DASHBOARD

Interface homme-machine

pour un voilier

BASTIN - BONNEAU

BORDOUX - PRÊCHEUR

MEA 4

Table des matières

Table des matières **2**

Introduction **3**

Mise en contexte **3**

Acquisition et traitement **4**

Capteur 1 4

Capteur 2 5

Shield 5

Traitement des données **6**

Choix de l'OS 6

Interface IHM **6**

Conclusion **6**

Webographie **6**

Introduction

Dans ce projet, nous avions pour but de produire une aide à la navigation sur un voilier. L'objectif était de recevoir des données à l'aide de capteurs et les afficher sur un tableau de bord, de la manière la plus intuitive possible.

Ce projet s'est divisé en plusieurs partie:

- Raphaël s'est occupé de la réception des différentes données nécessaires par capteurs
- Coline et David ont travaillé sur la configuration de la Raspberry
- Oscar a travaillé sur la partie graphique du dashboard, l'affichage des informations

Mise en contexte

Dans le monde marin, il peut être difficile de se repérer. de nombreux outils existent pour aider les usagers des différents type d'embarcations.

Afin de faciliter encore plus ces usagers, le CEM , Centre d'Entrainement Méditerranée, nous a fait la demande de la création d'un dashboard centralisant plusieurs informations concernant le bateau tel que les positions, accélérations, orientations, sa position GPS et de ce qui l'entoure, comme la force et direction du vent.

I. Acquisition et traitement

Dans un premier temps, il nous a paru pertinent d'acquérir des informations aussi classique qui importantes pour un voilier. Nous avons donc voulu connaître la position et l'orientation. Nous avons donc utilisé deux capteurs : un IMU gui regroupe les données gyroscopiques, magnétométriques et accélérométriques, ainsi que d'un GPS pour la positions du voilier. Cependant, l'objectif du projet était de créer une interface intuitive, mais aussi modulable où il serait possible d'ajouter des capteurs.

A. Capteur 1: IMU

Module boussole BNO055

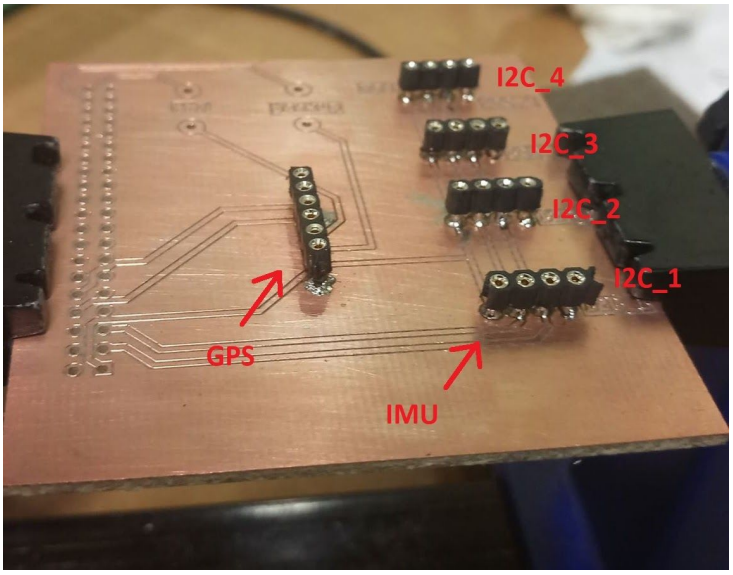
Ce module comporte un accéléromètre, un gyroscope et une boussole. Notre Raspberry ne comptant qu'un port serial, nous avons choisi de faire communiquer le module et la RAspberry en I2C.

B. Capteur 2: GPS

Module GPS PA6H

Comme son nom l'indique ce module est un GPS, qui nous donne la localisation du bateau à tout instant. Ce module communique en UARTH.

C. Shield



PCB servant de shield notre Raspberry

Comme on peut le voir ci-dessus, le shield nous permet de rendre le dashboard modulable en ajoutant, ou modifiant les capteurs connectés à la Raspberry (partie gauche du PCB). Les potentiels capteurs seront branchés en I2C, en parallèle, relié aux pins SCL et SDA de la Raspberry qui contiennent déjà le montage Pull-up dont l'I2C à besoin pour garder un niveau haut lorsque le bus n'est pas utilisé.

II. Traitement des données

A. Choix de l'OS

Pour ce projet nous avions vu deux systèmes d'exploitation possible:

La première idée fut d'utiliser une STM, l'utilisation des capteurs aurait été assez aisé mais l'interface graphique bien moins

La deuxième idée, celle que nous avons gardée, une Raspberry. Ayant un port HDMI, l'installation de l'interface graphique était instantanée. Pour ce qui est des capteurs, de nombreuses librairies existent ce qui facilite leur utilisation.

C'est donc vers la Raspberry que nous nous sommes tourné, plus particulièrement le modèle Pi 3 B.

B. Récupération des données

La première chose que nous faisons et de récupérer le squelette du code nécessaire à l'extraction des données pour capteurs. Dans notre cas, nous nous somme basé sur le site suivant : https://learn.adafruit.com/bno055-absolute-orientation-sensor-with-raspberry-pi-and-beaglebone-black/software

*Une fois ce squelette récupéré, nous importons nos librairies utiles pour la suite :

```
import socket
import random
import struct
import math
import time
```

Ensuite, nous allons connecter cette récupération de données à une socket de notre premier programme de l'IHM.

```
HOST = '127.0.0.1'  # The server's hostname or IP address
PORT_1 = 65432        # The port used by the server
PORT_2 = 65434
s1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print('Connected by', s1)
print('Connected by', s2)

s1.bind((HOST, PORT_1))
s1.listen(1)
conn_1, addr_1 = s1.accept()

s2.bind((HOST, PORT_2))
s2.listen(1)
conn_2, addr_2 = s2.accept()
cpt = 0
print('Connected by', addr_1)
print('Connected by', addr_2)
```

Enfin, nous envoyons nos valeurs en continue dans la socket.

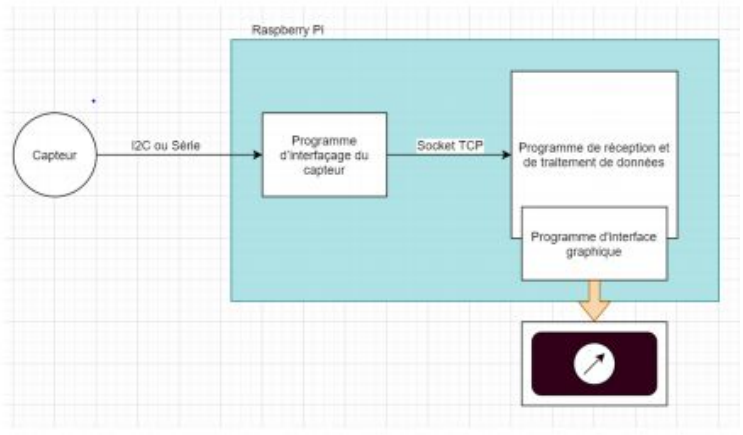
```
while True :

    val= 180*math.sin(cpt/10)

    # sender side
    data_1 = struct.pack('f', val)  # float -> bytes
    data_2 = struct.pack('f', val+90)
    cpt = cpt+1
    print('Connected by', conn_1)
    print('Connected by', conn_2)
    try:
        conn_1.sendall(data_1)
        conn_2.sendall(data_2)
    except :
        print('Not sent')
    time.sleep(0.1)
```

III. Interface IHM

A. Fonctionnement général

Ce système se voulant le plus modulaire possible, il est composé de plusieurs parties. Naturellement, un principe de programmation orientée objet est utile. Nous avons donc créé un interface graphique en utilisant la librairie Qt proposant de nombreuses fonctionnalitées.

L'architecture de notre interface sera donc la suivante :

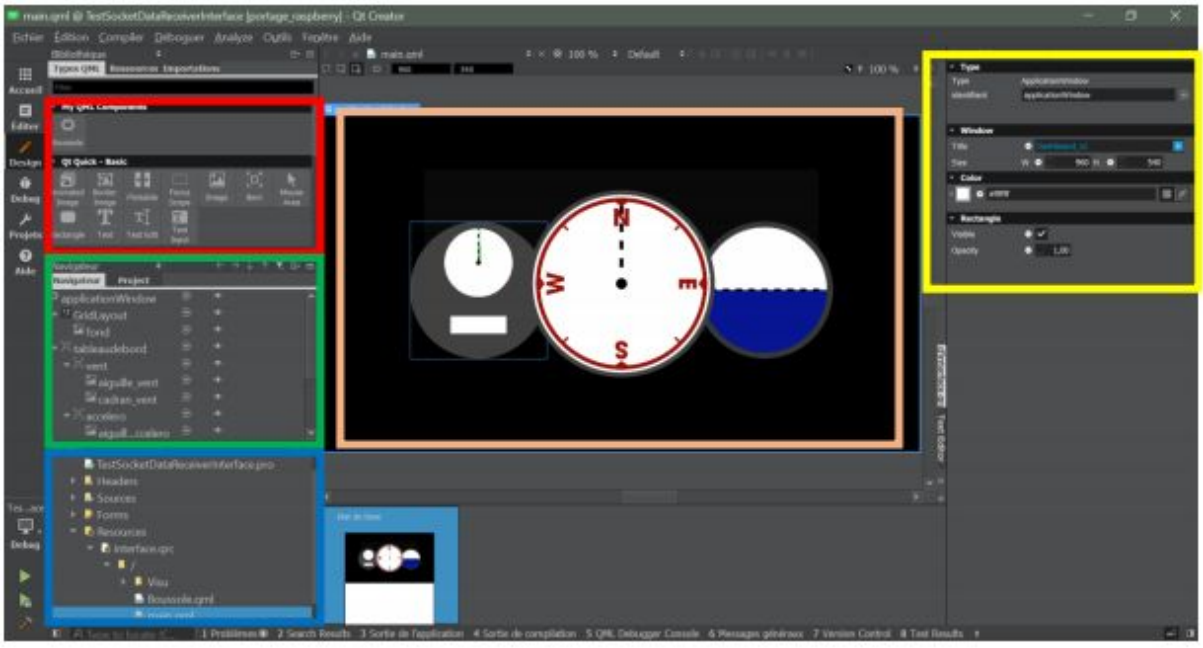


Le système est alors composé de 3 programmes. Le premier est propre au capteur et s'occupe de l'interface entre le capteur et la Raspberry. Le deuxième réceptionne les données des capteurs envoyées au premier programme et traite ces dernières. Enfin, le dernier programme récupère les données et a pour but d'animer les images selon les données réceptionnées.

B. Programme de génération d'interface

Comme dit précédemment, ce programme a pour but d'animer les images de notre écran selon les données obtenues de nos capteurs. Ce programme est un programme QML (Qt Markup Langage). C'est un langage déclaratif d'interface utilisateur qui est directement lié à une interface de développement graphique QTDesigner permettant de décrire nos fenêtres de manière graphique.

Notre fenêtre QTDesigner se présente alors comme ceci :



Lors de la modification de la fenêtre dans cette interface, le code QML va être généré automatiquement, il n'y a donc rien à faire sur ce code mise à part l'animation des images.

Conclusion

Au terme de ce semestre, nous avons conçu un prototype qui permet de se localiser et de s'orienter (boussole et inclinaison). Nous avons pu afficher les données sur un écran pour un rendu graphique dynamique. La présence de connecteurs sur le Shield permet un ajout de capteur en I2C. De plus, grâce à une communication en réseau (les sockets), il nous est possible d'ajouter d'autres capteurs qui ne devront pas forcément être reliés physiquement à la Raspberry.

Webographie

Dashboard RP3:
https://www.networkshare.fr/2015/05/tuto-dashboard-raspberry-pi-dashing/

Doc librairie serial python:
https://pythonhosted.org/pyserial/shortintro.html

Capteur :

centrale inertielle
https://www.gotronic.fr/art-module-boussole-bno055-27795.htm

GPS PA6H :

https://cdn-shop.adafruit.com/datasheets/GlobalTop-FGPMMOPA6H-Datasheet-V0A.pdf

Raspberry PI3 :

https://projetsdiy.fr/decouverte-test-configuration-raspberry-pi-3/
https://www.raspberrypi.org/documentation/configuration/
https://www.raspberrypi-france.fr/guide/

Raspberry Pinout :
https://fr.pinout.xyz/pinout/pin27_gpio0

Aide à la configuration des capteurs:
https://github.com/themrleon/RpiSoft-UART
https://github.com/dwelch67/raspberrypi/tree/master/uart01