



Mémoire de fin d'études en vue de l'obtention du
diplôme d'**Ingénieur d'Etat**

Filière

Electronique

Spécialité

Systèmes Embarqués

-Thème-

Développement d'un Système Embarqué dans un Tableau de Bord pour Scooter

Réalisé par:

BOUCHIBANE Amine

Soutenu le 25/06/2025

Composition du Jury :

NOM et Prénom	Établissement	Qualité
Mme. BOUCHAMA Samira	ENSTA	Présidente
Mme. CHOUAF Seloua	ENSTA	Encadrante
M. MAOUCHE Idris	FIBONOVA	Co-Encadrant
Mme. BELHACHAT Faiza	ENSTA	Examinatrice
M. BENZAOUI Mohamed Lassaad	USTHB	Examineur

Année Universitaire: 2024/2025

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

*Ce travail est humblement dédié à la mémoire de ma sœur
qui a toujours cru en moi.*

*Je dédie ce travail également à mes parents pour leurs
sacrifices et leur soutien sans faille. Leur confiance et leurs
prières m'ont accompagné à chaque étape, et c'est grâce à
eux que j'ai pu atteindre ce point dans ma vie.*

*À mon frère et mes sœurs, merci pour vos encouragements
constants et votre compréhension durant les périodes intenses
de ce travail. Votre soutien était d'une grande aide pour moi.
Je souhaite également exprimer ma profonde gratitude à tous
mes amis, pour leur accompagnement et leur encouragements
tout au long de ce parcours. Leur présence m'a offert la force
à dépasser les moments les plus exigeants.*

*Ce travail représente l'aboutissement de plusieurs années
d'études et l'achèvement de mon parcours académique. Il
symbolise la somme des connaissances acquises et la
transition vers une nouvelle étape de ma vie professionnelle.*

Amine

Remerciements

Je tiens à remercier toute personne m'ayant aidé durant cette période de réalisation de ce projet.

Tout d'abord, je tiens à remercier mon encadrante au niveau de l'école **Mme CHOUAF Seloua**, enseignante à l'ENSTA pour ses orientations méthodologiques, ses remarques constructives et son accompagnement tout au long de ce travail.

Je tiens également à remercier mon deuxième encadrant **M. MAOUCHE Idris**, ingénieur chez FIBONOVA, pour la confiance qu'il m'a accordée, sa disponibilité et les moyens qu'il a mis à ma disposition tout au long de mon stage.

Je remercie aussi l'équipe de l'entreprise FIBONOVA pour son accueil chaleureux, son soutien et ses conseils qui ont été d'une grande aide pour moi pour m'améliorer et atteindre des résultats satisfaisants pour ce projet.

Résumé

Ce projet a pour objectif le développement d'un système embarqué intégré à un tableau de bord pour scooter. La solution établit une communication bidirectionnelle avec un smartphone via Bluetooth pour l'échange de données, tout en proposant une interface utilisateur moderne sur un écran tactile.

Mots clés: Tableau de bord, Bluetooth , échange de données, écran tactile

Abstract

This project aims to develop an embedded system integrated into a scooter dashboard. The solution establishes a bidirectional communication with a smartphone via Bluetooth for data exchange, while also offering a modern user interface on a touchscreen.

Keywords: Dashboard, Bluetooth , data exchange, touchscreen

ملخص

يهدف هذا المشروع إلى تطوير نظام مدمج في لوحة قيادة لدراجة نارية. هذا النظام يقيم اتصالاً مع هاتف ذكي عبر تقنية البلوتوث لتبادل البيانات، مع توفير واجهة مستخدم عصرية على شاشة تعمل باللمس.

كلمات مفتاحية : لوحة القيادة, البلوتوث, تبادل البيانات, شاشة باللمس

Table de matières

Dédicace	I
Remerciements	II
Résumé	III
Liste des Acronymes	VIII
Liste des Figures	X
Liste des Tableaux	XI
Introduction générale	1
Chapitre 1 Les tableaux de bord connectés	3
1.1 Introduction	3
1.2 Tableaux de bord des motocycles	3
1.2.1 Fonctionnalités classiques intégrées	3
1.2.2 Limites des tableaux de bord traditionnelles	4
1.2.3 Vers une solution connectée	4
1.2.4 Choix de la technologie de communication	5
1.3 Technologie Bluetooth	6
1.3.1 Bluetooth low energy	6
1.3.2 Différence entre Bluetooth classique et BLE	7

1.3.3	Concepts d'ATT, GATT, et GAP	7
1.3.4	Fonctionnement du BLE	8
1.3.5	Notions de périphérique et de central	10
1.3.6	Avantages et limitations du BLE	10
1.3.7	Motivation pour le choix du BLE	11
1.4	Les microcontrôleurs	12
1.4.1	Composants d'un microcontrôleur	12
1.4.2	Champs d'utilisation des microcontrôleurs	12
1.5	Les interfaces homme machine	14
1.6	Conclusion	14
Chapitre 2 Étude du système embarqué proposé		15
2.1	Introduction	15
2.2	Analyse des besoins	15
2.2.1	Identifier les fonctionnalités Bluetooth	15
2.2.2	Affichage des informations	17
2.2.3	Horloge en temps réel	18
2.2.4	Contraintes techniques et exigences	18
2.3	Constitution matérielle et logicielle	19
2.3.1	Composants matériels	19
2.3.2	Composants logiciels	20
2.4	Choix du microcontrôleur	20
2.4.1	ESP32-S3	21
2.4.2	Atouts de l'ESP32-S3	21

2.5	Considérations pour l'écran d'affichage	22
2.5.1	Types des écrans envisagés	22
2.5.2	Critères de choix de l'écran TFT	23
2.6	Environnement de développement pour l'ESP32	23
2.7	Logiciel de développement de l'interface graphique	24
2.8	Schéma global de la solution embarquée	25
2.9	Conclusion	26

Chapitre 3 Développement et test du tableau de bord con-

necté 27

3.1	Introduction	27
3.2	Développement de la connectivité BLE sur ESP32	27
3.2.1	Services et caractéristiques utilisés	28
3.2.2	Gestion des événements	29
3.3	Échange des données	30
3.3.1	Réception des données	30
3.3.2	Envoi des données	32
3.3.3	Gestion des caractères spéciaux	33
3.3.4	Affichage des images indicatrices de la direction	34
3.4	Horloge à temps réel	34
3.4.1	Mode veille	34
3.4.2	Horloge RTC	35
3.4.3	Développement de l'horloge	35
3.4.4	Réglage de l'horloge	37

3.5	Alimentation du tableau de bord	37
3.6	Notion de table de partition	38
3.7	Développement de l'interface utilisateur	39
3.7.1	Interface d'accueil	40
3.7.2	Interface principale	41
3.7.3	Interface de réglage	42
3.8	Test et validation	43
3.8.1	Test en utilisant NRF Connect	43
3.8.2	Test avec une application personnalisée	44
3.8.3	Test de l'horloge	46
3.8.4	Test d'intégration dans un scooter	47
3.9	Conclusion	47
Conclusion générale		49
Références bibliographiques		XII
Annexes		XV

Liste des Acronymes

ASCII : American Standard Code for Information Interchange

ATT : ATtribute protocol

BLE : Bluetooth Low Energy

CAN : Controller Area Network

CPU : Central Processing Unit

DPSK : Differential Phase-Shift Keying

DQPSK : Differential Quadrature Phase-Shift Keying

EEZ : Enviro Experimental Zone

ESP : Espressif Systems

FHSS : Frequency-Hopping Spread Spectrum

GAP : Generic Access profile

GATT : Generic ATtribute profile

GFSK : Gaussian Frequency-Shift Keying

GPS : Global Positioning System

IDF : IoT Development Framework

IDE : Integrated Development Environment

IEEE : Institute of Electrical and Electronics Engineers

IP : Internet Protocol

I2C : Inter-Integrated Circuit

ISM : Industrial, Scientific, and Medical

LCD : Liquid Crystal Display

LED : Light Emitting Diode

LVGL : Light and Versatile Graphics Library

MTU : Maximum Transmission Unit

NRF : Nordic Radio Frequency

OLED : Organic Light-Emitting Diode

OSI : Open Systems Interconnection

OTA : Over-The-Air

PCB : Printed Circuit Board

PSRAM : Pseudo Static Random Access Memory

RAM : Random Access Memory

ROM : Read-Only Memory

RTC : Real-Time Clock

RTOS : Real-Time Operating System

SD : Secure Digital

SIG : Système d'Information Géographique

SoC : System on Chip

SPI : Serial Peripheral Interface

SRAM : Static Random Access Memory

STM : STMicroelectronics

TFT : Thin-Film Transistor

UART : Universal Asynchronous Receiver/Transmitter

USB : Universal Serial Bus

UTF-8 : Unicode Transformation Format - 8-bit

UUID : Universally Unique Identifier

WiFi : Wireless Fidelity

Liste des Figures

Figure. 1.1 Schéma du système global	5
Figure. 1.2 Concepts d'ATT, GATT, et GAP	8
Figure. 1.3 Fonctionnement du BLE	9
Figure. 1.4 Périphérique et central dans BLE	10
Figure. 1.5 Architecture d'un microcontrôleur[8]	13
Figure. 2.1 Transmission en mode FHSS	17
Figure. 2.2 Puce ESP32-S3-WROOM-1[14]	21
Figure. 2.3 ESP32-S3 avec écran 7 pouces tactile	23
Figure. 2.4 Fichiers issus de compilation UI	25
Figure. 2.5 Schéma global de la solution retenue	26
Figure. 3.1 Activation Bluetooth sur menu config	28
Figure. 3.2 Allocation statique	31
Figure. 3.3 Allocation dynamique de la mémoire	32
Figure. 3.4 Schéma et circuit du mode veille	36
Figure. 3.5 Alimentation de l'ESP32	38
Figure. 3.6 Déclaration de la table de partition dans menu config . . .	39
Figure. 3.7 Interface d'accueil en mode jour et nuit	40
Figure. 3.8 Affichage d'une demande de connexion	41

Figure. 3.9	Interface principale en mode nuit et jour	41
Figure. 3.10	Assigner la distance restante à la barre horizontale	42
Figure. 3.11	Interface de réglage en mode sombre et clair	42
Figure. 3.12	Tag de l'ESP et liste des services disponibles	44
Figure. 3.13	Requête de connexion	45
Figure. 3.14	Echange entre le smartphone et l'ESP	45
Figure. 3.15	Réception des appels	46
Figure. 3.16	RTC en mode veille	46
Figure. 3.17	Intégration dans un scooter	47

Liste des Tableaux

Tableau. 1.1	Comparaison entre Bluetooth Classic et BLE[2]	7
Tableau. 2.1	Caractéristiques de l'ESP32-S3[15]	22
Tableau. 3.1	Services et caractéristiques BLE pour le tableau de bord .	28
Tableau. 3.2	Normalisation des caractères spéciaux basée sur les codes UTF-8	33

Introduction générale

Depuis l'invention du motorcycle, le tableau de bord a été un point de connexion entre le conducteur et son véhicule. Initialement primitifs, ces dispositifs se limitaient à des fonctions élémentaires, telles que l'affichage de la vitesse via un simple compteur kilométrique mécanique. L'histoire du tableau de bord moto a connu une progression constante, inspirée par l'innovation technologique et l'évolution des attentes des conducteurs en ce qui concerne l'information, la sécurité et le confort.

À l'origine, le terme "tableau de bord" désignait une planche en bois ou en cuir placée à l'avant des calèches ou des premiers véhicules hippomobiles. Son rôle était de protéger le conducteur de la boue et des débris projetés par les sabots des chevaux. Avec l'apparition des premières automobiles, cette "planche" a été utilisée également pour la protection contre les projections et la chaleur du moteur. À partir des années 1930, les tableaux de bord ont commencé à s'enrichir en intégrant davantage d'instruments pour fournir des informations plus complètes au conducteur[1]. Cette évolution a continué avec l'intégration de l'électronique. Aujourd'hui, l'information ne se limite plus aux données du véhicule. L'Internet des Objets (IoT) a posé la nécessité d'une solution connectée et les conducteurs modernes attendent une intégration de leur smartphone et de leurs services numériques personnels dans l'environnement de conduite. Un tableau de bord moderne doit répondre à cette exigence en offrant une lecture enrichie des données du véhicule et une interaction avec le monde extérieur afin de transformer la conduite en une expérience plus intelligente, plus sûre et plus intégrée.

Le présent travail a pour but de concevoir et de développer une solution connectée et embarquée dans un tableau de bord pour motorcycle, basé sur un microcontrôleur capable d'établir une liaison avec un smartphone pour un échange de données bidirectionnel. L'objectif principal est de créer une interface homme-machine moderne et interactive, capable de fournir au conducteur

des informations essentielles sur son véhicule et des données issues de son smartphone. Ce mémoire est constitué des trois chapitres suivants:

Un premier chapitre qui introduit les tableaux de bord des motocycles et la nécessité d'une amélioration et intégration de l'IoT. Il présente aussi la technologie de communication à utiliser pour ce tableau de bord connecté et les critères de ce choix.

Un deuxième chapitre qui s'intéresse aux contraintes et exigences liées au système, aux fonctionnalités nécessaires, ainsi qu'au choix des composants matériels et logiciels pour ce projet et le critère des choix effectués.

Un dernier chapitre qui détaille les étapes de conception et de réalisation du tableau de bord connecté en se basant sur les composants choisis. Il inclue également le test de la solution finale pour valider ses performances et capacités.

Chapitre 1

Les tableaux de bord connectés

1.1 Introduction

Le tableau de bord d'un scooter, dans sa conception traditionnelle, est une interface d'informations limitée et peu intuitive; il affiche des données basiques comme la vitesse et le niveau de carburant. Suite au développement des technologies de communication, cette configuration ne répond plus pleinement aux attentes et aux besoins des conducteurs modernes. Ce contexte ouvre l'opportunité de développer une solution innovante d'intégration des fonctionnalités numériques avancées pour enrichir l'expérience de conduite.

1.2 Tableaux de bord des motocycles

Le tableau de bord du motocycle est un élément qui affiche les informations indispensables à la conduite. Il répond à trois grands objectifs généralement établis qui sont la surveillance des paramètres du véhicule comme la vitesse et le niveau de carburant, la sécurité grâce aux alertes signalées par des indicateurs et des témoins lumineux, et aussi le confort et l'assistance à la conduite, notamment sur les motos modernes qui intègrent des fonctionnalités avancées comme le GPS ou l'affichage des différents modes de conduite.

1.2.1 Fonctionnalités classiques intégrées

Bien que les tableaux de bord des scooters ont considérablement évolué avec la technologie, certaines fonctionnalités demeurent des standards inchangés et sont essentielles pour le conducteur. Ces éléments traditionnels incluent:

- Le compteur de vitesse, disponible en versions analogique ou numérique,

pour informer sur la vitesse instantanée.

- Le compteur kilométrique, qu'il soit total ou partiel, permet de suivre la distance parcourue.
- La jauge de carburant utilisées comme un indicateur vital du niveau restant.
- L'indicateur du régime moteur qui fournit des informations clés sur les performances du moteur.
- Une horloge est également intégrée pour la gestion du temps de trajet.
- Des voyants lumineux qui informent le conducteur sur des informations critiques telles que les clignotants, les feux de route, la pression d'huile et la température moteur.

1.2.2 Limites des tableaux de bord traditionnelles

Les tableaux de bord classiques ont plusieurs limites qui peuvent affecter l'expérience du conducteur. Leur lisibilité est souvent réduite surtout en conditions de conduite changeantes et ces systèmes souffrent d'un manque remarquable d'informations avancées, ne proposant ni l'analyse en temps réel de la consommation, ni des diagnostics détaillés essentiels pour l'entretien du véhicule. De plus, les anciens modèles des scooters ne disposent pas de liaisons modernes comme le Bluetooth ou le GPS intégré, chose qui limite les interactions. Enfin, ces tableaux de bord présentent une réduction de la réactivité, et une incapacité à donner des alertes personnalisées en cas de problème mécanique, ce qui peut influencer la sécurité et l'intervention rapide.

Pour cela, il serait intéressant de proposer une solution innovante pour ces limitations causées essentiellement par le manque des technologies modernes sur les tableaux de bord des motos.

1.2.3 Vers une solution connectée

La transition vers un tableau de bord connecté (figure 1.1) revient à plusieurs facteurs. Pour la sécurité, en affichant les informations essentielles directement sur le tableau de bord, on réduit la dépendance au téléphone et ainsi les risques d'accidents dus à la distraction. Un autre facteur est la nécessité d'afficher les

informations essentielles en temps réel et d'offrir une lecture rapide et intuitive. Aussi, et pour les professionnels, comme les livreurs, la connectivité avec leur smartphone optimise leur travail en leur offrant un accès direct aux commandes et aux trajets via le GPS intégré. La technologie numérique, notamment les écrans Thin Film Transistor (TFT) et Organic Light-Emitting Diode (OLED), qui sont largement utilisés dans plusieurs domaines, présente des avantages majeurs en ce qui concerne la clarté et la visibilité.

Pour cela, le développement d'un tableau de bord connecté, intégrant une connectivité smartphone et un microcontrôleur, est devenu une nécessité face aux limites des systèmes traditionnels. Cette approche permet d'améliorer l'expérience de conduite en offrant un accès en temps réel à des informations enrichies. Le smartphone sert d'interface utilisateur dynamique et de passerelle vers plusieurs données et services, tandis que le microcontrôleur assure la réception et le traitement des données de manière fiable. Cette approche crée un écosystème intelligent, améliorant non seulement la sécurité et le confort du conducteur, mais aussi l'efficacité et la personnalisation de la conduite.

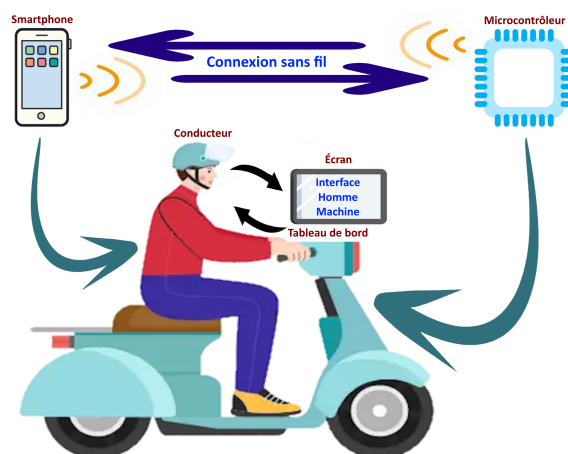


Figure 1.1: Schéma du système global

1.2.4 Choix de la technologie de communication

Il existe plusieurs technologies permettant de relier un tableau de bord de scooter à un smartphone, chacune possède des avantages et des inconvénients. Le choix de la technologie est important et repose sur les besoins spécifiques de l'application. Parmi ces technologies, on trouve le Wireless Fidelity (WiFi) qui offre un débit de données élevé avec une grande consommation énergétique. Un

autre choix est le Long Range (LoRa) qui est idéal pour les longues distances, mais son débit de données est extrêmement faible. Ces deux technologies sont incompatibles avec l’affichage dynamique des informations en temps réel nécessaires à un tableau de bord connecté.

Le Bluetooth, plus précisément le Bluetooth Low Energy (BLE) peut être considéré comme le choix le plus pertinent pour un tableau de bord connecté, grâce à ses multiples avantages.

1.3 Technologie Bluetooth

Le Bluetooth est une technologie de communication à courte distance qui opère dans la bande de fréquences radio Industriel, Scientifique, Médical (ISM) qui est autour de 2.4 GHz[2]. Il permet d’établir des liaisons non filaires directes et temporaires entre différents équipements et donc de remplacer celles filaires traditionnelles.

La technologie Bluetooth s’est imposée comme un standard mondial. Cette évolution est portée par le Bluetooth Special Interest Group (SIG), l’organisme officiel chargé de la supervision de la norme Bluetooth. Créé en 1998, il est formé d’un groupe d’ingénieurs de cinq entreprises qui sont ERICSSON, IBM, NOKIA, TOSHIBA, et INTEL. Aujourd’hui, le SIG comporte plus de 1800 membres actifs et se compose de plusieurs groupes de travail à chacun est donnée une fonction de développement[3]. Il existe deux catégories distinctes de Bluetooth qui sont le Bluetooth classique ou simplement appelé ”Bluetooth”, et le Bluetooth Low Energy (BLE), chacune optimisée pour des scénarios d’utilisation distincts.

1.3.1 Bluetooth low energy

Le BLE est une norme sans fil conçue pour les réseaux personnels à faible coût, nécessitant une faible consommation d’énergie et des débits de données réduits. Au niveau de la couche physique, le BLE fonctionne dans la bande ISM des 2,4 GHz. La bande passante assignée de 80 MHz est divisée en 40 canaux orthogonaux, avec des fréquences centrales espacées régulièrement de 2 MHz.

On distingue deux types de canaux d’annonce et de données. Les canaux d’annonce, qui sont au nombre de 3, sont utilisés pour la découverte de pé-

riphériques, l'établissement de connexions, et la diffusion de messages. En revanche, le deuxième type de canaux est représenté par les 37 canaux restants qui servent à échanger des données[4].

1.3.2 Différence entre Bluetooth classique et BLE

Le Bluetooth classique est la version originale et la plus ancienne de la technologie Bluetooth. Ce type de Bluetooth diffère du BLE comme indiqué dans le tableau 1.1, mais la principale différence entre eux réside dans leur optimisation et dans leurs cas d'utilisation.

Tableau 1.1: Comparaison entre Bluetooth Classic et BLE[2]

Caractéristiques	Bluetooth Classic	Bluetooth LE
Bande de fréquence	2.4GHz ISM (2.402 – 2.480 GHz)	2.4GHz ISM (2.402 – 2.480 GHz)
Canaux	79 canaux espacés de 1 MHz	40 canaux espacés de 2 MHz (3 publicité, 37 données)
Utilisation des canaux	FHSS (Frequency-Hopping Spread Spectrum)	FHSS (Frequency-Hopping Spread Spectrum)
Modulation	GFSK, $\pi/4$ DQPSK, 8DPSK	GFSK
Puissance d'émission	≤ 100 mW (+20 dBm)	≤ 100 mW (+20 dBm)
Topologie	Point à point	Point à point, Broadcast, Mesh

1.3.3 Concepts d'ATT, GATT, et GAP

Le BLE s'appuie sur une architecture bien définie afin de gérer une communication et un échange de données de manière efficace. Parmi les composants de cette architecture, trois concepts clés sont fondamentaux, qui sont l'ATT, le GATT et le GAP (Voir figure 1.2 sur la page suivante).

a. ATT

ATtribute protocol ou le protocole ATT est la base de la communication de données dans le BLE. Il définit la structure de base des informations échangées en les organisant sous forme d'attributs. Chaque attribut est une paire clé/valeur unique, identifiée par un Universally Unique Identifier (UUID), et possède des permissions spécifiques (lecture, écriture). L'ATT est le protocole de transport qui gère l'envoi et la réception de ces attributs entre les appareils connectés.

b. GATT

Au-dessus de l'ATT se trouve le Generic ATTribute profile (GATT). Le GATT fournit une structure logique pour organiser les données définies par l'ATT. Il définit comment les données sont regroupées en services et en caractéristiques. Un service est un ensemble d'attributs qui représente une fonctionnalité spécifique (par exemple, un service de batterie), tandis qu'une caractéristique est une information individuelle exposée par ce service (comme le niveau de charge de la batterie). Le GATT utilise l'ATT pour échanger ces services et caractéristiques entre les appareils.

c. GAP

Le Generic Access Profile est responsable de la gestion de la connexion au niveau le plus bas. C'est lui qui définit comment les appareils BLE se rendent détectables, comment ils se découvrent mutuellement, comment ils initient et établissent une connexion. Le GAP gère les étapes initiales pour que les appareils puissent se trouver et établir une liaison sécurisée avant l'échange de données via GATT et ATT.

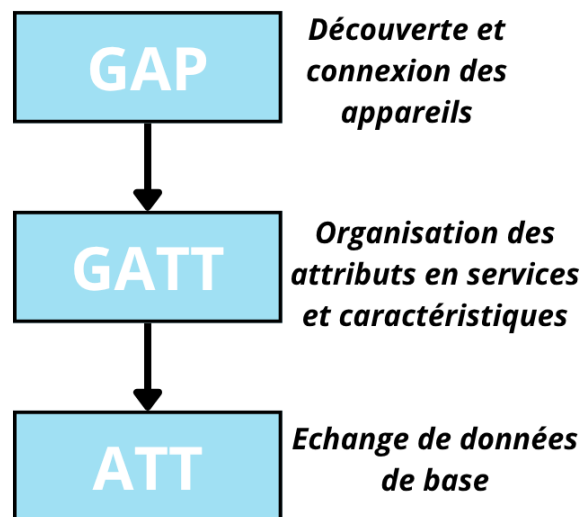


Figure 1.2: Concepts d'ATT, GATT, et GAP

1.3.4 Fonctionnement du BLE

Le BLE est une technologie de communication non filaire optimisée pour une très faible consommation d'énergie, ce qui est idéal pour les systèmes embar-

qués. Contrairement au Bluetooth Classique, le BLE n'est pas conçu pour le transfert de fichiers volumineux mais plutôt pour l'échange de petites quantités de données de manière discontinue.



Figure 1.3: Fonctionnement du BLE

Comme illustré sur la figure 1.3, son fonctionnement repose sur quelques concepts qui sont:

a. Diffusion d'annonce

Les appareils BLE commencent par diffuser des annonces (advertising) qui sont comme un petit message radio envoyé régulièrement. Ces annonces contiennent des informations de base comme le nom de l'appareil et les services qu'il propose. Un smartphone à proximité peut écouter ces annonces pour découvrir les appareils BLE disponibles sans avoir à établir une connexion complète. C'est un processus très économe en énergie.

b. Connexion

Lorsqu'un client détecte une annonce intéressante, il peut initier une connexion. Cette connexion est établie rapidement et de manière sécurisée. Si c'est la première fois que les deux appareils communiquent, un processus de "pairage" (ou appairage) peut être nécessaire. Cela implique l'échange de clés de sécurité pour garantir que seuls les appareils autorisés peuvent communiquer afin de renforcer la confidentialité des données.

c. Échange de données

Une fois la connexion établie, l'échange de données dans BLE est structuré selon le modèle GATT. Ce modèle organise les données en services et en caractéristiques. Le client peut lire des caractéristiques, écrire de nouvelles valeurs, ou s'abonner à des notifications pour être averti lorsque la valeur d'une caractéristique change (Read, Write, Notify).

d. Communication discrète

Contrairement au Bluetooth classique qui maintient souvent une connexion continue pour le streaming, le BLE est conçu pour des connexions courtes et intermittentes. Les appareils peuvent rapidement se connecter, échanger les données nécessaires, puis se déconnecter ou entrer en mode "sommeil" pour économiser de l'énergie. Cela rend le BLE particulièrement efficace pour des applications où les données sont échangées de manière irrégulière comme les mises à jour de capteurs ou les notifications[5].

1.3.5 Notions de périphérique et de central

Dans BLE, les notions de périphérique et de central sont importantes (voir figure 1.4). Un périphérique est un appareil qui annonce sa présence en envoyant des paquets de publicité (advertising packets) et accepte une connexion provenant d'un autre appareil BLE, appelé central. Tandis qu'un central est un appareil qui détecte et écoute des autres appareils BLE faisant de la publicité (advertising), il est également capable d'établir une connexion avec des périphériques BLE (généralement, plusieurs en même temps). À noter qu'un appareil peut être un périphérique dans une communication BLE et un central dans une autre[6].

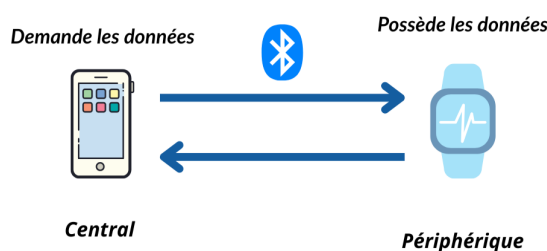


Figure 1.4: Périphérique et central dans BLE

1.3.6 Avantages et limitations du BLE

Comme toute technologie, le BLE présente à la fois des avantages distincts qui le rendent idéal pour plusieurs applications, mais aussi certaines limitations liées à sa nature "low energy" qui doivent être prises en compte lors de la conception de systèmes.

a. Avantages

Le BLE se distingue par sa faible consommation d'énergie. Au-delà de cet avantage énergétique, son adoption est facilitée par la disponibilité de la documentation et des spécifications accessibles gratuitement, ce qui simplifie le processus de développement. De plus, les modules et chipsets BLE sont généralement plus abordables que ceux d'autres technologies de communication sans fil comparables. Enfin, sa disponibilité dans la majorité des smartphones lui confère un grand avantage comparé à d'autres solutions [6].

b. Limitations

Malgré ses nombreux avantages, le BLE présente des limitations liées à sa conception optimisée pour la basse consommation. Son débit de données est fortement limité par les capacités du radio physique, ce qui le rend moins adapté aux applications nécessitant un streaming continu ou un grand volume de données. De plus, sa portée est relativement courte, influencée par la fréquence ISM utilisée, la performance et l'orientation de l'antenne, ce qui peut impacter la fiabilité de la connexion dans certains environnements. Enfin, le BLE requiert une passerelle (comme un smartphone ou un hub) car il ne supporte pas nativement la connectivité IP directe[6].

1.3.7 Motivation pour le choix du BLE

Le BLE est la technologie à utiliser pour un tableau de bord connecté afin d'établir une connectivité avec le smartphone en raison de son profil technique idéalement aligné avec les exigences du système. Les critères sur lesquels le BLE est utilisé sont:

- Consommation énergétique minimale pour préserver l'autonomie des appareils connectés (scooter et smartphone).
- Capacité à maintenir une connexion fiable et constante sur une distance suffisante.
- Débit suffisant pour afficher les données en temps réel.
- Compatibilité et intégration avec la majorité des smartphones et les sys-

tèmes d'exploitation (Android, iOS).

- Sécurité des communications en effectuant un chiffrement des échanges de données pour éviter les écoutes ou les accès non autorisés.

En se basant sur ses critères, le BLE est la technologie de communication à intégrer pour un tableau de bord connecté.

1.4 Les microcontrôleurs

Un microcontrôleur est un circuit intégré qui regroupe sur une seule puce les principaux composants nécessaires au fonctionnement d'un système embarqué. Il est conçu pour des tâches spécifiques et se distingue d'un microprocesseur par son intégration complète, ce qui le rend compact, économe en énergie et rentable[7].

1.4.1 Composants d'un microcontrôleur

Un microcontrôleur est composé de plusieurs éléments rassemblés sur une seule puce afin de compléter des tâches spécifiques. Comme illustré dans la figure 1.5, ces composants sont:

- Un processeur (CPU) pour l'exécution des instructions, avec un bus de données allant de 4 bits à 32 ou 64 bits.
- la mémoire vive (RAM) pour stocker les données de manière volatile, et la mémoire morte (ROM) pour stocker le programme.
- un oscillateur pour le cadencement. Il peut être utilisé pour les opérations liés au temps.
- différents bus de données, d'adresses, et d'adresses
- des périphériques, capables d'effectuer des tâches spécifiques comme les convertisseurs, les compteurs, les timers, etc.

1.4.2 Champs d'utilisation des microcontrôleurs

Les microcontrôleurs sont omniprésents dans la vie quotidienne. Leur polyvalence, leur faible coût et leur capacité à effectuer des tâches spécifiques de manière autonome les rendent indispensables dans une multitude de domaines.

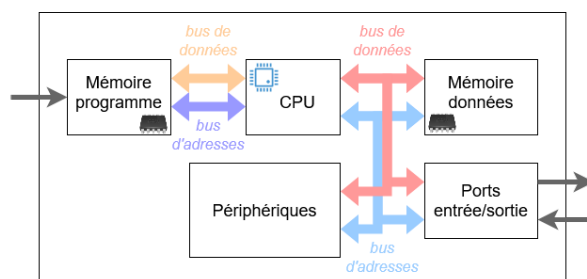


Figure 1.5: Architecture d'un microcontrôleur[8]

a. Électronique grand public et domotique

Ce champ est le plus visible. Les microcontrôleurs se retrouvent dans divers appareils variés tels que les télécommandes, les jouets, les appareils électroménagers, les systèmes de divertissement, et tous les dispositifs de la domotique (éclairage intelligent, thermostats connectés, serrures intelligentes, systèmes d'alarme).

b. Industrie et automatisation

Dans le secteur industriel, les microcontrôleurs sont utilisés dans l'automatisation et le contrôle des processus. On les trouve dans les systèmes de contrôle de machines, les équipements de production, les actionneurs, et les systèmes de surveillance. Ils contribuent à améliorer l'efficacité de la production, la qualité des produits. Des microcontrôleurs 8 bits sont souvent utilisés pour les capteurs simples et les actionneurs, tandis que des versions 32 bits gèrent des tâches de contrôle et de gestion plus complexes.

c. Objets connectés

L'explosion de l'IoT a propulsé les microcontrôleurs au premier plan. En profitant de leurs caractéristiques, ils sont les composants fondamentaux des objets connectés. Que ce soit pour des capteurs, des dispositifs de suivi, des appareils de surveillance à distance ou des systèmes domotiques avancés.

d. Autres domaines

Les microcontrôleurs sont présents dans d'autres champs comme la santé, les télécommunications, l'automobile, etc. Leur invention a marqué une avancée majeure dans le développement technologique actuel.

1.5 Les interfaces homme machine

Les Interfaces Homme-Machine (IHM), souvent désignées par User Interface (UI), représentent le point d'interaction entre l'utilisateur et un système. Leur conception vise à permettre aux humains de contrôler et d'échanger des informations avec divers dispositifs technologiques de manière efficace et intuitive.

Une bonne IHM doit garantir la pertinence et la clarté de l'information qu'elle transmet, tout en étant parfaitement compatible avec les capacités et les attentes de l'utilisateur qui cherche à intervenir au mieux selon ses propres objectifs et son approche. La conception d'une IHM est un domaine multidisciplinaire qui intègre des aspects de l'ergonomie, du design graphique et de l'ingénierie logicielle. Une IHM réussie prend en compte les capacités et les limites de l'utilisateur ainsi que les objectifs de la tâche pour créer une expérience fluide et satisfaisante.

1.6 Conclusion

En conclusion, ce chapitre a posé les bases nécessaires d'un tableau de bord connecté pour scooter, en explorant les généralités essentielles à sa compréhension. En proposant un tableau de bord communiquant via Bluetooth avec un smartphone Android, l'objectif est d'améliorer la sécurité et l'expérience de conduite grâce à l'intégration de fonctionnalités modernes.

Chapitre 2

Étude du système embarqué proposé

2.1 Introduction

Le chapitre précédent a donné une idée sur les tableaux de bord connectés, en précisant le besoin de développer une solution connectée et la technologie BLE qui sera intégrée. Pour la suite, nous allons nous pencher sur l'étude du système de ce tableau de bord connecté. Ce chapitre aura pour but d'approfondir les aspects techniques et fonctionnels nécessaires à la conception et à la réalisation de cette solution.

2.2 Analyse des besoins

L'analyse des besoins permet de garantir que le produit final réponde efficacement aux attentes des utilisateurs et aux objectifs du projet. Elle inclut toutes les spécifications et les variations potentielles du projet. Cette étape peut se résumer en quelques points cités ci-dessous:

2.2.1 Identifier les fonctionnalités Bluetooth

Vu que ce tableau de bord doit intégrer le Bluetooth pour la communication avec le smartphone, il est important de définir les fonctionnalités spécifiques de cette connectivité. Il s'agit de déterminer comment le Bluetooth sera utilisé pour répondre aux besoins du conducteur ainsi que le type de données qui seront échangées. Ces données peuvent être classées comme suit:

a. Nom de l'appareil

C'est le nom du smartphone connecté au tableau de bord. Il peut être le

nom du Bluetooth, modèle du téléphone, ou bien le fabricant. Ce nom permet de distinguer quel appareil demande de se connecter pour ne pas avoir une confusion lors de l'utilisation de plusieurs appareils.

b. Niveau de batterie

Cette données désigne le niveau de batterie réel du smartphone. Elle indique quand le conducteur doit recharger son téléphone.

c. Nom de l'appelant

Lorsqu'un appel est reçu, le smartphone envoie le nom de l'appelant ainsi que la durée de l'appel et d'autres détails. C'est très important de notifier le conducteur suite à un appel entrant.

d. Vitesse et kilométrage

Contrairement aux données précédentes, la vitesse et kilométrage sont envoyés par le tableau de bord vers le smartphone. Alors que ces données ne sont pas assez importantes mais ça permet d'établir un échange représentatif à partir du tableau de bord et d'autres informations plus importantes peuvent être envoyées en suivant le même principe.

e. Données de navigation

Ce sont les informations reliées à l'itinéraire choisi. Ces informations sont citées ci-dessous:

- La distance restante avant de changer de direction.
- Une instruction de navigation décrivant la position actuelle ou annonçant la prochaine direction à suivre.
- La distance restante avant l'arrivée à destination.
- L'heure d'arrivée estimée, se basant sur la vitesse du véhicule, la distance totale du trajet, et le trafic en temps réel. Cette information est susceptible de changer tout au long du trajet parcouru.
- Le temps restant avant l'arrivée, qui est calculé de manière dynamique par l'estimation de la différence entre l'heure d'arrivée et l'heure actuelle.

Il est toutefois essentiel de pouvoir activer et désactiver son module Bluetooth. Cette gestion de l'état permet d'économiser l'énergie lorsque la connectivité n'est pas nécessaire et de la réactiver rapidement quand le besoin se fait sentir. La diffusion est, elle aussi, une fonctionnalité cruciale pour que le système se fasse découvrir par le smartphone; en envoyant des paquets de diffusion à des intervalles de temps réguliers en utilisant le Frequency Hopping Spread Spectrum (FHSS) comme illustré dans la figure 2.1. Cette méthode consiste à diffuser l'information sur l'ensemble du spectre de fréquences disponible en changeant rapidement de canal. Pour ce faire, des sauts entre différentes fréquences sont effectués à des intervalles de temps fixes et prédéterminés. Chaque saut utilise une bande passante spécifique et connue à l'avance, ce qui permet à l'émetteur et au récepteur de rester synchronisés malgré ces changements constants [9].

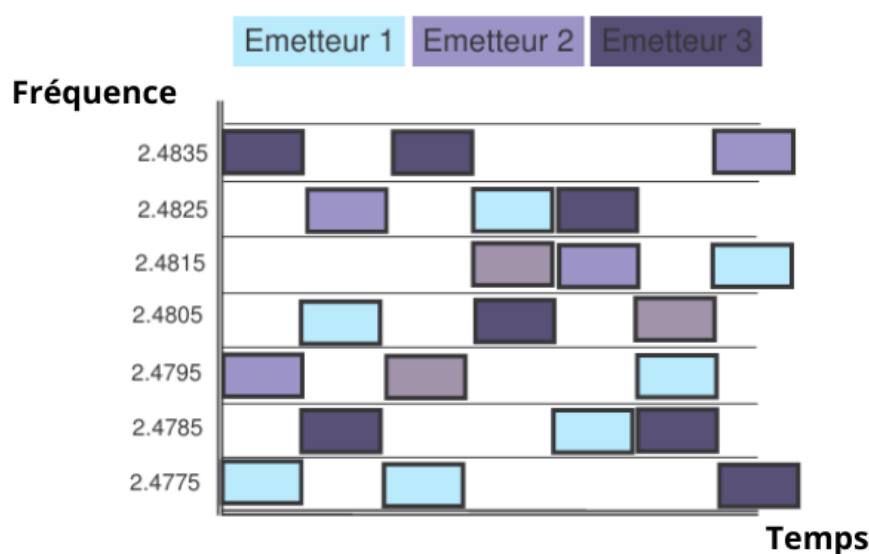


Figure 2.1: Transmission en mode FHSS

2.2.2 Affichage des informations

Dans la conception d'un tableau de bord connecté pour scooter, l'affichage agit comme une interface entre le système et le conducteur. Sa conception doit équilibrer la présentation des informations et les exigences de sécurité et d'ergonomie d'utilisation.

Les principes de conception d'interfaces sont la clarté, la simplicité et la réduction de l'encombrement visuel[10]. Un affichage minimaliste présentant uniquement les données critiques de manière lisible et simple est donc préféré pour garantir que le conducteur reste concentré sur la route. La priorisation de l'information est aussi essentielle pour améliorer la réactivité du conducteur face aux informations pertinentes.

2.2.3 Horloge en temps réel

Le développement d'une horloge en temps réel est un aspect clé pour un tableau de bord connecté afin de garantir la précision temporelle des informations affichées et des fonctionnalités dépendantes du temps. Cette horloge informe le conducteur du temps actuel sans qu'il prenne son smartphone en conduisant.

2.2.4 Contraintes techniques et exigences

Cette étape s'intéresse aux exigences et contraintes liées au développement et qui se divisent en contraintes matérielles et logicielles:

a. Contraintes matérielles

Elles incluent le choix du microcontrôleur, la taille de l'écran, la consommation d'énergie, la résistance aux conditions externes, etc.

b. Contraintes logicielles

Elles concernent le choix des protocoles de communication Bluetooth spécifiques, les performances en temps réel, les langages de programmation à utiliser, etc.

Les exigences définissent ce que le système doit faire et comment il doit se comporter pour répondre aux besoins des utilisateurs et aux objectifs du projet. Parmi ces exigences, on cite:

- Réception et affichage des notifications du smartphone.
- Connexion au smartphone via Bluetooth
- Mise à jour des informations en temps réel.
- Facilité d'utilisation et lisibilité.

Ainsi que d'autres exigences techniques liées à la solidité et à la conception du boîtier et d'alimentation, etc.

2.3 Constitution matérielle et logicielle

Afin de proposer une solution complète de tableau de bord de scooter avec un affichage minimaliste et une connectivité BLE, une combinaison entre matériel et logiciel est indispensable. Cette approche garantit la fiabilité et l'expérience utilisateur améliorée et sécurisée.

2.3.1 Composants matériels

La réussite d'un projet dépend des choix d'outils matériels. Pour ce projet, cette étape détermine la faisabilité technique, le coût et la facilité du développement.

a. Affichage minimaliste

Le cœur visuel du tableau de bord repose sur un bon affichage pour faciliter la lisibilité. Les options incluent un écran OLED pour son excellente lisibilité en plein soleil, leurs noirs brillants ainsi que pour leur faible consommation d'énergie et leur design fin qui accroît la liberté de conception [11].

Les matrices LED représentent une solution d'affichage robuste et énergétiquement économe, idéale pour les applications nécessitant une signalisation simple mais efficace. Un autre choix est l'écran TFT qui est un type d'écran LCD et qui utilise des transistors à couches minces pour contrôler chaque pixel individuellement. L'écran TFT offre une bonne qualité d'image ainsi qu'un taux de rafraîchissement rapide[12].

b. Module de connectivité

La communication avec le smartphone est assurée par un module BLE. Au milieu de cette architecture se trouve un microcontrôleur responsable de la gestion des données, de l'affichage et de la coordination de la connectivité. De plus, le microcontrôleur gère l'affichage de l'interface d'utilisateur sur l'écran.

c. Interface utilisateur

L'interaction avec le tableau de bord sera simplifiée pour minimiser la distraction et aussi augmenter le temps de rafraîchissement d'écran pour garantir un affichage en temps réel des informations et réduire le retard temporel. Des boutons tactiles ou des commandes physiques permettront une navigation entre les différents affichages et interfaces.

2.3.2 Composants logiciels

La partie logicielle donne vie au matériel utilisé. C'est un ensemble de codes et de commandes qui décrit le fonctionnement du système constitué d'un microcontrôleur.

En plus du tableau de bord, une application dédiée sur smartphone sera le point de gestion et de personnalisation du tableau de bord. Elle permettra une connexion BLE fluide, le transfert de données tout en évitant la surcharge d'informations. Un firmware sur un microcontrôleur est chargé de lire les données transmises via BLE et de les traduire en un affichage intelligent. Il gère la présentation des informations essentielles en temps réel et supporte les mises à jour Over-The-Air (OTA) pour des améliorations continues des performances et l'ajout de nouvelles fonctionnalités.

2.4 Choix du microcontrôleur

Le choix du microcontrôleur pour un tableau de bord connecté avec BLE est une étape importante, car c'est un élément clé du système qui doit assurer une performance optimale avec une consommation énergétique modérée. Une analyse rigoureuse des critères suivants est donc essentielle. Parmi les critères de référence pour ce choix, on cite:

- L'intégration d'une connectivité BLE et de la gérer.
- La capacité de traitement des données sans latence en ayant une mémoire Flash suffisante ainsi qu'une mémoire vive adaptée pour les données Bluetooth et l'affichage.
- La compatibilité avec l'écran d'affichage via SPI ou I2C, en supportant des bibliothèques graphiques.

- la robustesse et la durabilité du microcontrôleur face à l'environnement externe (température, humidité, etc).

Il existe plusieurs microcontrôleurs qui ont plusieurs avantages et inconvénients. Le fait de prendre un entre eux pour l'intégrer pour ce projet revient à faire un compromis entre ses avantages et inconvénients. Parmi les choix disponibles, on trouve l'ESP32-S3, le STM32WB, et le Nordic nRF52840. L'ESP32-S3 a été choisi...raison que nous aborderons dans cette suite.

2.4.1 ESP32-S3

L'ESP32-S3 (introduit par la figure 2.2) est un microcontrôleur conçu par Espressif Systems ¹, qui est une entreprise chinoise basée à Shanghai. C'est une solution fiable et populaire dans le domaine de l'Internet des Objets, vu qu'il intègre un module Bluetooth ou BLE. La production de l'ESP32 n'a démarré qu'à la fin de l'année 2016, ce qui en fait une nouvelle entrée dans la gamme des processeurs [13].



Figure 2.2: Puce ESP32-S3-WROOM-1[14]

2.4.2 Atouts de l'ESP32-S3

l'ESP32-S3 appartient à la série de SoC ESP32-S3. En considérant les caractéristiques mentionnées dans le tableau 2.1 de la page 22, il présente un compromis optimal entre performance, connectivité, consommation énergétique et facilité de développement.

¹ESPRESSIF SYSTEMS, disponible sur <https://www.espressif.com>

Tableau 2.1: Caractéristiques de l'ESP32-S3[15]

Processeur	Xtensa LX7 dual-core 32-bit, jusqu'à 240 MHz
Connectivité Wi-Fi	Wi-Fi (802.11 b/g/n) avec un débit allant jusqu'à 150 Mbps; antenne PCB intégrée.
Connectivité Bluetooth	Bluetooth LE 5.0, Bluetooth Mesh, avec différents débits, extensions de publicité, ensembles de publicité multiples et algorithme de sélection de canal.
Interfaces Intégrées	CAN, RS485, I2C; emplacement pour carte microSD; port USB intégré.
Mémoire	512 Ko de SRAM, 384 Ko de ROM; 8 Mo de PSRAM embarquée; 16 Mo de mémoire Flash embarquée.

2.5 Considérations pour l'écran d'affichage

L'écran d'affichage est l'interface principale pour le conducteur du tableau de bord connecté. Son choix repose sur des critères essentiels tels que la lisibilité, la consommation énergétique et la robustesse.

2.5.1 Types des écrans envisagés

Plusieurs technologies d'affichage peuvent être considérées, chacune avec ses propres avantages et inconvénients. Parmi ces technologies, on cite:

- **Écran OLED**

L'OLED offre une excellente lisibilité même en plein soleil, grâce à son contraste élevé et à l'émission de lumière propre à chaque pixel. Sa très faible consommation d'énergie en fait un choix de prédilection pour les applications autonomes. De plus, il est largement compatible avec des microcontrôleurs comme l'ESP32 et le STM32.

- **Écran TFT couleur compact**

Les écrans TFT couleur permettent un affichage plus détaillé et personnalisable, capable de présenter des graphiques avancés. Cependant, ils ont une consommation énergétique généralement plus élevée que les écrans OLED, ce qui doit être pris en compte pour la gestion de la batterie du scooter[12].

2.5.2 Critères de choix de l'écran TFT

Le choix de l'écran doit se baser sur une combinaison des exigences du projet. Pour un tableau de bord de moto, la lisibilité sous la lumière directe du soleil est un critère essentiel. Les écrans OLED sont les plus performants dans cette condition. Afin de minimiser la sollicitation de la batterie du scooter, l'écran doit avoir une consommation d'énergie modérée.

En tenant compte des avantages des écrans OLED, leur coût élevé et leur durée de vie potentiellement plus courte sont des facteurs critiques pour ce tableau de bord. En revanche, les écrans TFT, bien qu'ils n'offrent pas le même niveau de contraste, sont plus économiques et possèdent une durée de vie plus robuste face à l'affichage prolongé d'éléments fixes. Leur luminosité élevée est également un atout majeur pour la lisibilité en plein jour ou sous une lumière ambiante. De plus, la facilité d'intégration et la large disponibilité des bibliothèques pour les écrans TFT simplifient le processus de développement et optimisent le budget et le temps de réalisation du projet sans compromettre la fonctionnalité ou la lisibilité essentielle. Pour cela, un écran TFT 7 pouces tactile (Figure 2.3) a été utilisé pour le développement de ce projet.



Figure 2.3: ESP32-S3 avec écran 7 pouces tactile

2.6 Environnement de développement pour l'ESP32

Pour l'ESP32, l'environnement de développement permet d'écrire des programmes en langage C ou C++ qui peuvent s'exécuter sur le microcontrôleur, de les compiler et de les déployer via le "flashing". Cet environnement de développement

comprend généralement un éditeur de code, un compilateur C, des bibliothèques et des outils de débogage pour faciliter la création et le test des applications ESP32[13].

Il existe divers logiciels utilisés pour le développement avec l'ESP32. Pour ce projet, on a utilisé Espressif-IDE (Integrated Development Environment) qui est un environnement de développement basé sur Eclipse destiné au développement d'applications IoT utilisant l'ESP-IDF(Integrated Development Framework). Il est fourni avec les plugins Eclipse pour l'ESP-IDF, les plugins essentiels d'Eclipse, ainsi que d'autres plugins tiers de la plateforme Eclipse pour prendre en charge la création d'applications ESP-IDF[15].

2.7 Logiciel de développement de l'interface graphique

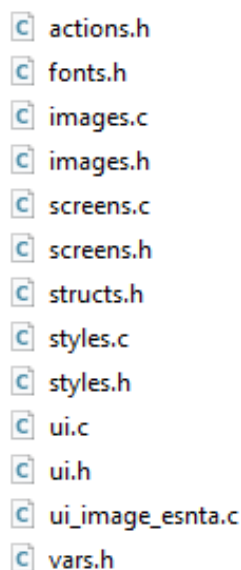
Pour la création de l'interface graphique d'utilisateur, nous nous orientons vers des outils de design performants afin de répondre aux contraintes du système.

Au cœur de cette approche se trouve Light and Versatile Graphics Library (LVGL), une bibliothèque graphique open-source conçue spécifiquement pour les microcontrôleurs et les systèmes embarqués à ressources limitées[16]. LVGL se distingue par sa capacité à créer des interfaces utilisateur riches et esthétiques, tout en conservant une consommation CPU minimale. Sa flexibilité lui permet de s'adapter à plusieurs écrans et périphériques d'entrée, ce qui en fait un choix idéal pour un affichage minimaliste sur un tableau de bord.

Pour profiter pleinement des capacités de LVGL, nous utiliserons Eez Studio[17]. Cet environnement IDE est spécifiquement conçu pour simplifier et accélérer le processus de conception d'interfaces graphiques embarquées. Eez Studio a été développé par Envoy² offre une suite d'outils visuels, de simulateurs et de fonctionnalités de débogage qui facilitent la création, la personnalisation et le test des interfaces basées sur LVGL. Lorsque l'interface est prête, cet IDE la compile en un code C pour faciliter son intégration dans le microcontrôleur et puis l'afficher sur l'écran. Les codes obtenus (Figure 2.4) sont répartis en plusieurs fichiers source (.c) et fichiers d'en-tête (.h).

Dans la figure 2.4, chaque fichier contient des informations sur une partie

²Envoy, disponible sur <https://www.envoy.eu/company/about-us/>



- actions.h
- fonts.h
- images.c
- images.h
- screens.c
- screens.h
- structs.h
- styles.c
- styles.h
- ui.c
- ui.h
- ui_image_esnta.c
- vars.h

Figure 2.4: Fichiers issus de compilation UI

du design. Par exemple, le fichier **"images.c"** contient les noms des images intégrées ainsi que leurs pointeurs pour que le microcontrôleur les prenne en considération. Un autre exemple est le fichier **"screens.c"** qui est essentiel pour la personnalisation car il contient les interfaces graphiques faites et leurs caractéristiques comme la couleur d'arrière-plan, la résolution, le nom, ainsi que les éléments qui se trouvent à l'intérieur avec leurs propres caractéristiques et positions sur l'écran.

2.8 Schéma global de la solution embarquée

La figure 2.5 qui suit, représente un schéma récapitulatif de la solution proposée pour le développement d'un tableau de bord connecté pour scooter, en intégrant les composants matériels et logiciels sélectionnés. Il inclut également les interactions et la méthode de communication entre ces composants dans le but de concevoir le tableau de bord connecté pour scooter.

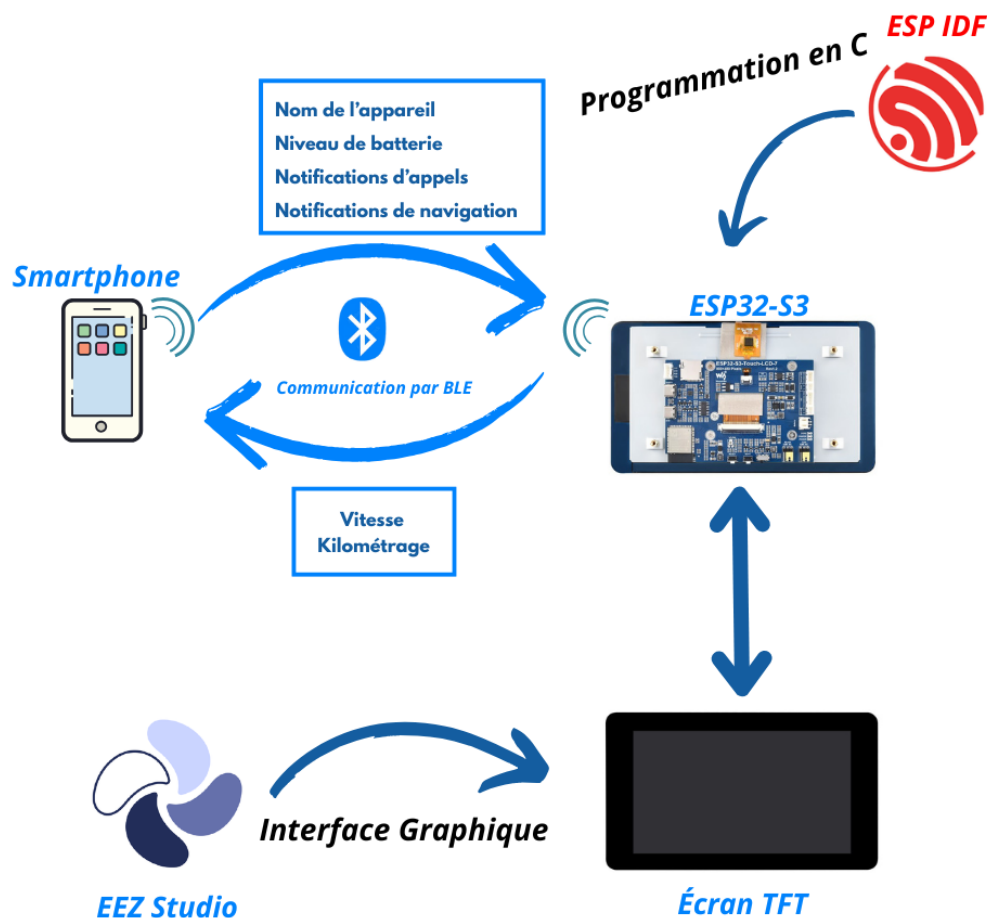


Figure 2.5: Schéma global de la solution retenue

2.9 Conclusion

En conclusion, cette recherche a posé les fondements permettant le développement d'un dispositif de tableau de bord connecté en validant des choix matériels clés, comme celui du microcontrôleur ESP32-S3 et de l'écran OLED TFT, pour leur compatibilité et performance avec les exigences de la sécurité la plus adéquate à l'objectif, l'ergonomie, mais aussi de convivialité. Au niveau logiciel, l'intégration de la librairie LVGL via Eez Studio apparaît comme un choix idéal afin d'aboutir à l'interface dynamique et moderne souhaitée. L'ensemble de ces choix et décisions s'inscrit dans la recherche de la solution moderne et ergonomique et permet une amélioration de l'expérience de conduite.

Chapitre 3

Développement et test du tableau de bord connecté

3.1 Introduction

Après la présentation d'une étude approfondie des spécifications et des exigences du tableau de bord connecté, qui a permis de définir ses fonctionnalités, il est temps de passer à la phase de développement. Ce chapitre portera sur la réalisation technique du système, où les choix et les solutions matérielles et logicielles seront mis en œuvre pour transformer le concept en une solution fonctionnelle.

3.2 Développement de la connectivité BLE sur ESP32

Le BLE est la technologie de communication entre le smartphone et le tableau de bord, plus précisément l'ESP32-S3. Son développement inclut l'initialisation, la diffusion, et l'établissement d'une connectivité fiable. Pour ce fait, on a opté à utiliser NimBLE [18] qui est une pile (stack) BLE open source et hautement configurable, initialement développée dans le cadre du projet Apache Mynewt. Sur les modules ESP32 d'Espressif, NimBLE est une option de choix pour implémenter la connectivité BLE vu qu'elle contient le GATT, l'ATT, et le GAP à la fois.

Afin de pouvoir inclure NimBLE dans un projet d'Espressif IDE, il faut l'activer dans le menu config à travers Espressif IDF. Une fois orienté vers le chemin du projet, on active le Bluetooth et le NimBLE comme illustré sur la figure 3.1 présentée sur la page 28. La bibliothèque de NimBLE possède une

en-tête ‘Nimble.h’ qui doit être déclarée au début pour pouvoir utiliser les fonctions qu’elle contient. Le NimBLE nécessite un Real Time Operating System

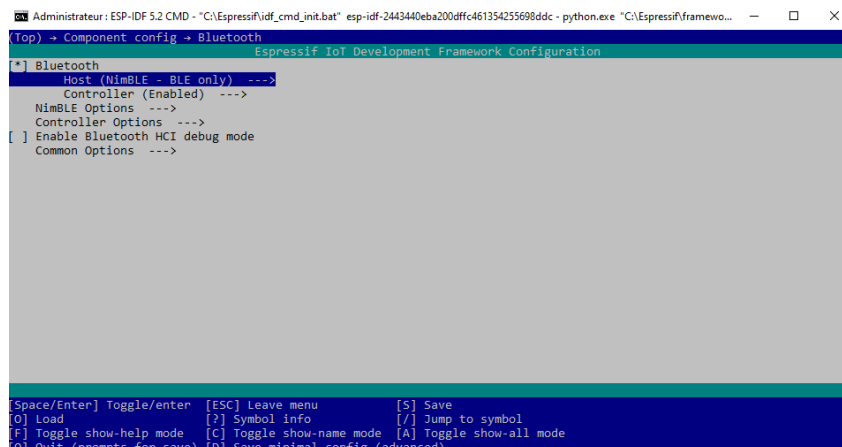


Figure 3.1: Activation Bluetooth sur menu config

(RTOS), un composant logiciel fondamental pour les systèmes embarqués qui exigent une gestion précise des tâches. Dans ce projet, nous avons utilisé FreeRTOS qui est un RTOS open-source maintenu par Real Time Engineers Ltd et largement adopté et conçu pour les systèmes embarqués à ressources limitées à base d’un microcontrôleur[19].

3.2.1 Services et caractéristiques utilisés

Les données à échanger sont réparties sur des services définis par des UUID et qui contiennent des caractéristiques de chaque donnée tel que le montre le tableau suivant :

Tableau 3.1: Services et caractéristiques BLE pour le tableau de bord

Service	UUID	Caractéristique	UUID
Infos sur l’appareil	0x180A	Nom de l’Appareil	0x2A00
Batterie	0x180F	Niveau de batterie	0x2A19
Notifications et alertes	0x1811	Appels	0x2A46
Données de Scooter	0x1234	Vitesse	0x5678
		Kilométrage	0xABCD
Données de navigation	0x0011	Instructions	0x0033
		Distance au prochain tour	0x0022
		Distance restante	0x0044
		Temps restant	0x0055
		Heure d’arrivée	0x0066

En observant le tableau 3.1, un service peut contenir plusieurs caractéristiques à la fois si elles appartiennent au même champ d’informations (par ex-

emple, les informations de navigation sont réunies en un seul service).

Quelques UUIDs sont prédéfinis par le Bluetooth SIG [2], comme le nom de l'appareil ou l'état de batterie. Tandis que d'autres sont personnalisés. Ces services seront exposés lors de la diffusion pour un client. La pile NimBLE gère le processus de diffusion, en permettant à l'ESP32 d'envoyer périodiquement des paquets de publicité contenant les UUIDs des services et d'autres données pertinentes.

3.2.2 Gestion des événements

La gestion des événements est un aspect fondamental de NimBLE. Lorsqu'un événement significatif se produit au niveau de la couche BLE, comme l'établissement ou la perte d'une connexion Bluetooth, une déconnexion inattendue, ou autre, ces événements sont relayés à l'application via des callbacks ou des fonctions de gestion ("event handlers"), qui permettent de réagir de manière appropriée.

Par exemple, à la suite d'une connexion réussie, le gestionnaire d'événements peut afficher un message pour le débogage ou initier une requête Maximum Transmission Unit (MTU) qui représente la taille maximale d'un paquet de données pouvant être transmis en une seule fois sans fragmentation. Elle est exprimée en octets et sa taille peut être modifiée dans le menu config. La taille standard ou par défaut du MTU dans BLE est de 23 octets; vu que certaines données du tableau de bord dépassent cette limite, la MTU a été définie à 128 octets dans le menu config.

En cas de déconnexion, le gestionnaire d'événements peut tenter une reconnexion automatique ou notifier l'utilisateur. Cette approche garantit une réactivité optimale du système et une gestion efficace des différents états de la liaison BLE. De même, lorsque la diffusion est terminée, le développeur peut rediffuser les paquets de publicité.

Pour ce projet, une fonction intitulée **"ble_gap_event"** s'intéresse à la gestion de ces événements. Lorsqu'une connexion est établie, l'ESP affiche un texte sur le terminal. En cas de déconnexion, un message s'affiche sur l'écran et le microcontrôleur lance une tâche qui termine cette connexion BLE pour permettre l'établissement d'une nouvelle connexion. Si la diffusion est terminée,

l'ESP rediffuse et affiche un message à titre indicatif.

3.3 Échange des données

L'échange bidirectionnel des données constitue le cœur de la connectivité du tableau de bord. Il implique l'envoi des informations depuis le microcontrôleur vers le smartphone (la vitesse du scooter et le kilométrage) ainsi que la réception de données en provenance du smartphone (les notifications d'appels, les instructions de navigation, etc.). Cette communication est gérée via le BLE, où les données sont structurées en services et caractéristiques (tableau 3.1) pour assurer une transmission efficace par les deux appareils.

3.3.1 Réception des données

Le tableau de bord reçoit plusieurs données de la part du smartphone. Par exemple, pour recevoir le nom du smartphone, on crée une fonction responsable de la gestion d'événements GATT pour une écriture dans la caractéristique 0x2A00 appartenant au service 0x180A. Elle contient comme arguments l'identifiant unique de la connexion BLE active et l'identifiant de la caractéristique GATT sur laquelle l'écriture a eu lieu. La fonction commence par extraire la longueur des données reçues et implémente une mesure de sécurité en tronquant les données si elles dépassent la taille allouée, prévenant ainsi les débordements du tampon utilisé pour le stockage de la donnée. Les données du nouveau nom sont ensuite copiées dans la variable du nom de l'appareil déjà définie et la chaîne est correctement terminée.

Listing 3.1: Réception des données

```
static int gatt_device_name_write_cb(uint16_t conn_handle, uint16_t
    attr_handle, struct ble_gatt_access_ctxt *ctxt, void *arg) {
    int len = ctxt->om->om_len;
    if (len > sizeof(device_name) - 1) len = sizeof(device_name) -
        1; // Prevent buffer overflow

    memcpy(device_name, ctxt->om->om_data, len);
    device_name[len] = '\0';
```

```
console_printf("Device Name: %s\n", device_name); //Nouveau nom  
de l'appareil  
return 0;  
}
```

C'est avec le même principe que chaque caractéristique possède sa fonction qui stocke la donnée sur un buffer pour la transmettre à la variable de cette donnée.

Dans la gestion des données, l'approche utilisée est l'allocation statique (figure 3.2). Cette méthode réserve une quantité de mémoire fixe et prédéfinie pour stocker les informations sur le buffer. L'avantage principal de l'allocation statique réside dans sa simplicité et sa rapidité d'accès.

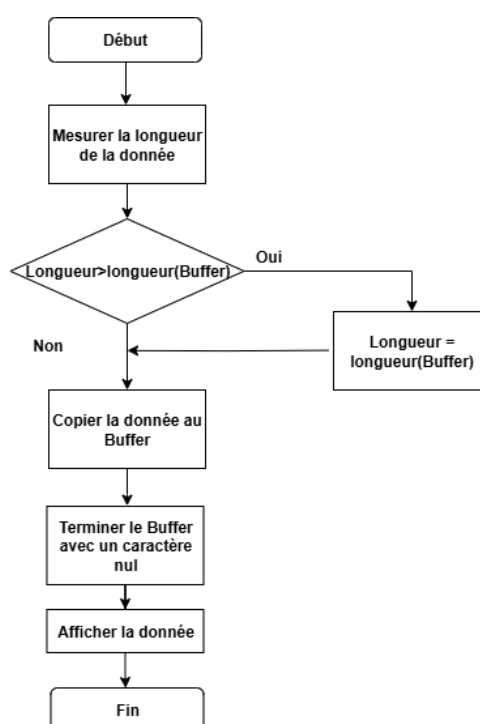


Figure 3.2: Allocation statique

Cependant, cette rigidité ne peut pas s'adapter aux variations imprévues de la taille des données. Lorsque les données dépassent la taille allouée de manière statique, le système rencontre des problèmes de débordement. Pour résoudre ce problème, on utilise l'allocation dynamique (voir la figure 3.3 de la page suivante) qui permet de réserver de la mémoire en temps réel pendant l'exécution du programme.

En utilisant l'allocation dynamique pour l'instruction de navigation qui prend différentes tailles selon son contenu, on assure la flexibilité dans l'utilisation des

ressources mémoire.

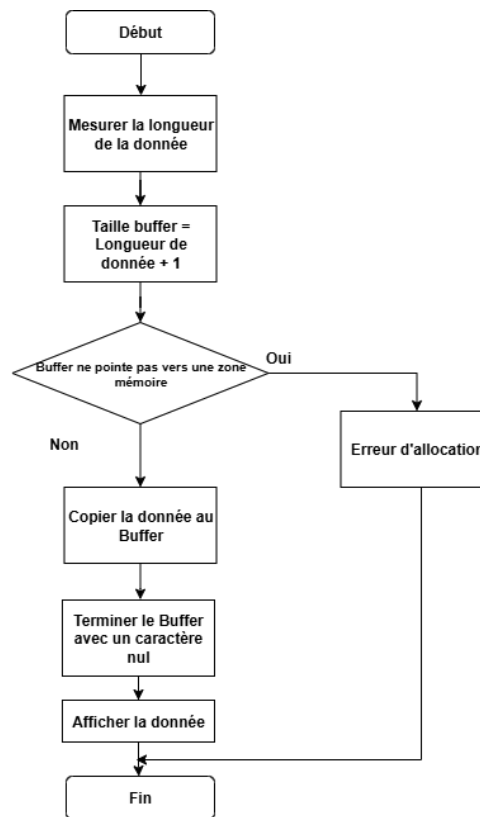


Figure 3.3: Allocation dynamique de la mémoire

3.3.2 Envoi des données

Comme le tableau de bord permet l'échange des données dans les deux sens, il est possible de transmettre au smartphone la vitesse et le kilométrage instantané (en chaque seconde).

La méthode de l'envoi est similaire à celle de la réception. On définit une fonction statique de rappel (callback) et chaque fois qu'un appareil BLE distant initie une requête de lecture sur la caractéristique de la vitesse ou du kilométrage, cette fonction est appelée. Son rôle est de fournir la valeur actuelle de la vitesse. Le code de la fonction commence par afficher la valeur de la vitesse sur la console de débogage et ensuite cette valeur est ajoutée au tampon de sortie **os_mbuf** qui sera utilisé par la pile NimBLE pour envoyer la réponse au client. La fonction retourne 0 en cas de succès ou un code d'erreur GATT (BLE_ATT_ERR_INSUFFICIENT_RES) si l'ajout des données au tampon échoue pour assurer une communication fiable de la vitesse du scooter. Le même principe est appliqué à la valeur du kilométrage.

Pour ce projet, et vu que la pratique ne nous a pas permis de manipuler des valeurs réelles de vitesse et kilométrage, la méthode adoptée était d'envoyer une valeur qui s'incrémente de un à chaque requête de lecture pour chaque caractéristique. Si à la première lecture, la vitesse envoyée était à 20 km/h, à la deuxième elle sera à 21 km/h et ainsi de suite.

3.3.3 Gestion des caractères spéciaux

Pour garantir un affichage précis et éviter les problèmes de compatibilité, notamment sur l'écran et la console qui ne supportent pas l'encodage complet des caractères spéciaux, il est nécessaire de remplacer ces caractères accentués et spéciaux par leurs équivalents American Standard Code for Information Interchange (ASCII) de base (voir le tableau 3.2). Pour cela, une fonction est conçue pour normaliser les caractères accentués français au sein d'une chaîne de caractères encodée en Unicode Transformation Format 8 bit (UTF-8), en les remplaçant par leurs équivalents non accentués ASCII. Elle opère en parcourant la chaîne avec deux pointeurs source et destination. Lors de la détection d'un caractère UTF-8 commençant par 0xC3 (typique pour les caractères accentués latins), la fonction examine le second octet pour identifier le caractère précis (comme 'à', 'é', 'ù', 'î', 'ô'). Elle écrit le caractère ASCII normalisé correspondant ('a', 'e', 'u', 'i', 'o') à la position destination, et avance la source de deux octets (le caractère UTF-8) et destination d'un seul octet (le caractère normalisé). Pour les caractères ASCII standard, ils sont simplement copiés. Enfin, la chaîne est terminée par un caractère nul pour garantir la compatibilité avec des systèmes qui pourraient ne pas supporter l'UTF-8 ou les caractères spéciaux, comme c'est le cas de notre tableau de bord.

Tableau 3.2: Normalisation des caractères spéciaux basée sur les codes UTF-8

Code Hexadécimal	Caractère original	Caractère normalisé
0xA0	à	a
0xA1	á	a
0xA8	è	e
0xA9	é	e
0xB9	ù	u
0xAE	î	i
0xAF	ï	i
0xB4	ô	o

3.3.4 Affichage des images indicatrices de la direction

Sur le texte indicatif de l'itinéraire, il peut y avoir une direction à suivre. En utilisant cette direction, il est possible d'afficher des flèches des directions (directe, à droite, gauche, etc.) pour améliorer la lisibilité de l'itinéraire. Le processus commence par extraire le texte et ensuite vérifier s'il contient des mots-clés des directions en utilisant la fonction **strstr()** qui prend comme arguments le texte et la chaîne de caractères à chercher. Une fois la direction identifiée, on affiche sur la console un mot indicatif de la direction ainsi que l'image d'une flèche du sens à prendre sur l'écran.

Listing 3.2: affichage des flèches selon la direction

```
void extract_direction(const char *place, char *result) {
    if (strstr(place, "gauche")) {
        strcpy(result, "à gauche");
        lv_obj_clear_flag(gauche, LV_OBJ_FLAG_HIDDEN); //afficher
        l'image
    }
    else {
        strcpy(result, "unknown");
    }
}
```

Le code ci-dessus explique comment savoir si le texte contient le terme "gauche" pour afficher une flèche à gauche. C'est le même principe pour les autres directions et instructions comme demi-tours, rond-point, etc.

3.4 Horloge à temps réel

L'intégration d'une horloge est fondamentale pour le tableau de bord connecté pour réduire la dépendance du conducteur au téléphone portable. Elle peut être implémentée soit avec un module externe, soit en tirant parti du Real Time Clock (RTC) intégré dans l'ESP32.

3.4.1 Mode veille

Le mode veille est une fonctionnalité des microprocesseurs qui consiste à désactiver différents composants pour économiser l'énergie. Il existe deux modes de veille. Le mode veille simple qui met en pause l'unité de traitement et d'autres

périphériques non essentiels, mais maintient la mémoire Random Access Memory (RAM) et certains périphériques. En revanche, le mode veille profonde est le mode d'économie d'énergie le plus agressif car il coupe l'alimentation de la plupart des composants de la puce, y compris la mémoire RAM, alors qu'il laisse le RTC et sa petite mémoire actives. L'ESP32 est très économique en énergie et offre plusieurs modes de veille adaptés à divers scénarios d'utilisation, notamment le mode veille léger et le mode veille profonde.

3.4.2 Horloge RTC

L'ESP32 utilise deux horloges matérielles pour maintenir l'heure système. Le premier est l'horloge à haute résolution qui offre une grande précision mais n'est pas disponible dans le mode veille. Le deuxième est le RTC qui est conçu pour maintenir l'heure et la date et gérer certaines fonctions à faible consommation d'énergie, même lorsque la puce principale de l'ESP est en mode veille ou mode veille profonde[15]. Pour cela, nous avons opté à l'utilisation de la RTC pour le développement d'une horloge temps réel.

3.4.3 Développement de l'horloge

Le but est de concevoir une horloge qui compte l'heure et la date en utilisant le RTC. Ce module nécessite des paramètres d'horloge pour commencer le comptage comme l'heure, la minute, le jour, etc. Ces paramètres seront réunis en une structure de temps utilisée par une fonction de comptage qui fait partie de la bibliothèque **"time.h"**. Afin de pouvoir modifier l'heure du système en définissant une nouvelle valeur de temps, on utilise la fonction **settimeofday()** définie dans la bibliothèque **"sys/time.h"**.

Listing 3.3: Initialisation du RTC

```
#include <stdio.h>
#include <time.h>          // Pour struct tm et mktime()
#include <sys/time.h>      // Pour struct timeval et settimeofday()

void set_time() {
    struct tm tm = {
        .tm_year = 2025 - 1900, //le comptage depuis 1900
        .tm_mon  = 4, //les mois commencent par 0
    };
}
```

```

        .tm_mday = 5,
        .tm_hour = 15,
        .tm_min  = 49,
        .tm_sec   = 0
    };
    time_t t = mktime(&tm);
    struct timeval now = {.tv_sec = t};
    settimeofday(&now, NULL);
}

```

Pour pouvoir récupérer le temps actuel et l'utiliser dans d'autres tâches comme l'affichage ou le réglage, on définit une fonction qui récupère le temps et qui utilise la fonction **localtime()**.

Un défi rencontré est que, lors d'une coupure d'alimentation de l'ESP32, l'horloge RTC (Real-Time Clock) réinitialise son comptage. Une solution envisagée consistait à utiliser la mémoire de la RTC pour stocker les paramètres temporels actuels et éviter la remise à zéro. Cependant, il s'est avéré que cette approche n'était pas idéale vu que, lors de l'extinction du motorcycle (et par conséquent de l'ESP32), le comptage du temps était interrompu et le temps écoulé entre l'arrêt et le prochain démarrage n'était pas pris en compte.

Pour résoudre ce problème de persistance temporelle, nous avons opté pour l'utilisation du mode veille profonde. Ce mode permet à l'ESP32 de réduire sa consommation électrique sans pour autant rester complètement éteint, tout en maintenant la RTC fonctionnelle et active pour assurer un comptage ininterrompu de l'heure comme l'illustre la figure 3.4.

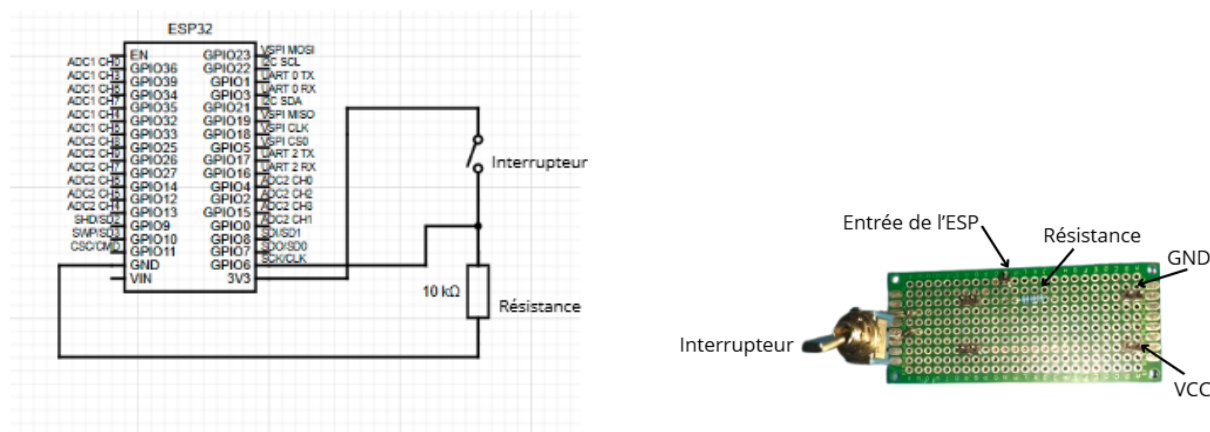


Figure 3.4: Schéma et circuit du mode veille

En pratique, la broche 6 de l'ESP32 est configurée en entrée et connectée à une résistance pull-down. La carte s'allume lorsque l'interrupteur est fermé, et elle entre en mode veille profonde si l'interrupteur est ouvert. Dans ce dernier cas, l'ESP32 attend constamment un signal sur cette même broche pour se réactiver. Il est important de souligner que, durant ce mode veille, la RTC reste activée et maintient son comptage pour garantir une mesure du temps continue.

3.4.4 Réglage de l'horloge

Le réglage de l'horloge interne de la RTC est une fonctionnalité importante pour assurer la précision et la persistance temporelle du tableau de bord connecté. Une fois configurée, la RTC maintiendra l'heure, même en cas de redémarrage (hors coupure de courant complète) ou d'entrée en modes de veille profonds. Si l'heure affichée ne convient pas à l'heure actuelle, il a été intéressant de développer une solution pour la régler et forcer le RTC à compter à partir des nouveaux paramètres d'horloge passés à la fonction **"set_time"**.

En utilisant l'écran TFT tactile intégré avec l'ESP32, il nous a été possible d'implémenter une interface permettant de saisir l'heure désirée directement sur l'écran. Des boutons virtuels sont affichés et interagissent au toucher pour ajuster les heures et les minutes. Une fois la configuration terminée, cette nouvelle heure sera transmise au RTC pour être les nouveaux paramètres de référence pour le comptage du temps.

3.5 Alimentation du tableau de bord

Pour alimenter le tableau de bord (figure 3.5), on utilise la batterie du scooter qui délivre une tension d'environ 12V. Vu que le microcontrôleur utilisé pour le tableau de bord est un ESP32-S3 qui fonctionne sous une tension de 3.3V[20], il est important d'abaisser la tension issue de la batterie pour l'alimentation de l'ESP32-S3.

L'activation et désactivation du mode veille se fait par l'utilisation de la broche 6. Lorsque cette broche reçoit un signal, la carte s'allume; sinon elle rentre en mode veille profond. Durant ce temps, l'ESP32 reste toujours alimenté depuis la batterie du scooter à travers le port UART de type C. En implémentation réelle, dès que le contact est mis et que le signal électrique

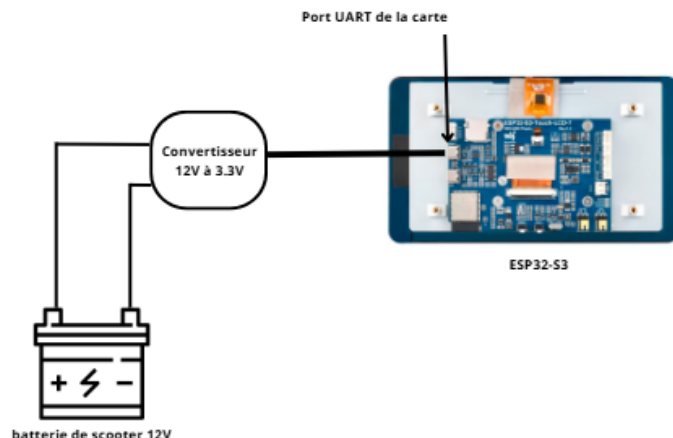


Figure 3.5: Alimentation de l'ESP32

correspondant est détecté sur cette broche, l'ESP32 interprète cette impulsion comme le signal de démarrage nécessaire pour activer et initialiser le tableau de bord connecté.

3.6 Notion de table de partition

La table de partition de l'ESP32 est un élément fondamental de l'organisation de la mémoire flash qui assure le bon fonctionnement de notre projet de tableau de bord connecté[15]. Elle définit comment la mémoire flash est divisée en différentes sections, ou partitions, chacune ayant un rôle spécifique. Pour ce projet, une gestion judicieuse de cette table est primordiale, car elle permet de séparer le code de l'application, les données du système d'exploitation (FreeRTOS), les données persistantes (telles que les configurations utilisateur ou les données d'étalonnage), et les partitions OTA nécessaires pour les mises à jour logicielles à distance.

Pour créer une table de partition, on crée un fichier csv où chaque ligne représente une partition de la mémoire flash. À chaque partition est assigné :

- Le nom de la partition. C'est une étiquette textuelle pour identifier la partition.
- Le type général de la partition. Il y a deux types principaux qui sont app pour les partitions contenant le code de l'application (firmware), et data pour les partitions contenant des données non-volatile.
- Le sous-type qui spécifie le rôle de la partition.

- L'adresse de départ de la partition dans la mémoire flash, exprimée en hexadécimal.
- La taille de la partition, également exprimée en hexadécimal.
- Des drapeaux optionnels, généralement pour indiquer si la partition est chiffrée ou autre.

Vu que le code de l'application pour ce projet intègre plusieurs fonctionnalités et bibliothèques (FreeRTOS, NimBLE, drivers, LVGL, esp_time, etc.), ainsi que les ressources intégrées (images, polices, interfaces, données de configuration incluses dans le binaire), la taille de la partition de l'application a été augmentée jusqu'à 6 Mégaoctets. Cette taille est très généreuse pour une application, mais reste un bon choix face au nombre d'images et des interfaces ainsi que les bibliothèques incluses. La nouvelle table de partition csv doit être indiquée dans le menu config pour que l'ESP32 la prenne en considération.

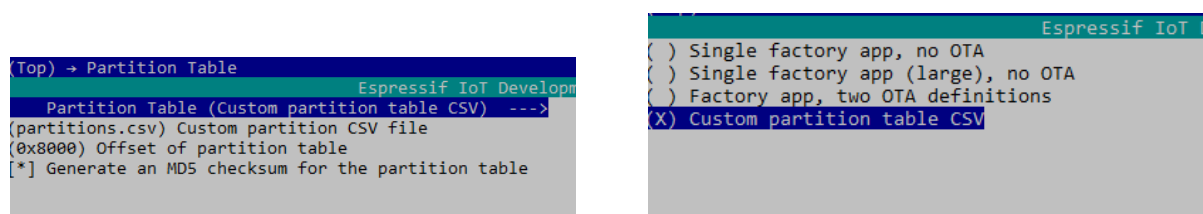


Figure 3.6: Déclaration de la table de partition dans menu config

Sur la figure 3.6, le choix de table de partition indique l'utilisation d'une table personnalisée. Par la suite, le nom de notre table de partition (partition.csv) est déclaré.

3.7 Développement de l'interface utilisateur

Une fois l'architecture matérielle et logicielle du système établie, l'étape suivante est le développement de l'interface utilisateur où l'expérience du conducteur prend forme. Pour cela, nous appuyons sur Eez Studio. Cet environnement de développement intégré sera l'outil pour concevoir et implémenter l'interface graphique de notre tableau de bord. Grâce à ses fonctionnalités dédiées à la création d'interfaces embarquées, Eez Studio permet de respecter les exigences d'affichage minimaliste et de lisibilité.

Trois interfaces sont développées pour ce tableau de bord:

- La première interface est une interface d'accueil qui remplacera l'interface traditionnelle d'un tableau de bord qui affiche les informations sur la vitesse, le niveau de carburant, etc.
- La deuxième est l'interface principale là où les données de navigation et l'horloge ainsi que d'autres informations sont accessibles.
- Pour effectuer des réglages au système, une troisième interface a été développée à cet effet. Le design de chaque interface est personnalisé pour les modes jour et nuit.

3.7.1 Interface d'accueil

Cette interface (voir la figure 3.7) remplace l'interface principale du tableau de bord. Elle contient le logo de notre école et d'autres textes descriptifs de ce projet, ainsi qu'un bouton qui initialise la diffusion des paquets de publicité afin que l'ESP32 soit reconnu par le smartphone.



Figure 3.7: Interface d'accueil en mode jour et nuit

Lorsqu'une demande de connexion est reçue depuis un smartphone, une zone de dialogue apparaît sur l'écran, proposant deux boutons : l'un pour accepter, l'autre pour rejeter cette connexion (voir la figure 3.8 sur la page 41). Si l'utilisateur appuie sur "Oui", l'ESP32 bascule vers la deuxième interface et commence à afficher les données reçues. En revanche, si "Non" est sélectionné, l'ESP32 se déconnecte immédiatement et la zone de dialogue disparaît.



Figure 3.8: Affichage d'une demande de connexion

3.7.2 Interface principale

L'interface principale est dédiée à la visualisation des données issues du smartphone (voir la figure 3.9). Ces données sont réparties sur l'écran d'une manière à faciliter leur lisibilité par le conducteur. Elle affiche également l'horloge RTC qu'on a déjà développée, ainsi que les données de navigation et les notifications d'appels. Tout comme l'interface précédente, elle possède deux modes d'affichage, clair et sombre.

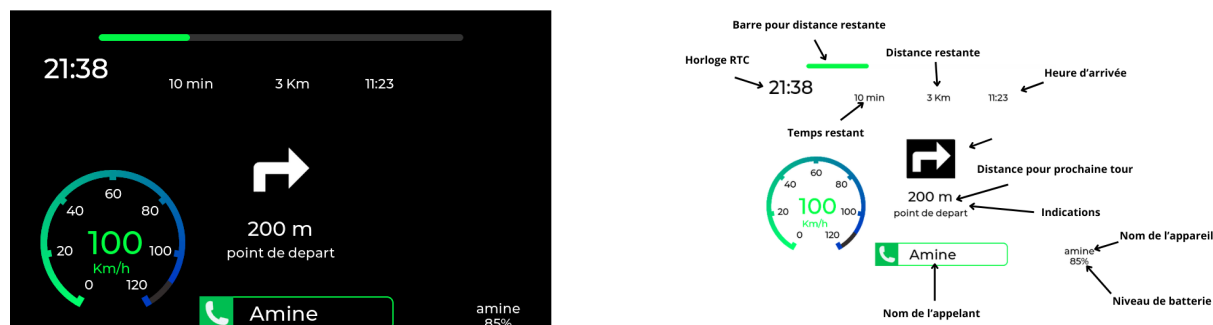


Figure 3.9: Interface principale en mode nuit et jour

Lorsqu'une notification d'appels est reçue, le nom de l'appelant est affiché sur sa zone de texte. Afin d'alerter le conducteur de cet appel, cette zone de texte apparaît pendant 1 seconde, puis disparaît pendant une autre seconde et apparaît de nouveau (affichage en clignotement). Ce processus est répété 5 fois avant que la zone de texte disparaisse. Les deux points entre les heures et les minutes clignotent également sur une durée de 1 seconde.

La barre horizontale à proximité de l'horloge indique la distance restante à arriver. Au début de l'itinéraire, on sauvegarde la distance maximale restante qui définit l'échelle de la barre. Ensuite, lorsque cette distance est réduite,

on assigne la distance parcourue à la barre de navigation jusqu'à atteindre la destination où la barre sera totalement chargée. Si le smartphone se déconnecte, on revient automatiquement à la première interface d'accueil (figure 3.10).

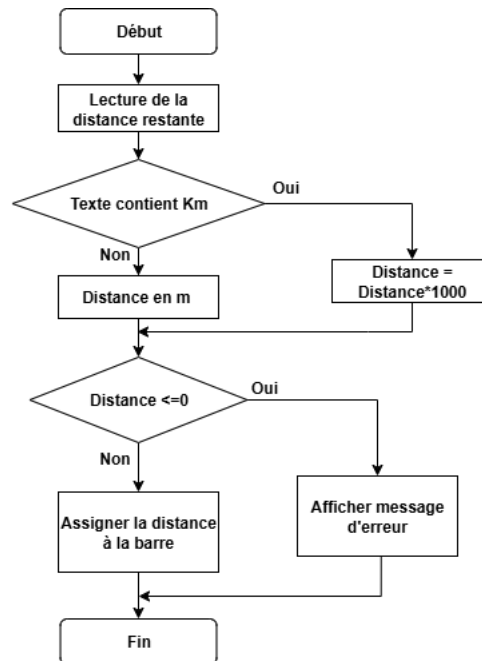


Figure 3.10: Assigner la distance restante à la barre horizontale

3.7.3 Interface de réglage

Cette interface s'intéresse aux réglages liés aux fonctionnalités du système (voir la figure 3.11). C'est là où l'on récupère l'heure actuelle pour la modifier et forcer la RTC sur un nouvel horaire. Il est aussi possible de permuter entre le mode sombre et clair via un bouton Switch. L'accès à cette interface se fait en appuyant sur un bouton situé derrière l'horloge.

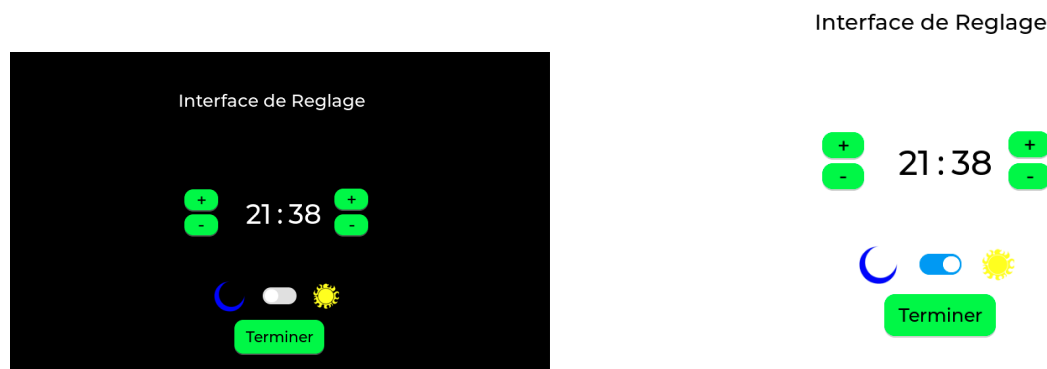


Figure 3.11: Interface de réglage en mode sombre et clair

Les deux boutons superposés situés à gauche de l'écran permettent d'augmenter et de décrémenter l'heure dans un intervalle de 0 à 23. De manière similaire, les deux boutons à droite sont dédiés au réglage des minutes, avec une plage allant de 0 à 59. L'interrupteur (switch) situé en dessous est conçu pour basculer entre le mode sombre (vers la gauche) et le mode clair (vers la droite), intégrant visuellement des icônes de lune et de soleil pour une meilleure compréhension. Une fois les réglages souhaités effectués, l'utilisateur appuie sur le bouton situé en bas de l'écran pour sauvegarder ces configurations et revenir à l'interface principale.

3.8 Test et validation

Suite au développement et à l'intégration de l'ensemble des composants matériels et logiciels, une phase de tests et de validation est indispensable pour garantir la fiabilité et la performance de notre tableau de bord connecté. Cette étape permet de vérifier la conformité du système aux spécifications initiales.

3.8.1 Test en utilisant NRF Connect

Avant de développer une application personnalisée pour ce tableau de bord, une application Android intitulée "nRF Connect" a été exploitée, il s'agit d'une application multiplateforme conçue par Nordic Semiconductors ¹ pour le test du BLE et ses fonctionnalités associées comme le scan, l'appairage, l'échange de données, etc. [21].

Sur l'interface d'accueil, lorsqu'on appuie sur le bouton pour commencer la diffusion, notre ESP32 apparaît sur l'application avec le tag qu'on lui a déjà assigné (ESP32-S3) et sur l'application, on aura une option de se connecter. Une fois connecté, la liste des services BLE sera disponible pour pouvoir lire ou écrire des données à travers ces services.

En appuyant sur un service, on aura accès aux caractéristiques qu'il contient avec la possibilité de lire ou d'écrire selon la permission mentionnée. L'avantage de cette application est qu'elle supporte une grande variété de types de données, soit en lecture, soit en écriture. Sur la figure 3.12, quelques services ne portent pas de nom, tandis que d'autres sont identifiés. Ceci est dû au fait que les

¹Nordic Semiconductor, disponible sur <https://www.nordicsemi.com>

services définis par le Bluetooth SIG (tels que précédemment mentionnés dans le tableau 3.1) portent un nom pour les différencier, par contre les services avec des UUID personnalisés, comme l'itinéraire, ne portent pas de nom significatif.

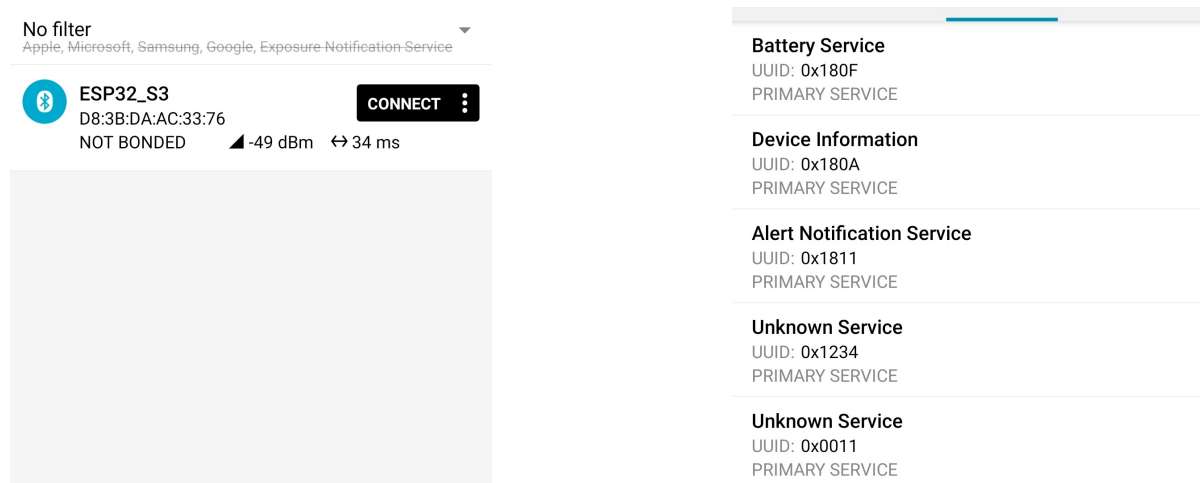


Figure 3.12: Tag de l'ESP et liste des services disponibles

3.8.2 Test avec une application personnalisée

Afin d'établir une communication entre le tableau de bord et le smartphone, une application Android a été développée (pour plus de détails, se référer au rapport de Master ²). Cette application se connecte automatiquement une fois que l'ESP32 commence la diffusion BLE et lui envoie et reçoit les données nécessaires de manière périodique, chaque seconde. Pour se connecter, on récupère le nom du smartphone afin de savoir d'où vient cette connexion. Par exemple, si le nom Bluetooth de notre smartphone est "amine", la zone de texte de l'interface d'accueil affichera "amine veut se connecter", suivi des deux boutons pour accepter ou refuser la connexion (figure 3.13).

²Bouchibane Amine, Développement d'une Application Mobile pour Communiquer avec le Tableau de Bord d'un Scooter, rapport de Master, ENSTA d'Alger, 2025.



Figure 3.13: Requête de connexion

Après l'acceptation de la requête, l'interface principale apparaît en affichant les données issues du smartphone.

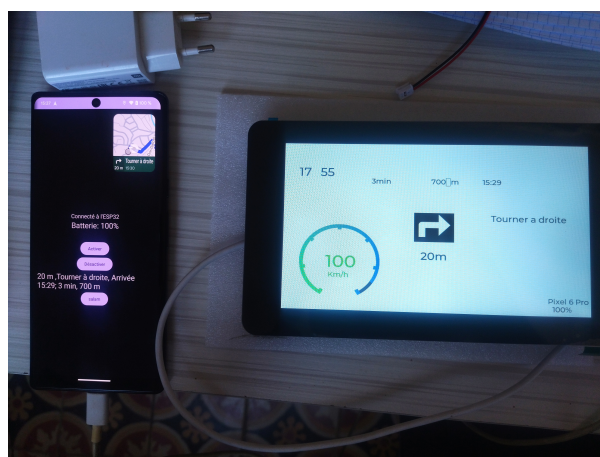


Figure 3.14: Echange entre le smartphone et l'ESP

Comme indiqué sur la figure 3.14, les informations de l'itinéraire sélectionné sur l'application Google Maps sont transmises au tableau de bord par l'intermédiaire de l'application réalisée, ensuite elles sont affichées sur l'écran TFT tactile. Pour la gestion des appels, lorsqu'un appel entrant est reçu, le nom de l'appelant s'affiche sur l'interface et clignote cinq fois avant qu'il ne disparaisse (voir la figure 3.15 sur la page suivante). À noter qu'un compteur se lance lorsqu'on répond à l'appel et la durée totale est affichée une fois que l'appel est terminé.

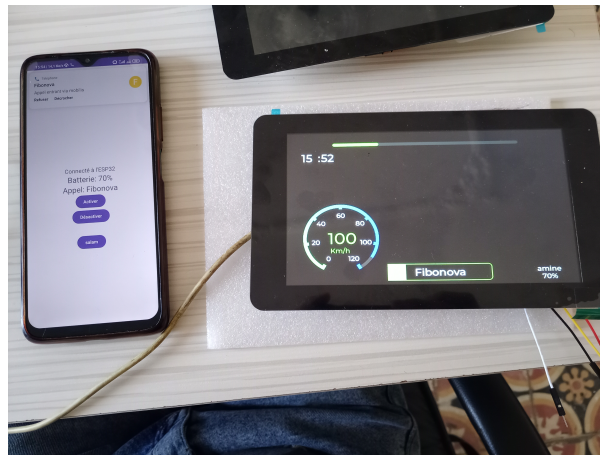


Figure 3.15: Réception des appels

3.8.3 Test de l'horloge

L'horloge de la RTC est accessible et configurable depuis la deuxième interface. Son réglage s'effectue en définissant l'heure et les minutes actuelles. Lorsque le mode veille profonde est activé, elle demeure fonctionnelle pour garantir l'absence de décalage temporel entre la période de mise en veille et le moment de la remise en marche.

Il est possible de suivre l'activité de la RTC en affichant l'heure sur la console chaque seconde. Par la suite, en activant le mode veille via l'interrupteur précédemment introduit par la figure 3.4 puis en rallumant l'ESP32 en fermant cet interrupteur, il a été prouvé que la RTC maintient effectivement le comptage de l'heure même en mode veille.

Mise en Marche	Mise en Marche
15 : 49	15 : 50
15 : 49	15 : 50
15 : 49	15 : 50
15 : 49	15 : 50
15 : 49	15 : 50
15 : 49	15 : 50
15 : 49	15 : 50
Mode Veille Active	15 : 50
	Mode Veille Active

Figure 3.16: RTC en mode veille

Comme indiqué sur la figure 3.16, avant d'activer le mode veille, l'heure était 15:49. Et après la deuxième mise en marche du tableau de bord, l'heure indiquée est 15:50. Ce qui prouve que le RTC reste actif en mode veille. Cette fonctionnalité est importée également à l'écran d'affichage pour la présenter au conducteur.

3.8.4 Test d'intégration dans un scooter

Ce tableau de bord est destiné aux véhicules à deux roues. Pour cela, il a été intégré sur un scooter, comme illustré sur la figure 3.17 pour tester son fonctionnement dans le monde réel et avoir une idée concernant les améliorations à apporter. L'expérience menée pour le test des fonctionnalités de base prévues par la solution proposée et précédemment décrites a fourni un retour positif, en attendant des résultats de tests plus étendus dans le temps et qui tiennent compte des retours d'expérience recensés auprès des utilisateurs potentiels.



Figure 3.17: Intégration dans un scooter

3.9 Conclusion

En conclusion, et après avoir parcouru les différentes étapes du développement de ce tableau de bord connecté, de la sélection du microcontrôleur ESP32 à l'intégration des fonctionnalités BLE et à la conception de l'interface utilisateur avec LVGL, ce chapitre a mis en lumière la complexité et la richesse de cette démarche. Les défis liés à la gestion de l'énergie et à la normalisation des carac-

tères ont été abordés pour démontrer une approche globale du développement. La phase de test et de validation à la fin a confirmé la performance et la fiabilité de notre solution de tableau de bord connecté.

Conclusion générale

En conclusion, ce projet a permis de concevoir et de développer avec succès une solution connectée dédiée au tableau de bord pour motocycle à base d'un microcontrôleur ESP32-S3, répondant aux exigences croissantes d'une expérience de conduite interconnectée. En intégrant des technologies, telles que le BLE, pour une communication fluide avec le smartphone ainsi que la bibliothèque graphique LVGL pour une interface utilisateur riche et moderne affichée sur un écran TFT tactile, nous avons démontré la faisabilité d'une solution embarquée performante. La gestion intelligente des données du véhicule (vitesse, kilométrage), l'intégration des notifications et des instructions de navigation, ainsi que l'optimisation de la gestion du temps via la RTC en mode veille profonde, prouvent la polyvalence et l'efficacité de l'architecture mise en œuvre.

Les chapitres précédents ont exploré en détail les contraintes posées par ce système embarqué et la phase de son développement à l'aide d'un ensemble de composants, ainsi que les tests effectués pour vérifier sa performance et son fonctionnement.

En vue de l'amélioration du potentiel du système développé, il serait intéressant de penser à l'intégration de fonctionnalités supplémentaires. Par exemple, l'ajout d'une sortie audio permet d'enrichir les notifications visuelles par des sons distincts pour les appels entrants, les messages ou les alertes de navigation dans le but de minimiser la nécessité de regarder sur l'écran pour lire les informations. Parallèlement, l'intégration de la prise en charge de plusieurs langues pour l'interface utilisateur et les notifications serait une amélioration majeure qui rend le produit accessible à un public plus large. À plus long terme, l'intégration de services basés sur le cloud pour la télémétrie avancée ou la personnalisation des profils de conduite, pourrait transformer ce tableau de bord en une plateforme connectée plus évolutive.

Références bibliographiques

- [1] Z.T.C. Rakotoson. Conception et réalisation d'un tableau de bord automobile sur Arduino. Mémoire de Licence, École Supérieure Polytechnique d'Antananarivo. Disponible via : <https://www.polytechnique.mg/these/rakotosonZo-tahinaC-ESPA-Lic-23.pdf>. Consulté le 04 mai 2025.
- [2] Bluetooth Special Interest Group (SIG). Disponible sur <https://www.bluetooth.com/>. Consulté le 5 mai 2025.
- [3] M.Van Droogenbroeck and J.-M. Wagner. Bluetooth : technologie et potentiel industriel. Technical report, 2002. Disponible via : <https://orbi.uliege.be/bitstream/2268/117128/1/Vandroogenbroeck2002Bluetooth.pdf>. Consulté le 31 mai 2025.
- [4] P. Patras M. Cominelli, " in *Proc. 13th ACM Conf. Security Privacy in Wireless F. Gringoli, "Dead on Arrival: An Empirical Study of the Bluetooth 5.1 Positioning System, and Austria: ACM 2020. Mobile Networks (WiSec)*, Linz. Association for Computing Machinery. Disponible sur : https://www.pure.ed.ac.uk/ws/files/109261746/Dead_on_Arrival_COMINELLI_DoA220819_AFV.pdf. Consulté le 02 mai 2025.
- [5] R.Hedyn. Bluetooth Low Energy: The Developer's Handbook. Pretince Hall, 2012. Disponible sur : <https://file2.engineering.com/engcom/images/products/books/bluetooth-low-energy.pdf>. Consulté le 15 mai 2025.
- [6] M.Afaneh. Intro To Bluetooth Low Energy. Novel Bits, 2018. Disponible via : <https://novelbits.io/introduction-to-bluetooth-low-energy-book/>. Consulté le 03 mai 2025.

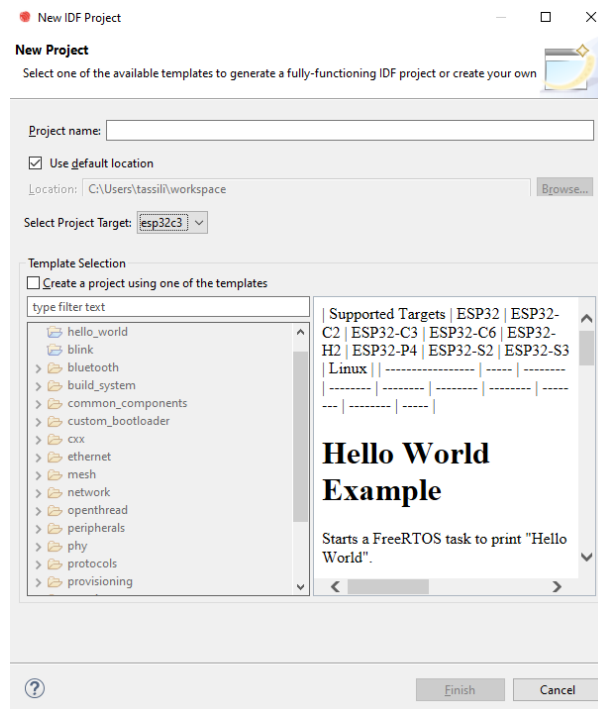
-
- [7] Lycée Raoul Dautry. Les microcontrôleurs – du microprocesseur à la carte de prototypage, partie 1 : microprocesseur et microcontrôleur. Disponible sur : <https://fr.scribd.com/document/698318991/QP1>. Consulté le 30 mai 2025.
- [8] Blaise Pascal Info. Les circuits intégrés – NSI. Disponible sur : <https://info.blaisepascal.fr/nsi-circuits-integres>. Consulté le 30 mai 2025.
- [9] E.B.A.Kiki-Sagbe. Ordonnancement ad hoc facilitant le saut rapide de fréquence avec des durées et à des moments pseudo-aléatoires, Décembre 2012. Disponible sur : https://espace.etsmtl.ca/id/eprint/1102/1/KIKI-SAGBE_Enagnon_Bill_Aurlus.pdf. Consulté le 01 juin 2025.
- [10] W.O.Galitz. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. 3ème édition. Disponible sur : https://profagaskar.wordpress.com/wp-content/uploads/2020/03/wiley_the_essential_guide_to_user_interf.pdf. Consulté le 05 juin 2025.
- [11] Continental Automotive.OLED Displays. Disponible sur <https://www.continental-automotive.com/en/solutions/user-experience/display-solutions/oled-displays.html>. Consulté le 09 juin 2025.
- [12] Reshine Display. Différence entre TFT et l'écran LCD. Disponible sur : <https://www.reshine-display.com/fr/what-is-the-difference-between-tft-and-lcd-display.html>. Consulté le 09 juin 2025.
- [13] N.Kolban. Kolban's Book on ESP32. Septembre 2018. Disponible sur : https://roberthart56.github.io/SCFAB/SC_lab/Electronics/Microcontrollers/ESP32/kolban-ESP32.pdf. Consulté le 17 mai 2025.
- [14] Puce ESP32-S3 : définition, brochage, processeur, application et carte de développement. Disponible sur <https://reversepcb.com/esp32-s3/>. Consulté le 09 juin 2025.

- [15] Espressif Systems. Espressif-ide documentation. <https://docs.espressif.com/projects/espressif-ide/en/latest/index.html>, 2025. Consulté le 21 avril 2025.
- [16] LVGL Project. LVGL Light and Versatile Graphics Library, 2025. Disponible sur <https://lvgl.io>. Consulté le 21 mai 2025.
- [17] EEZ Studio. Introduction, consulté le. Disponible sur <https://reversepcb.com/esp32-s3/>. Consulté le 10 juin 2025.
- [18] S.Janc. NimBLE portable Bluetooth stack from Apache Mynewt. CODECOUP, RIOT Summit, Amsterdam, 2018. Disponible sur : https://summit.riot-os.org/2018/wp-content/uploads/sites/10/2018/09/1_1-Szymon-Janc-NimBLE.pdf. Consulté le 10 juin 2025.
- [19] R.Barry. Mastering the FreeRTOS™ Real Time Kernel. Real Time Engineers Ltd. 2016. Disponible sur : https://www.freertos.org/media/2018/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf,. Consulté le 05 juin 2025.
- [20] ESP32 vs ESP8266, What's the Difference? Disponible sur <https://makeradvisor.com/esp32-vs-esp8266/>. Consulté le 30 juin 2025.
- [21] Nordic Semiconductor. nRF Connect. Disponible sur <https://www.nordicsemi.com>. Consulté le 14 juin 2025.

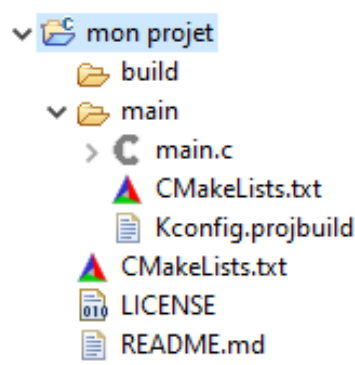
Annexes

1. Création d'un projet dans Espressif IDE

Sur la fenêtre principale du logiciel, on choisit **nouveau projet** et puis **nouveau projet Espressif IDF**, ensuite la fenêtre ci-dessous apparaît.

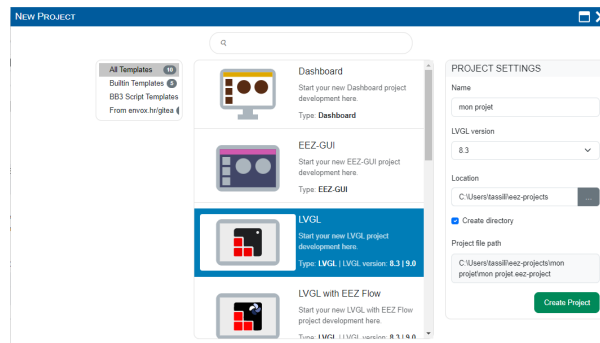


On choisit le chemin pour stocker le projet et on sélectionne le type de l'ESP32 à utiliser pour compiler le projet selon l'ESP32 sélectionnée. Il est possible de choisir une "Template" déjà existante pour ne pas faire à zéro, ou bien choisir un projet vide pour la personnalisation. Si on appuie sur "finish", le projet est créé et on aura les fichiers associés.



2. Création d'un projet dans EEZ Studio

Pour créer un nouveau projet dans EEZ Studio, on appuie sur **nouveau projet**.



On choisit LVGL vu que c'est la bibliothèque graphique utilisée pour ce projet. En insérant un nom au projet puis en appuyant sur **Create Project**, on aura un nouveau projet dans EEZ Studio.

3. Équivalents C de quelques Widgets LVGL

Les widgets utilisés dans LVGL sont compilés à un code C pour l'implémenter sur un microcontrôleur. Parmi ces widgets on cite:

Etiquette

```
{
    lv_obj_t *obj = lv_label_create(parent_obj); //créer l'étiquette
    lv_obj_set_pos(obj, 376, 275); //sa position
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_label_set_text(obj, "bonjour"); //texte sur
    l'étiquette
    lv_obj_set_style_text_color(obj, lv_color_hex(0xffffffff),
        LV_PART_MAIN | LV_STATE_DEFAULT); //style du texte
    lv_obj_set_style_text_font(obj, &lv_font_montserrat_24,
        LV_PART_MAIN | LV_STATE_DEFAULT); //font du texte
}
```

Image

```
{  
    lv_obj_t *obj = lv_img_create(parent_obj);  
    lv_obj_set_pos(obj, 363, 22);  
    lv_obj_set_size(obj, 98, 302);  
    lv_img_set_src(obj, &mon_image); //source de l'image  
    lv_obj_add_flag(obj, LV_OBJ_FLAG_ADV_HITTEST);  
    lv_obj_clear_flag(obj, LV_OBJ_FLAG_SCROLLABLE);  
}
```

Barre pour la distance restante

```
{  
  
    bar = lv_bar_create(parent_obj);  
    lv_obj_set_pos(bar, 131, 35);  
    lv_obj_set_size(bar, 539, 10);  
    lv_bar_set_value(bar, 25, LV_ANIM_OFF);  
    lv_obj_add_flag(bar, LV_OBJ_FLAG_CLICKABLE);  
    lv_obj_set_style_bg_color(bar, lv_color_hex(0xffdfdfd),  
        LV_PART_MAIN | LV_STATE_DEFAULT);  
    lv_obj_set_style_bg_color(bar, lv_color_hex(0xff21f325),  
        LV_PART_INDICATOR | LV_STATE_DEFAULT);  
}
```

4. Quelques commandes sur ESP IDF

L'ESP-IDF est le framework de développement d'Espressif Systems pour ses microcontrôleurs. Il inclut un système d'exploitation temps réel (FreeRTOS), des pilotes et des piles logicielles pour le Wi-Fi et le Bluetooth ou BLE.

- **idf.py menuconfig:** accéder au menuconfig.
- **idf.py build:** compiler le projet.
- **idf.py -p <PORT> flash:** Télécharger le firmware compilé sur l'ESP via le port série spécifié (<PORT>)).
- **idf.py clean:** Supprime tous les fichiers de sortie de la compilation pour une reconstruction complète lors du prochain build.