Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology

– CO1006 Introduction to Computing – Spring 2018 –

ARDUINO CAR

Ngo Duc Tuan - 1710364
Nguyen Duc Khoi - 1752302
Nguyen Tien Anh - 1752076
Ngo Nguyen Thuan - 1752525

May 31, 2018

Contents

List of Figures

List of Tables

Acknowledgement

First of all, we would like to express our special thanks and gratefulness to our lecturer Dr. Le Trong Nhan and our teacher assistant Mr. Truc D.T. Nguyen who have given us many useful advices and motivations through the process of completion of our project.

1 Introduction

In the course Introduction to Computing, students can study about Arduino. It is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs and turn it into an output. We can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so we use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

To apply the learned knowledge about Arduino, we decided to conduct a project about a controlled car. When we start our project, we set the target that the car can go backward, forward, turn left, and right with the controllable speed by using a software on a smartphone via the Bluetooth connection. Moreover, the car is also able to identify obstacles which oppose it then alter its path. We have used board Arduino UNO R3, ultrasonic sensor HC-SR04, ,module Bluetooth HC-06, and L29N Shield DC motor driver as the fundamental components of this car.

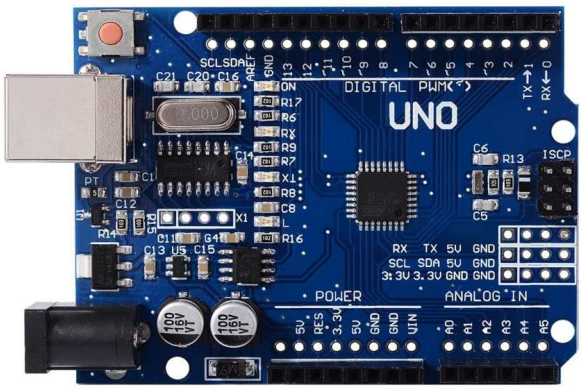


Figure 1: Arduino Uno R3

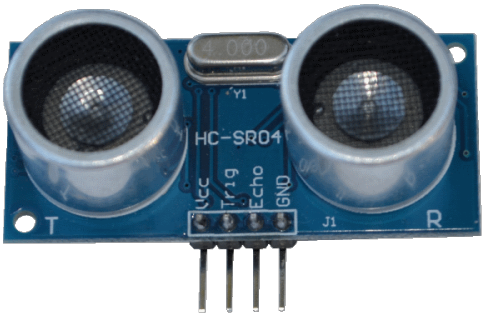


Figure 2: Ultrasonic sensor HC-SR04

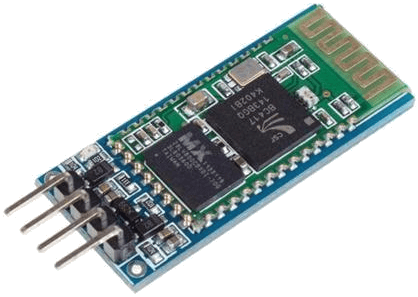


Figure 3: Module bluetooth HC-06

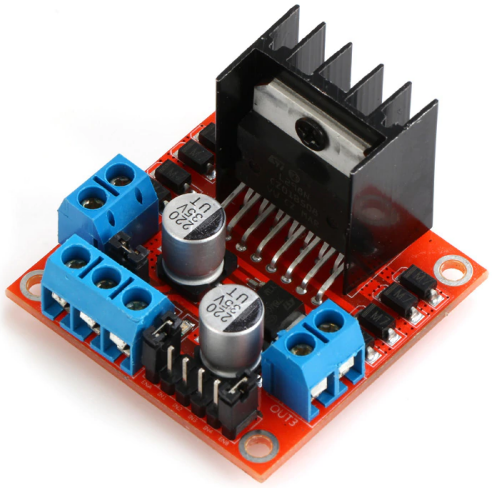


Figure 4: Motor shield L298N

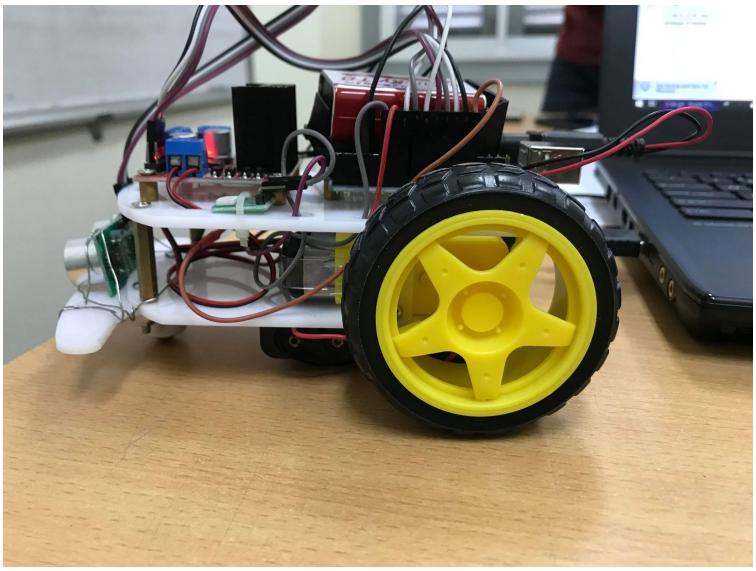


Figure 5: Real assembly 1

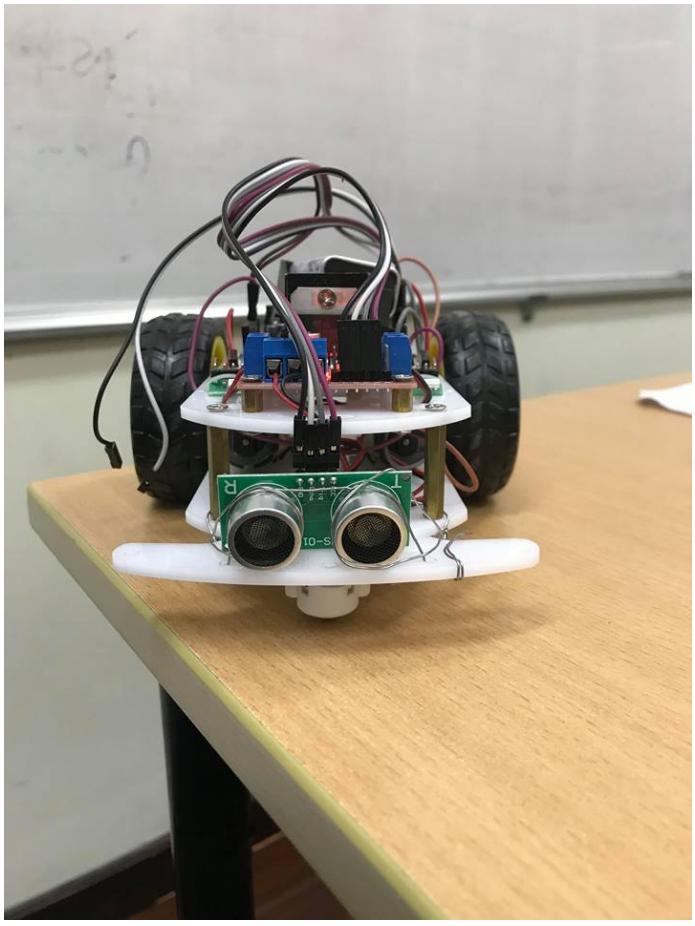


Figure 6: Real assembly 2

2 Methods

2.1 Schematic diagram and connection list

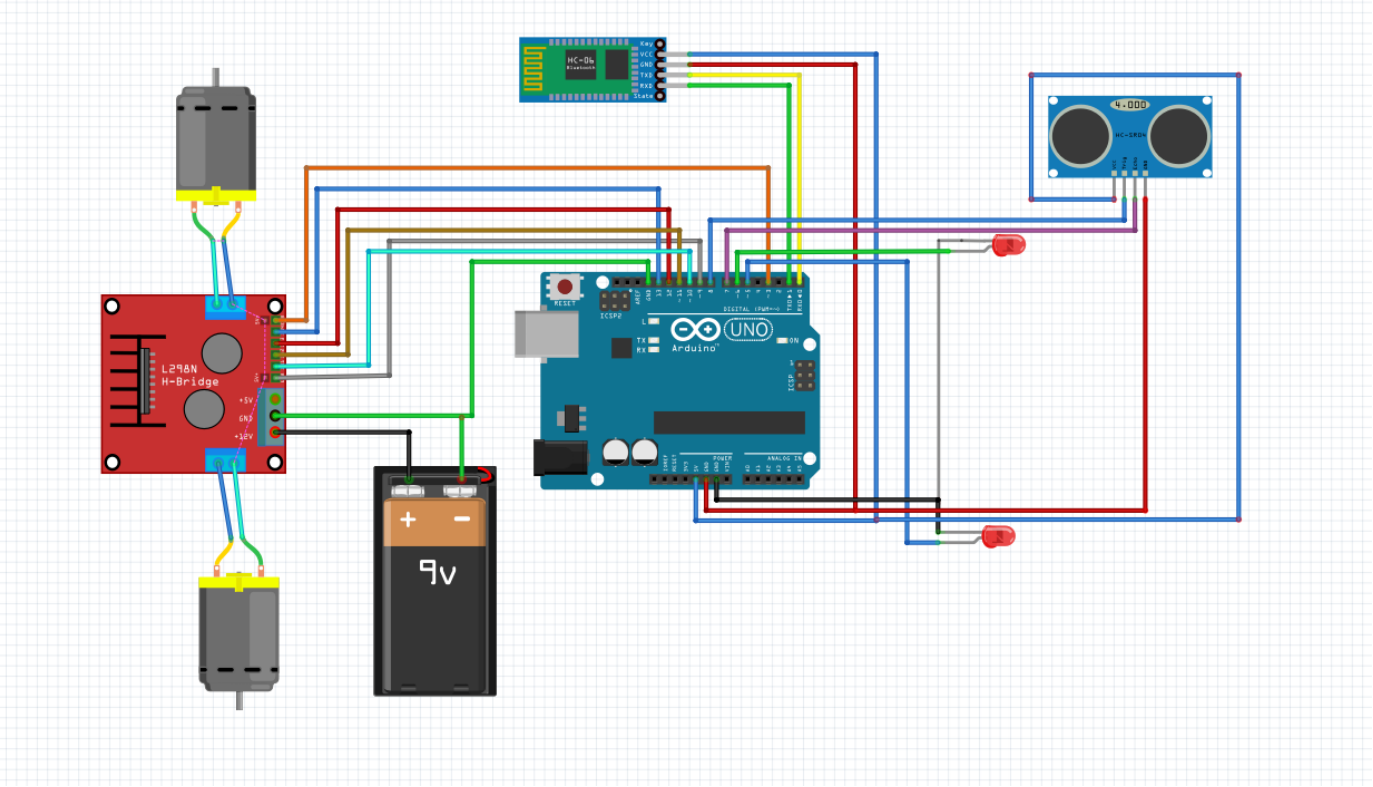


Figure 7: Schematic diagram of all components

Uno R3	L298N
GND	GND
IN1	13
IN2	12
IN3	11
IN4	10
ENA	9
ENB	3

Table 1: Connection betwen Uno R3 and L298N

Uno R3	HC-SR04
GND	GND
7	Echo
8	Trig
5V	VCC

Table 2: Connection betwen Uno R3 and HC-SR04

Uno R3	HC-06
GND	GND
TX→	RXD
RX←	TXD
5V	VCC

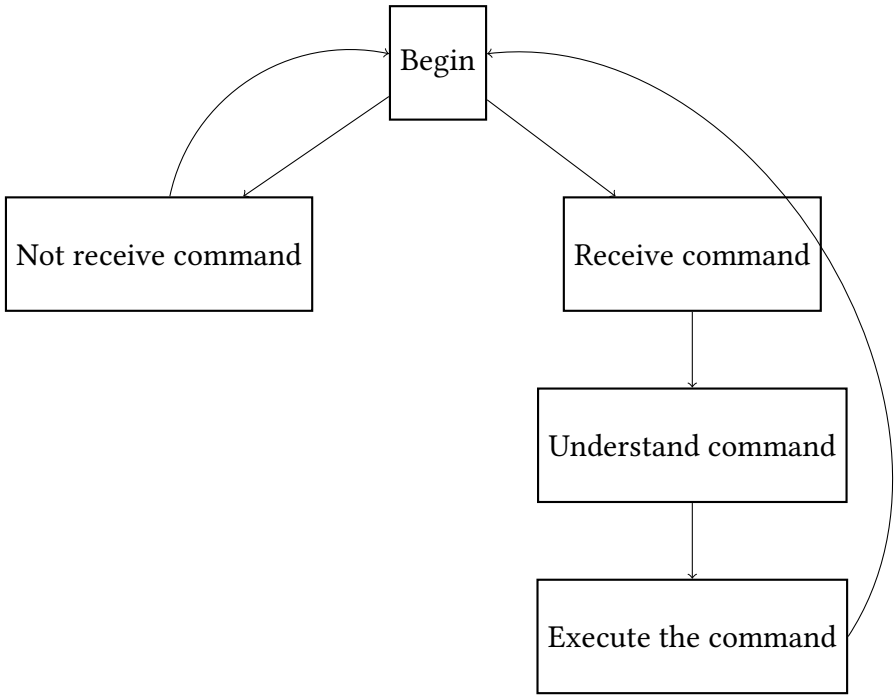
Table 3: Connection betwen Uno R3 and HC-06

2.2 Code structure

As mentioned above, an Arduino program contains 2 important parts, which are void setup and void loop. Void setup() is a part that contains the initialization of every pin for input or output of an Arduino project and is executed once for an entire program. While void loop() is executed repeatedly throughout the program and it contains the main part of the code. Some extended functions can be written for doing more specific tasks.
The main problem here is to use these two functions as efficient as possible to make the robot works.
These following steps is applicable for programming a robot that can follow a particular command.

$$\text{1.Receiving} \longrightarrow \text{2.Understanding} \longrightarrow \text{3.Executing}$$

In order for the robot to receive and follow the commands continuously, these steps are efficiently put in the loop function. Since the loop function keeps repeating, the robot is always ready for receiving new command and execute it immediately. Figure 1.x illustrates how the procedure works.



In this way, we are able to control the robot directly without any delay to transfer commands into arduino 's memory. The procedure also enables the robot to run exactly one single command for each loop. This is the key feature of this program technique and will be discussed later on other sections.
In the next few sections, we will introduce some appropriate pseudo code to finish each steps and explain why these pseudo code work correctly in the above structure.

2.3 Void loop() and idea of the main code

Recall that all the code written in the loop() function is executed over and over again once the program is still running. The huge number of loops per seconds enables the users to actively control the Adruino board.

With an attempt to enable the program to recognize different commands, the following programming style can be used.

```
Void loop()
Check for command.

if (receiving command 1)
do task 1;
else if (receiving command 2)
do task 2;
...
```

In the beginning of a loop, the compiler will check whether a command is sent to Arduino. If the Arduino receives a command, it will consider the command as a condition, then the compiler will check which case is satisfied and execute it. Now another loop begins and the Arduino is ready for another command.
This programming style is simple and easy to understand. However, what if the number of commands became very large and complex? If statement will be replaced by switch statement since switch statement is more readable and manageable than if statement. Moreover switch statement works faster because it has more efficient lookup method compared with if statement.

2.4 Void Runmotor (int a, int b, int c)

In this section, we will discuss the runmotor function which is directly used to control the car to move forward, backward, turn left and turn right with different speeds.
The function uses 3 different variables which are a, b and c. Variable a indicates which motor is considered (a=2 for left motor and a=1 for right one). Variable b indicates the direction of the motor (b=1 for forward, b=-1 for backward and b=0 for stop). Variable c determines the speed of rotation.

```
void runmotor(int a,int b, int c)
{
if(a==1&&b==1)
{
digitalWrite(inB,HIGH);
digitalWrite(inA,LOW);
analogWrite(v1,c);
}
if(a==2&&b==1)
{
digitalWrite(inD,HIGH);
digitalWrite(inC,LOW);
analogWrite(v2,c);

}
if(a==1&&b==-1)
{
digitalWrite(inB,LOW);
digitalWrite(inA,HIGH);
analogWrite(v1,c);
}
if(a==2&&b==-1)
{
digitalWrite(inD,LOW);
```

```
        digitalWrite(inC,HIGH);
        analogWrite(v2,c);
        }
        if(a==1&&b==0)
        {
        digitalWrite(inA, LOW);
        digitalWrite(inB, LOW);
        }
        if(a==2&&b==0)
        {
        digitalWrite(inC, LOW);
        digitalWrite(inD, LOW);
        }
        }
```

Consider the situation where a=1 and b=1. inB and inA is two pins which decide the direction of the motor 1. Since b=1, the motor rotates forward, pin inB is set HIGH and pin inA is set LOW. To control the speed, the motors are connected to an analog pin which can change it electric potential values. Different electric potential gives different signal to the motors.

The robot will move straight if two motor rotate in the same direction with the same speed. The robot will turn left when motor right rotates faster than motor left in the same direction.

2.5 Bluetooth connection

Bluetooth HC-06 enables users to control Arduino via Bluetooth connection.// Pins
VCC and GND: POWER
TXD: Serial output, connected to RX of the Arduino
RXD: Serial input, connected to TX of the Arduino

Working principle
On smartphone screen, when a button is pressed, a signal correspond to that button is sent to Arduino as a Serial signal. At the time the button is released, another signal is sent to Arduino. This means that the robot only receives signal at the moment the button is pressed and released and no signal is sent while the button is kept pressing.

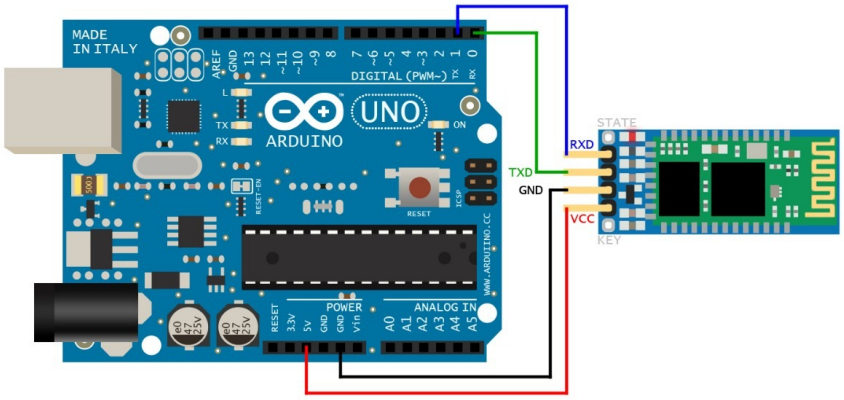


Figure 8: Schematic diagram of module bluetooth HC-06

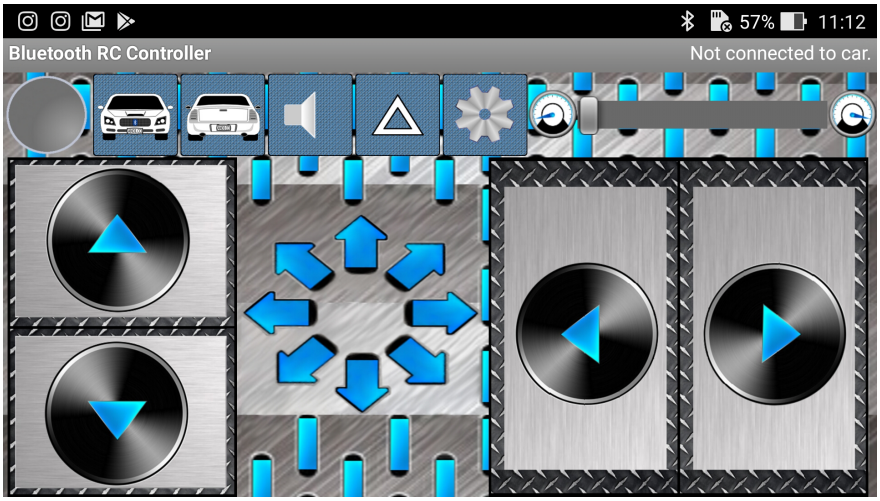


Figure 9: Control car by Bluetooth App on Android

2.6 Whole code

```
int inA=13;
int inB=12;
int inC=11;
int inD=10;
int ledR=6;
int ledL=5;
int v1=9; // analog pin of motor 1
int v2=3; // analog pin of motor 2
int s1=255; //electric potential values for high speed
int s2=130; //electric potential values for low speed
int signL=0; //indicate left led
int signR=0; //indicate right led
char c; //value of read command


void setup() {
// put your setup code here, to run once:
pinMode(inA, OUTPUT);
pinMode(inB, OUTPUT);
pinMode(inC, OUTPUT);
pinMode(inD, OUTPUT);
pinMode(v1, OUTPUT);
pinMode(v2, OUTPUT);
Serial.begin(9600);
pinMode(LED_BUILTIN, OUTPUT);
pinMode(ledR, OUTPUT);
pinMode(ledL, OUTPUT);
}

void loop() {
// put your main code here, to run repeatedly:
if(Serial.available()>0)
{
c=Serial.read();
switch(c)
{
case 'F': //forward
runmotor(1,1,s1);
runmotor(2,1,s1);
signL=0;
signR=0;
break;
case 'B': //backward
```

```
        runmotor(1,-1,s1);
        runmotor(2,-1,s1);
        signL=0;
        signR=0;
        break;
        case 'L': //turn left
        runmotor(1,1,s1);
        runmotor(2,0,s1);
        signL=1;
        break;
        case 'R': //turn right
        runmotor(1,0,s1);
        runmotor(2,1,s1);
        signR=1;
        break;
        case 'G': //turn left while moving forward
        runmotor(1,1,s1);
        runmotor(2,1,s2);
        signL=1;
        break;
        case 'I': //turn right while moving forward
        runmotor(1,1,s2);
        runmotor(2,1,s1);
        signR=1;
        break;
        case 'H': //turn right while moving backward
        runmotor(1,-1,s1);
        runmotor(2,-1,s2);
        signL=1;
        break;
        case 'J': //turn left while moving backward
        runmotor(1,-1,s2);
        runmotor(2,-1,s1);
        signR=1;
        break;
        default:
        runmotor(1,0,s1);
        runmotor(2,0,s1);
        signL=0;
        signR=0;
        }
        }
        if(signL==1) //left led blinks
        {
        digitalWrite(ledL, HIGH);
```

```
        delay(100);
        digitalWrite(ledL, LOW);
        delay(100);
        }
        if(signR==1) //right led blinks
        {
        digitalWrite(ledR, HIGH);
        delay(100);
        digitalWrite(ledR, LOW);
        delay(100);

        }
        }
```

2.7 Using ultrasonic module to make the car run automatically

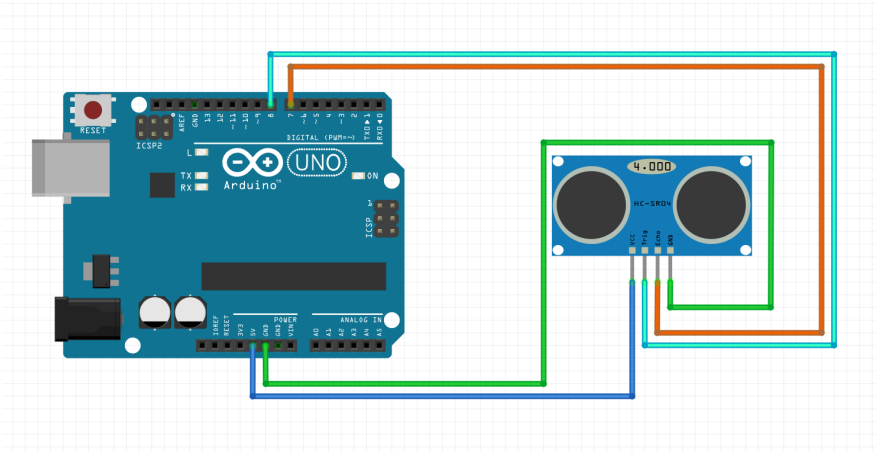


Figure 10: Schematic diagram of ultrasonic module HC-SRC04

We used ultrasonic module HC-SR04 to measure the distance in front of the car. Ultrasonic sensor HC-SR04 provides an output signal proportional to distance based on the echo. The sensor here generates a sound vibration in ultrasonic range upon giving a trigger, after that it waits for the sound vibration to return. Now based on the parameters, sound speed (240m/s) and time taken for the echo to reach the source, it provides output pulse proportional to distance. Because the sound speed is 340 m/s which is equivalent to 29,412 microSeconds/cm and the sound go through the distance 2 times, then we divide the pulse by $2 * 29,412$ to find the distance.

```
    unsigned long duration;
    int distance;

    /* Provide sound vibration from trig*/
    digitalWrite(trig,0);
```

```
    delayMicroseconds(2);
    digitalWrite(trig,1);
    delayMicroseconds(5);
    digitalWrite(trig,0);

    /* Calculate the time */
    duration = pulseIn(echo,HIGH);
    /* Calculate the distance*/
    distance = int(duration/2/29.412);
```

Then we will make the car run depends on the distance it receives:
- If the distance >15cm, the car will run forward.
- If the distance > 0cm and <=15cm, the car will turn left first. Because we just have one ultrasonic module, the car cannot recognize which is more efficient: turn left or right when it meets an object. Then we make it turn left and right alternatively. We add a variable s to determine the turning direction. For example, at the beginning s=0. If the car hits an object at the front, it will turn left first and s increases 1 unit. If there is another object at left side which hinders the car to turn left, after 3 times try to turn left, it means when s>=3, the car will turn right. When the car can go forward, we reset s=0.

```
  if(distance >15 )
     {
       s=0;
        runmotor(1,1,s1);
     runmotor(2,1,s1);
     }
      else if(distance<=15 && distance>0)
      {
     if(s<3)
     {
     digitalWrite(ledL, HIGH);
     delay(100);
     digitalWrite(ledL, LOW);
     delay(100);
     runmotor(1,1,s1);
     runmotor(2,-1,s1);
     }
     else if(s>=3)
     {
     digitalWrite(ledR, HIGH);
     delay(100);
     digitalWrite(ledR, LOW);
     delay(100);
     runmotor(1,-1,s1);
```

```
25        runmotor(2,1,s1);
          }
      }
```

3 Results

Our team conducted some experiments to check the ability of our robot.

3.1 Control car by using bluetooth:

After set up for the car, we used the software on our smartphone which had connected to the car by bluetooth to make it run forward, backward, turn left, turn right, forward left/right, backward left/right.
See the result at `https://youtu.be/Vt_Xj21v0WA`

3.2 Checking bluetooth distance:

We checked how far we can control the robot by using bluetooth to connect our smartphone and module HC-06 on the robot. We put the car on a straight line and make it run forward until we lose its control (the car cannot turn left/right or stop when we press the corresponding button on smartphone).

No.	Result
1	12m
2	12,5m
3	12m
4	13m

Table 4: Maximum distance

So the average maximum distance is around 12,5m.

3.3 Checking maximum speed:

We set the maximum speed 255 for each motor and make the car run forward. Then we measure the distance which it can go during each 10 seconds.

No.	Distance	Speed
1	320cm	0,32m/s
2	335cm	0,335m/s
3	325cm	0,325m/s
4	330cm	0,330m/s

Table 5: Maximum speed

So the average speed of the car is:

$$\frac{0,32 + 0,335 + 0,325 + 0,330}{4} = 0,3275 \pm 0,0025(m/s)$$

3.4 Checking the ability to run automatically by sensor:

See result at `https://youtu.be/TXFraKmx2a4`

4 Discussion

After 1 month working with the project, finally our team had built a complete car and some programs for which we can control the car remotely by using bluetooth or let the car run automatically by using ultrasonic module. We had learnt many thing new from arduino:
- interacting with other devices by using arduino through Serial port.
- controlling the motor by using module L295N: control the direction of the rotation of the motor and adjust its rotational frequency.
- using module bluetooh HC-06 to connect with the arduino.
- using a software on android to control the car remotely.
- using ultrasonic module to make the car run automatically.

With the existent software on android, we used it to control our car with bluetooth connection. When we touch a button on the screen, this software will recognize it and then send the corresponding keycode to the arduino board. We made the arduino board read this keycode and change to signal to the motor so the car can run with exact direction of the button we put.
With the ultrasonic module, we put it at the front side of our car and it gives us the data about the distance with the obstacles. From this data, we can make the car run forward, turn left/right automatically without hitting the objects.

On the other hand, during this time, we also encountered some problems. First, because the mechanic of the car was not great (2 wheels were not perfectly symmetric), the car cannot run smoothly. Sometimes when we let it run forward, it slightly run forward left/right, because the speed of 2 tires were not the same. The power of the motors was also a problem. It was quite weak so the car cannot climb over even though small inclines. Second, bluetooh connection was not perfect. The distance was small and it can be disturbed by other devices or suddenly disconnected.

In the future, we will improve the robot by replace by more powerful motors for which the car can run faster and more smoothly. We will try using wifi connection instead of bluetooh connection to enhance the maximum distance which we can connect to the car. Moreover, we will find another way to measure the distance in front of the car because using ultrasonic was not great and often interrupted.

In conclusion, although this is just a small project, we had learned many thing from it and also useful experiences when working with arduino and robot. They are very helpful for us to study in many later major subject.

References

[1] *Dfrduino Uno R3 - Arduino Compatible - Dfrobot.*
`https://www.dfrobot.com/product-838.html`

[2] *Ultrasonic Sensor Module HC SR04 Tutorial | Maxphi Lab*
https://www.maxphi.com/ultrasonic-sensor-interfacing-arduino-tutorial

[3] *JRodrigoTech/Ultrasonic-HC-SR04*
https://github.com/JRodrigoTech/Ultrasonic-HC-SR04, September 29, 2015

[4] *Arduino Bluetooth with HC-06*
https://techtutorialsx.com/2016/01/02/arduino-bluetooth-with-hc-06/, January 03, 2016

[5] *L298N H-Bridge DC Motor Driver Module Quick Start Guide*
https://www.bluetin.io/guides/l298n-h-bridge-dc-motor-driver-guide/,