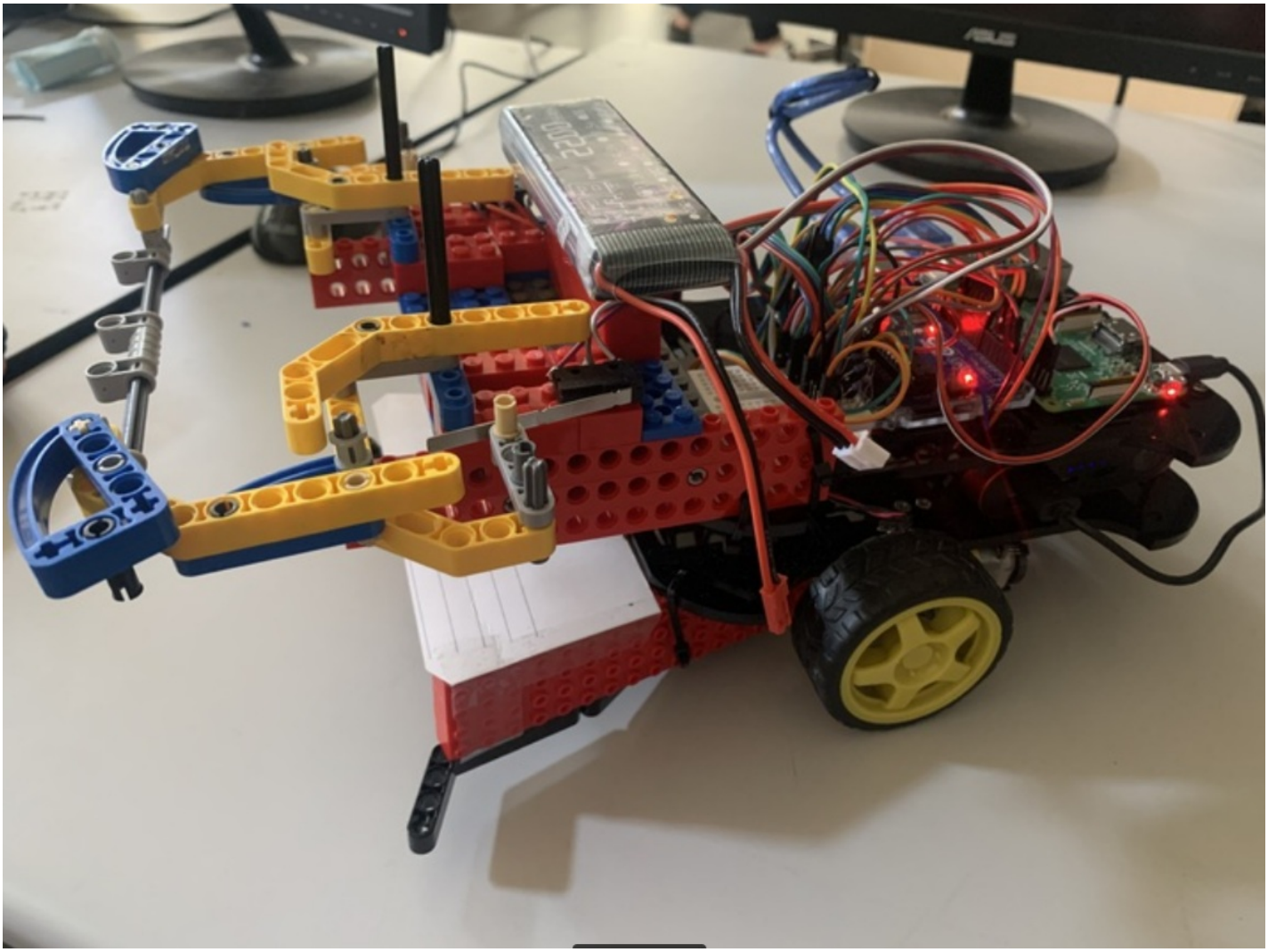
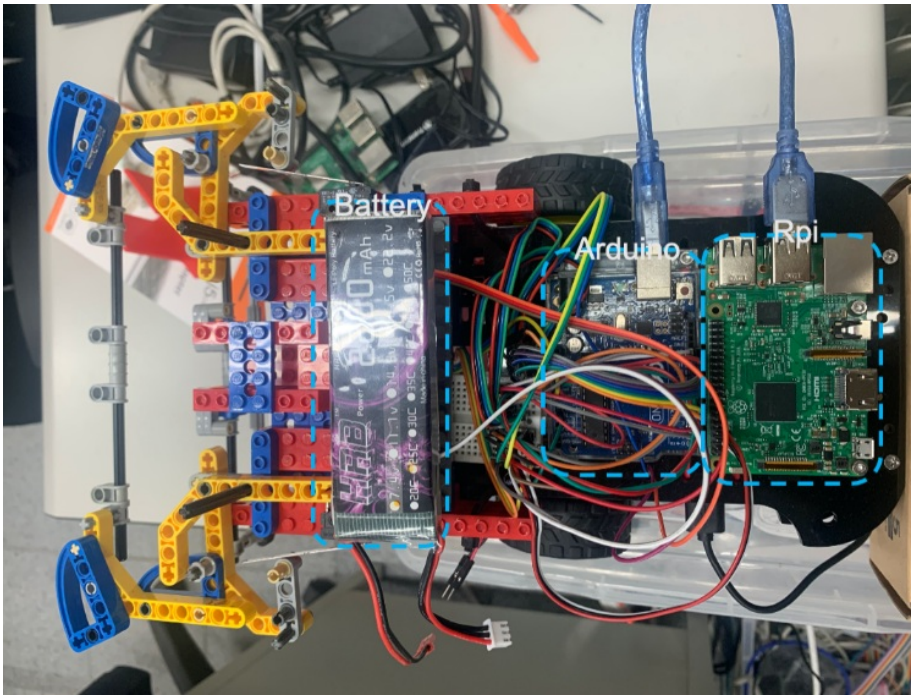
checkpoint3 report

硬體

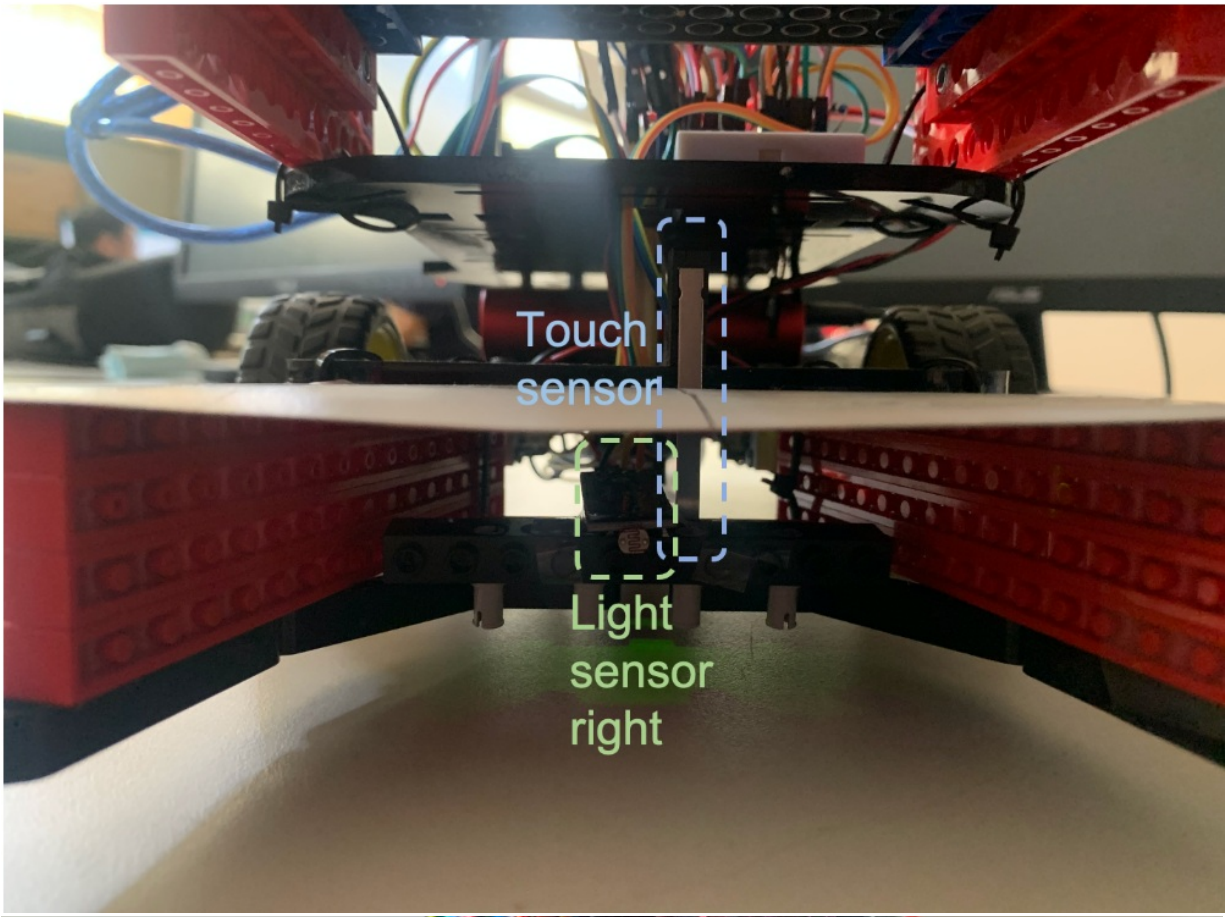
外觀

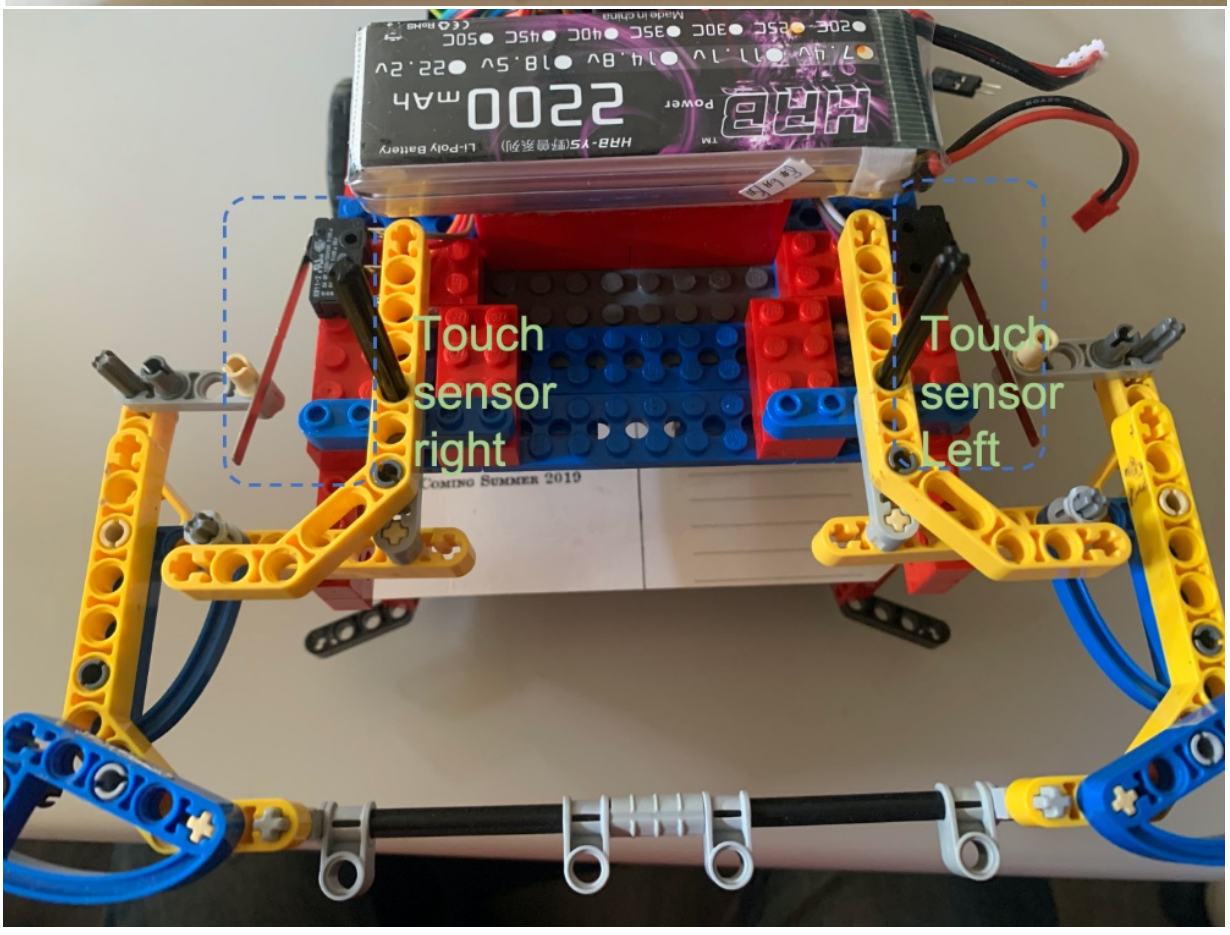


主要元件

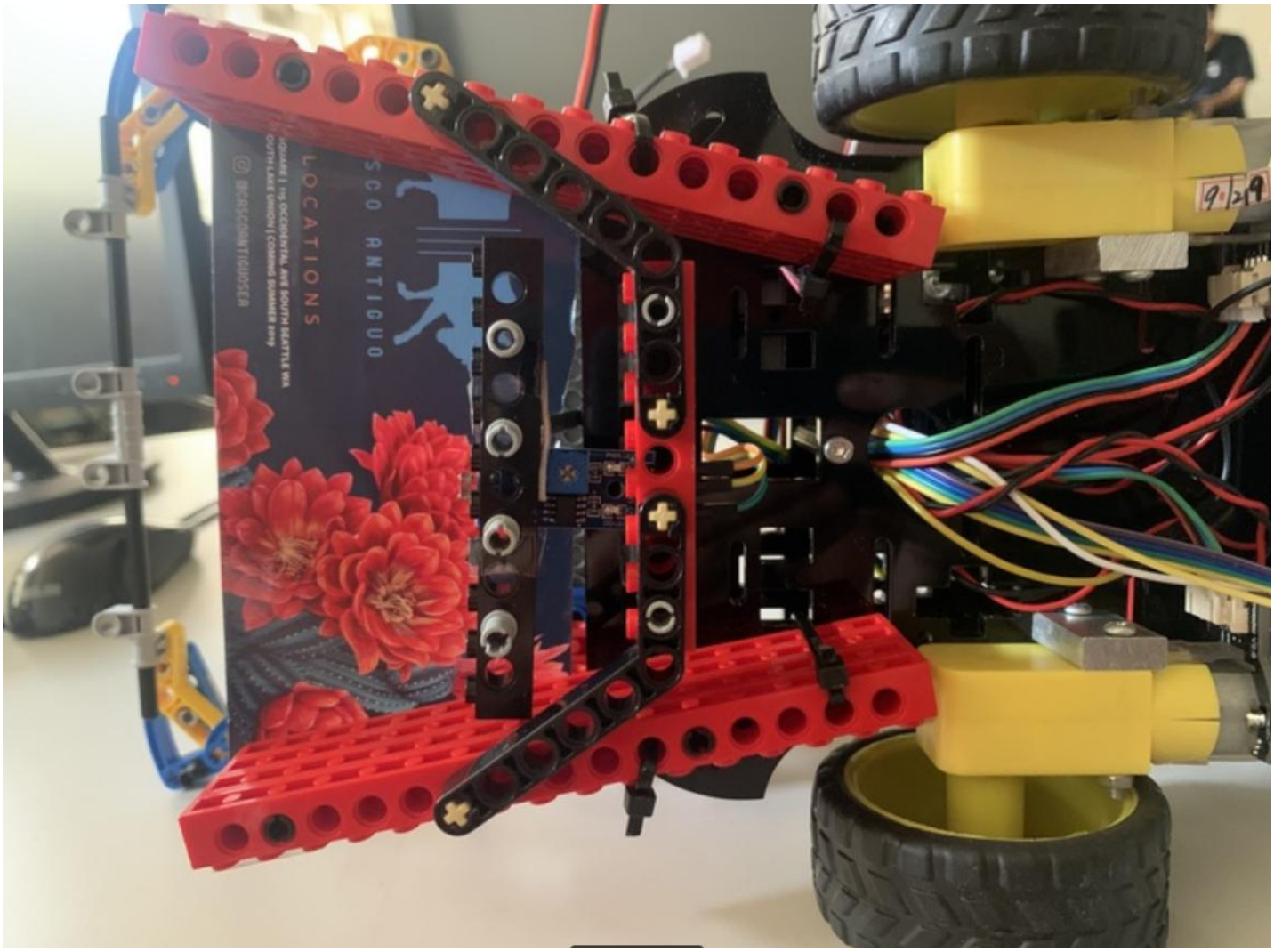


感測器配置


Touch
sensor
Light
sensor
right


Touch
sensor
right
Touch
sensor
Left

底部機構設計



軟體

感測器與GPIO

碰桿:

在樹莓派端程式使用wiringpi函式庫進行GPIO的控制。

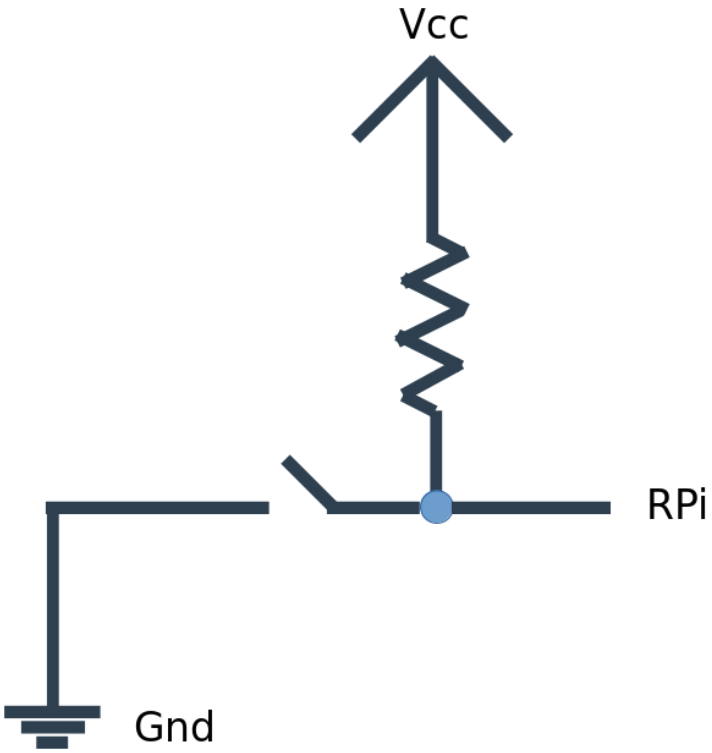
`wiringpi.wiringPiSetup() //初始化`

`wiringpi.pinMode(7, 0) //設定該pin為輸入腳位`

`wiringpi.digitalRead(7) //讀取數位輸入值`

數值範圍{0,1}，碰桿按下為0，鬆開為1。

電路設計:



使用一上拉電阻將訊號輸出值拉高至5V，唯碰桿按下時與COM接通，訊號輸出值降低至0V。

光敏電阻模組：

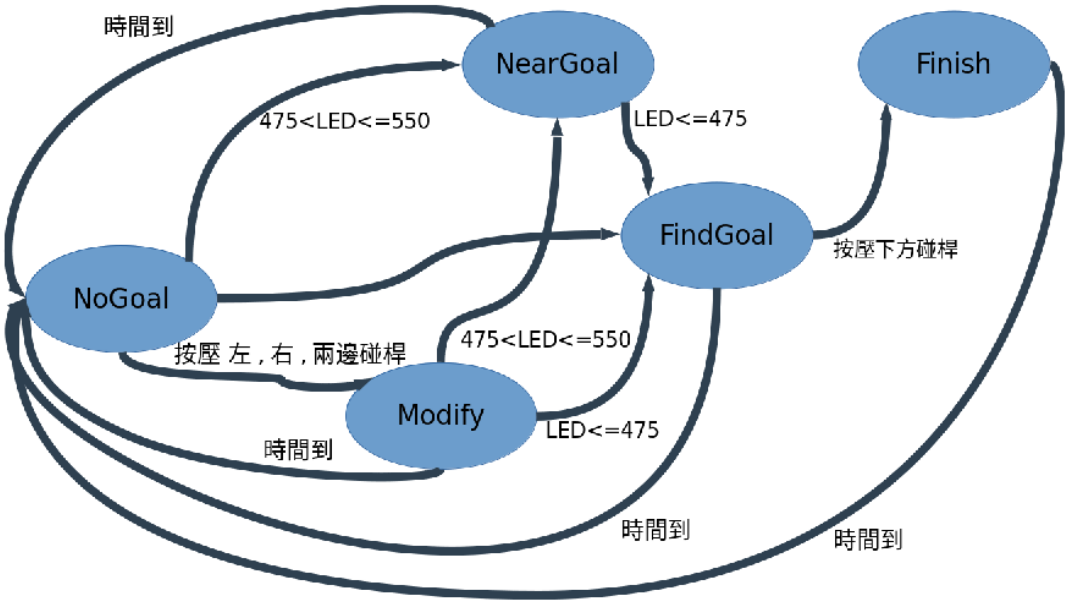
在Arduino端讀取類比輸入後，透過ROS將訊息發布至ROS topic，讓樹莓派端程式接收。

`pinMode(A0, INPUT); //在setup函式中，設定該pin為輸入腳位`

`led.data = analogRead(A0); //在loop函式中，讀取類比輸入值，儲存於Int32格式之msg的data中。`

數值範圍0~1023，光線越亮則數值越低。

有限狀態機



依據感測器數值和時間決定狀態。

Timer 計時中斷

使用rospy.Timer函式，每0.1秒會執行一次中斷函式(self.control_loop)，其內容包含：依據感測器數值更新狀態，以及依據狀態進行對應的馬達控制。

`self.timer = rospy.Timer(rospy.Duration(0.1), self.control_loop)`

程式碼

RPI

```
#!/usr/bin/env python
import wiringpi
import time
import random
import rospy
from std_msgs.msg import Int32
```

```
from std_msgs.msg import Int32


class Control(object):
    def __init__(self):
        wiringpi.wiringPiSetup()
        wiringpi.pinMode(7, 0)
        wiringpi.pinMode(8, 0)
        wiringpi.pinMode(9, 0)

        self.led=-1000
        self.last_led=-1000
        self.last_action=None
        self.now_action=None
        self.state="NoGoal"
        self.time=0

        self.sub_led = rospy.Subscriber("pub_led", Int32, self.cb_led, queue_size=1)
        self.pub_int = rospy.Publisher("array", Int32, queue_size=1)
        self.timer = rospy.Timer(rospy.Duration(0.1), self.control_loop)

        print("init done")


    def cb_led(self,msg):
        self.led=msg.data



    def motor_control(self):
        if self.now_action=="advance":
            self.pub_int.publish(1)
            print('~~~pub advance~~~')
        if self.now_action=="right":
            self.pub_int.publish(2)
            print('~~~pub right~~~')
        if self.now_action=="left":
            self.pub_int.publish(3)
            print('~~~pub left~~~')
        if self.now_action=="back":
            self.pub_int.publish(4)
            print('~~~pub back~~~')
        if self.now_action=="stop":
            self.pub_int.publish(5)
            print('~~~pub stop~~~')




    def control_loop(self,event):
        #for bump sensor
        my_input7 = wiringpi.digitalRead(7)
        my_input8 = wiringpi.digitalRead(8)
        my_input9 = wiringpi.digitalRead(9)

        if my_input7==0 and my_input8==0 and my_input9==1: #hit
            self.now_action="back"
            self.next_action="left"
            self.state="Modify"
            self.time=20

        if my_input7==1 and my_input8==0 and my_input9==1: #hit
            self.now_action="back"
            self.next_action="left"
            self.state="Modify"
            self.time=20

        if my_input7==0 and my_input8==1 and my_input9==1: #hit
            self.now_action="back"
            self.next_action="right"
            self.state="Modify"
            self.time=20

        if my_input7==1 and my_input8==1 and my_input9==1: #free

            #for LED sensor
            if self.led<=475:
                print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!FindGoal!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1!!!!")
                self.state="FindGoal"
                self.time=10

            elif self.led<=550 and self.led>475 and self.state!="Modify":
                print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~NearGoal~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
                if self.state!="NearGoal":
                    self.time=40
                    self.now_action="left"
```

```
                    self.next_action="right"
                self.state="NearGoal"




        if my_input9==0 : #have catched the ball
            self.state="Finish"



        #update
        if self.state=="Modify":
            if self.time==14:
                self.now_action=self.next_action
                self.next_action=None
            if self.time<0:
                self.state="NoGoal"
                self.time=0



        if self.state=="NearGoal":
            if self.time==30:
                self.now_action=self.next_action
                self.next_action="left"
            if self.time==10:
                self.now_action=self.next_action
                self.next_action=None
            if self.time<0:
                self.state="NoGoal"
                self.time=0


        if self.state=="FindGoal":
            self.now_action="advance"
            if self.time<0:
                self.state="NoGoal"
                self.time=0

        if self.state=="NoGoal":
            self.now_action="advance"

        if self.state=="Finish":
            self.now_action="stop"




        self.time=self.time-1
        self.last_led=self.led
        self.last_action=self.now_action
        self.motor_control()
        print(self.led,my_input7,my_input8,my_input9,self.state,self.now_action,self.time)




if __name__ == "__main__":
    rospy.init_node("Control")
    control=Control()
    rospy.spin()
```

Arduino

```
 #include <PID_v1.h>
#include <ros.h>
#include <math.h>
#include <std_msgs/Int32.h>
//#include <std_msgs/Float32MultiArray.h>
const byte encoder0pinA = 2;//A pin -> the interrupt pin 0
const byte encoder0pinB = 12;//B pin -> the digital pin 3
int in1 =8; //The enabling of L298PDC motor driver board connection to the digital interface port 5
int in2 =9; //The enabling of L298PDC motor driver board connection to the digital interface port 4
int ena =5;
const byte encoder1pinA = 3;//A pin -> the interrupt pin 0
const byte encoder1pinB = 13;//B pin -> the digital pin 3
int in3 =10; //The enabling of L298PDC motor driver board connection to the digital interface port 5
int in4 =11; //The enabling of L298PDC motor driver board connection to the digital interface port 4
int enb =6;
byte encoder0PinALast;
byte encoder1PinALast;
double durationright,abs_durationright;//the number of the pulses
```

```
double durationleft,abs_durationleft;
boolean Directionright;//the rotation direction
boolean Directionleft;
boolean resultright;
boolean resultleft;
ros::NodeHandle nh;
//int count=1;
double val_outputright;//Power supplied to the motor PWM value.
double val_outputleft;
double Setpointright=0;
double Setpointleft=0;
double Kp=0.6, Ki=5, Kd=0;

int MODE;
int count=0;



PID rightPID(&abs_durationright, &val_outputright, &Setpointright, Kp, Ki, Kd, DIRECT);
PID leftPID(&abs_durationleft, &val_outputleft, &Setpointleft, Kp, Ki, Kd, DIRECT);


void num(const std_msgs::Int32& msg){

   if (msg.data==1){
      MODE=1;
      Setpointleft=255;
      Setpointright=255;
   }
   if (msg.data==2){
      MODE=2;
      Setpointleft=100;
      Setpointright=100;
   }
   if (msg.data==3){
      MODE=3;
      Setpointleft=100;
      Setpointright=100;
   }
   if (msg.data==4){
      MODE=4;
      Setpointleft=255;
      Setpointright=255;
   }
   if (msg.data==5){
      MODE=5;
      Setpointleft=0;
      Setpointright=0;
   }
}
ros::Subscriber<std_msgs::Int32> sub("array",&num);
std_msgs::Int32 led;
ros::Publisher pub_led("pub_led", &led);


void setup()
{
   Serial.begin(57600);//Initialize the serial port
   pinMode(in1, OUTPUT);   //L298P Control port settings DC motor driver board for the output mode
   pinMode(in2, OUTPUT);
   pinMode(ena, OUTPUT);
   pinMode(in3, OUTPUT);   //L298P Control port settings DC motor driver board for the output mode
   pinMode(in4, OUTPUT);
   pinMode(enb, OUTPUT);
   pinMode(encoder0pinA, INPUT);
   pinMode(encoder0pinB, INPUT);
   pinMode(encoder1pinA, INPUT);
   pinMode(encoder1pinB, INPUT);
   nh.initNode();
   nh.subscribe(sub);
   rightPID.SetMode(AUTOMATIC);//PID is set to automatic mode
   rightPID.SetSampleTime(100);//Set PID sampling frequency is 100ms
   leftPID.SetMode(AUTOMATIC);//PID is set to automatic mode
   leftPID.SetSampleTime(100);//S
   EncoderInit();//Initialize the module


   pinMode(A0, INPUT);//analog input
   nh.advertise(pub_led);


   Serial.print("INIT DONE");
}
```

```
void loop()
{
      led.data = analogRead(A0);
      pub_led.publish( &led );

      if(MODE==1){advance();}//Motor Forward
      if(MODE==2){right();}
      if(MODE==3){left();}
      if(MODE==4){back();}
      if(MODE==5){Stop();}
      abs_durationright=abs(durationright);
      resultright=rightPID.Compute();//PID conversion is complete and returns 1
      if(resultright)
      {
        durationright = 0; //Count clear, wait for the next count
      }

      abs_durationleft=abs(durationleft);
      resultleft=leftPID.Compute();//PID conversion is complete and returns 1
      if(resultleft)
      {
        durationleft = 0; //Count clear, wait for the next count
      }

      nh.spinOnce();
}



void EncoderInit()
{
  Directionright = true;//default -> Forward
  pinMode(encoder0pinB,INPUT);
  attachInterrupt(0, wheelSpeedright, CHANGE);
  Directionleft = true;//default -> Forward
  pinMode(encoder1pinB,INPUT);
  attachInterrupt(1, wheelSpeedleft, CHANGE);
}
void wheelSpeedright()
{
  int Lstate = digitalRead(encoder0pinA);
  if((encoder0PinALast == LOW) && Lstate==HIGH)
  {
    int val = digitalRead(encoder0pinB);
    if(val == LOW && Directionright)
    {
      Directionright = false; //Reverse
    }
    else if(val == HIGH && !Directionright)
    {
      Directionright = true;  //Forward
    }
  }
  encoder0PinALast = Lstate;
  if(!Directionright)  durationright++;
  else  durationright--;
}
void wheelSpeedleft()
{
  int Rstate = digitalRead(encoder1pinA);
  if((encoder1PinALast == LOW) && Rstate==HIGH)
  {
    int valR = digitalRead(encoder1pinB);
    if(valR == LOW && Directionleft)
    {
      Directionleft = false; //Reverse
    }
    else if(valR == HIGH && !Directionleft)
    {
      Directionleft = true;  //Forward
    }
  }
  encoder1PinALast = Rstate;
  if(!Directionleft)  durationleft++;
  else  durationleft--;
}

void advance()//Motor Forward
{
     digitalWrite(in1,HIGH);
     digitalWrite(in2,LOW);
     analogWrite(ena,val_outputright);
     digitalWrite(in3,HIGH);
     digitalWrite(in4,LOW);
     analogWrite(enb,val_outputleft);
}
```

```
}
void right()//Motor Forward
{
     digitalWrite(in1,HIGH);
     digitalWrite(in2,LOW);
     analogWrite(ena,val_outputright);
     digitalWrite(in3,LOW);
     digitalWrite(in4,HIGH);
     analogWrite(enb,val_outputleft);
}
void left()//Motor Forward
{
     digitalWrite(in1,LOW);
     digitalWrite(in2,HIGH);
     analogWrite(ena,val_outputright);
     digitalWrite(in3,HIGH);
     digitalWrite(in4,LOW);
     analogWrite(enb,val_outputleft);
}

void back()//Motor reverse
{
     digitalWrite(in1,LOW);
     digitalWrite(in2,HIGH);
     analogWrite(ena,val_outputright);
     digitalWrite(in3,LOW);
     digitalWrite(in4,HIGH);
     analogWrite(enb,val_outputleft);
}

void Stop()//Motor stops
{
     digitalWrite(ena, LOW);
     digitalWrite(enb, LOW);
}
```

問題討論

實際使用光敏電阻後發現，它並不是最好的感測器。因為它不具有方向性，僅能參考類比輸入值判斷是否面對光球，然而當車子正對光球時，不一定能得到最低的數值。面對球的哪一面，地板和牆面反光都會有影響。完成任務需要一點運氣。