Iot

Rapport de Développement du Projet IoT pour la Prévention et la Prédiction des Inondations

Projet realiser par:

-ANGUILET Joan-Yves Darys
-WAFFO Adonis

Introduction

Ce projet, réalisé via la plateforme Wokwi, a pour objectif de développer une solution IoT permettant la prévention et la prédiction des inondations en milieu urbain. Le projet se divise en trois phases principales : le déploiement de la station de surveillance météorologique, le système de nettoyage des canaux d'évacuation d'eau, et l'évaluation des risques d'inondations et la gestion des alertes.

Phase 1 : Déploiement de la Station de Surveillance Météorologique

Objectifs

1. Mesurer les conditions météorologiques (température et humidité) avec un capteur DHT22.
2. Détecter le niveau d'eau à l'aide d'un capteur analogique.
3. Envoyer les données collectées à une plateforme IoT (ThingsBoard) toutes les 5 secondes.

Matériel Utilisé

- Capteur DHT22 pour mesurer la température et l'humidité.
- Capteur de niveau d'eau connecté à une broche analogique de l'ESP32.
- Capteur à ultrason HC-SR04 pour mesurer la distance.

Code Explicatif

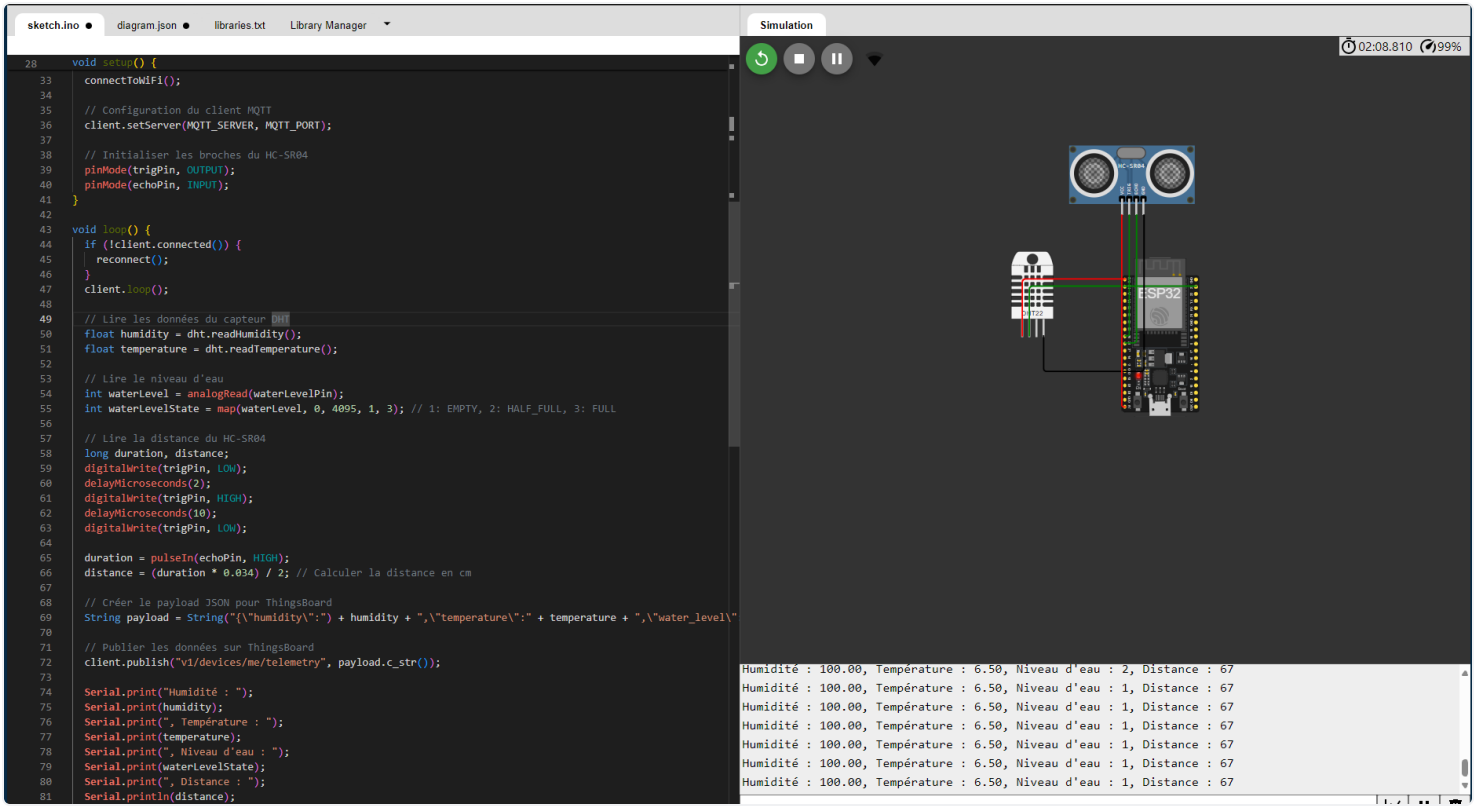
Le code ci-dessous explique chaque étape de la mise en place et de la collecte de données pour la station de surveillance météorologique :

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
```

```
// Paramètres Wi-Fi
#define SSID "Wokwi-GUEST"
#define PASS ""

// Paramètres MQTT
#define MQTT_SERVER "thingsboard.cloud"
#define MQTT_PORT 1883
#define TOKEN_TB "f4z0rbksk2bqqhamvskz"

// Configuration du capteur DHT
#define DHTPIN 23
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
WiFiClient espClient;
PubSubClient client(espClient);

const int waterLevelPin = 34; // Broche analogique pour le capteur de niveau d'eau

// Configuration du HC-SR04
const int trigPin = 25;
const int echoPin = 26;

void setup() {
  Serial.begin(115200);
  dht.begin(); // Initialiser le capteur DHT

  // Connexion au Wi-Fi
  connectToWiFi();

  // Configuration du client MQTT
  client.setServer(MQTT_SERVER, MQTT_PORT);

  // Initialiser les broches du HC-SR04
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Lire les données du capteur DHT
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Lire le niveau d'eau
```

```
  int waterLevel = analogRead(waterLevelPin);
  int waterLevelState = map(waterLevel, 0, 4095, 1, 3); // 1: EMPTY, 2: HALF_FULL, 3:
FULL

  // Lire la distance du HC-SR04
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration * 0.034) / 2; // Calculer la distance en cm

  // Créer le payload JSON pour ThingsBoard
  String payload = String("{\"humidity\":") + humidity + ",\"temperature\":" +
temperature + ",\"water_level\":" + waterLevelState + ",\"distance\":" + distance + "}";

  // Publier les données sur ThingsBoard
  client.publish("v1/devices/me/telemetry", payload.c_str());

  Serial.print("Humidité : ");
  Serial.print(humidity);
  Serial.print(", Température : ");
  Serial.print(temperature);
  Serial.print(", Niveau d'eau : ");
  Serial.print(waterLevelState);
  Serial.print(", Distance : ");
  Serial.println(distance);

  delay(5000); // Attendre 5 secondes avant la prochaine mesure
}

void connectToWiFi() {
  Serial.print("Connexion à ");
  Serial.print(SSID);
  WiFi.begin(SSID, PASS);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println(" connecté !");
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Tentative de connexion à MQTT...");
    if (client.connect("ESP32Client", TOKEN_TB, NULL)) {
```

```
      Serial.println("connecté !");
    } else {
      Serial.print("Échec de la connexion, code d'erreur : ");
      Serial.print(client.state());
      delay(2000);
    }
  }
}
```



Explication des Composants Clés

- **Connexion Wi-Fi** : La fonction `connectToWiFi` permet de connecter l'ESP32 au réseau Wi-Fi spécifié.
- **Capteur DHT22** : Utilisé pour mesurer l'humidité et la température, initialisé avec `dht.begin()`.
- **Lecture des Données** : Les données des capteurs sont lues et stockées dans des variables.
- **Communication MQTT** : Les données sont publiées sur ThingsBoard via MQTT.

Phase 2 : Déploiement du Système de Nettoyage des Canaux d'Évacuation d'Eau

Objectifs

1. Détecter la présence d'obstructions dans les canaux à l'aide d'un capteur à ultrason HC-SR04.
2. Utiliser un servomoteur pour enlever les obstructions détectées.
3. Envoyer des alertes à la plateforme ThingsBoard en cas d'obstruction.

Matériel Utilisé

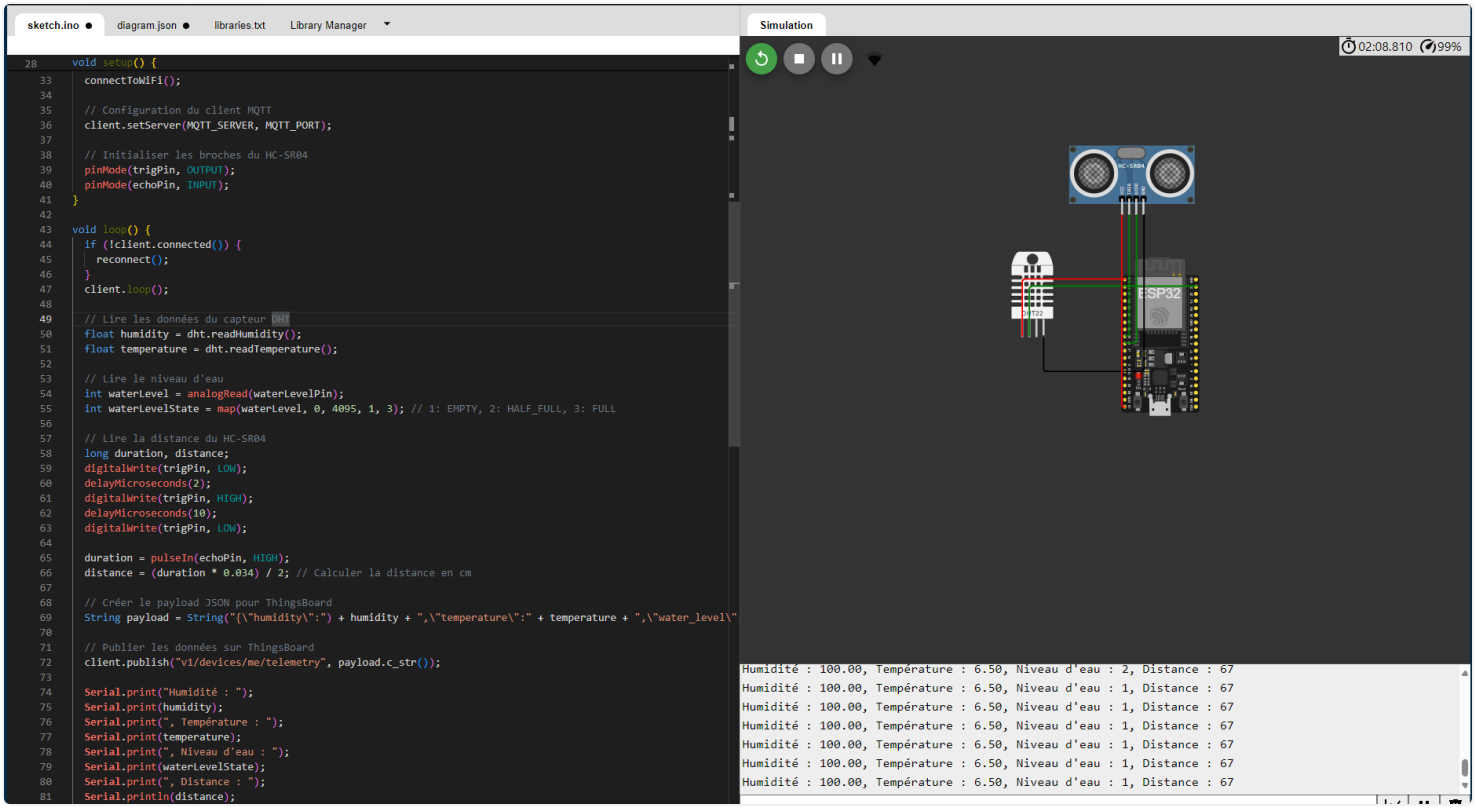
- Capteur à ultrason HC-SR04 pour la détection des obstructions.
- Servomoteur pour enlever les obstructions détectées.

Code Explicatif

Le code ci-dessous explique chaque étape de la mise en place et du fonctionnement du système de nettoyage des canaux d'évacuation d'eau :

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ESP32Servo.h>

// Paramètres Wi-Fi
#define SSID "Wokwi-GUEST"
#define PASS ""

// Paramètres MQTT
#define MQTT_SERVER "thingsboard.cloud"
#define MQTT_PORT 1883
#define TOKEN_TB "7ttpqfkz6ow2moq4hdpq" // Token du dispositif

WiFiClient espClient;
PubSubClient client(espClient);
Servo myServo; // Créer un objet servomoteur

const int trigPin = 2; // Pin de déclenchement
const int echoPin = 3; // Pin d'écho
const int servoPin = 9; // Pin du servomoteur

void setup() {
  Serial.begin(115200);
  myServo.attach(servoPin); // Attacher le servomoteur
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  connectToWiFi();
  client.setServer(MQTT_SERVER, MQTT_PORT);
  Serial.println("Simulation lancée!");
}

void loop() {
  long duration, distance;
```

```
  // Émettre un signal ultrasonique
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Lire le temps de réponse
  duration = pulseIn(echoPin, HIGH);

  // Calculer la distance
  distance = (duration * 0.034) / 2; // Distance en cm

  // Vérifier la distance et agir en conséquence
  if (distance < 5) {
    Serial.println("#1"); // Obstruction détectée
    Serial.println("Obstruction du Canal");
    if (client.connect("ESP32Client", TOKEN_TB, NULL)) {
      client.publish("v1/devices/me/telemetry", "{\"obstruction\":true}");
    }
  } else if (distance < 20) {
    myServo.write(180); // Soulever la plateforme
    delay(1000); // Attendre que le servomoteur se déplace
    myServo.write(0); // Ramener la plateforme
    Serial.println("#0"); // Désobstruction
    Serial.println("Desobstruction du canal");
    if (client.connect("ESP32Client", TOKEN_TB, NULL)) {
      client.publish("v1/devices/me/telemetry", "{\"obstruction\":false}");
    }
  } else {
    Serial.println("#0"); // Pas d'obstruction
    Serial.println("Pas d'obstruction");
  }

  delay(5000); // Attendre avant la prochaine mesure
}

void connectToWiFi() {
  Serial.print("Connexion à ");
  Serial.print(SSID);
  WiFi.begin(SSID, PASS);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println(" connecté !");
}
```



Explication des Composants Clés

- **Capteur HC-SR04** : Utilisé pour mesurer la distance des objets (déchets) dans le canal.
- **Servomoteur** : Utilisé pour soulever et enlever les déchets lorsqu'une obstruction est détectée.
- **Communication MQTT** : Envoi d'alertes à ThingsBoard en cas d'obstruction.

Phase 3 : Évaluation des Risques d'Inondations et Gestion des Alertes

Objectifs

1. Récupérer les données de la station météorologique et du système de nettoyage.
2. Utiliser un algorithme de machine learning pour prédire les précipitations.
3. Évaluer le risque d'inondation et envoyer des alertes appropriées.

```
import requests
import json

# Configuration de ThingsBoard
THINGSBOARD_URL = "http://thingsboard.cloud/api/v1/f4z0rbksk2bqqhamvskz"
HEADERS = {'Content-Type': 'application/json'}

# Fonction pour récupérer l'état du canal
```

```
def get_channel_status():
    # Remplacez par l'URL de votre API ou méthode pour obtenir l'état du canal
    response = requests.get("http://your_api_endpoint/channel_status")
    if response.status_code == 200:
        return response.json().get("obstruction", False)
    return False

# Fonction pour récupérer les dernières mesures
def get_latest_measurements():
    response = requests.get(f"{THINGSBOARD_URL}/telemetry", headers=HEADERS)
    if response.status_code == 200:
        return response.json()
    return None

# Fonction pour prédire la pluie (exemple simple)
def will_it_rain(humidity, temperature):
    # Implémentez votre modèle de prédiction ici
    # Exemple simple : si l'humidité est supérieure à 80%, il va probablement pleuvoir
    return humidity > 80

# Évaluer les risques d'inondation
def evaluate_flood_risk(water_level, will_rain):
    NULL, LOW, MODERATE, HIGH = 0, 1, 2, 3
    flood_risk = NULL

    if will_rain:
        if water_level == 1:  # EMPTY
            flood_risk = LOW
        elif water_level == 2:  # HALF_FULL
            flood_risk = MODERATE
        elif water_level == 3:  # FULL
            flood_risk = HIGH
    else:
        if water_level == 3:  # FULL
            flood_risk = LOW
        else:
            flood_risk = NULL

    return flood_risk

# Main
if __name__ == "__main__":
    channel_obstruction = get_channel_status()

    if channel_obstruction:
        print("Obstruction détectée dans le canal.")

        # Récupérer les dernières mesures
        measurements = get_latest_measurements()
        if measurements:
```

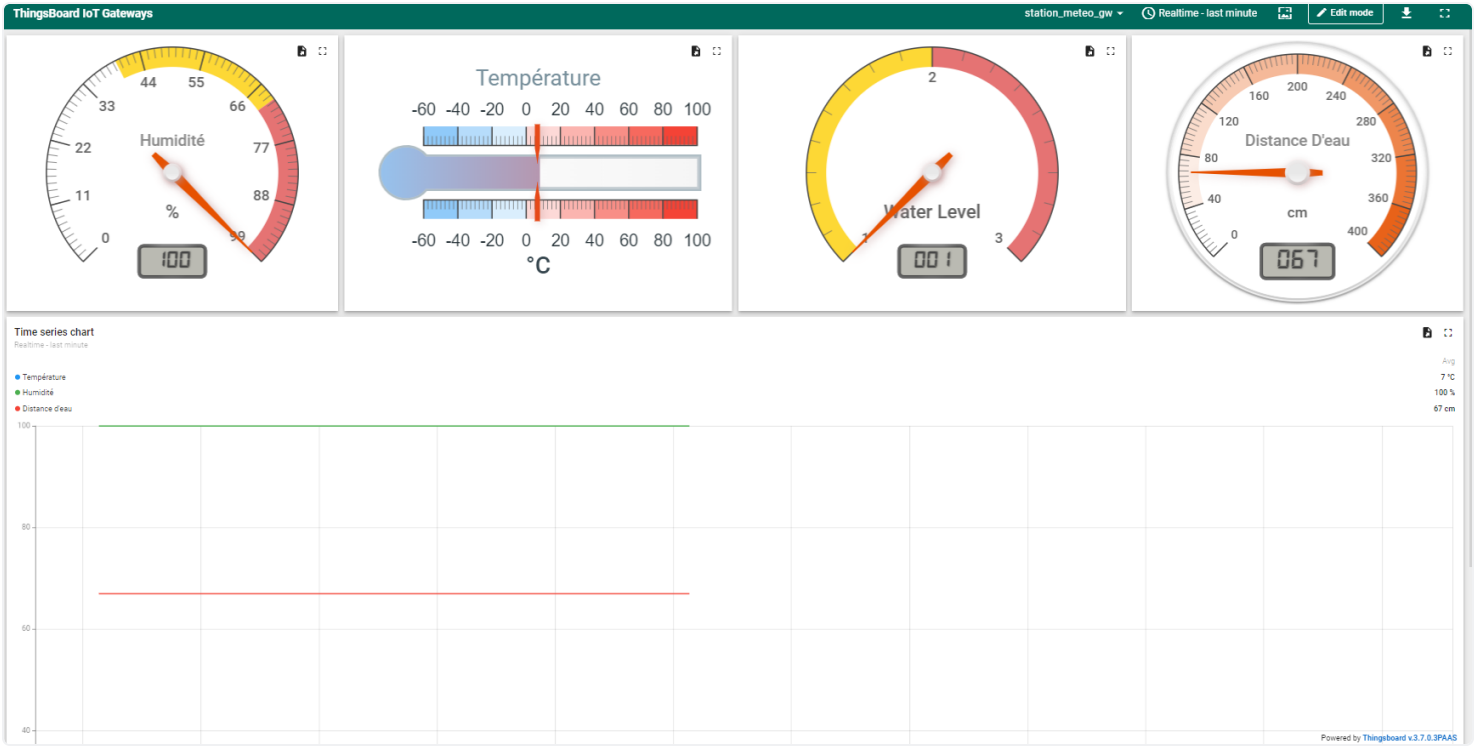
```
            humidity = measurements.get("humidity", 0)
            temperature = measurements.get("temperature", 0)
            water_level = measurements.get("water_level", 1)  # DEFAULT to EMPTY

            print(f"Mesures récupérées - Humidité: {humidity}, Température:
{temperature}, Niveau d'eau: {water_level}")

            # Prédire la pluie
            rain_prediction = will_it_rain(humidity, temperature)
            flood_risk = evaluate_flood_risk(water_level, rain_prediction)

            print(f"Niveau de risque d'inondation : {flood_risk}")
        else:
            print("Erreur lors de la récupération des mesures.")
    else:
        print("Aucune obstruction détectée dans le canal.")
```

Explication des Composants Clés

- **Lecture des Données** : Récupération des dernières mesures de ThingsBoard.
- **Évaluation des Risques** : Utilisation des règles prédéfinies pour évaluer les risques d'inondation.
- **Envoi d'Alerte** : Génération et envoi d'une alerte en fonction du risque évalué

```
PS C:\Users\JAN268\UrbanFloodPredictor> & C:/Users/JAN268/AppData/
/python3.12.exe c:/Users/JAN268/UrbanFloodPredictor/sys.py
Aucune obstruction détectée dans le canal.
PS C:\Users\JAN268\UrbanFloodPredictor>
```

Station Meteo



Le tableau de bord de ThingsBoard affiche les mesures en temps réel de la station météorologique :

1. **Humidité** : La jauge indique une humidité de 100%. Cela peut indiquer des conditions très humides, augmentant la probabilité de pluie.
2. **Température** : La jauge indique une température de 7°C. Une température plus basse associée à une humidité élevée peut également indiquer une probabilité accrue de précipitations.
3. **Niveau d'eau** : La jauge montre que le niveau d'eau est à 1, ce qui signifie qu'il est probablement vide ou très bas.
4. **Distance d'eau** : La jauge indique une distance de 67 cm. Cela montre la hauteur de l'eau mesurée par le capteur à ultrasons.

La partie inférieure du tableau de bord montre un graphique des séries temporelles des trois paramètres mesurés : température, humidité, et distance d'eau, permettant une surveillance continue et une analyse des tendances.

Station de nettoyage (widget)

visuel permettant d'afficher l'alert afin de savoir si il y a eu obstruction ou pas dans le canal

Conclusion

Ce projet a permis de développer une solution IoT complète pour la prévention et la prédiction des inondations en milieu urbain. Les différentes phases ont été mises en œuvre avec succès, de la collecte des données météorologiques à l'évaluation des risques d'inondation, en passant par le nettoyage des canaux d'évacuation. Cette solution peut être déployée dans les zones urbaines pour améliorer la gestion des inondations et minimiser les risques pour la population.