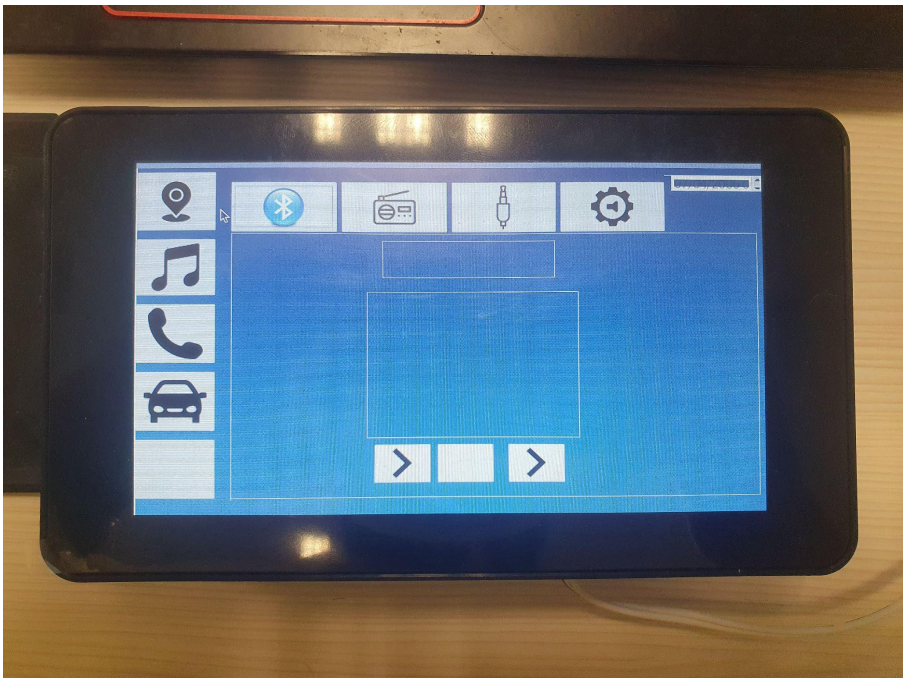
I.U.T de Nîmes-Université de Montpellier

Rapport Final
Ecran de voiture



Etudes & Réalisation - Semestre 3 - Semestre 4

Étudiants : GRANAUD Malcolm & MICHEL Bastien

Années :2021/2022

Professeurs : Stephane REYES, Franck LECAT, Frederic GIAMARCHI et Thierry FIOL

Département G.E.I.I

Sommaire

Présentation

Le module Études et Réalisations permet aux étudiants en IUT Génie Électrique et Informatique Industrielle de découvrir l'Apprentissage par Projets et Problèmes. Un projet consiste à étudier et réaliser un montage. Ce document est une trame pour commencer à étudier un montage électronique de A à Z.

Après une présentation du montage, ses applications, celui-ci est entièrement étudié et dimensionné. Par la suite, le schéma et la carte électronique (circuit imprimé) sont dessinés. Enfin, la fabrication du montage est réalisée et testée.

Un document (rapport de stage) est rédigé permettant de garder une trace du travail effectué.

Cahier des charges

Il s’agit de réaliser un prototype d’écran de voiture à l’aide d’une carte Raspberry PI 4 et d’un écran 7” de chez Raspberry. Pour réaliser l’interface graphique de l’écran nous avons choisi d’utiliser le logiciel QT version 5.12.

Constitution du matériel

Pour la réalisation de ce projet, nous avons utilisée :

- Une carte Raspberry PI 4ème génération
- Un écran 7” Touchscreen Display
- Une Carte SD 16 GB
- Nos ordinateurs portable
- Le logiciel QT5.12
- Oracle (logiciel qui nous a permis de réaliser une virtual box)
- Le logiciel Ubuntu

Organisation

Le projet doit se dérouler sur 13 séances de 3h. Celle-ci sont réparti entre les diverses actions qui permettront d'avancer le projet

1ère séance : Préface du projet

2ème séance : Câblage et Mise en service du Raspberry PI

3ème séance : Installation de QT sur l 1er partie : Préparer le Raspberry PI pour la compilation

4ème séance : Installation de QT sur le raspberry 2eme partie : Ubuntu Virtualbox - compilation de QT

5ème séance : Installation de QT sur le raspberry 3eme partie: Ubuntu Virtualbox - compilation de QT

6ème séance : Installation de QT sur le raspberry 4eme partie:

Ubuntu Virtualbox - compilation de QT

7ème séance : Installation de QT sur le raspberry 5eme partie:

Changement de méthode

8ème séance : Initiation à QT

9ème séance : Initiation à QT

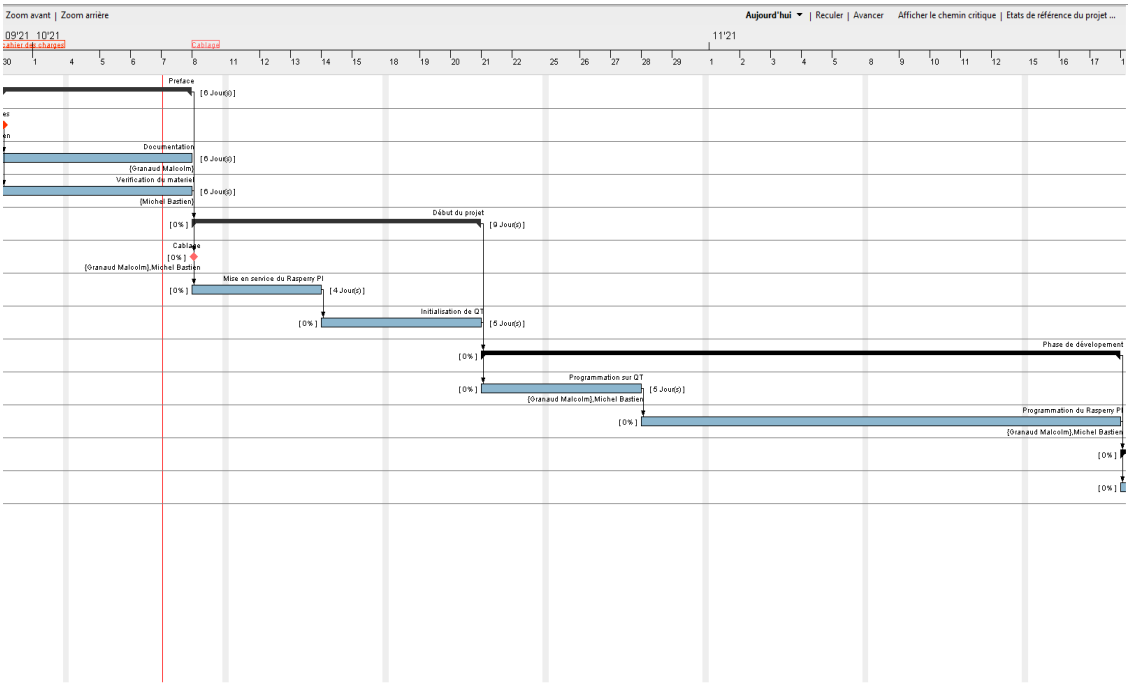
10ème séance : Création de l'interface graphique

11ème séance : Création de l'interface graphique

12ème séance : Rapport final

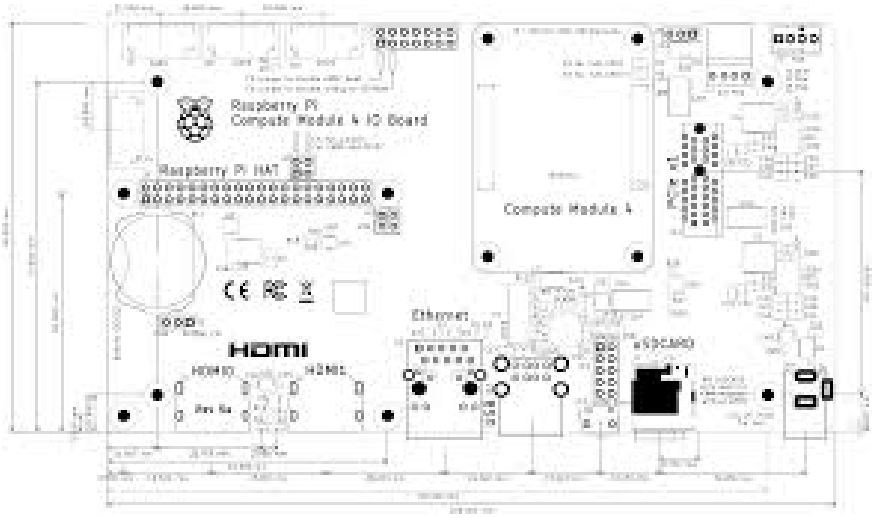
13ème séance : Présentation oral du projet au autres étudiants

Planning



Voici le premier diagramme de Gantt organisé lors de la 2eme séance d’ER3. Comme on peut le voir celui-ci n'est pas complet il sera complété lors de l’évolution du projet.

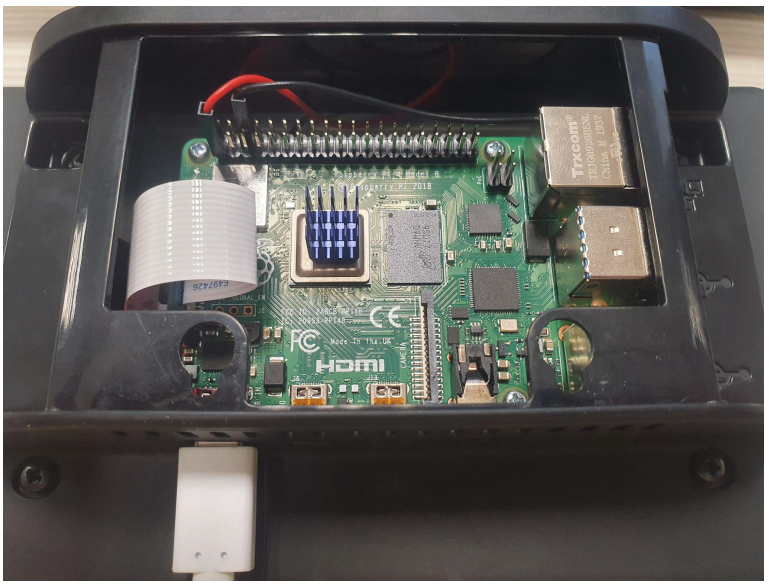
Etude de la carte électronique Raspberry PI 4



Dans un premier temps nous allons nous intéresser aux caractéristiques de la carte électronique Raspberry PI 4.En étudiant le datasheet de la carte nous avons pu constater que celle-ci est alimentée en 5V.

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Image reprise de la datasheet



Début du projet

Installation de QT sur le raspberry PI 1er partie

Pour l'installation de QT sur le raspberry PI 4 nous avons suivis la vidéo suivante : https://www.youtube.com/watch?v=TmtN3Rmx9Rk&t=173s

Le vidéaste a découper sa vidéo en 3 étapes qui sont :

- Préparer le Raspberry Pi pour la compilation
- Ubuntu Virtualbox - compilation de QT
- Installation de QT creator

1er partie : Préparer le Raspberry PI pour la compilation

Dans cette partie le vidéaste installe l'OS Raspbian Buster sur le raspberry. Il met à jour l'OS puis il redémarre le Raspberry ensuite il configure l'interface du Raspberry PI. Après avoir réalisé la configuration du Raspberry PI, il remet à jour le Raspberry PI puis il le reboot. Une fois redémarrer, il installe toutes les librairies dont il a besoin pour QT.

2eme partie : Ubuntu Virtualbox - compilation de QT

Dans cette partie le vidéaste utilise le logiciel Oracle qu'il avait téléchargé auparavant ainsi que le fichier d'installation de Ubuntu. Par la suite, il crée une virtualbox dans laquelle il a implanté le logiciel Ubuntu. Il procède par la suite à l'installation de Ubuntu dans la virtualbox et utilise l'utilisateur de commande afin de pouvoir réaliser l'installation de QT sur la virtualbox.

Nous avons décidé d'arrêter de suivre la vidéo et de recommencer car nous avons rencontré de nombreuses erreurs que l'on n'est pas parvenue à résoudre malgré l'aide de Mr FIOL.

Changement de méthode

Comme indiqué précédemment, nous avons décidé de changer de méthode afin de perdre moins de temps que nous en avons perdu. On a donc suivi les explications d'une autre personne sur son site dont voici le lien

Programmation de l'interface graphique

Après avoir réussi l'installation du logiciel Qt nous avons commencé la programmation de l'interface graphique. Vu que nous ne connaissions pas le logiciel Qt avant le début du projet, nous avons visionné plusieurs tutoriels sur youtube afin de pouvoir apprendre comment utiliser le logiciel.

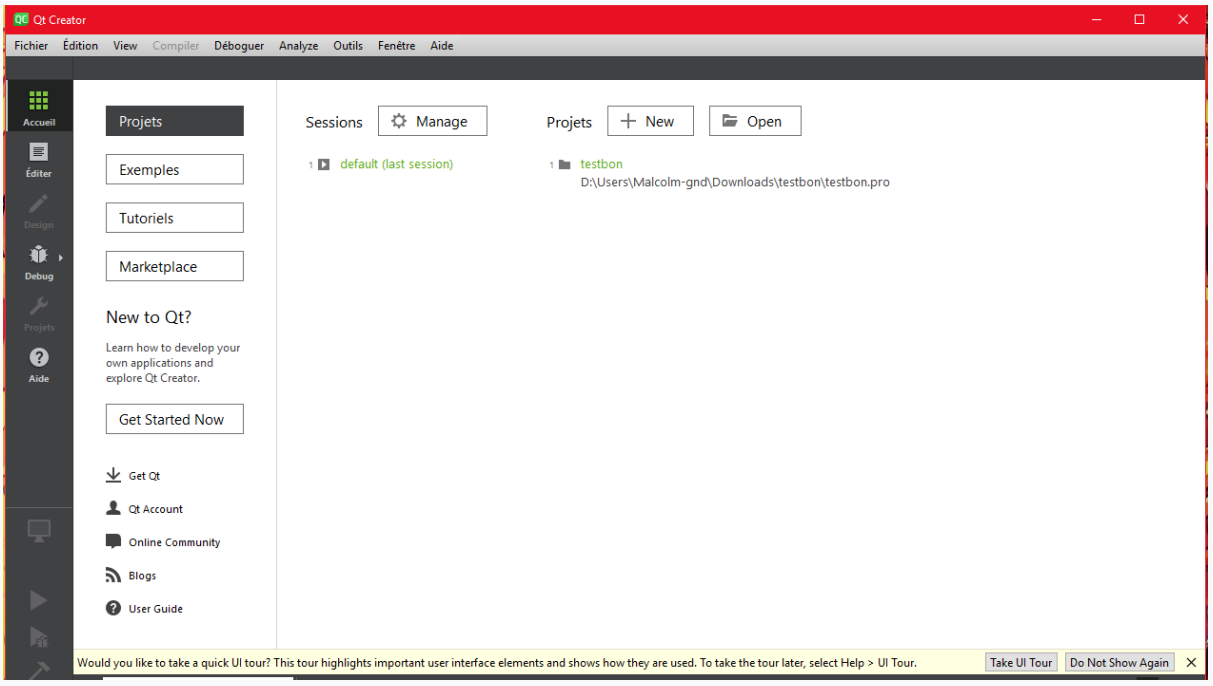
Début de la programmation de l'interface graphique

Nous avons dans un premier temps créer un projet sur l'ordinateur de Bastien pour plus de facilité au niveau de la programmation. Nous avons tout d'abord vérifié si nous pouvions lire le projet QT créé au préalable sur l'ordinateur de Bastien sur le Raspberry. Pour pouvoir lire correctement le projet sur le Raspberry nous avons envoyé le projet par mail. Nous l'avons ensuite récupéré depuis le Raspberry, puis ouvert le projet sur QT creator. On a pu constater que le projet pouvait être lu sur le Raspberry.

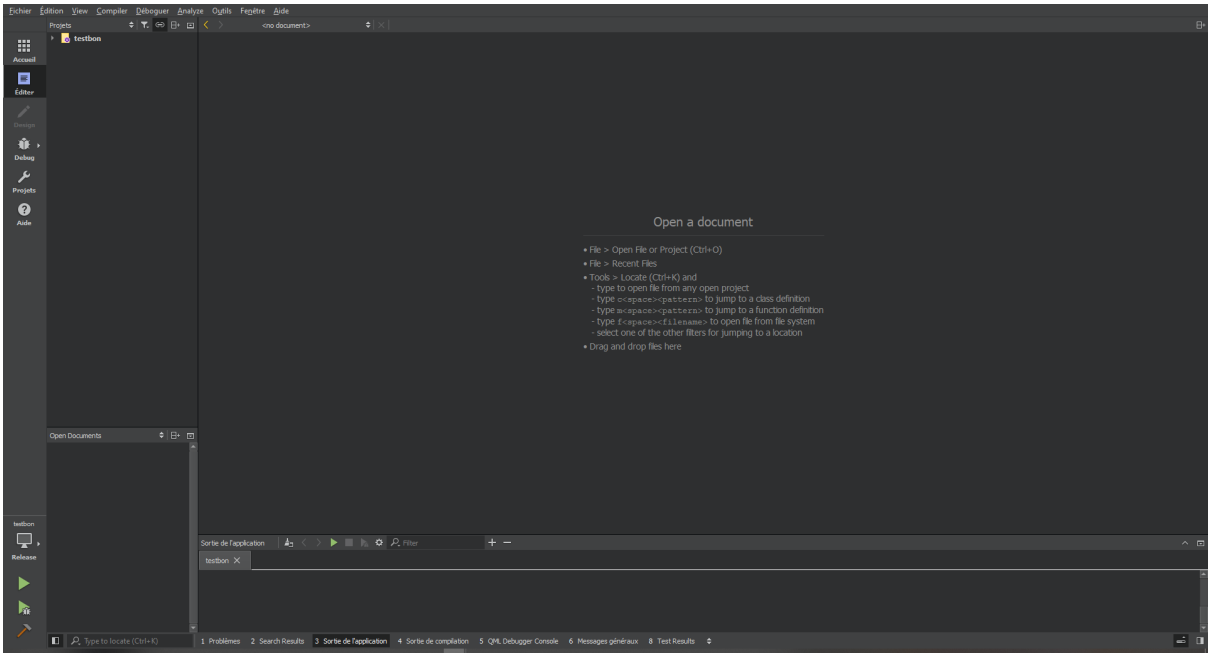
Après avoir réalisé cette vérification, nous avons enfin pu commencer la programmation de l'interface graphique sur QT creator. Dans un premier temps, on a recherché sur internet des exemples d'écrans de voiture pour se faire une idée des différentes pages à programmer.

Comment fonctionne QT creator

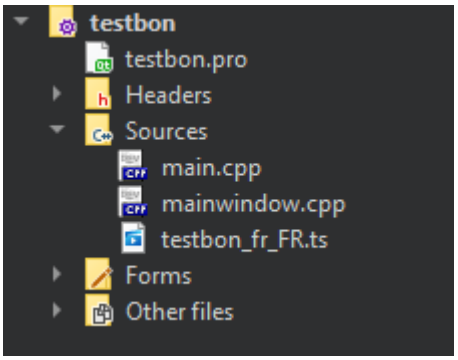
Quand on démarre QT creator, une page s'ouvre avec plusieurs rubriques. La rubrique projet est ouverte par défaut



Dans cette rubrique on sélectionne open puis on sélectionne le projet. Le projet s'ouvre alors.



En haut à gauche un fichier est alors visible. Il faut le dérouler pour pouvoir accéder aux différentes pages du projet.

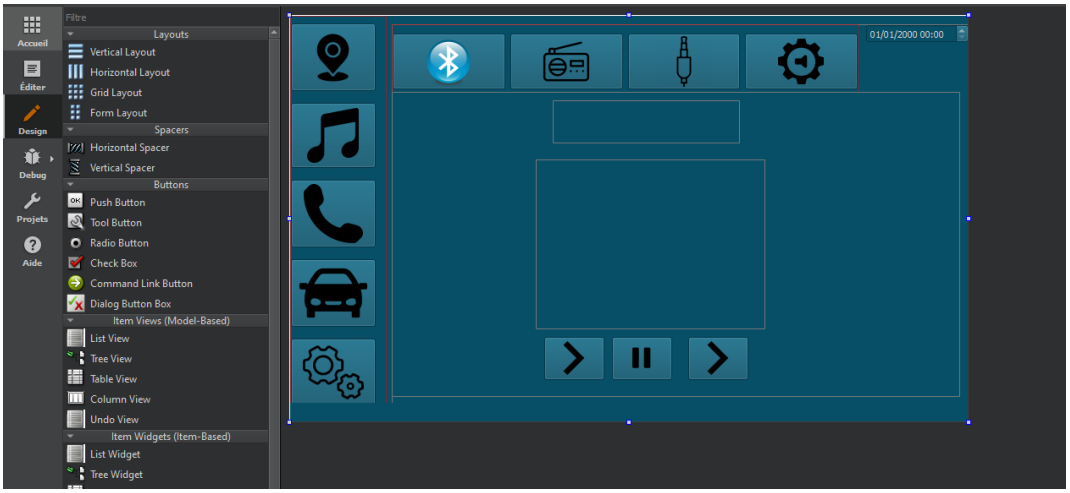


Le principal fichier sur lequel il faut travailler est le fichier source. Dans celui-ci nous retrouvons deux fichier un intitulé main.cpp et l'autre s'appel mainwindow.cpp .

```
void MainWindow::on_pushButton_3_clicked() //cette fonction s'active sur l'appui sur le bouton musique
{
    ui->musique->show();    //permet d'afficher l'ecran musqiue
    ui->GPS->hide();        //permet de cacher l'ecran GPS
    ui->telephone->hide();  //permet de cacher l'ecran telephone
}

void MainWindow::on_pushButton_clicked()
{
    ui->GPS->show();
    ui->musique->hide();
    ui->telephone->hide();
}

void MainWindow::on_pushButton_2_clicked()
{
    ui->telephone->show();
    ui->GPS->hide();
    ui->musique->hide();
}

void MainWindow::on_ble_bouton_clicked()
{
    ui->ble_frame->show();
    ui->radio_frame->hide();
}

void MainWindow::on_radio_bouton_clicked()
{
   ui->ble_frame->hide();
   ui->radio_frame->show();
}
```

Dans le fichier mainwindow.cpp, nous observons les différentes fonctions créer pour chacun des boutons utilisés.

Pour modifier l'interface graphique, il faut ce rendre dans la rubrique design (voir ci-dessous).



Réalisation de l'interface graphique

Après une longues réflexions, nous avons décidé de mettre un verticale layout sur le côté gauche de l'écran dans lequel nous avons placé 5 pushboutons représentant les 5 principales options de notre écran. Pour chacun des menus, nous avons utilisé des frames dans laquelle nous avons mis tout ce dont nous avions besoin (ex: un bouton Bluetooth dans le menu musique.). A chaque appui sur un des boutons principaux, nous affichons la frame qui correspond et cachant les autres frames des autres menus.

```
void MainWindow::on_pushButton_3_clicked() //cette fonction s'active sur l'appui sur le bouton musique
{
    ui->musique->show();    //permet d'afficher l'ecran musqiue
    ui->GPS->hide();        //permet de cacher l'ecran GPS
    ui->telephone->hide();  //permet de cacher l'ecran telephone
}
```

Créations des boutons de l'écran

Pour créer tous les boutons que nous avons utilisés, nous avons utilisé des push boutons. Une fois le bouton créer un menu du bouton est maintenant visible en bas à droite de l'écran. Ce menu possède plusieurs parties permettant de modifier le bouton comme bon nous semble. Pour l'insertion d'image dans le bouton, nous somme aller dans la partie arrêt normal puis en cliquant sur la petite flèche un explorateur de fichier s'ouvrent puis on sélectionne l'image de notre choix.

Utilisation des boutons

Avec un clic droit sur le bouton, sélectionnez "aller au slot" puis choisissez "clicked()". Cela permet de créer l'emplacement de la fonction dans le mainwindow.cpp, pour utiliser le bouton.

Bilan sur l'avancement de l'interface graphique

En ce qui concerne l'interface graphique, nous avons réalisé un interface graphique contenant 5 pages différentes :

- 1 page pour le gps
- 1 page pour la musique
- 1 page pour toute la partie téléphonique
- 1 page pour les paramètres généraux
- 1 page pour les informations du véhicules

Nous avons commencé le design de la page gps et musique en positionnant des boutons et de frame mais n'avons pas eu le temps de programmer les boutons présent dans les différentes pages. Nous avons aussi programmé l'affichage en temps réel de l'heure.

Le bus CAN

Une fois la partie interface graphique faite, Mr DURAND nous a proposé de réaliser la programmation d'un bus CAN sur le VMD. Le VMD est un Véhicule Multiplexé Didactique. Voici ci-dessous une image de la maquette du VMD

Présentation du bus CAN

Dans cette partie nous allons voir qu'est ce que le bus bus CAN

Dans les années 80, la demande de confort et le besoin de sécurité ne font qu'accroître. La société BOSCH développe dès le début des années 1980 une solution de multiplexage des informations circulant à bord de la voiture. Le bus CAN apparaîtra et sera normalisé durant l'année 1983. Le bus CAN se développe ce qui fait qu'on le retrouve dans d'autres secteurs de l'industrie tel que le médical, les produits numériques, les systèmes électrotechnique,etc …

Le bus CAN est composé de 2 câbles d'alimentation électrique 12V DC et 1 paire torsadée pour les transferts d'information.

Le bus CAN est composé d'une architecture simplifiée. En effet, le bus CAN est composé de 3 couches. On y retrouve la couche physique, la couche liaison de données et la couche applicative.

Création de la communication

Nous avons vu dans la partie précédente qu'est ce que le bus CAN, comment il est composé et pourquoi à t'il été créé. Maintenant nous allons faire l'étude de la maquette d'essai.

Dans un premier temps nous avons installé le shield raspberry pi can 2 sur la carte.

Ensuite nous avons installé la bibliothèque Canutils sur la raspberry ce qui nous a permis d'envoyer des trames assez facilement.

Nous avons suivie la notice pour l'installation:

https://www.thierry-lequeu.fr/data/PICAN2-UGB.pdf

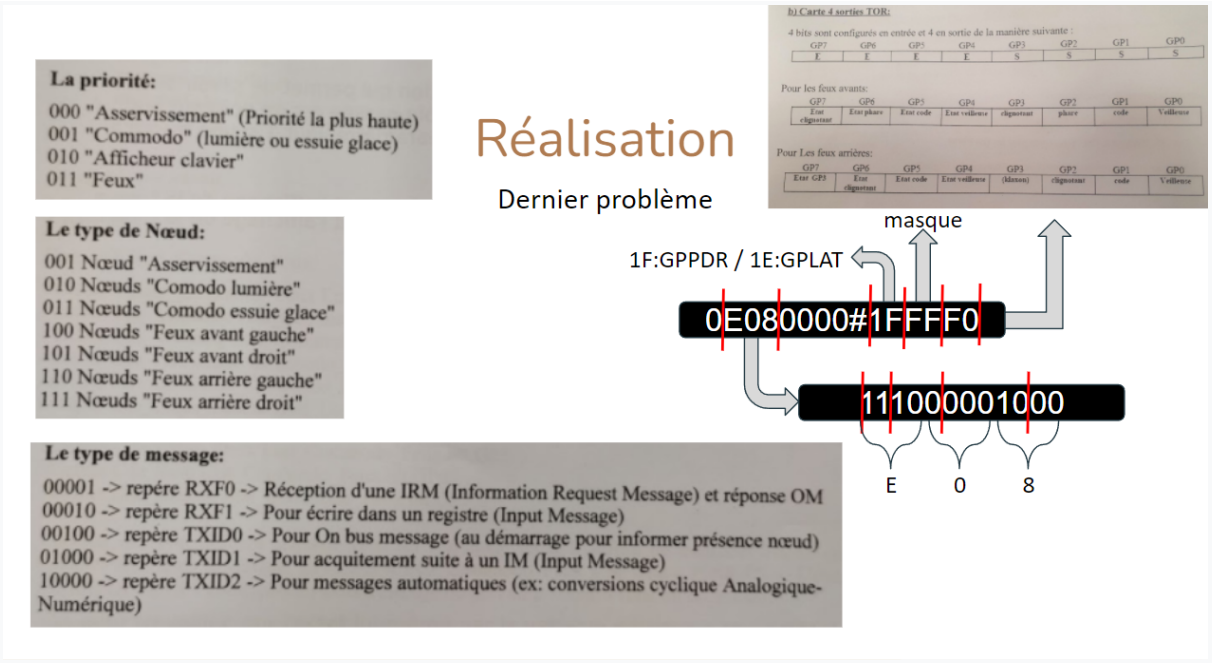
Pour vérifier si la trame partez bien nous avons utiliser le convertisseur CAN to USB avec l'application Cananalyser3

C'est à ce moment là que nous avons remarqué un problème avec la vitesse de transmission qui n'était pas la bonne pour la maquette de

simulation. Donc nous avons utilisé cette commande pour définir la vitesse.

```
sudo /sbin/ip link set can0 up type can bitrate 100000
```

Une fois que nous savions que les trames s'envoyaient, nous avons pu nous attaquer à quelle trame envoyer. Pour ça nous avons utilisé toute la documentation fournie. Cela nous a donner :



Donc pour allumer une led par exemple nous devons envoyer une première trame pour définir les entrées et les sorties:

0E080000#1FFFF0

Ensuite pour allumer les leds nous devons définir quelle led allumée donc nous envoyer une trame par exemple:

0E080000#1E0F0F

Cela allume donc toute les leds du bloc avec l'ID E08.

Ensuite pour associer cela avec les boutons sur QT j'ai décidé de créer des fichier exécutable en C (possibilité de directement écrire sur QT) avec le raspberry qui s'exécute avec l'appuis sur des boutons sur QT.

Conclusion

Malcolm : Durant ce projet, j’ai dans un premier temps étudier le fonctionnement de la Raspberry PI 4, savoir comment la connecter à l'écran, puis, avec Bastien nous avons galéré à installer QT sur le Raspberry. J’ai pu aussi mettre en pratique les compétences acquises en cours afin de réaliser des trames fonctionnelles sur le bus CAN. Ce projet m’a permis de découvrir comment réaliser une interface graphique.J’ai pu découvrir à travers ce projet comment réaliser la mise en place de l’éclairage de phare et de clignotants.

Bastien : Ce projet a été une bonne expérience et m'a permis de pouvoir améliorer mes compétences dans le domaine ou je souhaite me spécialiser (les systèmes embarqués automobile). De plus, j'ai pu améliorer ma capacité à m'adapter et à trouver des solutions a des gros problèmes comme l'installation de Qt sur le raspberry, la communication en BUS CAN avec la maquette de simulation. J’ai appris aussi à construire une interface graphique et à m'adapter au système embarqué de notre projet. Pour finir ce projet qui n'était à la base pas proposé dans les projets et qui est une demande faite par nos soins a plutôt bien fonctionné et permet d’avoir une suite pour nous et pour les prochain étudiants.

Poursuite de projet

Nous avons réalisé ce qui constitue les bases de ce projet. En effet, nous avons réussi à installer le logiciel d'interface graphique sur le Raspberry. Nous avons commencé à réaliser l'interface graphique de notre module et permis la communication avec le BUS CAN, mais il reste à faire le développement des différentes parties de l'interface (musique, gps, info du véhicule, paramètre généraux,...).

https://code.woboq.org/qt5/qtconnectivity/src/bluetooth/qlowenergycontroller_bluezdbus.cpp.html

Annexes

Installateur Qt Windows: https://www.qt.io/product/development-tools

Tuto Qt Creator : https://wiki.qt.io/QtCreatorWhitepaper

Adresse mail etudiant:

Bastien : bmbastien84@gmail.com / michel.bastien.pro@gmail.com

Malcolm : Malcolmgranaud6@gmail.com

Fichier du projet:

https://drive.google.com/drive/folders/1QD1dCkUWFlSUsftyCYObrVEoYrF9P6lY?usp=sharing