Centre professionnel du Nord vaudois
Filière d'automatique et d'électronique
Modules complémentaires techniques

CPNV - Centre Professionnel du Nord Vaudois

MCT - Modules complémentaires techniqeus

Communication interne

P2213

Rédacteur : Quentin Surdez

Relecture : Rafael Dousse

École : CPNV

Date : Yverdon-Les-Bains, le 31 mai 2022

Table des matières

Table des figures

1 Introduction

Ce document a pour but d'expliquer et de montrer notre évolution dans la communication interne du robot. La communication interne est la communication entre le Raspberry PI et l'Arduino Nano. Nous allons en premier lieu discuter des différents protocoles que nous avons pu essayer. Ensuite, nous parlerons de son intégration dans le projet pour chaque protocole. Nous verrons dans la conclusion notre choix final et nous argumenterons ce choix.

2 Construction

La construction du protocole s'est faite de manière organique. Nous avons regardé plusieurs tutos sur internet pour nous familiariser avec les communications entre le Raspberry PI et l'Arduino Nano Every. Ces éléments sont les seuls nécessitants une communication particulière au sein de notre projet.

2.1 UART

Nous avons commencé par nous intéresser au protocole de communication UART ou sérielle. Nous avons suivi un tuto de : aranacorp.com. Ce tutoriel nous a permis de prendre en main la communication.

Le code permettant de set up le Raspberry PI pour cette communication est quelque peu complexe. Le voici :

```
import serial,time
if __name__ == '__main__':

    with serial.Serial("/dev/ttyACM0", 9600, timeout=1) as arduino:
        time.sleep(0.1) #wait for serial to open
        if arduino.isOpen():
            #print("{} connected!".format(arduino.port))
            try:
                while True:
                    #cmd=input("Enter command : ")
                    arduino.write(cmd.encode())
```

FIGURE 2.1 – Set up du RPI pour comm UART

Nous pouvons observer que le set up est complexe et apparaît difficile à intégrer dans les différents programmes que nous allons développer par la suite. Nous devons tout d'abord spécifier le port sur lequel l'Arduino est connecté. Ensuite, il faut écrire la commande. Du côté de l'Arduino, la librairie Serial est utilisée.

Le plus grand désavantage de cette communication est le temps que prennent les informations pour être échangées. En effet, la vitesse de l'UART est définie par le $baudrate$ utilisé. Cependant, l'Arduino a un petit buffer et 9600 est la valeur par défaut. Donc la vitesse est de 9600bit/s. Qui plus est, la library pyserial est limitée à la lecture et l'envoi d'un byte. Des paramètrages peuvent être effectués pour en lire et en envoyer plus. Cependant cela impact la vitesse d'envoi et de réception.

2.2 I2C

En prenant connaissance des limites de la communication UART, nous nous sommes intéressés à la communication via le protocole I2C. Nous avons pu découvrir le code via un tutoriel : thegeekpub.com. Cela nous a permis de prendre en main le code à utiliser pour set up la communication du côté du Raspberry PI et de l'Arduino.

Voici le code du set up :

```
import smbus from SMBus

addr = 0x08
bus = SMBus(1)

while True:
        bus.write_byte(addr, 1)
```

FIGURE 2.2 – Set up de l'I2C du RPI

Nous pouvons observer que du côté du Raspberry le set up est rapide et simple. Du côté de l'Arduino la librairie Wire est utilisée. Voici le set up :

```
#include <Wire.h>

void receiveEvent(int howMany)
{while (Wire.available())
  {int c = Wire.read();
        switch (c)
    {case 1:
        Enavant();
      break;}}}
void setup()
{ Wire.begin(0x8);
  Wire.onReceive(receiveEvent);}
```

FIGURE 2.3 – Set up de l'I2C de l'Arduino

Nous pouvons voir que le set up en lui-même consiste de l'appel de deux fonction Wire.begin et Wire.onReceive. L'adresse doit être la même que celle donnée dans le Raspberry PI. Ensuite la fonction receiveEvent() permet de déterminer selon l'ordre reçu d'appeler telle ou telle fonction dans notre code. Cette fonctionnalité est détaillée dans le document "Explication Code Arduino".

La communication par I2C possède plusieurs avantages. Elle est plus rapide que l'UART, soit 100kbits/s. Elle est plus facile à intégrer dans les programmes que nous créons. Elle est flexible, c'est-à-dire que nous pouvons intégrer plusieurs Arduino slaves si nous en avons le besoin dans le futur.

3 Conclusion

Nous avons pu observer les différents protocoles testés dans le cadre de notre projet. Nous avons choisi le protocole de communication I2C. Les raisons pour ce choix sont les suivantes :

- Rapidité de la transmission de l'information
- Flexibilité pour ajouter des slaves qui communiqueraient avec le Raspberry
- Intégration facilitée dans les codes que nous créons et aucune incompatibilité avec le module de caméra ou OpenCV

Tous les codes incorporant une communication critique, possède le set up pour l'I2C du côté de l'Arduino et du côté du Raspberry PI.