Arduino Project

Yash Patel (yp2285)
Anais Roche (aer523)
Rushi Shah (rs7236)

Methodology

Hardware Setup

Two continuous servo motors are used for the maneuvering of the Boebot rover. The inputs range from 0 to 180 where a value around 90 stops the Boebot. Values from 90-180 will turn the continuous servo in one direction. Values from 0-90 will turn the servo in the opposite direction. For the standard servo, the values 0-180 represent degrees rather than speed.

The PING ultrasonic sensor first outputs out a high pulse for a few microseconds before it switches to an input. The duration which the ultrasonic detects is then converted into cm. A low pulse is set before the high pulse for the clearest reading. After the high pulse is outputted, the pin is set to low again to reset.

QTR-HD-15A Reflectance Sensor Array is an array of 15 IR LED and phototransistor pairs with analog inputs. Six pairs were selected as the line being tracked does not require all sensors to be reading. The input values from the phototransistors are used to determine if there is a line or not. The higher the value read by the sensor, the more likely a line is detected. The servo actuation is determined based on these readings accordingly.

Circuit

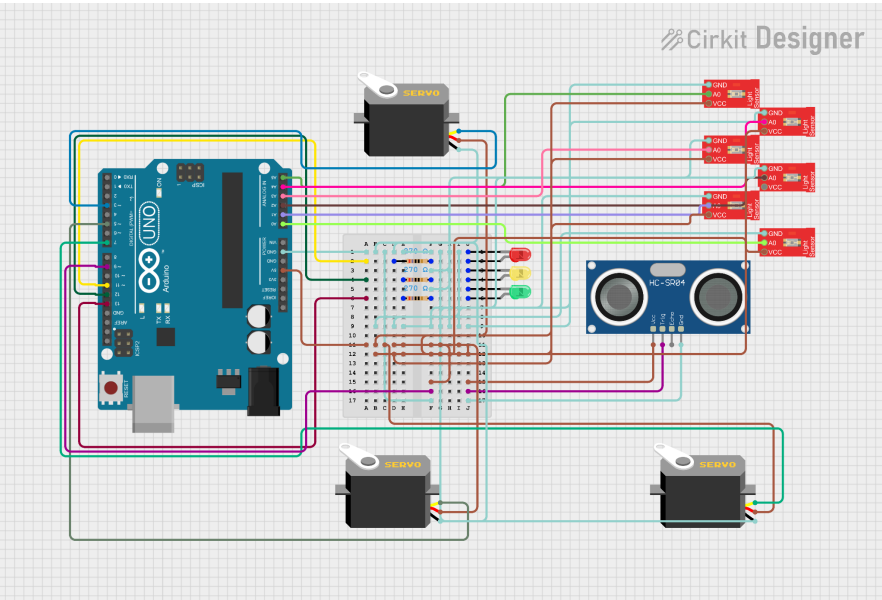

Fig 1: circuit diagram

Corrections to the circuit diagram: Due to the Cirkit Designer's limitations, there were no QTR-HD-15 Analog sensor models available so six light sensors are used in substitute for the six sensors used on the QTR sensor. The ultrasonic sensor is also a substitute for the PING sensor which only has three connections: GND, VCC, and SIG instead of the one shown in the diagram. Two continuous servo motors are connected to pin 3 and 5 for the wheels of the rover while the third servo connected to pin 7 is a standard servo used for the PING sensor. The PING sensor is connected to pin 9. The QTR analog sensor is connected to pins A0-A6. The three LEDs are connected to pins 11, 12, and 13. Although there is already a built in LED at pin 13, the green led is mounted with the rest of the LEDs for better visibility.

Line tracking using QTR sensors

Of the 15 pairs of IR-phototransistor pairs, only 6 are stored in the array sensorValues. Two pairs on the far ends of the sensor and four pairs in the center. To track lines we run a while loop that reads the sensor values of all six. As shown in figure 2, IR3 and IR2 are used for line following.

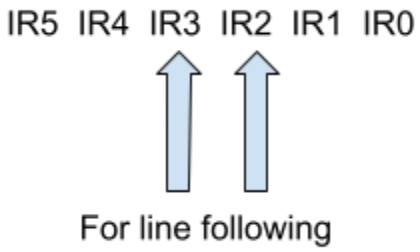

Fig 2: IR Array

Line Following Pseudo code

```
if (IR2 > 220 && IR3 > 220)
{
  Forward motion
}

else  if (IR2>220 && IR3<130)
{
   Turn right
}

else if (IR3>220 && IR2<130)
{
   Turn left
}
```

Fig 3: Line Follower - where 220 is a threshold value selected based on experimental readings of the sensor. When the sensor detects values less than 220, the sensor is reading a white background and above 220 means a black line has been detected. The sensor reading fluctuates around 40-90 due to lighting conditions and 220 was selected to ensure that there are no misreadings.

Intersection detection

An intersection is detected when all the sensors on the sensor array detect black as well as the outermost sensors on the array. When this condition is met, the line tracker function terminates and the robot is brought to a stop. If it is the first intersection, a red LED will blink once and will not search for the objects using the ultrasonicSweep function. The subsequent intersections will cause a yellow LED to blink once to indicate that an intersection has been found and then it will initiate the ultrasonicSweep function.

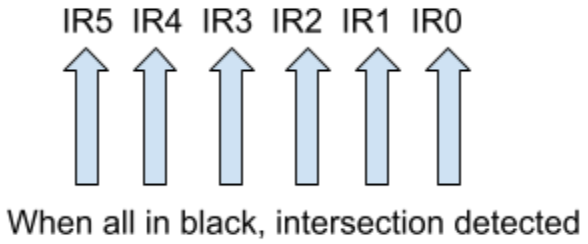

Fig 3: Intersection Condition- At an intersection, the entire array should be detecting a black line rather than just the center four sensors. IR0 and IR5 are positioned at the extremes of the sensor array.

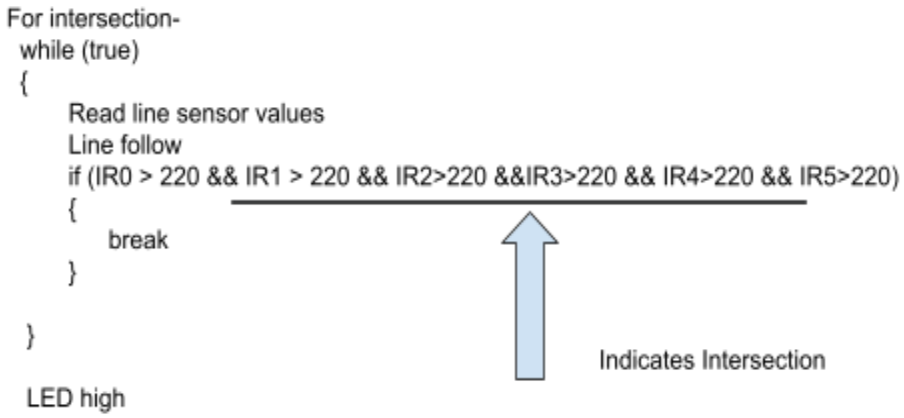

Fig 4: Intersection Condition - the sensor requires a reading of above 220 on all six sensors of the array in order to determine if an intersection has been reached.

Object Detection at intersections

When the robot reaches an intersection, it is brought to a full stop and then the ultrasonicSweep function is called. This sweeps the ultrasonic sensor mounted on a standard servo motor in the front of the rover to the right (0 degrees), front (90 degrees) and finally to the left (180 degrees). The position sweep of the servo motor is shown in figure 4. The desired detection range is 30-50 centimeters. A variable for the predetermined location distance, det_loc, is initialized to 50. With every iteration of the PING sensor readings, the value is compared to det_loc and if an object is found within that range, then an object is considered detected. Because the ultrasonic sensor only reads in three directions at each intersection, this minimizes the chance

for the ultrasonic sensor to pick up other items that are not in its immediate vicinity. When an object is detected the robot indicates it using the green LED blinking once. If no item is found, the robot will move forward and loop back to line tracking until the next intersection is found.

Subsequently, if an item is found, a boolean is changed to indicate which side of the intersection has an item. If there is more than one item at the intersection, multiple boolean values will switch to true. The rover will first turn right and call the lineTracker function. The PING sensor will also take in values in a while function. As long as the ping function does not detect anything within 8 cm, the rover will continue to move forward. Once an item is found, it will stop, blink a green LED once, and then make a 180 degree turn and return to the intersection. If there is more than one item, it will go straight to the next item ahead and repeat the approach. If there is no item, it will return to the main path and loop for the next intersection.

In the unique case where there is an additional third item ahead, similarly, a boolean value will be set to 1 if the rover detects something during the ultrasonicSweep. After it completes the other left and right sweep, it will detect something ahead and stop when the distance reaches 8cm before turning around and returning to the intersection.

At this stage, all items should have been detected and approached by the rover and the task is complete.

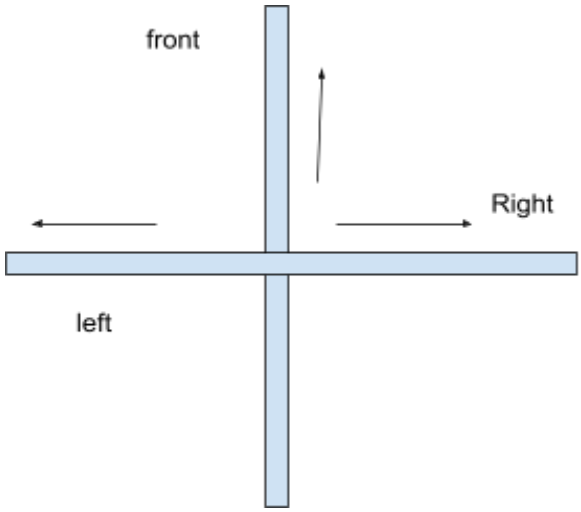

Fig 5: Servo sweep directions where right is 0 degrees, front is 90 degrees, and left is 180 degrees.

Object detection pseudo code

```
distance_threshold = 52;

// detect object at RIGHT
Servo position 0 degree
If (distance  from sensor < distance_threshold)
{
  Object  at right side
  greenLED HIGH
  delay
  greenLED LOW
}

// detect object at FRONT
Servo position 90 degree
If (distance  from sensor < distance_threshold)
{
  Object at front side
  greenLED HIGH
  delay
  greenLED LOW
}

// detect object at left
Servo position 180 degree
If (distance  from sensor < distance_threshold)
{
  Object  at left side
  greenLED HIGH
  delay
  greenLED LOW
}
```

Fig 6: object detection code

Robot Navigation code

```
//no object detected
if (left ==0 && right == 0)
 {
  Move forward
 }

//object  on the right
 if (right==1)
 {
   Turn right
   Follow line
   Stop robot before distance threshold
   Detect object
   Turn 180 degree
    Line follow
    Proceed for left if ==1 or take right turn
 }

//object  on the left
 if (left==1)
 {
   Turn left
   Follow line
   Stop robot before distance threshold
   Detect object
   Turn 180 degree
   Line follow
   Turn  left
    Move to further intersections
 }

//object  on the front
 if (front==1)
 {
   Follow line
   Stop robot before distance threshold
   Detect object
  Turn 180 degree
   Stop the  robot
 }
```

Fig 7: Navigation - Three cases for when there are objects (or no objects) at each intersection. The case where there is an object in front is for the unique case at the end of the track.

Appendix:
Project Code

```
#include<Servo.h>
#include <QTRSensors.h>

Servo servo1, servo2, servo3;  //defining three servo motor

const uint8_t SensorCount = 6; //selecting a number of sensors
uint16_t sensorValues[SensorCount];
int right = 0;                  //Counter used for detecting an object on right
int left = 0;                   //Counter used for detecting an object on left
int straight = 0;               //Counter used for detecting an object on the straight path
const int pingPin = 9;          // Assigning Arduino pin 9 to the Ping Sensor
int object_count = 0;           // Defining variable to calculate the number of objects
const int  redLED = 11;       // Assigning Arduino pin 11 to the Red LED
const int  greenLED = 13;   // Assigning Arduino pin 13 to the Green LED
const int yellowLED = 12;  // Assigning Arduino pin 12 to the Red LED


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

   //configure the sensors
  qtr.setTypeAnalog();
  qtr.setSensorPins((const uint8_t[]){A0, A1, A2, A3, A4, A5}, SensorCount);
 calibrate();

  //attach servos to pins
  servo1.attach(3);
  servo2.attach(5);
  servo3.attach(6);

  // Initialize with speed 0 and angle 0
  servo1.write(94);
  servo2.write(93);
  servo3.write(90);

  //pin configuration
```

```
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);

  //first intersection
  intersect();
  servo1.write(94);
  servo2.write(93);
  digitalWrite(redLED, HIGH);

  delay(3000);
  digitalWrite(redLED, LOW);
  forward();

}
void loop()
{
  intersect();
  servo1.write(94);
  servo2.write(93);
  digitalWrite(redLED, HIGH);
  delay(3000);
  digitalWrite(redLED, LOW);

  int object_count = ultrasonicSweep();
  delay(1000);

}

//ULTRASONIC SWEEP
int ultrasonicSweep()
{
  long det_loc = 52;

  // Detect object at RIGHT
  servo3.write(0);
  delay(2000);
  long loc_comp = ping();
  if (loc_comp < det_loc)
  {
```

```
  right = 1;
  object_count = object_count + 1;
  digitalWrite(greenLED, HIGH);
  delay(2000);
  digitalWrite(greenLED, LOW);

}

// detect object at FRONT
servo3.write(90);
delay(2000);
loc_comp = ping();
if (loc_comp < det_loc)
{
  straight = 1;
  object_count = object_count + 1;
  digitalWrite(greenLED, HIGH);
  delay(2000);
  digitalWrite(greenLED, LOW);
}

 // detect object at left
servo3.write(180);
delay(2000);
loc_comp = ping();
if (loc_comp < det_loc)
{
  left = 1;
  object_count = object_count + 1;
  digitalWrite(greenLED, HIGH);
  delay(2000);
  digitalWrite(greenLED, LOW);
}

//Servo3 Back to home position
servo3.write(90);
delay(2000);

if (left ==0 && right == 0)       //when no object is detected either on left or right side
{
```

```
  forward();
}

// Moving towards the object on right side
if (right==1)
{
  qtr.read(sensorValues);

  //turn right by 90, facing the object
  while(sensorValues[3]>200)
  {
   qtr.read(sensorValues);
   Serial.print(greeting);
   servo1.write(100);
   servo2.write(93);
  }

  while(sensorValues[3]<150)
  {
   qtr.read(sensorValues);
   Serial.print(greeting);
   servo1.write(100);
   servo2.write(93);
  }

  right = 0;

  //move towards the object upto 8 cm
  loc_comp = ping();
  while (loc_comp > 8)
  {
    loc_comp = ping();
    qtr.read(sensorValues);
    line_follow(sensorValues);
  }

  //stop the robot and indicate
  stop_bot();
  digitalWrite(greenLED, HIGH);
  delay(2000);
```

```
digitalWrite(greenLED, LOW);

//turn the ROBOT 180
while (sensorValues[0]<150)
{
  qtr.read(sensorValues);
  servo1.write(114);
  servo2.write(114);
}

while (sensorValues[0]>200)
{
  qtr.read(sensorValues);
  bot_rotate();
}

//move back to intersection
intersect();
stop_bot();

//Go towards object on the left
if (left == 1)
{
  left=0;
 //move towards the object upto 8 cm
  loc_comp = ping();
  while (loc_comp > 8)
  {
   loc_comp = ping();
   qtr.read(sensorValues);
   line_follow(sensorValues);
  }


  //stop the robot and indicate for object
  stop_bot();
  digitalWrite(greenLED, HIGH);
  delay(2000);
  digitalWrite(greenLED, LOW);
```

```
  //turn the ROBOT 180
  while (sensorValues[0]<150)
  {
   qtr.read(sensorValues);
   servo1.write(114);
   servo2.write(114);
  }

  while (sensorValues[0]>200)
  {
   qtr.read(sensorValues);
   bot_rotate();
  }

  //move back to intersection
  intersect();
  stop_bot();

  while(sensorValues[1]>200)
  {
     qtr.read(sensorValues);
     Serial.print(greeting);
     servo1.write(94);
     servo2.write(85);
  }

  while(sensorValues[1]<150)
  {
     qtr.read(sensorValues);
     Serial.print(greeting);
     servo1.write(94);
     servo2.write(85);
     Serial.print(greeting);
  }
}


else
{
 while(sensorValues[3]>200)
```

```
    {
       qtr.read(sensorValues);
       Serial.print(greeting);
       servo1.write(100);
       servo2.write(93);
    }

    while(sensorValues[3]<150)
    {
      qtr.read(sensorValues);
      Serial.print(greeting);
      servo1.write(100);
      servo2.write(93);
    }

    intersect();
    stop_bot();

   }

}

else if(left == 1 && right == 0)
{
   //turn left by 90, facing the object
   Serial.print(greeting);
   while(sensorValues[1]>200)
   {
    qtr.read(sensorValues);
    Serial.print(greeting);
    servo1.write(94);
    servo2.write(85);
   }

   while(sensorValues[1]<10)
   {
    qtr.read(sensorValues);
    Serial.print(greeting);
    servo1.write(94);
    servo2.write(85);
```

```
 Serial.print(greeting);
}

left = 0;

//move towards the object upto 8 cm
loc_comp = ping();
while (loc_comp > 8)
{
  loc_comp = ping();
  qtr.read(sensorValues);
  line_follow(sensorValues);
}

//stop the robot and indicate
stop_bot();
digitalWrite(greenLED, HIGH);
delay(2000);
digitalWrite(greenLED, LOW);

                    //turn the ROBOT 180
while (sensorValues[0]<150)
{
  qtr.read(sensorValues);
  bot_rotate();
}

while (sensorValues[0]>200)
{
  qtr.read(sensorValues);
  bot_rotate();
}

//move back to intersection
intersect();
stop_bot();

//turn left by 90, facing the object
while(sensorValues[1]>200)
{
```

```
      qtr.read(sensorValues);
      Serial.print(greeting);
      servo1.write(94);
      servo2.write(85);
     }

     while(sensorValues[1]<130)
     {
      qtr.read(sensorValues);
      Serial.print(greeting);
      servo1.write(94);
      servo2.write(85);
     }

     intersect();
     stop_bot();
     //forward();
  }

  if (straight == 1)
  {

//move towards the object upto 8 cm
     loc_comp = ping();
     while (loc_comp > 8)
     {
       loc_comp = ping();
       qtr.read(sensorValues);
       line_follow(sensorValues);
     }
     straight = 0;
     stop_bot();
     digitalWrite(greenLED, HIGH);
     delay(2000);
     digitalWrite(greenLED, LOW);

     //turn the ROBOT 180
     while (sensorValues[0]<150)
     {
       qtr.read(sensorValues);
```

```
      bot_rotate();
    }

    while (sensorValues[0]>200)
    {
      qtr.read(sensorValues);
      bot_rotate();
    }

    stop_bot();
    delay(10000);

  }

}

// LINE FOLLOW
void intersect()
{
  while (true)
  {
    qtr.read(sensorValues);

    line_follow(sensorValues);
    if (sensorValues[1] > 220 && sensorValues[4] > 220 && sensorValues[0]>220 &&
sensorValues[5]>220)
    {
      break;
    }

  }
}

void  line_follow(uint16_t sensorValues[6]){
  qtr.read(sensorValues);
  if (sensorValues[1] > 220 && sensorValues[3] > 220)
  {
    servo1.write(105);
    servo2.write(83);
    qtr.read(sensorValues);
```

```
  }
  if (sensorValues[1]>220 && sensorValues[3]<130)
  {
     servo1.write(100);  //slight_right
     servo2.write(93);
  }
  else if(sensorValues[3]>220 && sensorValues[1]<130)
  {
    servo1.write(94); //slight_left
    servo2.write(73);
  }
}

//FORWARD MOTION OF ROBOT
void forward()
{
  servo1.write(105);
  servo2.write(82);
  delay(500);
}

//STOP MOTION OF ROBOT
void stop_bot()
{
  servo1.write(94);
  servo2.write(93);
  delay(2000);
}

void bot_rotate()
{
  servo1.write(114);
  servo2.write(114);
}

//ULTRASONIC SENSOR READINGS
long ping(){
  long duration, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  cm = microsecondsToCentimeters(duration);

  //Serial.print(cm);
  Serial.println();

  delay(20);
  return cm;
}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29/ 2;
}
```