

Cahier des charges du projet Système Temps Réel

Conception d'un dispositif qui surveille l'humidité du sol autour
d'une plante



Réalisé par :

ABID Wissal

BOUZAIDA Nadine

3AGE1

Supervisé par :

M. JELASSI Khaled

Plan

- Introduction
- Présentation de projet
- Matériels et Logiciels
- Programmation
- Câblage
- Interface graphique
- Conclusion

Introduction

Ce cahier des charges vise à définir les exigences et spécifications pour la conception d'un dispositif de surveillance de l'humidité du sol. Ce projet s'attache à créer un système fonctionnel, utilisant FreeRTOS, capable de détecter et de signaler visuellement les niveaux d'humidité du sol pour prévenir les conséquences néfastes de la sécheresse sur la croissance des plantes. En mettant l'accent sur la simplicité et l'efficacité, ce dispositif cherche à offrir une solution automatisée et proactive pour garantir des conditions optimales de croissance.

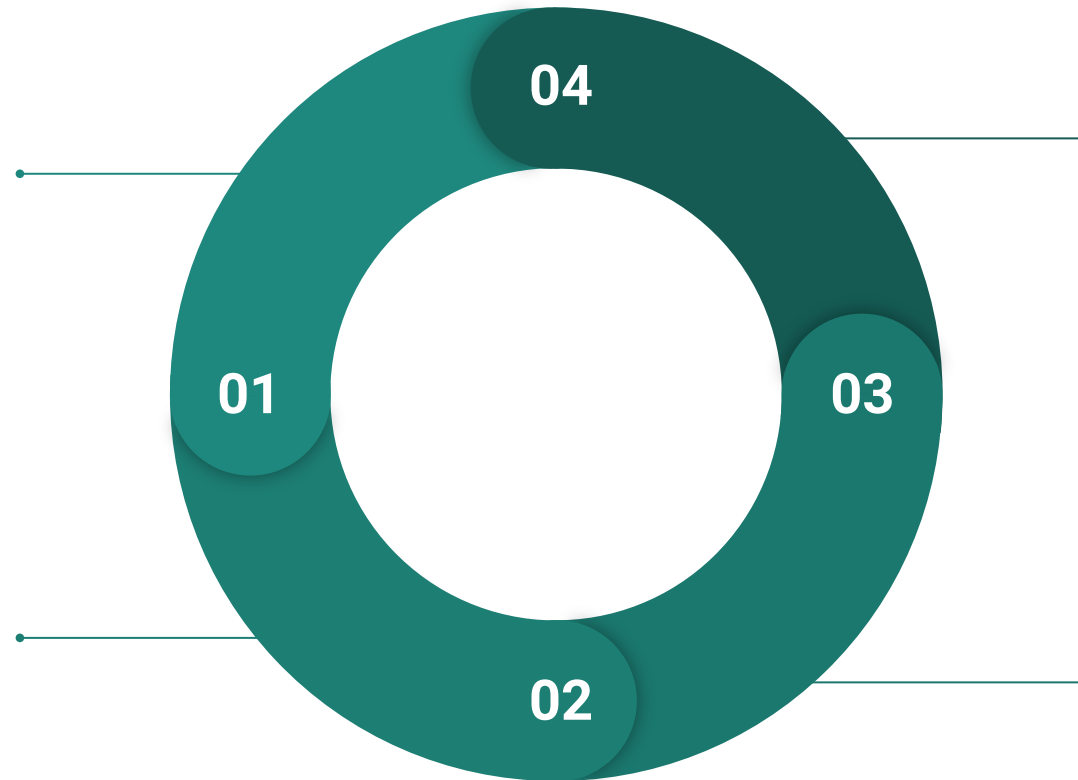
Cahier des charges

Données du Capteur d'Humidité

Les données reçues du capteur d'humidité du sol par le microcontrôleur STM32 seront traitées pour évaluer le niveau d'humidité du sol.

Transmission des Données

Les informations sur l'humidité du sol seront transmises du microcontrôleur STM32 vers un dispositif externe via USB. Ces données seront envoyées de manière continue pour permettre une surveillance en temps réel de l'humidité du sol.



Interface Utilisateur

Une interface sera développée pour visualiser l'évolution de l'humidité du sol en fonction du temps. Cette interface permettra de représenter graphiquement les variations d'humidité.

Gestion des Alertes

L'interface proposera un mécanisme d'alerte visuelle. En appuyant sur un bouton "Alerte", l'interface comparera la valeur actuelle de l'humidité du sol avec une valeur seuil définie. Si le niveau d'humidité dépasse cette valeur critique, une LED intégrée au système clignotera pour signaler un niveau d'humidité dangereusement bas.

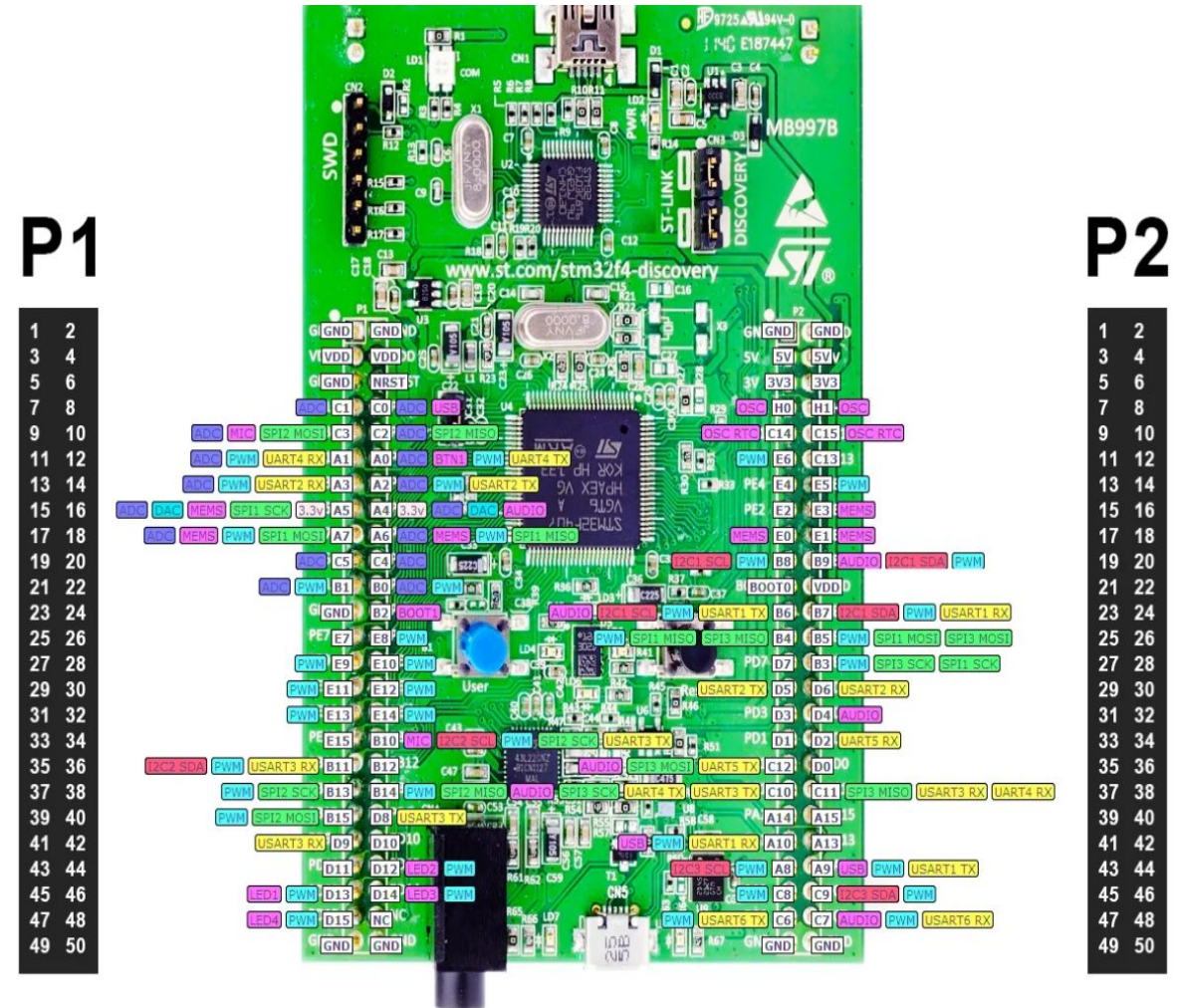


3

Matériels et Logiciels

STM32F407

Le STM32F407 est un microcontrôleur de la famille STM32, fabriqué par STMicroelectronics. Il appartient à la série STM32F4, qui est basée sur le noyau ARM Cortex-M4. Ce microcontrôleur est largement utilisé dans le domaine de l'électronique embarquée pour des applications telles que la robotique, l'automatisation, les systèmes embarqués, et bien d'autres. Il offre une puissance de traitement élevée, diverses interfaces de communication, des fonctionnalités de contrôle avancées, et il est programmable pour exécuter des tâches spécifiques dans des applications variées.



Capteur capacitif d'humidité du sol

Utilisé pour les plantes de jardin, la détection de l'humidité, l'agriculture intelligente.
Comprend un régulateur de tension intégré qui lui confère une plage de tension de fonctionnement de 3,3 ~ 5,5V.
Mesure les niveaux d'humidité du sol par détection capacitive.
Sortie analogique, insérez-la dans le sol et retour avec les données d'humidité du sol en temps réel.

Caractéristiques:

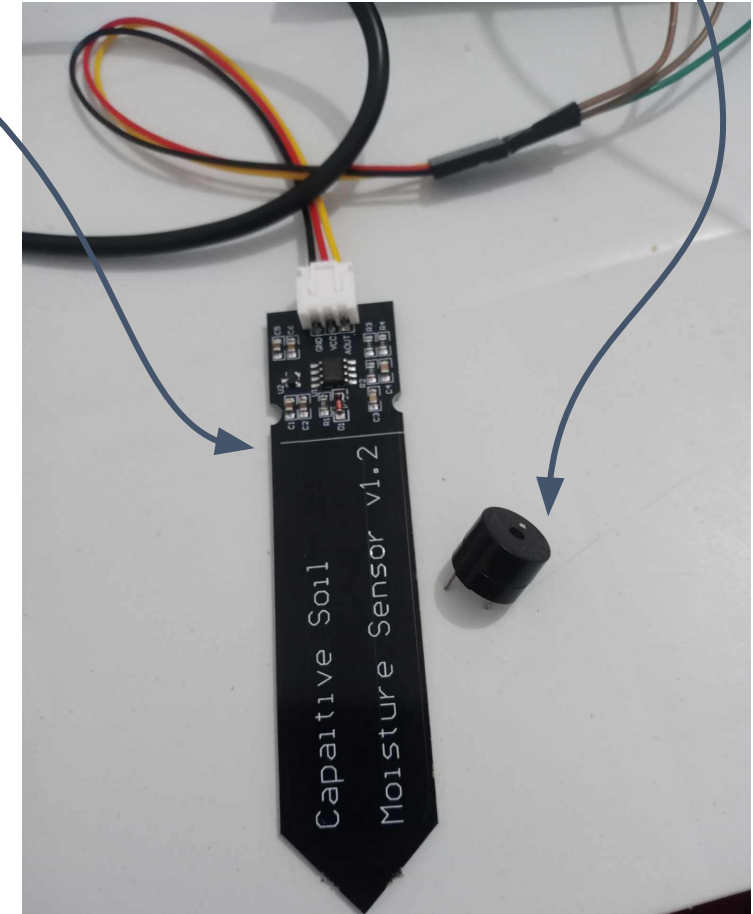
Tension de fonctionnement: 3,3 ~ 5,5 VDC

Tension de sortie: 0 ~ 3.0VDC

Interface: PH2.0-3P

Dimension: 98mm x 23mm / 3.86 "x 0.91"e

Buzzer



4

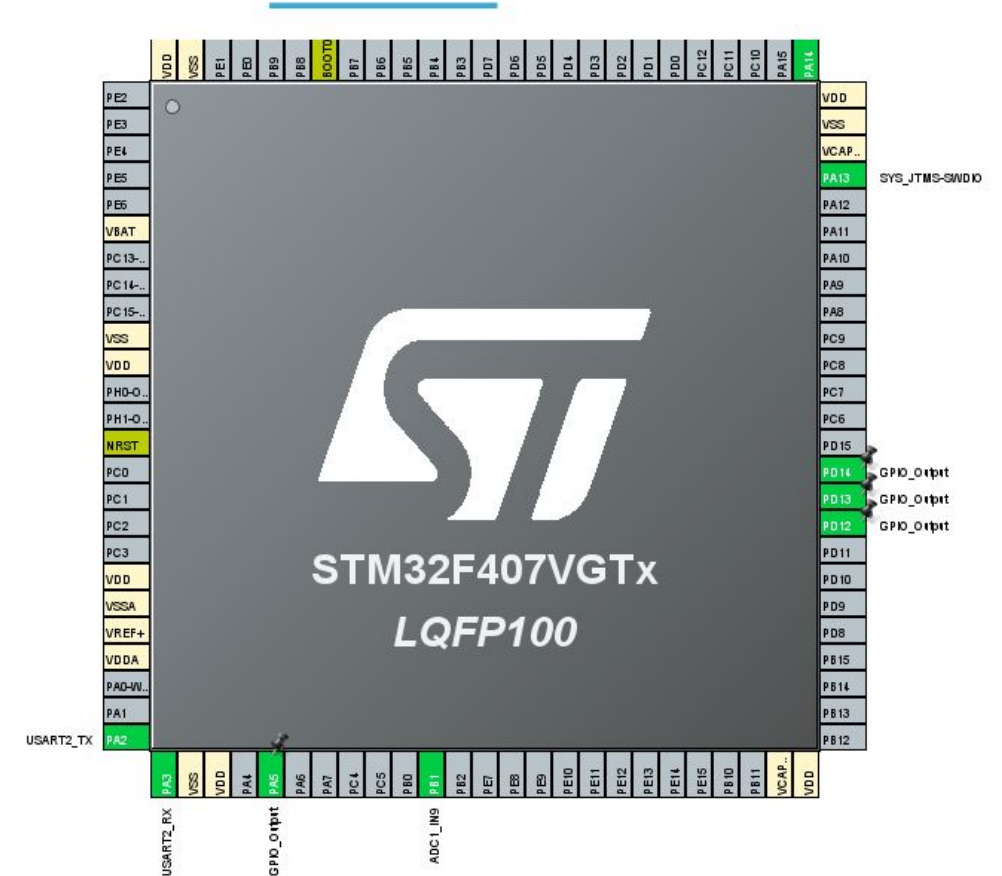
Programmation

Le code est conçu pour mesurer l'humidité du sol à l'aide d'un capteur analogique, contrôler un buzzer en fonction de cette mesure, et transmettre les données via UART.



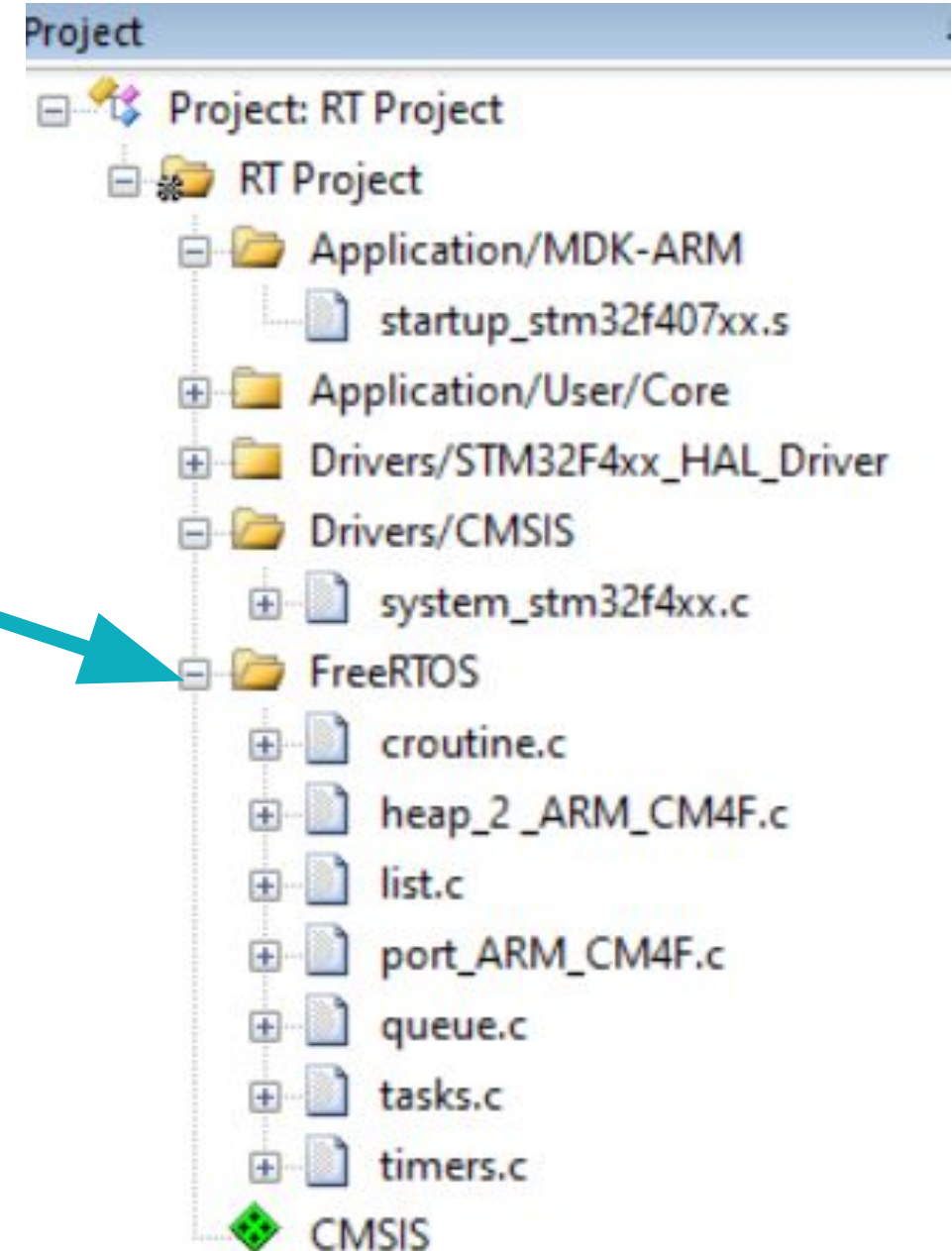
Utilisation de CubeMx

Nous commençons par configurer les broches à l'aide de l'outil graphique de configuration STM32CubeMX, puis nous procédons à la configuration de FreeRTOS.



Configuration de FreeRTOS

```
1  /* Includes -----  
2  #include "main.h"  
3  #include "freertos.h"  
4  #include "task.h"  
5  #include "semphr.h"  
6  |
```



Les Tâches :

```
/* Create tasks and start FreeRTOS scheduler */  
xTaskCreate(Task1, "task1", 128, NULL, 2, NULL); //pour activez le buzeer  
xTaskCreate(Task2, "task2", 128, NULL, 2, NULL); // pour activer les leds  
xTaskCreate(Task3, "task3", 128, NULL, 2, NULL); //semaphore  
  
vTaskStartScheduler();
```

La valeur de value = read()

read() : Cette fonction read semble être utilisée pour lire les données d'un capteur.

Task1

Contrôle du Buzzer en Fonction des Lectures d'un Capteur dans une Tâche FreeRTOS

```
75 // Tache 1 : activez le buzzer
76 void Task1(void) {
77     do {
78         xSemaphoreTake(xSemaphore1, portMAX_DELAY);
79         value = read();
80         if (value >= 300) {
81             HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 1);
82             HAL_Delay(100);
83         } else {
84             HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 0);
85             vTaskDelay(1000);
86         }
87     } while (1);
88 }
89
```

Task2

Contrôle des LEDs en Fonction des Lectures d'un Capteur dans une Tâche FreeRTOS :

- Si l'humidité est inférieure à 300, la LED 1 s'allume, indiquant une sécheresse.
- Si la valeur d'humidité est entre 300 et 600, cela correspond à un état normal et la LED 2 s'allume
- Si la valeur d'humidité est supérieure à 600, une alerte est déclenchée et la LED 3 s'allume.

```
90 // Tache 2 : activer les LEDs
91 void Task2(void) {
92     do {
93
94         xSemaphoreTake(xSemaphore1, portMAX_DELAY);
95         value=read();
96         if (value < 300) {
97             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
98             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 0);
99             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 0);
100         } else if (value < 300 || value > 600) {
101             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 1);
102             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 0);
103             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 0);
104         } else {
105             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 1);
106             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 0);
107             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 0);
108         }
109     } while (1);
110 }
111
```

Task3

Libération du Sémaphore
Périodiquement -

Dans cette tâche, le sémaphore est libéré à intervalles réguliers, permettant ainsi aux autres tâches de s'exécuter.

```
112 // Tache 3 : donner le sémaphore
113 void Task3(void) {
114     do {
115         xSemaphoreGive(xSemaphore1);
116         vTaskDelay(500);
117     } while (1);
118 }
119
```

Périphériques déclarés et initialisés dans le code

- **ADC (Convertisseur Analogique-Digital)**
 - **hadc1** : Configuré pour effectuer des conversions analogiques-numériques sur le canal **ADC_CHANNEL_9**.
- **Timer - htim14 :**

Configuré en mode base de temps pour générer des interruptions périodiques. Cependant, dans le code actuel, l'utilisation de ce timer n'est pas claire, car la fonction de rappel (callback) associée est vide.
- **UART (Universal Asynchronous Receiver-Transmitter)**
 - **huart2** : Utilisé pour la réception de données UART avec la fonction **HAL_UART_Receive_IT**.
- **GPIO (General Purpose Input/Output)**
- **Semaphore**
 - **xSemaphore1** : Les sémaphores sont utilisés pour synchroniser l'accès concurrent aux ressources partagées entre les tâches.

Nous avons utilisé Tera Term : pour visualiser les données

Tera Term: New connection

☐ TCP/IP

Host: myhost.example.com

☒ History

Service: ☐ Telnet

TCP port#: 22

☒ SSH

SSH version: SSH2

☐ Other

Protocol: UNSPEC

☒ Serial

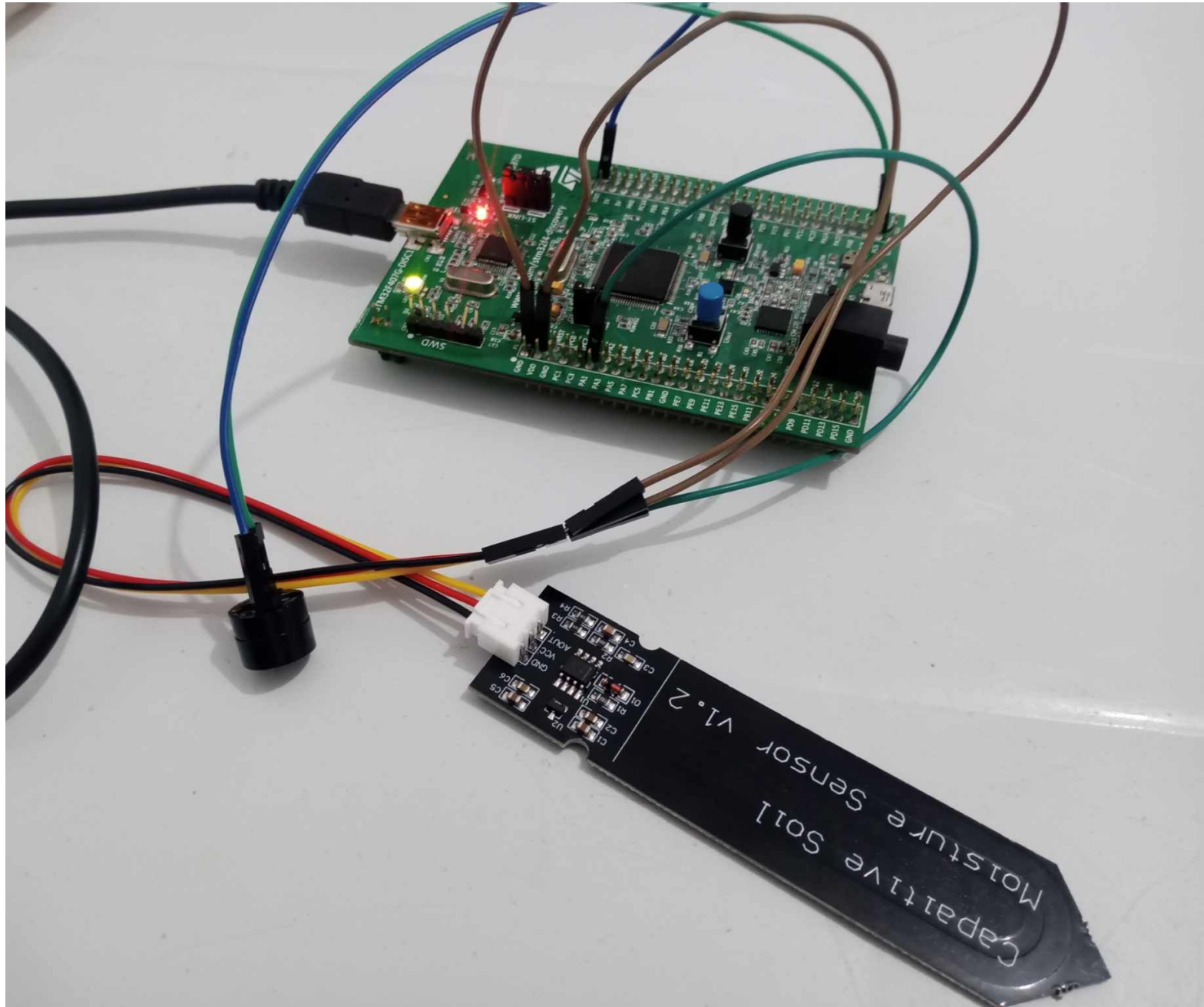
Port: COM3: STMicroelectronics STLink Virtu

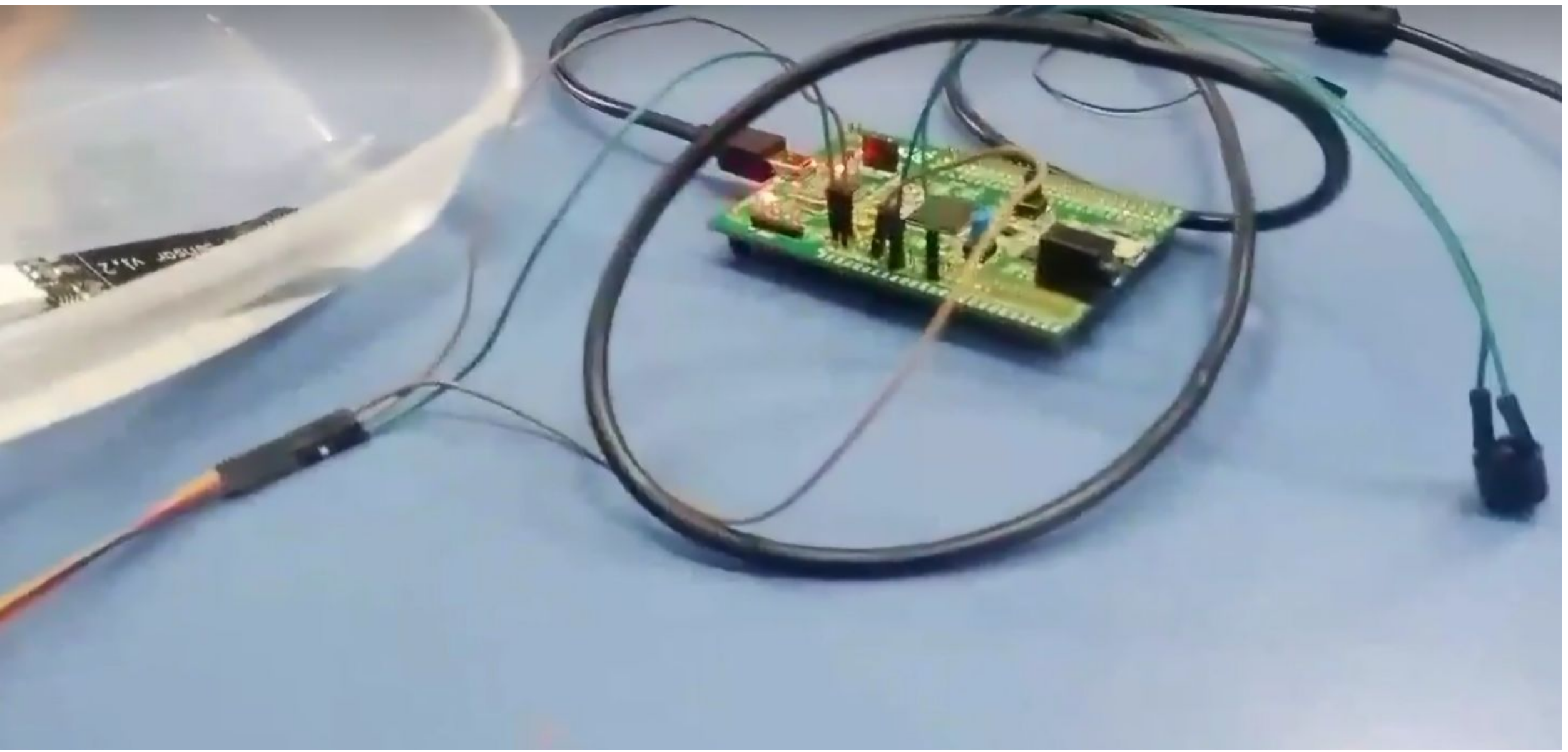
OK Cancel Help

5



Câblage :





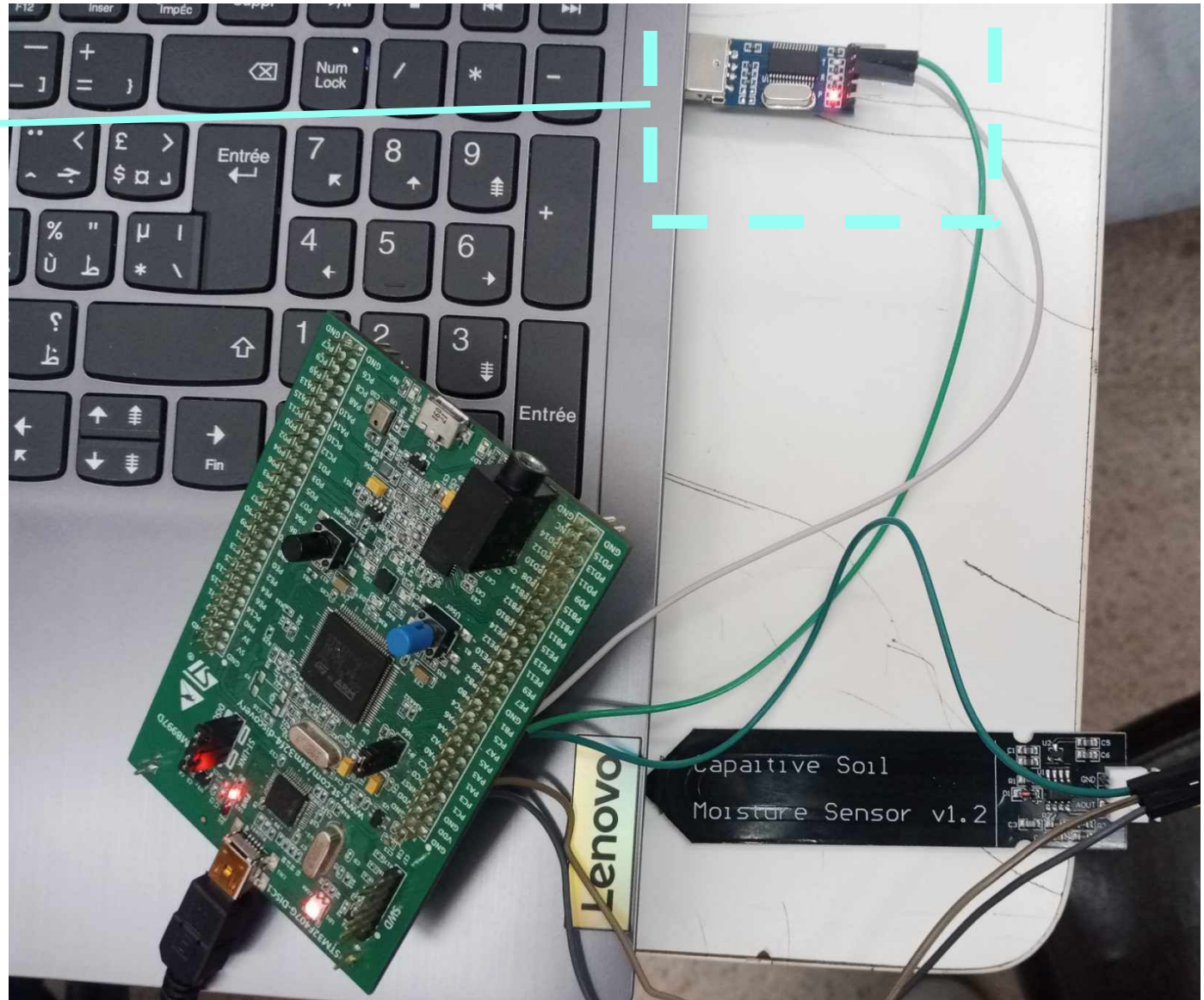


Nous avons mesuré les niveaux d'humidité et activé le buzzer en conséquence

Expression	Type	Value
 readValue	uint16_t	366
 Add new expression		

- NB :
 - Vous pouvez trouver la vidéo dans le dossier.
 - Nous avons travaillé sur ce code initialement dans un CubeIDE pour garantir le bon fonctionnement du matériel, avant même d'ajouter FreeRTOS. Vous

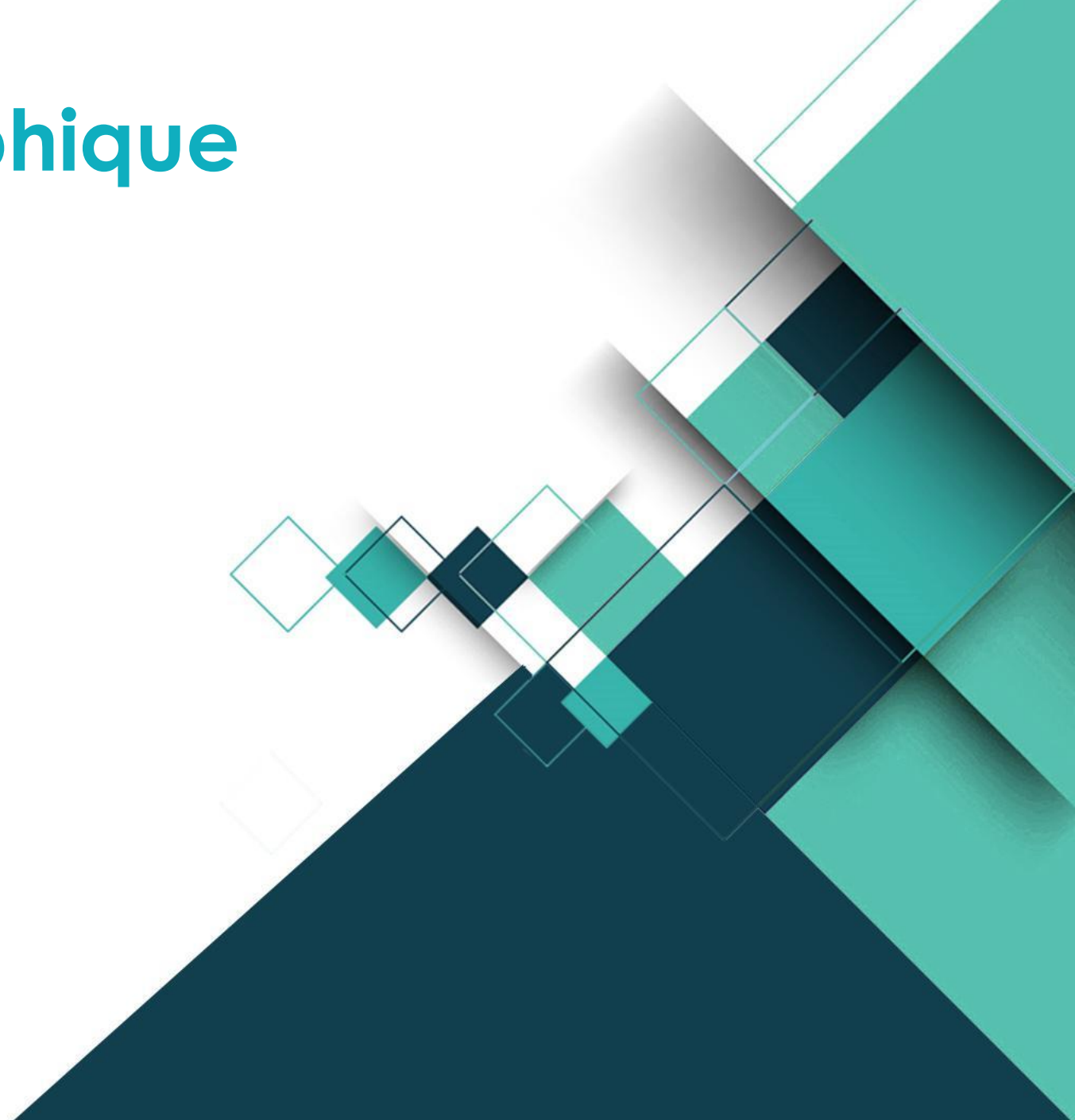
Nous avons utilisé le module adaptateur **USB-TT** après avoir configuré la connectivité USART sur la carte STM32F407, permettant ainsi la visualisation les valeurs d'humidité .



6

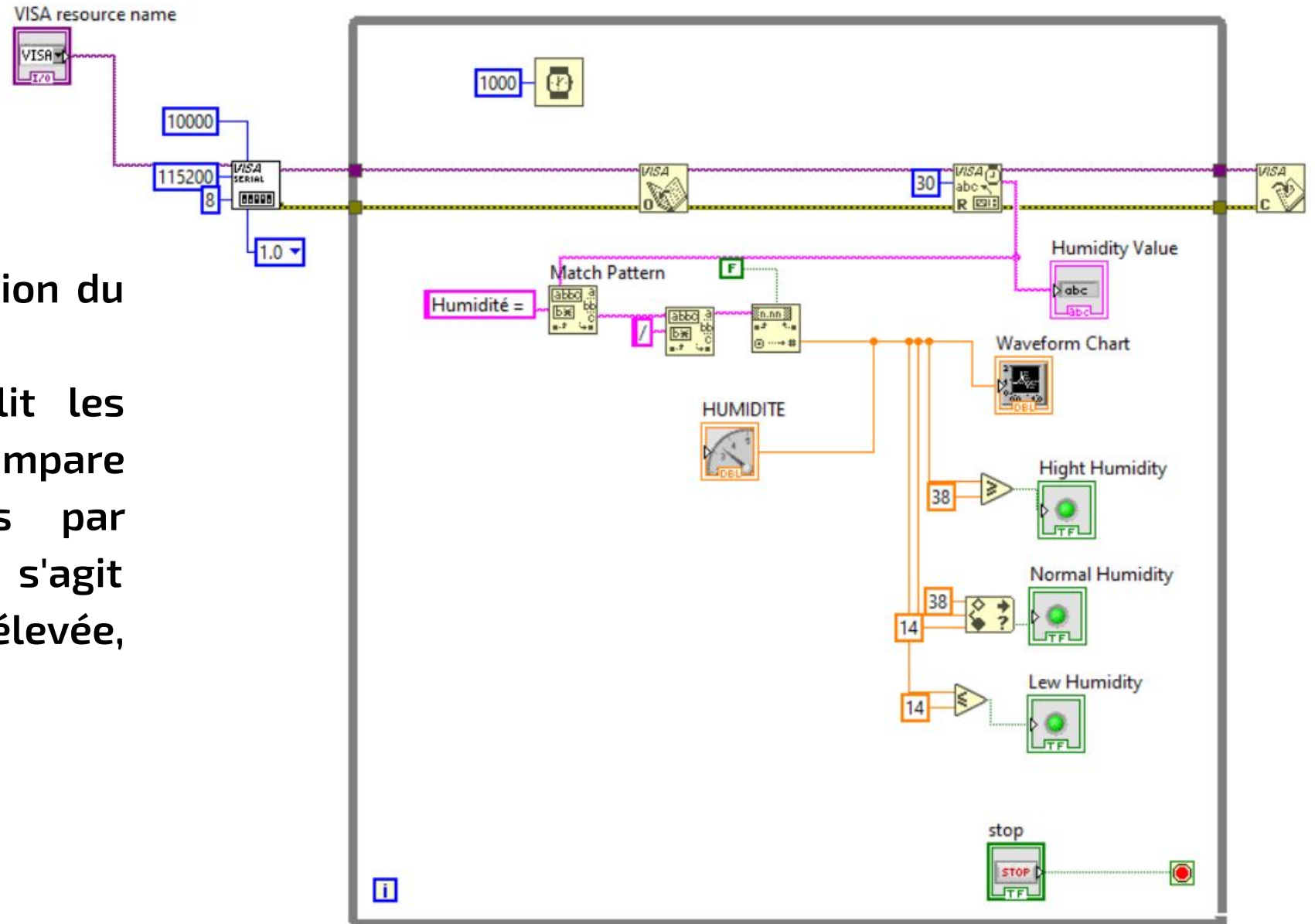
Interface Graphique

- Nous avons utilisé **LabVIEW**.
- Dans notre cas, nous devons lire le port série du module USB-TTL qui est connecté au PC.
- Nous devons donc utiliser les blocs "**VISA Serial**" qui nous permettent d'ouvrir le port série et de lire les données transmises par ce canal.
- Il suffit simplement de configurer la connectivité USART (**USART2**) sur la carte STM32F407.



On commence par la création du diagramme de blocs.

Il s'agit d'un code qui lit les valeurs d'humidité, les compare avec les seuils définis par l'utilisateur, et affiche s'il s'agit d'une température élevée, normale ou basse.



VISA resource name

1%

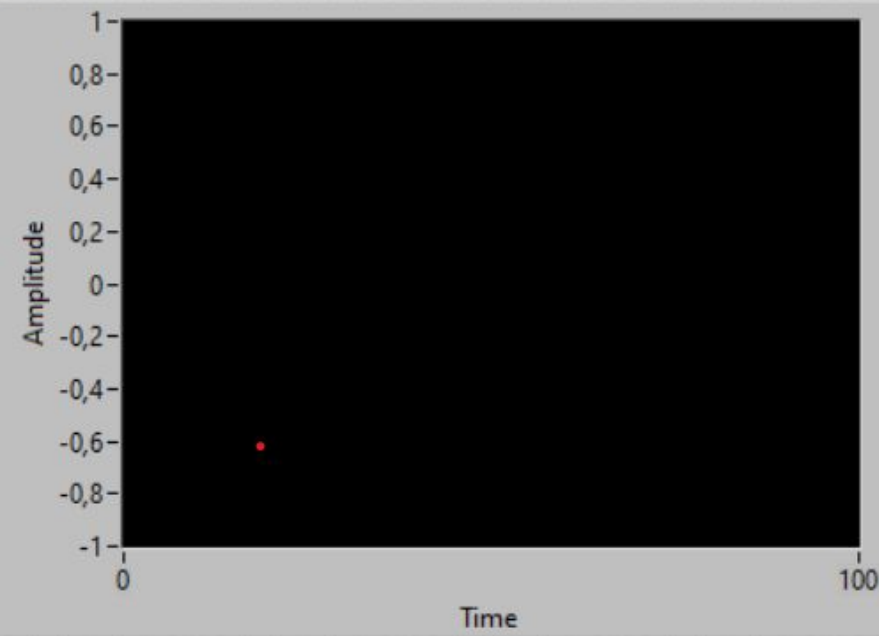
stop

STOP

Humidity Value

Waveform Chart

Plot 0



HUMIDITE



Low Humidity



Normal Humidity



Hight Humidity



Conclusion

- L'intégration du système de surveillance d'humidité du sol avec FreeRTOS apporte une gestion efficace des tâches, améliorant ainsi la fiabilité et la réactivité du dispositif.
- L'incorporation de FreeRTOS dans la conception renforce la robustesse du dispositif, offrant une solution fiable et évolutive pour la surveillance de l'humidité du sol, tout en permettant une adaptation aisée à des fonctionnalités futures.

Perspectives

- Nous avons envisagé un problème de communication série car nous ne parvenions pas à lire les données à partir du capteur analogique. Finalement, nous avons découvert qu'il était nécessaire d'utiliser un câble USB-TT. Cependant, en raison de contraintes de temps et de ressources, nous n'avons pas réussi à travailler efficacement avec ce câble.



**MERCI
POUR VOTRE ATTENTION**