République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté de Technologie
Département de Génie Électrique

MÉMOIRE DE FIN DE CYCLE MASTER

En
Automatique

Spécialités

Automatique et Informatique Industrielle
Automatique et Système

Thème

Conception et réalisation d'un système de suivi d'un véhicule basé sur les réseaux GPS et GSM

Présenté par : Mlle ATMANI Hamida
Présenté par : Mlle BENYAHIA Razika

Soutenu le 07 septembre 2020 devant le jury composé de :

Promoteur	**M. HADJI**	**U. A/Mira Béjaia.**
Examinatrice	**Mme. GAGAOUA**	**U. A/Mira Béjaia.**
Examinatrice	**Mme. MEZZAH**	**U. A/Mira Béjaia.**

Béjaia, Septembre 2020.

⁂ *Remerciements* ⁂

Nous allons en premier lieu remercier Dieu de nous avoir donné la santé et la capacité de mener ce travail jusqu'à la fin.

Nous tenons à exprimer nos vifs remerciements à M. HADJI pour son soutien, ses instructions, ses conseils, son temps, son écoute attentive et sa confiance en nous. Merci d'avoir été avec nous tout le temps.

Nous remercions Mme. GAGAOUA et Mme. MEZZAH qui nous ont fait l'honneur de faire partie du jury en tant qu'examinatrices.

Nous remercions chaleureusement nos parents, nos frères et sœurs, nos familles et nos amis.

⁂ *Dédicaces* ⁂

Nous dédions ce travail à notre Promoteur

À nos parents

À nos frères et sœurs

À nos familles

À nos amis.

Table des matières

Table des figures

Notations et symbols

ASCII : American Standard Code for Information Interchange
IDE : Integrated Development Environment
LED : Light Emitting Diode
SD : Secure digital
GPS : Global Positioning System
GSM : Global System for Mobile Communications
PIN : Personal Identification Number
SIM : Subscriber Identification Module
SMS : Short Message Service
MIT : Massachusetts Insitute of Technology
USB : Universal Serial Bus
EEROOM : Elecrically Erasable Programmable Read Only Memory
SPI : Serial Peripheral Interface

Introduction générale

De nos jours, la technologie est devenue part importante dans la vie de l'homme, elle s'est développer d'une manière exponentielle et touche un vaste nombre de domaines. L'évolution de la technologie GPS (système de positionnement global) est en plein essor. En effet, elle a donné naissance à plusieurs services, notamment le service de la Géolocalisation dans une zone isolée. Ce dernier offre diverses possibilités dans la localisation et le positionnement sur une carte géographique pour afficher le mouvement des différents objets équipés d'un récepteur GPS.

Grâce à ce procédé, on a réussi à mettre au point un système de suivi de véhicule nommé «Trackeur», afin d'accéder à des informations sur le lieu et l'endroit précis où se trouve le véhicule à localiser sur une carte géographique.

Ce rapport présente le déroulement du projet et permet de suivre la progression de notre travail ainsi que les résultats obtenus et les améliorations possibles.

L'objectif principal de ce projet est de réaliser un système de suivi de véhicule avec un microcontrôleur Arduino uno et un système de positionnement mondial (GPS) et d'un système mondial de communication mobile (GSM) ; pour y aboutir, nous avons réparti le travail de la manière suivante :

Le premier chapitre sera consacré à une étude approfondie sur l'Arduino. On mettra la lumière sur un modèle de base qui est l'Arduino UNO, son environnement de programmation, aussi les capteurs et actionneurs qu'elle supporte et son principe de fonctionnement afin de simplifier son utilisation.

Le deuxième chapitre illustre les différentes réalisations et les outils utilisés tel que le module GPS qui en est l'élément principal, ainsi que le module GSM permettant à l'utilisateur de localiser le véhicule via SMS. Enfin visualiser la position du véhicule sur une carte géographique.

Le troisième chapitre est consacré à la présentation de l'application Android «Trackeur_app». Nous présentant en premier lieu une aperçue générale sur le langage de programmation MIT, et dans le deuxième lieu la réalisation et la programmation de l'application sous MIT afin de simplifier la commande.

Le quatrième chapitre présente la conception de notre système de suivi de véhicule en décrivant

les étapes à suivre pendant la réalisation, telles que l'algorithme et le code source Arduino. Enfin, nous terminons avec une conclusion présentant les perspectives pour améliorer le projet et une bibliographie.

Chapitre I

Étude théorique

I.1 Introduction

L'Arduino est une carte de circuit imprimé en matériel libre supportant des composants électroniques tel qu'un microcontrôleur, et de différentes entrées/sorties analogiques et numériques (voir figure I.1). Elle peut être programmée pour analyser et produire des signaux électriques de manière à effectuer des taches (simuler ou créer des systèmes automatisés).

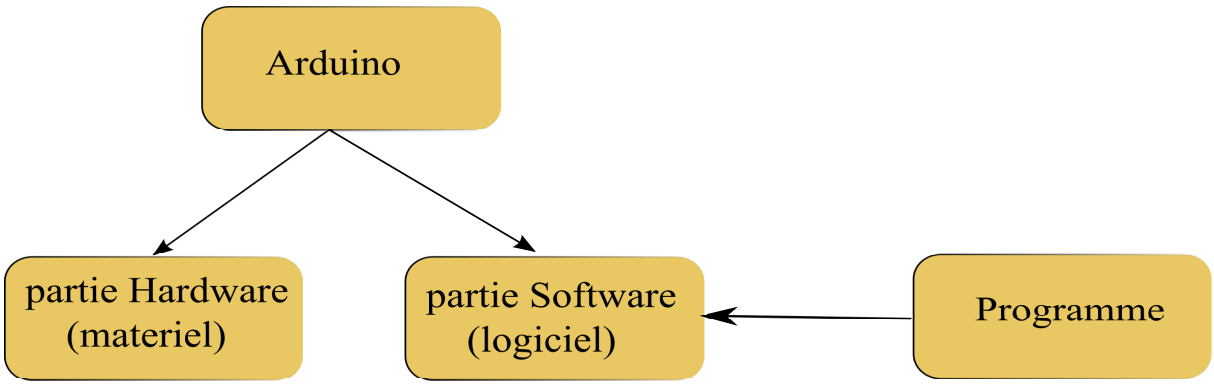

FIGURE I.1 – Schéma explicatif des parties de l'Arduino.

I.2 Pourquoi Arduino ?

Il y a de nombreuses cartes électroniques qui possèdent des plateformes basées sur des microcontrôleurs disponibles pour l'électronique programmée. Tous ces outils prennent en charge les détails compliqués de la programmation, et les intègrent dans une présentation facile à utiliser. De la même façon, le système Arduino simplifie la façon de travailler avec les microcontrôleurs, tout en offrant aux personnes intéressées plusieurs avantages par rapport à d'autres systèmes :

- **Peu coûteux :** les cartes Arduino sont relativement peu coûteuses par rapport aux autres plates-formes. La version là moins chère du module Arduino peut être assemblée à la main.
- **Multiplateforme :** Le logiciel Arduino (IDE) fonctionne sur les systèmes d'exploitation Windows, Macintosh OSX et Linux. La plupart des systèmes de microcontrôleurs sont limités à Windows.
- **Environnement de programmation simple et clair :** Le logiciel Arduino (IDE) est facile à utiliser pour les débutants, mais suffisamment flexible pour que les utilisateurs avancés en profitent également.
- **Logiciel Open Source et extensible :** Le logiciel Arduino est publié en tant qu'outils open source, disponibles pour l'extension par des programmeurs expérimentés.
- **Matériel extensible :** Les plans des cartes Arduino sont publiés sous une licence «Creative Commons», afin que les concepteurs de circuits expérimentés puissent créer leur propre version du module, l'étendre et l'améliorer. Même les utilisateurs relativement inexpérimentés peuvent créer la version maquette du module afin de comprendre comment il fonctionne et économiser de l'argent [1].

I.3 Partie Hardware (matériel)

La carte Arduino repose sur un circuit intégré associé à des entrées et sorties qui permettent à l'utilisateur de brancher différents types d'éléments externes.

I.3.1 Les différentes cartes

Actuellement, il existe plus de 20 versions de module Arduino, nous citons quelques-unes 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 programmable à l'aide du port USB cette version utilisant un microcontrôleur ATmega168 (ATmega328 pour une plus nouvelle version).
- LilyPad Arduino, 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'Arduino Diecimila, avec une interface d'USB et utilise un microcontrôleur ATmega168.
- L'Arduino Duemilanove ("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'Arduino Mega, en utilisant un microcontrôleur ATmega1280 pour I/O additionnel et mémoire.
- L'Arduino UNO, utilisations d'un microcontrôleur ATmega328.
- L'Arduino Mega2560, utilisations d'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'Arduino Esplora : ressemblant à un contrôleur visuel de jeu, avec un manche et des sondes intégrées pour le bruit, la lumière, la température, et l'accélération [2].

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

I.3.2 Architecture de la carte ArduinoUNO

Cette carte est basée sur un microcontrôleur ATmega 328 et des composants complémentaires (voir figure I.2). Pour pouvoir l'utiliser et se lancer, il suffit simplement de la connecter à un ordinateur à l'aide d'un câble USB (ou de l'alimenter avec un adaptateur secteur ou une pile, mais ceci n'est pas indispensable, l'alimentation étant fournie par le port USB) [2]. Les signaux

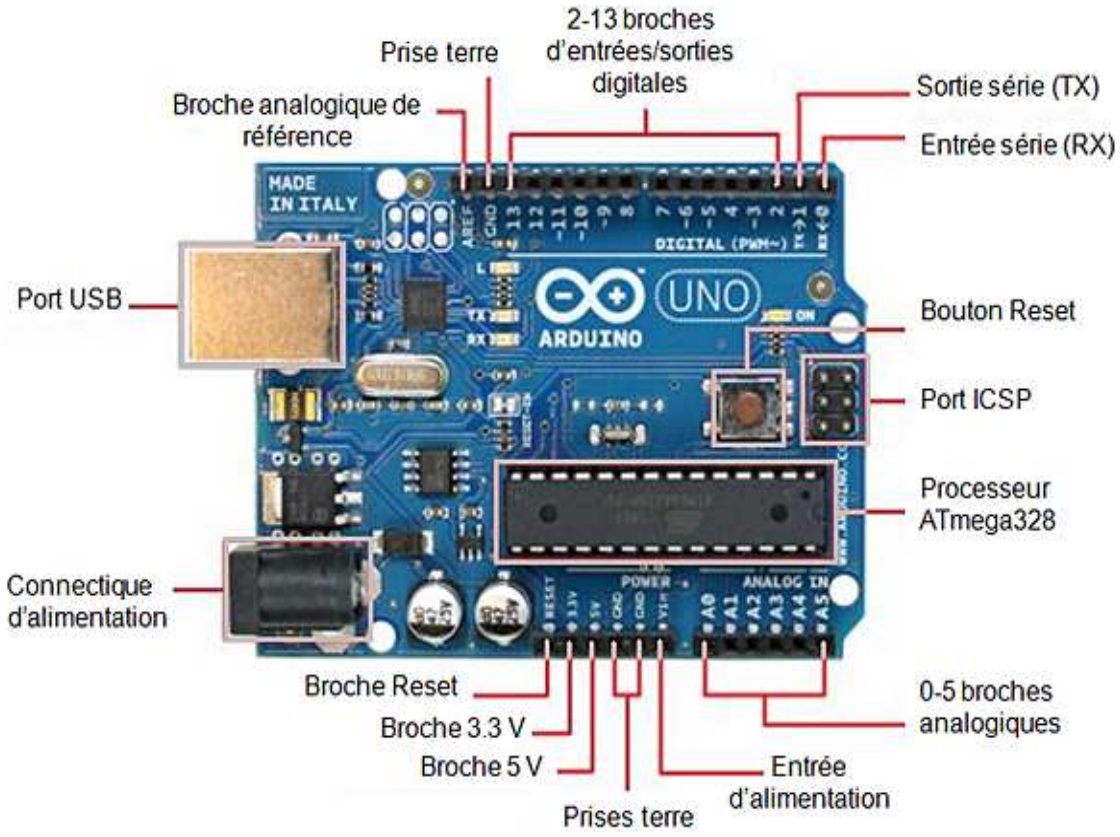

FIGURE I.2 – la carte arduino UNO [1].

d'entrée-sortie du microcontrôleur sont reliés à des connecteurs selon le schéma ci-dessous (voir figure I.3) :

I.3.3 Caractéristiques de la carte ARDUINO UNO

- Microcontrôleur : ATmega328
- Tension d'alimentation interne = $5V$
- Tension d'alimentation (recommandée)= 7 à $12V$, limites = 6 à $20V$
- Entrées/sorties numériques : 14 dont 6 sorties PWM
- Entrées analogiques = 6
- Courant max par broches E/S = $40mA$
- Courant max sur sortie $3,3V$ = $50mA$
- Mémoire Flash $32KB$
- Mémoire SRAM $2KB$

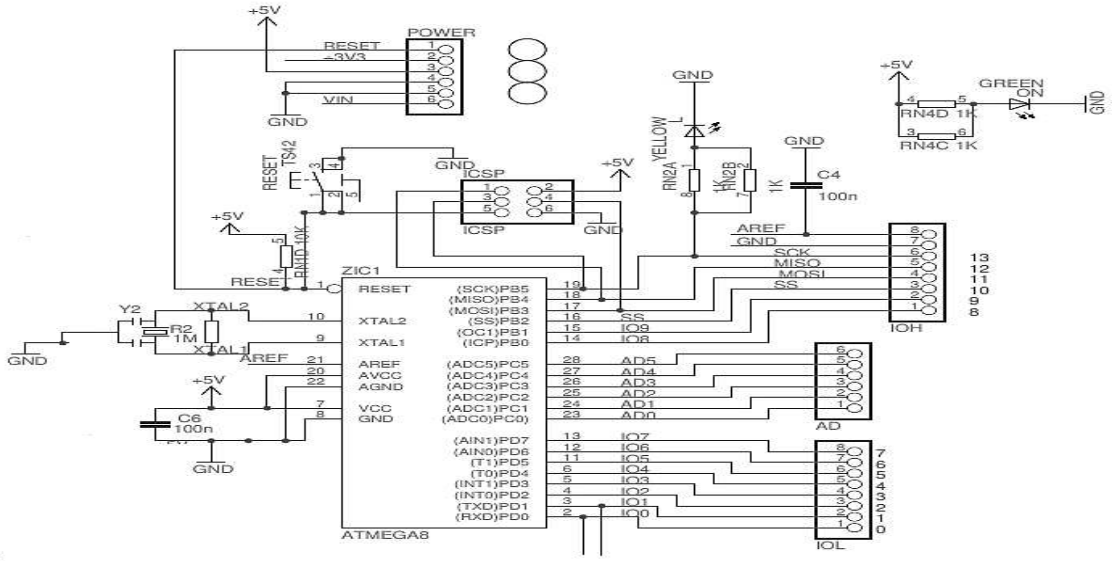

FIGURE I.3 – schéma de la carte arduino UNO [2]

- Mémoire EEPROM $1KB$
- Fréquence horloge = $16MHz$
- Dimensions = $68.6mmx53.3mm$ [3]

I.3.4 Le Microcontrôleur ATMega328

Un microcontrôleur ATMega328 est un circuit intégré qui rassemble sur une puce plusieurs éléments complexes dans un espace réduit. Aujourd'hui, en soudant un grand nombre de composants encombrants ; tels que les transistors, les résistances et les condensateurs ; tout peut être logé dans un petit boîtier en plastique noir muni d'un certain nombre de broches dont la programmation peut être réalisée en langage C (voir figure I.4) [3].

Le microcontrôleur ATMega328 est constitué d'un ensemble d'éléments qui ont chacun une fonction bien déterminée. L'architecture interne de ce circuit programmable se compose essentiellement de :

- Nombre de broches : 28
- Mémoire Flash : $32ko$ (programmable par interface série)
- Mémoire Données EEPROM : $1ko$
- Mémoire RAM : $2ko$

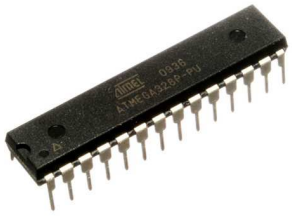

FIGURE I.4 – Microcontrôleur ATMega328

- 32 registres de travail d'accès rapide pour l'ALU
- Ports parallèles : 3, avec 23 broches E/S
- Fréquence d'horloge : $16Mhz$ (maxi tolérée = $20Mhz$)
  - Donc : 16 cycles d'horloge par microseconde
- Périphériques internes
  - 6 convertisseur Analogique/Numérique 10 bits, comparateur analogique
  - 1 timer 16 bits (T1), 2 timers 8 bits (T0, T2)
  - 6 canaux PWM, 1 chien de garde (watchdog)
  - SPI, USART, TWI (=I2C)
- 26 interruptions
- 5 modes d'économie d'énergie [3]

Brochage du Microcontrôleur ATMéga328 :

I.3.4.1 Les entrées/sorties

A - Entrées/sorties numériques

Chacune des 14 broches numériques de la carte UNO (numérotées de 0 à 13) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions **pinMode()**, **digitalWrite()** et **digitalRead()** du langage Arduino. Ces broches fonctionnent en $5V$. Chaque broche peut fournir ou recevoir un maximum de $40mA$ d'intensité et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de $20-50KOhms$. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction **digitalWrite** (broche, HIGH) [4].

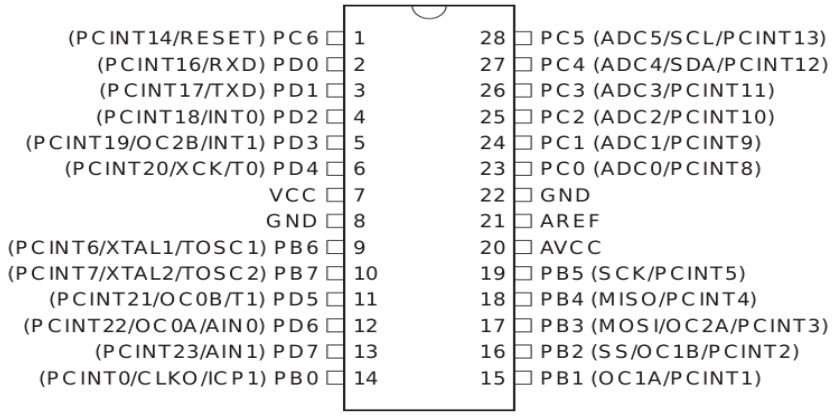

FIGURE I.5 – Brochage du Microcontrôleur ATMéga328 [3].

De plus, certaines broches ont des fonctions spécialisées :

- **Communication Série :**

  Une broche 0(RX) est utilisée pour recevoir et une broche 1(TX) pour transmettre les données série du niveau TTL. Ces broches sont connectées aux broches correspondantes du circuit intégré programmé en convertisseur USB-vers-série de la carte.

- **Interruptions Externes :**

  Broches 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur.

- **Impulsion PWM (largeur d'impulsion modulée) :**

  Broches 3, 5, 6, 9, 10, et 11. Fournissent une impulsion PWM 1-bits à l'aide de l'instruction **analogWrite()**.

- **SPI (Interface Série Périphérique) :**

  Broches 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches supportent la communication SPI (Interface Série Périphérique) disponible avec la librairie pour communication SPI.

- **I2C :**

  Broches 4 (SDA) et 5 (SCL). Supportent les communications de protocole I2C (ou interface

TWI (TwoWire Interface - Interface "2 fils"), disponible en utilisant la librairie Wire/I2C (ou TWI - Two-Wire interface - interface "2 fils").

- **LED :**

  Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

B - Entrées/Sorties analogiques

La carte UNO dispose de 6 entrées analogiques (numérotées de 0 à 5), chacune pouvant fournir une mesure d'une résolution de $10bits$ (c-à-d. sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction **analogRead()** du langage Arduino. Par défaut, ces broches mesurent entre le $0V$ (valeur 0) et le $5V$ (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction **analogReference()** du langage Arduino. La carte Arduino UNO intègre un fusible qui protège le port USB de l'ordinateur contre les surcharges en intensité (le port USB est généralement limité à $500mA$ en intensité). Bien que la plupart des ordinateurs aient leur propre protection interne, le fusible de la carte fournit une couche supplémentaire de protection. Si plus de $500mA$ sont appliqués au port USB, le fusible de la carte coupera automatiquement la connexion jusqu'à ce que le court-circuit ou la surcharge soit stoppé [4].

Autres broches :

- **AREF :** Tension de référence pour les entrées analogiques (c.-à-d. la valeur utilisée comme haut de plage d'entrée). Utilisée avec l'instruction **analogReference()**.
- **Reset :** Mettre cette broche au niveau BAS entraîne la réinitialisation (le redémarrage) du microcontrôleur. Typiquement, cette broche est utilisée pour ajouter un bouton de réinitialisation sur le circuit qui bloque celui présent sur la carte.

I.3.4.2 Les sources d'alimentation

La carte Arduino UNO peut-être alimentée soit via la connexion USB (qui fournit $5V$ jusqu'à $500mA$) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte.

- **VIN.** La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du $5V$ de la connexion USB ou autre source $5V$ régulée). On peut alimenter la carte à l'aide de cette broche, ou, si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.
- **5V.** La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte. Les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino. Le $5V$ régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du $5V$ régulé) ou de tout autre source d'alimentation régulée.
- **3V3.** Une alimentation de $3.3V$; Ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu de $5V$ [2].
- **GND.** Broche de masse (ou $0V$).

I.3.5 Les modules

I.3.5.1 Les capteurs

Les capteurs sont des composants matériels, qui une fois correctement connectés à votre carte Arduino, peuvent fournir des informations sur le monde extérieur. Ceux-ci ont comme rôle de saisir une grandeur physique et de la convertir en information numérique.

A - Les capteurs logiques ou TOR (tout ou rien)

Ce type de capteur renvoie une information de type logique c'est-à-dire 0 ou 1. En d'autres termes, la sortie peut prendre uniquement deux états. Par exemple, un clavier est une matrice de capteurs logiques car chaque touche est un interrupteur (ou un contact) qui peut être soit ouvert soit fermé [4].

B - Les capteurs analogiques

Ce type de capteur renvoie une valeur proportionnelle à la grandeur physique mesurée. Par exemple une photo-résistance (capteur de lumière) est un capteur qui convertit la luminosité en va-

leur électrique. Il existe des capteurs analogiques qui renvoient une information linéaire et d'autres dont la réponse suit une courbe différente.

I.3.5.2 Les actionneurs

Les actionneurs sont des composants matériels qui, une fois correctement connectés à votre carte Arduino, permettent d'agir sur le monde extérieur. Ceux-ci ont comme rôle de convertir une valeur électrique en action physique. Un actionneur engendre un phénomène physique à partir de l'énergie qu'il reçoit comme par exemple : Lumière à partir d'un courant électrique (diode électroluminescente, lampe...). Mouvement à partir d'un courant électrique (moteur électrique) [6].

I.4 Partie software (logiciel)

I.4.1 Environnement de développement (IDE)

Un langage de programmation est un langage permettant à un être humain d'écrire un ensemble d'instructions (code source) qui seront directement converties en langage machine grâce à un compilateur (c'est la compilation). L'exécution d'un programme Arduino s'effectue de manière séquentielle, c'est-à-dire que les instructions sont exécutées les unes à la suite des autres. Le langage utilisé par le logiciel Arduino pour programmer le microcontrôleur est basé sur les langages C/C++ [5].

I.4.2 La structure de base d'un programme Arduino

Un programme Arduino est composé de 3 parties, même si l'environnement de développement de l'Arduino (IDE) n'en fait apparaître que deux lors de la création d'un nouveau croquis (programme) :

```
 void setup() {
 //fonction executée à l'initialisation de la carte
}

\noindent void loop() {
 //fonction executée en boucle infinie
}
```

Un programme Arduino est composé de 3 parties, même si l'environnement de développement de l'Arduino (IDE) n'en fait apparaître que deux lors de la création d'un nouveau croquis (programme) :

- Inclusion des bibliothèques et déclaration des variables.
- **Setup () :** l'initialisation des variables et la description d'action qui ne seront exécutées qu'une seule fois au lancement du programme.
- **Loop () :** la fonction Loop () est une boucle infinie et les instructions qu'elle contient seront exécutés à chaque cycle du processeur.

Fonctionnement de la structure de base du programme

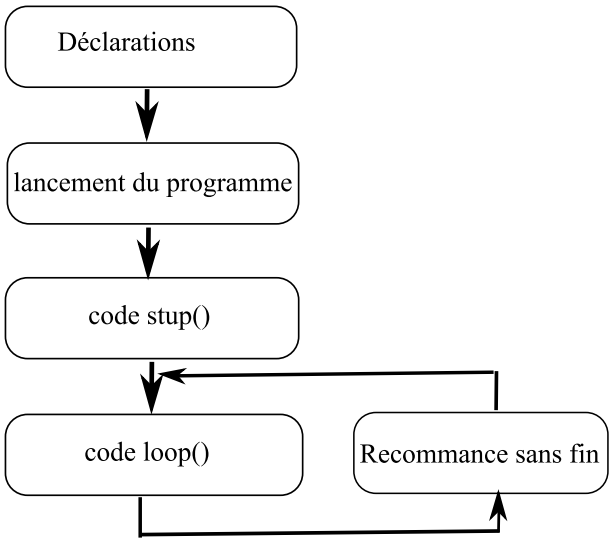

FIGURE I.6 – Fonctionnement de la structure de base du programme

I.4.3 Le moniteur série

Le moniteur série intégré de l'environnement Arduino assure une communication entre l'ordinateur et la carte Arduino. Le lancement du Moniteur série se fait simplement à partir de la barre d'outils en sélectionnant le même débit en bauds que celui configuré dans le programme (voir figure I.7) [5].

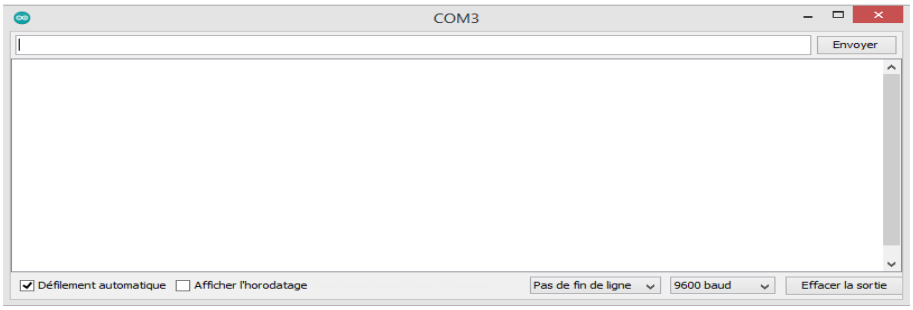

FIGURE I.7 – Interface du moniteur série

I.5 Représentation de la plateforme Fritzing

Fritzing est un logiciel open-source multiplateforme permettant de construire des schémas et des circuits électroniques que nous utilisons avec Arduino (voir figure I.8). Ce logiciel comporte trois vues principales :

- **La platine d'essai :** ou l'on voit les composants tel qu'ils sont dans la réalité et où l'on construit le montage.
- **La vue schématique :** représente le schéma fonctionnel du circuit.
- **Le circuit imprimé :** représente la vue du circuit imprimé tel qu'il sera sorti en PDF pour être imprimé.

Parmi les composants proposés par défaut sur Fritzing, on peut citer :

- Les composants électroniques standards (résistance, diode, transistor, etc...).
- Les circuits intégrés logiques simples.
- Les capteurs les plus courants
- Les composants de sorties les plus courants (LEDs, écran LCD, servomoteur, relais, etc...)
- Différents types d'alimentations.
- Les connecteurs les plus courants (USB, jack, microSD, etc...).
- Les différents types des cartes Arduino et microcontrôleurs [7].

I.6 Exemple LED clignotante

Le but c'est de Piloter une LED avec une carte Arduino UNO et de la faire clignoter à une vitesse d'un clignotement par seconde.

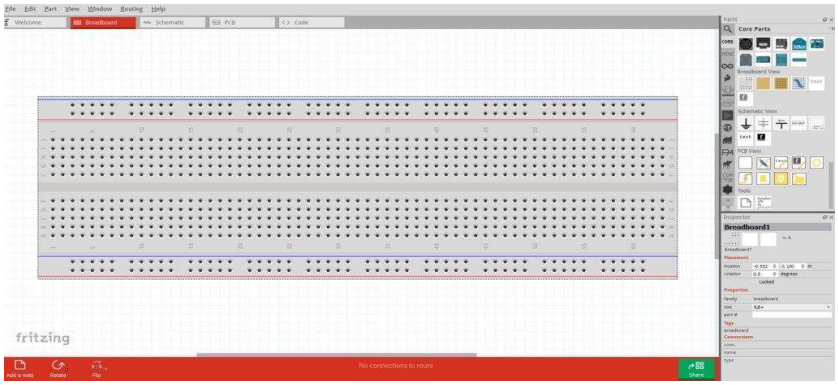

FIGURE I.8 – Plateforme de Fritzing [7]

I.6.1 Montage électronique de la LED

Pour qu'une LED s'allume il faut obligatoirement relier sa patte (−) à la borne (−) de l'alimentation et sa patte (+) à la borne (+) de l'alimentation. Si la LED est branchée à l'envers elle ne s'allumera pas (le courant ne la traversera pas).

De plus elle ne doit pas être traversée par un courant trop fort, c'est pour cela qu'il est indispensable de brancher une résistance en série avec la LED. On notera qu'une résistance n'a pas de sens de branchement (voir figure I.9).

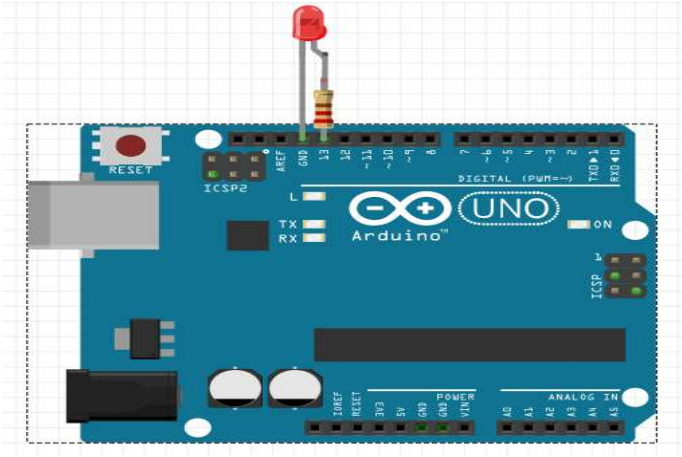

FIGURE I.9 – Diagramme de câblage

I.6.2 Programmation

```
void setup(){
pinMode(13, OUTPUT);
```

```
}
void loop(){
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```

A - Explications du programme

Le code est composé de deux fonctions. La première fonction est définie par :

voidsetup() {

pinMode (13, OUTPUT) ;

}

Son nom est setup. Le mot clé void indique que la fonction ne retourne pas de résultat. Le corps de la fonction est constitué d'une seule instruction qui règle la borne numérique numéro 13 de la carte Arduino en mode sortie (OUTPUT), c'est à dire qu'il sera possible de programmer sur cette borne l'absence ($0V$) ou la présence ($+5V$) de courant.

La fonction setup est automatiquement exécutée une seule fois lorsque le programme démarre. C'est dans cette fonction qu'il faut mettre toutes les initialisations nécessaires au fonctionnement du programme. La seconde fonction est définie par :

voidloop() {

digitalWrite (13, HIGH) ;

delay(1000) ;

digitalWrite (13, LOW) ;

delay (1000) ;

}

Elle s'appelle loop et comme son nom l'indique cette fonction s'exécute indéfiniment en boucle. Pour obtenir le clignotement de la LED il faut utiliser l'instruction digitalWrite qui permet de commander la présence ou non de courant sur une borne numérique de la carte Arduino [2].

Fonctions de manipulations des entrées/sorties numériques :

- **void pinMode() :** Cette fonction sert à définir une broche spécifique de l'Arduino comme

étant une entrée ou une sortie en utilisant les constantes : INPUT et OUTPUT respectivement.

- **void digitalWrite() :** sert à définir l'état d'une sortie en utilisant les constantes : HIGH(1 logique = $5V$) ou LOW (0 logique = $0V$).
- **digitalRead () :** sert à lire l'état d'une broche. Si celle-ci est reliée à un potentiel $5V$, le résultat de la lecture sera HIGH (1) ou LOW (0) si elle est reliée à un potentiel nul.
- **Void delay () :** La fonction Delay (ms) sert à effectuer une pause d'une certaine durée fixée par un entier passé dans la fonction à son appel. Cette fonction est une fonction bloquante, et donc pendant qu'elle se déroule, les opérations de manipulation des ports ne sont pas possibles.

I.7 Conclusion

Dans ce chapitre nous avons projeté la lumière sur une carte d'acquisition qui est l'Arduino UNO. Nous avons expliqué les deux parties essentielles de l'Arduino UNO (partie matériel et partie programmation) et nous avons cité un exemple de fonctionnent d'une LED.

Dans le prochain chapitre, nous allons présenter les différents outils utilisés pour réaliser notre système.

Chapitre II

Etude du système

II.1 Introduction

L'objectif de ce chapitre est d'expliquer les différentes étapes : l'étude et le choix des composants électroniques nécessaires pour le fonctionnement de notre système ainsi que des tests sur la plaquette d'essai.

II.2 Schéma de principe

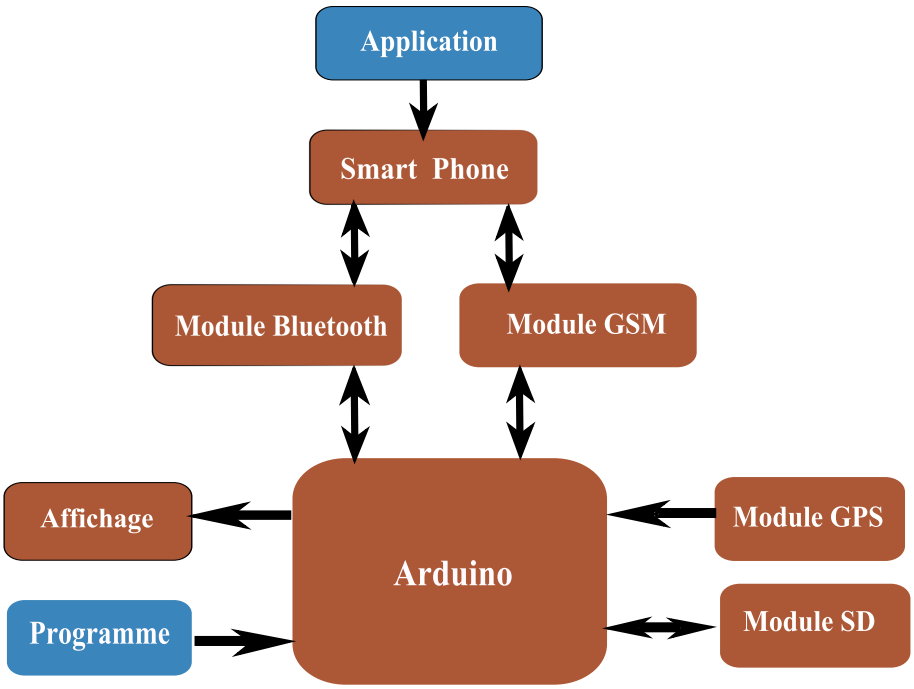

FIGURE II.1 – Schéma de principe.

L'Arduino UNO est utilisé pour contrôler tout le processus avec un module GPS NEO-6M et

un module GSM SIM800L. Le module GPS est utilisé pour détecter les coordonnées du véhicule, le module GSM pour envoyer ces coordonnées à l'utilisateur. Et un module SD pour stocker les coordonnées dans la mémoire. Ainsi qu'un module Bluetooth qui a pour but de contrôler le Marche/Arrêt du système.

Le suivi du véhicule se fait en deux méthodes : la première est la localisation sous demande qui consiste à envoyer les coordonnées GPS qu'en cas de demande de l'utilisateur par SMS via module GSM. La deuxième méthode est le «tracking» qui consiste à identifier et enregistrer périodiquement la position du véhicule dans un fichier TXT sur la carte SD. Dans les deux cas on va récupérer les données transmises depuis la carte Arduino vers l'appareil Android en utilisant une application mobile qui va nous permettre d'afficher l'emplacement du véhicule sur une carte géographique. Le système peut être installé dans le véhicule et l'alimenter.

II.3 Les modules utilisés

II.3.1 Module GPS NEO-6M

Il s'agit d'un module GPS simple doté d'une antenne, d'un adaptateur, d'une batterie et du port d'extension d'antenne sur une carte de circuit imprimé (voir figure II.2). Le module GPS permet de connaître votre position en temps réel, il est utilisé pour détecter la latitude et la longitude de n'importe quel endroit sur terre. Il communique avec l'Arduino via un port série [8].

FIGURE II.2 – MODULE GPS NEO-6M [8].

II.3.1.1 Caractéristiques du module GPS NEO-6M

- Un module GPS complet avec une antenne active intégrée et une EEPROM intégrée pour enregistrer les données des paramètres de configuration.
- L'antenne active en céramique intégrée de $25*25*4mm$ offre une forte capacité de recherche par satellite.
- Équipé de voyants d'alimentation et de signal et d'une batterie de sauvegarde de données.
- Alimentation : $3-5V$; Débit en bauds par défaut : 9600 bps.
- Interface : RS232 TTL [8].

II.3.1.2 Fonctionnement des récepteurs GPS

Les récepteurs GPS fonctionnent en fait en déterminant à quelle distance ils se trouvent d'un certain nombre de satellites. Ils sont préprogrammés pour savoir où se trouvent les satellites GPS à tout moment.

Les satellites transmettent des informations sur leur position et l'heure actuelle sous forme de signaux radio vers la Terre. Ces signaux identifient les satellites et indiquent au récepteur où ils se trouvent.

Le récepteur calcule ensuite la distance de chaque satellite en déterminant combien de temps il a fallu pour que les signaux arrivent. Une fois qu'il a des informations sur la distance d'au moins trois satellites et où ils se trouvent dans l'espace, il peut localiser votre position sur Terre.

Souvent le GPS prend du temps pour se connecter avec le réseau en raison de mauvaises conditions météorologiques. Pour que le GPS fonctionne correctement, il doit avoir une vue dégagée du ciel.

II.3.1.3 Communication avec la carte

Le module GPS NEO-6M dispose de 4 broches au total : VCC, RX, TX et GND, qui le connectent au monde extérieur. Il communique avec l'Arduino UNO via une communication série à l'aide des broches TX (émetteur) et RX (récepteur) (voir figure II.3).

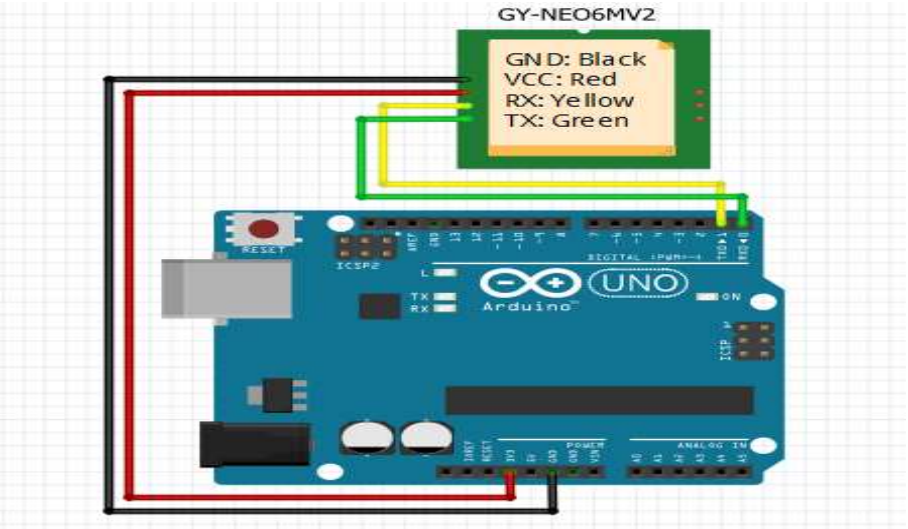

FIGURE II.3 – Câblage du module GPS NEO-6M avec Arduino UNO.

II.3.1.4 Bibliothèque TinyGPS++

Cette bibliothèque effectue beaucoup de tâches lourdes nécessaires pour recevoir des données du module GPS, telles que la lecture et l'extraction de données utiles en arrière-plan [6].

II.3.1.5 Affichage des coordonnées GPS

Programme :

```
#include<TinyGPS++.h>
#include<SoftwareSerial.h>

char data;
double latitude, longitude, vitesse ;

TinyGPSPlus gps;

void setup(){
Serial.begin(9600);
}
void loop(){
while (Serial.available())
{ data = Serial.read();
gps.encode(data);// Lire les données GPS;
if (gps.location.isUpdated()) //Si les données  changent afficher les nouvelles
données
{latitude = gps.location.lat();
longitude = gps.location.lng();
vitesse = gps.speed.kmph();
Serial.println("--------DONNEES GPS----------");
Serial.print("LATITUDE="); Serial.println(latitude);
```

```
Serial.print("LONGITUDE="); Serial.println(longitude);
Serial.print("VITESSE (km/h)="); Serial.println(vitesse);
}}}
```

Explication du programme :

Le programme commence par inclure la bibliothèque TinyGPS ++ et la bibliothèque série logicielle. La création de TinyGPSPlusobjet aidera à accéder aux fonctions spéciales liées à la bibliothèque. Ensuite la déclaration des variables.

```
#include<TinyGPS++.h>
#include<SoftwareSerial.h>

char data;
double latitude, longitude, vitesse ;

TinyGPSPlus gps;
```

Dans la fonction setup nous devons initier la communication série avec le PC ainsi qu'avec le module GPS.

```
Serial.begin(9600);
```

La boucle loop est l'endroit où on demande des informations. Pour que TinyGPS ++ fonctionne, on doit à plusieurs reprises canaliser les caractères vers celui-ci à partir du module GPS en utilisant le encode() méthode.

```
while (Serial.available())
{ data = Serial.read();
gps.encode(data);
```

Ensuite, on interroge le GPS objet pour voir si des champs de données ont été mis à jour et les afficher sur le moniteur série.

On obtient la latitude et la longitude ainsi que la vitesse en utilisant gps.location.lat (), gps.location.lng (), et vitesse = gps.speed.kmph() ; respectivement.

```
if (gps.location.isUpdated()) // Si les données changent afficher les nouvelles
données
{latitude = gps.location.lat();
longitude = gps.location.lng();
vitesse = gps.speed.kmph();
Serial.println("--------DONNEES GPS----------");
Serial.print("LATITUDE="); Serial.println(latitude);
Serial.print("LONGITUDE="); Serial.println(longitude);
Serial.print("VITESSE (km/h)="); Serial.println(vitesse);
}
```

Résultat :

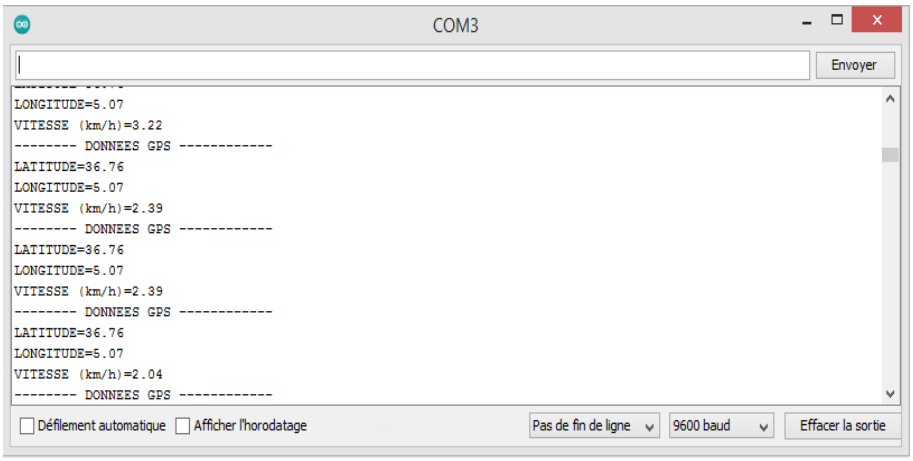

FIGURE II.4 – Affichage sur le moniteur série

II.3.2 Module GSM SIM800L

Le module SIM800l permet au microcontrôleur de communiquer en utilisant le réseau GSM en envoyant des commandes AT à l'UART (le port Série RS232), avec ce module on peut contrôler le microcontrôleur juste avec l'envoie d'un SMS, comme il peut nous informer sur l'état de notre système en nous envoyant un SMS. Il est compatible avec les microcontrôleurs tels l'ATméga (voir figure II.5) [9].

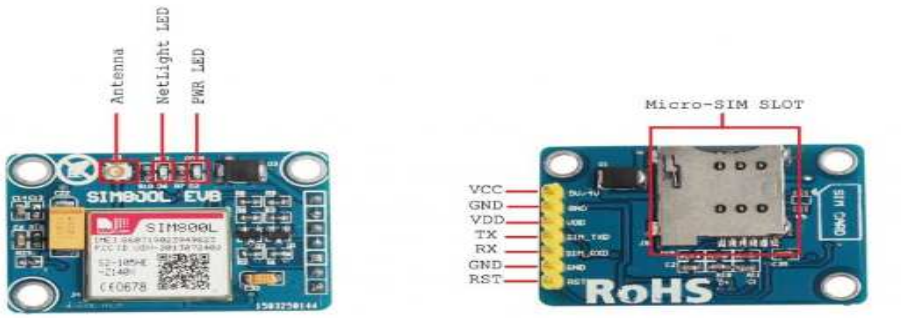

FIGURE II.5 – Module GSM SIM800L EVB [9].

GSM Antenne :

Communications GSM dépendent des antennes. L'antenne permet l'envoie et la réception des

signaux de communication. L'antenne que nous avons utilisée dans notre projet fournit un fonctionnement à la fois GSM quadri-bande Fréquences avec un gain $+2dBi$. Cette antenne fonctionne en Quad Band 890/960, 1710/1880MHz Fréquences et elle est omnidirectionnelle (voir figure II.6) [10].

FIGURE II.6 – Antenne GSM [10].

II.3.2.1 Description du brochage

Il y a 7 broches au total sur le SIM800L, que nous allons utiliser pour s'interfacer avec Arduino.

VCC : entrée de tension d'alimentation externe pour SIM800L

GND : Terre externe pour SIM800L

VDD : MicroContrroller Entrée de tension d'alimentation pour SIM800L

RST : réinitialiser la broche pour SIM800L

RXD : communication série (broche du récepteur)

TXD : communication série (broche de transfert) [9]

II.3.2.2 Connexion du module

Ce module communique via une liaison série avec une carte Arduino. Cette liaison s'établit sur deux broches RX et TX définies dans notre programme en tant que broches 3 et 2, VCC à $5v$ (voir figure II.7).

Une fois alimenté, le voyant LED du SIM800L clignote une fois toutes les 2 ou 3 secondes lorsqu'il a complètement enregistré notre Sim sur un réseau, lorsque le voyant LED clignote une

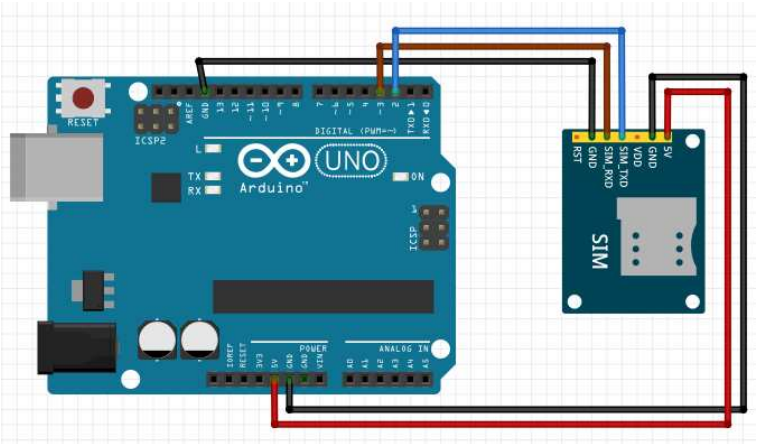

FIGURE II.7 – Connexion du module GSM SIM800L

fois toutes les secondes, cela signifie que le SIM800L est toujours à la recherche d'un réseau sur lequel s'inscrire. Lorsque le voyant LED ne clignote pas, cela signifie que l'alimentation fournit un faible courant.

II.3.2.3 Les Commandes AT

Le microcontrôleur commande le module GSM en utilisant des commandes AT. Les commandes AT (ou commandes de Hayes) constituent un langage de commande développé afin de commander les modems. Ce jeu de commandes s'est ensuite retrouvé dans tous les modems produits. Toutes les commandes doivent être précédées de AT [10].

Suivant le cahier des charges, on aura besoin des commandes AT suivantes :

ATD : Envoi d'un appel vocal ou autre.

ATA : Décroche la ligne lors d'un appel en cours.

AT+CMGF : Sélection du format des messages.

AT+CMGS : Envoi de messages.

AT+CMGR : Lecture des messages.

II.3.2.4 Envoi de SMS

Programme :

```
#include<SoftwareSerial.h>
```

```
SoftwareSerialmySerial(3, 2);
void setup()
{
Serial.begin(9600)
mySerial.begin(9600);

Serial.println("Initializing...");
delay(1000);
mySerial.println("AT");
updateSerial();
mySerial.println("AT+CMGF=1");
updateSerial();
mySerial.println("AT+CMGS=\"+213xxxx\"");
updateSerial();
mySerial.print("Hello world");
updateSerial();
mySerial.write(26);
}
voidloop()
{
}
voidupdateSerial()
{
delay(500);
while (Serial.available())
{
mySerial.write(Serial.read}
while(mySerial.available())
{
Serial.write(mySerial.read}
}
```

Explication du programme :

Le programme commence par inclure une bibliothèque SoftwareSerial.h et l'initialiser avec les broches Arduino auxquelles Tx et Rx du module SIM800L sont connectés.

```
#include<SoftwareSerial.h>
SoftwareSerialmySerial(3, 2);
```

En fonction de configuration : nous initialisons une liaison de communication série entre Arduino, Arduino IDE et module SIM800L à un débit de 9600 bauds.

```
Serial.begin(9600)
mySerial.begin(9600);

Serial.println("Initializing...");
delay(1000);
```

Maintenant que nous avons établi une connexion de base, nous allons essayer de communiquer avec le module SIM800L en envoyant des commandes AT.

```
mySerial.println("AT");
updateSerial();
mySerial.println("AT+CMGF=1");
updateSerial();
mySerial.println("AT+CMGS=\"+213xxxx\"");
updateSerial();
mySerial.print("Hello world");
updateSerial();
mySerial.write(26);
```

AT : C'est la commande AT la plus élémentaire. Si cela fonctionne, en cliquant sur AT on aura une réponse OK dans le moniteur série. Ensuiteon envoie des commandes pour interroger le module et obtenir des informations à son sujet.

AT + CMGF = 1 : Sélectionne le format du message SMS sous forme de texte.

AT + CMGS = + 213xxxx : Envoie un SMS au numéro de téléphone spécifié.

La boucle est maintenue vide car nous ne voulons envoyer des SMS qu'une seule fois. Si vous souhaitez envoyer des SMS une fois de plus, appuyez simplement sur la touche RESET de votre Arduino.

```
voidloop()
{
}
voidupdateSerial()
{
delay(500);
while (Serial.available())
{
mySerial.write(Serial.read}
while(mySerial.available())
{
Serial.write(mySerial.read}
}
```

Dans la partie en boucle du code, nous appelons la fonction personnalisée appelée updateSerial () qui attend en permanence toutes les entrées du moniteur série et l'envoie au module SIM800L via la broche D2 (Rx du module). Il lit également en continu la broche D3 (Tx du module) si le module SIM800L a des réponses.

Résultat :

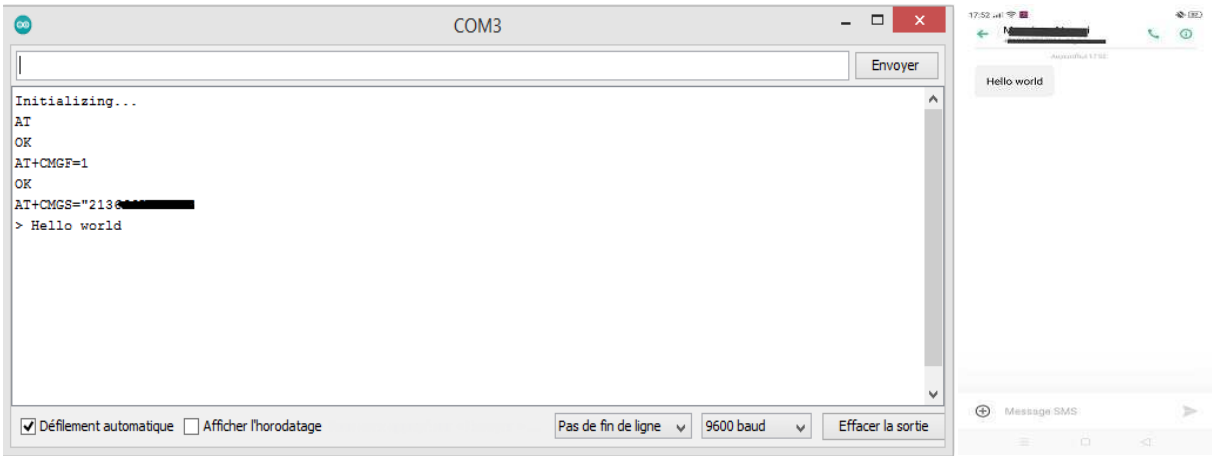

FIGURE II.8 – Les SMS envoyés depuis le module GSM SIM800L.

II.3.3 Module SD

Le module SD est utilisé pour transférer des données depuis et vers une carte SD standard (voir le module SD sur la figure II.9). La broche est directement compatible avec Arduino UNO. Cela nous permet d'ajouter du stockage de masse et l'enregistrement de données de position. Ce module utilise l'interface SPI standard pour la communication, qui implique des bus SPI, MISO, MOSI, SCK et une broche de signal CS. Grâce à la programmation, les données peuvent facilement être lues et écrites sur la carte SD en utilisant l'Arduino UNO [11].

FIGURE II.9 – Module SD [11].

II.3.3.1 Connexion du module

Le module de carte SD présente 6 broches pour permettre d'établir la connexion (voir figure II.10). 2 connexions pour l'alimentation et 4 pour établir la liaison SPI (interface périphérique série).

- $5V$ ou $3.3V$ pour l'alimentation du module
- GND la masse du module
- CS ou ChipSelect pour activer la communication
- MISO (Master Input, Slave Output) broche de transmission équivalente à la borne Tx d'un port série. Sortie du module
- MOSI (Master Output, Slave Input) broche de réception équivalente à la borne Rx d'un port série. Entrée du module
- SCK (Clock) horloge permettant de synchroniser la communication [11]

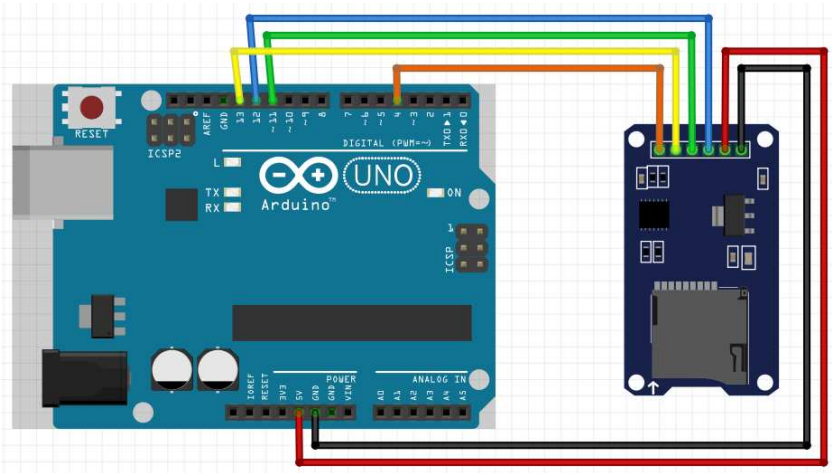

FIGURE II.10 – Connexion du module SD.

II.3.3.2 Écriture et lecture des coordonnées GPS sur la carte SD

Diagrame de cablage :

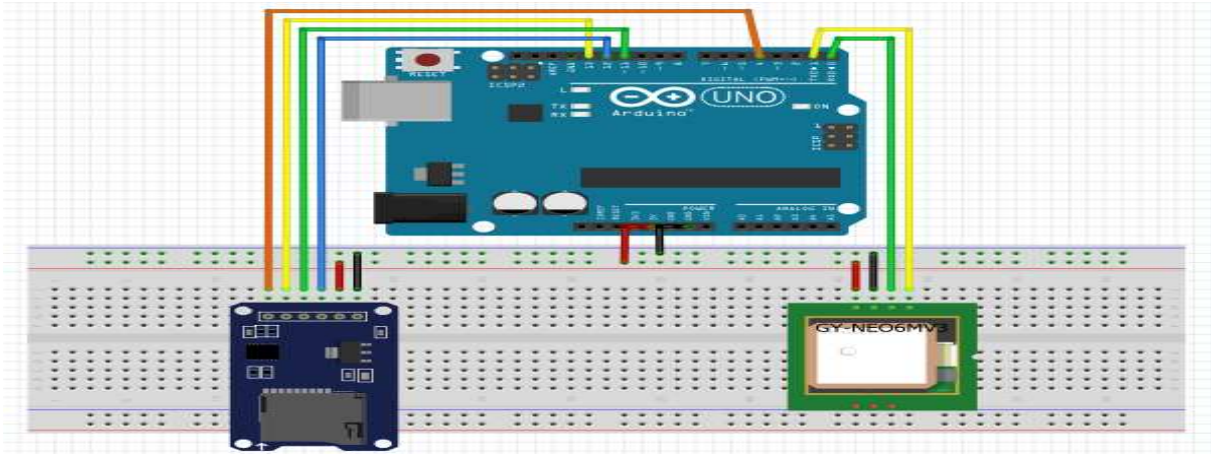

FIGURE II.11 – Diagramme de câblage.

Programme

```
#include<TinyGPS++.h>
#include<SoftwareSerial.h>
#include<SPI.h>
#include<SD.h>

File myFile;
constintchipSelect = 4;
char data;

doublelatitude;
doublelongitude;
doublevitesse;
TinyGPSPlusgps;

Voidsetup(){
Serial.begin(9600);
Serial.print(F("Initializing SD card..."));
pinMode(4, OUTPUT);

if(!SD.begin(chipSelect)){
Serial.println(F("Cardfailed, or not present"));
return;
}
Serial.println(F("cardinitialized."));
}
void loop()
{while(Serial.available())
{data = Serial.read();
gps.encode(data);
if(gps.location.isUpdated())
```

```
{ latitude = gps.location.lat();
longitude = gps.location.lng();
vitesse = gps.speed.kmph();
if(vitesse>0){
Serial.println("--------- DONNEES GPS ----------");
Serial.print("LATITUDE="); Serial.println(latitude);
Serial.print("LONGITUDE="); Serial.println(longitude);
Serial.print("VITESSE (km/h)="); Serial.println(vitesse);

File myFile = SD.open("datalog.txt", FILE_WRITE);
if(myFile){
myFile.print(longitude);
myFile.print("; ");
myFile.print(latitude);
myFile.println(" ;");
myFile.close();
}
else{
Serial.println(F("error opening datalog.txt"));
}
}}}
```

Explication du programme :

Le programme est presque le même que celui du GPS, sauf l'extrait de code ci-dessous :

```
#include<SPI.h>
#include<SD.h>
```

Pour écrire et lire à partir de la carte SD, on doit d'abord inclure le **SPI** et **SD** bibliothèques. Comme on doit initialiser le module de la carte SD à la broche Chip Select (CS) : dans notre cas, la broche 4.

```
constintchipSelect = 4;
```

Pour ouvrir un nouveau fichier sur la carte SD, on doit créer un fichier objet qui fait référence à notre fichier de données.

Le premier paramètre de cette fonction est le nom du fichier, **datalog.txt**, et le **FILE_WRITE**; l'argument nous permet de lire et d'écrire dans le fichier.

Cette ligne de code crée un fichier appelé datalog.txt sur notre carte SD.

```
\noindent File myFile = SD.open("datalog.txt", FILE_WRITE);
```

On utilise la fonction print () pour imprimer des données dans le fichier.

```
if(myFile){
myFile.print(longitude);
myFile.print("; ");
myFile.print(latitude);
```

```
myFile.println(" ;");
```

On ne peut écrire que dans un fichier à la fois, nous devons donc fermer un fichier avant de passer au suivant.

```
myFile.close();
```

Résultat :

Le résultat est tiré de la carte SD puis affiché sur l'écran du téléphone (voir figure II.12). Le fichier.txt nommé "datalog.txt" a été créé et les coordonnées ont été écrites dans le fichie.Txt.

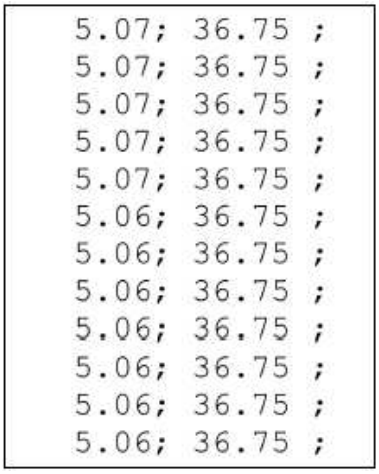

FIGURE II.12 – Affichage sur la carte SD

II.3.4 Module Bluetooth :

Le module Bluetooth HC-05 permet d'établir une liaison Bluetooth (liaison série) entre une carte Arduino et un autre équipement possédant une connexion Bluetooth (Smartphone, tablette, seconde carte Arduino, etc....) (voir figure II.13). L'appareil peut être utilisé en 2 modes ; mode de données et mode de commande.

Le mode de données est utilisé pour le transfert de données entre les appareils tandis que le mode de commande est utilisé pour modifier les paramètres du module Bluetooth.

Le module Bluetooth peut être configuré grâce à des commandes AT (ou commandes Hayes) [12].

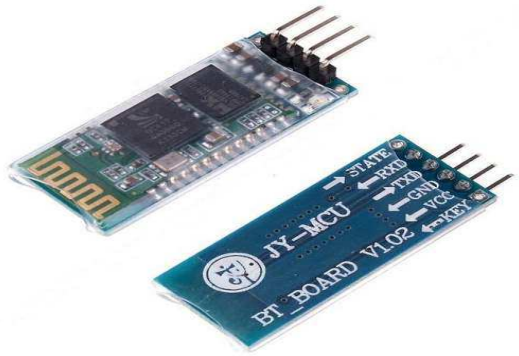

FIGURE II.13 – Module Bluetooth HC-05 [12].

II.3.4.1 Connexion du module Bluetooth HC-05

Le module Bluetooth HC-05 présente 6 broches pour permettre d'établir la connexion.

- VCC broche d'alimentation. Typiquement connectée à la broche $5V$ de l'Arduino.
- GND masse. Typiquement connectée à la broche GND de l'Arduino
- RX broche de réception. Typiquement connecté à la broche de transmission (TX) de l'Arduino
- TX broche de transmission. Typiquement connecté à la broche de réception (RX) de l'Arduino
- State retourne 1 lorsque le module est connecté
- Key ou EN doit être alimentée pour entrer dans le mode de configuration et ne doit pas être connecté pour être en mode communication [12].

II.3.4.2 Câblage du module Bluetooth HC-05

Ce module communique via une liaison série avec une carte Arduino. Cette liaison s'établit sur deux broches RX et TX définies dans notre programme en tant que broches 9 et 8, VCC au $5V$ etGND auGND (voir figure II.14).

II.3.4.3 Configuration du Module Bluetooth HC-05

Programme : Changement de nom et de code

```
/* HC-05 Config est un programme qui permet de configurer le module Bluetooth*/
```

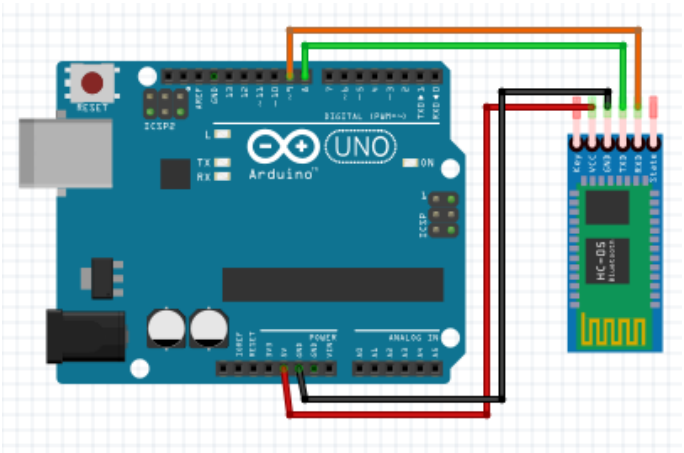

FIGURE II.14 – Câblage du module Bluetooth

```
//Création d'une liaison série sur les broches 8 et 9
#include<SoftwareSerial.h> //appel de la bibliothèque
Software Serial HC05 (8, 9); //RX, TX

voidsetup()
{
HC05.begin(9600); //initialisation connexion série Bluetooth à 9600
bauds
Serial.begin(9600); //initialisation liaison série à 9600 bauds
Serial.println("Entrer la commande AT:");
}

voidloop()
{
//si le HC-05 transmet, on écrit le message dans le moniteur série
if (HC05.available())
{
Serial.write(HC05.read());
}
//si on écrit dans le moniteur série, on transmet le message dans le
module HC-05
if (Serial.available())
{
HC05.write(Serial.read());
}}
```

Commentaires :

- Pour tester la communication, taper AT dans le moniteur série et le module doit répondre OK.
- Pour modifier le nom du module, taper AT+NAMENom_du_module (sans espace), le module répond OKsetname.

- Pour modifier le code PIN du module, taper AT+PINcode (sans espace), le module répond OKsetPIN.

II.3.4.4 Exemple : Commander une LED via Bluetooth

A - Branchement

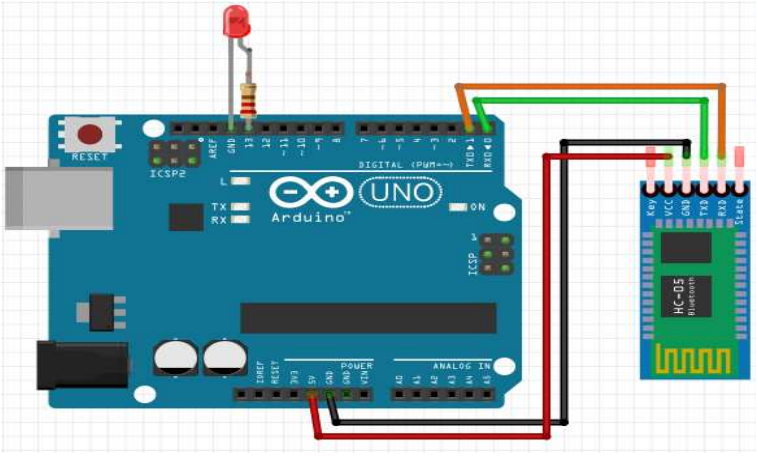

FIGURE II.15 – Branchement Led et module Bluetooth

B - Programme du côté Arduino

Le programme lit dans la variable "data" ce qui est reçu sur le port série.

- Si $data = 1$, alors on allume la led sur la broche 13,
- Si $data = 0$, alors on éteint la led sur la broche 13.

```
char data = 0;

void setup()
{
Serial.begin(9600);
pinMode(13, OUTPUT);
}

void loop()
{
  if(Serial.available() > 0)
  {
    data = Serial.read();
Serial.print(data);
Serial.print("\n");
    if(data == '1')
digitalWrite(13, HIGH);
```

```
    else if(data == '0')
digitalWrite(13, LOW);
  }
}
```

C - Programme du côté Android

La méthode la plus simple pour réaliser son application Android est d'utiliser le programme APP INVENTOR (voir figure II.16).

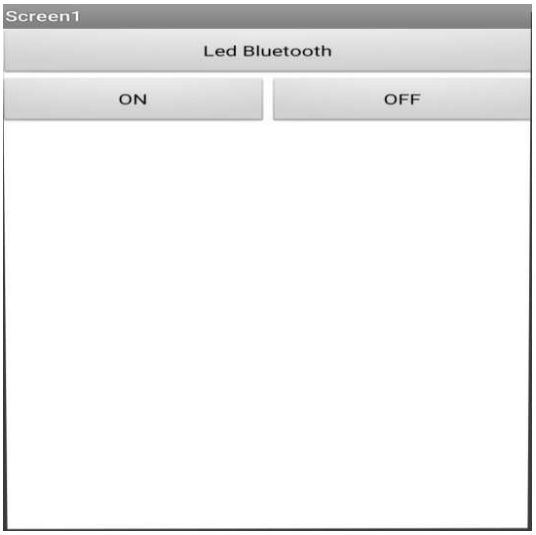

FIGURE II.16 – Application Android pour Bluetooth

D - Mise en œuvre

- Charger le programme Arduino et installer l'application ANDROID
- Brancher le module HC05
- Réaliser l'appairage de l'appareil ANDROID et du module HC05 (qui apparait comme QLF dans la liste des périphériques Bluetooth avec un code Pin).
- Lancer l'application sur l'appareil ANDROID
- Dans l'appli ANDROID, cliquer sur "Led Bluetooth" sélectionner QLF (la led va arrêter de clignoter pour rester allumée, ce qui indique que la communication Bluetooth est activée)
- Cliquer sur les boutons, ce qui permettra d'allumer ou d'éteindre la Led

II.4 Conclusion

Dans ce chapitre nous avons parlé sur le fonctionnement de notre système de suivi du véhicule tout en exposant les différents composants électroniques de base qui le construisent.

Dans le prochain chapitre nous allons présenter l'application Android «trackeur_app» qui va nous permettre d'afficher l'emplacement du véhicule.

Chapitre III

Développement de l'application Android

III.1 Introduction

La plateforme Android de smart phone devient de plus en plus importante pour les réalisateurs de logiciel, en raison de ses puissantes possibilités et sa gratuité.

Dans ce chapitre, nous allons décrire le processus de réalisation d'une interface de commande via l'environnement App Inventor. Cette application va nous permettre d'avoir le contrôle sur notre véhicule.

III.2 Définition du système d'exploitation mobile «Android»

Android c'est une plateforme complète pour appareil mobile, lancé par Google qui compose d'opérateurs mobiles, de fabricants de téléphones et d'éditeurs logiciels (voir figure III.1). Il équipe la majorité des téléphones portables du moment (smart phones). Son principal concurrent est Apple avec l'iPhone. Android est un système vous permettant de personnaliser votre téléphone, télécharger des applications (navigateur Internet, GPS, Facebook...), de plus il équipe également les tablettes tactiles [13].

FIGURE III.1 – Icône Android [13].

III.3 MIT App Inventor

App Inventor pour Android est une application développée par Google mais actuellement entretenue par Massachusette institut of Technology (MIT). Elle simplifie le développement des applications sous Android et le rend accessible même pour ceux qui ne sont pas familiers avec les langages de programmation. Cet environnement de programmation permet une programmation graphique basée sur l'assemblage de blocs (langage scratch) [14].

III.3.1 Pourquoi MIT App Inventor ?

Nous avons choisi de travailler avec App Inventor pour les raisons suivantes :

- Il s'agit d'un logiciel très bien documenté.
- Prise en main rapide pour les gens non spécialisés.
- Environnement simple et efficace.
- Pas de langage à apprendre, donc pas de risque d'erreur syntaxique.
- Logiciel libre, gratuit, et multiplateforme.

III.3.2 Structure d'un IDE App Inventor

L'IDE App Inventor est composé de deux interfaces : Designer, Blocs [15].

III.3.2.1 Interface Graphique

L'interface graphique contient nos propriétés (taille, couleurs, position, textes, ...) (voir figure III.2).

L'interface graphique est composée de quatre zones :

► **La zone palette :** c'est la zone où se trouvent tous les éléments qui vont composer l'application (voir figure III.3).

Dans notre application nous avons utilisé les composants graphiques suivants :

- Interface utilisateur : Bouton, Label, zone de texte, image, zone texte mot de passe, afficheur Web.
- Disposition : Arrangement horizontal, Arrangement vertical.
- Connectivité : BluetoothClient, Web.

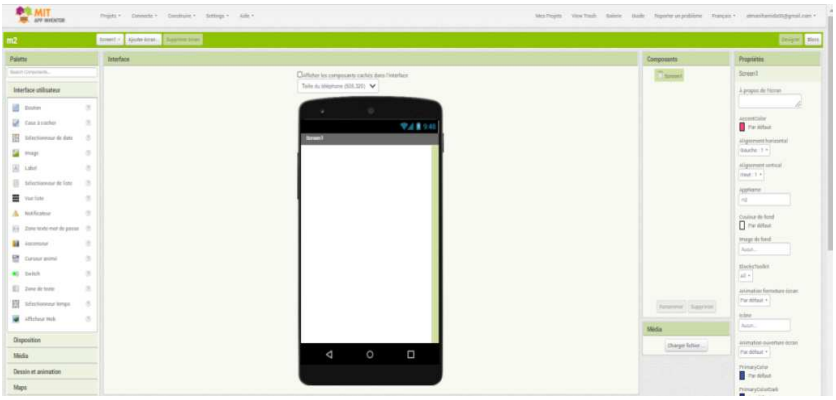

FIGURE III.2 – Interface de design d'App Inventor

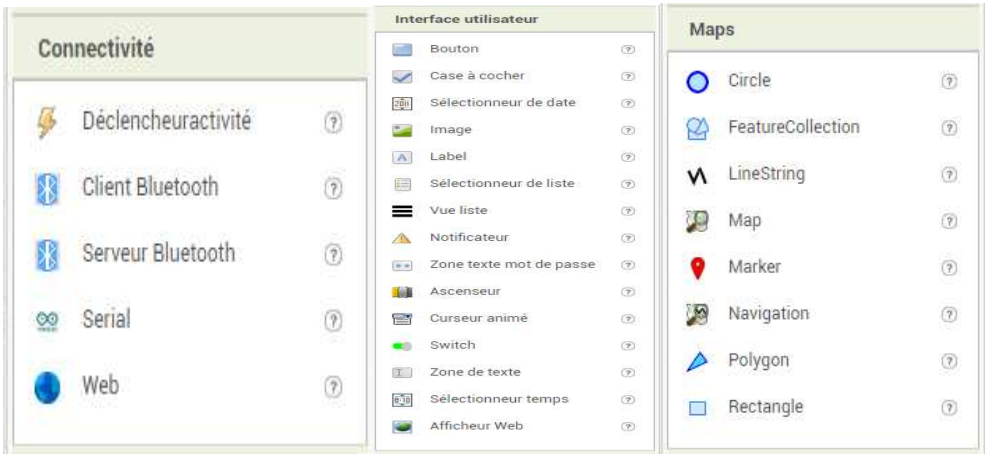

FIGURE III.3 – Composants graphiques de la palette

- Maps : Marker, Map.

► **La zone Interface :** Où l'écran s'affiche, il donne un aperçu visuel de l'application, il est possible d'ajouter plusieurs écrans en cliquant sur ce bouton. En haut de l'écran, il y a la case «Afficher les composants cachés dans l'interface», cette case permet d'afficher ou non les éléments de l'application, en cochant cette case.

► **La zone composants :** les éléments ajoutés sur l'écran vont être apparaitre dans cette interface sous forme de graphe.

► **La zone Média :** En cliquant sur «chargerfichier» il est possible de télécharger les médias (son, image,...) et les insérer dans l'application.

► **La zone Proprietés :** c'est la zone de réglage des propriétés de chaque élément (alignement, couleur, ...).

III.3.2.2 Interface blocs

Une fois les composants de l'écran de téléphone mis en place et désignés, nous passons à la deuxième phase de développement d'une application via App Inventor : l'interface Scratch [14].

L'interface Scratch permet d'imbriquer des éléments graphiques entre eux pour effectuer la partie programmation de l'application à développer. De cette partie, on peut assembler les différents blocs de l'application et indiquer comment les composants doivent se comporter et qui s'affichent dans l'émulateur virtuel (voir figure III.4).

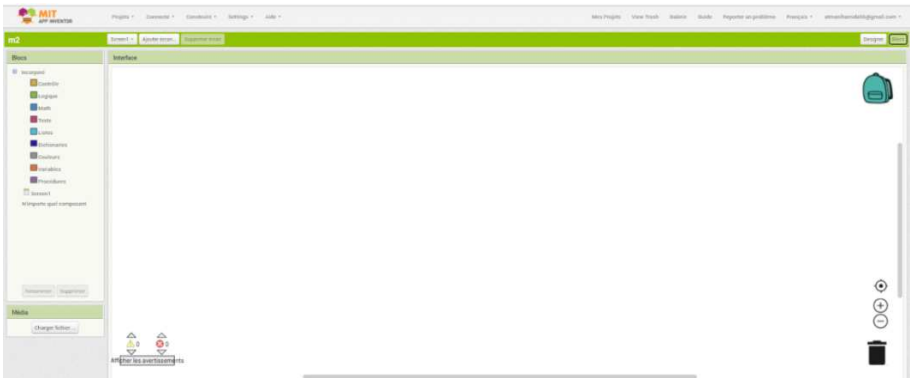

FIGURE III.4 – Interface des blocs d'App Inventor.

III.3.3 Connexion de l'App Inventor sur le Smart Phone

L'une des fonctions les plus intéressantes d'App Inventor c'est qu'on peut visualiser et tester l'application sur un appareil connecté pendant la construction.
App Inventor propose trois modes de connexion [15] :

- Connecter le téléphone via USB.
- Connecter au wifi via QR code, lancer l'application sur le Smartphone, puis saisir, ou scanner, ou bien faire entrer manuellement ce code pour que l'application puisse démarrer.
- Tester avec un émulateur.

III.4 Les étapes de développement de l'application «Trackeur_app»

Notre application est composée de trois écrans (Screen) indiqués ci-dessous :

⁕ Le premier est destiné pour l'authentification comme mesure de sécurité. L'application est verrouillée via un code secret pour assurer la Conservation et la protection des informations personnelles qui vous sont confiées.

Comme l'indique la figure III.5, vous introduisez le mot de passe puis vous cliquez sur valider pour accéder à l'application.

FIGURE III.5 – Authentification.

⁕ Le deuxième représente une interface de commande pour faciliter l'accès aux différentes tâches de notre application (voir figure III.6).

FIGURE III.6 – L'interface de commande l'application.

- Bouton SMS : Accéder à la messagerie et lire le SMS contenant la localisation du véhicule envoyée via module GSM.

- Bouton Bluetooth : Suite à cette action, notre appareil passera en mode association, le rendant détectable par notre module Bluetooth HC-05.Cliquez pour sélectionner l'appareil et lancer l'association.
- Bouton ON/OFF : Contrôler le Marche/Arrêt du système.
- Bouton => : Accéder à la carte géographique.

⁕ Le troisième écran est destiné à la carte géographique (voir figure III.7).

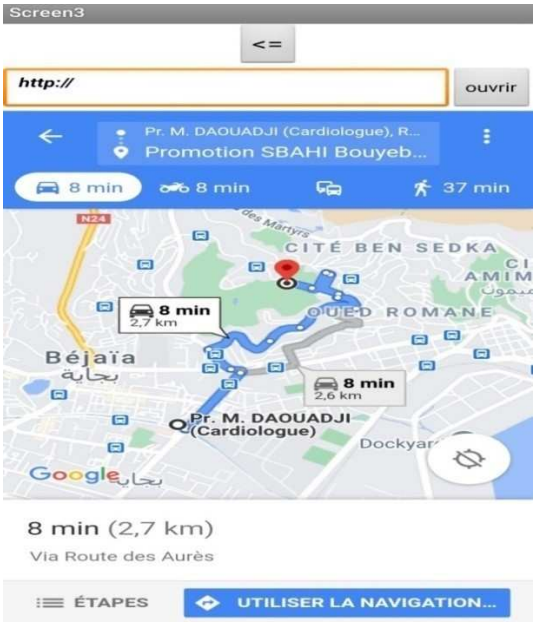

FIGURE III.7 – Carte géographique.

Cette carte géographique nous permet de :

- Copier le lien envoyé via module GSM, déterminant la localisation du véhicule et l'afficher sur la carte géographique
- Accéder à la navigation Google Maps
- Afficher l'itinéraire parcouru par le véhicule

III.5 Conclusion

Dans ce chapitre nous avons réalisé une application nommée «Trackeur_app» via la plateforme App Inventor, capable de lier une carte Arduino à un smart phone afin d'exécuter des ordres bien définis.

Une fois la partie software (Arduino) et les commandes sous smart phone (Application) sont faites, dans le prochain chapitre nous passons à la partie réalisation matériel.

Chapitre IV

Réalisation pratique

IV.1 Introduction

Un système de suivi de véhicule est un dispositif électronique installé dans un véhicule pour permettre au propriétaire de suivre son emplacement. Dans ce chapitre, on va réaliser un système de suivi de véhicule qui fonctionne en utilisant la technologie GPS et GSM.

Pour se faire, un Arduino UNO est interfacé en série à un module GSM et un Récepteur GPS.

Le module GPS donnera en continu les coordonnées du véhicule, le module GSM est utilisé pour envoyer ces mêmes coordonnées (latitude et longitude) au propriétaire. En effet,L'utilisateur envoie au numéro du module GSM un SMS, le système renvoie automatiquement une réponse de retour à ce portable indiquant la position du véhicule sous forme de lien comportant la latitude et la longitude en temps réel.

Ce système peut également être désigné comme un système de suivi antivol car il assure la récupération des véhicules volés. Si la voiture ne soit pas à son emplacement désigné ou utilisé par un utilisateur non autorisé, le véhicule peut être tracé et récupéré.

IV.2 La réalisation virtuelle «PROTEUS»

Avant de passer à la réalisation pratique, nous avons utilisé un logiciel CAO : il s'agit de ISIS-PROTEUS, qui permet de dessiner des schémas électroniques, de les simuler et de réaliser le circuit imprimé correspondant. PROTEUS est disponible et téléchargeable sur ce lien [11], et se compose de nombreux outils regroupés en modules au sein d'une interface unique. Ce dernier nous

permet de schématiser notre carte électrique et la simuler virtuellement comme le montre la figure suivante (voir figure IV.1) :

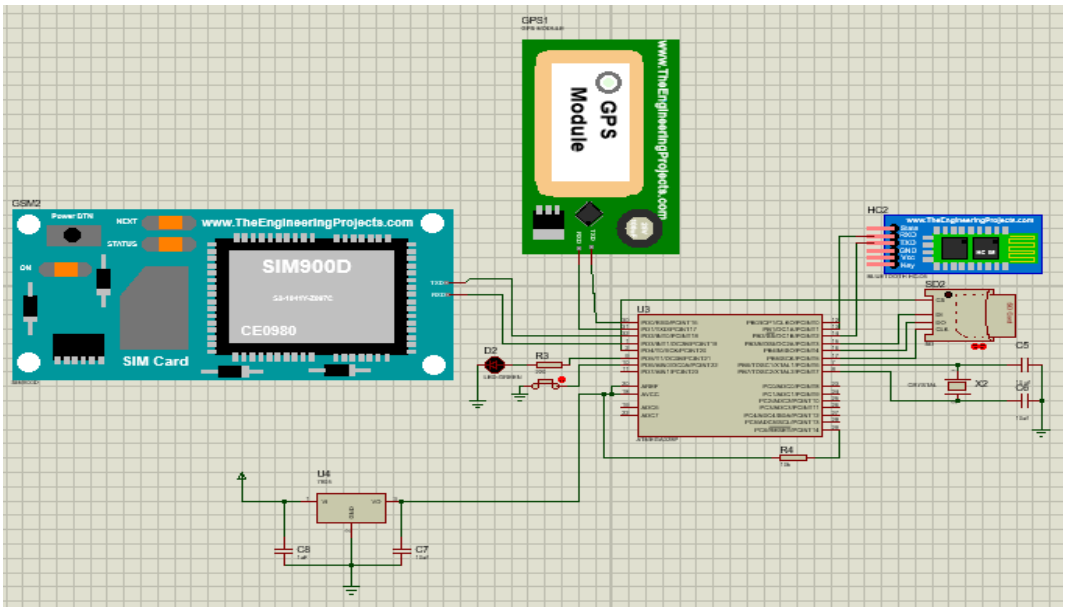

FIGURE IV.1 – La carte réalisée sous ISIS-PROTEUS

IV.3 Présentation du système

Notre système est un circuit électronique qui est réalisé à base d'une carte électronique dont les composants sont les suivants :

- La carte de prototypage.
- Un microcontrôleur ATMEGA 328.
- Des capacités et des résistances.
- Un quartz 16MHZ.
- Une LED.
- Des ports d'entrée/sortie.
- Un régulateur 7805
- Les modules utilisés :
  - Module GPS NEO-6M
  - Module GSM SIM800L
  - Module Bluetooth HC-05

– Module SD

IV.4 Réalisation pratique

La réalisation pratique, sur maquette, du système nécessite les composants suivants :

IV.4.1 La carte de prototypage

En général une plaque permet de maintenir et de relier électriquement un ensemble de composants électroniques entre eux, dans le but de réaliser un circuit électronique complexe (voir figure IV.2). On la désigne aussi par le terme de carte électronique.

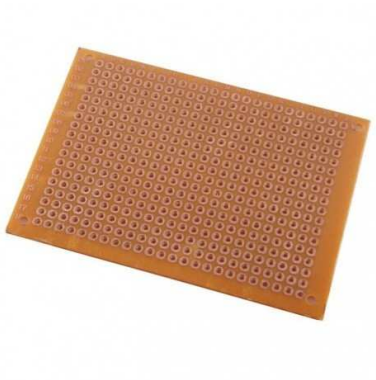

FIGURE IV.2 – Représentations de la carte prototypage.

IV.4.2 Le Quartz

Un quartz est un composant qui possède comme propriété utile d'osciller à une fréquence stable lorsqu'il est stimulé électriquement. Les propriétés piézoélectriques remarquables du minéral de quartz permettent d'obtenir des fréquences d'oscillation très précises, ce qui en fait un élément important en électronique numérique ainsi qu'en électronique analogique (voir figure IV.3).

Pour notre carte nous avons utilisé un Quartz de $16MHz$ pour donner la vitesse d'exécution nécessaire pour le microcontrôleur

FIGURE IV.3 – Représentations d'un quartz 16MHZ.

IV.4.3 Condensateurs

Le condensateur est un composant électronique qui peut être comparé à un réservoir d'énergie (voir figure IV.4). Il est constitué de deux armatures métalliques séparées par un isolant. Lorsqu'on applique une tension à ses bornes, l'isolant est soumis à cette tension et accumule de l'énergie électrostatique. Il estutilisé pour lisser la tension.

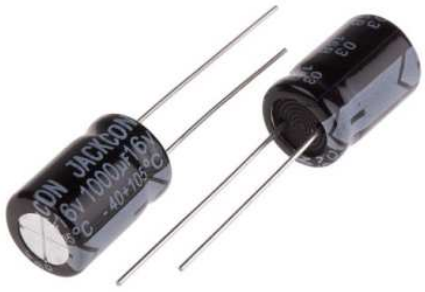

FIGURE IV.4 – Le condensateur chimique.

IV.4.4 LED

Les LEDs sont des diodes qui produisent de la lumière lorsque le courant y passe en polarisation directe de l'anode vers la cathode. Tout comme toute diode, la polarisation inverse bloque le courant et la LED dans ce cas n'émet aucune lumière.

La LED est caractérisée par des propriétés électriques, optiques, thermiques et géométriques (voir figure IV.5).

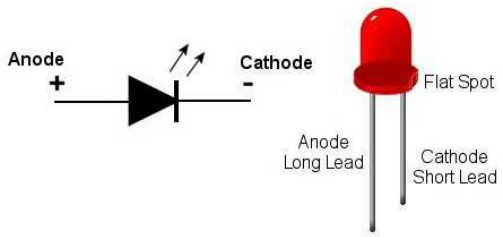

FIGURE IV.5 – Représentation d'une LED [5].

IV.4.5 Résistances

Une résistance est un composant électronique ou électrique dont la principale caractéristique est d'opposer à la circulation du courant électrique. Ce composant est caractérisé par sa valeur exprimée en Ohm et ces multiples (Ω, $K\Omega$, $M\Omega$) codé en couleur de la manière suivante (voir figure IV.6) :

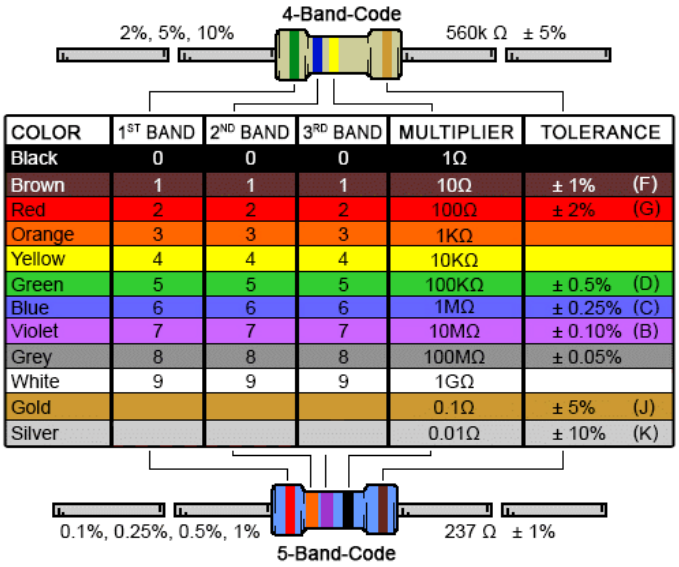

FIGURE IV.6 – Code des couleurs des résistances.

IV.4.6 Le régulateur de tension 7805

Les régulateurs de tension sont très courants dans les circuits électroniques (voir figure IV.7). Ils fournissent une tension de sortie constante pour une tension d'entrée variée. Dans notre cas, le 7805 IC est un IC régulateur iconique qui trouve son application dans la plupart des projets. Le

nom 7805 signifie deux significations, «78» signifie qu'il s'agit d'un régulateur de tension positive et «05» signifie qu'il fournit $5V$ en sortie. Notre 7805 fournira donc une tension de sortie de $+5V$.

FIGURE IV.7 – Représentations d'un régulateur 7805.

IV.5 Montage global

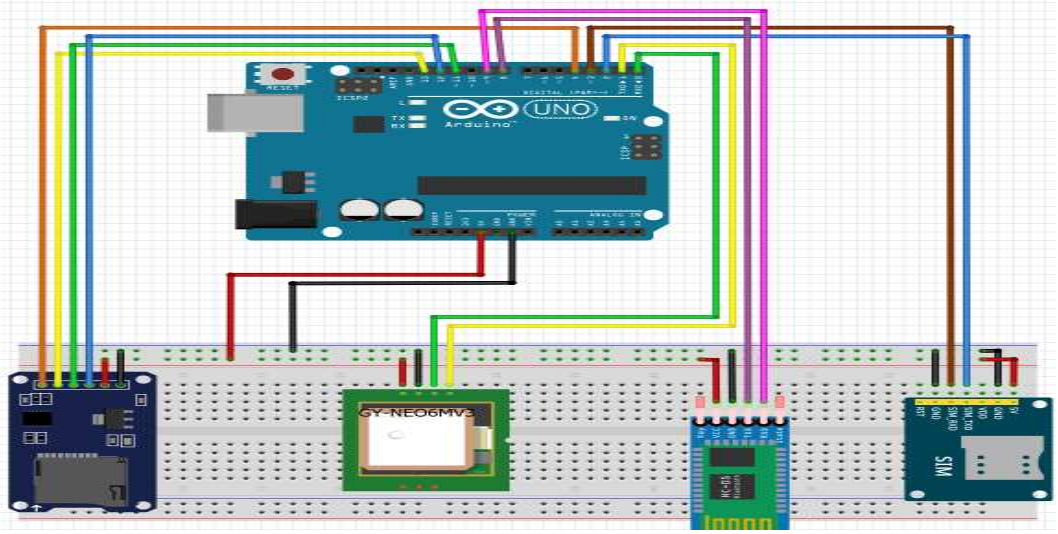

FIGURE IV.8 – Montage global sur Fritzing

IV.6 Test du système

Une fois que notre système est branché à l'arduino uno et une fois alimenté, le module GPS commence à extraire les coordonnées (latitude et longitude) désignant l'emplacement du véhicule et les enregistrer dans la carte SD pour ne pas manquer le mouvement de la voiture. Avec le module GSM on demande le lieu du véhicule à tout moment et on l'affiche sur une carte géographique (voir figures IV.9, IV.10, IV.11).

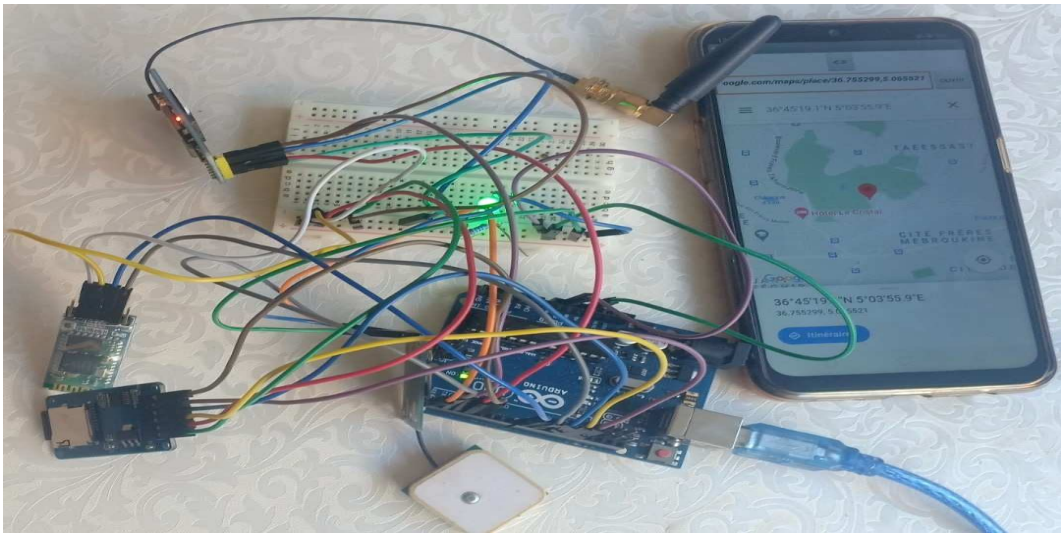

FIGURE IV.9 – Montage réel du Trackeur

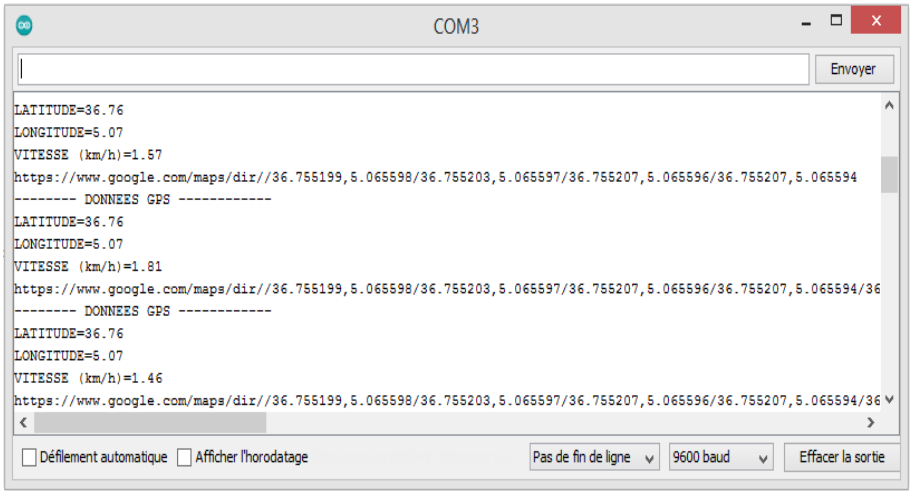

FIGURE IV.10 – Affichage de la position du véhicule et son chemin parcouru.

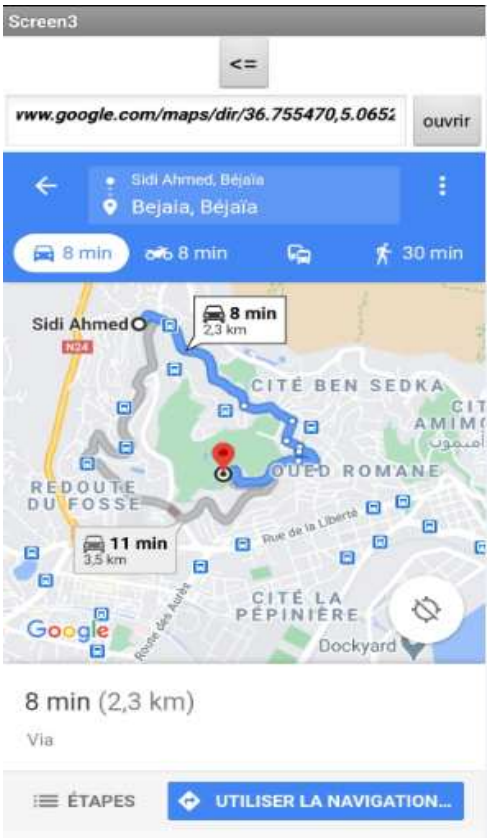

FIGURE IV.11 – Affichage du chemin parcouru par le véhicule sur une carte géographique.

IV.7 L'algorithme

Nous avons utilisé le logiciel Arduino IDE pour programmer le fonctionnement de notre système.

Dans le codage, nous incluons la bibliothèque de communication série pour permettre la communication série sur les broche de la carte ARDUINO avec les modules.

Dans la fonction principale GPS (), la carte va lire à partir du module GSM le message envoyé par l'utilisateur qui demande les coordonnées, et la chaîne du GPS, puis on va extraire les coordonnées de la chaîne reçue du module GPS. Si le message envoyé par l'utilisateur correspond à (GPS ON) la carte commande le module GSM d'envoyer alors les coordonnées à l'utilisateur sous forme de lien ≪"www.google.com/maps/place/" + String (latitude, 6) + "," + String (longitude, 6) ;≫ indiquant la position du véhicule. Au même temps les coordonnées GPS sont enregistrées et stockées dans la carte SD via module SD sous forme de lien ≪ "www.google.com/maps/dir/" +

String (latitude, 6) + ”,” + String (longitude, 6) ;» permettant à l'utilisateur de vérifier le chemin parcouru par le véhicule (voir figure IV.12).

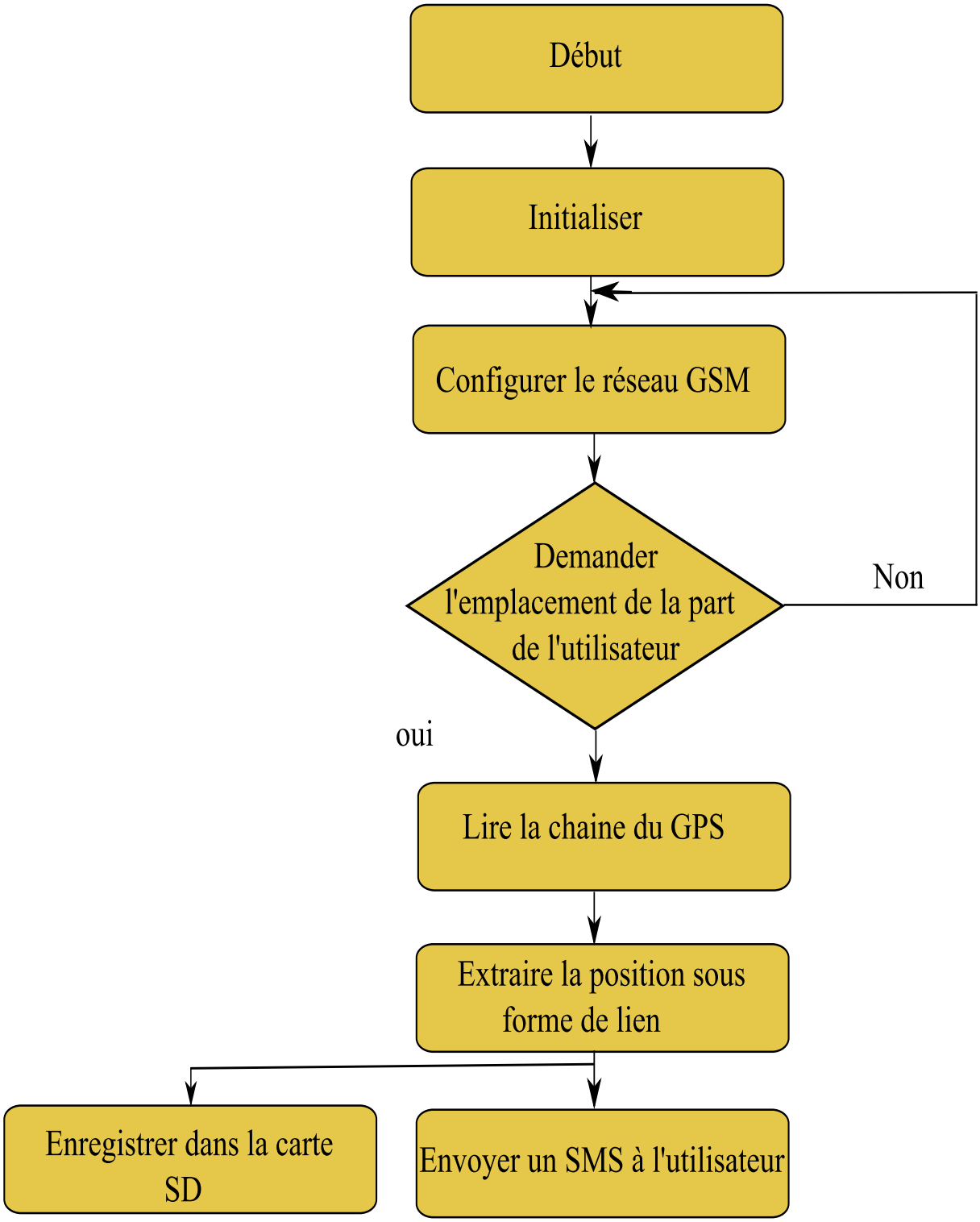

FIGURE IV.12 – L'organigramme de codage.

IV.8 Conclusion

Dans ce chapitre, on s'est intéressé à la structure générale du système de suivi et son fonctionnement en décrivant les étapes à suivre pendant la réalisation, tel que l'algorithme et le code source Arduino utilisé. Ensuite, une simulation sur le logiciel PROTEUS pour montrer le fonctionnement du système.

Conclusion générale et perspectives

Ce travail rentre dans le cadre du projet de fin d'étude pour l'obtention du diplôme de master en Automatique.

Nous sommes parvenus grâce à ce projet à réaliser un système de suivi du véhicule à base d'Arduino.

Ce travail facilitera à présent l'obtention de l'emplacement d'un véhicule et permettra l'enregistrement de ses déplacements.

Ce projet a fait l'objet d'une expérience intéressante, qui nous a permis d'enrichir nos connaissances théoriques et pratiques sur ce qui est programmation et réalisation.

Nos perspectives d'avenir sont de pouvoir améliorer ce Trackeur et le rendre plus performant en utilisant d'autres options de sécurité tels que : un détecteur de mouvement, une sirène d'alarme ainsi qu'un petit système pour immobiliser le véhicule en cas de vol. Il fonctionne comme la forme la plus économique du système qui peut aider à identifier un vol de véhicule.

De nos jours, le vol de véhicules augmente rapidement. Avec cette technologie cependant, le vol de véhicule peut être mieux contrôlé.

Bibliographie

[1] S. Landrault and H. Weisslinger, *"Arduino : Premiers pas en informatique embarquée"*, le blog d'eskimon ed., 2014.

[2] M. MCROBERTS, *"Début Arduino"*, apress ed., 2012.

[3] E. BARTMANN, *"Le grand livre d'Arduino"*, editions eyrolles ed., 2015.

[4] C. TAVERNIER, *"Arduino : Applications avancées : Claviers tactiles, télécommande par Internet, géolocalisation, applications sans fil"*, dunod ed., 2012.

[5] S. Mezah, *"Cours introduction aux systèmes embarqué temps réel," Cours : Systèmes embarqué temps réel, Université Abderrahmane Mira, Béjaia*, 2018.

[6] M. MARGOLIS, B. JEPSON, and N. R. WELDIN, *"Livre de recettes Arduino : des recettes pour commencer, développer et améliorer vos projets"*, o'reilly media ed., 2020.

[7] Y. MERGY, *"Prise en main de fritzing concevoir un circuit imprimé," Cours, Université de applied science potsdam*, 2010.

[8] Https ://lastminuteengineers.com/neo6m-gps-arduino-datasheet/, Consulté le 12 juin 2020.

[9] Https ://www.faranux.com/product/sim800l-v2-0-5v-wirelessgsm-gprs-module-quad- band/-datasheet, consulté le 10 juin 2020.

[10] D. REY, *"Interfaces GSM-2e éd. : Montages pour téléphones portables,"*, dunod ed., 2010.

[11] Http ://datalogger.pbworks.com/w/file/fetch/89507207/Datasheet/20-, consulté le 30 juillet 2020.

[12] "https ://components101.com/sites/default/files/component_datasheet/hc-05/20datasheet .pdf, consulté le 10 juin 2020."

[13] Https ://www.android.com/versions/pie-9-0/. Consulté le 25 aout 2020.

[14] S. C. POKRESS and J. J. D. VEIGA, *"MIT App Inventor : Enabling personal mobile computing," preprint arXiv.*

[15] H. XIE, Benjamin et ABELSON, "Skill progression in mit app inventor. in :symposium on visual languages and human-centric computing (vl/hcc)," *IEEE*, 2016.

Chapitre A

Annexe 1

```
#include<TinyGPS++.h>
#include<SD.h>
#include<SPI.h>
#include<SoftwareSerial.h>

TinyGPSPlusgps;
SoftwareSerialSIM800L(2, 3);

double latitude, longitude,vitesse;
String response;
intlastStringLength = response.length();
constintchipSelect = 4;
String link;
String link1 = "https://www.google.com/maps/dir/";

voidsetup(){
Serial.begin(9600);
SIM800L.begin(9600);

Serial.print("GPS QLF");
SIM800L.println("AT+CMGF=1");
delay(1000);
Serial.println("SIM800L est prêt !");
SIM800L.println("AT+CNMI=2,2,0,0,0");

Serial.print(F("Initializing SD card..."));
pinMode(4, OUTPUT);
if (!SD.begin(chipSelect)){
Serial.println(F("Cardfailed, or not present"));
return;
}
Serial.println(F("cardinitialized."));
}

voidloop(){
if (SIM800L.available() > 0){
response = SIM800L.readStringUntil('\n');
}

if (lastStringLength != response.length()){
```

```
GPS();
if (response.indexOf("ON") == 4) {

SIM800L.println("AT+CMGF=1");
delay(1000);
SIM800L.println("AT+CMGS=\"213####\"\r");
SIM800L.println(link);
delay(1000);
SIM800L.println((char)26);
delay(100);
}
}
File myFile = SD.open("datalog.txt", FILE_WRITE);
if (myFile){
GPS();
myFile.print(link1);
myFile.print("; ");
myFile.close();
}
else{
Serial.println(F("erroropening datalog.txt"));
}
}
voidGPS(){
if (Serial.available()){
gps.encode(Serial.read());
}
if (gps.location.isUpdated()){
latitude = gps.location.lat();
longitude = gps.location.lng();
vitesse = gps.speed.kmph();
link= "www.google.com/maps/place/" + String(latitude, 6) + "," + String
(longitude, 6);
link1 = link1 + "/" + String(latitude, 6) + "," + String(longitude, 6);
if (vitesse>0) {
Serial.println("--------- DONNEES GPS ----------");
Serial.print("LATITUDE="); Serial.println(latitude);
Serial.print("LONGITUDE="); Serial.println(longitude);
Serial.print("VITESSE (km/h)="); Serial.println(vitesse);
Serial.println(link1);
}
}}
```

Résumé

Un système de suivi de véhicule est très utile pour suivre le mouvement d'un véhicule à partir de n'importe quel endroit à tout moment. Dans ce travail, la carte Google en temps réel et le système de suivi des véhicules basés sur Arduino sont mis en œuvre avec le système de positionnement mondial (GPS) et le système mondial de technologie de communication mobile (GSM). Le module GPS fournit des coordonnées géographiques à intervalles de temps réguliers. Ensuite, le module GSM transmit l'emplacement du véhicule au téléphone portable du propriétaire (utilisateur) en termes de latitude et de longitude. Enfin, Google Maps affiche l'emplacement et le nom du lieu sur le téléphone portable. Ainsi, le propriétaire (utilisateur) pourra surveiller en permanence un véhicule en mouvement à l'aide du téléphone portable. Afin de montrer la faisabilité et l'efficacité du système, ce travail présente les résultats expérimentaux du système de suivi des véhicules.

MOTS-CLÉS : Application, Android, Arduino UNO**,** GSM, GPS, Géolocalisation, Module SD.

Abstract

A vehicle tracking system is very useful for tracking the movement of a vehicle from any location at any time. In this work, real time Google maps and Arduino based vehicle tracking system is implemented with Global Positioning System (GPS) and Global system for mobile communication (GSM) technology. GPS module provides geographic coordinates at regular time intervals. Then the GSM module transmits the location of vehicle to cell phone of owner (user) in terms of latitude and longitude. Finally, Google maps displays the location and name of the place on cell phone. Thus, owner (user) will be able to continuously monitor a moving vehicle using the cell phone. In order to show the feasibility and effectiveness of the system, this work presents experimental result of the vehicle tracking system. The proposed system is user friendly and ensures safety and surveillance at low maintenance cost

KEYWORDS : Application, Android, Arduino UNO, GSM, GPS, Géolocalisation, SD card.

ملخص

يعد نظام تتبع المركبات مفيدا جدا لتتبع حركة السيارة من أي مكان و في أي وقت، في هذا العمل، يتم تنفيذ خرائط (Google) في الوقت الفعلي و نظام تتبع المركبات المستند إلى آردوينو (Arduino) باستخدام نظام تحديد المواقع العالمي (GPS) و النظام العالمي للاتصالات المتنقلة (GSM). توفر وحدة GPS إحداثيات جغرافية على فترات زمنية منتظمة. ثم تنقل وحدة GSM موقع السيارة إلى الهاتف الخلوي للمالك (المستخدم) من حيث خطوط الطول والعرض. أخيرا، تعرض خريطة Google موقع واسم المكان على الهاتف الخلوي. وبالتالي، سيتمكن المالك (المستخدم) من مراقبة مركبة متحركة باستمرار باستخدام الهاتف الخلوي. من أجل إظهار جدوى وفعالية النظام، يقدم هذا العمل نتيجة تجريبية لنظام تتبع المركبات. النظام المقترح سهل الاستخدام ويضمن السلامة والمراقبة بتكلفة صيانة منخفضة.

الكلمات المفتاحية: التطبيق، اندرويد، آردوينو، نظام تحديد المواقع العالمي، النظام العالمي للاتصالات المتنقلة، تحديد الموقع الجغرافي، وحدة الذاكرة