```
#include <RedBot.h>
#include<RedBotSoftwareSerial.h>

//define and include libraries
#include <Arduino.h>
#include <Redbot.h>
#define IR_SMALLD_NECx
#include<IRsmallDecoder.h>
//-----------------------------------
------------------
//define variables/pins

//IR sensor & remote
IRsmallDecoder irDecoder(2); //IR sensor
on DP2
irSmallD_t irData;

//IR line followers & values
RedBotSensor IRleft = RedBotSensor(A0);
int valIRleft;
RedBotSensor IRcent = RedBotSensor(A1);
int valIRcent;
RedBotSensor IRright = RedBotSensor(A2);
int valIRright;
```

```
#define LINETHRESHOLD 990 //IR Value
threshold for white vs black line, needs
tuning


//motor variables
int speed;
int AIN1 = 13; //motor1 left
int AIN2 = 12;
int PWMA1 = 11;
int AIN3 = 8; //motor 2 right
int AIN4 = 9;
int PWMA2 = 10;

//ultrasonic sensor variables
int trigPin = 7;
int echoPin = 6;
float distance;

int controlState; // variable for the
remote buttons pressed

void setup() {
  Serial.begin(9600);
```

```
  //Input-Output for ultrasonic sensor
 pinMode(trigPin,OUTPUT);
 pinMode(echoPin,INPUT);
}

void loop() {
  //read keypress and pass to keypress
handler function
 while(irDecoder.dataAvailable(irData)){
    int keypress = irData.cmd;
    Serial.println(keypress);
    handleKeyPress(keypress);
  }
  //Various states based on the last
button pressed
  //will repeat until a new button is
pressed or idle
  if(controlState == 0){ //
    idle();
  }
  else if(controlState == 1){
    GoLeft();
  }
  else if(controlState == 2){
```

```
    GoForward();
  }
  else if(controlState == 3){
    GoRight();
  }
  else if(controlState == 4){
    Avoid();
    Serial.print(dist());
    Serial.println();
  }
  else if(controlState == 7){
    LineSens();
    //Print IR Values for tuning
    Serial.print(IRleft.read());
    Serial.print("\t");
    Serial.print(IRcent.read());
    Serial.print("\t");
    Serial.print(IRright.read());
    Serial.println();
  }
 delay(100); //delay - can be changed for
sensitivity
}
```

```
float dist(){
  float echoTime; //variable to store the
time it takes for a ping to bounce off an
object
  float 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 / 148.0;
//calculate the distance of the object
that reflected the pulse
  // (half the bounce time multiplied by
the speed of sound)
  return calculatedDistance;
}
void idle()
```

```
{
  //left motor stop
 digitalWrite(AIN1,LOW);
 digitalWrite(AIN2,LOW);
 analogWrite(PWMA1,0);
  //right motor stop
 digitalWrite(AIN3,LOW);
 digitalWrite(AIN4,LOW);
 analogWrite(PWMA2,0);
}
void GoForward()
{
  //left motor forward
 digitalWrite(AIN1,LOW);
 digitalWrite(AIN2,HIGH);
 analogWrite(PWMA1,120);
  //right motor forward
 digitalWrite(AIN3,LOW);
 digitalWrite(AIN4,HIGH);
 analogWrite(PWMA2,120);
}
void GoBack()
{
  //left motor forward
```

```
 digitalWrite(AIN1,HIGH);
 digitalWrite(AIN2,LOW);
 analogWrite(PWMA1,150);
  //right motor forward
 digitalWrite(AIN3,HIGH);
 digitalWrite(AIN4,LOW);
 analogWrite(PWMA2,150);
}
void GoLeft()
{
  //left motor backward
 digitalWrite(AIN1,LOW);
 digitalWrite(AIN2,HIGH);
 analogWrite(PWMA1,100);
  //right motor forward
 digitalWrite(AIN3,HIGH);
 digitalWrite(AIN4,LOW);
 analogWrite(PWMA2,100);
}
void GoRight()
{
  //left motor forward
 digitalWrite(AIN1,HIGH);
 digitalWrite(AIN2,LOW);
```

```
 analogWrite(PWMA1,100);
  //right motor backward
 digitalWrite(AIN3,LOW);
 digitalWrite(AIN4,HIGH);
 analogWrite(PWMA2,100);
}

float Avoid(){
  GoForward(); //Forward until object is
detected
  float distance;
  distance = dist();
  if(distance < 8){ //If an object is
detected change directions
    GoBack();
    delay(500);
    GoRight();
    delay(500);
  }
}

void LineSens()
{
  if(IRcent.read() > LINETHRESHOLD)
```

```
  {
    GoForward(); //motors forward if line
is under center sensor
  }
  else if(IRright.read() > LINETHRESHOLD)
  {
    GoRight(); // turn motors right if
  }
  else if(IRleft.read() > LINETHRESHOLD)
  {
    GoLeft(); //turn the motor left if
line is under black
  }
  if((IRleft.read() > LINETHRESHOLD) &&
(IRcent.read() > LINETHRESHOLD) &&
(IRright.read() > LINETHRESHOLD))
  {
    GoBack(); //stop motor if all sensors
detect black
  }
}
void handleKeyPress(int keyPress) //read
keypress and determine operation
{
```

```
  if(keyPress == 12) {
    controlState = 0; //button 0 pressed
  }

  if(keyPress == 16) {
    controlState = 1; //button 1 pressed
  }

  if(keyPress == 17) {
    controlState = 2; //button 2 pressed
  }

  if(keyPress == 18) {
    controlState = 3; //key 3 pressed
  }

  if(keyPress == 20) {
    controlState = 4; //key 4 pressed
  }
  if(keyPress == 24) {
    controlState = 7; //button 7 pressed
  }
}
```