Dispositif de monitoring peropératoire avec envoie de données sans fils

Unité :
Project medical device II

Auteur :
AMADOU Bilal 17060

11/11/2021

Table des figures

1 Salle d'opération 3
2 Paramètres à surveiller 4
3 Dispositif global 4
4 Circuit de commande de la pompe 5
5 Circuit de contrôle (montage NPN) 6
6 Dispositif de mesure de la pression 6
7 Câblage du saturomètre 7
8 Affichage des paramètres vitaux en temps réel 7
9 Remplacement des résistances 8
10 Boutons de l'esp32 8
11 Connection Bluetooth 9

Table des matières

1 Introduction **3**
1.1 Objectif 3
1.2 Cahier des charges 4

2 Circuit de mesure de la tension artérielle **5**
2.1 Pneumatique 5
2.2 Acquisition de la pression 6

3 Circuit d'acquisition de la saturation et de la fréquence cardiaque **7**
3.1 Câblage 7
3.2 Software 8
3.2.1 Commandes pour l'esp32 8
3.2.2 Commandes pour la Raspberry 9

1 Introduction

1.1 Objectif

Dans le cadre du projet medical device , il m'a été de demandé de prototyper un appareil de monitoring des paramètres vitaux doté d'une transmission sans fil.
En effet, certaines opérations nécessitent une rotation de la table à 180° ce qui mène très régulièrement à un enchevêtrement des câbles.

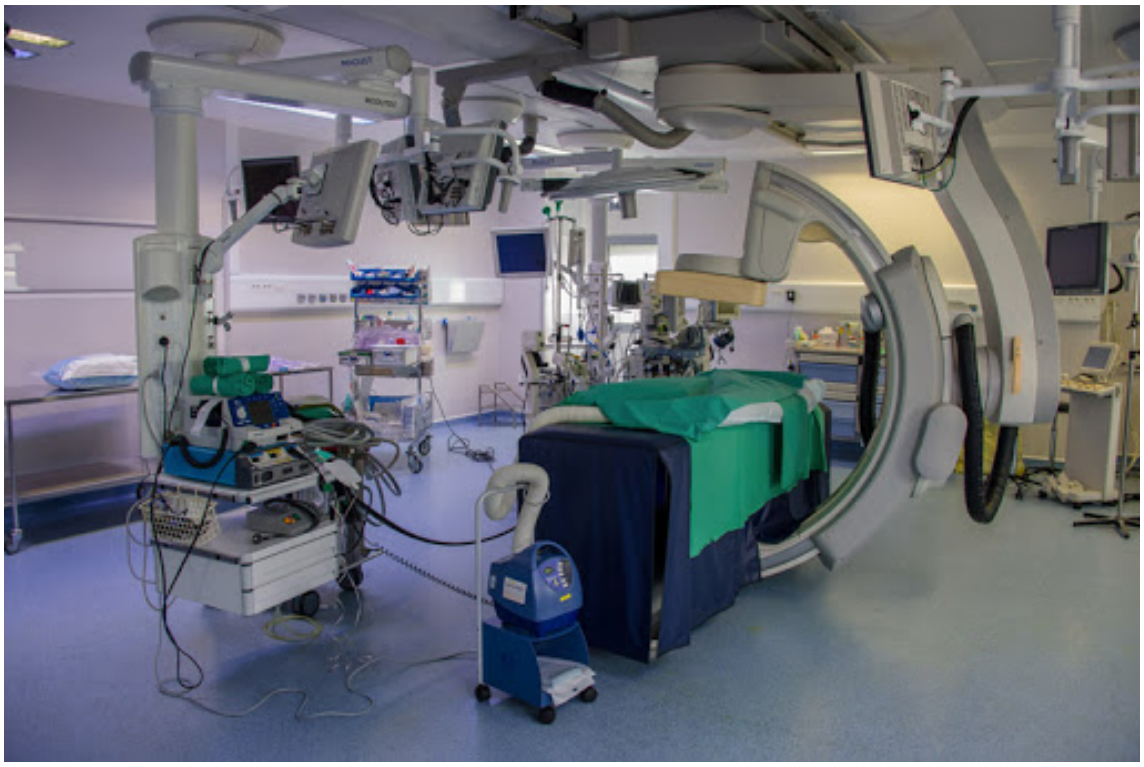

FIGURE 1 – Salle d'opération

Le présent document consigne des schémas et des explications des différents choix techniques qui ont été pris au long du projet.

1.2 Cahier des charges

L'objectif est de mesurer les 4 paramètres vitaux du patient (cfr. Figure 2). Pour ce faire nous devons concevoir un brassard et un saturomètre.

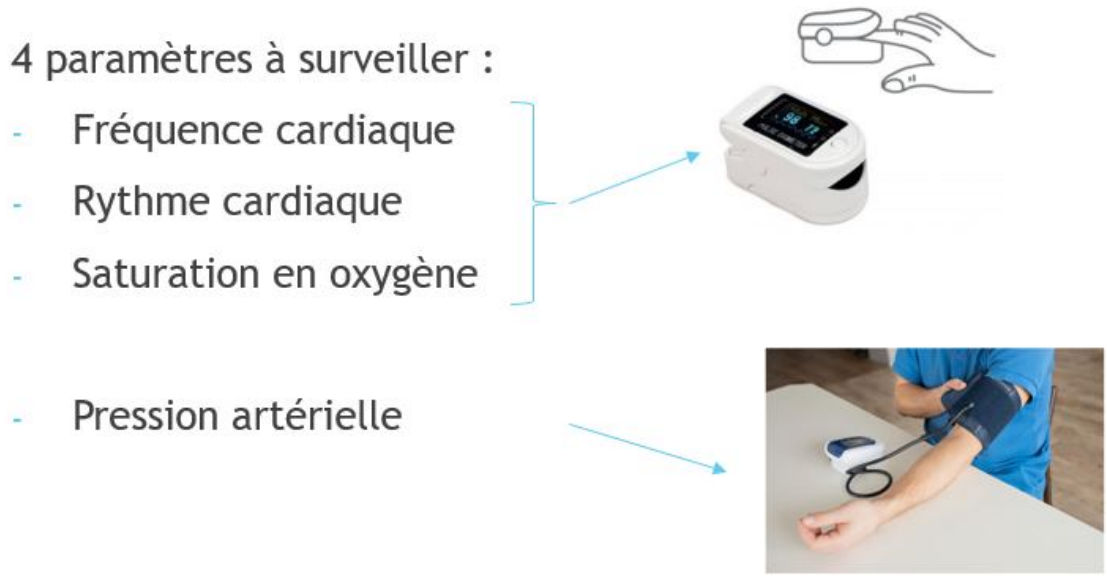

FIGURE 2 – Paramètres à surveiller

Ces deux circuits devront ensuite communiquer ensemble et envoyer l'ensemble des données à la centrale de monitoring (que l'on simulera avec une Raspberry pi).

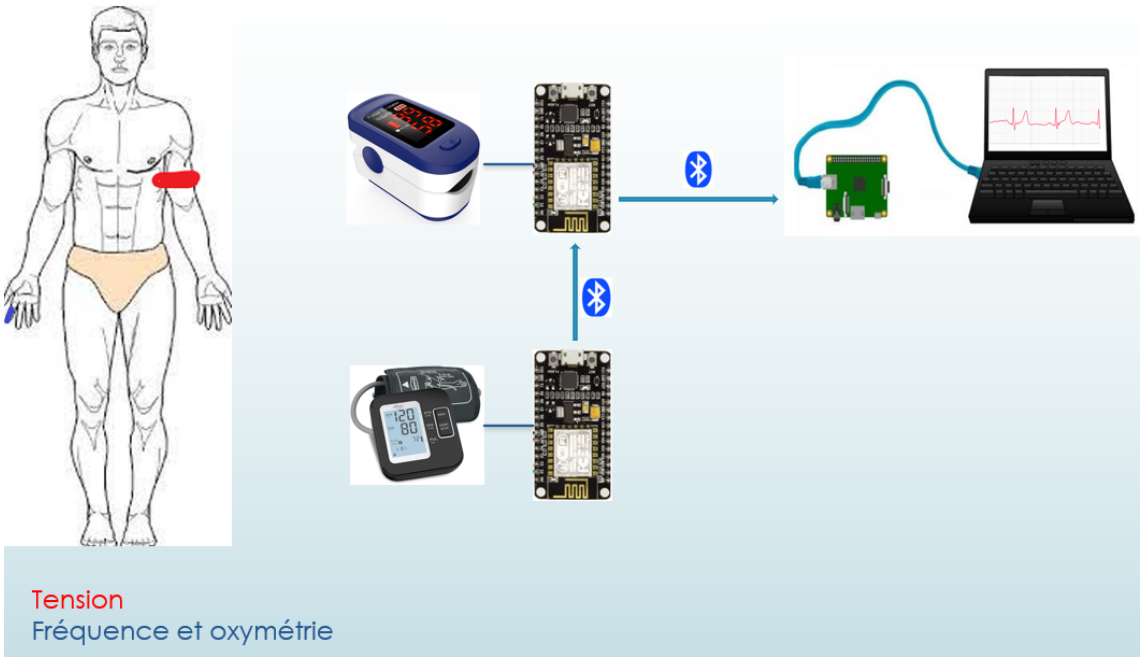

FIGURE 3 – Dispositif global

2 Circuit de mesure de la tension artérielle

2.1 Pneumatique

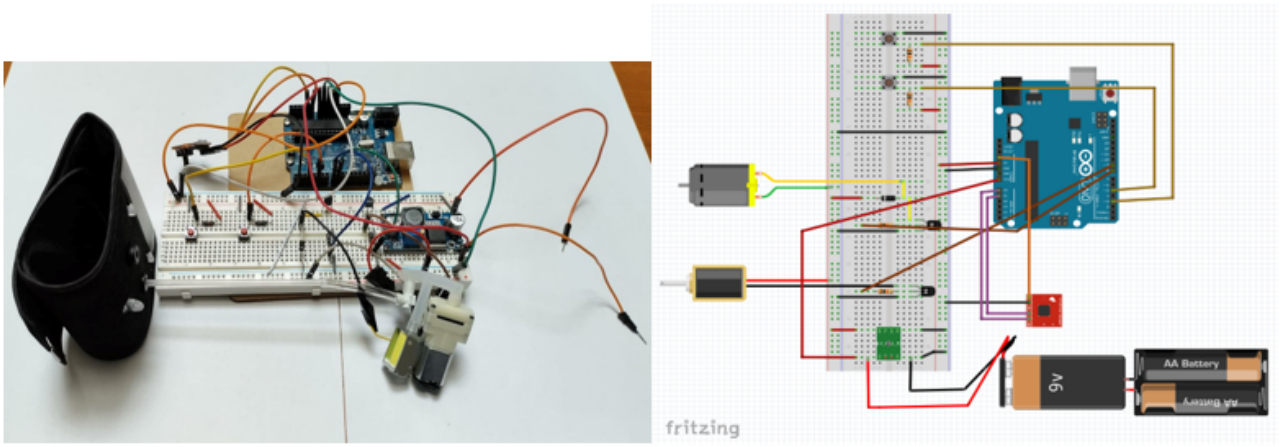

FIGURE 4 – Circuit de commande de la pompe

Le moteur et la valve ne sont pas alimentés par l'Arduino car celui-ci ne fournit pas assez de courant pour les faire fonctionner.
Nous utilisons donc une alimentation externe capable de fournir l'ampérage suffisant ainsi que 5V.

Dans le cas où nous brancherions l'Arduino à un pc (e.g. pour déboguer), la puissance doit provenir de l'alimentation externe.
Pour alimenter un Arduino par son port Vin, il faut y placer une tension comprise entre 7 et 12V.
L'alimentation est donc réglée sur 12 V (branché directement sur l'Arduino) et un étage de conversion permet d'alimenter le reste du circuit en 5V.

L'activation de la pompe et de la valve est assurée par 2 boutons. Lorsque ceux-ci sont appuyés, le signal en provenance de l'Arduino rend les transistors passants ce qui laisse le courant circuler.

Des résistances ont été ajoutées pour protéger les transistors.
En vertu de la loi d'Ohm : $R = \frac{U}{I}$. Le transistor génère une chute de tension de 0.7 V ; U = 4,3V.
Le transistor possède un gain β (dans notre cas 300) tel que $i_c = \beta * i_b$. Le moteur ayant besoin de 200 mA (courant du collecteur) pour fonctionner, le courant de la base vaut environ 666 μA.
La résistance doit donc avoir une valeur approximative de 7 $k\Omega$.

Le calcul précédent permet d'identifier la limite à partir de laquelle le transistor devient passant. Pour nous assurer que le transistor est suffisamment polarisé, nous utilisons des résistances de 2 $k\Omega$.

La diode est une diode de roue libre évitant un courant de retour en cas de coupure de courant.
L'electro-vanne requiert le même ordre de grandeur de puissance, le calcul et le montage restent

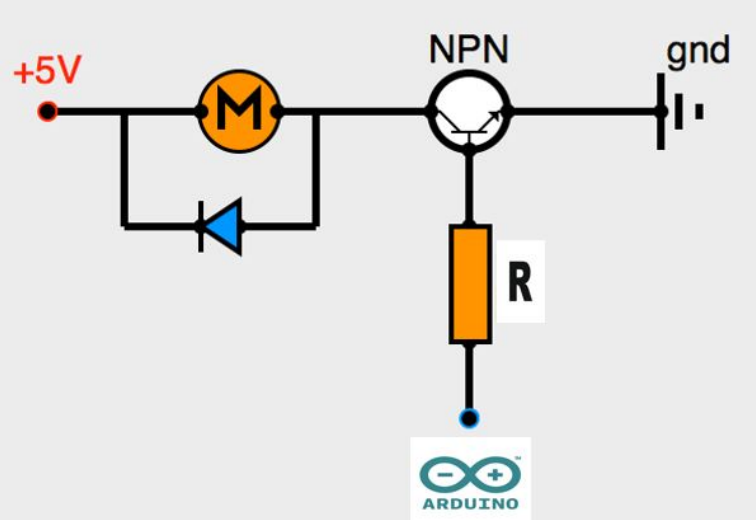

FIGURE 5 – Circuit de contrôle (montage NPN)

donc identiques si ce n'est que la diode de roue libre n'est pas nécessaire.

2.2 Acquisition de la pression

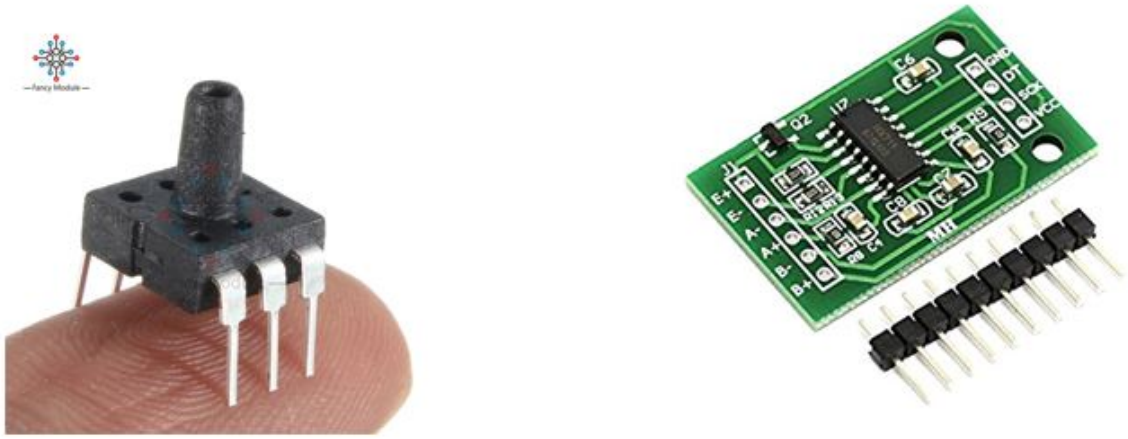

FIGURE 6 – Dispositif de mesure de la pression

La pression est mesurée à l'aide d'un capteur MPS20N0040D-S (cfr gauche de la Fig 6) et est ensuite traitée par un convertisseur analogique-numérique 24 bits HX711 (cfr droite de la Fig 6).

Il n'y à l'heure actuelle aucun capteur de pouls. Il sera nécessaire d'en ajouter un afin d'identifier les pressions artérielles systolique et diastolique.

Quand les bonnes tensions seront identifiées, il faudra ajouter un système de transmission sans fils.

3 Circuit d'acquisition de la saturation et de la fréquence cardiaque

3.1 Câblage

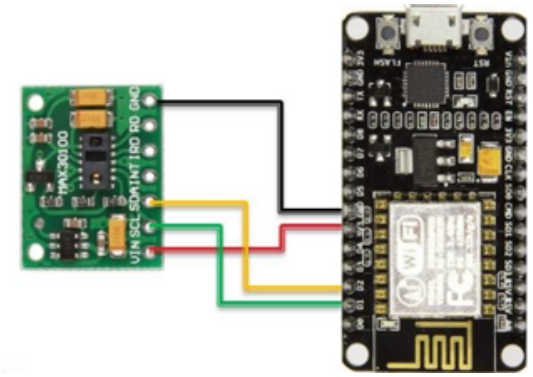

FIGURE 7 – Câblage du saturomètre

Le saturomètre est constitué d'un capteur émettant de la lumière rouge (max30100) relié à une carte électronique dotée d'un module Bluetooth (esp32).

L'esp32 envoie les données sur le port série d'une Raspberry pi ce qui permet de les récupérer afin de les plotter en temps réelle.

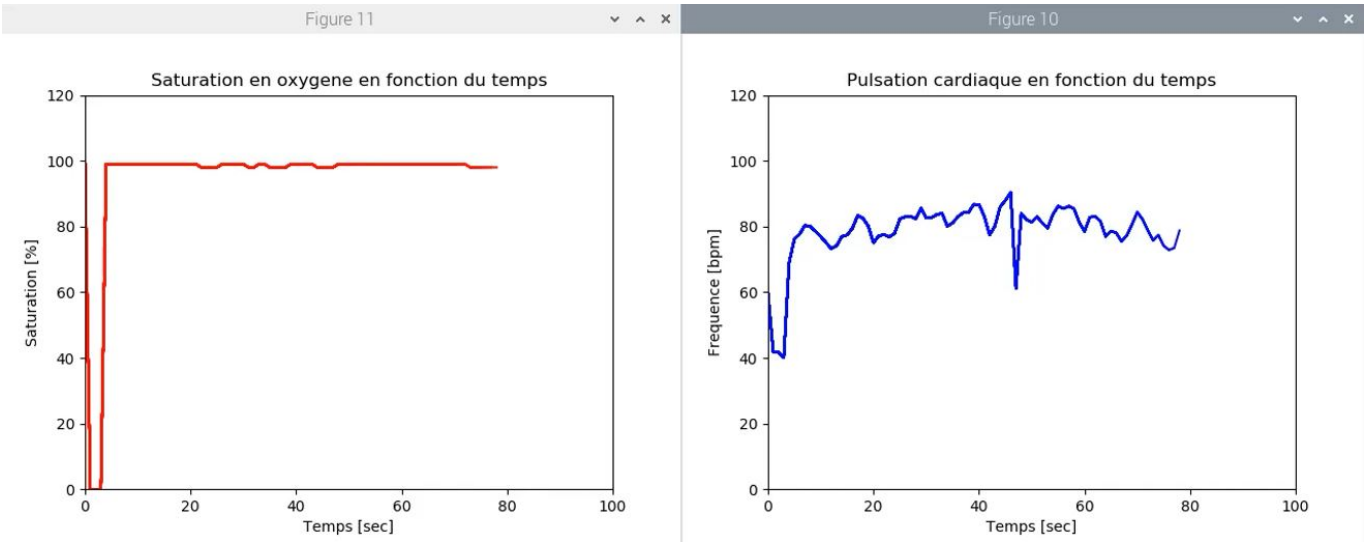

FIGURE 8 – Affichage des paramètres vitaux en temps réel

En raison d'un problème de conception, il a été nécessaire de déssouder les résistances intégrées au circuit et de les remplacer par 2 resistances de $5k\Omega$.

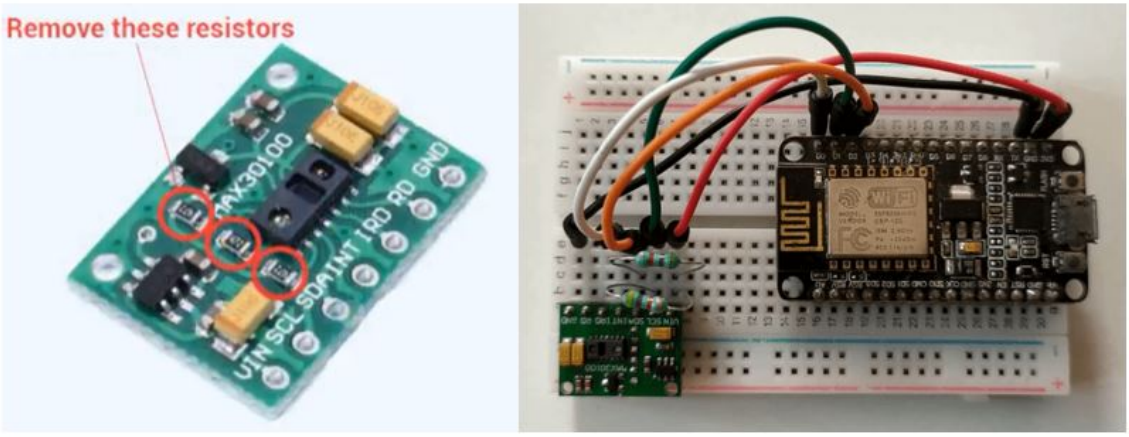

FIGURE 9 – Remplacement des résistances

3.2 Software

Deux programmes doivent tourner en parrallèle, le programme sur l'esp32 lit les données du capteur et les envoie sur le port série de la Raspberry pi. Un second programme lit le contenu du port série et l'affiche sur un graphe en temps réel.

3.2.1 Commandes pour l'esp32

On téléverse le programme grâce à un câble micro usb permettant le transfert de données. On appuie ensuite sur **boot** pour flasher la mémoire de la carte et sur le bouton **enable** pour lancer le programme en mémoire.

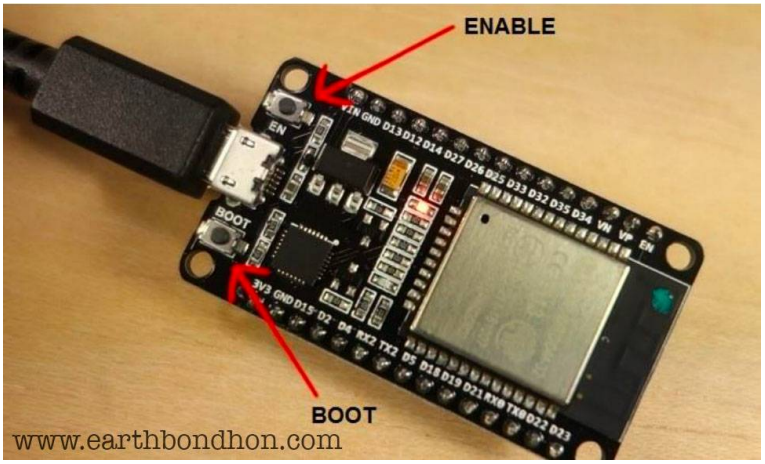

FIGURE 10 – Boutons de l'esp32

3.2.2 Commandes pour la Raspberry

On commence tout d'abord par connecter l'esp32 en Bluetooth. Il faut bien noter le port sur lequel nous sommes connectés.

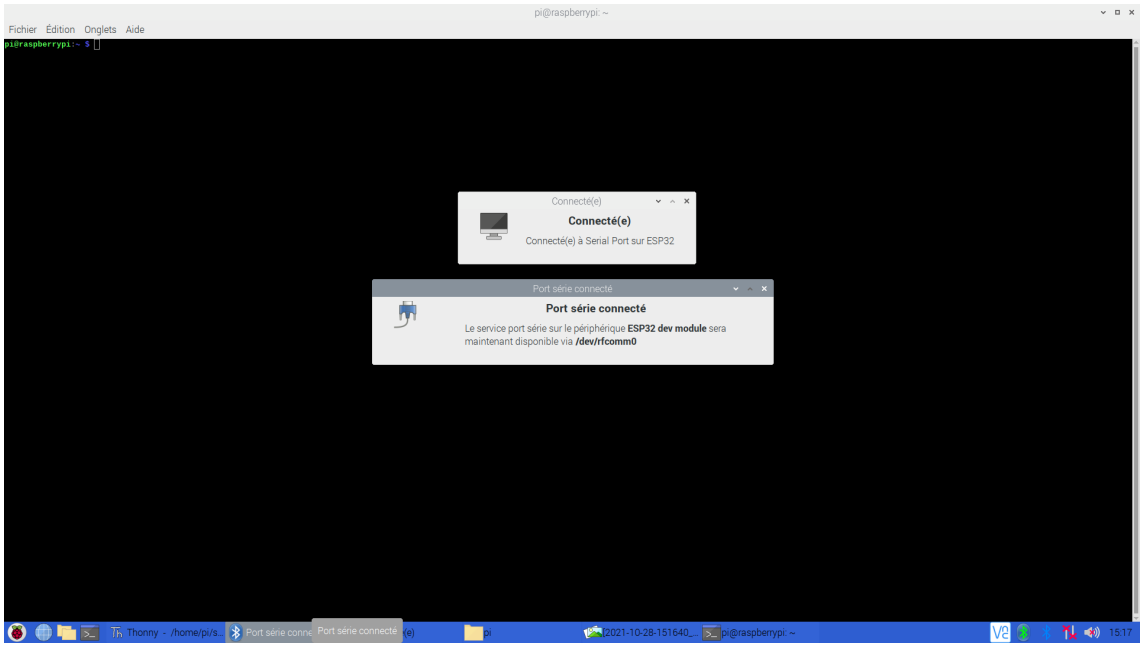

FIGURE 11 – Connection Bluetooth

Le programme de lecture du port est dans le fichier *serial*, on y accède via :

$ nano serial/serial_read.py

et on modifie la ligne

port = 'dev/rfcomm0'

Pour lancer l'acquisition des données et les enregistrer dans un fichier *.txt* on utilise la commande pipe (>)

$ sudo python serial/serial_read.py > /home/pi/Desktop/data.txt

Pour lancer un affichage en temps réel on utilise la commande

$ sudo python serial/rfcomm_plot.py