ME102B Final Project: **Gyroscope Controlled RC Car**

Yingying Chen

Due: December 8th, 2020

Product Description

As Christmas approaches, a popular gift for small children is an RC car. This year I wanted to surprise my younger cousin with a really unique RC car that she and I could play with together! One of her favorite cars is the classic Volkswagen bus, so I decided to make a RC Car with some recycled parts from an old Arduino project. This way she and I can improve it together since she also curious about robotics. Instead of the original joystick control, I thought a gyroscope control would be more unique and interesting to work with.

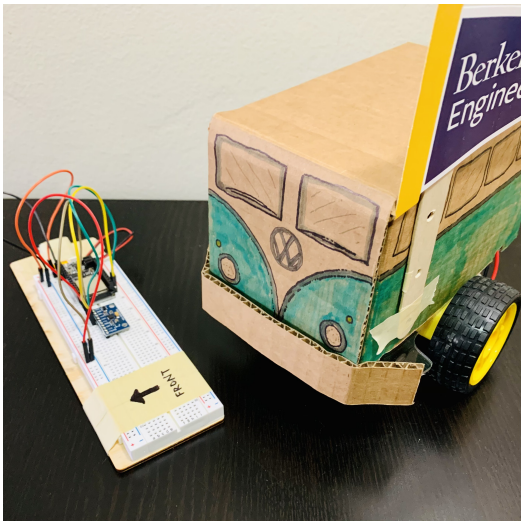


Figure 1: My RC Volkswagen bus (right) with its gyroscope controller (left)

For video demonstration of this project, visit https://youtu.be/7JwaCxzMmRQ

Electromechanical Details

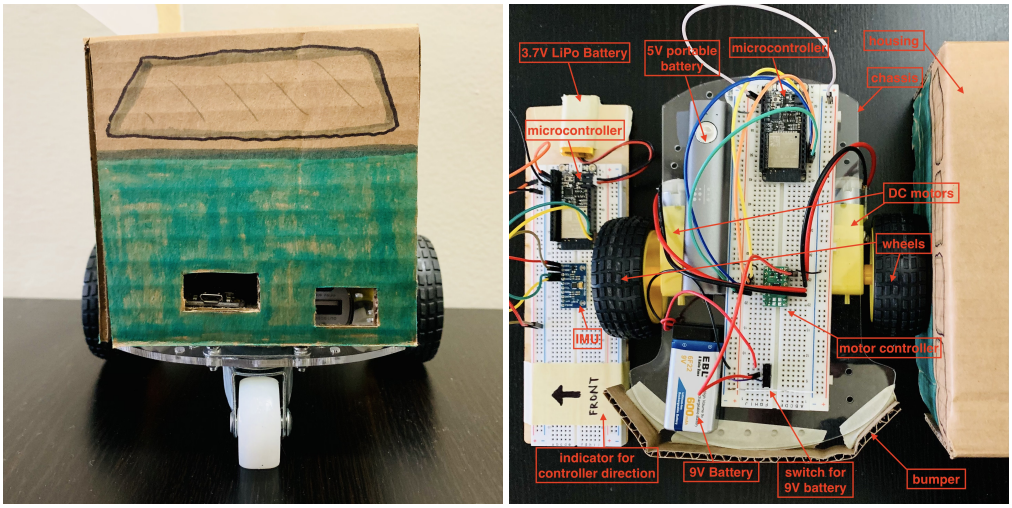
Building the Car

The components of the car are from the two wheel drive robot car kit for Arduino projects. The kit contains the following components.

- 2x DC Motors
- 2x Tire Tread Wheels
- 1x Caster Wheel
- 1x Acrylic Chassis

- 2x Mounting Brackets (screws and nuts included)

In addition to the kit, a breadboard is mounted to the chassis with Velcro. To build this kit, a Phillips head screwdriver is required. After assembly and some testing, I found that the caster wheel was impairing the movement of the car and not sufficient as a third supporting wheel due to the spinning of the wheel. Therefore, I taped the caster wheel in place which significantly improved the movement of the car. For the



(a) Microcontroller access, Microcontroller to battery connection.

(b) Annotated picture of sensors and circuit for the controller and car

Figure 2

cardboard housing, it sits on top on the chassis and held in place with Velcro. This allows the housing to be easily removable so that the 9V and portable battery charger can be recharged. In addition, there are two holes in the back of the car for access to the microcontroller and for the microcontroller to connect to the portable battery.

For future projects, the robot car kit also includes an optical encoder which could be used for feedback control implementation.

Controller Setup

For the prototype, the controller consists of a microcontroller and IMU on a breadboard. The breadboard is also fixed to a piece of laser cut slab of wood for better grip. For future iterations, the controller would be smaller with a and contained inside a ergonomic 3D printer container.

Circuit

The main components of the circuits are (1) two ESP32 feathers (2) two DC motors (3) inertial measurement unit (IMU) (4) dual motor driver carrier.

1. *Two ESP32 Feathers:* In addition to the microkit microcontroller, another microcontroller(esp32 feather, $20) is also used in this project for wireless control through WiFi. The controller microcontroller or the ”server” microcontroller, will create its own WiFi server for the car microcontroller or ”client” microcontroller to grab data. The controller microcontroller was powered by the included 3.7V LiPo battery, and the car microcontroller was powered by an old 5V portable charger.
2. *Two DC Motors:* Both DC motors operate from 4.5-9V. In the circuit, it is powered by a rechargeable 9V battery ($6). This battery is connected to the motor driver through a temporary switch for prototyping purposes. They are controlled through a motor driver to allow the car to have bidirectional motion.

3. *IMU:* The inertial measurement sensor (MPU9250/6500 9-Axis 9 DOF 16 Bit Gyroscope Acceleration Magnetic Sensor IIC/SPI, in microkit) is a chip that contains an accelerometer, gyroscope, and magnetometer. It is part of the controller side circuit and powered by the 3.3V from the microcontroller. It will evaluate the angle at which the user is holding the controller. For implementation in Arduino, this library by hideakitai is used to retrieve the pitch and roll of the controller set up.

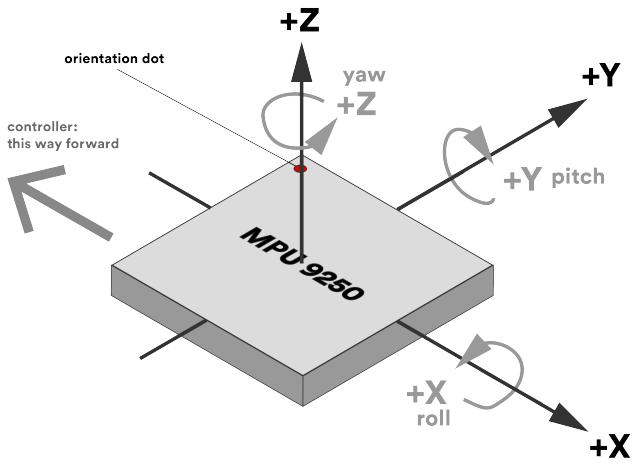
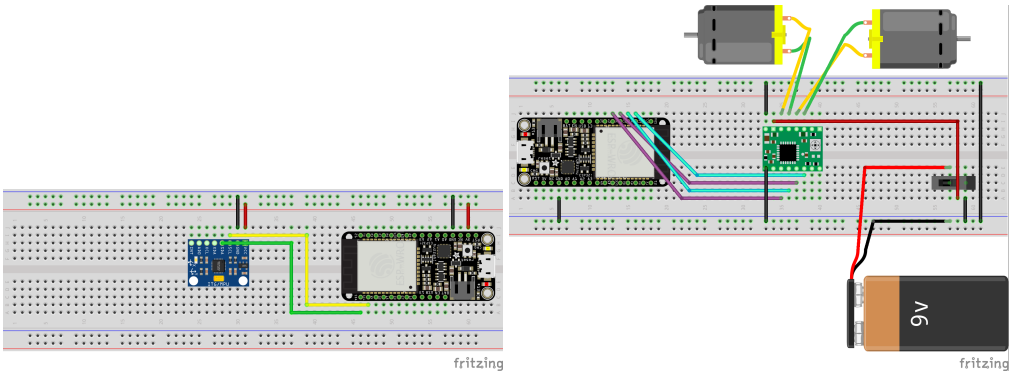


Figure 3: Diagram of IMU orientation. Only the pitch and roll of the IMU is used for this project.

4. *Motor Driver:* The dual motor driver carrier (DRV8833 Dual Motor Driver Carrier, in microkit) is capable of controlling 2 high powered actuators, such as these DC motors, with low power data signals. This motor driver uses an H-bridge to control the DC motors in two directions by switching the polarity of the connected 9V battery.

The total for the parts that were purchased for this project were $26 since the robot car kit, IMU, motor driver, and portable charger were from previous projects or from the ME102B microkit.



(a) Controller circuit diagram (b) Car circuit diagram

Figure 4

Finite State Machine and Arduino Code

Note that hysteresis is implemented as the IMU data is converted to PWM values. This is to prevent any noise and sensor inaccuracies to unnecessarily influence the PWM values of the motor. The hysteresis is not only applied to the back and forth or ”Y” values but also the turning values or ”X” values of the controller.

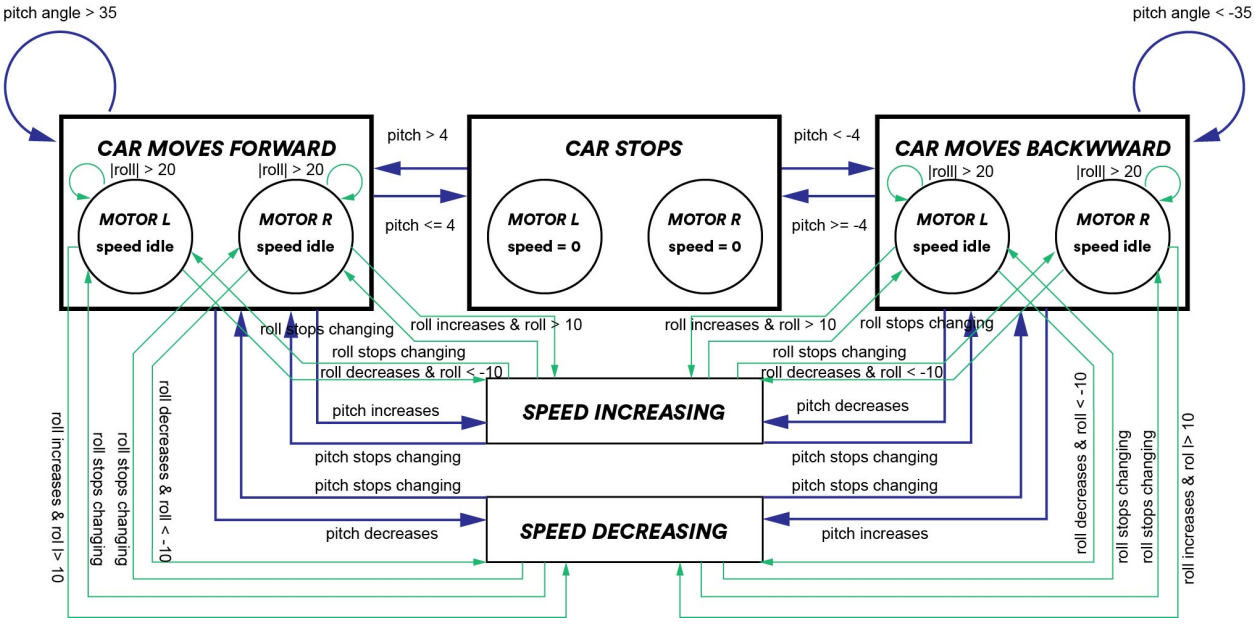


Figure 5: FSM of the system. The blue lines affect both motors identically, green lines affect individual motors for turning motion. Pitch or back and forth motion is capped at a value of 35/-35. Roll or turning motion is capped at 20/-20 so that any numbers not within that range cannot push the motors to full PWM values and stall the motor.

Arduino Code For the Arduino code, the client (car) asks for an update from the server (controller) every 100 ms. In reality, this delay is almost undetectable and the controller to car response is immediate to the user.

Appendix

Controller Code

```
// Import required libraries
#include "WiFi.h"
#include "ESPAsyncWebServer.h"

#include "MPU9250.h"
MPU9250 mpu;


int x;
int y;
const int upLim = 35;
const int loLim = -35;

// Set your access point network credentials
const char* ssid = "ESP32-Access-Point";
const char* password = "123456789";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

String readIMU_X() {
  return String(x);
}

String readIMU_Y() {
  return String(y);
}

void setup() {
  // Serial port for debugging purposes
  Serial.begin(9600);
  Wire.begin();
  delay(1000);
  Serial.println();

  // Setting the ESP as an access point
  Serial.print("Setting AP (Access Point)");
  // Remove the password parameter, if you want the AP (Access Point) to be
     open
  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/imuX", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send_P(200, "text/plain", readIMU_X().c_str());
  });

  server.on("/imuY", HTTP_GET, [](AsyncWebServerRequest * request) {
```

```
    request->send_P(200, "text/plain", readIMU_Y().c_str());
  });

  mpu.setup(0x68);
  mpu.calibrateAccelGyro();

  // Start server
  server.begin();
}

void loop() {
  if (mpu.update()) {
    x = mpu.getRoll();
    y = mpu.getPitch();
  }
  x = limitRange(x);
  y = limitRange(y);
}

int limitRange(int dir) {
  if (dir >= upLim) {
    dir = upLim;
  } else if (dir <= loLim) {
    dir = loLim;
  } else {
    dir = dir;
  }
  return dir;
}
```

Car Code

```
#include "WiFi.h"
#include "HTTPClient.h"

const char* ssid = "ESP32-Access-Point";
const char* password = "123456789";

//Your IP address or domain name with URL path
const char* serverNameX = "http://192.168.4.1/imuX";
const char* serverNameY = "http://192.168.4.1/imuY";

#define motorAPin1  14 //direction 1 for motor A (right)
#define motorAPin2  15 //direction 2 for motor A (right)
#define motorBPin1  32 //direction 1 for motor B (left)
#define motorBPin2  33 //direction 2 for motor B (left)

String read_x;
String read_y;

int pwm_x_L;
int pwm_x_R;
int pwm_y;
```

```
22 int X_val;
23 int Y_val;
24
25 int tot_pwm_L;
26 int tot_pwm_R;
27
28
29 // setting PWM properties
30 const int freq = 20000;
31 const int ledChannel1 = 0;  //direction 1 for motor A (right)
32 const int ledChannel2 = 1;  //direction 2 for motor A (right)
33 const int ledChannel3 = 2;  //direction 1 for motor B (left)
34 const int ledChannel4 = 3;  //direction 2 for motor B (left)
35
36 const int resolution = 8; //corresponds to dutycyle of 0-255
37
38 unsigned long previousMillis = 0;
39 const long interval = 100;
40
41 void setup() {
42   //Start serial
43   Serial.begin(9600);
44
45   // configure MOTOR A PWM functionalitites
46   ledcSetup(ledChannel1, freq, resolution);
47   ledcSetup(ledChannel2, freq, resolution);
48
49   // configure MOTOR B PWM functionalitites
50   ledcSetup(ledChannel3, freq, resolution);
51   ledcSetup(ledChannel4, freq, resolution);
52
53   // attach the channels to the GPIOs to be controlled (A)
54   ledcAttachPin(motorAPin1, ledChannel1);
55   ledcAttachPin(motorAPin2, ledChannel2);
56
57   // attach the channel to the GPIOs to be controlled (B)
58   ledcAttachPin(motorBPin1, ledChannel3);
59   ledcAttachPin(motorBPin2, ledChannel4);
60
61   WiFi.begin(ssid, password);
62   Serial.println("Connecting");
63   while (WiFi.status() != WL_CONNECTED) {
64     delay(500);
65     Serial.print(".");
66   }
67   Serial.println("");
68   Serial.print("Connected␣to␣WiFi␣network␣with␣IP␣Address:␣");
69   Serial.println(WiFi.localIP());
70
71 }
72
73 void loop() {
74   unsigned long currentMillis = millis();
75
```

```
76    if (currentMillis - previousMillis >= interval) {
77      // Check WiFi connection status
78      if (WiFi.status() == WL_CONNECTED ) {
79        read_x = httpGETRequest(serverNameX);
80        read_y = httpGETRequest(serverNameY);
81        //Serial.println("X: " + read_x + " Y: " + read_y);
82
83        // save the last HTTP GET Request
84        previousMillis = currentMillis;
85      }
86      else {
87        Serial.println("WiFi Disconnected");
88      }
89    }
90
91    X_val = read_x.toInt();
92    Y_val = read_y.toInt();
93
94    motorDrive();
95    //Serial.print("left: ");
96    //Serial.print(tot_pwm_L);
97    //Serial.print(" right: ");
98    //Serial.println(tot_pwm_R);
99
100 }
101
102 /* SERVICE ROUTINE: motorDrive()
103     Decription: Function to drive the motor forward and backward (drive both
            motors at same PWM)
104 */
105 void motorDrive() {
106   /*Control the PWM signal for each motor only if the angle of the
107     controller is higher or lower than -4*/
108   motorTurn();
109
110   if (Y_val > 4) {
111     pwm_y = map(Y_val, 0, 35, 100, 230); //Map the angle to a PWM signal from
            0 to 200
112
113     tot_pwm_L = pwm_y + pwm_x_L;
114     tot_pwm_R = pwm_y + pwm_x_R;
115
116     //    tot_pwm_L = pwm_y;
117     //    tot_pwm_R = pwm_y;
118
119     constrain(tot_pwm_L, 0, 255);
120     constrain(tot_pwm_R, 0, 255);
121
122     ledcWrite(ledChannel1, 0);
123     ledcWrite(ledChannel2, tot_pwm_R);
124
125     ledcWrite(ledChannel3, 0);
126     ledcWrite(ledChannel4, tot_pwm_L);
127   }
```

```
128   //We do the same for all 4 PWM pins
129   if (Y_val < -4) {
130     pwm_y = map(Y_val, 0, -35, 100, 230);
131
132     tot_pwm_L = pwm_y + pwm_x_L;
133     tot_pwm_R = pwm_y + pwm_x_R;
134
135     //    tot_pwm_L = pwm_y;
136     //    tot_pwm_R = pwm_y;
137
138     constrain(tot_pwm_L, 0, 255);
139     constrain(tot_pwm_R, 0, 255);
140
141     ledcWrite(ledChannel1, tot_pwm_R);
142     ledcWrite(ledChannel2, 0);
143
144     ledcWrite(ledChannel3, tot_pwm_L);
145     ledcWrite(ledChannel4, 0);
146   }
147
148   if (Y_val > -4 && Y_val < 4) {
149     ledcWrite(ledChannel1, 0);
150     ledcWrite(ledChannel2, 0);
151
152     ledcWrite(ledChannel3, 0);
153     ledcWrite(ledChannel4, 0);
154   }
155 }
156
157
158 /* SERVICE ROUTINE: motorTurn()
159     Decription: Function to drive the motor to turn
160     X_val positive = left
161     X_val negative = right
162 */
163 void motorTurn() {
164   if (X_val > 10) {
165     pwm_x_L = map(X_val, 10, 35, 0, -20);
166     pwm_x_R = map(X_val, 10, 35, 0, 20);
167   }
168
169   if (X_val < -10) {
170     pwm_x_L = map(X_val, -10, -35, 0, 20);
171     pwm_x_R = map(X_val, -10, -35, 0, -20);
172   }
173
174   if (X_val > -10 && X_val < 10) {
175     pwm_x_L = 0;
176     pwm_x_R = 0;
177   }
178 }
179
180 String httpGETRequest(const char* serverName) {
181   HTTPClient http;
```

```
182
183   // Your IP address with path or Domain name with URL path
184   http.begin(serverName);
185
186   // Send HTTP POST request
187   int httpResponseCode = http.GET();
188
189   String payload = "--";
190
191   if (httpResponseCode > 0) {
192     //Serial.print("HTTP Response code: ");
193     //Serial.println(httpResponseCode);
194     payload = http.getString();
195   }
196   else {
197     Serial.print("Error code: ");
198     Serial.println(httpResponseCode);
199   }
200   // Free resources
201   http.end();
202
203   return payload;
204 }
```