Autonomous RC Car

Logan Slade
# Table of Contents

Introduction .......................................................................................................................... 3
Concept ................................................................................................................................. 4
Requirements/Goals ............................................................................................................. 5
Materials ............................................................................................................................. 5
Method/Design ................................................................................................................... 6
  RC Car Body Prep ............................................................................................................. 6
  Electronics Hardware Setup ........................................................................................... 6
  Code ................................................................................................................................. 8
  Pixy Sensor Configuration .............................................................................................. 11
  Android App Connection ............................................................................................... 11
  Operation ....................................................................................................................... 12
Challenges ........................................................................................................................... 12
Future Application ............................................................................................................. 15
Appendix ............................................................................................................................ 16
  Arduino Code ................................................................................................................ 16
Introduction

One of the emerging technologies in today's world is the advancement of driver assistance systems in cars and the development of autonomous vehicles. Vehicles are relying less on the driver maintaining complete control of the vehicle with the goal that eventually the vehicle will be able to operate without a driver at all.


One of the many techniques used by autonomous vehicles is computer vision and image recognition. Through advanced image processing, computers can recognize and track images which can be used to help drive autonomous vehicles.

https://blogs.nvidia.com/blog/2016/01/05/eyes-on-the-road-how-autonomous-cars-understand-what-theyre-seeing/
Concept

I wanted to get experience with autonomous vehicles, but I could not make a full-size autonomous vehicle with a full-size car. So, I decided to turn a toy remote-controlled car into an autonomous vehicle. I wanted to use an already built car as the body of my autonomous vehicle to reduce the mechanical engineering design of this project and focus on the electrical side.

I wanted to make a remote-controlled car that ran autonomously, but I also wanted the option to operate it manually with a Bluetooth controller. My design was to have three modes for my car. The first mode would be one in which I would drive the car manually using a Bluetooth controller. The second mode was an autonomous driving mode in which the car would drive around on its own avoiding obstacles in its way. The third mode was a search and find mode. In this mode the car would drive around autonomously, but while it was driving around it would be searching for a specified object. Once it found the object, it would drive toward that object and stop as soon as it reached it.

I decided to use an Arduino as the “brains” of my autonomous car. I decided on the Arduino because it was inexpensive, was small enough for my project, had plenty of GPIO inputs and outputs for my needs. I also considered the Raspberry Pi Zero at first because it had built in Bluetooth and was smaller than the Arduino, but starting the program and shutting down the microcontroller was much more of a hassle due to the background OS that the Raspberry Pi runs on.

For the object recognition in my search and find mode I decided to use a Pixy Smart Vision Sensor. This sensor uses a camera with a microcontroller built into it which is programmed to detect objects based on their color. I went with this option because the Arduino does not have enough processing power to perform image recognition, and the Pixy camera was designed to be easily interfaced with an Arduino.

https://smile.amazon.com/gp/product/B00IUYUA80/ref=oh_aui_detailpage_o08_s00?ie=UTF8&psc=1
Requirements/Goals

A list of requirements and goals was developed to help guide me in the development of this project. The requirements I came up with are as follows.

- The car shall be controlled by a Bluetooth controller.
- The Bluetooth controller shall have a button to switch operating modes.
- The car shall be battery powered.
- The car shall be able to drive indoors on hard floor and low carpet.
- The car shall be able to find an object based off its color.
- The car shall not run into objects when driving autonomously.
- The car shall be able to run for at least 2 hours off battery power.
- The car shall have a visual indicator showing which driving mode it is currently in.
- The car shall not have loose parts that might fall off when driving.

Materials

To design this project the following materials were used.

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Uno R3</td>
<td>$22</td>
</tr>
<tr>
<td>RC Car</td>
<td>$10</td>
</tr>
<tr>
<td>Pixy Smart Vision Sensor</td>
<td>$70</td>
</tr>
<tr>
<td>Adafruit DRV8833 DC/Stepper Motor Driver</td>
<td>$10</td>
</tr>
<tr>
<td>HC-05 Bluetooth Module</td>
<td>$9</td>
</tr>
<tr>
<td>Ultrasonic Sensor (x3)</td>
<td>$8</td>
</tr>
<tr>
<td>Multicolor LED</td>
<td>$1</td>
</tr>
<tr>
<td>Jumper Wires</td>
<td>$5</td>
</tr>
<tr>
<td>55 Tie Point Mini Breadboard</td>
<td>$1</td>
</tr>
<tr>
<td>170 Tie Point Mini Breadboard</td>
<td>$1</td>
</tr>
<tr>
<td>9V Battery</td>
<td>$3</td>
</tr>
<tr>
<td>9V Battery Clip with Arduino Plug</td>
<td>$1</td>
</tr>
<tr>
<td>AA batteries (x3)</td>
<td>$4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$145</strong></td>
</tr>
</tbody>
</table>

In addition to the hardware used on the car itself, there was some other hardware necessary to build and run the car. These additional tools included a soldering iron and some solder to use to disassemble the RC car and solder wires to the pins connected to the motor batteries. In addition an Android phone is required along with the app “Arduino Car” which can be found in the Google Play Store.
Method/Design

RC Car Body Prep
The first part of building the autonomous RC car was to prepare the RC car body to add the electronics onto. This required removing any exterior body parts to leave only the car’s chassis. The connections to the two motors needed to be located to connect to the motor driver. The positive and negative terminals of the RC car’s batteries also had to be located to provide power to the motor. I soldered wires to the battery terminals to be able to connect to the motor driver and ground pin. The circuit board from the RC car could be left on or taken off when preparing the RC car body. I chose to take the board off to make more room for my electronics.

Electronics Hardware Setup
The next step was to set up the electronics hardware on the car. This involved mounting and wiring the components. The ultrasonic sensors and Pixy camera required specific positions to mount onto the car, but the rest of the components could be mounted however possible to fit onto the car. For the ultrasonic sensors, two are mounted on the front of the car and one on the back. The two on the front needed be positioned nearly symmetrical with one on the left and one on the right. The ultrasonic sensor on the back needed to be positioned at the center of the vehicle.
The Pixy camera was mounted in the front with the camera centered and facing forward.

The I2C cable for the Pixy camera connects the camera to the Arduino as shown below.

All other electronic components were connected to the Arduino as outlined in the table below.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>HC-05 TXD</td>
</tr>
<tr>
<td>D1</td>
<td>HC-05 RXD</td>
</tr>
<tr>
<td>D2</td>
<td>Motor Driver AIN1</td>
</tr>
<tr>
<td>D3</td>
<td>Motor Driver AIN2</td>
</tr>
<tr>
<td>D4</td>
<td>Motor Driver BIN1</td>
</tr>
</tbody>
</table>
D5  | Motor Driver BIN2  
D6  | Motor Driver SLP  
D7  | Front Left Ultrasonic Sensor Trigger and Echo  
D8  | Front Right Ultrasonic Sensor Trigger and Echo  
D9  | Rear Ultrasonic Sensor Trigger and Echo  
A3  | LED Blue  
A4  | LED Green  
A5  | LED Red  
5V  | All Ultrasonic Sensors Vcc, HC-05 Vcc  
Gnd | All Ultrasonic Sensors Gnd, HC-05 Gnd, Motor Driver Gnd, LED Gnd, Motor Battery Negative Terminal  

The connections for the motor driver are given in the following table.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIN1</td>
<td>Arduino Pin D2</td>
</tr>
<tr>
<td>AIN2</td>
<td>Arduino Pin D3</td>
</tr>
<tr>
<td>BIN1</td>
<td>Arduino Pin D4</td>
</tr>
<tr>
<td>BIN2</td>
<td>Arduino Pin D5</td>
</tr>
<tr>
<td>SLP</td>
<td>Arduino Pin D6</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>VM</td>
<td>Motor Battery Positive Terminal</td>
</tr>
<tr>
<td>AOUT1</td>
<td>Rear Motor Negative Terminal</td>
</tr>
<tr>
<td>AOUT2</td>
<td>Rear Motor Positive Terminal</td>
</tr>
<tr>
<td>BOUT1</td>
<td>Front Motor Positive Terminal</td>
</tr>
<tr>
<td>BOUT2</td>
<td>Front Motor Negative Terminal</td>
</tr>
</tbody>
</table>

After connecting the motor driver to the motors if during operation either motor was running in the opposite direction of the input, all I had to do to fix it was switch the two connections for that motor.

**Code**

Once the hardware was set up, the code was uploaded to the Arduino using a USB cable. I chose to use the online Arduino Create IDE to write and upload the code for this project.
When uploading the program to the Arduino the connection to the GPIO pin D0 had to be temporarily removed until uploading. This became somewhat of a hassle after many times having to upload new code. This must be done since the D0 pin is used for serial communication between the Arduino and computer.

The design for the code follows the outline shown in the diagrams below.
Pixy Sensor Configuration
The Pixy Smart Vision Sensor needed to be configured with color signatures in order to detect objects. To make the objects more easily detectable I used colors that really stood out from their surroundings.

To set the color signatures that the Pixy sensor would detect, I used the PixyMon software designed for the Pixy sensor. I used the CC signatures instead of the normal color signatures because then the camera would look specifically for the two colors next to each other. This helped reduce the number of faulty readings from the sensor.

Android App Connection
I designed this project to be used with an Android phone and the “Arduino Car” app by ElectroTex which can be found on the Google Play store. This was the best interface that I could find that was compatible with my Bluetooth HC-05 module.
Operation
With all the hardware, software, and interfacing set up, I could then operate the vehicle. I set the ‘Y’ button on the Arduino Car app to change the driving modes. The multicolor LED allowed me to see which driving mode the car was currently in. Red signified mode 1 (Bluetooth manual mode), green for mode 2 (autonomous driving), and blue for mode 3 (search and find).

Challenges
In the design of this project there were many challenges encountered. Some of the hardware issues that arose resulted in completely different physical designs of the car.

Originally, I wanted to use an 8bitdo SNES Bluetooth controller as the user input for the car.
However, the SNES controller would not work with the HC-05 module because it uses HID Bluetooth which I found out is incompatible with the HC-05. This caused an issue because I could not find another Bluetooth controller which had the proper interface I needed. While still looking for a different controller I created an early design of the RC car using a Raspberry Pi Zero W since it had Bluetooth built-in which was compatible with HID devices.

This design didn’t work well because the Raspberry Pi’s background OS forced me to have to run it off a remote desktop to start up the program and to properly shut it down. I also had to use a portable battery charger to power the Raspberry Pi which added a lot of weight to the car.

Another challenge I had with designing the car was that my early designs had terrible turning radius which caused a lot of difficulty in trying to avoid obstacles. I tried fixing this by using a different, smaller RC car body which helped the turning radius a little, but the RC car body was too small to fit all the hardware I needed on it.
The turning radius on the car was still not as good as it was out of the package, so I figured there must have been something wrong with the motor power delivery in my design. I found that my problem was my motor driver. I was initially using a L293D motor driver, but it’s output current was only 0.6 A which was not enough to run my motors. I replaced the L293D driver with an Adafruit DRV8833 motor driver which had a current output of 1.2 A. With the new motor driver my steering issue was fixed. It also helped with getting enough torque to move the car forward and in reverse.

Another issue that proved challenging was getting good input from the ultrasonic sensors to control the car. At first, I used only one ultrasonic sensor. This did not work well because the sensor had such a limited range of detection. When an object was to the side, just outside the ultrasonic sensor’s range, the vehicle would get stuck. I started by adding a second ultrasonic sensor to the front of the vehicle to increase the range of detection. This helped the car avoid hitting an object on its side and getting stuck. I also added an ultrasonic sensor to the back to help with a backup function I used to get the car out of situations where it got stuck.
The extra ultrasonic sensors helped a lot in keeping the car from getting stuck, but there were still plenty of times where the ultrasonic sensors were not enough. Another problem was when the car got going fast enough it couldn’t stop and reverse in time before hitting the object it was headed toward.

I also faced difficulties with the Pixy sensor. The sensor detects objects based off their color code but even with my very bright color-coded objects, in different lighting situations the camera would not recognize the colors as being the ones I had set. This would cause the car to pass right by the object without detecting it.

**Future Application**

With the current design of my autonomous RC car, it can be used as a toy or for entertainment purposes. However, the design could be improved and expanded to allow for many more applications.

One improvement that could be made to my design is to simplify some of the hardware on the car into a PCB board to reduce the clutter of wires and other components. I could also modify it so that instead of
having to use a 9V battery to power the Arduino and other components and 3 AA batteries to run the motor, I would only have to use one set of batteries to run the whole thing.

Another improvement may be to modify the object recognition in the search and find mode. Instead of using a Pixy sensor to recognize an object based off it’s color alone, a different camera could be used with some image processing software. Since the Arduino wouldn’t be able to handle this on its own, another microcontroller with better processing power, like a Raspberry Pi Zero, could be used to do the image processing and then send the data to the Arduino to use in its autonomous driving function.

The search and find function on this RC car could be expanded to a larger, more rugged vehicle, and with proper image recognition software it could be used in search and rescue efforts. Some functions of this design could also possibly be used to design autonomous drones. These drones could also be used in search and rescue missions with proper image recognition software.

Appendix

Arduino Code

#include <SPI.h>
#include <Pixy.h>
#include <NewPing.h>

#define frPin1 2
#define frPin2 3
#define lrPin1 4
#define lrPin2 5
#define MotorEn 6
#define flSensor 7 //left front sensor
#define frSensor 8 //right front sensor
#define backSensor 9
#define red A5
#define green A4
#define blue A3
#define horn A2

define Moves{FORWARD, BACKWARD, LEFT, RIGHT, FORWARDLEFT, FORWARDRIGHT, BACKLEFT, BACKRIGHT, STOPPED};

char state = ' ';
long unsigned int ldist, rdist, bdist;
long unsigned int closeDist = 10;
long unsigned int midDist = 60;
int avdCount = 0;
int repeats = 0;
int lastMove = STOPPED;
int mode = 1;
Pixy pixy; //pixy camera object
NewPing flSonar(flSensor, flSensor);
NewPing frSonar(frSensor, frSensor);
NewPing bSonar(backSensor, backSensor);

void setup() {

  //initialize pins
  pinMode(frPin1, OUTPUT);
  pinMode(frPin2, OUTPUT);
  pinMode(lrPin1, OUTPUT);
  pinMode(lrPin2, OUTPUT);
  pinMode(MotorEn, OUTPUT);
  pinMode(horn, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);

digitalWrite(frPin1, LOW);
digitalWrite(frPin2, LOW);
digitalWrite(lrPin1, LOW);
digitalWrite(lrPin2, LOW);
digitalWrite(MotorEn, LOW);
digitalWrite(horn, LOW);
digitalWrite(red, LOW);
digitalWrite(green, LOW);
digitalWrite(blue, LOW);

  Serial.begin(38400); //for interfacing with serial port for debugging
  pixy.init(); //initialize pixycam

  ldist = rdist = bdist = 0;
}

void loop() {
  mode1();
}

/**************************************************************************/
Mode Functions
**************************************************************************/

void mode1() { //Bluetooth controller mode
  //Serial.println("Starting mode 1...");
  mode = 1;
  stopped(0);
  digitalWrite(red, HIGH); //red light to signify mode 1
digitalWrite(green, LOW);
digitalWrite(blue, LOW);
while(true) {
    readBT();
}

void mode2() { //autonomous mode
    //Serial.println("Starting mode 2...");
    mode = 2;
    stopped(0);
    digitalWrite(red, LOW);
    digitalWrite(green, HIGH); //green light to signify mode 2
    digitalWrite(blue, LOW);

    while(true) {
        readBT();
        avoid();
    }
}

void mode3() { //search mode
    //Serial.println("Starting mode 3...");
    int objectClose = 10;
    mode = 3;
    stopped(0);
    digitalWrite(red, LOW);
    digitalWrite(green, LOW);
    digitalWrite(blue, HIGH); //blue light to signify mode 3

    while(true) {
        readBT();

        uint16_t blocks; //variable for receiving data from pixycam
        blocks = pixy.getBlocks(); //get data from pixycam

        if(blocks){ //if object was found in pixycam view
            Serial.print("Object detected. x = ");
            Serial.println(pixy.blocks[0].x);
            //move in direction of object
            
            ldist = flSonar.ping_cm();
            //Serial.print("ldist = ");
            //Serial.println(ldist);
            if(ldist == 0) {
                ldist = 200;
            }

            rdist = frSonar.ping_cm();
            //Serial.print("rdist = ");
            //Serial.println(rdist);
            //Serial.println();
            if(rdist == 0) {
                
            }
        }
    }
}
rdist = 200;
}

/* if(ldist < 5 && pixy.blocks[0].x < 160) {
   if(ldist < objectClose) {
      Serial.println("Reached object.");
      stopped(0);
      mode4(); //wait for reset or mode change
   }
   if(rdist < 5 && pixy.blocks[0].x > 160) {
      if(rdist < objectClose) {
         Serial.println("Reached object.");
         stopped(0);
         mode4(); //wait for reset or mode change
      }
   }
   if(ldist < objectClose && pixy.blocks[0].x < 160) {
      Serial.println("Reached object.");
      stopped(0);
      mode4(); //wait for reset or mode change
   }
   if(rdist < objectClose && pixy.blocks[0].x > 160) {
      Serial.println("Reached object.");
      stopped(0);
      mode4(); //wait for reset or mode change
   }
}

else { //move toward object and center in view area
   int x = pixy.blocks[0].x;
   if(x < 100) {
      Serial.println("Object to left");
      forwardLeft(25);
   } else if(x > 219) {
      Serial.println("Object to right");
      forwardRight(25);
   } else {
      Serial.println("Object straight ahead");
      forward(25);
   }
}

else { //no object found
   //Serial.println("No object found");
   avoid(); //run autonomously
}
}

void mode4() { //object found, waiting for reset or mode change
   //Serial.println("Found object. Waiting for input...");
mode = 4;
digitalWrite(red, HIGH);
digitalWrite(green, HIGH);
digitalWrite(blue, HIGH);
state = ' ';
while(true) {
    readBT();
}

void readBT() {
    state = ' ';
    if(Serial.available() > 0) { // Checks whether data is comming from the serial port
        state = Serial.read(); // Reads the data from the serial port
    }
}

switch(state) {
    case 'F': // forward
        forward(0);
        break;
    case 'G': // reverse
        backward(0);
        break;
    case 'L': // left
        left(0);
        break;
    case 'R': // right
        right(0);
        break;
    case 'Q': // forward left
        forwardLeft(0);
        break;
    case 'E': // forward right
        forwardRight(0);
        break;
    case 'Z': // back left
        backLeft(0);
        break;
    case 'C': // back right
        backRight(0);
        break;
    case 'X':
        if(mode == 4) {
            // Further code
mode3();
}
else {
    honk(200);
}
break;

case 'Y':
    switch(mode){
    case 1:
        mode2();
        break;
    case 2:
        mode3();
        break;
    case 3:
        mode1();
        break;
    case 4:
        mode1();
        break;
    default:
        mode1();
        break;
    }
break;

case 'S':
    stopped(0);
    break;

default:
    break;
}

/****************************
Motor Motion Functions
****************************/

void stopped(int delayTime) {
    digitalWrite(MotorEn, LOW);
    digitalWrite(frPin1, LOW);
    digitalWrite(frPin2, LOW);
    digitalWrite(lrPin1, LOW);
    digitalWrite(lrPin2, LOW);
    delay(delayTime);
    if(lastMove = STOPPED) {
        repeats++;
    }
else {
repeats = 0;
)
lastMove = STOPPED;
}

void forward(int delayTime) {
digitalWrite(lrPin1, LOW);
digitalWrite(lrPin2, LOW);
digitalWrite(frPin1, LOW);
digitalWrite(frPin2, HIGH);
digitalWrite(MotorEn, HIGH);
delay(delayTime);
if(lastMove == FORWARD) {
    repeats++;
} else {
    repeats = 0;
}
lastMove = FORWARD;
}

void backward(int delayTime) {
digitalWrite(lrPin1, LOW);
digitalWrite(lrPin2, LOW);
digitalWrite(frPin1, HIGH);
digitalWrite(frPin2, LOW);
digitalWrite(MotorEn, HIGH);
delay(delayTime);
if(lastMove == BACKWARD) {
    repeats++;
} else {
    repeats = 0;
}
lastMove = BACKWARD;
}

void left(int delayTime) {
digitalWrite(frPin1, LOW);
digitalWrite(frPin2, LOW);
digitalWrite(lrPin1, LOW);
digitalWrite(lrPin2, HIGH);
digitalWrite(MotorEn, HIGH);
delay(delayTime);
}

void right(int delayTime) {
digitalWrite(frPin1, LOW);
digitalWrite(frPin2, LOW);
digitalWrite(lrPin1, HIGH);
digitalWrite(lrPin2, LOW);
digitalWrite(MotorEn, HIGH);
delay(delayTime);
}

void forwardLeft(int delayTime) {
left(0);
digitalWrite(frPin1, LOW);
digitalWrite(frPin2, HIGH);
delay(delayTime);
if(lastMove == FORWARDLEFT) {
    repeats++;
} else {
    repeats = 0;
}
lastMove = FORWARDLEFT;
}

void forwardRight(int delayTime) {
right(0);
digitalWrite(frPin1, LOW);
digitalWrite(frPin2, HIGH);
delay(delayTime);
if(lastMove == FORWARDRIGHT) {
    repeats++;
} else {
    repeats = 0;
}
lastMove = FORWARDRIGHT;
}

void backLeft(int delayTime) {
right(0);
digitalWrite(frPin1, HIGH);
digitalWrite(frPin2, LOW);
delay(delayTime);
if(lastMove == BACKLEFT) {
    repeats++;
} else {
    repeats = 0;
}
lastMove = BACKLEFT;
}

void backRight(int delayTime) {
left(0);
digitalWrite(frPin1, HIGH);
digitalWrite(frPin2, LOW);
delay(delayTime);
if(lastMove == BACKRIGHT) {
    repeats++;
}
} 
else { 
    repeats = 0; 
}
lastMove = BACKRIGHT;
} 

void honk(int delayTime) {
digitalWrite(horn, HIGH);
delay(delayTime);
digitalWrite(horn, LOW);
}

="/********************
Autonomous Driving Functions
***************************/

void avoid() { //run autonomously avoiding objects

    ldist = flSonar.ping_cm();
    //Serial.print("ldist = ");
    //Serial.println(ldist);
    if(ldist == 0) {
        ldist = 200;
    }

    rdist = frSonar.ping_cm();
    //Serial.print("rdist = ");
    //Serial.println(rdist);
    //Serial.println();
    if(rdist == 0) {
        rdist = 200;
    }

    if(rdist < midDist || ldist < midDist) {
        if(rdist < midDist && ldist < midDist) { //both sensors are in close distance
            backward(300);
            stopped(0);
        }
        else if(rdist < closeDist) { //right sensor in close distance, left in mid distance
            backLeft(300);
            stopped(0);
        }
        else if(ldist < closeDist) { //left sensor in close distance, right in mid distance
            backRight(300);
            stopped(0);
        }
        else if(ldist < rdist) { //both sensors mid distance, left sensor closer to object
            forwardRight(100);
        }
    }
else { //both sensors mid distance, right sensor closer to object
    forwardLeft(100);
}
}
else if(rdist < midDist) { //right sensor in mid or close distance, left sensor far from object
    if(rdist < closeDist) { //right sensor in close distance, left sensor far from object
        backLeft(300);
    }
    else { //right sensor in mid distance, left sensor far from object
        forwardLeft(100);
    }
}
else if(ldist < closeDist) { //right sensor far from object, left sensor in close distance
    backLeft(300);
} else { //right sensor far from object, left sensor in mid distance
    forwardRight(100);
}
else { //both sensors far from object
    forward(100);
}
if(repeats > 20) {
    backOut();
}
void backOut() {
    int backDelay = 10;
    int backMax = 200;
    int backClose = 15;
    int bcount = 0;
    while(bcount < backMax) {
        if(lastMove == forwardLeft) {
            backRight(backDelay);
            lastMove = forwardLeft; //needed to keep moving backRight
        }
        else if(lastMove == forwardRight) {
            backLeft(backDelay);
            lastMove = forwardRight; //needed to keep moving backLeft
        }
        else {
            backward(backDelay);
        }
        bdist = bSonar.ping_cm();
        //Serial.print("bdist = ");
        //Serial.println(bdist);
        //Serial.println();
        if(bdist == 0) {
bdist = 200;
}
if(bdist < backClose) {
   bcount = backMax;
}
bcount++;