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技術応用教材ワークブック

本ＩｏＴ技術応用教材ワークブックは、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 技術応用教材
実習 1 デバイス制御
実習ガイド

1. この実習について

実習の位置づけ

この実習は、『IoT 技術基礎』で学んだ理論を、実際に手を動かして確認するための**実践教材**です。
IoT 基礎教材は全 7 章、各章 28 項目で構成されており、IoT システムの理論や仕組みを体系的に学習できます。
この実習では、それらの理論を実際のハードウェアとプログラミングで体験し、**「理解」を「実践」に変える**ことを目指します。

『IoT 技術基礎』 （理論・知識の学習）
- 第 1 章：IoT システム構成
- 第 2 章：クラウドコンピューティング
- 第 3 章：エッジコンピューティング
- 第 4 章：IoT データ活用技術
- 第 5 章：IoT 通信方式
- 第 6 章：IoT デバイス
- 第 7 章：IoT システム開発

⬇ 実践で確認

『IoT 技術応用』実習 1
（実践・体験による理解の深化）
- Part1：LED 制御の基礎
- Part2：センサーでデータ取得
- Part3：条件による制御
- Part4：データ記録と表示

実習の目的

この実習を通じて、以下のことを目指します：

知識の確認

- 基礎教材で学んだ理論が、実際にどう動くのかを確認する

- IoT システムの基本構成を実物で理解する
- センサー、アクチュエータ、データ処理の関係を体感する

技能の習得

- 電子回路の基本的な配線ができるようになる
- 簡単なプログラムを書いて動かせるようになる
- トラブルを自分で解決する力を身につける

応用力の育成

- 学んだ知識を組み合わせて、新しいものを作れるようになる
- 実際の問題を IoT で解決する方法を考えられるようになる

実習で作るシステム

4 回の実習で、段階的に機能を追加しながら、**実用的な環境モニタリングシステム**を完成させます。

完成システムの機能

- ✓ 温度・湿度を自動測定
- ✓ 測定値に応じて LED で警告
- ✓ データを SD カードに自動記録
- ✓ LCD ディスプレイに測定値を表示
- ✓ パソコンで記録データを分析可能

このシステムは、農業、倉庫管理、研究室など、実際の現場で使われている IoT システムと同じ構造です。

2. 実習の全体構成

4 回の実習の流れ

実習は 4 回に分けて進めます。各回で新しい機能を追加し、最終的に完全なシステムを作り上げます。

Part1：LED 制御の基礎

テーマ：出力の基本を学ぶ

作るもの

LED を点灯・点滅させるプログラムを作ります。

学ぶこと

- Arduino の基本的な使い方
- 電子回路の基礎（LED、抵抗）
- プログラムの基本構造（setup、loop）
- デジタル出力（digitalWrite）

対応するオンライン教材

- **第 1 章：**1-4 IoT システムの基本要素
- **第 6 章：**6-1 IoT デバイスとは、6-9 アクチュエータの基礎知識

- **第 7 章：**7-12 プログラミングの基本概念、7-14 デバイス制御の基本

Part2：センサーでデータ取得

テーマ：入力の基本を学ぶ

作るもの

温度・湿度センサー（DHT11）で環境データを測定し、パソコンに表示します。

学ぶこと

- センサーの接続方法
- ライブラリの使用方法
- シリアル通信の基礎
- データの取得と表示

対応するオンライン教材

- **第 1 章：**1-5 センサーとは何か
- **第 4 章：**4-3 データ収集の基礎
- **第 6 章：**6-3 センサーの基礎知識、6-4 温度・湿度センサーの基礎
- **第 7 章：**7-13 センサーデータ取得の基礎

Part3：条件による制御

テーマ：入力と出力を統合する

作るもの

温度に応じて LED の点滅パターンを変えるシステムを作ります。

学ぶこと

- 条件分岐（if 文）
- センサー値による制御
- システム統合の基礎
- 閾値の設定と調整

対応するオンライン教材

- **第 1 章：**1-14 IoT システムの処理の種類
- **第 3 章：**3-7 エッジでのデータ処理とは、3-8 リアルタイム処理の基礎
- **第 7 章：**7-12 プログラミングの基本概念、7-14 デバイス制御の基本

Part4：データ記録と表示

テーマ：実用的なシステムに仕上げる

作るもの

SD カードへのデータ記録と LCD ディスプレイへの表示機能を追加します。

学ぶこと

- SD カードへのファイル書き込み
- LCD ディスプレイへの文字表示
- SPI 通信と I2C 通信
- 複数モジュールの統合

対応するオンライン教材

- **第 1 章：**1-15 IoT システムのデータ保存方式
- **第 4 章：**4-7 データの保存方式、4-12 データの可視化基礎
- **第 5 章：**5-1 IoT 通信の基本概念、5-2 通信プロトコルとは
- **第 6 章：**6-11 ディスプレイの基礎

段階的な学習の流れ

Part1：出力（LED 点灯）
↓
Part2：入力（センサー測定）
↓
Part3：統合（条件による制御）
↓
Part4：完成（記録・表示機能の追加）
↓
実用的な IoT システムの完成！

3. 実習の進め方

各回の基本的な流れ

各回の実習は、以下の流れで進みます：

ステップ	時間	内容
1. 導入	5-10 分	今回の目標、デモンストレーション
2. 説明	10-15 分	回路の説明、理論の確認
3. 配線	15-20 分	実際に回路を組み立てる
4. プログラミング	20-30 分	プログラムを入力・修正する
5. 動作確認	10-15 分	動かして確認、トラブル対応
6. 実験・考察	10-15 分	実験を行い、結果を記録
7. まとめ	5-10 分	振り返り、次回の予告

ワークブックの使い方

各回にワークブックを配布します。ワークブックには以下が含まれています：

- **目標：**その回で学ぶこと
- **手順：**配線やプログラムの説明
- **記録欄：**実験結果や気づきを書く場所
- **考察問題：**理解を深めるための質問
- **チェックリスト：**確認すべき項目

記入のポイント

- 配線やプログラムで気づいたことをメモする
- エラーが出たときの対処法を書いておく
- 実験結果を正確に記録する
- 考察では「なぜそうなるのか」を考える
- わからないことは質問欄に書く

実習での注意事項

安全について

- 配線するときは、必ず USB ケーブルを抜く
- 部品を無理に押し込まない
- 間違った配線は発熱や故障の原因になる
- わからないことは、必ず先生に聞く

部品の取り扱い

- 部品の向きに注意する（LED、センサーなど）
- 足を曲げすぎない
- 静電気に注意する
- 紛失しないように管理する

プログラミングについて

- 大文字・小文字を正確に入力する
- セミコロン（;）を忘れない
- 括弧の対応を確認する
- エラーメッセージをよく読む
- 分からないときは、周りの人と協力する

困ったときは

エラーが出たとき

1. **落ち着いて**エラーメッセージを読む

2. **ワークブックの**トラブルシューティングを見る
3. **配線を**もう一度確認する
4. **プログラムを**もう一度確認する
5. **それでも解決しないとき**は、先生に質問する

質問の仕方

良い質問の例：

- 「○○のエラーが出ましたが、どこを確認すればいいですか？」
- 「配線は確認しましたが、LED が点灯しません」
- 「プログラムのこの部分の意味がわかりません」

エラーは学習のチャンスです。自分で解決できたら、大きな成長になります。

4. 準備するもの

各回で使用する部品

実習で使用する部品の参考情報です。

Part	新しく使う部品	継続使用
Part1	• Arduino Uno • LED • 抵抗 220Ω • ブレッドボード • ジャンパーワイヤー	—
Part2	• DHT11 センサーモジュール （部品単体の場合は抵抗 10KΩ） • ジャンパーワイヤー追加	Part1 の全部品
Part3	（新規部品なし）	Part2 までの全部品
Part4	• SD カードモジュール • microSD カード（32GB 以下） • LCD ディスプレイ • ジャンパーワイヤー追加	Part3 までの全部品

各自で準備するもの

- ノートパソコン（Arduino IDE インストール済み）
- USB ケーブル（A-B タイプ）

- 筆記用具
- ワークブック（各回配布）

任意で持参すると便利なもの

- デジタルカメラ / スマートフォン（回路の写真を撮る）
- ノート（追加のメモ用）
- USB メモリ（プログラムを保存する）

Arduino IDE のインストール

Arduino IDE は、Arduino のプログラムを書くためのソフトウェアです。

インストール方法（事前準備）

1. Arduino 公式サイト（https://www.arduino.cc/）にアクセス
2. 「Products」→「Software」→「Arduino IDE」→「Downloads」を選択
3. 使用している OS（Windows、Mac、Linux）に合わせてダウンロード
4. ダウンロードしたファイルを実行してインストール
5. インストール完了後、Arduino IDE を起動して確認

💡 **ポイント：**初回実習の前に、インストールしておくことが望まれます。初回起動後に、「Windows セキュリティ」の警告やドライバのインストールを促す画面が出ますが、「インストール」をクリックして許可します。

ワークブックの提出

各回の実習終了時に、ワークブックを提出してください。

- 全ての記入欄を埋める
- 実験結果を正確に記録する
- 考察を自分の言葉で書く
- わからなかったことも正直に書く

6. 学習のコツ

実習を最大限に活かすために

1. 事前に基礎教材を復習する

各回の実習に対応する基礎教材の章を、事前に見直しておくと理解が深まります。

- 該当する章・節を読み直す
- 理解が曖昧な部分をメモしておく
- 実習でどう使われるか想像してみる

2. 「なぜ？」を大切にする

手順通りに進めるだけでなく、「なぜそうするのか」を考えましょう。

- 「なぜこの配線が必要なのか？」
- 「なぜこのプログラムで動くのか？」
- 「もし○○を変えたらどうなるのか？」

3. エラーを恐れない

エラーは学習のチャンスです。

- エラーメッセージを読んで、原因を考える
- 自分で試行錯誤してみる
- 解決できたら、その方法をメモする

4. 周りと協力する

一人で悩まず、周りの人と教え合いましょう。

- 困っている人がいたら手伝う
- わからないことは聞き合う
- 気づいたことを共有する

5. 実験を楽しむ

教材通りだけでなく、自分でも試してみましょう。

- 数値を変えて試してみる
- 「こうしたらどうなる？」を実験する
- 自分なりの工夫を加えてみる

復習のすすめ

実習後の復習

実習後に基礎教材を見直すと、理解が深まります。

1. **実習で学んだこと**をワークブックで確認
2. **対応する章・節**をオンライン教材で復習
3. **理論と実践**の関係を整理する
4. **応用できそうなこと**を考える

各回の復習ポイント

Part	復習すべき基礎教材
Part1	第 6 章（IoT デバイス）、第 7 章（プログラミング）
Part2	第 1 章（センサー）、第 4 章（データ収集）
Part3	第 3 章（エッジ処理）、第 7 章（システム開発）

Part4	第 4 章（データ保存）、第 5 章（通信方式）

7. よくある質問

実習について

Q1: プログラミングの経験がありませんが、大丈夫ですか？

A: 大丈夫です。実習では、基礎から丁寧に説明します。ワークブックに従って進めれば、プログラミング初心者でも必ず動かせます。むしろ、実際に動かしながら学ぶのが、プログラミングを理解する最良の方法です。

Q2: 電子工作の経験がありませんが、大丈夫ですか？

A: 大丈夫です。配線は、ワークブックの図を見ながら進めれば問題ありません。

Q3: 自分のペースで進められますか？

A: はい。早く終わった人は発展課題に取り組めます。逆に、時間がかかっても問題ありません。

Q4: 家でも同じことができますか？

A: はい。Arduino Uno と部品は、インターネットや電子部品店で購入できます（全部で 5,000 円程度）。ワークブックのプログラムを使って、個人でも実験できます。

学習について

Q7: 基礎教材を全部学習していないとダメですか？

A: 全部学習していなくても、実習は理解できるように設計されています。ただし、対応する章を事前に読んでおくと、より深く理解できます。

Q8: 実習で作ったシステムは何に使えますか？

A: 完成したシステムは、以下のような場面で実際に使えます：

- 部屋の温度管理
- 温室の環境モニタリング
- 倉庫の温湿度記録
- 研究室の実験データ収集

実習後の発展課題で、さらに機能を追加することもできます。

Q9: もっと高度なことを学びたいです

A: 4 回の実習が終わった後、発展課題が用意されています。そこには、Wi-Fi 通信、複数センサー、クラウド連携などの応用課題が紹介されています。

8. 実習スケジュール

標準的な実習スケジュール

実習は、通常以下のスケジュールで実施します：

回	テーマ	所要時間	提出物
Part1	LED 制御の基礎	90 分	ワークブック
Part2	センサーでデータ取得	90 分	ワークブック
Part3	条件による制御	90 分	ワークブック
Part4	データ記録と表示	90-120 分	ワークブック

💡 **ポイント：**所要時間はあくまでも想定です。

準備のスケジュール

実習開始前

- Arduino IDE をインストールする
- ノートパソコンの動作確認
- USB ケーブルの準備
- 基礎教材の該当章を確認

各実習の前日

- 前回のワークブックを復習
- 今回の実習に対応するオンライン教材を読む
- 筆記用具を準備
- 質問事項をメモしておく

実習当日

- ノートパソコンを持参
- Arduino IDE が起動することを確認
- 前向きな気持ちで臨む！

9. 参考情報

Arduino について

Arduino とは

Arduino は、イタリアで開発された教育用のマイコンボードです。世界中で使われており、初心者でも扱いやすいのが特徴です。

Arduino の特徴

- **簡単：**プログラミングが初心者でも使いやすい
- **安価：**本体は 3,000 円程度
- **豊富な資料：**インターネットに情報がたくさんある

- **拡張性：**様々なセンサーやモジュールが使える

Arduino でできること

- LED やモーターの制御
- センサーでの環境測定
- ロボットの制作
- IoT システムの構築
- アート作品の制作

参考リンク

公式サイト

- **Arduino 公式：**https://www.arduino.cc/
- **Arduino 日本語リファレンス：**http://www.musashinodenpa.com/arduino/ref/

学習サイト

- **Arduino Project Hub：**https://create.arduino.cc/projecthub
- **Instructables：**https://www.instructables.com/circuits/arduino/

コミュニティ

- **Arduino Forum：**https://forum.arduino.cc/
- **Stack Overflow：**https://stackoverflow.com/questions/tagged/arduino

💡 **ヒント：**困ったときは、英語で検索すると情報がたくさん見つかります。

10. 最後に

実習を通じて得られるもの

この実習を終えたとき、以下のことができるようになります：

技術的なスキル

- ✓ Arduino で電子回路を組める
- ✓ 基本的なプログラムを書ける
- ✓ センサーでデータを取得できる
- ✓ データを記録・表示できる
- ✓ IoT システムを作れる

考え方・姿勢

- ✓ 理論を実践で確認する習慣
- ✓ 試行錯誤しながら学ぶ力
- ✓ 問題を自分で解決する力
- ✓ 学んだことを応用する力
- ✓ 協力して学ぶ姿勢

将来への広がり

この実習で学ぶ技術は、以下の分野につながります：

- **ロボティクス：**ロボットの制御
- **スマート農業：**農業の自動化・最適化
- **スマートシティ：**都市のインフラ管理
- **ヘルスケア：**健康管理システム
- **環境保護：**環境モニタリング
- **製造業：**工場の自動化

実習に向けて

実習は、**「失敗を恐れず、楽しみながら学ぶ」**場です。

- エラーが出ても大丈夫。それが学びのチャンスです。
- わからないことは恥ずかしくありません。しっかり解決を目指してください。
- 完璧を目指さなくても大丈夫。試行錯誤が大切です。
- 周りの人と協力しながら、一緒に学びましょう。

最も大切なのは、「自分の手で動かして、体験すること」です。

基礎教材で学んだ理論が、実際にどう動くのか。自分の目で見て、手で触れて、確かめてください。

IoT 技術応用
実習 1 デバイス制御
Part 1 （LED 制御の基礎）ワークブック

実習記録

氏名	
実習日	年　　月　　日

目次

1. 学習目標

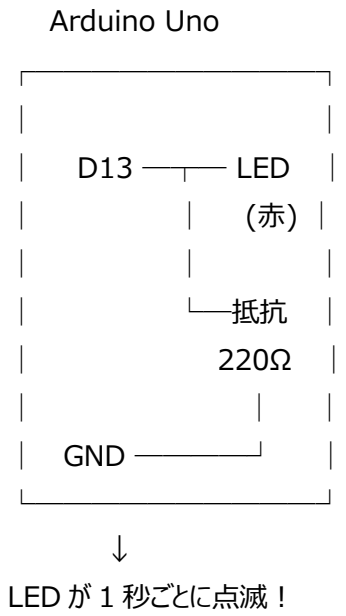
この Part で学ぶこと

- ✓ Arduino IDE のインストール
- ✓ Arduino と PC の接続方法
- ✓ ブレッドボードの使い方
- ✓ LED の配線方法
- ✓ Blink プログラムの理解
- ✓ プログラムの書き込み方法

できるようになること

- Arduino IDE を起動できる
- Arduino を PC に接続できる
- LED を正しく配線できる
- プログラムを Arduino に書き込める
- LED を点滅させられる
- delay の値を変更できる

完成イメージ



2. 準備するもの

必要な機材

- ☐ PC（Arduino IDE インストール用）
- ☐ Arduino Uno
- ☐ USB ケーブル（A-B タイプ）
- ☐ ブレッドボード
- ☐ LED（赤）1 個
- ☐ 抵抗 220Ω 1 個
- ☐ ジャンパーワイヤー 2 本

3. 作業手順チェックリスト

ステップ 1: Arduino IDE インストール

手順

1. Arduino 公式サイトにアクセス
   URL: https://www.arduino.cc/
2. Download ページへ移動
3. Windows 版をダウンロード
   ダウンロード開始時刻:

4. インストーラーを実行
   完了時刻:
5. Arduino IDE を起動
   起動確認: □ 成功 □ 失敗

トラブルメモ

問題が起きた場合は記録しておきましょう:
問題:

解決方法:

ステップ 2: Arduino 接続

手順

1. USB ケーブルを確認
   ケーブルタイプ: USB A-B
   (A=PC 側、B=Arduino 側)
2. Arduino に接続（B 端子側）
3. PC に接続（A 端子側）
4. 電源 LED （ボード上）点灯確認
   点灯した色:＿＿＿＿＿＿＿＿＿＿＿＿色
   （正解*:* 緑色）
5. デバイスマネージャーで確認
   Windows: スタート → デバイスマネージャー
   ポート(COM と LPT)を開く
   Arduino Uno が表示される
   COM ポート番号: COM＿＿＿

確認

- □ ボード上の緑色の LED が点灯している
- □ デバイスマネージャーに表示されている

トラブルメモ

問題:

解決方法:

ステップ 3: Arduino IDE 設定

手順

1. Arduino IDE を起動
2. ボード選択
   ツール → ボード → Arduino Uno
   確認: ☐ 完了
3. ポート選択
   ツール → シリアルポート → COM____
   (デバイスマネージャーで確認した番号)
   確認: ☐ 完了

確認

画面下部に以下が表示されているか:

「Arduino Uno on COM____」

- ☐ 表示されている
- ☐ 表示されていない

ステップ 4: LED 配線

⚠ **重要な注意**

配線作業の前に、必ず USB ケーブルを抜いてください！

☐ USB ケーブルを抜いた

LED の確認

1. LED を確認
   LED の足を比べてみよう
   長い足:_____mm（アノード、+側）
   短い足:_____mm（カソード、-側）

配線手順

- ☐ LED をブレッドボードに差す

配線メモ:

長い足 → 10a の穴に差す

短い足 → 12a の穴に差す

確認: ☐ 完了

- ☐ 抵抗 220Ω を接続

  配線メモ:

  一方の足 → 12c（LED の短い足と同じ行）

  他方の足 → 17c

  確認: ☐ 完了

- ☐ 赤いワイヤーで接続

  配線メモ:

  D13 ←→ 10e（LED の長い足）

  確認: ☐ 完了

- ☐ 黒いワイヤーで接続

  配線メモ:

  GND ←→ 17e（抵抗）

  確認: ☐ 完了

配線チェック

自分で確認:

- ☐ LED の向きは正しい（長い足が D13 側）
- ☐ 抵抗は接続されている
- ☐ D13 に赤ワイヤー接続
- ☐ GND に黒ワイヤー接続
- ☐ ワイヤーはしっかり差さっている

配線図

自分で描いてみよう:

ステップ 5: プログラム書き込み

手順

- ☐ USB ケーブルを接続
  (配線確認後に接続)
- ☐ Arduino IDE で Blink を開く
  ファイル → スケッチ例 → 01.Basics → Blink
- ☐ プログラムが開いた
  確認: ☐ 完了
- ☐ コンパイル（検証）
  ✓ ボタンをクリック
  結果: ☐ 成功 ☐ 失敗
  成功した場合: 「コンパイルが完了しました」と表示
- ☐ 書き込み
  → ボタンをクリック
  結果: ☐ 成功 ☐ 失敗
  成功した場合: 「マイコンボードへの書き込みが完了しました」

トラブルメモ

エラーメッセージ:

解決方法:

ステップ 6: 動作確認

確認

- ☐ LED が点滅している
  観察:
  点灯時間: 約 秒
  消灯時間: 約 秒

  正解: 約 *1* 秒ずつ
- ☐ 内蔵 LED（L）も点滅している
  (Arduino 基板上の小さな *LED)*

✓ 成功！

- ☐ 外付け LED が 1 秒ごとに点滅
- ☐ 内蔵 LED も同時に点滅

完了時刻:_____

4. プログラム記録シート

プログラムの構造を理解しよう

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

質問に答えよう

Q1: setup() に書いたことは何回実行される？

答え: _____回

Q2: loop() に書いたことは何回実行される？

答え: _____

Q3: pinMode(LED_BUILTIN, OUTPUT); の意味は？

答え:

Q4: digitalWrite(LED_BUILTIN, HIGH); の意味は？

HIGH にすると: _____
LOW にすると: _____

Q5: delay(1000); の意味は？

1000 の単位: _____
1000 は何秒？: _____ 秒

実験: delay の値を変えてみよう

実験 1: delay(500) に変更

プログラムを変更:

- ☐ delay(1000) を delay(500) に変更
- ☐ コンパイル
- ☐ 書き込み

結果:

なぜそうなった？

実験 2: delay(2000) に変更

結果:

実験 3: 自分で値を決める

delay(_____) に変更

予想:

結果:

気づいたこと・疑問

5. 確認問題

問題 1: LED の配線

LED の長い足はどちら側に接続しますか？

- ☐ A. 電源側（D13）

□ B. GND 側

答え: _____

理由を説明してください:

問題 2: 抵抗の役割

LED に抵抗を接続するのはなぜですか？

答え:

問題 3: setup と loop

setup() と loop() の違いを説明してください

setup():

loop():

問題 4: プログラムの理解

delay(1500) は何秒待ちますか？

計算:

答え: _____秒

問題 5: 応用問題

0.3 秒点灯、0.7 秒消灯を繰り返すには delay の値をいくつにすればよいですか？

計算:

0.3 秒 = _____ ミリ秒

0.7 秒 = _____ ミリ秒

プログラム:

```
digitalWrite(LED_BUILTIN, HIGH);
delay(________);  // 点灯時間
digitalWrite(LED_BUILTIN, LOW);
delay(________);  // 消灯時間
```

答え合わせ

自分で答え合わせをしよう:

正解数: ____/ 5 問

わからなかった問題:

問題__　__:

✅ 実習振り返りシート（IoT 技術応用 - 実習 1 Part 1）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習 1 デバイス制御
Part 2 （センサーデータの取得）ワークブック

実習記録

氏名	
実習日	年　　月　　日

目次

1. 学習目標

この Part で学ぶこと

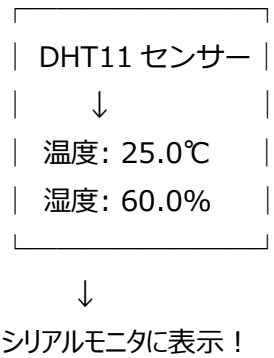
- DHT11 センサーの仕組み
- ライブラリのインストール方法
- センサーの配線方法
- 温湿度データの取得方法
- シリアルモニタの使い方
- nan の意味

できるようになること

- ライブラリをインストールできる
- DHT11 を正しく配線できる
- 温度と湿度を測定できる
- シリアルモニタでデータを確認できる
- センサーエラーを理解できる

完成イメージ

Arduino + DHT11



シリアルモニタに表示！

2. 準備するもの

必要な機材

- Part1 で作った回路（そのまま使用）
- DHT11 センサーモジュール 1 個
- ジャンパーワイヤー 3 本

💡 **ポイント**

Part1 で作った LED 回路はそのままで大丈夫です。DHT11 センサーを追加するだけです。

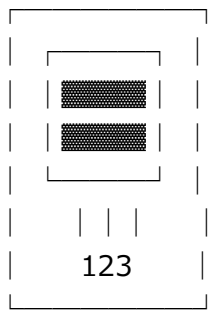
DHT11 センサーについて

DHT11 とは？

温度と湿度を同時に測定できるセンサーです。

- **測定範囲（温度）：** −20〜60℃（古いバージョンのものは 0〜50℃）
- **測定範囲（湿度）：** 5〜9 5％（古いものは 20〜90%）
- **精度（温度）：** ±2℃
- **精度（湿度）：** ±5%

DHT11 モジュールのピン配置



ピンの役割: **1 番ピン: VCC** → 5V（電源） **2 番ピン: DATA** → D2（データ通信） **3 番ピン: GND** → GND（グランド）

⚠ 注意

センサー部品単体の場合は、足は４本で、右から２番目が何もつながっていない NC ピンとなります。
モジュール型の場合は、基板に印刷されているピン名を確認してください。

3. 作業手順チェックリスト

ステップ 1: ライブラリインストール

ライブラリとは？

センサーを簡単に使うための「便利な道具セット」です。
DHT11 を使うには、専用のライブラリが必要です。

手順

1. Arduino IDE を起動
2. ライブラリマネージャーを開く
   スケッチ → ライブラリをインクルード → ライブラリを管理
   確認: ☐ 開けた
3. DHT sensor library を検索
   検索欄に入力: DHT
   表示された: ☐ はい ☐ いいえ
4. 「DHT sensor library」を選択
   作者: **Adafruit**
   確認: ☐ 見つけた
5. インストールをクリック
   完了時刻:
   結果: ☐ 成功 ☐ 失敗
6. 「Adafruit Unified Sensor」も同様に
   検索: Adafruit Unified Sensor
   インストール: ☐ 完了

✓ 確認

スケッチ → ライブラリをインクルード で
「DHT sensor library」が表示されますか？

- はい、表示される
- いいえ、表示されない

トラブルメモ

問題:

解決方法:

ステップ 2: DHT11 センサー確認

センサーの観察

1. DHT11 センサーモジュールを手に取る
2. ピンの位置を確認する

自分のセンサーのピン配置をメモしよう:

1 番ピン（左）:

2 番ピン（中央）:

3 番ピン（右）:

3. センサーの向きを確認
   格子模様が見える面が**正面**です

スケッチ（イラスト）

自分のセンサーを観察して描いてみよう

ステップ 3: センサー配線

⚠ **重要な注意**

配線作業の前に、必ず USB ケーブルを抜いてください！

- ☐ USB ケーブルを抜いた

配線手順

1. DHT11 をブレッドボードに差す

   配線メモ:

   1 番ピン（VCC） → **20a** の穴

   2 番ピン（DATA）→ **21a** の穴

   3 番ピン（GND） → **22a** の穴

   確認: ☐ 完了
2. VCC を 5V に接続

   配線メモ:

   20e ←赤ワイヤー→ Arduino **5V**

   確認: ☐ 完了
3. DATA を D2 に接続

   配線メモ:

   21e ←黄ワイヤー→ Arduino **D2**

   確認: ☐ 完了
4. GND を GND に接続

   配線メモ:

   22e ←黒ワイヤー→ Arduino **GND**

   確認: ☐ 完了

配線チェック

自分で確認:

- VCC は 5V に接続されている
- DATA は D2 に接続されている
- GND は GND に接続されている
- センサーの向きは正しい
- ワイヤーはしっかり差さっている

配線図

自分のシステム全体の配線図を描こう:

ステップ 4: プログラム作成

手順

1. USB ケーブルを接続
   (配線確認後に接続)
2. 新しいスケッチを開く
   ファイル → 新規ファイル
3. プログラムを入力
   次のページのプログラムを参照してください

プログラムコード

```
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(9600);
 dht.begin();
}

void loop() {
 float temp = dht.readTemperature();
 float humi = dht.readHumidity();
 Serial.print("温度: ");
 Serial.print(temp);
 Serial.println("℃");
 Serial.print("湿度: ");
 Serial.print(humi);
 Serial.println("%");
 delay(2000);
}
```

💡 入力のコツ

- 大文字・小文字に注意
- セミコロン（;）を忘れずに
- 括弧の対応を確認

プログラム入力チェックリスト

- #include <DHT.h> を入力

- #define DHTPIN 2 を入力
- #define DHTTYPE DHT11 を入力
- DHT dht(DHTPIN, DHTTYPE); を入力
- setup() を入力
- loop() を入力

コンパイルと書き込み

1. コンパイル（検証）
   結果: □ 成功 □ 失敗

エラーがある場合:

2. 書き込み
   結果: □ 成功 □ 失敗

ステップ 5: 動作確認

手順

1. シリアルモニタを開く
   ツール → シリアルモニタ
   または Ctrl+Shift+M
   確認: □ 開けた
2. ボーレート設定
   シリアルモニタ右下: **9600 baud**
   確認: □ 設定した
3. 温度・湿度が表示される
   記録:
   温度: ℃
   湿度: %
4. 値が妥当か確認
   室温として妥当: □ はい □ いいえ
   （目安: 温度 *16-30*℃、湿度 *30-70%*）

✓ 成功！

- シリアルモニタに温度・湿度が表示される
- 値が妥当（室温として合理的）
- 2 秒ごとに更新される

完了時刻:

トラブルシューティング

⚠ 「nan」と表示される場合

nan = Not a Number（数値ではない）
センサーエラーが起きています。

確認項目:

- センサーの配線を確認
  VCC は 5V ？
  DATA は D2 ？
  GND は GND ？
- センサーを差し直す
- 数秒待ってから確認
  （初回は *nan* が出ることがある）
- それでもダメなら教員に相談

トラブルメモ

問題:

解決方法:

4. プログラム記録シート

プログラムの理解

各行の意味を自分の言葉で書こう:

#include <DHT.h>
意味:

#define DHTPIN 2
意味:

DHT dht(DHTPIN, DHTTYPE);
意味:

Serial.begin(9600);
意味:

dht.begin();
意味:

float temp = dht.readTemperature();

意味:

Serial.print("温度: ");

意味:

Serial.println(temp);

意味:

Serial.print との違いは？

delay(2000);

意味:

観察と実験

実験 1: 今の室温を測定

測定結果:

温度: ℃

湿度: %

この値は妥当ですか？

- はい
- いいえ

実験 2: センサーに息を吹きかける

吹きかける前:

温度: ℃

湿度: %

吹きかけた後:

温度: ℃

湿度: %

どう変化しましたか？

なぜそう変化したと思いますか？

実験 3: センサーを手で温める

センサーを手のひらで包んで温めてみよう。

温める前:

温度:　　℃

温めた後:

温度:　　℃

何度上がりましたか？

　　℃上昇

気づいたこと・疑問

5. 確認問題

問題 1: ライブラリの役割

ライブラリとは何ですか？
なぜ必要ですか？
答え:

問題 2: DHT11 の配線

DHT11 モジュールの 3 つのピンはそれぞれどこに接続しますか？
VCC（1 番ピン）:
DATA（2 番ピン）:
GND（3 番ピン）:

問題 3: シリアルモニタ

Serial.begin(9600); の
9600 は何を表していますか？
答え:

シリアルモニタで同じ値に設定しないとどうなりますか？

答え:

問題 4: nan の意味

シリアルモニタに「nan」と表示されることがあります。
これは何を意味しますか？
答え:

どんな時に出ますか？
答え:

問題 5: float 型

float temp = dht.readTemperature();
の「**float**」は何を表していますか？
答え:

なぜ float 型を使うのですか？
（int 型ではダメな理由）
答え:

答え合わせ

正解数:　　/ 5 問
わからなかった問題はしっかり解決しよう！

☑ 実習振り返りシート（IoT 技術応用 - 実習 1 Part 2）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習 1 デバイス制御
Part 3（条件による制御）ワークブック

実習記録

氏名	
実習日	年　　月　　日

目次

学習目標

- if 文を使って条件分岐ができる
- 比較演算子（>=、<、==など）を理解する
- センサーの値に応じて LED を制御できる
- 入力と出力を組み合わせたシステムを作れる

完成するシステム

温度によって LED の動作が変わるシステムを作ります。

温度例：

- **25 度未満：**LED は消灯
- **25 度以上 28 度未満：**LED がゆっくり点滅（1 秒間隔）
- **28 度以上：**LED が速く点滅（0.2 秒間隔）

準備するもの

Part1 と Part2 で使用した回路をそのまま使います。

- Arduino Uno（Part1 の回路そのまま）
- DHT11 センサーモジュール（Part2 の回路そのまま）
- USB-A to B ケーブル
- PC（Arduino IDE インストール済み）

ステップ 1：if 文の基本を理解しよう

if 文とは？

if 文は「もし○○なら、△△する」という条件分岐を行うプログラムです。

基本の形：

```
if (条件) {
    実行する処理
}
```

日本語で書くと：

```
もし（条件が正しければ）{
    この中の処理を実行する
}
```

比較演算子

演算子	意味	例
==	等しい	temp == 25 温度が 25 度ちょうど
!=	等しくない	temp != 25 温度が 25 度ではない
>	より大きい	temp > 25 温度が 25 度より大きい
>=	以上	temp >= 25 温度が 25 度以上
<	より小さい	temp < 25 温度が 25 度より小さい
<=	以下	temp <= 25 温度が 25 度以下

⚠ **注意！**

== と = は違います！

- ==（イコール 2 つ）：比較「等しいかどうか」
- =（イコール 1 つ）：代入「値を入れる」

if 文の中では必ず == を使います！

練習問題

温度が **30 度**のとき、次の条件は正しい（○）か間違い（×）か答えなさい。

条件	○ or ×	理由
temp >= 25		
temp < 25		
temp == 30		
temp > 30		

ステップ 2：else if と else を理解しよう

複数の条件を判定する

1 つの条件だけでなく、複数の条件を順番に判定したい場合があります。

else if と **else** を使います：

```
if (条件 1) {
    処理 1
} else if (条件 2) {
    処理 2
} else {
    処理 3（それ以外）
}
```

動作の流れ：

1. まず条件 1 をチェック → 正しければ処理 1 を実行して終了
2. 条件 1 が間違っていたら条件 2 をチェック → 正しければ処理 2 を実行して終了
3. 条件 1 も条件 2 も間違っていたら処理 3 を実行

今回のプログラムでの使い方

```
if (temp >= 28) {
    // 温度が 28 度以上
    速く点滅
} else if (temp >= 25) {
    // 温度が 25 度以上 28 度未満
    ゆっくり点滅
} else {
```

```
    // 温度が 25 度未満
    消灯
}
```

ポイント：

- 上から順番にチェックされる
- 最初に当てはまった条件だけが実行される
- 厳しい条件（大きい数）を先に書く

理解度チェック

温度が **26 度**のとき、上のプログラムではどの処理が実行されますか？

答え：

（　　）が実行される

理由：

ステップ 3：プログラムを作成しよう

準備

- Arduino IDE を起動する
- Part2 のプログラムを開く（または新規作成）
- USB ケーブルで Arduino を接続する

プログラム全文

以下のプログラムを入力してください。

```
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11
#define LEDPIN 13

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(LEDPIN, OUTPUT);
}
```

```
void loop() {
  float temp = dht.readTemperature();
  float humi = dht.readHumidity();

  Serial.print("温度: ");
  Serial.print(temp);
  Serial.println("℃");

  Serial.print("湿度: ");
  Serial.print(humi);
  Serial.println("%");

  if (temp >= 28) {
    Serial.println("状態: 高温・速く点滅");
    digitalWrite(LEDPIN, HIGH);
    delay(200);
    digitalWrite(LEDPIN, LOW);
    delay(200);
  } else if (temp >= 25) {
    Serial.println("状態: やや高温・ゆっくり点滅");
    digitalWrite(LEDPIN, HIGH);
    delay(1000);
    digitalWrite(LEDPIN, LOW);
    delay(1000);
  } else {
    Serial.println("状態: 通常・消灯");
    digitalWrite(LEDPIN, LOW);
  }

  delay(2000);
}
```

プログラムの説明

プログラムの各部分を理解しましょう。下の空欄を埋めなさい。

コード	説明
#define LEDPIN 13	LED が ピンに繋がっていることを定義
pinMode(LEDPIN, OUTPUT)	D13 を モードに設定
if (temp >= 28)	温度が 以上なら
else if (temp >= 25)	温度が 以上 未満なら
else	温度が 未満なら
delay(200)	ミリ秒待つ

入力チェックリスト

プログラムを入力したら、以下をチェックしてください：

- 大文字・小文字は正確に入力したか
- セミコロン（;）は全て付いているか
- 括弧の対応は正しいか（{ と } の数が合っているか）
- 比較演算子は正しいか（>= など）
- インデント（字下げ）はできているか

ステップ 4：動作確認をしよう

プログラムの書き込み

1. チェックマーク（✓）をクリックして、プログラムを検証する
2. エラーがなければ、矢印（→）をクリックして書き込む
3. 「マイコンボードへの書き込みが完了しました」と表示されたら成功

⚠ エラーが出た場合

よくあるエラー：

- **セミコロン忘れ：**各命令の最後に ； が必要
- **括弧の対応ミス：**{ の数と } の数が同じか確認
- **スペルミス：**digitalWrite、Serial.print など正確に

シリアルモニタで確認

1. 「ツール」→「シリアルモニタ」を開く
2. 右下が「9600 baud」になっているか確認
3. 温度、湿度、状態が表示されているか確認

現在の表示内容を記録：

温度：　　℃
湿度：　　%
状態：

LED の状態：

ステップ 5：実験しよう

実験 1：手で温める

手順：

1. センサーを手で包んで温める
2. シリアルモニタと LED を観察する
3. 温度の変化と LED の動作を記録する

観察結果：

温度	LED の状態	シリアルモニタの表示
最初　（　）℃		
25℃超えた　（　）℃		
28℃超えた　（　）℃		

実験 2：冷ます

手順：

1. 手を離して、センサーを冷ます
2. 温度が下がる様子を観察する
3. LED の動作が変わるタイミングを記録する

観察結果：

28℃を下回ったとき：

25℃を下回ったとき：

実験 3：閾値（しきいち）の変更

閾値とは：動作が変わる境界の温度のこと（今は 28 度と 25 度）

実験内容：

現在の室温に応じて、変化の範囲を考えて変更し、動作を確認しましょう。

変更箇所：（値は例）

```
if (temp >= 30) {   // 28 → 30 に変更
      …
} else if (temp >= 23) {   // 25 → 23 に変更
      …
}
```

考察

問 1：if 文の順序について

もし、if 文の順序を逆にしたらどうなりますか？

```
if (temp >= 25) {
      ゆっくり点滅
} else if (temp >= 28) {
      速く点滅
}
```

予想：

温度が 30℃のとき、どちらの処理が実行されますか？

答え：

理由：

問 2：応用例を考えよう

今回学んだ if 文を使って、他にどんなシステムが作れますか？
センサーと動作の組み合わせを考えてください。
例：明るさセンサー ＋ 照明 → 暗くなったら照明をつける
アイデア 1：
センサー：
動作：

アイデア 2：
センサー：
動作：

問 3：複数条件の判定

温度**と**湿度の両方を条件にするには、どう書けばよいですか？
例：「温度が 28 度以上で、かつ、湿度が 70%以上のとき」
予想：

```
if (                                    ) {
    // 処理
}
```

ヒント：&& を使います

今回学んだこと

チェックリスト

今回の実習で学んだことをチェックしましょう：

- if 文の基本的な書き方を理解した

- 比較演算子（>=、<、==など）を使える
- else if と else の使い方を理解した
- 条件の順序が重要であることを理解した
- センサーの値に応じて LED を制御できた
- 閾値を変更して動作を調整できた
- 入力と出力を組み合わせたシステムを作れた

重要なポイント

1. if 文の構造

```
if (条件 1) {
    処理 1
} else if (条件 2) {
    処理 2
} else {
    処理 3
}
```

2. 比較演算子の使い分け

- ==：等しいかどうか
- >=：以上（その値も含む）
- >：より大きい（その値は含まない）

3. 条件の順序

厳しい条件（大きい数）を先に書く！

4. IoT の基本構造

入力（センサー）→ 判断（if 文）→ 出力（LED）

次回の予告

Part4 では：

- 複数のセンサーを組み合わせる
- データを SD カードに保存する
- LCD ディスプレイに表示する
- より実用的な IoT システムを作る

準備：今回作った回路はそのままにしておいてください。

トラブルシューティング

問題が起きたときは

問題 1：LED が期待通りに動作しない

確認すること：

- 現在の温度を確認（シリアルモニタ）
- if 文の条件を確認（>= の向きなど）
- LED の配線を確認（D13 に接続されているか）
- LEDPIN の定義を確認（13 になっているか）

問題 2：コンパイルエラーが出る

よくあるエラー：

- セミコロン（;）忘れ → 各行の最後に ; を付ける
- 括弧の対応ミス → { と } の数を確認
- スペルミス → digitalWrite、Serial など正確に
- else if を elseif と書いている → 間にスペースが必要

問題 3：シリアルモニタに何も表示されない

確認すること：

- ボーレートが 9600 になっているか
- 正しい COM ポートに接続されているか
- Serial.begin(9600) が setup() にあるか

よくある間違い

間違い	正しい書き方
if temp >= 25	if (temp >= 25)

	条件は括弧で囲む
if (temp = 25)	if (temp == 25) 比較は == を使う
elseif (temp >= 25)	else if (temp >= 25) else と if の間にスペース

if (temp >= 25)
digitalWrite(13, HIGH);
	if (temp >= 25) {
	digitalWrite(13, HIGH);
	}
	中括弧で囲む

発展課題（時間がある人向け）

チャレンジ 1：4 段階に分ける

3 段階ではなく、4 段階に分けてみましょう。

条件：

- 30 度以上：非常に速く点滅（100ms）
- 28 度以上 30 度未満：速く点滅（200ms）
- 25 度以上 28 度未満：ゆっくり点滅（1000ms）
- 25 度未満：消灯

プログラムを書いてみよう：

```
if (                        ) {
    // 処理
} else if (                        ) {
    // 処理
} else if (                        ) {
    // 処理
} else {
    // 処理
}
```

チャレンジ 2：湿度も使う

温度と湿度の両方を条件に使ってみましょう。

例：不快指数による制御

- 温度 28 度以上 かつ 湿度 70%以上：非常に速く点滅
- 温度 28 度以上：速く点滅
- 温度 25 度以上：ゆっくり点滅
- それ以外：消灯

ヒント： && を使います

例：if (temp >= 28 && humi >= 70) { ... }

チャレンジ 3：不快指数を計算

温度と湿度から不快指数を計算して、それに応じて LED を制御してみましょう。

不快指数の計算式：

float discomfort = 0.81 * temp + 0.01 * humi * (0.99 * temp - 14.3) + 46.3;

不快指数の目安：

- 85 以上：暑くてたまらない
- 80-85：暑くて汗が出る
- 75-80：やや暑い
- 70-75：暑くない
- 65-70：快い
- 60-65：何も感じない
- 55-60：肌寒い
- 55 未満：寒い

次回に向けて

復習しておくこと：

- if 文の書き方
- 比較演算子の種類と意味
- 条件分岐の考え方

予習（余裕があれば）：

- for 文（繰り返し）について調べる
- 配列について調べる
- SD カードについて調べる

参考資料

プログラムの構造まとめ

```
// ライブラリの読み込み
#include <DHT.h>

// 定数の定義
#define DHTPIN 2
#define DHTTYPE DHT11
#define LEDPIN 13

// センサーの初期化
DHT dht(DHTPIN, DHTTYPE);

// 最初に 1 回だけ実行される
void setup() {
  Serial.begin(9600);      // シリアル通信開始
  dht.begin();             // センサー初期化
  pinMode(LEDPIN, OUTPUT); // ピン設定
}

// 繰り返し実行される
void loop() {
  // センサーデータ取得
  float temp = dht.readTemperature();
  float humi = dht.readHumidity();
```

```
  // データ表示
  Serial.print("温度: ");
  Serial.println(temp);

  // 条件による制御
  if (条件 1) {
    // 処理 1
  } else if (条件 2) {
    // 処理 2
  } else {
    // 処理 3
  }

  delay(2000); // 2 秒待つ
}
```

☑ 実習振り返りシート（IoT 技術応用 - 実習 1 Part 3）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習 1 デバイス制御
Part 4（データ記録と表示）ワークブック

実習記録

氏名	
実習日	年　　月　　日

今回の目標

学習目標

- SD カードにデータを記録できる
- LCD ディスプレイに情報を表示できる
- 複数のモジュールを組み合わせたシステムを作れる
- データの活用方法を理解する

完成するシステム

温度・湿度を測定し、以下の機能を持つシステムを作ります：

- **LCD ディスプレイ：**温度・湿度をリアルタイム表示
- **SD カード：**測定データを自動記録
- **LED：**温度に応じて点滅（Part3 から）
- **シリアルモニタ：**動作状況を表示

準備するもの

今回追加する部品

- SD カードモジュール
- microSD カード（32GB 以下、FAT32 フォーマット済み）
- LCD ディスプレイ（16×2、I2C 接続）
- ジャンパーワイヤー（オス-オス）×10 本

Part3 から継続して使用

- Arduino Uno（DHT11 センサー）
- USB-A to B ケーブル
- PC（Arduino IDE インストール済み）

💡 **ポイント：Part3 の回路の LED と抵抗、それらの配線は取り除いてください。**DHT11 はそのまま使います。

ステップ 1：SD カードモジュールの理解

SD カードモジュールとは？

SD カードモジュールは、Arduino で一般的な SD カードを読み書きできるようにする部品です。

使用目的

- センサーデータの長期保存
- パソコンなしでのデータ記録
- 後からデータを取り出して分析

ピンの説明

ピン名	意味	接続先
GND	グランド（マイナス）	Arduino GND
VCC	電源（プラス）	Arduino 5V
MISO	データ受信	Arduino D12（固定）
MOSI	データ送信	Arduino D11（固定）
SCK	クロック信号	Arduino D13（固定）
CS	チップセレクト	Arduino D10（変更可）

SD カードの準備

使用できる SD カード

- 容量：32GB 以下
- フォーマット：FAT32
- 種類：マイクロ SD、マイクロ SDHC（SDXC は不可）

SD カードの差し込み方

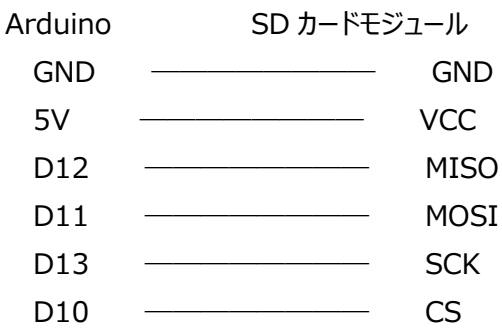
1. 金色の端子がある面を下にする
2. モジュールのスロットに差し込む
3. 「カチッ」と音がするまで押し込む
4. しっかり差し込まれているか確認

確認：SD カードを差し込みましたか？ □

ステップ 2：SD カードモジュールの配線

配線図

SD カードモジュール配線図



注意：D13 が LED とつながっている場合、LED 配線を取り除いてください。

配線手順

⚠️ **配線前の確認：**USB ケーブルを抜いて電源を切る

ステップ 1：GND を接続

- SD カードモジュールの GND → Arduino GND（黒いワイヤー推奨）

ステップ 2：VCC を接続

- SD カードモジュールの VCC → Arduino 5V（赤いワイヤー推奨）

ステップ 3：SPI ピンを接続

- SD カードモジュールの MISO → Arduino D12
- SD カードモジュールの MOSI → Arduino D11
- SD カードモジュールの SCK → Arduino D13

ステップ 4：CS を接続

- SD カードモジュールの CS → Arduino D10

配線チェックリスト

以下の項目を確認してください：

- GND は正しく接続されているか
- VCC は 5V に接続されているか（3.3V ではない）
- MISO、MOSI、SCK は正しいピンに接続されているか
- CS は D10 に接続されているか
- SD カードはしっかり差し込まれているか
- Part3 の回路（DHT11）はそのままか

気づいた点や修正した箇所：

ステップ 3：SD カード書き込みプログラム

プログラムの概要

Part3 のプログラムに、以下の機能を追加します：

- SD カードの初期化
- data.txt ファイルへのデータ書き込み
- 書き込み成功・失敗の表示

プログラム全文

以下のプログラムを入力してください：

```
#include <DHT.h>
#include <SD.h>
#include <SPI.h>

#define DHTPIN 2
#define DHTTYPE DHT11
#define LEDPIN 3
#define CSPIN 10

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(LEDPIN, OUTPUT);

  Serial.println("SD カード初期化中...");
  if (!SD.begin(CSPIN)) {
    Serial.println("SD カード初期化失敗");
    while (1);
  }
  Serial.println("SD カード初期化成功");
```

```
}

void loop() {
  float temp = dht.readTemperature();
  float humi = dht.readHumidity();

  Serial.print("温度: ");
  Serial.print(temp);
  Serial.println("℃");
  Serial.print("湿度: ");
  Serial.print(humi);
  Serial.println("%");

  // SDカードに書き込み
  File dataFile = SD.open("data.txt", FILE_WRITE);
  if (dataFile) {
    dataFile.print("温度:");
    dataFile.print(temp);
    dataFile.print(",湿度:");
    dataFile.println(humi);
    dataFile.close();
    Serial.println("データ保存成功");
  } else {
    Serial.println("ファイルオープン失敗");
  }

  // Part3のif文（温度によるLED制御）
  if (temp >= 28) {
    digitalWrite(LEDPIN, HIGH);
    delay(200);
    digitalWrite(LEDPIN, LOW);
    delay(200);
  } else if (temp >= 25) {
    digitalWrite(LEDPIN, HIGH);
    delay(1000);
    digitalWrite(LEDPIN, LOW);
    delay(1000);
  } else {
    digitalWrite(LEDPIN, LOW);
  }
```

```
  delay(2000);
}
```

プログラムの説明

新しく追加した部分を理解しましょう：

1. ライブラリの追加

#include <SD.h>
#include <SPI.h>
SD カードを使うためのライブラリです。

2. CS ピンの定義

#define CSPIN 10
SD カードの CS ピンが D10 に接続されていることを定義します。

3. SD カード初期化

if (!SD.begin(CSPIN)) {
  Serial.println("SD カード初期化失敗");
  while (1); // ここで停止
}
SD カードを初期化します。失敗したらプログラムを停止します。

4. ファイルへの書き込み

```
File dataFile = SD.open("data.txt", FILE_WRITE);
if (dataFile) {
  dataFile.print("温度:");
  dataFile.print(temp);
  dataFile.print(",湿度:");
  dataFile.println(humi);
  dataFile.close();
}
```

data.txt ファイルを開いて、温度と湿度を書き込み、ファイルを閉じます。

入力チェックリスト

- 大文字・小文字を正確に入力したか
- セミコロン（;）は全て付いているか
- 括弧の対応は正しいか
- #include の < > は半角か
- コンパイルエラーがないか確認したか

ステップ 4：SD カード書き込み動作確認

プログラムの書き込み

1. チェックマーク（✓）をクリックして検証
2. エラーがなければ、矢印（→）をクリックして書き込み
3. 「マイコンボードへの書き込みが完了しました」を確認
4. シリアルモニタを開く

シリアルモニタの確認

表示される内容：

SD カード初期化中...
SD カード初期化成功
温度: 23.5℃
湿度: 45.2%
データ保存成功
温度: 23.6℃
湿度: 45.3%
データ保存成功
...

確認項目：

- 「SD カード初期化成功」と表示されたか
- 温度と湿度が表示されているか
- 「データ保存成功」と表示されているか

SD カード内のファイル確認

手順

1. USB ケーブルを抜いて電源を切る
2. SD カードを取り出す（爪で押すと出てくる）
3. SD カードを PC で、中を確認する。
4. 「data.txt」ファイルを探す
5. data.txt をメモ帳などのテキストエディタで開く

ファイルの内容

以下のようなデータが記録されているはずです：
温度:23.5,湿度:45.2

温度:23.6,湿度:45.3
温度:23.5,湿度:45.1
…

動作確認記録

SD カード初期化の結果：

□ 成功 □ 失敗

data.txt に記録されたデータの例（最初の 3 行）：

ステップ 5：LCD ディスプレイの理解と配線

LCD ディスプレイとは？

LCD ディスプレイは、温度や湿度などの情報を文字で表示する画面です。

仕様

- 表示：16 文字×2 行　（今回利用するもの）
- 通信方式：I2C　　（ピン数が少なくて済む仕様）
  バックライトや、輝度調整などは、パーツによって異なります。また後ほどインストールするライブラリが異なる場合もあります。製品仕様はや使用方法はデータシートで確認してください。

ピンの説明

ピン名	意味	接続先
GND	グランド	Arduino GND
VCC	電源（5V）	Arduino 5V
SDA	データ線	Arduino A4（固定）
SCL	クロック線	Arduino A5（固定）

💡 **ポイント：**I2C は 4 本のピンだけで通信できるので、配線が簡単です。

LCD 配線図

```
Arduino                LCD ディスプレイ
  GND  ──────────────────  GND
  5V   ──────────────────  VCC
  A4   ──────────────────  SDA
  A5   ──────────────────  SCL
```

⚠️ **注意：**A4 と A5 は「アナログピン」です。D4 や D5 ではありません！

LCD 配線手順

⚠️ **配線前の確認：**USB ケーブルを抜いて電源を切る

- LCD GND → Arduino GND（黒いワイヤー）
- LCD VCC → Arduino 5V（赤いワイヤー）
- LCD SDA → Arduino A4
- LCD SCL → Arduino A5

配線完了チェック：

- SDA が A4 に接続されているか（D4 ではない）
- SCL が A5 に接続されているか（D5 ではない）
- VCC が 5V に接続されているか
- 全ての配線が抜けていないか

ステップ 6：LCD 表示プログラム

ライブラリのインストール

手順

1. Arduino IDE で「スケッチ」→「ライブラリをインクルード」→「ライブラリを管理」
2. 検索欄に「LiquidCrystal I2C」と入力
3. 「LiquidCrystal I2C by Frank de Brabander」を選択
4. 「インストール」をクリック
5. インストール完了後、ウィンドウを閉じる

確認：ライブラリをインストールしましたか？ □

プログラムへの追加

先ほどのプログラムに、LCD 表示機能を追加します。

追加する内容

1. ライブラリの追加（#include <SPI.h> の下に追加）

#include <LiquidCrystal_I2C.h>
※製品によって異なる場合あります。

2. LCD オブジェクトの作成（DHT dht の下に追加）

LiquidCrystal_I2C lcd(0x27, 16, 2);
説明：0x27 は LCD のアドレス、16 は文字数、2 は行数
※LCD のアドレスは製品によって異なる場合があります。

3. setup 関数に LCD 初期化を追加

Serial.println("SD カード初期化成功"); の下に追加：

```
lcd.init();
lcd.backlight();
lcd.print("Starting...");
```

4. loop 関数に LCD 表示を追加

Serial.println("%"); の下に追加：

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: ");
```

```
lcd.print(temp);
lcd.print("C");
lcd.setCursor(0, 1);
lcd.print("Humi: ");
lcd.print(humi);
lcd.print("%");
```

完成プログラム全文

```
#include <DHT.h>
#include <SD.h>
#include <SPI.h>
#include <LiquidCrystal_I2C.h>

#define DHTPIN 2
#define DHTTYPE DHT11
#define LEDPIN 3
#define CSPIN 10

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(LEDPIN, OUTPUT);

  Serial.println("SDカード初期化中...");
  if (!SD.begin(CSPIN)) {
    Serial.println("SDカード初期化失敗");
    while (1);
  }
  Serial.println("SDカード初期化成功");

  lcd.init();
  lcd.backlight();
  lcd.print("Starting...");
}

void loop() {
  float temp = dht.readTemperature();
```

```
float humi = dht.readHumidity();

Serial.print("温度: ");
Serial.print(temp);
Serial.println("℃");
Serial.print("湿度: ");
Serial.print(humi);
Serial.println("%");

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: ");
lcd.print(temp);
lcd.print("C");
lcd.setCursor(0, 1);
lcd.print("Humi: ");
lcd.print(humi);
lcd.print("%");

File dataFile = SD.open("data.txt", FILE_WRITE);
if (dataFile) {
  dataFile.print("温度:");
  dataFile.print(temp);
  dataFile.print(",湿度:");
  dataFile.println(humi);
  dataFile.close();
  Serial.println("データ保存成功");
} else {
  Serial.println("ファイルオープン失敗");
}

if (temp >= 28) {
  digitalWrite(LEDPIN, HIGH);
  delay(200);
  digitalWrite(LEDPIN, LOW);
  delay(200);
} else if (temp >= 25) {
  digitalWrite(LEDPIN, HIGH);
  delay(1000);
  digitalWrite(LEDPIN, LOW);
```

```
    delay(1000);
  } else {
    digitalWrite(LEDPIN, LOW);
  }

  delay(2000);
}
```

LCD コマンドの説明

コマンド	意味
lcd.init();	LCD を初期化する
lcd.backlight();	バックライトを点灯する
lcd.clear();	画面をクリアする
lcd.setCursor(列, 行);	カーソル位置を設定する 列：0〜15、行：0〜1
lcd.print("文字");	文字を表示する

表示位置の例

行 0: [0][1][2][3][4][5][6][7][8][9][10][11][12][13][14][15]
行 1: [0][1][2][3][4][5][6][7][8][9][10][11][12][13][14][15]

例：lcd.setCursor(0, 0);　// 1 行目の左端
　　lcd.print("Temp: 23.5C");

　　lcd.setCursor(0, 1);　// 2 行目の左端
　　lcd.print("Humi: 45.2%");

ステップ 7：最終動作確認

プログラムの書き込み

1. プログラムを検証（✓）してエラーがないか確認
2. プログラムを書き込み（→）
3. 書き込み完了を確認

動作確認チェックリスト

LCD ディスプレイの確認

- LCD のバックライトが点灯しているか
- 最初に「Starting...」と表示されたか
- 1 行目に「Temp: ○○C」と表示されているか

- 2 行目に「Humi: ○○%」と表示されているか
- 2 秒ごとに数字が更新されるか

シリアルモニタの確認

- SD カード初期化成功と表示されているか
- 温度・湿度が表示されているか
- 「データ保存成功」と表示されているか

SD カードの確認

- データが data.txt に記録されているか

動作記録

LCD に表示された内容：

1 行目：
2 行目：

現在の温度と湿度：

温度：　　℃
湿度：　　%

LED の動作：

□ 消灯　□ ゆっくり点滅　□ 速く点滅

実験：システムの総合動作確認

実験 1：温度変化の観察

目的

温度の変化に応じて、すべての出力が連動して変わることを確認する

手順

1. 現在の状態を記録する
2. DHT11 センサーを手で包んで温める
3. LCD の温度表示を観察する
4. LED の動作の変化を観察する
5. シリアルモニタの表示を確認する

実験記録表

時間	温度（℃）	湿度（%）	LED の状態
0 秒（初期）			
30 秒後			
60 秒後			
手を離して 30 秒後			

実験 2：データの長期記録

目的

SD カードにデータが継続的に記録されることを確認する

手順

1. 3 分間、システムを動作させ続ける
2. 電源を切って SD カードを取り出す
3. PC で data.txt を開く
4. 記録された行数を数える

実験結果

動作時間：　　分

記録された行数：　　行

計算上の行数：2 秒ごとに 1 行なので、3 分＝180 秒÷2＝90 行が期待値

実験結果は期待値と一致しましたか？

□ はい　□ いいえ

一致しなかった場合、その理由：

考察

考察 1：システムの統合

質問：Part1 から Part4 まで、どのように機能が追加されてきましたか？各回で学んだことを書いてください。

Part1：

Part2：

Part3：

Part4：

考察 2：データの活用

質問：SD カードに記録したデータは、どのように活用できますか？具体的な例を 3 つ書いてください。

活用例 1：

活用例 2：

活用例 3：

考察 3：実用化への課題

質問：今回作ったシステムを実際の現場（温室、倉庫、研究室など）で使うには、どんな改良が必要だと思いますか？

改良点 1：

改良点 2：

改良点 3：

統合化

LED 配線の復活

一旦、取り除いた LED と抵抗を、あらためて配線してください。
とりつけ位置や、配線方法を考えてください。
Arduino Uno の D13 に関しては、SD カードモジュールの SCK で利用していますので、使えません。
今回は、D3 を使います。あわせて、プログラムを変更します。

```
#define LEDPIN 3
```

この LED ランプが、高温アラートなどとして使うためには、どのようにプログラムすればよいでしょうか？
適切な if 文を考えてください。

設定と考察

自分で設定した内容と考察を書いてください。

まとめ

今回の成果

今回の実習で、以下の機能を持つ完全な IoT システムを作りました：

- ✓ センサーでデータを測定（温度・湿度）
- ✓ データを SD カードに記録
- ✓ LCD ディスプレイに表示
- ✓ シリアル通信で状態を報告
- ✓ 条件に応じて動作を変える（LED 制御）

これは、本格的な IoT システムの基本構造です！

学習のチェックリスト

今回学んだことをチェックしましょう：

- ☐ SD カードモジュールの配線ができた
- ☐ SD カードへのデータ書き込みができた
- ☐ LCD ディスプレイの配線ができた
- ☐ LCD への文字表示ができた
- ☐ 複数のモジュールを統合できた
- ☐ SPI と I2C の違いを理解した
- ☐ データの活用方法を理解した

重要なポイント

1. モジュールの統合

複数のモジュール（センサー、SD、LCD、LED）を 1 つのシステムに統合できました。

2. 通信方式の使い分け

- **SPI：**SD カード（高速、ピン数多い）
- **I2C：**LCD（低速で OK、ピン数少ない）

3. データの永続化

SD カードに保存することで、電源を切ってもデータが残ります。

4. ユーザーインターフェース

LCD があれば、パソコンなしでも情報を確認できます。

発展課題（余力がある人向け）

チャレンジ 1：タイムスタンプの追加

データに記録時刻を追加してみましょう。

必要なもの

- RTC モジュール（DS3231 など）

実装のヒント

millis()関数を使って、起動からの経過時間を記録する方法もあります。

```
unsigned long elapsed = millis() / 1000;  // 秒に変換
dataFile.print(elapsed);
dataFile.print(",温度:");
dataFile.print(temp);
```

チャレンジ 2：データのグラフ化

SD カードの data.txt を Excel で開いて、グラフを作成してみましょう。

手順

1. data.txt を開く
2. 「データ」→「区切り文字」でカンマ区切りを設定
3. データを選択して「挿入」→「グラフ」
4. 折れ線グラフを選択

分析のヒント

- 温度の最高値・最低値を見つける
- 平均値を計算する
- 温度と湿度の関係を見る

チャレンジ 3：LCD 表示のカスタマイズ

LCD 表示を工夫してみましょう。

アイデア例

- 温度が高いときに「WARNING!」と表示
- 快適度を「Comfortable」「Hot」などで表示
- 2 行をスクロールさせて情報を切り替える

実装例

```
if (temp >= 28) {
  lcd.print("*** HOT! ***");
} else if (temp >= 25) {
  lcd.print("** WARM **");
} else {
  lcd.print("Comfortable");
}
```

チャレンジ４：アラーム機能の追加

温度が閾値を超えたら、ブザーを鳴らす機能を追加してみましょう。

必要なもの

- 圧電ブザー
- ジャンパーワイヤー×2 本

配線

ブザーの+を D8、-を GND に接続

プログラム例

```
#define BUZZERPIN 8

void setup() {
  pinMode(BUZZERPIN, OUTPUT);
}

void loop() {
  if (temp >= 30) {
    tone(BUZZERPIN, 1000);  // 1000Hz の音を鳴らす
    delay(100);
    noTone(BUZZERPIN);
  }
}
```

トラブルシューティング

SD カード関連の問題

問題 1：SD カード初期化失敗

確認すること：

- SD カードがしっかり差し込まれているか
- 配線が正しいか（特に CS ピン）
- SD カードが 32GB 以下か
- SD カードが FAT32 でフォーマットされているか
- 別の SD カードで試してみる

問題 2：ファイルオープン失敗

原因：

- SD カードの容量が不足している
- ファイルが破損している
- 書き込み保護がかかっている

対策：SD カードをフォーマットし直す

問題 3：data.txt がない

確認すること：

- 「データ保存成功」と表示されているか
- CSPIN の設定が正しいか（10 になっているか）
- SD.begin(CSPIN)が成功しているか

LCD 関連の問題

問題 4：LCD に何も表示されない

確認すること：

- 配線が正しいか（SDA→A4、SCL→A5）
- VCC が 5V に接続されているか
- LiquidCrystal_I2C ライブラリがインストールされているか
- 製品に対応したライブラリか

問題 5：文字が表示されるが読めない

原因：コントラストの調整が必要
対策：LCD の裏側にある青い可変抵抗を小さなドライバーで回す

問題 6：文字化けする

原因：LCD のアドレスが違う
対策：LiquidCrystal_I2C lcd(0x27, 16, 2); を
例えば LiquidCrystal_I2C lcd(0x50, 16, 2); に変更

その他の問題

問題 7：コンパイルエラー

よくあるエラー：

- 'LiquidCrystal_I2C' does not name a type
  → ライブラリがインストールされていない
- expected ';' before ...
  → セミコロン忘れ
- expected '}' at end of input

→ 括弧の閉じ忘れ

問題８：動作が遅い

原因：lcd.clear()が重い処理
対策：毎回 clear せず、必要な部分だけ上書きする

```
// clear を使わない方法
lcd.setCursor(6, 0);
lcd.print("      ");  // スペースで消す
lcd.setCursor(6, 0);
lcd.print(temp);
```

自己評価

理解度チェック

各項目について、自分の理解度を記入してください。
（よく理解できた：◎、だいたい理解できた：○、あまり理解できなかった：△）

項目	理解度
SD カードモジュールの配線	
SD カードへのデータ書き込み	
LCD ディスプレイの配線	
LCD への文字表示	
SPI と I2C の違い	
複数モジュールの統合	
データの活用方法	

今後の学習

さらに学びたい人へ：

- Arduino 公式サイトで様々なプロジェクトを見てみる
- 他のセンサー（気圧、CO2、照度など）を試す
- ESP32 を使って Wi- 承知いたしました。続けます。 ```html Fi 通信に挑戦する
- 3D プリンターでケースを作る
- 太陽電池やバッテリーで動かす

参考リンク：

- Arduino 公式サイト：https://www.arduino.cc/
- Arduino 日本語リファレンス：http://www.musashinodenpa.com/arduino/ref/
- プロジェクト集：https://create.arduino.cc/projecthub

参考資料

完成システムの全体構成

```
┌──────────────────────┐
│    Arduino Uno       │
│                      │
│  D2 ← DHT11          │
│  D10 ← SD(CS)        │
│  D11 ← SD(MOSI)      │
│  D12 ← SD(MISO)      │
│  D13 ← SD(SCK)       │
│  D13 → LED           │
│  A4 ← LCD(SDA)       │
│  A5 ← LCD(SCL)       │
└──────────────────────┘
```

使用したライブラリー覧

ライブラリ名	用途	インストール方法
DHT.h	温湿度センサー	標準ライブラリ
SD.h	SD カード	標準ライブラリ
SPI.h	SPI 通信	標準ライブラリ
LiquidCrystal_I2C.h	LCD ディスプレイ	ライブラリマネージャーから

主要コマンドまとめ

SD カード関連

```
SD.begin(CSPIN);                    // 初期化
File file = SD.open("name.txt", FILE_WRITE);   // ファイルを開く
file.println("text");               // 書き込み
file.close();                       // ファイルを閉じる
```

LCD 関連

```
lcd.init();                         // 初期化
lcd.backlight();                    // バックライト ON
lcd.clear();                        // 画面クリア
lcd.setCursor(列, 行);              // カーソル位置設定
lcd.print("text");                  // 文字表示
```

DHT11 関連

```
dht.begin();                        // 初期化
float temp = dht.readTemperature();   // 温度取得
float humi = dht.readHumidity();      // 湿度取得
```

ピン使用状況

ピン	用途	備考
D2	DHT11 センサー	データピン
D3	LED	※変更先
D10	SD カード CS	変更可能
D11	SD カード MOSI	固定（SPI）
D12	SD カード MISO	固定（SPI）
D13	SD カード SCK	固定（SPI）
A4	LCD SDA	固定（I2C）
A5	LCD SCL	固定（I2C）

SPI と I2C の比較

項目	SPI	I2C
配線本数	4 本以上 (MISO, MOSI, SCK, CS)	2 本 (SDA, SCL)
速度	高速 (最大数十 MHz)	低速 (最大 400kHz)
複数デバイス	CS ごとに 1 ピン必要	同じピンで複数接続可
使用例	SD カード、センサー	LCD、RTC、センサー
特徴	高速だがピン多い	低速だがピン少ない

データファイルの例

data.txt の内容例：

温度:23.5,湿度:45.2
温度:23.6,湿度:45.3
温度:23.5,湿度:45.1
温度:23.7,湿度:45.4
温度:24.0,湿度:46.0
温度:24.5,湿度:47.2
温度:25.1,湿度:48.5
…

Excel での開き方：

1. Excel を起動
2. 「ファイル」→「開く」→data.txt を選択
3. 「区切り文字」を選択
4. 区切り文字で「カンマ」をチェック
5. 「完了」をクリック

グラフの作り方：

1. 温度の列を選択
2. 「挿入」→「グラフ」→「折れ線グラフ」
3. グラフタイトルや軸ラベルを設定

実用例の紹介

1. 農業用温室管理システム

- 24 時間 365 日の温湿度記録

- LCD で現在値を確認
- 異常時にアラーム
- データをグラフ化して作物の成長と関連付け

2. 食品倉庫の温度管理

- 法律で記録が義務付けられている
- 温度が基準を超えたら警告
- 1 年分のデータを保存
- 監査時にデータを提出

3. 博物館の環境モニタリング

- 美術品・文化財の保護
- 温湿度を厳密に管理
- 複数の部屋を同時監視
- 異常時に即座に対応

4. 研究室の実験記録

- 実験中の環境を自動記録
- 手書きより正確
- 後から分析可能
- 論文やレポートに使用

4 回の実習の総まとめ

学んだ技術の体系

回	テーマ	学んだこと	使った部品
Part1	出力制御	LED 点灯・点滅 digitalWrite, delay	LED、抵抗
Part2	センサー入力	温湿度測定 ライブラリ使用 シリアル通信	DHT11 センサー
Part3	条件分岐	if 文、比較演算子 センサー値による制御	Part1＋Part2
Part4	データ記録・表示	SD カード、LCD SPI、I2C 通信 システム統合	SD カード、LCD

習得したスキル

プログラミングスキル

- 変数の定義と使用
- 関数の理解（setup, loop）
- 条件分岐（if, else if, else）
- ライブラリの使用
- シリアル通信

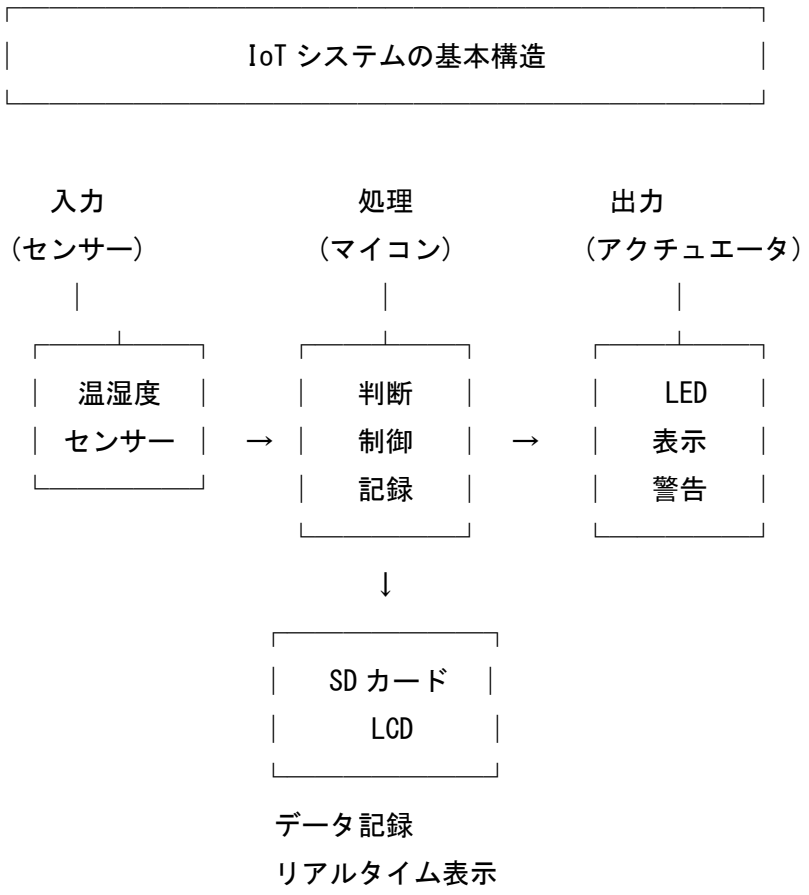
ハードウェアスキル

- ブレッドボードの使い方
- 配線の方法
- センサーの接続
- 複数モジュールの統合

問題解決スキル

- エラーの読み方
- デバッグの方法
- トラブルシューティング
- 動作確認の手順

IoT システムの基本構造



今後の発展方向

レベル 1：センサーを増やす

- 気圧センサー（BMP280）
- 照度センサー（TSL2561）
- CO2 センサー（MH-Z19）

レベル 2：通信機能を追加

- Wi-Fi（ESP32、ESP8266）
- Bluetooth（HC-05）
- LoRa（長距離無線）

レベル 3：クラウド連携

- データをクラウドに送信
- スマホから確認
- 遠隔地からアクセス
- 複数地点のデータ統合

レベル 4：AI 活用

- 機械学習で異常検知
- 予測モデルの構築
- 最適化アルゴリズム

最後のメッセージ

4 回の実習、お疲れ様でした！

IoT 技術の基礎を身につけました。これは、これからの時代に必要不可欠な技術です。

大切なのは、実際に手を動かしたことです。

プログラムを書いて、回路を作って、動かして、試行錯誤して…この経験が、一番の財産です。

エラーが出ても、諦めずに直しましたね。分からないことがあっても、調べたり、聞いたりして解決しましたね。それが、エンジニアリングです。

これからも、興味を持ち続けてください。

Arduino は、安くて手に入りやすいです。自分で買って、いろいろ試してみてください。

インターネットには、たくさんの情報があります。「Arduino プロジェクト」で検索すると、面白いアイデアがたくさん見つかります。

皆さんのアイデアで、新しいものを作ってください。

IoT 技術は、様々な問題を解決する力を持っています。環境問題、エネルギー問題、農業、医療…あらゆる分野で活用できます。

今回学んだことは、その第一歩です。

これからも、楽しみながら学び続けてください！

✅ 実習振り返りシート（IoT 技術応用 - 実習 1 Part 4）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習 1 デバイス制御
発展課題

🎉 4 回の実習が完了しました！

Part1 から Part4 まで、お疲れ様でした。あなたは今、以下のことができるようになりました：

- Arduino で電子回路を組める
- センサーでデータを取得できる
- 条件によって動作を変えられる
- データを SD カードに記録できる
- LCD に情報を表示できる
- **実用的な IoT システムを作れる**

これは素晴らしい成果です！

実習で学んだこと

技術的なスキル

Part	学んだこと	基礎教材との対応
Part1	デジタル出力、LED 制御	第 6 章、第 7 章
Part2	センサー入力、シリアル通信	第 1 章、第 4 章、第 6 章
Part3	条件分岐、システム統合	第 1 章、第 3 章、第 7 章
Part4	データ記録、LCD 表示、SPI/I2C	第 4 章、第 5 章、第 6 章

理論と実践の結びつき

基礎教材で学んだ理論を、実際のハードウェアで確認できました：

- **IoT システムの 4 層構造** → 実際に作って体験
- **センサーの種類と原理** → DHT11 で温湿度測定
- **エッジ処理** → Arduino でリアルタイム制御
- **データ保存と通信** → SD カードと LCD で実現

ここからが本当のスタート

実習で学んだ技術は、**基礎中の基礎**です。

ここから、あなたのアイデア次第で、無限の可能性が広がります。

このガイドの使い方

- 実習で作ったシステムを**さらに進化**させるアイデア

- レベル別の**発展課題**（初級→中級→上級）
- 自分で**プロジェクトを考える**ヒント
- 困ったときの**リソース**

次のステップに進みましょう！

2. 発展課題のレベル

発展課題は、3 つのレベルに分かれています。自分のペースで、興味のあるものから挑戦してください。

レベルの目安

レベル	難易度	必要な知識	所要時間
初級	**やさしい**	実習で学んだ内容	1-2 時間
中級	**ふつう**	実習＋少しの応用	3-5 時間
上級	**むずかしい**	新しい技術の習得が必要	1 週間以上

挑戦のコツ

- **初級から始める：**いきなり上級に挑戦せず、段階的に
- **一つずつ：**複数を同時にやらず、一つずつ確実に
- **記録する：**成功も失敗も記録して、学びにする
- **楽しむ：**完璧を目指さず、試行錯誤を楽しむ
- **共有する：**友達や先生に見せて、フィードバックをもらう

3. 初級課題 やさしい

実習で学んだ内容を少し変えて、機能を追加してみましょう。

初級 1

湿度でも制御してみよう

何を作るか

温度だけでなく、**湿度によっても LED の動作を変える**システムを作ります。

学べること

- 複数の条件を組み合わせる方法
- if 文の応用

必要な部品

実習で使った部品だけで OK（追加部品なし）

ヒント

考え方：

- 湿度が 60%以上のとき、LED を点灯させる
- 温度とは別のパターンで点滅させる

- 温度と湿度の両方が高いとき、さらに別の動作をさせる

プログラムの例：

```
if (humi >= 60) {
  // 湿度が高いときの処理
  digitalWrite(LEDPIN, HIGH);
  delay(100);
  digitalWrite(LEDPIN, LOW);
  delay(100);
}
```

挑戦してみよう

- 湿度 60%以上で LED が点滅するように変更
- 温度と湿度の両方が高いときの動作を追加
- SD カードに記録するメッセージを変更
- LCD に湿度の警告メッセージを表示

復習すべき基礎教材

第 7 章：7-12 プログラミングの基本概念（条件分岐）

初級 2

ブザーを追加してアラームを鳴らそう

何を作るか

温度が閾値を超えたら、**ブザーで音を鳴らす**機能を追加します。

学べること

- 新しい部品（ブザー）の使い方
- tone()関数の使い方
- 音による警告の実装

必要な部品

- 圧電ブザー（1 個）
- ジャンパーワイヤー（2 本）

配線

ブザーの接続：

- ブザーの+（長い足） → Arduino D8
- ブザーの-（短い足） → Arduino GND

プログラムのヒント

```
#define BUZZERPIN 8

void setup() {
  pinMode(BUZZERPIN, OUTPUT);
}
```

```
void loop() {
  // 温度が 30 度以上のとき
  if (temp >= 30) {
    tone(BUZZERPIN, 1000, 200);  // 1000Hz、200ms
  }
}
```

tone()関数：

- 第 1 引数：ピン番号
- 第 2 引数：周波数（Hz）
- 第 3 引数：鳴らす時間（ms）

挑戦してみよう

- 温度 30 度以上で「ピー」と鳴らす
- 温度 35 度以上で連続音を鳴らす
- 周波数を変えて音の高さを変える
- メロディを作ってみる（音階を組み合わせる）

復習すべき基礎教材

第 6 章：6-9 アクチュエータの基礎知識

初級 3

測定間隔を変更してみよう

何を作るか

現在 2 秒ごとの測定を、**ボタンで測定間隔を変更できる**ようにします。

学べること

- ボタン入力の基礎
- digitalRead()の使い方
- 変数の活用

必要な部品

- タクトスイッチ（1 個）
- ジャンパーワイヤー（2 本）

配線

ボタンの接続：

- ボタンの片側 → Arduino D3
- ボタンのもう片側 → GND
- D3 と GND の間に 10kΩ プルアップ抵抗

プログラムのヒント

```
#define BUTTONPIN 3
int interval = 2000;  // 測定間隔（ms）

void setup() {
```

```
  pinMode(BUTTONPIN, INPUT_PULLUP);
}

void loop() {
  // ボタンが押されたら
  if (digitalRead(BUTTONPIN) == LOW) {
    interval = 1000;   // 1 秒に変更
  }

  // 測定処理
  delay(interval);
}
```

挑戦してみよう

- ボタンを押すと 1 秒間隔になる
- もう一度押すと 5 秒間隔になる
- 押すたびに間隔が切り替わる（トグル動作）
- 現在の間隔を LCD に表示する

復習すべき基礎教材

第 7 章：7-13 センサーデータ取得の基礎

初級 4

データをグラフ化してみよう

何を作るか

SD カードに記録したデータを、**Excel でグラフにして分析**します。

学べること

- データ分析の基礎
- Excel の使い方
- グラフの作り方

必要なもの

実習で作ったシステムと Excel（または Google スプレッドシート）

手順

1. **データ収集：**
    - システムを数時間動かしてデータを記録
    - SD カードから data.txt を取り出す
2. **Excel で開く：**
    - Excel で data.txt を開く
    - 「データ」→「区切り文字」でカンマを選択
    - データが列に分かれる
3. **グラフ作成：**
    - 温度の列を選択

- 「挿入」→「グラフ」→「折れ線グラフ」
- 湿度も同様にグラフ化

挑戦してみよう

- 温度の時間変化をグラフにする
- 最高温度、最低温度を見つける
- 平均温度を計算する
- 温度と湿度の関係をグラフで確認
- 1 日の温度変化のパターンを分析

復習すべき基礎教材

第 4 章：4-12 データの可視化基礎、4-16 統計分析の基礎

4. 中級課題 ふつう

新しいセンサーを追加したり、機能を組み合わせたりして、より高度なシステムを作ります。

中級 1

複数のセンサーを使ってみよう

何を作るか

DHT11 に加えて、**照度センサーや気圧センサー**を追加して、より詳細な環境モニタリングシステムを作ります。

学べること

- 複数センサーの統合
- アナログ入力の使い方
- I2C 通信の応用

必要な部品（例）

- CdS セル（照度センサー）
- 10kΩ 抵抗
- または、BMP280（気圧センサー、I2C 接続）

アイデア 1：照度センサーを追加

配線：

- CdS セル → A0 と GND
- A0 と 5V の間に 10kΩ 抵抗

プログラム：

```
int lightValue = analogRead(A0);
Serial.print("明るさ: ");
Serial.println(lightValue);
```

アイデア 2：気圧センサーを追加

BMP280 の場合（I2C 接続）：

- VCC → 3.3V（または 5V）
- GND → GND
- SDA → A4
- SCL → A5

ライブラリをインストール：「Adafruit BMP280 Library」

挑戦してみよう

- 照度センサーで明るさを測定
- 暗くなったら LED を自動点灯
- 気圧センサーで気圧を測定
- 気圧変化から天気を予測
- すべてのデータを SD カードに記録
- LCD に複数の情報を交互に表示

復習すべき基礎教材

第 6 章：6-3 センサーの基礎知識、6-6 光センサーの基礎

中級 2

リアルタイムクロック（RTC）を追加しよう

何を作るか

DS3231 などの RTC モジュールを追加して、**正確な日時をデータに記録**します。

学べること

- RTC モジュールの使い方
- I2C 通信で複数デバイスを接続
- タイムスタンプの重要性

必要な部品

- DS3231 RTC モジュール
- ジャンパーワイヤー

配線

DS3231 の接続（I2C）：

- VCC → 5V
- GND → GND
- SDA → A4（LCD と共有）
- SCL → A5（LCD と共有）

💡 **ポイント：**I2C は SDA/SCL を共有できるので、LCD と RTC を同時に接続できます。

プログラムのヒント

ライブラリをインストール：「RTClib」

```
#include <RTClib.h>

RTC_DS3231 rtc;

void setup() {
  rtc.begin();
  // 初回のみ時刻設定
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}
```

```
void loop() {
  DateTime now = rtc.now();

  Serial.print(now.year());
  Serial.print('/');
  Serial.print(now.month());
  Serial.print('/');
  Serial.print(now.day());
  Serial.print(' ');
  Serial.print(now.hour());
  Serial.print(':');
  Serial.print(now.minute());
  Serial.print(':');
  Serial.println(now.second());
}
```

挑戦してみよう

- RTC から現在時刻を取得
- SD カードに日時付きでデータを記録
- LCD に現在時刻を表示
- 特定の時刻にアラームを鳴らす
- 1 時間ごとにデータを集計

復習すべき基礎教材

第 5 章：5-2 通信プロトコルとは（I2C）

中級 3

データをリアルタイムで PC に送信しよう

何を作るか

測定データを**リアルタイムで PC に送信**し、PC で受信してグラフ化します。

学べること

- シリアル通信の応用
- Python でのデータ受信
- リアルタイムグラフ作成

必要なもの

- Python 3.x（PC 側）
- 必要なライブラリ（pyserial、matplotlib）

Arduino 側のプログラム

データを CSV 形式で送信：

```
void loop() {
  float temp = dht.readTemperature();
  float humi = dht.readHumidity();
```

```
  // CSV 形式で送信
  Serial.print(temp);
  Serial.print(",");
  Serial.println(humi);

  delay(2000);
}
```

PC 側のプログラム（Python）

```
import serial
import matplotlib.pyplot as plt

# シリアルポート設定
ser = serial.Serial('COM3', 9600)  # ポート名は環境に応じて変更

temps = []
humis = []

try:
    while True:
        line = ser.readline().decode().strip()
        temp, humi = line.split(',')
        temps.append(float(temp))
        humis.append(float(humi))

        # グラフ更新
        plt.clf()
        plt.plot(temps, label='Temperature')
        plt.plot(humis, label='Humidity')
        plt.legend()
        plt.pause(0.01)

except KeyboardInterrupt:
    ser.close()
```

挑戦してみよう

- シリアル通信で PC にデータ送信
- Python でデータを受信
- リアルタイムでグラフを描画
- 異常値を検出したら警告
- 受信データを CSV ファイルに保存

復習すべき基礎教材

第 5 章：5-16 通信プロトコル HTTP とは（通信の基礎）

中級 4

Web ダッシュボードを作ろう

何を作るか

SD カードのデータを読み取って、**HTML で見やすいダッシュボード**を表示します。

学べること

- HTML の基礎
- JavaScript でのグラフ描画
- データの可視化

手順

1. **SD カードから data.txt を取り出す**
2. **HTML ファイルを作成**
3. **Chart.js ライブラリを使ってグラフ化**
4. **ブラウザで表示**

HTML のサンプル

```
<!DOCTYPE html>
<html>
<head>
  <title>環境モニタリング</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <h1>温湿度データ</h1>
  <canvas id="myChart"></canvas>

  <script>
    // data.txt から読み込んだデータ
    const data = [25.5, 26.0, 26.2, 25.8, ...];

    const ctx = document.getElementById('myChart');
    new Chart(ctx, {
      type: 'line',
      data: {
        labels: ['1', '2', '3', '4', ...],
        datasets: [{
          label: '温度',
          data: data
        }]
```

```
      }
    });
  </script>
</body>
</html>
```

挑戦してみよう

- 温度と湿度を同時にグラフ表示
- 最高値・最低値・平均値を表示
- 時刻軸でグラフを表示
- グラフのデザインをカスタマイズ
- 警告レベルを視覚的に表示

復習すべき基礎教材

第 4 章：4-12 データの可視化基礎、4-14 ダッシュボードの基礎

5. 上級課題 むずかしい

新しい技術を学びながら、実用的なシステムを構築します。挑戦的ですが、達成したときの喜びは大きいです！

上級 1

Wi-Fi 通信でクラウドにデータ送信

何を作るか

ESP8266 や ESP32 を使って、**Wi-Fi 経由でデータをクラウドに送信**します。

学べること

- Wi-Fi 通信の基礎
- クラウドサービスの利用
- HTTP 通信
- IoT プラットフォームの活用

必要なもの

- ESP8266 または ESP32 ボード
- クラウドサービスのアカウント（ThingSpeak、Ambient など無料）

システム構成

センサー → ESP32 → Wi-Fi → クラウド → Web ブラウザ

プログラムの概要

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "あなたの Wi-Fi SSID";
const char* password = "あなたのパスワード";

void setup() {
```

```
  // Wi-Fi 接続
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}

void loop() {
  // データ測定
  float temp = dht.readTemperature();

  // HTTP POST 送信
  HTTPClient http;
  http.begin("https://api.thingspeak.com/update");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  String postData = "api_key=YOUR_API_KEY&field1=" + String(temp);
  http.POST(postData);
  http.end();

  delay(60000);  // 1 分ごと
}
```

挑戦してみよう

- ESP32 で Wi-Fi 接続を確立
- ThingSpeak にデータを送信
- ThingSpeak のグラフで確認
- 複数のセンサー値を同時送信
- スマートフォンから確認
- 異常値を検出したらメール通知

復習すべき基礎教材

第 2 章：2-24 クラウド活用例（IoT プラットフォーム）

第 5 章：5-6 Wi-Fi の基礎、5-16 通信プロトコル HTTP とは

上級 2

スマートフォンアプリで制御

何を作るか

Bluetooth を使って、**スマートフォンアプリからシステムを制御**します。

学べること

- Bluetooth 通信
- スマートフォンアプリ連携
- モバイル UI 設計

必要なもの

- Arduino Unoと Bluetooth モジュール（HC-05 など）
- または ESP32（Bluetooth 内蔵）
- スマートフォン
- Bluetooth アプリ（MIT App Inventor など）

できること

- スマホから測定開始/停止
- スマホで現在の温湿度を表示
- スマホから警告温度を設定
- スマホで SD カードのデータを確認

プログラムのヒント

```
#include <SoftwareSerial.h>

SoftwareSerial BT(10, 11);  // RX, TX

void loop() {
  // スマホからコマンド受信
  if (BT.available()) {
    char cmd = BT.read();

    if (cmd == 'S') {
      // 測定開始
      measuring = true;
    } else if (cmd == 'T') {
      // 温度送信
      BT.print("Temp:");
      BT.println(temp);
    }
  }
}
```

挑戦してみよう

- Bluetooth 接続を確立
- スマホからコマンドを送信
- Arduino からデータを受信
- 簡単なアプリ UI を作成
- リアルタイムでデータ表示
- グラフをアプリに表示

復習すべき基礎教材

第 5 章：5-7 Bluetooth の基礎

上級 3

機械学習で温度予測

何を作るか

蓄積したデータを使って、**機械学習で数時間後の温度を予測**します。

学べること

- 機械学習の基礎
- 時系列データ分析
- Python での機械学習
- 予測モデルの構築

必要なもの

- 十分なデータ（数日～数週間分）
- Python 3.x
- scikit-learn、pandas、numpy

手順の概要

1. **データ収集：**数日間システムを動かしてデータを蓄積
2. **データ前処理：**Python でデータをクリーニング
3. **特徴量エンジニアリング：**時刻、曜日などを特徴量に
4. **モデル訓練：**線形回帰やランダムフォレストで学習
5. **予測実行：**未来の温度を予測
6. **評価：**予測精度を評価

Python コードのサンプル

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# データ読み込み
df = pd.read_csv('data.txt')

# 特徴量とターゲット
X = df[['hour', 'humidity', 'prev_temp']]
y = df['temperature']

# 訓練データとテストデータに分割
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# モデル訓練
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# 予測
predictions = model.predict(X_test)

# 精度評価
score = model.score(X_test, y_test)
print(f"R2 スコア: {score}")
```

挑戦してみよう

- 1 週間分のデータを収集
- Python でデータを前処理
- 線形回帰モデルで予測
- 予測精度を評価
- より高度なモデル（ランダムフォレストなど）を試す
- 予測結果をグラフで可視化

復習すべき基礎教材

第 4 章：4-17 予測分析の基礎、4-18 AI の活用基礎、4-19 機械学習の基礎

上級 4

完全自律型環境制御システム

何を作るか

温度・湿度・照度を測定し、**自動で換気扇、加湿器、照明を制御**する完全自律システムを構築します。

学べること

- 複雑なシステム統合
- リレー制御
- フィードバック制御
- 実用システムの設計

必要な部品

- DHT11（温湿度）
- CdS セル（照度）
- リレーモジュール（3ch 以上）
- 換気扇、加湿器、照明（AC100V 対応）
- LCD、SD カード、RTC

制御ロジックの例

- **温度制御：**
  - 28 度以上 → 換気扇 ON
  - 25 度以下 → 換気扇 OFF
- **湿度制御：**
  - 40%以下 → 加湿器 ON
  - 60%以上 → 加湿器 OFF
- **照明制御：**

- 照度が低い → 照明 ON
- 時刻が 22 時以降 → 照明 OFF（時刻優先）

安全上の注意

⚠️ **重要：**AC100V を扱うため、必ず以下を守ってください：

- 電気工事士の資格がある人の指導の下で作業
- リレーモジュールは定格電圧・電流を確認
- 配線は確実に接続し、絶縁処理を行う
- テストは十分に行い、異常があれば即座に停止

挑戦してみよう

- 複数センサーで環境を測定
- リレーで家電を制御
- ヒステリシス制御で安定化
- 時刻に応じた制御
- 異常時の安全停止機能
- 制御履歴を SD カードに記録
- スマホから遠隔監視

復習すべき基礎教材

第 1 章：1-3 IoT システムの全体像

第 7 章：7-5 システム設計の基本概念、7-14 デバイス制御の基本

6. 自分でプロジェクトを考えよう

発展課題に慣れたら、**自分独自の IoT プロジェクト**を考えてみましょう。

「こんなものがあったら便利」「こんな問題を解決したい」というアイデアを形にしてください。

プロジェクトの考え方

ステップ 1：問題を見つける

身の回りで困っていることを探します：

- 部屋が暑いか寒いかわからない
- 植物の水やりを忘れる
- 冷蔵庫のドアを開けっ放しにする
- ペットの様子が心配
- 勉強時間を記録したい

ステップ 2：解決方法を考える

IoT でどう解決できるか考えます：

- 温度センサー → 快適温度を通知
- 土壌湿度センサー → 水やり時期を通知
- ドアセンサー → 開いたままなら警告
- カメラ → 画像をスマホに送信
- タイマー → 時間を自動記録

ステップ 3：必要な部品をリストアップ

実現に必要なセンサーや部品を調べます

ステップ４：小さく始める

最初から完璧を目指さず、まず基本機能だけを実装

ステップ５：少しずつ改良

動いたら、機能を追加していく

プロジェクトアイデア集

家庭・生活

- スマート目覚まし時計（温度・照度に応じて起こす）
- 植物自動水やりシステム
- ペット自動給餌器
- エアコン遠隔制御システム
- 冷蔵庫の開閉記録システム

学習・仕事

- 学習時間記録システム
- 姿勢チェッカー
- 集中力測定システム
- TODO 管理デバイス

趣味・エンタメ

- 楽器練習時間記録
- ゲームプレイ時間記録
- 読書記録システム
- 運動記録デバイス

環境・エコ

- 電力消費モニター
- 水使用量記録
- ゴミ分別支援システム
- CO2 濃度モニター

安全・防犯

- 窓開閉検知システム
- 火災警報システム
- 忘れ物防止デバイス
- 地震検知・通知システム

プロジェクト計画シート

プロジェクトを始める前に、以下の項目を整理しましょう：

項目	内容
プロジェクト名	
解決したい問題	

実現する機能	
必要なセンサー	
必要なアクチュエータ	
データ記録方法	
表示方法	
予算	
完成目標時期	

7. 困ったときのリソース

学習リソース

公式ドキュメント

- **Arduino 公式：** https://www.arduino.cc/
- **Arduino 日本語リファレンス：** http://www.musashinodenpa.com/arduino/ref/

チュートリアルサイト

- **Arduino Project Hub：** https://create.arduino.cc/projecthub
- **Instructables：** https://www.instructables.com/circuits/arduino/
- **Hackster.io：** https://www.hackster.io/arduino

質問できる場所

オンラインコミュニティ

- **Arduino Forum：** https://forum.arduino.cc/
- **Stack Overflow：** https://stackoverflow.com/questions/tagged/arduino
- **Reddit r/arduino：** https://www.reddit.com/r/arduino/

日本語コミュニティ

- **Qiita：** Arduino 関連の技術記事
- **Zenn：** Arduino・IoT 関連の記事
- **teratail：** 日本語でプログラミング質問

質問のコツ

1. **何をしたいか**を明確に
2. **何が起きているか**を具体的に
3. **エラーメッセージ**を全文コピー
4. **試したこと**を説明
5. **コード**を共有（長い場合は Gist や Pastebin を利用）

部品の入手先

国内ショップ

- **秋月電子通商：**https://akizukidenshi.com/
- **千石電商：**https://www.sengoku.co.jp/
- **マルツオンライン：**https://www.marutsu.co.jp/
- **スイッチサイエンス：**https://www.switch-science.com/
- **Amazon：**Arduino 互換品やセンサーキット

購入のヒント

- 最初はキットを購入すると便利（必要な部品がセット）
- 予備を含めて購入（壊れることもある）
- データシートを確認（仕様を理解）

トラブルシューティング

よくあるトラブルと解決方法

プログラムが書き込めない

- ボードとポートの設定を確認
- USB ケーブルを差し直す
- Arduino IDE を再起動
- ドライバを再インストール

センサーが動かない

- 配線を確認（VCC、GND、信号線）
- ピン番号がプログラムと一致しているか確認
- ライブラリがインストールされているか確認
- センサーを予備と交換して確認

データがおかしい

- シリアルモニタで生データを確認
- 配線が正しいか再確認
- delay()で測定間隔を確保
- センサーの特性を理解（応答時間など）

システムが動かなくなる

- 無限ループに陥っていないか確認
- メモリ不足でないか確認
- 配線のショートがないか確認
- 電源容量が足りているか確認

8. 次のステップ

さらに学びたい人へ

IoT 技術は奥深く、学ぶことはまだまだたくさんあります。

技術的なスキルアップ

- **プログラミング：**C++、Python、JavaScript を深く学ぶ
- **電子工学：**回路設計、電源設計を学ぶ

- **通信技術：**Wi-Fi、Bluetooth、LoRaWAN を学ぶ
- **クラウド：**AWS IoT、Azure IoT を学ぶ
- **機械学習：**TensorFlow、PyTorch で AI を学ぶ

継続学習のコツ

習慣化のポイント

- **小さく始める：**週に 1 時間からで OK
- **記録する：**学んだことをブログやノートに
- **共有する：**SNS で作品を公開
- **仲間を作る：**一緒に学ぶ仲間を見つける
- **楽しむ：**興味のあることから始める

9. 最後に

🎓 あなたの旅は、ここから始まる

4 回の実習を通じて、あなたは **IoT システムを作る基礎**を身につけました。

でも、これは**スタート地点**に過ぎません。

覚えておいてほしいこと

1. 失敗は成功のもと

エラーが出ても、動かなくても、それは失敗ではありません。学びのプロセスです。

2. 完璧を目指さない

最初から完璧なものは作れません。まず動くものを作り、少しずつ改良していきましょう。

3. 楽しむことが一番

義務感で続けるより、「面白い！」「楽しい！」という気持ちを大切に。

4. 学び続ける

技術は日々進化します。常に新しいことを学び、試し続けましょう。

5. 共有する

作ったものを人に見せ、フィードバックをもらいましょう。教えることで、自分の理解も深まります。

実習を振り返って

この実習で、あなたは多くのことを学びました：

- 理論を実践で確認する方法
- 自分の手で作る喜び
- エラーと向き合う力
- 試行錯誤しながら学ぶ姿勢
- 協力して問題を解決する力

これらは、IoT 技術だけでなく、あらゆる学習と仕事に役立つスキルです。

今日からできること

1. このガイドの初級課題を一つ選ぶ
2. 必要な部品をリストアップする
3. まずは基本機能だけを実装してみる
4. 動いたら、友達や先生に見せる

5. 次の課題を選ぶ

小さな一歩が、大きな変化につながります。

さあ、次のステップへ進みましょう！

あなたの IoT プロジェクトが、素晴らしいものになることを願っています。

IoT 技術応用
実習 2　クラウド接続
実習ガイド

🎯 この実習で学ぶこと

実習の目標

実習 1 では、センサーを使ってデータを取得する基本を学びました。
実習 2 では、そのデータを**インターネットを通じてクラウドに送信**し、**どこからでもアクセスできる仕組み**を作ります。

💎 習得する 4 つのスキル

1. **Wi-Fi 通信技術**：ESP32 を使ってインターネットに接続する方法を学ぶ
2. **クラウド連携**：センサーデータをクラウドサービスに送信する技術を習得
3. **データ可視化**：クラウド上でデータをグラフ化し、分析する方法を理解
4. **システム設計**：IoT システム全体の構成を考え、実装する力を養う

この実習の最大の特徴は、**「データがインターネットを通じて世界とつながる」**という体験です。教室で測定したデータが、瞬時にクラウドに送信され、スマートフォンやパソコンからいつでも確認できるようになります。これが、**IoT の真の価値**です。

🛠️ 使用する機材・サービス

ハードウェア

- **ESP32 開発ボード**
  - Wi-Fi/Bluetooth 内蔵のマイコン
  - 実習 1 の Arduino Uno より高機能
  - インターネット接続が可能
- **DHT22 温湿度センサー**
  - 実習 1 の DHT11 より高精度
  - 測定範囲：温度-40～80℃、湿度 0～100%
  - 精度：温度±0.5℃、湿度±2%
- **USB-C ケーブル**：ESP32 とパソコンを接続
- **ブレッドボード・ジャンパー線**：配線用

ソフトウェア・サービス

- **Arduino IDE**：ESP32 のプログラミング環境

- **Ambient または Thingspeak**：クラウド IoT サービス
  - ✓ **Ambient**：日本語対応、登録簡単、無料プラン充実
  - ✓ **ThingSpeak**：グローバル標準、MathWorks 製、無料利用可能
  - ✓ どちらかを選択して使用します

実習 1 との違いは、**Wi-Fi 機能を持つ ESP32 を使う**ことと、**クラウドサービスを活用する**ことです。
これにより、データをインターネット経由で送信・共有できるようになります。

実習の進め方

回	テーマ	主な内容
Part 1	**Wi-Fi 接続とネットワーク基礎**	・ESP32 の基本操作 ・Wi-Fi 接続の実装 ・接続状態の確認
Part 2	**センサーデータ取得**	・DHT22 センサーの接続 ・温湿度データの取得 ・シリアル通信での確認
Part 3	**クラウドサービス連携**	・Ambient/ThingSpeak の設定 ・データ送信プログラム ・送信の確認とトラブル対応
Part 4	**データ可視化と分析**	・グラフ作成と設定 ・データの読み取りと分析 ・システムの改善

各 Part は独立していますが、**順番に進めることで理解が深まります。**
前の Part の内容を確実に理解してから次に進みましょう。

実習の全体像：データの流れを理解しよう

センサーで測定

DHT22 が温度と湿度を測定します
↓

ESP32 で処理

測定データを ESP32 が受け取り、デジタルデータに変換します
↓

Wi-Fi で送信

ESP32 が Wi-Fi 経由でクラウドサービスにデータを送信します
↓

クラウドで保存・表示

Ambient/ThingSpeak がデータを保存し、グラフ化します
↓

どこからでもアクセス

スマホや PC から、いつでもデータを確認できます
このデータの流れ全体を理解することが、IoT システムを設計する第一歩です。各 Part で、この流れのどの部分を学んでいるか意識しましょう。

実習開始前の準備チェックリスト

Part１　開始前

実習 1 の内容を復習し、Arduino IDE の基本操作を確認した
第 2 章「IoT のネットワーク技術」を学習した
ESP32 開発ボード、USB-C ケーブルを用意した
Arduino IDE に ESP32 ボードのサポートを追加した（初回のみ）

Part２　開始前

Part 1 の内容を復習し、Wi-Fi 接続ができることを確認した
DHT22 センサー、ブレッドボード、ジャンパー線を用意した
DHT ライブラリ（**DHT sensor library by Adafruit**）を Arduino IDE にインストールした
※実習 1 と異なる PC や Arduino IDE を利用する場合は、ライブラリマネージャで DHT ライブラリをインストールしてください。

Part３　開始前

Part 2 までの動作を確認し、センサーデータが取得できることを確かめた
第 5 章「IoT プラットフォーム」を学習した
Ambient または ThingSpeak のアカウントを作成した
チャンネルを作成し、API キーを控えた

Part 4 開始前

Part 3 でクラウドへのデータ送信が成功し、データが蓄積されていることを確認した
クラウドサービスの基本的な操作方法を理解した

📖 実習の進め方とワークブックの使い方

ワークブックの構成

各 Part には専用の**ワークブック**があります。ワークブックには以下の内容が含まれています：

- **学習目標**：この回で何を学ぶかを明確化
- **理論パート**：必要な知識の解説
- **実践パート**：実際の作業手順
- **確認パート**：理解度チェックと振り返り
- **記録欄**：気づきや疑問を記録

効果的な学習方法

予習：該当する教科書の章を読み、基礎知識を確認
理解：ワークブックの理論パートを読み、疑問点をメモ
実践：手順に従って作業し、動作を確認
試行錯誤：うまくいかない場合は原因を考え、調整
振り返り：確認問題に答え、学んだことを整理
試行錯誤は学びの宝庫です。エラーが出たり、うまく動かないことは当然のプロセスです。
なぜそうなったのかを考え、解決策を探すことで、深い理解につながります。

💡 実習を成功させるポイント

1. 理論と実践を結びつける
「なぜこのコードが必要なのか」「この設定は何をしているのか」を常に考えながら作業しましょう。
2. 段階的に進める
一度に全てを完成させようとせず、一つ一つの機能を確認しながら進めることが大切です。
3. エラーを恐れない
エラーメッセージは「問題を教えてくれる先生」です。落ち着いてメッセージを読み、原因を探りましょう。
4. 記録を残す
気づいたこと、疑問に思ったこと、解決方法などをワークブックに記録しましょう。後で見返すときに役立ちます。
5. 仲間と協力する
困ったときは周りの人に相談したり、成功体験を共有したりすることで、学びが深まります。

6. 発展課題にチャレンジ

基本ができたら、自分なりの改良や応用を考えてみましょう。創造力が育ちます。

🌟 実習 2 で身につく力

この実習を通じて、以下の力が身につきます：

技術的スキル

- Wi-Fi 通信の実装と設定
- クラウドサービスとの連携技術
- IoT システムの構築と運用
- データの取得・送信・可視化の一連のプロセス

思考力・問題解決力

- システム全体を俯瞰して考える力
- トラブルシューティング能力
- データから意味を読み取る分析力
- より良いシステムを設計する創造力

実社会での応用力

- 実際の IoT システムの仕組みを理解
- センサーネットワークの構築方法
- リモート監視システムの実現
- データ駆動型の意思決定の基礎

🚀 実習の先にあるもの

この実習で学ぶ技術は、以下のような実社会の応用につながっています：

- **スマート農業**：畑の温湿度をリアルタイムで監視し、最適な環境を維持
- **スマートホーム**：家の中の環境を外出先から確認・制御
- **環境モニタリング**：都市の気象データを収集し、分析
- **工場の自動化**：製造現場の状態を常時監視し、異常を早期検知
- **健康管理**：体温や活動量をクラウドで管理し、健康維持に活用

皆さんが今学んでいることは、**未来の社会を支える技術**なのです。

📚 参考資料とリンク

オンライン教材

- ESP32 公式ドキュメント（英語）
- Ambient 公式サイト（日本語）
- ThingSpeak 公式サイト（英語）

- Arduino IDE ESP32 サポートページ

各 Part のワークブックには、具体的な参考資料へのリンクが記載されています。必要に応じて参照してください。

❓ 困ったときは

トラブルシューティングの基本

1. **エラーメッセージを読む**：何が問題なのか、メッセージが教えてくれます
2. **配線を確認する**：接続が正しいか、緩んでいないかチェック
3. **コードを見直す**：スペルミス、記号の間違いがないか確認
4. **シリアルモニタで確認**：途中経過を出力して、どこで問題が起きているか特定
5. **一つ前の状態に戻る**：動いていた状態から、少しずつ変更を加える

質問する前に

- ワークブックの「よくある問題と解決方法」を確認しましたか？
- エラーメッセージの内容を記録しましたか？
- 何を試して、どんな結果になったか説明できますか？

これらを整理してから質問すると、的確なアドバイスが得られやすくなります。

🎓 さあ、はじめよう！

実習 2 では、**センサーデータをインターネットでつなげる**という、IoT の本質的な体験をします。

最初は難しく感じるかもしれませんが、一つ一つ丁寧に進めていけば、必ず理解できます。
わからないことがあったら、遠慮せず質問してください。
試行錯誤を楽しみながら、新しい技術を身につけていきましょう。

自分の手で、データを世界につなげる瞬間を体験してください！

IoT 技術応用教材
実習２　クラウド接続
Part１（Wi-Fi 接続とネットワーク基礎）ワークブック

📝 **実習記録**

氏名	
実習日	年　　月　　日

🎯 Part 1 の学習目標

この Part で学ぶこと

1. ESP32 マイコンの基本的な使い方を理解する
2. Wi-Fi 通信の仕組みと設定方法を学ぶ
3. ESP32 を Wi-Fi ネットワークに接続できるようになる
4. シリアルモニタで接続状態を確認する方法を習得する

Part 1 は、実習 2 全体の**基礎**となります。ここで確実に Wi-Fi 接続ができるようになることが、次のステップへの鍵です。焦らず、一つ一つ確認しながら進めましょう。

項目	内容
前提知識	実習 1 の内容、Arduino IDE の基本操作、第 2 章の学習
使用機材	ESP32 開発ボード、USB-C ケーブル、パソコン

📚 理論編：Wi-Fi 通信の基礎を学ぼう

1. ESP32 とは？

ESP32 は、中国の Espressif Systems 社が開発した **Wi-Fi と Bluetooth を内蔵したマイコン**です。

Arduino Uno との違い

機能	Arduino Uno	ESP32
Wi-Fi	❌ なし	✅ あり（2.4GHz 帯）
Bluetooth	❌ なし	✅ あり（BLE 対応）

動作周波数	16 MHz	最大 240 MHz
メモリ	2 KB RAM	520 KB RAM
価格	比較的高価	非常に安価

ESP32 の強み：インターネット接続が必要な IoT プロジェクトに最適です。

2. Wi-Fi 通信の仕組み

Wi-Fi とは？

Wi-Fi（ワイファイ）は、無線でインターネットに接続するための技術です。正式名称は「IEEE 802.11」という規格です。

Wi-Fi 接続の 3 つのステップ

1. **アクセスポイント（AP）の検索**
    - 周囲の Wi-Fi ネットワークを探します
    - 各ネットワークには「SSID」という名前があります
2. **認証（ログイン）**
    - SSID とパスワードを使って認証します
    - セキュリティ方式：WPA2、WPA3 など
3. **IP アドレスの取得**
    - DHCP（自動設定）で IP アドレスをもらいます
    - IP アドレスは、ネットワーク上の住所のようなものです

3. SSID と パスワード

SSID（Service Set Identifier）

Wi-Fi ネットワークの**名前**です。スマホの Wi-Fi 設定画面に表示されているネットワーク名が SSID です。

- 例：「School-WiFi」「Lab-Network」など
- 大文字・小文字を区別します

パスワード

ネットワークにアクセスするための**鍵**です。

- WPA2-PSK の場合、8 文字以上が一般的
- セキュリティのため、本来はプログラムには直接書かない方が良い（発展課題）

ESP32 は 2.4GHz 帯のみ対応：

多くの Wi-Fi ルーターは 2.4GHz と 5GHz の両方に対応していますが、ESP32 は **2.4GHz のみ**に対応しています。
接続するネットワークが 2.4GHz であることを確認してください。

4. シリアルモニタとは？

シリアルモニタは、ESP32 とパソコンの間で文字データをやり取りするツールです。

用途

- **デバッグ**：プログラムの動作状況を確認
- **データ表示**：センサーの値などを表示
- **エラー確認**：問題が起きた時の情報を見る

ボーレート（通信速度）

シリアル通信の速度を表す値です。プログラムとシリアルモニタで**同じ値**に設定する必要があります。

- 一般的な値：9600、115200 など
- 今回の実習では **115200** を使用します

✍️ 理解度チェック（理論編）

問 1：ESP32 の特徴

ESP32 が Arduino Uno より優れている点を 3 つ挙げてください。

1.
2.
3.

問 2：Wi-Fi 接続の用語

以下の用語を説明してください。

SSID：

IP アドレス：

ボーレート：

問 3：2.4GHz 帯について

ESP32 が 2.4GHz 帯のみに対応していることの利点と欠点を考えてみましょう。

利点：

欠点：

🔧 実践編：ESP32 を Wi-Fi に接続しよう

準備：開発環境のセットアップ

1. Arduino IDE で ESP32 ボードを追加

（初回のみ必要な作業です）

1. Arduino IDE を開く
2. 「ツール」→「ボード」→「ボードマネージャ」を開く
3. 検索欄に「esp32」と入力
4. 「esp32 by Espressif Systems」 が見つかればインストールで終了
5. 見つからない場合は、「ファイル」→「基本設定」を選択 基本設定画面の「設定」タブを開く
6. 「追加のボードマネージャの URL」に以下を入力：
   https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
7. 「OK」をクリック
8. あらためて「ツール」→「ボード」→「ボードマネージャ」を開く
9. 検索欄に「esp32」と入力
10. 「esp32 by Espressif Systems」をインストール

※バージョンによって、検索で出てくる場合と、追加しないと出てこない場合があります。

2. ESP32 をパソコンに接続

1. USB-C ケーブルで ESP32 とパソコンを接続
2. 「ツール」→「ボード」から「ESP32 Dev Module」を選択
3. 「ツール」→「ポート」から接続されたポートを選択
   - ✓ Windows: COM3、COM4 など

プログラム作成：Wi-Fi 接続プログラム

3. 基本的な Wi-Fi 接続プログラムを入力

以下のプログラムを新しいスケッチに入力してください。

```
#include <WiFi.h>                    // Wi-Fi 設定（あなたのネットワークに合わせて変更）
const char* ssid = "あなたの SSID";  // SSID に置き換える
const char* password = "あなたのパスワード"; // パスワードに置き換える

void setup() {                              // シリアル通信開始
  Serial.begin(115200);
  delay(1000);
  Serial.println();
  Serial.println("==============================");
  Serial.println("Wi-Fi 接続プログラム開始");
  Serial.println("==============================");
```

```
// Wi-Fi 接続開始
  Serial.print("接続先: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

// 接続完了まで待機
  Serial.print("接続中");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

// 接続成功
  Serial.println();
  Serial.println("Wi-Fi 接続成功！");
  Serial.println("==============================");

// 接続情報を表示
  Serial.print("IP アドレス: ");
  Serial.println(WiFi.localIP());
  Serial.print("MAC アドレス: ");
  Serial.println(WiFi.macAddress());
  Serial.print("電波強度(RSSI): ");
  Serial.print(WiFi.RSSI());
  Serial.println(" dBm");
  Serial.println("==============================");
}
void loop() { // 10 秒ごとに接続状態を確認
  delay(10000);
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Wi-Fi 接続中");
    Serial.print("電波強度: ");
    Serial.print(WiFi.RSSI());
    Serial.println(" dBm");
  } else {
    Serial.println("✗ Wi-Fi 切断されました");

// 再接続を試みる
    WiFi.reconnect();
```

```
  }
}
```

重要：SSID とパスワードの設定

プログラムの 3 行目と 4 行目の"あなたの SSID"と"あなたのパスワード"を、実際に接続する Wi-Fi ネットワークの情報に**必ず書き換えてください。**

4. プログラムを ESP32 に書き込む

1. 左上の「✓（検証）」ボタンをクリックして、エラーがないか確認
2. エラーがなければ、「→（書き込み）」ボタンをクリック
3. 書き込みが完了するまで待つ（30 秒～1 分程度）

書き込み中に「Connecting.....」と表示されて止まる場合は、ESP32 の **BOOT ボタン**を押して書き込みを開始してください。

5. シリアルモニタで動作確認

1. 右上の「🔍（シリアルモニタ）」アイコンをクリック
2. 右下のボーレート設定を「115200 baud」に変更
3. ESP32 のリセットボタンを押す　（USB コネクタの左側のボタン）
4. シリアルモニタに接続情報が表示されることを確認

成功時の表示例：

==========
Wi-Fi 接続プログラム開始
==========
接続先: School-WiFi 接続中..... ✓ Wi-Fi 接続成功！
==========
IP アドレス: 192.168.1.23 MAC アドレス: 24:6F:28:XX:XX:XX 電波強度(RSSI): -45 dBm
==========

接続情報の記録

あなたの ESP32 の接続情報を記録しましょう

接続先 SSID：

取得した IP アドレス：

MAC アドレス：

電波強度（RSSI）：

接続にかかった時間：

🔍 プログラムの解説

重要な関数の説明

WiFi.begin(ssid, password)

機能：指定された SSID とパスワードで Wi-Fi 接続を開始します。
戻り値：なし（接続完了を待つ必要があります）

WiFi.status()

機能：現在の Wi-Fi 接続状態を返します。
主な戻り値：

- WL_CONNECTED：接続成功
- WL_NO_SSID_AVAIL：SSID が見つからない
- WL_CONNECT_FAILED：接続失敗
- WL_IDLE_STATUS：接続準備中

WiFi.localIP()

機能：ESP32 に割り当てられた IP アドレスを返します。
例：192.168.1.23 のような値

WiFi.RSSI()

機能：Wi-Fi の電波強度（受信信号強度）を返します。
単位：dBm（デシベルミリワット）

- -30 dBm：非常に強い（アクセスポイントのすぐ近く）
- -50 dBm：強い（良好な接続）
- -70 dBm：普通（まあまあの接続）
- -90 dBm：弱い（接続不安定になる可能性）

WiFi.reconnect()

機能：Wi-Fi 接続が切れた時に、再接続を試みます。
用途：電波状況の悪化などで切断された時の回復

⚠️ トラブルシューティング

問題が起きた時の確認ポイント

❌「接続中.....」で止まる（接続できない）

原因と対策：

- **SSID やパスワードの間違い**
  - 大文字・小文字、全角・半角に注意して再確認
  - スペースや特殊文字が正しく入力されているか確認
- **5GHz 帯に接続しようとしている**
  - ルーターの設定で 2.4GHz 帯の SSID を確認
  - SSID に「-5G」などが付いている場合は 5GHz 帯です
- **電波が弱い**
  - アクセスポイントに近づいてみる
  - 障害物を減らす

❌ シリアルモニタに文字化けが表示される

原因と対策：

- **ボーレートの不一致**
  - シリアルモニタのボーレートを 115200 に設定
  - プログラムの Serial.begin(115200)と一致させる

❌ 書き込みエラーが出る

原因と対策：

- **ポート選択ミス**
  - 正しいポートが選択されているか確認
  - USB ケーブルを抜き差しして再認識させる
- **ボード設定ミス**
  - 「ESP32 Dev Module」が選択されているか確認
- **ESP32 が応答しない**
  - BOOT ボタンを押しながら書き込みを試す
  - リセットボタンを押してから再試行

☑ 確認テスト（実践編）

以下の項目ができたらチェックしましょう

- ☐ ESP32 がパソコンに認識される
- ☐ Arduino IDE で ESP32 にプログラムを書き込める
- ☐ Wi-Fi に接続できる（IP アドレスが表示される）
- ☐ シリアルモニタで接続情報が確認できる
- ☐ 電波強度（RSSI）の意味を理解している
- ☐ 接続が切れた時に再接続できる

発展問題：プログラムの改良

以下の機能を追加してみましょう（余裕があれば）：

1. 接続できなかった場合に、「接続失敗」と表示する
2. RSSI の値に応じて「強い」「普通」「弱い」と表示する
3. 接続にかかった時間を測定して表示する

追加した機能と工夫した点：

📝 振り返り

1. 今日学んだことで最も重要だと思うことは何ですか？

2. 難しかった点、つまずいた点はありましたか？それをどう解決しましたか？

3. Wi-Fi 接続ができるようになって、どんなことができそうだと思いますか？

4. 次回（Part 2）に向けて準備したいことや、確認したいことはありますか？

🎉 Part 1 完了！

ESP32 を Wi-Fi に接続することができました。
これで、**インターネットにつながるデバイス**を作る準備が整いました。

次の Part 2 では、この Wi-Fi 接続を活かして、センサーデータを取得します。
実習 1 で学んだセンサーの知識と、今日学んだネットワークの知識が組み合わさります。

一歩一歩、確実に IoT システムが形になっていきます。

☑ 実習振り返りシート（IoT 技術応用 - 実習２ Part１）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習２　クラウド接続
Part２（センサーデータ取得とシリアル通信）ワークブック

📝 **実習記録**

氏名	
実習日	年　　月　　日

🎯 Part 2 の学習目標

この Part で学ぶこと

1. DHT22 温湿度センサーの仕組みと特性を理解する
2. ESP32 とセンサーの配線方法を習得する
3. センサーライブラリを使ってデータを取得する
4. シリアル通信でデータを確認・デバッグする技術を学ぶ

Part 2 では、Part 1 の Wi-Fi 接続に**センサーによるデータ取得**を組み合わせます。実習 1 で学んだ DHT11 の知識を活かしながら、より高精度な DHT22 を使います。

📚 理論編：DHT22 センサーを学ぼう

1. DHT22 センサーとは？

DHT22（AM2302）は、温度と湿度を同時に測定できる**デジタル出力センサー**です。

DHT11 との比較

項目	DHT11（実習 1）	DHT22（実習 2）
温度範囲	－20～60℃	-40～80℃
温度精度	±2℃	±0.5℃
湿度範囲	5 ～95%	0～100%
湿度精度	±5%	±2%
価格	安価	やや高価

※製品のバージョンによって異なります。また、実際には製品の個体差もあります。

DHT22 の利点：測定範囲が広く、精度が高いため、実用的な IoT システムに適しています。

2. DHT22 の動作原理

温度測定

サーミスタという温度センサーを使用。温度が変わると電気抵抗が変化する性質を利用して温度を測定します。

湿度測定

静電容量式湿度センサーを使用。湿度が変わると静電容量が変化する性質を利用して湿度を測定します。

デジタル出力

センサー内部のマイコンがアナログ信号をデジタル信号に変換し、独自のプロトコルで ESP32 に送信します。

3. センサーのピン配置

DHT22 はセンサー部品単体の場合は、**4 本のピン**がありますが、1 本は NC ピンで使用するのは 3 本です。

1. **VCC（+）**：電源（3.3V または 5V）
2. **DATA**：データ信号線
3. **NC**：未接続（Not Connected）
4. **GND（-）**：グラウンド

部品単体で利用する場合は、プルアップ抵抗が必要です。データシートで確認してください。
モジュール化されている DHT22 モジュールは、**ピンは 3 本**で、通常、**プルアップ抵抗が内蔵**されています。

4. シリアル通信とデバッグ

シリアルモニタの活用

センサーから取得したデータを確認する最も基本的な方法は、**シリアルモニタ**を使うことです。

デバッグの重要性

- **データの確認**：センサーが正しく動作しているか確認
- **エラーの発見**：配線ミスやプログラムのバグを発見
- **動作の理解**：プログラムの流れを視覚化

✍️ 理解度チェック（理論編）

問 1：センサーの比較

DHT22 が DHT11 より優れている点を 3 つ挙げてください。

1.
2.
3.

問 2：測定原理

DHT22 は温度と湿度をどのような原理で測定していますか？

温度測定：

湿度測定：

問 3：測定間隔

DHT22 の最短測定間隔が 2 秒である理由を考えてみましょう。

考察：

🔧 実践編：DHT22 と ESP32 を接続しよう

準備：ライブラリのインストール

1. DHT センサーライブラリをインストール

1. Arduino IDE を開く
2. 「スケッチ」→「ライブラリをインクルード」→「ライブラリを管理」を選択
3. 検索欄に「DHT sensor library」と入力
4. 「DHT sensor library by Adafruit」を探してインストール
5. 依存ライブラリ「Adafruit Unified Sensor」のインストールを求められたら「すべてインストール」をクリック

ライブラリは 1 度インストールすれば、以降のプログラムでも使用できます。

配線：ESP32 と DHT22 を接続

2. 配線図に従って接続する

配線表

DHT22	ESP32	備考
VCC（+）	3.3V または 5V	電源（3.3V 推奨）
DATA	GPIO 4	データ信号線
GND（-）	GND	グランド

配線時の注意点：

- 配線前に**必ず ESP32 の電源を切る**
- VCC と GND を**絶対に逆に接続しない**（センサーが壊れます）
- ジャンパー線がしっかり挿さっているか確認
- ブレッドボードの縦の列を正しく使う

プログラム作成：温湿度データ取得

3. Part 1 の Wi-Fi 接続プログラムにセンサー機能を追加

以下のプログラムを新しいスケッチに入力してください。

```
#include <WiFi.h>
#include <DHT.h> // Wi-Fi 設定
const char* ssid = "あなたの SSID";
const char* password = "あなたのパスワード";

// DHT22 設定
#define DHTPIN 4  // DATA ピンを GPIO 4 に接続
#define DHTTYPE DHT22 // センサータイプを DHT22 に指定
DHT dht(DHTPIN, DHTTYPE);

void setup() { // シリアル通信開始
  Serial.begin(115200);
  delay(1000);
  Serial.println();
  Serial.println("==============================");
  Serial.println("ESP32 + DHT22 データ取得");
  Serial.println("==============================");
```

```
// DHT22 センサー初期化
  dht.begin();
  Serial.println("DHT22 センサー: 初期化完了");

// Wi-Fi 接続
  Serial.print("Wi-Fi 接続中: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println();
  Serial.println("✓ Wi-Fi 接続成功");
  Serial.print("IP アドレス: ");
  Serial.println(WiFi.localIP());
  Serial.println("==============================");
  Serial.println();
  }

void loop() {        // 温湿度データを読み取り
  float humidity = dht.readHumidity(); // 湿度を取得（%）
  float temperature = dht.readTemperature(); // 温度を取得（℃）

// データ読み取りエラーチェック
  if (isnan(humidity) || isnan(temperature)) {
   Serial.println("センサー読み取りエラー");
   delay(2000);
   return;
  }

// データを表示
  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(" %");
```

```
// 不快指数を計算（応用例）
  float discomfortIndex = 0.81 * temperature + 0.01 * humidity * (0.99 *
temperature - 14.3) + 46.3;
  Serial.print("不快指数: ");
  Serial.println(discomfortIndex);
  Serial.println("------------------");
  Serial.println();

// 2 秒待機（DHT22 の最短測定間隔）
  delay(2000);
}
```

プログラムのポイント：

- dht.readHumidity()：湿度を読み取る
- dht.readTemperature()：温度を読み取る
- isnan()：読み取りエラーをチェック（Not a Number の判定）
- delay(2000)：2 秒待機（DHT22 の制約）

4. プログラムを書き込んで動作確認

1. SSID とパスワードを自分の環境に合わせて変更
2. 配線を最終確認
3. 「検証」ボタンでエラーチェック
4. 「書き込み」ボタンで ESP32 に書き込み
5. シリアルモニタを開く（ボーレート: 115200）
6. 温湿度データが表示されることを確認

成功時の表示例：

```
================================
 ESP32 + DHT22 データ取得
================================
DHT22 センサー: 初期化完了 Wi-Fi 接続中: School-WiFi .....
✓ Wi-Fi 接続成功 IP アドレス: 192.168.1.23
================================
--- 測定データ --- 温度: 23.5 ℃ 湿度: 55.2 % 不快指数: 69.8 ------------------
```

測定データの記録

複数回測定して記録しましょう（5 回以上）

測定回	温度（℃）	湿度（%）	不快指数
1 回目			
2 回目			
3 回目			
4 回目			
5 回目			

測定結果の考察

データの変動：測定値は毎回同じでしたか？違った場合、なぜだと思いますか？

不快指数について：計算された不快指数は体感と合っていましたか？

🔍 プログラムの解説

重要な関数とコマンドの説明

dht.begin()
機能：DHT22 センサーを初期化します。setup()関数内で 1 回だけ実行します。

dht.readHumidity()
機能：現在の湿度を読み取ります。
戻り値：湿度（%）を浮動小数点数で返します
エラー：読み取り失敗時は NaN（Not a Number）を返します

dht.readTemperature()
機能：現在の温度を読み取ります。
戻り値：温度（℃）を浮動小数点数で返します
オプション：dht.readTemperature(true)とすると華氏（°F）で取得

isnan(値)

機能：値が NaN（無効な数値）かどうかを判定します。

戻り値：NaN なら true、正常な値なら false

用途：センサー読み取りエラーの検出に使用

不快指数について

不快指数（DI: Discomfort Index）は、温度と湿度から人間の快適さを数値化したものです。

計算式

DI = 0.81 × T + 0.01 × H × (0.99 × T - 14.3) + 46.3

（T: 温度[℃]、H: 湿度[%]）

快適度の目安

- **55 未満**：寒い
- **55〜60**：肌寒い
- **60〜65**：何も感じない
- **65〜70**：快適
- **70〜75**：暑くない
- **75〜80**：やや暑い
- **80〜85**：暑くて汗が出る
- **85 以上**：非常に暑い

⚠️ トラブルシューティング

問題が起きた時の確認ポイント

❌ **「センサー読み取りエラー」が繰り返し表示される**

原因と対策：

- **配線ミス**
  - 配線表と照らし合わせて再確認
  - 特に VCC と GND が逆になっていないか確認
  - ジャンパー線がしっかり挿さっているか確認
- **ピン番号の間違い**
  - プログラムの DHTPIN とハードウェアの接続が一致しているか
  - GPIO 4 に接続されているか確認
- **センサーの不良**
  - 別の DHT22 センサーで試してみる
  - センサーに物理的な損傷がないか確認

❌ **温度や湿度の値がおかしい（異常に高い/低い）**

原因と対策：

- **センサーの発熱**
  - ESP32 やセンサーが発熱していないか確認
  - センサーを基板から離して設置
- **測定環境の影響**

- 直射日光や暖房機器の近くを避ける
- 風通しの良い場所で測定

- **センサータイプの誤指定**
  - DHTTYPE DHT22 になっているか確認
  - DHT11 と DHT22 を間違えていないか

❌ **コンパイルエラー「'dht' was not declared」**

原因と対策：

- **ライブラリ未インストール**
  - 「DHT sensor library」と「Adafruit Unified Sensor」を両方インストール
  - Arduino IDE を再起動してみる
- **#include の記述ミス**
  - #include <DHT.h>が正しく書かれているか
  - 大文字・小文字を確認

✅ 確認テスト（実践編）

以下の項目ができたらチェックしましょう

- ☐ DHT ライブラリを正しくインストールできた
- ☐ ESP32 と DHT22 を正しく配線できた
- ☐ プログラムをエラーなくコンパイル・書き込みできた
- ☐ シリアルモニタで温湿度データが表示される
- ☐ データが 2 秒ごとに更新される
- ☐ 測定値が妥当な範囲にある
- ☐ エラー処理が正しく動作する

発展問題：プログラムの改良

以下の機能を追加してみましょう（余裕があれば）：

1. 温度が 25℃を超えたら警告メッセージを表示
2. 測定データの平均値を計算して表示
3. 測定回数をカウントして表示
4. 体感温度を計算する機能を追加

追加した機能と工夫した点：

📝 振り返り

1. DHT22 センサーの特性で、最も重要だと思うことは何ですか？

2. 実習 1 の DHT11 と比較して、DHT22 で変わったことは何ですか？

3. センサーデータをデバッグする際に、シリアルモニタがどう役立ちましたか？

4. 次回（Part 3）では、このデータをクラウドに送信します。期待することや不安なことはありますか？

🎉 Part 2 完了！

ESP32 と DHT22 を使って、温湿度データを取得できるようになりました。
Part 1 の Wi-Fi 接続に、Part 2 のセンサーデータ取得が加わりました。
次の Part 3 では、いよいよこのデータを**クラウドに送信**します。

インターネットとつながるセンサーシステムが、もうすぐ完成します！

☑ 実習振り返りシート（IoT 技術応用 - 実習２ Part２）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習２　クラウド接続
Part３（クラウドサービス連携）ワークブック

📝 **実習記録**

氏名	
実習日	年　　月　　日

🎯 Part 3 の学習目標

この Part で学ぶこと

1. クラウド IoT サービスの仕組みと役割を理解する
2. Ambient/ThingSpeak の設定とチャンネル作成を習得する
3. HTTP 通信でデータを送信する方法を学ぶ
4. 送信エラーの対処とデバッグ技術を身につける

🌟 Part 3 のゴール

この Part を完了すると、**ESP32 で測定したデータがインターネットを通じてクラウドに送信される**ようになります。
スマートフォンやパソコンから、いつでもどこでもデータが確認できる状態を実現します。

📚 理論編：クラウド IoT サービスを学ぼう

1. クラウド IoT サービスとは？

クラウド IoT サービスは、IoT デバイスからのデータを**インターネット上で受信・保存・可視化**するサービスです。

主な機能

- **データ受信**：デバイスから送られるデータを受け取る
- **データ保存**：時系列データとして蓄積
- **可視化**：グラフやダッシュボードで表示
- **API 提供**：外部アプリからのデータアクセス
- **アラート**：異常値の通知（サービスによる）

2. Ambient と ThingSpeak の比較

今回の実習では、以下の 2 つのサービスから選択します：

項目	Ambient	ThingSpeak
開発元	日本（アンビエントデータ）	米国（MathWorks）
言語	日本語	英語
無料プラン	8 チャンネル、3000 データ/日	4 チャンネル、3 百万データ/年
送信間隔	30 秒以上推奨	15 秒以上
データ保存	無期限	無期限
特徴	シンプル、日本語対応	MATLAB 連携、高機能

どちらを選ぶ？

- **Ambient**：初心者向け、日本語で使いやすい
- **ThingSpeak**：発展的な分析をしたい場合

本ワークブックでは、両方の手順を説明します。

3. HTTP 通信の基礎

HTTP とは？

HTTP（HyperText Transfer Protocol）は、Web ブラウザとサーバーがデータをやり取りするための通信規約です。

HTTP リクエストの種類

- **GET**：データを取得（情報を読む）
- **POST**：データを送信（情報を書く）

今回は、センサーデータをクラウドに送るため **POST** を使います。

API キーとは？

API キーは、あなたのチャンネルを識別するための**パスワードのようなもの**です。他人に知られないよう注意してください。

4. データ送信の流れ

1. **センサーでデータ測定**：ESP32 が DHT22 から温湿度を取得
2. **データの整形**：クラウドサービスが受け取れる形式に変換

3. **HTTP POST 送信**：Wi-Fi 経由でクラウドにデータを送信
4. **サーバーで受信**：クラウドサービスがデータを保存
5. **可視化**：Web ブラウザでグラフを確認

✍️ 理解度チェック（理論編）

問 1：クラウド IoT サービスの役割

クラウド IoT サービスを使う利点を 3 つ挙げてください。

1.
2.
3.

問 2：HTTP 通信

GET と POST の違いを説明してください。

GET：

POST：

問 3：API キー

API キーを他人に教えてはいけない理由は何ですか？

🔧 実践編：クラウドサービスにデータを送信しよう

パターン A：Ambient を使う場合

1. Ambient のアカウント作成

1. https://ambidata.io/ にアクセス
2. 「ユーザー登録（無料）」をクリック
3. メールアドレスとパスワードを入力して登録
4. 確認メールが届くので、リンクをクリックして認証
5. ログインする

2. チャンネルの作成

1. ログイン後、「チャンネルを作る」をクリック
2. チャンネル名を入力（例：「教室の温湿度」）
3. 「作成」をクリック

4. 表示される**チャンネル ID** と**ライトキー**をメモ

記録してください：

- チャンネル ID：_____
- ライトキー：_____

この情報はプログラムで使用します。

3. Ambient ライブラリのインストール

1. Arduino IDE で「ライブラリを管理」を開く
2. 「Ambient ESP32 ESP8266 lib」を検索
3. インストールをクリック

4. データ送信プログラム（Ambient 版）

以下のプログラムを入力してください：

```
#include <WiFi.h>
#include <DHT.h>
#include <Ambient.h>

// Wi-Fi 設定
const char* ssid = "あなたの SSID";
const char* password = "あなたのパスワード";

// DHT22 設定
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// Ambient 設定
WiFiClient client;
Ambient ambient;
unsigned int channelId = 00000; // あなたのチャンネル ID に変更
const char* writeKey = "xxxxx"; // あなたのライトキーに変更

void setup() {
 Serial.begin(115200);
 delay(1000);
 Serial.println("=== ESP32 + DHT22 + Ambient ===");

 // センサー初期化
 dht.begin();
```

```
 Serial.println("DHT22: 初期化完了");

 // Wi-Fi 接続
 WiFi.begin(ssid, password);
 Serial.print("Wi-Fi 接続中");
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.println("¥n✓ Wi-Fi 接続成功");
 Serial.print("IP アドレス: ");
 Serial.println(WiFi.localIP());

// Ambient 初期化
 ambient.begin(channelId, writeKey, &client);
 Serial.println("✓ Ambient 初期化完了");
 Serial.println("================================¥n");
}

void loop() { // 温湿度データ取得
 float humidity = dht.readHumidity();
 float temperature = dht.readTemperature();
 // エラーチェック
 if (isnan(humidity) || isnan(temperature)) {
  Serial.println("✗ センサー読み取りエラー");
  delay(2000); return;
 }

// データ表示
 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(" %");

// Ambient にデータ送信
 ambient.set(1, temperature);
```

```
// データ1: 温度
 ambient.set(2, humidity);

// データ2: 湿度
 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");
 }
  Serial.println("------------------¥n");

 // 30秒待機（Ambientの推奨間隔）
  delay(30000);
}
```

プログラムのポイント：

- ambient.set(1, value)：チャンネルのデータ 1 に値を設定
- ambient.send()：設定したデータを Ambient に送信
- 成功すると true、失敗すると false を返す

パターン B：ThingSpeak を使う場合

1. ThingSpeak のアカウント作成

1. https://thingspeak.com/ にアクセス
2. 「Get Started For Free」をクリック
3. MathWorks アカウントを作成（または既存アカウントでログイン）
4. メール認証を完了

2. チャンネルの作成

1. 「Channels」→「My Channels」→「New Channel」をクリック
2. チャンネル情報を入力：
    - Name:「Classroom Temperature and Humidity」など
    - Field 1:「Temperature」にチェック
    - Field 2:「Humidity」にチェック
3. 「Save Channel」をクリック
4. 「API Keys」タブで **Write API Key** をメモ

記録してください：

- Channel ID：_____

- Write API Key : _____

3. データ送信プログラム（ThingSpeak 版）

以下のプログラムを入力してください：

```
#include <WiFi.h>
#include <DHT.h>
#include <HTTPClient.h>

// Wi-Fi 設定
const char* ssid = "あなたの SSID";
const char* password = "あなたのパスワード";

// DHT22 設定
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// ThingSpeak 設定
const char* serverName = "http://api.thingspeak.com/update";
String apiKey = "あなたの Write API Key"; // ここに入力
void setup() {
 Serial.begin(115200);
 delay(1000);
 Serial.println("=== ESP32 + DHT22 + ThingSpeak ===");

// センサー初期化
 dht.begin();
 Serial.println("DHT22: 初期化完了");

 // Wi-Fi 接続
 WiFi.begin(ssid, password);
 Serial.print("Wi-Fi 接続中");
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
```

```
 Serial.println("¥n✓ Wi-Fi 接続成功");
 Serial.print("IP アドレス: ");
 Serial.println(WiFi.localIP());
 Serial.println("==================================¥n");
}
void loop() { // 温湿度データ取得
 float humidity = dht.readHumidity();
 float temperature = dht.readTemperature();
 // エラーチェック
 if (isnan(humidity) || isnan(temperature)) {
  Serial.println("✗ センサー読み取りエラー");
  delay(2000);
  return;
 }

 // データ表示
  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(" %");

 // ThingSpeak にデータ送信
  if(WiFi.status() == WL_CONNECTED) {
   HTTPClient http;

// URL を構築
  String url = String(serverName) + "?api_key=" + apiKey + "&field1=" +
String(temperature) + "&field2=" + String(humidity); http.begin(url);
  int httpResponseCode = http.GET();
  if (httpResponseCode > 0) {
   Serial.print("✓ ThingSpeak への送信成功 (応答コード: ");
   Serial.print(httpResponseCode);
   Serial.println(")");
  } else {
   Serial.print("✗ ThingSpeak への送信失敗 (エラー: ");
   Serial.print(httpResponseCode);
   Serial.println(")");
```

```
  }
  http.end();
 } else {
  Serial.println("✗ Wi-Fi 未接続");
 }
 Serial.println("------------------¥n");

// 20 秒待機（ThingSpeak の推奨間隔）
 delay(20000);
}
```

プログラムのポイント：

- HTTPClient を使って GET リクエストでデータ送信
- URL に API キーとデータをパラメータとして追加
- 応答コード 200 番台なら成功

動作確認

5. プログラムを書き込んで確認

1. SSID、パスワード、API キー等を正しく設定
2. プログラムを ESP32 に書き込み
3. シリアルモニタで送信成功メッセージを確認
4. クラウドサービスの Web ページにアクセス
5. グラフにデータが表示されることを確認

🎉 成功したら

Ambient/ThingSpeak のダッシュボードで、リアルタイムにデータが更新されるグラフが見られます。これが **IoT の醍醐味**です！

⚠️ トラブルシューティング

❌ **「送信失敗」が繰り返し表示される**

原因と対策：

- **API キーの間違い**
  - チャンネル ID や API キーを再確認
  - スペースや改行が入っていないか確認
- **送信間隔が短すぎる**
  - Ambient: 30 秒以上
  - ThingSpeak: 15 秒以上
- **Wi-Fi 接続が切れている**

- シリアルモニタで Wi-Fi 状態を確認
- 電波強度(RSSI)をチェック

❌ **グラフにデータが表示されない**

原因と対策：

- **送信成功しているか確認**
  - シリアルモニタで「送信成功」が出ているか
- **ブラウザの更新**
  - ページをリロード（F5 キー）
  - 数分待ってから確認
- **チャンネル設定**
  - 正しいチャンネルを見ているか確認
  - Field 設定が正しいか（ThingSpeak）

✅ 確認テスト（実践編）

以下の項目ができたらチェックしましょう

☐ クラウドサービスのアカウントを作成できた
☐ チャンネルを作成し、API キーを取得できた
☐ プログラムに API キーを正しく設定できた
☐ シリアルモニタで「送信成功」が表示される
☐ クラウドサービスの Web ページでデータが確認できる
☐ グラフがリアルタイムで更新される
☐ スマートフォンからもデータが見られる

発展問題：システムの改良

以下の機能を追加してみましょう（余裕があれば）：

1. 送信失敗時に 3 回まで自動リトライする
2. 不快指数もクラウドに送信する（データ 3）
3. 送信成功回数をカウントして表示
4. Wi-Fi 切断時に自動再接続する

追加した機能と工夫した点：

📝 振り返り

1. データがクラウドに送信される仕組みを、自分の言葉で説明してください。

2. クラウドサービスを使うことで、どんな可能性が広がると思いますか？

3. 送信に失敗した時、どのように原因を特定しましたか？

4. 次回（Part 4）では、蓄積されたデータを分析します。どんな分析をしてみたいですか？

🎉 Part 3 完了！

センサーデータをクラウドに送信できるようになりました。
これで、**どこからでもデータにアクセスできる** IoT システムが完成しました。
スマートフォンでグラフを見ることができるのは、まさに **IoT の力**です。
次の Part 4 では、蓄積されたデータを分析し、意味を読み取る方法を学びます。
データから価値を生み出す、最終段階です！

✅ 実習振り返りシート（IoT 技術応用 - 実習２ Part３）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習２　クラウド接続
Part４（データ可視化と分析）ワークブック

📝 **実習記録**

氏名	
実習日	年　　月　　日

🎯 Part 4 の学習目標

この Part で学ぶこと

1. クラウドサービスのグラフ作成機能を使いこなす
2. 時系列データの読み取りと分析方法を習得する
3. データから傾向やパターンを見つける力を養う
4. IoT システムの改善点を考える思考力を身につける

🌟 Part 4 のゴール

実習 2 の総仕上げとして、**蓄積されたデータを分析し、そこから意味を読み取る**スキルを身につけます。データを「見る」だけでなく、「読む」「考える」ことができるようになります。

📚 理論編：データ分析の基礎を学ぼう

1. データ可視化とは？

データ可視化は、数値データを**グラフや図で視覚的に表現**することです。

可視化の目的

- **理解しやすくする**：数字の羅列より、グラフの方が直感的
- **傾向を見つける**：時間変化やパターンが一目でわかる
- **異常を発見する**：通常と異なる値が目立つ
- **意思決定を支援**：データに基づいた判断ができる

2. 時系列データの読み方

時系列データは、時間の経過とともに変化するデータです。

見るべきポイント

1. **トレンド（傾向）**
    - 全体として増加・減少・横ばいのどれか？
    - 例：気温が徐々に上がっている
2. **周期性（パターン）**
    - 規則的に繰り返す変化はあるか？
    - 例：毎日の昼と夜の温度差
3. **変動の大きさ**
    - 値が大きく変動するか、安定しているか？
    - 例：湿度の変動が大きい日
4. **異常値**
    - 明らかに他と異なる値はないか？
    - 例：センサーエラーや特殊な環境変化

3. データから考察する

考察の 4 ステップ

1. **観察**：グラフから何が読み取れるか？
2. **疑問**：なぜそうなっているのか？
3. **仮説**：原因は何だと考えられるか？
4. **検証**：仮説を確かめる方法は？

考察の例

観察：午後 2 時頃に温度が最も高くなっている
疑問：なぜ午後 2 時なのか？
仮説：太陽の位置が高く、日射量が最大になるため
検証：日照時間データと比較してみる

4. グラフの種類と使い分け

グラフの種類	用途	特徴
折れ線グラフ	時間変化を見る	トレンドがわかりやすい
棒グラフ	大小を比較	値の違いが明確

散布図	2 つの値の関係	相関関係がわかる
ヒストグラム	データの分布	頻度がわかる

今回の実習では、主に**折れ線グラフ**を使います。時間とともに変化する温湿度を見るのに最適です。

✍️ 理解度チェック（理論編）

問 1：データ可視化の目的

データをグラフ化する利点を 3 つ挙げてください。

1.
2.
3.

問 2：時系列データの読み方

時系列データから読み取るべき 4 つのポイントを説明してください。

1. トレンド：

2. 周期性：

3. 変動：

4. 異常値：

🔧 実践編：データを可視化・分析しよう

準備：データの確認

1. 蓄積されたデータを確認

1. Ambient/ThingSpeak の Web サイトにログイン
2. 自分のチャンネルを開く
3. グラフが表示されていることを確認
4. データが十分蓄積されているか確認（最低でも数時間分）

データが少ない場合は、ESP32 を数時間動作させてからこの Part に進んでください。多くのデータがあるほど、分析が意味を持ちます。

パターン A：Ambient での可視化

2. グラフの表示設定

1. チャンネルページで「チャート」タブを開く
2. 期間を選択（1 時間、6 時間、24 時間、7 日間など）
3. 表示するデータを選択（温度、湿度、または両方）
4. グラフの色や線の太さを調整（お好みで）

3. グラフの読み取り

以下の項目をグラフから読み取って記録しましょう：

温度データの分析

最高温度：_____ ℃（時刻：_____）
最低温度：_____ ℃（時刻：_____）
平均温度（目視）：約_____ ℃
温度変化の傾向：

湿度データの分析

最高湿度：_____ %（時刻：_____）
最低湿度：_____ %（時刻：_____）
平均湿度（目視）：約_____ %
湿度変化の傾向：

パターン B：ThingSpeak での可視化

2. グラフのカスタマイズ

1. チャンネルページで各 Field のグラフを確認
2. 「Private View」タブで期間を選択
3. 「Charts」から複数のグラフを組み合わせ可能
4. 「MATLAB Analysis」で高度な分析も可能（発展）

ThingSpeak では、MATLAB を使った統計分析も可能です。平均値、最大値、最小値などを自動計算できます。

共通：データ分析の実践

4. 1 日の温湿度変化を分析

24 時間のデータから、以下を分析しましょう（データがあれば）

時間帯別の特徴

時間帯	温度の特徴	湿度の特徴
朝（6-9 時）		
昼（12-15 時）		
夕方（18-21 時）		
夜（0-3 時）		

5. 温度と湿度の関係を考察

グラフを見比べて、温度と湿度の関係について考えましょう。

観察と考察

観察：温度が上がると、湿度はどうなっていますか？

仮説：なぜそのような関係になると思いますか？

検証方法：この仮説を確かめるには、どんなデータが必要ですか？

6. 異常値や特異な変化を見つける

気づいたこと

異常値はありましたか？（急激な変化、明らかにおかしい値など）

原因は何だと考えられますか？（窓の開閉、エアコン、センサーエラーなど）

規則的なパターンは見つかりましたか？（周期的な変化など）

📊 発展的な分析

1. 複数日のデータ比較

1 週間のデータがあれば、曜日ごとの傾向を分析できます。

- 平日と休日で違いはあるか？
- 天気によって室内環境は変わるか？
- 最も快適だった日はいつか？

2. 快適性の評価

Part 2 で計算した不快指数を使って、快適性を評価しましょう。

快適性の分析

最も快適だった時間帯：

不快だった時間帯：

改善するには：

3. エネルギー効率の考察

温度データから、冷暖房の使用状況を推測できます。

エネルギーに関する考察

冷暖房が使われていると思われる時間帯：

エネルギーを節約できそうな時間帯：

提案：

💡 **システムの改善提案**

7. 実習 2 のシステム全体を振り返る

改善点の検討

1. ハードウェア面（センサー、配線など）

2. ソフトウェア面（プログラム、データ送信など）

3. データ分析面（グラフ、可視化など）

4. 実用化するなら（追加したい機能、改善したい点）

発展課題：新しい機能の提案

この IoT システムに追加すると良いと思う機能を考えてください：

1. アラート機能（温度が閾値を超えたら通知）
2. 他のセンサーの追加（CO2、照度など）
3. 制御機能（自動でエアコンを ON/OFF）
4. 機械学習（異常を自動検知）

提案する機能：

その機能が必要な理由：

実現方法のアイデア：

✅ 確認テスト（実践編）

以下の項目ができたらチェックしましょう

- ☐ クラウドサービスでグラフを表示できた
- ☐ データの最大値・最小値を読み取れた
- ☐ 時間帯による変化の傾向を理解した
- ☐ 温度と湿度の関係について考察した
- ☐ 異常値やパターンを発見できた
- ☐ データに基づいて改善提案ができた
- ☐ IoT システムの価値を実感できた

📝 最終振り返り

1. 実習 2 全体（Part 1-4）を通じて、最も重要だと思った学びは何ですか？

2. データを「見る」と「分析する」の違いを、自分の言葉で説明してください。

3. この IoT システムを、どんな場面で活用できると思いますか？（3 つ以上）

4. 実習 2 で学んだことを、今後どのように活かしていきたいですか？

🎉 実習 2 完了！

実習 2 の全 4Part を完了しました。
Part 1 で Wi-Fi 接続を学び、Part 2 でセンサーデータを取得し、Part 3 でクラウドに送信し、Part 4 でデータを分析しました。
これで、**完全な IoT システム**を構築・運用・分析できる力が身につきました。

センサー → ESP32 → Wi-Fi → クラウド → 可視化 → 分析
この一連の流れが、IoT の基本です。

ここで学んだ技術は、スマート農業、スマートホーム、環境モニタリングなど、様々な実社会の応用につながります。

☑ 実習振り返りシート（IoT 技術応用 - 実習２ Part４）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習 2　クラウド接続
発展課題

📖 発展課題について

実習 2 で構築した基本 IoT システムを**さらに発展させる**ための課題集です。
難易度別に分類されており、学生の興味や習熟度に応じて選択できます。

難易度の目安：

- **初級** 実習 2 の知識で取り組める（追加学習：少）
- **中級** 一部、追加の学習が必要（新しいライブラリや概念）
- **上級** 発展的な学習が必要（複数の技術を統合）

🎯 発展課題リスト

初級 課題 1：複数センサーの追加

目標： 温湿度以外のセンサーを追加し、より多くのデータを取得する

追加できるセンサー例：

- **BME280:** 気圧センサー（温湿度も測定可能）
- **BH1750:** 照度センサー（明るさ測定）
- **MQ-2:** ガスセンサー（煙・ガス検知）
- **HC-SR04:** 超音波距離センサー

学習内容：

- I2C 通信の基礎（BME280, BH1750）
- アナログ入力の使い方（MQ-2）
- 複数センサーの統合

ヒント：

- Ambient は最大 8 チャンネルまでデータ送信可能
- 各センサーのライブラリをインストール
- ambient.set()で追加データを設定

初級 課題 2：データ送信間隔の最適化

目標： 目的に応じた適切な送信間隔を設定する

検討事項：

用途	推奨間隔	理由
環境モニタリング	5〜10 分	ゆっくり変化する
室内快適性管理	1〜2 分	リアルタイム性重視

異常検知	30 秒～1 分	早期発見が重要
省電力運用	30 分～1 時間	バッテリー寿命優先

課題：

1. 3 種類の送信間隔（30 秒、5 分、30 分）でテスト
2. 各間隔でのバッテリー消費を推定
3. 最適な間隔を決定し、その理由を説明

中級 課題 3：アラート機能の実装

目標： 閾値を超えたら通知を送る機能を追加

実装例：

- **シリアルモニタでの警告:** 最も簡単（初級レベル）
- **LED 点滅:** 視覚的な警告（初級レベル）
- **ブザー鳴動:** 聴覚的な警告（初級レベル）
- **LINE 通知:** LINE Notify API を使用（中級レベル）
- **メール送信:** SMTP ライブラリを使用（中級レベル）

実装手順（LINE 通知の場合）：

1. LINE Notify のアクセストークンを取得
2. HTTPClient ライブラリで POST リクエスト
3. 温度が閾値（例:30 度）を超えたら通知

学習内容：

- LINE Notify API の使い方
- HTTP POST リクエストの実装
- 閾値判定ロジック

中級 課題 4：省電力モードの実装

目標： バッテリー駆動を想定し、Deep Sleep を活用する

Deep Sleep とは：

- ESP32 の低消費電力モード
- 通常: 約 80mA → Deep Sleep: 約 10μA（8000 分の 1）
- 定期的に起動→測定→送信→スリープのサイクル

実装手順：

1. esp_sleep_enable_timer_wakeup()で起動時間を設定
2. センサーデータを取得・送信
3. esp_deep_sleep_start()でスリープ
4. 設定時間後に自動的に再起動

注意点：

- Deep Sleep 中は、loop 関数は実行されない
- 起動するたびに setup 関数から実行される
- Wi-Fi 再接続に時間がかかる（高速化の工夫も可能）

中級 課題 5：ローカル Web サーバーの実装

目標： ESP32 自体が Web サーバーとなり、ブラウザでデータを表示

実装内容：

- ESP32 の IP アドレスにアクセスすると、Web ページが表示される
- リアルタイムの温度・湿度が表示される
- ボタンで LED の ON/OFF ができる（制御機能）

学習内容：

- WebServer ライブラリの使い方
- HTML の基礎
- HTTP リクエストとレスポンス
- デバイス制御の実装

発展： JavaScript で自動更新、グラフ表示なども可能

上級 課題 6：データベースへの保存

目標： クラウドサービスではなく、独自のデータベースに保存

実装オプション：

1. **Raspberry Pi で MySQL サーバー構築**
    - ローカルネットワーク内で完結
    - SQL の学習に最適
2. **Google Firebase Realtime Database**
    - NoSQL データベース
    - リアルタイム同期が強力
3. **AWS IoT Core + DynamoDB**
    - 本格的なクラウド環境
    - スケーラビリティが高い

学習内容：

- データベースの基礎
- SQL/NoSQL
- データの CRUD 操作
- セキュリティ（認証・認可）

上級 課題 7：機械学習による異常検知

目標： 過去のデータから正常パターンを学習し、異常を自動検知

アプローチ：

1. **統計的手法（比較的簡単）**
    - 移動平均と標準偏差で異常判定
    - 3σ（シグマ）ルール
2. **機械学習（中程度）**
    - Isolation Forest（異常検知アルゴリズム）

    - Python で実装、結果を ESP32 にフィードバック
3. **ディープラーニング（高度）**
    - LSTM で時系列予測
    - 予測値と実測値の差で異常判定

実装の流れ：

1. クラウドから過去データをダウンロード
2. Python（Jupyter Notebook）で分析
3. モデルの訓練と評価
4. リアルタイム異常検知の実装

上級 課題 8：スマートホーム統合

目標： 温湿度に応じて家電を自動制御する

実装例：

- 温度 28 度以上 → エアコン ON
- 湿度 60%以下 → 加湿器 ON
- 照度 100 lux 以下 → 照明 ON

制御方法：

1. **赤外線リモコン信号送信**
    - IR リモコンライブラリを使用
    - エアコンやテレビを制御
2. **スマートプラグ連携**
    - Tuya Smart、TP-Link Kasa など
    - API で家電を ON/OFF
3. **リレーモジュール使用**
    - 直接電源を ON/OFF
    - 電気工事の知識が必要（危険性に注意）

学習内容：

- 家電制御の仕組み
- 赤外線通信
- スマートホーム API
- 安全な電気回路の設計

📚 発展学習のための参考リソース

🌐 Web リソース

- **ESP32 公式ドキュメント:** https://docs.espressif.com/
- **Arduino 公式リファレンス:** https://www.arduino.cc/reference/
- **Random Nerd Tutorials:** https://randomnerdtutorials.com/（ESP32 チュートリアルが豊富）
- **GitHub:** 他の人の IoT プロジェクトを参考に

🎯 発展課題の評価ポイント

評価項目	観点
技術力	新しい技術を習得し、実装できたか
創造性	独自のアイデアや改善を加えたか
完成度	システムが安定して動作するか
ドキュメント	実装内容を適切に記録したか
プレゼン	成果を効果的に伝えられるか

※ 完璧な実装よりも、**挑戦したプロセス**を重視して評価しましょう

IoT 技術応用教材
実習 3　システム統合
実習ガイド

🌟 実習 3 へようこそ

実習 3 は、**IoT 技術基礎の集大成**です。これまで学んだすべての技術を統合し、実社会で使えるレベルの IoT システムを構築します。

実習 3 で達成すること

- ✅ **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 3 の全体構成

Part	テーマ	学習内容
Part 1	Python とデータ分析の基礎	Jupyter Notebook、Pandas、データ読み込み
Part 2	IoT データの高度な分析	Matplotlib、統計分析、異常値検出、予測
Part 3	Node-RED による IoT フロー構築	ビジュアルプログラミング、MQTT、ダッシュボード
Part 4	統合 IoT システムの開発	ESP32 + Python + Node-RED の完全統合
まとめ	振り返りと今後の展望	実習 1-3 の総括、発展課題の紹介

🔄 実習 1・2・3 の関係

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

このように、**段階的に学習**してきました。実習 3 では、これまでの知識を統合します。

💻 必要な機材・ソフトウェア

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

ソフトウェア（新規インストール必要）

⚠ **事前インストール必須**

※これらのダウンロードやインストールは、かなり時間がかかります。

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

インストール手順の詳細

1. Anaconda のインストール

1. https://www.anaconda.com/download にアクセス
2. お使いの 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

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

📋 事前準備チェックリスト

実習開始前に、以下をすべてチェックしてください：

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 の学習目標

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**

📖 教材の使い方

ワークブックの活用

各 Part（1-4）に対応したワークブックがあります。

- **記録:** 実行結果、気づき、疑問点を記入
- **考察:** 「なぜ？」「どうして？」を考える
- **提出:** 実習終了後、ワークブックを提出

進め方

1. 各 Part のワークブックを読む
2. 指示に従って、実際に手を動かす
3. つまずいたら、ワークブックの「ヒント」を確認
4. それでもわからなければ、すぐに質問する
5. 完了したら、次の Part へ進む

💡 実習を成功させるコツ

1. 焦らない

Python が初めてでも大丈夫。一歩ずつ進めば、必ずできます。

2. エラーを恐れない

エラーは学習の機会です。エラーメッセージをよく読んで、原因を探りましょう。

3. すぐに質問する

わからないことは、すぐに教員やクラスメートに質問してください。一人で悩まないことが大切です。

4. 試行錯誤を楽しむ

ワークブック通りだけでなく、「これを変えたらどうなる？」と実験してみてください。

5. 記録を残す

スクリーンショット、メモを取りながら進めると、後で振り返りやすくなります。

🚀 実社会への応用

実習 3 で学ぶ技術は、実際のビジネスで広く使われています。

応用例

- **スマート農業:** 温室の環境を自動管理
- **工場の予知保全:** 機械の異常を事前に検知
- **エネルギー管理:** 電力使用量を分析し、省エネ提案
- **医療・ヘルスケア:** 患者のバイタルサインを遠隔監視
- **環境モニタリング:** 大気質、水質を常時測定

皆さんが学ぶ技術は、**社会の役に立つ技術**です。

📝 評価について

実習 3 の評価は、以下の要素で行われます：

- 各 Part の達成度（システムが動作するか）

- ワークブックの記入内容（記録、考察）
- 最終レポート（学んだこと、今後の展望）
- 発展課題への挑戦（ボーナス点）

🎓 最後に

実習 3 は、IoT 技術基礎の総仕上げです。これまで学んだ Arduino、センサー、ネットワーク、クラウドのすべてが、ここで 1 つにつながります。

新しいツール（Python と Node-RED）を学びますが心配は不要です。どちらも初心者に優しく設計されています。

完成した時、皆さんは**「本格的な IoT システムを自分で作れる」**と自信を持って言えるようになっているはずです。

それでは、実習 3 を始めましょう！

IoT 技術応用教材
実習３　システム統合
Part１（Python とデータ分析の基礎）ワークブック

📝 **実習記録**

氏名	
実習日	年　　月　　日

🎯 Part 1 の学習目標

- Jupyter Notebook の基本操作を習得する
- Pandas を使って CSV データを読み込めるようになる
- データの基本統計量を理解する
- Python によるデータ分析の第一歩を踏み出す

ステップ 1：Jupyter Notebook の起動

1-1. Anaconda Navigator を開く

📌 **タスク：** Anaconda Navigator を起動し、Jupyter Notebook を開く

1. Anaconda Navigator を起動
2. 「Jupyter Notebook」の「Launch」ボタンをクリック
3. ブラウザが自動的に開き、ファイル一覧が表示される

✅ **チェックポイント 1：Jupyter Notebook が開きましたか？**

☐ ブラウザで Jupyter Notebook のホーム画面が表示されている
☐ ファイルやフォルダの一覧が見える

1-2. 新しいノートブックを作成

📌 **タスク：** 新規ノートブックを作成する

1. 右上の「New」をクリック
2. 「Python 3」を選択
3. 新しいタブが開き、ノートブックが表示される
4. ノートブックの名前を「実習 3_Part1」に変更（上部の「Untitled」をクリック）

1-3. Jupyter Notebook の基本操作

💡 **Jupyter Notebook の基本：**

- **セル:** コードを書く四角い枠
- **実行:** Shift + Enter でセルを実行
- **追加:** 「+」ボタンで新しいセルを追加
- **削除:** セルを選択して「はさみ」ボタン

📌 **タスク：** 最初の Python コードを実行

セルに以下を入力して、**Shift + Enter** を押してください：

print("Hello, Python!") print("実習 3 を始めます！")

実行結果を記録：

✅ チェックポイント 2：コードが実行できましたか？

☐ 「Hello, Python!」と表示された

☐ セルの左側に「In [1]:」のような番号が表示された

ステップ 2：Pandas のインポートとデータ読み込み

2-1. Pandas をインポート

📌 **タスク：** Pandas ライブラリをインポートする

新しいセルに以下を入力して実行：

import pandas as pd print("Pandas のインポート成功！")

💡 Pandas とは？

Pandas は、Python でデータ分析を行うための最も人気のあるライブラリです。

- 表形式のデータ（Excel や CSV のような）を簡単に扱える
- **pd** という略称で使うのが一般的
- データサイエンティストの必須ツール

2-2. CSV ファイルの準備

📌 **タスク：** 実習 2 で蓄積したデータをダウンロード

1. Ambient（または ThingSpeak）にアクセス
2. 自分のチャンネルを開く
3. 「データのエクスポート」→「CSV」を選択
4. ダウンロードした CSV ファイルを、デスクトップなど分かりやすい場所に保存
5. ファイル名を確認（例: ambient_data.csv）

ダウンロードしたファイルの情報を記録：

- ファイル名： _____
- 保存場所： _____

- ファイルサイズ： _____ KB

2-3. CSV ファイルを読み込む

📌 **タスク：** Pandas で CSV を読み込む
新しいセルに以下を入力（**ファイル名は自分のものに変更**）：
CSV ファイルを読み込む df = pd.read_csv('ambient_data.csv') print("データの読み込み成功！") print(f"データ数: {len(df)}行")

⚠️ **エラーが出た場合：**

FileNotFoundError が出た場合は、ファイルのパスが間違っています。
対処法 1: フルパスで指定
Windows の場合（例） df = pd.read_csv(r'C:¥Users¥あなたの名前¥Desktop¥ambient_data.csv') # Mac の場合（例） df = pd.read_csv('/Users/あなたの名前/Desktop/ambient_data.csv')
対処法 2: CSV ファイルを Jupyter Notebook と同じフォルダに移動
読み込んだデータの行数を記録： _____行

✅ **チェックポイント 3：データが読み込めましたか？**

☐ エラーなく実行できた
☐ 「データの読み込み成功！」と表示された
☐ データ数が表示された

ステップ 3：データの確認

3-1. データの最初の 5 行を表示

📌 **タスク：** データの先頭部分を確認する
最初の 5 行を表示 df.head()
表示されたデータについて記録：

- どんな列（カラム）がありますか？ _____
- 日時のデータは入っていますか？ ☐ はい / ☐ いいえ
- 温度・湿度のデータは入っていますか？ ☐ はい / ☐ いいえ

3-2. データの詳細情報を確認

📌 **タスク：** データの構造を確認する
データの詳細情報 df.info()
表示された情報を記録：

項目	値
総行数	_____行

列数	_____列
欠損値の有無	□ あり / □ なし

3-3. 列名を確認

📌 **タスク：** すべての列名を表示する

列名を表示 print(df.columns.tolist())

列名を記録：

💡 Ambient のデータ形式：

Ambient では、通常以下のような列名になっています：

- **created:** データが記録された日時
- **d1:** データ１（通常は温度）
- **d2:** データ２（通常は湿度）

ステップ 4：基本統計量の計算

4-1. describe()で統計量を一括表示

📌 タスク：データの基本統計量を計算する

基本統計量を表示 df.describe()

温度（d1）の統計量を記録：

統計量	値
平均（mean）	_____℃
標準偏差（std）	_____
最小値（min）	_____℃
最大値（max）	_____℃

湿度（d2）の統計量を記録：

統計量	値
平均（mean）	_____%

標準偏差（std）	_____
最小値（min）	_____%
最大値（max）	_____%

💡 **統計量の意味：**

- **平均（mean）：**すべてのデータの平均値
- **標準偏差（std）：**データのバラつき具合（大きいとバラバラ、小さいと安定）
- **最小値・最大値:** データの範囲
- **25%、50%、75%:** 四分位数（データを 4 等分した時の境界値）

4-2. 個別の統計量を計算

📌 タスク： 温度の統計を個別に計算する

```
# 温度（d1）の統計
print(f"平均温度: {df['d1'].mean():.2f}°C")
print(f"中央値: {df['d1'].median():.2f}°C")
print(f"最高温度: {df['d1'].max():.2f}°C")
print(f"最低温度: {df['d1'].min():.2f}°C")
print(f"温度の範囲: {df['d1'].max() - df['d1'].min():.2f}°C")
```

結果を記録：

ステップ 5：データの選択と抽出

5-1. 特定の列だけを表示

📌 タスク： 温度と湿度だけを表示する

```
# 温度と湿度だけを表示 df[['d1', 'd2']].head()
```

5-2. 条件でデータを抽出

📌 タスク： 温度が 25 度以上のデータを抽出する

```
# 温度が 25 度以上のデータ high_temp = df[df['d1'] >= 25] print(f"25 度以上のデータ:
{len(high_temp)}行") # 最初の 5 行を表示 high_temp.head()
```

25 度以上のデータは何行ありましたか？ _____行

5-3. データのソート（並び替え）

📌 タスク： 温度が高い順に並び替える

温度が高い順に並び替え df_sorted = df.sort_values('d1', ascending=False) # 上位 5 件を表示 print("最も暑かった時のデータ:") df_sorted.head()

最高温度は何度でしたか？ _____°C

その時の湿度は？ _____%

Part 1 まとめと振り返り

学んだこと

1. Jupyter Notebook の基本操作（セルの実行、追加、削除）
2. Pandas を使った CSV データの読み込み
3. データの確認方法（head()、info()、describe()）
4. 基本統計量の意味と計算方法
5. データの選択と抽出

考察：Python とデータ分析

問 1： Python と Excel を比べて、どちらが使いやすいと思いましたか？それはなぜですか？

問 2： データの統計量（平均、標準偏差など）を見て、何か発見はありましたか？

次のステップ

Part 2 では、このデータを**グラフで可視化**します。Matplotlib を使って、時系列グラフ、散布図、ヒストグラムなどを作成し、データのパターンを視覚的に理解します。

✅ Part 1 完了チェック

☐ Jupyter Notebook を起動できた
☐ Pandas をインポートできた
☐ CSV ファイルを読み込めた
☐ df.head()でデータを表示できた
☐ df.describe()で統計量を確認できた
☐ データの抽出・ソートができた

💾 **ノートブックの保存を忘れずに！**

「File」→「Save and Checkpoint」、または Ctrl+S（Mac: Cmd+S）

☑ 実習振り返りシート（IoT 技術応用 - 実習３ Part１）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習３　システム統合
Part２（IoT データの高度な分析）ワークブック

📝 **実習記録**

氏名	
実習日	年　　月　　日

🎯 Part 2 の学習目標

- Matplotlib を使ってデータをグラフ化できるようになる
- 時系列データの統計分析（平均、標準偏差、相関）を理解する
- 異常値を検出する方法を学ぶ
- 簡単な予測モデル（線形回帰）を作成する

ステップ 1：Matplotlib による可視化

1-1. Matplotlib のインポート

📌 タスク： 必要なライブラリをインポート

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
# 日本語フォント設定（文字化け防止）
plt.rcParams['font.sans-serif'] = ['MS Gothic', 'Yu Gothic', 'Hiragino
Sans']
plt.rcParams['axes.unicode_minus'] = False
# Jupyter Notebook 内にグラフを表示
%matplotlib inline
print("ライブラリのインポート完了")
```

1-2. 温度の時系列グラフ

📌 タスク： 温度データを折れ線グラフで表示

```
# Part 1 で読み込んだデータを使用
# df_iot: 実習 2 のデータ
plt.figure(figsize=(12, 5))
plt.plot(df_iot.index, df_iot['d1'], color='red', linewidth=1.5)
plt.title('温度の時系列変化', fontsize=16)
plt.xlabel('日時', fontsize=12)
plt.ylabel('温度 (°C)', fontsize=12)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

グラフから読み取れることを記録：

1-3. 温度と湿度の 2 軸グラフ

📌 タスク： 温度と湿度を同時に表示

```
fig, ax1 = plt.subplots(figsize=(12, 5))
# 温度（左軸）
ax1.plot(df_iot.index, df_iot['d1'], color='red', label='温度')
ax1.set_xlabel('日時', fontsize=12)
ax1.set_ylabel('温度 (°C)', fontsize=12, color='red')
ax1.tick_params(axis='y', labelcolor='red')
# 湿度（右軸）
ax2 = ax1.twinx()
ax2.plot(df_iot.index, df_iot['d2'], color='blue', label='湿度')
ax2.set_ylabel('湿度 (%)', fontsize=12, color='blue')
ax2.tick_params(axis='y', labelcolor='blue')
plt.title('温度と湿度の時系列変化', fontsize=16)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

温度と湿度の関係について気づいたことを記録：

1-4. ヒストグラム（分布の確認）

📌 タスク： 温度の分布を確認

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.hist(df_iot['d1'], bins=30, color='red', alpha=0.7, edgecolor='black')
plt.title('温度の分布', fontsize=14)
plt.xlabel('温度 (°C)', fontsize=12)
plt.ylabel('頻度', fontsize=12)
plt.grid(True, alpha=0.3)
plt.subplot(1, 2, 2)
plt.hist(df_iot['d2'], bins=30, color='blue', alpha=0.7, edgecolor='black')
plt.title('湿度の分布', fontsize=14)
plt.xlabel('湿度 (%)', fontsize=12)
plt.ylabel('頻度', fontsize=12)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

分布の特徴を記録：

- 温度は何度付近に集中していますか？ _____℃
- 湿度は何%付近に集中していますか？ _____%

ステップ 2：統計分析

2-1. 基本統計量の詳細分析

📌 タスク： 詳細な統計量を計算

```
# 基本統計量 print("=== 温度の統計 ===")
print(f"平均: {df_iot['d1'].mean():.2f}°C")
print(f"中央値: {df_iot['d1'].median():.2f}°C")
print(f"標準偏差: {df_iot['d1'].std():.2f}°C")
print(f"最大値: {df_iot['d1'].max():.2f}°C")
print(f"最小値: {df_iot['d1'].min():.2f}°C")
print(f"範囲: {df_iot['d1'].max() - df_iot['d1'].min():.2f}°C")
print("¥n=== 湿度の統計 ===")
```

```
print(f"平均: {df_iot['d2'].mean():.2f}%")
print(f"中央値: {df_iot['d2'].median():.2f}%")
print(f"標準偏差: {df_iot['d2'].std():.2f}%")
print(f"最大値: {df_iot['d2'].max():.2f}%")
print(f"最小値: {df_iot['d2'].min():.2f}%")
```

結果を記録：

統計量	温度	湿度
平均	_____°C	_____%
中央値	_____°C	_____%
標準偏差	_____°C	_____%

2-2. 相関分析

📌 タスク： 温度と湿度の相関を分析

相関係数の計算

```
correlation = df_iot['d1'].corr(df_iot['d2'])
print(f"温度と湿度の相関係数: {correlation:.3f}")
# 散布図
plt.figure(figsize=(8, 6))
plt.scatter(df_iot['d1'], df_iot['d2'], alpha=0.5)
plt.xlabel('温度 (°C)', fontsize=12)
plt.ylabel('湿度 (%)', fontsize=12)
plt.title(f'温度と湿度の相関 (相関係数: {correlation:.3f})', fontsize=14)
plt.grid(True, alpha=0.3)
plt.tight_layout() plt.show()
```

💡 **相関係数の解釈：**

- 1 に近い：強い正の相関（一方が増えると他方も増える）
- 0 に近い：相関なし
- -1 に近い：強い負の相関（一方が増えると他方は減る）

相関係数： _____

この値から何が言えますか？

2-3. 時間帯別の分析

📌 タスク： 時刻ごとの平均温度を計算

```
# 時刻を抽出
df_iot['hour'] = df_iot.index.hour
# 時刻別の平均温度
hourly_temp = df_iot.groupby('hour')['d1'].mean()
plt.figure(figsize=(12, 5))
plt.plot(hourly_temp.index, hourly_temp.values, marker='o', linewidth=2,
markersize=8)
plt.xlabel('時刻', fontsize=12)
plt.ylabel('平均温度 (°C)', fontsize=12)
plt.title('時刻別の平均温度', fontsize=14)
plt.grid(True, alpha=0.3)
plt.xticks(range(0, 24))
plt.tight_layout()
plt.show()
print(f"最も暑い時刻: {hourly_temp.idxmax()}時 ({hourly_temp.max():.2f}°C)")
print(f"最も涼しい時刻: {hourly_temp.idxmin()}時 ({hourly_temp.min():.2f}°C)")
```

結果を記録：

- 最も暑い時刻： _____時（_____℃）
- 最も涼しい時刻： _____時（_____℃）

ステップ 3：異常値検出

3-1. 3σ 法による異常値検出

📌 タスク： 統計的に異常な値を検出

平均と標準偏差

```
mean_temp = df_iot['d1'].mean()
std_temp = df_iot['d1'].std()
# 異常値の閾値（平均±3σ）
upper_limit = mean_temp + 3 * std_temp
lower_limit = mean_temp - 3 * std_temp
print(f"温度の正常範囲: {lower_limit:.2f}°C ～ {upper_limit:.2f}°C")
# 異常値の検出
outliers = df_iot[(df_iot['d1'] > upper_limit) | (df_iot['d1'] <
```

```
lower_limit)]
print(f"¥n異常値の数: {len(outliers)}件") if len(outliers) > 0: print("¥n異常値:")
print(outliers[['d1', 'd2']])
```

異常値検出結果を記録：

- 正常範囲： _____℃ ～ _____℃
- 異常値の数： _____件

3-2. 異常値の可視化

📌 タスク： 異常値をグラフで強調表示

```
plt.figure(figsize=(12, 5))
# 正常値
normal = df_iot[(df_iot['d1'] <= upper_limit) & (df_iot['d1'] >= lower_limit)]
plt.plot(normal.index, normal['d1'], color='blue', alpha=0.7, label='正常値')
# 異常値
if len(outliers) > 0: plt.scatter(outliers.index, outliers['d1'], color='red', s=100, label='異常値', zorder=5)
# 閾値線
plt.axhline(y=upper_limit, color='orange', linestyle='--', label=f'上限({upper_limit:.1f}°C)')
plt.axhline(y=lower_limit, color='orange', linestyle='--', label=f'下限({lower_limit:.1f}°C)')
plt.xlabel('日時', fontsize=12)
plt.ylabel('温度 (°C)', fontsize=12)
plt.title('異常値検出結果', fontsize=14)
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

✅ チェックポイント：異常値は見つかりましたか？

☐ 異常値が検出された / ☐ 異常値は検出されなかった

異常値がある場合、その原因は何だと思いますか？

ステップ 4：簡単な予測モデル

4-1. 線形回帰による温度予測

📌 タスク： 過去のデータから未来を予測

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# データの準備（時刻から温度を予測）
X = df_iot['hour'].values.reshape(-1, 1)
# 説明変数（時刻）
y = df_iot['d1'].values
# 目的変数（温度）
# モデルの訓練
model = LinearRegression() model.fit(X, y)
# 予測
y_pred = model.predict(X)
# 評価
r2 = r2_score(y, y_pred)
rmse = np.sqrt(mean_squared_error(y, y_pred))
print(f"決定係数 (R²): {r2:.3f}")
print(f"RMSE: {rmse:.2f}°C")
print(f"¥n 予測式: 温度 = {model.coef_[0]:.3f} × 時刻 + {model.intercept_:.3f}")
```

予測モデルの性能を記録：

- 決定係数 (R²)： _____
- RMSE： _____°C

4-2. 予測結果の可視化

📌 タスク： 実測値と予測値を比較

```
plt.figure(figsize=(12, 5))
# 時刻別の平均（実測値）
plt.plot(hourly_temp.index, hourly_temp.values, marker='o', label='実測値（平均）', linewidth=2)
# 予測値
X_pred = np.arange(0, 24).reshape(-1, 1)
y_pred_hourly = model.predict(X_pred)
plt.plot(X_pred, y_pred_hourly, linestyle='--', label='予測値', linewidth=2)
plt.xlabel('時刻', fontsize=12)
plt.ylabel('温度 (°C)', fontsize=12)
plt.title(f'温度予測モデル (R²={r2:.3f})', fontsize=14)
plt.legend()
plt.grid(True, alpha=0.3)
plt.xticks(range(0, 24))
plt.tight_layout()
plt.show()
```

💡 **決定係数（R²）の解釈：**

- 1 に近い：予測精度が高い
- 0.7 以上：まあまあ良い予測
- 0.5 以下：予測精度が低い

Part 2 まとめと振り返り

学んだこと

1. Matplotlib による多様なグラフ作成
2. 統計分析（平均、標準偏差、相関）
3. 異常値検出（3σ 法）
4. 線形回帰による予測モデル

考察：データ分析の重要性

問： データを分析することで、どんな新しい発見がありましたか？

次のステップ

Part 3 では、**Node-RED** を使って、データを**リアルタイムで処理・表示**するシステムを構築します。
ビジュアルプログラミングで、IoT のデータフローを設計します！

✅ Part 2 完了チェック

- ☐ Matplotlib でグラフを作成できた
- ☐ 統計分析の結果を解釈できた
- ☐ 異常値を検出できた
- ☐ 予測モデルを作成できた

☑ 実習振り返りシート（IoT 技術応用 - 実習３ Part２）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習３　システム統合
Part３（Node-RED による IoT フロー構築）ワークブック

📝 **実習記録**

氏名	
実習日	年　　月　　日

🎯 Part 3 の学習目標

- Node-RED の基本操作を習得する
- MQTT プロトコルによるデータ通信を理解する
- データ受信→処理→表示のフローを構築する
- ダッシュボードで視覚的にデータを表示する

ステップ 1：Node-RED の起動と基本操作

1-1. Node-RED の起動

📌 タスク：

1. ターミナル（コマンドプロンプト）を開く
2. 以下のコマンドを実行：
   node-red
3. 起動メッセージに表示される URL を確認
4. ブラウザで http://localhost:1880 を開く

✅ チェックポイント 1：Node-RED が起動しましたか？

☐ ブラウザで Node-RED エディタが表示されている
☐ 左側にノードのパレットが表示されている
☐ 中央にフローエディタ（ワークスペース）がある

1-2. Node-RED の画面構成

💡 Node-RED の 3 つのエリア：

エリア	役割
左：パレット	使用可能なノード（部品）の一覧
中央：ワークスペース	ノードを配置してフローを作成
右：情報/デバッグ	ノードの説明、デバッグ出力を表示

1-3. 最初のフロー作成（Hello World）

📌 タスク： シンプルなフローを作成

1. パレットから「inject」ノードをワークスペースにドラッグ
2. 「debug」ノードをワークスペースにドラッグ
3. inject ノードの右端と debug ノードの左端をつなぐ
4. inject ノードをダブルクリックして設定画面を開く
5. 「msg.payload」を「文字列」に変更し、「Hello, IoT!」と入力
6. 「完了」をクリック
7. 右上の「デプロイ」ボタンをクリック
8. inject ノードの左のボタンをクリック
9. 右側のデバッグタブで出力を確認

デバッグ出力に表示されたメッセージを記録：

💡 Node-RED の基本用語：

- **ノード:** 処理の単位（Arduino のブロックのようなもの）
- **フロー:** ノードをつないだデータの流れ
- **msg.payload:** ノード間で受け渡されるデータ
- **デプロイ:** フローを実行可能にする（保存＋実行）

ステップ 2：MQTT によるデータ受信

2-1. MQTT とは

💡 MQTT（Message Queuing Telemetry Transport）

IoT に最適な軽量な通信プロトコルです。

- **Publisher（送信側）：**データを送信するデバイス
- **Broker（仲介）：**データを中継するサーバー
- **Subscriber（受信側）：**データを受信するアプリ
- **Topic:** データの種類を識別する名前（例: "sensor/temperature"）

2-2. 公開 MQTT ブローカーの利用

📌 タスク： テスト用のブローカーに接続

使用する公開ブローカー:

- サーバー: test.mosquitto.org
- ポート: 1883

⚠️ **注意：**

公開ブローカーは誰でもアクセスできます。個人情報や API キーは送信しないでください。

2-3. MQTT 受信フローの作成

📌 タスク： MQTT でデータを受信するフローを作成

1. パレットから「mqtt in」ノードをドラッグ
2. 「debug」ノードをドラッグしてつなぐ
3. mqtt in ノードをダブルクリック
4. 「サーバー」の鉛筆アイコンをクリックして新規追加
5. サーバー: test.mosquitto.org、ポート: 1883 を入力
6. 「追加」をクリック
7. トピック: iot/test/temperature を入力
8. 「完了」をクリック
9. デプロイ

2-4. MQTT 送信フローの作成（テスト用）

📌 タスク： 自分でデータを送信してテスト

1. 「inject」ノードをドラッグ
2. 「mqtt out」ノードをドラッグしてつなぐ
3. inject ノードを設定:
    - msg.payload: 数値
    - 値: 25.5

4. mqtt out ノードを設定:
    - サーバー: 先ほどと同じ
    - トピック: iot/test/temperature
5. デプロイ
6. inject ノードをクリックして送信
7. デバッグタブで受信を確認

送受信が成功しましたか？

✅ チェックポイント 2：MQTT で送受信できましたか？

☐ mqtt in ノードが「接続済み」になっている
☐ データを送信すると、デバッグタブに表示される

ステップ 3：データ処理フローの構築

3-1. function ノードによるデータ加工

📌 タスク： 温度データを華氏に変換

1. mqtt in ノードと debug ノードの間に「function」ノードを挿入
2. function ノードをダブルクリック
3. 以下のコードを入力:

```
// 摂氏から華氏に変換
var celsius = msg.payload;
var fahrenheit = celsius * 9/5 + 32;
msg.payload = { "celsius": celsius, "fahrenheit": fahrenheit.toFixed(1) };
return msg;
```

4. デプロイして動作確認

変換結果を記録：

- 入力（摂氏）： _____°C
- 出力（華氏）： _____°F

3-2. switch ノードによる条件分岐

📌 タスク： 温度が 25 度以上の時だけ処理

1. mqtt in ノードの後に「switch」ノードを追加
2. switch ノードを設定:
    - プロパティ: msg.payload
    - 条件 1: >= 数値 25
    - 条件 2: < 数値 25

3. 出力 1 に「debug」ノード（名前：高温）
4. 出力 2 に「debug」ノード（名前：通常）
5. デプロイして、様々な温度でテスト

3-3. 移動平均の計算

📌 タスク： 過去 5 個のデータの平均を計算

1. function ノードを追加
2. 以下のコードを入力：

```
// グローバル変数に過去のデータを保存
var history = global.get('tempHistory') || [];
// 新しいデータを追加
history.push(msg.payload);
// 5 個以上になったら古いデータを削除
if (history.length > 5) {
 history.shift();
}
// 平均を計算
var sum = history.reduce((a, b) => a + b, 0);
var average = sum / history.length;
// グローバル変数を更新
global.set('tempHistory', history);
msg.payload = { "current": msg.payload, "average": average.toFixed(2),
"count": history.length };
return msg;
```

3. デプロイして動作確認

ステップ 4：ダッシュボードの作成

4-1. Dashboard 拡張のインストール

📌 タスク： node-red-dashboard をインストール

1. 右上のメニュー → 「パレットの管理」をクリック
2. 「ノードを追加」タブを選択
3. 検索欄に「node-red-dashboard」と入力
4. 「ノードを追加」ボタンをクリック
5. インストール完了まで待機（1〜2 分）
6. パレットに「dashboard」カテゴリが追加されたことを確認

✅ チェックポイント 3：Dashboard がインストールされましたか？

☐ パレットに「dashboard」グループが表示されている
☐ gauge、chart、text などのノードがある

4-2. 温度ゲージの作成

📌 タスク： 温度を針メーターで表示

1. 「mqtt in」ノードからデータを受信
2. 「gauge」ノードを追加してつなぐ
3. gauge ノードを設定:
    - グループ: 新規作成「温度モニター」
    - ラベル: 温度
    - 単位: °C
    - 範囲: 0 ～ 40
    - 色の範囲:
        - 0-20: 青
        - 20-30: 緑
        - 30-40: 赤
4. デプロイ
5. ブラウザで http://localhost:1880/ui を開く

ダッシュボードが表示されましたか？7

4-3. グラフの追加

📌 タスク： 温度の時系列グラフを表示

1. 「chart」ノードを追加
2. chart ノードを設定:
    - グループ: 温度モニター
    - ラベル: 温度推移
    - X 軸: 最近 30 分
    - Y 軸: 0 ～ 40
3. mqtt in ノードから chart ノードにもつなぐ
4. デプロイして確認

4-4. テキスト表示

📌 タスク： 現在の温度を数値で表示

1. 「text」ノードを追加
2. 設定:

- グループ：温度モニター
- ラベル：現在の温度
- 値のフォーマット：{{msg.payload}} °C

3. デプロイ

✅ チェックポイント 4：ダッシュボードが完成しましたか？

☐ ゲージが表示されている
☐ グラフが表示されている
☐ テキストで温度が表示されている
☐ データを送信すると、すべてが更新される

Part 3 まとめと振り返り

学んだこと

1. Node-RED の基本操作（ノード、フロー、デプロイ）
2. MQTT によるデータ通信
3. データ処理フロー（変換、条件分岐、移動平均）
4. ダッシュボードによる可視化

考察：Node-RED の利点

問： Node-RED と Arduino IDE のプログラミングの違いは何ですか？どちらが使いやすいと思いましたか？

次のステップ

Part 4 では、**ESP32、Python、Node-RED を統合**して、完全な IoT システムを構築します。
これまで学んだすべての技術を組み合わせます！

✅ Part 3 完了チェック

☐ Node-RED でフローを作成できた
☐ MQTT で送受信できた
☐ ダッシュボードを作成できた

☑ 実習振り返りシート（IoT 技術応用 - 実習３ Part３）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習３　システム統合
Part４（統合 IoT システムの開発）ワークブック

実習記録

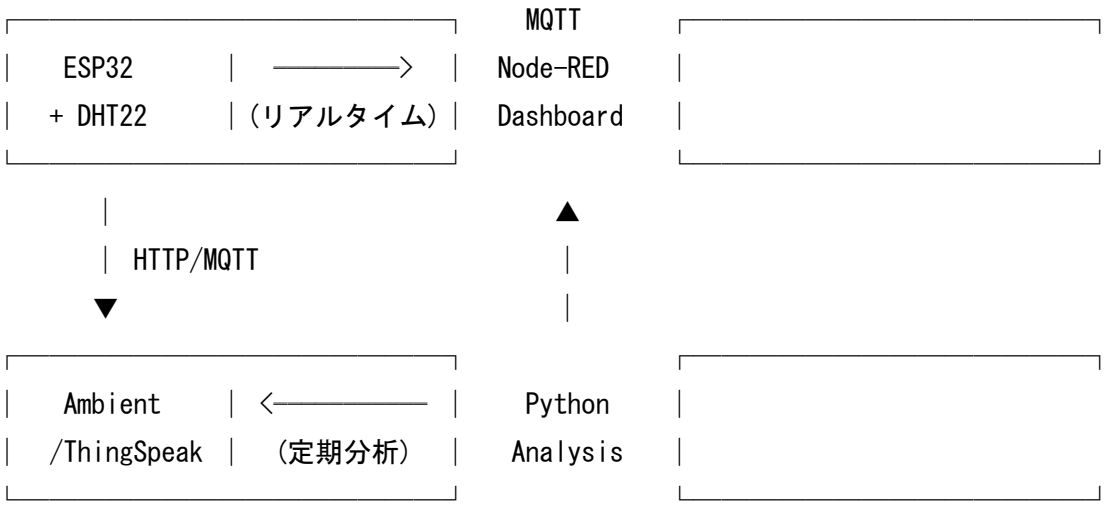
氏名	
実習日	年　　月　　日

Part ４の学習目標

- ESP32 から MQTT でデータを送信する
- Node-RED でリアルタイムにデータを受信・表示する
- Python で蓄積データを分析し、結果を Node-RED に送る
- 自動アラート機能を実装する
- 完全な IoT システムを構築・運用する

システム全体像

統合 IoT システムの構成



データの流れ：

1. ESP32 がセンサーデータを取得
2. MQTT で Node-RED に送信（リアルタイム表示）
3. 同時に Ambient/ThingSpeak にも保存（実習 2 の継続）
4. Python が定期的にクラウドデータを分析

5. 分析結果を MQTT で Node-RED に送信
6. Node-RED が異常時にアラート

ステップ 1：ESP32 の MQTT 対応プログラム

1-1. MQTT ライブラリのインストール

📌 タスク： Arduino IDE でライブラリをインストール

1. 「スケッチ」→「ライブラリをインクルード」→「ライブラリを管理」
2. 検索欄に「PubSubClient」と入力
3. 「PubSubClient by Nick O'Leary」をインストール

1-2. ESP32 プログラムの作成

📌 タスク： 以下のプログラムを作成・書き込み

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <Ambient.h>
// Wi-Fi 設定
const char* ssid = "あなたの SSID";
const char* password = "あなたのパスワード";
// MQTT 設定
const char* mqtt_server = "test.mosquitto.org";
const int mqtt_port = 1883;
const char* mqtt_topic_temp = "iot/esp32/temperature";
const char* mqtt_topic_hum = "iot/esp32/humidity";
// DHT22 設定
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
// Ambient 設定（実習 2 から継続）
unsigned int channelId = 12345;
// 自分のチャンネル
ID const char* writeKey = "あなたのライトキー";
WiFiClient espClient;
PubSubClient mqttClient(espClient);
Ambient ambient;
void setup() {
 Serial.begin(115200);
```

```
 dht.begin();
// Wi-Fi 接続
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
  }
  Serial.println("¥nWi-Fi 接続成功");
  // MQTT 設定
  mqttClient.setServer(mqtt_server, mqtt_port);
  // Ambient 初期化
  ambient.begin(channelId, writeKey, &espClient);
}
void reconnectMQTT() {
 while (!mqttClient.connected()) {
  Serial.print("MQTT 接続中...");
  String clientId = "ESP32-" + String(random(0xffff), HEX);
  if (mqttClient.connect(clientId.c_str())) {
   Serial.println("接続成功");
  } else {
   Serial.print("失敗 rc=");
   Serial.println(mqttClient.state());
   delay(5000);
  }
 }
}
void loop() {
 if (!mqttClient.connected()) {
   reconnectMQTT();
  }
  mqttClient.loop();
  // センサーデータ取得
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  if (isnan(temperature) || isnan(humidity)) {
   Serial.println("センサー読み取りエラー");
   return;
  }
  // シリアル出力
  Serial.printf("温度: %.1f°C, 湿度: %.1f%%¥n", temperature, humidity);
```

```
  // MQTT で送信
  char tempStr[10], humStr[10];
  dtostrf(temperature, 4, 1, tempStr);
  dtostrf(humidity, 4, 1, humStr);
  mqttClient.publish(mqtt_topic_temp, tempStr);
  mqttClient.publish(mqtt_topic_hum, humStr);
  Serial.println("MQTT 送信完了");
  // Ambient にも送信（実習 2 の継続）
  ambient.set(1, temperature);
  ambient.set(2, humidity);
  ambient.send();
  Serial.println("Ambient 送信完了");
  delay(10000); // 10 秒ごと
}
```

✅ チェックポイント 1：ESP32 からデータが送信されていますか？

☐ シリアルモニタに「MQTT 送信完了」と表示される
☐ Node-RED のデバッグタブでデータを確認できる
☐ Ambient にもデータが届いている

ステップ 2：Node-RED でのリアルタイム表示

2-1. ESP32 データ受信フローの作成

📌 タスク： ESP32 からのデータを受信

1. 「mqtt in」ノードを 2 つ配置
2. 1 つ目の設定:
    - トピック: iot/esp32/temperature
    - 名前: ESP32 温度
3. 2 つ目の設定:
    - トピック: iot/esp32/humidity
    - 名前: ESP32 湿度
4. それぞれを debug ノードにつないで動作確認

2-2. ダッシュボードの更新

📌 タスク： ESP32 データをダッシュボードに表示

1. 温度用:
    - mqtt in (temperature) → gauge
    - mqtt in (temperature) → chart
    - mqtt in (temperature) → text
2. 湿度用:
    - mqtt in (humidity) → gauge
    - mqtt in (humidity) → chart
    - mqtt in (humidity) → text
3. デプロイして http://localhost:1880/ui で確認

ダッシュボードが正しく更新されていますか？

ステップ 3：Python からの分析結果送信

3-1. Python MQTT ライブラリのインストール

📌 タスク： Jupyter Notebook で実行

```
!pip install paho-mqtt
```

3-2. 分析結果の MQTT 送信プログラム

📌 タスク： Jupyter Notebook で実行

```
import paho.mqtt.client as mqtt
import pandas as pd import time
# MQTT 設定
broker = "test.mosquitto.org"
port = 1883
topic_analysis = "iot/analysis/result"
# MQTT クライアントの初期化
client = mqtt.Client()
client.connect(broker, port)
# Part 2 で分析したデータを使用
# 例：温度の平均、最大、最小
avg_temp = df_iot['d1'].mean()
max_temp = df_iot['d1'].max()
min_temp = df_iot['d1'].min()
# 分析結果を JSON 形式で送信
result = { "average": round(avg_temp, 2), "maximum": round(max_temp, 2),
"minimum": round(min_temp, 2), "timestamp":
```

```
pd.Timestamp.now().strftime("%Y-%m-%d %H:%M:%S") }
import json message = json.dumps(result)
client.publish(topic_analysis, message)
print(f"分析結果を送信: {message}")
client.disconnect()
```

3-3. Node-RED で分析結果を受信

📌 タスク： 分析結果をダッシュボードに表示

1. 「mqtt in」ノード追加:
    - トピック: iot/analysis/result
2. 「json」ノード追加（JSON をオブジェクトに変換）
3. 「function」ノードで各値を抽出:

```
return [ { payload: msg.payload.average }, { payload: msg.payload.maximum },
{ payload: msg.payload.minimum } ];
```

4. 3 つの「text」ノードに接続:
    - 出力 1 → 平均温度
    - 出力 2 → 最高温度
    - 出力 3 → 最低温度

ステップ 4：自動アラート機能の実装

4-1. 閾値判定フローの作成

📌 タスク： 温度が 30 度を超えたらアラート

1. mqtt in (temperature) → switch ノード
2. switch ノードの設定:
    - 条件 1: > 30（高温アラート）
    - 条件 2: <= 30（正常）
3. 出力 1 → notification ノード（ダッシュボード通知）
4. notification 設定:
    - レイアウト: トースト
    - トピック: 高温警告
    - メッセージ: 温度が 30 度を超えました！

4-2. 通知カウンターの追加

📌 タスク： アラートの回数をカウント

1. switch ノードの出力 1 に function ノードを追加:

```
// グローバルカウンター var count = global.get('alertCount') || 0; count++;
```

```
global.set('alertCount', count);
msg.payload = count;
return msg;
```

2. text ノードに接続してアラート回数を表示

✅ チェックポイント 2：アラート機能が動作しますか？

☐ 温度が 30 度を超えると通知が表示される
☐ アラート回数がカウントされる

ステップ 5：システム全体のテストと最適化

5-1. 統合テスト

📌 **タスク：** 全システムが連携して動作するか確認

1. ESP32 が正常にデータを送信している
2. Node-RED ダッシュボードがリアルタイム更新されている
3. Ambient にもデータが蓄積されている
4. Python から分析結果を送信できる
5. アラート機能が動作する

統合テスト結果：

項目	結果	備考
ESP32→Node-RED	☐成功 ☐失敗	
ESP32→Ambient	☐成功 ☐失敗	
Python→Node-RED	☐成功 ☐失敗	
ダッシュボード表示	☐成功 ☐失敗	
アラート機能	☐成功 ☐失敗	

5-2. システムの改善

📌 課題： 以下の改善を 1 つ以上実装してください

- データ送信間隔の最適化（電力消費とリアルタイム性のバランス）
- 再接続ロジックの強化（Wi-Fi、MQTT 切断時の自動復旧）
- データの圧縮・バッチ送信（通信量削減）
- 複数センサーの追加（CO2、照度など）

- グラフの見やすさ向上（配色、範囲調整）

実装した改善内容を記録：

Part４ および 実習３ 全体のまとめ

🎉 **完成です！ESP32、Python、Node-RED を統合した完全な IoT システム**を構築しました

実習３で習得したスキル

Part	スキル
Part１	Python、Jupyter Notebook、Pandas
Part２	データ分析、可視化、統計、予測
Part３	Node-RED、MQTT、ダッシュボード
Part４	システム統合、自動化、アラート

最終考察：IoT システム開発で重要なこと

問１： ３つの実習（1, 2, 3）を通じて、最も重要だと感じたことは何ですか？
問２： このシステムを実社会でどう活用できると思いますか？具体的なアイデアを書いてください。
問３： さらに学びたい技術や、挑戦したい発展課題はありますか？

✅ 実習３ 完全達成チェック

☐ Part 1-4 をすべて完了した
☐ ESP32 から Node-RED にデータを送信できた
☐ Python でデータ分析ができた
☐ ダッシュボードが正常に動作した
☐ システム全体が統合されて動作した
☐ IoT システムを自分で設計・開発できる自信がついた

☑ 実習振り返りシート（IoT 技術応用 - 実習３ Part４）

振り返り日:＿＿＿＿年＿＿＿月＿＿＿日

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

できたこと／わかったこと

今日学んで理解できたことを、自分の言葉で書いてください。

難しかったこと

うまくいかなかったこと、難しかったことを書いてください。

自己評価

1（できなかった）～ 5（よくできた）で◯をつけてください。

項目	評価	コメント
理解度	1　2　3　4　5	
完成度	1　2　3　4　5	
積極性	1　2　3　4　5	
楽しさ	1　2　3　4　5	

次回への目標

次の Part で頑張りたいことや意気込みを書いてください。

自由記入欄（先生へのメッセージ・質問／感想等）

IoT 技術応用教材
実習 3　システム統合
発展課題

📋 発展課題について

基本の実習 3（Part 1-4）を完了した学生向けの、**追加の挑戦課題**です。

- **任意課題：** 必須ではありませんが、挑戦することで大きくスキルアップできます
- **難易度別：** 自分のレベルに合った課題を選択してください
- **加点評価：** 完成度に応じて、成績に加点されます
- **創造性歓迎：** 課題のアレンジ、独自のアイデア実装も高評価

🎯 難易度の説明

難易度	目安	推奨対象
初級	実習 3 の知識で対応可能	全員
中級	調査・工夫が必要	意欲的な学生
上級	高度な技術、時間を要する	特に興味がある学生

🌱 初級課題（推奨所要時間：1-2 時間）

初級-1：追加センサーの実装

初級

目標： DHT22 以外のセンサーを追加し、データを取得・表示する

実装内容：

- 照度センサー（CdS セル）、気圧センサー（BMP280）、CO2 センサー（MH-Z19）などを追加
- ESP32 で追加センサーのデータを取得
- MQTT で送信し、Node-RED ダッシュボードに表示

💡 **ヒント：**

- センサーのライブラリは Arduino Library Manager で検索
- 追加の MQTT トピックを定義（例: iot/esp32/light）
- Node-RED で新しい gauge や chart を追加

初級-2：アラート機能の拡張

初級

目標： より詳細なアラート条件を設定する

実装内容：

- 複数の閾値設定（警告レベル、危険レベル）
- 時間帯によって閾値を変える（昼/夜で異なる基準）
- アラート履歴の記録（CSV ファイルに保存）

💡 **ヒント：**

- Node-RED の switch ノードで多段階の条件分岐
- file ノードで CSV に書き込み
- 現在時刻の取得: new Date().getHours()

初級-3：グラフのカスタマイズ

初級

目標： Matplotlib のグラフをより見やすく、美しくする

実装内容：

- 複数の統計量を 1 つのグラフに表示（平均、最大、最小を重ねる）
- 配色の工夫（カラーマップの使用）
- 凡例、注釈の追加
- グラフを画像ファイルとして保存

💡 **ヒント：**

- plt.legend()で凡例追加
- plt.savefig('graph.png')で保存
- Matplotlib のギャラリーを参考に

🌿 中級課題（推奨所要時間：2-4 時間）

中級-1：LINE 通知の実装

中級

目標： 異常検出時に LINE で通知を送る

実装内容：

- LINE Notify API を使用
- Node-RED から HTTP リクエストで LINE 通知
- 温度、湿度、時刻などの情報を含めた通知メッセージ

💡 **ヒント：**

- LINE Notify のアクセストークンを取得（https://notify-bot.line.me/）
- Node-RED の「http request」ノードを使用
- POST メソッド、ヘッダーに認証情報

中級-2：時系列予測の実装

中級

目標： 過去のデータから未来の温度を予測する

実装内容：

- 移動平均法、または指数平滑法を実装
- 次の 1 時間後、3 時間後の温度を予測
- 予測値と実測値をグラフで比較
- 予測精度（RMSE、MAE など）を計算

💡 **ヒント：**

- Pandas の rolling()メソッドで移動平均
- scikit-learn の mean_squared_error()で精度評価
- より高度な予測には、Prophet、LSTM などを調査

中級-3：データベースへの保存

中級

目標： センサーデータをデータベースに蓄積する

実装内容：

- SQLite、MySQL、または InfluxDB を使用
- Node-RED からデータベースに書き込み
- Python でデータベースからデータを読み出して分析

💡 **ヒント：**

- SQLite：軽量で導入が簡単（node-red-node-sqlite）
- InfluxDB：時系列データに特化、Grafana と連携可能
- Python での接続：sqlite3、pymysql、influxdb-client ライブラリ

中級-4：複数 ESP32 の連携

中級

目標： 複数の ESP32 からデータを収集し、比較する

実装内容：

- 2 台以上の ESP32 を用意（異なる場所に設置）
- 各 ESP32 に識別 ID を付与
- Node-RED で複数デバイスのデータを統合
- デバイス間の温度差をグラフ化

💡 **ヒント：**

- MQTT トピックにデバイス ID を含める（例：iot/esp32_01/temperature）
- Node-RED の function ノードでデータを振り分け
- chart ノードで複数の線を表示

🌳 上級課題（推奨所要時間：4 時間以上）

上級-1：機械学習による異常検知

上級

目標： 教師なし学習で異常パターンを自動検出する

実装内容：

- Isolation Forest、Autoencoder などの異常検知アルゴリズムを実装
- 正常なデータでモデルを訓練
- リアルタイムデータに対して異常スコアを計算
- 異常度を Node-RED ダッシュボードに表示

💡 **ヒント：**

- scikit-learn の IsolationForest から始める
- 特徴量：温度、湿度、時刻、変化率など

- モデルを保存し、Node-RED から呼び出す（Flask API など）

上級-2：クラウドプラットフォームの活用

上級

目標： AWS IoT、Azure IoT、Google Cloud IoT などを使用

実装内容：

- ESP32 からクラウドにデータ送信
- クラウドサービスでデータ処理・保存
- Lambda（AWS）、Functions（Azure）などでサーバーレス処理
- クラウドから Node-RED または Web アプリに通知

💡 **ヒント：**

- AWS IoT Core: MQTT での接続、証明書認証
- 無料枠を活用（学生アカウントもあり）
- まずはチュートリアルから始める

上級-3：音声アシスタント連携

上級

目標： Google Assistant や Alexa と連携

実装内容：

- 「今の部屋の温度は？」と聞くと、音声で返答
- 「エアコンをつけて」などの音声コマンドで制御（ESP32 経由）
- IFTTT、Dialogflow、Alexa Skills などを活用

💡 **ヒント：**

- IFTTT が最も簡単（Webhook と連携）
- Node-RED から HTTP エンドポイントを公開
- ngrok でローカルサーバーを外部公開（開発時のみ）

上級-4：Web アプリケーションの開発

上級

目標： Node-RED ではなく、独自の Web アプリを開発

実装内容：

- React、Vue.js、または Flask で Web アプリ作成
- リアルタイムグラフ表示（WebSocket または SSE）
- ユーザー認証、データのダウンロード機能
- レスポンシブデザイン（スマホ対応）

💡 **ヒント：**

- バックエンド: Flask（Python）または Express（Node.js）
- フロントエンド: Chart.js や D3.js でグラフ
- WebSocket: Socket.IO ライブラリ

🎨 自由課題

上記以外にも、**独自のアイデア**を実装することを歓迎します！

アイデア例：

- スマート植物栽培システム（土壌水分センサー ＋ 自動水やり）
- ペット見守りシステム（カメラ ＋ 温度 ＋ 動体検知）
- エネルギー監視システム（電流センサーで消費電力測定）
- 空気質モニタリング（PM2.5、CO2、VOC センサー）
- スマートロック（RFID ＋ ESP32）
- 位置情報トラッキング（GPS ＋ 地図表示）

提出方法

提出物：

- 実装したコード（ESP32、Python、Node-RED フロー）
- 動作している様子の動画またはスクリーンショット
- レポート（実装内容、工夫した点、苦労した点、考察）

評価ポイント：

- **完成度:** 実装がきちんと動作しているか
- **工夫:** 独自のアイデア、改善があるか
- **技術力:** 新しい技術を習得したか
- **考察:** 課題から何を学んだか、今後の展望

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技術応用教材ワークブック

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