République Algérienne Démocratique Populaire
Ministère de l'Enseignement Supérieur et de la Recherches Scientifique
UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTE DU GENIE ELECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire

De fin d'études

En vue de l'obtention du diplôme de Master académique en Informatique
Spécialité : Réseaux, Mobilité, Systèmes Embarqués.

Thème :

Conception et réalisation d'un système d'alarme anti-intrusion par barrière infrarouge

Dirigé par :

M. HEMDANI Chabane

Réalisé par :

M^elle^. AOUES Nouzha

M^elle^. AMAROUCHENE Lynda

Promotion :

2016/2017

Remerciements :

En premier lieu nous remercions Dieu le tout puissant de nous avoir donner le courage, la santé et la volonté pour réaliser ce mémoire.

On ne remerciera jamais assez nos parents, sans lesquels, rien n'aurait été possible. Leur amour et leur dévotion, ont fait de nous ce que nous sommes.

Nous tenons à adresser nos plus sincères remerciements à notre promoteur ***M. HEMDANI*** *pour son aide, sa constante disponibilité et ses précieux conseils qui ont permis à ce travail de voir le jour.*

Nos remerciements s'adressent à tous les membres du jury pour l'honneur qu'ils nous ont fait en acceptant de juger notre travail.

Nos sentiments de gratitude vont à nos professeurs qui tout au long de notre cursus nous ont transmis leur savoir sans réserve.

Nos remerciements vont aussi à tous ceux et celles qui ont participé de près ou de loin à l'élaboration du présent travail

LYNDA & NOUZHA

Dédicaces :

Je dédie ce modeste travail en signe de respect, d'amour et de reconnaissance à :

Mes très chers parents qui m'ont toujours soutenu et à qui je dois tout, que Dieu vous protège et m'aide à vous rendre une miette de ce que vous avez fait pour moi.

Ma très chère sœurette que j'adore ***GHENIMA.***

Mes deux chers frères ***YOUCEF & MISSIPSA.***

La meilleure cousine au monde que je considère comme ma deuxième sœur ***CHABHA.***

Ma chère tante ***NACIMA*** *que j'aime beaucoup.*

Ma chère copine ***SAIDA*** *qui était toujours là pour moi.*

A tous mes oncles et tantes maternels et paternels.

A tous mes cousins et cousines.

A toute la promotion RMSE 2016/2017.

A ma chère amie ***NOUZHA*** *et toute sa famille.*

A tous ceux qui m'aime.

LYNDA ♥

Dédicaces :

Avec un énorme plaisir, un cœur ouvert que je dédie ce modeste travail :

A celle qui m'a donné la vie, qui s'est sacrifié pour mon bonheur et ma réussite, à ma cher maman **LEILA**.

A mon père **MEZIANE** qui a été mon ombre durant toutes les années de mes études, qui a veillé à me donner l'aide, à m'encourager et me protéger, que Dieu les gardes et les protèges.

A mes grands parents que j'aime beaucoup.

A ma cher sœur **MELISSA** et mon frère **BOUSSAD** que j'aime très fort.

A touts mes oncles et mes tantes.

A tous mes cousins et cousines.

A mon petit ange **MAELLE**. Je T'aime ma Princesse.

A ma chère amie **Celia** qui est comme une sœur pour moi.

A toute la promotion RMSE 2016/2017.

A tous mes Amis(es).

A ma meilleure amie **LYNDA** et toute sa famille.

A tous ceux que j'aime.

NOUZHA ❤

Sommaire :

Chapitre II : La Présentation de l'Arduino

Chapitre III : Conception

Chapitre IV : Réalisation

Liste des figures :

Liste des tables :

Introduction Générale

Introduction générale :

La sécurité est devenue une préoccupation majeure, d'autant plus que la plupart des pays développés ont enregistré une hausse significative des cambriolages de maison dans les dernières décennies. Renforcer la sécurité d'une maison est la première étape dans la prévention des cambriolages.

Le système d'alarme constitue un excellent moyen pour protéger une habitation.Le système de base se compose d'un panneau de commande et de détecteurs placés autour de l'habitation. Les détecteurs déclenchent l'alarme, soit par détection de mouvement, soit par ouverture d'une porte ou une fenêtre. Une fois déclenché, le système d'alarme active une sirène et envoie une alerte de sécurité vers des téléphones ou un centre de surveillance. Le simple fait de mettre une sirène sur la façade d'une habitation dissuade largement la plupart des cambrioleurs.

L'objectif de notre projet est de concevoir et réaliser un système d'alarme anti-intrusion par barrière infrarouge, en utilisant la technologie Arduino. Ce système permet de surveiller un local ou une habitation contre toute intrusion en se servant de la barrière IR, et d'alerter d'une intrusion en temps réel par alerte sonore (au moyen d'un Buzzer) et l'envoie d'un SMS (au moyen du module GSM).

Pour ce faire, nous avons divisé notre travail en quatre chapitres :

Dans le premier chapitre, nous allons présenter la domotique et les systèmes embarqués.

Dans le deuxième chapitre, nous présenterons la technologie Arduino, la description de la carte que nous allons utiliser (Arduino UNO), ainsi que l'environnement de développement Arduino.

Dans le troisième chapitre, nous présenterons l'approche que nous avons adoptée pour la conception de notre système embarqué.

Dans le quatrième chapitre, nous présenterons l'ensemble des composants matériels que nous avons utilisé, l'implémentation de notre application, à la fin nous allons effectuer une série de tests afin d'observer le comportement de notre système.

Chapitre I :
La Domotique
et
Les Systèmes Embarqués

I- La Domotique :

Introduction:

Depuis quelques années, les besoins de l'homme ne cessent d'augmenter face aux offres technologiques disponibles sur le marché. Confort, économie, sécurité et autonomie sont les aspects recherchés.

De nombreuses solutions ont été proposées depuis quelques décennies, pour une meilleure domination. La discipline chargée de ce domaine s'appelle « la domotique ».

I.1.Historique:[1]

Les premières applications de domotique sont apparues au début des années 1980, elles sont nées de la miniaturisation des systèmes électroniques et informatique. Durant ces années, la maison « motorisée » fait son apparition : stores, portails et portes de garages deviennent motorisés.

Dès les années 90, on parle de maison « automatisée », par le biais de planification de certaines actions (ouverture simultanée de tous les volets, extinction de lumières ...) la maison apporte confort supplémentaire et simplification des tâches quotidiennes.

Depuis les années 2000 à nos jours, on parle de maison « pilotable à distance », avec le développement d'internet, les automatismes sont désormais programmés et commandés via un appareil mobile (Smartphone, tablette, ordinateur).

I.2. Définition:[2]

La domotique (du latin "domus", c'est-à-dire maison) est la rencontre entre technologie informatique, électrotechnique et électronique qui transforment une maison en une maison "intelligente". C'est l'outil qui permet de gérer les installations, les appareils, les **automatismes** afin d'augmenter le confort de vie, à l'intérieur de sa propre résidence. L'immeuble commence à développer une intelligence à part, qui est caractérisée pas par la quantité de technologie qu'il contient, mais par le projet de l'**intégration** des technologies dans les demandes toujours en évolutions des individus.

Le mot domotique est rentré désormais dans le dictionnaire comme " Science qui s'occupe de l'application de l'électronique et de l'informatique dans la vie domestique (appareils ménagers et systèmes de contrôle) et de leur utilisation".

La **Figure I.1** représente les items relatifs à l'habitat intelligent et la relation entre eux et avec l'usager.

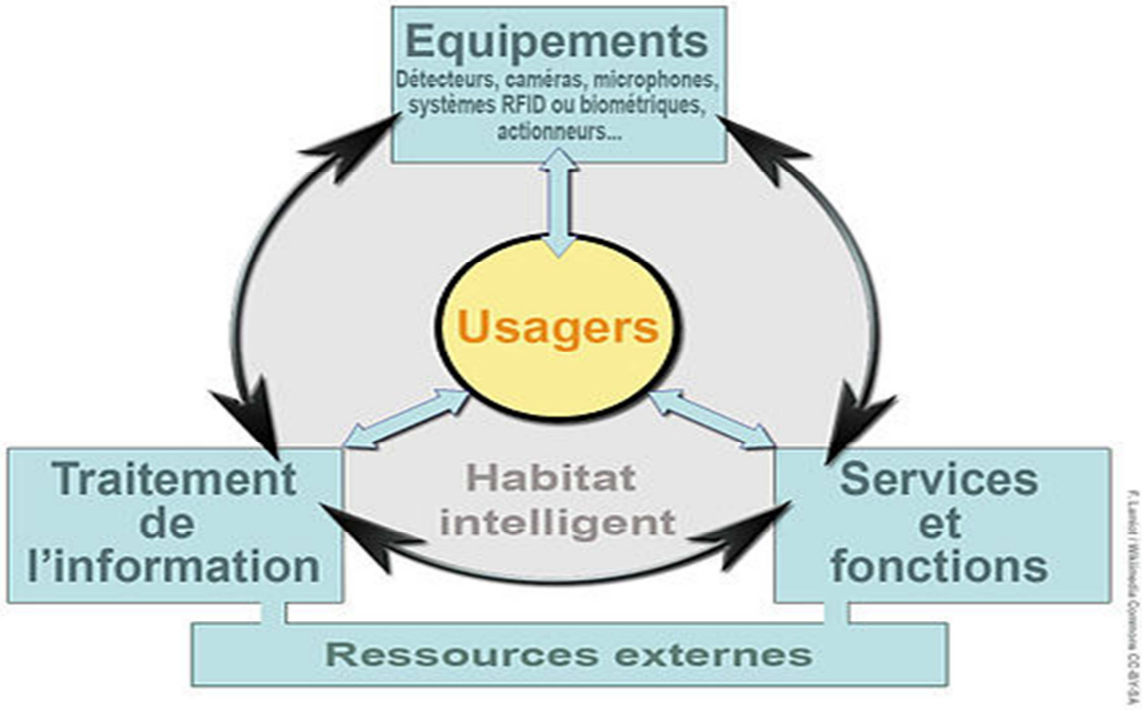


Figure I.1. Représentation schématique d'items relatifs à l'« habitat intelligent ».

I.3. **But de la domotique:**

Dans chaque établissement nous pouvons observer des gestes répétitifs ou des situations similaires qui peuvent être automatisés. Par exemple: allumer un escalier sombre et éteindre la lumière quand il n'y a personne, allumer et éteindre l'éclairage quand la maison est vide pour simuler la présence de quelqu'un, baisser automatiquement la climatisation quand la température extérieure baisse aussi, gérer les cameras ou l'alarme etc. Dans chaque secteur la domotique améliore le confort de vie et de sécurité de l'utilisateur final, avec des résultats jamais eu avant.

I.4. **Domaine d'application de la domotique :**

La domotique présente un panel de services domestiques, qu'on peut regrouper en quatre principaux domaines (confort, sécurité, économie d'énergie, communication et multimédia).

I.4.1. **Confort et simplicité :**

L'accroissement du niveau de confort des habitations a été le premier objectif de la domotique. Les fonctions de commande à distance simples qui agissent sur différents types d'appareils sont maintenant banalisées.

En effet, il est possible d'activer à distance des fonctions qui ont pour but de recréer une ambiance ou un état prédéfinis dans la maison. Il est donc facile d'imaginer un nombre illimité de fonctions qui pourraient faciliter le confort quotidien dans la maison (par exemple la cafetière s'allume et les volets s'ouvrent à 7h tous les matins).

I.4.2. **Sécurité:**

En termes de sécurité, la domotique permet entre autres de :

- Prévenir les risques provenant de l'extérieur (intrusion, cambriolage …) comme ceux provenant de l'intérieur (accidents domestiques).
- Surveiller à distance notre habitation depuis un Smartphone, une tablette ou un ordinateur.
- Etre averti d'un incident (alarmes techniques) par SMS et/ou par E-mail.

I.4.3. **Economie et performance énergétique:**

L'augmentation des couts de l'énergie aussi bien que l'émergence des préoccupations écologiques sont des enjeux importants de notre société actuelle. La domotique propose ainsi de réduire les consommations énergétiques des maisons en adaptant des consommations aux modes de vie des occupants et à l'environnement extérieur. Cela comprend la régulation de l'éclairage et du chauffage, le traitement de l'air, l'optimisation des ouvrants, la programmation horaire, les commandes à distance, les interrupteurs automatiques pour l'éclairage d'un escalier ou d'un couloir, l'ouverture ou la fermeture d'un volet selon l'ensoleillement …

I.4.4. **Communication et multimédia :**

Il est possible de gérer et diffuser ses bibliothèques de musiques et de vidéos dans différentes pièces, de sauvegarder ses données informatiques, d'avoir accès à distance à ses ordinateurs, de faciliter la mobilité et le télétravail. Ces systèmes sont en général indépendants et peuvent être pilotés par les fonctions domotiques.

La **Figure I.2**présente les différents domaines d'application de la domotique.

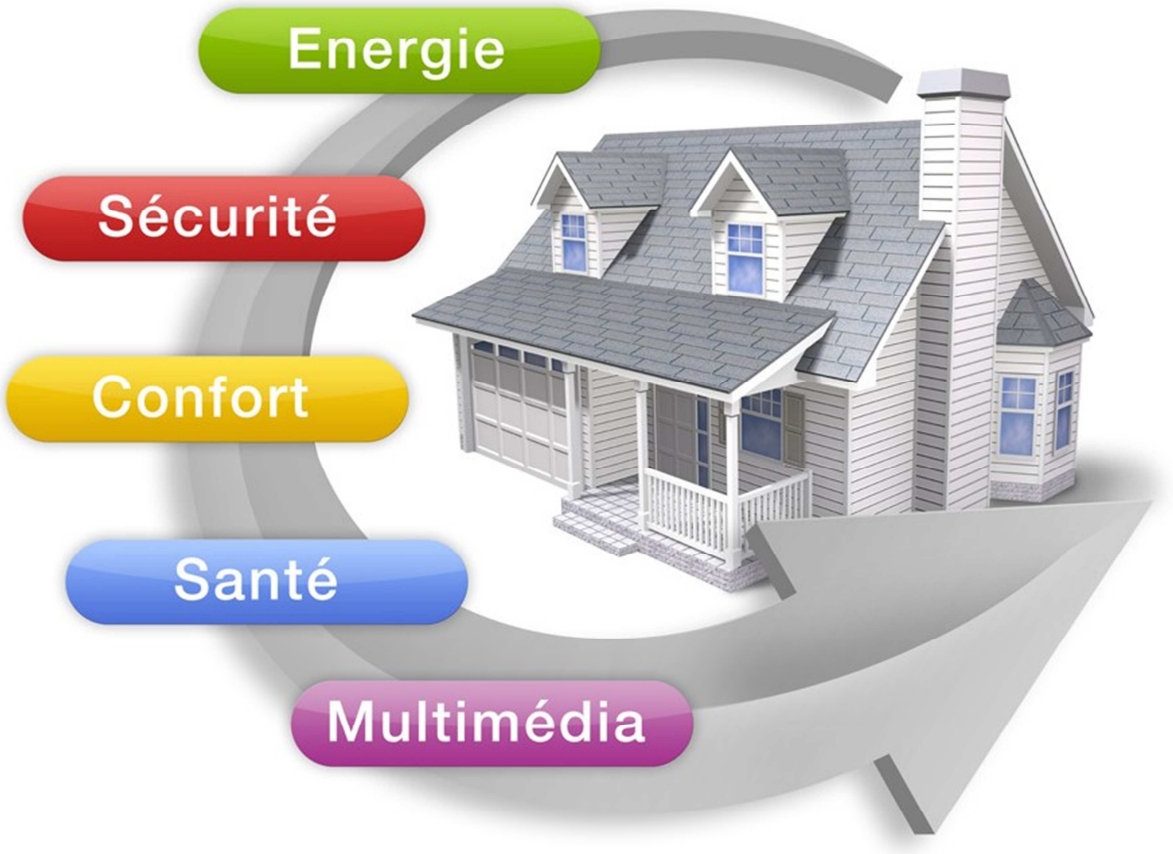


Figure I.2. Domaines d'application de la domotique.

I.5. Les composants principaux de la domotique :

Le fonctionnement de la domotique repose sur la communication entre plusieurs appareils électriques. Pour assurer cette communication et obtenir les effets souhaités, une maison domotique doit comporter plusieurs éléments.

I.5.1. Les automatismes :

Les automatismes ont pour objectif d'éviter d'effectuer des gestes qui pourraient être automatisés. Il peut s'agir par exemple d'appuyer sur un bouton pour allumer une lumière, ou bien d'éteindre la télévision après une heure de veille. Le fonctionnement des automatismes est directement lié aux capteurs, au système de programmation, et à la centrale domotique.

I.5.2. Le système de programmation domotique:

Un système de programmation est indispensable pour profiter des avantages d'une installation domotique. C'est grâce à lui qu'on peut choisir comment programmer les appareils électriques, en fonction de nos habitudes et de nos besoins.

I.5.3. **Les appareils domotiques :**

Il existe beaucoup d'appareils domotiques, puisqu'en fait tous les appareils électriques peuvent être domotiques s'ils sont intégrés au fonctionnement de l'installation. Les lampes, les écrans, les machines à laver, les radiateurs... autant d'appareils qui peuvent être contrôlés par la domotique. Des appareils spéciaux ont été développés pour répondre aux besoins d'automatisation. On pense notamment à l'aspirateur domotique et à la tondeuse domotique, ou encore aux installations HiFi comme les home cinema.

I.5.4. **Les capteurs domotiques:**

Les capteurs sont indispensables pour adapter le fonctionnement d'un objet domotique selon un critère défini. Par exemple, il existe des capteurs qui détectent le mouvement, et donc la présence humaine. D'autres détectent la luminosité, d'autres la température...

Ces capteurs convertissent une donnée en un signal. Ainsi, on peut par exemple régler un appareil électrique pour qu'il ne fonctionne que lorsque le capteur qui lui est associé constate un mouvement.

La **Figure I.3** présente un capteur à infrarouge passif.

Figure I.3. Capteur Domotique.

I.5.5. **Le réseau domotique:**

Pour transformer un logement en maison intelligente, il est nécessaire d'installer un réseau pour que les appareils électriques, les capteurs et le système de programmation puissent communiquer entre eux.

Il existe plusieurs types de réseau domotique :

- Le câblage domotique : chaque appareil est relié à la centrale domotique via un câble (souvent Ethernet RJ45).
- La domotique par courant porteur CPL (Courant Porteur en Ligne) : le réseau électrique fait circuler les données à travers les câbles électriques existants.
- La domotique sans fil : plus pratique d'installation et d'utilisation, le réseau s'appuie sur l'échange d'informations par ondes radio ou par infrarouge.

I.5.6. **L'interface domotique:**

L'interface domotique permet de paramétrer en temps réel les réglages de fonctionnement de vos appareils électriques. En fonction de vos habitudes et de votre rythme de vie, vous pouvez choisir une interface de gestion différente. Voici les principales interfaces possibles :

- Une télécommande domotique;
- Un écran de contrôle tactile;
- Un ordinateur ou une tablette;
- Un smartphone;
- Directement sur Internet.

I.5.7. **La centrale domotique :**

Tous les équipements ci-dessus doivent être connectés entre eux pour pouvoir communiquer. C'est pour cela qu'une centrale domotique est indispensable au fonctionnement d'un équipement domotique. Il existe des centrales domotiques pour chaque type d'installation : câblage domotique, domotique par courant porteur, et domotique sans fil.

La Figure I.4 représente une centrale domotique.

Figure I.4. Centrale Domotique.

I.6. **Fonctionnement de la domotique:**

Contrairement à une installation électrique classique, un système domotique repose sur un principe simple : la séparation des circuits de commande et de puissance. Le circuit de puissance distribue l'énergie. Le circuit de commande véhicule les infos de commande transmises par bouton poussoir, télécommande ou dalle tactile.

Une simple programmation au niveau du tableau électrique suffit alors pour établir les liens entre les organes de commandes et les récepteurs d'ordre, et ainsi configurer l'ensemble de l'installation électrique.

I.7. **Technologie de la domotique:**[3]

Pour l'intégration, à l'habitat, du système domotique, on fait appel à l'une des trois principales technologies, qui sont :

I.7.1. **Le Courant Porteur de Ligne(CPL) :**

Cette technologie s'installe dans le cadre de l'intégration d'un système domotique et permet de piloter les fonctions éclairages, volets roulants électriques, chauffage, alarme, détection technique et portier.

Ceci est possible grâce à l'émission, par les appareils de commande, d'ordres codés qui sont transmis à toute l'installation en utilisant le réseau électrique existant. Les récepteurs décodent ces ordres et effectuent la commande demandée. Il est ainsi possible de multiplier les points de commande n'importe où dans la maison avec un simple raccordement phase/neutre. L'affectation des commandes est indépendante du câblage, elle se fait par apprentissage. Cette technologie s'adapte aussi bien à la construction neuve qu'à la rénovation lourde.

La **Figure I.5** représente la technologie CPL.

Figure I.5. La Technologie CPL.

I.7.2. **La technologie Radio/Zigbee:**

Les appareils émetteurs Radio/ZigBee communiquent avec les contrôleurs BUS/SCS à travers une interface réceptrice connectée au bus. Ils s'installent sans aucun câblage pour permettre une extension de l'installation sans dégâts sur les murs. Le retour d'état est possible grâce à la communication bidirectionnelle entre les appareils.

I.7.3. **La technologie BUS/SCS :**

La technologie BUS/SCS utilise un câble 2 fils qui servent à la fois pour l'alimentation des interfaces de commandes (27 V non polarisé) et pour la transmission des ordres entre les points de commande et les contrôleurs installés dans le tableau électrique. Les points de commande sont tous raccordés au bus et n'utilisent pas le câblage traditionnel. Les contrôleurs sont raccordés d'une part au bus et d'autre part au réseau et aux charges à commander. La configuration du système se fait par le positionnement de cavaliers directement sur les appareils et/ou bien grâce à un logiciel. Elle permet d'affecter les commandes aux contrôleurs indépendamment du câblage et de créer ainsi des commandes individuelles, des commandes par zone et des commandes générales. Les systèmes domotiques utilisant la technologie BUS/SCS sont particulièrement adaptés aux constructions neuves lorsque les installations requièrent de nombreux points de commande (30 et plus).

La **Figure I.6** représente la technologie BUS/SCS.

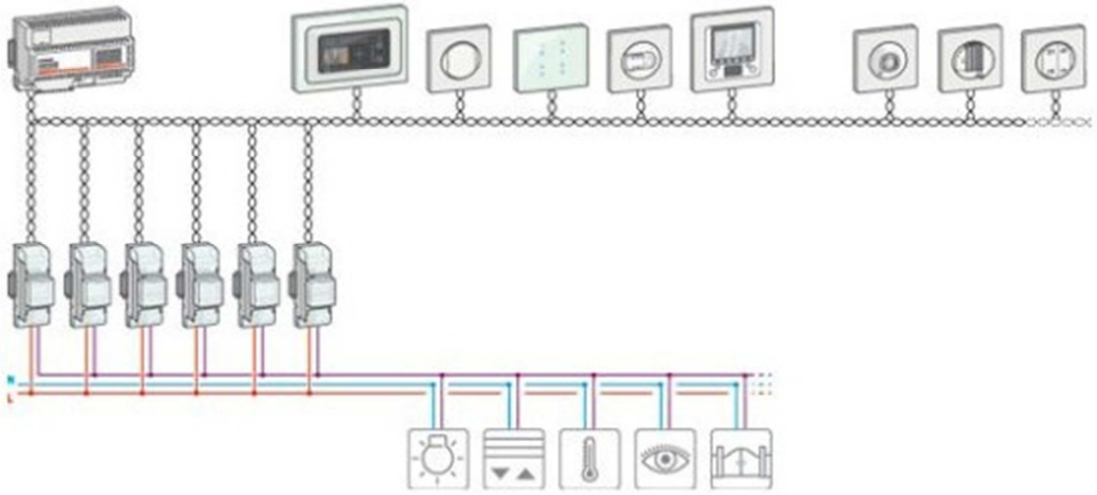


Figure I.6. Technologie BUS/SCS.

II-**Les systèmes embarqués:**

La technologie domotique est un ensemble des systèmes embarqués qui sont intégrés à des équipements afin de rendre une maison intelligente. Dans cette partie nous allons présenter les systèmes embarqués, leurs domaines d'application, et décrire l'interaction de ces systèmes avec leurs environnements.

II.1. **Définition:**[4]

Un système embarqué est un système électronique et informatique autonome ne possédant pas des entrées sorties standard comme un clavier ou un écran d'ordinateur. On le définit aussi généralement par le fait qu'il n'est pas visible en tant que tel, mais est intégré dans un équipement doté d'une autre fonction ; on dit aussi que le système est enfoui, ce qui traduit plus fidèlement le terme anglais embedded.

II.2. **Historique:** [4]

Les premiers systèmes embarqués sont apparus en 1971 avec l'apparition d'Intel 4004. L'Intel 4004 développé en 1971, le premier microprocesseur, était le premier circuit intégré incorporant tous les éléments d'un ordinateur dans un seul boîtier: unité de calcul, mémoire, contrôle des entrées / sorties. Alors qu'il fallait auparavant plusieurs circuits intégrés différents, chacun dédié à une tâche particulière, un seul microprocesseur pouvait assurer autant de travaux différents que possible. Très rapidement, des objets quotidiens tels que fours à micro-ondes, télévisions et automobiles à moteur à injection électronique ne

tardèrent pas à être équipés de microprocesseurs. Ce sont alors les débuts de l'informatique embarquée.

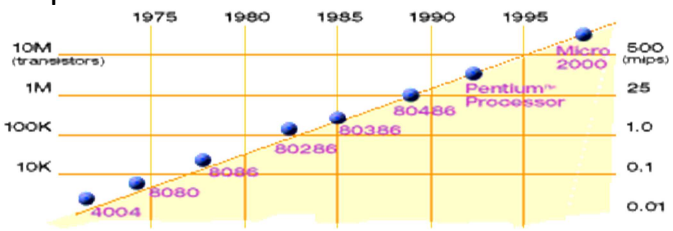


Figure I.7. Histoire des micro-processeurs.

II.3. Les contraintes des systèmes embarqués:

Les systèmes embarqués exécutent des tâches prédéfinies et ont un cahier des charges contraignant à remplir, qui peut être d'ordre :

- D'espace compté, ayant un espace mémoire limité de l'ordre de quelques Ko (bien que la taille vienne à être de moins en moins limitée grâce à la miniaturisation des éléments). Il convient de concevoir des systèmes embarqués qui répondent aux besoins au plus juste pour éviter un surcoût ;
- De puissance de calcul. Il convient d'avoir la puissance de calcul juste nécessaire pour répondre aux besoins et aux contraintes temporelles de la tâche prédéfinie. Ceci en vue d'éviter un surcoût de l'appareil et une consommation excédentaire d'énergie (courant électrique) ;
- D'autonomie. La consommation énergétique doit être la plus faible possible, due à l'utilisation de batteries et/ou, de panneaux solaires voire de pile à combustible pour certains prototypes ;
- Temporel, dont les temps d'exécution et l'échéance temporelle d'une tâche sont déterminés (les délais sont connus ou bornés *a priori*). Cette dernière contrainte fait que généralement de tels systèmes ont des propriétés temps réel ;
- De sûreté de fonctionnement. Car s'il arrive que certains de ces systèmes embarqués subissent une défaillance, ils mettent des vies humaines en danger ou mettent en périls des investissements importants. Ils sont alors dits « critiques » et ne doivent jamais faillir. Par « jamais faillir », il faut comprendre toujours donner des résultats justes, pertinents et ce dans les délais attendus par les utilisateurs (machines et/ou humains) des dits résultats.

II.4. Les caractéristiques des systèmes embarqués:

Un système embarqué est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Le terme désigne aussi bien le matériel informatique que le logiciel utilisé. Ses ressources sont généralement limitées.

Du fait de la nécessité d'une architecture physique confinée, la machine et le logiciel sont intimement liées et ne sont pas aussi discernables que dans un environnement classique de type PC. Par ailleurs, la conception de ces systèmes est généralement fiable (avions, système de freinage ABS) à cause de leurs utilisations dans des domaines à fortes contraintes mais également par ce que l'accès au logiciel est souvent difficile une fois le système fabriqué. Enfin, il y a une faible barrière entre les systèmes embarqués et les systèmes temps réels. Cependant un logiciel embarqué n'a pas forcément de contraintes temps réel.

La table suivante montre la comparaison des systèmes embarqués aux systèmes informatiques standards :

Système Informatique Standard :	**Système Embarqué :**
▪ Processeur standard • Multiples unités fonctionnelles (flottant) • Vitesse élevée (> GHz) • Consommation électrique élevée • Chaleur • Taille ▪ MMU (mémoire virtuelle) ▪ OS ▪ Cache ▪ Grand nombre de périphériques	▪ Microcontrôleur • Architecture adaptée ; • Vitesse faible (~200 MHz) ; • Mémoire limitée ; • Basse consommation ; • Petite taille => faible coût. ▪ Eventuellement processeur DSP (traitements de signal). ▪ Au mieux quelques Mo de mémoire. ▪ Eventuellement RTOS.

Table I.1 : Comparaison des systèmes embarqués avec les systèmes informatique standards.

II.5. Classification des systèmes embarqués:

II.5.1. Système Transformationnel :

Activité de calcul, qui lit ses données et ses entrées lors de son démarrage, qui fournit ses sorties, puis meurt.

II.5.2. **Système Interactif :**

Système en interaction quasi permanente avec son environnement, y compris après l'initialisation du système; la réaction du système est déterminée par les événements reçus et par l'état courant (fonction des événements et des réactions passés); le rythme de l'interaction est déterminé par le système et non par l'environnement.

II.5.3. **Système Réactif ou Temps Réel :**

Système en interaction permanente avec son environnement, y compris après l'initialisation du système; la réaction du système est déterminée par les événements reçus et par l'état courant (fonction des événements et des réactions passées); mais le rythme de l'interaction est déterminé par l'environnement et non par le système.

II.6. **Architecture d'un système embarqué:**

L'architecture d'un système embarqué est illustrée par le schéma suivant:

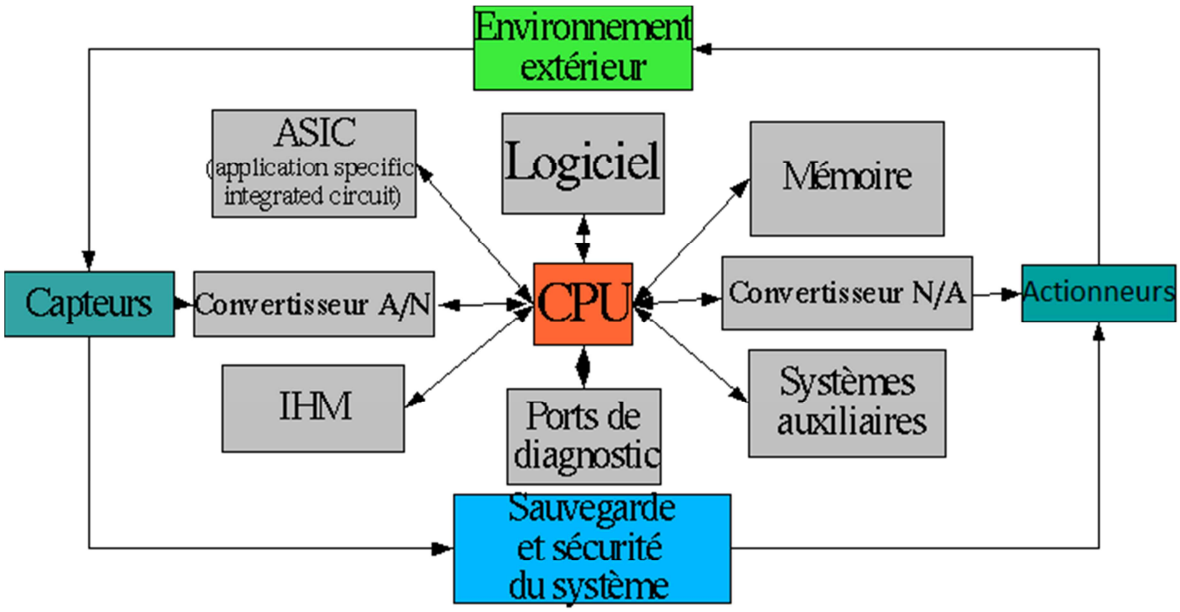


Figure I.8. Architecture d'un système embarqué.

Cette architecture peut varier selon les systèmes: on peut par exemple, ne pas trouver de systèmes auxiliaires dans de nombreux systèmes embarqués autonomes et indépendants. En revanche, l'architecture de base est la plupart du temps composée d'une unité centrale de traitement (CPU), d'un système d'exploitation qui réside parfois uniquement en un logiciel spécifique (ex: routeur), ou une boucle d'exécution (ex: ABS). De même l'interface IHM n'est

pas souvent existante, mais est souvent utile pour reconfigurer le système ou vérifier son comportement.

Le fonctionnement du système se résume ainsi:

- Il reçoit des informations de l'environnement extérieur qu'il converti en signal numérique.
- L'unité de traitement composée du CPU, de la mémoire, du logiciel, de l'ASIC et éventuellement de systèmes externes traite l'information.
- Le traitement génère éventuellement un signal qui est envoyée vers la sortie, les systèmes auxiliaires, les ports de monitoring ou l'IHM.

Conclusion :

Dans ce chapitre, nous avons fait une étude sur « la domotique », ces domaines d'utilisations ainsi ces composants. On a également présenté « les systèmes embarqués », leurs domaines d'application et leurs caractéristiques.

Dans le deuxième chapitre nous allons introduire la technologie Arduino, notamment la carte Arduino UNO puisqu'il c'est cette dernière qu'on utilisera tout au long de notre projet.

Chapitre II : Présentation de l'Arduino

Introduction:

Un système Arduino donne la possibilité d'allier les performances de la programmation à celles de l'électronique. Plus précisément, pour programmer des systèmes électroniques. Le gros avantage de l'électronique programmée c'est qu'elle simplifie grandement les schémas électroniques et par conséquent, le coût de la réalisation, mais aussi la charge de travail à la conception d'une carte électronique.

Un système Arduino est composé de deux choses principales : **le matériel** et **le logiciel.**

I- Présentation de l'Arduino:[5]

Une équipe de développeurs composée de Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis et Nicholas Zambetti a imaginé un projet répondant au doux nom d'**Arduino** et mettant en œuvre une petite carte électronique programmable et un logiciel multiplateforme, qui puisse être accessible à tout un chacun dans le but de créer facilement des systèmes électroniques.

Il s'agit d'une carte électronique basée autour d'un microcontrôleur Atmega du fabricant Atmel, dont le prix est relativement bas pour l'étendue possible des applications.

L'intérêt principal des cartes ARDUINO est leur facilité de mise en œuvre. ARDUINO fournit un environnement de développement s'appuyant sur des outils open source (ou code source ouvert), c'est-à-dire que le code source est distribué sous une licence permettant à quiconque de lire, modifier ou redistribuer ce logiciel.

II- Les différentes types de carte Arduino:[6]

Il existe deux types de cartes :

- Les cartes Arduino officielles (ou classique), compatible hardware et software avec le facteur de forme (la taille, la forme et d'autres spécifications physiques des composants)et l'ide Arduino.
- Les cartes dérivées d'Arduino, compatible avec les Shields Arduino classique (mais pas avec l'ide Arduino de base).

Il existe plusieurs cartes Arduino, on commence par décrire la carte Arduino Uno puisque c'est celle qu'on utilisera tout au long de notre travail.

III- La carte Arduino Uno:[7]

Le modèle UNO de la société ARDUINO est une carte électronique dont le cœur est un microcontrôleur ATMEL de référence ATMega328. Le microcontrôleur ATMega328 est un microcontrôleur 8bits de la famille AVR dont la programmation peut être réalisée en langage C.

La **Figure II.1** présente une carte Arduino UNO et montre ses principaux composants.

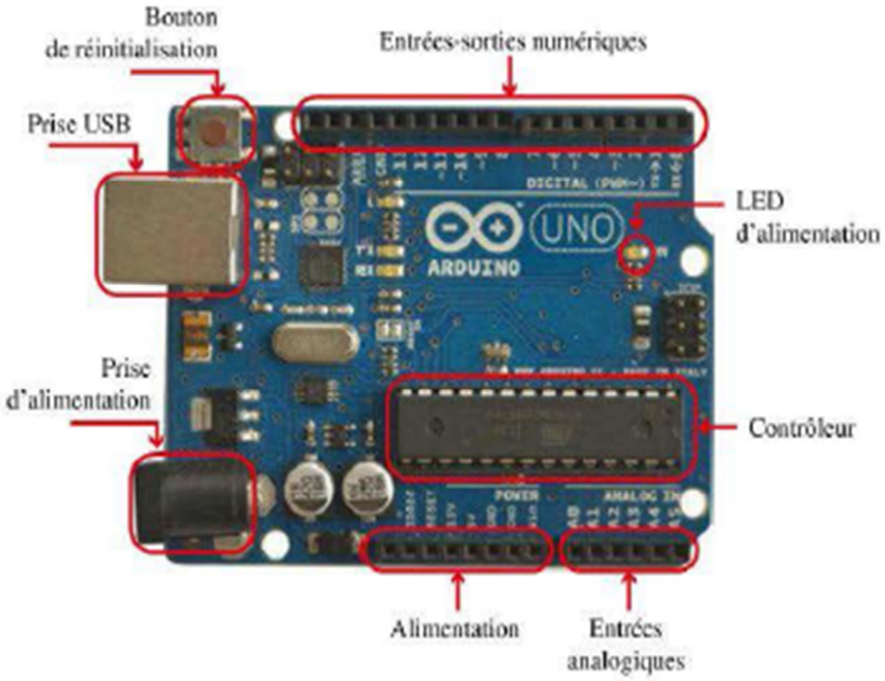


Figure II.1. Carte Arduino UNO.

Ces composants seront décrits en détail dans les prochaines sections.

III.1.Caractéristiques de la carte Arduino UNO:[7]

Le tableau suivant résume les caractéristiques essentielles de la carte Arduino UNO :

Catégorie	Valeur
Microcontrôleur	Atmega328
Fréquence d'horloge	16 Mhz
Tension de service	5 V
Tension d'entrée (Recommandée)	7-12 V
Tension d'entrée (Limites)	6-20 V

Ports Numériques	14 entrées/sorties (6 PWM)
Ports Analogiques	6 entrées analogiques
Courant maxi. Par broche d'E/S(c.c)	40 mA
Courant maxi. Par broche 3,3 V	50 mA
Mémoire	32 Ko Flash, 2 Ko SRAM, 1 Ko EEPROM
Dimensions	6,86 cm x 5,3 cm

Table II.1. Caractéristiques de la carte Arduino UNO.

III.2. **Le microcontrôleur ATmega328:**

Le microcontrôleur ATmega328 est un circuit intégré qui traite les informations qu'il reçoit et déclenche des actions suivant le programme qu'il a reçu. Il est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur :

- **Le processeur:**

C'est le composant principal du microcontrôleur. C'est lui qui va exécuter le programme qu'on lui donnera à traiter. On le nomme souvent le CPU.

- **La mémoire:**

Il en possède 3 types :

– La mémoire Flash: C'est celle qui contiendra le programme àexécuter.Cette mémoire est effaçable et réinscriptible. Sa capacité est de 32 Ko.

– RAM : c'est la mémoire dite "vive", elle va contenir les variables du programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur. Sa capacité est de 2 ko.

– EEPROM : C'est le disque dur du microcontrôleur. On peut y enregistrer des infos qui ont besoin de survivre dans le temps, même si la carte doit êtrearrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme. Sa capacité est de 1 Ko.

- **Horloge** :

C'est un élément essentiel au fonctionnement du microcontrôleur, c'est un signal rectangulaire qui cadence toute sa circuiterie interne. Sa fréquence est de 16 MHz.

- **Bus d'adresses** :

Le bus d'adresse permet l'accès aux différentes cases mémoires.

- **Bus de données :**

Le bus de données sert à transporter les données d'un bloc à un autre (de la mémoire vers le processeur ou vice versa). Sa taille est de 8 bits.

- **Bus de contrôle :**

Le bus de contrôle permet de se positionner en lecture ou en écriture sur ces différentes cases mémoire.

- **Ports d'Entrées/Sorties :**

Les ports d'E/S sont les connexions qui relient le microcontrôleur au monde extérieur. Ils constituent l'interface à laquelle la périphérie peut être connectée. Ces ports seront décrits en détail dans les prochaines sections.

- **Contrôleur d'interruption :[4]**

Le contrôleur d'interruption sert à effectuer une surveillance par interruption, l'unité centrale suit le cours normal de son programme et ne réagit que si une interruption se produit. Le travail de fond s'interrompt un court instant et bascule dans une routine d'interruption (ISR, ou Interrupt Service Routine). Celle-ci contient des instructions qui indiquent l'action à effectuer. Après cela on revient au travail de fond et à l'endroit précis ou l'interruption s'est produite, comme si rien ne c'était passé.

Le microcontrôleurATMega328 peut gérer 2 **interruptions externes sur ses broches INT0 et INT1**, respectivement broches 2 et 3. Il peut également gérer des interruptions de changement d'état sur 20 de ses broches. Une librairie Arduino a été développée afin de permettre l'utilisation de ces interruptions « arduino-pinchangeint ».

Les interruptions peuvent être déclenchées selon 4 modes :

LOW : la broche est à un état bas.

RISING : la broche passe d'un état bas à haut.

FALLING : la broche passe d'un état haut à bas.

CHANGE : la broche change d'état.

- **Timer:**

Un Timer est un registre à l'intérieur du microcontrôleur qui s'incrémente (ou se décrémente) chaque fois qu'il reçoit une impulsion d'un signal d'horloge. Ce signal d'horloge peut être propre au microcontrôleur ou bien extérieur à celui-ci. C'est comme une horloge, et peut être utilisé pour mesurer des

événements temporels. Le Timer peut être programmépar certains registres spéciaux. Le module Arduino UNO est construit autour du microcontrôleur AVR ATmega328d'Atmel qui possède 3 timers :

- Le timer0 : sur 8 bits, utilisé par les fonctions delay(), millis() et micros(). Il commande également des PWM(Pulse Width Modulation ou Modulation par Largeur d'Impulsion) sur les broches 5 et 6.
- Le timer1: sur 16 bits, qui compte de 0 à 65535 (0 à FFFF en hexadécimal) et qui est utilisé par la bibliothèque Servo ou bien pour de la PWM sur les broches 9 et 10.
- Le timer2 : sur 8 bits, qui est utilisé par la fonction Tone() ou bien pour de la PWM sur les broches 3 et 11.

La **Figure II.2** présente un schéma simplifié de la structure interne du microcontrôleur ATmega328.

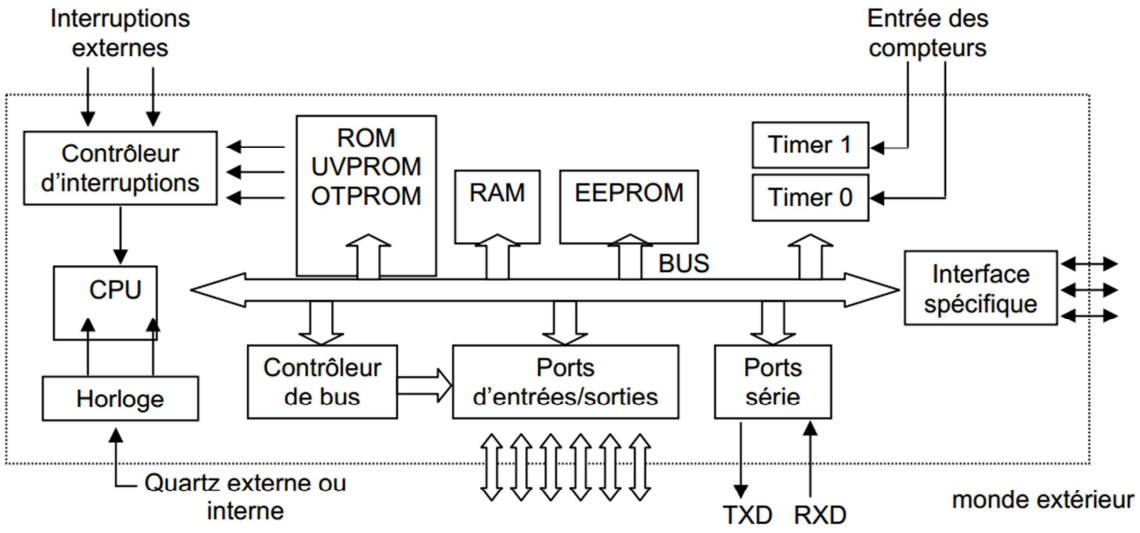


Figure II.2. Schéma simplifié du contenu du microcontrôleur ATmega328.

III.3. L'Alimentation :[6]

Le microcontrôleur fonctionne sous 5V, pour fonctionner la carte a besoin d'alimentation, qui se fait soit via le port USB soit via une alimentation externe. La source d'alimentation est sélectionnée automatiquement.

La carte peut être alimentée via le port USB qui fournit 5V, comme elle peut fonctionner avec une alimentation externe entre 7 et 12 V, elle possède un régulateur de tension interne qui se charge de la réduire. Une tension inférieure à 7V pourrait rendre la carte instable et cela parce que la broche de

5V peut fournir moins de 5V, et une tension supérieure à 12V pourrait endommager la carte parce que le régulateur de tension risque de chauffer.

Sur la carte on trouve les broches d'alimentation suivantes :

VIN : Broche d'alimentation externe.

GND : La broche de masse (0V).

5V : Cette broche fournit5V régulé.

3.3V : Cette broche fournit3.3V régulé.

III.4. Les broches numériques :[8]

Cette carte possède 14 broches d'Entrées/Sorties (de 0 à 13), elles fonctionnent en 5V et elles peuvent être configurées en entrée ou en sortie par programmation d'une manière dynamique.

Toutes ces Entrées/Sorties véhiculent des signaux TTL(Transisor-Transistor Logic), c'est-à-dire des signaux logiques ou signaux « tout ou rien » dont le niveau bas est nul et le niveau haut vaut 5V.

III.5. Les broches analogiques:

Cette carte possède 6 broches de lecture analogiques (de A0 à A5), elles sont prises en charge par un convertisseur CAN (Convertisseur Analogique Numérique) de 10 bits de résolutions, c'est-à-dire la sortie peut varier de 0 à 1023. Il se charge de convertir l'échantillon de tension en une grandeur numérique.

III.6. Les broches réservées : [8]

Certaines broches ont des fonctions spécialisées :

- Communication série de niveau TTL qui est disponible sur les broches **0** (entrée série asynchrone RX) et **1** (sortie série asynchrone TX).
- Deux entrées d'interruptions externes partagées avec les broches **2** et **3**.
- Les broches **3,5,6,9,10,11** peuvent être utilisées pour le signal PWM (Pulse With Modulation), c'est un signal numérique qui a une fréquence donnée a un rapport cyclique qui change.
- Quatre Entrées/Sorties d'interface avec le bus série normalisé de type SPI réparties comme suit : /SS sur la broche **10**, MOSI sur la broche **11**, MISO sur la broche **12** et SCK sur la broche **13**.
- Deux Entrées/Sorties d'interface série I2C réparties comme suit : SDA sur la broche **A4** et SCL sur la broche **A5**.

III.7. La communication :

La carte Arduino UNO peut communiquer avec un ordinateur, une autre carte Arduino ou d'autres microcontrôleurs et cela via les moyens suivants :

- **UART** (Universal AsynchronousReceiverTransmitter) : C'est un composant utilisé pour une communication série de niveau TTL, il fait la liaison entre l'ordinateur et le port série, il est disponible sur les broches 0 (RX) et 1 (TX).
- **I^2C** (Inter-Integrated Circuit) ou **TWI** (Two Wire Interface) :C'est un bus informatique permettant de relier facilement un microprocesseur et ses différents circuits.
- **SPI**(Serial Peripheral Interface) : C'est un bus de transmission de données série synchrone, ou les données peuvent circuler simultanément dans les deux sens.

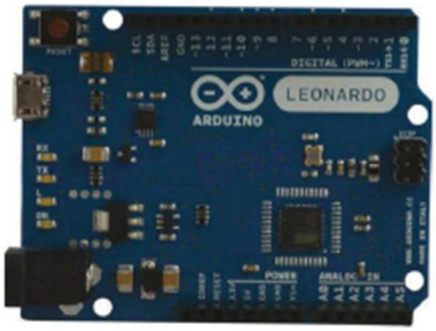
III.8. La Visualisation :

La carte Arduino UNO dispose de trois LED :

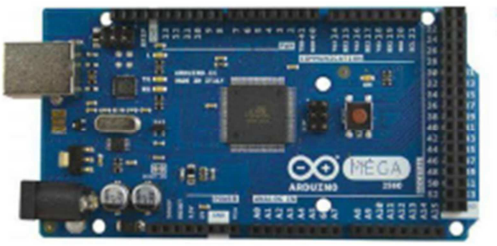
- Une connectée à la broche D13 du microcontrôleur sert à tester le matériel, clignote dès que la carte est sous tension.
- Les deux autres servent à visualiser l'activité sur la voie série (une pour l'émission et l'autre pour la réception).

IV- Les autres cartes Arduino:

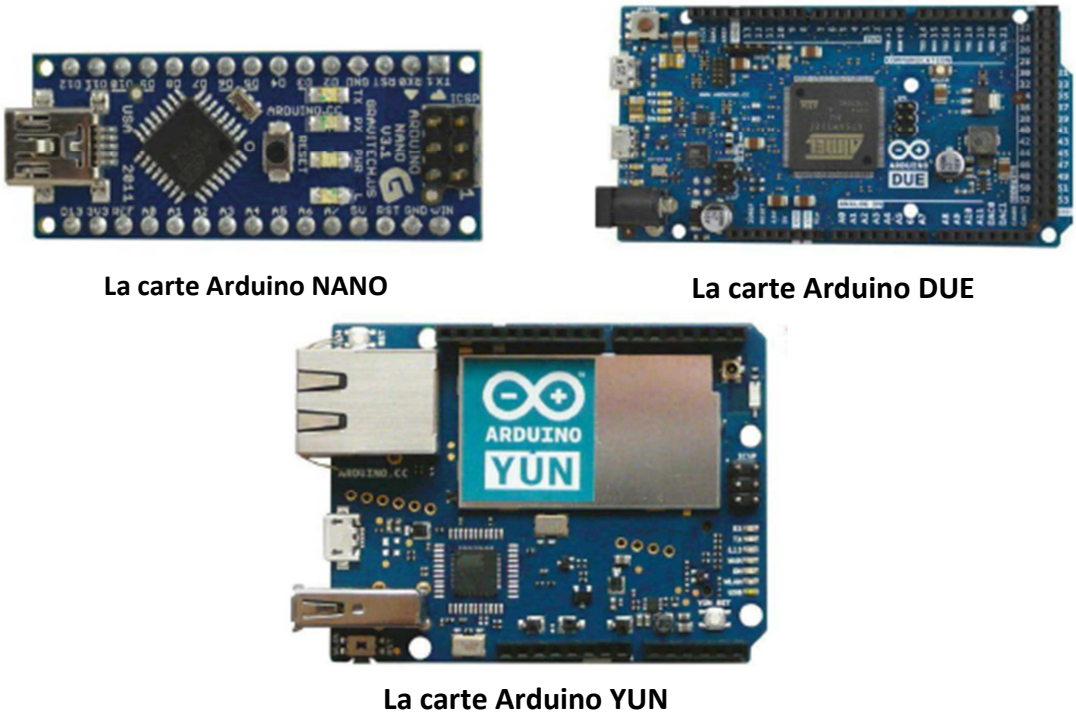
Plusieurs autres cartes Arduino existent, la figure II.3 présente quelques-unes :



La carte Arduino LEONARDO



La carte Arduino MEGA 2560



La carte Arduino NANO

La carte Arduino DUE

La carte Arduino YUN

Figure II.3. Autres cartes Arduino.

Le tableau suivant présente les caractéristiques des différentes cartes Arduino :

	Arduino LEONARDO	**Arduino MEGA 2560**	**Arduino NANO**	**Arduino DUE**	**Arduino YUN**
Microcontrôleur	ATmega 32U4	ATmega 2560	ATmega 328	AT91SAM3X8E	ATmega 32U4
SRAM	2.5 Ko	8 Ko	2 Ko	96 Ko	2.5 Ko
EEPROM	1 Ko	4 Ko	1 Ko	/	1 Ko
Mémoire flash	32 Ko	256 Ko	32 Ko	512 Ko	32 KO
E/S numériques (sorties PWM)	20 E/S (7PWM)	54 E/S (15PWM)	14E/S (6PWM)	54 E/S (15PWM)	20 E/S (5PWM)
Entrées analogiques	12	16	8	12	12
Taille	6.86cmx5.3cm	10.1cmx5.3cm	4.3cmx1.9cm	10.2cmx5.3cm	7cmx5.3cm
Tension d'alimentation	7-12V	7-12V	7-9V	7-12V	5 V

Fréquence d'horloge	16 MHz	16 MHz	16 MHz	84MHz	16 MHz
Autres options	/	/	/	2 sorties analogiques	Equipée d'un autre processeur de type Atheros AR9331

Table II.2. Caractéristiques des différentes cartes Arduino.

Le choix de la carte dépond de ce qu'on souhaite réaliser avec :

- Pour programmer à moindre coût, l'UNO est idéale.
- Pour brancher une multitude de capteurs, utiliser plusieurs périphériques en série, la MEGA est parfaite.
- Pour les mêmes propriétés que la MEGA, avec une puissance de calcul supérieure, il faut acquérir la DUE.
- Pour miniaturiser son système, il faut opter pour la NANO.
- Enfin si on cherche un véritable ordinateur embarqué, la YUN est le choix le plus évident.

V- IDE Arduino :[5]

Un environnement de développement a été créé pour programmer les différentes cartes Arduino avec son propre langage de programmation, qui est un mélange de C et de C++. Il offre une multitude de fonctionnalités.

L'environnement de développement Arduino fournit un éditeur de source, les opérations de compilation et de chargement dans la mémoire du microcontrôleur étant ramenées à des clicks sur des boutons dans l'IHM (très simple).

La **Figure II.4** présente l'interface du logiciel Arduino.

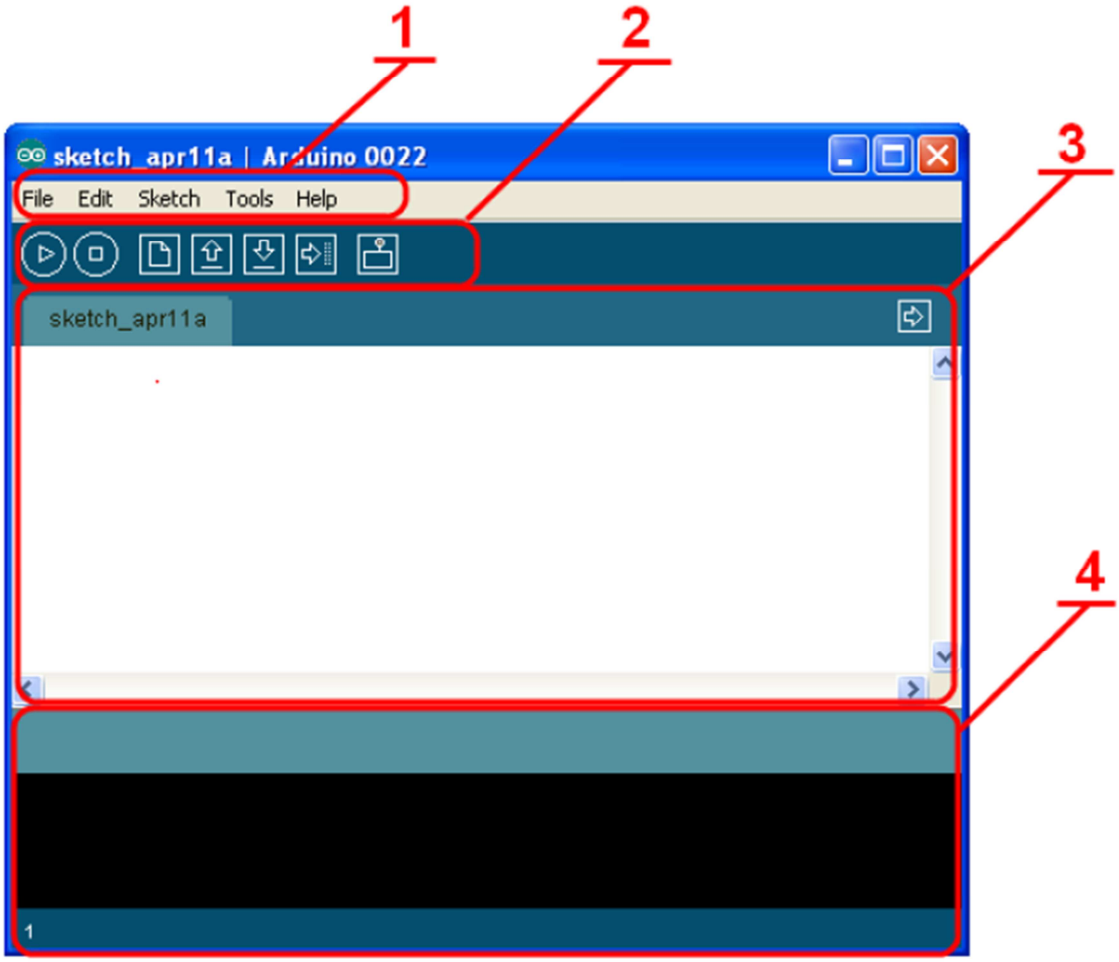


Figure II.4. Interface du logiciel Arduino.

1 : Barre de Menus.

2 : Barre d'icônes.

3 : Editeur.

4 : Console (affichage des résultats des différentes opérations).

V.1. Structure générale d'un programme Arduino :

Un programme ou sketch destiné à l'Arduino comporte toujours deux fonctions distinctes. Chaque fonction contient des blocs d'instructions.

La fonction setup() est appelée une seule fois lorsque le programme commence, elle est appelée fonction d'initialisation. Elle contient des instructions d'initialisation de certaines ressources de la carte, définition de la vitesse de fonctionnement du port série, etc.

La fonction loop() est appelée en permanence, elle comprend le code à exécuter et réalise l'essentiel du travail.

La **Figure II.5** montre la structure générale d'un programme Arduino.

```
1  //fonction d'initialisation de la carte
2  void setup()
3  {
4      //contenu de l'initialisation
5  }
6
7  //fonction principale, elle se répète (s'exécute) à l'infini
8  void loop()
9  {
10     //contenu de votre programme
11 }
```

Figure II.5. Structure générale d'un programme Arduino.

La **Figure II.6** montre un exemple d'un sketch (programme Arduino) qui permet de faire clignoter une LED.

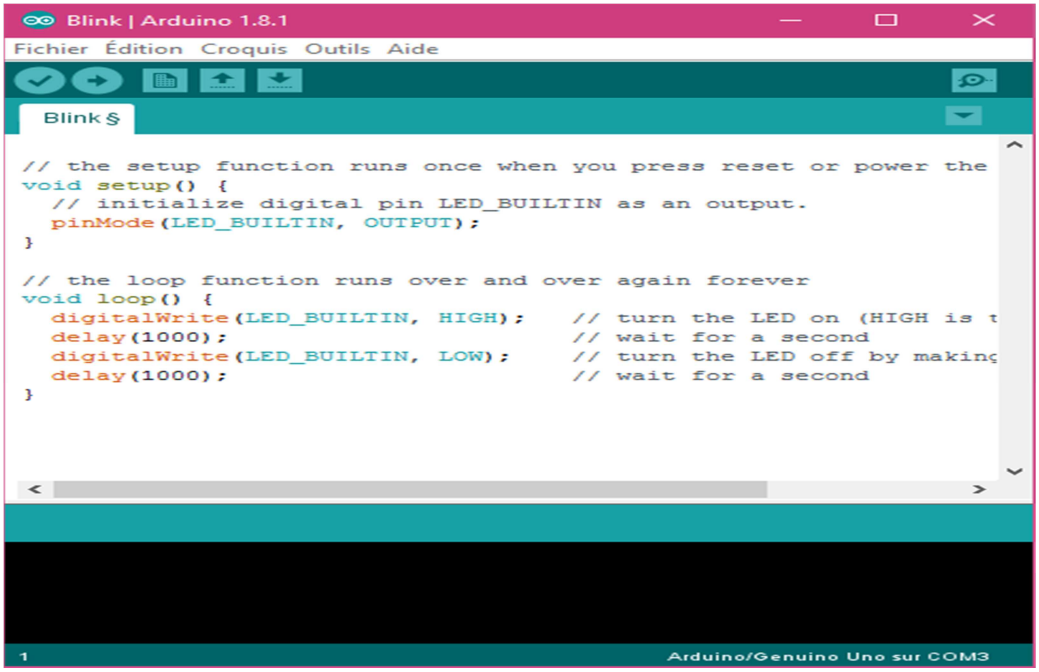


Figure II.6. Sketch pour clignoter une LED.

V.2. Transmission du sketch sur la carte Arduino:

Avant de téléverser le programme dans la carte, on doit passer par les étapes suivantes :

Etape 1 : Compilation.

Cette étape consiste à vérifier si le programme ne contient pas d'erreurs, et aussi à le traduire en langage machine (langage binaire).

Cette étape est réalisée par le compilateur. Le compilateur utilisé par l'Arduino est AVR GCC qui est un compilateur C/C++ pour processeur AVR. Elle se fait par un simple clic sur le bouton Compiler  .

La **Figure II.7** montre le rôle d'un compilateur.

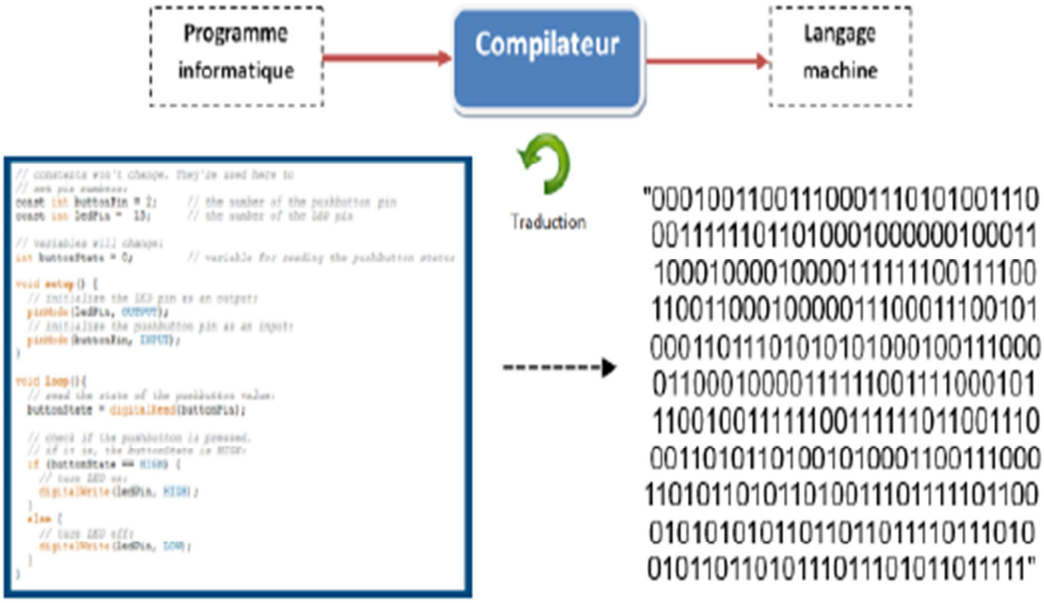


Figure II.7. Rôle d'un compilateur.

Etape 2 :Sélection de la carte Arduino.

Une fois le programme compilé avec succès, on doit sélectionner le type de la carte à utiliser dans le menu Outils.

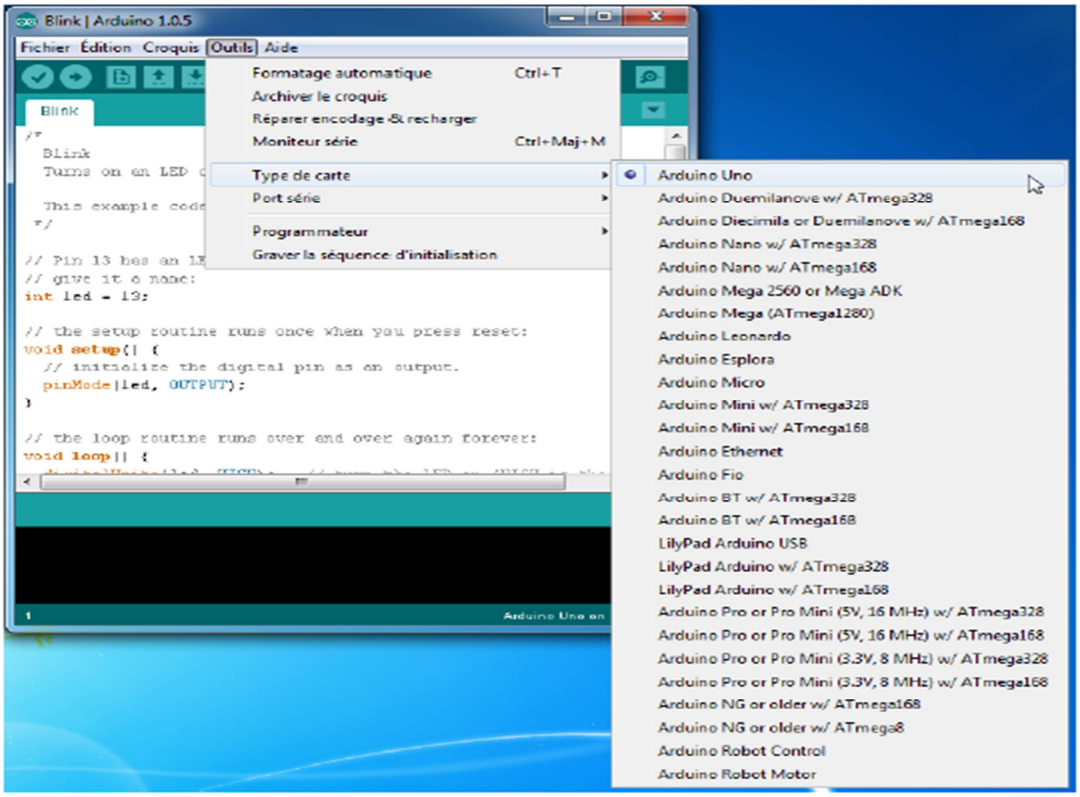


Figure II.8. Sélection de la carte à utiliser.

Etape 3 : Sélection du port de connexion de la carte.

Dans cette étape on sélectionne le port de connexion de la carte dans le menu Outils puis Port série.

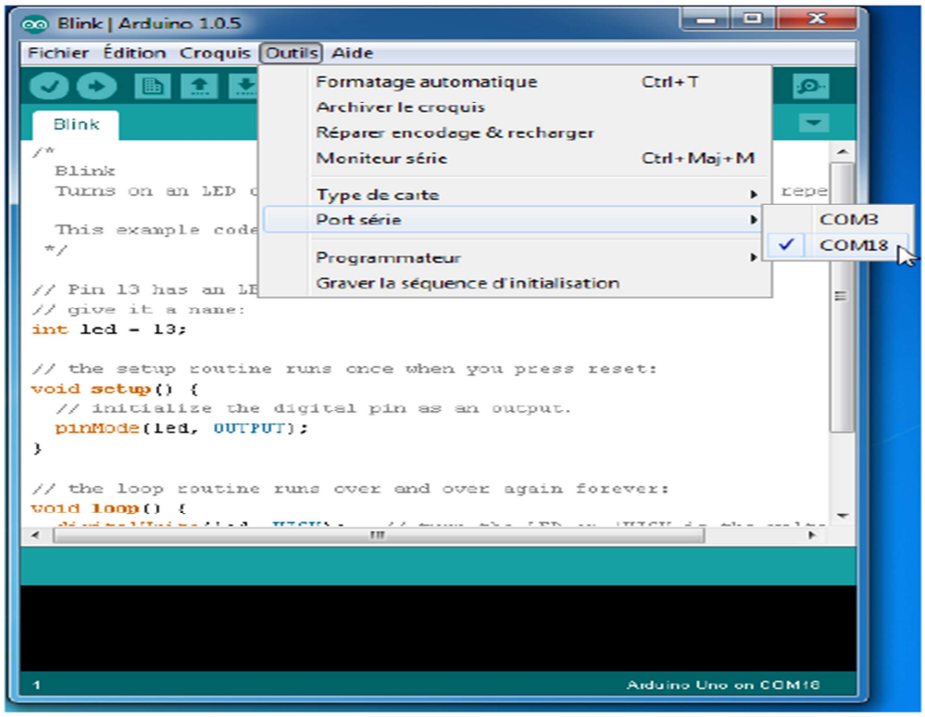


Figure II.9. Sélection du port de connexion de la carte Arduino.

Etape 4 : Téléverser le sketch dans la carte Arduino.

Dans cette étape on transmet le sketch dans la carte Arduino. La communication entre le PC et la carte se fait via le port USB. Cette étape se fait par un simple clic sur le bouton .

VI- Les Shields de la carte Arduino:

Les Shields Arduino appelés aussi carte d'interface ou boucliers sont des cartes qui se branchent sans soudure aux cartes Arduino ou à d'autres shields Arduino pour augmenter leurs capacités.

De nombreux Shields sont disponibles pour les cartes Arduino, nous allons présenter quelques-uns ici.

VI.1. Les Shields de communication:

VI.1.1. Le Shields Bluetooth :

Le Shield Bluetooth permet de détacher l'Arduino de son PC. Il permet aussi de contrôler son Arduino à partir d'un téléphone mobile ou tout autre appareil compatible Bluetooth grâce à de simples commandes série.

La **Figure II.10** présente un shield Bluetooth.

Figure II.10. Le Shield Bluetooth.

VI.1.2. Le Shield GSM :

Le Shield GSM permet à une carte Arduino de se connecter à internet, envoyer et recevoir des SMS et de passer des appels vocaux à l'aide de la bibliothèque GSM.

La **Figure II.11** présente un shield GSM.

Figure II.11. Le Shield GSM.

VI.1.3. Le Shield Xbee:

Le Shield Xbee permet à une carte Arduino de communiquer sans fil en utilisant Zigbee.

Figure II.12 présente un shieldXbee.

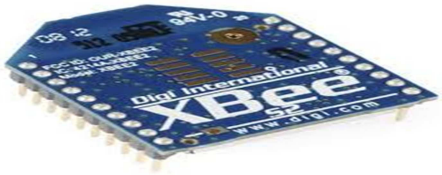


Figure II.12. Le Shield Xbee.

VI.1.4. **Le Shield Wifi:**

Le Shield Wifi permet à une carte Arduino de se connecter à internet via le réseau sans fil.

La **Figure II.13**présente un Shield Wifi.

Figure II.13. Le Shield Wifi.

VI.1.5. **Le Shield Ethernet :**

Le Shield Ethernet permet de relier la carte Arduino à internet en quelques secondes. Ce module se branche sur la carte Arduino, et se connecte au réseau avec un câble RJ45.

La **Figure II.14** présente un shield Ethernet.

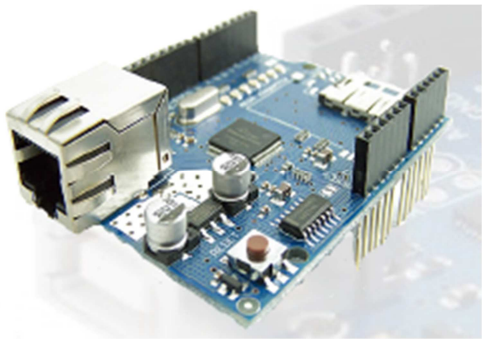
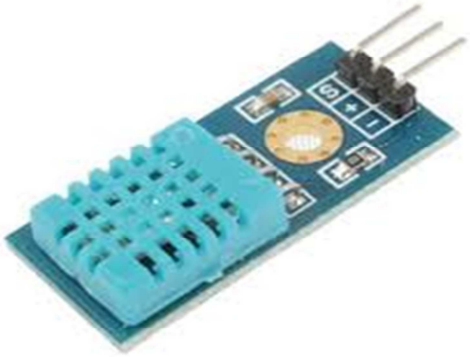


Figure II.14. Le Shield Ethernet.

VI.2. **Les capteurs :**

Les capteurs sont des dispositifs qui permettent de mesurer un paramètre physique tel que : température, humidité, mouvement…etc ou de capter sa présence, et transférer l'état observé en un signal analogique ou numérique.

La **Figure II.15** présente deux exemples de capteurs à gauche un capteur de température et à droite un capteur de mouvement.



Capteur de température Capteur de mouvement

Figure II.15. Exemples de capteurs.

VI.3. **Afficheur LCD (Liquid Crystal Display) :**

Ces afficheurs sont sous forme d'un petit circuit imprimé support comprenant tout à la fois l'afficheur lui-même et l'électronique de gestion. Cette électronique facilite la commande de l'afficheur par le microcontrôleur en le déchargeant de toute la partie gestion de l'afficheur et surtout de la génération, tout de même assez complexe, des signaux nécessaires. Ils permettent d'afficher des lettres, des chiffres et quelques caractères spéciaux.
La figure II.16 présente un afficheur LCD.

.

Figure II.16. Afficheur LCD.

VI.4. **Le Relais :**

Le shield Relais permet de relier à la carte Arduino des moteurs hautes puissances qui ne peuvent pas être contrôlés directement par les broches numériques Arduino. Le bouclier comporte quatre relais et quatre LED indiquent l'état marche / arrêt de chaque relais.

La figure II.17 présente unshield relais.

Figure II.17. Le shield relais.

VI.5. **Les moteurs :**

Les moteurs électriques permettent de faire tourner, de déplacer ou encore d'interagir avec divers éléments de la vie réelle.

Il existe trois grands types de moteurs électriques :

– les moteurs à courant continu (moteurs DC).
– les servomoteurs.
– les moteurs pas à pas.

L'Arduino Uno n'est pas conçu pour alimenter à lui tout seul des moteurs. La raison est que l'intensité en sortie disponible pour une broche est de l'ordre de 20 mA et ça ne fait vraiment pas beaucoup pour alimenter un petit moteur électrique. Mais, heureusement, il y a de très nombreuses solutions pour compléter l'Arduino et le faire piloter toutes sortes de moteurs.

La **Figure II.18** Présente un shield moteur.

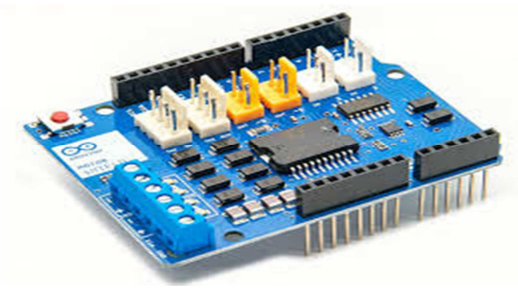


Figure II.18. Le Shield moteur.

VI.6. **Bibliothèques pour les Shields Arduino:**

Les Shields que l'on vient de voir ci-dessus, et tous les autres qui existent possèdent tous leur bibliothèque (Library), téléchargeable ou présente par défaut dans l'environnement de développement. Cette bibliothèque fonctionne exactement comme le pilote d'un périphérique de PC.

Conclusion :

Dans ce chapitre nous avons présenté la technologie Arduino, ses caractéristiques matérielles, son environnement de développement et les différentes cartes qui existent. Nous avons vu aussi les différents Shields Arduino.

Les déférents concepts traités dans ce chapitre nous aiderons à mieux comprendre notre environnement de développement et les notions fondamentales pour mener à bien notre travail.

Dans le prochain chapitre, nous allons présenter la conception de notre application.

Chapitre III: Conception

Introduction :

Dans ce chapitre, nous allons présenter la conception de notre projet qui consiste à réaliser un système d'alarme anti-intrusion par barrière infrarouge.

Un système anti-intrusion a pour objectif d'alerter au cas où des intrus pénétreraient dans des lieux sécurisés. Des capteurs (barrière Infrarouge, détecteurs de mouvement ou barrière laser…) repèrent l'intrusion et déclenchent l'alarme qui fait fuir les intrus. Donc, un système d'alarme anti-intrusion a pour vocation de détecter une tentative d'intrusion le plus tôt possible afin de mettre en œuvre des moyens de dissuasion.

L'alarme anti-intrusion se déclenche lors d'une détection d'intrusion selon des options choisies. Plusieurs actions peuvent être menées simultanément : une alarme sonore, des projecteurs qui s'allument, et envoies d'un SMS à un numéro de téléphone (société de télésurveillance, police, le propriétaire des lieux ou une personne de confiance).

I- Définition du système d'alarme anti-intrusion:

Le système d'alarme anti-intrusion est un système efficace pour la protection de biens et de personnes, capable de détecter toute intrusion et de signaler l'événement rapidement à l'aide d'alarmes sonores, visuelles et messages téléphoniques.

Le système est composé d'une centrale de commande, de capteurs (détecteurs) et des avertisseurs.

II- Structure générale de notre système:

Notre système est une alarme anti intrusion par barrière infrarouge à base d'un microcontrôleur AVR (ATmega328 d'Atmel), qui est un outil de sécurité qui détecte toute intrusion en déclenchant une alarme sonore et en envoyant un SMS au propriétaire au moyen d'un module GSM.

La **Figure III.1** présente les cinq blocs qui constituent notre système et qui sont :

- Bloc d'alimentation ;
- Bloc de détection ;
- Bloc de périphériques d'E/S ;
- Bloc de gestion du système ;
- Bloc d'alerte (avertisseurs).

Nous allons étudier chaque bloc en détail dans la section qui suit.

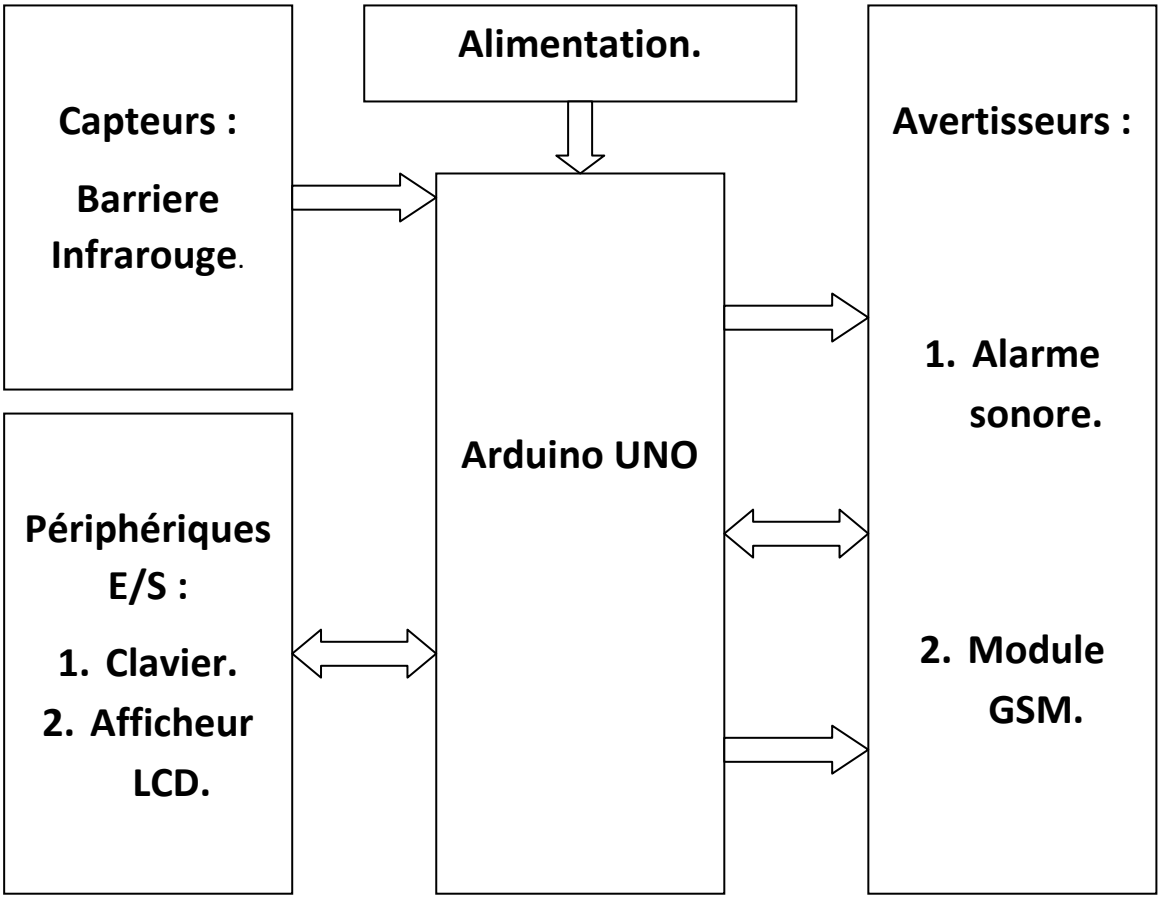


Figure III.1. Schéma générale de notre système d'alarme anti-intrusion.

III- **Conception matérielle:**

Les schémas de branchement des différents composants que nous avons utilisés avec la carte Arduino UNO ont été réalisé avec le logiciel Fritzing, ce que nous allons présenter dans les prochaines sections.

Fritzing : est un logiciel open-source multiplateforme permettant de construire des schémas des circuits que nous utilisons avec Arduino. Plusieurs vues sont disponibles : platine d'essai, schémas électriques et circuit imprimé. Il permet aussi l'export en image pour figurer sur internet.**[09]**

III-1. **Bloc d'Alimentation : [10]**

Pour alimenter notre système d'alarme anti intrusion, on a besoin d'un circuit d'alimentation qui permet de délivrer une tension continue de 5V et un courant qui ne doit pas dépasser 500 mA.

Pour alimenter le module GSM/GPRS (SIM900), on a besoin d'un circuit d'alimentation qui permet de délivrer une tension continue de 9V.

III-2. **Bloc de détection :**

La détection dans notre système se fait à l'aide d'une barrière infrarouge.

Le Capteur infrarouge est constitué d'un émetteur de lumière infrarouge et un récepteur qui détecte la réflexion de l'intensité lumineuse. En temps normal, le récepteur reçoit toujours la même quantité de rayonnement de la part de l'émetteur, même si des changements de météo modifient épisodiquement le taux de réception. Dès le moment où le récepteur ne reçoit plus les rayonnements émis par l'émetteur en quantité normale, la barrière considère qu'un corps interfère avec le champ de surveillance. C'est à ce moment-là que se déclenche le signal. Il dispose d'un potentiomètre qui sert à régler la distance ou bien l'ordre de détection, c'est-à-dire choisir à quelle distance le capteur nous alerte de la présence d'un obstacle. Il est constitué également de deux LEDs, une qui s'allume lorsque le capteur est mis sous tension et une autre qui s'allume lors de la détection d'un obstacle.

Spécifications techniques de la barrière IR utilisée :

- Numéro de modèle: FC-51
- Angle de détection: 35 °
- Tension de fonctionnement: 3.0V - 6.0V
- Plage de détection: 2cm - 60cm (réglable avec potentiomètre)
- Dimension globale: 4,5 cm (L) x 1,4 cm (L), 0,7 cm (H)
- Niveau de sortie actif: Sorties Niveau logique faible lorsque l'obstacle est détecté
- Niveau de sortie inactivé: Sorties Niveau logique élevé lorsque l'obstacle n'est pas détecté. Ce capteur dispose de 3 broches :

- **VCC :** c'est pour l'alimentation qui soit 3.3V ou bien 5V.
- **GND :** c'est la masse.
- **OUT :** c'est la sortie du signal de détection.

Figure III.2. Barrière infrarouge utilisée.

La **figure III.3.** Présente le schéma de câblage d'une barrière infrarouge avec la carte Arduino UNO.

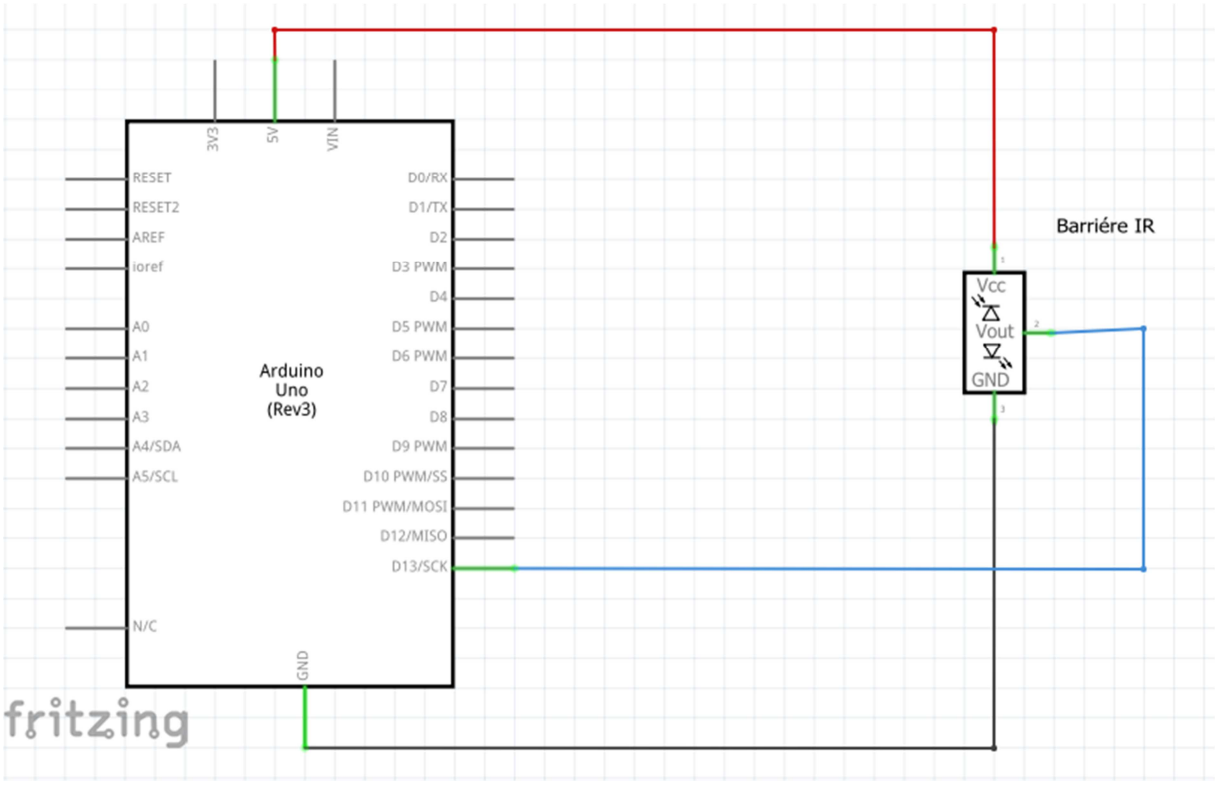


Figure III.3. Schéma de branchement de la barrière IR avec Arduino.

III.3. **Bloc de gestion du système :**

Ce bloc se charge de contrôler tous les autres composants, il s'agit de la carte Arduino UNO ou plus précisément du microcontrôleur AVR que nous avons présenté précédemment. Sa fonction est de vérifier et de traiter les données recueillies par le bloc de détection, ensuite envoyer les commandes à exécuter vers le bloc d'action de signalisation (avertisseurs) en fonction de la programmation choisie.

III.4**. Bloc d'alertes:**

Deux alertes sont menées simultanément :

- Une alerte sonore à l'aide d'un Buzzer ;
- Une alerte par l'envoie d'un SMS au moyen d'un module GSM.

III.4.1. **L'alerte sonore :**

Afin de dissuader l'intrus, il faut mettre en place un avertisseur sonore. Nous avons utilisé un buzzer qui produit un son caractéristique lorsqu'on lui applique une tension. Le buzzer assure deux fonctions différentes dans notre système :

Avertissement : le buzzer produit 3 bips sonores lors de la détection d'intrusion afin de donner une chance au propriétaire de désactiver l'alerte sonore et l'envoie du SMS dans le cas où il oublie de désactiver l'alarme.

Alarme : le buzzer produit une alarme sonore sans arrêt afin de dissuader l'intrus.

Ce buzzer possède deux broches :

- La première broche : c'est la masse (GND) ;
- La deuxième broche : c'est l'alimentation (VCC).

Figure III.4. Buzzer utilisé.

La **figure III.5** présente le schéma de câblage d'un Buzzer avec la carte Arduino.

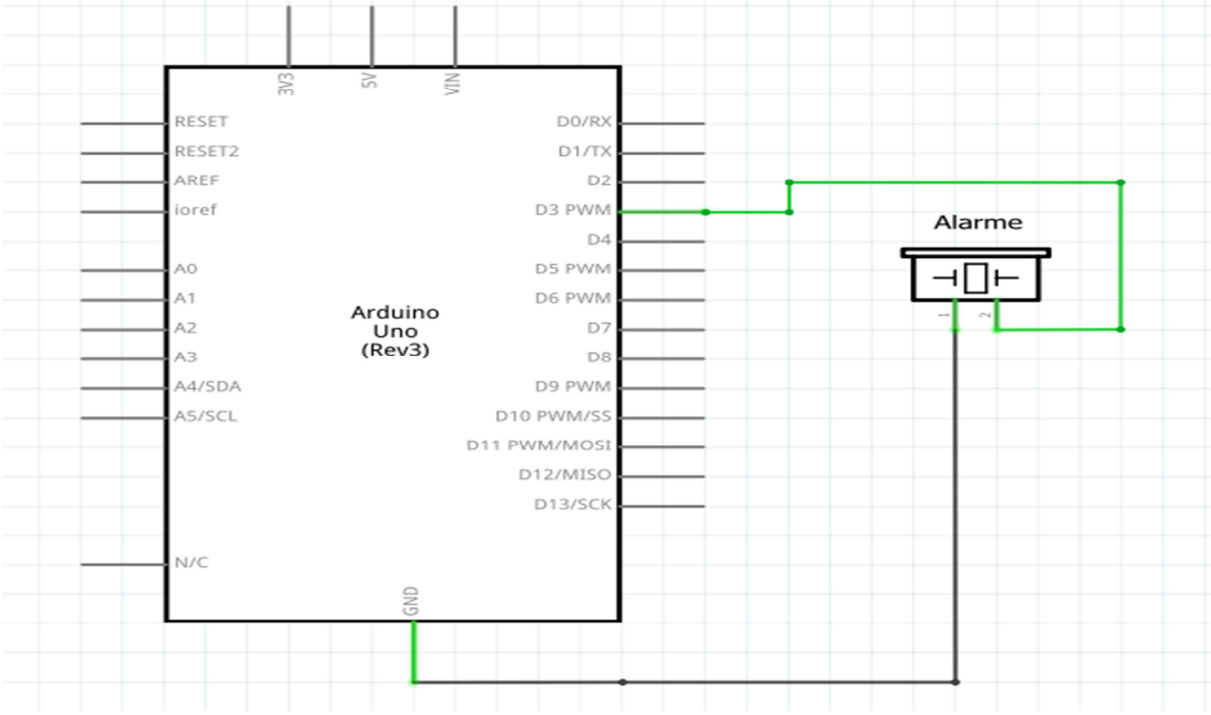


Figure III.5. Schéma de branchement du buzzer avec la carte Arduino.

III.4.2. **Signalisation par un module GSM :**

L'alarme GSM est aujourd'hui la solution la plus sûre pour une protection efficace. Elle se déclenchera dès qu'une intrusion sera détectée. L'alarme GSM dispose d'un module de communication GSM SIM900, qui intègre une carte SIM, ce module permet d'avertir le propriétaire sur son mobile grâce à un enregistrement des numéros à contacter effectué au préalable lors de la mise en fonctionnement de l'alarme.

- La norme GSM :

GSM (Global System for Mobile communications) est une norme numérique européenne utilisant plusieurs bandes de fréquences notamment de 900 à 1800 MHz. Le système GSM est aujourd'hui le principal système mobile.

La norme GSM autorise un débit de 9,6 kbps, ce qui permet de transmettre la voix ainsi que des données numériques de faible volume par exemple des messages textes (SMS) ou des messages multimédia (MMS).

- GSM SIM900 :

Le SIM900 est un composant GSM/GPRS fonctionnant sur les bandes (850, 900, 1800, 1900 MHz) et communiquant à travers une liaison série. Ce composant est capable d'envoyer et de recevoir des SMS, et aussi des appels. Il est compatible avec Arduino et les compatibles Arduino. Il est configuré et contrôlé via son UART en utilisant des commandes AT simples.

Les commandes AT sont définies dans la norme GSM. AT est l'abréviation de ATtention. Ces 2 caractères sont toujours présents pour commencer une ligne de commande sous forme de texte. Les commandes permettent la gestion complète du mobile. En général, il faut taper les commandes AT en MAJUSCULES. La commande AT tout cours doit donner la réponse "OK". Voici quelques commandes AT pour l'envoie d'un SMS :

- La commande **AT+CPIN** permet de déverrouiller la carte SIM.
- La commande **AT+CMGS** permet d'envoyer un SMS.
- La commande **AT+CMGF=1** permet d'activer le mode texte pour l'envoi du SMS. Cette commande renvoi le code **OK** en cas de réussite. Le temps d'attente maximal est de 1 secondes.

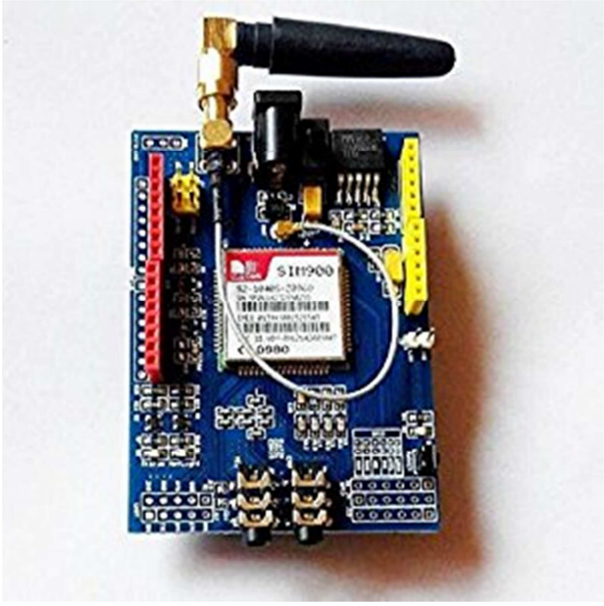


Figure III.6. SIM 900 utilisé.

La **figure III.7** présente le schéma de câblage de la SIM900 avec la carte Arduino.

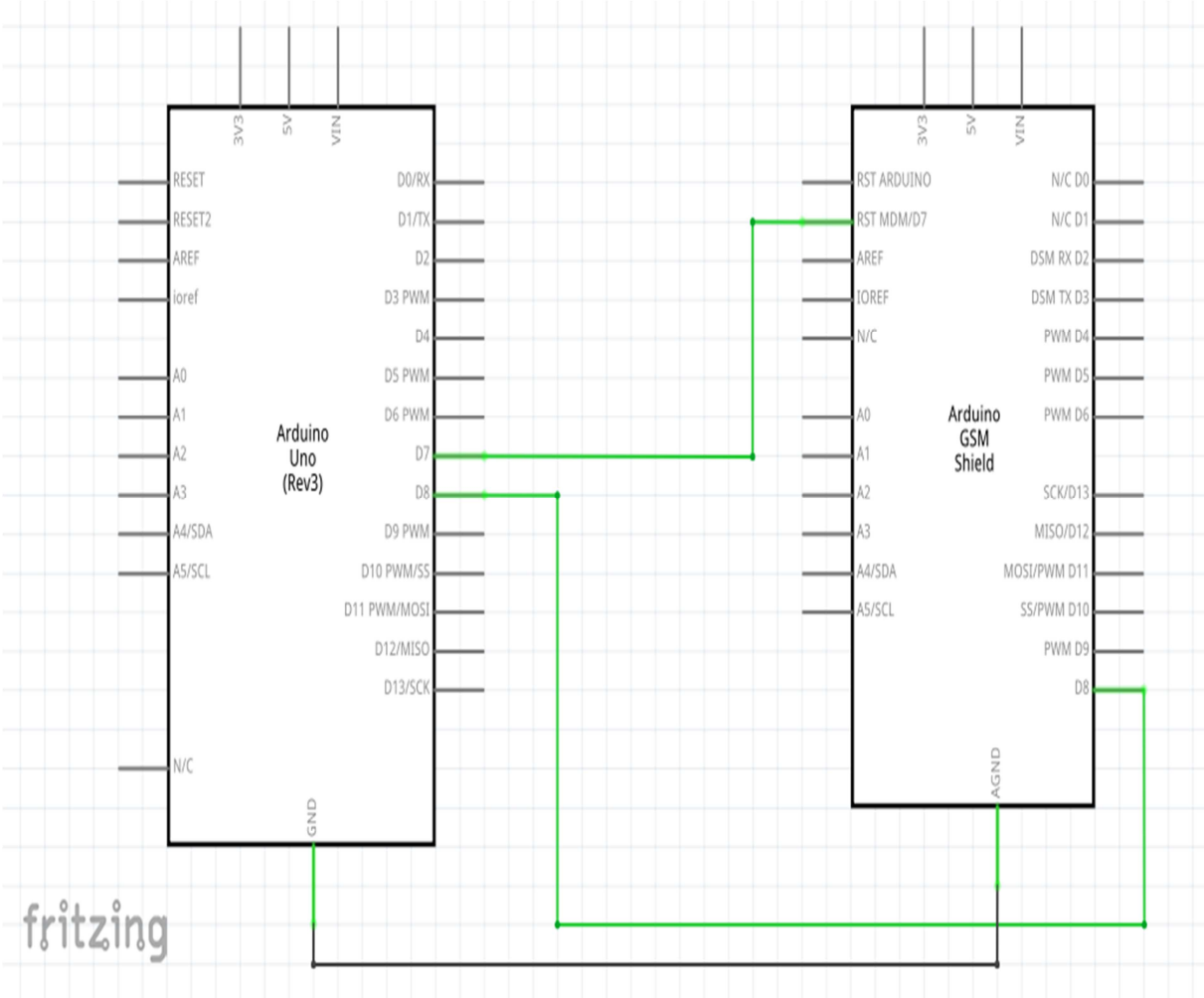


Figure III.7. Schéma de branchement du SIM900 avec la carte Arduino.

III.5. **Bloc de périphériques d'E/S :**

Ce bloc est utilisé pour configurer le code de l'activation ou désactivation de l'alarme, cela se fait à l'aide d'un clavier pour entrer le code d'activation ou désactivation de l'alarme, et un écran LCD pour afficher si le code saisi est correct ou erroné.

III.5.1. **Clavier utilisé :**

Le clavier est constitué de 16 boutons poussoirs interconnectés de façon à former une matrice 4x4 (4lignes et 4 colonnes).Un appui sur une touche court-circuite la ligne et la colonne correspondantes.

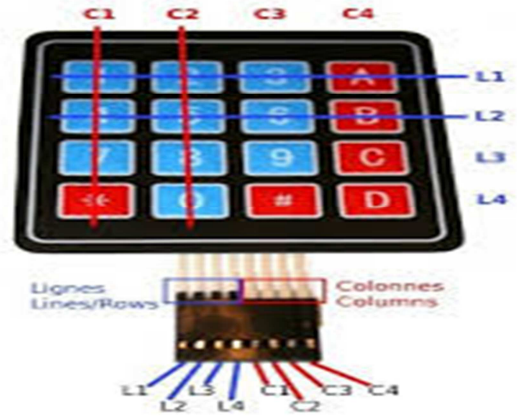


Figure III.8. Clavier utilisé.

Les lignes sont des sorties. Les colonnes sont des entrées maintenues au niveau haut par une résistance interne à Arduino. Le système envoie par balayage un niveau bas sur chaque ligne (1 seule à la fois) et balaye les colonnes en lecture. Quand il lit un niveau bas, c'est que la colonne est reliée par une touche appuyée à la ligne qui est basse à ce moment.

La figure III.9 présente le schéma de branchement du clavier 4X4 avec la carte Arduino.

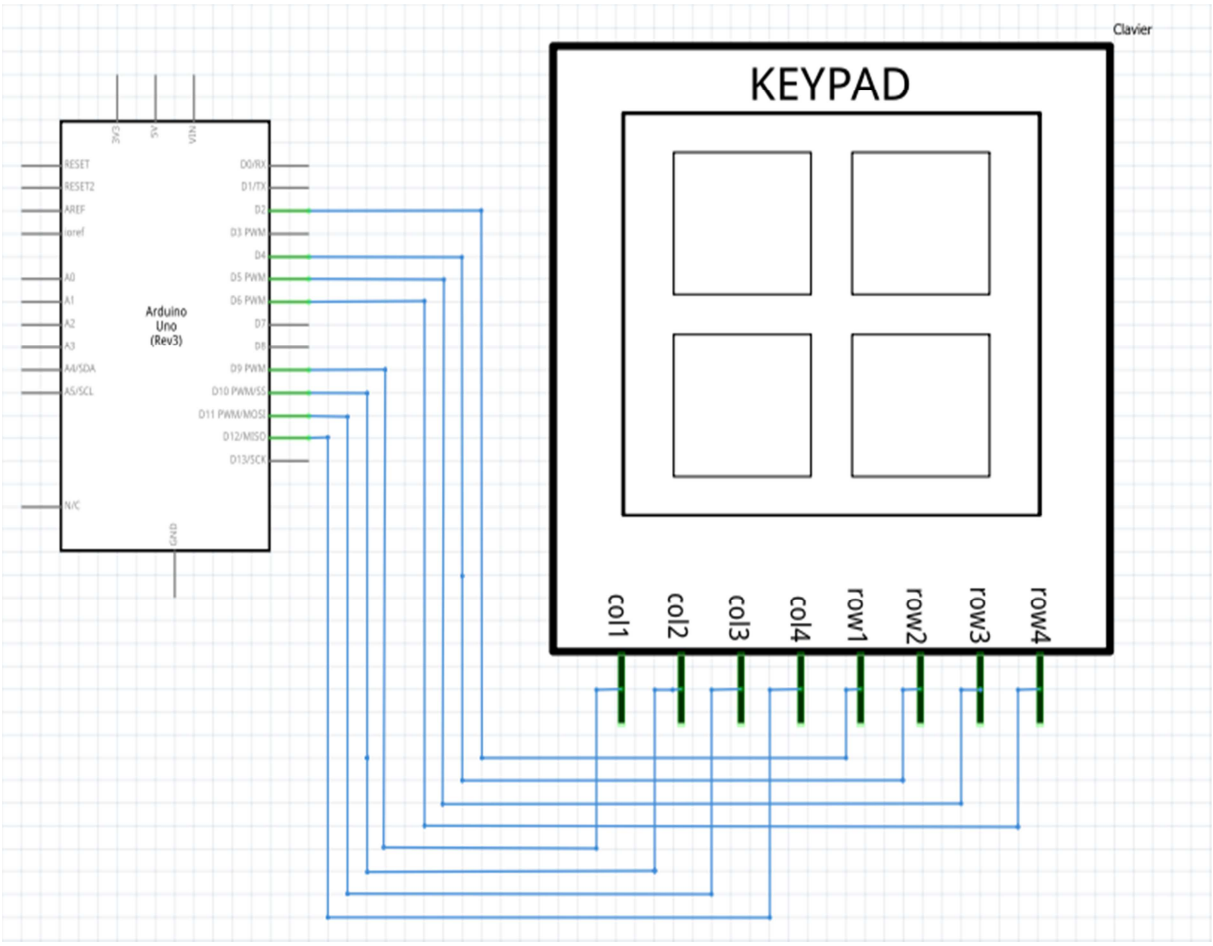


Figure III.9. Schéma de branchement d'un clavier matriciel 4X4avec la carte Arduino.

Ce dispositif est placé à l'entrée de la zone de surveillance pour activer l'alarme ou annuler les alertes, et cela en saisissant le code correspondant à chaque opération.

III.5.2. **Ecran LCD :**

Ce dispositif est placé à l'entrée de la zone sous surveillance pour visualiser le code correspondant à chaque opération (activation de l'alarme ou annulation d'alertes).

Un afficheur LCD est constitué de deux lames de verre, distantes de 20 micromètre environ. L'espace entre elles est rempli de cristal liquide. Il permet d'afficher 32 caractères reparties sur 2 lignes (dit afficheur 16X2). Il possède 16 broches qui permettent de raccorder l'afficheur avec notre Arduino, nous détaillons le rôle de chaque broche dans le tableau III.1.

Figure III.10. Ecran LCD utilisé.

On décrit les broches de cet écran LCD dans le tableau ci-dessous :

N° Broche	Nom	Fonction
1	VSS	Masse (0V).
2	VDD	Alimentation (5V).
3	VO	Une tension variable entre 0V et 5V permet le réglage du contraste de l'afficheur.
4	RS	Sélection du registre, grâce à cette broche capable de faire la différence entre une commande et une donnée.
5	R/W	Lecture ou écriture.
6	E	Entrée de validation active sur front descendant.
7	D0	Bus de données bidirectionnel.
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	A	Anode rétro éclairage (5V).
16	K	Cathode rétro éclairage (masse).

Table III.1. Description des broches d'un afficheur.

Pour un bon fonctionnement de l'afficheur LCD nous avons utilisé un potentiomètre qui permet de régler la luminosité de l'afficheur.

La figure III.11 Présente un schéma de câblage d'un afficheur LCD (16X2) et un potentiomètre avec la carte Arduino.

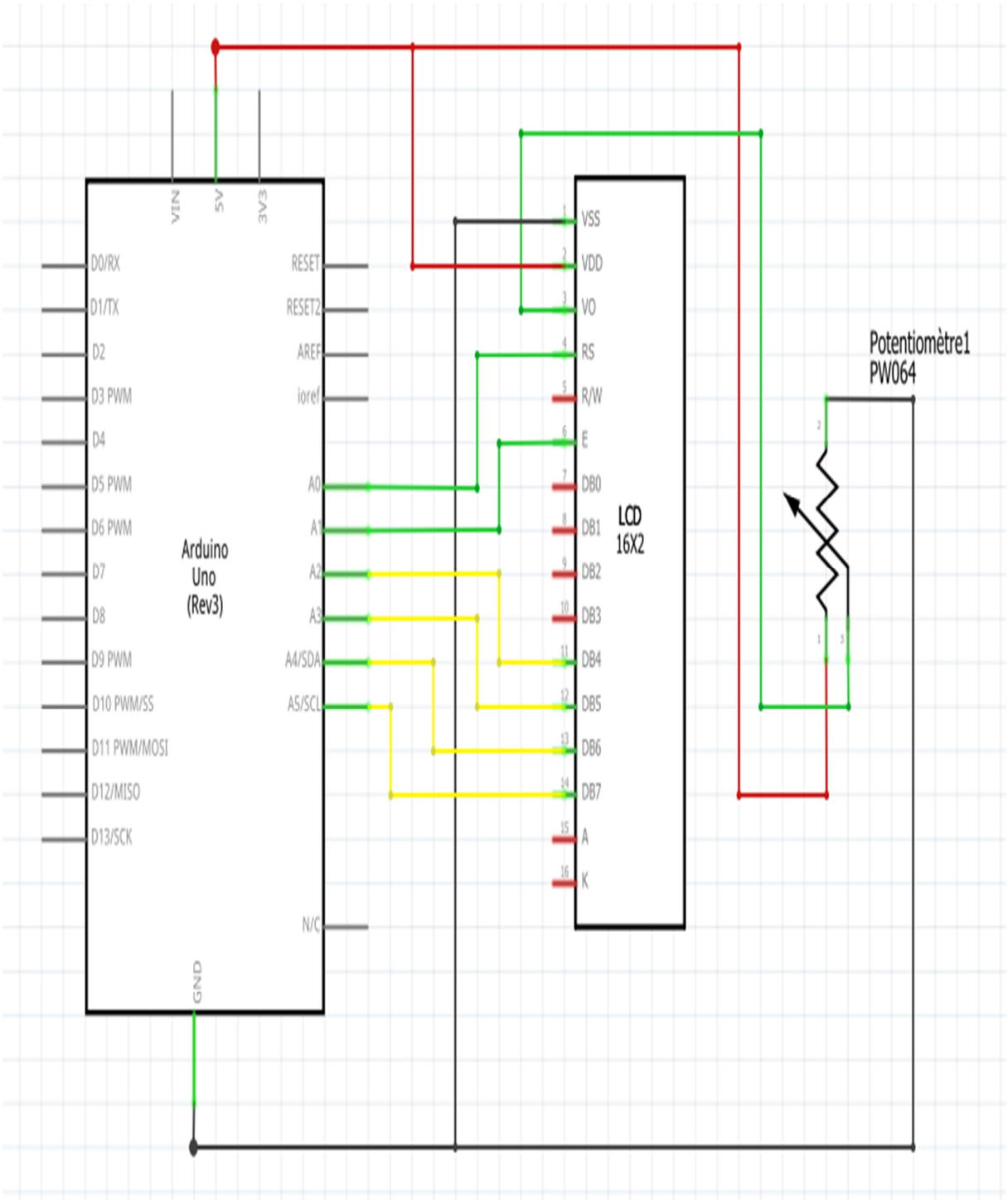


Figure III.11. Branchement de l'afficheur LCD avec la carte Arduino.

III.6. **Schéma générale du système :**

IV- **Organigramme de notre système :**

Le fonctionnement de notre système est résumé dans l'organigramme de la **figure III.13**.

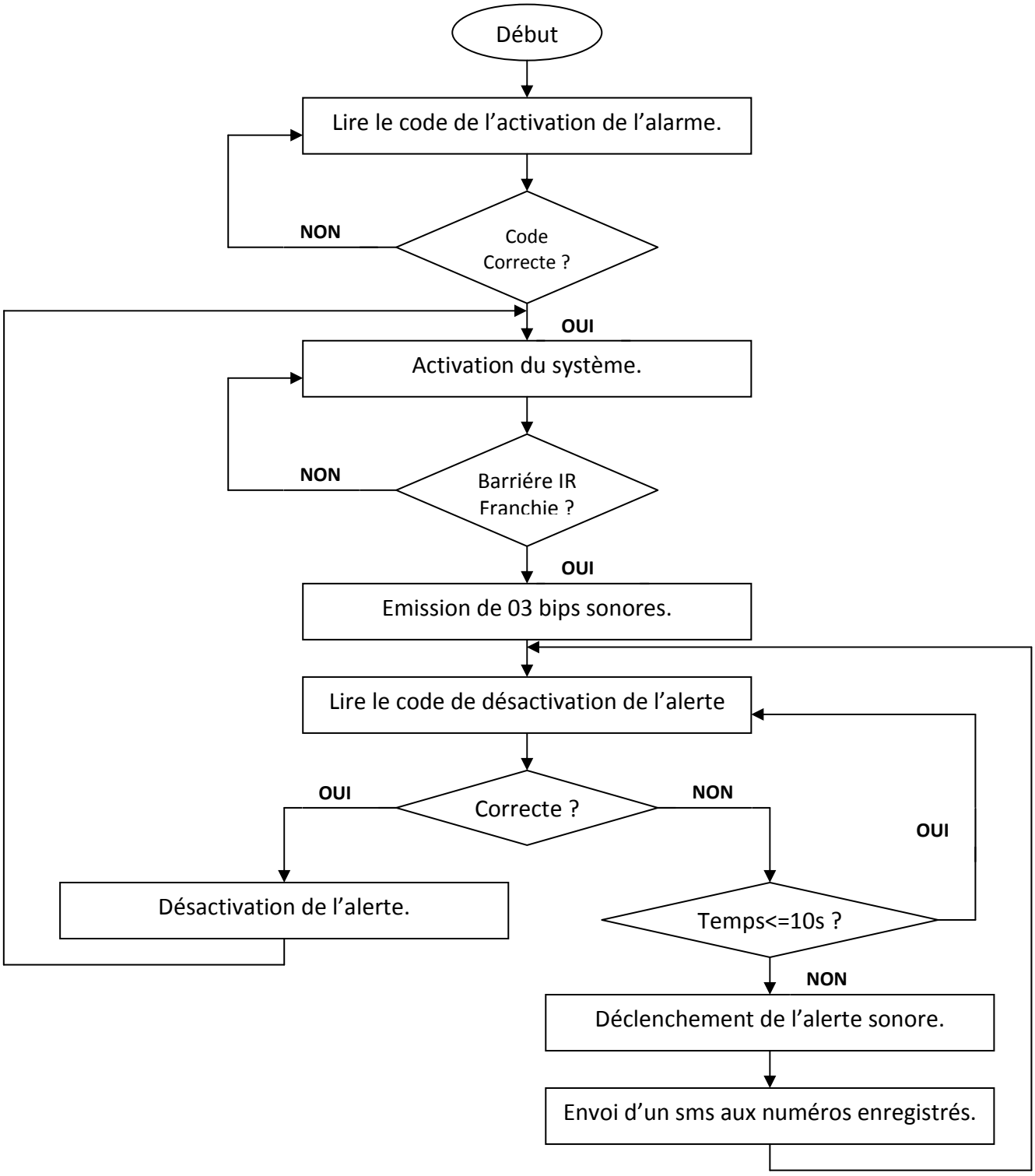


Figure III.13. Organigramme de notre système.

Notre système de surveillance anti-intrusion par barrière infrarouge permet de surveiller et de détecter toute intrusion. Au début, on doit saisir le code

d'activation de l'alarme, l'écran LCD affiche 'Code activ ?' et on saisit le code via le clavier, s'il est correct l'alarme sera activée sinon on doit ressaisir le code. Quand un intrus est détecté par le capteur PIR, le buzzer va faire trois bips sonores, et l'écran va afficher 'code de désactiv?', l'intrus aura10s pour saisir le bon code sur le clavier si le bon code est saisie les alertes seront désactivées et un message sera affiché sur l'écran LCD « Alarme désactivée». Sinon après 10s, si le code correct n'est pas encore saisi un message d'alerte sera envoyé aux numéros enregistrés et l'alarme sonore sera déclenchée sans arrêt.

Conclusion :

Dans ce chapitre nous avons présenté la conception matérielle et la conception logicielle de notre système.

Dans le prochain chapitre, nous allons présenter la réalisation de notre système, et nous allons exposer les résultats des tests qu'on a effectués sur notre système.

Chapitre IV: Réalisation

Introduction :

Dans ce chapitre, nous allons voire les étapes de réalisation de notre système. Nous allons présenter les outils utilisés, ainsi que l'implémentation de notre application. À la fin nous allons effectuer une série de tests afin d'observer le comportement de notre système.

I- Les composants matériels utilisés :

Pour la réalisation de notre projet nous avons utilisé un ensemble de composants matériels. Chaque composant a un rôle bien précis dans notre système.

Le tableau IV.1 ci-dessous présente les différents composants utilisés, le prix approximatif actuel ainsi que le rôle de chaque composant.

Composant	Nombre	Prix unitaire (DA)	Rôle
Carte Arduino UNO	1	3000	C'est le cœur de notre système, elle permet de gérer tous les autres composants.
Shield GSM/GPRS (SIM900)	1	15000	Permet d'envoyer un SMS aux numéros enregistrés sur sa carte SIM intégrée.
Barrière Infrarouge	1	650	Permet de détecter si un intrus pénètre dans notre zone de surveillance.
Buzzer	1	150	Permet de produire un son quand on lui applique une tension.
Clavier	1	400	Permet de saisir le code d'activation ou désactivation de notre système.
Ecran LCD	1	800	Permet d'afficher des messages d'informations.
Potentiomètre	1	50	Permet d'ajuster le contraste de l'écran LCD.
Plaque d'essai	1	600	Permet d'établir rapidement un circuit électronique.
Fils	40	15	Permettent de relier les différents composants.

Cable USB	1	150	Permet de faire communiquer la carte Arduino avec et l'ordinateur et en même temps l'alimenter.

Table IV.1. L'ensemble des composants de notre système.

Le prix total de notre système : 21400 DA

Notre projet est réalisé dans le cadre pédagogique et non pas dans le cadre commercial.

II- **Branchement des différents composants avec la carte Arduino :**

II.1. **Module GSM/GPRS (SIM900) :**

Pour câbler le SIM900 avec la carte Arduino UNO, on doit utiliser les broches D7 et D8 de cette dernière. Donc la broche D7 du SIM900 sera connecté à la broche D7 de la carte Arduino UNO, et la broche D8 du SIM900 avec la broche D8 de la carte Arduino UNO, ainsi une communication série sera établie d'une manière logicielle.

Notons que toutes fois les deux modules doivent partager la même masse (GND).

La figure IV.1 présente le branchement du module SIM900 avec la carte Arduino UNO.

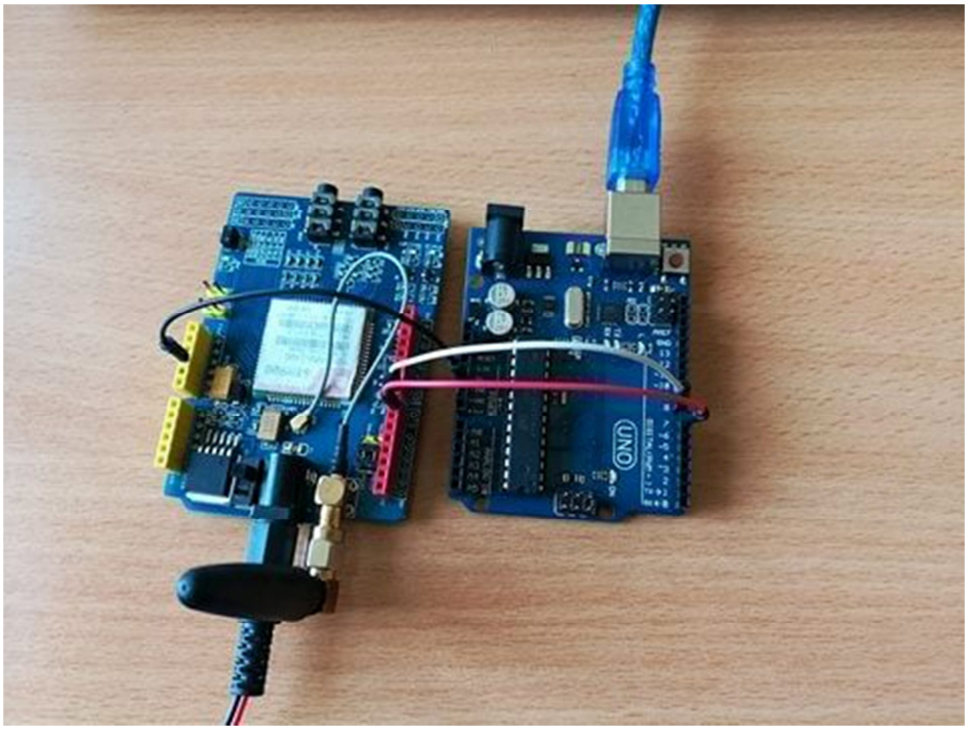


Figure IV.1. Branchement du module SIM900 avec la carte Arduino UNO.

II.2. **Barrière Infrarouge :**

Pour câbler la barrière IR avec la carte Arduino UNO, nous avons choisi d'utiliser la broche D13 de cette dernière pour recevoir la sortie de la barrière IR.

La figure IV.2 présente le branchement de la barrière infrarouge avec la carte Arduino UNO.

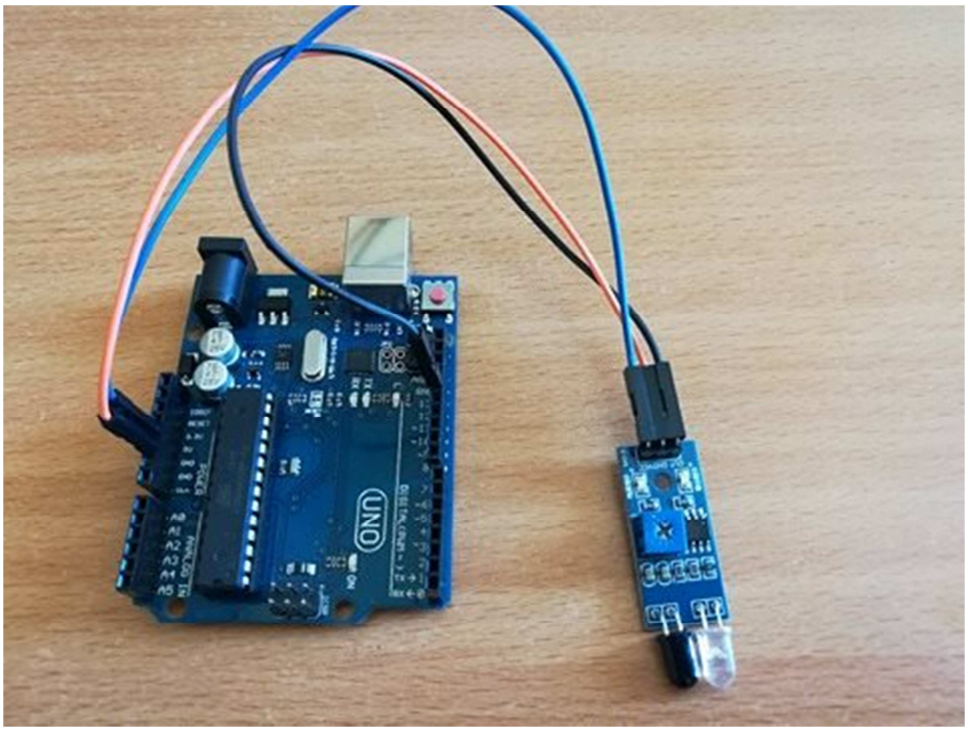


Figure IV.2. Branchement de la barrière infrarouge avec la carte Arduino UNO.

Les deux fils supplémentaires sont pour VCC (5V) et GND (0V).

II.3. **Buzzer :**

Pour câbler le Buzzer avec la carte Arduino UNO, nous avons choisi d'utiliser la broche D3 de cette dernière, puisque le buzzer a besoin d'une broche qui dispose d'un signal PWM (Pulse With Modulation) pour pouvoir utiliser la fonction **tone()** dans le programme.

La figure IV.3 présente le branchement du buzzer avec la carte Arduino UNO.

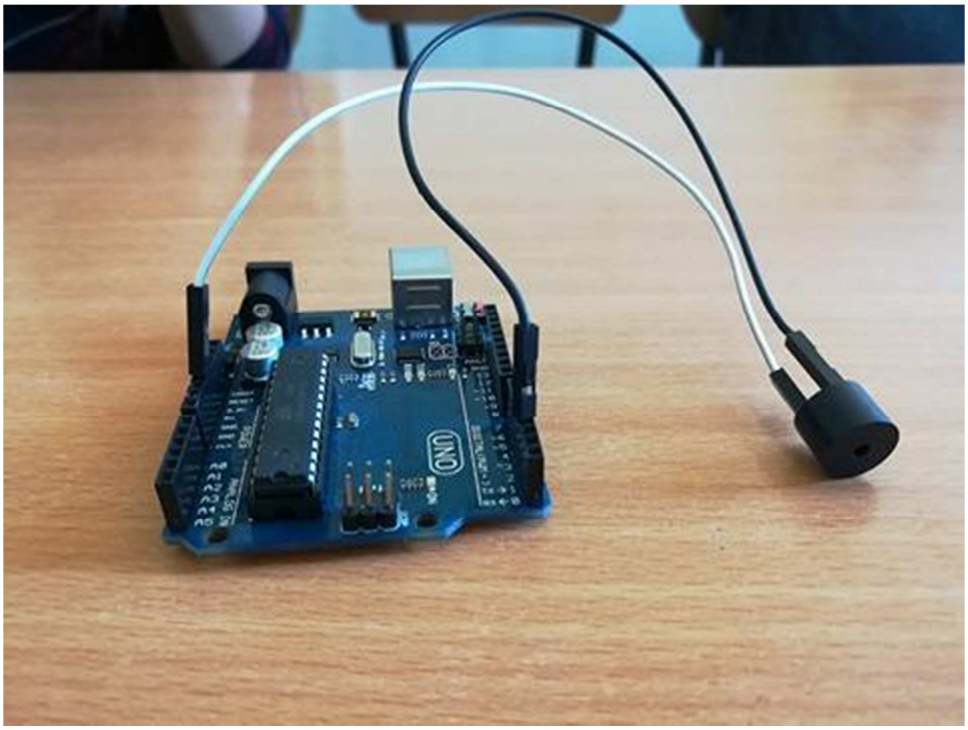


Figure IV.3. Branchement du buzzer avec la carte Arduino UNO.

L'autre fil apparaissant sur l'image, est relié à la masse de la carte Arduino.

II.4. **Clavier :**

Pour câbler le clavier avec la carte Arduino UNO, nous avons choisi d'utiliser les broches (D2, D4, D5, D6) pour les lignes respectivement (L1, L2, L3, L4) et les broches (D9, D10, D11, D12) pour les colonnes respectivement (C1, C2, C3, C4).

La figure IV.4 présente le branchement du clavier avec la carte Arduino UNO.

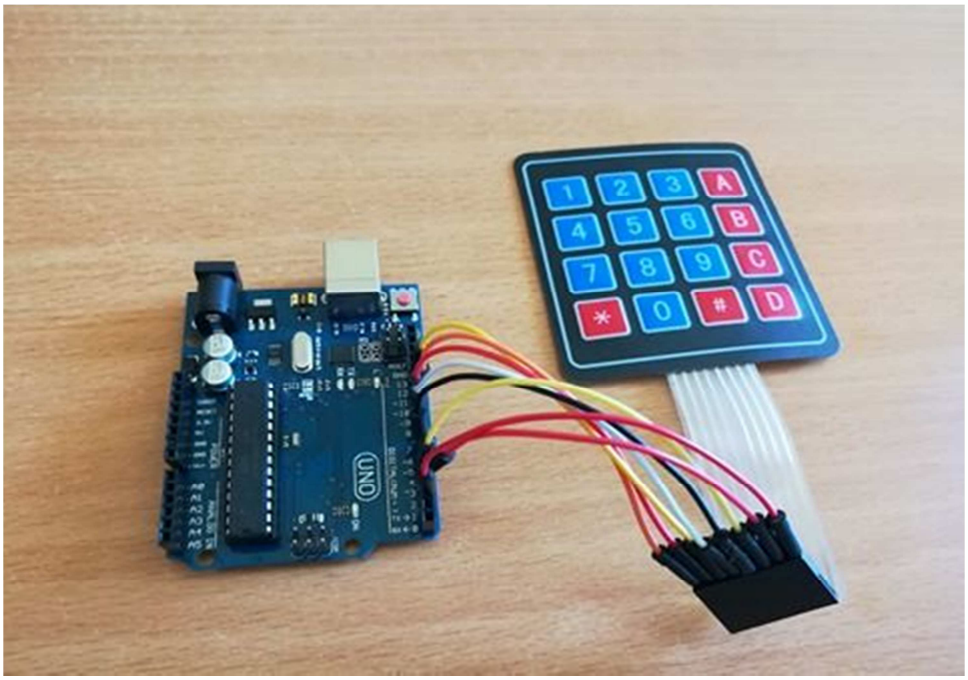


Figure IV.4. Branchement du clavier avec la carte Arduino UNO.

II.5. **Ecran LCD :**

Pour câbler l'écran LCD avec la carte Arduino UNO, nous avons utilisé les broches analogiques (A0, A1, A2, A3, A4, A5) de cette dernière, puisque ces ports peuvent être utilisés comme des ports numériques.

La figure IV.5 présente le branchement de l'écran LCD avec la carte Arduino UNO.

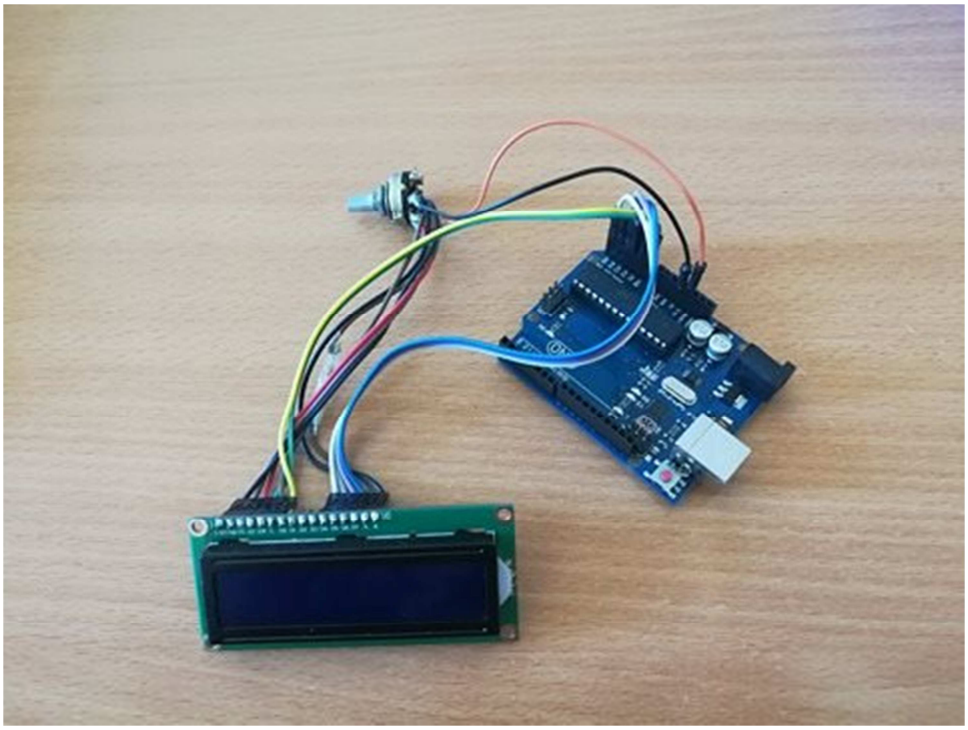


Figure IV.5. Branchement de l'écran LCD avec la carte Arduino UNO.

Les deux fils supplémentaires sont pour VCC (5V) et GND (0V).

III- **Implémentation de l'application :**

La partie logicielle est indispensable pour le fonctionnement de notre système, elle permet de programmer et contrôler chaque composant matériel connecté à notre carte Arduino.

Nous allons décrire et expliquer quelques fonctions du langage Arduino qui permettent de programmer quelques composants.

III.1. **Présentation de quelques fonctions du langage Arduino :**

Pour bien expliquer les fonctions du langage Arduino, nous allons prendre quelques exemples de code Arduino, ensuite on explique le rôle de chaque fonction dans le programme.

III.1.1. **Au niveau de la partie déclarative :**

Inclusion des librairies utilisées : On inclut les librairies des fonctionnalités utilisées :

- Inclusion de la librairie pour l'afficheur LCD alphanumérique :
```
#include <LiquidCrystal.h> // Inclusion de la librairie pour afficheur LCD
```
- Inclusion de la librairie pour le clavier matriciel :
```
#include <Keypad.h> // Inclusion de la librairie pour clavier matriciel
```
- Inclusion de la librairie pour un port série logiciel :
```
#include <SoftwareSerial.h> // Inclusion de la librairie pour le sim900
```

Déclaration des constantes: On déclare les constantes utiles dans le programme :

- Déclaration des constantes utilisées pour l'utilisation du clavier 4x4 :
```
const byte ROWS = 4; // 4 lignes
const byte COLS = 4; //  4colonnes
```
- Déclaration des constantes pour les broches utilisées dans le programme :
```
int buzzer=3; // Buzzer

int irPin=13; // Infrarouge
```
- Déclaration du code d'activation de notre système et le code de désactivation de l'alerte :
```
String Activ="1234"; // code activation alarme
String Desact="1122"; // code désactivation de l'alerte
```

Déclaration des variables :

- Déclaration variables globales des touches du clavier 4x4 :
```
// Position des touches
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
```
- Déclaration variables globales de lignes et de colonnes du clavier 4x4 et de la variable de l'état de l'infrarouge :

```
byte rowPins[ROWS] = {2,4,5,6}; // Broches utilisées pour les lignes
byte colPins[COLS] = {9,10,11,12}; // Broches utilisées pour les colonnes

int etat; // etat du capteur ir
```

Déclarations des objets utiles pour les fonctionnalités utilisées :

- Déclaration d'un objet LCD alphanumérique :
```
LiquidCrystal lcd(A0,A1,A5,A4,A3,A2); // Broches utilisées pour l'afficheur LCD
```

- Déclaration d'un objet clavier matriciel :
```
// Création d'un objet keypad = initialisation clavier
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
// Les broches de lignes sont automatiquement configurées en ENTREE avec pullup interne activé
// Les broches de colonnes sont automatiquement configurées en SORTIE
```

- Déclaration d'un port série logiciel pour la connexion du sim900 :
```
SoftwareSerial SIM900(7, 8); // Broches utilisées pour le sim900
```

III.1.2. **Au niveau de la fonction d'initialisation setup() :**

Initialisation des fonctionnalités utilisées :

- Vitesse de transmission du sim900 :
```
SIM900.begin(19200);    // Vitesse de transmission du SIM900
```

- Vitesse de transmission de l'arduino :
```
Serial.begin(19200);    // Vitesse de transmission de l'Arduino
```

- Initialisation de l'afficheur LCD :
```
lcd.begin(16,2);// Initialise le LCD avec 16 colonnes x 2 lignes
```

Configuration des broches utilisées :

- Les broches de lignes et d'entrée sont configurées automatiquement lors de l'initialisation du clavier
- Configuration des broches en sortie : la configuration des broches du clavier du LCD clavier se fait automatiquement lors l'initialisation.
- Configuration des broches en entrée : la configuration des broches du clavier se fait automatiquement lors l'initialisation.

- La configuration des broches du sim900 se fait automatiquement lors l'initialisation.
- La configuration des broches du buzzer et de l'infrarouge :

```
pinMode(buzzer, OUTPUT); // le Buzzer est défini en sortie
 pinMode(irPin, INPUT); // l'infrarouge est défini en entrée
```

Activation de notre système de surveillance :

On affiche sur l'écran LCD « Code activation », ensuite on saisi le bon code pour activer notre système. Si le code est correcte le système sera activé, sinon le buzzer va faire un bip et on ressaisi à nouveau le code.

Les différentes fonctions utilisées dans ce fragment de code seront bien décrite on ce qui suit.

```
lcd.setCursor(0,0);
 lcd.print("Code activation?");
 Serial.println("Code activation?");
 while (Activ_Alarme()!= Activ){
   tone(buzzer,100);
   delay(100);
   noTone(buzzer);
 }
```

III.1.3. **Au niveau de la boucle principale loop() :**

- On commence par lire l'état de l'infrarouge :

```
etat = digitalRead(irPin);
```

- Ensuite, on teste si une intrusion est détectée :
  - Le buzzer émettra 3 bips sonores :

```
tone(buzzer,100);
delay(100);
noTone(buzzer);
delay(100);
tone(buzzer,100);
delay(100);
noTone(buzzer);
delay(100);
tone(buzzer,100);
```

```
delay(100);
noTone(buzzer);
```

la fonction **tone()** est défini comme suit :

tone (broche, frequence, +/-duree); // produit un son de fréquence indiqué sur la broche.

la fonction **notone()** est défini comme suit :

noTone (broche) ; // stoppe la production de son sur la broche.

- On affiche sur l'écran LCD « code désactiv ? » :

```
lcd.clear();
lcd.setCursor(1,0);
lcd.print("Code desactiv?");
```

- Ensuite, on teste si le code saisi n'est pas correct :

```
tone(buzzer,100);  // Déclenchement de l'alerte.
envoyer_sms("Alerte"); // Envoi d'un sms
lcd.setCursor(0,0);
lcd.print("code desactiv?");
while(Activ_Alarme()!= Desact); //afficher code désactivation
noTone(buzzer);
lcd.clear();
lcd.setCursor(1,0);
lcd.print("alarme desactiv");
```

Sinon afficher « alarme desactive » :

```
lcd.clear();
lcd.setCursor(1,0);
lcd.print("alarme desactive");
```

III.2. **Définition des fonctions utilisées :**

III.2.1. La fonction **Activ_Alarme() :**

Cette fonction nous permet de lire le code saisi sur le clavier pour activer le système de surveillance. A chaque appui sur une touche on affiche « * » sur l'écran LCD. Lors de l'appui sur la touche « # » on sort de la fonction et on récupère le code saisi.

```
1.  String Activ_Alarme(){
2.   String code="";
3.   char key=keypad.getKey();
4.   lcd.Clear();
5.   lcd.setCursor(0,1);
6.    while (key!='#'){
7.      if (key!=NO_KEY){
8.        lcd.print("*");
9.        code+=key;
10.       }
11.     key=keypad.getKey();
12.  }
13.   return code;
14. }
```

- La fonction **getKey()** dans la 3éme ligne retourne la valeur de la clé pressée si elle existe. Elle ne bloque pas le programme.
- La ligne 4 permet d'effacer le contenu de l'écran et mettre le curseur en haut à gauche.
- La ligne 5 nous permet de positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 1.
- La constante **NO_KEY** dans la ligne 7 est retournée par la fonction **getKey()** si aucune touche n'est pressée.
- La ligne 8 permet d'afficher « * » sur l'afficheur LCD.

III.2.2. La fonction **envoyer_sms() :**

```
1.  void envoyer_sms(const char * msg) {
2.     SIM900.print("AT+CMGF=1\r");
3.     delay(100);
4.     SIM900.println("AT+CMGS=\"+213*********\"");
5.     delay(100);
6.     SIM900.println(msg);     // Envoyer le message.
7.     delay(100);
8.     SIM900.println((char)26) ;
9.     delay(100);
10.    SIM900.println();
11.    delay(5000);
12.  }
```

- La ligne 2 permet de saisir la commande AT qui permet d'envoyer un message.
- La ligne 3 permet de saisir le numéro de téléphone du destinataire au format international.
- La ligne 6 permet d'envoyer le message.
- La ligne 8 permet de terminer la commande AT par CTRL+Z dont le code ASCII est 26.

III.2.3. La fonction **Desactiv_Alerte() :**

Cette fonction nous permet de lire le code saisi sur le clavier pour désactiver l'alerte. L'intrus a 10 secondes pour saisir le code correcte sinon l'alerte se déclenche automatiquement. A chaque appui sur une touche on affiche « * » sur l'écran LCD. Lors de l'appui sur la touche « # » on sort de la fonction et on récupère le code saisi.

```
1.  String Desactiv_Alerte(){
2.    String code="";
3.    unsigned long timeInit, timeCour, attente;
4.    char key=keypad.getKey();
5.    lcd.setCursor(0,1);
6.    lcd.print("        ");
7.    lcd.setCursor(0,1);
8.    timeInit=millis();
9.    timeCour=timeInit;
10.   attente=0;
11.   while (key!='#' && attente <10000){
12.     if (key!=NO_KEY){
13.       lcd.print("*");
14.       code+=key;
15.     }
16.     key=keypad.getKey();
17.     timeCour=millis();
18.     attente=abs(timeCour-timeInit);
19.   }
20.  return code;
21. }
```

- la fonction millis() dans la ligne 8 Renvoie le nombre de millisecondes depuis que la carte Arduino a commencé à exécuter le programme courant. Ce nombre débordera (c'est à dire sera remis à zéro) après 50 jours approximativement.
- la fonction abs dans la 19éme ligne calcule la valeur absolue d'un nombre.

Le programme recommence en boucle les instructions de la fonction loop( ).

Conclusion :

Dans ce chapitre, nous avons présenté les différents composants utilisés dans notre système, leurs branchements avec la carte Arduino UNO et l'implémentation de la partie logicielle de notre application.

L'objectif de notre projet était de concevoir et réaliser un système d'alarme anti-intrusion pour surveiller une habitation contre toute intrusion. Après sa mise en marche, on peut considérer que l'objectif est atteint.

Conclusion Générale et Perspectives

Conclusion générale et perspectives :

Dès l'aube de l'humanité, l'homme cherche à se protéger et à protéger ses propriétés contre les risques de cambriolages et de vols.

L'objectif de notre projet était de développer un système d'alarme permettant la protection contre intrusion à des endroits spécifiques, en utilisant la technologie Arduino.

Notre système a été réalisé dans deux aspects : matériel et logiciel. Après sa mise en marche, il a donné une bonne performance en terme de qualité de surveillance.

Ce projet nous a permis de concevoir et réaliser notre premier système embarqué, de nous familiariser avec la technologie Arduino, et aussi d'approfondir nos connaissances en électronique.

En guise de perspectives, notre système peut être amélioré et commercialisé en ajoutant d'autres fonctionnalités enrichissantes telles que :

- Utilisation d'un relais et d'un klaxon pour une alerte sonore plus puissante.
- Utilisation d'une barrière infrarouge avec une distance de captage plus grande ou une barrière laser pour pouvoir surveiller un portail.
- Création d'un réseau de capteur pour surveiller toutes les portes d'accès d'une propriété ainsi que ses fenêtres.
- Ajout d'une caméra de surveillance.
- Ajout d'une alerte par envoi d'appel téléphonique.
- Ajout d'une alerte par l'envoi d'un message électronique (Email) via internet.

Bibliographies:

Ouvrages:

[5] Simon Landrault (Eskimon) et Hippolyte Weisslinger (Olyte), Arduino : Premiers pas en informatique embarqué, 2014.

[7] Simon Landrault (Eskimon) et Hippolyte Weisslinger (Olyte), Arduino : Premiers pas en informatique embarqué, 2014.

[8] Christian Tavernier, Arduino Maitriser sa programmation et ses cartes d'interfaces (Shields), Dunod 2e édition, 2014.

Sites webs:

Les sites de la domotique :

[1] https://sites.google.com/site/domotiquec2i

[2] http://www.easydom.com

[3] http://domotique-2.over-blog.com

Le site de l'université française PARIS-EST-MARNE-LA-VALLE :

[4] http://www-igm.univ-mlv.fr

Les sites de l'Arduino :

[6] www.arduino.cc

[9] http://www.locoduino.org

Le site:

[10]http://www.banggood.com/fr/SIM900-Quad-GSM-GPRS-Shield-Development-Board-For-Arduino-p-964229.html

Annexe

```
//Inclusion des librairies utilisées
#include <Keypad.h>                    // Inclusion de la librairie pour clavier matriciel
#include <LiquidCrystal.h>             // Inclusion de la librairie pour afficheur LCD
#include <SoftwareSerial.h>            // Inclusion de la librairie pour le sim900

// Déclaration d'un port série logiciel pour la connexion du sim900
SoftwareSerial SIM900(7,  8);          // Broches utilisées pour le sim900

//Déclaration d'un objet LCD alphanumérique :
LiquidCrystal lcd(A0,A1,A5,A4,A3,A2);    // Broches utilisées pour l'afficheur LCD

// Déclaration des constantes utilisées
const byte ROWS = 4;       // 4 lignes
const byte COLS = 4;       // 4 colonnes

int buzzer=3;              // Buzzer
int irPin=13;              // Infrarouge

String Activ="1234";       // Code activation alarme
String Desact="1122";      // Code désactivation de l'alerte

// Déclaration des variables utilisées
// Position des touches
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {2,4,5,6};     // Broches utilisées pour les lignes
byte colPins[COLS] = {9,10,11,12};  // Broches utilisées pour les colonnes

int etat; // etat du capteur ir

// Creation du Keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
// Les broches de lignes sont automatiquement configurées en ENTREE avec pullup interne activé
// Les broches de colonnes sont automatiquement configurées en SORTIE

// Les fonctions utilisées
String Activ_Alarme(){
  String code="";
  char key=keypad.getKey();
  lcd.setCursor(0,1);      // Positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 1.
  lcd.print("       ");
  lcd.setCursor(0,1);      // Positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 1.
```

```
  while (key!='#'){
    if (key!=NO_KEY){
      lcd.print("*");     // Afficher sur l'écran LCD "*"
      Serial.print("*");
      code+=key;
    }
    key=keypad.getKey();
  }
  return code;
}

String Desactiv_Alerte(){
  // Déclaration des variables
  String code="";
  unsigned long timeInit, timeCour, attente;
  char key=keypad.getKey();

  lcd.setCursor(0,1);    // Positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 1.
  lcd.print("       ");      // On affiche rien sur l'écran
  lcd.setCursor(0,1);    // Positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 1.

  timeInit=millis();     // Récupérer le nombre de millisecondes depuis que la carte Arduino a
commencé à exécuter le programme courant
  timeCour=timeInit;
  attente=0;

  while (key!='#' && attente <10000){
    if (key!=NO_KEY){    // NO_KEY est renvoyé par  get.Key() si aucune touche n'est pressée
      lcd.print("*");      // Afficher sur l'écran LCD "*"
      code+=key;
    }

    key=keypad.getKey();                  // Récupérer la valeur de la clé pressée
    timeCour=millis();
    attente=abs(timeCour-timeInit);    // Calculer la valeur absolue
  }
  return code;
}

void envoyer_sms(const char * msg) {

  SIM900.print("AT+CMGF=1\r"); // Commande AT pour envoyer un SMS.
  delay(100);

  SIM900.println("AT+CMGS=\"+213********\"") ; // N°  du destinataire, au format international.
  delay(100);
```

```
  SIM900.println(msg);          // Envoyer le message.
  delay(100);

  SIM900.println((char)26);     // Terminer la commande AT par CTRL+Z (code ASCII 26) .
  delay(100);

  SIM900.println();
  delay(5000);                  // Le temps que le message soit envoyé.
}

// La fonction d'initialisation setup()
void setup(){

  SIM900.begin(19200);          // Vitesse de transmission du SIM900
  delay(5000);                  // Attendre 5 seconde
  Serial.begin(19200);          // Vitesse de transmission de l'Arduino

  Serial.println("initialisation");
  lcd.begin(16,2);              // Initialise le LCD avec 16 colonnes x 2 lignes
  lcd.setCursor(0,0);           // Positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 0.
  lcd.print("Code activation?"); // Afficher sur l'écran LCD "Code activation?"

  while (Activ_Alarme()!= Activ){

    tone(buzzer,100);           // Déclenchement du buzzer
    delay(100);                 // Attendre 100 ms
    noTone(buzzer);             // Arreter le buzzer
  }

  pinMode(buzzer, OUTPUT);  // le Buzzer est défini en sortie
  pinMode(irPin, INPUT);        // l'infrarouge est défini en entrée

  lcd.clear();              // Effacer le contenu de l'écran et mettre le curseur en haut à gauche.
  lcd.setCursor(0,1);       // Positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 1.
  lcd.print("alarme activ");   // Afficher sur l'écran LCD "alarme activ"
}

//La boucle principale loop()
void loop(){

  etat = digitalRead(irPin); // Lire l'état de l'infrarouge
  if (etat == LOW){

    // Le buzzer produit 3 bips
    tone(buzzer,100);
    delay(100);
    noTone(buzzer);
```

```
      delay(100);
      tone(buzzer,100);
      delay(100);
      noTone(buzzer);
      delay(100);
      tone(buzzer,100);
      delay(100);
      noTone(buzzer);

      lcd.clear();              // Effacer le contenu de l'écran et mettre le curseur en haut à gauche.
      lcd.setCursor(1,0);  // Positionner le curseur de l'afficheur LCD à la colonne 1 et à la ligne 0.
      lcd.print("Code desactiv?");  // Afficher sur l'écran LCD "code desactiv"

      if (Desactiv_Alerte()!= Desact){

         tone(buzzer,100);            // Déclenchement de l'alerte
         envoyer_sms("Alerte");    // Envoi d'un SMS

         lcd.setCursor(0,0);    // Positionner le curseur de l'afficheur LCD à la colonne 0 et à la ligne 0.
         lcd.print("code desactiv?");    // Afficher "code desactiv?"


         while(Activ_Alarme()!= Desact);
         noTone(buzzer);      // Stopper le buzzer

         lcd.clear();              // Effacer le contenu de l'écran et mettre le curseur en haut à gauche.
         lcd.setCursor(1,0);    // Positionner le curseur de l'afficheur LCD à la colonne 1 et à la ligne 0.
         lcd.print("alarme desactive");  // Afficher sur l'écran LCD "alarme desactive"

      } else {

         lcd.clear();              // Effacer le contenu de l'écran et mettre le curseur en haut à gauche.
         lcd.setCursor(1,0) ;   // Positionner le curseur de l'afficheur LCD à la colonne 1 et à la ligne 0.
         lcd.print("alarme desactive");  // Afficher sur l'écran LCD "alarme desactive"
      }
   }
}
```

Le schéma général de notre système :