MEC 104 Report

Lab 2: Smart car

Group number: A34

Group numbers

Zhongpei.Wang 1928748
Zhihan.Zhang 1929070
Qihang.Yao 1930733

Contents

Abstract

In this experiment, we will use Arduino Uno to achieve two main goals. The first one is we can use IR remote control to send orders so that the smart car moving forward or backwards at different speeds. The second goal is we will use ultrasonic sensor to make the smart car recognize the object. Thus, it can stop moving to avoid dashing against the object. During the experiment, we firstly identify the components and the code structure we should use. Then, we construct the circuit and design the code. After that, we test this device many times to revise bugs in the circuit and code. Finally, we successfully achieve all the goals and have a deep understanding of electric machinery.

Introduction

Arduino has long been known to be a powerful tool for software and hardware editing. It is cheap, robust and multifaceted. We should not explore its functions and capabilities overnight, but step by step. Building a smart car with Arduino is an interesting project without losing its academic rigor. Through our design and debugging, our car can realize two major functions: one is to switch speed through an infrared remote control. The car has five speed gears: neutral, forward, fast forward, back and fast back. The second function is ultrasonic obstacle avoidance. When an obstacle comes within the range of the ultrasonic locator (within 100 cm), the car's speed will decrease and it will break when it is 50 cm away from the obstacle. These two seemingly basic functions represent the smart car's ability to interact with the outside world. They are also the basis for other viable functions. In this project, the writing of the code was undoubtedly

at the center of everything. By implementing the code on the virtual compiler, we successfully project the functionality of the program onto the corresponding hardware entity. The Arduino motherboard plays a big part in this. Countless integrated circuits complement each other to make our complex functions a reality. As a matter of fact, we encountered many difficulties and made all kinds of small mistakes in the experiment. But thanks to our unremitting efforts, these problems have been properly solved. We recorded the causes and solutions of these problems in the experimental report. Both experimental methods and debug procedure were all recorded. We hope this will help readers to understand the Arduino system better and have a constructive overlook of the future study.

Experimental procedure

1. Theory

1.1 Hardware

If you want a simple DC motor to run, a single power source connected to the motor will do the trick. But we're looking for more than that. In a standard motor control circuit, the presence of an NPN transistor is very important.

NPN transistors are essential components for motor motion control. By adjusting the pins, you can control whether the current can flow. When a voltage is applied to the base is high enough, it can rotate the motor through the action of the transistor. The 5V voltage generated by the Arduino I/O pins is enough to power up the transistor and run the entire circuit. With the PWM pin, the speed of the motor can be controlled by the frequency of fast switching the transistor because the momentum of the internal rotor of the motor will be maintained to keep it rotating during the period when the power is not on. At this point, the PWM signal cycle determines the speed of the motor [1]. As the DC motor rotates, the energy is absorbed and stored in the inductance of the motor coil. If the energy supply is suddenly lost, the energy dissipates in the form of a reverse voltage spike, which is harmful to the entire circuit.

Therefore, plugging a diode into a circuit is a very effective way to prevent the reverse voltage from damaging the circuit. In addition, an effective DC motor requires the ability to rotate in both directions. It is easy to imagine that this requires switching the direction of current in the circuit at any time. That is where a device called the H-Bridge comes in handy [2]. In common, there will be 16 pins in a H-Bridge device which make it looks like a multi-legged worms. Different pins have different functions so it is necessary to recognize well when we connect wires on the pins.

As for ultrasonic sensors, they are thought to have been inspired by the flight of bats. The principle is simple. The sensor sends out an ultrasonic wave forward, the ultrasonic wave encounters an obstacle, produces reflection, is accepted by the sensor again. Through this time difference, we can know the distance between the obstacle and the car [3].

1.2 Software

The simulation functions we will use in the program design part is quite easy.
Move forward:

```
void forward (int rate)
{
  digitalWrite(EN, LOW);
  digitalWrite(MC1, HIGH);
  digitalWrite(MC2, LOW);
  analogWrite(EN, rate);
}
```

Turn off the driver before you reset the driver input

Move backwards:

```
void reverse (int rate)
{
  digitalWrite(EN, LOW);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, HIGH);
  analogWrite(EN, rate);
}
```

When we change the magnitude of the variable “rate”, we can change the speed of the motors.
Brake:

```
void brake ()
{
  digitalWrite(EN, LOW);
  digitalWrite(MC1, LOW);
  digitalWrite(MC2, LOW);
  digitalWrite(EN, HIGH);
}
```

Turn on the driver such that the motor remains stopped

What we need to do is call these functions in turn when the system receives five different infrared signals from the remote controller.

2. Circuit design

In this experiment, we are required to use IR remote to control the rotating direction and the speed of the motor. Moreover, the ultrasonic sensor is used to detecting the object.

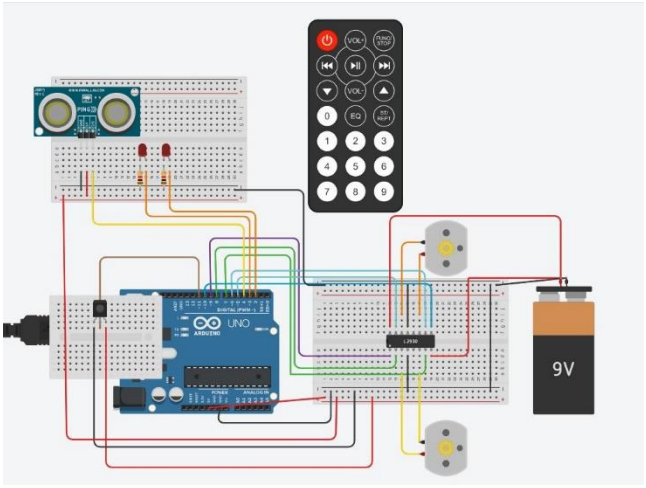


Figure 2: Circuit for smart car

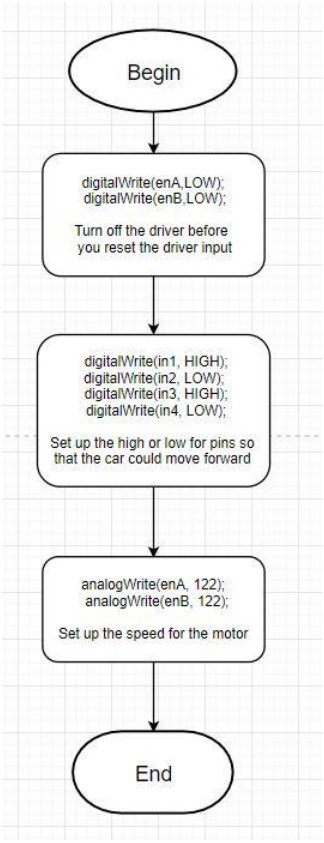
As Figure 2 shows, we connected the two motors to the L2930 chips and supplied them with the 9V battery. You should identify each pin of the L2930 in case the chip could break down. What is more, a small breadboard is applied for connecting the infrared sensor. You should pay attention to each pin of the sensor. Another breadboard is used to construct the ultrasonic sensor. Remember to connect the LED with a resistor. Otherwise, the LED will break down.

3. Code structure

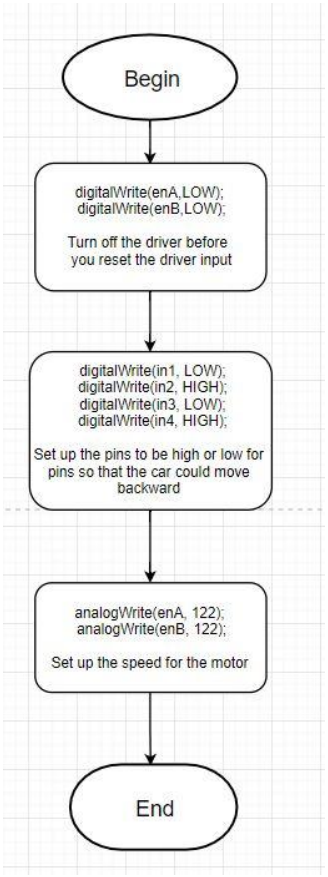
As figure 2 shows, we define the input and output pins for the electrical machinery, IR remote controller and ultrasonic sensor. More details of pins can be found in the Appendix.

For the basic functions, you can follow the flow charts below.

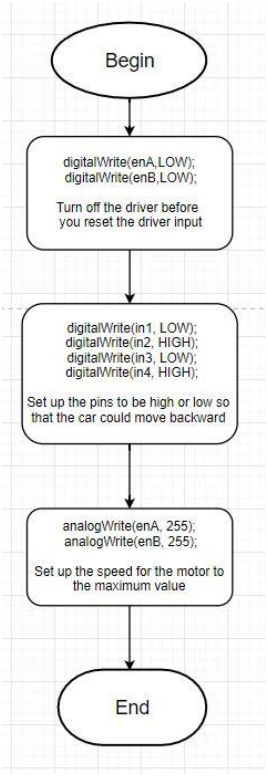
Forward: forward()



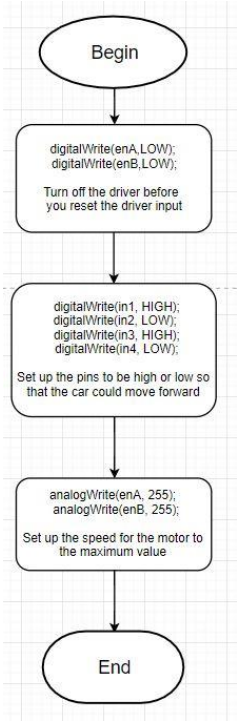
Backwards: backward()



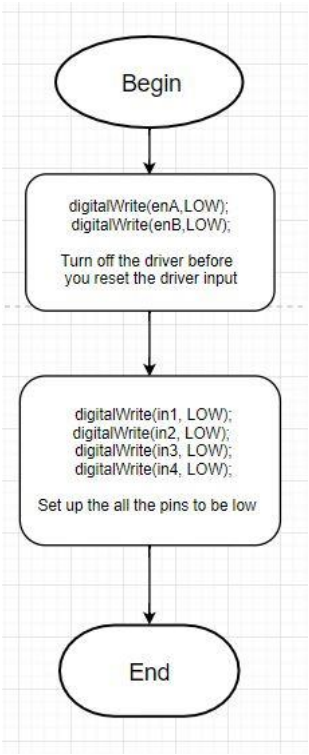
Fast backwards: fastbackward()



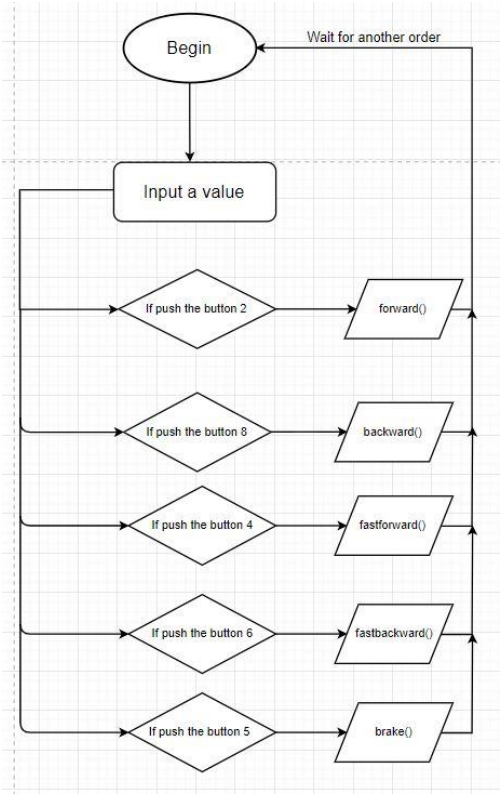
Fast forward: fastforward()


Begin
digitalWrite(enA,LOW);
digitalWrite(enB,LOW);
Turn off the driver before you reset the driver input
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
Set up the pins to be high or low so that the car could move forward
analogWrite(enA, 255);
analogWrite(enB, 255);
Set up the speed for the motor to the maximum value
End

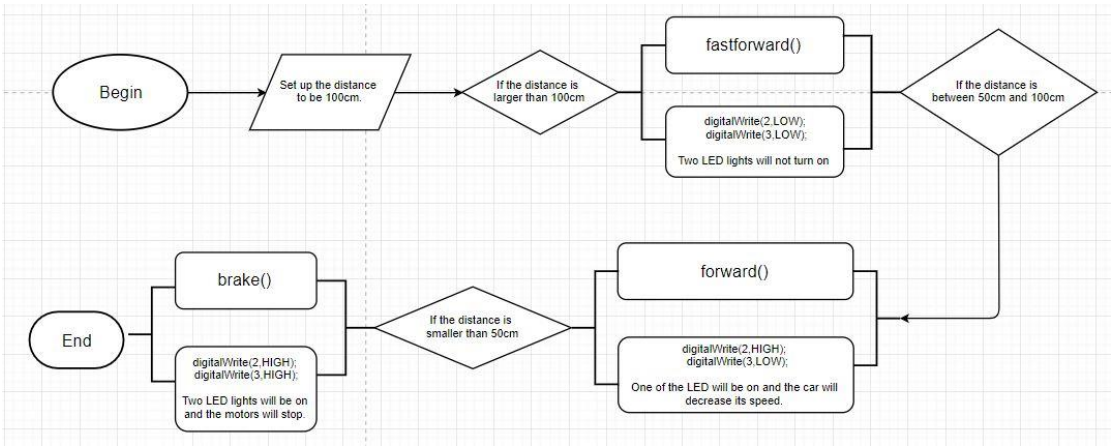
Break: brake()


Begin
digitalWrite(enA,LOW);
digitalWrite(enB,LOW);
Turn off the driver before you reset the driver input
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
Set up the all the pins to be low
End

For the advanced function, the flow chart for IR remote is shown below:



The flow chart for ultrasonic sensor is shown below:



4. Problems during debugging

The experimental car can be mainly divided into three parts, namely, the part responsible for controlling the movement of the two motors, the remote control end for receiving and processing remote signals, and the distance measuring part for processing distance information. The following will describe the problems arising from these three parts.

The first part is the moving part. No bugs were found in the separate tests of the moving part, which can normally carry out the steps of fast forward, forward, backward, fast backward and stop. The speed is expected and controllable.

The second is the distance finding module. When testing the distance finding module, we found that the two indicator lights could not be completely off when returning from the brake stop state to the state of fast progress, and the indicator lights were still on and off in the state of slow driving. Later, we found that after returning to the normal distance, we had jumped out of the algorithm of ranging operation and entered the algorithm of remote control. In the algorithm of ranging operation, it could not be realized to control both lights to be off completely. Therefore, we initialized the lights in the main function to solve the problem.

In this experiment, there are some big problems in the remote control plate. After the remote control module is connected, the traveling module produces a strange bug. Caused a bug where two motors would turn while not turning after giving specific instructions. That is, only the right motor rotates when going forward, and only the left motor rotates when going backwards. Fast forward normal, fast back all do not turn. Repeated screening of the code and circuit connections found no problems. But it was found that the left motor only rotated at full speed when moving forward. Since the problem requires only forward and backwards, we also tried to connect the two motors to the same end for control. After the discovery, there is still a problem, connected in the upper forward motionless, lower backward motionless. After the input and output terminals of the two motors are connected to the oscilloscope, it is found that there is no input on the side that does not rotate. After checking the code line by line, it was found that a line of seemingly insignificant code caused a bug. After querying the data, we found that some interfaces use the same one, while both infrared and PMW need to use a timer. The conflict of the timer led to the motor failure, and the problem was solved by changing the interface.

5. Results

After a long process of program modification, we finally arrived at the testing phase. The first is the task of controlling the speed with the infrared remote control. In the ley of different numbers on the remote controller, “2” key indicates a low speed forward (corresponding to 8,000 RPM). The “8” key represents a low speed backward (-8000 RPM). The “4” key represents a high speed forward (corresponding to 16000 RPM). The 6 keys represent a high speed backward (-16000 RPM). After pressing those keys we fortunately found that 2 motors rotated as expected. In the part of ultrasonic obstacle avoidance, we let the motor of the car rotate and place obstacles 50cm away. Then, we slowly moved the obstacle into the 50cm range. Two LED warning lights went on, and the motor started to brake automatically and eventually stopping. We have carried out the task successfully. A short video can show our experimental results.

6. Discussion

There are some problems in this code and circuit design. Although this project can complete all the experimental requirements, it lacks certain humanization in part of the design.

For example, when the obstacle avoidance program is carried out, the ultrasonic

rangefinder has an error with the actual distance when calculating the distance, so the determination of 100cm and 50cm will be farther than the actual distance. The problem may be because the ultrasonic rangefinder can not measure at a relatively close distance caused by the reserved distance, can be assembled in the car to move the ultrasonic rangefinder back to complete.

Secondly, due to code design problems, the ultrasonic rangefinder cannot be changed when the brake stops, that is, the control of the remote control will fail within 100cm, and it needs to be moved out of the range of 100cm manually. Therefore, some changes can be made to the code itself to integrate the ultrasonic ranging statements into the two forward statements. That is, it is judged before moving fast, and if it is less than 100cm, it is not executed. Judgment is made before moving forward. If the obstacle distance is less than 50cm, the moving instruction is not executed. This allows the vehicle to be operated with a remote control while performing obstacle avoidance, allowing it to still move backwards without the need for human movement.

The system of ranging the brake part is not made to slow down with the narrowing of the distance, but only a design of stopping the brake at a slow speed below 100cm and below 50cm.It is possible to change the function into a cyclic function with a decreasing variable to make the braking of the vehicle more humane. In other words, the braking speed of the vehicle is set through a circular statement. The motor speed is 255 when it is 100cm, and then the motor speed is reduced by 3 when it is close to 1cm until it finally stops.

At the same time, the whole software and system have some errors in actual operation and simulation. Since the output voltage is not absolutely stable, even at a fixed speed, the output speed of the electric motor is not a stable value. It will fluctuate around 16000 RPM at full speed, and around 8,000 RPM at half speed. This error cannot be corrected by code or other software Settings, but only by optimization of hardware components and the running environment.

Conclusion

In conclusion, we achieve all the goals. Our project demonstrates the fluency of the above code structure in the interaction between hardware and software. And we have learned the circuit characteristics needed for an excellent remote instruction receiving system to interact with a negative feedback environment. This will pave the way for our future study in electrical engineering.

Author list and contributions

No.	Name	Contribution description	Percentage	Signature
1.	Zhongpei.Wang	Code design, report	33.3%	王仲佩
2.	Zhihan.Zhang	Circuit design, report	33.3%	张之涵
3.	Qihang.Yao	Debug, report	33.3%	要启航

References

[1] Z. Chen, "*DC Motor Speed Control Method and System*", publication of CN103532452A, 2013, p. 2.
[2] F. Liu, "Environmental Protection and Energy Saving H-Bridge and SPWM DC Power Inverter", *Electronic Industry Press*, May 2010, pp 23-27.
[3] X. Zhou, " 200 Cases of Practical Remote Control Circuit", *China Electric Power Press*, 2019, pp 52-61.

Appendix

```
#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
// Motor A connections
int enA = 9;
int in1 = 8;
int in2 = 7;
// Motor B connections
int enB = 10;
int in3 = 6;
int in4 = 5;
int distanceThreshold = 0;
int cm = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);  // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  // Reads the echo pin, and returns the sound wave travel time
  return pulseIn(echoPin, HIGH);
}
```

```
void setup() {
  Serial.begin(9600);
   irrecv.enableIRIn();
    // Set all the motor control pins to outputs
   pinMode(enA, OUTPUT);
     pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
}
void forward()
  {
  digitalWrite(enA,LOW);
  digitalWrite(enB,LOW);
  analogWrite(enA, 122);
  analogWrite(enB, 122);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

}


 void backward()
 {
   digitalWrite(enA,LOW);
   digitalWrite(enB,LOW);
   analogWrite(enA, 122);
   analogWrite(enB, 122);
   digitalWrite(in1, LOW);
   digitalWrite(in2, HIGH);
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
 }
 void fastforward()
 {
   digitalWrite(enA,LOW);
   digitalWrite(enB,LOW);
   analogWrite(enA, 255);
   analogWrite(enB, 255);
   digitalWrite(in1, HIGH);
   digitalWrite(in2, LOW);
   digitalWrite(in3, HIGH);
   digitalWrite(in4, LOW);

 }
 void fastbackward()
 {
   digitalWrite(enA,LOW);
   digitalWrite(enB,LOW);
   analogWrite(enA, 255);
   analogWrite(enB, 255);
   digitalWrite(in1, LOW);
   digitalWrite(in2, HIGH);
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);

 }
```

```
void brake()
{

  digitalWrite(enA,LOW);
  digitalWrite(enB,LOW);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}

void sensor()
{
  // set threshold distance to activate LEDs
  distanceThreshold = 100;
  // measure the ping time in cm
  cm = 0.01723 * readUltrasonicDistance(4,4);
  // convert to inches by dividing by 2.54
  Serial.print(cm);
  Serial.print("cm, ");
  if (cm > distanceThreshold) {
     fastforward();
    digitalWrite(2,LOW);
    digitalWrite(3,LOW);
  }
  if (cm <= distanceThreshold && cm > distanceThreshold - 50) {
    forward();
    digitalWrite(2,HIGH);
    digitalWrite(3,LOW);
  }
  if (cm <= distanceThreshold - 50) {
    brake();
    digitalWrite(2,HIGH);
    digitalWrite(3,HIGH);
  }}
```

```
void loop(){

   cm = 0.01723 * readUltrasonicDistance(4, 4);
   distanceThreshold = 100;
 if(cm<=distanceThreshold){
  sensor();
   }
  else{
   digitalWrite(0,LOW);
   digitalWrite(1,LOW);

   if(irrecv.decode(&results)){
    switch(results.value){
     case 0xFD8877:  //forward,2
 forward();
       break;
     case 0xFD9867:  //backward,8
 backward();
       break;
     case 0xFD28D7:  //fast forward,4
   fastforward();
       break;
     case 0xFD6897:  //fast backward,6
 fastbackward();
       break;
     case 0xFDA857:  //Stop,5
  brake();
       break;
     default:
       break;

   }Serial.println(results.value,HEX);irrecv.resume();
 }
  }
}
```