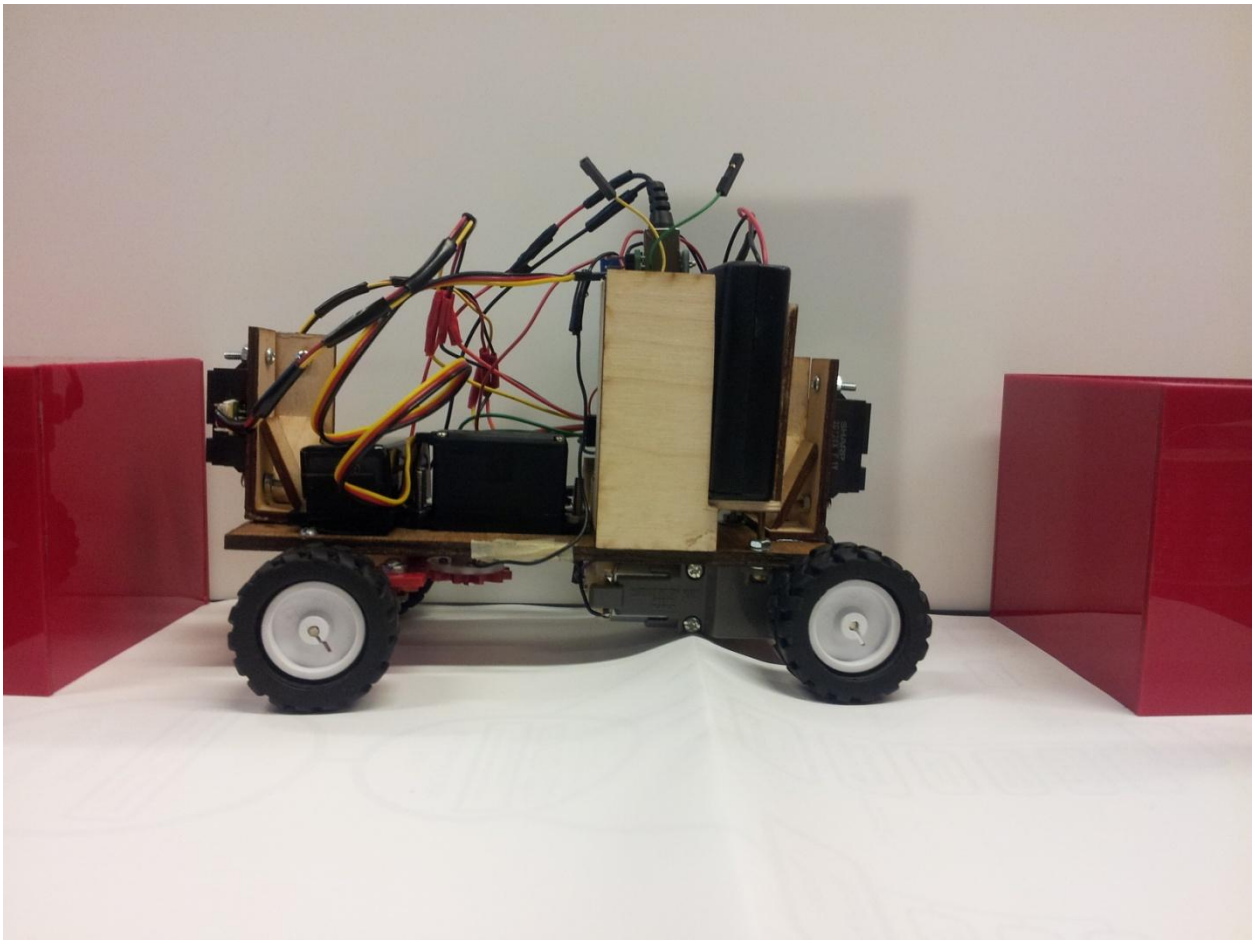
Automatic Parallel Parking Car

PH215-Mircocontrollers Projects in Physics



David Chen
Maggie Cao

Eric Forkosh

David Chen
Maggie Cao

Automatic Parallel Park Car

Objective: To wirelessly control the directions of the car and to parallel park hands-free between two objects.

It is important to control the car to right position before the parallel sequence starts because the car can be far away from the two objects and start parking at the wrong moment. Initially the car should be adjacent and parallel to the right of the front object before initialing the parking sequence. The Python code serially connects with the Xbee, that is directly connected to the laptop USB port. The Xbee transmitter, which transmits the specific keys pressed at that time, communicates to the Xbee receiver located on the car. The arrows keys controls the movement of the car. The "t" key will stop the car from its movement when it's controlled by python or during its parallel sequence. The "s" key will initiate the parallel sequence.

The mechanics and functionality of the car's parts facilitates parallel parking. Located on the back of the car, the two DC motors are connected in series so that the motor can move forward or backward. The front of the car is the most significant mechanism for parallel parking because it allows the car to steer at specific curvatures. The front wheel-steering system mechanism is organized two elements. One element is the servo, which rotates the gear no greater than 136 degrees (maximum right turn) and no less than 44 degrees (maximum left turn)or else the steering system will break and fail. The servo controls the front wheel steering. The second element is the rack and pinion mechanism. The pinion is connect right on wheel of the servo. The rack is guided by the pinion and allows the two wheels to move at various angles commanded by the servo.

In the parallel parking sequence, the car moves backwards until the back of the car is align with the back of the front object. When IR near the bottom right reads a valve greater than 12 cm, the car will stop. Then the front wheels steer right. The car then begins to move backward in a curve motion, until the back IR reads a value 19 cm away from the wall, then stop. In this process, the IR values tend to fluctuate. To compensate this problem, the distances values near 19 cm were average to be the 19cm threshold. The

front wheel rotates left and the car will move back in a curve motion. The car will stop if the following happens:

1. The IR near the bottom right of the car is too close near the wall.

2. The IR at the back detects the back object within a close range.

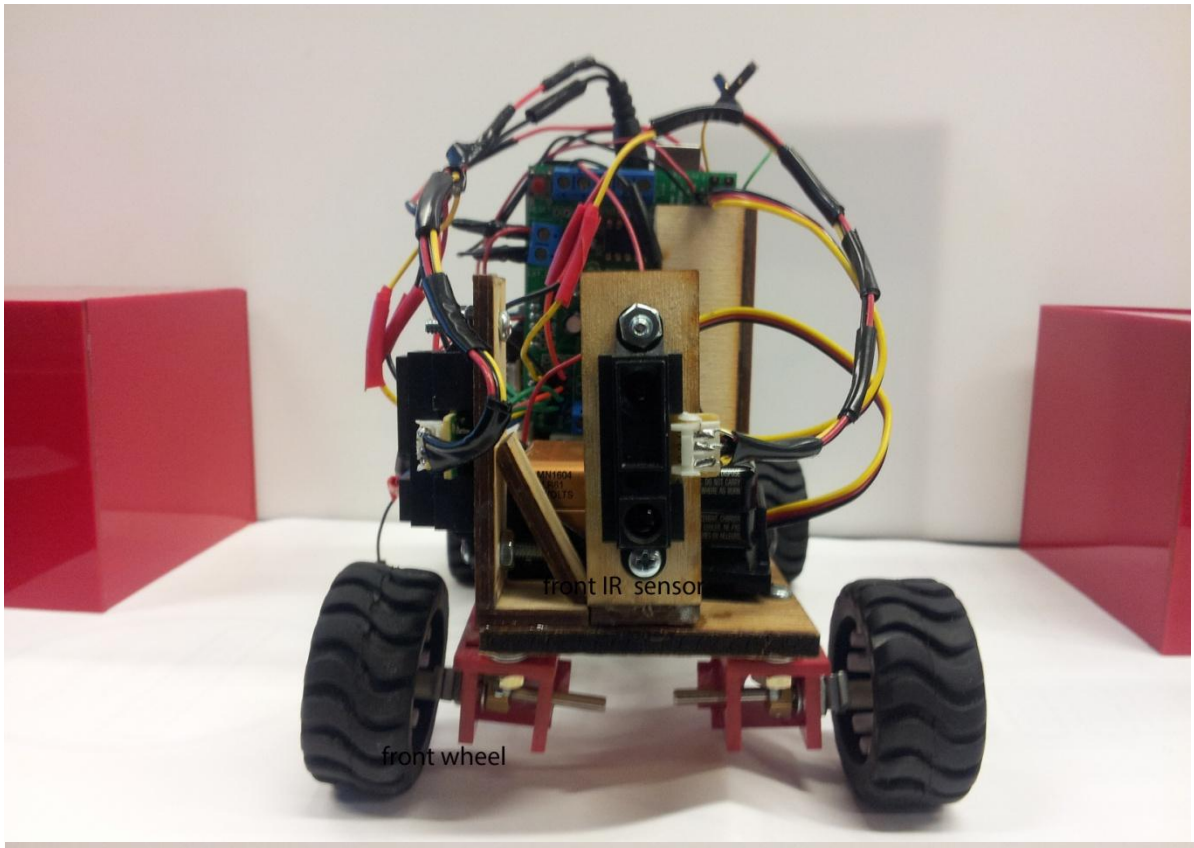
3.The IR near the top right and the IR near the bottom right reads the same value, which means the car is parallel to the wall.

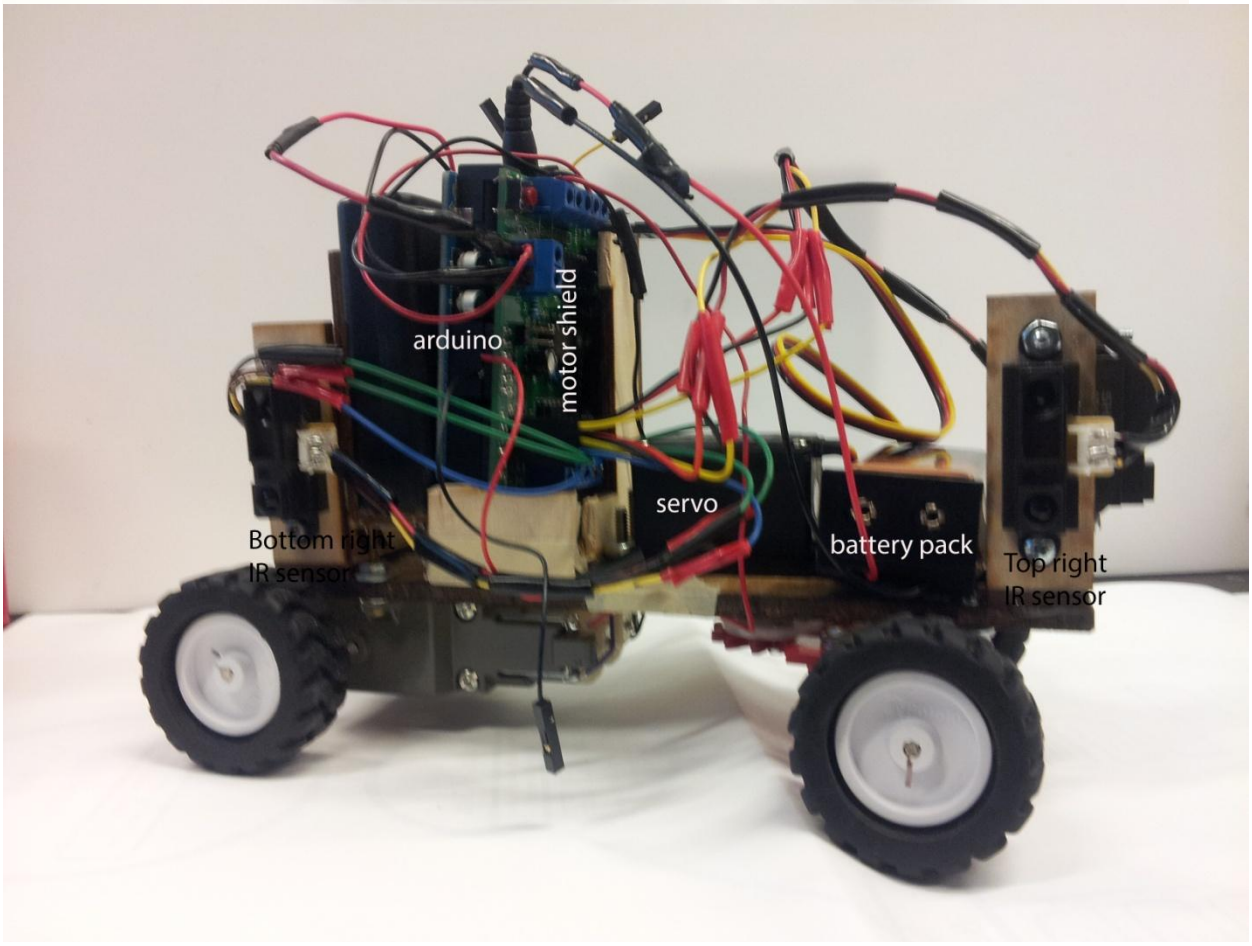
For conditions 1 and 2, the car will then steer right and move forward until the IR near the top right and the IR near the bottom right reads the same value, which means the car is parallel to the wall. After either of the 3 conditions, the car will move backward or forward until the car is equally space from the front and back object. Thus, parallel parking is complete.

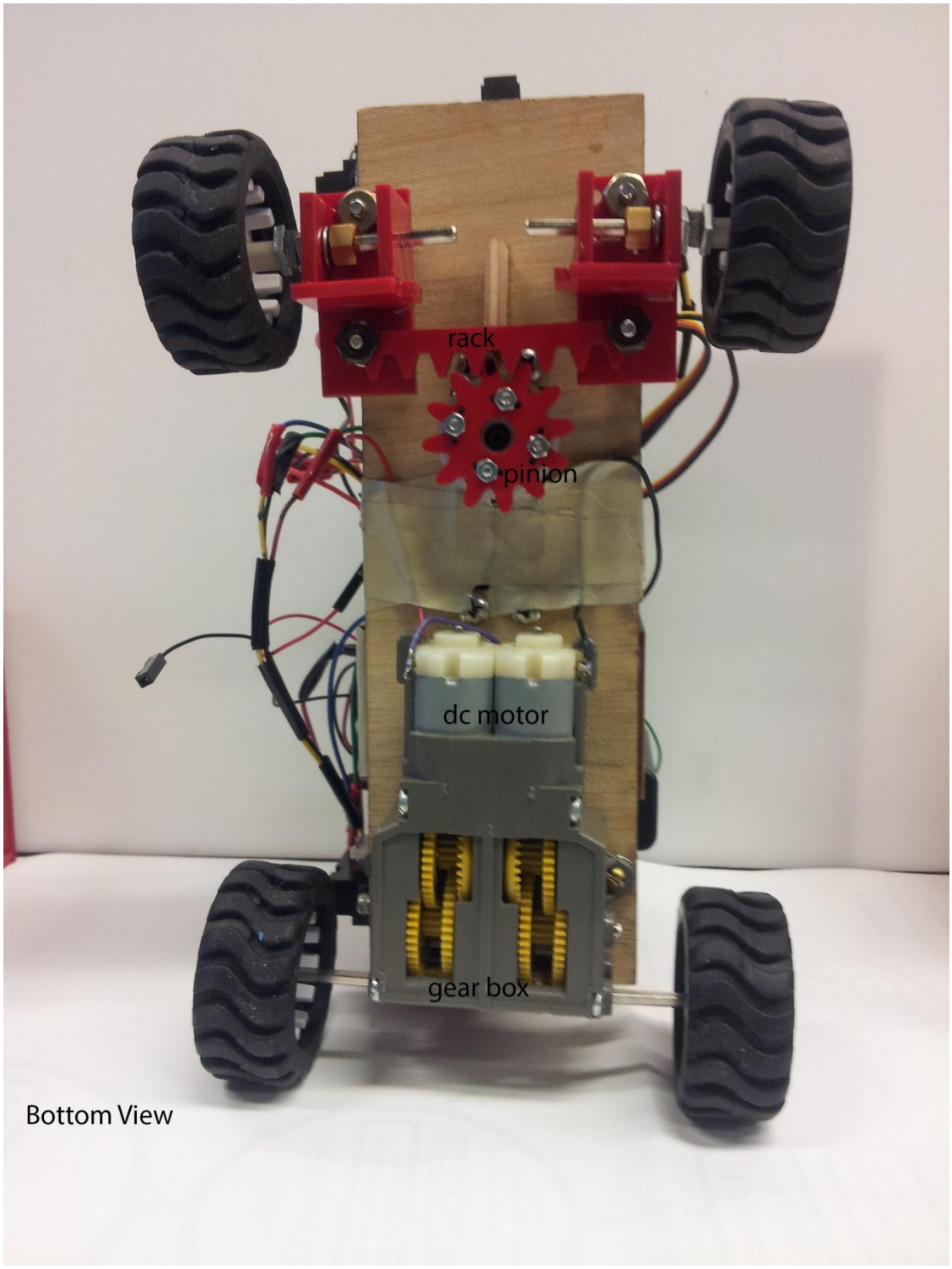
Parts List:

- Servo, Stepper, DC Motor Shield
- 180 degree Servo
- Red plexi
- Bass Wood
- 4 Infrared Proximity Sensor Short Range
- 2- 3 volt DC motors
- Gear box
- Xbee shield
- Xbee USB explorer
- 2 Xbee
- 4AA battery holder
- 9 Volt battery holder
- 4- 1.25 " diameter wheels

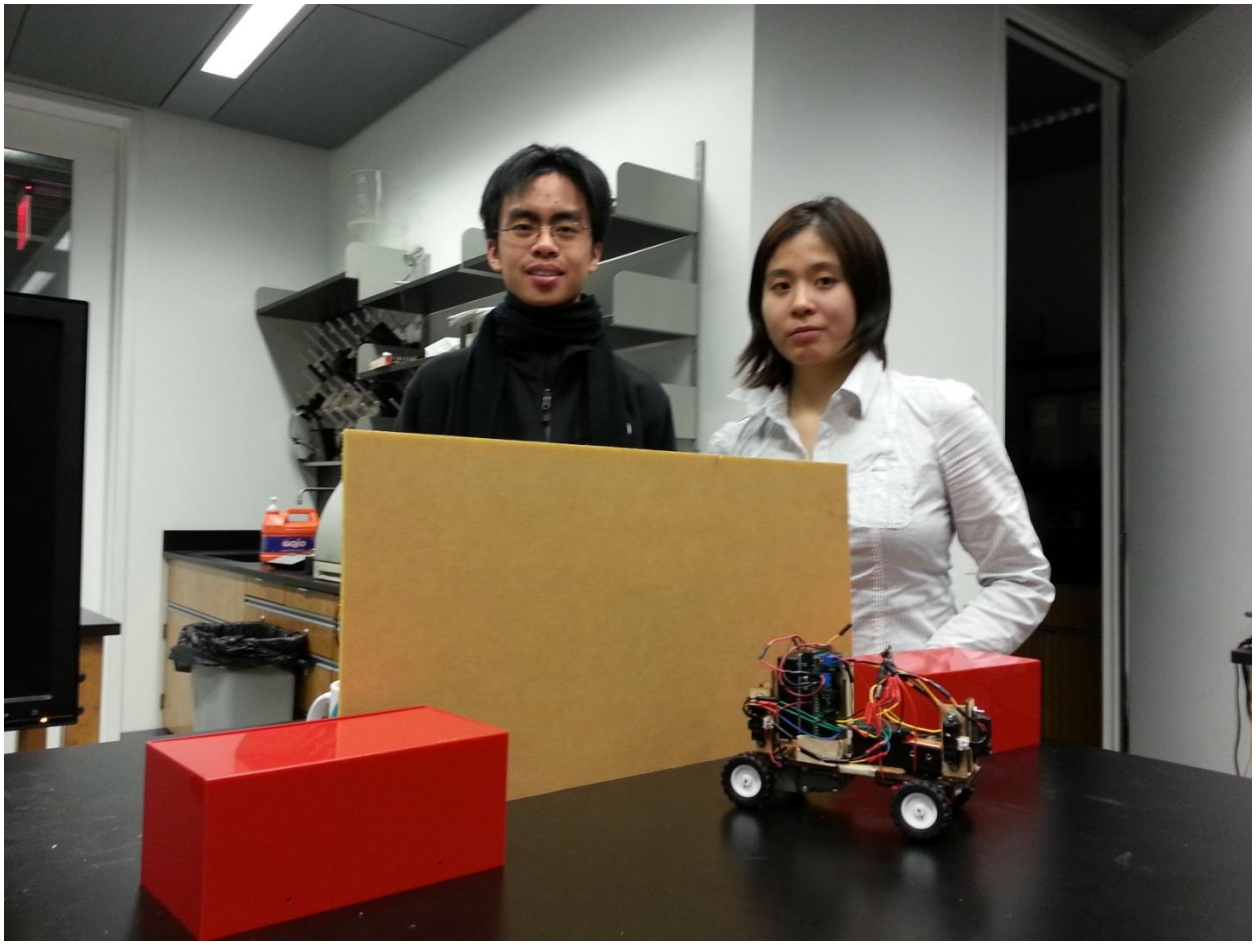
Pictures


front IR sensor
front wheel

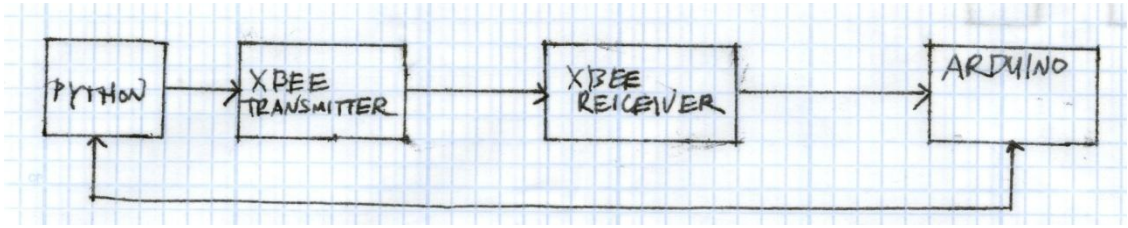

arduino
motor shield
servo
Bottom right
IR sensor
battery pack
Top right
IR sensor

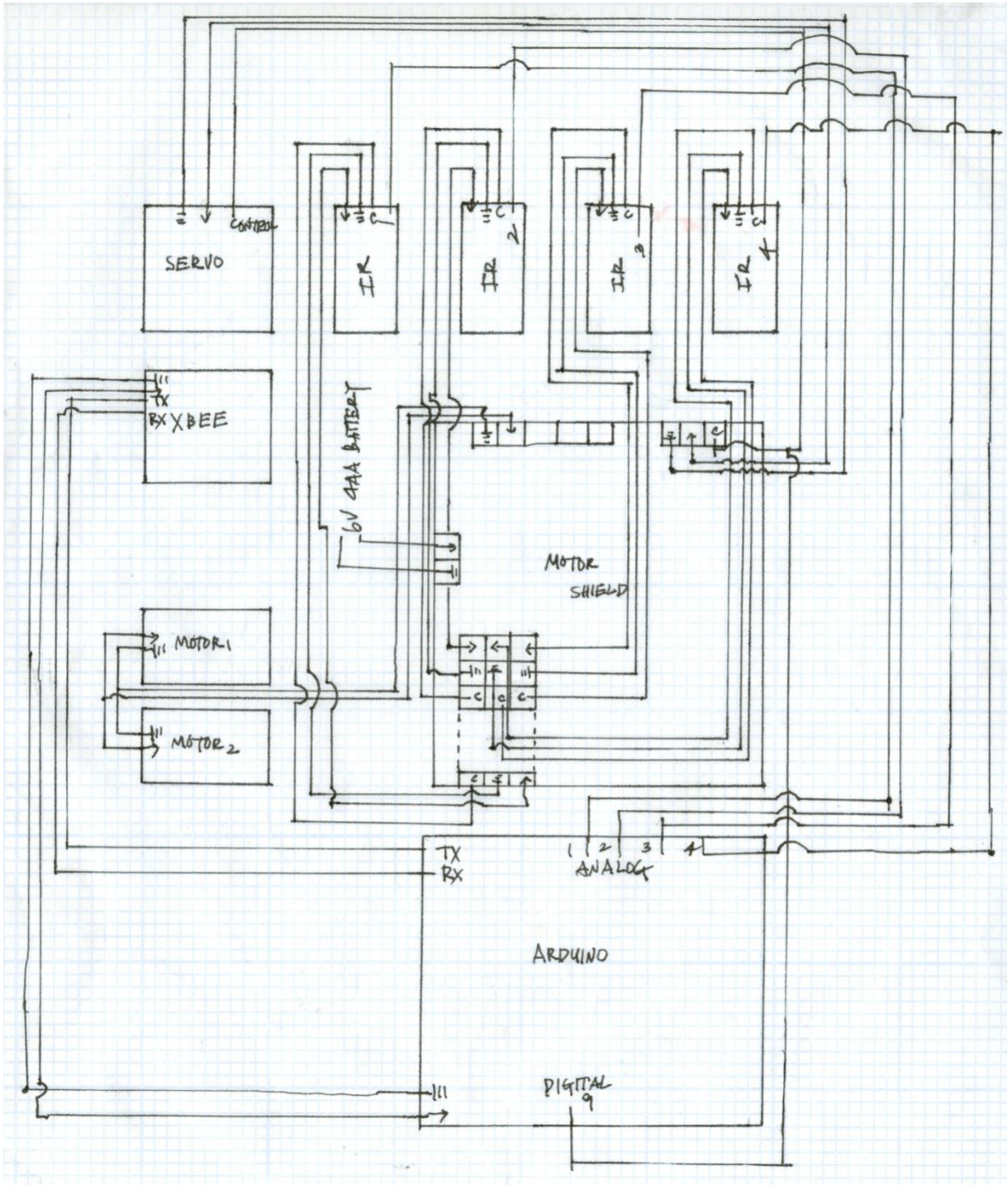


Bottom View



Schematic


PYTHON
XBEE TRANSMITTER
XBEE RECEIVER
ARDUINO


SERVO
CONTROL
IR
IR 2
IR 3
IR 4
TX
RX XBEE
6V AAA BATTERY
MOTOR SHIELD
MOTOR 1
MOTOR 2
TX
RX
ANALOG
1 2 3 4
ARDUINO
DIGITAL
9

The C Code:

```
#include <AFMotor.h> //This works with the motor shield that we got.

#include <Servo.h>


Servo myservo;

AF_DCMotor motor(2, MOTOR12_64KHZ); //Sets up motor on the motor shield.
String readString;

int IR_Front = A1;
int IR_Right_Front=A2;
int IR_Right_Back=A3;
int IR_Back=A4;

int volts_Front = analogRead(IR_Front);
int volts_Right_Front = analogRead(IR_Right_Front);
int volts_Right_Back = analogRead(IR_Right_Back);
int volts_Back = analogRead(IR_Back);

int distance_Front= (2914/(volts_Front +5))-1;
int distance_Right_Front= (2914/(volts_Right_Front +5))-1;
int distance_Right_Back= (2914/(volts_Right_Back +5))-1;
int distance_Back= (2914/(volts_Back +5))-1;


void setup() {
  Serial.begin(9600);
   myservo.attach(9);
     Straight;
    pinMode(IR_Front,INPUT);
     pinMode(IR_Right_Front,INPUT);
     pinMode(IR_Right_Back,INPUT);
     pinMode(IR_Back,INPUT);

     Stop(200);
     delay(1000);

}

void loop() {
  readString="";


    if ((Serial.available() >0)&&(readString != "l")&&(readString != "r")&&(readString != "u")&&(readString != "d")&&(readString != "s")
&&(readString != "t")) {
      char c = Serial.read();
      readString += c;
    }



  if (readString.length() >0)   //Sub-routines for some reason doesn't work
//  in the car-direction if-statements.
  {
    if(readString=="l") //Press left arrow key to steer left.
    {
    myservo.write(44);

    }
    if(readString=="r") //Press right arrow key to steer right.
    {
      Serial.println("right");
      myservo.write(136);

    }
    if(readString=="u") //Press up arrow to go forward.
```

```
  {
    Serial.println("up");
     motor.setSpeed(255);
 motor.run(FORWARD);
  }
  if(readString=="d") //Press down arrow key to go backward.
  {

    motor.setSpeed(200);
motor.run(BACKWARD);
  }
  if(readString=="s") //Parallel Parking Sequence starts when "s" is pressed.
  {
     Straight;  //Prepares the car to go backward,parallel to the adajacent object.
    while(distance_Right_Back != 0) //Car moves back until the
   // back of the car just lines up with the front object.
    {

     int volts_Right_Back = analogRead(IR_Right_Back);
     int distance_Right_Back= (2914/(volts_Right_Back +5))-1;


       if(distance_Right_Back >= 12) // The car stops when the Right Back IR sense a wall.
         {
          Stop(100);
          Serial.println(distance_Right_Back );
          break;
         }

       else
         {
         Backward(130);

         Serial.println(distance_Right_Back );
         }
     }

Serial.println("Stop");
delay(500);

MaxRightTurn();  //Prepares the car to go backwards with a right turn.
Serial.println("Right Turn");

delay(500);


//Starts going back until there is enough room to make a Back Left turn.

   while(distance_Back != 0 )
     {

       int volts_Right_Front = analogRead(IR_Right_Front);
       int volts_Right_Back = analogRead(IR_Right_Back);
       int volts_Back = analogRead(IR_Back);

       int distance_Right_Front= (2914/(volts_Right_Front +5))-1;
       int distance_Right_Back= (2914/(volts_Right_Back +5))-1;
       int distance_Back= (2914/(volts_Back +5))-1;

       int Back_Limit=1000;
       int Total_Distance;

      if(distance_Back<=19) //This is the most important part of parking. The
     // Back IR sense the distance from the wall before making a left back turn to complete the parking.
      {

        Total_Distance=0;
        for(int x=0 ;x<100;x++) //IR vaules tend to vary and do not decrease
       // steadily as the Back IR moves closer to the wall. The average threshold was taken
       //when the back of the car is 19cmm away from the wall.
```

```
    {

    int volts_Back = analogRead(IR_Back);
    int distance_Back= (2914/(volts_Back +5))-1;
    Serial.println(distance_Back);
    Total_Distance=Total_Distance+ distance_Back;

     }
   Back_Limit= Total_Distance/100 ; // Back Limit is the reliable thershold
   //value that decides when the car changes the steering direction before completing the parallel park.
  }




   Serial.println(Back_Limit);

     if(Back_Limit <= 19)
        {
        Stop(100);
        Serial.println("SSTOOOPP");
        Serial.println(Back_Limit);
        break;
        }


     else
        {
         Backward(130);
        }
 }

delay (500);
MaxLeftTurn();

  while(distance_Right_Back=!0)   //Backing up,leftward, to complete the park closer to the wall.
    {
     int volts_Right_Front = analogRead(IR_Right_Front);
     int volts_Right_Back = analogRead(IR_Right_Back);
     int volts_Back = analogRead(IR_Back);

     int distance_Right_Front= (2914/(volts_Right_Front +5))-1;
     int distance_Right_Back= (2914/(volts_Right_Back +5))-1;
     int distance_Back= (2914/(volts_Back +5))-1;




       if(distance_Right_Back <= 3) //When the back of the car is toooo close to the wall, the car stops.
         {
          Stop(100);
         Serial.println("Right Back");
         Serial.println(distance_Right_Back);
          break;
          }

       if(distance_Back <=3) //When the back of the car is tooo close to the object behind, the car stops.
          {
           Stop(100);
            Serial.println(" Back");
         Serial.println(distance_Back);
           break;
          }

     if(distance_Right_Front == distance_Right_Back<=7)  // IF the car has enough space to complete the parking without
     //hitting the wall or the object behind, car is parallel to the wall, the car stops.
```

```
          //Also when the Right IR sensors are equal in value, the car is parallel to the thing (wall, object)right of it.

                {
                Stop(100);
                 break;
                }

              else
                 {
                  Backward(130);
                 }
     }

MaxRightTurn();
delay(500);

   while(distance_Right_Front!=0)  //If the back of the car was too close to the wall
   //or the object behind, the car  turns right to repostion itself until it's
   //parallel to wall.
      {
       int volts_Front = analogRead(IR_Front);
       int volts_Right_Front = analogRead(IR_Right_Front);
       int volts_Right_Back = analogRead(IR_Right_Back);

       int distance_Front= (2914/(volts_Front +5))-1;
       int distance_Right_Front= (2914/(volts_Right_Front +5))-1;
       int distance_Right_Back= (2914/(volts_Right_Back +5))-1;



           if(distance_Right_Front==distance_Right_Back) //Car parallel to wall.
              {
             Stop(100);
              break;
              }

            else
              {
               Forward(100);
              }

     }

Straight(); //Prepares the car to go
delay(500);

   while(distance_Front != 0) //The car moves either forward to backwad
   //until the car is equidistant from the front and back object.
      {
       int volts_Front = analogRead(IR_Front);
       int volts_Right_Front = analogRead(IR_Right_Front);
       int volts_Right_Back = analogRead(IR_Right_Back);
       int volts_Back = analogRead(IR_Back);

       int distance_Front= (2914/(volts_Front +5))-1;
       int distance_Right_Front= (2914/(volts_Right_Front +5))-1;
       int distance_Right_Back= (2914/(volts_Right_Back +5))-1;
       int distance_Back= (2914/(volts_Back +5))-1;

           if( distance_Front < distance_Back) //Moves back if  the car is closer to the back object.
             {
              Backward(100);
             }

            else if (distance_Front > distance_Back)//Moves forward if the car is closer to the front object.
             {
              Forward(100);
             }

            else if (distance_Front== distance_Back)//Stops if the car is equidistant from both objects.
```

```
                {
                Stop(100);
                break;
                }

              else
              {

              }

     }

 }
   if(readString=="t")   //Car break when "t" is pressed
   {
     Serial.println("stop");
     motor.setSpeed(100);
  motor.run(RELEASE);
   }



   readString="";

  }

}


  //Sub-Routines


//Forward , Backward, Release.
void Forward(int SPEEDF)
{
   motor.setSpeed(SPEEDF);
   motor.run(FORWARD);
}

void Backward(int SPEEDB)
{
  motor.setSpeed(SPEEDB);
  motor.run(BACKWARD);
}

void Stop(int SPEEDS)
{
  motor.setSpeed(SPEEDS);
  motor.run(RELEASE);
}



//Turns
void Straight()
{
  int STRAIGHT=90;
  myservo.write(STRAIGHT);
}


void MaxLeftTurn()
{
  int MAX_LEFT_TURN=44;
  myservo.write(MAX_LEFT_TURN);
}

void MaxRightTurn()
{
  int MAX_RIGHT_TURN=136;
```

```
  myservo.write(MAX_RIGHT_TURN);
}
```

Python Code:

```
import serial
import time
import sys
from tkinter import *
root = Tk()


ser=serial.Serial('COM7', 9600)

isSerialOpen = True

if isSerialOpen:
    print ("serial connected")
    def key(data):
        frame.focus_force()
        #print ("pressed"),
        repr(data.keysym)
        if data.keysym=="Left":
           ser.write(b'l')
           print("pressed Left")
        if data.keysym=="Right":
            ser.write(b'r')
            print ("pressed Right")
        if data.keysym=="Up":
            ser.write(b'u')
            print ("pressed Up")
        if data.keysym=="Down":
            ser.write(b'd')
            print ("pressed Down")
        if data.keysym=="t":
            ser.write(b't')
            print ("pressed t")
        if data.keysym=="s":
            ser.write(b's')
            print ("pressed s")

frame = Frame(root, width=100, height=100)
frame.bind("<Key>", key)
frame.pack()
frame.focus_set()

root.mainloop()
```