Table des matières

Liste des tableaux

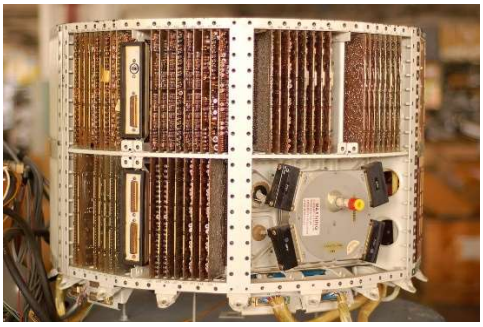
Liste des figures

1. Introduction – Généralités

1.1 Historique

Les premiers systèmes embarqués, c'est-à-dire les premiers ordinateurs autonomes ayant les ressources nécessaires afin de pouvoir fonctionner dans un environnement externe, ont fait leur apparition au début des années 1960 aux Etats-Unis.

Figure 1 : Illustration de l'Autonetics-D17, l'ordinateur de guidage



(Wikipédia, D17-B, 2018)

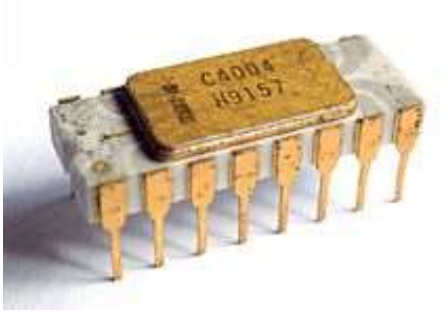
Le tout premier système embarqué a été conçu en 1962 dans le but de guider le missile nucléaire Minuteman I. Son poids était de 28 kg et contenait plusieurs circuits intégrés afin de réaliser les tâches qui lui étaient dédiées comme le guidage du missile selon le positionnement du missile.

Figure 2 : Illustration de l'Apollo Guidance Computer et son écran d'utilisation

(Wikipédia, Apollo Guidance Computer, 2018)

L'Apollo Guidance Computer a été également un des tous premiers systèmes embarqués, qui comme l'Autonetics D-17, a pour principal but de guider son appareil. Il a été conçu au début des années 1960 pour enfin être testé et utilisé en 1966 lors du programme Apollo qui représente un voyage sur la lune via une fusée spatiale. Deux de ce système étaient placés dans le vaisseau afin de la guider. Son poids était de 32 kg et contenait plusieurs centaines de circuits intégrés.

Figure 3 : Premier microprocesseur Intel 4004

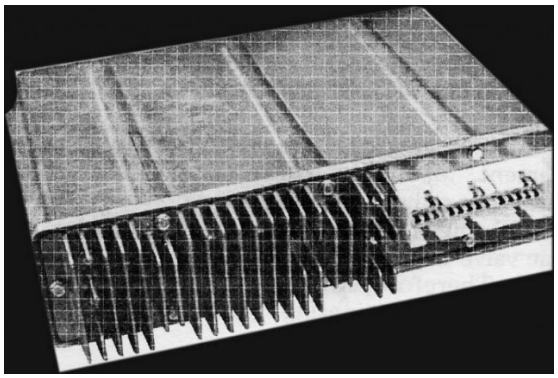


(Wikipédia, Intel 4004, 2018)

Ceci est le premier microprocesseur apparu en 1971 par l'entreprise Intel. C'est le premier système embarqué ayant la capacité à pouvoir incorporer tous les éléments d'un ordinateur (mémoire, contrôle d'accès, unité de calcul) dans un même circuit intégré. Auparavant, plusieurs circuits ayant des fonctions spécifiques devaient travailler ensemble pour accomplir une tâche. Or, depuis l'apparition de ce microprocesseur, toutes les tâches pouvaient être réalisées par un seul composant. C'est avec ce type de système que l'ère de l'informatique embarquée débuta en faisant son apparition dans l'industrie du multimédia, de l'électroménager et de l'automobile.

Dans l'automobile, les premiers systèmes embarqués sont apparus au début des années 1970. Les sociétés automobiles surfèrent sur la vague de l'innovation technologique des systèmes embarqués afin d'en tirer profit et de pouvoir les utiliser dans leurs propres véhicules.

Figure 4 : Premier Electronic Control Unit (ECU)



(Craig Delzangle, Embedded Systems in Automobiles)

L'Electronic Control Unit (Unité de Contrôle Electronique) représente tout système qui permet la gestion de fonctions dans un véhicule. L'un des premiers a été conçu par Chevrolet pour la Chevrolet Cosworth Vega en 1975, il permettait la gestion complète

du moteur et notamment la transmission automatique du carburant aux cylindres. Une dizaine de capteurs lui transmettaient les informations nécessaires à la réalisation de cette tâche.

1.2 Les systèmes embarqués …

1.2.1 Définition

Un système embarqué est un ensemble d'éléments informatiques et électroniques interagissant entre eux de façon autonome et complémentaire. Ces systèmes sont conçus de manière à pouvoir répondre spécifiquement aux besoins de leur environnement respectif.

Le terme « système » désigne l'ensemble des éléments qui constituent le système embarqué, souvent ces systèmes sont composés de sous-systèmes étant donné leur complexité.

Le terme « embarqué » représente la mobilité et l'autonomie du système en interaction directe avec son environnement dans l'exécution de tâches précises, afin de répondre à la finalité de celui-ci.

Contrairement aux systèmes classiques, les systèmes embarqués sont conçus pour réaliser des tâches bien précises. Certains doivent répondre à des contraintes de temps réel pour des raisons de fiabilité et de sécurité, indispensables selon l'utilisation du système.

Un système embarqué regroupe à la fois la partie software (logicielle) et la partie hardware (matériaux) étroitement liées afin de produire les résultats escomptés.

1.2.2 Contraintes

Du fait que ce type de système soit « embarqué » ou « enfoui », plusieurs contraintes lui sont imposées. Les systèmes embarqués se retrouvent aujourd'hui partout dans différents types d'environnements (téléphone, véhicule, avion…) et liés à différents types d'utilisations. C'est le secteur d'activité dans lequel est utilisé le système qui va permettre de définir ses contraintes. Voici un aperçu général des principales contraintes des systèmes embarqués :

Tableau 1 : Tableau des contraintes de système embarqué selon secteur d'activité

Secteur d'activité	Contraintes
Équipements scientifiques	Performances, fiabilité, coût
Équipements militaires et aérospatiaux	Performances, fiabilité, pérennité, intégration
Transports	Fiabilité, coût, interactivité
Informatique industrielle	Fiabilité, coût, pérennité
Matériel de bureau	Performance, coût, standardisation
Réseau et télécommunications	Performance, fiabilité, intégration
Électronique grand public	Performance, coût, design / intégration

(Institut d'électronique et d'informatique Gaspard-Monge, IGM, 2002)

Toutes ces contraintes sont à respecter lors de la conception d'un système embarqué. Certaines sont importantes comme le design ou le coût mais d'autres sont indispensables comme la fiabilité ou la performance, sans quoi le système ne peut être mis en production. La contrainte de la taille physique peut également s'ajouter, par exemple un téléphone portable a peu de place pour contenir un système embarqué. L'enjeu est donc de pouvoir réaliser un système puissant répondant aux contraintes précitées, sous forme physique assez réduite pour être exploitable.

Etant autonome, ces systèmes nécessitent une alimentation en énergie régulière afin d'avoir un fonctionnement stable et sûr du produit. Il est donc indispensable de prévoir une alimentation adaptée, même si une consommation trop élevée du système aurait un impact sur son coût.

1.2.3 Caractéristiques

Les résultats produits par un système embarqué pourront être au profit d'un environnement externe, ou autrement dit, à un autre système plus général qui englobe le système embarqué, faisant appel à ses services dès que celui-ci les nécessite. Ainsi, un système embarqué serait inutile lorsqu'il est seul ou isolé, puisque son but est de fournir un service telle une action ou la transmission d'informations à un autre système plus volumineux. D'ailleurs, le terme de système « enfoui » fait référence à l'intégration et au rattachement dissimulé du système à celui qui l'englobe.

Les systèmes embarqués évoluent le plus souvent dans des environnements instables et non maitrisés, les obligeant à anticiper tous évènements particuliers pouvant les

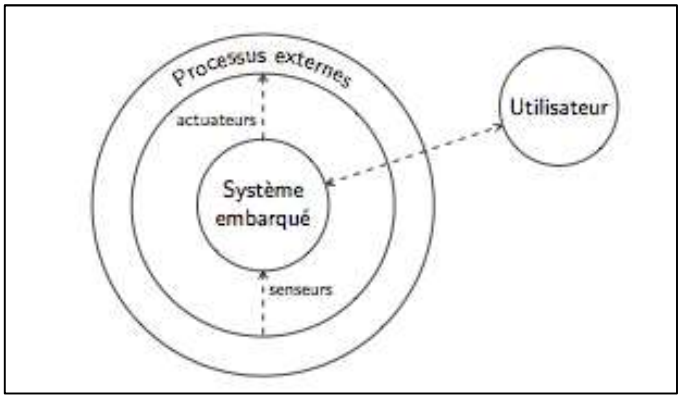
perturber. Notamment les chocs, les fortes températures, les vibrations, l'humidité ou encore d'autres circonstances pouvant porter atteinte à la fiabilité et à la performance du système.

En plus de leurs évolutions technologiques, ces systèmes doivent évoluer au niveau de la qualité des matériaux utilisés afin d'éviter toute défaillance du système qui pourrait être critique dans le cas d'une application médicale, aéronautique ou encore automobile, pouvant aller jusqu'à la mise en danger de vies humaines.

"A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe." (Real-Time Systems: Design Principles for Distributed Embedded Applications, Hermann Kopetz, (avril 2011), ISBN 978-1-4419-8237-7).

Les systèmes embarqués ont été réalisés dans le but de produire des tâches simples dans un concept d'entrées et de sorties de données. Cependant, le traitement de ces tâches peut être aussi complexe que dans un ordinateur classique.

Figure 5 : Représentation de l'environnement général d'un système embarqué et de ses interactions



(Sébastien Combéfis, Programmation de systèmes embarqués : de l'hardware au software)

Les senseurs sont synonymes de capteurs. Ils représentent les entrées au processus du système afin d'effectuer les calculs et les traitements nécessaires. Les actuateurs, synonymes d'actionneurs, sont les déclencheurs des actions émises par le système une fois le traitement réalisé. Ils représentent les sorties.

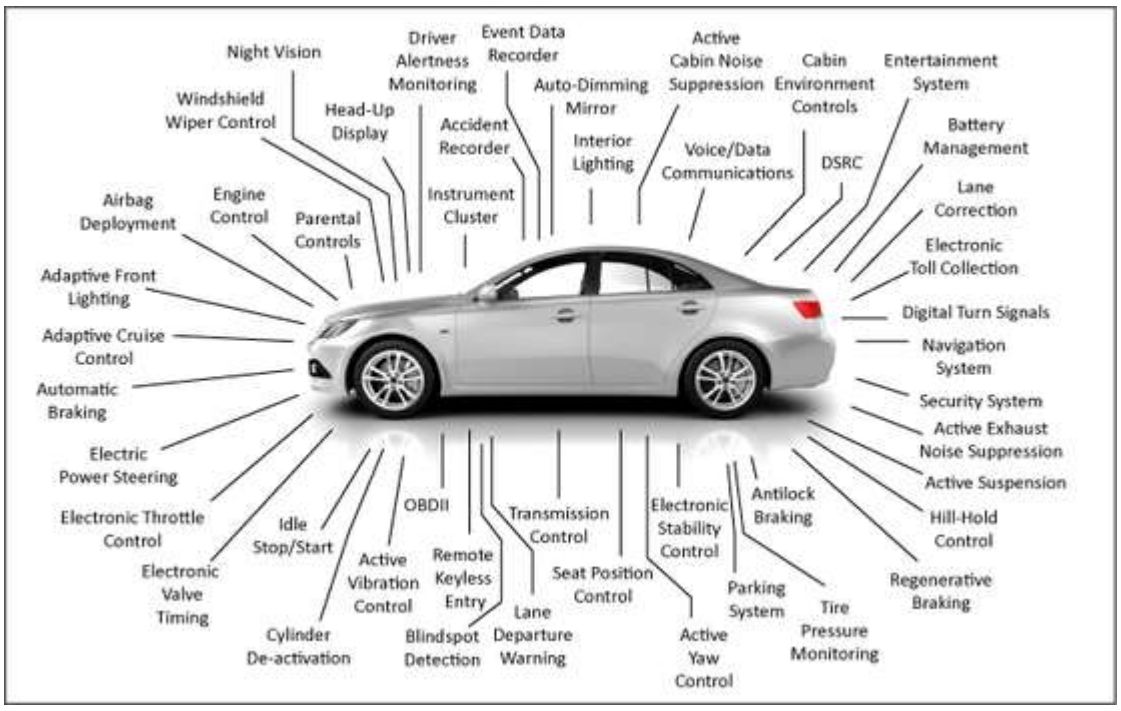
Cette figure illustre l'image d'encapsulation d'un système embarqué par son environnement (ici les processus externes) qui interagit directement avec lui. Des moyens de communications existent en tant qu'intermédiaire propre à chaque environnement.

1.3 … Dans l'automobile

1.3.1 Contexte

Aujourd'hui, un véhicule contient une grande quantité d'électronique et d'informatique : on retrouve plus de 100 capteurs, 30 à 50 calculateurs selon le type de véhicule et parfois près d'un million de lignes de codes pour les véhicules de dernière génération. Cette évolution s'explique par les demandes exigeantes des consommateurs et l'envie de différenciation des concurrents sur marché de l'automobile. S'ajoute à cela les contraintes économiques et écologiques où l'électronique embarqué répond à ces nouvelles attentes. De nouvelles fonctionnalités impliquent parfois une intégration électronique et informatique par le biais de systèmes embarqués. Voici une représentation des systèmes intégrés d'un véhicule moderne.

Figure 6 : Fonctions d'un véhicule moderne



Toutes ces fonctions se catégorisent selon leur domaine d'action :

- Habitacle / Confort (climatisation, siège chauffant, allumage automatique des feux…)
- Moteur / Transmission (contrôle injection, commande boite de vitesses…)
- Sécurité (ABS, Airbags, ESP, radar de recul…)

Toutes ces fonctions sont gérées par des ECUs (Electronic Control Unit) qui représentent les calculateurs présents dans les véhicules. Ce sont de petits boitiers noirs ayant chacun leurs spécificités et leurs rôles liés à des capteurs et des actionneurs.

Figure 7 : Illustration d'un calculateur moteur

Celui-ci est dédié uniquement au contrôle moteur, mais de nombreux autres calculateurs (ou ECU) sont présents dans le véhicule pouvant ainsi gérer d'autres fonctions. Voici une liste non exhaustive de différents types d'ECU :

Tableau 2 : Les différents types d'ECU

Abréviation	**Désignation**	**Utilité**
ECU ou ECM	Engine Control Unit	Système permettant la gestion du bloc moteur
SCU	Speed Control Unit	Système de régulation de vitesse, permet de rouler à une vitesse constante
TCU	Telematic Control Unit	Permet de connaitre le positionnement du véhicule et les coordonnées GPS en temps réel
BCM	Brake Control Module	Système représentant l'ABS, permettant l'aide au freinage lors des freinages d'urgences
BMS	Battery Management System	Système permettant de réguler la batterie du véhicule

Tous ces ECUs sont reliés à des capteurs et des actionneurs leur permettant d'envoyer et de traiter les informations. Une communication est donc présente entre tous ces composants électroniques via des bus de communication. Toute cette composition forme l'électronique embarqué du véhicule.

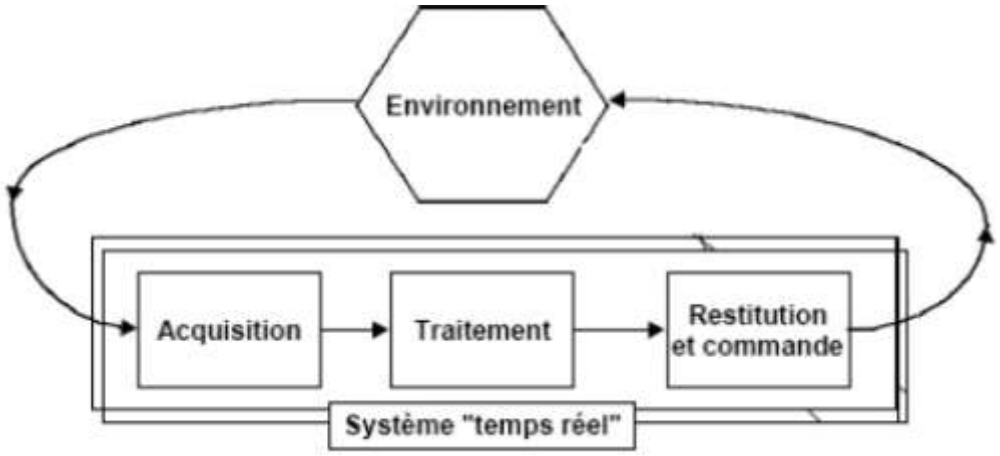
1.3.2 Le temps réel

Comme déjà spécifié, les systèmes embarqués sont soumis à des contraintes différentes selon leur domaine d'utilisation. Et bien, le temps réel est l'une des contraintes primordiales dans le secteur de l'automobile au niveau de la performance mais surtout au niveau de la sécurité. Devant un ordinateur classique, quelques minutes de latence ne pourront affecter que l'humeur de l'utilisateur, sur un système embarqué d'automobile seuls quelques secondes de latence suffisent à provoquer un accident avec des conséquences terribles.

Le temps réel est le fait d'être constamment en adéquation temporelle avec la réalité. Un système en temps réel est un système qui doit, non seulement, produire un résultat juste mais dans une durée limitée, sans quoi ce résultat deviendrait erroné. Ainsi, le système en temps réel doit fournir un résultat avec une contrainte de temps.

Le temps est déterminé par l'environnement dans lequel se trouve le système, celui-ci doit avoir l'image la plus réaliste de celle de son environnement externe qui évolue lui-même avec le temps.

Figure 8 : Représentation d'un système en temps réel avec son environnement



(Khalifa MANSOURI, 2007)

L'échelle de temps dépend de l'environnement dans lequel le système est utilisé, il peut varier de quelques millisecondes à plusieurs heures. Lorsque nous sommes dans notre véhicule, il est préférable de voir l'allure à laquelle nous roulons aux millisecondes prés,

l'affichage de la vitesse à quelques minutes d'intervalles rendrait les données complètement fausses.

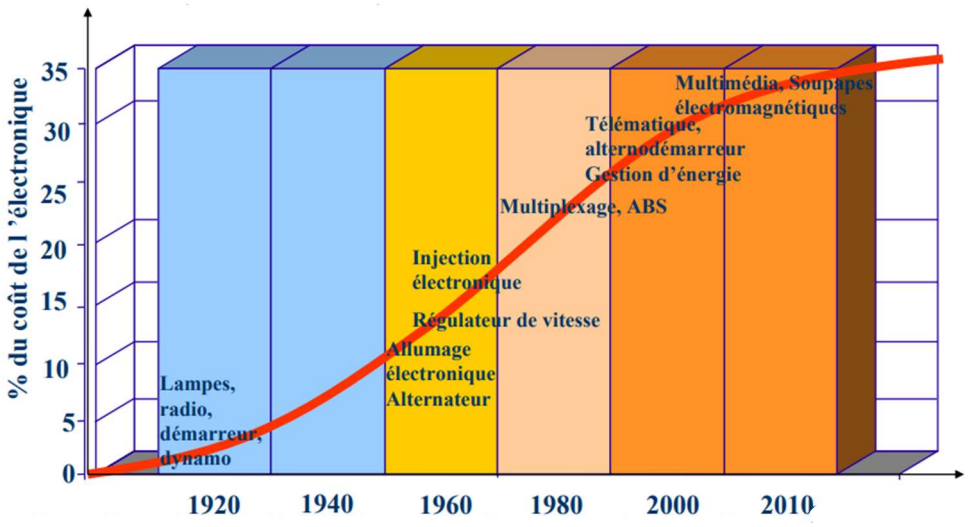
1.3.3 Quelques chiffres

Un véhicule renferme aujourd'hui plus de lignes de code qu'un avion Airbus de première génération, soit près d'un million de lignes pour les véhicules haut de gamme. (Institut Universitaire de Belford-Montbeliard, 2013)

On compte de nos jours jusqu'à 80 calculateurs par voiture, un constat flagrant sur l'évolution des systèmes embarqués dans l'automobile. (Marc Alias, Ingénieur automobile, 2014)

Ainsi, la part de l'électronique des véhicules a connu un essor de grande envergure comme en témoigne ce graphique.

Figure 9 : Evolution du coût de l'électronique dans l'automobile



(Khalifa MANSOURI, 2007)

Nous constatons que le coût de l'électronique représente aujourd'hui près de 40% du prix total du véhicule. Certains véhicules de luxe valent aujourd'hui plus de CHF 150'000.- due notamment à la quantité d'électronique qui y est intégrée pouvant proposer une multitude d'options.

Ainsi, les systèmes embarqués sont en plein essor avec les premières voitures autonomes contenant beaucoup plus d'informatique embarqué que des véhicules classiques et de plus en plus connectés au monde extérieur.

2. Fonctionnement des systèmes embarqués

2.1 Composition d'un système embarqué

2.1.1 Calculateur

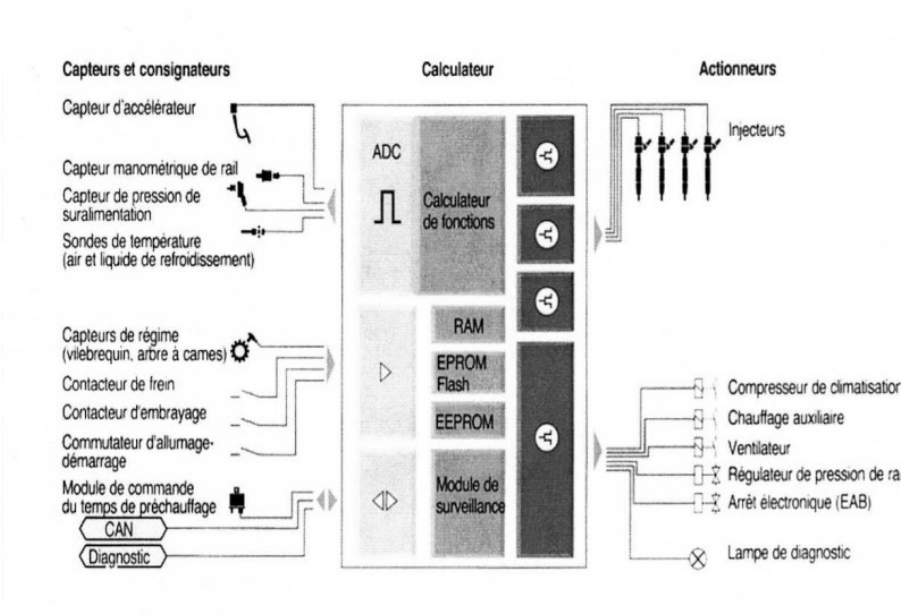
Le calculateur est l'élément principal d'un système embarqué automobile où régit la mémoire, la carte-mère ou encore le traitement logiciel. Chacun des calculateurs automobiles sont dédiés au pilotage d'une ou certaines tâches bien précises, ainsi de nombreux calculateurs sont présents dans les véhicules formant son système électronique.

Le calculateur correspond à un boitier contenant des broches électriques dotées de nombreux ports d'entrées et de sorties afin de permettre à la gestion des instruments de bord. Une carte programmable composée de circuits imprimés contient tout le traitement informatique du système, principalement codé en langage C++ ou Java, et s'accompagne d'autres éléments formant le calculateur.

Nous pouvons les qualifier de systèmes « intelligents » due à leur capacité de prise de décision en fonction des paramètres d'entrées via des capteurs (ou sondeurs).

Dans le cas d'un calculateur moteur, son but précis sera d'assurer les fonctions de pilotage d'un moteur en ajustant en temps réel les besoins du moteur. En recevant des signaux électriques de la part des capteurs (sonde de température, capteur de pression…), le calculateur peut traiter ces informations pour les transformer en actions par l'intermédiaire d'actionneurs (injecteur, vanne EGR…).

Figure 10 : Représentation d'un calculateur moteur et ses interactions



2.1.2 Actionneurs

Lorsque le traitement est réalisé par le calculateur, un signal électrique est transmis aux actionneurs permettant une action physique sur le véhicule.

Tableau 3 : Exemples d'actionneurs

Paramètre	Actionneurs (ou actuateurs)
Temps d'injection <=	**Injecteur**
Étincelle de combustion <=	**Bobine d'allumage**
Suppression NOx (oxyde d'azote) <=	**Vanne EGR**
Pression turbo <=	**Électrovanne turbo**
Production de froid <=	**Compresseur climatisation**
Verrouillage des portières <=	**Actionneur de porte**
Mouvement des vitres <=	**Moteur lève-vitre**

(Christophe Poupinel, Calculateurs moteur pour voiture)

Ces actionneurs (ou actuateurs) transforment le signal électrique reçu en énergie mécanique. Cette transformation d'énergie peut être réalisée par moteur, de façon magnétique, hydraulique ou optique.

2.1.3 Capteurs

Les capteurs sont des éléments essentiels au fonctionnement des calculateurs puisque ce sont ces composants qui sont en charge de transmettre l'information afin d'être traitée de manière optimale.

Leur principal objectif est donc de renseigner le calculateur qui va pouvoir agir en temps réel avec l'environnement, c'est pourquoi, ces capteurs envoient de façon constante des informations en continu au calculateur relié. De plus en plus de capteurs sont élaborés due à la sophistication des nouveaux véhicules.

Précisément, leurs tâches consistent à pouvoir transformer une grandeur physique (température, pression…) en un signal électrique afin de le transmettre au calculateur. En effet, des interrupteurs peuvent être considérés comme des capteurs puisque les informations qui résultent de l'action émise par l'utilisateur sont directement transmises au calculateur.

Tableau 4 : Exemples de capteurs

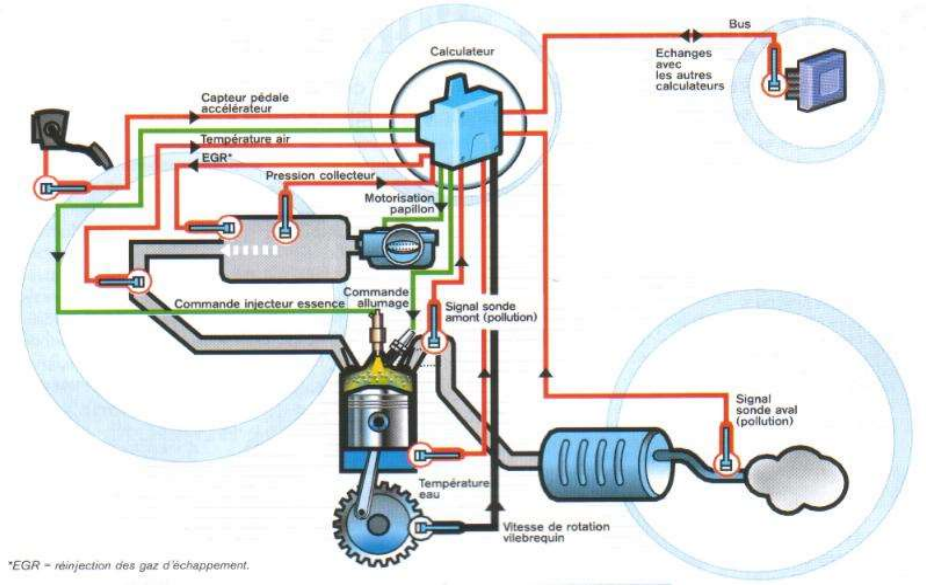
Information de grandeur physique	Signaux électriques
Température =>	Sonde de température
Position / Vitesse =>	Capteur PMH
Débit d'air =>	Débitmètre
Pression =>	Capteur pression
Vibrations =>	Capteur cliquetis
Angle =>	Capteur gyroscopique
Taux d'oxygène =>	Sonde lambda
Taux d'humidité =>	Capteur de pluie
Lumière =>	Capteur de luminosité

(Christophe Poupinel, Calculateurs moteur pour voiture)

2.1.4 Communication

Tous ces composants échangent entre eux par l'intermédiaire de faisceaux, ils correspondent à de petits câbles permettant la transmission des signaux électriques contenant les informations recueillies et à transmettre.

Figure 11 : Communication capteur – calculateur – actionneur



(Renault)

Cette figure illustre la communication et les échanges entre le calculateur et ses actionneurs et capteurs. En vert, les interactions avec les actionneurs comme la

commande d'allumage et en rouge, les interactions avec les capteurs, correspondant tous à des faisceaux électriques.

2.2 Architecture système automobile

2.2.1 Communication – Multiplexage

Dans un véhicule, chaque ECU (ou calculateur) gère son propre système, cependant il est possible pour un système de pouvoir interagir et échanger des informations avec tous les autres calculateurs contenus dans le véhicule.

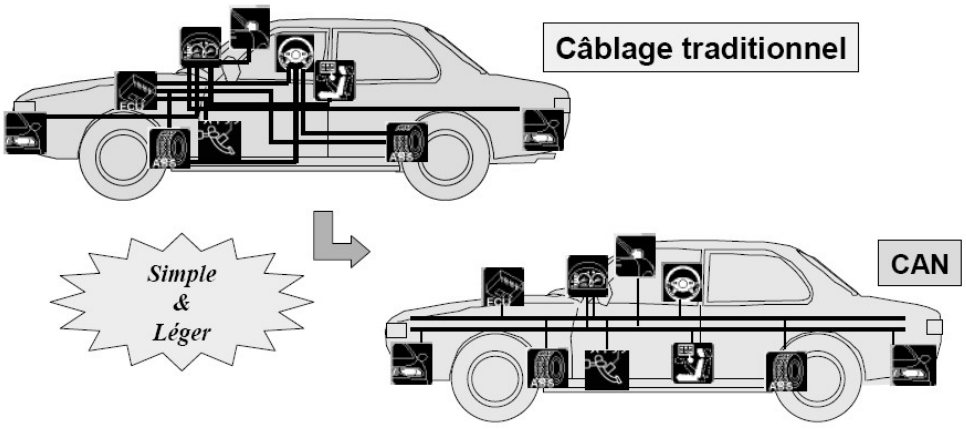
Pour ce faire, les calculateurs communiquent entre eux par types de langages différents suivants les fonctions qui leurs sont dédiées (gestion moteur, gestion habitacle…). Ceci représente un réseau multiplexé.

Lors de l'introduction des systèmes embarqués dans les véhicules, le réseau utilisé était point-to-point, c'est-à-dire que chaque système était relié directement à un autre par l'intermédiaire de câbles. Cette méthode convient lorsque peu de systèmes sont installés car elle nécessite un nouveau câblage à chaque composant ajouté.

Ainsi, lors de l'augmentation du nombre de systèmes embarqués dans les véhicules, des kilomètres de fils et de câbles se sont vus entasser dans les véhicules impliquant une forte quantité de poids mais surtout un risque de panne plus élevé. Ceci représentait notamment un coût important de la part du constructeur ainsi qu'un espace restreint pour l'éventuel ajout de système.

C'est ainsi que le multiplexage a dû faire place permettant les transitions d'informations sur un seul câblage via des protocoles de langages dédiés.

Figure 12 : Changement de méthode de communication automobile



(APEM, 2016)

CAN (Controller Area Network) fait partie des nombreux protocoles de communication d'un réseau multiplexé tels que le VAN, le LIN ou encore le MOST. Le bus CAN reste le protocole le plus utilisé dans les véhicules. Le protocole représente la manière dont sont encodées les informations circulant dans le bus.

Deux boitiers (ou supercalculateurs) électroniques spécifiques à la communication du bus sont présents dans l'habitacle afin de traiter les informations contenues. Un premier permet simplement d'encoder les informations émises par les capteurs ou calculateurs selon le protocole de communication et de les envoyer au bus de communication. Un second permet le décodage de ces informations pour les trier et les transmettre aux autres systèmes, tels que des calculateurs ou actionneurs.

En effet, plusieurs protocoles peuvent être utilisés dans un même véhicule dépendant de la catégorie de fonctionnalités que propose le système ayant des besoins en termes de fiabilités, de débits et de coûts différents. Un réseau automobile peut se retrouver partagé en plusieurs sous-réseaux correspondant aux différents domaines de fonctionnalités.

Ces « familles » de spécifications ont été classifié par la SAE (Society for Automotive Engineers) afin de déterminer les réseaux à adopter en fonction des besoins du système.

Tableau 5 : Classification réseaux selon la SAE

Classe	Débit	Usage	Exemples
A	<10kb/s	Contrôle de l'habitacle	LIN
B	10kb/s → 125kb/s	Transfert de données non-critiques	CAN-B (Low-speed CAN)
C	125kb/s → 1Mb/s	Communications temps-réel critiques	CAN-C (High-speed CAN)
D	>1Mb/s	Multimédia ou X-by-wire	MOST, FlexRay

(Ivan Stundia, Détection d'intrusion pour des réseaux embarqués automobiles, 2015)

L'usage correspond aux domaines des fonctionnalités nécessitant un réseau multiplexé. Le débit utilisé au réseau va dépendre des critères de temps réel demandés par les systèmes. La fiabilité, le coût, la sécurité ou encore la qualité de service font partie intégrante du choix du réseau.

2.2.2 Le bus CAN

Ce type de communication a été conçu et testé dans les années 1990 sur un modèle Mercedes puis a pris place dans la plupart des véhicules haut de gamme, pour être maintenant l'un des réseaux les plus utilisés de l'automobile.

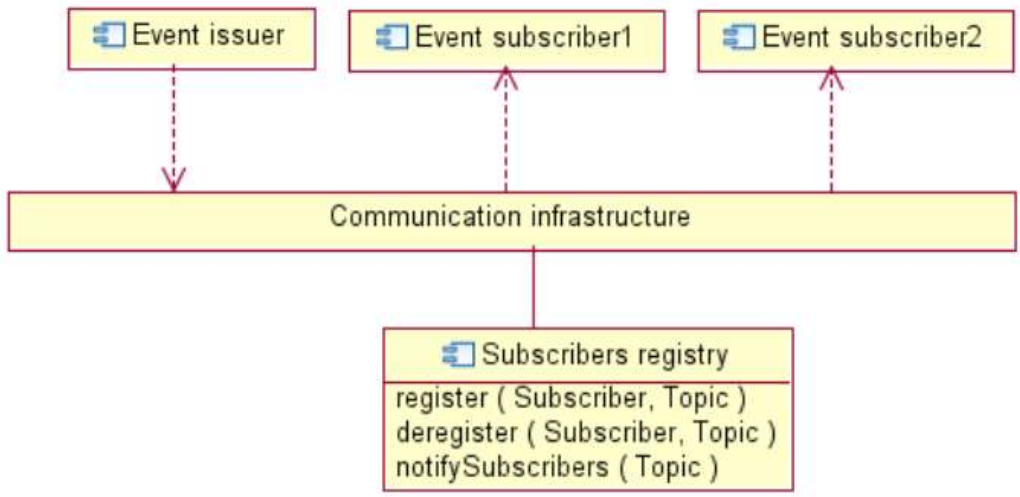
Ce type de réseaux a été normalisé en 1991 par la norme ISO 11898.

Il en existe deux types : le bus CAN-B (Low Speed) et le bus CAN-C (High Speed). Le premier est composé d'un identifiant de 11 bits, peut contenir jusqu'à 20 nœuds (système relié au bus) et peut atteindre un débit maximal de 125 kb/s. Le second est composé d'un identifiant de 29 bits, peut contenir jusqu'à 30 nœuds et peut atteindre un débit maximal de 1 mb/s. Nous nous intéresserons au bus CAN-B, même si les deux ont le même principe de fonctionnement avec seulement quelques caractéristiques différentes.

Physiquement, le bus est représenté par une paire de fils torsadée empêchant ainsi tout risque de parasite électrique. Contrairement aux faisceaux électriques, ce type de câblage est alimenté par un réseau permettant la détection et la correction d'erreurs grâce à son protocole.

Ce protocole de communication a pour principe de relier tous les systèmes à un même bus de communication permettant ainsi de faire passer les informations au même endroit et de les redistribuer à qui de droit. Cette méthode de communication fait référence au pattern Publish-Subrcribe qui consiste à une inscription de plusieurs composants à un bus de communication pouvant ainsi « s'abonner » au types d'informations qu'ils nécessitent ou qu'ils souhaitent transmettre. Ainsi, un registre se chargera d'enregistrer les « abonnements » des composants afin que chacun puisse recevoir les bonnes informations ou de pouvoir les transmettre.

Figure 13 : Architecture du pattern Publish-Subscribe



(Philippe Dugerdil, Professeur HES, 2018)

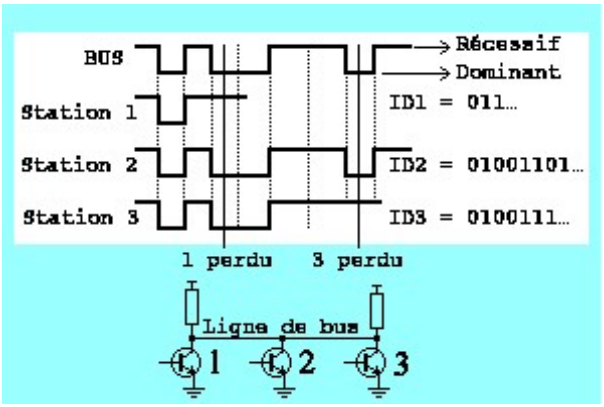
Les principaux avantages de ce pattern sont de pouvoir réduire fortement les chemins de communication, placer un intermédiaire entre tous ces composants et de standardiser les méthodes de communication.

Le bus CAN s'inspire donc de ce concept de modélisation en se basant sur le principe de diffusion générale où aucun nœud n'est visé particulièrement, mais grâce au contenu du message possédant un identifiant, ces nœuds (ou abonnés) peuvent savoir si ce message leur est destiné ou non. Ainsi, chaque système qui est constamment en écoute sur le bus peut faire le choix de garder l'information ou simplement de l'ignorer.

Avec un bus CAN fonctionnant de façon asynchrone, plusieurs nœuds transmettent simultanément des informations au bus de communication, l'obligeant ainsi à prioriser les informations à faire circuler selon les contraintes de temps réel qu'imposent certains systèmes.

Ceci est géré par le principe d'arbitrage, qui consiste à prioriser un message par l'intermédiaire de son identifiant. Les priorités sont définies par des valeurs binaires lors de l'analyse conceptuelle du réseau et ne pourront subir aucune modification dynamique. Ainsi, lors de l'émission d'un message de la part d'un nœud, son identifiant doit, de façon binaire, être le plus proche de la conception de la priorité réalisée par le bus de communication.

Figure 14 : Représentation du principe d'arbitrage



(Éric Delaunay, Le bus CAN)

Une comparaison bit à bit est donc réalisée de manière à différencier la priorité du message. On nomme bit récessif le bit égal à 1 et bit dominant celui égal à 0 puisque c'est cette valeur qui écrase l'autre. Ce schéma illustre bien cette « compétition » avec trois stations / nœuds souhaitant envoyer de façon simultané un message. Nous pouvons constater que c'est le nœud 2 qui a la plus haute priorité puisque c'est celui qui

correspond le plus à la conception de priorité du bus. Cependant, les messages des autres nœuds ne disparaissent pas, ils attendent que le bus soit de nouveau libre pour envoyer leurs messages pendant qu'eux-mêmes soient récepteurs du message.

En réalité ce sont des trames qui sont envoyées au bus de communication contenant le message à transmettre encodées par des valeurs binaires. C'est également dans ces trames que contiennent les identifiants du message mais d'autres éléments encore.

Une trame CAN se décompose de la façon suivante :

- Un identificateur de 11 bits permettant aux nœuds de savoir si l'information contenue leur est destinée. Permet également l'arbitrage des priorités.
- Un bit RTR (Remote Transmission Requete) est à 1 si la trame correspond à une trame de données ou 0 dans le cas d'une trame de requête de données.
- Le champ de contrôle indique la taille des données transmises en octets.
- Le champ de données contient le message allant jusqu'à 8 octets (64 bits), peut être vide en cas de requête de données.
- Un CRC (Cyclic Redundancy Check) de 16 bits qui permet de contrôler l'intégrité des données en vérifiant qu'elles n'ont pas subi d'altération.
- L'ACK pour Acknowledgement composée de 2 bits qui permet la non-répudiation de la trame.
- EOF pour End Of Trame composée de 7 bits désignant le nombre de bits d'attente avant la nouvelle émission de trame.

Figure 15 : Décomposition d'une trame

(Wikipédia, Bus de données CAN, 2018)

Ainsi, des méthodes comme le CRC ou le Ack garantissent une certaine gestion des erreurs. Lorsqu'un système reçoit des données erronées, une trame d'erreur est automatiquement émise par ce nœud pour alerter les autres composants du réseau. Ainsi, l'émetteur de la trame erronée doit renvoyer cette même trame.

2.3 Exemple de fonctionnement

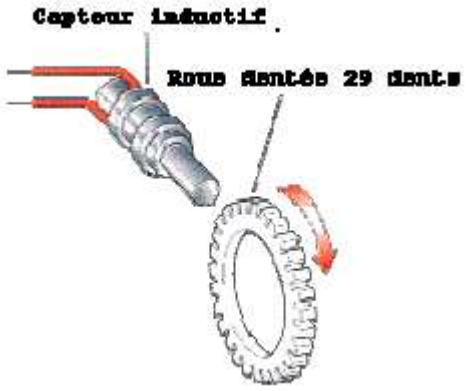
2.3.1 Système de freinage (ABS)

L'ABS (Anti Block System) est un système permettant de débloquer les roues lors d'un freinage d'urgence. Ce déblocage permet ainsi de diminuer la distance de glissement du véhicule et de rendre possible le contrôle directionnel du véhicule par le conducteur.

Ce système nécessite la présence de capteurs pour recueillir les informations, d'un calculateur pour les traiter et d'un actionneur pour mettre en marche le système d'antiblocage.

Quatre capteurs de mesure sont placés dans le véhicule, un au niveau de chaque roue. Ces capteurs aimantés permettent de s'accorder à une roue dentée fixée directement à la roue du véhicule. Une transmission de la vitesse de la roue dentée est donc émise par flux magnétique au capteur proportionnelle à la vitesse de la roue.

Figure 16 : Capteur de vitesse et roue dentée



(Claude Lahache, Principes de l'électronique automobile, 2009)

Ces capteurs émettent donc en continu au calculateur ABS la vitesse de rotation de chaque roue. Celui-ci a pour but de traiter les informations reçues afin de faire appel aux actionneurs lorsque le véhicule nécessite une intervention ABS.

Pour ce faire, le calculateur va constamment calculer, via son logiciel de traitement, le coefficient de glissement du véhicule qui est déterminer sur la base de la vitesse du véhicule et de la vitesse de la rotation des roues.

$$Coefficicent\ de\ glissement = \frac{\text{Vitesse du vehicule} - \text{Vitesse de rotation des roues}}{\text{Vitesse du vehicule}} * 100$$

(Wikipédia, 2018)

Un coefficient de glissement égal à 0% signifie une adhérence parfaite du véhicule contrairement à un coefficient égal à 100% qui signifierais un blocage complet des roues.

Lors du traitement du calculateur, un coefficient de glissement trouvé supérieur à 20% est considéré comme une perte d'adhérence et nécessite l'intervention du système ABS.

L'intervention est réalisée par des actionneurs qui ont pour but de réguler le freinage afin d'empêcher le blocage des roues. Ainsi, le calculateur envoie le signal à une valve régulatrice, nommée électrovanne, présente sur chaque roue et permettant de réguler les pressions de freinage.

Ce système d'électrovanne est associé au maître-cylindre qui représente la pièce de transmission des pressions de freinage au travers de canalisations hydrauliques. Enfin, elles communiquent également avec l'étrier de frein, par l'intermédiaire d'un cylindre récepteur, qui permet de réaliser l'action mécanique sur les plaquettes de frein.

L'électrovanne fonctionne en un cycle de trois phases.

La première consiste à laisser communiquer entre eux le maitre-cylindre et l'étrier de frein résultant d'un freinage « classique » où la pression de freinage reste en constante augmentation. Dans cette phase, le système ABS n'est pas en fonction.

Dès lors qu'une détection de blocage des roues est constatée, la communication entre le maitre-cylindre et l'étrier de frein est coupée faisant stagner la pression de freinage sur l'étrier. Ceci représente le durcissement de la pédale de frein et la constance de freinage.

Enfin, la dernière phase consiste à mettre en relation le cylindre récepteur lié a l'étrier avec une pompe de refoulement. Cette pompe a le rôle inverse du maitre-cylindre puisqu'elle permet la diminution de la pression de freinage.

Dès lors que les roues du véhicule se débloque, la première phase est rappelée, puis la seconde et enfin la dernière. Selon Robert Bosch, ces opérations sont effectuées plusieurs fois par seconde expliquant les vibrations de la pédale de frein lors du déclenchement du système ABS.

2.3.2 Système de détection de pluie

Contrairement à l'ABS, le système de détection de pluie est catégorisé comme un système de confort et doit être activé par le conducteur du véhicule afin de déclencher son fonctionnement.

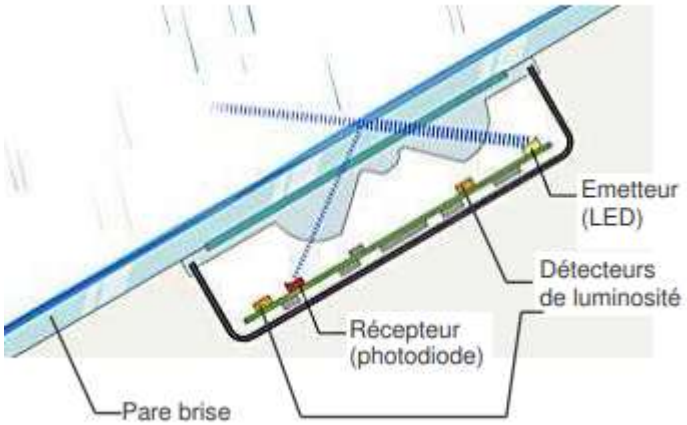
Encore une fois, ce système nécessite la présence d'un capteur afin de récolter les informations à traiter. Dans ce cas de figure, c'est un capteur de détection de pluie qui est utilisé. Il est composé de plusieurs éléments :

- Une ou plusieurs diodes électroluminescentes : Ces diodes permettent d'envoyer des signaux lumineux.
- Un prisme : Celui-ci permet la réflexion de la lumière.
- Une photodiode : Celui-ci réceptionne les signaux lumineux.

Un faisceau lumineux est constamment envoyé par la diode électroluminescente au pare-brise par l'intermédiaire d'un prisme de plexiglas. Ce prisme permet de réfléchir la lumière envoyée au pare-brise, lors d'une vitre sèche, ce signal lumineux est entièrement réfléchi et réceptionné par la photodiode.

Dès lors que des gouttes d'eau se déverse sur le pare-brise, la réflexion du prisme décroît et la quantité de luminosité reçue par la photodiode est inférieur à la quantité envoyée par la diode émettrice dues aux rayons lumineux diffractés par les gouttes.

Figure 17 : Représentation de la diffraction d'un signal lumineux



(Claude Lahache, Principes de l'électronique automobile, 2009)

Le capteur identifie la différence de luminosité envoyée et transforme cette donnée physique en un signal électrique. Ce signal envoyé en continu au calculateur d'habitacle par faisceaux électrique contient, non seulement, la présence ou non d'une diffraction lumineuse mais surtout de son degré.

Avec ces informations, le calculateur à habitacle peut transmettre au moteur à essuie-glace, représentant l'actionneur du système, la vitesse de balayage des essuie-glaces.

3. Réalisation d'un prototype

3.1 Contexte

Le but de ce prototype est de pouvoir simuler le fonctionnement d'un système embarqué par l'intermédiaire d'une mini voiture robotisée et d'une carte à puce programmable. Le principe serait de pouvoir ajouter des fonctionnalités supplémentaires à celles indispensables comme la simple conduite du véhicule.

Ainsi, un système de détection de collision s'ajouterais comme fonctionnalité de confort au mini robot par programmation via la carte à puce intégrée. Dès que le véhicule devra faire face à un obstacle, il devra par lui-même stopper son fonctionnement.

Enfin, ce robot doit pouvoir être piloter par l'intermédiaire d'une télécommande infrarouge permettant le contrôle directionnel du robot.

3.2 Outils utilisés

3.2.1 Outils physiques

Robot voiture

La voiture robot utilisée pour la réalisation de ce prototype provient de la marque Elegoo. Elle se présente sous la forme d'un kit à monter, comptant environ vingt modules différents dont principalement quatre roues, deux piles, quatre moteurs et une structure de base.

Figure 18 : Voiture robot assemblée

(Elegoo, 2018)

Carte Arduino Uno

Une carte programmable de marque Arduino et de type Arduino Uno est également présente sur le robot. Cette carte pourra contenir tout le traitement informatique permettant d'ajouter les fonctionnalités souhaitées au prototype. Cette carte dispose d'une sortie USB afin de la relier directement à son ordinateur de conception via un câble USB. Une sortie d'alimentation y est également présente afin d'alimenter la carte lors de la mise en marche du prototype. Enfin, de nombreux connecteurs (ou pins) représentant les entrées ou les sorties de données de la carte.

Figure 19 : Carte programmable Arduino Uno R3

(Go Tronic, 2018)

Carte pilote moteur

Cette carte imprimée de circuits intégrés permet d'actionner les moteurs du robot via l'envoi de signaux électriques, directement rattachée aux quatre moteurs. Cette carte est connectée et contrôlée par la carte Arduino.

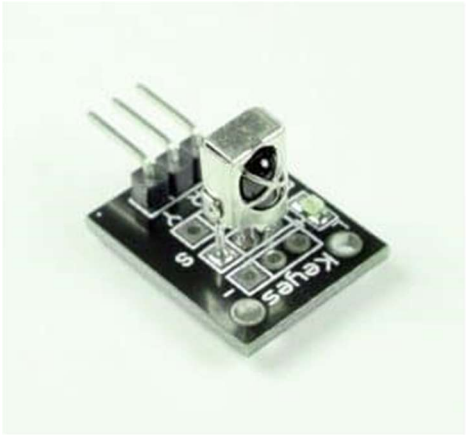
Figure 20 : Carte pilote moteur

(Elegoo, 2018)

Capteur infrarouge

Le capteur infrarouge permet de réceptionner les signaux émis par une télécommande infrarouge. Ce capteur est directement relié à la carte Arduino afin de traiter les signaux reçus.

Figure 21 : Capteur infrarouge



(Arduino, 2018)

Capteur d'ondes ultrasonores

Ce capteur envoie en continu des ondes sonores afin de détecter la présence d'un obstacle. Ainsi, le capteur envoie les données à la carte Arduino afin de traiter ces informations et de déterminer la distance de l'obstacle. Ce capteur se place à l'avant du prototype robot.

Figure 22 : Capteurs d'ultrason

(Arduino, 2018)

3.2.2 Outils informatiques

Pour le développement informatique des différents programmes présents dans ce prototype, c'est l'IDE (Integrated Developpement Environnement) Arduino qui a été utilisé. En effet, Arduino possède son propre environnement de programmation facilitant le développement sur carte programmable spécifique à Arduino. Cet IDE open-source développé en Java est complétement gratuit.

En plus d'un IDE, Arduino possède son propre langage de programmation applicable seulement sur les cartes de type Arduino. Ce langage est en réalité fortement inspiré du langage C++ et du C mais qui diffère dans l'utilisation des fonctions prédéfini, d'utilisation d'objets et de libraires Arduino, et un système de compilation adaptée a programmation électronique.

3.3 Fonctionnement

3.3.1 Généralités

Avant de passer aux explications des fonctionnalités du prototype, il est indispensable de présenter le fonctionnement de l'environnement Arduino et de ses interactions avec la carte.

Le logiciel de développement comprend un compilateur permettant de traduire le langage de programmation Arduino en langage machine. Lors de la compilation du code, le programme est prêt à être exécuté. Contrairement à une programmation classique, l'exécution du programme ne se fait pas via l'environnement de programmation mais directement sur la carte programmable.

En revanche, la possibilité de déverser le code compilé sur la carte programmable est possible via l'environnement de développement. Ainsi, toutes les instructions programmées sont envoyées au microprocesseur de la carte par l'intermédiaire du câble USB qui y est connecté. Ce programme est envoyé sous forme de signal électrique à la carte, qui devra être converti avant d'être réceptionné par le microcontrôleur.

Lors de l'alimentation de la carte, le code enregistré dans le microcontrôleur via sa mémoire flash est automatiquement exécuté et ce en continu. C'est la structure de l'architecture Arduino qui impose l'exécution des instructions en boucle.

Figure 23 : Structure des programmes Arduino

La fonction setup() constitue l'initialisation des données du programme, notamment pour définir les paramètres d'entrées et sorties de la carte. Elle ne renvoie rien et est exécutée une seule fois en tant que première instruction.

La fonction loop() va contenir le programme à exécuter et ceci en boucle de façon infinie jusqu'à que l'alimentation de la carte soit interrompue.

La création d'autres fonctions ou la déclaration de variable peut être réalisée en dehors de ces deux méthodes mais devront être appelées dans une de ces deux zones pour être traitées.

Le langage Arduino utilise une syntaxe similaire au C et C++, à la différence de constantes supplémentaires et de fonctions prédéfinies par l'intermédiaire de bibliothèques, adaptées selon les besoins électroniques.

3.3.2 Mise en marche du prototype

Pour la mise en marche du robot, il est indispensable de connaître la configuration de la communication entre la carte programmable et les moteurs des roues. Ainsi, la carte Arduino est reliée à la carte de pilotage des moteurs par l'intermédiaire de fils electriques.

Cette carte de pilotage est reliée directement à chacun des moteurs du robot par fils électriques. Lorsque la carte Arduino envoie des instructions de pilotage, elles sont réceptionnées par la carte de pilotage qui les traite et actionne les moteurs. Plus précisément, ce sont 6 sorties de fils électriques qui sont raccordées entre ces deux cartes ayant chacun un rôle et une information à délivrer.

Tableau 6 : Désignation des six sorties pour pilotage des roues

Intitulé	Désignation	Pin raccordé
ENA	Permet le contrôle vitesse des roues de droites	5
ENB	Permet le contrôle vitesse des roues de gauches	11
IN1	Permet la marche avant de la roue droite	6
IN2	Permet la marche arrière de la roue droite	7
IN3	Permet la marche arrière de la roue gauche	8
IN4	Permet la marche avant de la roue gauche	9

Les pins représentent l'emplacement de ces sorties sur la carte Arduino. Il est impératif de les initialiser afin que le programme sache sur quelles connectiques sont branchés les fils électriques.

Il est essentiel de préciser de quel type sont les pins, dans ce cas, les pins sont sortants puisque c'est la carte Arduino qui transmet l'information à la carte de pilotage.

C'est lors de l'initialisation des données de la carte que cette précision devra se faire, soit dans la fonction setup() par le biais de la fonction prédéfinie pinMode(int p, int p) comprenant deux paramètres : le premier correspond au numéro du pin assigner et le second un entier OUTPUT/INPUT (0 et 1), deux constantes également prédéfinies par Arduino.

Figure 24 : Extrait de code d'initialisation

```
/*Initailisation des valeur de pins*/
int in1 = 6;
int in2 = 7;
int in3 = 8;
int in4 = 9;
int ENA = 5;
int ENB = 11;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // Vitesse de debit de donnees
  // Declaration des pins de sortie
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
}
```

Une fois les initialisations réalisées, les instructions de commande de moteurs peuvent être programmées. Afin d'activer les différentes commandes précitées, un entier de type HIGH/LOW est proposé par Arduino. Cet entier représente plus précisément la valeur la tension électrique à transmettre, correspondant à 5V (valeur maximal) pour la valeur HIGH et 0V pour la valeur LOW. Ces valeurs sont utilisées par l'intermédiaire de la méthode digitalWrite(int p,int p) qui prend en compte deux paramètres : le pin sur lequel l'écriture doit être réalisée et la valeur de la tension à émettre.

Ainsi, l'utilisation de cette fonction peut être réalisée sur chacun des pins qui interviennent sur la mise en marche du véhicule en combinant la localisation des valeurs de tension à transmettre selon l'orientation du véhicule (marche avant, tourner à gauche...).

Figure 25 : Extrait de code pour une marche en avant

```
void enAvant()
{
  digitalWrite(ENA,HIGH);
  digitalWrite(ENB,HIGH);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
}
```

Pour faire avancer le véhicule, il est nécessaire d'actionner les deux moteurs à une tension de 5V et d'actionner les moteurs de marche avant gauche et droite également à 5V. Cette fonction doit être appelée dans la fonction loop() qui exécutera ces instructions tant que la carte programmable est alimentée.

Chaque possibilité d'orientation du véhicule se réfère à une fonction contenant les instructions nécessaires au fonctionnement du prototype. Dans le cas d'un changement de direction à gauche, il suffirait d'actionner seulement le moteur de marche avant droit et le moteur de marche arrière gauche afin de basculer sur la gauche. (aGauche()).

Concernant les sorties ENA et ENB, leur emplacement se situe sur des pins spéciaux supportant l'écriture analogique contrairement aux autres sorties précitées. En effet, ce type d'écriture permet de préciser la tension à transmettre et ainsi contrôler la vitesse des moteurs contrairement aux autres qui se limite à un choix binaire.

C'est donc avec la fonction analogWrite(int p, int p) sur le même principe de fonctionnement que le digital mais pourra prendre en second paramètre une valeur entre 0 et 255, cette dernière correspond à un maximum de 5V.

Enfin, une fonction permettant de rythmer le fonctionnement du robot est proposée par Arduino. C'est la fonction delay(int ms) prenant en paramètre le durée en milliseconde que l'utilisateur peut entrer pour mettre en attente le programme (et surtout la boucle).

3.3.3 Guidage par télécommande infrarouge

Jusqu'à présent, rien n'a pu indiquer au véhicule une action de s'arrêter, de reculer, de freiner ou de tourner à un instant souhaité par l'utilisateur. Autrement dit, aucune action directe de la part de l'utilisateur a pu faire appel aux fonctions d'orientations. Seul, la méthode delay() peut permettre l'arrêt provisoire du traitement pour faire appel à une autre fonction.

Un système de guidage peut donc être proposé par l'intermédiaire d'une télécommande à infrarouge et d'un capteur infrarouge qui viennent s'ajouter aux éléments précités. Ce principe repose sur le fait que lorsqu'une pression sur une des touches de la télécommande est réalisée, un signal infrarouge est émis à courte distance. Le capteur peut ainsi réceptionner le signal émis et en fonction de la touche pressée, réaliser le traitement souhaité.

Plus précisément, la télécommande envoie une série d'impulsions infrarouges codées en binaire. C'est ce signal que reçoit le capteur, mais afin de ne pas interférer avec d'autres signaux, ce signal est modulé par une fréquence en utilisant un protocole précis.

Dans le cas de notre télécommande et capteur, c'est le protocole NEC qui est utilisé et qui est caractérisé par l'envoi de 16 bits à deux reprises pour garantir la fiabilité de l'échange. De plus, la fréquence utilisée pour l'échange est de 38kHz permettant de réceptionner les signaux de cette fréquence précise.

Ces signaux sont décodés par le capteur afin de savoir quelle touche a été pressée par l'utilisateur. C'est avec l'utilisation des fonctions de typage de la libraire « IRemote.h » que tout ce processus de décodage pourra être réalisé.

Une connexion entre le capteur et la carte Arduino est établie par un fil électrique qui permettra à la carte de recevoir les données. Ainsi, le pin 12 a été utilisé comme pin d'entrée pour recevoir les données du capteur. Une déclaration de variable de type IRrecv prenant en paramètre le numéro de pin utilisé afin de déclarer l'arrivée d'une donnée de type infrarouge sur ce pin. Exemple : IRrecv donneeInfra(pin12) ;

Une variable de type decode_results est également déclarée afin de décoder la valeur infrarouge en un entier afin de pouvoir connaître la touche pressée par l'utilisateur. Chaque touche de la télécommande est identifiée par un code unique fourni.

Tableau 7 : Valeurs des touches de télécommande

Boutons télécommande	**Valeur associé**
Bouton Ok - Stop	16712445
Bouton haut – En avant	16736925
Bouton bas – En arrière	16754775
Bouton gauche	16720605
Bouton droite	16761405

(Elegoo, 2018)

Toutes ces valeurs sont déclarées en tant que constante afin de pouvoir faire appel aux bonnes instructions en fonction de ces valeurs. C'est précisément la fonction .value de l'objet de type decode_results qui retourne la valeur du bouton pressée par l'utilisateur.

Figure 26 : Extrait du traitement des signaux infrarouge

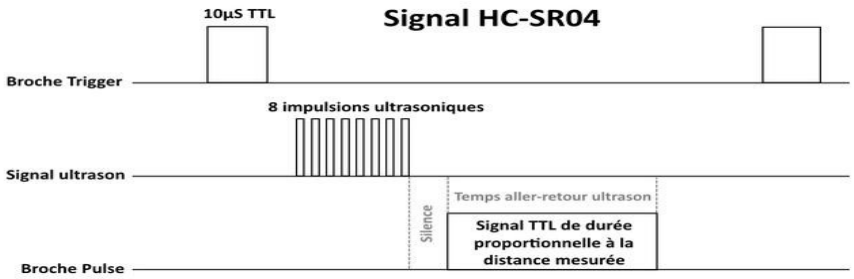
```
void loop() {
  if (irrecv.decode(&results)){ // si le signal a bien ete recu et decoder
    val = results.value; // recuperation de la valeur du bouton
    irrecv.resume(); // Reception du prochain code
    if(val == AVANT) {
      enAvant();
    } else if(val == ARRIERE) {
      enArriere();
    } else if(val == GAUCHE) {
      aGauche();
    } else if(val == DROITE) {
      aDroite();
    } else if (val == STOP){
      enStop();
    }
  } else {
    enStop();
  }
}
```

3.3.4 Détection d'obstacle

La détection d'obstacle se fait principalement par le capteur d'ondes sonores. Comme le précédent capteur, celui-ci transmet des informations directes à la carte Arduino et par l'intermédiaire de fils électriques. Ce capteur permet de mesurer les distances d'un obstacle qui se trouve dans son champ d'émission, soit un angle de 15° et une distance pouvant aller jusqu'à 400mm. (Cytron Technologies Sdn. Bhd, User's Manual – HC-SR04 Ultrasonic Sensor, 2013)

Pour ce faire, la carte Arduino envoie une impulsion de 5V d'au moins 10 microsecondes sur l'entrée « Trig » du capteur, c'est le moyen nécessaire pour que le capteur émet une série de 8 impulsions ultrasoniques de 40 kHz (inaudible à échelle humaine). En cas de réfléchissement, le capteur émet à son tour un signal de 5V à la carte Arduino via son entrée « Echo ».

Figure 27 : Représentation du signal ultrason



(Batteix Fabien, 2016)

Cette avec la durée du temps d'intervalle entre l'envoi de l'impulsion ultrasonique et la réception du signal Echo (Pulse) que la distance pourra être déduite (grâce également à la vitesse du son).

C'est la fonction pulseIn(int pin, int tension) qui prend en paramètre le numéro du pin ciblé (Echo) et la valeur HIGH pour la tension qui retourne le temps d'intervalle. Cependant, la durée retournée correspond à l'intervalle de temps entre l'envoi du signal Trig et la réception du signal Echo, il est donc nécessaire de diviser par 2 la valeur réceptionnée (c'est un aller-retour, or nous sommes intéressés que par le retour).

Voici donc la formule pour transformer le temps de la requête en distance réelle :

Distance (en mm) = Temps de réponse / 2 * Vitesse du son

La vitesse du son est de 340m/s. Afin d'obtenir un résultat en millimètres, il est préférable de convertir cette vitesse : 340m/s -> 340mm/us-> 340/1000 mm -> 0.34 mm

Le manuel d'utilisateur du capteur propose une formule avec un résultat directement en centimètres en se basant sur la précédente formule après plusieurs simplifications et conversions.

Distance (en cm) = Temps de réponse / 58

(Cytron Technologies Sdn. Bhd, User's Manual – HC-SR04 Ultrasonic Sensor, 2013)

C'est cette formule qui a été utilisée pour le traitement du temps de réponse et ainsi pouvoir déduire la distance de l'obstacle.

Figure 28 : Extrait de code fonction de calcul de distance

```
int Distance_obstacle()
{
  digitalWrite(Trig, LOW); // Initialise le trriger a 0 avant lancement
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH); // Lancement du trriger sur impulsion de 10 microsecondes
  delayMicroseconds(10);
  digitalWrite(Trig, LOW); //Apres 10 microsecondes on arrete
  float distance = pulseIn(Echo, HIGH); // Retour
  distance= distance/58;
  return (int)distance; //Conversion du float en entier
}
```

Cette fonction retourne un entier avec une unité de mesure en centimètre. Cette fonction doit être appelée dans la boucle loop() afin de connaître en continu la distance d'un éventuel obstacle. Des instructions peuvent être déclenchées selon la valeur de cette fonction. Exemple : Stopper le robot si un obstacle se trouve à 20cm.

Figure 29 : Extrait code de traitement distance d'obstacle

```
void loop() {
  distanceStop = Distance_obstacle(); // Appel fonction de distance
  if (distanceStop<=DISTANCE_MAX){ // Distance maximum 20cm pour traitement de l'obstacle
    enStop();
    delay(500); //Temps d'attente avant de passer a la prochaine instruction
    enArriere();
    delay(300);
    enStop();
    delay(1000);
  }else {
    if (irrecv.decode(&results)){
      val = results.value;
      irrecv.resume();
      if(val == AVANT) {
        enAvant();
      } else if(val == ARRIERE) {
        enArriere();
      } else if(val == GAUCHE) {
        aGauche();
      } else if(val == DROITE) {
        aDroite();
      } else if (val == STOP){
        enStop();
      }
    }
  }
}
```

Dès que le prototype rencontre un obstacle à 20 cm ou moins, une instruction de se stopper est lancée, suivie d'une courte marche arrière et d'une nouvelle mise en arrêt permettant à l'utilisateur de se rediriger via sa télécommande.

3.4 Déroulement du projet

Concernant le déroulement du projet, il a été indispensable pour moi de me familiariser avec les termes et les principes de bases de l'électronique pour la réalisation d'un projet alliant informatique et électronique.

Cette introduction m'a permis de mieux cibler les choix concernant les outils qui allait être utilisés pour la réalisation de ce prototype et ainsi me diriger vers des composants robuste, fiable et pouvant proposer des services dont ils m'étaient nécessaire.

Mes choix se sont donc portés vers Arduino pour l'environnement informatique et sa carte programmable complète pour la réalisation de ce projet. Arduino est présenté comme l'un des leaders des cartes programmables et connait une présence conséquente sur la communauté web avec notamment une plateforme de support de qualité.

Concernant le choix de la robot voiture, il a été plus compliqué de garantir l'achat de composants fiables et de qualités malgré des recherches approfondies sur le web. En effet, l'achat de ces composants se fait principalement via le web par des fournisseurs chinois garantissant une fiabilité incertaine. Elegoo s'est démarqué par ses nombreux retours clientèle positifs et une importante communauté d'utilisateurs sur le web.

Le produit est fourni avec des instructions pour la conception manuelle du robot qui s'est présentée sous forme de kit à monter. Grâce à un mode d'emploi complet, le montage a pu se faire sans complications. Concernant, l'application de l'environnement Arduino sur le robot Elegoo, des exemples basiques sont proposés afin de se familiariser avec le prototype.

Cependant un des problèmes majeurs rencontrés dans ce projet a été l'alimentation très sensible du robot. En effet, de fils électriques sont reliés depuis la carte Arduino à la batterie du prototype et d'autres reliés aux moteurs des roues. C'est cette dernière alimentation qui a été problématique puisque malgré le branchement correct des fils, le courant électrique ne pouvait circuler à cause de l'emplacement pas assez précis des fils électriques. Malgré une alimentation correcte vers la batterie mais mauvaise vers les moteurs, aucun traitement n'a pu être fonctionnel malgré des instructions correctes.

S'ajoute comme problèmes rencontrés, mais hors du contexte du projet, un servo moteur défaillant de par sa conception afin de faire pivoter à différents degrés le capteur d'ondes ultrasonique.

Au niveau développement, il a été assez complexe de comprendre précisément le fonctionnement des capteurs résultant du domaine de l'électronique. De nombreux tests pratiques et comparaisons m'ont permis de cibler le fonctionnement de ces capteurs.

Enfin, Arduino reste un langage très accessible par sa facilité de compréhension et sa simplicité de codage grâce notamment à un environnement de qualité. Une fois que les composants comme les actionneurs ou les capteurs ont été compris de leur fonctionnement, le développement du prototype peut être réalisé sans complications majeures.

4. Les risques de l'embarqué

4.1 Contexte

Il est important de spécifier que cette partie abordera les risques et les menaces des systèmes embarqués mais pas des solutions et des contremesures mises en place pour les éviter.

L'évolution technologique des systèmes embarqués dans l'automobile rendent ces systèmes de plus en plus vulnérables à toute menace pouvant les rendre inexploitables mais surtout dangereux. S'ajoute à cela la forte contrainte du temps réel qui nécessite une obligation de résultat pour certains systèmes.

Ainsi la complexité de ces systèmes les expose à des menaces de tous genre, principalement des attaques virales, des intrusions malveillantes ou encore le contrôle total du système par autrui. Ces risques auraient un impact direct sur la confidentialité, l'authenticité et la disponibilité des données du système.

Nous pouvons également rencontrer des menaces internes qui serait directement reliées à une défaillance du système en lui-même que ce soit au niveau matériel ou logiciel. Malgré la mise en place de normes tel que ISO 26262 [ISO11] afin de garantir une sécurité des systèmes automobiles, le risque de voir apparaître la menace existe toujours. Des lignes de codes erronées aura au mieux un impact sur la qualité du produit, au pire sur la sécurité directe de la personne qui l'utilise.

Que ce soit des risques internes ou externes, les impacts pourraient avoir de très lourdes conséquences. Les répercussions d'une défaillance du système de climatisation ne seront pas les mêmes que dans le système de freinage ou encore le système d'airbag.

Ainsi, les systèmes embarqués doivent garantir la sureté de fonctionnement à ses utilisateurs afin de préserver principalement ces quatre qualités non fonctionnelles :

- La fiabilité : C'est le fait de s'assurer que le système pourra toujours être fonctionnel après une certaine durée déterminée.
- La disponibilité : C'est le fait de pouvoir à tout instant fournir un service demandé.
- La maintenabilité : C'est le fait de pouvoir remettre en état de marche ou améliorer un système déjà utilisé.
- La sécurité : Ce point rejoint tous les précédents, c'est le plus important puisque c'est celui-ci qui s'assure qu'aucun accidents graves puisse survenir.

La sureté de fonctionnement n'est pas exclusive au domaine de l'automobile, il peut et doit être pris en compte pour tous les systèmes exerçant dans un environnement jugé critique.

4.2 Menaces interne

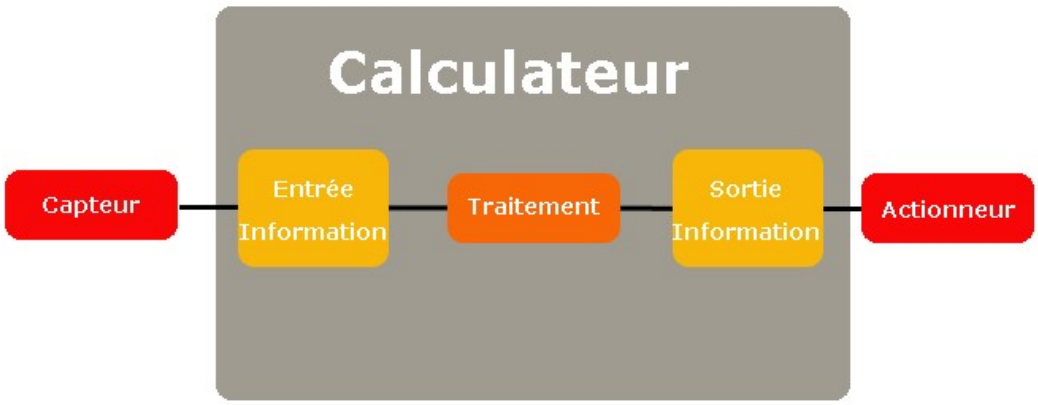
Les menaces internes regroupent ainsi toutes les défaillances que peut générer un système due à sa mauvaise conception ou maintenabilité. Nous allons ainsi nous consacrer sur la sureté de fonctionnement d'un système que ce soit au niveau logiciel ou matériel.

4.2.1 Hardware

Aujourd'hui, la durée de vie d'un véhicule dépasse les vingt années, ainsi les composants électroniques doivent s'adapter à cette durabilité d'existence malgré un environnement soumis à des contraintes de climat très divergents, de chocs et secousses ou encore de température extrême. Le matériel utilisé à la fabrication de ces composants doit ainsi pouvoir répondre à toutes ces contraintes afin de garantir une certaine fiabilité sans quoi le risque de court-circuit peut surgir.

Nous avons pu constater que l'électronique embarqué est composé d'une chaîne de composants communiquant entre eux.

Figure 30 : Chaine des composants électroniques



(Ruellet Laurent, Le multiplexage, 2002)

Ce sont tous ces composants qui sont susceptibles de subir une défaillance technique, et non pas seulement le calculateur qui est le cœur du système. Un capteur en panne empêchera toute entrée d'information ou rendra l'information non intègre ce qui provoquera un traitement impossible de la part du calculateur. De même pour les

actionneurs, si celui-ci ne peut déclencher l'action qui résulte du traitement du calculateur, c'est le processus entier qui est stoppé et le résultat escompté ne pourra parvenir.

Ce sont d'ailleurs plus fréquemment ces actionneurs et capteurs qui sont victimes de pannes dû principalement à leur positionnement dans l'habitacle qui les rendent plus exposés aux contraintes du véhicule contrairement à un calculateur qui est plus « enfoui » avec un positionnement stratégique afin de subir au minimum ces mêmes contraintes.

Tous ces composants communiquent entre eux par moyen de faisceaux électronique permettant l'envoi et la réception d'information. Ce même système peut communiquer avec d'autres systèmes via le bus CAN qui lui-même se compose également de câbles et de son système de centralisation des données. Ainsi, tous ces composants de communication représentent un risque puisque seul un élément suffit à mettre tout le système électronique en panne.

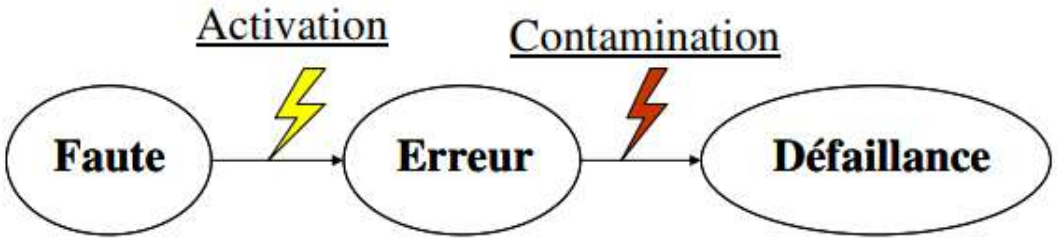
Il est donc du devoir du concepteur et fabricant de respecter toutes les normes et règles afin de garantir des composants irréprochables dans un environnement de système à temps réel critique. Des systèmes comme l'ABS, le régulateur de vitesse ou encore d'airbag se doivent de respecter une conception irréprochable dû à leur degré de responsabilité.

4.2.2 Software

La conception logicielle est tout aussi importante que la conception électronique dû à leur risque d'impact similaire sur le produit en cas de défaillance. C'est la partie « traitement » du calculateur qui est donc à élaborer avec précision puisqu'il ne doit en aucun cas présenter de faille. C'est devenu aujourd'hui la partie la plus critique du système due à la complexité des programmes pouvant présenter de nombreux « bugs » non maitrisés par les développeurs.

Une faille peut surgir à la suite d'une erreur qui est provoquée par une faute dans le système. La faille est donc la conséquence d'une faute, qui est visible de la part de l'utilisateur et rend le système inexploitable.

Figure 31 : Etape d'une défaillance système



(M. Portolan, Conception d'un Système Embarque Sur et Sécurisé. Micro et nanotechnologies/Microélectronique, Institut National Polytechnique de Grenoble - INPG, 2006)

Une faute ne génère pas d'erreur tant que les opérations comprenant cette faute ne sont pas appelées, lorsque le contraire se passe, l'activation de la faute se produit et génère donc une erreur. Bien entendu, un composant communique avec d'autres, échanges des informations et utilise le résultat de l'un pour les opérations d'un autre, ainsi un composant avec erreur peut transmettre des données erronées à un composant nécessitant son résultat. Ceci est la contamination du système puisque tous les composants seront en possession de données erronées provoquant inévitablement une défaillance.

La faute peut et doit être évitée lors de la conception du système par le biais de développement spécialisé en utilisant des environnements de programmation adapté. L'utilisation de système d'exploitation en temps réel est indispensable pour la réalisation d'un système de ce genre. Ce type de système d'exploitation ne garantit pas un résultat final mais fournit des services et outils facilitant le développement afin d'obtenir un comportement en temps réel.

C'est donc avec des tests d'utilisations et des comparaisons de résultats qu'une vérification des attentes du système pourra être faites évitant ainsi des fautes de conception.

Cependant des fautes peuvent surgir lors de la mise en production du système, lorsqu'il est exécuté, non due à une mauvaise conception, mais plutôt aux contraintes de son environnement tels que des interférences. Ainsi, leurs prises en charges doivent être réalisées lors de l'exécution du système par des solutions anticipées lors de la conception tels que la réplication ou la technique des points de reprise.

Ces solutions dépendent de la criticité du type de défaillance rencontré, comme déjà énoncé, certaines défaillances peuvent avoir des répercussions très grave pour l'utilisateur, principalement lorsque les composants touchés sont ceux relatifs à la

sécurité. Parfois, il est suffisant de savoir pour le système si un bug existe ou non mais dans d'autre cas (pour les plus critiques) une solution alternative doit être trouvée afin d'éviter la défaillance et de garantir la sureté de fonctionnement.

4.3 Menace externe

Aujourd'hui, les véhicules utilisent de plus en plus de systèmes embarqués avec une forte interconnexion les rendant vulnérables à tous types d'attaques malveillantes. Des outils comme le Wi-Fi, le Bluetooth, la géolocalisation ou encore le Keyless (clé sans contact) sont portes d'entrées à tous types de piratages informatiques.

Toutefois d'autres moyens moins complexes que le piratage informatique existe afin de porter atteinte au système.

Des personnes malintentionnées ayant chacune leurs propres motivations peuvent nuire à ces systèmes pouvant ainsi portées atteintes à la confidentialité des données de ces systèmes, à la performance du produit utilisé mais surtout à la sécurité de son utilisateur.

4.3.1 Attaquants et objectifs

Il est important de connaître le profil de ces personnes malveillantes et surtout leurs intentions afin de mieux comprendre les raisons de ces attaques. Un discernement précis permettra particulièrement de mettre en place des systèmes visant à contrer ces attaques. Toutes informations relatives aux attaques sont indispensables aux mesures de sécurité élaborées.

Nous retrouvons principalement des hackers ayant des connaissances poussées en informatique leurs permettant de s'introduire dans ces systèmes, principalement à distance en portant atteinte directement au logiciel du système. Les raisons peuvent être nombreuses, mais le vol de données reste la principale cause des hackers en plus de vouloir prouver que ces systèmes restent très vulnérables. Il peut être vu comme un défi de la part d'un hacker de pirater un système robuste, en prenant par exemple son contrôle ou en le sabotant.

Les concurrents restent encore aujourd'hui des potentiels attaquants aux systèmes dans un but d'analyser ou de voler des données portant directement atteinte à la confidentialité des données du système. Toutefois, un simple délinquant peut être considéré comme potentiel attaquant dans un but de dérober directement le produit lui-même en plus de son système.

Les principales motivations d'attaquer un système sont donc de pouvoir accéder à des données privées ou secrètes afin de les modifier, de les détruire ou encore de les exploiter afin de déduire d'autres informations inaccessibles.

Plus précisément dans l'automobile, ce sont les risques de vol de voiture, prise de contrôle du véhicule ou de ses quelques fonctions, vol de données utilisateurs, vol de données de conception qui ont été recensés.

4.3.2 Types d'attaques

Evidemment les systèmes embarqués ont différents moyens d'être attaqués allant des plus basiques au plus complexes, ayant chacune leurs spécificités visant différents buts à atteindre.

Attaques matérielles

Les attaques matérielles visent tous les composants du système embarqué ou ceux susceptibles de communiquer avec dans le but de le détruire, de l'abîmer ou de le saboter. Nous pouvons identifier deux types d'attaques matérielles :

- Intrusive : C'est-à-dire des attaques matérielles ayant un fort impact sur le fonctionnement du système pouvant aller jusqu'à sa destruction.
- Non intrusive : C'est-à-dire des attaques matérielles qui n'ont laissées aucunes traces visibles sur le système et est resté fonctionnel.

Le découpage physique d'un circuit intégré dans un système représente clairement une attaque matérielle intrusive puisque le composant serait complètement abîmé rendant impossible son utilisation, ou bien un composant comme le bus CAN rendrait tout l'environnement inutilisable. Cette attaque nécessite une connaissance précise de l'architecture de l'environnement mais également de l'architecture du système.

Il existe également les injections de fautes qui ont des conséquences très puissantes sur le système, elles consistent à introduire des modifications physiques de l'environnement de la carte par le biais d'envois lumineux puissants, fortes impulsions électriques ou magnétiques dans le but de modifier le contenu de la partie logicielle du système. Ainsi, des répercussions surviendront lors du traitement du système sans que le composant n'ait été abimée justifiant son classement d'attaque non intrusive.

Attaques par canaux cachés

On peut classer ce type d'attaque comme indirect puisqu'elles n'auront aucun impact direct sur le composant en lui-même. Elles permettent l'observation du système pour

ensuite en faire des analyses statistiques. Les éléments observés seront principalement :

- Analyse du temps d'exécution
- Analyse du courant
- Analyse de l'émission électromagnétique

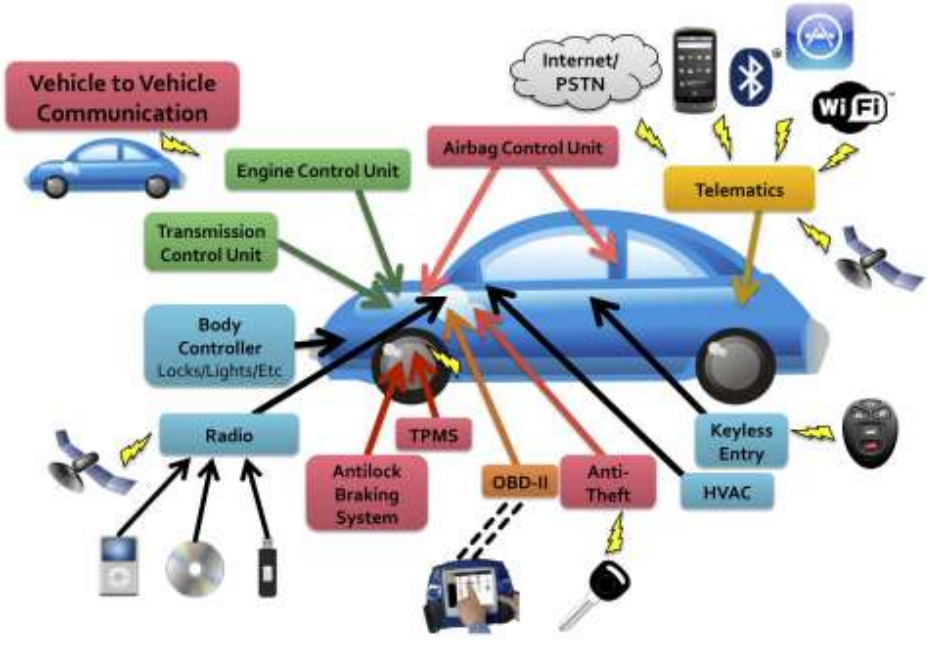
Ces techniques sont considérées comme des attaques malgré leurs rôles d'analyses puisqu'une intrusion au système est bien recensée, d'autant plus qu'elles seront accompagnées d'attaques matérielles ou logicielles.

Attaques logicielles

Les attaques logicielles dont le but principal est de prendre contrôle de l'entièreté ou de seulement une partie du réseau du véhicule grâce notamment à l'utilisation des ECUs présents en y injectant du code malveillant ou l'émission de trames erronées dans le réseau.

Pour ce faire, l'attaque devra interagir avec un point d'entrée au véhicule afin d'accéder à son réseau. Aujourd'hui la connectivité des véhicules est de plus en plus croissante impliquant une plus grande surface de vulnérabilité et ainsi une augmentation de ces points d'entrées à des intrusions malveillantes.

Figure 32 : Canaux de communication des véhicules actuels



(Dan Wallach, The truth about cars, 2011)

Des éléments comme la radio, le Wi-Fi ou encore le port OBD-II (port de connexion pour accéder aux informations du véhicule, utilisé comme outil de diagnostic) peuvent représentés des portes d'entrées à une éventuelle attaque. De plus, il a été relevé que de par sa conception, le bus CAN présenterait des failles laissant la possibilité aux attaquants du système d'accéder au réseau via les différents points d'entrées. En effet, cette structure de communication ne garantit aucune confidentialité des données puisque le simple fait de s'y connecter par le port OBD-II permet la lecture des données en claire.

Avant de se concentrer sur les points d'entrées, il est important de spécifier que généralement les attaques logicielles nécessitent une connaissance du réseau ou du système. C'est donc la raison du fort lien avec les attaques par canaux cachés qui ont ce rôle d'analyse de données afin de pouvoir cibler les failles du système et dans cet environnement, le bus CAN en est une parfaite illustration.

Ainsi, les nombreux moyens de communication présents aujourd'hui dans les véhicules modernes font office de premier passage à une attaque informatique.

Le Bluetooth d'une voiture peut représenter une de ces portes d'entrées notamment lors de la connexion de l'appareil téléphonique du conducteur. En effet, une simple connexion avec un appareil infecté d'un virus tel que cheval de Troie pourrait déclencher une attaque à son encontre et s'emparer de l'ECU en question.

L'ouverture à distance est maintenant présente sur toutes les voitures récentes, or ce système peut présenter une faille de sécurité. Malgré le chiffrement des signaux envoyés, des méthodes élaborées existent pour en déceler le chiffrement. Le chiffrement Keeloq est un exemple de code tournant qui consiste à enregistrer le signal émis par le la clé et de le rejouer plus tard pour le déverrouillage du véhicule. Cependant, ce signal peut être brouillé et intercepté par un autre boitier programmable pour être transmis au véhicule et ainsi le déverrouiller à l'insu de son propriétaire.

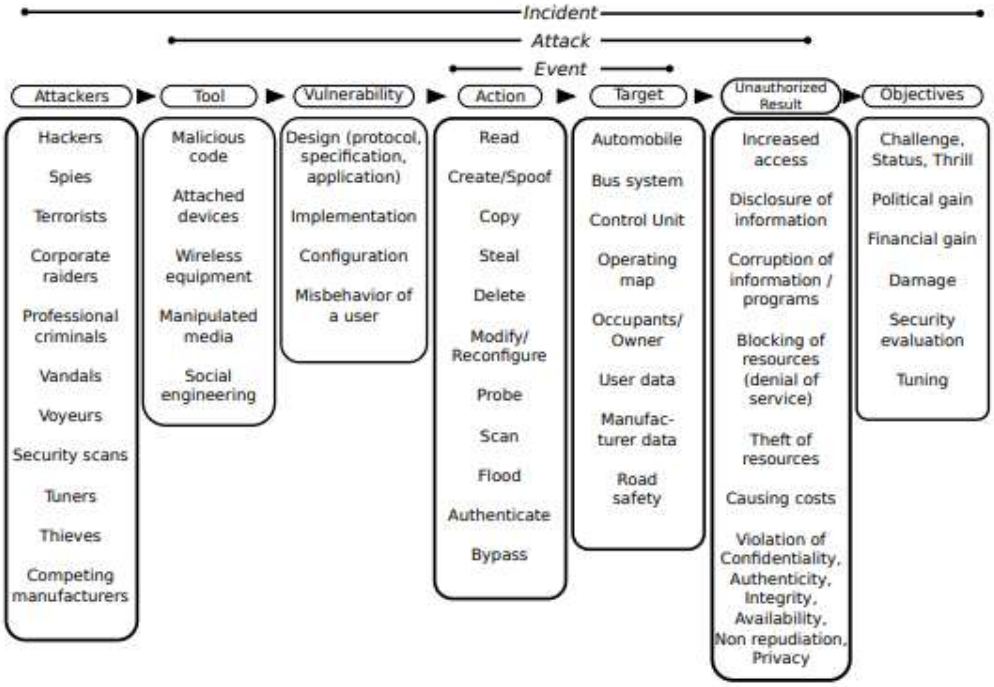
Dans le cas d'une voiture récente où un navigateur web y serait installé, des vulnérabilités pourrait être exposées au véhicule au même titre qu'un navigateur web dans un ordinateur classique via des virus, des vers ou autres types d'infections présentes sur le web pouvant ainsi donner accès aux données du système et potentiellement à son bus de communication.

Bien entendu, d'autres types de communication peuvent représenter des failles tels que le Wi-Fi, le port OBD-II, la prise USB ou même le lecteur-CD. Dès lors qu'une faille est exploitée par un attaquant, une multitude de possibilités sont offertes à lui afin d'accéder

à l'entièreté des systèmes et d'y réaliser au mieux de la lecture de données ou au pire des actions malveillantes de prise de contrôle.

Du fait des nombreux types d'attaquants, des différents moyens d'attaques, des diverses motivations de chacun, une multitude de combinaisons d'attaques sont réalisables sur les véhicules d'aujourd'hui (et du passé). Voici un modèle illustrant la quantité de scénarios possibles :

Figures 33 : Combinaisons d'attaques sur un système automobile



(Ivan Stundia, Détection d'intrusion pour des réseaux embarqués automobiles, 2015)

Bien que toutes ces menaces ne se soient pas toutes produites à ce jour, leur potentiel de survenir demeure existant.

Des démonstrations d'insécurités de véhicules ont été illustrées notamment lors de la prise de contrôle à distance des véhicules Tesla par des chercheures chinois en sécurité informatique. Ils s'avèrent que c'est à travers le Wi-Fi et le navigateur web que cette prise de contrôle a pu être réalisée en commandant des fonctionnalités tels que les essuies glaces, les vitres électriques ou encore l'ouverture du toit et ce pendant un véhicule en marche. Le constructeur a reconnu la faille et a pu la corriger malgré la mise en circulation du véhicule.

4.3.3 Conséquence d'une attaque

Une attaque contre un système classique aura bien sûr des effets néfastes sur celui-ci pouvant aller à sa destruction. Toutefois, une perturbation d'un système embarqué peut entraîner des répercussions non seulement sur le système mais surtout à son utilisateur dus aux fortes contraintes du système. Une attaque se conclut donc par la réalisation de l'objectif de l'attaquant au prix des nombreuses répercussions négatives incluant la mise en danger des utilisateurs.

Nous retrouvons deux catégories distinctes de conséquences :

- Fonctionnelles : Celle-ci décrit les effets néfastes sur les fonctionnalités du système, celles que l'attaquant a manipulée ou détériorée.
- Structurelles : Celles-ci représentent les conséquences directement liées à l'environnement du système, soit au véhicule. Ces conséquences ne sont pas liées aux objectifs de l'attaquant mais aux effets de bord qui ont été produit. Des impacts sur l'environnement incluent donc la mise en danger du conducteur.

Ainsi, les conséquences d'une attaque réussie varient selon le degré de gravité de celles-ci. Cependant des effets négatifs sont garantis à l'image de marque du véhicule, l'investissement financier de la part du propriétaire et dans les cas les plus graves : la mise en danger de sa propre vie.

5. Conclusion

Les systèmes embarqués sont donc des outils très puissants pouvant proposer une multitude de fonctionnalités et réaliser une quantité incalculable de tâches. Cependant, ces systèmes ne peuvent garantir une fiabilité de fonctionnement et une sécurité maximale à ses utilisateurs due à sa complexité de conception et son environnement constamment instable.

Ce travail démontre l'appartenance et le rattachement des systèmes embarqués dans le domaine de l'automobile avec une flagrante augmentation de leurs utilisations dans différents domaines, que ce soit la sécurité, le confort ou directement le fonctionnement du véhicule.

La réalisation du prototype a pu refléter la complexité de conception de ces systèmes mais également l'apport de réels avantages et bénéfices au sein de leur environnement. L'intégration de l'informatique dans l'électronique est également un constat à notifier, cela a permis à la création de fonctionnalités innovantes et utiles pour le domaine de l'automobile.

Ces systèmes apportent un niveau de sécurité supplémentaire à son utilisateur par l'intermédiaire de fonctionnalités d'aides et d'avertissements mais peuvent malheureusement représenter un danger lorsqu'ils défaillent.

Enfin, l'évolution du développement de ces systèmes n'a pas fini d'accroître avec l'exploration totale des limites de ces systèmes et l'élaboration de nouveaux afin de répondre à de nouveaux besoins.

Pour ma part, les systèmes embarqués restent indispensables dans le quotidien de chacun et bénéfique même a la société. Cependant, si des évolutions sont réalisées en termes de fonctionnalités et performances, elles doivent l'être également aux niveaux de la sécurité dû au trop grand nombre de faille encore présentes dans ces systèmes.

Bibliographie

[1] « Unité de commande électronique », *Wikipédia*. 04-avr-2017.

[2] « Sécurité informatique des véhicules », *Wikipédia*. 28-sept-2018.

[3] « Apollo Guidance Computer », *Wikipedia*. 15-sept-2018.

[4] « D-17B », *Wikipedia*. 29-août-2018.

[5] « Intel 4004 », *Wikipédia*. 11-août-2018.

[6] « Sécurité des systèmes cyber-physiques », *Wikipédia*. 03-août-2018.

[7] « Electronic control unit », *Wikipedia*. 10-juin-2018.

[8] « Système embarqué », *Wikipédia*. 06-avr-2018.

[9] « Sûreté de fonctionnement », *Wikipédia*. 01-oct-2017.

[10] « Code tournant », *Wikipédia*. 01-juill-2017.

[11] I. & Technologies, « Les systèmes embarqués dans notre quotidien : la révolution est en marche ! », *Ind. Technol.*, déc. 2013.

[12] I. & Technologies, « Systèmes embarqués ce qu'on attend du logiciel », *Ind. Technol.*, janv. 2004.

[13] I. Studnia, « Détection d'intrusion pour des réseaux embarqués automobiles: une approche orientée langage », p. 147.

[14] I. Studnia, « Détection d'intrusion pour des réseaux embarqués automobiles: une approche orientée langage », p. 147.

[15] A. C. Noubissi, A. A.-K. Séré, J. Iguchi-Cartigny, J.-L. Lanet, G. Bouffard, et J. Boutet, « Cartes à puce : Attaques et contremesures. », p. 8.

[16] E. Messerli, « Le système comprend une partie matériel et une partie logicielle », p. 13, 2018.

[17] G. Duc, « SR2I301 : Sécurité des Systèmes Embarqués - Introduction », p. 42.

[18] « Le multiplexage en automobile qu'est ce c'est? », *Panne-automobile.com*, 12-oct-2016. .

[19] retroetgeek13, « Arduino tuto capteur ultrason HC-SR04 », *RetroEtGeek*. .

[20] « Zoom : Calculateurs de moteur pour voiture », *Ooreka.fr*. [En ligne]. Disponible sur: https://entretien-voiture.ooreka.fr/astuce/voir/473457/calculateurs-de-moteur-pour-voiture. [Consulté le: 03-oct-2018].

[21] « Système embarqué : Définition ». [En ligne]. Disponible sur: https://www.ukonline.be/cours/embeddedsystems/programming/chapitre1-1. [Consulté le: 14-sept-2018].

[22] « Qu'est-ce qu'un faisceau électrique – Questions fréquemment posées de Brink », *Brink*. [En ligne]. Disponible sur: https://brink.eu/fr-fr/attelages/questions-frequemment-posees/quest-ce-quun-faisceau-electrique/. [Consulté le: 04-oct-2018].

[23] « Présentation d'Arduino - Arduino : premiers pas en informatique embarquée • Tutoriels • Zeste de Savoir », *Zeste de Savoir*. [En ligne]. Disponible sur: https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/742_decouverte-de-larduino/3414_presentation-darduino/. [Consulté le: 07-oct-2018].

[24] « Les systèmes embarqués - Dans les entrailles du Libre ». [En ligne]. Disponible sur: https://blog.fedora-fr.org/renault/post/Les-syst%C3%A8mes-embarqu%C3%A9s. [Consulté le: 20-sept-2018].

[25] « Le BUS CAN ». [En ligne]. Disponible sur: http://edelaunay.chez-alice.fr/buscan.htm. [Consulté le: 04-oct-2018].

[26] « Créez votre premier programme sur Arduino », *OpenClassrooms*. [En ligne]. Disponible sur: https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino/3285131-creez-votre-premier-programme-sur-arduino. [Consulté le: 24-sept-2018].

[27] « Cours Systèmes Embarqués:Le Bus CAN ». [En ligne]. Disponible sur: https://www.technologuepro.com/cours-systemes-embarques/cours-systemes-embarques-Bus-CAN.htm. [Consulté le: 15-sept-2018].

[28] « Cours Systèmes Embarqués: Introduction ». [En ligne]. Disponible sur: https://www.technologuepro.com/cours-systemes-embarques/cours-systemes-embarques-introduction.htm. [Consulté le: 15-sept-2018].

[29] « Can Somebody Steal Your Car By Calling It On The Phone? - The Truth About Cars ». [En ligne]. Disponible sur: https://www.thetruthaboutcars.com/2011/08/can-somebody-steal-your-car-by-calling-it-on-the-phone/. [Consulté le: 04-oct-2018].

[30] « A quoi servent les systèmes embarqués ? » [En ligne]. Disponible sur: https://www.orientation-education.com/article/a-quoi-servent-systemes-embarques. [Consulté le: 14-sept-2018].

[31] « Piratage d'une Tesla: ils transforment une voiture connectée en jouet téléguidé », *LExpansion.com*, 21-sept-2016. [En ligne]. Disponible sur: https://lexpansion.lexpress.fr/high-tech/piratage-d-une-tesla-ils-transforment-une-voiture-connectee-en-jouet-teleguide_1832943.html. [Consulté le: 02-oct-2018].

[32] Futura, « Système embarqué », *Futura*. [En ligne]. Disponible sur: https://www.futura-sciences.com/tech/definitions/technologie-systeme-embarque-15282/. [Consulté le: 14-sept-2018].

[33] K. Bouguerra, « Comment fonctionne l'ABS de votre voiture ? », *Motor1.com*. [En ligne]. Disponible sur: https://fr.motor1.com/news/176191/fonctionnement-systeme-abs-voiture/. [Consulté le: 05-oct-2018].

[34] F. Batteix, « Profil de skywodd », !-*APP.SHORT_TITLE-!* [En ligne]. Disponible sur: https://www.carnetdumaker.net/membres/skywodd/. [Consulté le: 06-oct-2018].

[35] F. Batteix, « Mesurer une distance avec un capteur à ultrason HC-SR04 et une carte Arduino / Genuino », !-*APP.SHORT_TITLE-!*, 09-nov-2016. [En ligne]. Disponible sur: https://www.carnetdumaker.net/articles/mesurer-une-distance-avec-un-capteur-ultrason-hc-sr04-et-une-carte-arduino-genuino/. [Consulté le: 06-oct-2018].

[36] 01net, « L'électronique embarquée, talon d'Achille de l'automobile moderne ? », *01net*. [En ligne]. Disponible sur: https://www.01net.com/actualites/l-electronique-embarquee-talon-d-achille-de-l-automobile-moderne-601063.html. [Consulté le: 19-sept-2018].