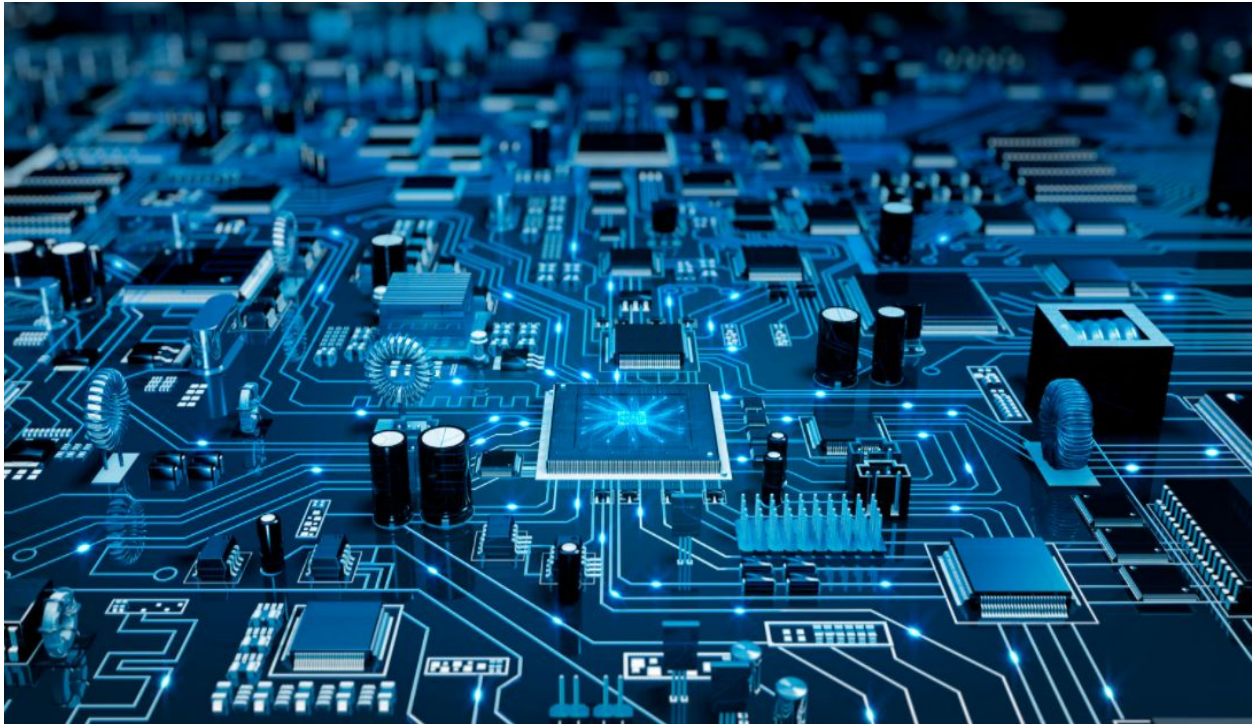


Rapport de Projet ESP32

André - Gala - Nicolas - Youssef



Sommaire:

- I. Référence des composants**
 - II. Contributions personnelles**
 - III. Accéder à la démo du projet**
 - IV. Explication du code**
-

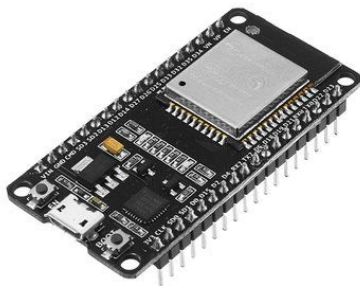
Introduction

Lors de ce second semestre à l'Efrei nous avons eu l'occasion d'explorer une nouvelle voie qu'est l'électronique. N'ayant, pour la plupart d'entre nous que peu ou pas d'expérience dans ce domaine là, les débuts ont bien entendu été difficiles et donc avons énormément appris. Une chose est principalement ressortie, c'est qu'on ne fait pas d'informatique sans électronique et qu'avant toutes choses, tout était basé sur des composants physiques

I. Référence des composants

Pour ce projet nous avons eu besoin que de trois composants majeurs :

- Une carte ESP32



-
- Une capteur de température et d'humidité DHT11



- Un capteur de distance via ultrason HC-SR04 :Ça prend en compte le temps de



parcours et la vitesse du son et peut calculer la distance.

Dispose de 4 broches, au *Gnd*, *VCC*, *Trig* et *Echo*. Les broches Ground et VCC du module doivent être connectées respectivement à la terre et aux broches de 5 volts de la carte Arduino et aux broches trig et echo de toutes les broches d' I/ O numériques de la carte Arduino

Bien entendu, nous avons aussi utiliser une breadboard pour interconnecter tous ces composants entre eux ainsi que des câbles.

II. Contributions personnelles

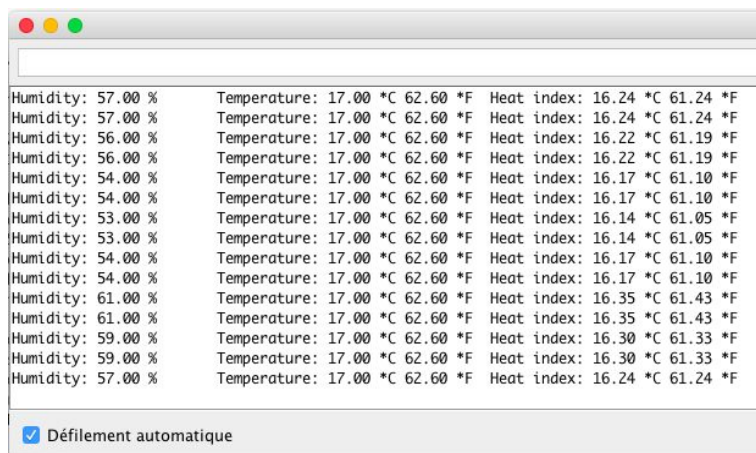
Nicolas Duchemann et Youssef Kriouile:

Nous concernant, nous nous sommes occupé de faire fonctionner le capteur de température ainsi que de l'API avec le site Ubidots

(<https://app.ubidots.com/ubi/insights/#/list>)

Voici comment nous nous sommes organisés:

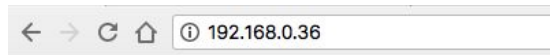
Tout d'abord, nous avons voulu faire marcher le capteur DHT11 seul pour essayer de comprendre son fonctionnement. Après quelques essais infructueux dues aux librairies qui n'étaient pas détecté, nous avons finalement réussi à obtenir un résultat:



Humidity: 57.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.24 °C 61.24 °F
Humidity: 57.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.24 °C 61.24 °F
Humidity: 56.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.22 °C 61.19 °F
Humidity: 56.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.22 °C 61.19 °F
Humidity: 54.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.17 °C 61.10 °F
Humidity: 54.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.17 °C 61.10 °F
Humidity: 53.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.14 °C 61.05 °F
Humidity: 53.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.14 °C 61.05 °F
Humidity: 54.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.17 °C 61.10 °F
Humidity: 54.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.17 °C 61.10 °F
Humidity: 61.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.35 °C 61.43 °F
Humidity: 61.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.35 °C 61.43 °F
Humidity: 59.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.30 °C 61.33 °F
Humidity: 59.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.30 °C 61.33 °F
Humidity: 57.00 %	Temperature: 17.00 °C 62.60 °F	Heat index: 16.24 °C 61.24 °F

☒ Défilement automatique

N'ayant pas opté tout de suite par la gestion des données par API, nous avons tout d'abord commencé par afficher ces valeurs dans notre navigateur web en écrivant de façon brute les données incorporées dans du code HTML. Cependant cela ne rendait vraiment pas bien :



ESP32 - DHT

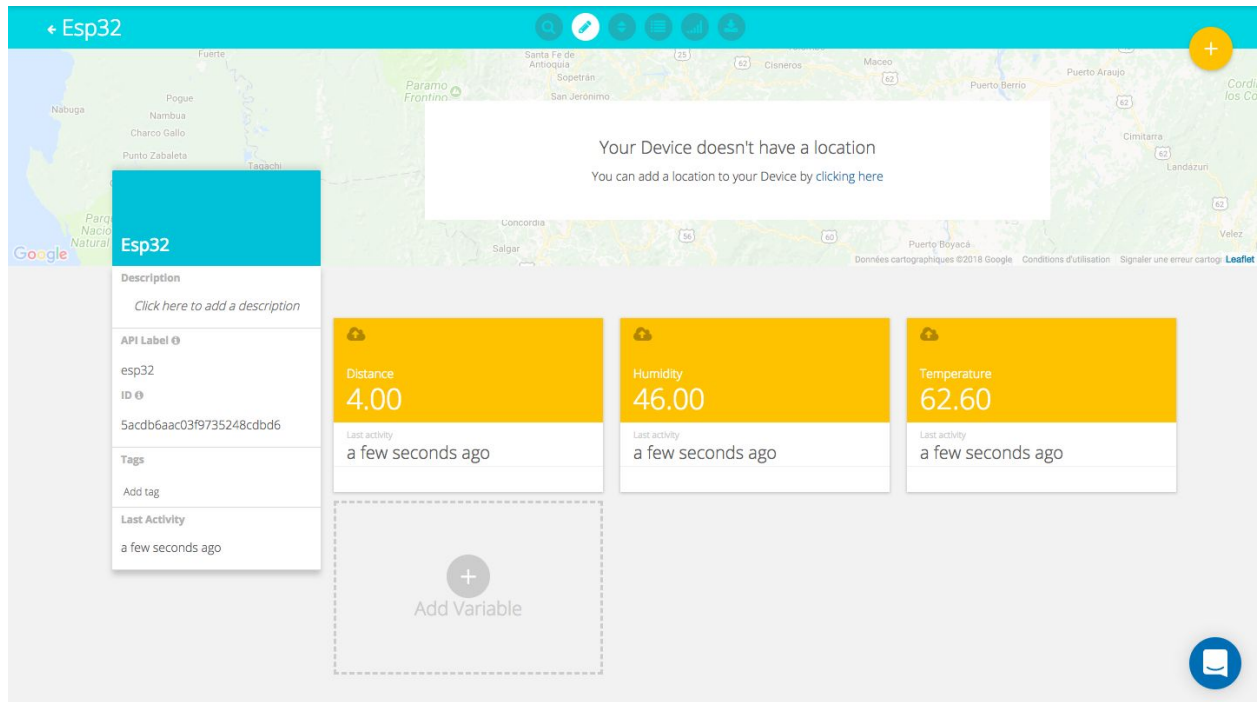
17.11 *C

62.80 *F

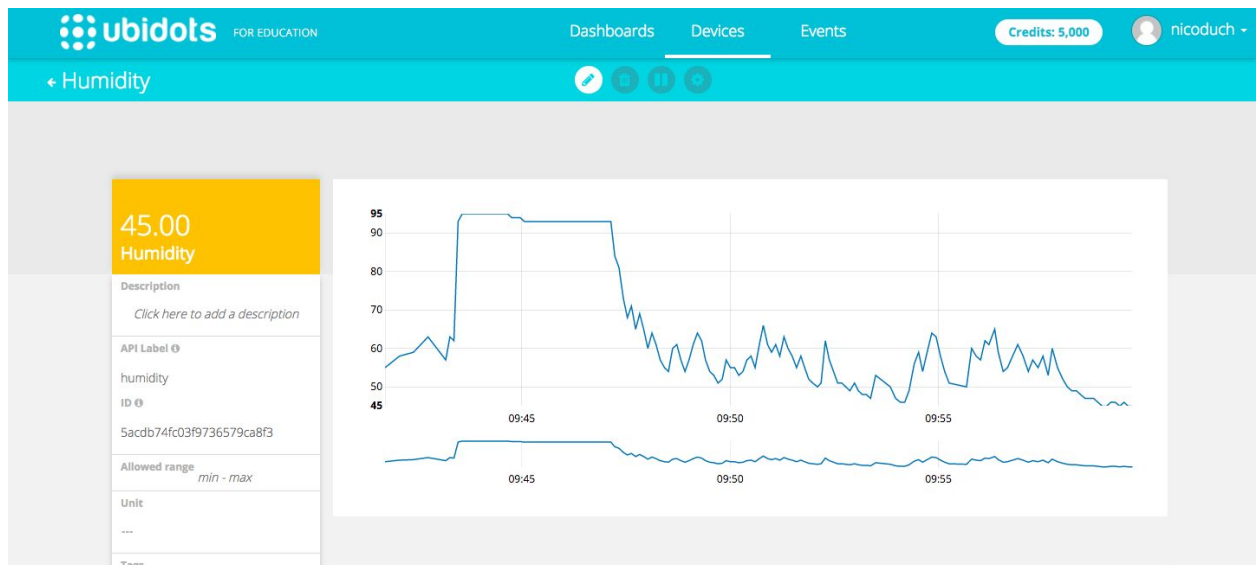
48.00 %

Après quelques recherches sur internet, nous avons finalement trouvé le site ubidots qui proposait exactement ce que nous recherchions.

Après quelques recherches, nous avons finalement compris comment le site fonctionnait. Après avoir créé mon nouvel appareil et initialiser ses variables sur le site nous avons pu générer des clés pour faire le lien entre notre programme et le site grâce à l'API.

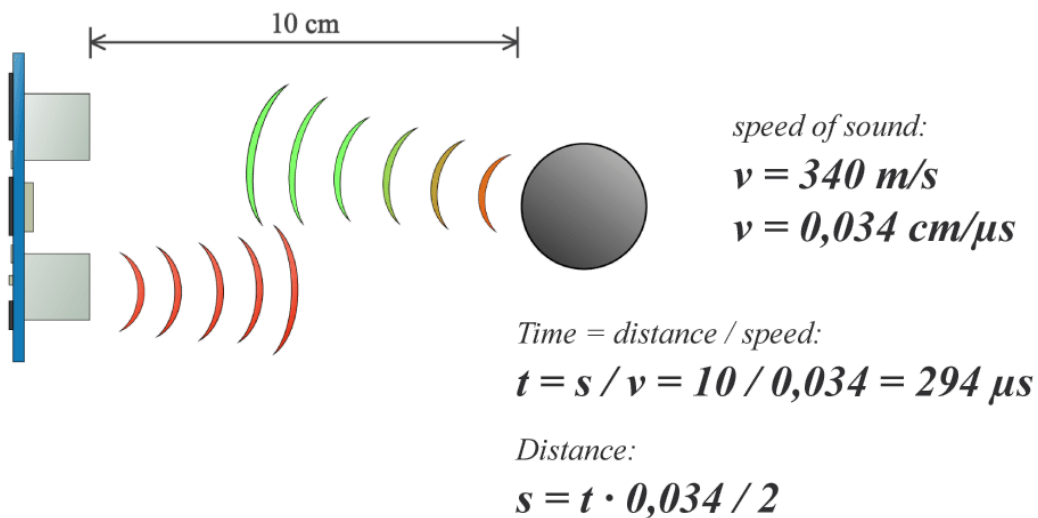


Le gros avantages de ce site c'est qu'il permet de traiter très facilement les données qu'il reçoit et à un rendu visuel très satisfaisant. Il permet en outre de générer des graphiques, d'envoyer des notifications sur nos smartphones si une certaine condition est atteinte (par exemple si la température dépasse un certain seuil) ou encore d'exporter les données reçues dans un fichier csv pour les traiter grâce au Big Data.

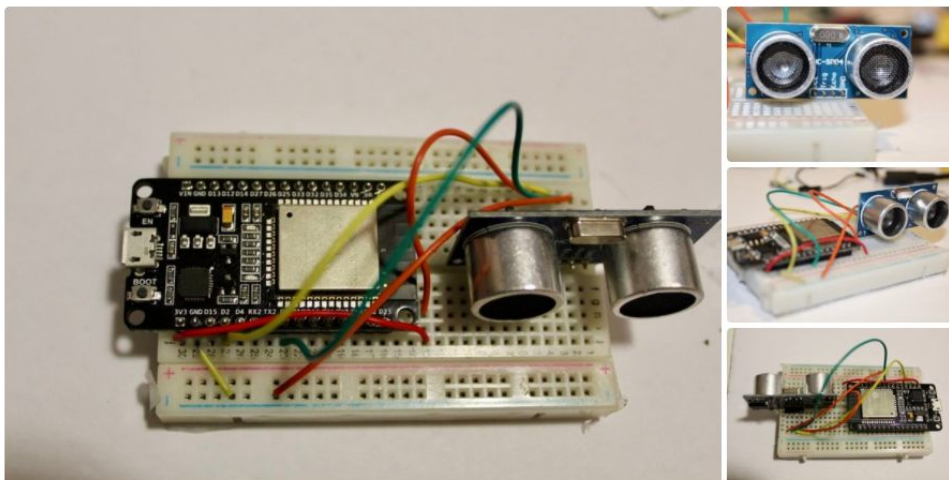


Gala Ramos & André Jiménez

Nous avons travaillé sur le capteur de distance avec l'appareil HC-SR04. Au début, nous avons besoin de comprendre comment l'appareil fonctionnait, et nous avons vu l'image suivante qui nous a fait comprendre qu'il y avait en fait deux parties pour ce capteur. Un capteur de sortie et un capteur d'entrée et comment ils interagissent entre eux.



En faisant de la recherche, nous avons pu connaître les matériaux nécessaires pour le faire fonctionner et comment connecter le HC-SR04 à l'ESP32 avec les fils grâce à la plaque de montage.



Voici comment cela fonctionne:

Le capteur à ultrasons HC-SR04 est utilisé ici pour mesurer la distance dans une plage de 2cm-400cm avec une précision de 3mm. Le module capteur se compose d'un émetteur ultrasonique, d'un récepteur et d'un circuit de commande. Le principe de fonctionnement du capteur à ultrasons est le suivant :

- Le signal de haut niveau est envoyé pour 10us à l'aide de Trigger.
- Le module envoie automatiquement huit signaux de 40 KHz, puis détecte si l'impulsion est reçue ou non.
- Si le signal est reçu, alors c'est par le haut niveau. Le temps de haute durée est l'intervalle de temps entre l'envoi et la réception du signal.

III. Accéder à la démo du projet

Pour accéder à la démo du projet, rendez-vous sur ce lien :

https://www.youtube.com/watch?v=azzvAo2_7t4&feature=youtu.be

IV. Explication du code

Concernant le code, nous allons utiliser uniquement deux librairies, les librairies Wifi et DHT. La librairie wifi est installée par défaut. En revanche, la librairie DHT se doit d'être installée manuellement. Dans l'IDE Arduino, il faut aller dans **Croquis -> Inclure une bibliothèque -> Gérer les bibliothèques** et installé **"DHT sensor library"**.

De plus, comme l'ESP32 n'est pas reconnue automatiquement par Arduino IDE, on doit installer manuellement Arduino Core (<https://github.com/espressif/arduino-esp32>)

Le code se décompose essentiellement en 4 parties :

- Déclaration de variables : C'est dans cette partie que l'on va affecter les clés générées par Ubidots à des variables qui permettront de faire le lien entre les deux parties.

```
// Setup variable ID's for Ubidots  
String temperature = "5acdb749c03f9736579ca8ef";  
String humidity = "5acdb74fc03f9736579ca8f3";  
String distancekey = "5acdbcfac03f97406a02a234";
```

- La fonction setup() : Permet de préparer la carte ESP32 et d'effectuer toutes les actions qui ne se réaliseront qu'une fois au lancement de la carte. Dans notre cas, scanner les réseaux wifi disponibles aux alentours à la recherche du réseau spécifier dans nos variables un peu plus haut:

```
//Setup variable for the wifi network  
const char* ssid = "AndroidAP Duchemann";  
const char* password = "canoute974";
```

Il devra ensuite s'y connecter:

```
WiFi.begin(ssid, password);  
  
// While the connection is not established, try again !  
while (WiFi.status() != WL_CONNECTED)  
{  
    delay(500);  
    Serial.print(".");  
}  
  
// Wifi Connected !  
Serial.println("");  
Serial.println("Wi-Fi connected");  
}
```

- La fonction Loop() : qui comme son nom l'indique sera la partie qui sera répétée indéfiniment. Cette partie est divisée en deux sous-parties : mesure des données et appel de la 4ème fonction.

```
void loop()
{
  //Setting the SR-04 Ultrasonic sensor
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, which will return the sound wave travel time in microseconds
  timeTravel = pulseIn(echoPin, HIGH);

  // Calculating distance
  distance= timeTravel*0.034/2;

  Serial.print("Distance: ");
  Serial.println(distance);
  // Read the current temperature and humidity measured by the sensor
  float temp = dht.readTemperature(true);
  float hum = dht.readHumidity();

  //Using the ubidots function in the bottom to convert data into JSON and send it to ubidots
  ubidots_saveValue(String(temperature), String(temp));
  ubidots_saveValue(String(humidity), String(hum));
  ubidots_saveValue(String(distancekey), String(distance));

  // Send some debug messages over USB
  Serial.println("Ubidots data");
  Serial.println("temperature: "+String(temp));
  Serial.println("humidity: "+String(hum));
  Serial.println("distance: ");
  Serial.println(distance);
  Serial.println(" Waiting 5 sec ..." );

  delay(5000);
}
```

De plus , nous avons ajouter des Serial.println() qui permet un bon débogage du programme.

- La fonction ubidots_saveValue(): Fonction principale de notre programme. C'est en effet elle qui permet de transmettre les données au site Ubidots.

```
String var = "{\"value\": " + value + "}";
String length = String(var.length());

// If we succeed to get the connection to the Ubidots API
if (client.connect("things.ubidots.com", 80))
{
    Serial.println("Connected to Ubidots...");
    delay(200);

    // Construct the POST request that we'd like to issue
    client.println("POST /api/v1.6/variables/"+variable_id+"/values HTTP/1.1");
    // We also use the Serial terminal to show how the POST request looks like
    Serial.println("POST /api/v1.6/variables/"+variable_id+"/values HTTP/1.1");
    // Specify the content type so it matches the format of the data (JSON)
    client.println("Content-Type: application/json");
    Serial.println("Content-Type: application/json");
    // Specify the content length
    client.println("Content-Length: " + length);
    Serial.println("Content-Length: " + length);
    // Using our own API token to publish the data
    client.println("X-Auth-Token: " + token);
    Serial.println("X-Auth-Token: " + token);
    // Specify the host
    client.println("Host: things.ubidots.com\n");
    Serial.println("Host: things.ubidots.com\n");
    // Send the actual data
    client.print(var);
    Serial.print(var+"\n");
}
else
{
    Serial.println("Ubidots connection failed...");
}
```

Nous avons alors juste à envoyer les données à l'URL correspondante à chaque variable grâce à la méthode POST, de spécifier que c'est un objet de type JSON et de fournir au site notre token qui permet de nous authentifier auprès de celui-ci et le tour est joué.