0623

【wireshark でのデータ確認】

今までデータを表示できなかった理由として, USB の COM が間違っていた.

正しくは COM7.

MONOSTICK が接続されていなくても，間違った COM で実行できるため，注意する.

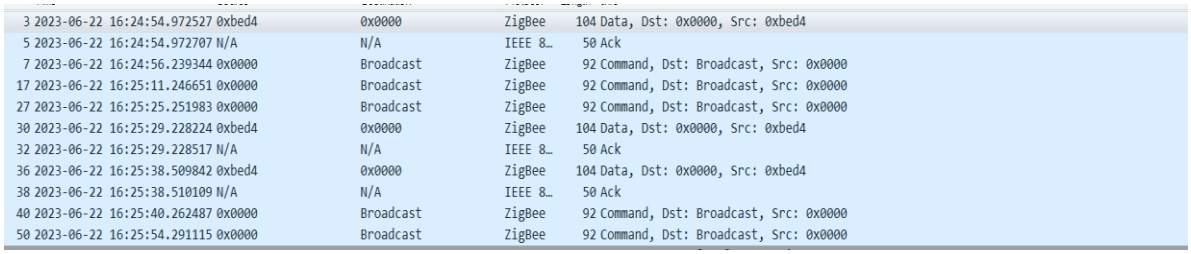
フィルターとして、

① 自身のネットワークの IP アドレスを示す 127.0.0.1 を指定

② ICMP を表示させないように「！」を前に付与.

Enddevice→Router→Coordinator のデータ送信を行った.

以下のように wireshark でデータの確認ができたため, Router を介していることが示せた.



3	2023-06-22 16:24:54.972527	0xbed4	0x0000	ZigBee	104	Data, Dst: 0x0000, Src: 0xbed4
5	2023-06-22 16:24:54.972707	N/A	N/A	IEEE 8…	50	Ack
7	2023-06-22 16:24:56.239344	0x0000	Broadcast	ZigBee	92	Command, Dst: Broadcast, Src: 0x0000
17	2023-06-22 16:25:11.246651	0x0000	Broadcast	ZigBee	92	Command, Dst: Broadcast, Src: 0x0000
27	2023-06-22 16:25:25.251983	0x0000	Broadcast	ZigBee	92	Command, Dst: Broadcast, Src: 0x0000
30	2023-06-22 16:25:29.228224	0xbed4	0x0000	ZigBee	104	Data, Dst: 0x0000, Src: 0xbed4
32	2023-06-22 16:25:29.228517	N/A	N/A	IEEE 8…	50	Ack
36	2023-06-22 16:25:38.509842	0xbed4	0x0000	ZigBee	104	Data, Dst: 0x0000, Src: 0xbed4
38	2023-06-22 16:25:38.510109	N/A	N/A	IEEE 8…	50	Ack
40	2023-06-22 16:25:40.262487	0x0000	Broadcast	ZigBee	92	Command, Dst: Broadcast, Src: 0x0000
50	2023-06-22 16:25:54.291115	0x0000	Broadcast	ZigBee	92	Command, Dst: Broadcast, Src: 0x0000

3 番目：Src の ShortAdress 0xbed4 は Enddevice である.

Dst の ShortAdress 0x0000 は Coordinator である.

次の画像で 64bit の ExtendAdress が表示されており、使用している Enddevice と確認済み.

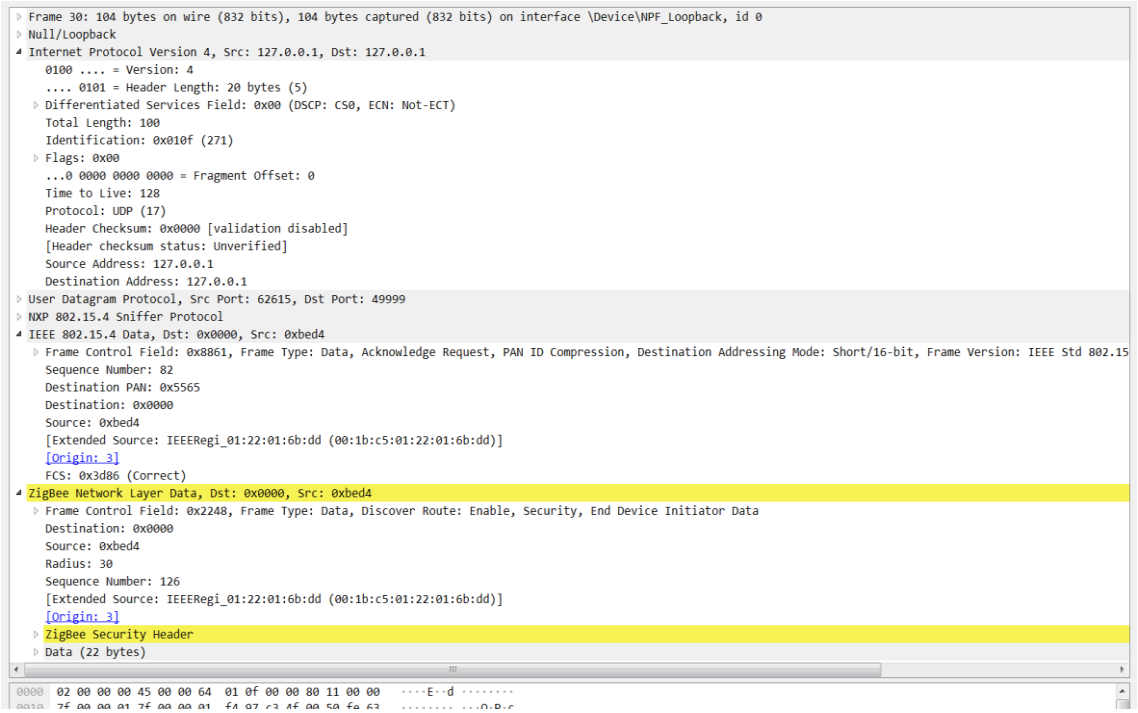
5 番目：そのデータの ACK を返している.

7 番目：Coordinator を示す 0x0000 がネットワークの電波を随時飛ばしている.

データ送信はプロトコルが Zibee と表示されているのに, ACK は IEEE なのか不明.

以下の URL は Zibee のフィルターをまとめている.

Wireshark · Display Filter Reference: ZigBee Application Support Layer



【スケジュール】

	スケジュール	実施したこと	できなかったこと	来週への課題
5/26 ~ 6/2	・JN5169にbeaconがないため JN5189を使用	・JN5189を検討結果使用しない ・pollコードを制御	・wiresharkの全般の理解	・JN5169を継続 ・E→Cの送信で検証 ・wiresharkでの確認
6/2 ~ 9	・E→Cでの送信を wiresharkで確認	・E→Cでの送信	・wiresharkでの正確な表示	・ArduinoかRaspberry Piを用いてAD変換を実施する.
6/9 ~ 16	・ラズパイの初期設定 ・フィルタありのwireshark	・ラズパイの初期設定 ・wiresharkのフィルタで表示内容を制限	・適切なフィルター表示	・wiresharkで指定の パケットのみを表示 ・ラズパイでデータ収集 ・AD変換のプログラミング (Pythonの予定)
6/16 ~23	・wiresharkでのデータ確認 ・PythonでのAD変換とUART通信プログラミング	・wiresharkでの送信データの確認 ・実際のセンサを用いた過程でのプログラム構築 ・UART通信を実現するプログラム構築	・プログラムの動作確認 (AD変換に必要なラズパイのチップが手元にないため)	・センサとチップを使用しプログラムの動作確認
6/23~30	・AD変換に必要なチップ (ADS1015)を実装 ・Pythonコードを実行			
6/30~7/7	・UART通信のプログラミング			

【ラズパイでの AD 変換】

AD 変換チップは発注済みで手元にない．届き次第、センサを用いてプログラムを実行する．

今回は Python での AD 変換をプログラミングした．

```python
import time
import Adafruit_ADS1x15

# ADS1015の設定をします
ads = Adafruit_ADS1x15.ADS1015()

# ゲインの設定です
# 指定によって、取得できる電圧の範囲が変わります

#  - 2/3 = +/-6.144V
#  -   1 = +/-4.096V
#  -   2 = +/-2.048V
#  -   4 = +/-1.024V
#  -   8 = +/-0.512V
#  -  16 = +/-0.256V
GAIN = 1

#
#「ゲイン1」の時の、「受信データ1単位」の電圧を計算します
#

# 「ゲイン1」の範囲はプラス・マイナス 4.096なので、8.192になります
RANGE = 4.096 * 2

# ADSは上記範囲を12bit(4096)で表すので、上記範囲を4096で割ると
# ADSのデータ「1」につき、電圧は0.002[V]になります。
# UNIT=単位
UNIT = RANGE / 4096
```

```python
# ループを100回繰り返します
for i in range(100):

    # ADSのアナログ入力ピンを指定します
    ads1015_pin  = 0

    data = 0
    volt = 0

    # ADS1015からデータを取得します
    data = ads.read_adc( ads1015_pin, gain=GAIN)

    # 取得データから電圧を計算します
    volt = data * UNIT

    print("受信データ：" + str(data))
    print("電圧       :" + "{:.3f}".format(volt))

    # 1秒待機します
    time.sleep(1)

print("done.")
```