Une image contenant texte

Description générée automatiquement

Rapport de projet

Une image contenant équipement électronique, circuit

Description générée automatiquement

BE Web Server Météo STM32

JACQ Léo

YALA AMINE

Année 2021/2022

Table des matières

[Introduction 3](#_Toc101189268)

[Centrale 4](#_Toc101189269)

[Présentation 4](#_Toc101189270)

[Structure du programme STM32 5](#_Toc101189271)

[Configuration des pin 5](#_Toc101189272)

[Structuration du main 6](#_Toc101189273)

[Récupération des données 8](#_Toc101189274)

[Module Ethernet 8](#_Toc101189275)

[Structure du programme Arduino 10](#_Toc101189276)

[Capteur 13](#_Toc101189277)

# Introduction

Les stations météos comportent souvent une centrale de capture fixe permettant l’acquisition des données sur une zone assez large. Le souci de cela est que pour obtenir plus de données il faut multiplier ces bases qui, bien que performantes, représentent un cout élevé en matériel et installation.

Le but de ce projet est de montrer qu’avec un réseau utilisant la technologie XBEE il est possible de multiplier ces zones de capture en les faisant communiquer et centraliser ces données pour les consulter.

La réalisation de ce projet se fera sur une base de carte STM32 NUCLEO avec différents capteurs.

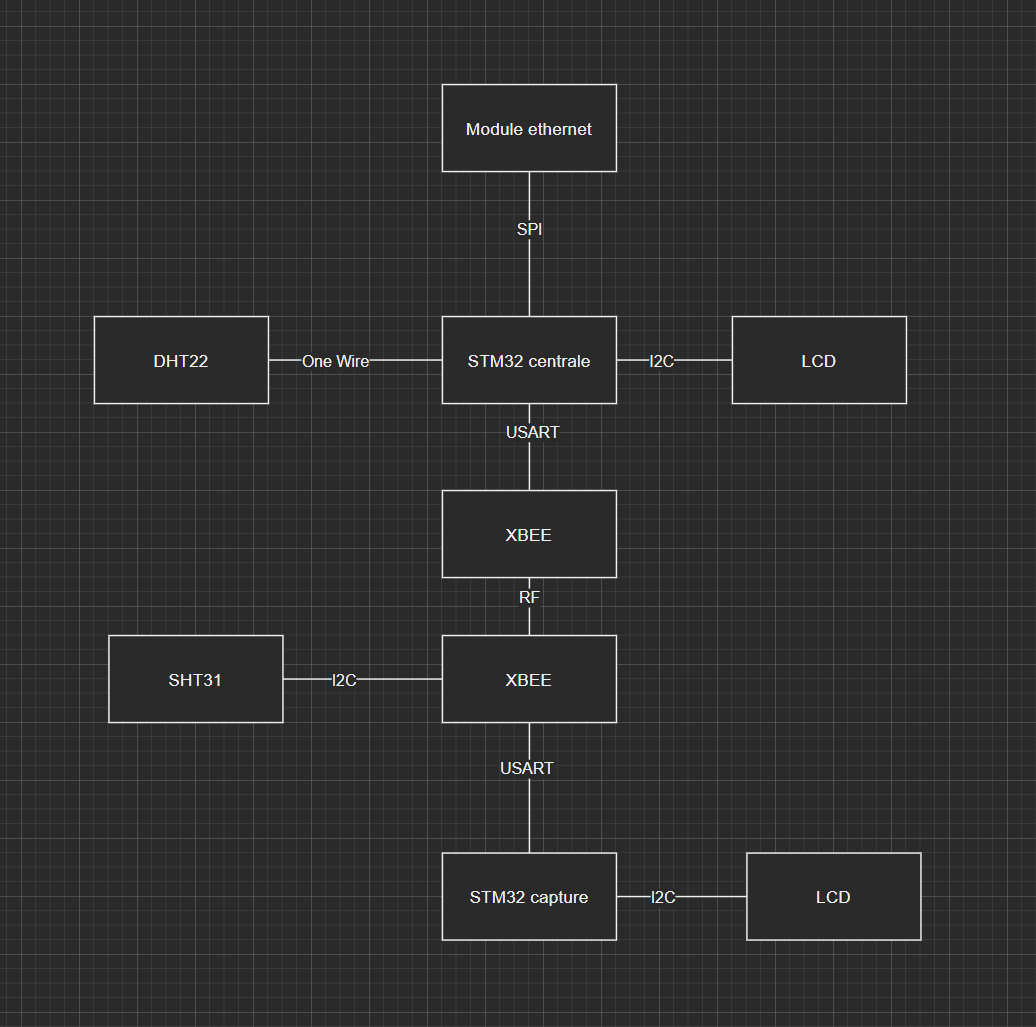
# Centrale

## Présentation

La partie centrale va devoir récupérer les données des capteurs afin de les afficher sur une page web. Un capteur sera aussi présente sur le dispositif central afin d’obtenir des données de températures et d’humidité localement.

Les données sont récupérées par mode de communication XBEE et transmises sur le réseau en Ethernet.

Le diagramme suivant illustre la communication entre chaque dispositif :



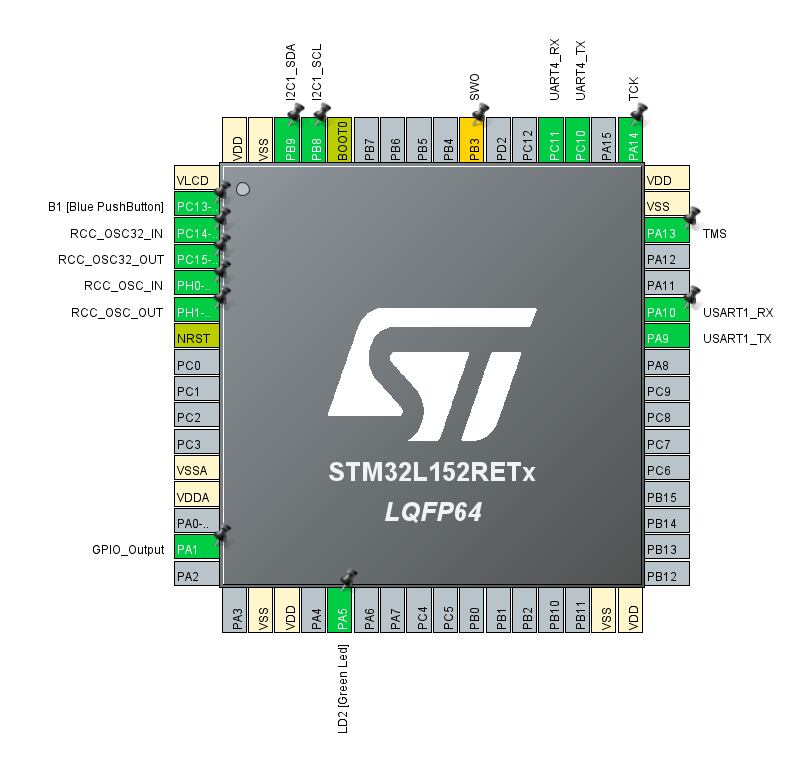
*Figure 1 : Diagramme général*

## Structure du programme STM32

### Configuration des pin

La carte Nucléo est donc équipée d’un module Xbee, d’un capteur DHT22 et d’un écran LCD.

Le module Xbee sert à la communication avec les autres capteurs dont il va réceptionner les données qui seront transmises à l’Arduino.



*Figure 2 : Définition des pins*

La figure ci-dessus présente la définition des pins pour les différents périphériques :

-I2C : SDA -> PB9 ; SCL -> PB8

-USART1 : RX -> PA10 ; TX -> PA9

-UART4 : RX -> PC11 ; TX -> PC10

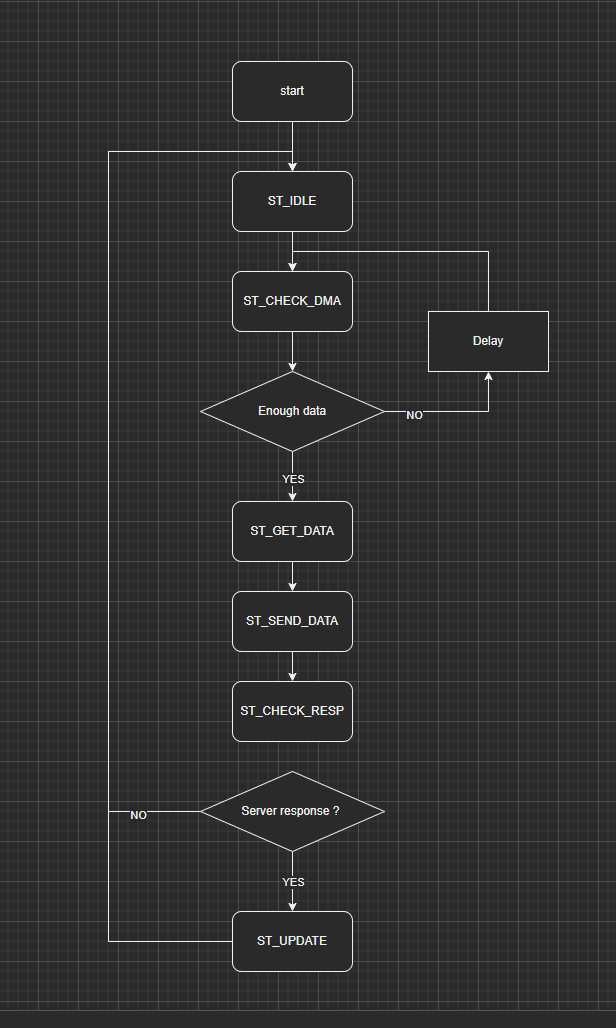
-DHT22 One Wire : PA1 output

### Structuration du main

Le programme est structuré avec une machine à état comportant les différentes parties de la routine principale.

Après le démarrage et l’initialisation des différents modules et périphérique, la routine principale commence.

* ST\_IDLE est un état de transition permettant la remise à zéros de certaines variables.
* ST\_CHECK\_DMA va vérifier si une interruption DMA a eu lieu ou si le programme doit se mettre en attente. Une fois les données DMA complètes et l’interruption levée, vient la collecte des données relative au dispositif centrale.
* ST\_GET\_DATA le même fonctionnement que celui présent sur les TP de base, à savoir la collecte de données de température et d’humidité ainsi que leur affichage grâce à l’écran LCD et au DHT22.
* ST\_SEND\_DATA va formater le paquet d’envois en concaténant les données du module centrale et des capteurs et transmettre la trame en UART à l’Arduino qui s’occupera de mettre à jour la page WEB.
* ST\_CHECK\_SERV\_RESP va vérifier si le serveur renvois des instructions à la carte en extrayant le message.
* ST\_UPDATE va renvoyer les instructions aux différents modules capteurs si le serveur à spécifié l’action.



*Figure 3 : Machine à états*

### Récupération des données

Les données capteurs n’arrivant pas à intervalle régulier, j’ai décidé d’implémenter un DMA sur la liaison RX de l’uart XBEE afin de pouvoir libérer du temps de traitement processeur et les manipulations de données. Tous les octets récupérés par le Xbee vont être inscrite à l’adresse mémoire spécifiée au DMA et la réception des données va pouvoir se faire en parallèle des autres taches sans manquer le moindre octet.

Les trames Xbee ont été définies avec un format fixe afin de simplifier leur utilisation. Des constantes vont alors pouvoir être utilisées pour définir la taille du buffer DMA, du nombre de dispositif connectés.

Les packages HAL implémentés dans le projet avec CubeIDE permettent de gérer plus facilement les interruptions et le DMA, pour le mettre en plus il suffit de le spécifier avec l’interface graphique et d’appeler la fonction HAL\_Receive\_DMA pour activer le procéder

Une image contenant texte

Description générée automatiquement

*Figure 4 : initialisation DMA*

## Module Ethernet

Le module Ethernet est un Arduino Ethernet shield 2. Ce dispositif s’adapte aux cartes format Arduino Uno et est compatible physiquement avec les carets STM32 Nucléo.

C’est un module basé sur le chipset Ethernet Wiznet W5500 qui propose une communication SPI ainsi qu’un port pour carte SD

Nativement les cartes STM32 Nucléo ne sont pas compatibles avec ce module, car ne dispose pas du connecteur Arduino ICPS nécessaire à la communication SPI.

Une image contenant texte, équipement électronique, circuit

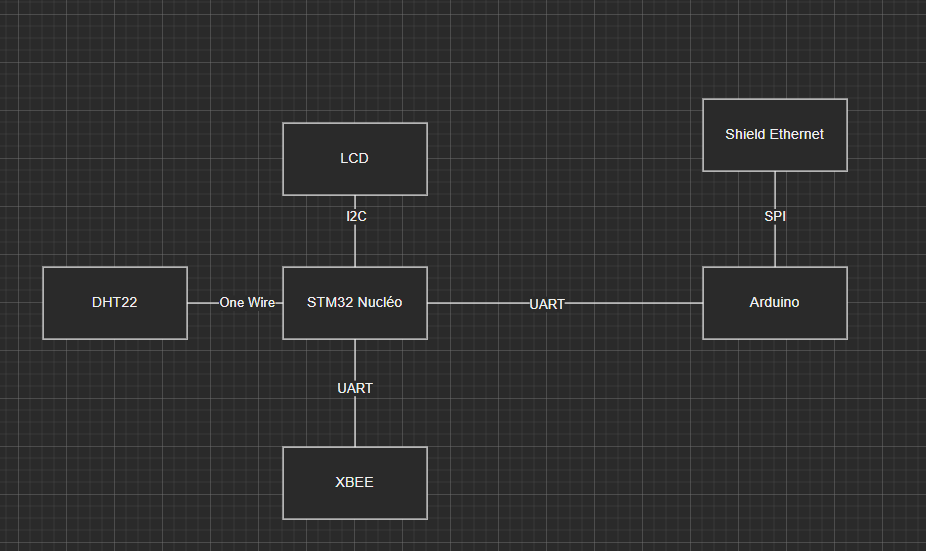
Description générée automatiquement

*Figure 5 : ICPS Arduino*

J’ai donc dû souder les fils nécessaires pour tenter d’établir la communication. Malheureusement, le manque de documentation, la complexité du portage et le temps ont fait que je n’ai pas réussi à adapter ce module. Les seuls exemples disponibles d’une adaptation réussite ont été effectués sur des cartes F401RE ou F103RE, j’avais à ma disposition une carte L152RE.

La solution à donc été de déporter le module sur une carte Arduino Uno et de relier UART cette Dernière à la carte Nucléo. La difficulté majeure a été de gérer l’envois périodique des données et l’accessibilité du serveur afin de ne pas perdre de données et d’éviter des temps morts lors d’une consultation via un navigateur WEB.

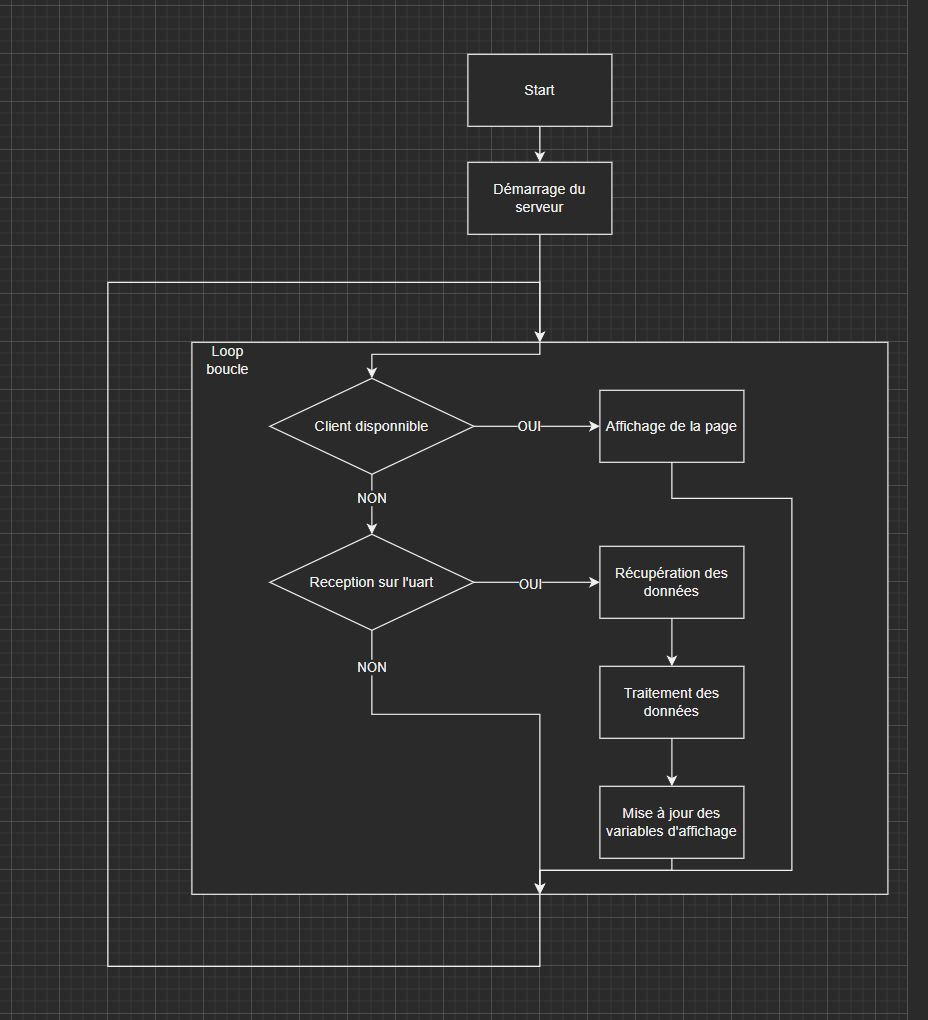
La figure ci-dessous présente donc le raccordement des différentes parties du module centrale :



*Figure 6 : Diagramme central*

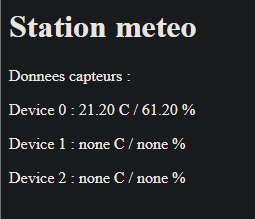
## Structure du programme Arduino

La difficulté d’implémenter un serveur collectant des données en parallèle avec des cartes mono-cœur et à programmation séquentielle est qu’il faut optimiser les temps d’exécution des différentes taches afin de revenir le plus vite possible en attente si un client se connecte. Le programme suit donc la logique suivante :



*Figure 7 : Routine Arduino*

Après la mise à jour des variables d’affichage l’Arduino va renvoyer une date et un horaire au STM32 afin que ce dernier puisse transférer cette donnée à tous les différents capteurs pour qu’ils puissent se mettre à jour.



*Figure 8 : Page HTML avec les données*

En vue des capacités de l’Arduino ç générer une page web il n’est pas possible de gérer le CSS sur ce genre d’affichage mais simplement des informations de base comme le montre la figure ci-dessus, ce qui est largement suffisant pour avoir un aperçus des données souhaitées.

# Capteur