E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Communication system in the charging process of electric vehicles

Grado en Ingeniería
en Tecnologías Industriales

Trabajo Fin de Grado

Juana Romero Aguinaga

Pablo Sanchis Gúrpide

Pamplona, 23/06/2014

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

INDEX:

1. CHARGING INFRASTRUCTURE AND STANDARDS

1.1 Introduction

The increasing public desire to reduce carbon emissions and transportation dependency on petroleum products is driving public interest and policy to alternative fuel sources and design. Most of the automotive manufacturers are either currently producing or planning to produce Electric Vehicles (EV) or Plug-in Hybrid Electric Vehicles (PHEV) to address this interest.

The commercial use of electric vehicles offers several major benefits for sustainable mobility and these are some of their advantages:

- Helping to attain major energy-environment policy goals: replacing conventional internal combustion engines by electric vehicles would result in major reductions in CO_2 emissions and in air quality improvements, especially in cities.
- Electric vehicle technology offers an opportunity to take the lead towards a sustainable green economy.
- Electric cars are more efficient than alternative transport technologies. Given the technology and infrastructure levels currently in place, grid-connected vehicles can become reality.

Electricity is a widely-available energy vector produced all over the world and its greater use in road transport can, while reducing greenhouse gas emissions, also simultaneously help to promote fuel diversification, strengthen energy security and improve air quality. And plug-in electric and hybrid vehicles have the potential to contribute significantly to meeting a range of environmental and economic challenges faced by the transportation system.

Integrating the charging system for electric vehicles into Europe's electricity networks will not require the development of any specific new technology. The European electricity industry is now engaged in promoting investment in the necessary infrastructure in order to foster the development of electric vehicles and ensure customers proper and convenient access to the electricity grid.

However, in order to ensure rapid market penetration and avoid any future incompatibility, it is vital to work out a cross-industry agreement on how to charge the vehicles and arrange for payment of the electricity. Standardising electric vehicle charging infrastructure will provide benefits to all stakeholders and developing standards is of the utmost importance to drive forward progress in European car and battery technology research, development and innovation.

1.2 Benefits of an agreed common standard

Standards play a key role in the development and deployment of technology in society, providing an indispensable basis for widespread market penetration and customer convenience. Agreed standards tend to encourage innovation, boost productivity and shape market structure in a way that enhances economic efficiency, reducing or eliminating technical barriers that can create market distortions.

For plug-in vehicles to become a success, both hardware (connector and cables) and communication software standards are a prerequisite to the establishment of a secure investment climate for the required infrastructure. Common standards will generate cost benefits and help to create economies of scale for both electricity companies and the automobile industry. They will also help to avoid the risk of stranded assets resulting from the deployment of interim proprietary solutions and foster the sharing of development costs.

Of course the customer will be the key determinant for the commercial success of electric transport. Common standards will help to ensure the driver enjoys a convenient recharging solution across the European Union that will avoid a multiplicity of different cables and adaptors and/or retrofit costs for adapting to new charging systems. Consumers will be able to choose their electricity supplier, and even more importantly, will be able to charge their vehicle in charging stations across Europe.

Experts from the electricity distribution business have already been working with automotive companies and original equipment firms to find an agreement on initiating standards for connecting electric vehicles to the power grids. This initiative provides a starting point, the aim being to draw up a roadmap for a rapid standardisation process. The common technical approach must then be further developed by the international standardisation bodies ISO and IEC.

The signatories to this declaration, integrate European electricity companies, distribution system operators, and national electricity sector associations, support the development of pre-standards for vehicle charging, with a view to driving forward market deployment. They hereby commit themselves to apply these pre-standards when developing infrastructure and vehicle connections, conscious that this approach will enable them to gain early experience with business models and to better assess the impact on the electricity grid when the standards are officially approved by ISO and IEC.

Hence standardization matters are currently divided between the two organizations in the following way:

- ISO: takes up the work related to the electric vehicle as a whole.

- IEC: takes up the work related to all the electric components and electric supply infrastructure.

1.3 Electric vehicle recharging standards

For wired charging, two options can be distinguished: AC and DC charging. Charging with alternating current (AC) is used for conventional and semi-fast charging at homes and offices and the majority of public recharging stations. Direct current (DC) is used for fast charging. Since all batteries require DC power to be charged, the AC power that is delivered by the electricity grid needs to be converted to DC at some point. An AC/DC convertor is thus needed between the grid and the battery. In the case of AC charging with regular mains power, power levels are low enough to install a small converter on-board the vehicle. For fast charging with higher power levels, a bigger and more expensive converter is needed that would not easily fit in a typical car. These high-power converters are therefore incorporated in the charging station and DC power is delivered to the car (See Figure 2 for a schematic drawing of these differences). Also, because of the higher power levels and related safety concerns, DC charging cables are always fixed to the charging station and there is thus only one plug that needs to be standardized. AC charging cables are often, but not always, loose cables with plugs on both ends and standardization may thus be necessary for both plugs.

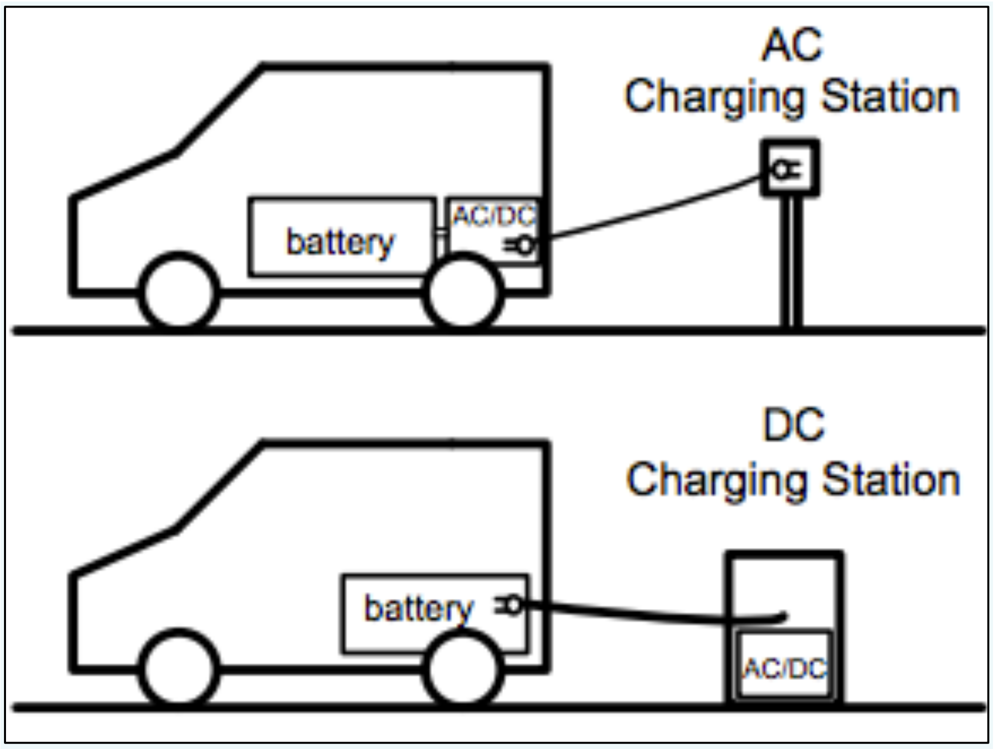


Figure 2

1.4 Specific standards: modes, types, and identification systems

For both AC and DC charging, multiple plug designs and charging modes have been developed and have been deployed throughout the world. Next to that, an even wider variety of identification and billing systems have been developed. In order to enable EV drivers to roam between networks and ultimately between countries, interoperability, and thus standardization, is necessary between the various modes, plugs, and identification and billing systems. Below we introduce these concepts and explain the major differences between the various options.

The charging mode:

The charging mode refers to power levels that charger and its connectors are rated for and the control and safety features that guarantee safe and efficient charging. The International Electrotechnical Commission (IEC) has recognized four different charging modes that vary in terms of complexity of the system and the speed with which a vehicle can be recharged.

Mode 1 charging encompasses charging from regular mains sockets (up to 16 Amperes) and it is done without any specific safety or communication features. This mode by definition requires the usage of a loose cable with plugs that match the car-side as well as the wall-side.

Mode 2 charging encompasses charging from regular mains sockets as well, but features a special cable with a so-called in-cable-control-box that controls the power level and thereby protects the user and the vehicle. Both Modes 1 and 2 are used in situations where a dedicated infrastructure is lacking (e.g. at home) or where the network operator has decided to offer a rather uncomplicated system. Because Mode 1 and 2 charging make use of regular sockets, plug designs vary per country and cross-border trips would require the use of several cables with varying plugs on the wall-side.

Mode 3 charging, which is to become the European standard, makes use of dedicated charging equipment which guarantees safe usage and which also enables communication between the charging equipment and the vehicle. Because of these additional features, a special cable and plug and socket combination are necessary.

Mode 4 charging entails the use of an AC/DC converter and charger in the charging equipment (instead of on-board the vehicle) and DC power is delivered to the vehicle. This mode is typically used for fast charging with power levels starting at 50 kW.

The plug type:

The plug type refers to the physical design of the plug with which the vehicle is connected to the charging equipment. There are three officially recognized plug designs for Mode 3 charging, these are designated as Types 1, 2, and 3.

Type 1 (Yazaki) is used mainly in the U.S. and Japan and is supposed to be used on a cable that is fixed to the charging equipment. In other words, the Type 1 plug is used specifically to plug into the car and therefore requires a car with a compatible inlet (the vehicle inlet).

The Type 2 plug (Mennekes), the new European standard, is used on loose cables and connects the cable to the charging equipment. On the car side, the cable can have any plug that matches the vehicle's inlet, but this often a Type 1 design because many cars have a Type 1 inlet anyway. Type 2 plugs are rated for higher power levels that Type 1 plugs and can therefore be used for semi-fast charging with chargers that make use of three-phase power connections.

The Type 3 plug (Scame) is mostly the same as the Type 2 plug, but its use is limited to several countries in southern Europe (i.e. Italy, France). These countries prescribe the use of so-called safety shutters on power outlets that are installed outside and the Type 3 socket features such shutters. Because the Type 2 and 3 plug and socket combinations are not compatible, travelling between, for instance, Germany and France would require an additional cable.

As for mode 4, DC fast charging, there is currently only one design that is used in practice. This is the CHAdeMO standard and this standard specifies the charging protocol as well as the physical design of the plug and vehicle inlet. This implies that a CHAdeMO charger, like all mode 4 chargers, makes use of a fixed cable. A large consortium of car manufacturers has agreed on a competing standard in which either a Type 1 or Type 2 plug is combined with additional pins for DC power. These are the Combo 1 and 2 plugs and are mean to be used on vehicles with a matching vehicle inlet that is also compatible with Type1 and Type 2 plugs.

No actual standardization has taken place yet in Europe and the various plug types are still in use and most countries have in fact not even agreed on a national standard, despite an announcement from the European Commission in January 2012. In this (proposed) directive two plug designs are selected to become the EU standard (Type 2 and Combo 2). To quote the report:

- Alternate Current (AC) slow recharging points for electric vehicles shall be equipped, for interoperability purposes, with connectors of Type 2 as described in standard EN62196-2:2012.
- Alternate Current (AC) fast recharging points for electric vehicles shall be equipped, for interoperability purposes, with connectors of Type 2 as described in standard EN62196-2:2012.
- Direct Current (DC) fast recharging points for electric vehicles shall be equipped, for interoperability purposes, with connectors of Type "Combo 2" as described in the relevant EN standard, to be adopted by 2014.

2. DESCRIPTION OF THE WORK PERFORMED

2.1 Introduction

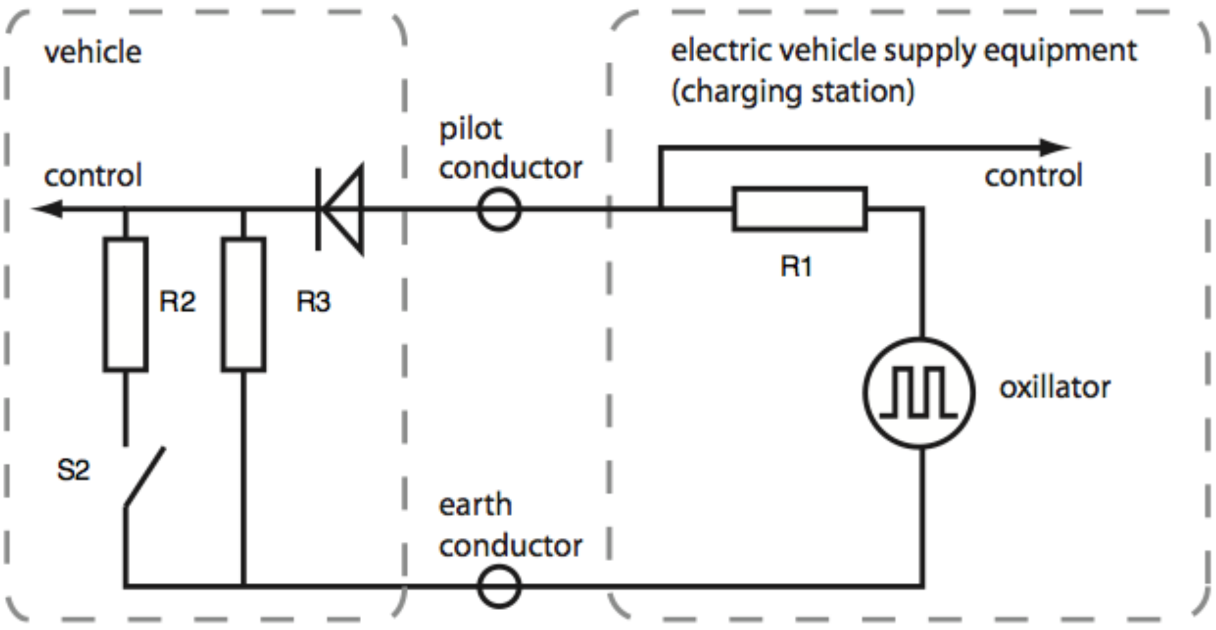
The charging process of electric vehicles needs to be checked and controlled constantly, in order to assure its safety and guarantee. This is the reason why we have found indispensable to create a program to establish the communication between the charging station and the electric vehicle. We will build a control mechanism (pilot function) for the supply of electrical energy to electric vehicles using a control pilot circuit with PWM (Pulse Width Modulation) and a control pilot wire.

The control pilot will be in charge of the verification that the vehicle is appropriately connected and it will energize and de-energize the system depending on the voltage read. It concerns all charging systems that ensure the pilot function with a pilot wire circuit with PWM for mode 2, mode 3 and mode 4 charging.

Our mains principles are the standardization, the safety, the simplicity and the price. And following these goals, we will try to get the suitable circuit and code.

2.2 Explanation of the program

The communication between the charging station and the electric vehicle will be establish as the following control pilot circuit:

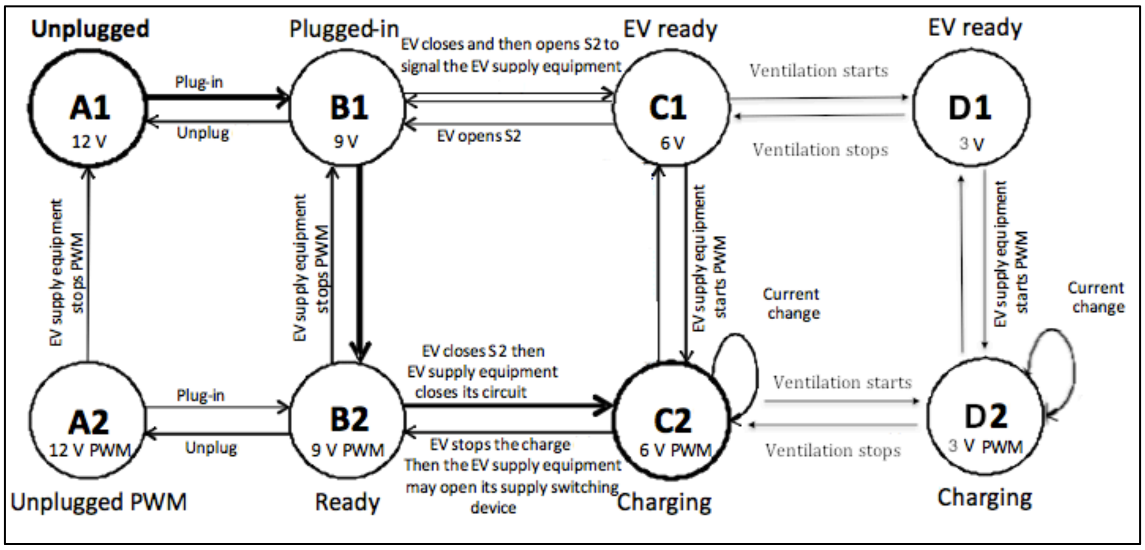


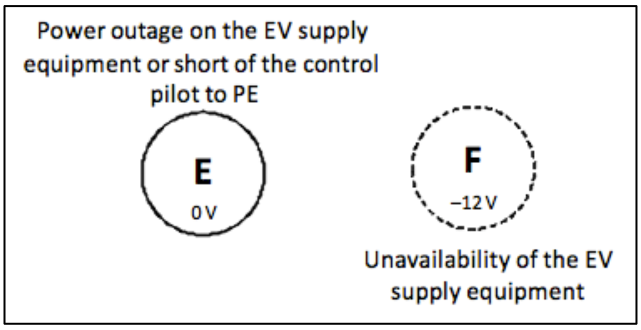
We need to build this pilot circuit for allowing the communication between the charging station and the EV (as it is the main element between the two bodies). Depending on the voltage read by the pilot conductor (DC voltage or PWM), we will determine different actions during the charging process.

First of all, there are different circuit parameters that shall be define:

- EV supply equipment generator: it has to be able to generate a steady state DC voltage (12V) or a square wave (between +12V and -12V). An error of ± 0.6V will be accepted.
- R1: The value of this resistance should be around 1kΩ with a maximum variability of 3 % (from 970Ω to 1030Ω). Even though 1% resistor are more recommended.
- Pilot Control: it will give us the voltage measured in that point and we will create it with the suitable software.
- Diode: it allows the electric current to pass only in one direction (from the source supply to R2 and R3 resistors.
- S2: switch contact in the EV.
- R2: its value is around 1.3kΩ but it can range between 1261Ω and 1339Ω.
- R3: its value is around 2.74kΩ. It can range between 2658Ω and 2822Ω.

First of all, we have to explain the different states in the connection process of the EV (IEC 62763) until the car is really charging. In this diagram we can observe the sequence from one state to the next one:


Unplugged
A1
12 V
Plug-in
Unplug
Plugged-in
B1
9 V
EV closes and then opens S2 to signal the EV supply equipment
EV opens S2
EV ready
C1
6 V
Ventilation starts
Ventilation stops
EV ready
D1
3 V
EV supply equipment stops PWM
EV supply equipment stops PWM
EV supply equipment starts PWM
Current change
EV supply equipment starts PWM
Current change
A2
12 V PWM
Unplugged PWM
Plug-in
Unplug
B2
9 V PWM
Ready
EV closes S2 then EV supply equipment closes its circuit
EV stops the charge Then the EV supply equipment may open its supply switching device
C2
6 V PWM
Charging
Ventilation starts
Ventilation stops
D2
3 V PWM
Charging


Power outage on the EV supply equipment or short of the control pilot to PE
E
0 V
F
−12 V
Unavailability of the EV supply equipment

As we can observe, the car is only getting charged in the State C2 and in the State D2 (charging with ventilation).

In State E, there is no power to the EV supply equipment. This state may be caused by any number of difficulties and shall not be used as a signalling state to convey specific information.

In State F the EV supply equipment is unavailable. Nevertheless, it is not recommended to use this state to signal unavailability of energy to the EV.

Depending on the state of the charging process, the pilot resistance values seen by the charging station will vary as indicated in the following chart:

State	Function	Value
A	Standby	Open circuit
B	Vehicle detected	2700 Ω
C	Charging (switch S2 closed)	880 Ω
D	Charging with ventilation	240 Ω

So depending on the voltage read before the resistor of the proximity, we can know the value of the resistance between the proximity contact and the protective conductor, mounted inside the plug. Once we know the value of the resistor of the proximity, we can impose the current limit for the cable, as indicated in the next chart:

Resistance	Current limit	Cable (width)
1500 Ω	13 A	1.5 mm^2
680 Ω	20 A	2.5 mm^2
220 Ω	32 A	16 mm^2
100 Ω	63 A	16 mm^2

Finally, we will determine that the current that can be sent will be the lower between the current limit for the cable and the current limit set for the equipment, so we will have to compare these two values and assigned as the real current of the circuit the lower of then.

2.3 Explanation of the code

2.3.1 Equipment used

For developing the code we have used an Arduino board that consists of an Atmel 8-bit AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. It comes with a simple integrated development environment that runs on regular personal computes and it allows writing the programs in C language.

An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields (printed circuit expansion boards that plug into the normally supplied Arduino pin-headers). Some of these shields communicate with the Arduino board directly over various pins.

Most boards include a 5 Volt linear regulator and a 16 MHz crystal or ceramic oscillator. An Arduino microcontroller is also pre-programmed with a boot loader that simplifies uploading the program to the on-chip flash memory, compared with other devices that typically need an external programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer.

There are many Arduino-compatible and Arduino-derived boards. Some are functionally equivalent to an Arduino and may be used interchangeably. Many are the basic Arduino with the addition of commonplace output drivers and others are electrically equivalent but change the form factor, sometimes permitting the continued use of Shields. But some variants use completely different processors, with varying levels of compatibility.

Between all the possible Arduino boards we have selected the Arduino Uno board because of its popularity and its appropriately characteristics.

Arduino Uno board:

The Arduino Uno is a microcontroller board based on the ATmega32. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega16U2 programmed as an USB-to-serial converter.

In summary these are the values of the different parameters:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3 V Pin	50mA
Flash Memory	32KB (ATmega328) of which 0.5KB used by boot loader
SRAM	2KB (ATmega328)
EEPROM	1KB (ATmega328)
Clock Speed	16MHz

According to the power of the Arduino Uno, it can be powered via the USB connection or with an external power supply. The power source is selected automatically. Also external (non-USB) power can come either from an AC-to-DC adapter or battery.

The board can operate on an external supply of 6 to 20 Volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 Volts.

The power pins are the following:

- Vin: The input voltage to the Arduino board when it is using an external power source (as opposed to 5 Volts from the USB connection or other regulated power source). We can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 to 12V), the USB connector (5V), or the Vin pin of the board (7 to 12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator and can damage the board.
- 3.3V: A 3.3 Volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND: Ground pins.

- IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Regarding the Inputs and Outputs of the Arduino Uno, it has 14 digital pins that can be used as both options using different functions (pinMode, digitalWrite and digitalRead) and there is a built-in LED connected to digital pin 13. These opens operate at 5 Volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50kOhms.

When programming the Arduino Uno, it can be done with the Arduino Software, available for free in the official site for Windows, Mac OS and Linux. It will only be necessary to buy the Arduino board, which current prices run around $30.

The Arduino Uno has a resettable polyfuse that protects the computer´s USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500mA are applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

So besides and Arduino Uno board provides us all the equipment we need in a very flexible and easy way, its low price determines us to select it as the best solution.

2.3.2 Explanation

Basically, the code is an active component that can react differently depending on external changes. It has three main functions:

1. Controlling the voltage before the resistor of the proximity we can guess the value of this resistor. Once we know the value of the resistor, we can determine the maximum current that can be sent. Different values of the resistor of the proximity mean that the cable has different width, so the maximum current that flows through the cable has to be different in each case.
   The next step to determine the real current that will be sent is to compare the maximum current depending on the resistor of the proximity and the current limit set for the equipment. The lower of these two values will be assigned as the current sent.
2. Controlling the pilot´s voltage we can determine in which state is the charging process (A, B or C). In state B and C we will produce a PWM signal to start the charging process.
3. It will be necessary to create a PWM signal with a different duty cycle depending on the value of the current sent.

First part of the code:

The first thing that needs to be done is to measure the voltage before the resistor of the proximity ("VolPP"). For this, we assign to a pin of the Arduino the function to measure the voltage (Pin A5). The function "analogRead" can read the value from an analog pin (preconfigured as an input with the function "pinMode"), so it will map the input voltage between 0 and 5 Volts into integer values between 0 and 1023. As we want to have a measurement of the voltage between 0 and 5 Volts, we will transform the range of the function multiplying it times 5/1023. This value will be storage in a new function of "float" type: $VPP = VolPP \cdot \frac{5}{1023}$; and that will be the voltage before the resistor of the proximity we want to know.

The next step is to check with the "if" function the value of the voltage we have just read, and depending on its value we can know how much is the value of the resistor of the proximity (variable of "int" type and called "prox"):

a) If the voltage ("VPP") is approximately between 2.9V and 3.15V, then the value of the resistor of the proximity is 1500Ω.
b) If the voltage ("VPP") is approximately between 1.93V and 2.18V, then the value of the resistor of the proximity is 600Ω.
c) If the voltage ("VPP") is approximately between 0.85V and 1V, then the value of the resistor of the proximity is 220Ω.
d) If the voltage ("VPP") is approximately between 0.42V and 0.51V, then the value of the resistor of the proximity is 100Ω.

Now, we are ready to determine the maximum current that can be sent to the car depending on the value of the resistor of the proximity. This value will be storage in a variable called "Ipp" and it will be of "int" type. We will use a "switch" function to determine its value as explained above:

a) If the resistor of the proximity ("prox") is 1500Ω, then the maximum current that can be sent is 13A.
b) If the resistor of the proximity ("prox") is 600Ω, then the maximum current that can be sent is 20A.
c) If the resistor of the proximity ("prox") is 220Ω, then the maximum current that can be sent is 32A.
d) If the resistor of the proximity ("prox") is 100Ω, then the maximum current that can be sent is 63A.

The last thing to do for assigning the real value of the current sent (variable of "int" type and called "I"), is to compare the value of the maximum current that can flows depending on the width of the cable ("Ipp") and the current limit set for the equipment (this value will be storage also in a "int" variable and it will be called "Ivp"). We will use the "min" function for comparing these two values and assigning to the variable "I" the lower.

Second part of the code:

It is required to measure the voltage of the pilot and we have to take into account that it can be a DC or an AC voltage. For this, as done in the first part of the code, we assign to a pin of the Arduino Uno the function for measuring the voltage (Pin A1). It will measure the input voltage with a number between 0 and 1023 (in a variable of “int” type and called “sensorValue”). As we want to have a measurement of the voltage between -12V and 12V (5V will correspond to a range from -12V to +12V), we will transform the range of the function subtracting 512 and then multiplying it times 24/1023. This value will be stored in a new function of “float” type called “Vpilot”: $Vpilot = (sensorValue - 512) \cdot \frac{5}{1023}$. It takes about 100 microseconds (0.0001s) to read an analog input and the frequency of the PWM signal that the Arduino Uno creates is approximately 490Hz. This means that the period of a PWM signal is around 0.002 s (2ms). So in case we are measuring an AC voltage we should make sure that we get the real voltage. For that, it will be necessary to make 20 measurements of the voltage using the “for” function.

Now we are ready to use the value of the pilot´s voltage (“Vpilot”) to know in which state the charging process is using the “if” function and assigning to the “boolean” variables “stateA”, “stateB” and “stateC” the value “true” or “false”:

a) If “sensorValue” is equal or greater than 950 (the pilot´s voltage is around 12V), then we are in the State A (the vehicle is not yet plugged-in to the electric post).
b) If “sensorValue” is between 875 and 895 (the pilot´s voltage is around 9V), then we are in the State B (the vehicle has been plugged but it is not ready to be charged).
c) If “sensorValue” is between 790 and 810 (the pilot´s voltage is around 6V), then we are in the State C (the vehicle is ready to be charged).

In the case of being in the State C we have to create a PWM signal for changing into State C2, and at this point the charging process will start. This is why in the State C an “int” variable called “PWMsignal” will have the value “1”. Also, we will activate a transistor attached to Pin 13 (preconfigured as an output) in this state because like this there will be current between the Emisor and the Collector of the transistor and the let attached before the Emisor will be on (this will make very easy to appreciate when the electric vehicle is being charged).

Third part of the code:

Once we know that we are in the State B or C (“PWMsignal=1”), we will have to create a PWM signal with a different duty cycle depending on the value of the current sent for changing into the State B2 or C2 and start the charging process. Nevertheless, if we are in State A (“PWMsignal=0”), we will just send 5V through the beginning of the circuit (Pin 9 of the Arduino Uno board) with the “digitalWrite” function.

When a PWM signal will be needed in the Pin 9, we will create it determining the duty cycle (a “float” variable called “DC”) and using the “analogWrite” function. It is very important that the frequency of the PWM signal has to be 1000 Hz because otherwise the electric vehicle will not detect the signal and it will not charge.

The duty cycle of the signal will be different depending on the current of the circuit (I) and it can have these two different values:

- If the current (I) is between 51A and 80A, then the duty cycle will be: $DC = \left(\left(\frac{I}{2.5} + 64\right)/100\right) \cdot 1023$
- In the rest of the cases, the duty cycle will be: $DC = \left(\left(\frac{I}{0.6}\right)/100\right) \cdot 1023$

Throughout the code we can also find the functions “Serial.print” and “Serial.println”. Both of them are just used to print in the Serial Monitor the value of the variable indicated.

2.3.3 Code

```
#include <PWM.h> //We include the library for
     changing the frequency of the PWM signal
int32_t frequency=1000; //We set the frequency of the
     PWM at 1000 Hz
//Declare all variables in head section

float Vpilot=0;
int sensorValue;
int PWMsignal=0; //Variable for knowing if the EV
     supply equipment produces a PWM signal or not
int Ipp=0; //We create a variable to indicate the
     maximum current that can flows in the cable. If
     any of the resistors below is connected, then
     the maximum current will always be 0 A (No
     current)
int Pin=9;
int Ivp=22; //We assign the current limit set for the
     equipment in A
int PP=A5; //Pin attached before the resistor of the
     proximity
int I=0; //Variable that contains the real current
     that will flow
int Mos=13; //Pin for Mosfet
int VolPP; //Variable to store the voltage before the
     resistor of the proximity (It is a number
     between 0 and 1023)
float VPP;
float v2;
float DC; //The duty cycle for PWM
int DCint; //The duty cycle for the output pin
boolean stateA=true;
boolean stateB=false;
boolean stateC=false;
int prox;
boolean del=false;

//The setup routine runs once when you press reset:

void setup() {

     Serial.begin(9600);     //Initialize     serial
          communication at 9600 bits per second
InitTimersSafe();
bool success=SetPinFrequencySafe(9,frequency);

}

//The loop routine runs over and over again forever:

void loop() {

     pinMode (PP, INPUT);
     VolPP=analogRead(A5); //Variable to store the
          voltage before the resistor of the
          proximity (It is a number between 0 and
          1023)
     v2=VolPP*5;
     VPP=v2/1023; //We transform the voltage from 0-
          1023 to 0-5V and we storage it in the
          variable VPP

     prox=0;

     //We check the voltage before the resistor of
          the proximity and depending on it we can
          know how much is the resistor and assign
          the maximum current

     if (VPP>2.9 && VPP<3.15){ //The resistor of the
          proximity is around 1500Ω
          prox=1500;
     }

     if (VPP>1.80 && VPP<2.18){ //The resistor of the
          proximity is around 680Ω
          prox=680;
     }
```

```
if (VPP>0.85 && VPP<1){ //The resistor of the
     proximity is around 220Ω
     prox=220;
}

if (VPP>0.42 && VPP<0.51){ //The resistor of the
     proximity is around 100Ω
     prox=100;
}

Serial.print("VPP:");
Serial.println(VPP);

//We check the voltage before the resistor of
     the proximity and depending on it we can
     know how much is the resistor and assign
     the maximum current

switch (prox){

     case  1500: //The resistor of the proximity
          is around 1500Ω
     Ipp=13;
     break;

     case 680: //The resistor of the proximity
          is around 680 Ω
     Ipp=20;
     break;

     case 220:  //The resistor of the proximity
          is around 220Ω
     Ipp=32;
     break;

     case 100:  //The resistor of the proximity
          is around 100Ω
     Ipp=63;
     break;

     default:
     Ipp=0;
}

I=min(Ipp,Ivp);

Serial.print("Proximity:");
Serial.print(prox);
Serial.print(" - "),
Serial.println(I);
```

```
//We make 12 measurements of the voltage (1.2ms)
     to make sure in case we have a PWM signal
     which one is the highest and the lowest
     voltage:

for (int i=0; i<20; i++){

     sensorValue = analogRead(A1);   //Read the
          input on pin A1 (0-1023)
     Vpilot=((sensorValue-512)*24/1023);
          //Vpilot is now between -12 and +12
}

//Note: 5V corresponds to a range of -12 to +12

Serial.print("sensorvalue:");
Serial.println(sensorValue);

pinMode(Mos, OUTPUT);

//We determine if there is or not a PWM signal:

if (sensorValue>=870){
     stateB=false;
     stateC=false;
     stateA=true;
     digitalWrite(Mos, LOW);
     PWMsignal=0; //State A, no pilot
}

if (sensorValue>=600 && sensorValue<=680){
     stateB=true;
     stateC=false;
     stateA=false;
     digitalWrite(Mos, LOW);
     PWMsignal=1; //If the pilot´s voltage is
          around 9V, it means we are in state B1
          and we create a PWM signal to go to
          state B2
}

else if(sensorValue>=430&&sensorValue<=460&&I>0)
{
     stateC=true;
     stateB=false;
     stateA=false;
     digitalWrite(Mos, HIGH);
     PWMsignal=1; //If the pilot´s voltage is
          around 6V, it means we are in state C1
          and we create a PWM signal to go to
          state C2
}
```

```
        Serial.print("PWMsignal:");
        Serial.println(PWMsignal);

        pinMode(Pin, OUTPUT);

        if (PWMsignal==0){ //When there is current on
             the Pin 9 and there is not a PWM signal,
             then the pilot´s voltage is 12 V

             digitalWrite(Pin, HIGH);
        }

        else if(PWMsignal==1){ //When there is a PWM
             signal on the pin 9, then the pilot´s
             voltage is a PWM signal between 12V and -
             12V

        //We create the PWM signal in the PIN 9 between
             12V and -12V

             //Preliminary operation without PWM

             if(I>51 && I<80){
                  DC=(I/2.5+64)/100;
             }

             else{
                  DC=(I/0.6)/100;
             }

             DCint=DC*255;

             //Next line disabled for testing

             pwmWrite(9, DCint); //We create the PWM
                  signal in the Pin 9 with a duty
                  cycle=DCint

             Serial.print("Dutycycle:");
             Serial.print(DCint);
             Serial.print(" - ");
             Serial.println(DC);

        }
}
```

2.4 Explanation of the circuit

2.4.1 Equipment used

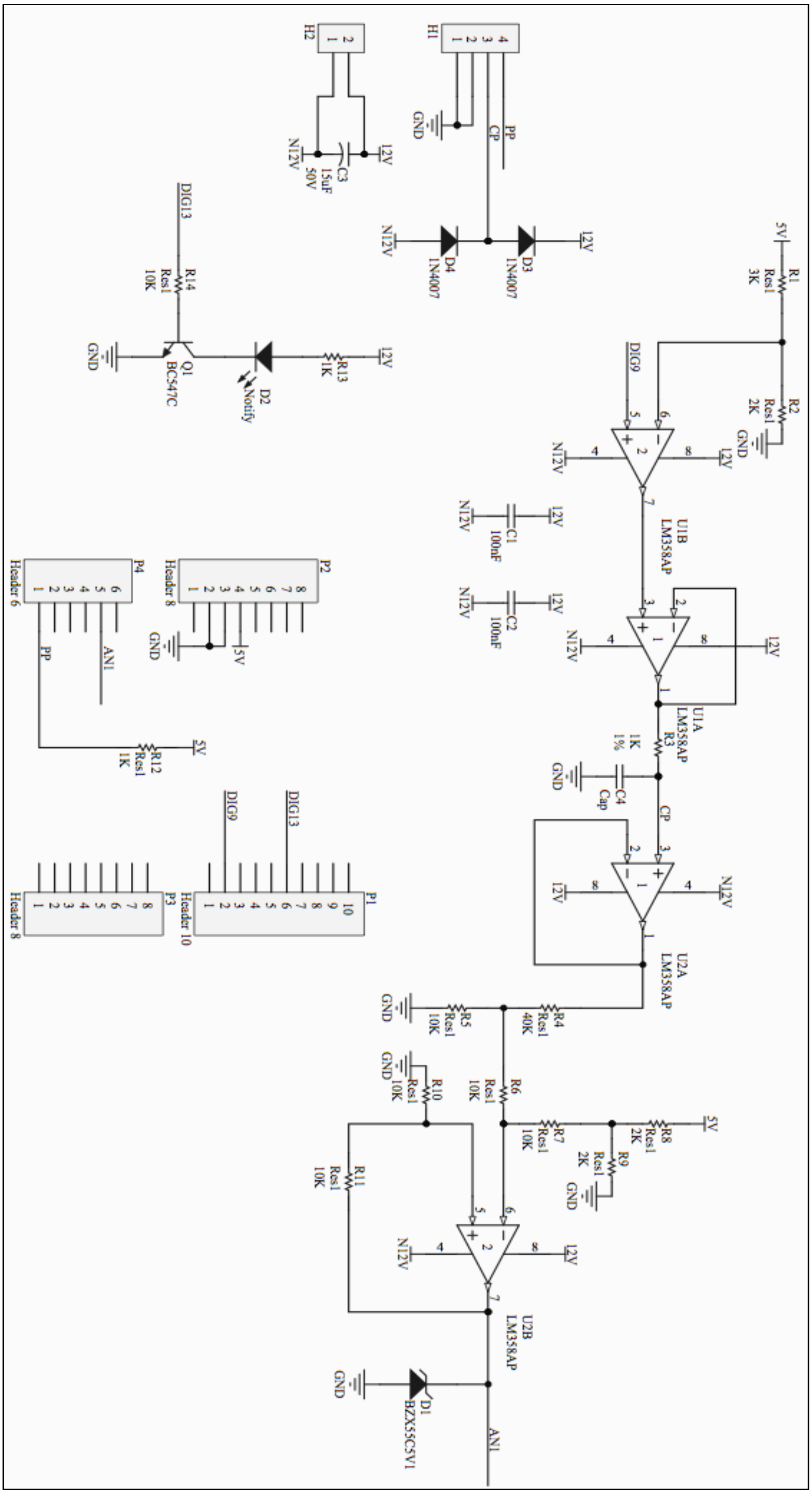
We have first tested the circuit on a Breadboard using the following components:

- Breadboard
- Wires
- Operational amplifiers (op amps) LM358AP
- Resistors
- Transistor BC547C
- Capacitors
- Zener diode (5.1V)
- External power supply
- Testing socket outlet
- Switch

We will use an Arduino Uno board and a USB cable to connect it to the computer and an oscilloscope for checking the voltage in different points of the circuit.

2.4.2 Circuit

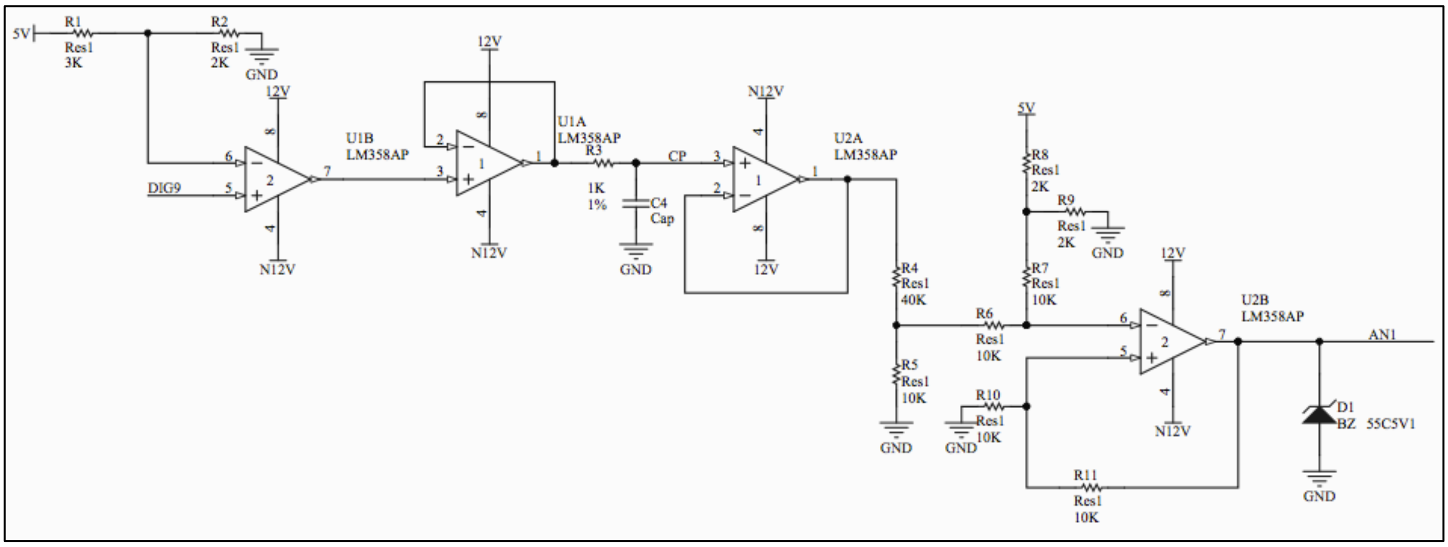
The design of the circuit is the following:



H1
PP
CP
GND
H2
C3
15uF
50V
12V
N12V
D3
1N4007
D4
1N4007
5V
R1
Res1
3K
R2
Res1
2K
DIG9
U1B
LM358AP
C1
100nF
C2
100nF
U1A
LM358AP
R3
1K
1%
C4
Cap
U2A
LM358AP
R4
Res1
40K
R5
Res1
10K
R6
Res1
10K
R7
Res1
10K
R8
Res1
2K
R9
Res1
2K
R10
Res1
10K
R11
Res1
10K
U2B
LM358AP
D1
BZX55C5V1
AN1
DIG13
R14
Res1
10K
Q1
BC547C
D2
Notify
R13
1K
P2
Header 8
P4
Header 6
R12
Res1
1K
P1
Header 10
P3
Header 8

As we can see, there are different components connected, but we can easily understand the circuit if we divide it in three parts:

First part (in the top of the diagram):



The first thing that needs to be done is to transform the range of the voltage between 0 and 5V power supply of the Arduino to a voltage between -12 and 12V. We will use the op amp U1B LM358AP (first op amp) and U1A LM358AP (second op amp) to get it.

The first op amp will be connected by the following way: the positive input will be connected to the Pin 9 of the Arduino and the negative input to 2V. We will obtain this 2V simply connecting in series two resistors of 3KΩ (R1) and 2KΩ (R2) and applying 5V (generated by the Arduino). Like this, in the output of this op amp we will be able to obtain 12V if the Pin 9 is High (5V) or -12V if the Pin 9 is Low (0V). It will be necessary to connect the positive power supply and the negative power supply of the op amp to an external power supply of +12V and -12V.

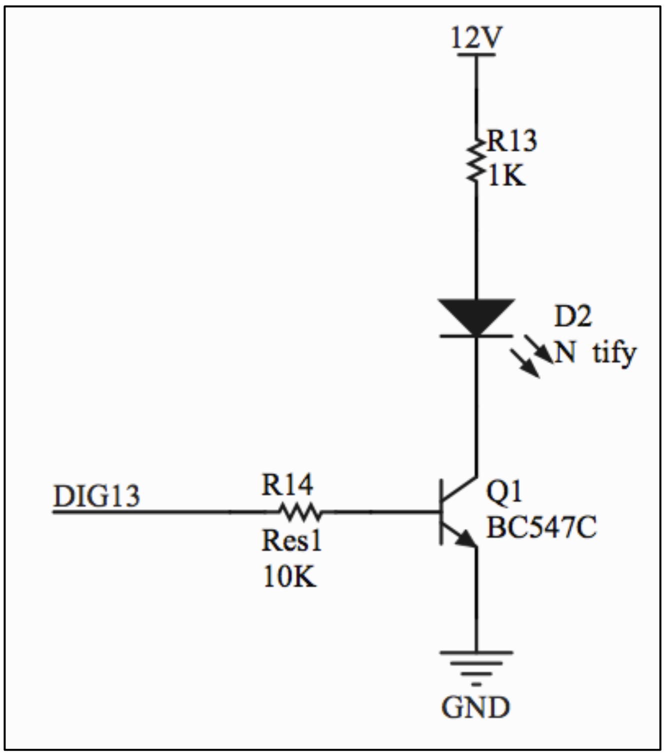
We use the second op amp to buffer the signal. For this, we will connect the output of the first op amp to the positive input of the second one and the negative input of the second op amp will be connected to its own output. As the first op amp, it will be powered with the external power supply.

After this, we have to use a 1KΩ resistor (R3), which must be a 1% precision (1% tolerance) and then we can connect the Control Pilot (CP) of the plug. If the charging process is in State A, the voltage read by the Control Pilot will be around 12V; if it is in State B, the pilot´s voltage will be around 9V; and if it is in State C, the pilot´s voltage will be around 6 V. An additional capacitor (C4) of 680 pF will be placed between the CP and the ground for filtering high frequency noise. Also, it is necessary to add a buffer op amp between the CP and the 40KΩ resistor.

As we want to measure the voltage with the Arduino and it can only receive a maximum of 5V, we will use a resistor of 40KΩ (R4) and another of 10kΩ (R5) connected in series. Like this, the voltage between these two resistors will be approximately a fifth part of the voltage before the R4 resistor. So before the R5 resistor the voltage will be approximately between -2.5V and 2.5V. We will then use the op amp called U2B LM358AP (fourth op amp) to raise this voltage to 0-5V range using a non-inverting summing configuration.

The last consideration we will have to have into account is to use a Zener diode of 5.1V before connecting the output of the fourth op amp to the pin of the Arduino where we will measure the voltage (Pin A1) for protecting the Arduino Uno board in case the voltage is higher than 5V.

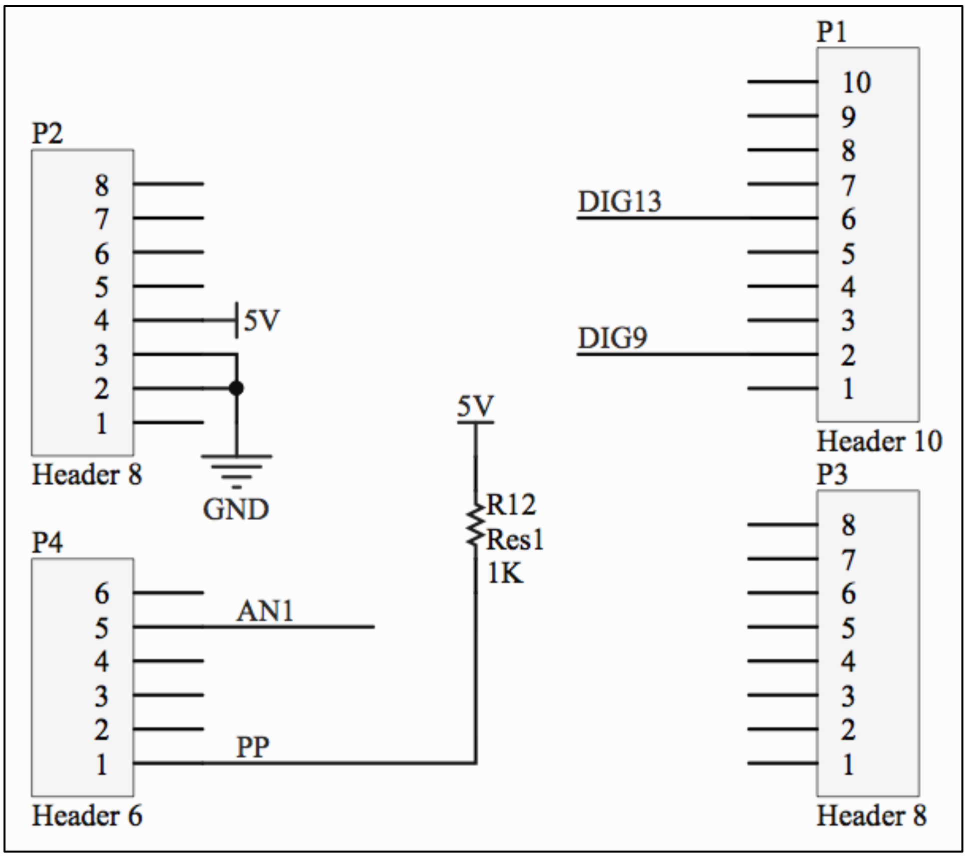
<u>Second part (in the bottom of the diagram):</u>



For having a better knowledge of when the car is charging, we will program a LED that will be on just in this case. For this, we will use a transistor (Q1 BC547C) with the Base connected to the Pin 13 of the Arduino with a 10kΩ resistor (R14). The LED will be placed between a 12V supply followed by a 1kΩ resistor (R13) and the Collector of the transistor. And the Emitter will be just connected to the ground.

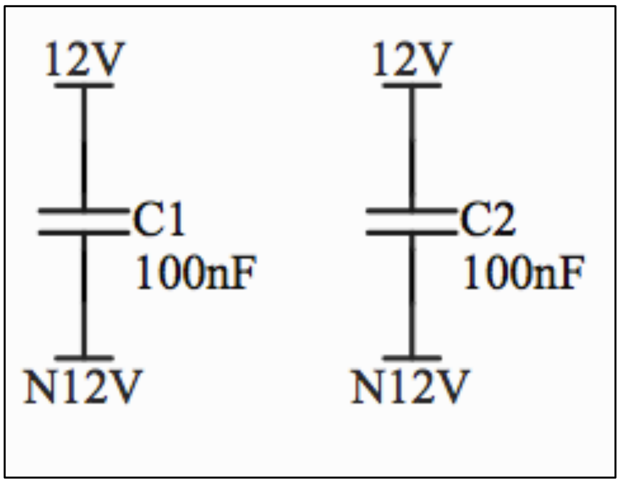
Third part (in the bottom of the diagram):

We can see the connections into the pins of the Arduino Uno board:

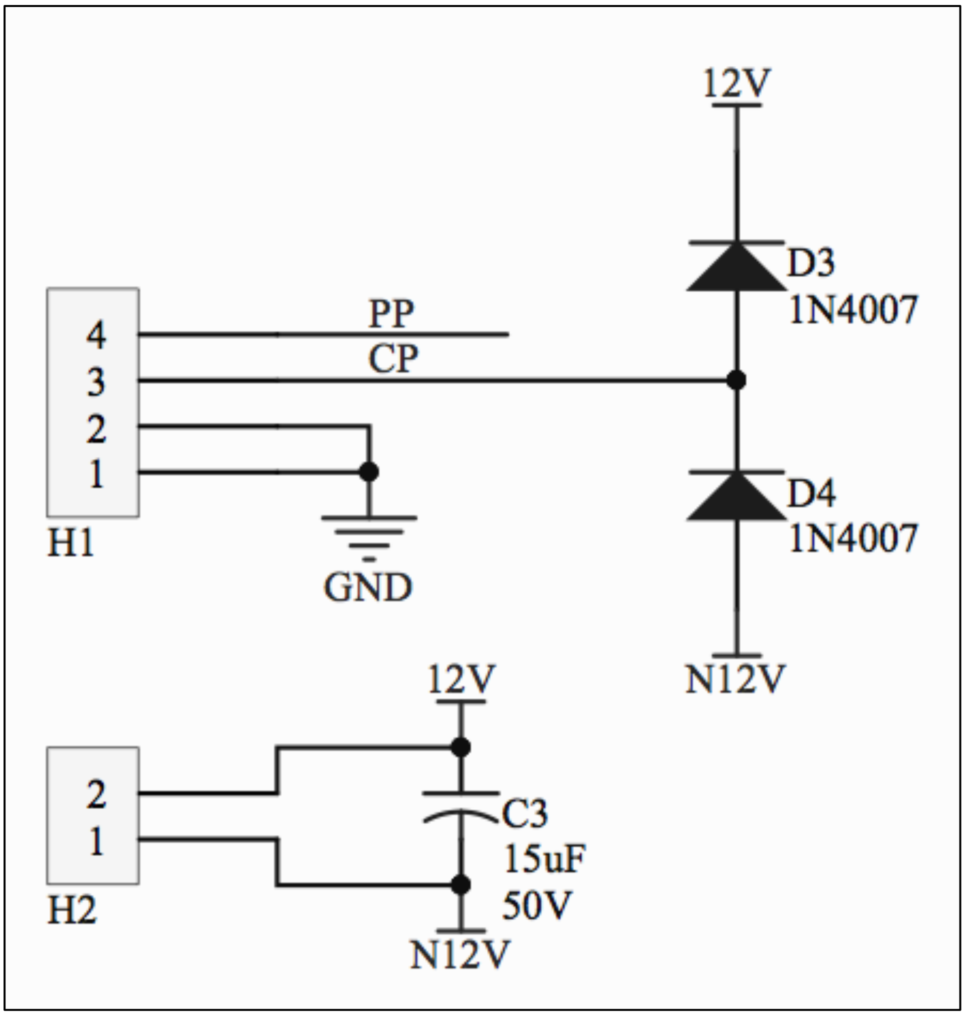


The proximity of the plug is connected between the analog pin A5 (Pin 1 of the Header number 6) and 5V followed by a 1kΩ resistor (R12):

Also we will place a capacitor of 100nF between the +12V and -12V power supply for each op amp for bypassing (for avoiding noise):

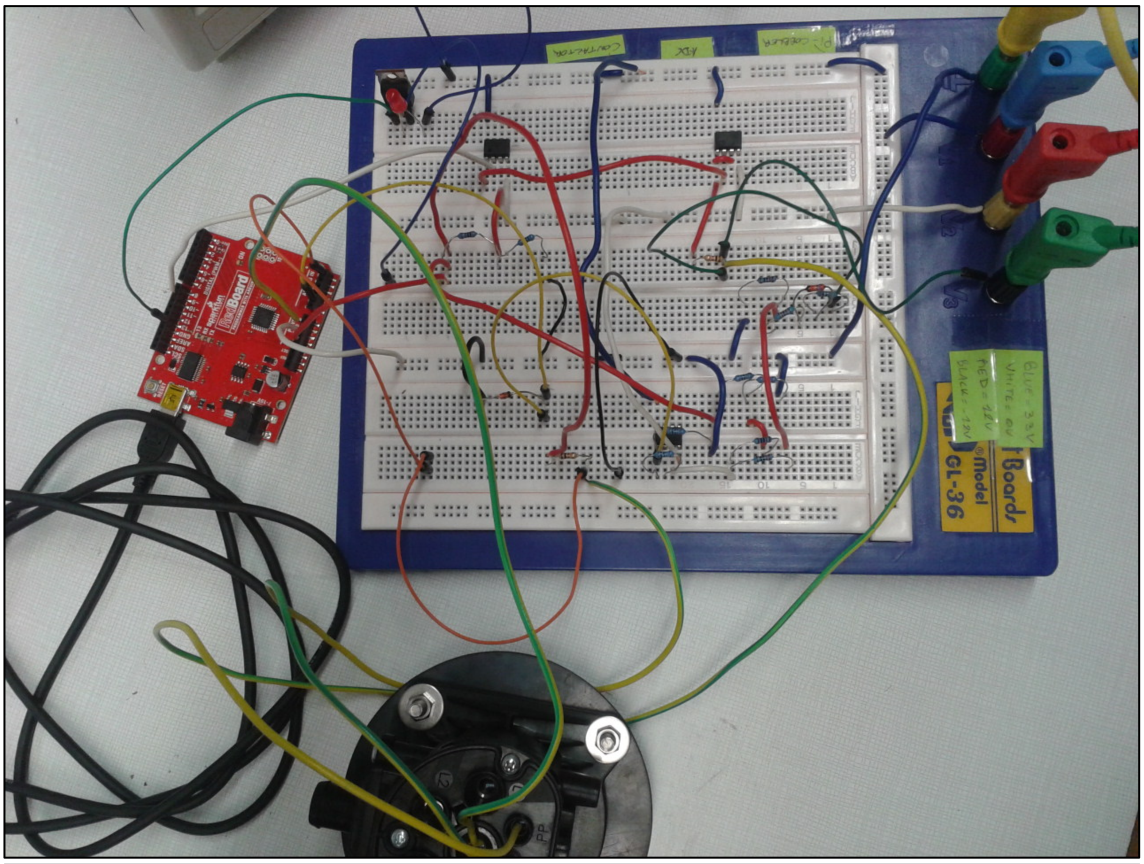


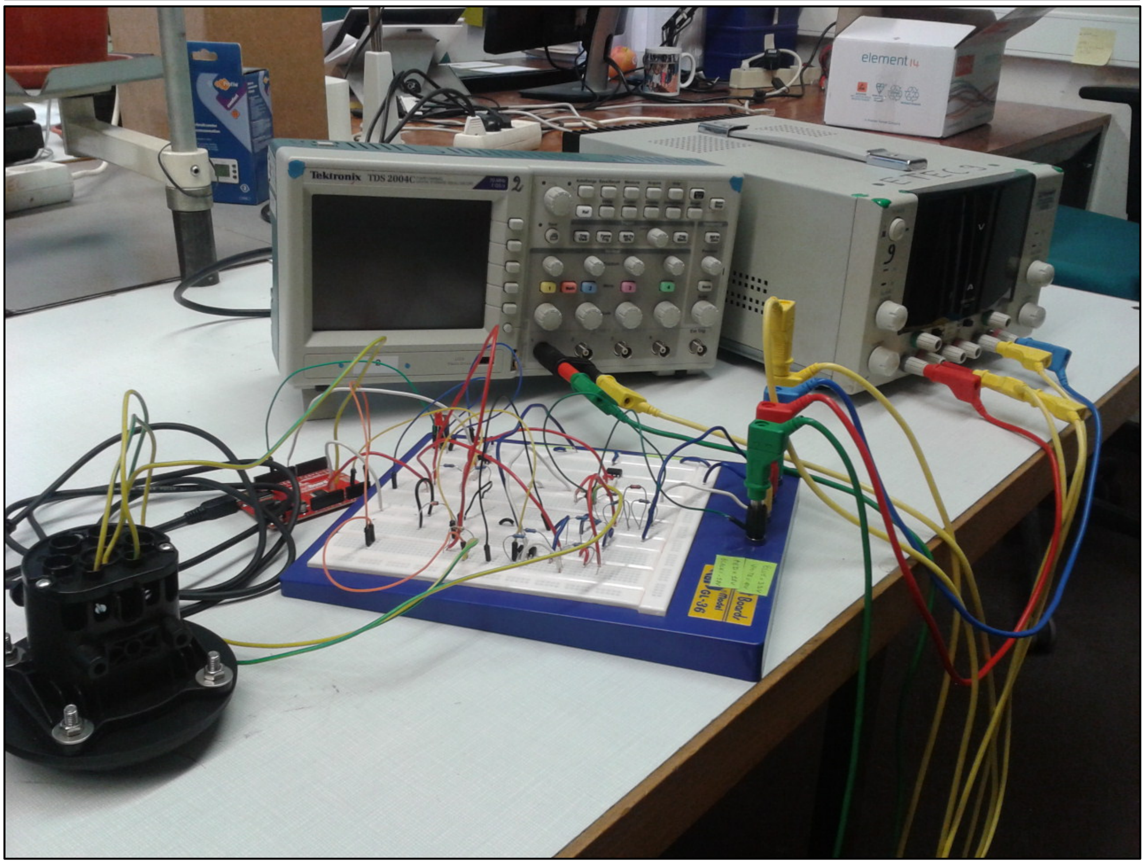
The last thing we can see in the left is the header H1, where we will connect the Control Pilot and the Proximity of the plug, and the header H2 where we will connect the +12V and -12V generated by the external power supply:



Between the +12V and -12V we will place a capacitor of 15uF (C3) for making the voltage of the power supply more stable. Two diodes will be used to protect the circuit from the Control Pilot, so they will not allow a voltage higher than +12V or lower than -12V.

In the next pictures we have a view of the real circuit and the entire assembly with the external power supply and the oscilloscope:


CONTACTOR
ADC
PI CABLE
BLUE 3.3V
WHITE 5V
RED 12V
BLACK -12V
Boards
Model
GL-36


Tektronix TDS 2004C
element14

2.5 Manufacturing process of the printed circuit

Once we have tested that the circuit is well planned and fixed, we have printed it on a spot to secure all the electronic components and connections between them.

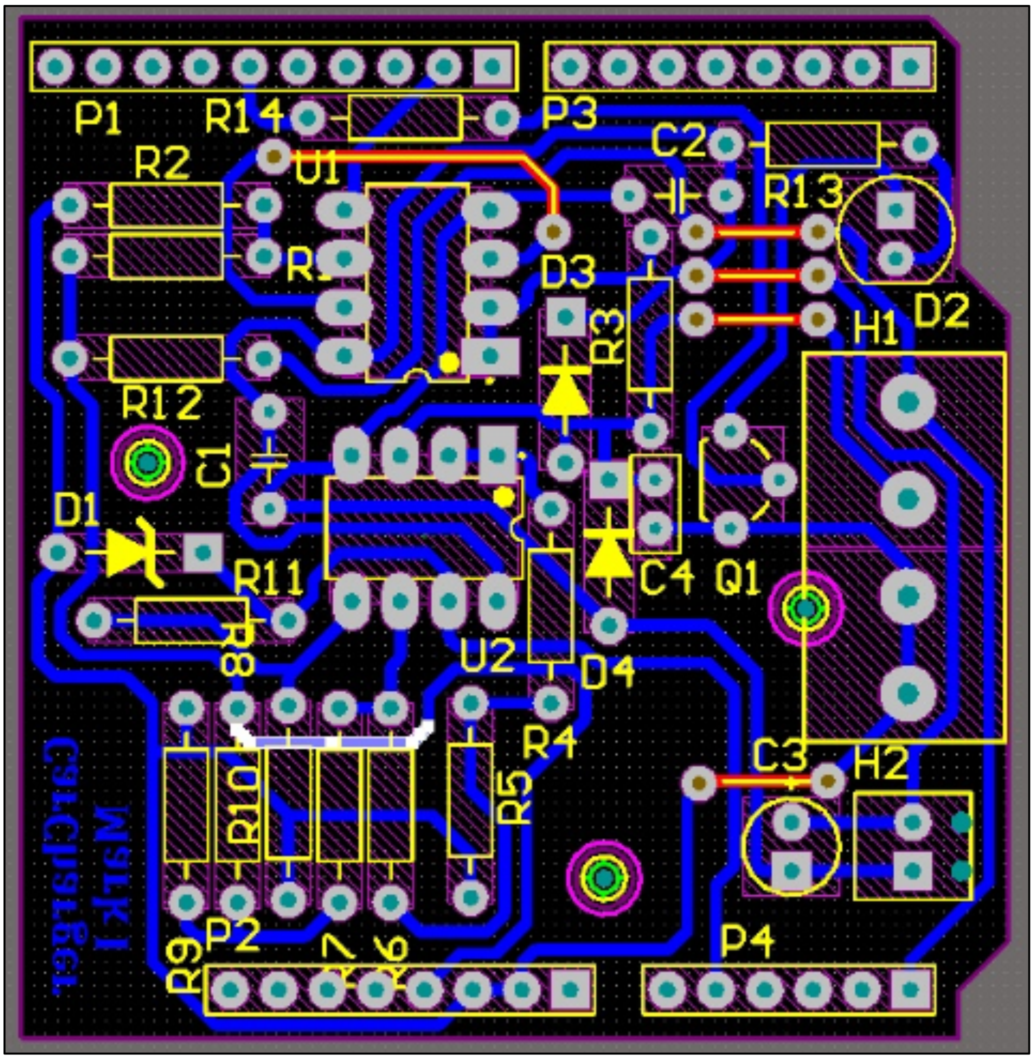
2.5.1 Board design

For this process, we have used a specific software package for printed circuit board called Altium Designer.

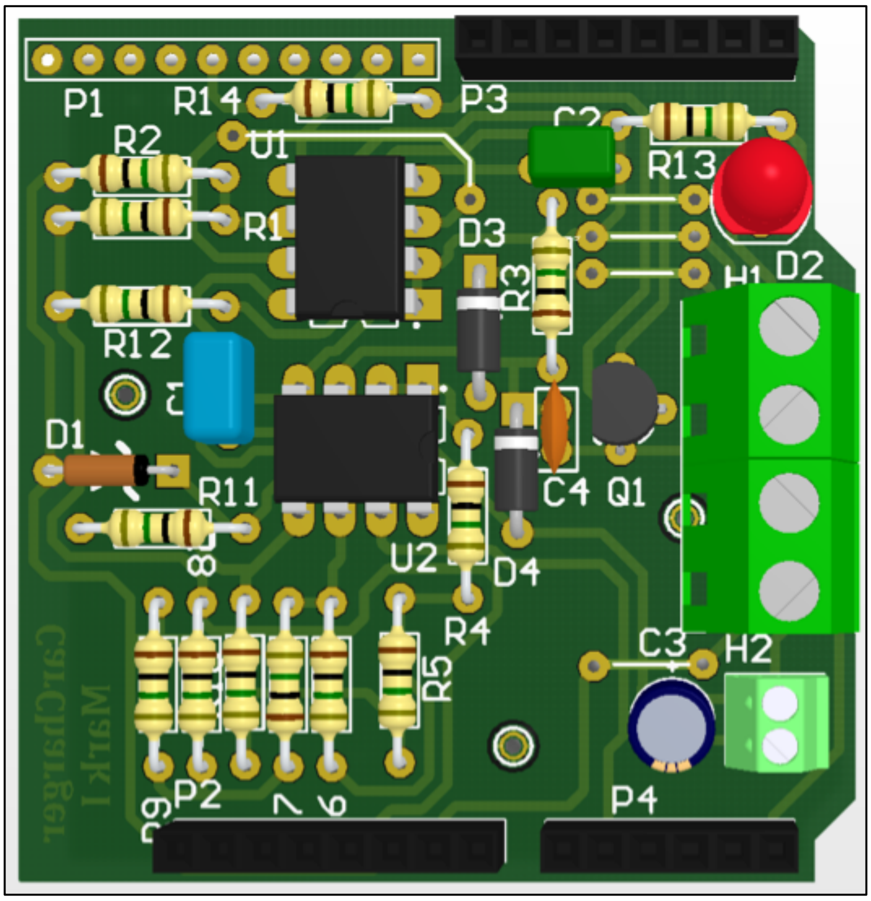
There are two different steps that need to be done with this software:

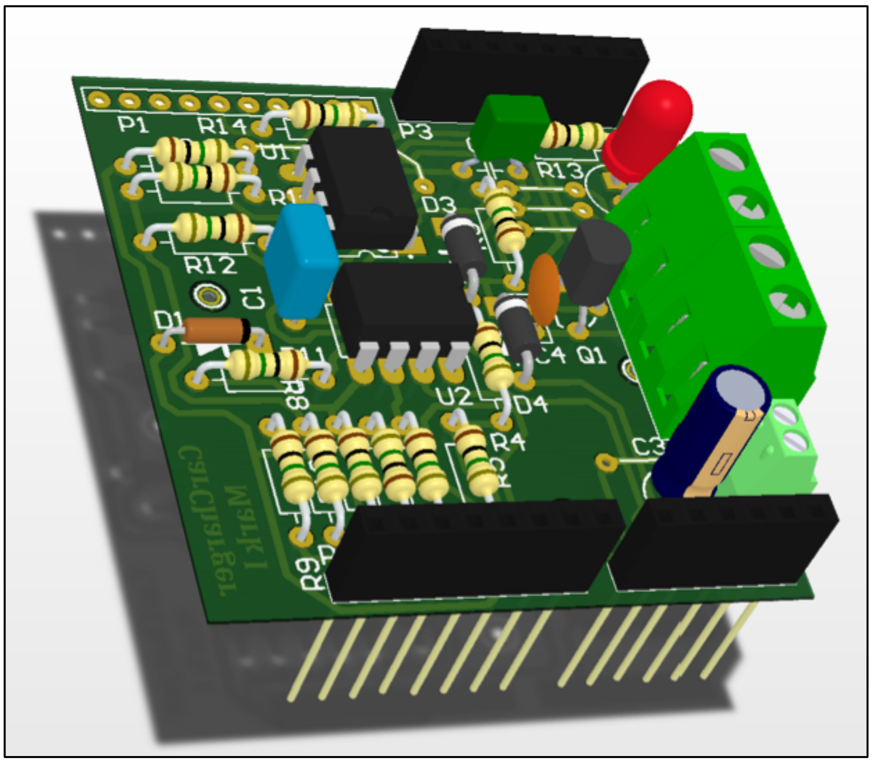
1. Draw the elements and connections in the Schematic module.
2. Design the printed circuit board in the PCB module.

Our design of the board is the following:

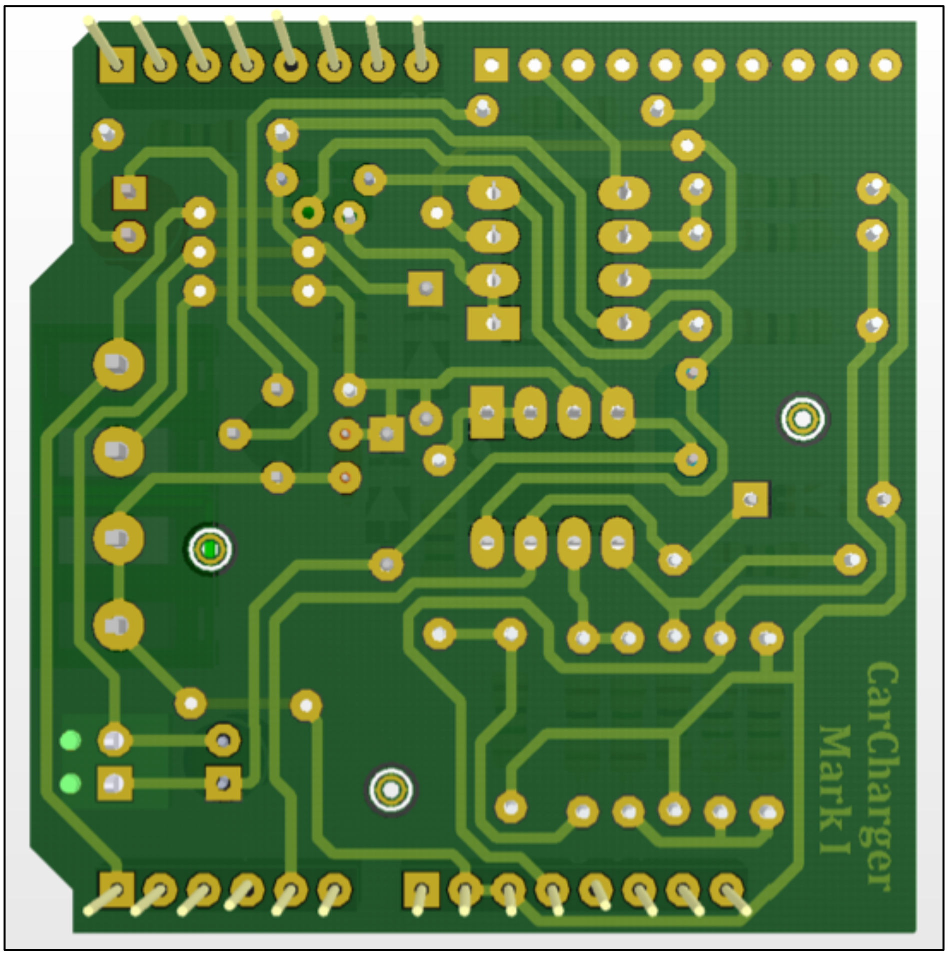


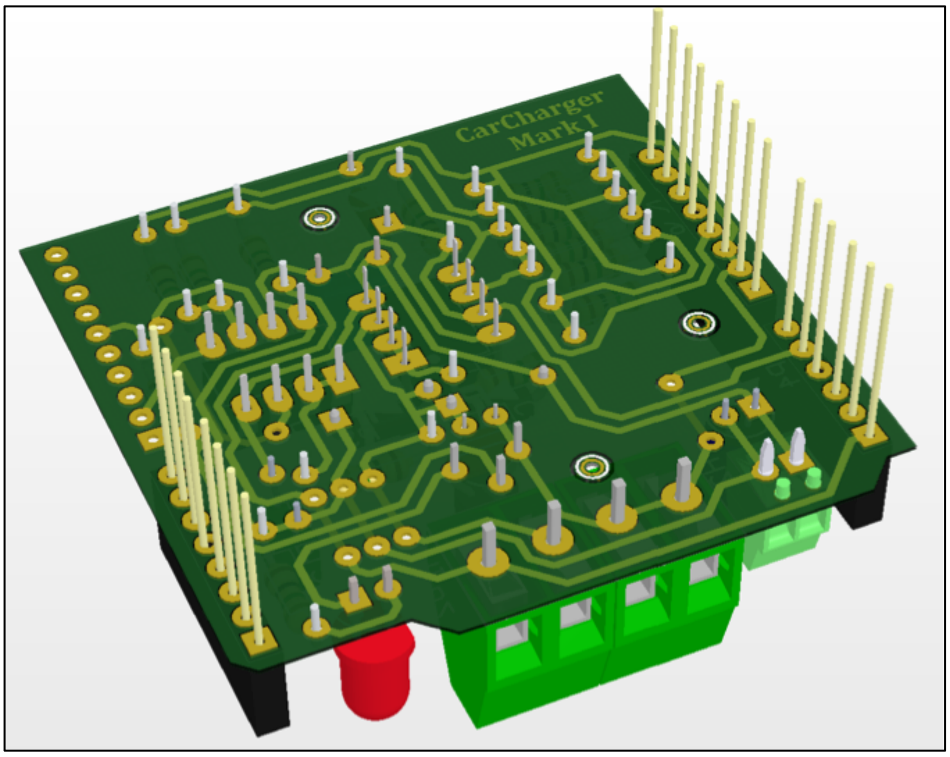
And here we can see a three dimensional view of the top and the bottom side of the PCB:





Top part of the PCB





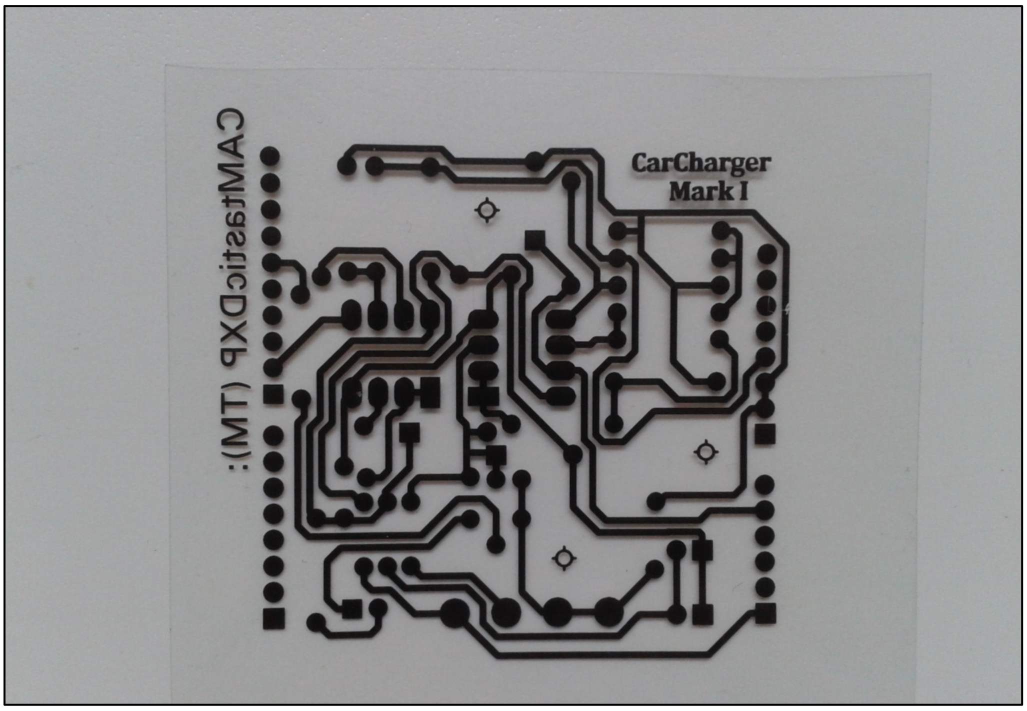
Bottom part of the PCB

2.5.2 Physical manufacturing process

Once the PCB is designed with the software, we are ready to build it. First we create it on the a special copper coated material cutting its profile and making the holes with a drilling machine as we can see in the next picture:

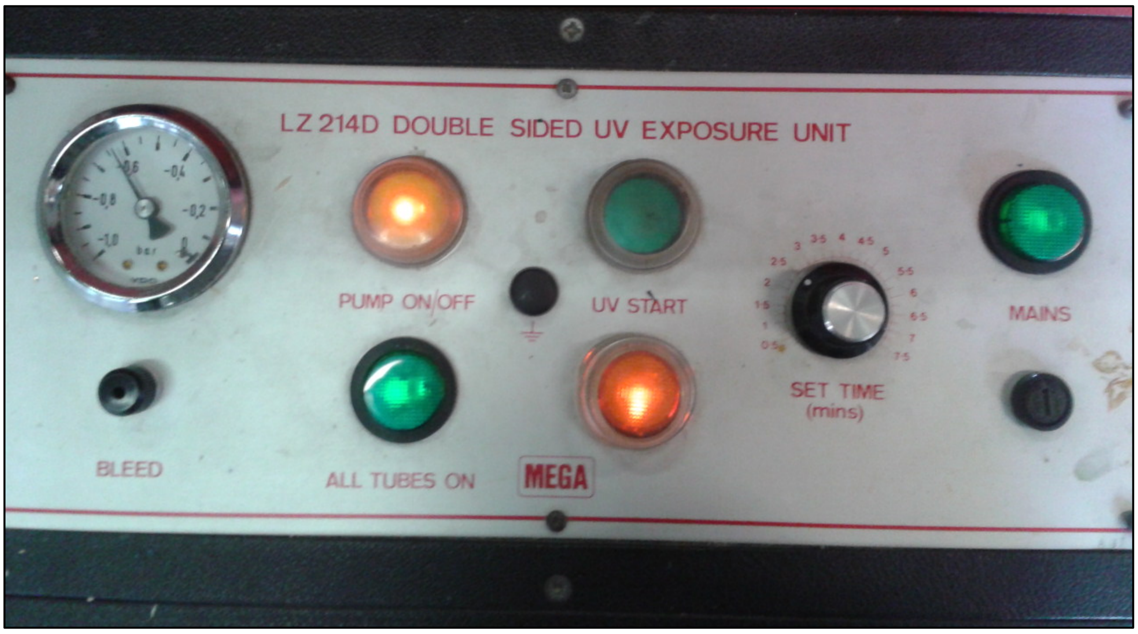
OPERATE ONLY WHEN THE VACUUM-SYSTEM IS SWITCHED ON!
UP
HOME
FREESDIEPTE INSTELLEN MET RINGEN OP TOOLS

After that, we need to print the PCB layout in a transparent sheet using an etchant resistant material. The Laser printers´ ink is the most commonly used material and the one we will use. However, we should ensure that the circuit is mirrored (the Altium Designer have this as an option when printing):

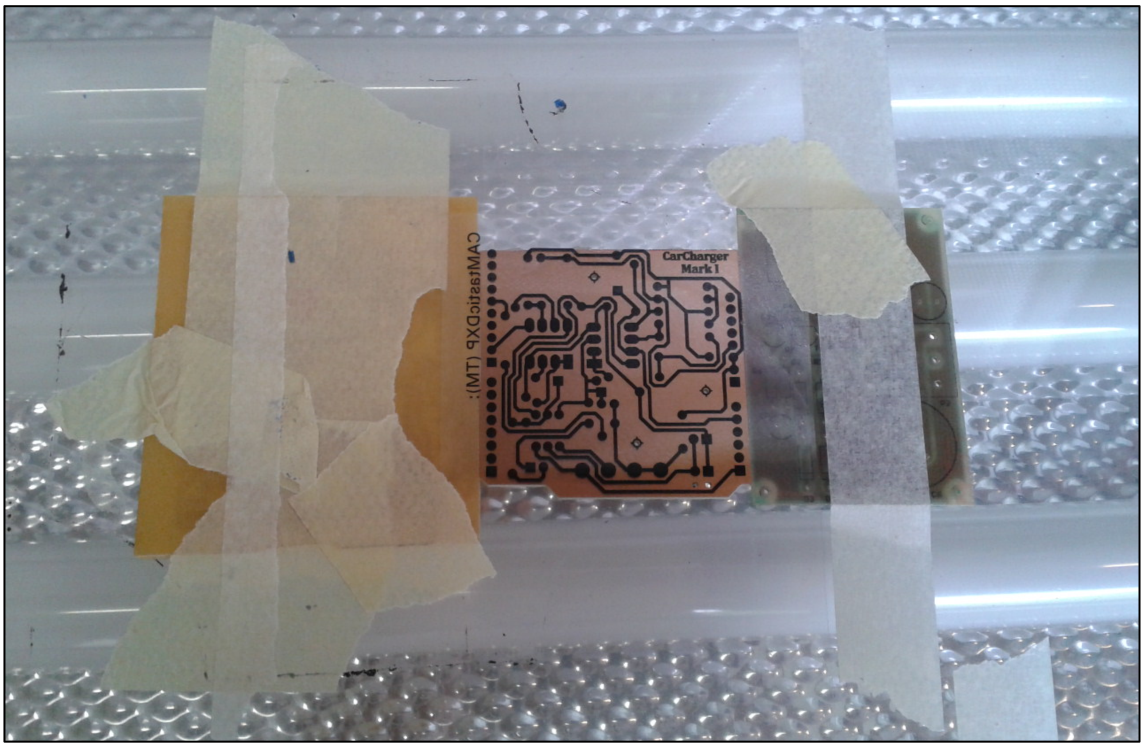


The next step is to place this copy above the board making sure that they fit perfectly and fixing them together with some tape. Now we can introduce them in an Ultraviolet machine and we will expose them to UV light for 2.5 minutes with a pressure of -0.6bar:

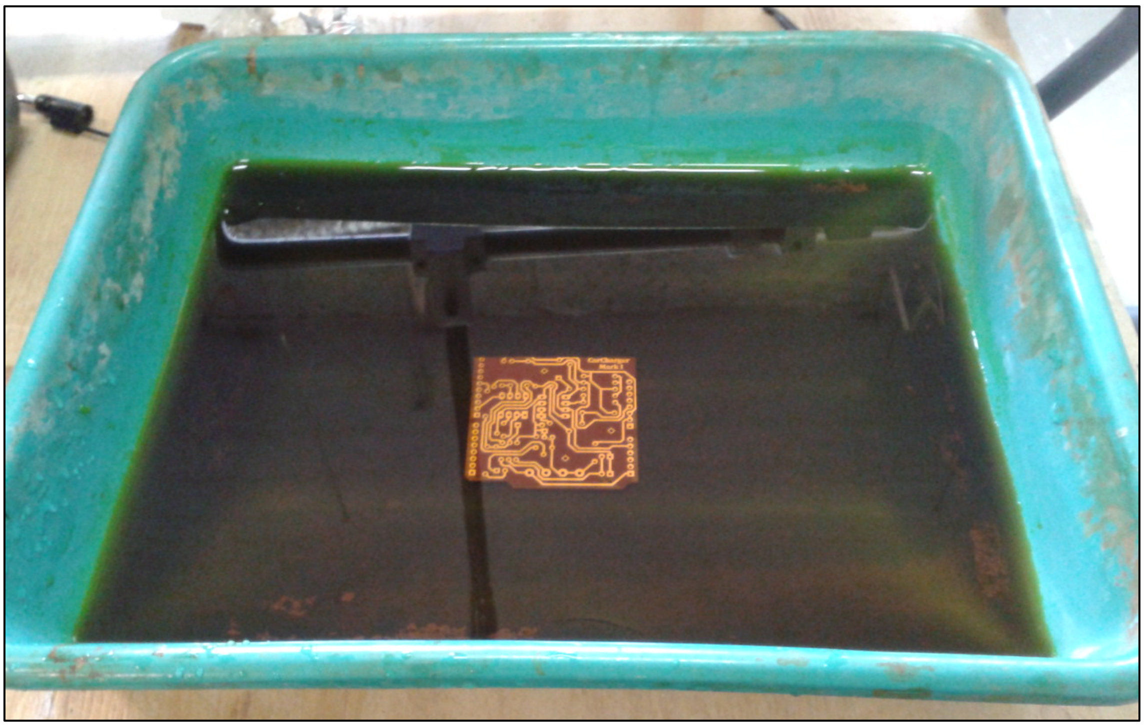
We can more or less see in the pressure indicator (on the left) the -0.6bar and in the setting time indicator 2.5 min:



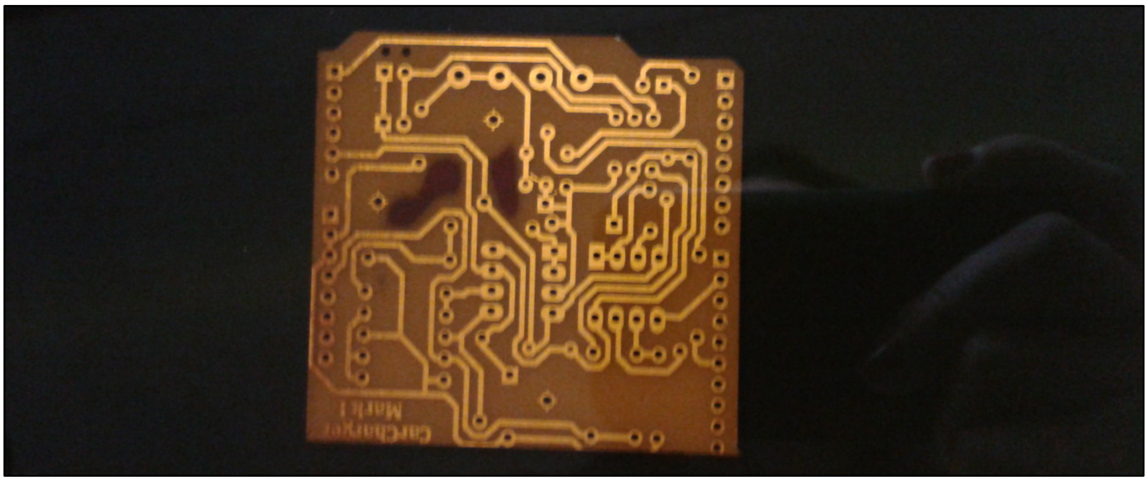
In the following two pictures we can appreciate how the copy of the circuit fits perfectly with the board and how we place them into the UV machine (both are fixed with tape):



After this process, the tracks and the holes will be printed in the board. We will then remove the paper and fix the PCB. This process will harden the part of the PCB that was not exposed to the UV light washing off the remainders from the PCB. This is the most dangerous part of the process because of the chemicals that we will be using, so additional protection will be extremely recommended for preventing chemical burns, problems due to the inhalation of the chemical products and splashes in the eyes. We will first introduce the board in Sodium hydroxide (NaOH) for around one minute for dissolving the waste of the ultraviolet light. It is important that the container has to be made entirely by plastic or glass:

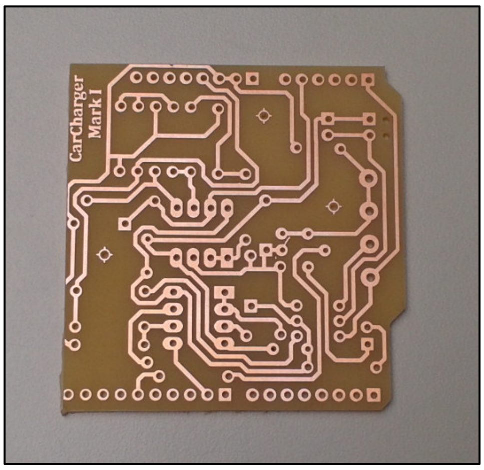


We will later wash the board with water and we will appreciate that the PCB structures have became golden, while the rest of the board shows plain copper. The second part will be to submerge the PCB on ferric chloride (extremely toxic) for about 10 minutes for destroying the remaining cooper:



The amount of etchant needed and how long the process takes will depend on different aspects as the size of the PCB, the amount of copper on the board, etc. Constant motion in not required but it will help, so we will try to shake the recipient that contains our PCB but ensuring that it is in constantly in contact with the solution. Once the reaction has begun, the copper will begin to disappear and as we can see in the picture above just some black spots remain in the board. Usually, the solution will clean the PCB beginning from the outside of the board to its middle.

When all the superfluous copper has been etched away, we will remove the board and clean it with cold water. The last step will be to clean the PCB board with Acetone and the final result will be the following:



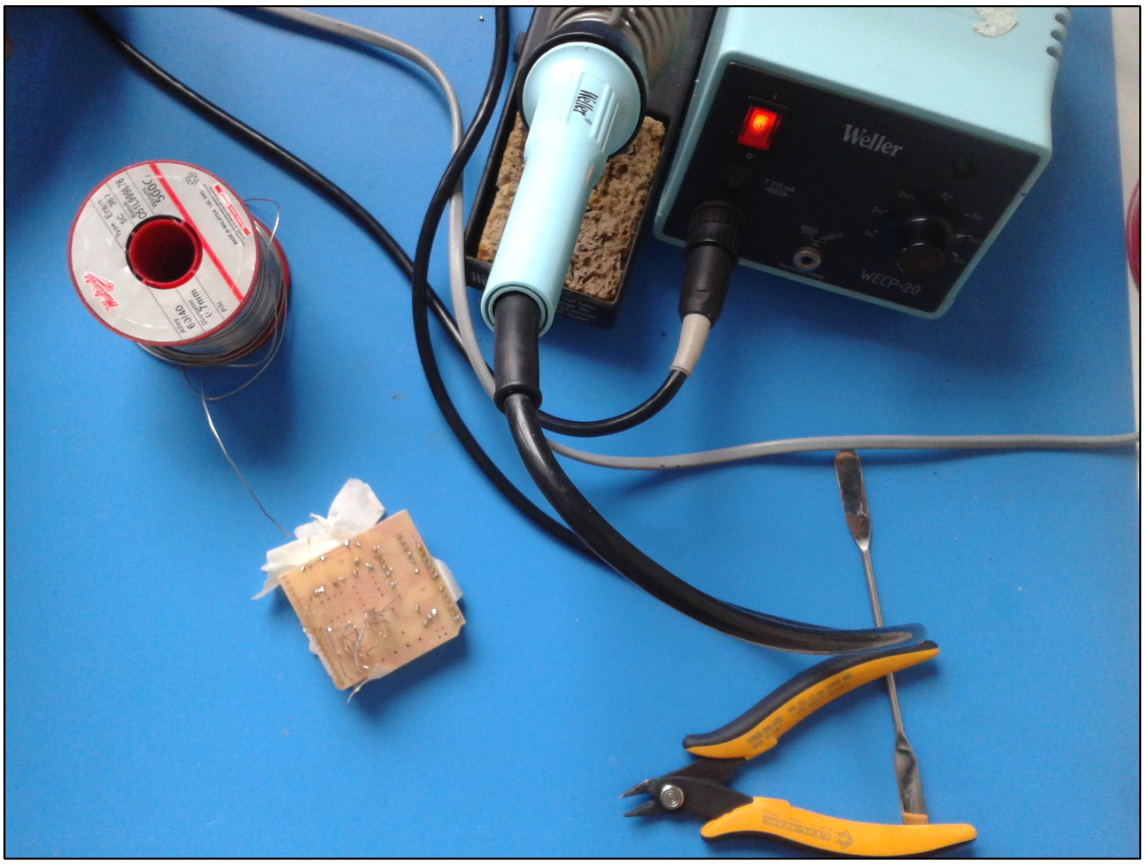
We can now place all the components in their place and start the soldering process of through-hole components (those which have leads that pass through the holes in the board and are soldered to the pad) into the printed circuit board. Like this, we will get a permanent connection between the electronic components and the PCB.

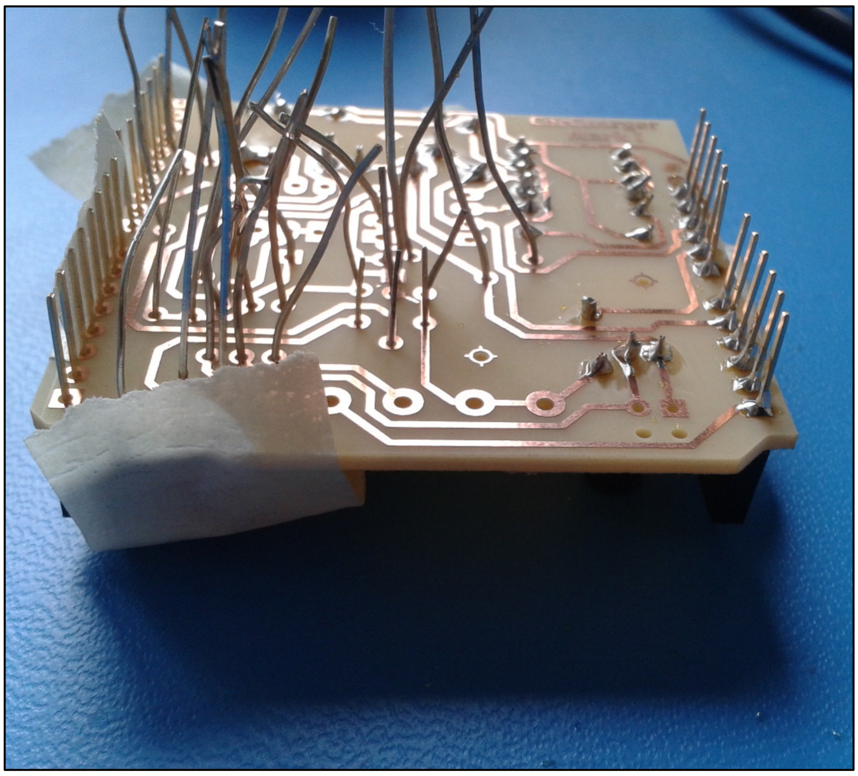
We will use some tape for fixing the components to the board and once the iron is heated (around 300 ºC), we can start to solder with the iron in one hand and the coil of solder in the other (flux-core solder with a diameter of 0.7 mm).

When soldering leads into circuit boards we want to heat the metal contact on the board and the lead itself. Applying too much heat can damage the circuit board or even the components. We will touch the tip of the iron to the crack between the lead and the metal pad on the board and after waiting a couple of seconds, we will dip the tip of the solder into the joint and placed a very small amount of solder at the connection (no more than the head of a pin). Once the solder has pooled a bit and it has soaked into the joint, we can remove the solder wire and then the iron (so the tip of the solder does not get stuck to the joint). The solder will begin to harden as soon as we remove the iron.

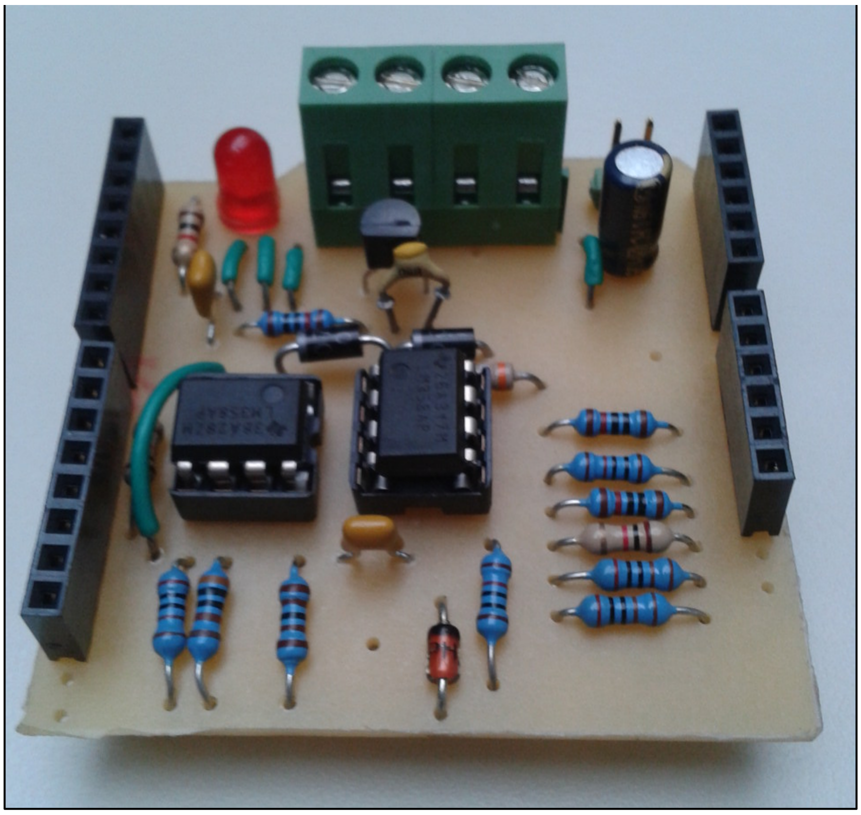
Using the proper amount of solder is very important while soldering small components. So if we apply too much solder and it pools up outside of the metal pad, it can cause a short. Too little solder and the component will not make a good connection with the circuit board and might not work the way we want it to. When getting the right amount of solder, it should look like a small cone with its base on the circuit board.

Here we can see some pictures of the soldering process:





And the result of the finished board is the following:

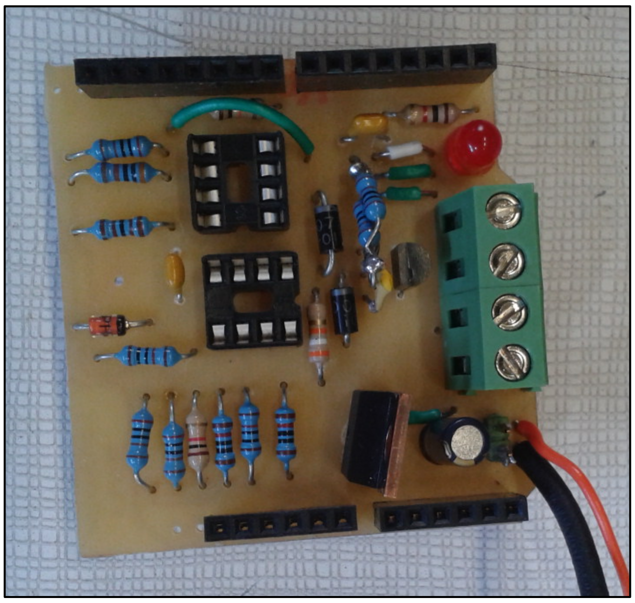


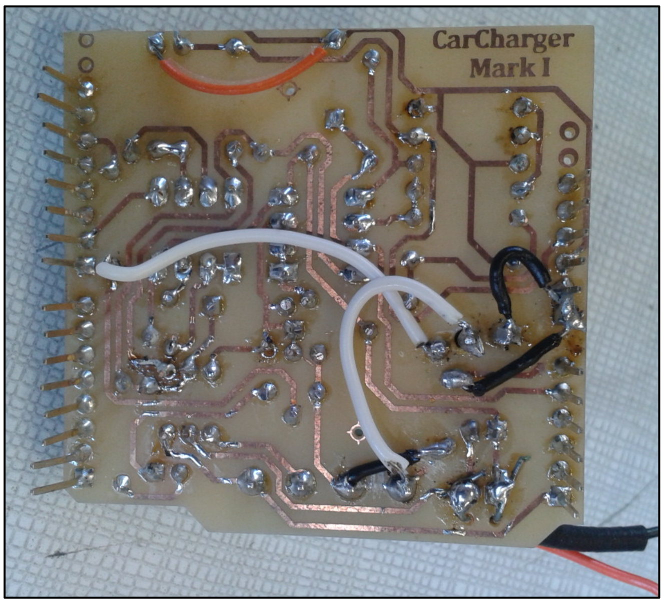
3. PRESENTATION OF RESULTS

3.1 First PCB design

Once the PCB is ready, it is time to test it. Before testing it on a live vehicle it is convenient to check that everything works as supposed with an oscilloscope.

We then realised that we had to add a Mosfet IRF540 and that we were missing an output for the contactor. We could manually make some holes and place the mosfet in the board and connect the gate to the Base of the transistor BC547C, the Drain to the contactor and the Source to the ground. As two pins of the headers were connected to the ground, we could break some tracks and leave one of them available for connecting the contactor. Even though making some manual changes after the manufacturing process it is not very advisable, it is cheaper and faster than rebuilding the entire PCB. And here we can see the result of these manual changes:



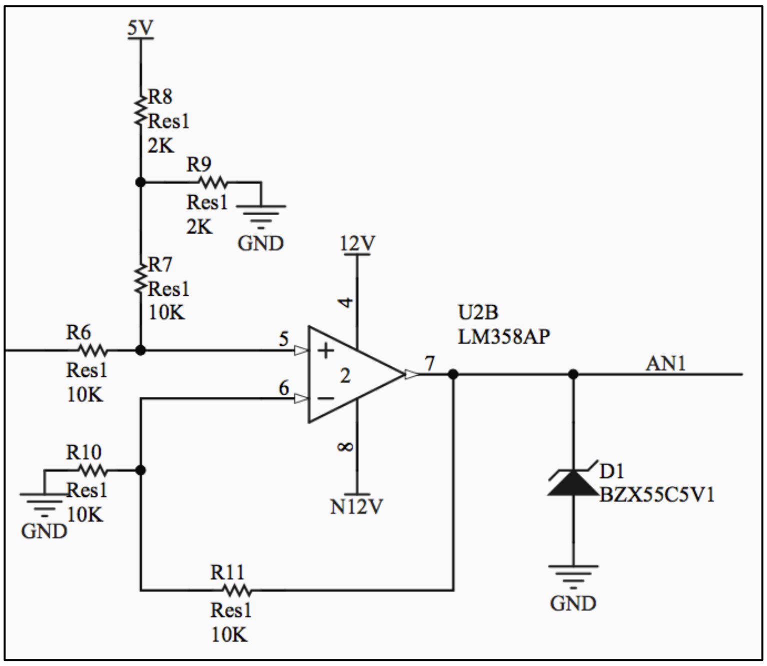


However, we realized while checking the voltage in different points of the board with an oscilloscope that the inverting and non inverting input terminals of the fourth op amp were switched. Therefor, we decided to redesign the PCB including the changes we had made and start the testing process again.

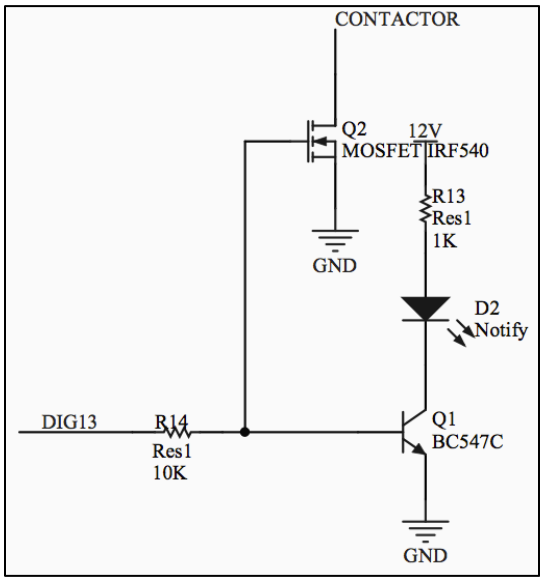
3.2 Second version of the PCB

The new circuit has three changes:

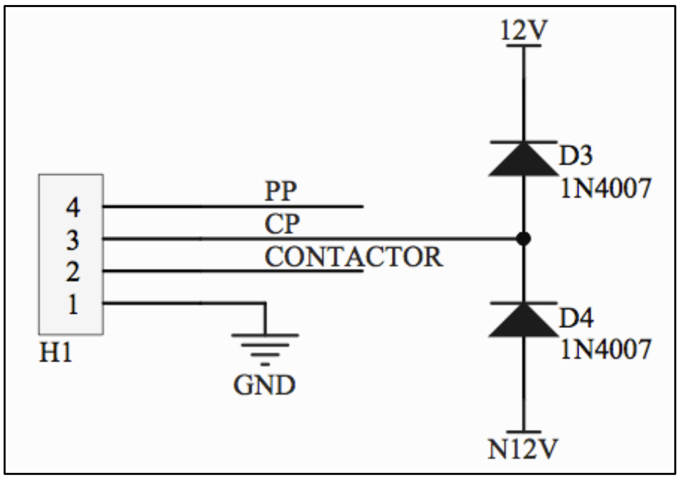
- Changing the inverting and non-inverting input terminals. As we can see in the following image, now the positive input is connected between R6 and R7:



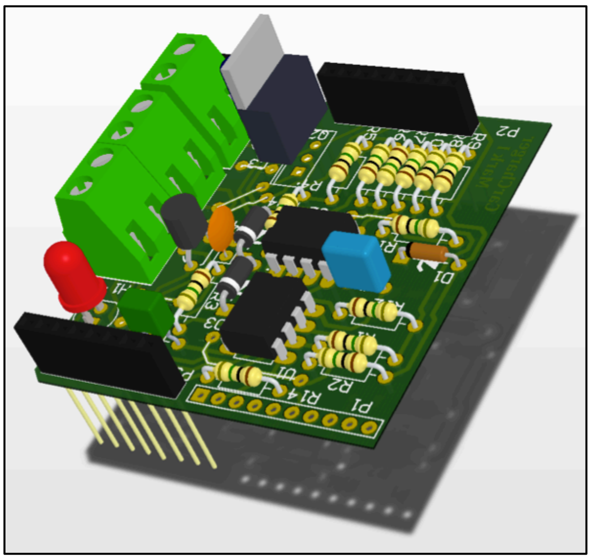
- Adding an IRF540 Mosfet (Q2) connecting the Gate between the resistor R14 and the Base of the Q1 BC547C transistor and the Drain to the contactor (LC1D12). Like this, when the Digital Pin 13 is High (State C), current will pass through the electromagnet producing a magnetic field, which attracts the moving core of the contactor and it will close. The electromagnet coil draws more current initially, until its inductance increases when the metal core enters the coil. The movement of the contactor is propelled by the movement of the core (the force developed by the electromagnet holds the moving and fixed contacts together). Nevertheless, when the Digital Pin 13 is Low, the contactor coil will be de-energized and the gravity or a spring will return the electromagnet core to its initial position and the contactor will open.

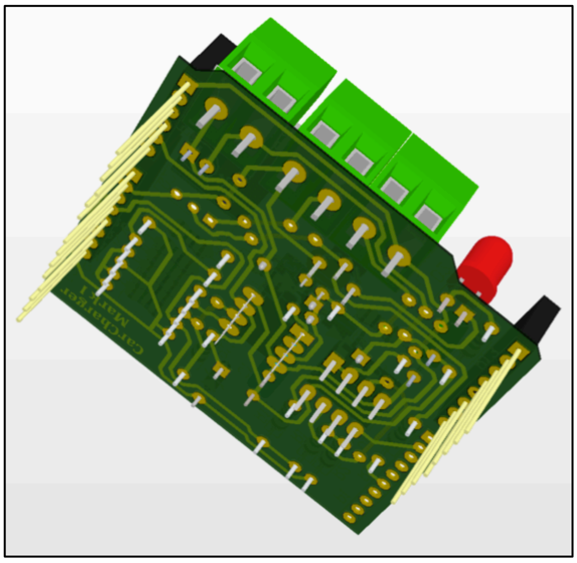


- Adding an output for the Drain of the Mosfet where the contactor is connected. As the pins 3 and 4 of the header H1 were connected the Ground, we have used the Pin 3 for connecting the contactor. Also an extra diode will be connected between the +12V of the contactor and the input of the contactor (in the picture of the contactor we can see a yellow cable where it is placed. The black mark in the cable indicates the orientation of the diode).

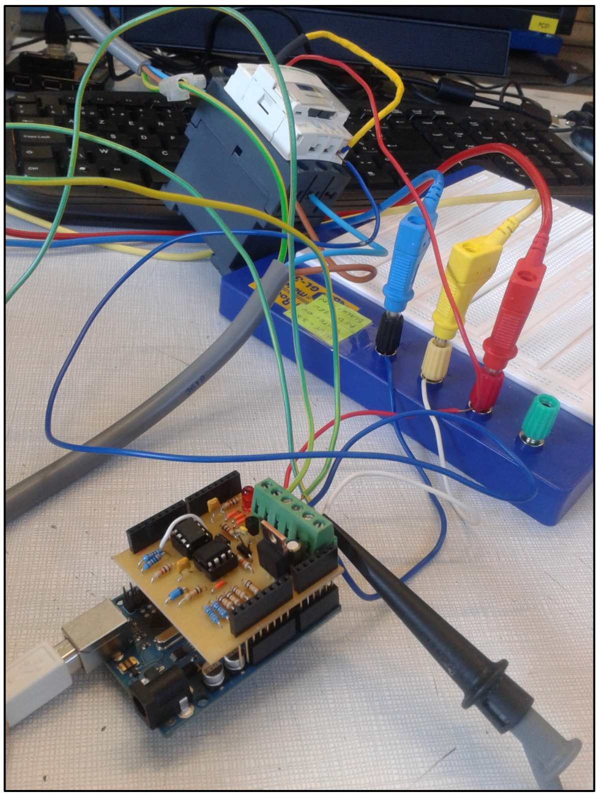
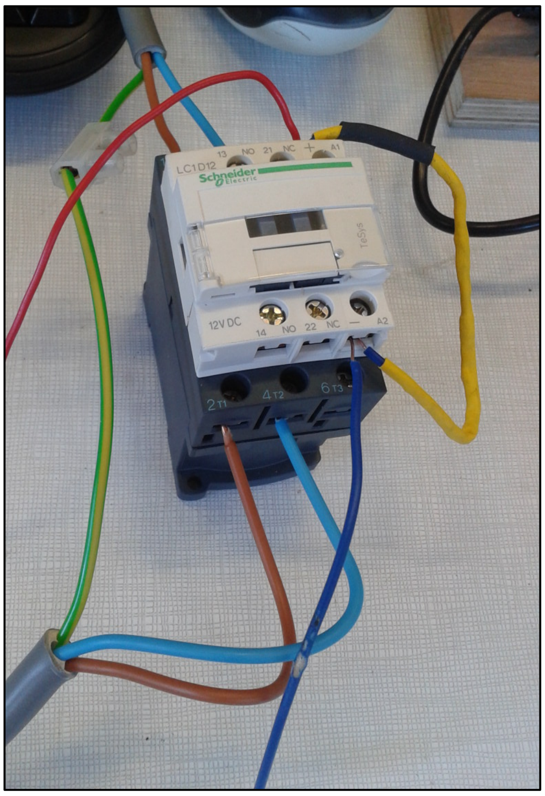


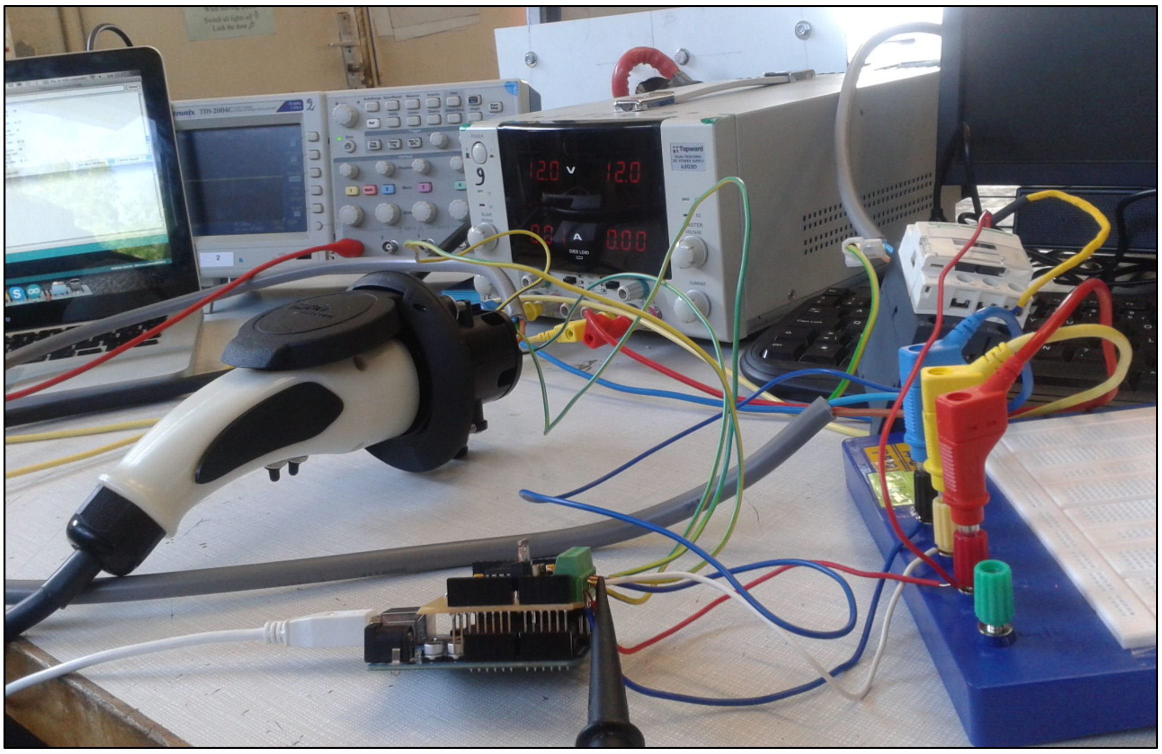
After making these changes, the PCB 3D design has the next look:





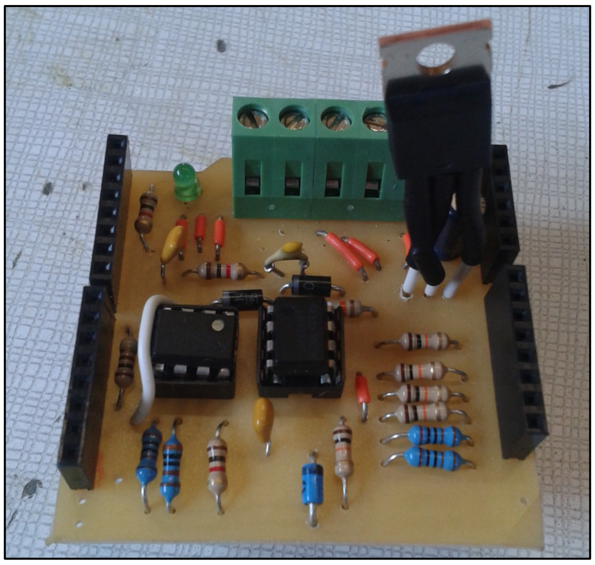
Then, we manufacture and test the PCB again. In the three following pictures we can see the assembly of the PCB above the Arduino board and the connections with the contactor, a more detailed zoom of the contactor and the entire assembly with the computer connected to the Arduino, the PCB, the contactor, the testing socket outlet, the switch, the oscilloscope and the power supply:

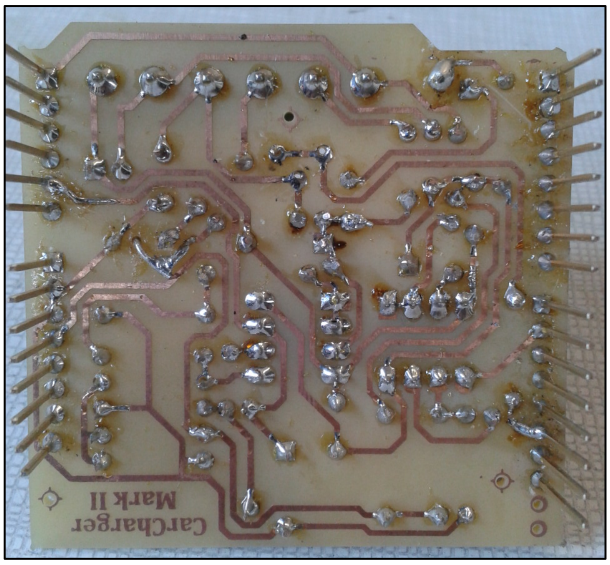





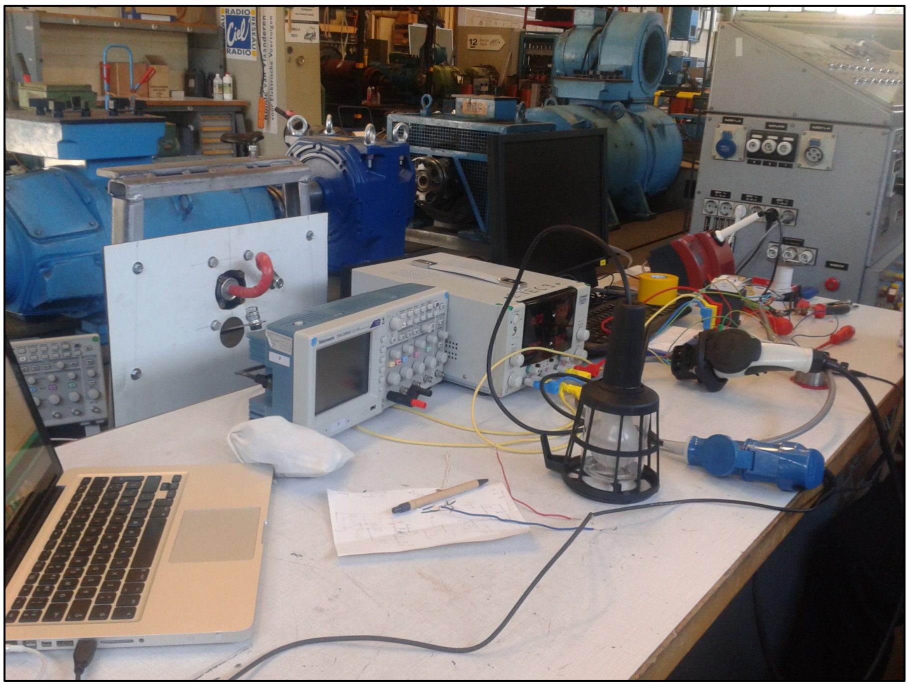
When analysing the results, the voltage values in the Control Pilot were correct and the PWM signal was generated in the State B and C but the LED and the contactor were not working correctly. At the beginning we though that the LED was not on because maybe the transistor was consuming too much power, so we took it off but then we realized that this was not the problem because even though in the State C the digital Pin 13 was High, after the resistor R14 (connected behind pin 13) the voltage was 0V. This was the reason why the contactor was not closed and the LED was never on because the Gate of the mosfet and the Base of the transistor were not receiving voltage. The resistor R14 (10kΩ) was consuming all the voltage, so the solution was to replace it for a smallest one of 100Ω.

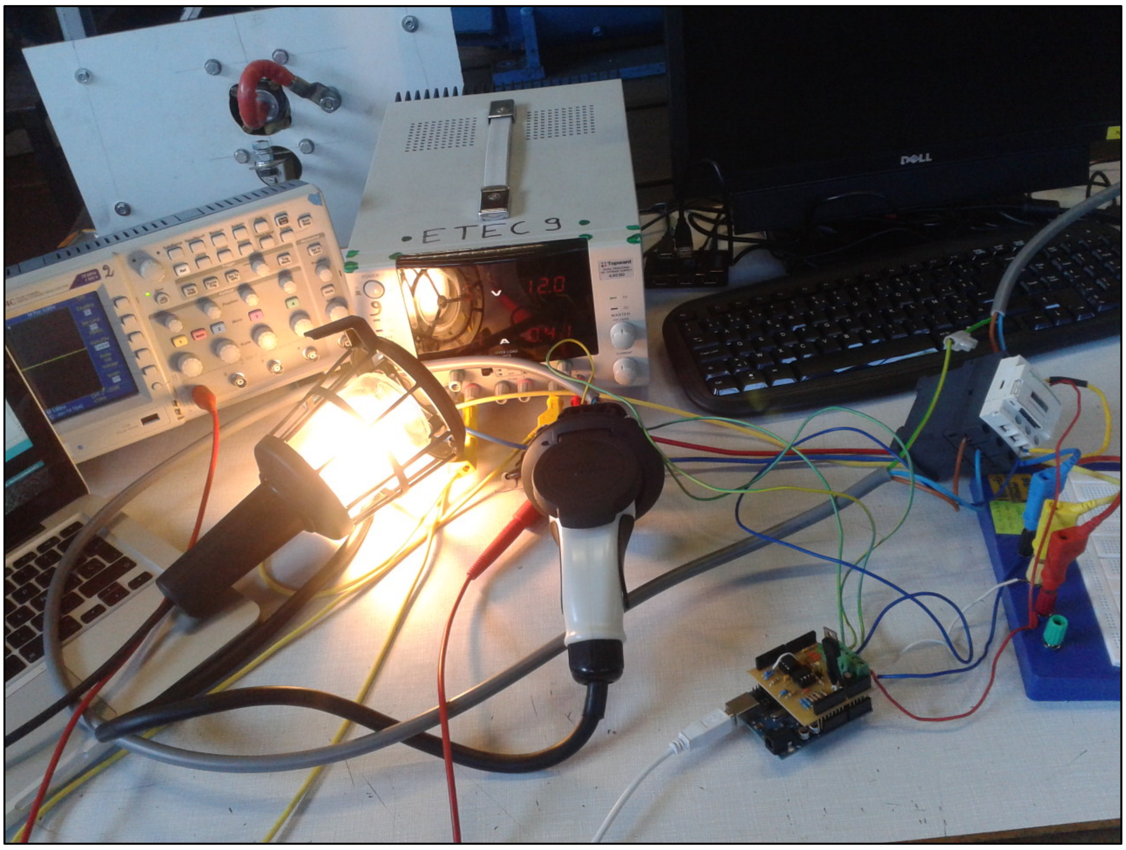
Once we made this change manually, the contactor was still not closed in the State C due to a switch of the Gate and the Drain of the Mosfet. We removed the mosfet and added some cables to change them and this is how the PCB looks:





We then connected the lamp to the assembly and successfully, as we can see in the next picture, it was lighting in the State C:





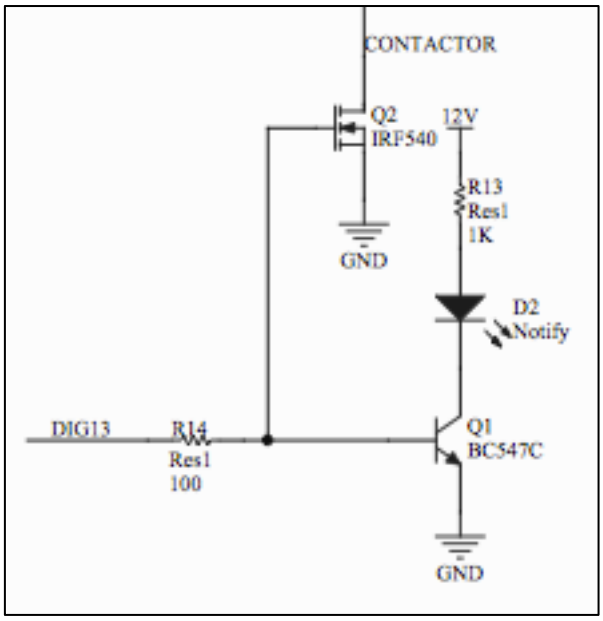
The final test it to plug the electric car and check if it is really charging. The first time we tried it was not charging because the frequency of the PWM signal that we were producing with the Arduino had a frequency around 500Hz and the car will only recognize signals with a 1000Hz frequency. Also it is very important to check the values of the voltage in the Analog Pin A1 every time we manufacture a new PBC because the voltage range to determine the different states of the charging process can vary due to use another resistances that will have a different value.

Once we made these changes in the code and we connected the contactor to two phases of 230V, it was successfully charging:

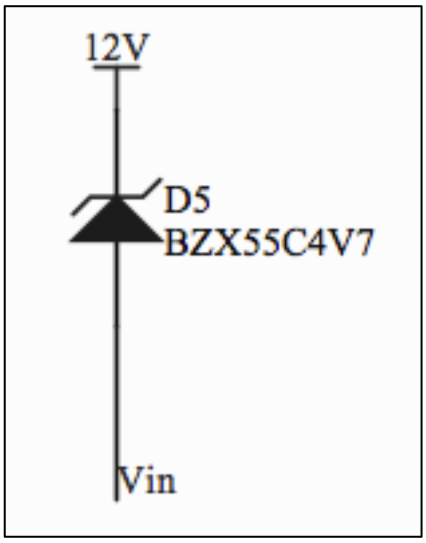
3.3 Third and final version of the PCB

We create the final PCB including the following changes:

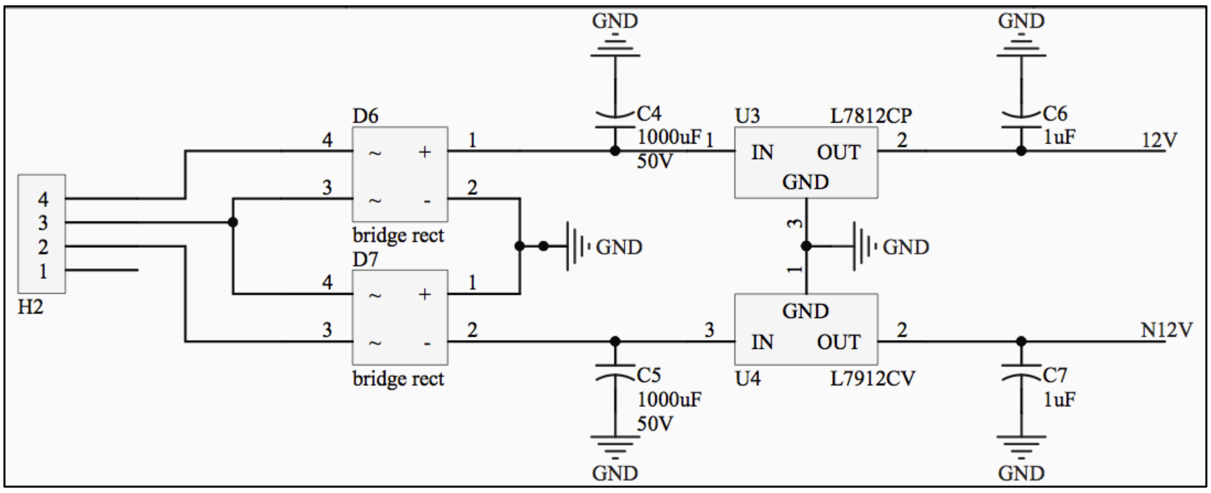
- The value of the resistor R14 is finally 100Ω:



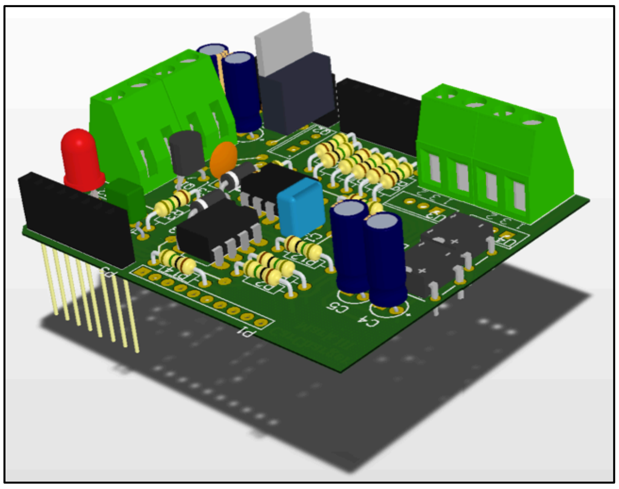
- We connect correctly the pins of the mosfet.
- For powering the Arduino without the need of the computer we add a 4.7V Zener Diode between +12V and the Vin Pin of the Arduino board:

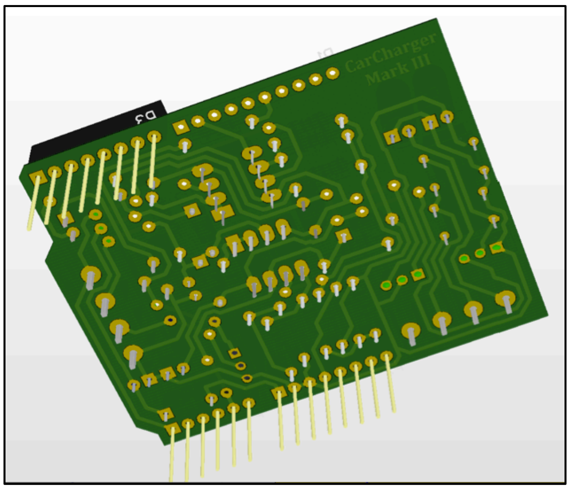


- For avoiding the need of an external power supply, we produced +12V and -12V with a transformer, two bridges rectifier, a 7812 series voltage regulator, a 7912 negative voltage regulator and some capacitors as we can see in the next diagram:

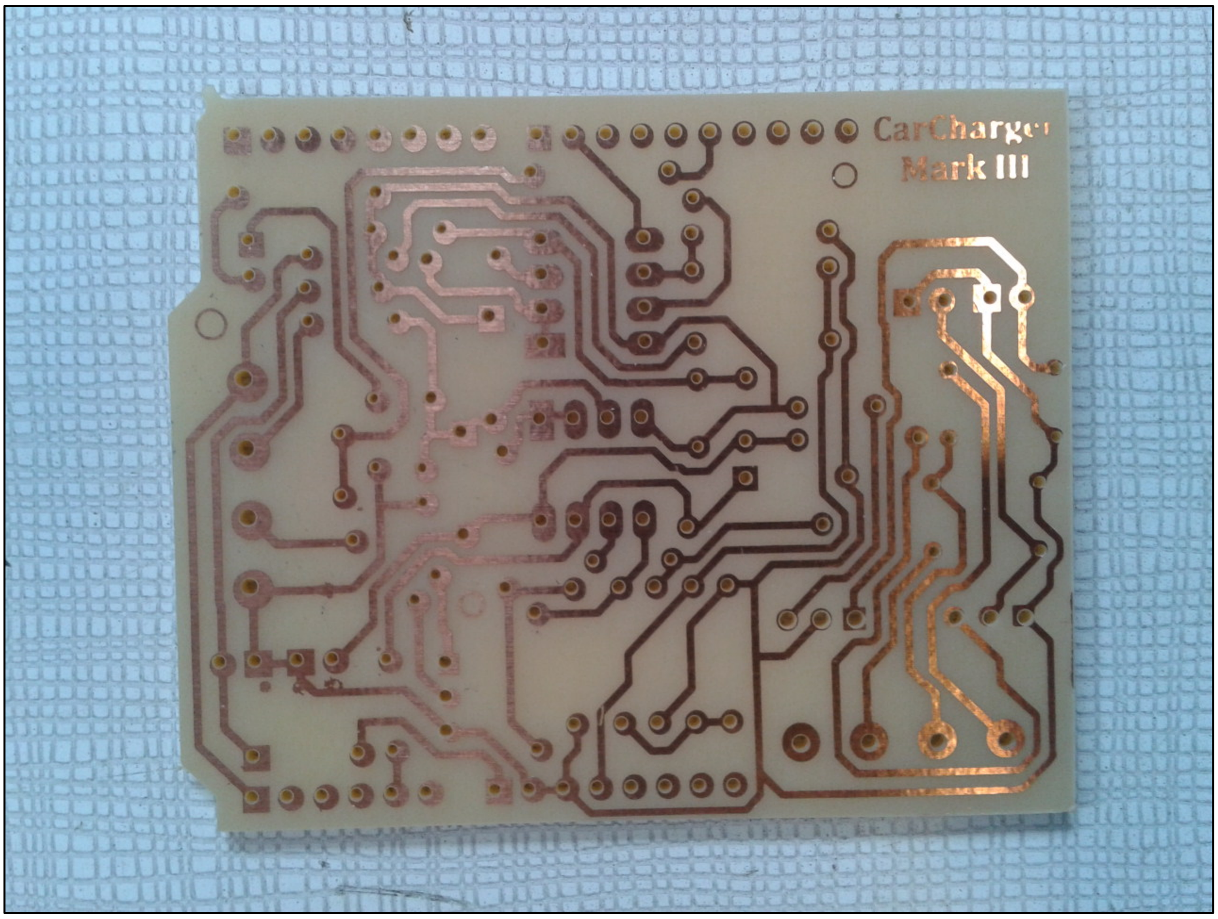


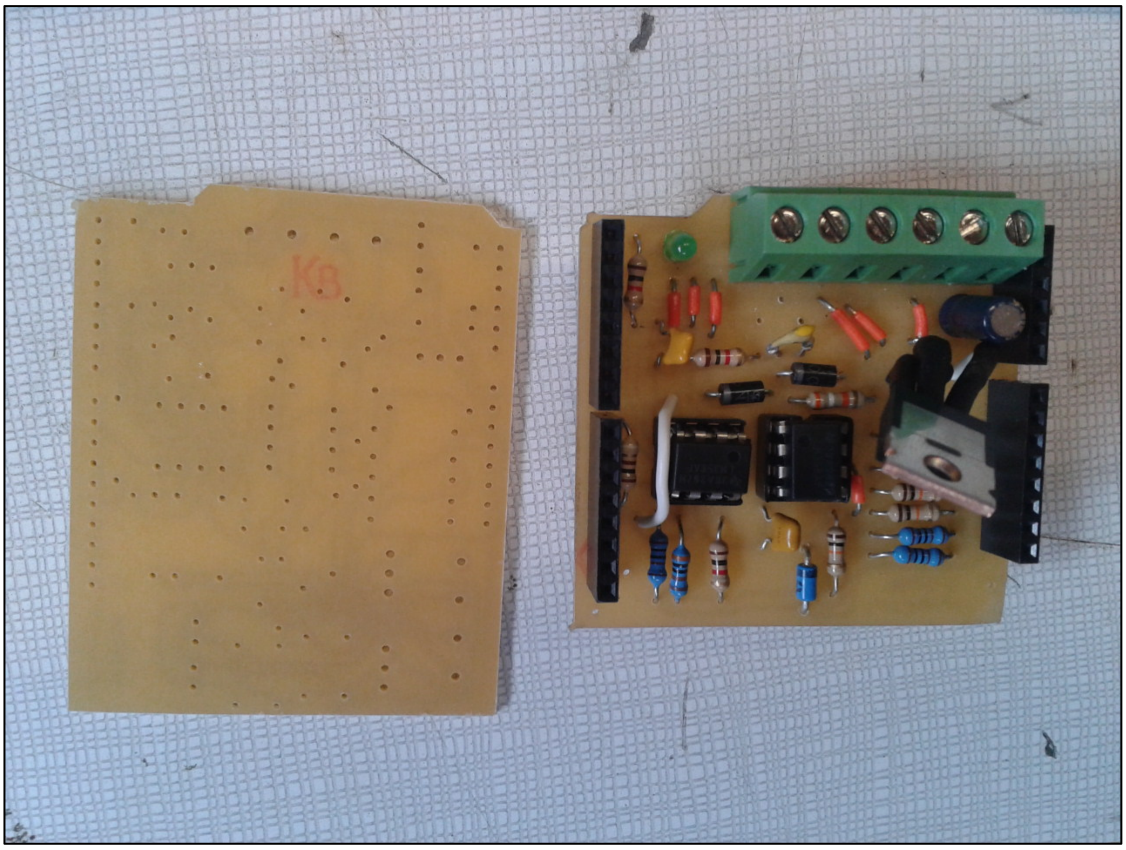
Our PCB computer design has the following appearance:





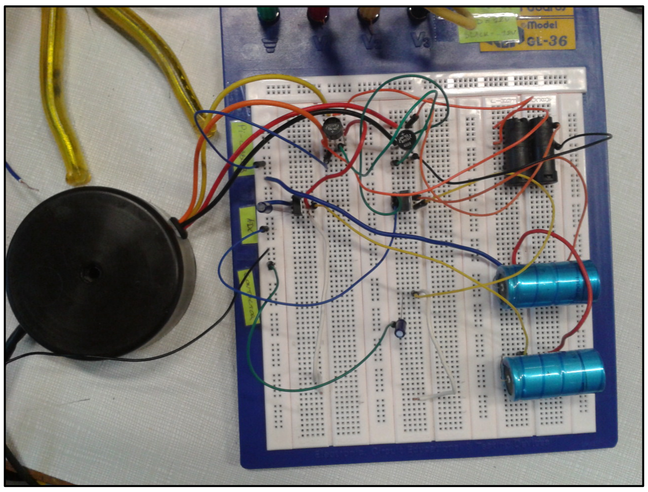
And here is the real PCB. This version is a little bit bigger than the previous models due to the additional components:





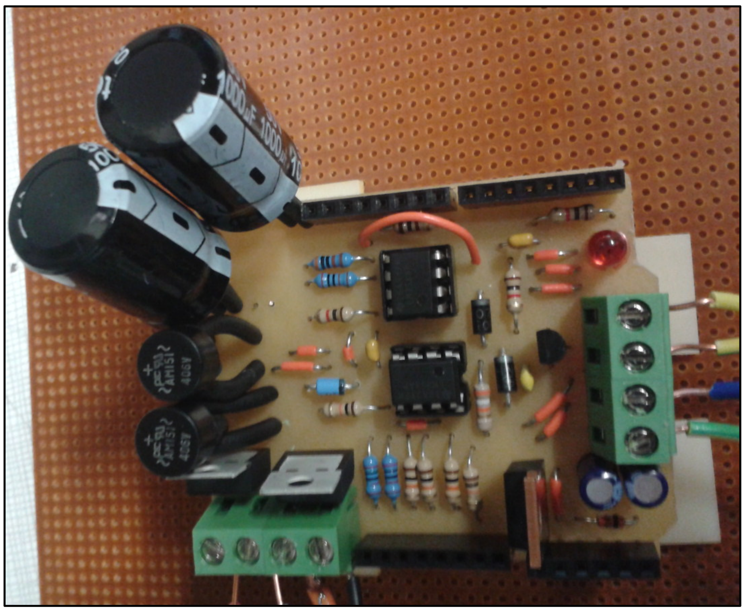
The PCB on the left is the 3rd design and the one the right the 2nd design

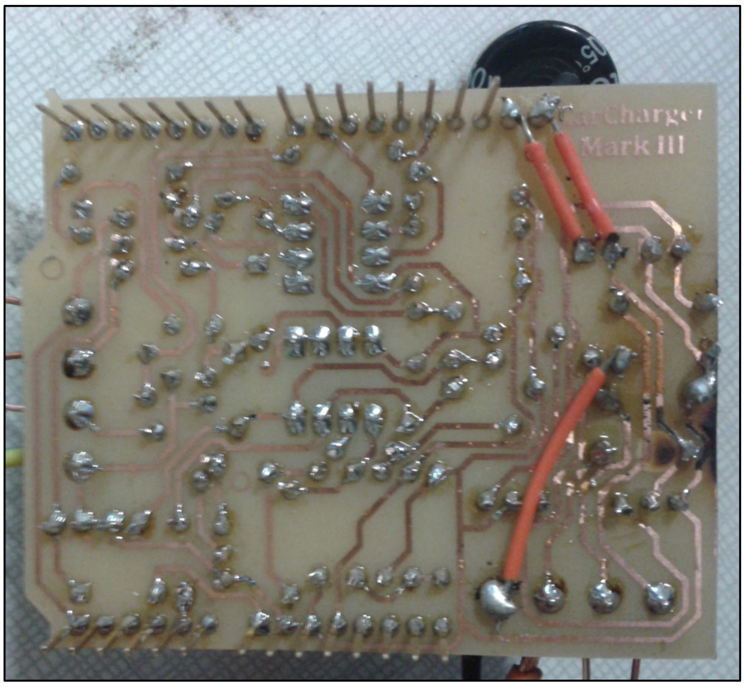
Before soldering the part of the circuit where we get +12V and -12V, we build it on a breadboard to make sure that it worked. We also added some fuses before the rectifier bridges. We used 680uF / 25V capacitors and when testing the circuit they explode. Even though we then used bigger capacitors (1000uF / 25V), they were still not good enough for our application. In the following picture we can see the assembly of the circuit in the breadboard and the connection with the transformer:



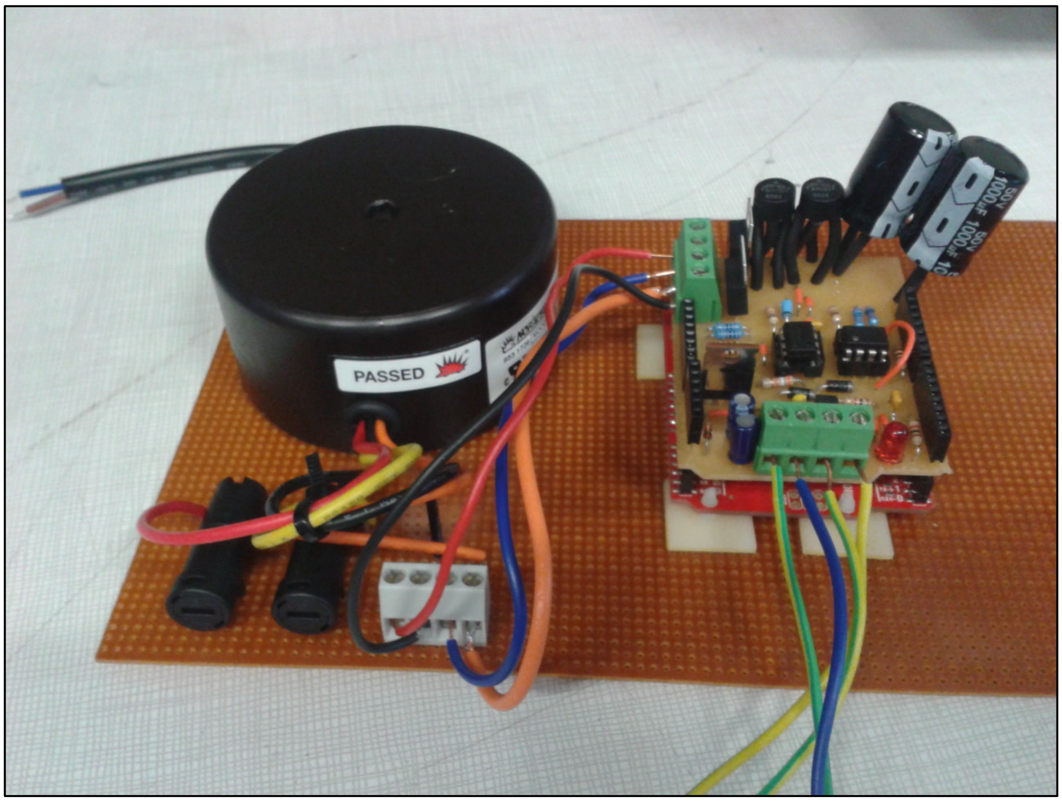
Finally with 1000uF/65V, as we can see in the picture above, we managed to get +12V and -12V from 220C (AC). It is very important to use capacitors of at least 1000uF/50V. Also we had made a remarkable mistake when connecting one

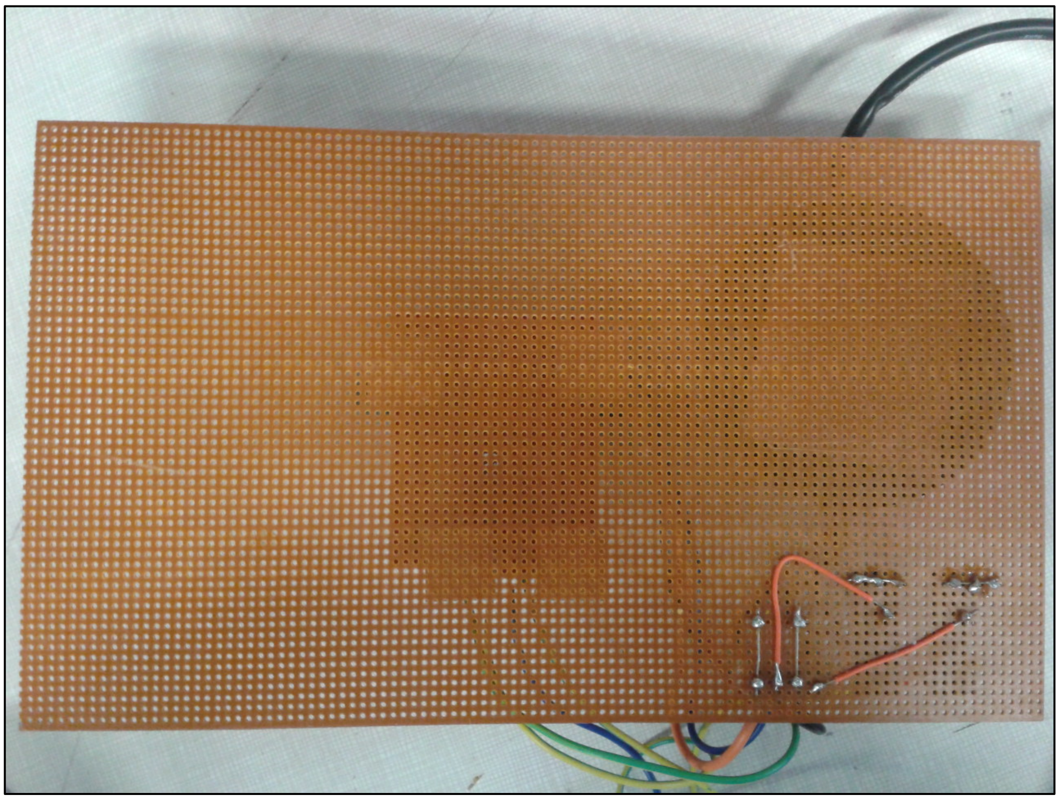
AC input of each bridge to the same 0 of the transformer. Basically, the rectifier bridges have their two AC inputs connected to 15V and 0. But this 15V and 0 are different for each bridge. As in the design of the PCB we had connected the two bridges to the same 0, we broke the connection between them and we added and additional cable to the free screw terminal of the PCB. When assembling the components to the PCB the size of the 1000uF/50V capacitors was too big to our design, so we placed one of them in two free holes of the board and making the appropriately connections with some extra wires. Moreover, the rectifier bridges that we used in the computer design where not the same as the ones we used, so we had to twist two of their legs so they were placed in the right position. Here we can see the top part of the PCB and the modifications made on the bottom part:



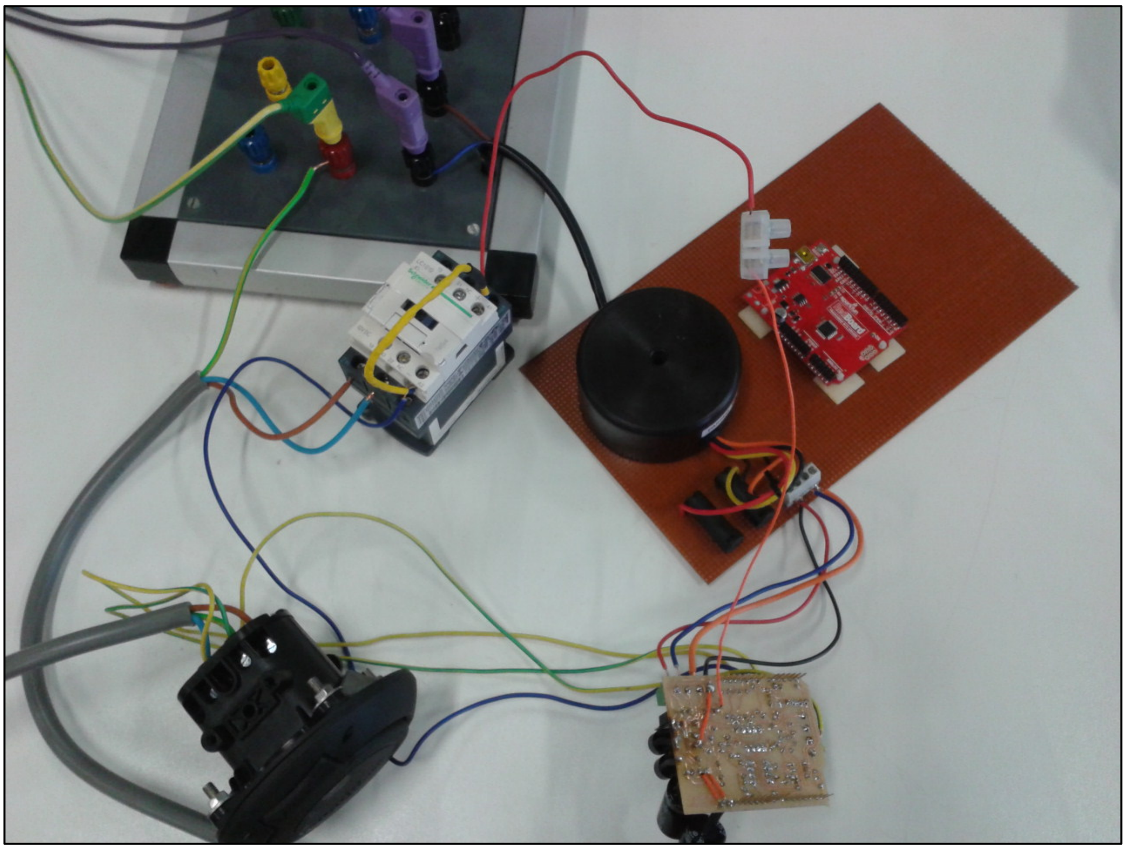


We fixed the transformer, the fuses, the Arduino board and the PCB in a solid structure as we can see in the next image:





We then connected the contactor, and we realised we had missed a terminal in the PCB to place the +12V wire of the contactor. Even though it would be convenient to make a new PCB with this additional terminal, we manually added a wire as we can see in the next picture:



When we were testing it, the negative circuit of the transformer to be able to get -12V was not working correctly. The fuse in this part of the circuit was broken probably due to a short circuit in the PCB. After verifying the tracks and connections in the PCB, the problem was not found.

3.4 Future improvements

A new version of the PCB with the last changes will be convenient. It should include:

- An additional output to connect the +12V of the contactor.
- The appropriately connections for the two "0" of the transformer.
- Bigger space for placing the two 1000uF/50V capacitors without problem.
- 4 extra terminals to connect the 4 outputs of the transformer on the PCB. We should make the board bigger to place the fuses (between the 15V and the 0 of the transformer) inside the board, so the

Probably after including these modifications the circuit will work perfectly but because of a lack of time we were not able to achieve the last step.

4. COST ANALYSIS OF THE SYSTEM

We have made a list of the prices of all the components we have used to allow the communication between the electric car and the charging station to have an idea about the manufacturing cost of the system:

COMPONENT	MANUFACTURER PART NO	UNITS	PRICE/UNIT (€)	TOTAL PRICE(€)
Arduino Uno board	A000046	1	23,27	23,27
Bakelite				max. 0,50€
8 Pin Headers		2	0,30	0,60
6 Pin Headers		2	0,30	0,60
Screw terminal block (2 Way)	MC000048	4	0,59	2,36
LED	LED5MMRJ	1	0,11	0,11
Op amp socket		2	0,40	0,80
LM358AP Op amp	LM358AP	2	0,392	0,784
5,1V Zener Diode	BZX85C5V1	1	0,243	0,243
4,7V Zener Diode	BZX79C4V7	1	0,103	0,103
1N4007 Diode	1N4007	3	0,188	0,564
BC547C Transistor	BC547CTA	1	0,017	0,017
IRF540 Mosfet	AUIRF540Z	1	1,69	1,69
100nF Capacitor	MCCC100V104KY5P	2	0,147	0,294
1uF Capacitor	EEUFC1H1R0	2	0,145	0,29
1000uF/50V Capacitor	ECA1HM102	2	0,89	1,78
1000pF Capacitor	D102K20Y5PH63J5R	1	0,085	0,085
100Ω Resistor	MCCFR0W4J0101A50	1	0,008	0,008
1kΩ Resistor	MCF 0.25W 1K	3	0,018	0,054
2kΩ Resistor	MF25 2K	3	0,042	0,126
3kΩ Resistor	MF25 3K	1	0,042	0,042
39kΩ Resistor	MCF 0.25W 39K	1	0,019	0,019
10kΩ Resistor	MCF 0.25W 10K	5	0,018	0,090
Transformer	MCFE030/15	1	28,26	28,26
Fuse holder	3101.0050	2	2,07	4,14
Fuse	70-065-63 2A	2	0,73	1,46
Bridge rectifier	AM151	2	0,48	0,96
7812 Voltage regulator	MC7812BTG	1	0,65	0,65
7912 Negative Voltage regulator	MC7912BTG	1	0,52	0,52
Contactor	LC1D12JD	1	40	40
Socket outlet		1	100	100
Wires				max. 1€

If we add the prices of all the components, the total cost will be approximately around 212€.

5. CONCLUSION

We will suggest as a future improvement to manufacture again the last design of the PCB and try to introduce on it the correct circuit to produce +12 and -12V without the need of the external power supply. Even though, we have been able to develop a communication system that will provide a safety and successful charging process with a very low price. The price of the system is approximately an 80% cheaper than the actual communication software in the charging process, so this means that definitely we have achieve our goal.

6. BIBLIOGRAPHY

Here are the different references of the information we have used:

- *IEC 62763*
- *P. Van den Bossche, EV Pilot Project State of the Art of Standardization for Electric Vehicle Charging Infrastructure January 2014*
- *e-mobility-nsr.eu*
- *www.smartev-vc.eu*
- *avt.inl.gov*
- *www.greenhoustontx.gov*
- *www.eurelectric.org*