Ecole Nationale Supérieure

D’Arts et Métiers

ÉCOLE NATIONALE SUPÉRIEURE D’ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

Université Hassan II

Casablanca

Master : Ingénierie Digitale pour les Métiers de la Santé

Département : Génie Electrique

Détection de Chocs & Surveillance Environnementale

Présenté par :

HIMEDI Makrame

Encadré par :

Pr. ZAZ GHITA

Année Universitaire : 2024 – 2025

PLAN

INTRODUCTION

MQTT / HTTP

Partie 1

Détection de chocs à l'aide d'un accéléromètre

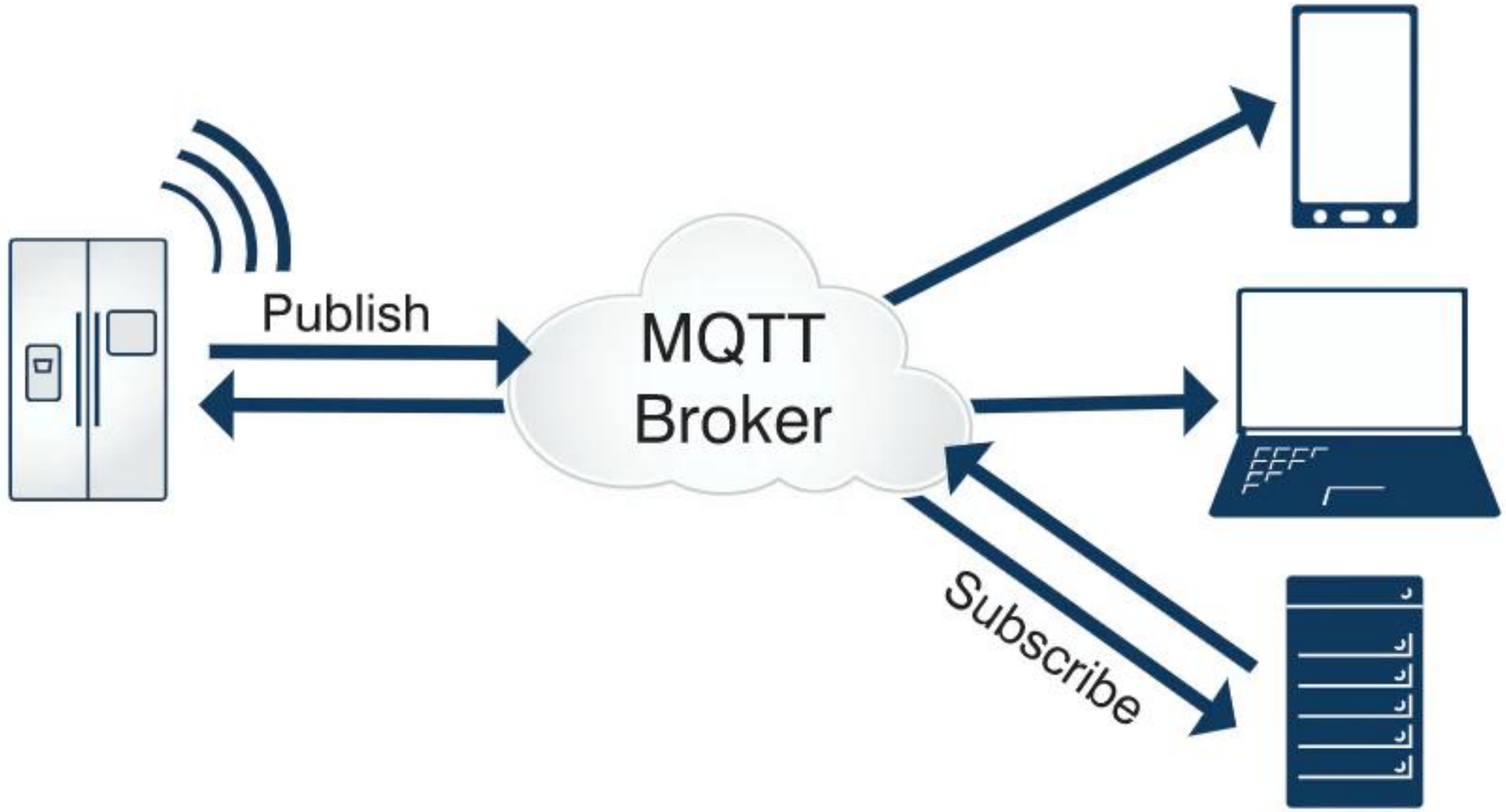
Partie 2

Serveur web basé sur un ESP32

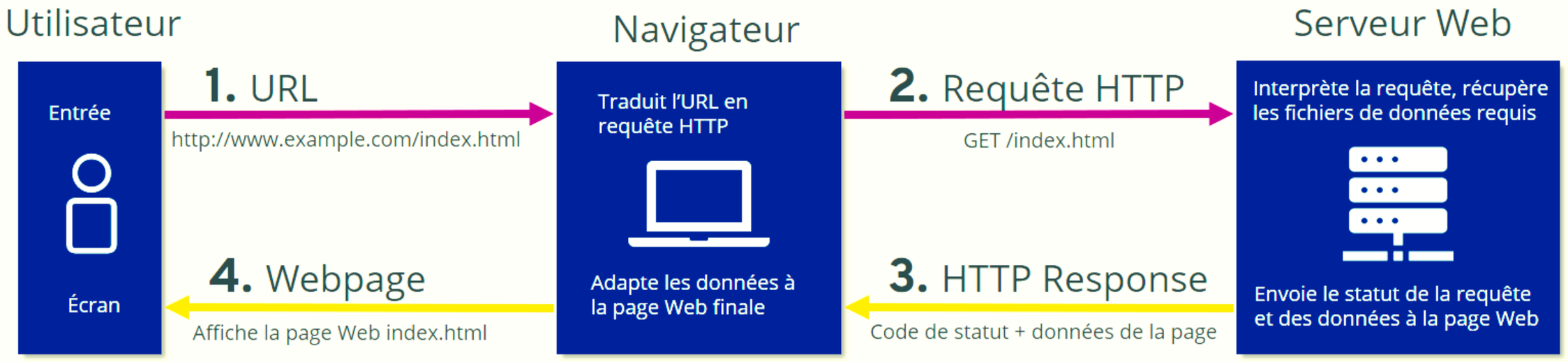
CONCLUSION

INTRODUCTION

MQTT (MESSAGE QUEUING TELEMETRY TRANSPORT)

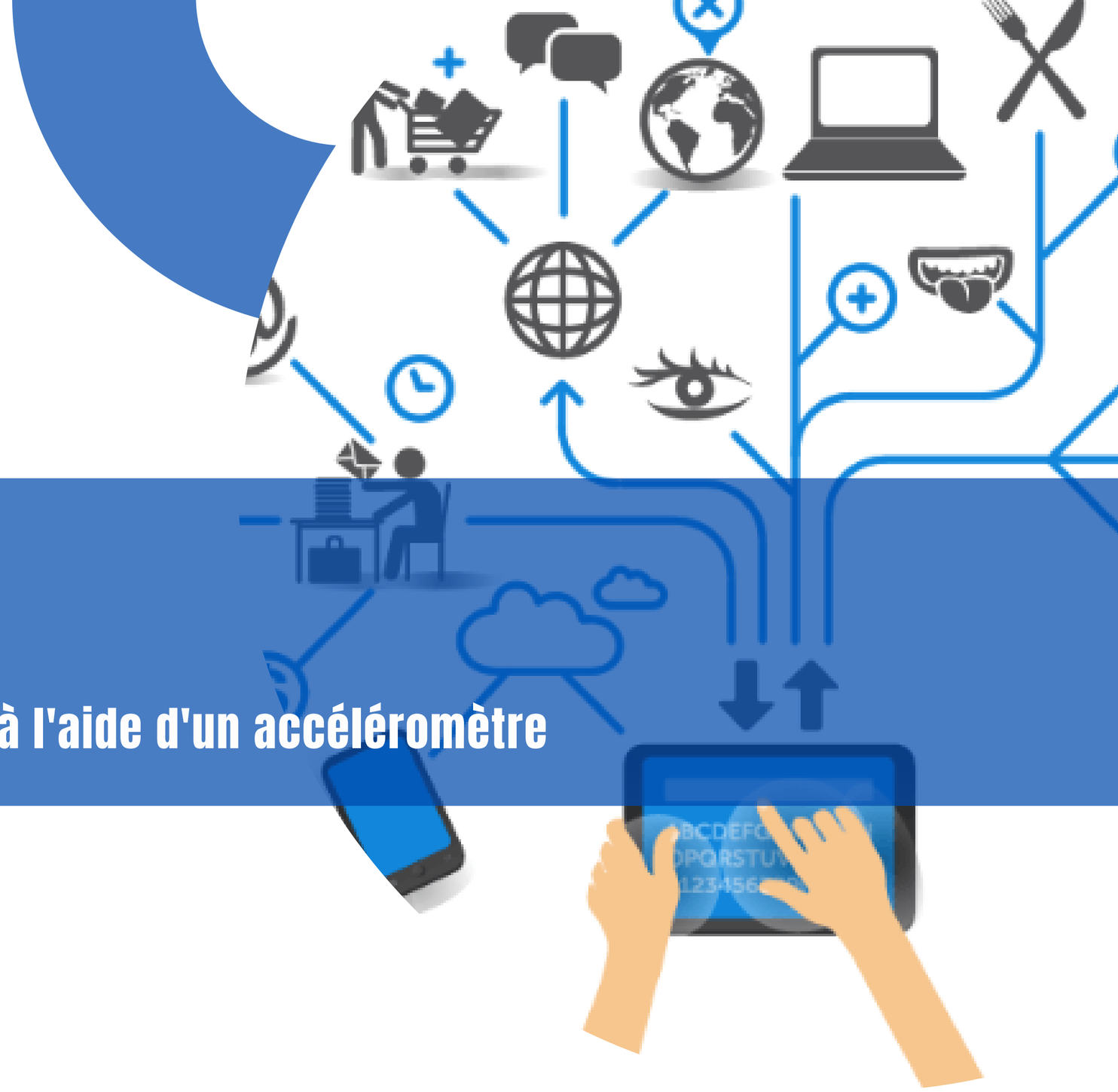


HTTP (HYPERTEXT TRANSFER PROTOCOL)



PARTIE 1 :

Détection de chocs à l'aide d'un accéléromètre



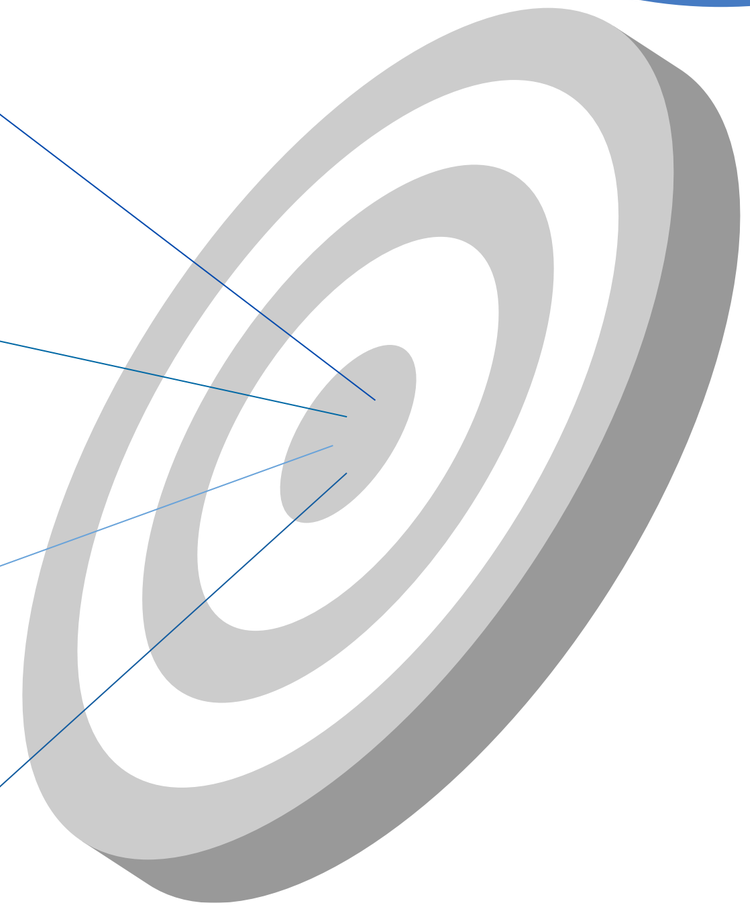
Les objectifs

Détection de chocs

Signalisation visuelle

Communication des données

Fiabilité et précision



Hardware

ESP32

Accéléromètre

LED

Wokwi

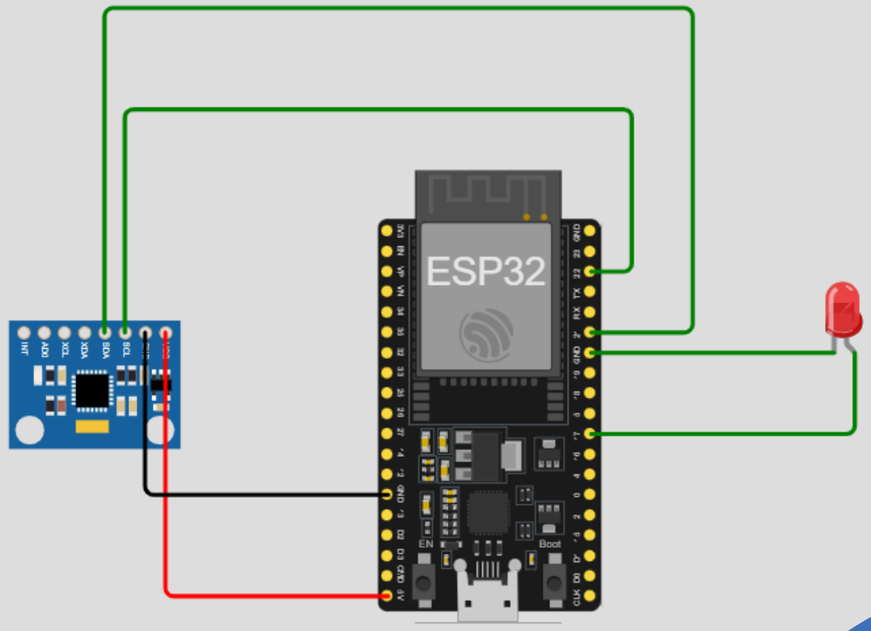
WOKWI SAVE SHARE project final by mak_07 Docs M

diagram.json | arduino_secrets.h ● | sketch.json | thingProperties.h

Untitled_jan02a.ino ● | libraries.txt | Library Manager

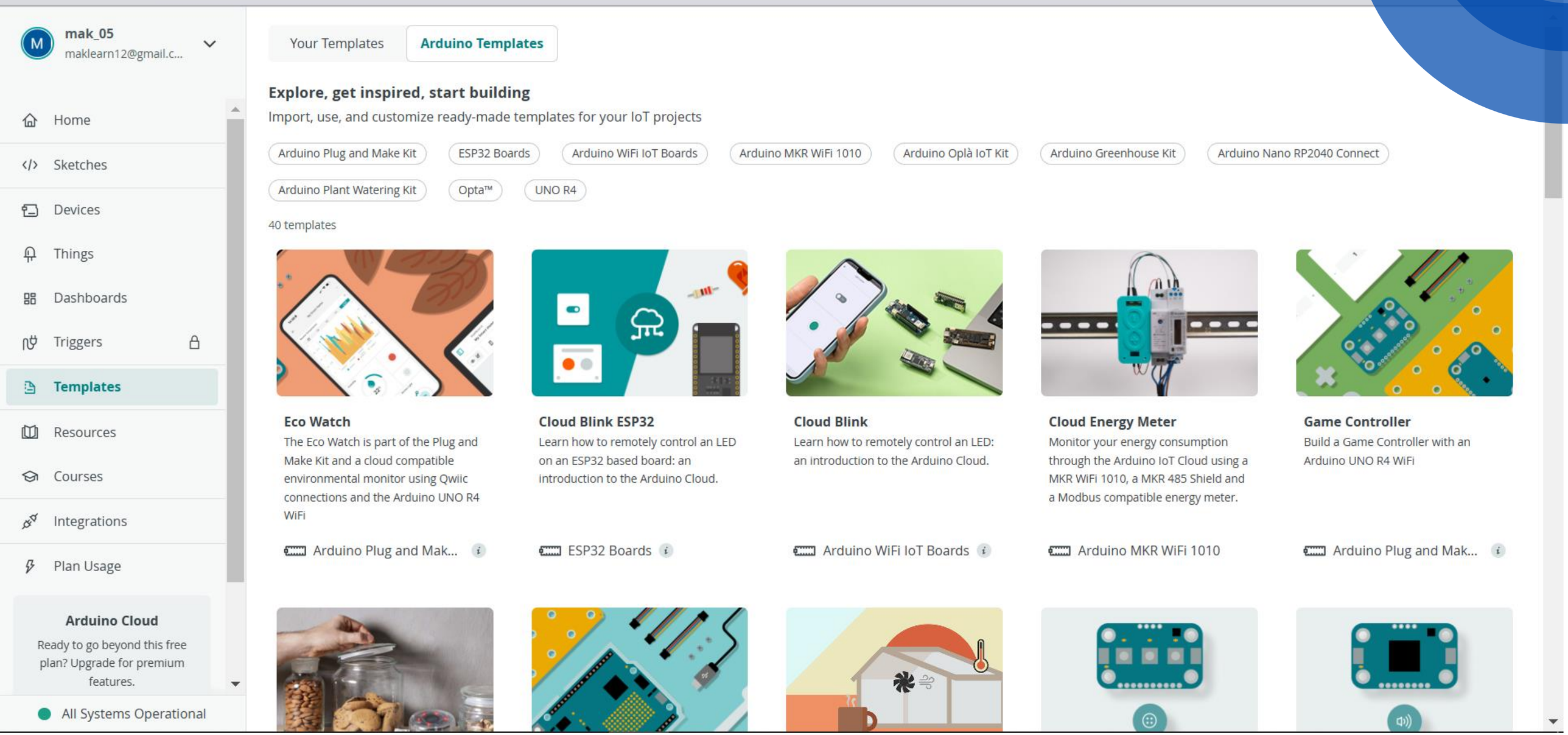
```
#include "arduino_secrets.h"
#include "thingProperties.h"
#include <Wire.h>

#define MPU6050_ADDR       0x68
#define ACCEL_XOUT_H       0x3B
#define TEMP_OUT_H         0x41    // Temperature data register
#define MPU_PWR_MGMT_1     0x6B
#define MPU_ACCEL_CONFIG   0x1C


// Shock detection configuration
#define SHOCK_THRESHOLD    2.0     // Threshold in g (adjust based on your ne
#define SHOCK_DURATION     500     // Minimum time between shock detections (
#define MOVING_AVG_SIZE    8       // Size of moving average window for noise

// Temperature configuration
#define TEMP_READ_INTERVAL 1000  // Temperature reading interval in ms

// Error handling
#define MAX_I2C_RETRIES    3

// Global variables
unsigned long lastShockTime = 0;
unsigned long lastTempReadTime = 0;

float accXHistory[MOVING_AVG_SIZE] = {0};
float accYHistory[MOVING_AVG_SIZE] = {0};
float accZHistory[MOVING_AVG_SIZE] = {0};
```

Simulation




```
AccX: 0.00 g, AccY: 1.05 g, AccZ: -1.00 g, Magnitude: 1.45 g, Temp: 85.00 °C
AccX: 0.00 g, AccY: 1.05 g, AccZ: -1.00 g, Magnitude: 1.45 g, Temp: 85.00 °C
AccX: 0.00 g, AccY: 1.05 g, AccZ: -1.00 g, Magnitude: 1.45 g, Temp: 85.00 °C
```

Arduino Cloud



Communication Arduino Cloud & Wokwi

PROFESSIONAL | EDUCATION | STORE

Search on Arduino.cc

CLOUD

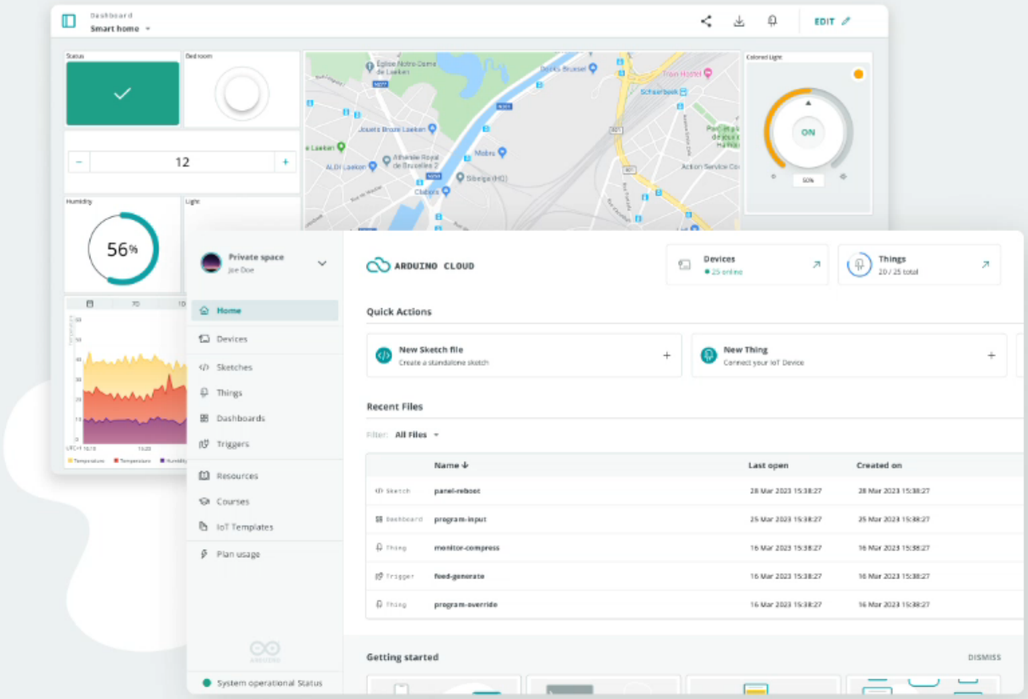
HOW IT WORKS | FEATURES | USE CASES | RESOURCES | SOLUTIONS | PLANS | GET STARTED

Try the Maker Monthly Plan for 1 month for FREE. Select monthly and use code: **HAPPYHOLIDAYS24**. Offer is valid for users who aren't on a paid plan. Cancel anytime.

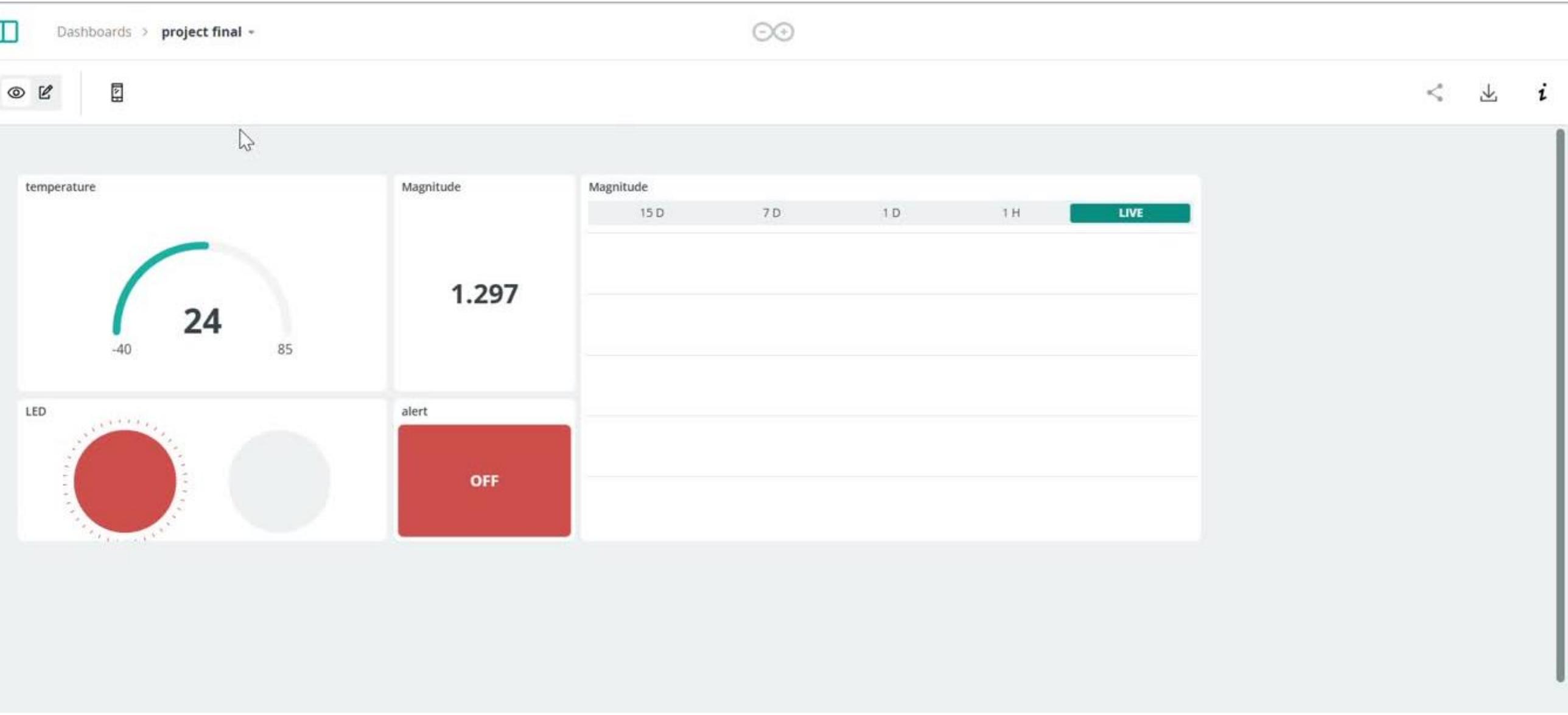
Bring your IoT projects to life quickly

Your next exciting journey to build, control and monitor your connected projects

GET STARTED FOR FREE | SEE PLANS



Dashboard Arduino Cloud



Partie Code

```
// Shock detection configuration
#define SHOCK_THRESHOLD   2.0    // Threshold in g (adjust based on your needs)
#define SHOCK_DURATION    500    // Minimum time between shock detections (ms)
#define MOVING_AVG_SIZE   8      // Size of moving average window for noise reduction
```

```
103  bool readAccelerometer(float& x, float& y, float& z) {
104    Wire.beginTransmission(MPU6050_ADDR);
105    Wire.write(ACCEL_XOUT_H);
106    if (Wire.endTransmission(false) != 0) {
107      Serial.println("Error: Failed to start accelerometer reading!");
108      return false;
109    }
110
111    if (Wire.requestFrom(MPU6050_ADDR, 6, true) != 6) {
112      Serial.println("Error: Failed to read complete accelerometer data!");
113      return false;
114    }
115
116    int16_t rawX = (Wire.read() << 8) | Wire.read();
117    int16_t rawY = (Wire.read() << 8) | Wire.read();
118    int16_t rawZ = (Wire.read() << 8) | Wire.read();
119
120    // Convert to g (±2g range)
121    x = rawX / 16384.0;
122    y = rawY / 16384.0;
123    z = rawZ / 16384.0;
124
125    return true;
126  }
```

Partie Code

```
156   // Initialize Arduino Cloud properties
157   initProperties();
158   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
159   setDebugMessageLevel(2);
160   ArduinoCloud.printDebugInfo();
161 }
```

```
 99  float calculateMagnitude(float x, float y, float z) {
100    return sqrt(x*x + y*y + z*z);
101  }
```

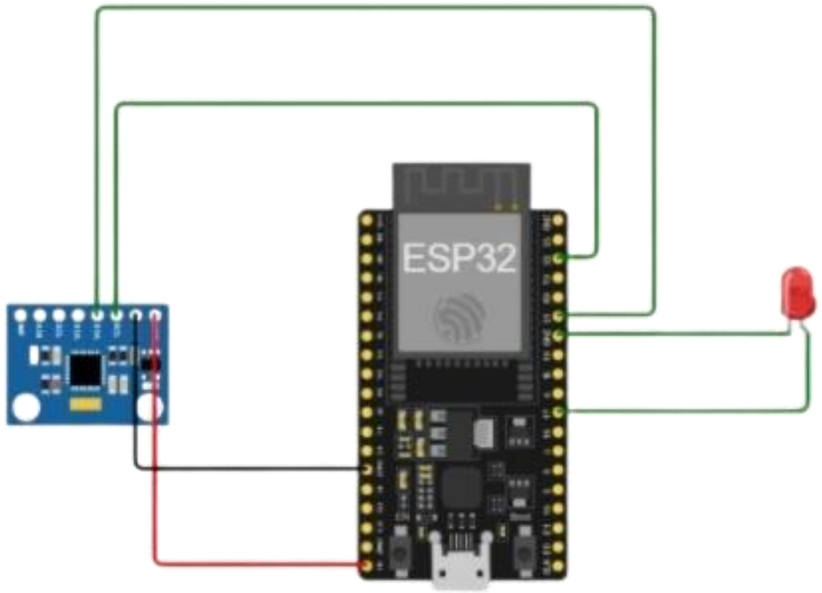
```
166 void loop() {
190   magnitude = calculateMagnitude(accX, accY, accZ);
191
192   // Check for shock and update shockDetected variable
193   shockDetected = detectShock(magnitude);
194
195
196   if (magnitude >= 2.0){
197    digitalWrite(ledPin, HIGH);
198    lED = true;
199   }else{
200    digitalWrite(ledPin, LOW);
201    lED = false;
202   }
203   // Print sensor data every 500ms
204   static unsigned long lastPrintTime = 0;
205   if (millis() - lastPrintTime > 500) {
206     Serial.printf("AccX: %.2f g, AccY: %.2f g, AccZ: %.2f g, Magnitude: %.2f g, Temp: %.2f °C\n",
207                   accX, accY, accZ, magnitude, temperature);
208     lastPrintTime = millis();
209   }
210
211   delay(10); // Short delay for stability
212 }
```

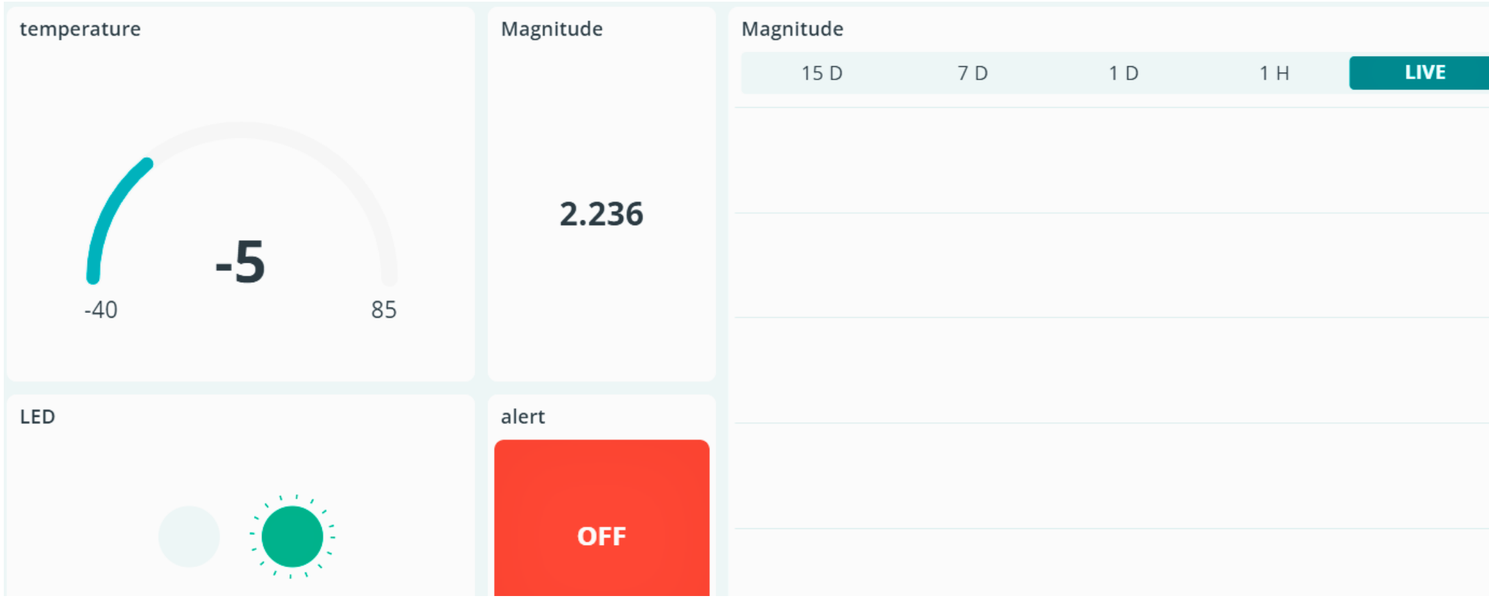
Partie Code

```
98
99    float calculateMagnitude(float x, float y, float z) {
100     return sqrt(x*x + y*y + z*z);
101   }
102
```

```
166   void loop() {
190     magnitude = calculateMagnitude(accX, accY, accZ);
191
192     // Check for shock and update shockDetected variable
193     shockDetected = detectShock(magnitude);
194
195
196     if (magnitude >= 2.0){
197      digitalWrite(ledPin, HIGH);
198      lED = true;
199     }else{
200      digitalWrite(ledPin, LOW);
201      lED = false;
202     }
203     // Print sensor data every 500ms
204     static unsigned long lastPrintTime = 0;
205     if (millis() - lastPrintTime > 500) {
206       Serial.printf("AccX: %.2f g, AccY: %.2f g, AccZ: %.2f g, Magnitude: %.2f g, Temp: %.2f °C\n",
207                     accX, accY, accZ, magnitude, temperature);
208       lastPrintTime = millis();
209     }
210
211     delay(10); // Short delay for stability
212   }
```

SIMULATION 1





WOKWI

ARDUINO IOT CLOUD

PARTIE 2 :

Serveur web basé sur un ESP32

OBJECTIFS

- Développer un système IoT intégrant le contrôle d'une LED ainsi que l'acquisition des données de température et d'humidité, accessibles via une interface web.
- Utiliser un ESP32 pour exécuter un web serveur accessible à partir d’un réseau Wi-Fi local.

Software
WEB SERVER

Un web server est un programme ou un dispositif qui gère les communications entre un client (comme un navigateur web) et un serveur.

Software

WEB SERVER

Rôle 1

Fournir une interface web accessible depuis un navigateur

Rôle 2

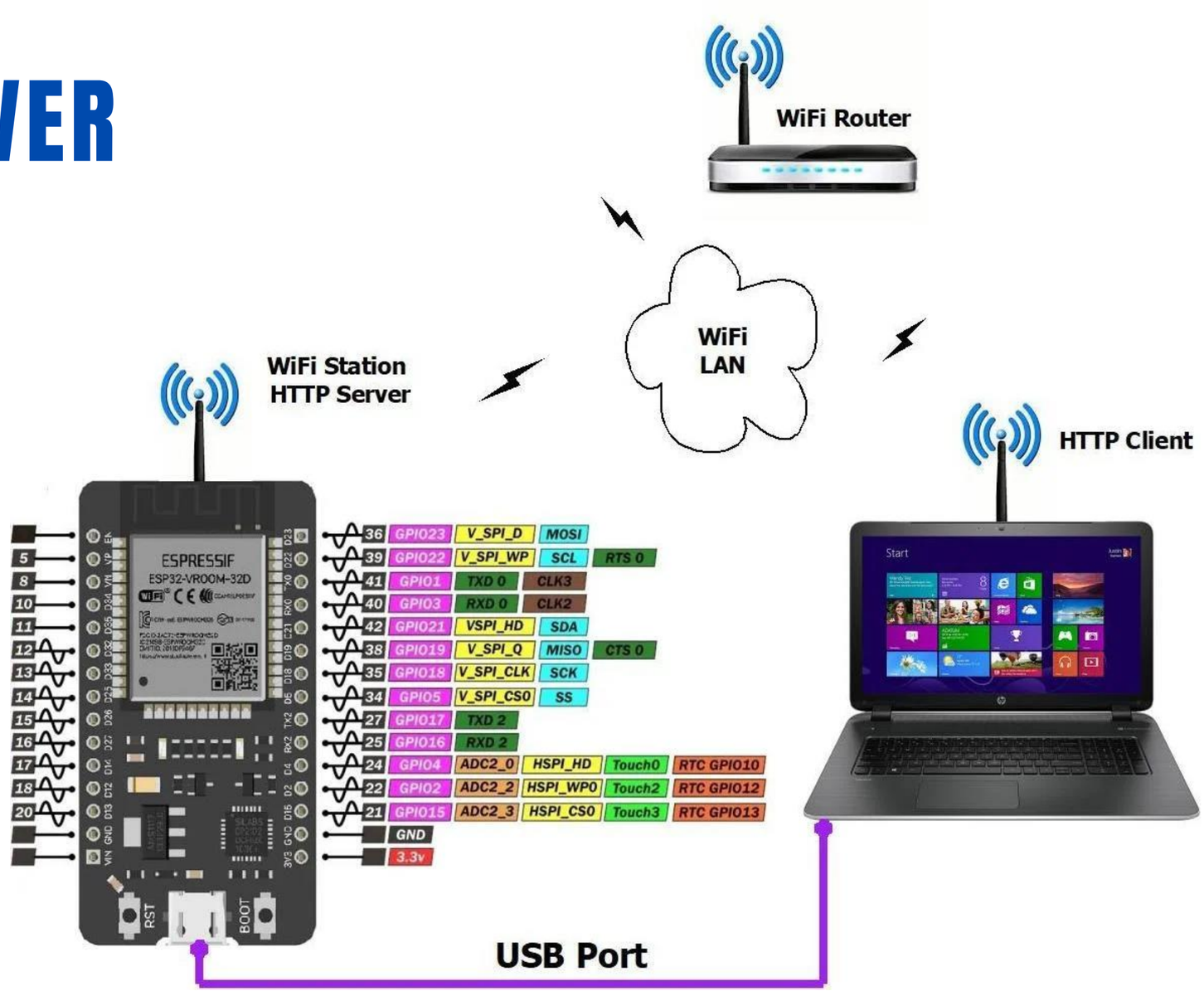
Recevoir les requêtes HTTP du navigateur utilisateur (allumer/éteindre la LED)

Rôle 3

Transmettre les données en temps réel (exemple : état du LED).

Software

WEB SERVER



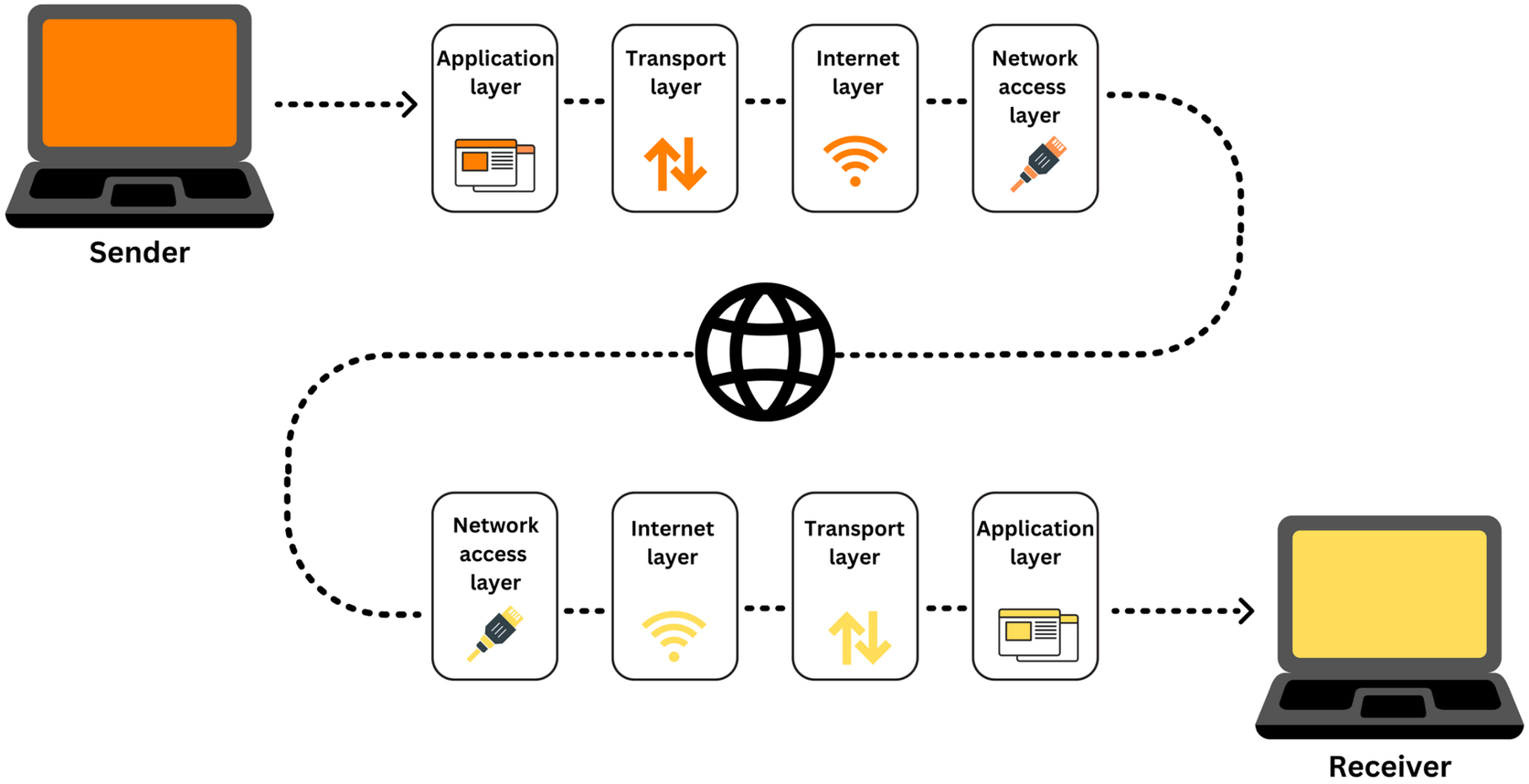
Software

PROTOCOLE TCP/IP

TCP/IP est un protocole standard de communication qui permet l'échange fiable de données sur un réseau, comme Internet, en assurant l'adressage (IP) et la transmission (TCP)

Software

PROTOCOLE TCP/IP



Software

PROTOCOLE TCP/IP

Rôle 1

Identification des appareils sur le réseau local

Rôle 2

Garantit que les données arrivent correctement.

Rôle 3

Permet d’envoyer et de recevoir des commandes pour contrôler la LED via le serveur web.

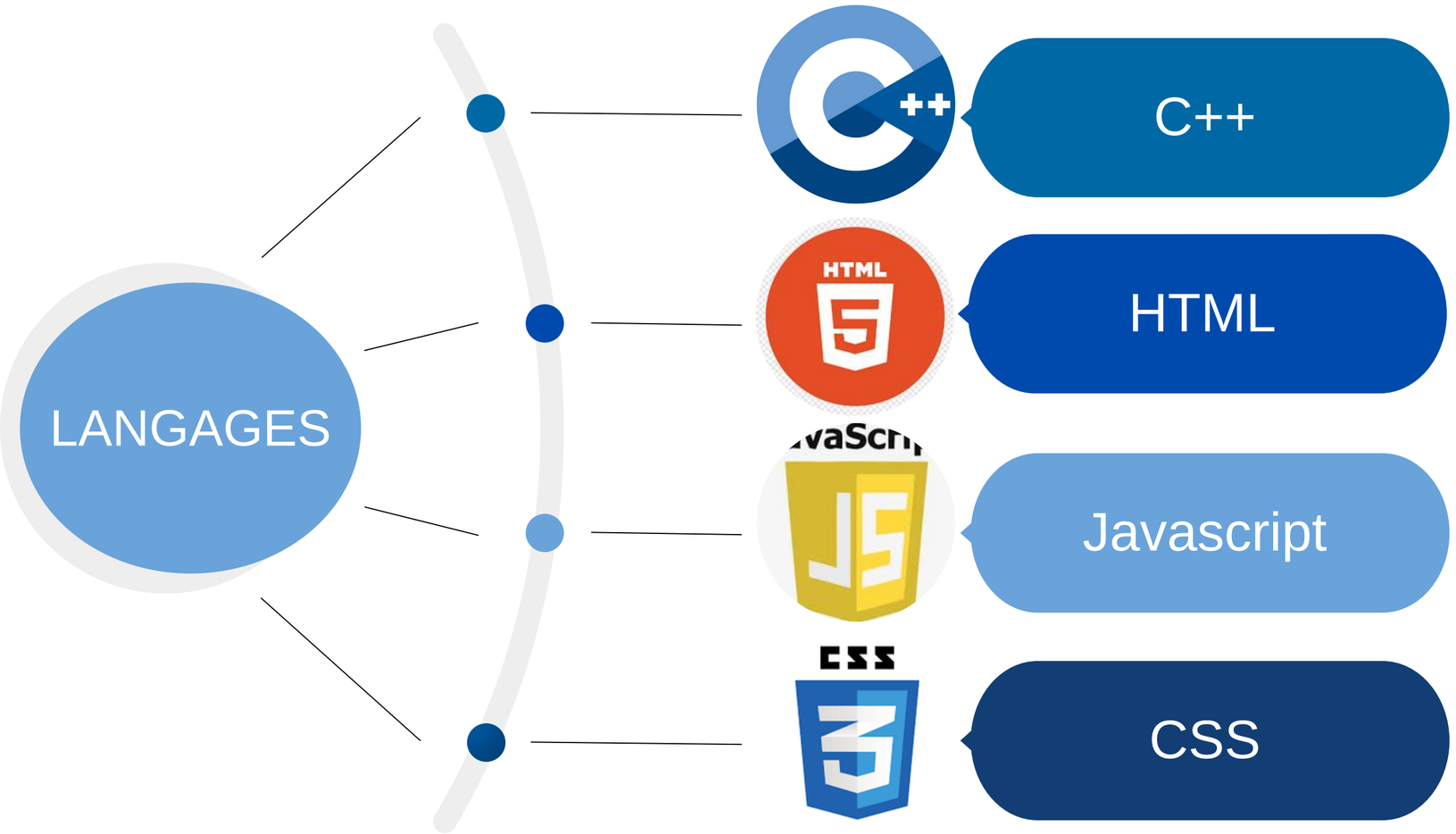
SOFTWARE

Environnement de développement:

Arduino IDE

Visual Studio Code

SOFTWARE



Utilisé pour programmer l’ESP32 (connexion Wi-Fi, gestion du web serveur, contrôle de la LED).

Structure de la page web pour interagir avec l’utilisateur.

Permettre des interactions dynamiques entre l'utilisateur et le serveur web ESP32

Pour styliser la page web et améliorer l’expérience utilisateur.

SOFTWARE

Bloc 1: Les bibliothéques

Webserver__1_.ino

```
1   #include <WiFi.h>
2   #include <WebServer.h>
3   #include <DHTesp.h>
```

SOFTWARE

Bloc 2:

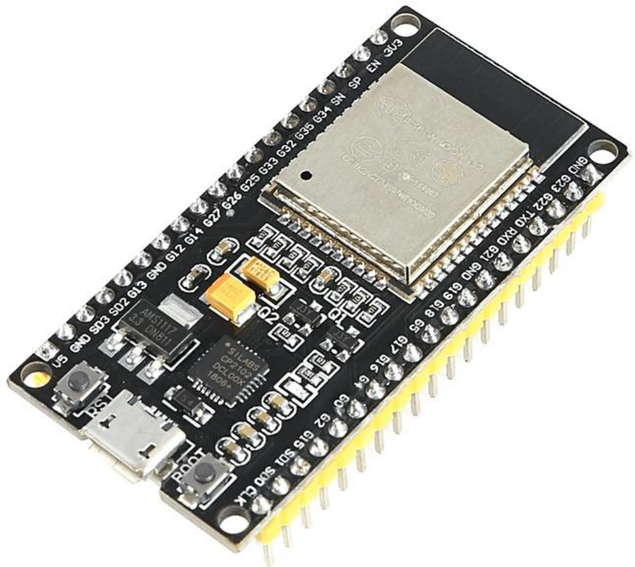
```
29  void setup() {
30    Serial.begin(115200);
31    dht.setup(DHT_PIN, DHTesp::DHT22);
32    pinMode(PIR_PIN, INPUT);
33    pinMode(LED_PIN, OUTPUT);
34    digitalWrite(LED_PIN, LOW);
35
36    WiFi.begin(ssid, password);
37    Serial.print("Connexion au WiFi");
38    while (WiFi.status() != WL_CONNECTED) {
39      delay(500);
40      Serial.print(".");
41    }
42    Serial.println("\nConnecté au WiFi");
43    Serial.print("Adresse IP: ");
44    Serial.println(WiFi.localIP());
45
46    server.on("/", handleRoot);
47    server.on("/data", handleData);
48    server.on("/toggle", handleLedToggle);
49    server.on("/stats", handleStats);
50    server.onNotFound(handleNotFound);
51
52    server.begin();
53    Serial.println("Serveur HTTP démarré");
54  }
55
```

SOFTWARE

Bloc 3:

```
void handleRoot() {
  String html = "<!DOCTYPE html><html><head>";
  html += "<meta charset='UTF-8'>";
  html += "<title>ESP32 Control Panel</title>";
  html += "<meta name='viewport' content='width=device-width, initial-scale=1'>";
  html += "<style>";
  html += "body { font-family: Arial, sans-serif; margin: 0; padding: 20px; background-color: #f0f0f0; }";
  html += ".container { max-width: 800px; margin: 0 auto; background-color: white; padding: 20px; border-radius: 10px; box-shadow: 0 0 10px rgba(0,0,0,0.1); }
  html += ".sensor-value { font-size: 24px; margin: 10px 0; padding: 15px; border-radius: 5px; background-color: #f8f9fa; }";
  html += ".button { background-color: #4CAF50; border: none; color: white; padding: 15px 32px; text-align: center; text-decoration: none; display: inline-blc
  html += ".button.off { background-color: #f44336; }";
  html += ".stats { margin-top: 20px; padding: 15px; background-color: #e9ecef; border-radius: 5px; }";
  html += "</style>";
  html += "<script>";
  html += "function updateData() {";
  html += "  fetch('/data').then(response => response.json()).then(data => {";
  html += "    document.getElementById('temperature').textContent = data.temperature.toFixed(1) + ' °C';";
  html += "    document.getElementById('humidity').textContent = data.humidity.toFixed(1) + ' %';";
  html += "    document.getElementById('pir').textContent = data.motion ? 'Mouvement détecté' : 'Pas de mouvement';";
  html += "    document.getElementById('led').textContent = data.led ? 'ON' : 'OFF';";
  html += "    document.getElementById('toggleBtn').textContent = data.led ? 'Éteindre LED' : 'Allumer LED';";
  html += "    document.getElementById('toggleBtn').className = 'button ' + (data.led ? 'off' : '');";
  html += "  });";
  html += "  fetch('/stats').then(response => response.json()).then(stats => {";
  html += "    document.getElementById('pirCount').textContent = stats.pirCount;";
  html += "    document.getElementById('ledOnTime').textContent = (stats.ledOnTime / 1000).toFixed(1);";
  html += "    document.getElementById('ledToggleCount').textContent = stats.ledToggleCount;";
  html += "  });";
  html += "}";
  html += "function toggleLed() {";
  html += "  fetch('/toggle').then(() => updateData());";
  html += "}";
  html += "setInterval(updateData, 2000);";
  html += "</script>";
  html += "</head><body onload='updateData()'>";
  html += "<div class='container'>";
  html += "<h1>Panneau de contrôle ESP32</h1>";
  html += "<div class='sensor-value'>Température: <span id='temperature'>--</span></div>";
```

SIMULATION 2



ESP32

WEB SERVER


CONCLUSION

MERCI POUR VOTRE ATTENTION