0614

【センサデータを UART 受信】

1- Sensor
　↓
2- Raspberry
　↓(UART)
3- EndDevice-Monostick
　↓(wireless communication)
4- Coordinator-Monostick
　↓ (UART)
5- PythonProgram (pc)

今回は 4 - 5 の UART 通信を作成した.
また, RaspberryPi にあるプログラムも変更をした.
今までは Coordinator で受信した時に print して表示させていたが, print は割り込み処理になり, 無線通信の妨げになる. そのため, 別に python プログラムを用意し, そこで表示させるようにした.
以下が追加情報である.

(Coordinator)

```
*/

u16bytesread = PDUM_u16APduInstanceReadNBO(sStackEvent.uEvent.sApsDataIndEvent.hAPduInst,0,"a\x10",&Rxbyte);
SendBytes(Rxbyte, 2); //受信したデータをUart通信で送信する関数
```

```
//Coordinatorで受信したデータをUart通信で送信する関数
void SendBytes(uint8_t bytes[], uint8_t size)
{
    int i = 0;
    for (i=0; i < size; i++)
    {
        vAHI_UartWriteData(DBG_E_UART_0, bytes[i]);
        // Check that the TX buffers are empty
        while((u8AHI_UartReadLineStatus(DBG_E_UART_0) & E_AHI_UART_LS_THRE) == 0) ;
        while((u8AHI_UartReadLineStatus(DBG_E_UART_0) & E_AHI_UART_LS_TEMT) == 0) ;
    }
}
```

vAHI_UartWriteData

void vAHI_UartWriteData(uint8 *u8Uart*, **uint8** *u8Data*);

Description

This function writes a data byte to the Transmit FIFO of the specified UART. The data byte will start to be transmitted as soon as it reaches the head of the FIFO.

If no flow control or manual flow control is being implemented for data transmission, the data in the Transmit FIFO will be transmitted as soon as possible (irrespective of the state of the local CTS line). Therefore, the function **vAHI_UartWriteData()** should be called only when the destination device is able to receive the data.

For UART0, if automatic flow control has been enabled for the local CTS line using the function **vAHI_UartSetAutoFlowCtrl()**, the data in the Transmit FIFO will only be transmitted once the CTS line has been asserted. In this case, **vAHI_UartWriteData()** can be called at any time to load data into the Transmit FIFO, provided that there is enough free space in the FIFO.

Refer to the description of **u16AHI_UartReadTxFifoLevel()** or **u8AHI_UartReadLineStatus()** for details of how to determine whether the Transmit FIFO already contains data.

Before this function is called, the UART must be enabled using the function **bAHI_UartEnable()**, otherwise an exception will result.

Parameters

u8Uart	Identity of UART: E_AHI_UART_0 (UART0) E_AHI_UART_1 (UART1)
u8Data	Byte to transmit

Returns

None

(PC の python プログラム)

```python
import serial
import time

def main():
    try:
        ser = serial.Serial(
            port='COM5',      # デバイスポート番号
            baudrate=115200,            # ボーレート
            parity=serial.PARITY_NONE,
            stopbits=serial.STOPBITS_ONE,
            bytesize=serial.EIGHTBITS,
            timeout=1
        )
        if ser.is_open: # シリアルポートがオープンされているかチェック
            print("Serial port opened successfully.")
        while True: # データ受信
            print("Waiting for data...")
            if ser.in_waiting > 0:    # 受信バッファにデータがあるか確認
                print("Data in buffer")
                high_data = ser.read(ser.in_waiting).decode('utf-8')
                low_data = ser.read(ser.in_waiting).decode('utf-8')
                val = (high_data << 8 ) + low_data
                valtage = val * (5.0 / 1024)
                print("Received data:", valtage)
            time.sleep(1)             # 1秒待機

    except serial.SerialException as e:
        print("Error opening or operating the serial port:", e)
    finally:
        if ser.is_open:
            ser.close()
            print("Serial port closed.")

if __name__ == "__main__":
    main()

```

以下が結果である.

```
C:¥Users¥郷平¥Desktop>python sensor.py
Serial port opened successfully.
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Data in buffer
Serial port closed.
Traceback (most recent call last):
  File "C:¥Users¥郷平¥Desktop¥sensor.py", line 37, in <module>
    main()
  File "C:¥Users¥郷平¥Desktop¥sensor.py", line 24, in main
    val = (high_data << 8 ) + low_data
          ~~~~~~~~~~~~~~~
TypeError: unsupported operand type(s) for <<: 'str' and 'int'

C:¥Users¥郷平¥Desktop>
```

「Data in buffer」と表示されているため, Coordinator から UART 通信でデータは受信できている.

【添付している参考論文について】

実験の条件設定，実験方法・結果が参考になる論文である．実際にノードを設置し，RSSI, PER, Throughput で通信時の評価している．

環境として，コンピュータラボであるため，テーブル，コンピュータ，モニター，教室の椅子で構成されている．また，デバイスは床から 130～140　cm の場所に配置．壁の厚さは約 35cm のレンガ造りで，建物の外壁はコンクリートでできており，外壁の大部分は窓になっている．

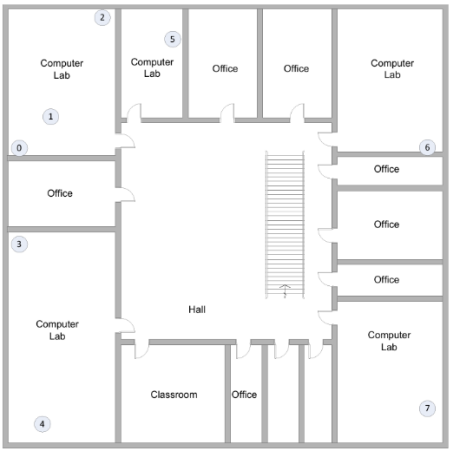


Figure 3.　The building 1st floor layout and ZigBee station deployement

TABLE II.
THE DISTANCE OF THE NODES IN THE EXPERIMENT

No	Position	Distance (m) approx.
0	Coordinator – Lab 28	-
1	Lab 28	1 m
2	Lab 28	10 m
3	Lab 29	6.5 m
4	Lab 29	18 m
5	Lab 27	12.5 m
6	Lab 24	26.5 m
7	Lab 20	31.5 m

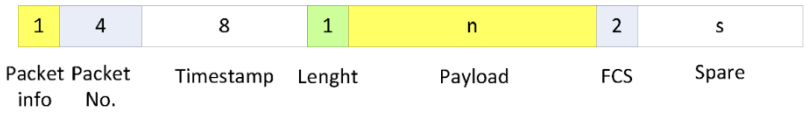
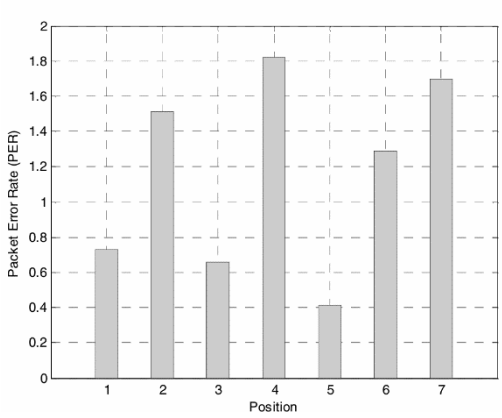


Figure 1.　Packet format in PSD file [5]

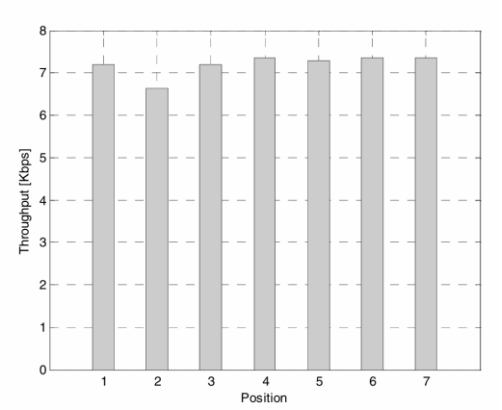
実験では，各ノードから Coordinator に 127 バイトの長さのパケットを 2,500 個送信するとされているが，テストごとに約 10,000 個のパケットが送信されるのは，テスト中に多数の ACK および CMD フレームが送信されるため．

TABLE III.
THE RESULTS OF ZIGBEE NODES MEASUREMENTS FOR THE DIFFERENT POSITIONS

Position	Packets Sent (No.)	PER (%)	RSSI Avg. (dBm)	RSSI Max. (dBm)	RSSI Min. (dBm)	Throughput (Kbps)	Test Time (min)
1	10087	0.73	-45.98	-40	-55	7.2	8.02
2	10865	1.51	-83.95	-66	-97	6.64	9.33
3	10091	0.66	-66.01	-64	-70	7.2	8.1
4	10153	1.82	-89.05	-82	-97	7.36	7.86
5	10126	0.41	-70.09	-66	-78	7.28	8.09
6	10231	1.29	-77.27	-71	-88	7.36	8.03
7	10171	1.7	-92.54	-91	-96	7.36	7.81



最大の PER は，位置 7, 4, 2 である．位置 6 も PER 値が高くなっているが，いずれの場合も，PER 値は許容範囲内としている．



パケット送信率が高いことから，障害物やノードの距離がすべての位置のリンク品質に影響を与えなかった．

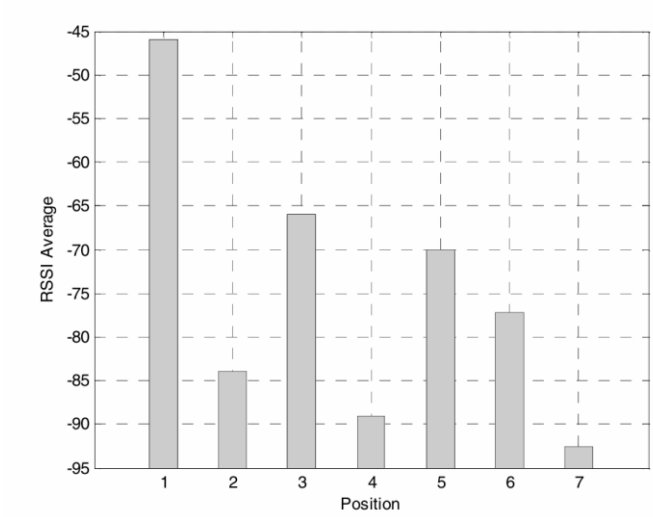
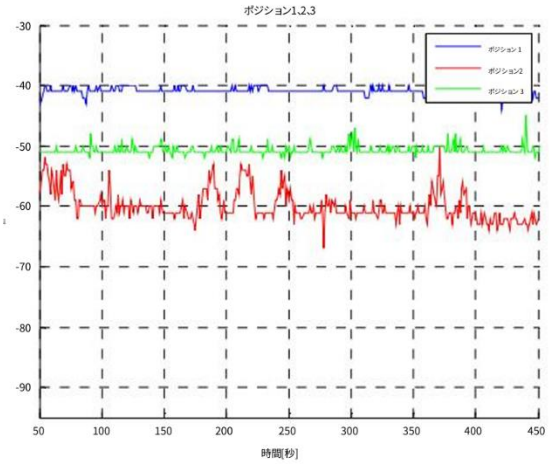
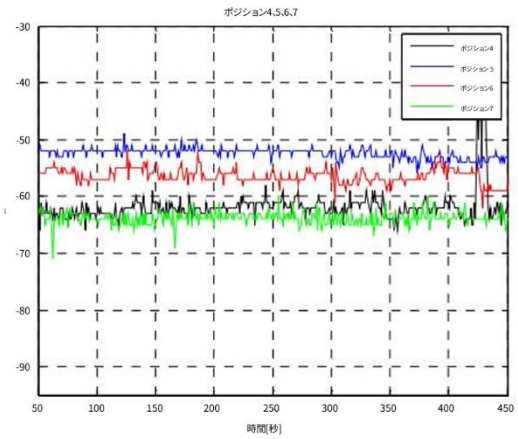


Figure 4. The RSSI [dBm] for 7 different positions of ZigBee station

位置 2 は，2 つのノードの間には，コンピューターとモニターが置かれたテーブルや教室の椅子がいくつか配置されており，RSSI　平均値が低くなっている可能性がある．





1 秒ごとの RSSI 平均値がグラフで表示され, RSSI の変化を分析して通信の安定性を判断する. 両方のグラフは, テスト中に信号強度に大きな違いがないことを示している.

【LQI】

パルアプリ パルビューア - MONO-WIRELESS.COM

https://www.musen-module.com/experiment-column/twelite-experiment3/

・パケット内に LQI 埋め込まれている.
・Router, Enddevice の切り替わり・経路も検出可能.
・タイムスタンプで 1 秒 64 カウントで遅延も測れる.
・中継フラグで, 何回中継したかがわかる.