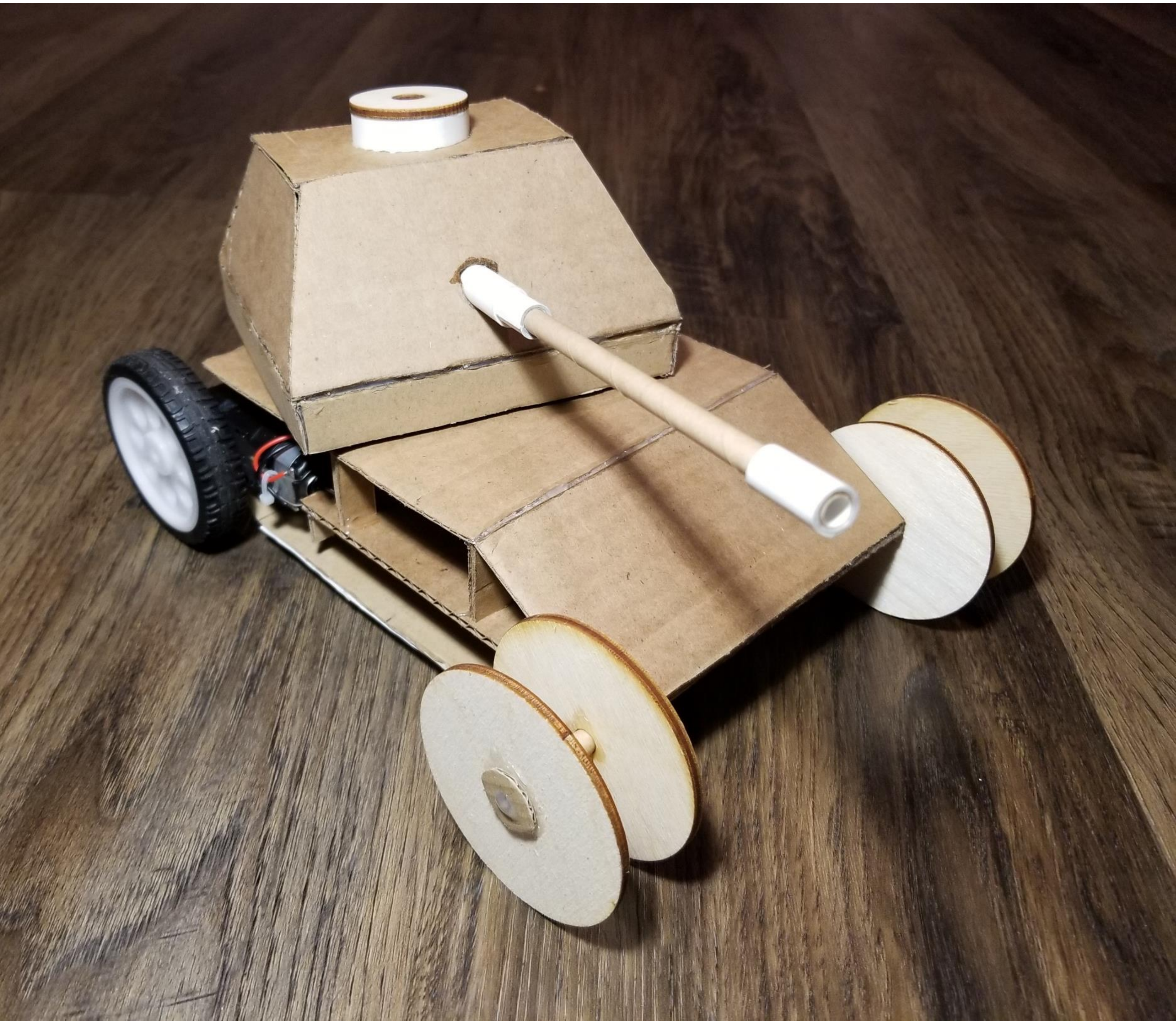
V-12T: Bluetooth Controlled Vehicle

By Kiaria Johnson

ME102B: Mechatronics Design

Dr. Hannah Stuart and Dr. Homayoon Kazerooni

University of California Berkeley | Fall 2020 | December 8, 2020



Product Description:

During the Fall 2020 semester, I learned how to control motors via Bluetooth connection between a phone application and a microcontroller. To this day, I enjoy messing around with remote controlled cars, so I figured it would be cool to make my own. I primarily used mixed materials like cardboard, plywood, and cardstock to construct the outer body and aesthetic features of this tank-like vehicle. Overall, the final product is a fun little Bluetooth controlled vehicle that can maneuver on flat, incline, and rough surfaces if desired.

Electronic and Mechanical Features:

<u>DC Gearbox motors and Mini Servo motor</u>: While the two DC motors supply power to actuate the wheels to move the vehicle, the servo motor adds extra character to the vehicle by actuating a turret made from cardboard in a sweeping pattern.

<u>Tank Body</u>: Almost entirely constructed from cardboard, the tank body houses the DC motors, mini servo, and Romeo BLE Mini microcontroller, in addition to a battery box that can be seen in Figure 2, which sits a level between the bottom trapezoidal cardboard piece and the mid-level cardboard flat, which supports the motors and microcontroller.

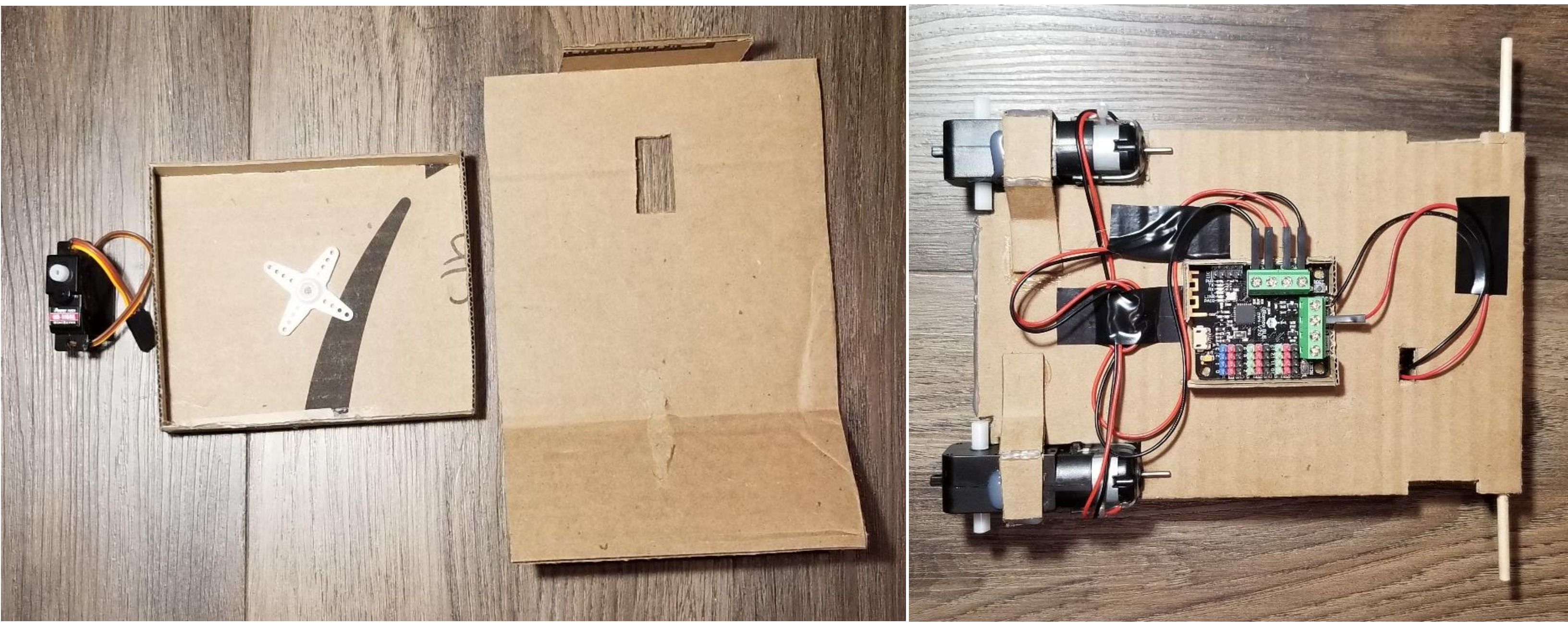


Figure 1: (Left) Top trapezoidal piece of tank with cutout for mini servo and (right) two DC gearbox motors fixed to trapezoidal base. The Romeo BLE Mini microcontroller is also shown (see part description).

Part Description:

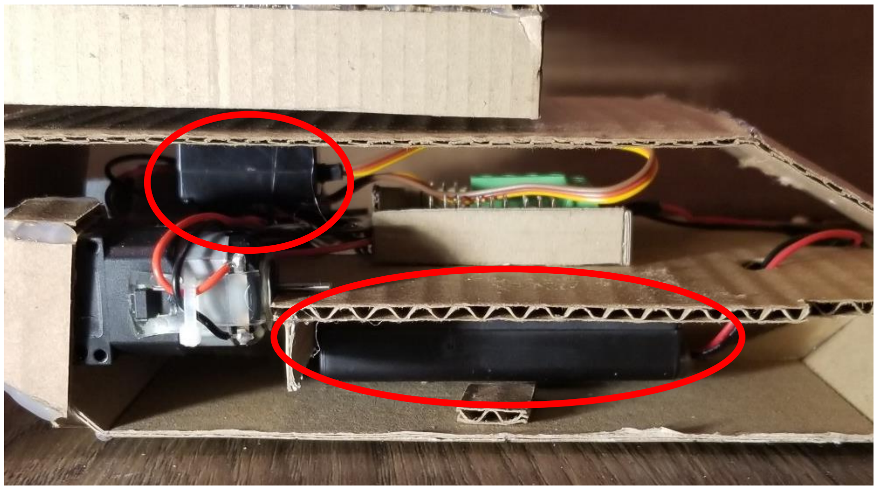


Figure 2: Side view highlighting the servo motor and battery pack within the tank body.

- DC gearbox motors [1]: possessing a 1:48 gear ratio, the T shaft output provides more torque at the cost of reduced speed compared to the motor shaft. The rear wheels interface directly with the output shaft.

- Mini Servo Motor [2]: actuates the turret sweep. Has potential to have its functionality modified via code for more advanced sequences apart from a simple sweep pattern.

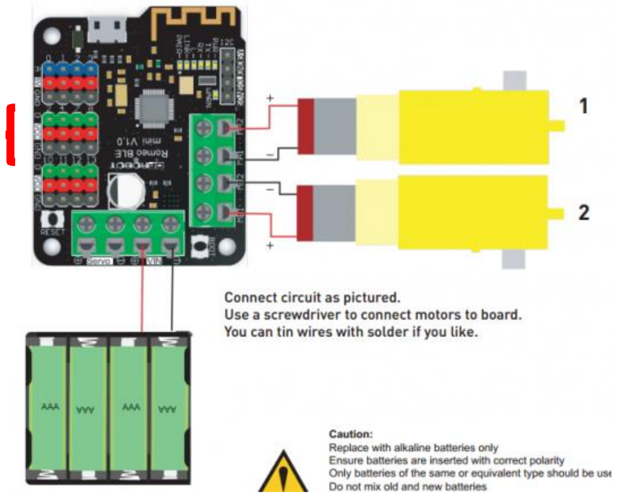
- Romeo BLE Mini [3]: this microcontroller has a servo and DC motor driver built in, in addition to being battery operated via the battery pack shown in Figure 2 amounting to 6 volts. Thus, each component listed above was connected and controlled by the microcontroller, which also acted as a voltage source.

- GoBLE iOS Application [4]: Possesses a scan function to connect to Bluetooth devices in the area and has a four-button directional pad to control motor direction for forward, reverse, and turning operations. There is also a digital joystick that can handle wider turns and reverse operations (not used), as well as a "kill switch" to stop the motors from operating when pressed at any point during operation.

Circuit:

The circuit is straight forward as shown below. The motors are connected to their respective inputs on the microcontroller and the battery pack is connected to V_{in}. The rest is all handled by the code which can be seen in Appendix I.

Servo connected across pin 9 (wires: digital (green), Vcc (red), GND (black))



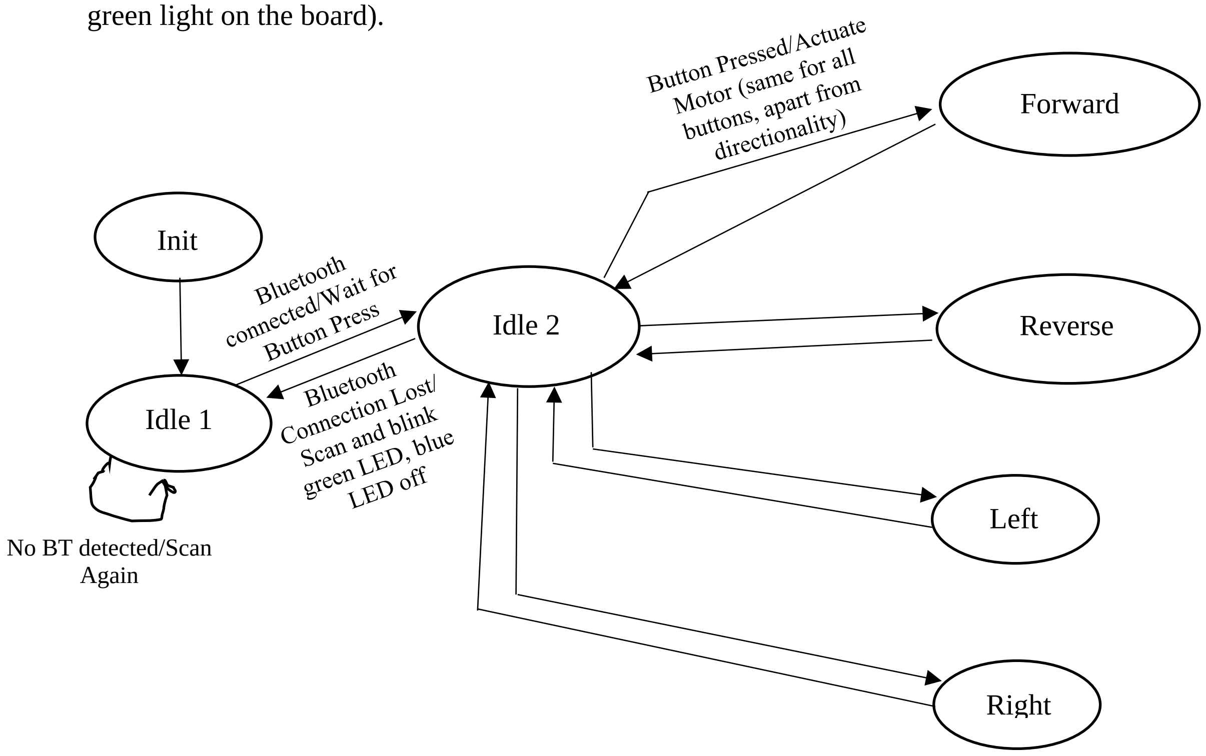
Connect circuit as pictured.
Use a screwdriver to connect motors to board.
You can tin wires with solder if you like.

Caution:
Replace with alkaline batteries only
Ensure batteries are inserted with correct polarity
Only batteries of the same or equivalent type should be use
Do not mix old and new batteries

Finite State Diagram:

The tank is controlled by a series of "if" statements that determine whether a specific motor control button is pressed. For instance, if the up button is pressed the motor will go in reverse (since it was built that way) so long as the button is held. Once it is released the motor will stop on the spot and the microcontroller waits until another button is pressed to command the motors accordingly (Note: joystick functionality not used, so its FSD operation is omitted).

Before motor actuation begins, the microcontroller waits until a Bluetooth connection has been established between the desired device and itself (indicated via by a solid blue light instead of a blinking green light on the board).


Button Pressed/Actuate Motor (same for all buttons, apart from directionality)
Forward
Init
Bluetooth connected/Wait for Button Press
Idle 2
Reverse
Idle 1
Bluetooth Connection Lost/ Scan and blink green LED, blue LED off
Left
No BT detected/Scan Again
Right

State Descriptions:

1) Init: initialize motor and servo pins and variables used.

2) Idle 1: blink green LED until a Bluetooth (BT) source is connected, then Blue LED set to HIGH.

3) Idle 2: wait for button press to actuate motors.

4) Forward: send PWM signal to both motors to run forward (as wired) at 200 out of 255 duty.

5) Reverse: send PWM signal to both motors to run reverse (as wired) at 75 out of 255 duty.

6) Left: send PWM signal to both motors at 100 out of 255 duty, where one runs forward and the other in reverse.

7) Right: send PWM signal to both motors at 100 out of 255 duty, where one runs forward and the other in reverse (motor direction opposite of Left command).

***Note: All arrows returning to Idle 2 from a directional button signal a button release and subsequent waiting for next button press OR the "Kill Switch" being pressed.

Appendix I: Arduino Code

```
/* -----Flamewheel Bluetooth Control Program

//------2016.6.29 by LL

//------Suitable for Romeo BLE Mini MCU

//https://www.dfrobot.com/index.php?route=product/product&product_id=1367&search=ble+mini&descr
iption=true#.V8AR1q11Zfc

*/

#include "GoBLE.h"

#include <Romeo_m.h>

#include <Servo.h>


Servo myservo;

int pos = 90;


#define LED 13


//GoBLE Goble(Serial);

int joystickX, joystickY;

int buttonState[7];

unsigned int led_count;

long current_t;

long prev_t = 0;

int count = 0;


void setup() {

  Romeo_m.Initialise();

  Goble.begin();

  pinMode(LED, OUTPUT);

  myservo.attach(9);

}

void loop() {
  if (Goble.available())
  {
    readGoBle();
    motorContrl();
  }
  current_t = millis();
  if (current_t - prev_t >= 75){
    updateServo();
//    Serial.println(current_t-prev_t);
    prev_t = current_t;
    count++;
  }
  delayLedBlink();//delay 10ms and led blink
}
//Read GoBLE values
void readGoBle()
{
  // read joystick value when there's valid command from bluetooth
  joystickX = Goble.readJoystickX();
  joystickY = Goble.readJoystickY();
  // read button state when there's valid command from bluetooth
  buttonState[SWITCH_UP]     = Goble.readSwitchUp();
  buttonState[SWITCH_DOWN]   = Goble.readSwitchDown();
  buttonState[SWITCH_LEFT]   = Goble.readSwitchLeft();
  buttonState[SWITCH_RIGHT]  = Goble.readSwitchRight();
  buttonState[SWITCH_SELECT] = Goble.readSwitchSelect();
  buttonState[SWITCH_START]  = Goble.readSwitchStart();
  /*Serial.println("======================");
  Serial.print("Joystick Value: ");
  Serial.print(joystickX);
  Serial.print("  ");
  Serial.println(joystickY);
  for (int i = 1; i <= 6; i++) {
    if (buttonState[i] == PRESSED) {
      Serial.print(" ID: ");
      Serial.print(i);
      Serial.print("\t ");
      Serial.println("Pressed!");
    }
    if (buttonState[i] == RELEASED){
      Serial.print("ID: ");
      Serial.print(i);
      Serial.print("\t ");
      Serial.println("Released!");
    }
  }*/
}
//Move according to GoBLE value
//Joystick left and right to turn bends, left and right buttons to spin on the spot
void motorContrl()
{
  if ((buttonState[SWITCH_UP] == PRESSED) || ((joystickX > 128) && (joystickY >= 64) &&
(joystickY <= 192)))
  {
    Romeo_m.motorControl(Forward, 75, Forward, 75); //go forward
    return;//end function
  }
  if ((buttonState[SWITCH_DOWN] == PRESSED) || ((joystickX < 128) && (joystickY >= 64) &&
(joystickY <= 192)))
  {
    Romeo_m.motorControl(Reverse, 200, Reverse, 200); //go backwards
    return;//end function
  }
  if (buttonState[SWITCH_LEFT] == PRESSED)
  {
    Romeo_m.motorControl(Reverse, 100, Forward, 100); //turn left
    return;//end function
  }
  if ((joystickY < 128 ) && (joystickX >= 64 ) && ( joystickX <= 192) )
  {
    Romeo_m.motorControl_M1(Forward, 80); //turn left big bend
    Romeo_m.motorControl_M2(Forward, 200);
    return;//end function
  }
  if ( buttonState[SWITCH_RIGHT] == PRESSED)
  {
    Romeo_m.motorControl(Forward, 100, Reverse, 100); //turn right
    return;//end function
  }
  if ((joystickY > 128) && (joystickX >= 64) && (joystickX <= 192))
  {
    Romeo_m.motorControl_M2(Forward, 80); //turn right big bend
    Romeo_m.motorControl_M1(Forward, 200);
    return;//big bend
  }
  Romeo_m.motorStop();//no stop button is pressed
}
//led blink funtion, each execution delay 10ms. every 100 times level inverted
void delayLedBlink()
{
  delay(10);
  led_count++;
  if (led_count > 100)
  {
    digitalWrite(LED, !digitalRead(LED));
    led_count = 0;
  }
}

// Update servo position without using delays
void updateServo()
{
  if (count <= 45){
    myservo.write(pos);
    pos++;
  }
  else if (count <= 120){ // Full range swept increment back to starting position
    myservo.write(pos);
    pos--;
  }
   else if (count < 150){
    myservo.write(pos);
    pos++;
  }
  else if (count == 150){
    myservo.write(pos);
    pos = 90;
    count = 0;
  }
}
```

Appendix II: Project Component Websites

[1] https://www.adafruit.com/product/3777

[2] http://www.chd.hk/Product_Detail.aspx?id=29

[3] https://www.dfrobot.com/product-1367.html

[4] https://apps.apple.com/us/app/goble-bluetooth-4-0-controller/id950937437

Wheels Interfaced with the Motor: https://www.adafruit.com/product/3763