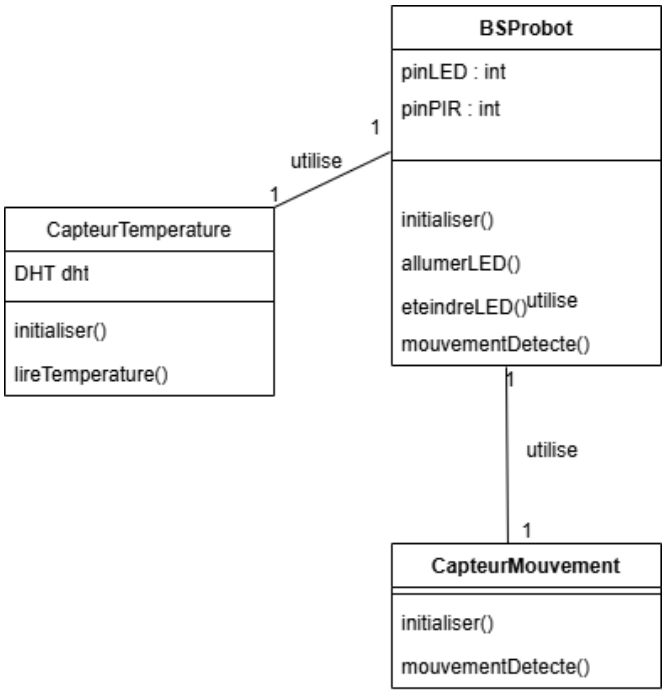
Conception et réalisation d'un système de détection de mouvement et de surveillance de la température avec ESP32, capteur PIR et capteur DHT22

But du tp :

Créer un système qui utilise un capteur de température et d'humidité (DHT22) pour mesurer la température et l'humidité, un capteur PIR pour détecter les mouvements, et une LED qui s'allume lors de la détection de mouvement.

Le projet complet est disponible sur GitHub (chaimabot/ESP32_LED_Control_Web_Server: Le projet **"ESP32 LED Control Web Server"** permet de contrôler deux LEDs via une interface web sur ESP32. Le serveur gère les requêtes HTTP pour allumer ou éteindre les LEDs.)

Diagramme de classe :



Matériel requis :

- Carte ESP32 DevKit C V4 : Pour gérer les entrées/sorties et exécuter le programme.
- Capteur de température et d'humidité DHT22 : Connecté au pin 4 de l'ESP32 pour lire la température et l'humidité.
- Capteur de mouvement PIR : Connecté au pin 5 de l'ESP32 pour détecter les mouvements.
- LED rouge : Connectée au pin 2 de l'ESP32, utilisée pour indiquer la détection de mouvement.
- Résistance de 220Ω : Pour protéger la LED en limitant le courant.

Câblage :

1. Capteur DHT22 :
    - VCC -> 3.3V de l'ESP32
    - GND -> GND de l'ESP32
    - SDA -> Pin 4 de l'ESP32
2. Capteur PIR :
    - VCC -> 3.3V de l'ESP32
    - GND -> GND de l'ESP32
    - OUT -> Pin 5 de l'ESP32
3. LED :
    - Anode (A) -> Pin 2 de l'ESP32
    - Cathode (C) -> GND via une résistance de 220Ω
4. Résistance de 220Ω :
    - Connectée entre la LED et le GND de l'ESP32

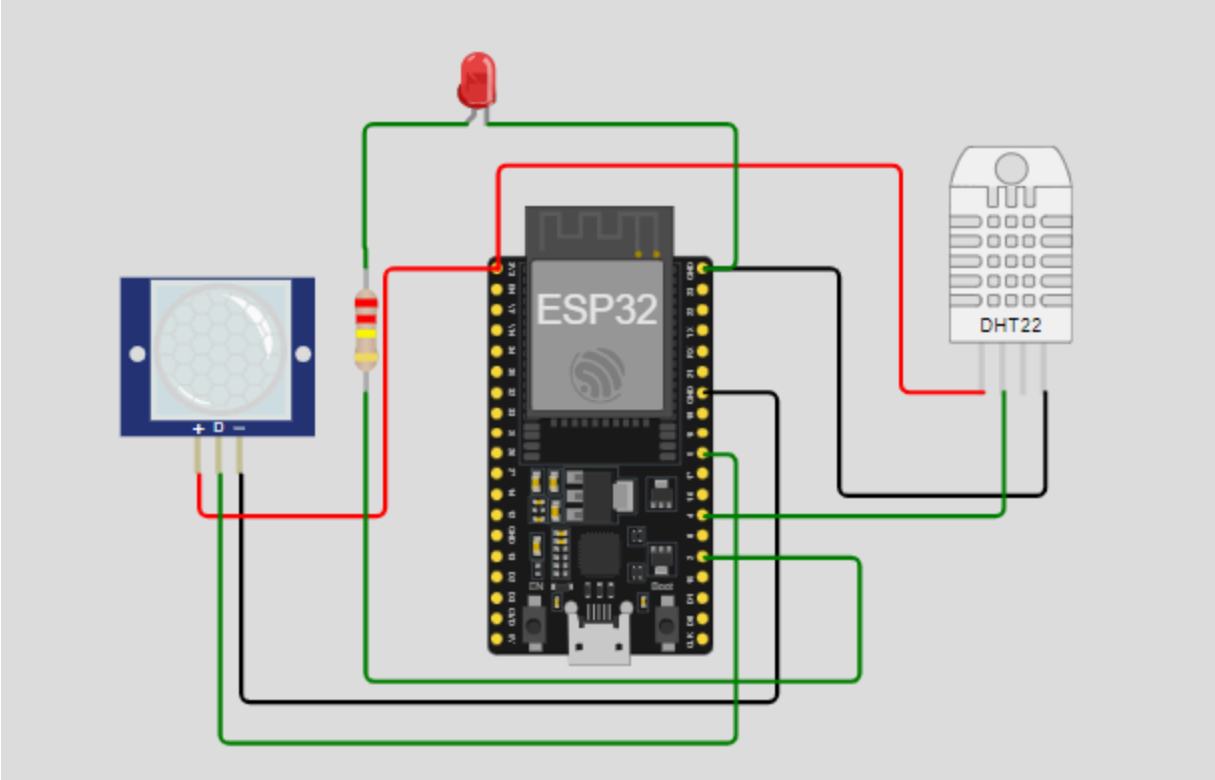


Figure 1 : Schéma

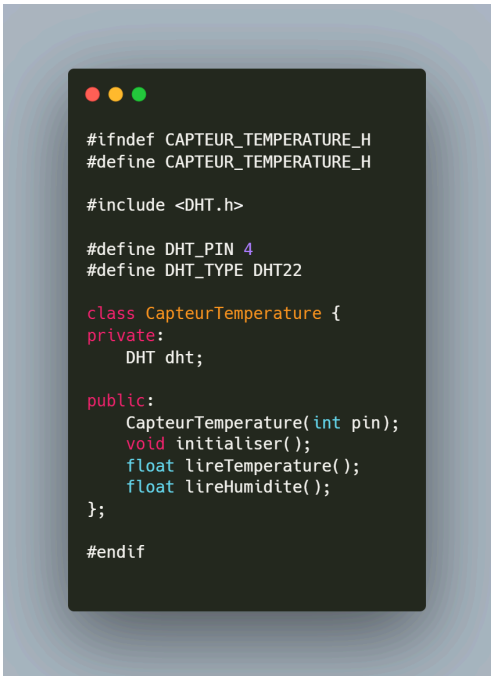
1. La classe CapteurTemperature (Gestion du capteur DHT22)

Attributs :

- **dht** : Un objet de la bibliothèque DHT qui gère le capteur de température et d'humidité.

Méthodes :

- CapteurTemperature(int pin) : Constructeur de la classe. Il initialise l'objet dht avec le pin où le capteur DHT22 est connecté.
- initialiser() : Initialise le capteur DHT22 (appelle la méthode begin() de la bibliothèque DHT).
- lireTemperature() : Lit la température à partir du capteur DHT22 et renvoie la valeur sous forme de flottant (°C).
- lireHumidite() : Lit l'humidité à partir du capteur DHT22 et renvoie la valeur sous forme de flottant (%).



```h
#ifndef CAPTEUR_TEMPERATURE_H
#define CAPTEUR_TEMPERATURE_H

#include <DHT.h>

#define DHT_PIN 4
#define DHT_TYPE DHT22

class CapteurTemperature {
private:
    DHT dht;

public:
    CapteurTemperature(int pin);
    void initialiser();
    float lireTemperature();
    float lireHumidite();
};

#endif
```

Figure 2 : CapteurTemperature.h

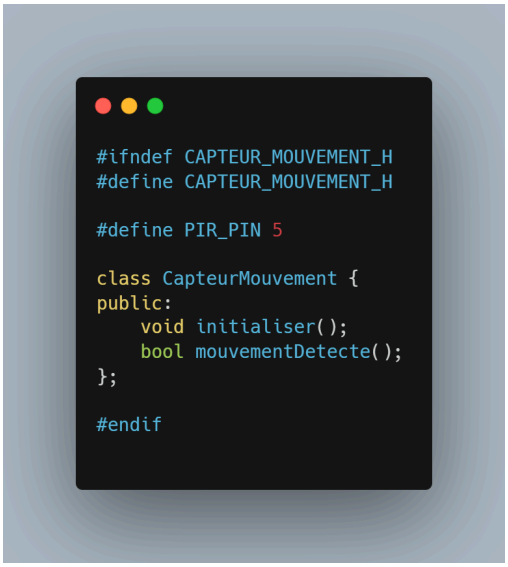
```cpp
#include "CapteurTemperature.h"

CapteurTemperature::CapteurTemperature(int pin) : dht(pin, DHT_TYPE) {}

void CapteurTemperature::initialiser() {
    dht.begin();
}

float CapteurTemperature::lireTemperature() {
    return dht.readTemperature();
}

float CapteurTemperature::lireHumidite() {
    return dht.readHumidity();
}
```

Figure 3 : CapteurTemperature.cpp

2. La classe CapteurMouvement (Gestion du capteur PIR)

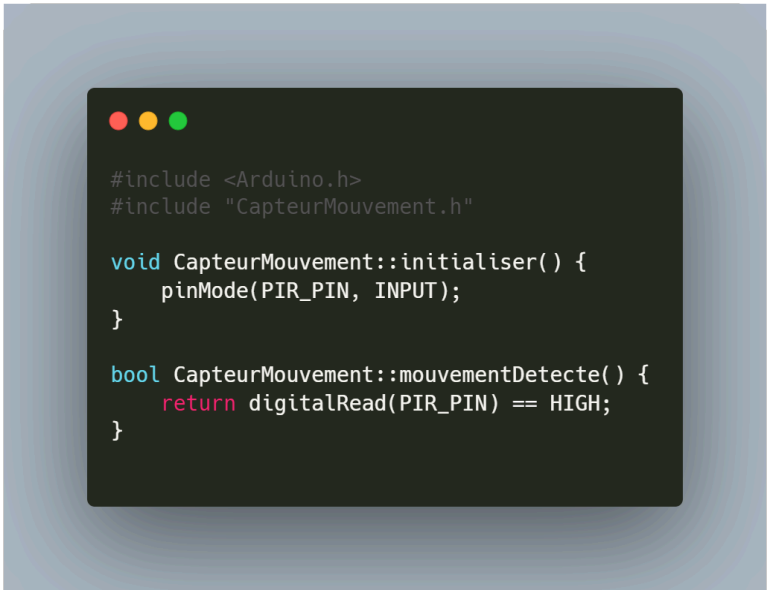
Méthodes :

- initialiser() : Initialiser le capteur PIR en configurant le pin connecté en mode INPUT.
- mouvementDetecte() : Vérifie si un mouvement a été détecté par le capteur PIR. Si le capteur détecte un mouvement, il renvoie true, sinon false.



```
#ifndef CAPTEUR_MOUVEMENT_H
#define CAPTEUR_MOUVEMENT_H

#define PIR_PIN 5

class CapteurMouvement {
public:
    void initialiser();
    bool mouvementDetecte();
};

#endif
```

Figure 4 : CapteurMouvement.h



```
#include <Arduino.h>
#include "CapteurMouvement.h"

void CapteurMouvement::initialiser() {
    pinMode(PIR_PIN, INPUT);
}

bool CapteurMouvement::mouvementDetecte() {
    return digitalRead(PIR_PIN) == HIGH;
}
```

Figure 5 : CapteurMouvement.cpp

3. La classe BSProbot (Gestion de la LED et du capteur PIR)

Attributs :

- **pinLED** : Pin auquel la LED est connectée.
- **pinPIR** : Pin auquel le capteur PIR est connecté.

Méthodes :

- BSProbot(int ledPin, int pirPin) : Constructeur de la classe. Initialise les pins pour la LED et le capteur PIR.
- initialiser() : Configure le pin de la LED en OUTPUT et le pin du capteur PIR en INPUT.
- allumerLED() : Allume la LED en mettant la pin à HIGH.
- eteindreLED() : Éteint la LED en mettant la pin à LOW.
- mouvementDetecte() : Vérifie si un mouvement a été détecté par le capteur PIR (via digitalRead() sur le pin du capteur).

```
#ifndef BSPROBOT_H
#define BSPROBOT_H

#include <Arduino.h>
#include "CapteurMouvement.h"

class BSProbot {
private:
  int pinLED;
  CapteurMouvement capteurMouv;

public:
  BSProbot(int ledPin = 2);

  void initialiser();
  void allumerLED();
  void eteindreLED();
  bool verifierMouvement();
};

#endif
```

Figure 6 : Bsprobot.h

```cpp
#include "BSProbot.h"

BSProbot::BSProbot(int ledPin) {
  pinLED = ledPin;
}
void BSProbot::initialiser() {
  pinMode(pinLED, OUTPUT);
  capteurMouv.initialiser();
}

void BSProbot::allumerLED() {
  digitalWrite(pinLED, HIGH);
}

void BSProbot::eteindreLED() {
  digitalWrite(pinLED, LOW);
}

bool BSProbot::verifierMouvement() {
  return capteurMouv.mouvementDetecte();
}
```

Figure 7 : bsprobot.cpp

4. Le fichier principal main (Gestion du flux principal)

Attributs :

- CapteurTemperature capteurTemp : Un objet de la classe CapteurTemperature pour gérer la lecture de la température et de l'humidité.
- CapteurMouvement capteurMouv : Un objet de la classe CapteurMouvement pour gérer la détection de mouvement.
- BSProbot robot : Un objet de la classe BSProbot pour gérer la LED et le capteur PIR.

Méthodes :

- setup() : Fonction d'initialisation. Elle initialise la communication série et configure les capteurs et la LED.
- loop() : Fonction principale exécutée en boucle. Elle :
    - Lit la température et l'humidité toutes les secondes.
    - Affiche ces valeurs sur le moniteur série.
    - Vérifie si un mouvement a été détecté par le capteur PIR. Si un mouvement est détecté, la LED s'allume. Sinon, la LED reste éteinte.

```ino
#include <Arduino.h>
#include "CapteurTemperature.h"
#include "CapteurMouvement.h"
#include "bsprobot.h"

CapteurTemperature capteurTemp(4);
CapteurMouvement capteurMouv;
BSProbot robot;

unsigned long previousMillis = 0;
const long interval = 1000;

void setup() {
  Serial.begin(115200);

  capteurTemp.initialiser();
  capteurMouv.initialiser();
  robot.initialiser();
}

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    float temp = capteurTemp.lireTemperature();
    float humidite = capteurTemp.lireHumidite();

    Serial.print("Température : ");
    Serial.print(temp);
    Serial.print(" °C | Humidité : ");
    Serial.print(humidite);
    Serial.println(" %");

    if (robot.mouvementDetecte()) {
      Serial.println("Mouvement détecté !");
      robot.allumerLED();
    } else {
      robot.eteindreLED();
    }
  }
}
```

Figure 8 : main.cpp

Ce TP montre comment créer un système avec l'ESP32 pour détecter les mouvements et mesurer la température et l'humidité. Il combine des capteurs (PIR et DHT22) et une LED, avec une programmation simple et efficace pour des applications pratiques comme la domotique.