Rapport Projet Master 1 TNSI

Application Android de monitoring de capteurs



Une application pour un circuit de voiture

Année 2015/2016

Tuteur : Mikael Desertot

Jean-Baptiste DURIEZ et Jordane QUINCY

Table des matières

[Introduction 1](#_Toc452648120)

[Le projet 2](#_Toc452648121)

[Technologies et matériels : 6](#_Toc452648122)

[Une carte microcontrôleur : Arduino Mega ADK for Android 6](#_Toc452648123)

[Un potentiomètre numérique : MCP41100 6](#_Toc452648124)

[Un capteur infrarouge : GP2Y0A21YK0F 7](#_Toc452648125)

[Une application : Android 7](#_Toc452648126)

[Réalisation 8](#_Toc452648127)

[Comptage des tours 8](#_Toc452648128)

[Contrôle des voitures 11](#_Toc452648129)

[Nos difficultés 15](#_Toc452648130)

[Potentielle évolution du projet : 16](#_Toc452648131)

[Conclusion 18](#_Toc452648132)

[Webographie et Bibliographie 19](#_Toc452648133)

[Annexe I : Code source du programme du thermomètre 21](#_Toc452648134)

[Annexe II : Code source de l’application Android 22](#_Toc452648135)

[Annexe III : Code source de l’application Arduino 23](#_Toc452648136)

# Introduction

Initialement, le but de notre projet était de créer une application mobile permettant la gestion de différents capteurs qui seraient reliés à une carte microcontrôleur. L’application mobile devait donc pouvoir récupérer les informations des capteurs, envoyées par la carte microcontrôleur, puis réagir en fonction des données reçues. Afin de nous familiariser avec les technologies utilisées, nous avons dans un premier temps réalisé une application sur le microcontrôleur permettant de récupérer les données d’un capteur de température et d’afficher les valeurs du capteur en console sur l’ordinateur (vous pouvez retrouver le code source de ce programme en Annexe I). Cette première étape nous a permis de prendre en main le côté électronique du projet et de pouvoir voir les possibilités que nous avions en termes de réel projet. Nous avons par la suite décidé de nous lancer sur un projet plus complet qu’une simple application récupérant les données d’un capteur de température : une application pour circuit de voiture électrique !

# http://i2.cdscdn.com/pdt2/0/2/3/1/700x700/impc6023/rw/circuit-de-course-electrique-2-voitures.jpgLe projet

Il arrive des fois de retrouver son vieux circuit de voiture électrique dans le grenier mais malheureusement il manque les manettes pour contrôler les voitures et on ne peut donc pas revivre ces petits moments qu’on aimait tant dans notre enfance ! Notre projet est là pour résoudre ce problème. Il consiste en une application mobile, permettant de contrôler les voitures d’un circuit électrique et de voir le nombre de tours effectués par les deux voitures depuis son smartphone. Pour cela nous avons utilisé une carte microcontrôleur, un capteur infrarouge, un potentiomètre numérique, et enfin un appareil tournant sous Android.

Le principe est simple, depuis l’application sur le téléphone vous avez une jauge permettant de faire accélérer ou freiner la voiture. Une fois que l’utilisateur change la jauge sur son téléphone, l’application envoie à la carte microcontrôleur la valeur de celle-ci, puis elle va contrôler le potentiomètre numérique, en fonction de la valeur reçu par l’application, ce qui va permettre de ralentir ou d’accélérer la voiture. C’est donc le potentiomètre numérique qui va contrôler les bolides de course. En effet le potentiomètre est relié au circuit et il est constitué d’une résistance qu’on peut faire varier afin de faire passer plus ou moins de courant. Ainsi plus la résistance du potentiomètre est grande, plus le courant dans le circuit est faible et donc plus la voiture ralentit, et inversement, plus la résistance du potentiomètre est petite, plus le courant dans le circuit est grand et plus la voiture accélère. Pour résumer, si l’utilisateur veut faire accélérer la voiture, il augmente la jauge sur l’application, la valeur est envoyée à la carte microcontrôleur qui va faire diminuer la résistance du potentiomètre numérique ce qui va augmenter le courant dans le circuit et donc faire accélérer la voiture.

Ceci est le fonctionnement pour contrôler les voitures, mais on compte également les tours des voitures pour voir qui fait le plus de tour.

Pour cela nous avons utilisé 1 capteur infrarouge. La carte microcontrôleur envoie toutes les 100ms la donnée du capteur infrarouge à l’application Android. L’Android reçoit donc les valeurs et un algorithme, que nous avons programmé va vérifier (en analysant les valeurs) si une voiture est passée devant le capteur, et si oui, laquelle. On développera plus dans la partie réalisation le fonctionnement de cet algorithme.

Afin de mieux comprendre le rôle et le fonctionnement de la carte microcontrôleur et de l’application Android, nous avons réalisé deux diagrammes d’états-transitions, un pour l’application Android ([Figure 1](#Figure1)) et un pour la carte ([Figure 2](#Figure2)).

Figure 1 : Diagramme états-transitions de l’application Android



Tentative de connexion à la carte microcontrôleur

Affichage alerte « Pas de carte »

Etat d’écoute (Attente)

Envoi valeur à la carte

Lecture valeur venant de la carte

Algorithme de détection de tour

Ajout tour pour voiture 1

Ajout tour pour voiture 2

Mise à jour de l’écran de l’application

Lancement de l’application

Echec de la connexion

Fin alerte

Connexion réussie

Perte de connexion à la carte

Nouvelle valeur pour la jauge

Fin d’envoi

Fin lecture

Pas de nouveau tour

Détection nouveau tour pour voiture 2

Fin ajout

Fin ajout

Détection nouveau tour pour voiture 1

Fin mise à jour de l’écran

Demande de fermeture de l’application

Figure 2 : Diagramme états-transitions de la carte microcontrôleur



Tentative de connexion à l’application

Attente

Récupération valeur du capteur infrarouge

Envoi de la valeur à l’application Android

Tentative lecture valeur reçue d’Android

Modification valeur du potentiomètre

Lancement carte microcontrôleur

Connexion réussie

Fin de récupération de la valeur

Fin d’envoi à Android

Nouvelle valeur

Fin modification potentiomètre

Echec connexion

1 seconde écoulée

Pas de nouvelle valeur

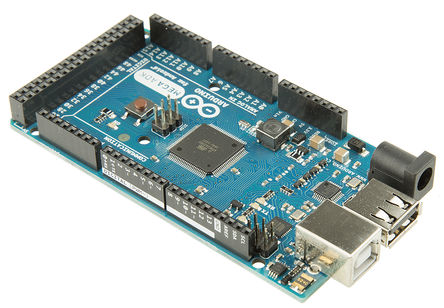
On voit bien sur le diagramme ([Figure 1](#Figure1)) qu’on essaye tout d’abord de se connecter à la carte microcontrôleur, si la connexion n’est pas établie alors on affiche une alerte pour dire qu’il n’y a pas de carte et on ferme l’application. Si la connexion est établie alors on fait 2 boucles en parallèle, une permettant d’envoyer à la carte microcontrôleur la valeur de la jauge (présente sur l’écran du smartphone ou de la tablette) si l’utilisateur la modifie. Et une autre boucle permettant de lire les valeurs du capteur infrarouge envoyées par la carte microcontrôleur, d’analyser ces valeurs grâce à notre algorithme d’analyse et de mettre à jour le nombre de tours des voitures (et donc l’écran) si jamais on ne détecte qu’une voiture a fait un tour de plus. Cette phase de « double boucles parallèles » est infinie est ne se terminera que si on demande à quitter l’application ou si on détecte une perte de connexion avec la carte microcontrôleur.

Pour la carte microcontrôleur, le diagramme est plus simple ([Figure 2](#Figure2)), parce que l’algorithme de recherche de tour a été fait sur l’application Android mais aussi parce que tout programme Arduino consiste tout simplement en une boucle infinie (et d’une phase de setup au lancement de l’application, que nous ne développerons pas ici). Dans cette boucle on va d’abord tester si on arrive à se connecter à l’application Android. Si la connexion ne s’établie pas, on se met en attente pendant 1 seconde et on teste, à nouveau, la connexion. Si la connexion est établie, on va lire la valeur du détecteur infrarouge, que l’on va envoyer directement à l’application Android. Puis on va essayer de lire une valeur que l’Android nous a potentiellement envoyé (valeur de la jauge, si modification par l’utilisateur), si on trouve une nouvelle valeur alors on va modifier la valeur du potentiomètre numérique avec cette nouvelle valeur, et sinon on finit la boucle et donc on teste à nouveau la connexion à l’application mobile.

# Technologies et matériels :

Pour notre projet nous avons utilisé différentes technologies et différents matériels :

## Une carte microcontrôleur : Arduino Mega ADK for Android

Arduino qui a vu le jour en 2001 en Italie, (appelé Genuino en 2015 pour les produits vendus en dehors des Etats-Unis suite à des querelles internes) est une plateforme de prototypage open-source, basé sur du « easy to use hardware and software » (utilisation qui se veut facile des matériels physiques et des logiciels). Arduino propose différentes cartes sur lesquels se trouve un microcontrôleur qui va nous permettre, dans notre cas, de commander des capteurs et d’envoyer les informations de ces derniers à une application mobile. Pour notre projet c’est l’Arduino Mega ADK for Android que nous avons utilisé. Il permet de contrôler des capteurs et de récupérer leurs données, mais aussi il permet facilement d’envoyer des informations vers une application Android, grâce notamment à une librairie open-source fournie par Android que nous avons utilisé : Android Accessory. La carte Arduino MEGA ADK for Android est l’une des seules cartes qui permet l’utilisation de cette librairie. Durant le projet nous avons utilisé l’IDE développé par Arduino, spécialement conçu pour programmer leurs cartes, la programmation de ces dernières se faisant en C ou C++. C’est cette carte qui sera notre interface entre l’application mobile est le circuit de voiture.

## Un potentiomètre numérique : MCP41100

Afin de gérer les voitures, il fallait pouvoir contrôler la tension présente dans le circuit électrique, et le composant électronique pour faire cela est le potentiomètre numérique. En effet le potentiomètre numérique est tout simplement une résistance dont la valeur est variable. Ainsi en reliant le potentiomètre au circuit électrique et en faisant varier la résistance on peut faire varier le courant dans le circuit électrique et donc faire varier la vitesse des voitures.

Afin de faire le lien entre le potentiomètre et le circuit de voitures nous avons dû ajouter un composant : le BUZ11 permettant d’augmenter l’intensité du courant. En effet, nous avons remarqué qu’en utilisant l’alimentation de la carte Arduino en sortie direct, on arrive à faire avancer les voitures, en revanche lorsqu’on récupère le courant en sortie du potentiomètre, bien que la tension soit suffisante (lorsqu’on met à zéro la résistance, la tension en sortie du potentiomètre est quasi similaire à la tension en sortie de la carte), l’intensité ne l’est pas du tout. Lorsqu’on mesure l’intensité en sortie du potentiomètre en mettant la résistance à zéro, on a une très grande perte d’intensité, et du le courant n’est plus assez puissant pour faire avancer les voitures.

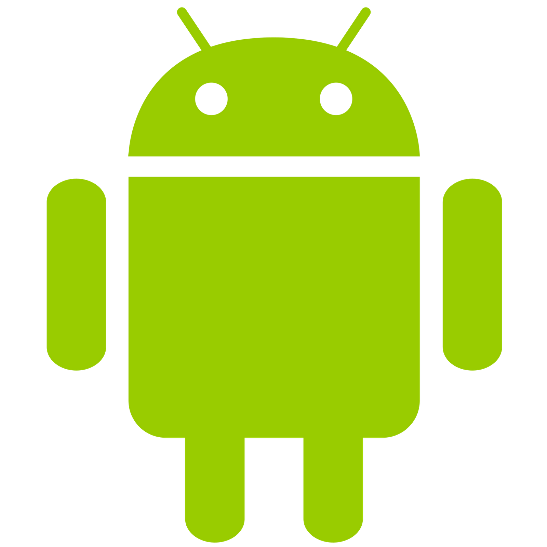
## Un capteur infrarouge : GP2Y0A21YK0F

Afin de détecter le passage des voitures, plusieurs choix étaient possibles, en effet nous pouvions utiliser des capteurs ultrasons, des capteurs de luminosité ou des capteurs infrarouges. Nous avons finalement choisi les capteurs infrarouges car nous en possédions un et après quelques tests, le capteur nous permettait de bien distinguer le passage des deux voitures.



## Une application : Android

Android est tout simplement le système d’exploitation le plus utilisé au monde avec plus de 80% des parts de marché pour les smartphones (source IDC – International Data Corporation). C’est sous cet OS de Google que nous avons développé notre application permettant de communiquer avec la carte Arduino et donc avec le circuit de voiture. Nous avons utilisé l’IDE de JetBrains, recommandé par Google : Android Studio.



# Réalisation

Au final, notre projet peut se résumer à deux réalisations différentes, la première s’occupant du comptage des tours, et la deuxième du contrôle des voitures. Nous allons donc voir le montage électronique de ces deux parties séparément pour que ça soit plus simple et surtout plus lisible. De même nous expliqueront le fonctionnement un peu plus en détails de chaque partie de manière séparée.

## Comptage des tours

Comme dit précédemment le comptage de tour se fait uniquement grâce à un capteur infrarouge. On retrouve sur la [Figure 3](#Figure3), le montage de ce capteur sur la carte Arduino en vue schématique et sur la [Figure 4](#Figure4) le même montage mais en vue un peu plus réaliste.

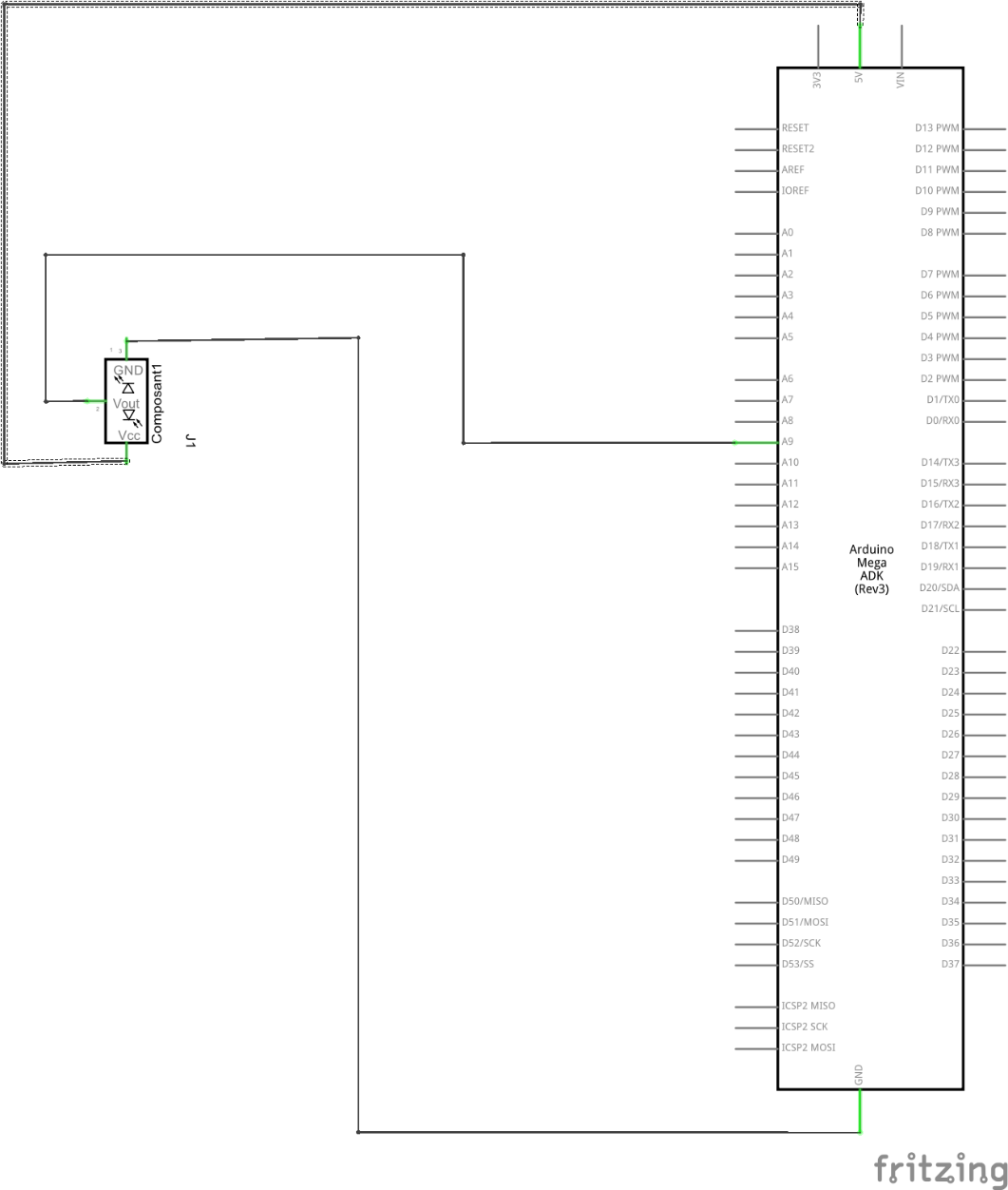


Figure 3 : Montage capteur infrarouge sur carte Arduino (vue schématique)

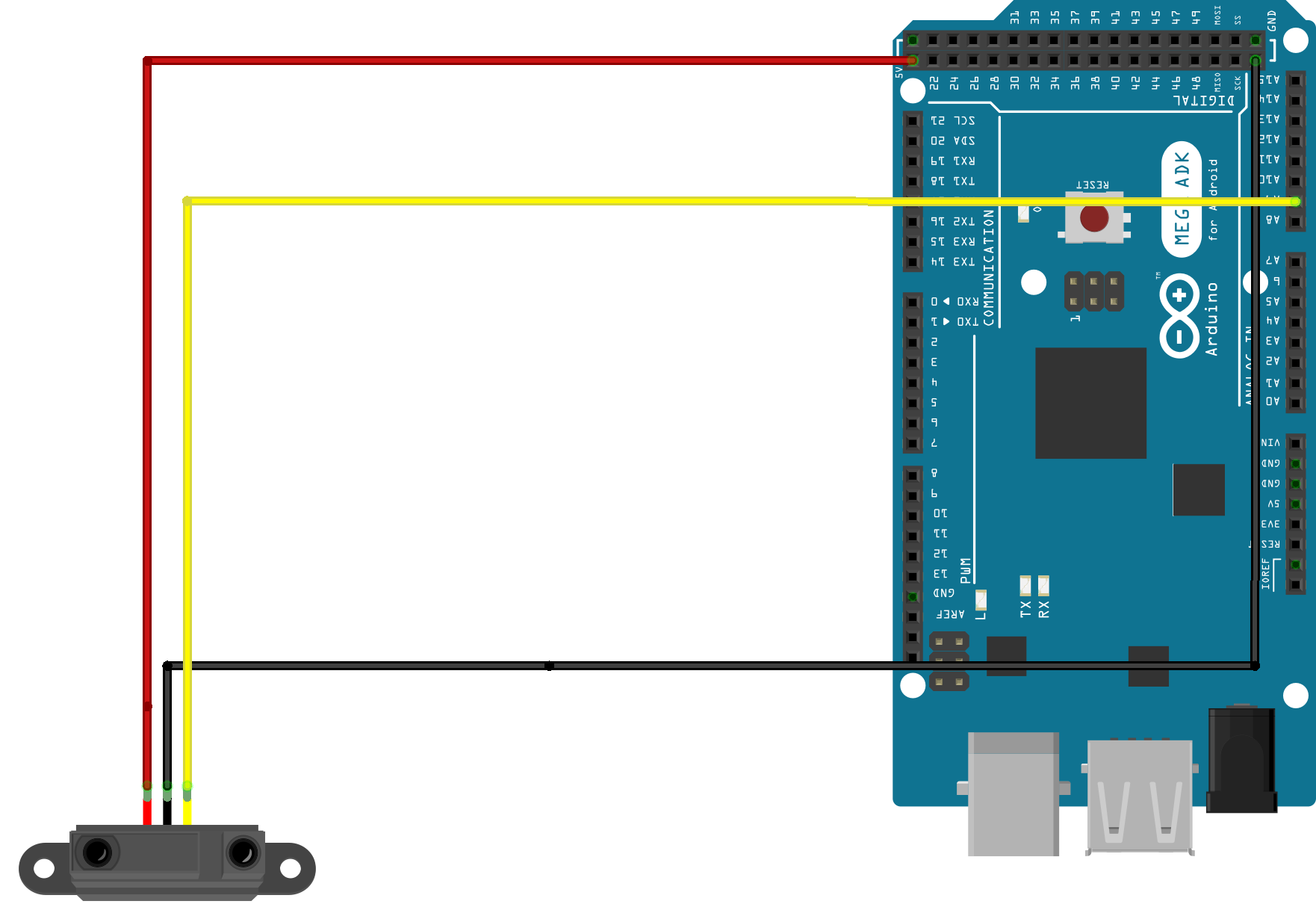


Figure 4 : Montage capteur infrarouge sur carte Arduino (vue réelle)

Le montage reste très simple, on branche au capteur l’alimentation de 5V que la carte Arduino propose en sortie (fil rouge sur la [Figure 4](#Figure4)), on branche également au capteur la masse de la carte Arduino afin de fermer le circuit (fil noir sur la [Figure 4](#Figure4)) et enfin on relie la sortie du capteur à la carte Arduino sur l’input A9 (fil jaune [Figure 4](#Figure4)). La valeur que nous envoie le capteur passe donc par ce fil jaune et on pourra lire la valeur sur le programme de la carte Arduino en lisant la valeur de l’input A9 (car on a branché le fil jaune sur cet input).

Afin de lire la valeur sur l’Arduino, il faut définir le pin A9 comme étant un « input » lors de la phase de setup du programme. Ensuite dans la boucle du programme on récupère le voltage que nous envoie le capteur sur le pin A9. En effet, le capteur renvoie un certain voltage, s’il n’y a pas d’obstacle devant le capteur le voltage est assez faible et constant (environ 1.3V dans notre cas), lorsque la voiture passe, le voltage que nous renvoie le capteur augmente puis diminue à nouveau une fois que la voiture est passée. Ce voltage est transformé en tableau de bytes afin de pouvoir être envoyé à l’application Android. L’application va ainsi pouvoir lire le voltage que la carte Arduino a envoyé, elle va le transformer en réel pour pouvoir l’analyse. C’est à ce moment que notre algorithme de recherche de nouveau tour ou non commence.

L’algorithme est conçu de la manière suivante : comme dit précédemment, lorsqu’une voiture passe devant le capteur, la tension envoyée par ce dernier augmente puis diminue. On obtient donc une courbe croissante puis décroissante (comme sur la [Figure 5](#Figure5)).

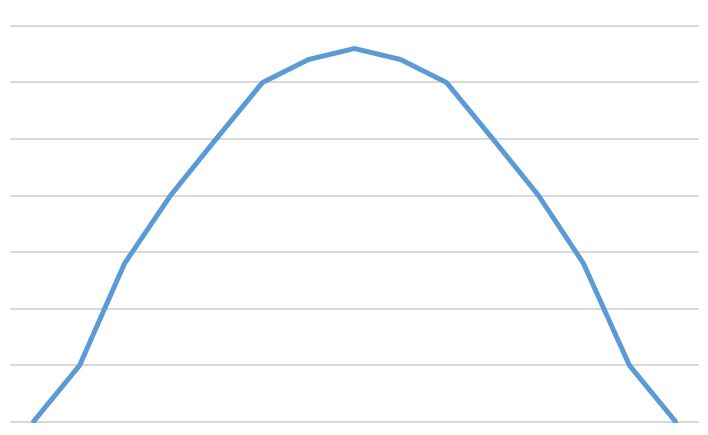


Figure 5 : Exemple de courbe de données lorsqu’une voiture passe devant le capteur

Le but de l’algorithme va donc de récupérer la valeur maximale des valeurs envoyées et ce, dès qu’une première valeur récupérée est au-dessus du voltage constant. Ensuite on compare le maximum qu’on récupère par rapport à certains seuils. En effet, afin de différencier les 2 voitures avec un seul capteur, il faut comparer les maximums qu’on récupère. La voiture sur l’extérieur du circuit est plus proche du capteur (car le capteur est situé sur le bord extérieur du circuit) et donc la courbe qu’on récupère lorsque cette voiture passe devant le capteur a une plus grande amplitude que la voiture intérieure qui, elle, est plus loin du capteur et dont la courbe est plus faible en amplitude. Ainsi, si le maximum est supérieur à 2.6V alors c’est que c’est la voiture extérieure du circuit qui a fait un tour, et si le maximum est entre 1.7V et 2.6V, alors c’est la voiture intérieure qui a fait un tour. On peut voir sur la [Figure 6](#Figure6) cette différence de courbe. Le point négatif de cette méthode est qu’il est impossible de détecter le passage des voitures, si les deux voitures passent en même temps (une seule des deux voitures sera détectée et ça sera la voiture extérieure car elle sera plus proche du capteur que la voiture intérieure).

Figure 6 : Superposition de courbe de valeurs pour le passage de la voiture extérieure et le passage de la voiture intérieure

## Contrôle des voitures

Pour le contrôle des voitures, le montage est un peu plus complexe. On va d’abord expliquer le montage du potentiomètre sur la carte. On peut voir sur la [Figure 7](#Figure7), le schéma électronique du potentiomètre lui-même.

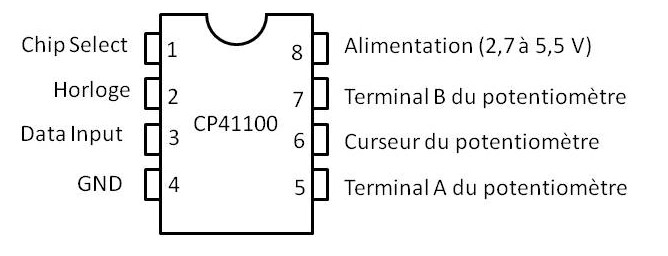


Figure 7 : Schéma électronique du potentiomètre

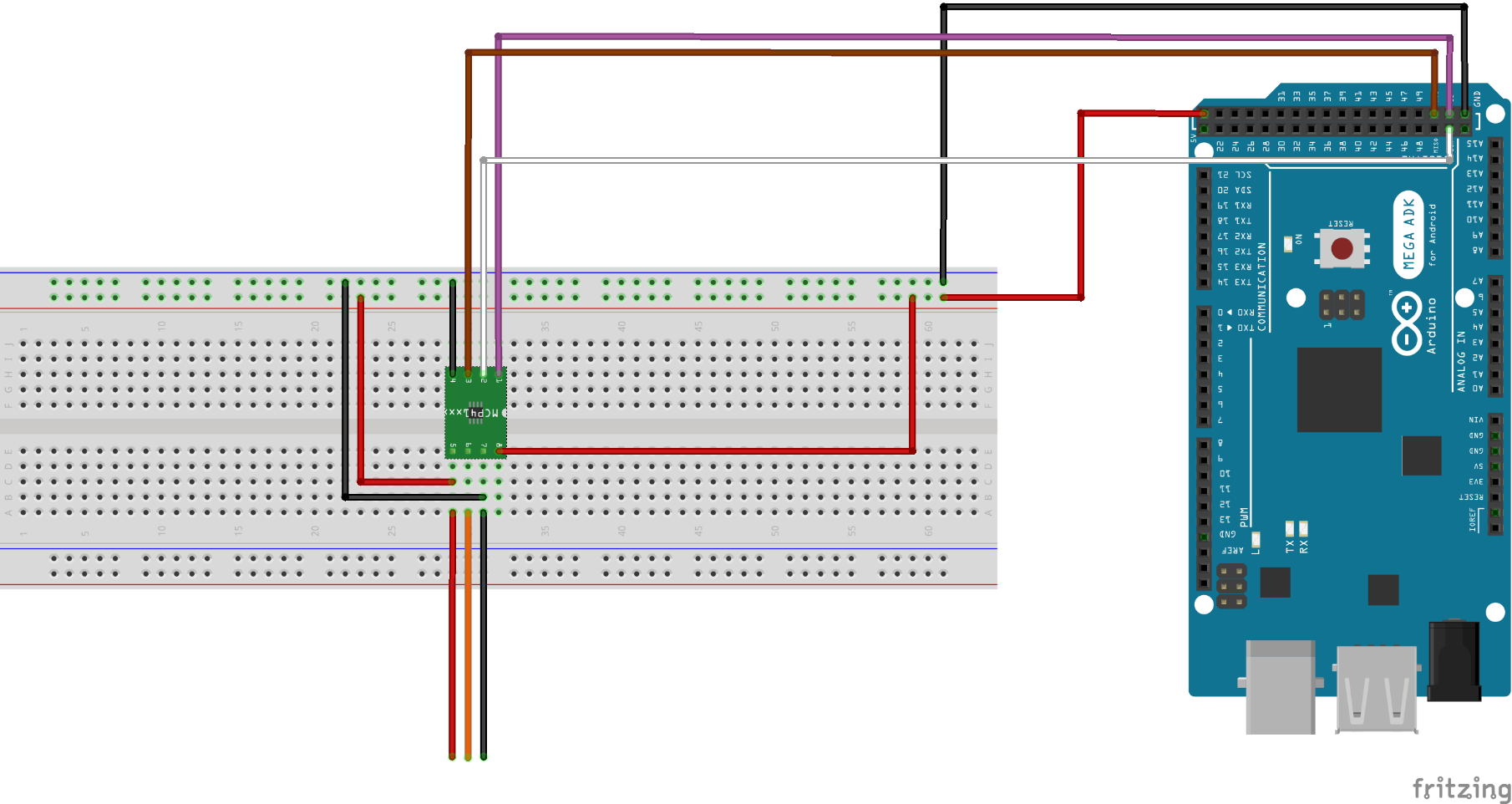
On a donc 6 inputs sur le composant (pins 1, 2, 3, 4 et 8) et 3 outputs (pins 5, 6 et 7). On va donc apporter les 6 inputs grâce à la carte comme on peut le voir sur la [Figure 8](#Figure8), et on va pouvoir utiliser les 3 outputs pour alimenter le circuit (en passant par le BUZ11).

Figure 8 : Montage potentiomètre sur carte Arduino

On apporte donc la source d’énergie via le fil rouge sur le pin 8 du potentiomètre. On connecte ensuite le chip select au pin ss de l’Arduino(slave select / fil violet), la carte Arduino peut contrôler plusieurs composants et c’est le pin ss qui permet de gérer ces multiples composants, on connecte ensuite l’horloge au pin SCK (correspond à l’horloge de l’Arduino, permet la synchronisation entre le potentiomètre et la carte Arduino) (fil blanc), le data input à la prise mosi (master output slave input) (fil marron) et enfin la masse à la masse (fil noir). Pour faire fonctionner le potentiomètre, nous avons dû ajouter une alimentation au niveau du terminal A du potentiomètre et la masse au niveau du terminal B du potentiomètre. Le terminal A correspond à la sortie de l’alimentation d’entrée du potentiomètre, le terminal B correspond à la masse en sortie du potentiomètre. Enfin, le curseur du potentiomètre est le courant qui est passé par la résistance du potentiomètre et qui a donc vu sa tension modifiée en fonction de la valeur de la résistance du potentiomètre.

C’est donc cette sortie (fil orange) qui doit apporter l’alimentation au circuit électrique. Nous avons regardé les caractéristiques du courant entre le curseur du potentiomètre et la masse de sortie (fil noir en bas) tout en faisant varier la valeur de la résistance du potentiomètre. Les résultats sont positifs puisque, lorsque la résistance du potentiomètre est au plus bas, la tension du courant en sortie est maximale (environ 9.5V) et plus la résistance du potentiomètre augmente, plus la tension en sortie diminue jusqu’à atteindre environ 0V lorsque la résistance est au maximum.

Nous arrivons donc bien à contrôler la tension en sortie du potentiomètre, mais comme dit précédemment (dans la description des technologies et matériels), l’intensité du courant diminue fortement en passant par le potentiomètre, il faut donc ajouter le BUZ11 avant de relier la sortie du potentiomètre au circuit de voiture afin d’avoir une intensité suffisante pour faire avancer les bolides.

On peut retrouver le montage du BUZ11 sur la [Figure 9](#Figure10), il faut simplement la masse au composant (fil noir), et on apporte le fameux fil orange qui est le courant qui est passé par la résistance du potentiomètre afin d’augmenter son intensité. La sortie du BUZ11 se trouve dans le fil noir tout en bas de la [Figure 9](#Figure10), et c’est l’alimentation (fil rouge) et ce fil noir que nous allons relier au circuit (à la place des manettes) (fils parant vers le bas).

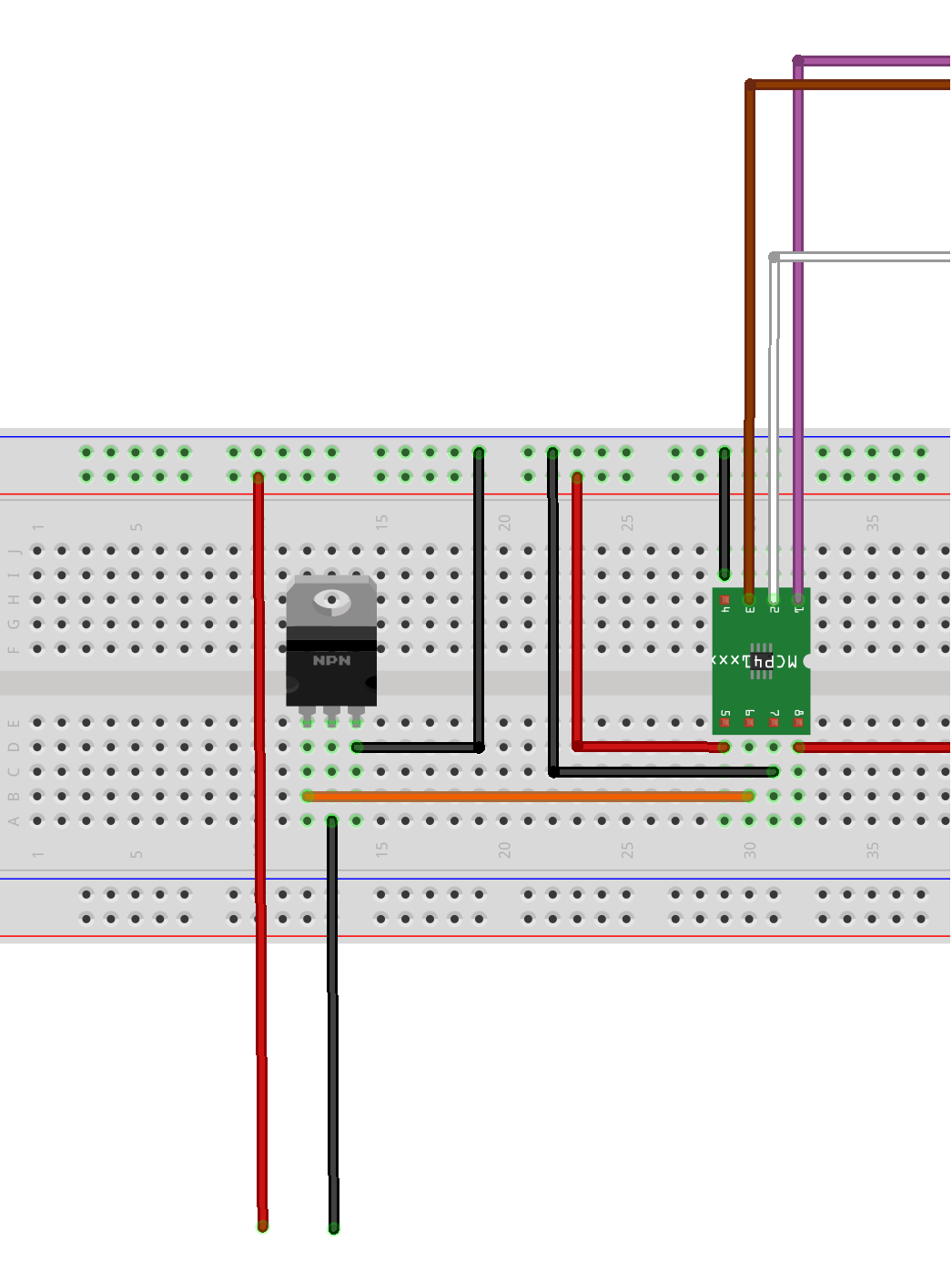


Figure 9 : Montage du BUZ11

Enfin côté application, pour régler la valeur du potentiomètre (et donc contrôler la voiture) il faut envoyer la valeur depuis l’application Android. Sur l’application on a une jauge allant de 0 à 255 (Ceux sont les valeurs acceptées par le potentiomètre). Lorsqu’on change la valeur de la jauge, on a un évènement qui est déclenché et qui va appeler une méthode pour envoyer cette valeur à l’Arduino (en la convertissant en tableau de bytes). Depuis l’Arduino, dans le setup, on précise que l’input 53 (correspond au pin ss de l’Arduino) est un potentiomètre, on essaye de lire ce que l’application Android nous envoie, et s’il y a quelque chose, on l’envoie au potentiomètre via le pin mosi.

Vous pouvez retrouver le code source de l’application Android en Annexe II et le code source de l’application Arduino en Annexe III.

# Nos difficultés

Les premières difficultés que nous avons rencontrées étaient d’établir une communication entre la carte Arduino et l’application Android. En effet la communication entre l’Android et l’Arduino fut presque immédiate mais l’inverse nous a demandé un peu plus de temps. Nous avons eu ensuite du mal à connecter nos composants fournis avec le TinkerKit car la librairie TinkerKit n’était plus à jour ce qui faisait planter la compilation. Nous avons donc modifié la librairie pour corriger les erreurs et nous avons fait un pull request sur le Github de TinkerKint afin de faire profiter notre correction à toute la communauté. (Dans l’application finale, nous n’utilisons pas de composants TinkerKit).

Ensuite les difficultés sont arrivées lorsque nous avons commencé le réel projet avec le circuit de voiture.

Nous avons tout d’abord eu du mal à détecter le passage des deux voitures avec le même capteur. En effet nous pensions même qu’il n’était pas possible de détecter le passage des deux voitures avec un capteur et qu’il en fallait deux. Mais finalement après réflexion nous avons trouvé un algorithme pour faire la détection avec un capteur.

Ensuite le branchement du potentiomètre numérique à la carte ne fut pas si facile. En effet il fallait trouver les bons pins à connecter et sur internet c’est principalement sur l’Arduino UNO que les explications sont faites, il fallait donc retranscrire le tout pour l’Arduino Mega ADK for Android, chose qui a donc pris un peu de temps. De plus nous n’avions pas du tout prévu que le courant en sortie du potentiomètre allait avoir une intensité trop faible (alors qu’en sortie de la carte, l’intensité est suffisante), nous avons donc pris beaucoup de temps pour savoir ce qui n’allait pas, pour savoir quel composant nous pouvions ajouter pour résoudre cela et revoir le montage pour intégrer le nouveau composant.

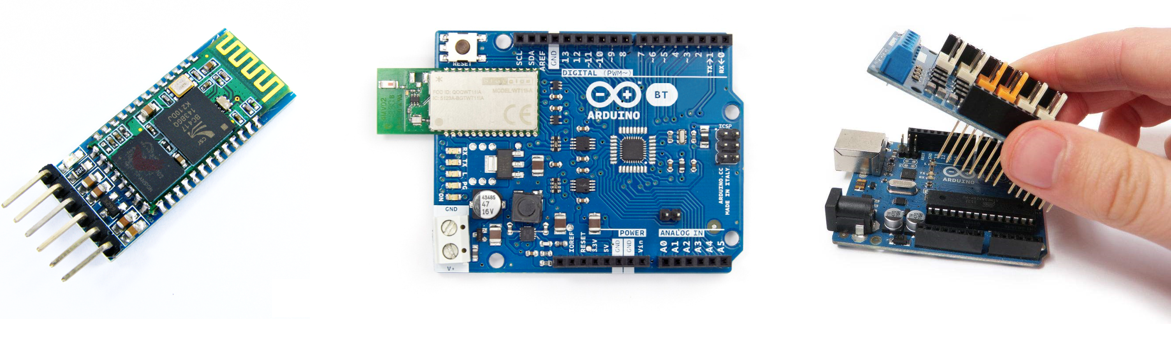
Enfin, pour la partie Android il n’y a pas eu trop de difficultés, l’application mobile reste assez simple, la plus grosse difficulté a été d’obtenir les autorisations pour pouvoir utiliser le vibreur ou l’USB de l’appareil.

# Potentielle évolution du projet :

Afin d’améliorer notre projet il y a plusieurs choses sur lesquels nous pouvons encore travailler :

La première amélioration serait de permettre la connexion à une deuxième application Android pour que 2 joueurs puissent jouer avec, chacun, leur voiture et leur smartphone sur le même circuit, il faudrait alors deux potentiomètres (un pour chaque voiture).

On peut aussi imaginer un moyen de communication sans fil pour l’Android et l’Arduino. Par exemple, un module Bluetooth sur la carte Arduino permettrait la connexion entre la carte Arduino et l’application Android sans passer par un câble USB. En effet il existe des petits modules Bluetooth qu’on peut relier à la carte permettant la connexion sans fil. Il existe même un Shield (carte à brancher à l’Arduino, sans soudure, pour ajouter plus de fonctionnalité) pour Arduino spécialement conçu pour ajouter cette fonctionnalité de Bluetooth.



Module Bluetooth HC-05 Shield Arduino BT Montage Shield sur Arduino

Toujours en pensant à supprimer les câbles, on pourrait également penser à une alimentation par une pile de 9 Volts pour la carte Arduino afin d’éviter à avoir à connecter la carte à l’ordinateur pour « juste » donner une source d’énergie. C’est très simple comme on peut le voir sur le montage de la [Figure 10](#Figure9) (même si c’est un carte Arduino UNO, cela fonctionnera parfaitement avec une carte Mega ADK).



Figure 10 : Montage pile et carte Arduino

On pourrait ajouter à l’application le nombre de fois qu’un joueur fait sortir la voiture du circuit, en regardant la tension dans le circuit et en détectant une brusque hausse de celle-ci.

Pour améliorer le comptage de tour, on pourrait ajouter un deuxième capteur afin de pouvoir détecter le passage des voitures lorsqu’elles finissent un tour en même temps.

Pour améliorer l’application, au lieu de simplement compter le nombre de tours, on pourrait afficher le temps qu’on met pour chaque tour et faire un système de points, en relation avec le temps d’un tour et le nombre de fois que le véhicule est sorti du terrain comme pénalité. On pourrait ajouter des challenges aléatoires qui apparaîtraient à tout moment, pour gagner plus de points, comme par exemple : ne pas sortir du terrain pendant 30 secondes. Les joueurs devront ainsi bien gérer leur vitesse pour ne pas sortir et gagner les points du challenge mais ne pas aller trop lentement non plus, parce que l’adversaire pourrait aller plus vite qu’eux.

Enfin une grosse évolution serait de pouvoir créer un circuit sur l’Android (avec un système de mapping), et pouvoir jouer à distance sur 2 circuits numériquement identiques mais physiquement distants. Ainsi deux amis pourraient jouer à distance, ils verraient sur leur circuit leur voiture mais aussi la voiture de l’adversaire qui bougera en même temps que sur le circuit de ce dernier (4 voitures en tout avec 2 circuits). Sur l’application on pourrait voir un point indiquant, sur le circuit numérique, la position des voitures. On pourrait d’ailleurs créer aussi une IA pour pouvoir jouer tout seul quand nos amis ne sont pas là.

# Conclusion

Ce projet a été très enrichissant pour nous, en effet, nous avons pu faire un projet non seulement informatique, mais aussi électronique. Grâce à cela nous avons pu découvrir des choses que nous ne connaissions pas et que nous n’aurions jamais vu au sein de nos cours, notamment au niveau électronique, avec les capteurs infrarouges ou encore l’utilisation d’un potentiomètre numérique. Cela nous a permis également de voir la liaison entre Arduino et Android (donc entre le côté électronique et informatique), mais aussi de développer nos compétences en développement d’application mobile qui est aussi un domaine peu abordé durant nos cours.

Ce projet nous a également permis de voir un peu de management de projet, en effet nous avons dû bien nous répartir les tâches pour bien avancer mais aussi savoir réagir rapidement en prenant les bonnes décisions lorsque des difficultés sont arrivées et notamment à la fin, lorsque nous n’arrivions pas à faire fonctionner les voitures avec le potentiomètre seul (sans BUZ11).

Enfin nous avons réussi à aller au bout de ce projet disciplinaire ce qui a permis de bien concrétiser tout l’aspect fonctionnel que nous avions préparé en amont.

# Webographie et Bibliographie

Référence [en ligne] Arduino.

<https://www.arduino.cc/en/Reference/HomePage>

Spécification [en ligne] Arduino.

<https://www.arduino.cc/en/Main/ArduinoBoardMegaADK>

Documentation [en ligne] Android.

<https://source.android.com/devices/accessories/aoa.html>

Référence [en ligne] Android.

<https://developer.android.com/adk/adk.html>

Librairie [en ligne] TinkerKit (github)

<https://github.com/TinkerKit/TinkerKit>

Tutoriel [en ligne] Instructables

<http://www.instructables.com/id/Digital-Potentiometer-MCP41100-and-Arduino>

Tutoriel [en ligne] ElectroniqueAmateur

<http://electroniqueamateur.blogspot.fr/2013/04/faire-varier-une-resistance-au-moyen-de.html>

Tutoriel [en ligne] JeffreySambells

<http://jeffreysambells.com/2011/05/15/identifying-your-android-usb-accessory>

John NUSSET. *Arduino pour les nuls*. First Interactive, 2014, 456p. ISBN-13 978-2754064293

Massimo BANZI. *Démarrez avec Arduino Principes de base et premiers montages*. Dunod, 2011, 128p. ISBN-13 978-2-10-056291-6

Sylvain HEBUTERNE. *Développer une application Android Programmation en Java sous Android Studio*. Expert IT, 2015, 418p. ISBN-13 978-2746097087

Table des Annexes

[Annexe I : Code source du programme du thermomètre](#Annexe1)

[Annexe II : Code source de l’application Android](#Annexe2)

[Annexe III : Code source de l’application Arduino](#Annexe3)

# Annexe I : Code source du programme du thermomètre

#include <TinkerKit.h>

#define PIN\_TEMP\_SENSOR I1 // thermistor connected on I1 (TinkerKit shield)

TKThermistor thermistor**(**PIN\_TEMP\_SENSOR**);** //create the helper class and set the pinmode (INPUT)

float celsius**;**

void setup**()** **{**

Serial**.**begin**(**115200**);**

**}**

void loop**()** **{**

celsius **=** thermistor**.**readCelsius**();** // Reading the value from the thermistor

Serial**.**print**(**"Temperature : "**);**

Serial**.**print**(**celsius**);**Serial**.**print**((**char**)**176**);**Serial**.**print**(**"C \r\n"**);** // (char)176 == ascii code of ° (degree sign)

delay**(**1000**);** // wait for a second

**}**

# Annexe II : Code source de l’application Android

Malheureusement nous avons reçu le BUZ11 seulement le Mercredi 01 Juin donc le code est toujours en évolution afin de totalement terminer les derniers petits réglages pour que l’application fonctionne correctement. Le code final se trouvera donc sur les feuilles données durant la soutenance.

# Annexe III : Code source de l’application Arduino

Malheureusement nous avons reçu le BUZ11 seulement le Mercredi 01 Juin donc le code est toujours en évolution afin de totalement terminer les derniers petits réglages pour que l’application fonctionne correctement. Le code final se trouvera donc sur les feuilles données durant la soutenance.