



Assembly of circuit with MPU Gyroscope for car and sending IoT data

Difficulty level: Difficult

Goals

Automotive IoT is the integration of gadgets, sensors, cloud computing, applications, and other such components into vehicles to function as a complex system for the connection of cars, predictive maintenance, fleet management, OEMs, insurance, and more.

The integration of the Internet of Things in the automotive industry allows manufacturers to implement sought-after innovations that can ultimately transform cars into near-artificial intelligence. At a didactic level, we are now going to develop some exercises using sensors for data acquisition, processed by the Arduino microcontroller.

This exercise intends to apply a gyroscope guided by a microcontroller to autonomously control and correct the direction of a vehicle in the event of a skid (for example) to preserve the safety of its occupants. This exercise is aided by three LEDs that help you understand which way a possible skid is happening. It is also possible to use this type of sensor to detect the inclination of a vehicle and can autonomously assist the force to be applied to the engine in case of going uphill or help the braking system in case of going downhill.

For the possible sending of data, it will be necessary to apply, for example, the ESP8266 ESP-01 module that allows the connection of several devices to the internet (or local network), and consequent sending of data from the sensors applied to the autonomous system.



Image-1: Understanding the application of MPU Gyroscope in a car and communicating with IoT.

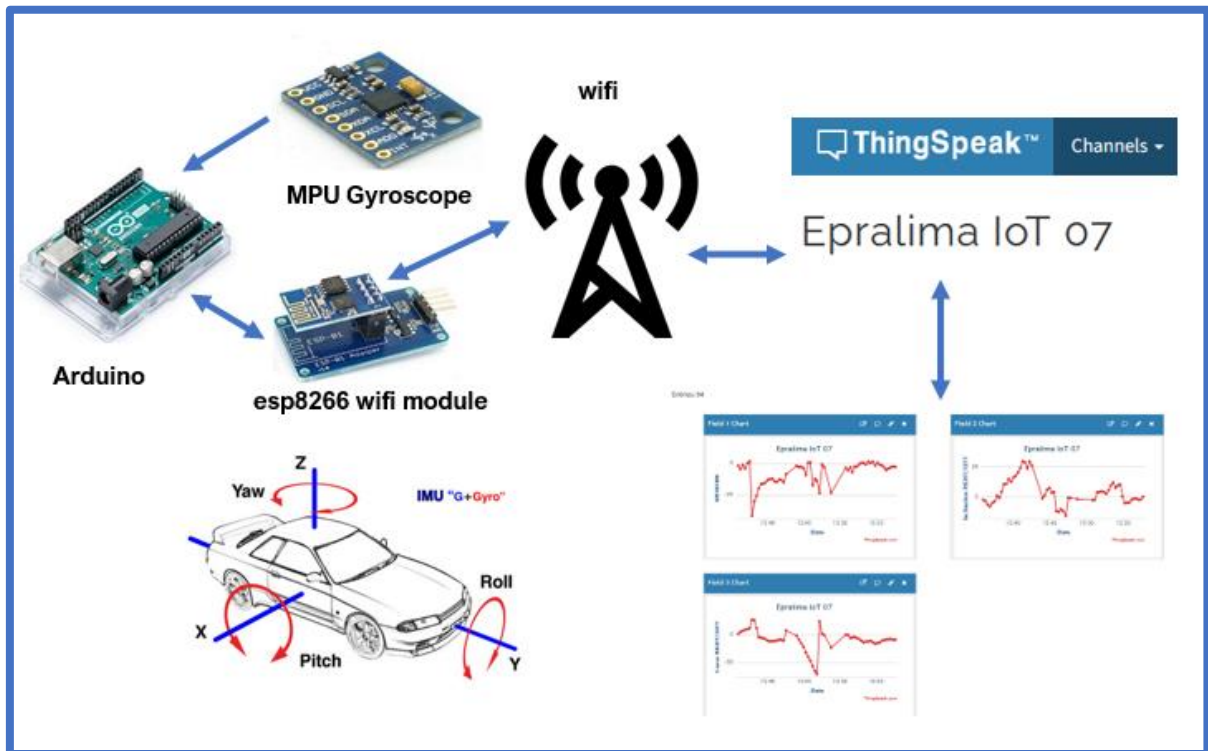


Image 1: application of MPU Gyroscope in a car and communicating with IoT

Skills

- The skills our students will gain are:
- Students' ability to build circuits will be developed.
- The ability to program the Arduino board and use the ESP8266 Module for Internet access will develop.
- The ability to receive data from the brightness sensor and send the received data to Thing Speak will be gained.
- Data analytics will improve their ability to connect with the Internet of Things.

Required materials and circuit diagram.

In this exercise we intend to learn how to draw diagrams (circuits), connect all the components correctly, develop software based on C language (Arduino), connect to the







wifi network, communicate with an IoT server, ThingSpeak and read server-generated graphics.

Quantity	Component
1	Arduino Uno R3
1	ESP01-8266
1	Power Supply (braedBoard)
1	BreadBoard
1	MPU6050
3	Led Green, Red and Blue
3	Resistor 330Ohm

Table 1 - Components List

Materials table

 Arduino	 ESP01 - 8266
 Bread Board + Power Supply	 MPU6050

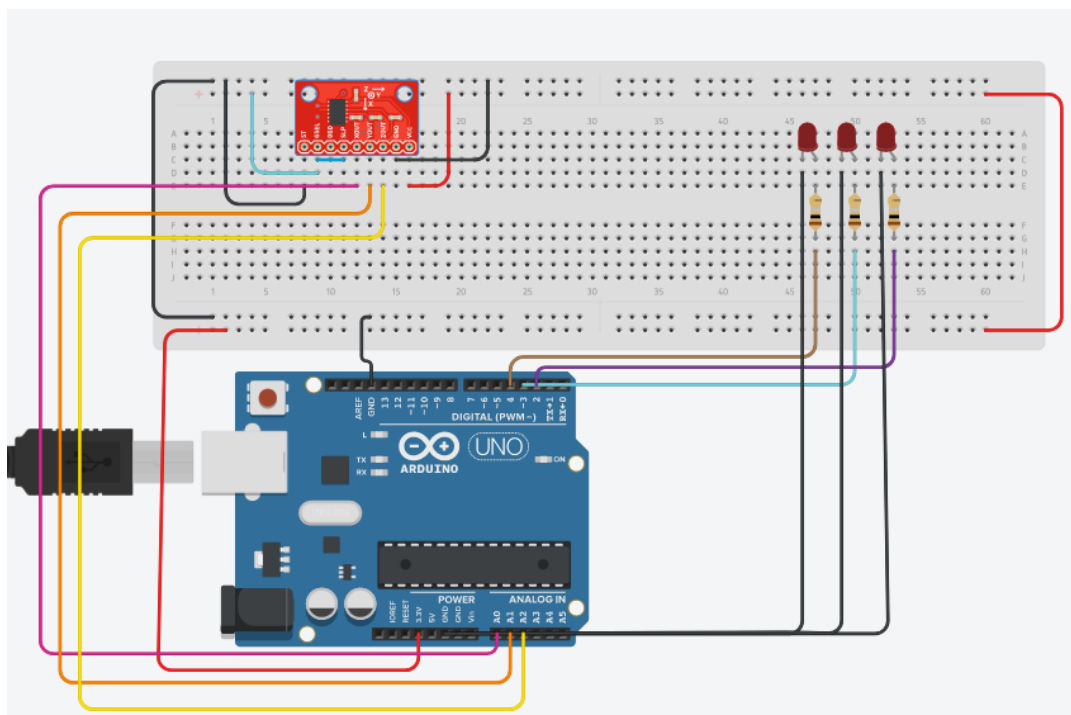
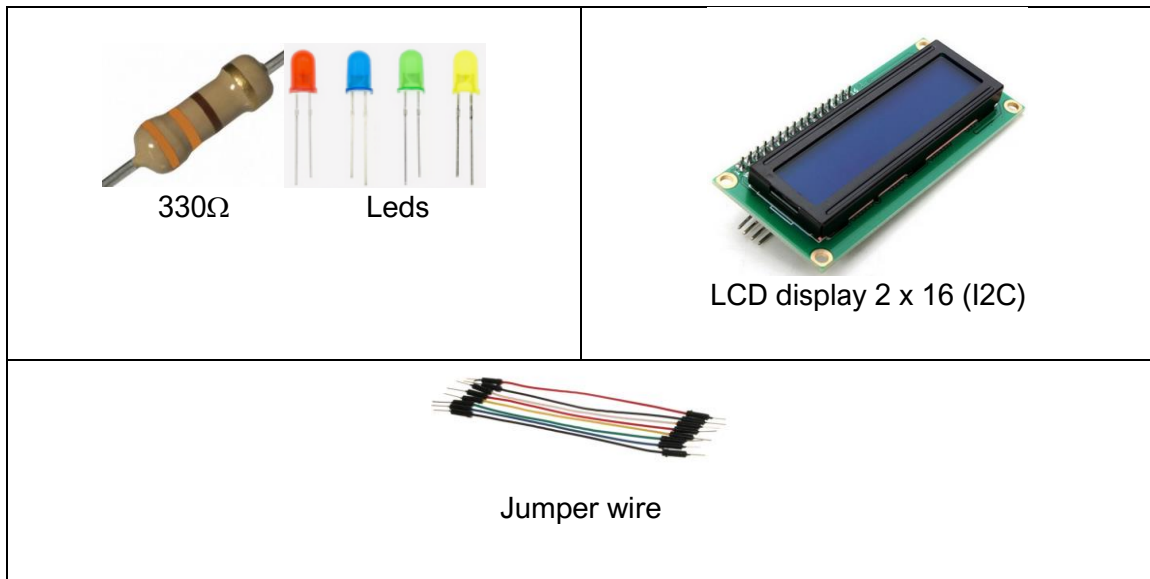


Image 2 – Diagram circuit

Implementation

Development of communication of microcontroller systems, and sensors, with the ThingSpeak IoT cloud.

The ESP8266 WiFi module (image 3) is a small shield with integrated TCP/IP protocol that can give any microcontroller access to the WiFi network. The ESP8266 is capable



of both hosting an application and offloading all WiFi network functions from another application processor. Each ESP8266 module is pre-programmed with an AT command making its firmware settings, meaning that we can simply connect this module to the Arduino working as any other WiFi shield would. This module has a great cost/benefit ratio and has a very large and constantly growing user community.

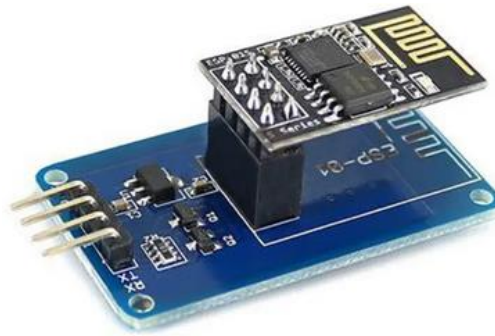


Image 3 - ESP01 – 8266

The MPU6050 is a Micro-Electro-Mechanical Systems (MEMS) that consists of a 3-axis Accelerometer and 3-axis Gyroscope inside it. This helps us to measure acceleration, velocity, orientation, displacement and many other motion-related parameters of a system or object.

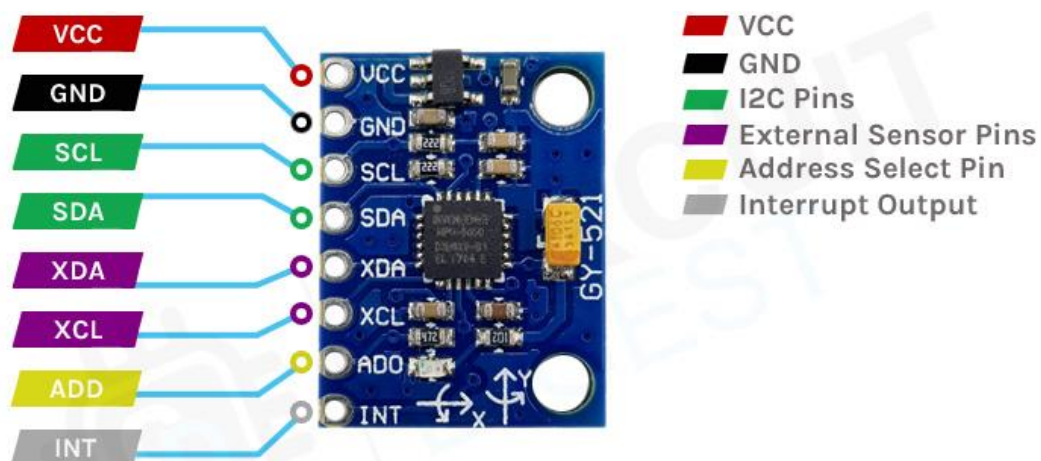


Image 4 MPU 6050

Implementation in practice

1. Assemble the circuit in the image 2;
2. Connect correctly ESP01-8266 image 5



Image 6 Real circuit in breadboard



4. Create a ThingSpeak account image 7

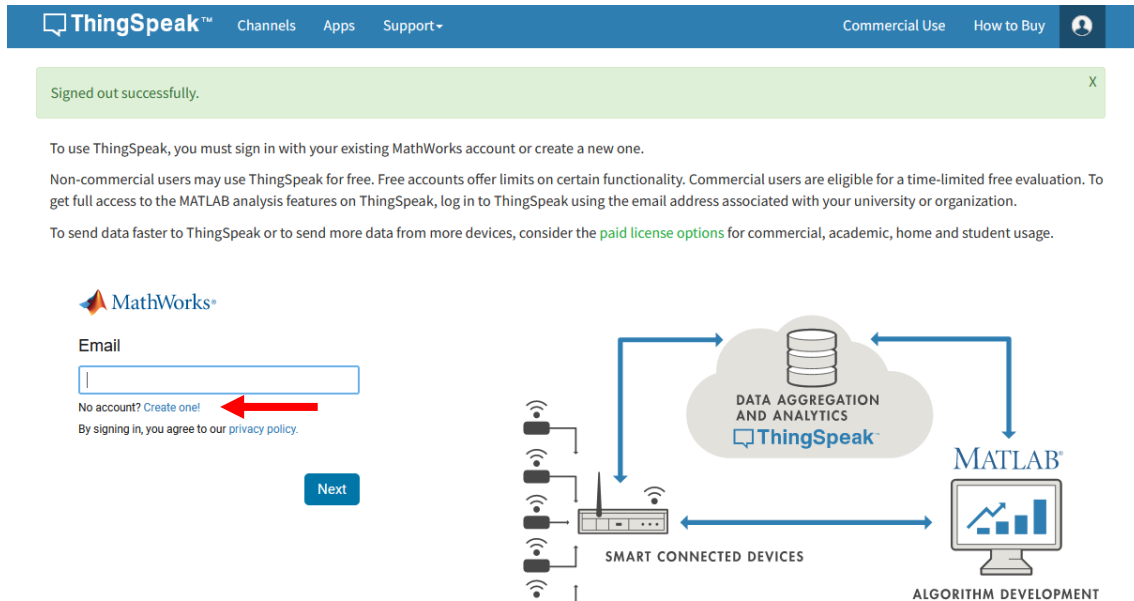


Image 7 - Thing Speak

5. Create a new channel image 8



Image 8 Interface ThingSpeak

6. Configure channel, with name, description, and fields. Image 9.

Note: The fields refer to data processed by the microcontroller and data from the sensors under study. Each field will generate a graph.



Epralima IoT 01

Channel ID: 2064208

Author: mwa0000029483576

Access: Private

This Channel presents the simulation of car night lighting based on the level and intensity of natural light

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

Channel Settings

Percentage complete 50%

Channel ID 2064208

→ Name Epralima IoT 01

→ Description This Channel presents the simulation of car night lighting based on the level and intensity of natural light

→ Field 1 Light level ☒

→ Field 2 State lights ON OFF ☒

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.

Image 9 Configure Channel

7. Save settings channel Image 10

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy AC

station that acquires data from an Arduino® device. [Learn More](#)

Link to GitHub

Elevation

Show Channel Location ☐

Latitude

Longitude

Show Video ☐

☒ YouTube ☐ Vimeo

Video URL

Show Status ☐

→ Save Channel

Image 10 Save settings channel

8. In this step, we will pay special attention to the api keys, as they are the ones that, through the string key, will allow access to the IoT repository in Arduino programming. Also very important are the API requests.



ThingSpeak™

Channels ▾ Apps ▾ Devices ▾ Support ▾

Commercial Use How to Buy AC

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Write API Key

→ Key

Generate New Write API Key

Read API Keys

→ Key

Note

Save Note Delete API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

GET `https://api.thingspeak.com/update?api_key=UC[redacted]CP&field1`

Read a Channel Feed

GET `https://api.thingspeak.com/channels/[redacted]feeds.json?api_key=D8`

Image 11 - API Keys



9. Programming Arduino

Inclusion of the necessary libraries and declaration of variables and constants inherent to the program's operation.

```
1 #include <SoftwareSerial.h>
2 #include <MPU6050_tockn.h>
3 #include <Wire.h>
4
5 SoftwareSerial ESP_Serial(10, 11); //PINOS QUE EMULAM A SERIAL, OI
6                                     // RX_AUX  --liga--> TX do ESP01
7                                     // TX_AUX  --liga--> RX do ESP01
8 #define serialcomSpeed 115200
9 #define DEBUG true
10 #define pinLedX 5
11 #define pinLedY 6
12 #define pinLedZ 7
13 #define MPU6050_ADDR 0x68
14
15 MPU6050 mpu6050(Wire);
16
17 float anguloX;
18 float anguloY;
19 float anguloZ;
20 unsigned long controleTempo;
21 long writingTimer = 17;
22 long startTime = 0;
23 long waitTime = 0;
24
25 boolean error;
26
27 String APIKey = "6";
```

Void setup() function for initializing parameters for starting the program.

```
34 void setup() {
35   Serial.begin(serialcomSpeed);
36   ESP_Serial.begin(serialcomSpeed);
37   startTime = millis();
38   InitWifiModuleESP();
39   Wire.begin();
40   mpu6050.begin();
41   mpu6050.calcGyroOffsets(false);
42   pinMode(pinLedX, OUTPUT);
43   pinMode(pinLedY, OUTPUT);
44   pinMode(pinLedZ, OUTPUT);
45   digitalWrite(pinLedX, HIGH);
46   digitalWrite(pinLedY, HIGH);
47   digitalWrite(pinLedZ, HIGH);
48   delay(500);
49   digitalWrite(pinLedX, LOW);
50   digitalWrite(pinLedY, LOW);
51   digitalWrite(pinLedZ, LOW);
52   // -----
53 }
```



AT commands

AT commands are the basic way to configure and trigger the ESP8266 when it is under control of an external device (like an Arduino, for example).

Current AT commands are direct descendants of the so-called "Hayes Standard" from 1981, used to allow personal computers to interact with telephone connections by directly controlling a mode.

The **InitWifiModule()** function initializes the ESP8266 through AT commands.

```
46 void InitWifiModuleESP() {
47     //Este procedimento envia os COMANDOS AT para o ESP 01
48     envioDadosESP_AT("AT+RST\r\n", 2000, DEBUG); //faz reset ao modulo;
49     envioDadosESP_AT("AT+CWMODE=1\r\n", 1500, DEBUG);
50     delay(100);
51     envioDadosESP_AT("AT+CWJAP=\"Epralima|\", \"*****\r\n\", 2000, DEBUG);
52     delay(500);
53     envioDadosESP_AT("AT+CIFSR\r\n", 1500, DEBUG);
54     delay(100);
55     envioDadosESP_AT("AT+CIPMUX=0\r\n", 1500, DEBUG);
56     delay(100);
57 }
```

The **envioDadosESP_AT(str,int,boolean)** function is responsible for sending AT commands to the ESP8266

```
59 String envioDadosESP_AT(String comando, const int timeout, boolean debug)
60 {
61     String resposta = "";
62     ESP_Serial.println(comando);
63     long int tempo = millis();
64     while((tempo+timeout) > millis()){
65         while(ESP_Serial.available()){
66             char c = ESP_Serial.read();
67             resposta+=c;
68         }
69     }
70     if(debug){
71         Serial.print(resposta);
72     }
73     return resposta;
74 }
```



The **startThingSpeakCmd(str,int,boolean)** function opens connection to ThingSpeak IoT analytics platform. The IP address of the ThingSpeak platform is: 184.106.153.149 with connection on port 80. The AT command to start ThingSpeak communication is AT+CIPSTART=PROTOCOL, IP_ADRESS, PORT.

```
106 void startThingSpeakCmd(void) {
107     ESP_Serial.flush();
108     String cmd="";
109     cmd = "AT+CIPSTART=\\"TCP\\",\\"";
110     cmd+="184.106.153.149";//Endereco IP thingerSpeak
111     cmd+="\\"",80";
112     ESP_Serial.println(cmd);|
113     Serial.print("Start Commands: ");
114     Serial.println(cmd);
115     if(ESP_Serial.find("Error"))
116     {
117         Serial.println("AT+CIPSTART error");
118         return;
119     }
120 }
```

The **EscreverParaThingSpeak** function generates a string to build an API Request.

Example:

GET /update?api_key=U.....P&field1= 0&field2= 0

```
100 void EscreverParaThingSpeak(void) {
101
102     startThingSpeakCmd();
103     String getStr = "";
104     getStr = "GET /update?api_key=";
105     getStr += APIKey;
106     getStr += "&field1=";
107     getStr += String(anguloX);
108     getStr += "&field2=";
109     getStr += String(anguloY);
110     getStr += "&field3=";
111     getStr += String(anguloZ);
112     getStr += "\r\n\r\n";
113     GetThingSpeakcmd(getStr);
114 }
115
```



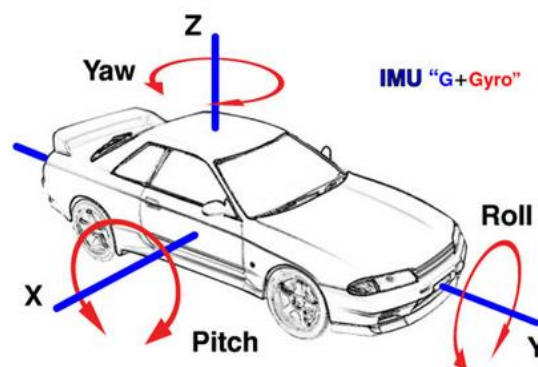
The **GetThingSpeak(str)** function, is responsible for determining and sending an API Request through the AT+CIPSEND command to write to the ThingSpeak channel, returning the message received by the response from the ThingSpeak data platform. The communication will be closed if the response is not favourable.

```
122 String GetThingSpeakcmd(String getStr) {
123
124   String command="";
125   command = "AT+CIPSEND=";
126   command += String(getStr.length());
127   ESP_Serial.println(command);
128   Serial.println(command);
129   int result = ESP_Serial.find(">");
130   if(result==1)
131   {
132     Serial.print("String GET--> ");
133     Serial.println(getStr);
134     ESP_Serial.print(getStr);
135     Serial.println(getStr);
136     delay(500);
137     String messageBody = "";
138     String linha="";
139     while(ESP_Serial.available()){
140       linha = ESP_Serial.readStringUntil('\n');
141       if(linha.length() == 1){
142         messageBody = ESP_Serial.readStringUntil('\n');
143       }
144     }
145     Serial.print("MessageBody received: ");
146     Serial.print(messageBody);
147     return messageBody;
148   }
```



The `inclinationUP_DOWN()`, `inclinationRIGHT_LEFT()` and `rotation()` These procedures update the angle of the MPU 6050 module.

```
115 void inclinationUP_DOWN() {
116     mpu6050.update();
117     anguloX = mpu6050.getAngleX();
118     if (anguloX >= 10) { // Car UP
119         digitalWrite(pinLedX, HIGH);
120     } else if (anguloX <= -10) { // Car DOWN
121         digitalWrite(pinLedX, HIGH);
122     } else {
123         digitalWrite(pinLedX, LOW);
124     }
125 }
126 void inclinationRIGHT_LEFT() {
127     mpu6050.update();
128     anguloY = mpu6050.getAngleY();
129     if (anguloY >= 10) { // car leaning to the right
130         digitalWrite(pinLedY, HIGH);
131     } else if (anguloY <= -10) { // car leaning to the left
132         digitalWrite(pinLedY, HIGH);
133     } else {
134         digitalWrite(pinLedY, LOW);
135     }
136 }
137 void rotation() {
138     mpu6050.update();
139     anguloZ = mpu6050.getAngleZ();
140     if (anguloZ >= 15 || anguloZ <= -10) {
141         digitalWrite(pinLedZ, HIGH);
142     } else {
143         digitalWrite(pinLedZ, LOW);
144     }
```



`inclinationUP_DOWN()` Pitch

`inclinationRIGHT_LEFT()` Roll

`rotation` Yaw



Results

In the analysis of the graphs, it is possible to observe that there was a negative angle that could mean that a car could be making a downhill road. It is possible to verify that there was a variation in the inclination of the car, which means that the road may have a small inclination in roll. It is possible to analyse the number of curves made by a car.

Entries: 64

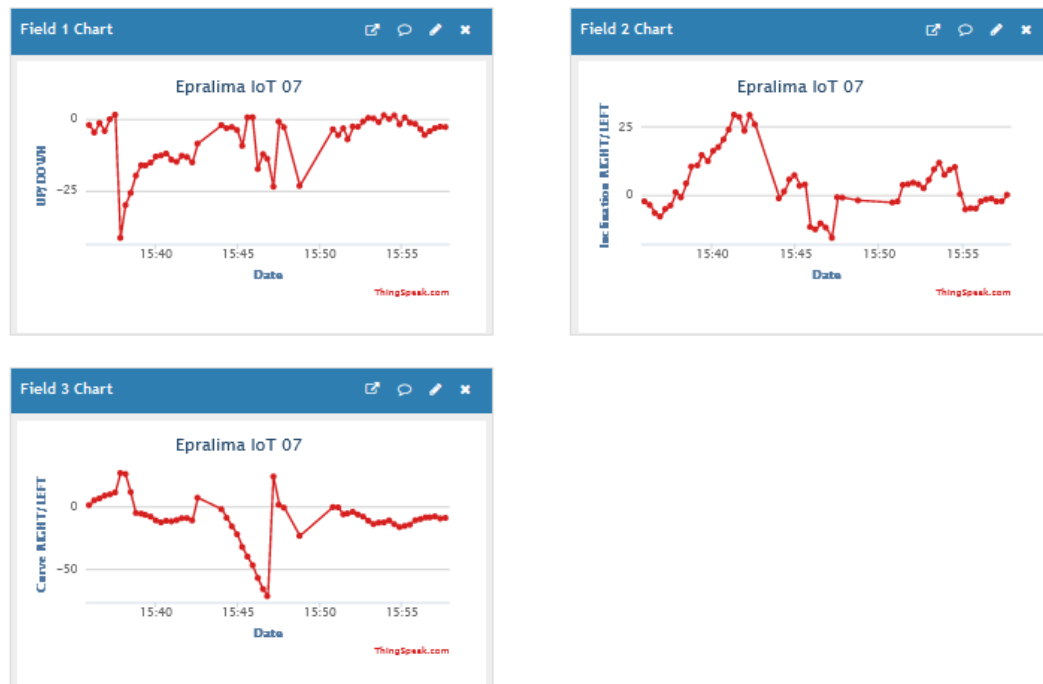


Image 12 – Results IoT ThingSpeak



The data acquired by the ThingSpeak IoT platform can also be exported to CSV files and consequently imported into datasheets as shown in Table 2

created at	entry_id	field1	field2	field3
20/04/2023 15:31	1	-2.36	-2.20	-0.04
20/04/2023 15:32	2	-0.78	2.79	-4.43
20/04/2023 15:33	3	-1.52	-0.71	0.63
20/04/2023 15:33	4	2.22	-5.09	-1.18
20/04/2023 15:34	5	-1.06	-2.38	0.31
20/04/2023 15:35	6	-2.10	-2.17	1.12
20/04/2023 15:36	7	-4.72	-3.47	4.93
20/04/2023 15:36	8	-1.42	-6.43	6.47
20/04/2023 15:36	9	-4.20	-7.72	8.69
20/04/2023 15:37	10	-0.06	-4.99	9.66
20/04/2023 15:37	11	1.47	-3.72	11.20
20/04/2023 15:37	12	-41.56	1.15	26.73
20/04/2023 15:38	13	-30.08	-0.76	25.98
20/04/2023 15:38	14	-25.85	4.36	11.52
20/04/2023 15:38	15	-19.79	10.50	-5.23
20/04/2023 15:39	16	-16.16	10.95	-5.71
20/04/2023 15:39	17	-16.28	14.70	-6.75
20/04/2023 15:39	18	-15.26	12.49	-8.07
20/04/2023 15:40	19	-13.15	16.21	-11.09
20/04/2023 15:40	20	-12.76	17.61	-12.70

Tabela 2 - DataSheet

In short

This exercise intends to apply a gyroscope guided by a microcontroller to autonomously control and correct the direction of a vehicle in the event of a skid (for example) in order to preserve the safety of its occupants. This exercise is aided by three LEDs that help you understand which way a possible skid is happening.

It is also possible to use this type of sensor to detect the inclination of a vehicle and can autonomously assist the force to be applied to the engine in case of going uphill or help the braking system in case of going downhill.

For possible sending of data, it will be necessary to apply, for example, the ESP8266 ESP-01 module that allows the connection of several devices to the internet (or local network), and consequent sending of data from the sensors applied to the autonomous system.