Electronic-tech Jack for IoT　www.eleki-jack-iot.com　トランジスタ技術 増刊

先進テクノロジで未来を切り開く

エレキジャックIoT

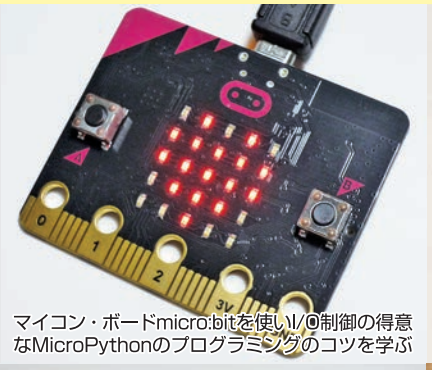
No. 4

特集

ラズパイ, STM32, micro:bit使用 PythonでI/O制御プログラミング

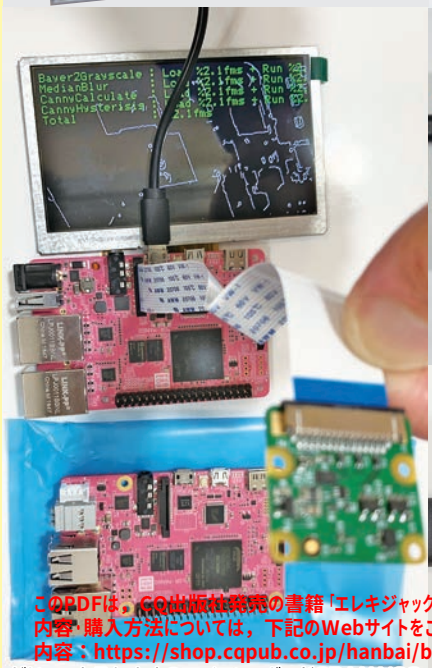
—Bluetoothメッシュ・ネットワークで広がるIoT

Bluetooth LEメッシュ・ネットワークの仕様と評価，n対n無線通信ネットワークを構築する

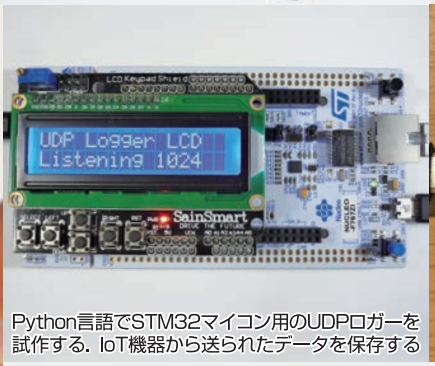


マイコン・ボードmicro:bitを使いI/O制御の得意なMicroPythonのプログラミングのコツを学ぶ

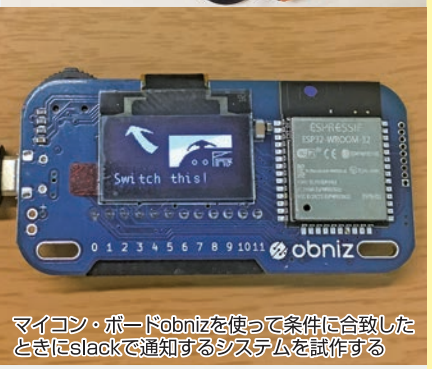
コマンドが公開されている市販ドローンの制御用コントローラをESP32マイコンで試作する



がじぇっとるねさすのマイコン・ボードGR-MANGOをオンライン・コンパイラのサンプルでテストする



Python言語でSTM32マイコン用のUDPロガーを試作する．IoT機器から送られたデータを保存する



マイコン・ボードobnizを使って条件に合致したときにslackで通知するシステムを試作する

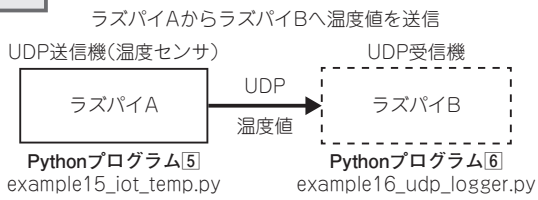
ESP32系マイコンM5Stackで電圧計を試作．搭載されているA-Dコンバータの使い方をマスタする

UDPで送られてきた文字データをTTGOの開発ボードT-KoalaにI²C接続したLCDに表示する

CQ出版社

Python プログラム 5
ラズベリー・パイで温度を UDP 送信

ラズベリー・パイ内蔵の温度センサから得られた温度値をLAN内にUDPブロードキャスト送信する．ラズベリー・パイは1台でも実験できる



リスト1-6のexample15_iot_temp.pyを起動すると，ラズベリー・パイ内蔵の温度センサの値をLAN内へUDPで送信します．Pythonプログラム 6 で紹介する受信プログラムで温度を表示できます．

2台のラズベリー・パイで通信するプログラムですが，1台のラズベリー・パイで，送信プログラムと受信プログラムを別々のLXTerminalで実行して実験することができます．プログラムの停止は[Ctrl]＋[C]です．

以下に，送信側となる**リスト1-6**のexample15_iot_temp.pyのおもな処理の流れについて説明します．

① 変数intervalに送信間隔（秒）を代入します．

② 変数temp_offsetにCPUの内部発熱による温度上昇値を代入します．例えば，室温が20 ℃で，内部発熱が17.8度だった場合，温度センサは37.8 ℃を示します．内部発熱の17.8度を減算することで，室温の目安値を得ることができます．

③ 内蔵温度センサから温度値を取得するための処理部です．

④ 変数temp_fに代入された浮動小数点数型の温度値を整数型に変換します．整数に変換する関数としては，他にも小数点以下を切り捨てるintがあります．測定値を丸めるときはroundを使用することで，0.5よりも大きな値が切り上げられます．ただし，ちょうど0.5のときは，結果が偶数になるように丸めます（四捨五入と比べ，丸め処理による累積誤差を減らすことができるため）．

⑤「temp._3, 温度値」の書式で，UDPのブロードキャスト送信を行います．複数のIoT温度センサを設置するときは，temp._3の末尾の数字を書き換えることで，受信側でセンサを特定することができます．

⑥ 変数intervalの秒数の間，sleep命令で待ち時間処理を行い，送信を待機します．

プログラム実行のようすを**図1-10**に示します．

リスト1-6　IoT温度計のサンプル・プログラム example15_iot_temp.py

```
#!/usr/bin/env python3

# Example 15 ラズベリー・パイを使ったIoT温度計

filename = '/sys/class/thermal/thermal_zone0/temp'      # 温度ファイル
udp_to = '255.255.255.255'                              # UDPブロードキャスト
udp_port = 1024                                         # UDPポート番号
device_s = 'temp._3'                                    # デバイス識別名
interval = 30 ←①                                       # 送信間隔(秒)
temp_offset = 17.8 ←②                                  # CPUの温度上昇値(要調整)

import socket                                           # ソケット通信ライブラリ
from time import sleep                                  # スリープ実行モジュール

while True:
    try:                                                # 例外処理の監視を開始
        fp = open(filename)                             # 温度ファイルを開く
    except Exception as e:                              # 例外処理発生時
        print(e)                                        # エラー内容を表示
        sleep(60)                                       # 60秒間の停止
③       continue                                        # whileに戻る

    temp_f = float(fp.read()) / 1000 - temp_offset      # 温度値tempの取得
    temp_i = round(temp_f) ←④                          # 整数に変換してtemp_iへ
    fp.close()                                          # ファイルを閉じる
    print('Temperature =',temp_i,'('+str(temp_f)+')')   # 温度値を表示する

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)      # ソケット作成
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST,1)    # ソケット設定
    udp_s = device_s + ', ' + str(temp_i)               # 表示用の文字列変数udp
    print('send :', udp_s)                              # 受信データを出力
    udp_bytes = (udp_s + '¥n').encode()                 # バイト列に変換
⑤   try:                                                # 作成部
        sock.sendto(udp_bytes,(udp_to,udp_port))        # UDPブロードキャスト送信
    except Exception as e:                              # 例外処理発生時
        print(e)                                        # エラー内容を表示
    sock.close()                                        # ソケットの切断
    sleep(interval) ←⑥                                 # 送信間隔の待ち時間処理
```

（IoT温度計・子機・送信側プログラムの実行）

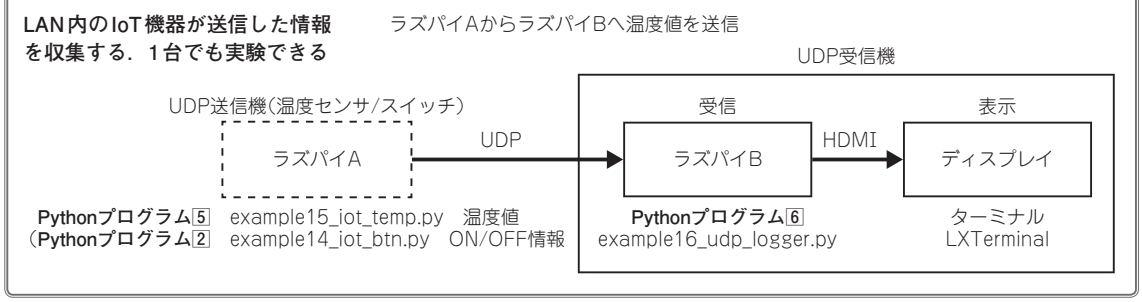
```
pi@raspberrypi:~ $ cd ~/iot/learning
pi@raspberrypi:~/iot/learning $
                              ./example15_iot_temp.py
Temperature = 30 (29.974)
send : temp._3, 30
Temperature = 29 (29.435999999999996)
send : temp._3, 29
Temperature = 30 (29.974)
send : temp._3, 30
Temperature = 31 (30.511999999999997)
send : temp._3, 31
```

（親機・受信側プログラムの実行）

```
pi@raspberrypi:~ $ cd ~/iot/learning
pi@raspberrypi:~/iot/learning $
                              ./example16_udp_logger.py
Listening UDP port 1024 ...
2019/06/01 16:45, temp._3, 30
2019/06/01 16:46, temp._3, 29
2019/06/01 16:46, temp._3, 30
2019/06/01 16:47, temp._3, 31
```

図1-10　サンプル・プログラムexample15_iot_temp.pyの実行結果例
各プログラムを実行した状態でタクト・スイッチを操作すると，PingまたはPongのメッセージの送受信が行える

Python プログラム 6
ラズベリー・パイで UDP のデータ受信



Pythonプログラム 5 から送ったデータなどをネットワーク経由で受信してLXTerminal上に表示します．**リスト1-7**のexample16_udp_logger.pyがプログラムです．LAN内の温度値やボタン押下通知などのUDPデータを受信し表示します．

プログラムを起動すると，送信されたUDPデータを受信し，**図1-11**のように受信日時とともに表示します．プログラムを停止するには，LXTerminal上で，[Ctrl]+[C]の操作を行います．

以下に，**リスト1-7**のプログラムexample16_udp_logger.pyのおもな処理にを説明します．

① 受信日時を表示するために，日時を扱うライブラリdatetimeを組み込みます．

② ライブラリsocketから通信ソケット用の変数(オブジェクト)sockを生成します．

③ 生成したソケットsockに，UDPポート1024をbind命令で接続します．引き数はタプル型の接続先のアドレスとポート番号です．ここでは，アドレスを指定せずに，すべてのアドレスを対象にし

```
pi@raspberrypi:~ $ cd ~/iot/learning
pi@raspberrypi:~/iot/learning $
                          ./example16_udp_logger.py
Listening UDP port 1024 ...
2019/06/01 17:24, temp._3, 30 ←IoT温度計
2019/06/01 17:24, Ping ←IoTボタン
2019/06/01 17:24, Pong
2019/06/01 17:25, temp._3, 29
2019/06/01 17:26, temp._3, 30
2019/06/01 17:26, Ping
2019/06/01 17:26, Pong
```

図1-11　サンプル・プログラムexample16_udp_logger.pyの実行結果例
プログラムを実行した状態で，IoTボタンや，IoT温度センサからのUDPデータを受信すると，受信時刻とともに表示出力する

リスト1-7　受信プログラム example16_udp_logger.py

```
#!/usr/bin/env python3

# UDPを受信する

import socket
import datetime ←①

print('Listening UDP port', 1024, '...', flush=True)        # ポート番号1024表示
try:
    sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)←②      # ソケットを作成
    sock.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)    # オプション
    sock.bind(('', 1024)) ←③                                   # ソケットに接続
except Exception as e:                                          # 例外処理発生時
    print(e)                                                    # エラー内容を表示
    exit()                                                      # プログラムの終了
while sock:                                                     # 永遠に繰り返す
    udp = sock.recv(64) ←④                                     # UDPパケットを取得
    udp = udp.decode() ←⑤                                      # 文字列へ変換する
    udp = udp.strip() ←⑥                                       # 先頭末尾の改行削除
    if udp.isprintable(): ←⑦                                   # 全文字が表示可能
        date=datetime.datetime.today() ←⑧                      ⑨# 日付を取得
        print(date.strftime('%Y/%m/%d %H:%M'), end='')          # 日付を出力
        print(', ' + udp, flush=True) ←⑩                       # 受信データを出力
```

ESP32＋センサ

送信機の準備 3
ESP32＋センサ・シールド・キット

SensorShield-EVK-003（ローム）は，8つのセンサ・モジュールとArduino用シールド基板（部品実装済み）が付属するセンサ評価用キットです．センサ・モジュールは個別に仕切られた小さなケースに収納されており，まるで私たちが慣れ親しんだ日本の弁当のように，いろいろなセンサを試用（味見）することができます（**写真3-3**）．

ここでは，5つのセンサ・モジュールを取り付けたセンサ・シールド基板を，Wi-FiとBluetooth 内蔵ESP32マイコンを搭載したIoT Express（CQ出版社）に取り付け，スマートフォンやラズベリー・パイでセンサ値を表示する方法について紹介します（**写真3-4**）．

● ローム製センサ・シールド・キットの組み立て方

センサ・シールド・キットに付属するセンサ・モジュールの中から，加速度センサ KX224-1053，気圧センサBM1383AGLV，地磁気センサBM1422AGMV，近接センサRPR-0521RS，カラー・センサBH1749NUCの5個を選び，**写真3-5**のように取り付けました．こ

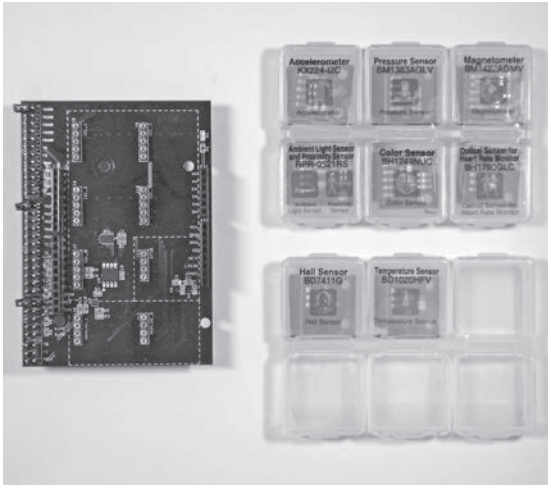


写真3-3　センサ・シールド・キットSensorShield-EVK-003に含まれるシールド基板（左・基板）**とセンサ・モジュール**（右・小型ケース）
電源回路などが実装されたArduino用シールド基板と8個のセンサ・モジュールのキット．センサ・モジュールは，個別に仕切られた小さなケースに収納されている

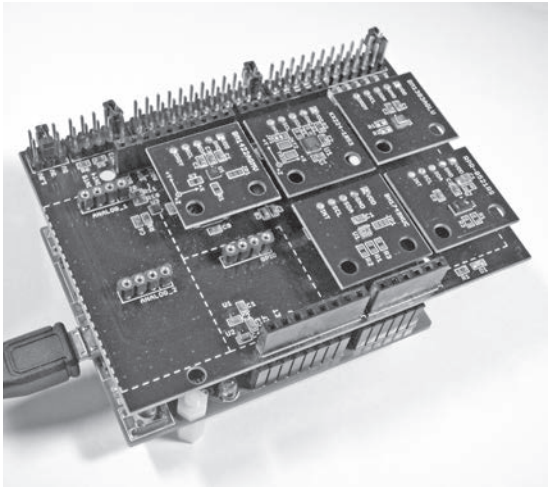
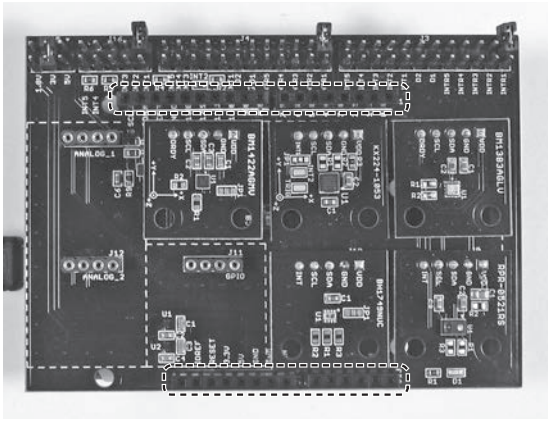


写真3-5　5つのセンサ・モジュールをセンサ・シールド基板に取り付けてIoT Expressに接続した
I²Cインターフェース搭載センサ・モジュールは，センサ・シールド基板上の端子J5，J6，J7，J9，J10のどこに接続してもかまわないが，GPIO接続用のJ11と，アナログセンサ用のJ8とJ12は空けておく



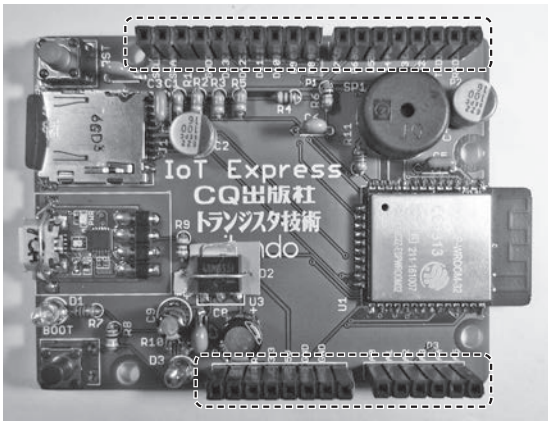


写真3-4　センサ・シールド基板（左）**とIoT Express**（右）
Arduino用シールドに対応したI/O端子を使って，センサ・シールド基板（左）とIoT Express（右）を接続することができる

MicroPython プログラム 3
micro:bit で温度センサ送信

今度は，温度センサ送信機のプログラムを制作します．**図4-9**のように，micro:bit内蔵の温度センサから値を読み取り，約5秒ごとにワイヤレスで送信します．送信された温度値は，前節の簡単ワイヤレス通信実験を行うPythonプログラムで受信します．

リスト4-3に示すワイヤレス温度センサのプログラムexample03_temp.pyの温度センサの処理内容を以下に示します．

① 構文whileを使って，以下の処理①～③を繰り返し実行します．

② 内蔵温度センサから温度値を取得するには，temperature命令を使用します．取得した数値をstr命令で文字列に変換し，変数txに代入します．

③ ワイヤレス送信を行うradio.send命令を使って変数txの内容を送信します．

④ 待ち時間処理を行う命令sleepを使って5秒間待機します．

他にも，明るさが変化したときに照度値を送信するプログラムexample03_illum.pyや，本体が傾いたときに重力加速度を送信するプログラムexample03_accem.pyも同じフォルダに収録しました．各プログラムの機能はp.60の**表4-1**を参照してください．

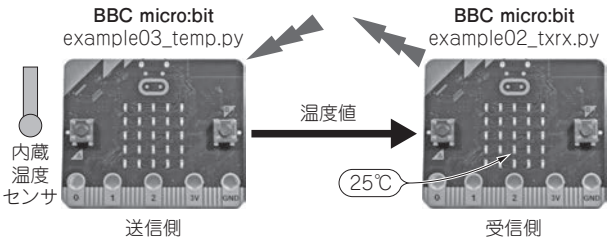


図4-9　BBC micro:bitを使ったワイヤレス温度センサ
送信側の内蔵温度センサから取得した温度値を送信し，受信側で表示する

リスト4-3　ワイヤレス通信を行うプログラム
example03_temp.py
本体の5×5 LED表示画面に[Temp]を表示後，5秒ごとに温度値を送信する

```
import radio
from microbit import *

radio.on()
print('Ready Temp')
display.scroll('Temp')
while True: ←①
    tx = str(temperature()) ←②
    print('Tx:',tx)
    radio.send(tx) ←③
    display.scroll(tx)
    sleep(5000) ←④
```

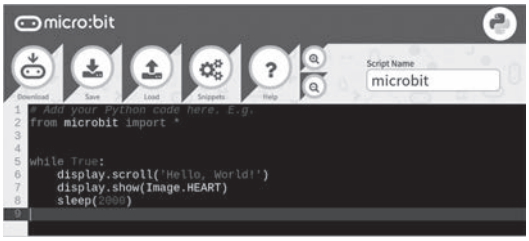
コラム2　micro:bit用Pythonエディタのバージョンの違い

micro:bit用Pythonエディタの機能アイコンは，バージョンによって表示が異なります．

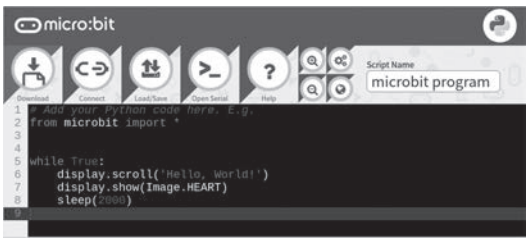
バージョン1.1（**図A**）では，左から順にDownload, Save, Loadと続き，大小合わせて7個のアイコンでしたが，バージョン2.0（**図B**）では，Download, Connect, Load/Save, Open Serialの9個になりました．

micro:bitにプログラムを書き込むには，本稿で説明した使い方に加え，あらかじめConnectアイコンでmicro:bitに接続しておけば，Downloadアイコンが Flashボタンに変わり，Flashボタンの押下で直接プログラムをmicro:bitに書き込むことができます．

バージョン2.0では，Open Serialのアイコンでシリアル通信用ターミナル・ソフトが開きます．



図A　バージョン1.1のPythonエディタ



図B　バージョン2.0のPythonエディタ

4 NUCLEO-F767ZIで製作するSTM32マイコン版UDPモニタ

これまでに製作したSTM32マイコン版IoTボタンが送信するボタン操作情報や，IoT温度センサが送信する温度値を受信し，Arduino用LCDシールドに表示する「UDPモニタ」を，STM32マイコン開発ボードNUCLEO-F767ZIで製作します(**図5-6**)．

ラズベリー・パイ用のudp_logger.pyを基に，STM32マイコン版udp_logger.py(LCD不要)と，LCD表示機能を追加した**リスト5-3**のudp_logger_lcd.py(要LCDシールド)を作成しました(**写真5-5**)．

STM32マイコン開発ボードNUCLEO-F767ZIにはArduino用シールドと同じピン配列の拡張端子が搭載されているので，Arduino用LCD Keypad Shield(DF ROBOT製，SainSmart製など)を取り付けることができます．LCD Keypad Shieldの動作電圧は5Vで，開発ボードの信号レベルは3.3Vのため，本来は信号レベル変換が必要ですが，実力的には装着するだけで使用できます．

LCD制御用ドライバlcd.pyは，MicroPythonのファームウェアを書き込むinstall.shを実行したときに，ラズベリー・パイ内にダウンロードされています．**図5-7**のようにコピー・コマンドを使い，ファイル名を変更せずに，STM32マイコンへ書き込み，続けてサンプル・プログラムudp_logger_lcd.pyを書き込みます．両方のファイルの書き込みが完了(緑色のLEDが消えてから5秒以上)してから，黒色のRESETボタンを押してください．

製作したUDPモニタを起動後，IoTボタンやIoT温度センサなどからUDPデータを受信すると，LCDの1行目に送信元IPアドレスが，2行目に受信データが表示されます．LCDなしのudp_logger.pyの場合は，動作ログとして受信データをシリアル出力します．serial_logger.pyなどでシリアル・データを確認できます．

以下に，**リスト5-3**のudp_logger_lcd.pyのおもにLCD表示部の処理内容について説明します．

① LCD制御用ドライバ(ライブラリ)lcd.py内のHD44780を組み込みます．
② 処理①で組み込んだHD44780の変数(オブジェクト)lcdを生成します．
③ 処理②で生成した変数lcd内で使用される配列変数PINSに，LCDのピン割り当て番号を代入します．
④ LCDを初期化する命令initを実行します．
⑤ 表示位置を指定する命令set_lineを使い，1行目の位置0を指定します．2行目の場合は，1です．
⑥ LCDへ文字列を表示する命令set_stringを使って，「UDP Logger LCD」を表示します．

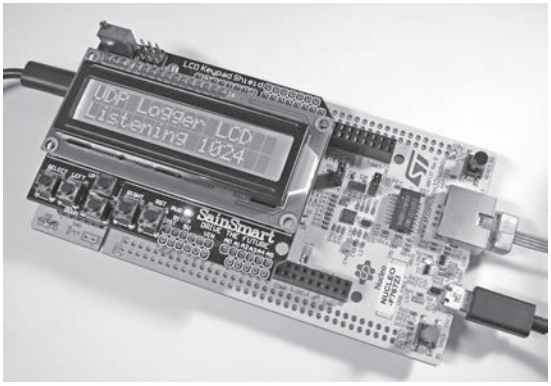


写真5-5　NUCLEO-F767ZIにArduino用LCDシールドを装着したときのようす
Arduino用シールドと同じピン配列の拡張端子が搭載されているので，Arduino用LCD Keypad Shield(DF ROBOT製，SainSmart製など)を取り付けることができる

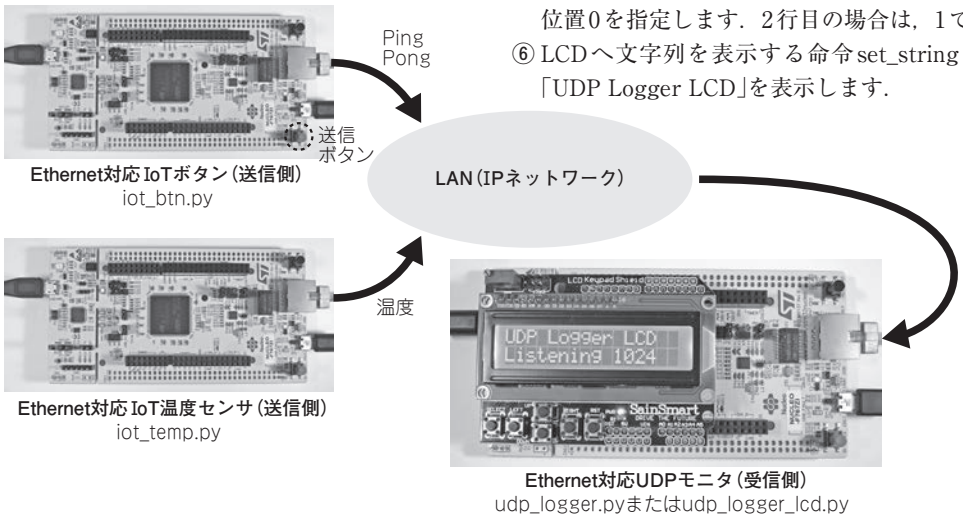


図5-6　NUCLEO-F767ZIで製作するSTM32マイコン版UDPモニタ
製作したSTM32マイコン版IoTボタンとIoT温度センサが送信するボタン操作情報や温度値を受信してLCD表示する

特別企画

Bluetooth Low Energyで
n対n無線通信ネットワークを構築

BLEメッシュ・ネットワークのすべて

使用機材
Bluetooth 5.0評価ボード Target Board for RX23W(ルネサス) 2枚
Androidスマートフォン

樋口 光彦
(ルネサス エレクトロニクス)

低消費電力のBluetooth Low Energy(LE)を使って，n対nの無線通信ネットワークを構築できるBluetooth Mesh Networkingを解説します．
Bluetooth Mesh Networkingの仕様，基本概念，動作の概要を見たあと，組み込み向けマイコンを使用して，Bluetooth Mesh通信の評価手順を解説します．

Bluetooth Meshとは？

Bluetooth Meshの想定するアプリケーションや特徴について紹介します．

● Bluetooth Meshの想定するアプリケーション

Bluetooth Meshはホーム・オートメーション，ビルディング・オートメーション，ファクトリー・オートメーションなどの分野で活用が期待されています．例としてホーム・オートメーションの照明制御があげられます(**図1**)．

Bluetooth Meshに対応した照明機器を各部屋に設置することで，スマートフォンのアプリケーションから各照明の制御が実現できます．また照明制御のための無線パケットがスマートフォンから直接届かない場合でも，各照明機器が無線パケットをリレーすることで最終的にすべての照明機器まで到達できます．

● Bluetooth Meshの特徴

Bluetooth LEの基本トポロジー(1対nのブロー

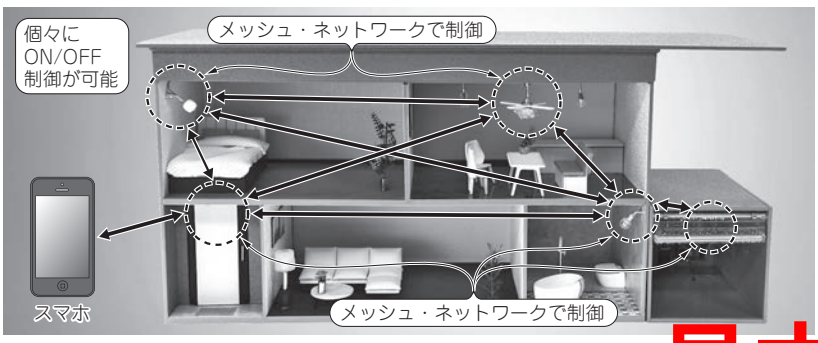


図1
ホーム・オートメーションでの照明制御

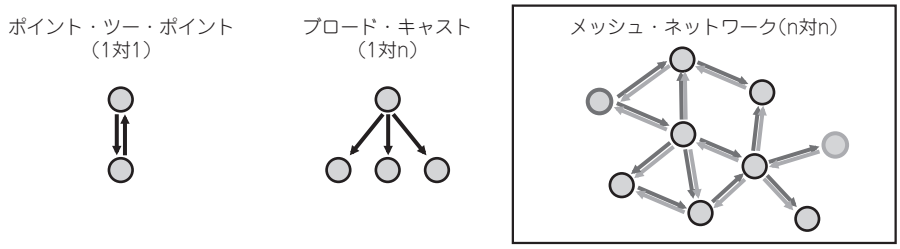


図2
Bluetooth LEのトポロジー

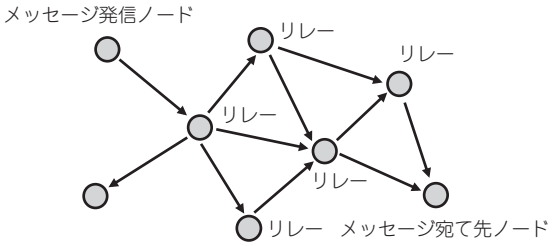


図3　メッシュ・ネットワークのメッセージ伝搬

ドキャスト，または1対1のポイント・ツー・ポイント）を組み合わせてメッシュ・ネットワークを構築します．

Bluetooth Meshは，Bluetooth LEを応用してn対nデバイス間での無線通信を実現します（**図2**）．

- n対nデバイス間での無線通信
- データを暗号化するセキュアな通信
- スマートフォンとの親和性

Bluetooth Meshの通信はネットワーク・キーと呼ばれる暗号鍵で暗号化されます．これによりネットワークに参加していないデバイスはネットワークの通信を盗聴することができません．さらにネットワークに参加している限定されたデバイスでアプリケーション・キーと呼ばれる暗号鍵を共有することでデバイス間でのセキュアな通信が可能です．

Bluetooth LEに対応したPCやスマート・スピーカ，スマートフォンであれば，ネットワーク内のデバイスを制御するためのコントロール・パネルやリモート・コントローラとして活用することもできます．

● Bluetooth Meshの基本動作

メッシュ・ネットワークに参加しているデバイスは**ノード**，ノード同士がやり取りする信号は**メッセージ**と呼ばれます．

メッシュ・ネットワーク内のあるノードがメッセージを送信すると，周囲のノードは次々とメッセージをリレーします．リレーが繰り返されることでメッセージがネットワーク全体に伝搬し，最終的にメッセージの宛て先ノードに到達します（**図3**）．またメッセージのリレー経路が固定されることはなく，各ノードが移動した場合でもメッセージはリレーされます．

さらにBluetooth Mesh Networking仕様はプロキシ機能，フレンド機能などノードのオプション機能を定義しており，多様なメッシュ・ネットワークを形成できます．オプション機能の詳細は次節の「ノードのオプション機能」の項目で解説します．

Bluetooth Mesh仕様の基本概念と通信動作

本節ではBluetooth Mesh仕様の基本的な概念やどのようにn対nのデバイス間での通信を実現しているかについて解説します．

● ノードを構成するエレメント，モデル，ステート

メッシュ・ネットワークに参加しているデバイスは「ノード」と呼ばれ，ノードはエレメント，モデル，ステートで構成されます（**図4**）

ESP32マイコンで作る電圧計
M5Stack/M5StickCの A-Dコンバータ 応用プログラミング

使用機材
M5Stack BASIC / M5Stack GRAY / M5Stick C×1
ミニ・ブレッドボード×1　可変抵抗器(10kΩ)×1
ジャンパ・ワイヤ(オス-オス)×3

国野 亘

A-Dコンバータはアナログ値をディジタル値に変換するデバイスです．マイコンでアナログ値が利用できるようになり，IoTプログラミングの応用範囲がグッと広がります．
液晶ディスプレイを搭載したESPマイコンとして注目されているM5Stackに搭載されているA-Dコンバータを使った電圧計のプログラムを紹介します．プログラムは3本あり，ステップアップしながら機能を追加していきます．ぜひA-Dコンバータ応用プログラミングを実際に試してみてください．

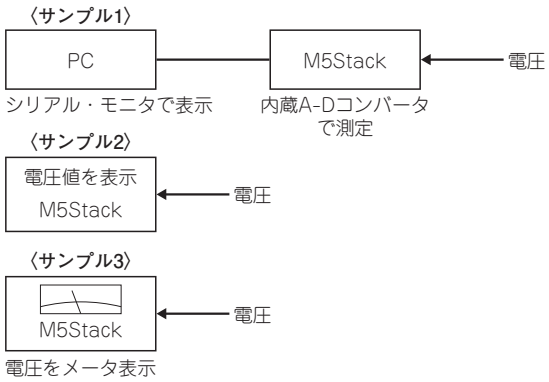


図1　3つのサンプル・プログラムをとおしてA-Dコンバータを利用したプログラミングのコツを学びます

液晶ディスプレイを搭載したESPマイコンとして注目されているM5StackとM5Stick CにはA-Dコンバータが搭載されています．

本稿では，A-Dコンバータの利用方法を学びながら液晶ディスプレイに電圧を表示する電圧計を作ります(**図1**)．

〈サンプル1〉はA-Dコンバータで得た値とその値から得た電圧値をシリアル・モニタに出力します．

〈サンプル2〉は電圧値をM5Stack，M5Stick Cの液晶ディスプレイに表示します．

〈サンプル3〉は電圧値を液晶ディスプレイにアナログ・メータ風にして表示します．

写真1　2インチの液晶ディスプレイを搭載したM5Stack GRAY(左)と，0.98インチのM5Stick C(右)

プログラミング前の準備

● ハードウェアは，ESPマイコン内蔵M5Stackと M5Stcick Cを使う

M5StackとM5Stick Cは，Wi-Fi機能，ディスプ

ラズパイ互換の拡張コネクタ／
ラズパイ・カメラやHDMIが使えるMbedボード

がじぇっとるねさす GR-MANGOをサンプル・プログラムで使ってみる

使用機材
マイコン・ボードGR-MANGO

岡宮 由樹(ルネサス エレクトロニクス)

ピンク色の基板が特徴的な「がじぇっとるねさす」ブランドから，マイコン・ボードGR-MANGOが発売されました．
マイコン・ボードGR-MANGOの特長とサンプル・プログラムを動かしてGR-MANGOの動作概要を簡単に紹介します．

● ラズベリー・パイとレイアウト互換

ラズベリー・パイ4の形状，拡張コネクタのピン・レイアウトに互換性があり(**写真1**)，ラズベリー・パイ用のケースや，PoE HATなどの拡張基板をそのまま装着できます．

マイコン・ボードGR-MANGOのOSはLinux系ではなく，Mbed OSを使用します．CMSISによるGPIO制御によって，センサやモジュールが使用できるほか，A-Dコンバータによってアナログ信号を扱うこともできます(**図1**)．

● 画像処理が得意なMbedボード

プログラム開発は，Arm社が提供するMbed環境で行います．

一般的なMbedボードはCortex-M，動作周波数は100MHz程度が多いなか，GR-MANGOのCPUはCortex-A9コアで動作周波数が528MHzのRZ/A2M(ルネサス エレクトロニクス)を使ったMbedボードに仕上がっています(**表1**)．

● GR-MANGOのI/O

比較的安く入手できるラズベリー・パイ・カメラ(MIPI CSI-2)による画像入力，Micro HDMIによる画像と音声の出力が可能です．その他にUSB Type A，USB Type C，Micro SD，オーディオ・ジャック(4極)，Ethernet，CAN，Groveコネク

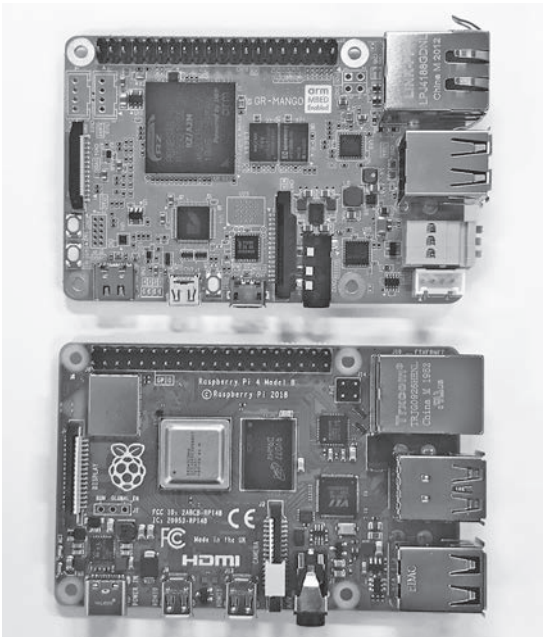


写真1　GR-MANGO(上)とラズベリー・パイ4B(下)．コネクタや端子のレイアウトがほぼ同じで汎用ケースなど流用できる場合が多い

クラウド・サービスThingSpeakとメッセージ・アプリSlackを活用する

obnizを用いた温度測定/通知システム

使用機材
Obniz Board(マイコン・ボード)
MCP9701(温度センサ)

吉澤 巧，松本 佳宣
(慶應義塾大学大学院 理工学研究科)

マイコン・ボードobnizをIoTシステムに応用してみました．obnizの特徴は，Wi-Fi経由の双方向通信が可能でディジタル端子に電流を多めに流せる点です．

obnizで温度を測定し(MCP9701)，測定した温度をクラウド・サービスThingSpeakにアップロードすることにより，遠隔地でも容易に測定値を確認することができます．さらに，温度が設定値を超えたときに，メッセージ・アプリSlackでユーザに通知するシステムを試作しました．

● obnizをIoTに応用

近年，IoT(Internet of Things)と呼ばれる，さまざまなモノをインターネットに接続する技術が盛んに応用されています．

センサ技術や通信技術の発展により，IoTの対象となっているものは身の回りに広がっており，パソコンやスマートフォンといった通信端末だけでなく，工場における状態監視センサや医療現場におけるウェアラブル・センサなど，さまざまな場所で活躍するようになりました．

今回の試作したシステムは，スマートフォンからマイコンを制御してセンサの測定とSlackでユーザに通知ができることです．これまでスマートフォンからマイコンを制御することは，比較的手間のかかるものでした．しかし，obniz Boardはそれを容易に実現できるマイコン・ボードです．

● マイコン・ボードobniz

obniz Boardとは，obniz Cloud上のライブラリの仕組みを使い，測定や制御ができます．アクチュエータなどもネットワーク経由で制御できます．また，クラウド上のシステムを利用してobniz Boardにプログラムを書き込むため，開発後のアップデートは簡単に行うことができます．

obnizは公式デバイスの購入，もしくは既存のマイコンにobnizOSをインストールすることでobinz Cloudを使用することができます．今回は公式デバイスのobniz Boardを使用して開発を行います．補足をすると，obniz OSは1ユーザあたり1台まで無料で使用することができます(以降obniz Boardをobnizと表記)．

● obnizのセットアップ

obnizを電源に接続すると，**写真1**のようになります．obnizが起動したら接続したいWi-FiのSSIDを選択して，パスワードを入力指定します．ディスプレイにQRコードと8桁の番号が表示されたら接続完了です．なお，最新OSの場合は画面が異なる可能性がありますが，SIMPLE METHODを選択すると同様の手順になります．

※記事で紹介したプログラムはhttps://eleki-jack-iot.com/からダウンロードできます．