République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté des **S**ciences de l'**I**ngénieur

Département d'Ingénierie des Systèmes Electriques

Mémoire de Master

Présenté par :

Mr. Bouisri Sid Ahmed

En vue de l'obtention du diplôme de Master en

Electronique des systèmes embarqués

Thème :

Etude et réalisation d'un système à temps réel pour la surveillance intelligente des automobiles

Président			
Rapporteurs	Mr SMAANI Billel	MCB	UMBB
Examinateurs			

- Promotion juin 2018 -

Remerciements

Tout d’abord, je tiens à remercier dieu, le généreux, le tout puissant qui nous a donnés la force et le courage, la volonté et les moyens nécessaires pour réaliser ce modeste travail.

Ma première reconnaissance va à mon encadreur **Mr Billel SMAANI** d’avoir accepté de diriger ce travail, ses apports et ses conseils toujours judicieux et ses encouragements. Je lui dis, Merci.

Ensuite pour les membres du jury qui ont accepté d’évaluer et de juger mon travail. Ma gratitude va également pour mes enseignants qui m’ont donné l’envie de poursuivre dans cette voie, Mes remerciements s’adressent aussi à tout le personnel du département génie électrique qui m’a aidée tout au long de mon cursus universitaire, sans oublier tous mes amis qui m’ont aidé de près ou de loin.

Je tiens à adresser mes remerciements les plus chaleureux à ma très chère famille pour son aide, sa générosité, ses sacrifices, son encouragement, et la grande patience dont ils ont su faire preuve.

Enfin, mes sincères remerciements vont à tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire, pour leur soutien et leurs encouragements, … merci à tous.

Tables des matières

Tables des matières

Tables des matières

Liste des acronymes

ABS: Anti Blockier System.

ALU: Arithmetic Logic Unit

ASIC: Application-Specific Integrated Circuit.

ARM: Advanced RISC Machine

AREF: Pin on the Arduino

ATM: Asynchronous Transfer Mode

AGC: The Apollo Guidance Computer

ASSP: Application Specific Standard Product

CPU: Central Processing Unit.

DSP: Digital Signal Processor.

EEPROM : Electrically Erasable Programmable Read-Only Memory ou mémoire morte effaçable électriquement ET programmable

E/S : Entrée / sortie

GSM: Global System for Mobile Communications

GND: Ground

IDE: Integrated Development Environment

LED: light-emitting diode

OS: Operating System

PC : Personal compture

RISC: Reduced Instruction Set Computer.

RAM: Random Access Memory.

ROM: Read-Only Memory

RTOS: Real-time operating system

Liste des acronymes

SD: Secure Digital

SSD: Solid-state drive

UNO: UN en italien

USB: Universal Serial Bus

Liste des figures

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

Introduction générale

La plupart des gens croient que les vols de voitures peuvent se produire dans un quartier miteux. Les individus, cependant, devraient être conscients afin de ne pas être attirés vers les tricheurs en faisant des erreurs régulières. Le cambriolage de voiture est une vedette parmi les pratiques criminelles les plus reconnus et les plus expérimentés. Cela peut réellement arriver à peu près n'importe quand et n'importe où. Les responsabilités de la propriété physique peuvent être modifiées sans l'assentiment du propriétaire légitime. La seule issue possible à de tels problèmes est de mettre en place un système de sécurité supplémentaire dans la voiture.

Le système devrait fonctionner raisonnablement bien même dans des circonstances défavorables pour atteindre le niveau de sécurité souhaité. Un système de surveillance contre le vol est une stratégie utilisée pour anticiper et informé le propriétaire à travers l'intermédiaire de réseaux sans fil.

Cependant, la possibilité de vols a encouragée les fabricants de systèmes de sécurité à développer une technologie moderne qui améliore la sureté ainsi que la sécurité. De plus, le système de sécurité automobile intelligent doit être rentable et primordiale.

Dans ce contexte, l'objectif de notre travail est de réaliser un système à temps réel permettant la surveillance intelligent des automobiles, ainsi que d'informer le propriétaire et de signaler l'acte de vol.

Dans le premier chapitre, nous allons décrire les systèmes embarqués et les systèmes temps réel, ainsi que leur application dans le domaine de l'automobile.

Au niveau du deuxième chapitre, nous allons présenter les composants utilisés pour la réalisation de notre système, cela en détaillant leurs fonctionnements.

Dans le dernier chapitre, nous allons exposer les parties réalisés du système ainsi que la solution proposé dans ce projet.

CHAPITRE I

LES SYSTEMES EMBARQUES ET L'AUTOMOBILE

Introduction

Un système embarqué est un système électronique et informatique conçu pour contrôler, accéder et traiter des données pour une application bien définit. Un tel système comprend un microcontrôleur mono-puce comme le cortex, ARM et également des microprocesseurs ainsi que des FPGA, des DSP et des circuits ASIC.

Les systèmes embarqués intelligents des automobiles ont évolué au cours des deux dernières décennies. Chaque année, les constructeurs automobiles emballent des systèmes embarqués dans leurs voitures pour différentes fonctionnalités, telles que l'allumage, la sécurité et les systèmes audio. Les innovations technologiques des systèmes embarqués dans le véhicule sont ambitieusement remises en question pour rendre l'efficacité énergétique du véhicule, le réseau intelligent et plus sûr.

Dans ce chapitre, nous allons présenter en détaille les systèmes embarqués et les systèmes temps réels.

I.1 Les systèmes embarqués

I.1.1. Historique

L'un des premiers systèmes modernes embarqués reconnaissables a été le « Apollo Guidance Computer » en 1967 (figure I.1), le système de guidage de la mission lunaire Apollo, développé par Charles Stark Draper du Massachusetts Institute of Technology.

Chaque mission lunaire était équipée de deux systèmes (AGC), un chargé du système de guidage inertiel et un pour le Module lunaire. Au commencement du projet, l'ordinateur AGC d'Apollo était considéré comme l'élément le moins fiable du projet. Par contre grâce à l'utilisation de nouveaux composants qu'étaient à l'époque les circuits intégrés, des gains substantiels sur la place utile et la charge utile ont été réalisés, avec une diminution supposée des risques déjà nombreux des missions [1].

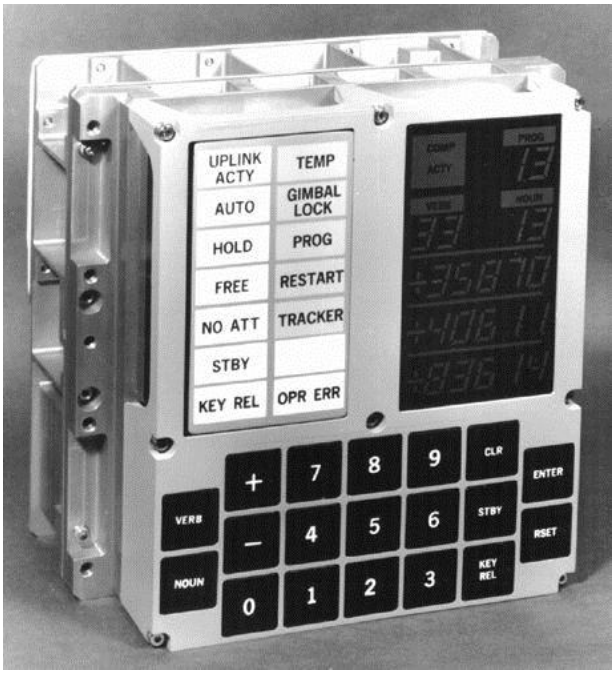


Figure I.1 : Vue du premier système embarqué « Apollo Guidance Computer »

Des dates relatives à l'évolution des systèmes embarquées :

- 1967 : Apollo Guidance Computer, premier système embarqué. Environ un millier de circuits intégrés identiques (portes NAND).
- 1960-1970 : Missile Minuteman, guidé par des circuits intégrés.
- 1971 : Intel produit le 4004, premier microprocesseur, à la demande de Busicom. Premier circuit générique, personnalisable par logiciel.
- 1972 : lancement de l'Intel 8008, premier microprocesseur 8 bits (48 instructions, 800kHz).
- 1974 : lancement du 8080, premier microprocesseur largement diffusé. 8 bits, (64KB d'espace adressable, 2MHz - 3MHz).
- 1978 : création du Z80, processeur 8 bits.
- 1979 : création du MC68000, processeur 16/32 bits.

I.1.2. Définition d'un système embarqué

Un système embarqué (système enfoui / intégré) est une combinaison de matériel informatique et de logiciel, soit fixe, soit programmable, conçu pour une fonction spécifique ou pour des fonctions spécifiques au sein d'un système plus important. Les systèmes embarqués sont des systèmes informatiques, mais peuvent aller de l'absence d'interface utilisateur (par exemple, sur des périphériques dans lesquels le système embarqué est conçu pour exécuter une

seule tâche) à des interfaces utilisateur graphiques complexes, comme dans appareils mobiles. Aussi, les interfaces utilisateur peuvent inclure des boutons, des voyants, la détection d'écran tactile et plus encore. Certains systèmes utilisent également des interfaces utilisateur distantes [2].

I.1.3. Exemples de systèmes embarqués

On trouve les systèmes embarqués aujourd'hui absolument partout, aussi bien dans la vie quotidienne que dans des domaines très spécifiques :

- **Systèmes grand public :** appareils photographiques et caméras, lecteurs DVD, chauffage et climatisation, éclairage, électroménager, etc.
- **Transport :** automobile, marine, maintenance, radar, aéronautique, etc.
- **Défense :** Lanceur, missile, contrôle de trajectoire, etc.
- **Secteur industrie et manufacturier :** chaînes de production, automates, production et distribution d'électricité, réacteurs chimiques, réacteurs nucléaires, raffineries, dispositifs de sécurité, aide à la maintenance, etc.
- **Information et communication :** téléphone, fax, routeurs, imprimante, téléphone mobile, GPS, etc.
- **Santé :** diagnostique, soins, implants, imagerie, etc.
- **Autre :** distributeur de billets, carte à puce, etc.

I.1.4. Architecture des systèmes embarqués

Un système embarqué typique composé de deux parties principales, comme montre la figure I.2 [2] :

- **Matériel embarqué :** Le matériel embarqué comprend principalement le processeur, la mémoire, le bus, les périphériques, les ports d'E / S et divers contrôleurs.
- **Logiciel embarqué :** Le logiciel embarqué contient généralement le système d'exploitation intégré et diverses applications.

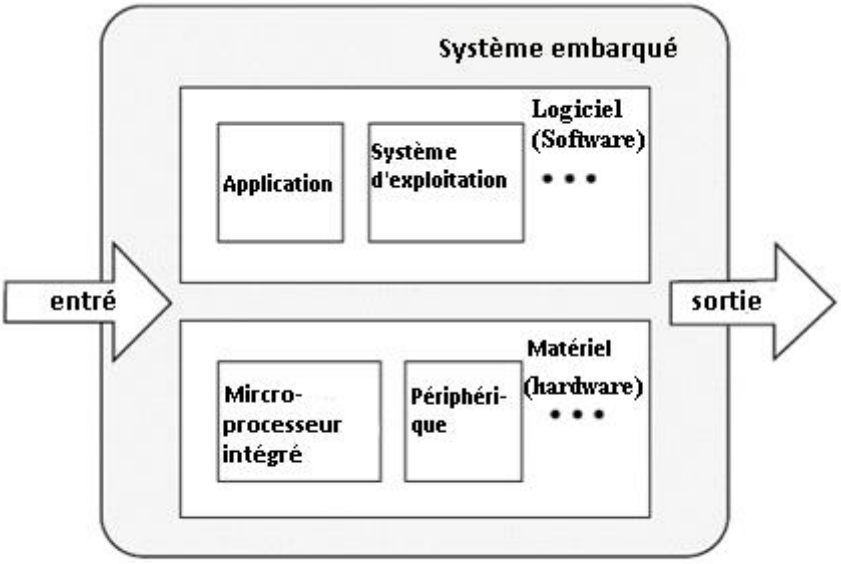


Figure I.2 : Architecture générale d'un système embarqué

L'entrée et la sortie sont des caractéristiques de tout système ouvert, et le système embarqué ne fait pas exception. Dans le système embarqué, le matériel et le logiciel collaborent souvent pour traiter les différents signaux d'entrée provenant de l'extérieur et produire les résultats du traitement sous une forme ou une autre. Le signal d'entrée peut être un dispositif ergonomique (tel qu'un clavier, une souris ou un écran tactile) ou la sortie d'un circuit de capteur dans un autre système embarqué. La sortie peut prendre sous forme de son, de lumière, d'électricité ou d'un autre signal analogique, ou d'un enregistrement ou d'un fichier pour une base de données.

I.1.5. La partie Hardware

Les systèmes embarqués peuvent être basés sur un microprocesseur ou un microcontrôleur. Dans les deux cas, il existe un circuit intégré (CI) au cœur du produit qui est généralement conçu pour effectuer le calcul pour les opérations en temps réel.

Les composants de base du système (microprocesseur, mémoire et modules d'entrée et de sortie) sont interconnectés par un bus système afin que toutes les parties communiquent et exécutent un programme (Figure I.3).

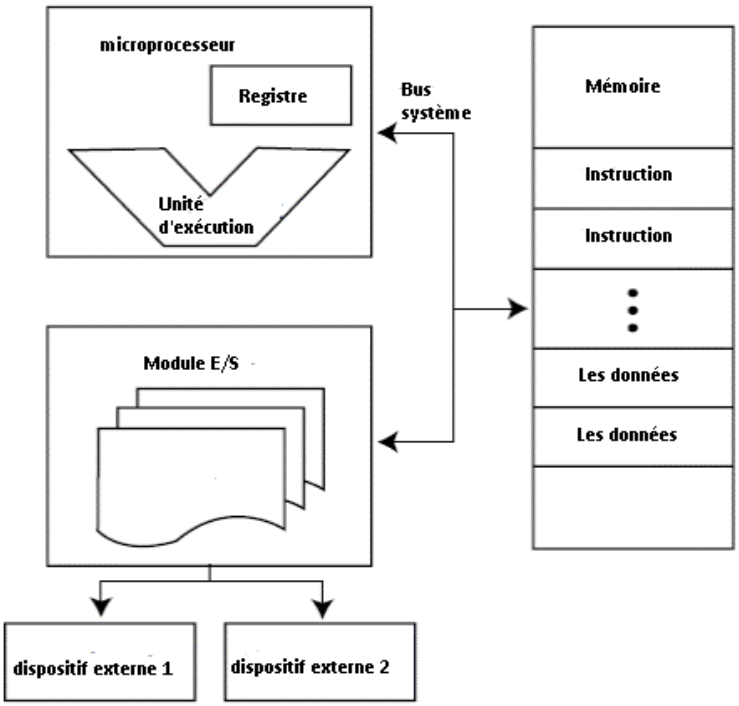


Figure I.3 : Architecture de base d'un système embarqué

Dans les systèmes embarqués, le rôle et la fonction du microprocesseur sont généralement les mêmes que ceux de la CPU dans un ordinateur à usage général. Le fonctionnement de l'ordinateur est la commande, l'exécution des instructions et le traitement des données.

Dans de nombreux cas, le microprocesseur d'un système embarqué est également appelé CPU. La mémoire est utilisée pour stocker des instructions et des données. Les modules d'entrées/sorties sont responsables de l'échange de données entre le processeur, la mémoire et les périphériques externes. Les périphériques externes incluent les périphériques de stockage secondaires (tels que le flash et le disque dur), les équipements de communication et les équipements terminaux.

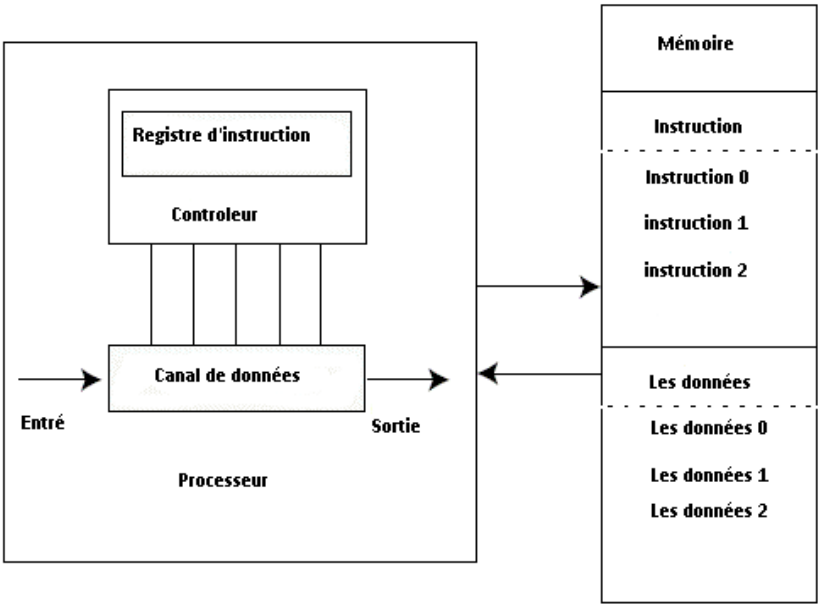
Le bus système fournit des données et contrôle la communication et la transmission du signal pour le processeur, la mémoire et les modules d'entrées/sorties.

Il existe essentiellement deux types d'architecture qui s'appliquent aux systèmes embarqués : l'architecture Von Neumann et l'architecture Harvard.

I.1.5.1. Architecture Von Neumann

L'architecture de Von Neumann (également connue sous le nom d'architecture de Princeton) a été proposée pour la première fois par John Von Neumann.

La caractéristique la plus importante de cette architecture est que les programmes et les données utilisent la même mémoire (comme le montre la figure I.4).



» » » » » »

Figure I.4 : Architecture de Von Neumann

Dans l'architecture Von Neumann, une instruction et des données partagent le même bus. Dans cette architecture, la transmission de l'information devient le goulot d'étranglement des performances de l'ordinateur et affecte la rapidité du traitement des données ; ainsi, il est souvent appelé le goulot d'étranglement Von Neumann. En réalité, la technologie de cache et de prédiction de branche peut résoudre efficacement ce problème.

I.1.5.2. Architecture Harvard

L'architecture de Harvard a été nommée d'abord après l'ordinateur Harvard Mark I.

Comparé à l'architecture de Von Neumann, un processeur d'architecture Harvard a deux caractéristiques exceptionnelles. Premièrement, les instructions et les données sont stockées dans deux modules de mémoire distincts ; les instructions et les données ne coexistent

pas dans le même module. Deuxièmement, deux bus indépendants sont utilisés comme chemins de communication dédiés entre l'unité centrale et la mémoire. De plus, il n'y a pas de connexion entre les deux bus. L'architecture de Harvard est illustrée sur la Figure I.5.

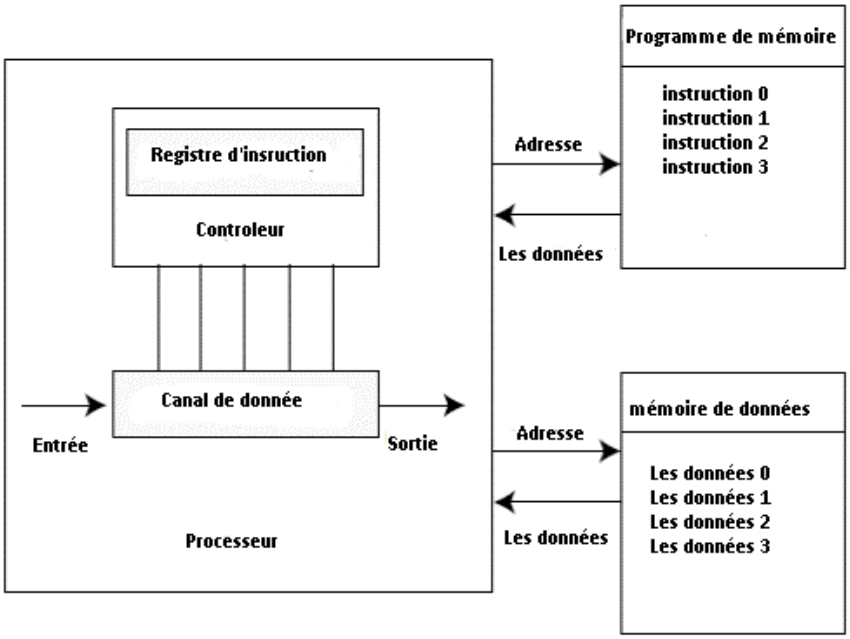


Figure I.5 Architecture de Harvard

Puisque l'architecture de Harvard a une mémoire de programme et une mémoire de données séparées, elle peut fournir une plus grande bande passante de mémoire de données, ce qui en fait le choix idéal pour le traitement de signal numérique.

La plupart des systèmes conçus pour le traitement du signal numérique (DSP) adoptent l'architecture de Harvard. L'architecture de Von Neumann se caractérise par une conception matérielle simple et un stockage flexible des programmes et des données. Elle est généralement choisie pour les systèmes à usage général et les systèmes les plus intégrés. Pour effectuer efficacement des lectures / écritures de mémoire, le processeur n'est pas directement connecté à la mémoire principale, mais à la mémoire cache. Généralement, la seule différence entre l'architecture de Harvard et l'architecture de Von Neumann est le cache simple ou double. Dans l'architecture de Harvard, le cache est souvent divisé en un cache d'instructions (cache I) et un cache de données (cache D), mais l'architecture de Von Neumann à un seul cache [5].

I.1.6. La partie Software

Un microcontrôleur industriel typique est assez simple par rapport à un ordinateur de bureau d'entreprise et dépend généralement d'un environnement de programme plus simple et nécessitant moins de mémoire. Cependant, les appareils les plus simples fonctionnent sur du métal nu (Un environnement nu est un système informatique ou un réseau dans lequel une machine virtuelle est installée directement sur le matériel plutôt que sur le système d'exploitation hôte (OS).) et sont programmés directement en utilisant le langage de code machine de la puce [2].

Souvent les systèmes embarqués utilisent des systèmes d'exploitation ou des plates-formes linguistiques adaptés à l'utilisation intégrée, en particulier lorsque les environnements d'exploitation en temps réel doivent être desservis. À des niveaux plus élevés de capacité de puce, tels que ceux trouvés dans les 'système sur une puce', les concepteurs ont de plus en plus décidé que les systèmes sont généralement assez rapides et tolèrent de légères variations de temps de réaction que les approches en temps quasi réel conviennent. Dans ces cas, des versions allégées du système d'exploitation Linux sont généralement déployées, bien que d'autres systèmes d'exploitation aient été réduits pour fonctionner sur des systèmes embarqués, notamment Embedded Java et Windows Embedded.

Généralement, le stockage de programmes et de systèmes d'exploitation sur des périphériques embarqués utilise soit la mémoire flash, soit la mémoire flash réinscriptible.

Le logiciel embarqué est la programmation spécialisée dans une puce ou sur le Firmware dans un dispositif embarqué pour contrôler ses fonctions. Les fabricants de matériel utilisent des logiciels intégrés pour contrôler les fonctions de divers périphériques et systèmes matériels. Le logiciel embarqué contrôle les fonctions de l'appareil de la même manière que le système d'exploitation d'un ordinateur contrôle la fonction des applications logicielles. N'importe quel appareil peut contenir des logiciels embarqués de ceux si simples que vous n'imaginez pas qu'ils ont un contrôle informatique, comme des grille-pain et des ampoules, à des systèmes de suivi complexes dans les missiles.

Le logiciel embarqué varie en complexité autant les périphériques qu'il est utilisé pour contrôler. Bien que le terme soit souvent utilisé de manière interchangeable avec le Micro-logiciel, le logiciel embarqué est souvent le seul code informatique fonctionnant sur un matériel, alors que le Micro-logiciel confie le contrôle à un système d'exploitation qui à son tour lance et contrôle les programmes [6].

I.1.6.1. Les langages de programmation des systèmes embarqués

Pour la programmation des systèmes embarquée, il faut noter que plusieurs langages de programmation se veulent dédiés à l'embarqué. Parmi ces langages se trouvent « Ada » et le langage Assembleur, ce dernier restant encore un choix approprié pour les systèmes soumis à des contraintes sévères de temps réel. Des langages proches de la machine comme le langage C et dans une moindre mesure le C++ sont aussi utilisés [7].

Le tableau 1.1 présente l'ensemble des langages considérés dans les systèmes embarqués.

Tableau I.1 Les langages les plus utilisés pour le développement de systèmes embarqués.

Classement	Langage	Catégorie
1	C	Mobile Entreprise Système embarqué
2	C++	Mobile Entreprise Système embarqué
3	Arduino	Système embarqué
4	Assembly	Système embarqué
5	Haskell	Entreprise Système embarqué
6	D	Web Système embarqué
7	LabView	Entreprise Système embarqué
8	VHDL	Système embarqué
9	Ladder Logic	Système embarqué
10	Erlang	Entreprise Système embarqué
11	Verilog	Système embarqué
12	Ada	Entreprise Système embarqué
13	TCL	Entreprise Système embarqué
14	Forth	Système embarqué

Les langages C et C++ confortent également leur place respective dans le dernier classement de l'IEEE des meilleurs langages pour les systèmes embarqués. On voit aussi d'autres langages tels qu'Arduino, Haskell, D, LabVIEW et VHDL qui sont bien classés.

En dehors de certains langages populaires (à usage général) qui reviennent dans les deux classements, il peut être important de donner quelques précisions sur les différents langages :

- **Arduino :** il s'agit du langage natif pour le microcontrôleur Arduino, qui est devenu la base d'un grand nombre de dispositifs de fabrication et de prototypage.
- **LabView :** créé par National Instruments, LabView est conçu pour l'acquisition de données et le contrôle industriel.
- **VHDL :** VHSIC Hardware Description Language (VHDL) est un langage de description matériel utilisé dans la création et l'analyse de circuits électroniques ;
- **Ladder Logic :** il s'agit d'un langage de programmation destiné au développement de contrôleurs logiques programmables industriels.
- **Erlang :** créé par Ericsson pour les applications de téléphonie embarquées, la publication d'Erlang en tant que langage open source en 1998 a renforcé sa popularité parmi les programmeurs qui développent des applications qui doivent gérer de nombreuses tâches simultanées.
- **Verilog :** comme VHDL, Verilog (ou Verilog HDL) est un langage de description matériel utilisé dans la création et l'analyse de circuits électroniques.
- **Ada :** à l'origine conçu pour le département de défense des États-Unis, Ada est utilisé pour des applications où la fiabilité est critique, comme les systèmes de contrôle aérospatial.
- **TCL :** il s'agit d'un langage de script destiné au prototypage rapide et supportant l'interface utilisateur graphique Tk utilisée principalement avec les systèmes Unix ;
- **Forth :** conçu à l'origine pour contrôler les radiotélescopes, Forth est toujours utilisé pour des applications telles que les boots loaders et d'autres firmwares.
- **Scade :** il s'agit d'un langage pour l'embarqué critique. C'est le langage de modélisation de SCADE Suite, un environnement de développement intégré pour la conception de systèmes critiques.

I.1.7. La classification des systèmes embarqués

On peut classifier les systèmes embarqués en fonction du domaine et de l'application envisagés [21].

I.1.7.1. Selon le type d'application visée

On peut distinguer quatre principaux types de systèmes embarqués en fonction du type d'application visé :

- **Usage général :** Les systèmes embarqués à usage général exécutent des applications similaires à celles exécutées sur des ordinateurs « traditionnels », mais ils sont embarqués dans des packages de petite taille. On a, par exemple, les assistants personnels (PDA) et les guichets automatiques bancaires (ATM).
- **Les systèmes embarqués de contrôle :** sont utilisés pour effectuer un contrôle en boucle rétroactive fermée de systèmes temps réel. On les retrouve notamment dans les moteurs de voiture, les centrales nucléaires et pour le contrôle aérien.
- **Les systèmes embarqués pour traiter des signaux :** sont utilisées pour réaliser des calculs sur des gros flux de données. Ces derniers se retrouvent typiquement dans le traitement audio et vidéo et dans les radars et sonars.
- **Communications et réseaux :** les systèmes embarqués sont également utilisés dans le domaine des communications et réseaux, pour effectuer de la transmission de données et réaliser des communications. Ils sont notamment utilisés dans la téléphonie et pour l'internet.

Ce premier classement caractérise les systèmes embarqués en fonction du type d'application pour lequel ils sont conçus et utilisés. Certains appareils peuvent, de prime abord, se retrouver dans plusieurs catégories. Par exemple, une console portable de jeu est à la fois un système embarqué à usage général puisqu'elle peut exécuter des jeux comme un ordinateur « *traditionnel* », mais elle peut également être spécialisée dans l'audio et la vidéo. Il faut aussi garder à l'esprit qu'un appareil peut contenir plusieurs systèmes embarqués, comme l'exemple de la voiture précédemment évoqué.

I.1.7.2. Selon le type de calculs

On peut également classer les systèmes embarqués selon le type de fonction qu'ils calculent :

- **Loi de contrôle :** Un système embarqué peut avoir pour but de s'assurer qu'une loi de contrôle soit respectée. Par exemple, un drone est normalement équipé d'un contrôleur PID (un contrôleur PID, Proportional-Integral-Derivative, calcule une valeur d'erreur entre un objectif et une valeur mesurée et produit des commandes à exécuter sur le système afin que celui-ci se rapproche de l'objectif) qui permet d'assurer sa stabilité en vol stationnaire.
- **Logique séquentielle :** Une autre fonction possible, par exemple présente dans une machine à laver, consiste à représenter une logique séquentielle, alternant entre différents modes de fonctionnement.
- **Traitement de signaux :** On retrouve également le traitement de signaux comme une fonction possible. Des systèmes embarqués sont spécialisés dans la compression et décompression de contenu multimédia, l'application de filtres pour des sons, etc.
- **Interface :** Des systèmes embarqués sont également utilisés pour jouer le rôle d'interface pour des applications spécifiques. Ils vont ainsi gérer des boutons, lampes ou alarmes sonores ou être utilisés pour gérer des entrées-sorties rapides comme dans des contrôleurs de disques durs ou SSD, par exemple.
- **Réagir a des fautes :** il existe des systèmes embarqués dont le but est de réagir à des fautes. Leur fonction est simple tant que tout va bien, ils ne font juste rien. Mais lorsqu'un comportement anormal est détecté, ils réalisent un diagnostic et tentent de corriger la faute.

I.1.7.3. Selon leur la taille et la complexité

Si on se penche plutôt sur des considérations plus techniques, on peut également classer les systèmes embarqués selon leur taille et complexité [3] :

- **Petite échelle :** Un système embarqué de petite échelle comporte typiquement un simple microcontrôleur sur 8 ou 16 bits, avec du hardware et software très simple. Le design de ces systèmes se retrouve au niveau d'une carte et les ressources, notamment en termes de mémoire, sont très limitées pour limiter la dissipation de puissance.

- **Moyenne échelle :** Un système embarqué de moyenne échelle comporte un ou quelques microcontrôleurs sur 16 ou 32 bits, un Digital Signal Processor (processeur de signal numérique) (DSP) ou un processeur avec un jeu d'instructions réduit (RISC). Le hardware et software de ces systèmes sont plus complexes. Côté hardware, on retrouve des Application Specific Standard Product (produit standard spécifique à une application, par exemple pour smartphones) (ASSP) ou Instruction Processor (IP) et ils peuvent être contrôlés par des Real-Time Operating Systems (systèmes d'exploitation temps réel) (RTOS).
- **Sophistiqué :** On peut aussi concevoir des systèmes embarqués sophistiqués qui ont une complexité hardware et software très grande. Ils sont construits à partie de processeurs évolutifs et configurables ou de Programmable Logic Array (PLA). Pour ces systèmes, un co-design et une intégration hardware/software sont importants.

I.1.8. Conception d'un système embarqué

Tout d'abord, un système embarqué doit être fiable, notamment car il doit généralement pouvoir fonctionner en autonomie. En particulier, on peut mesurer :

- sa **robustesse**, c'est-à-dire la probabilité qu'il fonctionne correctement à un certain temps t, sachant qu'il fonctionnait en t=0.
- sa **maintenabilité**, c'est-à-dire la probabilité qu'il fonctionne correctement un certain temps d'après qu'une erreur se soit produite.
- sa **disponibilité**, c'est-à-dire la probabilité qu'il fonctionne correctement à un temps t
- sa **sûreté**, c'est-à-dire qu'il ne cause pas de nuisances.
- sa **sécurité**, c'est-à-dire que ses communications sont confidentielles et authentifiées.

Il faut bien entendu maximiser tous ces critères. De plus, il est important de directement penser à ces aspects lors de la conception du système embarqué car il n'est pas forcément évident d'en modifier un existant après coup pour mieux les satisfaire. En effet, étant donné les contraintes du système embarqué, notamment en termes de puissance de calcul, il ne sera pas forcément possible de tous les maximiser. Un choix devra donc être posé selon l'application visée.

Un système embarqué doit également être efficace. Étant donné qu'il est conçu dans un but spécifique et précis, il faut qu'il soit plus efficace qu'un ordinateur « traditionnel », sans quoi son développement n'est pas justifié. Les aspects suivants doivent être pris en compte :

- **la consommation énergétique**.
- **la taille du code**, en particulier pour des systèmes embarqués de bas niveau déployés sur une petite puce.
- **l'exécution** du programme tant au niveau du temps que de la mémoire consommé.
- **le poids et la taille.**
- **le cout** de fabrication, mais aussi de conception et design.

Le tableau I.5.1 reprend ces différents aspects et propose une ou plusieurs mesures permettant de les évaluer. Ainsi, on peut mesurer la taille d'un code directement en octets, plutôt pour les petits systèmes embarqués de très bas niveau, ou en lignes de code pour ceux de haut niveau. Pour les performances de l'exécution du programme, on peut mesurer le nombre de millions d'instructions exécutées par seconde (MIPS) pour évaluer la vitesse de calcul « pure » ou le nombre d'opérations de lecture/écriture par seconde dans la mémoire, si le système embarqué doit manipuler beaucoup d'entrées/sorties.

Tableau I.2 : Les critères d'efficacité d'un système embarqué peuvent être mesurés à l'aide de différentes métriques que l'on peut combiner entre elles.

Critère	Mesure
Consommation énergétique	Watt
Taille du code	Octets Lignes de codes
Exécution du programme	MIPS Nombre de Read/Write par seconde
Poids et taille	Gramme, millimètre
Cout de fabrication	$ / €

Toutes ces caractéristiques, de fiabilité et d'efficacité, doivent évidemment être considérées en lien avec l'application pour laquelle le système embarqué est développé. En effet, certains critères ont sans doute moins de sens ou sont moins importants pour une application donnée. Par exemple, les considérations énergétiques sont sans doute moins cruciales pour le système embarqué qui gère votre machine à laver que pour celui qui est installé dans un volcan pour monitorer le niveau des émanations de dioxy de soufre (SO2) [3].

I.1.9. La fiabilité des systèmes embarqués

Les systèmes embarqués sont la plupart du temps dans des machines qui doivent fonctionner en continu pendant de nombreuses années, sans erreurs et, dans certains cas, tolérance aux fautes système fonctionne en présence d'erreur (système redondant). C'est pourquoi les logiciels sont toujours développés et testés avec plus d'attention que ceux pour les PC. Les pièces mobiles non fiables (par exemple les lecteurs de disques, boutons ou commutateurs) sont proscrites [8].

La question de la fiabilité peut inclure :

- Le système ne peut pas être éteint pour des réparations ou ce sont des réparations inaccessibles.

La solution peut être des pièces détachées supplémentaires ou un "mode mou" du logiciel qui fournit un fonctionnement partiel.

Par exemple : les câbles sous-marins, les balises de navigation, les puits de forage…

- Le système doit rester en marche pour des raisons de sécurité. Souvent, les sauvegardes sont effectuées par un opérateur.

Dans ce cas, le « mode mou » est toléré.

Par exemple : les systèmes de contrôle des réacteurs, les usines chimiques, les signaux de train…

- Un arrêt du système peut provoquer des pertes monétaires énormes s'il s'éteint.

Par exemple : les systèmes de ponts ou d'ascenseurs, les transferts de fond, les salles de bourse, les ventes ou services automatiques.

I.1.10. Les contraintes des systèmes embarqués

Les systèmes embarqués sont soumis à de nombreuses contraintes en fonction de leur domaine d'application, notamment ceux en temps réel puisque la contrainte temporelle prend une place prépondérante. Ainsi, en plus des différentes contraintes liées à l'embarquabilité, la contrainte temporelle devient prioritaire et le système embarqué doit donc échanger des données, en temps réel, si besoin. Parmi celles-ci, nous retrouvons des contraintes de [4] :

- **Dépendance** : fiabilité, maintenabilité, sécurité.
- **Efficacité** : consommation d'énergie, taille du programme, poids, coût, temps d'exécution.
- **Matérielle** : taille, poids, chaleur dégagée, processeur, mémoire disponible.

Ces contraintes doivent être prises en compte dès le début, car elles influent grandement sur la conception des systèmes embarqués et sur ses capacités futures. Malgré l'évolution de la technologie, ces contraintes restent importantes pour garantir aux entreprises une rentabilité par rapport au matériel utilisé. En effet, le coût et la performance des composants sont des éléments importants pour les entreprises dans le choix des systèmes embarqués, malgré une évidente amélioration de ces facteurs récemment. C'est pourquoi nous observons que la majorité des systèmes embarqués sont :

- Réactifs et capables de faire des traitements en temps réel.
- Petits et légers.
- Peuvent (selon les cas) posséder une batterie de grande capacité.
- Peuvent (selon les cas) résister à des conditions difficiles telles que des chocs, des vibrations, des interférences.

I.2 Les systèmes temps réel

Les systèmes embarqués sont généralement des systèmes temps réel et des systèmes informatiques soumis à des contraintes de temps [9].

I.2.1. Définition des systèmes temps réel

Un système est dit temps réel si l'exactitude des applications ne dépend pas seulement du résultat mais aussi du temps auquel ces résultats sont produits.

Une autre définition du temps réel, un système est dit temps réel s'il doit s'exécuter suffisamment vite par rapport à la dynamique du procédé contrôlé. La notion relative de vitesse

s'exprime par des contraintes temporelles. La notion « suffisamment vite » est relative à la dynamique du procédé contrôlé. Par exemple, le drone de « la figure II.1 », est considéré comme un système temps réel. De par sa forme d'aile delta, il est très instable en roulis ; par conséquent, il est important que la commande de vol se rafraîchisse fréquemment, par exemple toutes les 20 millisecondes.

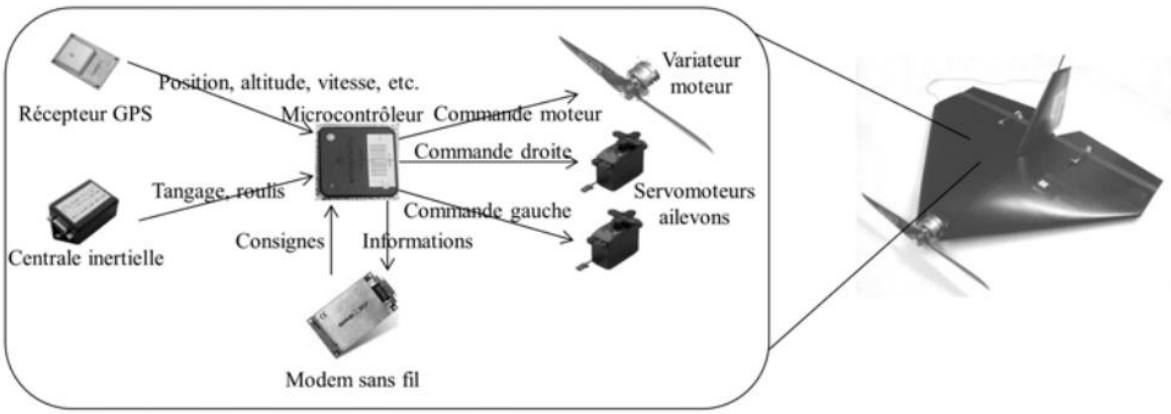


Figure I.6 Un exemple sur un système temps réel : un drone

I.2.2. Classification des systèmes temps réel

L'exactitude d'un calcul dépend non seulement de ses résultats, mais aussi du moment auquel ses sorties sont générées, un système en temps réel doit satisfaire à des contraintes de temps de réponse bornées ou subir de graves conséquences [10].

I.2.2.1. Temps réel mou

On parle de temps réel mou quand les événements traités trop tardivement ou perdus sont sans conséquence catastrophique pour la bonne marche du système. Les fautes temporelles sont tolérables (Ex. : jeux vidéo, applications multimédias, téléphonie mobile…).

I.2.2.2. Temps réel dur

On parle de temps réel dur quand les événements traités trop tardivement ou perdus provoquent des conséquences catastrophiques pour la bonne marche du système**.** Les fautes temporelles ne sont pas tolérables (Ex. : fonctions critiques avionique, véhicules spatiaux, automobile, transport ferroviaire…).

I.2.2.3. Temps réel ferme

Un retard, s'il arrive très peu souvent, peut être toléré .Les fautes temporelles sont autorisées dans une certaine limite, par exemple une erreur toutes les trois exécutions au plus exemple : téléphone.

I.2.3. Les modèles d'exécution en temps réel

Les événements en temps réel appartiennent à l'une des trois catégories : asynchrone, synchrone ou isochrone.

I.2.3.1. Les modèles asynchrones

Sont entièrement imprévisibles. Par exemple, l'événement qu'un utilisateur effectue un appel téléphonique. En ce qui concerne la compagnie de téléphone, l'action de passer un appel téléphonique ne peut pas être prédite.

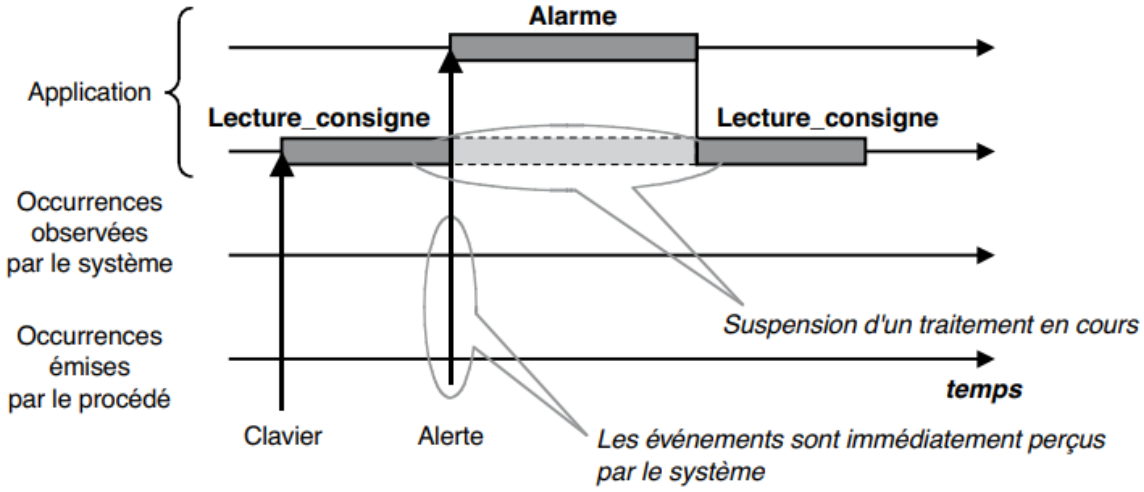


Figure I.7 Modèle d'exécution asynchrone

I.2.3.2. Les modèles synchrones

Sont prévisibles et se produisent avec une régularité précise s'ils doivent se produire. Par exemple, l'audio et la vidéo dans un film se déroulent de manière synchrone.

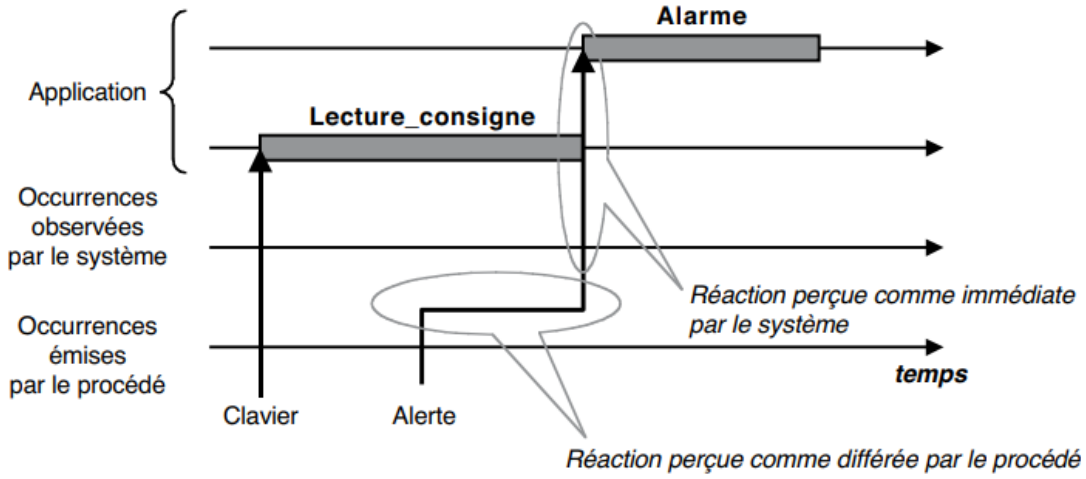


Figure I.8 Modèle d'exécution synchrone

I.2.3.5. Les modèles isochrones

Se produisent avec régularité dans une fenêtre de temps donnée. Par exemple, les octets audio d'une application multimédia distribuée doivent apparaître dans une fenêtre de temps lorsque le flux vidéo correspondant arrive. Isochrone est une sous-classe d'asynchrone.

I.2.4. Caractéristiques des systèmes en temps réel

Les systèmes en temps réel ont de nombreuses caractéristiques particulières qui sont inhérentes ou imposées. Cette section traitera de certaines de ces caractéristiques importantes.

I.2.4.1. Grand et complexe

La plupart des problèmes associés au développement de logiciels sont ceux liés à la taille et à la complexité. Rédaction de petits programmes ne présente pas de problème important, car ils peuvent être conçus, codés, maintenus et compris par une seule personne. Cette grandeur est liée principalement à la variété. La variété est celle des besoins et des activités dans le monde réel et leur reflet dans un programme. Le monde réel change continuellement. Il évolue. Il en

va de même pour les besoins et les activités de la société. Ainsi, les grands programmes, comme tous les systèmes complexes, doivent évoluer continuellement. Les programmes logiciels tendent à montrer la propriété indésirable de la taille. Ceci est principalement dû au changement continu.

Les systèmes en temps réel subissent une maintenance constante et des améliorations au cours de leur vie. Ils doivent donc être extensibles.

I.2.4.2. Manipulation de nombres réels

De nombreux systèmes en temps réel impliquent le contrôle de certaines activités d'ingénierie. Par exemple, considérons le modèle d'une usine dans la figure. Dans cet exemple, l'usine est l'entité contrôlée. L'usine produit un vecteur de variables de sortie qui changent avec le temps. Ces sorties sont comparées à un signal souhaité ou de référence pour produire un signal d'erreur. Le contrôleur utilise ensuite le signal d'erreur pour modifier les variables d'entrée.

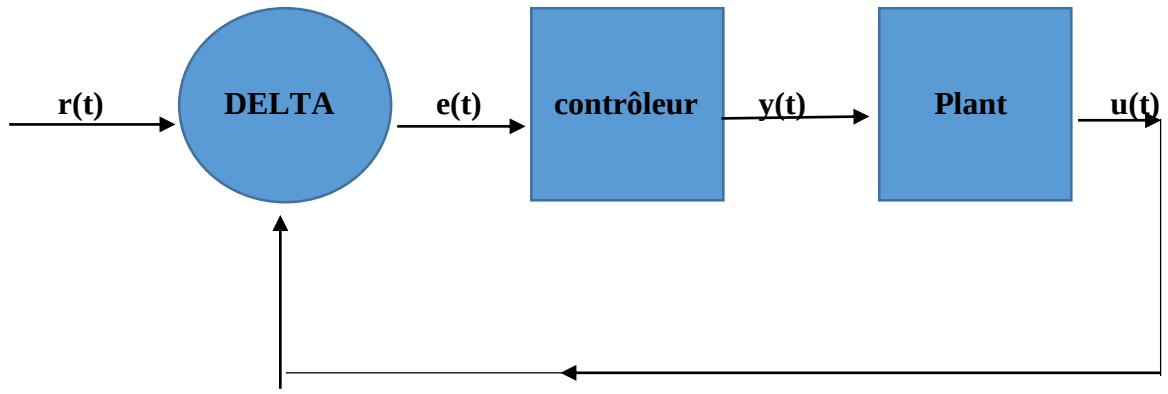


Figure I.9. Un contrôleur en temps réel simple

Un modèle mathématique de ce système est basé sur des équations différentielles de premier ordre. La sortie du système est liée à l'état interne du système et à ses variables d'entrée. Une exigence en temps réel de ce système est de passer à un nouvel ensemble de points dans une

période de temps fixe. Cela ajoute à la complexité des calculs. C'est une des raisons pour lesquelles les systèmes en temps réel peuvent être si complexes.

I.2.4.3. Fiable et sûr

Plus la société abandonne le contrôle de ses fonctions vitales aux ordinateurs, plus il devient impératif que ces ordinateurs n'échouent pas. Une défaillance dans un distributeur automatique de billets peut entraîner la perte irrémédiable de millions de dollars. Une composante défectueuse de la production d'électricité pourrait faire échouer un système de support de vie dans une unité de soins intensifs. Dans les environnements hostiles tels que les militaires, les systèmes doivent pouvoir échouer de manière contrôlée. Pour l'interaction de l'opérateur, nous devons minimiser la possibilité d'erreur humaine. La taille et la complexité des systèmes en temps réel exacerbent le problème de fiabilité. Toutes les difficultés attendues inhérentes à l'application doivent être prises en compte (y compris celles introduites par une conception de logiciel défectueuse).

I.2.4.4. Contrôle simultané des composants du système séparés

Un système embarqué en temps réel typique comprend des ordinateurs, des capteurs et des actionneurs. Il existe généralement plusieurs éléments externes coexistant avec lesquels l'ordinateur doit interagir simultanément. La nature même de ces éléments externes est qu'ils existent en parallèle. Les actions effectuées par l'ordinateur doivent être exécutées en séquence mais donnent l'allusion d'être simultanées. Dans certains cas, ce n'est pas possible. Un exemple de ceci est des données qui doivent être collectées et traitées à différents points géographiques. Dans ce cas, un système multiprocesseur distribué doit être utilisé [9].

Un problème majeur pour les systèmes qui doivent présenter une simultanéité est comment exprimer cette concurrence dans la structure du programme. Dans le passé, c'était au programmeur de faire face à ces problèmes. Les systèmes seraient conçus pour impliquer l'exécution cyclique d'une séquence de programme pour gérer les diverses tâches simultanées. Ce n'était pas conseillé car cela complique la tâche des programmeurs et oblige à considérer des structures qui ne sont pas pertinentes pour le contrôle des tâches en cours. Les programmes qui en résulteront seront plus obscurs et inélégants. Cela rend plus difficile la démonstration d'un programme correct. Cela rend également la décomposition du problème plus complexe. En outre, l'exécution parallèle du programme sur plus d'un processeur sera beaucoup plus difficile à réaliser, et le placement du code pour traiter les défauts devient plus problématique.

Nous aborderons plusieurs approches pour traiter ces problèmes dans le chapitre sur les systèmes d'exploitation en temps réel.

I.2.5. Ordonnancement de taches

Cette architecture logicielle peut être vue comme un ensemble de tâches synchronisées, communicantes et partageant des ressources critiques. Le rôle essentiel du système informatique est donc de gérer l'enchaînement et la concurrence des tâches en optimisant l'occupation du processeur, cette fonction est appelée l'ordonnancement. Ce principe d'ordonnancement est un point crucial des systèmes de contrôle commande ; en effet l'ordonnancement va déterminer les caractéristiques temporelles et être le garant du respect des contraintes de temps imposées à l'exécution de l'application [12].

L'ordonnancement dans le cas des systèmes temps réel à contraintes temporelles strictes a pour objectif principal de répondre aux deux cas suivants :

- **Fautes temporelles** : cela correspond à un non-respect d'une contrainte temporelle associée à une tâche comme le dépassement de la date limite d'exécution ou échéance. Cela induit la notion d'urgence d'une tâche.
- **Surcharge** : lors de l'occurrence d'une ou plusieurs fautes temporelles, l'ordonnanceur peut réagir en supprimant une ou plusieurs tâches de l'application, ce qui amène à la notion d'importance, c'est-à-dire le choix d'une tâche à exécuter par rapport aux spécifications fonctionnelles de l'application.

I.2.5.1. Analyse de l'ordonnancement hors ligne ou en ligne

L'analyse de l'ordonnancement se devise en deux catégories :

- **Une analyse de l'ordonnancement hors ligne** correspond à la construction d'une séquence d'exécution complète sur la base des paramètres temporels des tâches en utilisant une modélisation (réseaux de Petr, etc.) ou une simulation (animation ou énumération du modèle). L'ordonnanceur nécessaire est minimal puisque la séquence d'exécution est prédéfinie, il se réduit à un séquenceur. En revanche, l'application ainsi figée est peu flexible.

- **Une analyse de l'ordonnancement en ligne** correspond à un choix dynamique de la prochaine tâche à exécuter en fonction des paramètres de la tâche en utilisant une modélisation de l'algorithme d'ordonnancement et une simulation de l'exécution. L'ordonnancement a un coût temporel non négligeable ; en

revanche, l'application peut réagir à des événements ou des situations non prévus.

I.3 Système embarqué dans l'automobile

La première utilisation d'un ordinateur dans une voiture était dans le but de contrôler le moteur. Les constructeurs automobiles ont commencé à introduire les premières versions de systèmes contrôlés par ordinateur pour exécuter une fonction spécifique ; En 1968, Volkswagen a introduit le premier système électronique d'injection de carburant commandé par ordinateur (EFI) fabriqué par Bosch [11].

I.3.1. Systèmes embarqués dans l'automobile

Les systèmes mécaniques dans les automobiles sont largement remplacés par des systèmes électroniques. Aujourd'hui, l'industrie automobile fait grand usage des systèmes embarqués. Qu'il s'agisse de commandes d'essuie-glace ou de commandes de freins antiblocages complexes ou de coussins gonflables, les systèmes embarqués ont acquis le contrôle global des automobiles récentes. Les voitures construites autour de microcontrôleurs, de processeurs de signaux numériques ou utilisant les deux processeurs sont communément appelées unités de contrôle électronique. Aujourd'hui, de nombreuses voitures BMW et véhicules de luxe proposent un grand nombre de contrôleurs embarqués. Le premier système embarqué Volkswagen est venu en 1968.

Certaines des tendances actuelles des systèmes embarqués dans les automobiles comprennent les contrôleurs d'airbag, les systèmes de navigation, la radio par satellite, le régulateur de vitesse adaptatif, la conduite par câble, les écrans tête haute etc.

I.3.2. Système de freinage antiblocage

Le système de freinage antiblocage est utilisé dans les automobiles pour éviter que les véhicules ne dérapent surtout sur une route glissante. Ce système permet aux roues du véhicule d'avoir un meilleur contact avec la route. Ce système se compose essentiellement de capteurs pour suivre la vitesse, les vannes, la pompe et un contrôleur.

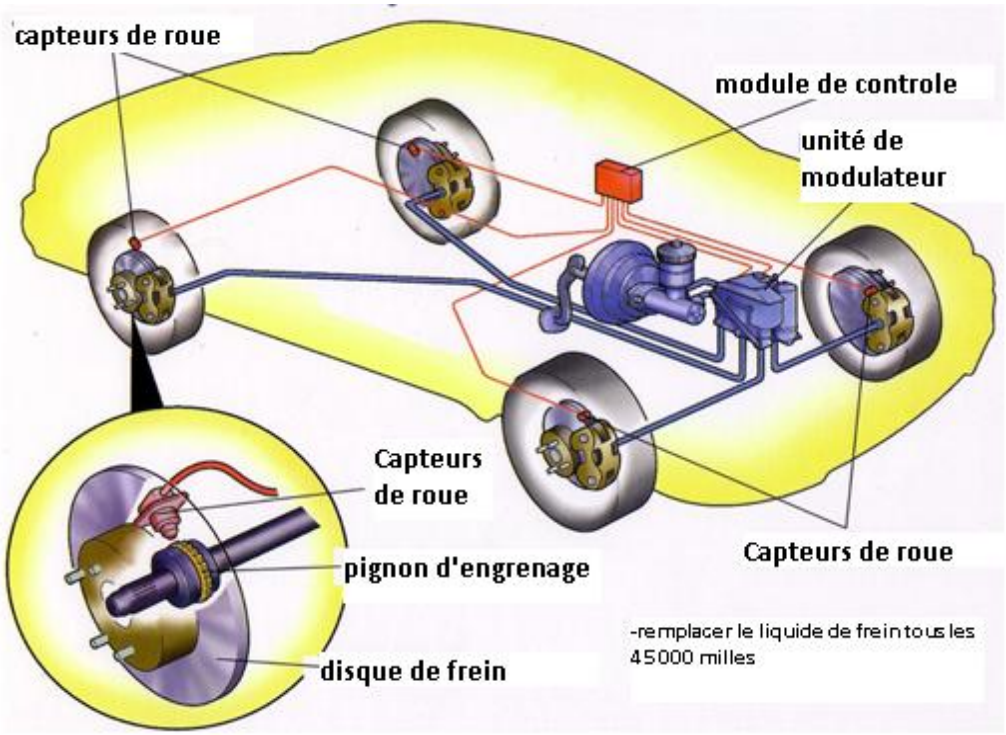


Figure I.10 Système de freinage anti-blocage

Les pièces de système de frein anti-blocage dans les automobiles :

- **Capteurs de vitesse :** Aide à connaître la décélération et l'accélération d'un véhicule.
- **Vannes :** Sur la ligne de frein des freins du véhicule, il y a des vannes. Les vannes sont placées dans 3 positions. Dans la première position, la soupape est maintenue ouverte pour faire passer la pression du maître-cylindre au frein. Dans la seconde position, la soupape est maintenue en position fermée, de sorte qu'une plus grande pression n'atteint pas et le conducteur n'a pas besoin de tirer plus fort sur la pédale de frein. Et enfin dans la position trois, la pression du frein est relâchée à travers cette valve.
- **Pompe :** redonne la pression au frein lorsque la vanne est relâchée.
- **Contrôleur :** Connu sous le nom de contrôleur de frein antiblocage surveille les capteurs et les vannes.

Il y a une unité de contrôle électronique dans le système qui surveille le mouvement de la roue. Si une roue dans les automobiles ralentit, les capteurs de vitesse diront aux soupapes

de réduire la pression sur le frein et ainsi la roue tourne plus vite. D'autre part, si la roue va plus vite, la pression sur la roue est augmentée, ralentissant ainsi la roue.

I.3.3. Système de contrôle d'airbag dans Automobiles

Les airbags sont généralement conçus pour se gonfler dans le cas d'impacts frontaux dans les automobiles. Lorsque le processus de collision se produit, un courant électrique est envoyé au système d'allumage. Ce courant électrique continue à chauffer le filament et allume ainsi la capsule qui enflamme à son tour les pastilles et génère le gaz. Lorsque le gaz se dilate, les coussins gonflables se gonflent également. Tout cela se produit dans un délai de 0,1 sec.

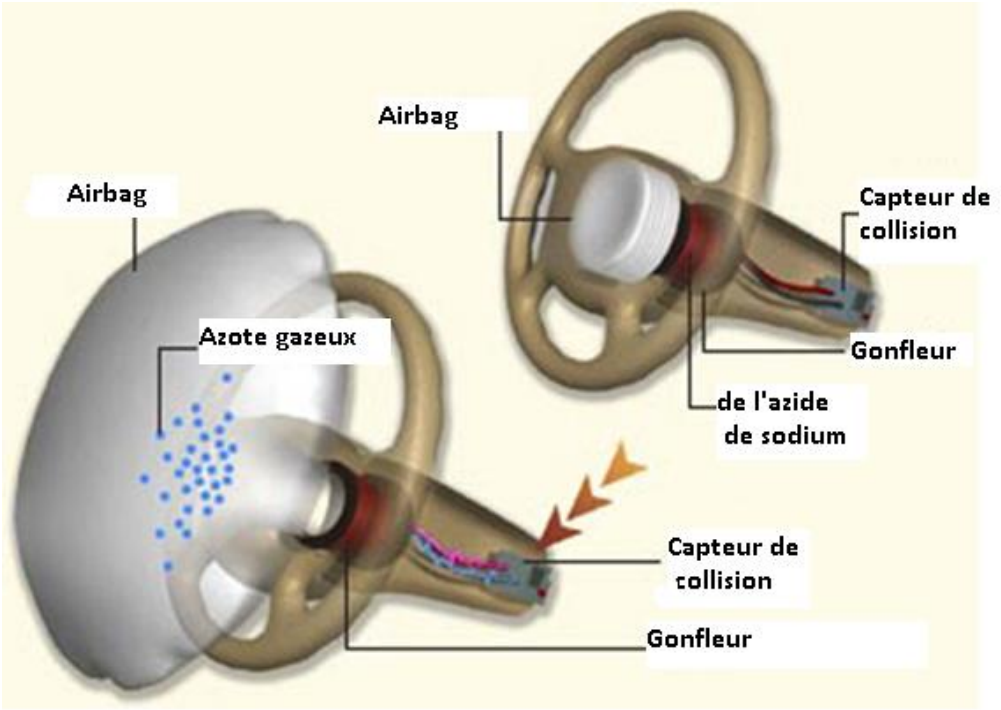


Figure I.11. Système de contrôle d'airbag

I.3.4. Systèmes de navigation

Les systèmes de navigation dans les automobiles gagnent une grande popularité. Ces systèmes sont constitués de différentes fonctions qui peuvent aider un homme commun. Ces systèmes reçoivent les signaux provenant des satellites et permettent de connaître la position et la direction du véhicule. Ce système consiste essentiellement en [11] :

- Récepteur GPS et antenne.
- Capteur
- Écran

- Base de données cartographique
- Ordinateur de navigation

GPS (Global Positioning System) L'antenne et le récepteur sont utilisés pour obtenir les informations des satellites et pour connaître la position du véhicule. Le capteur utilisé est essentiellement de deux types. Ce sont des capteurs de vitesse et de direction. **Les capteurs de vitesse** permettent de connaître la distance de déplacement du véhicule. Alors que le **capteur de direction** continue à détecter la direction du véhicule. Ce système utilise **un écran** à des fins d'affichage. L'ordinateur aide à vérifier les informations de l'antenne, les capteurs avec la base de données cartographiques et affiche sur l'écran à l'aide d'un circuit.

Figure I.12. Système de navigation dans une voiture

I.3.5. Système de contrôle du vol de véhicules utilisant des systèmes embarqués

À l'heure actuelle, le taux de criminalité augmente rapidement parce que c'est une preuve évidente du fait que les vols sont devenus une habitude. En particulier, ces véhicules peuvent subir d'énormes pertes de la part investie dans ces véhicules. Pour surmonter ce problème, il existe de nombreuses technologies disponibles sur le marché telles que les systèmes GPS, GSM et GPRS. De nos jours, la plupart des véhicules sont conçus avec des systèmes de contrôle antivol basés sur le GSM, ce qui assure une protection contre les vols même s'ils sont stationnés dans le parking. Cet article mentionne le système anti-vol pour les voitures, qui comprennent :

LoJack, OnStar, Security Plus et BMW Assist, Commando FM870, Car Shield, Viper 1002, Cobra Track5, Cobra 8510, Nissan Vision 2015 et VIN shield.

Figure I.13. Système antivol et de suivi intelligent pour les automobiles

I.3.6. Système de contrôle du vol de véhicules utilisant des systèmes GSM et GPS

Ce système de suivi de GSM et GPS pour le projet de voiture réduisent considérablement le temps, la main d'œuvre et fonctionne sans interférence d'humanoïde. Dans le monde moderne, il existe diverses technologies nouvelles comme le GPS, le GSM, la RFID, la reconnaissance biométrique. La communication mobile a été intégrée dans les véhicules à des fins de sécurité. Dans ces projets, la technologie GPS est utilisée pour trouver l'emplacement exact du véhicule et le GSM est utilisé pour envoyer le message au propriétaire du véhicule. À la fois si le véhicule semble être le vol, le propriétaire doit juste envoyer un SMS à ce véhicule, cela signifie qu'un véhicule sera arrêté toutes les portes seront fermées alors le vol sera bloqué dans la voiture.

Ainsi, ce sont des projets de système de contrôle de vol de véhicule basé sur le GPS et le GSM en mettant en œuvre ces projets de système de sécurité du véhicule, un véhicule peut être protégé contre les vols. À l'avenir, ce système anti-vol pour les voitures sera amélioré pour fonctionner comme un système intégré de sécurité des données pour les systèmes de communication automobile. Cela garantirait que toutes les données échangées dans le véhicule et à l'extérieur du véhicule sont protégées.

Conclusion

Dans ce chapitre, et dans sa première partie, nous avons présenté les systèmes embarqués. Cela en décrivant leurs caractéristiques, leurs architectures ainsi que leur fonctionnement.

Dans la deuxième partie, nous avons exposé les systèmes temps réel, leurs définitions, leurs caractéristiques et leurs architectures. De plus, nous avons cité diffèrent exemples de systèmes embarqué dans l'automobile.

CHAPITRE II

DESCRIPTION DE PRINCIPAUX COMPOSANTS FORMANT LE SYSTEME

Introduction

Afin de réalisé un système de surveillance intelligent pour l'automobile, on a besoin d'utilisé différent composants d'électronique.

Dans ce chapitre, nous allons décrire les composants électroniques utilisés pour la réalisation du notre système.

II.1. Les composants utilisés

II.1.1. Capteur de vibration

Un capteur est un dispositif qui détecte une entrée physique de l'énergie, et le convertit en un autre type d'énergie comme un signal électrique mesurable.

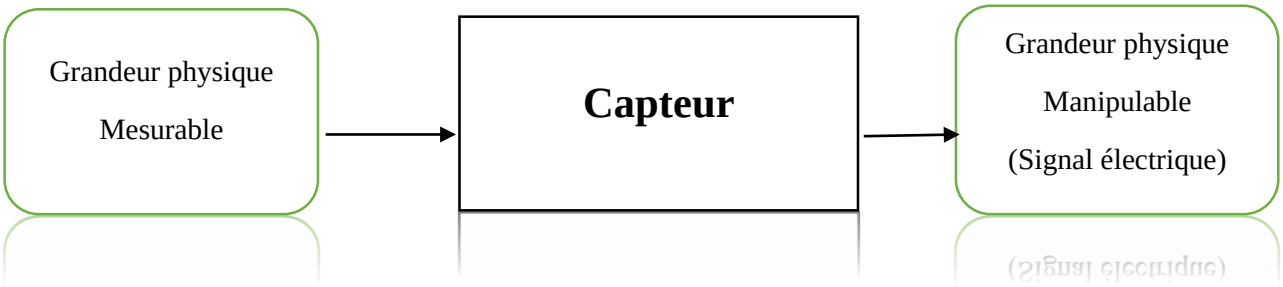


Figure II.1 : Schéma synoptique du capteur

Un capteur de vibration est un dispositif qui détecte un mouvement vibratoire, et le convertit en un signal électrique.

II.1.1.1. Rôle d'un capteur de vibration

Le but d'un transducteur de vibration est de convertir la vibration en un signal électrique. Il est généralement utilisé dans le contexte industriel pour évaluer la performance des machines industrielles, mais il peut aussi être utilisé à des fins différentes. Par exemple, un capteur de vibration peut être utilisé dans les alarmes de voitures et son rôle est de détecter les vibrations ou chocs sur le véhicule pour déclencher une alarme anti vol.

II.1.1.2. Capteur de vibration MSQ5 fabriqué par EBELCO

Nous avons utilisé le capteur MSQ5 EBELCO. C'est un capteur de vibration avancé à haute fréquence capable de détecter les effractions. C'est un mécanisme inestimable pour l'incorporation dans le système d'alarme de vol (figure II.2).

Figure II.2 : Capteur de vibration MSQ5

Sa taille est miniature (23(Largeur) x 23(hauteur) x 61(longueur) mm), il est rapide à installer avec une sensibilité réglable, ce capteur est adapté à tous types d'alarmes.

II.1.2. La carte Arduino

Arduino est une carte électronique en matériel libre basée autour d'un microcontrôleur Atmega du fabricant Atmel, qui peut être programmée pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses comme la domotique (le contrôle des appareils domestiques, éclairage, chauffage, etc.), l'électrotechnique industrielle et embarquée, le pilotage d'un robot, le modélisme (commander des appareils mobiles), etc [13] [20].

Les cartes Arduino possèdent un microcontrôleur facilement programmable ainsi que de nombreuses entrées-sorties. Plusieurs cartes Arduino existent et qui se différencient par la puissance du microcontrôleur ou par la taille et la consommation de la carte. Le choix du type de carte Arduino s'effectue en fonction des besoins de votre projet. La carte Arduino UNO est la carte la plus couramment utilisée qui constitue un bon choix pour les débutants.

L'ensemble des cartes Arduino se programment à l'aide d'un logiciel de programmation.

II.1.2.1. Les gammes de la carte Arduino

Actuellement, il existe plus de 20 versions de module Arduino, nous citons quelques-uns ainsi que leurs caractéristiques afin de montrer l'évolution de ces cartes [12] :

- 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 microcontrôleurATmega168.
- L'Arduino Nano, une petite carte programme à l'aide porte USB cette version utilisant un microcontrôleurATmega168 (ATmega328 pour une plus nouvelle version).
- Le 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'unATmega168.
- 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 d'USB/DC.
- L'Arduino Mega, en utilisant un microcontrôleur ATmega1280 pour I/O additionnel et mémoire.
- L'Arduino UNO, utilisations microcontrôleur ATmega328.
- L'Arduino Mega2560, utilisations un microcontrôleur ATmega2560, et possède toute la mémoire à 256 KBS. Elle incorpore également le nouvel ATmega8U2 (ATmega16U2 dans le jeu de puces d'USB de révision 3).
- L'Arduino Leonardo, avec un morceau ATmega3Ù4 qui élimine le besoin de raccordement d'USB et peut être employé comme clavier.
- L'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.

II.1.2.2. La carte Arduino UNO

L'UNO est le choix de prédilection d'un débutant, peu chère et facile à utiliser c'est celle que l'on conseille le plus souvent à ceux qui souhaitent se lancer dans l'aventure Arduino [16].

Elle mesure 69mm*54mm et pèse 25g, ça peut paraitre peu, mais lorsque vous serez sur un projet embarqué vous remarquerez qu'elle n'est pas si petite que ça. La Uno dispose de 14 entrées/sorties (I/O en anglais) sur lesquelles elle est capable de fournir jusqu'à 20mA (par pins digitaux) ainsi que 6 I/O analogiques. La différence entre ces 2 types de pins est que les pins digitaux fonctionnent grâce à des commandes HIGH et LOW équivalent à soit 5V soit 0V alors que les pins analogiques sont commandés grâce à une information codée sur 8bits (c'est à dire un octet) soit un nombre entier compris entre 0 et 255 où le 0 équivaut à 0V et 255 à 5V (mais les pins analogiques sont capables de lire plus que 256 valeurs, en tant qu'entrée un pin analogique fournira un nombre entre 0 et 1023 où le 0V correspondra à une tension de 0V et 1023 à une tension de 5V). Autres pins à connaitre, le 5V et le GND (la masse, le 0V) situés au-dessus des pins analogiques qui permettront d'alimenter vos composants, comme présente la figure III.3.

La UNO se connecte via un câble USB fourni à l'achat et dispose de 32Kb de mémoire Flash et 2KB de mémoire dynamique (utilisée par les variables) et peut être alimentée facilement par une pile 9V via le connecteur DC 2.5mm grâce à un simple câble d'adaptation. Le gros avantage du clone fourni par le shop face à l'arduino uno originale étant les connecteurs supplémentaires disponibles pour chacun des pins plus une série de connecteur très pratiques pour l'I2C, l'UART et de l'alimentation 3.3V.

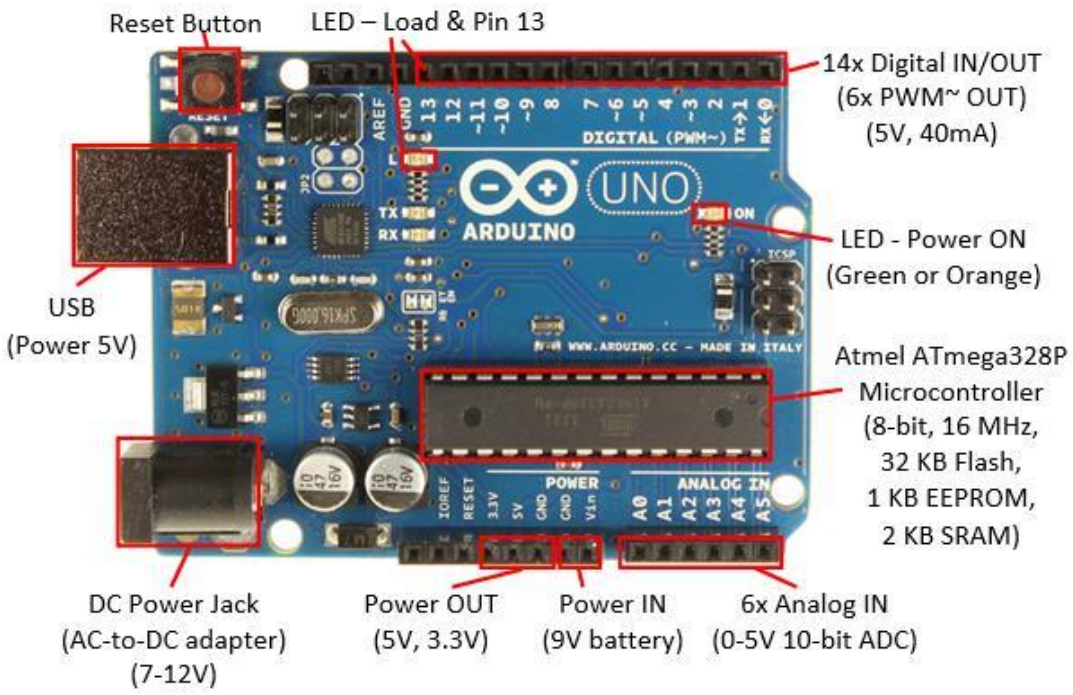


Figure III.3 : Description de la carte Arduino UNO

Le tableau ci-dessous regroupe l'ensemble de caractéristique de la carte Arduino UNO.

Tableau III.1 : Caractéristiques techniques d'Arduino UNO [17]

Microcontrôleur	ATmega328P
Tension de fonctionnement	5V
Tension d'entrée (recommandé)	7-12V
Tension d'entrée (limite)	6-20V
E / S numériques	14 (dont 6 fournissent une sortie PWM)
E / S numériques PWM	6
Pointes d'entrée analogiques	6
Courant CC par broche d'E / S	20 mA
Courant CC pour la broche de 3.3V	50 mA
Mémoire flash	32 Ko (ATmega328P) dont 0,5 Ko utilisé par bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Vitesse de l'horloge	16 MHz
LED_BUILTIN	13

Poids	25 g

- **PWM** : Modulation de largeur d'impulsion (Pulse Width Modulation).
- **LED_BUILTIN** : La plupart des cartes Arduino ont une broche connectée à une LED intégrée en série avec une résistance. La constante LED_BUILTIN est le numéro de la broche à laquelle est connectée la LED embarquée. La plupart des cartes ont cette LED connectée à la broche 13 numérique.
- **SRAM:** static random access memory. (Mémoire statique à accès aléatoire)
- **EEPROM :** Le microcontrôleur sur la carte Arduino et Genuino AVR possède une mémoire EEPROM: dont les valeurs sont conservées lorsque la carte est éteinte (comme un minuscule disque dur). Cette bibliothèque vous permet de lire et d'écrire ces octets.

II.1.3. Le module GSM /GPRS SIM800l

Le module SIMCOM SIM800L V2.0 GSM / GPRS est un module QUAD BAND GSM / GPRS compatible avec Arduino. Le module fonctionne pour ajouter à la fois des fonctionnalités GSM (appel vocal ou SMS) et des fonctionnalités GPRS. Les avantages de ces modules sont les niveaux série VCC et TTL qui ont une tension de 5V, de sorte que vous pouvez directement le connecter à Arduino ou à un autre système minimum avec 5V de niveau de tension. Il y a tellement de modules GPRS / GSM sur le marché qui ont besoin d'ajouter un régulateur 5V et un circuit convertisseur de niveau, alors que le module SIM800L V.2 GSM / GPRS a déjà un circuit régulateur intégré et un convertisseur de niveau TTL sur la carte (figure III.4) [18].

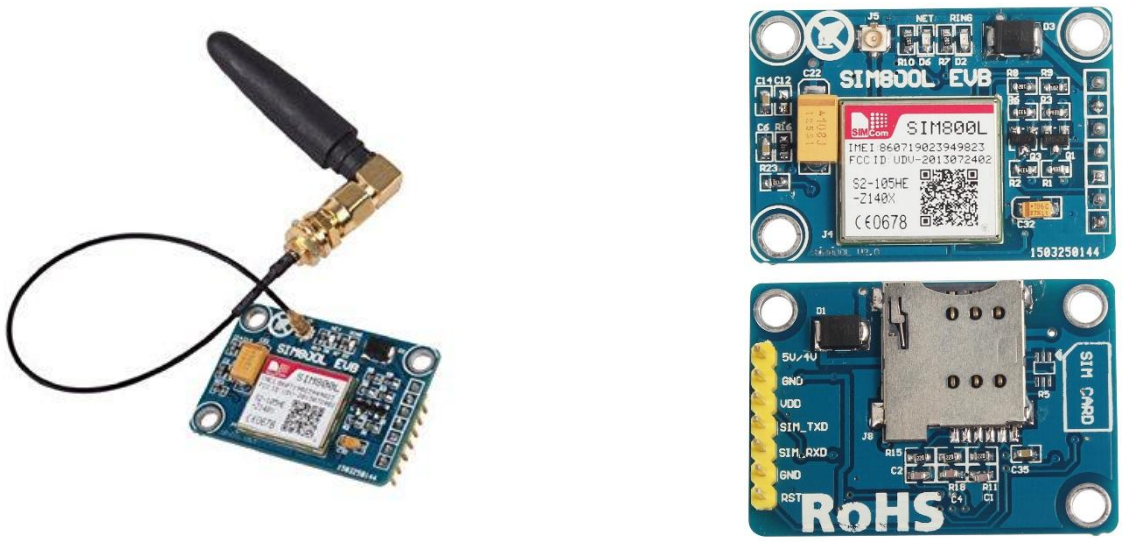


Figure III.4 : Module Sim800l EVB v2

II.1.3.1. Spécifications de SIM800L V2.0 GSM / GPRS Module

SIM800L V2.0 GSM / GPRS Module a beaucoup de spécifications, nous citons quelques-uns ci-dessous [18] :

- Interface série TTL compatible avec les microcontrôleurs 3.3V et 5V, compatible avec arduino.
- Ce module SIM800L possède un ensemble d'interface série TTL, un ensemble d'interface d'alimentation.
- En outre, il existe un ensemble d'interface d'antenne sur ce module.
- Support réseau : Quad-Band 850/900/1800/1900 MHz, il peut transmettre des informations vocales, SMS et données à faible puissance.
- Interface VDD TTL UART L'interface série TTL UART, vous pouvez connecter le MCU comme 51MCU ou ARM ou MSP430 directement. La broche de VDD est utilisée pour correspondre à la tension de la TTL.
- Tension de travail : 3.7V à 5V- Taille : 40mm x 28mm x 3mm.
- GPRS multi-slot classe 12/10.
- Station mobile GPRS classe B.
- Conforme à la phase GSM 2/2 +.
- Classe 4 (2 W @ 850/900 MHz).
- Classe 1 (1 W @ 1800 / 1900MHz).

II.1.4.Module GPS NEO6MV2

La série de modules NEO-6 est une famille de récepteurs GPS autonomes dotés du moteur de positionnement haute performance u-blox 6. Ces récepteurs flexibles et économiques offrent de nombreuses options de connectivité dans un boîtier miniature de 16 x 12,2 x 2,4 mm. Grâce à leur architecture compacte et à leurs options de puissance et de mémoire, les modules NEO-6 sont parfaits pour les appareils mobiles fonctionnant sur batterie avec des contraintes de coût et d'espace très strictes. Le moteur de positionnement u-blox 6 dispose d'un Time-To-First-Fix0 moins d'une seconde Le moteur d'acquisition dédié, avec 2 millions de corrélateurs, est capable d'effectuer des recherches massives de temps et de fréquence parallèles, ce qui lui permet de trouver des satellites instantanément. La conception et la technologie innovantes suppriment les sources de brouillage et atténuent les effets des trajets multiples, donnant aux récepteurs GPS NEO-6 d'excellentes performances de navigation, même dans les environnements les plus difficiles (figure III.5) [19].

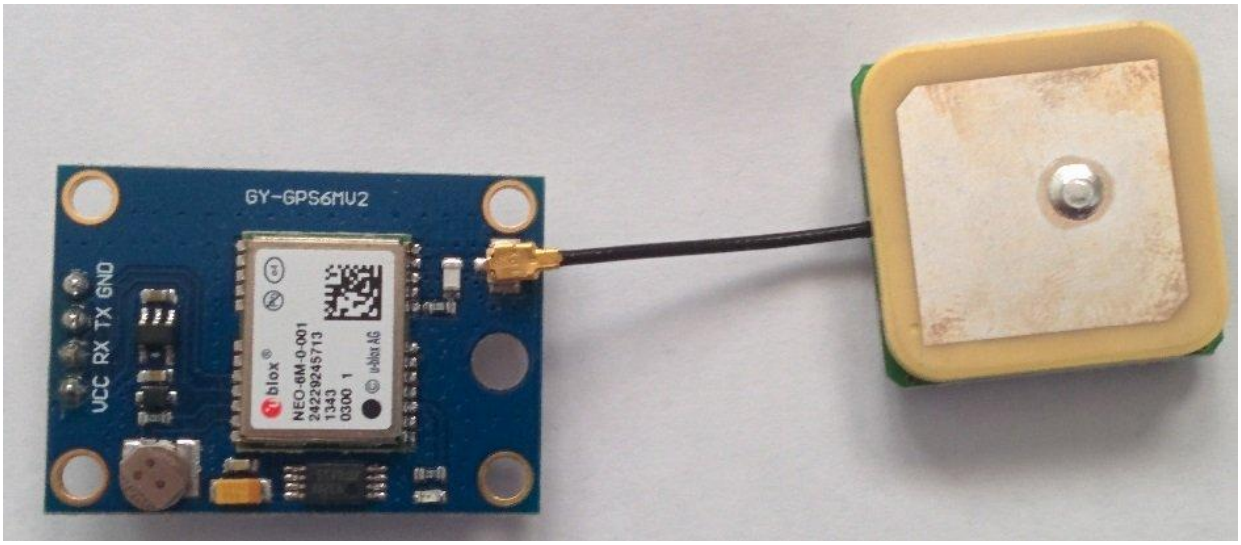


Figure III.5 Module GPS NEO-6m-V2

II.1.4.1. Caractéristiques du module GPS Neo6Mv2

L'ensemble de caractéristique du module GPS Neo6Mv2 est présenté comme :

- De type GY-NEO6MV2.
- Alimentation 3V-5V.
- Avec Flight Control EEPROM MWC.
- Antenne large céramique, fort signal, 25 x 25 mm.
- Paramètres de configuration sauvegardés en EEPROM.
- Avec batterie de backup de données.
- LED témoin de signal.
- Module miniaturisé 25 x 35 mm.
- Trou de fixation de 3mm.
- Baud rate par défaut 9600.
- Compatible avec de nombreux modules de contrôle de vol.

II.2. Outils de simulation et langage programmation

II.2.1. Arduino

Le logiciel de programmation des modules Arduino est une application multiplateformes (compatible Windows, Linux et Mac), servant d'éditeur de code et de compilateur, et qui peut transférer le Firmware (le programme) au travers de la liaison série (RS232, Bluetooth ou USB selon le module) [12].

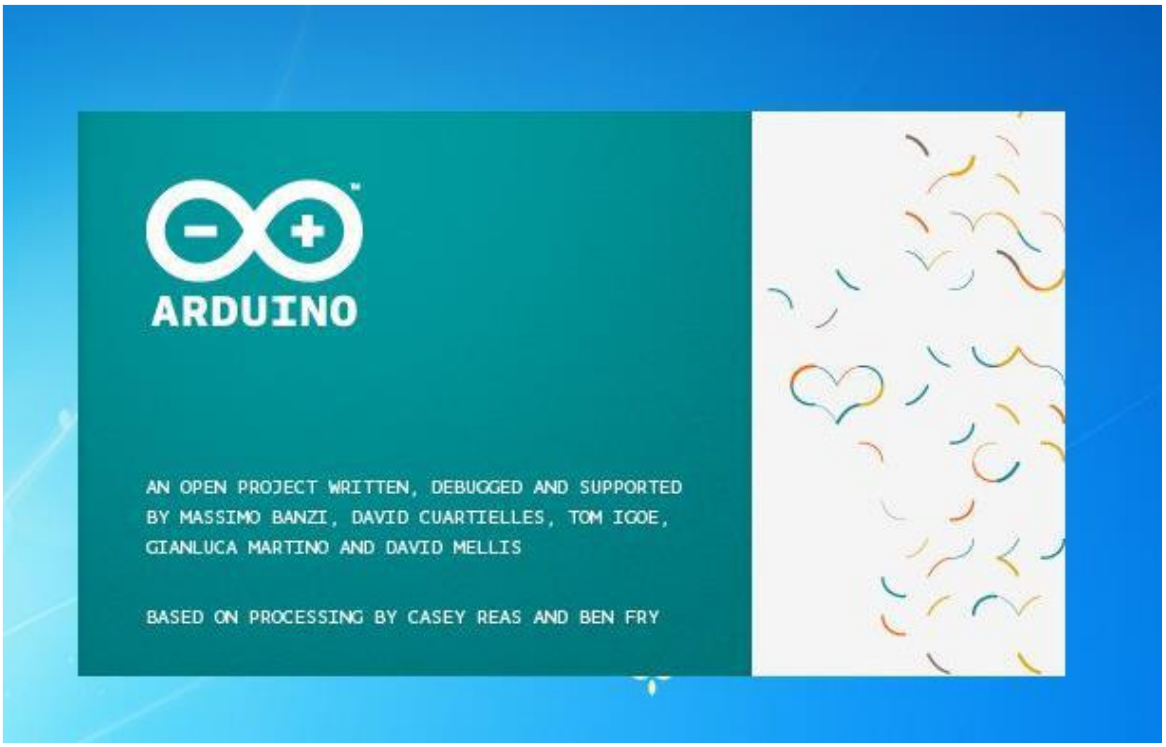


Figure II.6 L'écran de démarrage d'Arduino

L'interface de l'IDE (Integrated Development Environment) Arduino est plutôt simple (Fig.II.7), il offre une interface minimale et épurée pour développer un programme sur les cartes Arduino. Il est doté d'un éditeur de code avec coloration syntaxique et d'une barre d'outils rapide .Ce sont les deux éléments les plus importants de l'interface, c'est ceux que l'on utilise le plus souvent.

On retrouve aussi une barre de menus (Fig.II.7) plus classique qui est utilisé pour accéder aux fonctions avancées de l'IDE. Enfin, une console affichant les résultats de la compilation du code source, des opérations sur la carte, etc.

Figure II.7 Interface d'IDE Arduino

L'IDE Arduino permet :

- **d'éditer un programme** : des croquis (sketch en Anglais),
- **de compiler ce programme** dans le langage « machine » de l'Arduino,
- **de téléverser le programme** dans la mémoire de l'Arduino,
- **de communiquer** avec la carte Arduino grâce au terminal.

II.2.2. La suite du logiciel Proteus

Proteus est une suite logicielle destinée à l'électronique. Développé par la société Labcenter Electronics, les logiciels incluent dans Proteus permettent la CAO dans le domaine électronique. Deux logiciels principaux composent cette suite logicielle : ISIS, ARES, PROSPICE et VSM [20].

Cette suite logicielle est très connue dans le domaine de l'électronique. De nombreuses entreprises et organismes de formation (incluant lycée et université) utilisent cette suite logicielle. Outre la popularité de l'outil, Proteus possède d'autres avantages

- Pack contenant des logiciels facile et rapide à comprendre et utiliser
- Le support technique est performant
- L'outil de création de prototype virtuel permet de réduire les coûts matériel et logiciel lors de la conception d'un projet

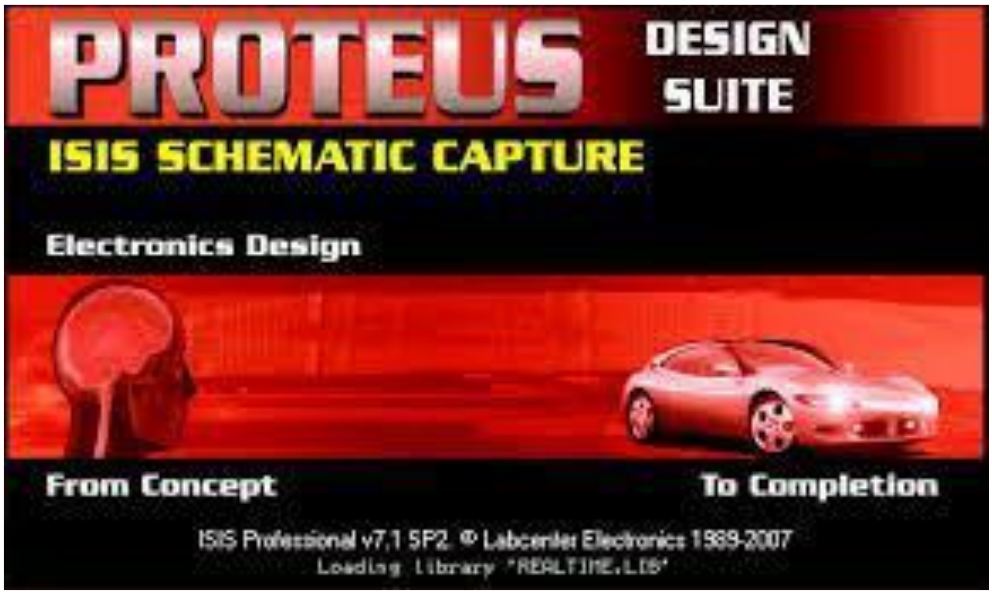


Figure II.8 la suite logicielle Proteus

Deux logiciels principaux composent cette suite logicielle : ISIS et ARES.

II.2.2.1. Présentation de l'outil ISIS

Le logiciel ISIS de Proteus est principalement connu pour éditer des schémas électriques. Par ailleurs, le logiciel permet également de simuler ces schémas ce qui permet de déceler certaines erreurs dès l'étape de conception. Indirectement, les circuits électriques conçus grâce à ce logiciel peuvent être utilisé dans des documentations car le logiciel permet de contrôler la majorité de l'aspect graphique des circuits [20].

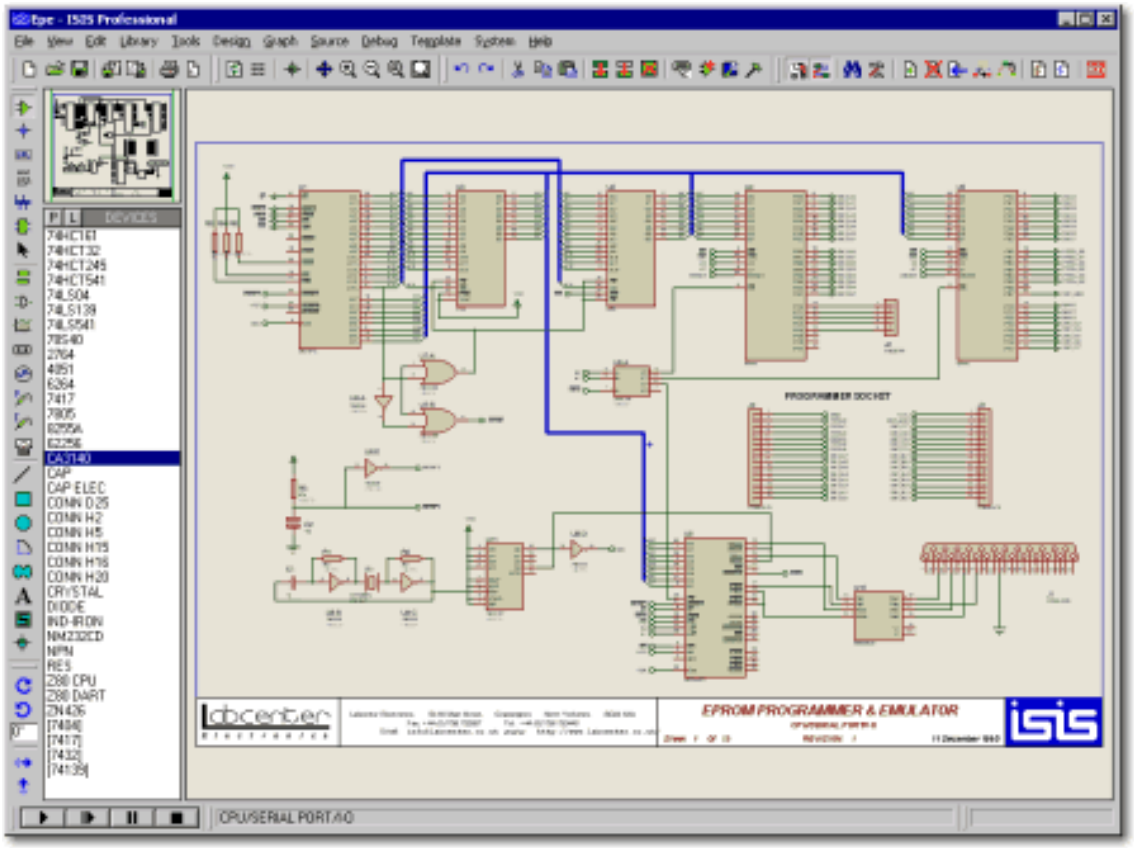


Figure II.9 Le logiciel ISIS de Preuteus

II.2.2.2. Présentation d'ARES

Le logiciel ARES est un outil d'édition et de routage qui complètement parfaitement ISIS. Un schéma électrique réalisé sur ISIS peut alors être importé facilement sur ARES pour réaliser le PCB de la carte électronique. Bien que l'édition d'un circuit imprimé soit plus efficiente lorsqu'elle est réalisée manuellement, ce logiciel permet de placer automatiquement les composants et de réaliser le routage automatiquement.

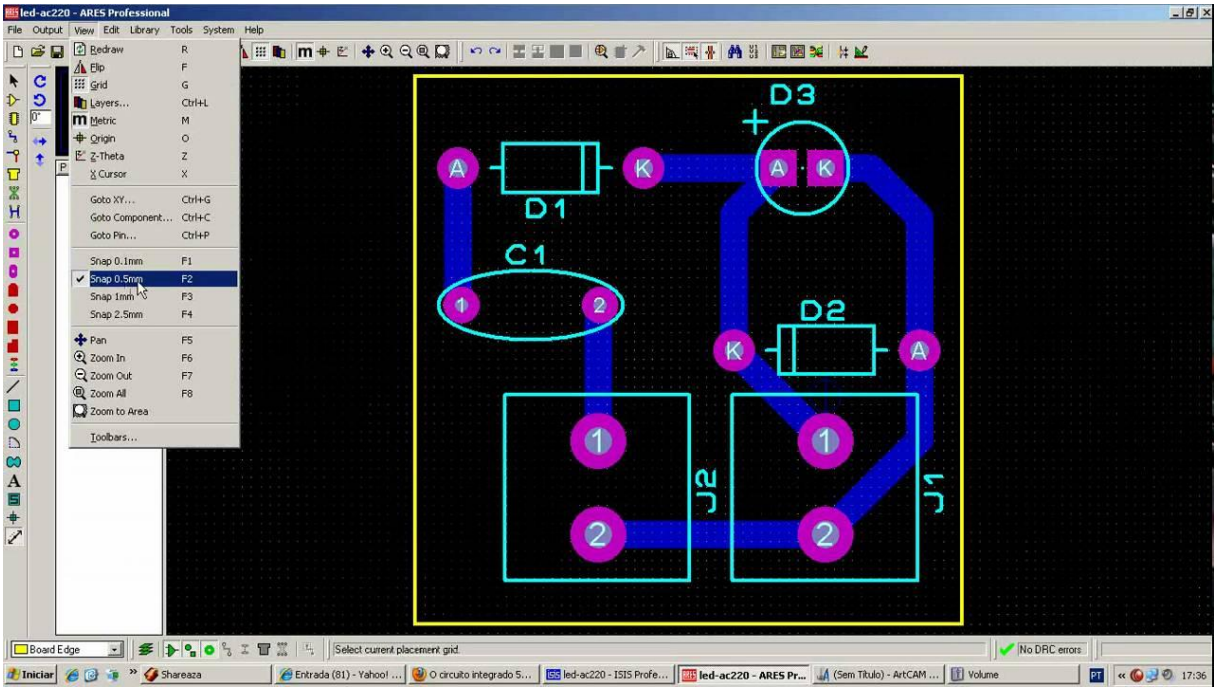


Figure II.10 Le logiciel ARES de Proteus

Conclusion

Dans ce chapitre, nous avons projeté la lumière sur les principaux composants électronique formant le système de surveillance pour l'automobile, en donnant ainsi leurs principes de fonctionnement et leurs architectures. Puis nous avons présenté les outils utilisés pour la conception du système ainsi que le langage utilisé pour la programmation de la carte.

CHAPITRE III
REALISATION ET TESTS

Introduction

Dans ce chapitre, nous nous attacherons à la programmation, réalisation, et tests du système de surveillance d'automobile. Donc, nous allons présenter l'organigramme du programme nécessaire pour le bon fonctionnement de notre système, ainsi que son mode de fonctionnement.

III.1. Description fonctionnel du système

Sur la figure III.1, nous présentons le schéma fonctionnel du système élaboré.

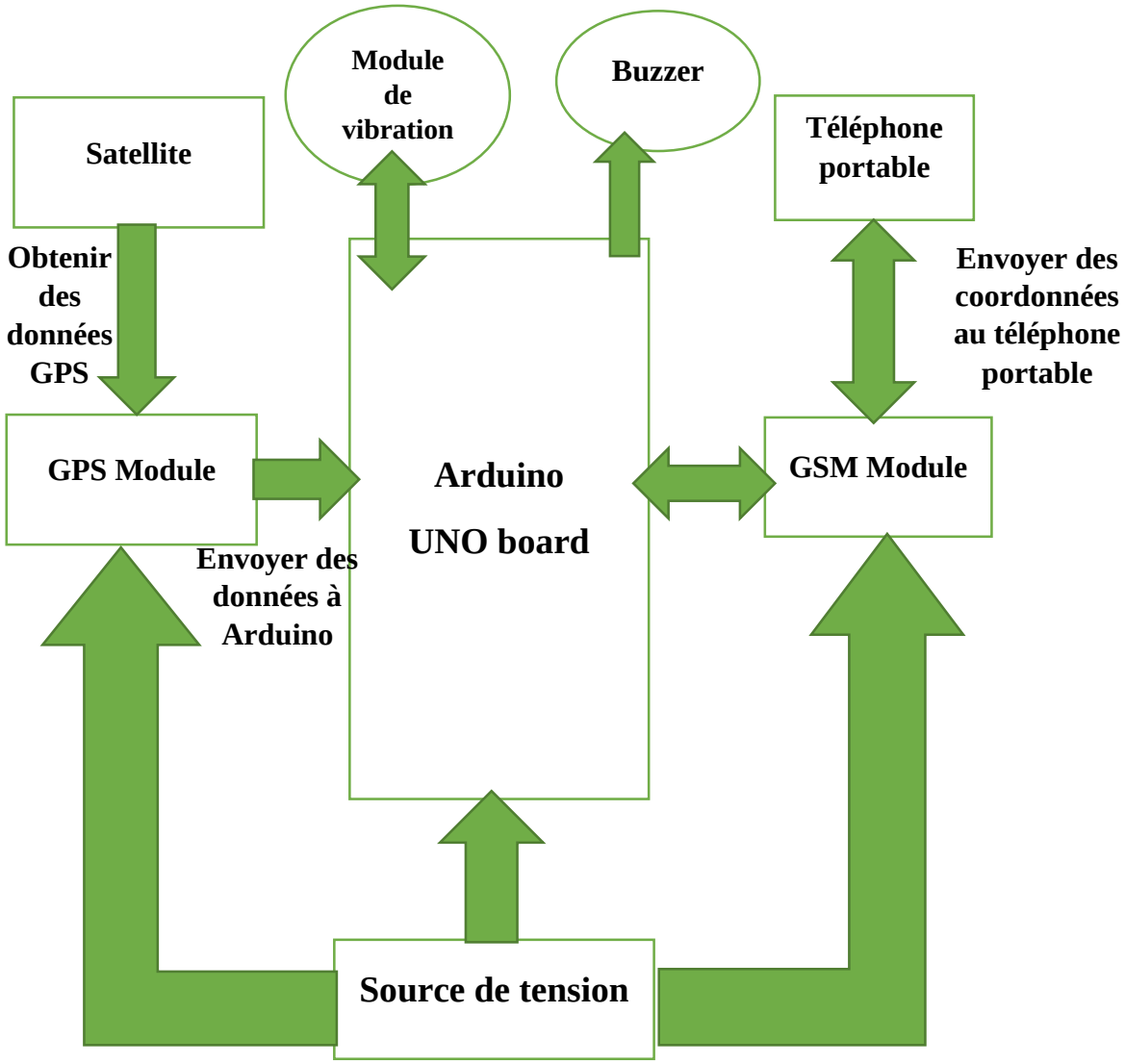


Figure III.1 Schéma fonctionnel du système de surveillance basé sur Arduino

Après la mise sous tension des composants électroniques du système, ce dernier sera en attente du SMS d'activation qui sera envoyé par le propriétaire ou l'utilisateur du système.

De manière globale, notre système fonctionne selon trois modes : le mode désactivé, le mode activé et le mode de localisation.

Dans le mode désactivé, le système de surveillance est à l'état OFF. Ce mode représente l'état initial ou l'état normal, dont lequel le système est en attente du SMS d'activation.

Dans le mode activé, le système est mis à l'état ON par l'utilisateur et via un SMS d'activation. Lorsque le contenu du SMS est bon, le système de surveillance est en cours de fonctionnement.

Dans le mode de localisation, le système définit les coordonnées GPS puis il les envois au portable de l'utilisateur à temps réel, permettant ainsi la localisation de l'automobile sur Map.

III.2. Schéma du circuit électrique

Dans cette partie, nous présentons le circuit électrique du montage réalisé ainsi que le fonctionnement du circuit et le rôle de chaque composant dans le circuit.

La figure III.3, montre le montage du circuit réalisé au cours de ce projet. Nous présentons ainsi la connexion de différents composants avec la carte Arduino. Cette dernière traite les données obtenues du capteur de vibration, du module GPS et du module SIM, et contrôle ainsi le Buzzer et surtout le module SIM.

Dans le système élaboré, le module GPS NEO-6m-V2 définit les cordonnées de localisation, puis il va les transmettre à l'utilisateur via le module SIM Sim800l EVB v2. Cela est effectué lorsque le capteur détecte un niveau de vibration élevé. Ainsi, l'utilisateur reçoit un SMS d'alerte et les cordonnés GPS qui permettent la localisation de l'automobile sur Map.

De plus, un signal sonore est généré par le Buzzer. L'activation du système est réalisée par l'utilisateur via un SMS d'activation ‘’ON’’.

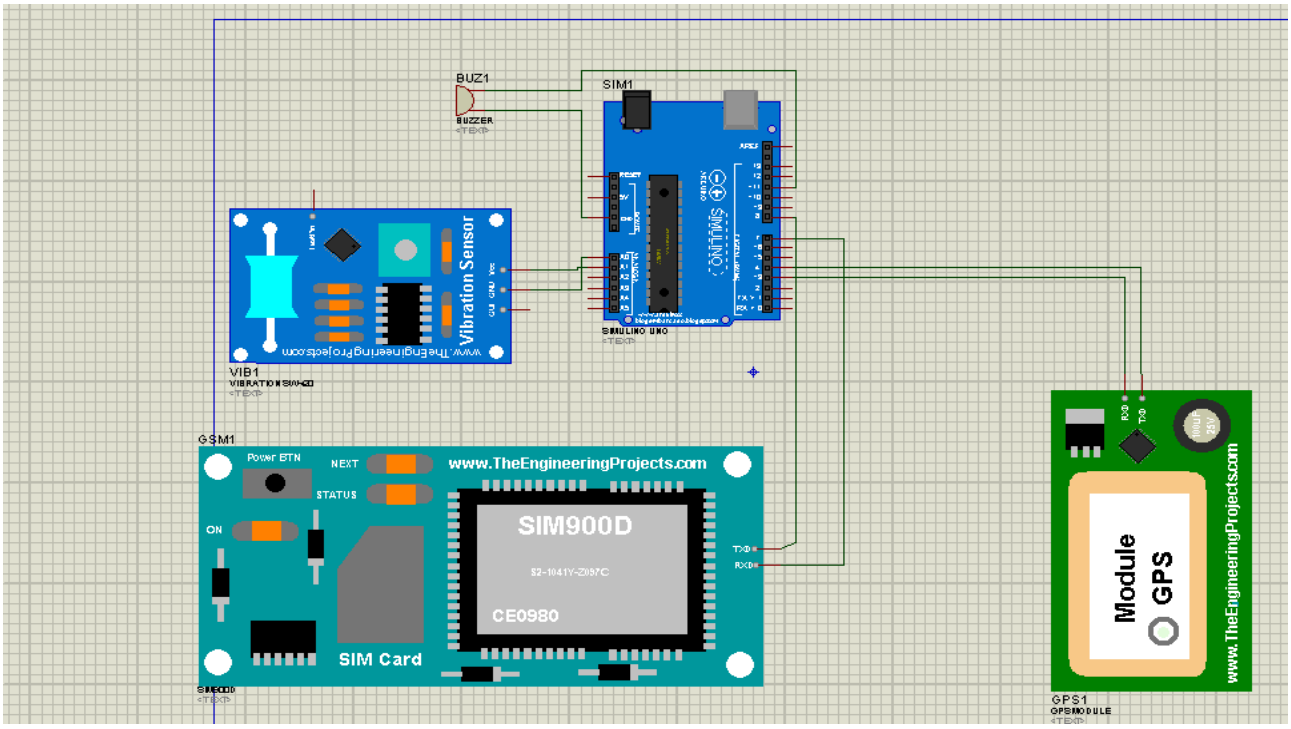


Figure III.3 Schéma du montage électrique du système

Les pins TX et RX du module GSM sont connectés â les pins 8 et 7 d'Arduino UNO.

La pine 5V du module GSM est reliée avec le pin 5V d'Arduino. Aussi, les pins du module GPS, TX et RX sont connectées avec les pins 4, 3 d'Arduino, et la pine Vcc avec le pin 3.3V d'Arduino. Le Buzzer est relié avec le pin 11 du module PWM d' Arduino. En fin, le détecteur de vibration est relié avec les pins analogiques A0 et A1 d'Arduino. Cela comme présente la figure ci-dessus.

III.3. Réalisation et tests

La figure III.4 montre une vue du teste élaboré du circuit électrique de command.

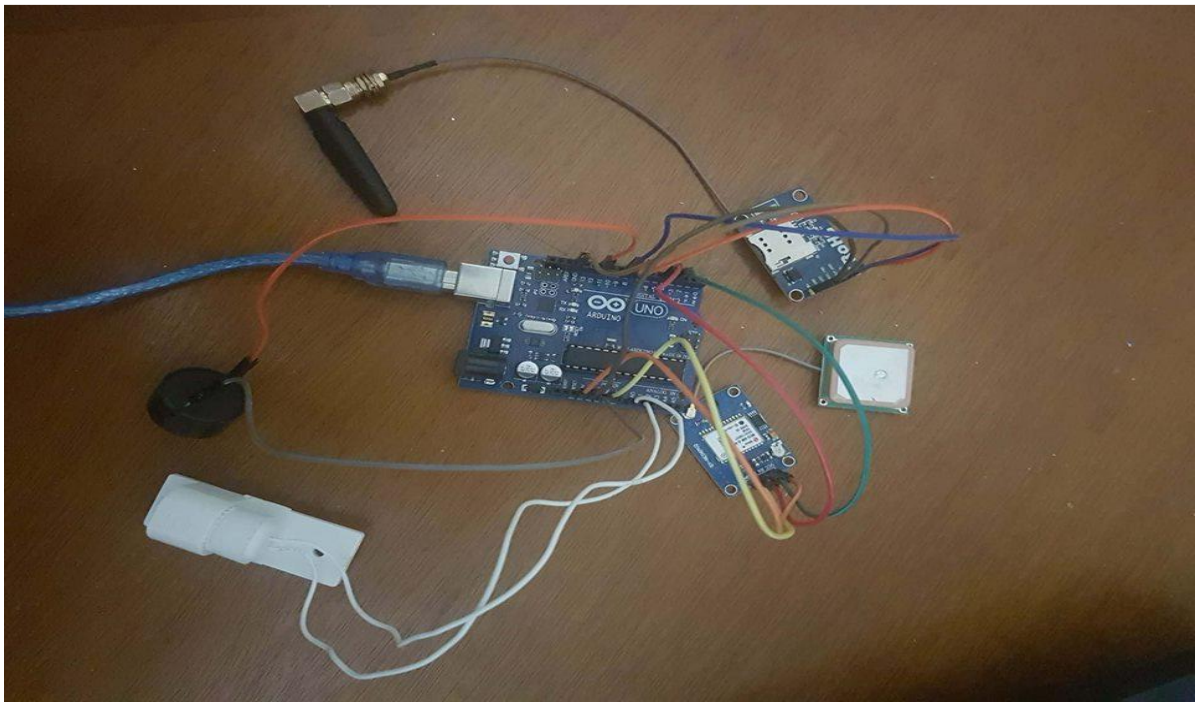


Figure III.4 vue du dessus du test élaboré sur le circuit électrique de command

Tester le code sur le circuit a été couronnée de succès, donc nous pouvons souder le circuit pour une meilleure protection.

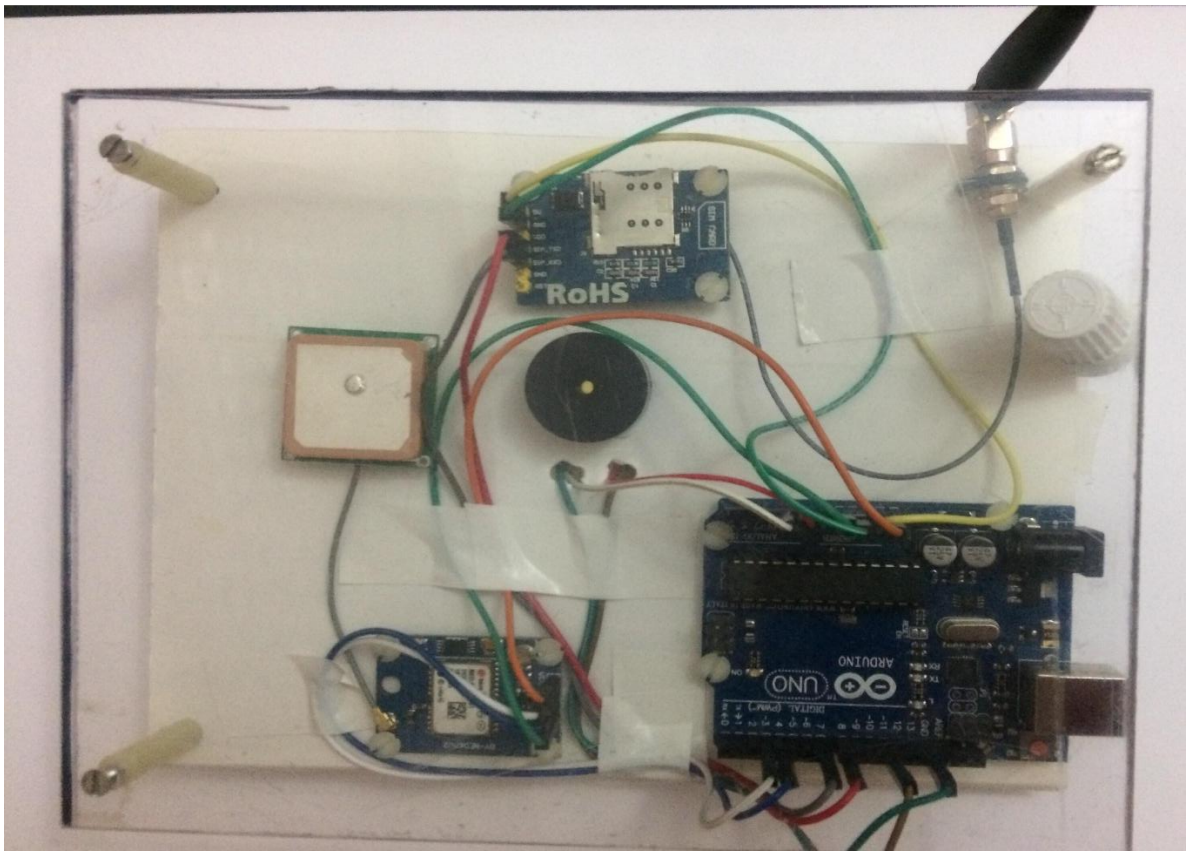
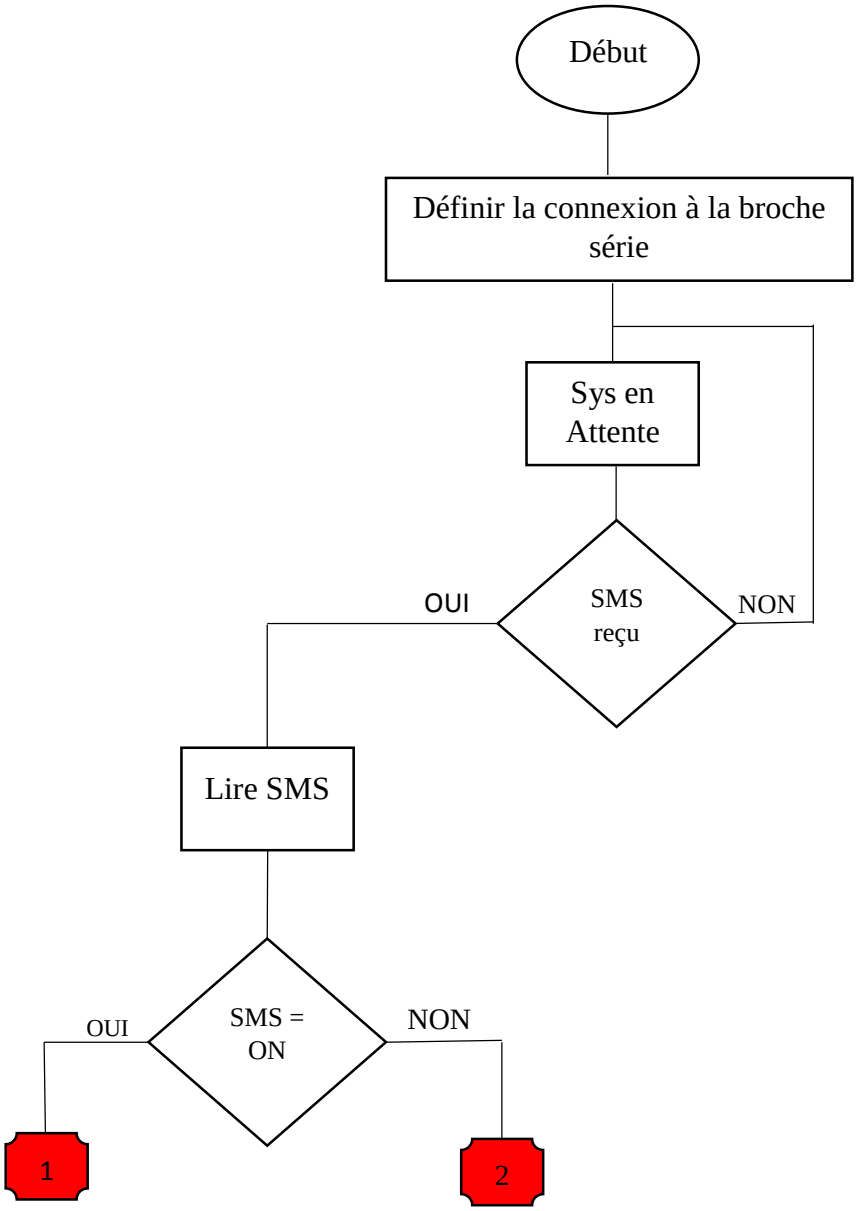


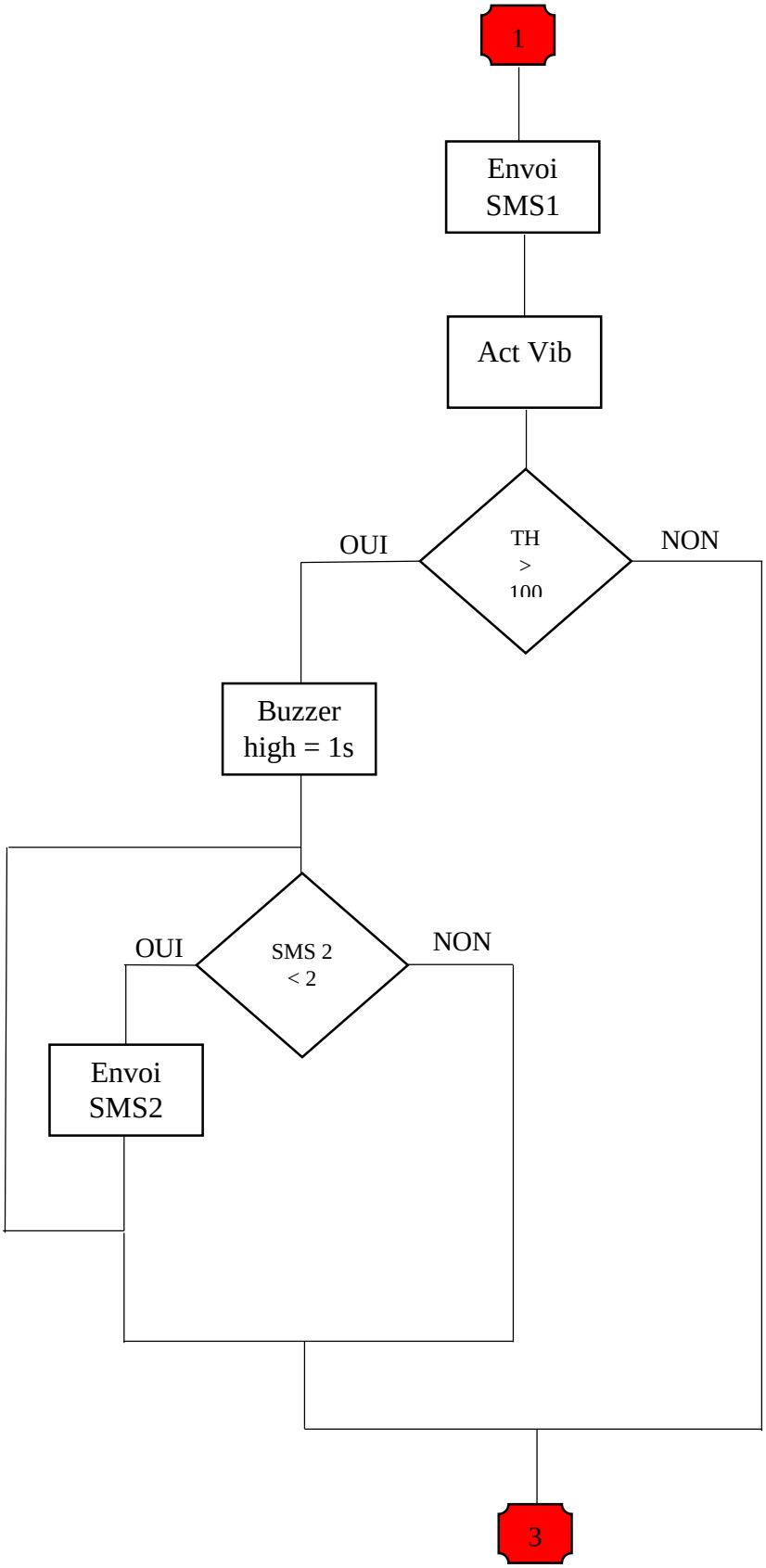
Figure III.5 vue du dessus du boitier qui contient le système réalisé.

Dans la figure III.5 on voit l'assemblage du prototype qui a été réalisé avec succès et prêt pour la mise en fonction.

III.4. Organigramme

Le fonctionnement de notre système est représenté graphiquement par l'organigramme présenté sur la figure suivante.




1
Envoi SMS1
Act Vib
TH > 100
OUI
NON
Buzzer high = 1s
SMS 2 < 2
OUI
NON
Envoi SMS2
3

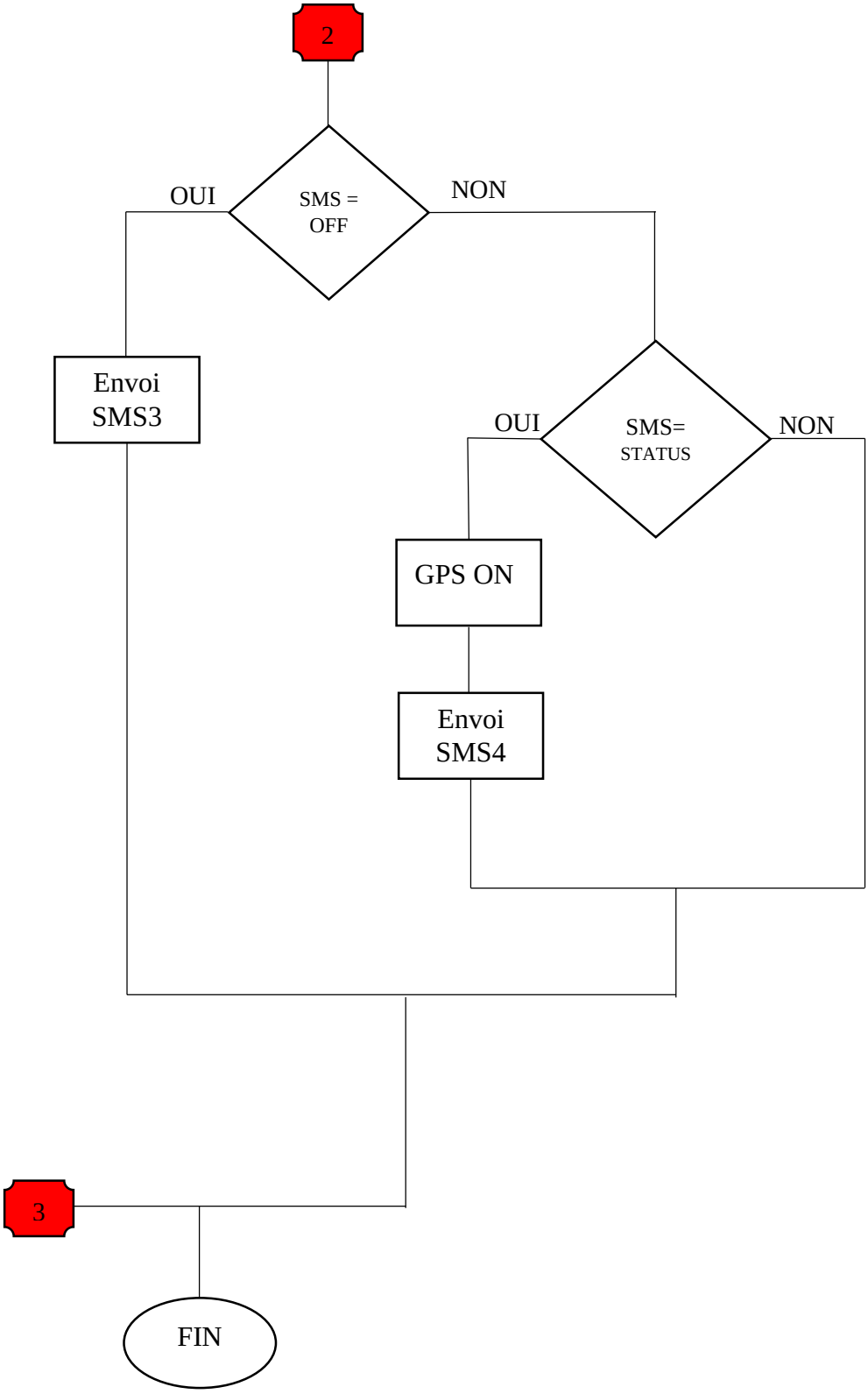


Figure III.2 Organigramme général du système

Guide d'utilisation

- Sys en Attente : Un mode ou le système est en Attente d'un SMS.
- SMS : C'est un SMS d'activation du système.
- SMS1 : c'est un SMS qui informe l'utilisateur que le système d'alarme est activé.
- SMS2 : c'est un SMS qui informe que le véhicule risque d'être volé.
- SMS3 : c'est un SMS qui informe l'utilisateur que le système d'alarme est désactivé.
- Act vib : Activation du détecteur de vibration.
- TH : est le seuil de vibration qui d'éclanche l'alarme.

Au début, et après la mise en marche de notre système par l'utilisateur, le système entre dans un mode d'attente, attente d'un SMS reçu.

Par la suite, et après que notre système reçoit un SMS, il le lit et si le contenue de SMS correspond à une des trois cas : ON, OFF, STATUS, le système commence la prochaine étape, si le SMS ne correspond pas à un des trois cas le système retourne à la première phase ' mode d'attente'.

Si le contenue de SMS reçu est ON, le système envoie un SMS à l'utilisateur pour lui informer que le système d'alarme est en marche. Apres que le système d'alarme soit activé le capteur de vibration sera activé et commence a détecté toute vibration dans la voiture, si il détecte une vibration importante le système envoie un SMS a l'utilisateur pour lui informer qu'il a détecté une vibration importante et de là un risque de vol.

Si le contenue de SMS reçu est OFF, le système envoie un SMS a l'utilisateur pour lui informer que le système d'alarme a été arrêté.

Si le contenue de SMS reçu est STATUS, le système met en marche le module GPS, et détecte la géolocalisation de véhicule, après il envoie un SMS a l'utilisateur avec les cordonnées GPS (longitude, latitude).

Conclusion

Dans ce chapitre nous avons présenté l'organigramme global du programme élaboré sous Arduino. En effet, l'organigramme réalisé décrit les principales étapes dans ce système d'anti vol de véhicule.

Puis, et dans un deuxième temps, nous sommes passés à réalisation des circuits électriques de ce système.

En dernier lieu, nous avons réalisé et testé le système. Les résultats obtenus confirment le bon fonctionnement du système, permettant ainsi la surveillance de la voiture.

D'après ces résultats, nous pouvons dire que notre système est simple à réaliser et non coûteux

CONCLUSION GENERALE

Conclusion général

Le vol de véhicules est devenu un crime de premier degré contre les biens de la nation, ce qui a poussé de nombreux propriétaires de voitures à s'appuyer sur des systèmes d'alarmes de voiture pour protéger leurs véhicules contre cet acte. Malheureusement, même les systèmes d'alarme ne suffisent pas à se protéger contre le vol des automobiles.

Le travail présenté dans ce mémoire, s'inscrit dans le cadre de notre projet de fin d'études, dont lequel nous avons réalisé un système à temps réel permettant la surveillance intelligente des automobiles.

Au début du mémoire, nous avons commencé par la description des systèmes embarquées et des systèmes temps réels.

Ensuite, nous avons présenté les éléments constituant le système, en donnant la définition, l'architecture et le principe de fonctionnement de chaque composant considéré. Aussi, nous avons cité les outils utilisés pour la conception du circuit électrique.

A la fin, nous avons présenté en détails le travail réalisé au cours de ce projet.

Le système réalisé permet, d'une part, la détection et la prévention contre un acte de vol et surtout d'informer le propriétaire en temps réel. Et d'autre part, il permet de repérer en cordonnée GPRS la position de l'automobile sur « Map ». Le système réalisé est basé sur l'utilisation d'une carte Arduino, d'un module GSM/GPRS et d'un capteur de vibration.

De même nous avons élaboré un programme sous Arduino pour permettre le contrôle des composants du système, ainsi que bon fonctionnement de ce dernier.

Le système réalisé se présente sous forme d'une carte électronique dans un boitier de poids faible, ce qui simplifier l'utilisation et l'installation cacher de ce dernier dans le porte bagage de l'automobile.

En mettant en œuvre ce projet, nous avons appris l'utilisation de technologie GPS et GSM, et même d'autres composants électroniques, comme les capteurs.

Le développement du système considéré a été une expérience intéressante et réussie du point de vue électronique. En effet, cela a permis non seulement de parfaire de nombreuses compétences mais aussi de découvrir d'autres technologies passionnantes et d'apprendre à les utiliser pour la réalisation et la mise en œuvre de tel projet.

Conclusion générale

En perspective, il est intéressant de rajouter un système d'anti vol permettant le blocage de plusieurs circuits et composants de l'automobile à temps réel, cela dans le cas d'un vol.

Références bibliographiques

[1] Histoire de système embarquée

https://fr.wikipedia.org/wiki/Syst%C3%A8me_embarqu%C3%A9#Histoire (consulté le 18/06/2018).

[2] Définition de système embarqué

https://internetofthingsagenda.techtarget.com/definition/embedded-system (consulté le 18/06/2018).

[3] Programmation de systèmes embarqués : de hardware au software Cour

https://www.ukonline.be/cours/embeddedsystems/programming/ (consulté le 18/06/2018).

[4] Cours Systèmes embarqués

http://pagesperso.lina.univ-nantes.fr/~prie-y/archives/VEILLE-2009-2012/2012/wsembarq/3.html#_Toc324843109 (consulté le 10/06/2018).

[5] Typical architecture embedded system course

https://ebrary.net/22041/computer_science/typical_architecture_embedded_system (consulté le 05/06/2018).

[6] Embedded software Course

https://internetofthingsagenda.techtarget.com/definition/embedded-software (consulté le 10/06/2018).

[7] Quels langages utilisez-vous pour le développement de systèmes embarqués ? Le 29 juin 2017, par Michael Guilloux, Chroniqueur Actualités

https://www.developpez.com/actu/146789/Quels-langages-utilisez-vous-pour-le-developpement-de-systemes-embarques/ (consulté le 28/05/2018).

[8] wikipedia système embarqué

https://fr.wikipedia.org/wiki/Syst%C3%A8me_embarqu%C3%A9 (consulté le 28/05/2018).

[9] Class 301: Overview of Embedded and Real-Time Systems

http://profile.iiita.ac.in/bibhas.ghoshal/lecture_slides_embedded/embeddedsystemcomponents notes1.pdf (consulté le 28/05/2018).

[10] SYSTÈMES TEMPS RÉEL EMBARQUÉS Spécification, conception, implémentation et validation temporelle 2e édition

http://excerpts.numilog.com/books/9782100713318.pdf (consulté le 05/05/2018).

[11] Embedded Systems in Automobiles- A Boon to Automobile Industry

https://www.mepits.com/tutorial/299/embedded-system/embedded-systems-in-automobiles-a-boon-to-automobile-industry (consulté le 10/05/2018).

[12] Baya Hamdani, ‘’Etude et conception d’un système automatique pour le contrôle et la régulation de la pression‘’, Mémoire de Master, université M’HAMED BOUGARA BOUMERDES, Algérie, 2017.

[13] A. Krama, A. Gougui ‘’Etude et réalisation d’une carte de contrôle par Arduino via le système Androïde’’. Université Kasdi Merbah Ouargla, Algérie, 2017.

[14] Arduino Products

https://www.arduino.cc/en/Main/Products (consulté le 19/05/2018).

[15] Choisir la carte Arduino adaptée à ses besoins

http://www.robot-maker.com/ouvrages/tuto-arduino/choisir-carte-arduino-adaptee/ (consulté le 12/05/2018).

[16] ARDUINO UNO REV3

https://store.arduino.cc/usa/arduino-uno-rev3 (consulté le 16/06/2018).

[17] SMS CONTROLLER BY USING SIM800L V2

http://www.instructables.com/id/SMS-Controller-by-Using-SIM800L-V2/ (consulté le 16/06/2018).

[18] GPS Module NEO6MV2

https://wiki.eprolabs.com/index.php?title=GPS_Module_NEO6MV2 (consulté le 15/06/2018).

[19] Proteus (ISIS et ARES)

http://www.elektronique.fr/logiciels/proteus.php (consulté le 15/06/2018).

[20] S. Landrault et H. Weisslinger, ‘’Arduino : Premiers pas en informatique embarquée’’, Le blog d'Eskimon, 2014.

[21] Hui ZHANG ‘Gestion de l’énergie renouvelable et ordonnancement temps réel dans les systèmes embarqués ’ Thèse de doctorat, Université de Nantes faculté des sciences et des techniques, France, 2012.

La carte Arduino UNO

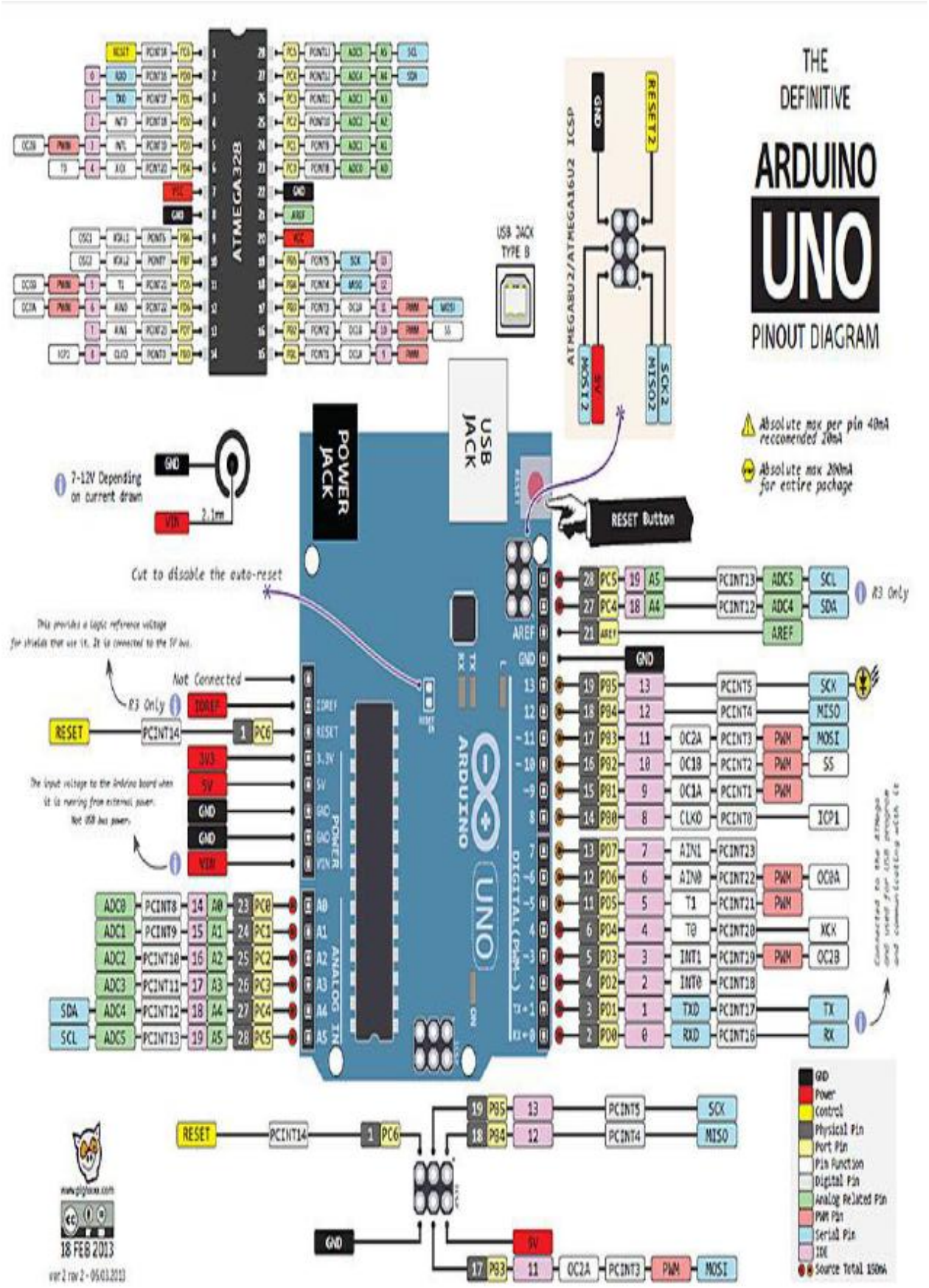
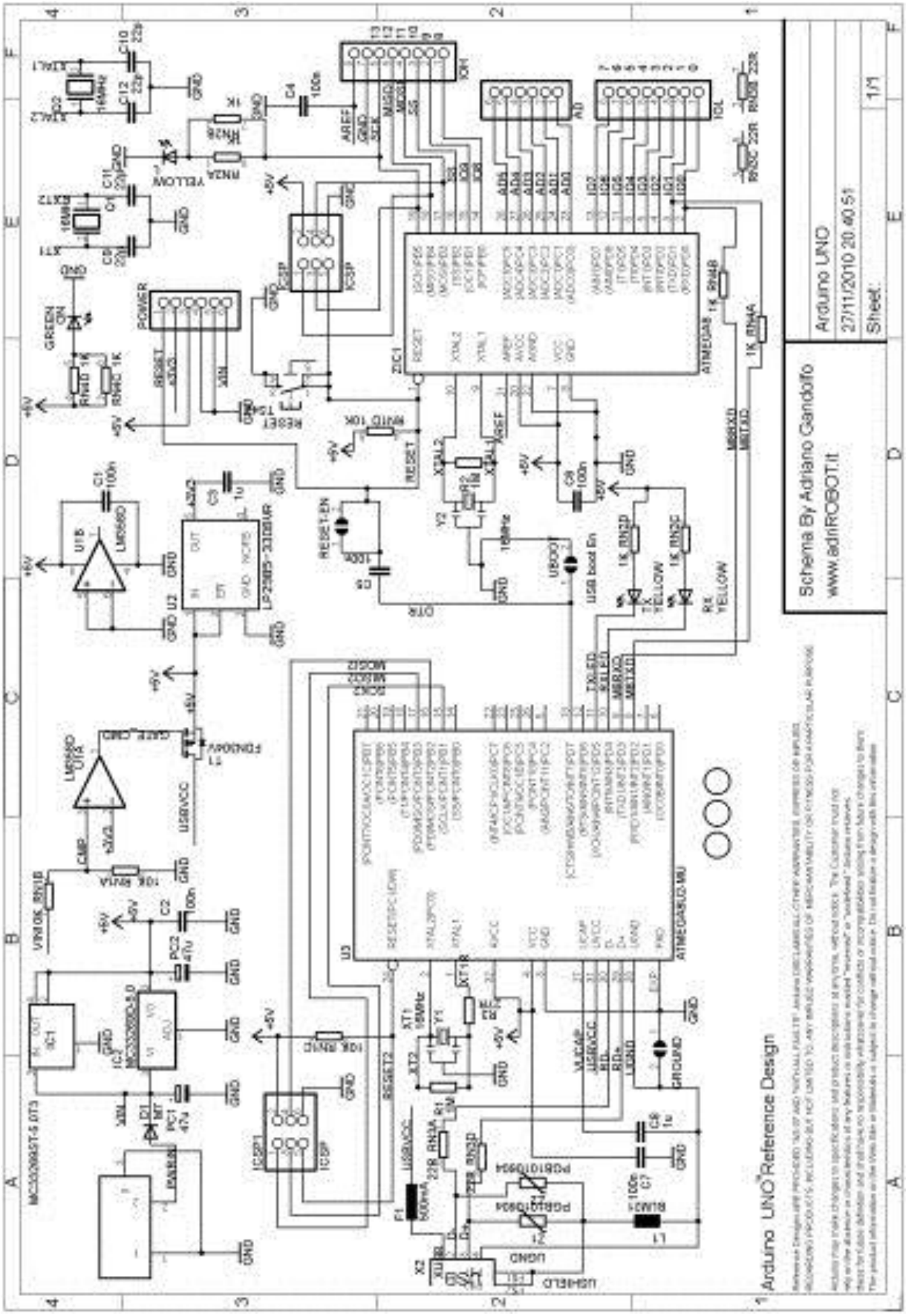


Schéma électrique de la carte Arduino UNO



Module GPS NEO6Mv2

SYNACORP TRADING & SERVICES
No.9, 1st Floor, Lrg 1/552, Bandar Tasek Mutiara, 14120 Simpang Ampat, S.Prai (S), Penang
Tel : +604.504.1617 Hunting Line : 012.4033.474 Fax : +604.502.1726
(Website) http://www.synacorp.my (Email) sales@synacorp.com.my

Arduino GY-NEO6MV2 GPS Module c/w Antenna & Flight Control EEPROM

GY-NEO6MV2 board features the u-blox NEO-6M GPS module with antenna and built-in EEPROM. This is compatible with various flight controller boards designed to work with a GPS module.

Technical Specifications:

- Power Supply Range: 3 V to 5 V
- Model: GY-GPS6MV2
- Ceramic antenna
- EEPROM for saving the configuration data when powered off
- Backup battery
- LED signal indicator
- Mounting Hole Diameter: 3 mm
- Default Baud Rate: 9600 bps
- Module size 23mm * 30mm
- Antenna size 12 * 12mm
- Cable:20mm

Features:

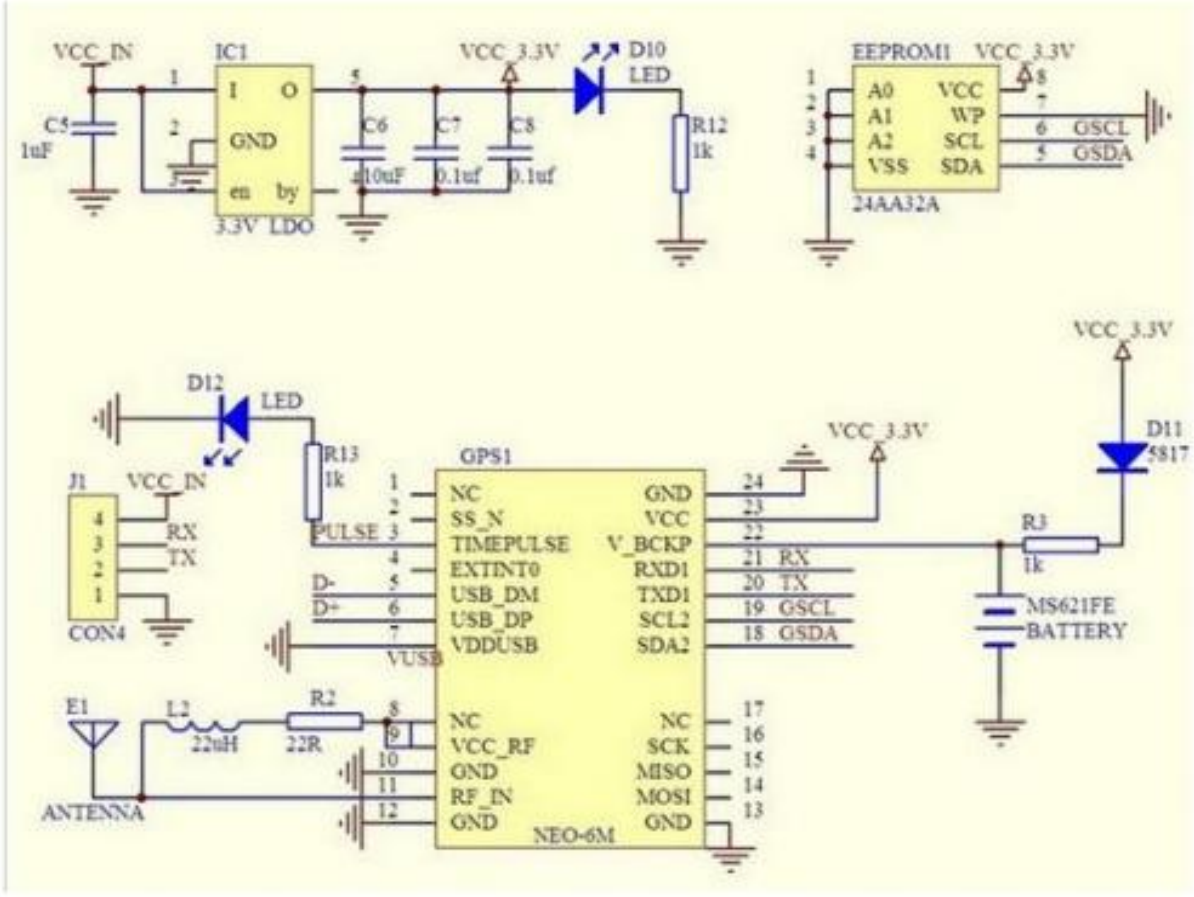
- Use XM37-1612 module, MTK Platform, with high-gain active antenna
- TTL level, compatible with 3.3V/5V system
- The default baud rate: 9600
- With rechargeable backup battery, can save the ephemeris data when it power down, and make the warm start.
- Suitable for RC quad copter, navigator

Schéma électrique du module GPS NEO6MV2

Pin out:

VCC: Connect 3.3V/5V
GND: Connect GND
TXD: Connect MCU.RX
RXD: Connect MCU.TX

Schematic:



Module sim800l GSM/GPRS

2.3. Functional Diagram

The following figure shows a functional diagram of SIM800L:

- GSM baseband
- GSM RF
- Antenna interface
- Other interface

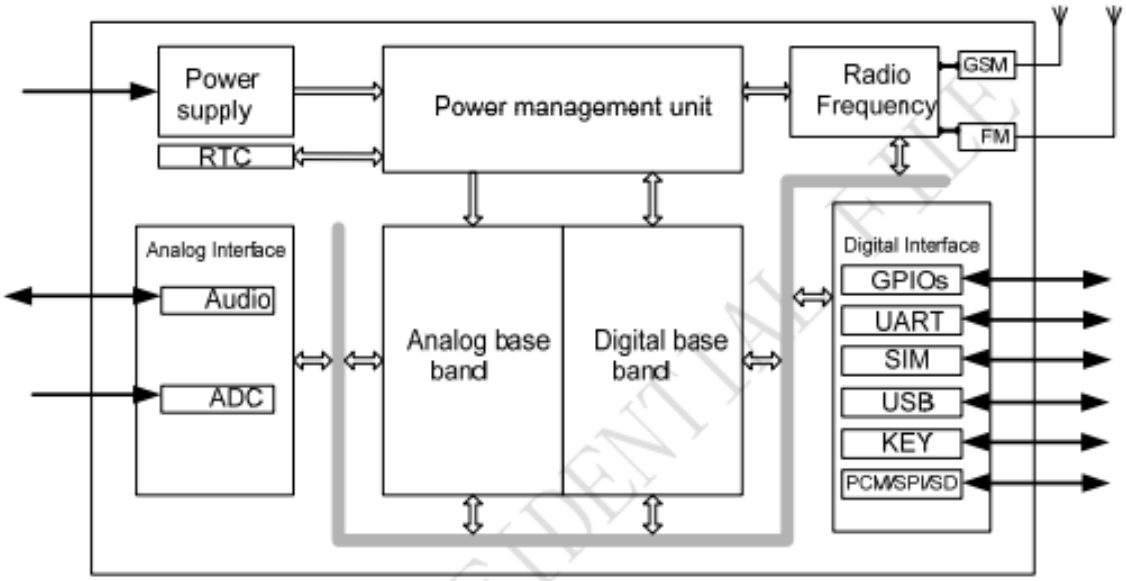


Figure 1: SIM800L functional diagram

Arduino E/S pin configuration

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

#include<SoftwareSerial.h>

#define SIM800_TX_PIN 8 // SIM800 TX est connecté à Arduino D8

#define SIM800_RX_PIN 7 // SIM800l RX est connecté à Arduino D7

SoftwareSerial SerialSIM800(SIM800_TX_PIN, SIM800_RX_PIN); //Create software serial object to communicate with SIM800l

 SoftwareSerial gpsSerial(4, 3); // La connexion série à l'appareil GPS

TinyGPSPlus gps; // The TinyGPS++ object


String MSG;

int num = 0;

int GroundPin = A0;

int sensePin = A1;

int buzzer = 11;

int sms_count = 0;

int sms_count2 = 0;

int threshold = 100;


void setup()

{

  Serial.begin(9600);

  gpsSerial.begin(9600);

  SerialSIM800.begin(9600);
```

```
  pinMode(GroundPin, OUTPUT);
  digitalWrite(GroundPin, LOW);
  pinMode(buzzer, OUTPUT);
  Serial.println("Testing GSM SIM800L");
  SerialSIM800.println("AT+CMGF=1");  // Définir en mode texte
  delay(1000);
  SerialSIM800.println("AT+CNMI=1, 2, 0, 0, 0"); // Définir sur la notification du nouveau message, Indicateur de nouveau message
}
```

Une partie du programme

```
void loop()
{
  while (SerialSIM800.available()) {
    MSG = SerialSIM800.readString();
    Serial.println("Hayaaaaaa");
    if (MSG.indexOf("ON") != -1) {
      Serial.println(MSG);
      Serial.println("Alarm set to ON");
      Alert();
      sms_count = 0;
    }

    else if (MSG.indexOf("Off") != -1) {
      Serial.println("Alarm set to OFF");
      SendTextMessage2();

    }

    else if (MSG.indexOf("STATUS") != -1) {
      Serial.println("Localisation de la voiture .");
      SendTextMessage3();
    }
  }}
```

```
void GPSMODE()
{
 gpsSerial.listen();
  do {
    while (gpsSerial.available() > 0)
      if (gps.encode(gpsSerial.read()))
        delay(1000);
    displayInfo();
    num++;
  } while (num < 20);
  num = 0;
  String buf;
  buf += F("your location is \nlat:");
  buf += String(gps.location.lat(), 6);
  buf += F("\nlong:");
  buf += String(gps.location.lng(), 6);
  Serial.println(buf);
  Serial.println("delay");
  delay(1000) ;
}
void SendTextMessage ()
{
  //Begin serial comunication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);
```

```
while (!Serial);


// Etre en communication série avec Arduino et SIM800l
SerialSIM800.begin(9600);
delay(1000);

Serial.println("Setup Complete!");
Serial.println("Sending SMS...");

// Définir le format SMS en ASCII
SerialSIM800.write("AT+CMGF=1\r\n");
delay(100);

// Envoyer une nouvelle commande SMS et un numéro de message

SerialSIM800.write("AT+CMGS=\"00213558266442\"\r\n");  //Working
delay(100);

//Send SMS content
SerialSIM800.write("Alert! l'alarme de votre voiture est Activé!!!.");
delay(100);
// Envoie Ctrl + Z / ESC pour indiquer que le message SMS est complet
SerialSIM800.write((char)26);
delay(100);
```

```
  Serial.println("SMS Sent!");
  sms_count++;
  delay(5000);
}
```

Résumé : Notre travail consiste à réaliser un système à temps réel pour la surveillance intelligente des automobiles. Le système réalisé permet, d'une part la détection et la prévention contre le vol d'un automobile, et d'autre part, il permet de repérer en cordonnés GPS la position de l'automobile volé. Le circuit du système réalisé est basé sur l'utilisation d'une carte Arduino UNO, un module GSM, un module GPRS et un capteur de vibration. Quand le capteur de vibrations détecte un niveau de vibration élevé, le module GSM via Arduino envoi un SMS d'alerte au téléphone portable du propriétaire. Ce dernier recevra par la suite les cordonné GPS permettant ainsi la localisation de l'automobile volé sur « Map », cela à temps réel. Le système se présente sous forme d'un petit boitier de poids faible qui contient tous les composants du dispositif réalisé.

Mots clés : système à temps réel de surveillance, Arduino UNO, module GSM, module GPRS , capteur de vibration.

Abstract: Our work is to build a real-time system cars smart monitoring. The realized system allows, on the one hand the detection and the prevention against the theft of an automobile, and on the other hand, it makes it possible to locate in GPS coordinates the position of the stolen automobile. The realized system circuit is based on the use of an Arduino UNO board, a GSM module, a GPRS module and a vibration sensor. When the vibration sensor detects a high vibration level, the GSM module via Arduino sends an alert SMS to the owner's mobile phone. The latter will subsequently receive the GPS coordinates allowing the location of the car stolen on «Map», this in real time. The system is in the form of a small, low-weight box that contains all the components of the realized device.

Key words: Real-time monitoring system, Arduino UNO, GSM module, GPRS module, vibration sensor.

ملخص: مهمتنا هي بناء نظام في الوقت الحقيقي للمراقبة الذكية للسيارات. يسمح النظام المتحقق، من جهة، بالكشف والوقاية من سرقة السيارات، ومن ناحية أخرى، يتيح تحديد موقع السيارة المسروقة في نظام GPS. تعتمد دارة النظام المحققة على استخدام لوحة Arduino UNO، ووحدة GSM، ووحدة GPRS ومستشعر اهتزاز. عندما يكتشف مستشعر الاهتزاز مستوى اهتزاز عالي، تقوم وحدة GSM عبر اردوينو بإرسال رسالة تنبيه إلى الهاتف المحمول الخاص بالمالك. وسيتلقى الأخير لاحقًا إحداثيات نظام تحديد المواقع العالمي التي تسمح بموقع السيارة المسروقة على «خريطة»، وذلك في الوقت الفعلي. النظام عبارة عن صندوق صغير منخفض الوزن يحتوي على جميع مكونات الجهاز المتحقق.

الكلمات المفتاحية: نظام مراقبة في الوقت الحقيقي، Arduino UNO، وحدة GSM، وحدة GPRS، جهاز استشعار الاهتزاز.