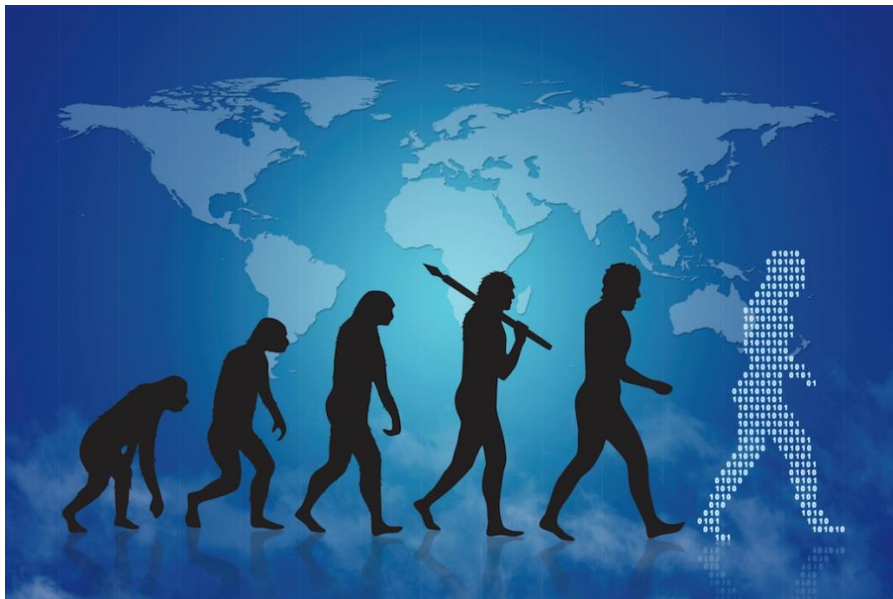


# Portfolio

***"Une personne qui n'a jamais commis  
d'erreurs n'a jamais tenté d'innover. "  
Albert Einstein***

---





# Sommaire

<b>Traceur .....</b>	<b>1</b>
<b>Robot-voiture autonome .....</b>	<b>2</b>
<b>Freescape Cup .....</b>	<b>3</b>
<b>Contrôleur de leds via Android .....</b>	<b>4</b>
<b>Jeu vidéo (Bataille navale) .....</b>	<b>5</b>

# Traceur



FIGURE 1 CARTE AVEC LA LOCALISATION DES DEUX CARTES EMBARQUEES

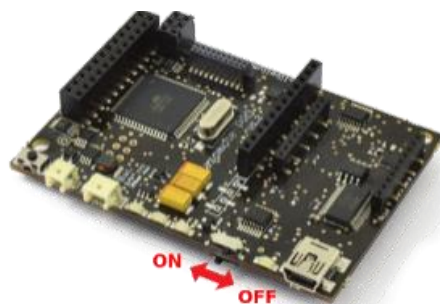


Figure 2 Carte embarquée Wapsmote



Figure 3 Carte embarquée Ublox

### **But**

Pouvoir connaître le placement d'un colis grâce à un système embarqué

### **Equipe**

Deux étudiants en 3<sup>ème</sup> année d'école d'ingénieur

### **Logiciels utilisés**

Plateforme MBED, Plateforme Wasmote

### **Matériel utilisé**

Carte embarquée Ublox C027

Carte embarquée Wapsmote

### **Langage de programmation**

Langage C, C++

### **Durée**

50 heures

(Février 2015 – Avril 2015)

### **Méthodes**

L'utilisateur entre les coordonnées d'arrivées dans un fichier texte ainsi que le numéro du téléphone devant être averti de l'arrivée du colis.

Le système embarqué reçoit ses propres coordonnées GPS toutes les 5 minutes et les envoie au serveur.

Le serveur compare les coordonnées.

Lorsque le système est arrivé à destination, le serveur envoie un message avec le numéro à joindre au système concerné.

Le système envoie un SMS à l'utilisateur pour avertir de son arrivé

### **Résultat final**

Réception de données GPS sur les systèmes embarqués

Envoi de trames sur un serveur

Réception de trame du serveur

Envoi d'un SMS lorsqu'on reçoit une trame correcte du serveur

# Robot-voiture autonome

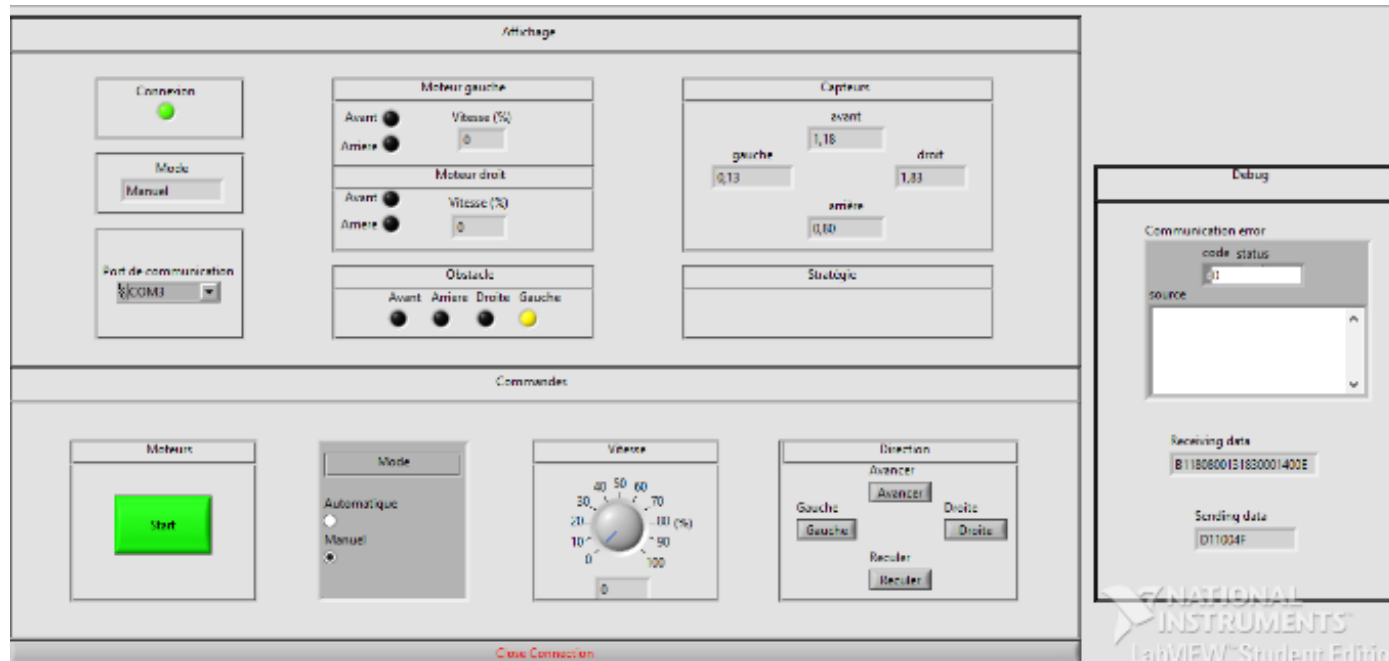


Figure 6 Monitoring avec Labview

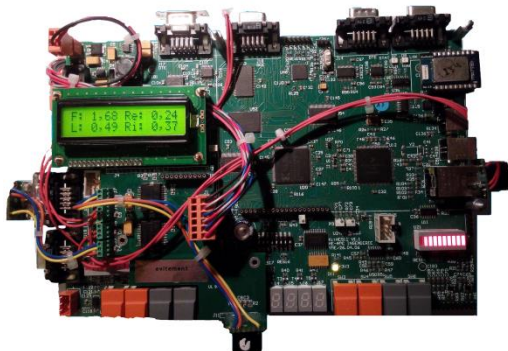


Figure 5 Voiture-robot (vue du dessus)

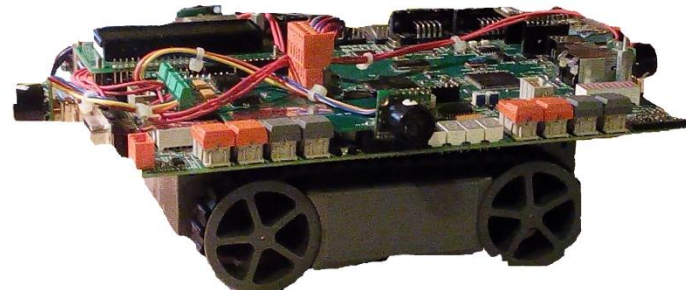


Figure 4 Voiture-robot (vue de gauche)

**But**

Le robot doit pouvoir se déplacer dans un environnement et éviter tous les obstacles. On doit pouvoir le guider depuis une interface.

**Equipe**

Seul

**Logiciels de programmation**

Codewarrior, LabVIEW

**Langage de programmation**

Langage C

**Durée**

150 heures  
(Février 2015 – Avril 2015)

**Méthodes**

Utilisation de capteurs ultrasoniques pour déterminer les distances de part et d'autres du robot.

Celui-ci se déplace de manière automatique et évite les obstacles.

Depuis l'interface utilisateur, on peut choisir de laisser le robot se déplacer seul ou choisir de le guider nous-même.

On peut voir la distance de chaque capteur et où se trouve un obstacle.

**Résultat final**

Le robot se déplace de manière automatique.

On peut le commander depuis l'interface utilisateur (choix des vitesses, commande du départ et de l'arrêt, choix du mode automatique ou manuel, possibilité de le guider nous-même)

# Freescaple Cup

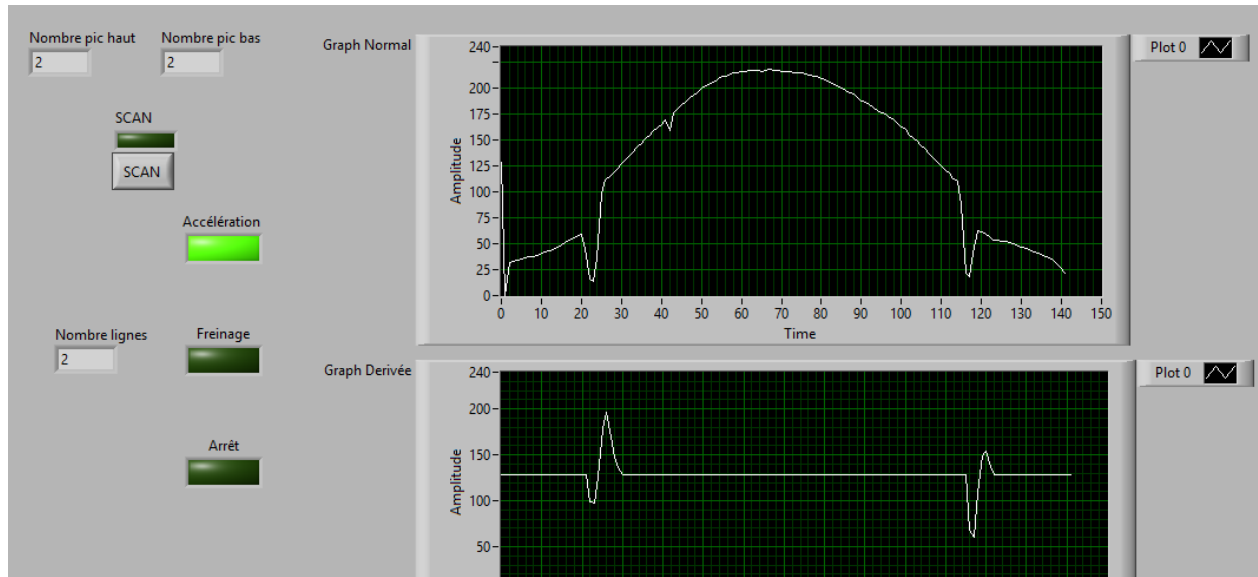


Figure 9 Monitoring Labview

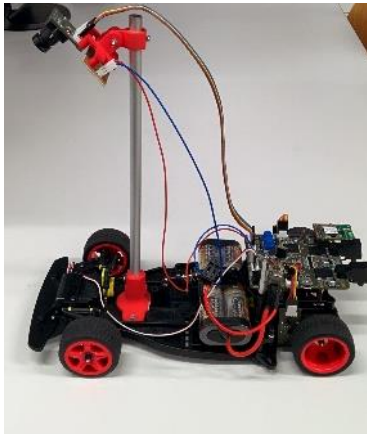


Figure 7 Voiture Freescaple Cup (vue de gauche)

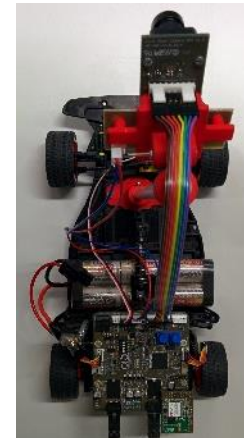


Figure 8 Voiture Freescaple Cup (vue du dessous)



**But**

Conception d'une voiture devant se déplacer seul sur un circuit, sans en sortir, le plus rapidement possible

**Equipe**

Deux étudiants en 3<sup>ème</sup> année d'école d'ingénieur

**Logiciels de programmations**

Codewarrior, LabVIEW

**Langage de programmation**

Langage C

**Durée**

200 heures

(Septembre 2014 – Février 2015)

**Méthodes**

Utilisation d'une caméra à l'avant permettant d'identifier les lignes noires de chaque côté du circuit  
Traitement de l'image et interprétation sur les moteurs

Visualisations de l'image perçue sur une interface utilisateur

**Résultat final**

Voiture faisant un tour de circuit sans sortir

Vitesse moyenne (environ 30-40% de la vitesse maximale)

Visualisation de l'image reçue de la caméra, de la vitesse donnée aux moteurs, dans quel état la voiture se trouve (accélération, freinage, arrêt)

## Contrôleur de leds via Android

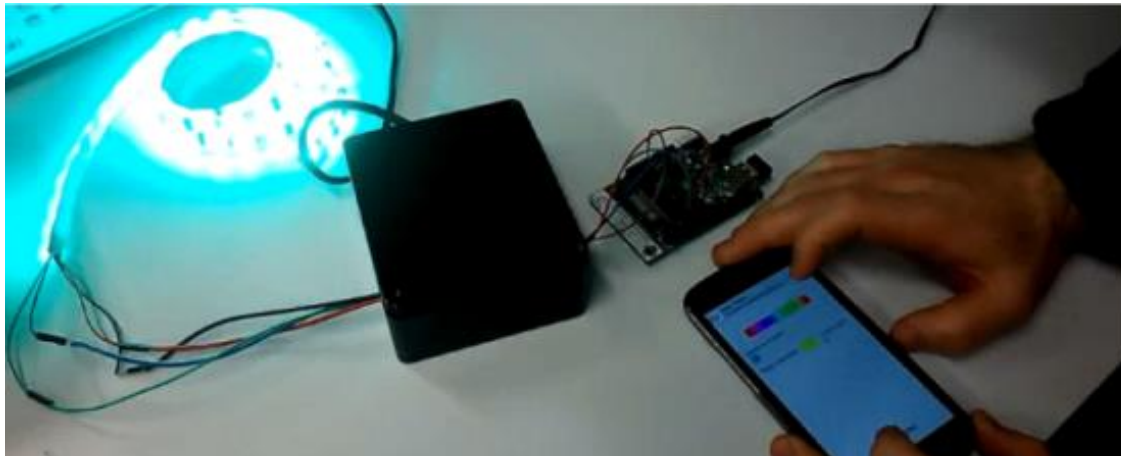


Figure 12 Montage complet avec la carte PIC24F, des transistors (dans la boîte noire) pour envoyer les signaux de couleurs en PWM sur le ruban de leds



Figure 11 Une des pages de l'application Android



Figure 10 PIC24F (de chez Microchip)

**But**

Création d'une application permettant de changer la couleur de leds à distance

**Equipe**

Deux étudiants en 3<sup>ème</sup> année d'école d'ingénieur

**Logiciels utilisés**

Eclipse (programmation du téléphone Android)  
MPLAB X IDE (programmation de la carte embarquée)

**Langage de programmation**

Langage C (pour le système embarqué et Java  
Android (pour les appareils Android

**Matériel utilisé**

PIC24F (carte embarquée)

**Durée**

200 heures  
(Septembre 2014 – Février 2015)

**Méthodes**

Envoi de la couleur en Bluetooth depuis un téléphone ou tablette Android  
Réception de la trame contenant les couleurs sur une carte embarquée  
Envoi de la couleur de la carte embarqué sur un ruban de leds

**Résultat final**

Changement de couleur en 4 méthodes (Rotation du téléphone, Shake, Slide, Entrée de valeurs manuellement)  
Envoi en Bluetooth sur la carte  
Changement de la couleur des Leds

## Jeu video (Bataille navale)

---



Figure 13 Première page du jeu vidéo

**But**

Création d'un jeu vidéo permettant à un joueur de jouer contre l'ordinateur

**Equipe**

Deux étudiants en 1ère année de DUT Génie Electronique et Informatique Industrielle

**Logiciels utilisés**

Eclipse, Librairie SQL

**Langage de programmation**

Langage C

**Durée**

200 heures  
(Février 2013 – Juin 2013)

**Méthodes**

Placement des bateaux sur une grille prédéfinie  
Placement des bateaux de l'ordinateur de manière aléatoire sur la même grille prédéfini  
Jeu alternatif entre le joueur et l'ordinateur

**Résultat final**

Placement des bateaux sur une grille sans dépassement

Placement des bateaux de l'ordinateur de manière aléatoire et non visible du joueur