Institut National des postes
et télécommunications Rabat
Filière Systèmes Embraqués
et Services Numériques

RAPPORT DE STAGE D'INITIATION

Effectué au sein de l'entreprise Molbiol Expert

Système de télésurveillance pour le suivi des patients atteints de maladies cardiaques

Présenté par:

- **FAHYM Abd Elfattah**

Enquadré par:

Mr.FAHYM Najim

Année de formation : 2021/2022

Dédicace

Je dédie ce travail :

A ma chère mère Mme Aych Naima et à mon cher père M. FAHYM Mohamed qui n'ont jamais cessé de me supporter, me soutenir et m'encourager durant mes années d'études. Qu ils trouvent ici le témoignage de ma profonde gratitude et Teconnaissance

A mes frères, mes grands-parents et ma famille qui me donnent de l'amour et de la vivacité.

A tous ceur qui m'ont aidé - de près ou de loin - et ceur qui ont partagé avec moi les moments d'émotion lors de la réalisation de ce travail et qui m'ont chaleureusement supporté et encouragé tout au long de mon parcours.

A tous mes amis qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

Merci!

FAHYM Abd Elfattah

Remerciement

En tout premier lieu, je remercie le bon Dieu, tout puissant, de m'avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés.

En effet il est difficile de remercier les personnes qui vous aident à mener à bien les tâches qui vous sont assignées, cependant je dois exprimer toute la gratitude que je ressens à leur égard.

Je souhaite saisir cette occasion pour remercier vivement Mme. GUENNOUNE Fatima la directrice du entreprise MolbiolExpert, qui m'a ouvert les portes de cette société d'excellence pour une durée de quatre semaine, afin d'effectuer mon stage d'initiation.

Je tiens à remercier mon Encadrant M. FAHYM Najim qui m'a enseigné dans plusieurs reprises durant la période du stage, et qui m'a donnée énormément d'information sur le domaine de biomédical.

Je ne saurais oublier le corps professoral de l'École Nationale des Postes et Télécommunications Rabat pour la formation d'excellence qui nous donne.

Merci!

Abstract

This internship report at Molbiol Expert begins with a general introduction followed by a contextualization, a problem statement, and objectives. The host organization, MolbiolExpert, is then presented along with its partners, positions, and activities. The operating principle of the telemonitoring system is also described, including a prototype and a global architecture. The necessary equipment and tools, such as NodeMcu (ESP8266), pulse oximeter MAX30102, OLED screen SSD1306, and AWS IoT Core, are presented. The report also details the electrical circuits and wiring diagrams, as well as the implementation of the telemonitoring system. The connection of ESP8266 to AWS IoT Core and programming for data transmission are explained.

Résumé

Ce rapport de stage chez Molbiol Expert commence par une introduction générale suivie d'une mise en contexte, une problématique et des objectifs. L'organisme d'accueil, MolbiolExpert, est ensuite présenté avec ses partenaires, ses postes et ses activités. Le principe de fonctionnement du système de télésurveillance est également décrit, avec un prototype et une architecture globale. Le matériel et l'outillage nécessaires, tels que le NodeMcu (ESP8266), l'oxymètre de pouls MAX30102, l'écran OLED SSD1306 et AWS IOT Core, sont présentés. Le rapport détaille également les circuits électriques et les schémas de câblage, ainsi que la réalisation du système de télésurveillance. La connexion de l'ESP8266 à AWS IoT Core et la programmation pour la transmission des données sont expliquées.

Contents

Table des figures

Introduction générale

Les maladies chroniques sont responsables de 65% des décès et demeurent la cause principale de tous les décès prématurés, où les maladies du cœur occupent la deuxième place. Les personnes atteintes de maladies chroniques ont besoin d'un suivi régulier de leurs états de santé. Souvent, elles ne sont pas en mesure de recevoir les soins nécessaires pour plusieurs raisons. Avec l'avènement de la technologie des télécommunications et d'internet, les systèmes de soins de santé se sont constamment améliorés avec le temps et de nombreux pays ont réalisé des gains importants d'espérance de vie de leurs populations. La télésurveillance s'est avérée capable d'améliorer considérablement les résultats du traitement de nombreuses maladies chroniques.

Notre projet vise à mettre en place un système d'e-santé basé sur la télésurveillance des signes vitaux de personnes souffrants de maladies cardiaques. Des capteurs connectés permettent de recueillir les signaux (ECG, PPG) et de transmettre les données vers un serveur puis l'utilisation d'un smartphone pour la visualisation. Ces données sont également accessibles à partir d'une plateforme web et peuvent être consultées en même temps par un médecin ou un autre membre du personnel médical. Notre système permettra d'éviter certaines complications de leur état de santé par la prise en charge immédiate, de contribuer à la rapidité de fournir un diagnostic ou un traitement et d'envoyer des alertes par SMS en cas d'arythmie ou de niveau anormal de SpO2.

Dans le cadre de mon stage au sein de l'association MolbiolExpert, j'ai pour objectif de réaliser un prototype de système de télésurveillance pour le suivi des patients atteints de maladies cardiaques. Pour cela, j'utiliserai le langage C avec l'IDE Arduino pour mettre en œuvre un système de mesure de la saturation d'oxygène en temps réel. Ce système sera basé sur un microcontrôleur ESP8266 relié à un capteur MAX30102 (Oxymètre de pouls MAX30102) et un afficheur OLED.

I- Mise en contexte, problématique et objectifs :

i) Problématique :

La mise en place d'un système de télésurveillance pour le suivi des patients atteints de maladies cardiaques répond à plusieurs problématiques. Tout d'abord, la disponibilité du personnel médical dans les centres de santé est souvent insuffisante, une situation qui s'est aggravée avec la pandémie de la Covid-19. Par ailleurs, les patients vivant dans des zones reculées rencontrent des difficultés pour accéder aux soins de santé, d'où l'importance d'un système de suivi à distance. En outre, la surveillance en temps réel des constantes vitales des patients est souvent négligée ou absente, alors que cela constitue un besoin essentiel. Enfin, la demande croissante de dispositifs médicaux portables et fiables pour une utilisation à domicile renforce la pertinence d'un système de télésurveillance convivial et continu pour surveiller les paramètres cliniques vitaux des patients atteints de maladies cardiaques.

ii) Objectifs :

Ce projet vise à améliorer la qualité des soins offerts aux personnes souffrant de maladies cardiaques en mettant en place un système d'E-santé basé sur la télésurveillance des signes vitaux. Ce système permettra un suivi en temps-réel, ce qui contribuera à la rapidité de fournir un diagnostic précis en cas de situation anormale d'un de ces signes vitaux. L'objectif principal de ce système est donc de prévenir les complications en détectant rapidement les signes vitaux anormaux et d'offrir une intervention médicale appropriée. Cela aidera également à réduire les coûts des soins de santé en évitant les consultations inutiles et en offrant un suivi continu et personnalisé. En somme, la mise en place de ce système de télésurveillance des signes vitaux contribuera à une meilleure prise en charge des patients atteints de maladies cardiaques, en permettant un suivi plus précis, une intervention rapide en cas de besoin et en prévenant les complications graves.

II- Présentation du l'organisme d'accueil et le service accueillant :

i) MolbiolExpert :

MolbiolExpert est une entreprise spécialisée dans le domaine de la biologie et offre une gamme complète de services de pointe dans le domaine du diagnostic moléculaire, des puces à ADN, du séquençage haut débit, de la chimie, de l'immunoanalyse et de l'anatomie pathologique. Selon le responsable de MolbiolExpert, l'entreprise a été créée dans le but de s'épanouir dans le domaine du vivant et d'étendre son expertise à tous les aspects de la biologie.

Les fondateurs de MolbiolExpert ont une solide expérience universitaire et professionnelle dans le domaine de la génétique humaine, du marketing et de la vente de produits biologiques pour le Maghreb. Ils ont suivi un parcours universitaire PhD en Génétique Humaine et ont ensuite travaillé pendant 16 ans dans le secteur des ventes, du marketing et de la gestion des applications pour le diagnostic moléculaire, les puces à ADN et le séquençage haut débit pour le Maghreb. Ils ont également étendu leur expertise dans différents domaines de la biologie tels que la chimie, l'immunoanalyse et l'anatomie pathologique. Ces expériences leur ont permis d'acquérir une expertise approfondie et de bien répondre aux besoins des clients.

MolbiolExpert se distingue également par sa connaissance approfondie des produits, qui lui permet de sélectionner les meilleurs partenaires et produits. Les dirigeants de l'entreprise ont pour passion et engagement de relever les défis avec des solutions de pointe répondant aux besoins de leur marché, en offrant une qualité de prestation de service optimale, en respectant les délais de livraison, en étant réactifs en cas d'urgence, en fournissant un support scientifique et en offrant un service après-vente irréprochable.

En somme, l'équipe de MolbiolExpert est motivée et passionnée par le travail qu'elle accomplit. Elle est fière de pouvoir s'appuyer sur ses clients fantastiques, ses partenaires solides et son équipe formidable pour continuer à offrir des services de qualité dans le domaine de la biologie .

ii) Les partenaires de Molbiol-Expert :

Les partenaires de Molbiol-Expert sont toutes des acteurs clés dans le domaine de la biologie moléculaire et du diagnostic, et elles peuvent donc être des partenaires de MolbiolExpert dans différents aspects de son activité, tels que la recherche, la production ou la commercialisation de produits et services.

Human : Une entreprise spécialisée dans la production d'anticorps, de réactifs et d'équipements de laboratoire pour la recherche et le diagnostic.

Tib Molbio : Une entreprise qui propose des tests moléculaires pour le diagnostic d'infections bactériennes, virales et fongiques.

R-Biopharm : Une entreprise qui fournit des kits de diagnostic pour la détection d'agents pathogènes, de contaminants alimentaires et de résidus de médicaments.

Machery-Nagel : Une entreprise qui fabrique des colonnes de chromatographie, des filtres, des pipettes et d'autres équipements de laboratoire.

ABCAM : Une entreprise spécialisée dans la production d'anticorps et de réactifs pour la recherche biomédicale et le diagnostic.
Heal Force : Une entreprise qui fournit des équipements de laboratoire, tels que des centrifugeuses, des incubateurs et des hottes de sécurité.

Dominique Dutscher : Une entreprise qui distribue des réactifs, des consommables et des équipements de laboratoire pour la recherche et le diagnostic.

iii) Arborescence des postes dans molbiol-expert :

iv) Les activités de molbiol-expert :

L'entreprise se spécialise dans l'importation, l'exportation, la distribution et la maintenance de dispositifs médicaux et de réactifs à usage de diagnostic in vitro. Elle propose également l'importation, la distribution et la commercialisation de matériel de laboratoire et médical, ainsi que de fournitures pour les laboratoires d'analyses médicales, comme des matériels, consommables, produits chimiques, produits biologiques et réactifs.

Molbiol-expert offre également des services de conseil, de consultation, d'expertise et d'assistance technique et scientifique réglementaire dans les domaines liés au matériel

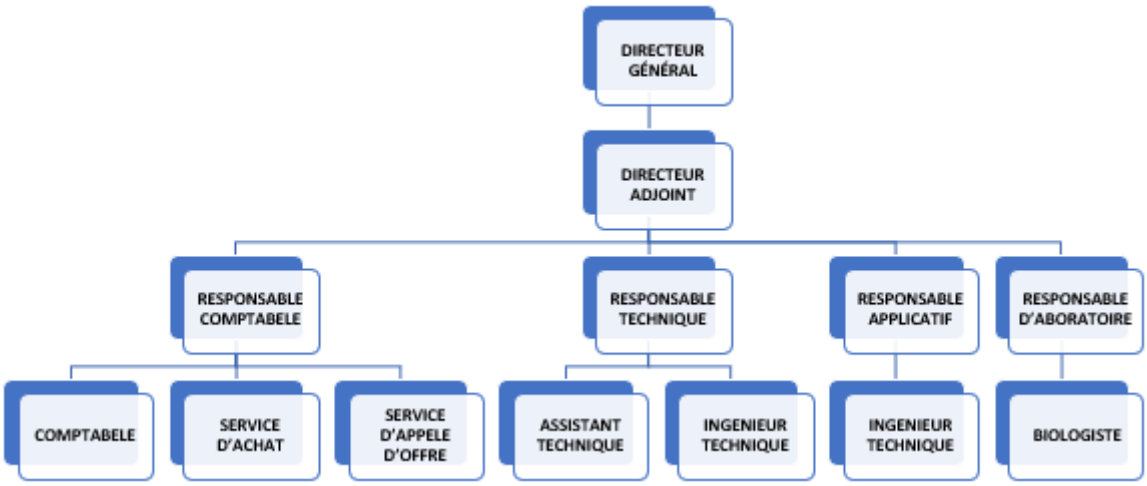


FIGURE 0.1: Arborescence des postes dans molbiol-expert

médical. Elle représente également le service médical d'assistance technique, et réalise des audits et des inspections.

Enfin, l'entreprise est également impliquée dans l'importation, l'exportation, l'achat et la vente ferme ou à la commission de divers produits et articles. Avec son expertise, ses produits et ses services, Molbiol-expert est une entreprise complète dans le domaine médical et de laboratoire, offrant une grande variété de solutions pour les professionnels de ce secteur.

v) Conclusion :

MolbiolExpert est une entreprise spécialisée dans le domaine de la biologie qui propose une large gamme de services de pointe dans les domaines du diagnostic moléculaire, des puces à ADN, du séquençage haut débit, de la chimie, de l'immunoanalyse et de l'anatomie pathologique. L'entreprise a été fondée par des experts universitaires en génétique humaine qui ont travaillé pendant 16 ans dans le secteur des ventes, du marketing et de la gestion des applications pour le diagnostic moléculaire, les puces à ADN et le séquençage haut débit pour le Maghreb. Ils ont également étendu leur expertise dans d'autres domaines de la biologie tels que la chimie, l'immunoanalyse et l'anatomie pathologique. Les partenaires de MolbiolExpert sont des acteurs clés dans le domaine de la biologie moléculaire et du diagnostic, et elles peuvent donc être des partenaires de MolbiolExpert dans différents aspects de son activité, tels que la recherche, la production ou la commercialisation de produits et services.
MolbiolExpert propose des services d'importation, d'exportation, de distribution et de maintenance de dispositifs médicaux et de réactifs à usage de diagnostic in vitro, ainsi que l'importation, la distribution et la commercialisation de matériel de laboratoire et médical. L'entreprise propose également des fournitures pour les laboratoires d'analyses médicales, comme des matériels, consommables, produits chimiques, produits biologiques et réactifs. MolbiolExpert offre également des services de conseil, de consultation, d'expertise et d'assistance technique et scientifique réglementaire dans les domaines liés au matériel médical. Elle représente également le service médical d'assistance technique, et réalise des audits et des inspections.

III- Principe de fonctionnement du système de télésurveillance :

i) prototype :

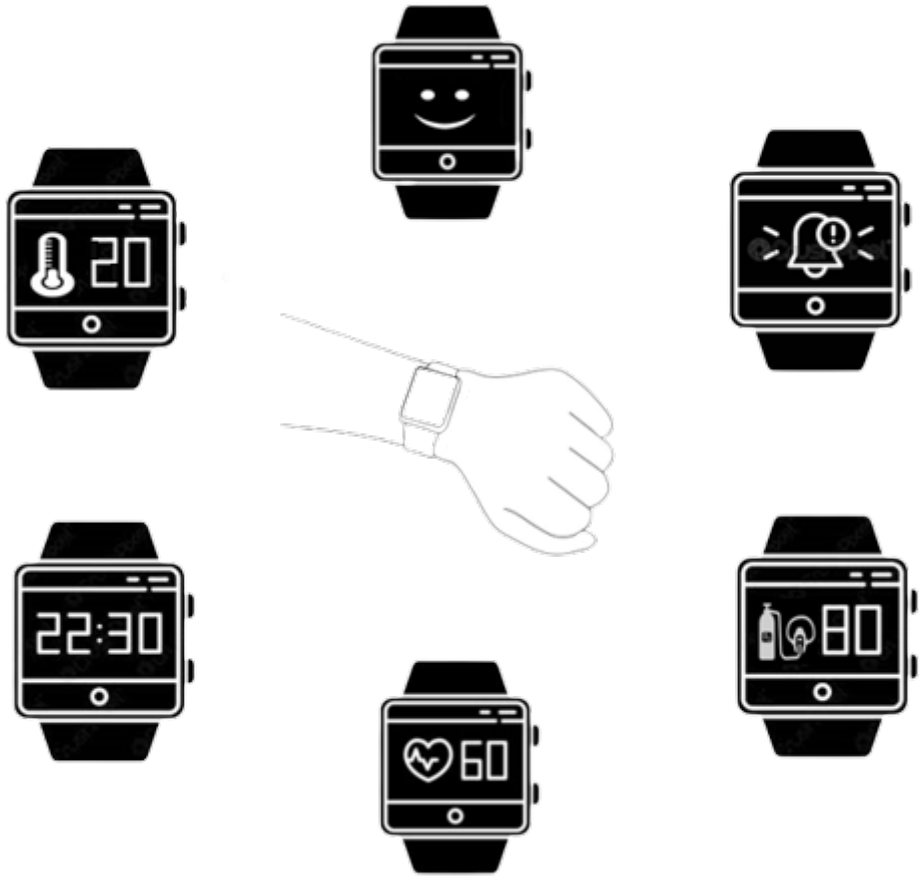


FIGURE 0.2: prototype du système de télésurveillance

ii) Architecture globale du système de télésurveillance :

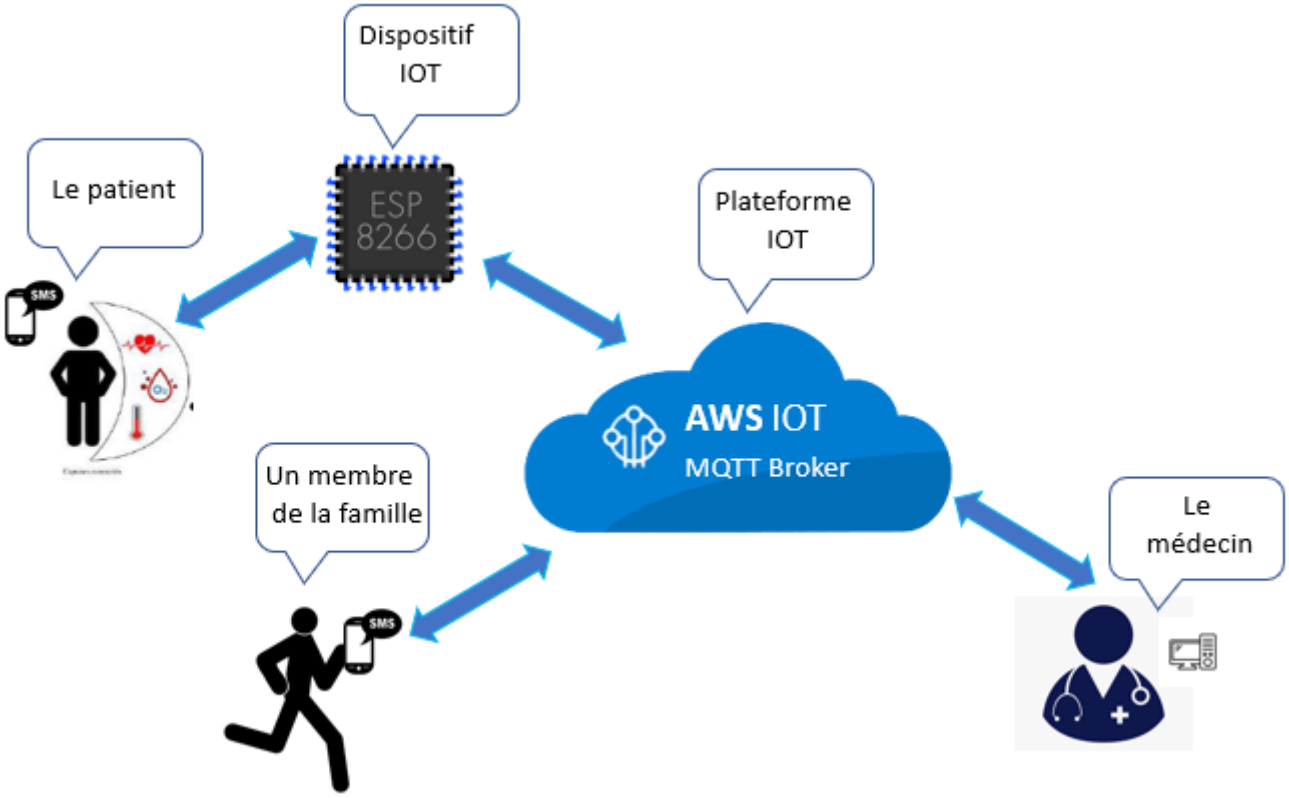
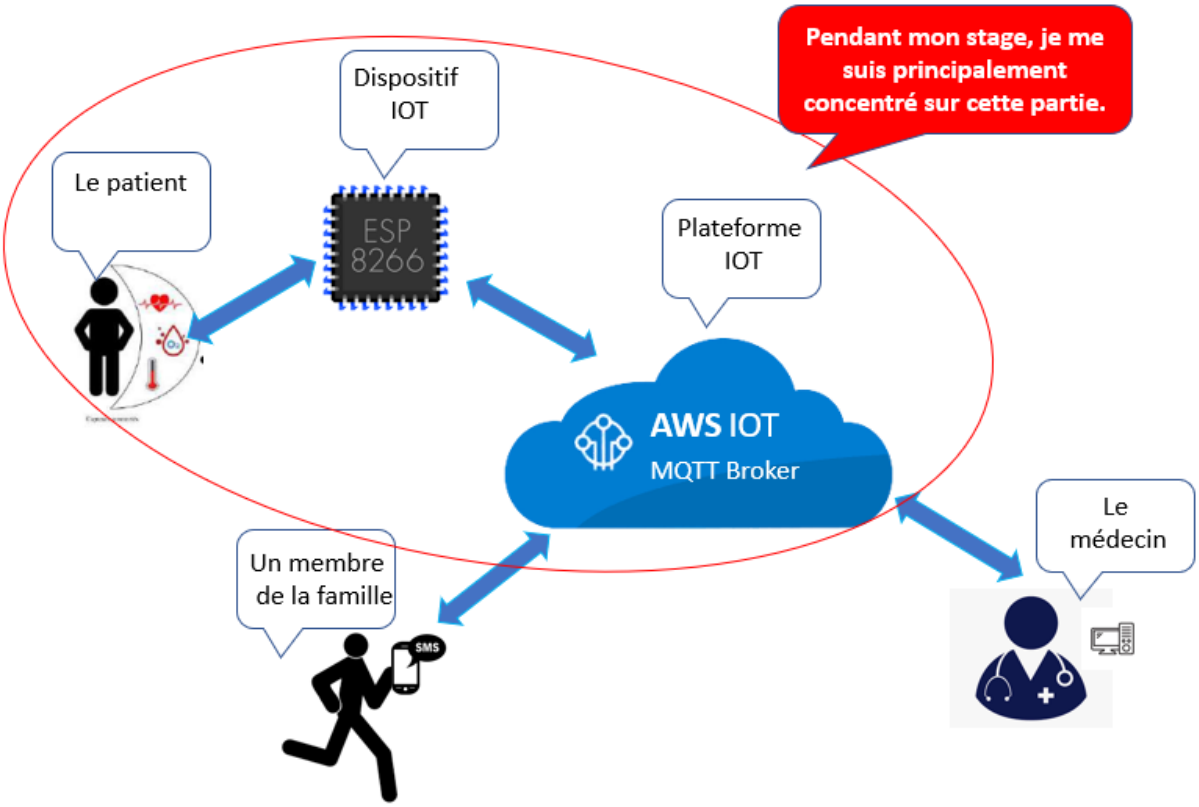


FIGURE 0.3: Architecture complète du système de télésurveillance

iii) Description du fonctionnement du système :

Le principe de fonctionnement du système de télésurveillance pour le suivi des patients atteints de maladies cardiaques repose sur l'utilisation d'un dispositif qui mesure la fréquence cardiaque (BPM), la température et la saturation en oxygène (SpO2) du patient. Ce dispositif prend la forme d'une « smartwatch » qui se compose d'un microcontrôleur de type esp8266 et d'un afficheur OLED SSD1306 I2C. Le capteur Max30105 doit être attaché à la main du patient.

Les données mesurées sont envoyées au serveur AWS IoT Core via WiFi. À chaque fois qu'un danger est détecté, le patient reçoit des notifications sur sa smartwatch. Si la situation du patient est très dangereuse, un membre de sa famille reçoit un appel ou un message pour sauver le patient. De plus, le médecin du patient peut suivre en temps réel la situation de son patient.



IV- Matérial et outillage :

i) NodeMcu (ESP8266) :

ESP8266 est le nom d'un tristement célèbre module WiFi qui est un système sur puce (SoC) développé par Espressif Systems, une société basée à Shanghai. Utilisé à l'origine avec les cartes Arduino pour les projets matériels compatibles WiFi, il est rapidement devenu une carte de développement autonome bon marché compatible Arduino. Il peut fonctionner en toute autonomie, sans microcontrôleur supplémentaire comme la carte Arduino par exemple.

Domotique et IOT – connecter des appareils à un réseau est une grande tendance en constante évolution de nos jours. Compte tenu de son prix bon marché, de sa configuration conviviale et de son énorme communauté qui contribue aux bibliothèques et aux projets open source, vous comprendrez immédiatement pourquoi cette puce suscite autant d'intérêt.

Ce microcontrôleur (MCU) peut être utilisé pour contrôler et surveiller des systèmes et des produits d'ingénierie, l'enregistrement des données des capteurs et plus encore. Tout cela en fait le matériel idéal pour les projets de domotique connectée. Il se présente sous de nombreuses formes, la NodeMcu (avec la dernière puce ESP8266-E12) étant la carte de développement la plus populaire d'entre elles. Image : La carte de développement NodeMcu est la variante ESP8266 la plus populaire.
Toutes les variantes de l'ESP8266 disposent d'un processeur principal ESP8266EX et d'un microcontrôleur Tensilica L106 32 bits. Il s'agit d'un SoC (System-On-Chip) à faible coût, haute performance, faible consommation d'énergie, facile à programmer, sans fil. Il fournit des capacités pour le Wi-Fi 2,4 GHz (802.11 b / g / n, prenant en charge WPA / WPA2), l'entrée / sortie à usage général (13 GPIO), le circuit inter-intégré (I^2C), la conversion analogique-numérique (CAN 10 bits), l'interface périphérique série (SPI), les interfaces I^2S avec DMA (partage de broches avec GPIO), UART (sur des broches dédiées, plus un UART en transmission seule peut être activé sur GPIO2) et la modulation de largeur

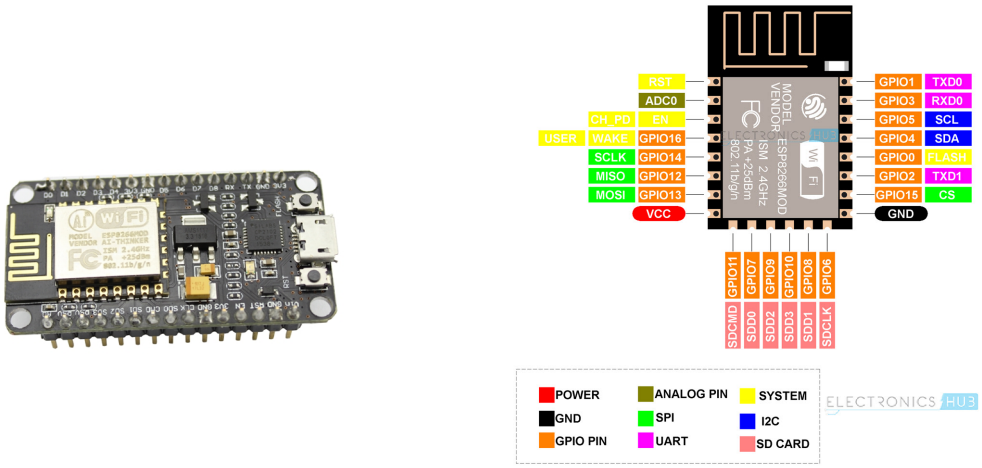


FIGURE 0.4: NodeMcu ou ESP8266

d'impulsion (PWM).

Il dispose d'un programmateur intégré et d'un régulateur de tension, qui permettent de clignoter et d'alimenter l'appareil via micro-USB. Le système fonctionne à 3,3 V.

ii) Oxymètre de pouls MAX30102 :

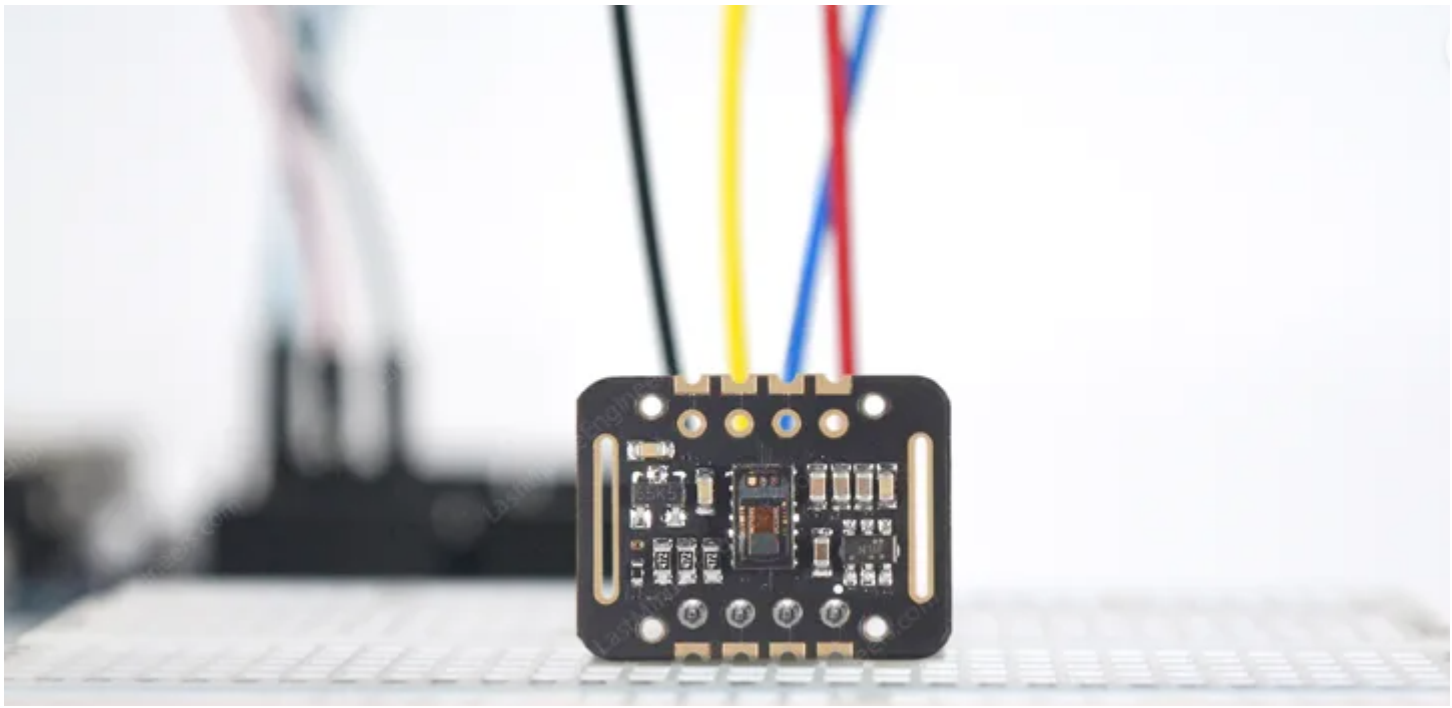
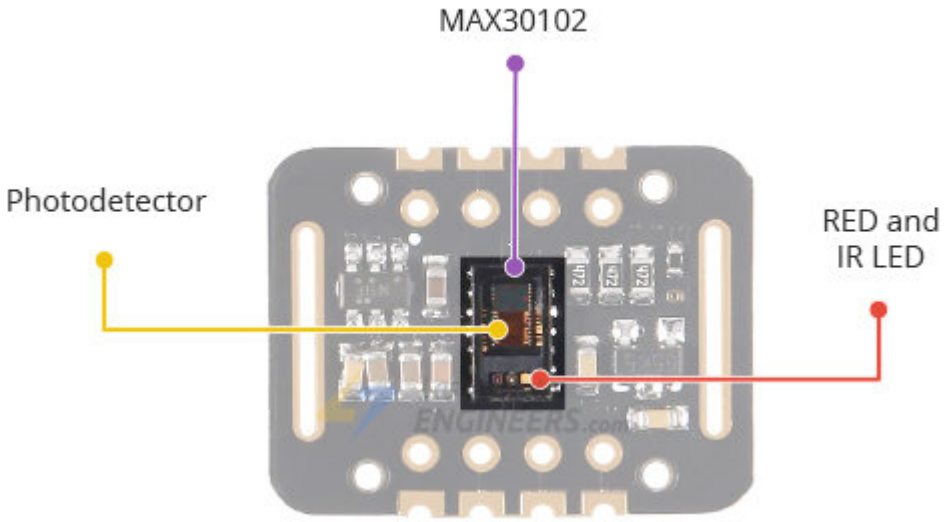


FIGURE 0.5: L'oxymètre de pouls et capteur de fréquence cardiaque MAX30102

L'oxymètre de pouls et capteur de fréquence cardiaque MAX30102 est un capteur biométrique plug-and-play basse consommation basé sur I2C. Il peut être utilisé par les étudiants, les amateurs, les ingénieurs, les fabricants et les développeurs de jeux et mobiles qui souhaitent intégrer des données de fréquence cardiaque en direct dans leurs projets.

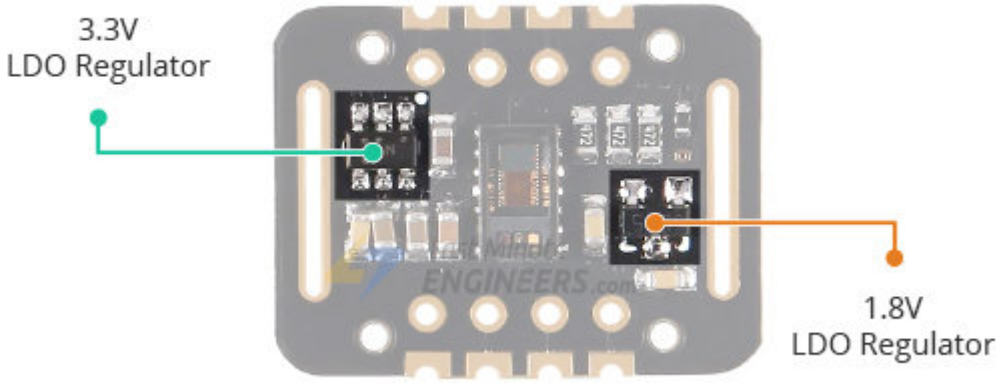
- # Présentation matérielle du module MAX30102

Le module est équipé du MAX30102 – un circuit intégré moderne (successeur du MAX30100) et d'oxymètre de pouls et de capteur de fréquence cardiaque d'Analog Devices. Il combine deux LED, un photodétecteur, une optique optimisée et un traitement du signal analogique à faible bruit pour détecter les signaux d'oxymétrie de pouls (SpO2) et de fréquence cardiaque (HR).
Derrière la fenêtre d'un côté, le MAX30102 dispose de deux LED – une LED ROUGE et une LED IR. De l'autre côté se trouve un photodétecteur très sensible. L'idée est que vous faites briller une seule LED à la fois, détectant la quantité de lumière qui rallume sur le détecteur, et, en fonction de la signature, vous pouvez mesurer le niveau d'oxygène dans le sang et la fréquence cardiaque.

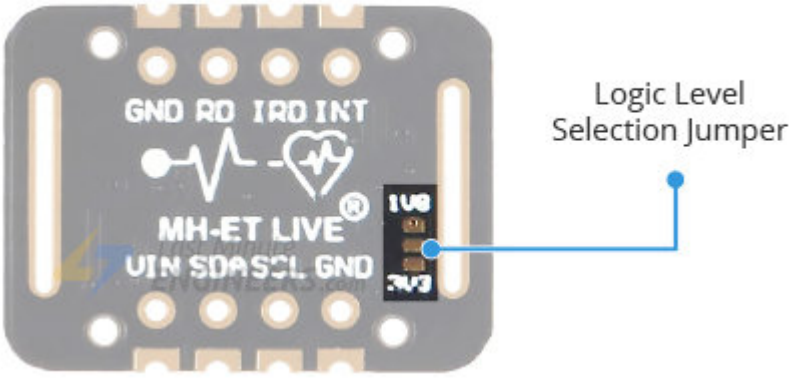


- Puissance requise

La puce MAX30102 nécessite deux tensions d'alimentation différentes : 1,8 V pour le circuit intégré et 3,3 V pour les LED RED et IR. Le module est donc livré avec des régulateurs 3.3V et 1.8V. À l'arrière du circuit imprimé, vous trouverez un cavalier de

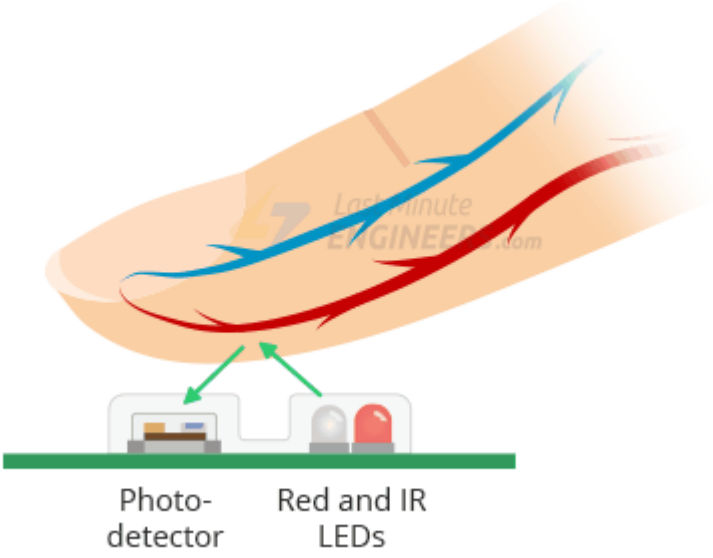


soudure qui peut être utilisé pour sélectionner entre le niveau logique 3.3V et 1.8V. Par défaut, le niveau logique 3.3V est sélectionné, ce qui est compatible avec les niveaux logiques pour Arduino. Mais vous pouvez également sélectionner le niveau logique 1.8V selon vos besoins. Cela vous permet de connecter le module à n'importe quel microcontrôleur avec des E/S de niveau 5V, 3,3 V et même 1,8 V. L'une des caractéristiques les plus importantes du MAX30102 est sa faible consommation d'énergie : le MAX30102 consomme moins de 600 uA pendant la mesure. Il est également possible de mettre le MAX30102 en mode veille, où il ne consomme que 0,7 uA. Cette faible consommation d'énergie permet une mise en œuvre dans des appareils alimentés par batterie tels que les combinés, les appareils portables ou les montres intelligentes.



- ## Comment fonctionnent l'oxymètre de pouls MAX30102 et le capteur de fréquence cardiaque ?

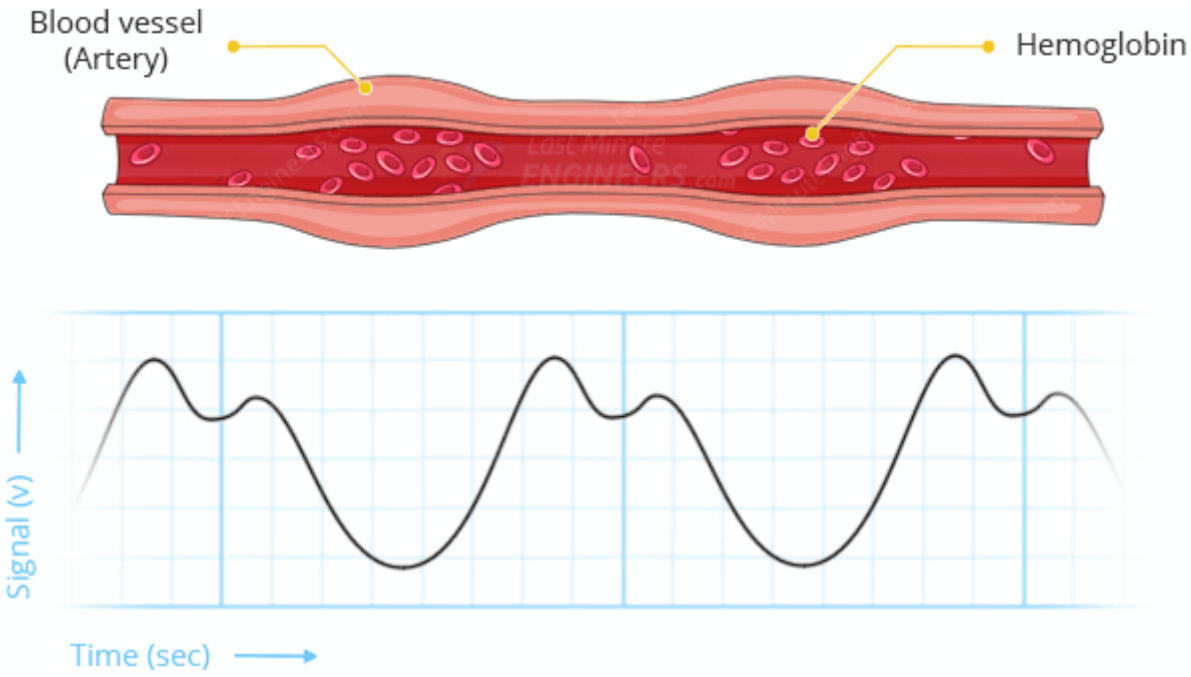
Le MAX30102, ou tout oxymètre optique de pouls et capteur de fréquence cardiaque d'ailleurs, se compose d'une paire de LED de haute intensité (RED et IR, toutes deux de longueurs d'onde différentes) et d'un photodétecteur. Les longueurs d'onde de ces LED sont respectivement de 660 nm et 880 nm.
Le MAX30102 fonctionne en projetant les deux lumières sur le doigt ou le lobe de l'oreille



(ou pratiquement partout où la peau n'est pas trop épaisse, de sorte que les deux lumières peuvent facilement pénétrer dans le tissu) et en mesurant la quantité de lumière réfléchie à l'aide d'un photodétecteur. Cette méthode de détection d'impulsions par la lumière est appelée photopléthysmogramme.
Le fonctionnement du MAX30102 peut être divisé en deux parties : la mesure de la fréquence cardiaque et l'oxymétrie de pouls (mesure du niveau d'oxygène du sang).

- ## Mesure de la fréquence cardiaque :

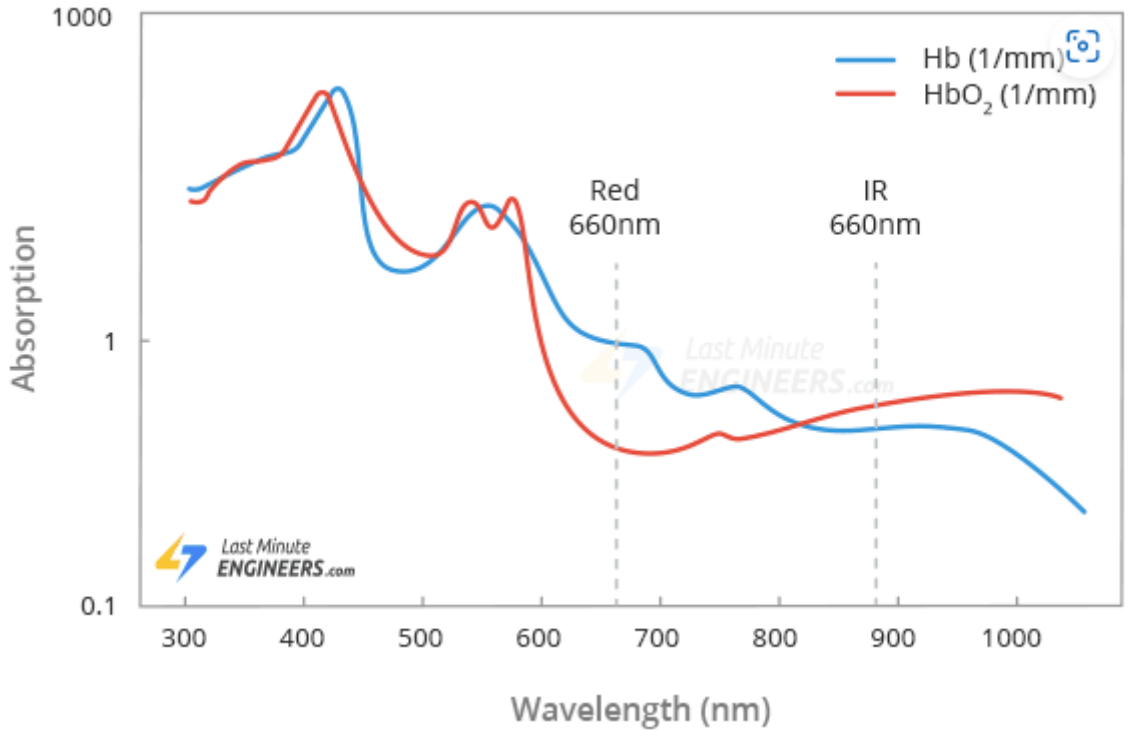
L'hémoglobine oxygénée (HbO2) dans le sang artériel a la caractéristique d'absorber la lumière IR. Plus le sang est rouge (plus l'hémoglobine est élevée), plus la lumière IR est absorbée. Lorsque le sang est pompé à travers le doigt à chaque battement de cœur, la quantité de lumière réfléchie change, créant une forme d'onde changeante à la sortie du photodétecteur. Au fur et à mesure que vous continuez à faire briller la lumière et à



prendre des lectures de photodétecteurs, vous commencez rapidement à obtenir une lecture du pouls du rythme cardiaque (HR).

- ## Oxymétrie de pouls :

L'oxymétrie de pouls est basée sur le principe que la quantité de lumière ROUGE et IR absorbée varie en fonction de la quantité d'oxygène dans votre sang. Le graphique suivant représente le spectre d'absorption de l'hémoglobine oxygénée (HbO2) et de l'hémoglobine désoxygénée (Hb). Comme vous pouvez le voir sur le graphique, le sang désoxygéné



absorbe plus de lumière ROUGE (660 nm), tandis que le sang oxygéné absorbe plus de lumière IR (880 nm). En mesurant le rapport entre la lumière IR et la lumière ROUGE reçue par le photodétecteur, le niveau d'oxygène (SpO2) dans le sang est calculé.

iii) SSD1306 OLED display :

L'écran SSD1306 contient une puce driver du même nom (SSD1306), il peut communiquer avec le périphérique maître (microcontrôleur, microprocesseur...) via le protocole I2C, le protocole SPI ou le protocole parallèle 8 bits. Dans cette rubrique, je vais montrer comment utiliser les protocoles I2C et SPI avec cet écran. Le protocole I2C n'a besoin que de 2 lignes : SDA (données série) et SCK (horloge série), une ligne supplémentaire est nécessaire qui est une ligne de réinitialisation (RST). Le protocole SPI est plus rapide que le protocole I2C mais il utilise plus de broches : SCK, SDA, CS (chip select : active low), D/C (data/command) et une broche de repos (RST).
Le SSD1306 OLED que j'ai utilisé est montré ci-dessous (vue arrière), le mode par défaut est SPI qui peut être changé en I2C en retirant les résistances R3 et en plaçant les résistances R1 et R8 (comme écrit sur la carte). Notez que la résistance de R1 = R3 = R8 = 0 ohm.

FIGURE 0.6: L'écran SSD1306

iv) AWS IOT core plateforme :

AWS IoT Core est une plateforme cloud fournie par Amazon Web Services qui permet aux appareils de se connecter de manière sécurisée au cloud et d'interagir avec d'autres appareils et services. Elle fournit un ensemble de services gérés pour la création d'applications IoT, tels que l'enregistrement d'appareils, la communication sécurisée et le traitement de données.

Avec AWS IoT Core, vous pouvez connecter de manière sécurisée une large gamme d'appareils tels que des capteurs, des caméras et des appareils électroménagers au cloud et effectuer des actions sur les données qu'ils génèrent. Vous pouvez utiliser AWS IoT Core pour envoyer des commandes aux appareils, recevoir des données de ces derniers, et analyser et traiter les données en temps réel.

AWS IoT Core prend en charge différents protocoles IoT, notamment MQTT, HTTP et WebSockets, et fournit un ensemble de SDK et d'API pour faciliter l'intégration de vos

appareils avec la plateforme. Elle propose également une intégration avec d'autres services AWS tels que AWS Lambda, Amazon Kinesis et Amazon S3 pour le traitement et le stockage de données.

Dans l'ensemble, AWS IoT Core fournit une plateforme évolutive et sécurisée pour la création d'applications IoT, vous permettant de créer, déployer et gérer rapidement et facilement des solutions IoT.

FIGURE 0.7: AWS IOT core plateforme

v) Conclusion :

L'ESP8266 est un module Wi-Fi très populaire qui peut être utilisé pour connecter des dispositifs électroniques à Internet. Il est équipé d'un microcontrôleur et peut être programmé à l'aide de diverses langues de programmation telles que Lua et Python. Il est également compatible avec l'IDE Arduino.

L'oxymètre de pouls MAX30102 est un capteur optique de fréquence cardiaque et d'oxygène dans le sang intégré dans un seul module. Il utilise la technologie de photopléthysmographie (PPG) pour mesurer la fréquence cardiaque et la saturation en oxygène du sang. Il dispose également d'un amplificateur de photodiode intégré pour améliorer la précision des mesures.

AWS IoT Core est une plateforme cloud fournie par Amazon Web Services pour la création d'applications IoT. Elle permet aux appareils de se connecter de manière sécurisée au cloud et d'interagir avec d'autres appareils et services. AWS IoT Core prend en charge différents protocoles IoT et fournit un ensemble de services gérés tels que l'enregistrement d'appareils, la communication sécurisée, le traitement de données, ainsi qu'une intégration avec d'autres services AWS pour le stockage et le traitement de données. En résumé, AWS IoT Core fournit une plateforme évolutive et sécurisée pour la création, le déploiement et la gestion de solutions IoT.

V- Circuits électriques : Schéma, câblage et branchement :

i) Interfaçage écran OLED SSD1306 avec Esp8266

A Câblage de l'écran OLED SSD1306 sur l'Esp8266 :

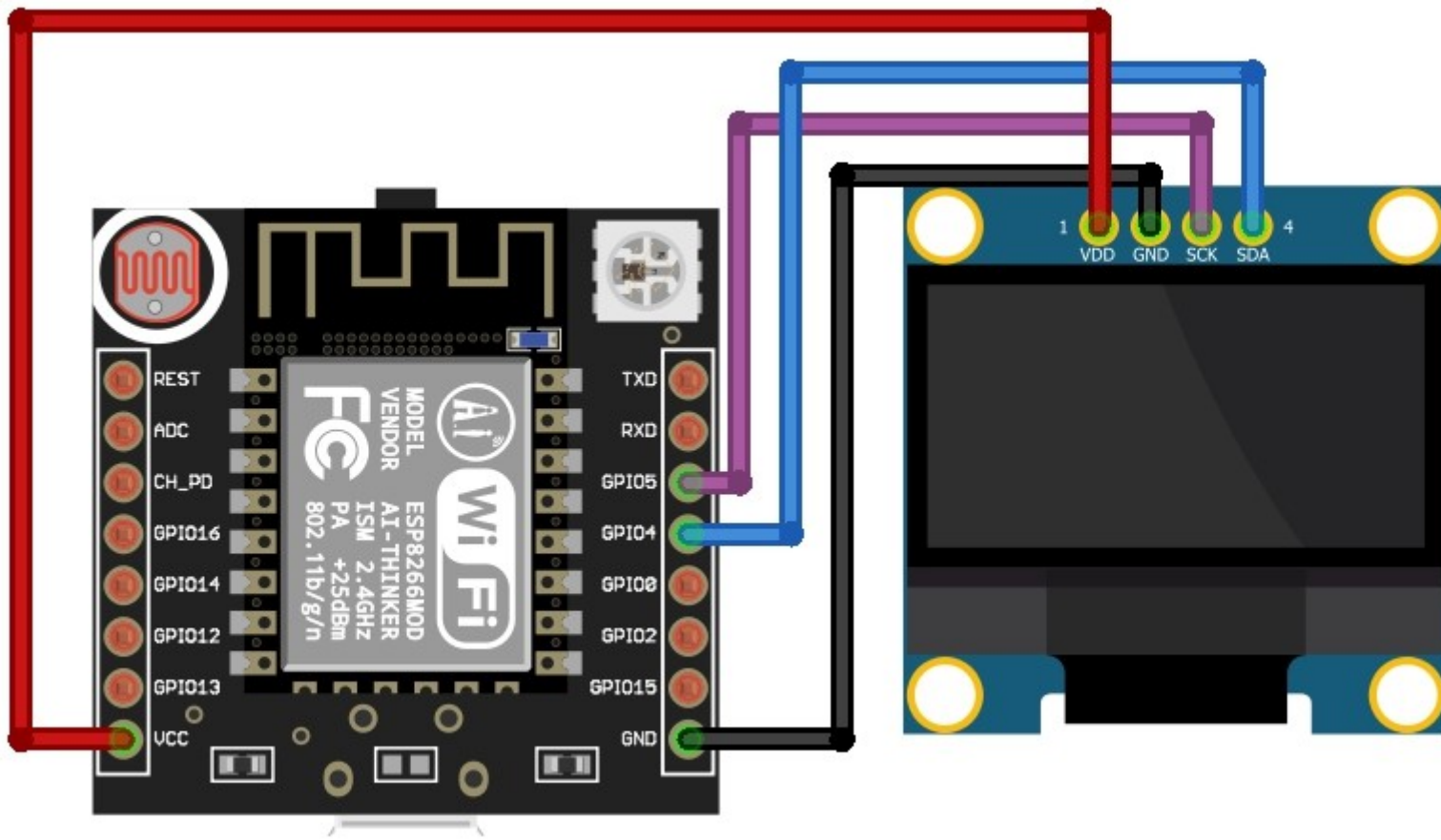


FIGURE 0.8: Câblage de l'écran OLED SSD1306 sur l'Esp8266

```
#include <Wire.h> // Include the Wire.h library to communicate in I2C
#include <Adafruit_GFX.h> // Include the Adafruit_GFX.h library for graphics support
#include <Adafruit_SSD1306.h> // Include Adafruit_SSD1306.h library for OLED display

#define SCREEN_WIDTH 128 // OLED screen width in pixels
#define SCREEN_HEIGHT 64 // Height of the OLED screen in pixels

// OLED screen initialization
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
    // Initializing serial communication for debugging
    Serial.begin(9600);

    // OLED display initialization with I2C address 0x3C (default for SSD1306 display)
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("Error: OLED screen not detected"));
        while (true);
    }

    // Clear OLED screen
```

```
22              display.clearDisplay();
23              display.display();
24          }
25
26          void loop() {
27              // Display the message "Hello!" on the OLED display
28              display.setTextSize(2);
29              display.setTextColor(WHITE);
30              display.setCursor(0, 0);
31              display.println("Bonjour !");
32              display.display();
33
34              // Wait 2 seconds before trying again
35              delay(2000);
36          }
37
38
```

ii) Interfaçage de l'oxymètre de pouls MAX30102 et du capteur de fréquence cardiaque avec Esp8266

A Câblage d'un module MAX30102 sur l'Esp8266 :

NIV est la broche d'alimentation. Vous pouvez le connecter à une sortie 3.3V ou 5V de votre Arduino.
SCL est la broche de l'horloge I2C, connectez-vous à la ligne d'horloge I2C de votre Arduino.
SDA est la broche de données I2C, connectez-vous à la ligne de données I2C de votre Arduino.
INT Le MAX30102 peut être programmé pour générer une interruption pour chaque impulsion. Cette ligne est à vidange ouverte, elle est donc tirée HAUT par la résistance embarquée. Lorsqu'une interruption se produit, la broche INT devient LOW et reste LOW jusqu'à ce que l'interruption soit effacée.
IRD Le MAX30102 intègre un pilote LED pour piloter les impulsions LED pour les mesures SpO2 et HR. Utilisez-le si vous souhaitez piloter vous-même la LED IR, sinon laissez-la déconnectée.
RD La broche est similaire à la broche IRD, mais est utilisée pour piloter la LED rouge. Si vous ne voulez pas piloter la LED rouge vous-même, laissez-la déconnectée.
GND est le sol. image.

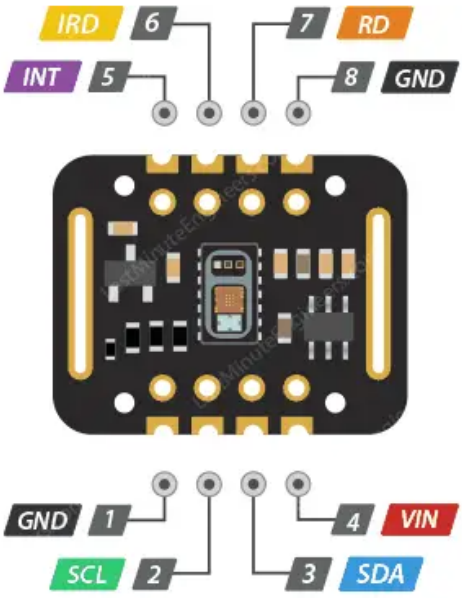


FIGURE 0.9: Brochage du module MAX30102

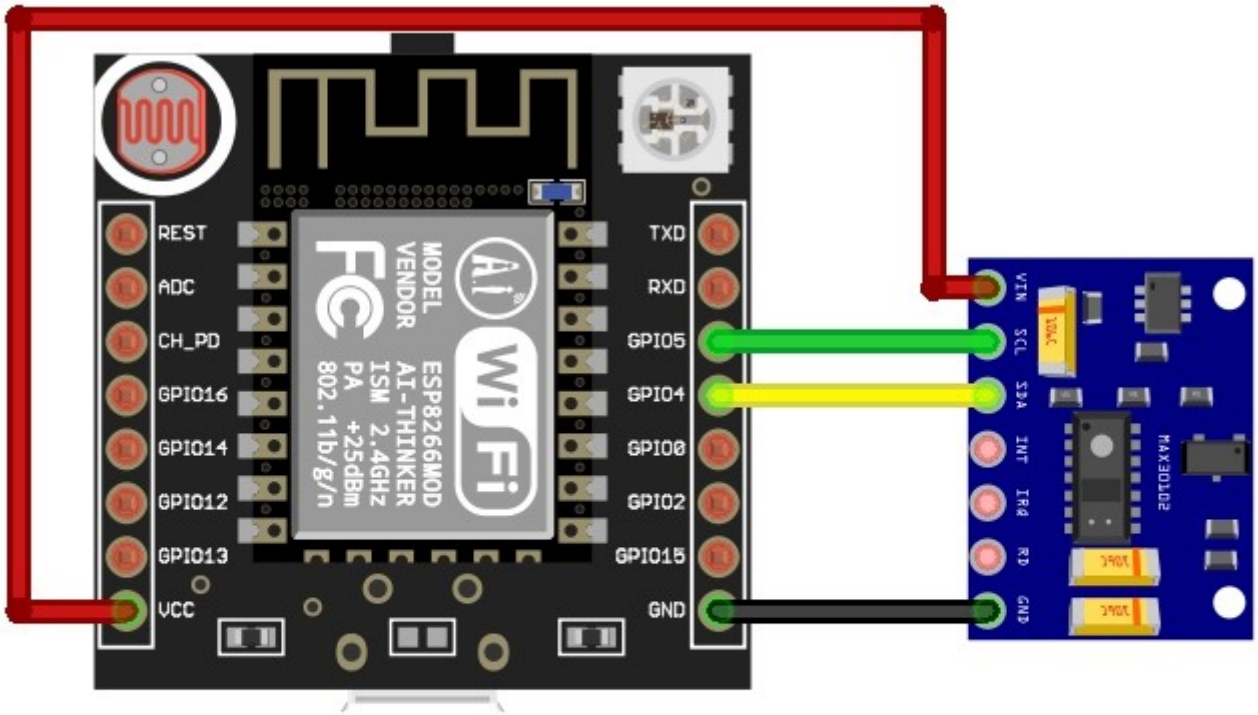


FIGURE 0.10: Câblage d'un module MAX30102 sur l'Esp8266

B Lecture du température

```
#include <Wire.h>

#include "MAX30105.h"
MAX30105 particleSensor;

void setup() {
    Serial.begin(9600);
    Serial.println("Initializing...");

    // Initialize sensor
    if (particleSensor.begin(Wire, I2C_SPEED_FAST) == false) { //Use default I2C port, 400kHz speed
        Serial.println("MAX30102 was not found. Please check wiring/power. ");
        while (1);
    }

    //The LEDs are very low power and won't affect the temp reading much but
    //you may want to turn off the LEDs to avoid any local heating
    particleSensor.setup(0); //Configure sensor. Turn off LEDs

    particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is required.
}

void loop() {
    float temperature = particleSensor.readTemperature();

```

```
            Serial.print("temperatureC=");
            Serial.print(temperature, 4);

            float temperatureF = particleSensor.readTemperatureF();

            Serial.print(" temperatureF=");
            Serial.print(temperatureF, 4);

            Serial.println();
        }

```

C Mesure de la fréquence cardiaque (BPM)

```
        #include <Wire.h>
        #include "MAX30105.h"
        #include "heartRate.h"

        MAX30105 particleSensor;

        const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is good.
        byte rates[RATE_SIZE]; //Array of heart rates
        byte rateSpot = 0;
        long lastBeat = 0; //Time at which the last beat occurred

        float beatsPerMinute;
        int beatAvg;

        void setup() {
            Serial.begin(115200);
            Serial.println("Initializing...");

            // Initialize sensor
            if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
                Serial.println("MAX30102 was not found. Please check wiring/power. ")
    ;
                while (1);
            }
            Serial.println("Place your index finger on the sensor with steady
    pressure.");

            particleSensor.setup(); //Configure sensor with default settings
            particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to
    indicate sensor is running
            particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
        }

        void loop() {
            long irValue = particleSensor.getIR();

            if (checkForBeat(irValue) == true) {
                //We sensed a beat!
                long delta = millis() - lastBeat;
                lastBeat = millis();

                beatsPerMinute = 60 / (delta / 1000.0);

                if (beatsPerMinute < 255 && beatsPerMinute > 20) {
                    rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in
     the array
                    rateSpot %= RATE_SIZE; //Wrap variable

```

```
                    //Take average of readings
                    beatAvg = 0;
                    for (byte x = 0 ; x < RATE_SIZE ; x++)
                    beatAvg += rates[x];
                    beatAvg /= RATE_SIZE;
                }
            }

            Serial.print("IR=");
            Serial.print(irValue);
            Serial.print(", BPM=");
            Serial.print(beatsPerMinute);
            Serial.print(", Avg BPM=");
            Serial.print(beatAvg);

            if (irValue < 50000)
            Serial.print(" No finger?");

            Serial.println();
        }

```

D Mesure de la saturation en oxygène (SpO2)

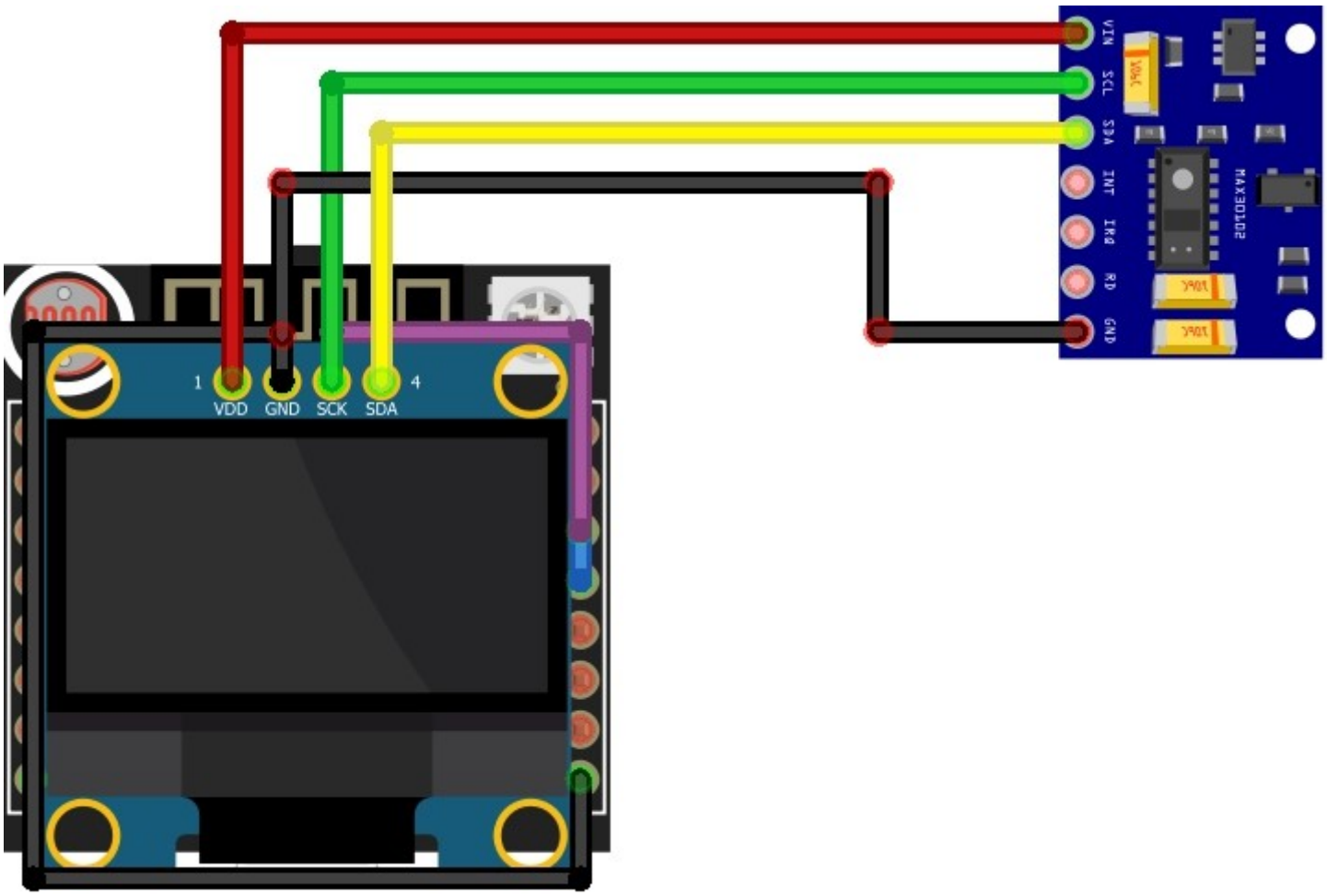
```
        #include <Wire.h>
        #include "MAX30105.h"
        #include "spo2_algorithm.h"

        MAX30105 particleSensor;

        #define MAX_BRIGHTNESS 255

        #if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
        //Arduino Uno doesn't have enough SRAM to store 100 samples of IR led data and red led data in 32-bit format
        //To solve this problem, 16-bit MSB of the sampled data will be truncated. Samples become 16-bit data.
        uint16_t irBuffer[100]; //infrared LED sensor data
        uint16_t redBuffer[100];  //red LED sensor data
        #else
        uint32_t irBuffer[100]; //infrared LED sensor data
        uint32_t redBuffer[100];  //red LED sensor data
        #endif

        int32_t bufferLength; //data length
        int32_t spo2; //SPO2 value
        int8_t validSPO2; //indicator to show if the SPO2 calculation is valid
        int32_t heartRate; //heart rate value
        int8_t validHeartRate; //indicator to show if the heart rate calculation is valid

        byte pulseLED = 11; //Must be on PWM pin
        byte readLED = 13; //Blinks with each data read

        void setup()
        {
            Serial.begin(115200); // initialize serial communication at 115200 bits per second:

            pinMode(pulseLED, OUTPUT);
            pinMode(readLED, OUTPUT);

```

```
35          // Initialize sensor
36          if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port, 400kHz speed
37          {
38              Serial.println(F("MAX30105 was not found. Please check wiring/power."));
39              while (1);
40          }
41
42          Serial.println(F("Attach sensor to finger with rubber band. Press any key to start conversion"));
43          while (Serial.available() == 0) ; //wait until user presses a key
44          Serial.read();
45
46          byte ledBrightness = 60; //Options: 0=Off to 255=50mA
47          byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
48          byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
49          byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
50          int pulseWidth = 411; //Options: 69, 118, 215, 411
51          int adcRange = 4096; //Options: 2048, 4096, 8192, 16384
52
53          particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange); //Configure sensor with these settings
54      }
55
56      void loop()
57      {
58          bufferLength = 100; //buffer length of 100 stores 4 seconds of samples running at 25sps
59
60          //read the first 100 samples, and determine the signal range
61          for (byte i = 0 ; i < bufferLength ; i++)
62          {
63              while (particleSensor.available() == false) //do we have new data?
64              particleSensor.check(); //Check the sensor for new data
65
66              redBuffer[i] = particleSensor.getRed();
67              irBuffer[i] = particleSensor.getIR();
68              particleSensor.nextSample(); //We're finished with this sample so move to next sample
69
70              Serial.print(F("red="));
71              Serial.print(redBuffer[i], DEC);
72              Serial.print(F(", ir="));
73              Serial.println(irBuffer[i], DEC);
74          }
75
76          //calculate heart rate and SpO2 after first 100 samples (first 4 seconds of samples)
77          maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);
78
79          //Continuously taking samples from MAX30102.  Heart rate and SpO2 are calculated every 1 second
80          while (1)
81          {
82              //dumping the first 25 sets of samples in the memory and shift the last 75 sets of samples to the top
83              for (byte i = 25; i < 100; i++)
84              {
85                  redBuffer[i - 25] = redBuffer[i];
86                  irBuffer[i - 25] = irBuffer[i];
```

```
                }

                //take 25 sets of samples before calculating the heart rate.
                for (byte i = 75; i < 100; i++)
                {
                    while (particleSensor.available() == false) //do we have new data ?
                    particleSensor.check(); //Check the sensor for new data

                    digitalWrite(readLED, !digitalRead(readLED)); //Blink onboard LED with every data read

                    redBuffer[i] = particleSensor.getRed();
                    irBuffer[i] = particleSensor.getIR();
                    particleSensor.nextSample(); //We're finished with this sample so move to next sample

                    //send samples and calculation result to terminal program through UART
                    Serial.print(F("red="));
                    Serial.print(redBuffer[i], DEC);
                    Serial.print(F(", ir="));
                    Serial.print(irBuffer[i], DEC);

                    Serial.print(F(", HR="));
                    Serial.print(heartRate, DEC);

                    Serial.print(F(", HRvalid="));
                    Serial.print(validHeartRate, DEC);

                    Serial.print(F(", SPO2="));
                    Serial.print(spo2, DEC);

                    Serial.print(F(", SPO2Valid="));
                    Serial.println(validSPO2, DEC);
                }

                //After gathering 25 new samples recalculate HR and SPO2
                maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);
            }
        }

```

VI- Réalisation du système de télésurveillance :

i) Circuits électriques et prototype :



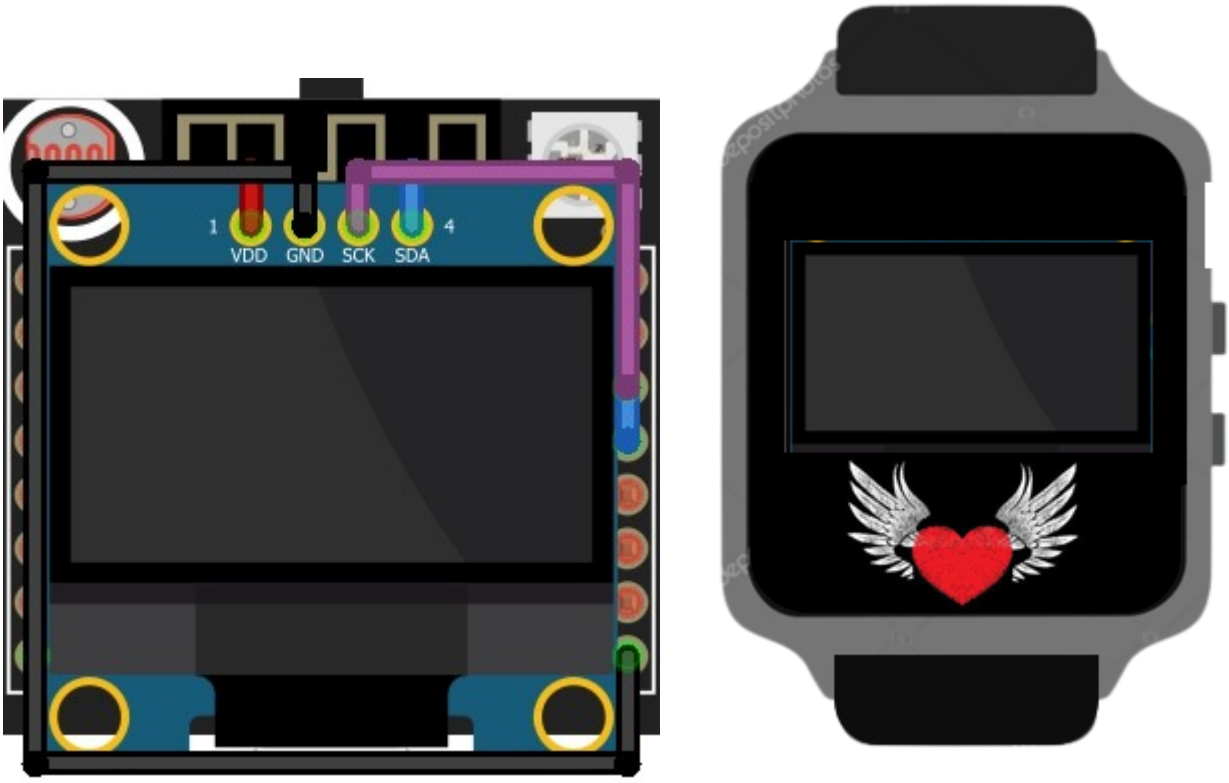


FIGURE 0.11: Circuits électriques et prototype

ii) Connection de l'ESP8266 à AWS IoT Core :

La connexion de l'ESP8266 à AWS IoT Core implique plusieurs étapes. Voici un aperçu des étapes à suivre :
1. Créer un compte AWS : Si vous n'avez pas de compte AWS, vous devez d'abord en créer un. Vous pouvez vous inscrire à l'aide de ce lien : https ://aws.amazon.com/fr/free/.
2. Créer un groupe de sécurité : Un groupe de sécurité permet de spécifier les règles de trafic entrant et sortant pour les instances EC2. Pour créer un groupe de sécurité, vous devez vous rendre dans le tableau de bord AWS et sélectionner l'option "Groupe de sécurité" dans la section "Sécurité, Identité et Conformité".
3. Créer un certificat : Pour communiquer avec AWS IoT Core, vous devez créer un certificat et une clé privée. Vous pouvez générer ces éléments à l'aide du kit de développement logiciel AWS IoT (SDK). Vous pouvez télécharger le SDK ici : https ://aws.amazon.com/fr/iot-sdk/.
4 Configurer l'ESP8266 : Vous devez configurer l'ESP8266 pour qu'il puisse communiquer avec AWS IoT Core. Pour ce faire, vous pouvez utiliser l'IDE Arduino et la bibliothèque AWS IoT. Vous pouvez télécharger la bibliothèque AWS IoT ici : https ://github.com/aws/aws-iot-device-sdk-arduino-yt.
5. Publier des messages : Une fois que votre ESP8266 est configuré, vous pouvez publier des messages sur AWS IoT Core. Vous pouvez le faire en utilisant la méthode "publish" de la bibliothèque AWS IoT.

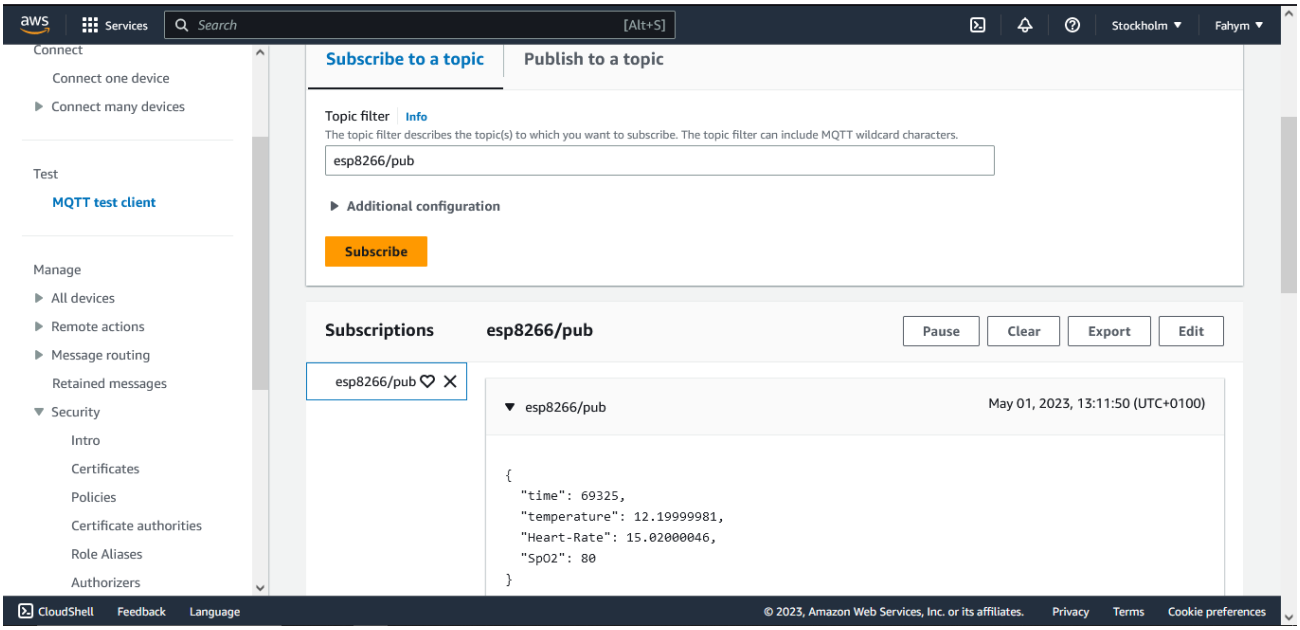


FIGURE 0.12: Connection de l'ESP8266 à AWS IoT Core

iii) Réalisation finale du prototype :

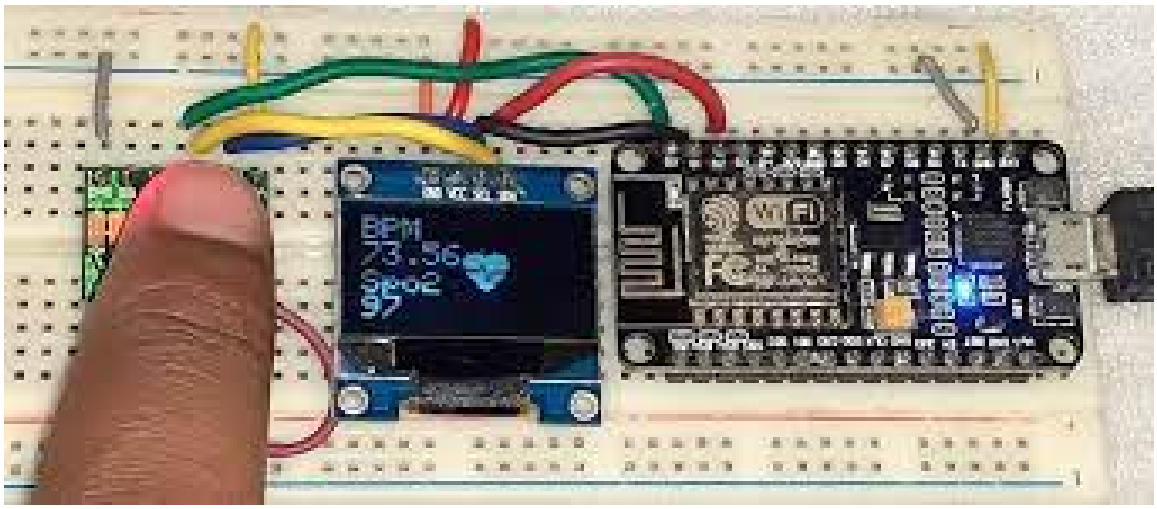


FIGURE 0.13: Prototype finale

Conclusion générale

En conclusion de ce rapport de stage chez Molbiol Expert, nous pouvons affirmer que cette expérience a été très enrichissante pour moi. J'ai pu acquérir des connaissances théoriques et pratiques dans le domaine de l'IoT, de la télémédecine et de la surveillance à distance. J'ai également pu découvrir les différentes activités de Molbiol Expert et comprendre son fonctionnement. J'ai été impressionné par la qualité des prestations fournies par cette entreprise et la rigueur de ses équipes.

Au cours de ce stage, j'ai travaillé sur la réalisation d'un prototype de système de télésurveillance pour la mesure de la fréquence cardiaque et de la saturation en oxygène. J'ai ainsi pu mettre en pratique mes connaissances théoriques et développer mes compétences en matière de circuits électroniques, de programmation et de cloud computing.

Enfin, je tiens à remercier l'ensemble de l'équipe de Molbiol Expert pour son accueil chaleureux, sa disponibilité et son encadrement durant ce stage. Cette expérience a été très formatrice pour moi et m'a permis de développer des compétences clés pour mon avenir professionnel.

ANNEXES

Programme en Arduino qui utilise l'oxymètre de pouls MAX30102 pour mesurer la température corporelle, la fréquence cardiaque et la saturation en oxygène, puis affiche ces mesures sur un écran OLED SSD1306

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include "MAX30105.h"
#include <MAX30105.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

MAX30105 particleSensor;

void setup() {
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
    display.display();
    delay(2000);
    display.clearDisplay();

    Wire.begin(4, 5);
    particleSensor.begin(Wire, I2C_SPEED_FAST);
    particleSensor.setup();
    particleSensor.setPulseAmplitudeRed(0x0A);
    particleSensor.setPulseAmplitudeGreen(0);
}

void loop() {
    int32_t ir, red;
    float temperature, bpm, spo2;

    temperature = particleSensor.readTemperature();
    ir = particleSensor.getIR();
    red = particleSensor.getRed();
    bpm = particleSensor.getHeartRate();
    spo2 = particleSensor.getSpO2();

    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 0);
    display.println("Temperature: " + String(temperature) + " C");
    display.setCursor(0, 10);
    display.println("Heart Rate: " + String(bpm) + " bpm");
    display.setCursor(0, 20);
    display.println("SpO2: " + String(spo2) + " \%");
    display.display();

    delay(1000);
}
```

Programme Arduino qui permet de connecter un module ESP8266 à AWS IoT Core

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <time.h>
#include "secrets.h"



float temperature, bpm, spo2;

unsigned long lastMillis = 0;
unsigned long previousMillis = 0;
const long interval = 5000;

#define AWS_IOT_PUBLISH_TOPIC   "esp8266/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp8266/sub"

WiFiClientSecure net;

BearSSL::X509List cert(cacert);
BearSSL::X509List client_crt(client_cert);
BearSSL::PrivateKey key(privkey);

PubSubClient client(net);

time_t now;
time_t nowish = 1510592825;


void NTPConnect(void){
    Serial.print("Setting time using SNTP");
    configTime(TIME_ZONE * 3600, 0 * 3600, "pool.ntp.org", "time.nist.gov");
    now = time(nullptr);
    while (now < nowish)
    {
        delay(500);
        Serial.print(".");
        now = time(nullptr);
    }
    Serial.println("done!");
    struct tm timeinfo;
    gmtime_r(&now, &timeinfo);
    Serial.print("Current time: ");
    Serial.print(asctime(&timeinfo));
}


void messageReceived(char *topic, byte *payload, unsigned int length){
    Serial.print("Received [");
    Serial.print(topic);
    Serial.print("]: ");
    for (int i = 0; i < length; i++)
    {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

```

```
60
61          void connectAWS(){
62              delay(3000);
63              WiFi.mode(WIFI_STA);
64              WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
65
66              Serial.println(String("Attempting to connect to SSID: ") + String(
    WIFI_SSID));
67
68              while (WiFi.status() != WL_CONNECTED)
69              {
70                  Serial.print(".");
71                  delay(1000);
72              }
73
74              NTPConnect();
75
76              net.setTrustAnchors(&cert);
77              net.setClientRSACert(&client_crt, &key);
78
79              client.setServer(MQTT_HOST, 8883);
80              client.setCallback(messageReceived);
81
82
83              Serial.println("Connecting to AWS IOT");
84
85              while (!client.connect(THINGNAME)){
86                  Serial.print(".");
87                  delay(1000);
88              }
89
90              if (!client.connected()) {
91                  Serial.println("AWS IoT Timeout!");
92                  return;
93              }
94              // Subscribe to a topic
95              client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);
96
97              Serial.println("AWS IoT Connected!");
98          }
99
100
101         void publishMessage(){
102             StaticJsonDocument<200> doc;
103             doc["time"] = millis();
104             doc["temperature"] = temperature;
105             doc["Heart-Rate"] = bpm;
106             doc["SpO2"] = spo2;
107             char jsonBuffer[512];
108             serializeJson(doc, jsonBuffer); // print to client
109
110             client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
111         }
112
113
114         void setup(){
115             Serial.begin(115200);
116             connectAWS();
117         }
118
119
120         void loop(){
121
122             temperature = 12.2;
```

```
            bpm = 15.02;
            spo2 = 80 ;

            if (isnan(temperature) || isnan(bpm) ||isnan(bpm) )  // Check if any reads failed and exit early (to try again).
            {
                Serial.println(F("Failed to read from DHT sensor!"));
                return;
            }

            Serial.println("Temperature: " + String(temperature) + " C");
            Serial.println("Heart Rate: " + String(bpm) + " bpm");
            Serial.println("SpO2: " + String(spo2) + " %");

            delay(2000);

            now = time(nullptr);

            if (!client.connected()){
                connectAWS();
            }
            else{
                client.loop();
                if (millis() - lastMillis > 5000){
                    lastMillis = millis();
                    publishMessage();
                }
            }
        }


```

Programme en Arduino qui utilise l'oxymètre de pouls MAX30102 pour mesurer la température de l'homme, la fréquence cardiaque et la saturation en oxygène. Les mesures sont affichées sur un écran OLED SSD1306 connecté à une ESP8266 avec la communication I2C

```
        #include <ESP8266WiFi.h>
        #include <WiFiClientSecure.h>
        #include <PubSubClient.h>
        #include <ArduinoJson.h>

        #include <Wire.h>
        #include <Adafruit_SSD1306.h>
        #include "MAX30105.h"
        #include <MAX30105.h>

        #include <time.h>
        #include "secrets.h"

        #define SCREEN_WIDTH 128
        #define SCREEN_HEIGHT 32
        Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

        MAX30105 particleSensor;
        int32_t ir, red;
        float temperature, bpm, spo2;
```

```
21
22        unsigned long lastMillis = 0;
23        unsigned long previousMillis = 0;
24        const long interval = 5000;
25
26        #define AWS_IOT_PUBLISH_TOPIC   "esp8266/pub"
27        #define AWS_IOT_SUBSCRIBE_TOPIC "esp8266/sub"
28
29        WiFiClientSecure net;
30
31        BearSSL::X509List cert(cacert);
32        BearSSL::X509List client_crt(client_cert);
33        BearSSL::PrivateKey key(privkey);
34
35        PubSubClient client(net);
36
37        time_t now;
38        time_t nowish = 1510592825;
39
40
41        void NTPConnect(void){
42            Serial.print("Setting time using SNTP");
43            configTime(TIME_ZONE * 3600, 0 * 3600, "pool.ntp.org", "time.nist.gov");
44            now = time(nullptr);
45            while (now < nowish){
46                delay(500);
47                Serial.print(".");
48                now = time(nullptr);
49            }
50            Serial.println("done!");
51            struct tm timeinfo;
52            gmtime_r(&now, &timeinfo);
53            Serial.print("Current time: ");
54            Serial.print(asctime(&timeinfo));
55        }
56
57
58        void messageReceived(char *topic, byte *payload, unsigned int length){
59            Serial.print("Received [");
60            Serial.print(topic);
61            Serial.print("]: ");
62            for (int i = 0; i < length; i++)
63            {
64                Serial.print((char)payload[i]);
65            }
66            Serial.println();
67        }
68
69
70        void connectAWS(){
71            delay(3000);
72            WiFi.mode(WIFI_STA);
73            WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
74
75            Serial.println(String("Attempting to connect to SSID: ") + String(
    WIFI_SSID));
76
77            while (WiFi.status() != WL_CONNECTED)
78            {
79                Serial.print(".");
80                delay(1000);
81            }
82
83            NTPConnect();
```

```
84
85          net.setTrustAnchors(&cert);
86          net.setClientRSACert(&client_crt, &key);
87
88          client.setServer(MQTT_HOST, 8883);
89          client.setCallback(messageReceived);
90
91
92          Serial.println("Connecting to AWS IOT");
93
94          while (!client.connect(THINGNAME)){
95              Serial.print(".");
96              delay(1000);
97          }
98
99          if (!client.connected()) {
100             Serial.println("AWS IoT Timeout!");
101             return;
102         }
103         // Subscribe to a topic
104         client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);
105
106         Serial.println("AWS IoT Connected!");
107     }
108
109
110     void publishMessage(){
111         StaticJsonDocument<200> doc;
112         doc["time"] = millis();
113         doc["temperature"] = temperature;
114         doc["Heart-Rate"] = bpm;
115         doc["SpO2"] = spo2;
116         char jsonBuffer[512];
117         serializeJson(doc, jsonBuffer); // print to client
118
119         client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
120     }
121
122
123     void setup(){
124         display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
125         display.display();
126         delay(2000);
127         display.clearDisplay();
128
129         Wire.begin(4, 5);
130         particleSensor.begin(Wire, I2C_SPEED_FAST);
131         particleSensor.setup();
132         particleSensor.setPulseAmplitudeRed(0x0A);
133         particleSensor.setPulseAmplitudeGreen(0);
134
135         Serial.begin(115200);
136         connectAWS();
137
138
139     }
140
141
142     void loop(){
143
144         readSensor();
145         displaySSD1306();
146         if (isnan(temperature) || isnan(bpm) ||isnan(bpm) ) {
147             Serial.println(F("Failed to read from DHT sensor!"));
```

```
            return;
        }

        Serial.println("Temperature: " + String(temperature) + " C");
        Serial.println("Heart Rate: " + String(bpm) + " bpm");
        Serial.println("SpO2: " + String(spo2) + " %");

        delay(2000);

        now = time(nullptr);

        if (!client.connected()){
            connectAWS();
        }
        else{
            client.loop();
            if (millis() - lastMillis > 5000){
                lastMillis = millis();
                publishMessage();
            }
        }
    }

    void readSensor(){
        temperature = particleSensor.readTemperature();
        ir = particleSensor.getIR();
        red = particleSensor.getRed();
        bpm = particleSensor.getHeartRate();
        spo2 = particleSensor.getSpO2();
    }

    void displaySSD1306(){
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
        display.setCursor(0, 0);
        display.println("Temperature: " + String(temperature) + " C");
        display.setCursor(0, 10);
        display.println("Heart Rate: " + String(bpm) + " bpm");
        display.setCursor(0, 20);
        display.println("SpO2: " + String(spo2) + " \%");
        display.display();

        delay(1000);
    }
```

Le fichier "secrets.h"

```
    #include <pgmspace.h>
    #define SECRET

    const char WIFI_SSID[] = "HTL-WS";
    const char WIFI_PASSWORD[] = "1234567890";

    #define THINGNAME "ESP8266"

    int8_t TIME_ZONE = -5; //NYC(USA): -5 UTC

    const char MQTT_HOST[] = "aab79spckx1od-ats.iot.eu-north-1.amazonaws.com";
```

```
static const char cacert[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKW1JbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
)EOF";



static const char client_cert[] PROGMEM = R"KEY(
-----BEGIN CERTIFICATE-----
MIIDWjCCAkKgAwIBAgIVAL+FaPfimhdF7/UEOZqo+2gAZh71MA0GCSqGSIb3DQEB
CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTz1BbWF6b24uY29t
IEluYy4gTD1TZWF0dGxlIFNUPVdhc2hpbmd0b24gQz1VUzAeFw0yMzA0MjEwNTA4
NDRaFw0OOTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJb1QgQ2VydGlmaWNh
dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCTuWRvavBqV2+AFcay
Hl1TxDLXb2H1LLdlsPAiwwxHkQ/4OgWpDaH0vqluirniDZ0/LjxEjDky+CN/yYC7
UZzehv7+Sio9SriobfFQ5pZj0X1ru9+UNzoD38CvwTvBC1X2+4ZvTz+I+EtxKJUa
47z0gsNEE9nt0dBAnizSmg8433QwR9VDItKBiexL4MrLqN5+7EeZDz4XZbMW1yfr
PKKzalvn5cZaog47Saevj0sDd21Z1kMAJ/PIq144LHg0bW+yNd7QWgsYkHdDZ+um
QGUzWr/sCyDf2OSkajV6MB9BkgCjD5mF6XEjcdO0zFA7jT7eQzx1SbnikXn8cJlK
m0EBAgMBAAGjYDBeMB8GA1UdIwQYMBaAFCn/5UdHpGh09jQ4z8RLAGTrAPTmMB0G
A1UdDgQWBBR7jYLN75XEOmkUnVFQ3X0kdEAd1zAMBgNVHRMBAf8EAjAAMA4GA1Ud
DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEAoiZp+iIJejxvlDHrrFoffsfO
/xjdcR1tAFPUkD/hXycXFtv3xa0aApnBEYZPfymAISdlcS+D30evqZWAyZclBL1r
0q10eSK4NA6TKrGNZ0EluwQspbJgpwT7wCubDV6sMgrGp/nikoVWHK1iIPEvWvvG
SZM0KD5NnUOYIdmf9LkhPs3goBKYAp4UG/YWu4jeLY56cNI1Jrl23kKNKIjWl85K
spJXlkjblYNKixZAYgRxDwG91c+WEAoeXCPe/Ek1m7JUlzChilZ0qHPiMCyCamf0
QtZh5Rk5uLUspya0eJNWAqvWJrR3imVfyyW7tkM/N/Cm4Em/4XFcN0z17+ygGw==
-----END CERTIFICATE-----


)KEY";



static const char privkey[] PROGMEM = R"KEY(
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAk7lkb2rwaldvgBXGsh5dU8Qy129h9Sy3ZbDwIsMMR5EP+DoF
qQ2h9L6pboq54g2dPy48RIw5Mvgjf8mAu1Gc3ob+/koqPUq4qG3xU0aWY9F9a7vf
lDc6A9/Ar8E7wQtV9vuGb08/iPhLcSiVGuO8zoLDRBPZ7dHQQJ4s0poPON90MEfV
QyLSgYnsS+DKy6jefuxHmQ8+F2WzFtcn6zyis2pb5+XGWqIOO0mnr49LA3dtWdZD
ACfzyKteOCx4NG1vsjXe0FoLGJB3Q2frpkBlM1q/7Asg39jkpGo1ejAfQZIAow+Z
helxI3HTjsxQ040+3kM8dUm54pF5/HCZSptBAQIDAQABAoIBACmuBkt1dc3bS6ds
/XnmHVj0I5Arag0eB8TWbnqwY7eRk1rrLAutwANpBC8fTyEbzEE2T2Tb14us/X8h
oIO5nDQhDLM8w7w1Z/KFyq+Tz9zsD6IlCQktp93N9PTh5XUr0J8OCrk/N+Lyg36y
```

```
        /5btqKjvth5wJ231Y/fHaUgmlaLq44Pz4hucY1zRm6rvxJqapZK7lMpU1QJXEoCp
        2hJoECu85uTTwkULjYbyjl0D1tMjE15ogMGVR47/kRQl8b1NcGI7UjhVFgFwjnBg
        YVa9bEIacZb3canNDSphmM++3EnJUO2r4NgDxJ4jKBU4lvUFgnsxdL0hC5jCwual
        K2XCYUECgYEAxJQZjKEOHV2RmzwwZ73u5Nc5ci1a44ZcE5R4frhfUvAZRzVvskeV
        WP56QI18WDeTZx0qY77+Sd/ru526XsoV4e+KFeh76u6c4qjyED39cblmjcU6pr82
        ZCrt4d9hbQzSKuhyY0O1tFU2Ixd6HU/LFWWY2JEtmIHka/4/wBUoSOkCgYEAwGDI
        NPQBlq35HOa3hfNc6NeuhVE2Ep0qECcBKImZUBkVNUp+SXKtMz5xUx6qWtpDRJ0r
        ealwiDizAgKcqLeXne955YNvVjEkk/7AwpicLeu/HrxgdzsGx0+tpvG99SK3IW5n
        BGRVl7QVean/K6htAssxpxIXOCeXAUOe0LkZqFkCgYAv+iFeNDSYsCyIYaFCwJVc
        nraziiRDoZg5YyQhcWg6esUAnycasOPfa4R02tM9SJseMH0XCRdQ+mizSqSg29uJ
        YAgonqwXw5LgsvEZS7femKxR75AIAGUK/3s9hGJn1hg5RrAcT/s5/w4OdH1mSAI9
        v6UVPiGNti4EuXHftIo3eQKBgHHP0CIjyVzPh/VlOWXx2CGYmBMfPJI+kMgFi1Xk
        MeOFM1rXu2bQhU8vvfOizL37KpjJMxNC/uRYRu8FhJD8LKKvLrpty8x+Plmf+YIm
        rx8rvWH9qERhk10O8nk1w1/nLdiFbETY5xpc2+eifufhOLzSYYmp1jrOkTOaQbbS
        Cy/hAoGAetC4q7bmT5CLtEYkrgNcQ0FQaJ/10gpvmF0OCCnMlqnCHoNLLxG7DmjS
        vNV69nJboYJNnvm8Eb+bsdEZQEvKTO64JNdqw4dpUga00dnK0uu75lnrn2JYRUjo
        aAVhAgp2+TpI1s9H8FR4utEZZIVRl2Bwn8vab1nQguRhsJWETVo=
        -----END RSA PRIVATE KEY-----


        )KEY";


```

Bibliographie

https ://molbiolexpert.com/
https ://www.charika.ma/societe-molbiol-expert-499135
"Getting Started with ESP8266" de Daniyal Syed et Muhammad bin Mazhar
"IoT Projects with ESP32" de Agus Kurniawan
"AWS IoT Core : Getting Started with AWS IoT Core" de Ravi Kalakota
Site officiel d'Arduino : https ://www.arduino.cc/
Site officiel de NodeMCU : https ://nodemcu.readthedocs.io/en/release/
Site officiel de AWS IoT Core : https ://aws.amazon.com/fr/iot-core/
Tutoriel de connexion de NodeMCU à AWS IoT Core :
https ://medium.com/@esmayclin/aws-iot-esp8266-nodemcu-ec677fb18f96
Tutoriel pour l'interfaçage de l'oxymètre de pouls MAX30102 avec NodeMCU :
https ://how2electronics.com/pulse-oximeter-using-max30102-heart-rate-sensor-with-esp8266/
Tutoriel pour l'interfaçage de l'écran OLED SSD1306 avec NodeMCU :
https ://randomnerdtutorials.com/esp8266-0-96-inch-oled-display-with-arduino-ide/

https ://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/
https ://esp8266-shop.com/esp8266-guide/esp8266-information/
https ://microcontrollerslab.com/esp8266-heart-rate-pulse-oximeter-max30102/
https ://simple-circuit.com/arduino-ssd1306-oled-i2c-spi-example/