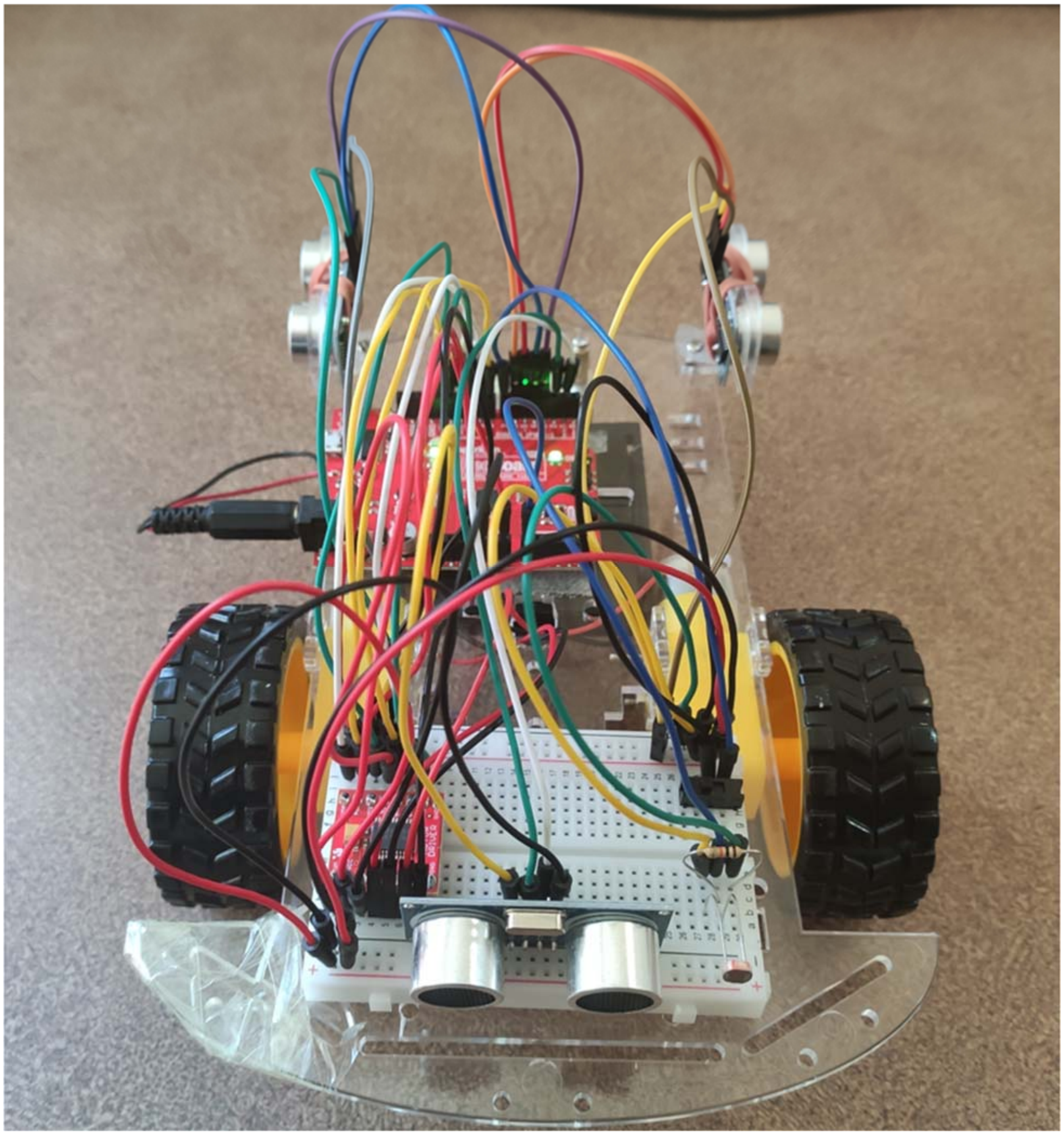
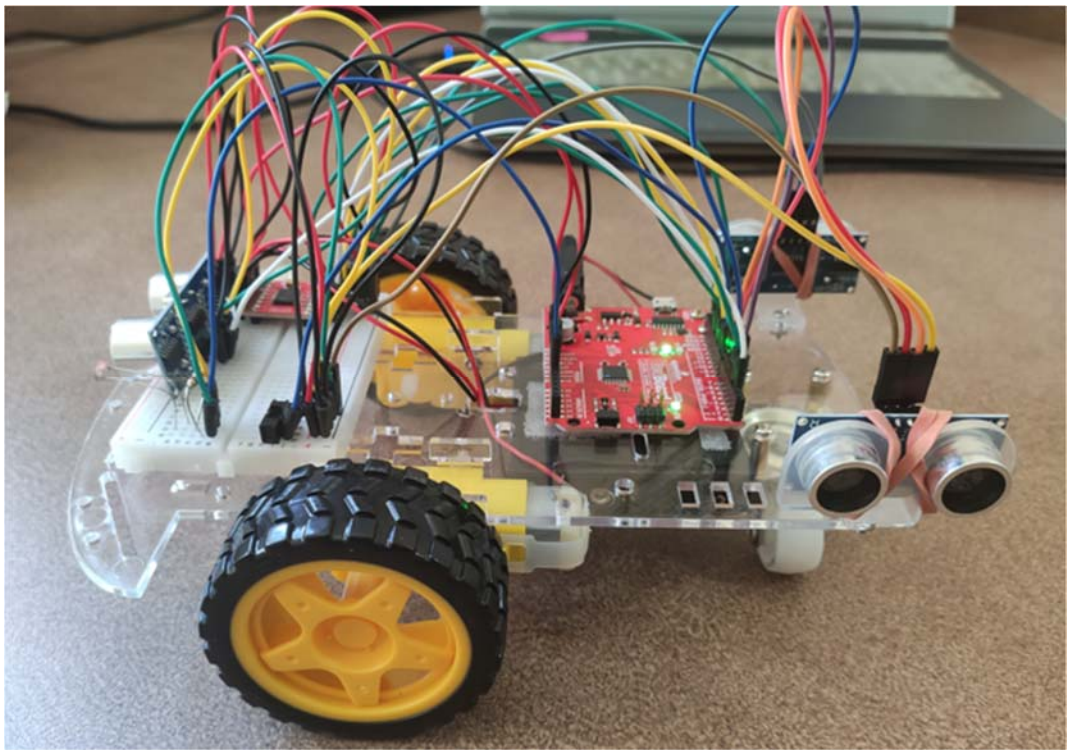
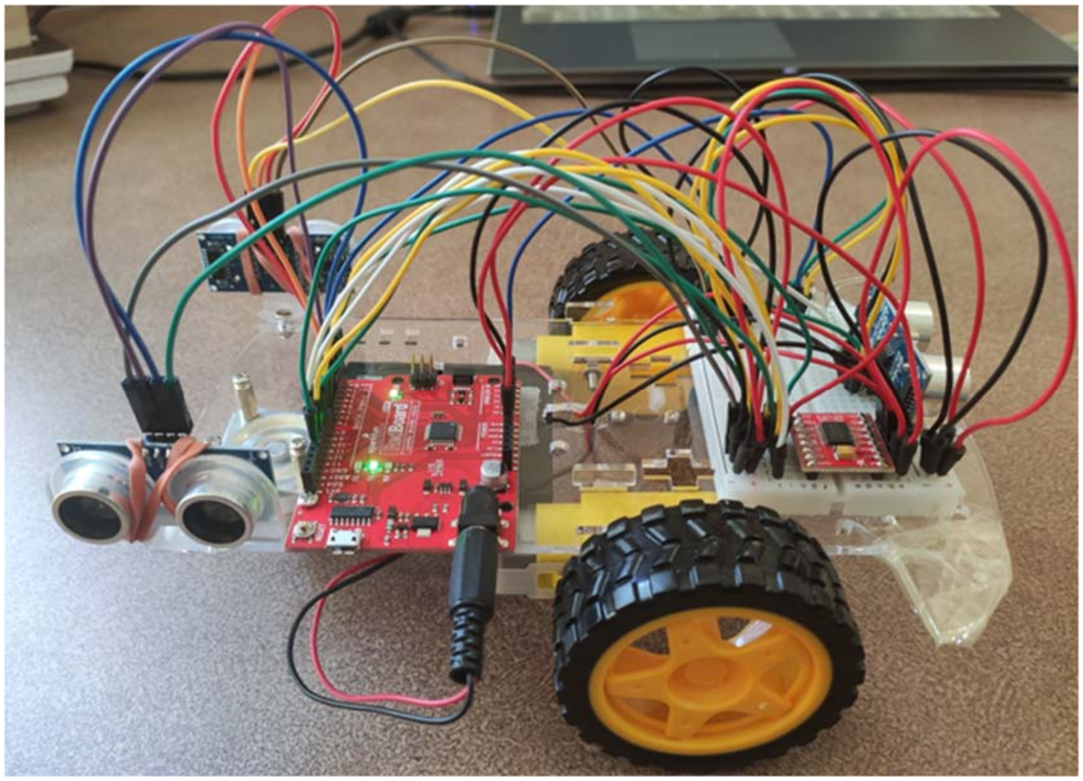
Locomotion Robot(Obstacle Avoiding)



The Robot’s Goal: the robot will move on the floor and avoid obstacle based on the feedback of three ultrasound sensors and one photoresistor.

- If there’s an obstacle on the left side,it wil move to the right side;
- If there’s an obstacle on the right side,it will move to the left side;
- If there’s an obstacle in front,it will reverse and turn to the side which has more space(compare the left ultrasound sensor’s distance and the right ultrasound sensor’s distance);
- If the robot detects the high brightness ,it will stop there.

The effectors: two differential steerable wheels and one caster.

The actuators: two DC motors with gear boxes controlled by the motor driver.

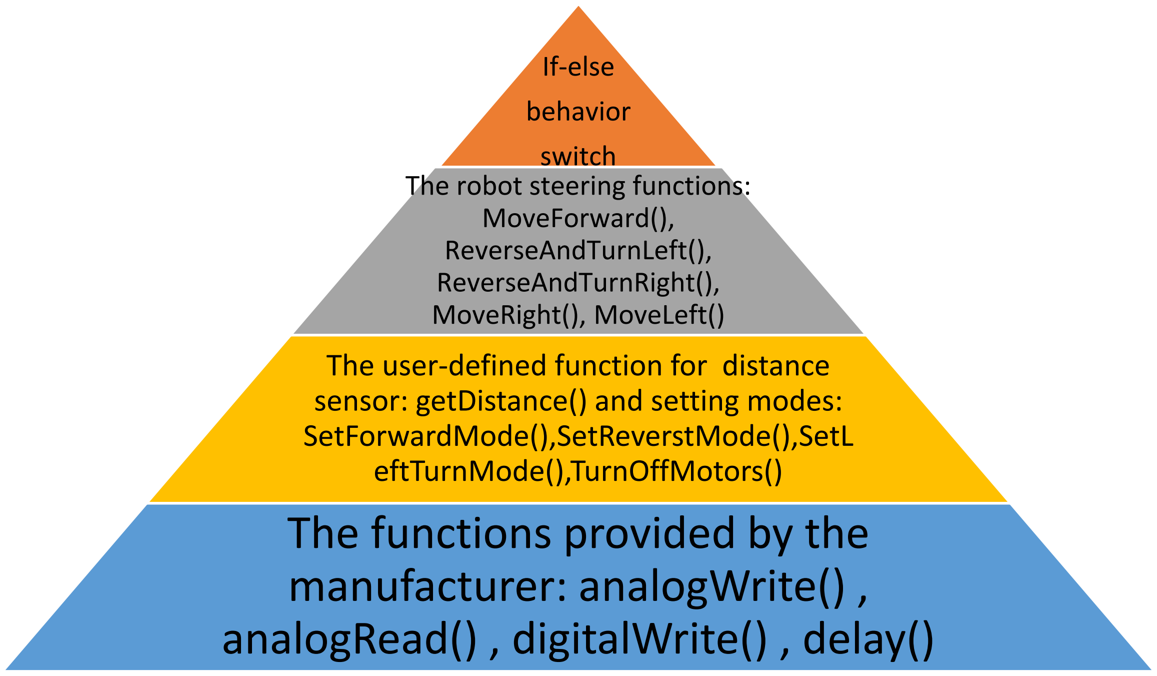
The controller: The Sparkfun RedBoard(Arduino platform).

The sensors: one photoresistor and three ultrasound sensors.

Design Decisions

- The robotics has two major subfields: locomotion robotics and manipulation robotics.Considering the available SIK components,locomotion robot is a more practical ;
- Comparing with locomotion robot,the manipulation robot needs to compute the inverse kinematics and dynamics which is more complicated;
- After viewing the previous COMP444 final projects' videos on Youtube,I found that the locomotion robot is also the choice of the majority;
- Reading over the whole textbook,the textbook is also focusing on introducing the relevant knowledge about the locomotion robot rather than the manipulation robot,so more knowledge could be extracted from the textbook easily;
- There are two similar locomotion robots in SIK projects,which provide a great deal of valuable instructions to build my own project;
- According to the textbook,the two differential steerable driven wheels and one caster is the most common and typical locomotion robot design;
- Considering the robot's task,three ultrasound sensors are going to be deployed:the front ultrasound sensor is used to prevent the robot from hitting the wall and measuring the appropriate distance from the wall before turning;the two ultrasound sensors of both sides are used to avoid obstacles ;
- The photoresistor should be installed in front,so that it can detect the high brightness as soon as possible;
- The idea using high brightness to signify the destination comes from the textbook: "A light can be used to mark a specila area,such as the battery recharging station or the end of a maze"---quoted from the textbook.

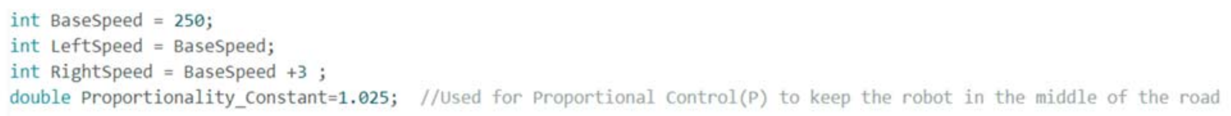
Control Architecture and Program Implementations


If-else behavior switch
The robot steering functions: MoveForward(), ReverseAndTurnLeft(), ReverseAndTurnRight(), MoveRight(), MoveLeft()
The user-defined function for distance sensor: getDistance() and setting modes: SetForwardMode(),SetReverstMode(),SetLeftTurnMode(),TurnOffMotors()
The functions provided by the manufacturer: analogWrite() , analogRead() , digitalWrite() , delay()

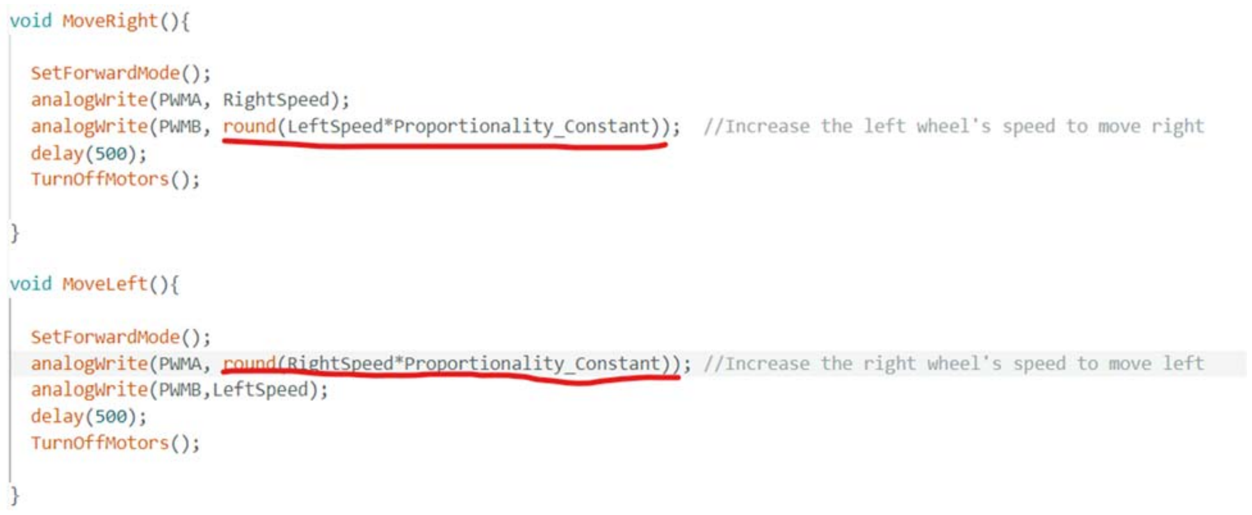
<u>The Subsumption Architecture of the behavior-based control system</u>

- The deliberative control requires a large amount of internal representations and planning computations,which is not fit for this task;
- The reactive control is part of other type of control,so it couldn't become an independent control system in most cases;
- The hybrid control's middle layer is very difficult to design,so ruled out;
- The behaviour-based control system grew out of reactive control system,and inherits many advantages from the reactive control system: react in real-time;
- Behavior-based control is easy to be implemented with the Subsumption Architecture(Bottom-up),by which we could design and debug the system incrementally;

- The feedback control(closed loop control) is applied in this project,the controller will decide what action to do based on the feedback sensory information;
- Proportional Control is used to adjust the robot's trajectory by changing single wheel's speed;

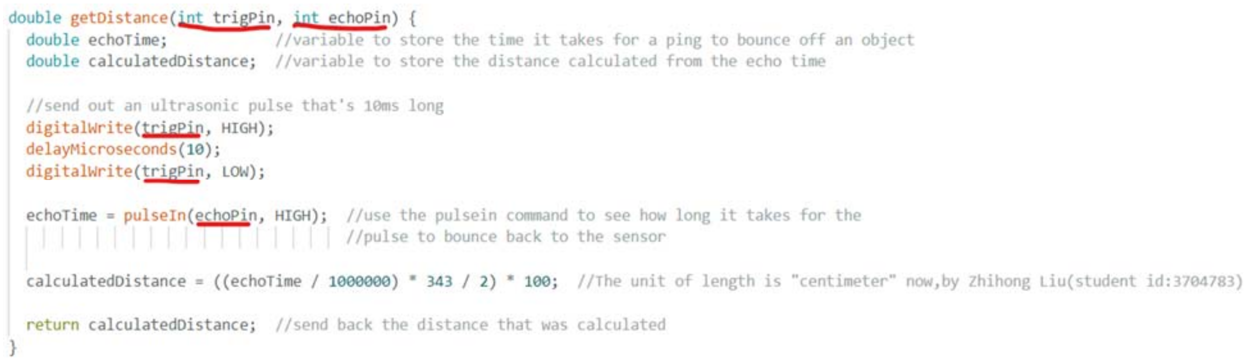


```
int BaseSpeed = 250;
int LeftSpeed = BaseSpeed;
int RightSpeed = BaseSpeed +3 ;
double Proportionality_Constant=1.025;  //Used for Proportional Control(P) to keep the robot in the middle of the road
```

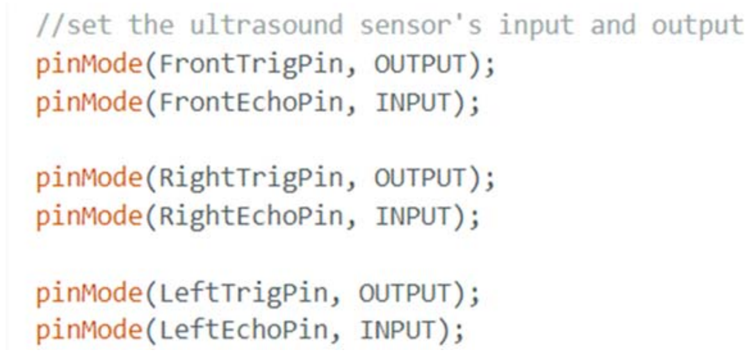


```
void MoveRight(){

  SetForwardMode();
  analogWrite(PWMA, RightSpeed);
  analogWrite(PWMB, round(LeftSpeed*Proportionality_Constant));  //Increase the left wheel's speed to move right
  delay(500);
  TurnOffMotors();

}

void MoveLeft(){

  SetForwardMode();
  analogWrite(PWMA, round(RightSpeed*Proportionality_Constant)); //Increase the right wheel's speed to move left
  analogWrite(PWMB,LeftSpeed);
  delay(500);
  TurnOffMotors();

}
```

- The **getDistance()** function: the original version of this function provided by the manufacturer takes no parameter as there's only one ultrasound sensor whose variables have been set as global variables,so the original version could invoke these global variables inside its function body directly.However it doesn't work for my project as there are three ultrasound sensors,so I rewrite this function to take two parameters:

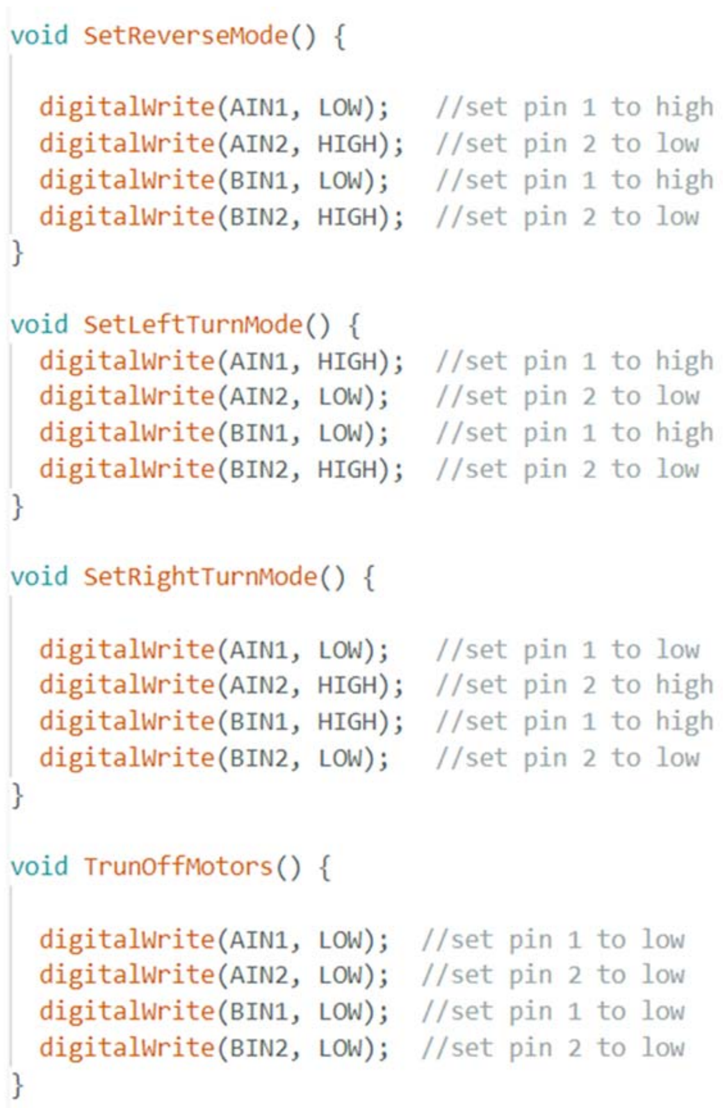


```
double getDistance(int trigPin, int echoPin) {
  double echoTime;          //variable to store the time it takes for a ping to bounce off an object
  double calculatedDistance;  //variable to store the distance calculated from the echo time

  //send out an ultrasonic pulse that's 10ms long
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime = pulseIn(echoPin, HIGH);  //use the pulsein command to see how long it takes for the
                                      //pulse to bounce back to the sensor

  calculatedDistance = ((echoTime / 1000000) * 343 / 2) * 100;  //The unit of length is "centimeter" now,by Zhihong Liu(student id:3704783)

  return calculatedDistance;  //send back the distance that was calculated
}
```



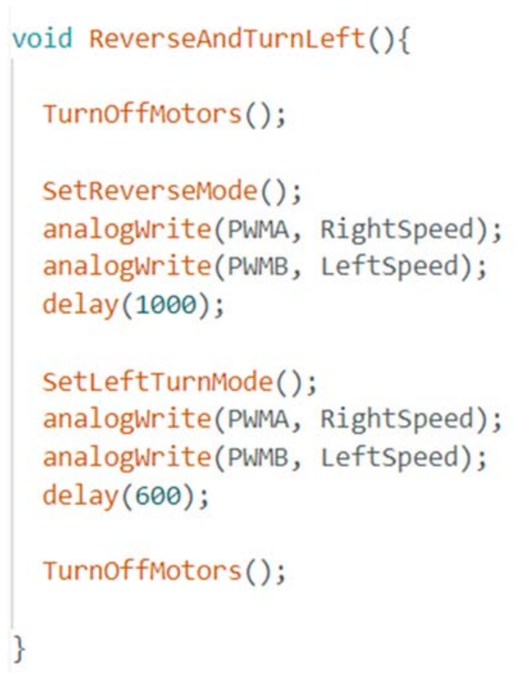
```
//set the ultrasound sensor's input and output
pinMode(FrontTrigPin, OUTPUT);
pinMode(FrontEchoPin, INPUT);

pinMode(RightTrigPin, OUTPUT);
pinMode(RightEchoPin, INPUT);

pinMode(LeftTrigPin, OUTPUT);
pinMode(LeftEchoPin, INPUT);
```

SetForwardMode(),SetReverseMode(),SetLeftTurnMode(),SetRightTurnMode(),TurnOffots(). These functions are frequently used,so I defined them as a set of subroutines;

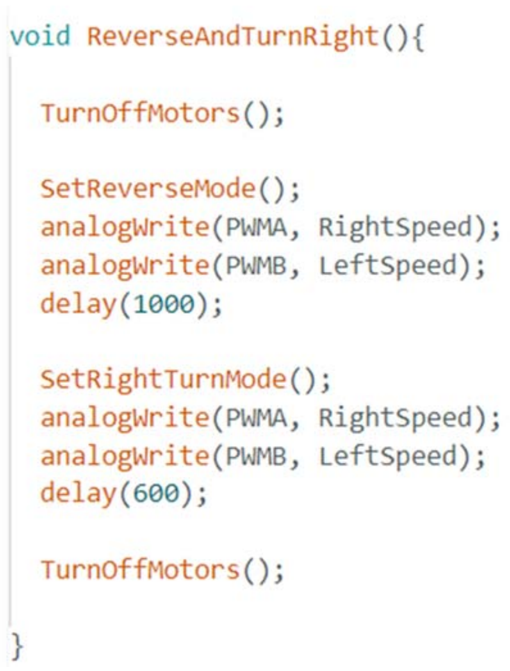


```
void SetReverseMode() {

  digitalWrite(AIN1, LOW);   //set pin 1 to high
  digitalWrite(AIN2, HIGH);  //set pin 2 to low
  digitalWrite(BIN1, LOW);   //set pin 1 to high
  digitalWrite(BIN2, HIGH);  //set pin 2 to low
}

void SetLeftTurnMode() {
  digitalWrite(AIN1, HIGH);  //set pin 1 to high
  digitalWrite(AIN2, LOW);   //set pin 2 to low
  digitalWrite(BIN1, LOW);   //set pin 1 to high
  digitalWrite(BIN2, HIGH);  //set pin 2 to low
}

void SetRightTurnMode() {

  digitalWrite(AIN1, LOW);   //set pin 1 to low
  digitalWrite(AIN2, HIGH);  //set pin 2 to high
  digitalWrite(BIN1, HIGH);  //set pin 1 to high
  digitalWrite(BIN2, LOW);   //set pin 2 to low
}

void TrunOffMotors() {

  digitalWrite(AIN1, LOW);  //set pin 1 to low
  digitalWrite(AIN2, LOW);  //set pin 2 to low
  digitalWrite(BIN1, LOW);  //set pin 1 to low
  digitalWrite(BIN2, LOW);  //set pin 2 to low
}
```

- The **Reverse and Turning** mechanism:



```
void ReverseAndTurnLeft(){

  TurnOffMotors();

  SetReverseMode();
  analogWrite(PWMA, RightSpeed);
  analogWrite(PWMB, LeftSpeed);
  delay(1000);

  SetLeftTurnMode();
  analogWrite(PWMA, RightSpeed);
  analogWrite(PWMB, LeftSpeed);
  delay(600);

  TurnOffMotors();

}
```



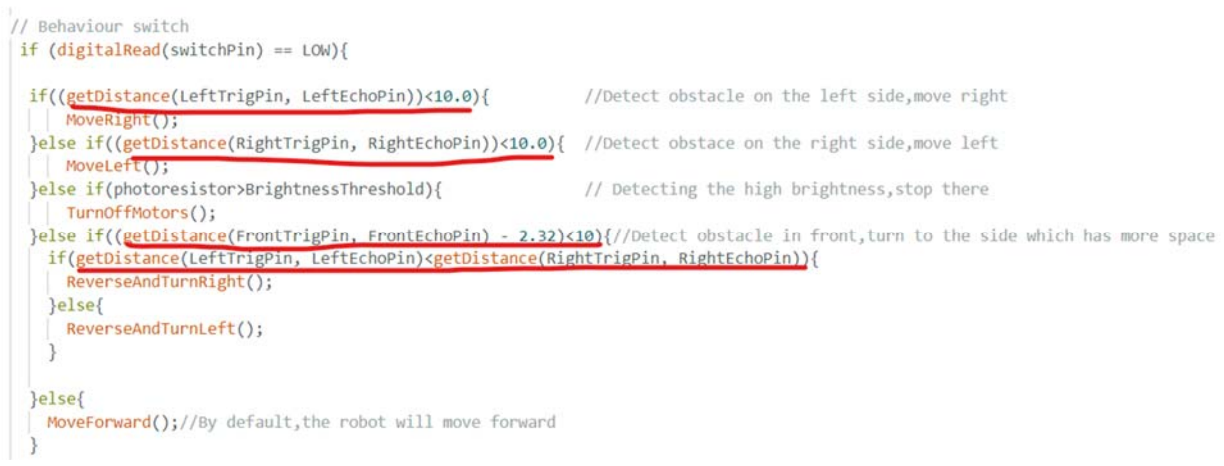
```
void ReverseAndTurnRight(){

  TurnOffMotors();

  SetReverseMode();
  analogWrite(PWMA, RightSpeed);
  analogWrite(PWMB, LeftSpeed);
  delay(1000);

  SetRightTurnMode();
  analogWrite(PWMA, RightSpeed);
  analogWrite(PWMB, LeftSpeed);
  delay(600);

  TurnOffMotors();

}
```

- The **Sensor problem** in **Coding**.

  “It is generally not a good idea to separate what the robot senses,how it senses it,how it processes it and how it uses it”---Quoted from the textbook.

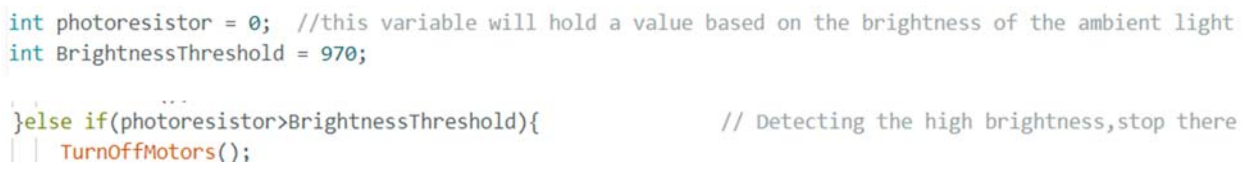
  I did meet with several sensor-related problems .Finally it turns out that I violated the principle above quoted from the textbook.Becuase I separated these parts in different places,the “Global sensor variables VS Local sensor variables” issue makes the program can’t get the correct and instantaneous sensory data,producing the wrong behaviours.

  So combing the author’s principle and my own debug experience,the best approach is handle sensor in writing the code is to **invoke these sensor functions whenever we need the sensory data** instead of defining variables to store the sensory data;



```
// Behaviour switch
if (digitalRead(switchPin) == LOW){

 if((getDistance(LeftTrigPin, LeftEchoPin))<10.0){         //Detect obstacle on the left side,move right
     MoveRight();
 }else if((getDistance(RightTrigPin, RightEchoPin))<10.0){  //Detect obstace on the right side,move left
     MoveLeft();
 }else if(photoresistor>BrightnessThreshold){               // Detecting the high brightness,stop there
     TurnOffMotors();
 }else if((getDistance(FrontTrigPin, FrontEchoPin) - 2.32)<10){//Detect obstacle in front,turn to the side which has more space
   if(getDistance(LeftTrigPin, LeftEchoPin)<getDistance(RightTrigPin, RightEchoPin)){
     ReverseAndTurnRight();
   }else{
     ReverseAndTurnLeft();
   }

 }else{
   MoveForward();//By default,the robot will move forward
 }
```

- The **Stopping Mechanism**: measuring the brightness in ordinary and non-light environment ,setting this brightness value as threshold ,once the photoresistor detects a higher birghtness value, it will stop there;



```
int photoresistor = 0;  //this variable will hold a value based on the brightness of the ambient light
int BrightnessThreshold = 970;

}else if(photoresistor>BrightnessThreshold){              // Detecting the high brightness,stop there
    TurnOffMotors();
```

Testing and Calibration

"In control theory,the parameters that determine the magnitude of the system's response are called **gains**"---quoted from the textbook

- The main purpose of the testing is to make sure the program and all other components could work properly to achieve the desired goal;
- It involves repetitive trials and errors to find the **optimal gains** ,which include :

delay duration;

photoresistor threshold;

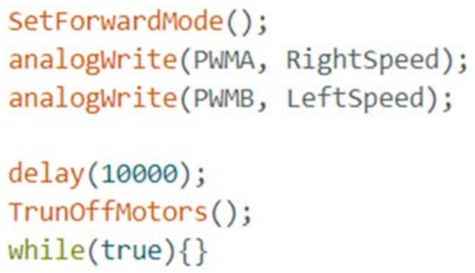
ultrasound sensor trigger distance ;

voltage for driven motor which directly decides the running speed.;

proportionality_constant which is used to change the single wheel's speed.

Testing for moving forward

- Goal : find the **<u>optimal voltage</u>** for driven motor to get the appropriate running <u>speed</u>;
- A single straight corridor needs to be prepared;
- Program preparation: Invalidate all other irrelevant codes inside the "void loop()",only keep the moving forward part;



```
SetForwardMode();
analogWrite(PWMA, RightSpeed);
analogWrite(PWMB, LeftSpeed);

delay(10000);
TrunOffMotors();
while(true){}
```

Some problems encountered during testing

- As the batteries are being consumed,even though with the same parameters,the robot will produce behaviours with a little discrepancies;
- This set of SIK-based hardware is more suitable for continuous and long-distance running,if the robot is applied with complex combination of discrete and short-distance running logic,it will be stuck somewhere.

Outcome

- This project does meet all the designed expection: the robot could move and avoid the obstacles effectively based on the feedback of sensors;
- The feedback control and proportional control can effectively adjust the robot's gesture accordingly;
- If I was to undertake the same project again,I would replace **the chassis** at the beginning. The original chassis provided by the manufacturer is a baseplate,not professional chassis.All the components are fixed with 3M Tape,not screw and nut.And the available space and notches are quite limited,so it's very difficult to install the accessories;
- I may also upgrade the controller board to others such as Raspberry Pi,which has much more powerful computation capability and available pins and ports to connect more accessories.