AGH University of science and technology

Automatyka i Robotyka

Final project report

Embedded Systems I

Student:

JHON VELASQUEZ

Professor:

dr inż. Mariusz Pauluk

Kraków, June 13th 2020

CONTENT INDEX

Introduction

The present report describes the process of completing the final project of the course Modelling and Simulation of Cyber-Physical Systems, the design and implementation of a follower line robot that avoid obstacles in its path. Even though the objective has not been achieved, it was obtained the control models for each motor of the mobile robot. There are also mentioned the problems related to the performance of the car due to the lack of quality on the sensors. Conclusions that can be taken into account for future projects are shown at the end.

Review of Related Literature

The presented Project was proposed by the professor in charge, who also provided most of hardware components.

2.1. Robot traction system configuration

The robot is a three-wheeled differentially robot. It has two independent wheels, which permits fair maneuverability. In the next figure, it can be shown what has been described and how the robot's frame was received at the beginning of the project.

Figure 1. Bottom view of the robot

Design

3.1. General schematic

In the figure below is shown the schematic of the system. There are the physical components and how they interact between them.

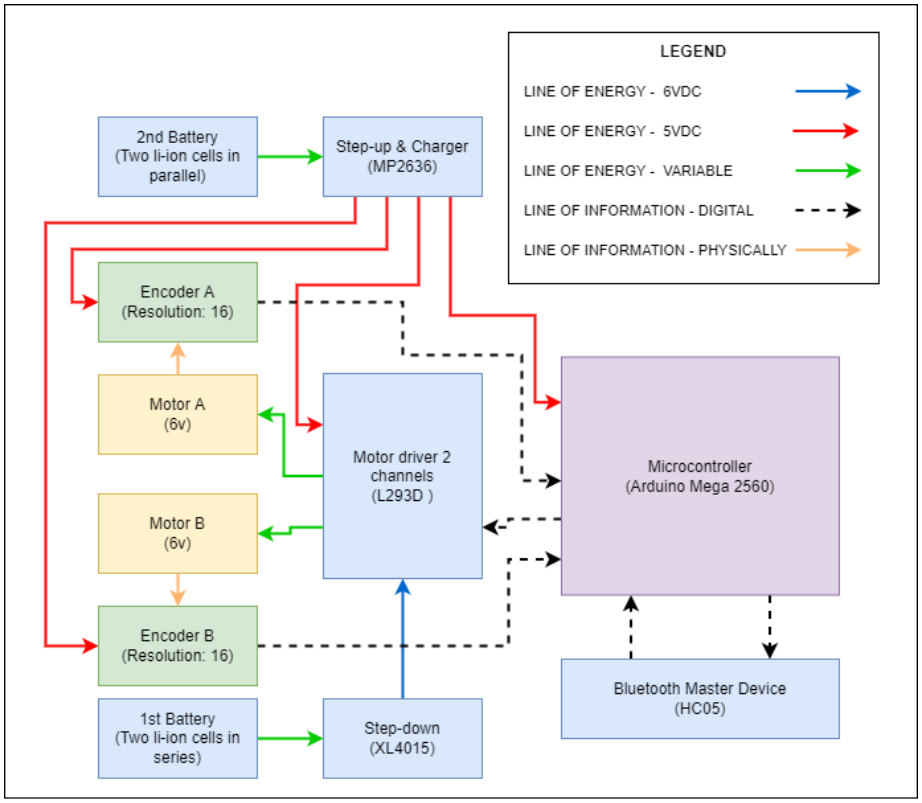

Figure 2. General schematic

3.2. Energy system

The energy system is composed by two separated energy source providers:

- 6VDC source: 2 Li-ion cells in series for obtaining nominal 7.4V so it can be reduced to 6VDC by the XL4015 step-down module.
- 5VDC source: 2 Li-ion cells in parallel for obtaining nominal 3.7V so it can be boosted to 5VDC by the MP 2636 step-down module.

At the beginning there was only the 5VDC source, however it didn’t provide enough current. This was shown by the low speed of motors. As an improvement, 6VDC were obtained from two Li-ion cells in series, as the cells were already available. Additionally, separating the high current consuming components from the digital components, helps to reduce the noise on these.

Figure 3. 2 Li-ion cells in parallel, nominal 3.7V

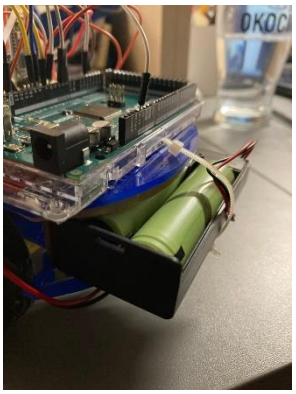

Figure 4. Li-ion cells in series, nominal 7.4V

3.3. Comunication system

The microcontroller is connected to a HC-05 Bluetooth module (Figure 5), which is set up as a master device. Its purpose is for showing data of the system on real time and for allowing the input of instructions.

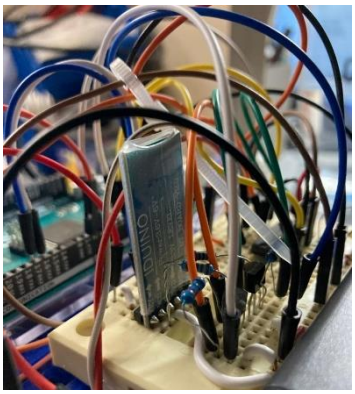

Figure 5. HC-05 Bluetooth module

As part of the project, there is a HC-06 BT Slave Module linked to the HC-05 master in the robot. The HC06 is also connected to a USB to TTL module, so it can send an receive data straight from the USB port on a laptop without Bluetooth (Figure 6).

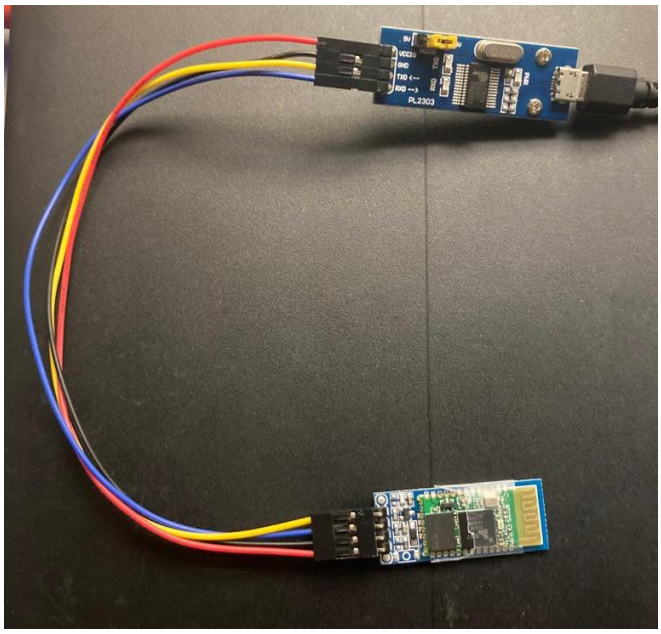

Figure 6. HC-06 BT Module connected to a TTL-USB module adapter

3.4. Sensing system

The only sensors in the system are the same, encoders (Figure 7. Sparkfun double channel encoder). It is from Sparkfun and use the reflection of infrared light principle for detecting the tooth of the a disk connected to the motor's shaft.

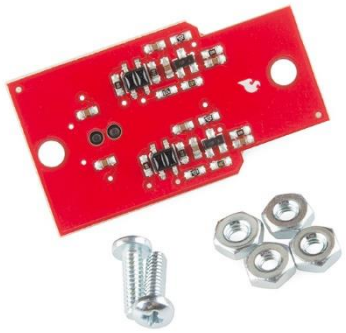

Figure 7. Sparkfun double channel encoder

About the toothed disk, it came with the robot's frame. It was 3D printed and has a resolution of 16 (Figure 8). In the same image, it can be observed how is the disk connected to the motor's shaft.

Figure 8. Disk of the encoder

3.5. Actuation system

In the system, there are only two actuators, two motors. They can be seen in Figure 8, each one is a DC Motor connected to a gearbox with 48:1 ratio of velocity reduction.

3.5.1. Motor driver L293D

So these components can interact with the microcontroller, there's the two channel driver L293D.

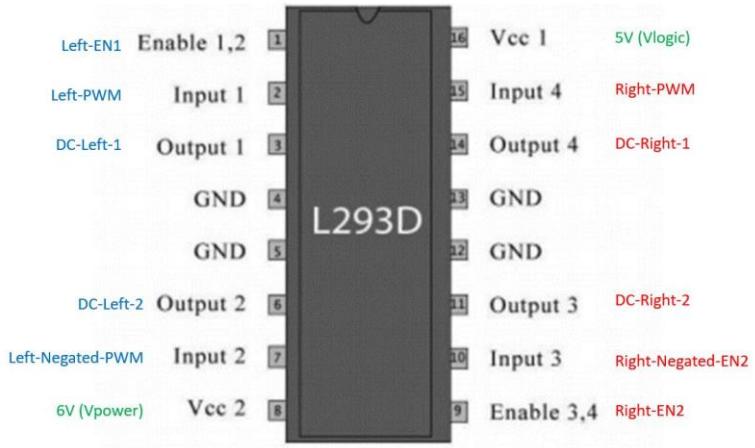

Figure 9. Two channel motor driver with pin description

3.5.2. Signal adaptation for the driver

The driver shown in Figure 9 is usually controlled with three signals with the logic shown in theTable 1.

Table 1. Common motors driver input use

Enable 1,2	Input 1 (I1)	Input 2 (I2)	Motor rotation
High	High	High	Stop
High	High	Low	Clockwise
High	Low	High	Anti-clockwise
Low	x	x	Stop

However, in order of reducing the control to two signals and for improving the code in charge of making the motors work, it has been used another configuration of the pins, the logic is shown in Table 2.

Table 2. Driver pin usage in the project

Enable 1,2	Input 1 (I1)	Input 2 (I2)	Motor rotation
High	PWM	Negated PWM	Clockwise, Anti-clockwise and Stop
Low	X	X	Stop

In order to have an effect in the motor rotation, it is necesary to change the PWM duty cicle. As it is seen in Figure 10, the average voltage is calculated as the average value in a period of time. If the duty cycle is 50%, it means that the average velocity is zero. If the duty cycle is over 50%, the average value is positive; and gets to be the máximum positive value when the duty cycle turns 100%. It Works in the same way for negative values.

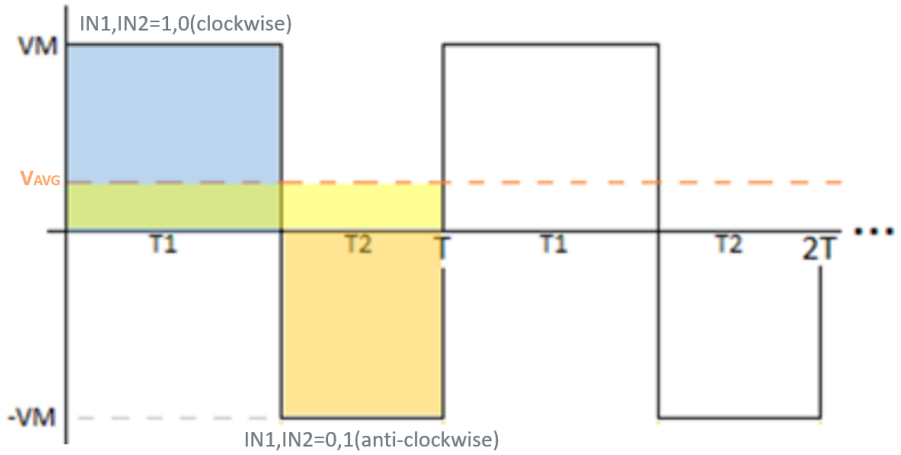

Figure 10. Obtaining an average voltaje through driver inputs

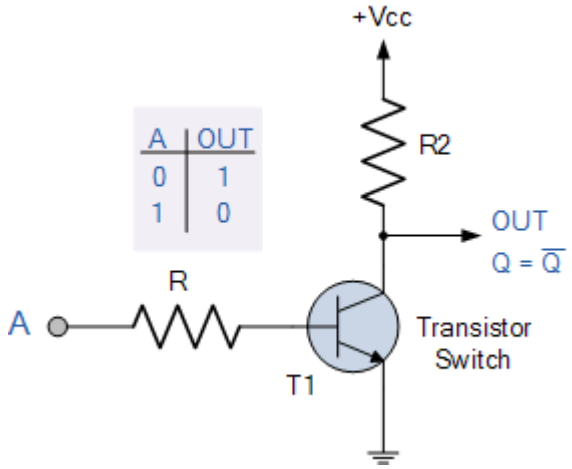

Figure 11. Transistor configuration used for obtaining negated PWM

3.6. Control system

3.6.1. System identification

It has been prepare a code for obtaining the step response of the motors. The y axis is in RPS.

MotorA (Left)	MotorB (Right)
0.69	0.99
0.69	0.99
0.69	0.99
0.69	0.99
0.69	0.99
1.22	1.35
1.22	1.35
1.22	1.35
1.22	1.35
1.56	1.73
1.56	1.73
1.56	1.73
1.8	1.95
1.8	1.95
1.8	1.95

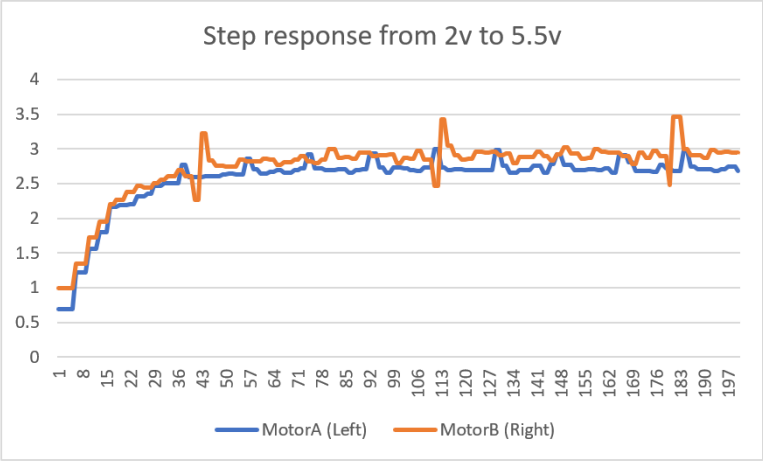

Figure 12. Step responses of the motors

Using Matlabs System Identification Toolbox it was possible to get each model in continuous-time transfer function, see Figure 13 and Figure 14.

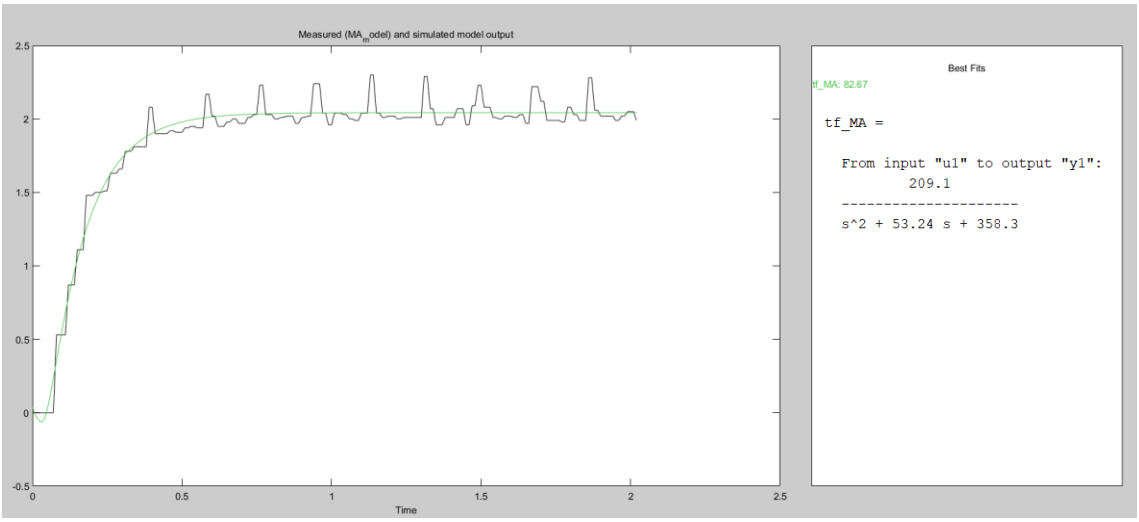

Figure 13. Fitted model for MotorA and step response

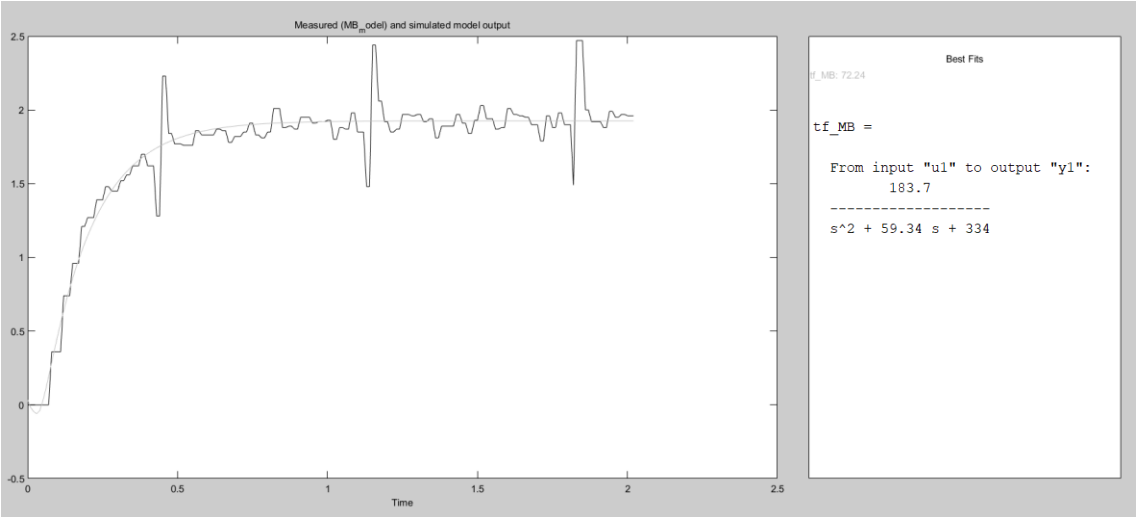

Figure 14. Fitted model for MotorB and step response

3.6.2. Control design

An script for obtaining the parameters for the controller in space state automatically was done. It takes the models and gives the values of the proportional gains and integral gains, as well as gains for observer matrix.

The process is explained in the next sections from the transfer function and through the MatlabScript.

3.6.2.1. Obtaining parameters of transfer function

```
%Loading transfer funciton model for MotorB
tf_MB=load("ES_transfer_models_A_B.mat").tf_MB;

%Obtaining parameters of transfer function
MB_a1=tf_MB.Numerator;
MB_b1=tf_MB.Denominator(1);
MB_b2=tf_MB.Denominator(2);
MB_b3=tf_MB.Denominator(3);
```

Figure 15. Obtaining parameters of transfer function

3.6.2.2. Obtaining Integral-Plant Space State model

```
%Obtaining Integral-Plant Space State model
MB_A_I=[(-1*MB_b2/MB_b1) (-1*MB_b3/MB_b1) 0;1 0 0;0 1 0];
MB_B_I=[(MB_a1/MB_b1);0;0];
MB_C_I=[0 1 0];
MB_D_I=0;
MB_ss_I=ss(MB_A_I,MB_B_I,MB_C_I,MB_D_I);
```

Figure 16. Obtaining Integral-Plant Space State model

3.6.2.3. Obtaining discretized Integral-Plant Space State model matrixes

```
%Obtaining discretized Integral-Plant Space State model matrixes
ss_disc_MB_I=c2d(MB_ss_I,dt);
MB_d_A_I=ss_disc_MB_I.A;
MB_d_B_I=ss_disc_MB_I.B;
MB_d_C_I=ss_disc_MB_I.C;
MB_d_D_I=ss_disc_MB_I.D;
```

Figure 17. Obtaining discretized Integral-Plant Space State model matrixes

3.6.2.4. Obtaining discretized Plant Space State matrixes

```
%Obtaining discretized Plant Space State matrixes
MB_d_A=MB_d_A_I(1:2,1:2);
MB_d_B=MB_d_B_I(1:2);
MB_d_C=MB_d_C_I(1:2);
MB_d_D=MB_d_D_I;
```

Figure 18. Obtaining discretized Plant Space State matrixes

3.6.2.5. Obtaining desired poles for proportianl-integral and observator gain

```
phi=0.6;
wn=8;
M=exp(-(phi*pi/((1-phi^2)^0.5)))*100;
Ts=4/(wn*phi);
s1=-1*phi*wn+wn*sqrt(1-phi^2)*1i;
s2=-1*phi*wn-wn*sqrt(1-phi^2)*1i;
s3=-10*phi*wn;
z1=exp(s1*dt);
z2=exp(s2*dt);
z3=exp(s3*dt);

s1_o=-15*phi*wn+wn*sqrt(1-phi^2)*1i;
s2_o=-15*phi*wn-wn*sqrt(1-phi^2)*1i;
z1_o=exp(s1_o*dt);%25
z2_o=exp(s2_o*dt);%25
```

Figure 19. Obtaining desired poles for proportianl-integral and observator gain

3.6.2.6. Calculating proportianl-integral and observator gain

```
KB = acker(MB_d_A,MB_d_B,[z1 z2]);
KB_I= acker(MB_d_A_I,MB_d_B_I,[z1 z2 z3]);
LB= acker(MB_d_A',MB_d_C',[z1_o z2_o])';
```

Figure 20. Calculating proportianl-integral and observator gain

3.6.2.7. Simulink simulation

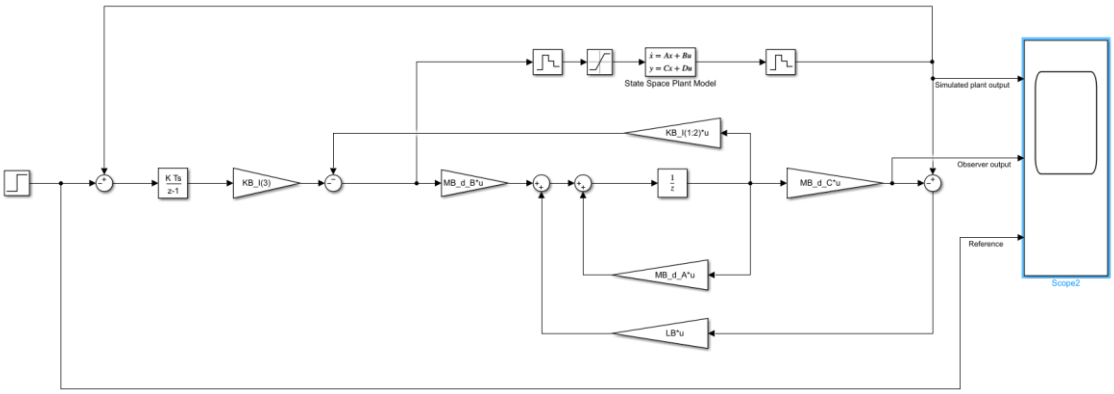

Figure 21. Simulink model

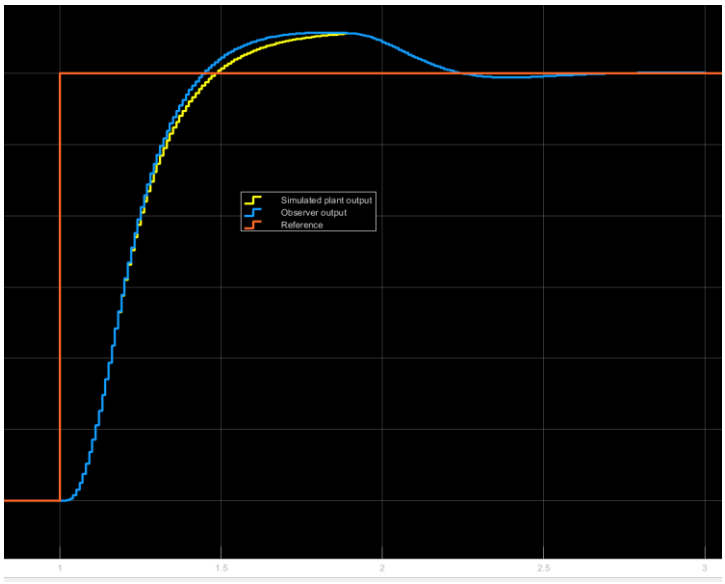

Figure 22. Simulink model response

3.6.3. Route generation

For obtaining the values of the velocity of each motor for a given radio, there has been implemented a function, which can be understood with the Figure 23.

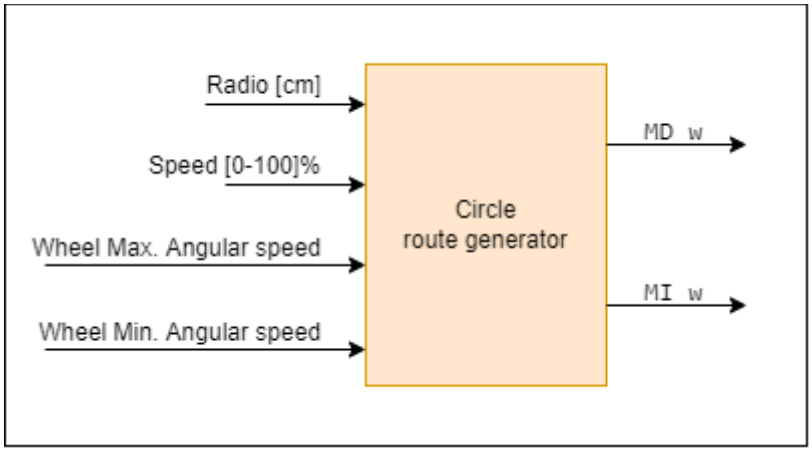

Figure 23. Circle route generator function block

The code was first implemented in Matlab and when the correct results were comprobed, the equations were writen in c++ along the control code.

```
rt=20; %radio del circulo [cm]
circle_speed=80; %percentage of velocity [0-100]%

s=13;
M_d=6.5;
Theta_vel=0;
Theta_vel_min=0;
Theta_vel_max=0;
M_vel_min=-2.5;%RPS
M_vel_max=2.5;%RPS

MI_Theta_vel_min=M_vel_min*M_d/(2*rt+s)
MI_Theta_vel_max=M_vel_max*M_d/(2*rt+s)

MD_Theta_vel_min=M_vel_min*M_d/(2*rt-s)
MD_Theta_vel_max=M_vel_max*M_d/(2*rt-s)

if MI_Theta_vel_min<MI_Theta_vel_max
    a=MI_Theta_vel_min;
    b=MI_Theta_vel_max;
else
    b=MI_Theta_vel_min;
    a=MI_Theta_vel_max;
end

if MD_Theta_vel_min<MD_Theta_vel_max
    c=MD_Theta_vel_min;
    d=MD_Theta_vel_max;
else
    d=MD_Theta_vel_min;
    c=MD_Theta_vel_max;
end
```

Figure 24. First part of Matlab script for obtaining circle route

```

if (a<=c)
       Theta_vel_min=c
elseif (a>c && d>a)
       Theta_vel_min=a
else
       Theta_vel_min=0
end

if (d<=b)
       Theta_vel_max=d
elseif (d>b && b>c)
       Theta_vel_max=b
else
       Theta_vel_max=0
end

Theta_vel_min=0;
Theta_vel=Theta_vel_min+(circle_speed/100)*(Theta_vel_max-(Theta_vel_min));

MD_w=Theta_vel*(2*rt-s)/(M_d)
MI_w=Theta_vel*(2*rt+s)/(M_d)
```

Figure 25. Second part of Matlab script for obtaining circle route

3.6.4. Control implementation

Taking the model seen in Figure 21 and with the matrixes on section 11, the next function is obtained in c++. This is invoked each 10ms, after sensing the velocity

```
void controlarVelocidadMotorB(float vel_ref){
  if(((MB_varVel-vel_ref)>0 && (MB_u>0)) || ((MB_varVel-vel_ref)<0 && (MB_u<0)))
    e_u_equal_B=1;
  else
    e_u_equal_B=0.0;

  if(e_u_equal_B && satB)
      MB_intError=((MB_varVel-vel_ref-(MB_u_saturated-MB_u)*20)*dt)+MB_intError;
  else
      MB_intError=((MB_varVel-vel_ref)*dt)+MB_intError;

  MB_u=-MB_ace_bef*MB_K_11-MB_vel_bef*MB_K_12-MB_intError*MB_K_13;

  if((MB_u>Vm)){
    MB_u_saturated=Vm;
    satB=1;
  }else if((MB_u<-Vm)){
    MB_u_saturated=-Vm;
    satB=1;
  }else{
    MB_u_saturated=MB_u;
    satB=0.0;
  }

  motorVoltaje(pinPWMB,MB_u_saturated);
  MB_ace_bef=(MB_d_B11*MB_u)+MB_L_11*(MB_varVel-MB_vel_bef)+(MB_d_A11*MB_ace_bef+MB_d_A12*MB_vel_bef);
  MB_vel_bef=(MB_d_B21*MB_u)+MB_L_21*(MB_varVel-MB_vel_bef)+(MB_d_A21*MB_ace_bef+MB_d_A22*MB_vel_bef);
}
```

Figure 26. Implementd code in c++ for controlling the motor

3.6.5. Results

The final car's assembly is shown in the figure below.

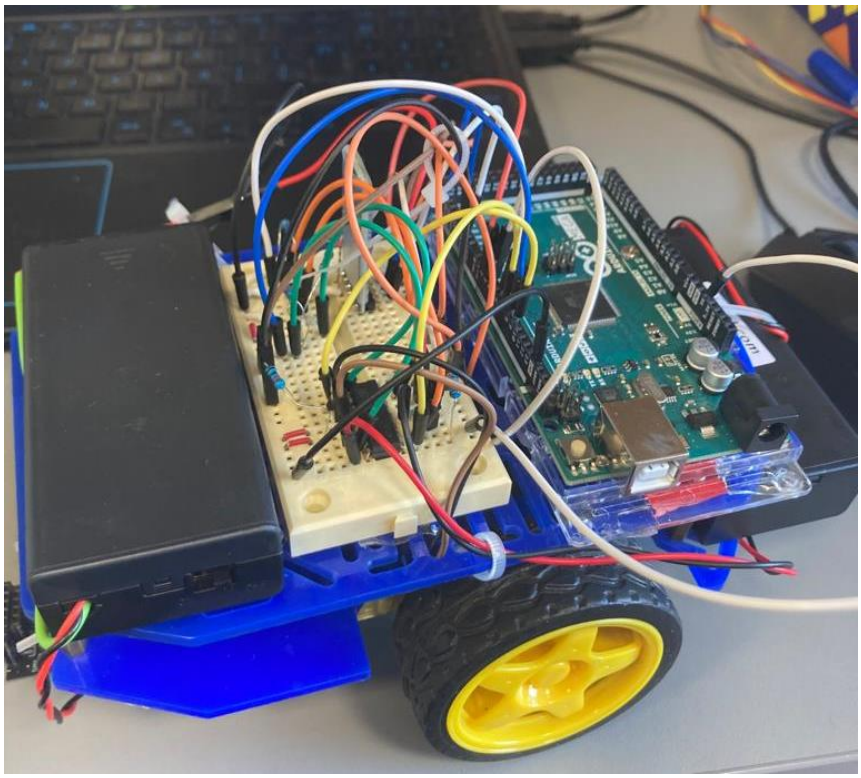

Figure 27. Final assembly

The step response with the motor's velocities controlled are shown below.

Figure 28. Step response of the controlled motors

In Figure 29, it is observed how the motors follow different references even in negative directions.

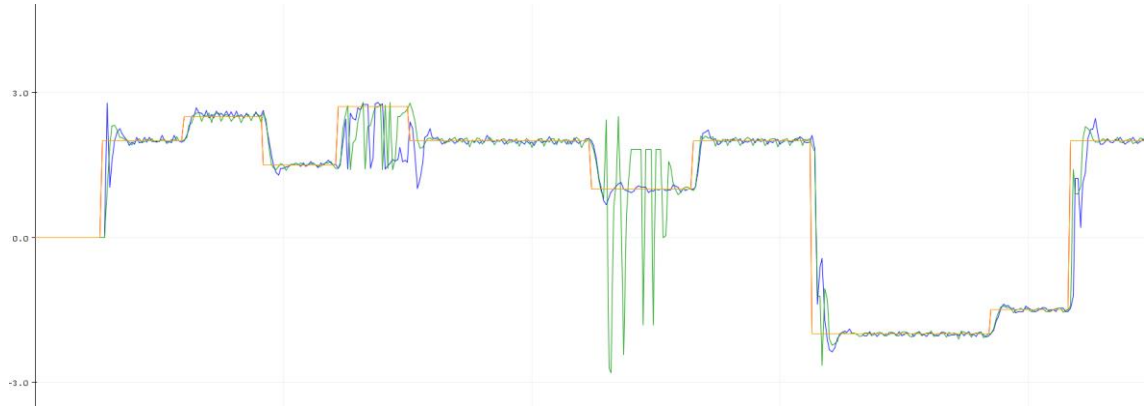

Figure 29. Different references for the motors and controlled response

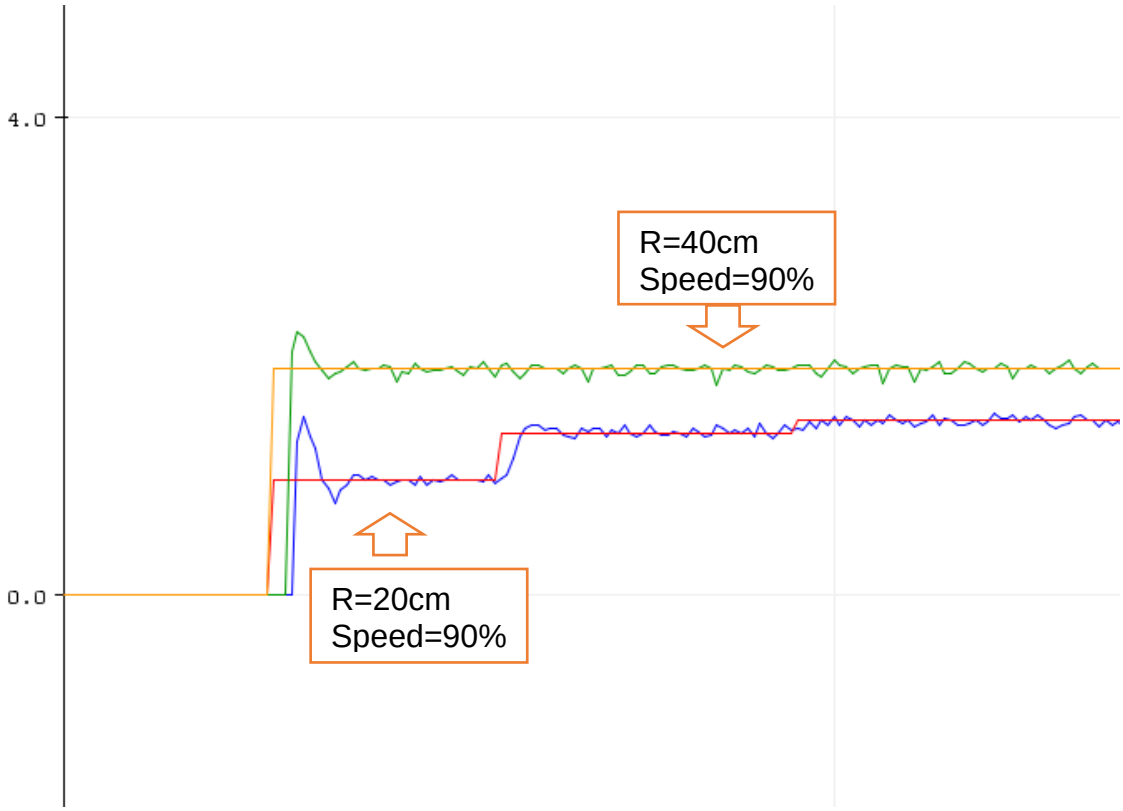

Figure 30. Reference for 20cm and 40cm of circle to draw with 90% percent of speed

3.6.6. Problems encountered

- As there was not a double channel encoder, it was no table to sense the direction of rotation. As a simplification, the sign of the velocity was considered the same as the velocity reference. This provoques error in the sensing such the negative peaks on the Figure 31. Problems generated by not having rotation sensor, these happing when the reference is low such below 1 RPS, however still makes it work. There was many problems and approaches when trying to solve this by code.

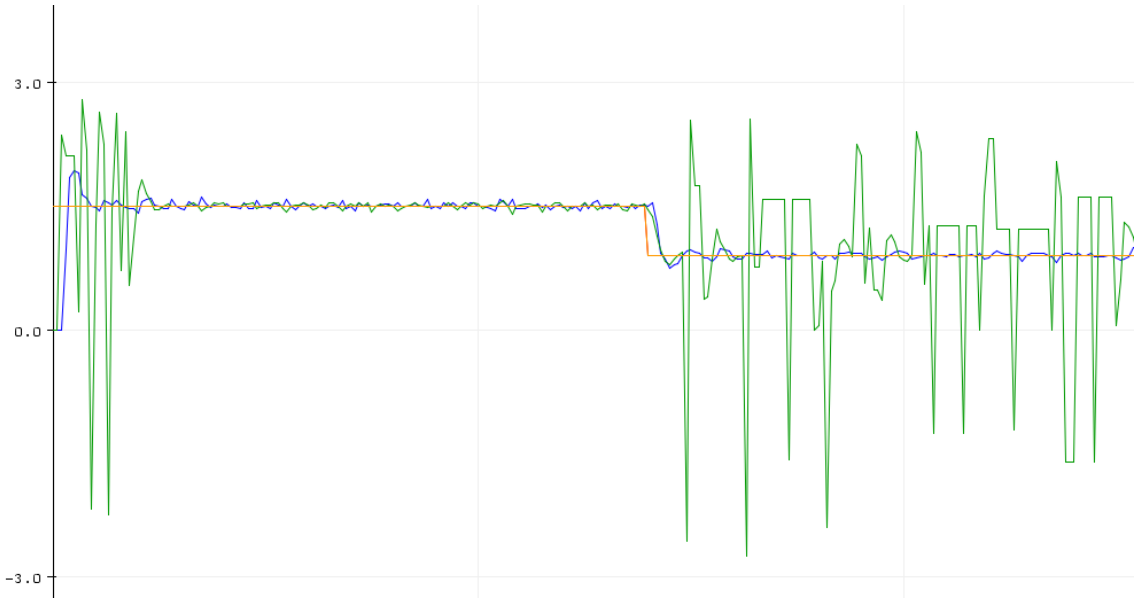

Figure 31. Problems generated by not having rotation sensor

- The low resolution of the enconder introduces considerable error for low speeds. These values are below 1.5 RPS or higher values on the period of 41.6 ms. This is also because the period is larger in low speeds and the controller consider that there are no changes in speed when increasing the control variable, so it ends in overshooting.

- Data transmisión, in the developed program executed sequentially, via BT at speed of 9600 bps provoques considerable delays as it takes 0.8 ms in sending only 1 byte. However, higher speeds provoque lose in data as the connection is BT, and possibly beacuse noise generated by the motors.

- In the sequential execution code, when having two encoders, there can be observed lower velocity readings in the one that is after. The reason is that it has to wait for the first reading to be done.

3.6.7. Conclusions

- The set up of a mobile robot with control of each motor has been prepared and well documentated.

- It is needed a two-channels encoder for having the information of the rotation direction.

- It should be considered to increase the current resolution of 16.

- The errors in sensing motor’s velocity and comunication velocity, introduced by the fact that the code is executed sequentially, suggest there could be an improvment if it is excecuted in an Real Time Operating System enviorenment.