令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」

ＩｏＴ技術応用教材資料２

本ＩｏＴ技術応用教材資料は、文部科学省の教育政策推進事業委託費による委託事業として、一般社団法人全国専門学校情報教育協会が実施した令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」の成果物です。

情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

目　次

IoT技術応用 3
はじめに 3

IoT技術応用_実習１ 8
Part1 8
Part2 17
Part3 28
Part4 37

IoT技術応用_実習２ 49
導入編 49
Part1 56
Part2 64
Part3 72
Part4 80

IoT技術応用_実習３ 88
導入編 88
Part1 94
Part2 102
Part3 110
Part4 118

ＩｏＴ技術応用_実習２

Part2


IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習１ デバイス制御
実習２ クラウド接続
実習３ システム統合
Mirai no tobira


IoT技術応用
実習２ クラウド接続
Part１：Wi-Fi接続とネットワーク基礎
Part２：センサーデータ取得とシリアル通信
Part３：クラウドサービス連携
Part４：データ可視化と分析
Mirai no tobira

実習２ クラウド接続

IoT技術応用

実習１ デバイス制御 → **実習２ クラウド接続** → 実習３ システム統合

Part１：Wi-Fi接続とネットワーク基礎

☞ **Part２：センサーデータ取得とシリアル通信**

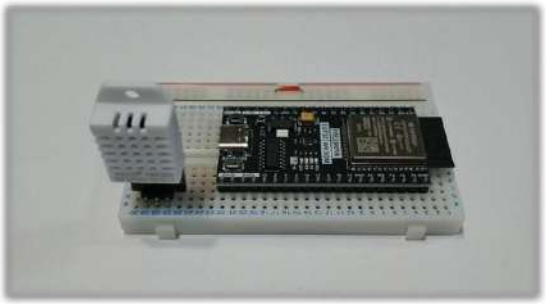
Part３：クラウドサービス連携

Part４：データ可視化と分析

Part２：センサーデータ取得とシリアル通信

実習２ クラウド接続

高精度センサー DHT22 を使う



Part２:センサーデータ取得とシリアル通信の達成目標

実習２ クラウド接続

Part１の「Wi-Fi接続」に「データ取得」機能を追加します

- ✓ 高精度な DHT22 センサーを ESP32 に配線できる
- ✓ DHT ライブラリを使って温度・湿度データを取得できる
- ✓ シリアルモニタで取得したデータを確認できる
- ✓ センサーの読み取りエラーを検出できる

センサーの比較：DHT11 vs DHT22

実習２ クラウド接続

実習１の DHT11（青）と 今回の DHT22（白）の違い



	DHT11 (実習1)	DHT22 (実習2)
測定範囲	狭い（−20〜60℃）	✅ 広い（−40〜80℃）
温度精度	±２℃	✅ ±0.5℃
湿度精度	±５%	✅ ±２%

※ 製品のバージョンによって異なり、実際は個体差もあります

ステップ１：ライブラリの確認

実習２ クラウド接続 Part1

DHT22 も、実習１で使った DHT11と同じライブラリで動作

- **ライブラリの確認**
  - ✓ 「ライブラリを管理」で「DHT sensor library by Adafruit」がインストール済みか確認
- **未インストールの場合**
  - ☞ 検索してインストールします

ステップ２：DHT22 の配線

実習２ クラウド接続 Part1

ESP32 に DHT22 センサーを接続します

- **ピンの確認：**センサーのピン（VCC, DATA, GND）を確認します
- **配線（合計３本）：**
  - DHT22 **VCC**（＋）→ ESP32 **3.3V または 5V**
  - DHT22 **DATA** → ESP32 **GPIO 4**（D4ピン）
  - DHT22 **GND**（－）→ ESP32 **GND**



ステップ3：プログラムの作成

実習 2 クラウド接続 Part1

Part 1 の Wi-Fi 接続コードに、センサー読み取り機能を追加

```
#include <WiFi.h>
#include <DHT.h>
const char* ssid = "...";
const char* password = "...";
#define DHTPIN 4 // DATAピンをGPIO 4に接続
#define DHTTYPE DHT22 // センサータイプを「DHT22」に変更
DHT dht(DHTPIN, DHTTYPE);
void setup() { Serial.begin(115200);
 dht.begin();
 WiFi.begin(ssid, password); ... }
void loop() { // 温湿度データを読み取り
 float humidity = dht.readHumidity(); float temperature = dht.readTemperature();
 if (isnan(humidity) || isnan(temperature)) { Serial.println("センサー読み取りエラー"); delay(2000);
return;}
Serial.println("--- 測定データ ---"); Serial.print("温度: "); Serial.print(temperature);
Serial.println(" ℃"); Serial.print("湿度: "); Serial.print(humidity); Serial.println(" %");
delay(2000); // DHT22のため2秒以上待つ }
```


エラーチェック：isnan()

実習 2 クラウド接続 Part1

センサーは 配線ミスや故障で正しく値を返せないことがある

→ その場合「**nan** (Not a Number)」という特殊な値が返される

```
if (isnan(humidity) || isnan(temperature)) {
 Serial.println("センサー読み取りエラー");
 return; }
```

- **isnan()** は、「この値はnanか？」をチェックする命令
- || は「または」という意味
  「もし湿度が nan、または 温度がnanなら」エラーとして処理する
- return; で、それ以上処理（データ表示）をせずにloopの最初に戻る

ポイント
センサーデータを扱うときは、必ずエラーチェックを入れる癖をつけましょう
これにより、異常なデータが送られるのを防げます

ステップ4：動作確認と実験

実習 2 クラウド接続 Part1

プログラムを書き込み、動作を確認します

1. プログラムを **ESP32** に**書き込む**
2. **シリアルモニタ**を開き、ボーレートを「**115200**」に設定する
3. Wi-Fi接続が成功した後、2秒ごとに温湿度が表示されるか確認する
4. **実験：**センサーに息を吹きかけたり、手で温めたりして、
   実習 1 の時と比べて値の変化に違いがあるか観察する

実習 2 Part 2 のまとめ

実習 2 クラウド接続 Part 2

- ✓ **高精度な DHT22 センサーの使い方を学んだ**
- ✓ **#define DHTTYPE を変えるだけで、**
  同じライブラリが使えることを確認した
- ✓ **isnan() を使ったセンサーのエラーチェック方法を学んだ**
- ✓ **センサーの特性（測定間隔）に合わせて delay() を設定する**
  重要性を理解した

NEXT

実習２ Part ３：クラウドサービス連携

実習2 クラウド接続

Part１の「Wi-Fi接続」と
Part２の「データ取得」が揃いました
Part３で、この2つを組み合わせ
測定したデータをクラウドに送信します

NEXT

IoT技術応用

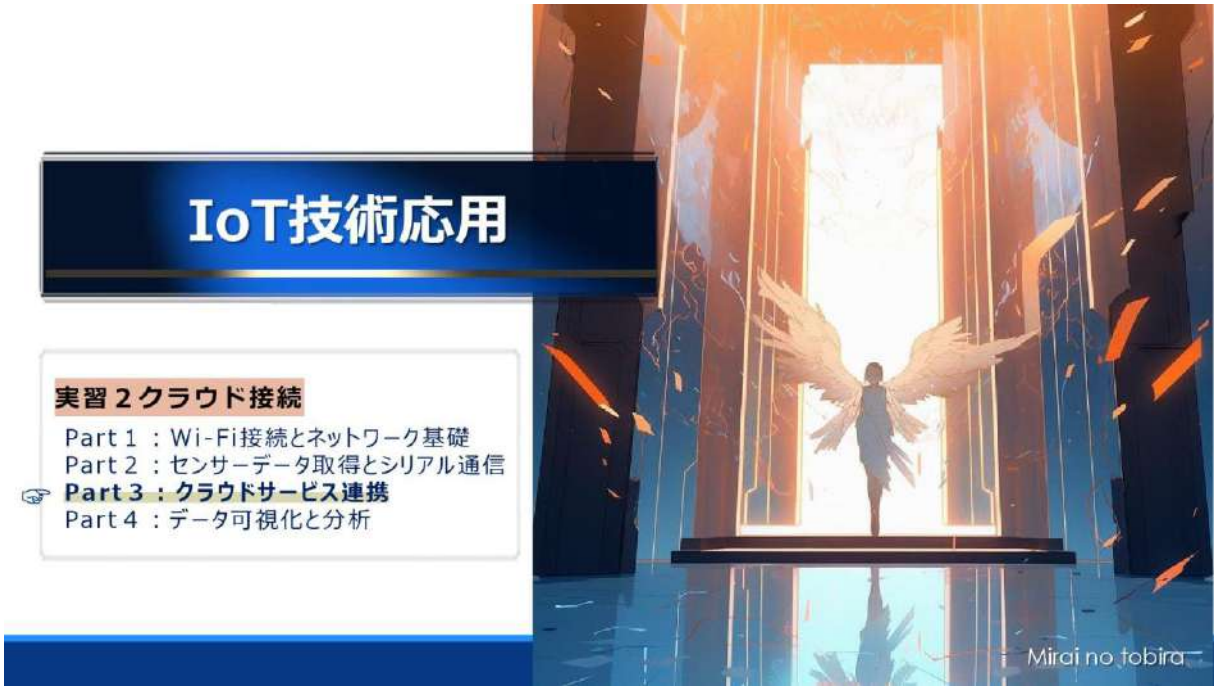
実習２クラウド接続
Part１：Wi-Fi接続とネットワーク基礎
Part２：センサーデータ取得とシリアル通信
Part３：クラウドサービス連携
Part４：データ可視化と分析



ＩｏＴ技術応用_実習２

Part3


IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習１ デバイス制御
実習２ クラウド接続
実習３ システム統合
Mirai no tobira


IoT技術応用
実習２クラウド接続
Part１：Wi-Fi接続とネットワーク基礎
Part２：センサーデータ取得とシリアル通信
Part３：クラウドサービス連携
Part４：データ可視化と分析
Mirai no tobira

実習２ クラウド接続

IoT技術応用

実習１ デバイス制御 → **実習２ クラウド接続** → 実習３ システム統合

Part１：Wi-Fi接続とネットワーク基礎
Part２：センサーデータ取得とシリアル通信
☞ **Part３：クラウドサービス連携**
Part４：データ可視化と分析

実習２ Part３：クラウドサービス連携

実習２ クラウド接続

測定したデータをクラウドに送信しよう





Part３：クラウドサービス連携 の達成目標

実習２ クラウド接続

Part１の「接続」とPart２の「測定」を組み合わせ
IoTの核心である「データ送信」を行います

- ✓ クラウドIoTサービス（Ambientなど）にアカウント登録
- ✓ データを受け取る「チャンネル」を作成
- ✓ ESP32 から温度・湿度データをクラウドに送信
- ✓ Webブラウザで、送信されたデータがグラフになるのを確認
- ✓ どこからでもデータにアクセスできる状態の作成

クラウドIoTサービスとは？

実習２ クラウド接続 Part３

IoTデバイスから送られてくるデータを
インターネット上で 受信、保存、可視化してくれるサービス

- ▶ **データ受信**：デバイスからの HTTP通信を受け取る窓口
- ▶ **データ保存**：送られてきたデータを時系列でデータベースに蓄積
- ▶ **可視化**　：蓄積したデータを自動でグラフ化
- ▶ **API**　：他のプログラムからデータを読み書きする機能

今回は、日本語で使いやすい「**Ambient**」か、高機能な「**ThingSpeak**」を使います

ステップ1：クラウドの設定 (Ambient)

実習 2 クラウド接続 Part 3

データを受け取る「器」を用意します

1. **Ambient** (ambidata.io) **でユーザー登録（無料）**
2. **ログインし「チャンネルを作る」をクリック**
3. **チャンネル名（例：「部屋の温湿度」）を入力して作成**
4. **作成されたチャンネルの「チャンネルID」と「ライトキー」をメモ**

⚠ **重要なキー**

チャンネルID：どのチャンネルに送るかの「宛先ID」／ **ライトキー：**データを書き込むための「パスワード」
これら２つは、次のプログラムで使います

ステップ2：送信プログラム （Ambient)

実習 2 クラウド接続 Part 3

Ambient への送信機能を追加します

Part２のコードに追加し、
ライブラリのインストール：
「ライブラリを管理」で
「Ambient ESP32 ESP8266 lib」
をインストール

```
#include <WiFi.h>
#include <DHT.h>
#include <Ambient.h>
WiFiClient client; Ambient ambient;
unsigned int channelId = 12345;
const char* writeKey = "abcdefg12345";
void setup() { ambient.begin(channelId, writeKey, &client); }
void loop() { float temp = dht.readTemperature();
 float humi = dht.readHumidity();
 if (isnan(temp) || isnan(humi)) {
  Serial.println("センサー読み取りエラー");
  delay(2000); return; }
 ambient.set(1, temp); ambient.set(2, humi);
 bool result = ambient.send();
 if (result) { Serial.println("✓ Ambientへの送信成功"); }
 else { Serial.println("✗ Ambientへの送信失敗"); }
delay(30000); }
```

ステップ3：動作確認

実習 2 クラウド接続 Part 3

プログラムを書き込み、すべてが連携するか確認します

1. シリアルモニタの確認：
   - ESP32 を起動し、シリアルモニタ（115200 baud）を開く
   - 「Wi-Fi接続成功」「Ambient への送信成功」と表示されるか確認
2. クラウドの確認：
   - Ambientの自分のチャンネルページをブラウザで開く
   - データが送信され、自動でグラフが描画されることを確認

成功の証
手元のセンサーで測ったデータが、インターネット上のグラフに反映されたら成功
スマートフォンからでも同じグラフが見えることを確認してみましょう

（参考）ThingSpeak の場合

実習 2 クラウド接続 Part 3

ThingSpeak を使う場合、ライブラリは不要、HTTP通信を自分で組み立てる

チャンネル設定：Field 1を「Temperature」、Field 2を「Humidity」として作成
APIキー：「Write API Key」をメモします

```
#include <HTTPClient.h>
String apiKey = "YOUR_WRITE_API_KEY";
void loop() {
 float temp = ... float humi = ...
 if (WiFi.status() == WL_CONNECTED) { HTTPClient http;
  String url = "http://api.thingspeak.com/update?api_key=" + apiKey + "&field1=" + String(temp) + "&field2=" +  String(humi);
  http.begin(url);
  int httpResponseCode = http.GET();
  Serial.print("応答コード: "); Serial.println(httpResponseCode);
  http.end();
  }
 delay(20000);
}
```

トラブルシューティング

実習２ クラウド接続 Part３

送信に失敗する場合の確認項目例

1. APIキー
2. 送信間隔
3. Wi-Fi接続
4. ファイアウォール

実習２ Part３ のまとめ

実習２ クラウド接続 Part３

- ✓ クラウドIoTサービスの役割（受信・保存・可視化）を理解した
- ✓ Ambient / ThingSpeak でチャンネルを設定しAPIキーを取得
- ✓ ライブラリ (Ambient) や HTTP通信 (ThingSpeak) を使ってデータを送信した
- ✓ センサーデータをクラウド上でリアルタイムにグラフ化できた
- ✓ 「モノ」と「クラウド」が繋がる IoT の基本形を完成させた

NEXT

実習２ Part ４：データ可視化と分析

実習2： クラウド接続

実習２ Part４では、
クラウドに蓄積したデータを「分析」します。
データを「見る」から「読む」へ
データからどんなことが分かるか、
その意味を読み解いていきます

NEXT

IoT技術応用

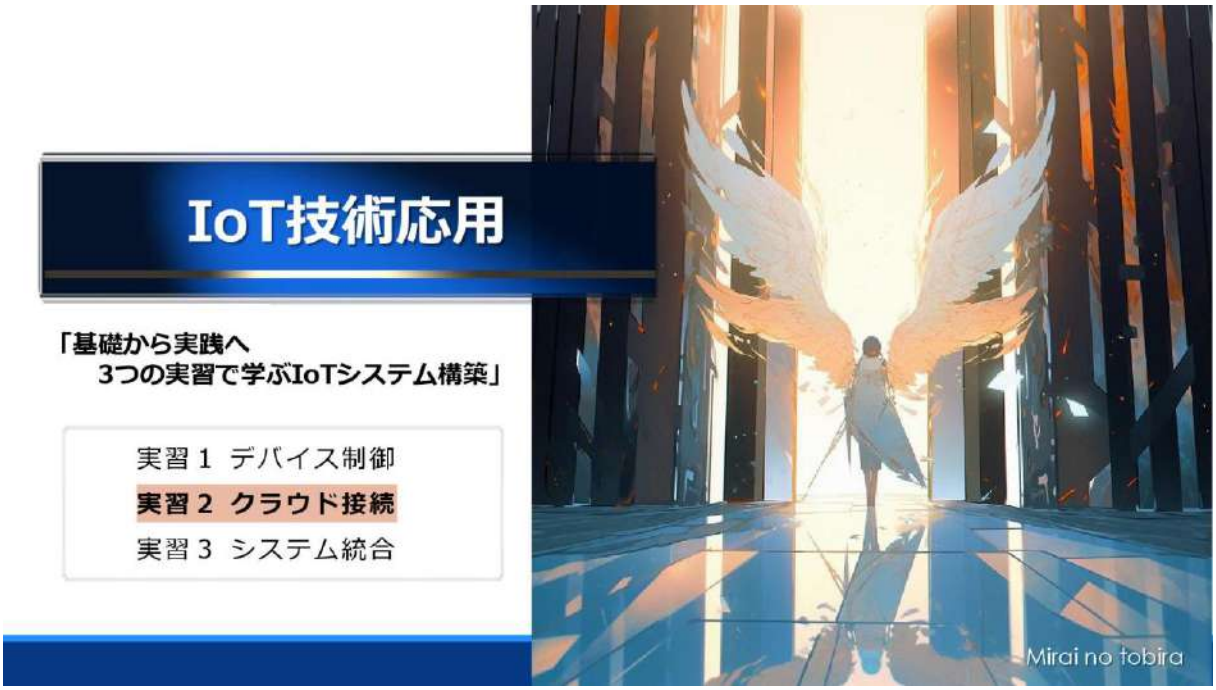
実習２ クラウド接続

Part１ ：Wi-Fi接続とネットワーク基礎
Part２ ：センサーデータ取得とシリアル通信
Part３ ：クラウドサービス連携
Part４ ：データ可視化と分析



ＩｏＴ技術応用_実習２

Part4


IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習１ デバイス制御
実習２ クラウド接続
実習３ システム統合
Mirai no tobira


IoT技術応用
実習２ クラウド接続
Part１：Wi-Fi接続とネットワーク基礎
Part２：センサーデータ取得とシリアル通信
Part３：クラウドサービス連携
Part４：データ可視化と分析
Mirai no tobira

実習２ クラウド接続

IoT技術応用

実習１ デバイス制御 → **実習２ クラウド接続** → 実習３ システム統合

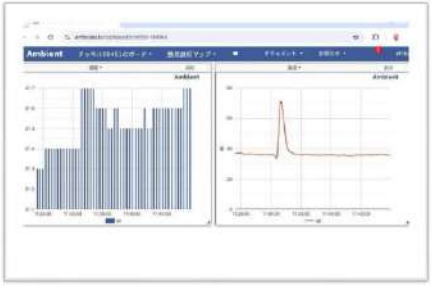
Part１：Wi-Fi接続とネットワーク基礎
Part２：センサーデータ取得とシリアル通信
Part３：クラウドサービス連携
☞ **Part４：データ可視化と分析**

実習２ Part４：データ可視化と分析

実習２ クラウド接続

データから意味を読み取ろう





Part４：データ可視化と分析 の達成目標

実習２ クラウド接続

データを「集める」から「活かす」へ進みます

実習２の総仕上げです

- ✓ **クラウドサービスのグラフ機能を設定・操作できる**
- ✓ **蓄積された時系列データを見て、傾向を説明できる**
- ✓ **「なぜこうなったか？」をデータから考察できる**
- ✓ **「もっとこうしたい」というシステムの改善点を提案できる**

なぜ「可視化」するのか？

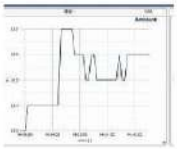
実習２ クラウド接続 Part４

数字の羅列（ログ）とグラフを比較してみましょう

数字の羅列（ログ）
→ 傾向がわからない

```
25.1, 55.2
25.0, 55.3
24.9, 55.3
25.2, 55.1
```

グラフ（可視化）
→ 一目で傾向がわかる



- ▶ **理解しやすくなる：** 直感的に状況を把握できる
- ▶ **傾向を発見できる：** 「夜中に温度が下がっている」などのパターンがわかる
- ▶ **異常を発見できる：** 「急に温度が跳ね上がった」などの異常値が目立つ
- ▶ **意思決定が支援できる：** データにもとづいた、合理的な判断ができる

時系列データの読み方

実習 2 クラウド接続 Part 4

グラフを見るときに注目するポイント

1. **トレンド（傾向）：**
   全体として上がっているか、下がっているか、横ばいか？
2. **周期性（パターン）：**
   「毎日同じ時間に上がって下がる」など、規則的な繰り返しはあるか？
3. **変動の大きさ：**
   値は安定しているか、それとも激しく上下しているか？
4. **異常値：**
   「一瞬だけ変な値」など、周りと明らかに違う点はないか？

ステップ1：グラフの読み取り

実習 2 クラウド接続 Part 4

Ambient/ThingSpeak のダッシュボードを見て データを読み取ってみましょう

1. クラウドサービスにログインし、自分のチャンネルを開く
2. 設定変更で、チャートの種類を選択
3. 分析したい内容がわかりやすいチャートや集計方法を選択する
4. さまざまなチャートを見て、何が分析できるかも考えてみる

ワークブックに記録しよう
「グラフの読み取り」の欄に、読み取った情報を記録してください

ステップ2：データ分析と考察

実習 2 クラウド接続 Part 4

データ考察の４ステップ

データを見たら、次は「なぜ？」と考える

1. **観察：** 例「温度が急激に高くなっているときがある」
2. **疑問：** 例「なぜその時間なのか？」
3. **仮説：** 例「日当たりが良くなったから？」「息を吹きかけたから？」
4. **検証：** 例「他の時間のデータも見る」「設置場所を変えてみる」

ステップ3：システムの改善提案

実習 2 クラウド接続 Part 4

データを見て気づいたことを元に、「もっとこうしたい」を考えます

例：

「温度が35度を超えたら危ない。LED等で知らせる機能が欲しい」
→ アラート機能の実装

「30秒ごとのデータは細かすぎる。5分ごとにして電池を長持ちさせたい」
→ Deep Sleepの実装

「温度と湿度だけでなく、明るさ（照度）も知りたい」
→ 複数センサーの追加

ワークブックに記録しよう

「システムの改善提案」の欄に、「追加したい機能」や「改善したい点」を自由に書いてください

実習２のまとめ

実習２ クラウド接続

この実習で、IoTの重要な流れを完成させました

センサーで測定 (Part2) → **Wi-Fiで送信** (Part1,3)
→**クラウドで分析** (Part4)

- ✓ **ESP32** を使って、モノを**インターネットに接続**した
- ✓ **クラウドサービス**を使って、**データを蓄積・可視化**した
- ✓ **時系列データ**を分析し、そこから意味を読み取る**「考察」**を体験した

NEXT

実習３ システム統合

実習2： クラウド接続

実習３では、
「データ分析」と**「システム統合」**を本格的に行います

Python を使った高度なデータ分析と
Node-RED を使った
リアルタイム・ダッシュボードの構築
に挑戦します


NEXT
IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習１ デバイス制御
実習２ クラウド接続
実習３ システム統合
Mirai no tobira

ＩｏＴ技術応用_実習３

導入編

IoT技術応用

「基礎から実践へ
3つの実習で学ぶIoTシステム構築」

実習1 デバイス制御
実習2 クラウド接続
☞ 実習3 システム統合



実習3 学習のゴール

IoT技術応用

- ☑ Pythonでデータを分析し、パターンを発見する
- ☑ Node-REDでビジュアルプログラミング、リアルタイム表示
- ☑ ESP32 + Python + Node-REDを統合した完全なシステム構築
- ☑ 自動アラート、予測モデルなどの高度な機能実装



実習１～３ ステップ

IoT技術応用

		達成したこと
実習１	デバイス制御	Arduino UnoとDHT11で**データを取得**できた
実習２	クラウド接続	ESP32でWi-Fi接続し、クラウドに**データを送信**できた
実習３	システム統合	蓄積したデータを**分析**し、**高度なシステム**を構築する

実習３に必要な機材・ソフトウェア

IoT技術応用

ハードウェア（実習2から継続）

- ESP32開発ボード
- DHT22温湿度センサー
- ブレッドボード、ジャンパーワイヤー
- PC（Windows、Mac、またはLinux）

ソフトウェア（新規インストール）

ソフトウェア	用途	ダウンロード	サイズ
Anaconda	Python、Jupyter Notebook	anaconda.com/download	約600MB
Node.js	Node-REDの実行環境	nodejs.org	約50MB
Node-RED	IoTフロー構築	Node.js後にコマンド実行	約30MB

IoT技術応用 実習３

実習３ ソフトウェア インストール

IoT技術応用

1. Anacondaのインストール

1. https://www.anaconda.com/download にアクセス
2. 利用するOS（Windows/Mac/Linux）に合わせてダウンロード
3. ダウンロードしたファイルを実行
4. インストール時、「Add Anaconda to my PATH」はチェック推奨
5. インストール完了後、「Anaconda Navigator」が起動することを確認）

2. Node.jsのインストール

1. https://nodejs.org/ にアクセス
2. 「LTS」版（推奨版）をダウンロード
3. ダウンロードしたファイルを実行し、指示に従ってインストール
4. コマンドプロンプト（またはターミナル）で確認:

```
node -v
```

バージョン番号が表示されればOK

実習３ ソフトウェア インストール

IoT技術応用

3. Node-REDのインストール

1. コマンドプロンプト（またはターミナル）を開く
2. 以下のコマンドを実行:

```
npm install -g node-red
```

3. インストール完了まで数分待つ
4. 起動確認:

```
node-red
```

「Server now running at http://127.0.0.1:1880/」と表示されればOK

5. ブラウザで http://localhost:1880 を開いて確認

IoT技術応用 実習３

事前準備チェックリスト

IoT技術応用

ソフトウェア
- ☐ Anacondaがインストールされている
- ☐ Jupyter Notebook が起動できる（Anaconda Navigator経由）
- ☐ Node.jsがインストールされている（`node -v` でバージョン確認）
- ☐ Node-REDがインストールされている（`node-red`で起動確認）

ハードウェア・データ
- ☐ ESP32とDHT22センサーが正常に動作する
- ☐ 実習2でAmbient/ThingSpeak にデータが蓄積されている
- ☐ AmbientまたはThingSpeak からデータをCSVでダウンロードできる

ネットワーク
- ☐ Wi-Fi接続が安定している
- ☐ インターネットに接続できる

実習３ 学習目標

IoT技術応用

知識面の目標
- Pythonの基本文法とデータ分析手法を理解する
- 統計分析（平均、標準偏差、相関）の意味を理解する
- MQTTプロトコルの仕組みを理解する
- ビジュアルプログラミングの概念を理解する

技術面の目標
- Jupyter NotebookとPandasでデータを扱える
- Matplotlibでグラフを作成できる
- Node-REDでデータフローを構築できる
- 複数の技術を統合したシステムを構築できる

応用面の目標
- データから意味ある情報を抽出できる
- 異常値を検出し、自動アラートを実装できる
- 予測モデルの基礎を理解する
- 実社会で使えるレベルのIoTシステムを設計できる


NEXT
IoT技術応用
実習３ システム統合
Part1：Pythonとデータ分析の基礎
Part2: IoTデータの高度な分析
Part3：Node-REDによるIoTフロー構築
Part4：統合IoTシステムの開発
Mirai no tobira

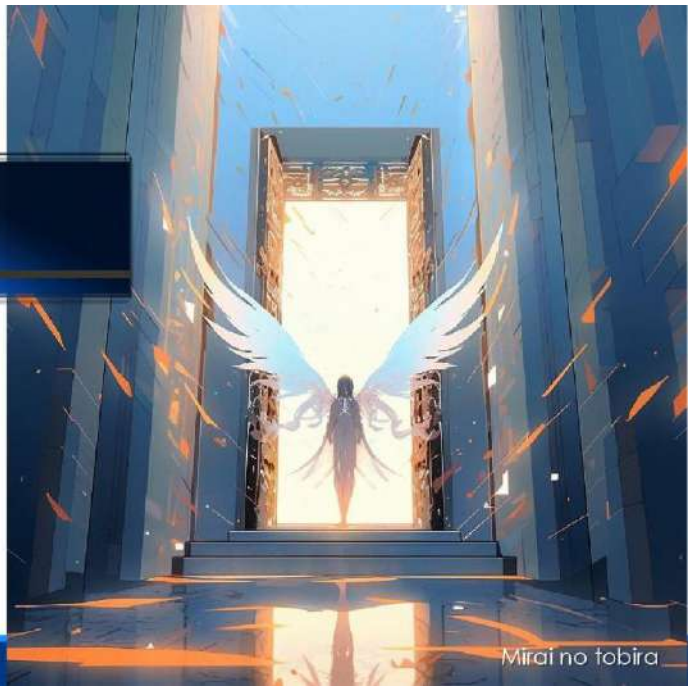
ＩｏＴ技術応用_実習３

Part1

IoT技術応用

「基礎から実践へ
3つの実習で学ぶIoTシステム構築」

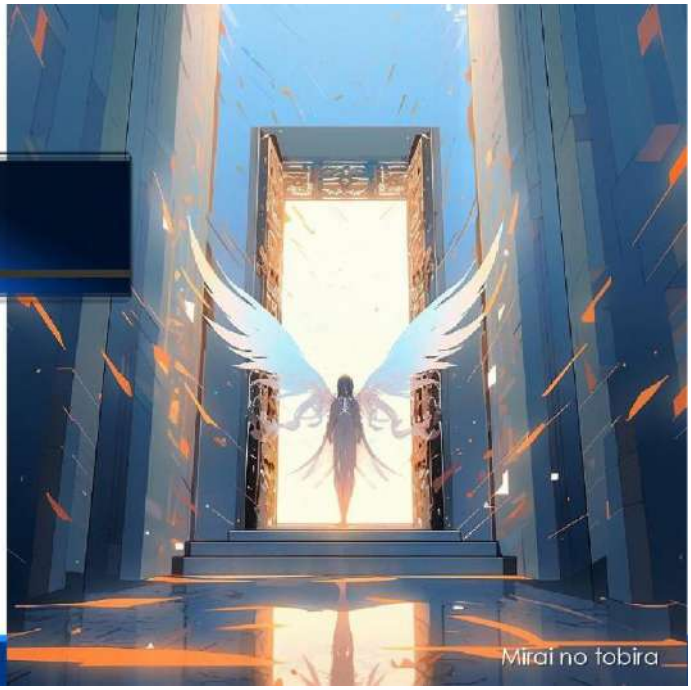
実習1 デバイス制御
実習2 クラウド接続
実習3 システム統合



IoT技術応用

実習3 システム統合

☞ **Part 1 ：Pythonとデータ分析の基礎**
Part 2 ：IoTデータの高度な分析
Part 3 ：Node-REDによるIoTフロー構築
Part 4 ：統合IoTシステムの開発



実習３ システム統合

IoT技術応用

実習１ デバイス制御 → 実習２ クラウド接続 → **実習３ システム統合**



- **Part１：Python とデータ分析の基礎**
- Part２：IoTデータの高度な分析
- Part３：Node-RED によるIoTフロー構築
- Part４：統合IoTシステムの開発

Part１：Python とデータ分析の基礎

実習３ システム統合

Jupyter Notebook、Pandas、データ読み込み

Part 1 ：Python とデータ分析の基礎 の達成目標

実習 3 システム統合

実習2で「蓄積」したデータを
「分析」する第一歩を踏み出します

- ✓ Jupyter Notebook で Python コードを実行できる
- ✓ Pandas ライブラリをインポートし、CSVを読み込める
- ✓ 読み込んだデータの「行数」や「列名」を確認できる
- ✓ データの「平均値」「最大値」「最小値」を計算できる

ステップ1 : Jupyter Notebook の起動

実習 3 システム統合

データ分析で最もよく使われるツール、
「Jupyter Notebook」を使います

1. **Anaconda Navigator**を起動します。
2. 「Jupyter Notebook」の**「Launch」**をクリックします
3. ブラウザが起動し、ファイル一覧が表示されます
4. 右上の「New」→**「Python 3」**で新しいノートブックを作成します

🎯 **基本操作**
「セル」と呼ばれる枠にコードを書き、**Shift + Enter**で実行します。
タスク： 最初のセルに `print("Hello, Python!")` と入力し、実行してみましょう

ステップ2：Pandas とデータ読み込み

実習 3 システム統合 Part 1

Python のデータ分析用ライブラリ「Pandas」を使います

Pandasとは？
Excelのような表形式データを扱うための強力なツールです

データの準備：実習 2 の Ambient/ThingSpeak から、
蓄積したデータをCSV形式でエクスポート（ダウンロード）します

```
# Pandasライブラリを「pd」という名前で読み込む
import pandas as pd
# CSVファイルを読み込む
# 'your_data.csv'はダウンロードしたファイル名に変更
df = pd.read_csv('ambient_data.csv') print("データ読み込み成功！")
```


ステップ3：データの確認

実習 3 システム統合 Part 1

読み込んだデータ（df）がどんなものか、中身を見てみよう

- `df.head()`：データの先頭 5 行を表示します
  → 列名（'created', 'd1', 'd2'など）とデータの中身を確認できます
- `df.info()`：データの詳細情報を表示します
  → 全体の行数、列ごとのデータ型、欠損値の有無がわかります
- `df.columns()`：すべての列名を一覧表示します

ステップ4：基本統計量の計算

実習 3 システム統合 Part 1

Pandas を使えば、大量のデータでも一瞬で統計量を計算できます

df.describe()：数値データの統計量をまとめて表示

統計量	意味
count	データの個数
mean	平均値
std	標準偏差（データのばらつき）
min / max	最小値 / 最大値
25% / 50% / 75%	四分位数（データを4分割した境界）

ワークブックに記録しよう

`df.describe()` の結果を見て、温度（d1）と湿度（d2）の「平均」「最小値」「最大値」をワークブックに記録してください

ステップ5：データの選択と抽出

実習 3 システム統合 Part 1

データ全体から、必要な部分だけを取り出す方法

▶ **特定の列だけ選択：**

- `df[['d1', 'd2']]`

▶ **条件でデータを抽出**（例：温度が25度以上）：

- `high_temp = df[df['d1'] >= 25]`

▶ **データを並び替え**（例：温度が高い順）：

- `df_sorted = df.sort_values('d1', ascending=False)`

ワークブックで実行しよう

ワークブックのタスクを実行し、「25度以上のデータ」が何行あったか、「最も暑かった時のデータ」は何かを確認してください

実習３ Part１のまとめ

実習３ システム統合 Part１

- ✓ Jupyter Notebook で Python コードを実行する方法を学んだ
- ✓ Pandasを使い、pd.read_csv() でデータを読み込んだ
- ✓ df.head(), df.info() でデータの概要を確認した
- ✓ df.describe() で平均・最大・最小などの統計量を計算した
- ✓ データの抽出 df[条件] とソート df.sort_values() を学んだ

NEXT

実習３ Part ２：IoTデータの高度な分析

実習３ システム統合

実習３ Part ２では、
このデータをグラフで**「可視化」**します
Matplotlib というライブラリを使い、
時系列グラフや相関図を作成して、
データに隠された**パターン**を発見します


NEXT
IoT技術応用
実習3 システム統合
Part 1 ：Pythonとデータ分析の基礎
Part 2 ：IoTデータの高度な分析
Part 3 ：Node-REDによるIoTフロー構築
Part 4 ：統合IoTシステムの開発
Mirai no tobira

ＩｏＴ技術応用_実習３

Part2


IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習1 デバイス制御
実習2 クラウド接続
実習3 システム統合
Mirai no tobira


IoT技術応用
実習3 システム統合
Part 1 ：Pythonとデータ分析の基礎
Part 2 ：IoTデータの高度な分析
Part 3 ：Node-REDによるIoTフロー構築
Part 4 ：統合IoTシステムの開発
Mirai no tobira

実習３ システム統合

IoT技術応用

実習１ デバイス制御 → 実習２ クラウド接続 → **実習３ システム統合**



Part１：Python とデータ分析の基礎

☞ **Part２：IoTデータの高度な分析**

Part３：Node-RED によるIoTフロー構築

Part４：統合IoTシステムの開発

Part２：IoTデータの高度な分析

実習３ システム統合

Matplotlib、統計分析、異常値検出、予測モデル

Part 2 ：IoTデータの高度な分析 の達成目標

実習 3 システム統合

実習 3 Part 1 の「データ読み込み」から進み、
データから「意味」を読み解きます

- ✓ Matplotlib でデータを「可視化」できる
- ✓ 温度と湿度の「関係性（相関）」を分析できる
- ✓ 統計的な手法で「異常値」を検出できる
- ✓ 機械学習の初歩である「線形回帰」で簡単な予測ができる

ステップ 1 ：Matplotlib による可視化

実習 3 システム統合

「Matplotlib」は、Pythonでグラフを
描画するための標準ライブラリです

```
# 必要なライブラリをインポート
import matplotlib.pyplot as plt
# Jupyter Notebook内にグラフを表示するおまじない
%matplotlib inline
# 日本語フォントの設定（推奨）
plt.rcParams['font.sans-serif'] = ['MS Gothic', 'Yu Gothic']
```

- ▶ `plt.plot()`：折れ線グラフ
- ▶ `plt.scatter()`：散布図
- ▶ `plt.hist()`：ヒストグラム
- ▶ `plt.title()`,`plt.xlabel()`,`plt.ylabel()`：タイトルやラベル

タスク１：時系列グラフと２軸グラフ

実習３ システム統合 Part２

時間の経過とともにデータがどう変化したかを見てみよう

- **時系列グラフ：**
  - 横軸を「日時」、縦軸を「温度」にしてプロットします
- **２軸グラフ：**
  - 単位が異なる「温度（℃）」と「湿度（%）」を同時に表示するテクニック
  - `ax1.twinx()` を使って右側にも縦軸を作ります

ワークブックで実行しよう
ワークブックのコードを実行し、グラフが表示されることを確認してください
グラフから何がわかるか考察してください

タスク２：詳細統計、相関分析

実習３ システム統合 Part２

もう少し詳しい統計を見ます

- **詳細統計**
  - 温度の平均、中央値、標準偏差などを個別に計算する
- **標準偏差**
  - データのバラつき具合を表す
  - 大きいと温度変化が激しい。小さいと安定している
- **相関分析**
  - 相関とは２つの変数の関係性
  - correlation = df[d1].corr(df[d2])

相関係数とは？
-1から1までの値で、2つのデータの関係性を示します
- **１に近い**　：正の相関（温度が上がると湿度も上がる）
- **−１に近い**：負の相関（温度が上がると湿度は下がる）
- **0に近い**　：相関なし

ステップ３：異常値検出（3σ法）

実習３ システム統合 Part２

データ分析では、ノイズやセンサーエラーなど
「異常な値」を見つけることが重要です

3σ法：
統計学的な手法で「平均値 ± （標準偏差 × 3）」の範囲から外れたデータを異常値とみなします

```
# 平均と標準偏差を計算
mean_temp = df_iot['d1'].mean()
std_temp = df_iot['d1'].std()
# 異常値の閾値（上限と下限）
upper_limit = mean_temp + 3 * std_temp
lower_limit = mean_temp - 3 * std_temp
# 閾値から外れたデータを抽出
outliers = df_iot[(df_iot['d1'] > upper_limit) | (df_iot['d1'] < lower_limit)]
```

ワークブックで実行しよう
コードを実行し、異常値が何件見つかったか確認してください

ステップ4：簡単な予測モデル

実習３ システム統合 Part２

機械学習ライブラリ「scikit-learn」を使い
「時刻から温度を予測する」モデルを作ります

線形回帰：
データに最もフィットする
直線を引く手法です

```
# 必要なライブラリをインポート
from sklearn.linear_model import LinearRegression
# 説明変数 X（時刻）と 目的変数 y（温度）を準備
X = df_iot['hour'].values.reshape(-1, 1)
y = df_iot['d1'].values
# モデルを作成し、学習させる
model = LinearRegression()
model.fit(X, y)
# 予測を実行
y_pred = model.predict(X)
```

ポイント
`model.fit(X, y)` が「学習」で、`model.predict(X)` が「予測」です。これは機械学習の基本パターンです

予測モデルの評価

実習 3 システム統合 Part 2

作成したモデルがどれくらい当たっているかを評価します

決定係数 (R^2)：
モデルの当てはまりの良さ
1 に近いほど良い予測

```
from sklearn.linear_model import LinearRegression
# 時刻から温度を予測
X = df['hour'].values.reshape(-1, 1)
y = df['d1'].values

model = LinearRegression()
model.fit(X, y)

print(f"決定係数 (R²): {model.score(X, y):.3f}")
```

考察

単純な直線では、実際の温度変化（曲線）をうまく予測できていないことがわかります
より高度な予測には、線形回帰以外のモデルが必要になります

実習 3 Part 2 のまとめ

実習 3 システム統合 Part 2

- ✓ Matplotlib で時系列グラフ、2軸グラフ、ヒストグラムなどを作成した
- ✓ 相関係数や時間帯別集計など、統計的な分析を行った
- ✓ 3σ法を使って、データから異常値を検出した
- ✓ scikit-learn を使い、線形回帰モデルで予測を体験した

NEXT

Part３：Node-REDによるIoTフロー構築

実習３ システム統合

実習３ Part３では、

分析（Python）から視点を変え、

リアルタイム処理を行います

Node-RED というビジュアルツールを使い

MQTTデータを受けて動く

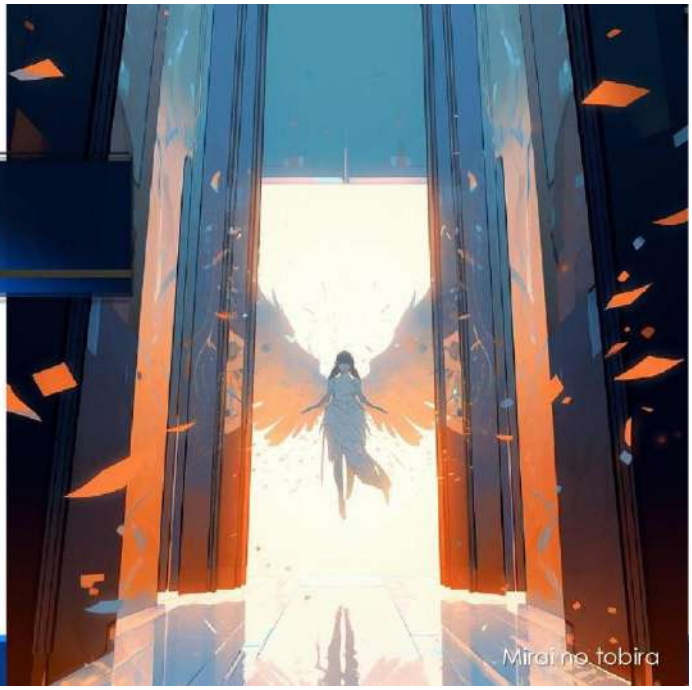
ダッシュボードを作ります

NEXT

IoT技術応用

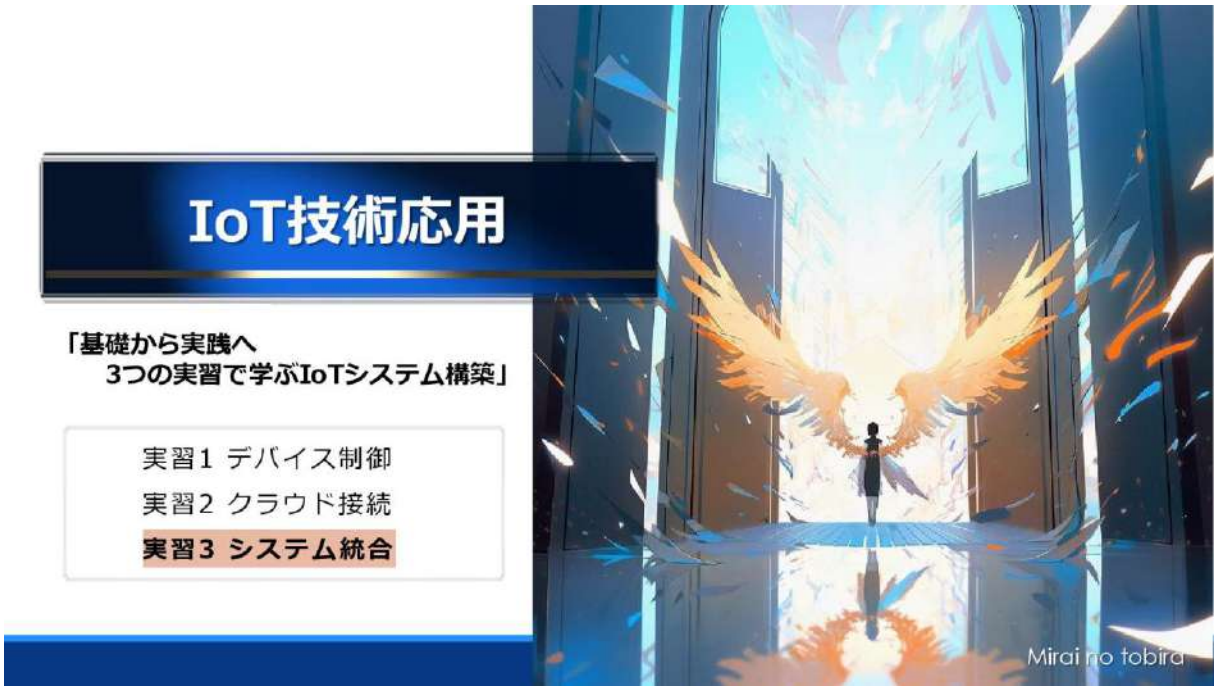
実習３ システム統合

Part１：Pythonとデータ分析の基礎
Part２：IoTデータの高度な分析
Part３：Node-REDによるIoTフロー構築
Part４：統合IoTシステムの開発



ＩｏＴ技術応用_実習３

Part3


IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習1 デバイス制御
実習2 クラウド接続
実習3 システム統合
Mirai no tobira


IoT技術応用
実習3 システム統合
Part1 ：Pythonとデータ分析の基礎
Part2 ：IoTデータの高度な分析
Part3 ：Node-REDによるIoTフロー構築
Part4 ：統合IoTシステムの開発
Mirai no tobira

実習３ システム統合

IoT技術応用

実習１ デバイス制御 → 実習２ クラウド接続 → **実習３ システム統合**



Part１：Python とデータ分析の基礎
Part２：IoTデータの高度な分析
☞ **Part３：Node-RED によるIoTフロー構築**
Part４：統合IoTシステムの開発

Part３：Node-RED によるIoTフロー構築

実習３ システム統合

ビジュアルプログラミングでIoTシステムを構築

Node-REDとは？

実習３ システム統合

ブラウザ上で「ノード」と呼ばれる機能ブロックを繋げて
プログラムを作成するツールです

- **ビジュアルプログラミング：**
  コードをあまり書かずに、処理の流れ（フロー）を作れる
- **IoTに最適：**
  MQTT、HTTP、センサー、データベースなど、IoT用のノードが豊富
- **リアルタイム処理：**
  データが流れてくると、即座に処理が実行される

ステップ１：最初のフロー

実習３ システム統合 Part３

基本操作を学びます

1. **起動：**ターミナルで node-red と入力し、ブラウザで http://localhost:1880 を開く
2. **配置：**左のパレットから「inject」と「debug」ノードをドラッグ
3. **接続：**injectの右側とdebugの左側を線でつなぐ
4. **実行：**右上の「デプロイ」ボタンを押し、injectの左のボタンを押す
5. **確認：**右側の「デバッグ」タブにタイムスタンプ（数値）が表示される

ポイント
`inject`がデータ（**msg.payload**）を送り出し、`debug`がそれを受け取って表示します

ステップ２：MQTTによるデータ受信

実習３ システム統合 Part３

MQTTは、IoTで最も標準的な通信プロトコル（通信ルール）の一つです

- **Pub/Sub モデル** ：「送信者」と「受信者」が直接通信しません
- **Broker（仲介人）**：すべてのメッセージを仲介します
- **Topic（宛先）** ：データの種類ごとに「トピック」（例：`iot/test/temperature`）を決めます

今回使うBroker
テスト用の公開Broker **test.mosquitto.org** を使います

MQTT受信フローの作成

実習３ システム統合 Part３

特定のトピック（宛先）に来たデータを受信してみよう

1. **ノード配置：**「mqtt in」ノードと「debug」ノードを配置し、接続します
2. **サーバー設定：**
   1. mqtt in をダブルクリック。「サーバー」の鉛筆アイコンを押します
   2. サーバー：test.mosquitto.org, ポート：1883 を入力
3. **トピック設定：**
   トピック：iot/test/temperature を入力
4. **デプロイ：**「デプロイ」ボタンを押します
   ノードの下に「接続済み」と表示されれば成功です

テスト
ワークブックの手順に従って、テスト用の「mqtt out」フローを作り、データを送信してデバッグタブで受信できるか確認しましょう

ステップ3：データ処理フロー

実習３ システム統合 Part３

受信したデータを「加工」したり「分岐」させたりします

- **functionノード：**
  - JavaScriptコードで自由にデータを加工できます

    例：摂氏を華氏に変換する、平均値を計算する
- **switchノード：**
  - データの値に応じて、処理を分岐させます

    例：「温度が25度以上なら」Aの処理、「25度未満なら」Bの処理

ステップ4：ダッシュボード作成

実習３ システム統合 Part３

受信したデータをリアルタイムでグラフ化します

1. ライブラリ導入：

右上のメニュー「パレットの管理」→「ノードを追加」タブで「node-red-dashboard」を検索し、インストールします

2. フロー作成：

1. パレットに「dashboard」グループが追加されます
2. mqtt in ノードに、「**gauge**」（メーター）、「**chart**」（グラフ）、「**text**」（数値）ノードを接続します

3. 確認：

デプロイ後、ブラウザで http://localhost:1880/ui にアクセスします

実習３ Part３のまとめ

実習３ システム統合 Part３

- ✓ Node-RED で、ノードを繋いでフローを作成する方法を学んだ
- ✓ MQTT プロトコルを使い、リアルタイムでデータを受信した
- ✓ function や switch ノードで、データを処理・分岐させた
- ✓ Node-RED Dashboard で、リアルタイムダッシュボードを作成した

NEXT

実習３ Part４ ：統合IoTシステムの開発

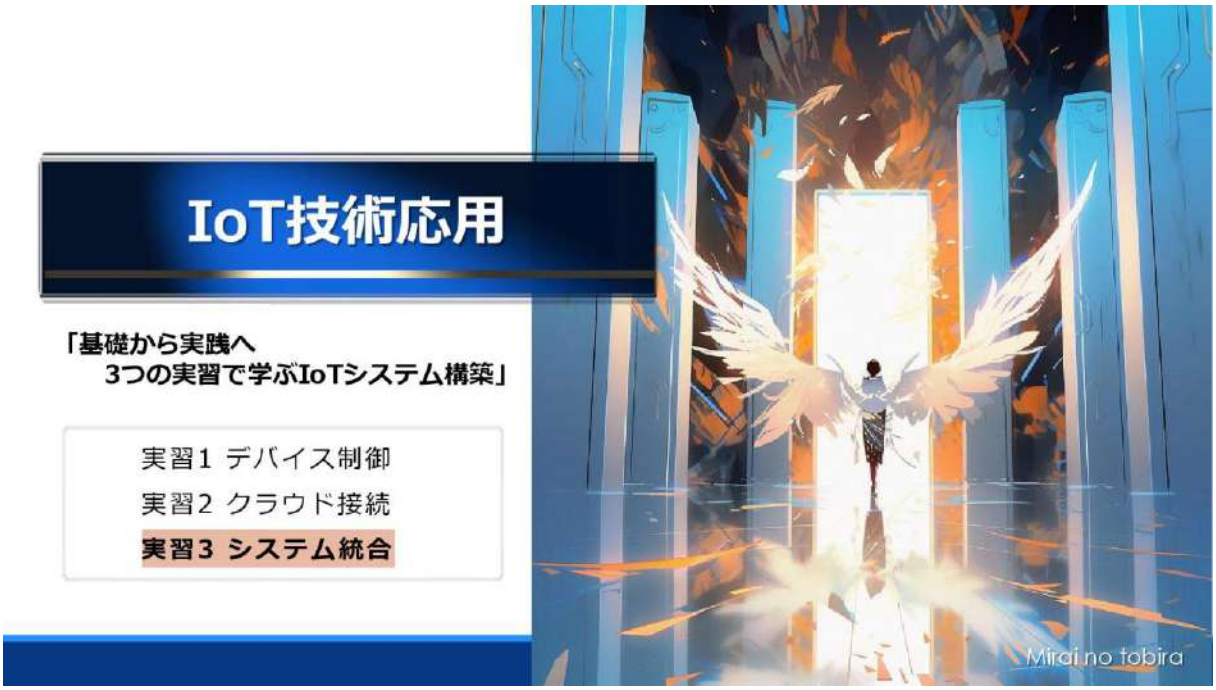
実習３ システム統合

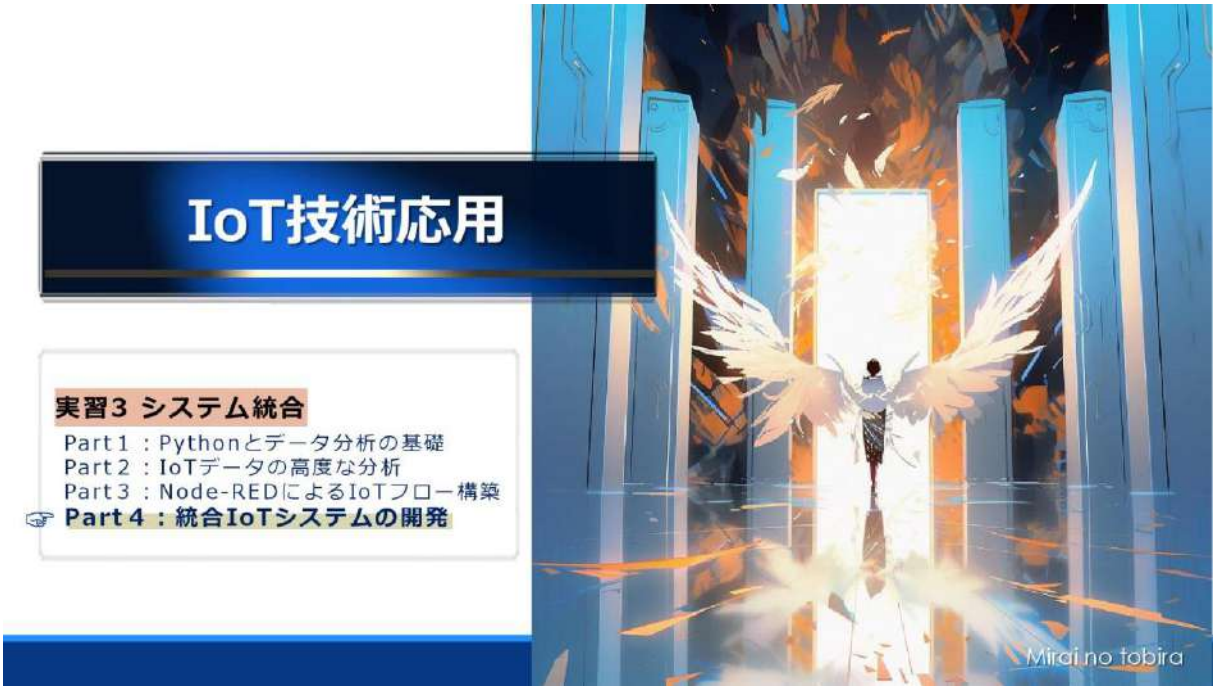
実習３ Part４は、IoT応用実習の最終回です
実習２の **ESP32**、**実習３ Part 1,2**の **Python**、
そして今回の **Node-RED をすべて統合**した
「完全なIoTシステム」を構築します


NEXT
IoT技術応用
実習3 システム統合
Part１：Pythonとデータ分析の基礎
Part２：IoTデータの高度な分析
Part３：Node-REDによるIoTフロー構築
Part４：統合IoTシステムの開発
Mirai no tobira

ＩｏＴ技術応用_実習３

Part4


IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習1 デバイス制御
実習2 クラウド接続
実習3 システム統合
Mirai no tobira


IoT技術応用
実習3 システム統合
Part1 ：Pythonとデータ分析の基礎
Part2 ：IoTデータの高度な分析
Part3 ：Node-REDによるIoTフロー構築
Part4 ：統合IoTシステムの開発
Mirai no tobira

実習３ システム統合

IoT技術応用

実習１ デバイス制御 → 実習２ クラウド接続 → **実習３ システム統合**



Part１：Pythonとデータ分析の基礎
Part２：IoTデータの高度な分析
Part３：Node-REDによるIoTフロー構築
☞**Part４：統合IoTシステムの開発**

Part４：統合IoTシステムの開発

実習３ システム統合

ESP32 + Python + Node-RED の完全統合

Part４のゴール（実習３の総仕上げ）

実習３ システム統合

これまで学んだ全ての技術を連携させ
実用的な IoTシステムを完成させます

- **[ハード]** ESP32 がデータを測定し、MQTT で送信する
- **[リアルタイム]** Node-RED がデータを受信し、ダッシュボードを即時更新する
- **[バッチ分析]** Python が蓄積データを分析し、分析結果を MQTT で送信する
- **[自動化]** Node-RED がリアルタイムデータを監視し、閾値を超えたら自動でアラートを出す

統合IoTシステムの全体像

実習３ システム統合 Part４

[ESP32] ー (MQTT) → [Node-RED] → [Dashboard]
↓ (HTTP)　　↑ (MQTT)
[Ambient] ←(Python)┘

- **リアルタイム系（MQTT）：**
  ESP32 → Node-RED → Dashboard
- **バッチ分析系（HTTP/MQTT）：**
  ESP32 → Ambient → Python → Node-RED

ステップ１：ESP32 (MQTT送信)

実習３ システム統合 Part４

実習２のコードを改造し、Ambient への送信と同時に、MQTT で Node-RED にも送信します

1. **ライブラリ導入：**
   Arduino IDEで「**PubSubClient**」をインストールします
2. **コード追加：**
   PubSubClient を設定し、loop() 内で mqttClient.publish() を呼び出します

```
#include <PubSubClient.h>
const char* mqtt_server = "test.mosquitto.org";
PubSubClient mqttClient(espClient);
void loop() { if (!mqttClient.connected()) reconnectMQTT();
 mqttClient.loop(); float temp = ...
 mqttClient.publish("iot/esp32/temperature", tempStr);
 ambient.set(1, temp); ambient.send(); delay(10000); }
```


ステップ２：Node-RED （リアルタイム受信）

実習３ システム統合 Part４

Part 3で作成したダッシュボードを、ESP32 から送られてくるデータで動かします

▶ **mqtt in ノードのトピックを変更：**

Part３のテスト用トピック（iot/test/temperature）から、ESP32が実際に送信するトピック（iot/esp32/temperature）に変更します

▶ **デプロイ：**

デプロイし直し、/ui ページを開きます

動作確認

ESP32が起動していると、10秒ごとにダッシュボードのゲージやグラフが自動で更新されるはずです

ステップ３：Python (分析結果送信)

実習３ システム統合 Part４

Part２で分析した結果（平均気温など）を
Python から Node-RED のダッシュボードに送信します

1. ライブラリ導入：

Jupyter Notebookで !pip install paho-mqtt を実行し、MQTTライブラリを入れます

2. 送信コード：

Part２の分析コードの最後にMQTTで分析結果（平均、最大、最小）を送信するコードを追加します

```
import paho.mqtt.client as mqtt import json
# (Part 2の分析... avg_temp, max_temp を計算)
client = mqtt.Client() client.connect("test.mosquitto.org", 1883)
result = { "average": avg_temp, "maximum": max_temp, ... }
message = json.dumps(result)
client.publish("iot/analysis/result", message)
client.disconnect()
```


ステップ４：Node-RED (分析結果受信)

実習３ システム統合 Part４

Python から送られてきた分析結果を
ダッシュボードに表示します

フローの作成：

- **mqtt in ノード：**
  トピック iot/analysis/result を購読
- **json ノード：**
  送られてきたJSON文字列を、扱いやすいオブジェクト形式に変換します
- **function ノード：**
  msg.payload.average や msg.payload.maximum を取り出します
- **text ノード：**
  分析結果（平均温度など）をダッシュボードに表示します

動作確認
Jupyter Notebookの分析コードを実行すると、ダッシュボードの「分析結果」欄が更新されることを確認します

ステップ５：自動アラートの実装

実習３ システム統合 Part４

リアルタイムデータ（ESP32から）を監視し
異常があれば通知します

フローの作成：
1. ESP32 の mqtt in (temperature) ノードから線を引き出す
2. **switch ノード**を接続
   - 条件：>（より大きい） 30（閾値）に設定
3. switch ノードの出力（30度を超えた場合）を「**notification**」ノードに接続
4. notification ノードを設定（例：「高温警告！温度が30度を超えました」）
5. デプロイ

動作確認
センサーを手で温め、温度を30度以上にしてみましょう。ダッシュボード（/ui）の右上に警告通知が表示されれば成功です

実習３のまとめ

実習３ システム統合 Part４

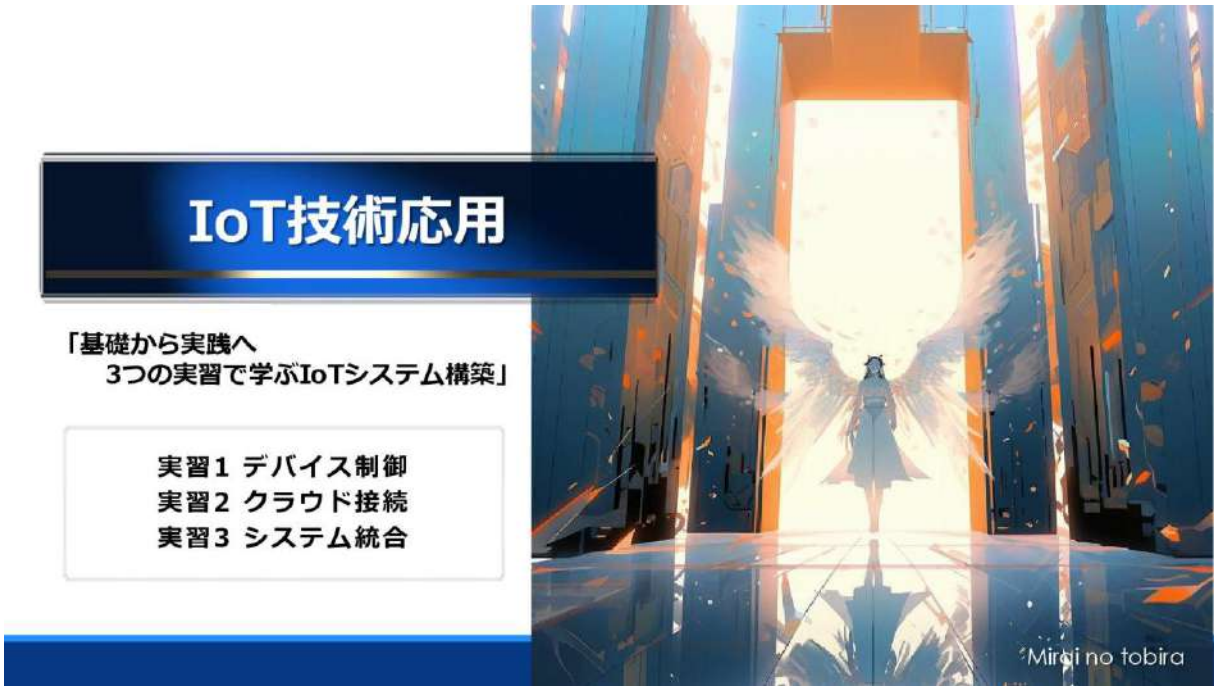
実習１～３の全ての技術を統合し、実用的なシステムを完成させました

- ▶ **実習１（デバイス）　：Arduino でハードウェア制御の基礎を学んだ**
- ▶ **実習２（クラウド）　：ESP32 でデータをクラウドに蓄積した**
- ▶ **実習３（分析・統合）：Python でデータを分析し、Node-RED でリアルタイム処理と可視化を行った**

これで、IoTシステムの「入力」「処理」「出力」「記録」「通信」「分析」「可視化」の全てを自分で実装できるスキルを身につけることができました

NEXT

発展課題
余力のある人は「発展課題ガイド」を参考に、自分だけの機能を追加してみよう


IoT技術応用
「基礎から実践へ
3つの実習で学ぶIoTシステム構築」
実習1 デバイス制御
実習2 クラウド接続
実習3 システム統合
Mirai no tobira

令和7年度「地方やデジタル分野における専修学校理系転換等推進事業」
情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

ＩｏＴ技術応用教材資料 2

令和８年２月
一般社団法人全国専門学校情報教育協会
〒164-0003 東京都中野区東中野 1-57-8 辻沢ビル３F
電話：03-5332-5081 FAX.03-5332-5083

●本書の内容を無断で転記、掲載することは禁じます。