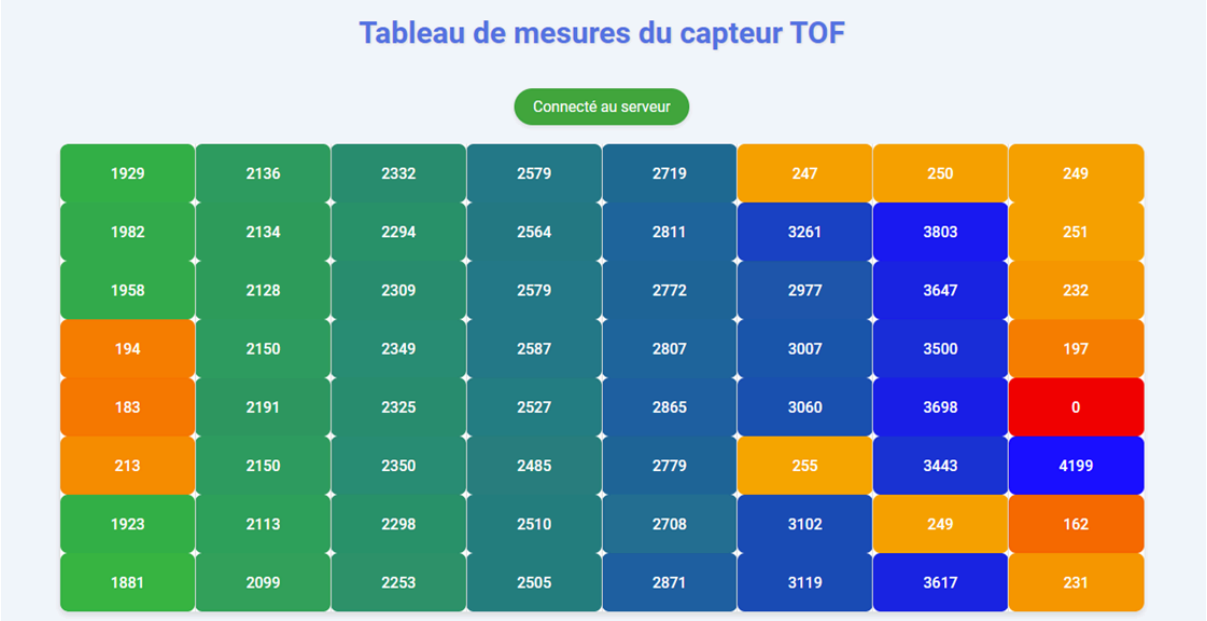
Projet
Capteur à temps de vol (TOF)



INSTITUT UNIVERSITAIRE DE TECHNOLOGIE
UNIVERSITÉ D'ANGERS

Etudiant : Sidney LEFRANCOIS / Emmanuel CAMIER

Groupe : 330

Encadrant: Philippe LUCIDARME

2024-2025

IUT Angers Cholet : 4 Bd de Lavoisier, 49000 Angers

Téléphone : 02 44 68 87 00

Décharge de responsabilités

Ce rapport présente le travail réalisé par un groupe d'étudiants dans le cadre d'un projet pédagogique. Les auteurs et l'Université d'Angers ne garantissent pas que l'information, les documents, la méthodologie et le matériel présentés dans ce document soient complets, conformes à l'état de l'art et exacts ni n'assurent en toutes circonstances la sécurité des biens, des personnes et des utilisateurs. Les auteurs et l'Université d'Angers ne seront pas tenus responsables des dommages éventuels qui pourraient résulter de l'utilisation du contenu du présent rapport.

Licence

Sommaire

Remerciements

Au nom de Sidney LEFRANCOIS et Emmanuel CAMIER, nous souhaitons exprimer notre profonde gratitude envers Monsieur Philippe Lucidarme, notre enseignant du projet, pour sa précieuse guidance, son soutien constant et les ressources qu'il nous a fournies tout au long de cette période. Sa patience et son expertise ont été d'une aide inestimable dans notre parcours.

Nous tenons également à exprimer notre reconnaissance envers nos camarades de classe qui ont apporté leur soutien et leurs idées pour affiner certains aspects de nos conceptions. Leur collaboration a été cruciale pour une meilleure détermination de certains designs, et nous sommes reconnaissants de leur contribution.

Ensemble, ces contributions ont enrichi notre expérience et ont rendu possible l'accomplissement de notre projet. Nous leur sommes infiniment reconnaissants pour leur soutien inestimable.

Introduction

Trop brut

Comment assurer une mise à jour dynamique des données sur une interface web sans nécessiter de rafraîchissement continu de la page ?

Cette problématique comporte 3 points importants qui permettront de répondre à cette question :

- **Connexion continue entre le serveur et le client**
- **Transmission des données en temps réel**
- **Optimisation de la performance et de la réactivité**

Ces 3 facteurs qui permettent de répondre à la question, sont donc 3 avantages que présente la technologie “Web Sockets”.

Une solution comme “WebSocket” établit une connexion persistante entre le serveur et le client, permettant un échange constant de données sans interruption.

Dès qu'une donnée est générée, elle est immédiatement envoyée à l'interface web, garantissant une mise à jour instantanée et fluide sans attendre un rafraîchissement manuel de la page.

La communication en temps réel via cette technologie réduit la latence, ce qui permet à l'interface de réagir rapidement aux changements, tout en offrant une expérience utilisateur sans délai.

Cahier des charges

Objectifs

Les objectifs du projet sur le capteur temps de vol étaient tout d'abord de réussir à communiquer avec le capteur et de récupérer ses données. Le défi ensuite était de créer un communication la plus rapide et sans interruption entre l'ESP32 et un Serveur python. Alors est venue la solution des Websocket, un procédé de communication très répandu et très efficace. Ce procédé permet de créer un lien entre deux systèmes sans besoin de requête à chaque transmission.

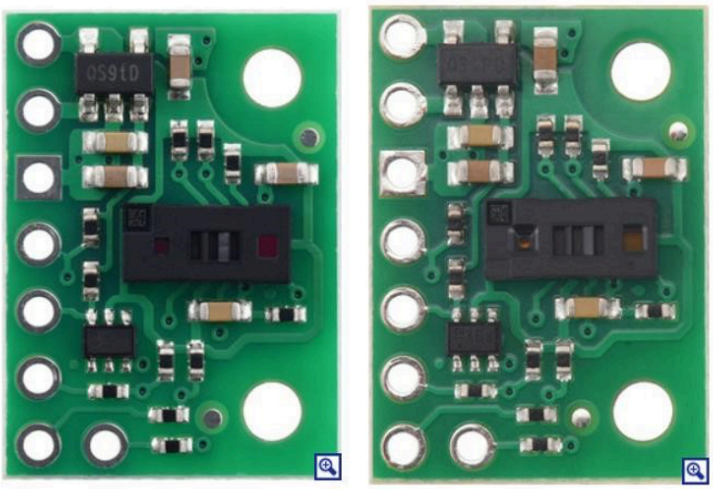
Livrables

- Capteur ToF
    - Lire les mesures du capteur
    - Afficher les donnés sous forme 8x8 dans les consoles
- ESP32:
    - Envoyer les données sur un port du serveur
- Web Socket:
    - Comprendre le fonctionnement des Web Socket
    - Mise en place d'une connection entre ESP32 et Serveur
- Serveur:
    - Recevoir les données provenant de l'ESP32
    - Traiter et afficher les données sur un page web

Choix des technologies

Capteur VL53L5CX

Présentation du Capteur



Index 1 : VL53L5CX Time-of-Flight 8×8-Zone Distance Sensor Carrier with Voltage Regulator, 400cm Max

Il s'agit d'une carte porteuse/dérivation pour le capteur de télémétrie laser VL53L5CX de ST, qui offre une télémétrie rapide et précise **jusqu'à 4 m grâce à** une interface numérique I²C. Il peut mesurer les distances absolues de plusieurs cibles simultanément dans plusieurs zones, fournissant suffisamment de données pour une carte de profondeur avec une résolution allant jusqu'à 4x4 ou 8×8. La carte comprend un régulateur linéaire de **3,3 V** et des décalages de niveau qui lui permettent de fonctionner sur une plage de tension d'entrée de **2,5 V à 5,5V**.

Le VL53L5CX se distingue des précédents capteurs de temps de vol de ST par sa sortie de télémétrie multizone : son champ de vision est divisé en un certain nombre de zones, configurables sous la forme d'une grille 4×4 ou 8×8, et le capteur fournit des lectures distinctes pour chaque zone (qui peut inclure plusieurs cibles par zone). Cela fait effectivement de l'VL53L5CX un lidar 3D de base, car au lieu de ne mesurer qu'une seule distance (lidar 1D), il peut fournir suffisamment de données pour générer une carte de profondeur à basse résolution de l'environnement dans son champ de vision.

Caractéristiques et spécifications

- Dimensions : 0,5 » × 0,7 » × 0,085 » (13 mm × 18 mm × 2 mm)
- Poids sans les broches d'en-tête : 0,5 g (0,02 oz)
- Tension de fonctionnement : 2,5 V à 5,5 V
- Courant d'alimentation : ~100 mA (moyenne typique pendant la plage active avec les paramètres par défaut). Le courant de crête peut atteindre 150 mA
- Portée maximale : 4 m (13 pi) avec une résolution de 1 mm
- Résolution : 1 mm de profondeur, 4×4 ou 8×8 zones de capteurs
- Portée minimale : 20 mm (0,8 po)
- Émetteur : VCSEL (laser à cavité verticale) invisible de classe 1 à 940 nm – sans danger pour les yeux
- Détecteur : réseau de réception SPAD (diode à avalanche à photon unique)
- Champ de vision complet typique : 63° en diagonale (45° horizontal/vertical)
- Format de sortie (I²C) : histogramme

Connexions

Au moins quatre connexions sont nécessaires pour utiliser la carte VL53L5CX

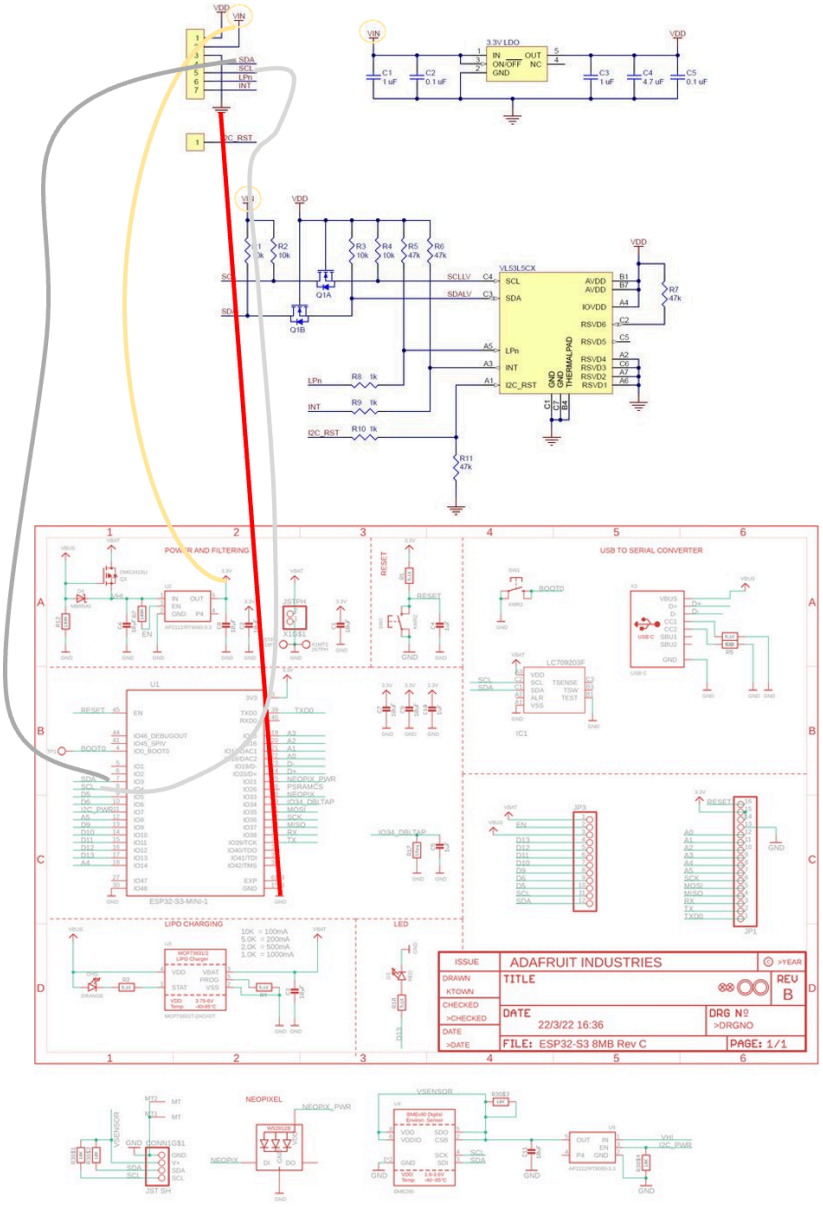
- **VIN**
- **GND**
- **SCL**
- **SDA**

La broche VIN doit être connectée à une source de **3,3 V** et **GND** doit être connectée à 0 volts. Un régulateur de tension linéaire embarqué convertit le VIN en une alimentation de 3,3 V, accessible via la broche VDD, pour le circuit intégré VL53L5CX. Des tensions d'alimentation comprises entre 2,5 V et 3,6 V peuvent également être connectées au VDD (avec le VIN laissé déconnecter) pour contourner le régulateur et alimenter directement la carte.

Brochage

ÉPINGLER	Description
VDD	**Sortie** 3,3 V régulée. Jusqu'à environ 100 mA sont disponibles pour alimenter les composants externes. (Si vous souhaitez contourner le régulateur interne, vous pouvez utiliser cette broche comme entrée pour les tensions comprises entre 2,5 V et 3,6 V avec le VIN déconnecté.)
NIV (NIV)	Il s'agit de la connexion principale de l'alimentation de 3,3 V à 5,5 V. Les décaleurs de niveau SCL et SDA tirent les lignes I²C haut à ce niveau.
GND	La connexion à la terre (0 V) pour votre alimentation électrique. Votre source de contrôle I²C doit également partager un terrain commun avec cette carte.
SDA	Ligne de données I²C décalée par niveau : HIGH est VIN, LOW est 0 V
SCL (Anglais seulement)	Ligne d'horloge I²C décalée par niveau : HIGH est VIN, LOW est 0 V
Indicateurs clés de performance (KPI	Cette broche est une entrée de désactivation active-faible I²C ; la carte le tire vers le haut sur VDD pour activer la communication I²C par défaut. L'activation de cette broche basse désactive la communication I²C (généralement utilisée dans le cadre d'un processus de modification des adresses I²C). *Cette entrée n'est pas décalée de niveau.*
INT	Sortie d'interruption programmable (niveau logique VDD). *Cette sortie n'est pas décalée par niveau.*
I2C_RST	Cette broche est une entrée de réinitialisation I²C active-élevée ; la carte le tire vers le bas à GND par défaut. L'enfoncement de cette broche vers le haut réinitialise l'interface I²C (mais pas l'ensemble du capteur). *Cette entrée n'est pas décalée de niveau.*

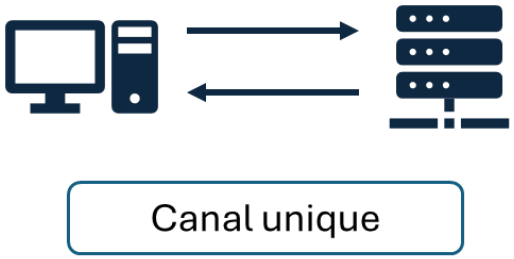
Index 2 : DataSheet du Capteur VL53LCX



WebSockets

Qu’est ce que les WebSockets ?

Les WebSockets sont une technologie de communication dite, bidirectionnelle (ou full-duplex), entre un client et un serveur via un canal unique et persistant. (Index 1)



Index 3 : Représentation d’un canal bidirectionnel

Contrairement aux requêtes HTTP classiques, où le client doit envoyer une requête pour obtenir des données, les WebSockets permettent au serveur d'envoyer des données au client sans que celui-ci ne demande constamment.

D’un point de vue imagé, c’est comme si, on construisait un tunnel entre un client et un serveur, qu’une seule fois, et qu’on envoyait des données l’un vers l’autre sans arrêt, sans interruption.

Fonctionnement

Une fois la connexion établie, le client et le serveur peuvent échanger des messages en temps réel. Cela se fait sur une seule connexion TCP (Transmission Control Protocol), ce qui réduit la surcharge par rapport à plusieurs connexions HTTP.

Index 4: Fonctionnement TCP

Un système ouvre une “socket”(point d'accès à une connexion TCP) et se met en attente passive de demandes de connexion d'un autre système. Ce fonctionnement est communément appelé *ouverture passive*, et est utilisé par le côté *serveur* de la connexion. Le côté *client* de la connexion effectue une *ouverture active* en 3 temps :

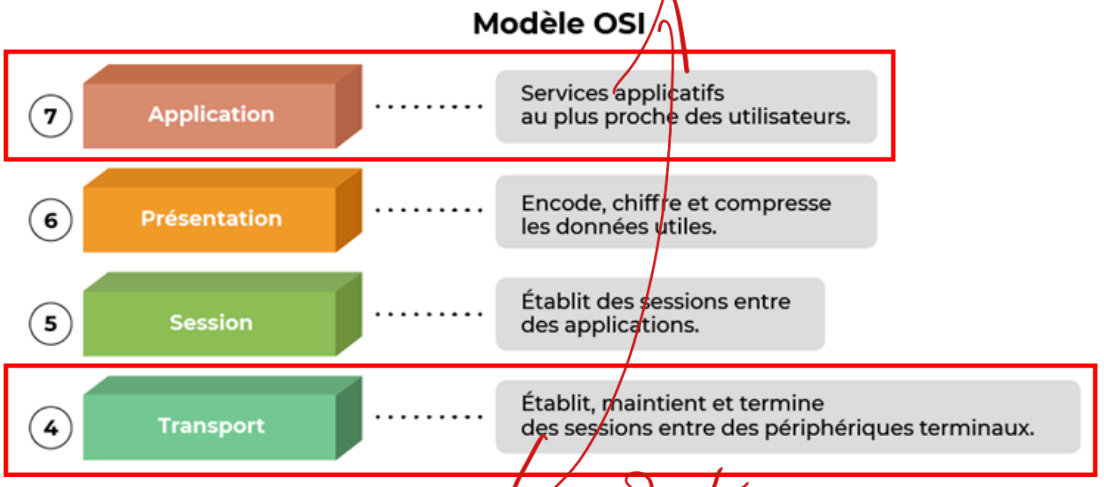
- Le client envoie un segment SYN au serveur,
- Le serveur lui répond par un segment SYN/ACK,
- Le client confirme par un segment ACK.

Durant cet échange initial, les numéros de séquence des deux parties sont synchronisés : (Index 2)

- Le client utilise son numéro de séquence initial dans le champ "Numéro de séquence" du segment SYN (x par exemple),
- Le serveur utilise son numéro de séquence initial dans le champ "Numéro de séquence" du segment SYN/ACK (y par exemple) et ajoute le numéro de séquence du client plus un (x+1) dans le champ "Numéro d'acquittement" du segment,
- Le client confirme en envoyant un ACK avec un numéro de séquence augmenté de un (x+1) et un numéro d'acquittement correspondant au numéro de séquence du serveur plus un (y+1).

Modèle OSI

Le Protocole WebSockets, se situe dans la partie 7 - Applications - du modèle OSI (une norme de communication de tous les systèmes informatiques en réseau).



Index 5 : Modèle OSI

Les WebSockets utilisent la couche 4 - Transport, pour la transmission de données, et donc se servent du protocole TCP pour cela.

Processus de Connexion

Le processus d’établissement d’une connexion WebSocket commence par une requête HTTP standard. Le client envoie une requête HTTP avec des en-têtes spécifiques pour demander à passer à une connexion WebSocket.

Index 6 : Partie Client

Si le serveur prend en charge les WebSockets, il répond avec un code d'état HTTP et des en-têtes similaires confirmant la mise à niveau de la connexion.

Index 7 : Partie Serveur

Après ce premier échange, la connexion est établie, donc à ce stade le client et le serveur peuvent échanger des messages en temps réel sans avoir besoin d’envoyer des requêtes HTTP pour chaque message.

Les WebSockets s'appuient sur HTTP pour établir la connexion initiale, mais une fois celle-ci établie, ils utilisent un protocole différent qui permet une communication en temps réel et bidirectionnelle. Cette interaction initiale avec HTTP est ce qui rend les WebSockets faciles à intégrer dans des applications web existantes, tout en offrant des capacités de communication avancées.

Avantages des WebSockets

- **Communication en temps réel** : Les WebSockets permettent une interaction instantanée, ce qui est idéal pour les applications nécessitant une mise à jour en temps réel, comme la transmission de données de capteurs.

- **Économie de bande passante** : En utilisant un seul canal pour les communications, les WebSockets réduisent la quantité de données envoyées et reçues, ce qui peut être crucial pour des appareils à ressources limitées comme l'ESP32.

- **Moins de latence** : La latence est généralement inférieure à celle des requêtes HTTP, car il n'y a pas besoin d'établir une nouvelle connexion pour chaque échange de données.

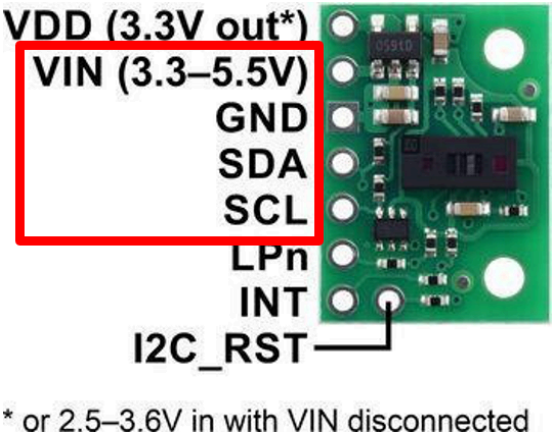
Réalisations

Schéma matériel

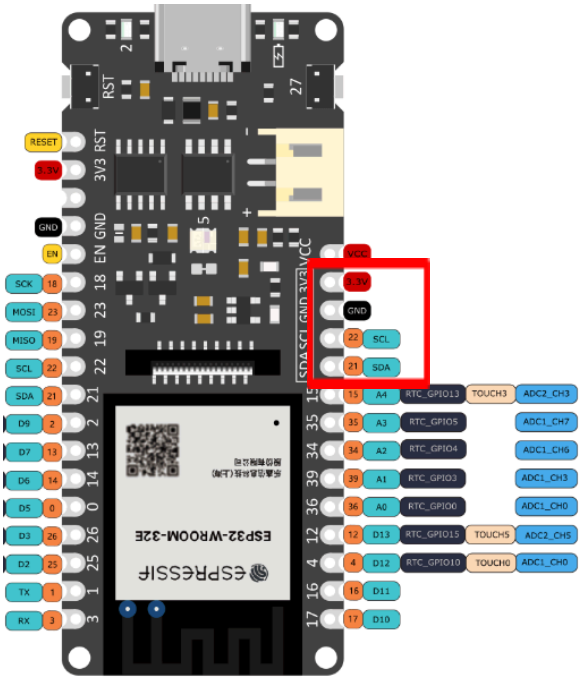
Le composant VL53L5CX est un composant qui nécessite une alimentation et une communication avec une tension entre 3,3V et 5,5V. Il est alimenté grâce à la broche VIN et communique à l'aide des ports I2C (SDA SCL).

L'ESP32 lui, alimenté par un câble USB C peut fournir une tension de 5,5V et de 3,3V au capteur, mais ne peut uniquement communiquer en 3,3V.

Pour effectuer une communication il faut brancher les broches SDA (data) et SCL (clock) du capteur VL53L5CX aux broches appropriées de l'ESP-32 FIREBEETLE 2 (SDA et SCL). Pour la connexion matérielle, seulement 4 branchements sont nécessaires.

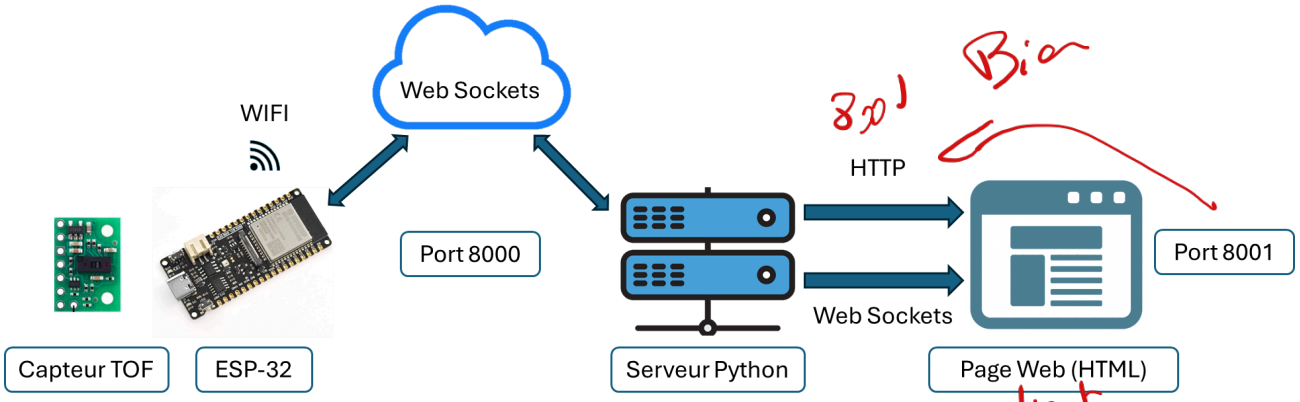


Index 8 : Pins du Capteur



Index 9 : Pins de l'Esp-32 Firebeetle 2

Schéma générale



Index 10 : Schéma Connexion vers Serveur

Notre architecture de projet est représentée en Index X. Il illustre notre point de départ jusqu'à la fin.

Pour commencer, on vient récupérer les valeurs du capteur TOF, donc les valeurs distances qu'il faudra afficher sous forme de matrice. Pour cela, on programme sur l'ESP32, une partie qui récupère ces valeurs du capteur, ensuite on les affiche sur le terminal sous forme de matrice 8x8. Puis ces valeurs, il faut les envoyer vers le serveur créé.

Pour cela on utilise la technologie WebSocket expliquée précédemment, cela va permettre d'envoyer ces données sans interruption de manière persistante.
Le serveur reçoit ces données, et les envoie donc via les WebSockets vers la page web (HTML) pour les afficher de manière dynamique sans "rafraichissement" de la page.

Cette architecture est en 2 parties distinctes. Il y a 2 ports différents, le port **8000**, est lui dédié à l'ESP-32, tandis que le port **8001**, lui, est pour tous les "clients" web.

Nous avons opté pour cette architecture, car cela simplifie le code, surtout côté serveur. Pour faire simple, on voulait envoyer des données vers le serveur et la page Web, sans recevoir les données sur notre ESP-32. Autrement dit, on envoie des données et on les récupère sur le terminal de l'ESP-32, car le serveur aurait renvoyé à tout ceux connecté aux serveurs les données, ce qui n'est pas l'objectif même si cela marche.
Pour conclure cela permet de séparer, de clarifier, notre code, mais aussi l'envoie de données. L'autre solution envisagée, était que chaque système voulant se connecter au serveur, devait "s'identifier", par exemple: ESP-32", et " Clients Web ", ce qui dans l'ensemble du projet semblait moins simple, que la solution gardée et expliquée dans cette partie.

Esp-32

Le code est divisé en plusieurs parties, allant de l'initialisation jusqu'à la gestion en temps réel des données du capteur et leur envoi au serveur. Voici une description détaillée de chaque section.

Partie 1 : Importation des Bibliothèques

- **WiFi.h** et **WiFiMulti.h** : Gèrent la connexion WiFi.
- **WebSocketsClient.h** : Gère les connexions WebSocket pour envoyer des données au serveur.
- **Wire.h** : Permet la communication I2C avec le capteur.
- **SparkFun_VL53L5CX_Library** : Bibliothèque pour utiliser le capteur de distance ToF (VL53L5CX).

Partie 2 : Déclaration des Variables

- **WiFiMulti** : Gestionnaire pour les réseaux WiFi.
- **webSocket** : Objet permettant de gérer la communication via WebSocket.
- **myImager** : Instance pour interagir avec le capteur VL53L5CX.
- **sendContinuous** : Variable pour activer/désactiver l'envoi en continu des données.
- **buttonPressed** : Gère l’état du bouton pour éviter les détections multiples.

Partie 3 : Fonction setup()

```
void setup() {
    USE_SERIAL.begin(115200);
    USE_SERIAL.println();

    // Connexion WiFi
    WiFiMulti.addAP(ssid, password);
    while (WiFiMulti.run() != WL_CONNECTED) {
        delay(100);
    }

    // Configuration WebSocket
    webSocket.begin("192.168.10.64", 8000, "/");
    webSocket.setReconnectInterval(2000);

    // Configuration du bouton et de l'I2C
    pinMode(27, INPUT_PULLUP);
    Wire.begin();
    Wire.setClock(400000);

    // Initialisation du capteur
    USE_SERIAL.println("Initializing sensor board. This can take up to 10s. Please wait.");
    if (myImager.begin() == false) {
        USE_SERIAL.println(F("Sensor not found - check your wiring. Freezing"));
        while (1);
    }

    myImager.setResolution(8 * 8);
    myImager.setRangingFrequency(15);
    myImager.startRanging();
}
```

Index 11 : Code Esp-32 | Partie Setup()

Initialisation WiFi :

- Le module ESP32 se connecte au réseau WiFi en utilisant le SSID et le mot de passe.
- La boucle while assure une connexion stable avant de continuer.

Configuration WebSocket :

- Initialise une connexion WebSocket au serveur sur l'adresse 192.168.10.64 et le port 8000.

Initialisation du capteur VL53L5CX :

- Configuration de la résolution en 8x8 pixels (grille).
- Réglage de la fréquence de mesure à 15 Hz.

Partie 4 : Fonction loop()

```
void loop() {
    webSocket.loop();

    // Lecture de l'état du bouton
    if (digitalRead(27) == LOW) {
        if (!buttonPressed) { // Détection du front descendant
            sendContinuous = !sendContinuous; // Basculer l'état
            buttonPressed = true; // Marquer comme traité
        }
    } else {
        buttonPressed = false; // Réinitialiser pour le prochain appui
    }

    // Si l'envoi en continu est activé
    if (sendContinuous) {
        if (myImager.isDataReady()) {
            if (myImager.getRangingData(&measurementData)) {
                String message = "";
                int imageWidth = sqrt(myImager.getResolution());

                for (int y = 0; y <= imageWidth * (imageWidth - 1); y += imageWidth) {
                    for (int x = imageWidth - 1; x >= 0; x--) {
                        message += String(measurementData.distance_mm[x + y]) + "\t";
                    }
                    message += "\n";
                }

                //USE_SERIAL.println(message);
                webSocket.sendTXT(message);
            }
        }
        delay(1); // Ajustez la fréquence d'envoi si nécessaire
    }
}
```

Index 12 : Code Esp-32 | Partie loop()

Gestion du bouton :

- Lit l'état de la broche connectée au bouton.
- Active ou désactive l'envoi en continu lors d'un appui détecté.

Capture des données du capteur :

- Vérifie si les données sont prêtes via " isDataReady() "
- Récupère les distances mesurées et les organise en matrice.
- Formate les données en chaîne de caractères pour l'envoi.

Envoi des données via WebSocket :

- Envoie la matrice formatée au serveur WebSocket.

Serveur

Pour le projet, nous avons développé un serveur WebSocket en Python en utilisant la bibliothèque websockets. Notre serveur joue un rôle essentiel dans l'établissement de la communication en temps réel entre le capteur et les clients, généralement des navigateurs web. Pour faciliter la gestion des données, nous avons choisi de travailler avec deux ports distincts : nous utilisons le port 8000 pour recevoir les données du capteur et de l'ESP32 et le port 8001 pour les transmettre aux clients co.

Fonctionnement du serveur:

Le serveur est conçu pour supporter plusieurs connexions simultanées. Il utilise une architecture asynchrone. Il repose sur deux gestionnaires principaux :

- Le premier gestionnaire est dédié au port 8000. Il reçoit les données envoyées par le capteur. Il les affiche ensuite dans la console pour validation et les prépare à être diffusées.
- Le deuxième gestionnaire, qui s'exécute sur le port 8001, prend en charge les connexions aux navigateurs. Ce deuxième gestionnaire a pour mission de transmettre les données reçues en temps réel aux différents clients connectés.

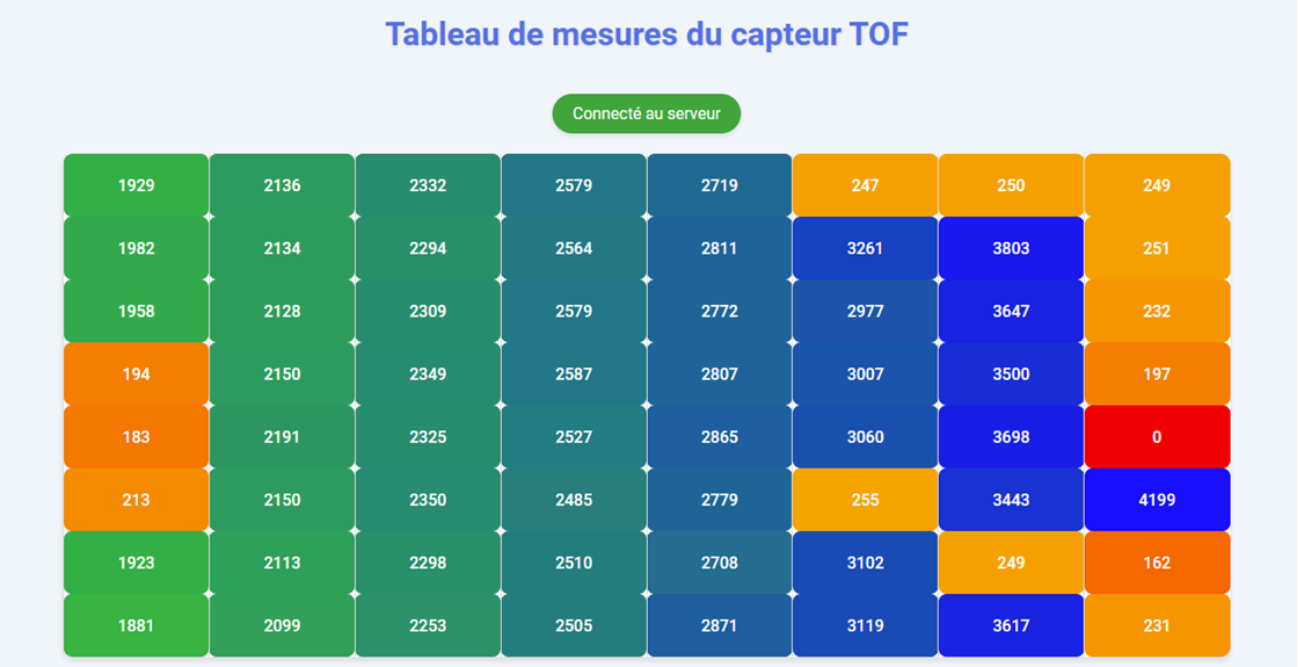
Diffusion des données:

Nous avons intégré dans notre solution un système de diffusion qui va diffuser les données que nous recevons sur le port 8000 à tous les clients connectés au port 8001. Une liste de clients actifs est maintenue pour garder la diffusion en temps réel puisque différents clients peuvent se connecter à des moments différents. En cas de déconnexion d'un client, le serveur continue son fonctionnement normal, c'est-à-dire continuer à diffuser sans interruption aux autres utilisateurs.

Nous avons choisi de séparer la réception et la diffusion des données sur deux ports pour clarifier la gestion du code et pour ne pas renvoyer les données au capteur. En exploitant la programmation asynchrone, de nombreuses connexions peuvent être gérées efficacement en parallèle par le serveur, cela garantit une communication rapide, fiable et fluide.

Ce serveur constitue donc l'une des clés de notre projet puisqu'il va nous permettre de mettre en place une interaction en temps réel entre le capteur et les utilisateurs, tout en assurant une transmission efficace et stable.

Sur ce serveur, est également hébergée une page web permettant d'afficher les données recueillies par le capteur. Nous avons conçu cette page web pour afficher une matrice 8x8 qui se met à jour en temps réel grâce à une connexion WebSocket. Nous avons utilisé une grille CSS pour organiser les cellules, chaque cellule représentant une valeur qui change de couleur en fonction de son contenu. Nous avons choisi cette structure pour offrir une visualisation claire et dynamique des données reçues.
Lorsqu'une matrice est envoyée par le serveur, elle est reçue par la page web via WebSocket. Nous avons programmé une fonction pour analyser ces données et mettre à jour la matrice affichée, ajustant les couleurs de fond des cellules selon la valeur reçue. Ainsi, la page réagit instantanément aux nouvelles informations transmises par le serveur, ce qui permet une interaction en temps réel avec les données du capteur.



Conclusion et Perspectives

Conclusion

En conclusion, le projet aboutit à une intégration réussie d’un capteur Time-of-Flight (TOF) avec un serveur, tout en exploitant la technologie WebSocket. Cette démarche a pleinement répondu aux exigences spécifiées dans le cahier des charges.

Nous avons observé que la rapidité des mesures fournies par le capteur, ainsi que sa fiabilité dans la collecte des données, ont permis d'envoyer des informations vers notre page web à une vitesse à la fois correcte et fiable. Cela constitue un atout majeur pour les applications nécessitant des données en temps réel.

Bien que la technologie des WebSockets puisse sembler complexe au premier abord, notamment lorsqu'elle est appliquée à un microcontrôleur tel que l’ESP-32, les avantages qu'elle offre sont indéniables. En effet, cette technologie a permis d'assurer un suivi constant et fluide des mesures, sans interruption. Cela témoigne de la fiabilité de la technologie WebSocket dans des environnements exigeants.

Perspectives

Pour améliorer davantage ce projet, une perspective intéressante serait d’implanter la reconnaissance de gestes, cela permettrait une utilisation de ces données afin de l’associer à un produit, système utile pour le quotidien, tel que le défilement des pages webs, sur un écran associé à ce produit.

Une autre perspective serait d’améliorer la page web, spécifiquement la partie visuel, donc HTML/CSS. Avec un visuel 3D sur la lecture des données. Enfin rendre ce serveur accessible partout, car ce serveur n’est qu’une extension trouvée, et non créée.

Résumé

Français

Ce projet consiste à développer un système de mesure basé sur un capteur TOF (Time of Flight). Les données collectées sont transformées en matrice afin d'être transmises à un serveur via un protocole de communication optimisé. Ces informations sont ensuite retransmises sur une page web interactive pour permettre une visualisation en temps réel. Ce projet combine des compétences en traitement de données, communication réseau et développement web, en mettant l’accent sur la précision et l’efficacité.

Anglais

This project aims to develop a measurement system based on a TOF (Time of Flight) sensor. The collected data is transformed into a matrix format to be transmitted to a server using an optimized communication protocol. The information is then retransmitted to an interactive web page to enable real-time visualization. This project combines data processing, network communication, and web development skills, with a focus on precision and efficiency.