الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة ابي بكر بلقايد - تلمسان
Université Aboubakr Belkaïd-Tlemcen
كلية التكنولوجيا
Faculté de TECHNOLOGIE

Mémoire de fin d'études

Pour l'obtention du **diplôme** de **MASTER**

Filière : **Électronique**

Spécialité : Instrumentation

Réalisé par :
M. Belbachir Mohammed Anes
M. Ammour Oussama

Intitulé du Sujet

Etude et réalisation d'un système de gestion de salle de dilatation pour une clinique d'ophtalmologie.

Soutenu le 29 Juin 2022, Devant le jury composé de :

Mme.Bouazza née Guen Ahlem	-Professeur	-Univ de Tlemcen	-Présidente
M.Massoum Noureddine	-MCB	-Univ de Tlemcen	-Examinateur
M.Brixi Nigassa M.E.A	-MCB	-Univ de Tlemcen	-Encadrant
M.Slami Ahmed	-Doctorant	-Univ de Tlemcen	-Co-Encadrant
M.Lazouni Mohammed El Amine	-MCA	-Univ de Tlemcen	-Invité

Année Universitaire : 2021/2022

Dédicace

“

À l'homme, mon précieux offre du dieu, qui doit ma vie, ma réussite et tout mon respect : mon cher père ***Omar****,*

À la femme qui a souffert sans me laisser souffrir, qui n'a jamais dit non-âmes exigences et qui n'a épargné aucun effort pour me rendre heureux : mon adorable mère ***Hayet****,*

À mon frére ***Mouad*** *et mes chères sœurs* ***Chaimaa*** *et* ***Alaa*** *qui n'ont pas cessée de me encourager et soutenir tout au long de mes études. Que Dieu les protège et leurs offre la chance et le bonheur,*

À mon cher oncle ***Bouziane*** *et sa femme* ***Hbiba*** *qui je considérais comme ma deuxième mère, tous les moments, en gage de ma profonde estime pour l'aide que vous m'avez apportée. Vous m'avez soutenu, réconfortés et encouragés,*

À mon cher ami ***Hichem Rouigueb****, tu as toujours offert soutien et réconfort, j'exprime envers toi une profonde admiration, reconnaissance et attachement inconditionnels,*

Sans oublier mon cher binôme ***Oussama*** *pour son soutien moral, sa patience et sa compréhension tout au long de ce projet et toutes nos années d'études,*

Merci.

”

- Mohammed Anes

Dédicace

“

À mon cher père ***Kouider****,*

À ma chère mère ***Bachira****,*

Que nulle dédicace ne puisse exprimer ce que nous leurs Devons, pour tous leurs sacrifices, leur bienveillance, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,

À mon cher frère ***Ibrahim****,mes chères sœurs* ***Khadija****,* ***Fatima*** *et* ***Souad****, pour leurs encouragements permanents, je leur dédie ce modeste travail en témoignage de mon grand amour et ma gratitude infinie,*

À mes chèrs oncles ***Idriss*** *et* ***Mohammed****, puisse Dieu vous donne santé, bonheur et surtout réussite,*

À mon cher ami ***Hichem****, pour son aide et son soutien dans les moments difficiles,*

Sans oublier mon cher binôme ***Mohammed Anes****, pour son soutien moral, sa sympathie et sa patience,*

À tous ceux qui me sont chèrs, à vous tous,

Merci.

”

- Oussama

Remerciements

Tout d'abord, nous remercions Allah le tout puissant de nous avoir donnés le courage et la patience nécessaires à mener ce travail à son terme.

Nous tenons à remercier tout particulièrement notre encadrant **M. BRIXI-NIGASSA Mohammed El Amine**, pour l'aide compétente qu'il nous a apportée, pour sa patience et son encouragement. Son œil critique nous a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Nous tenons à remercier également notre cher professeur **M. BENAHMED Nasr Eddine**, Vous êtes le professeur qui a réussi à nous inspirer, à nous donner confiance en nous et en l'avenir mais aussi qui a réussi à nous donner l'envie d'apprendre. Merci pour tout ce que vous avez fait.

Que les membres de jury trouvent, ici, l'expression de nos sincères remerciements pour l'honneur qu'ils nous font en prenant le temps de lire et d'évaluer ce travail.

Un très grand remerciement et une très grande reconnaissance sont destinés à **Mme BOUAZZA née GUEN Ahlem** professeur de la Faculté de technologie Université d'Abou-BekrBelkaid Tlemcen, pour l'attention qu'il a bien voulu porter à ce travail en acceptant de le jurer et le discuter.

On tient aussi à remercier **MASSOUM Nourredine**, professeur de la Faculté de technologie Université d'Abou-BekrBelkaid Tlemcen, pour l'intérêt qu'il a accordé à ce travail en acceptant de l'examiner.

On est reconnaissants pour le temps qu'ils nous ont accordé, leurs qualités pédagogiques et scientifiques, leur franchise et leur sympathie. On a beaucoup appris d'eux et on leur adresse notre gratitude pour tout cela.

Pour finir, nous souhaitons remercier toute personne ayant contribué de prés ou de loin à la réalisation de ce travail.

Résumé

Ce travail a pour objectif la réalisation d'un système de gestion d'une salle de dilatation simple à mettre en place et peu coûteux pour une clinique d'ophtalmologie. Pour cela, nous nous reposons sur trois composants de base qui sont une carte NodeMCU intégrant un module Wifi, une application Android **DRM App** et une base de données Firebase. Lorsqu'un patient arrive à la clinique d'ophtalmologie pour une consultation, il passe par une réception, une salle de réfraction avant d'arriver chez le médecin.

Lorsque le médecin ausculte le patient, il peut alors décider de l'envoyer en salle de dilatation pour dilater ses pupilles, c'est là que notre système entre en jeu. Vu que la clinique dispose d'un nombre important de médecins et par conséquent de patients, le personnel soignant gérant cette salle de dilatation peut très vite être dépassé et ne plus se souvenir quel est le médecin traitant du patient, combien de gouttes on lui administré, quel type de collyre, etc...

Grâce à notre application Android **'DRM App'**, on peut introduire le nom du patient, définir le type de collyre à lui mettre, savoir précisément ou il est assis et quel est son médecin traitant. Aussi, par un simple clic dans l'application, nous pourrons incrémenter le nombre de gouttes administrées. Dans la partie électronique qui marche en parallèle avec l'application Android, on pourra incrémenter le nombre de gouttes en cliquant sur un bouton poussoir, tout en allumant à chaque fois une LED. L'afficheur OLED permet d'afficher les informations (nom du patient, type de collyre, nombre de gouttes, etc…) de plusieurs patients en même temps. Ces deux parties fonctionnent en parfaite adéquation grâce à la synchronisation des données via Firebase.

Tous les tests effectués ont montré le bon fonctionnement de notre système et nous encouragent à développer encore plus ce système ou d'autres systèmes du même genre.

Mots clés : Ophtalmologie, Dilatation, Arduino, clinique, Wifi, NodeMCU, Firebase, MIT App Inventor, Android.

Abstract

The aim of this work is to create a management system for a dilation room that is simple to set up and inexpensive for an ophthalmological clinic. To do this, we use three basic components which are a NodeMCU board integrating a Wifi module, an Android application called **"DRM App"** and a Firebase database. When a patient arrives at the ophthalmological clinic for consultation, he goes through a reception, a refraction room before arriving at the doctor.

When the doctor auscultates the patient, he can then decide to send him to the dilation room to dilate his pupils, this is where our system intervenes. Since the clinic has a large number of doctors and therefore patients, the nursing staff managing this dilation room can very quickly be overwhelmed and no longer remember which doctor a patient belongs to, how many drops they have been administered, what type of eyewash, etc...

Through to our Android application **'DRM App'**, we can enter the patient's name, define the type of eyewash to put on his eyes, know precisely where he is sitting and to which doctor he belongs. Also, with a simple click in the application, we can increase the number of drops administered.In the electronic part that works in parallel with the Android application, you can increase the number of drops by clicking on a push button, while lighting a LED each time.The OLED display allows you to display information (patient name, type of eyewash, number of drops, etc.) for several patients at the same time. These two parts work in perfect harmony thanks to the synchronization of data via Firebase.

All the tests carried out have shown very good results of our system and encourages us to further develop this system or other systems of the same kind.

Keywords : Ophthalmology, Dilation, Arduino, clinic, Wifi, NodeMCU, Firebase, MIT App Inventor, Android.

ملخص

الهدف من هذا العمل هو إنشاء نظام إدارة لغرفة تمدد يكون من السهل إنشاؤها وغير مكلف لعيادة طب العيون. لهذا، نعتمد على ثلاثة مكونات أساسية هي بطاقة NodeMCU التي تدمج وحدة Wifi وتطبيق Android, DRM' 'App وقاعدة بيانات Firebase. عندما يصل المريض إلى عيادة طب العيون للاستشارة، يمر عبر حفل استقبال، غرفة انكسار قبل وصوله إلى الطبيب.

عندما ينظر الطبيب إلى المريض، يمكنه بعد ذلك أن يقرر إرساله إلى غرفة التوسع لتوسيع حدقته، حيث يأتي نظامنا. نظرًا لأن العيادة بها عدد كبير من الأطباء وبالتالي المرضى، يمكن أن يغمر الموظفون الذين يديرون غرفة التوسيع هذه بسرعة كبيرة ولم يعودوا يتذكرون الطبيب الذي ينتمي إليه المريض، وعدد القطرات التي يتم إعطاؤها، ونوع قطرات العين، وما إلى ذلك...

باستخدام تطبيق App DRM ، يمكنك تقديم اسم المريض، وتحديد نوع قطرات العين التي يجب وضعها عليه، ومعرفة مكان جلوسه بالضبط والطبيب الذي ينتمي إليه. أيضًا، بنقرة بسيطة في التطبيق، يمكننا زيادة عدد القطرات المعطاة. في الجزء الإلكتروني الذي يعمل بالتوازي مع تطبيق ،Android يمكننا زيادة عدد القطرات بالنقر فوق زر الضغط، بينما نقوم في كل مرة بتشغيلLED. تسمح لك شاشة OLED بعرض المعلومات (اسم المريض ونوع وعدد قطرات العين وما إلى ذلك) للعديد من المرضى في نفس الوقت. يعمل هذان الجزءان في تناغم تام بفضل تزامن البيانات عبر .Firebase

أظهرت جميع الاختبارات التي تم إجراؤها مدى جودة عمل نظامنا وتشجيعنا على تطوير هذا النظام أو الأنظمة الأخرى من نفس النوع.

كلمات مفتاحية : طب العيون، التوسع، العيادة، ،Arduino، ،Wifi ،NodeMCU، App MIT Firebase، Android. Inventor،

Table des matières

Table des figures

Liste des tableaux

Liste des sigles et acronymes

J-C	*Jésus-Christ*
VA	*Veterans Affairs*
APD	*Agents Pharmaceutiques Diagnostiques*
AINS	*Anti-Inflammatoires Non Stéroïdiens*
DPAR	*Défaut Pupillaire Afférent Relatif*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
NodeMCU	*Node MicroController Unit*
SoC	*System on a Chip*
SDK	*Sotfware Development Kit*
IoT	*Internet of Things*
OLED	*Organic Light-Emitting Diodes*
MIT	*Massachusetts Institute of Technology*
BaaS	*Backend as a Service*
JSON	*JavaScript Object Notation*
NoSQL	*Not only Structured Query Language*
DRM App	*Dilatation Room Management Application*

Introduction générale

Chaque jour, la vie d'innombrables personnes dépend de la qualité des systèmes de santé. Ces systèmes ont des responsabilités importantes à l'égard de gens de tous âges et pour un développement sain des individus, des familles et des sociétés dans le monde entier.

Aujourd'hui plus que jamais, les systèmes de santé de tous les pays, riches ou pauvres, influencent la vie des gens. Les outils permettant d'améliorer la qualité de ces systèmes de santé ne cessent de se développer, améliorant ainsi la vie quotidienne des patients, mais aussi les conditions de travail du personnel soignant et médecins. Dans ce travail, nous nous intéressons justement à un système permettant d'améliorer la gestion d'une salle de dilatation d'une clinique d'ophtalmologie ici à Tlemcen.

Cette problématique nous a été posée par les responsables de la clinique. Ainsi, contrairement à un cabinet médical, ou on peut avoir un à deux médecins seulement, la clinique peut travailler avec cinq médecins pour une grande salle de dilatation. Ceci a une répercussion sur le nombre de malades présents dans la salle de dilatation. Par ailleurs, le problème constaté par les dirigeants de cette clinique, c'est que à partir d'un certain nombre de patients, le personnel médical est vite dépassé et ne sait plus qui est le médecin traitant du patient, quel type de collyre on lui administre, combien de gouttes ont déjà été administrées, sachant aussi que le personnel soignant ne s'occupe pas uniquement de la dilatation mais ont aussi d'autres taches tels que la réfraction, mesure de la tension oculaire, etc.

Afin de remédier à ce problème, nous proposons un système simple basé sur une carte électronique NodeMCU intégrant un module WiFi, une base de données Firebase et une application Android. Sur la carte électronique sont branchés un bouton poussoir, trois LEDs et un buzzer pour chaque patient. Un afficheur OLED est rajouté pour afficher les informations des patients, ce dernier peut prendre en charge jusqu'à quatre patients. L'application Android créée **'DRM App'** contient quant à elle une partie dédiée aux médecins et une autre dédiée au personnel soignant. Le lien entre cette partie électronique et application Android est assuré par le biais de la base de données Firebase. La partie électronique permet l'incrémentation du nombre de gouttes administrées au patient à partir de l'appuie sur le bouton poussoir. Cette information est directement enregistrée au niveau de la base de données Firebase. Les LEDs permettent quant à elles de dire au personnel soignant par un simple regard combien de gouttes a reçu le patient.

La partie application Android permet de rentrer le nom du patient, le type de collyre, le nombre de gouttes administrées, qui est le médecin traitant du patient et à quelle place il est assis. En fait les deux parties sont complémentaires et permettent dans une moindre mesure de laisser le choix au personnel soit d'utiliser son smartphone soit la carte électronique.

La démarche que nous avons suivi pour réaliser ce mémoire est la suivante :

- Dans **le premier chapitre**, nous commençons par présenter de manière générale l'ophtalmologie et son historique avant de s'intéresser à la dilatation et les différents types de collyres dilatants. A la fin de ce chapitre, nous exposerons la problématique de notre projet et proposerons une solution.
- **Le deuxième chapitre** est consacré à la présentation des composants matériels et logiciels qui nous ont été nécessaires pour mener à bien notre projet. Nous commencerons par présenter l'Arduino de manière générale en mettant l'accent sur son logiciel IDE. Nous nous intéresserons après à la carte NodeMCU ESP8266 intégrant un module Wifi utilisé dans notre projet. Nous entamerons après cela la description de l'outil MIT App Inventor que nous avons choisi pour développer notre application Android **'DRM App'**. Nous terminerons par quelques notions sur l'outil Firebase de Google qui nous a été nécessaire pour la création de notre base de données.
- Dans **le troisième chapitre**, nous détaillerons le concept de notre système de gestion de salle de dilatation en présentant dans un premier temps l'interface de notre application Android **'DRM App'** ainsi que le programme. Nous détaillerons le circuit électronique à base de carte NodeMCU et son programme réalisé pour faire fonctionner notre système de gestion de salle de dilatation ainsi que la base de données Firebase créées. Nous terminerons ce chapitre en montrant le principe et le bon fonctionnement de notre système.

Nous clôturerons ce travail par une **conclusion générale** et quelques perspectives visant à améliorer et à faire progresser un peu plus le travail réalisé.

Chapitre 1

Généralités sur l'ophtalmologie et la dilatation

1.1 Introduction

Lorsqu'on interroge les gens sur leurs cinq sens la vue, l'ouïe, le goût, l'odorat et le toucher, la plupart disent accorder une grande importance à la vue et que c'est le sens qu'ils craignent le plus de perdre. Néanmoins, le fonctionnement du sens de la vision n'est pas toujours bien compris par beaucoup d'entre nous et les problèmes pouvant affecter l'œil humain ne sont pas toujours bien connus, à moins d'y faire face ou que l'un des membres de notre entourage se trouve affecté. Lorsque c'est le cas, nous sommes dans l'obligation de consulter un ophtalmologue pour un examen de l'œil. Ce dernier peut faire une simple consultation pour corriger la vue (acuité visuelle) comme il peut approfondir l'examen en dilatant les yeux. Cette dilatation des yeux permettra au médecin de voir à l'intérieur de l'œil et par conséquent, le médecin pourra établir un diagnostic pour dire si le patient présente des problèmes oculaires ou certains facteurs de risque pouvant amener à une diminution ou perte totale de la vision.

Dans ce chapitre, nous nous intéresserons justement à la dilatation des yeux. Par ailleurs, nous commencerons tout d'abord par définir qu'est-ce que l'ophtalmologie. Nous procéderons à un petit tour d'horizon sur son histoire avant de voir par la suite qu'est-ce qu'une dilatation, son histoire, dans quels buts dilater les yeux et les procédures que doit suivre le personnel médical avant et après la dilatation.

1.2 Définition de l'ophtalmologie

L'ophtalmologie est une spécialité axée sur les soins médicaux et chirurgicaux des yeux. Les ophtalmologues sont les seuls médecins formés pour prendre en charge tous les soins oculaires et visuels. Ils peuvent prescrire des lunettes et des lentilles de contact, délivrer des médicaments, diagnostiquer et traiter des infections, des maladies oculaires, et pratiquer des interventions chirurgicales.

Fig. 1.1 : Modèle 3D de l'œil humain. [1]

1.3 Histoire de l'ophtalmologie

L'histoire connue de l'ophtalmologie remonte aux premiers jours de l'histoire écrite, où les premières observations et spéculations sur l'œil ont été consignées. Au fil des ans, la compréhension de l'anatomie et de la physiologie de l'œil a continué à se développer, et plusieurs percées majeures ont eu lieu. Cela a conduit à l'état actuel de nos connaissances sur les yeux et la santé oculaire.

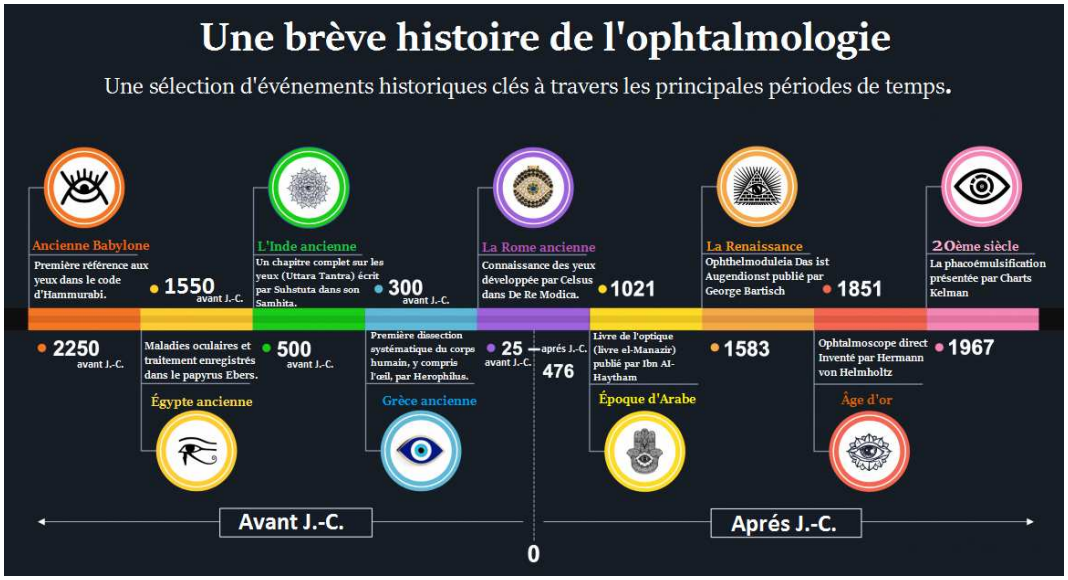


FIG. 1.2 : L'histoire de l'ophtalmologie [2]

1.3.1 Histoire ancienne

En 800 avant J-C, un chirurgien indien nommé Sushruta a décrit 76 maladies oculaires, ainsi que plusieurs techniques et instruments ophtalmologiques. Il s'est particulièrement intéressé à la chirurgie de la cataracte, et a été désigné comme le premier chirurgien de la cataracte.[3]

FIG. 1.3 : La Sushruta Samhita (traité de médecine ayurvédique).[4]

Dans les temps anciens, les conceptions anatomiques de l'œil étaient essentiellement spéculatives. La sclérotique et la cornée faisaient partie de la couche externe de l'œil, tandis que la pupille et le liquide oculaire occupaient la partie centrale. On pensait que ce liquide s'écoulait vers le cerveau par un tube. Aristote a introduit l'empirisme dans ces structures fantaisistes en disséquant les yeux des animaux, et il a ainsi découvert trois couches à l'intérieur de l'œil [3].

Rufus d'Ephèse a avancé le concept d'une quatrième couche, la couche épithéliale qui recouvre l'œil. Il a également noté que l'œil possède deux chambres, l'une remplie d'eau s'étendant de la cornée au cristallin, et l'autre remplie d'un fluide visqueux occupant l'espace entre le cristallin et la rétine.

Les études de Galien ont également eu un impact sur notre compréhension de l'œil, car il a décrit l'anatomie de la cornée, du cristallin et du nerf optique. Vésale a encore fait progresser la connaissance de la structure de l'œil, avec la découverte des couches de la sclérotique, de la rétine, de la choroïde et de la cornée, qui se rejoignent en un point.

1.3.2 Histoire contemporaine

Au Moyen Âge, des lentilles à main et des microscopes ont été utilisés pour étudier la structure et la fonction de l'œil, faisant ainsi progresser de manière significative la perception scientifique de l'anatomie de cet organe. Cependant, on ne savait toujours pas pourquoi la pupille changeait de taille, ni quelle était la nature de la rétine. En outre, la chambre postérieure de l'œil n'avait pas encore été découverte. Voici quelques points marquants de cette période :

- Georg Joseph Beer introduit l'opération de Beer comme traitement de la cataracte.
- Le baron Michael Johann Baptist de Wenzel, qui était l'oculiste du roi George III, a fait preuve d'une habileté remarquable dans l'ablation des cataractes et a légitimé ce domaine.
- Ernst Abbe est renommé pour le développement de divers instruments optiques utilisés dans le domaine de l'ophtalmologie
- Hermann von Helmholtz a inventé l'ophtalmoscope en 1851.
- Le premier hôpital dédié à la pratique de l'ophtalmologie a ouvert ses portes en 1805 à Londres.

  Il existe toujours, et est connu sous le nom de Moorfields Eye Hospital. Sir Stewart Duke Elder y a fondé l'Institut d'ophtalmologie, ce qui a fait de cet hôpital le plus grand hôpital ophtalmologique du monde. [5]

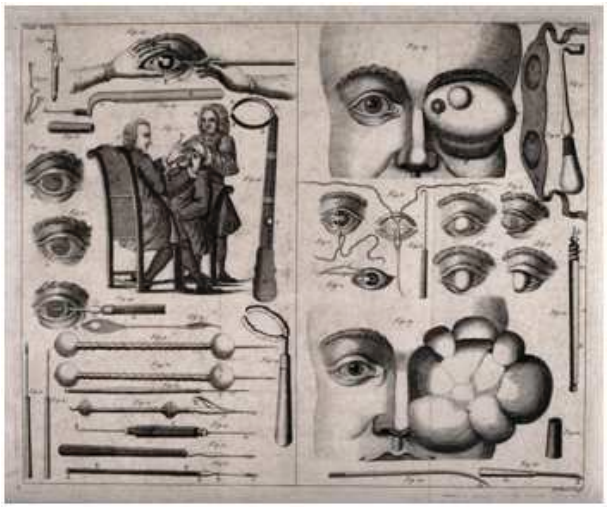


FIG. 1.4 : Les premiers instruments d'ophtalmologie. [6]

1.3.3 Histoire moderne

L'introduction de l'ophtalmoscope au 19ème siècle a entraîné une période de consolidation et d'approfondissement des connaissances sur l'œil et le traitement de diverses maladies oculaires. Cela a augmenté le niveau de précision possible dans le diagnostic et le traitement des conditions ophtalmologiques. En particulier, le traitement chirurgical du glaucome a été affiné à cette époque, ce qui a grandement contribué à améliorer les résultats pour les patients.

Tout au long du 20ème siècle, les recherches dans le domaine de l'ophtalmologie se sont encore développées. Plusieurs sous-spécialités ont été introduites pour se concentrer sur des zones ou des maladies particulières de l'œil. Il s'agit, entre autres, des sous-spécialisations de la cataracte, du glaucome, de la cornée et de l'oncologie. [3] La plupart de ces maladies oculaires nécessitent de faire un examen approfondi de l'œil, faisant ainsi appel à la dilatation des yeux.

1.4 L'examen du fond d'œil (Fundoscopie)

Chaque fois que l'on rend visite à un médecin ophtalmologue, celui-ci fait une série de tests : il contrôle la vision, mesure la tension oculaire et observe la rétine, une membrane mince et transparente destinée à recevoir les impressions lumineuses qui assurent la vision. C'est cette membrane, ainsi que la papille optique (extrémité du nerf optique),qui fera l'objet d'un examen du fond de l'œil.

La fundoscopie consiste à administrer des gouttes ophtalmiques qui dilatent les pupilles afin d'examiner le cristallin, évaluer l'état de la rétine ou de déterminer précisément ou se situe le problème de vision.

FIG. 1.5 : Examen du fond de l'œil.[7]

1.5 Définition de la dilatation

Une dilatation est une méthode thérapeutique qui augmente le diamètre de la pupille. Cette opération a pour but de fournir au médecin ophtalmologue un diamètre pupillaire suffisant afin qu'il puisse examiner pleinement le nerf optique et la rétine. Cet examen est essentiel pour prévenir et traiter les troubles oculaires susceptibles d'entraîner une perte partielle ou totale de la vision. Elle se pratique le plus souvent par l'instillation d'un collyre. [8]

FIG. 1.6 : Gouttes traditionnelles pour la dilatation des yeux. [9]

1.6 Historique de la dilatation

La première loi sur le permis d'exercer l'optométrie a été adoptée il y a près de 120 ans, par contre, les médecins ophtalmologues ne peuvent dilater les patients que depuis 50 ans.

À la fin des années 1960, les ophtalmologistes commençaient à s'inquiéter de certains aspects de la pratique. Ils parvenaient à résoudre bon nombre des problèmes de leurs

patients grâce à la correction réfractive (lunettes et lentilles), mais ils se heurtaient à une impasse lorsque le phoroptère, dispositif de réfraction ne permettait pas d'obtenir une vision normale. S'ils pouvaient utiliser leurs ophtalmoscopes directs pour visualiser le pôle postérieur, cela ne leur donnait pas toujours suffisamment d'informations pour résoudre le problème. Ils étaient obligés de référer leurs patients à la communauté ophtalmologique et souvent, ils ne revoyaient pas leurs patients.

C'est donc en 1970 qu'un groupe d'ophtalmologistes en herbe s'est réuni dans un hôtel près de l'aéroport LaGuardia à New York et a décidé de faire quelque chose pour élargir leur champ d'activité. [10] (Cette réunion est désormais connue sous le nom de "Conférence LaGuardia").

Après s'être battus avec la communauté médicale, les défenseurs de l'ophtalmologie ont réussi à faire adopter une loi à Rhode Island qui donnait aux ophtalmologistes la possibilité de dilater les patients. De nombreux États ont rapidement suivi le mouvement.

L'un des principaux arguments de la communauté médicale contre l'utilisation de gouttes dilatantes par les ophtalmologistes était leur manque de formation à l'utilisation des médicaments et leur manque de compétences dans les procédures de diagnostic effectuées après la dilatation. L'ophtalmologie a contré cet argument en faisant en sorte qu'un certain nombre d'ophtalmologistes du système VA, qui avaient obtenu la capacité de dilater avant tous les médecins non VA, aillent former les légions de praticiens désireux d'obtenir des privilèges de dilatation.

L'accès de l'ophtalmologie aux APD a été établi par les praticiens qui ont prouvé leur compétence dans des domaines tels que l'examen du fond de l'œil à la lampe à fente et l'ophtalmoscopie indirecte binoculaire. La seule façon de prouver cette compétence était de s'exercer, et la seule façon de s'exercer était de dilater beaucoup de personnes. Il n'est pas difficile de comprendre comment la dilatation est devenue une partie intégrante de l'examen complet.

1.7 La dilatation des yeux

Un examen de dilatation des yeux implique l'application de gouttes ophtalmiques spécialement conçues pour dilater les pupilles (Mydriaticum® collyre ; Néosynéphrine®collyre ; Skiacol® collyre). La dilatation des pupilles permet de faire pénétrer plus de lumière dans les yeux grâce à l'augmentation du diamètre pupillaire, cela permet au médecin de mieux diagnostiqué des affections qu'il ne pourrait pas voire autrement.

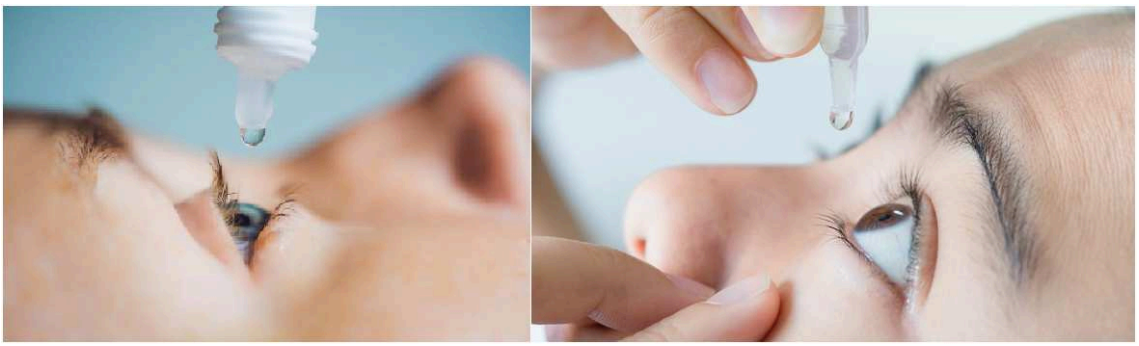


Fig. 1.7 : Application de collyre à un patient (adulte [11] - enfant [12])

Les gouttes ophtalmiques maintiennent les pupilles dilatées pendant toute la durée de l'examen oculaire afin que l'ophtalmologiste puisse examiner minutieusement chaque œil. Cet examen lui permet de vérifier les stades précoces des maladies, ce qui peut faire la différence entre des problèmes de vision graves et une affection gérable.

Fig. 1.8 : Pupille gauche dilatée – Pupille droite normale.

Lorsque l'ophtalmologiste procède à un examen des yeux dilatés, il commence par administrer des gouttes ophtalmiques à chaque œil. Il faut ensuite environ 15 à 30 minutes pour que les yeux se dilatent complètement, cela dépend bien sûr du patient et des différentes pathologies qu'il peut présenter. Une fois qu'ils sont dilatés, le médecin peut effectuer les tests suivants [13] :

- Test de réponse pupillaire : le médecin braque une petite lampe de poche dans chaque œil pour voir comment les pupilles réagissent à la lumière, si c'est le cas, alors la dilatation n'est pas totale.
- Test de fonction des muscles oculaires : L'ophtalmologiste demande au patient de garder la tête immobile et de suivre un objet en mouvement, tel qu'un stylo, uniquement avec ses yeux, afin de tester la force et la réponse de ses muscles oculaires.

La nécessité de faire dilater les yeux dépend de l'âge, de l'état de santé et du risque de maladie oculaire. L'ophtalmologiste détermine si le patient a besoin d'une dilatation des yeux en fonction des critères suivants :

- Antécédents familiaux : S'il a des antécédents familiaux de maladies oculaires génétiques ou s'il a déjà rencontré des problèmes oculaires.
- Les maladies à risque : Certaines maladies, comme le diabète, le patient s'expose à un risque important pouvant même le mener à la cécité visuelle.

- Troubles de la vision : un rendez-vous chez l'ophtalmologiste pour un problème de vision, il peut dilater les yeux du patient pour évaluer la vision en profondeur.

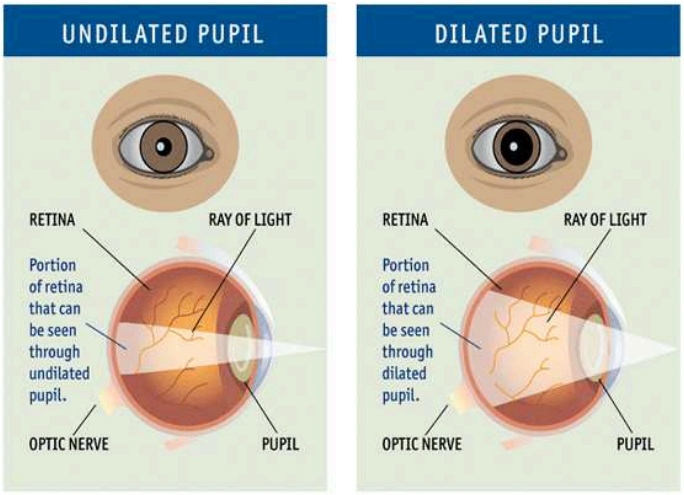


FIG. 1.9 : La différence entre une pupille délattée et non dilatée. [14]

1.8 Les collyres en ophtalmologie

Les gouttes ophtalmiques sont des gouttes liquides appliquées directement à la surface de l'œil. Les gouttes oculaires contiennent généralement du sérum physiologique pour correspondre à la salinité de l'œil. Les gouttes ophtalmiques peuvent également contenir un ou plusieurs médicaments pour traiter une grande variété de maladies oculaires. Selon l'affection traitée, ils peuvent contenir des stéroïdes, des antihistaminiques, des bêtabloquants, des AINS, des antibiotiques, des antifongiques ou des anesthésiques topiques.

FIG. 1.10 : Un ensemble de gouttes ophtalmique. [15]

Lors de l'application de gouttes ophtalmiques, le patient doit pencher légèrement la tête en arrière. L'ophtalmologue tire doucement sur la paupière inférieure et presse le tube jusqu'à ce qu'une goutte soit distribuée.

FIG. 1.11 : Démonstration de l'application d'un collyre à un patient. [16]

1.8.1 Mydriatiques topiques (gouttes dilatantes)

Le but des mydriatiques est de dilater la pupille, facilitant ainsi l'examen de l'œil avec un ophtalmoscope ou une lampe à fente. Le choix du mydriatique topique est déterminé par la durée d'action de chaque goutte. Les mydriatiques les plus utilisés chez l'adulte sont le tropicamide et la phényléphrine car leurs effets dilatateurs s'estompent plus rapidement.

Chez les enfants, l'atropine et la skiacol sont les plus couramment utilisées. En cas de doute, le mieux est d'utiliser une goutte de tropicamide 1% dans chaque œil pour faciliter l'examen par ophtalmoscopie directe. [17]

Il est essentiel d'effectuer une évaluation des pupilles (y compris le DPAR) avant d'utiliser des gouttes dilatantes. L'examen d'un DPAR après l'utilisation de gouttes dilatatrices peut entraîner des résultats confus car on ne sera pas en mesure d'interpréter les signes cliniques.

Médicament	Spécialités et Conditionnements	Excipients à effet notoire	Indications	Durée d'action
Atropine	**Atropine faure® :** 1% boite de 100 unidoses de 0,4mL **Atropine alcon® :** 0,3%, 0.5%, 1% flacon de 10mL	**Atropine alcon® :** nitrate phénylmercurique	Préparation à certains examens réalisés par l'ophtalmologiste. Traitement de certains affections de l'oeil.	40 minutes à 2 semaines
Cyclopentolate	**Skiacol®** 0,5% **:** unidose de 0,5mL	Chlorure de benzalkonium	Préparation à certains examens réalisés par l'ophtalmologiste (fond d'oeil) ou à certaines opérations de l'oeil.	20-30 minutes Peut durer 12 heures
Phényléphrine	**Néosynéphrine faure® :** 2,5% boites de 1 et de 20 unidoses de 0,4mL 5% flacon de 5mL 10% flacon de 5mL et boite de 100 unidoses de 0,4mL	Flacon : thiomersal	Pour obtenir une mydriase à visée diagnostique, thérapeutique, pré-opératoire.	6 heures
Tropicamide	**Mydriaticum® :** 0,5% flacon de 10mL ou unidose de 0,4mL	Chlorure de benzalkonium	Préparation à certains examens réalisés par l'ophtalmologiste (fond d'oeil) ou à certaines opérations de l'oeil	10-15 minutes

Tab. 1.2 : Mydriatiques topiques et leur durée d'action.

1.9 Le diagnostic après la dilatation

Grâce à cette dilatation des yeux, le médecin ophtalmologiste pourra évaluer l'état de la rétine, le nerf optique, les vaisseaux sanguins et d'autres parties qu'il ne peut pas voire lors d'un examen des yeux non dilatés. Il pourra ainsi déterminer entre autres si le patient présente des problèmes liés aux [18] :

- Le diabète (différents stades) : Le diabète peut entraîner une série de problèmes oculaires, notamment la rétinopathie diabétique, le glaucome et la cataracte.

- L'hypertension artérielle : l'hypertension artérielle peut entraîner un durcissement des vaisseaux sanguins de la rétine, des fuites et des hémorragies.
- Dégénérescence maculaire : l'ophtalmologiste recherchera la dépigmentation, l'agglutination des pigments et la néo-vascularisation, c'est-à-dire la croissance de nouveaux vaisseaux sanguins sous la rétine du patient.
- Décollement de la rétine : Le décollement de la rétine se produit lorsque cette dernière se détache des vaisseaux sanguins qui l'alimentent en oxygène. Cette affection est considérée comme une urgence médicale.
- Glaucome : Le glaucome est une maladie qui endommage le nerf optique lorsque du liquide s'accumule à l'avant de votre œil (augmentation de la tension oculaire). [19]

1.10 La problématique de notre projet

Comme nous l'avons expliqué, la dilatation exige une certaine précision,et c'est ce que chaque médecin cherche à faire. Lorsqu'on est dans un cabinet médical à un voire deux médecins, cette dilatation reste gérable par le personnel soignant ou le médecin lui-même. Dès lors que le nombre de médecin augmente, il devient très difficile de tenir le compte du temps de dilatation, types et nombre de gouttes administrées, etc. C'est justement le problème rencontré dans une clinique d'ophtalmologie ici à Tlemcen.

Cette clinique dispose d'un étage de consultation ophtalmologique. Sur ce dernier, travaille jusqu'à 5 médecins pour une seule et grande salle de dilatation. Ce qu'ont constaté les dirigeants de la clinique, c'est qu'il y a souvent une mauvaise coordination entre le personnel de la clinique, patient et médecins, et ce à cause d'une charge de travail importante.

A partir de ce constat, nous proposons d'aider le personnel soignant en leur fournissant un système de gestion d'une salle de dilatation. Le but de ce système est de coordonner et synchroniser l'information sur la dilatation d'un patient entre le personnel soignant sur le nombre de gouttes appliquées, le temps de dilatation, quel est le médecin traitant du patient, est-ce que ce dernier nécessite une intervention ou une procédure particulière.

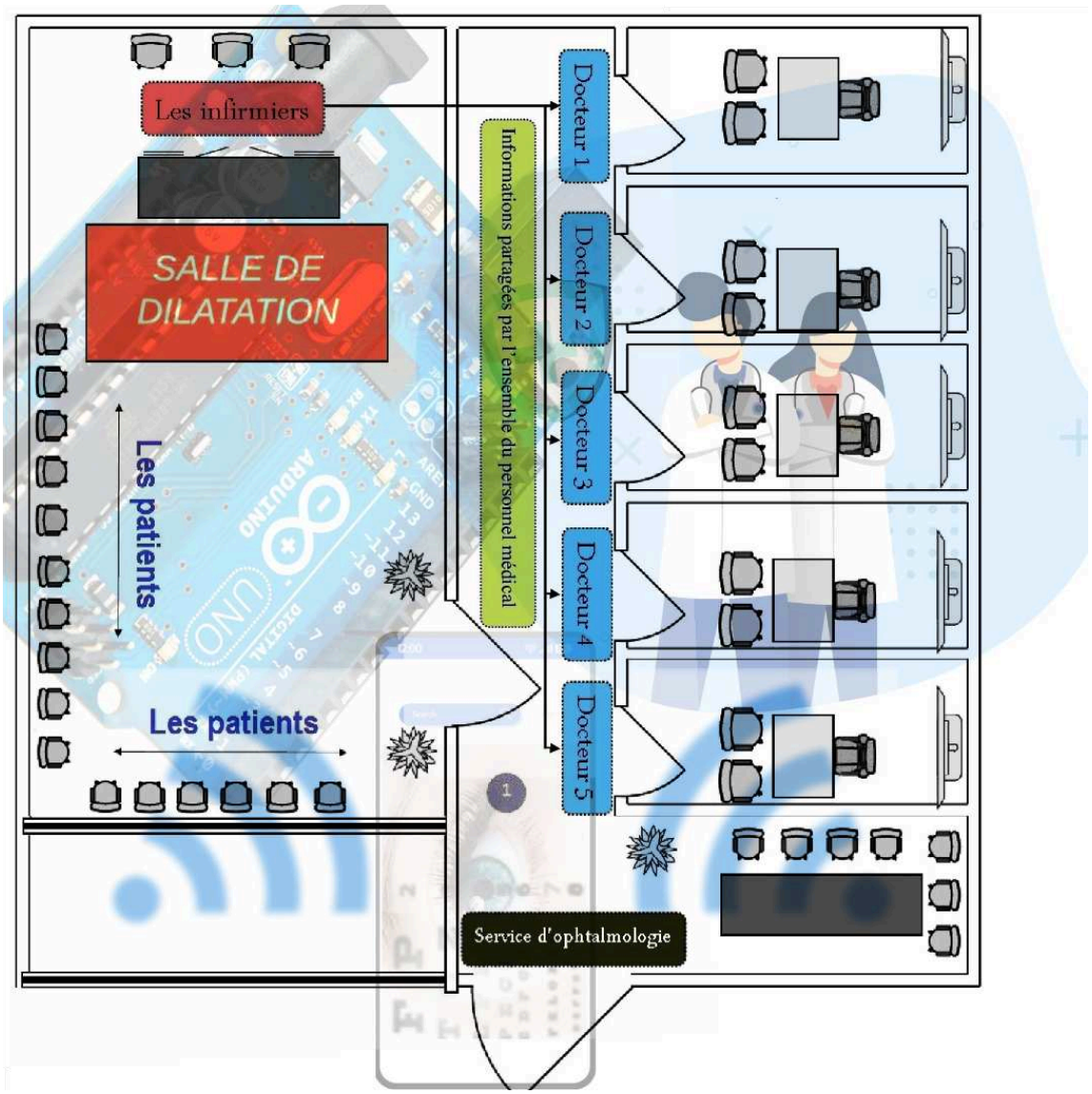


FIG. 1.12 : Plan pour le principe de fonctionnement de notre système.

1.11 Conclusion

Dans ce chapitre, nous avons commencé par présenter l'ophtalmologie de manière général avant de définir qu'est-ce que la dilatation et quel est son rôle en ophtalmologie. Cette dernière permet entre autres la détection précoce de certaines maladies ou facteurs de risques pouvant mener à une perte partielle ou totale de la vue. Nous avons ensuite présenté les différentes gouttes ophtalmique ou collyres utilisés pour dilater les yeux. A partir de là nous avons posé la problématique de notre travail qui consiste à réaliser un système de gestion d'une salle de dilatation. Ce système permettra de fluidifier le travail du personnel soignant et médecins et tendra à améliorer la qualité de service offerte aux patients venant faire une consultation ophtalmologique. Dans le chapitre suivant, nous allons présenter les différents outils nous permettant de réaliser un tel système.

Chapitre 2

Outils matériels et logiciels nécessaires à la réalisation de notre système de gestion de salle de dilatation

2.1 Introduction

Après avoir posé le contexte et la problématique de notre travail, nous présenterons dans ce chapitre les différents composants matériels et logiciels qui nous ont été nécessaires pour mener à bien ce projet.

Nous commencerons par une présentation sur l'Arduino et le module Wifi ESP8266, avant de détailler la carte de développement NodeMCU utilisée dans notre projet. Une description de l'écran OLED SSD1306 0,96 pouces I2C, son fonctionnement avec l'Arduino et ses caractéristiques seront détaillés. Le buzzer sera lui aussi présenté et son fonctionnement avec la carte Arduino détaillé. Dans un deuxième temps, nous présenterons l'outil MIT App Inventor utilisé pour créer notre application Android.

Enfin, à la fin de ce chapitre, quelques notions sur l'utilisation de l'outil de gestion de base de données Firebase de Google seront données.

2.2 Concept de notre système de gestion de salle de dilatation

Le concept que nous avons imaginé repose sur la base d'une carte à microcontrôleur du type Arduino, un module Wifi ESP8266, une application Android et une base de données Firebase.

L'objectif ici est de créer une application Android dans laquelle les noms des patients, le type de collyre et le nombre de gouttes qu'ils ont reçus sont introduits. Ces informations sont synchronisées sur l'ensemble des smartphones du personnel médical. Le nombre de gouttes administrées peut être soit introduit via le smartphone du personnel soignant ou bien via l'action sur un bouton physique branché sur la carte Arduino. Aussi, le médecin aura toutes les informations sur la prise en charge de son patient sur son smartphone si besoin, et pourra décider de faire revenir le patient en consultation pour l'examen du fond d'œil.

Le schéma synoptique présenté en figure 2.1 va nous permettre de mieux comprendre le fonctionnement global du système.

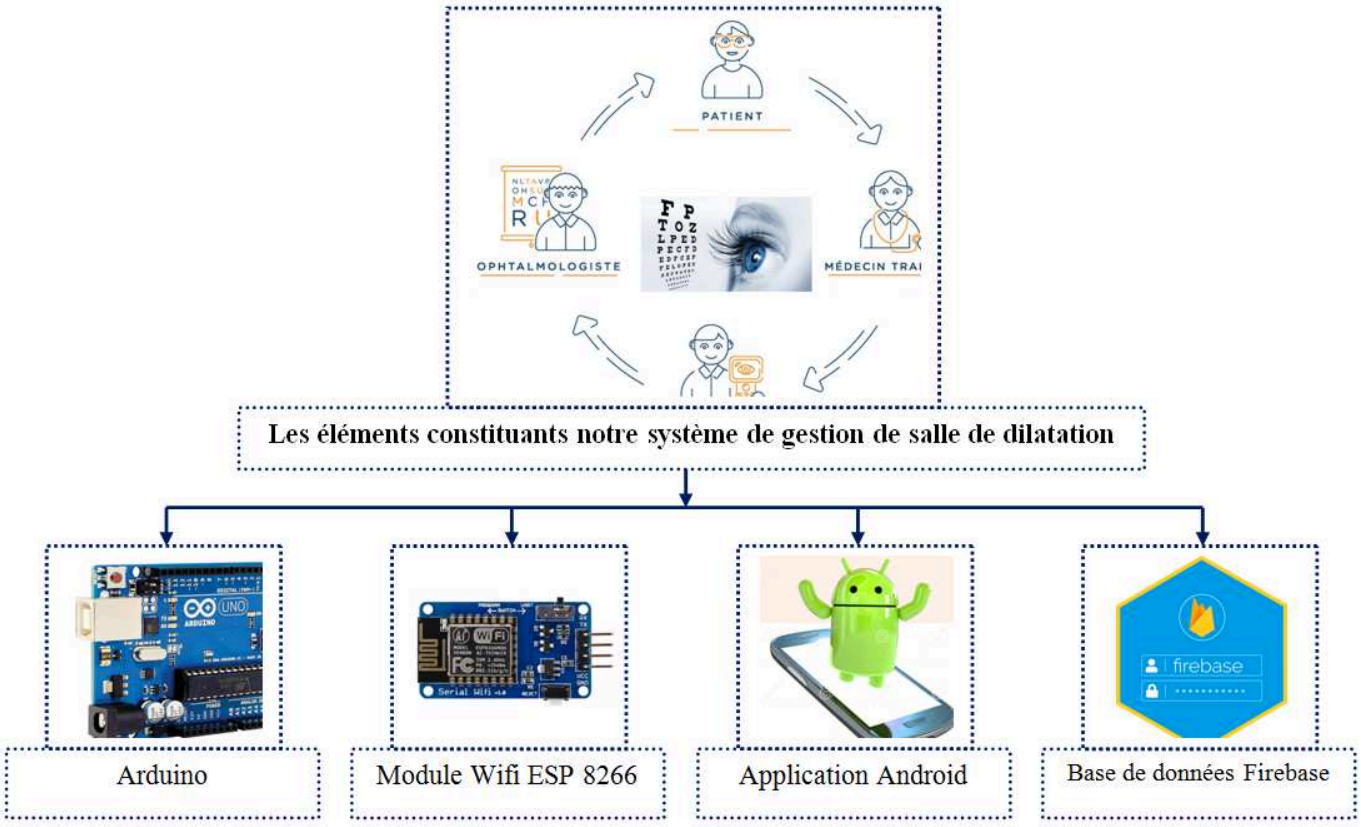


FIG. 2.1 : Le schéma synoptique de système à réaliser.

Pour détailler les composants de notre système, nous avons choisi de diviser ces derniers en deux grandes parties : partie matérielle et partie logicielle

2.3 La partie Hardware du système

Dans cette partie, nous définissons les composants électroniques constituants notre système de gestion de salle de dilatation.

2.3.1 Arduino

Arduino est une plate-forme de développement et de prototypage électronique open source. La plate-forme intègre deux composants de base : carte électronique et IDE (logiciel de programmation). Il existe de nombreux types de cartes de développement Arduino, comme illustré à la Figure 2-2

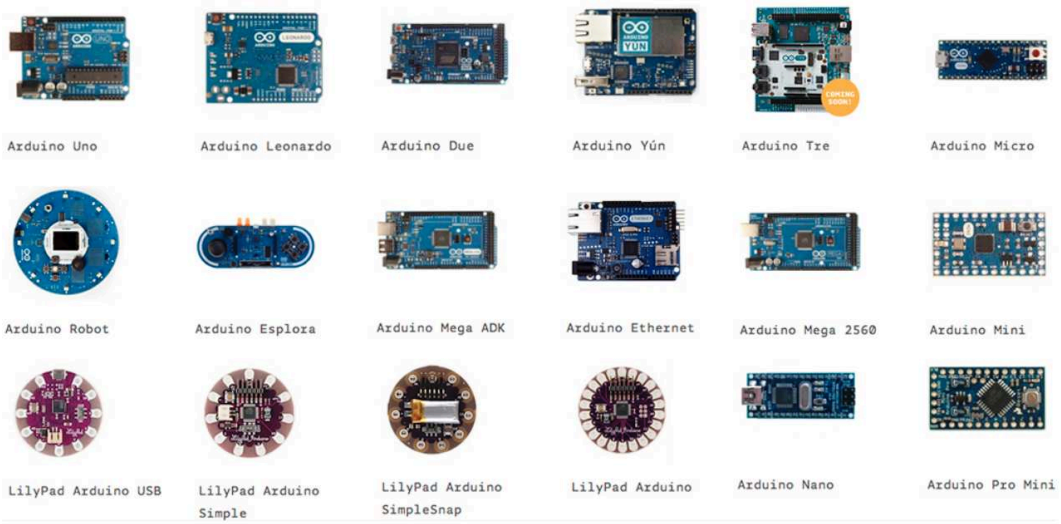


FIG. 2.2 : Exemples de cartes Arduino.

Comme nous le verrons plus loin, notre choix se portera sur une carte à base d'Arduino avec un module Wifi intégré.

2.3.2 IDE Arduino

L'IDE est un logiciel multi plateforme open source contenant des outils permettant l'édition, la compilation et le téléchargement d'un programme réalisant une fonction donnée. Il comprend également des outils de débogage, de production de programmes complexes, d'autres fonctions et des outils visuels. L'interface du logiciel Arduino se présente comme illustré par la figure 2.3

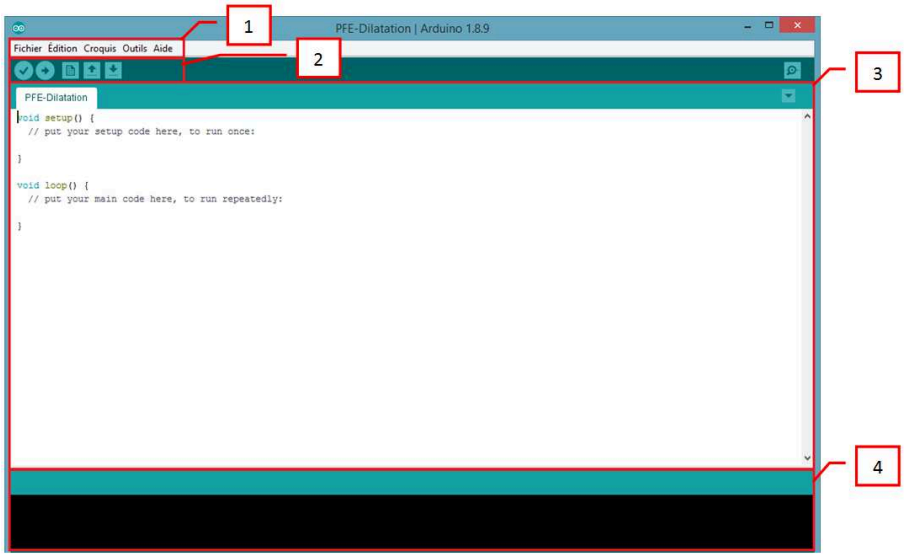


FIG. 2.3 : interface de l'IDE Arduino.

	Description
1	Options de configuration du logiciel.
2	Boutons pour la programmation des cartes.
3	Programme à créer.
4	Débogueur (affichage des erreurs de programmation).

TAB. 2.1 : Description d'interface de l'IDE Arduino.

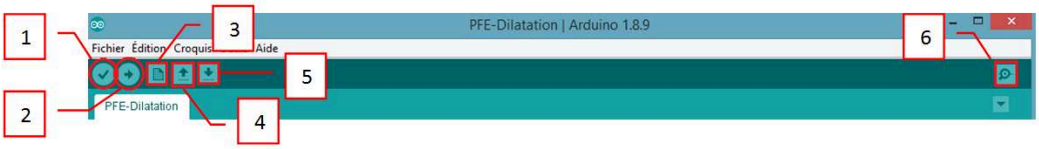


FIG. 2.4 : Les boutons de logiciel Arduino.

Boutons :

	Description
1	Permet de vérifier et actionner un module qui cherche les erreurs dans le programme.
2	Compiler et envoyer le programme vers la carte.
3	Créer un nouveau fichier.
4	Charger un programme existant.
5	Sauvegarder le programme en cours.
6	Moniteur série : Permet la communication entre la carte Arduino et l'IDE.

TAB. 2.2 : Description des boutons de logiciel Arduino.

2.3.3 Module Wifi ESP8266

Pour que des appareils sans fil puissent communiquer entre eux, il faut respecter un certain ensemble de normes.

Le WiFi ou réseau local sans fil est classé sous les normes 802.11 de l'IEEE (Institute of Electrical and Electronics Engineers). Les normes IEEE ont été certifiées par la WiFi Alliance, une organisation à but non lucratif créée pour certifier les produits IEEE 802.11 et les promouvoir en tant que normes de réseau local sans fil. [20]

Dans notre projet, nous avons besoin d'un module Wifi pouvant être relié à une carte Arduino afin d'assurer la communication sans fil avec l'application Android. Plusieurs choix s'offraient alors à nous. Soit partir sur une solution basée sur une carte Arduino avec un module Wifi séparé, soit partir sur une solution combinée.

Fig. 2.5 : Exemples de modules Wifi ESP8266. [21]

Nous avons opté pour une solution alliant une carte électronique basée sur la plateforme Arduino avec un module WiFi ESP8266. Cette solution est appelée NodeMCU ESP8266.

2.3.4 NodeMCU ESP8266

Le NodeMCU est un environnement de développement logiciel et matériel open-source construit autour d'un système sur puce SoC. Le NodeMCU conçu et fabriqué par Espressif Systems contient un système d'exploitation moderne et un SDK. Cela en fait un excellent choix pour les projets d'internet des objets de toutes sortes. [22]

Fig. 2.6 : Carte NodeMCU ESP8266. [23]

Le NodeMCU est constitué de 30 broches réalisant chacune une fonction bien définie, comme le montre la Figure 2-7 ci-dessous.

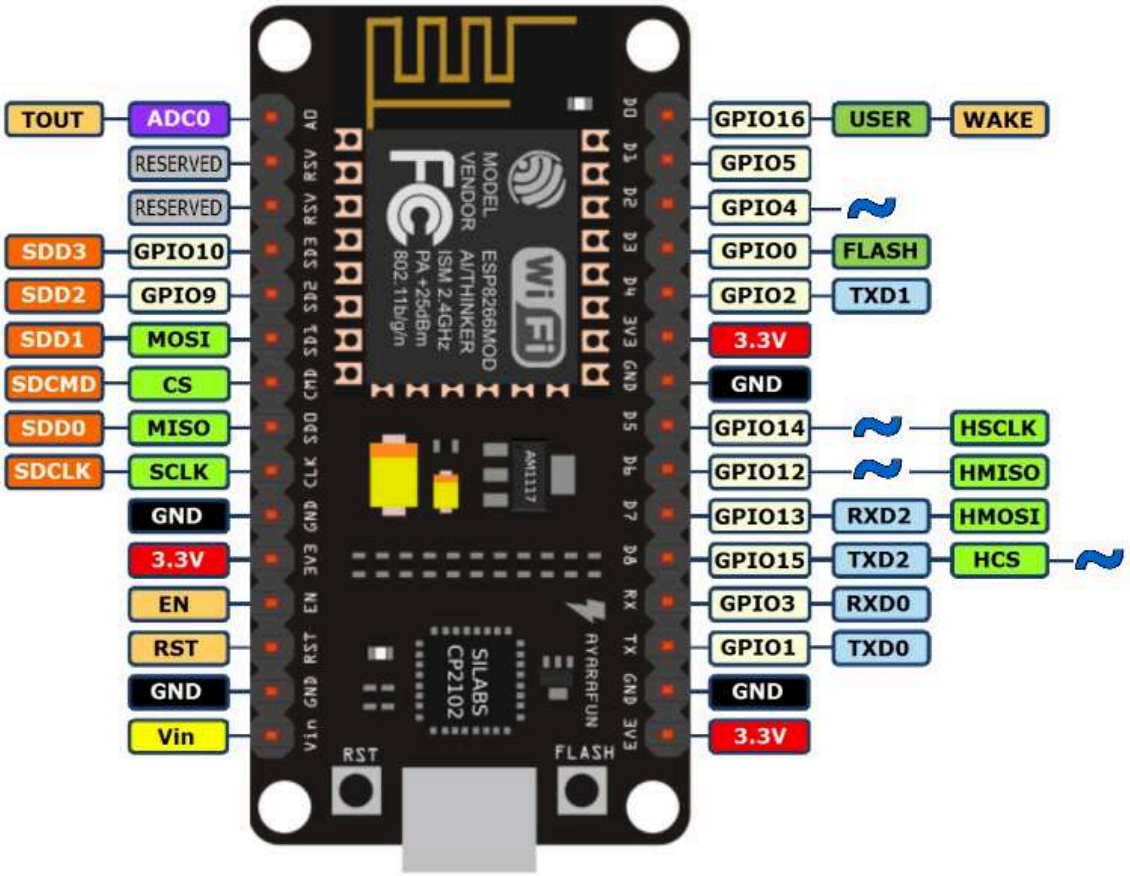
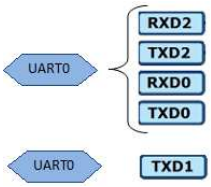


Fig. 2.7 : Brochage NodeMCU esp8266.[24]

Le tableau 2.3 qui suit montre en détail les fonctions de chaque broche du NodeMCU [25]

Alimentation	Il y a quatre broches d'alimentation. La broche VIN et trois broches 3.3V	Vin Elle est utilisée pour alimenter directement le NodeMCU et ses périphériques. La puissance délivrée sur VIN est régulée par le régulateur embarqué sur le module NodeMCU 3.3V Les sorties du régulateur de tension embarqué et peuvent être utilisées pour alimenter des composants externes.
GND	La masse	GND Sont les broches de masse de NodeMCU/ESP8266
GPIO	NodeMCU/ESP8266 possède 17 broches GPIO	GPIO Peuvent être assignées à des fonctions telles que I2C, I2S, UART, PWM, télécommande IR, lumière LED et bouton de manière programmatique. Chaque GPIO à activation numérique peut être configurée en pull-up ou pull-down interne, ou en haute impédance.
I2C Inter-Integrated Circuit	D1 SCL (serial clock) est utilisé pour l'horloge synchrone entre les dispositifs maîtres et esclave. D2 SDA (serial Data) est utilisé pour l'échange de données entre les dispositifs maître et esclave.	I2C Sont utilisés pour connecter des capteurs et des périphériques I2C. Le maître et l'esclave I2C sont tous deux pris en charge.
UART	NodeMCU/ESP8266 possède 2 interfaces UART 	UART Permettent une communication asynchrone (RS232 et RS485). UART0 (broches TXD0, RXD0, TXD2 et RXD2) peut être utilisé pour la communication. Cependant, l'UART1 (broche TXD1) ne comporte qu'un signal de transmission de données
PWM	La carte dispose de 4 canaux de modulation de largeur d'impulsion (PWM).	Peut être implémentée de manière programmatique et sert à contrôler les moteurs numériques et les LEDs.
Canal ADC	Le convertisseur analogique-numérique	ADC0 Est utilisé pour convertir le signal analogique en forme numérique. L'ESP8266 possède un CAN 10 bits intégré avec un seul canal CAN, c'est-à-dire qu'il n'a qu'une seule broche d'entrée CAN pour lire la tension analogique d'un dispositif externe.
SPI	L'interface périphérique série (SPI) est un protocole de connexion d'interface de bus lancé à l'origine par Motorola Corp. Le NodeMCU dispose de deux SPI (SPI et HSPI) en modes esclave et maître	MOSI (Master In Slave Out) Le maître reçoit des données et l'esclave transmet des données via cette broche. MISO (Master Out Slave In) Le maître transmet des données et l'esclave reçoit des données via cette broche. SCLK (Serial clock) Le maître génère cette horloge pour la communication, qui est utilisée par l'esclave. Seul le maître peut déclencher une horloge série. CS (Chip Select) Le maître peut sélectionner le dispositif esclave par le biais de cette broche pour commencer à communiquer avec lui.
SDIO	NodeMCU est doté d'une interface d'entrée/sortie numérique sécurisée (SDIO)	SDIO Est utilisée pour interfacer directement les cartes SD
Contrôle	Permettent de commander le NodeMCU Ces broches comprennent la broche d'activation de puce (EN), la broche de réinitialisation (RST) et la broche WAKE.	WAKE Est utilisée pour faire sortir la puce de sa mise en veille. EN Est utilisée pour réinitialiser la puce ESP8266 RST La puce ESP8266 est activée lorsque la broche EN est tirée vers le HAUT. Lorsqu'elle est tirée vers le BAS, la puce fonctionne avec une puissance minimale.

TAB. 2.3 : Fonctions des broches de la carte NodeMCU ESP8266.

Un avantage non négligeable du NodeMCU est qu'il peut être programmé avec l'IDE Arduino, nous n'avons pas besoin d'apprendre un autre langage de programmation. Par ailleurs, il faut noter que pour fonctionner correctement avec l'IDE Arduino, la bibliothèque ESP8266doit être installée.

2.3.5 Installation des bibliothèques ESP8266 dans l'IDE

Après l'ouverture de l'IDE Arduino, il suffit accéder simplement au menu « Préférences ».

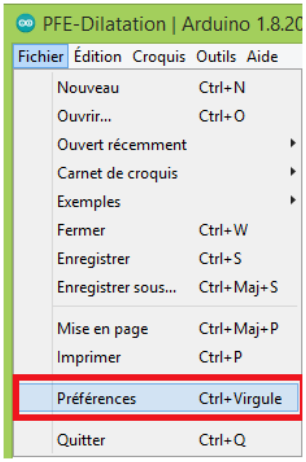


FIG. 2.8 : Menu des préférences.

Dans la section URL de gestion des cartes supplémentaires, nous introduirons l'URL suivante.(Voir la figure 2-9) :

http://arduino.esp8266.com/stable/package_esp8266com_index.json

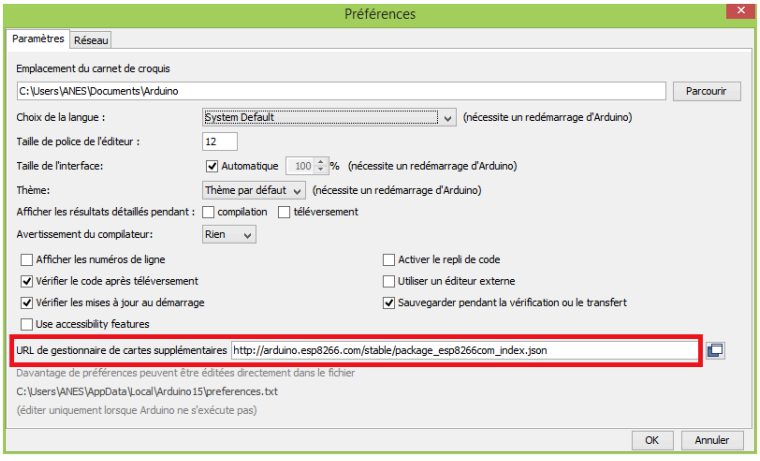


FIG. 2.9 : Insertion du lien d'installation de la bibliothèque ESP8266 dans l'IDE.

Puis il faudra aller dans le menu outil, chercher le Type de carte : Arduino/GenuineUno et sélectionner le Gestionnaire de cartes.

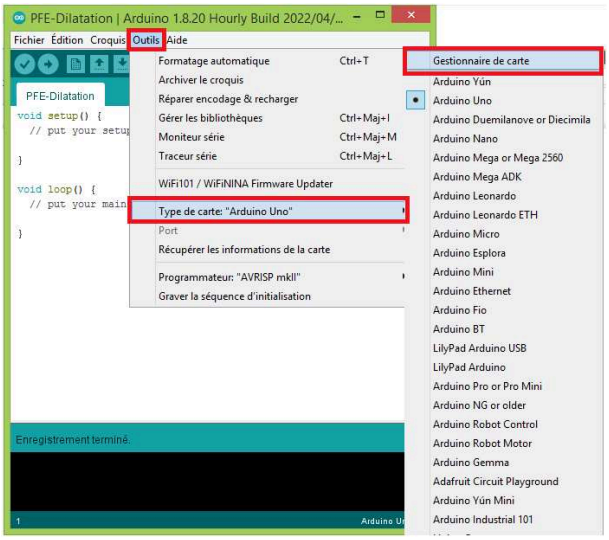


FIG. 2.10 : Menu type de carte de l'IDE.

Une nouvelle fenêtre s'affiche, il suffira alors de faire une recherche sur l'ESP8266 pour l'installer.

FIG. 2.11 : Installation de la bibliothèqueESP8266.

Une fois la bibliothèque ESP8266 installée, il suffit de sélectionner la carte NodeMCU 1.0 (ESP-12E Module) dans le menu type de carte comme le montre la figure 2-12.

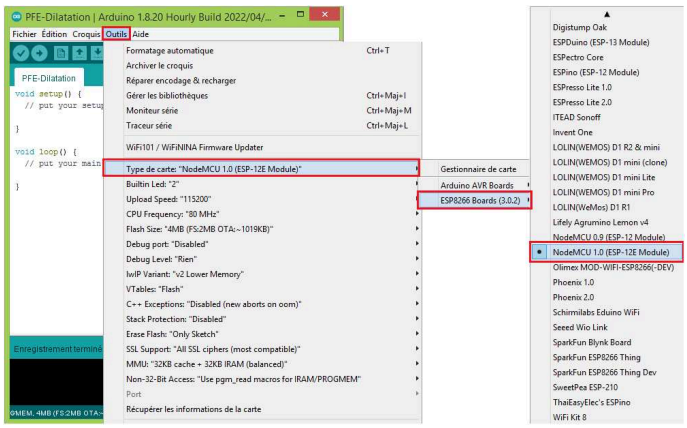


FIG. 2.12 : Choix de la version de la carte NodeMCU.

2.4 Afficheur OLED I2C

Les afficheurs électroniques sont des dispositifs d'affichage permettant de présenter des images fixes ou mobiles, du texte ou de la vidéo transmise par voie électronique, sans produire d'enregistrement permanent.

L'écran à diodes électroluminescentes organiques (OLED) que nous utiliserons dans ce projet est le modèle SSD1306, un écran monocolore de 0,96 pouce avec 128×64 pixels, comme le montre la figure suivante.

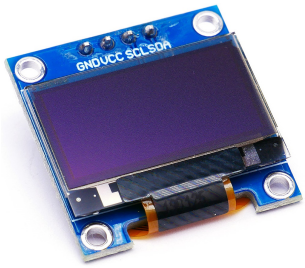


Fig. 2.13 : Afficheur OLED I2C. [26]

2.4.1 Caractéristiques

Un écran OLED fonctionne sans rétroéclairage car il produit sa propre lumière. C'est pourquoi l'écran présente un contraste aussi élevé, un angle de vision extrêmement large et peut afficher des niveaux de noir profonds. L'absence de rétroéclairage réduit considérablement la puissance requise pour faire fonctionner l'OLED. En moyenne, l'écran utilise environ 20mA de courant, bien que cela dépende de la partie de l'écran qui est éclairée.

La tension de fonctionnement du contrôleur SSD1306 est de 1.65V à 3.3V tandis que le panneau OLED nécessite une tension d'alimentation de 7V à 15V. Toutes ces différentes exigences en matière d'alimentation sont satisfaites grâce à un circuit de pompe de charge interne. Cela permet de le connecter facilement à un Arduino ou à tout microcontrôleur à logique 5V sans utiliser de convertisseur de niveau logique. [27] Voici les spécifications complètes :

Technologie d'affichage	OLED
Interface MCU	I2C / SPI
Taille de l'écran	0,96 pouce de large
Résolution	128×64 pixels
Tension de fonctionnement	3,3V - 5V
Courant de fonctionnement	20mA max
Angle de vue	160°
Caractères par rangée	21
Nombre de rangées de caractères	7

Tab. 2.5 : Les spécifications complètes de l'afficheur OLED I2C. [28]

2.4.2 Mémoire OLED I2C

Quelle que soit la taille du module OLED, le pilote du SSD1306 dispose d'une zone de mémoire intégrée de 1KB (Graphic Display Data RAM) pour l'écran qui contient le motif binaire à afficher. Cette zone de mémoire de 1K est organisée en 8 pages (de 0 à 7). Chaque page contient 128 colonnes/segments (bloc 0 à 127). Et chaque colonne peut stocker 8 bits de données (de 0 à 7). [27]

8 pages x 128 segments x 8 bits de données = 8192 bits = 1024 octets = 1KB de mémoire.

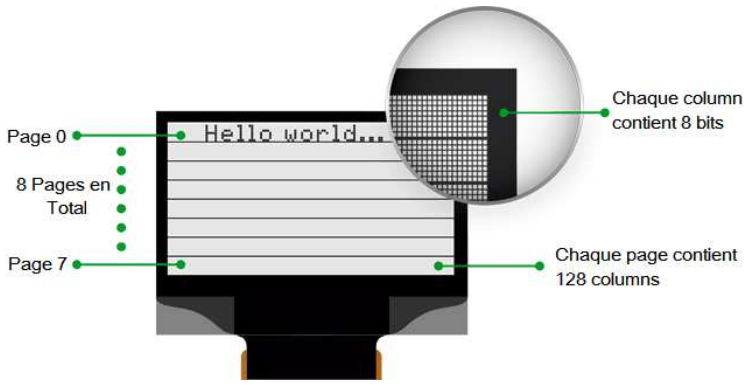


Fig. 2.14 : L'ensemble de la mémoire 1K d'OLED.

2.4.3 Brochage du module d'affichage OLED

Avant de donner un exemple de branchement avec une carte arduino, nous présentons son brochage en figure 2.15.

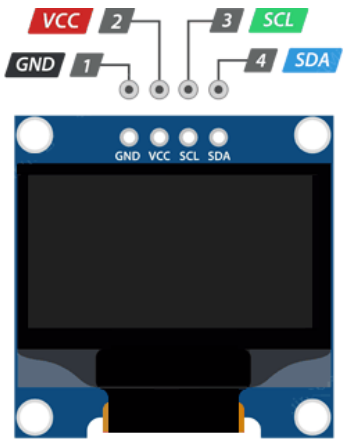


Fig. 2.15 : Les broches de l'afficheur OLED I2C.

Broche	Description
GND	Doit être connecté à la masse de l'Arduino.
VCC	Est l'alimentation de l'écran que nous connectons à la broche 5 volts de l'Arduino.
SCL	Est une broche d'horloge série pour l'interface I2C
SDA	Est une broche de données série pour l'interface I2C

Tab. 2.7 : Description des broches de l'afficheur OLED I2C.

2.4.4 Exemple de brochage d'un afficheur OLED I2C avec une carte Arduino UNO

Avant de télécharger du code et d'envoyer des données à l'écran, nous allons connecter l'écran à l'Arduino.

Avant d'envoyer des données à l'écran par le biais d'un programme injecté dans la carte Arduino, nous allons connecterl'écran à cette dernière.Les connexions sont assez simples. Nous commencerons par connecter la broche VCC à la sortie 5V de l'Arduino et la broche GND à la masse. Il nous reste maintenant deux broches utilisées pour la communication I2C à connecter à la carte Arduino. Ces broches nommées SCL (ligne d'horloge) et SDA (ligne de données) sont connectées aux broches analogiques A5 et A4 de l'Arduino Uno.

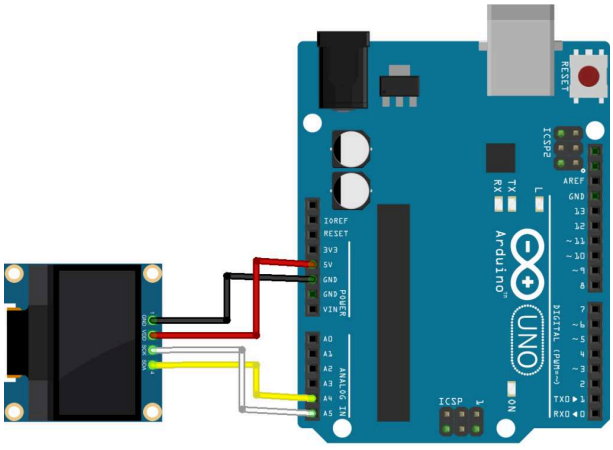


Fig. 2.16 : OLED I2C avec carte arduino UNO.

2.5 Buzzer d'arduino

Un buzzer arduino est également appelé buzzer piézo. Il s'agit essentiellement d'un minuscule haut-parleur qu'on peut connecter directement à un Arduino. On peut lui faire émettre un son à une fréquence que nous définissons. Le buzzer produit un son basé sur l'effet piézoélectrique inverse.

Le buzzer produit le même son bruyant quelle que soit la variation de tension qui lui est appliquée. Il est constitué de cristaux piézoélectriques entre deux conducteurs. Lorsqu'un potentiel est appliqué à ces cristaux, ils poussent sur un conducteur et tirent sur l'autre. Cette action de poussée et de traction produit une onde sonore. La plupart des buzzers produisent un son de l'ordre de 2 à 4 kHz.

FIG. 2.17 : Piézo Buzzer.[29]

2.5.1 Brochage de buzzer avec arduino UNO

Un buzzer étant de faible puissance, il peut être branché directement sur le micro-contrôleur sur n'importe laquelle de ses broches de sorties. Sur la carte Arduino, on doit spécifier sur quelle broche se trouve le buzzer, à quelle fréquence (en Hertz, Hz) on veut qu'il émette un son et pendant combien de temps (en millisecondes).

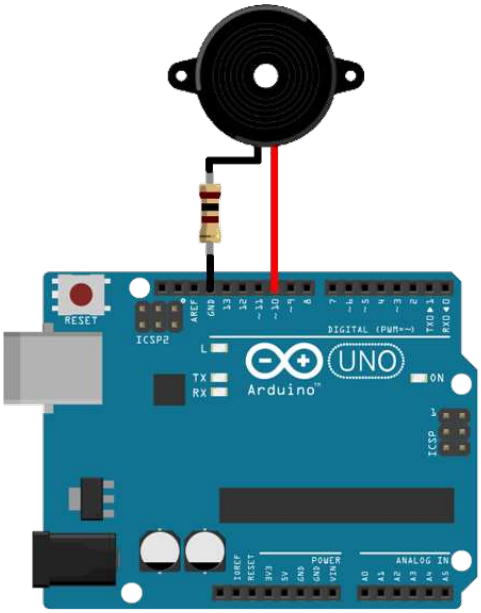


FIG. 2.18 : Exemple de brochage buzzer avec arduino UNO.

2.6 Partie Software

Dans cette section, nous introduirons l'outil de création d'applications Android, MIT App Inventor. Cet outil permet de créer des applications Android de manière simple et

conviviale permettant la communication avec une carte Arduino par le biais d'une base de données Firebase de Google que nous détaillerons dans la fin de ce chapitre.

2.6.1 Application Android

Android est un système d'exploitation mobile qui a été développé par Google pour être principalement utilisé dans les appareils à écran tactile comme les téléphones mobiles et tablettes, téléviseurs, voitures, montres/bracelets connectés, chacun étant doté d'une interface utilisateur unique. [30] Ces applications Android peuvent être créées par des moyens multiples. Pour notre part, nous opterons pour l'outil MIT App Inventor.

MIT App Inventor

MIT App Inventor est un environnement de programmation visuel et intuitif développée par Google et désormais géré par le MIT, permet à chacun de créer des applications entièrement fonctionnelles pour smartphones et tablettes.[31] Cet outil basé sur des blocs facilite la création d'applications complexes à fort impact en beaucoup moins de temps que les environnements de programmation traditionnels.

Fig. 2.19 : Interface haute de MIT App Inventor.

EnvironnementMIT App Inventor

Pour pouvoir utiliser MIT App Inventor, il est nécessaire de créer un compte sur la plateforme, cela permet notamment de sauver tous ses projets et d'y accéder à tout moment.

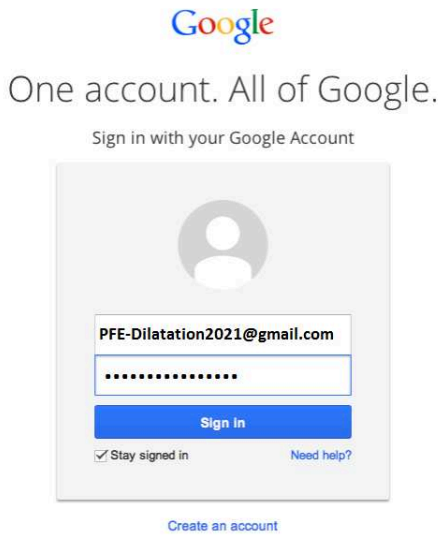


Fig. 2.20 : Log à App Inventor avec un nom d'utilisateur et un mot de passe gmail.

L'environnement MIT App Inventor comprend deux fenêtres principales : Designer et blocks.

Desinger : C'est là qu'on définit l'aspect et la convivialité d'application, spécifier les fonctionnalités qu'elle doit avoir et le choix des éléments pour l'interface utilisateur. Cette fenêtre Designer comprend quatre sous-fenêtres, comme l'illustre la figure 2-21.

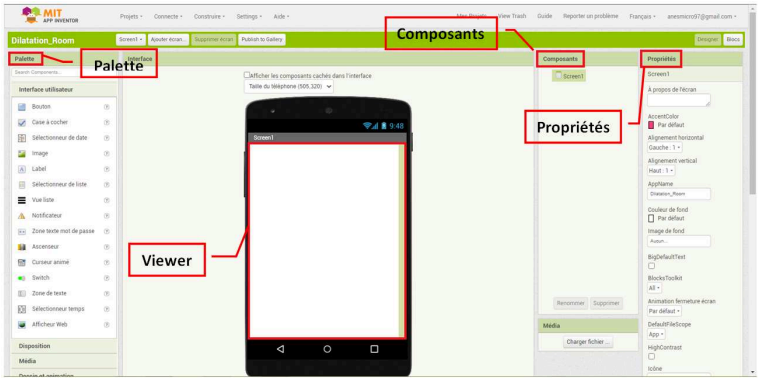


FIG. 2.21 : Fenêtre Designer sous MIT App Inventor.

	Description
1	Palette : C'est là qu'on retrouve les composants de base de MIT APP Inventor, il suffit de les faire glisser vers la partie Viewer pour lesajouter à notre application.
2	Viewer : C'est la partie dans laquelle l'interface de notre application sera définie
3	Composants : Inclut tous les éléments que nous avons ajoutés à notre application.
4	Propriétés : Permet de visualiser les différentes propriétés relatives à chaque composant ajouté.

TAB. 2.9 : Description du fenêtre Designer sous MIT App Inventor.

Block : L'éditeur de blocs nous permet de programmer le comportement de l'application en assemblant des blocs en fonction des composants ajoutés à son interface (Figure 2-22).

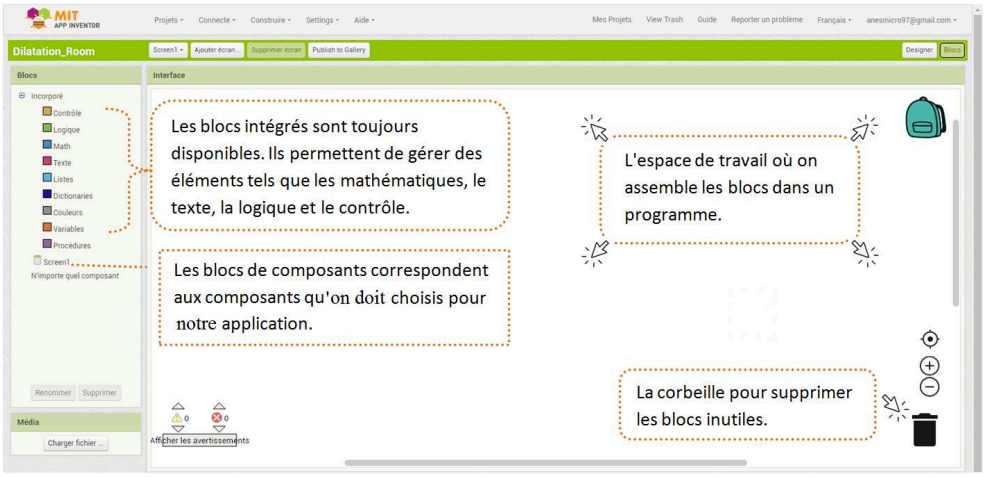


FIG. 2.22 : Fenêtre Blocks sous MIT App Inventor.

Dès que la programmation de l'application Android est terminée, elle est automatiquement sauvegardée. On peut ensuite tester l'application de trois manières distinctes avant l'installation :

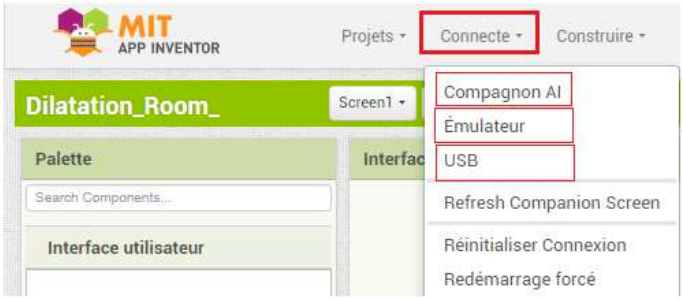


FIG. 2.23 : Onglet connecte dans MIT App Inventor.

Compagnon AI : c'est un outil utile pour visualiser en permanence notre application en temps réel sur un appareil Android à chaque étape du processus de développement.

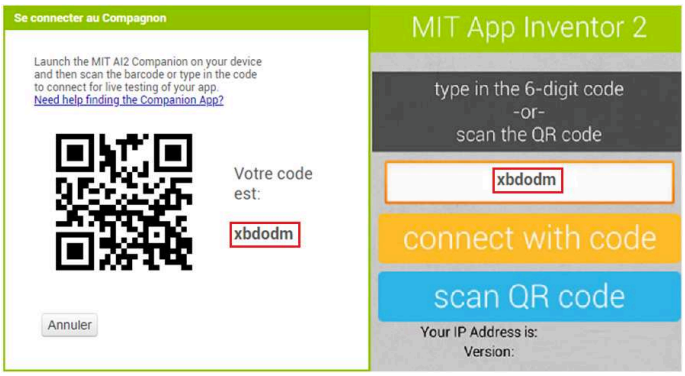


FIG. 2.24 : Code QR à scanner + code sur MIT App Inventor pour tester l'application.

Émulateur : Celui-ci rend possible le test de l'application sur un émulateur Android « aiStarter » installé sur un ordinateur.

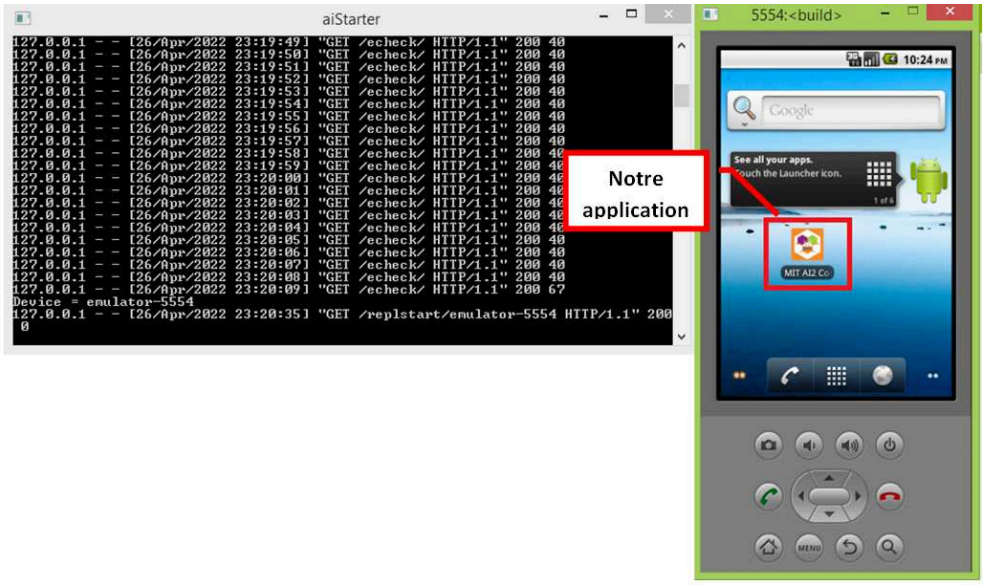


Fig. 2.25 : Émulateur de MIT App Inventor.

USB : Cette option permet de tester l'application sur un smartphone connecté à un ordinateur via un câble USB à l'aide du logiciel « aiStarter ».

Si les tests des applications sont concluants, l'application peut être installée sur un smartphone. MIT App Inventor permet d'installer une application de deux façons distinctes :

- Par numérisation d'un code QR de l'application AiCompanion installée sur le smartphone.
- En téléchargeant le fichier « .APK » sur son ordinateur, le copier et lancer sur le smartphone.

Fig. 2.26 : Choix du type d'installation de l'application dans le menu « Construire » de MIT App Inventor.

2.6.2 Firebase de Google

Firebase est un Backend-as-a-Service (Baas). Il nous fournit une variété d'outils et de services pour nous aider à développer des applications de qualité, à accroître leur base d'utilisateurs et à réaliser des bénéfices. Il est construit sur l'infrastructure de Google.

FIG. 2.27 : Logo de Firebase. [32]

Firebase est classé comme un programme de base de données NoSQL (Les bases de données sont principalement désignées comme des bases de données non relationnelles ou distribuées.), qui stocke les données dans des documents de type JSON.

Grâce à cela, la plateforme permet notamment de gérer l'authentification des utilisateurs, de tester ses applications sur l'ensemble des plates-formes (web, iOs, Android), de procéder aux mises à jour à distance, obtenir et analyser les rapports de crash, etc. [32]

Création d'un serveur Firebase

Pour pouvoir créer une base de données Firebase, on doit ouvrir le site :

https://firebase.google.com/

La page d'accueil de Firebase s'affiche, ensuite on va cliquer sur le bouton « Get started » pour créer un projet (Figure 2-28)

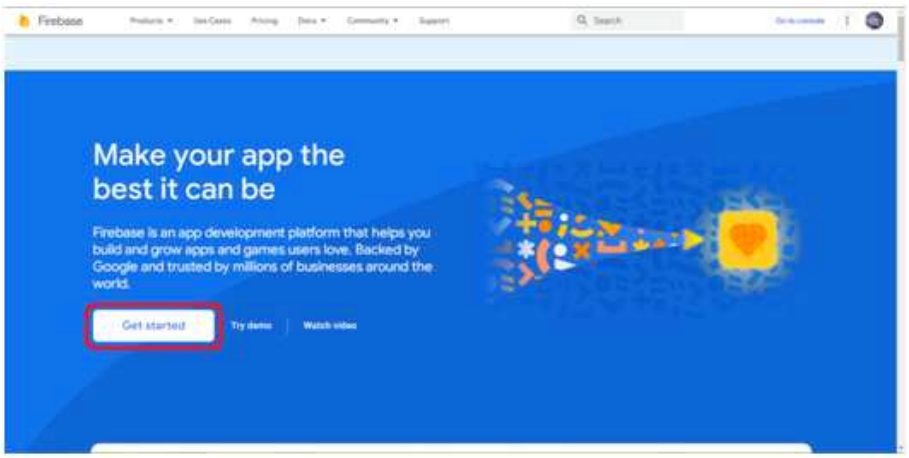


FIG. 2.28 : Page d'accueil de Firebase.

Après cela, il faudra s'inscrire via un compte Gmail afin d'utiliser ce service. Une fois la session ouverte, une nouvelle page apparaît (figure 2-29), on peut alors modifier un projet existant ou en créer un autre.

Fig. 2.29 : Fenêtres des projets créés sur Firebase.

Base de données en temps réel Firebase (Firebase Realtime Database)

La base de données en temps réel de Firebase est une base de données hébergée dans le « cloud ». Les données sont stockées en JSON et synchronisées en temps réel avec chaque client connecté. Lorsqu'on crée des applications multiplateformes en utilisant Android, tous les clients partagent une instance de base de données en temps réel et reçoivent automatiquement des actualisations avec les données les plus récentes.[33] Nous allons maintenant créer notre base de données (figure 2-30).

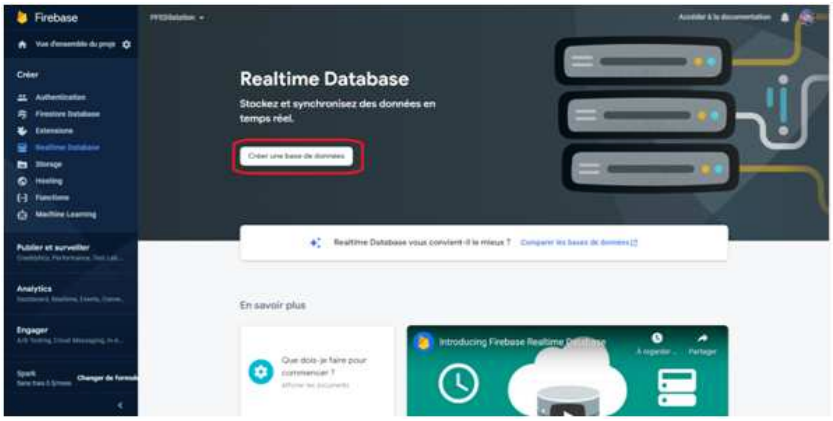


Fig. 2.30 : Fenêtre de création de base de données dans Firebase.

Une fois la base de données créée, les liens d'ouverture de session et d'authentification doivent être récupérés et entrés dans MIT App Inventor. Pour ce faire, on accède simplement à l'onglet RealtimeDatabase et sélectionné pour créer une base de données comme indiqué dans la figure 2-31.

Le lien de notre base de données :

https://pfedilatation-default-rtdb.firebaseio.com/

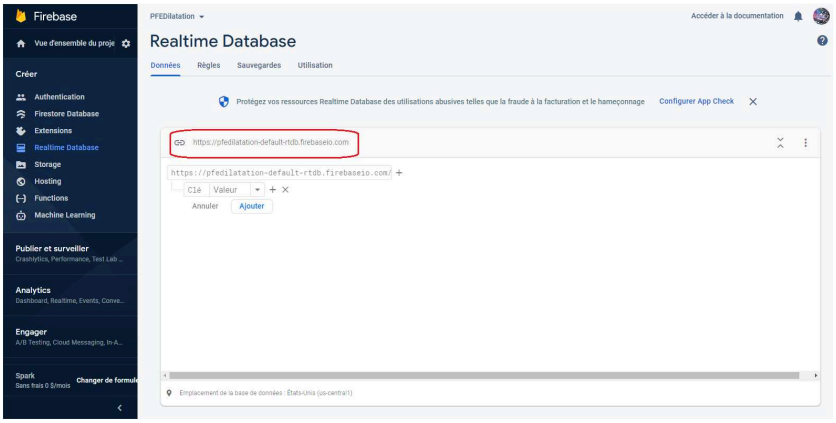


FIG. 2.31 : Lien de la base de données.

Le mot de passe de notre base de données :

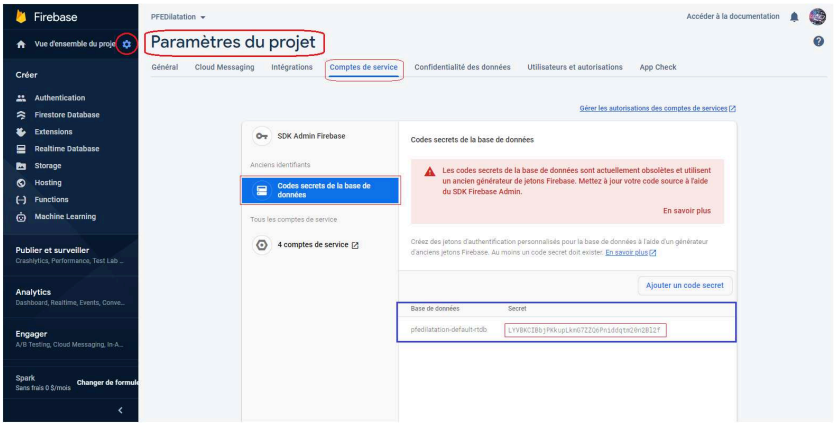


FIG. 2.32 : Mot de passe de la base de données.

2.7 Conclusion

Dans ce chapitre, nous sommes intéressés aux différents composants et outils nécessaires à la réalisation du système de gestion de salle de dilatation. Pour plus de clarté, nous avons scindé ce chapitre en deux parties : partie matérielle et partie logicielle. Dans la section matérielle, nous avons présenté l'Arduino dans son ensemble avant de se focaliser sur la carte NodeMCU incorporant un module Wifi. Nous avons aussi décrit le fonctionnement de l'écran OLED I2C et du buzzer avec la carte Arduino. Dans la partie logicielle, nous avons présenté l'outil de développement d'application Android MIT App Inventor ainsi que le service Firebase de Google.

Dans le prochain chapitre, nous commencerons la réalisation notre projet de gestion de salle dilatation à l'aide des outils présentés dans ce chapitre.

Chapitre 3

Conception de système de gestion de salle de dilatation

3.1 Introduction

Dans ce dernier chapitre, nous exposerons le circuit et l'application **'DRM App'** réalisés pour la gestion de la salle de dilatation. Nous commencerons par détailler l'application Android réalisée avant de présenter le circuit électronique à base de carte NodeMCU fonctionnant avec cette application, Nous finirons ce chapitre par des tests pratiques montrant le principe et le bon fonctionnement de notre système de gestion de salle de dilatation.

3.2 Description de notre système de gestion de la salle de dilatation

La figure 3.1 montre le schéma bloc de notre système de gestion de la salle de dilatation. Le principe de fonctionnement de ce dernier repose sur l'échange d'informations entre les trois composants de base de notre système, à savoir la carte NodeMCU, base de données Firebase et l'application Android. En fait tout commence lorsque le patient arrive à la clinique pour une visite médicale, Le patient passe d'abord par une réception, une salle de réfraction ensuite chez le médecin. Arrivé à cette étape, le médecin peut décider en fonction de la consultation effectuée d'envoyer le patient en salle de dilatation, c'est là que notre système entre en jeu.

D'abord le membre du personnel soignant prend connaissance auprès du médecin, le nom du patient et le type de collyre, il fait ensuite rentrer ces informations dans la base de données via l'application Android, en choisissant le bon type de collyre à attribuer et la place ou sera assis le patient. A partir de là, l'administration du collyre au patient peut commencer en incrémentant à chaque application le nombre de gouttes.

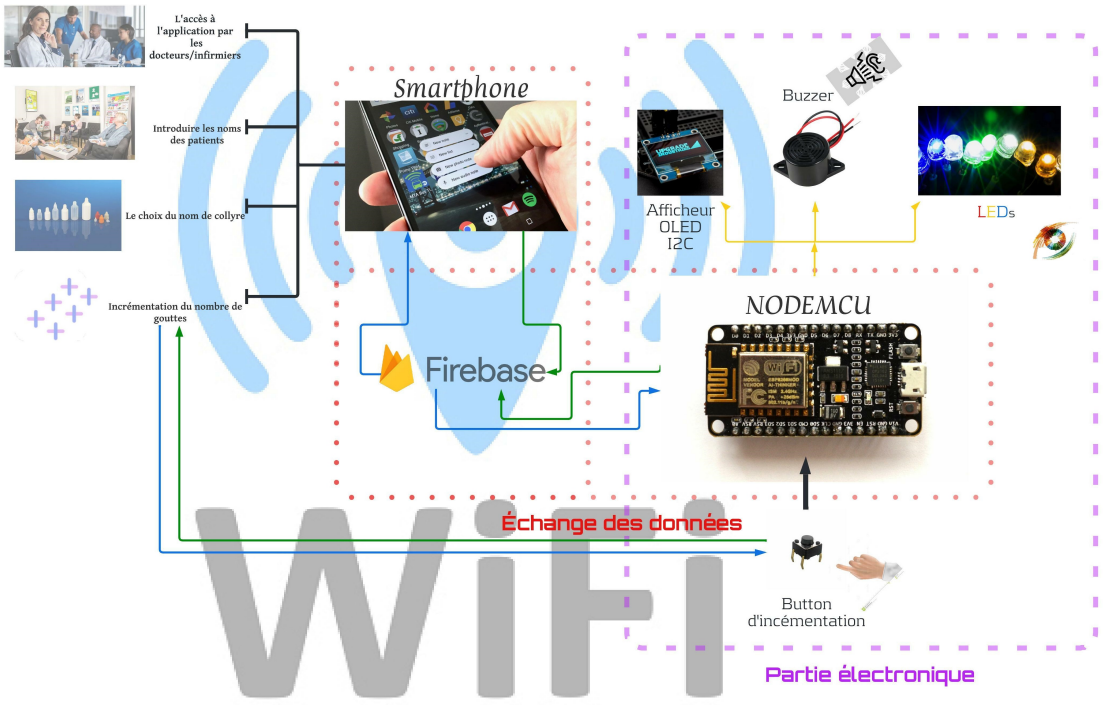


Fig. 3.1 : Schéma bloc de notre système.

Cette incrémention peut se faire soit via l'application soit via un bouton branché sur la carte NodeMCU. Sur cette même carte, nous branchons aussi trois LEDs, un buzzer et un afficheur OLED, A chaque fois que l'infirmier met une goutte de collyre à un patient et actionne le bouton d'incrémentation (ou via application), une LED s'allume. Passé un certain temps (en fonction des gouttes), 10 min par exemple, la LED commence à clignoter et un buzzer commence à émettre un son, ceci indique qu'il est temps d'ajouter la deuxième goutte. L'infirmier vient alors lui administrer sa deuxième goutte et il valide l'opération pour confirmer que ça a été fait, et ainsi se fait le reste de l'opération jusqu'à atteindre la dilatation.

Notons par ailleurs que le médecin a accès via son compte dans l'application Android à la liste de tous ses patients, ou ils sont assis et ou sont-ils niveau dilatation.

Dans ce qui suit, nous allons détailler la manière dont le design de l'application a été élaboré et comment ont été programmés les différents blocs afin que tout soit synchronisés avec Firebase et la carte NodeMCU.

3.3 L'application Android 'DRM App'

Comme nous l'avons présenté dans le deuxième chapitre, la programmation de l'application Android à partir de MIT App Inventor doit se faire en deux parties :

- Création de l'interface de notre application dans la partie Designer de MIT App Inventor.
- Programmation des composants de l'interface Android (Boutons, screen, Firebase, etc..)

3.3.1 Partie Designer

Notre application Android se compose de six fenêtres principales (screen), comme le montre la figure 3-2 suivante.

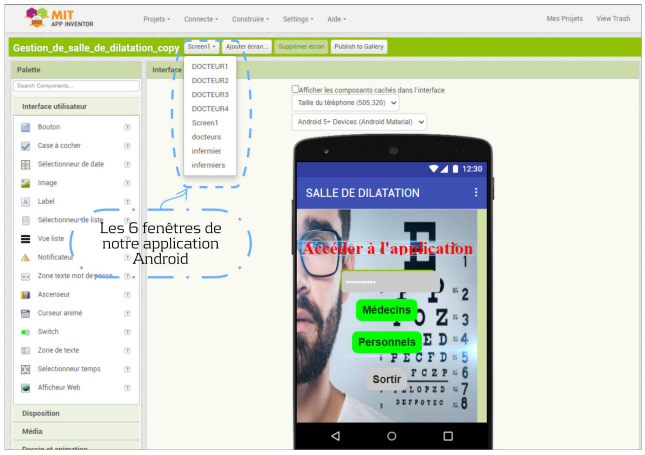


FIG. 3.2 : Les six (6) fenêtres principales .

Chaque fenêtre permet de réaliser une opération donnée. Nous allons dans ce qui suit voir le rôle justement de chacune de ces fenêtres.

En fait lorsque vous lancez l'application depuis votre smartphone vous arrivez à l'écran principal (Nommé Screen 1 dans notre application). Cette fenêtre est une première sécurité à l'accès à l'application, car un mot de passe doit être introduit, ensuite le manipulateur doit dire est-ce qu'il est médecin ou bien s'il fait partie du personnel soignant(figure 3-3).

En choisissant médecin ou personnels, vous êtes renvoyés sur une deuxième fenêtre ou un identifiant ou mot de passe doit être introduit.

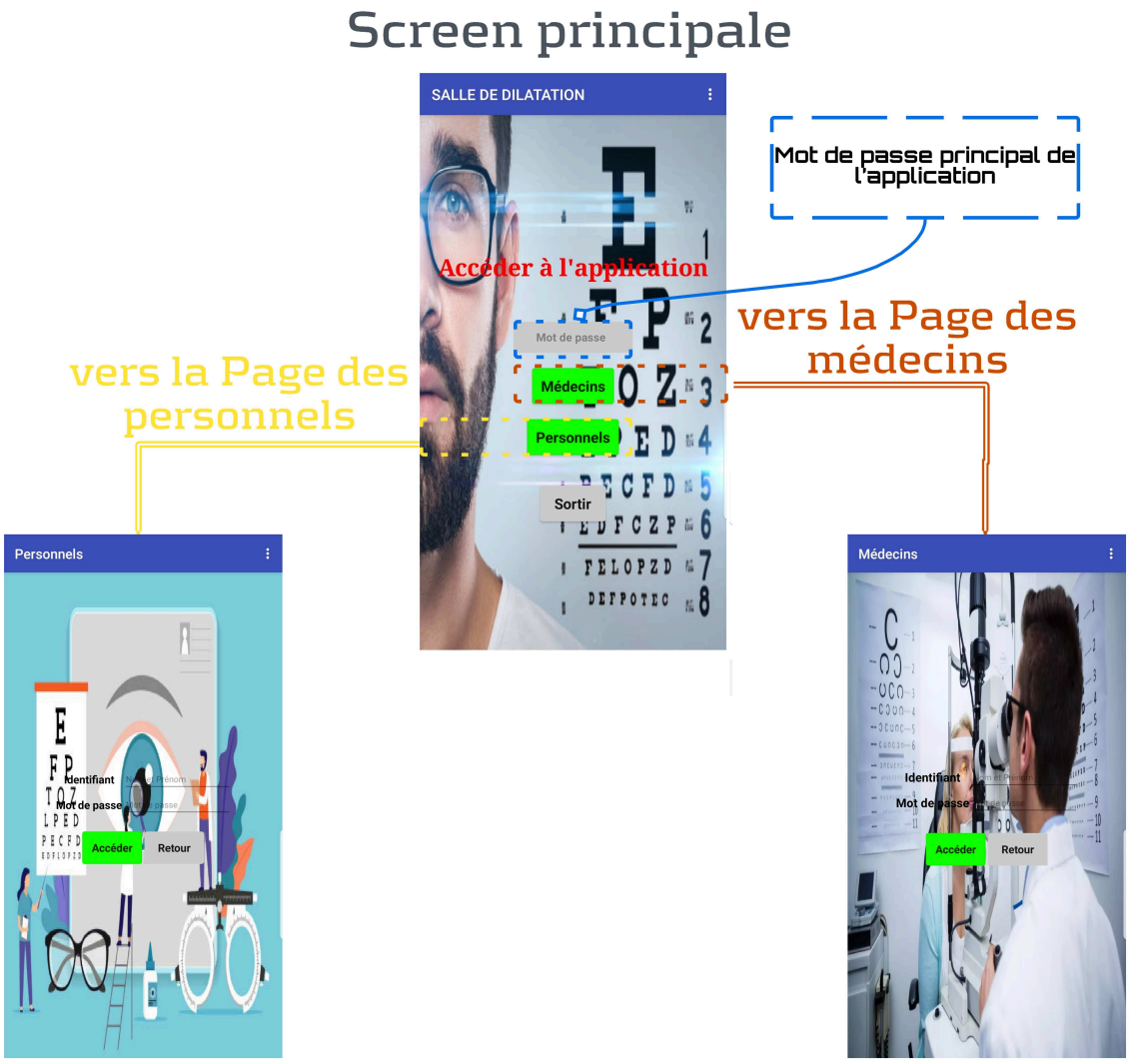


FIG. 3.3 : Les trois fenêtres principales d'accès à l'application et aux données patients.

Personnels

Dans la fenêtre principale du personnel, nous avons accès au nombre de places disponibles pour tous les médecins. A partir de là, il s'agira d'introduire le nom du patient en question avec le type de collyre et cliquer sur ajouter.

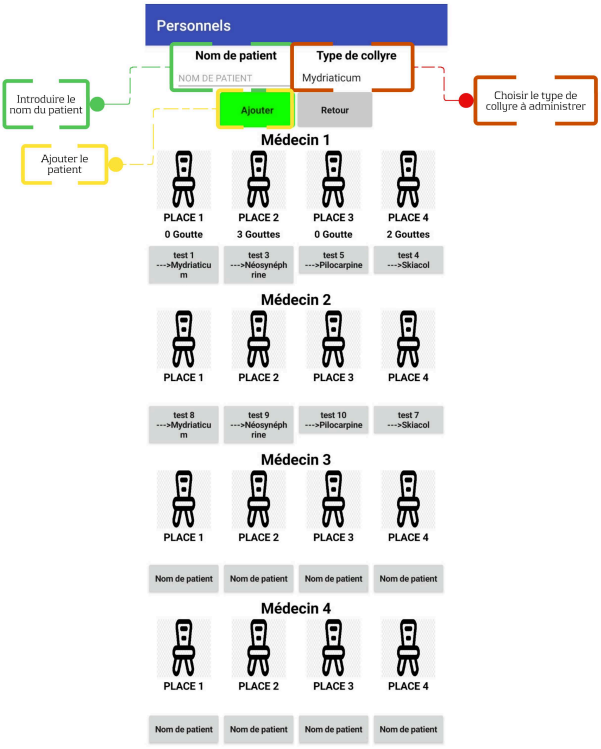


Fig. 3.4 : La fenêtre principale des personnels.

En cliquant sur ajouter, une notification apparaît nous demandant de choisir le médecin traitant du patient, il suffira alors de sélectionner le bon médecin.

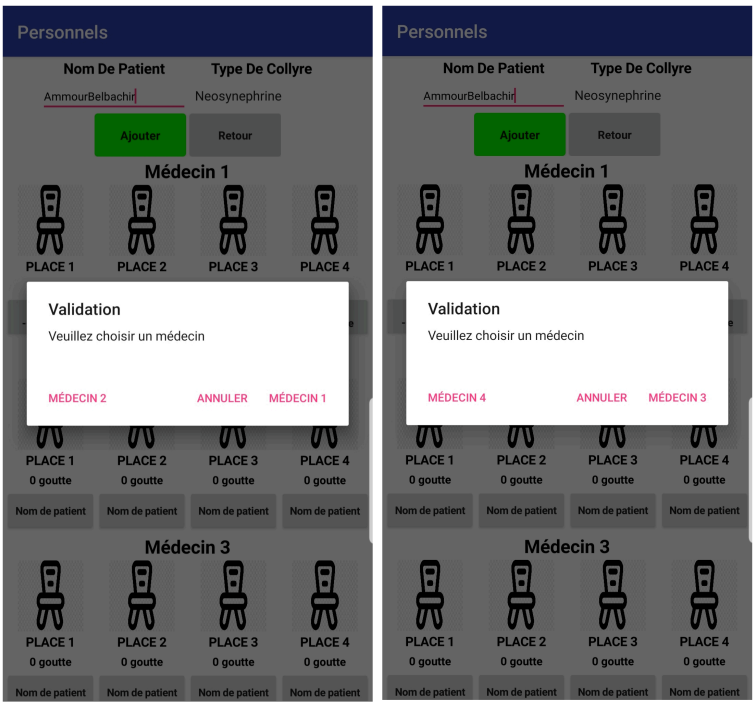


Fig. 3.5 : Notification de validation du médecin traitant du patient.

Après avoir choisi le médecin, nous allons dans l'application et sélectionner la case nom du patient dans la place ou nous souhaitons faire asseoir le patient, une liste alors

apparaît afin de sélectionner le patient en question.

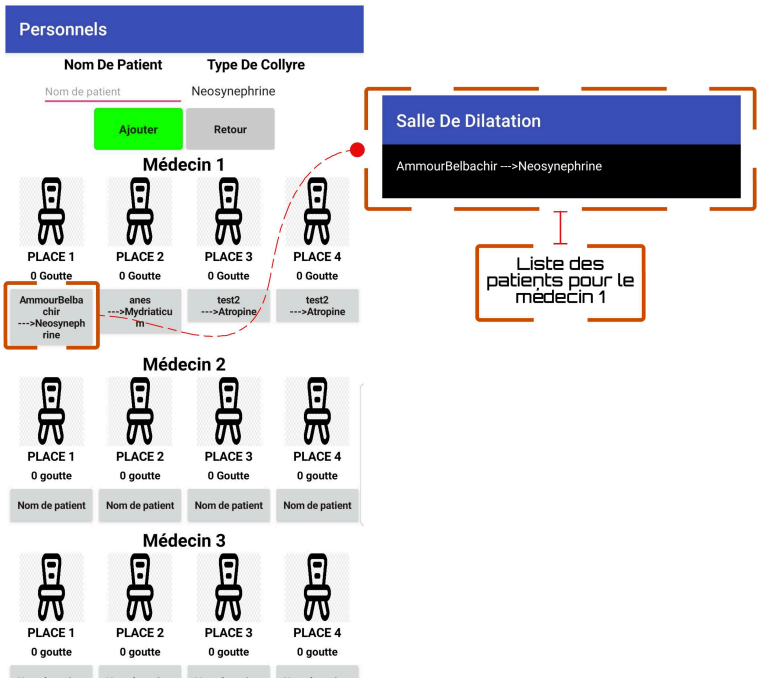


FIG. 3.6 : Choix de la place ou sera assis le patient.

L'incrémentation peut alors commencer tout simplement en cliquant sur l'icône représentant une chaise. Une notification de confirmation alors apparaît, en cliquant sur oui, on commence à incrémenter le nombre de gouttes administrées au patient.

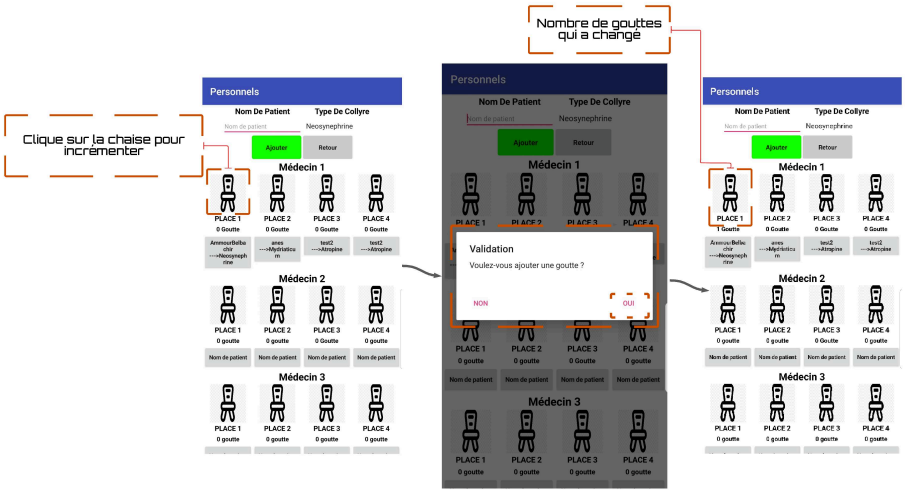


FIG. 3.7 : Notification de confirmation d'ajout d'une goutte.

Médecins

Dans la fenêtre principale de chaque médecin, apparaissent les places ainsi que les patients les occupants. Après avoir effectué le fond d'œil, le médecin devra alors cliquer sur le bouton supprimer, la liste de tous ses malades apparaît, il devra ensuite sélectionner le patient et le supprimer de la base de données.

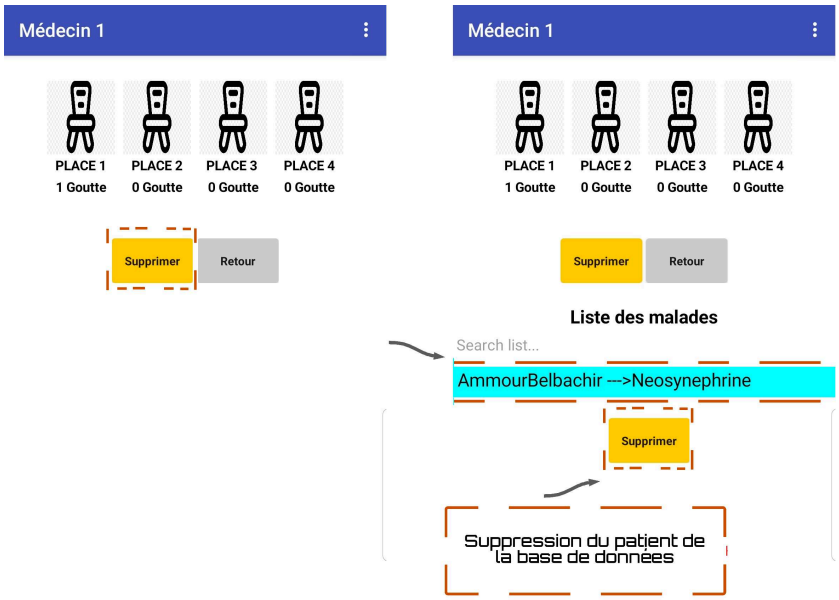


FIG. 3.8 : La liste des malades dans la fenêtre du médecin.

3.3.2 Partie Blocks

Les différents blocs de programmes composant notre application Android sont présentés dans ce qui suit.

Partie 1

Cette partie est dédiée à la vérification du mot de passe principal de l'application dans un premier temps.Dans un deuxième temps, elle aura pour rôle de nous rediriger vers une autre fenêtre qui sera celle du personnel ou médecins en fonction du choix effectué.

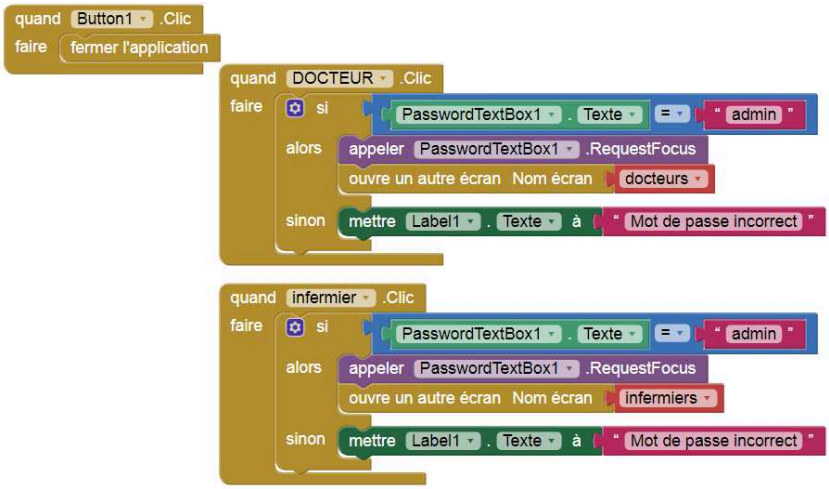


FIG. 3.9 : Block de mot de passe principal de l'application.

Partie 2

Étant donné que les identifiants des médecins et personnels médical sont enregistrés dans la base de données Firebase, cette partie consistera à récupérer ces informations afin de les utiliser pour vérifier si les identifiants et mots de passes introduits sont corrects.

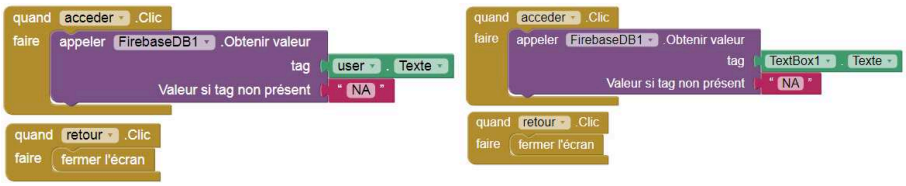


Fig. 3.10 : Block de vérification des identifiants et des mots de passe.

Partie 3

Cette partie vient compléter la partie précédente, car c'est dans cette partie qu'on pourra vérifier si les identifiants et mots de passes introduits correspondent à ceux récupérés à partir de la base de données Firebase.

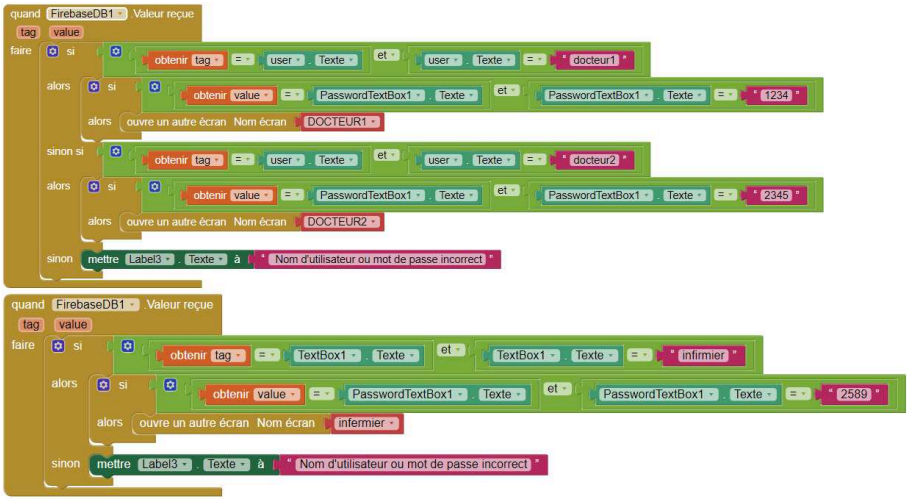


Fig. 3.11 : Block de vérification de la compatibilité des identifiants.

Partie 4

Cette partie permet d'initialiser une liste de malades vide afin qu'elle puisse être remplie ultérieurement.

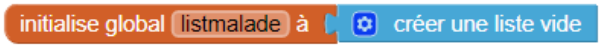


Fig. 3.12 : Initialisation d'une liste de malades.

Partie 5

Ici, il s'agira de récupérer la liste des patients du médecin en question (ici par exemple médecin 1), tout en évitant de l'afficher. Cette liste s'affichera dès que l'on appuiera sur le bouton supprimer comme présenté dans la partie interface de l'application.

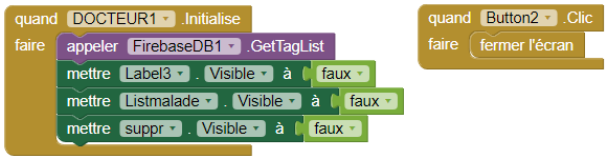


FIG. 3.13 : Block de récupération de la liste des patients du médecin.

Partie 6

Ici, pour le premier bloc, on récupère la liste des malades à partir de la base de données Firebase en indexant chaque ligne de cette liste. Pour le deuxième bloc, il s'agira de mettre à jour cette liste des malades à chaque changement. Dans ce cas, nous récupérons la liste d'un seul médecin. Pour récupérer celle des autres médecins, on reprend les mêmes blocs en faisant appel à chaque fois à un autre FirebaseDB (par exemple pour médecin 2, on récupère FirebaseDB2).

FIG. 3.14 : Blocks de récupération et de la mise à jour de la liste des patients .

Partie 7

Le premier bloc permet la récupération de l'état de chaque bouton'stat, stat2, stat3, stat4'affecté à chaque place (Application ou carte NodeMCU). L'état de ces boutons change à chaque fois qu'il y a action sur l'application ou bien la carte NodeMCU. Pour le deuxième bloc, on récupère l'état du bouton (Ici un seul exemple qui est le bouton 'stat') et on modifie l'affichage du nombre de gouttes à chaque action sur ce même bouton.Le même bloc est reproduit pour les autres boutons.

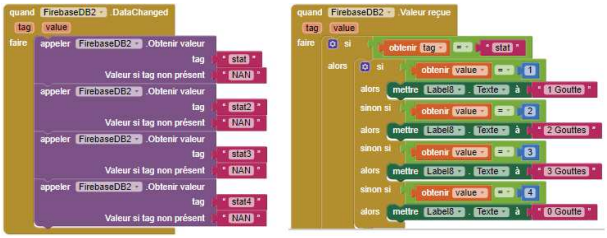


FIG. 3.15 : Blocks de récupération l'état de bouton 'stat' .

Partie 8

Cette partie est dédiée à l'affichage de la liste des malades dans la fenêtre du médecin en question et ce après un simple appuie sur le bouton supprimer. Le médecin pourra à partir de cette liste supprimer le patient ayant terminé sa consultation en appuyant sur un deuxième bouton supprimer. Si nous effectuons un appuie long sur le bouton supprimer, alors cette liste deviendra invisible.

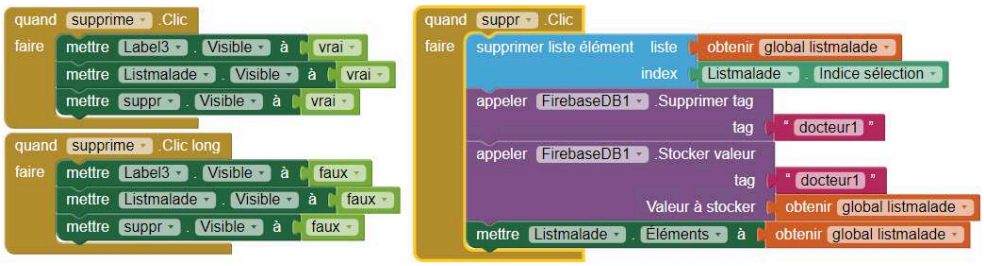


FIG. 3.16 : Blocks d'affichage de la liste des malades dans la fenêtre du médecin.

Partie 9

Dans un premier temps, une initialisation des variables représentant les boutons d'incrémentation du nombre de gouttes et les listes de malades pour chaque médecin (dans cet exemple nous avons pris en considération uniquement deux médecins). Dans la deuxième partie, pour chaque médecin, on récupère le nom du patient à partir de la liste des malades. Ici, ce bloc est dédié à un seul médecin avec quatre patients(places). Le même bloc est reproduit en fonction du nombre de médecin dont nous disposons (dans notre travail quatre médecins).

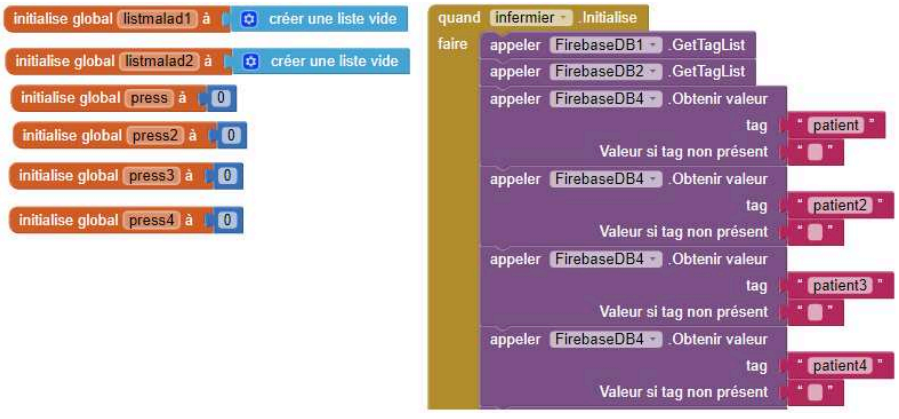


FIG. 3.17 : Blocks d'initialisations et récupération des variables des boutons et les listes de malades.

Partie 10

Dans cette partie, on programme le bouton 'ajouter' qui sert à affecter un patient à un médecin donné. En cliquant sur ce bouton, une notification de confirmation s'affiche permettant d'éviter de se tromper sur le médecin auquel a été affecté le malade. La

deuxième partie sert à rajouter le patient dans la liste des malades affectée au médecin en question.

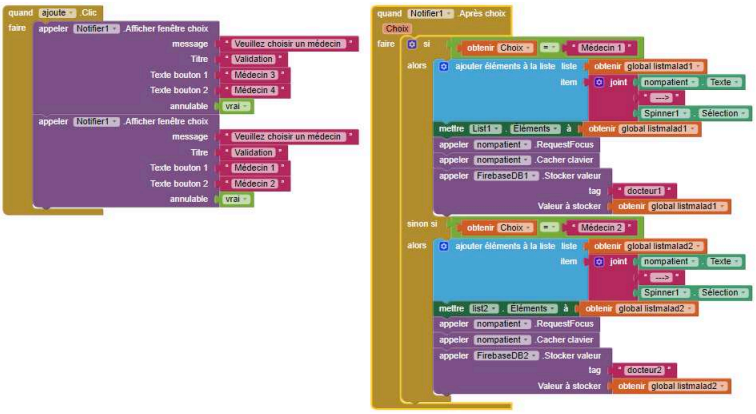


Fig. 3.18 : Blocks de programmation du bouton ajouter et la notification de choix du médecin.

Partie 11

Dans cette partie, on programme le clic sur l'image pour lancer l'incrémentation. Après ce clic, une notification de confirmation est affichée afin de valider l'opération. Ensuite, après validation, l'état du bouton 'stat' dans cet exemple change et par conséquent le nombre de gouttes est actualisé.

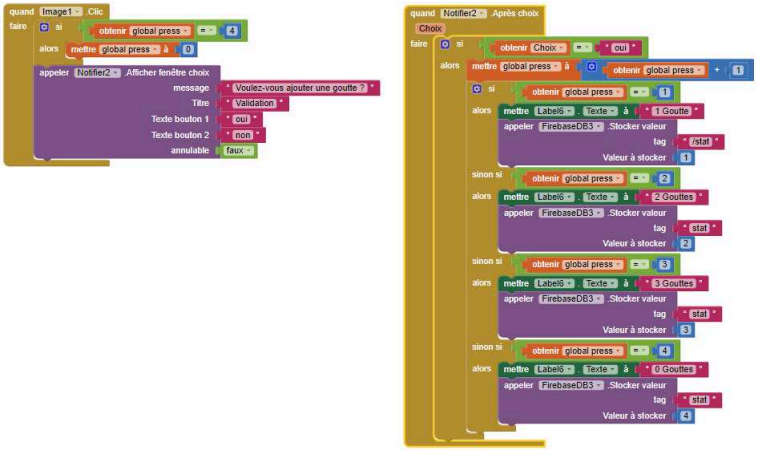


Fig. 3.19 : Blocks d'incrémentation avec un clic sur l'image .

Partie 12

Cette partie permet de rapporter les valeurs des boutons 'stat' à partir de Firebase. Le deuxième bloc permet quant à lui de vérifier l'état de ces boutons 'stat' et afficher le nombre de gouttes en fonction de sa valeur.

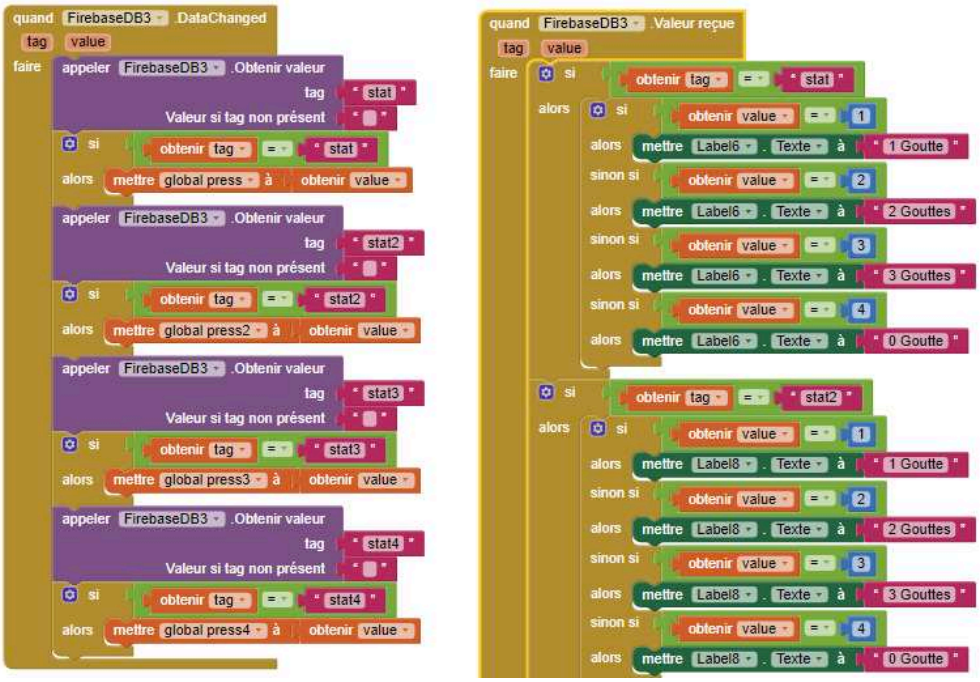


FIG. 3.20 : Blocks de rapportation et vérification des valeurs du 'stat' .

Partie 13

le premier bloc permet d'obtenir la liste des patients d'un médecin (ici médecin 1) à partir de la base de données Firebase. Le deuxième bloc va servir à afficher le nom du patient sélectionné et stocker cette information dans la base de données Firebase. Le troisième bloc quant à lui permet de récupérer le nom du patient choisi à partir de la base de données Firebase.

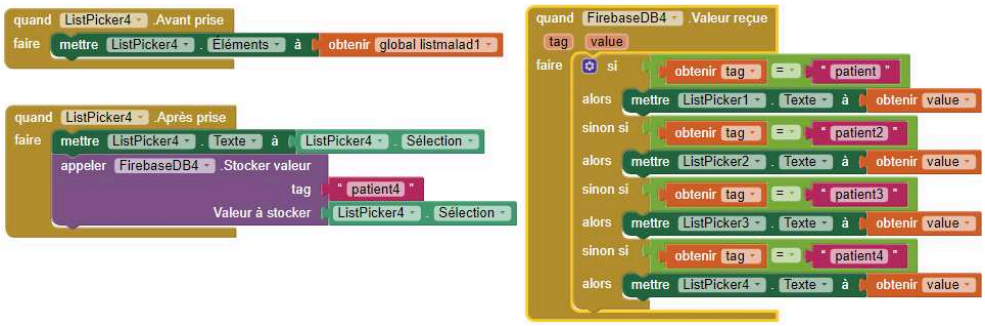


FIG. 3.21 : Blocks de rapportation-affichage-récupérations de la liste des patients d'un médecin .

3.4 Base de données Firebase

La base des données Firebase va nous servir dans notre cas à stocker les identifiants et mots de passe créés, les listes des patients ainsi que l'état des boutons d'incrémentation de nombre de gouttes.

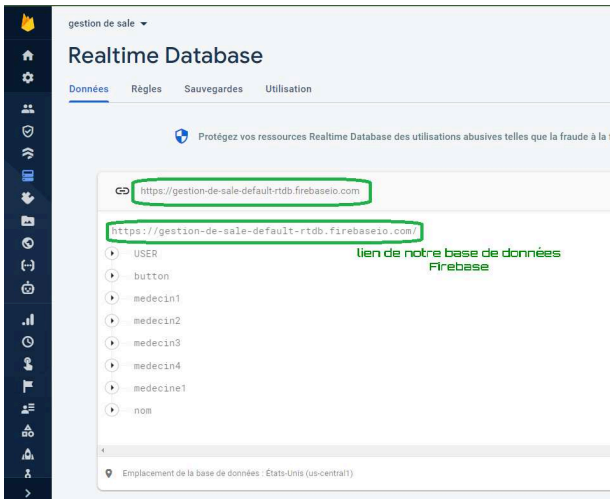


FIG. 3.22 : Le lien de notre base de données Firebase.

Dans la figure 3-22, nous pouvons voir le lien de notre base de données Firebase, ainsi que les parties suivantes :

Section USER

Cette partie est dédiée à l'enregistrement des identifiants de chaque médecin ainsi que le personnel médical.

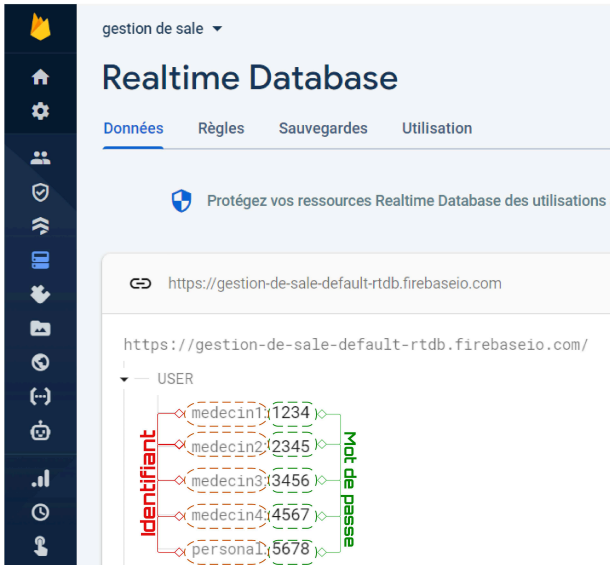


FIG. 3.23 : Les identifiants et les mots de passe créés dans la base de données Firebase.

Section button

Cette partie est dédiée à la récupération de l'état des boutons d'incrémentation du nombre de gouttes, que ça soit par application ou bien par boutons poussoirs.

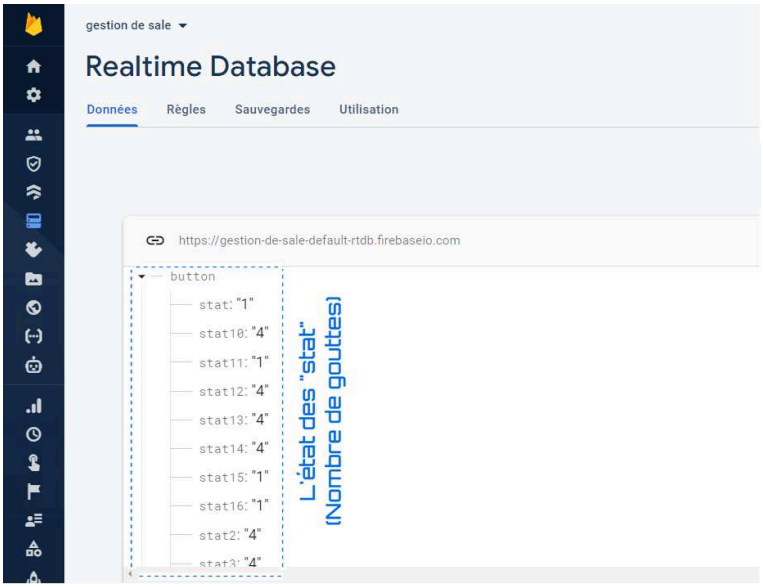


FIG. 3.24 : L'état des boutons d'incrémentation créés dans la base de données Firebase.

Section medecin

Cette section sert à stocker la liste des patients affectés à chaque médecin.

FIG. 3.25 : Les listes des patients de chaque médecin créés dans la base de données Firebase.

Section nom

Elle sert à stocker le nom du patient avec le type de collyre.

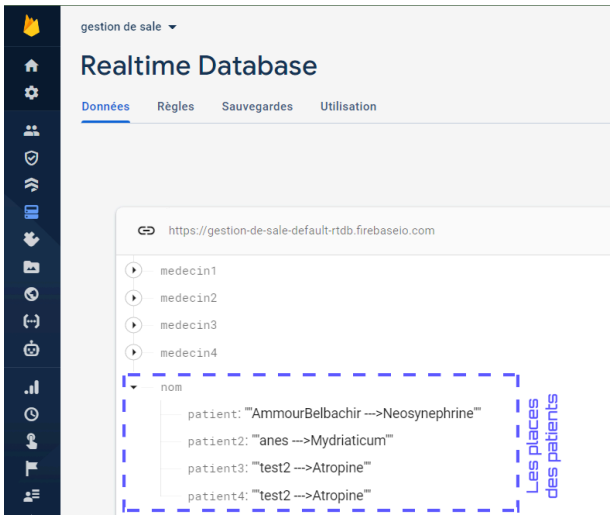


FIG. 3.26 : Les listes de nom du patient avec le type de collyre créés dans la base de données Firebase.

L'URL et le code d'authentification (figure 3.27) de notre base de données Firebase doivent être insérés dans le programme Arduino et dans l'application Android que nous avons créé.

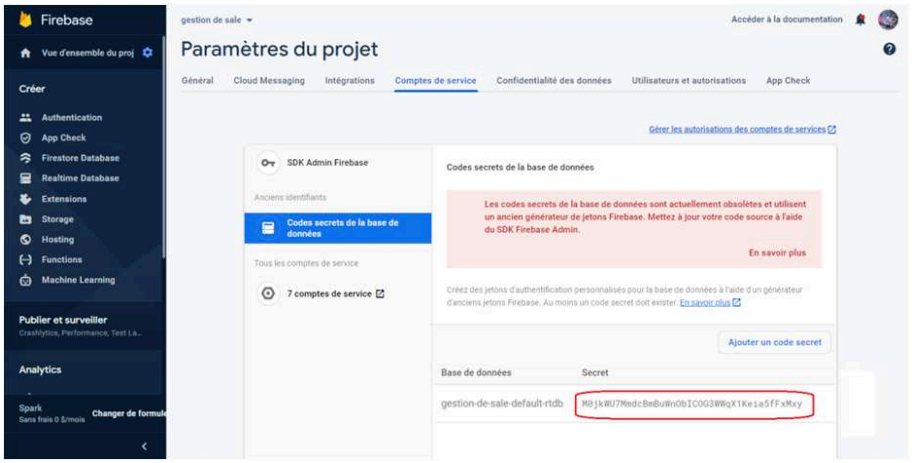


FIG. 3.27 : Code d'authentification de notre base de données Firebase.

3.5 Circuit électronique de notre système de gestion de salle de dilatation

Comme nous l'avons vu dans le chapitre précédent, la partie électronique de notre système de gestion de salle de dilatation se compose principalement d'une carte NodeMCU ESP8266 intégrant un module Wifi avec laquelle on vient brancher un bouton d'incrémentation de nombre de gouttes, trois LEDs (une LED pour chaque goutte) et un buzzer et ce pour chaque patient. Un afficheur OLED I2C est rajouté à la carte NodeMCU et va servir à afficher pour chaque patient le nom, le type de collyre, combien de goutte il a reçu et à quelle place il est assis. Cet afficheur peut gérer jusqu'à 4 places (4 patients).

Après chaque application de collyre, le personnel vient cliquer sur le bouton poussoir branché à la carte NodeMCU (ou bien au niveau d'application). Le nombre de gouttes est alors incrémenté et une première LED vient s'allumer. Passé un certain temps et quand vient le moment de rajouter une deuxième goutte, le buzzer commence à émettre un son et la LED qui était allumée commence à clignoter.

Ceci a pour but de rappeler au personnel médical parfois dépassé par le travail de rajouter une goutte au patient. Une fois cette deuxième goutte administrée, le personnel vient appuyer encore une fois sur le bouton d'incrémentation, le buzzer arrête d'émettre un son, la LED qui clignotait se rallume et une deuxième LED s'allume à son tour. La même opération est reproduite pour la troisième goutte.

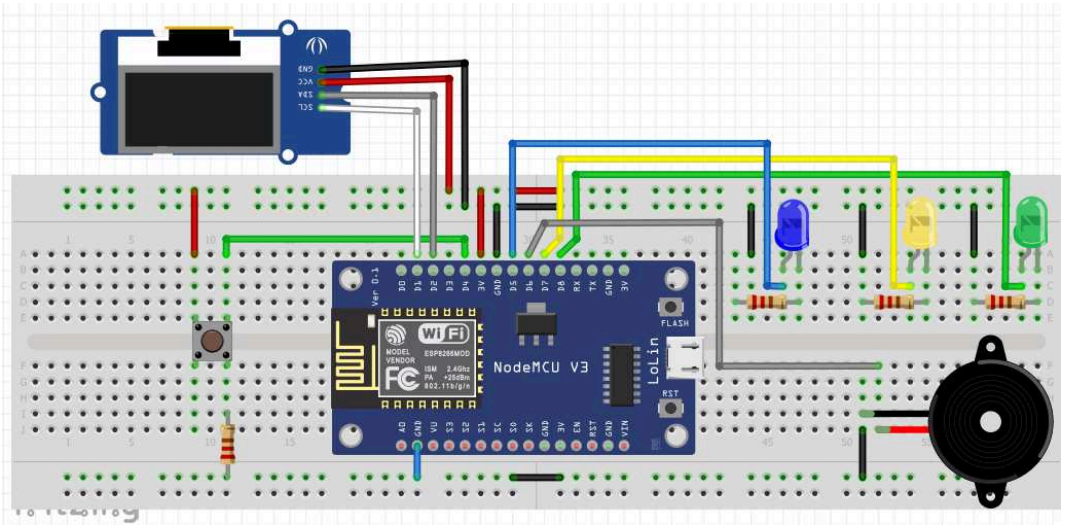


Fig. 3.28 : Circuit proposé pour notre système de gestion de salle de dilatation.

3.5.1 Programmation de la carte NodeMCU

Afin de faire fonctionner le circuit correctement, il est nécessaire d'injecter un programme par le biais de l'IDE Arduino à la carte NodeMCU.

Afin de mieux cerner le programme Arduino élaboré pour notre système, nous l'avons divisé en plusieurs parties. Chaque partie sera expliquée dans ce qui suit.

Partie 1

Cette partie qui est le début de notre programme Arduino a pour but la déclaration de toutes les bibliothèques nécessaires au bon fonctionnement de notre système.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <ArduinoJson.h>
```

FIG. 3.29 : Déclaration des bibliothèques

Partie 2

Cette partie permet de définir trois informations capitales qui sont : les caractéristiques de notre écran OLED, le lien et le code d'authentification Firebase afin que la carte NodeMCU puisse y avoir accès et enfin le nom et mot de passe du réseau WiFi nécessaires pour la connection de la carte NodeMCU à la base de données Firebase

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define FIREBASE_HOST "gestion-de-sale-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "M0jkWU7MmdcBmBuWnObICOG3WWqX1Keia5fFxMxy"
#define ssid "B_M_ANES"
#define password "A_OUSSAMA"
```

FIG. 3.30 : Partie de définition d'écran OLED et Firebase.

Partie 3

Ici, il s'agira de synchroniser et stocker en temps réel nos données sur le serveur Firebase. En plus, nous faisons appel à la bibliothèque « Adafruit » dans le but d'initialiser les dimensions de l'afficheur (longueur et largeur) avec les caractéristiques déjà définies.

```
FirebaseData firebaseData;
FirebaseJson json;
Adafruit_SSD1306 display (SCREEN_WIDTH , SCREEN_HEIGHT);
```

FIG. 3.31 : synchronisation et stockage dans Firebase-initialisation des dimensions de l'OLED.

Partie 4

Nous déclarons ici les différentes variables qui seront utilisées ultérieurement dans notre programme.

```
int pressed=0;
int load;
String nom ,nom2,nom3,nom4;
int buz;
```

FIG. 3.32 : Déclaration des variables.

Partie 5

Dans Cette partie, nous allons initialiser et démarrer l'affichage via notre écran OLED. Ensuite, il s'agira de définir les broches sur lesquelles sont branchées les différentes LEDs, buzzer, et boutons poussoirs.

```
void setup() {
  display.begin (SSD1306_SWITCHCAPVCC, 0x3C);
  Serial.begin(115200);

  pinMode(D4, INPUT);//Bouton poussoir
  pinMode(D8, OUTPUT);digitalWrite(D8 ,LOW);//led verte
  pinMode(D7, OUTPUT);digitalWrite(D7 ,LOW);//led jaune
  pinMode(D5, OUTPUT);digitalWrite(D5 ,LOW);//led bleu
  pinMode(D6, OUTPUT);digitalWrite(D6 ,LOW);//buzzer
```

FIG. 3.33 : Démarrage d'affichage et déclaration des pins.

Partie 6

Cette partie permet de lancer la connexion entre le réseau Wifi que nous avons déjà déclaré et la carte NodeMCU, avant de se connecter à la base de données Firebase

```
WiFi.mode(WIFI_STA);
WiFi.begin(ssid , password);
Serial.print("connecting....");
while (WiFi.status() != WL_CONNECTED){
  delay(300);
  Serial.print(".");
}
Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print(" IP Adress: ");
Serial.println(WiFi.localIP());
Firebase.begin(FIREBASE_HOST,FIREBASE_AUTH);
Firebase.reconnectWiFi(true);}
```

FIG. 3.34 : La connexion entre Wi-Fi et le NodeMCU.

Partie 7

Cette partie est faite spécialement pour la déclaration de fonctions que nous avons nous même créé.

- **Void checkpress :** Cette fonction est dédiée à la vérification de l'état du bouton poussoir et incrémentation.
- **Void firebase :** Cette fonction sert à récupérer l'état du bouton depuis Firebase.
- **Void firebasnompatient :** Cette fonction sert à récupérer les noms des patients (dans cet exemple,nous traitons 4 patients).

```
void checkpress(){

    byte readbutton = digitalRead(D4);
    delay(50);// fonction millis pour debounce.
    if(readbutton== HIGH ){   pressed++;   }
    firebase();
    delay(50);}

 void firebase(){

      if(Firebase.get(firebaseData,"/button/stat")){
        if (firebaseData.dataType() == "string") {
          load = firebaseData.stringData().toInt();
          Serial.println("button stat :");
          Serial.println(load);}}}
```

```
void firebasnompatient(){

    if(Firebase.get(firebaseData,"/nom/patient")){
      if (firebaseData.dataType() == "string") {
        nom = firebaseData.stringData();
        Serial.println("nom de patient :");
        Serial.println(nom);}}

      if(Firebase.get(firebaseData,"/nom/patient2")){
      if (firebaseData.dataType() == "string") {
        nom2 = firebaseData.stringData();
        Serial.println("nom de patient2 :");
        Serial.println(nom2);}}

      if(Firebase.get(firebaseData,"/nom/patient3")){
      if (firebaseData.dataType() == "string") {
        nom3 = firebaseData.stringData();
        Serial.println("nom de patient3 :");
        Serial.println(nom3);}}

      if(Firebase.get(firebaseData,"/nom/patient4")){
      if (firebaseData.dataType() == "string") {
        nom4 = firebaseData.stringData();
        Serial.println("nom de patient4 :");
        Serial.println(nom4);}}}
```

FIG. 3.35 : Les Fonction checkpress, firebase et firebasnompatient.

- **Void affich :** Cette fonction a pour rôle l'envoie à l'écran OLED du nom du patient, type de collyre, nombres de gouttes et la place ou il est assis.

```
void affich(){
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(20,00);
  display.print("place 1");
  display.setTextSize(1);
  display.setCursor(0,20);
  display.print(nom);
  display.setTextSize(2);
  display.setCursor(15,40);
  display.print(pressed);
  display.print(" goutte");
  display.display();}
```

FIG. 3.36 : La Fonction affich.

Partie 8

Dans cette partie, il s'agira de tester l'état des variables 'pressed' (bouton poussoir) et 'load' (l'état du bouton stat dans l'application), le nombre de goutte et le nom de patient sont affichés. Cette information est envoyée à la base de données Firebase pour y être stockée. Ensuite, après avoir administré la première goutte, quand arrive le moment de mettre la deuxième, le buzzer commence à émettre un son et la LED de la première goutte commence à clignoter jusqu'à ce qu'on administre la deuxième goutte et par conséquent on incrémente (par application ou bien bouton poussoir) pour passer à l'étape suivante et allumer la deuxième LED tout en laissant la première LED allumée. La même opération est effectuée pour la troisième goutte.

```
void loop() {
    checkpress();
    firebasnompatient();
    affich();

    if(pressed == 1 || load == 1) {
          pressed=1;
          Firebase.setString(firebaseData,"/button/stat",pressed);
          firebasnompatient();
          affich();
          digitalWrite(D8,HIGH);
          Serial.println("led vert on");
          delay(3000);//temps necessaire pour delaté
          for(buz=0; buz<1000; buz++) {

          checkpress();
          if(pressed==2 || load == 2){buz=1000;}
          tone(D6, 6000, 250);
          delay(200);
          noTone(D6);
          digitalWrite(D8,LOW);
          delay(300);
          digitalWrite(D8,HIGH);}}
```

FIG. 3.37 : L'état des variables 'pressed' et 'load' pour quelles sont égale à 1.

Partie 9

Dans cette partie, en cliquant sur le bouton (application ou bouton poussoir) pour incrémentation pour la quatrième fois (pressed=4 ou load=4),nous reprenons un cycle depuis le début. Ce qui veut dire que 'pressed' et 'load' deviendront égale à 0 et toutes les LEDs s'éteignent indiquant que le nombre de goutte est égal à zéro (0).

```
if(pressed == 4 || load == 4){
    pressed=4;
    Firebase.setString(firebaseData,"/button/stat",pressed);
    pressed=0;
    firebasnompatient();
    affich();
    noTone(D6);
    digitalWrite(D5,LOW);
    digitalWrite(D7,LOW);
    digitalWrite(D8,LOW);
    Serial.println("all leds off ");}}
```

FIG. 3.38 : L'état des variables 'pressed' et 'load' pour quelles sont égale à 4.

3.6 Tests pratique de notre système de gestion de salle de dilatation

Dans cette partie, il s'agira de démontrer le bon fonctionnement de notre système de gestion de salle de dilatation.

Avant de commencer ces tests, nous avons commencé par câbler notre carte NodeMCU. Le câblage de cette carte a été effectué pour un seul médecin pour vérifier le fonctionnement et la synchronisation des données avec l'application **'DRM App'** et la base de données Firebase. Ainsi, le bouton poussoir est branché sur la broche D4. Les LEDs sont branchées sur les broches D5, D7, D8 respectivement pour la LED bleu, jaune et verte. Le buzzer est branché sur la broche D6 de la carte NodeMCU. Enfin, Les broches de l'écran OLED : SDA, SCL, GND, VCC sont quant à elles respectivement branchées sur les broches : D2, D1, GND, VCC de la carte NodeMCU.

Notons par ailleurs que l'alimentation de la carte NodeMCU provient du port USB de cette dernière.

3.6.1 Démarches suivies pour effectuer ces tests et résultats obtenus

La première étape par laquelle nous devons commencer est d'abord le lancement de l'application Android **'DRM APP'** et introduire le mot de passe principal de l'application. Après, il faudra choisir soit médecin soit personnel comme le montre la figure 3-39.

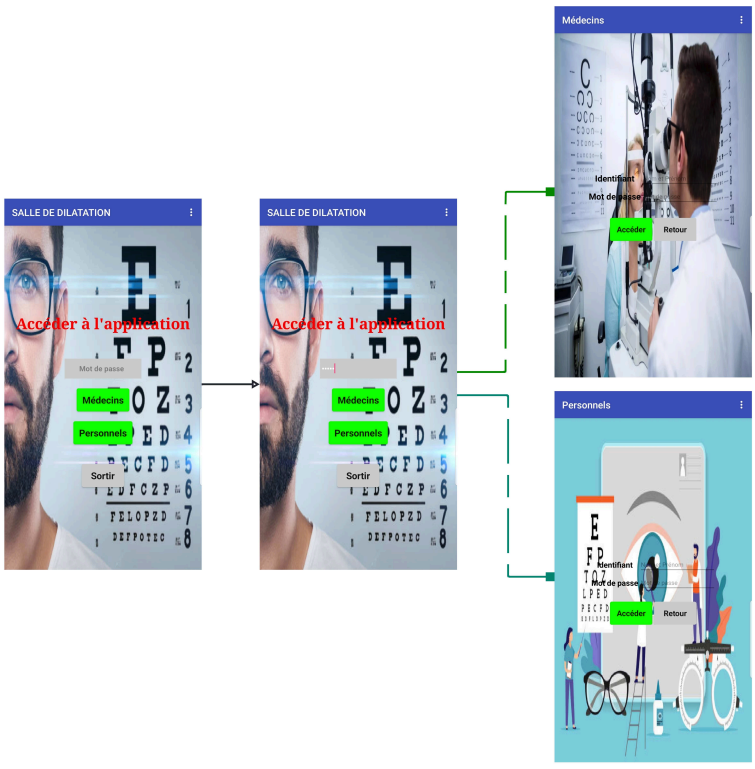


FIG. 3.39 : Interface de choix médecins/personnels.

Utilisation du compte 'Personnels'

Nous allons dans un premier temps travailler avec le compte 'Personnels'. Pour cela, sur la fenêtre 'Personnels' affichée précédemment, nous allons introduire l'identifiant et mot de passe du personnel soignant. Une fois cette opération effectuée, nous arrivons sur la fenêtre de la figure 3-40 suivante :

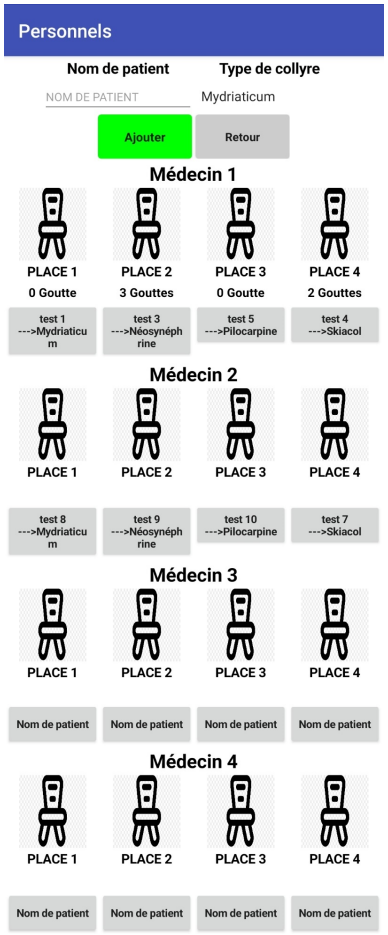


FIG. 3.40 : La fenêtre du personnels.

Dans la section 'nom de patient' et 'type de collyre', nous introduirons le nom du patient et le type de collyre à lui administrer selon les recommandations du médecin. Par exemple, patient nommé 'lazouni' pour qui nous allons administrer des gouttes d'atropine (figure 3-41.)

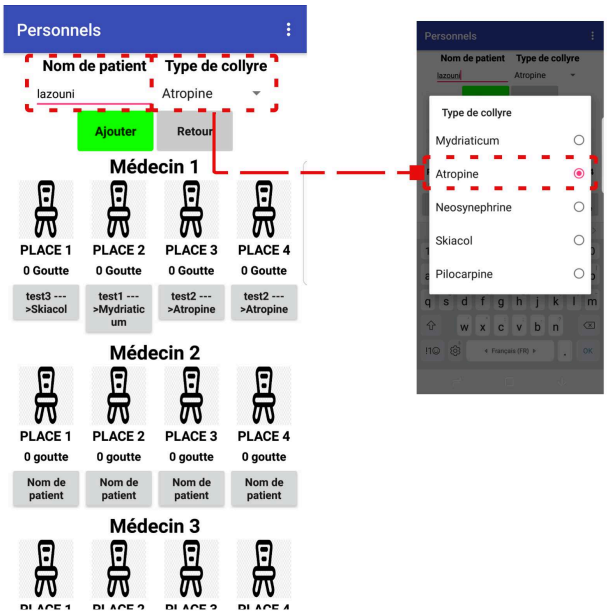


Fig. 3.41 : Introduction du nom de patient et le type de collyre à lui administrer.

En cliquant sur ajouter, une notification apparaît nous demandant de valider le médecin traitant du patient comme le montre la figure 3-42.

Fig. 3.42 : Notification de choix du médecin.

Remarque : la notification qui s'affiche ici elle pour quatre médecins, en fait nous avons deux notifications superposées, en cliquant sur annuler sur la première notification, la deuxième notification s'affiche. Ce problème est du à la limitation de l'option 'notification de MIT APP Inventor à trois boutons seulement. En choisissant médecin 1, le patient 'lazouni' est automatiquement rajouté à la liste des patients du médecin 1.

Il suffira alors de le faire asseoir sur une place vide, place 1 par exemple, pour cela on clique sur le bouton sous l'icône place 1 (encadré en rouge), la liste de tous les patient du médecin 1 alors apparaît, il suffire alors de sélectionner le patient 'lazouni'.

Fig. 3.43 : Affectation d'une place au patient.

A ce moment-là, l'incrémentation peut commencer à partir du moment où le patient recevra la première goutte. Pour ce faire, il suffit de cliquer sur l'icône en forme de chaise sous lequel est écrit 'PLACE 1' (place occupée par 'lazouni'). A ce moment-là, une notification alors apparaît nous demandant de valider l'opération.

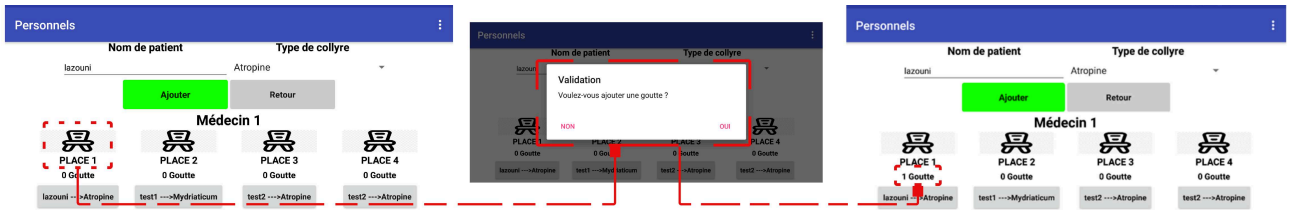


Fig. 3.44 : L'incrémentation et la notification de validation de la première goutte.

Cette même opération d'incrémentation peut être réalisée directement dans la partie électronique de notre système, en appuyant tout simplement sur le bouton poussoir branché à la carte NodeMCU. La carte NodeMCU étant en synchronisation constante avec l'application Android **'DRM App'** et la base données Firebase, la moindre modification dans l'application s'affichera directement au niveau de l'afficheur OLED.

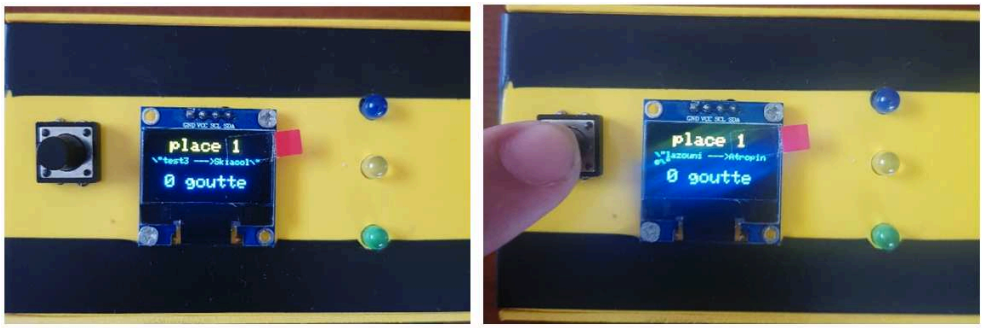


FIG. 3.45 : L'incrémentation depuis le bouton poussoir.

Ainsi, comme le montre la figure 3-46 précédente, dès lors qu'on affecte le patient 'lazouni via l'application Android **'DRM App'**, cette information est directement récupérée au niveau de la carte NodeMCU et affichée. Dès que l'incrémentation commence (bouton poussoir ou application), alors le nombre de gouttes change sur l'afficheur OLED et la première LED est aussitôt allumée.

FIG. 3.46 : Nombre de gouttes sur l'afficheur OLED et la première LED allumée.

Ce qui est intéressant avec la partie électronique, c'est que si le personnel oublie de rajouter une goutte après un certains temps, alors la LED qui était allumée commence à clignoter et le buzzer à emmètre un son. L'afficheur OLED affiche aussitôt un message qu'il faut rajouter une goutte au patient.

FIG. 3.47 : Affichage d'un message afin de notifier le personnel soignant sur la nécessité d'ajouter une goutte au patient.

La même procédure est répétée à chaque fois qu'on ajoute une goutte, jusqu'à ce que le patient revienne chez le médecin.

FIG. 3.48 : Répétition de la procédure d'incrémentation pour la deuxième et troisième goutte.

Utilisation du compte 'Médecins'

Maintenant, en utilisant le compte médecin, ce dernier aura accès à sa liste de patients en salle de dilatation comme le montre la figure 3-49.

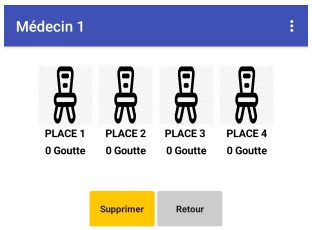


FIG. 3.49 : La fenêtre des médecins (ex : médecin1).

En fait, la seule manipulation allouée au médecin, c'est que quand il termine d'ausculter son patient, il lui suffire de supprimer ce dernier de sa liste des patients. Pour cela, il doit cliquer sur le bouton supprimer. La liste de ses patients alors s'affiche, il lui suffira de sélectionner le patient à supprimer et cliquer sur le deuxième bouton supprimer.

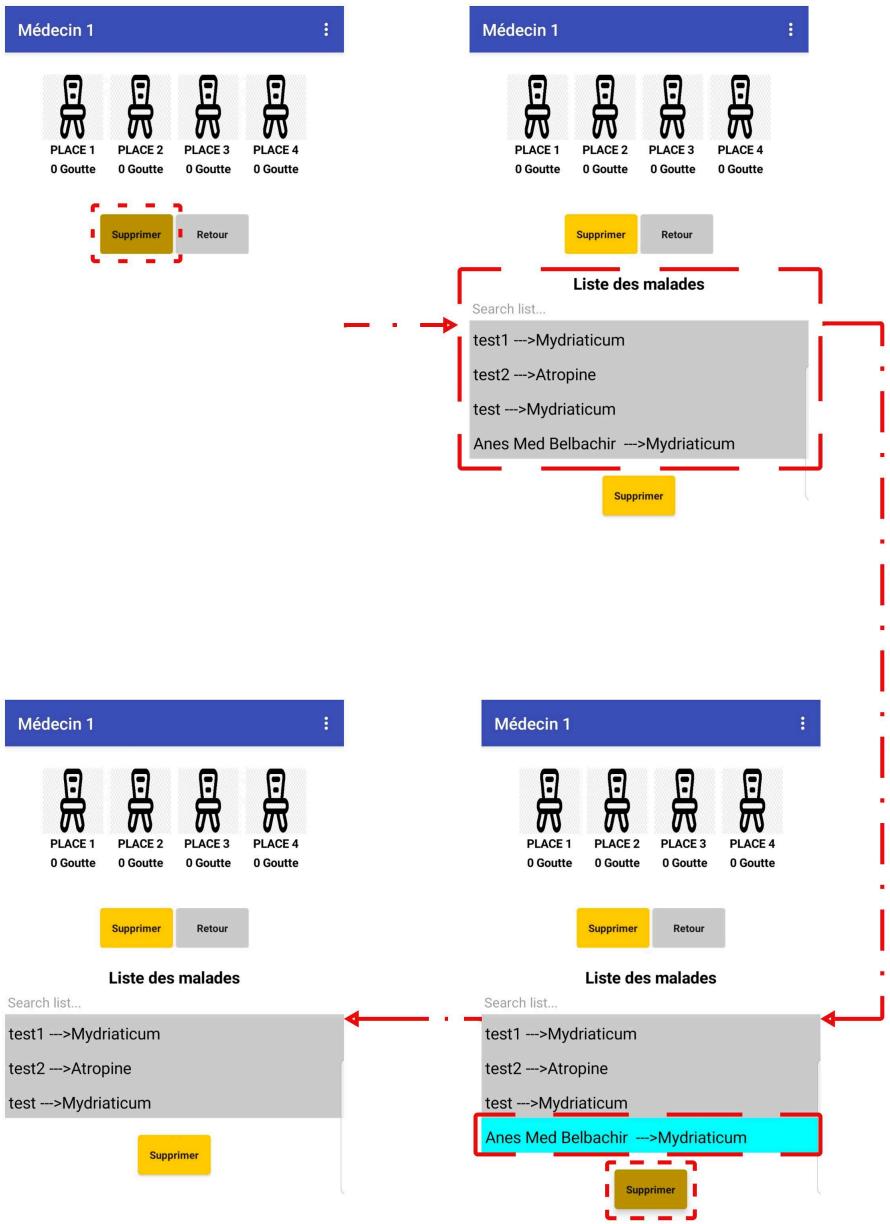


FIG. 3.50 : Les étapes de suppression du patient par le médecin après la fin de la consultation.

3.7 Conclusion

Dans ce chapitre, nous avons présenté en détails le concept de notre système de gestion de salle de dilatation. Nous avons commencé par donner le schéma bloc de fonctionnement de notre système. A partir de là, nous avons entamé la partie application Android **'DRM App'** que nous avons créée ainsi que la base de données Firebase, avant de détailler la partie circuit électronique composant notre système.

Nous avons terminé ce chapitre avec des tests pratiques qui ont montré le bon fonctionnement de notre système de gestion de salle de dilatation et des résultats très prometteurs en vue d'un développement futur plus poussé.

Conclusion et perspectives

Conclusion générale

Le but de ce projet est d'apporter une solution visant à améliorer la gestion d'une salle de dilatation d'une clinique d'ophtalmologie et par conséquent améliorer la qualité de service offerte aux patients venant faire une consultation.Pour ce faire, nous avons réalisé un système de gestion de salle de dilatation facile à mettre en place et peux coûteux. Pour mener à bien ce projet, nous nous sommes reposés sur trois composants importants qui sont : Carte électronique NodeMCU intégrant un module WiFi sur laquelle sont branchés des LEDs, buzzer, afficheur OLED, Application Android que nous avons créé et appelé **'DRM App'**et une base de données Firebase.

Pour plus de clarté, nous avons divisé ce travail en trois grandes parties :

Dans **la première partie**, nous avons commencé par présenter un petit historique sur l'ophtalmologie et la dilatation avant de détailler ces derniers. Nous avons ensuite parlé des types de collyres qu'on utilise pour la dilatation. Nous avons fini par présenter la problématique existante dans la clinique avant de donner le concept de notre système de gestion de la salle de dilatation.

Dans **la deuxième partie**, nous avons présenté les différents composants matériels et logiciels qui nous ont été nécessaires afin de mener à bien notre projet. Nous avons commencé par présenter l'Arduino de manière générale avant de décrire la carte NodeMCU intégrant un module Wifi utilisée dans notre projet. Nous avons par la suite parlé des outils de création d'applications MIT App Inventor avant de terminer avec un petit descriptif sur l'outil Firebase de Google.

Dans **la troisième partie**, nous avons présenté le système de gestion de la salle de dilatation tel que nous l'avons conçu.Nous avons commencé par montrer comment nous avons procédé pour créer notre application tout en détaillant l'interface et le programme de cette dernière.Nous avons par la suite présenté les différentes données stockées dans la base de données Firebase, tels que les noms d'utilisateurs, mots de passes, les listes des patients et les boutons. Nous avons par la suite présenté le circuit électronique réalisé ainsi que le programme Arduino permettant de faire fonctionner notre système.Les tests pratiques réalisés ont permis de montrer le bon fonctionnement de notre système quelque soit la manière choisie par le personnel soignant, en passant par l'application Android ou bien en utilisant le circuit électronique réalisé.

Grâce à un tel système, ou les trois composants de notre système sont en communication et synchronisation permanente, on donne plus de flexibilité au personnel soignant ce qui facilite grandement leur travail.

Perspectives

Au cours de ce travail, nous avons pu atteindre tous les objectifs souhaités, en réalisant un système parfaitement fonctionnel. Néanmoins, ce travail peut être amélioré et certains aspects de notre système peuvent être optimisés. A cet effet, nous proposons quelques perspectives.

- Au niveau de circuit électronique :

  On peut augmenter le nombre d'entrées/sorties du NodeMCU en rajoutant un registre à décalage Parallèle/série de type 74HC165. Ce composant peut être monté en cascade, ainsi on peut avoir plusieurs entrées : 16, voire plus.

  On peut ajouter aussi des circuits intégrés capables de multiplier le nombre d'entrées du NodeMCU, ils supportent le protocole SPI ou I2C.

- Au niveau de la base de données :

  Afin de garantir la sécurité de données patientes, nous prévoyons d'éviter le stockage des données dans le cloud et les services en ligne. Pour ce faire, cette base de données peut être stockée sur le réseau local de la clinique.

- Au niveau de l'application Android :

  Rajouter les temporisations avec une amélioration au niveau de notification afin d'indiquer n'importe quelle changement relatif à chaque patient.

Bibliographie

[1] *Ophtalmologie oculus échantillon closeup.* 123RF.com.

[2] *EYEWIKI-History of Ophthalmology Timeline.png.*

[3] *History of Ophthalmology -mrcophth.com/Historyofophthalmology/Introductory.*

[4] *The Susruta-Samhita or Sahottara-Tantra | LACMA Collections.*

[5] J. Bendix. "Study : EHR notes may enhance, prolong racial bias". In : (2 mar. 2022).

[6] B. Boyd et
bibinitperiod Boyd. *Modern ophthalmology : the highlights : the account of a master wintnessing a 60 year epoch of evolution and progress (1950-2010).* Panama : Jaypee-Highlights Medical Publishers, 2010. isbn : 978-9962-678-16-8.

[7] *Fundus oculi examination - Science Photo Library.*

[8] *DILATATION DES PUPILLES EN OPHTALMOLOGIE-ophtalmologie-lariboisiere.*

[9] *Traditional Eye Dilation Drops[bvoptometry].* Bright Vision Optometry | Chino Hills, CA 91709 | Eye Doctors | Optical.

[10] B. B. K. Editor Senior. *Legalizing Optometry.*

[11] herve. *Des gouttes dans les yeux pour mieux voir de près : cette innovation israélienne est très proche de la commercialisation | Telavivre.*

[12] *Pupil dilation : Causes, Symptoms And Treatment.* EyeMantra. 16 mar. 2021.

[13] D. B. Elliott. "Evidence-based eye examinations". In : (), p. 10.

[14] *IS EYE DILATION NECESSARY IN AN EYE EXAMINATION.* Shroff Eye Mumbai. Section : Blog. 30 déc. 2016.

[15] *Eye Drops Images From STOCKADOBE.* Adobe Stock.

[16] *Goutte App-OPENCLINICALSKILLS.*

[18] K. Henning et al. "Routine dilated fundus examination diagnostic yield". In : *Ophthalmology* 105.11 (1er nov. 1998). Publisher : Elsevier, p. 1983-1984. issn : 0161-6420, 1549-4713.

[19] O. D. Benjamin P. Casella. "Better flow for better patient care -optometrytimes". In : (16 fév. 2022).

[20] *What Is Wi-Fi ? - Definition and Types.* Cisco.

[21] *ESP8266-module de communication WIFI-emi.ac.ma/oumnad/ESP8266/ESP8266-* .

[22] *NodeMCU Documentation-nodemcu.readthedocs.io-.*

[23] w. r. s.r.o web revolution. *NodeMCU ESP8266 ESP-12N V1.0 Wifi CP2102 IoT Lua 267 | GM electronic COM.*

[24] *Robotics - 3D Printing - Internet of Things : NodeMCU v1.0 pinout diagram.*

[25] *Basics using Arduino IDE | NodeMCU-electronicwings-/nodemcu/arduino-ide.*

[26] *OLED I2C-nettigoSite-original.JPG.*

[28] G. TRONIC. *Afficheur OLED 0,96" I2C TF052*. GO TRONIC.

[30] *What is Android ? -androidauthority- Here's everything you need to know.* Android Authority. 19 fév. 2022.

Webographie

[17] *Mydriatique-Résultats de rechreche*. VIDAL. Section : Santé. URL : https://www.vidal.fr/recherche.html (visité le 04/05/2022).

[27] *Guide for I2C OLED Display with Arduino | Random Nerd Tutorials*. 23 mai 2019. URL : https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/ (visité le 27/05/2022).

[29] *Pic-Buzzer-core-electronics.jpg*. URL : https://core-electronics.com.au/media/catalog/product/1/6/160-01_1.jpg (visité le 27/05/2022).

[31] *MIT App Inventor | Explore MIT App Inventor*. URL : https://appinventor.mit.edu/ (visité le 26/04/2022).

[32] *Firebase*. Firebase. URL : https://firebase.google.com/ (visité le 26/04/2022).

[33] *Firebase Documentation*. Firebase. URL : https://firebase.google.com/docs?hl=fr (visité le 27/04/2022).

Annexes

Annexe A

Programme Arduino complet.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <ArduinoJson.h>

#define FIREBASE_HOST "gestion-de-sale-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "M0jkWU7MmdcBmBuWnObICOG3WWqX1Keia5fFxMxy"

#define ssid  "B_M_ANES"              // Nom de WI-FI
    #define password  "A_OUSSAMA"   // Mot de passe

FirebaseData firebaseData;
FirebaseJson json;
Adafruit_SSD1306 display (SCREEN_WIDTH , SCREEN_HEIGHT);

int pressed=0, scroll=0;
int load,load2,load3,load4;
String nom ,nom2,nom3,nom4;
int buz;

void setup() {
  display.begin (SSD1306_SWITCHCAPVCC, 0x3C);
  Serial.begin(115200);


  pinMode(D4, INPUT);// button poussoir
  pinMode(D8, OUTPUT);digitalWrite(D8 ,LOW);//led vert
  pinMode(D7, OUTPUT);digitalWrite(D7 ,LOW);//led jaune
  pinMode(D5, OUTPUT);digitalWrite(D5 ,LOW);//led blue
  pinMode(D6, OUTPUT);digitalWrite(D6 ,LOW);//buzzer


  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid , password);

  Serial.print("connecting....");

  while (WiFi.status() != WL_CONNECTED){
    delay(300);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected␣to␣");
```

```
  Serial.println(ssid);
  Serial.print("␣IP␣Adress:␣");
  Serial.println(WiFi.localIP());

   Firebase.begin(FIREBASE_HOST,FIREBASE_AUTH);
   Firebase.reconnectWiFi(true);

 }
     void checkpress();
     void firebase();
     void firebasnompatient();
     void affich();

void loop() {
    checkpress();
    firebasnompatient();
    affich();

    if(pressed == 1 || load == 1) {
          Firebase.setString(firebaseData,"/button/stat",pressed);
          firebasnompatient();
          digitalWrite(D8,HIGH);
          Serial.println("led␣verte␣on");
          for(int i=0; i<10; i++)
          {affich();
          delay(1500);}
          for(buz=0; buz<1000; buz++) {

          checkpress();
          if(pressed==2 || load == 2){buz=1000;}
          tone(D6, 6000, 250);
          delay(200);
          noTone(D6);
          digitalWrite(D8,LOW);
          delay(300);
          digitalWrite(D8,HIGH);
          display.clearDisplay();
                  display.setTextSize(2);
                  display.setTextColor(SSD1306_WHITE);
                  display.setCursor(20,00);
                  display.print("place␣1");
                  display.setTextSize(1);
                  display.setCursor(0,20);
                  display.print(nom);
                  display.setTextSize(1);
                  display.setCursor(10,40);
                  display.print("AJOUTER␣UNE␣AUTRE␣GOUTTE");
                  display.display();
          }

```

```
100           }
101     if(pressed == 2 || load == 2){
102           Firebase.setString(firebaseData,"/button/stat",pressed);
103           firebasnompatient();
104           digitalWrite(D7,HIGH);
105           Serial.println("led jaune on");
106           for(int i=0; i<10; i++)
107           {affich();
108           delay(1500);}
109           for(buz=0; buz<1000; buz++) {
110
111               checkpress();
112               if(pressed==3 || load == 3){buz=1000;}
113               tone(D6, 6000, 250);
114               delay(200);
115               noTone(D6);
116               digitalWrite(D7,LOW);
117               delay(400);
118               digitalWrite(D7,HIGH);
119               display.clearDisplay();
120                       display.setTextSize(2);
121                       display.setTextColor(SSD1306_WHITE);
122                       display.setCursor(20,00);
123                       display.print("place 1");
124                       display.setTextSize(1);
125                       display.setCursor(0,20);
126                       display.print(nom);
127                       display.setTextSize(1);
128                       display.setCursor(10,40);
129                       display.print("AJOUTER UNE AUTRE GOUTTE");
130                       display.display();
131
132           }
133           }
134
135     if (pressed == 3 || load == 3){
136           Firebase.setString(firebaseData,"/button/stat",pressed);
137           firebasnompatient();
138           digitalWrite(D5,HIGH);
139           Serial.println("led blue on");
140           for(int i=0; i<10; i++)
141           {affich();
142           delay(1500);}
143           for(buz=0; buz<1000; buz++){
144
145           checkpress();
146           if(pressed==4 || load == 4){buz=1000;}
147           tone(D6, 6000, 250);
148           delay(200);
149           noTone(D6);
```

```
150           digitalWrite(D5,LOW);
151           delay(600);
152           digitalWrite(D5,HIGH);
153           display.clearDisplay();
154                   display.setTextSize(2);
155                   display.setTextColor(SSD1306_WHITE);
156                   display.setCursor(20,00);
157                   display.print("place␣1");
158                   display.setTextSize(1);
159                   display.setCursor(0,20);
160                   display.print(nom);
161                   display.setTextSize(1);
162                   display.setCursor(10,40);
163                   display.print("AJOUTER␣UNE␣AUTRE␣GOUTTE");
164                   display.display();
165           }
166      }
167
168     if(pressed == 4 || load == 4){
169         pressed=4;
170         Firebase.setString(firebaseData,"/button/stat",pressed);
171         firebasnompatient();
172         noTone(D6);
173         digitalWrite(D5,LOW);
174         digitalWrite(D7,LOW);
175         digitalWrite(D8,LOW);
176         Serial.println("all␣leds␣off␣");
177         pressed=0;
178
179        }
180   }
181 void checkpress(){
182
183     byte readbutton = digitalRead(D4);
184
185     if(readbutton== HIGH ){   pressed++;   }
186     firebas();
187     delay(50);
188   }
189
190  void firebase(){
191
192       if(Firebase.get(firebaseData,"/button/stat")) {
193         if (firebaseData.dataType() == "string") {
194           load = firebaseData.stringData().toInt();
195           Serial.print("button␣stat␣:");
196           Serial.println(load);
197          }
198       }
199
```

```
200       if(Firebase.get(firebaseData,"/button/stat2")) {
201         if (firebaseData.dataType() == "string") {
202           load2 = firebaseData.stringData().toInt();
203           if (load2==4){load2=0;}
204           Serial.print("button stat 2 :");
205           Serial.println(load2);
206          }
207       }
208
209       if(Firebase.get(firebaseData,"/button/stat3")) {
210         if (firebaseData.dataType() == "string") {
211           load3 = firebaseData.stringData().toInt();
212           if (load3==4){load3=0;}
213           Serial.print("button stat 3 :");
214           Serial.println(load3);
215          }
216       }
217   }
218
219  void firebasnompatient(){
220
221       if(Firebase.get(firebaseData,"/nom/patient")) {
222         if (firebaseData.dataType() == "string") {
223           nom = firebaseData.stringData();
224           Serial.print("nom de patient :");
225           Serial.println(nom);
226           }
227       }
228
229         if(Firebase.get(firebaseData,"/nom/patient2")) {
230         if (firebaseData.dataType() == "string") {
231           nom2 = firebaseData.stringData();
232           Serial.print("nom de patient2 :");
233           Serial.println(nom2);
234           }
235         }
236         if(Firebase.get(firebaseData,"/nom/patient3")) {
237         if (firebaseData.dataType() == "string") {
238           nom3 = firebaseData.stringData();
239           Serial.print("nom de patient3 :");
240           Serial.println(nom3);
241           }
242         }
243       }
244
245   void affich(){
246     display.clearDisplay();
247     display.setTextSize(2);
248     display.setTextColor(SSD1306_WHITE);
249     display.setCursor(20,00);
```

```
250     display.print("Place␣1");
251     display.setTextSize(1);
252     display.setCursor(0,20);
253     display.print(nom);
254     display.setTextSize(2);
255     display.setCursor(15,40);
256     display.print(pressed);
257     display.print("␣goutte");
258     display.display();
259     checkpress();
260     delay(1500);
261 
262     display.clearDisplay();
263     display.setTextSize(2);
264     display.setTextColor(SSD1306_WHITE);
265     display.setCursor(20,00);
266     display.print("Place␣2");
267     display.setTextSize(1);
268     display.setCursor(0,20);
269     display.print(nom2);
270     display.setTextSize(2);
271     display.setCursor(15,40);
272     display.print(load2);
273     display.print("␣goutte");
274     display.display();
275     checkpress();
276     delay(1500);
277 
278     display.clearDisplay();
279     display.setTextSize(2);
280     display.setTextColor(SSD1306_WHITE);
281     display.setCursor(20,00);
282     display.print("Place␣3");
283     display.setTextSize(1);
284     display.setCursor(0,20);
285     display.print(nom3);
286     display.setTextSize(2);
287     display.setCursor(15,40);
288     display.print(load3);
289     display.print("␣goutte");
290     display.display();
291     checkpress();
292     }
```