DOCUMENTATION DU PROJET : SURVEILLANCE DE LA TEMPERATURE, DE L'HUMIDITE ET DE LA PRESSION AVEC ARDUINO MEGA ET FREERTOS

Introduction

Ce projet utilise une carte Arduino Mega 2560, le système d'exploitation temps réel FreeRTOS, un capteur DHT11 pour mesurer la température et l'humidité, ainsi qu'un capteur BMP280 pour mesurer la température et la pression atmosphérique. Les données collectées sont affichées sur le moniteur série.

FreeRTOS permet une gestion efficace des ressources grâce à la création de tâches simultanées pour la lecture des capteurs et l'affichage des données.

Matériels nécessaires

- Arduino Mega 2560
- Capteur DHT11 (température et humidité)
- Capteur BMP280 (température et pression)
- Câbles de connexion (Dupont)
- Breadboard (facultatif)
- Ordinateur avec l'IDE Arduino
- Bibliothèques Arduino :
    - `DHT.h` (pour le DHT11)
    - `Adafruit_BMP280.h` (pour le BMP280)
    - `Arduino_FreeRTOS.h` (pour FreeRTOS)

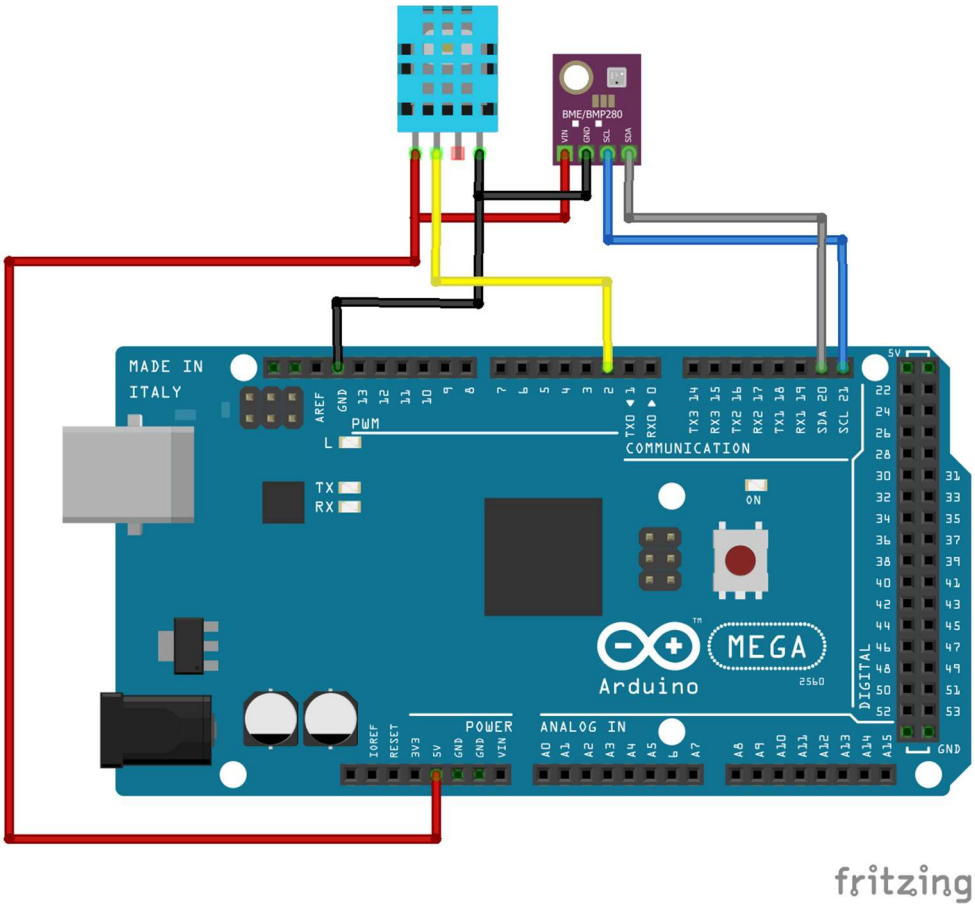
Schéma de câblage

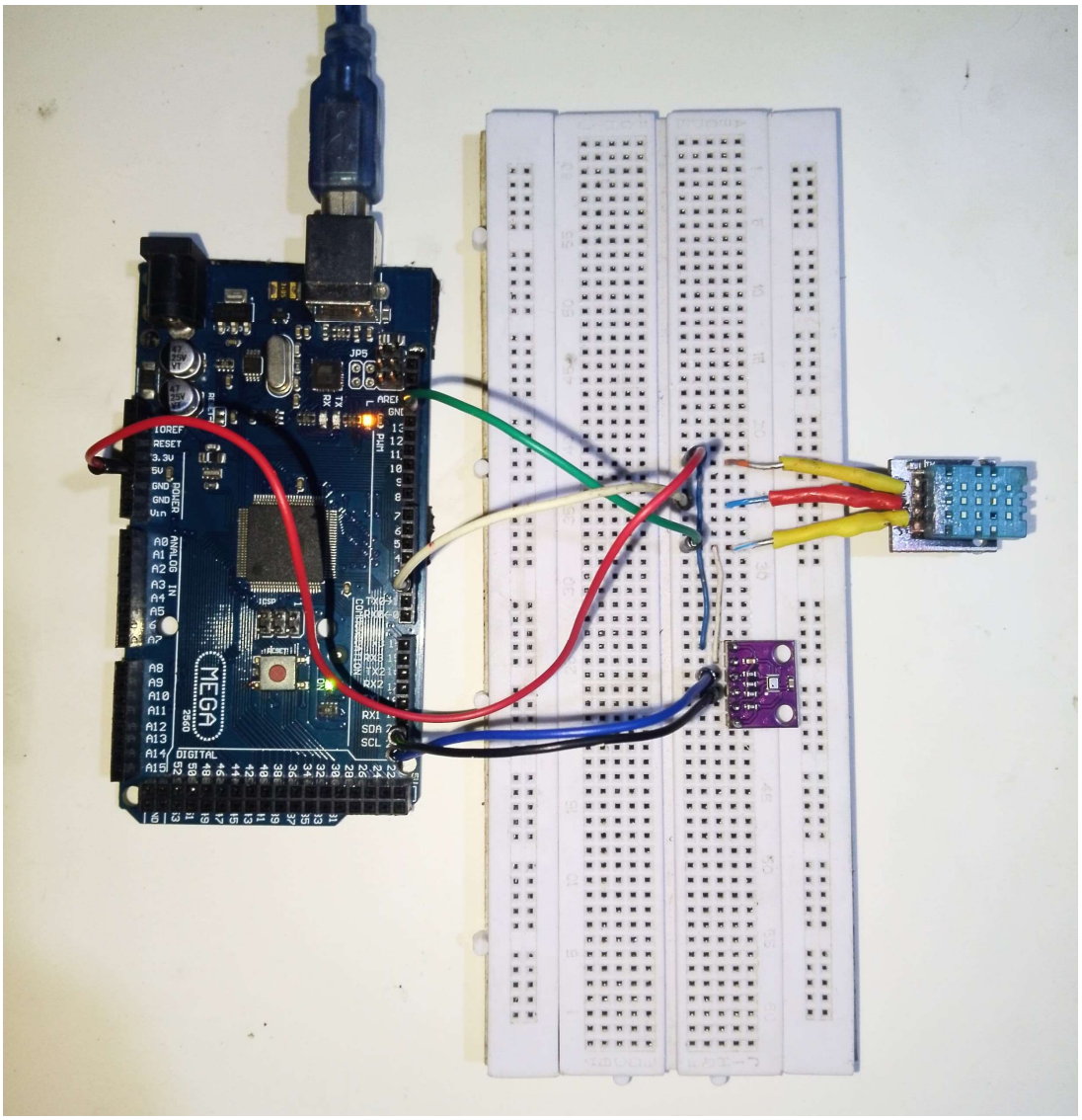
DHT11

Broche DHT11	Connexion Arduino
VCC	5V
GND	GND
DATA	Pin 2

BMP280

Broche BMP280	Connexion Arduino
VCC	3.3V
GND	GND
SCL	Pin 21 (SCL)
SDA	Pin 20 (SDA)





Fonctionnement du système

Description

1. **Lecture des capteurs** :
    - Le DHT11 mesure la température et l'humidité.
    - Le BMP280 mesure la température et la pression atmosphérique.
2. **Gestion des tâches** :
    - Une tâche est responsable de la lecture des données du DHT11.
    - Une autre tâche est dédiée à la lecture des données du BMP280.
    - Une troisième tâche affiche les données collectées sur le moniteur série.
3. **Planification** : FreeRTOS planifie et exécute les tâches en temps réel.

Avantages de FreeRTOS

- Permet une exécution parallèle des tâches.
- Optimise l'utilisation de la mémoire et des ressources processeur.
- Garantit une réponse rapide et fiable pour chaque tâche.

Code Source

Voici le code complet pour ce projet :

```
#include <Arduino_FreeRTOS.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <DHT.h>

#define DHTPIN 2      // Pin connectée au DHT11
#define DHTTYPE DHT11 // Type de capteur

DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP280 bmp;

// Variables globales
float temperatureDHT = 0.0;
float humidity = 0.0;
float temperatureBMP = 0.0;
float pressure = 0.0;

// Prototypes des tâches
void TaskReadDHT(void *pvParameters);
void TaskReadBMP(void *pvParameters);
void TaskPrintData(void *pvParameters);

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Initialisation...");

  // Initialisation du DHT11
  dht.begin();
  Serial.println("DHT11 initialisé avec succès.");

  // Initialisation du BMP280
  if (!bmp.begin(0x76)) {
    Serial.println("Erreur : BMP280 non détecté !");
    while (1);
  }
  Serial.println("BMP280 initialisé avec succès.");

  // Création des tâches
  xTaskCreate(TaskReadDHT, "Read DHT11", 256, NULL, 1, NULL);
  xTaskCreate(TaskReadBMP, "Read BMP280", 256, NULL, 1, NULL);
  xTaskCreate(TaskPrintData, "Print Data", 256, NULL, 1, NULL);
```

```
  // Démarrage du planificateur
  vTaskStartScheduler();
}

void loop() {
  // Vide, car FreeRTOS gère les tâches
}

void TaskReadDHT(void *pvParameters) {
  (void) pvParameters;

  while (1) {
    temperatureDHT = dht.readTemperature();
    humidity = dht.readHumidity();

    // Temporisation de 2 secondes
    vTaskDelay(2000 / portTICK_PERIOD_MS);
  }
}

void TaskReadBMP(void *pvParameters) {
  (void) pvParameters;

  while (1) {
    temperatureBMP = bmp.readTemperature();
    pressure = bmp.readPressure() / 100.0F; // Conversion en hPa

    // Temporisation de 500 ms
    vTaskDelay(500 / portTICK_PERIOD_MS);
  }
}

void TaskPrintData(void *pvParameters) {
  (void) pvParameters;

  while (1) {
    Serial.print("Température (DHT11) : ");
    Serial.print(temperatureDHT);
    Serial.println(" °C");

    Serial.print("Humidité (DHT11) : ");
    Serial.print(humidity);
    Serial.println(" %");

    Serial.print("Température (BMP280) : ");
    Serial.print(temperatureBMP);
    Serial.println(" °C");

    Serial.print("Pression (BMP280) : ");
    Serial.print(pressure);
    Serial.println(" hPa");

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

    // Temporisation de 1 seconde
    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
```

Tests et Validation

1. **Vérification des connexions** :
    - Assurez-vous que tous les câbles sont correctement connectés selon le schéma de câblage.
2. **Compilation et Upload** :
    - Chargez le code sur la carte Arduino Mega via l'IDE Arduino.
3. **Surveillance des sorties** :
    - Ouvrez le moniteur série (à 9600 bauds).
    - Vérifiez les valeurs affichées pour la température, l'humidité et la pression.

Améliorations possibles

1. **Ajout d'un écran LCD** pour afficher les données sans utiliser le moniteur série.
2. **Envoi des données à un serveur** pour la surveillance à distance.
3. **Utilisation de capteurs supplémentaires** pour mesurer d'autres paramètres (luminosité, qualité de l'air, etc.).

Conclusion

Ce projet montre comment utiliser FreeRTOS sur Arduino Mega pour gérer plusieurs tâches en parallèle et collecter des données à partir de capteurs multiples. Il peut être étendu pour inclure des fonctionnalités avancées comme la connectivité Internet ou l'intégration avec des syst