

```
/*
 ************************************************************************************************************

    This file is part of Library for 8051.
    Library for 8051 is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    Library for 8051 is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with Library for 8051. If not, see <http://www.gnu.org/licenses/>.
 ************************************************************************************************************



***
**    File        : lcd.h
**    Author      : Sriharsha
**    Website     : www.zuna.in
**    Email       : helpzuna@gmail.com
**    Description: This is include file for LCD library for 8051 family MCU's
***/
#ifndef ___lcd__h___
#define ___lcd__h___

#ifndef SDCC
#define SDCC 0
#endif
#ifndef KEIL
#define KEIL 1
#endif

#ifndef TOOLCHAIN
#define TOOLCHAIN SDCC
#endif

#if TOOLCHAIN   == SDCC
//#include "../SDCC\include\mcs51/8051.h"             // 8051 Peripheral Address preprocessor file
#include "CH559.H"
#elif TOOLCHAIN == KEIL
#include <reg51.h>
#else
#error "Invalid Toolchain, Please check 'TOOLCHAIN' macro (SDCC/KEIL)"
#endif

//#include <stdio.h>

// If you are using R/W Pin
#define LCD_RW_ENABLED 1

// LCD Type (20x4,16,2)
#define _LCD_TYPE_ 1                    // 1 -> 40x2 , 2-> 16x2
#define LCD_MODE   2                    // 1-> 8-bit Mode,2-> 4-Bit Mode

// LCD Pin definitions
// RS of LCD Direction,OUTPUT and Bit Position
#define LCD_RS_PORT P3
#define LCD_RS_DIR  P3
#define LCD_RS_BIT  1 << 3

// EN of LCD Direction,OUTPUT and Bit Position
#define LCD_EN_PORT P3
#define LCD_EN_DIR  P3
#define LCD_EN_BIT  1 << 4

#ifdef LCD_RW_ENABLED
// RW of LCD Direction,OUTPUT and Bit Position
#define LCD_RW_PORT P3
#define LCD_RW_DIR  P3
#define LCD_RW_BIT  1 << 1
#endif

// If 8-bit mode
#if LCD_MODE == 1
// D0 of LCD Direction,OUTPUT and Bit Position
#define LCD_D0_PORT P1
#define LCD_D0_DIR  P1
#define LCD_D0_BIT  1 << 0

// D1 of LCD Direction,OUTPUT and Bit Position
#define LCD_D1_PORT P1
#define LCD_D1_DIR  P1
#define LCD_D1_BIT  1 << 1

// D2 of LCD Direction,OUTPUT and Bit Position
#define LCD_D2_PORT P1
#define LCD_D2_DIR  P1
#define LCD_D2_BIT  1 << 2
```

```
89
90  // D3 of LCD Direction,OUTPUT and Bit Position
91  #define LCD_D3_PORT P1
92  #define LCD_D3_DIR  P1
93  #define LCD_D3_BIT  1 << 3
94  #endif
95
96  // Default Mode 4 - Bit (It also includes 8-bit pin definitions)
97  // D4 of LCD Direction,OUTPUT and Bit Position
98  #define LCD_D4_PORT P3
99  #define LCD_D4_DIR  P3
100 #define LCD_D4_BIT  1 << 5
101
102 // D5 of LCD Direction,OUTPUT and Bit Position
103 #define LCD_D5_PORT P3
104 #define LCD_D5_DIR  P3
105 #define LCD_D5_BIT  1 << 6
106
107 // D6 of LCD Direction,OUTPUT and Bit Position
108 #define LCD_D6_PORT P3
109 #define LCD_D6_DIR  P3
110 #define LCD_D6_BIT  1 << 7
111
112 // D7 of LCD Direction,OUTPUT and Bit Position
113 #define LCD_D7_PORT  P3
114 #define LCD_D7_DIR  P3
115 #define LCD_D7_BIT  1 << 2
116
117 // RS(Register Select) LOW and HIGH macros
118 #define LCD_COMMAND_MODE_RS  LCD_RS_PORT &= ~LCD_RS_BIT
119 #define LCD_DATA_MODE_RS     LCD_RS_PORT |=  LCD_RS_BIT
120
121 // RW(Register Select) LOW and HIGH macros
122 #define LCD_RW_LOW       LCD_RW_PORT &= ~LCD_RW_BIT
123 #define LCD_RW_HIGH      LCD_RW_PORT |=  LCD_RW_BIT
124
125 // Enable LOW and HIGH macros
126 #define LCD_ENABLE_LOW         LCD_EN_PORT &= ~LCD_EN_BIT
127 #define LCD_ENABLE_HIGH        LCD_EN_PORT |=  LCD_EN_BIT
128
129 // LCD standard commands
130
131 #define LCD_4BIT_1_LINE_5_x_7              0x20
132 #define LCD_4BIT_2_LINE_5_x_7              0x28
133 #define LCD_8BIT_1_LINE_5_x_7              0x30
134 #define LCD_8BIT_2_LINE_5_x_7              0x38
135 #define LCD_ENTRY_MODE                     0x06
136 #define LCD_DISPLAY_OFF_CURSOR_OFF         0x08
137 #define LCD_DISPLAY_ON_CURSOR_ON           0x0E
138 #define LCD_DISPLAY_ON_CURSOR_OFF          0x0C
139 #define LCD_DISPLAY_ON_CURSOR_BLINKING     0x0F
140 #define LCD_DISPLAY_SHIFT_ENTIRE_LEFT      0x18
141 #define LCD_DISPLAY_SHIFT_ENTIRE_RIGHT     0x1C
142 #define LCD_MOVE_CURSOR_LEFT_ONE_CHAR      0x10
143 #define LCD_MOVE_CURSOR_RIGHT_ONE_CHAR     0x14
144 #define LCD_CLEAR_ALSO_DDRAM               0x01
145 #define LCD_BACK_HOME                      0x02 //Vuelve a la posición columna 0
146 #define LCD_INIT_FINAL_CMM                 0x05
147
148 // LCD Row Addresses
149 #define LCD_ADDRESS_ROW1 0x80
150 #define LCD_ADDRESS_ROW2 0x94
151 #define LCD_ADDRESS_ROW3 0xC0
152 #define LCD_ADDRESS_ROW4 0xD4
153
154 // Delay macros
155 //#define ___delay_ms___(x)  __delay_loop_((long) x*50)
156 //#define ___delay_us___(x)  ___//_loop_((long) x)
157
158 // Prototype Declaration
159 //extern void lcdWrite(unsigned char,int,int);
160 //extern void lcdPrint(unsigned char *,int,int);
161 extern void lcdWrite(unsigned char);
162 extern void lcdPrint(unsigned char *);
163 extern void lcdBegin(void);
164 extern void lcdBegin_7066(void);
165 extern void lcdReset(void);
166 extern void lcdSetCursor(int,int);
167 extern void lcdClear(void);
168 //extern void lcdWriteint(signed int);
169
170 extern void lcd_command(unsigned char);
171 extern void lcd_data(unsigned char);
172 extern void lcdEnable(void);
173 extern void lcd_port_init(void);
174 extern void lcdwrite4Bits(unsigned char);
175 extern void lcdwrite8Bits(unsigned char);
176 extern void lcdSendbyte(unsigned char);
177
178 //extern void ___delay_loop_(signed long);
179
180 #endif
```

Listado 3.3 – *LCD.h*

```
#include "include\CH559_2.H"
#include "include\DEBUG.H"
#include "include\LCD.h"
//#include <compiler.h>
//#include <stdint.h>
//#include <stdio.h>
#include <stdlib.h>
#include <math.h>


//////////////////////
//P1 DEF
//////////////////////
SFR(PORT_CFG, 0xC6); // Port Configuration Register
SFR(P1_DIR, 0xBA);    // P1 port direction control register
SFR(P1, 0x90);        // P1 port input and output register

SBIT(LED6, 0x90, 6); // accessing pin 1.6


//////////////////////
//TIMER0 DEF
//////////////////////
#define mTimer0ClkFsys( ) (T2MOD |= bTMR_CLK | bT0_CLK)
#define mTimer0Clk4DivFsys( ) (T2MOD &= ~bTMR_CLK;T2MOD |=  bT0_CLK)
#define mTimer0Clk12DivFsys( ) (T2MOD &= ~(bTMR_CLK | bT0_CLK))
#define mTimer0CountClk( ) (TMOD |= bT0_CT)


#define mTimer0RunCTL( SS ) (TR0 = SS ? START : STOP)

#ifndef     BUAD
#define     BUAD    57600
#endif

//////////////////////
//PWM DEF
//////////////////////
#define SetPWMClk(CK_SE) (PWM_CK_SE = CK_SE)
#define SetPWMCycle(Cycle) (PWM_CYCLE = Cycle)
#define SetPWM1Dat(dat) (PWM_DATA = dat)
#define SetPWM2Dat(dat) (PWM_DATA2 = dat)
#define PWMPINAlter( )  { P4_DIR |= bPWM2_ | bPWM1_;PIN_FUNC |= bPWM1_PIN_X;}

////////////////////////////////////////////////////////////////////////////////
// PWM functions
////////////////////////////////////////////////////////////////////////////////

void  InitPWM1(uint8_t polar)
{
    PWM_CTRL &= ~bPWM_CLR_ALL;
    PWM_CTRL &= ~bPWM_MOD_MFM;
    PWM_CTRL |=   bPWM_IE_END;
    PWM_CTRL |=   bPWM_OUT_EN;
    PWM_CTRL  |= bPWM_IF_END;
    if(polar){
        PWM_CTRL |= bPWM_POLAR;
    }
    else{
        PWM_CTRL &= ~bPWM_POLAR;
    }
}

/*******************************************************************************
* Function Name  : InitPWM2(UINT8 polar)
* Description    : PWM
* Input          : polar=0;
                   polar=1;
* Output         : None
* Return         : None
*******************************************************************************/
void  InitPWM2(uint8_t polar)
{
    PWM_CTRL &= ~bPWM_CLR_ALL;
    PWM_CTRL &= ~bPWM_MOD_MFM;
    PWM_CTRL |=   bPWM_IE_END;
    PWM_CTRL |= bPWM2_OUT_EN;
    PWM_CTRL  |= bPWM_IF_END;
    if(polar){
        PWM_CTRL |= bPWM2_POLAR;
    }
    else{
        PWM_CTRL &= ~bPWM2_POLAR;
    }
}

/*******************************************************************************
* Function Name  : PWMInterrupt(void)
* Description    : PWM
*******************************************************************************/
void    PWMInterrupt( void )// interrupt INT_NO_PWM1 using 1
{
    if(PWM_CTRL & bPWM_IF_END)
    {
        PWM_CTRL |= bPWM_IF_END;
        //printf("PWM_DATA  %02X\n",(UINT16)PWM_DATA);
    }
}

////////////////////////////////////////////////////////////////////////////////
// TIMER0 functions
////////////////////////////////////////////////////////////////////////////////
void    mInitSTDIO( )
{
```

```
105     uint32_t x;
106     uint8_t x2;
107
108     SM0 = 0;
109     SM1 = 1;
110     SM2 = 0;                                                                    // Mode 1
111                                                                 // Timer 1 generate baud
112     RCLK = 0;                                                                   // RX clk
113     TCLK = 0;                                                                   // TX clk
114     PCON |= SMOD;
115     x = 10 * FREQ_SYS / BUAD / 16;                                              // If system
          clock changed, please check whether x overflow
116     x2 = x % 10;
117     x /= 10;
118     if (x2 >= 5){
119     x +=1;
120     }// Data round
121
122     TMOD = TMOD & ~ bT1_GATE & ~ bT1_CT & ~ MASK_T1_MOD | bT1_M1;
123     T2MOD = T2MOD | bTMR_CLK | bT1_CLK;                                         // Select
          Timer1 clock
124     TH1 = 0-x;                                                                  // Set baud
          count
125     TR1 = 1;                                                                    // Start
          Timer1
126     TI = 1;
127     REN = 1;                                                                    // Enable
          COM0
128 }
129
130 void mTimer0ModSetup(uint8_t mode)
131 {
132     TMOD &= 0xf0;
133     TMOD |= mode;
134 }
135
136 /*******************************************************************************
137 * Function Name  : mTimer0SetData(UINT16 dat)
138 * Description    : CH559Timer0 TH0
139 * Input          : UINT16 dat;
140 * Output         : None
141 * Return         : None
142 *******************************************************************************/
143 void mTimer0SetData(uint16_t dat)
144 {
145     UINT16 tmp;
146     tmp = 65536 - dat;
147     TL0 = tmp & 0xff;
148     TH0 = (tmp>>8) & 0xff;
149 }
150
151 /*******************************************************************************
152 * Function Name  : mTimer0Interrupt()
153
154 *******************************************************************************/
155 void    mTimer0Interrupt( void )// interrupt INT_NO_TMR0 using 1
156 {
157     CAP1 = !CAP1;
158 //  mTimer0SetData(0x2000)
159 }
160
161 ////////////////////////////////////////////////////////////////////////////////
162 // GPIO functions
163 ////////////////////////////////////////////////////////////////////////////////
164
165
166 void CH559GPIODrivCap(uint8_t Port,uint8_t Cap)
167 {
168    if(Port >= 4){
169
170    }
171    if(Cap == 0){
172     PORT_CFG &= ~(bP0_DRV << Port);
173   }
174   else{
175     PORT_CFG |= (bP0_DRV << Port);
176   }
177
178 }
179
180 /*******************************************************************************
181 * Function Name  : CH559GPIOModeSelt(UINT8 Port,UINT8 Mode,UINT8 PinNum)
182 *******************************************************************************/
183 void CH559GPIOModeSelt(uint8_t Port,uint8_t Mode,uint8_t PinNum)
184 {
185   uint8_t Pn_DIR,Pn_PU;
186
187   switch (Mode){
188         case 0:
189            PORT_CFG &= ~(bP0_OC << Port);
190            Pn_DIR &= ~(1<<PinNum);
191            Pn_PU &= ~(1<<PinNum);
192            break;
193         case 1:
194            PORT_CFG &= ~(bP0_OC << Port);
195            Pn_DIR &= ~(1<<PinNum);
196            Pn_PU |= 1<<PinNum;
197            break;
198         case 2:
199            PORT_CFG &= ~(bP0_OC << Port);
200            Pn_DIR |= ~(1<<PinNum);
201            break;
202         case 3:
203            PORT_CFG |= (bP0_OC << Port);
204            Pn_DIR &= ~(1<<PinNum);
```

```
205             Pn_PU &= ~(1<<PinNum);
206             break;
207         case 4:
208             PORT_CFG |= (bP0_OC << Port);
209             Pn_DIR |= 1<<PinNum;
210             Pn_PU &= ~(1<<PinNum);
211             break;
212         case 5:
213             PORT_CFG |= (bP0_OC << Port);
214             Pn_DIR &= ~(1<<PinNum);
215             Pn_PU |= 1<<PinNum;
216             break;
217         case 6:
218             PORT_CFG |= (bP0_OC << Port);
219             Pn_DIR |= 1<<PinNum;
220             Pn_PU |= 1<<PinNum;
221             break;
222         default:
223             break;
224     }
225     if(Port == 0){
226         P0_DIR = Pn_DIR;
227         P0_PU = Pn_PU;
228     }
229     if(Port == 1){
230         P1_DIR = Pn_DIR;
231         P1_PU = Pn_PU;
232     }
233     if(Port == 2){
234         P2_DIR = Pn_DIR;
235         P2_PU = Pn_PU;
236     }
237     if(Port == 3){
238         P3_DIR = Pn_DIR;
239         P3_PU = Pn_PU;
240     }
241
242 }
243
244 /*******************************************************************************
245 * Function Name  : CH559P4Mode()
246 *******************************************************************************/
247 void CH559P4Mode( )
248 {
249     P4_DIR |= 0xff;
250     P4_PU |= 0xff;
251     P4_CFG |= bP4_DRV;
252 }
253
254 /*******************************************************************************
255 * Function Name  : CH559GPIOInterruptInit()
256 *******************************************************************************/
257 void CH559GPIOInterruptInit()
258 {
259     GPIO_IE &= ~bIE_IO_EDGE;
260     GPIO_IE |= bIE_RXD1_LO;
261 }
262
263
264
265 ////////////////////////////////////////////////////////////////////////////////
266 // LCD functions
267 ////////////////////////////////////////////////////////////////////////////////
268
269
270
271 ////////////////////////////////////////////////////////////////////////////////
272 // Own functions
273 ////////////////////////////////////////////////////////////////////////////////
274 void delayu () // For 1uS
275     {
276      TMOD = 0x01;      // Timer 0 Mode 1
277      TH0= 0xFF;       //initial value for 1ms
278      TL0 = 0xFF;
279      TR0 = 1;       // timer start
280      while (TF0 == 0); // check overflow condition
281      TR0 = 0;      // Stop Timer
282      TF0 = 0;     // Clear flag
283     }
284
285
286 void delaym () //For 1mS
287   {
288     TMOD = 0x01;      // Timer 0 Mode 1
289     TH0= 0xFC;       //initial value for 1ms
290     TL0 = 0x18;
291     TR0 = 1;       // timer start
292     while (TF0 == 0); // check overflow condition
293     TR0 = 0;      // Stop Timer
294     TF0 = 0;     // Clear flag
295   }
296
297 void delayS()
298   {
299   uint16_t j;
300
301   for ( j=0;j<1000;j++)
302    {
303
304    delaym();
305
306    }
307
308
309   }
```

```
310
311 void LED_ON_time()
312 {
313     //uint32_t T_ON, uint32_t T_OFF
314     //CH559GPIOModeSelt(1,6,6);
315     LED6=1;
316     delayS();
317
318     //CH559GPIOModeSelt(1,0,6);
319     LED6=0;
320     delayS();
321 }
322
323 void LED_PWM_MODE (uint16_t frec,uint8_t duti)
324 {
325     uint32_t TperiodoUS = (1000000 / frec);
326     uint32_t TperiodoON = TperiodoUS * duti / 100;
327     uint32_t TperiodoOFF = TperiodoUS - TperiodoON;
328
329     LED_ON_time();
330     //TperiodoON, TperiodoOFF
331 }
332
333 ////////////////////////////////////
334 //LCD funct
335 /*** Function    : lcdEnable
336 **    Parameters  : None
337 **    Return      : None
338 **    Description : It will trigger LCD Enable Signal
339 **/
340 void lcdEnable(void)
341 {
342     LCD_ENABLE_HIGH;
343     delayS();
344     LCD_ENABLE_LOW;
345 }
346
347 /*** Function    : lcd_port_init
348 **    Parameters  : None
349 **    Return      : None
350 **    Description : It will Initiate the I/O Port for LCD
351 **/
352 void lcd_port_init(void)
353 {
354 // SET RS Port as OUTPUT and Write LOW to PORT
355 LCD_RS_DIR  |= LCD_RS_BIT;
356 LCD_RS_PORT &= ~LCD_RS_BIT;
357 //LCD_RS_PORT |= LCD_RS_BIT;
358
359 // SET RW Port as OUTPUT and Write LOW to PORT
360 //LCD_RW_DIR  |= LCD_RW_BIT;
361 //LCD_RW_PORT &= ~LCD_RW_BIT;
362 //LCD_RW_PORT |= LCD_RW_BIT;
363
364 // SET EN Port as OUTPUT and Write LOW to PORT
365 LCD_EN_DIR  |= LCD_EN_BIT;
366 LCD_EN_PORT &= ~LCD_EN_BIT;
367 //LCD_EN_PORT |= LCD_EN_BIT;
368
369 // If 8- Bit Mode
370 #if   LCD_MODE == 1
371 // SET D0 Port as OUTPUT and Write HIGH to PORT
372 LCD_D0_DIR  |= LCD_D0_BIT;
373 LCD_D0_PORT |= LCD_D0_BIT;
374 // SET D1 Port as OUTPUT and Write HIGH to PORT
375 LCD_D1_DIR  |= LCD_D1_BIT;
376 LCD_D1_PORT |= LCD_D1_BIT;
377 // SET D2 Port as OUTPUT and Write HIGH to PORT
378 LCD_D2_DIR  |= LCD_D2_BIT;
379 LCD_D2_PORT |= LCD_D2_BIT;
380 // SET D3 Port as OUTPUT and Write HIGH to PORT
381 LCD_D3_DIR  |= LCD_D3_BIT;
382 LCD_D3_PORT |= LCD_D3_BIT;
383 // SET D4 Port as OUTPUT and Write HIGH to PORT
384 LCD_D4_DIR  |= LCD_D4_BIT;
385 LCD_D4_PORT |= LCD_D4_BIT;
386 // SET D5 Port as OUTPUT and Write HIGH to PORT
387 LCD_D5_DIR  |= LCD_D5_BIT;
388 LCD_D5_PORT |= LCD_D5_BIT;
389 // SET D6 Port as OUTPUT and Write HIGH to PORT
390 LCD_D6_DIR  |= LCD_D6_BIT;
391 LCD_D6_PORT |= LCD_D6_BIT;
392 // SET D7 Port as OUTPUT and Write HIGH to PORT
393 LCD_D7_DIR  |= LCD_D7_BIT;
394 LCD_D7_PORT |= LCD_D7_BIT;
395 // If 4-Bit Mode
396 #elif LCD_MODE == 2
397 // SET D4 Port as OUTPUT and Write HIGH to PORT
398 LCD_D4_DIR  |= LCD_D4_BIT;
399 LCD_D4_PORT |= LCD_D4_BIT;
400 // SET D5 Port as OUTPUT and Write HIGH to PORT
401 LCD_D5_DIR  |= LCD_D5_BIT;
402 LCD_D5_PORT |= LCD_D5_BIT;
403 // SET D6 Port as OUTPUT and Write HIGH to PORT
404 LCD_D6_DIR  |= LCD_D6_BIT;
405 LCD_D6_PORT |= LCD_D6_BIT;
406 // SET D7 Port as OUTPUT and Write HIGH to PORT
407 LCD_D7_DIR  |= LCD_D7_BIT;
408 LCD_D7_PORT |= LCD_D7_BIT;
409
410 // Else display an error
411 #else
412 #error "Invalid LCD TYPE (That should be either '1' or '2')"
413 #endif
414 }
```

```
415
416 /*** Function    : lcdwrite8Bits
417 **   Parameters  : None
418 **   Return      : unsigned char
419 **   Description : It will write 8-bit to LCD PORT
420 **/
421 void lcdwrite8Bits(unsigned char lByte)
422 {
423     if(lByte & 0x80)
424        LCD_D7_PORT |=  LCD_D7_BIT;
425     else
426        LCD_D7_PORT &= ~LCD_D7_BIT;
427     if(lByte & 0x40)
428        LCD_D6_PORT |=  LCD_D6_BIT;
429     else
430        LCD_D6_PORT &= ~LCD_D6_BIT;
431     if(lByte & 0x20)
432        LCD_D5_PORT |=  LCD_D5_BIT;
433     else
434        LCD_D5_PORT &= ~LCD_D5_BIT;
435     if(lByte & 0x10)
436        LCD_D4_PORT |=  LCD_D4_BIT;
437     else
438        LCD_D4_PORT &= ~LCD_D4_BIT;
439 #if LCD_MODE == 1
440     if(lByte & 0x08)
441           LCD_D3_PORT |=  LCD_D3_BIT;
442        else
443           LCD_D3_PORT &= ~LCD_D3_BIT;
444        if(lByte & 0x04)
445           LCD_D2_PORT |=  LCD_D2_BIT;
446        else
447           LCD_D2_PORT &= ~LCD_D2_BIT;
448        if(lByte & 0x02)
449           LCD_D1_PORT |=  LCD_D1_BIT;
450        else
451           LCD_D1_PORT &= ~LCD_D1_BIT;
452        if(lByte & 0x01)
453           LCD_D0_PORT |=  LCD_D0_BIT;
454        else
455           LCD_D0_PORT &= ~LCD_D0_BIT;
456 #endif
457 lcdEnable();
458 }
459
460 //#define   LOWNIB(x)   P1OUT = (P1OUT & 0xF0) + (x & 0x0F)
461
462 /*** Function    : lcdwrite4Bits
463 **   Parameters  : None
464 **   Return      : unsigned char
465 **   Description : It will write 4-bit to LCD PORT
466 **/
467 void lcdwrite4Bits(unsigned char val)
468 {
469     if(val & 0x08)
470        LCD_D7_PORT |=  LCD_D7_BIT;
471     else
472        LCD_D7_PORT &= ~LCD_D7_BIT;
473     if(val & 0x04)
474        LCD_D6_PORT |=  LCD_D6_BIT;
475     else
476        LCD_D6_PORT &= ~LCD_D6_BIT;
477     if(val & 0x02)
478        LCD_D5_PORT |=  LCD_D5_BIT;
479     else
480        LCD_D5_PORT &= ~LCD_D5_BIT;
481     if(val & 0x01)
482        LCD_D4_PORT |=  LCD_D4_BIT;
483     else
484        LCD_D4_PORT &= ~LCD_D4_BIT;
485     lcdEnable();
486 }
487
488 /*** Function    : lcdSendbyte
489 **   Parameters  : None
490 **   Return      : unsigned char
491 **   Description : It will Send byte to LCD based on 8-bit/4-bit
492 **/
493 void lcdSendbyte(unsigned char lByte)
494 {
495     #if LCD_MODE == 1
496     lcdwrite8Bits(lByte);
497     #elif LCD_MODE == 2
498     lcdwrite4Bits(lByte >> 4);
499     lcdwrite4Bits(lByte);
500      delayS();
501     #endif
502 }
503
504 /*** Function    : lcd_command
505 **   Parameters  : None
506 **   Return      : unsigned char
507 **   Description : It will Send byte to lcd as command
508 **/
509 void lcd_command(unsigned char lByte)
510 {
511    LCD_ENABLE_LOW;
512    LCD_COMMAND_MODE_RS;
513    lcdSendbyte(lByte);
514 }
515
516 /*** Function    : lcd_data
517 **   Parameters  : None
518 **   Return      : unsigned char
519 **   Description : It will Send byte to LCD as data
```

```
**/
void lcd_data(unsigned char lByte)
{
   LCD_ENABLE_LOW;
   LCD_DATA_MODE_RS;
   lcdSendbyte(lByte);
}



/*** Function    : lcd_data
**   Parameters  : None
**   Return      : unsigned char
**   Description : It will send one char to LCD as data
**/
void lcdWrite(unsigned char lByte)
{
   lcd_data(lByte);
}

/*void lcdWrite_2(unsigned char lByte, int line,int col)
{
   uint8_t lBpos;
   lBpos=lcdSetCursor(line,col);
   lcd_command(lByte);
   delaym();
   LCD_RW_LOW;
   lcd_command(lBpos);
   delaym();
   lcd_command(lBpos&~(0x80));
   delaym();
   LCD_RW_LOW;

}*/


/*** Function    : lcdPrint
**   Parameters  : None
**   Return      : unsigned char*
**   Description : It will send String to LCD
**/
void lcdPrint(unsigned char *lByte)
{
   for(;*lByte!='\0';lByte++)
   lcdWrite(*lByte);
}

/*void lcdPrint_2(unsigned char *lByte,int line,int col)
{
   for(;*lByte!='\0';lByte++)
   lcdWrite(*lByte,line,col);
   col++;
}*/

/*** Function    : lcdSetCursor
**   Parameters  : None
**   Return      : int,int
**   Description : It will set row and column of the lcd
**/
void lcdSetCursor(int line,int col)
{
#if _LCD_TYPE_ == 1
if(line==1)
lcd_command(LCD_ADDRESS_ROW1 + col);
else if(line == 2)
lcd_command(LCD_ADDRESS_ROW2 + col);
else if(line == 3)
lcd_command(LCD_ADDRESS_ROW3 + col);
else
lcd_command(LCD_ADDRESS_ROW4 + col);
#elif _LCD_TYPE_ == 2
if(line==1)
lcd_command(LCD_ADDRESS_ROW1 + col);
else
lcd_command(LCD_ADDRESS_ROW2 + col);
#endif
}

/*uint8_t lcdSetCursor_2(int line,int col)
{
uint8_t lBpos;
#if _LCD_TYPE_ == 1
if(line==1)
lBpos=LCD_ADDRESS_ROW1 + col;
//lcd_command(LCD_ADDRESS_ROW1 + col);
else if(line == 2)
lBpos=LCD_ADDRESS_ROW2 + col;
//lcd_command(LCD_ADDRESS_ROW2 + col);
else if(line == 3)
lBpos=LCD_ADDRESS_ROW3 + col;
//lcd_command(LCD_ADDRESS_ROW3 + col);
else
lBpos=LCD_ADDRESS_ROW4 + col;
//lcd_command(LCD_ADDRESS_ROW4 + col);
#elif _LCD_TYPE_ == 2
if(line==1)
lcd_command(LCD_ADDRESS_ROW1 + col);
else
lcd_command(LCD_ADDRESS_ROW2 + col);
#endif

return lBpos;
}*/

/*** Function    : lcdClear
```

```
**    Parameters  : None
**    Return      : None
**    Description : It will Clear the LCD screen and DDRAM of the LCD
**/
void lcdClear(void)
{
    lcd_command(LCD_CLEAR_ALSO_DDRAM);
}

/*** Function    : lcdReset
**    Parameters  : None
**    Return      : None
**    Description : It will Reset the LCD for 4-bit Mode
**/
void lcdReset(void)
{
    /*LCD_D7_PORT &= ~LCD_D7_BIT;
    LCD_D6_PORT &= ~LCD_D6_BIT;
    LCD_D5_PORT |= LCD_D5_BIT;
    LCD_D4_PORT |= LCD_D4_BIT;

    lcdEnable();
    delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5
     delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5
    lcdEnable();
    delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5
     delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5
    lcdEnable();
    delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5
     delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5

    LCD_D7_PORT &= ~LCD_D7_BIT;
    LCD_D6_PORT &= ~LCD_D6_BIT;
    LCD_D5_PORT |= LCD_D5_BIT;
    LCD_D4_PORT &= ~LCD_D4_BIT;

    lcdEnable();
    delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5
     delaym();//1
     delaym();
     delaym();
     delaym();
     delaym();//5*/

     lcd_command(0x30);
     delayS();
     delayS();
     //5
     //1
     lcd_command(0x30);
     delayS();
     lcd_command(0x30);
     delayS();

}

/*** Function    : lcdBegin
**    Parameters  : None
**    Return      : None
**    Description : It will initiate the lcd
**/
void lcdBegin(void)
{
    lcd_port_init();
     #if LCD_MODE == 1
    lcd_command(LCD_8BIT_2_LINE_5_x_7);
    //_lcd_delay;
     #elif LCD_MODE == 2
    lcdReset();
    lcd_command(LCD_4BIT_2_LINE_5_x_7);
    delayS();
    lcd_command(LCD_CLEAR_ALSO_DDRAM);
    delayS();
    delayS();
    //delaym();


```

```
730     #endif
731     lcd_command(LCD_DISPLAY_ON_CURSOR_ON);
732     delayS();
733 
734     lcd_command(LCD_ENTRY_MODE);
735     delayS();
736     //lcd_command(LCD_ADDRESS_ROW1);
737 }
738 
739     #define LCD_REFRESH_DELAY    5000
740 
741     #if _LCD_TYPE_ == 1
742     //#define String1   "     Hello World     "
743     __code unsigned char String1[]={'H','e','l','l','o',' ','W','o','r','l','d'};
744     //#define String2   "     Hai 8051        "
745     __code unsigned char String2[]={'H','a','i',' ','8','0','5','1',' ',' ',' '};
746     #define String3   "     This is         "
747     #define String4   "     LCD TEST        "
748     // Otherwise if 16x2 define below string
749     #elif _LCD_TYPE_ == 2
750     #define String1     'Hello World'
751     #define String2   "    Hai 8051     "
752     #endif
753 
754 void lcd_puts(unsigned char* str){
755     while((*str) != '\0'){
756         lcd_data(*str++);
757     }
758 }
759 
760 void main()
761 {
762 
763 
764 
765 
766     mInitSTDIO();
767 
768 //     mTimer0ModSetup(1);
769 //     mTimer0ClkFsys( );
770 //     mTimer0SetData(1000);
771 //     mTimer0RunCTL( 1 );
772 
773 
774 
775 
776 
777     ////////////////////////////////////
778 
779 
780     ////////////////////////////////
781     //PWM
782     SetPWMClk(12);
783     InitPWM1(0);
784     InitPWM2(1);
785     SetPWMCycle(127);
786     IE_PWMX = 1;
787     SetPWM1Dat(50);
788     SetPWM2Dat(50);
789     ////////////////////////////////
790 
791 
792 
793     ////////////////////////////////
794     //P1
795     PORT_CFG = 0b00101101;
796     P1_DIR = 0b11110000;
797     P1 = 0x00;
798     ////////////////////////////////
799 
800     uint8_t PWM_MODE=1;
801     uint16_t frec = 100;
802     uint8_t duti = 50;
803     uint32_t TperiodoMS = (1000 / frec);
804     uint32_t TperiodoON = TperiodoMS * duti / 100;
805     uint32_t TperiodoOFF = TperiodoMS - TperiodoON;
806 
807     ET0 = 1;
808     EA = 1;
809 
810     ////////////////////////////////
811     //LCD
812     delayS();
813     lcdBegin();
814     //lcd_command(LCD_DISPLAY_ON_CURSOR_OFF);
815 
816 
817 
818     while(1)
819     {
820 
821     #if _LCD_TYPE_ == 1
822     lcdSetCursor(1,0);
823     lcd_puts(String4);
824     delayS();
825     delayS();
826     delayS();
827     delayS();
828     delayS();
829     lcdSetCursor(2,0);
830     lcd_puts(String2);
831     delayS();
832     delayS();
833     delayS();
834     delayS();
```

```
835     delayS();
836     lcd_command(LCD_ENTRY_MODE);
837     delayS();
838     lcdSetCursor(3,0);
839     delayS();
840     lcdPrint(String4);
841     delayS();
842     delayS();
843     delayS();
844     delayS();
845     delayS();
846     lcd_command(LCD_ENTRY_MODE);
847     delayS();
848     lcdSetCursor(4,0);
849     delayS();
850     lcd_puts(String2);
851     delayS();
852     delayS();
853     delayS();
854     delayS();
855     delayS();
856     lcdClear();
857     delayS();
858     #elif _LCD_TYPE_ == 2
859    lcdSetCursor(1,1);
860     lcdPrint(String1);
861     delayS();
862     lcdSetCursor(2,0);
863     lcdPrint(String2);
864     delayS();
865     lcdClear();
866
867     #endif
868
869
870
871    }
872 }
```

Listado 3.4 – *LCD control with CH559*

The main code uses the micro controller's own functions (Pulse-Width Modulation (PWM), General Purpose Input/Output (GPIO) and TIMER0). It is important to use TIMER0 to have a stable delay function, as the time between instructions must be respected for the display to start working. . Although some functions in this code are not used in the main function due to the variations that the code has undergone to obtain the current one, it is important that they are reflected as it will help in a future improvement.

3.2.1.3 External peripheral control for menu navigation with button to reprogram the micro controller and to select options

First of all, it should be made clear that as the specific hardware code (PEC11R-4120F-S0018) was not available at the time the code was created, a joystick (HW-504) with a button function for reprogramming was used and, as it was not possible to make the Liquid-Crystal Display (LCD) work, with the 4 Light-Emitting Diode (LED)s on the EValuation Test (EVT) board, it was possible to move around them as an example of how to use them [2].

The main code will be shown below:

```
1
2
3  #include "include\CH559_2.H"
4  #include "include\DEBUG.H"
5  //#include <stdint.h>
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <math.h>
9
10 /******************************** (C) COPYRIGHT *******************************
11 * File Name          : ADCManual.C
12 * Author             : WCH
13 * Version            : V1.3
14 * Date               : 2016/6/24
15 * Description        :
16 *******************************************************************************/
17
18
19 uint8_t ADCChannel[8] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
20
21  #define _INT_  1
22
23 //#pragma  NOAREGS
24
25
```

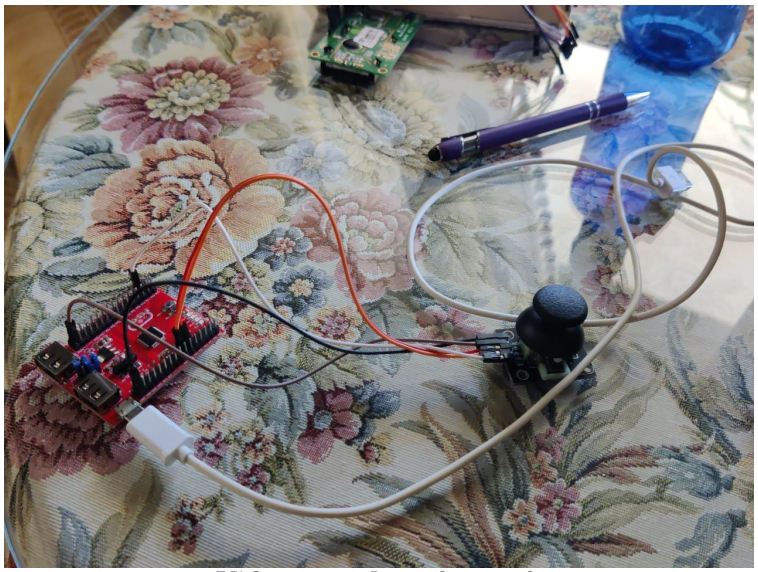
```
/*******************************************************************************
* Function Name  : InitADCManual()

*******************************************************************************/
static inline void delay() {
    uint32_t ki;
    for (ki = 0; ki < (700); ki++){}
        __asm__("nop");
}

void InitADCManual()
{
    P1_IE = 0x00;
    ADC_SETUP |= bADC_POWER_EN;
    ADC_CK_SE = 0x02;
    ADC_CTRL &= ~MASK_ADC_CYCLE;
    ADC_CTRL &= ~(bADC_CHANN_MOD1 | bADC_CHANN_MOD0);
    ADC_CHANN = ADCChannel[0];
//    ADC_EX_SW |= bADC_RESOLUTION;
   ADC_EX_SW &= ~bADC_RESOLUTION;

}


/*******************************************************************************
* Function Name  : ADCChanelChange()
* Input          : uint8_t Chanel
* Output         : None
* Return         : uint16_t ADCValue
*******************************************************************************/
uint16_t ADCChanelChange(uint8_t Chanel)
{
  unsigned short ADCValue = 0;
   ADC_CHANN = Chanel;
//    ADC_EX_SW |= bADC_RESOLUTION;
    delay();
   ADC_CTRL |= bADC_SAMPLE;
    delay();
   ADC_CTRL &= ~bADC_SAMPLE;
   while((ADC_STAT & bADC_IF_ACT) == 0);
   ADC_STAT |= bADC_IF_ACT;
   ADCValue = ADC_FIFO;
  return ADCValue;
}

#ifdef _INT_
/*******************************************************************************
* Function Name  : InitADCInterrupt()
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************/
void InitADCInterrupt()
{
    ADC_SETUP |= bADC_IE_FIFO_OV;
//   ADC_SETUP |= bADC_IE_AIN7_LOW;
    ADC_SETUP |= bADC_IE_ACT;
    IE_ADC = 1;
}

/*******************************************************************************
* Function Name  : ADCInterrupt(void)
*******************************************************************************/
void    ADCInterrupt( void ) //__interrupt INT_NO_ADC __using 1
{
    uint16_t ADCValue = 0;
    if(ADC_STAT & bADC_IF_ACT)
    {
       ADC_STAT |= bADC_IF_ACT;
    }
    ADCValue = ADC_FIFO;
     //printf("FIFOCnt:%02X  ADC_DATA:%04X  \n",(uint16_t)(ADC_STAT&3),(uint16_t)ADC_FIFO);

#if 1
    ADC_CTRL |= bADC_SAMPLE;
    delay();
    ADC_CTRL &= ~bADC_SAMPLE;
#endif
}
#endif

//#pragma  NOAREGS

/*******************************************************************************
* Function Name  : CH559GPIODrivCap(UINT8 Port,UINT8 Cap)

*******************************************************************************/
void CH559GPIODrivCap(UINT8 Port,UINT8 Cap)
{
   if(Port >= 4){

   }
   if(Cap == 0){
    PORT_CFG &= ~(bP0_DRV << Port);
  }
  else{
    PORT_CFG |= (bP0_DRV << Port);
  }

}

/*******************************************************************************
* Function Name  : CH559GPIOModeSelt(UINT8 Port,UINT8 Mode,UINT8 PinNum)
*******************************************************************************/
void CH559GPIOModeSelt(uint8_t Port,uint8_t Mode,uint8_t PinNum)
```

```
{
  uint8_t Pn_DIR,Pn_PU;

  switch (Mode){
        case 0:
           PORT_CFG &= ~(bP0_OC << Port);
           Pn_DIR &= ~(1<<PinNum);
           Pn_PU &= ~(1<<PinNum);
           break;
        case 1:
           PORT_CFG &= ~(bP0_OC << Port);
           Pn_DIR &= ~(1<<PinNum);
           Pn_PU |= 1<<PinNum;
           break;
        case 2:
           PORT_CFG &= ~(bP0_OC << Port);
           Pn_DIR |= ~(1<<PinNum);
           break;
        case 3:
           PORT_CFG |= (bP0_OC << Port);
           Pn_DIR &= ~(1<<PinNum);
           Pn_PU &= ~(1<<PinNum);
           break;
        case 4:
           PORT_CFG |= (bP0_OC << Port);
           Pn_DIR |= 1<<PinNum;
           Pn_PU &= ~(1<<PinNum);
           break;
        case 5:
           PORT_CFG |= (bP0_OC << Port);
           Pn_DIR &= ~(1<<PinNum);
           Pn_PU |= 1<<PinNum;
           break;
        case 6:
           PORT_CFG |= (bP0_OC << Port);
           Pn_DIR |= 1<<PinNum;
           Pn_PU |= 1<<PinNum;
           break;
        default:
           break;
    }
    if(Port == 0){
       P0_DIR = Pn_DIR;
       P0_PU = Pn_PU;
    }
    if(Port == 1){
       P1_DIR = Pn_DIR;
       P1_PU = Pn_PU;
    }
    if(Port == 2){
       P2_DIR = Pn_DIR;
       P2_PU = Pn_PU;
    }
    if(Port == 3){
       P3_DIR = Pn_DIR;
       P3_PU = Pn_PU;
    }

}

/*********************************************************************
* Function Name  : CH559P4Mode()
* Input          : None
* Output         : None
* Return         : None
*********************************************************************/
void CH559P4Mode( )
{
   P4_DIR |= 0xff;
   P4_PU |= 0xff;
   P4_CFG |= bP4_DRV;
}

/*********************************************************************
* Function Name  : CH559GPIOInterruptInit()
*********************************************************************/
void CH559GPIOInterruptInit()
{
   GPIO_IE &= ~bIE_IO_EDGE;
   GPIO_IE |= bIE_RXD1_LO;
}

/*********************************************************************
* Function Name  : GPIOInterrupt(void)
* Input          : None
* Output         : None
* Return         : None
*********************************************************************/
/*
static inline void delayms(uint16_t ms) {
    uint16_t ki;
    for (ki = 0; ki < (ms*12000UL); ki++){}
        __asm__("nop");
}

static inline void delays(uint16_t sec) {
    uint16_t ki;
    for (ki = 0; ki < (sec*10000UL); ki++){}
        delayms(1);
}
*/

/*
static inline void delay(uint32_t ms)
{
```

```
236     uint32_t i;
237     for (i = 0; i < (ms * 28); i++){}
238     //__asm__("nop");
239 }*/
240 void notificar_con_led(uint8_t puerto, uint8_t pin)
241 {
242      CH559GPIOModeSelt(puerto, 6, pin);
243      delay();
244      CH559GPIOModeSelt(puerto,0,pin);
245      delay();
246 }
247
248
249 uint8_t subir_led(uint8_t actual_led)
250 {
251
252
253     CH559GPIOModeSelt(1,0,actual_led);
254
255     if (actual_led<7)
256        {
257        actual_led=actual_led+1;
258        }
259
260     CH559GPIOModeSelt(1, 6, actual_led);
261     delay();
262     return actual_led;
263 }
264
265 uint8_t bajar_led(uint8_t actual_led)
266 {
267
268     CH559GPIOModeSelt(1,0,actual_led);
269
270     if (actual_led>4)
271        {
272        actual_led=actual_led-1;
273        }
274
275     CH559GPIOModeSelt(1, 6, actual_led);
276     delay();
277     return actual_led;
278 }
279
280 main( )
281 {
282
283
284      unsigned short ADCDat;
285      uint8_t i = 0;
286      //CfgFsys( );
287      delay();
288
289      mInitSTDIO( );
290
291
292      InitADCManual();
293        ADCDat=ADC_FIFO;
294       //printf("FIFOCnt:%02X  InvalidADC_DATA:%04X  \n",(uint16_t)(ADC_STAT & 3),(uint16_t)ADCDat);
295       CH559GPIOModeSelt(1,0,4);
296       CH559GPIOModeSelt(1,0,5);
297       CH559GPIOModeSelt(1,0,6);
298       CH559GPIOModeSelt(1,0,7);
299       uint8_t actual_led=4;
300       CH559GPIOModeSelt(1,6,actual_led);
301
302 #ifdef _INT_
303      InitADCInterrupt();
304      EA = 1;
305
306      ADC_CTRL |= bADC_SAMPLE;
307      delay();
308      ADC_CTRL &= ~bADC_SAMPLE;
309 #endif
310      while(1)
311      {
312
313            ADC_SETUP|=bADC_POWER_EN;
314           //for(i=0;i<7;i++){
315             ADCDat = ADCChanelChange(ADCChannel[2]);
316              //printf("FIFOCnt:%02X  ADC_DATA%X:%04X  \n",(uint16_t)(ADC_STAT&3),(uint16_t)i,(uint16_t)
                  ADCDat);
317             if (ADCDat >=   1000 )
318               {
319
320               actual_led=subir_led(actual_led);
321
322
323               }
324
325             if (ADCDat <=   20 )
326               {
327
328               actual_led=bajar_led(actual_led);
329
330
331               }
332
333
334             //delay();
335
336
337
338           //}
339            ADC_SETUP&=~bADC_POWER_EN;
```

```
340
341
342
343         }
344 }
```

Listado 3.5 – *Joystick control with CH559*



Video 3.1 – *Joystick example*

Apart from the 3 pins that the microphone would need for the rotary encoder, you also have to add the two USB connections (positive and negative) where the microphone is reprogrammed. The total number of pins required for this function is 5.

3.2.1.4 ADC reading to be able to read the consumption by means of the configuration offered by the switching transistor (BTS6143D)

For the voltage reading with Analog to Digital Converter (ADC), we have used the Printed Circuit Board (PCB) control itself containing the connectors, which is where the BTS6143D component is located, with the necessary configuration to read the consumption that will be explained in section 3.2.3. The aim is to be able to measure the voltage, which will be reflected by the Light-Emitting Diode (LED)s on the EValuation Test (EVT) board, as the Liquid-Crystal Display (LCD) has not been made to work.

Although only one Analog to Digital Converter (ADC) channel has been used in the example, 8 channels would be required for the product and only the *CH559* has that number of channels so there is no need to add any configuration such as channel switching.

```
1
2
3  #include "include\CH559_2.H"
4  #include "include\DEBUG.H"
5  //#include <stdint.h>
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <math.h>
9
10 /********************************* (C) COPYRIGHT *******************************
11 * File Name          : ADCManual.C
12 * Author             : WCH
13 * Version            : V1.3
14 * Date               : 2016/6/24
```

```
15 *****************************************************************************/
16
17
18 uint8_t ADCChannel[8] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
19
20  #define _INT_  1
21
22 //#pragma  NOAREGS
23
24
25 /*****************************************************************************
26 * Function Name  : InitADCManual()
27 * Input          : None
28 * Output         : None
29 * Return         : None
30 *****************************************************************************/
31 static inline void delay() {
32     uint32_t ki;
33     for (ki = 0; ki < (700); ki++){}
34         __asm__("nop");
35 }
36
37 void InitADCManual()
38 {
39     P1_IE = 0x00;
40     ADC_SETUP |= bADC_POWER_EN;
41     ADC_CK_SE = 0x02;
42     ADC_CTRL &= ~MASK_ADC_CYCLE;
43     ADC_CTRL &= ~(bADC_CHANN_MOD1 | bADC_CHANN_MOD0);
44     ADC_CHANN = ADCChannel[0];
45     //     ADC_EX_SW |= bADC_RESOLUTION;
46     ADC_EX_SW &= ~bADC_RESOLUTION;
47 }
48
49
50 /*****************************************************************************
51 * Function Name  : ADCChanelChange()
52 * Input          : uint8_t Chanel
53 * Output         : None
54 * Return         : uint16_t ADCValue
55 *****************************************************************************/
56 uint16_t ADCChanelChange(uint8_t Chanel)
57 {
58   unsigned short ADCValue = 0;
59    ADC_CHANN = Chanel;
60 //     ADC_EX_SW |= bADC_RESOLUTION;
61     delay();
62    ADC_CTRL |= bADC_SAMPLE;
63     delay();
64    ADC_CTRL &= ~bADC_SAMPLE;
65    while((ADC_STAT & bADC_IF_ACT) == 0);
66    ADC_STAT |= bADC_IF_ACT;
67    ADCValue = ADC_FIFO;
68   return ADCValue;
69 }
70
71 #ifdef _INT_
72 /*****************************************************************************
73 * Function Name  : InitADCInterrupt()
74 * Input          : None
75 * Output         : None
76 * Return         : None
77 *****************************************************************************/
78 void InitADCInterrupt()
79 {
80     ADC_SETUP |= bADC_IE_FIFO_OV;
81 //    ADC_SETUP |= bADC_IE_AIN7_LOW;
82     ADC_SETUP |= bADC_IE_ACT;
83     IE_ADC = 1;
84 }
85
86 /*****************************************************************************
87 * Function Name  : ADCInterrupt(void)
88 *****************************************************************************/
89 void    ADCInterrupt( void ) __interrupt INT_NO_ADC __using 1
90 {
91     uint16_t ADCValue = 0;
92     if(ADC_STAT & bADC_IF_ACT)
93     {
94         ADC_STAT |= bADC_IF_ACT;
95     }
96     ADCValue = ADC_FIFO;
97      //printf("FIFOCnt:%02X   ADC_DATA:%04X   \n",(uint16_t)(ADC_STAT&3),(uint16_t)ADC_FIFO);
98
99 #if 1
100     ADC_CTRL |= bADC_SAMPLE;
101     delay();
102     ADC_CTRL &= ~bADC_SAMPLE;
103 #endif
104 }
105 #endif
106
107 //#pragma  NOAREGS
108
109 /*****************************************************************************
110 * Function Name  : CH559GPIODrivCap(UINT8 Port,UINT8 Cap)
111 *****************************************************************************/
112 void CH559GPIODrivCap(UINT8 Port,UINT8 Cap)
113 {
114    if(Port >= 4){
115
116    }
117    if(Cap == 0){
118     PORT_CFG &= ~(bP0_DRV << Port);
119   }
```

```
120     else{
121       PORT_CFG |= (bP0_DRV << Port);
122     }
123 
124 }
125 
126 /*********************************************************************
127 * Function Name  : CH559GPIOModeSelt(UINT8 Port,UINT8 Mode,UINT8 PinNum)
128 *********************************************************************/
129 void CH559GPIOModeSelt(uint8_t Port,uint8_t Mode,uint8_t PinNum)
130 {
131   uint8_t Pn_DIR,Pn_PU;
132 
133   switch (Mode){
134         case 0:
135            PORT_CFG &= ~(bP0_OC << Port);
136            Pn_DIR &= ~(1<<PinNum);
137            Pn_PU &= ~(1<<PinNum);
138            break;
139         case 1:                                                          //
140            PORT_CFG &= ~(bP0_OC << Port);
141            Pn_DIR &= ~(1<<PinNum);
142            Pn_PU |= 1<<PinNum;
143            break;
144         case 2:                                                          //
145            PORT_CFG &= ~(bP0_OC << Port);
146            Pn_DIR |= ~(1<<PinNum);
147            break;
148         case 3:                                                          //
149            PORT_CFG |= (bP0_OC << Port);
150            Pn_DIR &= ~(1<<PinNum);
151            Pn_PU &= ~(1<<PinNum);
152            break;
153         case 4:                                                          //
154            PORT_CFG |= (bP0_OC << Port);
155            Pn_DIR |= 1<<PinNum;
156            Pn_PU &= ~(1<<PinNum);
157            break;
158         case 5:                                                          //
159            PORT_CFG |= (bP0_OC << Port);
160            Pn_DIR &= ~(1<<PinNum);
161            Pn_PU |= 1<<PinNum;
162            break;
163         case 6:                                                          //
164            PORT_CFG |= (bP0_OC << Port);
165            Pn_DIR |= 1<<PinNum;
166            Pn_PU |= 1<<PinNum;
167            break;
168         default:
169            break;
170    }
171    if(Port == 0){
172       P0_DIR = Pn_DIR;
173       P0_PU = Pn_PU;
174    }
175    if(Port == 1){
176       P1_DIR = Pn_DIR;
177       P1_PU = Pn_PU;
178    }
179    if(Port == 2){
180       P2_DIR = Pn_DIR;
181       P2_PU = Pn_PU;
182    }
183    if(Port == 3){
184       P3_DIR = Pn_DIR;
185       P3_PU = Pn_PU;
186    }
187 
188 }
189 
190 /*********************************************************************
191 * Function Name  : CH559P4Mode()
192 * Input          : None
193 * Output         : None
194 * Return         : None
195 *********************************************************************/
196 void CH559P4Mode( )
197 {
198    P4_DIR |= 0xff;                                                     //
199    P4_PU |= 0xff;                                                      //
200    P4_CFG |= bP4_DRV;                                                  //
201 }
202 
203 /*********************************************************************
204 * Function Name  : CH559GPIOInterruptInit()
205 * Input          : None
206 * Output         : None
207 * Return         : None
208 *********************************************************************/
209 void CH559GPIOInterruptInit()
210 {
211    GPIO_IE &= ~bIE_IO_EDGE;                                            //
212    GPIO_IE |= bIE_RXD1_LO;                                             //
213 }
214 
215 /*********************************************************************
216 * Function Name  : GPIOInterrupt(void)
217 * Input          : None
218 * Output         : None
219 * Return         : None
220 *********************************************************************/
221 /*
222 static inline void delayms(uint16_t ms) {
223     uint16_t ki;
224     for (ki = 0; ki < (ms*12000UL); ki++){}
```

```
225         __asm__("nop");
226 }
227
228 static inline void delays(uint16_t sec) {
229     uint16_t ki;
230     for (ki = 0; ki < (sec*10000UL); ki++){}
231         delayms(1);
232 }
233 */
234
235 /*
236 static inline void delay(uint32_t ms)
237 {
238    uint32_t i;
239    for (i = 0; i < (ms * 28); i++){}
240    //__asm__("nop");
241 }*/
242 void notificar_con_led(uint8_t puerto, uint8_t pin)
243 {
244     CH559GPIOModeSelt(puerto, 6, pin);
245     delay();
246     CH559GPIOModeSelt(puerto,0,pin);
247     delay();
248 }
249
250 main( )
251 {
252
253
254     unsigned short ADCDat;
255     uint8_t i = 0;
256     //CfgFsys( );                                                    //
257     delay();                                                  //
258
259     mInitSTDIO( );                                                 //
260
261
262     InitADCManual();                                               //
263       ADCDat=ADC_FIFO;                                             //
264      //printf("FIFOCnt:%02X  InvalidADC_DATA:%04X  \n",(uint16_t)(ADC_STAT & 3),(uint16_t)ADCDat);
265
266
267
268 #ifdef _INT_
269     InitADCInterrupt();                                            //
270     EA = 1;                                                        //
271
272     ADC_CTRL |= bADC_SAMPLE;                                       //
273     delay();
274     ADC_CTRL &= ~bADC_SAMPLE;
275 #endif
276     while(1)                                                       //
277     {
278
279           ADC_SETUP|=bADC_POWER_EN;                                  //
280         //for(i=0;i<7;i++){
281           ADCDat = ADCChanelChange(ADCChannel[0]);                 //
282            //printf("FIFOCnt:%02X  ADC_DATA%X:%04X  \n",(uint16_t)(ADC_STAT&3),(uint16_t)i,(uint16_t)
                ADCDat);
283               if (ADCDat >=   404 )
284      {
285
286         notificar_con_led(1,7);
287
288
289      }
290      if (ADCDat >=   341 )
291      {
292
293
294          notificar_con_led(1,6);
295
296      }
297      if (ADCDat >=   310 )
298      {
299
300
301          notificar_con_led(1,5);
302
303      }
304      if (ADCDat >=   280 )
305      {
306
307
308          notificar_con_led(1,4);
309
310      }
311         //}
312          ADC_SETUP&=~bADC_POWER_EN;
313
314
315
316     }
317 }
```

Listado 3.6 – *CH559 ADC reading*

3.2.1.5 Creation of a signal generator

This function is the last in complexity and need for other software components, as it requires a menu to be navigable and visible, and therefore occupies this place.

This function could be performed by all micro controllers compared as it would require 8 General Purpose Input/Output (GPIO) pins, which they all have.

Due to various setbacks, there is no code for this function but it is not necessary to have the code to analyse whether the micro controllers can perform it, since as mentioned above, at the physical level 8 pins are needed for it.

The signal generator function is part of the program software as can be seen in the flowchart 2.16 when entering a Pulse-Width Modulation (PWM) channel you can choose values for the parameters frequency, duty and phase in Pulse-Width Modulation (PWM) mode.

3.2.2 Software implementation of the product

Once the separate functions of the product are known and the hardware requirements for choosing a micro controller are known, a count is made of the total number of pins needed.

Function	Number of pins
3.2Function 1	$8GPIO$
3.2Function 2	$7(6GPIO + 1PWM)$
3.2Function 3	$6(3GPIO + 1PROGpin + 2USB(+/-))$
3.2Function 4	$8ADC$
Total pins required	29

Table 3.4 – *Total pins required for micro controller*

Function 5 has no hardware requirement as the necessary pins have been taken into account in function 1.

Once the hardware requirements for the choice of the micro controller are known, it is considered that the only valid micro controller is the *CH559L*, so this has been chosen for the design.

3.2.3 Basic PCB for software needs

Having knowledge of the previous product hardware, the necessary components will be added to comply with the product software implementation and to accommodate the necessary configurations for proper operation.

A study will be made of the Printed Circuit Board (PCB)s, the first one containing the micro controller and the second one containing the connectors, which were named PCB1 and PCB2 [2.3.1.1] and will be called MicrocontrollerPCB and ConnectorsPCB respectively.

3.2.3.1 MicrocontrollerPCB analysis

The analysis shall be done in descending order of voltage, starting at +12 V (supported from the other Printed Circuit Board (PCB)) and shall take this order:

12 V to 5 V voltage regulator with its respective components for implementation, 5 V powered components

(Liquid-Crystal Display (LCD), Universal Serial Bus (USB), 5 V to 3.3 V voltage regulator) and 3.3 V powered components (rotary encoder), ending with the pin header to connect to the other Printed Circuit Board (PCB). Regarding the Light-Emitting Diode (LED)s, which previously had a protection resistor, in this case they are not needed as the CH559 has an output voltage of 3.3 V on its pins.

The micro controller will not be discussed here as it has been discussed in the previous subsections [3.2.2].

3.2.3.2 12 V to 5 V voltage regulator

To reduce the voltage from 12 V to 5 V, two components will be compared: LM7805 and MCP16301. The amount of components needed in their configuration, the price and the availability in the GRANASAT laboratory will be analysed [3].

Application Information (continued)

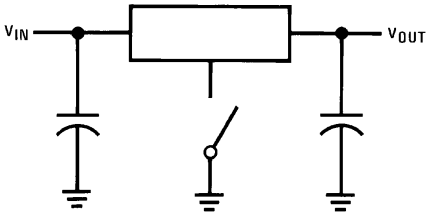


Figure 16. Regulator Floating Ground

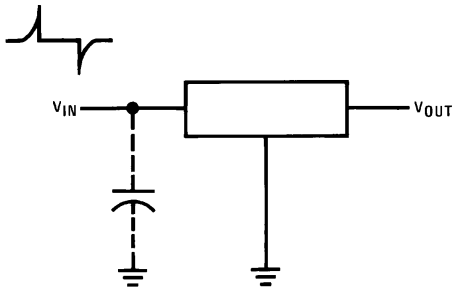


Figure 17. Transients

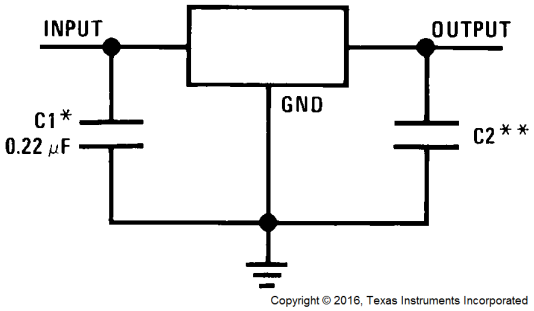
When a value for $\theta_{(H-A)}$ is found, a heat sink must be selected that has *a value that is less than or equal to this number.*

$\theta_{(H-A)}$ is specified numerically by the heat sink manufacturer in this catalog or shown in a curve that plots temperature rise vs power dissipation for the heat sink.

8.2 Typical Applications

8.2.1 Fixed Output Voltage Regulator

The LM340x and LM7805 Family devices are primarily designed to provide fixed output voltage regulation. The simplest implementation of LM340x and LM7805 Family is shown in Figure 18.



*Required if the regulator is located far from the power supply filter.
**Although no output capacitor is needed for stability, it does help transient response. (If needed, use 0.1-µF, ceramic disc).

Figure 18. Fixed Output Voltage Regulator

8.2.1.1 Design Requirements

The device component count is very minimal. Although not required, TI recommends employing bypass capacitors at the output for optimum stability and transient response. These capacitors must be placed as close as possible to the regulator. If the device is located more than 6 inches from the power supply filter, it is required to employ input capacitor.

Typical Applications (continued)

8.2.1.2 Detailed Design Procedure

The output voltage is set based on the device variant. LM340x and LM7805 Family are available in 5-V, 12-V and 15-V regulator options.

8.2.1.3 Application Curve

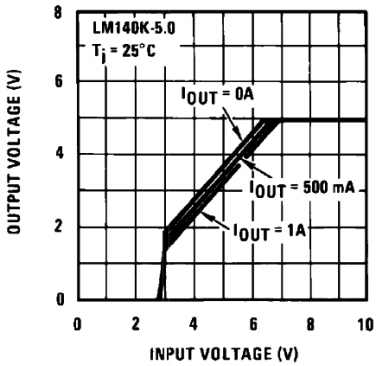
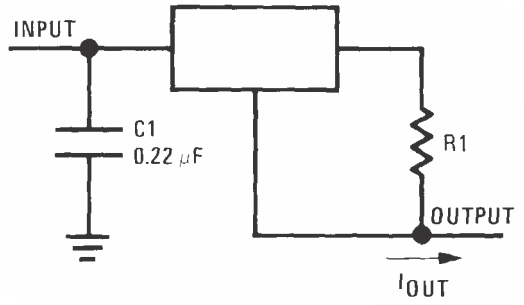


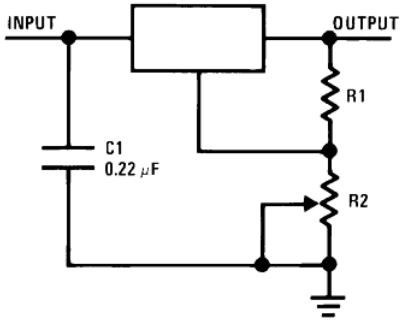
Figure 19. V_{OUT} vs V_{IN}, V_{OUT} = 5 V

8.3 System Examples



I_{OUT} = V2–3 / R1 + I_Q
ΔI_Q = 1.3 mA over line and load changes.

Figure 20. Current Regulator



V_{OUT} = 5 V + (5 V/R1 + I_Q) R2 5 V/R1 > 3 I_Q,
load regulation (L_r) ≈ [(R1 + R2)/R1] (L_r of LM340-5).

Figure 21. Adjustable Output Regulator

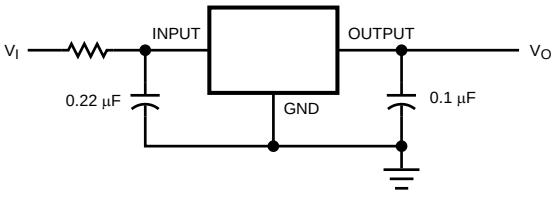


Figure 22. High Input Voltage Circuit With Series Resistor

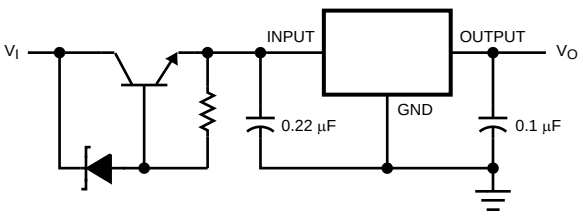
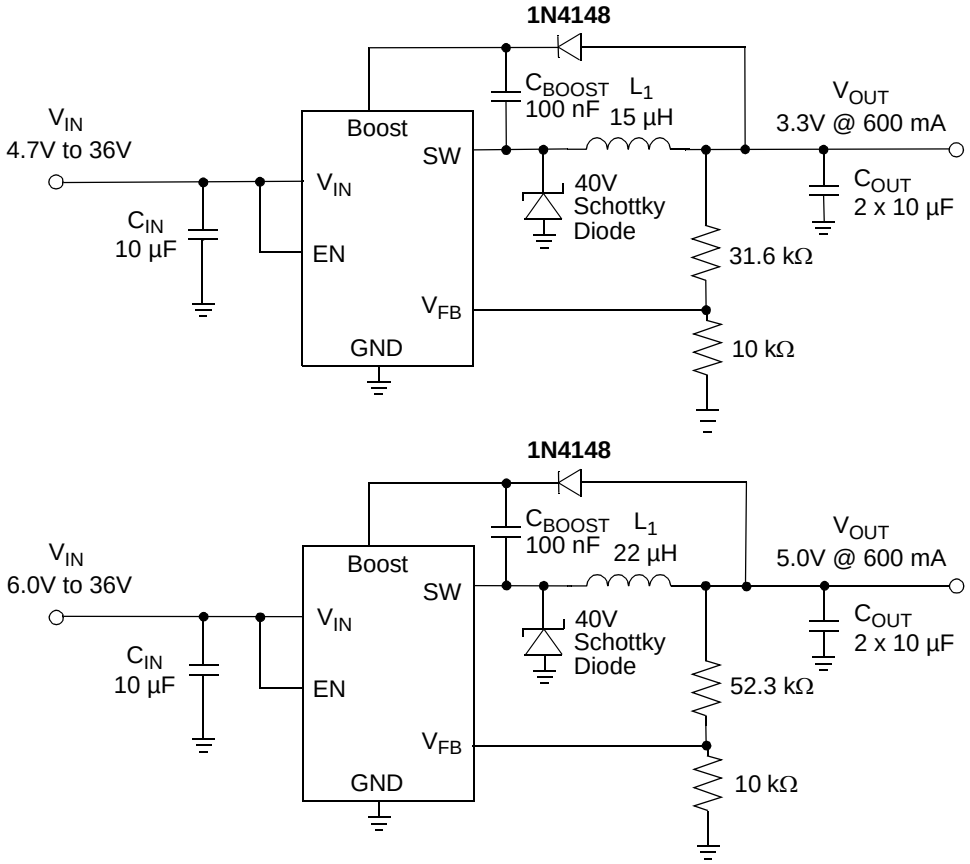
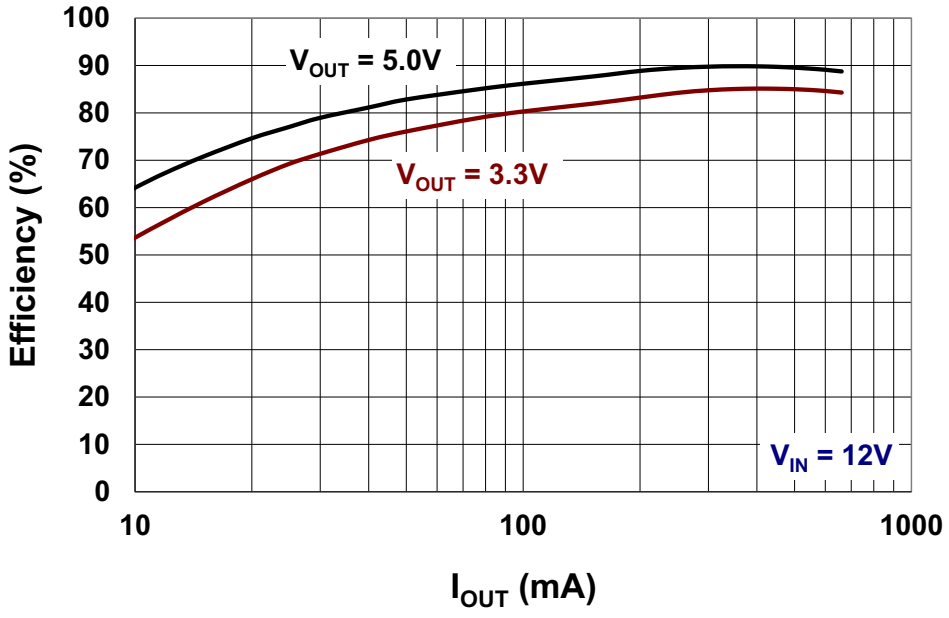


Figure 23. High Input Voltage Circuit implementation With Transistor

Typical Applications





If the LM7805 is chosen, two capacitors would be needed in addition to the regulator itself.

In the case of the MCP16301, it requires more components than the LM7805, which may increase the price.

Table 3.5 – *12 to 5 V Voltaje regulator*

	LM7805	MCP16301
Number of components needed	3	9
Price of main component (€)	1.42	1.29
Availability	YES	NO
Price (€) (with all components)	2.01 (1.42 + 0.24 + 0.35)	3.99 (1.29 + 0.24 + 4x0.35 + 0.70 + 2x0.18)

After comparing the two components, the LM7805 is chosen as the 12 V to 5 V voltage regulator.

3.2.3.3 5 V to 3.3 V voltage regulator

This time the LM1117 and again the MCP16301 will be compared [4].

9.3 System Examples

Several circuits can be realized with the LM1117. The circuit diagrams in this section demonstrate multiple system examples that can be utilized in many applications.

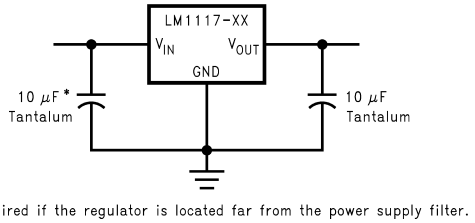


Figure 9-3. Fixed Output Regulator

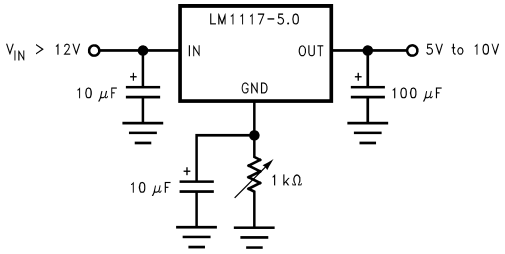


Figure 9-4. Adjusting Output of Fixed Regulators

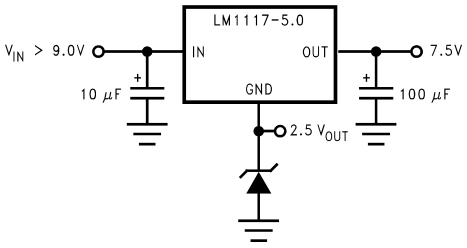


Figure 9-5. Regulator With Reference

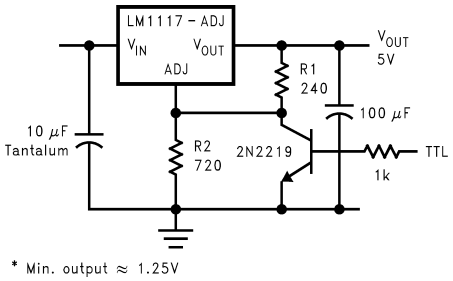


Figure 9-6. 5-V Logic Regulator With Electronic Shutdown*

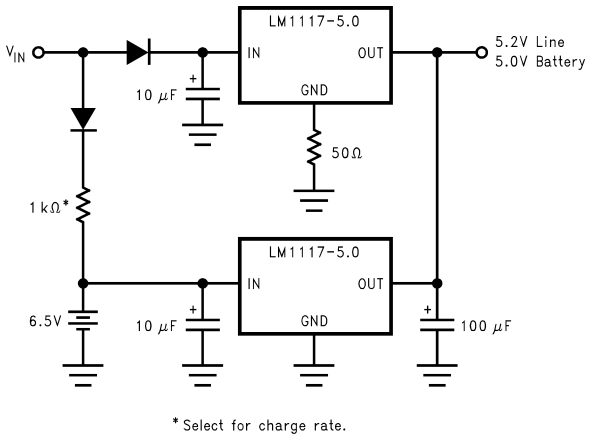


Figure 9-7. Battery Backed-Up Regulated Supply

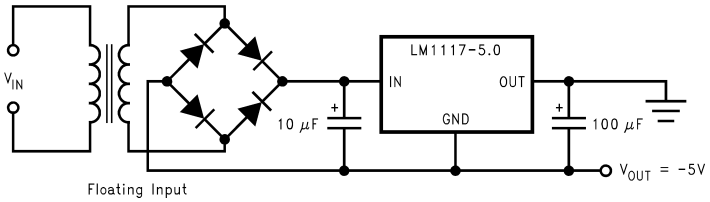


Figure 9-8. Low Dropout Negative Supply

Table 3.6 – *5 to 3.3 V Voltage regulator*

	LM1117	MCP16301
Number of components needed	3	9
Price of main component (€)	1.06	1.29
Availability	NO	NO
Price (€) (with all components)	2.86 $(1.06 + 2x0.85 + 0.12)$	3.99 $(1.29 + 0.24 + 4x0.35 + 0.70 + 2x0.18)$

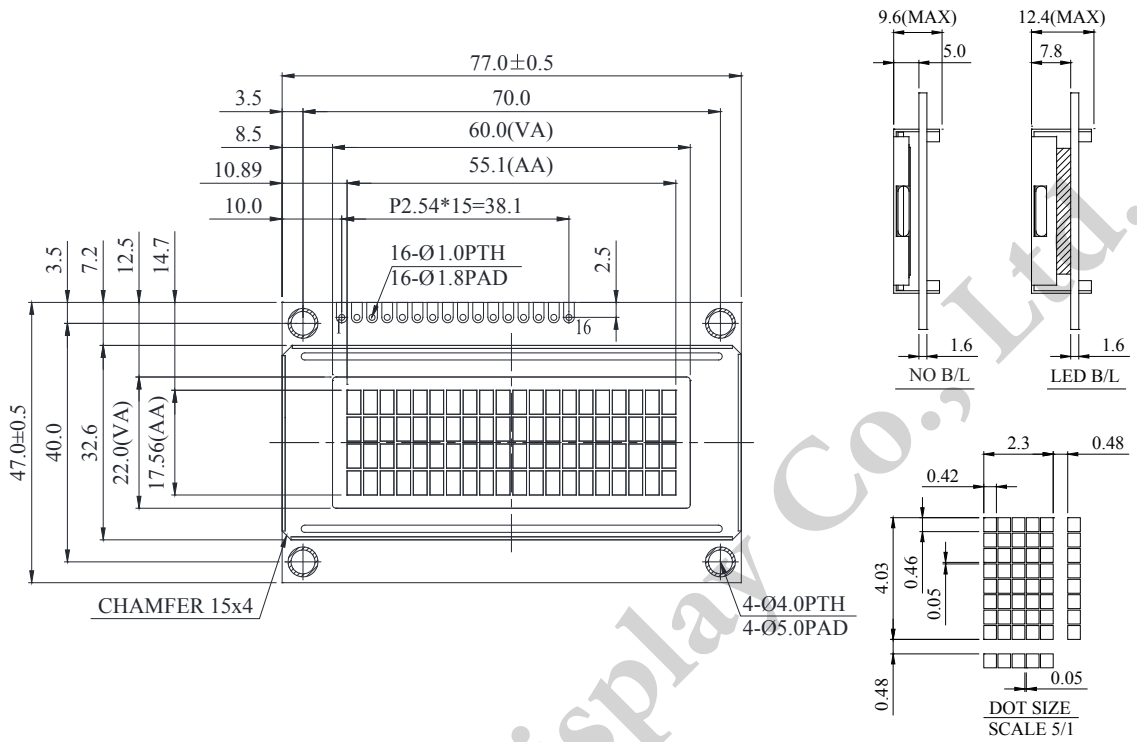
After analysis, the LM1117 was chosen as the 5 V to 3.3 V voltage regulator because of its price.

3.2.3.4 Elements to be maintained from the previous design

There are elements that will not be analysed due to the condition that they are the same as in the previous design, such as the Liquid-Crystal Display (LCD), the rotary encoder, Universal Serial Bus (USB) type B.

Liquid-Crystal Display (LCD) and rotary encoder data sheets will be presented [8] and :

WH2004D Character 20x4



Feature

1.5x8 dots includes cursor
2.Bulit-in controller (ST7066 or Equivalent)
3.5V power supply (Also available for 3V)
4.N.V, optional for 3V power supply
5.1/16 duty cycle
6.LED can be driven by PIN1, PIN2, PIN15, PIN16 or A and K
7.Interface : 6800, option SPI/I2C (RW1063 IC)

Pin No.	Symbol	Description
1	V_{SS}	Ground
2	V_{DD}	Power supply for logic
3	V_O	Contrast Adjustment
4	RS	Data/ Instruction select signal
5	R/W	Read/Write select signal
6	E	Enable signal
7	DB0	Data bus line
8	DB1	Data bus line
9	DB2	Data bus line
10	DB3	Data bus line
11	DB4	Data bus line
12	DB5	Data bus line
13	DB6	Data bus line
14	DB7	Data bus line
15	A	Power supply for B/L +
16	K	Power supply for B/L -

Mechanical Data

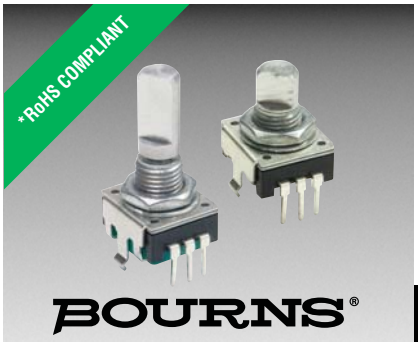
Item	Standard Value	Unit
Module Dimension	77.0 x 47.0	mm
Viewing Area	60.0 x 22.0	mm
Mounting Hole	70.0 x 40.0	mm
Character Size	2.30 x 4.03	mm

Electrical Characteristics

Item	Symbol	Standard Value typ.	Unit
Input Voltage	VDD	3/5	V
Recommended LCD Driving Voltage for Normal Temp. Version module @25°C	VDD-VO	4.50	V

Display Character Address Code

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	---	---	20
DD RAM Address	00	01														13
DD RAM Address	40	41														53
DD RAM Address	14	15														27
DD RAM Address	54	55														67



Features

- Push switch option
- Compact, rugged design
- High reliability
- Metal bushing/shaft

BOURNS® PRO AUDIO

PEC11R Series - 12 mm Incremental Encoder

Electrical Characteristics

Output	2-bit quadrature code
Contact Rating	10 mA @ 5 VDC
Insulation Resistance	100 megohms @ 250 VDC
Dielectric Withstanding Voltage	
Sea Level	300 VAC minimum
Electrical Travel	Continuous
Contact Bounce (15 RPM)	2.0 ms maximum**
RPM (Operating)	60 maximum**

Environmental Characteristics

Operating Temperature Range	-30 °C to +70 °C (-22 °F to +158 °F)
Storage Temperature Range	-40 °C to +85 °C (-40 °F to +185 °F)
Humidity	MIL-STD-202, Method 103B, Condition B
Vibration	10~55~10 Hz / 1 min. / Amplitude 1.5 mm
Shock	100 G
Rotational Life	30,000 cycles minimum
Switch Life	20,000 cycles minimum
IP Rating	IP 40

Mechanical Characteristics

Mechanical Angle	360 ° continuous
Torque	
Running	50 to 200 gf.cm (0.68 to 2.7 oz.-in.)
Mounting	10.2 kgf.cm (8.83 lb.-in.) maximum
Shaft Side Load (Static)	2.04 kgf (4.5 lbs.) minimum
Weight	5 gm (0.17 oz.) maximum
Terminals	Printed circuit board terminals
Soldering Condition	
Wave Soldering	Sn95.5/Ag2.8/Cu0.7 solder with no-clean flux: 260 °C max. for 3 ±1 sec.
Hand Soldering	Not recommended
Hardware	One flat washer and one mounting nut supplied with each encoder

Switch Characteristics

Switch Type	Contact Push ON Momentary SPST
Power Rating (Resistive Load)	10 mA at 5 V DC
Switch Travel	0.5 ± 0.3 mm
Switch Actuation Force	610 ± 306 gf (8.47 ± 4.24 oz.-in.)
Contact Resistance	100 milliohms @ 5 VDC

How To Order

PEC11R - 4 0 20 F - S 0012

Model

Terminal Configuration
- 4 = PC Pin Horizontal/Rear Facing

Detent Option
- 0 = No Detents (12, 18, 24 pulses)
- 1 = 18 Detents (18 pulses)
- 2 = 24 Detents (12, 24 pulses)
- 3 = 12 Detents (12, 24 pulses)

Standard Shaft Length
- 15 = 15.0 mm
- 20 = 20.0 mm
- 25 = 25.0 mm
- 30 = 30.0 mm

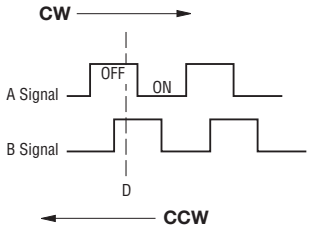
Shaft Style
- F = Metal Flatted Shaft
- K = Metal Knurled Shaft[1]

Switch Configuration
- S = Push Momentary Switch
- N = No Switch

Resolution
- 0012 = 12 Pulses per 360 ° Rotation
- 0018 = 18 Pulses per 360 ° Rotation
- 0024 = 24 Pulses per 360 ° Rotation

[1] Metal knurled shaft without switch is available in 15, 20 and 30 mm shaft lengths.
Metal knurled shaft with push momentary switch is available in 15 and 20 mm shaft lengths.

Quadrature Output Table



*RoHS Directive 2002/95/EC Jan. 27, 2003 including annex and RoHS Recast 2011/65/EU June 8, 2011.
**Devices are tested using standard noise reduction filters. For optimum performance, designers should use noise reduction filters in their circuits.
Specifications are subject to change without notice.
The device characteristics and parameters in this data sheet can and do vary in different applications and actual device performance may vary over time.
Users should verify actual device performance in their specific applications.

3.2.3.5 Connector for MicrocontrollerPCB

In this subsection it should be noted that there was already a 10 in 2 header pin where the 12 V, Ground (GND) and 8-channel connections were established to activate the transistors of the BTS6143D. Having explained this, it is now time to talk about the 8 channels needed to read the Analog to Digital Converter (ADC), so it is necessary to study whether to use two 10 in 2 pin header components or a single 20 in 2 pin header component.

The price of two 2x5 components is 3.90€, while the price of one 2x10 component is 0.86€. This makes that finally a 2x10 pin header (20 in 2) is chosen for the connection to this board.

3.2.3.6 ConnectorsPCB analysis

On this board, the modifications that have been made are minimal, just distribute the components to be able to connect the channels for the voltage reading of the Analog to Digital Converter (ADC), accommodate the Light-Emitting Diode (LED)s for visual verification to the voltage of the *CH559* pins and add a resistor to each BTS6143D to adapt it to its current sensor configuration (which is where the voltage will be measured with the Analog to Digital Converter (ADC) to read the consumption). For cost and space reasons, it has been decided that the connector, as on the MicrocontrollerPCB board, will be a 2x10 pin header.

Regarding the Light-Emitting Diode (LED)s, which previously had a protection resistor, in this case they are not needed as the CH559 has an output voltage of 3.3 V on its pins.

The only analysis that remains to be done on this board, and therefore in the chapter, is the analysis of the resistor needed to adapt the voltage that the ADC will read, which will be placed at the output of the BTS6143D (pin 4).

3.2.3.7 Voltage measurement to read consumption with BTS6143

To carry out this analysis there are several parameters that we had to choose, such as the maximum current at the output of the BTS6143D (6 A) and by connecting several automotive lights 5.6 A was achieved. The following document explains how the analysis was carried out and tested to ensure the correct operation of this configuration.

Voltage measurement to read consumption with BTS6143

August 14, 2022

1 Voltage measurement to read consumption with BTS6143

1.1 Introduction

- In order to read consumption of drivers, BTS6143 is neccessary because it has diagnostic feedback in Is (pin 4). CH559 ADC works in 3.3V and his ADC pins are (1,2,43,44,45,46,47,48).

[1]:
```
from IPython.display import IFrame
IFrame("IMG/datasheet_BTS.pdf", width=900, height=600)
```

[1]:
```
<IPython.lib.display.IFrame at 0x1b632b9e700>
```

1.2 Theoretical calculations

BTS pins description is in page 2 and current sense output in page 9.

Infineon technologies

Data sheet BTS 6143 D

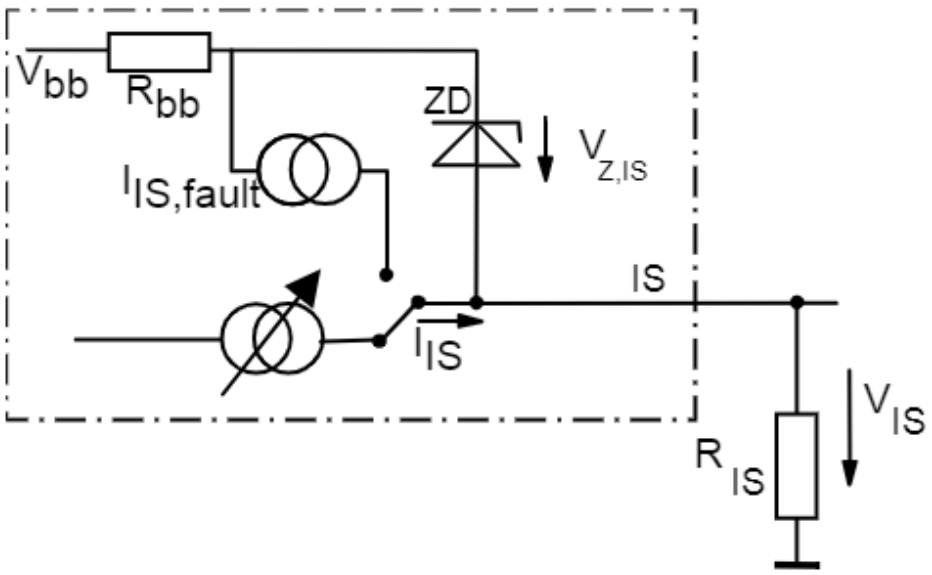
Pin	Symbol		Function
1	OUT	O	**Output;** output to the load; pin 1 and 5 must be externally shorted* .
2	IN	I	**Input;** activates the power switch if shorted to ground.
Tab/(3)	Vbb	+	**Supply Voltage;** positive power supply voltage; tab and pin3 are internally shorted.
4	IS	S	**Sense Output;** Diagnostic feedback; provides at normal operation a sense current proportional to the load current; in case of overload, overtemperature and/or short circuit a defined current is provided (see Truth Table on page 8)
5	OUT	O	**Output;** output to the load; pin 1 and 5 must be externally shorted* .

*) Not shorting all outputs will considerably increase the on-state resistance, reduce the peak current capability and decrease the current sense accuracy

BTS pins

Current sense output

Normal operation



$V_{Z,IS}$ = 67 V (typ.), R_{IS} = 1 kΩ nominal (or 1 kΩ /n, if n devices are connected in parallel). $I_S = I_L/k_{ilis}$ can be only driven by the internal circuit as long as V_{out} - V_{IS} > 5V. Therefore R_{IS} should be less than

$$\frac{V_{bb} - 5V}{7.5mA}.$$

Note: *For large values of R_{IS} the voltage V_{IS} can reach almost V_{bb}. See also overvoltage protection.*
If you don't use the current sense output in your application, you can leave it open.

BTS output sense current

This equation in function of Vbb (12V then) is neccessary to got Ris:

$$R_{IS} < \frac{V_{bb} - 5V}{I_{S_{max,lim}}} = \frac{12V - 5V}{7.5mA} = 933.33\Omega \quad (1)$$

$I_{S_{max,lim}}$=7.5 mA because $I_{L_{max,lim}}$=75A and K = 10000.

Diagnostic Characteristics					
Current sense ratio, static on-condition $k_{ILIS} = I_L : I_{IS}$, $I_{IS} < I_{IS,lim}$ 16), $V_{IS} < V_{OUT}$ - 5V, $V_{bIN} > 4.5$ V	k_{ILIS}	--	10 000	--	
IL = 30A, Tj = -40°C:		8300	10000	11000	
Tj = +25°C:		8300	9700	10600	
Tj = +150°C:		8300	9300	10000	
IL = 7.5A, Tj = -40°C:		7500	10000	11400	
Tj = +25°C:		8000	9700	10800	
Tj = +150°C:		8200	9300	10200	
IL = 2.5A, Tj = -40°C:		7100	10000	13400	
Tj = +25°C:		7700	9700	12500	
Tj = +150°C:		8000	9300	12000	
IL = 0.5A, Tj = -40°C:		5000	12000	21000	
Tj = +25°C:		5500	12000	19000	
Tj = +150°C:		6000	12000	18000	
I_{IN} = 0 (e.g. during deenergizing of inductive loads):		--	0	--	
Sense current under fault conditions 17) V_{ON}>1V, typ T_j = -40...+150°C:	$I_{IS,fault}$	4.0	5.2	7.5	mA
Sense saturation current V_{ON}<1V, typ T_j = -40...+150°C:	$I_{IS,lim}$	4.0	6.0	7.5	mA

BTS current sense rate

With:

$$V_{ON} < 1V \tag{2}$$

K_{illis} is rate I_L : I_S so if K = 10000 and we need $I_{L_{TOP}}$= 6A, $I_{S_{TOP}}$=600uA. {TOP} referencing max current for this design. CH559 ADC max voltage is 3.3V. Trhought last image R_{IS} is calculated:

$$R_{IS} = \frac{3.3V}{600uA} = 5.5k\Omega$$

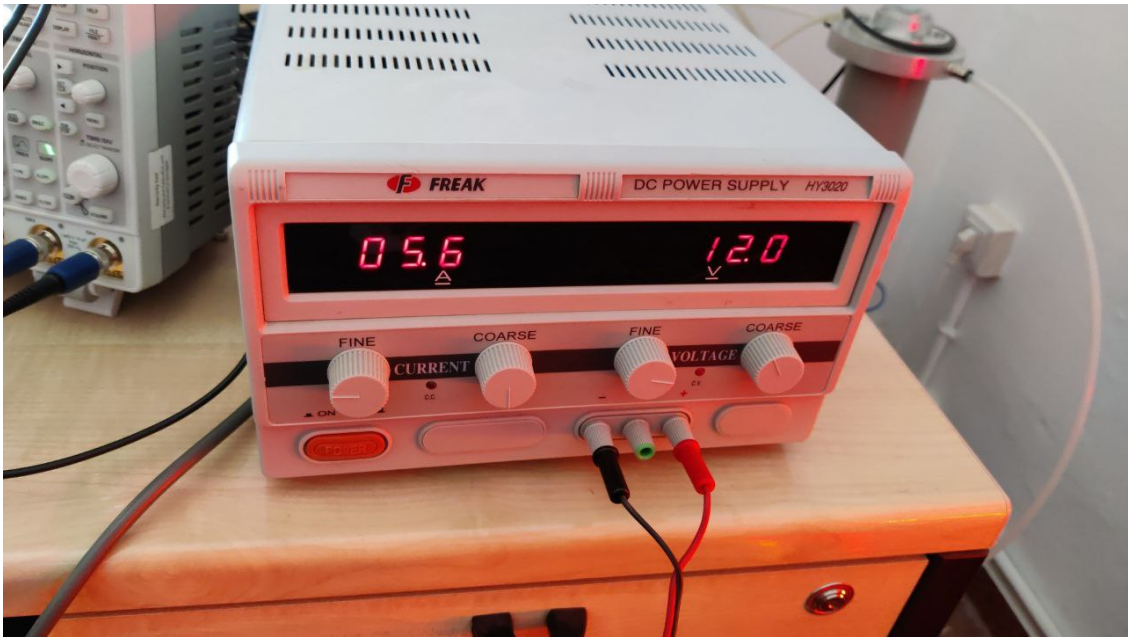
SMD 1206 $4.7k\Omega$ resistors (commercial value) are used:

$$V_{ADC} = 4.7k\Omega \cdot 600uA = 2.82V$$

Once theoretical calculations are completed, it will be checked.

1.3 Check calculations

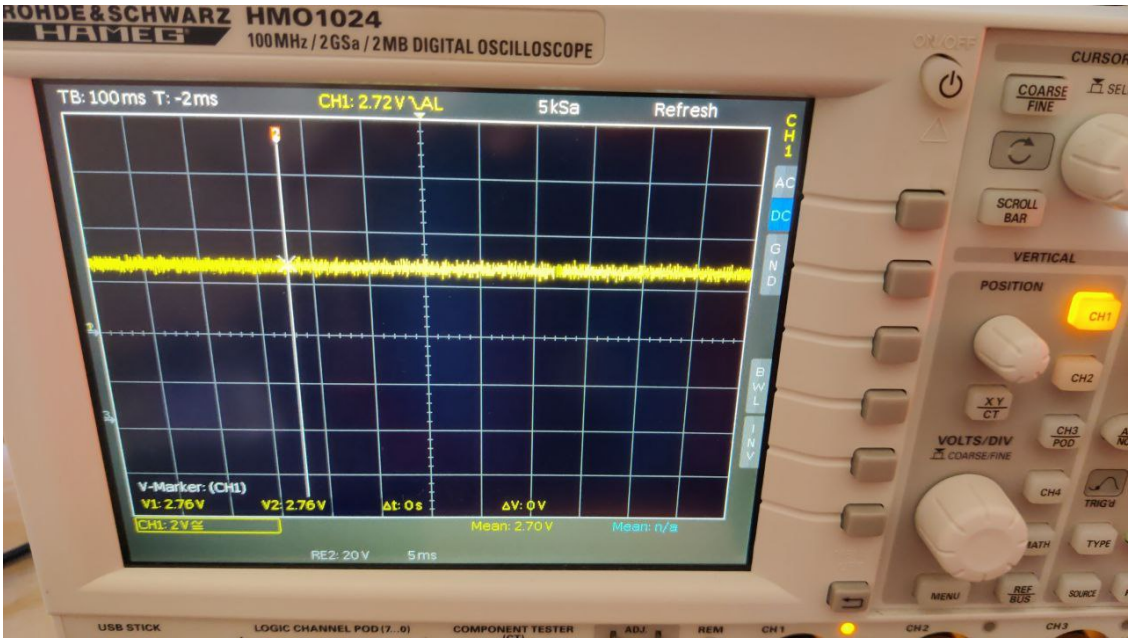
The resistors nominal value is $4.62k\Omega$, output current $I_L = 5.6A$ and powered at 12V:



Current and voltage conditions

$$V_{ADC_{CHECK}} = 4.62k\Omega \cdot 560uA = 2.59V$$

ADC voltage is checked by oscilloscope and this the result:



ADC voltage validation

Finally $V_{ADC_{CHECK}} = 2.70V$

$$I_S = \frac{V_S}{4.62k\Omega} = 584\mu A \tag{3}$$

$$I_O = I_S \cdot 10000 = 5.84A \tag{4}$$

$$P_D = V \cdot I_O = 12V \cdot 2.29A = 70W \tag{5}$$

$$E_{r_{PD}} = |1 - \frac{70W}{67.2W}| = 0.04 = 4\% \tag{6}$$

1.4 Conclusion

Once validation is checked, theoretical calculations agree with validation (V_{ADC} is between 0V and 3.3V).

Power consumption has a small error but the result may be valid.

Relative error could be due to ratio of output current to sense current (k=9590).

Therefore the resistor chosen for the configuration is a 4.7 kΩ resistor.

This last analysis concludes this subsection.

3.2.4 Enhanced PCB to maximise operability, repairability and testability.

This sub-section will consider the placement of the plates in the product box and therefore which parts are visible without having to release any of them, the symmetry in the design to save time in the placement of mechanical components, the placement of components so that the product can be fixed and the implementation of mechanisms to test the product with as little effort as possible.

It will start by analysing operability, followed by repairability and ending with testability:

Operability refers to the ability to operate or manipulate the product and in this case the aim is to make it as easy as possible. This is why it has been thought that at the time of design it should be taken into account:

- The separation of the pin header from the edges of the Printed Circuit Board (PCB) in order to be able to manipulate the connector without having problems of space, as depending on the place of the connector on the Printed Circuit Board (PCB), it could make disconnection difficult. This is taken into account for both boards.

- In addition to aligning the mechanical components (Liquid-Crystal Display (LCD), rotary encoder and Universal Serial Bus (USB)) so that when drilling holes in the case, the process is faster, it has been taken into account that most people are right-handed and that is why the rotary encoder would be to the right of the Liquid-Crystal Display (LCD). This measurement is mainly for the MicrocontrollerPCB.

The repairability or ability of an object to be repaired will be analysed below, taking into account the following points:

- Most of the components, especially on the MicrocontrollerPCB, have been placed in the BOTTOM layer, which is the one that is visible once the boards are anchored to the case, so it would be easier to check them. It should be noted that the LM7805 in particular will use the TO-220 packaging so that in the event of a breakage, the pins can be cut and unsoldered individually from the board, making it easier for the repair engineer or technician to replace them.

The testability or the degree and ease with which the software can be effectively tested will take into account the following points:

- That all Light-Emitting Diode (LED)s (on both Printed Circuit Board (PCB)s) are on the bottom layer which is the visible layer so that they can fulfil their role as a visual reference.

- The use of test points to check if the voltages at certain critical points for the system are correct.

- The implementation of a reset button, also on the bottom layer so that it can be pressed without difficulty.

Chapter 4

System description and design

This chapter will give an overview of the schematics, followed by detailed descriptions of the modules and ending with the 3D models of the components, following the order of Microcontroller PCB followed by ConnectorsPCB.

Finally a 3D image of the PCBs, as a result of the multi-assembly project, will be shown.

4.1 System overview

This 3D figure is the result of the multi-assembly project where all Printed Circuit Board (PCB) projects are related in a single representation. The latest and improved version is shown, the other versions will be shown in the appendix and the reasons why they needed to be improved will be given.

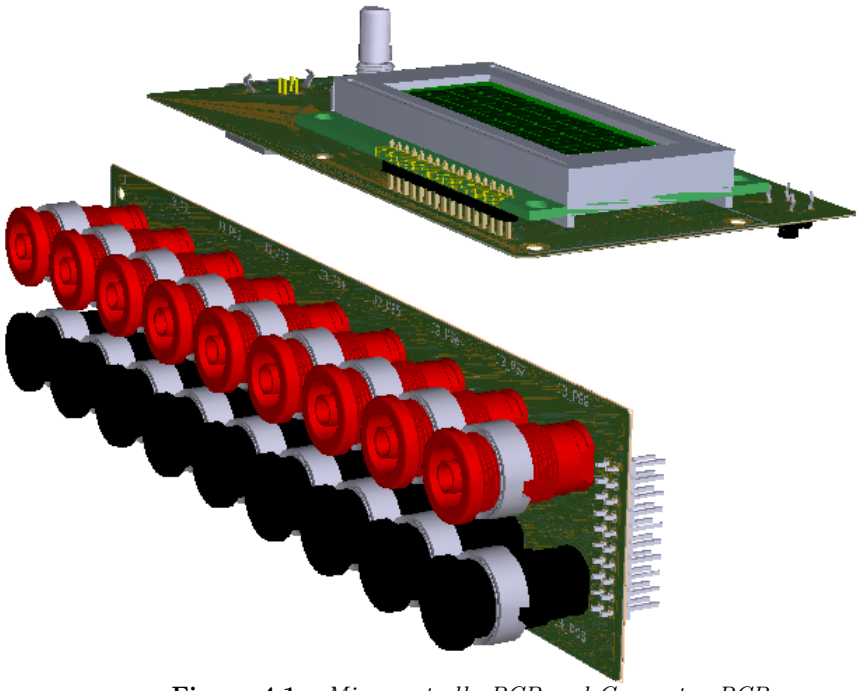


Figure 4.1 – *MicrocontrollerPCB and ConnectorsPCB*

4

4.2 Electronics design

4.2.1 MicrocontrollerPCB

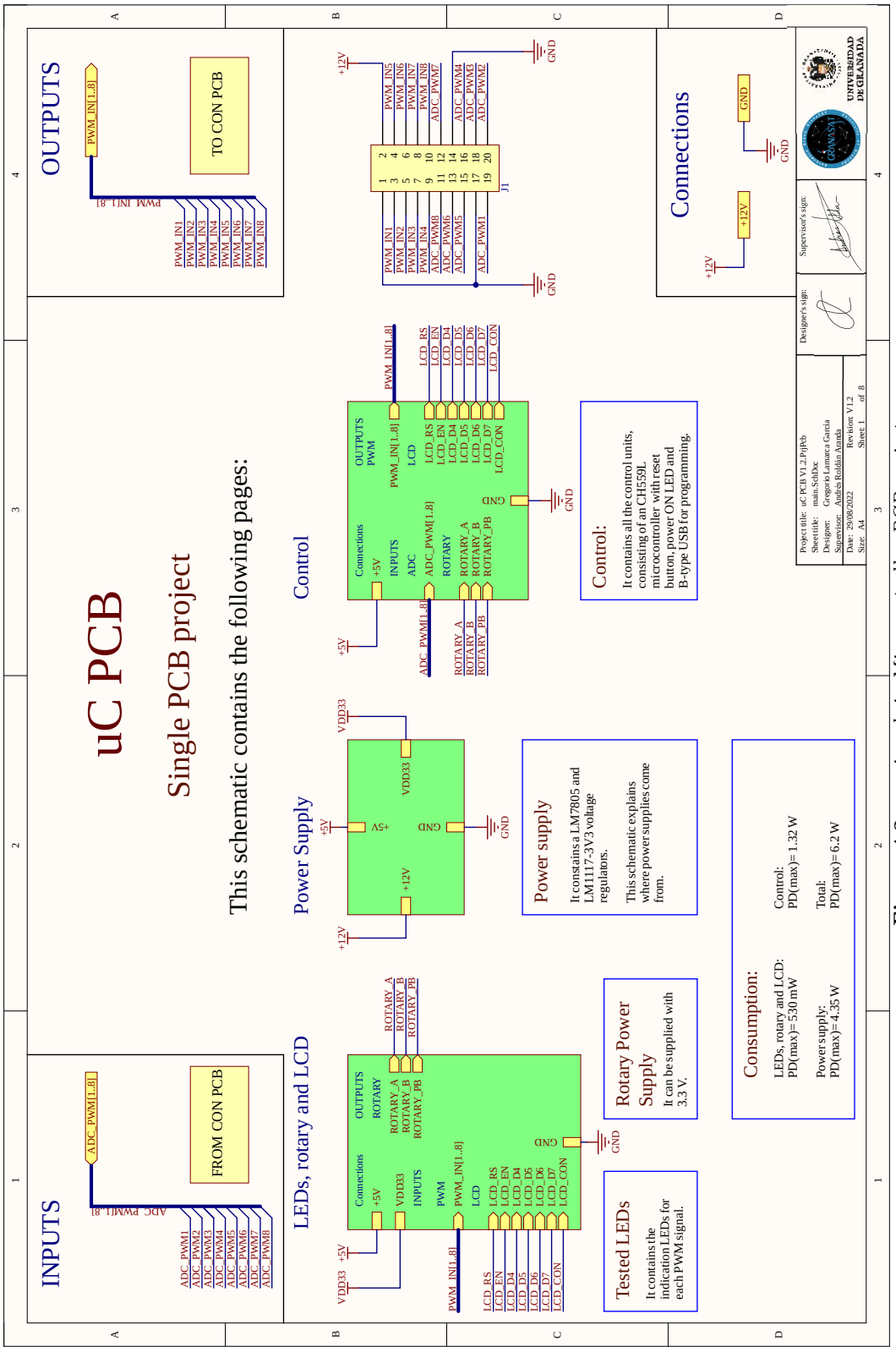


Figure 4.2 – *main.sch in Microcontroller PCB project*

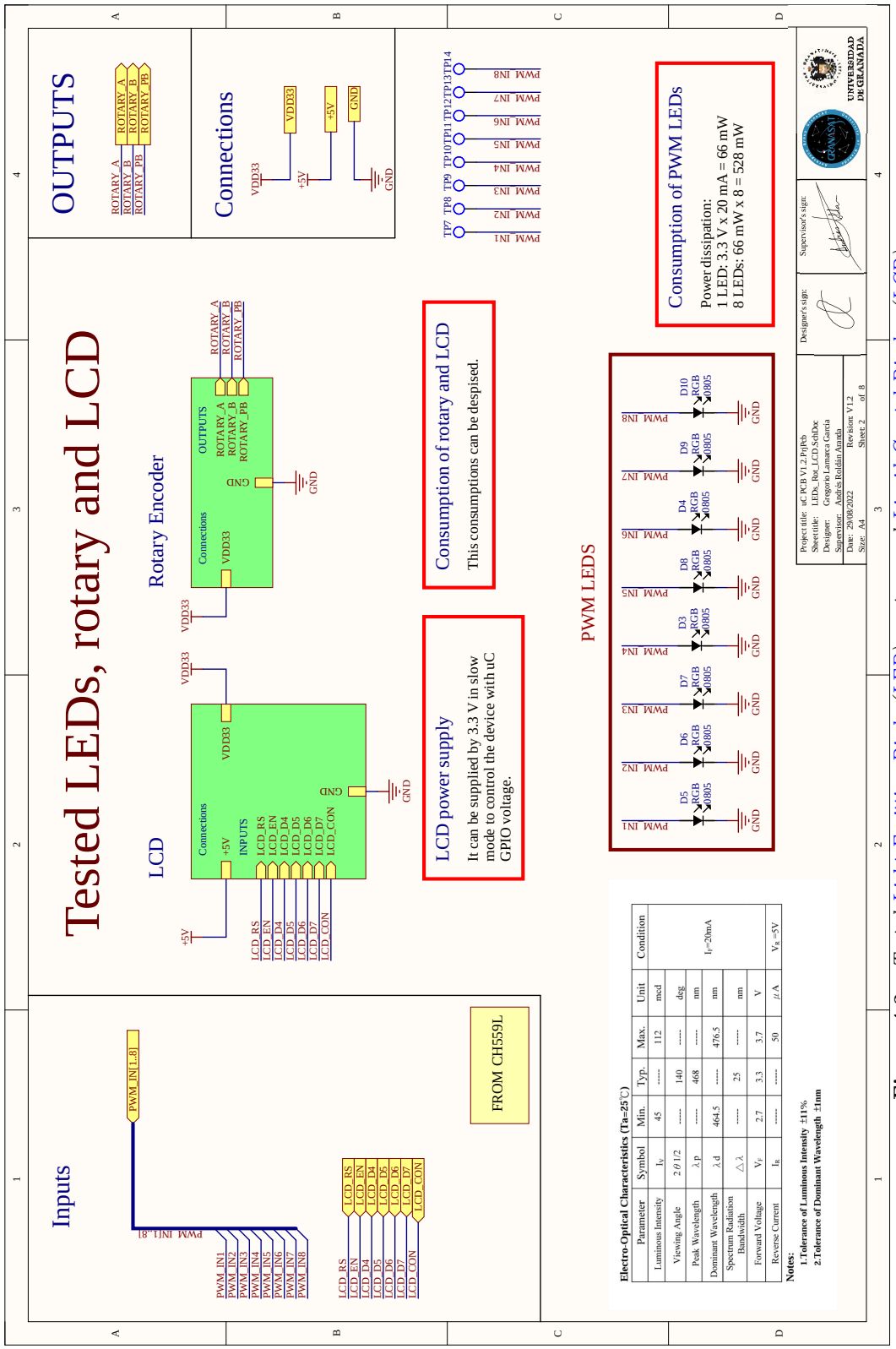


Figure 4.3 – *Tested Light-Emitting Diode (LED)s, rotary and Liquid-Crystal Display (LCD)*

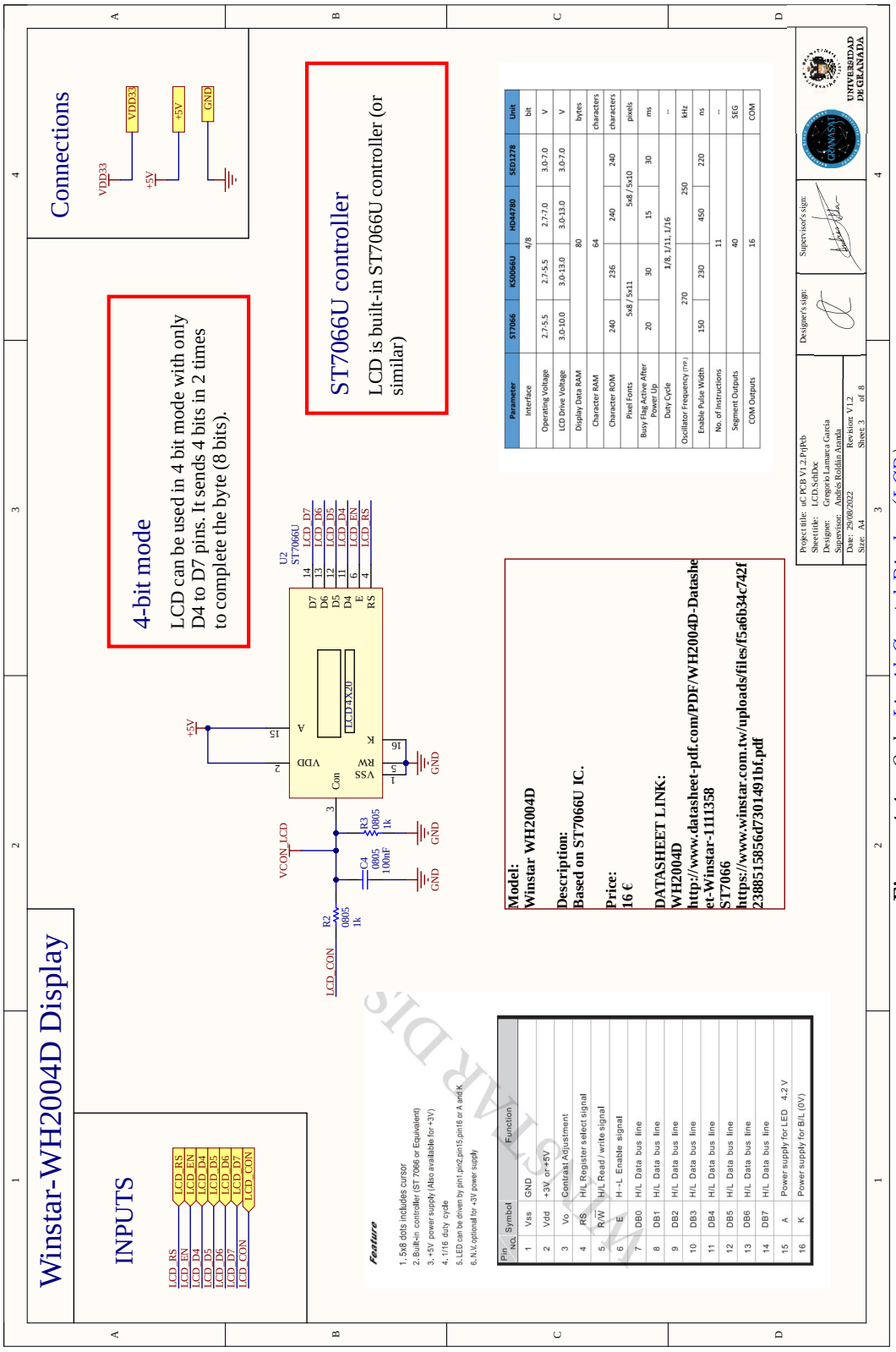


Figure 4.4 – *Only Liquid-Crystal Display (LCD)*

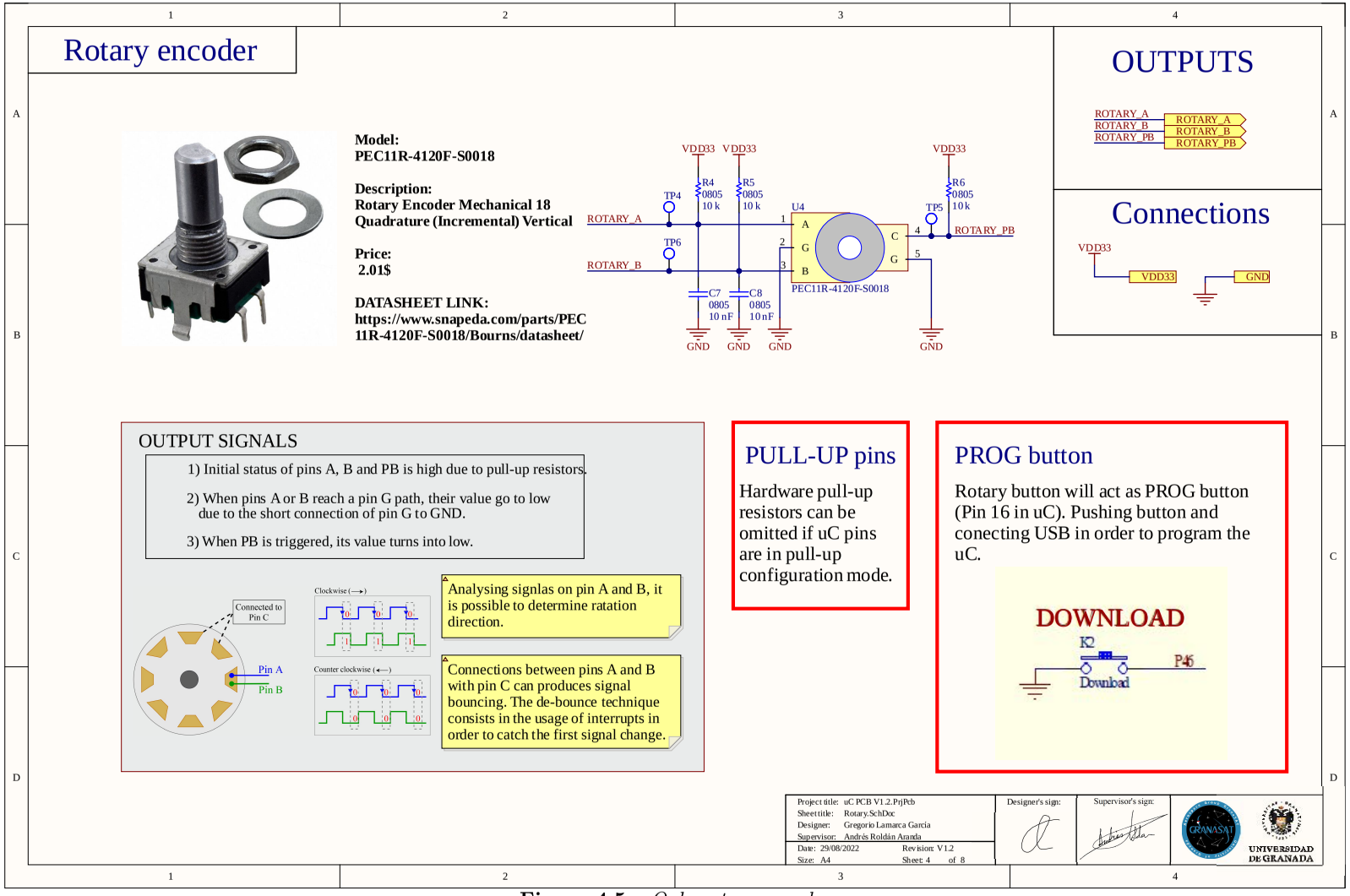


Figure 4.5 – *Only rotary encoder*

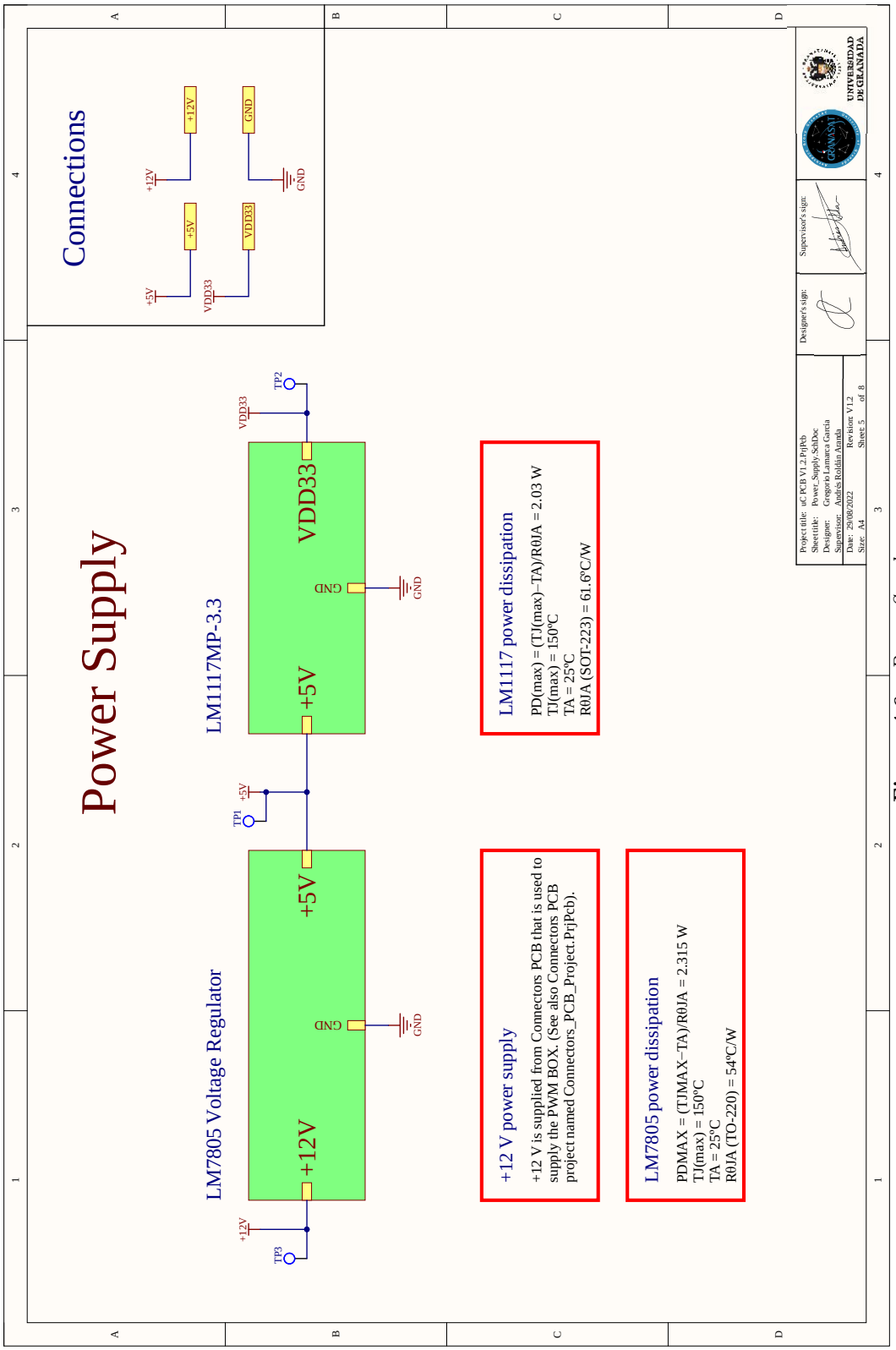


Figure 4.6 – *Power Supply*

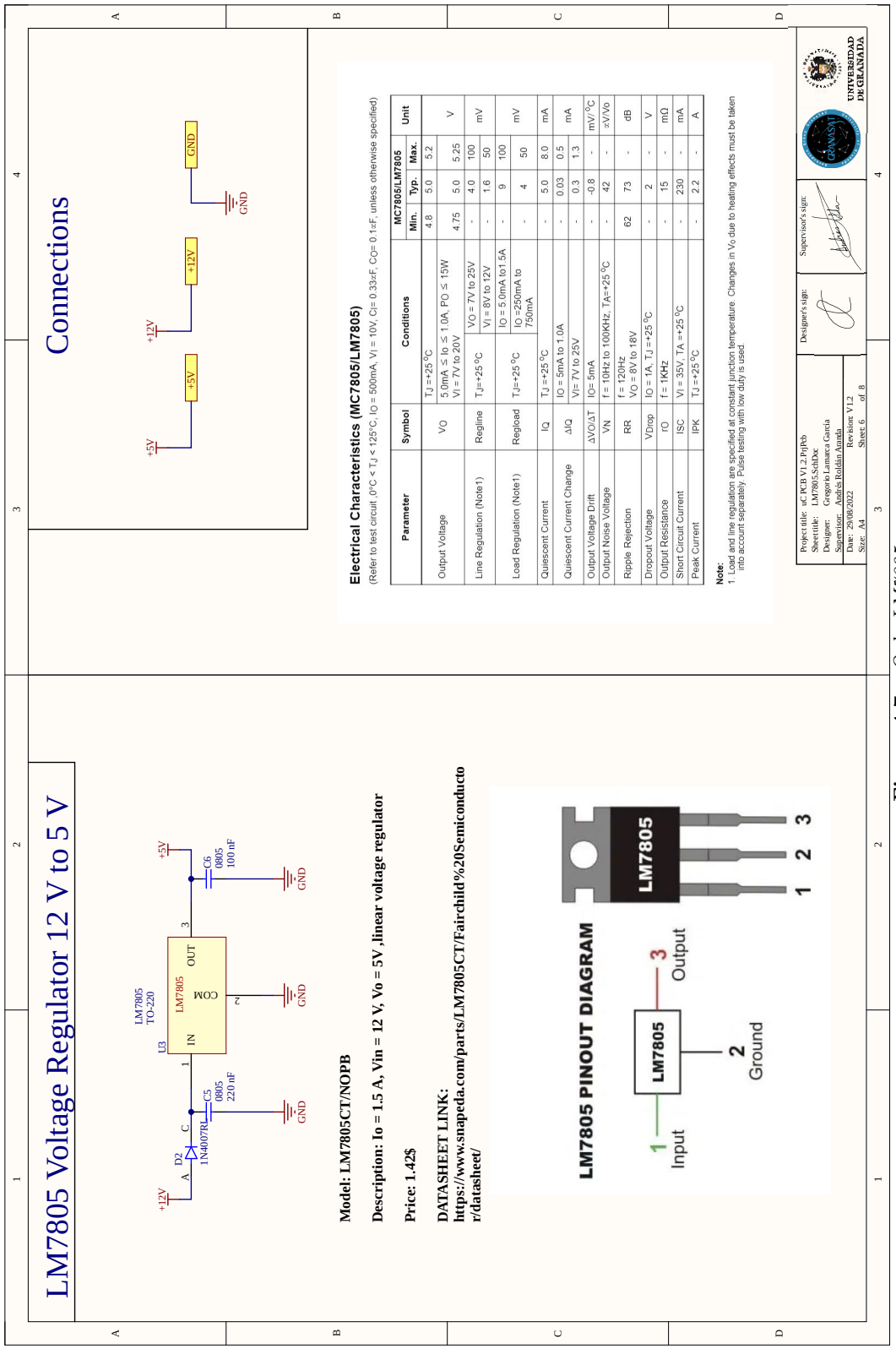


Figure 4.7 – *Only LM7805*

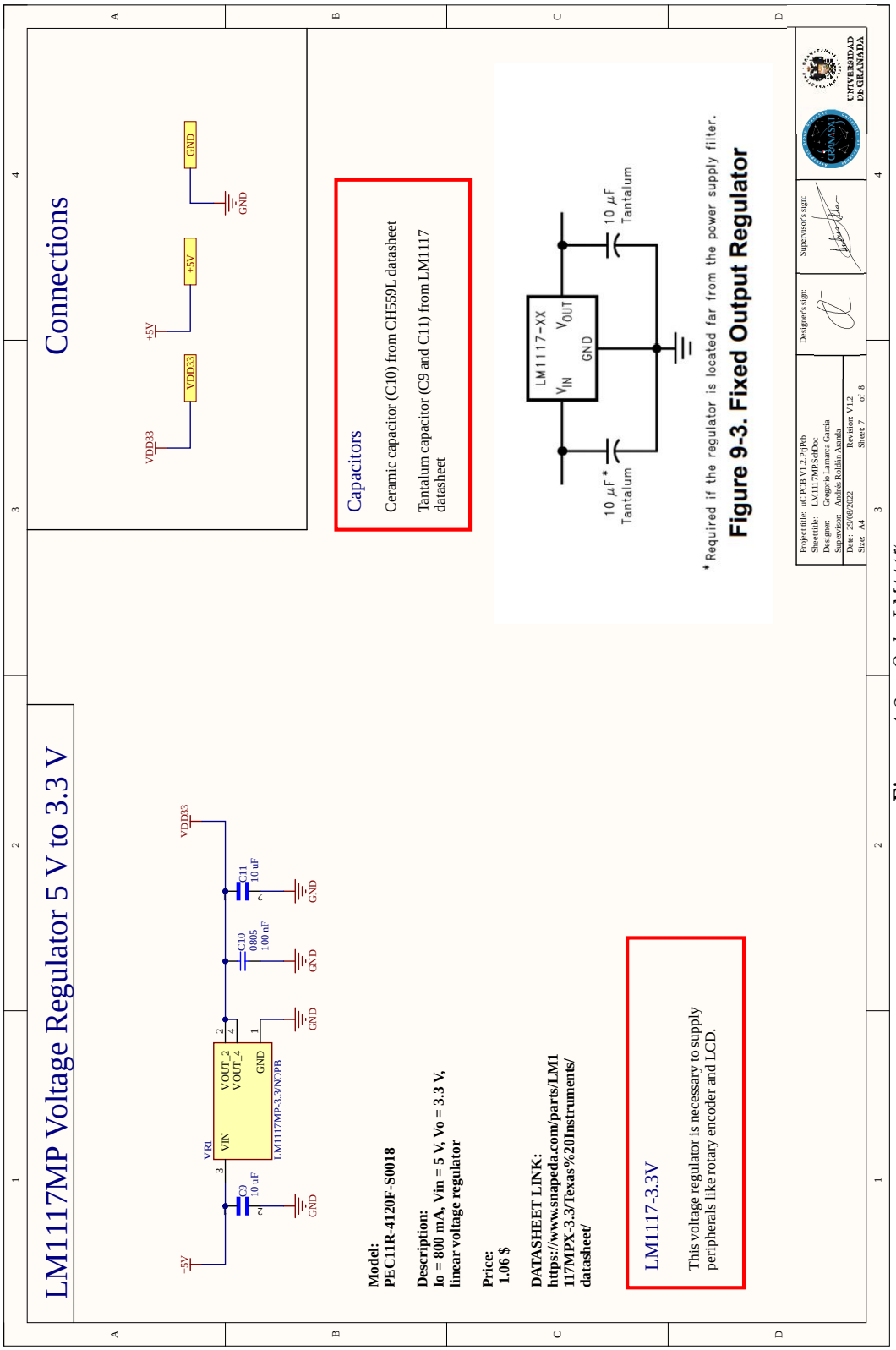


Figure 4.8 – *Only LM1117*

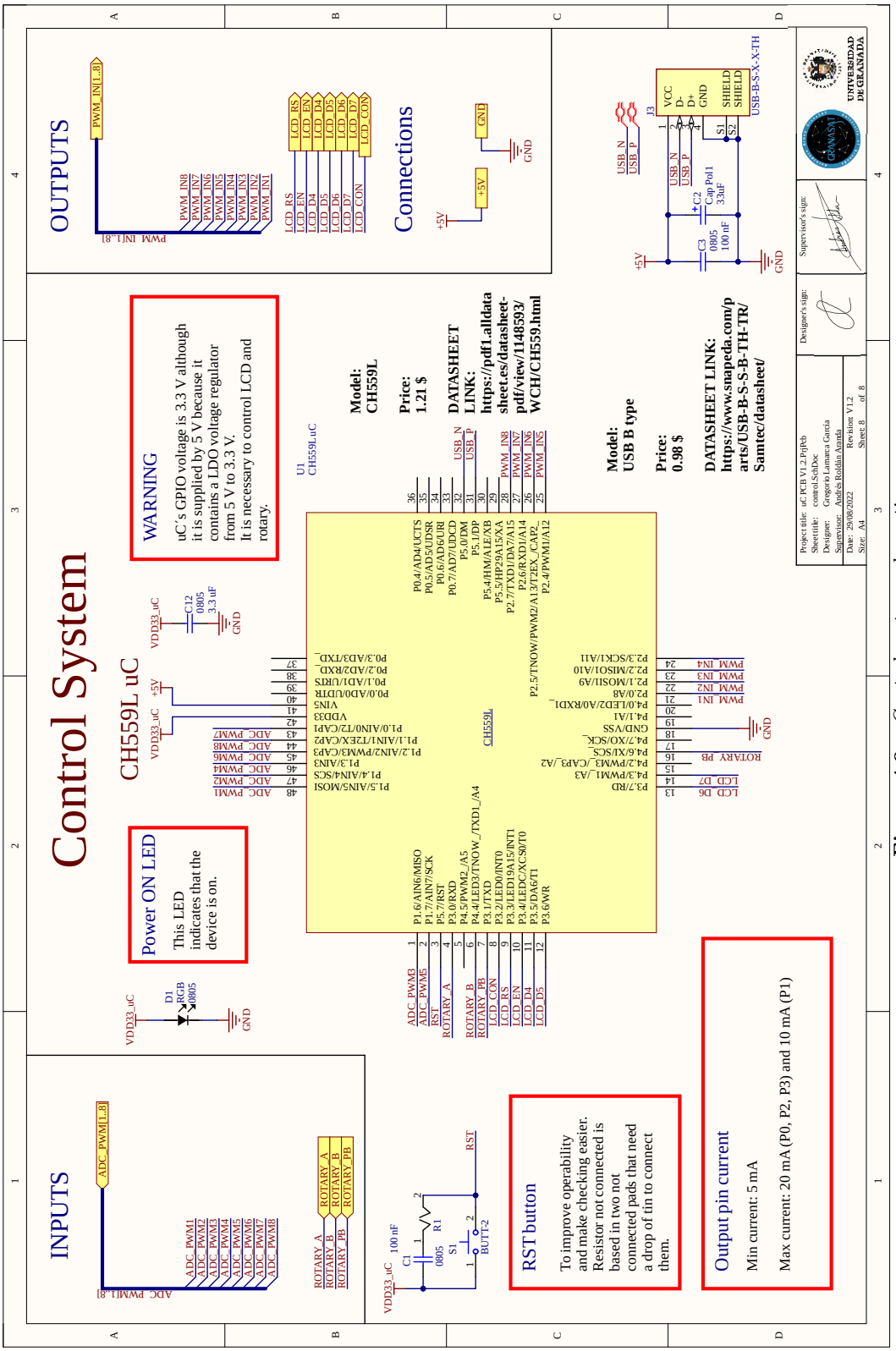


Figure 4.9 – *Control system schematic*

Starting with the first sheet of the schematics, we will first describe the components that each sheet has, the connections they need, the price of each component and an explanation of the chosen configuration with the support of data sheets.

4.2.1.1 Pin header

The main component of 4.2 is the pin header that connects the MicrocontrollerPCB and the ConnectorsPCB.

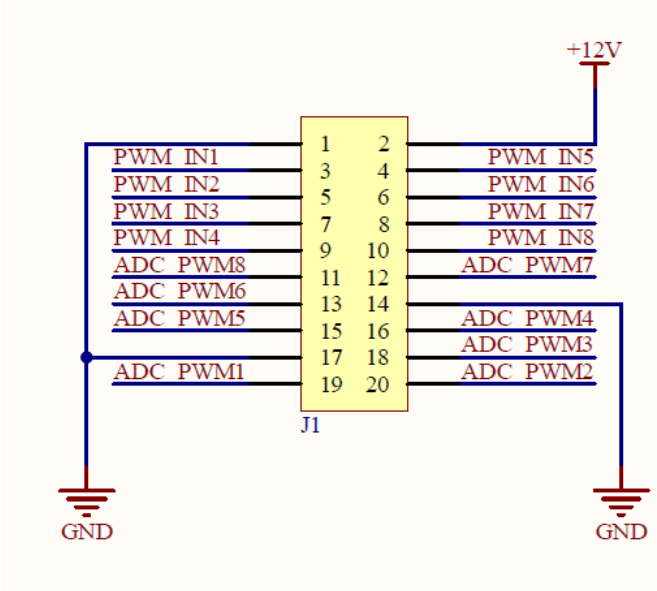


Figure 4.10 – *Pin header in MicrocontrollerPCB*

The pin header connections would look like this, being of vital importance that this order is respected on the other board. As the connections are exchanges of inputs and outputs, they will be specified so that it is completely clear where they come from. The price of the component is 0.86 €.

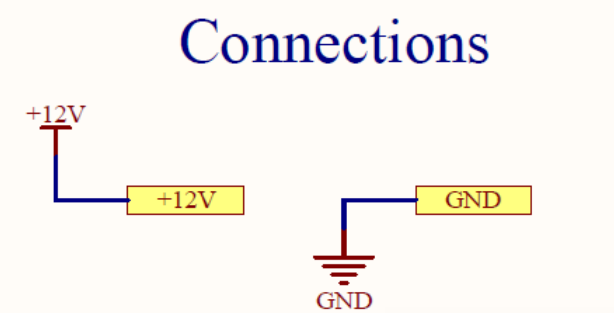


Figure 4.11 – *Connections in pin header of MicrocontrollerPCB*

The 12 V and Ground (GND) connection comes from the other bale when the voltage generator is connected to power the product. The 12 V are brought to this Printed Circuit Board (PCB) for space reasons, as this board has larger dimensions to use a voltage regulator at 5 V, which are only used on the MicrocontrollerPCB.

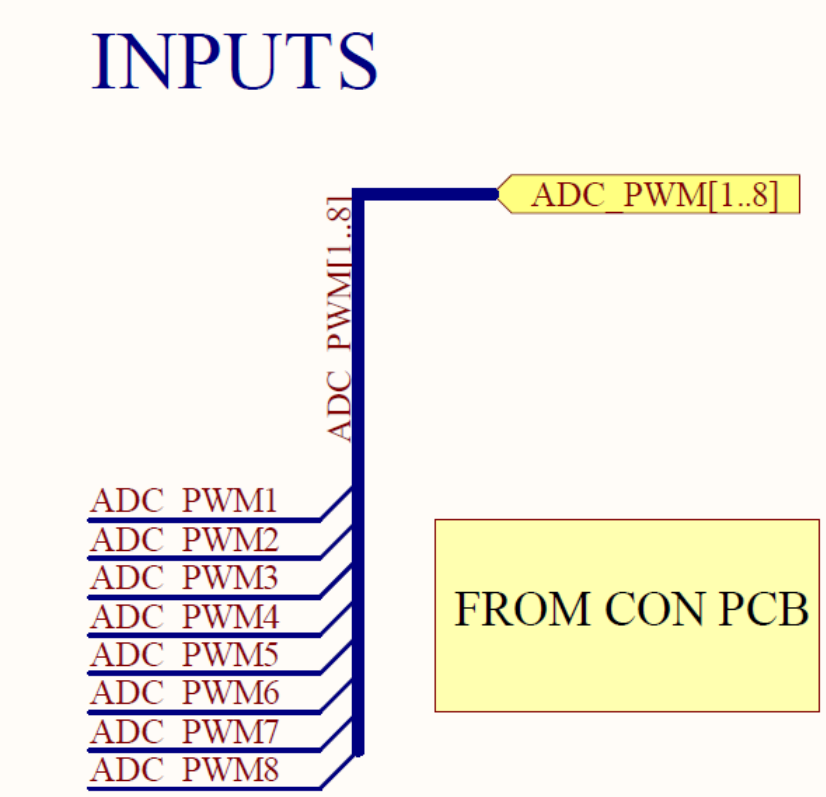


Figure 4.12 – *Inputs in pin header of MicrocontrollerPCB*

ADC-PWM is the name given to the connections that come from ConnectorsPCB and carry the analogue voltage signals that are read into the micro controller via the ADC channels.

It can be seen how the analogue channels have been interleaved with Ground (GND) to avoid cable couplings between connectors, although their effects have been studied and are negligible at the working frequency of the BTS6143D.

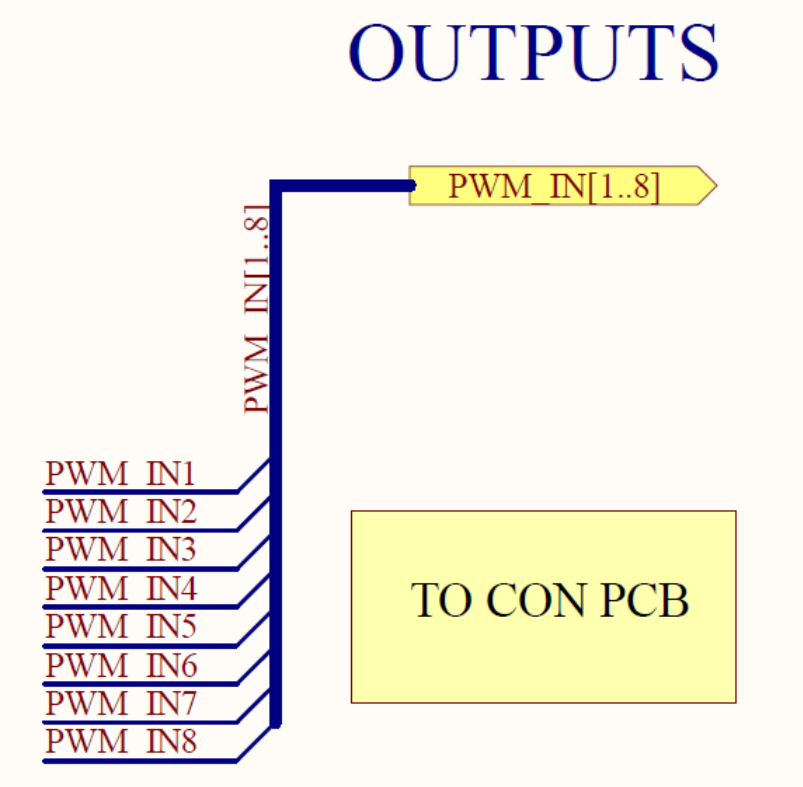


Figure 4.13 – *Outputs in pin header of MicrocontrollerPCB*

PWM-IN are the output signals that go to the ConnectorsPCB used to drive the transistors that cause the BTS6143D to connect the 12 V to its output and thus power the load.

4.2.1.2 LEDs in PWM-IN

The second schematic [4.3] shows the 8 Light-Emitting Diode (LED)s for checking the correct operation of the pins. The price of each LED is 0.90 €.

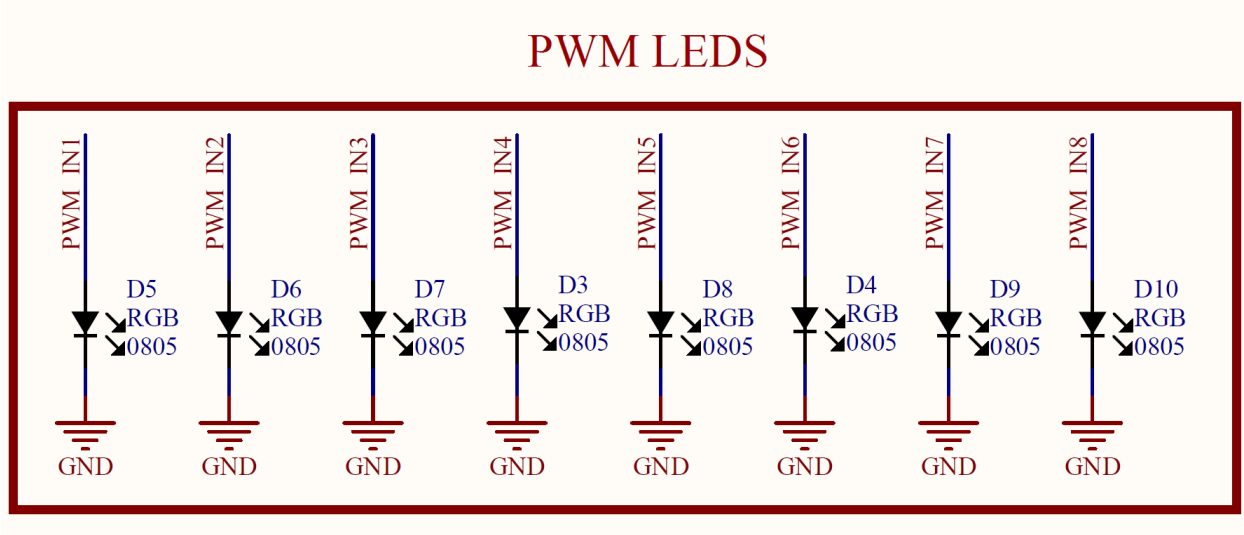


Figure 4.14 – *LEDs of MicrocontrollerPCB*

Below is the data sheet of the Light-Emitting Diode (LED)s which explains why a protective resistor is not necessary.

4

Electro-Optical Characteristics (Ta=25℃)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
Luminous Intensity	I_V	45	-----	112	mcd	I_F=20mA
Viewing Angle	2θ 1/2	-----	140	-----	deg	
Peak Wavelength	λ p	-----	468	-----	nm	
Dominant Wavelength	λ d	464.5	-----	476.5	nm	
Spectrum Radiation Bandwidth	$\triangle\lambda$	-----	25	-----	nm	
Forward Voltage	V_F	2.7	3.3	3.7	V	
Reverse Current	I_R	-----	-----	50	μA	V_R =5V

Notes:

1.Tolerance of Luminous Intensity ±11%

2.Tolerance of Dominant Wavelength ±1nm

Figure 4.15 – *LEDs' data sheet of MicrocontrollerPCB*

As the normal pin-to-pin voltage of the diode is 3.3 V and matches the voltage provided by the micro controller, the protection resistor is removed.

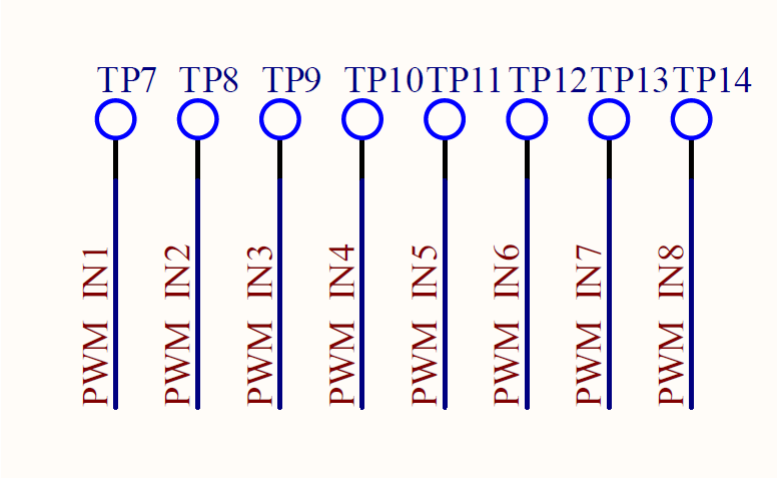


Figure 4.16 – *LEDs' test points of MicrocontrollerPCB*

One test point has been placed to improve testability. The individual and total consumption of the Light-Emitting Diode (LED)s can be seen in the figure below.

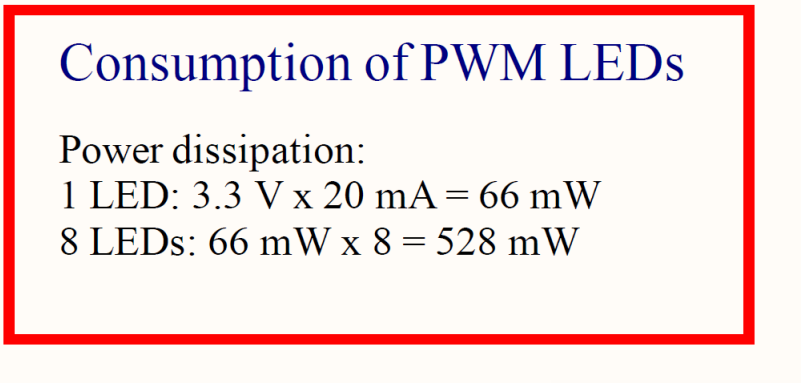


Figure 4.17 – *LEDs' consumption of MicrocontrollerPCB*

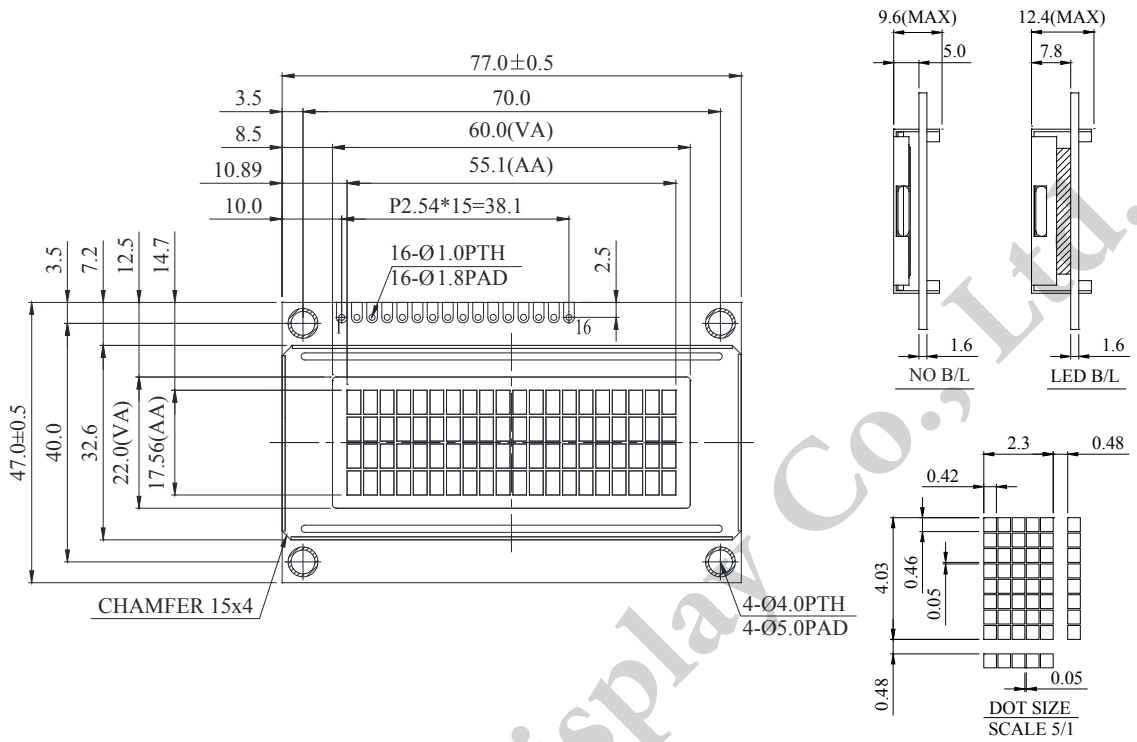
4.2.1.3 LCD

In the schematic of the Liquid-Crystal Display (LCD) screen [4.4], its connections and pins are correctly shown as it has a sheet dedicated only to this component, so we will only estimate the total price taking into account the other components necessary for its configuration. The total price would be 16.48 €, 16 € for the Liquid-Crystal Display (LCD) and 0.48 € for the two resistors and the capacitor.

Pin 3, which is the contrast pin, has a low-pass filter and voltage divider which, powered by Pulse-Width Modulation (PWM), can be controlled from the BRIGHTNESS menu of the software.

The mechanical characteristics of the Liquid-Crystal Display (LCD) will be presented below:

WH2004D Character 20x4



Feature

1.5x8 dots includes cursor
2.Bulit-in controller (ST7066 or Equivalent)
3.5V power supply (Also available for 3V)
4.N.V, optional for 3V power supply
5.1/16 duty cycle
6.LED can be driven by PIN1, PIN2, PIN15, PIN16 or A and K
7.Interface : 6800, option SPI/I2C (RW1063 IC)

Pin No.	Symbol	Description
1	V_{SS}	Ground
2	V_{DD}	Power supply for logic
3	V_O	Contrast Adjustment
4	RS	Data/ Instruction select signal
5	R/W	Read/Write select signal
6	E	Enable signal
7	DB0	Data bus line
8	DB1	Data bus line
9	DB2	Data bus line
10	DB3	Data bus line
11	DB4	Data bus line
12	DB5	Data bus line
13	DB6	Data bus line
14	DB7	Data bus line
15	A	Power supply for B/L +
16	K	Power supply for B/L -

Mechanical Data

Item	Standard Value	Unit
Module Dimension	77.0 x 47.0	mm
Viewing Area	60.0 x 22.0	mm
Mounting Hole	70.0 x 40.0	mm
Character Size	2.30 x 4.03	mm

Electrical Characteristics

Item	Symbol	Standard Value typ.	Unit
Input Voltage	VDD	3/5	V
Recommended LCD Driving Voltage for Normal Temp. Version module @25°C	VDD-VO	4.50	V

Display Character Address Code

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	---	---	20
DD RAM Address	00	01														13
DD RAM Address	40	41														53
DD RAM Address	14	15														27
DD RAM Address	54	55														67

4.2.1.4 Rotary encoder

The rotary encoder, like the Liquid-Crystal Display (LCD), has its own schematic sheet [4.5] explaining how it works. The total price would be 2.81 € (2.01 € for the rotary encoder, 0.12 € for the capacitor and 0.18€ for the resistor).

For a better electrical behaviour, it has to be adapted with resistors and capacitors as shown in the picture. It also works without the matching but the signals generated by the matching will be less stable. A test point has been placed on each pin to make testing easier.

The mechanical characteristics of the rotary encoder will be presented below [1]:

Applications

Level control, tuning and timer settings in:

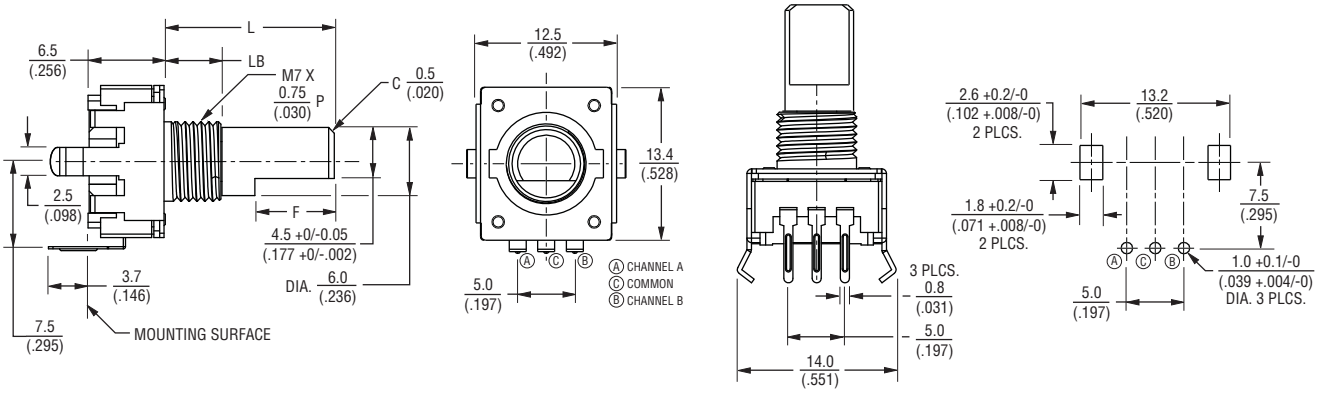
- Audio-visual equipment
- Consumer electric appliances
- Radios
- Musical instrumentation
- Communications equipment

PEC11R Series - 12 mm Incremental Encoder

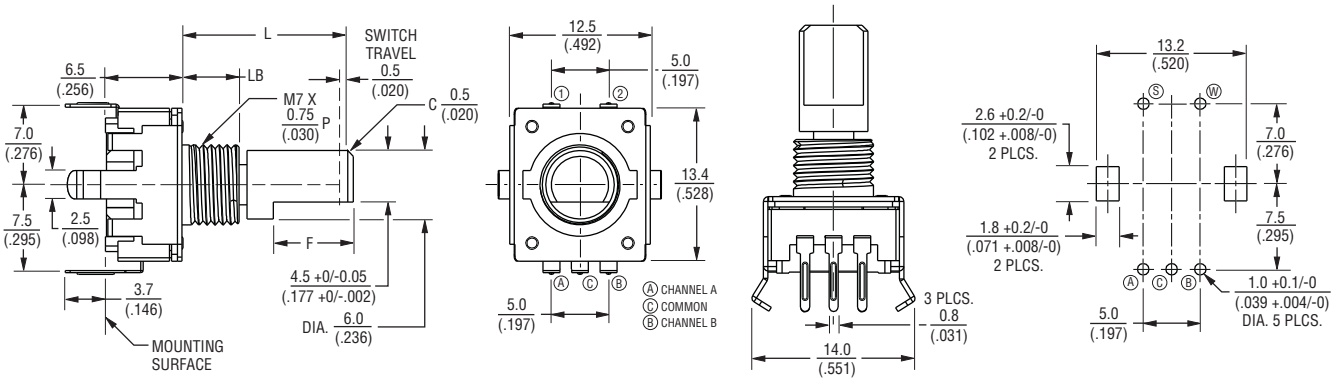
BOURNS®

Product Dimensions

PEC11R-4xxxF-Nxxxx



PEC11R-4xxxF-Sxxxx



L	LB	F
15 (.591)	5.0 (.197)	7.0 (.276)
20 (.787)	7.0 (.276)	10.0 (.394)
25 (.984)	7.0 (.276)	12.0 (.472)
30 (1.181)	7.0 (.276)	12.0 (.472)

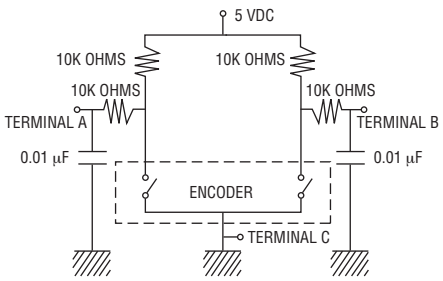
DIMENSIONS: MM (INCHES)

TOLERANCES: < 10 (.394) = ± 0.3 (.012)

≥ 10 (.394) = ± 0.5 (.020)

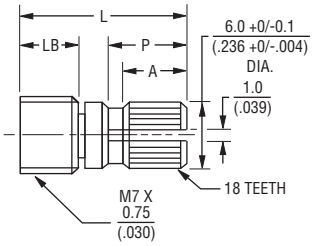
Switch Circuit

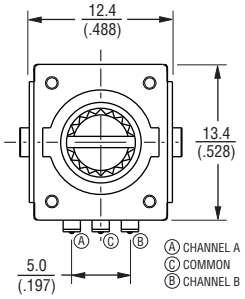
Suggested Filter Circuit



Product Dimensions

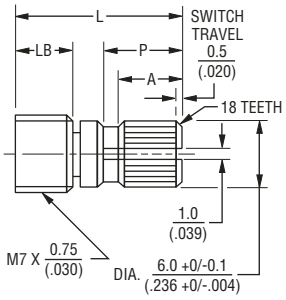
PEC11R-4xxxK-Nxxxx

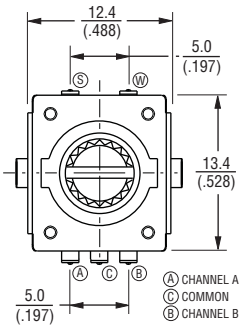




L	LB	P	A
15 (.591)	5.0 (.197)	7.0 (.276)	6.0 (.236)
20 (.787)	7.0 (.276)	7.0 (.276)	6.0 (.236)
30 (1.181)	7.0 (.276)	16.0 (.630)	12.0 (.472)

PEC11R-4xxxK-Sxxxx





L	LB	P	A
15 (.591)	5.0 (.197)	7.0 (.276)	6.0 (.236)
20 (.787)	7.0 (.276)	7.0 (.276)	6.0 (.236)

DIMENSIONS: MM (INCHES)

TOLERANCES: < 10 (.394) = ±0.3 (.012)

≥ 10 (.394) = ±0.5 (.020)

BOURNS®

Asia-Pacific: Tel: +886-2 2562-4117 • Fax: +886-2 2562-4116
Europe: Tel: +41-41 768 5555 • Fax: +41-41 768 5510
The Americas: Tel: +1-951 781-5500 • Fax: +1-951 781-5700
www.bourns.com

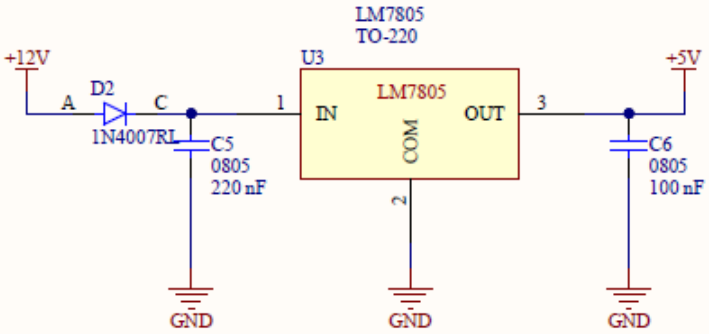
REV. 06/13

4.2.1.5 LM7805

The LM7805 [4.7] is the 12 V to 5 V voltage regulator chosen for the design and the price was reviewed in [3.2.3.2]. The price of LM7805 component and configuration is 1.92 €

The packaging chosen was TO-220 due to its repairability when placed in the bottom layer.

Below is their description in the project followed by the mechanical description and application obtained from the data sheet [3]:



Model: LM7805CT/NOPB

Description: Io = 1.5 A, Vin = 12 V, Vo = 5V ,linear voltage regulator

Price: 1.42$

DATASHEET LINK:
https://www.snapeda.com/parts/LM7805CT/Fairchild%20Semiconductor/datasheet/

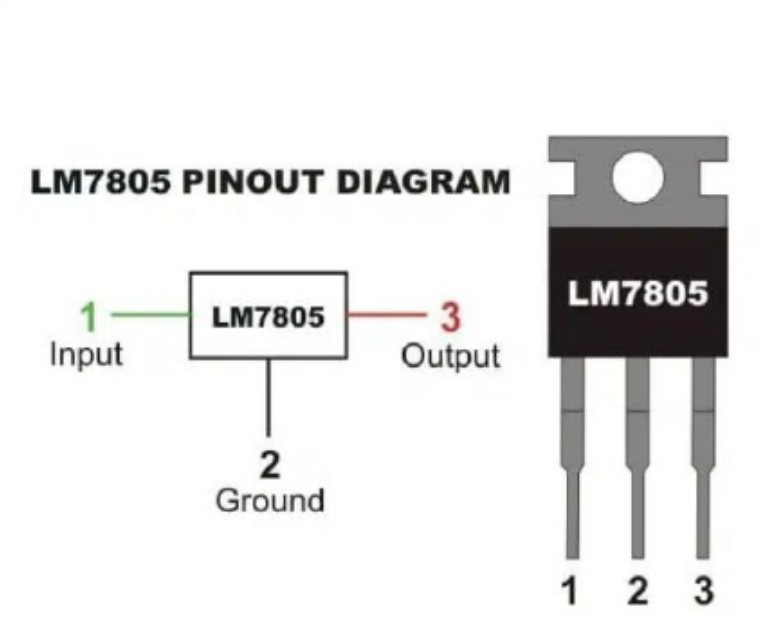


Figure 4.18 – *LM7805's description of MicrocontrollerPCB*

LM340, LM340A and LM7805 Family Wide V_{IN} 1.5-A Fixed Voltage Regulators

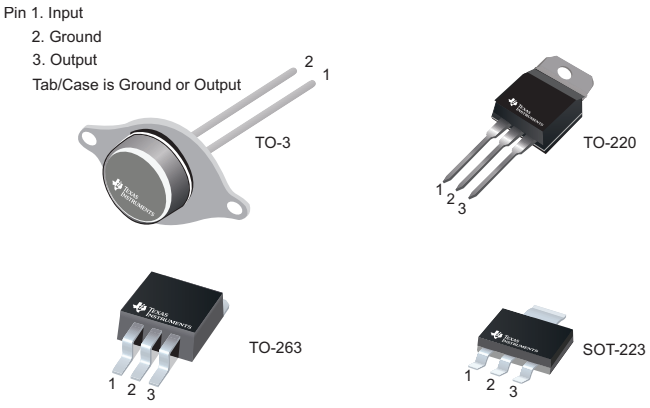
1 Features

- Output Current up to 1.5 A
- Available in Fixed 5-V, 12-V, and 15-V Options
- Output Voltage Tolerances of ±2% at T_J = 25°C (LM340A)
- Line Regulation of 0.01% / V of at 1-A Load (LM340A)
- Load Regulation of 0.3% / A (LM340A)
- Internal Thermal Overload, Short-Circuit and SOA Protection
- Available in Space-Saving SOT-223 Package
- Output Capacitance Not Required for Stability

2 Applications

- Industrial Power Supplies
- SMPS Post Regulation
- HVAC Systems
- AC Inventors
- Test and Measurement Equipment
- Brushed and Brushless DC Motor Drivers
- Solar Energy String Invertors

Available Packages



3 Description

The LM340 and LM7805 Family monolithic 3-terminal positive voltage regulators employ internal current-limiting, thermal shutdown and safe-area compensation, making them essentially indestructible. If adequate heat sinking is provided, they can deliver over 1.5-A output current. They are intended as fixed voltage regulators in a wide range of applications including local (on-card) regulation for elimination of noise and distribution problems associated with single-point regulation. In addition to use as fixed voltage regulators, these devices can be used with external components to obtain adjustable output voltages and currents.

Considerable effort was expended to make the entire series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

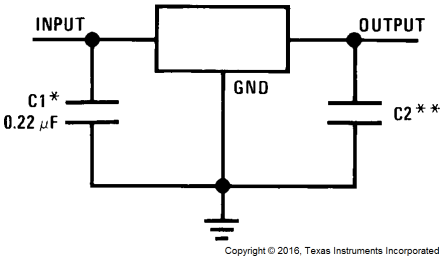
LM7805 is also available in a higher accuracy and better performance version (LM340A). Refer to LM340A specifications in the *LM340A Electrical Characterisitcs* table.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM340x LM7805 Family	DDPAK/TO-263 (3)	10.18 mm × 8.41 mm
	SOT-223 (4)	6.50 mm × 3.50 mm
	TO-220 (3)	14.986 mm × 10.16 mm
	TO-3 (2)	38.94 mm x 25.40 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Fixed Output Voltage Regulator



*Required if the regulator is located far from the power supply filter.
**Although no output capacitor is needed for stability, it does help transient response. (If needed, use 0.1-µF, ceramic disc).

4.2.1.6 LM1117

The LM1117 [4.8] is the 5 V to 3.3 V voltage regulator chosen for the design and the price was reviewed in [3.2.3.3]. The price of LM1117 component and configuration is 2.86 €.

The packaging chosen was SOT DCY.

Below is their description in the project followed by the mechanical description and application obtained from the data sheet:

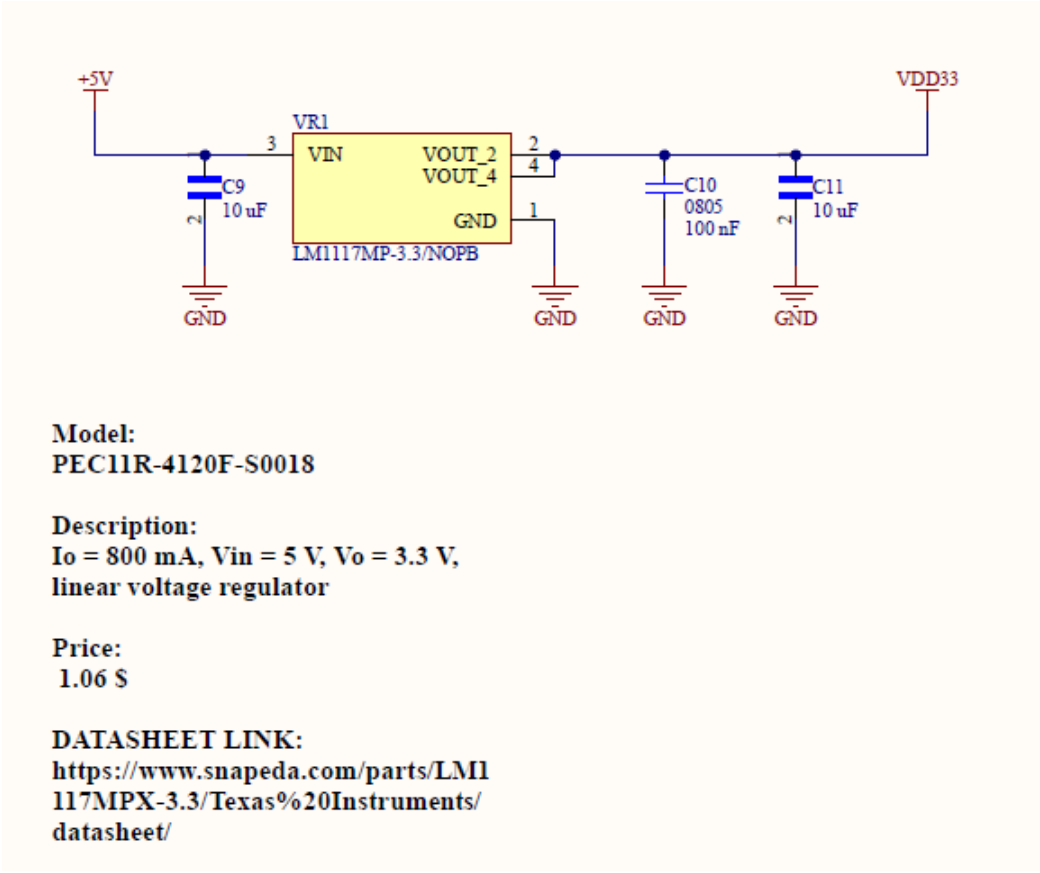


Figure 4.19 – *LM1117's description of MicrocontrollerPCB*

5 Device Comparison Table

I_{OUT}	PARAMETER	LM1117	TLV1117	UNIT
800 mA	Input voltage range (max)	15	15	V
	Load regulation accuracy	1.6	1.6	%
	PSRR (120 Hz)	75	75	dB
	Recommended operating temperature	0 – 125	-40 – 125	°C
	SOT-223 T_{JA}	61.6	104.3	°C/W
	TO-220 T_{JA}	23.8	30.1	°C/W
	TO-252 T_{JA}	45.1	50.9	°C/W
	TO-263 T_{JA}	41.3	27.5	°C/W
	WSON-8 T_{JA}	39.3	38.3	°C/W

6 Pin Configuration and Functions

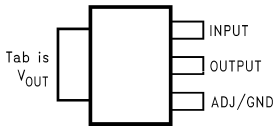


Figure 6-1. 4-Pin SOT DCY Package (Top View)

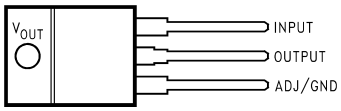


Figure 6-2. 3-Pin TO-220 NDE Package (Top View)

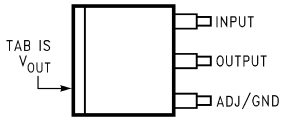


Figure 6-3. 3-Pin TO-263 KTT Package (Top View)

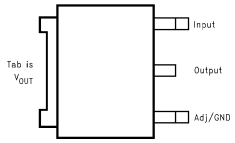
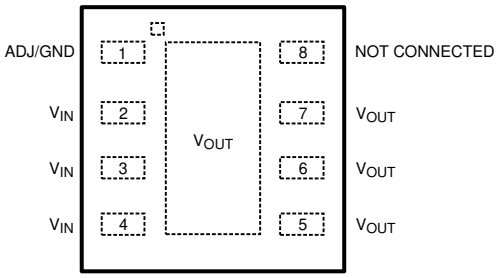


Figure 6-4. 3-Pin TO-252 NDP Package (Top View)



When using the WSON package pins 2, 3, and 4 must be connected together and pins 5, 6, and 7 must be connected together.

Figure 6-5. 8-Pin WSON NGN Package (Top View)

Table 6-1. Pin Functions

PIN						I/O	DESCRIPTION
NAME	TO-252	WSON	SOT-223	TO-263	TO-220		
ADJ/GND	1	1	1	1	1	—	Adjust pin for adjustable output option. Ground pin for fixed output option.
V_{IN}	3	2, 3, 4	3	3	3	I	Input voltage pin for the regulator
V_{OUT}	2 , TAB	5, 6, 7, TAB	2, 4	2, TAB	2, TAB	O	Output voltage pin for the regulator

4.2.1.7 CH559L and USB B type

Finally the CH559L micro controller is described with the chosen settings such as the power LED and reset button, followed by the description of the USB- B [4.9].

The price of this module, counting the micro controller and USB is 4.08€ (1.21€ for CH559L, 0.98€ for USB, 0.90€ for LED, 3 x 0.12€ for capacitors, 0.45€ for electrolytic capacitor and 0.18€ for the button).

Note that the pin voltage is 3.3 maximum as the CH559 has an internal regulator.

The maximum pin current is 20 mA and the minimum is 5 mA and can be controlled by code.

4.2.1.8 Final result of MicrocontrollerPCB

This subsection shows how the Printed Circuit Board (PCB) has looked from different angles to appreciate the result as much as possible.

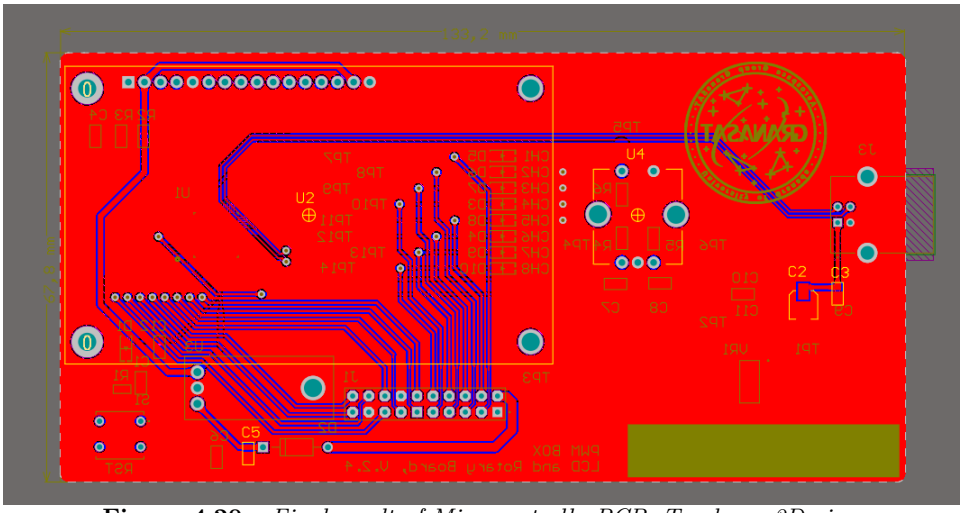


Figure 4.20 – *Final result of MicrocontrollerPCB. Top layer 2D view*

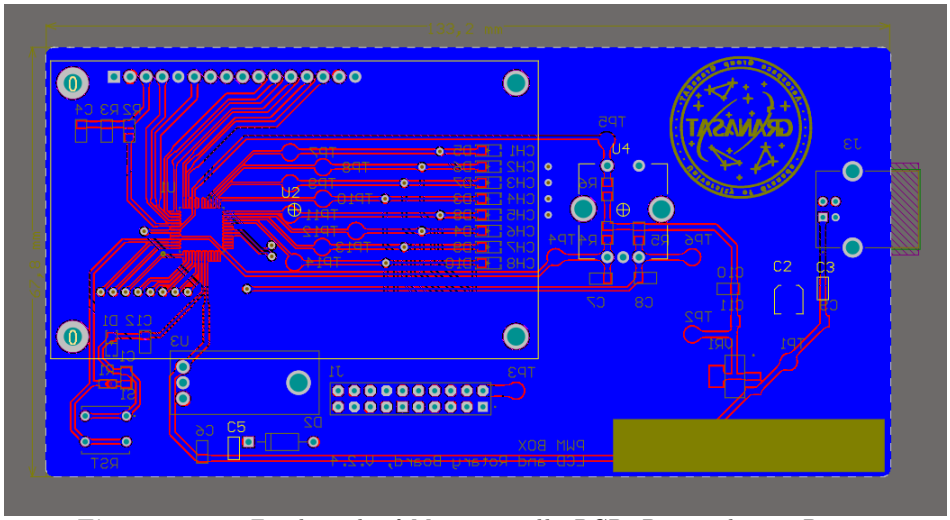


Figure 4.21 – *Final result of MicrocontrollerPCB. Bottom layer 2D view*

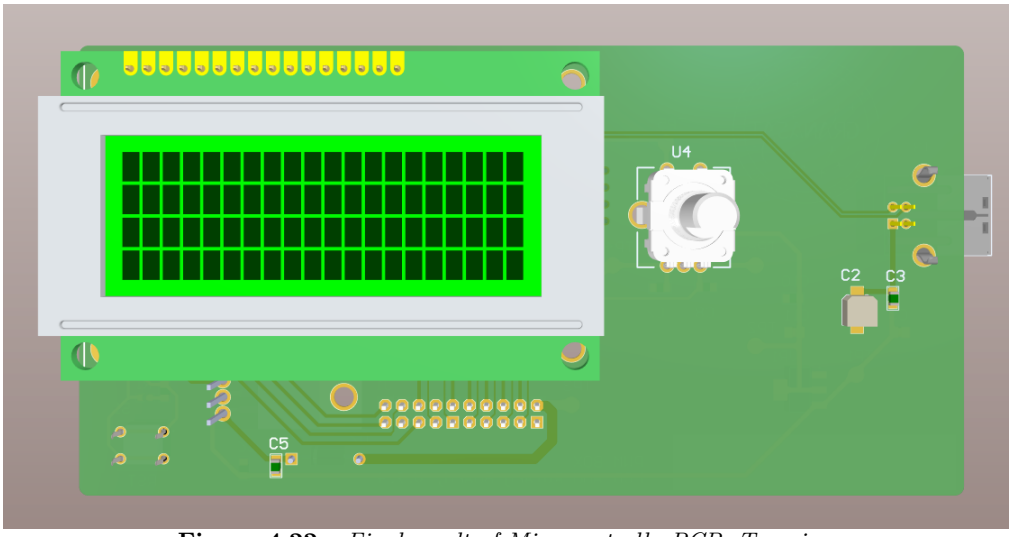


Figure 4.22 – *Final result of MicrocontrollerPCB. Top view*

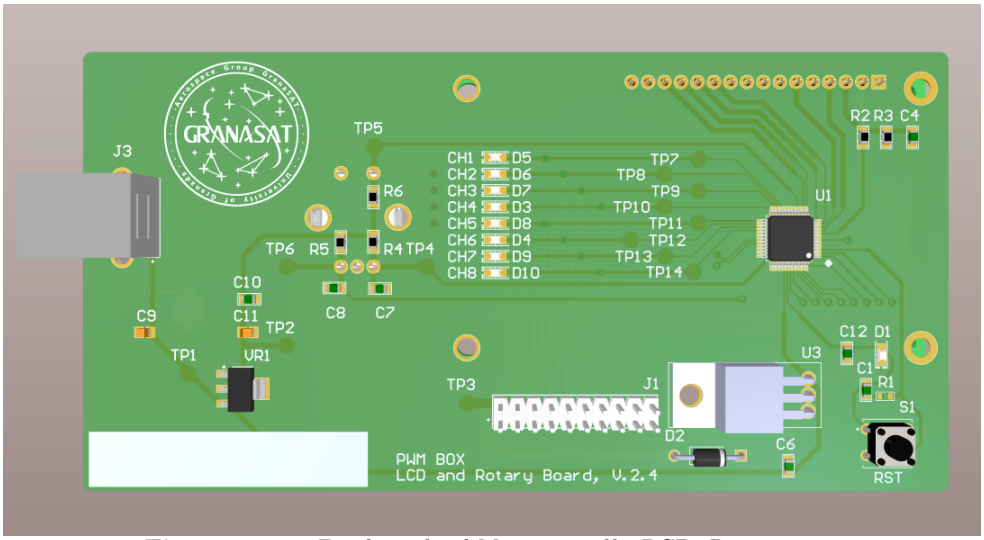


Figure 4.23 – *Final result of MicrocontrollerPCB. Bottom view*

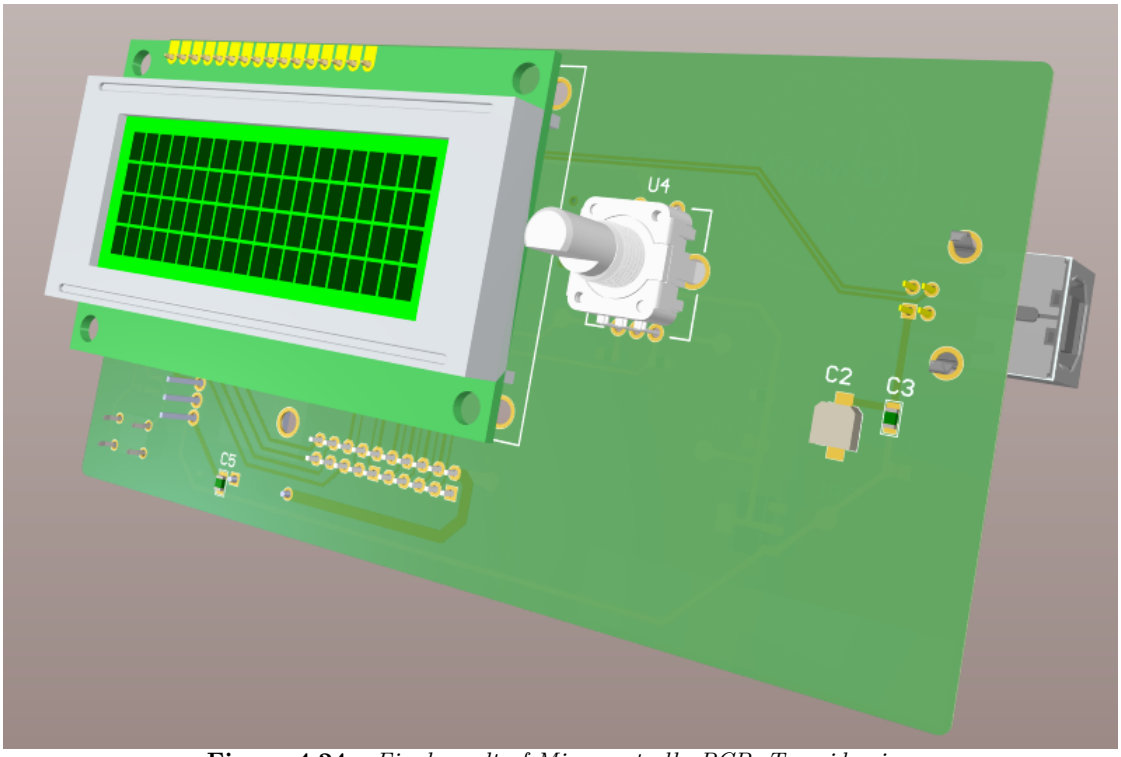


Figure 4.24 – *Final result of MicrocontrollerPCB. Top-side view*

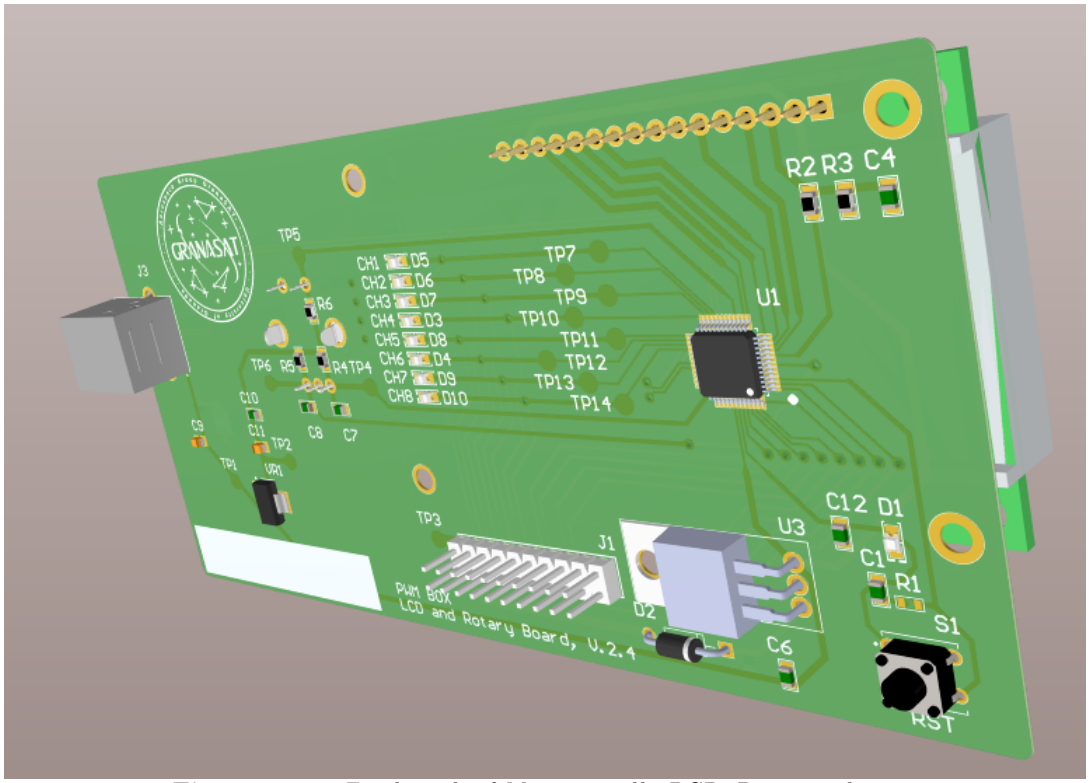


Figure 4.25 – *Final result of MicrocontrollerPCB. Bottom-side view*

4.2.1.9 Others documents

Other important project documents are attached in this section.These are the following documents in the order in which they are listed:

Top assembly: Top layer, top overlay, multi-layer, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Bottom assembly: Bottom layer, bottom overlay, multi-layer, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Drill drawing: Drill drawing, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Drill guide: Drill guide, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Top PCB prints: Top overlay, multi-layer, mechanical 1, border outline, top assembly, mechanical 15,top layer y template.

Bottom PCB prints: Bottom overlay, multi-layer, mechanical 1, border outline, bottom assembly, mechanical 15, bottom layer y template.

It should be noted that the GND drawing is not shown as it obscured all other information.

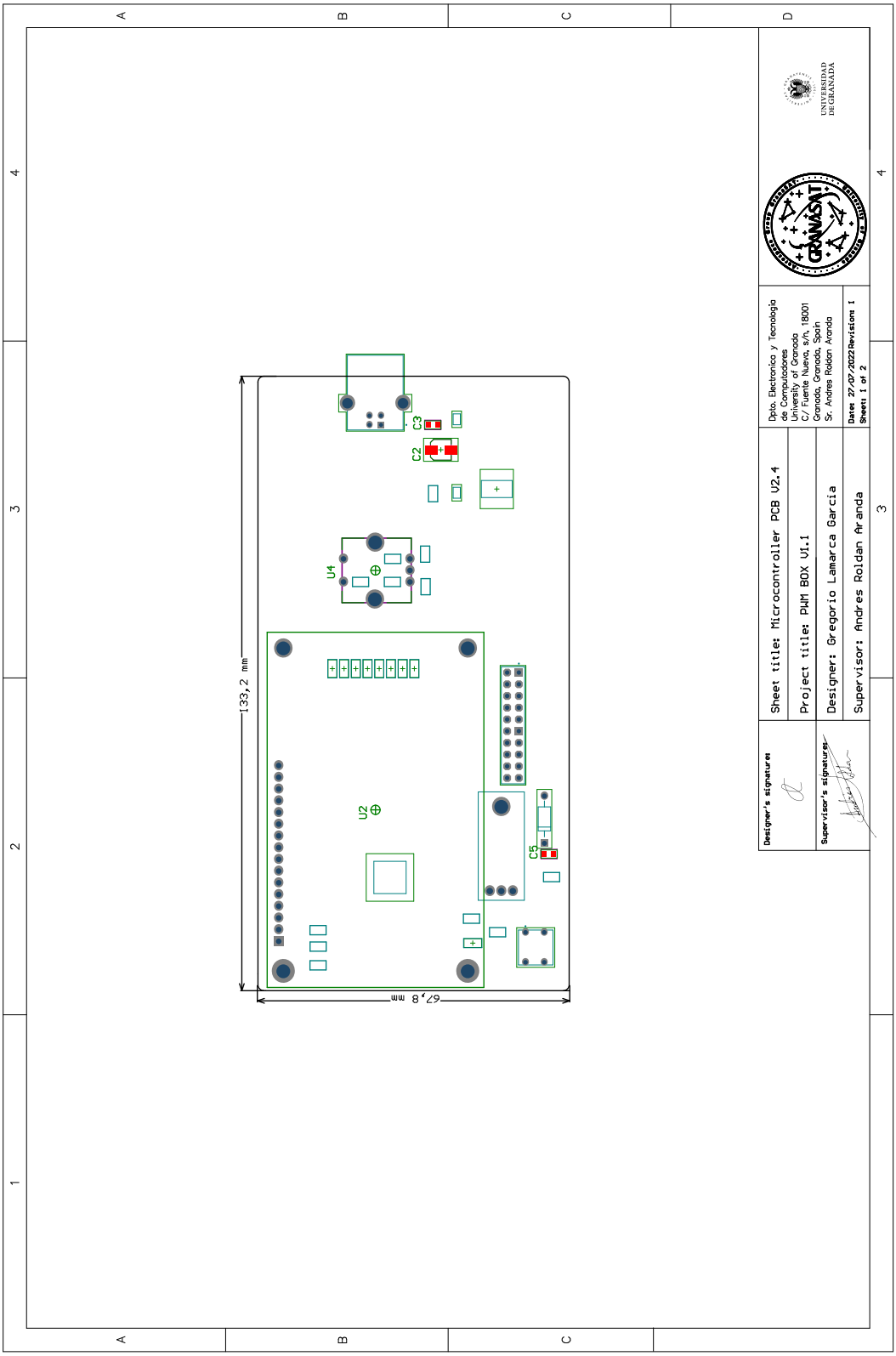


Figure 4.26 – *Top assembly in Microcontroller PCB project*

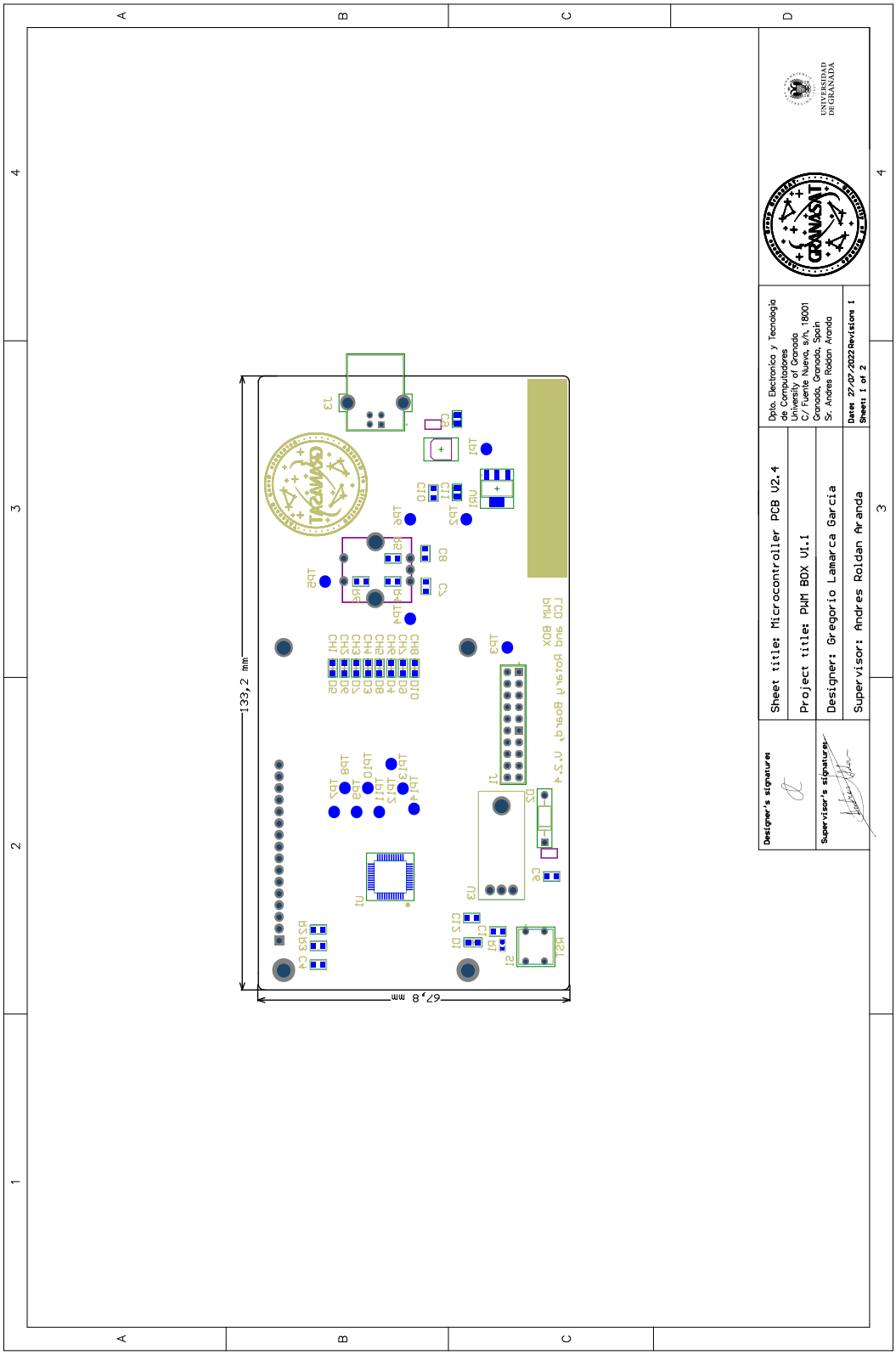


Figure 4.27 – *Bottom assembly in Microcontroller PCB project*

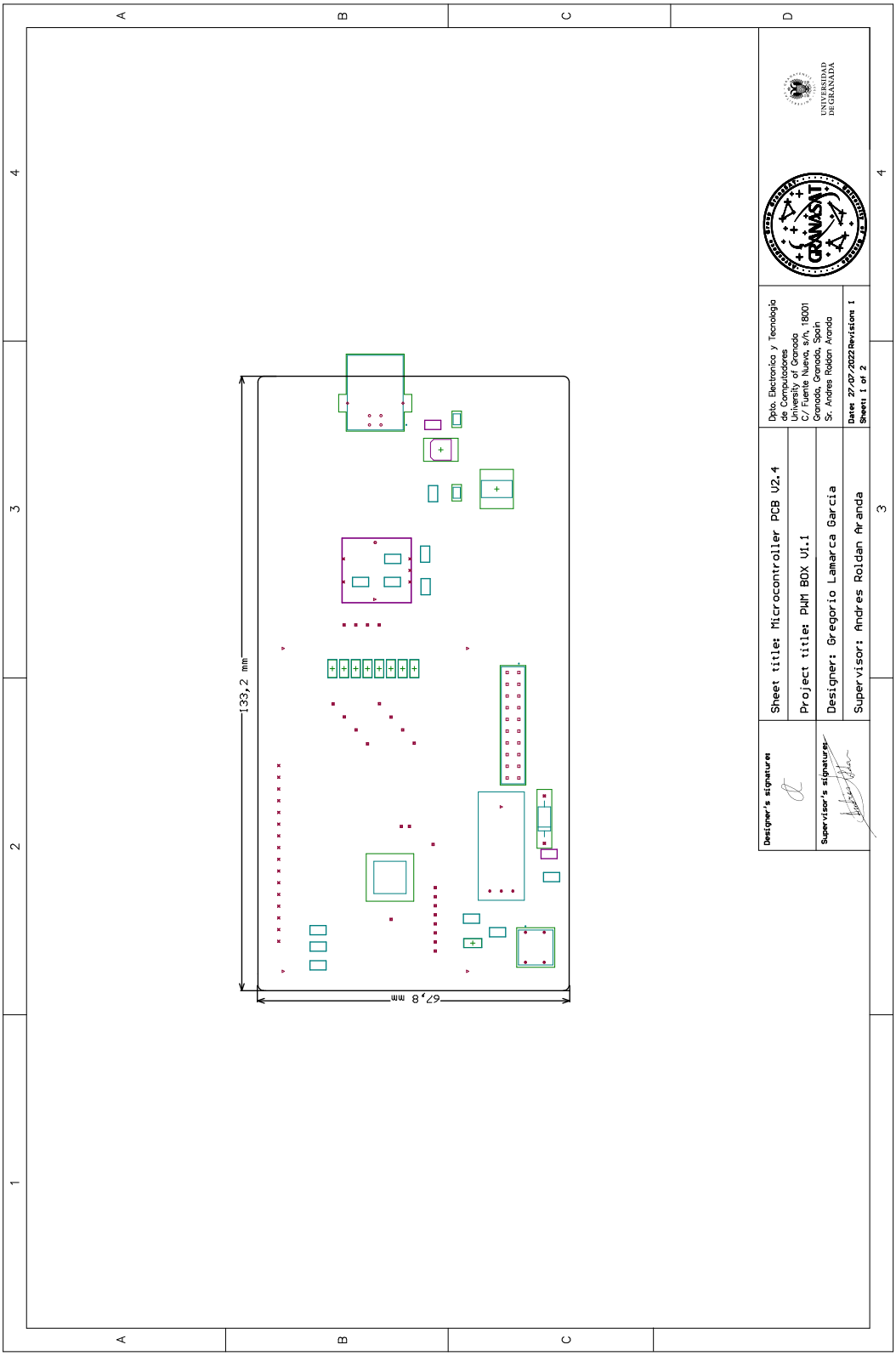


Figure 4.28 – *Drill drawing in Microcontroller PCB project*

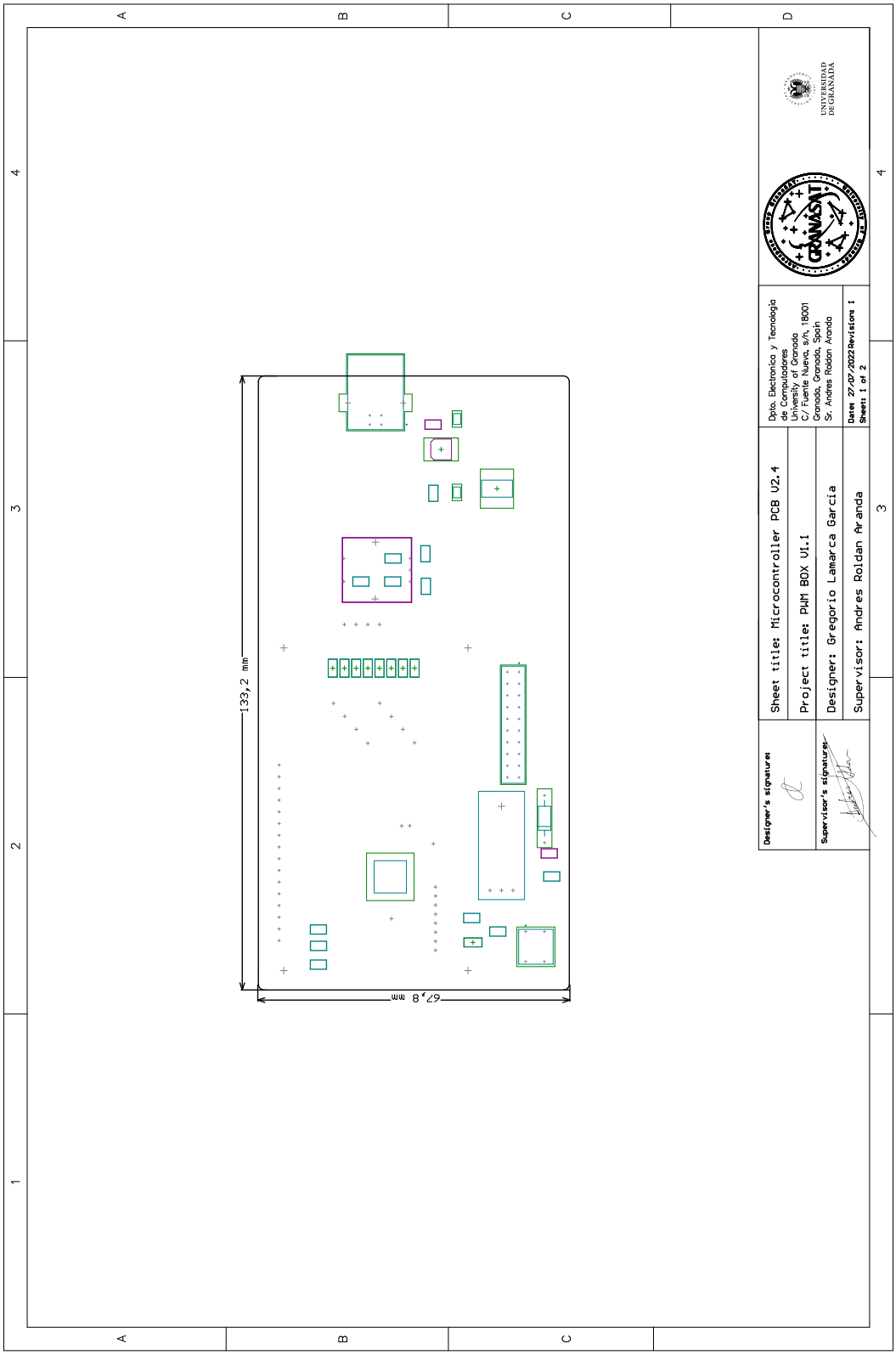


Figure 4.29 – *Drill guides in Microcontroller PCB project*

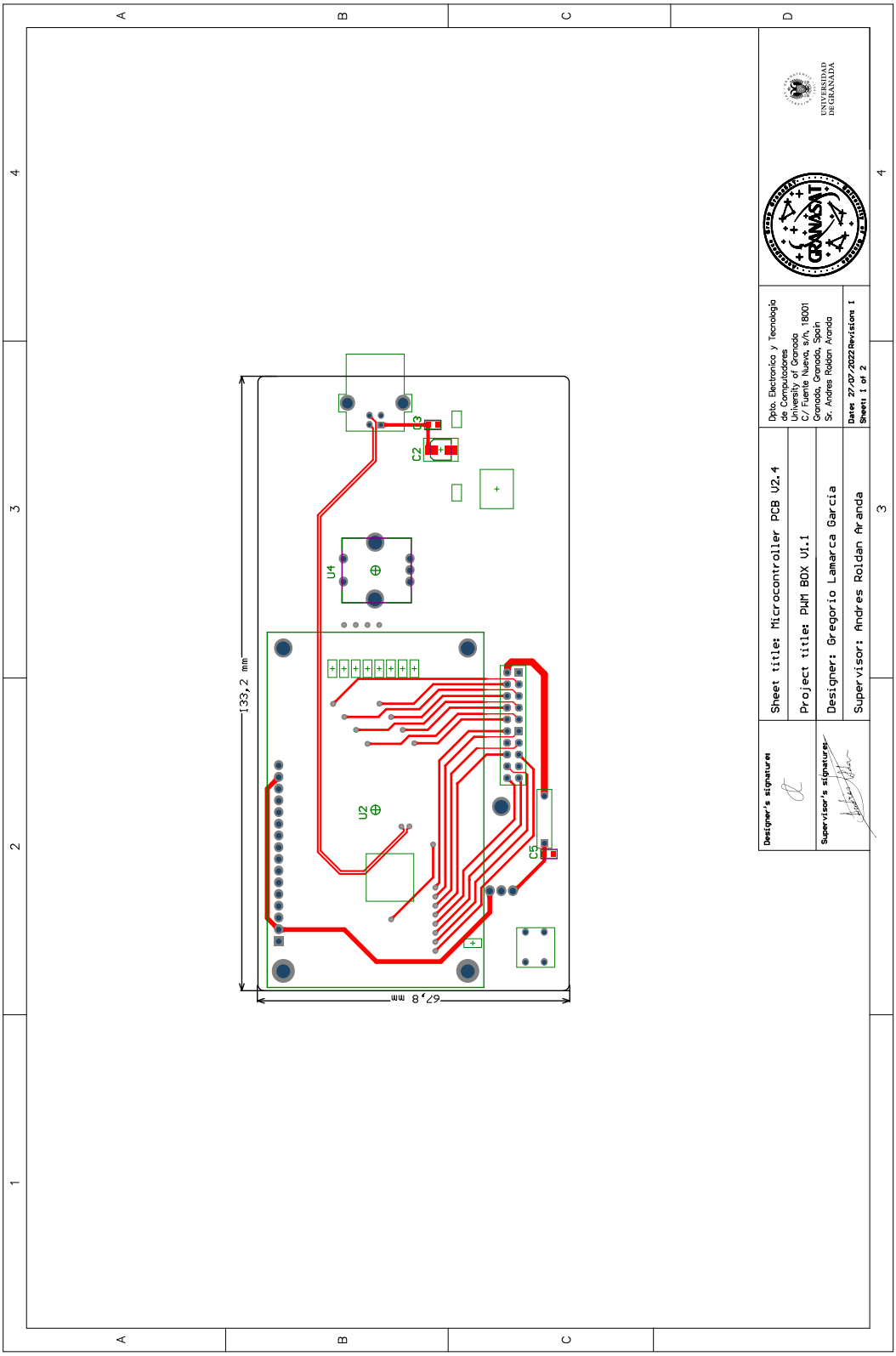


Figure 4.30 – *Top PCB prints in Microcontroller PCB project*

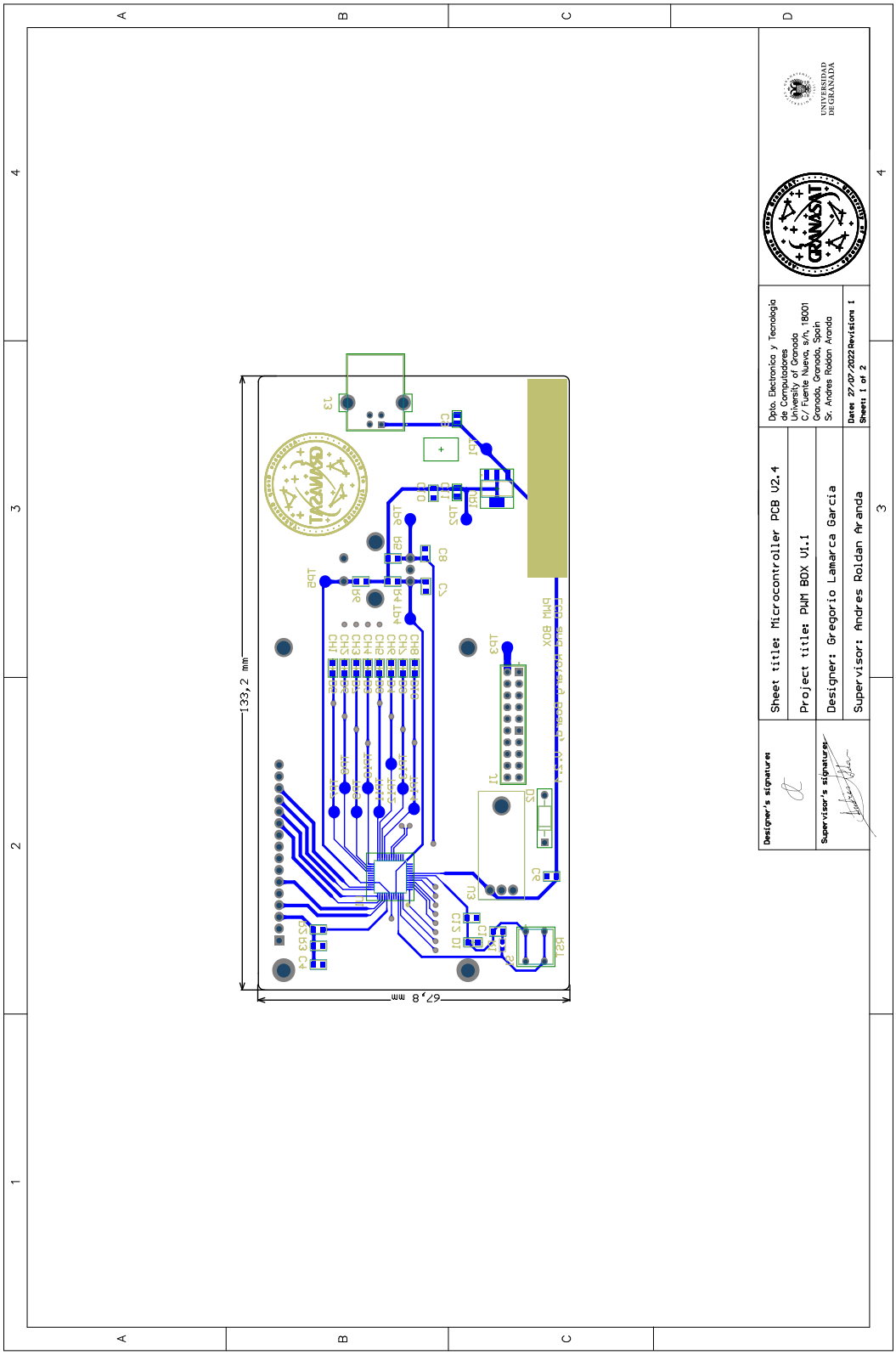


Figure 4.31 – *Bottom PCB prints in Microcontroller PCB project*

4.2.2 ConnectorsPCB

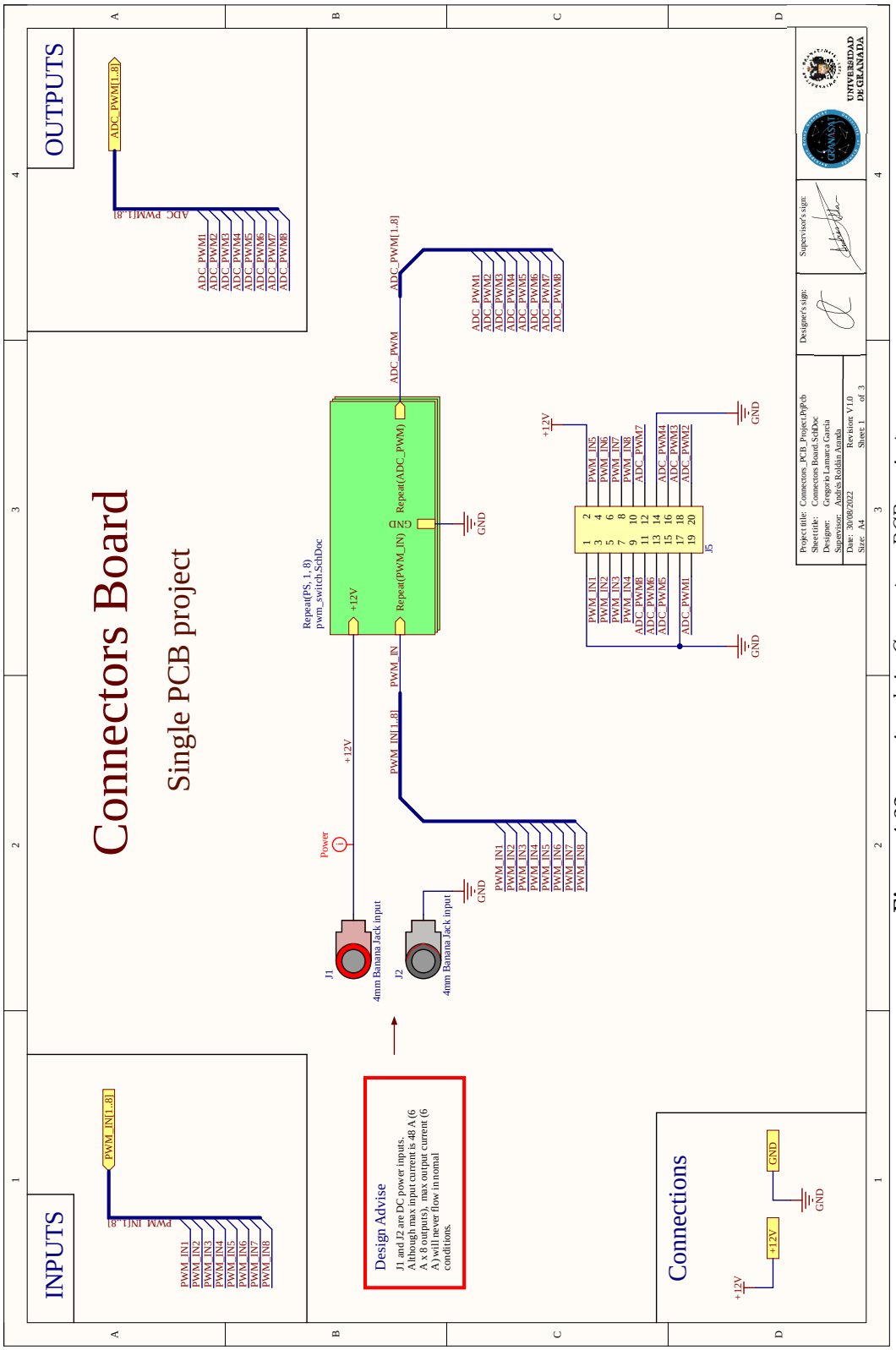


Figure 4.32 – *main.sch in Connectors PCB project*

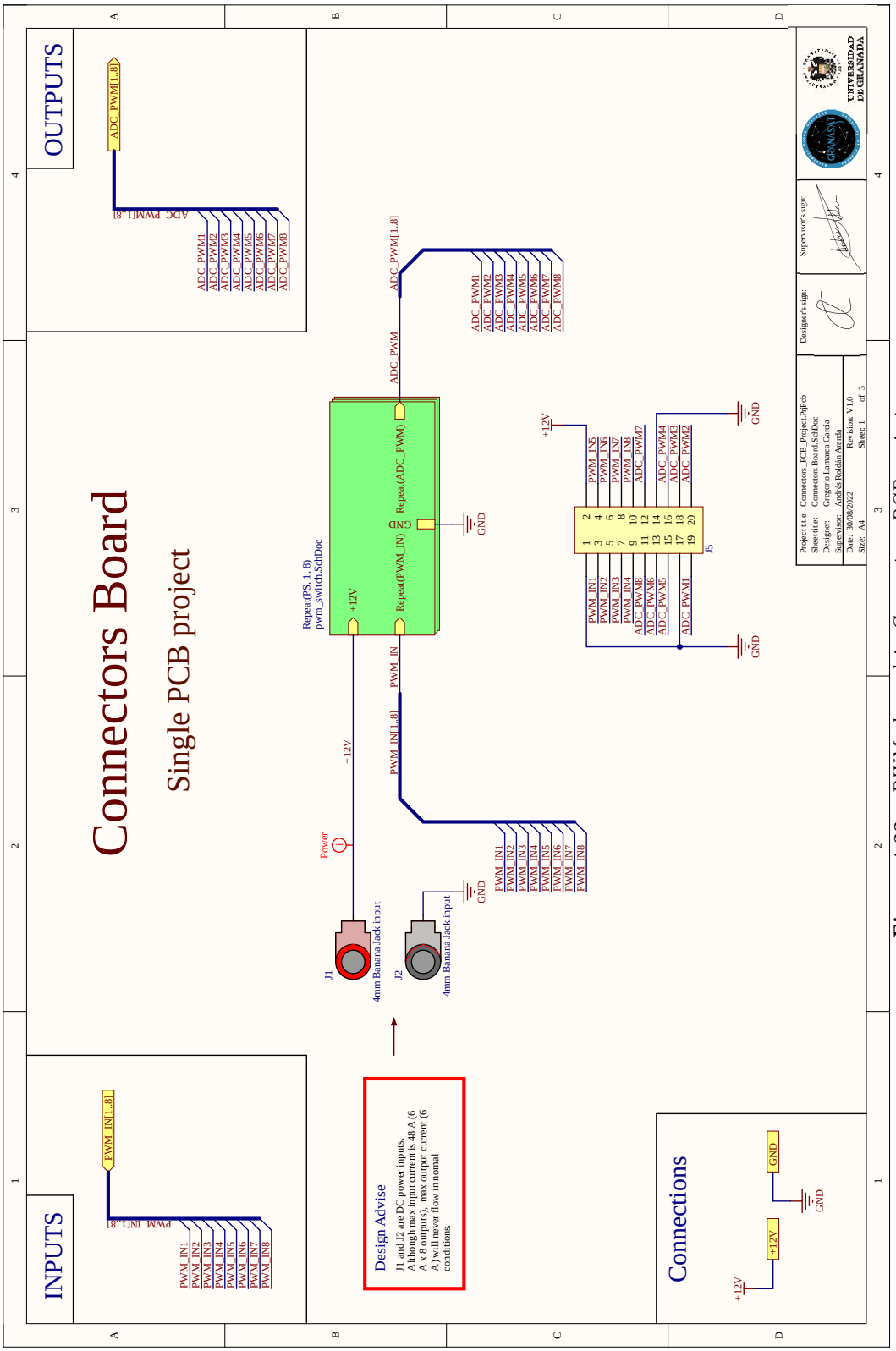


Figure 4.33 – *PWM channel in Connectors PCB project*

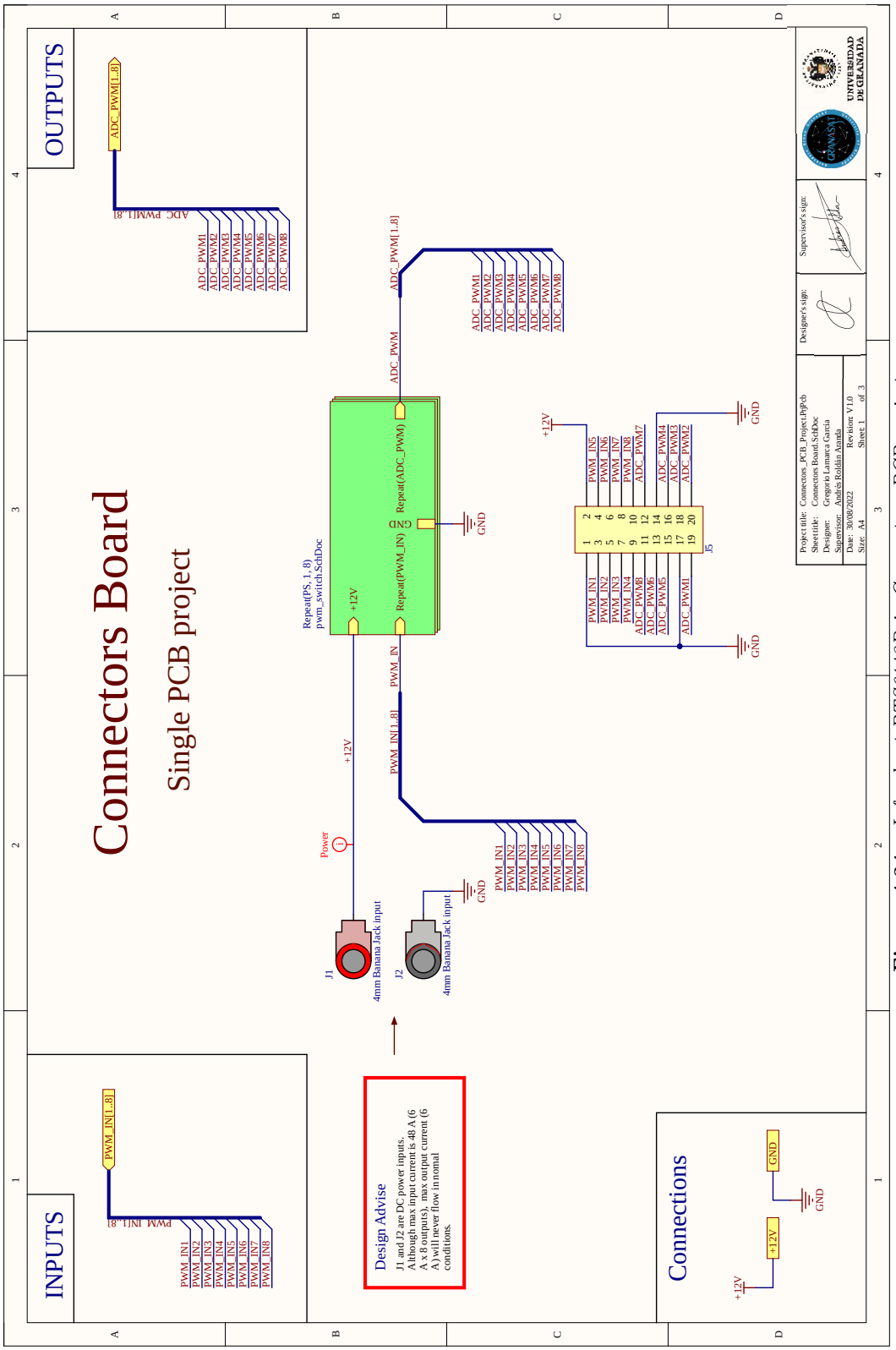


Figure 4.34 – *Info about BTS6143D in Connectors PCB project*

4.2.2.1 Pin header

The pin header in ConnectorsPCB [4.32] connects the ConnectorsPCB and the MicrocontrollerPCB. It is important that both pin headers in the boards have the same connections so that when connecting there is no error that could be detrimental. The price of the component is 0.86 €.

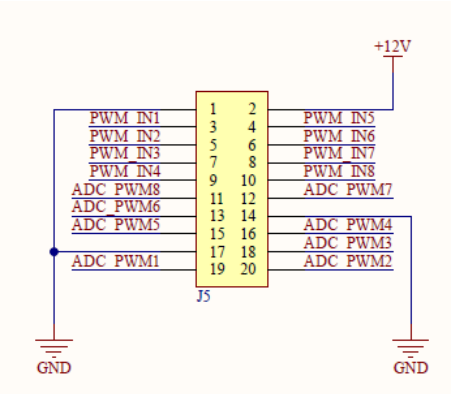


Figure 4.35 – *Pin header in ConnectorsPCB*

The connections that are shared with the other Printed Circuit Board (PCB) and the power supply are shown below:

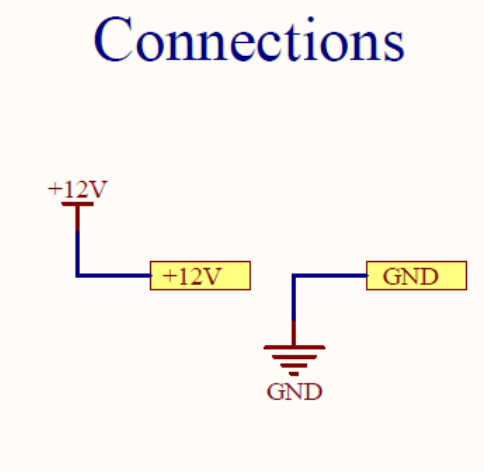


Figure 4.36 – *Power supply in ConnectorsPCB*

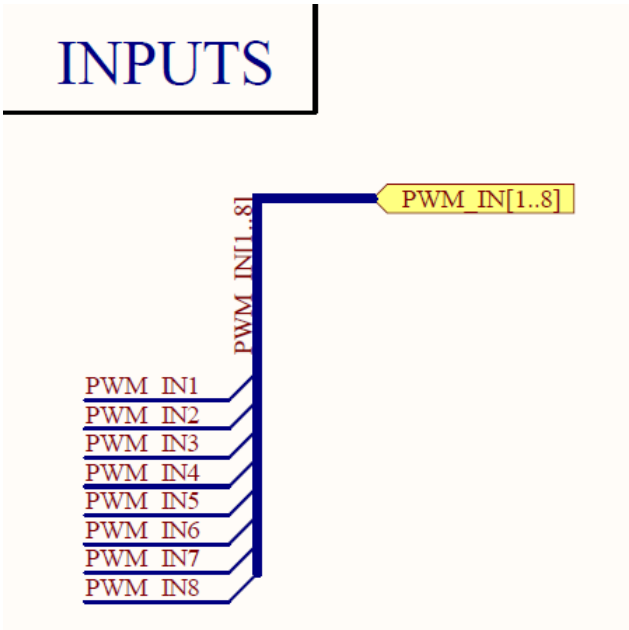


Figure 4.37 – *PWM channels in ConnectorsPCB*

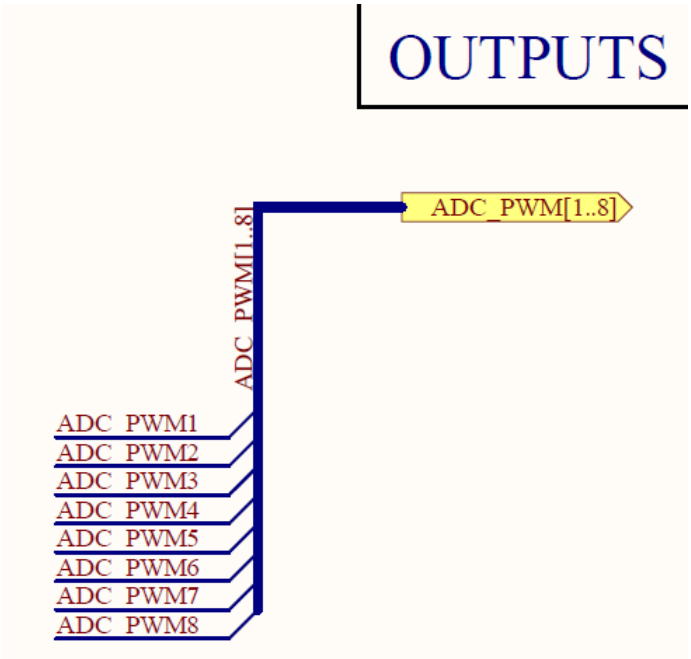


Figure 4.38 – *ADC channels in ConnectorsPCB*

The system power supply (12 V and GND) is obtained by connecting a voltage source to connectors J1 and J2 which are shown in the figure below:

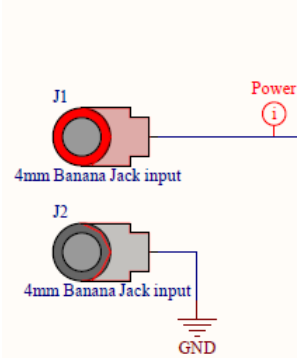


Figure 4.39 – *Power Supply connectors in ConnectorsPCB*

4.2.2.2 PWM channels

The Pulse-Width Modulation (PWM) channels are the main modules of this Printed Circuit Board (PCB) and can be seen in the schematic [4.33], which has a structure repeated 8 times for the 8 channels. It has PWM-IN connections as inputs and ADC-PWM connections as outputs, these channels being powered from connectors J1 and J2 [4.39]. Each connector of this type is priced at 1.52 € and each channel has 2 apart from the 2 power connectors.

Starting with the PWM-IN network, there is an LED to visually check if the signal is correct, which causes the input pin (pin 2) of the BTS6143D to drive and thus the output (pins 1 and 5) to have 12 V. It has a protection diode in case of reverse polarity in the power supply. Once the input pin of the BTS6143D has been activated as a reaction for activating the PWM-IN network, the output has a capacitor to stabilise the voltage as it is switched and also has a pull-down resistor so that in the event that there is no load (2 Ω at least) the consumption is controlled.

The minimum load is 2 Ω as the maximum output current is 6 A, which is the maximum current for which the current sense pin (pin 4) of the BTS6143D has been configured.

The datasheet of the BTS6143D is shown below:

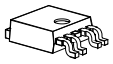
Smart Highside Power Switch

Reversave™
- Reverse battery protection by self turn on of power MOSFET

Features
- Short circuit protection with latch
- Current limitation
- Overload protection
- Thermal shutdown with restart
- Overvoltage protection (including load dump)
- Loss of ground protection
- Loss of V_{bb} protection (with external diode for charged inductive loads)
- Very low standby current
- Fast demagnetisation of inductive loads
- **E**lectro**s**tatic **d**ischarge (**ESD**) protection
- Optimized static **e**lectro**m**agnetic **c**ompatibility (**EMC**)

Product Summary			
Operating voltage	$V_{bb(on)}$	5.5 ... 38	V
On-state resistance	R_{ON}	10	mΩ
Nominal current	$I_{L(nom)}$	8	A
Load current (ISO)	$I_{L(ISO)}$	33	A
Current limitation	$I_{L12(SC)}$	75	A

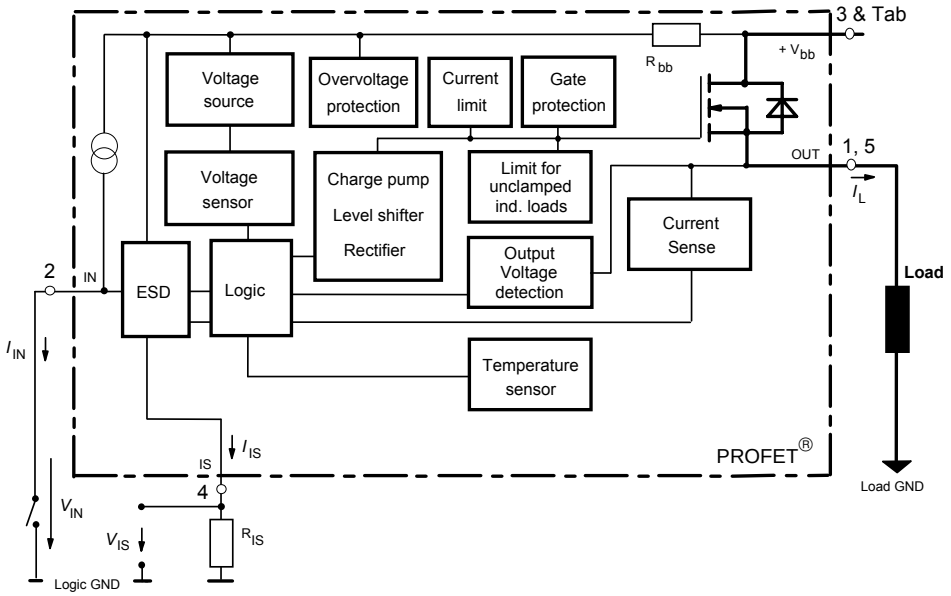
Package
TO-252-5-1
(DPAK 5 pin; less than half the size as TO 220 SMD)



Diagnostic Function
- Proportional load current sense (with defined fault signal in case of overload operation, overtemperature shutdown and/or short circuit shutdown)

Application
- Power switch with current sense diagnostic feedback for 12V and 24 V DC grounded loads
- All types of resistive, inductive and capacitive loads
- Replaces electromechanical relays, fuses and discrete circuits

General Description
N channel vertical power FET with charge pump, current controlled input and diagnostic feedback with load current sense, integrated in Smart SIPMOS® chip on chip technology. Providing embedded protective functions.



Pin	Symbol		Function
1	OUT	O	**Output;** output to the load; pin 1 and 5 must be externally shorted* .
2	IN	I	**Input;** activates the power switch if shorted to ground.
Tab/(3)	V_{bb}	+	**Supply Voltage;** positive power supply voltage; tab and pin3 are internally shorted.
4	IS	S	**Sense Output;** Diagnostic feedback; provides at normal operation a sense current proportional to the load current; in case of overload, overtemperature and/or short circuit a defined current is provided (see Truth Table on page 8)
5	OUT	O	**Output;** output to the load; pin 1 and 5 must be externally shorted* .

*) Not shorting all outputs will considerably increase the on-state resistance, reduce the peak current capability and decrease the current sense accuracy

Maximum Ratings at T_j = 25 °C unless otherwise specified

Parameter	Symbol	Values	Unit
Supply voltage (overvoltage protection see page 4)	V_{bb}	38	V
Supply voltage for full short circuit protection [1]	V_{bb}	30	V
Load dump protection $V_{LoadDump} = U_A + V_s$, U_A = 13.5 V R_I= 2 Ω, R_L= 1.5 Ω, t_d= 400 ms, IN= low or high	$V_{Load\ dump}$ [2]	45	V
Load current (Short-circuit current, see page 5)	I_L	self-limited	A
Operating temperature range Storage temperature range	T_j T_{stg}	-40 ...+150 -55 ...+150	°C
Power dissipation (DC)	P_{tot}	59	W
Inductive load switch-off energy dissipation [3] single pulse I_L = 20 A, V_{bb}= 12V T_j=150 °C:	E_{AS}	0.3	J
Electrostatic discharge capability (ESD) (Human Body Model) acc. ESD assn. std. S5.1-1993; R=1.5kΩ; C=100pF	V_{ESD}	3.0	kV
Current through input pin (DC) Current through current sense pin (DC) see internal circuit diagrams page 9	I_{IN} I_{IS}	+15, -120 +15, -120	mA
Input voltage slew rate $V_{bb} \le$ 16V : V_{bb} > 16V [4]:	dV_{bIN} / dt	self-limited 20	V/µs

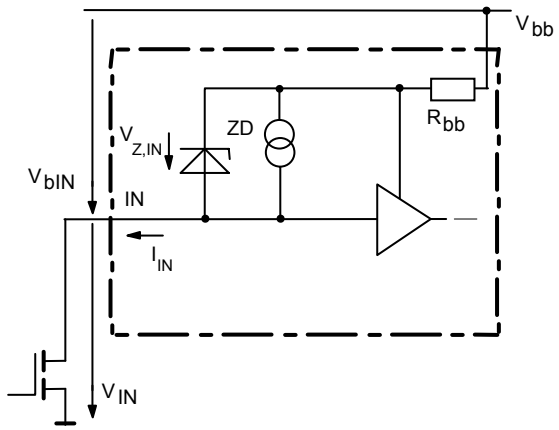
1) Short circuit is defined as a combination of remaining resistances and inductances. See schematic on page11.

2) $V_{Load\ dump}$ is setup without the DUT connected to the generator per ISO 7637-1 and DIN 40839

3) See also diagram on page 11.

4) See also on page 8. Slew rate limitation can be achieved by means of using a series resistor R_{IN} in the input path. This resistor is also required for reverse operation. See also page 10.

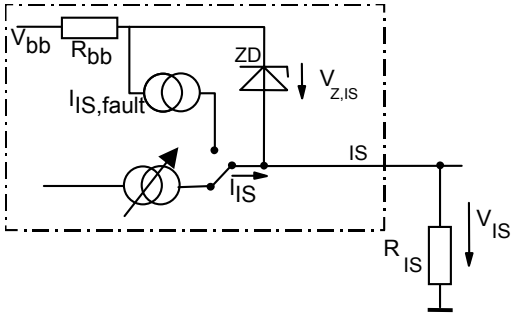
Input circuit (ESD protection)



ESD-Zener diode: 67 V typ., max 15 mA;

Current sense output

Normal operation

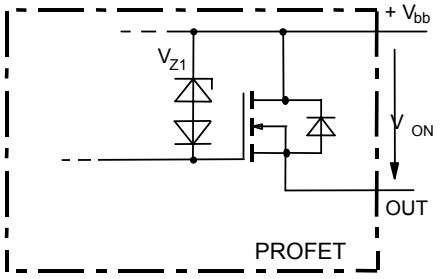


$V_{Z,IS}$ = 67 V (typ.), R_{IS} = 1 kΩ nominal (or 1 kΩ /n, if n devices are connected in parallel). $I_S = I_L/k_{ilis}$ can be only driven by the internal circuit as long as V_{out} - V_{IS} > 5V. Therefore R_{IS} should be less than

$$\frac{V_{bb} - 5V}{7.5mA}.$$

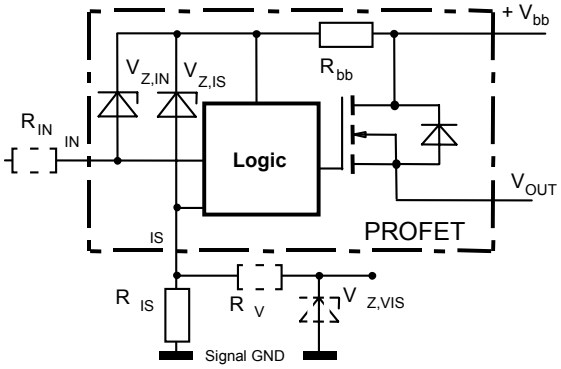
Note: *For large values of R_{IS} the voltage V_{IS} can reach almost V_{bb}. See also overvoltage protection.*
If you don't use the current sense output in your application, you can leave it open.

Inductive and overvoltage output clamp



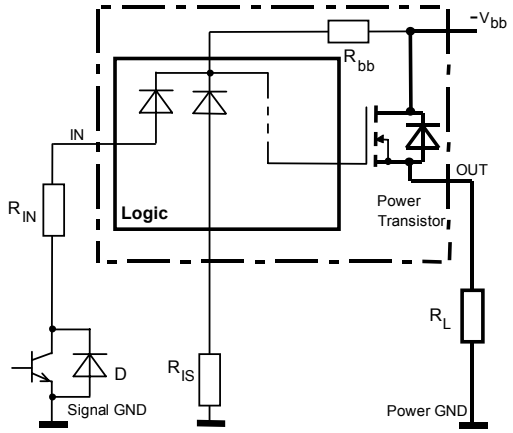
V_{ON} is clamped to $V_{ON(Cl)}$ = 42 V typ

Overvoltage protection of logic part



R_{bb} = 100 Ω typ., $V_{Z,IN}$ = $V_{Z,IS}$ = 67 V typ., R_{IS} = 1 kΩ nominal. Note that when overvoltage exceeds 67 V typ. a voltage above 5V can occur between IS and GND, if R_V, $V_{Z,VIS}$ are not used.

Reversave™ (Reverse battery protection)

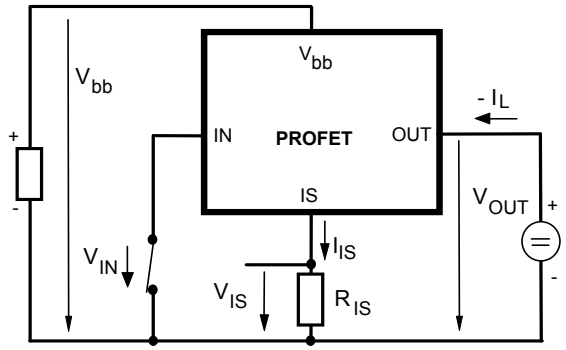


R_{IS} typ. 1 kΩ. Add R_{IN} for reverse battery protection in applications with V_{bb} above 16V;

recommended value: $\frac{1}{R_{IN}} + \frac{1}{R_{IS}} = \frac{0.08A}{|V_{bb}| - 12V}$

To minimise power dissipation at reverse battery operation, the overall current into the IN and IS pin should be about 80mA. The current can be provided by using a small signal diode D in parallel to the input switch, by using a MOSFET input switch or by proper adjusting the current through R_{IS}.
Since the current via R_{bb} generates additional heat in the device, this has to be taken into account in the overall thermal consideration.

Inverse load current operation

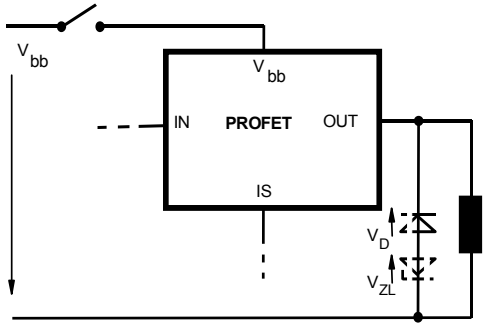


The device can be operated in inverse load current mode (V_{OUT} > V_{bb} > 0V). The current sense feature is not available during this kind of operation (I_{IS} = 0). In case of inverse operation the intrinsic drain source diode is eventually conducting resulting in considerably increased power dissipation.
The transition from inverse to forward mode can result in a delayed switch on.
Note: *Temperature protection during inverse load current operation is not possible!*

V_{bb} disconnect with energised inductive load

Provide a current path with load current capability by using a diode, a Z-diode, or a varistor. (V_{ZL}+V_D<39 V if R_{IN} = 0). For higher clamp voltages currents at IN and IS have to be limited to 120 mA.

Version a:



After selecting the resistor in subsection [3.2.3.7], pin 4 with the voltage for subsequent reading on the MicrocontrollerPCB is obtained as the output of the channel.

The price per channel is 6.42 (2.02€ for the BTS6143D, 2 x 1.52€ for the connectors, 4 x 0.18€ for the resistors, 0.12€ for the capacitor, 0.36€ for the transistor and 0.16€ for the diode).

4.2.2.3 Final result of ConnectorsPCB

This subsection shows how the Printed Circuit Board (PCB) has looked from different angles to appreciate the result as much as possible.

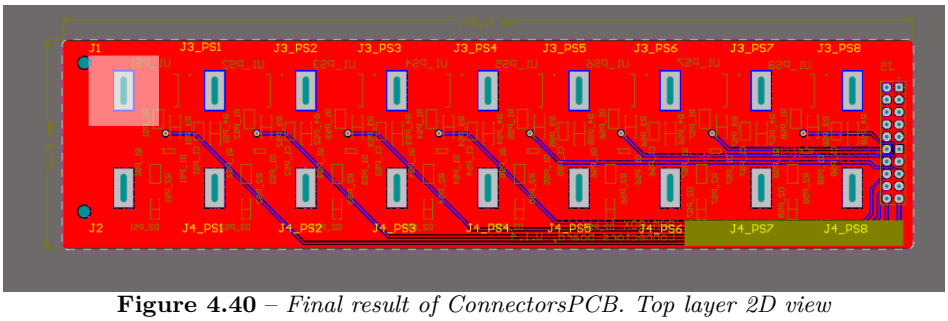


Figure 4.40 – *Final result of ConnectorsPCB. Top layer 2D view*

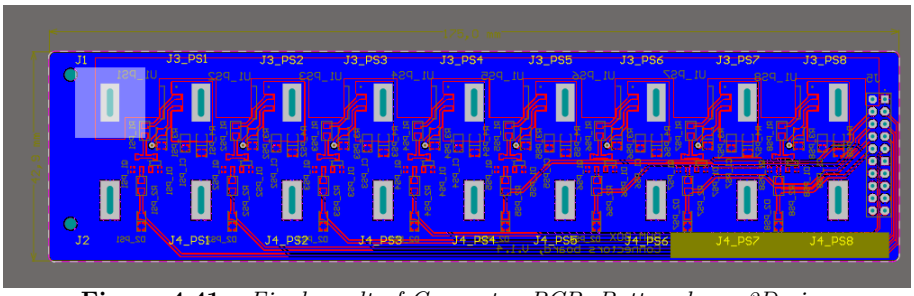


Figure 4.41 – *Final result of ConnectorsPCB. Bottom layer 2D view*

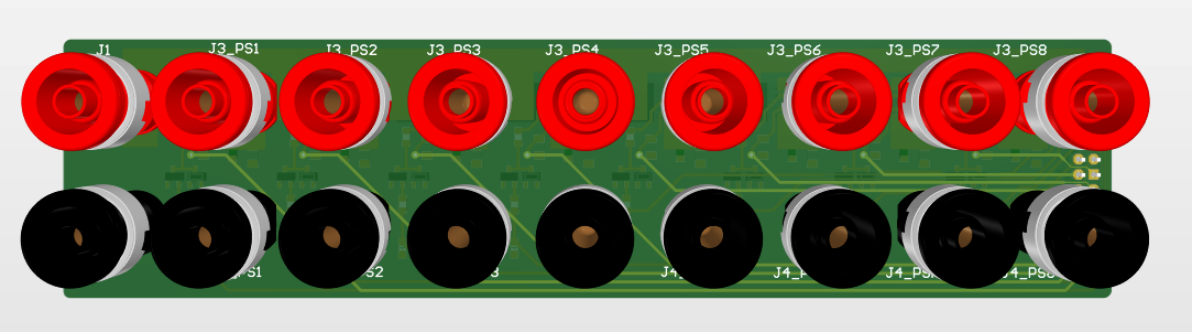


Figure 4.42 – *Final result of ConnectorsPCB. Top view*

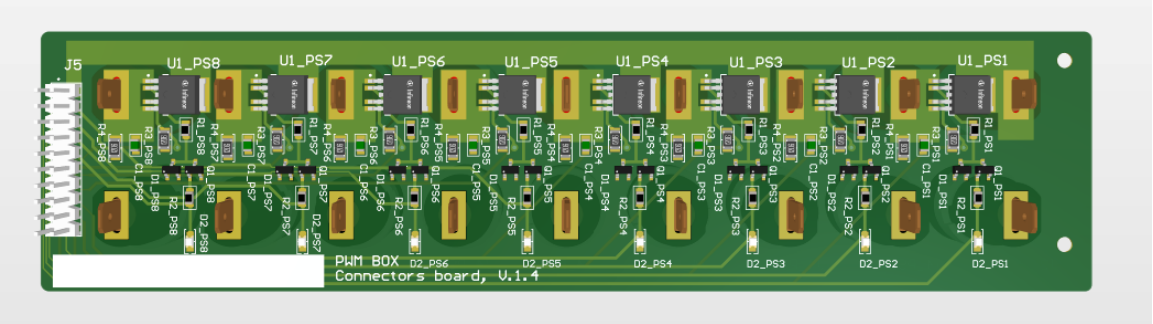


Figure 4.43 – *Final result of ConnectorsPCB. Bottom view*

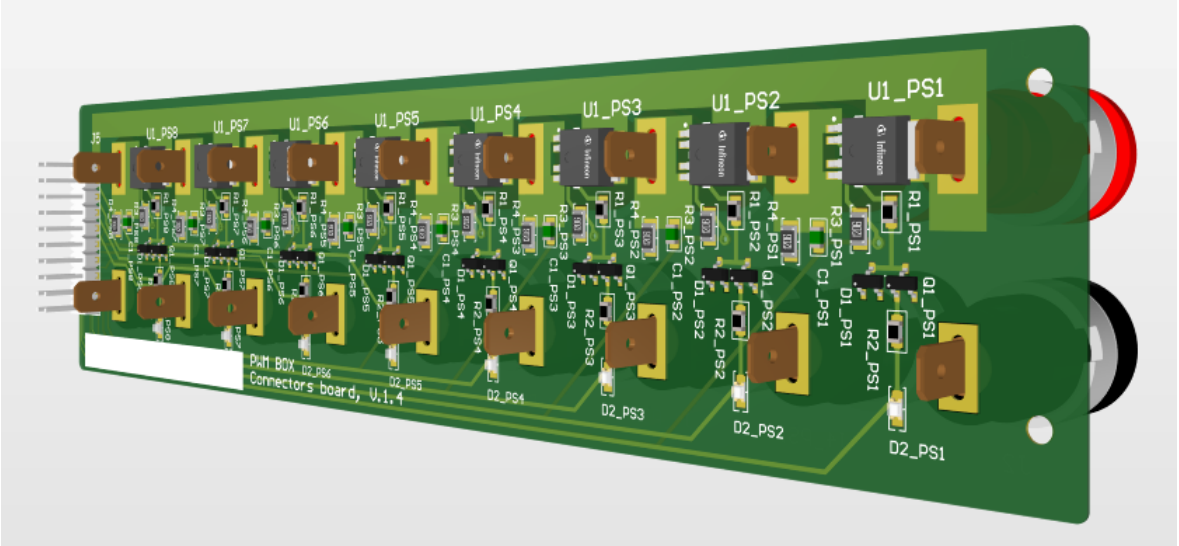


Figure 4.44 – *Final result of ConnectorsPCB. Bottom-side 1 view*

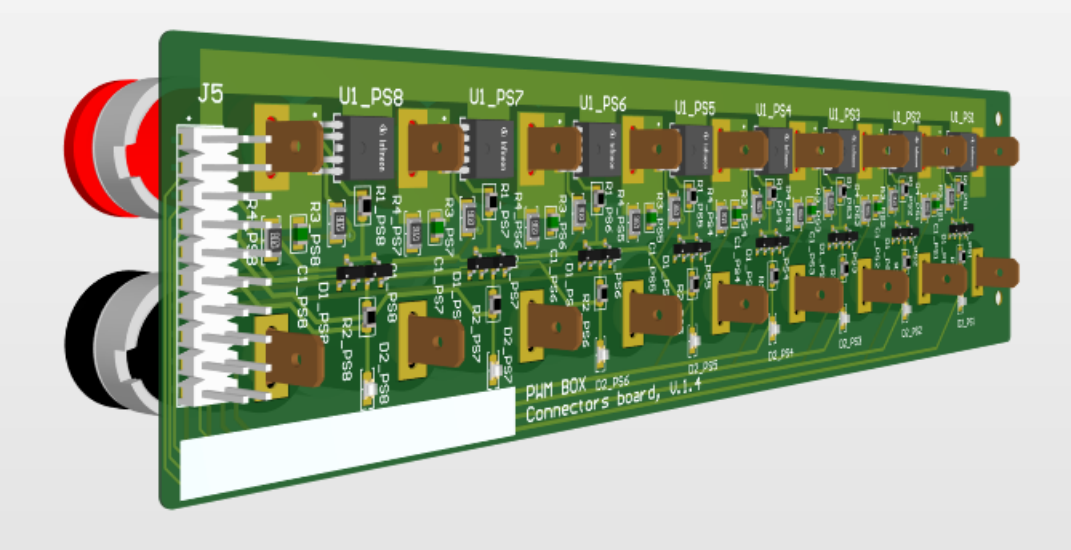


Figure 4.45 – *Final result of MicrocontrollerPCB. Bottom-side 2 view*

4.2.2.4 Others documents

Other important project documents are attached in this section.These are the following documents in the order in which they are listed:

Top assembly: Top layer, top overlay, multi-layer, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Bottom assembly: Bottom layer, bottom overlay, multi-layer, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Drill drawing: Drill drawing, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Drill guide: Drill guide, mechanical 1, border outline, top assembly, bottom assembly, mechanical 15 y template.

Top PCB prints: Top overlay, multi-layer, mechanical 1, border outline, top assembly, mechanical 15,top layer y template.

Bottom PCB prints: Bottom overlay, multi-layer, mechanical 1, border outline, bottom assembly, mechanical 15, bottom layer y template.

It should be noted that the Ground (GND) drawing is not shown as it obscured all other information.

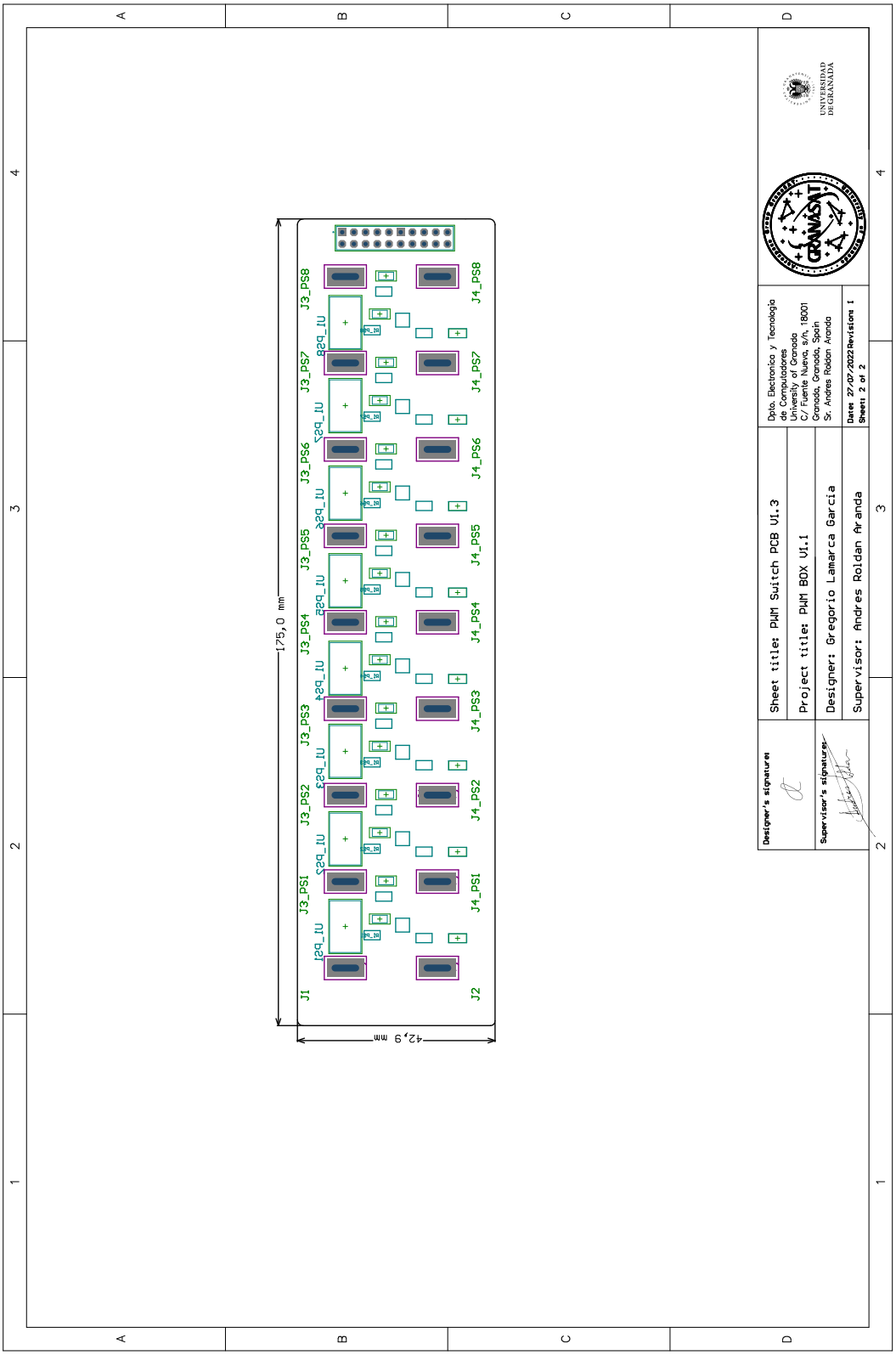


Figure 4.46 – *Top assembly in Connectors PCB project*

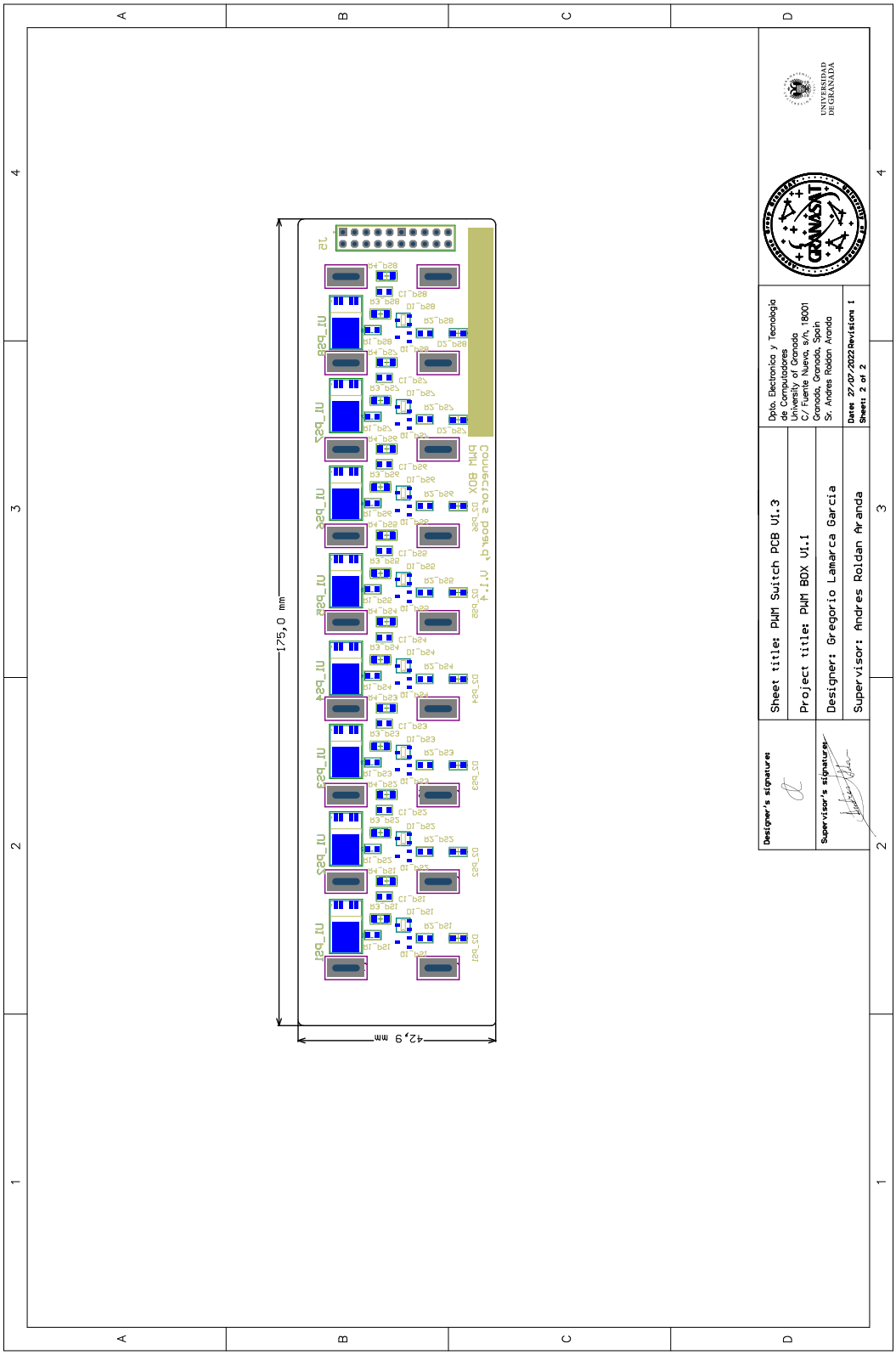


Figure 4.47 – *Bottom assembly in Connectors PCB project*

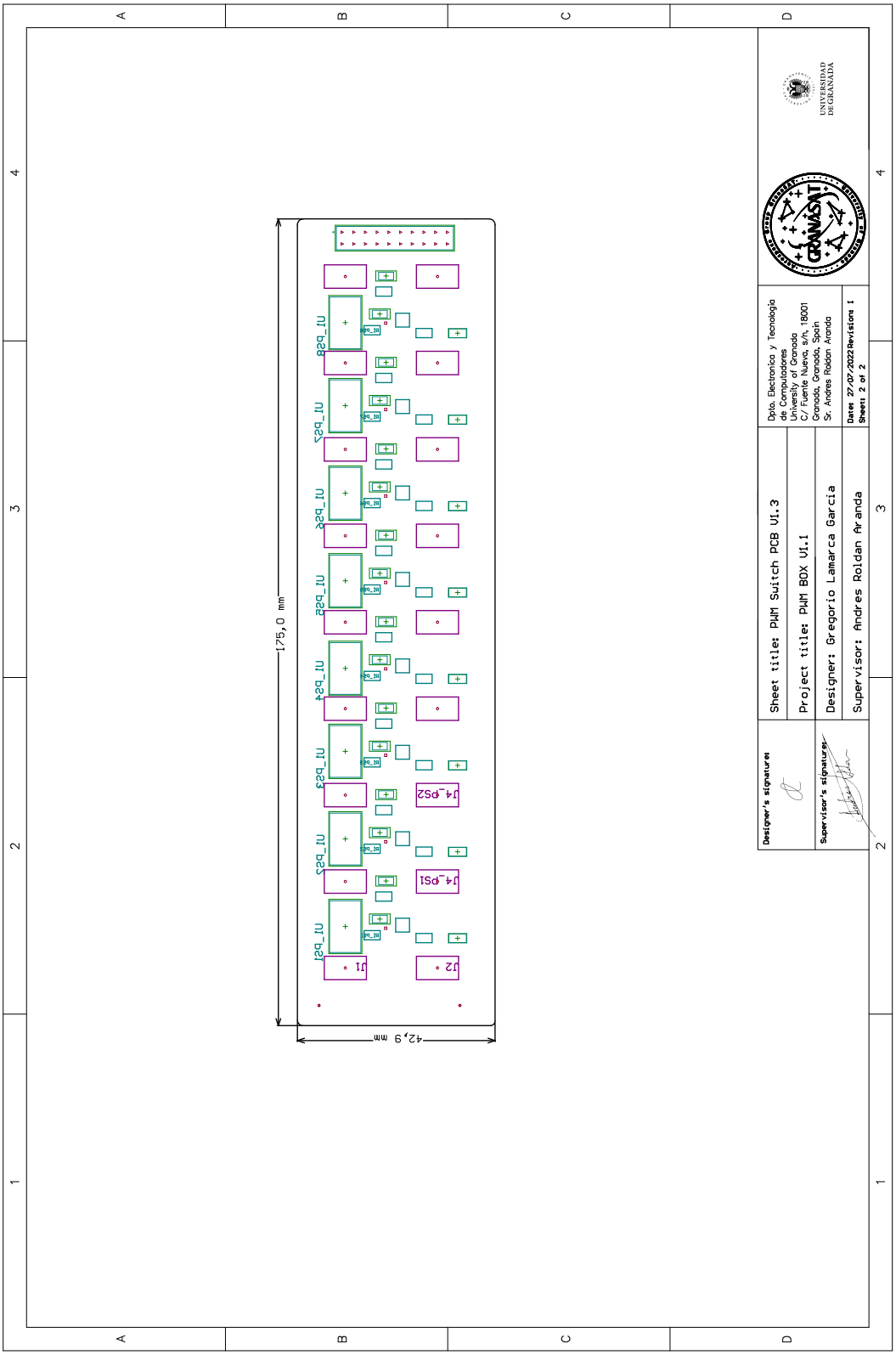


Figure 4.48 – *Drill drawing in Connectors PCB project*

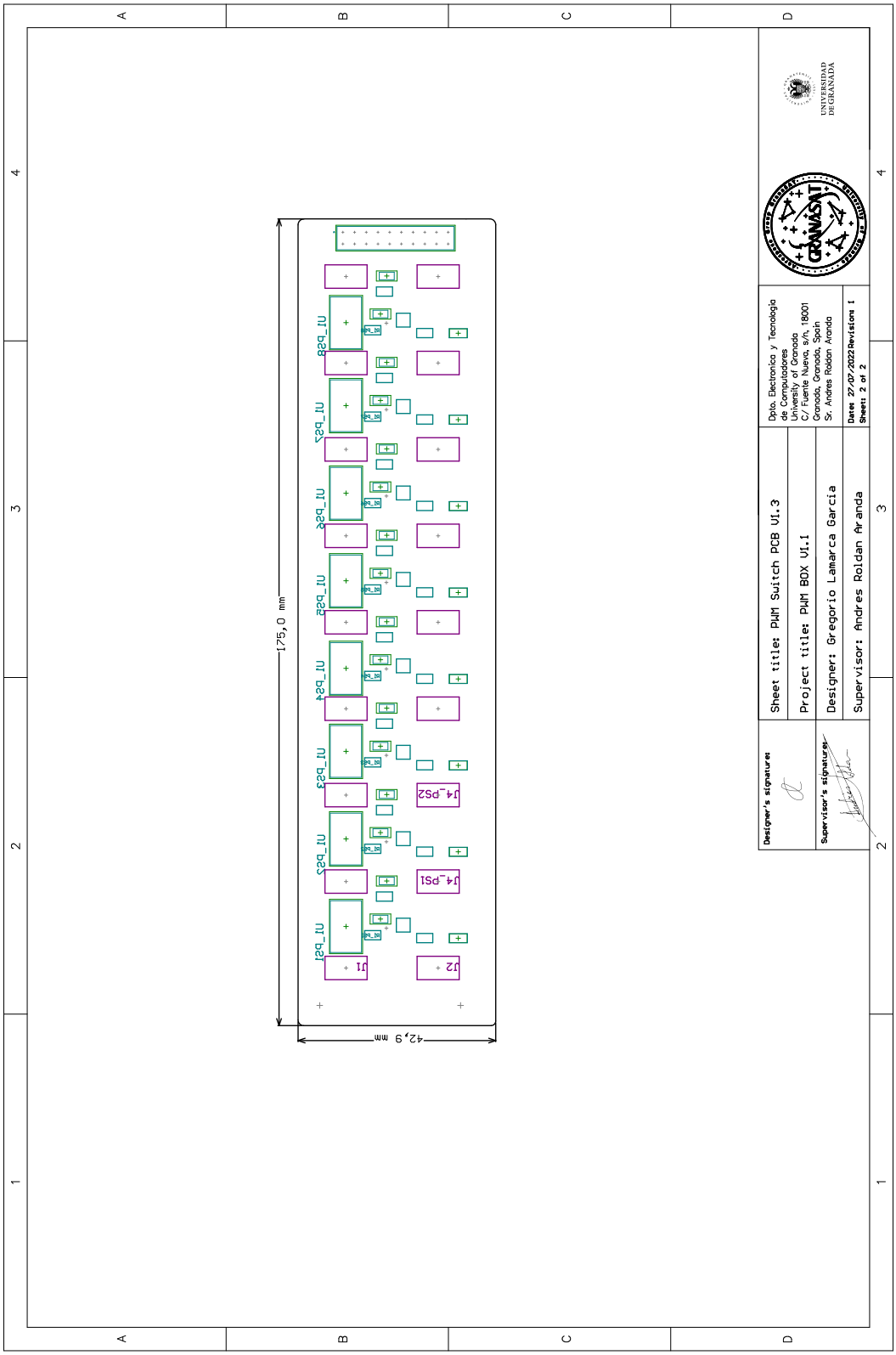


Figure 4.49 – *Drill guides in Connectors PCB project*

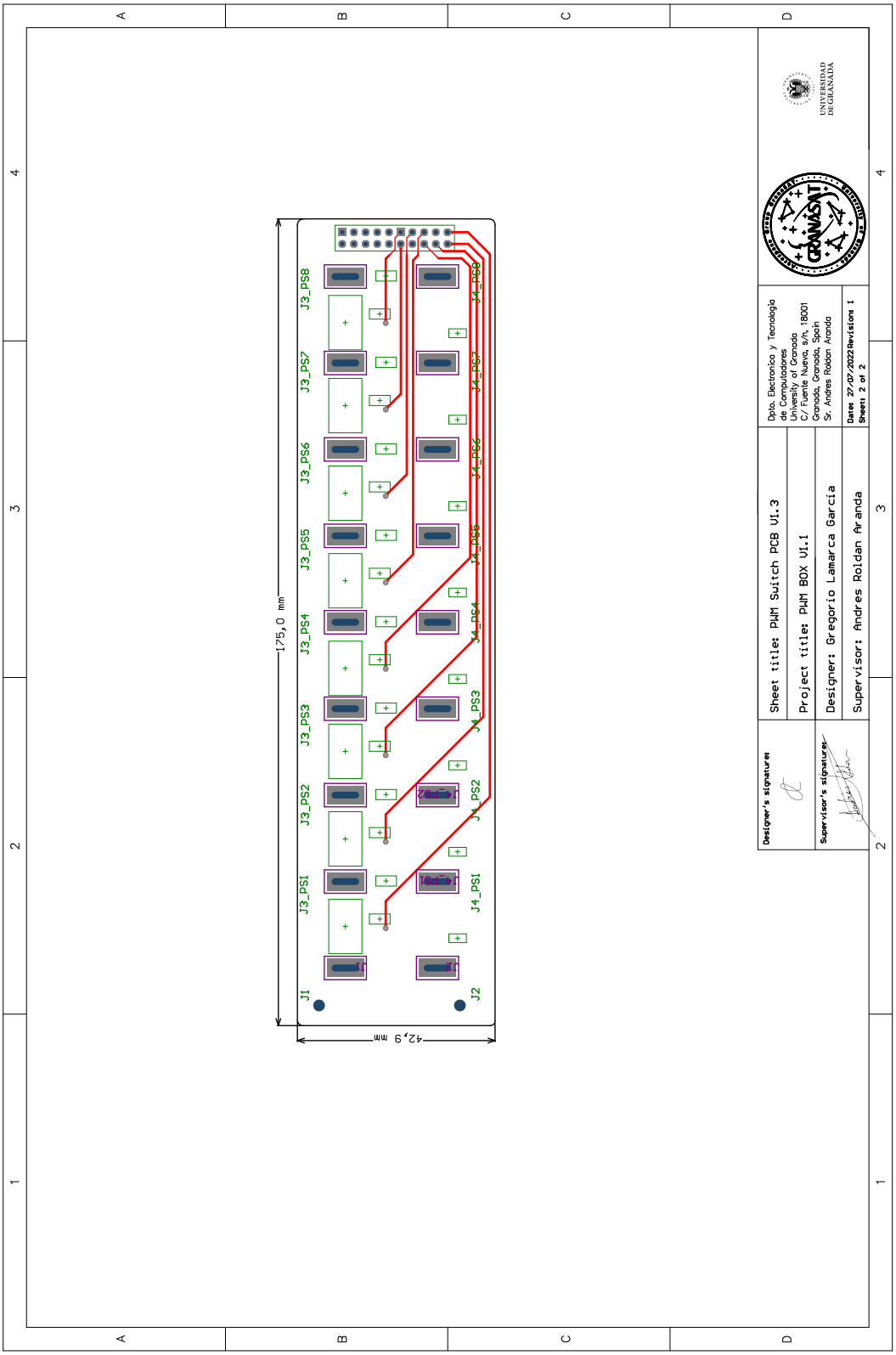


Figure 4.50 – *Top PCB prints in Connectors PCB project*

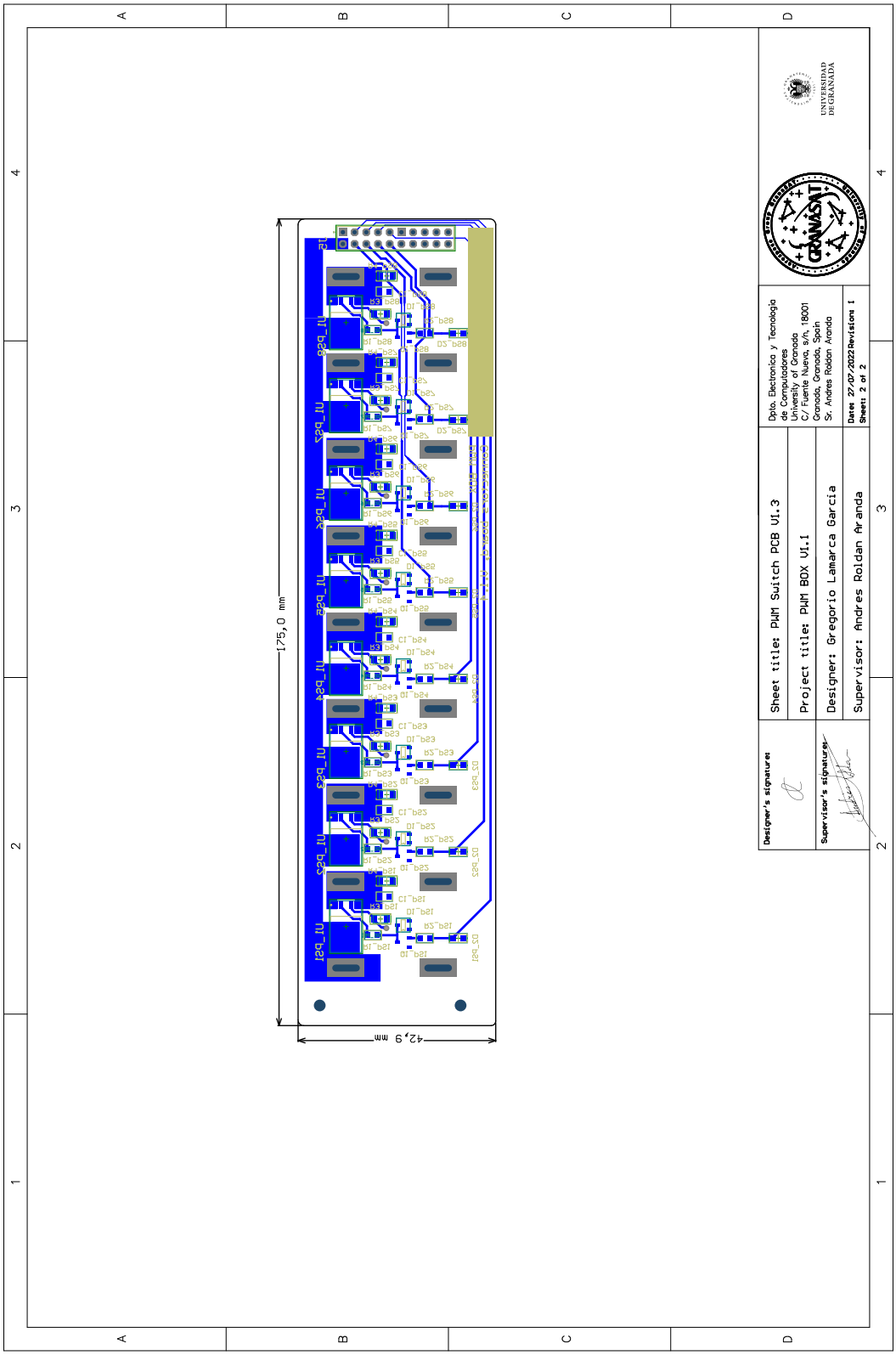


Figure 4.51 – *Bottom PCB prints in Connectors PCB project*

Chapter 5

Budget description of product

This section will take stock of the cost of design, taking into account the cost of materials and the time spent on design.

The MicrocontrollerPCB will be quoted first and then the ConnectorsPCB. Finally, the labour costs will be budgeted and the total will be calculated.

5.1 Budget for MicrocontrollerPCB

Table 5.1 – *MicrocontrollerPCB budget*

ITEM	Module	Units	Cost/Unit (€)	Cost (€)
1N4007RL	Diode	1	0.35	0.35
Tantalum Cap	Capacitor	2	0.85	1.7
0805 CAP	Capacitor	9	0.12	1.08
0805 RES	Resistor	5	0.18	0.9
BUTT-4	Button	1	0.18	0.18
Cap pol	Capacitor	1	0.45	0.45
WH2004	LCD	1	16	16
CH559L	Micro controller	1	1.21	1.21
LM1117	Voltage regulator	1	1.06	1.06
LM7805	Voltage regulator	1	1.42	1.42
PEC11R-4120F-S0018	Rotary encoder	1	2.01	2.01
PIN HEADER 10x2	Pin header	1	0.86	0.86
RGB 0805	LED	9	0.90	8.10
USB-B	USB	1	0.98	0.98
MicrocontrollerPCB	PCB	1	8.90	8.90
Total				45.20

5.2 Budget for ConnectorsPCB

Table 5.2 – *ConnectorsPCB budget*

ITEM	Module	Units	Cost/Unit (€)	Cost (€)
BAS21.215	Diode	8	0.16	1.28
0805 CAP	Capacitor	8	0.12	0.96
0805 RES	Resistor	16	0.18	2.88
1206 RES	Resistor	16	0.18	2.88
BTS6143D	Switching Transistor	8	2.02	16.16
PIN HEADER 10x2	Pin header	1	0.86	0.86
RGB 0805	LED	8	0.90	7.20
2N7002P	MOSFET	8	0.36	2.88
4mm Banana Jack input	Female banana jack input	18	1.52	27.36
ConnectorsPCB	PCB	1	8.30	8.30
Total				70.76

5.3 Overall budget

The cost per hour of a junior engineer in Spain is on average €12.31, so that will be the price used in this budget.

The hours spent on this project are estimated at 4 hours per day for 6 months (March, April, May, June, July and August), so an estimated 720 hours.

Sub-project	Budget [€]
MicrocontrollerPCB cost	45.20
ConnectorsPCB cost	70.76
Labour cost (720h)	8,863.20
TOTAL BUDGET	8979,16

Table 5.3 – *Total costs for the development of the project*

Chapter 6

Conclusions, future work and lessons learned

6.1 Conclusions

Through the development of this project, key knowledge has been put into practice when working with a real product.

Fields of hardware design and the study of low-level micro controllers have been involved in the realisation of this work, thus bringing product engineering closer to the bachelor.

It touches on areas of electronics that are necessary for a robust, fault-free robust design with no failures when it comes to testing them. We work on getting to know in depth how to work with a micro controller, how to get the most out of it and how to choose the right one for the project. when it comes to getting the most out of it and choosing the right one for the project. We study the mechanical design for the placement of the components in order to optimise the work to be carried out to optimise the work to be done and cut down on time.

The skills acquired during this project are of the utmost utility because by facing real problems with the design of a real product, the engineer is trained to perform what he/she will have to do in the near future in a short time will have to do in short margins of time, with skill and experience to anticipate to anticipate any inconveniences that may arise.

Due to sub-optimal organisation of the project, which has hindered the project's the project, it has been learned to appreciate the value of a better organisation of the project from the beginning of the project, which would have saved the project time and resources.

In summary, the author is satisfied with the work carried out which, although it needs some improvements that could not be made due to lack of time, raises the idea of improving it through study and training with future projects and motivating him to continue along the path of hardware design.

6.2 Future work

Although this project had the implementation at both hardware and software level, it has not been possible to finalise the software design. hardware and software, it has not been possible to complete the software design completely.

Future improvements for the product could be made in another final degree project, guided by this work to understand what has been done at hardware level and learning how to use the CH559 micro controller, to finally implement the complete product and even add some improvements that the work did not have initially:

- Electronics:
  - Create a new version of the PCBs, once the software has been implemented, to correct any errors that may be found in the implementation.
  - Enable through the usb configuration the option of obtaining data through the serial port (IIC) or (SPIO) for analysis on external equipment.
  - Study the use of joysticks to replace the rotary encoder in order to make the product cheaper and speed up menu navigation.
  - Study the use of an OLED screen to replace LCD to make the product cheaper and the interface more attractive.
  - Pin reallocation to squeeze the maximum potential out of the PWM channels when making signals that increase slowly over time.
- Software:
  - Implement the final software for this hardware design.
  - Increase the product's utilities by providing security (use of password) and connection to the cloud to save data for later analysis or edit parameters online.
  - Create a calibration function whereby using the controller and power resistors, the product can be calibrated so that the consumption reading can be optimised.
- Testing:
  - Carry out all the tests that could not be carried out.
  - Ensure that the product can be used for a continuous and uninterrupted period of time.
  - Ensure that the consumption reading is optimised to minimise error.

6.3 Lessons learned

The maturation of the learner with the realisation of the learner, both in terms of putting his or her knowledge into practice and in learning to cope to put their knowledge into practice and to learn to cope with a real project. a real project, brings with it lessons learned that are countless, but here is a brief summary of the most important ones but a brief summary of these highlights will be given here:

- Organisation, although costly at the outset and entailing a level of prior knowledge of what is to be done, is fundamental to the success of a project.
- Acquiring knowledge from any source of information available to the engineer is essential, as it is also a source of new solutions.
- Theoretical study of the subject to be worked on, which is the source of knowledge, is essential for the correct implementation of any technique to be used, thus ensuring a controlled and satisfactory result.

6

Bibliography

[1] Audio, B. P. *PEC11RSeries-12mmIncrementalEncoder*, 2011. https://www.farnell.com/datasheets/2360546.pdf Accessed: May 2022.

[2] Components101. *Joystick Module*, 2018. https://components101.com/modules/joystick-module Accessed: August 2022.

[3] Instruments, T. *LM340, LM340A and LM7805 Family Wide VIN 1.5-A Fixed Voltage Regulators*, 2016. https://www.ti.com/lit/ds/symlink/lm340.pdf Accessed: May 2022.

[4] Instruments, T. *LM1117 800-mA, Low-Dropout Linear Regulator*, 2022. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ti.com/lit/ds/symlink/lm1117.pdf?ts=1662012883486&ref_url=https%253A%252F%252Fwww.google.com%252F Accessed: May 2022.

[5] kprasadvnsi. *English Docs for CH559 Microcontroller*. https://kprasadvnsi.github.io/CH559_Doc_English/ Accessed: July 2022.

[6] Sitronix. *ST7066U Dot Matrix LCD Controller/Driver*, 2006. https://www.newhavendisplay.com/app_notes/ST7066U.pdf Accessed: May 2022.

[7] technologies, I. *PROFET® Data sheet BTS 6143 D*, 2003. https://pdf1.alldatasheet.com/datasheet-pdf/view/79514/INFINEON/BTS6143D.html Accessed: April 2022.

[8] WINSTAR. *WH2004D Character 20x4*, 2020. https://www.winstar.com.tw/uploads/files/f5a6b34c742f2388515856d7301491bf.pdf Accessed: May 2022.

Appendix A

Project budget

Appendix B

Link budget

Appendix C

Electronics schematics

Appendix D

Electronics BOM

Appendix E

Electronics Gerber drawings

Appendix F

Electronics assembly

Appendix G

Infineon's DAVE user's guide