



Assembly of circuit with speed variation (PWM) water sensor with 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 can be applied when weather conditions provide rainy weather and thus water can be detected on the vehicle's windows helping to clean autonomously, controlling the cleaning speed of the water.

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 sensors in a car and communicating with IoT.

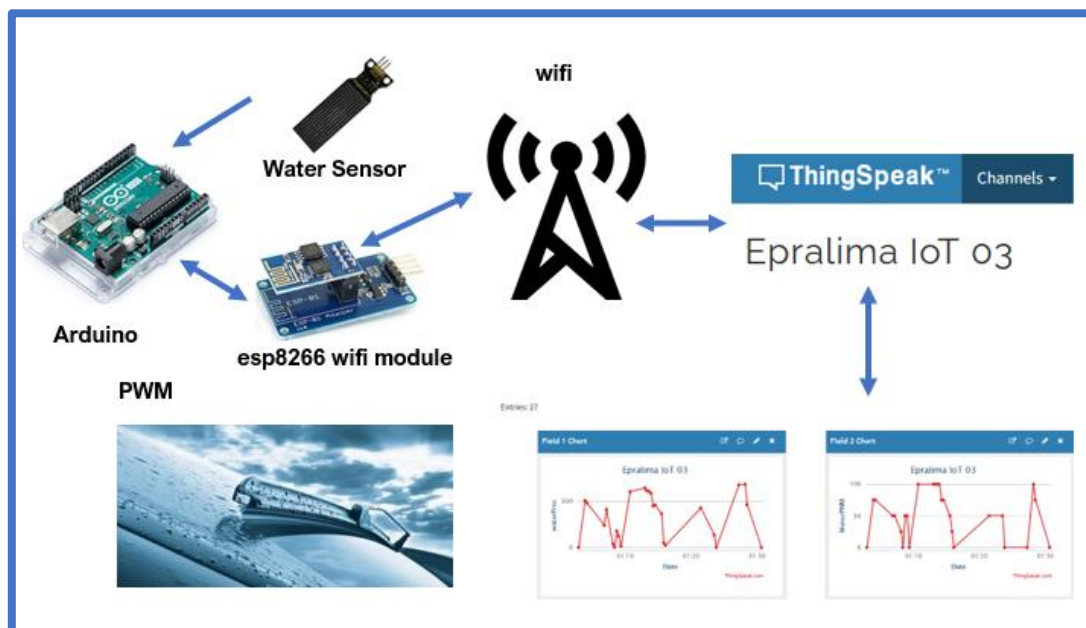


Image 1: application of water sensor in a car and communicating with IoT



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.


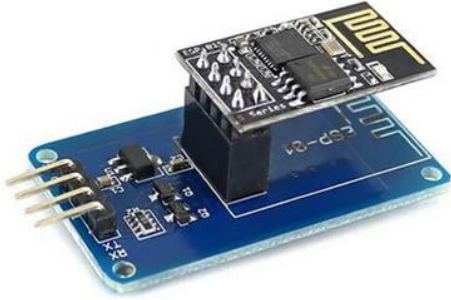
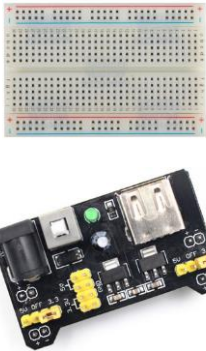




In this exercise we intend to learn how to draw diagrams (circuits), connect all the components correctly, develop software based on C language (Arduino), connect to the wifi network, communicate with an IoT server, ThingSpeak and read server-generated graphics.

Quantity	Component
1	Arduino Uno R3
1	ESP01-8266
1	Power Supply (braedBoard)
1	BreadBoard
1	Water sensor.
1	DC Motor
1	Module bridge L298N

Table 1 - Components List



Materials table

	
Arduino	ESP01 - 8266
	
Bread Board + Power Supply	Water Sensor
	
DC Motor	L298N Bridge Motor Module
	
Jumper wire	

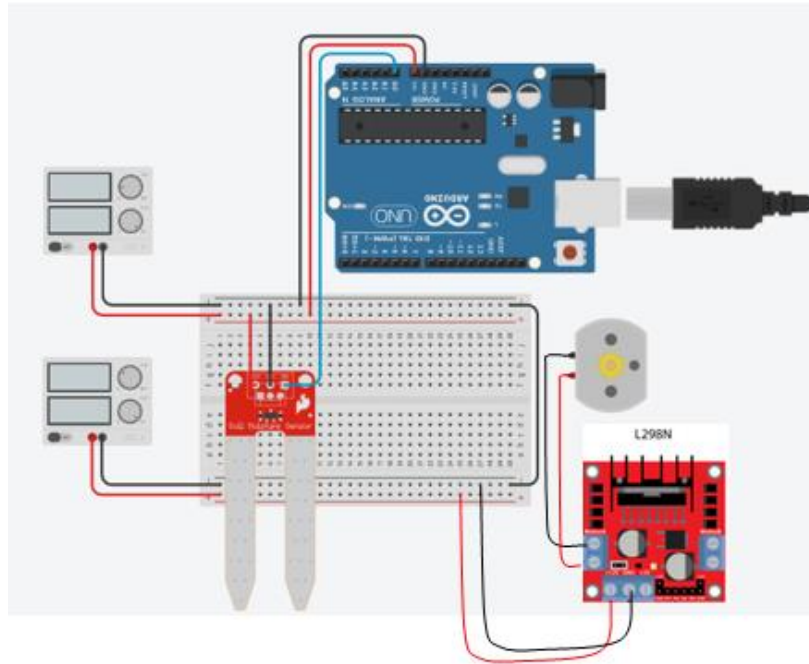


Image 2 – Diagram circuit

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.

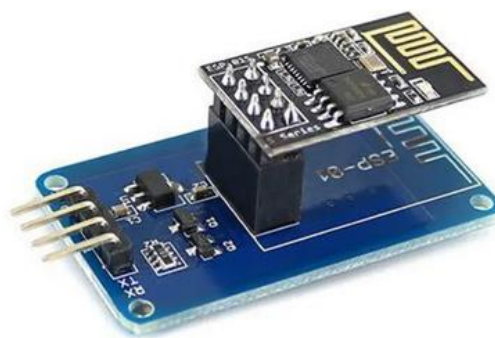


Image 3 - ESP01 – 8266



The module water sensor (image 4) is easy- to-use, portable and cost-effective, it is designed to identify and detect water level and water drop. This sensor measures the volume of water drop and water quantity through an array of traces of exposed parallel wires. Compared with its competitors, this sensor is not only smaller and smarter but also ingeniously equipped with following features:

Smooth conversion between water quantity and analogy quantity.

Strong flexibility, this sensor outputs basic analogy value.

Low power consumption and high sensitivity.

Directly connected to microprocessor or other logic circuits, suitable for a variety of development boards and controllers such as Arduino controller, STC single-chip microcomputer, AVR single-chip microcomputer.



Image 4 Water Sensor

Implementation in practice

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

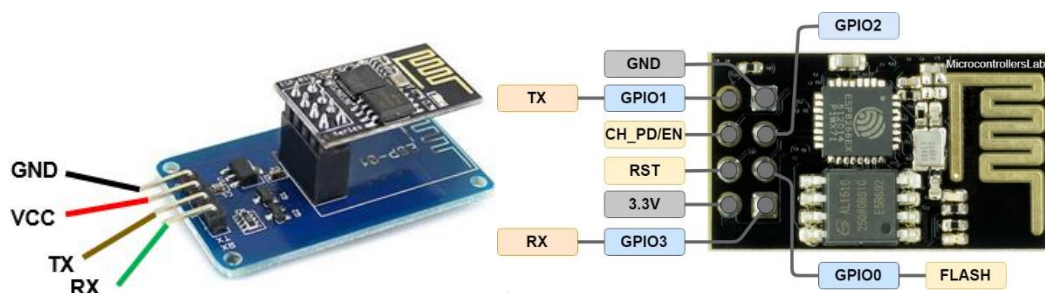


Image 5 ESP-01 Connections



3. Real assembled circuit image 6

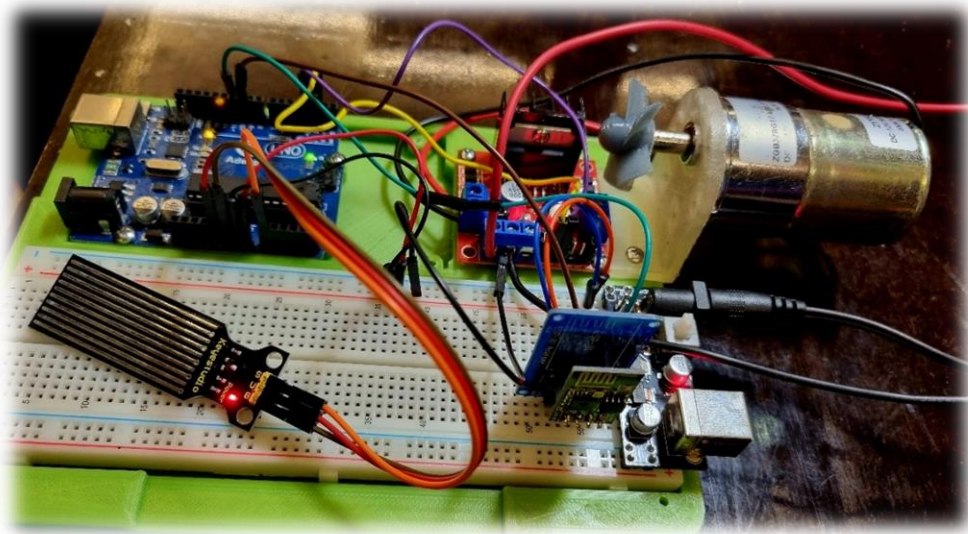
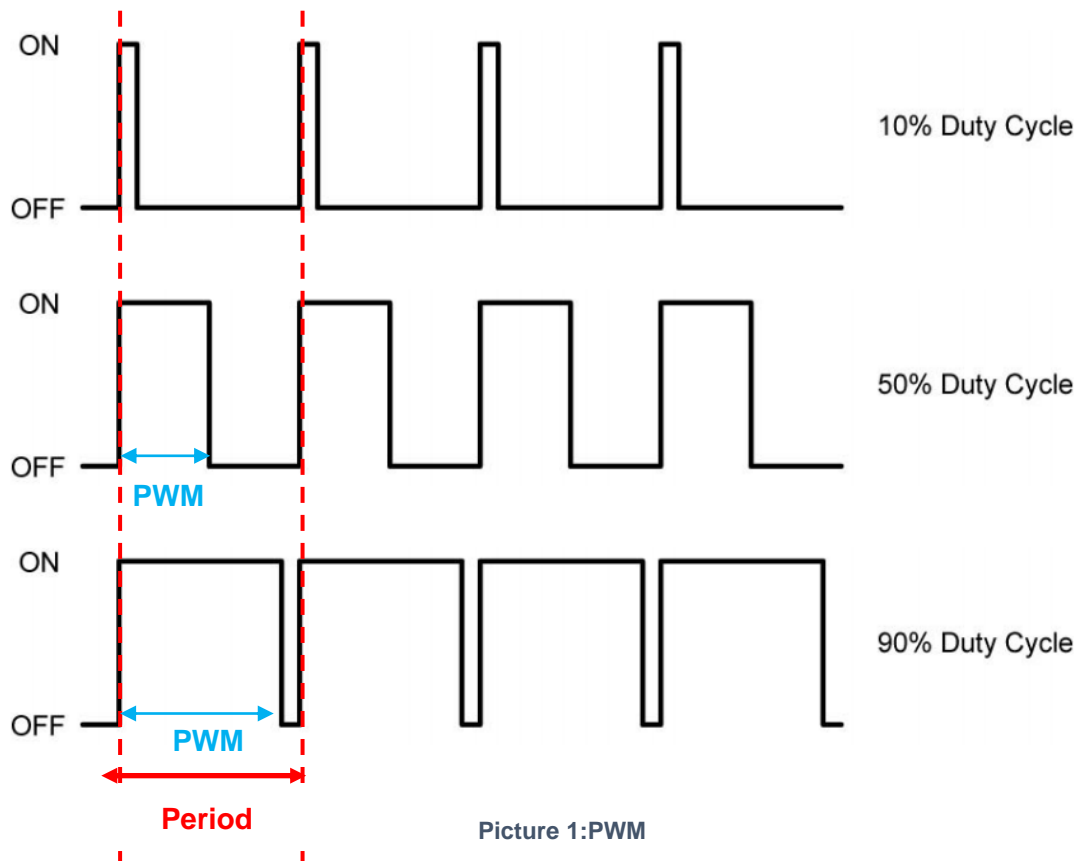


Image 6 Real circuit in breadboard

4. PWM (Pulse Width Modulation)

The PWM can be implemented in several areas of electronics. One of its uses is in power supply, DC motors speed control, light control, servo motors control and several other applications. Through PWM we can control speed and power.



Picture 1:PWM



PWM operation

Considering a square wave, to obtain the correct operation of the PWM we must vary the pulse width of the wave. To calculate we need the period and the pulse width and its result is called duty-cycle, as it is defined by the equation:

$$\text{Duty Cycle} = 100 \frac{\text{Pulse Width}}{\text{Period}}$$

Duty cycle: percentage value;

Pulse Width: sequence of time in which the signal is in high level;

Period: duration of a wave cycle.

5. Create a ThingSpeak account image 7

The screenshot shows the ThingSpeak website interface. At the top, there's a navigation bar with 'ThingSpeak™', 'Channels', 'Apps', 'Support', 'Commercial Use', 'How to Buy', and a user icon. Below the navigation bar, a green message box says 'Signed out successfully.' with a close button 'X'. The main content area contains text about signing in with a MathWorks account or creating a new one. It also mentions that non-commercial users can use ThingSpeak for free, while commercial users are eligible for a time-limited free evaluation. A link to 'paid license options' is provided. Below the text, there's a MathWorks logo and a form with an 'Email' input field. A red arrow points to the 'Create one!' link. Below the form, there's a 'Next' button. To the right of the form, there's a diagram illustrating the ThingSpeak architecture. The diagram shows 'SMART CONNECTED DEVICES' sending data to a cloud labeled 'DATA AGGREGATION AND ANALYTICS ThingSpeak™'. The cloud then sends data to a computer labeled 'MATLAB®' under the heading 'ALGORITHM DEVELOPMENT'.

Image 7 - Thing Speak



6. Create a new channel image 8

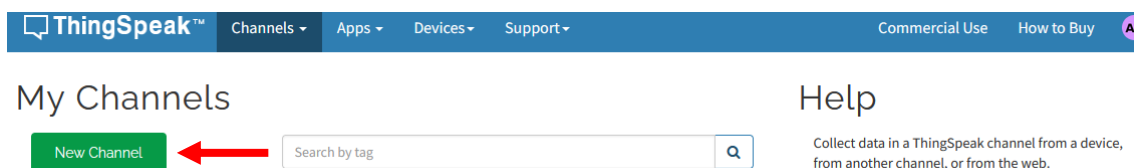


Image 8 Interface ThingSpeak

7. 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.

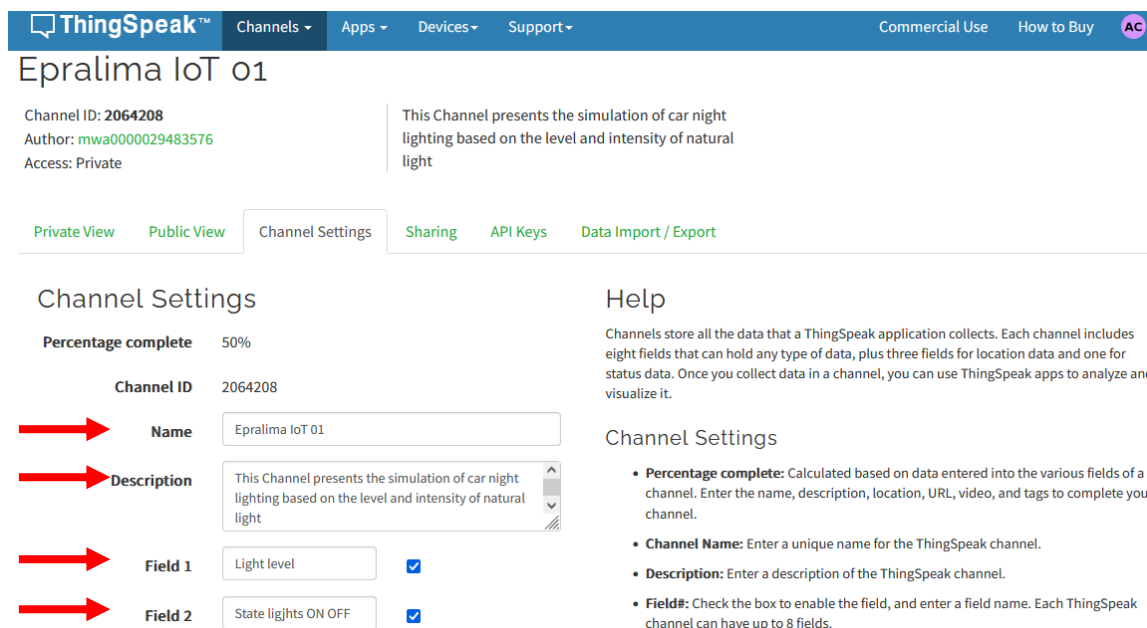


Image 9 Configure Channel



8. 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

9. 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[redacted]
```

Read a Channel Feed

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

Image 11 - API Keys



10. Programming Arduino

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

```
1 #include <SoftwareSerial.h>
2
3 SoftwareSerial ESP_Serial(10, 11); //PINOS QUE EMULAM A SERIAL, ONDE O PINO 10
4                                     // RX_AUX  --liga--> TX do ESP01
5                                     // TX_AUX  --liga--> RX do ESP01
6 #define serialcomSpeed 115200
7 #define DEBUG true
8 #define sentido1Motor 5
9 #define sentido2Motor 6
10 #define waterPres A0
11
12 int valueWater=0;
13 int valuePWM=0;
14
15
16 long writingTimer = 17;
17 long startTime = 0;
18 long waitTime = 0;
19
20 boolean error;
21
22 String APIKey = "          ";
```

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

```
31 void setup() {
32     Serial.begin(serialcomSpeed);
33     ESP_Serial.begin(serialcomSpeed);
34     startTime = millis();
35     InitWifiModuleESP();
36     pinMode(sentido1Motor, OUTPUT);
37     pinMode(sentido2Motor, OUTPUT);
38     pinMode(waterPres, INPUT);
39
40 }
```

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(valueWater);
95     getStr += "&field2=";
96     getStr += String(valuePWM);
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 **SensorWaterRead()** function receives data from the water sensor on an analogy port (A0) on the Arduino. Depending on the amount of water, it turns the engine on or off.

```
101 void SensorWaterRead(void) {  
102     valueWater=analogRead(waterPres);  
103     Serial.print("Water: ");  
104     Serial.println(valueWater);  
105     switch(dutyCicle(valueWater)) {  
106         case 0:  
107             stopMotor();  
108             valuePWM=0;  
109             break;  
110         case 1:  
111             runMotorFowardPWM(64);  
112             valuePWM=25;  
113             break;  
114         case 2:  
115             runMotorFowardPWM(128);  
116             valuePWM=50;  
117             break;  
118         case 3:  
119             runMotorFowardPWM(192);  
120             valuePWM=75;  
121             break;  
122         case 4:  
123             runMotorFowardPWM(255);  
124             valuePWM=100;  
125             break;  
126         default:  
127             stopMotor();  
128     }  
129 }  
  
131 int dutyCicle(int qtd_Water) {  
132  
133     if(qtd_Water<30) {  
134         return 0;  
135     }  
136     if(qtd_Water>=30 && qtd_Water<100) {  
137         return 1;  
138     }  
139     if(qtd_Water>=100 && qtd_Water<450) {  
140         return 2;  
141     }  
142     if(qtd_Water>=450 && qtd_Water<550) {  
143         return 3;  
144     }  
145     if(qtd_Water>=550 && qtd_Water<700) {  
146         return 4;  
147     }  
148 }  
149 }
```



```
201 void stopMotor(void) {
202     analogWrite(sentidolMotor, 0);
203     analogWrite(sentido2Motor, 0);
204 }
205
206 void runMotorFowardPWM(int velocity) {
207     analogWrite(sentidolMotor, velocity);
208     analogWrite(sentido2Motor, 0);
209 }
210
211 void runMotorBackwardPWM(int velocity) {
212     analogWrite(sentidolMotor, 0);
213     analogWrite(sentido2Motor, velocity);
214 }
```

Results

Considering that there is a certain amount of water on the front glass of the car, it is possible to automatically turn on the windshield wiper blades. It is possible to analyse the graph and observe a decrease in water and the turning on/off of the motor that activates the windshield wiper blades. With PWM (Pulse With Modulation) technology, it is possible to control the cleaning speed of the water depending on the amount of water.

Entries: 27

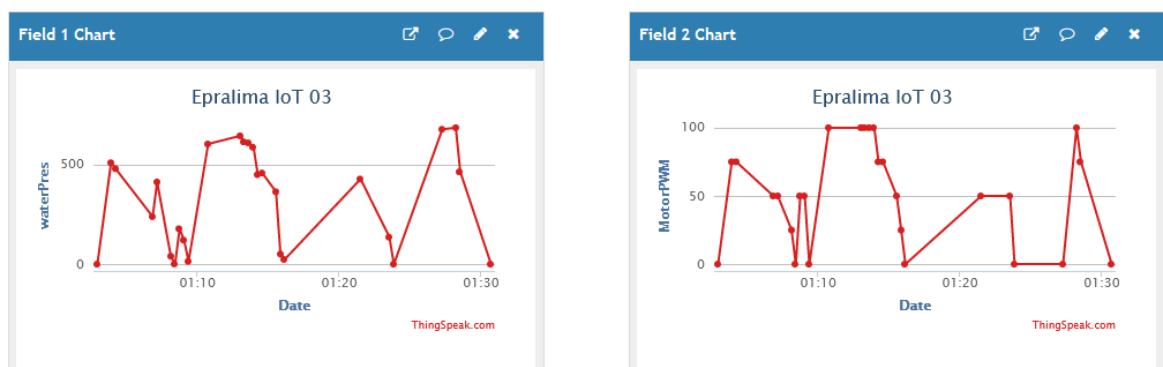


Image 12 – Results IoT ThingSpeak



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

	A	B	C	D	E
1	created_at	entry_id	field1	field2	latitude
2	17/04/2023 01:02	1	0	0	
3	17/04/2023 01:03	2	510	75	
4	17/04/2023 01:04	3	480	75	
5	17/04/2023 01:06	4	239	50	
6	17/04/2023 01:07	5	413	50	
7	17/04/2023 01:08	6	40	25	
8	17/04/2023 01:08	7	1	0	
9	17/04/2023 01:08	8	178	50	
10	17/04/2023 01:09	9	121	50	
11	17/04/2023 01:09	10	14	0	
12	17/04/2023 01:10	11	604	100	
13	17/04/2023 01:13	12	645	100	
14	17/04/2023 01:13	13	615	100	
15	17/04/2023 01:13	14	610	100	
16	17/04/2023 01:13	15	588	100	
17	17/04/2023 01:14	16	450	75	
18	17/04/2023 01:14	17	458	75	
19	17/04/2023 01:15	18	364	50	
20	17/04/2023 01:15	19	51	25	
21	17/04/2023 01:16	20	24	0	
22	17/04/2023 01:21	21	428	50	

Tabela 2 - DataSheet

In short

This can be applied when weather conditions provide rainy weather and thus water can be detected on the vehicle's windows helping to clean autonomously, controlling the cleaning speed of the water.