

Projet multidisciplinaire 2 et gestion de projet - TRAN-H201-Groupe 3

Suivi de santé à domicile Covid-19

El Mokhtari Younes

El Qaisy Driss

Mvukiyehe Melissa

Ouassari Bilal

Pauwels Lise

Van Leer Cyril

van Melsen Guillaume

Tuteur : Foucart Adrien

Lecteur : Ercek Rudy

Table des matières

1	Introduction	1
2	Méthodologie de groupe	3
2.1	Gestion de projet	3
2.2	Fonctionnement de groupe	4
2.2.1	Communication et organisation	4
2.2.2	SWOT	5
3	Symptômes et choix des capteurs	7
3.1	Symptômes de la Covid-19	7
3.2	Choix des capteurs	9
3.3	Capteur de température	11
3.4	Capteur de pouls	12
3.5	Capteur de courant	13
3.6	Oxymètre	14
3.7	Récapitulatif	14
4	Emplacement des capteurs	15
5	Microcontrôleur	17
5.1	l'ESP32	17
5.2	Choix du langage de programmation	18
6	Protection des données	20
6.1	La RGPD	20
6.2	Cryptage de données	21

6.3 SSL	23
7 Base de données	24
7.1 Concepts clefs d'une base de données temporelle	24
7.2 Queries	25
7.3 Interaction avec la base de données Biomed3	26
7.3.1 Choix du langage	27
7.3.2 Explication du code	27
8 Graphana	30
8.1 Paramétrage du panel	30
8.2 Paramétrage du graphique	31
8.2.1 Graphique	31
8.2.2 Valeurs pertinentes du graphique	32
9 Traitement de données	33
9.1 Récupération des données	33
9.2 Explications du code	33
10 Interprétation des données	36
10.1 Interprétation des données	36
10.2 Explication du code	37
11 Simulateur	40
11.1 Création d'un objet patient	40
11.1.1 Explication du code	40
12 Site Web	44
12.1 Langages	44
12.2 XAMPP	45
12.3 Page d'accueil	45
12.4 Questionnaire	46
12.5 Page de relevés	48
12.6 Notifications	52
12.7 Amélioration	53

13 Prototype périodique	54
13.1 Batterie	54
13.1.1 Découpage du code	54
13.1.2 Boucle	55
13.1.3 Calculs	55
13.1.4 Choix	57
13.2 Code	58
13.2.1 Capteurs	58
13.2.2 WiFi	60
13.2.3 InfluxDB	61
13.2.4 Deep-Sleep	62
13.2.5 Setup	63
13.3 Boîtier	64
13.4 Construction	64
14 Prototype continu	66
14.1 Batterie	66
14.1.1 Découpage du code	66
14.1.2 Calculs et boucle	67
14.2 Code	69
14.2.1 Capteurs	69
14.2.2 WiFi	70
14.2.3 InfluxDB	70
14.2.4 Setup	70
14.2.5 Loop	71
14.3 Boîtier	72
14.4 Construction	72
15 Tests	74
15.1 Calibrage	74
15.2 Expériences	74
15.2.1 Expérience n°1	74
15.2.2 Expérience n°2	76
15.2.3 Tests complémentaires	79
15.2.4 Expérience n°3	85

15.2.5 Expérience n°4	87
15.2.6 Confort et ergonomie des prototypes	91
16 Mesures d'hygiène	92
17 Conclusion	93
A Gestion de projet	95
B Site Web	99
C Prototype périodique	107
D Manuel d'utilisation	108

Résumé

Nombre de mots dans le rapport : 19063

Dans le cadre du projet de deuxième année d'études d'ingénieur civil à l'ULB option biomédicale, il a été demandé aux étudiants de réaliser un dispositif de suivi de santé à domicile Covid-19. Pour cela, un ou plusieurs symptômes de la maladie doivent être décelés chez le patient portant l'appareil. Ce dernier doit également être capable d'analyser l'évolution de l'état de la personne. En cas d'inclinaison de la maladie, un dispositif continu sera plutôt utilisé, avec la possibilité de connexion d'un oxymètre pour une analyse plus précise de la situation.

Le dispositif est constitué d'un microcontrôleur, qui recueille les informations des capteurs, directement en contact avec la peau du patient.

Cependant, certains symptômes ne sont pas mesurables comme la perte d'odorat ou les douleurs thoraciques. Pour résoudre ce problème, un questionnaire sera rempli par le patient régulièrement.

Dans les deux cas, les informations sont stockées dans une base de données pour ensuite être analysées afin de déterminer l'état de santé du patient mais aussi certains cas particuliers tels qu'un mauvais placement de l'appareil ou un retrait complet.

Une page web est mise à disposition du patient afin de lui communiquer les informations lier à son état de santé. Une augmentation de température ou de pouls, ou même des capteurs mal placés sur la peau, toutes les informations nécessaires arriveront également sur sa boîte mail.

1

Introduction

Fin 2019, les premiers malades de la Covid-19 apparaissent en Chine. La maladie gagna ensuite d'autres continents, dont l'Europe, et dès mars 2020, la Belgique instaura un premier confinement. Cette pandémie a donc eu (et a toujours) de grandes répercussions, y compris dans la vie des étudiants.

Par conséquent, le projet biomédical des BA2 de cette année s'inscrit parfaitement dans l'actualité. Dans le cadre de la lutte contre la propagation de la Covid-19, il s'agira d'élaborer un suivi de santé à domicile destiné aux personnes ayant eu un contact rapproché avec une personne contaminée. L'objectif est de suivre l'apparition et l'aggravation d'un des symptômes du virus afin d'alerter les équipes médicales si la situation s'avère être alarmante. Le dispositif devra également permettre de détecter un non-respect des règles de quarantaine.

Ce projet demande une élaboration matérielle et une élaboration logistique. L'élaboration matérielle fait référence aux deux prototypes qui devront être livrés à la fin du projet : l'un fonctionne de façon périodique, l'autre en continu. Tous deux devront servir de capteurs pour relever des données liées aux symptômes ainsi que d'un ESP32 pour envoyer ces relevés à une base de données. L'élaboration logistique est centrée autour de ces relevés : elle fait notamment référence à l'organisation de la base de données, au traitement et à l'analyse des relevés, à la constitution d'une interface de visualisation de ces données ainsi qu'à la simulation de celles-ci.

Le présent document aborde les différentes parties du projet qu'il a fallu développer dans le but de livrer le suivi de santé tel que décrit dans le cahier des charges.

Le rapport s'intéressera d'abord à la méthodologie de groupe. Il traitera ensuite des symptômes de la Covid-19 et le choix des capteurs qui s'en est suivi, puis expliquera le fonctionnement général du microcontrôleur. Il s'intéresse également à la protection des données récoltées par l'ESP32,

la conservation de ces données dans une base de données temporelle (InfluxDB), et à l'interface graphique Grafana, qui permet de les visualiser après leur stockage. Le rapport abordera par après le traitement et l'interprétation de ces données, ainsi qu'au simulateur qui permet de tester ces codes. De plus, ce document expliquera l'élaboration du site web, qui constitue une interface complémentaire pour le patient et les équipes médicales dans un souci de transmission de données.

Enfin, le rapport s'intéressera à la conception et à la construction des prototypes périodiques et continus, à leur désinfection, et aux tests effectués pour tester leur efficacité.

Il convient de préciser que l'ensemble des codes développés tout au long de ce projet sont disponibles dans un répertoire créé spécialement à cet effet¹.

1. Celui-ci est disponible au lien suivant : <https://gitlab.biomed.ulb.ovh/biomedd3/biomed3>

Méthodologie de groupe

2.1 Gestion de projet

Pour tout bon déroulement d'un projet, il est nécessaire de prendre du temps pour s'organiser et se structurer. Il faut ainsi définir les différentes tâches et se rendre compte des différentes contraintes qui sont imposées comme le budget, le nombre de personnes au sein du groupe, l'expertise des membres et le temps. Pour cela, plusieurs outils ont été mis en place :

Le diagramme de Gantt

Un diagramme de gantt est un planning permettant de connaître l'état d'avancement du projet. Il sera possible de visualiser plus facilement les différentes tâches préalablement attribuées et mettre les deadlines imposées. Il a été modifié hebdomadairement en fonction des retards et de l'avancement des membres (voir en exemple celui du 2ème quadrimestre en annexe A.1).

Pv's et Status report

Pour ne pas perdre toutes les informations évoquées lors des réunions hebdomadaires, il a fallu résumer ces réunions dans un fichier.

Au premier quadrimestre, une simple retranscription de chaque réunion a été faite. Ensuite, au 2ème quadrimestre, un template nous a été donné pour avoir un résumé de réunion plus clair et plus concis à l'aide d'un powerpoint. Il est composé de plusieurs slides : l'ordre du jour, une photo du diagramme de gantt en cours, un récapitulatif des activités faites et à faire, une visualisation des risques et problèmes sous forme de plan et un résumé des discussions importantes (voir exemple en annexe : A.2).

Gestion risques et problèmes

Tout au long d'un projet, il est possible, voire inévitable, que des problèmes surviennent. L'un des moyens pour minimiser au mieux leur impact est d'essayer de les prédirer avant même qu'ils ne surgissent pour pouvoir déjà leur trouver des solutions. C'est donc ce qui a été fait. Dans ce cas-ci, il était important d'y consacrer du temps chaque semaine pour contrôler leurs effets sur le projet. Au 2eme quadrimestre, un template en excel nous a été fourni pour y résumer tous les risques et problèmes dont le groupe était confronté, ainsi que sa probabilité de chance et son impact (voir annexe A.4). En outre, une slide dans le status report a été aussi consacrée à cette partie. Au 1er quadrimestre, un tableau par réunion a été réalisé mais ne comportait pas la probabilité de chance et l'impact possible de ces problèmes et risques (A.4).

Budget plan

Le projet avait un certain coût. Cette année, le budget était de 100 euros. Cependant, deux kits ont été fournis aux étudiants, un pour le prototype périodique et un autre pour le continu, ce qui a rabaissé le budget à 70€. Il était donc nécessaire de faire un plan du budget pour voir où se focaliseraient les dépenses et s'assurer que ce budget ne serait pas dépassé. De ce fait, le prix était un paramètre non-négligeable dans nos différents choix (voir chapitre 3.2). Voici un tableau résumant nos différentes dépenses 2.1 :

Achats :	Prix :
Capteurs de température (STS35)	2 x 12 euros
Capteurs de pouls (MAX30102)	1 x 1.48 euros
Capteurs de courant (INA3221)	2 x 1.40 euros
Brassard de sport	2 x 3.17 euros
Batterie	25 euros
Quantité de PLA pour imprimante 3D (+/- 0.2 Kg)	4euros
Reste :	6.38 euros

TABLE 2.1: Budget plan

2.2 Fonctionnement de groupe

2.2.1 Communication et organisation

Tout au long de l'année , à cause de la crise sanitaire, il a fallu trouver une manière de communiquer. Toutes les réunions se sont donc tenues sur Microsoft Teams de manière hebdomadaire

sur un temps de midi. À chacune d'elles, un PV/Status report, une mise à jour du diagramme de gantt et une analyse des risques et problèmes ont été établis (2.1) . Ces fichiers ont été placés sur le groupe teams du groupe. Depuis le départ, tout autre document est soigneusement classé sous divers dossiers dans un Drive, facilitant l'organisation et la recherche d'informations déjà traitées.

2.2.2 SWOT

Deux SWOTS ont été réalisés, un par quadrimestre. Le SWOT est un outil très utilisé en entreprise. Il permet d'avoir une vue d'ensemble de la situation actuelle. Il consiste en une évaluation comprenant les forces (Strengths), les faiblesses (Weaknesses), les opportunités (Opportunities) et les menaces (Threats) permettant de maximiser les points positifs et remédier aux points négatifs [App19]. La première analyse a été faite en semaine 12, en décembre 2020 (2.1) :

Strengths (Forces internes) :	Weaknesses (Faiblesses internes) :
<ul style="list-style-type: none"> - Bonne répartition du travail - Ponctualité aux réunions - Bonne communication - Persévérance et bonne cohésion 	<ul style="list-style-type: none"> - Motivation plus faible avec le confinement - Lacunes en informatique pour certains - Non-respect des deadlines imposées par le groupe - Sujet inédit
Opportunities (Opportunités) :	Threats (Menaces) :
<ul style="list-style-type: none"> - Pas mal d'informations accessibles sur internet - Aide extérieure (famille, tuteur, ...) - Matériel à disposition 	<ul style="list-style-type: none"> - Covid-19 : confinement - Charge de travail extérieur énorme - 2^e quadrimestre plus court que le 1^{er}

FIGURE 2.1: Swot : 1er quadrimestre

De ce swot, voici les différentes observations qui ont été faites :

1. Le non-respect des deadlines imposées par le groupe peut s'expliquer par un petit manque de motivation lié au confinement et par le fait que le sujet est inédit. Cela demande donc plus de travail et de temps d'adaptation.
2. Malgré ces lacunes et cette motivation qui veut tendre à baisser : la persévérance du groupe, sa bonne cohésion et la bonne communication entre ses membres sont des atouts forts permettant de rester actifs.

Ainsi, la motivation du groupe est un agent primordial dans cette crise sanitaire qu'il a fallu optimiser au maximum.

La deuxième analyse a été faite en semaine 27, en février 2021 (2.2) :

Strengths (Forces internes) :	Weaknesses (Faiblesses internes) :
<ul style="list-style-type: none"> - Bonne communication et bonne entente - Bonne organisation - Disponibilité de chaque membre du groupe - Détermination des membres - Team building - Régularité 	<ul style="list-style-type: none"> - Les problèmes de batterie - État de santé mentale - Théorie statistique peu utilisé (monitoring) - Précipitation à certaines tâches - Pas assez polyvalent - Non respect de différentes deadlines imposés par le groupe
Opportunities (Opportunités) :	Threats (Menaces) :
<ul style="list-style-type: none"> - Ressources extérieurs des membres - Projet intéressant - Tuteur très disponible et compréhensif - Ressources matériels (imprimante 3D, multimètre, fer à souder,...) - Possibilité de communiquer avec les autres groupes 	<ul style="list-style-type: none"> - Covid - Charge de travail - Temps limité et deadlines imposées - Outils à distance - Confrontation théorie- pratique

TABLE 2.2: Swot : 2ème quadrimestre

Au 2ème quadrimestre, le groupe a pu faire un team-building pour poser une ambiance agréable, ce qui a valu une bonne coopération dans le groupe (une bonne entente, communication et une bonne régularité).

Cependant, le temps limité et l'impact de la Covid a amené son lot de difficulté :

- La charge de travail dans un temps très court a causé une même difficulté que le 1er quadrimestre : le non-respect des deadlines imposés par le groupe. Mais aussi un manque de temps pour l'optimisation de certaines tâches.
- L'impossibilité des réunions de projet en présentiel a eu comme conséquence une non-polyvalence de la part de chaque membre du groupe (Par exemple : le prototype devait resté chez un membre du groupe et ne pouvait pas être partagé pour éviter au minimum les contacts).
- Grâce à certains membres du groupe, des ressources matérielles et des aides extérieures ont pu être utilisées pour le projet. Toutefois, l'utilisation de ces ressources à distance restait tout de même compliqué.

Symptômes et choix des capteurs

3.1 Symptômes de la Covid-19

Les recherches effectuées ont permis de dresser une liste aussi exhaustive que possible des différents symptômes liés à la maladie du virus SARS-CoV-2, reprise ci-dessous (3.1). Il a été décidé, par soucis de clarté et de compréhension, de diviser les symptômes en trois catégories : les *symptômes principaux*, qui sont les symptômes les plus représentatifs de la Covid-19 et qui, sans autre cause évidente, sont significatifs d'une infection au virus ; les *symptômes secondaires*, qui sont des symptômes liés à la Covid-19 mais qui peuvent également provenir de beaucoup d'autres causes, sont les symptômes qui ne sont à considérer comme significatifs d'une infection au virus que lorsqu'ils sont plusieurs à être présents et, finalement, les *symptômes propres aux personnes âgées*, qui sont les symptômes de la Covid-19 apparaissant uniquement chez les personnes âgées. [Le-20] Selon une étude américaine [Lar+20], l'ordre d'apparition des symptômes le plus probable est le suivant : fièvre, toux, nausées/vomissements et diarrhée. Mesurer une apparition de fièvre est donc le meilleur moyen d'être rapidement au courant d'une infection à la Covid19 et d'éviter la contamination de tiers.

2021 a sonné l'arrivée de nouveaux variants du virus qui doivent donc également pouvoir être détectés par l'appareil. Il appert que les symptômes du variant anglais de la Covid19 - les personnes touchées par le Sars-Cov-2 et par le variant anglais constituent l'immense majorité des malades de la Covid19 - sont, heureusement pour le projet, les mêmes que ceux de la maladie traditionnelle et ne diffèrent que par leur intensité ou leur fréquence d'apparition. Les symptômes mesurés par les présents dispositifs, à savoir la fièvre, les problèmes de pouls et les problèmes d'oxygénation du sang, ne semblent pas différer entre ces deux variants.

Symptômes principaux	Symptômes secondaires	Symptômes personnes âgées
Perte de goût et/ou d'odorat	Douleurs musculaires	Confusion brutale
Douleurs thoraciques	Nez bouché ou qui coule	Assoupissement
Fièvre	Maux de gorge	Se tromper régulièrement
Toux	Maux de tête	Troubles de la mémoire
Difficultés respiratoires	Fatigue	Chutes soudaines
	Perte d'appétit	
	Diarrhée aqueuse	

TABLE 3.1: Liste des symptômes de la Covid-19 [Le-20] [www20] [Jin+20]

Après examen de cette liste , il a été décidé de mesurer deux indicateurs physiologiques avec le prototype périodique et trois, dont un d'imposé, avec le prototype continu : la température et le pouls d'une part et la température, le pouls et le taux d'oxygénation du sang de l'autre. Ces deux données sont faciles à mesurer avec des capteurs peu onéreux là où la toux, par exemple, si on voulait la mesurer avec un microphone, demanderait des algorithmes dont la conception requiert bien plus de temps et de moyens que ce dont nous disposons.

Il a été également décidé que, bien que les mesures prises par le capteur de pouls seraient utilisées pour surveiller l'avancement de la maladie, car tout problème corporel ou physiologique amène une modification du pouls, l'intérêt principal du capteur serait, en collaboration avec les autres capteurs, de vérifier si le patient porte ou non le kit de dépistage.

Enfin, pour augmenter la précision et le nombre de données récoltées, un questionnaire sera soumis régulièrement au patient pour surveiller l'évolution des symptômes trop difficiles à mesurer avec des capteurs dans le cadre de ce projet.

3.2 Choix des capteurs

Dans le cadre de ce projet, il a été demandé de créer un dispositif médical capable de détecter et de mesurer différents paramètres physiologiques chez une personne. Dans un contexte lié à la Covid-19, l'appareil doit pouvoir détecter l'apparition d'au moins un symptôme chez une personne ayant eu un contact récent avec un patient testé positif, de suivre l'évolution et l'aggravation éventuelle de ce(s) symptôme(s) à son domicile au cours du temps afin d'alerter, si nécessaire, les équipes médicales et de suivre le respect des mesures de quarantaine imposées au patient.[Erca]

Pour répondre à cet objectif, des capteurs seront utilisés pour mesurer les différentes informations nécessaires à l'analyse de l'évolution des symptômes. Selon Techniques de l'Ingénieur [PK20], *le capteur (en anglais sensor ou transducer) est un dispositif qui fournit un signal électrique proportionnel à la grandeur physique à mesurer.* Il existe aujourd'hui une multitude de capteurs qui permettent de mesurer tout un ensemble de grandeurs physiques tels que les capteurs de température ou de vibration utilisés en industries. L'idée, ici, est de trouver des capteurs capables de détecter et de mesurer les différents symptômes de la Covid-19 pour pouvoir poser le bon diagnostic.

Dans la recherche des capteurs à implémenter, certaines conditions du cahier des charges devaient être respectées. En effet, aucune des dimensions du prototype ne doit dépasser 10 cm, à l'exception éventuelle d'un ou deux capteurs qui pourraient être déportés par un fil, le tout en minimisant la gêne du patient dans ses mouvements. L'idée est donc d'avoir la majorité des capteurs à l'intérieur du boîtier, soudés à l'ESP32, et 1 ou 2 capteurs qui jouiront si nécessaire d'une extension filaire.[Erca]

De plus, le budget pour la réalisation de ce projet était limité à 100€ pour tous les frais liés à ce dernier. Une trentaine d'euros en ont déjà été soustraits pour l'achat des ESP32 fournis dans les kits, il reste donc 70€ dans le budget pour acheter le reste. L'autonomie du prototype périodique doit être la plus longue possible pour parvenir à fonctionner pendant toute la durée de la quarantaine, soit 14 jours. Il en va de même pour le prototype continu qui doit fonctionner au moins 8h avec une batterie de 1500 mAh[Erca]. Les capteurs doivent également être efficaces dans la mesure où ils doivent fournir des données physiologiques correctes et précises sur l'état du patient pour ne pas fausser le diagnostic. Enfin, une librairie doit être disponible pour utiliser ces capteurs. Le choix des capteurs dépendra donc de quatre paramètres : efficacité, consommation, utilisabilité (c'est à dire le fait de pouvoir utiliser dans notre cas le capteur grâce à une librairie fournie) et prix.

Pour finir, expliquons la démarche suivie pour le choix des capteurs. Il a fallu dans un premier temps s'intéresser à l'ESP32 (pour plus d'informations, voir 5.1) et aux différentes entrées que peut recevoir ce dernier. La fiche technique du constructeur indique que le microcontrôleur se compose de 48 pins pour connecter les capteurs avec différents types d'entrées (ADC,I2C,TOUCH etc.)[ESP20b ; ESP20a] Dans les capteurs choisis pour ce projet, tous utilisent le protocole I2C. Cela permet ainsi d'éviter des problèmes de signal des entrées ADC de mauvaise qualité sur l'ESP32.[Ercb] Il a fallu ensuite analyser les types de capteurs disponibles sur le marché. Un guide, écrit par Rui et Sarah Santos, met à disposition les capteurs les plus connus et utilisés pour l'Arduino, et par extension à l'ESP32[SS20]. Sur base de cela, le site Octopart a été utilisé pour choisir les capteurs selon les critères précédemment cités. Celui-ci fournit directement les datasheets des composants et offre une comparaison sur base de différents critères tels que le prix, la disponibilité, le matériau utilisé, la précision etc[Oct20]. L'achat final du capteur s'est fait sur un ensemble de sites tels qu'Amazon, Aliexpress, Banggood etc en prenant en compte la disponibilité, le prix et le délai de livraison des capteurs.

Lien du comparatif des capteurs

Pour accéder au fichier Excel, vous pouvez appuyer ici ou aller directement à l'adresse suivante :

<https://drive.google.com/file/d/1S66HvFSxn51M9e9kYzS2ZV8uelaFqwXJ/view?usp=sharing>

3.3 Capteur de température

Un des principaux symptômes liés à la Covid-19 est la fièvre, où le corps augmente sa température pour détruire les corps étrangers. Pour détecter ce symptôme, la précision de la mesure a été privilégié sur les autres paramètres. En effet, les capteurs de température se distinguent par leur degré de précision allant de $\pm 0.1^\circ\text{C}$ pour les plus précis à $\pm 1^\circ\text{C}$ pour les plus imprécis. En choisissant de maximiser la précision, 3 capteurs de température disponibles sur le marché répondaient parfaitement aux critères sélectionnés (Table 3.2) :

Capteurs	Précision	Consommation veille/actif	Tension	Librairie	Prix
TMP117	0.1°C	150 nA / 3.5 µA	1.8 V à 5.5 V	Disponible	23 euros/pc
STS35	0.1°C	0.2-20 µA / 600-1500 µA	2.15 V à 5.5 V	Disponible	12 euros/pc
Si7051	0.1°C	0.06-0.62 µA / 3.5-4 mA	1.9 à 3.6 V	Disponible	15 euros/pc

TABLE 3.2: Comparatif des capteurs de températures[Texc] [Sena] [Sil]

Le STS35 a finalement été choisi (voir Figure 3.1), son prix étant plus avantageux et la livraison plus rapide que pour les 2 autres (1 semaine). Ce capteur consomme très peu de courant, fournit des mesures précises et est accompagné d'une librairie pour coder sur le microcontrôleur, ce qui remplit parfaitement les critères de sélection.

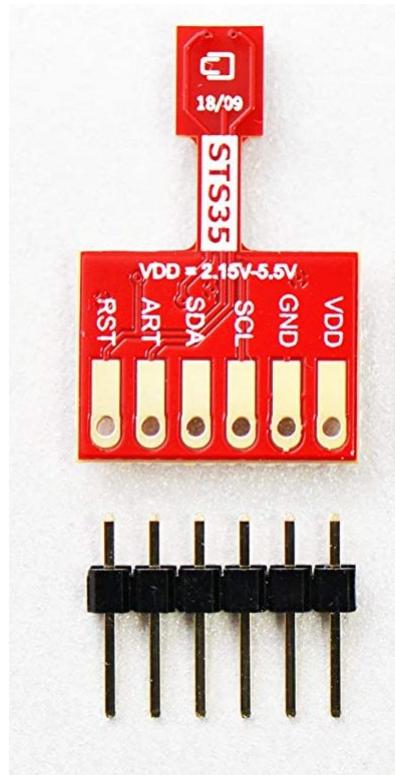


FIGURE 3.1: Capteur de température : STS35[Senb]

3.4 Capteur de pouls

Le pouls est une donnée physiologique importante pour détecter toute anomalie corporelle. Un capteur de pouls sert à mesurer la fréquence des battements du cœur du patient et à détecter par la même occasion le port du dispositif par ce dernier tout au long de sa quarantaine. Sur le marché, les capteurs de pouls se différencient par leur fonctionnalités supplémentaires telles qu'un oxymètre ou un électrocardiogramme intégrés et les prix varient en fonction de ces critères. Pour le dispositif continu, les responsables du projet fournissent un oxymètre bluetooth avec un capteur de pouls intégré. Il a été décidé d'acheter un capteur de pouls simple pour le prototype périodique qui n'en dispose pas initialement. 5 capteurs répondent aux critères sélectionnés (Table 3.3) :

Capteurs	Fonctionnalité	Consommation veille/actif	Tension	Librairie	Prix
MAX86150	Pouls/Oxymètre/ECG	0.7 µA / 20 mA	3.1 à 5 V	Disponible	27 euros/pc
SEN-15219	Pouls/Oxymètre	Non trouvée / 100mA	1.71 à 3.63 V	Disponible	34 euros/pc
MIKROE-2000	Pouls/Oxymètre	0.7-10 µA / 0.6-1.2 mA	1.7 à 5.0 V	Disponible	22 euros/pc
Pulsesensor.com	Pouls	Datasheet non disponible	Non trouvée	Disponible	6.01 euros/pc
MAX30102	Pouls/Oxymètre	0.7-10 µA / 0.6-1.2 mA	1.7 à 5.0 V	Disponible	1.47 euros/pc

TABLE 3.3: Comparatif des capteurs de pouls[Maxb] [Spa] [Mik] [Maxa]

Après discussion, le MAX30102 (voir Figure 3.2) a été choisi pour le prototype du kit périodique. Il dispose d'une faible consommation et est couramment utilisé dans les projets Arduino. Le rapport efficacité/prix est conforme à ce qui était recherché et ce capteur remplit donc parfaitement les critères de sélection.

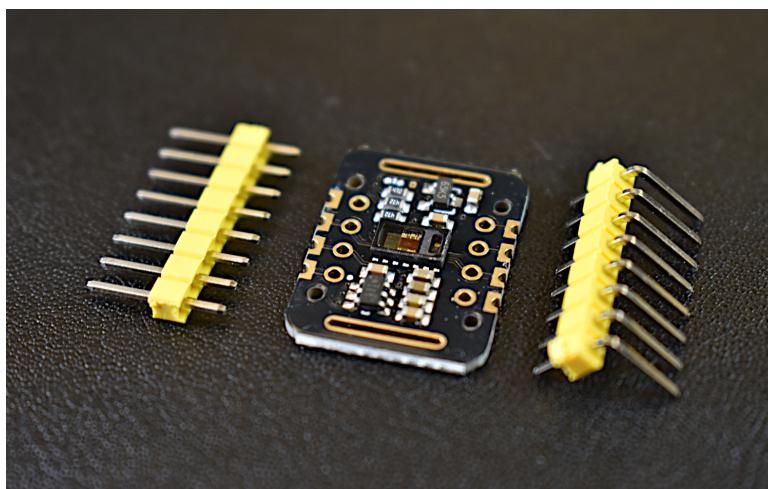


FIGURE 3.2: Capteur de pouls : MAX30102[Maxc]

3.5 Capteur de courant

Le courant et la tension sont deux données très importantes à connaître lorsqu'on s'intéresse à la consommation et directement à l'autonomie d'un appareil électrique. Dans ce cadre-ci, une estimation théorique de l'autonomie des prototypes sera demandé et pour vérifier cela en pratique, une connaissance de la tension et du courant utilisés permettra de lier pratique et théorie. Deux capteurs ont été trouvés intéressants à implémenter dans les différents prototypes pour mesurer ces valeurs (Table 3.4) :

Capteurs	Plage de courant / tension	Consommation veille/actif	Tension	Librairie	Prix
INA219	3.2 A / 0V à 26V	1 mA	3 V à 5 V	Disponible	0,98 euros/pc
INA3221	1.6 A / 0V à 26V	350 µA	2.7 V à 5 V	Disponible	1,40 euros/pc

TABLE 3.4: Comparatif des capteurs de courant (et de tension)[Texa] [Texb]

Le choix final a été porté sur le INA3221 (voir Figure 3.3) car il dispose de 3 channels pour la prise de tension et de courant, contrairement au INA219 qui n'en dispose que d'un, avec une différence de prix minime. De plus, il est beaucoup plus performant sur le long en terme de consommation car il ne consomme que 350 µA, ce qui très avantageux pour le prototype périodique où chaque mA compte.

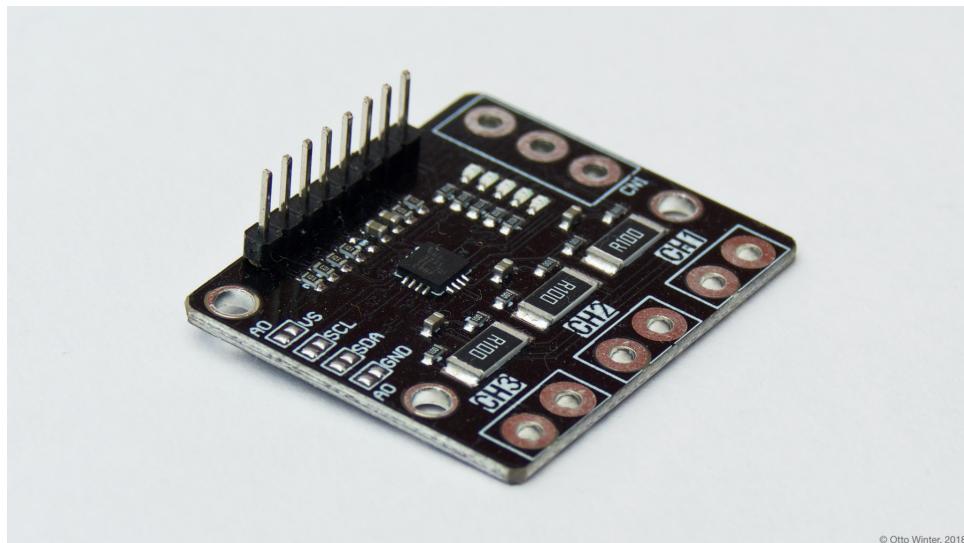


FIGURE 3.3: Capteur de courant : Ina3221[Texd]

3.6 Oxymètre

Pour finir, un oxymètre bluetooth a été fourni pour le prototype continu afin d'obtenir la saturation d'oxygène dans le sang et la pulsation cardiaque (voir Figure 3.4). Celui-ci va permettre de connecter le prototype continu avec lui-même en une connection BLE (Bluetooth Low Energy) afin de récolter les paramètres physiologiques cités juste avant. Notons par ailleurs qu'il faut deux piles AAA pour alimenter l'oxymètre.



FIGURE 3.4: Oxymètre de pouls : OX-831[Pul]

3.7 Récapitulatif

Symptômes Prototypes	Fièvre	Pouls	Taux d' O_2 dans le sang
Prototype périodique	STS35	MAX30102	X
Prototype continu	STS35		Oxymètre

TABLE 3.5: Tableau comparatif des capteurs utilisés pour chaque symptôme

4

Emplacement des capteurs

Une fois les capteurs choisis, il convient de déterminer l'emplacement de ceux-ci de façon à optimiser le confort du patient et la précision des mesures.

Capteur de température

Le but de ce capteur est de relever la température centrale du corps interne. Cependant, cette donnée-là ne peut pas être identifiée sur n'importe quelle partie du corps. En effet, certaines parties du corps rendent mieux compte de la température centrale car elles sont un meilleur indicateur de la chaleur interne. De ce fait, le capteur devrait mesurer des données plus réalistes s'il est placé dans des parties telles que l'oreille, le rectum et les aisselles [Bla20].

Toutefois, ces endroits ne sont pas propices au port d'un dispositif et donc d'un capteur. Effectivement, le placement d'un dispositif à ces endroits-là est matériellement impossible. Similairement, un capteur à ces endroits-là provoqueraient une gêne chez l'utilisateur : le patient serait encombré par le capteur lui-même et par le fil qui relie le capteur au dispositif. En outre, cette gêne pourrait provoquer un dysfonctionnement du dispositif (Par exemple : si l'utilisateur tire sur le câble).

Il était plus sûr de mettre le capteur aux abords du dispositif.

Capteur de pouls

Une explication semblable vaut pour le capteur de pouls. Pour optimiser la mesure du rythme cardiaque, il conviendrait de mettre le capteur dans des zones plus vascularisées, comme la main, ce qui provoquerait à nouveau une gêne non-négligeable pour le patient [Aud21]

L'Oxymètre

L'oxymètre était fourni par l'université. Il est conçu de telle manière à devoir être porté au doigt. Il n'y a donc pas de choix à faire ici. Cependant, ceci n'est pas problématique, puisqu'il fonctionne en bluetooth. Il ne constraint donc pas le placement du dispositif et des autres capteurs.

Décision de l'emplacement

Puisque l'oxymètre n'intervient pas dans le placement des autres capteurs et de l'enveloppe grâce à sa fonctionnalité bluetooth, seul le capteur de température et de pouls sont décisifs.

Afin d'éviter des câbles trop longs qui pourraient gêner l'utilisateur, et donc le conduire à endommager le prototype, il fut décidé de placer les deux capteurs à proximité de l'enveloppe. De plus, il fut décidé de maximiser le confort du patient, qui devra malgré tout porter le dispositif pendant au moins deux semaines. De ce fait, le groupe a choisi de placer le tout sur le bras. Un calibrage des capteurs pourra être effectué ultérieurement afin de contrer le manque de réalisme dû à l'emplacement (voire section 15.1).

Microcontrôleur

5.1 l'ESP32

L'ESP32 est un microcontrôleur, c'est-à-dire un circuit intégré composés d'un processeur, de différents types de mémoires et d'un système d'entrées-sorties [La-ND]. Ces spécificités principales sont la capacité de connexion à d'autres appareils par le biais d'un réseau WiFi ou de Bluetooth.

Pour alimenter le prototype en tension lorsqu'il est porté par un patient, une batterie Lipo est connectée à l'aide d'un connecteur à deux broches + et - sur l'ESP32 [MCU21]. Ce type de connectique est très pratique car il permet de rajouter une batterie au circuit sans nécessité de soudure. Durant l'entièreté de l'élaboration du projet, le port micro-USB a été utilisé pour téléverser les différents codes sur l'ESP32 mais a aussi permis de l'alimenter en tension. Enfin, c'est à l'aide de ce même port que la batterie pourra être rechargée.

En effet, lorsqu'un ordinateur est connecté à l'ESP32, ce dernier sert d'intermédiaire entre la source de tension et la batterie, ce qui permet sa recharge sans devoir la déconnecter du circuit.

L'ESP32 est un microcontrôleur muni d'un bus informatique, une "*voie d'accès numérique partagée entre les différents composants d'un circuit électronique.*" (PCMag), de type I²C, pour Inter-Integrated Circuit. Ceci signifie que le bus hiérarchise les différents composants du circuit en un composant 'maître', ici le microcontrôleur, et en composants 'esclave', qui reçoivent leurs ordres et communiquent avec leur maître et uniquement avec lui. Cette communication se fait au travers de deux câbles, un câble SDA, pour Serial Data, utilisé pour échanger les données, et un câble SCL, pour Serial Clock, utilisé pour synchroniser les échanges.

Ce type de communication est utilisé pour connecter les capteurs de température et de courant dans les deux prototypes ainsi qu'un capteur cardiaque dans le prototype périodique.

En ce qui concerne le bouton poussoir et la LED, ils seront connectés à deux autres pins. ces dernières définies comme "OUTPUT" pour la LED et "INPUT" pour le bouton. Ce procédé sera détaillé dans le chapitre 13

Enfin, les différents composants seront reliés à la terre par la pin GND et à la tension par la pin 3V3

5.2 Choix du langage de programmation

Au lancement de ce projet, deux langages de programmation ont été proposés pour réaliser les codes qui seront téléverser dans les deux microcontrôleurs, le C++ et le MicroPython. Ces deux derniers possèdent chacun des avantages et des inconvénients détaillés dans le tableau ci-dessous¹ (5.1) :

Arduino(C++)	MicroPython
Exécution plus rapide du code	Exécution moins rapide du code
Élaboration du code un peu plus compliquée car la compilation est assez lente	Le code se compile assez rapidement, ce qui facilite les modifications
Plus compliqué de déboguer le code	Instructions assez faciles lors d'un problème, ce qui permet de déboguer plus rapidement
Aucun apprentissage académique du langage bien que certains membres de l'équipe possèdent déjà des notions de base	Langage appris lors du cours de BA1 INFO-H100 et donc maîtrise correcte de la part de chaque membre du groupe
Grand nombre de bibliothèques de code pour les capteurs ainsi que pour la connexion à un réseau WiFi ou à une base de données	Il est plus compliqué de trouver des exemples de codes en MicroPython pour les différentes parties nécessaires
Code assez instinctif car très visuel, ce qui rend plus facile le découpage du code en différentes étapes	Visuel du code très compact ce qui rend plus difficile à analyser et interpréter
La communauté travaillant sur Arduino semble plus enclue au partage d'informations et de projets	Des exemples de codes et de projets semblent plus difficiles à trouver en MicroPython

TABLE 5.1: Comparaison des langage C++ et MicroPython pour le codage de l'ESP32

1. sur base du QuicKStart-3 (2020) fourni par Rudy Ercek via l'UV

Le choix de l'équipe s'est finalement porté sur le langage C++. Malgré sa connaissance un peu moins profonde, il offre un plus grand choix de bibliothèques et d'exemples de codes, ce qui a permis une élaboration plus rapide.

En ce qui concerne les problèmes pour déboguer le code, il existe de nombreux forums qui aident à régler les petits problèmes rencontrés.

6

Protection des données

Cette année, le projet biomédical demande une collecte de données à caractère personnel. Ces informations étant enregistrées et transmises, il est nécessaire de respecter les normes relatives au respect de la vie privée (*RGPD*)¹. C'est un règlement de l'Union Européenne qui peut entraîner des sanctions s'il n'est pas respecté. Des recherches ont donc été effectuées. Cependant, le faire en pratique sort du cadre du projet. De ce fait, il s'agit ici que de simples recherches théoriques qui ne furent pas appliquées là où cela est normalement indispensable.

6.1 La RGPD

Les informations à caractère personnel d'un individu sont toutes les informations qui le rendent identifiable directement ou indirectement auprès d'une tierce personne. Dans le cadre de ce projet, il s'agit du nom, du prénom et de l'adresse e-mail de l'utilisateur. Ce sont donc ces informations qui seront masquées.

Les informations restantes, c'est-à-dire les valeurs de température, pouls et taux d'oxygène, étant traitées et enregistrées dans la base de données, la RGPD est également d'application et il est donc obligatoire que la personne concernée donne son consentement explicite sur le stockage de ses données médicales mesurées [Aut20b].

En raison de la RGPD, chaque personne concernée a un nombre de droit qu'il faut être capable de respecter (6.1) :

1. RGPD = Règlement Général sur la Protection des Données.

Droits du citoyen	Explication
Droit à l'information	La personne concernée doit en être obligatoirement informée.
Droit d'accès	La personne concernée peut accéder à ses données n'importe quand et même demander une copie pour vérifier si elles sont correctes.
Droit à la limitation	Dans certaines circonstances, la personne concernée peut limiter le traitement de données. C'est à dire que les données peuvent être stockées mais que toute autre activité de traitement doit être arrêtée.
Droit d'opposition	La personne concernée peut s'opposer à la collecte de données à caractère personnel si le traitement repose sur ces bases juridiques : l'intérêt légitime et l'exécution d'une mission d'intérêt public ou relevant de l'exercice de l'autorité publique.
Droit de rectification	La personne concernée peut demander à rectifier ses données si elles comportent des fautes.
Droit à la portabilité des données	La personne concernée a le droit d'obtenir ses données pour les réutiliser pour d'autres services.
Droit à l'effacement	La personne concernée peut exiger l'effacement des données s'il n'y a plus de raisons valables de les traiter.
Droit de ne pas être soumis à une décision individuelle automatisée	La personne concernée ne peut pas faire l'objet d'une décision entièrement automatisée, Au risque d'affecter cette personne ou d'avoir des effets juridiques.

TABLE 6.1: Les droits des citoyens rendus exercables pour un individu par la RGPD [Aut20a]

6.2 Cryptage de données

Le but du cryptage est de rendre un texte totalement illisible par une tierce personne pour avoir un échange le plus sécurisé possible. Pour cela, quelques techniques ont été élaborés tout au long des décennies dans le but d'optimiser cette sécurisation.

Choix de la méthode de cryptage

Il existe plusieurs méthodes de cryptage. Il a donc fallu les comparer pour trouver laquelle était la plus sûre. Le tableau ci-dessous montre que le cryptage hybride semble être le plus efficace à utiliser. (6.2) :

Méthodes de cryptage	Avantage	Inconvénient
Symétrique	Rapide à déchiffrer	Même clé pour crypter et décrypter. Si un pirate arrive à l'encoder, il arrivera à comprendre tous les messages qui suivent.
Asymétrique	Deux clés différentes : une clé publique et une clé privée qui reste secrète et n'est pas partagée.	Lent, ça prend plus de temps à déchiffrer.
Hybride	Mélange de cryptage symétrique et asymétrique : on prend le meilleur de chacun. C'est donc le plus sécurisé et le plus efficace.	Difficile d'exploitation

TABLE 6.2: Tableau de comparaison des différentes méthodes [RD12]

Cryptage hybride

Cette technique de cryptage utilise les deux méthodes ; un cryptage asymétrique pour le premier échange et un cryptage symétrique pour les échanges suivants où le serveur et le client auront tous les deux une clé secrète.

Parmi les différents algorithmes utilisés pour le cryptage asymétrique, les plus connus sont l'algorithme RSA . Il se base sur l'utilisation des nombres premiers et une propriété qui sera fondamentale à son fonctionnement.[Tou07]

Pour bien comprendre le principe du cryptage hybride, voici une image permettant de comprendre cela de manière plus clair (6.1) :

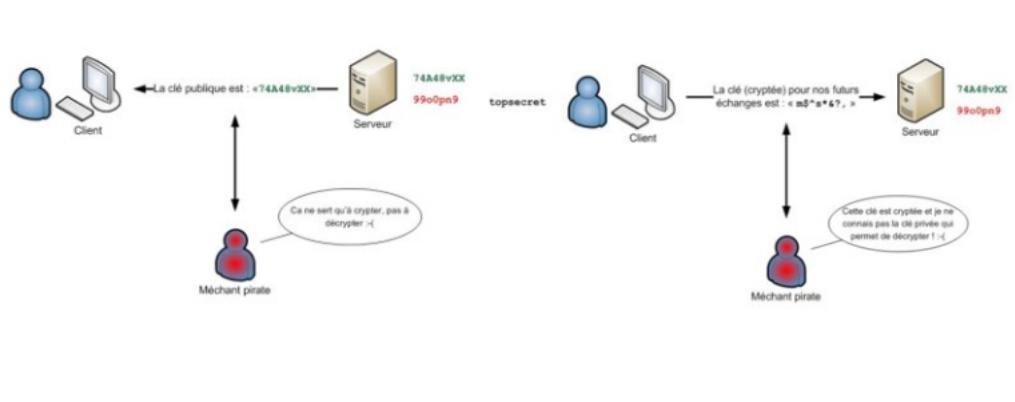


FIGURE 6.1: Fonctionnement du cryptage hybride [RD12]

Au départ, le serveur crée une clé secrète et une clé publique. Seule la clé publique sera partagée avec les autres. Une personne pourra donc crypter un message. Ce qui est important à comprendre est que, dans le cas du cryptage asymétrique, un pirate sera incapable de décrypter le message ainsi crypté juste à l'aide de la clé publique (la complexité est beaucoup trop grande).

Le message que le client cryptera sera une nouvelle clé qui sera envoyé au serveur. Dans ce cas-ci, seul le serveur sera en capacité de déchiffrer le message à l'aide de sa clé secrète. Après cette étape primordiale, les deux ordinateurs ont connaissance d'une même clé : voici arrivé alors à une méthode symétrique rendant les échanges plus rapide et plus facile.

6.3 SSL

Le SSL (Secure Sockets Layer) est un certificat numérique permettant d'avoir une liaison cryptée entre un client et un serveur [Dig]. Il s'appliquerait dans notre cas au site Web et à l'ESP32.

Sans ce certificat, les échanges d'informations en http (*HyperText Transfer Protocol*) serait totalement visible par une tierce personne. Utiliser SSL permet donc d'avoir un tunnel sécurisé (Figure 6.2) reprenant la technique de cryptage hybride expliquée juste au-dessus (section 6.2). Le mot "HTTP" devient "HTTPS" signifiant que le site est sécurisé.

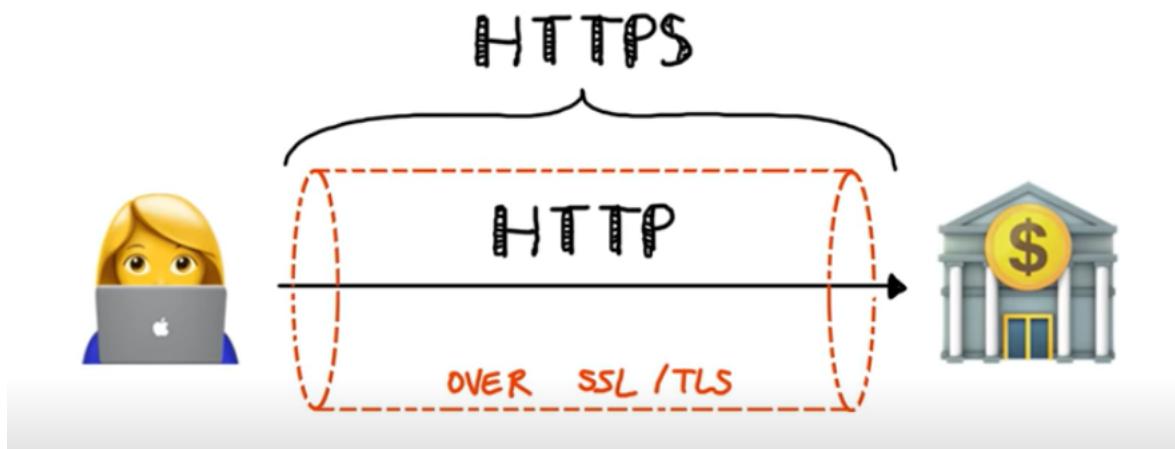


FIGURE 6.2: Représentation d'une interaction cryptée par SSL [Coo18]

Le site web, ayant été fait en local (voir chapitre 12), il n'est pas possible de sécuriser la communication sans devoir utiliser notre budget déjà assez limité.

L'implémentation du SSL dans l'ESP32 demande un mémoire trop importante. La valeur de la mémoire dépend de la configuration de celui-ci ([ESP]). Plusieurs forums ont évoquées une impossibilité à exploiter le SSL à cause d'un manque de mémoire ([Per19],[Kol17]). C'est pourquoi, étant donné que crypter était une partie secondaire du projet et qu'en plus, plusieurs réseaux étaient déjà mis en place sur l'ESP, il semblait plus astucieux de ne pas instaurer le SSL pour éviter des erreurs.

Base de données

Pour stocker les relevés de température, de pouls et d'oxygène envoyés par l'ESP32, il fut conseillé de travailler avec une base de données temporelle. Un serveur InfluxDB (version 1.8) a été créé à cet effet¹. Pour y accéder, un identifiant et un mot de passe ont été fournis par le tuteur.

7.1 Concepts clefs d'une base de données temporelle

L'utilisation d'InfluxDB nécessite de maîtriser certaines notions clefs d'une base de données temporelle. Celles-ci sont essentielles à la compréhension du mécanisme de stockage des relevés dans la base de données. Cette section expliquera donc ces notions[Inf20b].

Une base de données temporelle peut stocker différentes mesures comme, par exemple, la température et la fréquence cardiaque. C'est ce que l'on appelle le "measurement". Il s'agit donc du type de données observées. Il est fortement conseillé de choisir un nom évocateur pour le "measurement" afin d'éviter toute confusion.

Pour chaque "measurement", la base de données retiendra les différentes valeurs qu'un ordinateur lui envoie, et elle y associera sa date et son heure d'arrivée. Il s'agit ici du "timestamp", qui est exprimé en RFC3339 UTC. C'est bien cette association systématique entre valeur mesurée et heure d'arrivée qui caractérise une base de données temporelle.

Afin de faciliter l'utilisation de ces données, InfluxDB fait appel aux "field keys" et "tag keys". Le field key est une chaîne de caractère à laquelle on associe les données mesurées. On pourrait le voir comme une variable qui prend différentes valeurs au cours du temps. Ces valeurs sont les mesures

1. l'API de cette base de données est disponible à l'adresse suivante : <https://influx.biomed.ulb.ovh>

que reçoit la base de données et sont appelées "field values". Elles peuvent être de n'importe quel type : entier, chaîne de caractères, booléens,...

Les "tag keys" (et les "tag values" correspondantes) sont une information complémentaire de la donnée, qui permettent entre autre de la retrouver plus facilement. Il ne s'agit pas d'une information obligatoire, mais elle fait partie intégrante du classement des données.

Pour illustrer ces concepts, un exemple en rapport avec le projet peut être utile (7.1) :

timestamp	Jean	Emma	type de prototype
2021-02-14T06:12:00Z	69	72	périodique
2021-02-14T06:17:00Z	71	75	périodique
2021-02-14T06:23:00Z	70	71	périodique
2021-02-14T06:28:00Z	68	73	périodique
2021-02-14T06:33:00Z	73	67	périodique
2021-02-18T12:41:00Z	75	78	continu
2021-02-18T12:46:00Z	77	80	continu
2021-02-18T12:51:00Z	73	76	continu
2021-02-18T12:56:00Z	78	81	continu

FIGURE 7.1: Exemple fonctionnement InfluxDB

Le tableau ci-dessus indique le rythme cardiaque de Jean et Emma au cours du temps, en battements par minute (bpm). Celui-ci peut être mesuré avec le prototype périodique et le prototype continu.

Dans ce cas-ci, le "measurement", c'est-à-dire le type de mesures, est le rythme cardiaque. Le timestamp associé à chaque observation est repris dans la colonne "time". Les "field keys", dans ce cas-ci les personnes chez qui les mesures furent effectuées, sont Jean et Emma. À chaque mesure, on peut associer le type de prototype qui a effectué la mesure. Il s'agit du "tag key".

7.2 Queries

Comme son nom l'indique, une base de données permet de stocker des données pour une utilisation ultérieure. Aussi bien le stockage que la récupération de données se fait au moyen d'instructions spécifiques à chaque sorte de base de données, appelées "queries" [Jai18]. Ces instructions de base seront détaillées dans cette section.

Il convient de remarquer que, puisqu'aucun membre du groupe était familier avec le fonctionnement d'InfluxDB, ces instructions furent d'abord testées sur un serveur local, à même la

console.

Tout d'abord, il y a les queries en rapport avec la base de données même, c'est-à-dire "CREATE DATABASE" et "SHOW DATABASES"[Inf20d]. Celles-ci permettent respectivement de créer une nouvelle base de données et de voir sur le serveur quelles bases de données sont abritées. Ces queries ne seront pas utilisées dans le cadre de ce projet.

Une commande similaire, "SHOW MEASUREMENTS"[Inf20d], permet de visualiser au sein d'une base de données les différentes mesures, en l'occurrence les symptômes dont on relève des données.

Pour ajouter une donnée dans un "measurement", il faut suivre la procédure suivante[Inf20c] :

INSERT (measurement),FieldKey=(nom du fieldkey) value=(valeur),TagKey=(nom du tagkey) value=(valeur)

Lors de l'utilisation de cette query, il faut être prudent avec l'utilisation des virgules et des espaces : une erreur et la base de données n'accepte pas le relevé. Il faut également faire attention, pour chaque mesure, à utiliser à chaque fois le même schéma : les noms des TagKeys et des Field-Keys ne peuvent pas changer au fur et à mesure de l'encodage des données, sinon la base de données refuse le relevé.

Pour sélectionner les données d'une mesure, il suffit d'utiliser la commande suivante[Inf20a] :

SELECT * FROM (nom du measurement)

Cette commande est assez générale : il est possible de sélectionner certains tags, ou alors d'optimiser le temps de réaction en remplaçant * par "DATA".

7.3 Interaction avec la base de données Biomed3

Le travail sur une console est assez limité et ne permet pas d'automatiser l'enregistrement de données. Afin de systématiser cela, il a fallu recourir à un langage de programmation.

7.3.1 Choix du langage

À priori, n'importe quel langage de programmation aurait pu convenir. Initialement, le groupe pensait se tourner vers le C++, par souci d'homogénéité avec le codage de l'ESP32. Seulement, python s'est rapidement avéré plus simple, tout d'abord parce qu'il s'agit d'un langage maîtrisé par l'ensemble des membres du groupe, ce qui facilite la coopération. De plus, python est un langage réputé pour le traitement de données, et possède de surcroît des librairies spécifiques à InfluxDB.

7.3.2 Explication du code

Pour interagir avec la base de données, deux codes furent élaborés : une pour entrer des données fictives dans la base de données (ce qui servira dans le simulateur), et une pour les récupérer (nécessaire pour le traitement de données).

Cependant, même s'il s'agit de deux codes séparés, la structure est la même : il y a d'abord import de la librairie InfluxDB, puis connexion à la base de données. Ensuite, le code fait ce qu'il doit faire à proprement parler, à savoir le stockage de données fictives ou la récupération de données (fictives ou réelles cette fois).

Connexion à la base de données

Chaque fichier de code qui doit interagir avec la base de données doit d'abord s'y connecter². Puisque la procédure à suivre est toujours la même, il sera plus efficace de l'expliquer avant de passer à l'explication du code test élaboré (7.2).

```
#Import InfluxDBClient library
from influxdb import InfluxDBClient
from random import randint

#influxdb settings
host='influx.biomed.ulb.ovh'
db='biomed3' #replace by your database

#influxdb credentials
username='biomed3' #replace by your database user login
password='Or1nqhDW5ynBvs4a' #replace by your database user password

#init the influxdb client
client = InfluxDBClient(host=host, port=80, username=username, password=password, database=db)

#test db connectivity (not mandatory) and if it is ok, it should return influxdb version
client.ping()
```

FIGURE 7.2: Connexion à InfluxDB

2. Le code a été élaboré à partir des Quickstarts disponibles sur l'UV

Le code commence par la définition des constantes :

- les réglages de la base de données
 - le "host" : le serveur abritant la base de données du groupe
 - "db" : le nom de la base de données du groupe (BIOMED3)
- les données d'utilisateur : l'accès au serveur est limité, il faut donc introduire l'identifiant et le code fournis par le tuteur, sans quoi l'accès à la base de données est refusé.

Une fois ces constantes définies, il est assez aisé d'initialiser le client InfluxDB, qui permet d'accéder aux informations stockées dans la base.

Pour résumer, voici un schéma résumant cette partie (7.3) :

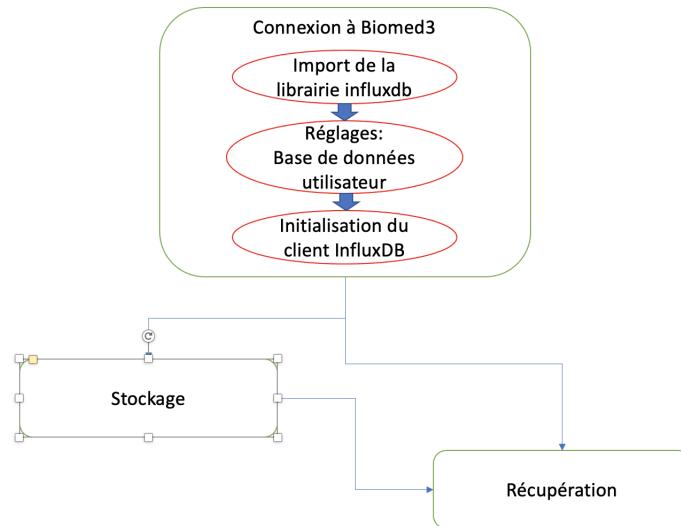


FIGURE 7.3: Schéma global

Stockage de données

Après avoir initialisé le client InfluxDB, le code initialise une boucle. A l'intérieur de celle-ci, des données fictives pour la température et le pouls sont générées puis écrites dans la base de données en associant chaque valeur à la mesure correspondante (soit le measurement). La boucle se termine par l'ordre "sleep" qui permet d'attendre un certain temps avant de générer et de noter les valeurs suivantes. Ceci simule le délai qu'il y aura entre les mesures prises par les capteurs sur le prototype (7.4).

```

for i in range(10):
    u= str(randint(30,50))
    line1='testgrafana,id=temperature data={}'.format(u)
    client.write_points(line1,protocol='line1') #write the line
    sleep(60)
}

```

FIGURE 7.4: Stockage des données

Récupération de données

Le code de la récupération des données est relativement court : après la connexion à la base de données, puis de déterminer le bon "query". Pour récupérer des données, on utilise le query "SELECT ... FROM". La requête la plus générale est "SELECT * FROM..." qui récupère toutes les données. Mais si l'on veut être plus précis, on peut par exemple ne sélectionner qu'un seul tag ou un seul type de valeurs (7.5).

```

query='SELECT "data" FROM testgrafana'      #testfor ou testgrafana
results=client.query(query)
points = list(results.get_points(measurement='testgrafana'))

```

FIGURE 7.5: Récupération des données

Le code transforme ensuite les données récupérées en liste : le temps, la température, l'oxygène,... afin de rendre leur utilisation plus facile (7.6).

```

temp = []
for i in range(len(points)):
    temp.append(points[i]["data"])

time = []
for i in range(len(points)):
    time.append(points[i]["time"])

```

FIGURE 7.6: Création de listes

8

Graphana

Grafana [Gra] s'est imposé comme l'outil le plus pratique pour visualiser les données obtenues par l'ESP32 et envoyées sur le serveur InfluxDB. Grafana est une application web qui peut jouer le rôle d'interface visuelle d'une base de données. Il a été recommandé par le guide du projet¹. En se connectant à l'aide des identifiants fournis par M. Foucart, une interface paramétrable selon les besoins du projet est accessible. De plus, Grafana s'actualise automatiquement et permet donc un suivi des données en temps réel.

8.1 Paramétrage du panel

Pour afficher des données, il faut d'abord créer un "panel". Un panel peut être vu comme un tableau où seront affichées les données. Ce panel peut prendre plusieurs formes : un graphique standard en 2D, un graphique en barre ou en camembert ou encore un compteur (comme ceux que l'on retrouve dans les voitures et qui affichent la vitesse de la voiture). Ensuite, après avoir créé le panel, il faut assigner des mesures que Grafana ira lire dans le serveur InfluxDB. Par exemple, soit la mesure 'Testgrafana' qui contient une liste de valeurs X à un moment donné, on attribue au couple (X;temps) un "id", par exemple 'Oxygène'. Cette mesure étant stockée dans InfluxDB, il faudra introduire dans la partie "Query" que l'on va venir prendre des mesures stockées dans 'Testgrafana' et qui ont pour id 'Oxygène'.

L'exemple suivant montre des valeurs prises dans 'Testgrafana' dont l'id correspond à 'Oxygen'
8.1. Après avoir pris des valeurs il ne reste qu'à paramétrier le type de graphique voulu.

1. Quickstarts disponibles sur l'UV

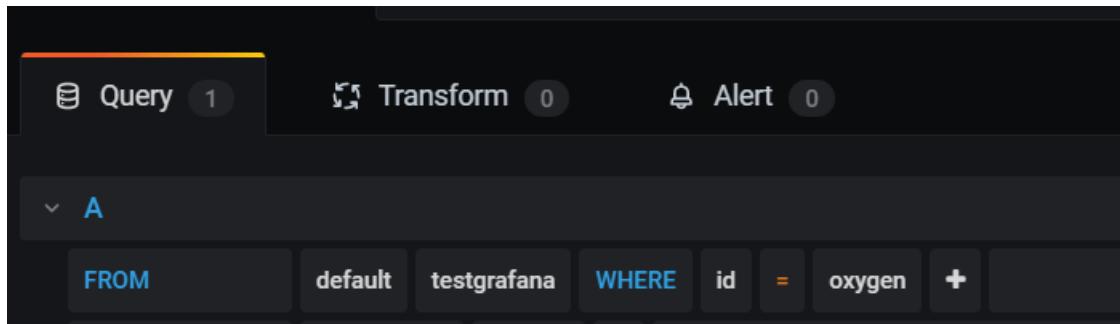


FIGURE 8.1: Exemple de paramétrisation d'un panel dans Grafana

8.2 Paramétrage du graphique

Pour l'affichage des données, Grafana permet de choisir entre 12 types de graphiques. Il est également possible de paramétrer la mise en avant de certains types de valeurs : asymptotes horizontales pour une limite de la mesure à ne pas dépasser, ou bien des valeurs spécifiques telles que la moyenne et les extrêmes. Ces valeurs doivent être choisies en fonction des besoins.

Ces choix de paramétrisation sont importants pour faciliter l'accès aux données, ils peuvent éviter de devoir analyser tout le graphique. En effet, si les valeurs importantes sont mises en avant, il est possible de faire un compte-rendu rapide et efficace de la situation du patient et ainsi d'agir rapidement.

8.2.1 Graphique

Le choix du groupe s'est porté sur une combinaison entre un graphique en 2D et une jauge pour chaque mesure à effectuer (température et pouls). Le premier choix se justifie par le fait que ces 2 mesures évoluent en fonction du temps, il est donc pertinent de montrer un graphique qui indiquera l'évolution de la mesure en fonction du temps. Il permet aussi de retrouver l'origine temporelle d'une potentielle dégradation de l'état de santé du patient et d'avoir une vision globale de sa santé pendant la période de port du kit.

En ce qui concerne le second choix, la jauge alliée à l'actualisation de la page Web permet de connaître la dernière mesure effectuée par l'ESP32. C'est un outil qui fournit des données concernant la situation instantanée et permet donc aussi de prendre des décisions. Par exemple, s'il y a une irrégularité sur le graphique et que 10 à 15 min plus tard le patient montre des mesures tout à fait normales, il serait assez logique de penser que cette irrégularité n'est pas forcément liée à l'apparition d'un symptôme.

Inversement, une situation stable sur le graphique mais une valeur instantanée élevée ou basse peut être le signe d'une aggravation brusque et potentiellement dangereuse de l'état de santé du patient.

8.2.2 Valeurs pertinentes du graphique

En ce qui concerne les valeurs importantes du graphique, il a été décidé de faire apparaître deux asymptotes horizontales sur le graphique. Il s'agit d'asymptotes telles que $y = \max(\text{valeur admissible du symptôme})$ et $y = \min(\text{valeur admissible du symptôme})$.

Par exemple, la valeur maximale admissible de la température du patient est de 37,5°C [www20]. Au-delà, le patient commence à avoir de la fièvre. Ainsi, si le patient admet des valeurs hors de cet intervalle délimité par les asymptotes celles-ci seront indiquées en rouge et donc d'autant plus visibles sur le graphique (8.2).

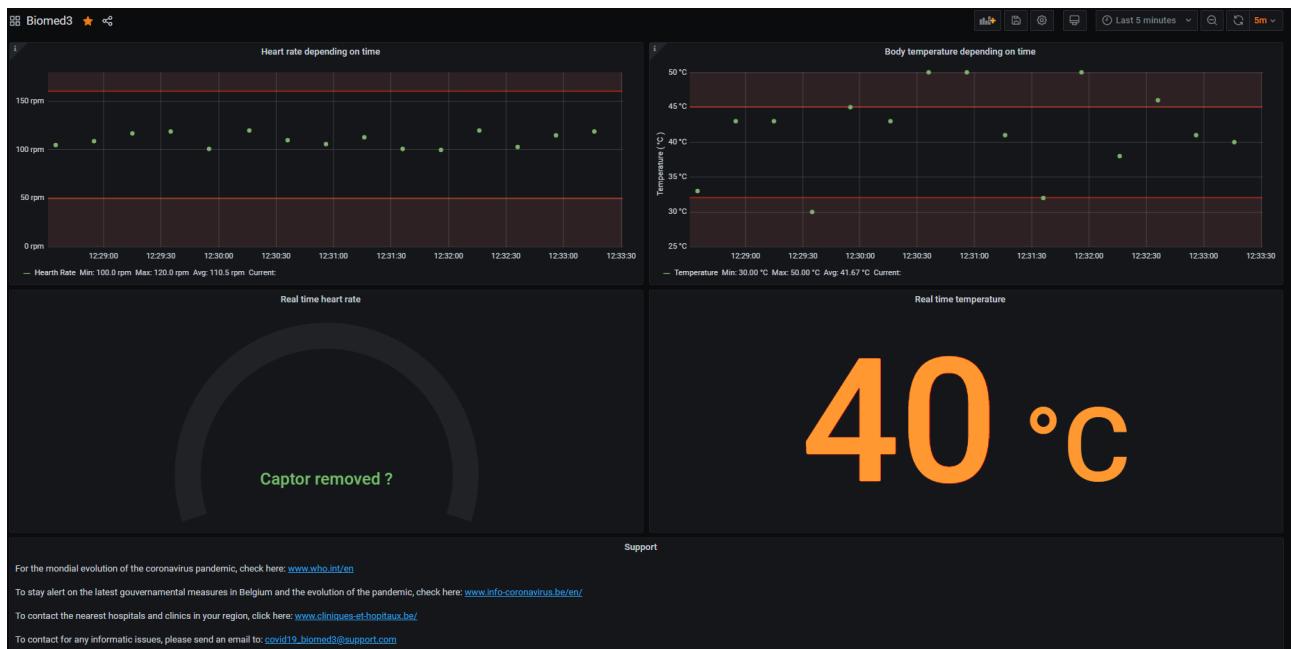


FIGURE 8.2: Grafana : exemple d'interface

Traitement de données

Il est nécessaire de pouvoir traiter les données obtenues à l'aide des capteurs. En effet, sans cela, les capteurs ne seraient d'aucune utilité. Le traitement de données se fera à l'aide d'un code Python, reliant la base de données InfluxDB (contenant les données obtenues des capteurs) et un algorithme permettant de donner sens à ces données. Ce code permet aussi d'envoyer des e-mails dans le cas où les mesures sont alarmantes, ou qu'il y a un autre problème.

9.1 Récupération des données

La récupération des données est primordial pour le code de traitement de données. Pour cela, l'utilisation de la librairie InfluxDB sur python est nécessaire. La récupération des données se fait en 3 étapes. La première, consiste en la création d'une liste vide qui contiendra les données. Ensuite la deuxième étape se fait à l'aide d'une "Query" de type "SELECT" qui va chercher dans la base de données une données ayant un tag spécifique (correspondant à la donnée voulut). Finalement, on prend la dernière valeur se trouvant dans la base de données et on l'ajoute dans la liste vide grâce à la commande ".append". On réitère l'étape 2 et 3 chaque 5 minutes (temps nécessaire pour l'apparition d'une nouvelle valeur sur la base de données¹). Ce processus se fait pour chaque type de données que l'on veut traiter (fréquence cardiaque, température ou taux de saturation en oxygène). C'est ainsi que sont créées et actualisées les listes de données à traiter et à interpréter.

9.2 Explications du code

Il faut d'abord importer la librairie influxdb afin d'utiliser la classe influxDBClient, qui servira par la suite à importer les données de la base de données.[tim; Rud20]

1. Dans le cas du prototype périodique

```
#Import InfluxDBClient library
from influxdb import InfluxDBClient
```

FIGURE 9.1: Import influxdb

Ensute, les différents paramètres nécessaires à la connexion à la base de données sont définis ci-dessous. À partir de là, un client est créé depuis la classe influxdb sur base des identifiants.[tim; Rud20]

```
#influxdb settings
host='influx.biomed.ulb.ovh'
db='biomed3' #replace by your database

#influxdb credentials
username='biomed3' #replace by your database user login
password='Or1nqhDW5ynBvs4a' #replace by your database user password

#init the influxdb client
client = InfluxDBClient(host=host, port=80, username=username, password=password, database=db)
```

FIGURE 9.2: login database

On définit une variable query, qui est une instruction en influxQL qui permet de soumettre une requête à la base de données. Ici, 'data' représente la field value et 'testgrafana' le measurement. Cette commande permet de récupérer les données de température depuis le measurement. Le « client.query(query) » permet d'envoyer la requête query à la base de données. Le résultat envoyé sera stocké dans une variable "results" qui, dans ce cas-ci, représente un objet de type "result-set".[tim; Rud20]

```
query='SELECT "data" FROM testgrafana' #testfor ou testgrafana
results=client.query(query)
```

FIGURE 9.3: query

Afin de pouvoir analyser les données pertinentes facilement, il est d'abord nécessaire de les récupérer et de les rassembler dans une liste. Pour cela, il faut convertir les résultats obtenus de type "resultset" en une liste; la méthode « getpoints » permet de transformer la variable "results" en un ensemble de données qui sera mis en liste de dictionnaires. Les données peuvent ensuite être rassemblées dans une liste en utilisant une boucle for comme ci-dessous. Pour chaque élément de la liste qui sont des dictionnaires, les valeurs dont les clés sont "data" sont récupérées dans la liste "temp". Ces valeurs correspondent donc à la température mesurée.[tim; Rud20]

```
points = list(results.get_points(measurement='testgrafana'))  
temp = []  
for i in range(len(points)):  
    temp.append(points[i]["data"])
```

FIGURE 9.4: liste finale

10

Interprétation des données

Après les avoir converties en listes, le programme peut interpréter les données. Pour le moment, il peut utiliser et interpréter les deux types de données récoltées par les deux capteurs installés sur le kit périodique : des données de température et des données de pouls. Pour faciliter l'utilisation et des possibles changements dans l'utilisation de l'ESP32, la fréquence d'utilisation des capteurs par le kit de dépistage est définie en tant que variable globale du programme.

10.1 Interprétation des données

Tout d'abord, il faut se renseigner sur la partie médicale des données à traiter. On distinguera 2 cas : le patient en bonne santé (pré-covid) et le patient 'fragile'. Le patient fragile englobe les femmes enceintes, les bébés, les personnes âgées, les personnes à antécédents de maladie cardio-vasculaire ou (et) pulmonaire et les personnes atteintes d'une autre maladie. Pour le cas de la température [Top19], une personne en bonne santé présente des symptômes de la COVID-19 à partir de 38°C tandis que pour une personne fragile, on parle de 37,2°C. De plus au delà de 39°C on parle de fièvre sévère dans les 2 cas. Pour la fréquence cardiaque[Pro], la plage de variation est beaucoup plus étendue. En effet, pour un patient en bonne santé, une fréquence cardiaque supérieure à 100 bpm est considérée comme inquiétante (c'est donc un symptôme) tandis que pour un patient fragile on parle de 90 bpm. Dans les 2 cas une fréquence cardiaque supérieur à 120 bpm ou inférieur à 40 bpm est considéré comme étant très alarmante. Finalement pour le taux d'oxygène [Le-] une valeur inférieure à 94% est un signe symptomatique et si on atteint un taux inférieur à 90% la donnée est alors considérer comme alarmante. Pour le taux d'oxygène il n'y a pas vraiment de différence entre le patient sain et fragile.

10.2 Explication du code

Tout d'abord, il faut créer la fonction qui enverra un mail automatique au patient après avoir analyser ces données. La variable de la fonction est le message approprié à la situation, et l'objet du mail est "information suivi COVID-19". Il sera envoyé depuis la boîte mail "biomed3ulb@gmail.com" et le destinataire est le patient. On peut éventuellement ajouter un deuxième destinataire (comme par exemple les urgences). Le code (voir Figure 11.1.1) ne supporte pas plus de deux boîtes mail destinataires, mais cela nous suffit dans le cadre de ce projet.

```

52     def envoie_mail(message):
53         from_addr = 'biomed3ulb@gmail.com'
54         to_addr = 'driss.elqaisy@gmail.com'
55         msg = MIMEText()
56         msg['From'] = from_addr
57         msg['To'] = ', '.join(to_addr)
58         msg['Subject'] = 'Informations suivi COVID-19'
59         body = message
60         msg.attach(MIMEText(body, 'plain'))
61         email = "biomed3ulb@gmail.com"
62         password = "azeqsd12"
63         mail = smtplib.SMTP('smtp.gmail.com', 587)
64         mail.ehlo()
65         mail.starttls()
66         mail.login(email, password)
67         text = msg.as_string()
68         mail.sendmail(from_addr, to_addr, text)
69         mail.quit()

```

FIGURE 10.1: fonction envoie mail

La fonction "no wifi" va comparer le timestamp de la dernière valeur enregistrée au temps réel. Si cela fait plus de 10 minutes qu'il n'y a plus de valeurs enregistrées, un mail sera envoyé au patient pour lui demander de se reconnecter au wifi.

```

def no_wifi(time):
    points = time
    if int(time.strftime("%M")) - 10 > int((points[-len(points)-1])["time"])[14:16]):
        z = "Rentrez chez vous"
        envoie_mail(z)

```

FIGURE 10.2: fonction envoie mail

La prochaine fonction sert à supprimer toutes les valeurs aberrantes de la liste afin de ne pas influencer le résultat final. La fonction a comme variable la liste des valeurs mesurées et elle supprimera toutes les valeurs aberrantes. Pour la température, ces valeurs correspondent aux valeurs inférieures à 30°C ou supérieures à 45°C. La fonction renverra enfin une nouvelle liste dépouillée des valeurs insensées.

Afin de simplifier l'analyse des données, le groupe a créé une fonction qui renvoie la moyenne des valeurs de température qui se trouvent dans la liste "propre". Si la liste est vide, la fonction renvoie la valeur -1 : cela sera utile afin de distinguer cette valeur-là de toutes les autres valeurs.

```

def clean_list_temp(values_list):
    """La fonction prend en entrée une liste de données de température (a priori provenant de la DB). Elle va
    en enlever
        les valeurs aberrantes et donner une liste de valeurs utilisables en sortie."""
    liste_propre = []
    for value in values_list:
        if value >= 30 and value <= 45:
            liste_propre.append(value)
    return liste_propre

```

FIGURE 10.3: suppression de valeurs aberrantes

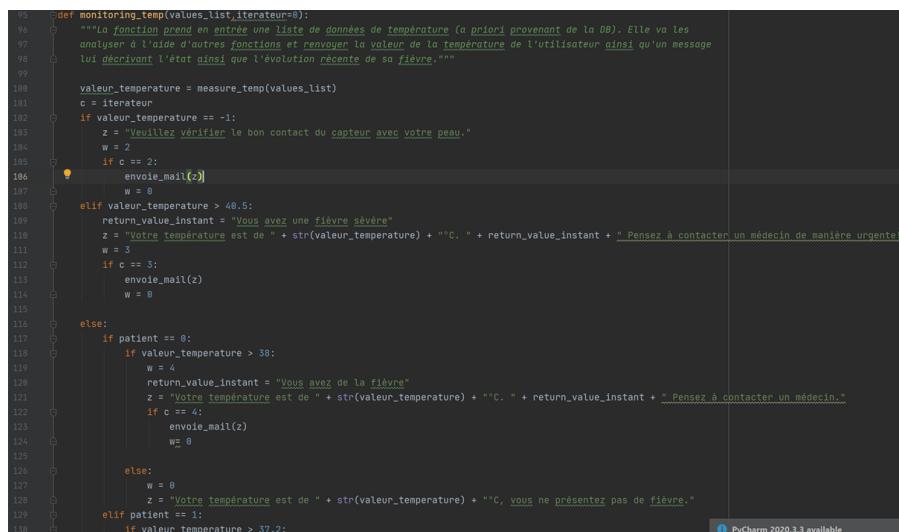
```

def measure_temp(values_list):
    """La fonction prend en entrée une liste de données de température (a priori provenant de la DB). Elle va
    faire une
        moyenne des données récoltées pendant la dernière minute et peut donner deux valeurs différentes en sortie
    : la
        valeur de la température mesurée ou -1 s'il n'y avait que des données aberrantes, c'est-à-dire si le
    capteur a un
        problème ou qu'il n'était pas correctement porté."""
    valeurs_minute = clean_list_temp(values_list[-nombre_mesures_minute_temp:])
    if len(valeurs_minute) == 0:
        x = -1
    else:
        x = mean(valeurs_minute)
    return x

```

FIGURE 10.4: moyenne de la température

Finalement, il y a la fonction monitoring qui renverra le mail en fonction de l'analyse des données. Le premier cas est le cas où la moyenne est égale à -1. Dans ce cas-ci un mail sera envoyé avec comme message "Veuillez vérifier le bon contact du capteur avec votre peau". Ensuite, il y a le cas où la température moyenne est de plus de 40,5°C (une fièvre sévère), le patient sera averti et on lui demandera de contacter un médecin de manière urgente. Le patient sera aussi notifié si la température est de plus de 38°C pour l'avertir de sa fièvre. Il y a également le cas des personnes fragiles, qui elles seront notifiées d'une fièvre dès que la température est de plus de 37,2°C. Voici le code de traitement de données (11.1.1) :



```

95     def monitoring_temp(values_list, iterateur=0):
96         """La fonction prend en entrée une liste de données de température (a priori provenant de la DB). Elle va les
97         analyser à l'aide d'autres fonctions et renvoyer la valeur de la température de l'utilisateur ainsi qu'un message
98         lui décrivant l'état ainsi que l'évolution récente de sa fièvre."""
99
100        valeur_temperature = measure_temp(values_list)
101        c = iterateur
102        if valeur_temperature == -1:
103            z = "Veuillez vérifier le bon contact du capteur avec votre peau."
104            w = 2
105            if c == 2:
106                envoie_mail(z)
107                w = 0
108        elif valeur_temperature > 40.5:
109            return_value_instant = "Vous avez une fièvre sévère"
110            z = "Votre température est de " + str(valeur_temperature) + "°C. " + return_value_instant + " Pensez à contacter un médecin de manière urgente."
111            w = 3
112            if c == 3:
113                envoie_mail(z)
114                w = 0
115
116        else:
117            if patient == 0:
118                if valeur_temperature > 38:
119                    w = 4
120                    return_value_instant = "Vous avez de la fièvre"
121                    z = "Votre température est de " + str(valeur_temperature) + "°C. " + return_value_instant + " Pensez à contacter un médecin."
122                    if c == 4:
123                        envoie_mail(z)
124                        w = 0
125
126                else:
127                    w = 0
128                    z = "Votre température est de " + str(valeur_temperature) + "°C, vous ne présentez pas de fièvre."
129            elif patient == 1:
130                if valeur_temperature > 37.2:

```

FIGURE 10.5: fonction monitoring

Le code traite également les données de pouls et d'oxygène mais le principe reste exactement le même; le code commence par supprimer toutes les valeurs aberrantes, ensuite il fait une moyenne de toutes les valeurs . Il analyse ensuite cette valeur afin de prévenir le patient si nécessaire en lui envoyant un mail. Le code averti aussi le patient si cela fais plus de 30 minutes que toutes les valeurs enregistrés sont aberrantes.

11

Simulateur

11.1 Cration d'un objet patient

Afin de pouvoir tester le code de traitement de données, la cration d'un code permettant la simulation d'un patient est nessaire. Ce code a t cr  en Python, car il y est simple de manipuler des listes et il s'agit du langage le plus connu des membres de l'équipe. Le code consiste en la cration d'un objet patient gen rant une temperature, une frequence cardiaque et aussi un taux d'oxygne de manire raliste.

11.1.1 Explication du code

Tout d'abord, une classe "Patient" a t cr e et prend comme paramtre un matricule, un âge et une "condition". La "condition" n'est autre que l'tat de sant du patient : le patient peut tre une femme enceinte, une personne atteinte de problmes cardiaques ou une personne atteinte de problmes respiratoires. La premire mthode associe  cette classe consiste en la catgorisation du "Patient". Si ce dernier a dj une condition telle que le patient est une femme enceinte ou a des problmes cardiaques ou respiratoires, alors il sera catgoris comme tant un patient fragile. De mme, si l'ge du patient est suprieur  60 ans il sera aussi catgoris comme patient fragile. Si, au contraire, le patient n'a aucune condition spcifique et est âg de moins de 60 ans, il sera catgoris comme patient sain. Cette mthode permet donc de connaître si le "Patient" gen r  est sain ou fragile, information utilis  dans le code de traitement de donnes. La deuxime mthode consiste en la gen ration d'une liste de temperature. Elle prend en entr e une variable (un entier) appell  "hasardeur" dont la valeur par dfaut vaut 2 et renvoie une valeur v. La gen ration des valeurs de la liste se fait sur base de probabilit s afin que la temperature d'un patient soit simul e de la manire la plus raliste possible. Tout d'abord on dfinit une variable "hasard" qui prend une valeur enti re entre 1 et 100 et une variable x qui vaut la valeur de hasardeur. Les variables "hasard" et x seront

```
# Définition d'une classe Patient décrit à l'aide d'un matricule, d'un âge et d'une condition physique. La classe
# Patient contient 3 fonctions.
# Le Patient peut renvoyer son état en fonction des données du Patient et peut générer une liste de sa température.
class Patient:
    def __init__(self, matricule, age, condition):
        self.matricule = matricule # Matricule du patient
        self.age = age # Age du patient
        self.condition = condition # Condition physique du patient

    # Fonction vérifiant si le Patient est malade, âgé ou dans un état fragile face au covid.
    def patient_malade(self):
        if self.age > 60 or self.condition == ('Femme enceinte' or 'Problème respiratoire' or 'Problème cardiaque'):
            x = 1 # Renvoie 1 si Patient fragile
        else:
            x = None # Renvoie None si Patient en bonne santé
        return x

    # Fonction vérifiant si le Patient est en bonne santé.
    def patient_sain(self):
        if self.patient_malade() == 1: # Si le patient a déjà été catégorisé comme malade, renverra None. Cela évite la possibilité d'être dans les 2 cas.
            x = None
        else:
            x = 2 # Sinon renverra 2
        return x
```

FIGURE 11.1: Définition de la classe "Patient" et méthode de l'état de santé du patient

utiliser comme condition pour les différents cas possibles. Le premier cas possible (cas avec une température de corps normal) est lorsque "hasard" vaut entre 6 et 100 et si x et différent de 1 et 0. Dans ce cas là, la méthode générera une valeur de température entre 36 et 37°C (de manière aléatoire) et la rajoutera dans une liste de vide. De plus elle attribuera à la variable v la valeur 2. On a donc une probabilité de 0.96 que le patient aie une température normale. Le deuxième cas possible ,qui représente un cas de fièvre légère, a lieu lorsque hasard est compris entre 3 et 6 ou lorsque x vaut 0. Cette fois-ci, ce sera une valeur entre 38 et 40°C qui sera ajoutée à la liste et v vaudra 0. De manière analogue, le cas représentant une fièvre sévère fera ajouter une valeur comprise entre 41 et 43°C et v vaudra 1. Finalement le cas simulant une donnée aberrante ajoutera une valeur comprise entre 45 et 78°C et v vaudra 2.

```
def température(self, hasardeur = 2):
    hasard = randint(1, 100)
    x = hasardeur
    if 6 <= hasard <= 100 and x != 0 and x != 1:
        self.liste.append(randint(36, 37))
        v = 0
    elif 3 <= hasard < 6 or x == 0:
        self.liste.append(randint(38, 40))
        v = 1

    elif 1 < hasard < 3 or x == 1:
        self.liste.append(randint(41, 43))
        v = 2
    else:
        self.liste.append(randint(45, 78))
        #sleep(1)
        v = 3
    return v
```

FIGURE 11.2: Définition de la méthode "Température" qui permet de simuler la température du patient

Il est clair qu'un patient atteint d'une fièvre ne verra pas sa température baisser dans les 5 minutes. C'est alors que la variable `v` aura un rôle important. En effet, lors de l'appel de la méthode "Température" dans la méthode "Simulation" (défini au paragraphe précédent), on introduit comme variable d'entrée "hasardeur", la valeur `v` qui a été renvoyé lors de l'exécution précédente (pour la première exécution on a mis une valeur par défaut qui vaut 2). Ceci permet d'assurer que pendant les 30 prochaines itérations, une valeur correspondant au bon `v` sera ajouté à la liste. Une fois la 30 ème itération atteinte (pour un cas donné), on entre comme variable d'entrée dans "Température" la valeur 2.

```
while i == 1:
    # Boucle infinie
    x = self.temperaturer(alpha) # hasardeur vaudra la valeur d'alpha
    if x == 1 :
        alpha = 0 #alpha dépend de la valeur renvoyé par température
        n = n+1
    if x== 2 :
        alpha = 1
        n1 = n+1
    if n == 30 : #à la 30ème itération , on repasse avec hasardeur qui vaut 2 (pour les 2 cas)
        n = 0
        alpha = 2
    if n1 == 30:
        n1 = 0
        alpha = 2
```

FIGURE 11.3: Appel de la méthode "Température" avec comme variable d'entrée la variable de sortie précédente

De manière similaire, mais indépendante les unes des autres, la méthode "O2" et "pulse" ont été définies avec les même idées pour le réalisme de la simulation. On a donc besoin d'une méthode qui va appeler les 3 méthodes définies précédemment et lier le tout au serveur InfluxDB. Une boucle while qui ne sera jamais satisfaite sera utilisé pour faire tourner le code de manière continue. La méthode simulation appellera donc les 3 autres méthodes à chaque itération et enverra la dernière valeur de chacune des 3 listes générées par les méthodes précédente sur le serveur InfluxDB avec pour chaque type de valeur un tag associé. Ainsi, le code permet de simuler 3 paramètres physiologiques que sont la température, la fréquence cardiaque (pulse) et la saturation d'oxygène dans le sang. Cette simulation le fait de manière réaliste car elle permet une transition d'un cas a un autre de manière fluide et cohérente tout en ayant une certaine diversité des cas possibles.

```
x2 = self.liste[-1]
y2 = self.liste1[-1]
z2 = self.liste2[-1]

line1 = 'temperature_td,id=temperature_td temp={}'.format(x2)
client.write_points(line1, protocol='line') # write the line

line2 = 'pulse_td,id=pulse_td pulse={}'.format(y2)
client.write_points(line2, protocol='line') # write the line

line3 = 'o2_td,id=o2_td o2={}'.format(z2)
client.write_points(line3, protocol='line') # write the line
sleep(300)
```

FIGURE 11.4: Appel de la méthode "Température" avec comme variable d'entrée la variable de sortie précédente

12

Site Web

Pour permettre au patient d'avoir accès à ses relevés de santé, et en complément de Grafana, il fut décidé de développer une page web. Ceci offrait l'avantage de regrouper facilement toutes les données, sans inonder le patient de mails permanents : lui, ainsi que les équipes médicales, peuvent y avoir accès quand bon leur semble. Le site web est constitué de 4 pages : une page d'accueil, une page pour le questionnaire 3.1, une pour les relevés, et une pour les notifications.

12.1 Langages

Pour développer ce site web, deux langages furent utilisés : python et HTML.

Le HTML est un langage conçu pour la structuration et la mise en forme des pages web. Ainsi, la saisie de texte, l'ajout de lien, d'images, de textes gras ou en italique, la couleur de fond... sont gérés par ce langage. Il s'agit donc de la base de l'élaboration du site.

Sans s'étendre trop longtemps sur le sujet, il est intéressant de souligner le fonctionnement du HTML[Ref21]. Celui-ci fonctionne par ouverture et fermeture d'instructions. Ainsi, si l'on souhaite mettre un texte en titre, on écrira l'instruction suivante :

```
<title> (le titre en question) </title>
```

Seulement, dans le cadre de ce projet, le HTML s'est avéré insuffisant, car il ne permet que de structurer une page. Or ici, un code était nécessaire pour permettre la connexion à la base de données. La récupération des données pour les afficher, et le stockage des réponses doivent donc passer par un autre code, pas écrit en HTML. Le choix de python paraissait évident, puisqu'il s'agit du langage le mieux maîtrisé par les membres du groupe, et que par ailleurs, un code de récupération et de stockage de données fut déjà réalisé au premier quadrimestre.

Il convient de noter qu'il est possible de coder une page web simple entièrement en python : les instructions ont la même forme qu'en HTML, si ce n'est qu'il faut imprimer toutes les instructions

(usage de l'instruction 'print' en python). Cela a le désavantage d'être assez lourd et peu lisible, et c'est pourquoi, quand cela est possible, du HTML fut utilisé.

12.2 XAMPP

Un site web est un ensemble de fichiers HTML . Pour que ce site soit accessible à tous via un simple lien, il aurait fallu louer un serveur (c'est à dire héberger le site). Ayant une limite au niveau du budget et un temps limité ,il a été jugé comme non-prioritaire d'héberger le site.

C'est pourquoi, celui-ci a été fait à partir de XAMPP, une application permettant de mettre en place un serveur Web local gratuitement. Il offre plusieurs serveurs et dans notre cas, Apache a été celui utilisé. [ION19]

Pour activer le site, il fallait commencer par ouvrir le panel de contrôle de XAMPP et activer Apache (voir photo 12.1).

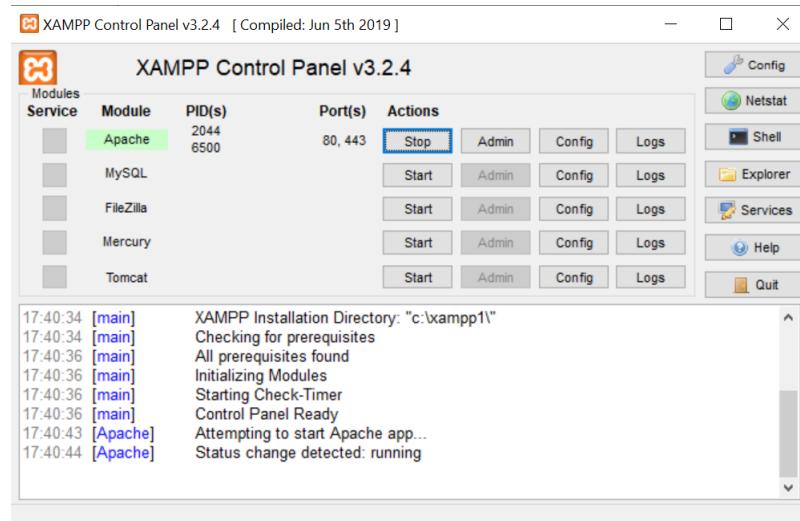


FIGURE 12.1: Panel de contrôle de XAMPP avec le serveur Apache activé

Une fois fait et après avoir mis le document (codé en python ou en html 12.1) dans un dossier dédié aux fichiers pouvant être exécuté, il ne restait plus qu'à ouvrir le site internet et taper dans la barre de recherche : <http://localhost/> + [NOM DU DOCUMENT]

12.3 Page d'accueil

La page d'accueil fut codée en HTML. Elle permet via des liens d'accéder aux deux autres pages.

Le code commence par l'initialisation du document, puis la mise en page et ensuite ,elle reprend les liens vers les différentes pages (12.2) :

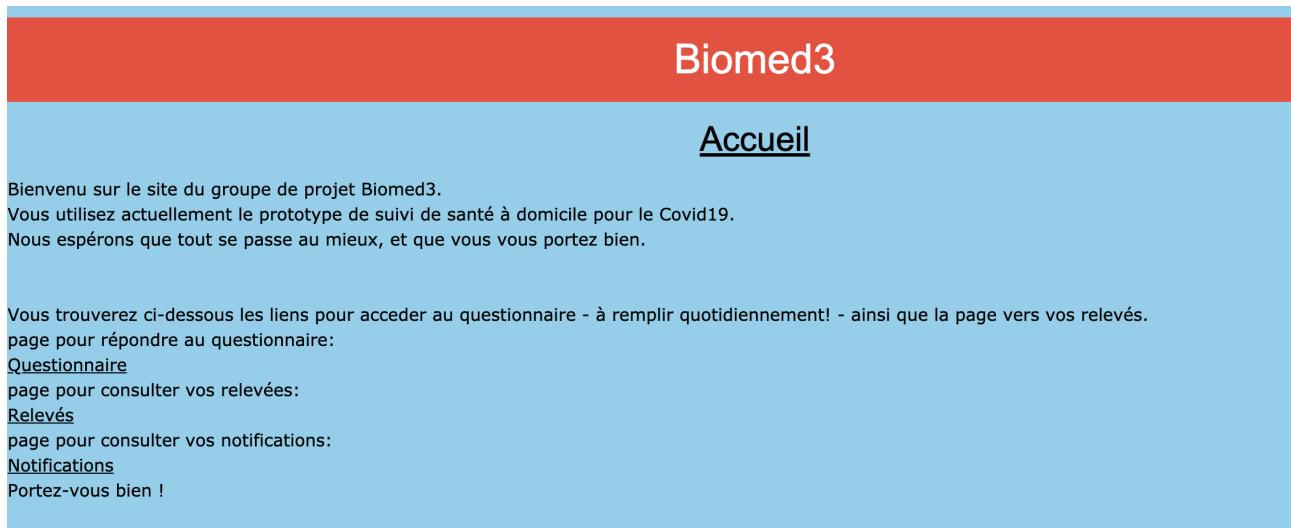


FIGURE 12.2: Page d'accueil

12.4 Questionnaire

Comme annoncé, un questionnaire a été réalisé pour pouvoir avoir une information sur les différents symptômes non-mesurables (3.1). Pour son fonctionnement, 3 fichiers ont été conçus.

Le premier fichier, codé en HTML (B.2), est la page principale où l'utilisateur répondra à différentes questions : excepté la première question pour le nom et prénom, la suite est à réponses multiples pour éviter toute erreur de frappe de la part de l'utilisateur (12.3). Les questions ont été prises dans un questionnaire donné aux patients d'un hôpital¹ avant une intervention chirurgicale.

1. L'hôpital CHIREC

The screenshot shows a web page titled "Biomed3" with a red header. Below the header, there's a section titled "Questionnaire". It includes instructions: "Veuillez répondre par : - '1' si la réponse est oui - '2' si la réponse est non". A text input field labeled "Insérer Nom/prénom?" contains "Nom/prenom". There are five dropdown menus for symptoms: "Avez-vous eu un des symptômes suivant : La toux?", "Une difficulté à respirer?", "Une douleur au niveau du thorax?", "Une perte de goût ou d'odorat?", and "Choisissez la valeur". To the right of the form is a large image of a COVID-19 virus particle with the text "COVID-19" and "Coronavirus Disease 2019". At the bottom left is a date and time input field showing "jj-mm-aaaa" and "--:--" with a "Submit" button. Below the input field is the text "Time: 00:59:31".

FIGURE 12.3: Page du questionnaire

Après avoir soumis son questionnaire, celui-ci est renvoyé sur une autre page lui confirmant la bonne réception de ses réponses. Cette deuxième page a été codée en python pour, en plus d'afficher la bonne réception, créer un fichier "txt" où les réponses seront insérées (12.4).

```

1 #C:\Users\Melissa\AppData\Local\Microsoft\WindowsApps\python.exe
2 import os
3 print("Content-type: text/html\n\n") #keep this in all .py files to execute code.
4 import cgi
5
6 print('<body bgcolor="powderblue">')
7 print("<strong>Merci beaucoup d'avoir répondu au questionnaire!</strong>")
8 print("<p>Les données ont bien été enregistrés!<br>")
9 print("N'oubliez pas de vous lavez les mains régulièrement et restez chez vous.")
10 print('Que la maladie ne vous soit pas favorable...")
11 print("<p><em>Signé: L'équipe Biomed3</em><p!>")
12 print("&#128151;")
13 print("<br/>")
14 #Prend les données du questionnaire
15 form=cgi.FieldStorage()
16 question1 = form.getvalue("question1")
17 question2 = form.getvalue("question2")
18 question3 = form.getvalue("question3")
19 question4 = form.getvalue("question4")
20 question5 = form.getvalue("question5")
21 temps = form.getvalue("appt")
22 date = form.getvalue("trip-start")
23
24 #Ecrit les données dans un fichier
25 with open('réponse', 'w') as f:
26     f.write(str(question1))
27     f.write("\n")
28     f.write(str(question2))
29     f.write("\n")
30     f.write(str(question3))
31     f.write("\n")
32     f.write(str(question4))
33     f.write("\n")
34     f.write(str(question5))
35     f.write("\n")
36     f.write(str(date))
37     f.write("\n")
38     f.write(str(temps))
39     f.write("\n")
40 f.close()

```

FIGURE 12.4: Code de réception de réponse au questionnaire

Pour rentrer les données dans la base de données, il a fallu faire un autre code python fonctionnant de manière perpétuelle avec une boucle infinie. Pour éviter que le code ne lise le même fichier "txt", il a été nécessaire d'établir une condition. La seule condition qui nous permettait de comparer les réponses du fichier à un autre était le temps à lequel le questionnaire était soumis (12.5). Avec cette condition, il a été possible de faire une boucle "while" où :

- Le fichier lu rentre les données dans la base de données s'il a été remis à un temps différent que le fichier lu précédemment.
- les données, supposées déjà rentrées précédemment, ne seront pas remises à nouveau dans la base de données si le fichier lu a été remis en même temps (même date/heure).

```

18     jour = "0"
19     heure = "0"
20     j = 1
21     while j == 1: # Il lit le fichier préalablement fait avec les réponses du questionnaire et mets les réponses sous forme de liste
22         with open('réponse', 'r') as f:...
23             # Si l'heure et la date est différente que celui du fichier lu ayant : cela veut dire que c'est bien un fichier différent
24             if liste[5] != jour and liste[6] != heure:
25                 jour = liste[5]
26                 heure = liste[6]      #Le nouveau fichier lu devient la nouvelle référence
27                 client.switch_database('biomed3')
28
29             json_body = [
30                 {
31                     'measurement': 'questionnaireTEST',
32                     'tags': {"Personne": liste[0]},
33                     "fields": {"question1": liste[1], "question2": liste[2], "question3": liste[3], "question4": liste[4]}
34                 }
35             ]
36
37             client.write_points(json_body)
38             query = 'SELECT * FROM questionnaireTEST'
39             results = client.query(query)
40
41             print(results)
42
43             j = 0
44
45         else:
46             break
47
48         print("Le nouveau fichier lu devient la nouvelle référence")
49
50         j = 0
51
52     else:
53         break
54
55

```

FIGURE 12.5: Condition de comparaison pour la lecture des fichiers

12.5 Page de relevés

L'objectif de cette page est que le patient ait une vue claire, précise et chiffrée de l'évolution de ses différents symptômes au cours du temps. Les relevés de température et du taux d'oxygène du patient, ainsi que l'instant auquel la-dite mesure fut prise, sont donc repris sous forme de tableau.

Cette affichage nécessite la connexion à la base de données. De ce fait, la page fut codée en python. Le code se divise en deux parties : la récupération des données, et l'affichage à proprement parler (12.6). Ces deux parties sont écrites dans deux fichiers séparés, dans un souci de clarté mais aussi pour pallier à un problème, traité ci-dessous.

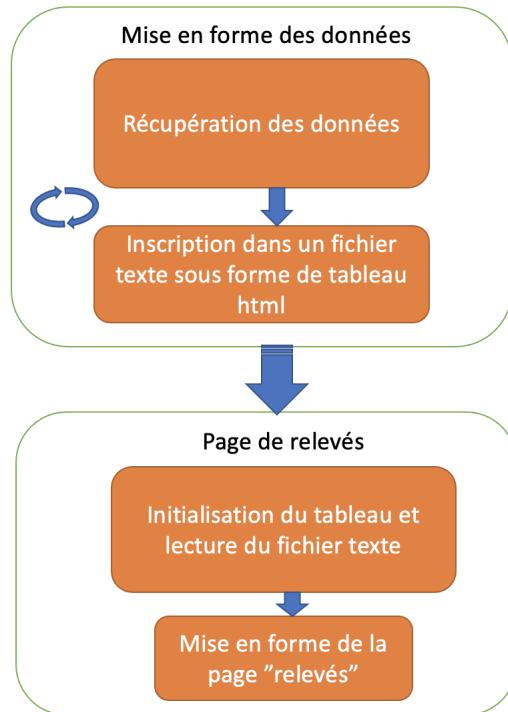


FIGURE 12.6: Schéma de fonctionnement de la page "relevés"

Le premier fichier(B.4, B.5 et B.6) commence par se connecter à la base de données. Il définit ensuite trois fonctions qui récupèrent respectivement des données de température(12.7), de rythme cardiaque et de saturation en oxygène, et les mémorise sous forme de liste. Ces fonctions renvoient deux listes : une avec les mesures à proprement parler, et une avec le temps.

```

def recuper_temp():

    #récupération
    query = 'SELECT temp FROM Experience4' # testfor ou testgrafana
    results = client.query(query)
    points = list(results.get_points(measurement='Experience4'))

    # création d'une liste avec les mesures de température
    temp = []
    for i in range(len(points)):
        temp.append(points[i]["temp"])

    # création d'une liste avec les temps des mesures de température
    time = []
    for i in range(len(points)):
        time.append(points[i]["time"])

    time = time_list_format(time)

    return temp, time

```

FIGURE 12.7: Fonction de récupération

Ensuite, trois autres fonctions sont définies, qui appellent les fonctions de récupération, et

qui écrivent dans un fichier texte ces données sous forme d'un tableau HTML à deux entrées (la mesure et le temps). (12.8).

```
def tableau_temp():
    temp_time = recuperer_temp()
    temp = temp_time[0]
    time = temp_time[1]
    with open('temperature', 'w') as f:
        for i in range(len(temp)):
            f.write('<tr>')
            f.write('\n')
            f.write('<td>')
            f.write(time[i])
            f.write('</td>')
            f.write('\n')
            f.write('<td>')
            f.write(str(temp[i]))
            f.write('</td>')
            f.write('\n')
            f.write('</tr>')
            f.write('\n')
```

FIGURE 12.8: Inscription des données dans un fichier

Le fichier se termine par une fonction main (12.9), qui initialise une boucle infinie et qui appelle les 3 fonctions écrivant les données dans un tableau. Cette boucle infinie sert à rafraîchir la page : à chaque fois qu'une nouvelle donnée entre dans InfluxDB, les fichiers textes seront mis à jour. Le code principal sert simplement à initialiser le code.

```
def main():
    i = 1
    while i == 1:
        tableau_temp()
        tableau_bpm()
        tableau_o2()
        sleep(10)

#Code principal
main()
```

FIGURE 12.9: Fonction "main" et code principal

Le deuxième fichier (B.7 et B.8) s'occupe de la mise en page. Il s'agit donc d'abord de faire les réglages de mise en page (l'allure du site web se retrouve ci-dessous : 12.11). Ensuite, les tableaux sont initialisés, puis les données formatées sont lues et inscrites dans le code. Il y a un tableau par mesure, divisé en deux colonnes : l'heure de mesure bien formatée et la mesure elle-même (12.10).

Cette découpe en deux parties, et l'utilisation de deux fichiers python séparés, s'est avéré nécessaire pour pallier un problème. En effet, l'import du module influxdb, indispensable pour la

```


| Date et heure | temperature |
|---------------|-------------|
|---------------|-------------|


with open('temperature', 'r') as f:
    a = f.read()
    print(a)

print('</table>')

```

FIGURE 12.10: Affichage des données

connexion à la base de données, ne permettait pas à la page de s'afficher correctement : une page blanche s'affichait. Il a donc fallu contourner ce problème en inscrivant les données dans des documents séparés, que le code d'affichage doit lire et imprimer. Toutefois, ce problème a l'avantage de séparer les deux parties, ce qui rend le code plus clair.

Biomed3

Relevés de données médicales

Bienvenu sur votre page personnelle reprenant toutes vos informations concernant l'évolution de votre santé

Le dispositif périodique relève des données de température et de rythme cardiaque

Voici les résultats pour la température

Date et heure	temperature (°C)
2021-03-13, 16:52:27	20.62
2021-03-13, 16:52:29	20.61
2021-03-13, 16:52:30	20.58
2021-03-13, 16:52:31	20.56
2021-03-13, 16:52:32	20.56
2021-03-13, 16:52:33	20.56
2021-03-13, 16:52:34	20.54
2021-03-13, 16:52:35	20.52
2021-03-13, 16:52:36	20.51
2021-03-13, 16:52:38	20.5
2021-03-13, 16:52:39	20.5
2021-03-13, 16:52:40	20.47
2021-03-13, 16:52:41	20.47
2021-03-13, 16:52:42	20.47
2021-03-13, 16:52:43	20.45
2021-03-13, 16:52:44	20.44
2021-03-13, 16:52:45	20.44
2021-03-13, 16:52:47	20.41

FIGURE 12.11: Page web pour les relevés

12.6 Notifications

Il s'agit d'une page où les différentes notifications du traitement de données s'affichent régulièrement (12.14). Cela permet au patient d'avoir accès à l'ensemble des notifications (y compris les positives), sans encombrer sa boîte mail, qui ne sera utilisée que si la situation est jugée inquiétante.

À l'image de ce qui fut fait pour la page des relevés, l'affichage des notifications se fait en deux temps. Une première partie se trouve à même le code du traitement de données : dans la boucle, les résultats du monitoring seront inscrits dans un fichier texte, avec l'heure d'exécution du traitement de données grâce au module "datetime". Ceci se fait au moyen d'une fonction appelée dans le traitement de données. (12.12).

```
def notification(x,y,z,i):
    if i == 20:
        with open('monitoring', 'a') as f:
            f.truncate(0)
        with open('monitoring', 'a+') as f:
            f.write(str(datetime.now()))
            f.write('\n')
            f.write('<br>')
            f.write(str(x))
            f.write('\n')
            f.write('<br>')
            f.write(str(y))
            f.write('\n')
            f.write('<br>')
            f.write(str(z))
            f.write('\n')
            f.write('<br>')
```

FIGURE 12.12: Inscription des notifications dans un fichier texte

Ensuite, un code python (B.3) séparé s'occupera de l'affichage des résultats, de façon analogue à la page de relevés : le design de la page à proprement parlé (qui est le même que pour toutes les autres pages, par souci de cohérence), et l'affichage des notifications, qui passe par la lecture des données (12.13).

```
#notifications

with open('monitoring', 'r') as f:
    a = f.read()
    print(a)
```

FIGURE 12.13: Affichage des notifications

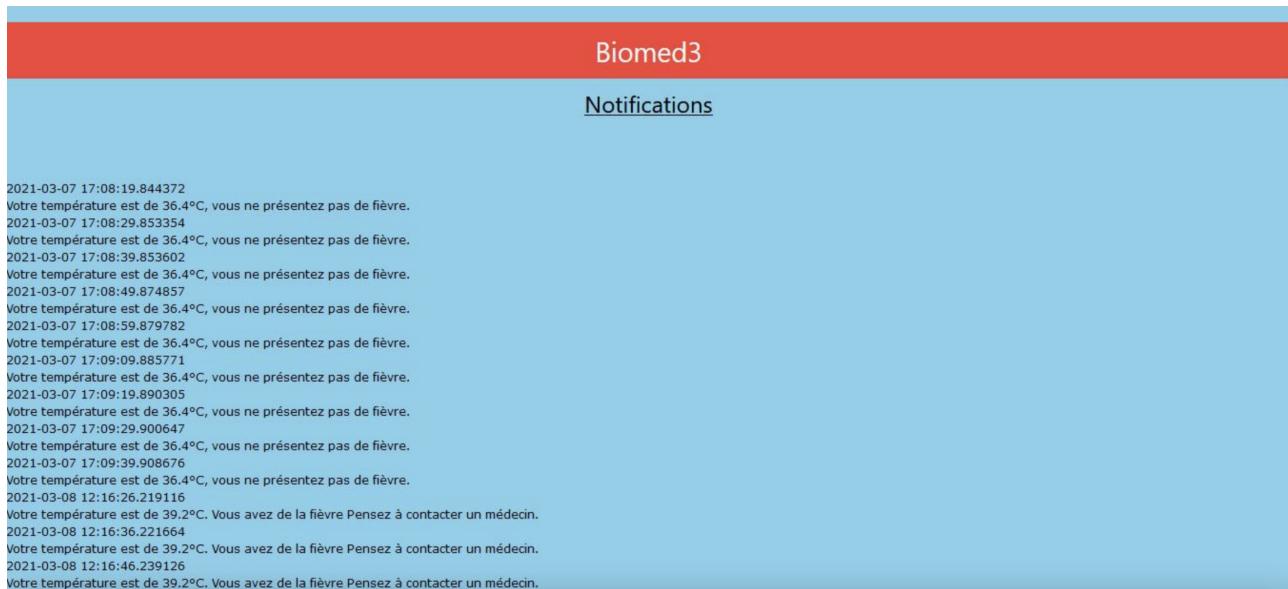


FIGURE 12.14: Page web pour les notifications

12.7 Amélioration

Le développement Web sort du cadre de ce projet. La page peut donc encore être grandement améliorée à différents niveaux.

Tout d'abord, les pages devraient être cryptées pour respecter le règlement de la RGPD, voir chapitre 6. En effet, cela sort du cadre de ce projet, et est trop complexe à réaliser.

Ensuite, il faudrait rendre cette page personnalisable : chaque patient doit avoir accès à ses propres tableaux, et ceci doit être protégé par un mot de passe.

13

Prototype périodique

13.1 Batterie

13.1.1 Découpage du code

Afin de maximiser l'autonomie de la batterie de l'ESP32, il est nécessaire de correctement identifier les différentes parties du code ainsi que leur fonction. En effet, il est inutile d'activer certaines fonctionnalités de L'ESP32 si elles ne sont pas utilisées.

De cette manière, il est possible de découper le code en 3 sections. Une première qui se contente de recueillir les informations envoyées par les capteurs, une deuxième qui transmet ces informations à une base de données au moyen d'un réseau WiFi et une dernière qui met l'ESP32 au repos. [San19]

En regardant la fiche technique de l'ESP32 et le rôle de chaque partie du code, il est possible de mettre l'appareil en 'sleep mode'. [ESP20a]

Durant la première étape, l'ESP32 recueille les données des capteurs et les stocke dans sa mémoire. Le microcontrôleur se met donc en 'modem-sleep mode', un mode où toutes ses fonctionnalités sont activées à l'exception du WiFi et du Bluetooth.

En ce qui concerne la deuxième étape, le WiFi doit être activé pour envoyer les données à la base de données. L'ESP32 se met donc en 'active mode', l'équivalent du modem-sleep mode avec le WiFi et le Bluetooth en plus.

Finalement, lorsque le microcontrôleur est au repos, il est mis en 'deep-sleep mode'. Bien que l'ESP32 ne fasse rien durant cette partie, il doit toujours avoir son microprocesseur allumé ainsi

que son RTC Timer pour se réveiller après un certain temps. [Las18]

13.1.2 Boucle

Une fois que les différents sleep modes de l'ESP32 sont définis, il est possible de définir une boucle qui sera répétée durant toute la période où le kit périodique sera porté par le patient.

Au premier quadrimestre, le code du prototype périodique fonctionnait de la manière suivante. Les différents capteurs étaient supposés prendre une donnée toutes les minutes pour les stockées dans une liste et puis l'appareil se mettait en deep sleep jusqu'à la prochaine minute (voir annexe C.1).

Une fois que cinq données se trouvaient dans la liste, le WiFi était activé et permettait à l'ESP32 d'envoyer toutes ces données. Cependant, l'utilisation d'un "timestamp" s'avéra nécessaire pour permettre d'organiser chronologiquement les différentes données dans la base de données. Cette piste fut explorée à multiples reprises mais il n'a pas été possible de récupérer ce timestamp [San18].

Pour résoudre ce problème, l'équipe s'est tournée vers une boucle plus simple à exécuter, voir 13.1. Les différents capteurs récupèrent des informations soit en une mesure, comme c'est le cas pour le capteur de température, soit en 10 secondes pour plus de précision, comme c'est le cas pour le capteur cardiaque. Une fois ces données recueillies, l'ESP32 se connecte au WiFi du domicile pour envoyer les données et enfin, le deep-sleep est enclenché pour finir cette boucle qui durera au total 5 minutes.

Ci-dessous se trouve un schéma qui décrit le fonctionnement de la boucle du prototype périodique.

13.1.3 Calculs

Dans le cahier des charges, le prototype périodique est supposé avoir une autonomie de 14 jours, soit la durée d'une quarantaine liée à la Covid-19. Il est donc nécessaire de déterminer quelle batterie sera utilisée avec le prototype et, pour cela, la consommation de ce dernier devra être déterminée analytiquement.

Puisque la consommation peut se mesurer en Ampère*heure, il faut déterminer, pour chaque partie du code utilisé, la période de temps durant laquelle elle est active ainsi que le courant dont elle a besoin. [Sch19]

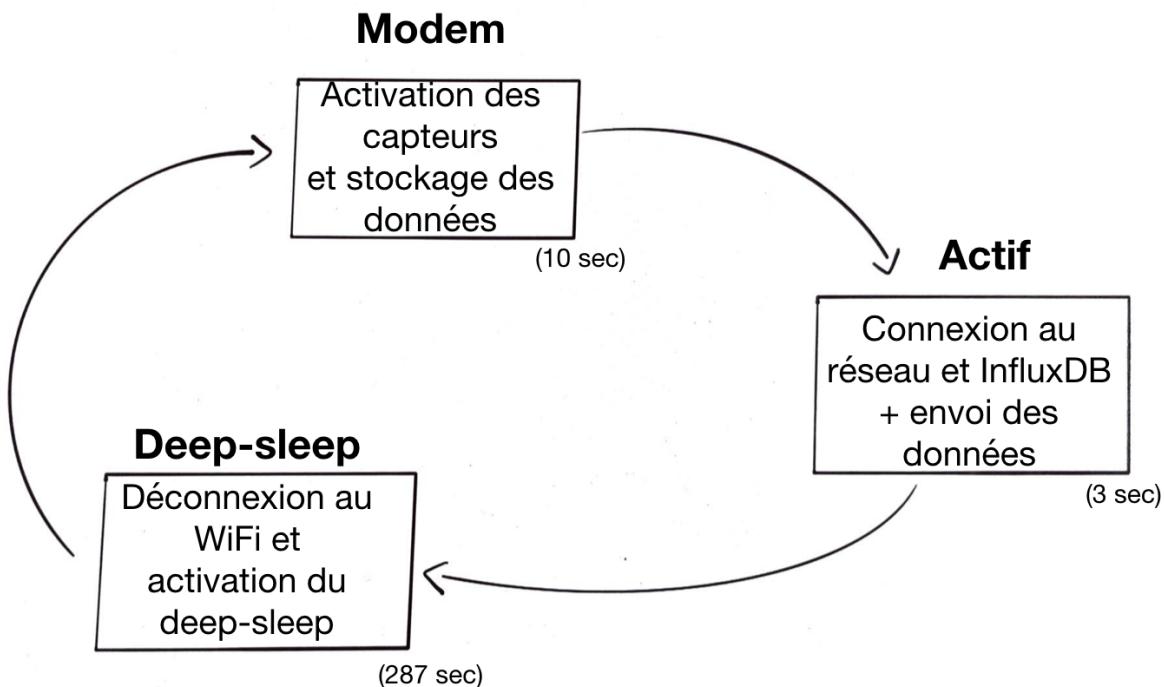


FIGURE 13.1: schéma de la boucle du prototype périodique

Commençons par définir la période de temps pendant laquelle la partie modem sera utilisée. Nous savons que la boucle de notre code dure 300 secondes. Par chaque heure qui passe, 12 itérations de cette boucle auront lieu. Ensuite, le prototype doit pouvoir rester actif 14 jours, soit 336 heures. Nous savons que dans chacune de ces boucles, la partie modem sera activée pendant 10 secondes, il s'agit du temps nécessaire pour l'acquisition des mesures des deux capteurs. Ainsi, il est possible de trouver la quantité de temps durant laquelle l'ESP32 sera en mode modem. Cette quantité est de $\frac{10*12*366}{3600}$ heures.

Pour ce qui est de la partie active, à l'exception du fait que ce mode ne sera actif que 3 secondes dans une boucle, le calcul est tout à fait similaire. Ces 3 secondes sont la durée dont a besoin l'ESP32 pour se connecter au WiFi du domicile, envoyer les valeurs à la base de données et se déconnecter. Ainsi, durant la totalité de l'expérience, le prototype sera en mode actif pendant $\frac{3*12*366}{3600}$ heures.

Enfin, le reste du temps où le prototype ne sera ni en mode modem ni en mode actif, il sera en deep-sleep. La période durant laquelle prototype sera en mode deep-sleep est donc de $\frac{287*12*366}{3600}$ heures. En effet, par boucle de 5 minutes, soit 300 secondes, il ne reste plus que 287 secondes pour le deep-sleep.

Le tableau qui suit, voir 13.1, regroupe le courant que requiert les différents composants du circuit en fonction du mode utilisé, [SEN19] [ESP20a] [Int18]. Dans les parties active et deep-sleep, les deux capteurs sont en veille, ce qui explique une valeur de courant nettement inférieur à celle dans la partie modem.

Ensuite, il est possible de retrouver les différentes valeurs de courant dont l'ESP32 aura besoin. On remarque que la partie active nécessite une très grande quantité de courant à cause de l'activation du WiFi. Ceci sera compensé en n'utilisant que très peu ce mode, ce qui permet de faire usage du deep-sleep - qui n'a besoin que de très peu de courant - la majorité du temps.

partie :	modem	active	deep-sleep
capteur de température	0.6	$0.2 * e^{-3}$	$0.2 * e^{-3}$
capteur cardiaque	6	$10 * e^{-3}$	$10 * e^{-3}$
ESP32	44	240	$10 * e^{-3}$
Total	50.6 mA	240.0102 mA	0.0202 mA

TABLE 13.1: Tableau de comparaison du courant nécessaires différents modes en mA

En prenant en compte toutes ces informations, il est enfin possible de calculer la consommation du prototype périodique pendant 14 jours. Ainsi :

$$Conso = conso_{modem} + conso_{active} + conso_{deep-sleep} \quad (13.1)$$

$$Conso = 51.8 * \frac{10 * 12 * 366}{3600} + 240.0102 * \frac{3 * 12 * 366}{3600} + 0.0202 * \frac{287 * 12 * 366}{3600} \quad (13.2)$$

$$\Leftrightarrow Conso = 1393mA.h \quad (13.3)$$

13.1.4 Choix

Suite aux calculs de consommation et à la facilité de recharge, le choix de la batterie s'est porté sur une batterie Lipo 3.7V d'une capacité de 1500 mAh, soit 100 mAh de plus que ce dont le code à besoin en théorie pour fonctionner. Ceci donne donc une marge de fonctionnement d'un jour supplémentaire pour le prototype

Malheureusement, durant la construction du prototype, les soudures d'un capteur de courant entre l'ESP32 et la batterie ont provoqué un court-circuitage de la batterie, la rendant inutilisable. Le mauvais câblage et celui qui aurait dû être fait seront détaillés dans la section 13.4.

Afin de tout de même permettre les différents tests avec le prototype périodique, la batterie fournie pour le prototype continu sera utilisée. Il s'agit également d'une batterie Lipo 3.7V de 1500 mAh.

De plus, deux petites batteries de 500 mAh ont été prêtées au groupe. Elle ont été utilisées pour faire des tests trois fois plus courts en durée et ainsi augmenter l'efficacité des différents essais.

13.2 Code

13.2.1 Capteurs

Dans la partie modem du code se trouvera la collecte des informations émises par les différents capteurs. Dans le code, chaque capteur possède une fonction qui effectue son setup et une valeur peut alors être recueillie. Les deux fonctions seront appelées au sein du troisième fonction qui lit et stocke les données fournies par les deux premières fonctions.

Température

Comme le montre la portion de code ci-dessous, voir 13.2, une première fonction crée le setup du capteur et par la suite, une instruction précédée d'un "return" permette de prendre la température. [Clo20].

```
//Se connecter au STS35 (0K)
void setupSTS35() {
    sts35.address(0x4B);
}

//Prendre la température (0K)
float initializeSTS35() {
    return sts35.readTemperature();
}
```

FIGURE 13.2: code sts35

Durant les premiers tests, le groupe s'est rendu compte qu'il existe un décalage entre la mesure de température prise par un thermomètre classique sous l'aisselle et celle prise par le capteur. Cette calibration est une constante qui peut être changée à n'importe quel moment. Ainsi, après plusieurs essais de mesures, voir 15.1, la meilleure calibration semble être de 4.5°C. Cependant, la fonction, 15.1 contient une condition if qui empêche cette calibration si le capteur n'est pas correctement porté.

```

void checkTemp(float t) {
    if (t > 25) {
        temp = t + calibration;
        Serial.println(temp);
    }
}

```

FIGURE 13.3: code calibration

Cardiaque

Le codage de l'utilisation du capteur cardiaque est assez similaire à celui du capteur de température. Une première fonction s'occupe de créer le setup du capteur et une valeur est ensuite retournée après utilisation.

Cependant, le capteur cardiaque est un objet très sensible. Ainsi, le capteur va d'abord stocker une grande quantité de mesures pendant 9 secondes dans une liste. Ensuite, la valeur moyenne de cette liste sera donnée et permet ainsi d'obtenir la valeur la plus réaliste possible.

Stockage des données

À ce stade, il suffit d'activer la collecte de ces valeurs. Pour cela, une nouvelle fonction, 13.4, appelle les fonctions précédemment définies. Cette dernière permet de stocker les valeur de température et de fréquence cardiaque dans deux variables "bpm" et "temp" mais aussi d'effectuer le calibrage lorsqu'il est possible.

```

void read_data_from_sensors() {
    bpm = initializeMAX30102();
    temp = initializeSTS35();
    checkTemp(temp);
}

```

FIGURE 13.4: code stockage

LED

Enfin, une petite fonction, 13.5, a été rajoutée pour prévenir le patient à travers le prototype si l'un des capteurs n'est pas correctement positionné. En effet, dans cette fonction, après stockage des différentes données, une LED rouge qui se trouve sur le boîtier clignotera 4 fois si la température recueillie est inférieure à 25°C ou si le bpm est inférieur à 30.

```

void lightUpLED() {
    pinMode(LED_BUILTIN, OUTPUT);
    if (temp < 25 || bpm < 30) {
        for (int i = 0; i<4; i++) {
            digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
            delay(500);                         // wait for a second
            digitalWrite(LED_BUILTIN, LOW);       // turn the LED off by making the voltage LOW
            delay(500);
        }
    }
}

```

FIGURE 13.5: code LED

13.2.2 WiFi

WiFiManager

Pour envoyer les données des capteurs jusqu'à une base de données, il faut que l'ESP32 se connecte à un réseau WiFi. Cependant, le prototype ne connaît pas à l'avance le nom et le code du réseau WiFi du domicile du patient qui le portera. C'est pourquoi le WiFiManager est utilisé durant ce projet [Tza15].

Pour connecter le microcontrôleur au réseau WiFi du domicile, ce dernier doit d'abord créer son propre réseau. En effet, le patient pourra, à l'aide d'un téléphone ou d'un ordinateur, se connecter au WiFi généré par l'ESP32.

Quand ceci est fait, une fenêtre s'ouvrira sur l'appareil pour permettre au patient de choisir à quel réseau l'ESP32 doit se connecter. Il suffit alors que la personne rentre le code de son réseau WiFi pour que le microcontrôleur s'y connecte et désactive son propre réseau. Cette technique permet de connecter l'appareil à un WiFi sans connaître le nom et le code du réseau à l'avance.

Pour permettre une connexion unique entre le microcontrôleur et le patient, il a été décidé d'utiliser comme code du WiFi généré par l'ESP32 la "MacAddress" de ce dernier, [San20]. Il s'agit d'un identifiant unique de l'ESP32 et sera fourni au patient dans un petit fascicule lors de la réception du prototype.

Ceci permet de remplacer le "patient ID" lors de la connexion au réseau tout en assurant une connexion unique car seul le patient aurait accès à cette MacAddress.

```

void lightUpLEDWiFi(boolean b) {
    pinMode(LED_BUILTIN, OUTPUT);
    if (b) {
        digitalWrite(LED_BUILTIN, HIGH);
    }
    else {
        digitalWrite(LED_BUILTIN, LOW);
    }
}

```

FIGURE 13.6: code LED wifi

De plus, la LED rouge sera à nouveau utilisée pour le WiFi. En effet, la LED s'allume au début du setup et ne s'éteindra que lorsque toute la fonction WiFiManager est effectuée, c'est-à-dire lorsque le microcontrôleur est connecté au réseau du domicile. Ceci permet de bien avertir le patient lorsqu'il doit se connecter au réseau WiFi, sans quoi aucune valeur ne sera envoyée à la base de données.

Boucle de connexion et reset

Lorsque le prototype n'arrive pas à se connecter au réseau WiFi auquel il l'est habituellement, une fonction sera utilisée et créera un "timer" de 10 secondes. Si au bout de ce laps de temps l'ESP32 n'est toujours pas connecté au WiFi, un reset sera effectué : l'ESP32 produira à nouveau son propre réseau en attendant que le patient s'y connecte.

Enfin, si le patient désire changer de réseau WiFi (par exemple : le patient se rend compte que le WiFi à l'étage est moins puissant quand il est connecté à la box WiFi du rez-de-chaussée), il est possible de provoquer une déconnexion du premier réseau.

En effet, il y a sur le côté de la boîte un bouton poussoir. Il suffit d'appuyer dessus pour éteindre le WiFi, provoquer un reset du prototype et la fonction WiFiManager sera réitérée.

13.2.3 InfluxDB

Une fois que les valeurs des capteurs sont collectées, il faut les envoyer à la base de données InfluxDB. Pour cela, il faut à nouveau inclure une bibliothèque avant le setup mais aussi introduire un certain nombre de données telles que : le nom de la base de données, son port d'accès, le nom de l'hôte et le nom de l'utilisateur ainsi que son mot de passe. [Sch20]

Une fois toutes ces données fournies, il est possible de se connecter à la base de données à l'aide d'une simple instruction, voir 13.7, qui sera stockée dans une fonction. Cette instruction est recopiée ci-dessous.

```
influx.setDbAuth(INFLUXDB_DATABASE, INFLUXDB_USER, INFLUXDB_PASSWORD);
```

FIGURE 13.7: code connect database

Une fois la connexion établie, une deuxième fonction, voir 13.8, s'occupe de créer l'objet qui sera envoyé à la base de données. Pour cela, un objet "measurement" est créé. Il est possible de voir que la portion de code a été prise au moment de la deuxième expérience.

Une série de "Serial.print()" suivent, ils ne sont utiles que pour nous au moment de l'élaboration du code, ils ne seront jamais visibles pour le patient.

Puisqu'il s'agit du prototype périodique, il a été décidé d'envoyer trois types de données différentes à la base de données : la fréquence cardiaque, la température et l'intensité du réseau WiFi.

Pour cela, trois ".addValue" doivent être rajoutées à l'objet "measurement", et le tout sera envoyé à la base de données à l'aide d'une dernière instruction dans le setup.

```
void store_measurement(float t, int b, long r) {
    Serial.println("Température: ");
    Serial.println(t);
    Serial.println("Battement par minutes: ");
    Serial.println(b);
    Serial.println("RSSI: ");
    Serial.println(r);
    measurement.addValue("temp", t);
    measurement.addValue("bpm", b);
    measurement.addValue("rss", r);
}
```

FIGURE 13.8: code influx

Enfin, le même système de boucle de reconnexion utilisé pour le réseau WiFi est utilisé en cas de perte de connexion entre le microcontrôleur et la base de données.

13.2.4 Deep-Sleep

Une fois que toutes les valeurs ont bien été envoyées à la base de données, l'ESP32 se met en mode deep-sleep pendant 287 secondes. Cette étape est cruciale car elle permet au prototype d'avoir une autonomie de 14 jours. Ainsi, une fonction, voir 13.9, est appelée pour mettre le microcontrôleur en deep-sleep. Ce dernier ne se réveillera que lorsque le timer de 287 secondes se sera écoulé [San19].

Cependant, avant cette étape, le WiFi allumé lors de la partie active doit être éteint, ce qui est fait par l'appel de la fonction "shutDownWifi". De plus, pour minimiser la consommation de la

batterie et par mesure de précaution, les ports ADC sont éteints.

```
void deep_sleep(int time_in_microseconds) {
    shutDownWifi();
    adc_power_off(); //Eteindre les ports ADC pour minimiser la consommation en deep-sleep
    delay(500);
    esp_sleep_enable_ext0_wakeup(GPIO_NUM_14, HIGH);
    esp_sleep_enable_timer_wakeup(time_in_microseconds);
    esp_deep_sleep_start();
}
```

FIGURE 13.9: code deepsleep

13.2.5 Setup

Pour commencer le setup, la fréquence du CPU, le microprocesseur central, est fixée à 80MHz pour optimiser la durée de vie du prototype. Ensuite, les ports SDA et SCL sont définis pour permettre à la communication en I2C de bien avoir lieu. Enfin, les ports ADC sont réactivés car ils avaient été éteints juste avant le deep-sleep précédent, voir 13.10

Dès lors, puisque toutes les fonctions sont bien définies, il suffit de toutes les appelées une à une dans le setup principal du code.

Ceci permet donc de former une boucle de 5 minutes qui se répétera durant les 14 jours d'utilisation.

Enfin, la fonction pour provoquer le reset manuel du prototype a été rajouté en début de boucle.

```
void setup() {
    Serial.begin(115200);
    Wire.begin(33, 32);
    pinMode(PIN_PUSH_BUTTON, INPUT);
    if (first_time || ESP_SLEEP_WAKEUP_EXT0) {
        setupWiFiManager();
    }
    setupSTS35();
    setupMAX30102();
    read_data_from_sensors();
    store_measurement(temp, bpm, rssi);
    lightUpLED();
    setupWifi();
    setupInfluxDB();
    if (getConnectedToWifi() && getAccesToDatabase()) {
        influx.write(measurement);
    }
    deep_sleep(287000000);
}
```

FIGURE 13.10: code setup

Il est important de rappeler que l'intégralité de ce code est disponible sur le gitlab fourni par l'équipe.

13.3 Boîtier

Le boîtier du prototype périodique a été réalisé en plastique noir à l'aide d'une imprimante 3D, dont la facilité de modification du modèle, le faible coût en matériaux et la rapidité d'exécution ont permis l'utilisation de plusieurs versions du boîtier afin d'arriver au meilleur design possible. Le boîtier mesure 9,45 cm de longueur, 8,10 cm de largeur en comptant les fixations de sangle et 2,60 cm d'épaisseur. Des ouvertures ont été prévues sur les côtés du corps central de manière à pouvoir passer le bouton poussoir, la LED et les capteurs et à les rendre visibles et accessibles au patient. De part et d'autre du corps principal, deux encoches de fixation ont été imaginées dans le but de passer une sangle pour pouvoir facilement porter le dispositif sur le bras ou ailleurs. Le boîtier se ferme à l'aide de deux vis qui le ferment solidement mais permettent un accès facile et rapide à l'intérieur de l'appareil.

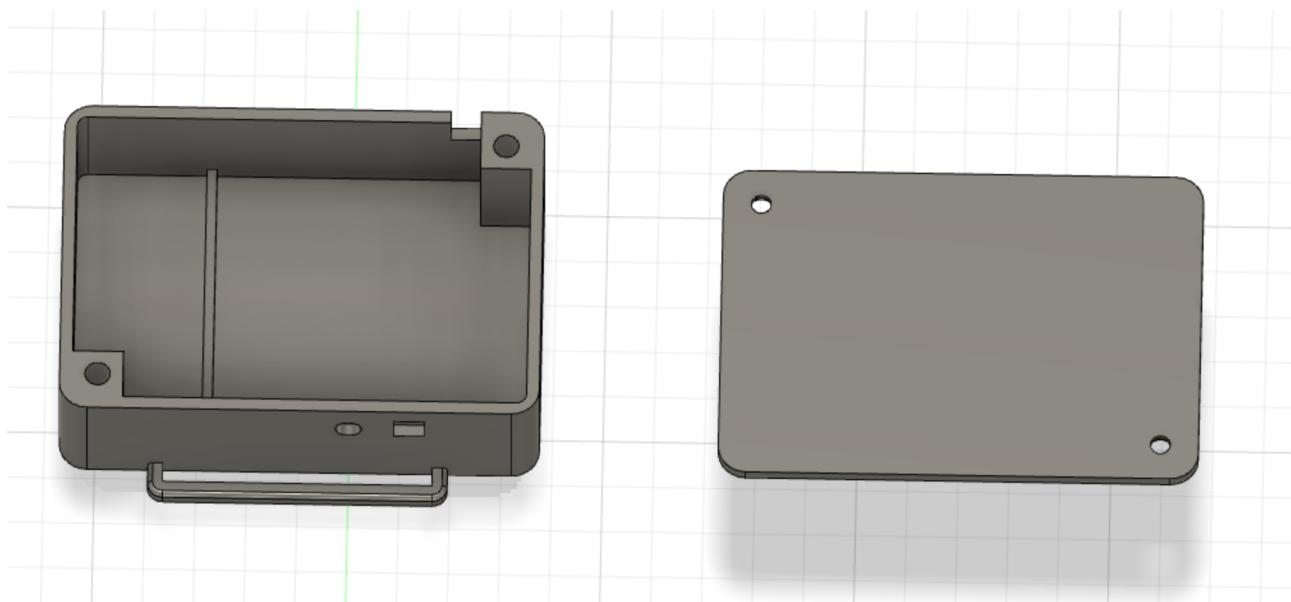


FIGURE 13.11: Plan 3D du prototype périodique.

13.4 Construction

Le schéma suivant indique l'ensemble des composants du prototype ainsi que leur agencement dans le circuit électrique. Il est en outre montré qu'une grande boucle de tension et une grande boucle de terre font le tour de tous les composants pour les lier entre eux et aux bornes corres-

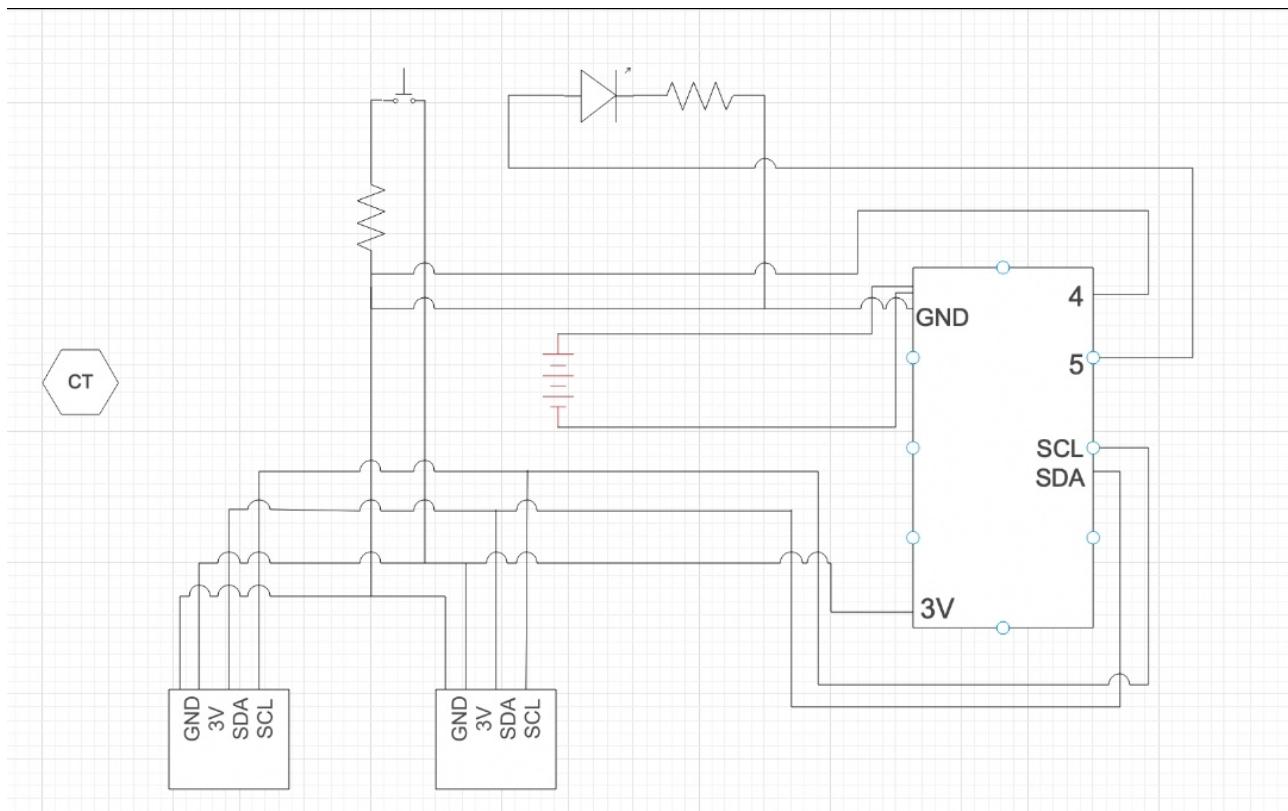


FIGURE 13.12: Schéma électrique du montage du prototype périodique réalisé à l'aide du logiciel EdrawMax

pondantes de l'ESP32. On peut également observer que la diode led est raccordée au port n°5 de l'ESP32, et que les capteurs de température et de pouls et le bouton poussoir sont raccordés au port n°4. Les capteurs sont aussi raccordés aux ports SCL et SDA de l'ESP32. Tout ce circuit est alimenté par une batterie, directement reliée à l'ESP32.

En pratique, le circuit a été assemblé à l'aide de différentes soudures entre les différents composants et des fils électriques et isolé avec de la gaine thermorétractable et du tape isolant.

Au premier assemblage, la batterie avait été malencontreusement soudée en court-circuit avec le capteur de courant qui devait être utilisé pour rendre compte de son état de charge. Après détection de la coquille, il a bien entendu été décidé de retirer le capteur de courant et de faire directement passer la batterie par l'ESP32. Heureusement, une soudure de trop au niveau de l'interrupteur de la batterie a permis d'ouvrir le circuit et a ainsi sauvé cette batterie-là. Après recherches sur internet à propos du capteur de courant, le INA3221, qui semble présenter des problèmes de manière récurrente dans les circuits dans lesquels il est utilisé, il a été décidé de ne plus en faire usage et de se contenter d'un multimètre pour tout compte-rendu de l'état de charge de la batterie, à la fois dans une optique de gain de temps et de minimisation des risques.

Prototype continu

Durant le second quadrimestre, un second prototype a été développé, dont les fonctionnalités diffèrent du premier. En effet, ce prototype est continu.

Ceci signifie que le prototype doit envoyer des données bien plus fréquemment afin de permettre un suivi plus poussé de la personne qui le porte. Ceci a du sens car ce prototype est réservé aux patients infectés par la Covid-19.

Le langage de programmation utilisé sera toujours le C++. Ainsi les motivations concernant le choix du langage expliquées au chapitre 5.2 sont toujours d'application.

14.1 Batterie

14.1.1 Découpage du code

Pour ce prototype, le processus concernant le découpage du code est l'opposé de celui utilisé pour le périodique. En effet, deux contraintes sont imposées par le cahier des charges :

1. L'utilisation d'une batterie 1500 mAh.
2. Le fonctionnement du prototype pendant au moins 8 heures avant de le recharger.
3. L'utilisation d'un oxymètre relié à l'ESP32 par bluetooth.
4. L'envoi presque continu des valeurs à la base de données

Il est donc impératif de les prendre en considération dans l'élaboration du code pour remplir les différentes attentes du cahier des charges.

Nous allons donc calculer la quantité de courant que les différentes parties du code utiliseraient. Pour ce prototype, il n'y pas de deep-sleep car les données doivent être constamment envoyées. Il n'y aura donc qu'une partie modem où un oxymètre connecté par bluetooth et un capteur de tem-

pérature récupéreront des données ainsi qu'une partie active où le WiFi sera activé et les valeurs des différents capteurs seront envoyées à la base de données.

Après plusieurs tests et recherches sur la quantité de courant utilisé par le bluetooth, il apparaît que le microcontrôleur ne consomme les 95 à 100 mA repris dans la datasheet que lorsqu'il se connecte pour la première fois au réseau bluetooth, c'est-à-dire dans le setup. Le reste du temps, le prototype se met automatique en mode BLE (Bluetooth Low Energy) et consomme alors beaucoup moins, [myw19].

Comme pour le périodique, le capteur de température n'est actif que pour la partie où il récupère des données, c'est-à-dire la partie modem, et se met ensuite en veille dans la partie active. Pour ce qui est de l'oxymètre, il requiert une certaine quantité de courant car le bluetooth est activé mais il n'est pas pris en compte dans la partie active car nous prenons une valeur de 240 mA pour l'ESP32, ce qui englobe le WiFi et le bluetooth.

Il est possible de retrouver au tableau 14.1 les valeurs de courant nécessaires pour les différents modes, [SEN19] [ESP20a] [myw19].

partie :	modem	active
capteur de température	0.6	$0.2 * 10^{-3}$
bluetooth	2.11	0
ESP32	44	240
Total	46.71	240.0002

TABLE 14.1: Tableau de comparaison du courant nécessaires différents modes en mA

14.1.2 Calculs et boucle

Suite aux contraintes fixées par le cahier des charges, il a été décidé de créer un code qui permettra au prototype de fonctionner le plus de temps possible, au minimum 8 heures, à l'aide d'une batterie ayant une capacité de 1500 mAh tout en permettant un suivi continu du patient.

Pour cela, un premier test fut effectué où le prototype était constamment en mode actif. Puisque ce mode requiert un courant de 240.0002 mA, le prototype a tenu un peu moins de 6h30. En effet (14.1),

$$\frac{1500mAh}{240.0002A} = 6.25heures \approx 6h30 \quad (14.1)$$

Pour résoudre ce problème, il a été décidé de créer une boucle qui serait un combinaison du mode bluetooth et du mode actif. Pour cela, il a fallu déterminer la durée minimale possible entre l'envoi de deux valeurs à la base de données, sachant qu'entre temps le WiFi doit être éteint et puis rallumé. Cet durée fut fixée expérimentalement, et vaut 4.2 secondes. Un temps inférieur ne permet pas un envoi correcte des données, ou provoque des pics de courant.

Ainsi, il a été décidé de former une boucle de 5.2 secondes, comme le montre le schéma 14.1, qui permettrait d'envoyer les données assez fréquemment sans causer de problèmes. Il est possible de retrouver ci-dessous un schéma de la boucle type.

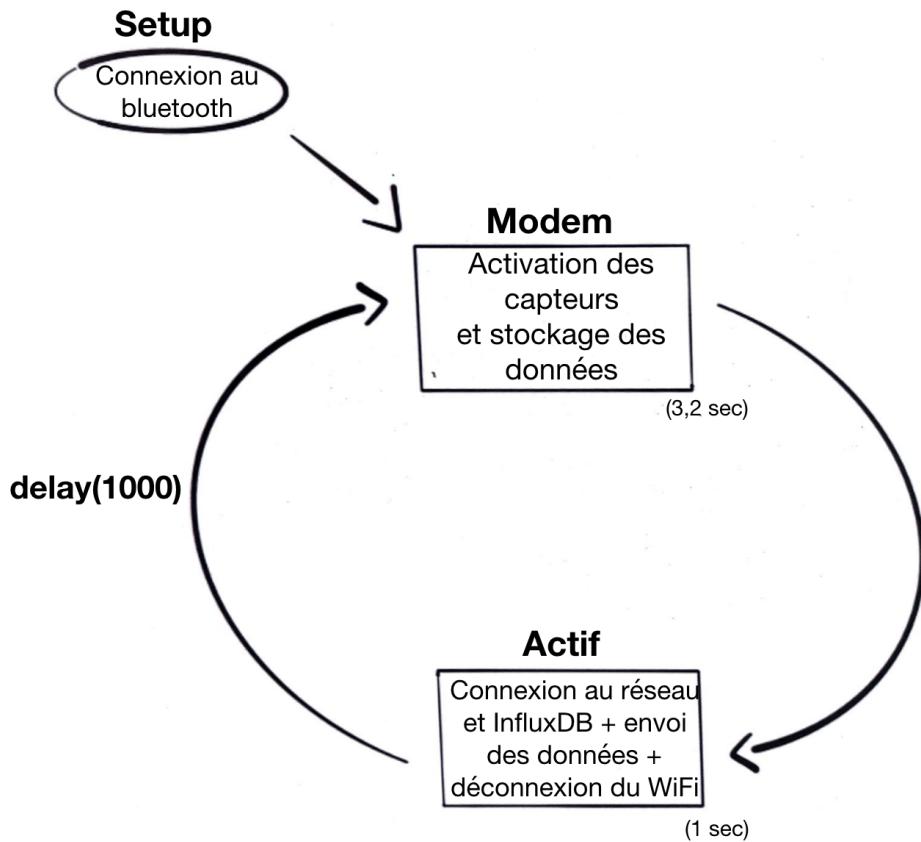


FIGURE 14.1: schéma de la boucle du prototype continu

Dans cette boucle, l'ESP32 est en modem pendant 4,2 secondes (prise des données + délai de 1 seconde) et en mode actif le reste du temps (soit une seconde). Il est donc possible de déterminer le courant que nécessite ce code. Ainsi (14.2),

$$\frac{4.2}{5.2} * 46.71 + \frac{1}{5.2} * 240.0002 = 83.88mA \quad (14.2)$$

Il a donc été décidé de tester ce code en le faisant tourner sur une batterie d'une capacité de 500 mAh. En regardant le calcul (14.3),

$$\frac{500mAh}{83.88mA} = 5.9h \quad (14.3)$$

En pratique, la batterie a tenu un peu plus de 5 heures Cependant, la batterie n'étant pas chargée complètement (environ 4.0V au lieu de 4.170V) et en tenant compte de certaines consommations annexes, comme le setup, ces cacluls sont tout à fait acceptables.

En transposant ces calculs à une batterie de 1500 mAh, soit de capacité trois fois plus élevée, le prototype continu devrait avoir une autonomie de plus de 15 heures avant de devoir être rechargé.

14.2 Code

La boucle du prototype continu étant établie, le code peut être écrit. Les étapes principales sont détaillées dans les sections qui suivent.

14.2.1 Capteurs

Température

La portion de code concernant le capteur de température est identique à ce que a été écrit pour le prototype périodique. Veuillez donc vous référer à 13.2.1 pour plus de détails. La seule différence par rapport au prototype périodique est l'utilisation de la LED par rapport aux capteurs.

En effet, dans le prototype continu, la fonction faisant clignoter la LED si l'un des capteurs se retrouvait mal placé a été enlevée par soucis de consommation. Puisque les boucles de ce prototype ne durent que 5 secondes, la LED risquerait de s'allumer trop souvent et provoquer une diminution de l'autonomie de la batterie.

Oxymètre

En ce qui concerne l'oxymètre, un code fourni par l'un des tuteurs du projet, [Erc20], a été employé. Ce code est composé d'une première fonction qui effectue le "setup" de l'oxymètre et une seconde qui permet de recueillir trois informations différentes que sont la fréquence cardiaque, la saturation d'oxygène dans le sang et l'indice de perfusion. Cette dernière information sera envoyée à la base de données mais ne sera pas exploitée dans la suite de ce projet.

14.2.2 WiFi

Tout comme le capteur de température, la portion de code concernant le WiFiManager et son reset est identique à celle du prototype périodique. Veuillez donc vous référer à 13.2.2 pour plus de détails.

Ce changement concernant l'utilisation de la LED est dû à la différence de conception du code. En effet, une boucle de 5 secondes - au lieu de 5 minutes pour le périodique - engendrerait une consommation trop importante de la LED et donc une diminution de l'autonomie du prototype.

14.2.3 InfluxDB

La partie relative à la connexion à la base de données et à la création d'un objet est similaire à celle faite pour le prototype périodique. La seule différence réside dans les données stockées dans cet objet. En effet, pour le prototype continu, voir 14.2, les valeurs envoyées à InfluxDB sont la température, la fréquence cardiaque, la saturation d'oxygène dans le sang et l'indice de perfusion.

```
void store_measurement(float t, int s, int b, float p) {
    Serial.println("temp");
    Serial.println(t);
    Serial.println("spo2");
    Serial.println(s);
    Serial.println("bpm");
    Serial.println(b);
    Serial.println("pi");
    Serial.println(p);
    measurement.addValue("temp", t);
    measurement.addValue("spo2", s);
    measurement.addValue("bpm", b);
    measurement.addValue("pi", p);
}
```

FIGURE 14.2: code influx continu

14.2.4 Setup

Comme son nom l'indique, le "setup" du code sera utilisé pour initialiser les différentes portions tout en définissant les ports SDA et SCL pour permettre à la communication en I2C de bien avoir lieu. De plus, la fonction WiFiManager est appelée pour connecter le microcontrôleur au réseau du domicile. Cette fonction est entourée de deux autres fonctions qui allument la LED tant que l'ESP32 n'est pas connecté au réseau du domicile. Contrairement au prototype périodique, c'est dans la "loop" que se trouve le cœur du code.

```

void setup() {
    Serial.begin(115200);
    Wire.begin(21, 22);
    pinMode(PIN_PUSH_BUTTON, INPUT);
    setupSTS35();
    setupOximeter();
    lightUpLED(true);
    setupWiFiManager();
    lightUpLED(false);
    shutDownWifi();
}

```

FIGURE 14.3: code setup continu

14.2.5 Loop

A l'aide de toutes les fonctions précédentes, il enfin possible d'écrire en code la boucle expliquée dans la section 14.1.2.

Pour cela, le capteur de température et l'oxymètre sont appelés pour recueillir les quatres valeurs de température, saturation en oxygène, fréquence cardiaque et indice de perfusion. Ensuite, ces valeurs sont stockées dans un objet "measurement".

Le WiFi est activé uniquement pour envoyer cet objet dans la base de données puis éteint directement après. Un "delay" d'une seconde est effectué pour s'assurer qu'il y a au moins 5 secondes entre l'envoie de deux objets et le cycle peut recommencer.

Enfin, la fonction pour provoquer le reset manuel du prototype a été rajouté en début de loop, voir 14.4.

```

void loop() {
    button_state = digitalRead(PIN_PUSH_BUTTON);
    if( button_state == HIGH ) {
        Serial.println("Suppression des réglages et redémarrage...");
        wifiManager.resetSettings();
        ESP.restart();
    }
    initializeOximeter();
    initializeSTS35();
    store_measurement(temp, spo2, heart_rate, pi);
    setupWifi();
    setupInfluxDB();
    influx.write(measurement);
    shutDownWifi();
    delay(1000);
}

```

FIGURE 14.4: code loop continu

Il est important de rappeler que l'intégralité de ce code est disponible sur le gitlab fourni par l'équipe.

14.3 Boîtier

Le boîtier du dispositif continu est très semblable au boîtier du dispositif périodique et ne diffère de ce dernier que par la couleur, puisque celui-ci est rouge. Il mesure donc également 9,45 cm de longueur, 8,10 cm de largeur et 2,60 cm d'épaisseur.

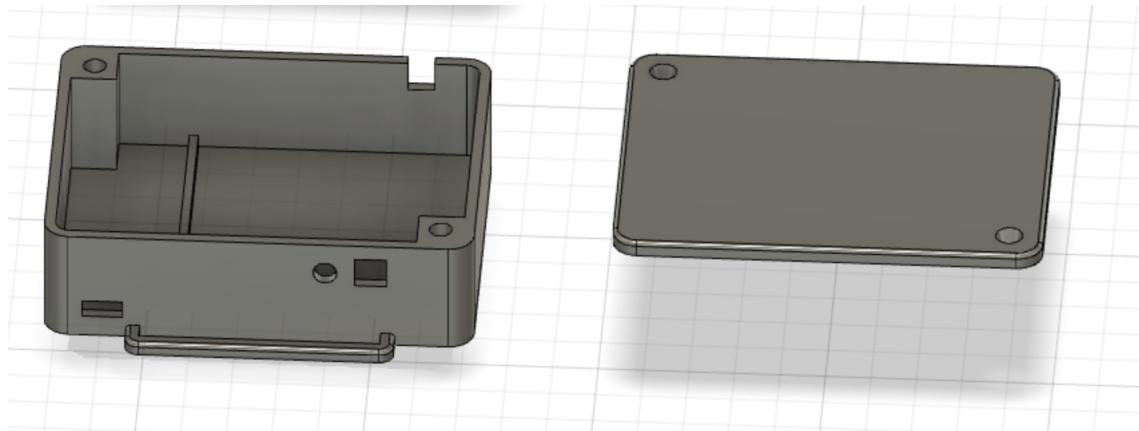


FIGURE 14.5: Plan 3D du prototype continu.

14.4 Construction

Comme on peut l'observer sur le schéma suivant (Figure 14.6), les montages des prototypes périodique et continu sont très similaires, l'unique différence résidant dans la présence d'un unique capteur de température dans le circuit électrique du prototype continu.

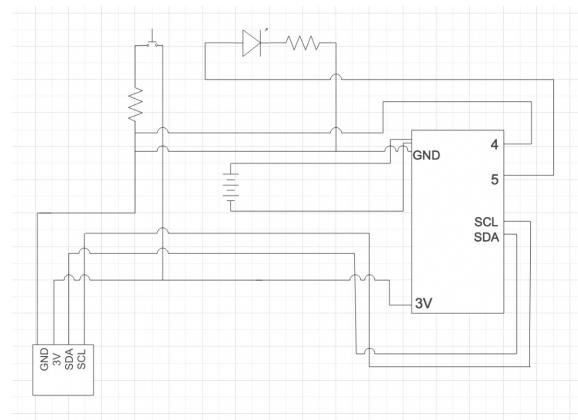


FIGURE 14.6: Schéma électrique du montage du prototype continu réalisé à l'aide du logiciel EdrawMax

Comme pour le dispositif périodique, le circuit du dispositif continu a été assemblé à l'aide de soudures et isolé avec de la gaine thermorétractable et du tape isolant.

Comme pour le dispositif périodique également, une erreur de montage de la batterie en court-circuit avec le capteur de courant a été commise au premier montage du circuit. Cependant, à l'inverse du dispositif périodique, la batterie n'était ici pas fournie avec un interrupteur et s'est donc entièrement déchargée, ce qui a donc empêché son recharge et l'a rendue inutilisable. Au deuxième montage, après suppression du capteur de courant du circuit, un autre problème s'est présenté et la batterie a été branchée à l'envers sur l'ESP32. Bien que cela ne présente aucun inconvénient supérieur au non fonctionnement du circuit, cet évènement a conduit le groupe à essayer de recharger cette batterie, ce qui l'a à nouveau entièrement déchargée et rendue inutilisable.

Après ces deux contre-temps majeurs, qui ont malheureusement beaucoup écourté le temps disponible pour les expériences demandées, les mises en circuit des batteries ont bénéficié d'une attention toute particulière et ne présentent, à l'heure actuelle, plus aucun souci.

15

Tests

15.1 Calibrage

Durant les premiers essais avec le prototype, il s'est rapidement avéré que le capteur de température donnait des valeurs décalées par rapport à la température corporelle du patient. Ainsi, plusieurs tests ont été effectuée afin de déterminer le calibrage adéquat.

Aux bouts de plusieurs tentatives, il fut décidé qu'un calibrage de +4.5°C semblait le plus approprié.

15.2 Expériences

Aux vues des conditions sanitaires, l'évaluation du prototype devait se faire à distance. Dans ce cadre, il fut demandé de réaliser plusieurs expériences, qui devaient être filmées et faire l'objet d'un compte rendu dans le présent rapport.

15.2.1 Expérience n°1

Fonctionnalités de base

La première expérience consistait en l'évaluation des fonctionnalités de base du prototype périodique, à savoir l'envoi des données mesurées par les capteurs vers InfluxDB. Il fut également demandé de mettre en évidence l'intensité du réseau WiFi auquel le prototype est connecté. Pour cela, le groupe s'est servi des différentes instructions nécessaires pour récolter des données de fréquence cardiaque, de température et de d'intensité du réseau. Ces trois données ont ensuite été stockées dans un objet "measurement" qui a été envoyé toutes les 40 secondes à la base de données. Les 3 graphiques sont repris dans les annexes (voir ??).

Il convient de préciser que, durant cette expérience, le prototype n'était pas directement porté, ce qui explique les valeur de 0 bpm et 20.6 °C.

Tentatives de reconnexion

Dans la suite de l'expérience 1, il fut demandé de simuler une perte de connexion avec le réseau WiFi du domicile ou la base de données et de faire en sorte que le prototype essaie de rétablir cette connexion.

Pour cela, il convenait d'utiliser une fonction qui fait office de "timer" et laisse l'ESP32 se reconnecter au réseau ou à la base de données pendant 10 secondes. Si cette tentative échoue, le microcontrôleur redémarre et réitère l'expérience.

Dans la vidéo de présentation du prototype, le lecteur constatera que trois tests différents ont été effectués. Ces tests sont :

1. La désactivation du réseau WiFi et de l'accès à Internet. Il est possible de voir sur le moniteur série que deux boucles de 10 secondes sont effectuées. A la fin de la première, il est annoncé "Délai de connexion dépassé pour le WiFi" à la fin de la première et "Délai de connexion dépassé pour InfluxDB" à la fin de la seconde.
2. La désactivation de l'accès à Internet seulement. L'esp32 va mettre une seconde avant de se connecter au réseau du domicile et annoncer "Connecté au WiFi". Cependant, le second timer va se dérouler et annoncer "Délai de connexion dépassé pour InfluxDB" au bout de 10 secondes.
3. Aucune désactivation. En moins de 2 secondes, le moniteur série confirme que le microcontrôleur est connecté au WiFi du domicile et à la base de données.

Enfin, la tension à été prise au borne de la batterie à l'aide d'une multimètre au début et à la fin de la première expérience. La valeurs sont de 4.040 V au début et 3.925 V à la fin de l'expérience.

15.2.2 Expérience n°2

L'expérience 2 consiste à tester l'autonomie et le confort du prototype périodique sur 14 jours, soit 336h. Comme pour les autres expériences, la batterie a été chargé au maximum au préalable et le dispositif est resté constamment allumé. Malheureusement, pour une raison encore inconnue, le prototype n'a pas pu tenir les 336h minimum du cahier des charges mais plusieurs tests ont été menés pour avoir des données et essayer de trouver la cause du problème. Lors de notre premier test, au bout d'un peu plus de 8h00, nous avons donc obtenu les résultats suivants (voir figure 15.1) :



FIGURE 15.1: Ensemble de graphiques sur Grafana des paramètres physiologiques et de la puissance du WiFi en fonction du temps fournis par le prototype périodique lors de l'expérience 2

Comme on peut le voir sur la figure 15.1 ci-dessus, des données sont bien obtenues pour la pulsation cardiaque, la température et la puissance du signal WiFi, à 5 minutes d'intervalle en utilisant la boucle deep-sleep pour minimiser au maximum la consommation. Seule la mesure de la tension a été prise manuellement une fois toutes les 3 heures et postée chaque fois sur le groupe Teams pour suivre l'évolution. Analysons les données de chaque graphique pour comprendre ce qui s'est produit (conclusions positives et négatives).

Pour la température, on peut remarquer sur la figure 15.2 une hausse lors de la première heure après le début du test. Cela correspond au fait que le prototype périodique a été porté pour vérifier que les valeurs sont correctes et bien envoyées à la base de données InfluxDB. En parallèle, le sys-

tème de monitoring a été lancé pour vérifier l'envoi des mails (voir 15.2.4). Concernant les valeurs en elles-mêmes, on constate que la température est de 35.76 °C au début du port du prototype puis atteint les 37.36°C à la fin, avec des valeurs qui varient entre 36.2°C et 36.8°C. On obtient donc bien des données cohérentes avec notre calibrage si on compare avec la littérature médicale où la température corporelle oscille entre 36.2°C et 37.7°C. Dès que le prototype a été retiré, on observe que la température redescend à des valeurs variant entre 18.91°C et 21.23°C (de 16h37 à 23h15), ce qui est tout à fait cohérent étant donné que le capteur de température fonctionne toutes les 5 minutes et prélevent la température de la pièce dans laquelle il se trouve.



FIGURE 15.2: Graphique de la température en fonction du temps fourni par le prototype périodique lors de l'expérience 2

En ce qui concerne la pulsation cardiaque, on observe sur le graphique 15.3 ci-dessous des données assez variables lors du port du prototype, avec des pulsations allant de 70 bpm à 0 bpm puis remonte à 30bpm et redescend finalement à 0 avant même de retirer le dispositif. Ce phénomène peut s'expliquer par le fait que la zone où le prototype est porté (c'est à dire au dessus du coude mais en dessous du biceps) est composé de beaucoup moins de capillaires sur sa surface, rendant la tâche plus difficile pour détecter un pouls plus en profondeur. Tandis que sur la surface de doigt, qui est beaucoup plus dense en capillaire, le capteur est beaucoup plus performant car on obtient des valeurs très précises en un court laps de temps. Mis à part cela, dès le retrait du dispositif, la pulsation cardiaque stagne à 0 bpm, ce qui est cohérent étant donné que le capteur n'est pas en contact avec la peau et ne reçoit pas de valeurs.

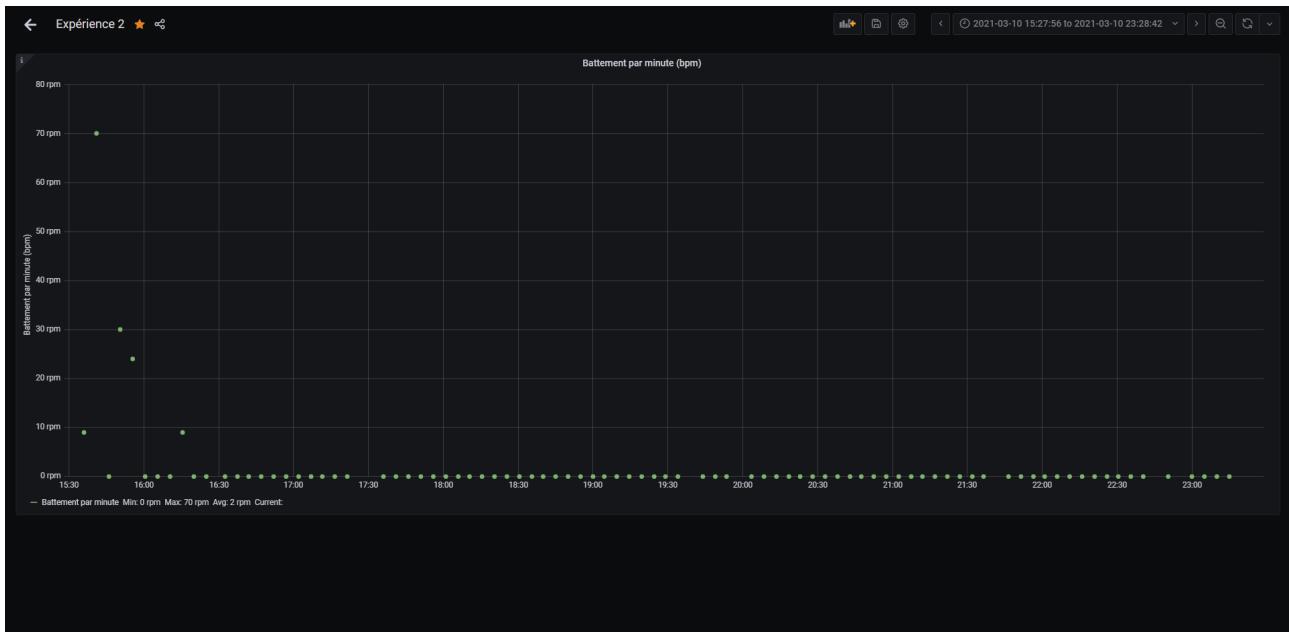


FIGURE 15.3: Graphique de la pulsation cardiaque en fonction du temps fourni par le prototype périodique lors de l'expérience 2

Pour ce qui est de la puissance du WiFi, le RSSI est envoyé correctement si on regarde la figure 15.4 ci-dessous, avec des valeurs de -60dBm à -12dBm. Lors du port du prototype, la personne en charge du test s'est également déplacée momentanément dans la pièce et on peut voir que le RSSI varie de -47 dBm à -53 dBm de 15h36 à 16h36. Dès le retrait du prototype, le dispositif n'a pas bougé de place et on peut remarquer que sur l'heure qui suit le retrait, la valeur de RSSI stagne autour des -35dBm. Enfin, à partir de 19h, le prototype a été rapproché du routeur WiFi et on obtient des valeurs autour de -22 dBm, qui témoignent de la proximité de l'appareil et confirment les valeurs de RSSI fournis par l'ESP32.

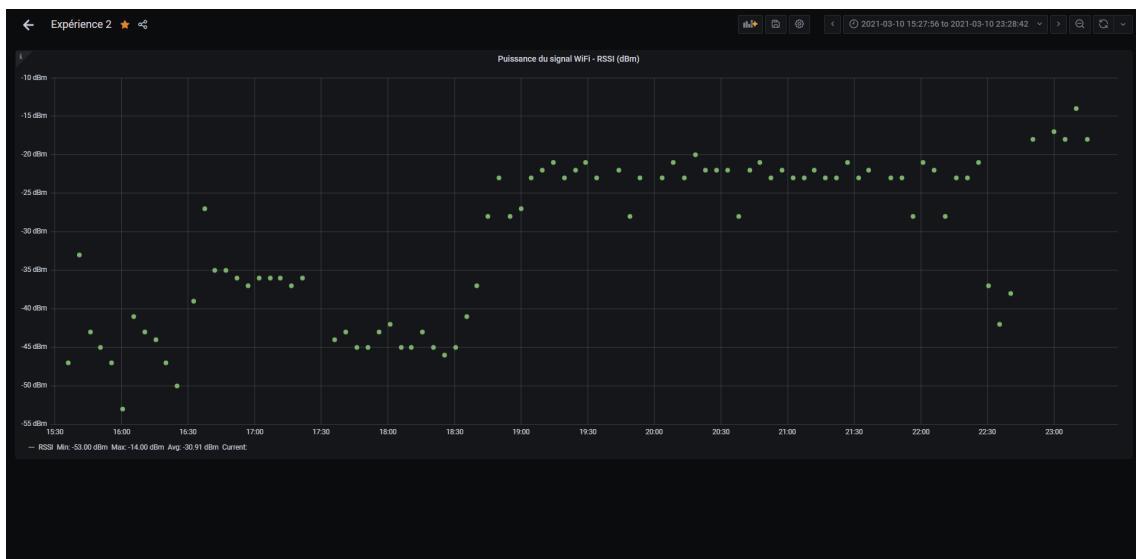


FIGURE 15.4: Graphique de la puissance du signal WiFi en fonction du temps fourni par le prototype périodique lors de l'expérience 2

Enfin, la tension de la batterie a été prise manuellement toutes les 3 heures pour suivre l'évolution de la décharge de la batterie. Le capteur de courant était initialement prévu pour mesurer automatiquement la tension mais a été finalement retiré après concertation avec le groupe suite à l'incident avec la batterie fournie par les organisateurs du projet. Par précaution donc, il a été décidé de mesurer manuellement la tension avec un multimètre pour garder un prototype fonctionnel pour les expériences. Cela étant dit, voici les résultats obtenus pour la tension :

Temps en heures depuis la première mesure	Tension de la batterie de 1500 mAh
0	4.066V
3h	3.880V
7h	3.652V
8h	3.442V

TABLE 15.1: Évolution de la tension de la batterie 1500 mAh au cours de l'expérience 2

Comme on peut le voir dans le tableau 15.1 ci-dessus, la tension diminue en moyenne de 0.06V toutes les heures, ce qui est beaucoup trop par rapport à ce qui a été calculé en théorie pour notre boucle deep-sleep (0.00192V, soit 30fois plus par heure!). Face à cet évènement inattendu, une série de tests complémentaires ont été mené pour déterminer l'origine du problème. Cela fera l'objet de la section suivante.

15.2.3 Tests complémentaires

Pour les tests complémentaires, il a été question dans un premier temps de lister tout ce qui serait susceptible de modifier la consommation totale de courant pendant l'expérience. Il y a la batterie de 1500 mAh, les capteurs, l'ESP32, le code téléchargé et le cycle deep-sleep. Pour chaque "suspect" potentiel, un test ou une re-vérification des données a été mené pour éliminer tous les doutes que le groupe avait. Ainsi, avec un raisonnement logique, des hypothèses ont été émises pour essayer de trouver la source du problème.

Hypothèse n°1 : la batterie

La première chose qui a été vérifié est la capacité réelle de la batterie de 1500 mAh fournie par les organisateurs du projet. En effet, il y a un risque que la capacité de cette dernière ne soit pas celle indiquée ce qui fausserait tous nos résultats. Pour partir sur de bonnes bases, il fallait exclure cette éventualité. Pour ce faire, le groupe a effectué la même expérience, mais avec une batterie

de 500 mAh¹. Si l'expérience 2 avec la batterie de 1500 mAh a duré 8h, celle de 500 mAh devrait logiquement tenir 2.66h, soit 2h et 40 minutes environ. Sur le graphique 15.5 ci-dessous, le même test a été fait et la batterie a effectivement tenu 2h et 35 minutes, ce qui nous permet d'affirmer que la batterie de départ était sans soucis.

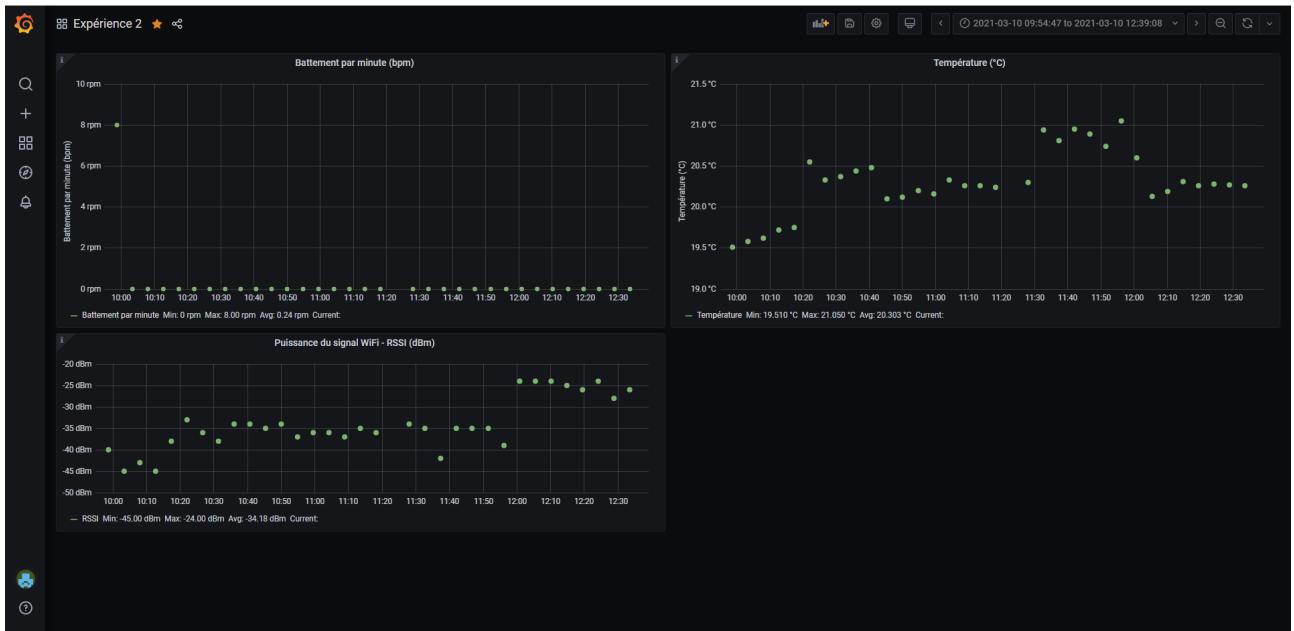


FIGURE 15.5: Test complémentaire à l'expérience 2 avec une batterie de 500 mAh

Si le prototype périodique n'a su tenir que 8 heures contrairement au 351 heures en théorie, cela veut dire que le dispositif consommait en moyenne 187.5 mA par heure contre les 4.2 mA par heure initialement calculé, soit environ un excès de 183 mA chaque heure, ce qui est énorme pour un dispositif censé être à 95% du temps en deep-sleep sur les 5 minutes avec une consommation de 10µA. Pour atteindre les 187.5 mA tout en gardant les 95% de deep-sleep à la même consommation fournie dans la datasheet, il faudrait que sur les 5% des 5 minutes, l'ESP32 doit consommer jusqu'à 3700mA durant ce laps de temps, ce qui est clairement impossible. Le soucis viendrait alors soit du mode deep-sleep qui consomme davantage que prévu, soit du WiFi qui ne s'éteint pas correctement avant d'entrer en hibernation, soit les deux. Mais avant de s'intéresser à cette hypothèse, il convient malgré tout de passer en revue les différents capteurs pour les exclure du problème.

1. Dans une tentative de remplacer les batteries défectueuses, le groupe a pu récupérer d'autres batteries grâce à des connaissances

Hypothèse n°2 : les capteurs

Si l'on se réfère aux datasheets respectives du STS35 et du MAX30102, le premier consomme en mode actif ou single shot mode (c'est à dire lorsqu'il prend une mesure) de 0.6 jusqu'à 1.5 mA et le dernier consomme 6.8 jusqu'à 7.4 mA (les 6.2 mA supplémentaires proviennent de la LED rouge qui aide à prendre le pouls) [Sena] [Maxa]. Dans le mode de fonctionnement ainsi que dans le code, sur les 300 secondes de chaque cycle, le capteur cardiaque est programmé pour s'allumer pendant 10 secondes pour la prise du pouls et le capteur de température moins d'une seconde pour prendre sa mesure, tous les deux sont ensuite éteints pour minimiser la consommation pendant la période d'hibernation de l'ESP32. Il est quasi-certain que le capteur cardiaque n'est plus alimenté ou que très peu (0.7 µA en Ultra Low Shutdown Current) étant donné qu'après avoir effectué sa mesure, la LED rouge s'éteint. A priori, il en va de même pour le capteur de température. Pour vérifier cela différemment, la commande `adc_power_off()` a été ajouté et permet sur Arduino IDE de couper directement l'alimentation des ports ADC (la commande `adc_power_on()` est bien évidemment appelé dès le réveil de l'ESP32). Même si les capteurs sont laissés allumés volontairement, le STS35 ne consommera que 1.5mA pendant 1h soit au bout de 8h, 12mA et pour le MAX30102, 7.4mA pendant 1h, soit au bout de 8h, 59.2mA (ce qui est peu probable étant donné que la lumière du capteur est éteinte). Ceci n'explique donc pas l'excès de consommation qui a été rencontré lors du l'expérience 2.

Hypothèse n°3 : le code téléversé

Le code du prototype périodique a été passé au peigne fin pour vérifier si aucune erreur ne s'y est glissé par distraction. Des tests effectués sur l'ESP32 connecté en USB, tout fonctionnait et s'affichait sur le port de communication de l'Arduino. Le Wifi et les capteurs sont bien éteints avant de rentrer en mode hibernation. Malheureusement, ce n'est qu'en connectant la première fois la batterie et après avoir fait un test complet qu'il s'est avéré qu'il y avait un excès de consommation de courant dont on ne connaît pas l'origine. L'hypothèse que le code soit le soucis ne peut pas vraiment être vérifié, étant donné que dans les faits, des données sont obtenues à intervalle régulier mais avec une autonomie limitée de 8h.

Hypothèse n°4 : le cycle deep-sleep

Le cycle deep-sleep qui a été implémenté consiste à prendre des mesures grâce aux différents capteurs pendant 10 secondes, les envoyer pendant au maximum 3 secondes et passer le dispositif en mode hibernation (deep-sleep) les 287 prochaines secondes. Au total, des cycles de 5 minutes

sont obtenus avec l'ESP32 à 95% du temps en mode faible consommation. En reprenant les différentes valeurs des datasheets, les calculs théoriques ont permis d'obtenir une consommation moyenne de 4.2731 mA, soit avec une batterie de 1500 mAh, une autonomie de 351h (voir équations (15.2) et (15.3)) :

$$\frac{287}{300} \text{consodeepsleep} + \frac{10}{300} \text{consumodemsleep} + \frac{3}{300} \text{consostativemode} = \text{consototale} \quad (15.1)$$

$$\frac{287}{300}(0.2 \cdot 10^{-3} + 10 \cdot 10^{-3} + 10 \cdot 10^{-3}) + \frac{10}{300}(1.5 + 7.4 + 47) + \frac{3}{300}(0.2 \cdot 10^{-3} + 10 \cdot 10^{-3} + 240) = 4.2731 \text{mA} \quad (15.2)$$

$$\frac{1500 \text{mAh}}{4.2731 \text{mA}} = 351.02 \text{h} \quad (15.3)$$

Or, le prototype n'a su tenir que 8h lors de l'expérience 2, s'il y'a bien un élément qui peut influencer de façon significative la consommation totale, c'est bien le deepsleep car il intervient 95% du temps. Un test a donc été fait avec une batterie de 500 mAh et le même prototype pour vérifier que ce mode ne consomme pas plus que les 10 µA inscrit dans la datasheet de l'ESP32. Dans le code téléchargé, l'ESP32 devait resté allumé pendant 10 secondes sans prendre de mesures et retourner en mode hibernation pendant 287 secondes et voici les résultats obtenus :

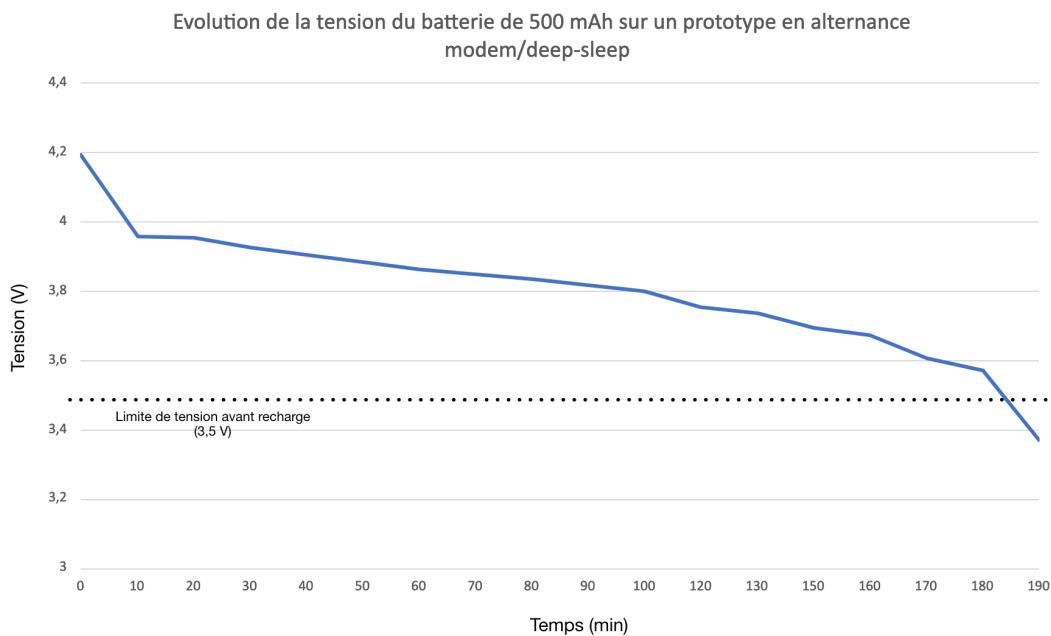


FIGURE 15.6: Evolution de la tension sur le prototype périodique avec cycle deepsleep sans WiFi

On voit que la tension de la batterie décroît très rapidement et le prototype s'est arrêté après 3.06h. Or, si l'on recalcule la consommation selon le nouveau mode de fonctionnement comme aux équations (15.2) et (15.3), on a :

$$\frac{287}{297}(10 \cdot 10^{-3}) + \frac{10}{297}(47) = 1.5921 \text{ mA} \quad (15.4)$$

$$\frac{500 \text{ mAh}}{1.5921 \text{ mA}} = 314.03 \text{ h} \quad (15.5)$$

Le dispositif aurait du tenir au moins 300h, soit 100 fois plus qu'en pratique. Le problème vient donc vraisemblablement du cycle deepsleep qui consomme beaucoup trop par rapport à sa véritable valeur de 10µA. L'ESP32 aurait sans doute du mal à réguler la consommation dès que ce dernier rentre en mode hibernation. Tentons à présent de déterminer la réelle consommation en mA lorsque le prototype effectue sa boucle (voir équations (15.6) et (15.7)) :

$$\frac{500 \text{ mAh}}{3.06 \text{ h}} = 163.39 \text{ mA} \quad (15.6)$$

$$\frac{287}{297}(x) + \frac{10}{297}(47) = 163.39 \text{ mA} \quad (15.7)$$

$$\Leftrightarrow x = 167.44 \text{ mA} \quad (15.8)$$

On a une consommation de 167.44 mA pour le deepsleep, ce qui est excessivement élevée. En injectant cette valeur dans notre premier calcul à l'équation (15.2) pour la batterie de 1500 mAh, on obtient :

$$\frac{287}{300}(0.2 \cdot 10^{-3} + 10 \cdot 10^{-3} + 167.44) + \frac{10}{300}(1.5 + 7.4 + 47) + \frac{3}{300}(0.2 \cdot 10^{-3} + 10 \cdot 10^{-3} + 240) = 164.45 \text{ mA} \quad (15.9)$$

$$\frac{1500 \text{ mAh}}{164.45 \text{ mA}} = 9.12 \text{ h} \quad (15.10)$$

On remarque qu'on obtient une autonomie de 9.12h, qui est assez proche des 8h tenu par le prototype lors de l'expérience. Donc, l'hypothèse la plus probable jusqu'à présent est que le deepsleep soit à l'origine de la consommation supplémentaire, ce qui a limité fortement le dispositif à 8

heures de fonctionnement.

Hypothèse n°5 : l'ESP32

L'hypothèse que l'ESP32 soit défectueux ou bien endommagé a été envisagé dans la mesure où ce dernier est utilisé tout au long de l'expérience 2 et pour les tests complémentaires. Aucun dégâts visibles sur le microcontrôleur n'a été signalé et donc ne permet de valider cette hypothèse car pour ce faire, il aurait fallu faire des test plus en profondeur en vérifiant chaque composant, ce qui est en dehors de nos capacités. Un doute a cependant été émis concernant la puce de WiFi de l'ESP32 qui consommerait plus que prévu étant donné que le prototype n'a duré que 8h et qu'en théorie, si le Wifi du dispositif restait constamment allumé, on obtiendrait une autonomie de 6.25 heures avec une capacité de 1500 mAh. Dans la datasheet du constructeur, on peut voir que la consommation de l'ESP32 avec le WiFi activé tourne autour des 240 mA [ESP20a]. Un test a été réalisé avec une batterie de 500 mAh où le prototype envoie des données toutes les secondes. Une autonomie de 2.08 heures a été calculé en théorie et une de 2.13 heures a été rencontrée en pratique en mesurant la tension toutes les 10 minutes :

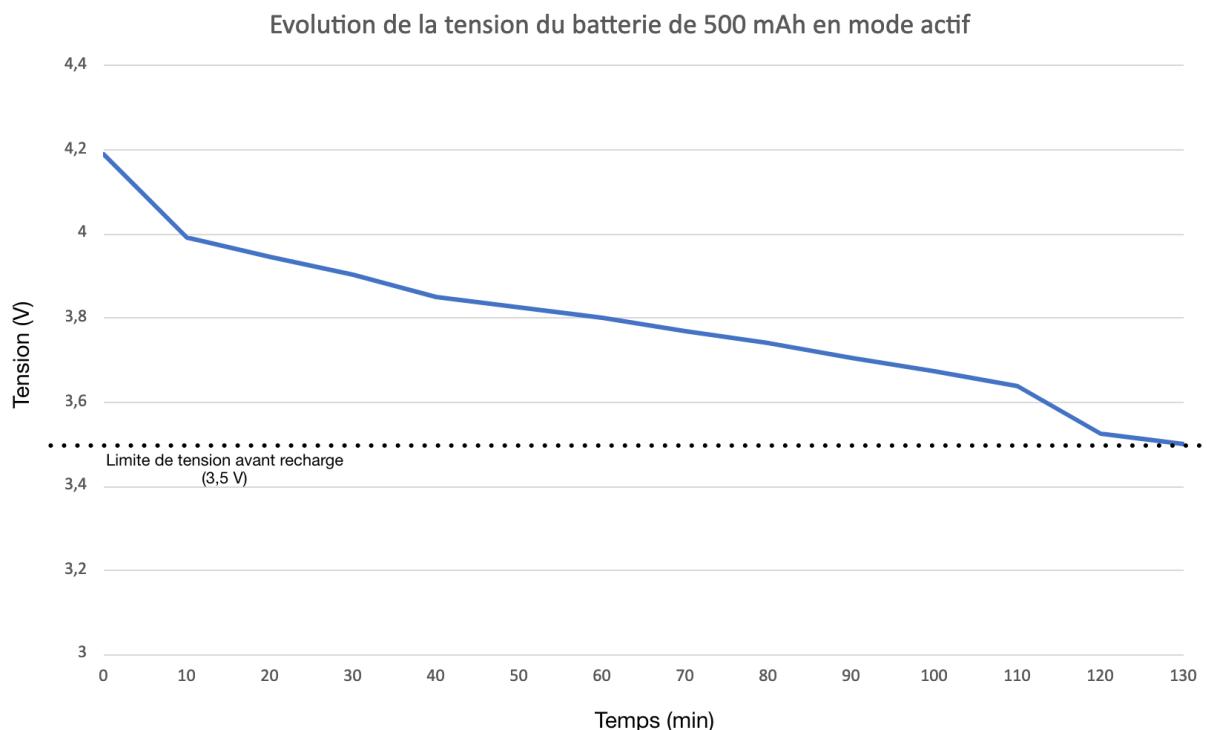


FIGURE 15.7: Test complémentaire à l'expérience 2 avec une batterie de 500 mAh

En multipliant par 3 la durée obtenue pour une batterie 500 mAh, on retombe bien sur les 6.25 heures d'autonomie prévus pour celle de 1500 mAh. On peut donc conclure, sur base du gra-

phique 15.7, que le WiFi ne consomme pas plus que prévu et que le problème ne proviendrait pas de là.

Conclusion des hypothèses

En conclusion, chaque hypothèse a été vérifiée et a permis de tirer des réponses concernant le problème qui a été rencontré. L'hypothèse la plus probable reste le cycle deep-sleep où l'on remarque suite au test que l'excès de consommation provient vraisemblablement d'une mauvaise gestion de l'ESP32 lorsque celui-ci doit basculer en mode hibernation.

15.2.4 Expérience n°3

La troisième expérience a été conduite en parallèle à la deuxième expérience. Plus précisément, le monitoring était activé lorsque le prototype était porté, sans quoi il y aurait eu un envoi constant de mail demandant de remettre correctement les capteurs. En effet, lorsque l'état du patient est anormal, données nulles ou aberrantes, un mail est envoyé à partir de l'adresse mail "biomed3ulb@gmail.com" pour prévenir le patient de son état. Si tout se passe bien, aucun mail ne sera envoyé

Durant le premier test, le calibrage des températures était trop haut de 1 degré, ce qui a finalement permis de simuler une montée de fièvre, voir 15.8

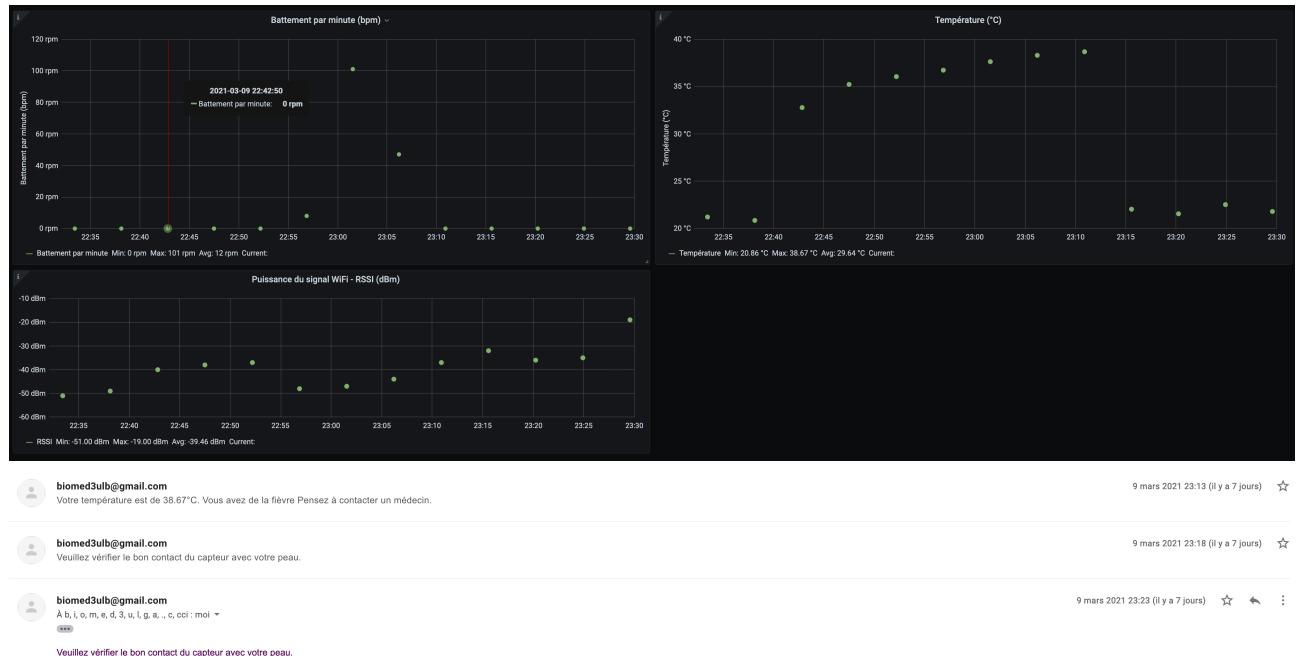


FIGURE 15.8: Comparaison des valeurs de températures et des e-mails envoyés

A partir de 23 :05, le patient indique une température corporelle de plus de 38°C. Le système de monitoring attend alors la prochaine valeur pour voir si elle est également au dessus de 38°C, ce qui est le cas. Un mail est donc envoyé à la personne lui signalant :

"Votre température est de 38.7°C. Vous avez de la fièvre. Pensez à contacter un médecin."

Ensuite, le capteur n'est plus en contact avec la peau, Le système de monitoring attend de recevoir deux valeurs en dessous de 30°C avant d'envoyer comme mail :

"Veuillez vérifier le bon contact du capteur avec votre peau."

En ce qui concerne le capteur cardiaque, le groupe a pris un autre intervalle de temps où le prototype était porté et le monitoring activé. Comme le montre l'image ci-dessous, voir 15.9, la fréquence cardiaque est nulle jusque 22 :05. Un mail est donc bien envoyé à 22 :07 pour prévenir le patient de remettre son capteur correctement. Ensuite, le capteur est mis et donne une valeur de 108 bpm à 22 :15. Le patient reçoit donc bien un mail qui informe la personne que "Votre fréquence cardiaque est de 108.0Bpm. Votre fréquence cardiaque est élevée ou instable Pensez à contacter un médecin."



FIGURE 15.9: Comparaison des valeurs de fréquences cardiaque et des e-mails envoyés

15.2.5 Expérience n°4

Pour l'expérience 4 qui concerne le prototype continu, il a été demandé de réaliser un test de 10h à 12h en portant le dispositif 30 minutes à 1 heures et en envoyant les données dans InfluxDB de façon continue. Une batterie imposée de 1500 mAh a été fournie pour ce test. Le prototype s'est arrêté après 8h de fonctionnement dans un premier test avec le port du dispositif pendant 35min. Cependant, une heure avant l'arrêt du prototype, la tension de la batterie était de 3.874V, ce qui suggère que le prototype pouvait à priori encore fonctionner quelques heures sans problèmes. Les résultats obtenus sont les suivants :



FIGURE 15.10: Graphiques sur Grafana des paramètres physiologiques en fonction du temps fournis par le prototype continu lors du test

On peut voir sur la figure 15.10 que des valeurs de température, de pulsation cardiaque et d'oxygène sont envoyées toutes les 5 secondes. Les mesures de tension, comme pour le prototype périodique, ont été prises manuellement toutes les heures cette fois-ci pour avoir une approche plus continue, étant donné que l'expérience ne doit durer qu'entre 10 et 12h. Analysons les données de chaque graphique pour en tirer des conclusions.

Pour la température, on observe sur le graphique 15.11 qu'on obtient des valeurs variant de 35.7°C à 37.13°C, avec un pic de 38.1°C lorsque le prototype était porté pendant les 35 min de test (de 17h08 à 17h43). Après le retrait, la température oscille autour de 20°C, comme avec le prototype périodique, car le capteur est exposé à l'air de la pièce, et mesure donc la température ambiante.

Dans l'ensemble, les valeurs obtenues sont cohérentes avec seulement un pic de 38.1°C qui peut être expliqué par l'échauffement du capteur lorsque le brassard est adapté sur le bras.

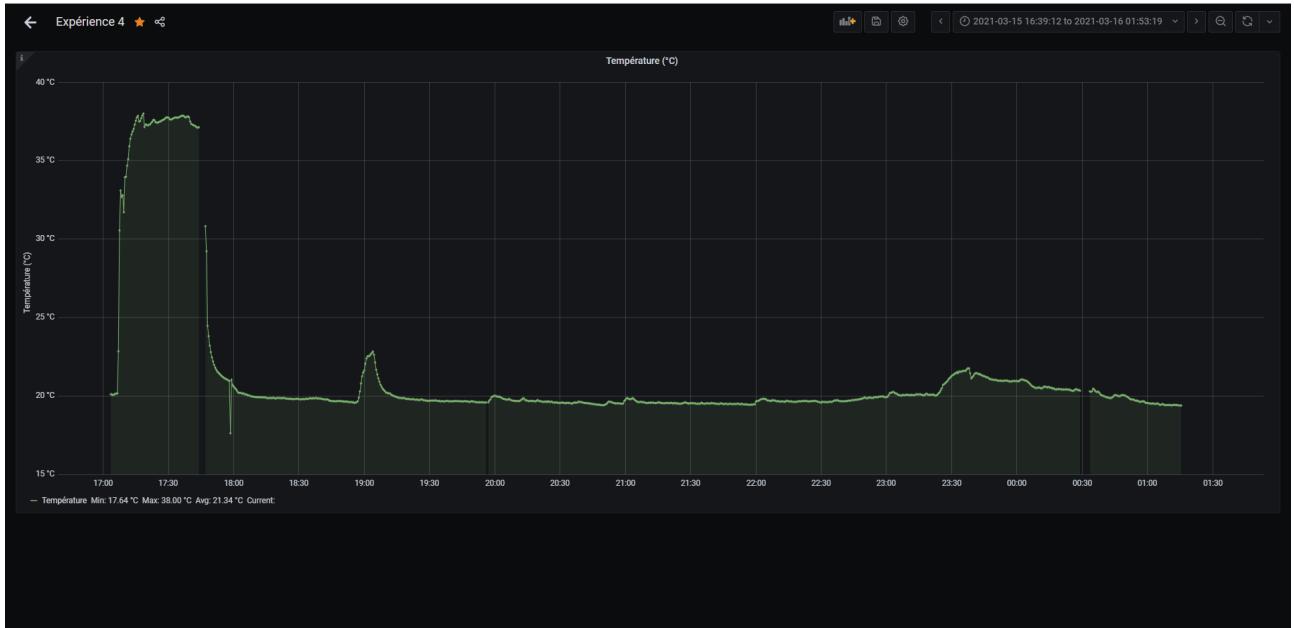


FIGURE 15.11: Graphique de la température en fonction du temps fourni par le prototype continu lors de l'expérience 4

En ce qui concerne la pulsation cardiaque, les valeurs obtenues sur la figure 15.12 pendant le port du prototype sont assez correctes dans l'ensemble avec 80 jusqu'à 90 bpm mais des pics de 110 voire 120 bpm sont reçus par l'oxymètre, ce qui semble assez élevé étant donné que la personne était censé être au repos lors du l'expérience. On ne peut pas vraiment savoir si l'oxymètre est fautif étant donné que ce sont des données qu'il calcule lui même et qu'il renvoie directement par bluetooth. A part cela, on obtient bien des valeurs égales à 0 lorsque l'oxymètre est retiré

Pour le SPO₂ sur le graphique 15.13, les données stagnent à 99% lorsque l'oxymètre est porté et à 0% dès le retrait de ce dernier. Il n'y a donc aucun soucis à signaler sur ce paramètre physiologique, tout semble bien fonctionné.

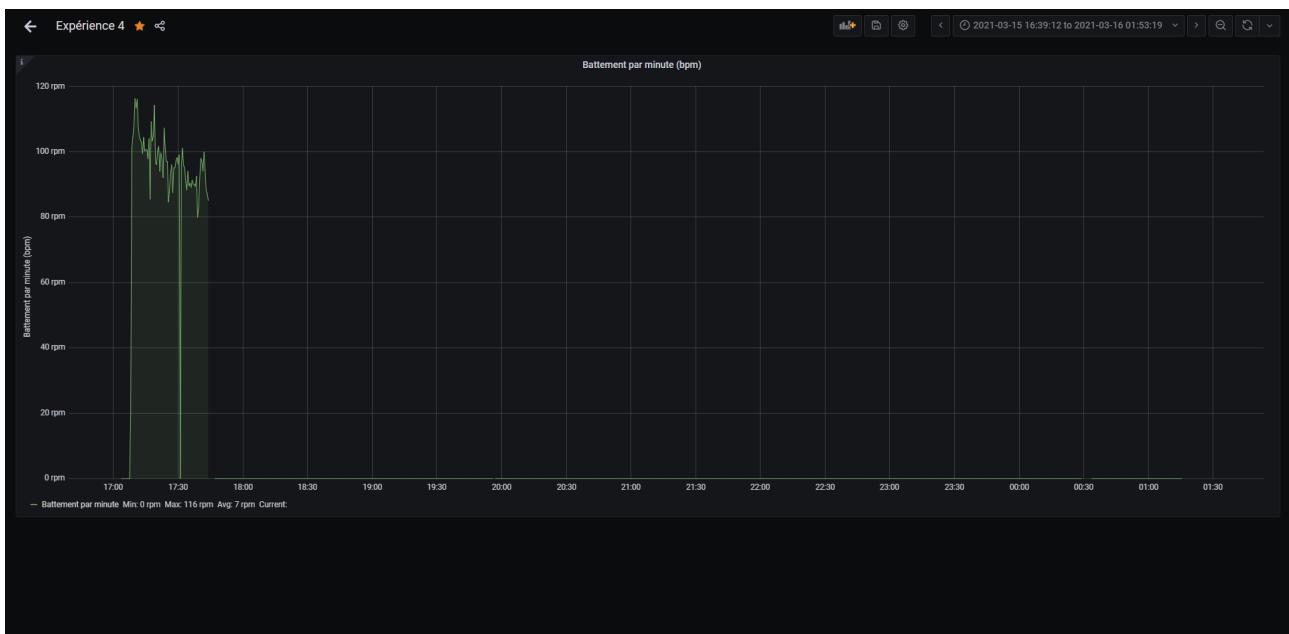


FIGURE 15.12: Graphique de la pulsation cardiaque en fonction du temps fourni par le prototype continu lors de l'expérience 4



FIGURE 15.13: Graphique du taux d'oxygène dans le sang (SP02) en fonction du temps fourni par le prototype continu lors de l'expérience 4

Enfin, finissons avec la tension mesurée toutes les heures et regardons l'évolution de la décharge. On constate depuis le tableau 15.14 que la batterie dispose d'une tension raisonnable après 8h de test. Cependant, étant donné que l'expérience a continué la nuit, le reste des tensions n'a pas pu être mesurée. En même temps, l'ESP32 s'est brusquement arrêté quelques minutes après 8h sans en comprendre la cause. Mais en extrapolant sur base de l'évolution de la tension, on peut conclure que le prototype aurait pu tourner jusqu'à atteindre au minimum les 10h convenues pour cette expérience. En effet, le même test a été fait quelques jours auparavant avec une batterie de 500 mAh pour vérifier le bon déroulement avant l'expérience finale et a pu tenir un peu plus de 5 heures. En multipliant par 3 pour une batterie de 1500 mAh, on arrive à 15 heures d'autonomie, comme montré dans l'extrapolation par une droite sur le graphique 15.14.

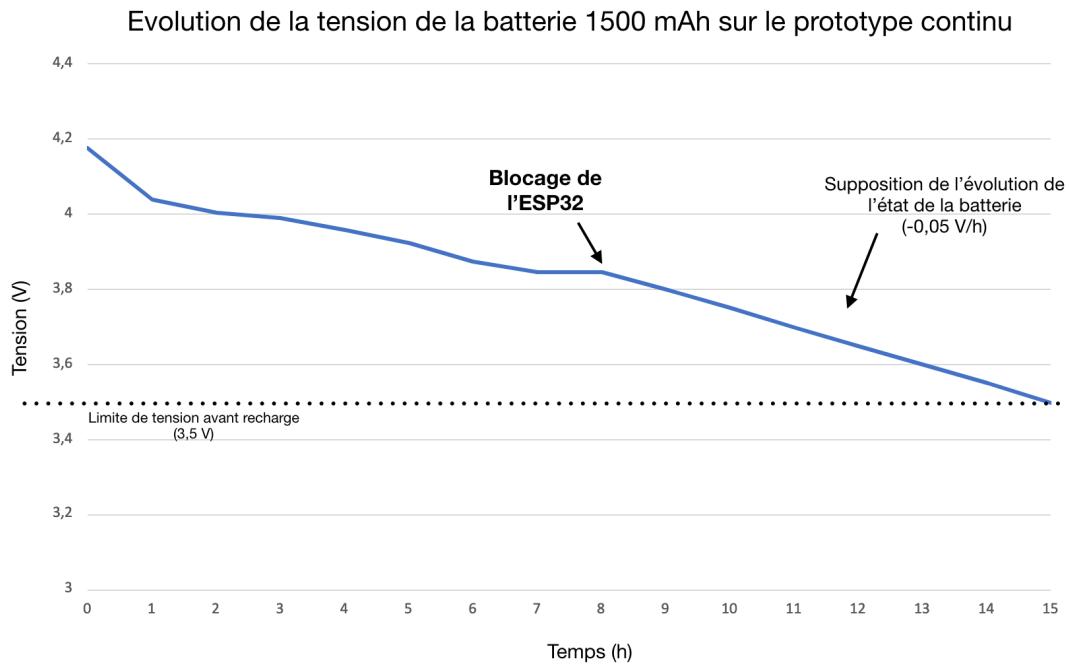


FIGURE 15.14: Graphique de la tension de la batterie de 1500 mAh en fonction du temps mesurée avec un multimètre lors de l'expérience 4 et extrapolation

15.2.6 Confort et ergonomie des prototypes

Les deux prototypes disposant d'une boîte similaire, un commentaire général sera fait pour expliquer le confort lors du port des prototypes. Ils disposent tous les deux d'une extension filaire pour les capteurs à insérer dans le prolongement du brassard pour bien se positionner sur la peau une fois refermé. L'installation des prototypes était plutôt simple à effectuer, il suffisait de fermer la sangle, d'insérer sa main en premier et d'ensuite le glisser jusqu'à atteindre la zone qui se situe juste après le coude. Pour la personne qui a effectué les tests, la taille de la sangle était adéquate et peut même convenir pour des personnes disposant de plus gros bras. Une fois fixé, le prototype ne gênait pas beaucoup et appliquait une bonne pression. Cependant, les capteurs étant libre de bouger, il a fallu à plusieurs reprises les déplacer pour bien les positionner dû à une certaine résistance des fils. Pour combler cela, il a été décidé dans le chapitre 4 au point 4, d'installer directement les capteurs à proximité de la boîte pour ne plus rencontrer de soucis. En mouvement, le prototype était plutôt solide, il n'y avait pas mouvement de la batterie et de l'ESP32 à l'intérieur étant donné qu'ils ont été calés à l'intérieur de la boîte. Par contre, il ne faut pas effectuer de mouvement brusque avec et l'utiliser avec précaution pour ne pas endommager les câbles soudés à l'intérieur, ce qui semble assez naturelle. Pour finir, la taille de la boîte était ni trop grande, ni trop petite, mais une diminution légère de la hauteur aurait été apprécié pour pouvoir glisser le prototype entre la peau et un vêtement comme un pull par exemple. Néanmoins, il était tout à fait possible de mettre un habit plus ample par dessus le bras pour se protéger du froid par exemple. Pour visualiser des photos de l'installation du prototype, un manuel d'utilisation se trouve à l'annexe D.

16

Mesures d'hygiène

Dans le cas de la crise sanitaire, il a fallu redoubler les mesures d'hygiènes pour éviter au maximum une propagation du virus. Entre la désinfection des mains, le port du masque en extérieur et le nettoyage/stérilisation des objets, le dispositif que l'utilisateur porte ne fait pas exception. C'est pour cela qu'il faut informer le patient et les équipes médicales des produits à utiliser pour désinfecter le dispositif. Celui-ci se divise en 3 parties facilement détachables les unes des autres : un boîtier en plastique, un brassard et un circuit avec les capteurs.

Si l'utilisateur veut nettoyer le prototype en cas de résidus désagréables (sueurs ou tâches apparentes), il peut nettoyer le boîtier avec un produit tel que de l'eau savonneuse déposée sur un chiffon. Le brassard, quant à lui, peut être mis à la machine à laver, à la température la plus haute possible.

L'équipe médicale s'occupera de la désinfection du prototype lors du passage du dispositif entre deux patients. Il conviendra dans ce cas-ci de retirer préalablement le circuit intérieur, afin de désinfecter la boîte de fond en comble et de minimiser les risques d'endommagement du circuit. L'enveloppe est en plastique, elle peut donc être désinfecter à l'aide d'eau javel diluée ou d'un désinfectant à base d'alcool [Nor20]. Pour la sangle, il suffit, comme pour le nettoyage, de la mettre à la machine à laver. Plus la température est haute, plus la désinfection est totale. [RAD]

17

Conclusion

Au terme du projet de BA2, l'objectif était de concevoir, dans le cadre de la lutte contre la Covid-19, deux prototypes (un périodique, et un continu) respectant le cahier des charges, ainsi qu'un service de traitement de données qui alerte également le patient et les équipes médicales si nécessaire, une simulation qui permet de tester ce code, et une interface qui permet au patient de consulter les mesures effectuées par les prototypes.

Dans ce cadre, le groupe Biomed3 a tenté de concevoir un prototype périodique - qui utilisait initialement des capteurs de température, de pouls, et de courant - et un prototype continu - qui utilisait aussi un oxymètre (et son capteur cardiaque intégré). Le choix des capteurs s'est fait après analyse des symptômes du Sars-Cov-2, mais aussi des conditions du marché. Dans cette conception, le groupe s'est heurté à des problèmes de batterie, qui ont mené au retrait du capteur de courant sur les deux prototypes afin de minimiser les risques et d'assurer un prototype fonctionnel. Il a également fallu analyser les possibles méthodes de désinfection des prototypes.

Parallèlement à la conception et à la construction des prototypes, qui s'est déroulée tout au long de l'année, un code de traitement de données a été conçu : un code analyse les données envoyées par l'ESP32, l'analyse, et prévient les personnes concernées par mail si des résultats surprenants apparaissent. Un code de simulation fut également développé pour tester ce dispositif.

Deux interfaces graphiques furent aussi paramétrées et développées pour permettre au patient et aux équipes médicales d'avoir accès aux données : d'une part Grafana (une interface graphique), et d'autre part un site web, développé en local. Ce dernier permet d'avoir accès aux mesures extraites de la base de données et formatées sous forme de tableau (un par mesure), mais contient également une page dite de notifications, ou l'entièreté des analyses du système de monitoring apparaît. De plus, il héberge également un questionnaire, qui permet aux équipes médicales d'évaluer les symptômes non-mesurables de la Covid-19.

Les tests d'évaluation de projet ont permis de solidifier les liens entre ces parties : Grafana fut

utilisé pour vérifier l'envoi des données, et les expériences menées sur les prototypes ont permis de vérifier la fonctionnalité du système de traitement de données et la fonctionnalité du site web. Toutefois, des problèmes d'autonomie des prototypes sont apparus au cours de ces expériences, et des tests complémentaires furent effectués dans une tentative d'expliquer l'origine de ce défaut.

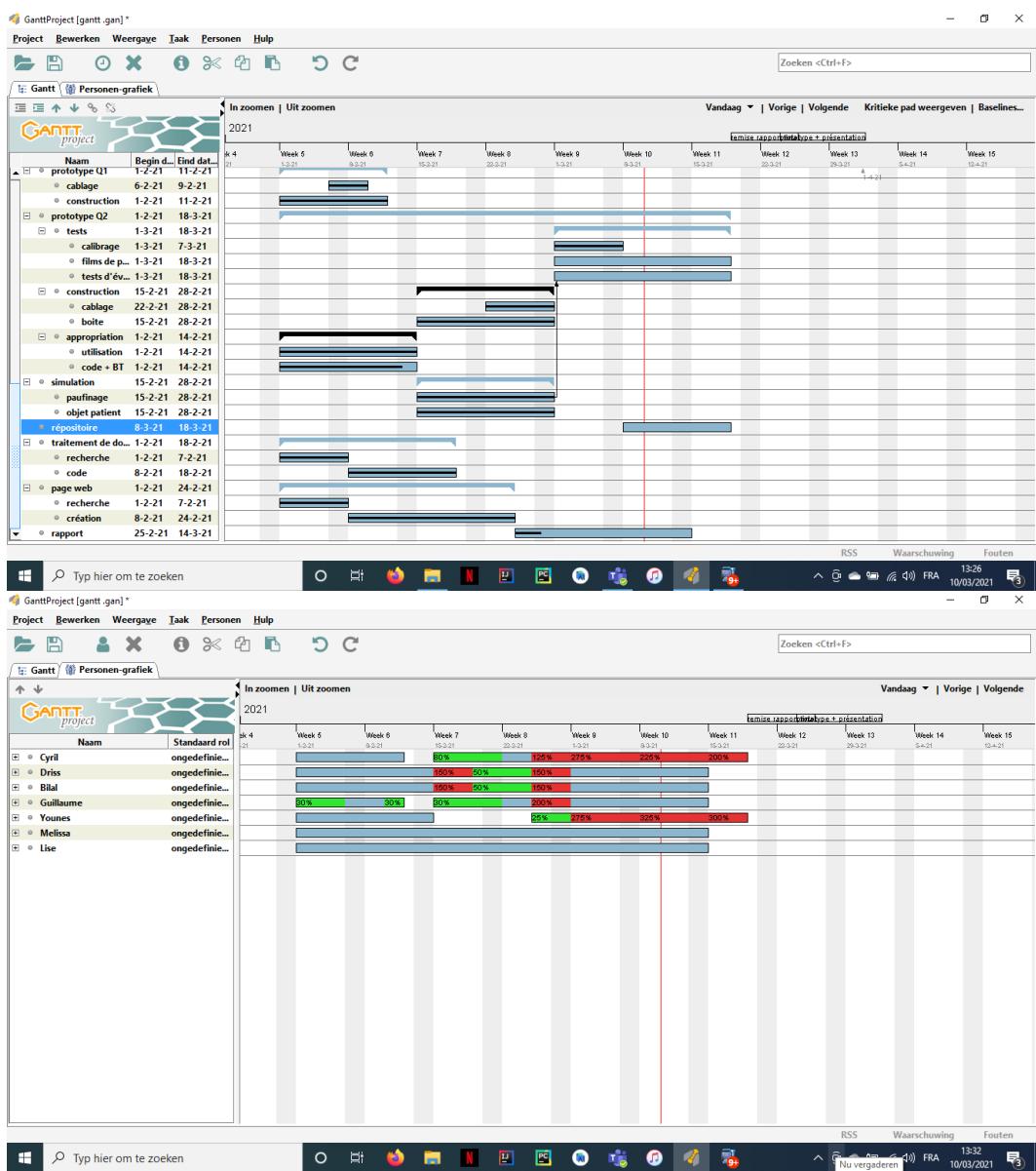


FIGURE A.1: Diagramme de gantt et répartition des tâches de ce 2eme quadrimestre

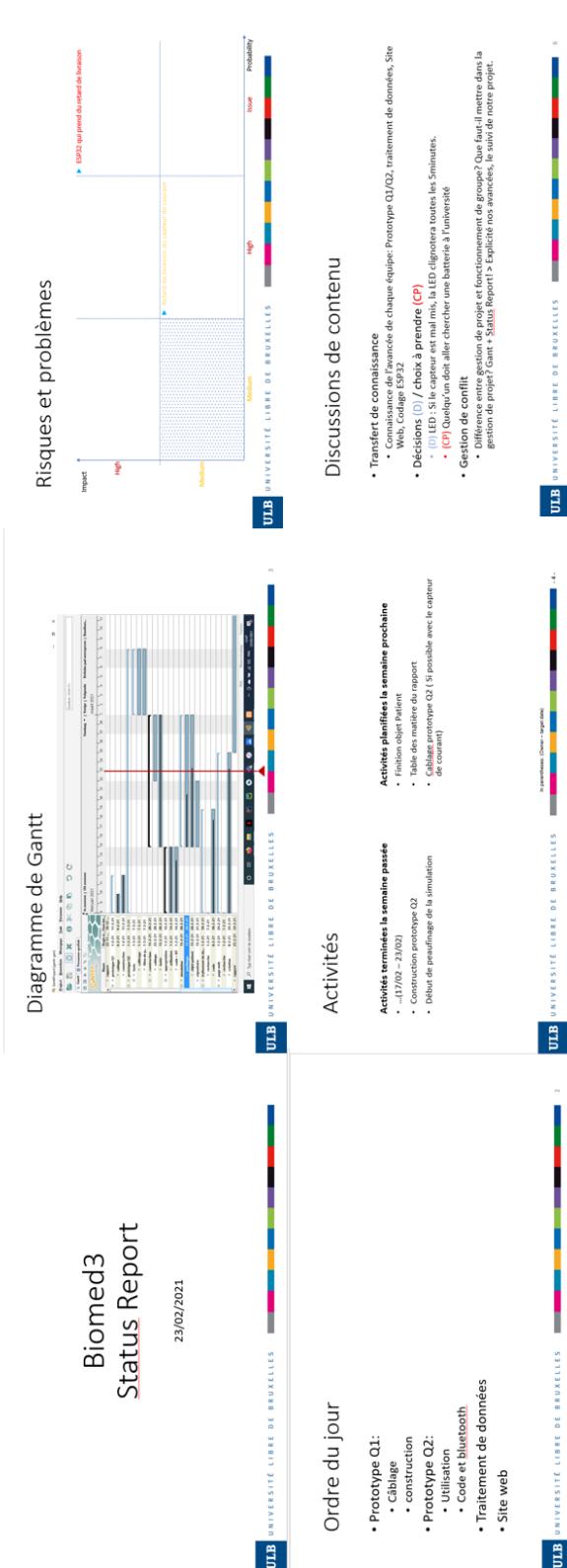


FIGURE A.2: Status report de la réunion du 23/02/21

Gestion des risques et des problèmes :
SEMAINE 8 (6NOV 2020)

Tâches :	Problèmes survenus pendant la semaine :	Risques qui peuvent survenir :
<i>Choix de l'interface</i>		<ul style="list-style-type: none"> - Mauvaise gestion du temps due aux lacunes du groupe - Erreur de synchronisation/problème d'affichage
<i>Codage de l'interface</i>	Problème de connexion internet	Problème de connexion internet
<i>InfluxDB</i>	Problème à se connecter au serveur Biomed3	<ul style="list-style-type: none"> - Complexité du code - Mauvaise gestion du temps due aux lacunes du groupe
<i>Traitement de Données</i>		Mauvaise gestion du temps due aux lacunes du groupe
<i>Prototype</i>		Possible problème de connexion à la base de données

FIGURE A.3: Tableau de la gestion des risques et problèmes : semaine 8, 1er quadrimestre

Issues												
Ref.	Domain	Issue name	Issue Description			Interdependancies	Raised by	Closed date	Status	Owner	Action / Solution	Comment
1	Prototype Q2	Le proto périodique à cramé	Le prototype périodique ne connecte plus en câble de la terre détacher de la batterie	Younes	#####				Closed		Un nouveau esp32 du même type (même dimension) a été recommandé	
2	Prototype Q1	La batterie n'est pas arrivé dans les temps voulus (et semble ne pas arriver pour l'évaluation finale)	Retard dû à la situation actuelle du Covid et au Nouvel an Chinois en même temps	Younes	#####				New		Une batterie de remplacement a été demandé aux organisateurs pour faire les tests.	
3	Prototype Q1	Tests d'évaluation	Le groupe n'a pas vu le message sur BIOMED décrivant les expériences à réaliser	High	Lise	20/03/2021	4/03/2021	Closed			Les tests ont déraré (d'activité des membres)	
4	Prototype Q2	Tests d'évaluation	Le groupe n'a pas vu le message sur BIOMED décrivant les expériences à réaliser	Mediu	Lise	2/03/2021	1/03/2021	Closed			Début du test	
5	Prototype Q1	Problème de batterie	Le capteur de courant a endommagé la batterie, qui a court-circuité	Mediu	Lise	2/03/2021	9/03/2021	Closed			Début du test	
6	Prototype Q2	Problème de batterie	Le capteur de courant a endommagé la batterie, qui a court-circuité.	High	Lise	4/03/2021	4/03/2021	Closed			Utilisation de la batterie du prototype continu	
7	Prototype Q1	Autonomie	On a inversé la terre et les 3V en connectant la nouvelle batterie sur l'esp, ce qui l'a court-circuité	High	Lise	4/03/2021	5/03/2021	Closed			Achut d'une autre batterie de 1500mAh	
8	Prototype Q2	Autonomie	La batterie tient 8h au lieu de 14h puis la batterie a tenu 8h au lieu de la 15-aine prévue	Mediu	Lise	5/03/2021	In Progress	In Progress			Cyril a reçu deux autres batteries de 500mAh	
9	Prototype Q1	Problème de courant	Les tests à effectuer pour l'évaluation finale sont compromis, pour le périodique et le continu	High	Lise	5/03/2021	5/03/2021	Closed			A priori il s'agit d'un bug de l'espace code, qui sont un risque, car les tests avec une batterie de 500 étaient concluants.	
Risks												
Ref.	Domain	Risk name	Risk Description			Interdependancies	Raised by	Closed date	Status	Owner	Mitigation action, if any	Comment
1	prototype Q1	câblage	risque au niveau des choix de fils à utiliser et de la couleur	Younes	17/02/2021	Closed				Extraction et accès à une pompe à désoudeage	Low	
2	site web	complexité du code	comporte le problème de la port	Lise	18/02/2021	Closed					High	
3	Prototype Q1	Batterie	retard de livraison	Younes	18/02/2021	Closed					High	
4	Traitemenr de données	differents cas	plusieurs cas à faire en fonction de si la personne est à risque	Dress	19/02/2021	Mitigated					Medium	
5	Prototype Q2	Cramage	Risque de cramer l'ESP32 (on l'a fait ou pas)	Younes	18/02/2021	Closed				Obtention d'un autre ESP32 (c'est en chemin)	Low	
6	Prototype Q1	court-circuit	certains points de souder sont assez proches les uns des autres et il tape utilisé pour les séparer	cyril	17/02/2021	Closed				recherche d'autres isolants	Medium	
7	Traitemenr de données	Envio d'email	Difficulté rencontrer pour l'envio des e-mails notamment au fait que c'est une nouveauté on n'a pas recommandé un esp qui n'est pas encore là	Dress	17/02/2021	Closed				Recherche de fonction simple à manier sur internet	High	
8	Prototype Q2	arrivée esp32	retarde le câblage pour le prototype Q2	Cyril	23/02/2021	Closed					-1	
9	Prototype Q2	capteur de courant	difficulté pour la simulation des sous-catégories du cas "patient malade"	Younes	23/02/2021	Closed				Input Likelihood & Impact	Medium	
10	Simulation	Réalisme	difficulté pour la simulation des sous-catégories du cas "patient malade"	Dress	23/02/2021	Closed				Augmentation des cas possibles pour les cas "patient malade"	-1	
11	Traitemenr de données	Synchronisation	les différentes fonctions ne s'effectuent pas en même temps causant un décalage des données traitées.	Dress	23/02/2021	Closed				Découpage en plusieurs scripts	High	
12	Prototype Q2	Autonomie de la batterie	colégation des batteries 500mAh ne permettent pas de faire une estimation, d'envisager la durée de vie d'une batterie 1500mAh	Lise	13/03/2021	Not Mitigated					Medium	
13	Prototype Q1	Bug dans un test	un bug de codification ou un court-circuitage de la batterie n'est pas à exécuter	Lise	12/03/2021	Closed				Les tests avec la batterie de 500mAh permettent de tirer des conclusions	Medium	
14	Prototype Q2	Bug dans un test	un bug de codification ou un court-circuitage de la batterie n'est pas à exécuter	Lise	12/03/2021	Not Mitigated				Les tests avec la batterie de 500mAh permettent de tirer des conclusions	Medium	
15	Traitemenr de données	Bug dans un test	le code fonctionne, mais il se tourne mal que des cas spécifiques non accessibles (non-pérmissibles) provoquent un bug tout à fait logique	Lise	15/03/2021	Not Mitigated					High	
16	Site web	Affichage	un bug dans l'affichage sera à problématique	Lise	16/03/2021	Closed				Les tests sont concluents donc à priori ça devrait aller	Medium	
17	Site web	Affichage	les données sont nombreuses, on pourrait reprocher un manque de clarté à la défense	Lise	16/03/2021	Open					Low	

FIGURE A.4: Excel des risques et problèmes survenus au 2eme quadrimestre



Site Web

```
<!DOCTYPE html>
<html>

<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">

<head>
<title>Biomed3</title>
<br>
<div class="w3-container w3-red">
<h1 style="text-align:center;">Biomed3</h1> </div>

<h2 style="text-align:center;"><ins>Accueil</ins></h2>

</head>

<body bgcolor="skyblue">
&nbsp; Bienvenu sur le site du groupe de projet Biomed3.
<br>
&nbsp; Vous utilisez actuellement le prototype de suivi de sant&eacute &agrave domicile pour le Covid19.
<br>
&nbsp; Nous esp&eacuterons que tout se passe au mieux, et que vous vous portez bien.
<br>
<br>
<br>

&nbsp; Vous trouverez ci-dessous les liens pour acceder au questionnaire - &agrave remplir quotidiennement! - ainsi que la page vers vos relev&eacutes.

<br>
&nbsp; site pour r&eacutepondre au questionnaire:
<br>

&nbsp; <a href = " http://localhost/question.html" > Questionnaire <a>
<br>

&nbsp; site pour consulter vos relev&eacutes:
<br>

&nbsp; <a href = "http://localhost/relev%C3%A9s.py." > Relev&eacutes <a>
<br>
&nbsp; page pour consulter vos notifications:
<br>

&nbsp; <a href = http://localhost/bioweb/notifications.py > Notifications <a>
<br>

&nbsp; Portez-vous bien !
</body>
</html>
```

FIGURE B.1: Code HTML pour la création de la page d'accueil

```

<!DOCTYPE html>
<html>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <head>
    <title>Questionnaire</title>
  </head>
  <body bgcolor="skyblue">
    <form action = "qmain.py" method="post">
      <p>
        <div class="w3-container w3-red">
          <h1 style="text-align:center;">Biomed3</h1> <br>
          <h2 style="text-align:center;"><ins>Questionnaire</ins></h2>
          <p style="float:right;"> <br>
            <h6 style="color:red;"> *Veuillez r&eacutepondre par :<br>
              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;- "1" si la r&eacuteponse est oui <br>
              &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;- "2" si la r&eacuteponse est non </h6>
            <br>
          </p>
          <p>
            &nbsp;&nbsp;<stan style="border: 1px solid black"><em>Ins&eacuteuter Nom/pr&eacuteénom?</em></stan> <br>
            &nbsp;&nbsp;&nbsp;&nbsp;<input type="text" placeholder="Nom/prenom" name="question1"/>
          </p>
          <p>
            &nbsp;&nbsp;&nbsp;&nbsp;<h4> - Avez-vous eu un des sympt&ocirc;mes suivant : </h4>
            &nbsp;&nbsp;&nbsp;<stan style="border: 1px solid black"><em>La toux?</em></stan> <br>
            &nbsp;&nbsp;&nbsp;&nbsp;<select name = "question2">
              <option value = "Ok">Choisissez la valeur</option>
              <option value = "1">1</option>
              <option value = "2">2</option>
            </select>
          </p>
          <p>
            &nbsp;&nbsp;&nbsp;<stan style="border: 1px solid black"><em>Une difficult&eacute; ; &agrave; respirer?</em></stan> <br>
            &nbsp;&nbsp;&nbsp;&nbsp;<select name = "question3">
              <option value = "Ok">Choisissez la valeur</option>
              <option value = "1">1</option>
              <option value = "2">2</option>
            </select>
          </p>
          <p>
            &nbsp;&nbsp;&nbsp;<stan style="border: 1px solid black"><em>Une douleur au niveau du thorax?</em></stan> <br>
            &nbsp;&nbsp;&nbsp;&nbsp;<select name = "question4">
              <option value = "Ok">Choisissez la valeur</option>
              <option value = "1">1</option>
              <option value = "2">2</option>
            </select>
          </p>
          <p>
            &nbsp;&nbsp;&nbsp;<stan style="border: 1px solid black"><em>Une perte de go&ucirc;t ou d'odorat?</em></stan> <br>
            &nbsp;&nbsp;&nbsp;&nbsp;<select name = "question5">
              <option value = "Ok">Choisissez la valeur</option>
              <option value = "1">1</option>
              <option value = "2">2</option>
            </select>
          </p>
          <p>
            &nbsp;&nbsp;<h4><em>-heure de d&eacute;position du questionnaire</em></h4> <br>
            &nbsp;&nbsp;&nbsp;&nbsp;<input type="date" id="start" name="trip-start">
            <input type="time" id="appt" name="appt">
          </p>
          <p>
            &nbsp;&nbsp;&nbsp;&nbsp;<input type="submit" value="Submit"/>
          </p>
        <p>Time: <span id="datetime"></span></p>
        <script>
          var dt = new Date();
          document.getElementById("datetime").innerHTML = dt.toLocaleTimeString();
        </script>
      </form>
    </body>
  </html>

```

FIGURE B.2: Code HTML pour la création du questionnaire

```
1 ► #!C:\Users\Gebruiker\AppData\Local\Microsoft\WindowsApps\python.exe
2 import os
3
4 print("content-type: text/html\n\n")
5 import cgi
6
7 # import du style pour bannière
8 print("""
9     <meta name="viewport" content="width=device-width, initial-scale=1">
10    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">""")
11
12 # début du head
13 print("<head>")
14
15 # titre de la page
16 print("""<title>
17 Notifications
18 </title>""")
19
20 # style
21 print("""<style>
22 table, th, td {
23     border: 1px solid black;
24 }
25 </style>""")
26
27 print("<br>")
28
29 # titres
30 print("""<div class="w3-container w3-red">
31 <h1 style="text-align:center;">Biomed3</h1> </div>""")
32
33 print("""<h2 style="text-align:center;"><ins>Notifications</ins></h2>""")
34
35 print("</head>")
36
37 #fond d'écran
38 print('<body bgcolor="skyblue">')
39 print("""
40 <br>
41 <br>
42 <br>""")
43
44 #texte de début
45
46 #notifications
47
48 with open('monitoring','r') as f:
49     a = f.read()
50     print(a)
51
```

FIGURE B.3: Code de la page affichant les notifications

```

1  #Import des modules
2  import ...
3
4
5  #influxdb settings
6  host='influxdb.biomed.ulb.ovh'
7  db='biomed3' #replace by your database
8
9  #influxdb credentials
10 username='biomed3' #replace by your database user login
11 password='Or1nqhDW5ynBvs4a' #replace by your database user password
12
13 #init the influxdb client
14 client = InfluxDBClient(host=host, port=80, username=username, password=password, database=db)
15
16 #test db connectivity (not mandatory) and if it is ok, it should return influxdb version
17 client.ping()
18
19
20 #définition de fonctions
21
22 def time_format(t):
23     word_list = []
24     for elem in t:
25         word_list.append(elem)
26
27     f = ""
28     for i in range(len(word_list)):
29         if i < 10:
30             f += word_list[i]
31         elif i == 10:
32             f += ","
33         elif i > 10 and i < 19:
34             f += word_list[i]
35
36     return f
37
38
39 def time_list_format(t):
40     for i in range(len(t)):
41         t[i] = time_format(t[i])
42     return t
43
44
45 def recuper_temp():
46
47     #récupération
48     query = 'SELECT temp FROM Experience4' # testfor ou testgrafana
49     results = client.query(query)
50     points = list(results.get_points(measurement='Experience4'))
51
52     # création d'une liste avec les mesures de température
53     temp = []
54     for i in range(len(points)):
55         temp.append(points[i]["temp"])
56
57     # création d'une liste avec les temps des mesures de température
58     time = []
59     for i in range(len(points)):
60         time.append(points[i]["time"])
61
62     time = time_list_format(time)
63
64     return temp, time
65

```

FIGURE B.4: Code relevant les données pour afficher dans un tableau (partie 1)

```

66     def recuperation():
67         # récupération
68         query = 'SELECT bpm FROM Experience4' # testfor ou testgrafana
69         results = client.query(query)
70         points = list(results.get_points(measurement='Experience4'))
71
72         # création d'une liste avec les mesures de température
73         bpm = []
74         for i in range(len(points)):
75             bpm.append(points[i]["bpm"])
76
77         # création d'une liste avec les temps des mesures de température
78         time = []
79         for i in range(len(points)):
80             time.append(points[i]["time"])
81
82         time = time_list_format(time)
83
84         return bpm, time
85
86     def recuperation():
87         #récupération
88         query = 'SELECT spo2 FROM Experience4' # testfor ou testgrafana
89         results = client.query(query)
90         points = list(results.get_points(measurement='Experience4'))
91
92         # création d'une liste avec les mesures de température
93         o2 = []
94         for i in range(len(points)):
95             o2.append(points[i]["spo2"])
96
97         # création d'une liste avec les temps des mesures de température
98         time = []
99         for i in range(len(points)):
100            time.append(points[i]["time"])
101
102         time = time_list_format(time)
103
104         return o2, time
105
106     def tableau_temp():
107         temp_time = recuperation()
108         temp = temp_time[0]
109         time = temp_time[1]
110
111         with open('temperature', 'w') as f:
112             for i in range(len(temp)):
113                 f.write('<tr>')
114                 f.write('\n')
115                 f.write('<td>')
116                 f.write(time[i])
117                 f.write('</td>')
118                 f.write('\n')
119                 f.write('<td>')
120                 f.write(str(temp[i]))
121                 f.write('</td>')
122                 f.write('\n')
123                 f.write('</tr>')
124
125

```

FIGURE B.5: Code relevant les données pour afficher dans un tableau (partie 2)

```

126     def tableau_bpm():
127         bpm_time = recuper_bpm()
128         bpm = bpm_time[0]
129         time = bpm_time[1]
130         with open('bpm', 'w') as f:
131             for i in range(len(bpm)):
132                 f.write('<tr>')
133                 f.write('\n')
134                 f.write('<td>')
135                 f.write(time[i])
136                 f.write('</td>')
137                 f.write('\n')
138                 f.write('<td>')
139                 f.write(str(bpm[i]))
140                 f.write('</td>')
141                 f.write('\n')
142                 f.write('</tr>')
143                 f.write('\n')
144
145     def tableau_o2():
146         o2_time = recuper_o2()
147         o2 = o2_time[0]
148         time = o2_time[1]
149         with open('oxygène', 'w') as f:
150             for i in range(len(o2)):
151                 f.write('<tr>')
152                 f.write('\n')
153                 f.write('<td>')
154                 f.write(time[i])
155                 f.write('</td>')
156                 f.write('\n')
157                 f.write('<td>')
158                 f.write(str(o2[i]))
159                 f.write('</td>')
160                 f.write('\n')
161                 f.write('</tr>')
162                 f.write('\n')
163
164     def main():
165         i = 1
166         while i == 1:
167             tableau_temp()
168             tableau_bpm()
169             tableau_o2()
170             sleep(10)
171
172     #Code principal
173     main()

```

FIGURE B.6: Code relevant les données pour afficher dans un tableau (partie 3)

```

1 ► #!C:\Users\Gebruiker\AppData\Local\Microsoft\WindowsApps\python.exe
2 import os
3
4 print("content-type: text/html\n\n")
5 import cgi
6
7 #import du style pour bannière
8 print("""
9     <meta name="viewport" content="width=device-width, initial-scale=1">
10    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">""")
11
12 #début du head
13 print("<head>")
14
15 #titre de la page
16 print("""<title>
17 Relev&eacutees
18 </title>""")
19
20 #style
21 print("""<style>
22 table, th, td {
23     border: 1px solid black;
24 }
25 </style>""")
26
27 print("<br>")
28
29 #titres
30 print("""<div class="w3-container w3-red">
31 <h1 style="text-align:center;">Biomed3</h1> </div>
32
33 <h2 style="text-align:center;"><ins>Relev&eacutees de donn&eacute;es m&eacute;dicales</ins></h2>""")
34
35 print("""Bienvenu sur votre page personnelle reprenant toutes vos informations <br>
36 concernant l'&eacute;volution de votre sant&eacute;""")
37 print("</head>")
38
39 #fond d'écran
40 print('<body bgcolor="skyblue">')
41 print("""
42 <br>
43 <br>
44 <br>""")
45
46 #prototype périodique
47 print("""Le dispositif p&eacute;riodique rel&egraveve des donn&eacute;es de temp&eacute;ature et de rythme cardiaque""")
48 print("<br>")
49
50 #tableau de température
51 print("""Voici les r&eacute;sultats pour la temp&eacute;ature
52     <br>""")
53
54 print('<p style="float:right"> </p>')
55
56
57 print("""<table style="width:50%">
58     <tr>
59         <th>Date et heure</th>
60         <th> température (°C) </th>
61     </tr>""")
62
63
64 with open('temperature','r') as f:
65     a = f.read()
66     print(a)
67
68 print('</table>')
69
70 print("""<br>
71 <br>
72 <br>""")
73
74

```

FIGURE B.7: Code affichant les tableaux de mesures (partie 1)

```
76 #tableau du rythme cardiaque
77 print("""Voici les r&eacute;sultats pour le rythme cardiaque
78 <br>""")
79
80 print("""<table style="width:50%">
81 <tr>
82 <th>Date et heure</th>
83 <th> température (°C) </th>
84 </tr>""")
85
86 with open('bpm','r') as f:
87     a = f.read()
88     print(a)
89
90 print('</table>')
91
92 print("""<br>
93 <br>
94 <br>""")
95 <br>"""
96
97
98 #prototype continu
99 print("""Le dispositif continu rel&egraveve &eacute;galement le taux d'oxyg&egravene dans le sang""")
100
101 #tableau de l'oxygène
102 print("""Voici les r&eacute;sultats pour l'oxygène
103 <br>""")
104 print('<p style="float:right"> </p>')
105
106 print("""<table style="width:50%">
107 <tr>
108 <th>Date et heure</th>
109 <th>oxygène (%) </th>
110 </tr>""")
111
112 with open('oxygène','r') as f:
113     a = f.read()
114     print(a)
115
116 print('</table>')
```

FIGURE B.8: Code affichant les tableaux de mesures (partie 2)

Prototype périodique

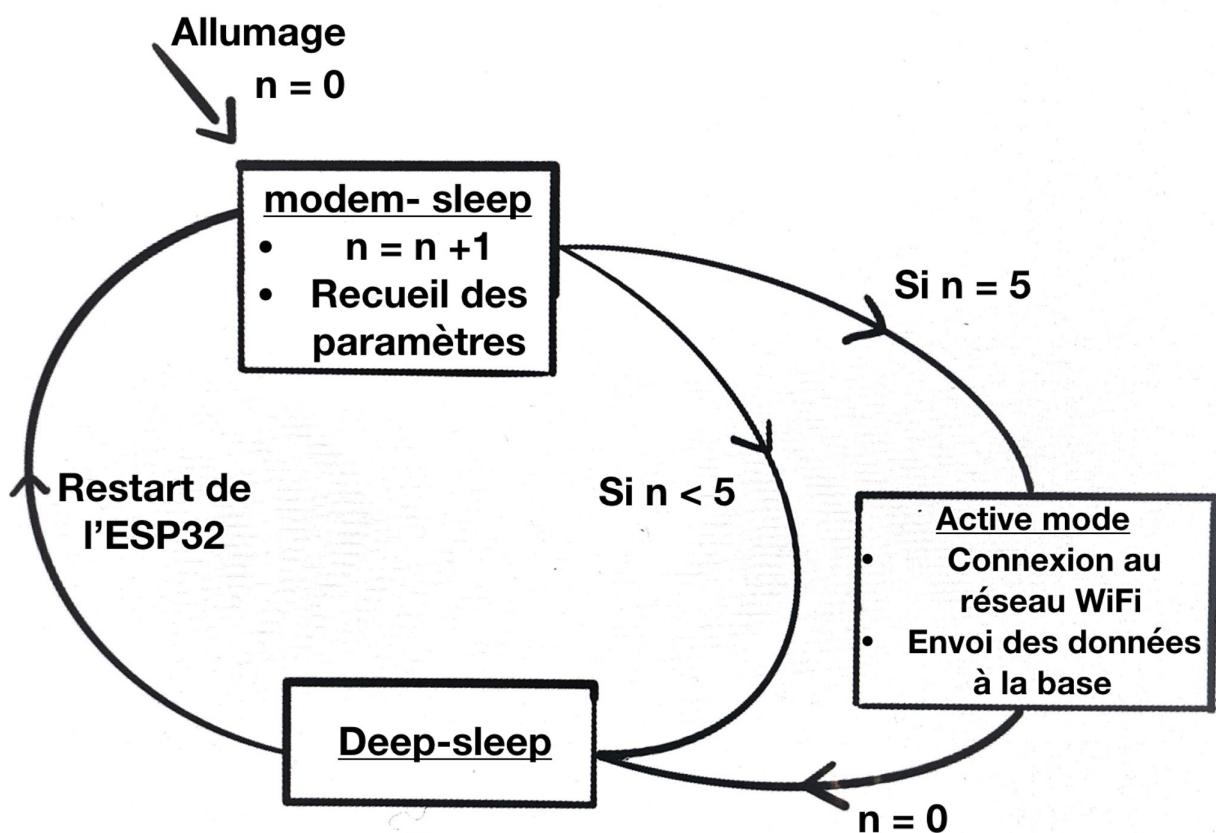


FIGURE C.1: schéma de la boucle du prototype périodique faite au premier quadrimestre

D

Manuel d'utilisation

GUIDE D'INSTRUCTION

Suivi de santé à domicile

Covid



BIOMED3

Moyen de contact :

Biomed3ulb@gmail.com

FIGURE D.1: Manuel d'utilisation : page de garde

TABLE DES MATIERES

I. INTRODUCTION..... 1

II. CONNEXION AU WIFI 2

III. MISE EN PLACE DU DISPOSITIF ... 4

IV. HYGIENE 6

nettoyage 6

désinfection 7

V. RECHARGE DE LA BATTERIE..... 8

FIGURE D.2: Manuel d'utilisation : table des matières

I. INTRODUCTION

Ce dispositif est un suivi de santé à domicile spécialement conçu dans le cadre de la crise sanitaire de la Covid-19.

Si vous recevez ce guide, et les dispositifs qui vont avec, c'est que vous avez été en contact et/ou que vous êtes atteints du SARS-COV-2.

Ceux-ci permettront de suivre l'apparition et l'évolution de différents symptômes caractéristiques de la maladie. Ces relevés seront disponibles en temps quasi-réel sur une page-web, à laquelle vous accéderez via un nom d'utilisateur et un mot de passe fourni par votre médecin traitant ou une équipe médicale. Celui-ci est disponible au lien suivant :

<http://localhost/biomed3.html>

Vous porterez d'abord un premier dispositif, qui mesurera votre température et votre rythme cardiaque.

Si jamais votre état se dégrade, vous porterez le second prototype, qui mesurera également votre taux d'oxygène dans le sang.

Dans les deux cas, si des mesures inquiétantes apparaissent, vous serez averti par mail.



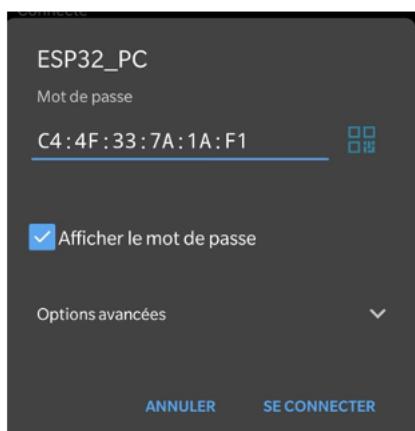
Attention : si vous êtes en possession de ces dispositifs, vous avez été en contact avec une personne positive. Vous êtes donc en quarantaine, et par conséquent, toute sortie est interdite.

FIGURE D.3: Manuel d'utilisation : page 1

II. CONNEXION AU WIFI

Afin de pouvoir envoyer les données récoltées par le dispositif pour analyse de vos symptômes, celui-ci doit être connecté au wifi.

Dans ce but, recherchez d'abord le wifi émis par le dispositif.



Veuillez-vous y connecter en entrant comme mot de passe la MacAdress de l'ESP, fournit avec le dispositif.

2

FIGURE D.4: Manuel d'utilisation : page 2

Une fois cette connexion établie, tentez de vous connectez au réseau de l'ESP32. Une page s'affichera alors, où vous pourrez cliquer sur « Configure Wifi ».

Enfin, vous pourrez choisir votre propre réseau wifi, et entrer votre mot de passe. Une fois cette étape finie, votre dispositif devrait être connecté.

3

Pensez à vérifier la qualité de votre réseau : l'analyse de vos symptômes en dépend.

FIGURE D.5: Manuel d'utilisation : page 3

III. MISE EN PLACE DU DISPOSITIF



Le dispositif est constitué d'une boîte – qui renferme les éléments électroniques -, des capteurs et d'une sangle.

Refermez la sangle, et placez les capteurs du côté intérieur, par la fente, de façon à ce qu'ils soient contre la peau quand vous le passerez au bras.

Veillez à ce que le tout ne soit pas trop serré au départ, sinon les capteurs seront écrasés.



4

FIGURE D.6: Manuel d'utilisation : page 4



Passez la sangle autour de votre poignet et remontez le boîtier jusqu'à la bonne position.

Le boîtier devrait se situer un peu au-dessus du coude.

Ajustez ensuite la sangle de façon à ce que le dispositif soit stable.



Attention à bien placer les capteurs contre la peau ! Ceci est impératif pour permettre la bonne transmission et analyse des données !

Il convient de faire passer les capteurs au travers de la fente.



5

FIGURE D.7: Manuel d'utilisation : page 5

Dans le cas où vous devriez utiliser le second dispositif, vous devrez également porter un oxymètre. Celui-ci mesure votre taux d'oxygène.



Il convient de le placer sur votre doigt, comme le montre l'image.

IV. HYGIENE

NETTOYAGE

Pour éliminer les tâches apparentes de votre boîtier, une simple loque humide suffit.

Idéalement, il convient de nettoyer le dispositif régulièrement afin d'enlever tout résidu, qui provoquerait un sentiment désagréable. Le brassard peut être passé à la machine à laver si besoin



Attention : le boîtier comporte des interstices. Il est donc important de ne pas trop humidifier votre loque, afin de ne pas endommager les circuits intérieurs.

6

FIGURE D.8: Manuel d'utilisation : page 6

DESINFECTION

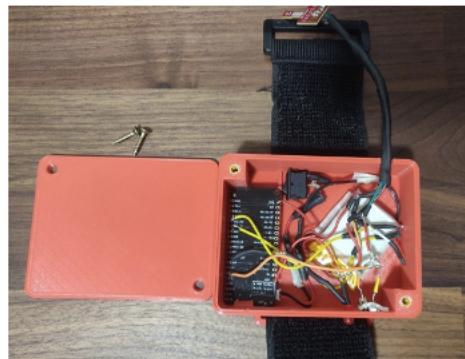
La désinfection peut se faire avec de l'eau de Javel ou un désinfectant compose d'alcool.

Afin d'éviter tout endommagement dû au liquide, il convient de d'abord retirer le dispositif interne.



Pour cela, dévisser d'abord le couvercle de la boîte

Veuillez ensuite retirer le dispositif intérieur



Attention : à priori, ceci ne concerne que les équipes médicales. Le boîtier ne doit être désinfecté que lorsqu'il est donné à un autre patient

7

FIGURE D.9: Manuel d'utilisation : page 7

V. RECHARGE DE LA BATTERIE

La recharge de la batterie se fait à l'aide d'un câble micro-USB, qui vous est fourni avec le dispositif.

Insérez votre câble en bas à gauche de la lampe LED.



Bibliographie

- [App19] APPRENDRE-GESTION (2019). « Cours sur l'analyse Swot ». In : URL : <https://apprendre-gestion.com/cours-analyse-swot/>.
Explication et meilleure compréhension de l'utilité d'une analyse Swot
.
- [Ard19] ARDUINO-FRANCE (2019). « Qu'est-ce que le bus I2C et comment fonctionne-t-il sur Arduino ? » In : URL : <https://www.arduino-france.com/tutoriels/qu'est-ce-que-le-bus-i2c-et-comment-fonctionne-t-il-sur-arduino/>.
Tutoriel expliquant ce que sont les bus I2C et plus spécifiquement leur utilisation avec Arduino.
- [Aud21] AUDREY (2021). *Qu'est-ce que l'oxymétrie?* URL : <https://www.girodmedical.com/blog/qu-est-ce-que-loxymetrie/>.
Explication de l'utilité de l'oxymètre.
- [Aut20a] AUTORITÉ-DE-PROTECTION-DES-DONNÉES (2020a).
Règlement général sur la protection des données (RGPD) - INAMI. URL : <https://inami.fgov.be/fr/professionnels/information-tous/Pages/reglement-general-protection-donnees.aspx>.
Les informations sont données sur le site officiel des autorités et donne tout ce qui est nécessaire pour la RGPD.
- [Aut20b] — (2020b). *Covid-19 | Autorité de protection des données.* URL : <https://www.autoriteprotectiondonnees.be/citoyen/themes/covid-19>.
Donne des informations concernant la RGPD dans le cas de la situation pandémique

actuelle.

- [Bla20] BLAIZE, Aurélie (2020). *Comment prendre sa température : sous le bras, en frontal, rectal?* URL : <https://sante.journaldesfemmes.fr/fiches-sante-du-quotidien/2647123-comment-prendre-sa-temperature-sous-le-bras-thermometre-frontal-rectal-aisselle/>.
Donne des informations sur les endroits où la prise de température est favorisée.
- [Clo20] CLOSED CUBE (2020). *ClosedCubeSTS35Arduino*. URL : https://github.com/closedcube/ClosedCube_STS35_Arduino.
Recueil d'exemples de code très utiles pour le capteur de température et informations complémentaires en cas de problèmes.
- [Coo18] COOKIE-CONNECTÉ (2018). *Comprendre le chiffrement SSL / TLS avec des emojis (et le HTTPS)*. URL : <https://www.youtube.com/watch?v=7W7WPMX7arI>.
Bonne représentation graphique du cryptage en SSL.
- [Dig] DIGICERT (p. d.). « Qu'est-ce qu'un certificat SSL? » In : *Site Web* (). URL : <https://www.digicert.com/fr/ssl-certificate/>.
C'est un fournisseur de certificat numérique. Il est expliqué ce qu'est le SSL.
- [Erca] ERCEK, O. Debeir R. (p. d.[a]). *Enoncé Projet BA2 Biomed 2020-2021*. Sous la dir. de École Polytechnique de BRUXELLES. URL : https://uv.ulb.ac.be/pluginfile.php/2104891/mod_resource/content/2/Enonc%C3%A9%20Projet%5C%20BA2%5C%20Biomed%5C%202020-2021.pdf.
Enoncé du projet Biomed de BA2.
- [Ercb] — (p. d.[b]). *QuickStarts pour la réalisation du projet*. Sous la dir. de École Polytechnique de BRUXELLES. URL : <https://uv.ulb.ac.be/mod/resource/view.php?id=675867>.
Quickstart des responsables du projet.
- [Erc20] ERCEK, Rudy (2020). *BLEOximeter*. URL : <https://gitlab.biomed.ulb.ovh/rercek/bleoximeter>.

Recueil d'exemples de code directement utilisés pour l'oxymètre.

- [ESP] ESPRESSIF (p. d.). *ESP-TLS*. URL : https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_tls.html?fbclid=IwAR2A3AbaBPvzy3VpbV4aR4IzeYmqi3h-vw8TFzdyTfMA-jLWEQD9d_UC5FI. Exemple de mémoire que prends le SSL et le TLS dans l'ESP32.
- [ESP20a] ESPRESSIF-SYSTEMS (2020a). *ESP32 Series, Datasheet*. URL : https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Fiche technique de l'ESP32 renseignant le nombre de pins et la consommation du microcontrôleur.
- [ESP20b] — (2020b). *ESP32 Technical Reference Manual*. URL : https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf#page=627&zoom=100,76,76. Fiche technique de l'ESP32 informant sur les protocoles de communication des ports pour les capteurs.
- [Gra] GRAFANA (2020). *Grafana documentation*. Grafana Labs. URL : <https://grafana.com/docs/grafana/latest/> (visité le 10/12/2020). Site officiel de Grafana reprenant toutes les informations utiles concernant le paramétrage de l'interface.
- [Hen21] HENRY, Apolline (2021). « Symptômes Covid-19 : variant anglais, durée, que faire ? » In : URL : <https://www.topsante.com/medecine/maladies-infectieuses/zoonoses/covid-symptomes-variant-anglais-duree-que-faire-634746>. Article de Top Santé relayant les différences et les similitudes entre les symptômes de la Covid19 et ceux du variant anglais.
- [Inf20a] INFLUXDATA (2020a). *Explore data using InfluxQL*. URL : https://docs.influxdata.com/influxdb/v1.8/query_language/explore-data/. Ce document, tiré du site officiel d'InfluxDB, donne différents queries utiles au traitement de données.

- [Inf20b] INFLUXDATA (2020b). *InfluxDB key concepts*. URL : https://docs.influxdata.com/influxdb/v1.8/concepts/key_concepts/.
Explication des concepts clefs d’InfluxDB par le site officiel.
- [Inf20c] — (2020c). *InfluxDB line protocol tutorial*. URL : https://docs.influxdata.com/influxdb/v1.8/write_protocols/line_protocol_tutorial/.
Cette page, tirée du site officiel d’InfluxDB, explique comment insérer des relevés dans la base de données.
- [Inf20d] — (2020d). *Query Language : Sample data*. URL : https://docs.influxdata.com/influxdb/v1.8/query_language/sample-data/.
Ce document, tiré du site officiel d’InfluxDB, donne différents queries utiles à la gestion de bases de données.
- [Int18] INTEGRATED, Maxim (2018). *MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health*. URL : <https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf> (visité le).
Document du constructeur qui permet de connaître la quantité de courant dont le capteur à besoin en fonction de son activité .
- [ION19] IONOS (2019). *Tutoriel XAMPP : Installation et premiers pas*. URL : <https://www.ionos.fr/digitalguide/serveur/outils/tutoriel-xampp-creer-un-serveur-de-test-local/>.
Aide à la compréhension de XAMPP.
- [Jai18] JAIN, Abhishek (2018). *InfluxDB Tutorial - InfluxDB Query Structure, Measurement, Tag key-value, Field Key-value - Part2*. URL : <https://www.youtube.com/watch?v=nu8XEcsG2L8&t=35s/>.
Cette vidéo illustre le fonctionnement de base d’une console InfluxDB et permet de comprendre le fonctionnement des queries.
- [Jak16] JAKOBOWICZ, Emmanuel (2016). *Les étapes pour utiliser Python en data science*. URL : <https://www.stat4decision.com/fr/etapes-utiliser-python-en-data-science/>.

Cette page fournit des avantages à l'utilisation de Python pour le traitement de données.

- [Jin+20] JIN, Xi et al. (2020). « Epidemiological, clinical and virological characteristics of 74 cases of coronavirus-infected disease 2019 (COVID-19) with gastrointestinal symptoms ». In : *Gut*, p. 1002-1009. URL : <https://gut.bmjjournals.com/content/69/6/1002.full>.
Une étude sur différents patients atteints du Covid-19 dans le but de mesurer la fréquence d'apparition de différents symptômes.
- [Le-] LE-JOURNAL-DES-FEMMES (2021). *Taux de saturation en oxygène o2 : définition, mesure et normes*. URL : <https://sante.journaldesfemmes.fr/fiches-anatomie-et-examens/2526200-taux-de-saturation-o2-definition-mesure-normes/> (visité le 12/03/2021).
Cette page fournit des informations sur le taux d'spo2 normal d'une personne .
- [Kol17] KOLBAN (2017). URL : <https://www.esp32.com/viewtopic.php?t=858>.
Forum où le problème de stockage du SSL dans l'ESP32 a été évoqué.
- [La-ND] LA-LANGUE-FRANÇAISE (N.D.). « Microcontrôleur : définition de microcontrôleur ». In : URL : <https://www.lalanguefrancaise.com/dictionnaire/definition/microcontroleur>. Article expliquant les composants et les fonctionnalités de base d'un microcontrôleur.
- [Lar+20] LARSEN, Martin et al. (2020). « Modeling the Onset of Symptoms of COVID-19 ». In : *Frontiers in Public Health* 8, p. 473. URL : <https://www.frontiersin.org/article/10.3389/fpubh.2020.00473>.
Publication sur l'ordre d'apparition des symptômes de la Covid19.
- [Las18] LAST-MINUTE-ENGINEERS (2018). *ESP32 Deep Sleep Its Wake-up Sources*. URL : <https://lastminuteengineers.com/esp32-deep-sleep-wakeup-sources/>.
Enumération des différents composés actifs en fonction du sleep mode.
- [Maxa] MAXIM-INTEGRATED (p. d.[a]). *Datasheet du capteur cardiaque MAX30102*. URL : <https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf>.

Données techniques du capteur MAX30102.

- [Maxb] MAXIM-INTEGRATED (p. d.[b]). *Datasheet du capteur cardiaque MAX86150*. URL : <https://datasheets.maximintegrated.com/en/ds/MAX86150.pdf>.

Données techniques du capteur MAX86150.

- [Maxc] — (p. d.[c]). *MAX30102*. URL : <https://makersportal.com/shop/max30102-heart-rate-and-pulse-oximeter-sensor>.

Source de l'image du capteur cardiaque MAX30102.

- [MCU21] MCUCITY (2021). « ESP32 Lolin32 Lite ». In : URL : <https://www.mcucity.com/product/3190/esp32-lolin32-lite>.

Article fourni dans les Quickstarts qui permet de visualiser les différentes entrées et sorties du microcontrôleur.

- [Meh21] MEHR, Martial (2021). *Covid-19 / variant anglais - Les symptômes sont-ils différents entre les deux types de virus?* URL : <https://www.lindependant.fr/2021/02/18/covid-19-variant-anglais-les-symptomes-sont-ils-differents-entre-les-deux-types-de-virus-9380274.php>.

Article de L'Indépendant relayant les différences et les similitudes entre les symptômes de la Covid19 et ceux du variant anglais.

- [Mik] MIKROE (p. d.). *Datasheet du capteur cardiaque MIKROE-2000 (MAX30100)*. URL : <https://download.mikroe.com/documents/datasheets/MAX30100%5C%20DS%5C%20.pdf>.

Données techniques du capteur MIKROE-2000.

- [myw19] MYWANG-ESPRESSIF (2019). *Feature : Bluetooth modem sleep with external 32.768kHz xtal under light sleep*. URL : <https://github.com/espressif/esp-idf/issues/947?fbclid=IwAR2DVLqx0Tkhc7cd9g6CxukTsFm3yVMJ-1H8Zb4YP6ZHCRVINrvG0zM0qyM1>. Blog internet discutant du courant nécessaire lorsque le mode bluetooth de l'ESP32 est activé.

- [Nor20] NORD-NETTOYAGE (2020). *Coronavirus (covid 19) : comment bien désinfecter les surfaces? Nos conseils.* URL : <https://www.nord-nettoyage.com/desinfection-surfaces-coronavirus-covid19/>.
Le site est accès sur le nettoyage.
- [NXP14] NXP (2014). *I²C-bus specification and user manual.* URL : <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>.
Manuel d'utilisation des bus I²C.
- [Oct20] OCTOPART (2020). *Datasheets, Electronic Parts, Components, Search.* URL : <https://octopart.com/>.
Site comparatif pour composants électroniques, utilisé pour la recherche des capteurs.
- [OMS] OMS (2020). *Coronavirus disease (COVID-19) – World Health Organization.* URL : <https://www.who.int/emergencies/diseases/novel-coronavirus-2019> (visité le 08/12/2020).
Site officiel de l'OMS qui donne les dernières actualités sur l'évolution du COVID-19 dans le monde et donne des informations pratiques sur la maladie.
- [PK20] PARMANTIER, Yves et Frédéric KRATZ (2020). « Capteurs ». In : URL : <https://www.techniques-ingenieur.fr/base-documentaire/genie-industriel-th6/capteurs-42678210/capteurs-r400/>.
Définition du capteur.
- [PCM] PCMAG (p. d.). *Definition of bus.* URL : <https://www.pcmag.com/encyclopedia/term/bus>.
Définition du terme 'bus' en informatique par le magazine PCMag.
- [Per19] PERMAL (2019). *mbedtls memory usage per connection / mbedtls_ssl_context.* URL : <https://www.esp32.com/viewtopic.php?t=10612>.
Forum où le problème de stockage du SSL dans l'ESP32 a été évoqué.
- [Pro] PROACTIV (2021). *Quel est le rythme cardiaque normal d'un adulte?* ProActiv BE. URL : <https://www.pro-activ.com/fr-be/hart-en-cholesterol/hartgezondheid/>

wat-is-een-normale-hartslag-voor-volwassenen (visité le 10/03/2021).

Cette page fournit des informations sur la fréquence cardiaque d'une personne.

- [Pul] PULSE-OXIMETER (p. d.). *OX831*. URL : <https://images.izi.ua/28634368>.
Source de l'image de l'oxymètre OX-831.

- [RAD] RADIO-CANADA (p. d.). *Comment entretenir ses vêtements durant la pandémie?* URL : <https://ici.radio-canada.ca/nouvelle/1696829/questions-reponse-manitoba-tissu-covid-19-lavage-lessive-chimie>.
Permet de savoir comment désinfecter le brassard.

- [RD12] RASIDY, Quentin et Lionel DEBUYS (2011-2012). « Le cryptage ». In : *Presses universitaires*.
Explication générale sur le cryptage hybride.

- [Ref21] REFSNES-DATA (2021). *HTML Basic*. URL : https://www.w3schools.com/html/html_basic.asp.
Cette page, et les liens vers lesquels elle renvoie, permettent d'apprendre les bases de l'HTML rapidement : toute l'information nécessaire y est concentrée.

- [Rud20] RUDY, Ercek (2020). *Projects · Rudy Ercek / InfluxDBNoteBookTest*. GitLab. URL : <http://gitlab.biomed.ulb.ovh/rercek/influxdbnotebooktest> (visité le 10/12/2020).
comprendre comment convertir les résultats de type resultset en liste
.

- [SS20] SANTOS, Rui et Sara SANTOS (2020). « Ultimate Guide for Arduino Sensors and Modules ». In : p. 164. URL : http://tecnologix.altervista.org/wp-content/files/Ultimate_Guide_Arduino_Sensors_Modules.pdf.
Guide des capteurs les plus connus sur Arduino.

- [San18] SANTOS, Sara (2018). *Getting Date and Time with ESP32 on Arduino IDE (NTP Client)*. URL : <https://randomnerdtutorials.com/esp32-ntp-client-date-time-arduino-ide/>.
Explication et exemples d'utilisation du timestamp.

- [San19] SANTOS, Sara (2019). *ESP32 Deep Sleep with Arduino IDE and Wake Up Sources*. URL : <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>.
Indication des différents sleep modes qui peuvent être utilisés avec l'ESP32 et explication de la méthode pour sauvegarder des données après un restart.
- [San20] — (2020). *Get ESP32/ESP8266 MAC Address and Change It (Arduino IDE)*. URL : <https://randomnerdtutorials.com/get-change-esp32-esp8266-mac-address-arduino/>.
Explication de ce qu'est la MacAdress et comment la récupérer.
- [Sch19] SCHNEIDER, Ramon (17-05-2019). « Comment bien choisir une batterie LiPo? » In : URL : <https://www.galaxus.ch/fr/page/comment-bien-choisir-une-batterie-lipo-11999>.
La page web explique en quoi consiste la mesure en Ampère-heure .
- [Sch20] SCHÜRG, Tobias (2020). *InfluxDB-Client-for-Arduino*. URL : <https://github.com/tobiasschuerg/InfluxDB-Client-for-Arduino>.
Recueil d'exemples de code très utiles pour la base de données et informations complémentaires en cas de problèmes.
- [SEN19] SENSIRION (2019). *Datasheet STS3x-DIS*. URL : https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/3_Temperature_Sensors/Datasheets/Sensirion_Temperature_Sensors_STS3x_Datasheet.pdf.
Document du constructeur qui permet de connaître la quantité de courant dont le capteur à besoin en fonction de son activité.
- [Sena] SENSIRION (p. d.[a]). *Datasheet du capteur de température STS35*. URL : https://www.glynshop.com/erp/owweb/Daten/Datenblaetter/Sensirion/STS3x_DIS.pdf.
Données techniques du capteur STS35.
- [Senb] — (p. d.[b]). *STS35*. URL : https://images-na.ssl-images-amazon.com/images/I/91qsL4yNugL._SL1500_.jpg.

Source de l'image du capteur de température STS35.

- [Sil] SILABS (p. d.). *Datasheet du capteur de température Si7051*. URL : <https://www.silabs.com/documents/public/data-sheets/Si7050-1-3-4-5-A20.pdf>.
Données techniques du capteur Si7051
.
- [Spa] SPARKFUN (p. d.). *Datasheet du capteur cardiaque SEN-15219 (MAX30101 MAX32664)*. URL : <https://cdn.sparkfun.com/assets/4/3/c/2/b/MAX32664.pdf>.
Données techniques du capteur SEN-15219
.
- [Texa] TEXAS-INSTRUMENTS (p. d.[a]). *Datasheet du capteur de courant (et de tension) INA3221*. URL : https://www.ti.com/lit/ds/symlink/ina3221.pdf?ts=161599233336&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FINA3221.
Données techniques du capteur INA3221.
- [Texb] — (p. d.[b]). *Datasheet du capteur de courant (et de tension) INA219*. URL : https://www.ti.com/lit/ds/symlink/ina219.pdf?ts=1616066460903&ref_url=https%253A%252F%252Fwww.google.com%252F.
Données techniques du capteur INA219
.
- [Texc] — (p. d.[c]). *Datasheet du capteur de température TMP117*. URL : https://www.ti.com/lit/ds/symlink/tmp117.pdf?ts=1615968105797&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTMP117%252F
Données techniques du capteur TMP117.
- [Texd] — (p. d.[d]). *INA3221*. URL : <https://esphome.io/components/sensor/ina3221.html>.
Source de l'image du capteur de courant INA3221.
- [tim] TIMGATES42 (p. d.). *influxdata/influxdb-python*. URL : <https://github.com/influxdata/influxdb-python>.

Comprendre comment utiliser la classe influxdbclient

.

- [Top19] TOPSANTE.COM (20 fév. 2019). *Fièvre : à partir de quelle température s'inquiéter - Top Santé*. Section : médecine. URL : <https://www.topsante.com/medecine/troubles-orl/grippe/fievre-quand-faut-il-s-inquieter-630698> (visité le 10/03/2021). Cette page fournit des informations sur la température d'une personne.
- [Tou07] TOUBOUL, Jonathan (2007). « Nombres premiers et cryptologie : l'algorithme RSA ». In : *interstices*. URL : <https://interstices.info/nombres-premiers-et-cryptologie-lalgorithme-rsa/>.
Approche très théorique du chiffrement RSA utile à la compréhension.
- [Tza15] TZAPU (2015). *WiFiManager*. URL : <https://github.com/tzapu/WiFiManager>.
Recueil d'exemples de code très utiles pour le WiFiManager et informations complémentaires en cas de problèmes.
- [Le-20] LE-VIF (2020). « Coronavirus : quels sont les symptômes à surveiller ? » In : *Site-LeVif-FR*. URL : <https://www.levif.be/actualite/sante/coronavirus-quels-sont-les-symptomes-a-surveiller/article-news-1343677.html>.
Une liste des différents symptômes du Covid-19 écrite par le journal LeVif apportant de nouvelles informations.
- [www20] WWW.BELGIUM.BE (2020). *Home | Coronavirus COVID-19*. URL : <https://www.info-coronavirus.be/fr/>.
Une liste des différents symptômes du Covid-19 écrite par le gouvernement.