令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」

ＩｏＴ技術応用教材資料１

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

情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

目　次

IoT技術応用 3
はじめに 3
IoT技術応用_実習１ 8
Part1 8
Part2 17
Part3 28
Part4 37
IoT技術応用_実習２ 49
導入編 49
Part1 56
Part2 64
Part3 72
Part4 80
IoT技術応用_実習３ 88
導入編 88
Part1 94
Part2 102
Part3 110
Part4 118

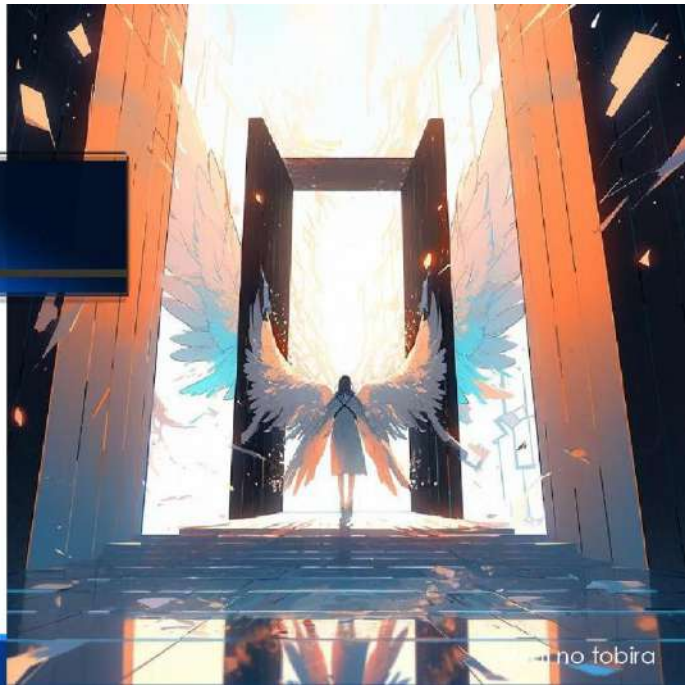
ＩｏＴ技術応用

はじめに

IoT技術応用

「基礎から実践へ
3つの実習で学ぶIoTシステム構築」

実習１ デバイス制御
実習２ クラウド接続
実習３ システム統合



IoT技術応用 実習ガイド

IoT技術応用

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。基礎教材は全7章で構成され、IoTシステムの理論や仕組みを体系的に学習しましたが、『**IoT技術応用**』では、それらの理論を３つの実習で、実際のハードウェアとプログラミングを体験し、**「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

『IoT技術応用』
（実践・体験による理解の深化）
実習1 デバイス制御
実習2 クラウド接続
実習3 システム統合

学習のゴール

IoT技術応用

- ✓『IoT技術基礎』で学んだ知識を実習によって**「使えるスキル」**に変える
- ✓「センサー → クラウド → 分析」というIoTシステムの**「実践的な流れ」**を体験する
- ✓ハードウェアとソフトウェアを連携させる**「システム構築力」**を習得する



実習構成

IoT技術応用

実習 1 デバイス制御
Arduino Unoによる
センサーデータ取得、
SDカード記録、
LCD表示

→

実習 2 クラウド接続
ESP32 (Wi-Fi)による
クラウド（Ambient
/ ThingSpeak）への
データ送信・可視化

→

実習 3 システム統合
Pythonによるデータ分
析とNode-REDによる
ダッシュボード構築、
MQTT連携

IoT技術基礎		IoT技術応用		
第1章 IoTシステム構成	→	実習 1	実習 2	
第2章 クラウドコンピューティング	→		実習 2	
第3章 エッジコンピューティング	→			実習 3
第4章 IoTデータ活用技術	→	実習 1		実習 3
第5章 IoT通信方式	→	実習 1	実習 2	
第6章 IoTデバイス	→	実習 1		
第7章 IoTシステム開発	→	実習 1		実習 3

実習で使用する主なツール・技術

IoT技術応用

Arduino Uno / ESP32	：	IoTデバイスのコアとなるマイコン
Arduino IDE	：	マイコンのプログラミング環境（C++）
各種センサー/デバイス	：	DHT11/22, SDカード, LCDなど
クラウドサービス	：	Ambient, ThingSpeak（データ蓄積・可視化）
Python (Pandas)	：	蓄積データの分析
Node-RED / MQTT	：	リアルタイムデータ処理、ダッシュボード構築

実習の進め方

IoT技術応用

導入ガイド	：	実習の概要、環境準備、データの入手方法等
ワークブック（Part1-4）	：	実際にコードや考察を記入する教材
オンライン講義（動画）	：	各Partの解説とデモ
発展課題	：	知識とスキルをさらに磨く追加課題





実習ステップ

IoT技術応用

[準備] **導入ガイド**を読み、環境と機材を準備する

⬇

[視聴] **オンライン講義**（動画）を視聴する

⬇

[実践] **ワークブック**のタスクに沿って、自分で配線やコード実行する

⬇

[記録] **ワークブック**に実行結果を記録する

⬇

[考察] 気づきや振り返りを、自分の言葉でまとめる

⬇

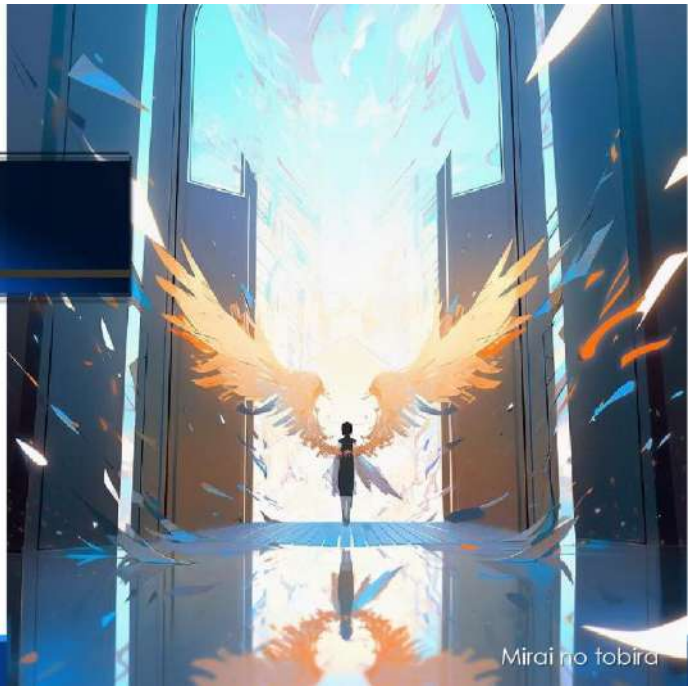
[挑戦] 発展課題に挑戦し、さらにスキルを磨く

NEXT

IoT技術応用

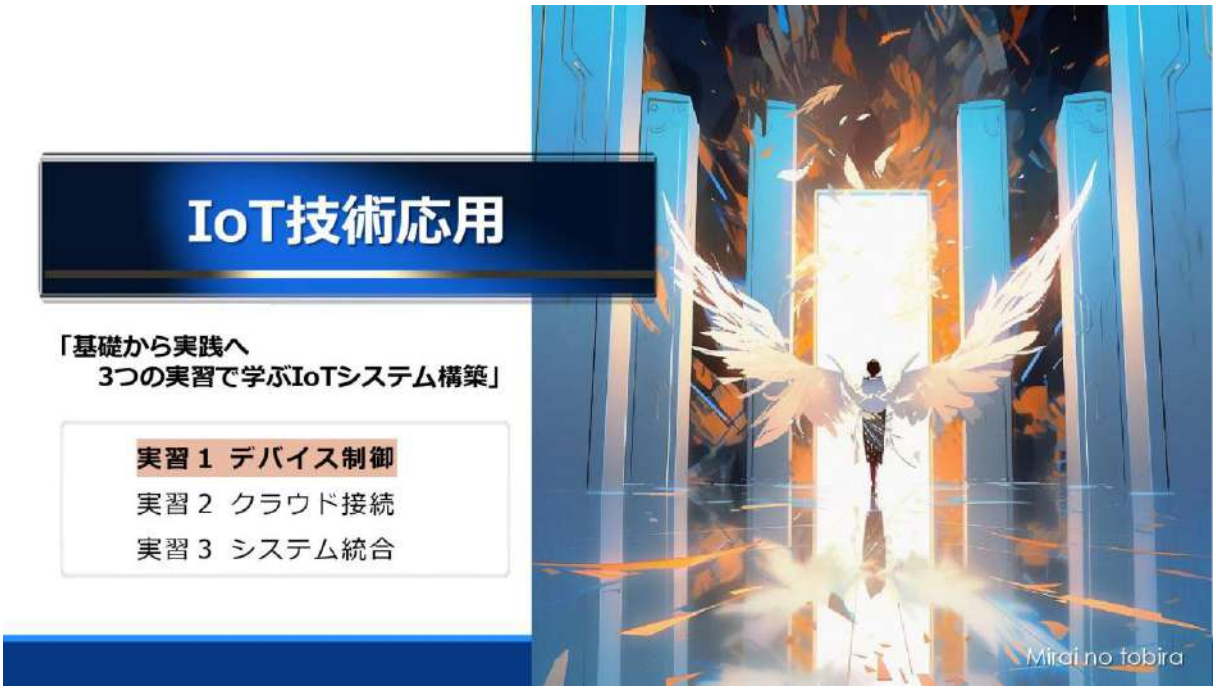
実習1： デバイス制御

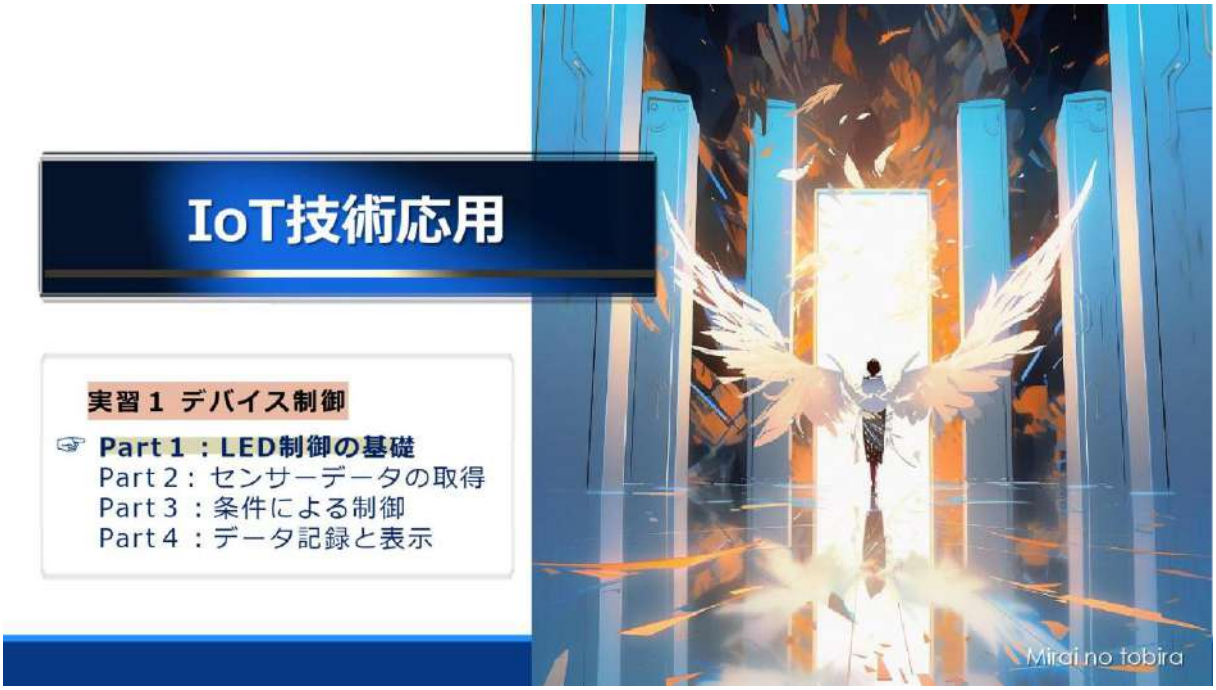
☞ **Part 1：LED制御の基礎**
Part 2: センサーデータの取得
Part 3：条件による制御
Part 4：データ記録と表示
実習2 クラウド接続
実習3 システム統合



ＩｏＴ技術応用_実習１

Part1


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


IoT技術応用
実習１ デバイス制御
Part１：LED制御の基礎
Part２：センサーデータの取得
Part３：条件による制御
Part４：データ記録と表示
Mirai no tobira

実習１ デバイス制御

IoT技術応用

実習１デバイス制御 → 実習２ クラウド接続 → 実習３システム統合

- **Part１：LED制御の基礎**
- Part２：センサーデータの取得
- Part３：条件による制御
- Part４：データ記録と表示



Part１：LED制御の基礎

実習１ デバイス制御

Arduino IDE と最初のプログラム「Lチカ」

Part１：LED制御の基礎 実習ステップ

実習１ デバイス制御

Arduino IDE と最初のプログラム「Lチカ」

- ✓ Arduino IDE をインストール
- ✓ Arduino を PCに接続して設定
- ✓ ブレッドボードに LEDと抵抗を配線
- ✓ プログラムを Arduino に書き込み、LEDを点滅させる
- ✓ プログラムを変更し、点滅間隔を変える

Part１：LED制御の基礎 の 準備

実習１ デバイス制御

「準備するもの」

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



ステップ１：Arduino IDE の準備

実習１ デバイス制御 Part１

Arduino IDE のインストール

- **Arduino IDE をダウンロード**
  - Arduino公式サイト **arduino.cc** にアクセス
  - **「Product」 → 「Software」 → 「Arduino IDE 」**
    → 「Windows Win 10 or newer (64-bit)」 → 「DOWNLOAD」
- ダブルクリックして**インストール**
- Arduino IDE を**起動**

ステップ２：Arduino 接続と設定

実習１ デバイス制御 Part１

1. **Arduino を PC に接続**
2. **Arduino と PC を USBケーブル**で接続
3. **PC が Arduino を認識しているか確認**
4. **Arduino IDE を設定（ボードの選択）**

「ツール」→「ボード」→「Arduino Uno」

「ツール」→「シリアルポート」→「COMxx (Arduino Uno)」

重要
Windows の「デバイスマネージャー」を開き、**「ポート (COMとLPT)」**という項目を開きます
「Arduino Uno (COMxx)」と表示されていればOK。COMのあとの数字、番号をワークブックにメモしてください

ステップ３：LED の配線

実習１ デバイス制御 Part１

- **ブレッドボード**
- **LEDの向き：**
  - ✓ 長い足（アノード, +）と
    短い足（カソード, −）を確認
- **抵抗の役割：**
  - LEDに流れる電流を制限し、壊れるのを防ぐ
- **配線：**
  - Arduino D13 → LEDの長い足
  - LEDの短い足 → 抵抗（220Ω）
  - 抵抗の反対側 → Arduino GND





ステップ４：プログラムの書き込み

実習１ デバイス制御 Part１

サンプルプログラム「Blink」を開く

1. **配線完了後、USBケーブル**を接続
2. **Blink を開く：「ファイル」→「スケッチ例」**
   →「01.Basics」→「Blink」
3. **検証：✓**（チェックマーク）**ボタンでコンパイル**
4. **書き込み：→**（矢印）**ボタンで Arduino に書き込み**

Blink プログラムの解説

実習 1 デバイス制御 Part 1

```
// 最初に1回だけ実行される
void setup() {
 pinMode(LED_BUILTIN, OUTPUT);     // D13ピンを「出力用」に設定
}
// ずっと繰り返し実行される
void loop() {
 digitalWrite(LED_BUILTIN, HIGH); // D13ピンに電気を流す（LED点灯）
 delay(1000);                     // 1000ミリ秒（1秒）待つ
 digitalWrite(LED_BUILTIN, LOW);  // D13ピンの電気を止める（LED消灯）
 delay(1000);                     // 1000ミリ秒（1秒）待つ
}
```

- setup()　：電源が入った時に1回だけ実行される初期設定
- loop()　：ずっと繰り返されるメイン処理
- digitalWrite()：ピンのON (HIGH) / OFF (LOW) を切り替える命令
- delay()　：指定した時間（ミリ秒単位）だけ処理を止める命令

ステップ 5 ：実験

実習 1 デバイス制御 Part 1

Delay の数値を変更して、動作の違いを観察する

- **実験1：両方の delay(1000) を delay(500) に変更**
  → 0.5秒点灯、0.5秒消灯。速く点滅する
- **実験2：両方の delay(1000) を delay(2000) に変更**
  → ゆっくり点滅する
- **他の実験例：点灯を delay(2000)、消灯を delay(500) に変更**
  → 長く光って、短く消えるリズムになる

実習１ Part１のまとめ

実習１ デバイス制御 Part１

- ✓ Arduino IDE の接続と設定方法を学んだ
- ✓ LEDと抵抗の正しい配線方法を学んだ
- ✓ setup() と loop() という基本構造を理解した
- ✓ pinMode(), digitalWrite(), delay() の使い方を学んだ
- ✓ Lチカを成功させ、ハードウェア制御の基礎を理解した

NEXT

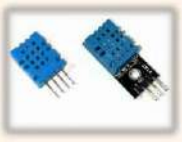
実習１ Part２: センサーデータの取得

実習１ デバイス制御

実習１ Part２では、**「入力」**を学びます

DHT11温湿度センサーを使って、

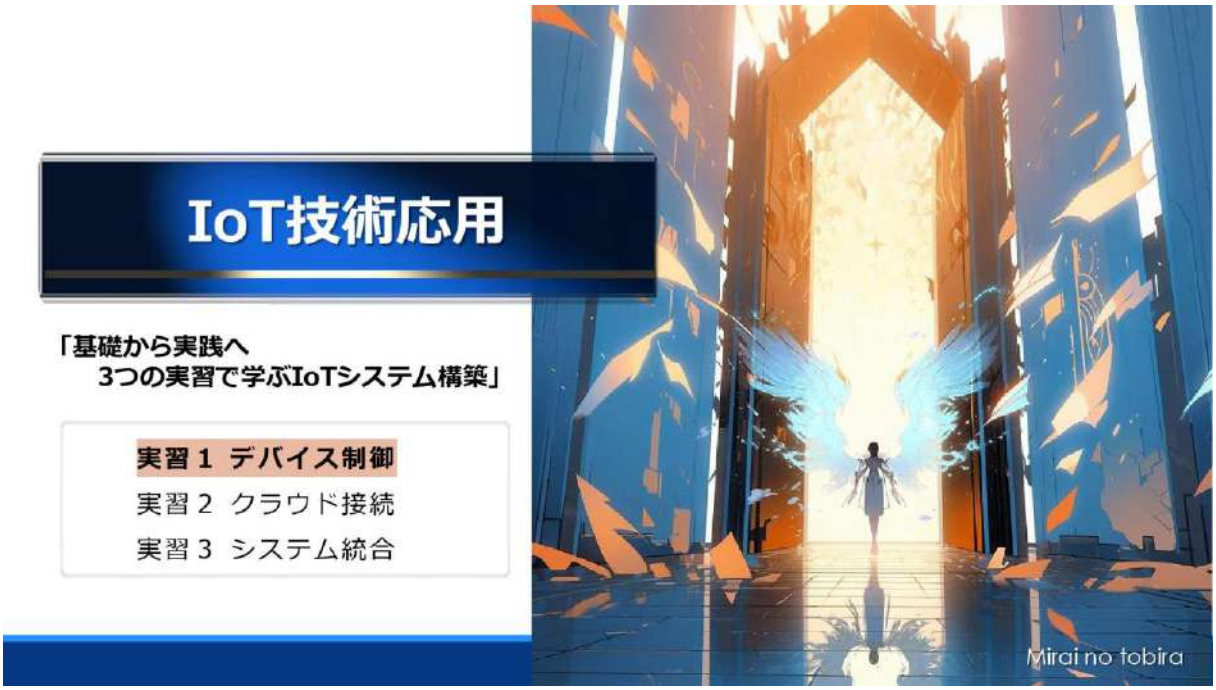
環境データを取得します

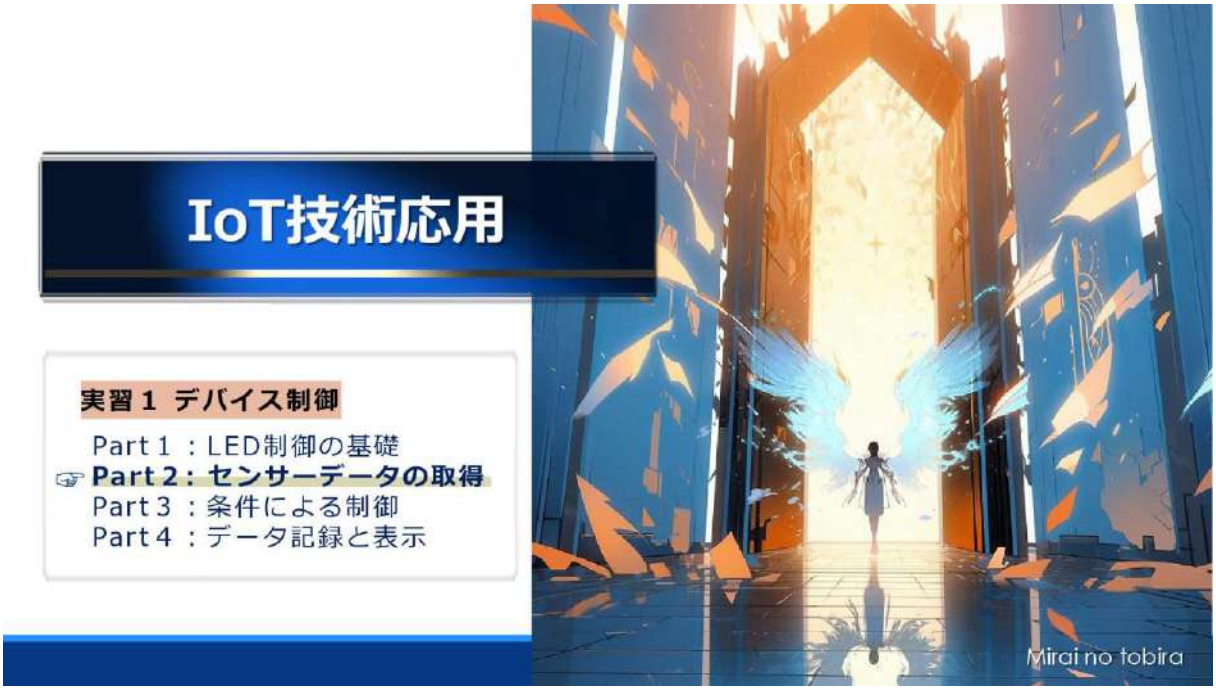



NEXT
IoT技術応用
実習１ デバイス制御
Part１：LED制御の基礎
Part２：センサーデータの取得
Part３：条件による制御
Part４：データ記録と表示
Mirai no tobira

ＩｏＴ技術応用_実習１

Part2


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


IoT技術応用
実習１ デバイス制御
Part１：LED制御の基礎
Part２：センサーデータの取得
Part３：条件による制御
Part４：データ記録と表示
Mirai no tobira

実習１ デバイス制御

IoT技術応用

実習１ デバイス制御 → 実習２ クラウド接続 → 実習３ システム統合

Part１：LED制御の基礎

Part２：センサーデータの取得

Part３：条件による制御

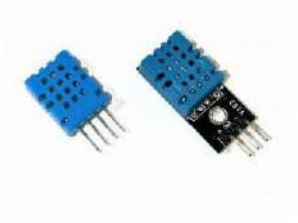
Part４：データ記録と表示



Part２：センサーでデータ取得

実習１ デバイス制御

DHT11温湿度センサーとシリアルモニタ





Part２：センサーでデータ取得 の達成目標

実習１ デバイス制御

Part１の「出力」に続き、Part２では「入力」を学ぶ

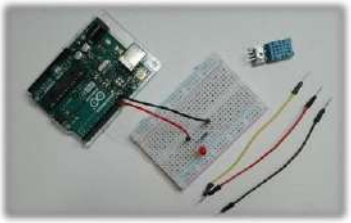
- ✓ DHT11 センサーを正しく配線できる
- ✓ センサー用の「ライブラリ」を利用できる
- ✓ 温度と湿度を測定できる
- ✓ 測定したデータを PCの「シリアルモニタ」に表示できる
- ✓ センサーエラー（nan）の意味がわかる

センサーでデータ取得の 準備

実習１ デバイス制御

「準備するもの」

- ➢ Part１で作成した回路（LED、抵抗）
- ➢ DHT11 温湿度センサー ×1
  （部品単体の場合は、抵抗器10KΩ）
- ➢ ジャンパーワイヤー ×3



ステップ１：ライブラリ インストール

実習１ デバイス制御 Part２

ライブラリという便利な道具のセットをインストール

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」**
3. 「ライブラリーマネージャ」検索欄に**「DHT sensor library」**と入力
4. **「DHT sensor library by Adafruit」**を見つけて**「インストール」**
5. 依存ライブラリも**「すべてインストール」**
6. 「インストールド」表示で終了

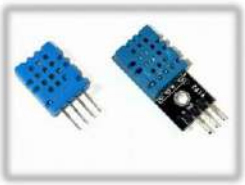
ステップ２：センサーの確認

実習１ デバイス制御 Part２

Part１の回路に、DHT11 を追加配線

ピンの確認：センサーのピン（VCC, DATA, GND）を確認します

- **配線**
  - ✓ **DHT11 VCC**（+） → **5V**
  - ✓ **DHT11 DATA**（中央） → **D2**
  - （NC → ×）
  - ✓ **DHT11 GND**（−） → **GND**



ステップ３：センサーの配線

実習１ デバイス制御 Part２

Part１の回路に、DHT11 を追加配線

ピンの確認：センサーのピン（VCC, DATA, GND）を確認します

- **配線**
  - ✓ **DHT11 VCC （+） → Arduino 5V**
  - ✓ **DHT11 DATA（中央） → Arduino D2**
  - ✓ **DHT11 GND （−） → Arduino GND**



ステップ４：プログラムの作成

実習１ デバイス制御 Part２

```
#include <DHT.h>          // ライブラリの読み込み

#define DHTPIN 2          // センサーのDATAピンをD2に接続
#define DHTTYPE DHT11     // 使うセンサーの種類 DHT

dht(DHTPIN, DHTTYPE);     // センサーを「dht」という名前で準備

void setup() {
Serial.begin(9600);       // PCと通信を開始 (9600ボーレート)
dht.begin();              // センサーを起動
}

void loop() {                                   // データを測定
float temp = dht.readTemperature();             // 温度を測定
float humi = dht.readHumidity();                // 湿度を測定

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

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

delay(2000); // 2秒待つ
}
```

Part２：センサーでデータ取得　IoT技術応用 実習１

ステップ４：プログラムの作成

実習１ デバイス制御 Part２

```
#include <DHT.h>          // ライブラリの読み込み

#define DHTPIN 2          // センサーのDATAピンをD2に接続
#define DHTTYPE DHT11     // 使うセンサーの種類 DHT

dht(DHTPIN, DHTTYPE);     // センサーを「dht」という名前で準備

void setup() {
Serial.begin(9600);       // PCと通信を開始 (9600ボーレート)
dht.begin();              // センサーを起動
}

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

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

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

delay(2000); // 2秒待つ
}
```


ステップ４：プログラムの作成

実習１ デバイス制御 Part２

```
#include <DHT.h>          // ライブラリの読み込み

#define DHTPIN 2          // センサーのDATAピンをD2に接続
#define DHTTYPE DHT11     // 使うセンサーの種類 DHT

dht(DHTPIN, DHTTYPE);     // センサーを「dht」という名前で準備

void setup() {
Serial.begin(9600);       // PCと通信を開始 (9600ボーレート)
dht.begin();              // センサーを起動
}

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

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

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

delay(2000); // 2秒待つ
}
```

Part２：センサーでデータ取得　IoT技術応用 実習１

ステップ４：プログラムの作成

```
13 void loop() {                                  // データを測定
14 float temp = dht.readTemperature();            // 温度を測定
15 float humi = dht.readHumidity();               // 湿度を測定
16
17 Serial.print("温度: ");
18 Serial.print(temp);
19 Serial.println("℃");
20
21 Serial.print("湿度: ");
22 Serial.print(humi);
23 Serial.println("%");
24
25 delay(2000); // 2秒待つ
26 }
```


ステップ５：動作確認と実験

実習 1 デバイス制御 Part 2

プログラムを書き込み、動作を確認

1. **プログラム**を **Arduino** に書き込みます
2. IDE右上の**「シリアルモニタ」**（虫眼鏡アイコン）を**クリック**します
3. シリアルモニタ右下の**「ボーレート」**を **9600 baud** に合わせます
4. 温度と湿度が２秒ごとに表示されるか**確認**します

重要な概念

実習 1 デバイス制御 Part 2

大切な概念の確認

float型：小数点のある数字を扱うデータ型
⇒ 温度や湿度は小数点以下も大切なので、floatを使います

Serial.print() と Serial.println () :
プリント()は改行しない、プリントライン()は改行する
⇒ この違いで、表示の仕方が変わります

nan：Not a Number、「数字ではない」という意味
⇒ センサーエラーのサイン、配線ミスやセンサーの不良が原因です

実験：センサーの反応

実習 1 デバイス制御 Part 2

センサーが正しく反応するか試す

- **実験1：センサーに息を吹きかける**
  → 温度と湿度はどう変化しますか？
- **実験2：センサーを手で軽く包む**
  → 温度はどう変化しますか？

ワークブックに記録しよう
「観察と実験」の欄に、変化した値と、なぜそうなったかの考察を記録してください

実習１ Part２のまとめ

実習１ デバイス制御 Part２

- ✓ 「ライブラリ」を使うと複雑なセンサーが簡単に使えることを学んだ
- ✓ DHT11センサーで温度と湿度の２つのデータを取得できた
- ✓ Serial.begin() と Serial.print() の使い方を学んだ
- ✓ 「シリアルモニタ」でデバイスの動作の確認方法を習得した
- ✓ float型（浮動小数点数）を使って、小数点以下の値を扱った

NEXT

実習１ Part３：条件による制御

実習１ デバイス制御

実習１ Part３では、
Part１の**「出力（LED）」**と
Part２の**「入力（センサー）」**を
組み合わせます
「温度がxx度以上ならLEDを光らせる」
という**システム**を作ります


NEXT
IoT技術応用
実習１ デバイス制御
Part１：LED制御の基礎
Part２：センサーデータの取得
Part３：条件による制御
Part４：データ記録と表示
Mirai no tobira

ＩｏＴ技術応用＿実習１

Part3


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


IoT技術応用
実習１： デバイス制御
Part１：LED制御の基礎
Part２：センサーデータの取得
☞ Part３：条件による制御
Part４：データ記録と表示
Mirai no tobira

実習１ デバイス制御

IoT技術応用

実習１ デバイス制御 → 実習２ クラウド接続 → 実習３ システム統合

Part１：LED制御の基礎
Part２：センサーデータの取得
☞ **Part３：条件による制御**
Part４：データ記録と表示



実習１ Part３：条件による制御

実習１ デバイス制御

温度によって LED の動作が変わるシステム

温度が

- **25℃未満のとき　　：LED 消灯**
- **25℃以上28℃未満のとき：LED がゆっくり点滅**
- **28℃以上のとき　　：LED が速く点滅**

⇒ 温度という「条件」によって、LED の動作が変わる

実習１ Part３：条件による制御 の達成目標

実習１ デバイス制御

Part１の「出力」とPart２の「入力」を組み合わせ
Arduino に「判断」させます

- ✓ if文（もし～なら）の使い方がわかる
- ✓ 比較演算子（>=, < など）を使える
- ✓ 温度に応じてLEDの点滅パターンを変えることができる
- ✓ 「入力→処理→出力」というシステムの基本形を作れる

Part３：条件による制御の 準備

実習１ デバイス制御

「準備するもの」

Part２で組み立てた回路をそのまま使います

- Part２で作成した回路
  - Arduino Uno
  - LEDと抵抗（D13, GND）
  - DHT11センサー（D2, 5V, GND）
- USBケーブル
- PC（Arduino IDE）

ステップ１：比較演算子

実習１ デバイス制御 Part３

If文の条件を書くときに使う記号「比較演算子」

比較演算子：

temp == 25　（温度が25と等しい）
temp != 25　（温度が25ではない）
temp > 25　（温度が25より大きい）
temp >= 25　（温度が25以上）
temp < 25　（温度が25より小さい）
temp <= 25　（温度が25以下）

ステップ２：else if, else

実習１ デバイス制御 Part３

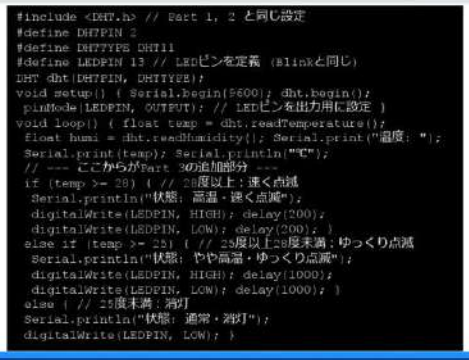
3つ以上のパターンに分ける場合は
else if と else を使います

```
if (temp >= 28) {
 // 条件1：温度が28度以上なら              // → 速く点滅
} else if (temp >= 25) {
 // 条件2：28度未満 かつ 25度以上なら      // → ゆっくり点滅
} else {
 // 条件3：それ以外（25度未満）なら        // → 消灯
}
```

ステップ３：プログラムの作成

実習１ デバイス制御 Part３

Part２のプログラムに、if文とLED制御のコードを追加します



```
#include <DHT.h> // Part 1, 2 と同じ設定
#define DHTPIN 2
#define DHTTYPE DHT11
#define LEDPIN 13 // LEDピンを定義 (Blinkと同じ)
DHT dht(DHTPIN, DHTTYPE);
void setup() { Serial.begin(9600); dht.begin();
 pinMode(LEDPIN, OUTPUT); // LEDピンを出力用に設定 }
void loop() { float temp = dht.readTemperature();
 float humi = dht.readHumidity(); Serial.print("温度: ");
 Serial.print(temp); Serial.println("℃");
 // --- ここからがPart 3の追加部分 ---
 if (temp >= 28) { // 28度以上：速く点滅
  Serial.println("状態: 高温・速く点滅");
  digitalWrite(LEDPIN, HIGH); delay(200);
  digitalWrite(LEDPIN, LOW); delay(200); }
 else if (temp >= 25) { // 25度以上28度未満：ゆっくり点滅
  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 { // 25度未満：消灯
 Serial.println("状態: 通常・消灯");
 digitalWrite(LEDPIN, LOW); }
```



```
#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("℃");
 // --- ここからが Part3の追加部分 ---
 if (temp >= 28) { // 28度以上：速く点滅
  digitalWrite(LEDPIN, HIGH); delay(200);
  digitalWrite(LEDPIN, LOW); delay(200);
 } else if (temp >= 25) { // 25度以上28度未満：ゆっくり点滅
  digitalWrite(LEDPIN, HIGH); delay(1000);
  digitalWrite(LEDPIN, LOW); delay(1000);
 } else { // 25度未満：消灯
  digitalWrite(LEDPIN, LOW); }
}
```

ステップ４：動作確認と実験

実習１ デバイス制御 Part３

プログラムを書き込み、動作を観察しましょう

1. **書き込み**：**プログラム**を **Arduino** に書き込みます
2. **シリアルモニタ確認**：**現在の温度**と**「状態」**メッセージを確認します
3. **LED確認**：LEDが**シリアルモニタの状態通り**に動作していますか？
4. **実験**：センサーを手で温めて、**温度**を変えてみよう

ワークブックに記録しよう
温度が境界（25℃/28℃）を超えたときのLEDの動作とシリアルモニタの表示を記録してください

実験：閾値（しきい値）の変更

実習１ デバイス制御 Part３

閾値（しきい値）を変更しよう

- ➢ **「閾値」**とは、動作が変わる境界の値のこと
  （今回はプログラムでは、28と25）
- ✓ この値を実験しやすい値に変更してみましょう
  例：アラームを25度、注意を17度に設定する

```
if (temp >= 30) {      // 28 → 25 に変更   // 速く点滅 }
else if (temp >= 27) { // 25 → 17 に変更   // ゆっくり点滅 }
else {                                     // 消灯 }
```

ワークブックに記録しよう
「実験3：閾値の変更」の欄に、変更した値と、その動作結果を記録してください

ステップ５：シリアルモニタへのメッセージ追加

実習１　デバイス制御 Part３

LEDがどの状態か、シリアルモニタにも表示させる

それぞれの条件文の中に、表示用メッセージを追加します

if (temp >= 28) の中、digitalWrite(LEDPIN, HIGH); の前に追加

```
Serial.println ("状態: 高温・速く点滅");
```

else if (temp >= 25) の中にも追加

```
Serial.println ("状態: やや高温・ゆっくり点滅");
```

else　の中にも追加

```
Serial.println ("状態: 通常・消灯");
```


実習１ Part３のまとめ

実習１　デバイス制御 Part３

- ✓ if文を使った条件分岐を学んだ
- ✓ else if, else を使った複数条件分岐を学んだ
- ✓ 比較演算子の使い方を学んだ
- ✓ 「出力」と「入力」を統合し「処理」を追加した
- ✓ 閾値の調整を学んだ

NEXT

実習１ Part４：データ記録と表示

実習1：デバイス制御

実習１ Part４では、

システムを**さらに実用的**にします

高度なセンサーを使い

SDカードにデータを記録し

LCDディスプレイに結果を表示します

NEXT

IoT技術応用

実習１ デバイス制御

Part１：LED制御の基礎
Part２：センサーデータの取得
Part３：条件による制御
☞ **Part４：データ記録と表示**



ＩｏＴ技術応用_実習１

Part4


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


IoT技術応用
実習1 デバイス制御
Part１：LED制御の基礎
Part２：センサーデータの取得
Part３：条件による制御
☞ Part４：データ記録と表示
Mirai no tobira

実習１ デバイス制御

IoT技術応用

実習１ デバイス制御 → 実習２ クラウド接続 → 実習３ システム統合

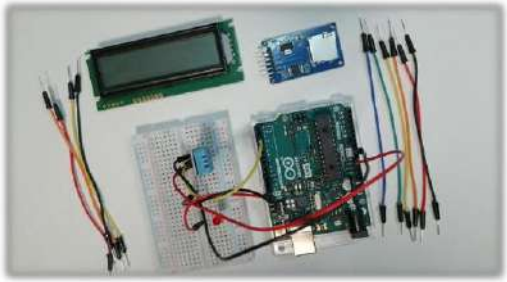
Part１：LED制御の基礎
Part２：センサーデータの取得
Part３：条件による制御
☞ **Part４：データ記録と表示**

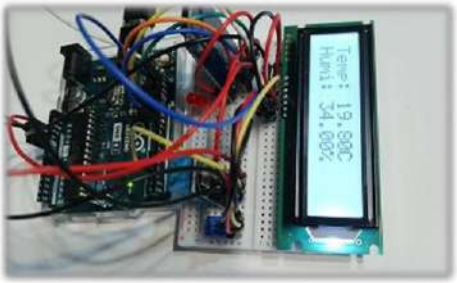


Part４：データ記録と表示

実習１ デバイス制御

SDカード と LCDディスプレイ でシステム完成





IoT技術応用 実習１

Part４：データ記録と表示 の達成目標

実習１　デバイス制御

**Part３のシステムに「記録」と「表示」の機能を追加し、
実用的なシステムを完成させます**

- **【記録】** 測定した温度と湿度を、
  SDカードにCSVファイルとして自動記録する
- **【表示】** 現在の温度と湿度を、
  LCDディスプレイにリアルタイム表示する
- **【統合】** これら全ての機能（センサー、LED、SD、LCD）を
  同時に動かす

新しい通信方式：SPI と I2C

実習１　デバイス制御

新しい「通信方式（プロトコル）」を使います

通信方式	特徴	使用デバイス	配線
SPI (エスピーアイ)	高速通信が可能 配線が多い (4本)	**SDカード**	MISO, MOSI, SCK, CS
I2C (I²C) (アイツーシー)	低速だが配線が少ない (2本) 多くのデバイスを接続可能	**LCDディスプレイ**	SDA, SCL

ポイント
「どんなデバイスを」「どう繋ぐか」で、適切な通信方式が変わることを理解しましょう

SDカードモジュール

実習１ デバイス制御 Part４

SDカードモジュールを理解します

VCC → 電源（+）
GND → グランド（−）
MISO → データ受信
MOSI → データ送信
SCK → クロック信号
CS → チップセレクト



※ SDカードの準備：microSDカードをモジュールに差し込みます

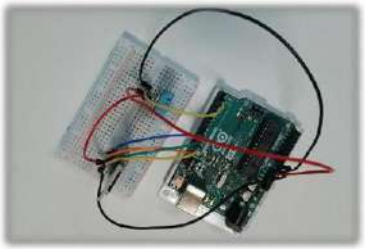
ポイント
使えるSDカードは、32GB以下、FAT32形式でフォーマットしておきます

ステップ1：SDカードの配線 (SPI)

実習１ デバイス制御 Part４

SDカードモジュールを接続します

配線（合計6本）：
VCC → 5V
GND → GND
MISO → Arduino D12
MOSI → Arduino D11
SCK → Arduino D13
CS → Arduino D10



ポイント
一旦 LEDの配線（Arduino D13 – LED - 抵抗 - GND）は、取り除いてください。後で再設定します

SDカードのプログラム

実習１ デバイス制御 Part４

SDカードを扱うには、2つのライブラリが必要

```
#include <DHT.h>
#include <SD.h>
#include <SPI.h>
#define DHTPIN 2
#define DHTTYPE DHT11
#define LEDPIN 13
#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);
}
```


SDカードのプログラム

実習１ デバイス制御 Part４

ライブラリを読み込み

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

定数を定義

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

DHT dht(DHTPIN, DHTTYPE);
```

setup関数

```
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");
}
```

SDカードのプログラム

実習１ デバイス制御 Part４

loop関数

```
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カードに書き込む処理

```
 // 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");
  }
```


SDカードのプログラム

実習１ デバイス制御 Part４

```
// 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);
 }

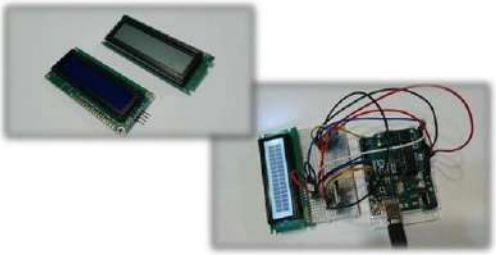
 delay(2000);
}
```

ステップ2：LCDの配線 (I2C)

実習１ デバイス制御 Part４

LCDディスプレイを接続します
I2C なので配線はわずか4本です

- 配線（合計4本）：
  VCC → 5V or 3.3V
  GND → GND
  SDA → Arduino A4
  SCL → Arduino A5



Part４：データ記録と表示　IoT技術応用 実習１

LCD のプログラム

実習１ デバイス制御 Part４

I2C用のライブラリをインストールします

- ライブラリのインストール：
  「ライブラリを管理」で「LiquidCrystal I2C」を検索し、
  インストールします

ポイント
LCDによって、インストールするライブラリが異なる場合があります
それにともない一部の命令も異なる場合があります
各LCDのデータシートを確認してください

Part４：データ記録と表示　IoT技術応用 実習１

LCD のプログラム

実習１ デバイス制御 Part４

```
#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 のプログラム

ライブラリを追加

#include <SPI.h> の下に1行追加

```
#include <LiquidCrystal_I2C.h>
```

LCDオブジェクトを作る

DHT dht(DHTPIN, DHTTYPE); の下に1行追加

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

0x27は LCDのアドレス（※LCDによって異なります） 16は文字数、2は行数

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3行追加

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

これで、LCDが初期化され、バックライトが点灯して「Starting...」と表示される

LCD のプログラム

実習 1 デバイス制御 Part 4

loop関数にLCD表示を追加

Serial.println("%"); の下に、8行追加

```
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("%");
```

これで、1行目に温度、2行目に湿度が表示される

ステップ 3 ：動作確認と実験

実習 1 デバイス制御 Part 4

- ✓ **LCDディスプレイ**：温度と湿度が表示され、更新されているか？
- ✓ **シリアルモニタ**：温度、湿度と「データ保存成功」と表示されているか？
- ✓ **SDカード**：電源を抜き、PC で SDカードのファイルの中身を確認
  「data.txt」にデータが記録されているか？

ワークブックに記録しよう

「動作確認記録」と「実験記録表」に、各デバイスの動作を記録してください
SDカードに記録されたデータ（data.txtの中身）も、記載しておきましょう

ステップ４：統合システム

実習１ デバイス制御 Part４

- 一旦、配線をはずしていたLEDを再度配線します
- これで、LCDの表示、SDカードへのデータ保存、そしてLEDの点滅で温度や湿度の規定値超過アラートが視認できるシステムが完成します
- 手順
  - LED と抵抗をブレッドボードの適切な位置にさす
    - LED の短い方（−）と抵抗は接続するように配置する
  - 抵抗の片方を GND に接続する
  - LED の長い方（+）を、Arduino D3 に接続する
  - プログラムの定義を変更・確認する
    - #define LEDPIN 3
  - Arduinoに書き込んで実行し、動作を確認する

実習１：デバイス制御 まとめ

IoT技術応用

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）**
- ✓ **条件に応じて動作を変える（LED制御）**
- ✓ **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**

NEXT

実習２：クラウド接続

IoT技術応用

実習１では、PCやSDカードという
「ローカル」にデータを保存しました

実習２では、
「クラウド」にデータを送信します
使用するマイコンも **Wi-Fi機能付きの**
ESP32 に変わります

NEXT

IoT技術応用

「基礎から実践へ
3つの実習で学ぶIoTシステム構築」

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



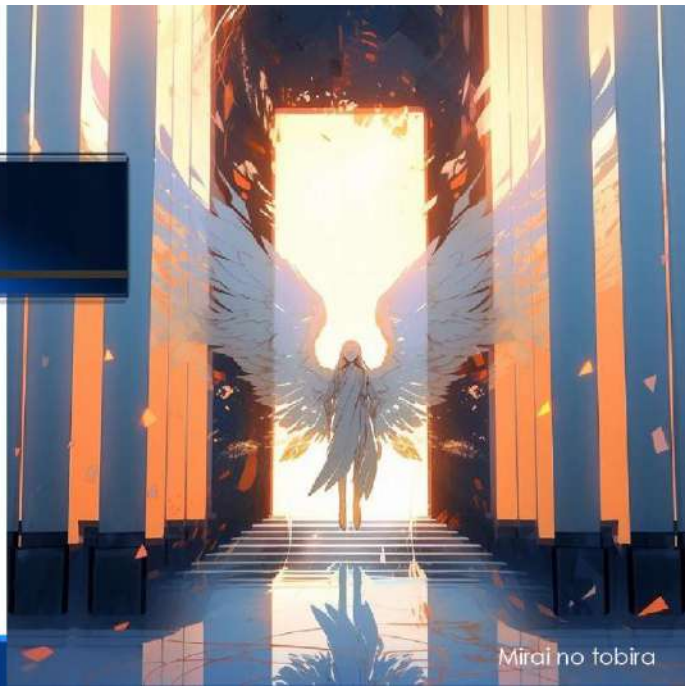
ＩｏＴ技術応用_実習２

導入編

IoT技術応用

「基礎から実践へ
3つの実習で学ぶIoTシステム構築」

実習1： デバイス制御
☞ 実習2： クラウド接続
実習3： システム統合



実習2 学習のゴール

IoT技術応用

- 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
- 「データがインターネットを通じて世界とつながる」
  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
  これが、**IoTの真の価値**



習得するスキル

IoT技術応用

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

- **「IoTのつながる力」センサー、インターネット、クラウド**
  この3つが組み合わさることで、私たちの生活を便利にする
  さまざまなサービスが生まれています



実習2 クラウド接続

IoT技術応用

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

- **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とシリアル通信**
  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

使用する機材・サービス

IoT技術応用

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 or Thingspeak**
  クラウドIoTサービス
  - **Ambient**
    日本語対応、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

実習ステップ

IoT技術応用

1. **センサーで測定**
   DHT22が温度と湿度を測定します
2. **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
3. **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
4. **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
5. **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

実習の進め方

IoT技術応用

ワークブックの構成

- ▶ **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

実習成功のポイント

IoT技術応用

💡 **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」を常に考える

1．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、原因を探る

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

3．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する

実習２ クラウド接続

IoT技術応用

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

- **Part１：Wi-Fi接続とネットワーク基礎**
  ESP32の基本操作／Wi-Fi接続の実装／接続状態の確認
- **Part２：センサーデータ取得とシリアル通信**
  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
- **Part３：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／送信確認とトラブル対応
- **Part４：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

実社会とのつながり

IoT技術応用



すでに世界中で使われている、実用的な技術


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

ＩｏＴ技術応用_実習２

Part1

IoT技術応用

「基礎から実践へ
3つの実習で学ぶIoTシステム構築」

実習１ デバイス制御
実習２ クラウド接続
実習３ システム統合



IoT技術応用

実習２ クラウド接続

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



実習２ クラウド接続

IoT技術応用

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

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

Part１：Wi-Fi接続とネットワーク基礎

実習２ クラウド接続

ESP32 でインターネットにつながろう





Part１：Wi-Fi接続とネットワーク基礎 の達成目標

実習２ クラウド接続

実習１の「スタンドアロン」から進化し
デバイスを「ネットワーク」に繋げます

- ✓ ESP32 マイコンの基本的な使い方がわかる
- ✓ ESP32 を Wi-Fiに接続できる
- ✓ IPアドレスや電波強度（RSSI）を確認できる
- ✓ シリアルモニタで接続状態をデバッグできる

ESP32 の特徴

実習２ クラウド接続

実習１で使った Arduino Uno から
より高機能な ESP32 に変わります

機能	Arduino Uno (実習1)	ESP32 (実習2)
Wi-Fi	✗ なし	✓ あり (2.4GHz帯)
Bluetooth	✗ なし	✓ あり
動作周波数	16MHz	✓ 最大240MHｚ
メモリ	2KB	✓ 520KB
性能（CPU/メモリ）	標準	✓ 非常に高い

Wi-Fi通信の仕組み

実習 2 クラウド接続 Part1

Wi-Fi に接続する３つのステップ

ステップ１：アクセスポイントの検索
周囲の Wi-Fiネットワークを探す
各ネットワークには SSID という名前がついている

ステップ２：認証
SSID とパスワードを使ってログインする

ステップ３：IPアドレスの取得
DHCPで自動的にIPアドレスがふられる

開発環境準備

実習 2 クラウド接続 Part1

Arduino IDE に、ESP32 を扱うための設定を追加

1. **ボードマネージャの URL追加**（すでにインストールされている場合は不要）
「ファイル」→「基本設定」「追加のボードマネージャ」で ESP32用の URLを追加
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
2. **ボードのインストール**
「ボードマネージャ」で「esp32 by Espressif Systems」をインストール
3. **ボードの選択**
「ツール」→「ボード」→「ESP32 Dev Module」を選択
4. **ポートの選択**
ESP32 を接続し、対応する COMポートを選択

⚠ ポートが表示されない場合ワークブックの指示に従い、インストールしてください

Wi-Fi 接続プログラム

実習 2 クラウド接続 Part1

ESP32 を Wi-Fi に接続するための基本コード

```
#include <WiFi.h>
const char* ssid = "あなたのSSID";
const char* password = "あなたのパスワード";
void setup() {
 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");
  WiFi.reconnect();
 }
}
```


SSIDとパスワード

実習 2 クラウド接続 Part1

プログラムが動作するために
自分の環境に合わせて2ヶ所を書き換える必要

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

1. **SSID：接続したいWi-Fiネットワークの「名前」**
   大文字・小文字も正確に
2. **パスワード：そのWi-Fiのパスワード**

注意
ESP32は 2.4GHz帯のWi-Fiにのみ対応、5GHz帯の SSIDでは接続できません

動作確認

実習２ クラウド接続 Part1

ESP32 にプログラムを書き込みます

1. ESP32 をPCに接続する
2. シリアルモニタを開く
3. ボーレートを「115200」に設定する
4. ESP32 のリセットボタンを押す

「接続中.....」の後に「✓ Wi-Fi接続成功！」と表示されればOK

実習２ Part１のまとめ

実習２ クラウド接続

- ✓ ESP32 が Wi-Fi/Bluetooth 内蔵の強力なマイコンであることを学んだ
- ✓ Arduino IDE で ESP32 ボードを設定する方法を学んだ
- ✓ WiFi.begin() でWi-Fiに接続する方法を習得した
- ✓ WiFi.status() で接続状態を確認できるようになった
- ✓ IPアドレス、MACアドレス、RSSI の意味を理解した

NEXT

実習２ Part２：センサーデータの取得

実習2： クラウド接続

実習２ Part２では **センサー**を接続します
実習１より**高精度な DHT22** 温湿度センサーを使い
測定したデータを**シリアルモニタ**で確認します

NEXT

IoT技術応用

実習２ クラウド接続

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



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

ＩｏＴ技術応用教材資料 1

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

●本書の内容を無断で転記、掲載することは禁じます。