Final Project Report: Obstacle Avoiding Vehicle

Adnan Ahmetović

Electrical and Electronics Engineering, Faculty of Engineering and Natural Sciences,
International University of Sarajevo

EE325: Embedded Systems

Dr. Tarik Namas

12. June, 2021

Table of Contents

Abstract

One of the most important aspects of autonomous driving vehicles' pick and place tasks is trajectory planning. In this project, I created a ‘rough’ version of an autonomous driving vehicle that is both compact and fully functional.

This ‘rough’ smart-vehicle is built to sense any obstacle in its path, avoid it and adapt its trajectory to the real-time situation. Ultrasonic sensor was adapted to implement a real-time obstacle avoidance system for this RWD vehicle, so that the vehicle can continually detect surroundings, avoid obstacles, and continue moving.

This model has a wide range of applications, including vacuum cleaners, parking systems, scientific exploration, emergency rescue, and other isolated environments.

I have used an Arduino UNO with a Motor Shield along with geared DC motors to make the vehicle, and for sensing, I choose an Ultrasonic Sensor, which detects any obstacles in the smart vehicle’s path accurately and efficiently. The Arduino is programmed to move the smart-vehicle backward and find another path, when an obstacle appears in front of it.

Throughout the construction of this model, I extended my knowledge about Arduino coding language, the Motor Shield functionality, and the operation and features of an ultrasonic sensor.

Finally, the goal of this project is to create a smart vehicle model that is beneficial to the todays autonomous driving restrictions.

Introduction

Problems addressed

- Need for a vehicle that can perform trajectory planning effectively;
- Need for a system capable of detecting obstacle and moving along a predetermined path;
- Need for the detection of unexpected impediments;
- Need to minimize the risk regarding the upper limit of a human eye;
- Need to assist the physically handicapped, by incorporating new technologies in wheelchairs;
- Need to address the flaws of inaccurate sensors like CCD cameras, CMOS image sensors, GPS, and so on;
- Need to address requirements of advanced mapping devices during exploration of unknown environments;

Solutions offered

- Vehicle recognizes and avoids obstacles in the trajectory
- Vehicle employs an Ultrasonic Sensor, which is capable of detecting impediments that may occur unexpectedly, such as an animal in front of a wheelchair;
- Vehicle can work without significant disturbance in the environment;
- Vehicle can identify miniature features that the human eye may miss;
- Vehicle can map different topographies and terrains;

Components Used

- Arduino Uno;
- Motor Driver- L298N;
- Ultrasonic Sensor- HC-SR04;

- Servo Motor
- Jumper Wires;
- 2x DC Motors;
- 2x Gears for DC Motors;
- 2x 9V Batteries;
- Cardboard Chassis/Frame;
- Plastics Reinforcements;
- 2x Rear Wheels (ø70mm);
- 2x Front Wheels (ø25mm);
- Front Wheels Axle;
- Standard DC Connector (5.5 mm x 10mm);
- 2x Capacitors for DC Motor;
- Switch;
- Boost Converter:

    - 2x Diodes (>12V ratings);
    - Inductor/Transformer (Step-up coils);
    - Polarized Capacitor (1000 μF);
    - MOSFET (high switching frequency required~ AN7915T or else);
    - NPN Transistor (2n2222) {MOSFET alternative};
    - Resistors;
    - Switch;

Features of the components used

Arduino Uno Microcontroller

An Arduino is a open-source microcontroller development board. The Arduino Uno is the most widely used version of the Arduino. When compared to other development boards, it is reasonably inexpensive, plugs directly into a computer's USB connection, and is simple to set up and operate. I choose this board instead of course default (piccolo C2000) for my project just because of errors I was facing with T.I. library pack software (CCS).

Figure 1. (Arduino Uno)

Some of the key features of the Arduino Uno include:

- Open source design. Large existing community at arduino.cc/forum/ which makes troubleshooting straight forward .
- Easy USB interface. It contains the USB interface support which enables the board to act as a serial device and provide the functionality to connect the board to other interfaces.
- Convenient power management and built-in voltage regulation. Up to 12v can easily be regulated to both 5v and 3.3v
- Easy-to-find and cheap, microcontroller

- The Microcontroller ATmega328p which is used on the Arduino UNO board is easily available. The board contains other components like Timers, external and internal interrupts,PWM pins and Multiple sleep modes.
- The pins used in the board acts as an oscillator that has a frequency of 16Mhz which is fast enough for most applications.

Some other specifications are

- A 16 MHz clock.
- 32 KB of flash memory
- 13 digital pins and 6 analogue pins.
- ICSP connector to re-bootload your chip and for bypassing the USB port and interfacing the Arduino directly as a serial device
- LED attached to digital pin 13 for and easy debugging of code.
- Reset button to reset the program on the chip.

Servo Motor

Servo motor is a device that is used to turn to a specific position. Their turning circle is usually 180°. By using Arduino we can rotate it to a specific given position. A servo motor has everything built in: a motor, a feedback circuit, and most important, a motor driver. It just needs one power line, one ground, and one control pin.

Figure 2. (Servo Motor)

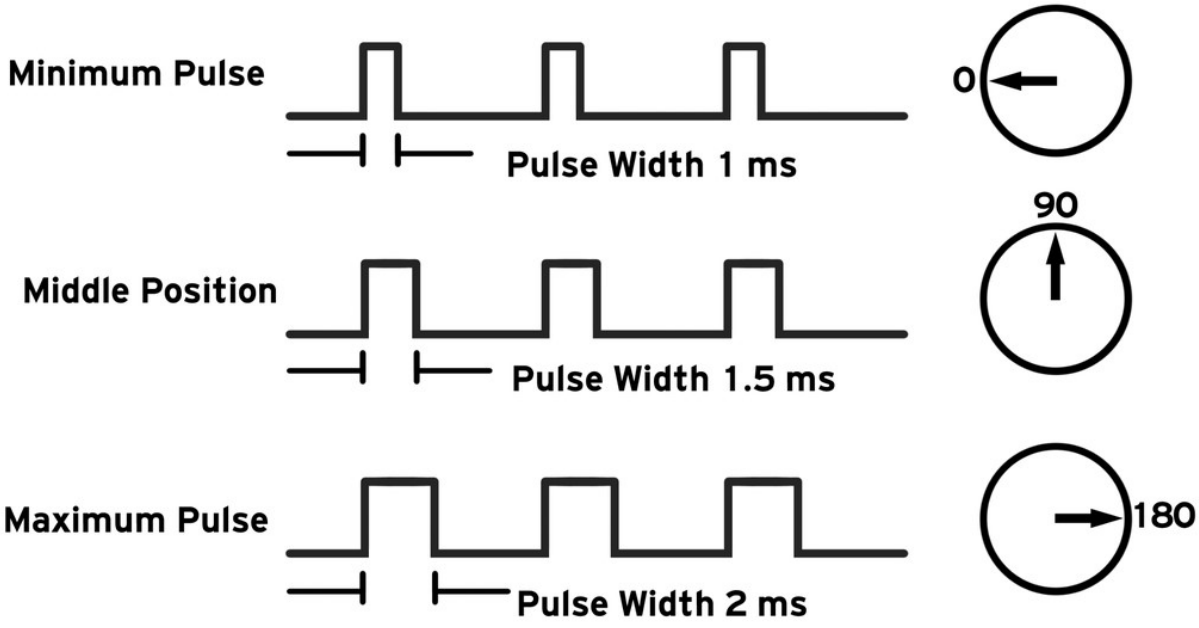


Figure 3. (Cycles of Servo Motors)

Controlling the exact pulse time:

It's a square wave, like PWM (12.). Each cycle of the signal lasts 20 milliseconds, and the value is usually LOW. The signal is HIGH for 1 to 2 milliseconds at the beginning of each cycle. It depicts 0 degrees at 1 millisecond and 180 degrees at 2 milliseconds. In the middle, it represents a value ranging from 0 to 180.

This is an excellent and reliable strategy. Important thing to remember is that utilizing the Servo library disables PWM capabilities at pins 9 and 10 on the Arduino UNO and related boards.

The code will simply declare the servo object and then we are able to initializes the servo by using the *servo.attach()* function. This part will be explained to more extend later.

.

Ultrasonic Sensor

The Ultrasonic Sensor emits high-frequency sound pulses and then measures the time it takes for the echo of the sound to return. On the front of the sensor, there are two apertures one for transmitting the signal (sound) and other for receiving. They are marked with letters T & R.

- Tiny speaker “T” to transmit opening ultrasonic waves;
- Microphone “R” to receive the ultrasonic waves;

Figure 4. (HC-SR04)

The ultrasonic sensor calculates distance by using bellow method:

In air, the speed of sound is approximately 341 meters per second. This information, coupled with the time delay between sending and receiving the sound pulse, is used by the ultrasonic sensor to calculate the distance from receiver to an object. The following mathematical formula has been used :

$$\text{Distance} = \frac{Time\ x\ Speed\ of\ Sound}{2}$$

Time = the amount of time between transmission of an ultrasonic wave and receiving it back. This number is divided by two since the sound wave must travel to and from the item.

Some things that are formed or positioned in such a way that the sound wave bounces off the object but is redirected away from the sensor may not be detected by an ultrasonic sensor.

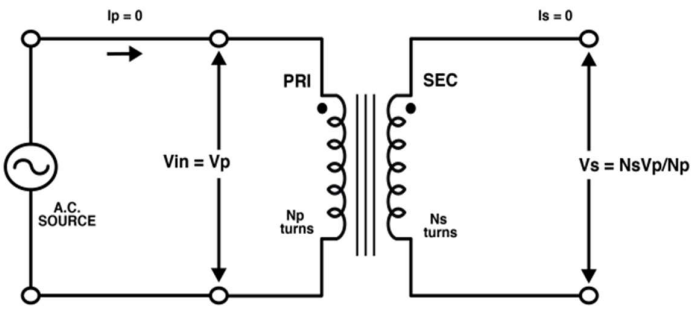
It is also possible for the object to be too small to reflect enough of the sound wave back to the sensor to be detected.

Objects can absorb the sound wave all together (cloth, carpeting, etc), which means that there is no way for the sensor to detect them accurately.

These are important factors to consider when designing and programming a robot using an ultrasonic sensor . But we can minimize those problem's impact with todays AI High Resolution Cameras.

Boost Converter

A Boost converter steps up a DC voltage from the input to the output. Virtually only way to literally increase voltage in any circuit is by using transformer. Since Op Amps usually require some sort of external power. Also for this project classic amplifier weren't available. Transformers can increase or decrease voltage,
by changing magnetic field between two windings (Figure 5.).



(Figure 5. Transformer Circuit)

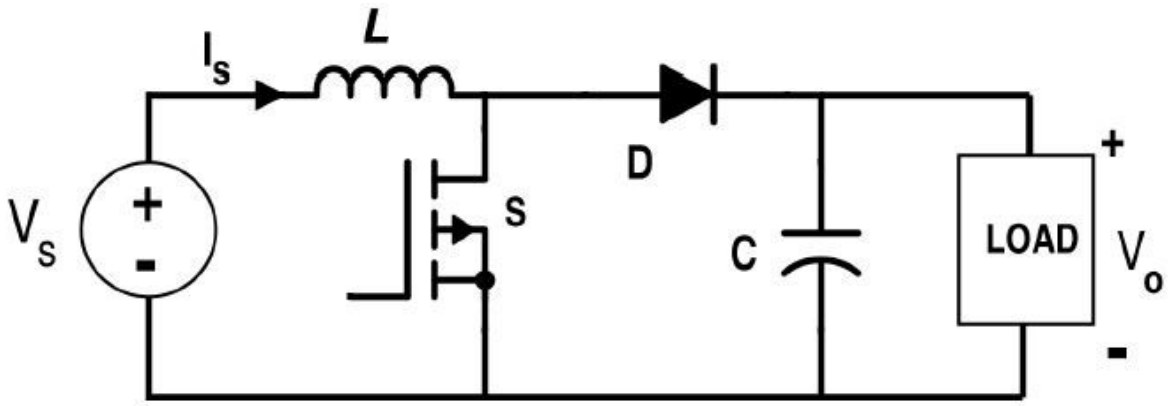
Transformers consist out of two windings, one with more turns and other with less. We call them primary and secondary winding. If we want for example to increase voltage, we will have less winding turns on primary and more on secondary. When we let AC trough an transformer the constant polarity change of current, will result in changes of the magnetic flux polarity inside the transformer. Those are the only important basics to understand working principle of boost converter.
Since only DC is available in projects circuit, we need somehow to produce the changes in magnetic flux inside the transformer.

We will do this other way around, by constantly switching current on and off. Transistor will do the job, but simple switch that someone is clicking will be sufficient, to understand working principle.

For real purpose use, it's best to use MOSFET or some sort of high frequency switching transistor. Then just by simply connecting capacitor to base of transistor, we will

get switching device. But for our circuit to work, we need other energy storing device, which is inductor in most cases.

The key principle that drives the boost converter is the tendency of an inductor to resist changes in current by either increasing or decreasing the energy stored in the inductor magnetic field. In a boost converter, the output voltage is always higher than the input voltage. Opposite to this is lower output current.



(Figure 6. Boost Converter Circuit)

Testing the components

The Arduino has two different types of input pins, those being analog and digital. This project will focus on the Digital Input Pins.

To check the working of our ultrasonic sensor

- The serial monitor allows the computer to connect serially with the Arduino;
- It takes data that the Arduino is receiving from sensors and other devices and displays it in real-time on the computer;
- Having this feature is extremely useful for debugging code and understanding the values that the device is receiving;
- *Navigate to File --> Examples --> 1.Basics --> AnalogReadSerial;*

- To use the serial monitor, which resembles a magnifying glass, press the button. The analog pin in the serial monitor is now reading the numbers. The numbers will increase and decrease as we turn the knob.;

The numbers will be in the range 0 to 1023. This is because the analog pin converts a value between 0 and 5V to a discreet number.

Digital inputs can serve as the foundation for a variety of digital communication protocols.

- By creating a 5V (HIGH) pulse or 0V (LOW) pulse, we can create a binary signal.
- This is important for interacting with digital sensors such as ultrasonic sensor, as well as interacting with other devices.
- To demonstrate the use of a digital input, connect a switch from digital pin 2 to 5V, a 10K resistor from digital pin 2 to ground, and run the following code:
- *File - Examples - 2.Digital - Button*
- When the switch is pressed, the electrical connections in the switch has less resistance than the resistor, and the electricity no longer connects to ground. Instead, electricity flows between 5V and the digital pin.
- This is because electricity always chooses the path of least resistance.

Shields

Shields are expansion adapter boards that sit on top of the Arduino Uno and provide it with additional functionality.
Because the Arduino is open source hardware, anyone with the desire can create an Arduino shield for whatever task they desire.
Hence, there are countless number of Arduino shields available, out of which the some of the official ones are –

- Wireless SD Shield
- Ethernet Shield
- Motor Shield

Motor Shield

The Arduino Motor Shield enables users to quickly control motor direction and speed using an Arduino. Because Arduino pins are universal, it is quite easy to connect any type of extension to Arduino. Motor tends to draw a lot of current when compared to other components, to protect them and power them, we have to use these sort of shields, because shields enables us to power a motor using a separate power supply of up to 12V.

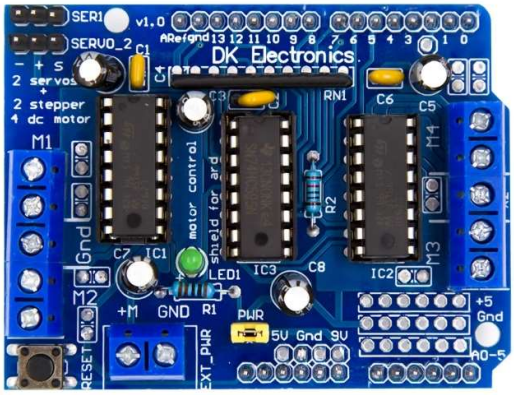


Figure 7. (L293D- Driver Module)

The motor shield that I'm going to use (L298N) has 2 channels, allowing the control of two DC motors, or one stepper motor.

An external power supply, the motor shield (L298N) can safely supply up to 12V and 2A per motor channel.
There are pins on the Arduino that are always in use by the shield. By addressing these pins it is possible to select a motor channel to initiate, specify the motor direction (polarity), set motor speed (PWM), stop and start the motor, and monitor the current absorption of each channel .

The pin breakdown is as follows:

Function	***Channel A***	***Channel B***
Motor Forward	Digital 7	Digital 4
Motor Backward	Digital 6	Digital 5
Trigger	Analog 1	Analog 1
Echo	Analog 0	Analog 0

PWM

When we supply direct current to a DC motor, the magnetic field inducted inside the motor, within the magnetic field produced with permanent magnets is going to produce rotational force on the shaft, which makes the motor spin. If we supply higher voltage to motor the motor is going to spin faster. This is great, but it is incredibly inconvenient for practical uses. The motor becomes unpredictable and hard to control, especially with microcontroller. We need some way to convert simple analog signal to digital pulse.

The way and a far better method of controlling DC motors is to use pulse width modulation or PWM. PWM sends a series of pulses to the motor. Each pulse is of the maximum voltage that the motor can handle, so a 6V motor will receive 6V pulses and a 12V motor will receive 12V pulses. The width of the pulses is varied to control the motor speed; narrow pulses cause the motor to spin slowly. Increasing the pulse width will increase the motor's speed.

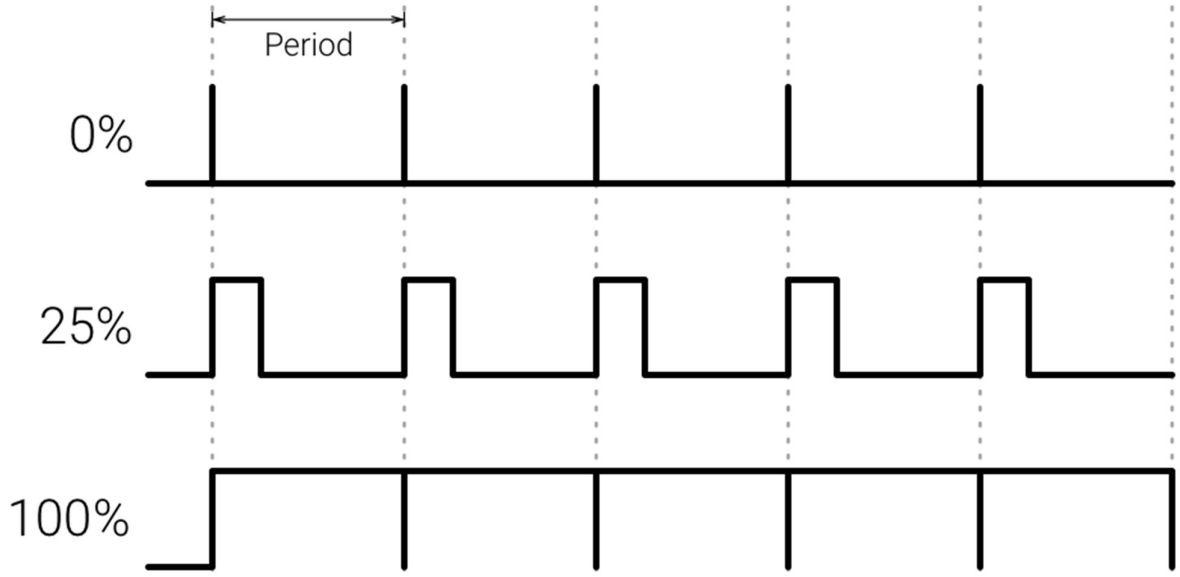


Figure 8. (PWM Graph)

The amplitude of this pulse is our voltage and the period od the pulse is what we call a Duty Cycle. Explanation above gives us ability to convert various analog signals (ex. Sinusoid) to digitally readable signal type (pulse). With higher frequency pulses we will get the same effect, as higher voltage in analog signals. To calculate PWM output voltage, we use following formula: $\frac{Duty}{256} \times 5V$

Procedure/Implementation

Getting the hardware ready

Note: For the full building process, please check the documentation provided along with the report.

- Considering the scenario that everything is already prebuild;
- *S*witch the prototype vehicle on and providing power to L298N Shield;
- Connecting Arduino Uno to Computer and programming it;
- The USB cable powers the device, but Arduinos can *also* run standalone by using an external power supply, which are batteries in this case;

Implementation Problems

After connecting everything and programming the Arduino, some problems were recognized immediately.

First of being the default position of servo motor, which is attached to Ultrasonic Sensor. This part can be fixed with other code, so there is no need to do anything, hardware wise.

After, I noticed major problem which consists out of few parts, which are going to be split into following categories:

Powertrain of the vehicle isn’t capable of moving the mentioned, regardless the motors being used are geared. Some pragmatic changes are needed to be done.

Lowering the weight of the vehicle. I had done this by cutting the unnecessary parts of bodywork (sharp angles and cut parts on Figure 9.), using lighter plastics and smaller front wheels. But this doesn't come without compromises. The cardboard frame needed to be strengthen, which has been done with very thin plastic parts (hatched part on Figure 9.).

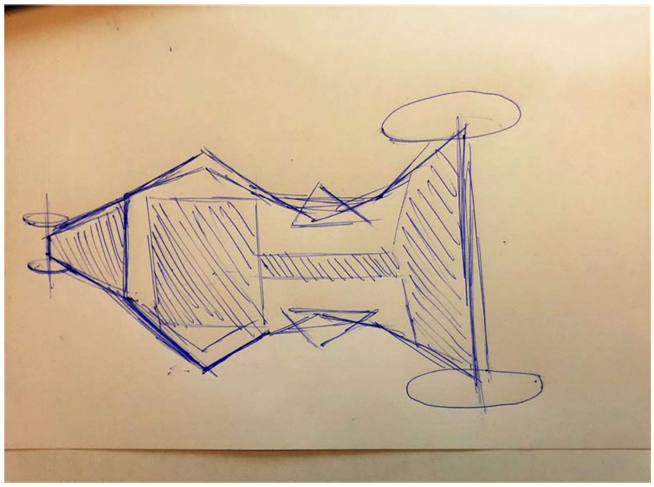


Figure 9. (Improved Bodywork Sketch)

Adding one more power source to increase power ratings, as well as capacity of the batteries. Two 9v batteries were grounded and one is connected directly to Arduino for powering Servo motor with Ultrasonic sensor and other is connected to L298N Shield, just to power two motors. Important mention here is that, two batteries need to be almost identical, since they are alkaline. If batteries have different capacity or power ratings they will try to charge each other to get into equilibrium state and since they are not rechargeable, this will destroy the batteries.

After changing the batteries the vehicle started working correctly, but only when going straight, the wheels slip out when making a turn (ex. Right Wheel Forward, Left Wheel Backward) or similar situations. This effect is direct consequence of lack of wheels grip. Solution was to type few small pieces of dual-sided tape on rear wheels or to try the vehicle on carpet. This solved the grip problem, but also introduced much more complicated one.

The motors don't have enough torque to turn the wheel at all, when making turn. Only solution for this problem is voltage increase, since lowering weight isn't possible at all at this point. 12V batteries (Figure 10.) are incredibly hard to find and they tend to have very low capacity, because they are mostly used in remote controllers . At this point, decision has been made. To solve voltage problem, only solution is to build boost converter (Figure 6. {page 10}).

Figure 10. (12V Batteries)

Getting the software ready

- Download and install the our microcontroller IDE (integrated development environment). Most of microcontroller use some variation of the C and C++ programming languages.
- Setting the board-type and serial port.
- Compiling and testing the code.

Writing the code

1. Including all the necessary libraries, required for this project.

```
#include <Servo.h>          //Servo motor library.
#include <NewPing.h>        //Ultrasonic sensor function library.
```

2. Next, setting the output pins and defining motors and sensors.

```
// L298N control pins
const int LMF = 7; //Left Motor Forward
const int LMB = 6; //Left Motor Backward
const int RMF = 4; //Right Motor Forward
const int RMB = 5; // Right Motor Backward


//sensor pins
#define trig_pin A1 //Analog input 1
```

```
#define echo_pin A2 //Analog input 2

#define maximum_distance 200
boolean goesForward = false;
int distance = 100;

NewPing sonar(trig_pin, echo_pin, maximum_distance); //Sensor Function
Servo ser1; //Servo Motor Name
```

3. Then, writing the code using previously defined variables, with functions, setup, loop, etc…

```
  void setup(){

  pinMode(RMF, OUTPUT);
  pinMode(LMF, OUTPUT);
  pinMode(LMB, OUTPUT);
  pinMode(RMB, OUTPUT);

  ser1.attach(10); //our servo pin

  ser1.write(115);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
}

void loop(){

  int distanceRight = 0;
  int distanceLeft = 0;
  delay(50);

  if (distance <= 20){
    moveStop();
```

```
    delay(300);
    moveBackward();
    delay(400);
    moveStop();
    delay(300);
    distanceRight = lookRight();
    delay(300);
    distanceLeft = lookLeft();
    delay(300);

    if (distance >= distanceLeft){
      turnRight();
      moveStop();
    }
    else{
      turnLeft();
      moveStop();
    }
  }
  else{
    moveForward();
  }
    distance = readPing();
}

int lookRight(){
  ser1.write(50);
  delay(500);
  int distance = readPing();
  delay(100);
  ser1.write(115);
  return distance;
}

int lookLeft(){
  ser1.write(170);
  delay(500);
  int distance = readPing();
  delay(100);
  ser1.write(115);
  return distance;
  delay(100);
}

int readPing(){
  delay(70);
```

```
  int cm = sonar.ping_cm();
  if (cm==0){
    cm=250;
  }
  return cm;
}

void moveStop(){

  digitalWrite(RMF, LOW);
  digitalWrite(LMF, LOW);
  digitalWrite(RMB, LOW);
  digitalWrite(LMB, LOW);
}

void moveForward(){

  if(!goesForward){

    goesForward=true;

    digitalWrite(LMF, HIGH);
    digitalWrite(RMF, HIGH);

    digitalWrite(LMB, LOW);
    digitalWrite(RMB, LOW);
  }
}

void moveBackward(){

  goesForward=false;

  digitalWrite(LMB, HIGH);
  digitalWrite(RMB, HIGH);

  digitalWrite(LMF, LOW);
  digitalWrite(RMF, LOW);

}

void turnRight(){

  digitalWrite(LMF, HIGH);
  digitalWrite(RMB, HIGH);
```

```
  digitalWrite(LMB, LOW);
  digitalWrite(RMF, LOW);

  delay(500);

  digitalWrite(LMF, HIGH);
  digitalWrite(RMF, HIGH);

  digitalWrite(LMB, LOW);
  digitalWrite(RMB, LOW);



}

void turnLeft(){

  digitalWrite(LMB, HIGH);
  digitalWrite(RMF, HIGH);

  digitalWrite(LMF, LOW);
  digitalWrite(RMB, LOW);

  delay(500);

  digitalWrite(LMF, HIGH);
  digitalWrite(RMF, HIGH);

  digitalWrite(LMB, LOW);
  digitalWrite(RMB, LOW);
}
```

Schematic

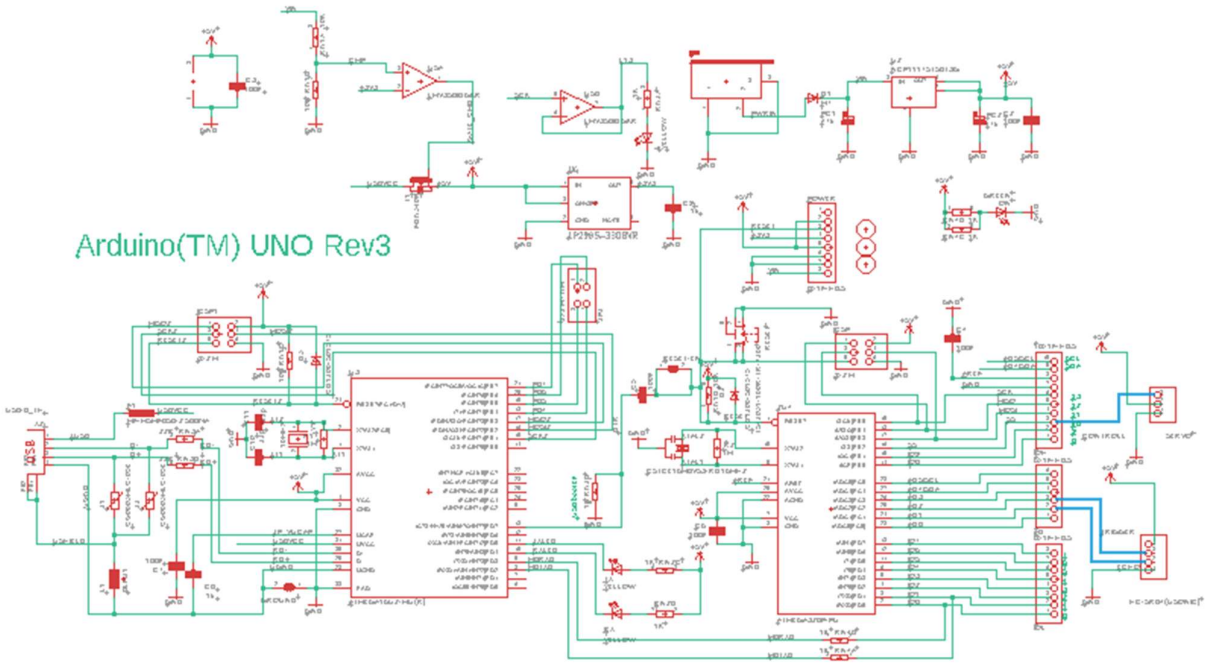


Figure 11. (EAGLE Design of the Circuit)

Above displayed circuit represents the internals of ATMEGA328P-PU microcontroller, which is used inside Arduino Uno board. The circuit currently at this stage lacks L298N driver shield, which is going to be added later on. The 3 and 4 pin headers on the right represent Servo Motor and Ultrasonic Sensor, this part is going to be explained in detail, within Figure 12.

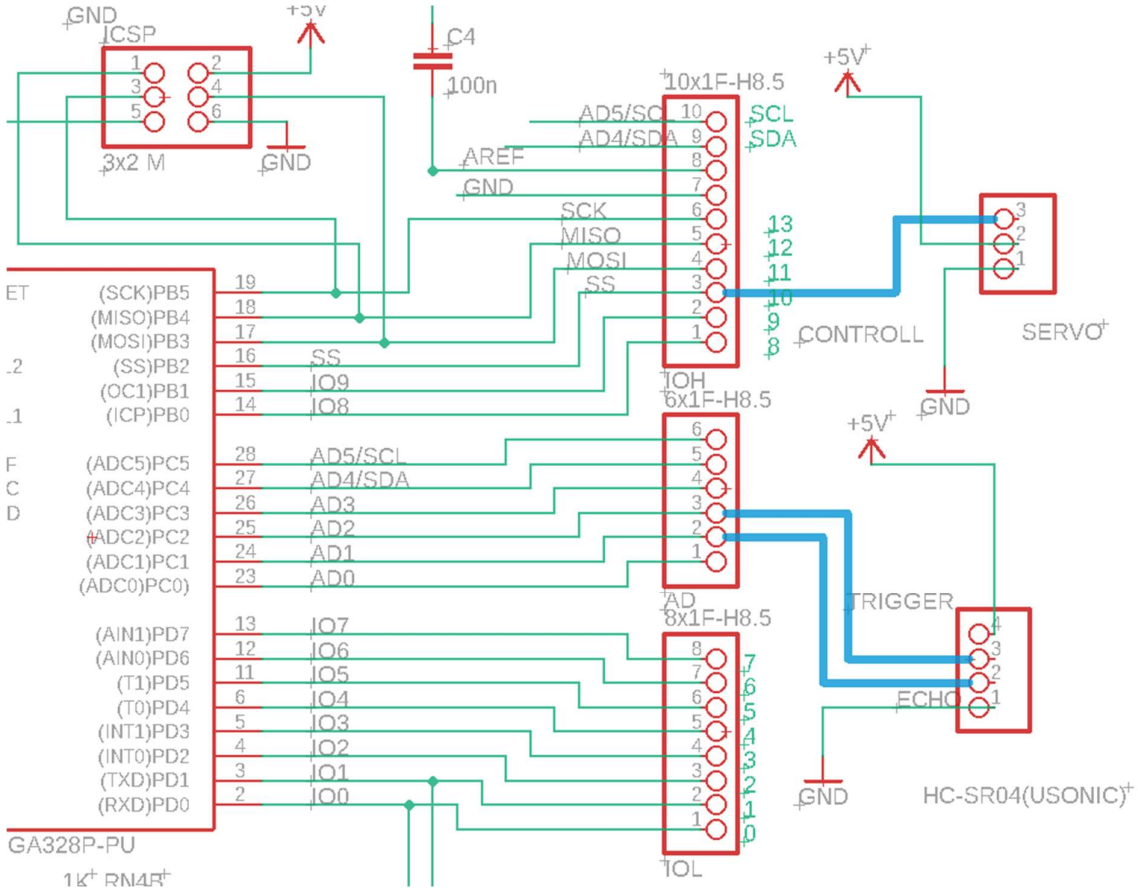


Figure 12. (Closer Look at Servo and HC-SR04 Connections)

As seen on Figure 12, on the right side, HC-SR04 Ultrasonic Sensor and Servo Motor are situated. The BUS connection represents the simple jumper wire. It is noted that Ultrasonic sensor has 4 pins, which from pins 1 and 4 are power pins (1- Ground, 4- VCC). Pins 3 and 2 are connected to Arduino's analog input (A1 and A2). This pins represent Trigger and Echo functions. The names of the pins itself are self-explanatory. Next, going to Servo Motor, it can be seen that Servo has only one Control pin, which has been connected to Arduino's digital pin 10. The rest are just for power (Servo Pin 1 and 3). Now, for the next figure, we are going to neglect the Servo Motor and Ultrasonic sensor, to show connection of L298N Driver Shield.

Note: The L298N Driver wasn't available, so L298 has been used, but principle remains the same. The pinout is almost identical, only difference are "Enable" pins, but they are not important, since they should be shorted anyways.

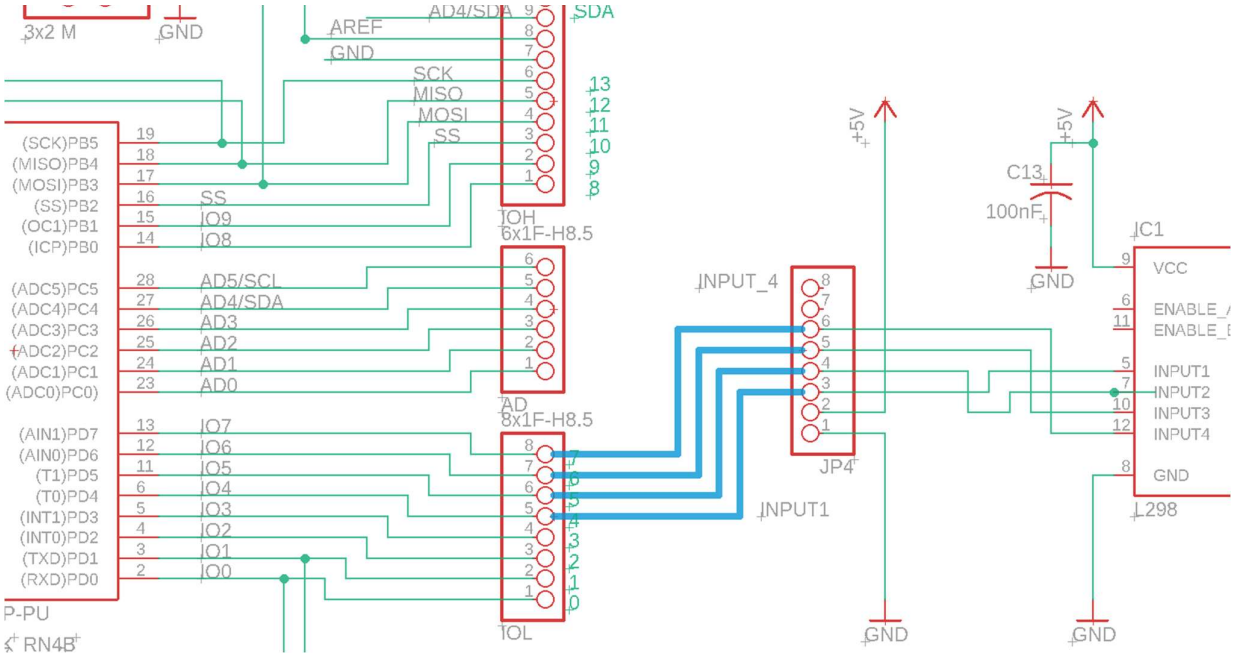


Figure 13. (L298N Driver Connection)

Here, we connected Input 1 pin on the Driver to Arduino's digital pin 4, Input 2 to digital pin 5, and so on… This step is crucial since, we need to define those pins in our code. If we switch pins, the motors aren't going to spin properly. For example, when signal is let thought digital pins 4 and 5 in manners that Pin4=HIGH and Pin5=Low, the driver is going to detect the signal on Input 1 and 2 and let voltage to our DC motor, but in what polarity? The motor is going to spin forward, and vice versa. If both Input's receive same signal (HIGH or LOW) the motor is going to stop.

Now, only thing left to do is connect our DC Geared Motors to Output of the Driver and viola, our smart Vehicle is working.

Working Principle

- The vehicle is switched on by giving it 9V DC power from an external battery. I decided to put Arduino's and Driver's PS on different switches.
- The motors will start to rotate and thus the robot starts moving forward.
- During this time, the ultrasonic sensors continuously keeps calculating the distance between the vehicle and the reflective surface and it does this in full 180°, because of the servo motor that's constantly moving.
- This information gets processed by the ARDUINO.
- If the distance between the vehicle and the obstacle is less than the specified value, the vehicle changes its path or moves backwards...
- This process continues until the power is switched off.

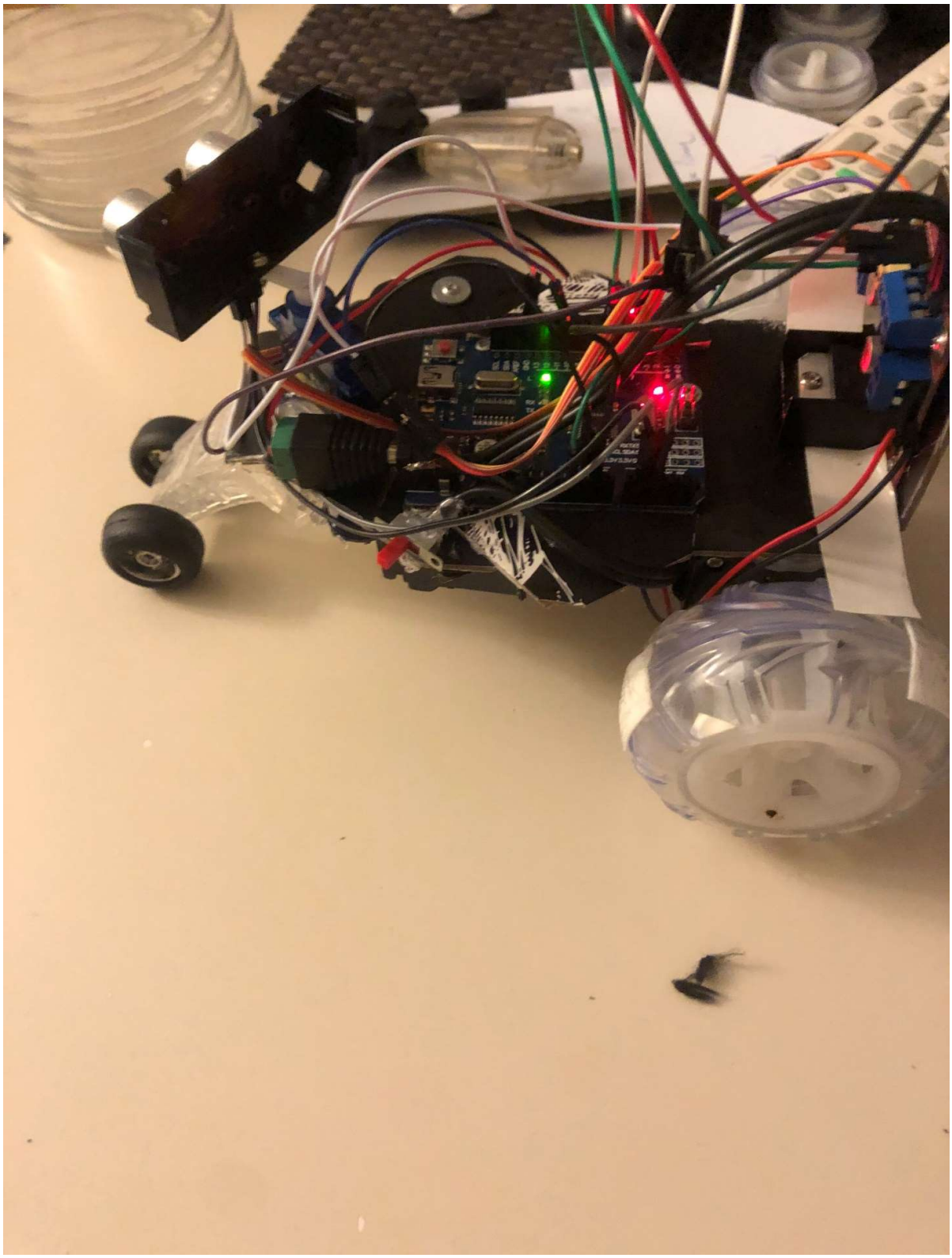


Figure 14. (Vehicle Prototype)

DC Motor Analysis

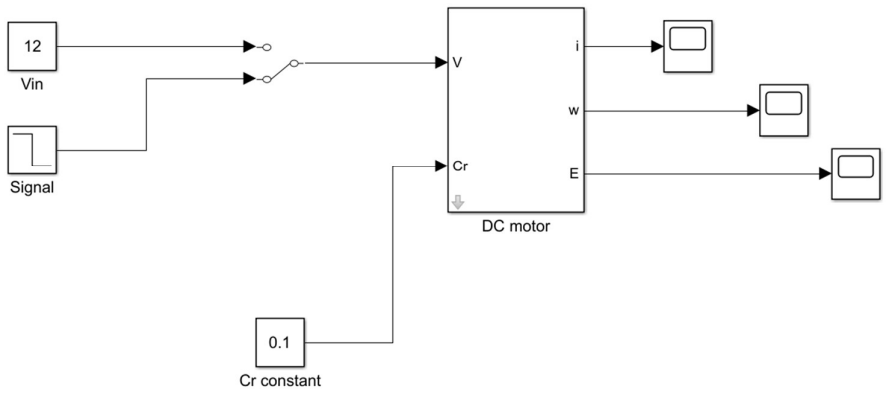


Figure 15. (Simulink model of the DC motor)

Figure 15. is rough representation of DC motor, that has been made in Simulink. All of the values have been considered and Figure 17 is Step Response.

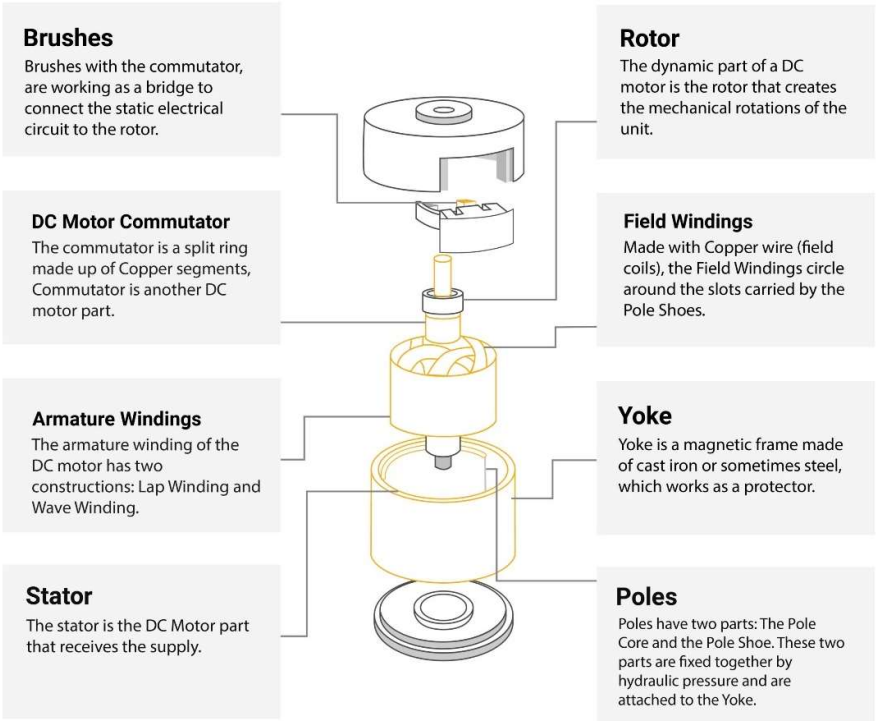
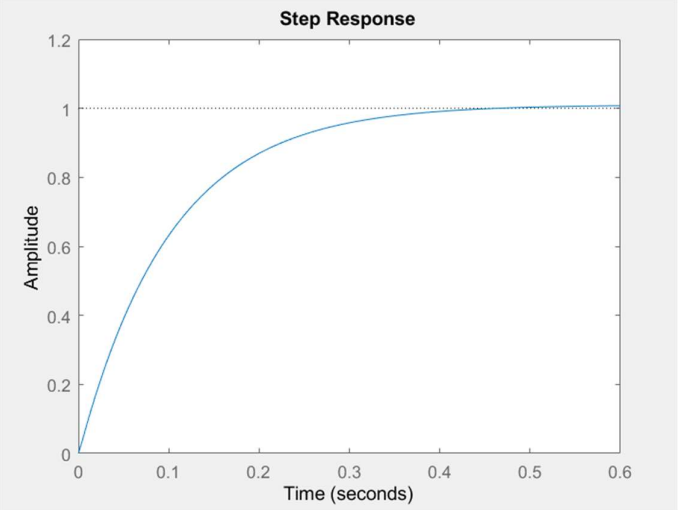


Figure 16. (Construction of DC motor)



R = 5.5; [Ohm]

L = 0.0028; [H]

phi = 0.5; [Wb]

k = 1; []

J = 0.5; [kg m^2]

Cr = 0.1; [Nm] Cr constant

V = 12; [V]

ans =

{0.2000

21.8000}

Figure 17. (Step Response of DC motor)

Figures displayed above represent motors figurative specifications, in following manners:

R- Internal resistance of Armature;

L- Inductance of our windings;

Phi (Ø) - Flux produced per pole in Wb (weber);

k- electromotive force constant (usually represented as [Ke], with values of around 0.01 V/rad/sec ;

J- Moment of inertia of the rotor;

Cr- Motor torque constant;

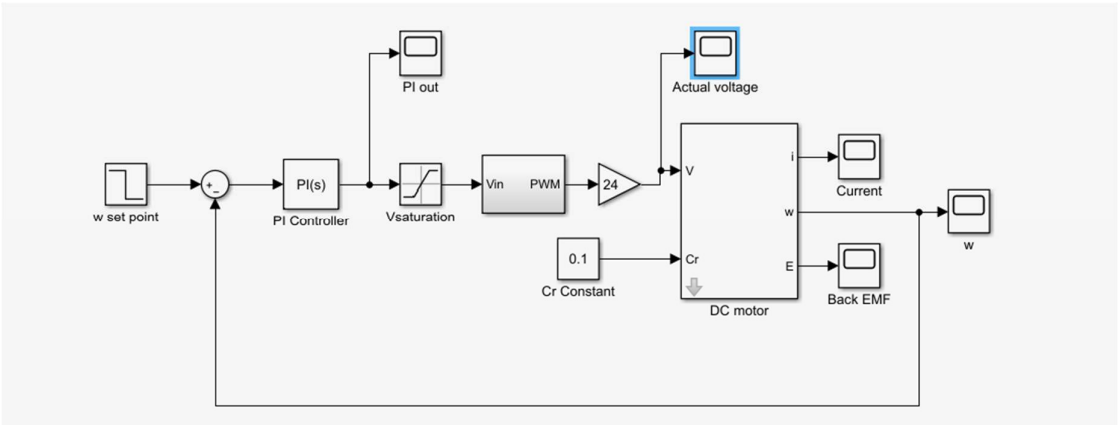


Figure 18. (Simulink Model of a PWM Control through a PI Regulator)

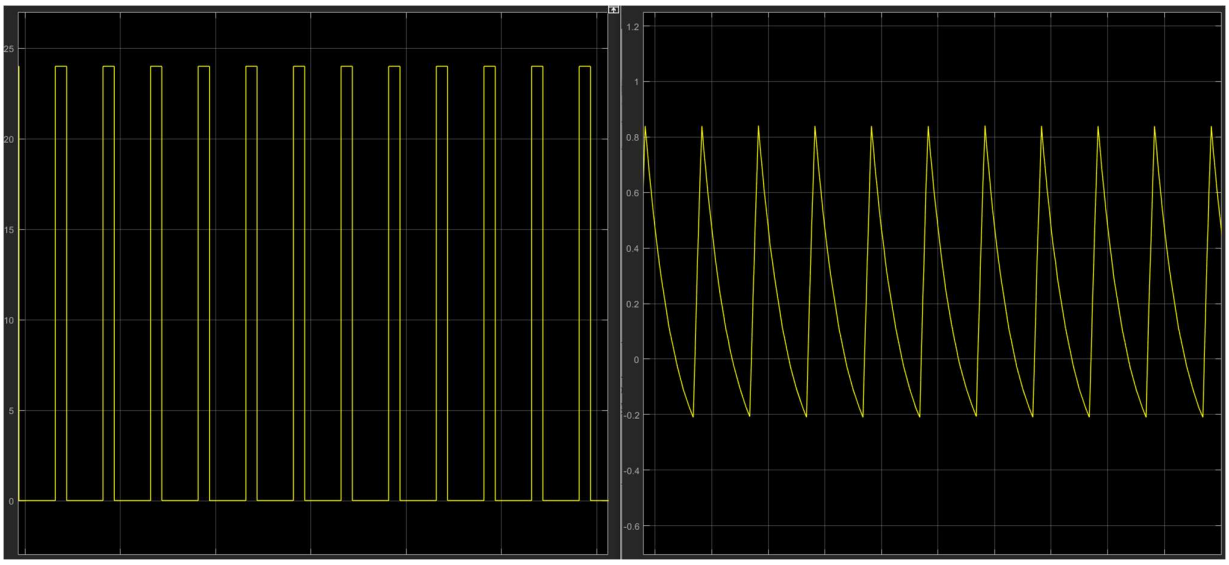


Figure 19. (Actual Voltage- left and Current-right)

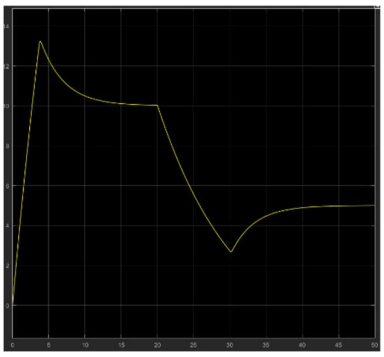


Figure 20. (Angular Velocity)

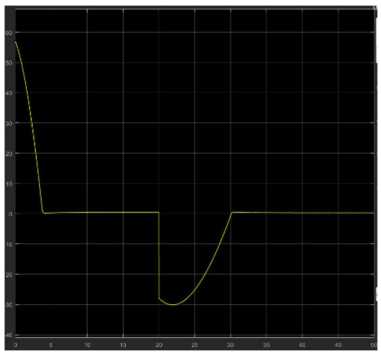


Figure 21. (PI out)

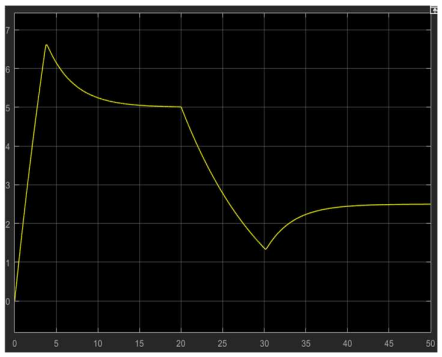


Figure 22. (Back EMF)

Applications

- This vehicle can be used for avoiding concealed paths, such examples are industrial robots that avoid workers inside the factory.
- The modified code with same logic is used in smart Vacuum cleaners.
- They have great importance in scientific exploration and emergency rescue, especially for the places that aren't reachable or dangerous for people.
- Ultrasonic Sensor's are useful in car's parking system.
- It can also be used in terms of assembling automobiles, as well as many other industries.
- The system used in this prototype can be very useful for people with special medical needs, as disability. It is possible to put such system on wheelchair's or crutches, so that they can receive some sort of signal, when obstacle is detected. For ex.: noise, vibration, full stop, etc…

Further implementations –

- This technique can also be used as a aid device for blind people. For example, blind people can wear a device that has Ultrasonic sensor attached to it (ex. poles for blind, belts, bracelet), or some other sensor, whose output of vary in according to the object position changes. Then they could receive a vibration on specific side, that will warn them of obstacle. This technique enables blind people to navigate easily .
- On top of simple function, like obstacle avoiding. It is possible to make robots, with same principle and add temperature, pressure, humidity sensors to monitor the atmospheric conditions around it. This is useful for environments not suitable for humans.
- Same technology can be improved and used to make fully usable autonomous drive street car, with possibility of line following.
- This technology can be used to build services robots, for purposes of household/restaurant work and so many other indoor applications.

References

1. PCB design of L298N Driver Shield- caternuson-L298-Motor-Driver-PCB, https://github.com/caternuson/L298-Motor-Driver-PCB

2. T. Lozano-Perez, “Autonomous robot vehicles”, Eds. Ingemar J. Cox, and Gordon T. Wilfong, Springer Science & Business Media, 2012.

3. G. Song, Y. Kaijian, Z. Yaoxin, and C. Xiuzhen, "A surveillance robot with hopping capabilities for home security", IEEE Transactions on Consumer Electronics 55.4, 2009.

4. H. Durrant-Whyte, and T. Bailey, "Simultaneous localization and mapping: part I.", IEEE robotics & automation magazine 13.2, 2006, pp. 99-110.

5. R.O. Duda, P.E. Hart, and D.G. Stork, “Pattern classification”, John Wiley & Sons, 2012.

6. Figure [16.]- Linquip Web Site, https://www.linquip.com/blog/dc-motor-parts/

7. IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, 2016, pp. 189-194.

8. Figures [1.], [2.], [4.], [7.]- are open source, obtained from official Arduino website, https://www.linquip.com/blog/dc-motor-parts/.

9. Arduino Datasheet/PCB design- Arduino wesite, https://content.arduino.cc/assets/Pinout-UNOrev3_latest.png