

01OVEOQ-Innovation wireless platforms for the internet of things

Exercise: Exam Type 3 – Exercise 1

Candidate: Yenchia Yu (s287513)

Device: Pycom FiPy , Pycom PySense 2.0 X

1. Write the firmware code to switch on a LED on the FiPy

As the code in ./ex1-1-led/main.py shows, to switch on a LED on the FiPy, we first need to import library “pycom” and stop the heartbeat with `pycom.heartbeat(False)`. Then we can use `pycom.rgbled()` function with RGB color in hexadecimal format as parameter to switch on LED in different color.

2. Write the firmware code to connect with the FiPy via WiFi

1) Setup pybytes

To upload the firmware code via Wi-Fi, we need to use Pybytes platform (<https://pybytes.pycom.io>). Once we created an account on the platform, we can add devices and configure their network connections to the platform. Then, we can use “Pycom firmware updater” to update the firmware of FiPy and insert the activation token of pybytes during updating. After the update, FiPy can automatically connect to the WiFi we configured and also the pybytes platform.

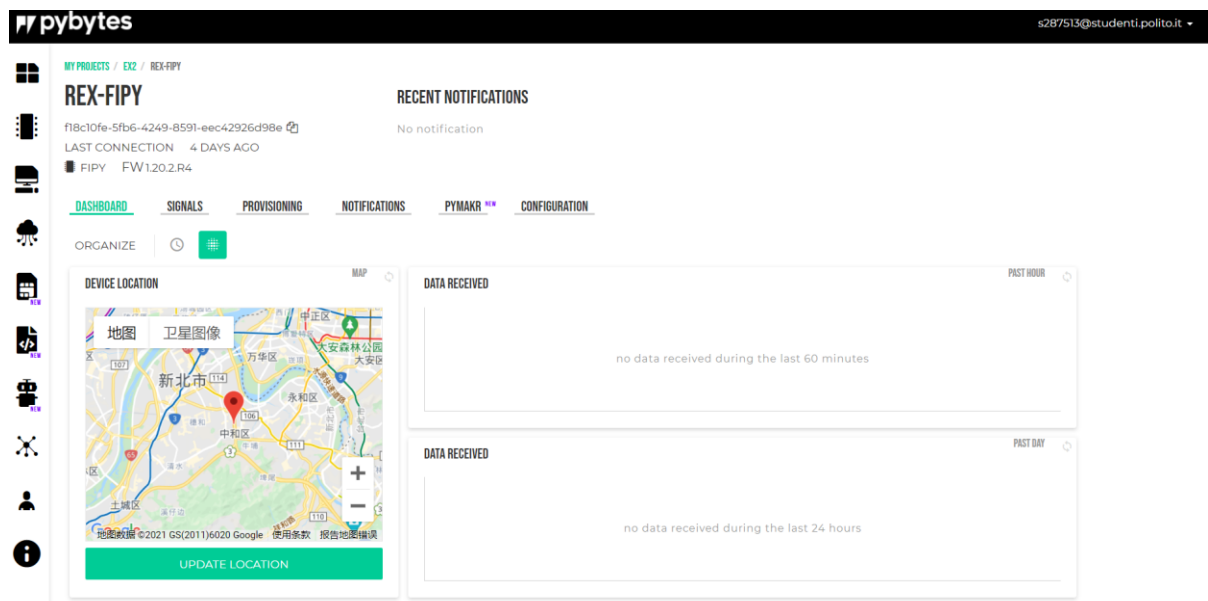


Fig.1 My FiPy device on Pybytes platform

2) Use online pymark to upload firmware code to switch LED

After previous configuration, we can use the online pymark platform to program our FiPy and upload the firmware code via WiFi. To realize the function to switch on LED with different colors in different WiFi connection status, we need to write our code in `boot.py` (As Fig.2 shows) since the autoconnection is started before `main.py`. With `boot.py` as Fig.2 shows, the LED on FiPy will switch on red color when the connection is not yet available. After the connection is established, the LED will switch on green color.

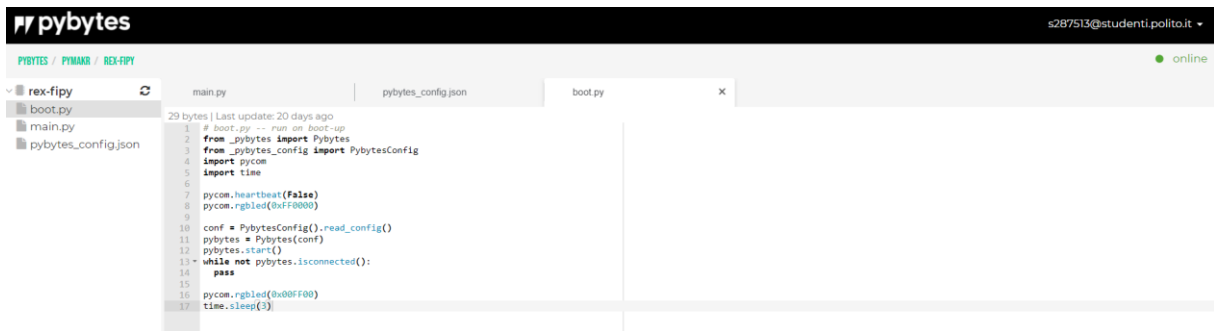


Fig.2 Online Pymark platform and boot.py code for ex2

3. Write the firmware code to read sensor/GPS/NFC data

In this exercise, I will read the data of Temperature and Humidity sensor and Accelerometer on PySense 2.0 X. To do so, library SI7006A20.py and LIS2HH12.py will be used.

- 1) Print the read data on a screen via wifi

To print data on a screen via WiFi as Fig.3 shows, `pybytes.send_signal()` function is used. It sends data to pybytes platform and display. To balance the workload on FiPy, the sending period is set to 10 seconds.

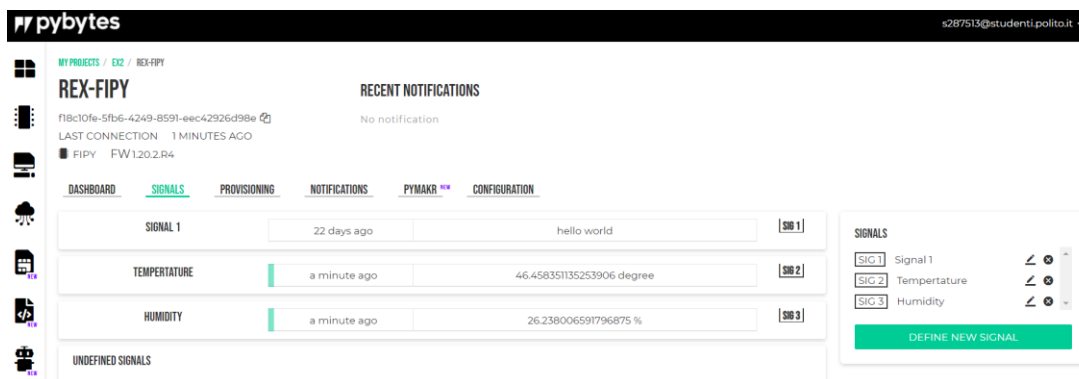


Fig.3 Print the read data on screen via WiFi

- 2) Print the read data on a screen via wifi

To print data on a screen via Bluetooth, we setup FiPy as a GATT server and run app “BLE scanner” on a smart phone as GATT client (as Fig.4 shows). To balance the workload on FiPy, the sending period is set to 5 seconds.

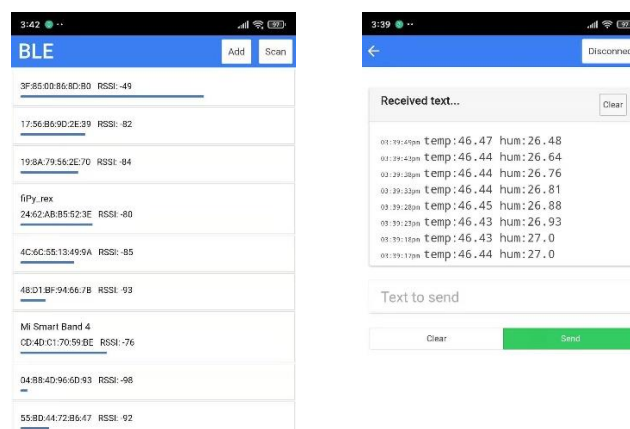


Fig.4 Print the read data on screen via Bluetooth

- 3) Switch on a Led selecting a color from a graduated scale associated to data

To switch on a Led selecting a color from a graduated scale associated to the sensor data, we define the color matrix as Table.1 shows. To balance the workload on FiPy, the updating period of LED color is set to 500 milliseconds, same as the updating period of sensor data.

Table.1 Color matrix to select LED color according to Accelerometer data

Range(°)	Pitch axis	Roll axis
(5,90)	bright green	bright red
(-90,-5)	dark green	dark red
[-5,5] and else	off	off

4. Write the firmware code to setup a radio link

In order to display the signal level with different LED color, I defined a function name “get_color(index)” which will calculate a color according to the input index. The range of index is [0,90]. When the value of index change from 0 to 90, the output color change from green to red. In the test of each radio link, the signal level rssi will be mapped to this range and generate corresponding LED color.

- 1) Wifi

With ./ex1-4-radio-link-wifi/main.py, FiPy will first scan the spectrum and print the results in pymark terminal. As Fig.5 shows, it found 4 APs. Then it connected to AP with ssid “Rex_YYJ” and continuously printing it’s signal level and channel and accordingly change the color of LED on FiPy.

```
Scanning the spectrum...
(ssid='Rex_YYJ', bssid=b'\x001\x92\xb7\xca\xaa', sec=3, channel=1, rssi=-49)
(ssid='andychu37', bssid=b'\x07\xd4j\x0c(', sec=3, channel=6, rssi=-78)
(ssid='\u8a0a\u865f\u826f\u597d', bssid=b'\x00\xad$\xbd\xa4\x9d', sec=3, channel=11, rssi=-90)
(ssid='22230517', bssid=b'\x14\t\xdc\xb1\x8e\xa8', sec=4, channel=11, rssi=-91)
Stop scanning.

Connecting to AP...
WiFi connected succesfully
('192.168.1.103', '255.255.255.0', '192.168.1.1', '192.168.1.1')
Updating AP's signal level and LED color...
rssi: -45 channel: 1
rssi: -46 channel: 1
```

Fig.5 Output of the test with WiFi

- 2) BLE

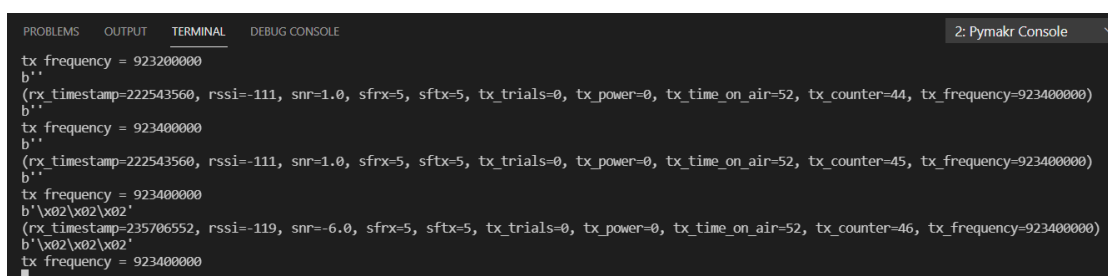
With ./ex1-4-radio-link-BLE/main.py, FiPy will first scan the spectrum and print the results in pymark terminal. In this test, I used app “BLE Tool” on smart phone to create a GATT server named “Rex”. As Fig.6 shows, it founds two different BLE advertisement. Then it connected to GATT server named “Rex” and continuously printing it’s signal level and accordingly change the color of LED on FiPy.

4) LoRa

To use LoRaWAN, we first need to register a network (here I chose The Things Network) and create our application and device on the platform to get the euis and key, which is needed when connecting to the gateways(OTTA). However, due to the TTN gateway coverage in Taiwan is not very high, there were some problems with connection at the beginning of the exercise. At last, I found an outdoor gateway in Nation Taiwan University (Located in: 25.015528,121.5383137) and successfully connected to the gateway.

Due to my test location is in Taiwan, the LoRa region is set to LoRa.AS923 and as Fig.9 shows, the initial tx_frequency is 923200000Hz. But after receiving the first feedback, the tx_frequency is adjusted to 923400000Hz. When the test location is around 300 meters away from the gateway, the connection was easy to establish and the rssi is around -110 dBm. But when the test location is around 500 meters away from the gateway, due to the blocking of buildings, the connection was difficult to established.

In the TTN console, we can monitor the data sent from FiPy to application(uplink, as Fig.10 shows) and also the data send from application to FiPy(downlink, as Fig.11 shows)



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 2: Pymkr Console
tx frequency = 923200000
b'\x02\x02\x02'
(rx_timestamp=222543560, rssi=-111, snr=1.0, sfrx=5, sftx=5, tx_trials=0, tx_power=0, tx_time_on_air=52, tx_counter=44, tx_frequency=923400000)
b'\x02\x02\x02'
tx frequency = 923400000
(rx_timestamp=222543560, rssi=-111, snr=1.0, sfrx=5, sftx=5, tx_trials=0, tx_power=0, tx_time_on_air=52, tx_counter=45, tx_frequency=923400000)
b'\x02\x02\x02'
tx frequency = 923400000
b'\x02\x02\x02'
(rx_timestamp=235706552, rssi=-119, snr=-6.0, sfrx=5, sftx=5, tx_trials=0, tx_power=0, tx_time_on_air=52, tx_counter=46, tx_frequency=923400000)
b'\x02\x02\x02'
tx frequency = 923400000
```

Fig.9 Output of the test with LoRa

APPLICATION DATA

pause

clear

Filters

uplink

downlink

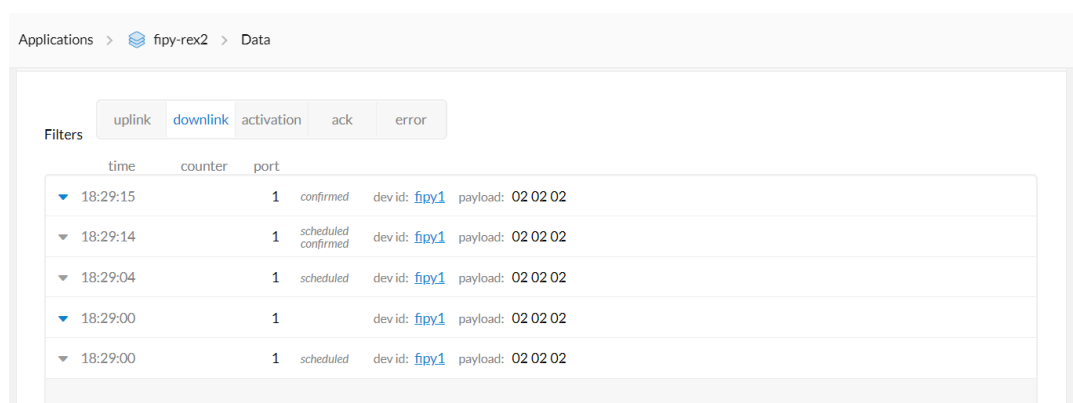
activation

ack

error

time	counter	port				
<div>⚡</div> 16:16:39			dev id: fipy1	dev addr: 26 01 65 DC	app eui: 70 B3 D5 7E D0 04 23 E5	dev eui: 00 17 58 83 75 3B 3C 44
<div>⚡</div> 16:15:48			dev id: fipy1	dev addr: 26 01 25 6F	app eui: 70 B3 D5 7E D0 04 23 E5	dev eui: 00 17 58 83 75 3B 3C 44
<div>⚡</div> 16:15:11			dev id: fipy1	dev addr: 26 01 22 D6	app eui: 70 B3 D5 7E D0 04 23 E5	dev eui: 00 17 58 83 75 3B 3C 44

Fig.10 Uplink data on TTN console



Applications > fipy-rex2 > Data					
Filters					
uplink downlink activation ack error					
time	counter	port			
18:29:15	1	confirmed	dev id: fipy1	payload: 02 02 02	
18:29:14	1	scheduled confirmed	dev id: fipy1	payload: 02 02 02	
18:29:04	1	scheduled	dev id: fipy1	payload: 02 02 02	
18:29:00	1		dev id: fipy1	payload: 02 02 02	
18:29:00	1	scheduled	dev id: fipy1	payload: 02 02 02	

Fig.11 Downlink data on TTN console