



Assembly of circuit with Ultrasonic sensor for car and sending IoT data

Difficulty level: Medium

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 proximity sensor driven by a microcontroller to avoid collisions autonomously and protect passengers. This exercise is aided by a buzzer to emit sound signals, the closer the greater the sound alerts, as well as a red LED that should light up in case of extreme proximity.

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 Ultrasonic sensor in a car and communicating with IoT.

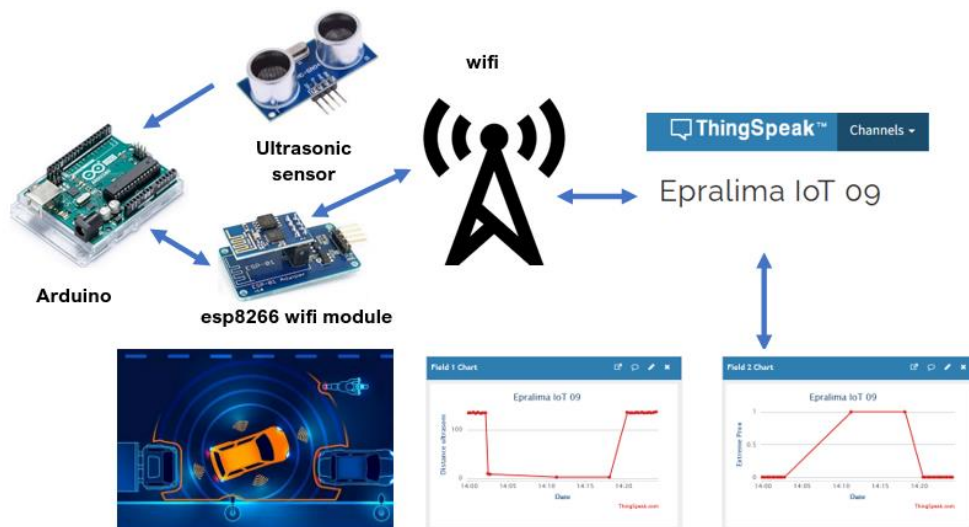


Image 1: application of Ultrasonic sensor in a car and communicating with IoT



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.


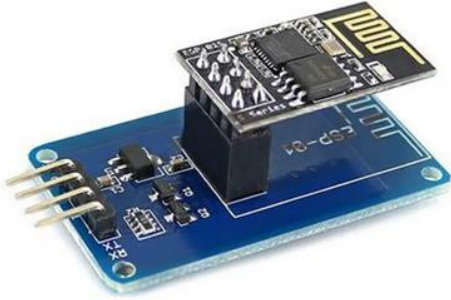
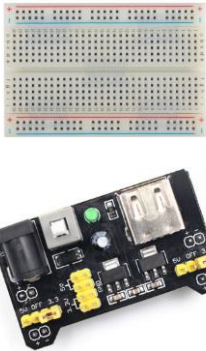

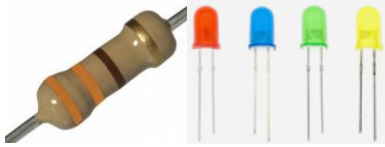


In this exercise we intend to learn how to draw diagrams (circuits), connect all the components correctly, develop software based on C language (Arduino), connect to the wifi network, communicate with an IoT server, ThingSpeak and read server-generated graphics.

Quantity	Component
1	Arduino Uno R3
1	ESP01-8266
1	Power Supply (braedBoard)
1	BreadBoard
1	HC-SR04Sensor
1	Buzzer active
2	Led Green, Red and Blue
2	Resistor 330Ohm

Table 1 - Components List



Materials table

	
Arduino	ESP01 - 8266
	
Bread Board + Power Supply	HC-SR04
	
330 Ω Leds	Buzzer Active
	
Jumper wire	

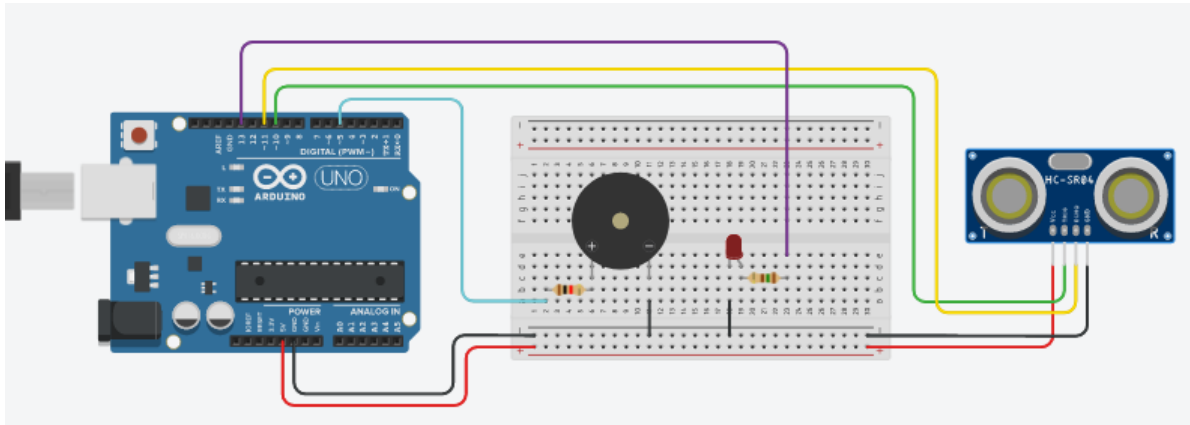


Image 2 – Diagram circuit

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.

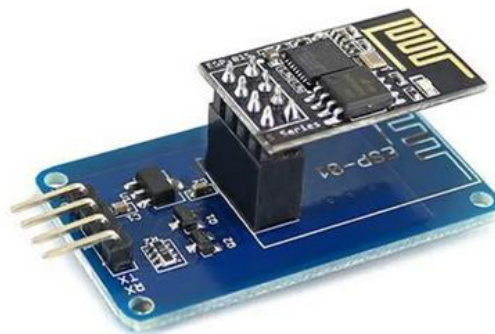


Image 3 - ESP01 – 8266



HC-SR04 is a widely used and popular sensor that can be used for measuring the distance between itself and the object that is placed in front of the sensor. Connecting to Arduino itself is simple, the sensor requires only two codes for controlling and uses 5V voltage. There is also a library that provides us with distance information.

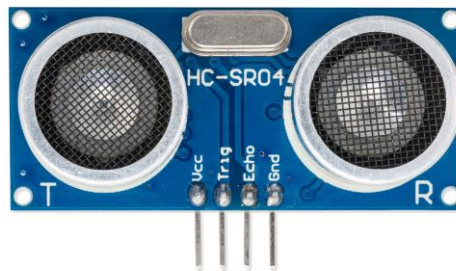


Image 4 HC

Implementation in practice

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

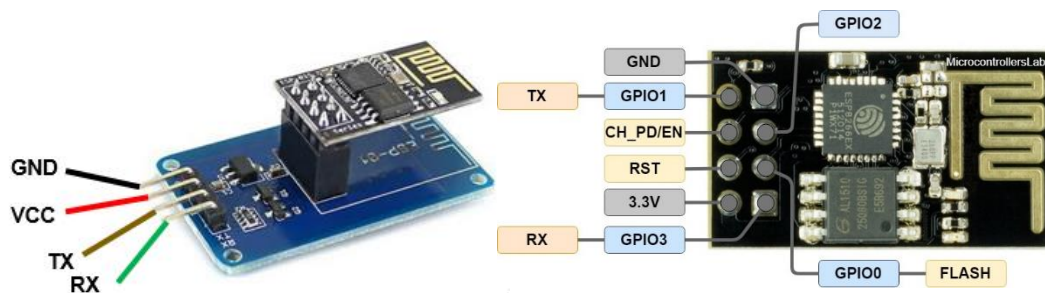


Image 5 ESP-01 Connections



3. Real assembled circuit image 6

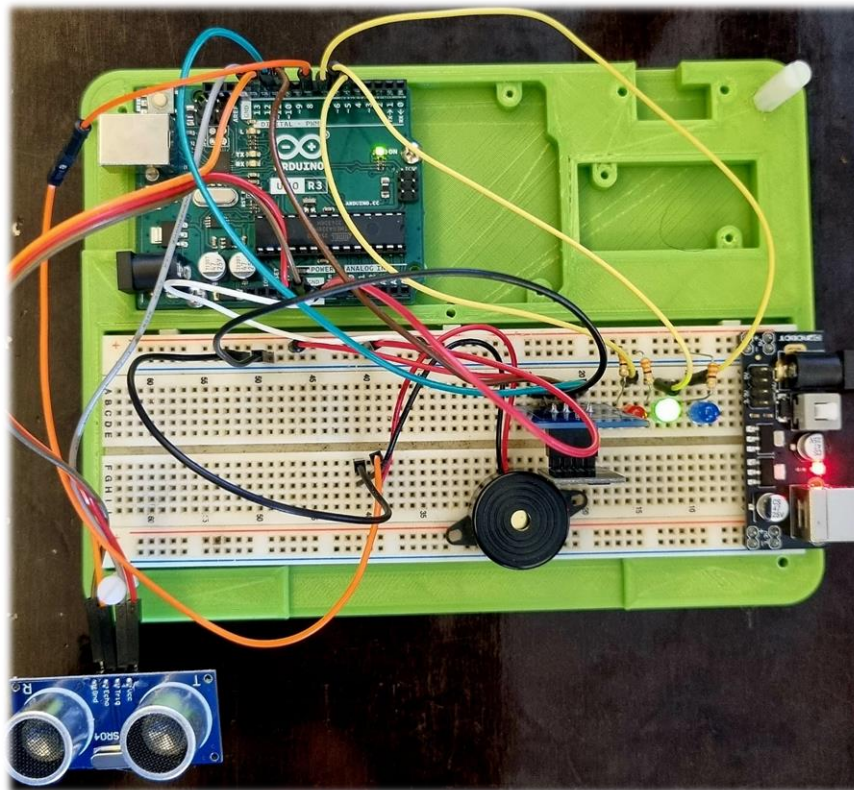


Image 6 Real circuit in breadboard

4. Create a ThingSpeak account image 7

ThingSpeak™ Channels Apps Support

Commercial Use How to Buy

Signed out successfully.

To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the [paid license options](#) for commercial, academic, home and student usage.

MathWorks®

Email

No account? [Create one!](#)

By signing in, you agree to our [privacy policy](#).

Next

SMART CONNECTED DEVICES

DATA AGGREGATION AND ANALYTICS
ThingSpeak

MATLAB®

ALGORITHM DEVELOPMENT

Image 7 - Thing Speak



5. Create a new channel image 8

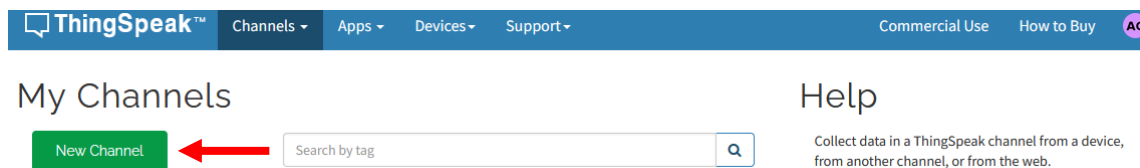


Image 8 Interface ThingSpeak

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.

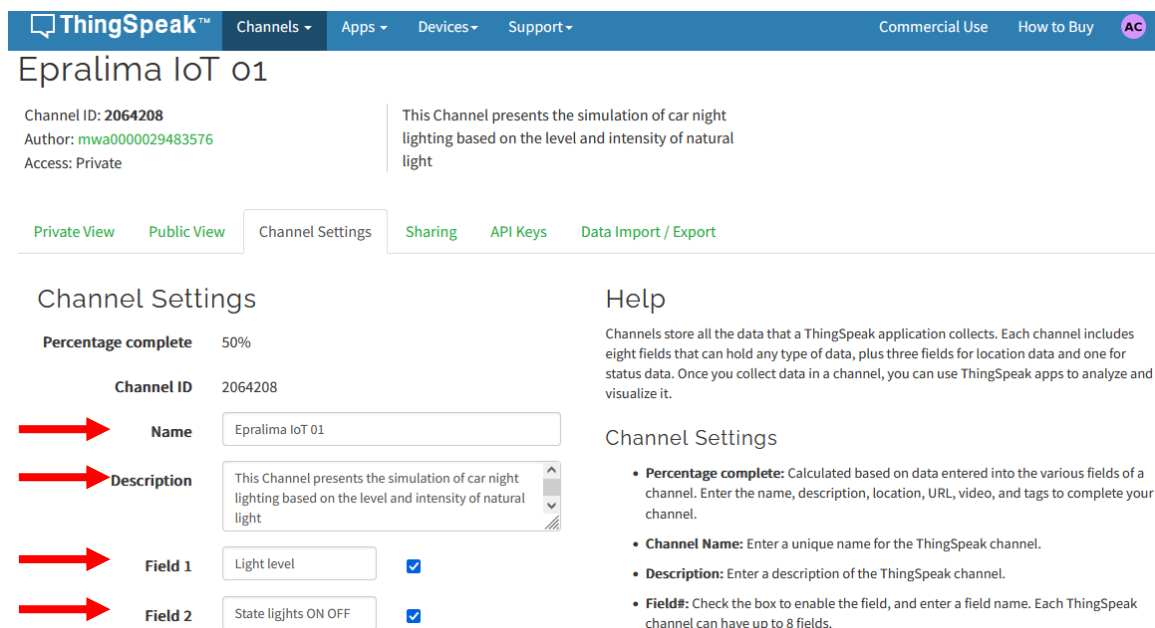


Image 9 Configure Channel



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]&field1=<input>
```

Read a Channel Feed

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

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 <Wire.h>
3 #include <HCSR04.h>
4 SoftwareSerial ESP_Serial(10, 11); //PINOS QUE EM
5                                     // RX_AUX  --lig
6                                     // TX_AUX  --lig
7
8 #define serialcomSpeed 115200
9 #define DEBUG true
10 #define p_trigger 13
11 #define p_echo 12
12 #define buzzer 8
13 #define ledRed 5
14 #define ledGreen 6
15
16 UltraSonicDistanceSensor distanceSensor(p_trigger,
17
18 int distance=0;
19 int extrem_proximity=0;
20 unsigned long controleTempo;
21 long writingTimer = 17;
22 long startTime = 0;
23 long waitTime = 0;
24
25 boolean error;|
26 String APIKey = "          ";
27 void ativarBuzzer(void);
28 void buzzerStop(void);
29 void readDistance(void);
```

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

```
32 void setup() {
33   Serial.begin(serialcomSpeed);
34   ESP_Serial.begin(serialcomSpeed);
35   startTime = millis();
36   InitWifiModuleESP();
37   Wire.begin();
38   pinMode(buzzer, OUTPUT);
39   pinMode(ledRed, OUTPUT);
40   pinMode(ledGreen, OUTPUT);
41   // -----
42 }
43
```



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

```
87 void EscreverParaThingSpeak(void) {
88
89     startThingSpeakCmd();
90     String getStr = "";
91     getStr = "GET /update?api_key=";
92     getStr += APIKey;
93     getStr += "&field1=";
94     getStr += String(distance);
95     getStr += "&field2=";
96     getStr += String(extrem_proximity);
97     getStr += "\r\n\r\n";
98     GetThingSpeakcmd(getStr);
99 }
```



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 `ativarBuzzer()`, `buzzerStop()` e `readDistance()` These functions read distance and active/stop buzzer.

```
101 void ativarBuzzer(void) {
102     tone(buzzer, 988, 8);
103     delay(10);
104     tone(buzzer, 0, 8);
105     delay(10);
106 }
107 void buzzerStop(void) {
108     tone(buzzer, 0, 8);
109 }
110 void readDistance(void) {
111     distance = distanceSensor.measureDistanceCm();
112     if (distance <= 5) {
113         ativarBuzzer();
114         extrem_proximity = 1;
115         digitalWrite(ledRed, HIGH);
116         digitalWrite(ledGreen, LOW);
117     } else if (distance > 5 && distance <= 60) {
118         buzzerStop();
119         digitalWrite(ledRed, LOW);
120         digitalWrite(ledGreen, HIGH);
121         extrem_proximity = 0;
122     } else {
123         buzzerStop();
124         digitalWrite(ledRed, LOW);
125         digitalWrite(ledGreen, HIGH);
126         extrem_proximity = 0;
127     }
128 }
```

Results

Analysing the graphics, it is possible to observe the distance of obstacles with extreme proximity.

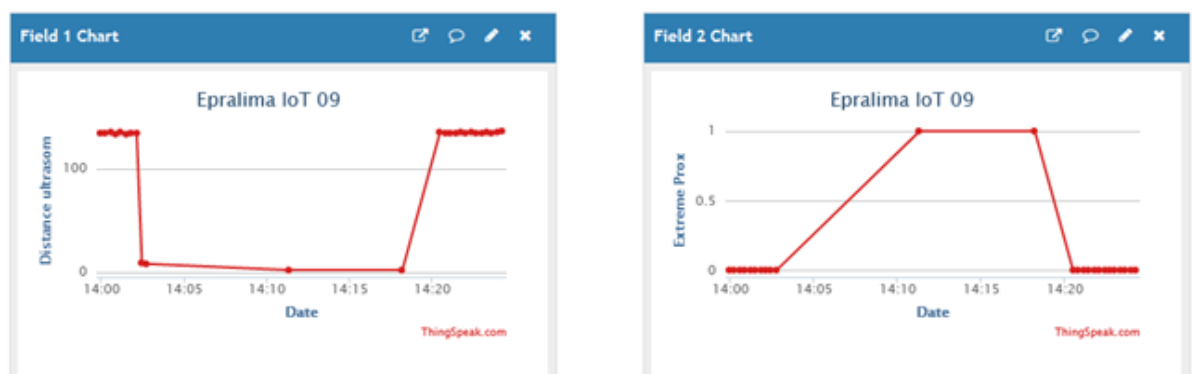


Image 12 – Results IoT ThingSpeak



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
22/04/2023 13:59	1	134	0
22/04/2023 14:00	2	134	0
22/04/2023 14:00	3	135	0
22/04/2023 14:00	4	133	0
22/04/2023 14:01	5	135	0
22/04/2023 14:01	6	133	0
22/04/2023 14:01	7	134	0
22/04/2023 14:02	8	134	0
22/04/2023 14:02	9	9	0
22/04/2023 14:02	10	8	0
22/04/2023 14:11	11	2	1
22/04/2023 14:18	12	2	1
22/04/2023 14:20	13	135	0
22/04/2023 14:20	14	134	0
22/04/2023 14:21	15	134	0
22/04/2023 14:21	16	134	0
22/04/2023 14:21	17	135	0
22/04/2023 14:22	18	134	0
22/04/2023 14:22	19	135	0
22/04/2023 14:22	20	134	0

Tabela 2 - DataSheet

In short

This exercise intends to apply a proximity sensor driven by a microcontroller in order to avoid collisions autonomously and protect passengers. This exercise is aided by a buzzer to emit sound signals, the closer the greater the sound alert, as well as a red LED that should light up in case of extreme proximity.

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.