【クラウド経由の温湿度モニタリングシステム レポート】

【報告者】

東京国際工科専門職大学　工科学部　情報工学科　グループ2

・TK210196　中山友貴

・TK210531　山田達成

・TK210546 田中琉唯

・TK210562　石倉惇杜

・TK220205　湯浅太貴

【作成日】

2023/07/31

**1. 実装システムの「依頼内容」**

　10秒に一度複数のセンサを用いてセンサノード周囲の温湿度を取得し、その取得したデータをインターネットサーバに送信する。その後、インターネットサーバで受信した温湿度データをファイルで保存する。スマホからインターネットサーバへデータ表示リクエストが届いた場合、今まで蓄積していた温湿度データを外部ノードであるスマホ上の画面に表形式で表示する。その際、ブラウザを用いてインターネットサーバへ、スマホから表示リクエストを送信し、スマホ上には温湿度それぞれの平均値も表示する。確認ボタンを追加し、表と平均温湿度の表示/非表示を行う。温度・湿度データを追加する際は外部ノード上のブラウザ（Google Chrome）から行う。センサノードと外部ノードを複数にした際も動作するように変更する。

**2. システム全体の実装方法**

　まず、Raspberry Piを用いてDHT11とBME280を使用し、DHT11とBME280から温湿度データを取得し、Socket通信でサーバに送信する。次に、WindowsPCでRaspberry PiからSocket通信によって送られた温湿度データを受信し、CSV形式で保存をする。

次に、二台目のWindowsPCでFlaskを使用し、WebブラウザにHTML形式のデータを表示するプログラムを実行する。最後に、Flask を用いてスマホからインターネットサーバへ表示リクエストを送信し、送られたらインターネットサーバはこれまで蓄積された温度・湿度データをインターネットサーバとスマホの画面に表示する。

**3. 全体構成**

　センサノードはRaspberry Piを用いてDHT11、BME280（ともに温湿度計測に使用）、サーバノードはPC（Windows）を2台（受信：１台、表示プログラム実行：１台）、外部ノードはiPhoneまたはAndroid（インターネットサーバへ表示リクエストを送る際に使用）、Webブラウザ（スマホからインターネットサーバへの表示リクエストを送る際に使用）を用いる。これら６台のノードは大学無線LAN、または個人宅の無線LANで接続するものとする。また、全体を「センサからデータ収集」と「データ表示」に分け、２つは同時に動作させないものとする。

　今回教員からの追加要望により追加したセンサはBME280であり、外部ノードにはWebブラウザを追加した。追加要望を受ける前はセンサノードでは温湿度を取得する際はDHT11のみを使用していたが、BME280を追加することで、各々のセンサから複数個の温湿度データを取得することができ、温湿度データの信憑性を上げることが可能になった。同様に外部ノードでは追加要望を受ける前はiPhoneのみ使用していたが追加要望によりWebブラウザを追加することが必要になった。

**4. ディレクトリ構成**

　本システムのディレクトリ構造をRaspberryPi側とローカルPC側に分けて説明する。図１に本システムのファイル構成を示す。

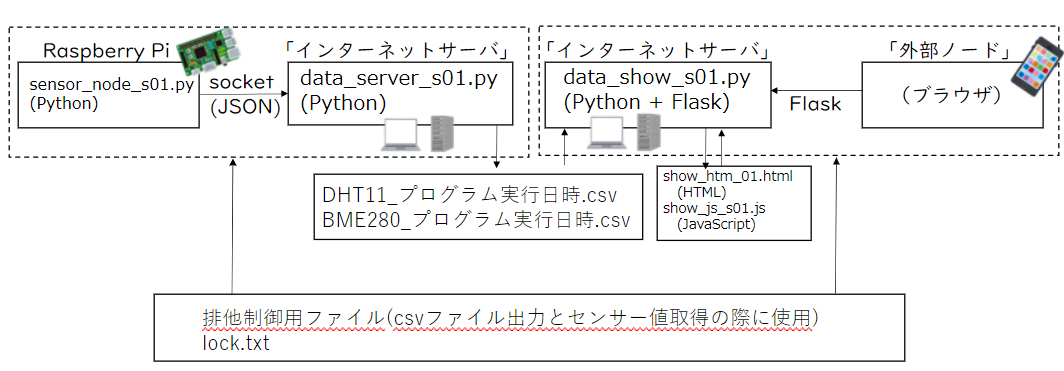


図１：本システムのファイル構成

4.1 RaspberryPi側

　sensor\_node\_s01.pyはRaspberryPiに接続されているDHT11とBME280から温度、および湿度データを取得し、各々のデータをjson形式に変換する。その後、ソケット通信を用いてローカルPC側に各々のjson形式のデータを送信するファイルである。lock.txtは排他制御に用いるファイルである。図２に RaspberryPi側のディレクトリ構造を示す。

　・sensor\_node\_s01.py

　・lock.txt

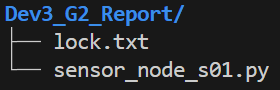


図２：RaspberryPi側のディレクトリ構造

4.2 ローカルPC側

　dataディレクトリには BME280\_プログラム実行日時.csv、およびDHT11\_プログラム実行日時.csvが存在する。 BME280\_プログラム実行日時.csvにはBME280で取得した温度、および湿度データ、 DHT11\_プログラム実行日時.csvにはDHT11で取得した温度、および湿度データが保存される。statickディレクトリには、cssディレクトリとjsディレクトリが存在する。 cssディレクトリには表示内容のセンタリングやボタンのスタイルやテキスト表示/非表示や表、および横線をクラスとして定義しているstyle.css、jsディレクトリにはボタンの挙動や温度、および湿度の平均値計算やデータが追加されたときの表の追加処理をしているshow\_js\_s01.jsが存在する。templatesディレクトリにはWebブラウザの表示内容を記述しているshow\_htm\_01.htmlが存在する。その他に、 RaspberryPi側から送信された json形式のデータを受け取り、DHT11のデータを DHT11\_プログラム実行日時.csvに、BME280のデータを BME280\_プログラム実行日時.csvに保存するdata\_server\_s01.pyが存在する。また、リダイレクト処理やWebブラウザ上からのフォーム入力処理やshow\_htm\_01.htmlの内容を表示するためのdata\_show\_s01.pyが存在する。また、排他制御を行うためのlock.txtが存在する。lock.txtを用意した理由は、BME280とDHT11を用いたデータ測定する際の競合を防ぐためである。図３にローカルPC側のディレクトリ構造を示す。

　・BME280\_プログラム実行日時.csv

　・DHT11\_プログラム実行日時.csv

　・style.css

　・show\_js\_s01.js

　・show\_htm\_01.html

　・data\_server\_s01.py

　・data\_show\_s01.py

　・lock.txt

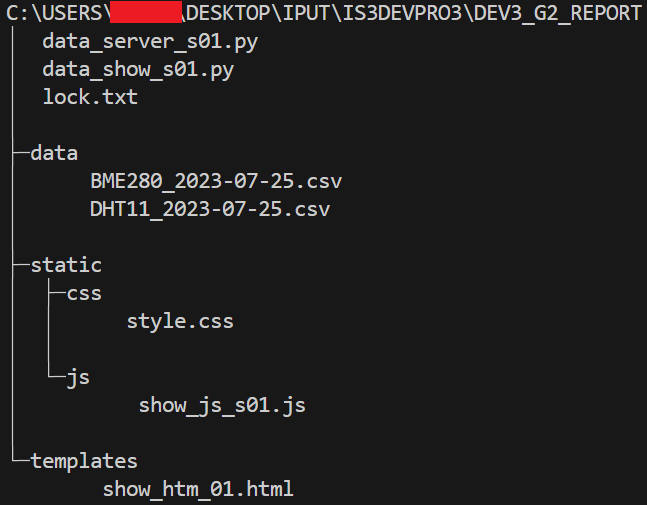


図３：ローカルPC側のディレクトリ構造

**5. センサノード上の機能**

　2班では、イテレーション1の後にお客様の発言で挙げられていた「機械的に集計するためにデータが取れない場合は10秒後などに値を記録してほしい」、「JSON形式に変換しているかの確認をしてほしい」と、イテレーション2の後にお客様の発言で挙げられていた「センサノードを複数にして動作してほしい」という意見に対応するべく、「Raspberry Piを使用してDHT11とBME280センサから温湿度データを取得する機能」と「Socket通信を介してサーバノードに温湿度データを送信する機能」の2つを採用した。

　センサノード上の機能の実現方法として、2班ではRspberry Pi上で動作するPythonのバージョンは3.9.2、PC上で動作するPythonのバージョンは3.11.3とした。以下にセンサノード上で使用したライブラリと機能の説明を挙げる。

adafruit\_dht…DHT11の温湿度データを取得するためのAdafruitのライブラリ。

インストール方法としてはターミナル上で以下のコマンドを実行する。

(コマンド：pip3 install adafruit-circuitpython-dht)

adafruit\_bme280…bme280の温湿度,気圧データを取得するためのAdafruitのライブラリ。

インストール方法としてはターミナル上で以下のコマンドを実行する。

(コマンド：pip3 install adafruit-circuitpython-bme280)

board … ボード固有のピン指定に使用するPython標準ライブラリ。

time … sleep（待ちの処理）等で使用するPython標準ライブラリ。

datetime … 日時関連の機能で使用するPython標準ライブラリ。

sys … コマンドライン引数等のシステム関連の機能で使用するPython標準ライブラリ。

json … JSON形式のデータを処理するためのPython標準ライブラリ。

decimal … 小数点の演算に使用するPython標準ライブラリ。

fasteners … 排他制御用のPythonライブラリ。

**5. センサノード上の機能の実装方法とその実装のテスト**

　2班が実装したセンサノード上の「機能」のソフトウェアの実装方法としては、sensor\_node\_s01.py上でDHT11とBME280の温湿度データを取得し、decimalモジュールのquantize()メソッドを使用して小数点以下１桁まで表示させる。そのデータをJSON形式に変更し、ソケット通信を用いてDHT11とBME280から取得した温湿度データを指定したサーバに送信する。その時fastenersモジュールを使用して排他制御を行い、複数のプロセスが同時にファイルにアクセスして競合を起こすのを防いでいる。

　テスト内容としては、イテレーション１の後にお客様の発言で挙げられていた「機械的に集計するためにデータが取れない場合は10秒後などに値を記録してほしい」、「JSON形式に変換しているかの確認をしてほしい」と、イテレーション2の後にお客様の発言で挙げられていた「センサノードを複数にして動作してほしい」という意見に対応するべく、「取得した温湿度データが10秒おきに表示されているかの確認」、「取得した温湿度データがJSON形式に適切に変換されているかの確認」を行った。

　具体的に、1つ目の「取得した温湿度データが10秒おきに表示されているかの確認」のテストでは、DHT11とBME280センサから取得した温湿度データがターミナル上に10秒おきに表示されているかの確認を行った。図４に各センサから取得した温湿度データを10秒おきに表示したテスト結果を示す。

グラフィカル ユーザー インターフェイス

自動的に生成された説明

図４：各センサから取得した温湿度データを10秒おきに表示したテスト結果

　2つ目の「取得した温湿度データがJSON形式に適切に変換されているかの確認」のテストでは、DHT11とBME280センサから取得した温湿度データをJSON形式に変換するコードとJSON形式に変換されたデータを確認するためにprint文を用いてターミナル上に表示するコードを追記し、正しくJSON形式に変換されているかの確認を行った。

　図５,および図６に各センサから取得した温湿度データがJSON形式に変換していることを確認したテスト結果を示す。

コンピューターのスクリーンショット

自動的に生成された説明

図５：DHT11センサから取得した温湿度データをJSON形式に変換したテスト結果

コンピューターのスクリーンショット

自動的に生成された説明図６：BME280センサから取得した温湿度データをJSON形式に変換したテスト結果

　センサノード上の不具合の対処として、BME280の温湿度取得エラーをDHT11の温湿度取得エラーの対応と同様にした。つまり、温湿度データの取得ができるまでループを回している。

**7. サーバノード上の機能**

　2班では、イテレーション1の後にお客様の発言で挙げられていた「インターネットサーバで受信した温湿度データをCSV形式のファイルで保存してほしい」という意見に対応するべく、「センサノードからSocket通信で温湿度データを取得する機能」と「受け取った温湿度データをCSVファイルとして保存​する機能」、「保存したCSVをFlaskを用いてブラウザに表示​する機能」の3つを採用した。サーバノード上の機能の実現方法として、2班ではRspberry Pi上で動作するPythonのバージョンは3.9.2、PC上で動作するPythonのバージョンは3.11.3とした。以下にサーバノード上で使用したライブラリと機能の説明を挙げる。

csv … csvファイルを扱うために使用するPython標準ライブラリ。

time … sleep（待ちの処理）等で使用するPython標準ライブラリ。

datetime … 日時関連の機能で使用するPython標準ライブラリ。

sys … コマンドライン引数等のシステム関連の機能で使用するPython標準ライブラリ。

json … JSON形式のデータを処理するためのPython標準ライブラリ。

fasteners … 排他制御用のPythonライブラリ。

**8. サーバノード上の機能の実装方法とその実装のテスト**

　2班が実装したサーバノード上の「機能」のソフトウェアの実装方法としては、data\_server\_s01.py上でサーバのIPアドレスとポート番号を指定し、接続を待機する。DATA\_DIR変数で指定されたディレクトリにファイルが保存されるようにしており、datetimeモジュールによってファイル名に現在の日付が含まれるような設定にしている。

また、csv\_write\_for\_DHT11およびcsv\_write\_for\_BME280関数内で、温湿度データをCSVファイルに書き込むようにしている。fastenersモジュールを使用して排他制御を行い、同時にファイルにアクセスして競合を起こすのを防いでいる。server\_data\_recv関数内で送信されたJSON形式のDHT11とBME280の温湿度データを抽出し、CSVファイルに書き込む処理を行っている。その後、Flaskを用いて、Webブラウザ上に表示されるようにしている。また、ユーザがWebブラウザからデータを入力することで、入力されたデータがCSVファイルに保存されるようになっている。

　テスト内容としては、「インターネットサーバで受信した温湿度データをCSV形式のファイルで保存してほしい」という意見に対応するべく、「センサノードからSocket通信で温湿度データを取得できているかの確認」、「受け取った温湿度データをCSVファイルとして保存​されているかの確認」、「保存したCSVがFlaskを用いてブラウザに表示されているかの確認​」を行った。

　具体的に、1つ目の「センサノードからSocket通信で温湿度データを取得できているかの確認」のテストでは、センサノードとサーバノードのターミナル上で表示されているIPアドレスが同じであり、センサノードから取得した温湿度データがJSON形式で送信されていることから、Socket通信で温湿度データを取得できていることを確認した。図７にセンサノードから取得した温湿度データをJSON形式で送信したテスト結果を示す。

テキスト

自動的に生成された説明

図７：センサノードから取得した温湿度データをJSON形式で送信

　2つ目の「受け取った温湿度データをCSVファイルとして保存​されているかの確認」のテストでは、センサノードとサーバノードのターミナル上で表示されている日付、また、保存されたCSVファイルの日付が同じであることから、DHT11とBME280センサから取得した温湿度データが正しくCSVファイルとして保存されていることが分かる。図８に受け取った温湿度データをCSVファイルとして保存​したテスト結果を示す。

グラフィカル ユーザー インターフェイス

中程度の精度で自動的に生成された説明

図８：受け取った温湿度データをCSVファイルとして保存

　3つ目の「保存したCSVがFlaskを用いてブラウザに表示されるかの確認​」のテストでは、Flaskフレームワークを使用してWebブラウザ上に温湿度データ等を表示している。ブラウザとターミナル上で表示されているIPアドレスが同じであることから、Flaskを用いてCSVがブラウザに表示されていることが分かる。図９に受け取った温湿度データをCSVファイルとして保存したテスト結果を示す。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明図９：受け取った温湿度データをCSVファイルとして保存

　サーバノード上の不具合の対処として、lock.txtがないと排他制御ができなかったエラーとdataディレクトリがないとCSVが生成できないエラーを修正した。4.2節で説明したディレクトリ構造にすることによってエラーを修正した。

**9. センサノード、サーバノード、外部ノードの接続による動作検証**

　本章では、2班が行ったセンサノード、サーバノード、外部ノードの接続による動作検証について記述する。初めにsensor\_node\_s01.py上でDHT11とBME280の温湿度データを取得する。取得したデータをJSON形式に変換した後、ソケット通信を用いてDHT11とBME280から取得した温湿度データを指定したサーバに送信する。

　次に、data\_server\_s01.py上でサーバのIPアドレスとポート番号を指定し、接続を待機する。また、csv\_write\_for\_DHT11およびcsv\_write\_for\_BME280関数内で、温湿度データをCSVファイルに書き込むようにしている。server\_data\_recv関数内で送信されたJSON形式のDHT11とBME280の温湿度データを抽出し、CSVファイルに書き込む処理を行っている。

　次に、Flaskを用いて、Webブラウザ上に表示している。show\_htm\_01.htmlとshow\_htm\_01.jsとsttle.cssでブラウザに表示される入力フォーム・温湿度データの表・温湿度の平均値・確認ボタンの表示を行っている。ここでユーザがWebブラウザ内に存在する入力フォームからデータを入力することで、入力されたデータがCSVファイルに保存されるようになっている。また、Flaskを用いて複数の外部ノード(Smartphone, Android)で表示されるようにしている。図１０に外部ノードを複数にしてブラウザに表示した結果を示す。

ダイアグラム

自動的に生成された説明 図１０：外部ノードを複数にしてブラウザに表示した結果

　主な不具合の内容として、lock.txtがないと排他制御ができなかったエラーとdataディレクトリがないとCSVが生成できないエラーを修正した。4.2節で説明したディレクトリ構造にすることによってエラーを修正した。

**10. DHT11の精度検証**

指摘事項としてDHT11の精度をどう担保するのかということが挙がった。

そこで、DHT11のメーカー公称値の精度を確認するとともに、他のセンサ等も使用し、DHT11の精度の検証を行った。

使用したセンサ類は以下の通りである。

* DHT11（精度：湿度：25℃で±5％、温度： 25℃で±2℃）
* SwitchBot温湿度計

（センシリオン社製SHTC3、温度精度：±0.4°C（-20°C～0°Cにおいて）、±0.2°C（0°C～65°Cにおいて）、±0.3°C（65°C~80°Cにおいて）

湿度精度：±2%（25°C、10%~90%RHにおいて）、±4%（25°C、0%~10%, 90%~99%RHにおいて））

* 謎メーカの温湿度計
* IKEA温度計
* ダイソー温度計
* 空調機器
* MHI製ルームエアコン（定格出力2.2Kw（0.9Kw~3.0Kwで可変））
* 謎の中華メーカ製家庭用除湿機（170w）

扇風機で常に風を回し、温度ムラを無くす。

湿度設定(共通)

除湿機の設定を45％に設定

検証結果

温度設定：ドライ24℃の場合

DHT11

温度：23~24℃

湿度：50％

switchBot温湿度計

温度：23.3℃

湿度：44％

IKEA温度計

温度：23℃

ダイソー温度計

温度：23.4℃

謎メーカの温湿度計

温度：23.0℃

湿度：50％

温度設定：冷房18℃の場合

DHT11

温度：19℃

湿度：48％

switchBot温湿度計

温度：19.6℃

湿度：40％

IKEA温度計

温度：20℃

ダイソー温度計

温度：20.1℃

謎メーカの温湿度計

温度：19.7℃

湿度：44％

温度設定：暖房30℃の場合

DHT11

温度：30~31℃

湿度：48~50％

switchBot温湿度計

温度：30.6℃

湿度：41％

IKEA温度計

温度：30℃

ダイソー温度計

温度：30.6℃

謎メーカの温湿度計

温度：30.5℃

湿度：46％

以上の結果から、

DHT11の温度の精度は実測で±1℃程度

DHT11の湿度の精度は実測で最大10％ほどの開きが見られた。

2回目の検証では、BME280, ADT7410を追加し、検証を行った。

BME280精度、分解能公称値：

温度精度：±1℃　分解能：0.01℃　　温度測定範囲：-40℃～85℃

湿度精度：±3％　分解能：0.008%　湿度測定範囲：0％～100％

ADT7410精度、分解能公称値：

温度精度：±0.5℃　分解能：0.0078℃　　温度測定範囲：-55℃～150℃

検証結果

温度設定：冷房18℃の場合

DHT11

温度：18.1℃

湿度：50％

switchBot温湿度計

温度：18.1℃

湿度：43％

IKEA温度計

温度：19℃

ダイソー温度計

温度：18.3℃

BME280

温度：17.54℃

湿度：45.55％

ADT7410

温度：17.9375℃

温度設定：ドライ24℃の場合

DHT11

温度：23.9℃

湿度：51％

switchBot温湿度計

温度：23.6℃

湿度：47％

IKEA温度計

温度：24℃

ダイソー温度計

温度：23.7℃

謎メーカの温湿度計

温度：23.5℃

湿度：52％

ADT7410

温度：23.5625℃

BME280

温度：23.12℃

湿度：45.55％

温度設定：暖房30℃の場合

DHT11

温度：31.7℃

湿度：53％

switchBot温湿度計

温度：30.7℃

湿度：44％

IKEA温度計

温度：29℃

ダイソー温度計

温度：30.1℃

謎メーカの温湿度計

温度：29.9℃

湿度：49％

ADT7410

温度：31.625℃

BME280

温度：30.93℃

湿度：42.06％

検証について

精度検証のパートにて、ドライ24℃設定で検証していたパートがあった。DHT11の公称値は25℃での計測であるが、なぜ24℃設定なのか。

検証した部屋の空調負荷と、エアコンの最低出力が上手く釣り合うのが24℃設定の時であるためである。24℃よりも設定温度を上げると、室温の振れ幅が大きくなる。

この件に関してメーカー(三菱重工冷熱株式会社)に問い合わせを行った。

回答(要約、冷房運転の場合)：

空調負荷(熱源など)**＞**エアコンの最低出力(0.9Kw)の場合は、インバータ制御にて室温を細かく調節できるが、空調負荷**＜**エアコンの最低出力(0.9Kw)の場合は、インバータ制御で絞れる出力の最低値を空調負荷が下回ってしまう。

そのため、このままコンプレッサーを回して運転を続けてしまうと設定温度以上の室温低下が発生するため、コンプレッサーを停止、または最低出力で動かすというON, OFF制御のような状態に切り替わってしまう。

インバータ制御よりも、 ON, OFF制御のような状態のほうが室温を一定にキープすることは難しいため、室温のばらつきが激しくなる。

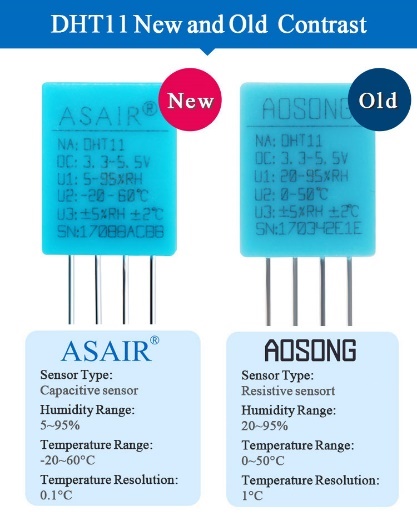
との回答をいただいた。

また、室温検知に関しては、温湿度センサはその場所の温度、湿度を測っているため、温度、湿度ムラなどには1つのセンサでは対応できず、複数のセンサを複数の場所に設置し、複合的にセンシングし、判断する必要がある。

DHT11の個体差について

SEED社の資料によると、DHT11は見た目が同一で中身とスペックが違うものが2種類存在するようである。

この件に関しても、精度に影響していた可能性が高い。



新旧DHT11のデータ

https://jp.seeedstudio.com/Grove-Temperature-Humidity-Sensor-DHT11.html

検証結果を踏まえて

* DHT11の温度の精度は実測で±1℃程度
* DHT11の湿度の精度は実測で最大10％ほどの開き
* 場合によっては高い精度が出ることもある
* この差をどう捉えるかは用途次第
* 静電容量方式のDHT 11を使用すると、比較的高い精度が得られるかもしれない。（未検証）

**10. 所感**

　今回は、主にセンサ関連の精度の検証を担当した。本来、温湿度センサの精度検証は、しっかりとした実験環境を用意できなかった。

家庭用のルームエアコンは、設定温度の厳密に追従するようにはできていない。(下記の画像はドライ：24℃設定の場合)また、外的要因により、室温、湿度を一定に保つことは非常に難しい。また、温湿度センサの類も精度がしっかりと担保されたものではないため、精度の高い検証ができなかったのが心残りである。

しかし、自分の持っている知識を活かせてよかったと感じた。

