

Embedded Systems Labs 2017 Fall

I2C

電機四 B03901124 李昂軒

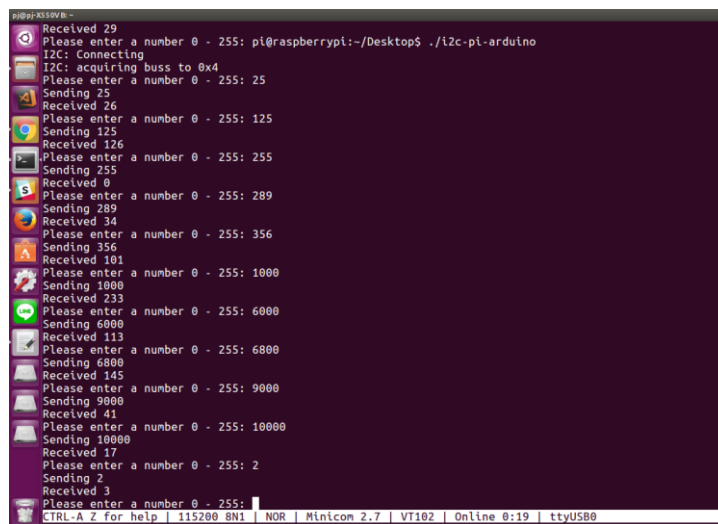
電機四 B03901134 袁培傑

✓ Asynchronous IO techniques:

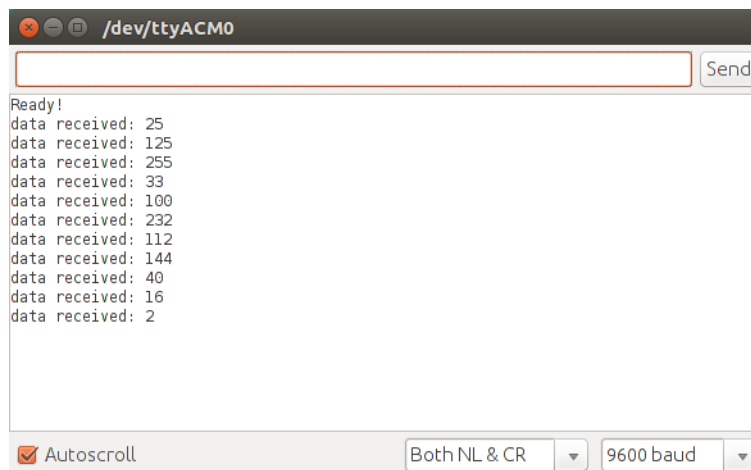
➤ Approaches:

由於要將 code 改成 Asynchronous，因此我們須將 readNumber 的 function，從 RPi 中原先的程式碼內的 while 迴圈裡取出來，額外放在外面。如果是 python 檔的話，則利用 GPIO 的 event_detect 來偵測，如果 Arduino 接收到數字的話，則會觸發 RPi 的 event_detect，藉此間接讀取從 Arduino 回傳回來的數值；如果是 c 檔的話，則使用 wiringPi.h 裡頭的函式，來呼叫 int main 外的 function 來間接讀取回傳值。

➤ Results:



```
pi@raspberrypi: ~  
Received 29  
Please enter a number 0 - 255: pi@raspberrypi:~/Desktop$ ./i2c-pi-arduino  
I2C: Connecting  
I2C: acquiring buss to 0x4  
Please enter a number 0 - 255: 25  
Sending 25  
Received 26  
Please enter a number 0 - 255: 125  
Sending 125  
Received 126  
Please enter a number 0 - 255: 255  
Sending 255  
Received 0  
Please enter a number 0 - 255: 289  
Sending 289  
Received 34  
Please enter a number 0 - 255: 356  
Sending 356  
Received 101  
Please enter a number 0 - 255: 1000  
Sending 1000  
Received 233  
Please enter a number 0 - 255: 6000  
Sending 6000  
Received 113  
Please enter a number 0 - 255: 6800  
Sending 6800  
Received 145  
Please enter a number 0 - 255: 9000  
Sending 9000  
Received 41  
Please enter a number 0 - 255: 10000  
Sending 10000  
Received 17  
Please enter a number 0 - 255: 2  
Sending 2  
Received 3  
Please enter a number 0 - 255:   
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Online 0:19 | ttyUSB0
```



```
/dev/ttyACM0  
Ready!  
data received: 25  
data received: 125  
data received: 255  
data received: 33  
data received: 100  
data received: 232  
data received: 112  
data received: 144  
data received: 40  
data received: 16  
data received: 2  
  
☒ Autoscroll Both NL & CR 9600 baud
```

➤ Discussions:

其實改寫成 Asynchronous 沒有太多困難，主要就照著講義上的改寫就好，不過還要額外注意，在改寫 c 的時候，makefile 也要一併改寫，不然會讀取不到 wiringPiSetup()...等函式。

✓ 3D accelerator:

➤ Approaches:

我們上網查詢了 ADXL345 的 data sheet，找到若是要調整其輸出頻率，則須在 address 0x2C 的位置處寫入不同數字，而 200Hz 對應到的則是 0x0B，因此寫法如下。

```
bus.write_byte_data(0x53, 0x2C, 0x0B)
```

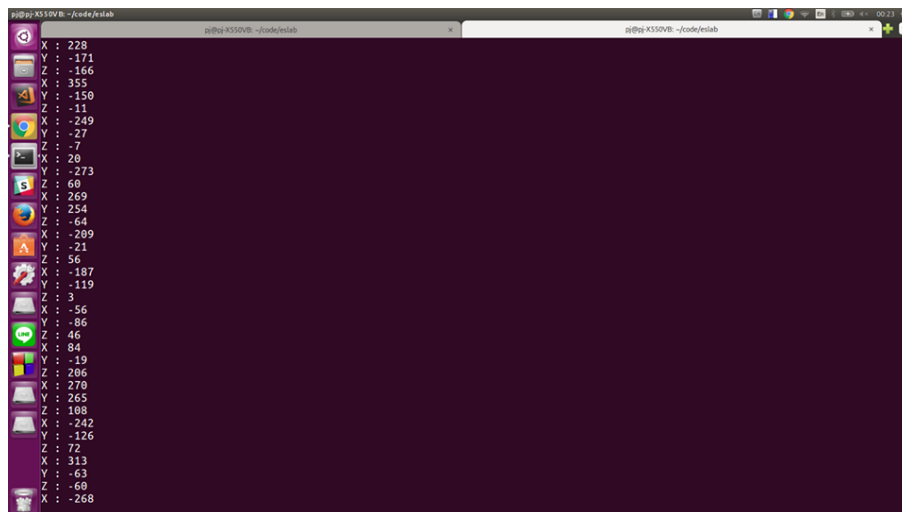
接著，若是要讀取 xyz 三軸之值，則須分別讀取 0x32 至 0x37 六個值，這六個值依次代表 x 值的 LSB、MSB，y 值的 LSB、MSB 以及 z 值的 LSB、MSB。

```
data0 = bus.read_byte_data(0x53, 0x32)
```

```
data1 = bus.read_byte_data(0x53, 0x33)
```

接著我們對每個軸的兩筆資料進行了一些處理，我們先將 LSB 跟 MSB 接成 16bits 的資料，再取後 10bits(因為我們有設定 ADXL345 的 output resolution 為 10 bits)。最後為了讓輸出平均分散在原點附近，我們將其範圍設定在-512 至+511 的區間，結果如下圖所示。

➤ Results:



➤ Discussions:

這次實驗比較麻煩的点就在于如何去調整其輸出的頻率，要先上網找尋相關的資料，才知道要如何去設定。而我們也有利用 python 的 time 功能做了一下實測，發現只要 0.6 秒多就可以跑完 200 次的偵測以及 output，不過我們最後還是以 data sheet 上的設定為主，因此沒有做調整。