IOT ROVER

Project report

April 2016

3rd Electronics

Jorge Crespo

Project supervisor: Eva Murphy

Content

1 Introduction and objectives

1.1 Summary of the project objectives

The objective of the project will be to develop and build an little ROVER car with sensors to interact with its environment and avoid collisions (Distance sensors), an GPS receiver to know every time it's exact position and an connexion to the internet through an mobile broadband connection (GPRS or 3G) to send data about its location and receive instructions from the user. We will use to read the data and send the instructions an webpage accessible from any computer with internet connection, and if possible we will try to develop an specific app for android.

1.2 Reasons to choose this project

This is an excellent opportunity to put in practise the knowledge acquire during the degree. This is also a very interesting project related with the current trend of connect everything to the internet (Know as Internet of things) During the development of the project we will have the opportunity of work with different modules and see how they work in a real environment. We will have to look for the GPS and mobile internet devices, see how to connect it to the PIC and develop a program to interface everything and make it work together all right. Also this project evolves sensors to measure distance to avoid collisions and will need a power stage to control the motors using the logic signals from the PIC, and even may be necessary to program and control stage using a PID to make the motors work all right. In summary, with this project I will put into practice a lot of the knowledge acquired during the years of university.

1.3 Applications

An autonomous and remote controlled car may be a very interesting development, it's useful for a very big range of things, and in a close future we will see autonomous vehicles of different sizes in our roads. Some of the possible applications are:

Rescue teams:

Have access to dangerous zones and localize people in disasters like earthquakes, and even give the help in first instance in case a human can get access to the victim.

Recognise dangerous zones:

In case of contamination nuclear, radiological, bacteriological or chemical it's very dangerous to send humans to develop their labours. A ROVER can do the job without take risk.

Help people with mobility problems:

For people who need help in their daily chores this kind of vehicle may be very useful

Big scale developments:

We could use autonomous vehicles as couriers, to supress the deliverer, make it safer and save money or even as public transport system, with the same advantages.

Even for this little development controlled by the internet we could think about different functions that our rover could do without many changes. You could leave it at home, install a camera on the robot and use it to have a look to your dog while you are working. Or imagine that you are at the pool and you are too lazy for going to the kitchen for a beer, well, you could send the robot and control it from your mobile.

2 Background

2.1 Actual developments

Currently the autonomous vehicles are real. There are autonomous vehicles like the development of Google (Google 2015), which is a fully autonomous car, or the development of Tesla (TeslaMotors 2015), which depends of a human driver but automatics some of the functions of the driving. Also small vehicles more similar to our development are currently used by armies of some countries and for police departments for dangerous operations as deactivation of explosives, but there are not any solution in the market for your own use for a competitive price.

Another developments at big scale have been done by some classic car manufactures, like Volkswagen, Mercedes-Benz or General Motors, there are also importants the developments of Google and Tesla which we have mention and also in the industry of the trucks there are quite interesting developments made by Daimler, Scania and Volvo, these trucks have been tested with great results. We can be sure about that in a few years we will see a final product in the market.

2.2 Comparison

This project will try to be an approach to what needs an autonomous vehicle to work allright and what need an real time connection throught the internet to control a vehicle. We will try to make it work as better as possible while we keep the design as low cost as possible, an also we would like to make it open source, sharing all the code (GPLv3) (gnu n.d.) and all the documentation (CC by-nc-sa)(Creativecommons.org n.d.) in order to allow anyone to replicate the project, or even that other people continue and improve the project.

3 Research

3.1 Concept selection

3.1.1 Original concept

In the beginning was quite difficult to decide which project to develop. The only condition given was to use a PIC, the PIC18F45k20 (Microchip 2010) to control the project. Also, personally, I wanted to develop something related with the internet of things, so I wanted a module to have internet connection through a mobile connection or using Wi-Fi. With these two conditions in mind I develop two different proposals.

3.1.2 Selected concept

One of the options was the autonomous car controlled over internet and the other one was a system to automate different aspects of a home, like control of the temperature, lights, irrigation…

The Rover was selected because to work with GPS is very interesting and also because the control system is more interesting involving more factors.

3.2 Components review

3.2.1 Motor controllers

Our design will have two motors, one for each track. Our traction will be differential and we also want to be able to go to the front and to the back. So we need a system that allows us to control the velocity and the direction of the motors. For this purpose, the best approach is the utilization of an H bridge. Between the differents options in the market for this kind of integrated device, we have choose a L298N (STMicroelectronics 2000). It is a dual H bridge embedded in a chip which is very used in field of Do It Yourself robotics. It have been tested by a lot of people, it's cheap and accomplish with our requirements. The only problem with this chis is that it doesn't have embedded flyback diodes, so we will have to add these diodes externally. For this work, 8 general purpose low cost 1N4007 diodes (Vishay 2002) will be great.

Illustration 2. L298 motor controller

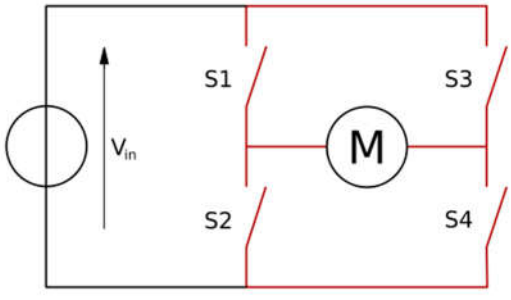


Illustration 1. H Bridge. (BUTTAY 2006)

3.2.2 Motors

As we are going to use H bridge, and assuming that our robot doesn't have special requirements of speed, torque or consumption, we are going to use the starter DC brushed motors which are included in the chassis that we will use.

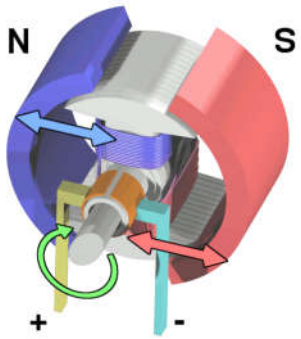


Illustration 3. DC Motor. (Wapcaplet 2006)

3.2.3 LCD modules

For the LCD module, has been chosen a standard display with 2 lines of 16 characters. It has to be based on the Hitachi HD 44780 (Hitachi 1998) as there is a lot of information about this chip and there are a lot of open free libraries to interactuate with screens based on this this chip since different microcontrollers (Including our PIC).

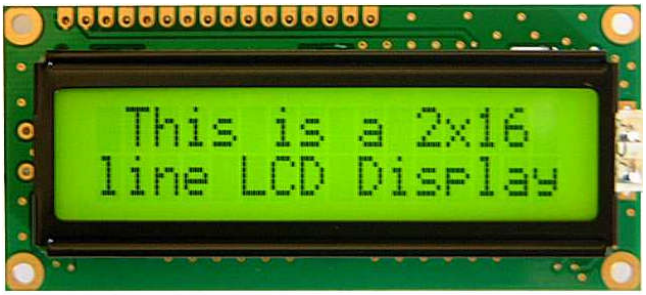


Illustration 4. LCD 16x2 Display. (Electrosome n.d.)

The major ploblem of these modules is that the major part of then are designed to operate at a logic voltage of 5 volts, but our microcontroller operates at 3.3 volts, so will have to use a LCD module rated for 3.3 volts, which is more difficult of find and also is more expensive.

3.2.4 Distance sensors

For this purpose we have evaluate two main options. Sensors based in ultrasounds and sensors based in infrared. For each technology there has been reviewed one sensor, chosen based on the quantity of documentation about it and availability. The best options are:

- Infrared sensor Sharp GP2Y0A21YK (Sharp n.d.), it cost around 12€ and give us an analogue output of 3.1V at 10 cm to 0.4V at 80 cm

Illustration 5. Sharp GP2Y0A21YK. (Sharp n.d.)

- Ultrasonic sensor HC-SR04. It's very cheap, around 1.50€ the unit. The function is a little more difficult, we should send a trigger signal to start the measuring, and then the module will answer us with an echo signal with the same duration that has wait the module to receive the ultrasonic back, so then we have to calculate based on the sound velocity, how much meters are between the unit and the obstacle.

Illustration 6. Hc-sr04 Distance Sensor.(Labyteacora n.d.)

Two options would have work all right in our design, and after review both options there was chosen the infrared sensor, because the output is analogy, and it gives us an interesting opportunity to work with the embedded Analog to Digital Converter

3.2.5 Power sources

In this project we have seen that there will be many different components working at different voltages and requesting different currents. Which the rover will go in its own, we will need a battery. There are different kinds of technology that we can use in our battery: Simple alkaline cells, which are rejected because is more interesting to be able to recharge the batteries. In rechargeable batteries we can choose Lead-Acid, Ni-MH or Li-Po batteries, between other options. There has been choose a Li-Po battery because the its most recent technology, offers more energy per unit of volume and currently the tendency in the world of the modalism is the use of Li-Po batteries, so we can find more information and there are more options available.

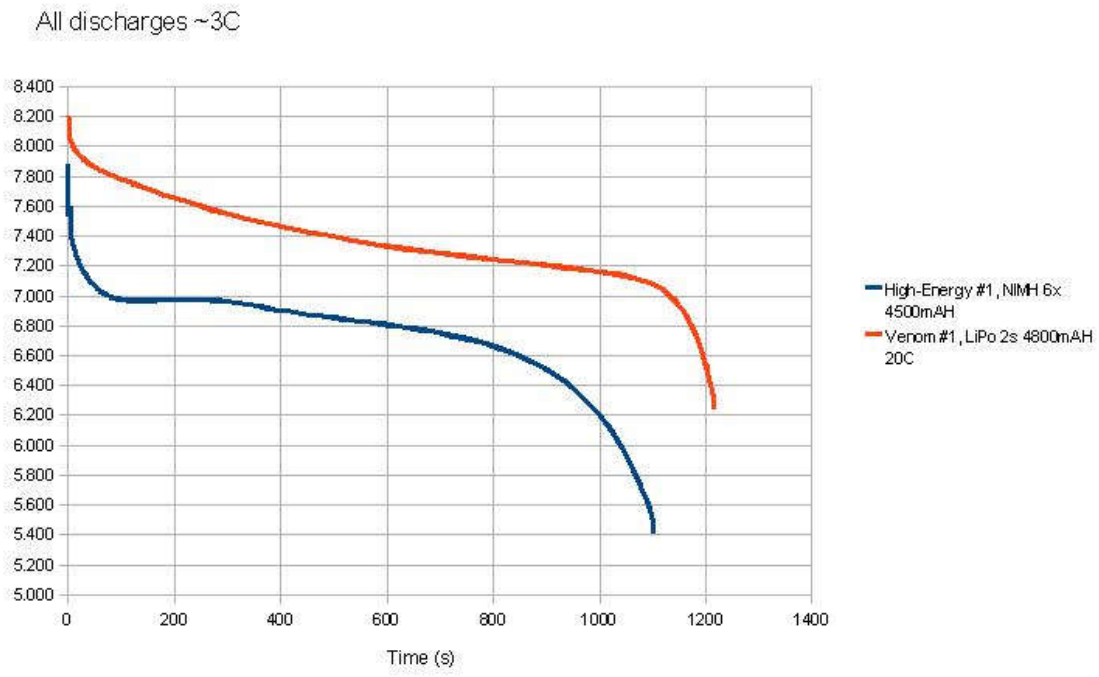


Illustration 7. Discharge curves for LiPo and NiMH batteries.(RCUniverse n.d.)

3.2.6 DC/DC convertors

Furthermore, in our project there are components working at 3.3 volts and at 5 volts. To generate this voltage there has been assess the famous series of Texas Instruments LM7805 (Texas Instruments 1995), with the variation UA78M33 (Texas Instruments 2015) for 3.3 volts, or the use of the commuted PWM DC/DC converter also from Texas Instruments LM2596 (Texas Instruments 2013). This last option has a better performance, which is very important in an embedded device, but was chosen the option of the linear regulators based on the LM78XX because needs less components and simplifies our work. But for a production device there should be seriously considerated the use of a commuted DC/DC converter.

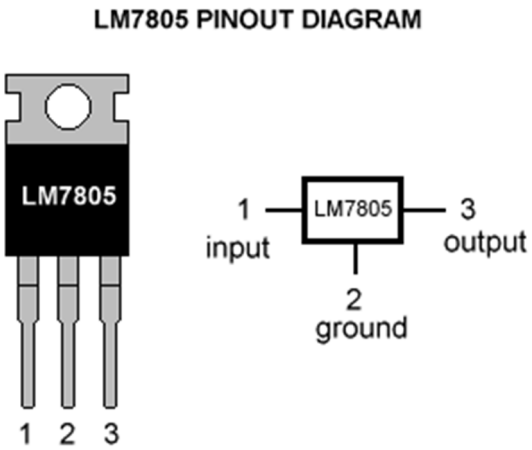


Illustration 8. (Techinc n.d.)

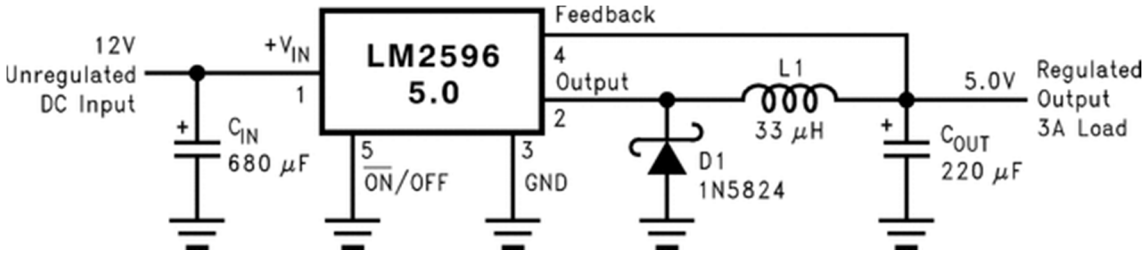


Illustration 9. LM2596 Typical operation. As we see, it needs some aditional components.(Texas Instruments n.d.)

3.2.7 GPS+GPRS modules

The project will need connectivity to the Internet and GPS connection. For this purpose we will use a module with connectivity throught the serial port of the PIC (Microchip 2010), an will consist of a GPS module and a GPRS module. For the GPS and GPRS we have a lot of different options, but any of them has specific documentation for work with an PIC microcontroller. In any case, all the solutions I have found have specific libraries for Arduino and Raspberry and only needs 4 wires (RX, TX, VCC and GND) so we have two options, find a way to make it work with our PIC, writing or adapting the libraries for Arduino, or to connect an Arduino to the PIC and make them speak together. The best options that I have found are:

- Adafruit Fona 808 (Adafruit n.d.): It's a board based on the SIM808 integrated device. It includes GPRS and GPS connectivity, and also a wide variety of libraries and examples. It's the cheapest option (44€) and accomplish with our requirements. Adafruit has a service for Internet Of Things applications, where you can program inputs and outputs and control it from any internet brownser.

Illustration 10. Fona 808 (Adafruit n.d.)

- Cooking hacks SIM908 (Hacks n.d.): Very similar to the previous development but with a modern chipset. We also have a lot of documentation about projects with this device. It cost 99€

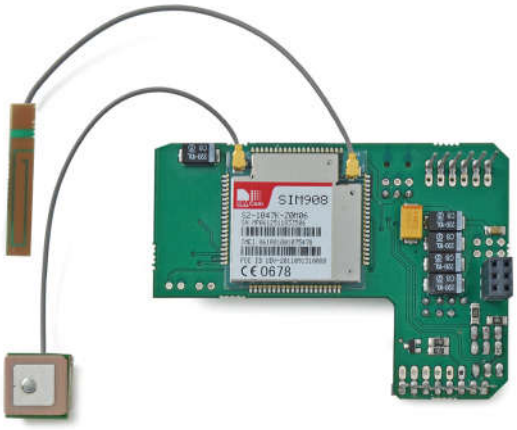


Illustration 11. SIM908

- SparqEE CELLv1.0 (SparqEE n.d.) + Independent GPS module. This module doesn't includes GPS, so we would have to buy it apart, the main advantage is that with the module you receive the right to use the servers and the IDE of the company to develop your project. The module cost 95€ and a GPS module around 35€

Illustration 12. SparqEE Cell

Illustration 13. GPS module

3.2.8 Arduino

Arduino (Arduino 2016) is an open software and open hardware development board based on a Atmel microcontroller. An Arduino mega has been included in our project because has not be possible to stablish connection between the GPRS and GPS board and the PIC microcontroller using the embedded serial communication port. In the case of Arduino there are Open Source libraries and examples to make it work, so the implementation has been easy. This unexpected problem and the solution using an Arduino carry another problem, the communication between the PIC and the Arduino. It

has been solved developing the software to make a parallel communication using 8 ports of the Arduino as outputs and 8 ports of the PIC as imputs.

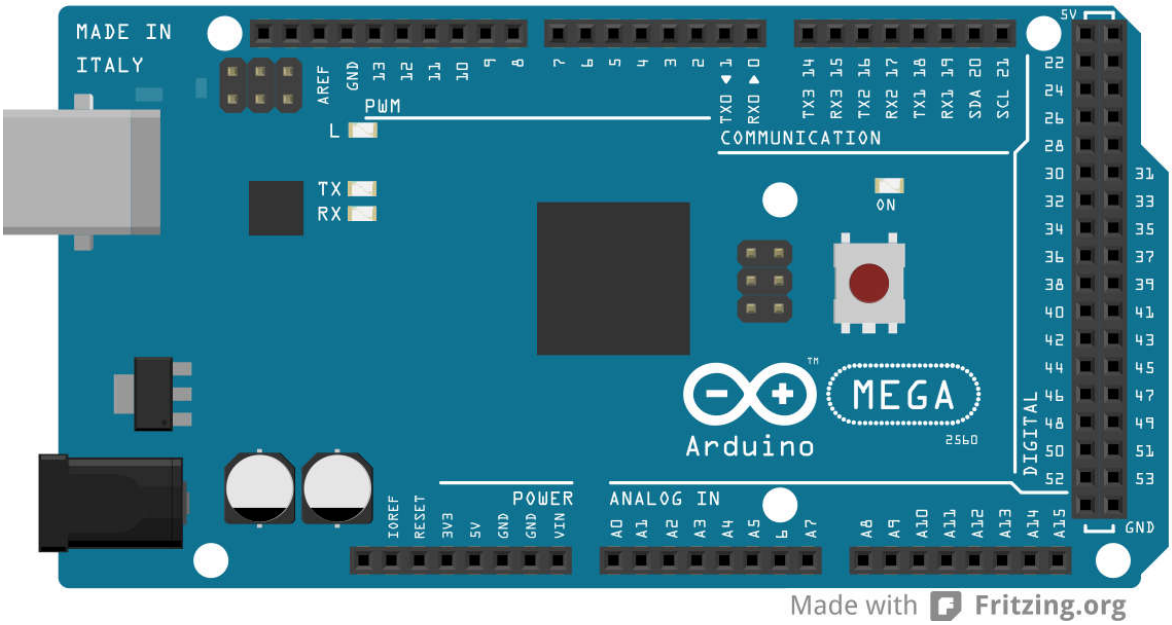


Illustration 14. Arduino Mega (Arduino 2016)

3.2.9 Chassis

For the chassis has been evaluated the option of the 3D printing or to buy a commercial chassis. At the end, due to the lack of time and the fact that here I don't have full access to a 3D printer, has been chosen to buy a commercial development.

Between the different options, has been chose the Rover 5 Robot Chassis, because the IT have made projects with it and the Access was easy.

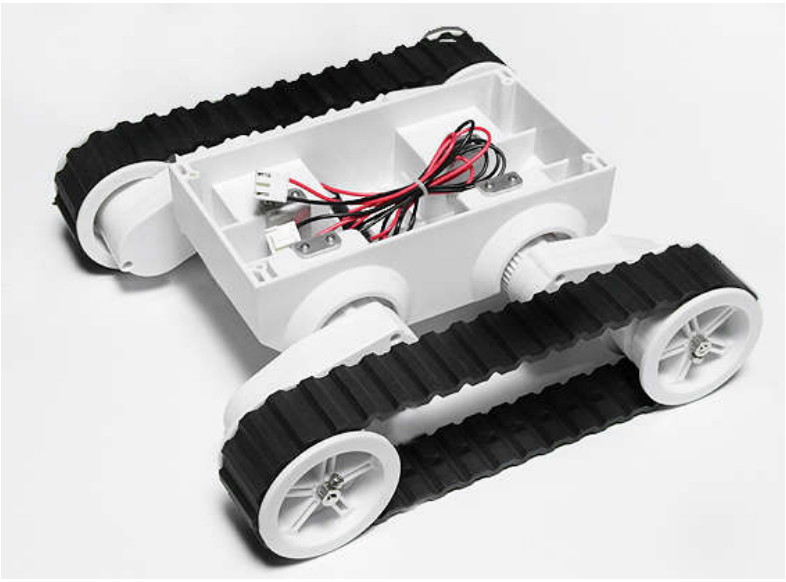


Illustration 15. Rover 5 chassis

4 Ethical considerations

4.1Environment impact

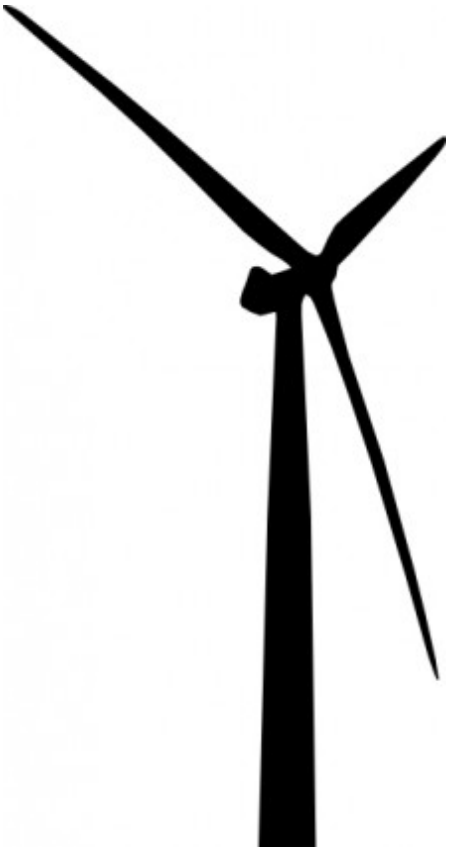


Illustration 16. Wind generator

It will make use of electric energy with a build-in battery, which is more eco-friendly that use gas, like most of the actuals vehicles.

4.2Society impact

Autonomous vehicles will have a very big impact in our society. They will substitute the drivers of any vehicle at the same time that will be created a big industry for design manufacture and maintain these vehicles. The cost of the transports will be lower, making products cheaper, travels cheaper, and allowing very low cost of transport, which will make cheaper alternatives to traditional shops like e-commerce. Also make it open source would make this development more accessible and will allow other people continue the development in the best conditions, being free if the purpose is not commercial, for example for NGO or for academic works.

Illustration 17. Creative Commons logo (Creativecommons.org n.d.)

4.3 Health and safety

Of course, the design should accomplish with all the normative of the European Union referring to electrical devices and to vehicles, but we are going to think bigger. In the future with autonomous vehicles we could reduce collisions and death people by car accidents to a very small number. Human mistakes will be a think of the past and we will only care about mechanical, climatological and other accidents.

4.4 Legislations

This kind of autonomous vehicle will not have the right to driving on the road. The legislation should change and the vehicle should demonstrate that it is 100% safe.

All the pieces of Our development are ROHS compliant, but the tin that has been used to solder manually a few cables and components has a percentage of plumb, which makes the global project not ROHS compliant. With standard equipment in impossible to solder without lead, we would have need special solder stations of high temperature and also the process is more complicated.

Also, for this kind of product we should be sure about that the product comply with the CE normative for Europe and the FCC normative for the United States. This is a process which requires a deep knowledge of the directives in order to determinate if the project accomplish or not.

Illustration 19. ROHS logo

Illustration 18. CE Logo

5 Hardware development

5.1 Hardware diagram:

In the next diagram we can see a basic approach to the structure of the modules of our project. The power connections from the voltage regulators to the components have not been drawn in order to keep the diagram more readable.

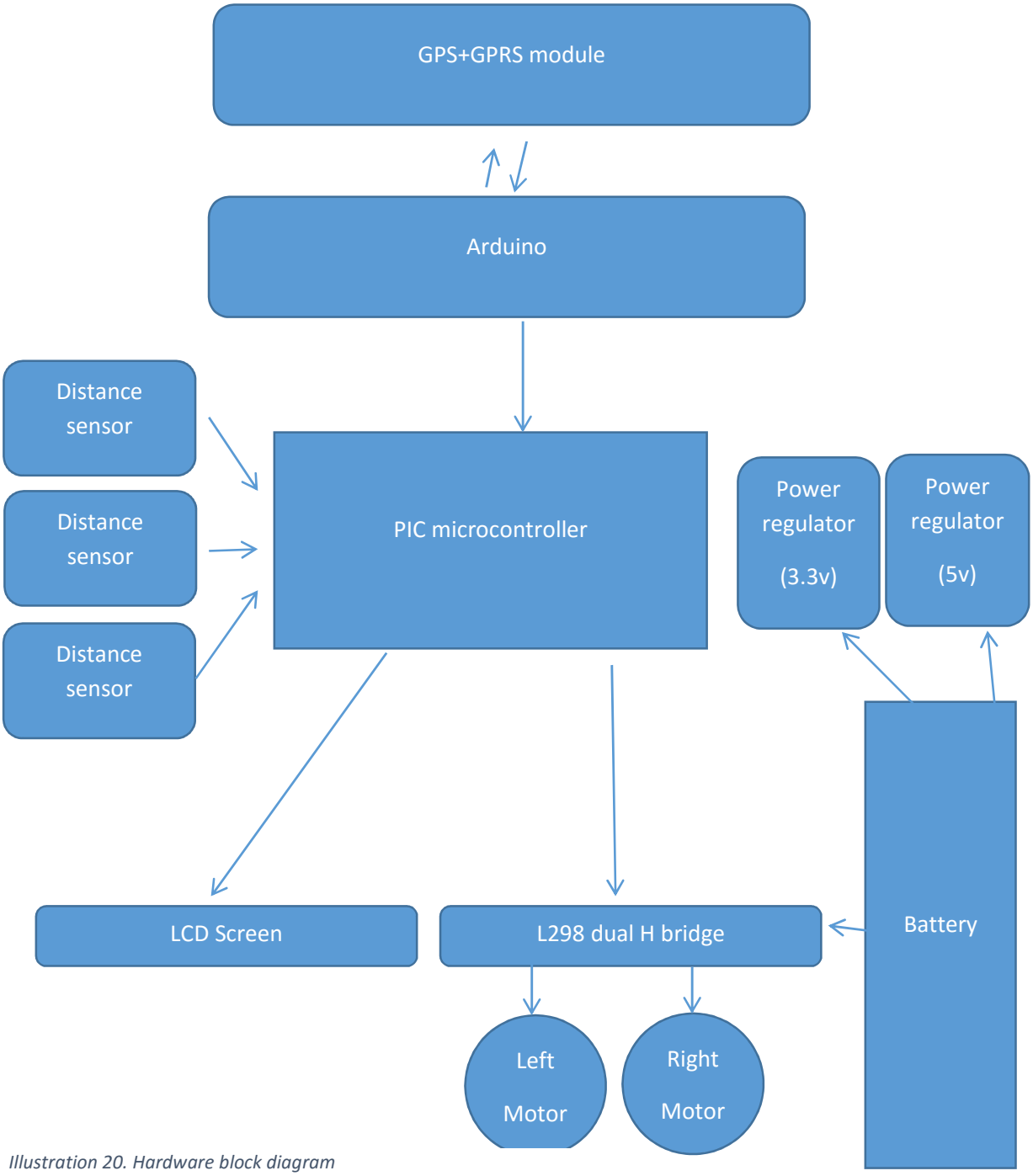
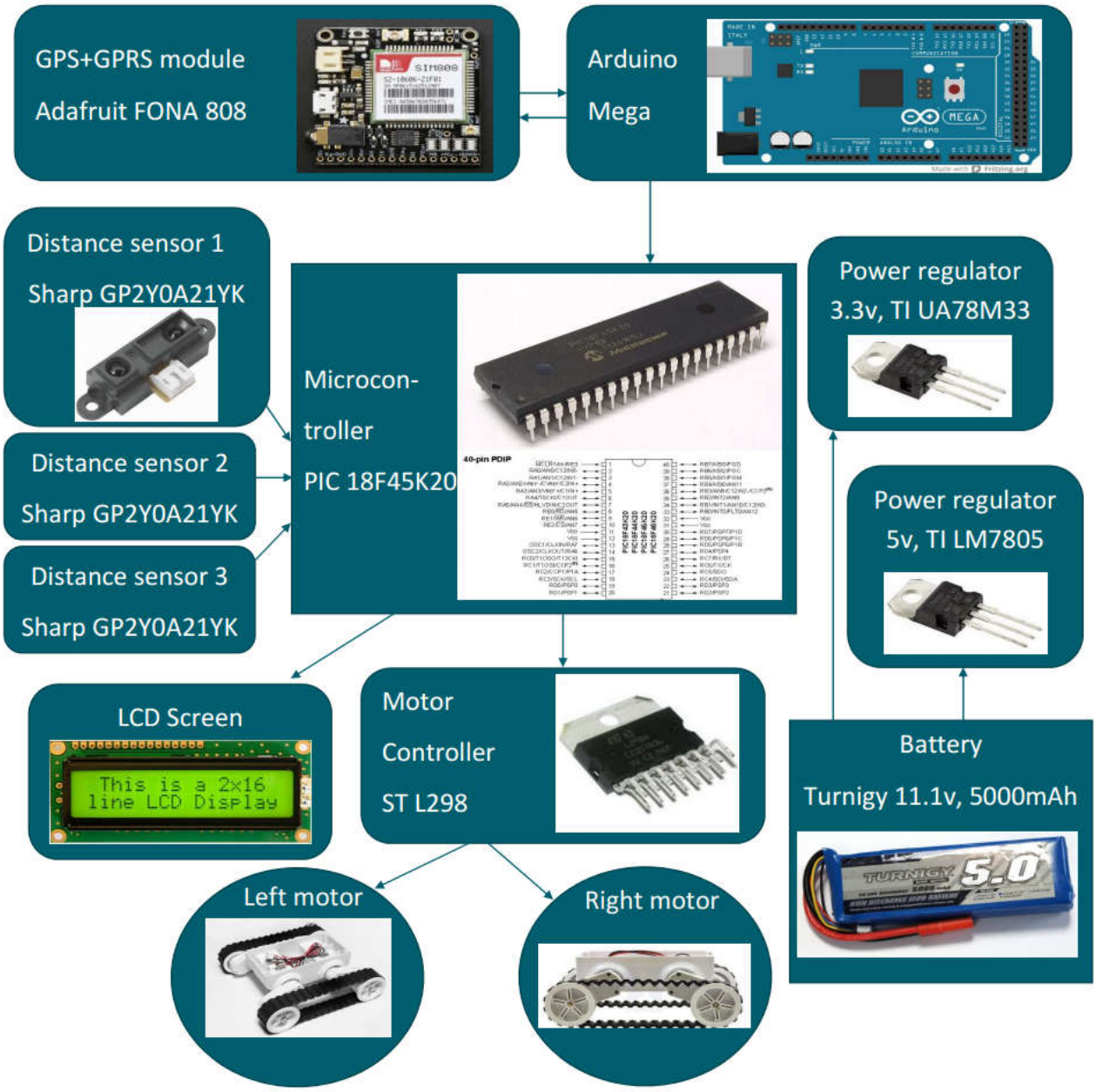


Illustration 20. Hardware block diagram

5.2 Diagram with photos:



5.3 Circuit design (Proteus)

A design in Proteus has been made in order to simulate all the possible components and test it before connect the real components. The Arduino and the GPS+GPRS module has not simulated, and the imput from the Arduino to the PIC has been made using one switch for each input, at the bottom left. Also we can see the power regulators in the top, the three distance sensors in the left, the motor controller plus the motors in the right and the lcd screen in the bottom. The PIC microcontroller is in the center.

The simulation of the device is a very helpful tool that allows us to test everything without danger for the components, and also the embedded tool in proteus for debugging allows us to find errors in the code in an easy and secure form.

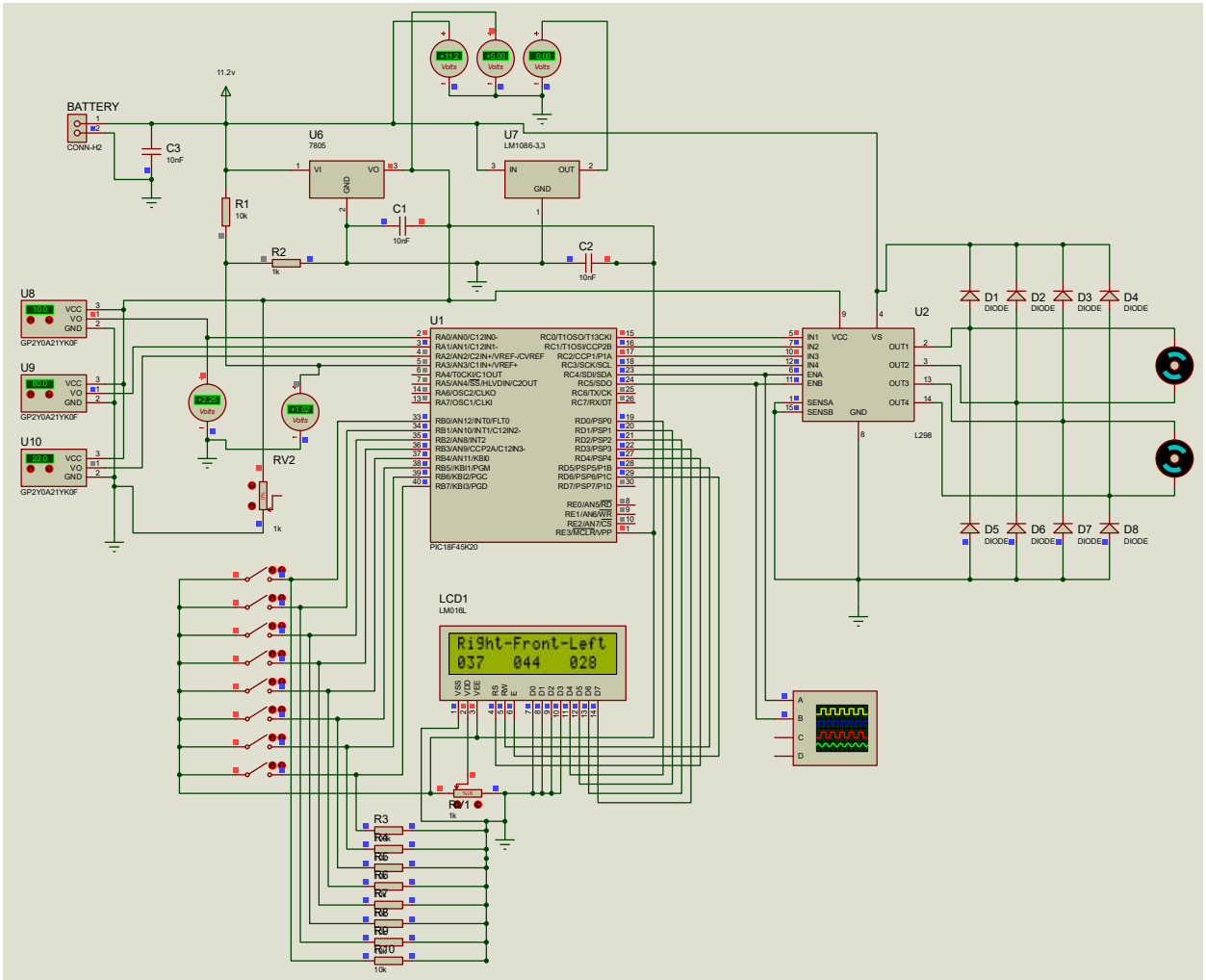


Illustration 21. Proteus design.

5.4 Mounting hardware

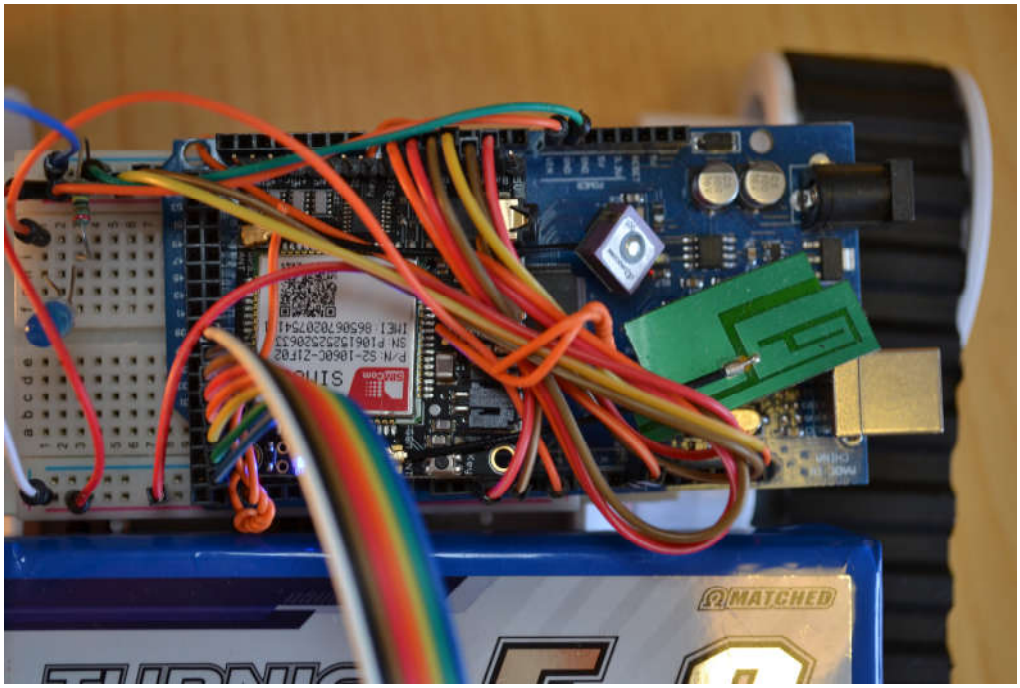


Illustration 22. Arduino and Fona module

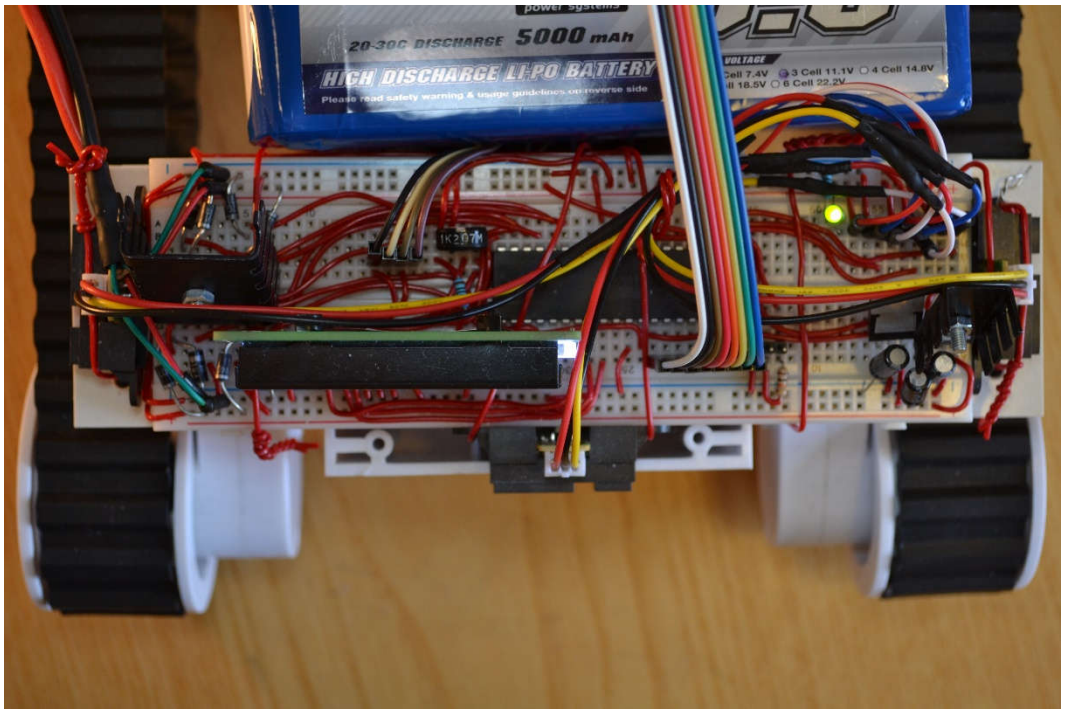


Illustration 23. Protoboar with PIC detail

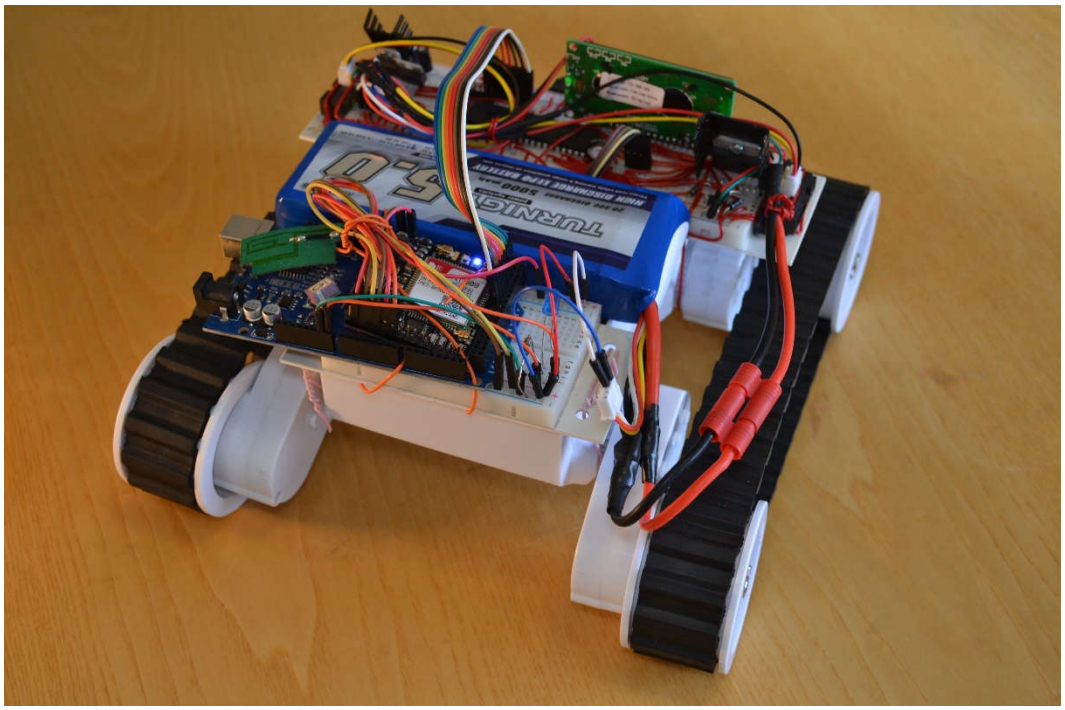


Illustration 24. General view (Back)

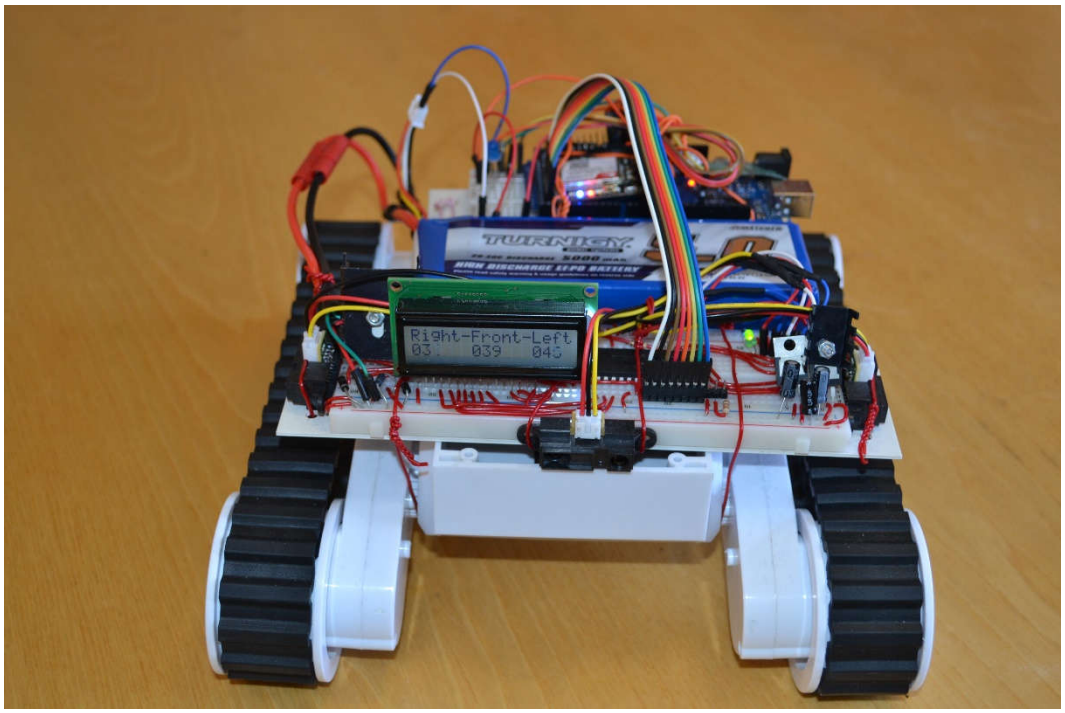


Illustration 25. General View (Front)

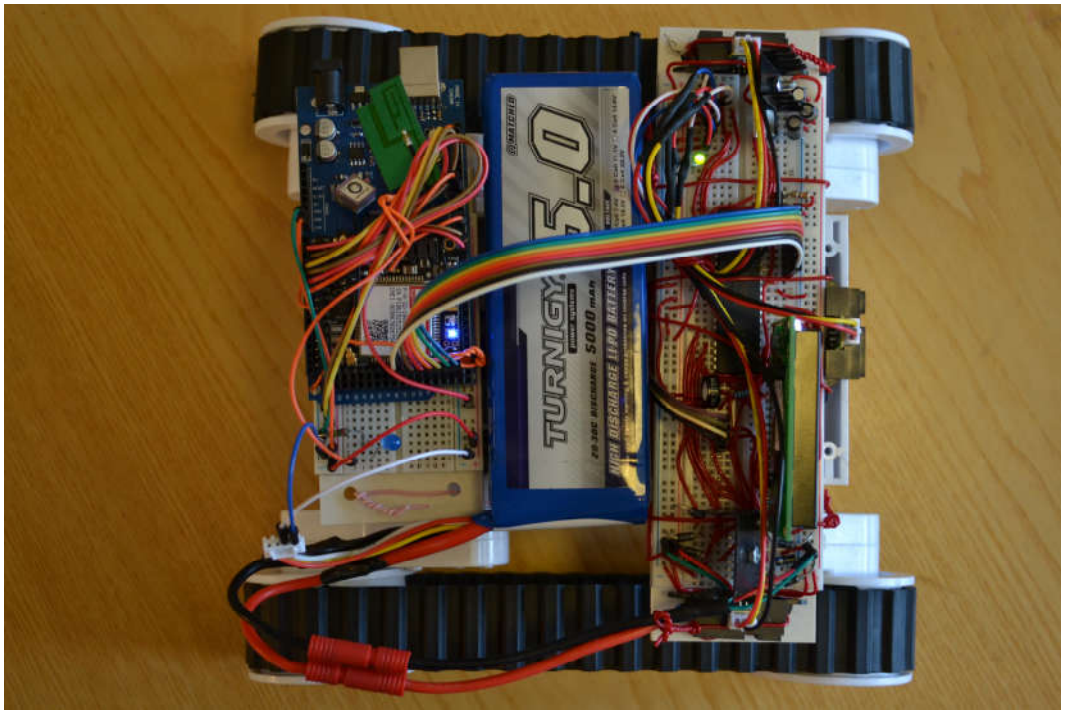


Illustration 26. General View (Up)

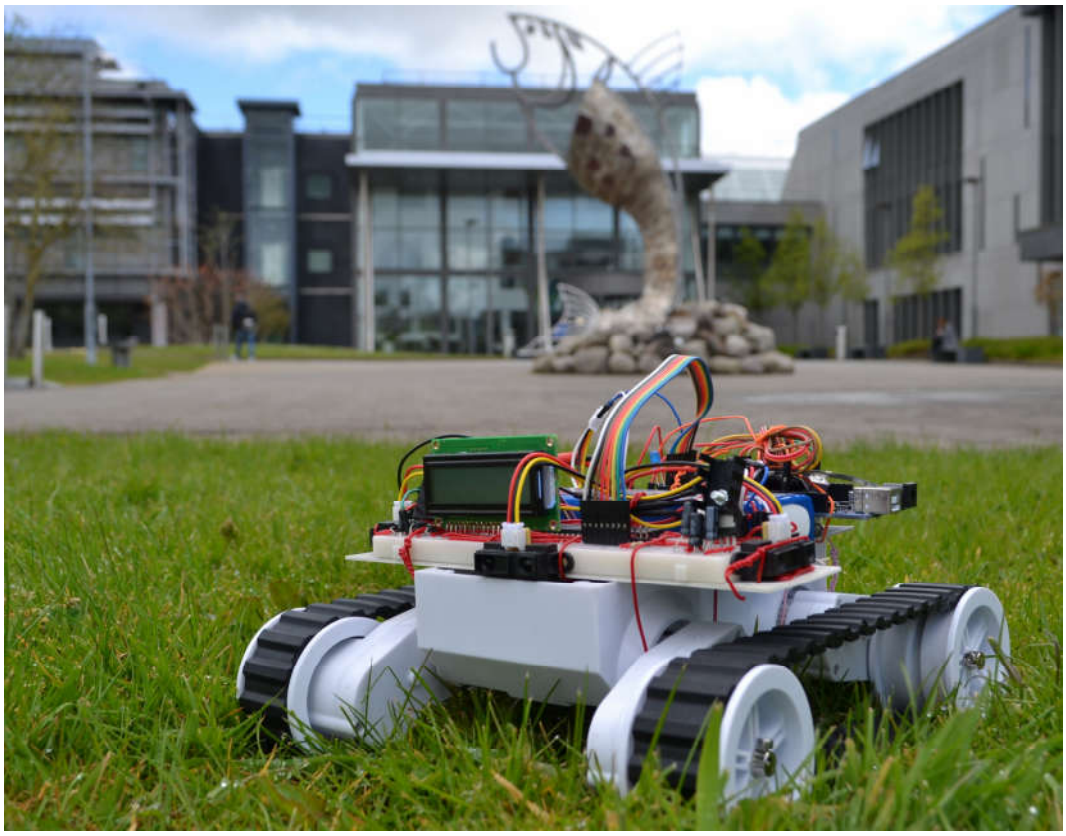


Illustration 27. Robot in action

6 Software development

The development of the software to make everything work in perfect harmony has been the most intense task of the project.

Ther is a main program writed for the Arduino, and also a there has been writing a little programme for the Arduino, to interact with the GPS+GPRS board and to send data to the PIC.

6.1 PIC

6.1.1 IDE (MPLab-X)

6.1.2 Programming the PIC

6.1.3 Libraries

One external library developed by a third (Extremeelectronics 2013) has been used to interact with the LCD library. This library is open source and free to use for non-commercial applications, so it's perfect for our project and help us to save a lot of time writing drivers for the LCD screen. As this development is done by a third, the code will not be commented in this report.

6.1.4 Code

To make the software more legible and also increment the portability or changes, the program has been developed in different modules to interact with each piece of hardware and realize specific actions. We will see one by one all this modules and all the functions of each one. There has been included only the ".c" files in this report, as the ".h" files only includes the declaration of the functions. Also all the code is commented, it's easy to follow without extra information

6.1.4.1 Main.c

This is our main module, it starts all the different modules and starts a loop which just call a function to make an ADC and the go function, which is the main control function.

```
#include <xc.h>
#include "ADC.h"
#include "motors.h"
#include "lcd_hd44780_pic16.h"
#include "TimerPWM.h"
#include "driver.h"
#include "parallel.h"

void main(void) {
```

```
    ADCInit();//Start Analog to digital converter
    MotorInit();//Start the motor controller
    TimerPWMInit();//Start the generation of PWM signal using
timers
    LCDInit(LS_NONE); //Initialize the LCD Module
    parallelInit();//Initialize the reception from the Arduino over
the parallel port
    SetMotorLeft(0);//Start motor left at speed 0
    SetMotorRight(0);//Start motor right at speed 0

    /* Endless loop*/
    while(1) {
        ADCInitiateConversion();//One analog read.
        Go();//Take decissions to avoid obstacles. And send orders
to obstacles.
    }
}
```

6.1.4.2 Driver.c

This module takes the decisions to move the rover based on the data received from the Arduino through the serial port. There are also a few functions to perform basic moves.

```
#include <xc.h>
#include "motors.h"
#include "lcd_hd44780_pic16.h"
#include "ADC.h"
#include "driver.h"
#include "parallel.h"
#include <stdlib.h>
#define _XTAL_FREQ 1000000

/*This is our basic function to move the ROVER.
 * It is call one time per program cycle
 * has two modes, one for manual control
 * and another is the automatic mode.
 * The mode is selected from the webpage
 * and received from the Arduino
 * */
void Go(void) {
    ReadParallel();/*One lecture of the parallel port.
                    * This sets the mode and speed of both motors*/


    if(ReadMode()==0){//Manual mode
        signed int SpeedLeft, SpeedRight;
        SpeedLeft=ReadParallelSpeedLeft();
        SpeedRight=ReadParallelSpeedRight();

        if(SpeedRight>0  &&  SpeedLeft>0){//If  we  go  straight,
considerate distance sensors
            if          ((ReadDistanceCentral()<37)          ||
(ReadDistanceRight()<30) || (ReadDistanceLeft()<30)){//Danger!!
                SetMotorLeft(0);
                SetMotorRight(0);
            }else{//If there are not obstacles, listen the orders
```

```
                SetMotorLeft (SpeedLeft);
                SetMotorRight(SpeedRight);
            }
        }else{//If we are not going straight, ignore sensors
            SetMotorLeft (SpeedLeft);
            SetMotorRight(SpeedRight);
        }

    }else if (ReadMode()==1){//Automatic mode

        if (ReadDistanceCentral()<37){//Obstacles in the front
            if(ReadDistanceRight()>ReadDistanceLeft()){
                TurnRight135();
            }else{
                TurnLeft135();
            }
        }else if (ReadDistanceRight()<30){//Obstacles in the right
            TurnLeft90();
        }else if (ReadDistanceLeft()<30){//Obstacles in the left
            TurnRight90();
        }else{//No obstacles!!
            SetMotorRight(40);
            SetMotorLeft(40);
        }
    }
}

/*The next functions make some basic funcions in the robot
 *turn a different number of degrees to the right or the left
 * and go a little bit back
 */
void TurnRight90(void){
    SetMotorRight(-100);
    SetMotorLeft(100);
    __delay_ms(100);
}

void TurnLeft90(void){
    SetMotorRight(100);
    SetMotorLeft(-100);
    __delay_ms(100);
}

void TurnRight135(void){
    SetMotorRight(-100);
    SetMotorLeft(100);
    __delay_ms(150);
}

void TurnLeft135(void){
    SetMotorRight(100);
    SetMotorLeft(-100);
    __delay_ms(150);
}

void Turn180(void){
    SetMotorRight(100);
    SetMotorLeft(-100);
    __delay_ms(200);

}
```

```
void GoBack(){
    SetMotorRight(-50);
    SetMotorLeft(-50);
    __delay_ms(500);
}
```

6.1.4.3 Motors.c

This is the module that receives a number from -100 to +100 and writes the pins to set the direction of the motors and also send the data to the modulo to generate PWM signals and control the speed.

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include "TimerPWM.h"
#include "motors.h"

/*Pins for the ports to the motor controller*/
#define ML1 RC0
#define ML2 RC1
#define MR1 RC2
#define MR2 RC3

static signed int MotorRightSpeed=0;
static signed int MotorLeftSpeed=0;

/*
 * Start the port used to control the motors
 * Also set the motors to be stopped
 */
void MotorInit(){
    TRISC = 0b11000000; // Motors are on PORTC and output (Minus
the Tx and Rx)
    ML1=1;
    ML2=0;
    MR1=1;
    MR2=0;
    SetPWMMotorLeft(0);
    SetPWMMotorRight(0);
}

/*Function to set the speed of the left motor
 * ML1 and ML2 control the direction, and later we set the PWM of
that motor to set the speed
 * the function works with a signed integer from -100 to +100
 */
void SetMotorLeft(signed int speed){

    MotorLeftSpeed=speed;
    //Set direction
    if(speed>0){
        //Go forward
        ML1=1;
        ML2=0;
```

```
    }else if (speed<0){
        //Go Backward
        ML1=0;
        ML2=1;
    }

    //Set speed
    SetPWMMotorLeft(abs(speed));//We need two independent PWM
}

/*Same thing for the other motor
 */
void SetMotorRight(signed int speed){

    MotorRightSpeed=speed;
    //Set direction
    if(speed>0){
        //Go forward
        MR1=1;
        MR2=0;
    }else if (speed<0){
        //Go Backward
        MR1=0;
        MR2=1;
    }

    //Set speed
    SetPWMMotorRight(abs(speed));//We need two independent PWM
}
```

6.1.4.4 TimerPWM.c

Module for generate the two necessary PWM signals to control the speed of the motors. It has been done using a timer because this microcontroller only has one hardware PWM generator, and we need two independent signals, one for each wheel.

```
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>

/*Variables to keep saved the PWM values
 */
static unsigned int PWMRight;
static unsigned int PWMLeft;

/*This functions starts the timer and the timer interruptions
 *
 */
void TimerPWMInit(){
    GIE=1;//Enable global interruptions
    TMR0IE=1;//Timer0 Interruption
    T08BIT=1;//8 bit mode
    T0CS=0;//Internal transition
    PSA=1;//Timer 0 with no preescaller (1:1)
    T0PS2=0;//Preescaller, dont care
    T0PS1=0;//
```

```
    T0PS0=1;//

    TMR0IF=0; //Overflow flag
    TMR0H=0x00;//Timer start at 156 to 256 (1ms)
    TMR0L=156;
    TMR0ON=1;//Start timer
}

/*Sets the speed of the motor.
 * It receives a value from 0 to 100 from the motor.c module
 * and converts it to values from 20 to 0, which are used by
 * the interruption
 */
void SetPWMMotorLeft (unsigned int percentage){
    PWMLeft=20-percentage/5;
}
void SetPWMMotorRight (unsigned int percentage){
    PWMRight=20-percentage/5;
}

/*Returns the actual value of the PWM variables
 */
unsigned int ReadPWMMotorRight (){
    return PWMRight;
}
unsigned int ReadPWMMotorLeft (){
    return PWMLeft;
}
```

6.1.4.5 ADC.c

Modulo to read the analogs inputs and save the last value to a variable, which can be read calling from any module to a function. It makes use of a interruption to read the input and also displays the value of the last distance lectures in the LCD.

```
#include <xc.h>
#include "lcd_hd44780_pic16.h"
#include "ADC.h"
/*Constants for keep in memory the last measures from the ADC*/
static unsigned int DistanceLeft=50;
static unsigned int DistanceRight=50;
static unsigned int DistanceCentral=50;
static unsigned int Battery;

void ADCInit(void) {

    /* Configure the ADC - input to ADC is AN0 (PORTA.0)*/
    TRISA0 = 1; // configure PORTA.0 as an input
    TRISA1 = 1; // configure PORTA.1 as an input
    TRISA2 = 1; // configure PORTA.2 as an input
    TRISA3 = 1; // configure PORTA.3 as an input

    ANS0 = 1; // disable PORTA.0's digital input
    ANS1 = 1; // disable PORTA.1's digital input
    ANS2 = 1; // disable PORTA.2's digital input
    ANS3 = 1; // disable PORTA.3's digital input

    ADON = 1; // enable ADC
```

```
    ADCON1 = 0; // reference voltages are internal
    ADCON2 = 0x91; // right justified, 4 TAD, FOSC/8
    ADIF = 0; // clear the ADC conversion complete interrupt flag
    ADIE = 1; // enable ADC interrupt

    PEIE = 1; // set the global peripheral interrupt enable bit
    GIE = 1; // set the global interrupt enable bit

}

void ADCInitiateConversion(void) {
    GO = 1;//Initiate one read from the ADC
}

//Returns the 10 bits read from the ADC
unsigned int ADCGetOutput(void) {
    int temp;
    temp = ADRESL;
    return temp + (ADRESH << 8);
}

//Returns the 8 low bits read from the ADC
unsigned char ADCGetOutputLowByte() {
    return ADRESL;
}

//Returns the 2 hight bits read from the ADC
unsigned char ADCGetOutputHighByte() {
    return ADRESH;
}

//Updates the variable. Will be call from the ADC interrupt.
void SetDistanceLeft (unsigned int lecture){
    DistanceLeft=5+(1023-lecture)*0.042;//Not exact, but it gives
us an idea.
}

//Returns the constant. Will be called when calculating route to
avoid collisions
unsigned int ReadDistanceLeft (){
    return DistanceLeft;
}

void SetDistanceRight (unsigned int lecture){
    ShowLecturesInLCD ();
    DistanceRight=5+(1023-lecture)*0.042;
}

unsigned int ReadDistanceRight (){
    return DistanceRight;
}

void SetDistanceCentral (unsigned int lecture){
    DistanceCentral=5+(1023-lecture)*0.042;
}

unsigned int ReadDistanceCentral (){
    return DistanceCentral;
}

void SetBattery (unsigned int lecture){
```

```
    //Measure bridge with 997+9800 ohm resistors
    Battery=((lecture*3.548)-11)*100;//Hight accuracy!.
}

unsigned int ReadBattery (){
    return Battery;
}

//Function to write the distances in the LCD. Called one time per
cicle of lectures
void ShowLecturesInLCD (void){
//Show values in LCD
    LCDGotoXY(0,0);
    LCDWriteString("Right-Front-Left");

    LCDGotoXY(0,1);
    LCDWriteInt(DistanceRight, 3);

    LCDGotoXY(6,1);
    LCDWriteInt(DistanceCentral, 3);

    LCDGotoXY(12,1);
    LCDWriteInt(DistanceLeft,3);
}
```

6.1.4.6 Parallel.c

This is the module that reads the port b, which comes from the Arduino and converts the byte received to a speed for the left motor and another speed for the right motor. Both can be consulted from any othe module calling the correspondent functions.

```
void ReadParallel(){
    unsigned char lecture=PORTB;
    unsigned char left=lecture>>4;//MSB
    unsigned char right=lecture & 0x0f;//LSB
    SpeedRight=ConvertParallelSpeed(right);
    SpeedLeft=ConvertParallelSpeed(left);

    if (left==0x0c){
        Mode=1;
    }else{
        Mode=0;
    }
}


/*Sends the lectures to other functions
 */
signed int  ReadParallelSpeedRight(){
    return SpeedRight;
}
signed int ReadParallelSpeedLeft(){
    return SpeedLeft;
}
unsigned int ReadMode(){
    return Mode;
}
```

```
/*This is the module that converts the two groups of 4 bits to
speeds for the motors
 * a 0x01 means -100, 0x02 for -80 ...
 * for 0x00, 0x06 and in default cases, the motors stop
 */
signed int ConvertParallelSpeed (char velocity){
    switch(velocity){
        case 0x01:
            return -100;
            break;
        case 0x02:
            return -80;
            break;
        case 0x03:
            return -60;
            break;
        case 0x04:
            return -40;
            break;
        case 0x05:
            return -20;
            break;
        case 0x06:
            return 0;
            break;
        case 0x07:
            return 20;
            break;
        case 0x08:
            return 40;
            break;
        case 0x09:
            return 60;
            break;
        case 0x0a:
            return 80;
            break;
        case 0x0b:
            return 100;
            break;
        default:
            return 0;
            break;
    }
}
```

6.1.4.7 Interrupts.c

This is the last module, here we have configured two different kind of interruption, one for the timer (to generate the PWM signals) and another to read the analog port.

```
#include <xc.h>
#include "ADC.h"
#include "PWM.h"
#include "TimerPWM.h"
#include <usart.h>
#define PIN1 RC5//Pins for the PWM signal
```

```
#define PIN2 RC4

void interrupt isr(void) {//One interrupt
    /* If the interrupt was caused by an ADC conversion complete
     * Here we will read one diferent input in each iteration of
the interrupt
     * First time port0, later port1... and when end, repeat from
the port0 again
     * It calls the functions of the ADC.c module to update the
variables
     */
    if (ADIF == 1) {

        static unsigned int Readport=0;
        unsigned int read=ADCGetOutput();

        switch (Readport){
            case 0://Read port 0. Left sensor.
                CHS3=0;
                CHS2=0;
                CHS1=0;
                CHS0=1;
                SetDistanceLeft (read);
                Readport++;
                break;
            case 1://Read port 1. Central sensor.
                CHS3=0;
                CHS2=0;
                CHS1=1;
                CHS0=0;
                SetDistanceCentral (read);
                Readport++;
                break;
            case 2://Read port 2. Central sensor.
                CHS3=0;
                CHS2=0;
                CHS1=1;
                CHS0=1;
                SetDistanceRight (read);
                Readport++;
                break;
            case 3://Read port 3. Battery voltage.
                CHS3=0;
                CHS2=0;
                CHS1=0;
                CHS0=0;
                SetBattery (read);
                Readport=0;
                break;
            default:
                CHS3=0;
                CHS2=0;
                CHS1=0;
                CHS0=0;
                Readport=0;
                break;
        }

        ADIF=0;//Reset interrupt flag
    }
```

```
    /*If the interrupt was caused by a timer overfloat.
     * This is used to generate by software the PWM signals
     * as this microcontroller can only manage by hardware one PWM
output.
     * The two PWM signals are generated with just one timer
     * The dutty cycle goes from 0 (100%) to 20 (0%)in steps of a
5%,
     * so the precission is not very hight, but enought for our
purpose
     */
    if (TMR0IF==1){

        static unsigned int PWMLeft;
        static unsigned int PWMRight;
        static unsigned int iteration;

        if (iteration==20){
            PIN1=0;
            PIN2=0;
            iteration=0;
            PWMRight=ReadPWMMotorRight();
            PWMLeft=ReadPWMMotorLeft();
        }
        if (PWMRight==iteration){
            PIN1=1;
        }
        if (PWMLeft==iteration){
            PIN2=1;
        }
        iteration++;

        TMR0IF=0;//Clear interrupt flag
        TMR0H=0x00;//Start the timer from 0
        TMR0L=0x156;
    }
}
```

6.2 Arduino

6.2.1 IDE (Arduino)

6.2.2 Programming the Arduino

6.2.3 Libraries:

Here we are going to use different libraries developed by the manufacturer of the FONA (Adafruit). This libraries are very helpful, as they contains all the necessary functions to read data from the GPS, send data throught the mobile internet connection and also includes the protocol used to stablish the connection between the Arduino and the servers. This protocol is calles MQTT and sends a push notification always that a value in

the server changes, so, it doesn't need to send request from the Arduino each time, saving data and time:

We will divide the code in parts to explain it better, but everything is in the same file (Not separated modules).

The used libraries developed by thirds are:

- Adafruit_SleepyDog.h (Adafruit 2016d)
    - This library is used to reset the Arduino is case that pass a certain amount of time withought reception of signal from the servers.

- SoftwareSerial.h (Arduino 2015)
    - The communication between the Arduino and the FONA board is make by a serial communication made by software (instead of using the embedded hardware pins)

- Adafruit_FONA.h (Adafruit 2016a)
    - This is the basic library with the functions to access to the FONA board

- Adafruit_MQTT.h (Adafruit 2016c)
    - MQTT protocol, to talk between the Arduino and a server with MQTT protocol support.

- Adafruit_MQTT_FONA.h (Adafruit 2016c)
    - Library to stablish a MQTT connection while using a FONA module.

6.2.4 Examples

A big part of the code is based in one example developed by Adafruit (Adafruit 2016b) to work with an Arduino and a Fona. The code has been adapted to work allright with our project. Also, the part of the GPS has been taken from another example also from Adafruit (Adafruit 2015) and again adapted to fit in the project.

6.2.5 Code

6.2.5.1 Headers

Here we have the invocation of the libraries, the definition of pins, global variables, etc.

```
#include <Adafruit_SleepyDog.h>
#include <SoftwareSerial.h>
#include "Adafruit_FONA.h"
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_FONA.h"
```

```
/*************************          FONA          Pins
**********************************/
#define FONA_RX 2
#define FONA_TX 10
#define FONA_RST 4
#define LedPin 6


//PIC Parallel Pins
#define P0 36
#define P1 34
#define P2 32
#define P3 30
#define P4 28
#define P5 26
#define P6 24
#define P7 22

//Global Variables
signed int  StartStopInt=0, counter=0;
float latitude, longitude, speed_kph, heading, altitude;

SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);

Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

/************************        WiFi        Access        Point
*******************************/
#define FONA_APN       "hs.vodafone.ie"
#define FONA_USERNAME  ""
#define FONA_PASSWORD  ""

/************************           Adafruit.io           Setup
*******************************/
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "jorgecrce"
#define AIO_KEY         "5c9f4f8e308a307c1594a197b4e6e028410dc3e1"

/************  Global  State  (you  don't  need  to  change  this!)
*****************/
// Store the MQTT server, username, and password in flash memory.
// This is required for using the Adafruit MQTT library.
const char MQTT_SERVER[] PROGMEM    = AIO_SERVER;
const char MQTT_USERNAME[] PROGMEM  = AIO_USERNAME;
const char MQTT_PASSWORD[] PROGMEM  = AIO_KEY;

// Setup the FONA MQTT class by passing in the FONA class and MQTT
server and login details.
Adafruit_MQTT_FONA    mqtt(&fona,    MQTT_SERVER,    AIO_SERVERPORT,
MQTT_USERNAME, MQTT_PASSWORD);

// You don't need to change anything below this line!
#define halt(s) { Serial.println(F( s )); while(1);  }

// FONAconnect is a helper function that sets up the FONA and
connects to
// the GPRS network. See the fonahelper.cpp tab above for the
source!
```

```
boolean FONAconnect(const __FlashStringHelper *apn, const
__FlashStringHelper *username, const __FlashStringHelper
*password);

/*****************************                                    Feeds
************************************/
// Setup a feed called 'StartStop' for subscribing to changes.
const char StartStop_FEED[] PROGMEM = AIO_USERNAME
"/feeds/StartStop";
Adafruit_MQTT_Subscribe StartStop = Adafruit_MQTT_Subscribe(&mqtt,
StartStop_FEED);

// Feed location for publish GPS coordinates
const char LOCATION_FEED[] PROGMEM = AIO_USERNAME
"/feeds/location/csv";
Adafruit_MQTT_Publish location_feed = Adafruit_MQTT_Publish(&mqtt,
LOCATION_FEED);

/*************************                Sketch                Code
**********************************/

// How many transmission failures in a row we're willing to be ok
with before reset
uint8_t txfailures = 0;
#define MAXTXFAILURES 3
```

6.2.5.2 Setup

This function runs only once. Here we configure the pins that will be use to send the data to the PIC, and also starts the FONA, connects it to the cellular network and connects it to the internet throught the GPRS network.

```
void setup() {

  pinMode(LedPin, OUTPUT);

  pinMode (P0, OUTPUT);
  pinMode (P1, OUTPUT);
  pinMode (P2, OUTPUT);
  pinMode (P3, OUTPUT);
  pinMode (P4, OUTPUT);
  pinMode (P5, OUTPUT);
  pinMode (P6, OUTPUT);
  pinMode (P7, OUTPUT);

  while (!Serial);

  // Watchdog is optional!  Better disabled...
  //Watchdog.enable(16000);

  Serial.begin(115200);

  Serial.println(F("Adafruit.IO to PIC Over Arduino"));

  mqtt.subscribe(&StartStop);
```

```
  Watchdog.reset();
  delay(5000);  // wait a few seconds to stabilize connection
  Watchdog.reset();

  // Initialise the FONA module
  while      (!      FONAconnect(F(FONA_APN),      F(FONA_USERNAME),
F(FONA_PASSWORD))) {
    Serial.println("Retrying FONA");
  }

  Serial.println(F("Connected to Cellular!"));

  Watchdog.reset();
  delay(5000);  // wait a few seconds to stabilize connection
  fona.enableGPS(true);
  delay(1000);
  Watchdog.reset();
  mqtt.connect();

}
```

6.2.5.3 Loop

This function runs continuously . here we wait for incoming packets from the MQTT server, and in case we receive a new packet, updates the data that is sending to the PIC. Also, each 20 iterations checks the GPS to send a new value to the server.

```
void loop() {
  Watchdog.reset();
  MQTT_connect();
  Watchdog.reset();

  // this is our 'wait for incoming subscription packets' busy
subloop
  Adafruit_MQTT_Subscribe *subscription;

  while ((subscription = mqtt.readSubscription(5000))) {
    if (subscription == &StartStop) {
      Serial.print(F("Lectura: "));
      Serial.println((char *)StartStop.lastread);
      char *value = (char *)StartStop.lastread;
      StartStopInt=atoi(value);
      PICSpeed(StartStopInt);
      digitalWrite(LedPin, HIGH);
    }
  }
  if (counter==20){//Send GPS coordinates one of each 20 iterations
    counter=0;
    bool  gpsFix  =  fona.getGPS(&latitude,  &longitude,  &speed_kph,
&heading, &altitude);
    logLocation(latitude, longitude, altitude, location_feed);
  }
  counter++;

  Watchdog.reset();
}
```

6.2.5.4 PICSpeed

This is the function that converts the number received from the server to a speed for the left motor (4 high significant bits) and le right motor (4 lowest significant bits). Later the pic makes the deconversion to values between -100 to +100

```
void PICSpeed(int value){//Left p7-P6-P5-P4 - Right P3-P2-P1-P0.
  switch (value){
    case -100://Front +80 - +80
      digitalWrite(P7, HIGH);
      digitalWrite(P6, LOW);
      digitalWrite(P5, HIGH);
      digitalWrite(P4, LOW);
      digitalWrite(P3, HIGH);
      digitalWrite(P2, LOW);
      digitalWrite(P1, HIGH);
      digitalWrite(P0, LOW);
      //return '1';
      break;
    case -80://Front Left +40 - +60
      digitalWrite(P7, HIGH);
      digitalWrite(P6, LOW);
      digitalWrite(P5, LOW);
      digitalWrite(P4, LOW);
      digitalWrite(P3, HIGH);
      digitalWrite(P2, LOW);
      digitalWrite(P1, LOW);
      digitalWrite(P0, HIGH);
      //return '2';
      break;
    case -60://Front Right +60 - +40
      digitalWrite(P7, HIGH);
      digitalWrite(P6, LOW);
      digitalWrite(P5, LOW);
      digitalWrite(P4, HIGH);
      digitalWrite(P3, HIGH);
      digitalWrite(P2, LOW);
      digitalWrite(P1, LOW);
      digitalWrite(P0, LOW);
      //return '3';
      break;
    case -40://Back -60 - -60
      digitalWrite(P7, LOW);
      digitalWrite(P6, LOW);
      digitalWrite(P5, HIGH);
      digitalWrite(P4, HIGH);
      digitalWrite(P3, LOW);
      digitalWrite(P2, LOW);
      digitalWrite(P1, HIGH);
      digitalWrite(P0, HIGH);
      //return '4';
      break;
    case -20://Back - Right -40 - -60
      digitalWrite(P7, LOW);
      digitalWrite(P6, HIGH);
      digitalWrite(P5, LOW);
      digitalWrite(P4, LOW);
```

```
  digitalWrite(P3, LOW);
  digitalWrite(P2, LOW);
  digitalWrite(P1, HIGH);
  digitalWrite(P0, HIGH);
  //return '5';
  break;
case 0://STOP
  digitalWrite(P7, LOW);
  digitalWrite(P6, LOW);
  digitalWrite(P5, LOW);
  digitalWrite(P4, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P2, LOW);
  digitalWrite(P1, LOW);
  digitalWrite(P0, LOW);
  //return '6';
  break;
case 20://Back left // -60 - -40
  digitalWrite(P7, LOW);
  digitalWrite(P6, LOW);
  digitalWrite(P5, HIGH);
  digitalWrite(P4, HIGH);
  digitalWrite(P3, LOW);
  digitalWrite(P2, HIGH);
  digitalWrite(P1, LOW);
  digitalWrite(P0, LOW);
  //return '7';
  break;
case 40://Right +40 - 0
  digitalWrite(P7, HIGH);
  digitalWrite(P6, LOW);
  digitalWrite(P5, LOW);
  digitalWrite(P4, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P2, LOW);
  digitalWrite(P1, LOW);
  digitalWrite(P0, LOW);
  //return '8';
  break;
case 60://Left 0 - +40
  digitalWrite(P7, LOW);
  digitalWrite(P6, LOW);
  digitalWrite(P5, LOW);
  digitalWrite(P4, LOW);
  digitalWrite(P3, HIGH);
  digitalWrite(P2, LOW);
  digitalWrite(P1, LOW);
  digitalWrite(P0, LOW);
  //return '9';
  break;
case 80: 40 - -40
  digitalWrite(P7, HIGH);
  digitalWrite(P6, LOW);
  digitalWrite(P5, LOW);
  digitalWrite(P4, LOW);
  digitalWrite(P3, LOW);
  digitalWrite(P2, HIGH);
  digitalWrite(P1, LOW);
  digitalWrite(P0, LOW);
  //return 'a';
  break;
```

```
    case 100://-40 - +40
      digitalWrite(P7, LOW);
      digitalWrite(P6, HIGH);
      digitalWrite(P5, LOW);
      digitalWrite(P4, LOW);
      digitalWrite(P3, HIGH);
      digitalWrite(P2, LOW);
      digitalWrite(P1, LOW);
      digitalWrite(P0, LOW);
      //return 'b';
      break;
    case 1://Automatic mode
      digitalWrite(P7, HIGH);
      digitalWrite(P6, HIGH);
      digitalWrite(P5, LOW);
      digitalWrite(P4, LOW);
      digitalWrite(P3, LOW);
      digitalWrite(P2, LOW);
      digitalWrite(P1, LOW);
      digitalWrite(P0, LOW);
      //return 'c' in left;
      break;
    default:
      digitalWrite(P7, LOW);
      digitalWrite(P6, LOW);
      digitalWrite(P5, LOW);
      digitalWrite(P4, LOW);
      digitalWrite(P3, LOW);
      digitalWrite(P2, LOW);
      digitalWrite(P1, LOW);
      digitalWrite(P0, LOW);
      //return '0'; //(Speed=0)
      break;
  }
}
```

6.3 Control Webpage

For the remote control over a web browser, has been chose to use the service Adafruit IO, again for it's easy integration with the FONA module (The examples used to develop the Arduino software are based on Adafruit IO. It is service where you can create feeds of data. This feeds saves a integer number and can also save GPS coordinates.

We are using two feed, the first one is called StartStop and it is the responsible of start the car and send specific values. From example the value 0 is to stop, the value 1 is to go on automatic mode, -100 means to go to the front and a little to the right… The value received by the Arduino from this feed is interpreted in the function PICSpeed to send precise speed values to both motors.

There is a second feed, called location, with saves the GPS streams send directly from the Arduino each 20 iterations of the main code.

//Añadir capturas de pantalla de la web

7 Usage

For the utilization, we have three dashboards:

7.1 Mode dashboard:

Switch between manual and automatic modes.

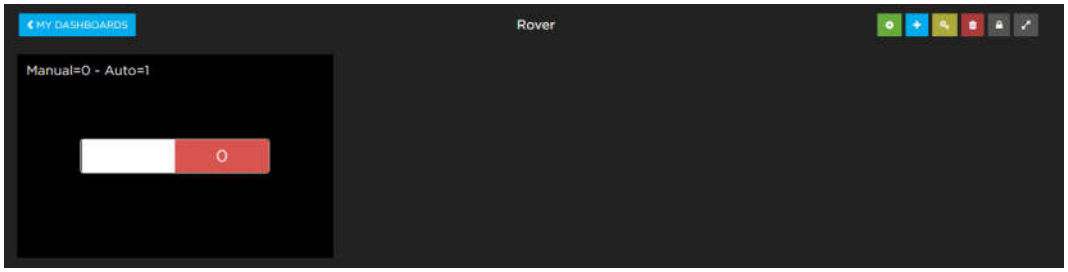


Illustration 28. Mode dashboard

7.2 Manual move dashboard:

Orders to move the rover manually:

Illustration 29. Manual dashboard

7.3 GPS dashboard:

Show the GPS coordinates in a map.

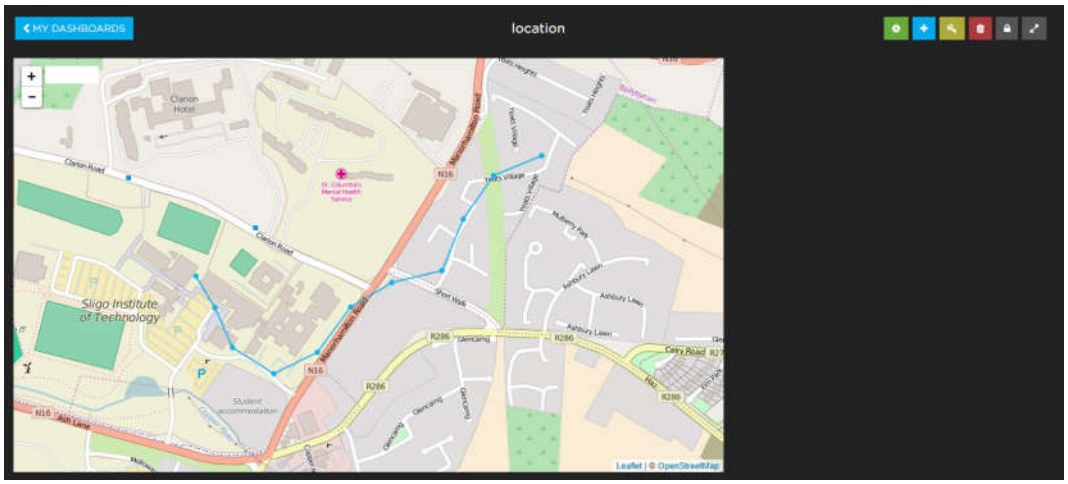


Illustration 30. Location dashboard

8 Final conclussions

During the execution of this project, many areas studied as part of my degree in **electronic engineering** have been utilized, including mounting **analog** and **digital** electronic components and developing all the program for the **PIC** microcontroller and for the **Arduino**. This project has also covered areas outside the scope of my degree, for example **informatics** and **automation** and **mechatronics** engineering.

9 Links to resources

In the Github of there project you can download all the code, this project, photos, the poster a video and anything related with the project:

https://github.com/jorgecrce/IOT-Rover

10Illustration table

11 References

Adafruit, Fona 808. Available at: https://www.adafruit.com/product/2542https://www.adafruit.com/product/2542 [Accessed May 19, 2016].

Adafruit, 2016a. Fona library. Available at: https://github.com/adafruit/Adafruit_FONA.

Adafruit, 2015. GPS Fona Example. Available at: https://github.com/openhomeautomation/arduino-geofencing/tree/master/geo_fencing_io.

Adafruit, 2016b. MQTT FONA example. Available at: https://github.com/adafruit/Adafruit_MQTT_Library/tree/master/examples/mqtt_fona.

Adafruit, 2016c. MQTT libraries. Available at: https://github.com/adafruit/Adafruit_MQTT_Library.

Adafruit, 2016d. SleepyDog library. Available at: https://github.com/adafruit/Adafruit_SleepyDog.

Arduino, 2016. Arduino. Available at: http://www.arduino.cc/.

Arduino, 2015. Software Serial library. Available at: https://www.arduino.cc/en/Reference/SoftwareSerial.

BUTTAY, C., 2006. H_bridge @ upload.wikimedia.org. Available at: https://commons.wikimedia.org/wiki/File:H_bridge_operating.svg.

Creativecommons.org, creativecommons.org. Available at: https://creativecommons.org/licenses/by/2.0/https://creativecommons.org/licenses/by/2.0/.

Electrosome, 16x2-Character-LCD @ electrosome.com. Available at: https://electrosome.com/wp-content/uploads/2013/05/16x2-Character-LCD.jpg.

Extremeelectronics, 2013. LCD Library for PIC16. Available at: http://extremeelectronics.co.in/pic16f877a-tutorials/setup-and-use-of-lcd-library-for-pic/http://extremeelectronics.co.in/pic16f877a-tutorials/setup-and-use-of-lcd-library-for-pic/.

gnu, Quick guide GPLv3. Available at: http://www.gnu.org/licenses/quick-guide-gplv3.htmlhttp://www.gnu.org/licenses/quick-guide-gplv3.html.

Google, 2015. Google self-driving car project. *Google Self-Driving Car Project*. Available at: https://www.google.com/selfdrivingcar/ [Accessed January 18, 2015].

Hacks, C., SIM908. Available at: https://www.cooking-hacks.com/gprs-gps-quadband-module-for-arduino-sim908 [Accessed April 19, 2016].

Hitachi, 1998. Hd44780. *Datasheet*, pp.1–60. Available at: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf.

Labyteacora, hc-sr04. Available at: https://labyteacora.files.wordpress.com/2015/07/front1.jpg.

Microchip, 2010. PIC18F45K20 Datasheet. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/41303G.pdf [Accessed January 15, 2015].

RCUniverse, Discharge curves for Li-PO and Ni-MH. Available at: http://www.rcuniverse.com/forum/attachment.php?attachmentid=1326435&d=1375489161.

Sharp, Sharp GP2Y0A21YK. Available at: http://cdn.active-robots.com/media/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/g/p/gp2d12_2.jpg.

SparqEE, SparqEE CELLv1.0. Available at: http://www.sparqee.com/portfolio/sparqee-cell/ [Accessed May 19, 2016].

STMicroelectronics, 2000. L298 Dual Full-bridge driver Datasheet. , pp.1–13.

Techinc, Lm7805 Diagram. Available at: https://wiki.techinc.nl/images/5/54/Lm7805-pinout-diagram.gif.

TeslaMotors, 2015. Tesla Motors. Available at: https://www.teslamotors.com/ [Accessed January 18, 2015].

Texas Instruments, 2015. µA78Mxx Positive-Voltage Regulators Datasheet. Available at: http://www.ti.com/lit/ds/symlink/ua78m33.pdf.

Texas Instruments, LM2596 Diagram. Available at: http://www.ti.com/diagrams/custom_diagram_1_LM2596.gif.

Texas Instruments, 2013. LM2596 SIMPLE SWITCHER ® Power Converter 150 kHz 3A Step-Down Voltage Regulator Datasheet. , (April). Available at: http://www.ti.com/lit/ds/symlink/lm2596.pdf.

Texas Instruments, 1995. LM78XX Series Voltage Regulators LM78XX Series Voltage Regulators Datasheet. , (February), pp.1–6. Available at: http://www.ti.com/lit/ds/symlink/lm7805c.pdf.

Vishay, S., 2002. General Purpose Plastic Rectifier 1N4001 thru 1N4007 Vishay. , (1), pp.1–5. Available at: http://www.vishay.com/docs/88503/1n4001.pdf.

Wapcaplet, 2006. Electric_motor_cycle_2 @ upload.wikimedia.org. Available at: https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_2.png.