

UNE MINUTERIE COMMANDÉE PAR UNE APPLICATION ANDROID VIA BLUETOOTH CONTEXTE: PROJET IOT

Document Technique

OBJET:	Document décrivant la réalisation d'une minuterie commandée par une application Android via Bluetooth.
Auteurs:	Takwa FAKHFAKH Radhouene BELHADJ ALAYA
Encadrant académique:	Sonia BEN REJEB CHAOUCH
Groupe:	3 GTR
Année Universitaire:	2018/2019

CONTENU DU DOSSIER

- ▼ Etude Théorique
- ▼ Etude Pratique

Table des Matières

Introduction	4
Chapitre 1: Étude Théorique	5
Définition	5
Cas d'utilisations	5
Applications grand public	5
Maison intelligente	5
Soins aux personnes handicapées et âgées	5
Applications commerciales	5
Médecine et santé	5
Transport	6
Applications industrielles	6
Fabrication	6
Agriculture	6
Applications d'infrastructure	6
Déploiements à l'échelle métropolitaine	6
Gestion de l'énergie	6
Connectivité	6
La technique NarrowBand IoT	8
Les avantages de NB-IOT	9
Les inconvénients de NB-IOT	9
La technique LoRa	9
Les avantages de LoRa	9
Les inconvénients de LoRa	9
La technique Sigfox	10
Les avantages de Sigfox	10
Les inconvénients de Sigfox	10
Outils de simulations	10
Iotify	10
MATLAB	11
Netsim	11
BevyWise	11
Ansys IoT	12
IBM Bluemix	12
Packet Tracer 7	13
Conclusion	13
Chapitre 2: Étude Pratique	14
Introduction	14
Cahier de charge	14
Problématique	14
Solution	14

Matériels	14
Logiciel	14
Environnement de travail	15
Le bipeur	15
Le module Bluetooth	15
L'afficheur LCD1602	16
La carte STM32F4	17
La platine d'expérimentation	18
Réalisation	18
Câblage	18
Le câblage de tout le projet	18
Tableau de câblage de l'afficheur LCD1602	20
Tableau de câblage du bipeur	21
Tableau du module bluetooth HC-05	21
Code utilisé	21
La configuration de bus de communication UART4	21
La configuration de PWM1	23
La configuration de TIM4	24
Le main code	26
Conclusion	27

INTRODUCTION

L'Internet des Objets ou encore IoT, est de plus en plus présente dans notre vie. L'IoT se manifeste beaucoup à l'aide de capteurs qui servent à récupérer des données et des effecteurs. Les capteurs servent à prélever les données et les stocker dans une zone de stockage pour les utiliser. Les effecteurs sont les outils qui, à l'aide des données récupérées, modifieront l'environnement pour améliorer la vie des usagers.

Dans le cadre de notre matière, nous allons découvrir ce domaine à travers une étude théorique et une étude pratique.

La première partie intitulée “Étude théorique” s’articule autour de la définition de l’IOT, les cas d’utilisation, les techniques de connectivité et les outils de simulation.

La deuxième partie sous le nom “Étude pratique” présente un cas d'utilisation de la technique IOT avec la carte STM32F4.

CHAPITRE 1: ÉTUDE THÉORIQUE

1. Définition

L'Internet des objets (IoT) est un réseau d'appareils physiques, de véhicules, d'appareils électroménagers et d'autres éléments intégrant de l'électronique, des logiciels, des capteurs, des actionneurs et une connectivité qui permet à ces objets de se connecter, de collecter et d'échanger des données créant des possibilités d'intégration plus directe du monde physique dans des systèmes informatiques, entraînant des améliorations de l'efficacité, des avantages économiques et une réduction des efforts de l'homme [1].

2. Cas d'utilisations

L'utilisation de l' IOT est divisée en quatre concepts différents:

2.1 Applications grand public

2.1.1 Maison intelligente

Les appareils IoT font partie du concept d'automatisation de la maison, qui peut inclure des systèmes d'éclairage, de chauffage et de climatisation, et des systèmes de sécurité. Une maison automatisée pourrait être basée sur une plate-forme ou des concentrateurs contrôlant un ensemble des appareils intelligents.

2.1.2 Soins aux personnes handicapées et âgées

Ce sont des systèmes qui utilisent des techniques d'assistance aux personnes handicapées et aux personnes âgées afin d'assurer une meilleure qualité de vie.

Ce type d'application peut inclure des capteurs qui surveillent les urgences médicales telles que les chutes ou les convulsions.

2.2 Applications commerciales

2.2.1 Médecine et santé

Les dispositifs IoT peuvent être utilisés pour activer la surveillance de la santé à distance et les systèmes de notification d'urgence. Certains hôpitaux ont commencé à mettre en place des «smart beds» capables de détecter leur occupation et le moment où un patient tente de se lever. Ils peuvent également s'ajuster pour garantir une pression et un soutien appropriés au patient sans interaction manuelle des infirmières.

2.2.2 Transport

L'interaction dynamique entre les composants d'un système de transport permet la communication inter et intra-véhicule, le contrôle de la circulation, le stationnement intelligent, les systèmes de péage électronique, la gestion de la logistique et du parc, le contrôle des véhicules, la sécurité et l'assistance routière.

2.3 Applications industrielles

2.3.1 Fabrication

Les systèmes intelligents IoT permettent la fabrication rapide de nouveaux produits, une réponse dynamique à la demande des produits et l'optimisation en temps réel des réseaux de production, de fabrication et de chaîne logistique, en mettant en réseau des machines, des capteurs et des systèmes de contrôle.

2.3.2 Agriculture

Il existe de nombreuses applications d'IOT dans l'agriculture, telles que la collecte de données sur la température, l'humidité, la vitesse du vent... Ces données peuvent être utilisées pour automatiser les techniques agricoles, prendre des décisions éclairées pour améliorer la qualité et la quantité, minimiser les risques et le gaspillage et réduire les efforts nécessaires à la gestion de l'agriculture.

2.4 Applications d'infrastructure

2.4.1 Déploiements à l'échelle métropolitaine

Plusieurs déploiements à grande échelle de l'IOT permettent une meilleure gestion des villes et des systèmes. Par exemple, Songdo, en Corée du Sud, est la première ville intelligente: une grande partie de la ville est câblée et automatisée. , avec peu ou pas d'intervention humaine.

2.4.2 Gestion de l'énergie

Un nombre important d'appareils consommateurs d'énergie (commutateurs, prises de courant, ampoules, téléviseurs,...) intègrent déjà la connectivité Internet, ce qui leur permet de communiquer avec les services publics pour équilibrer la production et la consommation d'énergie, ainsi que pour l'optimiser.

2.4.3 Surveillance de l'environnement

Les applications de surveillance de l'environnement de l'IOT utilisent généralement des capteurs pour contribuer à la protection de l'environnement en surveillant la qualité de l'air ou de l'eau et les conditions atmosphériques.

3. Connectivité

La technologie IOT a une grande diversité au niveau de la connectivité.

Il y a plusieurs moyens pour connecter des objets IOT (Cellulaire, satellite, WiFi, Bluetooth, RFID, NFC, LPWAN et Ethernet) et dans chacun de ces moyens, il peut y avoir différents fournisseurs (par exemple, T-Mobile, Verizon, AT & T, Sprint, etc.) [5].

Ce tableau comparatif présente les différentes technologies de connectivité:

Protocole	Norme	Fréquence	Portée	Vitesse de transmission
Bluetooth	La spécification fondamentale de Bluetooth 4.2	2,4 GHz (ISM)	50-150 m (Smart/BLE)	1 Mbit/s (Smart/BLE)
Zigbee	ZigBee 3.0 basé sur IEEE802.15.4	2,4 GHz	10-100 m	250 Kbit/s
Z-Wave	Z-Wave Alliance ZAD12837/ITU-T G.9959	900MHz (ISM)	30 m	9,6 / 40 / 100 Kbit/s
6LowPAN	RFC6282	(adaptée et utilisée sur une variété d'autres médias de mise en réseau, dont Bluetooth Smart (2,4 GHz), ZigBee ou RF à faible consommation (Sub-GHz))	N/A	N/A
Thread	Thread, basée sur IEEE802.15.4 et 6LowPAN	2,4 GHz (ISM)	N/A	N/A
Wi-Fi	basée sur 802.11n (actuellement la norme la plus utilisée pour un usage privé)	bandes de 2,4 GHz et 5 GHz	environ 50 m	600 Mbit/s maximum, mais les vitesses habituelles sont plus proches de 150 Mbit/s, en fonction de la fréquence de canal utilisée et du nombre d'antennes (la dernière norme 802.11-ac devrait permettre des vitesses pouvant

				atteindre 500 Mbit/s à 1 Gbit/s)
Technologie cellulaire	GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G)	900 / 1 800 / 1 900 / 2 100 MHz	35 km max pour GSM ; 200 km max pour HSPA	35-170 Kbit/s (GPRS), 120-384 Kbit/s (EDGE), 384 Kbit/s-2 Mbit/s (UMTS), 600 Kbit/s-10 Mbit/s (HSPA), 3-10 Mbit/s (LTE)
NFC	ISO/CEI18000-3	13,56MHz (ISM)	10 cm	100–420 Kbit/s
Sigfox	Sigfox	900 MHz	30-50 km (environnements ruraux), 3-10 km (environnements urbains)	10-1 000 bit/s
Neul	Neul	900 MHz (ISM), 458 MHz (UK), 470-790 MHz (White Space)	10 km	de quelques bit/s à 100 Kbit/s
LoRaWAN	LoRaWAN	variable	2-5 km (environnement urbain), 15 km (environnement suburbain)	0,3-50 Kbit/s

Tableau 1: Comparaison entre les protocoles IOT [5]

3.1 La technique NarrowBand IoT

NB-IoT est l'initiative «clean sheet» du 3GPP (Third Generation Partnership Project), l'organisation à l'origine de la normalisation des systèmes cellulaires, pour répondre aux besoins des appareils à très faible débit de données qui doivent se connecter à des réseaux mobiles, souvent alimentés par piles.

En tant que norme cellulaire, l'objectif de NB-IoT est de standardiser les dispositifs IoT pour qu'ils soient interopérables et plus fiables.

NB-IoT est une technologie sans fil de niveau cellulaire qui utilise la modulation OFDM, les puces sont plus complexes, mais les budgets de liaison sont meilleurs. Cela signifie que les utilisateurs bénéficient du niveau de performance élevée associée aux connexions cellulaires, mais au prix d'une complexité et d'une consommation d'énergie accrues.

NB-IoT est conçu pour envoyer et recevoir de petites quantités de données, quelques dizaines ou centaines d'octets par jour générés par des périphériques IoT produisant peu de données. Il est basé sur les messages, similaire à Sigfox et LoRa, mais avec une vitesse de modulation beaucoup plus rapide qui peut gérer beaucoup plus de données que ces technologies. Cependant, NB-IoT n'est pas un protocole de communication basé sur IP tel que LTE-M (une autre technologie cellulaire LPWA associée aux applications IoT). Il consomme moins d'énergie que le LTE-M, mais il est conçu pour des communications moins fréquentes [3].

3.1.1. Les avantages de NB-IOT

Si NB-IOT existait et était déployé:

- ❑ La couverture serait très bonne. Les appareils NB-IoT reposent sur la couverture 4G, ils fonctionnent donc bien à l'intérieur et dans les zones urbaines denses.
- ❑ Ses temps de réponse sont plus rapides que ceux de LoRa et peuvent garantir une meilleure qualité de service [3].

3.1.2. Les inconvénients de NB-IOT

Il est difficile de mettre en œuvre des transferts de fichiers. Certaines spécifications de conception de NB-IoT permettent d'envoyer de plus grandes quantités de données à un périphérique [3].

3.2. La technique LoRa

LoRa est une technologie de modulation non cellulaire pour LoRaWAN. Les deux termes LoRa et LoRaWAN ne sont pas interchangeables. LoRaWAN est le protocole standard pour les communications WAN et LoRa est utilisé comme technologie de réseau étendu. .

LoRa est utilisé principalement avec deux manières:

- ❑ LoRaWAN, qui a été déployé principalement en Europe. Il a une très petite capacité de message, moins que 12 octets.
- ❑ Symphony Link est un système sans fil basé sur la technologie LoRa, conçu pour surmonter les limitations d'un système LoRaWAN. Il est souvent inclus dans les solutions réseau plus complexes LoRa (principalement aux États-Unis et au Canada). il est conçu pour des applications industrielles [3].

3.2.1. Les avantages de LoRa

- ❑ Il est parfait pour les applications à un seul bâtiment.
- ❑ LoRa est efficace dans le cas où, nous avons besoin d'une bidirectionnalité.
- ❑ Les périphériques LoRa fonctionnent bien lorsqu'ils sont en mouvement, ce qui les rend utiles pour le suivi des actifs en déplacement [3].

3.2.2. Les inconvénients de LoRa

- ❑ Débit de données est inférieur à celui de NB-IoT.

- ❑ Temps de latence est plus long que celui de NB-IoT [3].

3.3. La technique Sigfox

La technique Sigfox utilise des connexions à très faible bande passante. Elle propose des modules radios les moins chers. Sigfox est uniquement une liaison montante. Elle permet aussi de faire une communication de bout en bout [3].

3.3.1. Les avantages de Sigfox

- ❑ Faible consommation d'énergie.
- ❑ Il prend en charge une large zone de couverture dans les zones où il se trouve [3].

3.3.2. Les inconvénients de Sigfox

- ❑ La communication est mieux dirigée du terminal vers la station de base. Il a une fonctionnalité bidirectionnelle, mais sa capacité de la station de base au point final est limitée, et vous aurez un budget de liaison inférieur à celui en hausse.
- ❑ La mobilité est difficile avec les appareils Sigfox [3].

4. Outils de simulations

Cette section a pour objectif de présenter quelques outils de simulation utilisés dans IOT.

4.1 Iotify

Cet outil permet de simuler des installations IoT à grande échelle dans un laboratoire IoT virtuel. L'utilisateur a la possibilité de générer un trafic personnalisable à partir de milliers de points de terminaison virtuels et de tester la plateforme en termes d'échelle, de sécurité et de fiabilité afin d'identifier et de résoudre les problèmes avant de déployer le produit final. Ce dernier simule un trafic réseau important pour voir comment la latence du réseau affecte les performances globales du système [4].

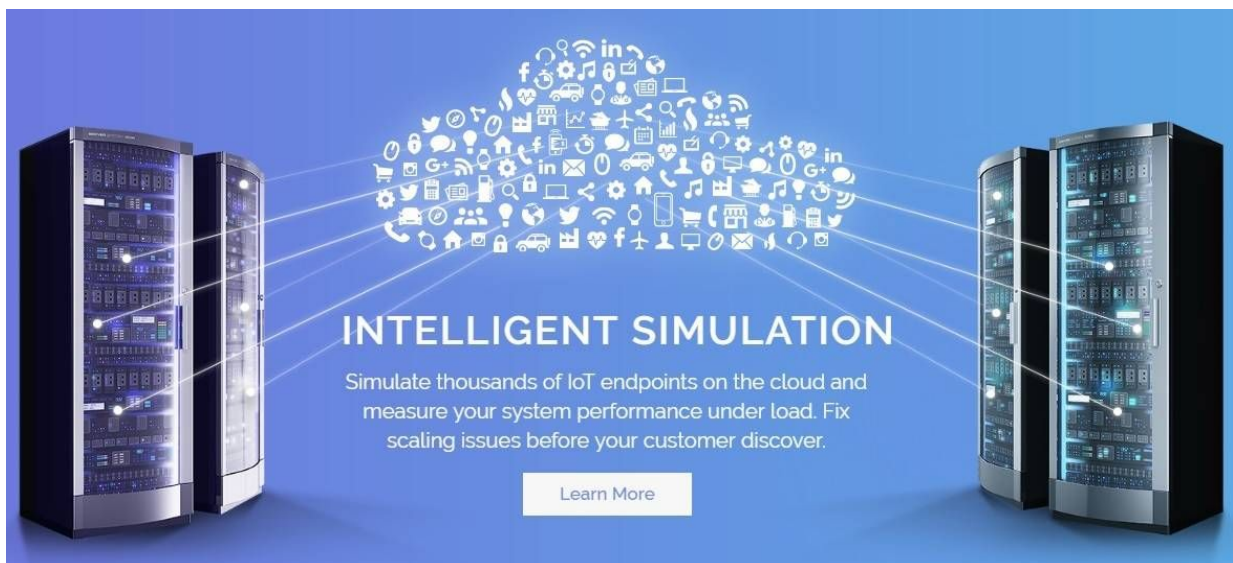


Figure 1: Iotify [4]

4.2 MATLAB

MATLAB propose un module IoT intéressant qui permet de développer et de tester des appareils intelligents, ainsi que de collecter et d'analyser des données IoT dans le cloud.

Les plateformes Iot collectent des données à partir d'appareils intelligents, les regroupent dans le cloud, puis les analysent en temps réel. Les modèles et les algorithmes sont ensuite extraits et les ingénieurs peuvent ensuite utiliser ces informations pour créer des algorithmes prototypes et les exécuter dans le cloud. A l'aide du MATLAB, il est possible de prototyper des appareils intelligents avec Arduino et Raspberry Pi [4].

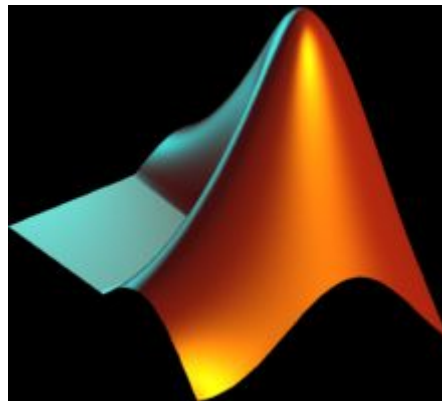


Figure 2: MATLAB [4]

4.3 Netsim

NetSim est capable de tester les performances d'applications réelles sur un réseau virtuel. Ce simulateur prend en charge plusieurs sources et destinations et peut être étendu à des centaines de nœuds. On peut simuler une grande variété de situations à l'aide des scénarios du type «What-if» et des métriques de test telles que la perte, les retards, les erreurs, la qualité de service, etc.[4].

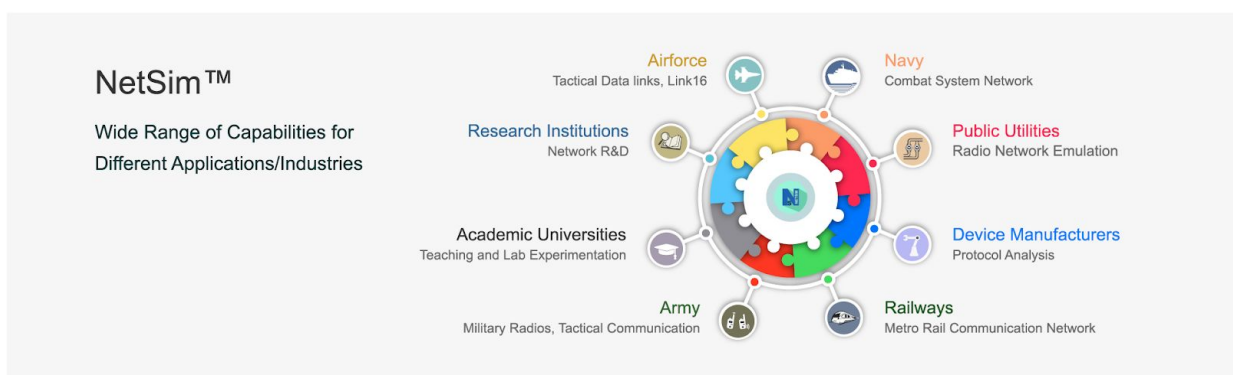


Figure 3: NetSim [4]

4.4 BevyWise

Cet outil a une interface utilisateur propre et puissante. Il permet de créer et d'ajouter les périphériques nécessaires en un temps réduit. La configuration des appareils IoT simulés de manière à publier des messages à une heure très précise. Ce dernier peut stocker des données de simulation dans des fichiers FLAT ou dans des bases de données MySQL et SQLite [4].



Figure 4: Bevywise [4]

4.5 Ansys IoT

Cet outil peut être utilisé dans une variété de domaines, y compris les équipements vestimentaires et médicaux, les drones, les voitures connectées, les équipements industriels, etc. Les solutions de simulation IoT d'Ansys aident à construire des appareils plus abordables et plus rentables [4].



Figure 5: Ansys IoT [4]

4.6 IBM Bluemix

Bluemix d'IBM est une plateforme cloud innovante avec des tableaux de bord intégrés à la console Web qui permettent de surveiller et d'analyser les données IoT simulées, puis de les utiliser pour créer et optimiser un ensemble d'applications. L'outil prend en charge une grande variété de fonctions permettant de manipuler, de stocker des données, d'interagir avec les médias sociaux [4].

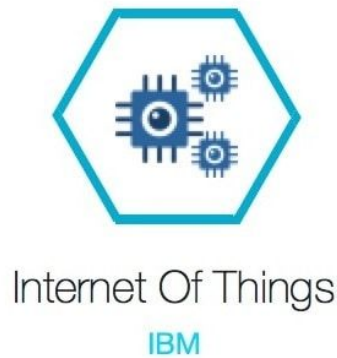


Figure 6: IBM Bluemix [4]

4.7 Packet Tracer 7

Packet Tracer est un simulateur de CISCO qui intègre dans sa nouvelle version de nouvelles fonctionnalités telles que de nouveaux appareils, IoT, capteurs et langages de programmation. Les nouveaux périphériques ajoutés pour permettre le travail dans un système IOT sont: Passerelle domestique IoT, Nouveau serveur «Internet of Everything» et «VM Management», Serveur d'enregistrement pour périphériques IoT, Périphériques IOE et capteurs de périphériques IoE: panneau solaire, wattmètre, voiture, passerelle domestique sans fil, wattmètre, détecteur de mouvement, capteur de température, capteur de convoyeur, etc. [9].

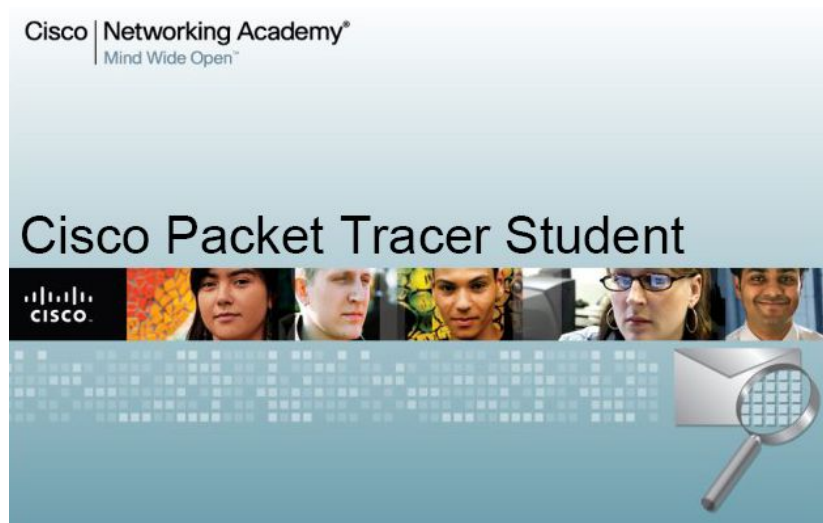


Figure 7: Packet Tracer [9]

Conclusion

Nous avons effectué dans ce chapitre une étude théorique des cas d'utilisation de l'IOT ainsi que citer les outils de simulation utilisés. Nous allons dans le chapitre suivant nous focaliser sur une étude pratique dans un domaine bien précis en citant les matériels et en détaillant le code source utilisé.

CHAPITRE 2: ÉTUDE PRATIQUE

1. Introduction

Nous vivons une époque marquée par une augmentation rapide et continue au niveau des technologies telles que les Smartphones et les Tablettes. Depuis leurs apparitions, les smartphones, sont devenus un véritable phénomène de société et le secteur ne cesse de se développer et de s'élargir. Ce nouveau support devient une nécessité qui fait partie de notre quotidien même dans le domaine de la domotique, et grâce à l'amélioration considérable des protocoles de communication sans fil, les maisons intelligentes sont la tendance et le thème le plus intéressant dans l'IOT.

2. Cahier de charge

Cette partie introductive présente quelques notions de base nécessaires à la compréhension de la suite du rapport. Ainsi, nous allons présenter quelques solutions existantes.

2.1. Problématique

De nos jours nous perdons trop d'énergie électrique car nous n'organisons pas l'utilisation de nos appareils électroménagers (climatiseur, micro-ondes,...). C'est dans ce cadre que se situe la problématique de notre projet.

2.2. Solution

Après une mûre réflexion, nous avons eu l'idée de notre cas d'utilisation qui accomplit ces objectifs. Nous avons essayé de concevoir une minuterie commandée par une application Android via le Bluetooth. Ce système peut être intégré dans une micro-onde, dans un système de climatisation etc...

Après avoir défini notre système, cette partie sera consacrée à la spécification du matériel et du logiciel utilisés.

2.2.1. Matériels

Dans notre système, nous allons utiliser une carte « Stm32F4Discovery » et nous allons afficher un message sur un afficheur LCD « 1602 ». L'alerte sonore sera diffusée sur un bipper à fréquence variable. Afin de contrôler notre système via une application Android nous allons utiliser le module Bluetooth « HC-05 ».

2.2.2. Logiciel

Nous allons utiliser le IAR Embedded Workbench pour la programmation et le débogage de la carte STM32F4.

3. Environnement de travail

Tout au long cette section nous allons décrire notre environnement de travail.

3.1. Le bipeur

Un bipeur (en anglais beeper ou buzzer) est un élément électromécanique qui produit un son quand on lui applique une tension. Certains nécessitent une tension continue, d'autres nécessitent une tension alternative [10].



Figure 8: Bipeur [10]

Dans notre cas, nous avons travaillé sur un bipeur à fréquence variable qui est activée par un signal d'horloge externe. Cette fréquence est généralement comprise entre 2 kHz et 4 kHz. Ce type de bipeur peut générer des bips graves, aigus, plus graves ou plus aigus. Il est cependant plus complexe à mettre en œuvre dans un circuit électronique car il faut générer le signal électrique de fréquence variable à leur application.

3.2. Le module Bluetooth

Bluetooth est une norme de technologie sans fil pour échanger des données sur de courtes distances (en utilisant les ondes radio UHF à courte longueur d'onde dans la bande ISM 2,4 à 2,485 GHz) à partir d'appareils fixes et mobiles, et pour la construction de réseaux personnels (PAN). La portée est d'environ 10 mètres [7].

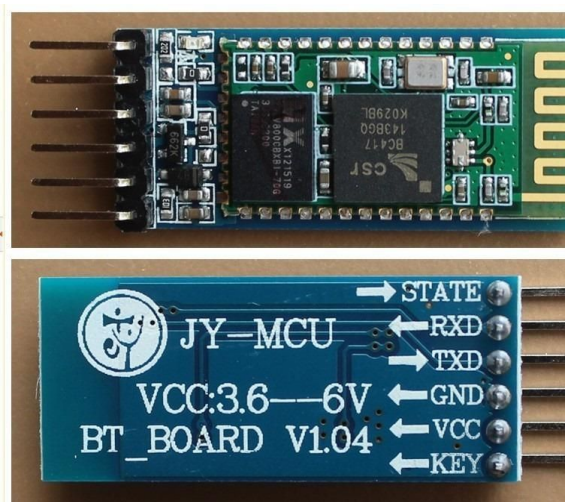


Figure 9: Module bluetooth HC-05 [7]

Ce module est conçu en deux faces :

- La face frontale de la carte comprend les puces radios et la mémoire, cristal de 26 MHz, l'antenne ou réseau d'adaptation. La partie droite comporte les broches de connexion pour l'alimentation et les signaux ainsi qu' un régulateur (de 5V à 3V), LED et décalage de niveau [7].

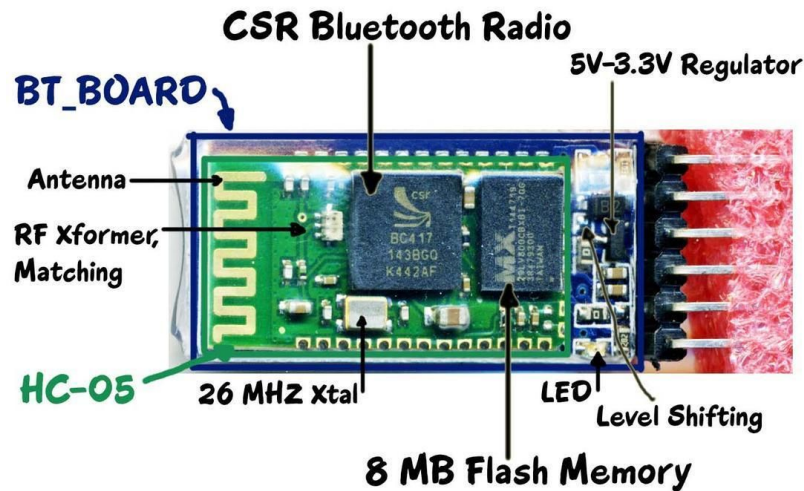


Figure 10: La face frontale de bluetooth HC-05 [7]

- Dans la deuxième face, nous trouverons les Pinouts de ce module:



Figure 11: Les pins de bluetooth HC-05 [7]

- ☐ KEY: est en état haut (1 logique) avant que l'alimentation soit appliquée, les forces sont en mode de commande d'installation. LED clignote lentement (2 secondes)
- ☐ VCC: +5V
- ☐ GND: 0V
- ☐ TXD: Transmission de données de HC-05 Vers la carte STM32F4
- ☐ RXD: Recevoir les données transmis par la carte STM32F4
- ☐ ETAT: Indique si le module est connecté à la carte STM32F4 ou non [7].

3.3. L'afficheur LCD1602

Nous avons choisi l'afficheur LCD1602 pour sa taille et son faible coût.



Figure 12: LCD1602 [8]

Ce tableau contient une description de chaque pin de cet afficheur.

Pin No.	Symbol	Description
1	VSS	Ground(0V).
2	VDD	Power supply for logic (+5V)
3	V0	Power supply for LCD driver
4	RS	Register Select Input: "High" for Data register (for read and write) "Low" for Instruction register (for write), Busy flag, address counter (for read)
5	R/W	Read/Write signal: " High" for Read mode. "Low" for Write mode.
6	E	Enable. Start signal for data read /write.
7	DB0	Data input/output (LSB)
8	DB1	Data input/output
9	DB2	Data input/output
10	DB3	Data input/output
11	DB4	Data input/output
12	DB5	Data input/output
13	DB6	Data input/output
14	DB7	Data input/output (MSB)
15	LED(+)	Anode of LED backlight
16	LED(-)	Cathode of LED backlight

Figure 13: Description de l'afficheur LCD1602 [8]

Pour afficher nos messages sur le LCD, nous avons utilisé la bibliothèque (MT WH1602). Cette bibliothèque nous permet d'utiliser des fonction simples pour afficher nos messages. Nous allons utiliser une pour écrire un caractère "MT WH1602 WriteData(code)" et une pour effacer tous les caractère "MT WH1602 ClearDisplay()".

3.4. La carte STM32F4

Nous avons utilisé la carte STM32F4 comme la partie commande pour notre système.

Cette petite carte, moins de 7 cm par 10 cm, est « propulsée » par un cortex M4 modèle STM32F407VGT6, 1 MB de ram et 192 KB de flash.

Dans cette carte, nous avons utilisé 11 Sorties pour l'afficheur LCD, 1 Sortie (PWM) pour le bipeur et 2 Sorties pour le module bluetooth HC-05.

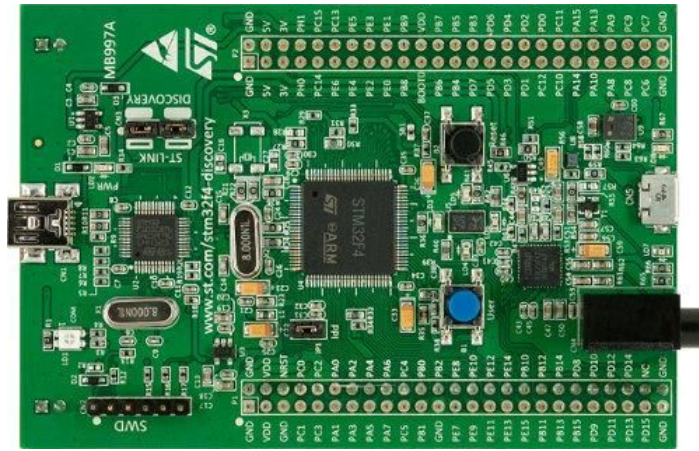


Figure 14: La carte STM32F4

3.5. La platine d'expérimentation

Un dispositif pour tester nos prototype de circuit électronique, c'est un système simple qui ne nécessite pas de soudure.

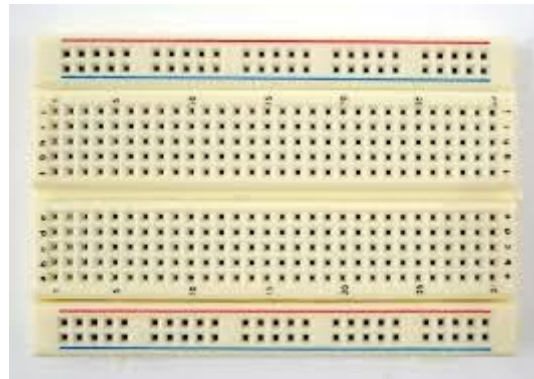


Figure 15: La platine d'expérimentation

4. Réalisation

Cette partie est consacrée pour la démonstration du travail réalisé. Nous commençons, tout d'abord, par l'aperçu du câblage des équipements. Ensuite, nous documentons le code utilisé pour développer notre application.

4.1. Câblage

4.1.1. Le câblage de tout le projet

Les figures 16, 17, 18 et 19 illustrent le câblage de notre projet.

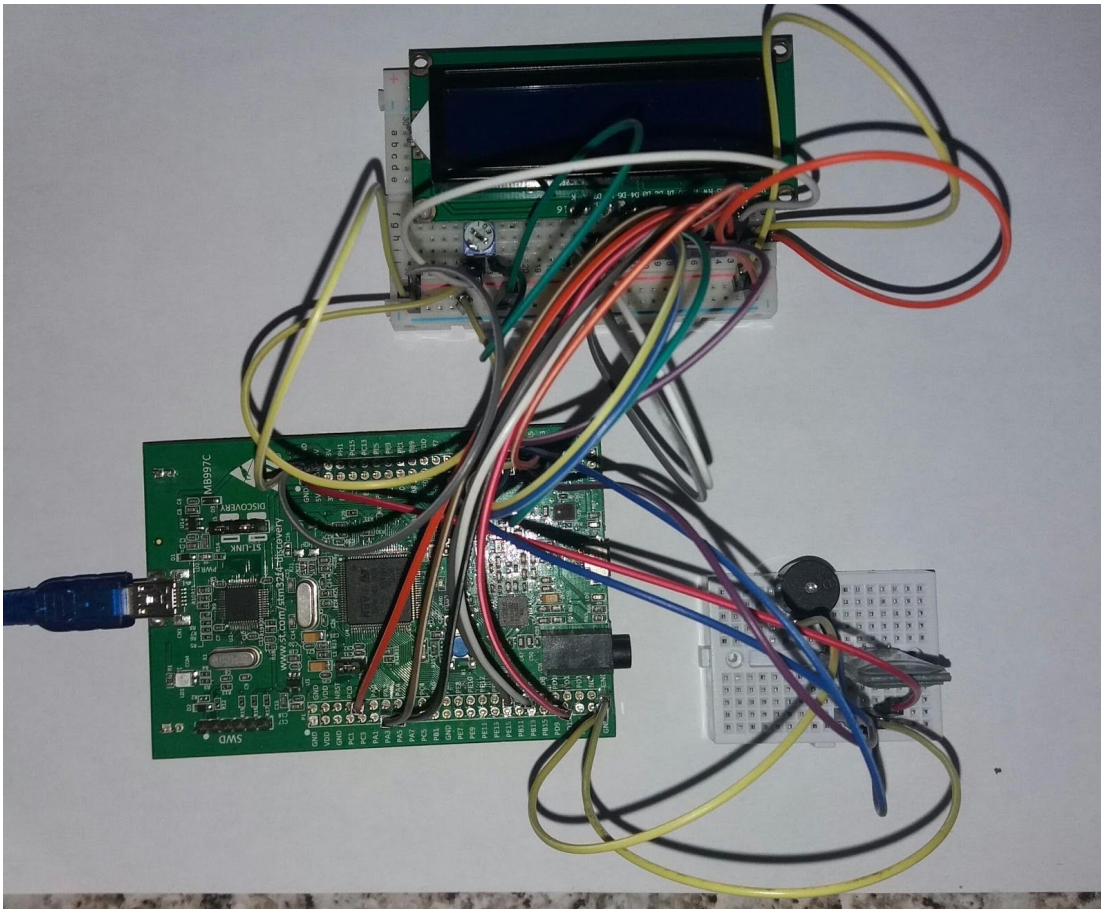


Figure 16: Le câblage de tout le projet

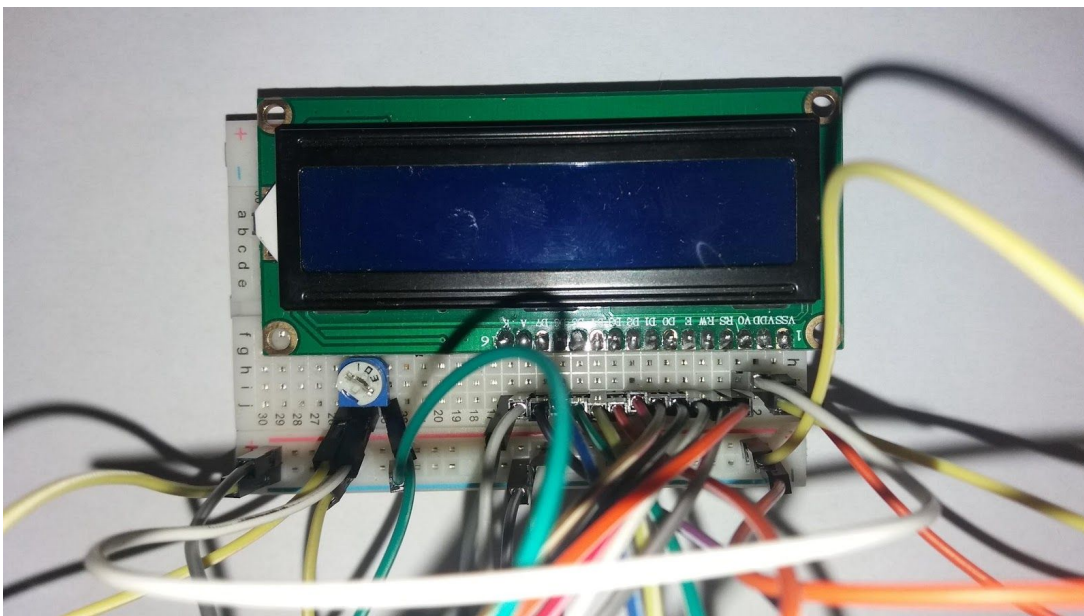


Figure 17: Le câblage de l'afficheur LCD1602

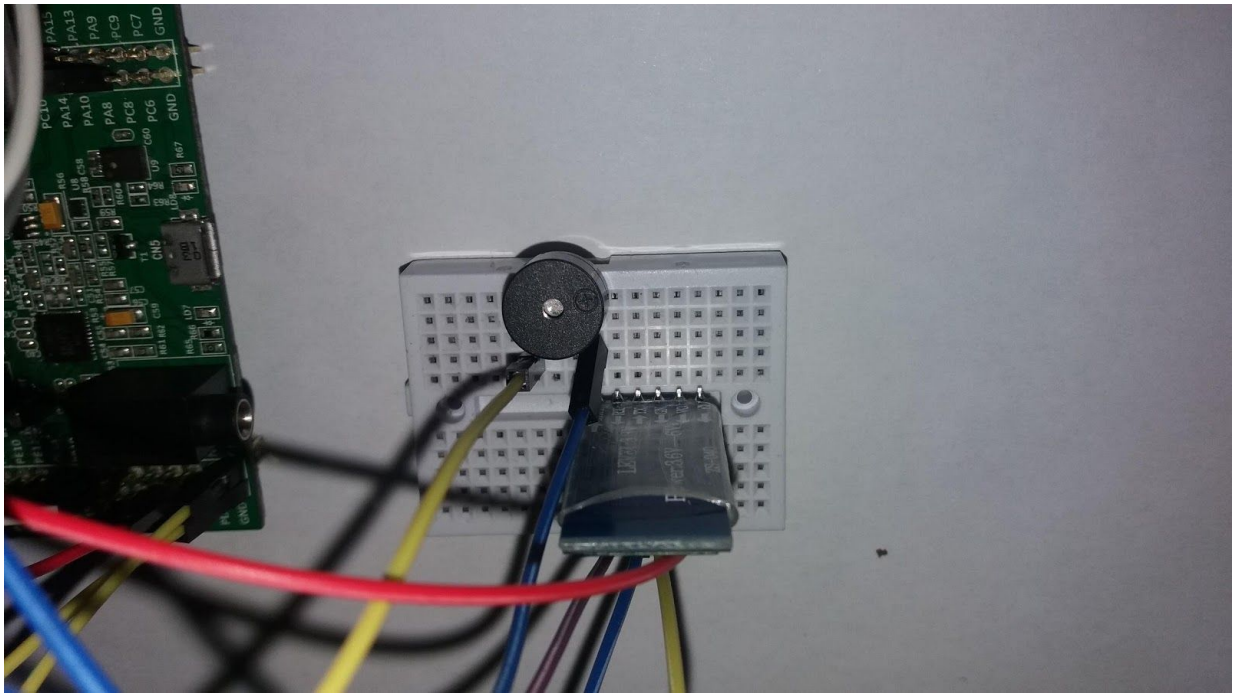


Figure 18: Le câblage du bipueur et le module bluetooth

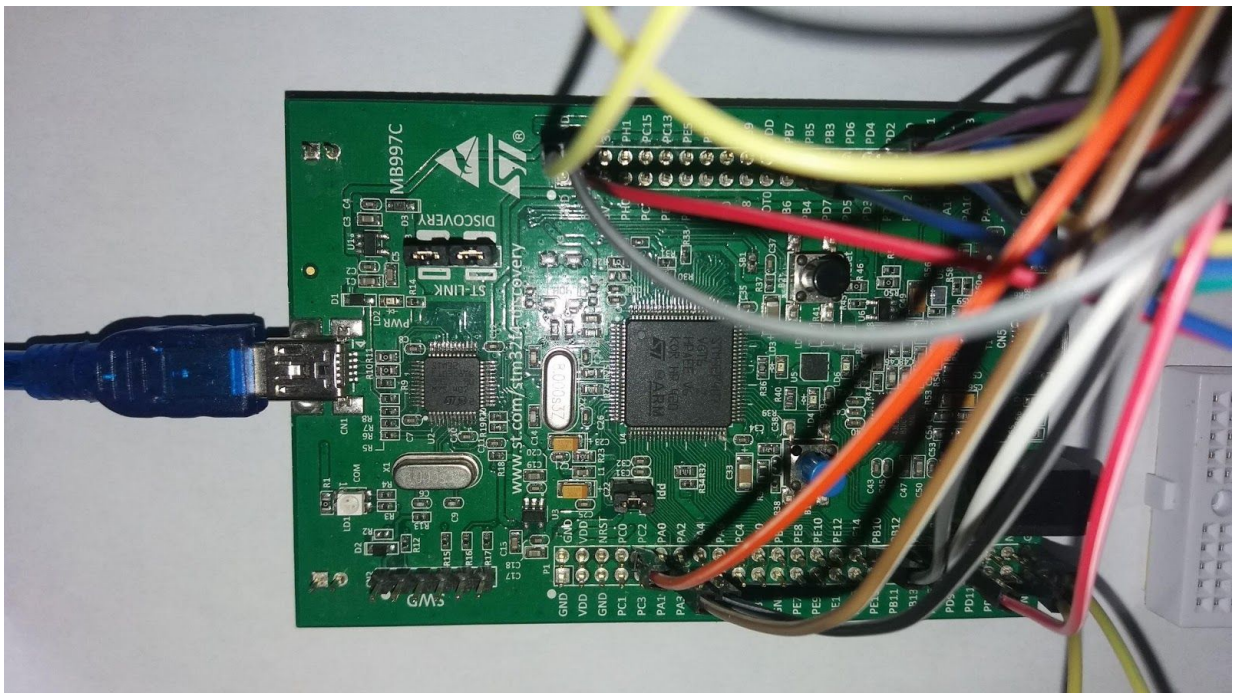


Figure 19: Le câblage de la carte STM32F4

4.1.2. Tableau de câblage de l'afficheur LCD1602

La figure 20 illustre le tableau de câblage de l'afficheur LCD1602.

Pin LCD	Pin STM32F4
VSS	GND
VDD	5V
RS	PC2
RW	PB10
E	PB10
D0	PA5
D1	PA3
D2	PD11
D3	PA15
D4	PA10
D5	PA8
D6	PC12
D7	PD2
A	5V
K	GND

Figure 20 : Tableau de câblage de l'afficheur LCD1602

4.1.3. Tableau de câblage du bipeur

La figure 21 présente le tableau de câblage du bipeur.

Pin bipeur	Pin STM32F4
La borne(+)	PB4
La borne(-)	GND

Figure 21: Tableau de câblage du bipeur

4.1.4. Tableau du module bluetooth HC-05

La figure 22 indique comment les pins du module Bluetooth HC-05 sont connectées avec la carte STM32F4.

Pin HC-05	Pin STM32F4
RX	PC10
TX	PC11
VCC	5V
GND	GND

Figure 22: Tableau de câblage du module bluetooth HC-05

4.2. Code utilisé

Durant cette section, nous allons documenter le code utilisé pour développer notre application.

4.2.1. La configuration de bus de communication UART4

Tout d'abord nous devons définir l'uart qui est un émetteur-récepteur asynchrone universel, la trame uart est constituée des bits suivants:

- Un bit de start qui est a 0
- Les bits de donnée (8 ou 9) bits
- Un bit de parité si ce dernier est activé

- Un ou deux bits de stop



Figure 23: Trame UART

La figure 24 présente la configuration de bus de communication UART4.

```

void GPIO_UART_Config(void)
{
    GPIO_InitTypeDef    GPIO_InitStructure;
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

    GPIO_PinAFConfig(GPIOC, GPIO_PinSource10, GPIO_AF_UART4);
    GPIO_PinAFConfig(GPIOC, GPIO_PinSource11, GPIO_AF_UART4);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}

void USART_Config(void)
{
    USART_InitTypeDef USART_InitStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_UART4, ENABLE);

    USART_InitStructure.USART_BaudRate = 9600;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Rx;
    USART_Init (UART4, &USART_InitStructure);
    USART_Cmd (UART4, ENABLE);
}

void NVIC_UART_Config(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_InitStructure.NVIC_IRQChannel = UART4_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

```

(1)

(2)

(3)

Figure 24: La configuration de bus de communication UART4

(1): La première chose est de configurer les ports à utiliser dans l'uart.

Nous avons utilisé 2 Port de la carte « C11, C10 », après nous avons activé le clock pour le bloc C et nous avons associé les pins à l'uart et au 3ème lieu nous avons configuré les pins.

(2): La deuxième étape est de configurer l'uart, nous activons tout d'abord le clock, la ligne suivante nous permet de définir le débit de transmission « 9600 », la longueur de la séquence de donnée en bit « 8 » et définir la longueur du bit stop « 1 » ainsi que l'activation du bit de parité « non activé ».

(3) : La 3ème partie c'est pour configurer l'interruption car nous devons interrompre le système à la réception des données.

Nous avons défini dans cette partie la priorité par rapport au groupe et au sous-groupe «0»

4.2.2. La configuration de PWM1

La figure 25 montre la configuration de PWM1.

```
void PWM_Test (void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_BaseStruct;
    TIM_OCInitTypeDef TIM_OCStruct;

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

    GPIO_PinAFConfig(GPIOB, GPIO_PinSource4, GPIO_AF_TIM3);

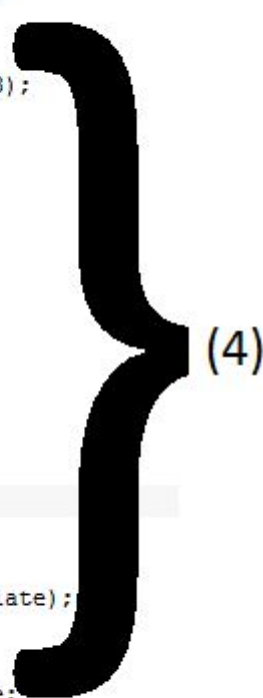
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    TIM_BaseStruct.TIM_Prescaler = 0;
    TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_BaseStruct.TIM_Period = 8000;
    TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_BaseStruct.TIM_RepetitionCounter = 0;

    TIM_TimeBaseInit(TIM3, &TIM_BaseStruct);
    TIM_PrescalerConfig(TIM3, 0, TIM_PSCReloadMode_Immediate);

    TIM_OCStruct.TIM_OCMode = TIM_OCMode_PWM2;
    TIM_OCStruct.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCStruct.TIM_OCPolarity = TIM_OCPolarity_Low;
    TIM_OCStruct.TIM_Pulse = 5000; //duty cycle
    TIM_OC1Init(TIM3, &TIM_OCStruct);    /// ocl chanel 1()
    TIM_OC1PreloadConfig(TIM3, TIM_OCPreload_Enable);

    TIM_Cmd(TIM3, ENABLE);
    TIM_CtrlPWMOutputs(TIM3, ENABLE);
}
```



(4)

Figure 25: La configuration de PWM1

(4): Pour pouvoir générer du son sur notre bipeur à fréquence variable, nous devons lui envoyer un signal pwm, c'est pour cela nous avons utilisé une pin pwm « B4 »: un timer pour cadencer ce bloc « Tim3 ». Au premier lieu, nous devons associer le pwm au timer.

Pour la configuration du timer: nous allons appliquer la formule suivante :

$$\text{Fréquence de sortie} = \text{Fréquence d'entrée} / (\text{Prescaler}+1) * (\text{Période}+1)$$

4.2.3. La configuration de TIM4

La figure 26 indique comment le TIM4 est configuré.

```
void TM_TIMER_Init(void) {
    NVIC_InitTypeDef  NVIC_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_BaseStruct;

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
    TIM_BaseStruct.TIM_Prescaler = 9999;

    TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up;

    TIM_BaseStruct.TIM_Period = 8399;
    TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_BaseStruct.TIM_RepetitionCounter = 0;

    TIM_TimeBaseInit(TIM4, &TIM_BaseStruct);
    /* Start count on TIM4 */
    TIM_Cmd(TIM4, ENABLE);

    TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE)

    NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

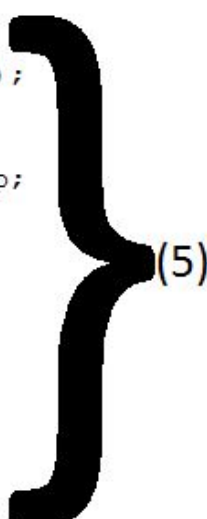


Figure 26: La configuration de TIM4 (1)

(5): Pour ce bloc, nous avons configuré le timer 4 qui va à chaque seconde interrompre le programme est faire une tâche comme le timer 3, la formule appliquée est:

$$\text{Fréquence de sortie(1HZ)} = \text{Fréquence d'entrée} / (\text{Prescaler}+1) * (\text{Période}+1)$$

- Fréquence de sortie = 1HZ
- Fréquence d'entrée = 84Mhz
- Prescaler =9999 et Période =8399


```

void TIM4_IRQHandler()
{
    if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)
    {
        TIM_ClearITPendingBit(TIM4, TIM_IT_Update
    }
    if (uart_off==1)
    {
        buzzer();
    }
    if (z==1)
    {
        if (y==0)
        {
            if (x==0) {z=0;}
            x--;
            y=59;
        }
        else
        { y--; }
        x1=(x/10)+0x30;
        x2=(x%10)+0x30;
        x3=(y/10)+0x30;
        x4=(y%10)+0x30;
    }
    else
    {
        x1=0x42;
        x2=0x4F;
        x3=0x4F;
        x4=0x4D;
        MT_WH1602_ClearDisplay();
        MT_WH1602_Delay(2000);
        MT_WH1602_WriteData(x1);
        MT_WH1602_Delay(100);
        MT_WH1602_WriteData(x2);
        MT_WH1602_Delay(100);
        MT_WH1602_WriteData(x2);
        MT_WH1602_Delay(100);
        MT_WH1602_WriteData(x3);
        MT_WH1602_Delay(100);
        MT_WH1602_WriteData(x4);
        MT_WH1602_Delay(100);
        TIM_ITConfig(TIM4, TIM_IT_Update, DISABLE);
        fin=0;
    }
}
}
}
}

```

(6)

(7)

(8)

(9)

Figure 27: La configuration de TIM4 (2)

(6): Pendant l'interruption du timer 4, nous avons exécuté une tâche qui est décrite en ce bloc est au bloc qui lui précède.

- (7): Tester s'il y a plus de donnée à recevoir => appeler la fonction buzzer() qui va afficher sur l'afficheur lcd le temps et faire fonctionner le bipleur.
- (8): Tester et décrémenter les variables x(secondes) et y (minutes).
- (9): Afficher la chaîne de caractères quand le temps écoule.

4.2.4. Le main code

La figure 28 présente le code utilisé dans la fonction main.

```
int main()
{
    GPIO_UART_Config();
    USART_Config();
    NVIC_UART_Config();
    MT_WH1602_Init();
    MT_WH1602_FunctionSet(1, 0, 0);
    MT_WH1602_Delay(1000);
    MT_WH1602_FunctionSet(1, 0, 0);
    MT_WH1602_Delay(1000);
    MT_WH1602_FunctionSet(1, 0, 0);
    MT_WH1602_Delay(1000);
    MT_WH1602_FunctionSet(1, 1, 1);
    MT_WH1602_Delay(1000);
    MT_WH1602_DisplayOnOff(1, 0, 0);
    MT_WH1602_Delay(1000);
    MT_WH1602_ClearDisplay();
    MT_WH1602_Delay(2000);
    TM_TIMER_Init();// init timer 4
    PWM_Test ();// init timer3 et pwm2
    while(1)
    {
        USART_ITConfig(UART4, USART_IT_RXNE, ENABLE);
        TIM3->CCR1=0;

        if (uart_off==1)
        {if(fin==1)
            {TIM3->CCR1=0;}
            else
            {TIM3->CCR1=5000;}}
    }
}
```

(10)

(11)

Figure 28: Le Contenu de la fonction main()

(10): Les fonction de configuration.

(11): Initialisation de l'afficheur LCD.

CONCLUSION

Ce système nous a permis de concevoir et de réaliser un projet orienté domotique servant à économiser l'énergie via l'organisation du temps du fonctionnement des électroménagers.

Ce projet était une bonne occasion pour toucher le domaine des systèmes embarqués grâce à notre étude pratique.

Du point de vue technique, ce projet nous a permis de nous adapter à l'environnement du développement des systèmes embarqués, de même il nous a permis d'enrichir nos connaissances dans l'IOT.

Tout au long de l'élaboration du projet, nous avons rencontré plusieurs difficultés tant au niveau réalisation qu'au niveau de l'étude théorique et nous avons réussi, malgré ces difficultés, de présenter une application opérationnelle.

Nous voulons signaler les larges perspectives offertes tels que le développement Android permet de sécuriser l'accès à notre module bluetooth avec le cryptage des données transférées entre la carte et l'application, l'enregistrement de l'historique des opérations sur un cloud, l'amélioration de notre système pour automatiser toute une maison intelligente.

Bibliographie

- [1]:<https://www.lemagit.fr/definition/Internet-des-objets-IoT>
- [2]:<https://www.citylab.com/life/2018/06/sleepy-in-songdo-koreas-smartest-city/561374/>
- [3]:<https://medium.com/iotforall/iot-connectivity-101-3f6fcee49a17>
- [4]:<https://windowsreport.com/iot-simulators/>
- [5]:<https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about-fr>
- [6]:<https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox>
- [7]:<https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To>
- [8]:http://www.geeetech.com/wiki/index.php/1602_LCD
- [9]:<https://www.imedita.com/blog/cisco-packet-tracer-7-0/>
- [10]:<http://www.composelec.com/bipeur.php>