الجمهورية الجزائرية الديمقراطية الشعبية
RépubliqueAlgérienneDémocratiqueetPopulaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement Supérieur et de la Recherche scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Electronique

Option : Electronique des systèmes embarquès

Réf:.....

Mémoire de Fin d'Etudes
En vue de l'obtention du diplôme:

MASTER

Thème

Réalisation d'un système de contrôle autour de Raspberry Pi pour la domotique

Présenté par :
Ben khalfa Ismail
Soutenu le : 23Juin 2018

Devant le jury composé de :

Mr Benelmir Okba.	**MCB**	**Président**
Mr Bkhouche Khaled.	**MCA**	**Encadreur**
Mme Ouarhlent Saloua.	**MAA**	**Examinateur**

Annéeuniversitaire: 2017 / 2018

الجمهورية الجزائرية الديمقراطية الشعبية
RépubliqueAlgérienneDémocratiqueetPopulaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignementSupérieuret de la recherchescientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Electronique
Option : Electronique des systèmes embarquès

Mémoire de Fin d'Etudes
En vue de l'obtention du diplôme:

MASTER

Thème

Réalisation d'un système de contrôle autour de Raspberry Pi pour la domotique

Présenté par :

Ben khalfa Ismail

Avis favorable de l'encadreur :

Bekhouche Khaled

Avis favorable du Président du Jury

Benelmir Okba.

Cachet et signature

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

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Electronique
Option : Electronique des systèmes embarquès

Thème :

Réalisation d'un système de contrôle autour de Raspberry Pi pour la domotique

Proposé par : **Bekhouche Khaled**
Dirigé par : **Bekhouche Khaled**

RESUME

Notre projet de fin d'étude consiste en un système de commande via internet d'une maison intelligente, c'est-à-dire ce qu'on appelle internet des objets (Internet of Things). Dans ce projet on a lié trois dispositif qui différent côté matériel et côté logiciel. Le dispositif de niveau haut est le Smartphone qui intègre l'application Android. Il présente l'interface utilisateur. Le deuxième dispositif de niveau intermédiaire est le Raspberry pi 3 qui permet efficacement de relié l'utilisateur aux différents actionneurs et capteurs existants dans la maison. Cela est fait via les méthodes d'internet client/serveur. Le dernier dispositif de bas niveau est l'arduino qui en recevant des commandes de l'utilisateur via Raspberry et après un certain traitement exécute les tâches programmées. Les informations des capteurs sont transmet aussi de Arduino vers Raspberry via I2C ensuite vers l'utilisateur via internet.

ملخص

يتكون مشروعنا لنهاية الدراسة من نظام تحكم لمنزل ذكي عبر الانترنت أي ما يسمى بانترنت الاشياء,في هذ المشروع قمنابربط ثلاث اجهزة مختلفة ذات لغة برمجة مختلفة,حيث ان جهاز المستوى الاعلى هو الهاتف الذي يحمل تطبيق الاندرويد لتقديم واجهة المستخدم وجهاز المستور الثاني هو الراسبيري باي 3,فهو يربط المستخدم مع الاجهزة المختلفة في المنزل عن طريق الانترنت بستعمال برتكولTCP/IP (خادم / عميل) اما المستوى الثالث والاخير هو الاردوينو, يتلقى اوامر من المستخدم عن طريق الراسبيري باي عبر برتوكول I2C .

Remerciement

Je remercie Dieu Allah le tout puissant de m'avoir donné courage et

Patience, qui m'a permis d'accomplir ce modeste travail.

Je tiens en premier à exprimer ma grande gratitude envers mon

Encadreur Dr.Bekhouche Khaled, qui m'a apporté son aide et ses

Conseils précieux et de m'avoir inspiré ce sujet et suivi de très près sans

Ménager son temps ni si efforts, qu'il trouve dans ce travail mon

Témoignage respectueux.

Je remercie Dr. Benelmir Okba, maitre assistance B au département d'électronique, pour avoir bien voulu accepter de présider le jury.

Je remercie Mme Ouarhlent Saloua, maître de conférences A d'avoir accepté d'examiner ce mémoire et d'être membre du jury.

Je tiens à exprimer mes sincères remerciements pour mes parents, ma famille et toute personne ayant participé de loin ou de prés pour l'aboutissement de ce modeste travail.

Dédicaces

A mon très cher père, et
Ma très chère mère pour sont amour et ses encouragements.
A mes très chères sœurs, et à ma belle sœur.
A mon frère et mon beau frère.
A toute ma famille et mes amis.

BENKHALFAISMAIL

Tables des matières

Chapiter 3 : Réalisation et duscution

Tables des figures

Chapitre 1

Chapitre 2

Chapitre 3

Liste des tables

Liste des abréviations

GPIO	General purposeInput/Output
HDMI	High Difinition Multimedia Interface
IDE	Integrated Development Environment
IdO	Internet des Objets
IoT	Internet of Things
IP	Internet Protocol
LCD	Liquid Crystal Display
LED	Light-EmettingDiod
M2M	Machine to Machine
MAC	Media Access Control
PWM	Pulse Width Modulation
RAM	Random Access Memory
RFID	Radio Frequency IDentification
SSH	Serveur Secure Shell
USB	Universal Serial Bus
WiFi	wireless Fidelity
WSAN	Wireless Sensor and Actuator Network
TIC	Technologie de Information et de Communicatio
RCA	Radio Corporation of America
PIR	Passive Infrared
CPU	Central Processing Unit

Introduction générale

Introduction générale

Les progrès continus des technologies électroniques des systèmes embarquès et des réseaux de capteurs permettent maintenant d'envisager le déploiement de services sécurisés et optimisés distribués sur des réseaux d'objets communicants intelligents interconnectés: c'est la vision de l'Internet des Objets (IoT). Nous assistons actuellement au déploiement d'une nouvelle génération d'objets interconnectés dotés de capacités de communication de détection et d'activation (réseaux de transport d'information sans fil, RFID, WSAN, etc.) pour de nombreuses applications. Ainsi l'interconnexion d'objets dotés de capacités avancées de traitement va conduire à une révolution en termes de création et de disponibilité de service et va profondément changer notre façon d'agir sur notre environnement.

La domotique a surtout elle-même évolué, si bien que le terme est quelque peu dépassé. La domotique servait à automatiser sa maison ; aujourd'hui on parle de domotique 2.0, ou de « maison intelligente », pour bien marquer l'évolution de ce monde. Les différents domaines de la maison ne se contentent plus d'être automatisés et pilotables, ils communiquent ensemble, permettant à la maison de réagir selon différents évènements [1].

Le mémoire est organisé en une introduction générale, trois chapitres et une conclusion générale. Dans le premier chapitre on décrit le concept de l'internet of things et ces applications.
Par la suite, au deuxième chapitre, le principe de fonctionnemet et la programmation des deux dispositifs utilisés dans notre proj, Raspberry et Arduino, sont bien détaillées. Finalement, le plus important de ce travail qui est la ralisation d'une maison intélligente est présenté de point de vue logiciel et materiel.

Chapitre 1

Internet des objets et domotique

1.1. Introduction

l'Internet des objets (IdO) est une « infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution ». En réalité, la définition de ce qu'est l'Internet des objets n'est pas figée. Elle recoupe des dimensions d'ordres conceptuel et technique. Et domotique est ensemble des technologies de l'électronique de l'information et des télécommunications utilisées dans les domiciles. Elles visent à assurer des fonctions de sécurité, de confort, de gestion d'énergie et de communication qu'on peut retrouver dans une maison.

Dans ce chapitre, nous allons parler de concepts généraux sur l'internet des objets et la maison intelligente.

1.2. L'internet des objets

Le terme Internet of Things désigne généralement des scénarios où le réseau la connectivité et la capacité de calcul s'étendent aux objets, aux capteurs et aux objets du quotidien qui ne sont pas normalement ordinateurs considérés, permettant à ces dispositifs de générer, d'échanger et de consommer des données avec intervention humaine minimale [2].

L'Internet des objets est un réseau de réseaux qui réconcilie le virtuel et le réel. Il permetd'identifier directement et sans ambiguïté des entités numériques et des objets physiques etainsi de pouvoir récupérer, stocker, transférer et traiter des données qui y sont rattachées sansdiscontinuité entre tous les points du réseau. L'Internet des objets est aujourd'hui présentdans les équipements, les capteurs et les données que la plupart des entreprises exploitent,chaque objet présent sur la terre pourrait disposer d'une adresse Internet [3].

1.2.1. Historique

Le terme «Internet des Objets» (IoT) a été utilisé pour la première fois en 1999 par le pionnier de la technologie britannique Kevin. Ashton pour décrire un système dans lequel les objets du monde physique pourraient être connectés à Internet par des capteurs. Ashton a inventé le terme pour illustrer la puissance de la connexion des étiquettes d'identification par radiofréquence (RFID) utilisé dans les chaînes d'approvisionnement des entreprises à Internet afin de compter et de suivre les marchandises sans avoir besoin de intervention. Aujourd'hui, l'Internet des Objets est devenu un

terme populaire pour décrire des scénarios dans lesquels .la connectivité Internet et la capacité de calcul s'étendent à divers objets, appareils, capteurs et tous les jours articles [2].

Alors que le terme «Internet of Things» est relativement nouveau, le concept de combiner des ordinateurs et des réseaux pour surveiller et contrôler les appareils a été autour depuis des décennies. À la fin des années 1970, par exemple, les systèmes de la surveillance à distance des compteurs sur le réseau électrique via des lignes téléphoniques était déjà utilisée commercialement. Années 1990, les progrès de la technologie sans fil ont permis aux entreprises et aux industries de «machine à machine» (M2M) solutions pour la surveillance de l'équipement et l'exploitation pour se généraliser. Beaucoup de ces premiers M2M Cependant, les solutions reposaient sur des réseaux spécialisés fermés et sur des systèmes propriétaires ou spécifiques à l'industrie. normes, plutôt que sur les réseaux basés sur le protocole Internet (IP) et les normes Internet [2].

Utiliser l'IP pour connecter des périphériques autres que des ordinateurs à Internet n'est pas une idée nouvelle. Le premier Internet "Device" - un grille-pain IP qui peut être activé et désactivé sur Internet - a été présenté à conférence Internet en 1990. Au cours des années suivantes, d'autres «choses» ont été permises par IP, y compris un soda machine à l'Université Carnegie Mellon aux États-Unis et un pot de café dans la salle de Trojan à l'Université de Cambridge au Royaume-Uni (qui est resté connecté à Internet jusqu'en 2001). De ces débuts fantaisistes, un solide domaine de recherche et de développement dans le «réseau d'objets intelligents» a contribué à jeter les bases internet des objets d'aujourd'hui [2].

1.2.2. L'objet

Un objet est, avant toute chose, une entité physique; par exemple, un livre, une voiture, une machine à café électrique ou un téléphone mobile. Un objet connecté est un matériel électronique qui peut communiquer avec un Smartphone, une tablette tactile et/ou un ordinateur, ses caractéristiques pouvant évoluer au cours du temps (position, niveau de batterie, etc.). Il peut envoyer et recevoir des informations, via une liaison sans fil, Bluetooth ou WiFi, etc. Autres définitions s'accordent à dire qu'un objet possède des capacités de calcul, d'acquisition (capteur) et d'action (actionneur) [4].

Cependant, cette définition exclut les objets inertes identifiés par RFID. En effet, une puce RFID classique ne peut pas être considérée comme un dispositif de calcul, celle-ci se résumant à un identifiant stocké dans une mémoire. De même, si l'on considère les cas extrêmes, un simple code-barres peut être employé pour identifier un objet, ce dernier étant exempte d'une quelconque partie

électronique. L'intérêt principal d'un objet connecté est l'interactivité, la possibilité de récupérer des informations, ou d'envoyer des statistiques, de garder le contact, etc [4].

1.2.3. Type d'objet

Il est raisonnable de considérer que l'internet des objets est composé d' objets actifs, capables d' accomplir des calculs, d' effectuer des mesures sur l'environnement ou d'influer sur celui-ci, et d'objets passifs qui n'ont pas d'autres aptitudes que celles d'être suivis et détectés par des objets actifs.

- **Objets actif**

Un objet actif peut stocker tout ou partie de son identité et échanger directement ces informations avec d'autres objet actif.

Fig 1.1 : Objets actif

- **Objet passif**

Par extension, l'identité d'un objet passif n'est pas directement stockée dans celui-ci, a l'exception de l'identifiant, et nécessite l'utilisation d'une infrastructure tierce capable de stocker ces information [3].

Fig 1.2 : Objet passif.

1.2.4. Objectif d'internet des objets

L'objectif ambitieux derrière l'internet des objets d'après le concept d'objet étant posé, les différentes définitions présentées et la littérateur que nous étudiée à ce sujet nous conduisent à conclure que l'internet des objets est une infrastructure globale permettant notamment :

- aux objets actifs d'échanger des informations ou d'en collecter à propos des objets passifs, ce qui rejoint les définitions d'un réseau M2M mondial.
- de stoker et de rendre accessible l'identité des objets, les informations produites par ces derniers et toutes les connaissances nécessaires pour que les objets gagnent en autonomie, ce au travers de représentations, de structures et de formats de donnée manipulables par les machines.
- offrir les abstractions nécessaires aux êtres humains pour interagir avec ces machines, et par extension avec le monde physique, aussi simplement qu'avec le monde virtuel que nous connaissons aujourd'hui [4].

1.2.5. Avantages et risqué

Comme toutes les technologies, l'Internet des Objets présente des avantages et des risques. Développer la politique approches à cette industrie. Voici un résumé des avantages et risques d'IoT .

1.2.5.1. Avantages

Les avantages de l'IoT couvrent tous les domaines du style de vie et des affaires. Voici une liste de certains des avantages que l'IdO a à offrir :

A. Amélioration de l'engagement du client

Les analyses actuelles souffrent de points aveugles et de failles importantes dans la précision et comme indiqué, l'engagement reste passif. L'IoT transforme complètement ceci pour réaliser un engagement plus riche et plus efficace avec le public [5]

B . Optimisation de la technologie

Les mêmes technologies et données qui améliorent l'expérience client améliorent également l'utilisation de l'appareil et contribuent à des améliorations plus importantes de la technologie [5].

C. Web data pour le marketing des entreprises

Les entreprises peuvent choisir d'employer leur propre traqueur analytique pour connaître les comportements de leurs clients sur le web. Il est aussi possible d'externaliser les tâches à des firmes de marketing qui sont établies dans cet espace. Quel modèle de navigation web suivent les visiteurs quand ils entrent ou sortent du site internet, quels types d'appareils ils utilisent, et autres données révélatrices sur ces visiteurs seront alors agrégées afin de réaliser une image plus complète de ce qu'ils représentent. Tout cela va permettre d'enrichir votre analyse et vos prévisions marketing et pourra être implanté rapidement [5].

D. Identification des sites dangereux

Les entreprises commerciales offrent des services de sécurité qui autorisent le réseau administrateurs à traquer les échanges M2M et visites de site internet de la part d'ordinateurs d'entreprises. Il s'agit aussi de démasquer les sites dits « dangereux » et les adresses IP d'ordinateurs d'entreprises qui les visite de manière régulière. Cette pratique limite les risques de mise en danger des sites à cause de virus ou d'un programme malveillant. Puisque ce service de surveillance est disponible sur le cloud, les entreprises sont en mesure de les mettre en place rapidement et facilement [5].

1.2.5.2 risques

Bien que l'IoT offre un ensemble impressionnant d'avantages, il présente également un ensemble important de défis. Voici une liste de quelques-uns de ses principaux problems :

A. Sécurité

L'Internet des objets crée un écosystème d'appareils connectés en permanence qui communiquent sur des réseaux. Le système offre peu de contrôle malgré toutes les mesures de sécurité. Cela laisse les utilisateurs exposés à différents types d'attaquants [5].

B. Confidentialité

La sophistication de l'IoT fournit des données personnelles substantielles dans les moindres détails sans la participation active de l'utilisateur [5].

C. Complexité

Certains trouvent les systèmes IoT compliqués en termes de conception, de déploiement et de maintenance, compte tenu de l'utilisation de plusieurs technologies et d'un grand [5].

D. Flexibilité

Beaucoup s'inquiètent de la flexibilité d'un système IoT pour s'intégrer facilement avec un autre. Ils s'inquiètent de se retrouver avec plusieurs systèmes conflictuels ou verrouillés [5].

E. Conformité

L'IoT, comme toute autre technologie dans le domaine des affaires, doit être conforme à la réglementation. Sa complexité fait en sorte que la question de la conformité semble incroyablement difficile [5].

1.2.6. Application de l'internet des objets

On peut distinguer différentes catégories d'applications :

1.2.6.1 villes intelligentes

Une ville intelligente est une zone urbaine qui utilise différents capteurs de collecte de données électroniques pour fournir des informations permettant de gérer efficacement les ressources et les actifs. Cela comprend les données collectées auprès des citoyens, des dispositifs mécaniques, des actifs, traitées et analysées pour surveiller et gérer les systèmes de circulation et de transport, les centrales électriques, les réseaux d'approvisionnement en eau, la gestion des déchets, les systèmes d'information, les écoles, les bibliothèques et les hôpitaux [6].

Le concept de ville intelligente intègre les technologies de l'information et de la communication (TIC) et divers dispositifs physiques connectés au réseau (l'Internet des objets ou IoT) pour optimiser l'efficacité des opérations et des services urbains et se connecter aux citoyens [6].

FIg 1.3 : Représentation d'une ville intelligente [6].

Les réseaux intelligents ou « smart grids » sont des réseaux d'électricité qui, grâce à des technologies informatiques, ajustent les flux d'électricité entre fournisseurs et consommateurs. En collectant des informations sur l'état du réseau, les smart grids contribuent à une adéquation entre production, distribution et consommation. Il est nécessaire de différencier smart grid et compteur communicant (ou « smart meter »), qui renseigne le consommateur sur sa demande en électricité. « Smart grids » est une appellation générale pour l'ensemble des technologies et des infrastructures « intelligentes » installées. Chez le particulier, le compteur communicant est une première étape dans la mise en place des smart grids [7].

Les réseaux intelligents peuvent être définis selon quatre caractéristiques en matière de :

Flexibilité : ils permettent de gérer plus finement l'équilibre entre production et consommation.

Fiabilité : ils améliorent l'efficacité et la sécurité des réseaux.

Accessibilité : ils favorisent l'intégration des sources d'énergies renouvelables sur l'ensemble du réseau.

Économie : ils apportent, grâce à une meilleure gestion du système, des économies d'énergie et une diminution des coûts (à la production comme à la consommation) [7].

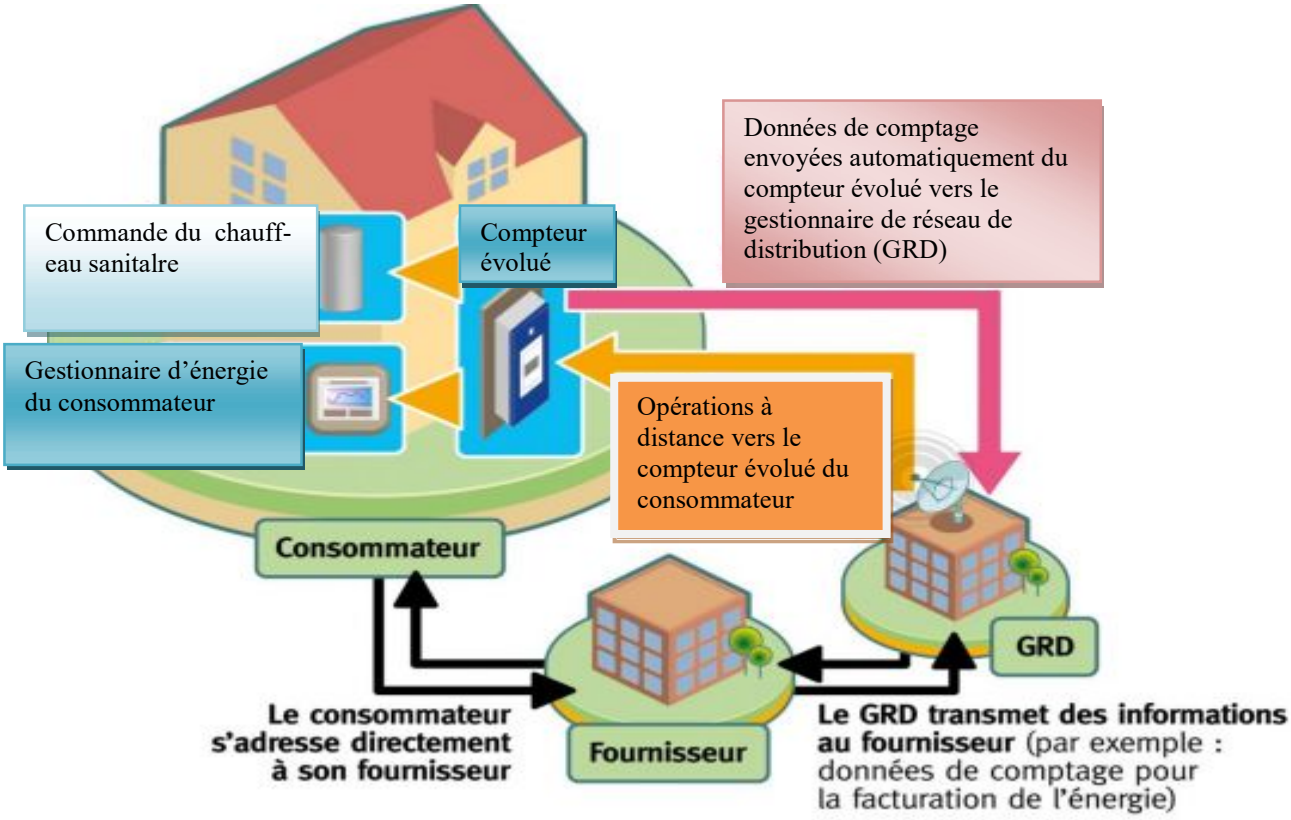

Fig 1.4 : Representation les réseaux intelligent [7].

1.2.6.2 Transport et logistique

Voitures, trains, bus et vélos se voient de plus en plus dotés de capteurs, actuateurs et d'une logique de traitement des informations. Les routes aussi peuvent être munies de capteurs et tags (étiquettes) qui envoient des informations sur la circulation aux stations de contrôles mais aussi directement aux voyageurs pour mieux gérer le trafic, améliorer la sécurité routière et guider les tourists [1].

1.2.6.3. Soinset santé

Les objets connectés permettent de suivre et identifier en temps réel et à la demande, les outils, équipement et médicaments. Pour avoir des informations instantanément sur un patient peut souvent être déterminant [1].

1.2.6.4. Environnements intelligents

Capteurs et actuateurs distribués dans plusieurs maisons et bureaux peuvent augmenter le confort dans ces environnements, le chauffage peut s'adapter à la météo, l'éclairage suivant l'horaire et la position du soleil , des incidents domestiques peuvent être évités avec des alarmes et beaucoup d'énergie pourrait être économisée. Les environnements intelligents peuvent aussi améliorer l'automatisation en milieu industriel avec un déploiement massif de tags RFID associés aux différentes étapes de la production. La ville intelligente est un exemple d'environnement intelligent. Le quartier d'affaires de Songdo en Corée du Sud est la première ville intelligente opérationnelle [1].

1.2.7. Composants d'un système IOT

L'internet des objets n'est pas une technologie à part-entière mais plutôt un système intégrant plusieurs autres systèmes. Lier un objet ou un lieu à Internet est un processus plus complexe que la liaison de deux pages Web. Divers composant sont de mise, L'IOT en exige sept :

1. Une étiquette physique ou virtuelle pour identifier les objets et les lieux.

2. Un moyen de lire les étiquettes physiques, ou de localiser les étiquettes virtuelles.

3. Un dispositif mobile (smartphone, tablette, ordinateur portable).

4. Un logiciel additionnel pour le dispositif mobile.

5. Un réseau sans fil de type 2G, 3G ou 4G.

6. L'information sur chaque objet lié.

7. Un affichage pour regarderl'informationsurl'objetlié [1].

Le tableau suivant résume les principaux systèmes technologiques nécessaires à l'implantation d'une solution IOT.

Type de système	Enjeux	Technologies employées
Identification	Reconnaître chaque objet de façon unique et recueillir les données stockées au niveau de l'objet.	Code-barres, URI, GPS, radio-identification(RFID), ADN...etc.
Capteurs	Recueillir des informations présentes dans l'environnement pour enrichir les fonctionnalités du dispositif.	Luxmètre, capteur de proximité, thermomètre, hydromètre, accéléromètre, gyroscope...etc.
Connexion	Connecter les systèmes entre eux.	Câbles, fréquences radio, Bluetooth, Wi-Fi, ZigBee, Z-Wave, NFC...etc.
Intégration	Intégrer les systèmes pour que les données soient transmises d'une couche à l'autre.	Middleware (simple et complexe), analyse décisionnelle des systèmes complexe.
Traitement de données	Stocker et analyser les données pour lancer des actions ou pour aider à la prise de décisions.	Tableur, base de données, entrepôt de données, progiciel de gestion intégré(PGI).
Réseaux	Transférer les données dans les mondes physiques et virtuels.	Internet

Tableau 1-1 Composants d'une solution IOT [1].

1.3. Domotique et Maison intelligent

1.3.1. Définition de la Maison Intelligente

La maison intelligente est un paradigme (une manière de voir les choses, souvent présenté comme objet du futur et parfois comme un fantasme) qui se positionne en successeur de la domotique, bénéficiant des avancées en informatique ubiquitaire que l'on dénomme aussi l'informatique ambiante, intégrant notamment l'internet des objets. Outre la dimension dominante de l'informatique, la maison intelligente telle que (re)présentée dans les années 2010 se veut également plus centrée utilisateur, s'éloignant de l'approche technophile caractéristique de la « domotique des années 1990 » [8].

À l'avenir, le chauffage, la climatisation, l'éclairage, la gestion des flux (eau, énergie, aliments, déchets, information...) et la sécurité pourraient être pour tout ou partie gérées par un système informatique, auto-apprenant dédié (centralisé ou non), en interaction avec les besoins des occupants, éventuellement en utilisant des énergies et des ressources moins luisantes pour l'environnement, avec ou sans câblage. La maison s'adapterait aux habitudes et aux goûts de ses habitants et invités (éventuellement malvoyants, handicapés, âgés, malades, etc.), grâce à un profilage de ces derniers, communiqué au système gérant la maison. Certains imaginent aussi une maison intelligente et autonome pour ses besoins en eau, thermies, frigories ou électricité, capable *«* de détecter d'elle-même, des dysfonctionnements ou des changements de paramètres susceptibles de présenter un danger » [8].

1.3.2. Définition de la domotique

Le mot domotique vient de domus qui signifie «domicile » et du suffixe –tique qui fait référence à la technique. la domotique est l'ensemble des techniques de l'électronique, de physique du bâtiment, d'automatisme, de l'informatique et des télécommunications utilisées dans le bâtiment, plus ou moins « interopérables » et permettant de centraliser le contrôle des différents systèmes et sous-systèmes de la maison et de l'entreprise (chauffage, volets roulants, porte de garage, portail d'entrée, prise électrique, etc.). la domotique vise à apporter des solutions techniques pour répondre aux besoins de confort (gestion d'énergie, optimisation de l'éclairage et du chauffage), de sécurité (alarme) et de communication (commandes à distance, signaux visuels ou sonores, etc.) que l'on peut trouver dans les maisons, les hôtels, les lieux publics…etc**[7]**.

A l'origine, la domotique avait donc pour but d'automatiser sa maison : ouverture et fermeture automatiques des volets, ouverture du portail électrique, gestion du chauffage, gestion de l'éclairage, etc. Ainsi avant l'ère des smartphones, il était par exemple possible d'activer son chauffage à distance en passant un coup de téléphone à sa maison, ou encore en lui envoyant un SMS. C'était tout à fait réalisable. Seulement une telle installation était relativement compliquée à mettre en place et, il faut bien l'avouer, couteuse. Cette époque a malheureusement laissé des traces, puisque pour beaucoup encore aujourd'hui, domotique rime avec cher et compliqué. Pourtant, ce domaine a énormément évolué et il existe de nombreuses solutions simples à mettre en place et tout à fait abordables pour le grand public [7].

La domotique a surtout elle-même évolué, si bien que le terme est quelque peu dépassé. La domotique servait à automatiser sa maison ; aujourd'hui on parle de domotique 2.0, ou de « maison intelligente », pour bien marquer l'évolution de ce monde. Les différents domaines de la maison ne se contentent plus d'être automatisés et pilotables, ils communiquent ensemble, permettant à la maison de réagir selon différents évènements **[7]**.

1.3.3. Domaines de la domotique

Les services offerts par la domotique couvrent 3 domaines principaux :

- Assurer la protection des personnes et des biens en domotique de sécurité.
- Veiller au confort de vie quotidien des personnes âgées, entre autres, en installant une domotique pour les personnes à mobilité réduite.
- Faciliter les économies d'énergie grâce à la réactivité maîtrisée d'une maison intelligente [6].

1.3.3.1. Protection des personnes et des biens

La domotique permet le suivi des personnes âgées ou handicapées. En matière de sécurité domestique, rien n'est laissé au hasard. Alarmes, détecteurs de mouvement ou d'intrusion, interphones et portiers vidéo, téléphones, simulateurs de présence, etc. se combinent pour détourner les visiteurs indésirables et arbitrer toutes les fonctions [9].

D'autres systèmes de détection sont prévus pour surveiller les enfants, prévenir les risques d'accident (incendie, fuite de gaz, etc.) et signaler des pannes (inondation, coupure de courant électrique, etc.). La domotique de sécurité passe également par la centralisation de la surveillance et du contrôle de toutes les zones de la maison. Des capteurs de mouvements, de bris de glace,

d'ouverture, etc., des poignées biométriques, l'automatisme des volets… sont installés sur les ouvertures et préviennent de toute intrusion, car l'ensemble est couplé à des alarmes silencieuses sans fil ou des sirènes. Pour l'intérieur des pièces, des micros ultrasensibles, des caméras invisibles, des champs magnétiques, des détecteurs de fumées assurent aussi une grande sécurité s'ils sont judicieusement positionnés [9].

1.3.3.2. Confort de la vie quotidienne

Avec une installation domotique, on pourra aujourd'hui avoir une maison vivante et économe. Le fait de rendre la maison intelligente assurera un résultat basse-consommation évident. L'habitat offre aussi un bien-être sur-mesure, avec un confort en permanence. Manipuler ses volets roulants ou battants en pressant un bouton est devenu chose courante de nos jours. De même qu'ouvrir le portail ou la porte du garage depuis sa voiture. Plus globalement, tout ce qui se fait avec un interrupteur ou une poignée peut être automatisé et piloté à partir d'un poste fixe, ou à distance via une télécommande, un ordinateur ou un Smart phone [9].

1.3.3.3. Les économies d'énergie

La domotique permet de diminuer jusqu'à 10 % des factures d'énergie. Grâce aux automatismes et à des capteurs, les équipements électriques inter-reliés pilotent au plus juste la consommation énergétique (chauffage, éclairage, eau, ventilation, etc.), tout en gardant sous contrôle le confort des zones occupées.

Le but principal de la domotique est d'éviter le gaspillage en supprimant les dépenses inutiles. Les systèmes de régulation permettent de maîtriser la consommation d'électricité, de gérer le chauffage et la production d'eau chaude sanitaire, avec un niveau de confort optimal. Un détecteur de présence placé dans chaque pièce, par exemple, commande instantanément l'allumage ou l'extinction des éclairages, la mise en route ou l'arrêt du chauffage, etc [9]. Au jardin par exemple, l'arrosage s'automatise et le détecteur crépusculaire se charge d'allumer les lumières dès la tombée de la nuit et ainsi de lancer l'irrigation des plantes.

La maison intelligente utilise la programmation domotique via des scénarios qu'on peut déterminer en fonction des besoins spécifiques, évitant les pertes thermiques inutiles et palliant les risques d'oubli ou de sécurité [9].

1.4. Conclusion

On a vu dans ce chapitre qu'est-ce que l'Internet des Objets leur objectif ainsi quelques domaines d'application et Avantages et risques. comme on a vu dans ce chapitre la domotique et Domaines d'emploi. Dans le chapitre suivant, on va voir en détails c'est quoi les cartes électronique Raspberry Pi et Arduino, ses composants, son domaine d'utilisation.

Chapitre 2
RASPBERRY PI ET ARDUINO

2.1. Introduction

ArduinoUno et Raspberry Pi sont tous les deux des circuits de la taille d'une carte de crédit, ils diffèrent par de nombreux aspects. Raspberry Pi est un nano-ordinateur que fonctionne sous Linux et que peut être interfacé avec des composants électronique externes, tandis qu'Arduino est une carte à microcontrôleur rudimentaire qui n'est pas dotée d'un quelconque système d'exploitation. A cause de leurs caractéristiques, on a choisis ces deux cartes électroniques à notre projet, Arduino Uno et Raspberry Pi 3 pour commander notre maison.

2.2. Raspberry Pi

2.2.1. L'histoire du raspberry pi

Un petit retour sur l'histoire du raspberry pi et sur sa cible du marché initiale permet d'en comprendre les contraintes et les limitations. Le concept de raspberry pi a été dévoilé autour de 2006 par le créateur de jeux vidéo EbenUpton et ses collègues de laboratoire d'informatique de l'université de Cambridge , en Angleterre. Ils étaient préoccupés par la baisse par la baisse de niveau de connaissances et de compétences des niveaux étudiants en informatique en comparaison de leurs ainés. EbenUpton a alors décidé de créer un ordinateur à faible prix, car il supposait que les parents craignaient de laisser leurs enfants jouer avec des PC modernes relativement couteux. C'est décente idée qu'est parte le développement du Raspberry Pi très bon marché. Cet ordinateur devait donner aux jeunes l'opportunité d'apprendre et de pratiquer la programmation, sans que leurs parents nes'inquiètent des dégâts éventuels sur la machine [10]

2.2.2. Définition de la carte Raspberry Pi

Le Raspberry Pi est une série de petits ordinateurs monocarte développés au Royaume-Uni par la Fondation Raspberry Pi pour promouvoir l'enseignement de l'informatique de base dans les écoles et les pays en développement. Le modèle original est devenu beaucoup plus populaire que prévu. marché cible pour des utilisations telles que la robotique. Il n'inclut pas les périphériques (tels que les claviers, les souris et les étuis). Cependant, certains accessoires ont été inclus dans plusieurs bundles officiels et non officiels [11].

2.2.3. Les composants standards de Raspberry Pi

La figure 2.1 représente un Raspberry standard :

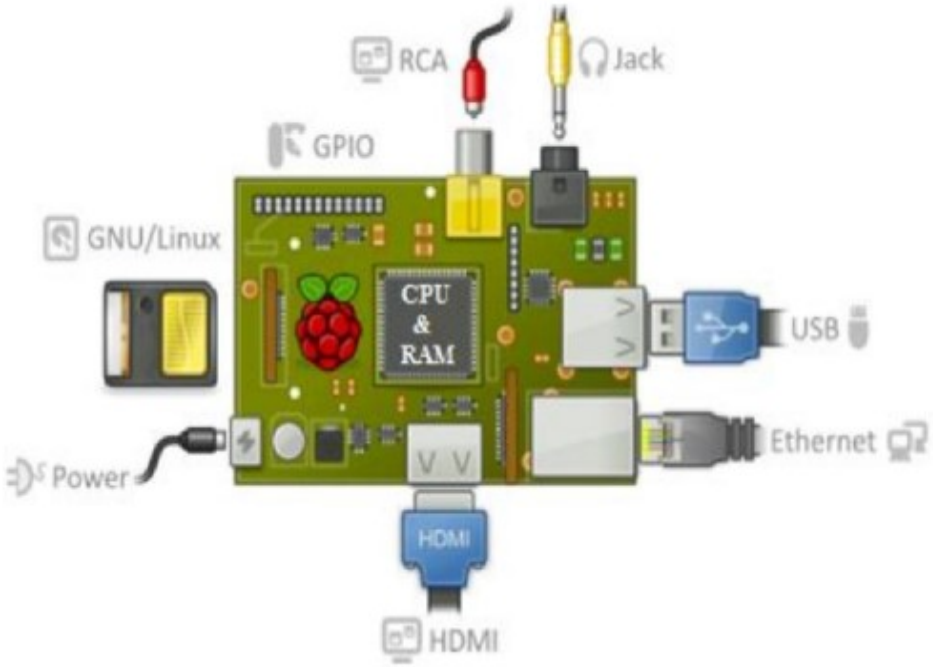

Fig 2.1 : Les composants standards d'un Raspberry [12].

- **Processeur ARM** : Les architectures ARM sont des architectures de processeurs, à faible consommation, introduites à partir de 1983 par « Acorn Computers » et développées depuis 1990 par « ARM Ltd ».
- **Mémoire vive RAM :** C'est la mémoire dans laquelle le Raspberry place les données lors de son traitement.
- **Une connectique variée :**

- **HDMI :** « High DefinitionMultimedia Interface » permet de relier le Raspberry PI à un dispositif compatible : écran LCD ou un vidéoprojecteur…
- **Port USB 2.0 :** Le port « Universal Serial Bus » est un port série qui sert à connecter le Raspberry aux autres périphériques.
- **Port Ethernet :** C'est un port qui correspond au protocole international ETHERNET de réseau local à commutation de paquets.

- **Prise RCA :** « Radio Corporation of America » est un connecteur électrique utilisé dans le domaine audio/vidéo.
- **un slot les cartes SD :** Le Raspberry a besoin d'une mémoire externe supplémentaire pour fonctionner. Ce slot permet de connecter la mémoire externe.
- **une prise jack :** C'est une connectique audio-vidéo.
- **GPIO : «** General Purpose Input/Output » sont des ports d'Entrée/Sortie [12].

2.2.4. Modèles et caractéristiques

Dans ce paragraphe on va faire une brève présentation des modèles et les caractéristiques du Raspberry Pi, les modèles les plus couramment diffusés et ferons l'impasse sur les tout premiers modèles, ainsi que sur la gamme professionnelle [12].

Fig 2.2: Différents modèles du Raspberry Pi [12].

Chacun de ces modèles ayant des caractéristiques bien spécifiques pour cela nous avons décidé de présenter les caractéristiques de chaque modèle dans un schéma qui fait une comparaison entre ces différents modèles :

Modèle	A	A+	B	B+	B2
CPU	Monocœur ARM 700 MHz				Quadricœur ARM 900 MHz
GPU	Décodeur vidéo Broadcam VideoCore IV				
RAM	256 MO		512 MO		1 GO
USB	1 * USB2.0		2 * USB2.0	4 * USB2.0	
Audio/vidéo	Jack 3.5, composite et HDMI	HDMI et jack audio/vidéo	Jack 3.5, composite et HDMI	HDMI et jack audio/vidéo	
Ethernet	0	10/100	0	10/100	
Entrées/sorties	GPIO 26 pts	GPIO 40 pts	GPIO 26 pts	GPIO 40 pts	
OS	Officiel : Raspbian Tiers : Fedora, XBMC/Kodi, OSMC				
Stockage	SD	Micro SD	SD	Micro SD	
Dimensions	86*54*17	65*54*17	86*54*17		
Poids	45g	23g	45g		
Consommation	1.5W	1W	3.5W	3W	

Tableau 2.1: Les caractéristiques des modèles[12].

2.2.5. Connexion du système

Sauf si vous intégrez votre Raspberry Pi dans un projet ou si vous l'utilisez en tant que centre multimédia, vous aurez besoin d'attacher un clavier, une souris, un moniteur et probablement un dongleWiFi. Comme cela ajoute à plus de trois connexions, même avec un modèle Raspberry Pi B, vous aurez besoin d'un concentrateur USB pour fournir suffisamment de prises USB. La Figure 2.3 montre un système Raspberry Pi typique.

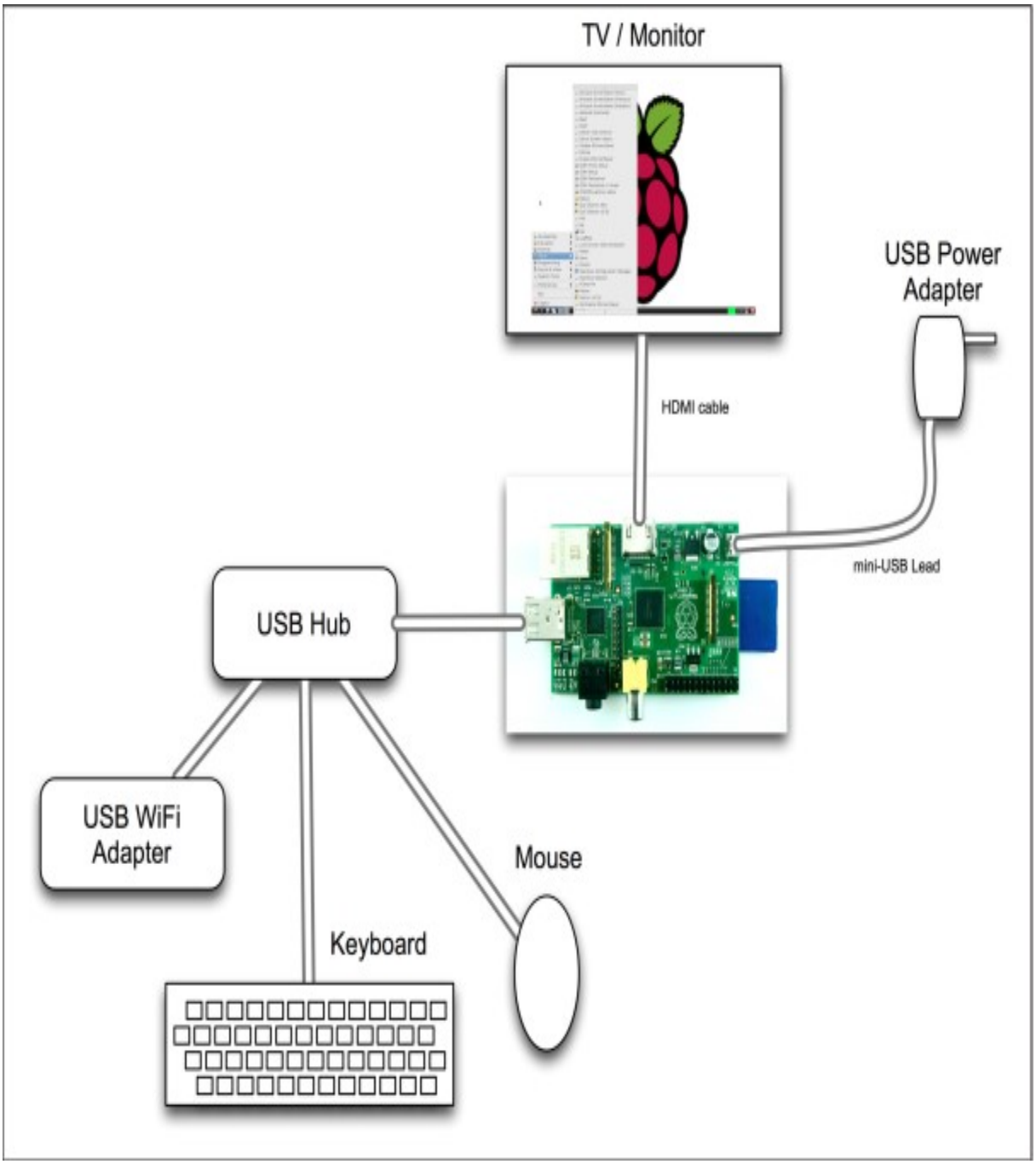

Fig 2.3 : Un système de Raspberry Pi typique [13].

➢ Acquisition du matériel nécessaire

Le tableau 2.2 liste ce qu'il vous faut pour obtenir un système Raspberry Pi totalement fonctionnel [13].

Article	Distributeur et n° d'article	Informations complémentaires
Raspberry Pi, Modèle A ou B	Farnell (www.farnell.com) Newark (www.newark.com) RS Components (www.rs-components.com)	La différence entre les deux modèles est que le modèle B a une connexion réseau.
Chargeur USB	Farnell : 1734526	Chargeur USB 5 V USB. Doit pouvoir fournir une alimentation de 700 mA (3 W), mais 1 A (5 W) est préférable.
Câble micro-USB	RadioShack : 55048949 Farnell : 2115733 Adafruit : PID 592	
Clavier et souris	Magasin d'informatique	Tout clavier USB fera l'affaire. Les claviers et les souris sans fil qui sont fournis avec leur propre adaptateur USB fonctionnent également.
Télé/moniteur avec prise HDMI	Magasin d'informatique ou d'électronique	
Câble HDMI	Magasin d'informatique ou d'électronique	
Carte SD (préparée)	SK Pang : RSP-2GBSD Newark : 96T7436 Farnell : 2113756	
Adaptateur Wi-Fi*	http://elinux.org/RPi_VerifiedPeripherals#USB_WiFi_Adapters	Elinux.org fournit une liste à jour des adaptateurs Wi-Fi.
Hub USB*	Magasin d'informatique	
Adaptateur HDMI vers DVI*	Newark : 74M6204 Maplins : N24CJ Farnell : 1428271	
Câble Ethernet*	Magasin d'informatique	
Boîtier*	Adafruit, SK Pang, ou Alliedelec.com	
		* Ces articles sont optionnels.

Tableau 2.2 :Composants un système Raspberry Pi [13].

2.2.6. Caractéristiques du Raspberry Pi3 modèle B

Le Raspberry Pi 3 (modèle B) est un ordinateur miniature de la taille d'une carte de crédit. Il repose sur un processeur à quadruple cœur ARM Cortex-A53, le BCM2837 de Broadcom, cadencé à 1,2 GHz. Cela signifie qu'il est 50 à 60% plus rapide que le

Raspberry Pi 2 B. La communication par Wi-Fi 802.11n et du Bluetooth 4.1 est maintenant intégrée au Raspberry Pi 3 ; ce nouveau modèle est toujours rétro compatible avec les modèles précédents.

Fig 2.4 : Raspberry Pi 3 [14].

- Processeur BCM2837 de Broadcom, cadencé à 1,2 GHz.
- Quadruple cœur ARM Cortex-A53 à 64 bits.
- Wi-Fi 802.11 b/g/n intégré.
- Bluetooth 4.1 (*Classic* & *Low Energy*) intégré.
- Coprocesseur multimédia à double cœur Videocore IV®.
- Mémoire LPDDR2 de 1 Go.
- Supporte les distributions GNU/Linux ARM les plus récentes et Windows 10 IoT.

- Connecteur micro-USB pour alimentation 2,5 A.
- 1x port Ethernet 10/100.
- 1x connecteur audio/vidéo HDMI.
- 1x connecteur audio/vidéo RCA.
- 4x port USB 2.0.
- 40 broches d'E/S à usage général.
- Antenne sous forme de puce.
- Connecteur DSI pour écran.
- Lecteur de carte micro-SD.
- Dimensions : 85 x 56 x 17 mm [14].

2.2.6.1 Choisir son systèmed'exploitation

Le Raspberry Pi est compatible avec les systèmes d'exploitation GNU/Linux. Certaines distributions se sont spécialisées pour ce matériel spécifique. La liste des systèmes d'exploitation compatibles se trouve sur cette page du site officiel. Mon choix s'est porté vers la **Raspbian « Wheezy »** qui est une distribution dérivée de Debian que je manipule tous les jours. Il existe également une autre distribution basée sur Arch Linux qui offre des performances equivalents [15].

Le système d'exploitation doit être stocké sur une carte SD à insérer dans le lecteur du Raspberry PI. Pour mettre le système Raspbian sur la carte SD, j'ai utilisé mon PC portable sous Ubuntu 12.10 qui dispose d'un lecteur intégré de carte SD. La procédure suivante est donc à adapter selon votre matériel [15].

Au premier démarrage, vous allez avoir droit au lancement automatique de l'utilitaire de configuration **Raspi-config**qu'il est possible de rappeler par la suite en saisissant la commande suivante dans un terminal : sudoraspi-config

Raspi-config se présente dans la figure 2.5 [15]:

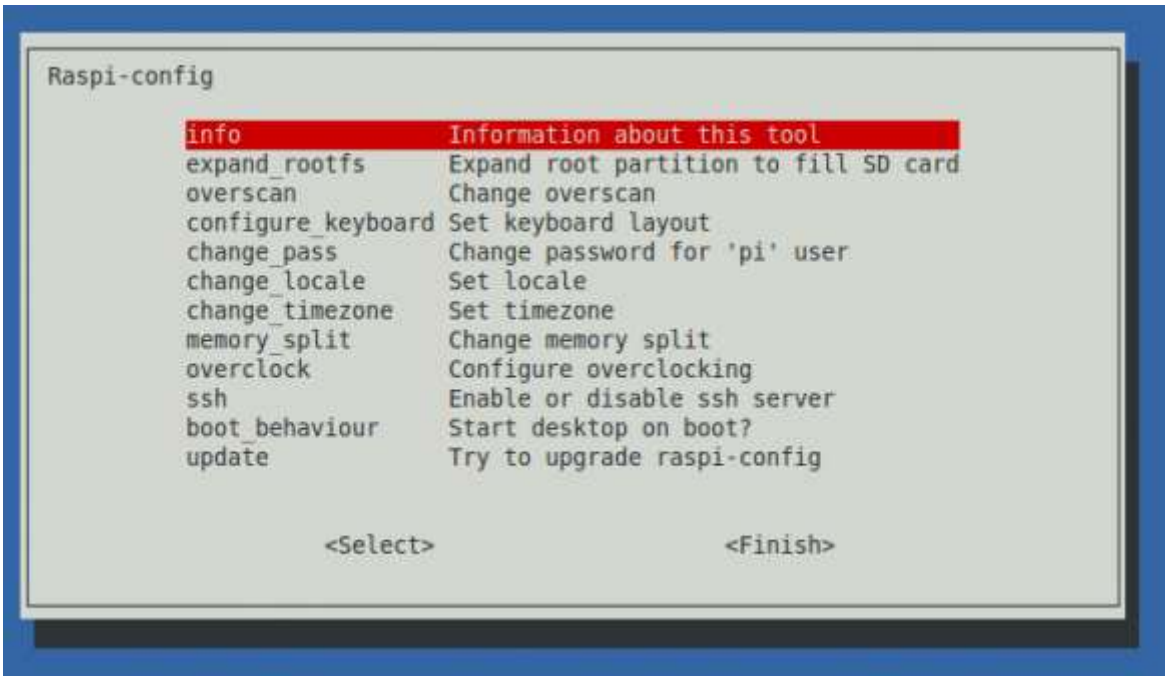

Fig 2.5 : configuration Raspberry Pi 3 [15].

Les différentes fonctions permettent de :

- **expand_rootfs** : étendre la partition principale à la taille maximale de la carte SD (par exemple si vous avez une carte SD de 16 Go et que vous n'utilisez pas cette fonction, alors vous resterez avec une partition principale de 2 Go) .
- **overscan** : force l'affichage de marge pour s'adapter à des écrans 16/9 ou supérieur .
- **configurer** : le clavier (**configure_keyboard**), le mot de passe par défaut (raspberry) de l'utilisateur par défaut (pi) (**change_pass**), votre lieu géographique (**change_locale**) et le temps (**change_timezone**) .
- **overclocking**: comme Raspberry est une société intelligente, elle autorise l'overclocking de son matériel sans perte de la garantie (mais avec une durée de votre matériel qui risque de diminuer). Fonction à activer si vous voulez utiliser le Raspberry en mode graphique (comme client léger) ou tout autre besoin consommateur de CPU [15].

On a le choix d'overclockingcomme le montre la figure 2.6:

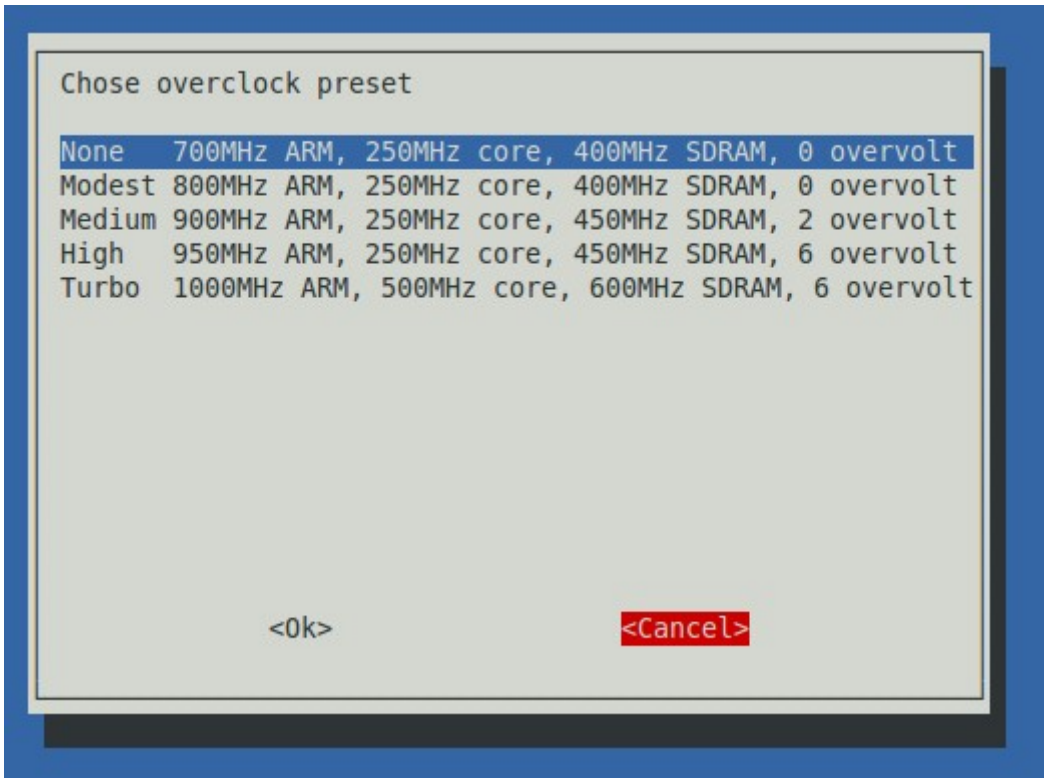

Fig 2.6 :choixd'overclocking Raspberry Pi 3 [15].

- **Memory-split :** pour définir la quantité de mémoire vive dédiée au GPU (64 Mo par défaut) .
- **ssh** : permet de définir si le serveur SSH doit être lancé par défaut .
- **boot_behavor** : choisir si on veut démarrer le boîtier en mode texte (utilisation serveur) ou graphique (client léger avec l'environnement OpenBox + LXDE) .
- **Update** : qui est en fait un raccourci vers la commande apt-get update &&apt-get upgrade et ainsi mettre à jour votre système (qui doit être connecté à Internet) [15].

2.2.6.2 Le port GPIO Raspberry Pi 3 Model B

Permettent de contrôler d'autres composants électroniques ainsi que des interfaces telles que des LED, des moteurs et des relais. Ces diverses interfaces sont généralement regroupées sous le terme de « sorties ». Pour ce qui est des « entrées », votre Raspberry Pi peut lire et interpréter l'état de boutons, d'interrupteurs, de capteurs de température, de lumière, de mouvement ou de proximité, etc., la liste est très longue. En utilisant les interfaces numériques asynchrones (port série vu au chapitre précédent) et synchrones (SPI et I2C), les

applications potentielles sont encore plus nombreuses. Le Raspberry B+ comporte 40 broches [16].

Fig 2.7 :Le port GPIO [16].

2.2.6.3. Installation de RPi.GPIO

Téléchargez et installez la bibliothèque RPi.GPIO Python. À partir d'une fenêtre de terminal sur votre Pi, tapez les commandes suivantes pour récupérer et installer la bibliothèque RPi.GPIO.

sudo apt-get install python-dev

sudo apt-get install python-rpi.gpio [16].

2.2.6.4. Configuration de I2C

La distribution est préconfigurée avec le support I2C. Si vous utilisez Raspbian, vous avez besoin de quelques modifications de configuration faire:

Editez le fichier / etc / modules en utilisant la commande sudo nano / etc / modules et ajoutez les lignes suivantes à la fin de celui-ci

i2c-bcm2708

i2c-dev

L'utilisation de modules I2C est en fait un très bon moyen d'interfaçage avec le Pi. Il réduit le nombre de fils dont vous avez besoin pour tout connecter (deux seulement 4) et il y a quelques modules I2C vraiment soignés disponibles.La Figure 2.8 montre une sélection de modules I2C disponibles auprès d'AdaFruit. D'autres fournisseurs tels que Sparkfun ont aussi des appareils I2C. De gauche à droite, il desaffichages de matrice de LED,Un afficheur LED à quatre chiffres à sept segments, un contrôleur PWM / Servo 16 canaux et un Module d'horloge en temps réel. D'autres modules I2C disponibles incluent des émetteurs radio FM, des télémètres à ultrasons, Écrans OLED et différents types de capteurs [16].

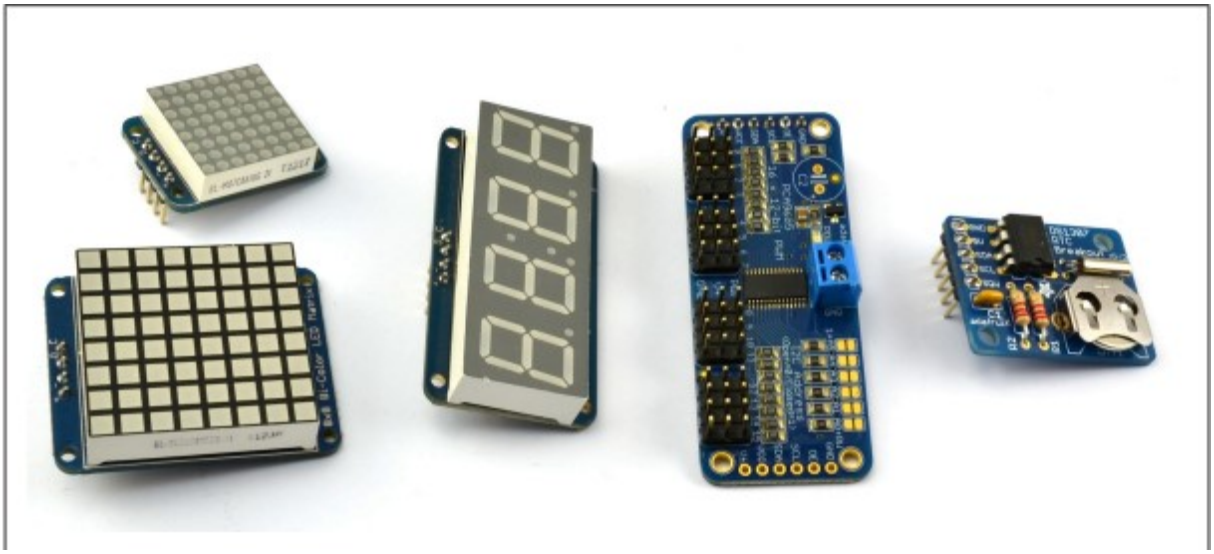

Fig2.8 :. I2C Modules [16].

2.2.6.5. Utiliser les outils I2C

Installez et utilisez les outils i2c. A partir d'une fenêtre de terminal sur votre Pi, tapez les command pour récupérer et installer les outils i2c

sudo apt-get install python-smbus

sudo apt-get install i2c-tools

Attachez votre périphérique I2C au Pi et exécutez la commande:

sudo i2cdetect -y 1 [16].

2.2.7. Présentation du python

Python est un langage de programmation, dont la première version est sortie en 1991. Créé par Guido van Rossum, il a voyagé du Macintosh de son créateur, qui travaillait à cette époque au Centrum voorWiskunde en Informatica aux Pays-Bas, jusqu'à se voir associer une

organisation à but non lucratif particulièrement dévouée, la Python Software Foundation, créée en 2001 [3].

Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Dès l'instant où vous l'installez sur votre ordinateur, vous disposez de nombreuses fonctionnalités intégrées au langage. Ainsi, il existe ce qu'on appelle des bibliothèques qui aident le développeur à travailler sur des projets particuliers [3].

Python est un langage de programmation interprété, c'est-à-dire que les instructions que vous lui envoyez sont transcrites en langage machine au fur et à mesure de leur lecture. D'autres langages (comme le C / C++) sont appelés langages compilés, car avant de pouvoir les exécuter, un logiciel spécialisé se charge de transformer le code du programme en langage machine [3].

En contrepartie, un langage compilé se révélera bien plus rapide qu'un langage interprété (la traduction à la volée de votre programme ralentit l'exécution), bien que cette différence tende à se faire de moins en moins sentir au fil des améliorations.
Programmation des entrées / sorties avec python [3].

Les broches marquées GPIO peuvent être utilisées comme broches d'entrée / sortie. En d'autres termes, n'importe quelle broche peut être programmée comme une entrée ou une sortie. Dans ce sens on va utiliser plusieurs langages de programmation capable de contrôler ces broches comme le C, Java, Bash... mais dans notre projet on a opté pour le python pour contrôler ces broches [3].

Le module GPIO est installé par défaut sur les versions les plus récentes de la distribution Raspbian Linux. Mais pour les versions plus anciennes, on doit probablement l'installer et effectuer une mise à jour. Pour cela on exécute la commande suivante :

"sudoapt-getinstall python-rpi.gpio"

Ensuite on exécute la commande suivante pour la mise à jour :

"sudoapt-get update".

Remarque

Avant d'utiliser les broches, vous devez indiquer au module GPIO comment votre code y accédera. Le Raspberry Pi autorise deux numérotations : celle de la sérigraphie du connecteur de la carte (GPIO.BOARD), ou la numérotation électronique de la puce (GPIO.BCM). À nous de choisir celle qu'on veut [3].

Voici maintenant un petit programme led.py pour contrôler une LED :

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)    # broche 18 comme sortie
print "LED on"
GPIO.output(18, GPIO.HIGH)  # led allumee
time. sleep (1)          #attendre pour une seconde
print "LED off"
GPIO.output(18, GPIO.LOW)   # led etteinte
```

Fig 2.9: programme contrôler une LED

Pour tester le programme on exécute la commande suivante sur la console :

"sudo python led.py" [3].

2.3. Arduino

2.3.1. Historique

L'Arduino a été inventé 2005 par l'enseignent Massimo Banzi dans une école de dessin à Ivea en Italie avec l'aide de David Cuartielle ingénieur en microcontrôleurs et aussi leur étudiant Mellis qui est spécialiste dans les langages de programmation [17].

2.3.2 Définition

La carte Arduino est une plateforme utilisée pour réaliser des projets électroniques plus développé. Elle est composée d'un circuit physique programmables est dit microcontrôleurs et de logiciel utilisé pour créer et télécharger le code de l'ordinateur à la carte [17].

2.3.3. Composition

Il nous permet de réalisé plusieurs projets tel que :

Contrôler les appareils domestiques .

• Robotique.

• Jeu de lumière.

• Communiquer avec PC.

• Télécommander un appareil mobile.

• Etc [17].

2.3. 4 Les gammes de la carte Arduino

Actuellement, il existe plus de 20 versions de module Arduino, nous citons quelques un afin d'éclaircir l'évaluation de ce produit scientifique et académique:

- Le NG d'Arduino, avec une interface d'USB pour programmer et usage d'un ATmega8.

- L'extrémité d'Arduino, avec une interface d'USB pour programmer et usage d'un Microcontrôleur ATmega8.

- L'Arduino Mini, une version miniature de l'Arduino en utilisant un microcontrôleur ATmega168.

- L'Arduino Nano, une petite carte programme à l'aide porte USB cette version utilisant un microcontrôleur ATmega168 (ATmega328 pour une plus nouvelle version).

- Le LilyPadArduino, une conception de minimaliste pour l'application wearable en utilisant un microcontrôleur ATmega168.

- Le NG d'Arduino plus, avec une interface d'USB pour programmer et usage d'un ATmega168.

- L'Arduino Bluetooth, avec une interface de Bluetooth pour programmer en utilisant un microcontrôleur ATmega168.

- L'ArduinoDiecimila, avec une interface d'USB et utilise un microcontrôleur ATmega168.

- L'ArduinoDuemilanove ("2009"), en utilisant un microcontrôleur l'ATmega168 (ATmega328 pour une plus nouvelle version) et actionné par l'intermédiaire de la puissance d'USB/DC.

- L'ArduinoMega, en utilisant un microcontrôleur ATmega1280 pour I/O additionnel et mémoire.

- L'Arduino UNO, utilisations microcontrôleur ATmega328.

- L'Arduino Mega2560, utilisations un microcontrôleur ATmega2560, et possède toute la mémoire à 256 KBS. Elle incorpore également le nouvel ATmega8U2 (ATmega16U2 dans le jeu de puces d'USB de révision 3).

- L'Arduino Leonardo, avec un morceau ATmega3Ù4 qui élimine le besoin de raccordement d'USB et peut être employé comme clavier.

- L'ArduinoEsplora : ressemblant à un contrôleur visuel de jeu, avec un manche et des sondes intégrées pour le bruit, la lumière, la température, et l'accélération [16].

Parmi ces types, nous avons choisi une carte Arduino UNO (carte Basique). L'intérêt principal de cette carte est de faciliter la mise en œuvre d'une telle commande qui sera détaillée par la suite.

L'Arduino fournit un environnement de développement s'appuyant sur des outils open source comme interface de programmation. L'injection du programme déjà converti par l'environnement sous forme d'un code « HEX » dans la mémoire du microcontrôleur se fait d'une façon très simple par la liaison USB. En outre, des bibliothèques de fonctions "clé en main" sont également fournies pour l'exploitation d'entrées-sorties. Cette carte est basée sur un microcontrôleur ATmega 328 et des composants complémentaires. La carte Arduino contient une mémoire morte de 1 kilo. Elle est dotée de 14 entrées/sorties digitales (dont 6 peuvent être utilisées en tant que sortie PWM), 6 entrées analogiques et un cristal a 16 MHz, une connexion USB et Possède un bouton de remise à zéro et une prise jack d'alimentation. La carte est illustrée dans la figure si dessous [18].

2.3.5. Carte Arduino UNO

L'ArduinoUno est une carte microcontrôleur basée sur l'ATmega328 (datasheet). Il a 14 numérique broches d'entrée / sortie (dont 6 peuvent être utilisées comme sorties PWM), 6 entrées analogiques, une céramique 16 MHz résonateur, une connexion USB, une prise d'alimentation, un en-tête ICSP et un bouton de réinitialisation. Il contient tout nécessaire pour supporter le microcontrôleur; il suffit de le connecter à un ordinateur avec un câble USB ou l'alimenter avec un adaptateur AC-to-DC ou une batterie pour commencer. L'Uno diffère de toutes les cartes précédentes en ce sens qu'il n'utilise pas la puce de pilote FTDI USB vers série. Au lieu de cela, il dispose de l'Atmega16U2 (Atmega8U2 jusqu'à la version R2) programmé comme un USB-to-serial convertisseur.

2.3.5.1. Description de la carte

- Microcontrôleur ATmega328.
- Tension de fonctionnement 5V.
- Tension d'entrée (recommandée) 7-12V.
- Tension d'entrée (limites) 6-20V.
- E / S numériques 14 (dont 6 fournissent une sortie PWM).
- Pointes d'entrée analogiques 6.
- Courant DC par borne E / S 40 mA.
- Courant DC pour 3,3 V Pin 50 mA.
- Mémoire Flash 32 Ko (ATmega328) dont 0,5 Ko utilisé par bootloader.
- SRAM 2 Ko (ATmega328).
- EEPROM 1 KB (ATmega328).

Vitesse d'horloge 16 MHz [19].

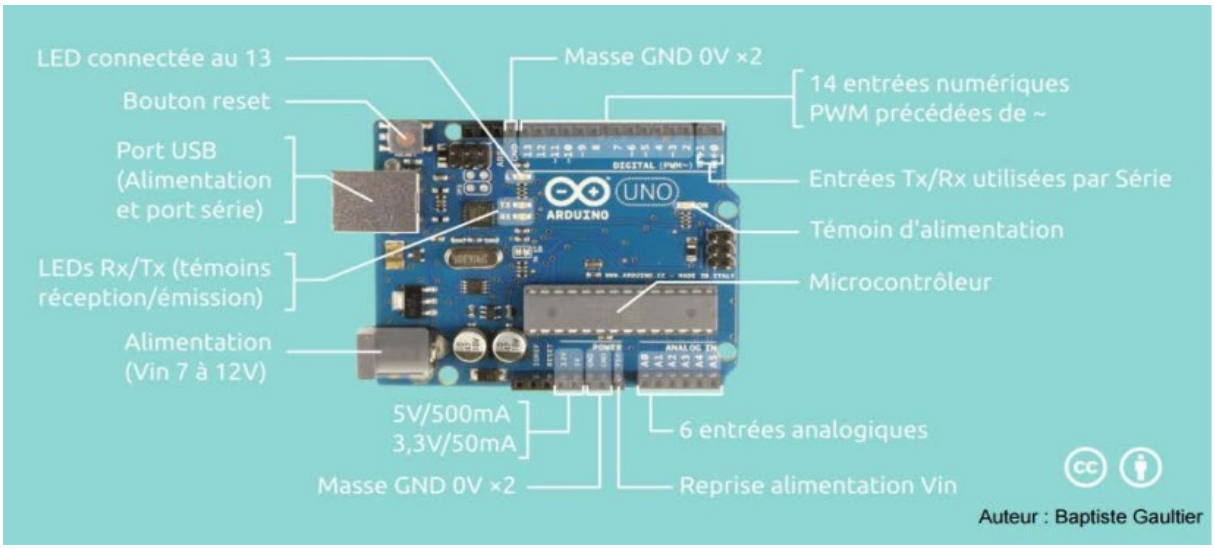

Fig 2.10 : Description de la carte ArduinoUno [17].

2.3.5.2. Puissance

L'ArduinoUno peut être alimenté via la connexion USB ou avec une alimentation externe. La puissance la source est sélectionnée automatiquement. L'alimentation externe (non USB) peut provenir d'un adaptateur AC-DC (wall-wart) ou d'une batterie. Le L'adaptateur peut être connecté en branchant une fiche positive de centre de 2,1 mm dans la prise d'alimentation de la carte. Pistes à partir d'une batterie peut être insérée dans les connecteurs Gnd et Vin du connecteur POWER. La carte peut fonctionner sur une alimentation externe de 6 à 20 volts. Si fourni avec moins de 7V, cependant, la broche 5V peut fournir moins de cinq volts et la carte peut être instable. Si vous utilisez plus de 12V, le Le régulateur de tension peut surchauffer et endommager la carte. La plage recommandée est de 7 à 12 volts [19].

2.3.5.3. Mémoire

L'ATmega328 a 32 Ko (avec 0,5 Ko utilisé pour le bootloader). Il a également 2 Ko de SRAM et 1 Ko
d'EEPROM (qui peut être lu et écrit avec la bibliothèque EEPROM) [19].

2.3.5.4. Entrée et sortie

Chacune des 14 broches numériques sur l'Uno peut être utilisée comme une entrée ou une sortie, en utilisant pinMode (), DigitalWrite () et digitalRead () fonctions. Ils fonctionnent à 5 volts. Chaque broche peut fournir ou recevoir un maximum de 40 mA et dispose d'une

résistance de pull-up interne (déconnectée par défaut) de 20-50 kOhms. Dans De plus, certaines broches ont des fonctions spécialisées:

- **Série**: 0 (RX) et 1 (TX). Utilisé pour recevoir (RX) et transmettre (TX) des données série TTL. Ces broches sont connectés aux broches correspondantes de la puce série ATmega8U2 USB-to-TTL.

- **Interruptions externes**: 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur un bas niveau. valeur, un front montant ou descendant ou un changement de valeur. Voir la fonction attachInterrupt () pour détails.

- **PWM**: 3, 5, 6, 9, 10 et 11. Fournit une sortie PWM 8 bits avec la fonction analogWrite ()
- **SPI**: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches supportent la communication SPI en utilisant la bibliothèque SPI.

- **LED**: 13. Il y a une LED intégrée connectée à la broche 13 digitale. Quand la broche est haute, le La DEL est allumée, quand la broche est BAS, elle est éteinte.

- **L'Uno a 6 entrées analogiques** : notées A0 à A5, chacune fournissant 10 bits de résolution (c.à-d.1024 valeurs différentes). Par défaut, ils mesurent de la terre à 5 volts, mais est-il possible de changer l'extrémité supérieure de leur gamme en utilisant la broche AREF et la fonction analogReference (). De plus, certains les broches ont des fonctionnalités spécialisées.

- **TWI**: broche A4 ou SDA et broche A5 ou SCL. Soutenez la communication TWI en utilisant la bibliothèque de fil.

- **AREF** :Tension de référence pour les entrées analogiques. Utilisé avec analogReference ().

- **Rset**. Apportez cette ligne LOW pour réinitialiser le microcontrôleur. Généralement utilisé pour ajouter un bouton de réinitialisation à boucliers qui bloquent celui sur le plateau [19].

2.3.5.5. la communication

L'ArduinoUno a un certain nombre de facilités pour communiquer avec un ordinateur, un autre Arduino, ou autres microcontrôleurs. L'ATmega328 fournit une communication série UART TTL (5V), qui est disponible sur les broches numériques 0 (RX) et 1 (TX). Un ATmega16U2 sur la carte canalise cette série communication via USB et apparaît comme un port de communication virtuel vers un logiciel sur l'ordinateur. Le '16U2 Le microprogramme utilise les pilotes USB COM standard et aucun pilote externe n'est nécessaire. Cependant, sur Windows, un fichier .inf est requis. Le logiciel Arduino comprend un moniteur série qui permet à des données textuelles simples de être envoyé vers et depuis le tableau Arduino. Les voyants RX et TX sur la carte clignotent lorsque les données sont en cours transmis via la puce USB-série et la connexion USB à l'ordinateur (mais pas pour communication sur les broches 0 et 1) [19].

Une bibliothèque SoftwareSerial permet la communication série sur n'importe quelle broche numérique de l'Uno. L'ATmega328 prend également en charge les communications I2C (TWI) et SPI. Le logiciel Arduino comprend un Bibliothèque de fils pour simplifier l'utilisation du bus I2C; Voir la documentation pour plus de détails. Pour la communication SPI, utilisez la bibliothèque SPI [19].

2.3.6. Téléchargement du logiciel et configuration de l’ordinateur Sur Windows

Télécharger la version Windows du logiciel Arduino ici : http://www.arduino.cc/en/Main/ Software 50 Mo environ.

- Installer le logiciel.
- Dézipper le pilote FTDI USB Drivers.zip.
- Brancher l’Arduino et pointer l’installeur Windows vers le pilote.
- Et voilà ! la carte est prête à accueillir un programme Utilisateur [20].

2.3.7. Le logiciel

C'est un logiciel de programmation par code, code qui contient une cinquantaine de commandes différentes. A l'ouverture, l'interface visuelle du logiciel ressemble à ceci: des boutons de commande en haut, une page blanche vierge, une bande noire en bas [20].

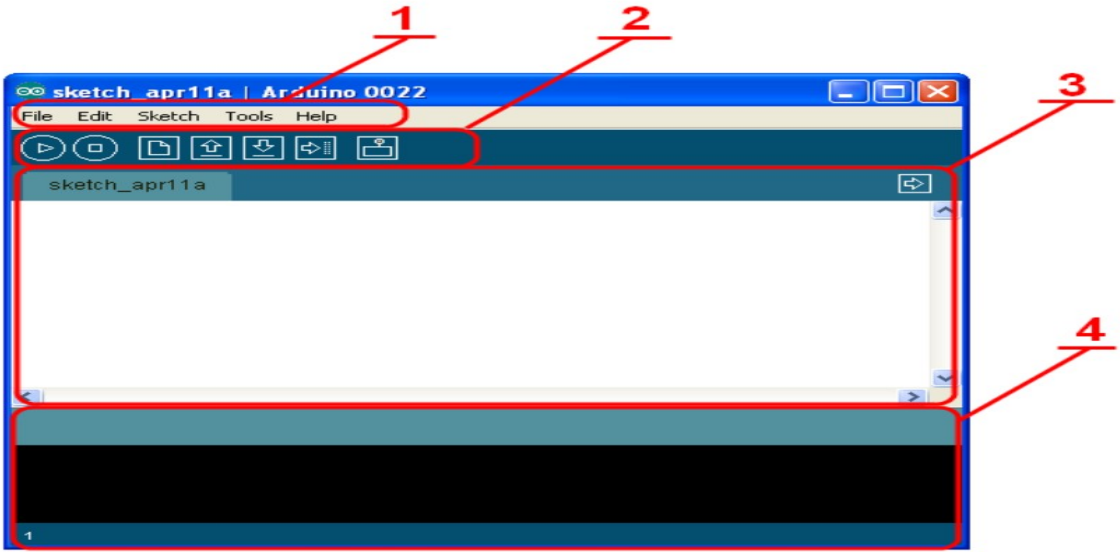

Fig 2.11 : Présentation IDE Arduino [21].

- **1:**options de configuration du logiciel.
- **2:**boutons pout la programmation des cartes.
- **3:** programme à créer.
- **5:**débogueur (affichage des erreurs de programmation).

Le menu File dispose d'un certain nombre de choses qui vont être très utiles :

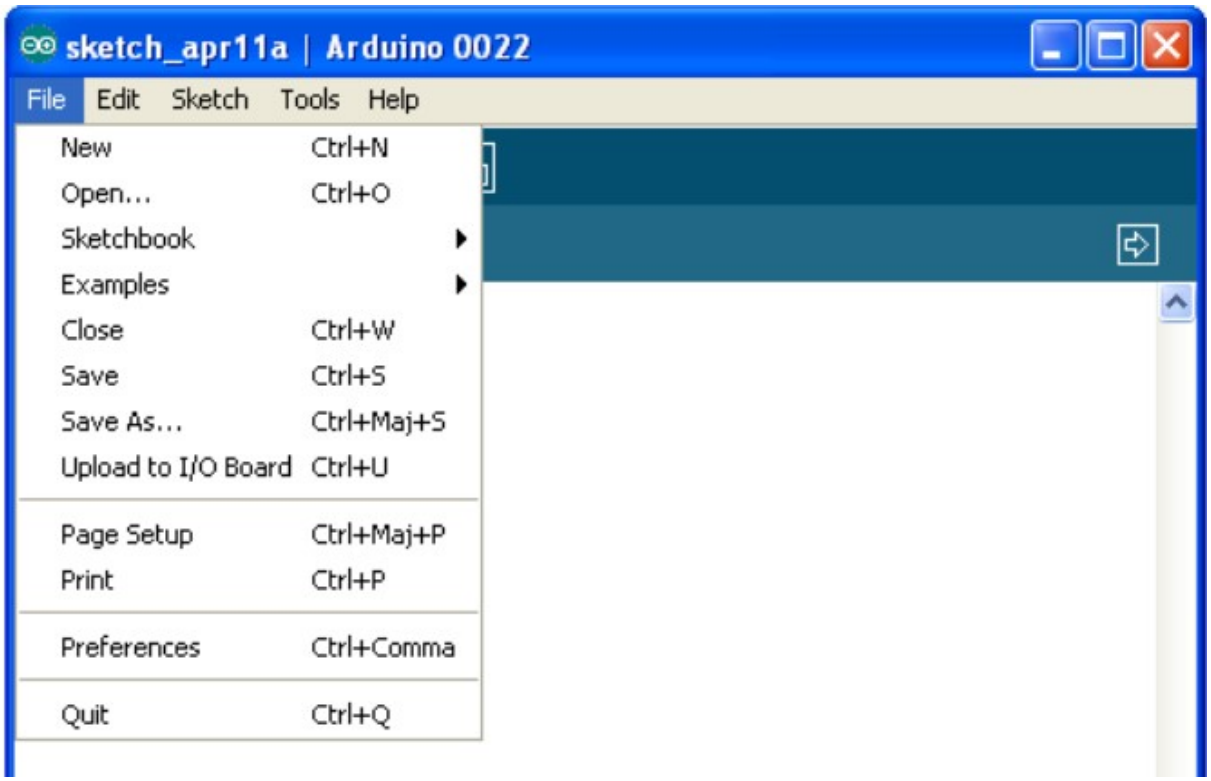

Fig 2.12:Le menu fichier sous windows[21].

- **New (nouveau) :** va permettre de créer un nouveau programme. Quand on appuie sur cebouton, une nouvelle fenêtre, identique à celle-ci, s'affiche à l'écran.

- **Open... (ouvrir) :** avec cette commande, on peut ouvrir un programme existant.
- **Save / Save as... (enregistrer / enregistrer sous...) :** enregistre le document en cours demande où enregistrer le document en cours.

- Examples (exemples) : ceci est important, toute une liste se déroule pour afficher les noms d'exemples de programmes existant.

Les boutons :

Fig 2.13 :Présentation des boutons du logisielArduino [21].

- **1 :** permet de vérifier le programme, il actionne un module qui cherche les erreurs dans le programme.
- **2 :** Créer un nouveau fichier.
- **3 :** Sauvegarder le programme en cours.
- **4 :** Liaison série.
- **5 :** Stoppe la vérification.
- **6 :** Charger un programme existant.
- **7 :** Compiler et envoyer le programme vers la carte [21].

2.3.8. Désigner le bon port Série (USB-Série)

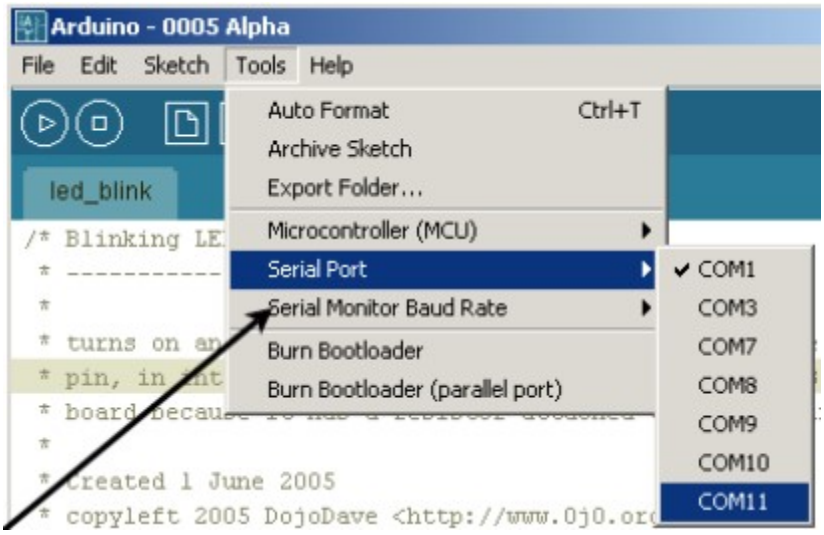

Fig 2.14 : Choisissez serial port [20].

2.3.9. Le langage Arduino

Le projet Arduino était destiné à l'origine principalement à la programmation multimédia interactive en vue de spectacle ou d'animations artistiques. C'est une partie de l'explication de la descendance de son interface de programmation de Processing. Processingest une librairie java et un environnement de développement libre. Le logiciel fonctionne sur Macintosh, Windows, Linux et Android [21].

2.4. Conclusion

Dans ce chapitre dédié à la présentation du Raspberry Pi et Arduino, Nous avons présenté les différents composants de ces deux cartes, son utilité spécialement pour notre projet sur la domotique, ensuite on a vu comment utiliser les broches pour contrôler les composants électroniques.

Chapitre 3
REALISATION ET DUSCUTION

3.1. Introduction

La réalisation est l'aboutissement logique d'un travail de conception bien fait. Le travail de développement de notre projet n'est autre que la concrétisation des besoins exprimés précédemment. Nous allons dans ce chapitre décrire brièvement les différentes étapes de la réalisation du projet, on achemine notre travail par l'environnement matériel et logiciel qui assure la réalisation de notre plateforme électronique et de ses applications.

3.2. Schéma et organigramme global de système

Dans ce travail, on a réalisé un système IoT à base de Raspberry Pi 3 comme il est montré sur la ffigure 3.1.

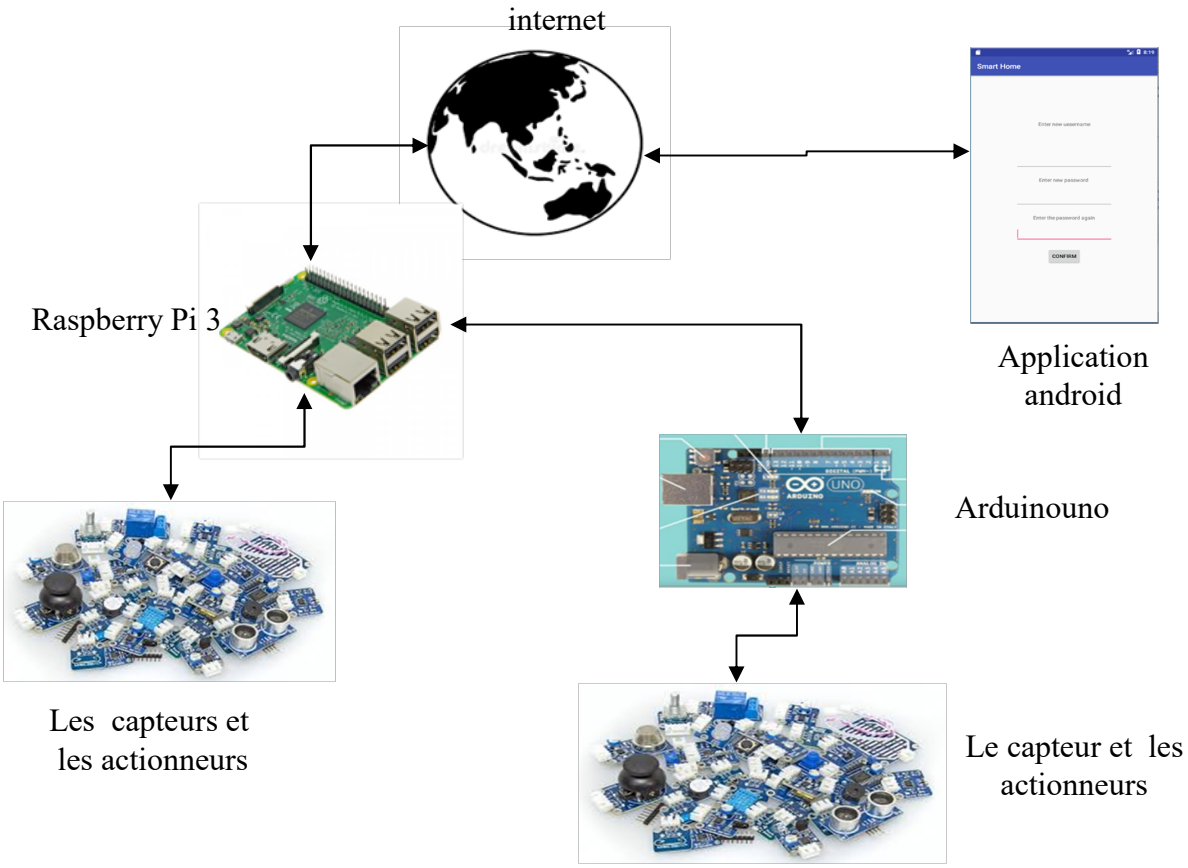

Fig.3.1 : Schéma global du système réalisé.

Nous avons utilisé l'application Android pour contrôler et demander des informations à Raspberry Pi via Internet. Raspberry, qui à son tour reçoit des commandes de l'application et met en œuvre ou envoie à Arduino et les capteurs connexes. L'Arduino garde et contrôle l'une des pièces de la maison, puis envoie les informations et reçoit les commandes de Raspberry.

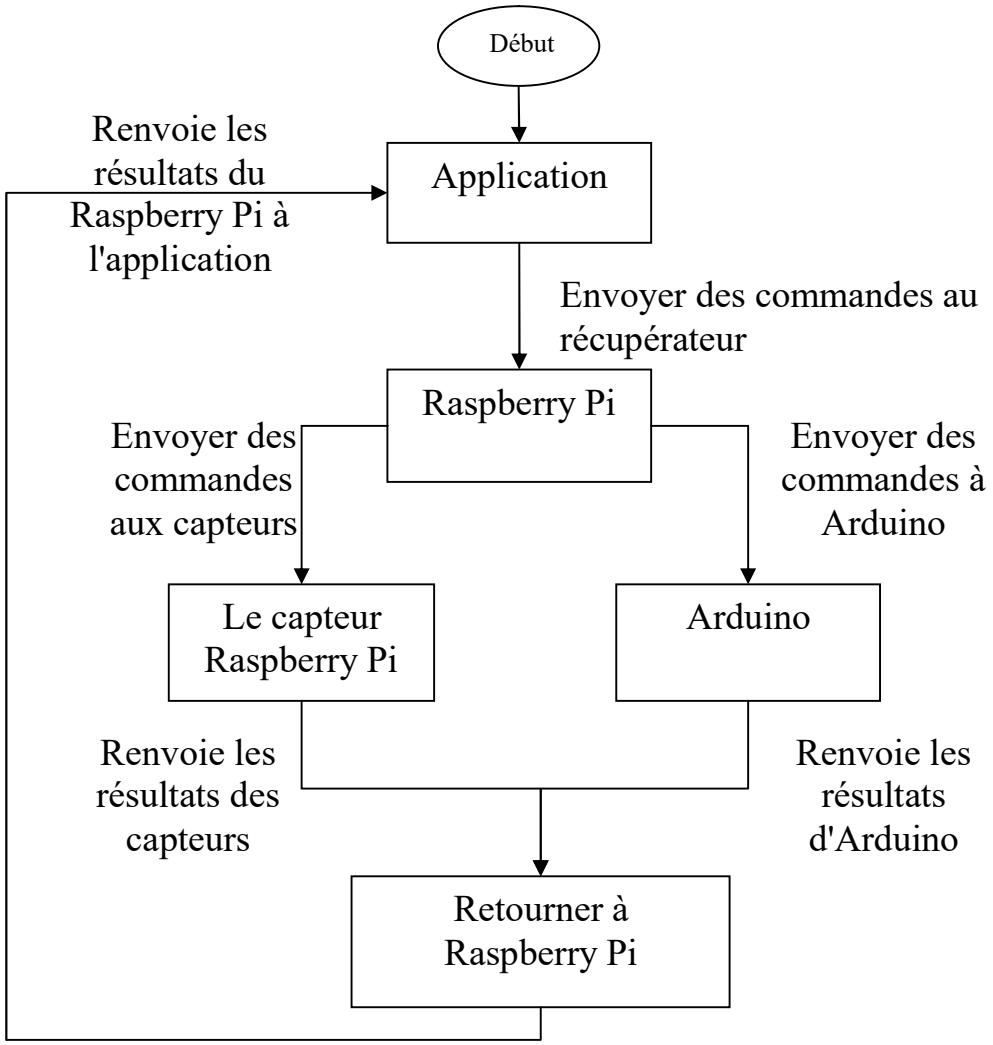

Fig.3.2 : L'organigramme global du système réalisé.

3.3. Partie matériel

3.3.1. Capteurs

3.3.1.1. Capteur de contact

C'est une sorte de bouton poussoir qui s'actionne grâce à la force exercée sur un petit levier. Il en existe de plusieurs formes. En voici quelques uns :

Fig 3.3 : Les différents types de capteurs de contact [22].

Ils font partie de la famille des capteurs tout-ou-rien. On les appelle aussi capteurs de collision, ou microrupteurs (car ils sont petits et permettent de laisser passer ou non le courant). Ils sont assez faciles à fixer mais il faut prévoir une soudure pour relier les pattes au montage [22].

3.3.1.2. Capteur de lumière

La photorésistance (LDR) est une forme de capteur de lumière (Fig.3.4). C'est un composant très simple à mettre en œuvre et qui permet une interaction intéressante avec l'environnement.

Le principe est assez simple : plus il y a de lumière, plus la résistance est basse. L'obscurité provoque une résistance importante. Il s'agit donc d'un capteur de variation qu'il faudra connecter à l'Arduino ou Raspberry Pi avec un pin analogique. Le pin analogique transforme une tension reçue entre OV et 5V reçue en valeur entre 0 et 1 024 (10 bits) [22].
le schéma de montage à réaliser pour que votre carte Arduino puisse lire correctement la photorésistance est montré sur la figure 3.4.

Fig 3.4 : Quelques photorésistances [22].

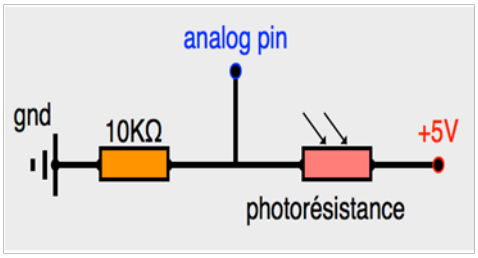

Fig 3.5 : Connexion d'une photo résistance à un pin analogique [22].

3.3.1.3. Servomoteur

A la différence d'un moteur continu, le Servomoteur ne tourne pas sur lui-même de façon continu. Un servomoteur tourne certes sur un axe, mais suivant un angle allant généralement de 0 à 180°. Certains peuvent également faire plusieurs tours, on les appelle parfois des servotreuils, d'autre ne vont qu'à 90° maximum ou encore jusque 360°, on les appelle alors servomoteur à rotation continue [23].

En réalité un servomoteur est un moteur continu équipé d'un réducteur (des engrenages), dont l'objectif est de réduire la vitesse et d'augmenter le couple (la puissance) ; et d'un potentiomètre qui permet au servomoteur de garder l'angle d'inclinaison choisit [23].

Fig 3.6 : servomoteur 0 à 180 [23].

3.3.1.4 Capteurs de température DHT11 et DH22

DHT11 (ou DHT22) est un capteur composite contient une sortie de signal numérique calibré de la température et de l'humidité. Application d'une technologie de collecte de modules numériques dédiés et de la technologie de détection de température et d'humidité, afin de garantir une grande fiabilité et une excellente stabilité à long terme du produit. Le

capteur comprend un sens résistif des composants humides et un dispositif de mesure de température NTC [24].

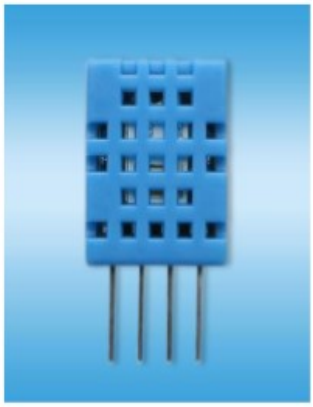

Fig 3. 7: capteur dht11 [24].

Caractéristiques

Faible coût, stabilité à long terme, mesure de l'humidité relative et de la température, excellente qualité, réponse rapide, capacité anti-interférence élevée, transmission de signal à longue distance, sortie de signal numérique et étalonnage précis [24].

3.3.1.5. Capteur de son

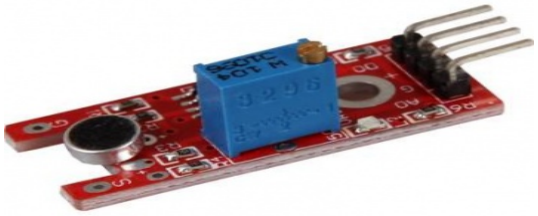

Fig 3.8 : Capteur de son [25].

Ce module est composé de trois éléments fonctionnels. Le capteur situé à l'avant du module effectue la mesure, le signal analogique est ensuite envoyé sur l'amplificateur. Celui-ci amplifie le signal en fonction du gain déterminé par le potentiomètre et envoie le signal à la sortie analogique du module [25].

Il convient de noter que le signal est inversé: plus la valeur mesurée par le capteur est haute, plus la tension de sortie est faible.La troisième partie est composée d'un comparateur

qui commute la sortie numérique et la diode lorsque le signal tombe en dessous d'une certaine valeur. La sensibilité peut être ajustée au moyen du potentiomètre comme décrit ci-dessous:

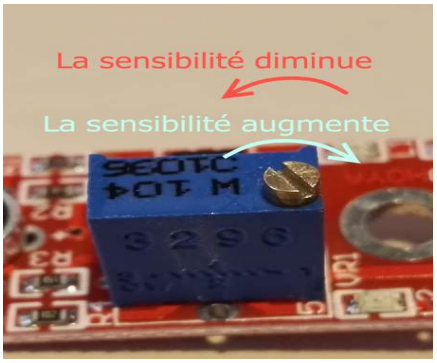

Fig 3.9 : potentiometer de la sensibilité [25].

Ce type de capteur ne délivre pas des valeurs absolues mais des valeurs relatives. On définit une valeur limite par rapport à une valeur normale donnée et le module émet un signal si cette limite est dépassée [25].

3.3.1.6. Capteur de gaz MQ-9

Le module Grove-GasSensor (MQ9) est utile pour détecter les fuites de gaz (à la maison et dans l'industrie). Il peut détecter LPG, CO et CH4. Basé sur son temps de réponse rapide. Les mesures peuvent être prises dès que possible. La sensibilité peut également être ajustée par le potentiometer [26].

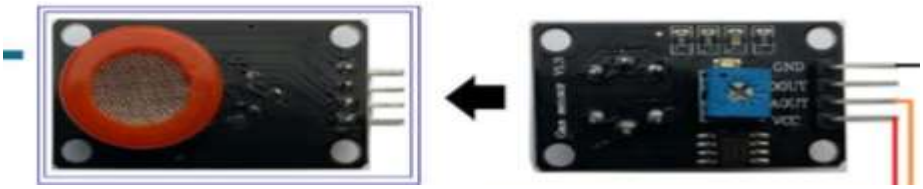

Fig 3.10 : capteur de gaz MQ-9 [26].

3.3.2. Raspberry pi

Dans notre projet, nous avons utilisé le Raspberry Pi 3 pour être un intermédiaire entre l'application utilisateur et les appareils ménagers.

3.3.2.1. Connexion entre Arduino et Raspberry

Pour la connexion entre Arduino et Raspberry nous avons utilisé le protocole "I2C" où nous avons utilisé pour le Raspberry les pins GPIO 8 ,GPIO 9 et pour Arduino les pin "A5" et "A4" comme le montre la figure 3.11 .

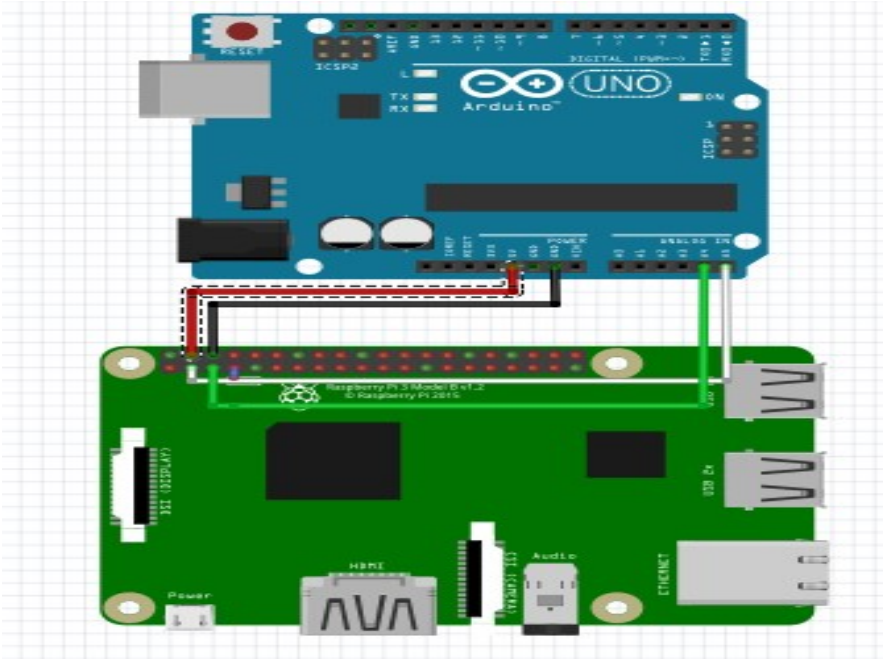

Fig 3.11 : Connexion entre Arduino et Raspberry Pi 3.

3.3.2.2 Contrôle d'éclairage

L'éclairage est nécessaire dans la maison, nous avons donc mis en place dans l'application Raspberry un panneau de commande pour l'éclairage ou bien manuellement par un coupeur électrique.

Nous avons utilisé comme matériel :

_Lampe 220v, Il est connecté au GPIO 13 à travers un relais.

-Capteur de lumière: Assurant que la lampe fonctionne ou non, Il est connecté au GPIO 8.

-Interrupteur électrique : pour allumer et éteindre la lampe, Il est connecté au GPIO 11.

-Relais 5v avec son circuit de commande.

La figure 3.12 montre la boucle de connexion du circuit :

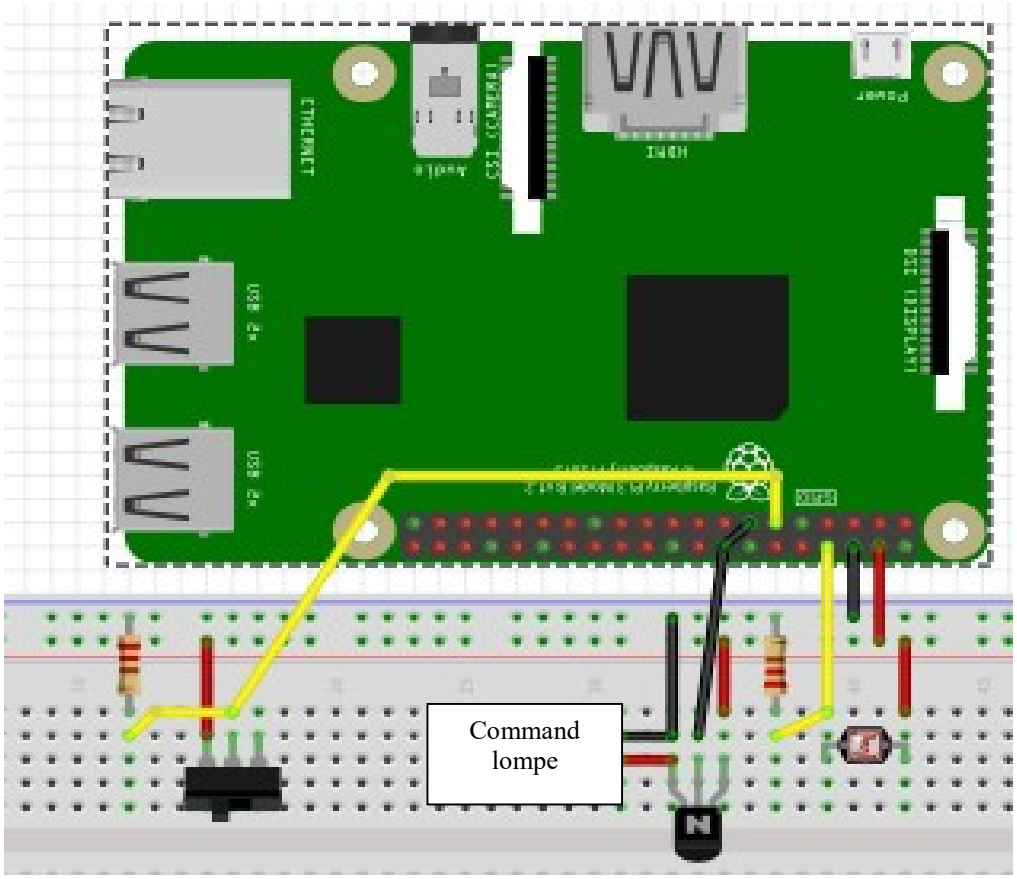

Fig 3.12 : systeme d'éclairage.

3.3.2.3 Protection

La sécurité est devenue un élément clé dans le choix d'une maison. Et l'une des plus grandes craintes de vol et de cambriolage. Par conséquent, nous avons intégré divers composants pour détecter un intrus, en ouvrant les portes et les fenêtres.

Nous avons utilisé comme matériel :

-capteur de son, Il est connecté au GPIO 13.

-Capteur de contact de porte, Il est connecté au GPIO 15.

-Le capteur de contact pour la fenêtre, Il est connecté au GPIO 16.

La figure suivante montre la boucle de connexion du circuit :

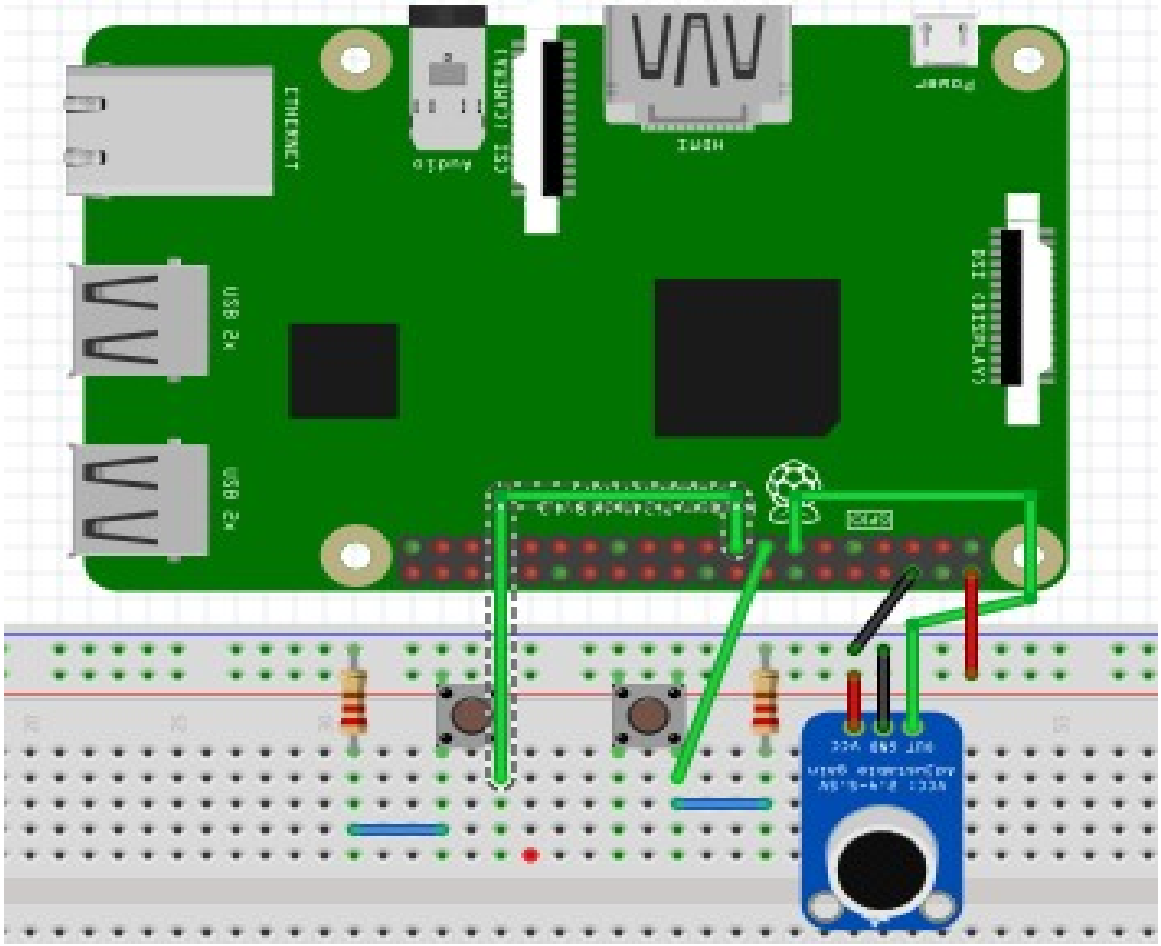

Fig 3.13 : Connexion de systèmedeprotection.

3.3.3. Fonctions d'Arduino

3.3.3.1. Contrôle de la température

La dépense de chauffage ou de climatisation est une part importante du budget énergétique. Ainsi, afin de réduire ce coût, nous avons créé un dispositif prenant en charge l'ouverture et la fermeture des fenêtres en fonction d'une température réglée par l'utilisateur, Il ouvre ou ferme la fenêtre pour ajuster la température ambiante.

Nous avons utilisé comme matériel :

- capteurs DHT 22: Afin de mesurer la température, Il est connecté au port Arduino 6..

- Un Servomoteur:Pour ouvrir et fermer la fenêtre, Il est connecté au port Arduino 9.

-Le ventilateur est utilisé pour le refroidissement, Il est connecté au port Arduino 11.

La figure 3.14 montre la boucle de connexion du circuit :

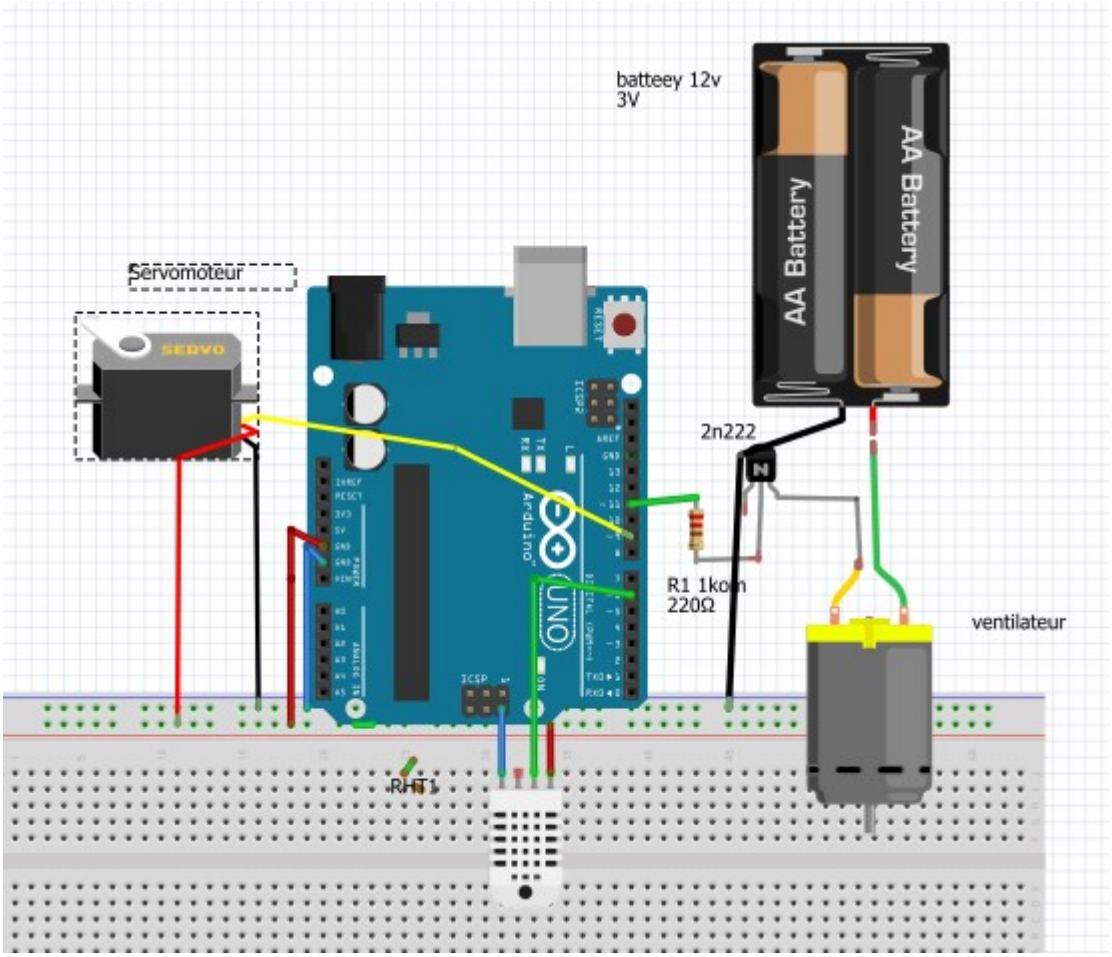

Fig 3.14 : Connexion de système de ventilation.

3.3.3.2. Surveillance du gaz

La sécurité est devenue un élément primordial dans le choix d'une maison. Et l'une des plus grande crainte d'accident reste l'incendie et étouffement. Ainsi nous avons associé différents composants afin de créer un détecteur de gaz. Toute fuite de gaz dans l'air de la pièce va ouvrir la fenêtre en utilisant un relais statique réalisé à base des transistors de puissance.

Nous avons utilisé comme matériel :

- capteur de gaz, Il est connecté au port Arduino A2.

-Un Servomoteur:Pour ouvrir et fermer la fenêtre, Il est connecté au port Arduino 9.

La figure 3.15 montre la boucle de connexion du circuit :

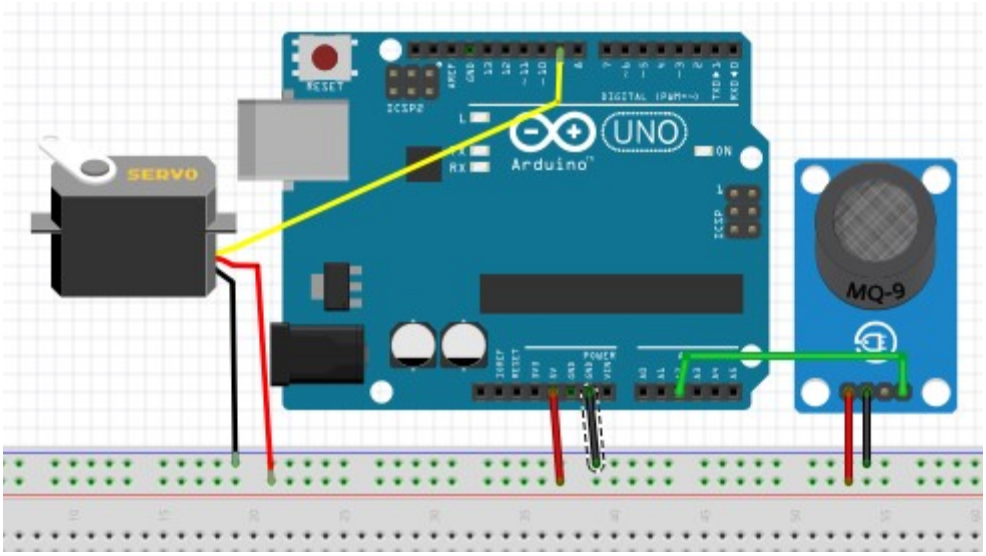

Fig 3.15 : systeme Surveillance du gaz.

3.3.3.3. Contrôled'éclairage

L'éclairage est nécessaire dans la maison, nous avons donc mis en place dans l'application Android un panneau de commande pour l'éclairage ou bien manuellement par un coupeur électrique.

Nous avons utilisé comme matériel :

_Lampe 220v, Il est connecté au port Arduino 13 à travers un relais.

-Capteur de lumière: Assurant que la lampe fonctionne ou non, Il est connecté au port Arduino A0.

-Interrupteur électrique :pour allumer et éteindre la lampe, Il est connecté au port Arduino .

La figure 3.16 montre la boucle de connexion du circuit :

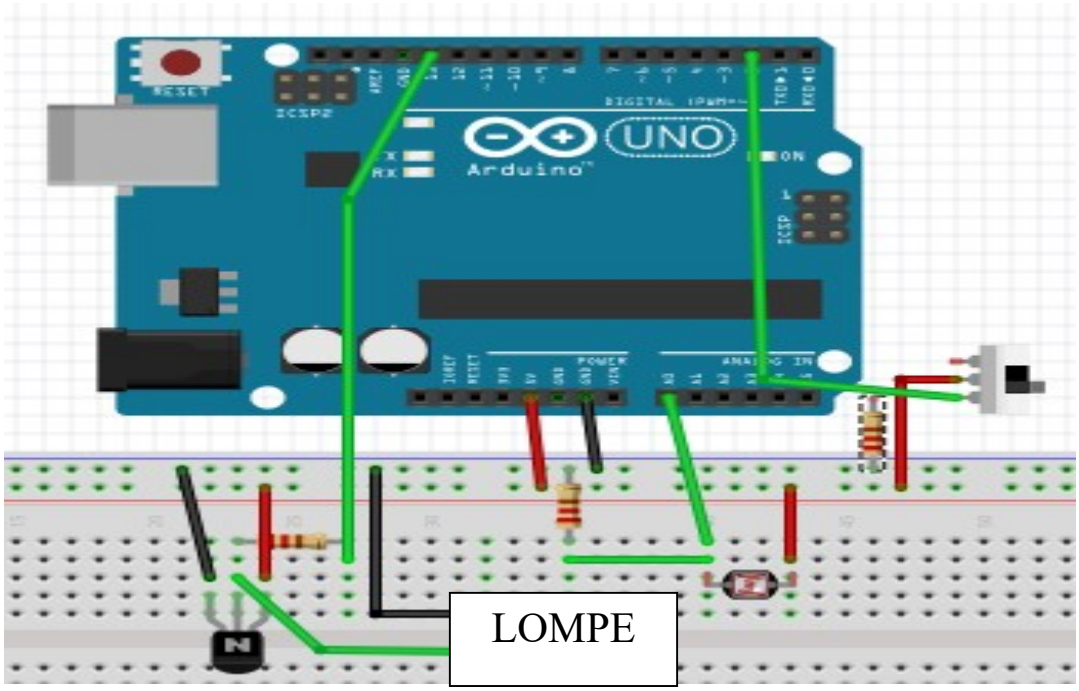

Fig 3.16 : systeme d'éclairage.

3.3.3.4 Protection

La sécurité est devenue un élément clé dans le choix d'une maison. Et l'une des plus grandes craintes de vol et de cambriolage. Par conséquent, s nouavons intégré divers composants pour détecter un intrus, en ouvrant les portes et les fenêtres.

Nous avons utilisé comme matériel :

-capteur de son, Il est connecté au port Arduino 8.

- capteur de contact de porte, Il est connecté au port Arduino 4.

-Le capteur de contact pour la fenêtre, Il est connecté au port Arduino 7.

La figure 3.17 montre la boucle de connexion du circuit :

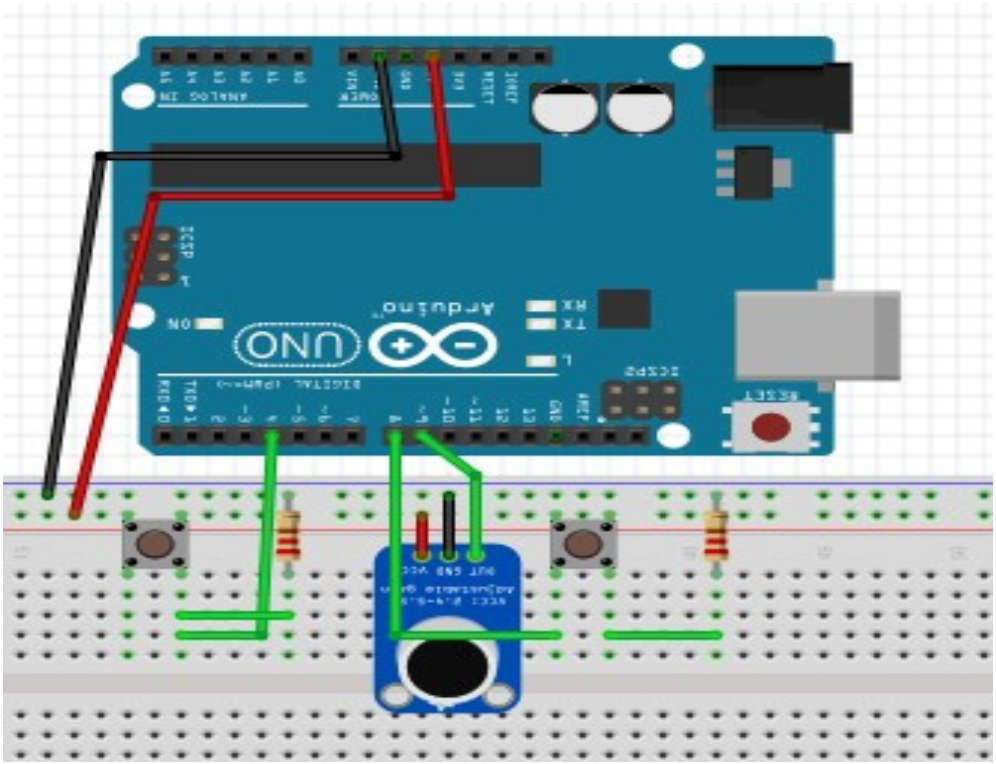

Fig 3.17 : systeme de protection.

3.4. Partie logiciel

Une application mobile est un type de logiciel ou programme conçu pour s'exécuter sur un appareil mobile, tel qu'un Smartphone ou une tablette. Les applications mobiles servent souvent à fournir aux utilisateurs des services similaires à ceux du PC.

3.4.1. Android Studio

Android Studio fournit des outils pour les phases de test et de publication du processus de développement, ainsi qu'un environnement de développement unifié pour la création d'applications pour tous les appareils Android. L'environnement de développement inclut des modèles de code avec un exemple de code pour les fonctionnalités d'application communes, des outils et des cadres de test étendus et un système de génération flexible [27].

Android Studio propose :

- un environnement de développement robuste.

- un moyen simple de tester les performances sur d'autres types de périphériques.

- Assistants et modèles pour les éléments communs trouvés sur tous les programmeurs Android.

- un éditeur complet avec une gamme d'outils pour accélérer le développement de votre application [27].

3.4.1.1. Ouvrir un nouveau projet

Après avoir installé l'IDE pour Android Studio, double-cliquez sur l'icône de l'application Android Studio pour le démarrer. Comme le montre la figure 3.18. Choisissez Démarrer un nouveau projet Android Studio dans la fenêtre de bienvenue et nommez le projet avec le même nom. Pour l'application.

Lorsque vous choisissez un domaine d'entreprise unique, gardez à l'esprit que les applications publiées sur Google Play doivent avoir un nom de package unique. Étant donné que les domaines sont uniques, le nom de l'application précédé de votre nom ou du nom de domaine de votre société doit fournir un nom de package suffisamment unique. Si vous ne prévoyez pas de publier l'application, vous pouvez accepter le domaine d'exemple par défaut.

Fig 3.18 : écran Android Studio.

Spécifier le nom de l'application, Dans la fenêtre suivante lorsque vous cliquez sur un nouveau projet dans la fenêtre précédente, comme indiqué dans la Figure 3.19 :

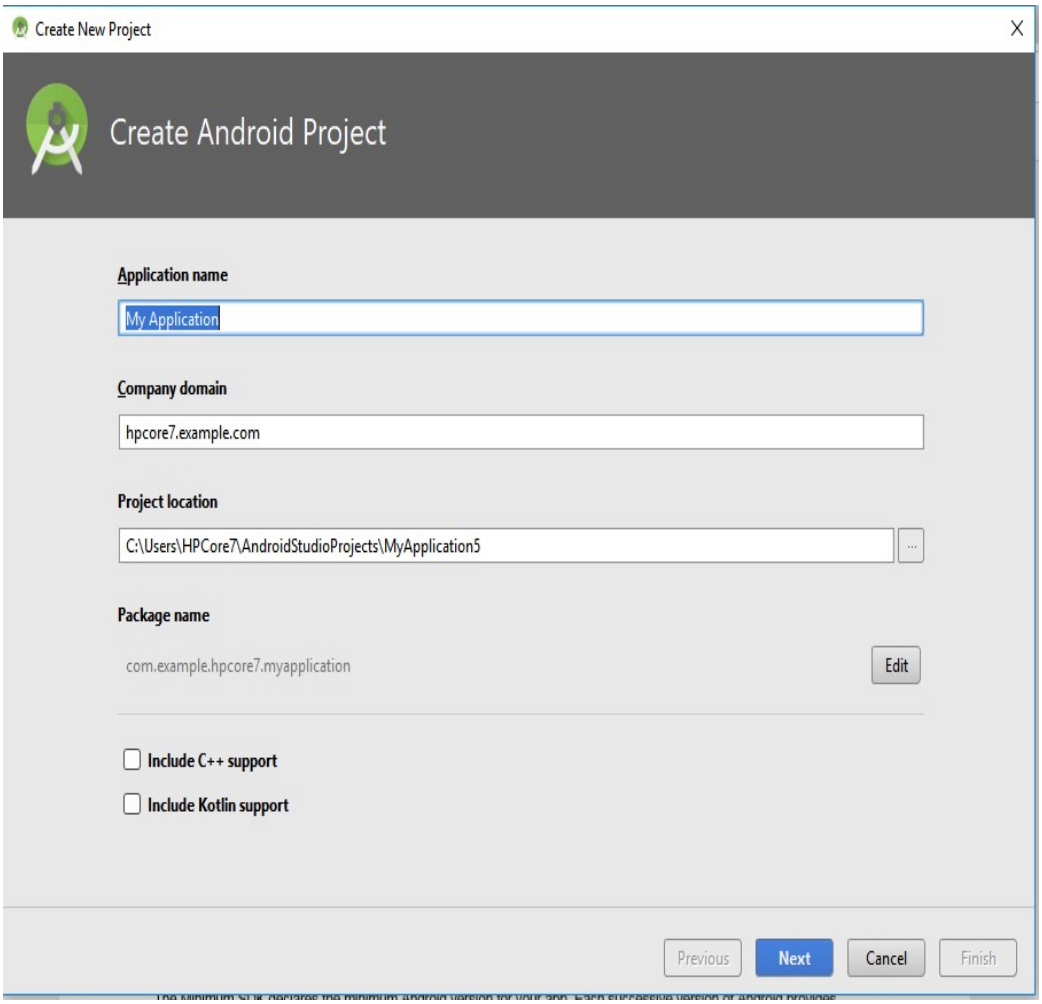

Fig 3.19: Configurer l'écran du nouveau projet.

➢ Choisir les appareils cibles et le SDK minimum

Lors du choix des appareils Android cibles, le téléphone et la tablette sont sélectionnés par défaut, comme illustré dans la figure ci-dessous. Le choix montré dans la figure pour le SDK minimum - API 15: Android 4.0.3 (IceCream Sandwich) - rend votre application compatible avec 97% des appareils Android actifs sur le Google Play Store.

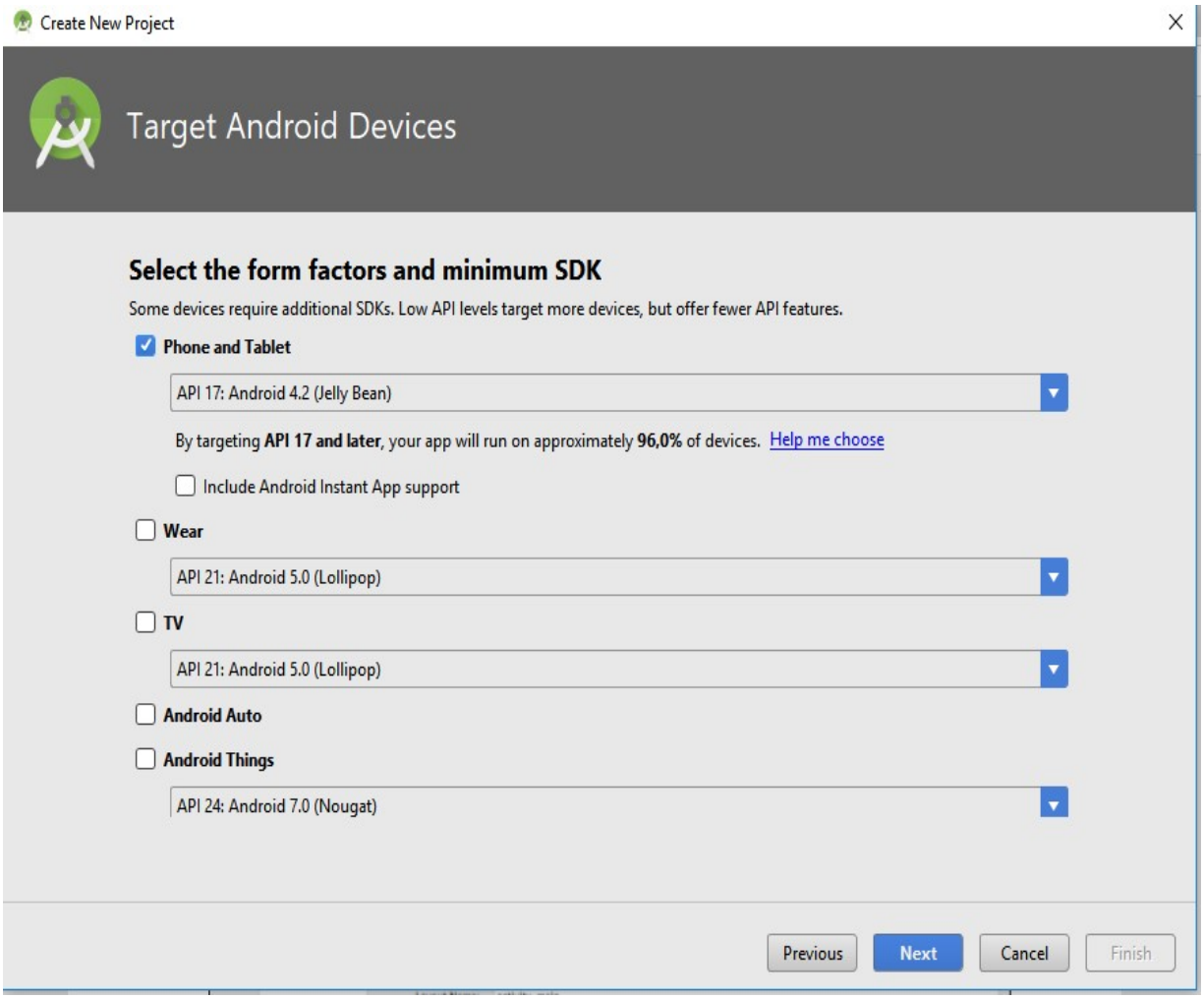

Fig 3.20 : Écran Cibler les appareils Android.

Différents appareils exécutent différentes versions du système Android, comme Android 4.0.3 ou Android 4.4. Chaque version successive ajoute souvent de nouvelles API non disponibles dans la version précédente. Pour indiquer quel ensemble d'API est disponible, chaque version spécifie un niveau d'API. Par exemple, Android 1.0 est le niveau d'API 1 et Android 4.0.3 est le niveau d'API 15.

Le SDK minimum déclare la version Android minimale pour votre application. Chaque version successive d'Android offre une compatibilité pour les applications qui ont été créées à l'aide des API des versions précédentes. Votre application doit donc toujours être compatible avec les futures versions d'Android tout en utilisant les API Android documentées.

Sélectionnez l'interface d'application appropriée:

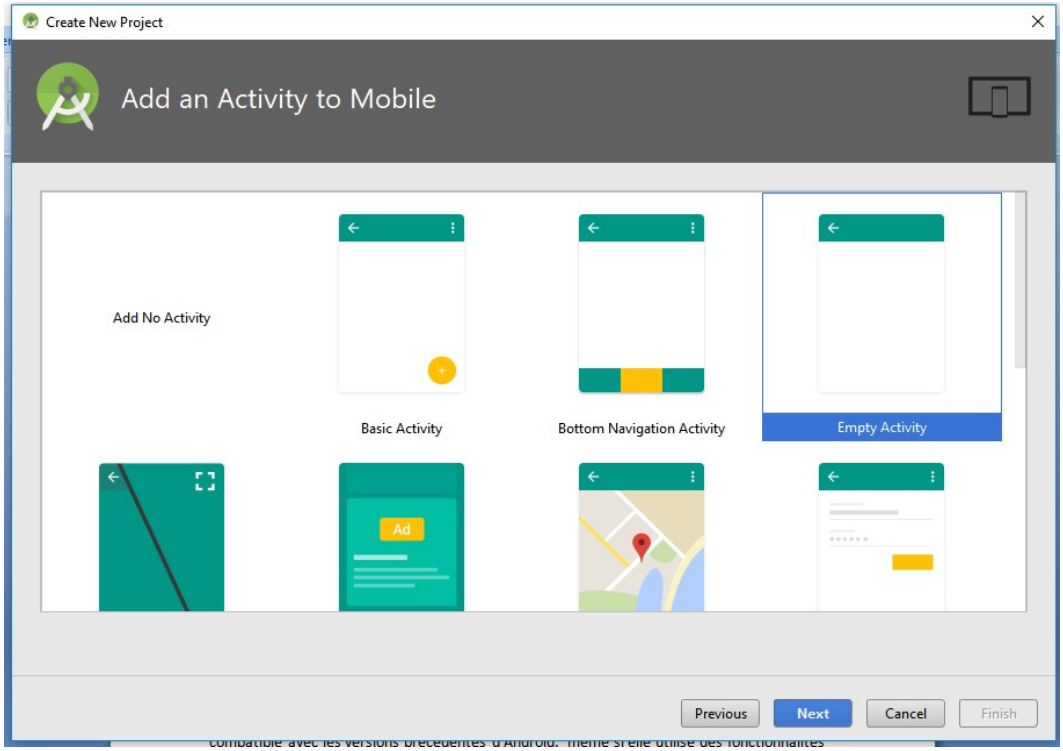

Fig 3.21 : Ajouter une activité à Mobile Choisissez.

Pouvez personnaliser l'activité après avoir choisi votre modèle dans la fenêtre précédente. Par exemple, le modèle Activité vide fournit une activité unique accompagnée d'une ressource de mise en page unique pour l'écran.

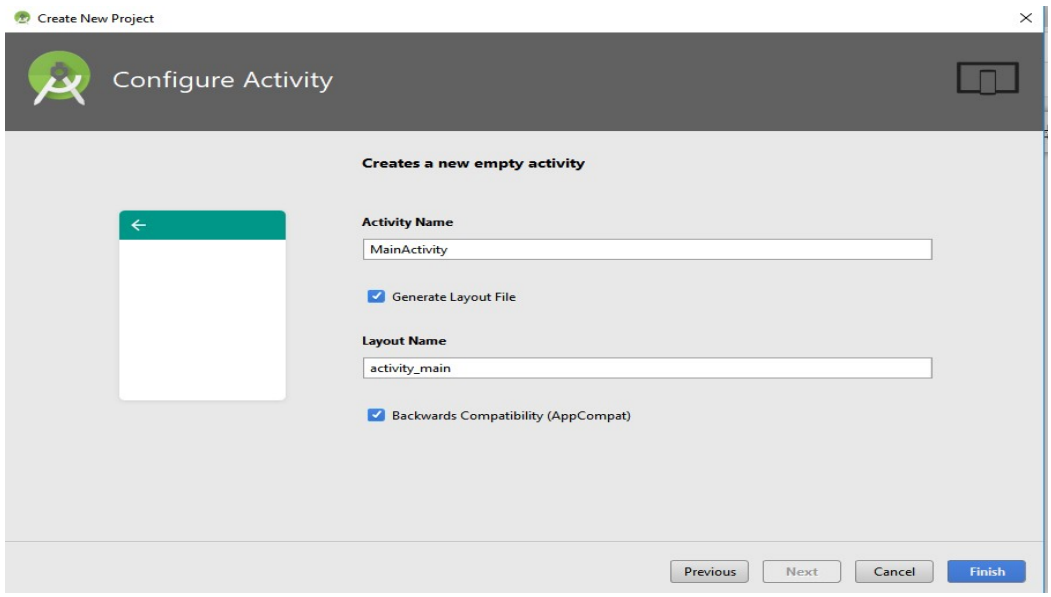

Fig 3.22 : personnaliser l'activité.

3.4.1.2. L'interface d'Android Studio

La fenêtre principale d'Android Studio est composée de plusieurs zones logiques, ou panneaux, comme indiqué dans la figure 3.23.

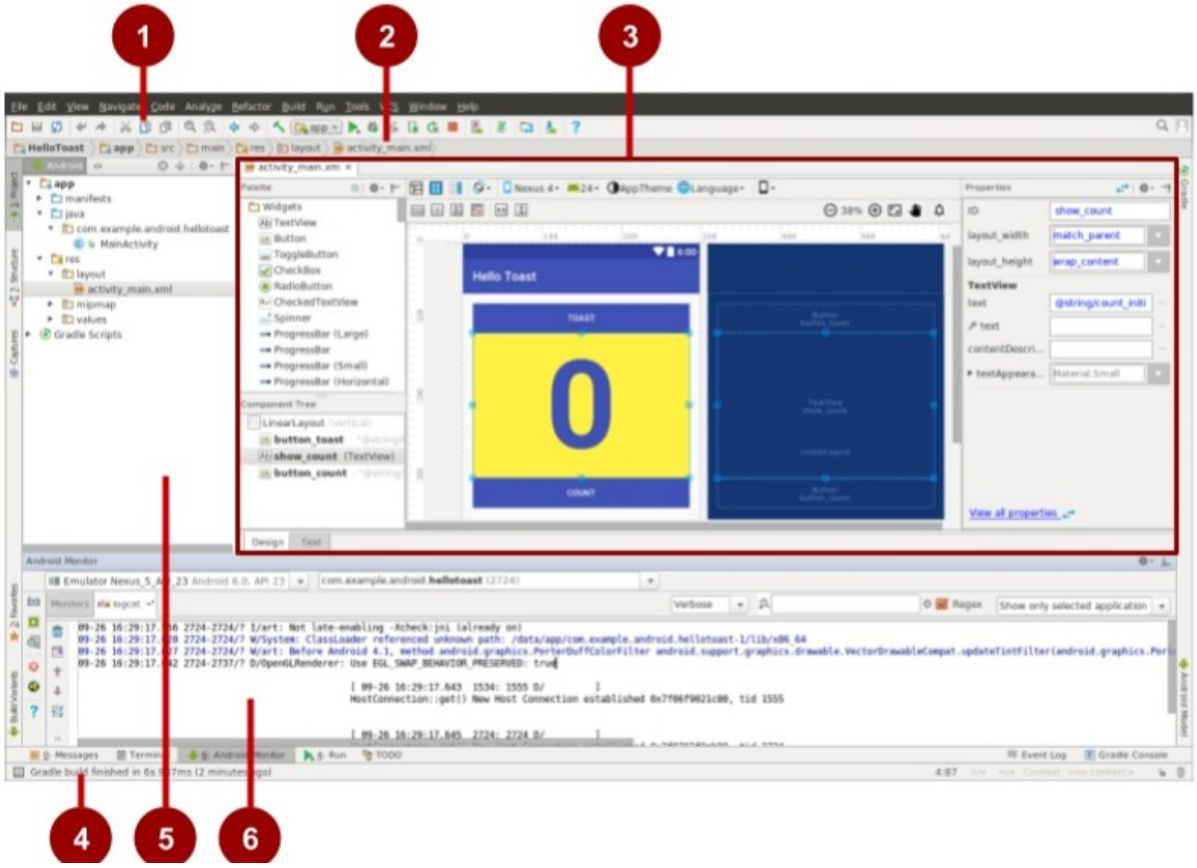

Fig 3.23 : L'interface d'Android Studio.

Dans la figure ci-dessus:

1. La barre d'outils. La barre d'outils effectue un large éventail d'actions, y compris l'exécution de l'application Android et le lancement d'outils Android.

2. La barre de navigation. La barre de navigation permet de naviguer dans le projet et d'ouvrir les fichiers pour les modifier. Il fournit une vue plus compacte de la structure du projet.

3. Le volet de l'éditeur. Ce volet affiche le contenu d'un fichier sélectionné dans le projet.
Par exemple, après avoir sélectionné une mise en page (comme le montre la figure), ce volet affiche l'éditeur de mise en page avec des outils pour modifier la mise en page. Après avoir sélectionné un fichier de code Java, ce volet affiche le code avec des outils pour éditer le code.

4. La barre d'état. La barre d'état affiche l'état du projet et d'Android Studio lui-même, ainsi que tous les avertissements ou messages. Vous pouvez regarder la progression de la construction dans la barre d'état.

5. Le volet Projet. Le volet projet affiche les fichiers du projet et la hiérarchie du projet.

6. Le volet Monitor. Le volet de contrôle permet d'accéder à la liste TODO pour la gestion des tâches, à Android Monitor pour surveiller l'exécution de l'application (illustrée dans la figure), au logcat pour l'affichage des messages de journal et à l'application Terminal pour effectuer des activités Terminal.

Pour revenir à cette vue depuis une autre vue, cliquez sur l'onglet Projet vertical dans la colonne située à l'extrême gauche de la fenêtre Projet, puis choisissez Android dans le menu local situé en haut du volet Projet, comme indiqué dans la figure 3.24.

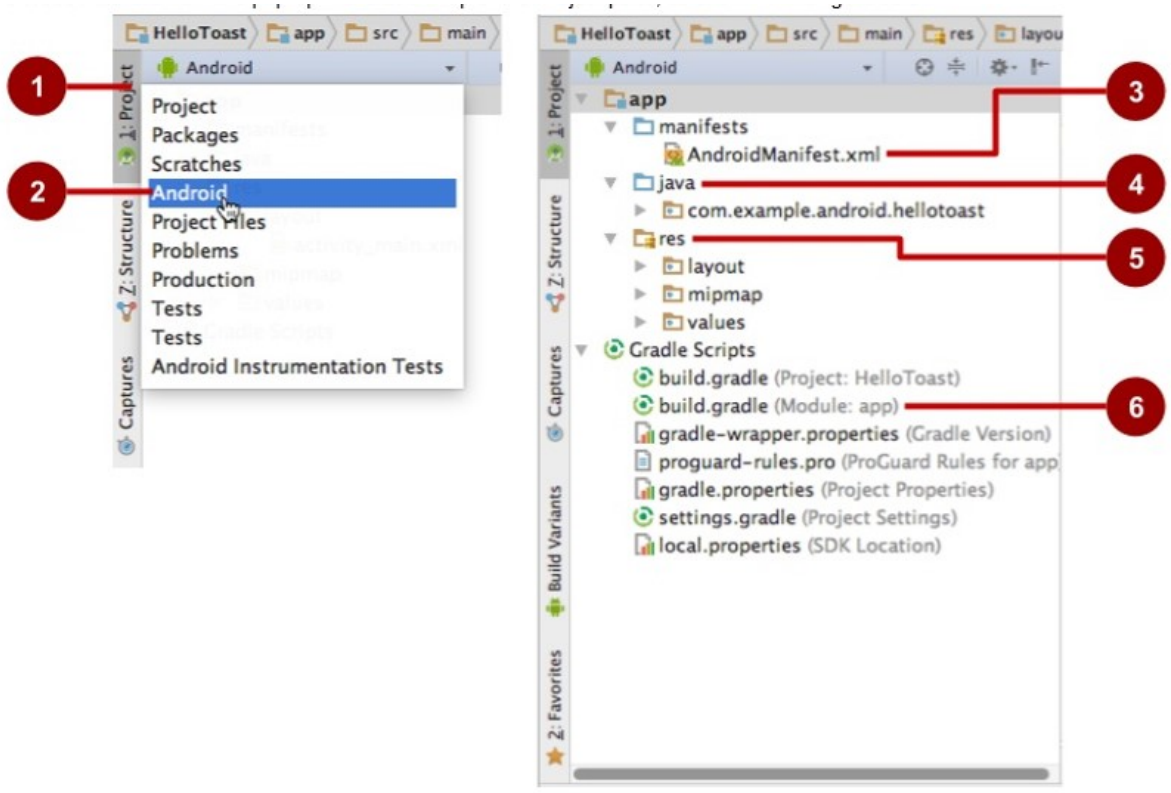

Fig 3.24 : Présentation de la vue Android.

Dans la figure ci-dessus:

1. L'onglet Projet. Cliquez pour afficher la vue du projet.
2. La sélection Android dans le menu déroulant du projet.

3. Le fichier AndroidManifest.xml. Utilisé pour spécifier des informations sur l'application pour l'environnement d'exécution Android. Le modèle que vous choisissez crée ce fichier.

4. Le dossier java. Ce dossier inclut des activités, des tests et d'autres composants dans le code source Java. Chaque activité, service et autre composant est défini comme une classe Java, généralement dans son propre fichier. Le nom de la première activité (écran) que l'utilisateur voit, qui initialise également les ressources à l'échelle de l'application, est habituellement MainActivity

. 5. Le dossier res. Ce dossier contient des ressources, telles que des dispositions XML, des chaînes d'interface utilisateur et des images. Une activité est généralement associée à un fichier de ressources XML qui spécifie la disposition de ses vues. Ce fichier est généralement nommé après son activité ou fonction.

6. Le fichier build.gradle (Module: App). Ce fichier spécifie la configuration de construction du module. Le modèle que vous choisissez crée ce fichier qui définit la configuration de construction, y compris l'attribut minSdkVersion qui déclare la version minimale de l'application et l'attribut targetSdkVersion qui déclare la version la plus récente (la plus récente) pour laquelle l'application a été optimisée. Ce fichier inclut également une liste de dépendances, qui sont des bibliothèques requises par le code, telles que la bibliothèque AppCompat pour prendre en charge un large éventail de versions d'Android [66].

3.4.1.3. Introduction à l'éditeur de layout

L'éditeur de disposition apparaît lorsque vous ouvrez un fichier de mise en page XML. Correspondant aux chiffres de la figure 3.25, les régions de l'éditeur sont les suivantes:

1 Palette: liste des vues et des groupes de vue que vous pouvez faire glisser dans votre mise en page.

2 Arborescence des composants: affiche la hiérarchie de votre mise en page.

3 Barre d'outils: Boutons permettant de configurer l'apparence de votre disposition dans l'éditeur et de modifier certains attributs de disposition.

4 Éditeur de conception: mise en page en conception ou en vue Blueprint, ou les deux.
5 Attributs: contrôles pour les attributs de la vue sélectionnée [29].

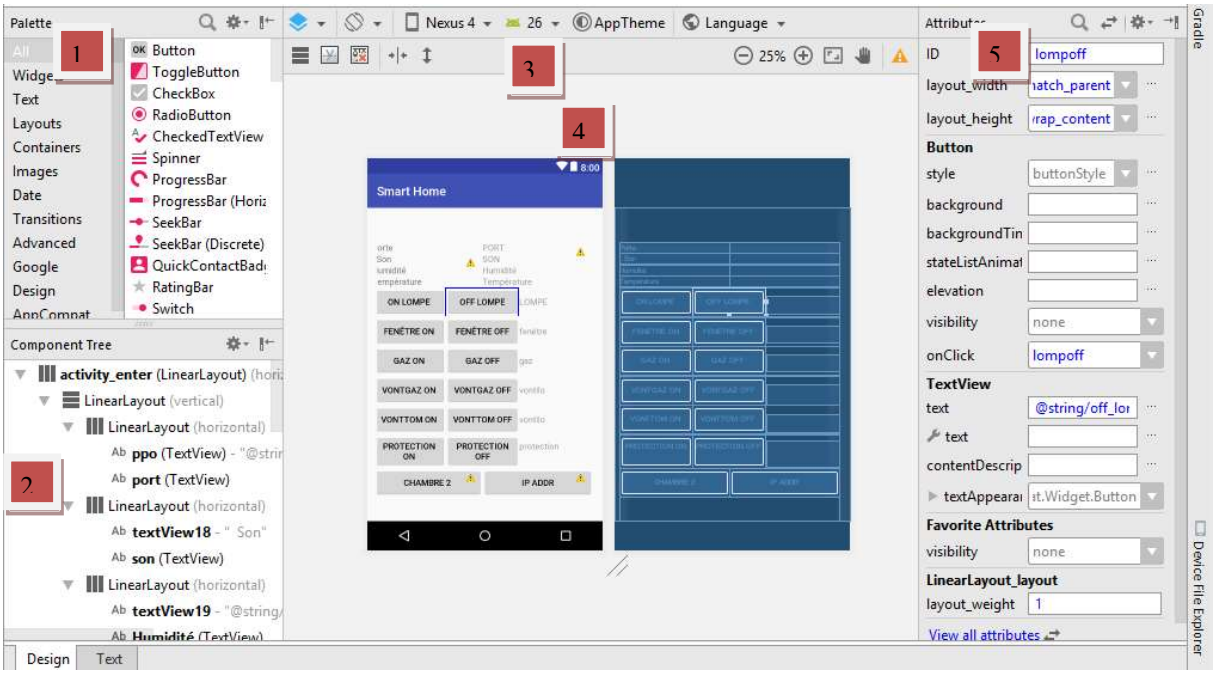

Fig 3.25 : L'éditeur de disposition.

Lorsque vous ouvrez un fichier de mise en page XML, l'éditeur de conception s'ouvre par défaut (comme illustré à la figure 3.26).

Pour modifier la présentation XML dans l'éditeur de texte, cliquez sur l'onglet Texte en bas de la fenêtre. Dans l'éditeur de texte, vous pouvez également afficher la palette, l'arborescence des composants et l'éditeur de conception en cliquant sur Aperçu dans la partie droite de la fenêtre. La fenêtre Attributs n'est pas disponible dans l'éditeur de texte.

Les boutons de la rangée supérieure de l'éditeur de conception vous permettent de configurer l'apparence de votre mise en page dans l'éditeur. Cette barre d'outils est également disponible dans la fenêtre de prévisualisation de l'éditeur de texte [29].

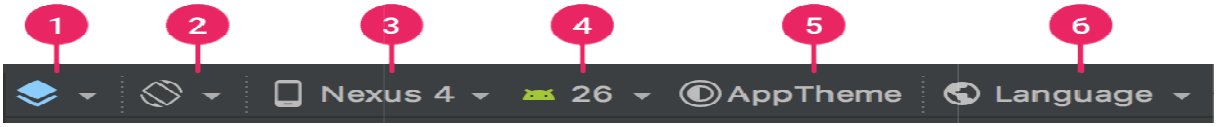

Fig 3.26: Buttons in the Layout Editor Toolbar that configure the layout appearance [29].

1 Conception et plan: sélectionnez le mode d'affichage de votre mise en page dans l'éditeur. Sélectionnez soit la vue Création (un aperçu du monde réel de votre disposition), la vue Plan (uniquement les contours pour chaque vue) ou Design + Plan pour les deux côte à côte.

2 Orientation de l'écran et variantes de mise en page: sélectionnez l'orientation de l'écran en mode paysage ou portrait ou d'autres modes d'écran pour lesquels votre application propose des dispositions alternatives, telles que le mode nuit. Ce menu contient également des commandes pour créer une nouvelle variante de disposition.

3 Type et taille de l'appareil: sélectionnez le type d'appareil (téléphone / tablette, Android TV ou Wear OS) et la configuration de l'écran (taille et densité). Vous pouvez choisir parmi plusieurs types de périphériques préconfigurés et vos propres définitions AVD ou démarrer un nouveau fichier AVD en sélectionnant Ajouter une définition de périphérique dans la liste.

4 Version de l'API: Sélectionnez la version d'Android sur laquelle afficher votre mise en page.

5 Thème de l'application: sélectionnez le thème de l'interface utilisateur à appliquer à l'aperçu. (Cela ne fonctionne que pour les styles de mise en page pris en charge, donc de nombreux thèmes de cette liste entraînent une erreur) .

6 Langue: sélectionnez la langue à afficher pour vos chaînes d'interface utilisateur. Cette liste n'affiche que les langues disponibles dans vos ressources de chaîne. Si vous souhaitez modifier vos traductions, cliquez sur Modifier les traductions dans le menu déroulant (voir Localisation de l'interface utilisateur avec l'éditeur de traductions) [29].

3.4.1.4. Ajouter un composant de projet

La liste des modèles fournis dans Android Studio ne cesse de croître. Android Studio regroupe les modèles en fonction du type de composant qu'ils ajoutent, tel qu'une activité ou un fichier XML, comme indiqué dans la figure 3.27 [30].

Pour ajouter un composant de projet Android à l'aide d'un modèle, utilisez la fenêtre Projet. Cliquez avec le bouton droit sur le dossier dans lequel vous souhaitez ajouter le nouveau composant et sélectionnez Nouveau. En fonction des composants qui peuvent être ajoutés au dossier sur lequel vous avez cliqué, une liste de types de modèles s'affiche, comme ceux illustrés à la figure 3.28 [30].

Lorsque vous sélectionnez le modèle que vous souhaitez ajouter, une fenêtre d'assistant correspondante s'affiche et vous demande les informations de configuration du composant, telles que son nom. Après avoir entré les informations de configuration, Android Studio crée

et ouvre les fichiers de votre nouveau composant. Il exécute également une construction Gradle pour synchroniser votre projet [30].

Bien que vous puissiez également utiliser le menu Fichier> Nouveau d'Android Studio pour créer un nouveau composant de projet Android, la navigation vers le dossier de votre choix dans la fenêtre Projet garantit que vous créez le composant au bon endroit [30].

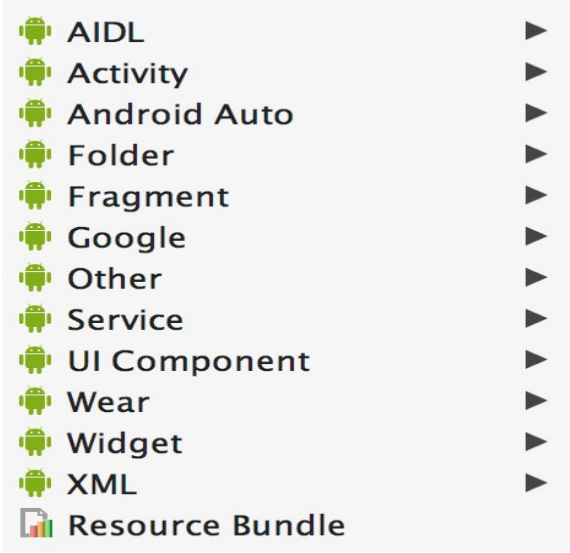

Fig3.27 : Le menu des modèles, accessible via le menu Fichier [30].

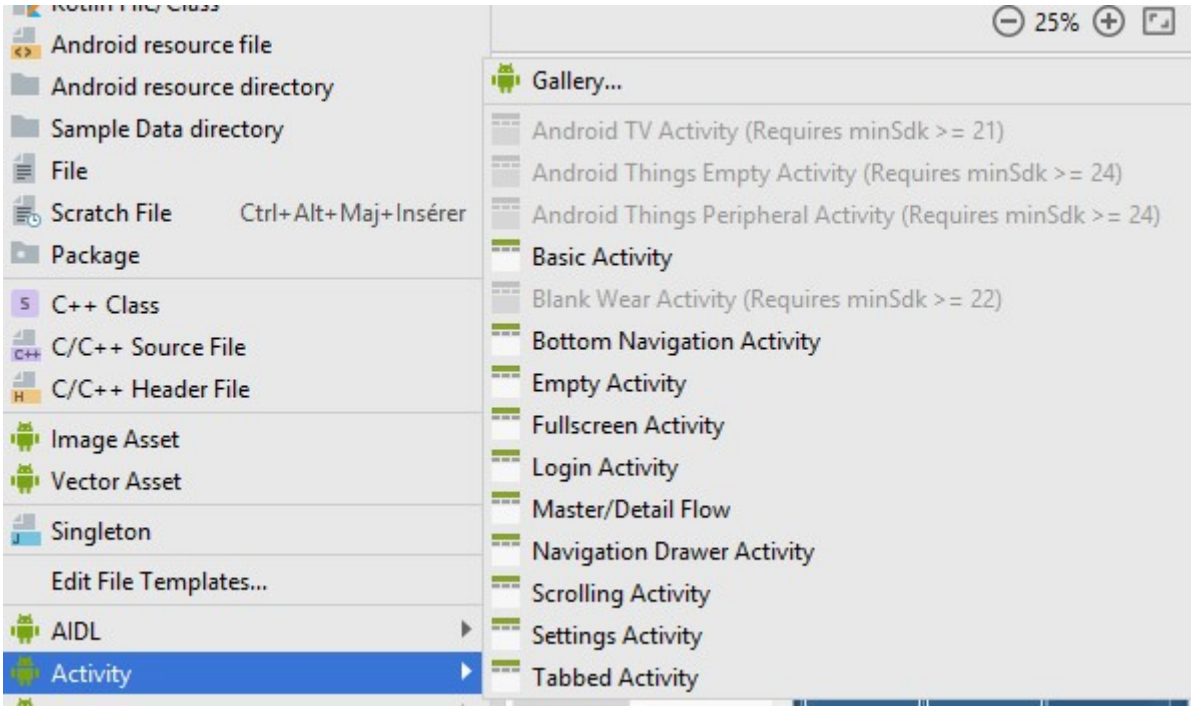

Fig 3.28 : types de modèles s'affiche.

3.4.1.5. Présentation du manifested'application

Chaque projet d'application doit avoir un fichier Android Manifest.xml (avec précisément ce nom) à la racine du jeu de sources du projet. Le fichier manifeste décrit des informations essentielles sur votre application pour les outils de construction Android, le système d'exploitation Android et Google Play.Parmi beaucoup d'autres choses, le fichier manifeste est nécessaire pour déclarer ce qui suit :

Le nom du package de l'application, qui correspond généralement à l'espace de nom de votre code. Les outils de construction Android l'utilisent pour déterminer l'emplacement des entités de code lors de la construction de votre projet. Lors du conditionnement de l'application, les outils de construction remplacent cette valeur par l'ID de l'application à partir des fichiers de construction Gradle, qui est utilisé comme identifiant d'application unique sur le système et sur Google Play [31].

Les composants de l'application, qui incluent toutes les activités, services, récepteurs de diffusion et fournisseurs de contenu. Chaque composant doit définir des propriétés de base telles que le nom de sa classe Kotlin ou Java. Il peut également déclarer des fonctionnalités telles que les configurations de périphérique qu'il peut gérer et les filtres d'intention qui décrivent comment le composant peut être démarré [31].

Les autorisations dont l'application a besoin pour accéder aux parties protégées du système ou d'autres applications. Il déclare également toutes les autorisations que d'autres applications doivent avoir s'ils veulent accéder au contenu de cette application.Les fonctionnalités matérielles et logicielles requises par l'application, ce qui affecte les appareils pouvant installer l'application depuis Google Play [31].

narre.xml | AndroidManifest.xml | activity_ipaddress.xml | Demarre.java | MainActivity.java | activity_main2.xml | Main2Activity.java

manifest

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.hpcore7.smarthome">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="@string/app_name"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />
17             </intent-filter>
18         </activity>
19         <activity android:name=".IPAddress" />
20         <activity android:name=".Demarre" />
21         <activity android:name=".Enterpassword" />
22         <activity android:name=".Createpassword" />
23         <activity android:name=".Main2Activity"></activity>
24     </application>
25
26 </manifest>
```

Text | Merged Manifest

Gradle | Device File Explorer | Event Log | Gradle Console

Fig 3.29 : le fichier manifeste.

3.4.1.6. Présentation du page activité d'application

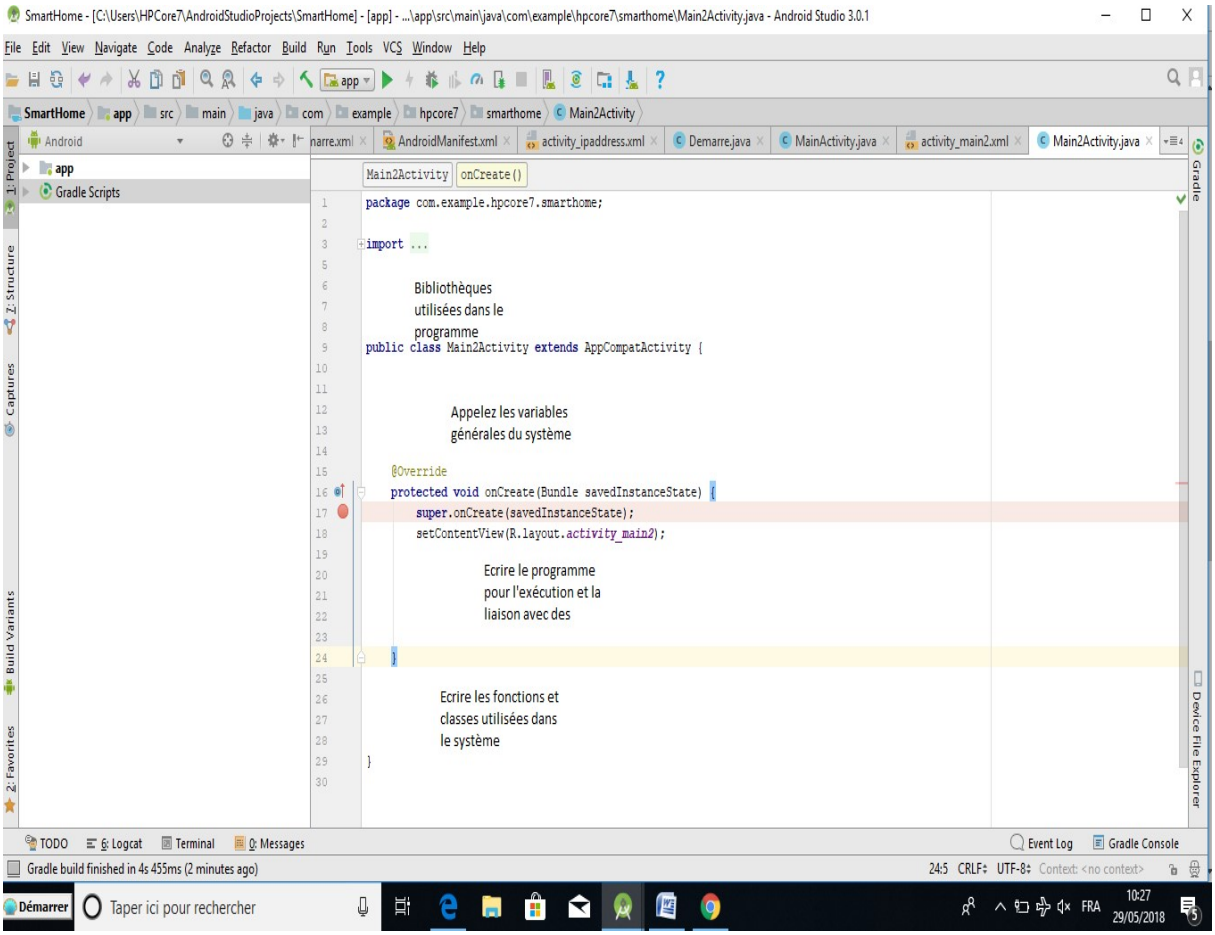

Fig 3.30 :page activité.

3.4.1.7. Première exécution

A. Exécution de l'application

Le SDK Android permet de :

• Installer l'application sur une vraie tablette connectée par USB

• Simuler l'application sur une tablette virtuelle AVD (Android Virtual Device): C'est une machine virtuelle comme celles de VirtualBox et VMware, mais basée sur QEMU. QEMU est en licence GPL, il permet d'émuler toutes sortes de CPU dont des ARM7, ceux qui font tourner la plupart des tablettes Android [32].

B. Assistant de création d'une tablette virtuelle

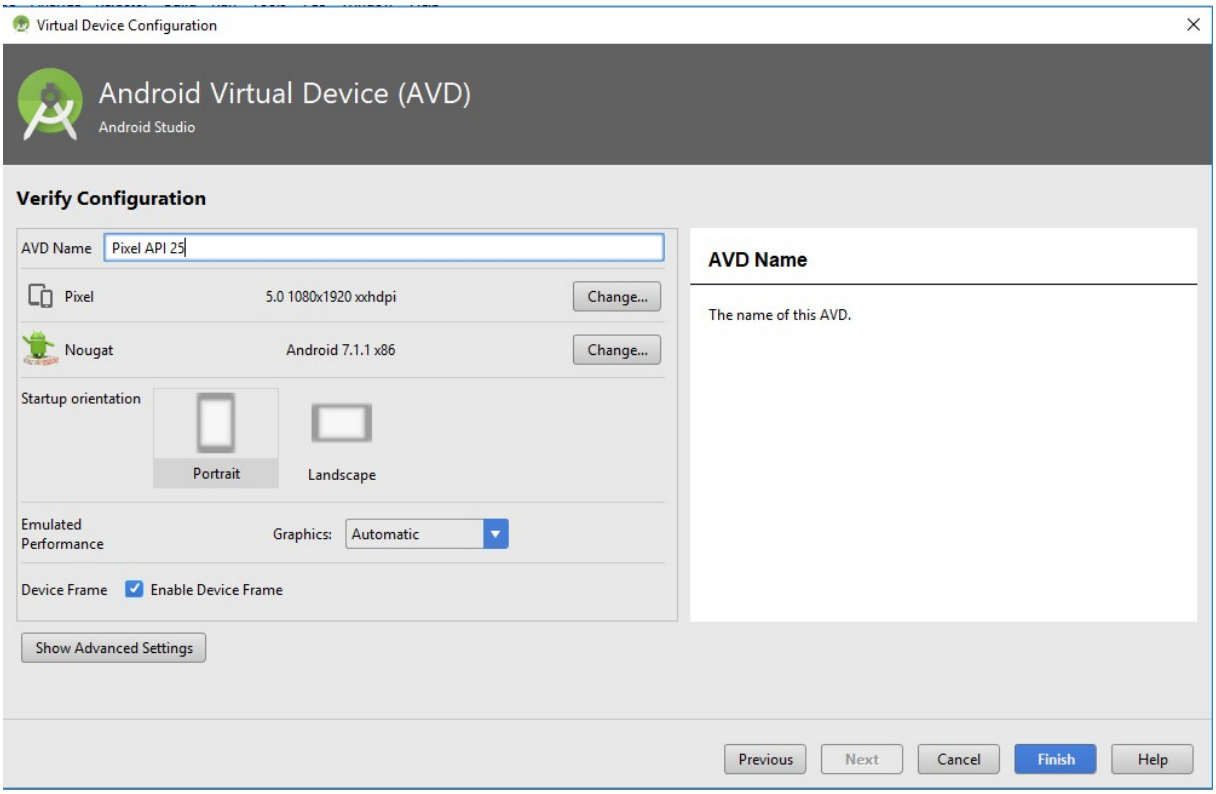

Fig 3.31 : Création d'un AVD

C. Caractéristiques d'un AVD

* L'assistant de création de tablette demande :

• Modèle de tablette ou téléphone à simuler,

• Version du système qu'il doit contenir.

• Orientation et densité de l'écran.

• Options de simulation :

– Snapshot : mémorise l'état de la machine d'un lancement à l'autre, mais exclut Use Host GPU.

– Use Host GPU : accélère les dessins 2D et 3D à l'aide de la carte graphique du PC.

• Options avancées :

– RAM : mémoire à allouer, mais est limitée par votre PC,

– Internalstorage : capacité de la flash interne.

– SD Card : capacité de la carte SD simulée supplémentaire (optionnelle) [32].

D. Lancementd'une application

Bouton vert pour exécuter, bleu pour déboguer :

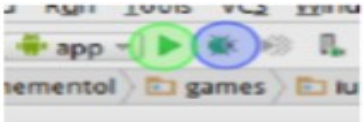

Figure 3.32: Barre d'outils pour lancer une application [32].

Sélectionnez l'un des périphériques de la fenêtre illustrée à la Figure 3.33 pour installer l'application

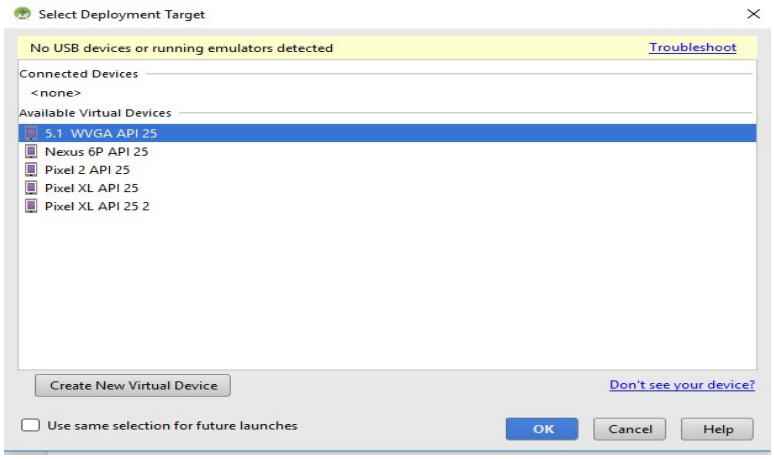

Fig 3.33 : Liste des appareils disponibles.

3.4.1.8. Les objets les plus importants de programmation

A. Buton:

Pour afficher un bouton dans une activité, ajoutez un bouton au fichier XML de disposition de l'activité:

```
46    <Button
47        android:id="@+id/button5"
48        android:layout_width="match_parent"
49        android:layout_height="wrap_content"
50        android:onClick="benkhelfa1"
51        android:text="IPADDRESS" />
```

Pour spécifier une action lorsque le bouton est enfoncé, définissez un écouteur de clic sur l'objet bouton dans le code d'activité correspondant:

```
26      public void benkhelfal(View view)
27      {
28          String text2 = ip.getText().toString();
29          String text3 = portt.getText().toString();
30
31          if(text2.equals("") || text3.equals("") )
32          {
33              Toast.makeText( context: IPAddress.this,  text: "
```

B. TextView:

Un élément d'interface utilisateur qui affiche du texte à l'utilisateur. L'exemple de code suivant montre une utilisation typique, avec une mise en page XML et du code pour modifier le contenu de la vue de texte:

```
<TextView
    android:id="@+id/vontilotomp"
    android:layout_width="match_parent"
    android:layout_height="30dp"
    android:layout_weight="1"
    android:hint="vontilo" />
```

Cet exemple de code montre comment modifier le contenu de la vue de texte définie dans la mise en page XML précédente [32]:

```
txtshow = (TextView)findViewById(R.id.gaz);
txtshow.setText(massige);
```

C. EditText:

Un élément d'interface utilisateur pour entrer et modifier du texte. Lorsque vous définissez un widget de texte d'édition, vous devez spécifier l'attribut TextView_inputType. Par exemple, pour une entrée de texte en clair, définissez inputType sur "text":

```
<EditText
    android:id="@+id/portt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="number" />
```

Le choix du type d'entrée configure le type de clavier affiché, les caractères acceptables et l'apparence du texte d'édition. Par exemple, si vous souhaitez accepter un numéro secret, comme une broche ou un numéro de série unique, vous pouvez définir inputType sur "numericPassword". Un type d'entrée de "numericPassword" génère un texte d'édition qui

n'accepte que les chiffres, affiche un clavier numérique lorsqu'il est mis au point et masque le texte entré pour la confidentialité [32].

3.4.1.9. Explication de l'application

L'organisation de mon application est montrée sur l'organigramme de la figure 3.34.

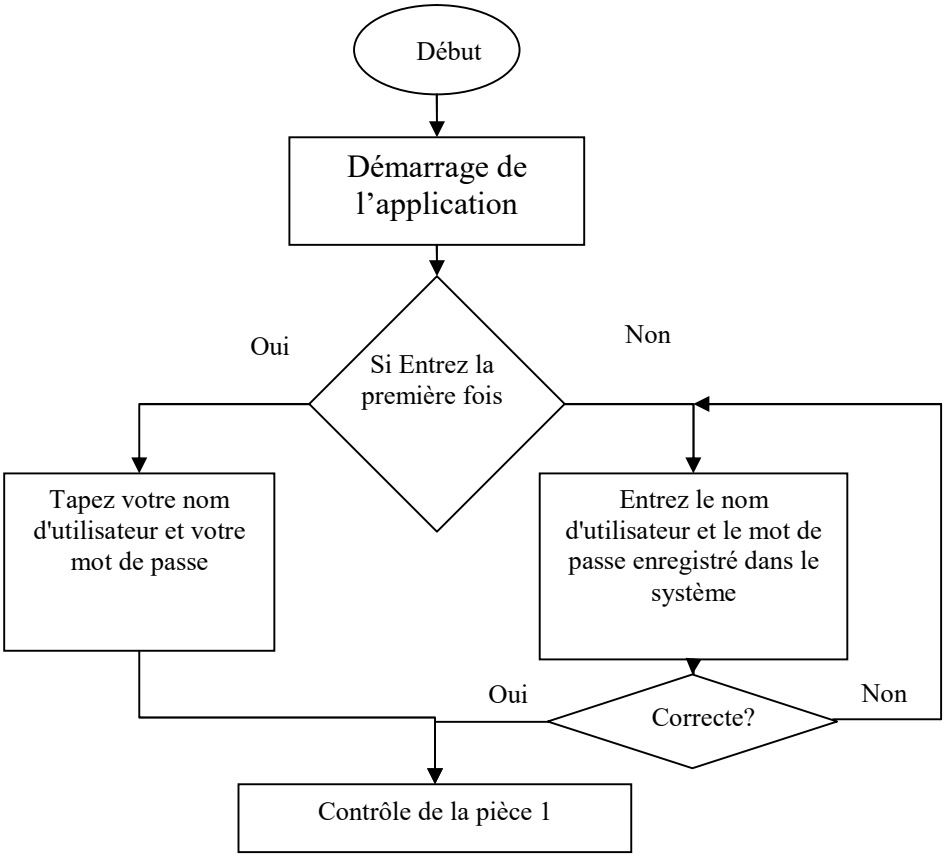

Fig 3.34 : L'organigramme de l'application.

Lorsque l'application démarre, elle vérifie s'il existe déjà des données enregistrées ou si l'application s'ouvre la première fois, comme indiqué dans le code suivant :

A. Ouvre l'aplication

```java
package com.example.hpcore7.smarthome;

import ...

public class MainActivity extends AppCompatActivity {

    String password;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        SharedPreferences settings = getSharedPreferences("PREFS" , 0);
        password = settings.getString( key: "password" , defValue: "");

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                if (password.equals("")) {
                    Intent intent = new Intent(getApplicationContext(), Createpassword.class);
                    startActivity(intent);
                    finish();
                } else {
                    Intent intent = new Intent(getApplicationContext(), Enterpassword.class);
                    startActivity(intent);
                    finish();
                }
            }
        }, delayMillis: 2000);
    }
```

Fig 3.35 : code pageOuvre l'aplication.

Ci-après l'explication du code précédent.

- Cette partie du code est utilisé pour lire le mot de passe stocké dans l'application.

```java
SharedPreferences settings = getSharedPreferences("PREFS" , 0);
password = settings.getString( key: "password" , defValue: "");
```

- Ce code teste si le mot de passe existe dans l'application ou non, S'il n'existe pas, il sera dirigé vers la classe Createpassword pour créer un mot de passe. Si non, Il appelle la classe Enterpassword pour entrer mot de passe.

```
if (password.equals("")) {
    Intent intent = new Intent(getApplicationContext(), Createpassword.class);
    startActivity(intent);
    finish();
}else {
    Intent intent = new Intent(getApplicationContext(), Enterpassword.class);
    startActivity(intent);
    finish();
}
```

B. Saisie de nom d'utilisateur et de mot de pass

Si c'est la première fois que l'application est invoquée, l'utilisateur entre le nom d'utilisateur et le mot de passe et appui sur le bouton « confirm » pour l'enregistrer comme le montre la figure 2. Le code réalisant cette fonction est illustré dans la figure 3.36.

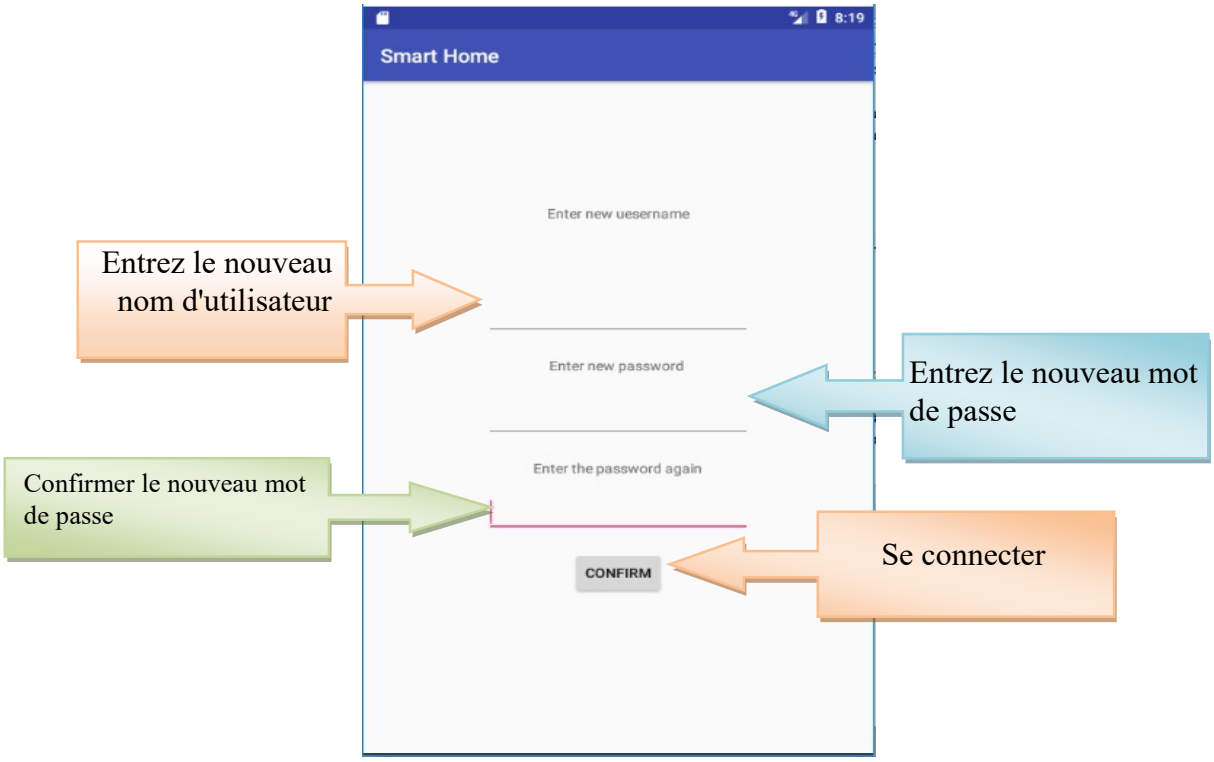

Fig 3.36 : entre le nom d'utilisateur et le mot de passé.

```java
package com.example.hpcore7.smarthome;
import ...
public class Createpassword extends AppCompatActivity {
 EditText editText1 ,editText2 ,editText3 ;
 Button button ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_createpassword);
    editText1 = (EditText) findViewById(R.id.editText1);
    editText2 = (EditText) findViewById(R.id.editText2);
    editText3 = (EditText) findViewById(R.id.editText3);

    }

    @RequiresApi(api = Build.VERSION_CODES.GINGERBREAD)
    public void buttonD(View view)
    {
        String text1 = editText1.getText().toString();
        String text2 = editText2.getText().toString();
        String text3 = editText3.getText().toString();

        if(text2.equals("") || text3.equals("") || text1.equals(""))
        {
            Toast.makeText( context: Createpassword.this, text: "No password entered!" ,Toast.LENGTH_SHORT).show();
        } else {
            if(text2.equals(text3)) {
                SharedPreferences settings = getSharedPreferences("PREFS" , 0);
                SharedPreferences.Editor editor = settings.edit();
                SharedPreferences.Editor editor1 = settings.edit();
                editor.putString("password" , text2);
                editor.apply();
                editor1.putString("uesrname" , text1);
                editor1.apply();

                Intent intent = new Intent(getApplicationContext(), IPAddress.class);
                startActivity(intent);
                finish();
            } else {
                Toast.makeText( context: Createpassword.this, text: " password doesn't match!" , Toast.LENGTH_SHORT).show();
                 }
        }
    }
}
```

Fig 3.37 : entre le nom d'utilisateur et le mot de passé.

Le code de la figure 3.36 est expliqué en détail en ce qui suit.

- Ce code permet de faire la liaison entre l'objet et le code java pour l'utiliser.

```java
editText1 = (EditText) findViewById(R.id.editText1);
editText2 = (EditText) findViewById(R.id.editText2);
editText3 = (EditText) findViewById(R.id.editText3);
```

- Ce code lit les textes dans les objets EditText.

```java
String text1 = editText1.getText().toString();
String text2 = editText2.getText().toString();
String text3 = editText3.getText().toString();
```

- Ce code s'assure que le mot de passe est valide puis le stocke dans l'application.

```
else {
if(text2.equals(text3)) {
SharedPreferences settings = getSharedPreferences("PREFS" ,0);
SharedPreferences.Editor editor = settings.edit();
SharedPreferences.Editor editor1 = settings.edit();
        editor.putString("password" , text2);
        editor.apply();
        editor1.putString("uesrname" , text1);
        editor1.apply();
```

Cette page s'ouvre lorsque vous redémarrez l'application et que le nom d'utilisateur et le mot de passe sont saisis pour permettre l'accès à l'application.

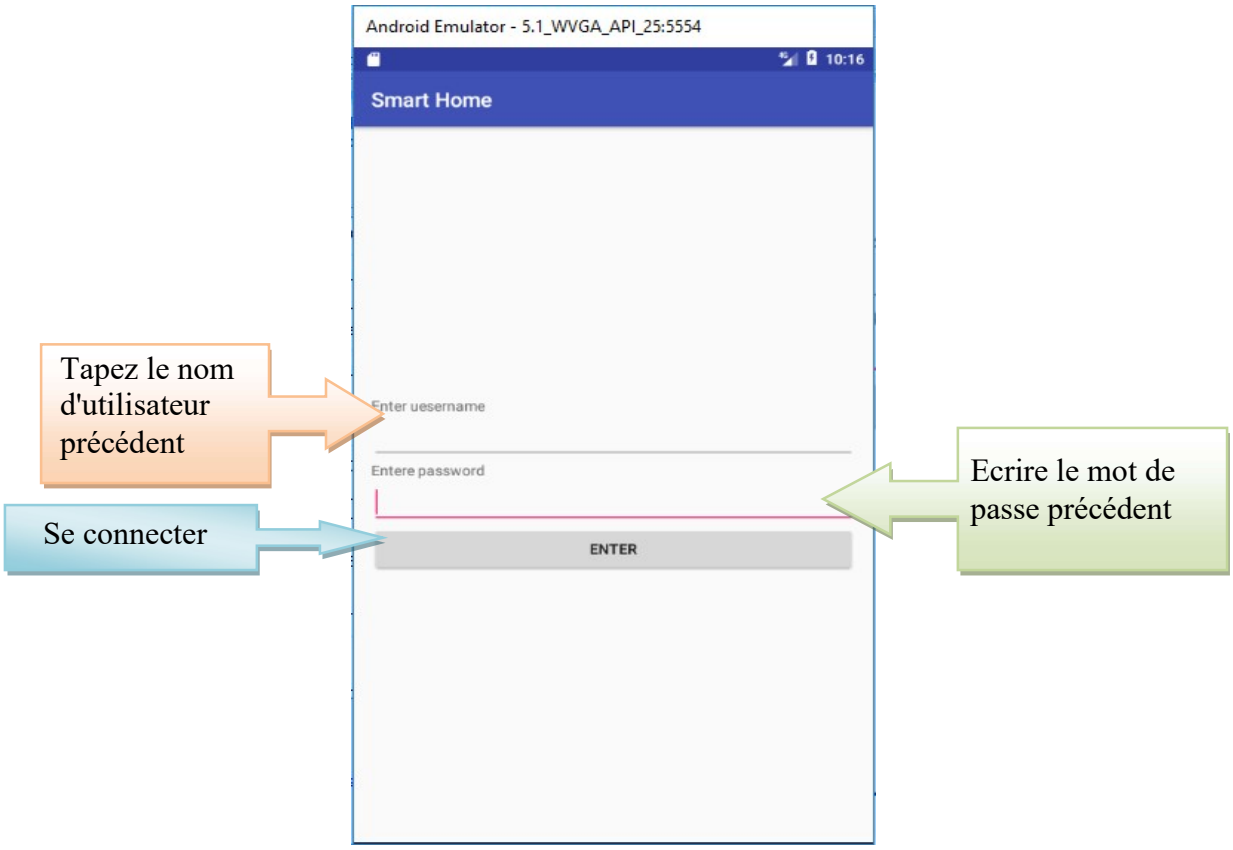

Fig 3.38 : Entrez la page du mot de passé.

Ce code compare le mot de passe et le nom d'utilisateur dans l'applicationAvec le mot de passe et le nom d'utilisateur entré par l'utilisateur.

```
public class Enterpassword extends AppCompatActivity {
    EditText editTextu, editTextp;
    String password;
    String uesrname;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_enterpassword);

SharedPreferences settings = getSharedPreferences("PREFS", 0);
password = settings.getString( key: "password",  defValue: "");
uesrname = settings.getString( key: "uesrname",  defValue: "");

        editTextu = (EditText) findViewById(R.id.editTextu);
        editTextp = (EditText) findViewById(R.id.editTextp);
    }
    public void buttonE(View view ) {

        String text2 = editTextp.getText().toString();
        String text3 = editTextu.getText().toString();

        if (password.equals(text2) && uesrname.equals(text3)) {
            Intent intent = new Intent(getApplicationContext(), IPAddress.class);
            startActivity(intent);
            finish();
        } else {
            Toast.makeText( context: Enterpassword.this,  text: " Wrong password! ", Toast.LENGTH_SHORT).show();
             }
    }
}
```

Fig 3.39 : Mot de passe du programme.

C. Menu principale de l'application

Cet interface permet à l'utilisateur soit d'envoyer des ordres vers la maison : Gaz ON, Protection ON, …etc, soit de recevoir des information des capteurs : Humidité, Température, …etc

Fig 3.40 : Menu principale de l'application.

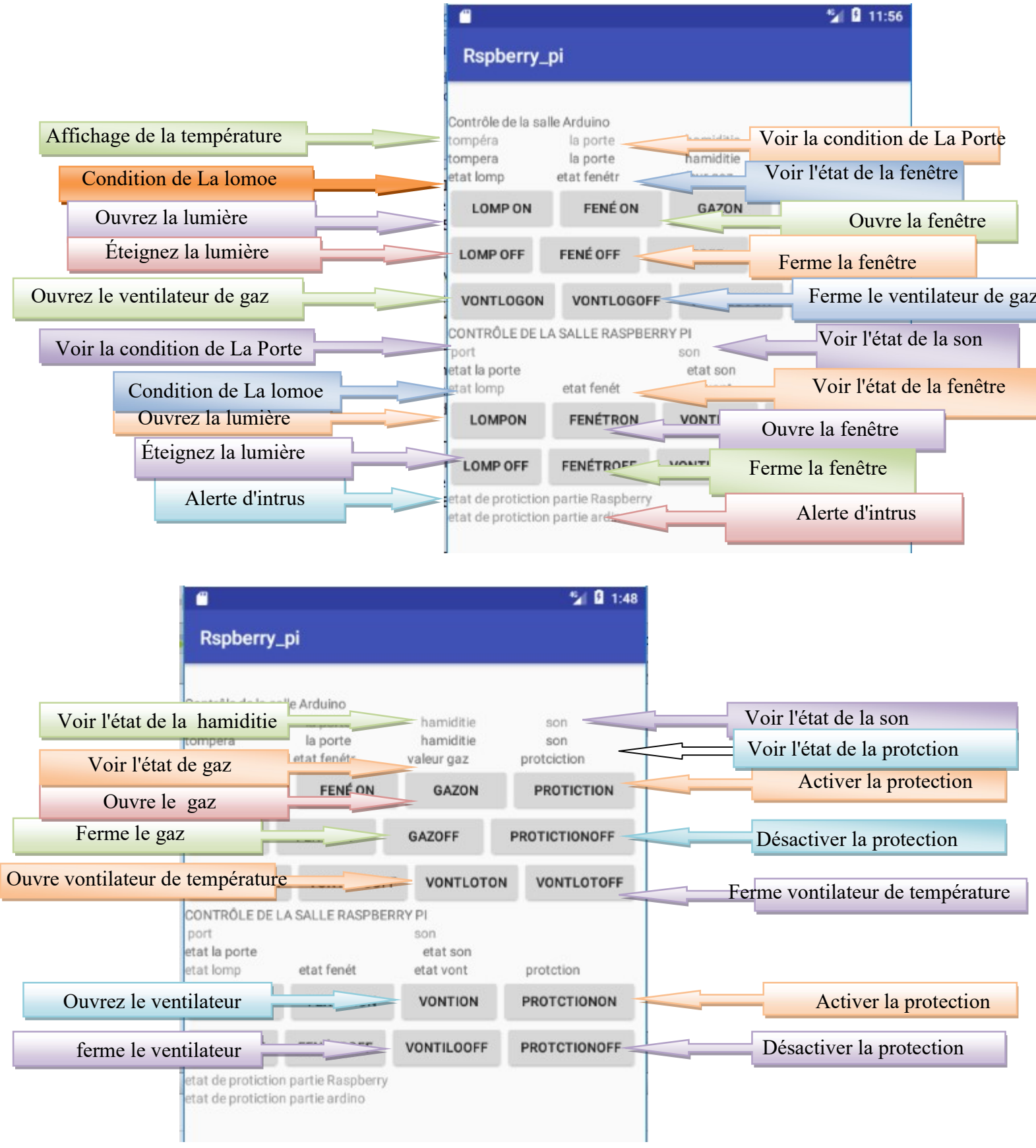

Fig 3.41 : Expliquer menu principale de l'application.

Connextion et communication avec au Raspberry pi 3 comme server

Dans ce code Android, l'application envoie des commandes vers Raspberry pi 3.

```
classmyTask extends AsyncTask<Void ,Void ,Void > {
///// Définition de la classe
@Override
protected Void doInBackground(Void... params) {
//// Exécution de la classe


try {
s = new Socket(ip, port);
/////Ici, nous avons créé un objet de la classe "socket" pour se connecter à
/////un autre programme sur le serveur et utiliser le port 2004
pp = new PrintWriter(s.getOutputStream());
/////L'objet "pp" que nous utiliserons pour écrire dans l'objet "s"
br= new BufferedReader(isr);
/////L'objet "br" sera utilisé pour lire les données que l'utilisateur va
/////entrer dans l'application

pp.write(message);
pp.flush();
/////Lisez le message et envoyez

pp.close();
br.close();   /////Fermer les objets
s.close();


        } catch (Exception e) {
e.printStackTrace();
        }
```

Pour envoyer des informations à la maison, nous utilisons un bouton qui à son tour appelle le client pour envoyer l'information. Le code sera comme suit:

```
public void lompon(View view) {
massige ="cc";

    myTask mt = new myTask();

    mt.execute();
}
```

L'application Adroid comme server

L'application Android reçoit les informations de la maison via Rasperry (Client) comme le montre le programme suivant :

```
class recervimplementsRunnable {        ////Définition de la classe

privateSocket s;
privateServerSocketss;
privateInputStreamReaderisr;///Définir des objets et des variables
BufferedReaderbrr;
    Handler h = new Handler();

    String MSSM;

@Override
public voidrun(){//////Exécution de la classe
try{
ss= new ServerSocket(2004) ;
/////Ici nous avons créé un objet de la classe ServerSocket pour recevoir
/////touteinformation du client Raspberry.
while(true){
s = ss.accept();
/////Ici, nous appelons la fonction accept (), qui retourne un objet de son
/////type Socket,Il peut échanger des blocs avec n'importe quel autre objet
/////Socket.
isr= new InputStreamReader(s.getInputStream());
/////L'objet "isr" nous allons l'utiliser pour lire le contenu de l'objet "s"
brr= new BufferedReader(isr);
MSSM = brr.readLine();
/////Utilisé pour extraire et stocker le contenu de l'objet B dans la
/////variable en K.
h.post(new Runnable() {

@Override
public void run() {

txtshow.setText(MSSM);

Toast.makeText(getApplicationContext(),"MSSM",Toast.LENGTH_LONG).show();

///// Dans cette section, les informations reçues sont affichées
                    }
                });
            }
}catch(IOException e) {
        }
    }
```

Le code suivant est écrit dans "onCreate " de la page activity_demarre pour exécuter la classe recerv du code précédent.

```
45    protected void onCreate(Bundle savedInstanceState) {
46        super.onCreate(savedInstanceState);
47        setContentView(R.layout.activity_demarre);
48
49        Thread myThread = new Thread(new recerv());
50        myThread.start(); }
```

3.4.2. Programmation python dans Raspberry pi 3 :

3.4.2.1. Introduction à Python:

Python est un langage facile à apprendre et leur code est plus lisible, il est donc plus facile à maintenir. Il est parfois jusqu'à 5 fois plus concis que le langage Java par exemple, ce qui augmente la productivité du développeur et réduit automatiquement le nombre de bugs.
L'introduction la plus facile à Python est à travers IDLE, un environnement de développement Python. Ouvrez IDLE depuis le bureau ou le menu des applications [33]:

Fig 3.42:Ouvrez IDLE python [33].

Avec IDLE maintenant ouvert, nous pouvons commencer à créer du code.

```
print ("Hello World")
```

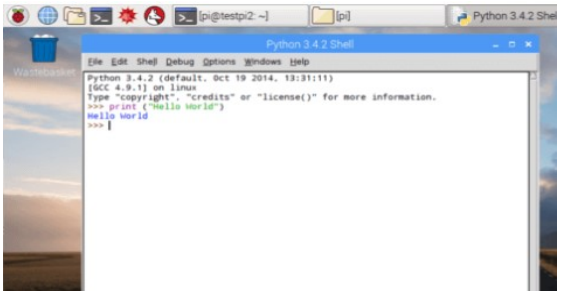

Fig 3.43 : IDLE maintenant python.

Ce que vous voulez est un moyen d'écrire des programmes plus longs et être capable de les exécuter encore et encore. Pour ce faire, tout ce dont nous avons besoin est un simple éditeur de texte, à partir de la fenêtre IDLE sélectionnez "Fichier" dans le menu puis "Nouveau fichier" (ou Ctrl + N), cela ouvrira une nouvelle fenêtre et nous pourrons écrire notre code [33].

Pour exécuter le programme, sélectionnez simplement l'option "Exécuter" dans le menu puis "Exécuter le module". Vos résultats de votre programme seront alors affichés dans la fenêtre IDLE [33].

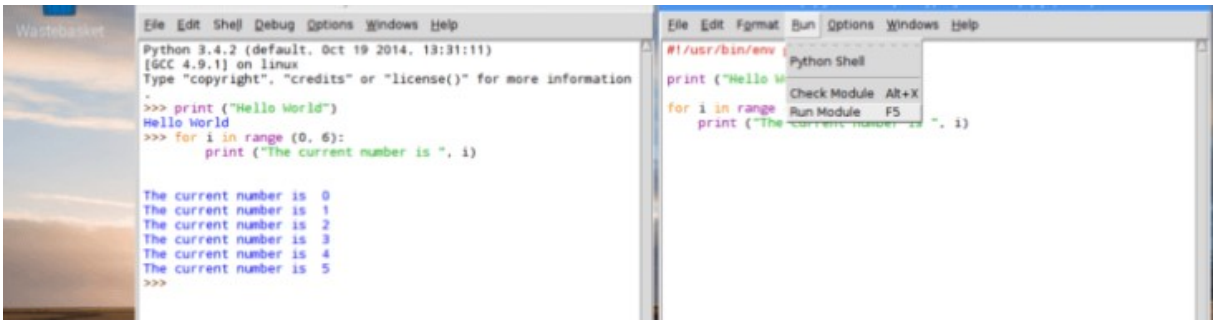

Fig 3.44 : exécuter le programme python [33].

Une autre façon d'exécuter votre code une fois qu'il a été enregistré est directement à partir de l'interface de ligne de commande (CLI) ou de l'écran du terminal.

3.4.2.2. L'application Raspberry pi 3 proposée

Le programme de l'application Raspberry est écrit en langage Python version 3.

L'organigramme de la figure 3.45 montre L'organigramme du programme python dans Raspberry pi 3.

On commence par connexion à internet

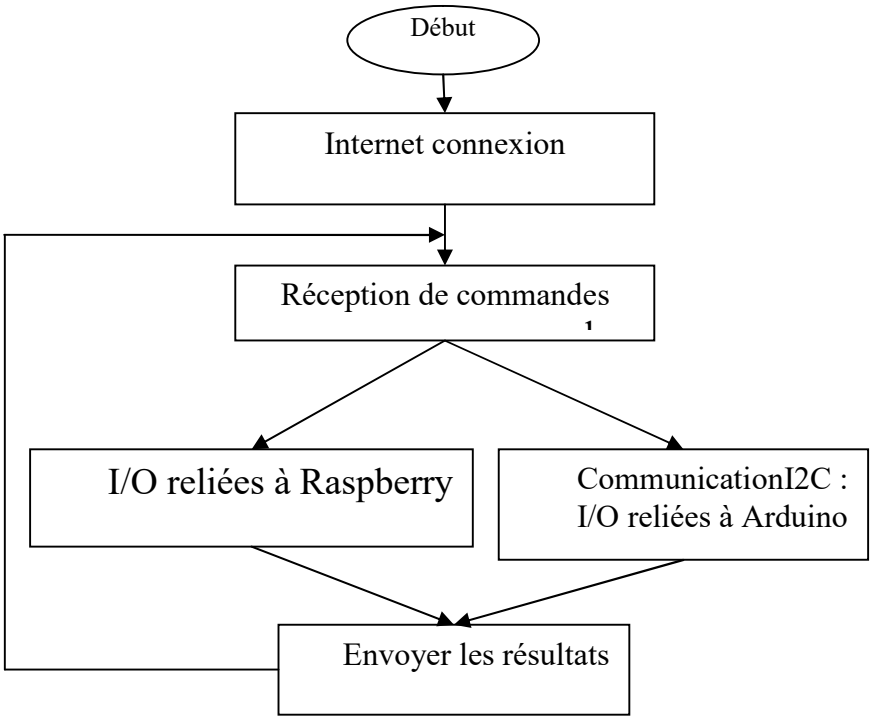

Fig3.45:L'organisation de l'application.

A. Réception de commandes

Le programme de server est une classe qui reçoit des commandes envoyées par l'utilisateur et par la suite envoyées aux autres classes chargées du traitement.

```
import sock
s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
s.bind (('192.168.43.5', 2004))
s.listen (1)
while 1:
c, addr = s.accept()
print "connection from: " + str(addr[0])
data = c.recv(1024)
```

- La première ligne appelle la bibliothèque de socket et cette ligne doit être écrite au début de tout programme qui appelle les fonctions du logiciel réseau
- La deuxième ligne est un plug-in nommé "s" qui dépend du protocole IP / TCP (c'est le nom à gérer avec le plug dans le programme).
- Dans la troisième ligne sera dans cette étape est de déterminer l'adresse du serveur « serveur » que le programme est exécuté sur elle, tout comme le port que le programme recevra la sélection de réseau entrant.Dans ce cas, l'adresse réseau est "192.168.43.5" et le numéro de port est 2004.
- Dans la quatrième ligne, les connexions du réseau sont en attente et le nombre de connexions autorisées à recevoir en même temps peut être déterminé dans notre programme.
- Dans la cinquième ligne, une boucle répétitive infinie a été créée.
- Dans la sixième ligne, la connexion est acceptée et l'adresse IP du client est donnée et un port local est donné au programme (le port est donné sans intervention du programmeur), où la variable "addr" contient l'adresse réseau et le numéro de port du client.
- Dans la septième ligne, "connectionfrom: " est imprimé en plus de l'impression de l'adresse réseau du client.

- Dans la huitième ligne, les données envoyées par le client sont stockées dans la variable "data".

Remarque: Python renvoie un objet "accept" utilisé pour envoyer et recevoir des données.
Send & recv: Utilisé avec l'objet créé pour envoyer et recevoir des données dans notre programme, nous avons seulement reçu les données.

B. CommunicationI2C

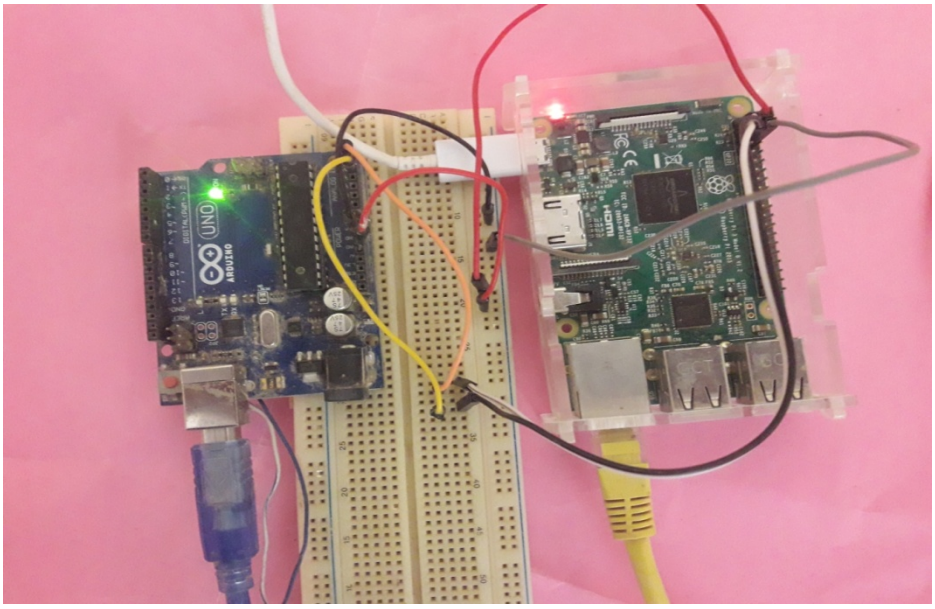

Fig 46 : connction Arduino Raspberry.

Cetteclass est responsable de la communication entre Raspberry Pi et l'Arduino, où elle envoie les ordres à l'Arduino et reçoit les résultats de celui-ci.

```
bus = smbus.SMBus(1)
address = 0x04
defwriteNumber(value):
bus.write_byte(address, value)
return -1
defreadNumber ():
number = bus.read_byte(address)
return number
while True:
va = var
writeNumber(va)
t = readNumber()
```

C. Contrôle Raspberry Pi

➢ Contrôle de la lumière

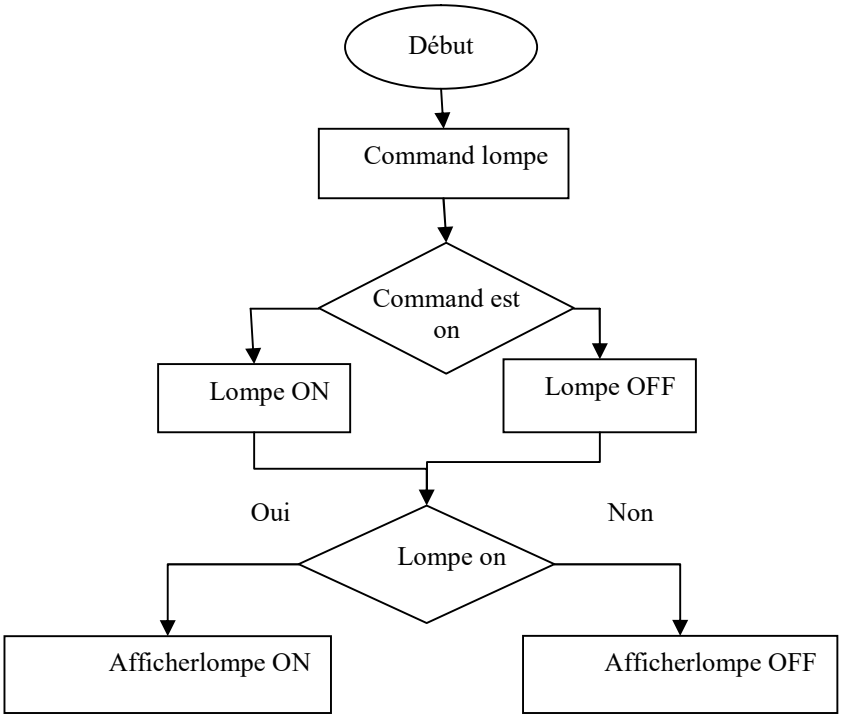

Fig 3.47 : Système de lumière de la maison(Raspberry).

Pour s'assurer que la lampe est réellement allumée, nous avons utilisé le capteur de lumière LDR avec le code suivant:

While 1 :

```
benkhalfa_commandAP(self.vaz)
benkhalfa_ON_OFF_L(1, self.vaz)
t = benkhalfa_ldr()
```

Ces fonctions se trouvent dans l'annexe (Python Program /partie control Raspberry).

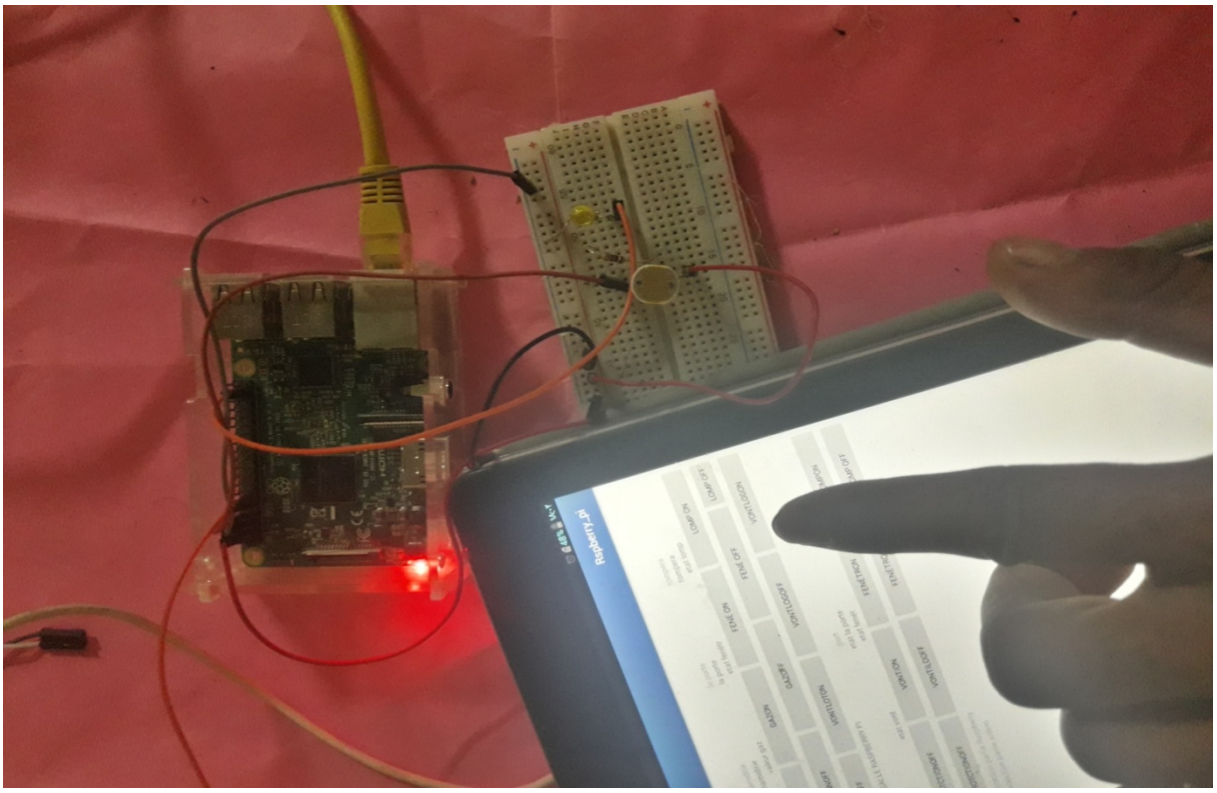

Fig 3.48: Allumez la lumière.

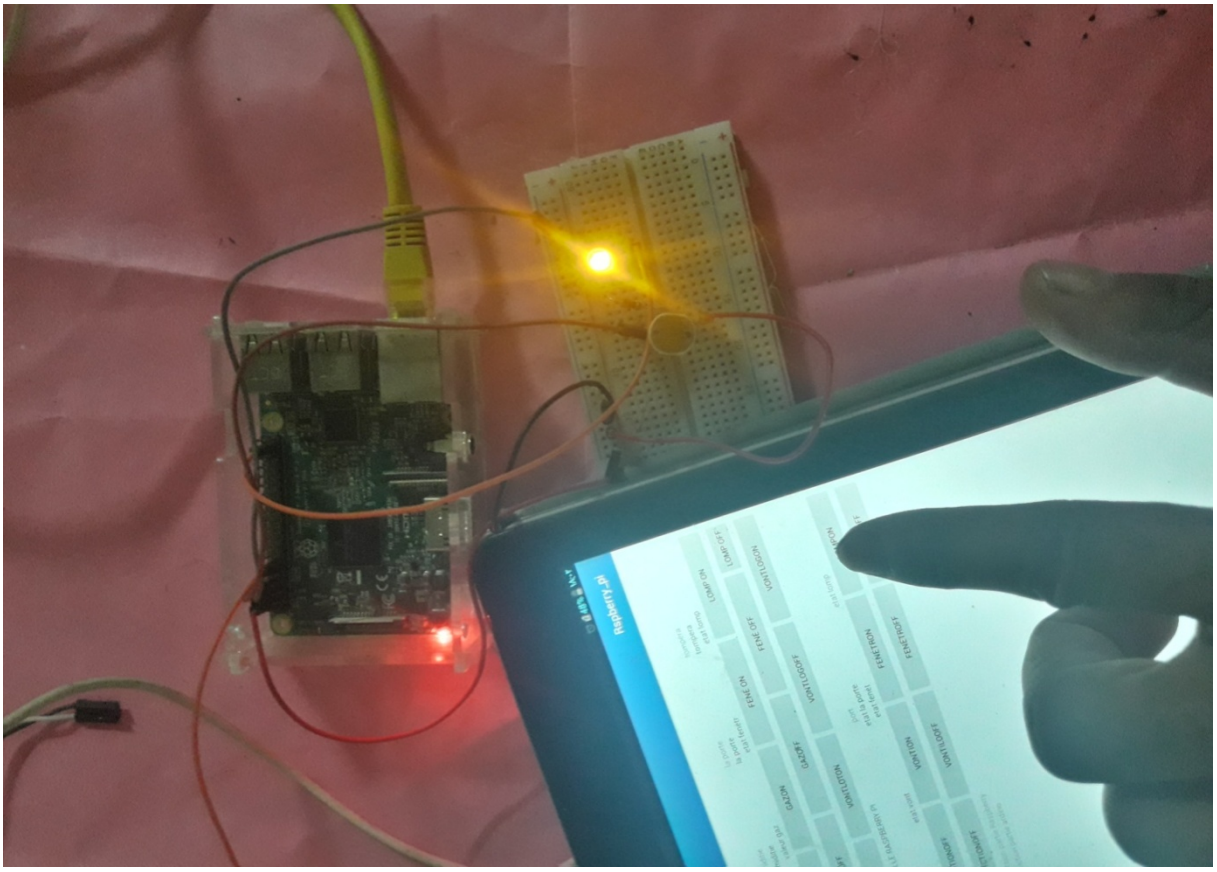

Fig 3.49: Éteins la lumière.

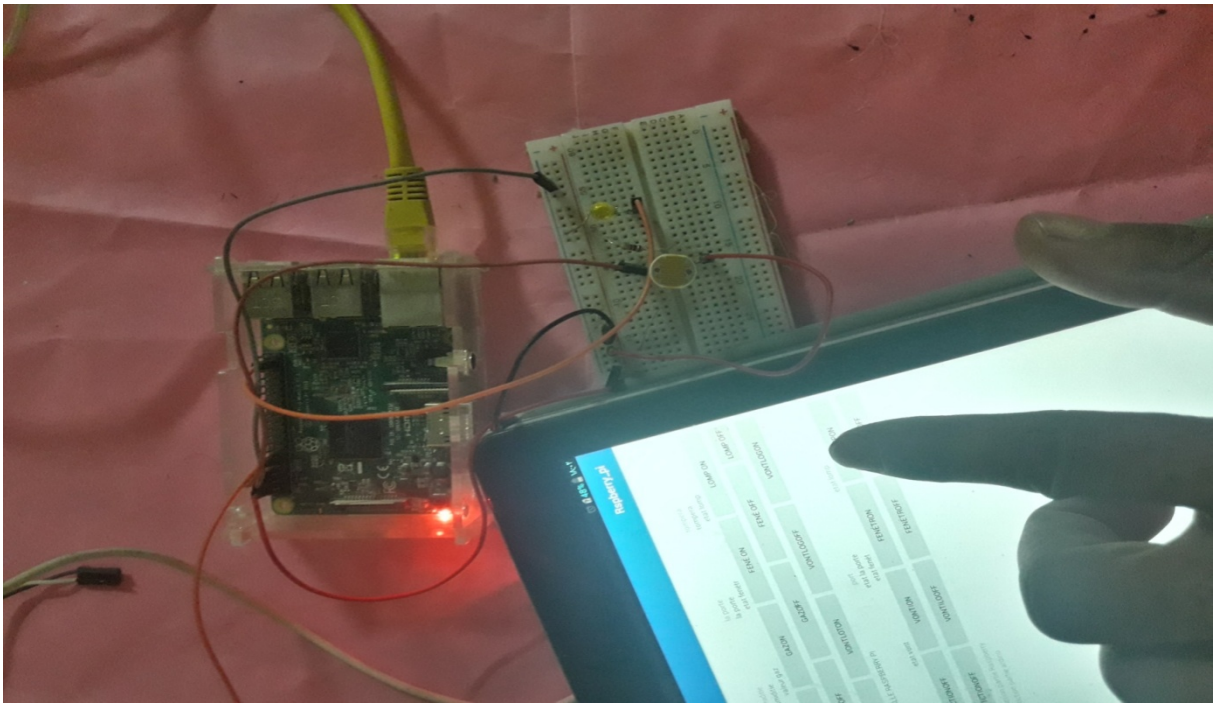

Fig 3.50: Après avoir arrêté l'éclairage.

- **Protection**

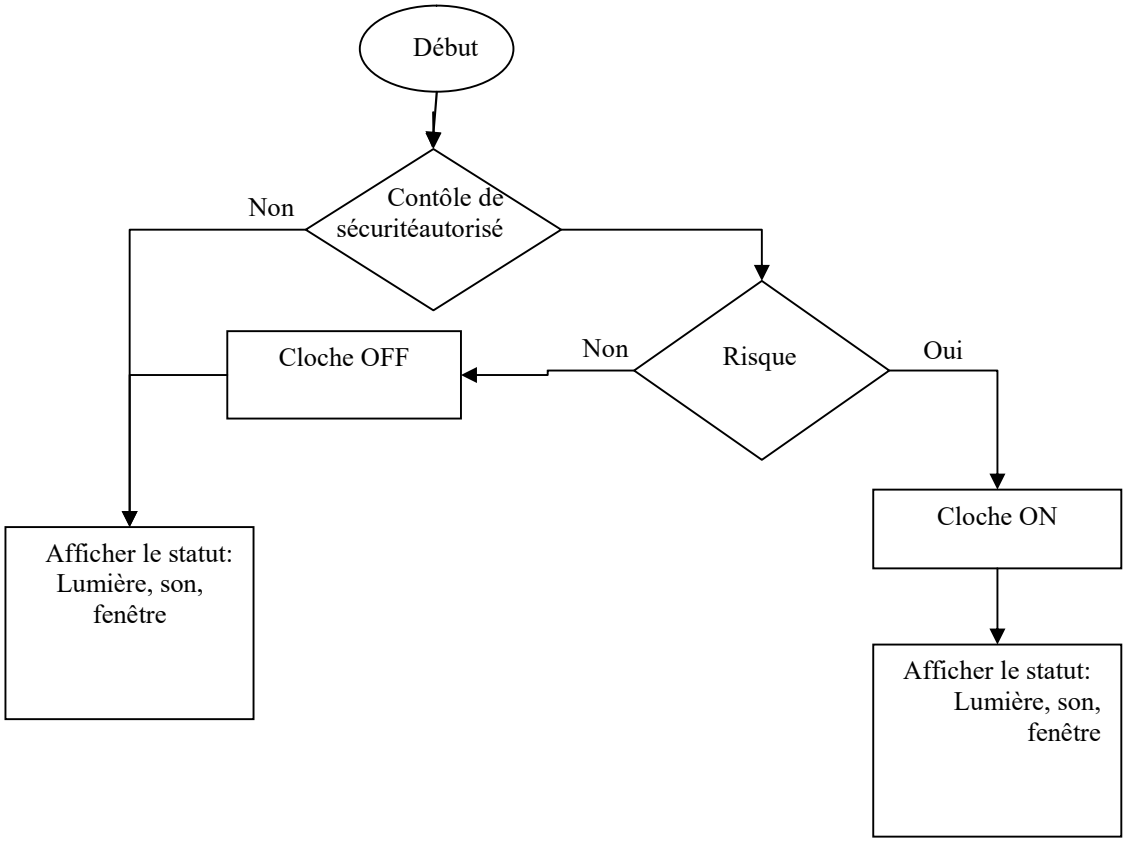

Fig 3.51: Système de protection de la maison(Raspberry).

Le programme est comme suit :

While 1 :

```
ss = son()
pp = porte()
ff = fetr()
protction(pp ,ff , ss)
protction1(var)
```

Ces fonctions se trouvent dans l'annexe (Python Program /partie control Raspberry).

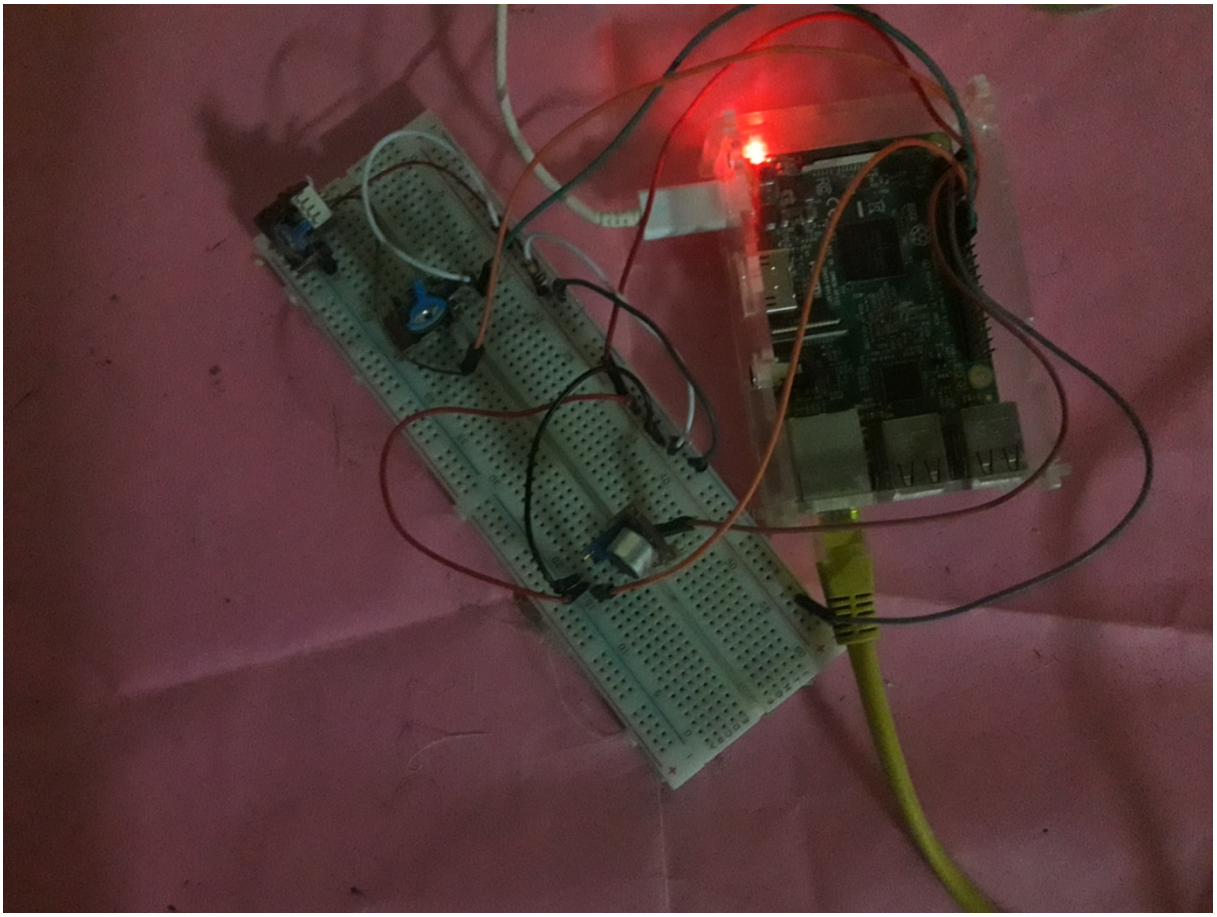

Fig 3.52:Image système de protection de la maison(Raspberry).

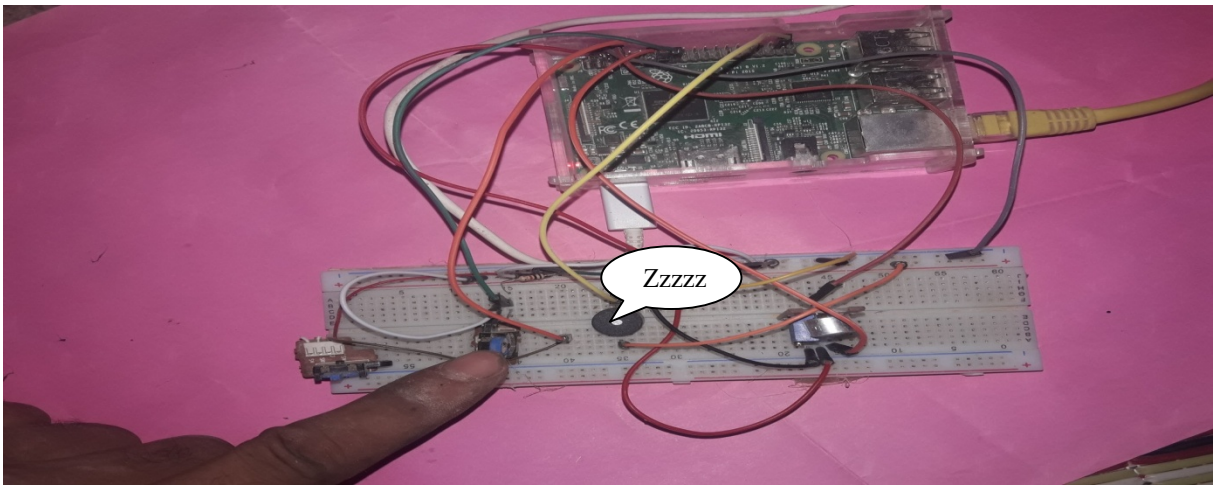

Fig 3.53: Protection de la maison (partie de la fenêtre).

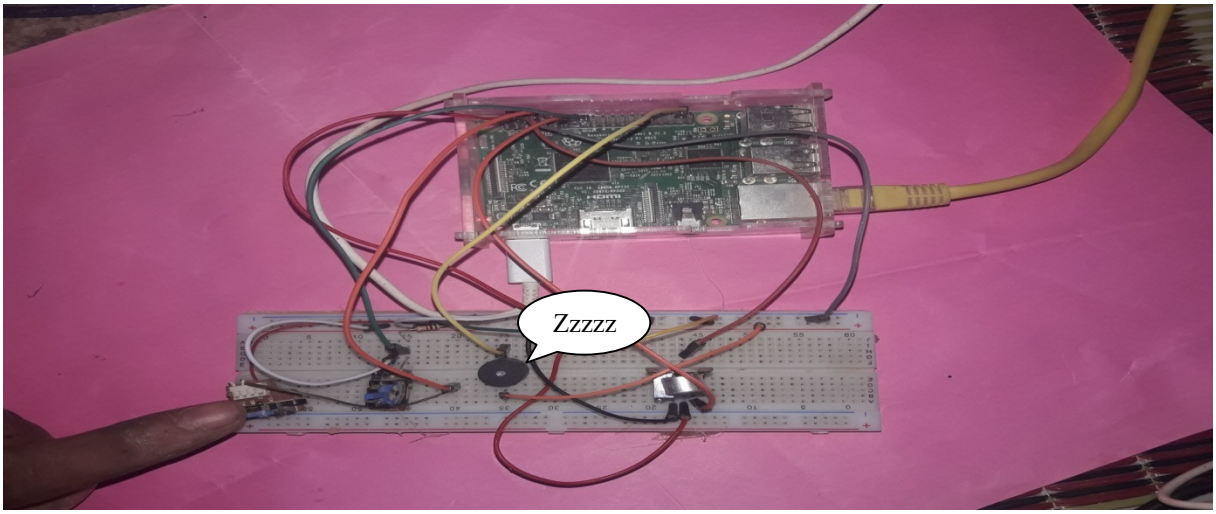

Fig 3.54: Protection de la maison (partie de la porte).

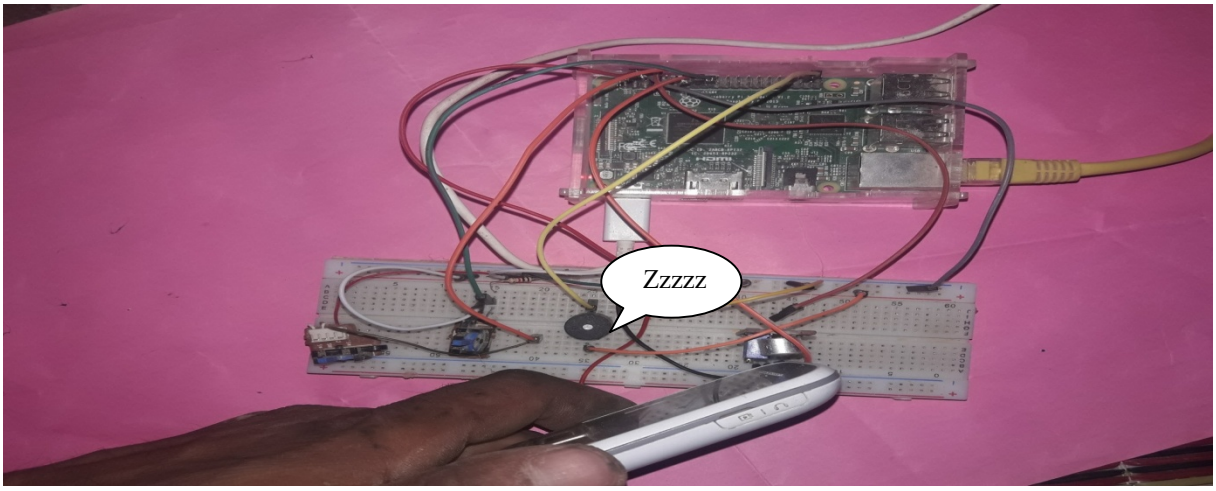

Fig 3.55: Protection de la maison (partie de son).

- **Contrôle de la fenêtre et du ventilateur**

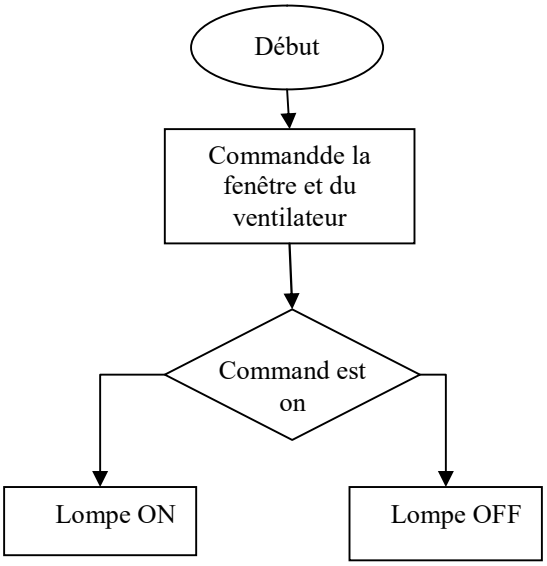

Fig 3.56 : Organigramme Contrôle de la fenêtre et du ventilateur.

Le programme est comme suit :

```
While 1 :
        vontilo(1 , self.vaa)

        if self.ll == 0 :
            setup()
            self.ll = self.ll + 1

        servodown(self.va)
```

Ces fonctions se trouvent dans l'annexe (Python Program /partie control Raspberry).

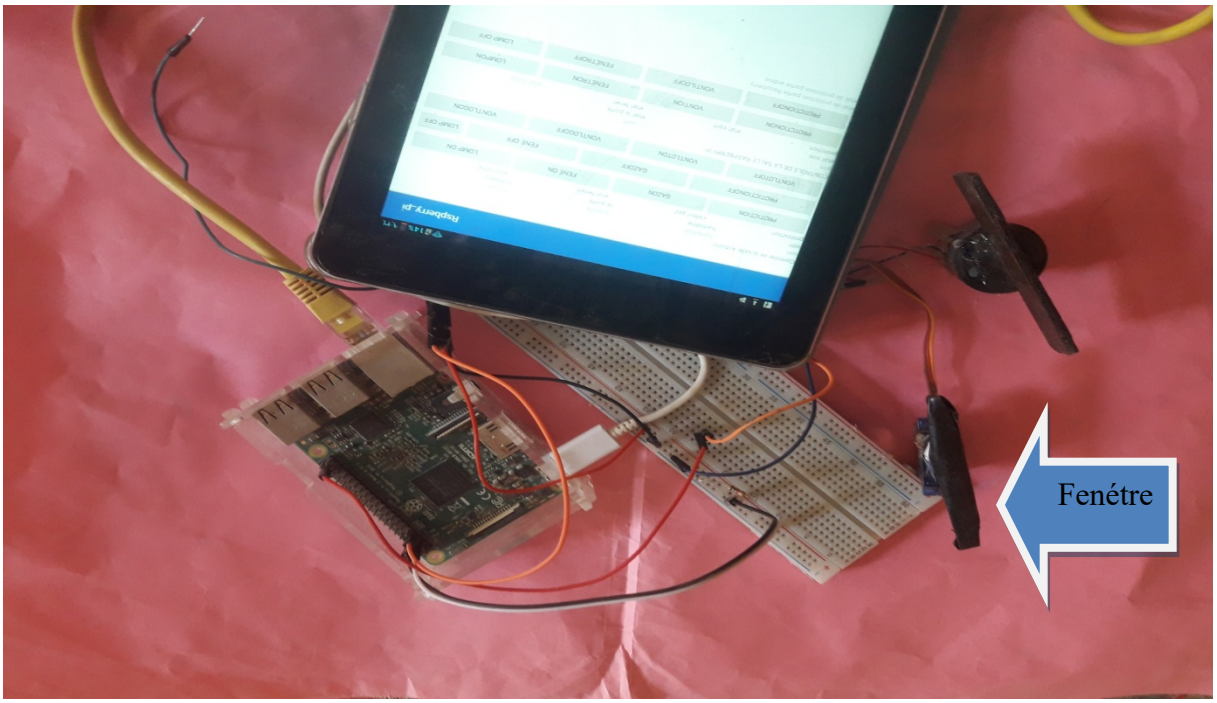

Fig 3.57: Contrôle de la fenêtre et du ventilateur

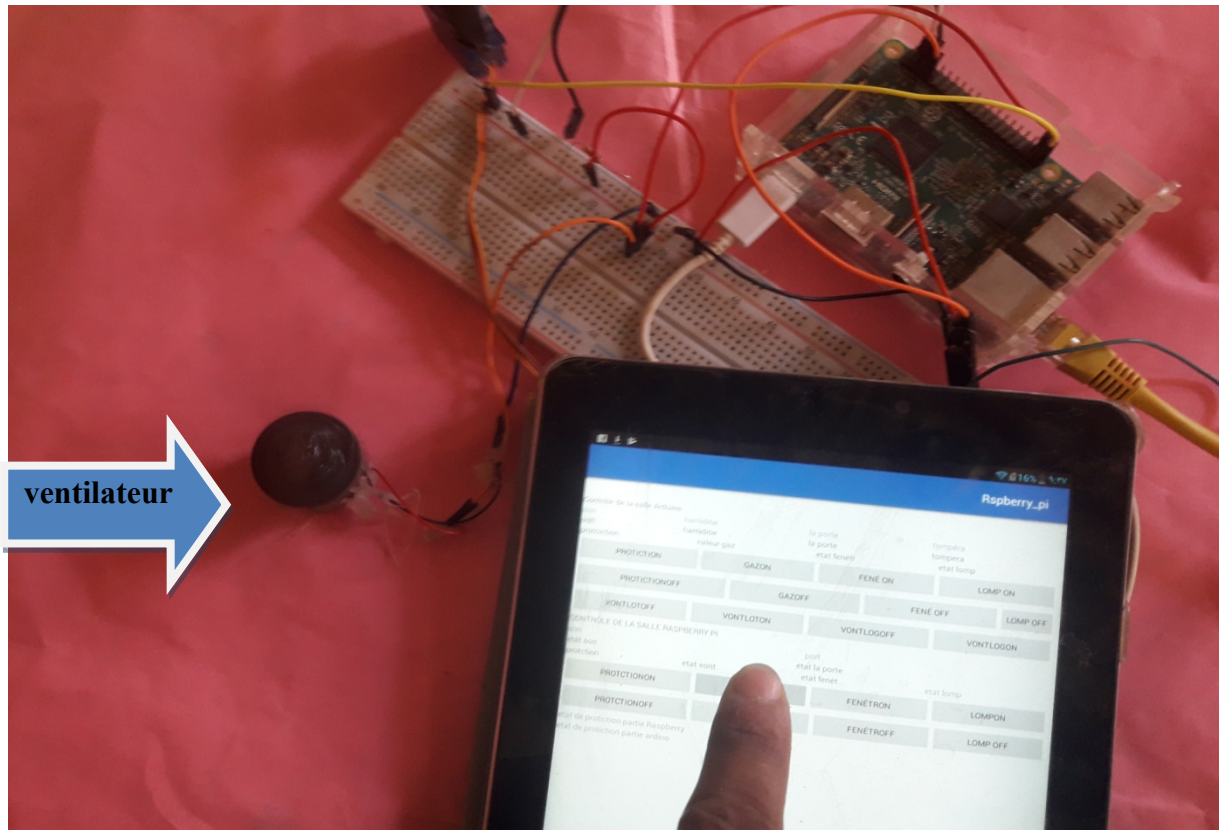

Fig 3.58: Activer le ventilateur.

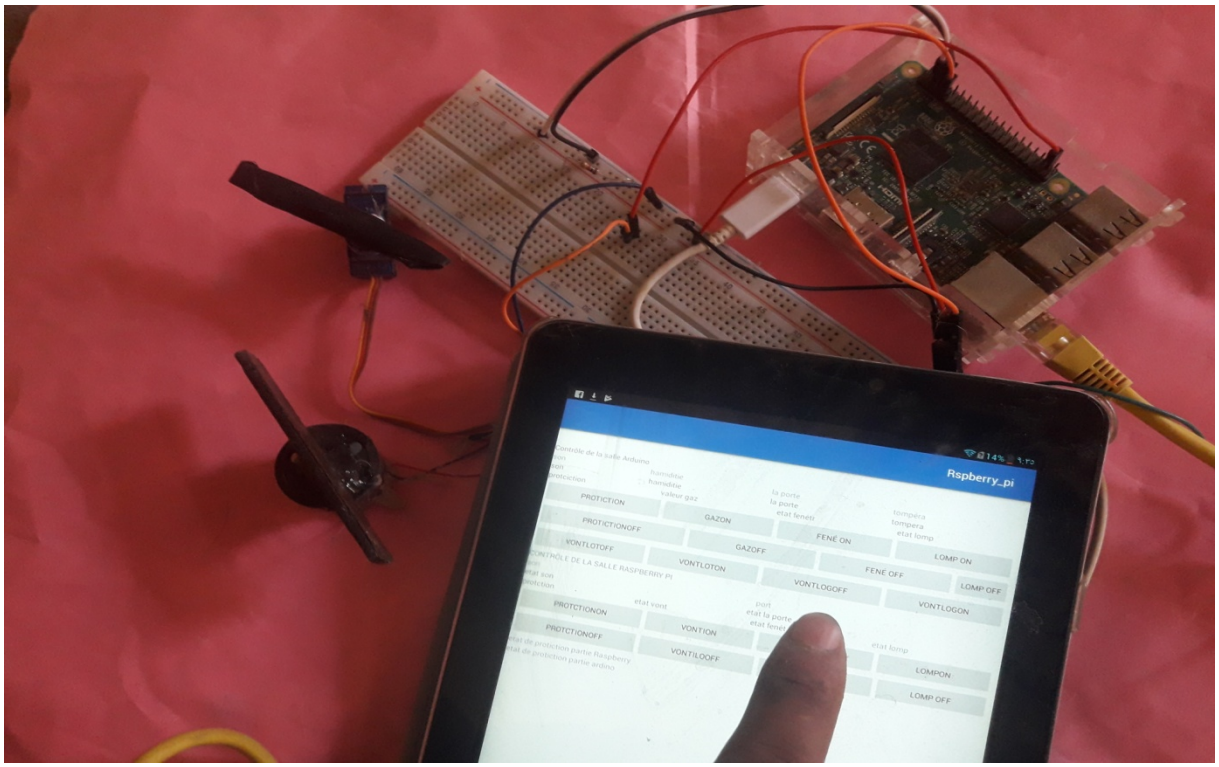

Fig 3.59: Ouvre la fenêtre.

D. Transmission des résultats

Le programme de client est une classe qui envoie des commandes vers l'utilisateur.

```
1  import socket
2  c = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
3  c.connect((self.mot , 2004))
4  c.send(messag)
5  c.close()
```

- La première ligne appelle la bibliothèque de socket et cette ligne doit être écrite au début de tout programme qui appelle les fonctions du logiciel réseau
- La deuxième ligne est un plug-in nommé "c" qui dépend du protocole IP / TCP (c'est le nom à gérer avec le plug dans le programme).
- La troisième ligne se connecte au serveur via l'adresse réseau et le numéro de port,Où l'adresse réseau est chargée par la variable "self.mot" et le numéro de port est 2004.
- La quatrième ligne envoie les données de la variable "messag" au serveur.
- La cinquième ligne termine la connexion avec le serveur.

3.4.3.3. Programmation Arauino

Dans ce travail, nous avons utilisé l'Arduino pour contrôler une partie de la maison comme il est montré dans la figure suivante sous forme d'un organigramme.

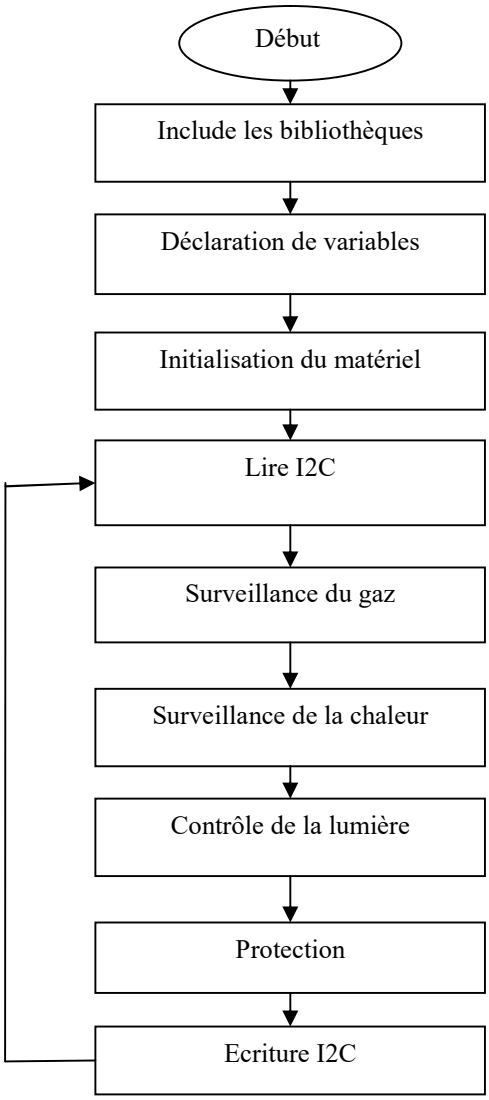

Fig 3.60 : L'organisation de l'application.

A. Communication I2C:

Dans cette section, l'Arduino échange des informations avec Raspberry Pi comme le montre la figure

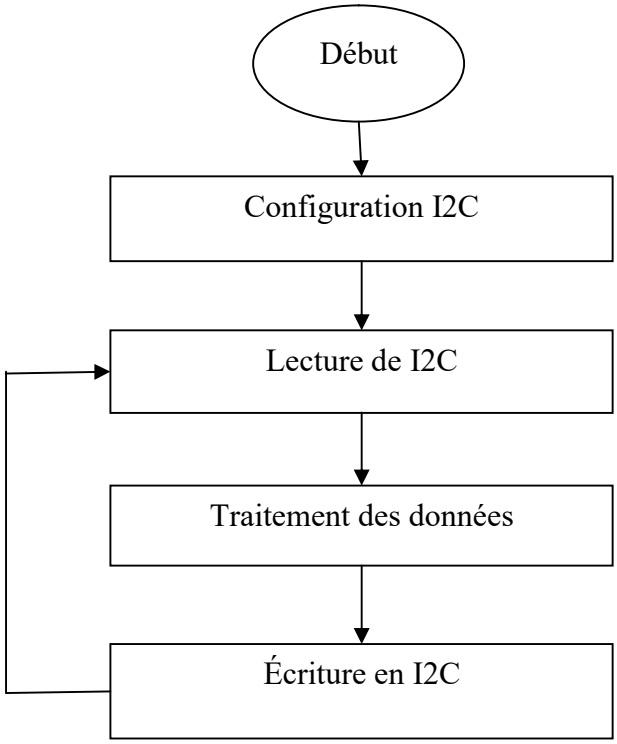

Fig 3.61 : L'organigramme de protocol I2C.

Pour utiliser I2C sur Arduino, procédez comme suit

- Configuration I2C :

Pour commencer à utiliser le protocole I2C, vous devez d'abord appeler la bibliothèque "Wire qui permet l'utilisation de ce protocole et lui donner sa propre adressecomme suit :

```
#include <Wire.h>
#define SLAVE_ADDRESS 0x04
```

- Lecture de I2C :

Pour recevoir des informations en utilisant le protocole I2C dans Arduino, la fonction doit être écrite comme suit:

```
void setup() {
   Wire.begin(SLAVE_ADDRESS 0x04);  // rejoindre le bus i2c avec adresse
   Wire.onReceive(receiveEvent); //enregistrer un événement

}
void loop() {

}
void receiveEvent(int howMany) {
  while (1 < Wire.available()) { // boucle à travers tout sauf le dernier
    char c = Wire.read(); // recevoir un octet en tant que personnage
    Serial.print(c);         // imprime le personnage
  }
}
```

La fonction "Wire.onReceive(receiveEvent)" appelle la fonction "receiveEvent" qui lit à son tour les informations de I2C En utilisant la phrase "Wire.read()".

Traitement des données : A ce stade, nous traitons l'information en fonction de nos besoins, par exemple:Imprimé.

- Écriture en I2C :

La fonction " Wire.onRequest(requestEvent)" appelle la fonction " requestEvent", qui à son tour envoie l'information via le protocole I2C en utilisant le terme " Wire.write(m)" comme le montre la figure suivante:

```
void setup() {
   Wire.begin(SLAVE_ADDRESS 0x04); // rejoindre le bus i2c avec adresse
   Wire.onRequest(requestEvent); //enregistrer un événement
}
void loop(){

}
void requestEvent() {
  Wire.write(m); // répondre avec un message de 6 octets

}
```

A. Surveillance du gaz

Dans cette partie du programme, nous mesurons le pourcentage de gaz dans la pièce et ensuite mettons en œuvre selon cette lecture un énoncé des instructions comme le montre l'organigramme de la figure 3.62.

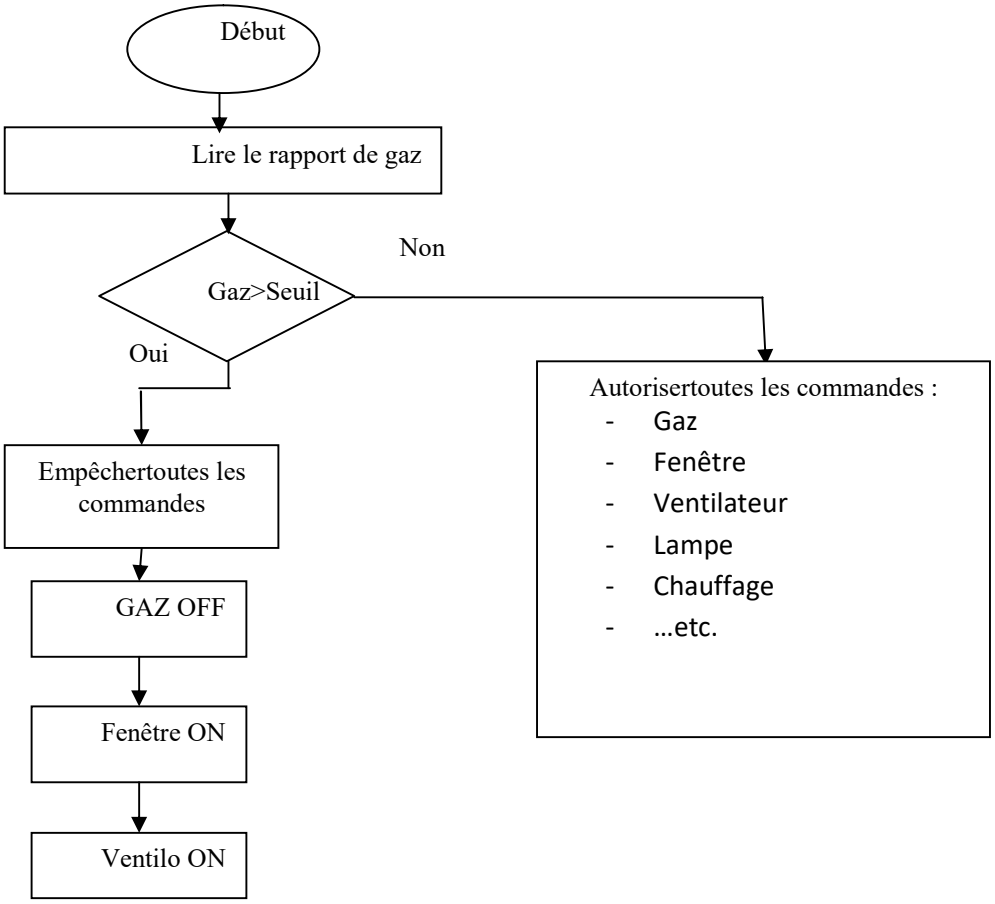

Fig 3.62 : Système Surveillance du gaz.

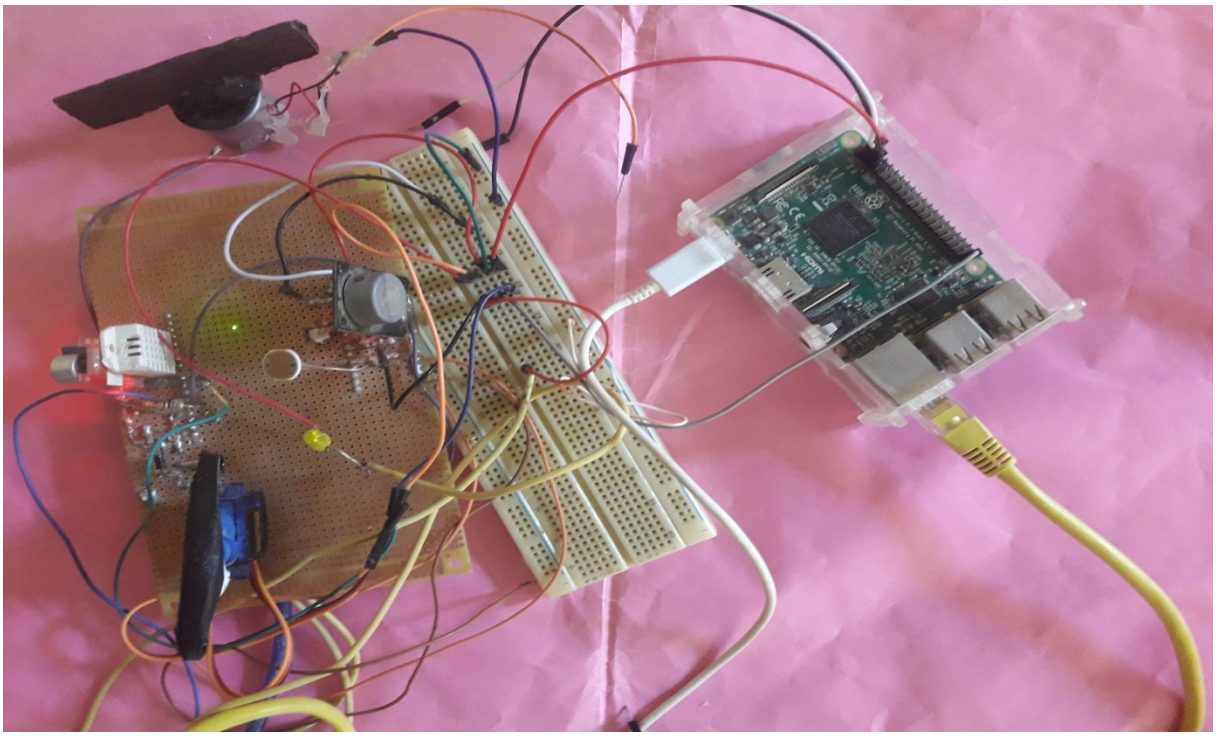

Fig 3.63: Image système Surveillance du gaz.

La lecture du pourcentage de gaz est comme suit.

```
int gaz(){
  sensorValue = analogRead(A2);
  sensorVoltage = sensorValue/1024*5.0;
  return(sensorVoltage);
}
```

Si le rapport de gaz est supérieur à un certain seuil, le programme éteindra le gaz et allumera le ventilateur et ouvrira la fenêtre.
Si le gaz est inférieur au seuil, le programme laisse l'utilisateur libre de choisir les fonctions voulues.

Void loop{

m = gaz();

servo(sorv, m , t);

temp(tompee ,m , vontoo); }

Ces fonctions se trouvent dans l'annexe (Program arduino).

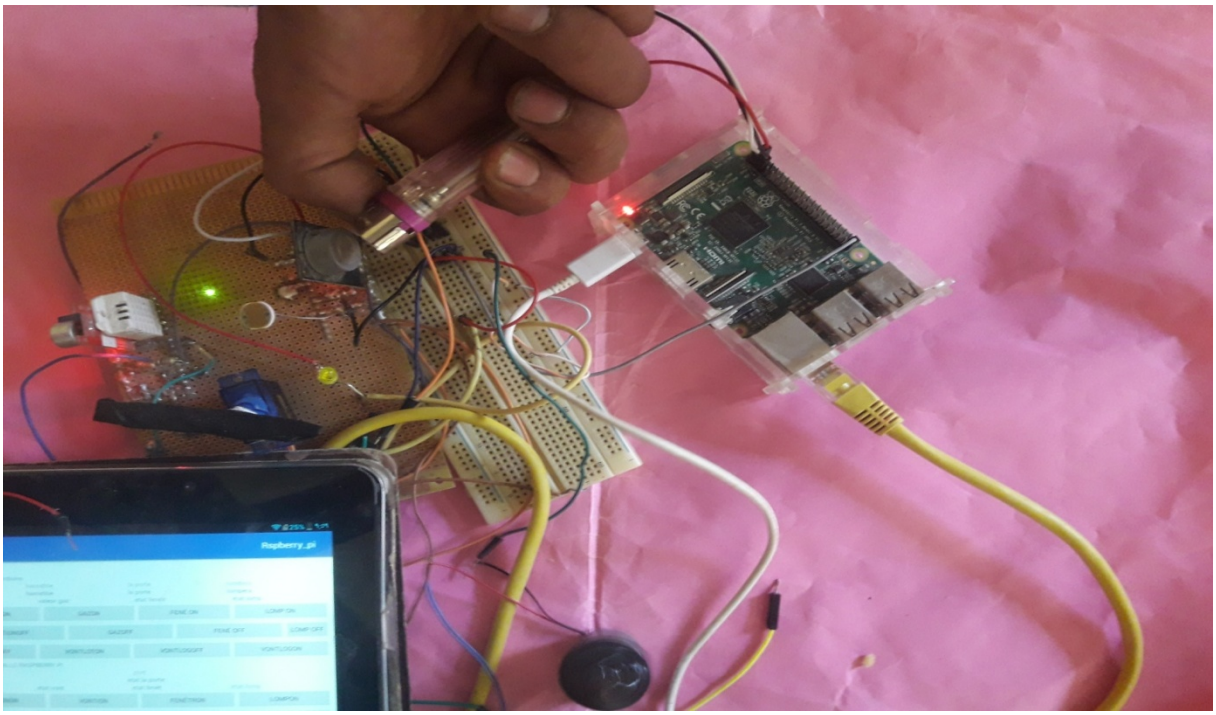

Fig 3.64: Fuite de gaz.

C. Surveillance de la chaleur

Dans cette partie du programme, nous mesurons le pourcentage de température et d'humidité dans la pièce et exécutons ensuite selon cette lecture l'action appropriée comme indiqué dans la figure 3.65.

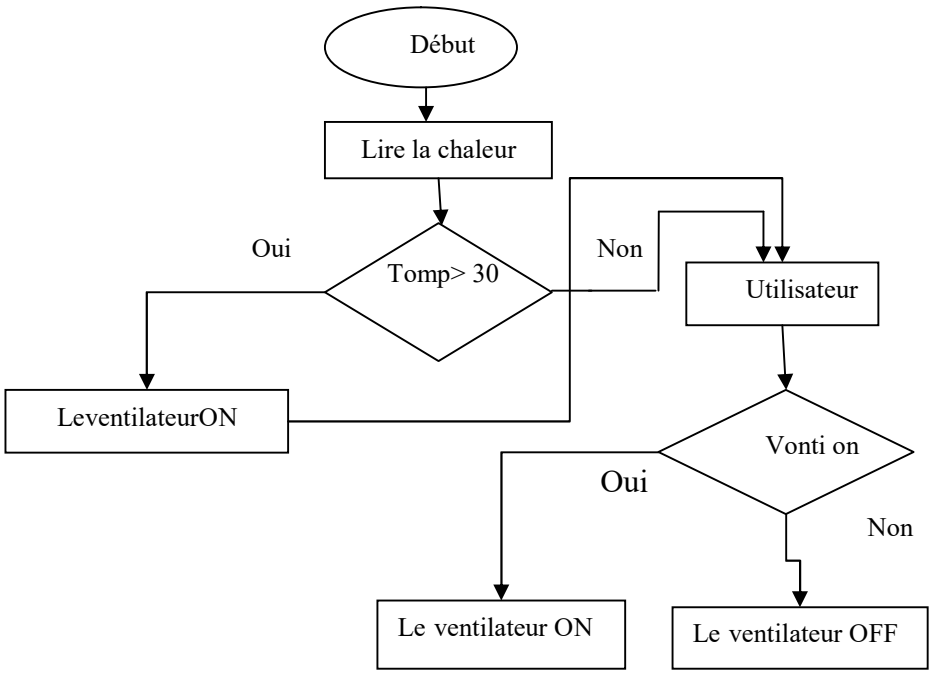

Fig 3.65 : Système de Surveillance de la chaleur.

Afin de mesurer la température et l'humidité à l'aide d'un DHT22, nous avons utilisé la bibliothèque "SimpleDHT22" puis créé un objet pour mesurer la température comme suit:

```
Void loop{
servo( sorv,  m , t);
 temp(tompee ,m , vontoo);
}
```

Ces fonctions se trouvent dans l'annexe (Program Arduino).

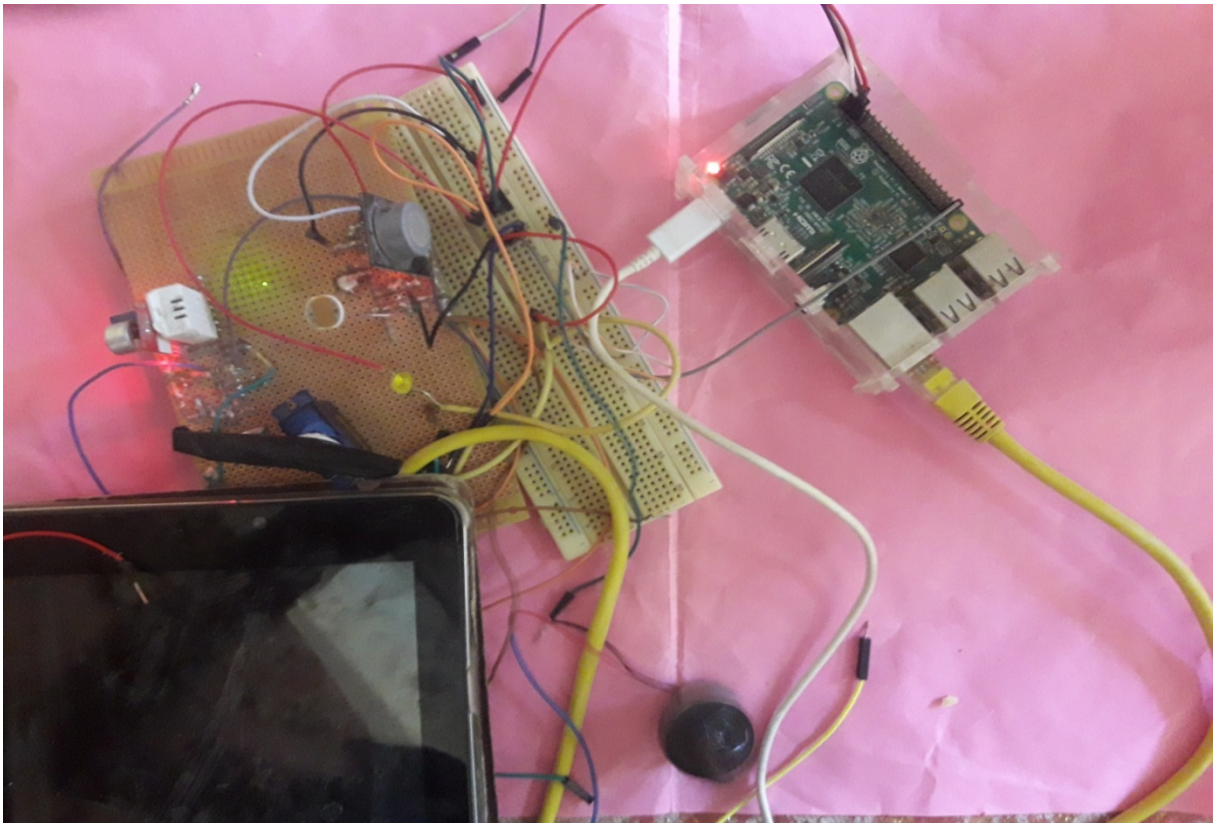

Fig 3.66: Image Système de Surveillance de la chaleur.

D. Contrôle de la lumière

Dans cette partie du programme, on contrôle la luminance comme indiqué dans l'organigramme de la figure 3.67.

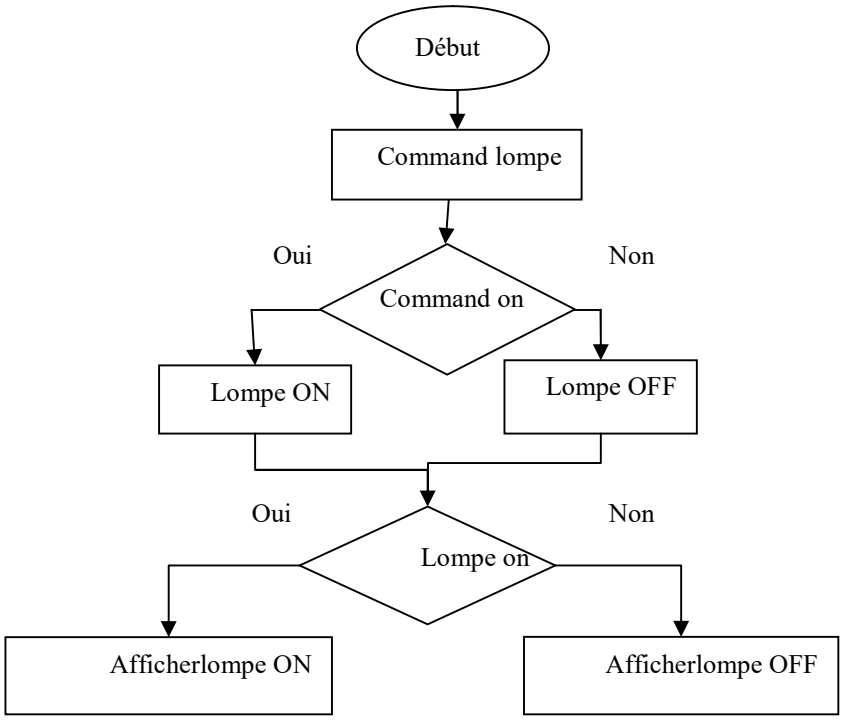

Fig 3.67 : Système de lumière de la maison.

Pour s'assurer que la lampe est réellement allumée, nous avons utilisé le capteur de lumière LDR avec le code suivant:

```
int LDR(){
  float ft = analogRead(A0);
  ft = (ft/1024*5.0);
  Serial.println(ft);
  if( ft  >= 4.0 ){

    return(12);
  }else{
    return(15);
  }
}
```

Void loop(){

int lo = asencron(lompeee);

lompe(lo ,m);

delay(100);

llompee = LDR();}

Ces fonctions se trouvent dans l'annexe (Program Arduino).

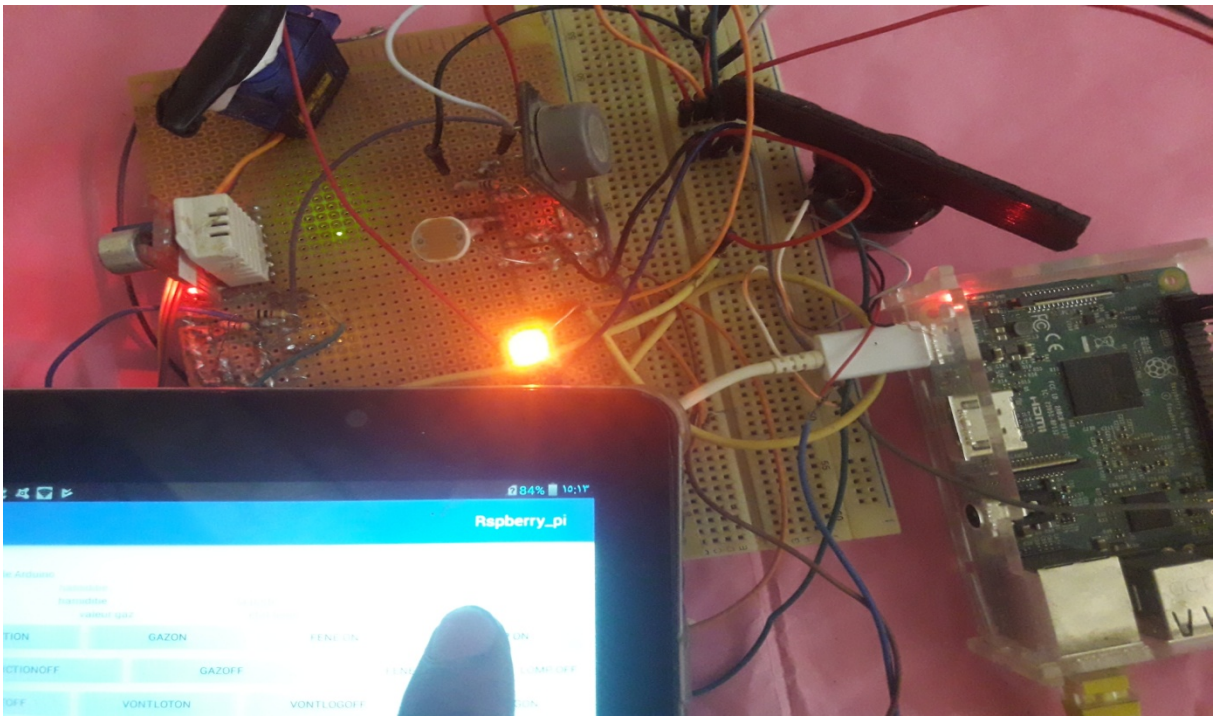

Fig 3.68: Allumez la lumière.

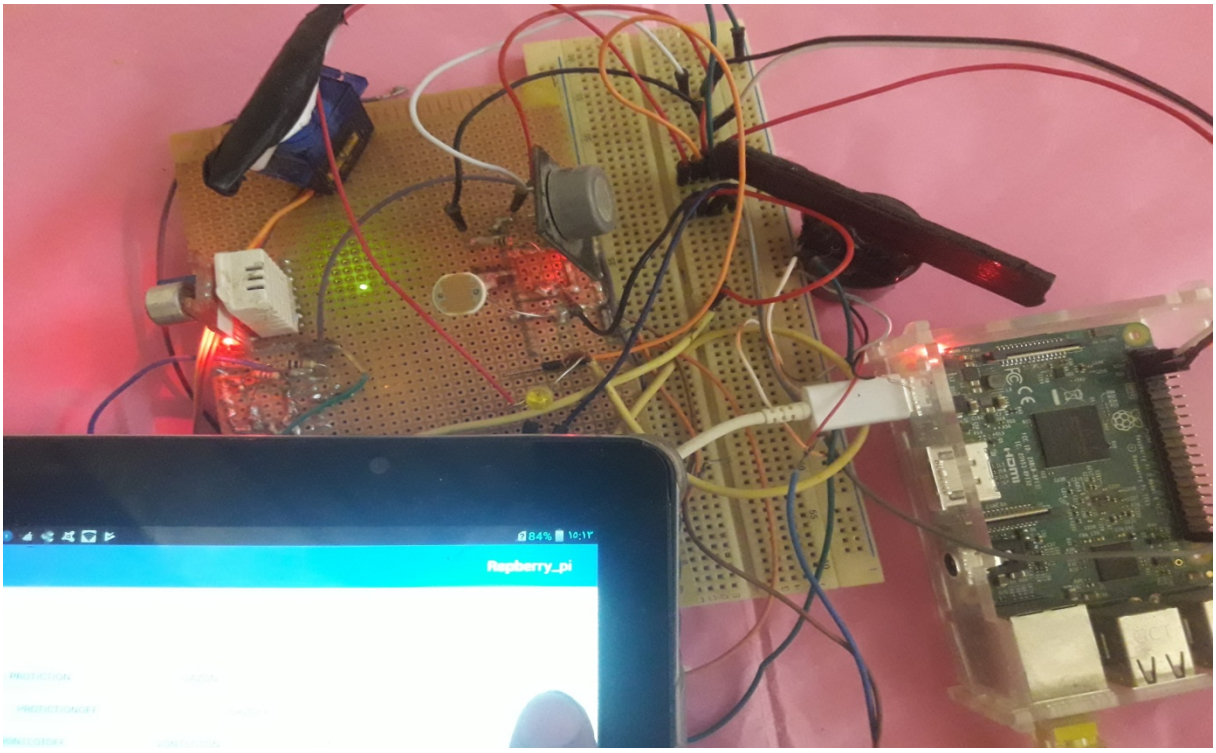

Fig 3.69 : Éteins la lumière.

E. Protection

Cette partie du programme surveille la présence d'un intrus à la maison selon la demande de l'utilisateur comme le montre la figure 3.70.

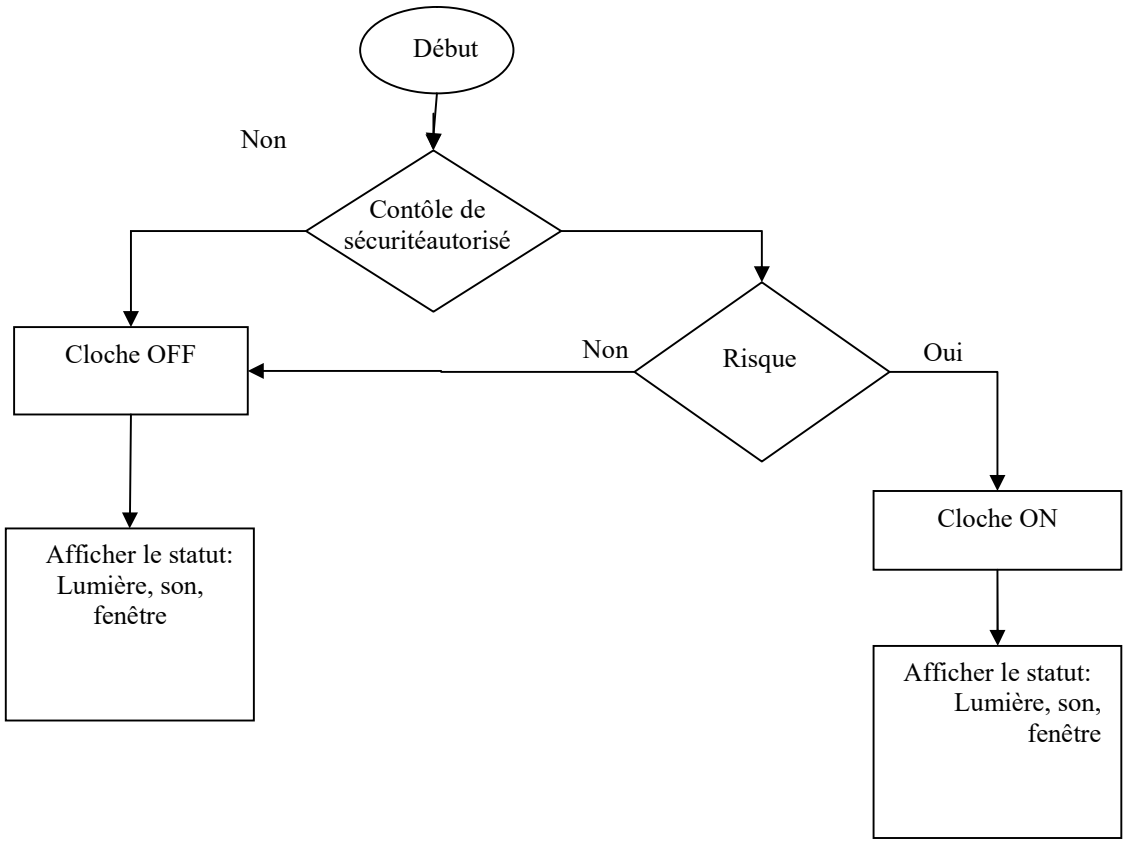

Fig 3.70: Système de protection de la maison.

Le code suivant montre la fonction de securité.

```
Void Loop(){
poortt = pp();
 ssoonn = fen();
 ssoonnm = sonnnn();
 compar(ssoonn, poortt ,ssoonnm);
compp = prisent(compar, tesst );
}
```

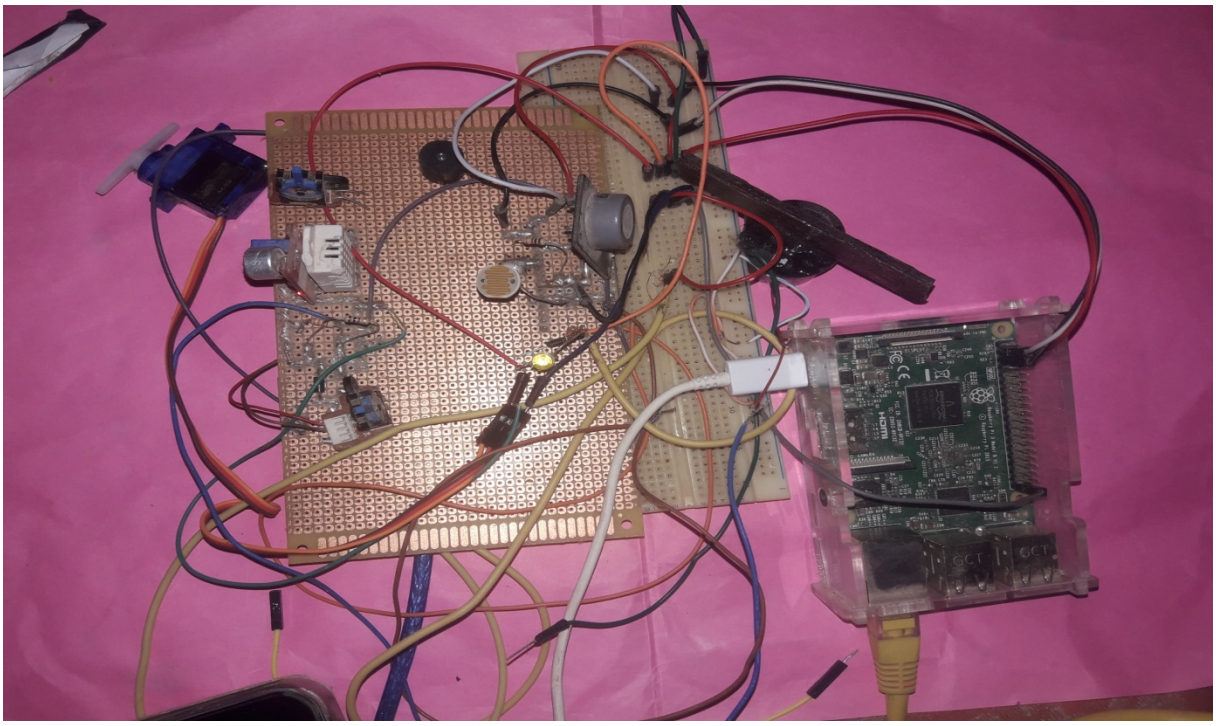

Fig 3.71 : Image système de protection de la maison.

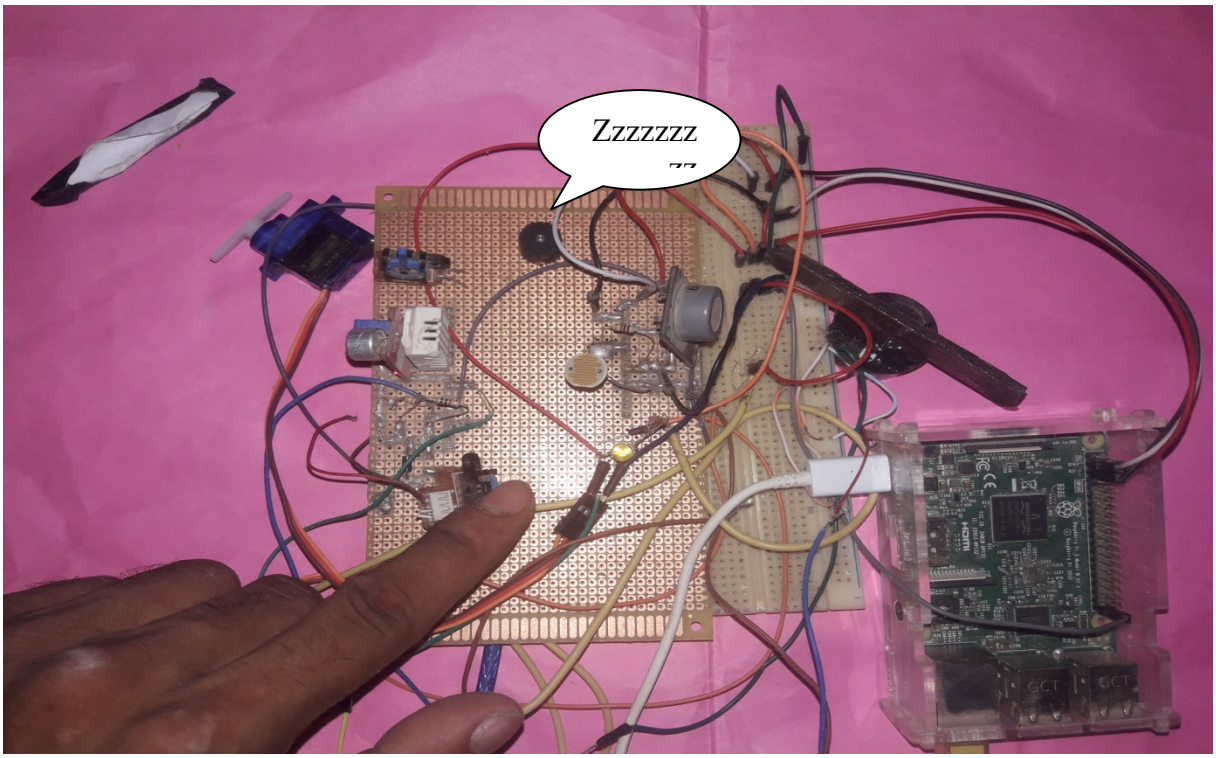

Fig 3.72: Protection de la maison (partie porte).

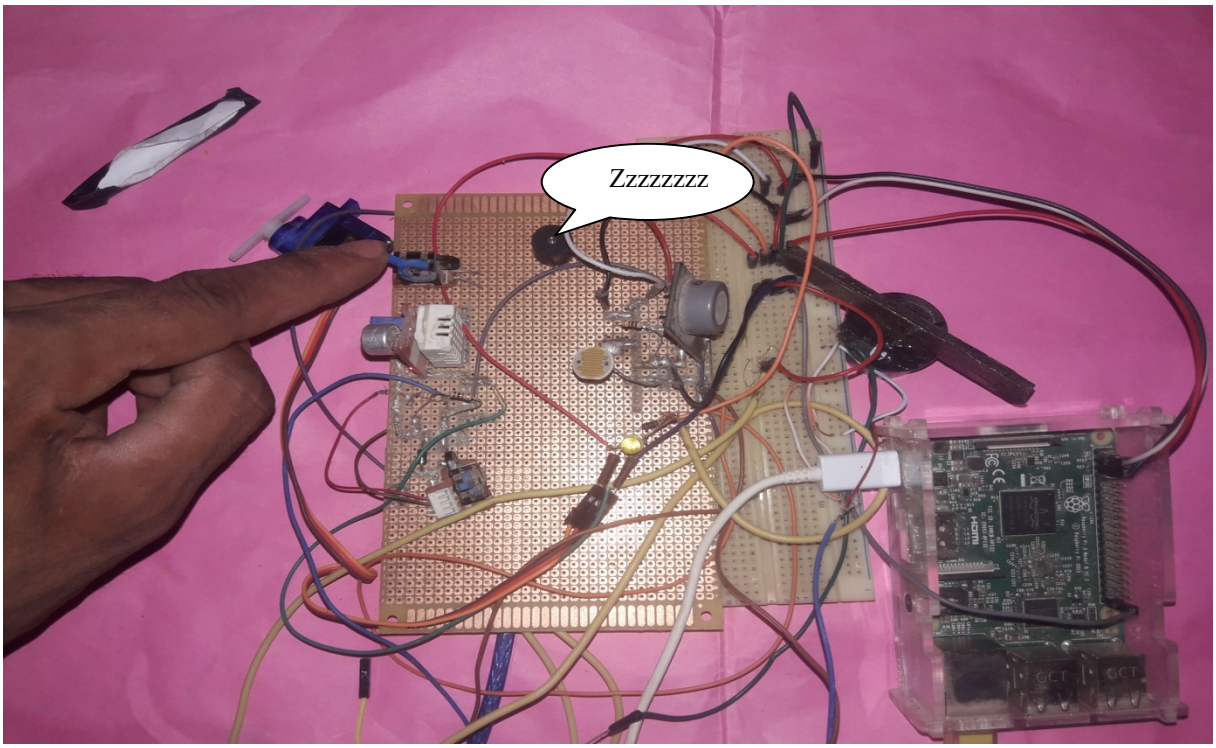

Fig 3.73: Protection de la maison (partie fenétr)

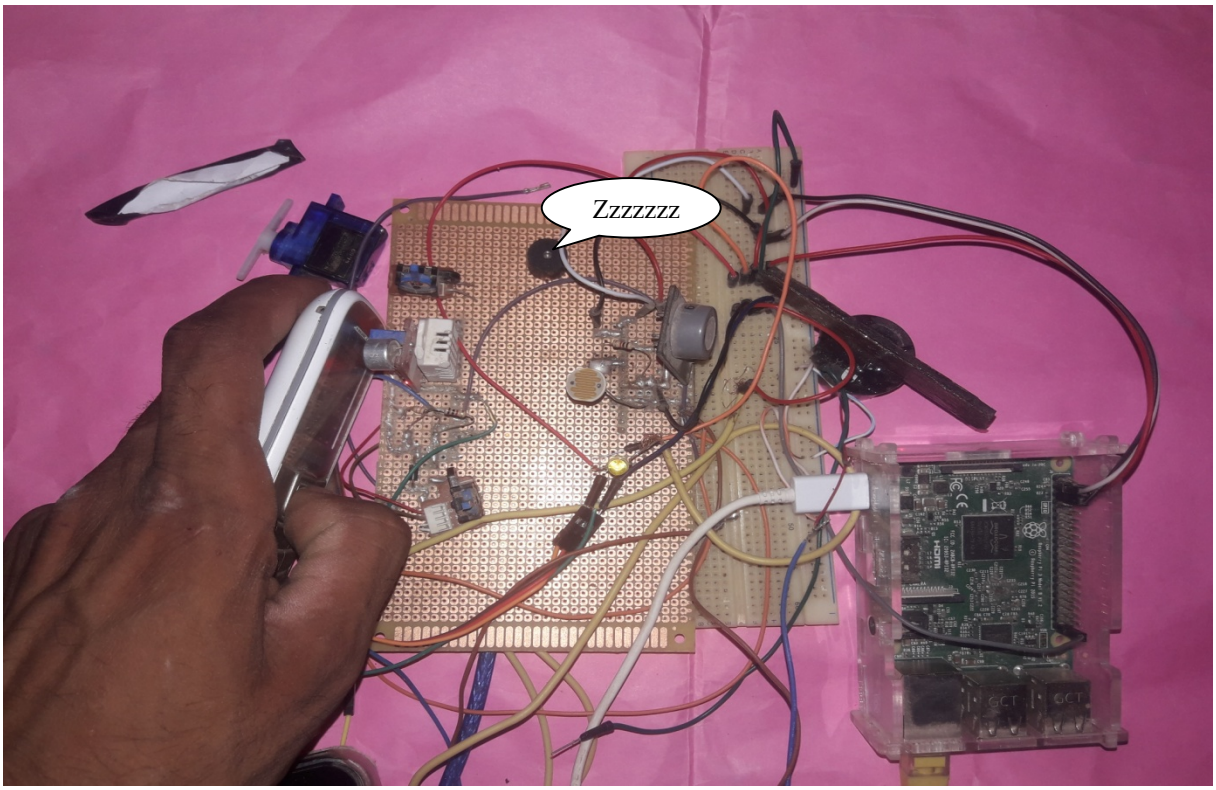

Fig 3.74: Protection de la maison (partie audio)

3.5. Conclusion

Dans ce chapitre on montré la relation entre les différents dispositifs constituants notre système. L'utilisateur à travers l'application Android gère toutes les fonctionnalités du système soit pour la transmission de commandes vers Raspberry et par la suite via Arduinovers les différents actionneurs : ventilo, fenêtre, lampe, …etc. Soit la lecture d'état de son système : température, humidité, gaz, …etc

Conclusion général

Conclusion général

Au bout de notre cursus en master "Electronique des Systèmes Embarqués ", nous avons été chargés de réaliser un projet de fin d'études. Notre travail s'est basé sur le développement d'une application sur les technologies mobiles (Smartphone) et la réalisation et la conception d'une maison intelligente. Ce projet était alors une occasion d'apprendre à travailler en d'une façon autonome et efficace. Ceci nous a amené à découvrir une nouvelle plateforme de développement et à enrichir nos connaissances théoriques ainsi que pratiques et notre expérience dans le domaine des IoT. Ce dernier qui est devenu un grand domaine de recherche et un grand marché de travail au monde.

Dans ce projet on a travaillé sur trois niveaux. Le premier niveau est l'interface utilisateur, c'est-à-dire l'application Android. L'utilisateur à tout droit de commander via internet les différents actionneurs dans sa maison ainsi que recevoir toute information des capteurs installés. Le deuxième niveau est l'application Raspberry, qui a plusieurs fonctions :

D. Assure la communication client/serveur avec l'utilisateur.
E. Contrôle des E/S numériques.
F. Intermédiaire entre l'utilisateur et Arduino pour plus de fonction surtout les entrées analogiques.

Dans le troisième niveau, le niveau bas, on trouve l'Arduino qui communique via I2C avec Raspberry pour la transmission des informations et pour la reception des commandes.

Finalement, on espère par notre travail, apporter une validation pratique de ces techniques et donner une bonne cause pour mieux explorer ce domaine d'internet des objets.

Bibliographie

Bibliographie

[1] Achat Asalas et Laoubi Lyes « Conceptin et réalissation d'une application mobile cross-platform pour I'nternet of things », Master, Université de Béjaia, 2016/2017.

[2] Carolyn Marsan « Understanding the Issues and of a More Connected World » , Internet Society ,October 2015.

[3] Baghdadli Chahrazed et Ferouani Zineb « Developpment d'une Application deployee sur un reseau de capteur sans fil supportant 6LoWPAN », Master, Université Abou Bakr Belkaid-Tlemcen, 2014/2015.

[4] Bouharaoua Abderrahim et Boukli Haccene « Automatisation d'une maison intelligente via une application android », Master, Université Abou Bakr Belkaid-Tlemcen, 2016/2017.

[5] WWW.TUTORIALSPOINT.COM/Internet of things**.**

[6] https://fr.wikipedia.org/wiki/Ville_intelligente/ les technologies de l'information et de la communication (TIC).

[7]https://www.connaissanced esenergies.org/fiche-pedagogique/reseau-intelligent-smart-grid.

[8]https://fr.wikipedia.org/wiki/Maison_intelligente

[9] Hamouchi Hamid et Jbari Atman « Conciptin et réalisation d'une centrale embarquée de la domotique », Master, Université Mohammed V de Rabat , 2014/2015.

[10] Gasmi Mohamed « Commande visuelle d'un robot mobile », Master, Université Mohamed Khider Biskra ,2016/2017.

[11]https://en.wikipedia.org/wiki/Raspberry_Pi.

[12] Bouamor Boujemaa et Oucha Mohamed « conception et réalistion d'un système de surveillance d'une salle des serveurs à base de Raspberry Pi », Master, Université Sidi Mohamed Ben Abdellah ,2015/2016.

[13] Simon Monk « Programez un Raspberry Pi », DUNOD ,2013.

[14] https://www.elektor.fr/raspberry-pi-3-model-b, (elektor).

[15] http://nicolargo.developpez.com/tutoriels/raspberry-pi/-installation

[16] Simon Monk « Raspberry Pi Kookbook » O'REILLY, 2014.

[17] Riahi Ilham « Etude, simulation et réallisation de mini-générateurs BF et d'un mini-Voltmètre AC-DC piloté par une carte ArduinoUno R3 », Master, Université Aboubakr Belkaid- tlemcen ,2017.

[18] Krama Abdelbasset et GouguiAblelmoumen «Etude et réalistion d'une carte de contrôle par Arduino via le système Androide », Master, Universite kasdi Merbah Ouargla.

[19] https://www.farnell.com/datasheets/1682209.pdf.

[20] http://www.craslab.org/interaction/files/LivretArduinoCRAS.pdf, (Initiation à la mise en oeuvre matérielle et logicielle de l'Arduino novembre 2006).

[21] http://projet.eu.org/pedago/sin/tutos/arduino.pdf (Classes de 2nde SI-CIT et de première SI).

[22] https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino/les-capteurs-electroniques

[23] https://ardwinner.jimdo.com/arduino/v-les-servomoteurs

[24] https://www.mouser.com/.../DHT11-Technical-Data-Sheet-Translat.

[25] http://sensorkit.fr.joy-it.net/index.php?title=KY-038_Capteur_sonore,(Sensorkit X40).

[26] https://www.scribd.com/document/.../Datasheet-sensor-MQ9.

[27] https://android-studio.fr.uptodown.com/windows.

[28] Chryssa Aliferi « Android Programming Cook Book », Exelixis Media P.C., 2016

[29] https://developer.android.com/studio/write/layout-editor.

[30] https://developer.android.com/studio/projects/templates.

[31] https://developer.android.com/guide/topics/manifest/manifest-intro.

[32] https://developer.android.com/reference/android/.

[33] https://www.raspberrypi.org/documentation/usage/python/README.md.

Les Annexes

Le mode de réalisation dans un stéréogramme

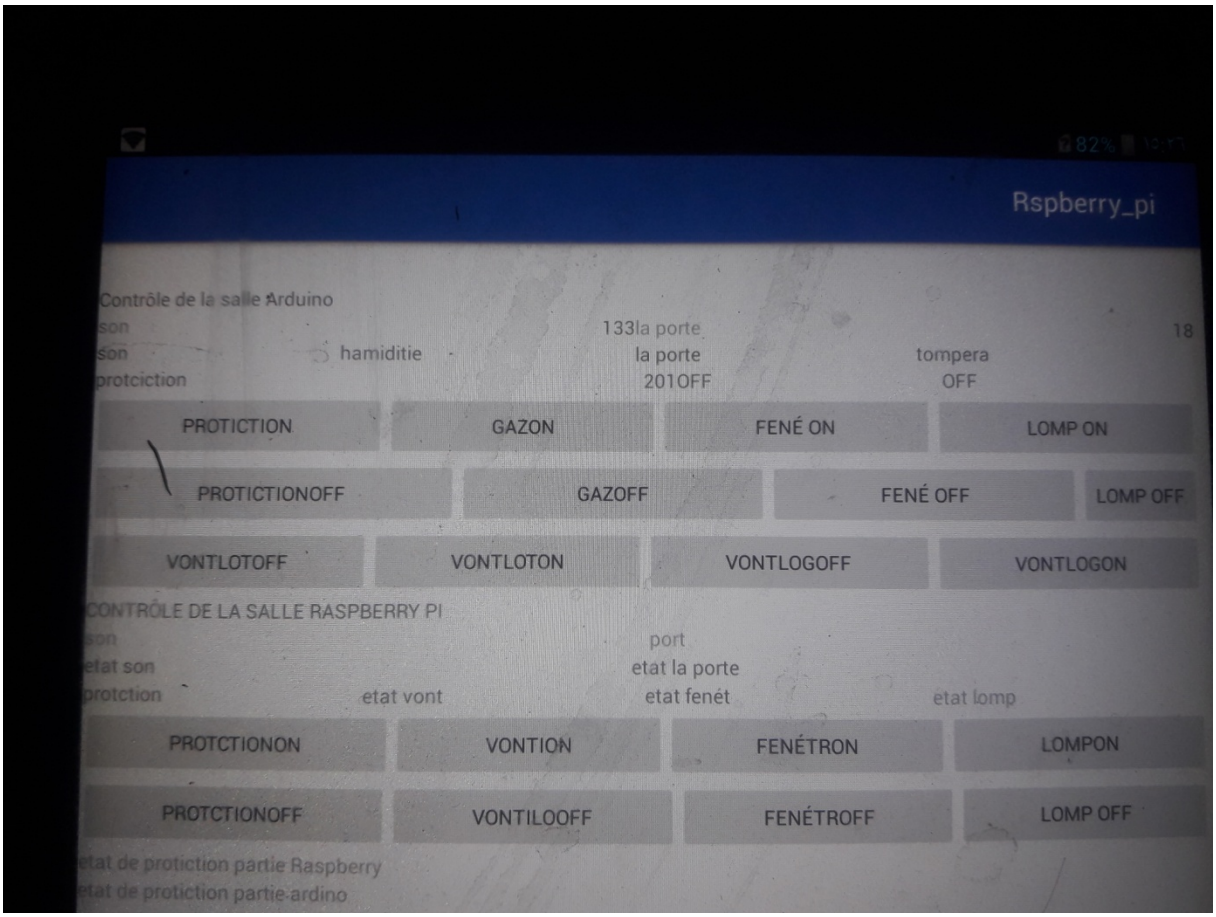

Figure 1.

Figure 2.

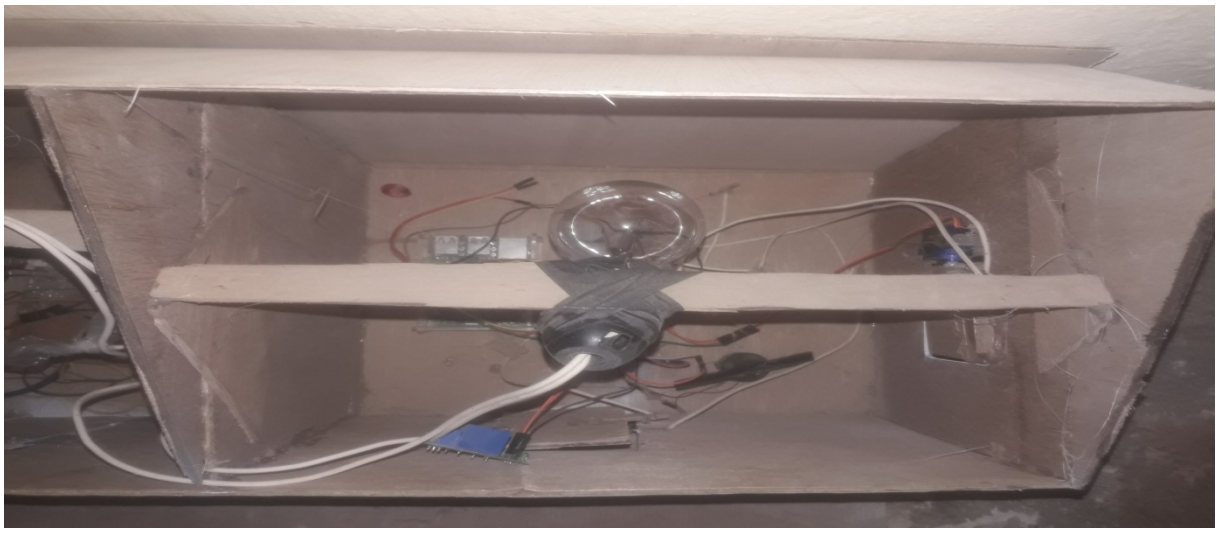

Figure 3.

Figure 4.

Figure 5.

Figure 6.

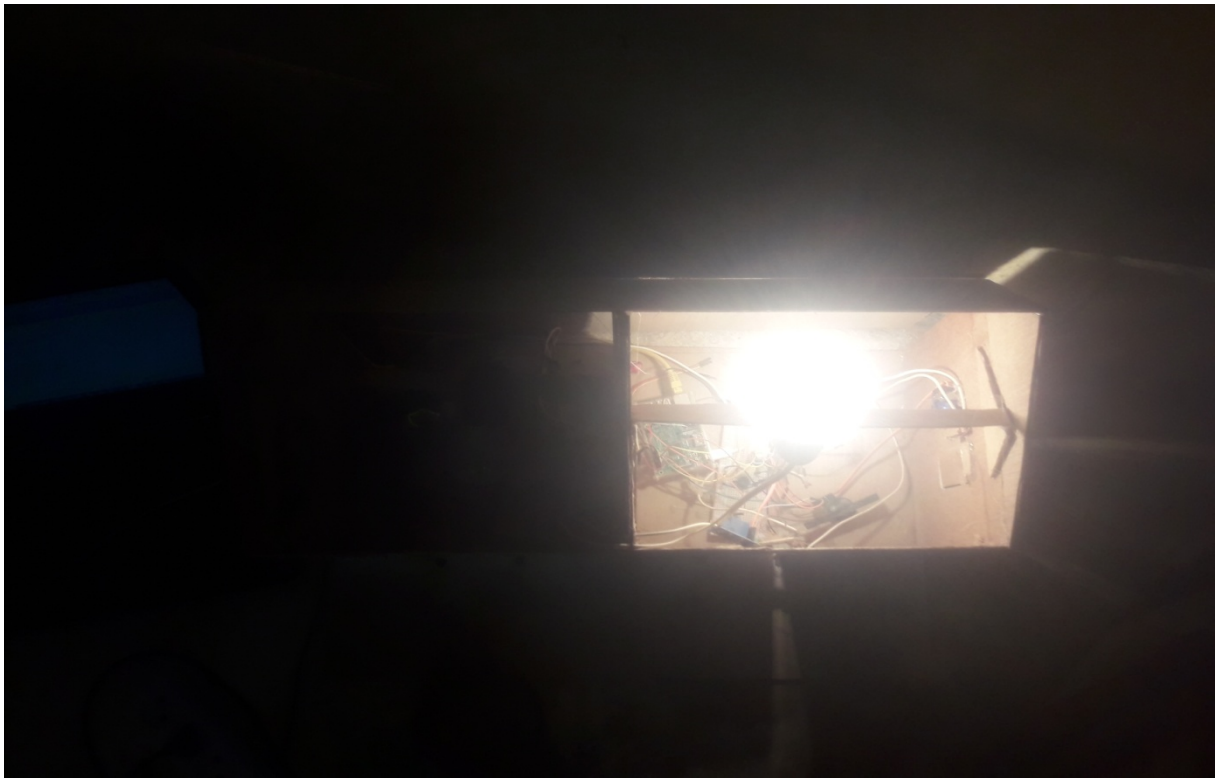

Figure 7.

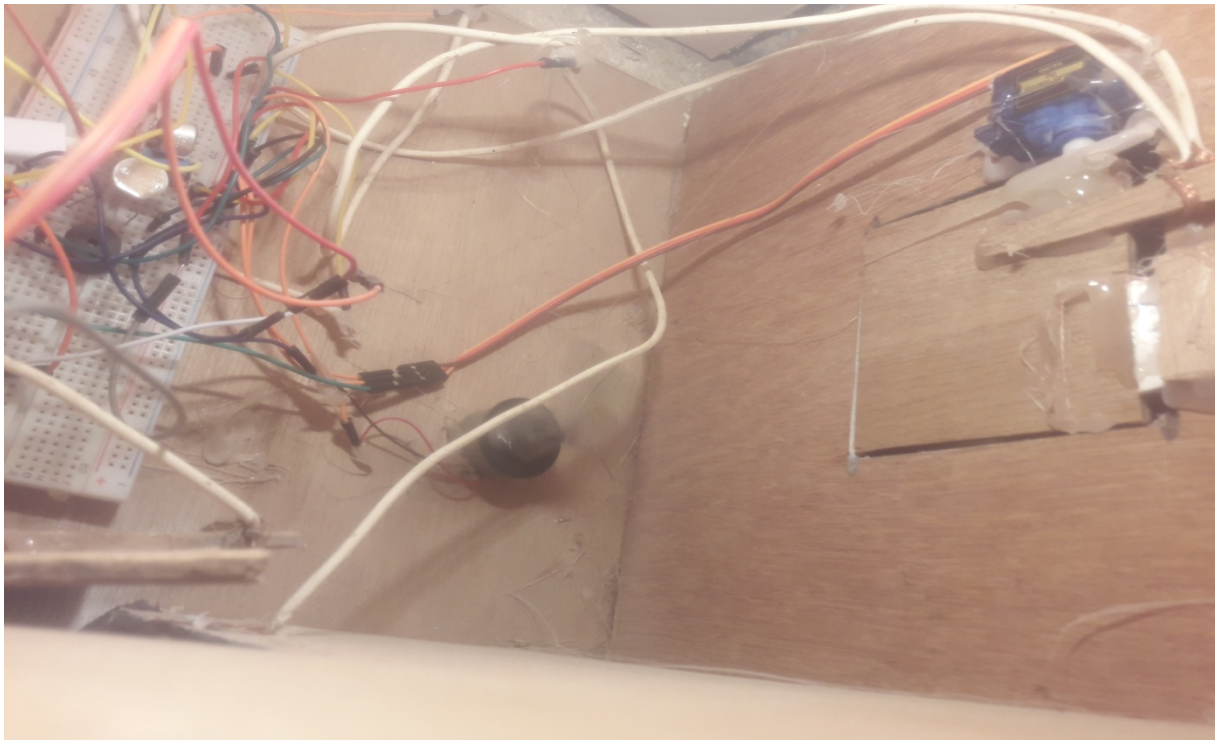

Figure 1.

Figure 8.

Figure 9.

Programme Android

```
package com.example.hpcore7.rspberry_pi;


import android.os.AsyncTask;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class MainActivity extends AppCompatActivity {

    private static String ip = "192.168.43.5";
    private static int port = 2004;
    private static Socket s;
    private static ServerSocket ss;
    private static InputStreamReader isr;
    private static BufferedReader br;
    String massige ="";
    String massige1 ="";
    private static PrintWriter pp;
    EditText test1;
    TextView txtshow1;
    TextView txtshow2;
    TextView txtshow3;
    TextView txtshow4;
    TextView txtshow5;
    TextView txtshow6;
    TextView txtshow7;
    TextView txtshow8;
    TextView txtshow9;
    TextView txtshow10;
    TextView txtshow11;
    TextView txtshow12;
    TextView txtshow13;
    TextView txtshow14;
    TextView txtshow15;
    TextView txtshow16;
    private String ipaddress;
    private int portnum;
    int foo;
    int foo1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        txtshow1 =(TextView) findViewById(R.id.porte);
       txtshow2 =(TextView) findViewById(R.id.hamiditie);
       txtshow3 =(TextView) findViewById(R.id.sonn);
        txtshow4 =(TextView) findViewById(R.id.lompeonoff);
        txtshow5 =(TextView) findViewById(R.id.fenétreonoff);
        txtshow6 =(TextView) findViewById(R.id.gazonoff);
        txtshow7 =(TextView) findViewById(R.id.protiction);
        txtshow8 =(TextView) findViewById(R.id.portR);
        txtshow9 =(TextView) findViewById(R.id.sonR);
        txtshow10 =(TextView) findViewById(R.id.lompR);
        txtshow11 =(TextView) findViewById(R.id.fenétreR);
        txtshow12 =(TextView) findViewById(R.id.vontiloR);
        txtshow13 =(TextView) findViewById(R.id.protictionR);
        txtshow14 =(TextView) findViewById(R.id.tomperature);
        txtshow15 =(TextView) findViewById(R.id.testard);
        txtshow16 =(TextView) findViewById(R.id.testrasp);

        Thread myThread = new Thread(new recerv());
        myThread.start();
        Thread myThread1 = new Thread(new recerv1());
        myThread1.start();

    }

    public void lompeON(View view) {
        massige = "cc";

        myTask mt = new myTask();

        mt.execute();
    }

    public void fenétreON(View view) {
        massige = "ee";

        myTask mt = new myTask();

        mt.execute();
    }

    public void gazon(View view) {
        massige = "b";

        myTask mt = new myTask();

        mt.execute();
    }

    public void protictionON(View view) {
        massige = "dd";

        myTask mt = new myTask();

        mt.execute();
    }

    public void protictionOFF(View view) {
        massige = "e";
```

```
        myTask mt = new myTask();

        mt.execute();
    }

    public void lompeOFF(View view) {
        massige = "d";

        myTask mt = new myTask();

        mt.execute();
    }

    public void fenétreOFF(View view) {
        massige = "f";

        myTask mt = new myTask();

        mt.execute();
    }

    public void gazoff(View view) {
        massige = "bb";

        myTask mt = new myTask();

        mt.execute();
    }

    public void vontilogazon(View view) {
        massige = "ff";

        myTask mt = new myTask();

        mt.execute();
    }

    public void vontilogazoff(View view) {
        massige = "g";

        myTask mt = new myTask();

        mt.execute();
    }

    public void vontilotompon(View view) {
        massige = "a";

        myTask mt = new myTask();

        mt.execute();
    }

    public void vontilotompoff(View view) {
        massige = "aa";

        myTask mt = new myTask();

        mt.execute();
    }
```

```
public void protcttionR(View view) {
    massige = "bb";

    myTask mt = new myTask();

    mt.execute();
}



public void fenétroffR(View view) {
    massige = "aaa";

    myTask mt = new myTask();

    mt.execute();
}

public void fenétronR(View view) {
    massige = "aa";

    myTask mt = new myTask();

    mt.execute();
}

public void vontilooR(View view) {
    massige = "abb";

    myTask mt = new myTask();

    mt.execute();
}

public void vontiloonR(View view) {
    massige = "ab";

    myTask mt = new myTask();

    mt.execute();
}

public void protcttioffR(View view) {
    massige = "a";

    myTask mt = new myTask();

    mt.execute();
}

public void lomponR(View view) {
    massige = "acc";

    myTask mt = new myTask();

    mt.execute();
}

public void lompoffR(View view) {
    massige = "ad";
```

```
        myTask mt = new myTask();

        mt.execute();
    }


    class myTask extends AsyncTask<Void ,Void ,Void > {
        @Override
        protected Void doInBackground(Void... params) {


            try {
                s = new Socket(ip, port);
                pp = new PrintWriter(s.getOutputStream());
                isr = new InputStreamReader(s.getInputStream());
                br = new BufferedReader(isr);


                pp.write(massige);
                pp.flush();

                pp.close();
                isr.close();
                br.close();
                s.close();


            } catch (Exception e) {
                e.printStackTrace();
            }


            return null;
        }
    }

    class recerv implements Runnable {

        private  Socket s;
        private ServerSocket ss;
        private  InputStreamReader isr;
        BufferedReader brr;
        Handler h = new Handler();

        String MSSM;

        @Override
        public void run(){
            try{
                ss = new ServerSocket(2004);
                while (true){
                    s = ss.accept();
                    isr = new InputStreamReader(s.getInputStream());
                    brr = new BufferedReader(isr);
                    MSSM = brr.readLine();
                      foo = Integer.parseInt(MSSM);
                    h.post(new Runnable() {
                        @Override
                        public void run() {
                            if(foo >= 0 && foo < 100){
```

```
                                foo = foo - 20;
                                String str = Integer.toString(foo);
                                txtshow14.setText(str);
                            }
                            else if(foo == 210){
                                txtshow1.setText("ON");
                            }
                            else if(foo == 211){
                                txtshow1.setText("OFF");
                            }

                            else if(foo >= 100 && foo < 200){
                                txtshow2.setText(MSSM);
                            }
                            else if(foo == 216){
                                txtshow3.setText("ON");
                            }
                            else if(foo == 217){
                                txtshow3.setText("OFF");
                            }
                            else if(foo == 208){
                                txtshow4.setText("ON");
                            }
                            else if(foo == 209){
                                txtshow4.setText("OFF");
                            }
                            else if(foo == 212){
                                txtshow5.setText("ON");
                            }
                            else if(foo == 213){
                                txtshow5.setText("OFF");
                            }
                            else if(foo >= 201 && foo <207){
                                txtshow6.setText(MSSM);
                            }

                            else if(foo == 214){
                                txtshow7.setText("ON");
                            }
                            else if(foo == 215){
                                txtshow7.setText("OFF");
                            }
                            else if(foo == 218){
                                txtshow15.setText("Il n'y a pas de
danger");
                            }
                            else if(foo == 219){
                                txtshow15.setText("Il y a un risque");
                            }

                        }
                    });
                }
            }catch(IOException e) {
            }
        }
    }

    class recerv1 implements Runnable {

        private  Socket s1;
```

```
        private ServerSocket ss1;
        private  InputStreamReader isr1;
        BufferedReader brr1;
        Handler h = new Handler();

        String MSSM1;

        @Override
        public void run(){
            try{
                ss1 = new ServerSocket(2005);
                while (true){
                    s1 = ss1.accept();
                    isr1 = new InputStreamReader(s.getInputStream());
                    brr1 = new BufferedReader(isr);
                    MSSM1 = brr1.readLine();
                    foo1 = Integer.parseInt(MSSM1);
                    h.post(new Runnable() {
                        @Override
                        public void run() {

                           if(foo1 == 300){
                                txtshow8.setText("ON");
                            }
                            if(foo1 == 301){
                                txtshow8.setText("OFF");
                            }
                            else if(foo1 == 303){
                                txtshow9.setText("ON");
                            }
                            else if(foo1 == 304){
                                txtshow9.setText("OFF");
                            }
                            else if(foo1 == 305){
                                txtshow10.setText("ON");
                            }
                            else if(foo1 == 306){
                                txtshow10.setText("OFF");
                            }
                            else if(foo1 == 350){
                                txtshow11.setText("Il n'y a danger");
                            }
                            else if(foo1 == 340){
                                txtshow11.setText("Il n'y a pas de
danger");
                            }
                           // if(foo1 == 309){
                            //    txtshow11.setText("ON");
                            //}
                            //if(foo1 == 310){
                             //   txtshow11.setText("OFF");
                            //}
                           // else if(foo >= 100 && foo < 200){
                              //  txtshow13.setText(MSSM1);
                            //}
                            if(foo1 == 307){
                                txtshow16.setText("ON");
                            }
                            if(foo1 == 308){
                                txtshow16.setText("OFF");
                            }
```

```
                        }
                    });
                }
            }catch(IOException e) {
            }
        }
    }

}
```

Programme python

```
import RPi.GPIO as GPIO
import smbus
import socket
import decimal
import sys
from threading import Thread, RLock
import time

GPIO.setmode(GPIO.BOARD)
        GPIO.setup(7, GPIO.OUT)
        GPIO.setup(37, GPIO.OUT)
        GPIO.setup(8, GPIO.IN)
        GPIO.setup(11, GPIO.IN)
        GPIO.setup(13, GPIO.IN)
        GPIO.setup(12, GPIO.IN)
        GPIO.setup(16, GPIO.IN)
        GPIO.setup(40, GPIO.OUT)
        GPIO.setup(38, GPIO.OUT)


class Afficheur(Thread):

        def __init__(self):
        Thread.__init__(self)
        self.fff = ''
        self.por = 0
        self.daata = 0
        def run(self):
        while 1:

        s = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
        s.bind(('', 2004))

        s.listen(3)
        while 1:

        c, addr = s.accept()
        print "connection from: " + str(addr[0])
        self.fff = addr[0]

        thex = ddd(addr[0])
        thex1 = ddd1(addr[0])



        data = c.recv(1024)
        l = data
        if str(l) == 'a':
        self.daata = 0
        elif str(l) == 'aa':
        self.daata = 1

        elif str(l) == 'b':
        self.daata = 5
        elif str(l) == 'bb':
```

```
        self.daata = 10
        elif str(l) == 'c':
        self.daata = 15
        elif str(l) == 'cc':
        self.daata = 20
        elif str(l) == 'd':
        self.daata = 25
        elif str(l) == 'dd':
        self.daata = 30
        elif str(l) == 'e':
        self.daata = 35
        elif str(l) == 'ee':
        self.daata = 40
        elif str(l) == 'f':
        self.daata = 45
        elif str(l) == 'aa':
        self.daata = 0
        elif str(l) == 'aaa':
        self.daata = 1

        elif str(l) == 'ab':
        self.daata1 = 5
        elif str(l) == 'abb':
        self.daata1 = 10
        elif str(l) == 'ac':
        self.daata1 = 15
        elif str(l) == 'acc':
        self.daata1 = 20
        elif str(l) == 'ad':
        self.daata1 = 25
        elif str(l) == 'add':
        self.daata1 = 30
        elif str(l) == 'ae':
        self.daata1 = 35
        elif str(l) == 'aee':
        self.daata1 = 40
        elif str(l) == 'af':
        self.daata1 = 45



        if not data:
        break
        print "from connected user: " + str(l)

        v = 0
        if v == 0 and self.fff == addr[0]:
        v = 1

        thex.start()
        thex1.start()


        c.close()


class exemple(Thread):
        def __init__(self):
        Thread.__init__(self)
        self.list = 0
        def run(self):
```

```
        bus = smbus.SMBus(1)
        address = 0x04


        def writeNumber(value):
        bus.write_byte(address , value)
        return -1
        def readNumber():
        number = bus.read_byte(address)
        return number
        i = 0

        while True:

        #var = int(Decimal(thread_1.daata))
        #if not var:
        #continue

        var = thread_1.daata
        va = var


        writeNumber(va)
        #print "RPI: Hi Arduino, I sent you ", var

        time.sleep(0.45)
        t = readNumber()
        self.list = t



class ddd(Thread):
        def __init__(self, mot):
        Thread.__init__(self)
        self.mot = mot
        self.listt = 0
        def run(self):

        messag = ' '


        while True:
        messag = str(theex.list)
        c = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
        c.connect((self.mot , 2004))

        c.send(messag)
        c.close()

        #print(h)
        time.sleep(0.45)

class exemple1(Thread):
        def __init__(self):
        Thread.__init__(self)
        self.list = 0
        self.l= 0
        self.hh = 0
        self.hhh = 0
        self.g = 0
        self.k = 0
```

```
self.j = 0
self.ll=0
self.lll=0
self.vaa = 0
self.var = 0
self.va = 0
self.vq = 0
self.vaz =0
def run(self):
def son():
if GPIO.input(13) == True:
return(305)
else:
return(306)
def porte():
if GPIO.input(12) == True:
return(300)
else:
return(301)
def fetr():
if GPIO.input(16) == True:
return(303)
else:
return(304)
def vontilo(comand1 , comand2):
if comand1 < 2 :
if comand2 == 5:
GPIO.output(40, true)
else:
GPIO.output(40, true)
else:
time.sleep(0.25)

def protction(p ,f , s ):
if p == 511 or s == 521 or f == 501:
self.lll = 700

def protction1(comand):

if self.lll == 700:
GPIO.output(38, True)
else:
comand == 0
GPIO.output(38, False)
if comand == 15:
self.lll =770
#self.llll =7070
else:
self.lll= 700
def benkhalfa_ldr():
if GPIO.input(8) == True :
return 307

else:
return 308


def benkhalfa_ON_OFF_L(command1, command2):
if command1 < 2:
if command2 == 20:
GPIO.output(37, True)
```

```
else:

GPIO.output(37, False)

else:
time.sleep(0.25)
def benkhalfa_commandAP(command):
if command == 20:
com = 20
else:
com = 25

if com == 20:
self.hhh = 0
self.hh +=1
else:
self.hh = 0

self.hhh +=1

if self.hh == 1 :
self.l = 4
if self.hhh == 1 :
self.l = 5
p = 0
def setup():
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
global p
p = GPIO.PWM(7,50)
p.start(2.5)
p.ChangeDutyCycle(2.5)

def servoup(var):
global p
#var = 20
if var == 0:
p.ChangeDutyCycle(7)
time.sleep(1)
else:
time.sleep(1)
def servodown(var):
global p
#var =25
if var == 1:
p.ChangeDutyCycle(2.5)
time.sleep(1)
else:
time.sleep(1)
def close():
p.stop()
i = 0

while True:

var = thread_1.daata

if var < 5:
self.va = var
elif var <= 10 and var > 5:
self.vaa = var
```

```
        elif var < 30 and var > 15:
        self.vaz = var
        elif var < 40 and var > 30 :
        self.vr = var
        elif var <50 and var > 40:
        self.vq = var
        i+=1
        time.sleep(0.45)
        benkhalfa_commandAP(self.vaz)
        benkhalfa_ON_OFF_L(1, self.vaz)
        t = benkhalfa_ldr()

        ss = son()
        pp = porte()
        ff = fetr()
        protction(pp ,ff , ss)
        protction1(1)

        vontilo(1 , self.vaa)

        if self.ll == 0 :
        setup()
        self.ll = self.ll + 1

        servodown(self.va)


        servoup(self.va)

        switch (i) {
        case 1:  monthString = self.list = t;
        break;
        case 2:  monthString = self.list = ss;
        break;
        case 3:  monthString = self.list = pp;
        break;
        case 4:  monthString = self.list = ff;
        i = 0;
        break;

        self.list = t



class ddd1(Thread):
        def __init__(self, mot):
        Thread.__init__(self)
        self.mot = mot
        self.listt = 0
        def run(self):

        messag = ' '


        while True:
        messag = str(theex1.list)
        c = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
        c.connect((self.mot , 2005))

        c.send(messag)
        c.close()
```

```
#print(h)
time.sleep(0.45)

i = 0
thread_1 = Afficheur()

thread_1.start()
theex = exemple()
theex.start()
theex1 = exemple()
theex1.start()
```

Programme Arduino

```
#include <Wire.h>
        #include <SimpleDHT.h>
        #include <Servo.h>
        Servo myservo; // create servo object to control a servo
        SimpleDHT22 e;

        #define SLAVE_ADDRESS 0x04
        int cxw;
//int fenétrr =7;
        int number = 0;
        int pos = 0;    // variable to store the servo position
        int lemp = 13;
        int son = 8;
        int portt = 4;
        int ldr = A0;
        int vontilo = 12;
        int vonttemp = 11;
        const byte interruptPin = 2 ;
volatile byte state = HIGH;
        byte t = 0;
        int ssoonn;
        int llompee ;
        byte soon = 0;
        byte port = 0;
        int j = 0;
        int g;
        int H;
        int sorv;
        int l;
        int protictionn =0;

        byte h = 0;
        float sensorVoltage;
        float sensorValue;
        int gaze ;
        int tompee ;
        int lompeee ;
        int prisontee  ;
        int vontoo;
        int poortt;
        int ssoonnm;
        int compp;
        int tesst;


        void setup() {
        // put your setup code here, to run once:
        myservo.attach(9);  // attaches the servo on pin 9 to the servo
object
        myservo.write(0);
        pinMode(lemp, OUTPUT);
        pinMode(vontilo, OUTPUT);
        pinMode(vonttemp, OUTPUT);
        pinMode(son, INPUT);
        pinMode(4, INPUT);
```

```
        pinMode(7, INPUT);
//pinMode(port, INPUT);

        pinMode(interruptPin, INPUT_PULLUP);
        attachInterrupt(digitalPinToInterrupt(interruptPin), blink,
CHANGE);
        Serial.begin(9600); // start serial for output
        digitalWrite(lemp, LOW);
// initialize i2c as slave
        Wire.begin(SLAVE_ADDRESS);

// define callbacks for i2c communication
        Wire.onReceive(receiveData);
        Wire.onRequest(sendData);

        gaze = 1;
        tompee = 10;
        lompeee = 25;
        prisontee = 30;
        poortt = 0;
        }

        void loop() {
        // put your main code here, to run repeatedly:
        if( number < 10 && number >= 0){
        tompee = number;
        }
        if( number < 20 && number >= 10){
        gaze = number;
        }
        if( number < 30 && number >= 20){
        lompeee = number;
        }
        if( number < 40 && number >= 30){
        prisontee = number;
        }
        if( number < 50 && number >= 40){
        sorv = number;
        }
        if( number < 60 && number >= 50){
        vontoo = number;
        }if( number < 70 && number >= 60){
        tesst = number;
        }
        Serial.println(number);
        int m = gaz();

        servo( sorv,  m , t);
        temp(tompee ,m , vontoo);
        int lo = asencron(lompeee);
        lompe(lo ,m);
        delay(100);
        llompee = LDR();
        poortt = pp();
        ssoonn = fen();
        ssoonnm = sonnnn();
        compar(ssoonn, poortt ,ssoonnm);
        compp = prisent(compar, tesst );

        j++;
        e.read(6, &t,  &h, NULL);
```

```
        switch(j){
        case 1:{
        cxw = t ;
        break;
        }
        case 2:{
        cxw = h + 100;
        break;
        }
        case 3:{
        cxw = sensorVoltage + 201;

        break;

        }
        case 4:{
        cxw = ssoonn ;
        break;
        }
        case 5:{
        cxw = poortt ;
        break;
        }
        case 6:{
        cxw = compp ;
        break;
        }

        case 7:{
        cxw =llompee ;
        j =0;
        break;

        }
        }

        }
        void receiveData(int byteCount){


        while(Wire.available()) {
        number = Wire.read();
        }
        }

////////////////////////////////////////////
        void sendData(){
        //cxw = h;


        Wire.write(cxw);



        }

//////////////////////////////

        void lompe(int i,int cmgaz){
        if(cmgaz < 2){
        if( i == 4){
```

```
        digitalWrite(lemp, HIGH);

        }else{

        digitalWrite(lemp, LOW);
        }
        }else{
        delay(20);
        }

        }
//////////////////////////
        int LDR(){
        float ft = analogRead(A0);
        ft = (ft/1024*5.0);
        Serial.println(ft);
        if( ft  >= 4.0 ){

        return(208);
        }else{
        return(209);
        }
        }
//////////////////////////
        int asencron(int k){

        if(k == 20){
        g = 0;
        H++;
        }
        if(k == 25){
        H = 0;
        g++;
        }
        if( H == 1){
        l = 4;
        return(l);
        }
        if(g == 1){
        l= 5;
        return(l);
        }

        return(l);
        }
//////////////////////////
        void blink() {
        if(l == 4){
        l = 5;

        }else{
        l = 4;

        }
        }

////////////////////////////////
        int gaz(){
        sensorValue = analogRead(A2);
        sensorVoltage = sensorValue/1024*5.0;
        return(sensorVoltage);
```

```
        }
////////////////////////////////////
        void vogaz(int com, int cmga){

        if( com != 10){
        com = 5;
        }
        if(cmga > 1.9 ){

        digitalWrite(vontilo, HIGH);
        }
        else if(com  == 5){

        digitalWrite(vontilo, HIGH);
        }
        else{

        digitalWrite(vontilo, LOW);
        }


        }


////////////////////////////////////
        void servo(int com, int comg ,int comt){
        if( comg > 1.9 ){
        for (pos  ; pos <= 45; pos += 1) { // goes from 0 degrees to 180
degrees
        // in steps of 1 degree
        myservo.write(pos);              // tell servo to go to position in
variable 'pos'
        delay(15);                       // waits 15ms for the servo to
reach the position
        }
        }
        else if( comt > 45){
        for (pos  ; pos <= 45; pos += 1) { // goes from 0 degrees to 180
degrees
        // in steps of 1 degree
        myservo.write(pos);              // tell servo to go to position in
variable 'pos'
        delay(15);                       // waits 15ms for the servo to
reach the position
        }
        }
        else if (com == 40){
        for (pos ; pos <= 45; pos += 1) { // goes from 0 degrees to 180
degrees
        // in steps of 1 degree
        myservo.write(pos);              // tell servo to go to position in
variable 'pos'
        delay(15);                       // waits 15ms for the servo to
reach the position
        }
        }
        else{
        for (pos ; pos = 0; pos -= 1) { // goes from 0 degrees to 180
degrees
        // in steps of 1 degree
        myservo.write(pos);              // tell servo to go to position in
```

```
variable 'pos'
        delay(15);                          // waits 15ms for the servo to
reach the position
        }
        }
        }
////////////////////////////////////
        void temp(int com,int comg, int comm){
        if(com != 1){
        com = 0;
        }
        e.read(6, &t,  &h, NULL);
        if(comg > 2 ){
        digitalWrite(vonttemp, HIGH);
        if(t > 30){
        if( comm == 50){
        digitalWrite(vonttemp, LOW);
        }else{
        digitalWrite(vonttemp, HIGH);
        }
        }
        else if( com ==1){
        digitalWrite(vonttemp, HIGH);


        }else{
        digitalWrite(vonttemp, LOW);
        }
        }else{
        delay(20);
        }
        }
//////////////////////////
//int prisent(){

        //if(y != 30){
        //  y = 35;
        // }
//  int vid = y;
//  if(vid == 30){

        // }
        ////////////////////////////////////
        int sonnnn(){
        if(digitalRead(son) == 1){
        return(216);
        }else{
        return(217);
        }
        }
        ////////////////////////////////////
        int pp(){
        if(digitalRead(4) == 1){
        return(210);
        }else{
        return(211);
        }
        }
        ////////////////////////////////////
        int fen(){
        if(digitalRead(7) == 1){
```

```
return 212;
}else{
return 213;
}
}

//////////////////////////////////////
int compar(int com, int com1 ,int com2){
if(com == 216 || com1 == 400 || com2 == 210){
return(5555);
}else{
return(5500);
}
}
int prisent(int com, int com1 ){

if(com != 30){
com = 35;
}
if(com == 30){
if( com == 5555 ){
int f = 206;
return(f);
}else{
int g = 208;
return(g);
}
}else{
int b = 208;
return(b);
}

}
```