Année académique : 2023-2024

**Projet** :

Etude d’un système

d’acquisition,

d’analyse de signaux

et de

détection de défauts

**Participants**:

DEKO Kodzo Curtis

N’WOUI Gbati Patrice

**Chargé de l’UE** :

Pr ADJALLAH Hloindo Kondo

**Sommaire**

1. Introduction………………………………………………...………………………………………02
2. Conception et architecture………………….……………………………………………….02
3. Conception………………………………………………………………………………..0
4. Architecture du logiciel………………………………………………………………0
5. Acquisition et traitement de données…………………………...........................04
6. Acquisition…………………………………………………………………………………0
7. Analyse, traitement et détection des défauts…………………………….0
8. Résultats pratiques………………………………………………………………………………..05
9. Mise en œuvre du comportement du système en cas de détection de défauts……………………………………………………………………………………….0
10. Acquisition des données sur google sheets et lecture sur écran LCD……………………………………………………………………………………………...0
11. Perspectives……………………………………………………………………………………………06
12. Conclusion……………………………………………………………………………………………….06
13. Référence………………………………………………………………………………………………..07
14. Introduction

Ce projet s'inscrit dans le contexte de l'importance de la détection des défauts dans les systèmes électroniques et mécanique dont la température est l'un des facteurs les plus importants dans leur fonctionnement. Il consiste en la conception et la réalisation d'un système d'acquisition, d'analyse, de traitement de données et de détection de défauts utilisant l'ESP32, et un capteur de température et d'humidité (DHT11).

1. Conception et Architecture du système
2. Conception

Dans cette partie nous présentons les différents composants utilisés dans la réalisation de notre projet ainsi que leurs caractristiques et leur rôle dans le système.

1. **Esp32**
   1. **Caractéristiques :**

* Connectivité wifi et bluetooth
* Processeur double cœur cadencé jusqu'à 240 MHz
* Température de fonctionnement : -40°C à +85°C.
* Tension d'alimentation : 2,2V à 3,6V
  1. **Rôles :**

L’ESP32 est utilisé dans notre projet pour l'acquisition des données de température et d’humidité en provenance du DHT11, leur traitement et leur transmission à googlesheets pour une analyse et un traitement ultérieur plus sophistiqués.

1. **Capteur DHT11**
   1. Rôle :

* Le DHT11 a pour rôle de mesurer la température et l'humidité dans les environnements domestiques et industriels. Dans le cadre de ceux projet nous utiliserons la température du milieu ambiant tout en faisant varier cette dernière avec une source de chaleur extérieur afin de mettre en évidence le fonctionnement de notre système.
  1. **Caractéristiques :**
* Tension d'alimentation : 3.3V à 5.5V,
* Temps de réponse : 2 secondes,
* Plage de mesure  d'humidité : 20 à 90%  relative,
* Plage de mesure de température : 0 à 50°C.

1. **Fan et buzzer** :

Le rôle du **ventilateur** est de dissiper la chaleur et de dimunier la température dans notre système en cas de surchauffe ou d'élevation de la température, le système est conçu pour mettre en marche le ventilateur aumatiquement en cas de dépassement du seuil de température. Le **buzzer** joue un rôle d'alerteur lorsque la température dépasse un seuil. Un relais 5V permettra la commande du ventilateur.

Nous utilisons également un **écran LCD** pour afficher les données collectées par le DHT11 et constituant notre interface utilisateur nous permettent de connaitre l’état d’évolution de notre système, des **LEDs** pour signaler l’état de fonctionnement du système **(verte en cas de bon fonctionnement et rouge en cas d’un fonctionnement anormal).** Tout l’ensemble du système sera câblé sur une **breadboard**.

1. Architecture du logiciel

Pour ce projet nous avons utiliser l'environnement de développement **Arduino** pour l'implémentation de notre programme. **L’IDE Arduino** simplifie le développement sur **l’ESP32** grâce à son environnement convivial et ses nombreuses bibliothèques compatibles. Il permet de compiler et de télécharger facilement le code, d’utiliser un moniteur série, et de bénéficier d’une large documentation et d’exemples. L’installation et la gestion des cartes ESP32 dans l’IDE sont également très faciles grâce au gestionnaire de cartes intégré.

1. Acquistion et traitement de données
2. Acquisition
3. **Les données**

En général, les systèmes d’acquisition de données sont utilisés pour collecter une grande variété de types de données en fonction d’applications.

* Données physiques : température, pression, déplacement et vibration, force et poids
* Données chimiques : concentration de gaz, pH et conductivité, …
* Données électriques : courant, tension, résistance ;
* Donnés biologiques : signaux biomédicaux, paramètres physiologiques ;
* Données environnementales : qualité de l’air, humidité,
* Données mécaniques : vitesse, accélération, position, orientation,…

Dans le cas de ce projet seul les données acquises sur la température et l’humidité seront utilisé.

1. **Composants principaux**

* **Capteur**: Le DHT11 fournis deux types de données, les données de température et d'humidité. Dans le cadre de notre projet nous allons utilisé uniquement les données de température pour la suite du travail (traitement et détection de défauts)
* **Interfaces de communication** : **l’écran LCD** affichera les données collectées par le **DHT11**. **Google sheets** pour recevoir en ligne les données du DHT11 transmis par l'ESP32 grâce à un réseau wifi.

1. Analyse, traitement et détection de défauts

L'ESP32 analyse les données de température reçues pour détecter les conditions anormales du système :

* Si la température est en dessous de 30°C, le système est en fonctionnement normal et la Led verte est allumée
* Si la température est au-dessus de 30°C, le système est en fonctionnement anormal et la Led rouge s’allume.

Les résultats sont envoyés vers **google sheets** via **wifi** pour un suivi et une analyse ultérieure, perttant une intervention rapide et efficace en cas de défauts.

1. Résultats pratiques

Une image contenant Appareils électroniques, Ingénierie électronique, fils électriques, Composant de circuit

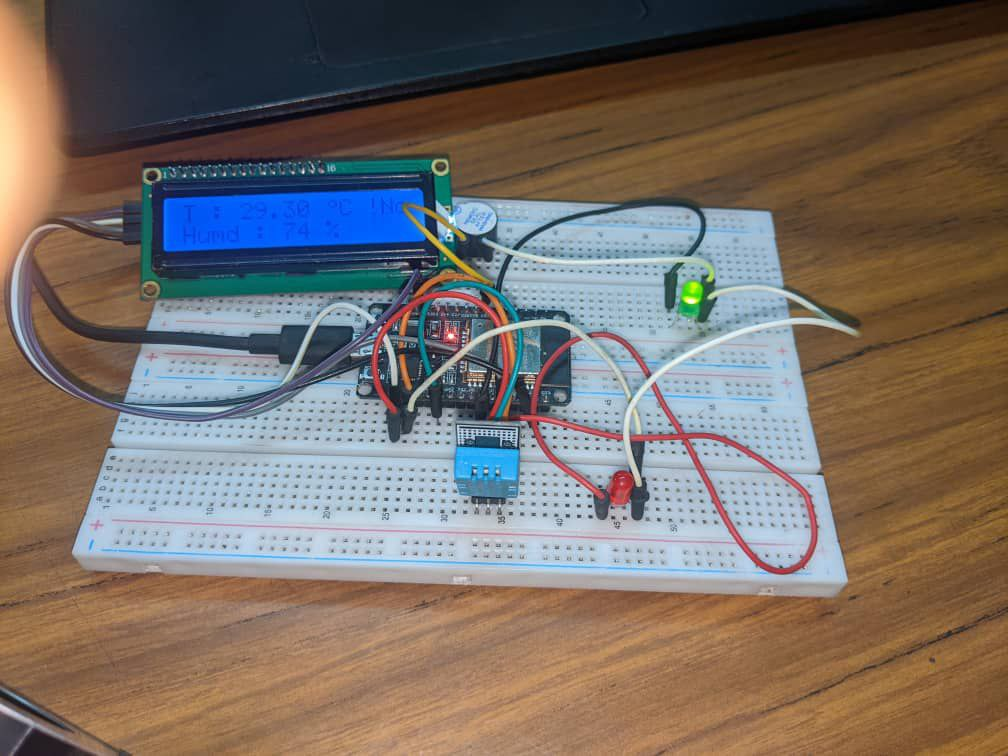
Description générée automatiquement

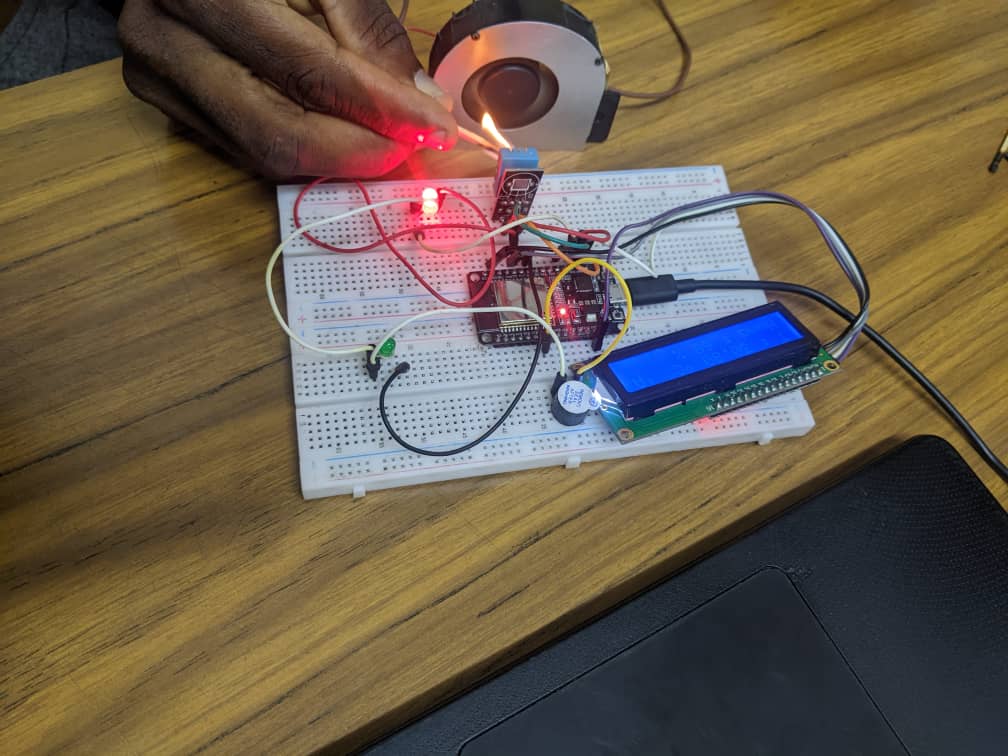
Figure 1: Montage du système

1. Mise en œuvre du comportement du système en cas de détection de défauts

Dans le programme, nous avons défini le comportement comme suit :

* A une température en dessous de 30°C, l’échantillon est à son état normal et une LED verte s’allume



* A une température au-dessus de 30°C, un défaut est détecté : le buzzer se déclenche, la LED rouge s’allume et le ventilateur se met en marche.

1. Acquisition des données sur google sheets et lecture sur écran LCD

Les données enregistrées sur google sheet se présentent comme ci-dessous :



L’écran LCD affiche la température à chaque étape d’enregistrement sur le google sheet

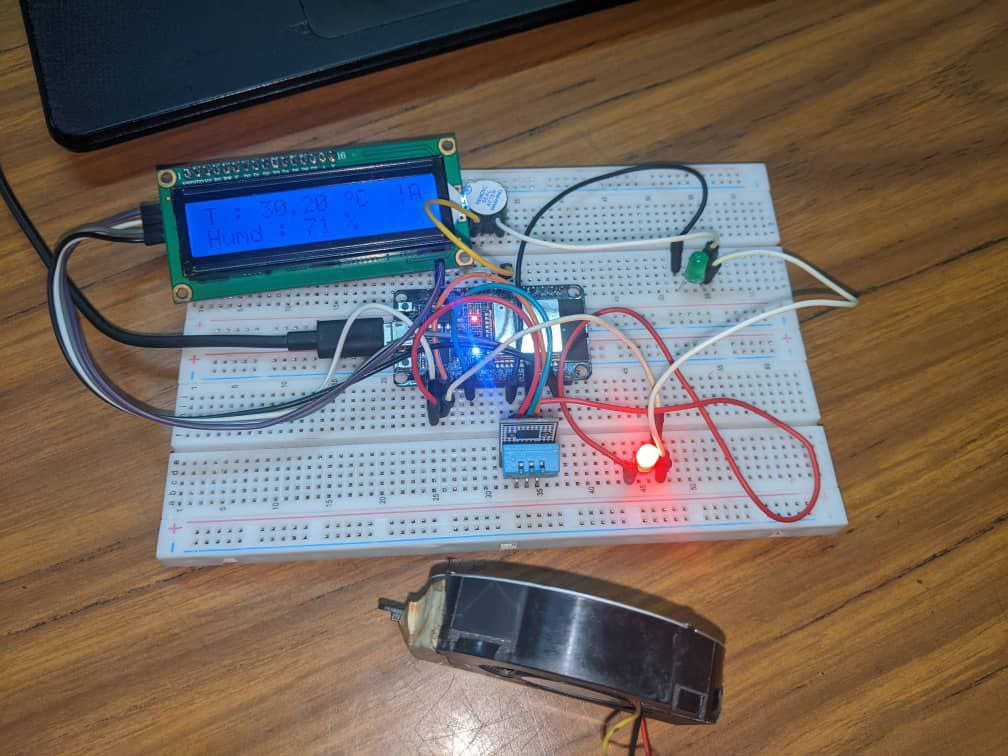


Figure 3 Affichage de la température sur l'écran LCD

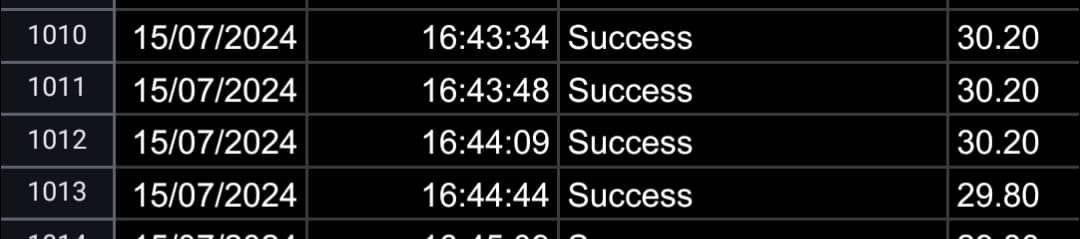


Figure 2 Données enregistrées sur google sheet à 16h43

1. Code

#include <LiquidCrystal\_I2C.h>

#include "WiFi.h"

#include <HTTPClient.h>

#include "DHT.h"

// Définitions des broches pour les composants

#define ON\_BOARD\_LED\_PIN 2

#define DHTPIN 4

#define DHTTYPE DHT11

#define FAN\_PIN 21

#define BUZZER\_PIN 33

#define GREEN\_LED\_PIN 18

#define RED\_LED\_PIN 19

// Initialisation de la bibliothèque LiquidCrystal\_I2C

LiquidCrystal\_I2C lcd(0x27, 16, 2);

DHT dht11(DHTPIN, DHTTYPE);

// Informations WiFi

const char\* ssid = "ENSI\_WIFI";

const char\* password = "Ensi-ul@2021\_";

// URL de l'application Web Google Script

const String WEB\_APP\_URL = "https://script.google.com/macros/s/AKfycbw5\_vMaaUyW73PwcZX9dCql7Z\_momI27U5t2ADwHec7JYkYnmnGBgsamW9Z1lNWUDaL/exec";

// Statut de lecture du capteur

String Status\_Read\_Sensor = "";

float Temp;

int Humd;

// Emplacement du caractère personnalisé pour le degré

#define DEGREE\_CHAR\_NUM 0

// Définition du caractère personnalisé pour le degré

byte degree\_Char[] = {

B00110,

B01001,

B01001,

B00110,

B00000,

B00000,

B00000,

B00000

};

// Fonction pour obtenir une valeur d'une chaîne de caractères en utilisant un séparateur

String getValue(String data, char separator, int index) {

int found = 0;

int strIndex[] = { 0, -1 };

int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++) {

if (data.charAt(i) == separator || i == maxIndex) {

found++;

strIndex[0] = strIndex[1] + 1;

strIndex[1] = (i == maxIndex) ? i + 1 : i;

}

}

return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

}

// Fonction pour obtenir les données du capteur DHT11

void Getting\_DHT11\_Sensor\_Data() {

Humd = dht11.readHumidity();

Temp = dht11.readTemperature();

if (isnan(Humd) || isnan(Temp)) {

Serial.println(F("Failed to read from DHT sensor!"));

Status\_Read\_Sensor = "Failed";

Temp = 0.00;

Humd = 0;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temp : Err");

lcd.setCursor(0, 1);

lcd.print("Humd : Err");

} else {

Status\_Read\_Sensor = "Success";

Serial.println("-------------");

Serial.printf("Status\_Read\_Sensor : %s | Humidity : %d%% | Temperature : %.2f°C\n", Status\_Read\_Sensor.c\_str(), Humd, Temp);

Serial.println("-------------");

if (Temp < 30.0) {

digitalWrite(GREEN\_LED\_PIN, HIGH);

digitalWrite(RED\_LED\_PIN, LOW);

digitalWrite(FAN\_PIN, LOW);

lcd.setCursor(0, 0);

lcd.print("T : ");

lcd.print(Temp);

lcd.print(" ");

lcd.write(DEGREE\_CHAR\_NUM);

lcd.print("C !Normal");

lcd.setCursor(0, 1);

lcd.print("Humd : ");

lcd.print(Humd);

lcd.print(" %");

Serial.println(Temp);

} else {

digitalWrite(GREEN\_LED\_PIN, LOW);

digitalWrite(RED\_LED\_PIN, HIGH);

digitalWrite(FAN\_PIN, HIGH);

digitalWrite(BUZZER\_PIN, HIGH);

delay(5000);

digitalWrite(BUZZER\_PIN, LOW);

lcd.setCursor(0, 0);

lcd.print("T : ");

lcd.print(Temp);

lcd.print(" ");

lcd.write(DEGREE\_CHAR\_NUM);

lcd.print("C !ALERTE");

lcd.setCursor(0, 1);

lcd.print("Humd : ");

lcd.print(Humd);

lcd.print(" %");

}

}

}

// Fonction pour envoyer des données à Google Sheets

void sendDataToGoogleSheets() {

if (WiFi.status() == WL\_CONNECTED) {

digitalWrite(ON\_BOARD\_LED\_PIN, HIGH);

String Send\_Data\_URL = WEB\_APP\_URL + "?sts=write";

Send\_Data\_URL += "&srs=" + Status\_Read\_Sensor;

Send\_Data\_URL += "&temp=" + String(Temp);

Send\_Data\_URL += "&humd=" + String(Humd);

Serial.println("Send data to Google Spreadsheet...");

Serial.println("URL : " + Send\_Data\_URL);

HTTPClient http;

http.begin(Send\_Data\_URL.c\_str());

http.setFollowRedirects(HTTPC\_STRICT\_FOLLOW\_REDIRECTS);

int httpCode = http.GET();

Serial.print("HTTP Status Code : ");

Serial.println(httpCode);

if (httpCode > 0) {

String payload = http.getString();

Serial.println("Payload : " + payload);

}

http.end();

digitalWrite(ON\_BOARD\_LED\_PIN, LOW);

}

}

// Fonction pour lire des données depuis Google Sheets

void setup() {

Serial.begin(115200);

Serial.println();

pinMode(GREEN\_LED\_PIN, OUTPUT);

pinMode(RED\_LED\_PIN, OUTPUT);

pinMode(FAN\_PIN, OUTPUT);

pinMode(BUZZER\_PIN, OUTPUT);

lcd.init();

lcd.backlight();

lcd.createChar(DEGREE\_CHAR\_NUM, degree\_Char);

lcd.clear();

pinMode(ON\_BOARD\_LED\_PIN, OUTPUT);

Serial.println(F("DHT11 test!"));

dht11.begin();

lcd.setCursor(0, 0);

lcd.print("WIFI mode : STA");

Serial.println("WIFI mode : STA");

WiFi.mode(WIFI\_STA);

delay(1000);

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

int connecting\_process\_timed\_out = 40; // 20 secondes

while (WiFi.status() != WL\_CONNECTED) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Connecting to");

lcd.setCursor(0, 1);

lcd.print("SSID");

digitalWrite(ON\_BOARD\_LED\_PIN, HIGH);

delay(500);

lcd.clear();

digitalWrite(ON\_BOARD\_LED\_PIN, LOW);

delay(500);

if (--connecting\_process\_timed\_out == 0) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Connecting");

lcd.setCursor(0, 1);

lcd.print("failed !");

delay(1000);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("ESP32 restarted");

delay(1000);

ESP.restart();

}

}

digitalWrite(ON\_BOARD\_LED\_PIN, LOW);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("WiFi connected");

lcd.setCursor(0, 1);

lcd.print("successfully.");

delay(1000);

Serial.println("WiFi connected");

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Get data from");

lcd.setCursor(0, 1);

lcd.print("DHT11");

delay(1000);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Please wait...");

delay(1000);

delay(2000);

}

void loop() {

Getting\_DHT11\_Sensor\_Data();

sendDataToGoogleSheets();

delay(2000); // Attente avant la prochaine lecture

}

1. Perspectives

* Ce projet ouvre la voie à plusieurs améliorations :  L'intégration de capteurs supplémentaires pourrait permettre une surveillance plus complète des conditions environnementales.
* L'utilisation d'algorithmes d'apprentissage automatique pourrait améliorer la détection de défauts et prédire les pannes avant qu'elles ne se produisent.
* Enfin, le développement d'une interface utilisateur mobile ou web permettrait une gestion et une surveillance à distance plus conviviales et efficaces du système.

1. Conclusion

Ce projet a démontré l'efficacité de l'ESP32 dans la surveillance des systèmes en temps réel. En utilisant des composants comme le DHT11, des LED, un buzzer et un ventilateur, nous avons créé un système capable de détecter et de réagir aux anomalies de température. L'ESP32 envoie également des données à Google Sheets via WiFi pour un suivi et une analyse faciles. Cette solution améliore la fiabilité et la sécurité des systèmes électriques.

1. Références

* <https://www.omega.fr/prodinfo/acquisition-de-donnees.html>
* [Acquisition de données (DAQ) - Le guide ultime | Dewesoft](https://dewesoft.com/fr/blog/c-quoi-l-acquisition-donnees)