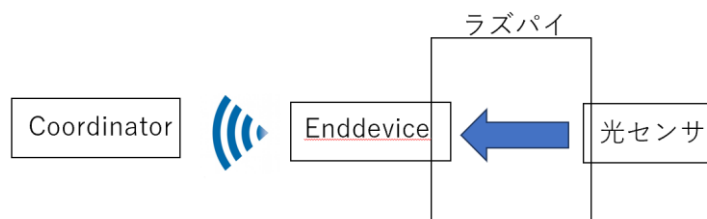


0111



現在の関係図である。

ラズパイには Enddevice と光センサが接続されている。

1. 光センサのセンサ値をラズパイを介して, Enddevice が取得する。
2. センサ値と Enddevice が取得したデータをラズパイで表示し、確認する。
3. Enddevice はセンサ値を Coordinator へ送信する。
4. Coordinator で受信したセンサ値を表示し、確認する。

【前回】

16 進数のセンサ値を Enddevice から Coordinator へ送信し, Coordinator 側で受信したセンサ値を 16 進数として表示することができていた.

【今回】

Coordinator 側で受信したセンサ値を 16 進数ではなく, 10 進数で表示するようにする.
そこで, Coordinator のコードに、受信した 16 進数を 10 進数に変換するビット演算を追加した.

以下は Coordinator 側の受信したデータを処理する app_coordinator_endpoint.c である.

```
int hexToDecimal(uint8_t hex[]) { //16進数から10進数に変換
    uint16 decimal = 0;
    uint16 length = 0;
    uint16 i;

    while (hex[length] != '\0') { // 16進数の桁数を求める
        length++;
    }

    for (i = 0; i < length; i++) { // 各桁の16進数を10進数に変換
        uint8_t currentChar = hex[i];
        uint16 currentDigit;

        if (currentChar >= '0' && currentChar <= '9') {
            currentDigit = currentChar - '0';
        } else if (currentChar >= 'A' && currentChar <= 'F') {
            currentDigit = 10 + (currentChar - 'A');
        } else if (currentChar >= 'a' && currentChar <= 'f') {
            currentDigit = 10 + (currentChar - 'a');
        } else {
            DBG_vPrintf(TRACE_APP, "Error\n"); // エラー処理: 16進数以外の文字が含まれている場合
            return -1;
        }

        decimal = (decimal << 4) | currentDigit; // 10進数に変換
    }

    return decimal;
}
```

```
app_sleeping_enddevice.c | app_coordinatorc | app_endpointc

/* Process incoming cluster messages for this endpoint... */
// DBG_vPrintf(TRACE_APP, "    Data Indication:\r\n");
// DBG_vPrintf(TRACE_APP, "    Profile :%x\r\n", sStackEvent.uEvent.sApsDataIndEvent.u16ProfileId);
// DBG_vPrintf(TRACE_APP, "    Cluster :%x\r\n", sStackEvent.uEvent.sApsDataIndEvent.u16ClusterId);
// DBG_vPrintf(TRACE_APP, "    EndPoint:%x\r\n", sStackEvent.uEvent.sApsDataIndEvent.u8DstEndpoint);

uint8 u8TempPayload;
uint16 u16bytesread;
uint16 i;

DBG_vPrintf(TRACE_APP, "received data16:");

for(i = 0; i < 6; i++){ //受信した16進数
    u16bytesread = PDUU_u16APduInstanceReadNBO(sStackEvent.uEvent.sApsDataIndEvent.hAPduInst,i,"b",&tmpString);
    DBG_vPrintf(TRACE_APP, "%x", tmpString);
}

DBG_vPrintf(TRACE_APP, "\n");
DBG_vPrintf(TRACE_APP, "received data10:");

uint8 decimalNumber = hexToDecimal(tmpString); //16進数から10進数に変換
for(i = 0; i < 6; i++){
    u16bytesread = PDUU_u16APduInstanceReadNBO(sStackEvent.uEvent.sApsDataIndEvent.hAPduInst,i,"b",&decimalNumber);
    DBG_vPrintf(TRACE_APP, "%d", decimalNumber);
}

DBG_vPrintf(TRACE_APP, "\n");

DBG_vPrintf(TRACE_APP, "-----Finish data-----\n");

/* free the application protocol data unit (APDU) once it has been dealt with */
PDUU_eAPduFreeAPduInstance(sStackEvent.uEvent.sApsDataIndEvent.hAPduInst);
}
break;
```

以下は Enddevice と Coordinator どちらも共有する utils.c である.

```

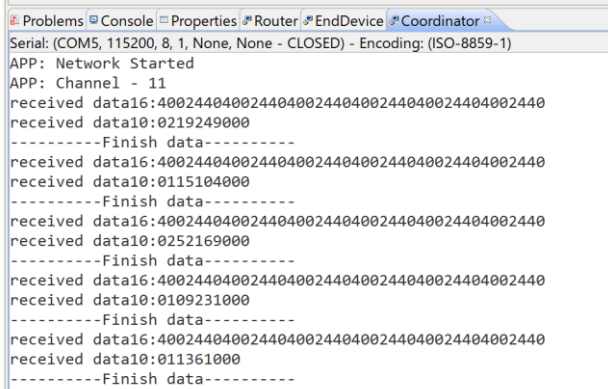
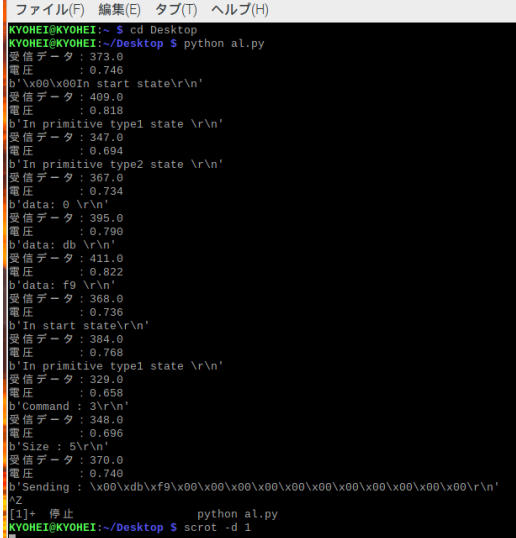
PUBLIC int CMD(){
    if(cmd == 3){
        cmd = 0;
        return 3;
    }else{
        return 0;
    }
}

void vReadCharInterrupt ()
{
    uint8_t rxByte = uSAHI_UartReadData (DBG_F_UART_0);

    switch(state){
        case START: //最初を表す文字
            if(rxByte == 0xAA)
            {
                state = PRIMITIVE_TYPE;
                DBG_vPrintf(TRUE, "In start state\n");
                break;
            }
            break;
        case PRIMITIVE_TYPE: //コマンド処理
            DBG_vPrintf(TRUE, "In primitive type1 state \n");
            if(rxByte == 0xBB){
                DBG_vPrintf(TRUE, "In primitive type2 state \n");
                state = WAIT_DATA;
                break;
            }
            break;
        case WAIT_DATA: //センサデータ(4回ループ 1byteごとに呼ばれる 4byte)
            DBG_vPrintf(TRUE, "data: %x \n", rxByte);
            tmpString[indexCount] = rxByte;
            indexCount++;
            if(indexCount == 3){
                cmd = 3;
                indexCount = 0;
                state = START;
            }
            break;
    }
}

```

以下はラズパイと Coordinator 側の実行結果である.



上記の結果から, 受信した 16 進数を表す received data16 とラズパイのセンサ値が一致していない. 期待表示内容は
received data16 : 00dbf9
received data10 : 56313
しかし, この「56313」もラズパイに表示してあるセンサ値と一致していない.

`Odecimal = (decimal << 4) | currentDigit;` について

16 進数の各桁を 10 進数に変換している.

具体的には、`(decimal << 4)` は `decimal` のビットを左に 4 回シフトし、これにより 16 進数の各桁を 10 進数に変換している. そして、`| currentDigit` は現在の 16 進数の桁を加えている.

例えば, 1A を変換する場合だと

1. 最初に `decimal` は 0 である.
2. '1' を処理すると, `(0 << 4) | 1` となり, `decimal` は 1 になる.
3. 'A' を処理すると, `(1 << 4) | 10` となり, `decimal` は 26 になる.

これにより, 16 進数 1A は 10 進数の 26 に変換できる.

アルベルト先生からは,

16 進数を受け取るときは, `int8_t` の配列 → 1 つまたは複数の `int32_t` の値の変換が必要とのことだったが, 実装できなかったため, 上記のコードで試した. 再度検討すべき.