（ver.24.07.23）

安価な栽培環境測定装置の作成マニュアル

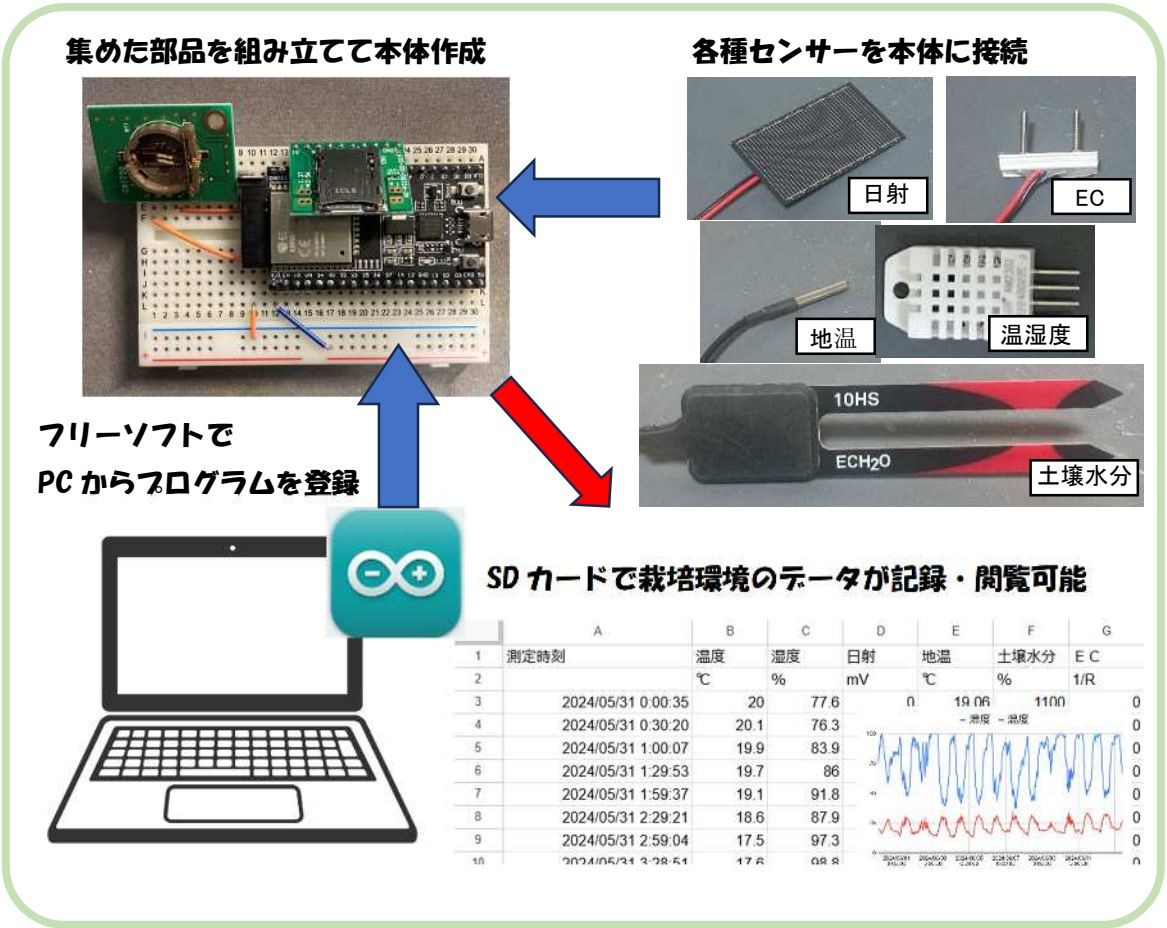
（実践版）

作物生産性の向上には、栽培環境を作物の生育に適した状態に整えることが重要です。

施設園芸では高度な栽培管理のために、施設内の温度や湿度、土壌の温度や水分などを各種センサーで測定して栽培に活用してきました。しかし、測定装置の導入コストの高さや測定点数が限られることから、露地栽培をはじめ導入が難しい現場も多くあります。

一方で、近年では「ものとインターネットを繋ぐ」IoT 技術の発展に伴い、安価で高性能な IoT 端末が広く流通し、プログラム技術も一般に広まってきたことから「安価に栽培環境を測定する機器」を「自作する」ことが可能になってきました。

本マニュアルでは、そんな IoT 端末を活用して、低コストで省電力かつ、各種のセンサーに対応する栽培環境測定装置を作成する方法をまとめました。



活用方法の例

- 導入コストが高く導入できなかったほ場のモニタリングに
- ほ場内のばらつきが大きいほ場の多点モニタリングに
- これまでデータ化してこなかった項目の見える化に
- 電源がなく導入ができなかった露地ほ場などのモニタリングに
- 複数の農地の栽培環境の比較検証に
- 施肥方法の検討など、多数のデータを収集したい場合に
- 後継者に技術をデータとして伝承するための基礎データ収集に

など

安価で自作も可能な装置だからこそ、アイデア次第で様々な活用方法が実現できます！

初めて電子機器を扱う人のために、本マニュアルとは別に最低限の材料と工程で温湿度の測定のみを行う装置を作成する入門版のマニュアルを用意しています。

入門版マニュアルで使用するソフトウェアや部品は本マニュアルでも活用できます。まず簡単な測定を試してみたいという方は、一度入門版マニュアルで試作することをお勧めします。

目次

〇著者・発行者
著者：愛知県農業総合試験場　環境基盤研究部　環境安全研究室
発行者：愛知県農業総合試験場

〇問い合わせ先
愛知県農業総合試験場
環境基盤研究部　環境安全研究室（山本）
〒480-1103 愛知県長久手市岩作三ケ峯 1-1
TEL：0561-41-9511（ダイヤルイン）
E メール：nososi@pref.aichi.lg.jp
https://www.pref.aichi.jp/site/nososi/

免責事項

本マニュアルを用いて作成する栽培環境測定装置および、配布するプログラムの利用又は利用不能で生じた直接又は間接的損害について、一切の責任は負わない。

愛知県農業総合試験場は、配布するプログラムに不具合やエラーや障害が生じないことを保証しない。

愛知県農業総合試験場は、配布プログラムに欠陥があると判明した場合、訂正や補修をする義務を負わない。

はじめに

本マニュアルで作成する栽培環境測定装置は、安価に流通する IoT 端末と各種センサーを使用することで、低コストでの栽培環境測定と、データの記録が可能です。

装置は 10 ㎝四方程度の容器に収まり、省電力設計のため乾電池（単三乾電池 4 本）で 1 か月以上使用できます。センサーを除いた 1 台当たりのコストは 5000 円程度と、複数導入しやすく、ほ場間や、ほ場内によるばらつきなどを測定するのに向いています。

材料から自作するため、配線やプログラムの変更で必要なセンサーだけを選択することや、同じセンサーを複数同時接続するなど、自由度の高い改良も可能です。

なお、本マニュアルでは下記のステップにより温湿度、地温、土壌水分、EC、日射量を 1 系統ずつ測定する仕様を紹介します。

ステップ１　装置の材料の入手
ステップ２　プログラム用ソフトウェア入手・インストール
ステップ３　本体の組立て（測定装置本体の完成）
ステップ４　プログラムの書き込み
ステップ５　センサーの準備と接続
ステップ６　プログラムの調整（完成）

本マニュアルで作成する栽培環境測定装置の基本的な性能
〇データの記録方法：マイクロ SD カードに経時的に保存
（csv 形式のデータとして保存）
〇動作電源：単三乾電池４本（直列で 5V 以上を確保する必要があります）
〇測定可能な項目：温湿度、温度（地温水温）、土壌水分、日射量*、EC*など
*日射量と EC については、後述する自作の簡易センサーの使用を想定

ステップ1

装置の材料の入手

○販売店の一例

本マニュアルで作成する環境測定装置では、いくつかの専門的な電子部品を材料とします。これらの電子部品は、各電子部品販売店のオンラインショップで購入できます。代表的な販売店として以下の店舗を紹介します。

（URL は令和６年６月時点確認のものとなります）

① 秋月電子通商　URL https://akizukidenshi.com/catalog/default.aspx
② スイッチサイエンス　URL https://www.switch-science.com/
③ マルツオンライン　URL https://www.marutsu.co.jp/
④ 共立エレショップ　URL https://eleshop.jp/shop/
⑤ Digkey 日本　URL: https://www.digikey.jp/
⑥ マウザーエレクトロニクス（日本）　URL: https://mouser.jp/

その他、一部の部品については、Amazon 等の通販サイトからも入手可能なほか、家電量販店やホームセンター、100 円ショップ等で購入できる材料も使用します。

使用する材料のほとんどは①の秋月電子通商で入手可能なため、本マニュアルでは基本的に秋月電子通商での材料の調達を想定しています。

次ページに、用途ごとに各部品・材料の販売先 URL 等を記載しています。部品の種類が多いため必要な部品を確認しながら購入してください。各 URL のリンクから、販売ホームページに移動できます。(キーボードの ctrl を押しながらクリック)

（一部のセンサーは高額なものを使用しています。装置はセンサーが 1 種類だけでも動きますので、用途に合わせて準備してください。）

○各部品の販売 URL 等

○本体の作成に必要な部品

数量	部品名	備考、URL 等
1	ESP32DevkitC	https://akizukidenshi.com/catalog/g/g115673/
1	ブレッドボード	https://akizukidenshi.com/catalog/g/g112366/
1	ジャンパー線	https://akizukidenshi.com/catalog/g/g102315/
1	ジャンパーワイヤ	https://akizukidenshi.com/catalog/g/g105159/
1	マイクロ SD カードリーダー	https://akizukidenshi.com/catalog/g/g105488/
1	RTC モジュール	https://akizukidenshi.com/catalog/g/g115488/
1	パワーmosfet	https://akizukidenshi.com/catalog/g/g107597/
1	電池ボックス	https://akizukidenshi.com/catalog/g/g100311/
1	マイクロ SD カード *1	https://akizukidenshi.com/catalog/g/g129380/
1	USBtypeB ケーブル（データ通信用）*2	https://akizukidenshi.com/catalog/g/g107607/

*1：マイクロ SD カードについては容量が６４GB 以上のものは使用できません。なお、測定結果のデータ量は少ないので、８GB か１６GB のもので問題ありません。

*2：USBtypeB ケーブルは必ず『通信用』を用意してください。『電源用』は使用できません。

○各種センサー

数量	部品名	備考、URL 等
1	温湿度センサー	https://akizukidenshi.com/catalog/g/g107002/
1	地温センサー*1	https://www.mouser.jp/ProductDetail/DFRobot/DFR0198?qs=Zcin8yvlhnPOt9ZkO3roFQ%3D%3D
1	土壌水分センサー *2	https://daiki.ocnk.net/product/280
1	日射センサー用ソーラーパネル	https://akizukidenshi.com/catalog/g/g116017
1	EC センサー用電極	自作します

*1：地温センサーは秋月電子通商では取り扱いがないため、マウザーエレクトロニクス（日本）の販売ページを使用しています。

*2：土壌水分センサーは大起理科工業が取り扱う METER 社の１０HS センサーを使用しています。低価格なセンサーは、施肥などの影響を強く受けるため、正確な測定には活用できません。

※参考（低価格な水分センサー）　https://akizukidenshi.com/catalog/g/g113550/

〇その他必要な部品

●抵抗器

数量	部品名	備考、URL 等
1(100)	4.7kω1/6W 抵抗	https://akizukidenshi.com/catalog/g/g116472/
1(100)	510ω1/4W 抵抗	https://akizukidenshi.com/catalog/g/g108533/
1(100)	22ω1W 抵抗	https://akizukidenshi.com/catalog/g/g108808/

各 100 本入りでの販売になります。

ωとｋωを間違えないように注意してください。

22ω抵抗は W 数が小さいと発熱するので、必ず１W を選択してください。

●コネクタ部品

数量	部品名	備考、URL 等
1(100)	XH コネクタ用 コンタクト	https://akizukidenshi.com/catalog/g/g112265/
１０	2P ハウジング	https://akizukidenshi.com/catalog/g/g117146/
１０	3P ハウジング	https://akizukidenshi.com/catalog/g/g112256/
１０	4P ハウジング	https://akizukidenshi.com/catalog/g/g112257/
１０	2P ポスト	https://akizukidenshi.com/catalog/g/g117150/
１０	3P ポスト	https://akizukidenshi.com/catalog/g/g117151/

センサー類の接続やケーブルの延長に使用します。

コンタクトは 100 個セット、ハウジングとポストは単品売りです。ハウジングとポストについては 10 個ずつ程度を確保すると余裕があります。

（ハウジングについては、温湿度センサーに４P を、地温・土壌水分に３P を、EC・日射・電源に２P を使用します。）

●自作 EC センサー電極用部品

数量	部品名	備考、URL 等
２本 以上	ステンレスねじ M3×２０mm	ホームセンター等で購入（ナット付き６本セットなどで購入）
1	ケーブルカバー （ケーブルモール）	ホームセンター等で購入 （３㎜径のねじのねじ山が内側に収まるもの）

EC センサーに使用する電極はサビないようにステンレスねじを使用します。測定は抵抗器と組み合わせて使用しますが、ねじの長さや太さで使用する抵抗器が変わるため指定の規格のねじを使用してください。

●センサー等の延長ケーブル

数量	部品名	備考、URL 等
任意	２芯ケーブル	ホームセンター等で購入 AWG22～28 程度のキャブタイヤケーブルを推奨
任意	３芯ケーブル	

ケーブルは使用する長さに合わせて、長めに購入してください。

●その他消耗品（ホームセンター、家電量販店等で購入してください。）

数量	部品名	備考、URL 等
任意	熱収縮チューブ	ケーブルの延長などで被覆に使用します。 必要に応じて購入してください。 内径 1.5 ㎜、６㎜、10 ㎜があるとよいです。
1	糸はんだ	必ずヤニ（フラックス）入りを選んでください。 慣れないうちは細いものの方が扱いやすいです。
1	ステンレス用糸はんだ	自作 EC センサーを作成する際、ステンレスねじのはんだ付けに必要です。
1	ステンレス用はんだフラックス	ステンレスへのはんだ付けに必要です。（専用のフラックスになります）
任意	基板用シリコンスプレー	基盤の防水加工が必要な場合に使用します。 シリコン“オイル”スプレーは使用できません。
任意	防水パテ	密閉用に使用します。ホームセンターで扱いのあるエアコンパテで問題ありません。
任意	両面テープ	部品の固定などで利用できます。
任意	マニキュア、瞬間接着剤など	絶縁性があり、乾燥が比較的早く、塗膜が丈夫なため、電線や回路の被覆に利用できます。

●工具類

数量	部品名	備考、URL 等
1	ペンチ もしくはニッパー	配線用のピンやケーブルを切断するのに使用します。 細かい作業があるため、先が細く、ある程度太い針金も切れるものが良いです。
1	はんだごて	一部の部品にはんだ付けが必要になります。
1	ホットボンド	各種部品の固定、補強、防水等に使用できます。
1	圧着ペンチ	https://akizukidenshi.com/catalog/g/g114357/ コネクタを固定するのに使用します。
任意	ワイヤストリッパ	https://akizukidenshi.com/catalog/g/g115131/ ケーブル類の被覆をはがすのに便利です。

圧着ペンチは専用のものを用意したほうが、安定してコネクタを作成できます。

ワイヤストリッパはカッターナイフ等でも代用できます。

●外装

100 円均一のタッパー等にドリルで穴をあけても対応可能ですが、一例として電気ケーブル用のアウトレットボックス(日動電工)を紹介します。

数量	部品名	備考、URL 等
1	アウトレットボックス 40B4AP	ホームセンター等で購入。
1	２号コネクタ 2k16	ホームセンター等で購入。
1	16 ㎜径塩ビ管	ホームセンター等で購入。

アウトレットボックス、２ 号コネクタ、16 ㎜径塩ビ管を使用することで、塩ビ管をさした場所に２号コネクタ経由で自由に設置ができ、設置が容易になります。

ステップ２

プログラム用ソフトウェア入手・インストール

自作モニタリング装置では、無料で入手可能で、扱いが容易な「Arduino IDE」をプログラム用のソフトウェアとして使用します。IDE とは統合開発環境のことを指しており、インターネット環境下ではこのソフトウェアーつでプログラムの作成から書き込みまで対応可能です。

Arduino IDE は本来、Arduino という電子工作の教材に使用されるソフトウェアですが、ESP32 シリーズでも使用できます。

○ソフトウェアの入手（ダウンロード～インストール）

Arduino IDE は Arduino Foundation のホームページからダウンロードできます。

URL　https://www.arduino.cc/en/software

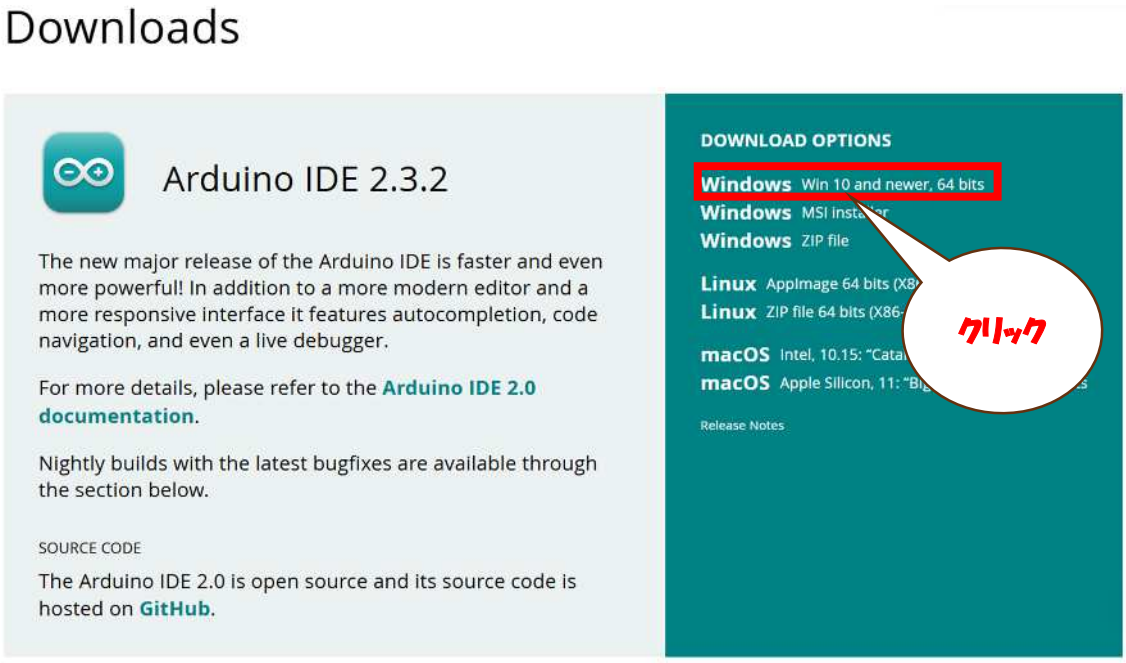


図１．ArduinoIDE のダウンロードページ

URL を開くと図のページに移動します。(図１)

令和６年６月時点での最新バージョンは２.３.２です。「**Windows**　Win 10 and newere 64bits」をクリックしてください。

続いて寄付の募集、メールアドレスの登録などが表示されますが、赤枠で囲った

JUST DOWNLOAD

を選択して登録等を行わずにダウンロードできます。（図２）

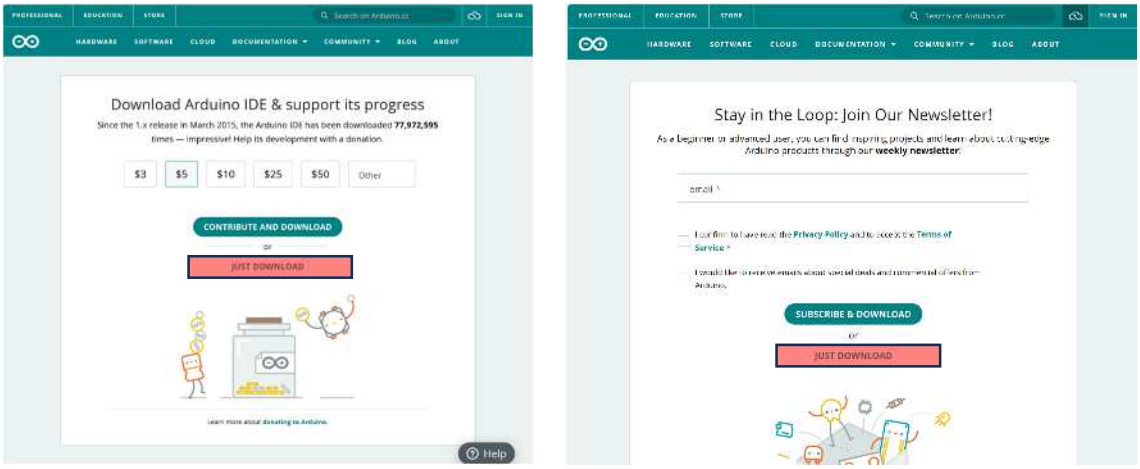


図２．支援金募集ページ（左）とニュースレター登録ページ（右）

その後、「arduino-ide_2.x.x_Windows_64bit.exe」がダウンロードされます（x にはバージョン情報が入ります）。セキュリティで「デバイスに問題を起こす可能性があります。」と表示されることありますが、保存してください。

保存場所は WindowsPC の場合、特に指定が無ければユーザー名フォルダ内のダウンロードフォルダに保存されます。

ダウンロードされたファイルを実行すると、Arduino ID セットアップウィザードが起動します。こちらでもセキュリティ警告が出ることがありますが、実行してください。

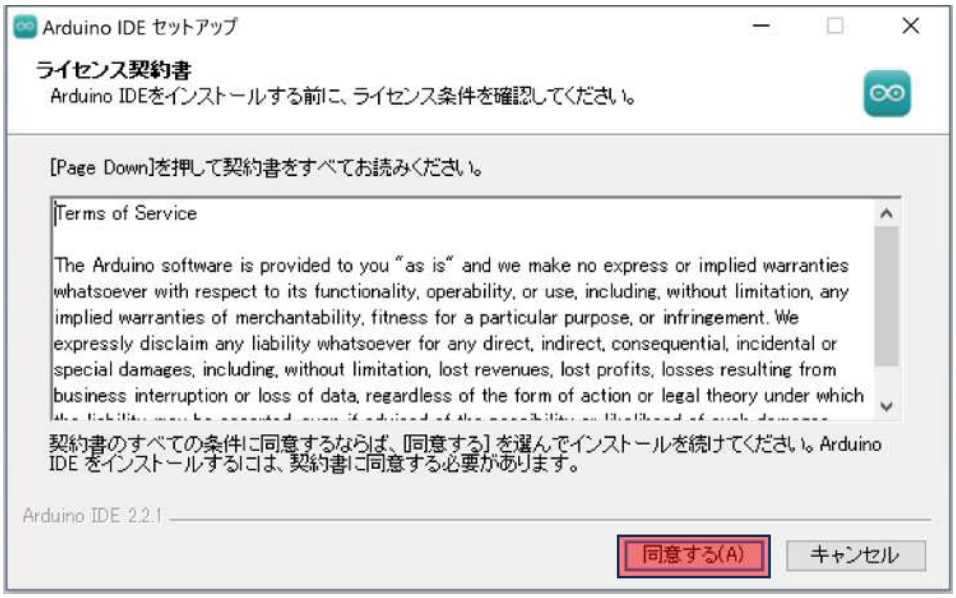


図３．ライセンス契約（セットアップ）

契約書内容を確認の上、「同意する」を選択してください。（ソフトウェアの提供者（Arduino）は使用に当たって一切の保証も責任も負わないとする内容です）

引き続き、どのユーザーにインストールするかの指定（図４）が表示されます。

基本的に初期状態のまま「次へ」を選択します。ただし、**ユーザー名に半角英数字記号以外が使われている場合は正常に機能しない**ので、その場合は必ず「すべてのユーザー用にインストールする」を選んでください。

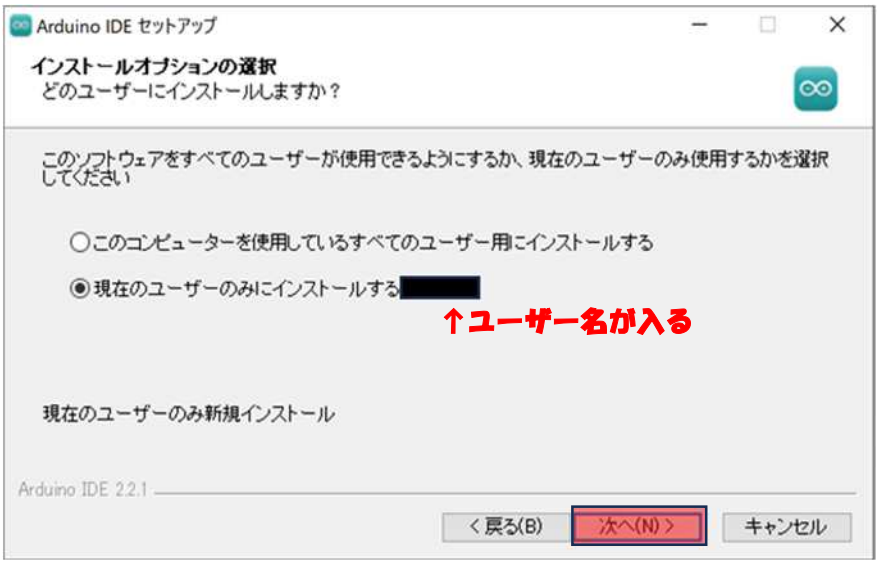


図４．インストールするユーザーの設定

続いて、インストール先のフォルダを指定します。初期状態で C ドライブが指定されますが、他の場所でも特に問題はありません。必要がなければ初期状態のままで「インストール」を選択してください。

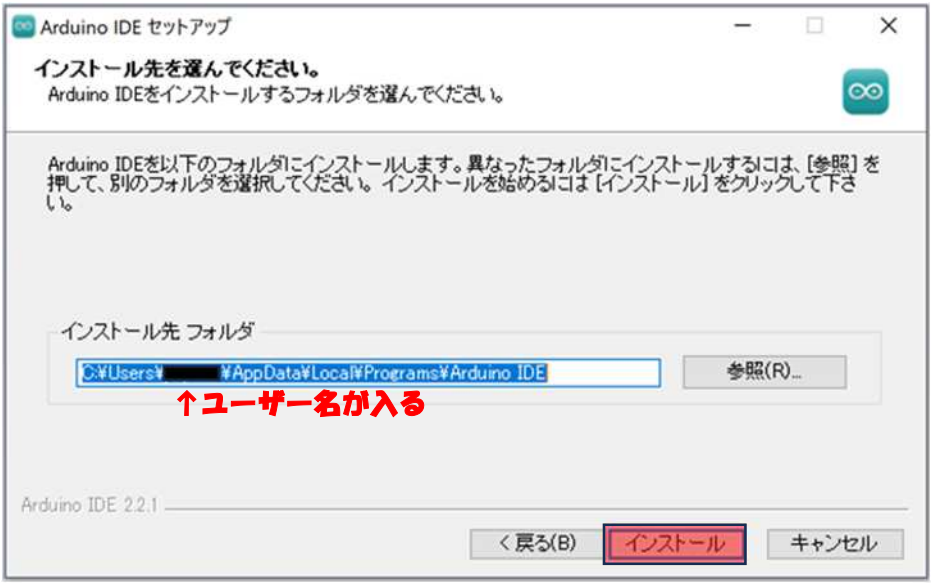


図５．インストール先の選択

インストールが終了するまで待ちます。

図６．インストールの完了

図６の画面が表示されたらインストール完了です。

Arduino IDE を実行にチェックが付いているのを確認して「完了」を選択してください。ソフトウェアが自動で起動します。

○Arduino IDE の初期設定

Arduino IDE はインターネット上からサンプルプログラムの収集や、アップデートを行うことが可能です。初回起動時にはセキュリティから接続の許可を求められることがありますが、特に問題が無ければ「許可」してください。

続けて、Arduino IDE の初期設定を行います。

図７．起動画面

起動時にインターネット上からサンプルプログラム等のリストのダウンロードを求められることがあります。必要なサンプルを入手するために「許可」してください。

初期設定では、左上の「File」をクリックして「Preference」を選択します。

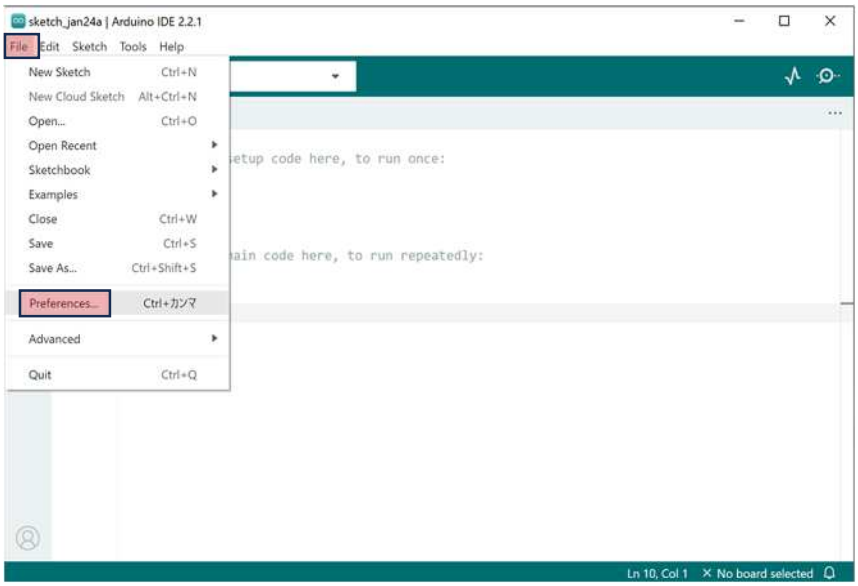


図８．初期設定

Preference をクリックすると設定画面が表示されるので、使用言語を日本語にします。

初期設定は英語になっていますので、Language 欄の English になっている部分を選択し、日本語に変更して OK をクリックします。

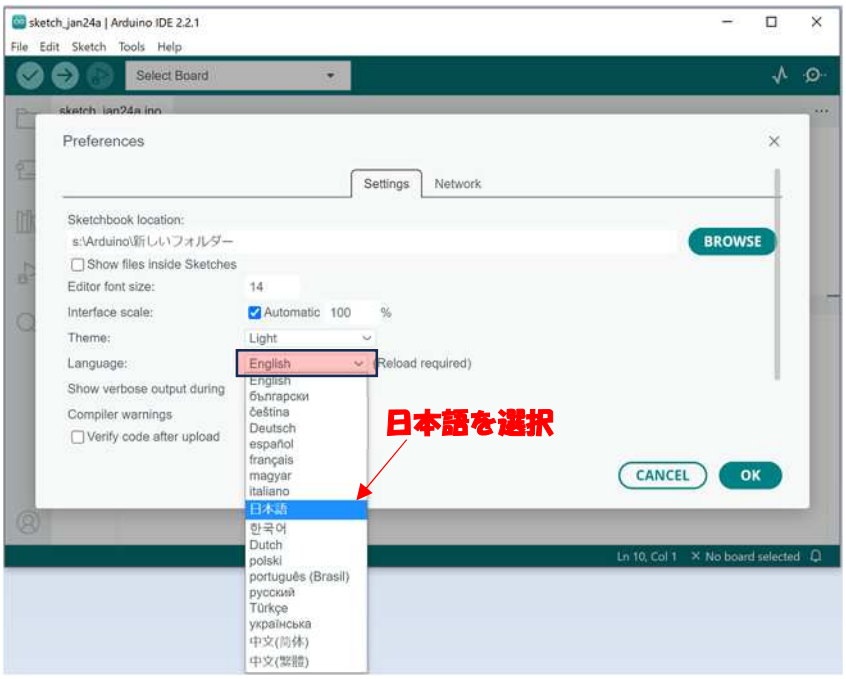


図９．日本語化

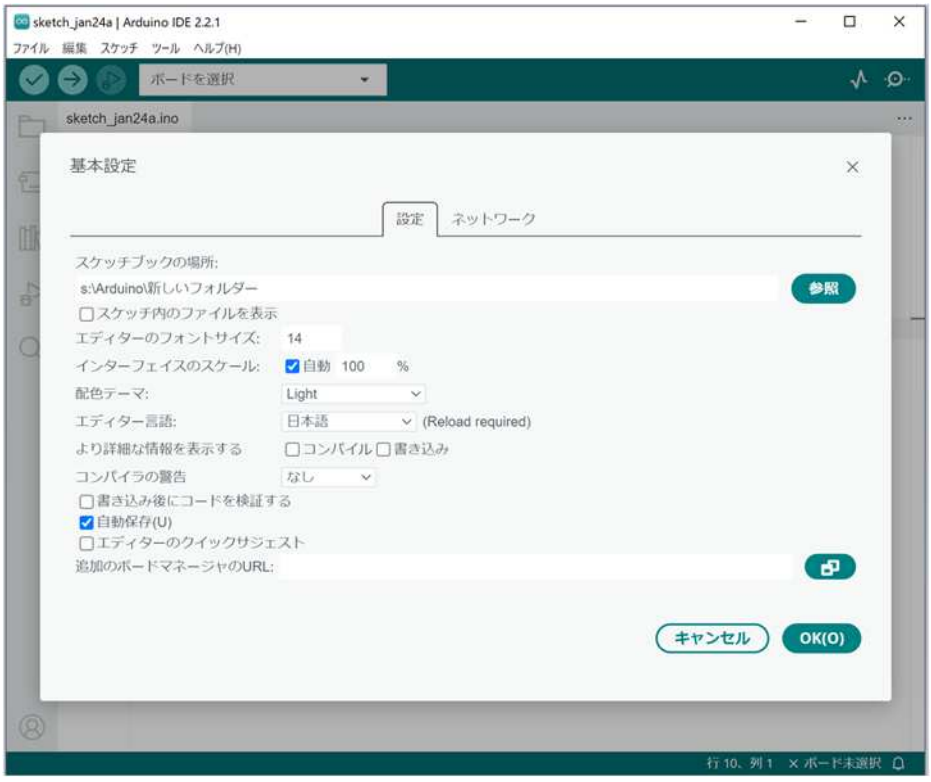


図１０．日本語化した基本設定画面

日本語化が完了すると、図のような表記になります。（図１０）

スケッチブックの場所で指定したフォルダ（図では s ドライブ）に作成したプログラムが保存されていきます。（保存先は右側の参照から変更できます。）

エディターのフォントサイズはプログラム中の文字の大きさを変更できます。

自動保存にチェックが入っていると、プログラムの検証・書き込みを行うたびにプログラムを上書き保存します。自分でプログラムの内容を変更するとき注意してください。

○Arduino IDE への ESP32 の登録

今回使用する IoT 端末（ESP32）を Arduino IDE で使用するためには、ESP32 を登録する必要があります。

次の図１１を参考に登録をしてください。

図１１．ボードマネージャに ESP32 を登録

ArduinoIDE の画面で、左端にあるアイコンのうち、上から２番目のアイコンをクリックします。

画面左側にボードマネージャが展開するので、ボードマネージャと書かれた下の検索枠（最初は「検索をフィルタ」と記載されています）に「ESP32」と入力します。

インストールする候補が表示されるので、「esp32 by Espressif Systems」を探してインストールをクリックします。（令和６年６月時点での最新バージョンは 3.0.1 ですが、最新版は情報が少なく動作不良などが起きる可能性があるため 2.0.17 を推奨します。）

インストールができたら、実際に ESP32 がボードマネージャで登録されたかを確認します。

画面上の「ボードを選択」という枠をクリックしてください。

「不明 COM１」や「他のボードとポートを選択」といった情報が表示されるので「他のボードとポートを選択」をクリックします。（この時点では ESP32 は PC と接続していません。）

図１２．ボードを選択

ボードを検索の枠の下に、候補が並びますが、この中に ESP32 の表現があれば無事にインストールできています。今回は ESP32 Dev Module を使用するので、検索欄に「esp32 dev」と入力してリストの中に ESP32 Dev Modulre があることを確認してください。

図１３．ボードの登録確認

ボードマネージャの登録が確認出来たら Arduino IDE の登録は終了です。

Arduino IDE に ESP32 が登録されたことで、今後 Arduino IDE を使用してプログラムを ESP32 に書き込むことができるようになります。

ボードマネージャの登録がうまくいかないときは、

画面上のファイルタブから基本設定を選び、基本設定の一番下にある追加のボードマネージャの URL の部分に

https://dl.espressif.com/dl/package_esp32_index.json

と入力してください。（旧バージョン時の対応です）

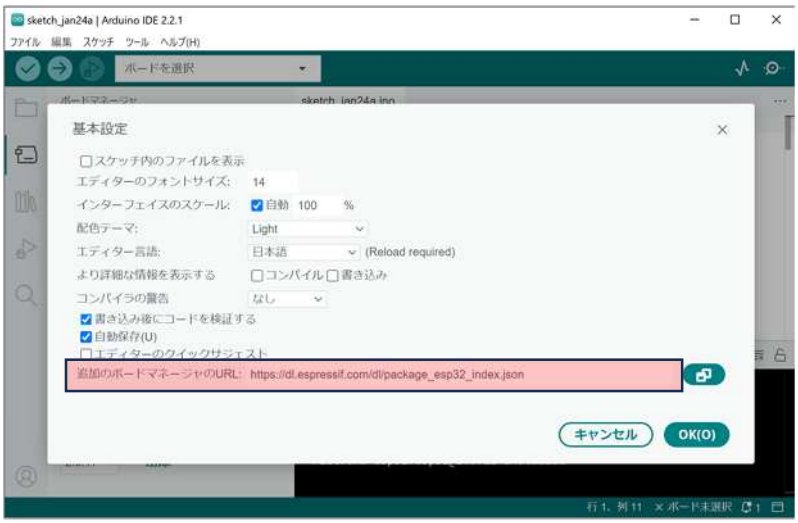


図１４．ボードマネージャの登録がうまくいかないときの対応

ステップ3

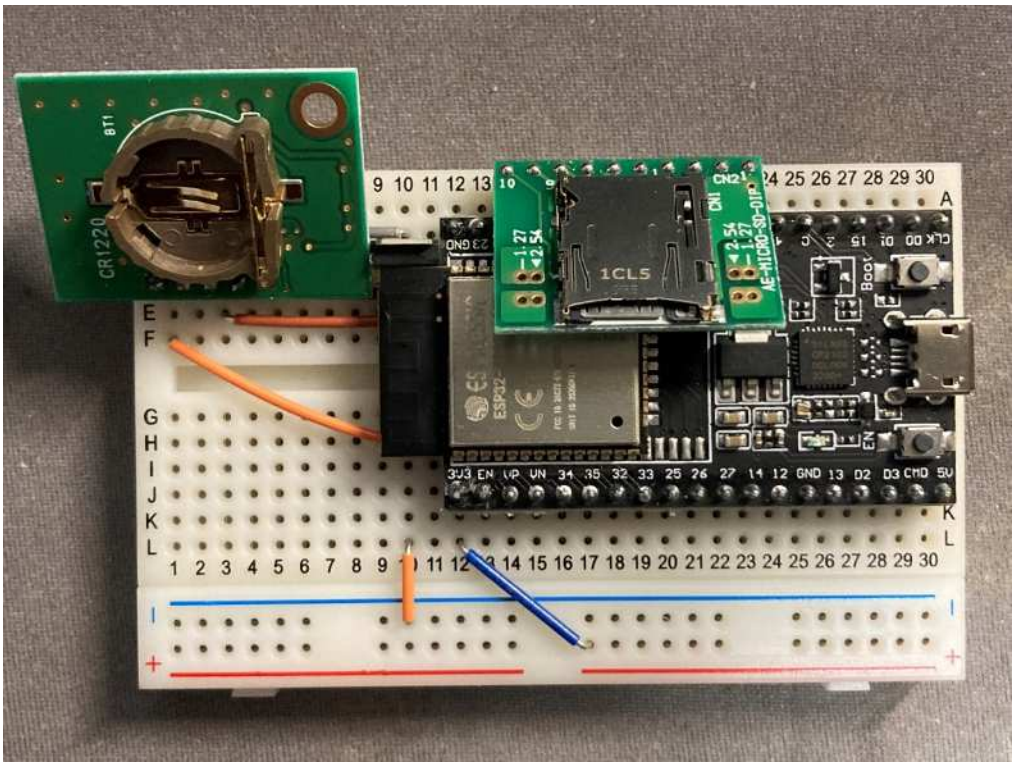
本体の組立て：測定装置本体の完成

プログラム用のソフトウェアの準備ができたら、環境測定装置の本体を組み立てます。

本体は、

・中核となる ESP32DevkitC
・データを保存するためのマイクロ SD カードリーダー
・現時刻を記録するための RTC(リアルタイムクロック)
・消費電力を抑えるためのパワーmosfet２SK4017

の４つの部品をブレッドボード上に配置して、ジャンパー線で配線して完成です。



完成した本体の写真

この本体に各種センサーを接続することで、センサーの測定結果を記録時間と合わせてマイクロ SD カードに保存できるようになります。

一部の部品について、はんだ付けが必要なので、最初に必要な部品のはんだ付けを行います。

○はんだ付けについて

ブレッドボードによる作成は、ピンやジャンパー線を差し込むだけで簡単に回路を組み立てることができますが、接続用のピンが付いていない SD カードリーダーや RTC のような部品ははんだ付けが必要となります。

はんだ付けを始める前に、次の道具と部品の用意をしてください。

- はんだごて（一般的なもので可）
- 糸はんだ（ヤニ：フラックス入りのものを選ぶ。慣れるまでは細い方が扱いやすい。）
- はんだごてのスタンド（短時間なら濡れ雑巾などでも可。）
- ブレッドボード（本体に使用するものを流用。）
- マイクロ SD カードリーダー
- RTC モジュール
- マイクロ SD カードリーダーと RTC モジュールに付属のピン
- ペンチ、ニッパー等

このほかに、マスキングテープやクリップなどがあると部品の固定がしやすくなります。

作業の前に注意事項の確認を！

●はんだごてはスイッチを入れる（コンセントを刺す）と高温になります。作業スペースを十分に広く取り、まずははんだごてを安定して置ける場所を確保してください。専用のスタンドが望ましいですが、短時間なら絞った濡れ雑巾を用意してこて先は浮かせるようにして置くこともできます。

●はんだごて自体スイッチが付いていないものはコンセントを抜いて冷えるまで高温が続きます。コンセントはいつでも抜けるようにして（もしくはスイッチ付きタブを使用する）、しっかり冷えてから動かしてください。

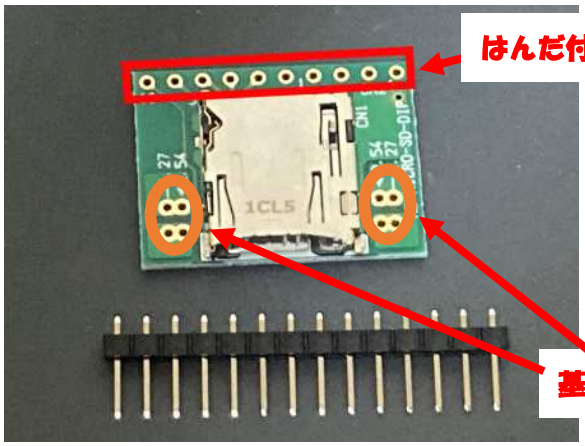
●はんだが溶ける際の煙には、人体に害のある鉛が含まれるので、はんだ付けを行う際は換気の良い場所を選んでください。（鉛を含まない鉛フリーのはんだも流通していますが、鉛入りに比べてやや扱いにくいです。）

○マイクロ SD カードリーダーのはんだ付け

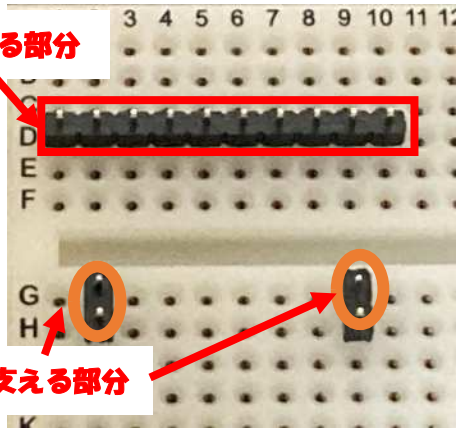
マイクロ SD カードリーダーにピンを 10 本はんだ付けします。

安定しない場所でのはんだ付けは難しいので、カードリーダーに付属する接続ピンは 14 本あるので、10 本、２本、２本に割って右の写真のように、ブレッドボードに設置します。ピンの設置が終わったら、SD カードリーダーを穴にピンが収まるように設置してはんだ付けを行います。

はんだ付けを行う部分は図の赤枠で示した部位ですが、オレンジで囲った部分もピンを。（赤枠以外ははんだ付けしません）



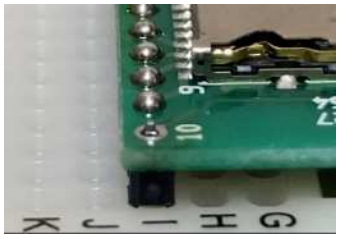
SD カードスロットと付属のピン（１４本）



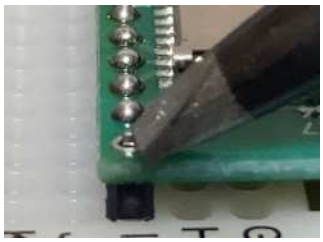
ピンをカットしてブレッドボードに

はんだ付けを行うときは、写真にあるような穴の周りにある金属の輪の部分（ランド）と、その穴を通したピンをはんだでつなぐことになります。

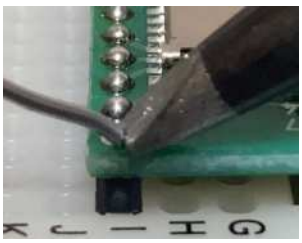
ピンのはんだ付けは手順を守れば比較的容易に行えます。次の写真を参考に、順に操作を行ってください。



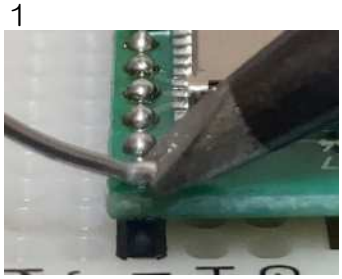
1

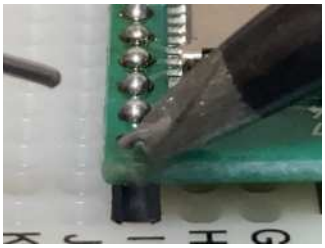


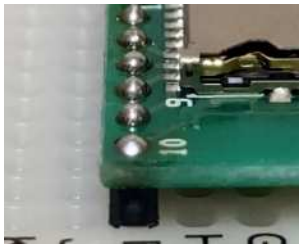
2



3







４　　５　　６

１．ピンがランドを通るように設置する（ピンは若干接続穴から飛び出した状態になる）
２．十分に温まったはんだごてを、はんだ付けする部分のランドとピンにあてがって、『ランドとピンを』温めておく（温度が低いとはんだがしっかり広がらない）
３．糸はんだをはんだごての先に触れさせて溶かす（こて先はランドとピンから外さない、こて先が離れるとはんだが広がらない）
４．糸はんだを足しながら、糸はんだが溶けて広がるのを待つ（はんだをこすりつけるのではなく、溶かして広げるイメージです）
５．はんだごては動かさずに、糸はんだを遠ざける（先にこてを動かすときれいに仕上がりません）
６．数秒待ってからはんだごてを離す

初めてはんだごてを使うときは、すぐにピンのはんだ付けを行わず、十分温まったはんだごてのこて先に、糸はんだをつけてはんだの溶ける感覚を確認することを推奨します。

こて先に付いたはんだは、濡れ雑巾やはんだクリーナーでぬぐい取ってください。こて先が汚れていると、上手にはんだ付けはできません。

はんだは温度が高い状態で広がるので、写真のようにあらかじめはんだ付けを行う部分を温めることが重要です。溶かしたはんだを冷めたピンにつけようとしても、熱されたこて先側に残ってしまうため、上手にはんだ付けはできません。必ずはんだ付けをするピンとランドを温めるようにしてください。

接続場所を温める→はんだを供給する→はんだを離す→こてを離す

の手順ををリズムよく行うと、きれいにはんだ付けができるようになります。

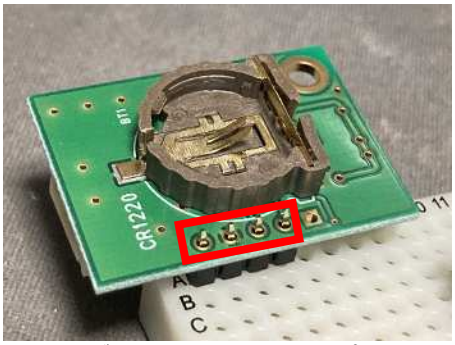
はんだ付けが苦手な人は、はんだ用のフラックスを「ほんの少しだけ」はんだ付けする部分に塗布すると、はんだが良く広がり作業がしやすくなります。

はんだ付けに失敗してしまった場合は、はんだ吸いとり線を使用してはんだを除去します。はんだが溶けにくい場合は、フラックスを別に用意して若干量塗布すると溶けやすくなります。

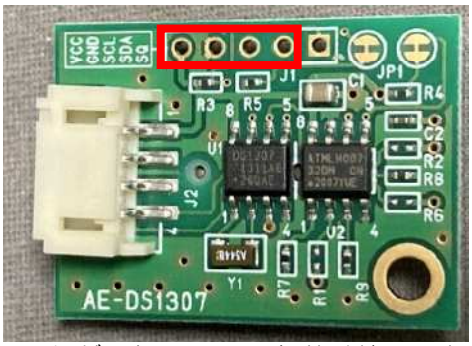
はんだが隣のランドに広がってしまった場合も、同様にはんだ吸いとり線で取り除くか、フラックスを塗布して改めてはんだごてで溶かすことで、ランドごとに分かれてひろがります。

ORTC のはんだ付け

RTC は部品の形状上、ブレッドボードに刺してはんだ付けをする場合は、ピン４本をＡ列の１～４番の穴に差し込み、その部分に重ねて設置します。このとき、写真の一番右側の四角いランド（端子の名前が SQ になっているもの）は使用しないため、ピンは VCC、GND、SCL、SDA の４本のみをはんだ付けします。

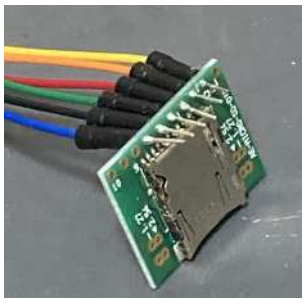


ブレッドボードの A1～4にピンを設置



ランドが四角い SQ の部分は使用しない

マイクロ SD カードリーダーも、RTC も付属のピンを使うと取り回しが悪くなってしまうので、ケーブル状のジャンパー線を使ってはんだ付けしてもよいです。この場合、ブレッドボードから離して接続できるので、SD カードリーダーは抜き差しがしやすくなります。



ピンの代わりにケーブル状のジャンパー線をはんだ付けした場合

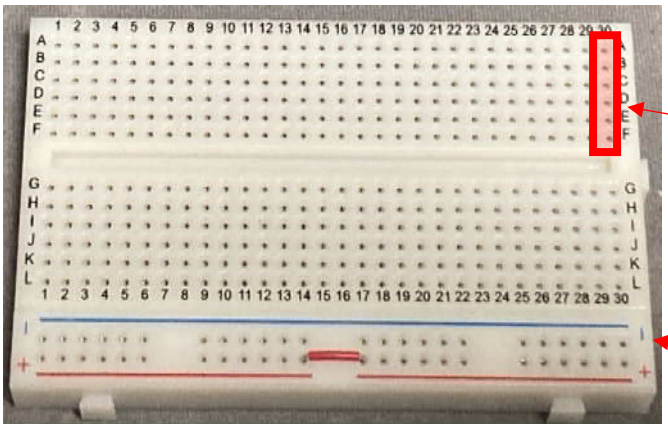
（ケーブルの色を変えることで、どの色がどの機能かわかるようにしています。はんだ付け後、動作に問題がないことが確認出来たら飛び出しているピンはニッパー等で切り取ります。）

マイクロ SD カードリーダーと RTC のはんだ付けが終わったら、本体の組立てを行います。

○ブレッドボードについて

本体の土台には、ブレッドボードを使用します。

ブレッドボードは電子機器の試作に使われる基盤で、決まったルールで穴と穴が電気的につながるようになっているため、穴にピンを差し込むだけで回路を形成できます。



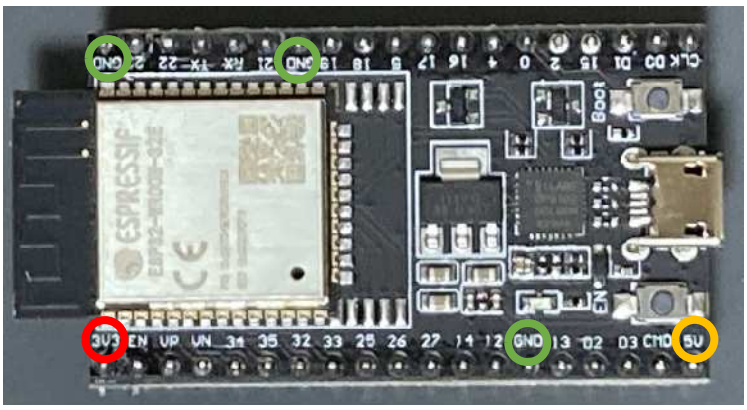
ブレッドボードの表側

手前のマイナス（青）とプラス（赤）のマークのあるエリアは、他のエリアが縦方向につながっているのに対して、横方向に広くつながっています。この部分を利用して測定装置本体やセンサー、SD カードなどに電力を供給できます。（プラスのラインの中央には電源を２種類に分けられるように穴があるため、通常はジャンパー線で接続されています。）

○ESP32DevkitC について

ESP32DevkitC は ESP32 の開発用のキットで、ブレッドボードに接続できるようにあらかじめピンが設置されています。また、ESP32 とPC を接続するための通信部品（シリアルコンバーター）と、電源の電圧を一定に保つための部品（レギュレーター）、USB の差込口などが一通りそろっています。

ESP32DevKitC には、長辺にピンが多く出ており、それぞれのピンには白い文字で番号やアルファベットが記載されています。それぞれの番号やアルファベットはピンの役割を示しています。



今回使用するピンについて簡単に説明します。

●3.3V（赤色で囲った部分）

ESP32 本体を動作させる電圧は 3.3V です。（2.7～3.6V の範囲で動作します）

各種センサーも多くのものが 3.3V で動作します。USB 端子から供給される電圧は５V ですが、電圧を 3.3V に下げる部品（レギュレーター）が設置されているため、適切な電圧が得られるようになっています。

●５V（黄色で囲った部分）

USB から供給される電圧が出力されます。一部の部品が５V で動作することや、電源に乾電池を使用する場合の電源入力端子として使用します。

●GND（緑色で囲った部分）

グランドの意味です。電池のマイナス端子にあたります。各装置は先の５V や 3.3V をプラス端子に、GND にマイナス端子を接続することで電流が流れます。GND ピンは複数ありますが、内部で全てつながっているので、電気的には同じピンになります。

●数字

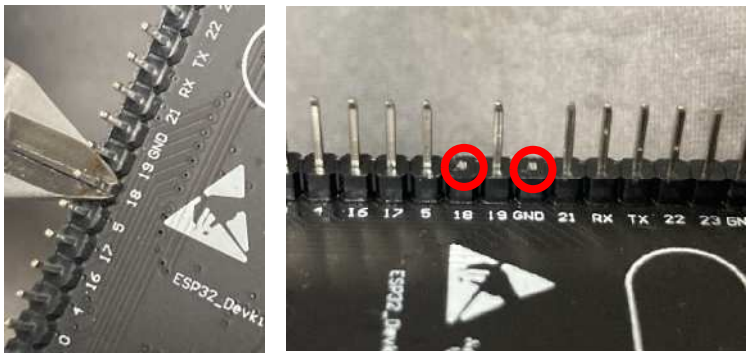
プログラムで任意の役割を持たせることができるピンです。入力と出力に対応しているため、GPIO ピンと呼ばれます。センサーの読み取り用の端子や部品との通信用の端子を接続します。

○本体の組立て

●最初にすること

ESP32DevkitC の 18 番ピンとその二つ隣の GND ピンを付け根の部分で切り取ります。写真のようにニッパーなどで切り取ってください。

この処置により SD カードリーダーの接続を単純にすることができます。



18 番ピンと二つとなりの GND を付け根で切り取り

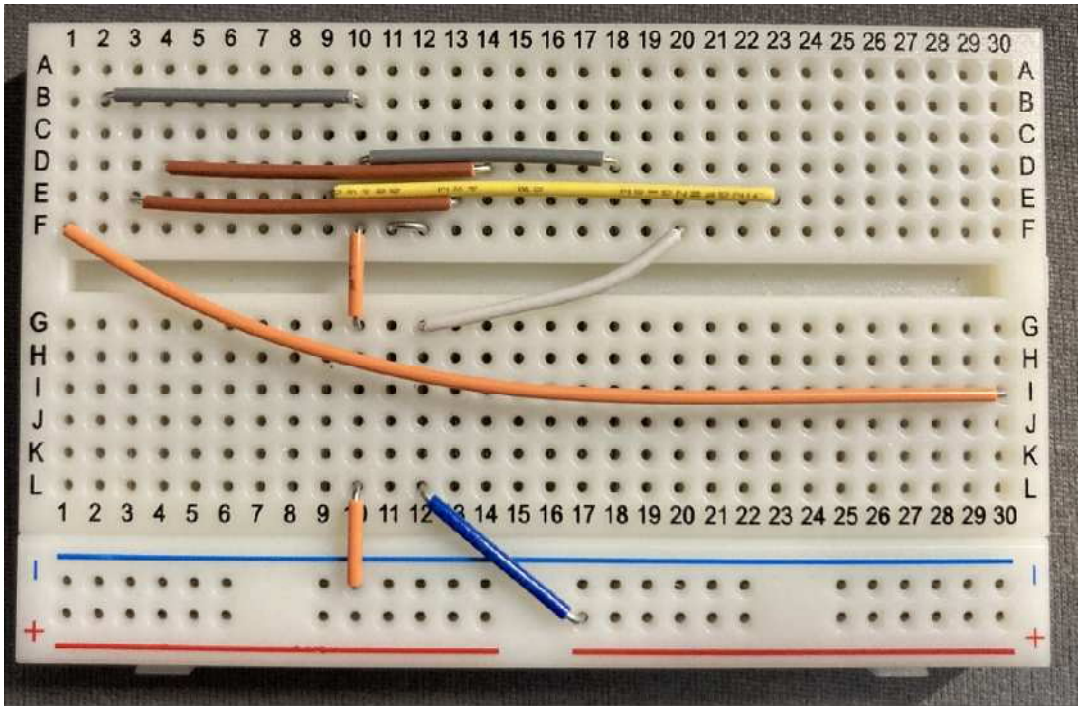
●ブレッドボードにジャンパー線で配線を施す

ESP32 と SD カードリーダー等を接続するために、ジャンパー線を使って配線を行います。

ジャンパー線の設置場所は次の説明と画像を参考にしてください。アルファベットと番号はブレッドボードのものを使用しています。この番号は ESP32 の基盤にプリントされた数字とは異なるので注意してください。括弧内の色はサイズの合うジャンパー線の被覆の色を参考に記載しています。

- G12～F20（白：若干曲げる）
- F11～F12（被覆なし：写真で見えにくいので注意）
- E9～E23（黄色：長い黄色を切って作る）
- F10～G10（オレンジ）
- L10～手前のマイナスのライン（オレンジ）
- L12～右手前のプラスのライン（青）
- F1～I30（長いオレンジを曲げてつなげる）

・B2～B10（グレー）
・D10～D18（グレー）
・D４～D１４（茶：ほかのジャンパー線の間に収まる）
・E3～E13(茶：ほかのジャンパー線の間に収まる)
合計 11 本を接続します。（最初に接続されているプラスのラインの赤は抜きます。）
ジャンパー線の接続間違いがあると、正しく動作しません。特にプラスとマイナスをつなぎ間違えた場合は、部品が発熱する恐れがあるので、注意してください。



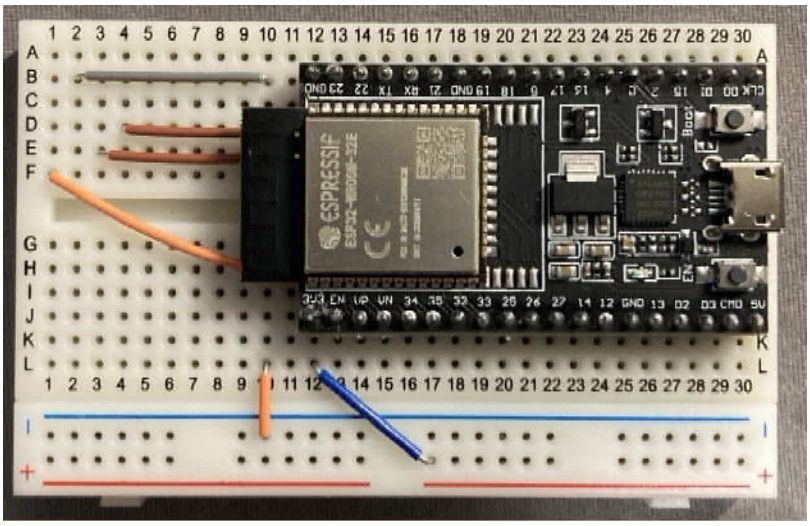
ブレッドボードにあらかじめ必要な配線を施しておく（11 本）

それぞれの配線は、使用している部品を、ESP32DevkitC の各 GOIO ピン、および電源となる 5V、3.3V、GND を接続しています。特に、使用している RTC は５V 以上での動作が必要なので、直接５V 電源を引き出しています。

●部品の設置

OESP32 DevkitC

ブレッドボードへの配線が終わったら、ESP32DevKitC を次の写真のようにブレッドボードのＢ列およびＪ列にの１２～３０番の穴に差し込みます。しっかりと奥まで差し込まないと接触不良になることがあります。（抜き差しはできますが、慎重に抜かないとピンが曲がるので注意してください）

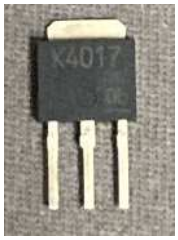


ブレッドボードの右端に寄せて、手前に穴が二列できるように刺します。

○mosfetSK4017

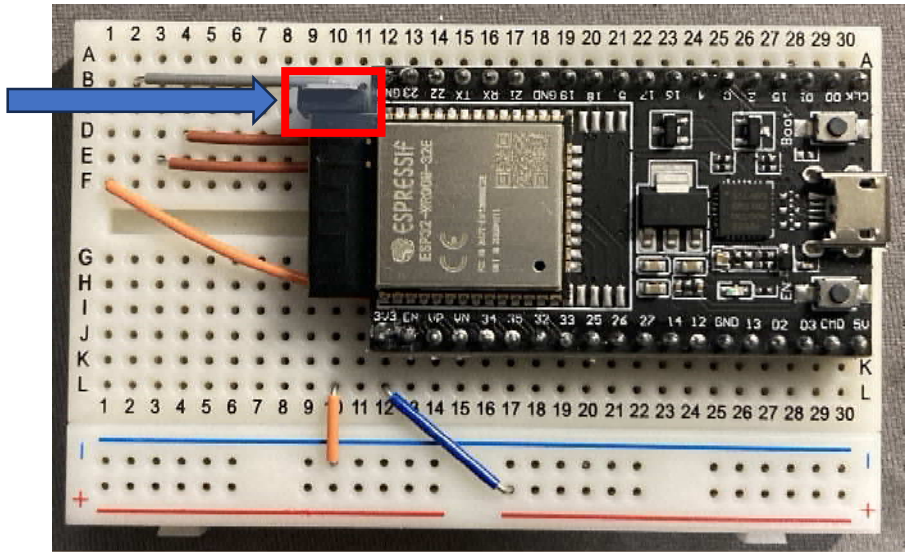
　mosfetSK4017 を印刷されている側（黒い面）を正面として、C9～11 にかけて差し込みます。この部品は若干足が狭いため、しっかりと奥まで差し込みます。

　mosfet はスイッチの役割を果たします。センサーや SD カードは接続しているだけで待機電力が発生しますが、mosfet を使って測定時のみ電流が流れるようにすることで、消費電力を抑えることができます。



文字が見える方が正面

左がＣ９
中央がＣ１０
右がＣ１１

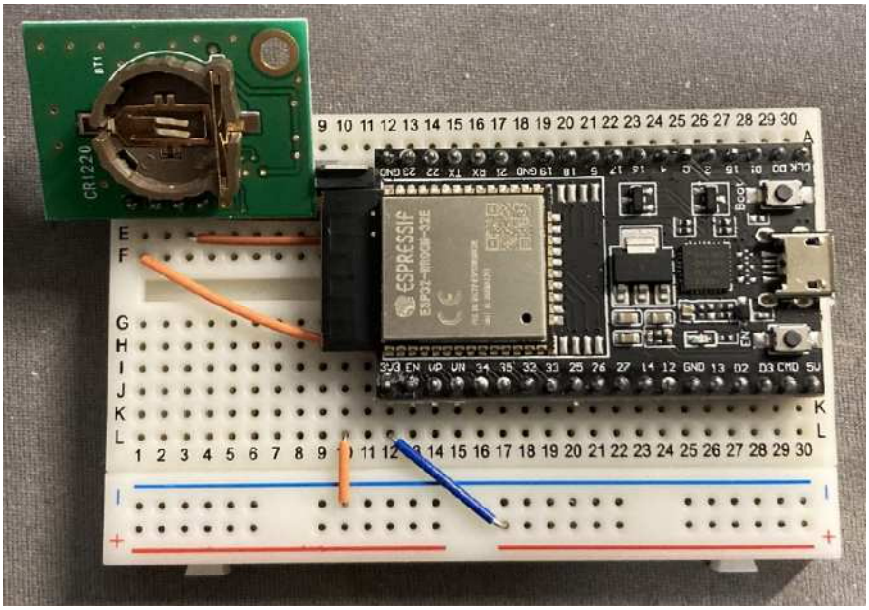


赤枠の部分に mosfet を差し込む

○RTC モジュール

RTC をVCC がC１に刺さるようにして、C1～C4 にかけて差し込みます。裏側のコネクタが出っ張っている都合上、斜めに差し込むことになります。

RTC にはボダン電池が付いているので、本体の電源が停止していても現在時刻を記録することができます。通常、このボタン電池の寿命は数年です。（写真ではボタン電池は取り付けていません。）

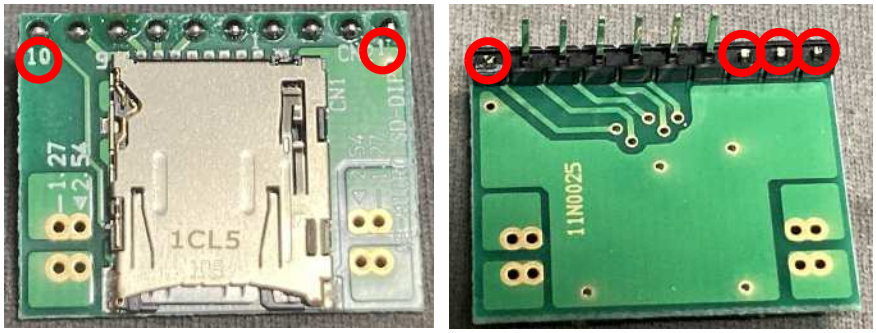


RTC モジュールの裏側が出っ張っているので斜めに差し込み

OSD カードリーダー

SD カードリーダーにはんだ付けしたピンのうち、実際に通信等に使用されるのは２番～７番までの６本です。動作に不要な１番、８番、９番、１０番のピンを付け根で切り取って他のピンへの干渉を防ぎます。

番号はカードリーダーの基盤に１と１０が記載してある（赤丸部分）ので、注意して切ってください。

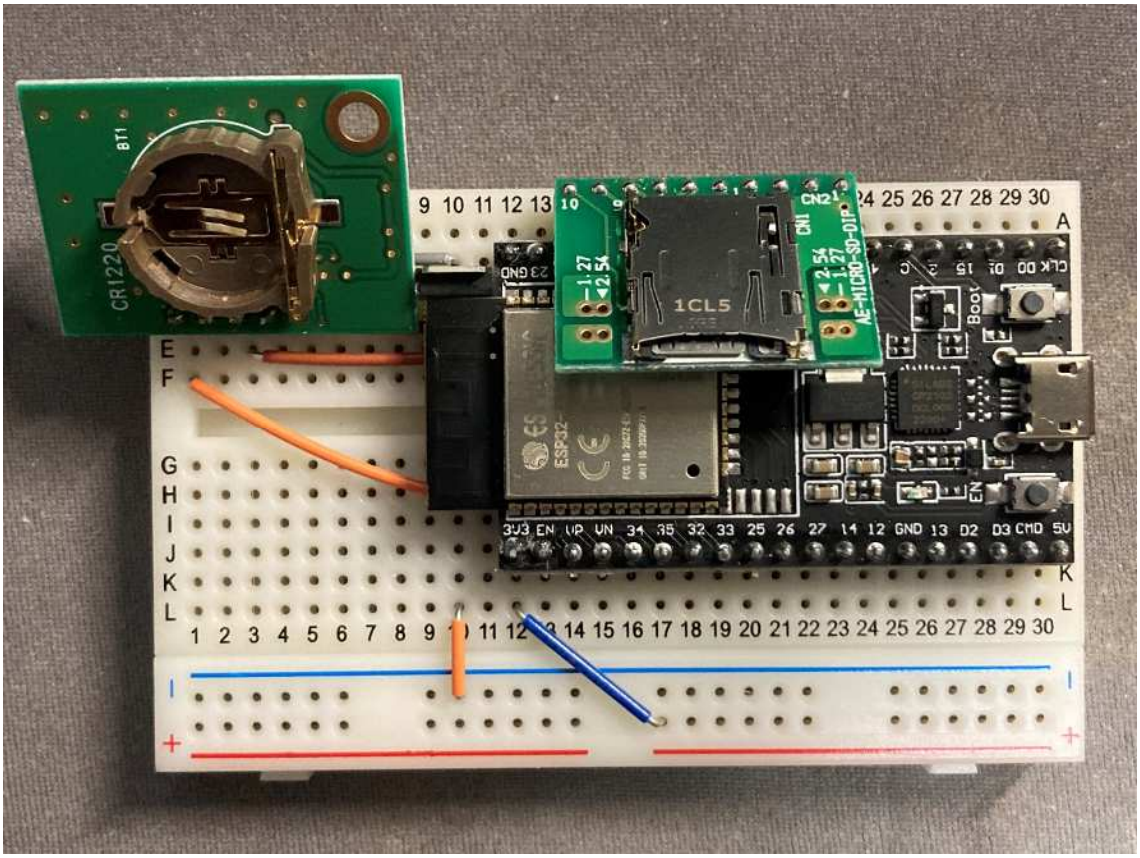


SD カードリーダーのピンの加工

ピンを切り終わったら、写真のように ESP32DevKitC にかぶせるように、A17～

A22 にわたって差し込みます。（A17 に SD カードリーダーの７番ピンが刺さります）
　RTC 同様斜めに差し込むことになります。SD カードはデータを確認する際に抜き差しする必要があるため、ピンの代わりにケーブル状のジャンパー線などをはんだ付けして、ある程度自由に動かせるようにすると扱いやすくなります。

ここまでで本体部分の組立てが完成しました。



　RTC には付属のボタン電池を、SD カードスロットには別に用意した１６GB 以下のマイクロ SD カードを差し込んでください。

　手前側にある ESP32 の GPIO３４、３５、３２、３３、２５、２６、２７、１４、１３、VP、VN が各センサーの読み取り端子として使用できます。（１２は使用できません）。
　SD カードと RTC を接続している周辺には、他の配線は行わないので、動作確認ができたら RTC と SD カードリーダーはホットボンド等を使って外れないように固定するとよいです。

ステップ４

作成した本体へのプログラムの書き込み

ステップ３で作成した測定装置本体に、ステップ２でインストールした Arduino IDE を使用してプログラムを書き込みます。

書き込むプログラムは、本マニュアルと合わせて農業総合試験場ホームページにアップロードされた、サンプルプログラムを使用します。マニュアルと合わせてあらかじめダウンロードしておいてください。

ファイル名：sample_ESP32_aichinososhi

サンプルプログラムを実行するためには、実際に装置を動作させるのに必要なプログラムをまとめた、「ライブラリ」を事前に PC にインストールする必要があります。

ライブラリはインターネットに接続した状態で Arduino IDE のライブラリマネージャーからダウンロードできます。

サンプルプログラムでは、

・センサーとして、温湿度、地温、日射量、EC、土壌水分

・部品として、マイクロ SD カードリーダー、RTC モジュール

を取り付けます。このうち、ライブラリをインストールする必要があるのは

●RTC モジュール：DS1307

●地温センサー：ds18b20

●温湿度センサー：AM2302（DHT22）

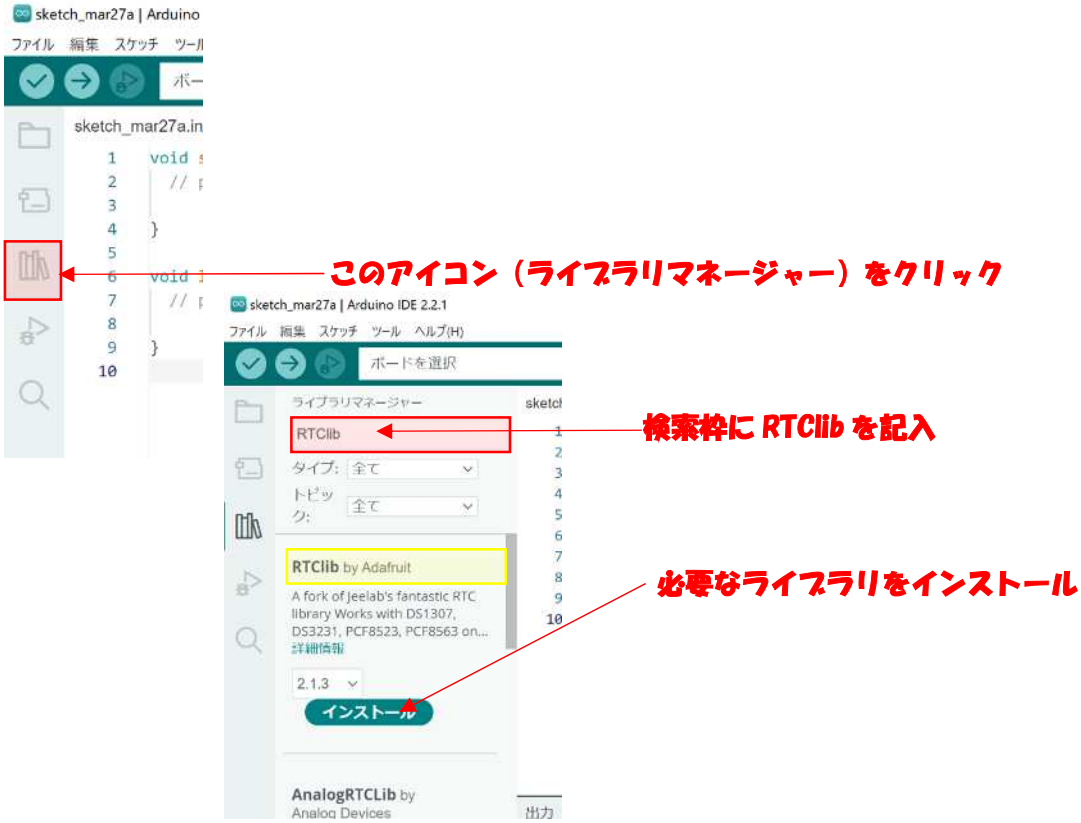
の３種類です。その他の部品やセンサーはライブラリを使用しないか、初めから Arduino IDE に基本のライブラリとして登録されているものを使用します。

通常、ライブラリにはサンプルプログラムも付いているので、プログラムについて詳しくない人でも、一部の簡単な修正で装置を使用することができます。

○ライブラリの取得

次の図を参考にして、各ライブラリをインストールしてください。ライブラリは Arduino IDE を使用した PC の中に保存されます。（ライブラリは一度取得すれば次回以降も使用できます）

●RTC モジュールのライブラリを取得



図のように、Arduino IDE の左端のアイコンの中から上から３つ目のライブラリマネージャーのアイコンをクリックし、検索窓に「RTClib」と入力すると候補がでます。

今回は「RTClib by Adafruit」を探してインストールしてください。ライブラリの詳細を知りたい場合は、詳細情報をクリックするとライブラリの詳細が記載されたウェブページにアクセスできます。ライブラリ自体の説明が記載されているほか、センサーや部品の接続方法などについても説明されているものもあります。（大半が英語表記です）

インストールをクリックすると、次のように依存関係をインストールの表示が出ますので、全てをインストールを選択してください。

「依存関係なしにインストール」を選択するとプログラムが動きません。

全てをインストールを選択すると、Arduino IDE の下側でインストール作業が始まります。インストールが終わったら次のライブラリをインストールします。

●温度センサーds18b20 のライブラリを取得

RTC モジュールのライブラリと同様に、検索をフィルタの部分に
「dallas」
と入力してください。候補の中から、
「DallasTemperature by Miles Burton...」
を選択します。

先ほどの RTClib と同様に、依存関係のライブラリとして OneWire が存在するので、全てをインストールを選択します。

●温湿度センサーAM2302 のライブラリを取得

最後に AM2302（DHT22）のライブラリをインストールします。

同様に検索をフィルタの部分に
「DHT22」
と入力してください。候補の中から、
「DHT sensor library by Adafruit」
を選択します。

依存関係にあるライブラリとして Adafruit Unified Sensor が存在するので全てをインストールを選択します。

以上で今回使用するライブラリは準備完了です。

サンプルプログラムでは SD カードやその他のセンサーについてはライブラリの登録は不要です。

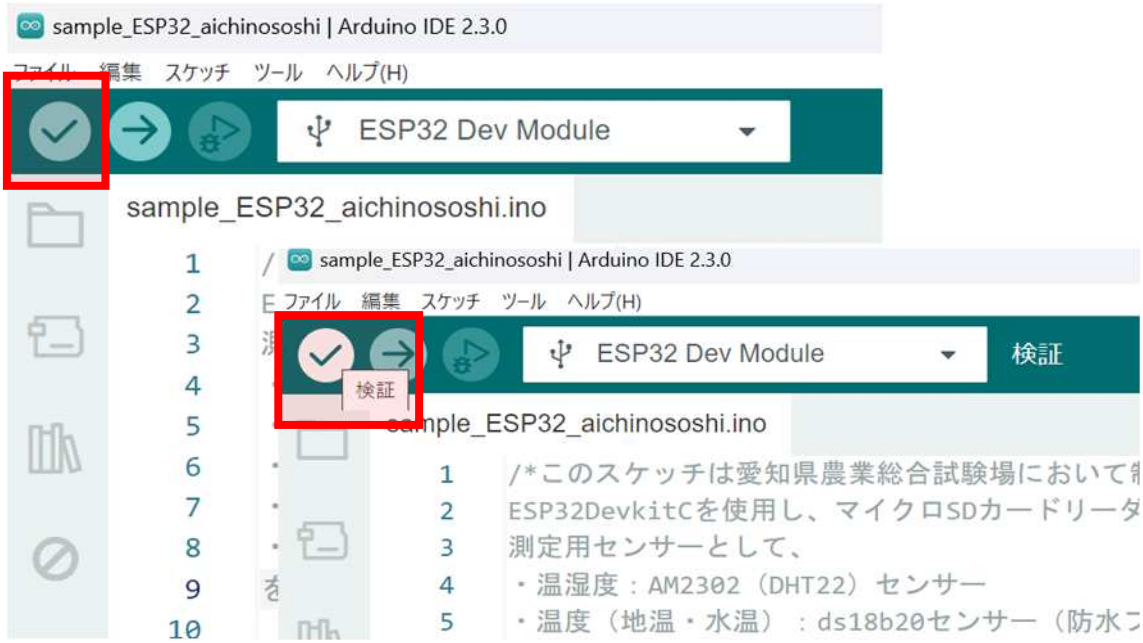
○プログラム（スケッチ）のチェック

サンプルプログラムの準備と必要なライブラリのインストールが終わったら、プログラムが正常なことを確認して、測定装置に書き込みます。

はじめに、Arduino IDE 上でサンプルプログラムを展開します。
この時点で ESP32DevkitC は PC と接続する必要はありません。

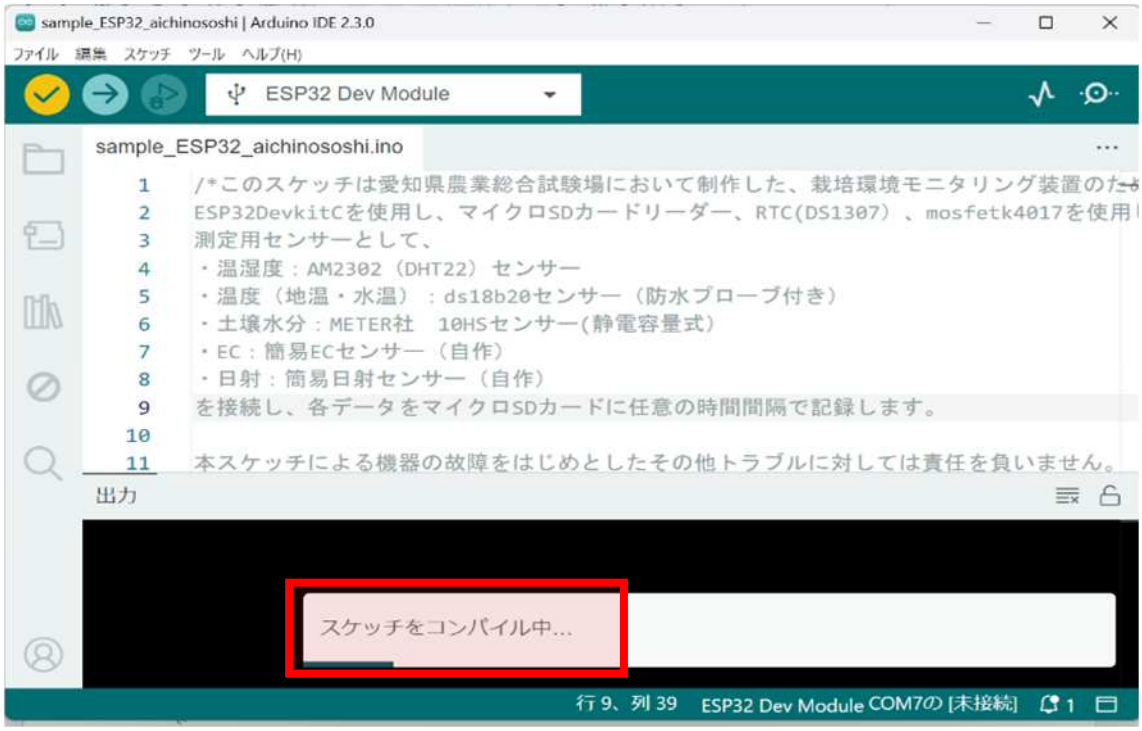
PC で Arduino IDE を開いたら、左上の「ファイル」をクリックして、開くを選択します。続いて、ダウンロードしたサンプルプログラムを確認して選択します。（特にダウンロード先を指定していない場合、WindowsPC では C ドライブのユーザー名フォルダに含まれているダウンロードフォルダに保存されます。）

サンプルプログラムの展開ができたら、Arduino IDE の左上にある✓マークをクリックしてください。（マウスカーソルを重ねると検証と表示されます。）

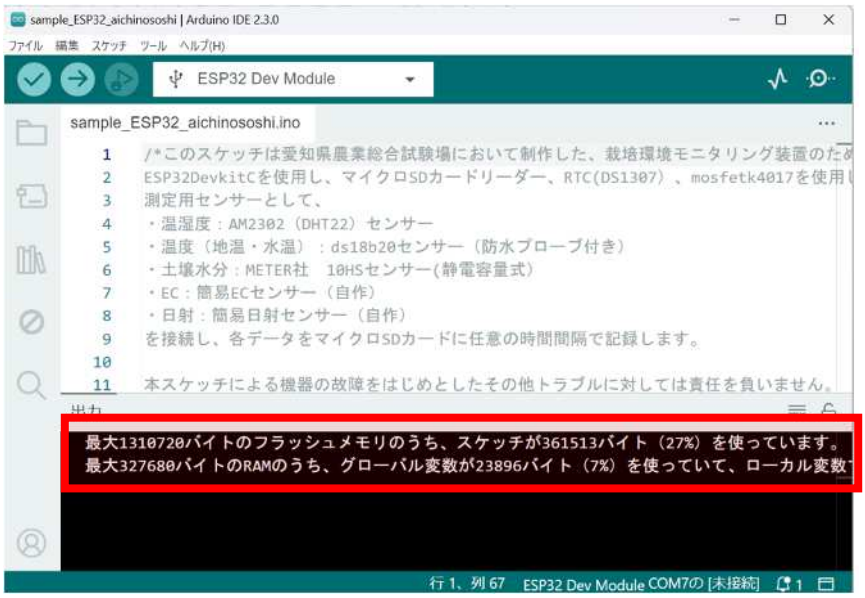


検証のボタンをクリックすると、Arduino IDE の下側に黒いスペースが展開し、スケッチをコンパイル中の表示が出ます。

コンパイルはプログラムの記述が、プログラムとして正しいかを確認する作業です。ライブラリが不足している、記述が間違っているなどの問題がないかの確認がされます。

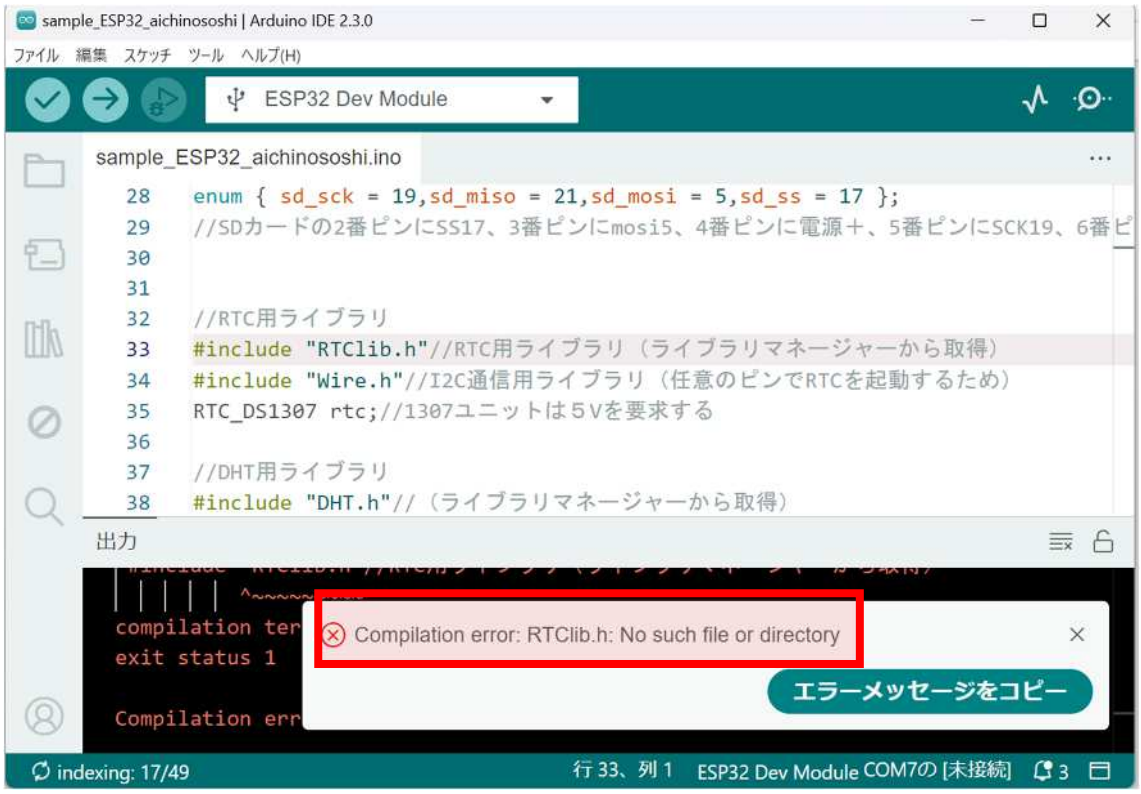


コンパイル中に問題が無ければ、コンパイル完了と表示され、図のようにプログラムに使用した容量が説明されます。



<<参考>>

コンパイル時に、必要なライブラリの登録がされていない等の問題があると、エラーが表示されます。次の図は RTClib のライブラリが登録されていなかった場合の表示です。



エラーメッセージとして、コンパイルエラー、RTClib というファイルもしくはディレクトリは存在しません。と表示されます。また、エラーに関係するプログラム部分（33 行

目）が薄い赤でマーカーされます。

　このようにプログラムのコンパイルによって、プログラムに不備がないかの確認をすることができます。

　この他によくあるエラーとして、

Compilation error: stray ’¥343’ in program

　プログラム中に全角文字が使用されている。（サンプルプログラムの展開時に、間違ってキーボードに触れてしまい、文字が入力された場合など）

Compilation error: expected declaration before ’}’ token

}が余分にあるなど（もしくは{がたりないなど

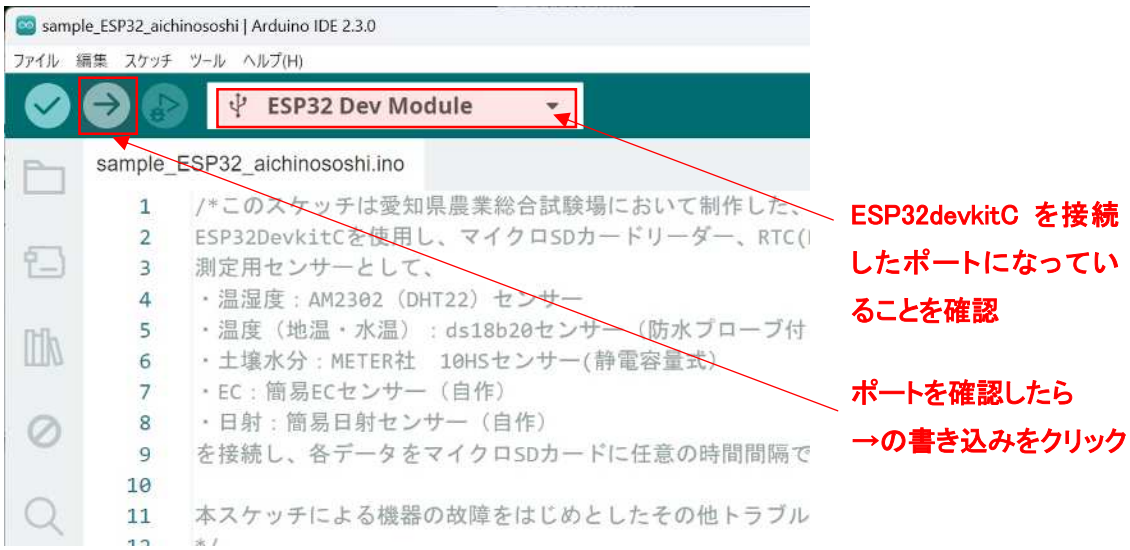
　エラーが出た場合は場所も示されるので、エラーについて調べることで対応できる可能性があります。

○プログラムの ESP への書き込み

測定装置の ESP32DevkitC の USB コネクタと PC を USB ケーブルで接続します。

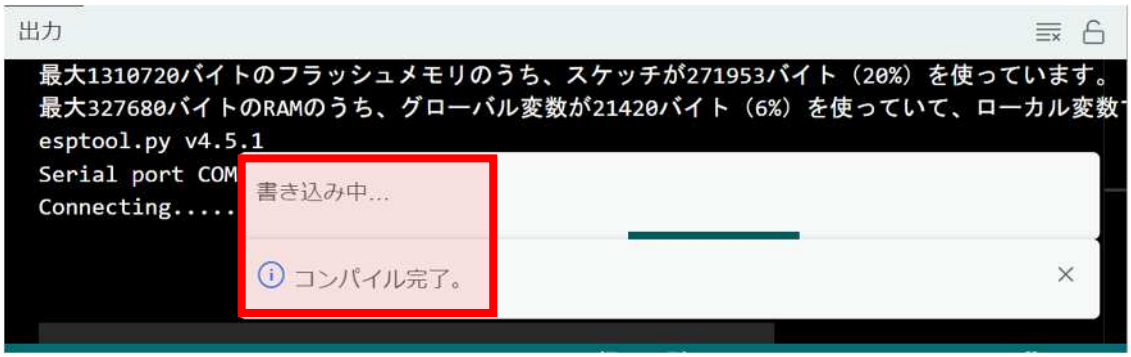
ステップ 2 を参考に、ESP32 と PC を接続するポートに間違いがないことを確認してインクルード（書き込み）の→をクリックします。

ポート設定が間違っていると書き込みが進まずやり直しになります。

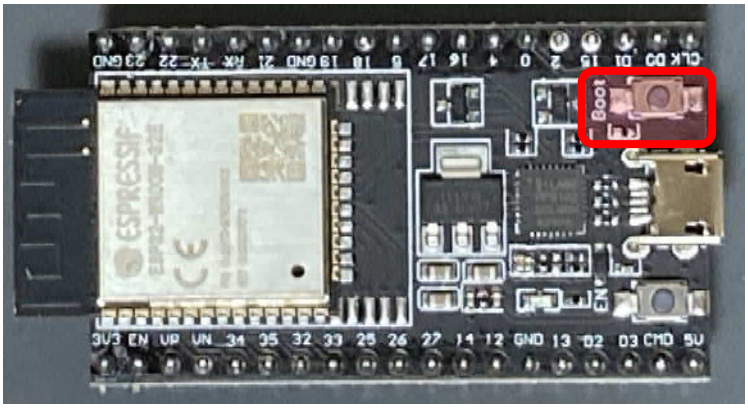


書き込みクリックすると、最初にもう一度コンパイル（検証）が実行されます。コンパイルが無事に終了すると、書き込み待機状態になります。

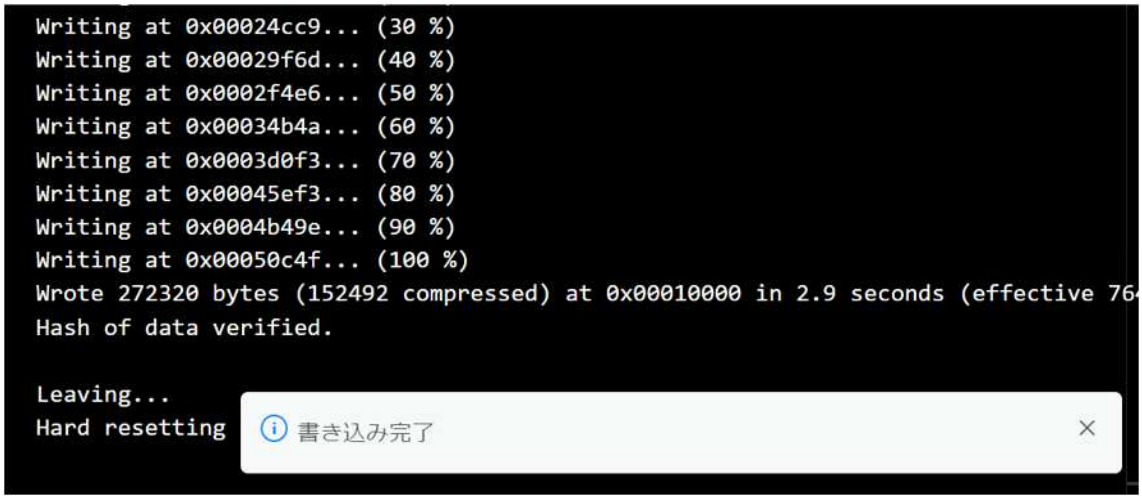
待機状態では書き込み中の表示が出るほか、黒い背景の Connecting の後ろに…が増えていきます。



書き込み待機状態になったら、ESP32DevkitC の Boot ボタンを長押しすると、書き込みが実行されます。（使用している PC によっては、Boot ボタンを押さなくても自動で書込みが開始します。）



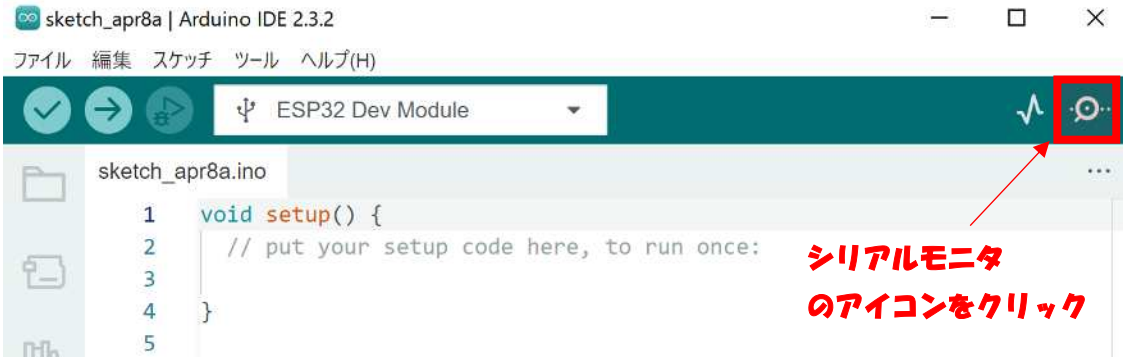
書込みが開始したら、ESP32 の Boot ボタン（赤色枠内のボタン）を長押し ArduinoIDE のウィンドウ下側で書込みの進捗状況が表示されるので、書き込みが終了するまで待ちます。



プログラムの書き込みができたら、実際に動作していることを確認します。

○本体の動作確認と RTC の初期設定

サンプルプログラムでは動作を確認するために、Arduino IDE のシリアルコンバーターという機能を使用して、動作内容を Arduino IDE のウィンドウ内に表示できます。プログラムを書き込み終わった状態で、Arduino IDE のシリアルモニタのアイコンをクリックします。



シリアルモニタのアイコンをクリックすると、Arduino IDE の下半分に動作状況が表示されます。

サンプルプログラムでは、

・RTC で取得した現在時刻
・温湿度センサーで取得した温湿度
・温度センサーで取得した温度（地温として表示）
・土壌水分センサーの出力値（mV で表示）
・日射量（mV で表示）
・土壌 EC（センサーの出力値で表示）

の他に、「処理完了」のメッセージが表示されます。このメッセージが出るとスリープモードに入り、10 秒後に再起動して繰り返しデータを表示します。

この時点ではセンサー類を接続していないため、各測定結果は正しい値が出力されません。

RTC はプログラムを ESP32 に書き込みした時の PC の現在時刻を記録して、以降は 1 秒ごとに時間をカウントしていきます。表示される時刻が大きく異なっている場合は、RTC が正しく接続されていない可能性があります。

SD カードは、正しく動作していれば SD カード内にデータを保存したファイルが作成されます。シリアルモニタに「処理完了」の表示が出たら、一度 ESP32 から USB を抜いて電源を切り、SD カードを抜いてカードリーダーなどを使ってを PC で確認してください。「test1.csv」というファイルができていれば正しく動作しています。

csv ファイルはエクセルで開くことができるので、ファイルを開いて測定時刻が正しく記載されていることを確認してください。

SD カードにファイルができていない場合、SD カードリーダーの接続位置がずれていること、SD カードの容量が 64GB 以上で装置に対応していないことなどが考えられます。

正しくプログラムの書き込みができている場合、SD カードを刺した状態で ESP32 に USB を接続する（電源をいれる）とファイルの一番下に新しいデータが記録されるようになります。このデータはスリープから復帰するたびに記録されるので、後で紹介するようにスリープ時間を修正することで、30 分おき、1 時間おきなどのデータ収集が可能になります。

ステップ５

各センサーの準備と接続

ステップ４で本体の動作が確認できたら、使用する各センサーを準備します。
本マニュアルでは、
〇温湿度センサー
〇地温センサー
〇土壌水分センサー
〇簡易 EC センサー
〇簡易日射センサー
を１つずつ取り付けることを想定しています。

各センサーはジャンパー線などを使用して測定装置本体のブレッドボードに取り付け、ESP32DevkitC の GPIO ピンやプラス、マイナスの電源に接続します。
センサーには購入した時点でケーブルが付いているものや本体だけのものがありますが、実際に使用する場面では、ケーブルの長さは設置場所に合わせて延長する必要があります。
このため、初めに各センサーをはんだ付けやコネクタを利用して、測定装置本体とケーブルで繋げられるようにします。

○センサー接続のためのケーブルについて

測定装置はブレッドボードで作成しているので、センサーを接続する際にもジャンパー線などを経由してブレッドボードに接続します。

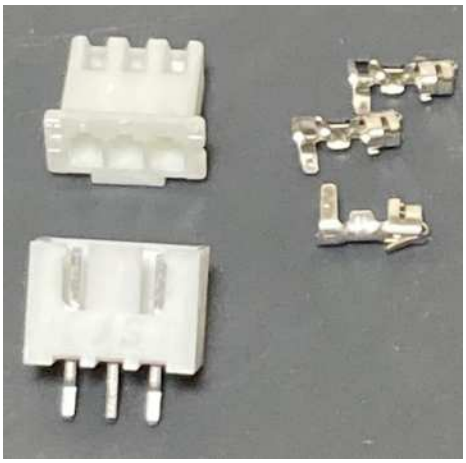
センサーを自由に設置するためには、十分な長さを持ったケーブルを用意する必要があります。ケーブルとセンサーの接続にははんだ付けを行う方法もありますが、コネクタを使用することで簡易にセンサーを接続することができます。

●ケーブルの末端をコネクタにする

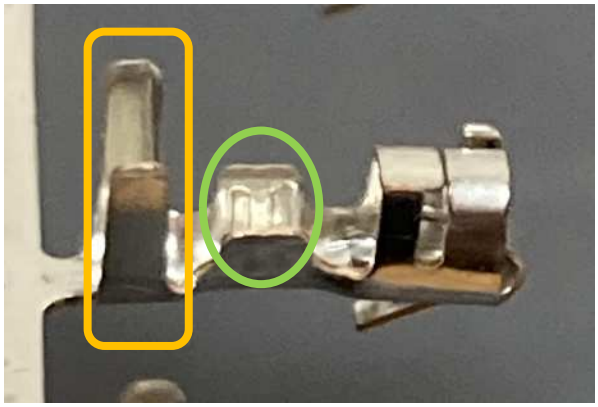
ブレッドボードを使用するときは、コネクタとして JST（日本圧着端子製造株式会社）の XH コネクタを使用します。XH はケーブルの太さや端子の間隔(2.54：ブレッドボードの穴の間隔と同じ)に対応する規格で、ジャンパー線と組み合わせるのにちょうど良いサイズです。

JST のコネクタは、ケーブルの電線部分を固定する「コンタクト」と呼ばれる金属部品と、そのコンタクトを納める「ハウジング」、ハウジングとオスメスの関係になる「ポスト」の 3 種類の部品で構成されます。本マニュアルではハウジングにジャンパー線をさして直接ブレッドボードに接続します。ポストやコネクタピンを利用することでケーブル同士をつないで延長することもできます。

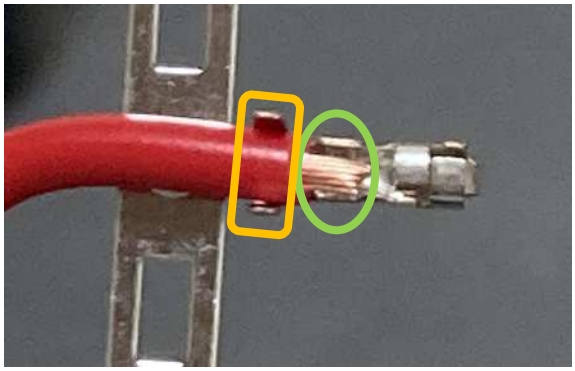
コンタクトは写真のような形状をしており、被覆電線をおおよそ２～３㎜程剥離して、その銅線部分を締め付ける部分と、被覆の部分を締め付ける 2 か所で圧着します。ペンチで曲げることでも対応できますが、専用の圧着ペンチを使用することでよりきれいに、しっかりと固定できます。慣れないうちは圧着がゆるくて抜けてしまうことや、コンタクトが曲がってしまうこともあるので、コンタクトはいくつか余分に用意するとよいです。



左上ハウジング、右上、コンタクト
左下ポスト



コンタクトを拡大した写真
緑丸部分：導線を圧着する部分
黄枠部分：被覆部分を圧着して固定する部分



実際に圧着する前の状態

２～３mmほど被覆をむいたケーブルを、コンタクトに差し込み、導線を圧着する部分に差し込んだら、ペンチや圧着ペンチでしっかりと固定します。この部分でコンタクトと電線は電気的につながった状態になります。

使用するケーブルが細すぎる場合は固定しにくくなります。ケーブルを太いものに変更する、圧着時に被覆をむいた部分の長さを調節するなどの工夫をすると固定しやすくなります。

導線部分を圧着したら、ビニール被覆された部分も動揺に圧着します。ビニール被覆部は強く締め付けすぎると切れてしまうので、ビニールにコンタクトが食い込んで抜けなくなる程度にします。

ケーブルの末端にコンタクトを固定したら、コンタクトの爪の向きとハウジングの穴の向きを合わせて、ハウジングにコンタクトを差し込みます。間違ってつけてしまったときや、ケーブルが断線して付け直す必要があるときは、爪の部分をカッターナイフなどで押し込むことで、ハウジングからコンタクトを抜くことができます。

○ケーブルをコネクタで延長する

今回使用する温度センサーds18b20 はあらかじめケーブルが付いたものを使用しますが、ケーブルの長さが１ｍ程度のため、センサーを地面に埋設するためには延長が必要になる可能性があります。

この場合、延長用のケーブルを

●はんだ付け

●圧着端子

●熱収縮チューブ等（ビニールテープでも可）

などを併用してつなげる必要があります。

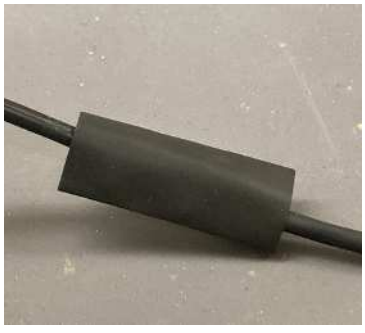
ds18b20 センサーではプラス、マイナス、データ通信の３本の電線でつながっているので、延長するときは同じく３本の導線がある３芯のケーブルを使用します。

今回は、延長したいケーブルの両側をコネクタにして、ポストを使って接続する方法を提示します。ポストはもともと基盤に接続するためのピンとコネクタを差し込むピンが付いているため、両側からコネクタを接続することが可能です。ただし、強度は不十分なので、接続後は熱収縮チューブやビニールテープなどで十分に保護する必要があります。

両端をコネクタにする

ポストを刺して接続

両端を接続して補強



熱収縮チューブで覆う

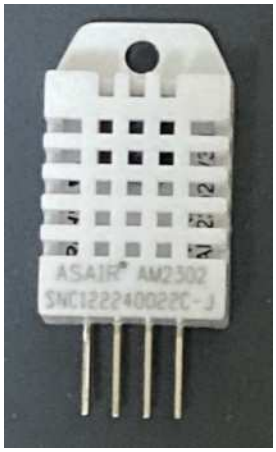
ドライヤー等で熱して収縮させる＊

（＊写真ではコネクタ部分を内径 10 ㎜の熱収縮チューブで覆った後、さらに両端を６㎜のチューブで覆っています。）

○各センサーについて

○温湿度センサーAM2302（DHT22）

本マニュアルでは、温湿度用のセンサーとして AM2302 を推奨します。1,000 円程度と比較的安価ですが、温度湿度ともに十分な精度が期待できます。（データシート上の湿度精度±２%、温度精度±0.5℃）



AM2302 は左の写真のように４本のピンが付いたセンサーです。

正面から見ると網目状になっている部分の内側に、温度と湿度を測定するセンサーが組み込まれています。

４本出ているピンの役割は写真の向きで左から

１．VCC（３．３V）

２．データ通信

３．NC（何にもつながっていない）

４．GND（マイナス端子）

になっています。

センサーを接続するときは、３芯ケーブルに４本ピンのコネクタをつけて、コネクタに差し込みます。配線が細く抜けやすいので、動作の確認ができたらホットボンドなどを利用して接続部を漏電対策と合わせて保護します。

延長ケーブルの逆側の端は、ジャンパー線でブレッドボードに接続できるように、3 本ピンのコネクタにしておきます。この時、ケーブルの内側の電線の被覆の色を確認し、どれがセンサーのどの端子につながっているかわかるようにしてください。

（通常、赤色が 3.3V、黒色や青色を GND、黄色や白色をデータ通信線に使用します。）

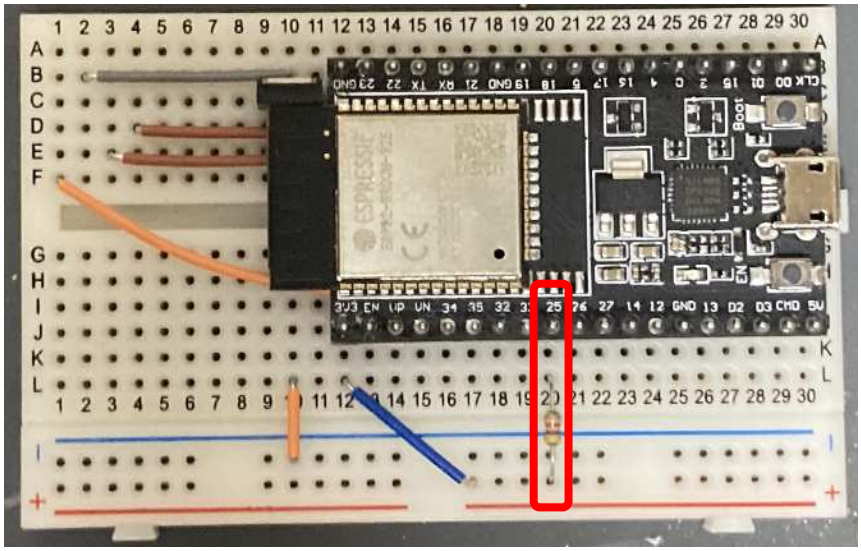
このセンサーを使用するためにはプルアップ抵抗として抵抗器を設置する必要があります。（本マニュアルではプルアップ抵抗についての詳細な説明は割愛します。）

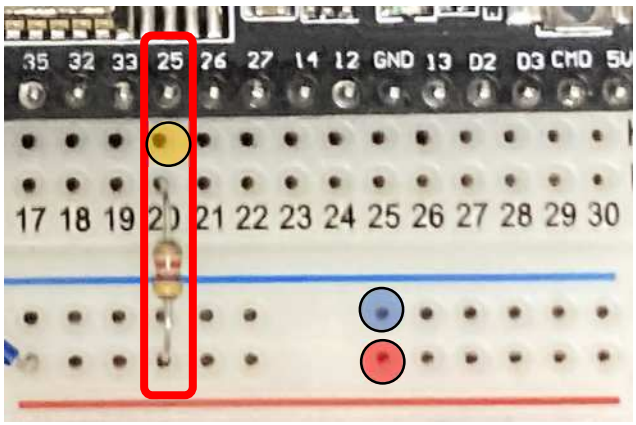
プルアップ抵抗は本体側のブレッドボードに接続して対応します。ブレッドボードの任意の GOIO ピン（サンプルプログラムでは２５を選択）に、データ通信ピンを接続し、接続したデータ通信ピンとブレッドボードの手前にあるプラスのエリアを４．７ｋΩ抵抗で接続します。

以下を参考にして、各配線を接続してください。

データ通信線：GPIO25
VCC：ブレッドボード手前のプラスのエリア(3.3V とつながっている)
GND：ブレッドボード手前のマイナスのエリア(mosfet の中央の足とつながっている)
プルアップ抵抗：GPIO25 と、ブレッドボード手前のプラスのエリアにまたがって設置する

プルアップ抵抗はそのままだと足が長すぎるため、ニッパー等で 1～1.5 ㎝程度に切って折り曲げて差し込みます。





・ESP32 の GPIO25（ブレッドボードの L-20）と手前のプラスのエリアを 4.7ｋΩ抵抗でつなぐ
・ESP32 の GPIO25（ブレッドボードの K-20）にデータ通信ピンを接続
・プラスのラインに VCC を接続
・マイナスのラインに GND を接続

各配線の接続ができればセンサーの設置は完了です。

○地温センサーds18b20

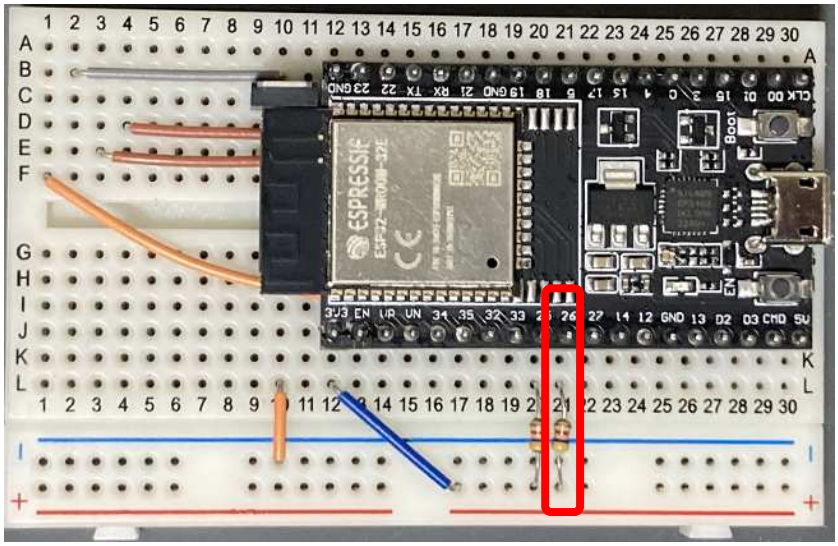
地温センサーとして販売されている ds18b20 はあらかじめ 1m のケーブルと接続されている防水加工済みのものを使用します。

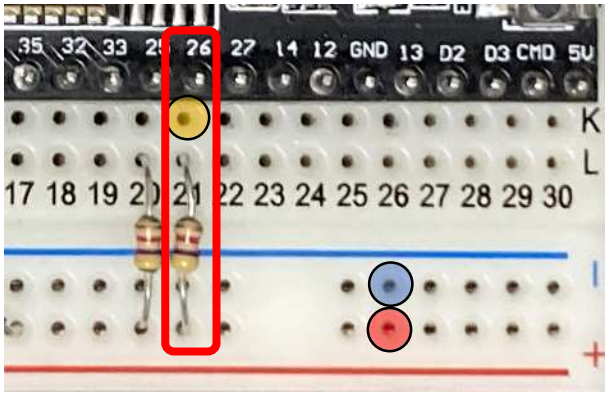
ケーブルは通常、赤、黄、黒の三本になっており、それぞれ、
赤色：3.3V
黄色：データ通信
黒色：GND
に対応しています。

ケーブルの長さが１mなので、測定装置の設置場所の都合で長さが足りない場合は、３芯ケーブルを使って延長します。

購入時点では通常末端部分はコネクタになっていないものもあるので、３ピンのハウジングを使用してコネクタにします。接続部をコネクタにできたら、ジャンパー線を使ってブレッドボードに接続します。このセンサーにも 4.7kωのプルアップ抵抗が必要です。
以下を参考に接続してください。

データ通信線：GPIO26
VCC：ブレッドボード手前のプラスのエリア(3.3V とつながっている)
GND：ブレッドボード手前のマイナスのエリア(mosfet の中央の足とつながっている)
プルアップ抵抗：GPIO26 と、ブレッドボード手前のプラスのエリアにまたがって設置





（AM2302 の抵抗も接続した状態です）

・ESP32 の GPIO26（ブレッドボードの L-21）と手前のプラスのエリアを 4.7ｋΩ抵抗でつなぐ
・ESP32 の GPIO26（ブレッドボードの K-21）にデータ通信ピンを接続
・プラスのラインに VCC を接続
・マイナスのラインに GND を接続

○土壌水分センサー　10HS

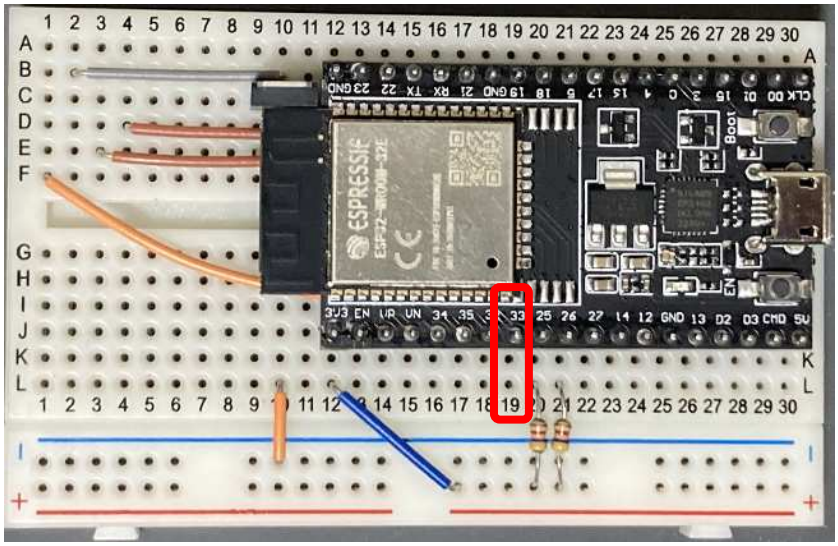
土壌水分センサー10HS は購入時に５mのケーブルの末端が、３ 端子のイヤホンジャックになっています。このままイヤホンジャックとして使用する場合は、イヤホンジャックのコネクタを購入してジャンパー線などをはんだ付けすることで対応できます。

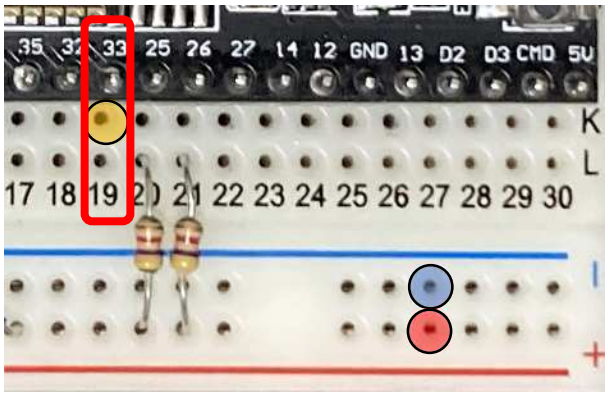
今回はケーブルを切断してコネクタにすることで接続します。

ケーブルを切断すると 2 本の被覆されたケーブルと、1 本のむき出しのケーブルが出てきます。製品のロットによってケーブルの色が異なることがありますが、付属のマニュアルの記載では、白または茶色のケーブルが電源供給なので 3.3V に、赤色またはオレンジ色のケーブルがデータ出力なので任意の GPIO（今回は 33）に、むき出しのケーブルは GND に接続できるよう、コネクタを圧着します。GND のケーブルはむき出しなので、熱収縮チューブやビニールテープで他のケーブルに接触しないよう保護する必要があります。。

10HS センサーにはプルアップ抵抗は必要ありません。

データ通信線（赤もしくはオレンジ）：GPIO33
VCC（白または茶）：ブレッドボード手前のプラスのエリア(3.3V とつながっている)
GND（むき出しの線）：ブレッドボード手前のマイナスのエリア(mosfet の中央の足とつながっている)





・ESP32 の GPIO33（ブレッドボードの K-19）にデータ通信ピンを接続
・プラスのラインに VCC を接続
・マイナスのラインに GND を接続

○簡易土壌 EC センサー

本測定装置では、土壌 EC センサーは自作の電極を利用した簡易センサーを使用します。EC は 2 点の電極の間にあるイオンの量を電気の流れやすさから推定した指標です。イオンが多い、つまり肥料が多い場合、電極間に電気が流れやすく、EC の値は上昇します。逆に、電気は抵抗があると流れにくくなります。

このことから、2 本の電極を用意して、電極間の抵抗値を測定することで、EC を計算することができます。

電極の長さや電極間の距離が出力結果に影響しますが、標準液で校正することで一般的に EC の単位として使用されている dS/m に変換することができます。

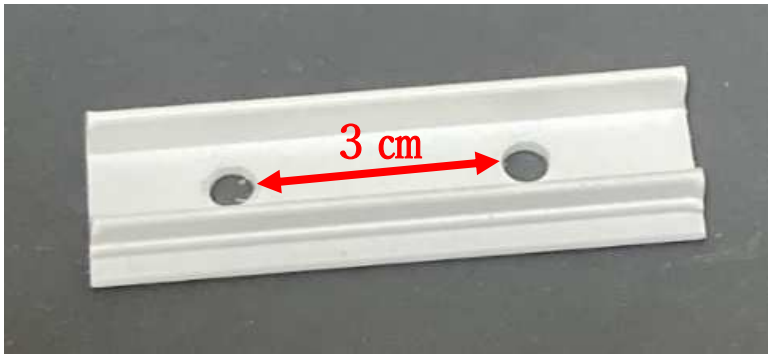
この簡易センサーの原案は、家庭用コンセントを EC センサーとして使用する海外の事例ですが、土壌に埋設した場合、電極が鉄でできている家庭用コンセントでは容易にさび付いてしまうため、ステンレスのネジを電極として活用します。また、電極の土台として、ケーブルカバーを短く切ったものを使用しています。ケーブルカバーはカッターなどで 5 ㎝程の長さに切って使用します。

ケーブルカバーは、使用するねじの頭がカバー内に収まる大きさであれば特に問題はありませんが、朝日電機株式会社が販売している M-030H　ABS モール 30 ㎝が扱いやすいかと思います。（ホームセンターなどで販売されています）

使用するステンレスねじはねじの径が 3mm、長さが 20mm のもの（M3×２０mm）を選びます。固定するためにナット付きのものが望ましいです。

使用したケーブルカバー(5 ㎝にカット)とステンレスねじ

初めに、ケーブルカバーにねじの径より若干広い 3.2 ㎜径のドリルで 2 か所穴をあけます。ねじの径より若干広い穴にすることでまっすぐに固定しやすくなります。この時穴の距離が近いほど精度が出ますが、設置場所によるばらつきを大きく受けるので、3 ㎝程度の間隔が望ましいです。



ケーブルカバーに 3.2 ㎜の穴を約 3 ㎝間隔で開ける

ステンレスねじのネジ山部分に、ステンレス用のはんだを盛りつけます。ステンレスにはんだ付けをする場合は、専用のはんだと、専用のフラックスが必要になります。どちらもホームセンター等で購入できます。はんだを盛り付けるときは、クリップで挟んだり段ボールなどに差し込んだりして固定するとよいです。

ステンレス用のフラックスは強酸性なので扱いには注意してください。

ネジ山のみぞ部分に、ステンレス用フラックスをごく少量たらします（1 滴より少ない量で十分です）。フラックスが付いたネジ山に、ステンレス用はんだを通常のはんだと同じようにはんだ付けします。後でこのはんだにケーブルを固定するので、ネジ山が完全に埋まるよう多めにはんだを盛り付けます。蒸発したフラックスが白く表面に残るので、綿棒等でぬぐい取ってください。

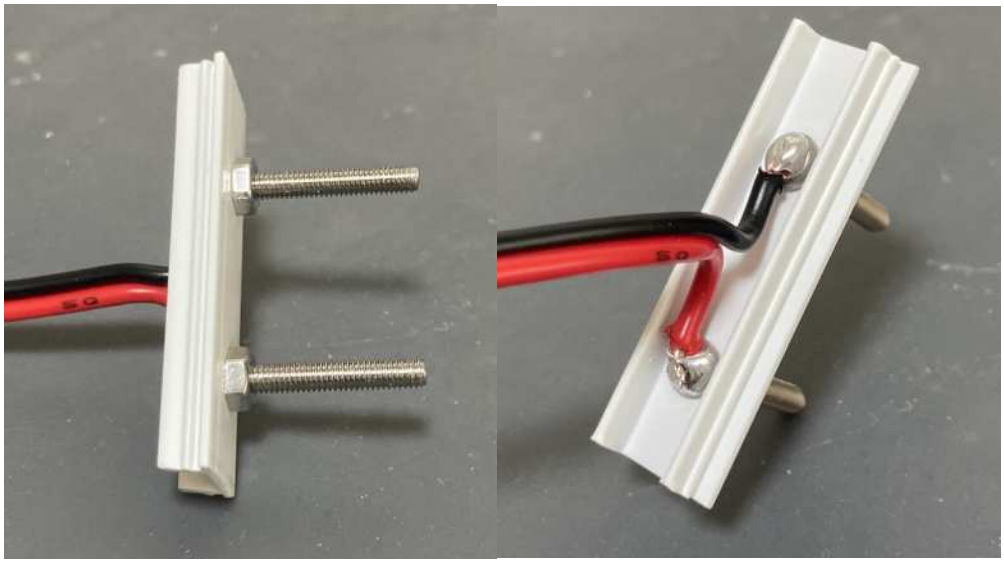
ねじ山にはんだを盛り付けたステンレスねじ

ネジ山部分にはんだを盛りつけたら、2 芯のケーブルをこの部分に取り付けます。ケーブルの先端の被覆を 3 ㎜程度はがして、ねじってまとめ、ネジ山のはんだに押さえつけな

がらはんだごてでネジ山のはんだを溶かします。この際フラックスは必要ありません。

ねじ山のはんだをつかして、ケーブルを接続

ステンレスねじとケーブルのはんだ付けが終了したら、ケーブルカバーにあけた穴に、ステンレスねじを差し込み、ナットで固定します。



この時のネジの間隔と長さが EC の出力に影響する要素となります。

ねじを土台のケーブルカバーに固定したら、電極部分以外が漏電しないように、ホットボンドを充てんし、ケーブルカバーの内側をしっかりと密封します。

ネジ山部分やケーブルの接続部、ケーブルカバーの端など水が浸入しやすい部分を特に注意して全体的に埋めるようにしてください。

ホットボンドで全体を埋める

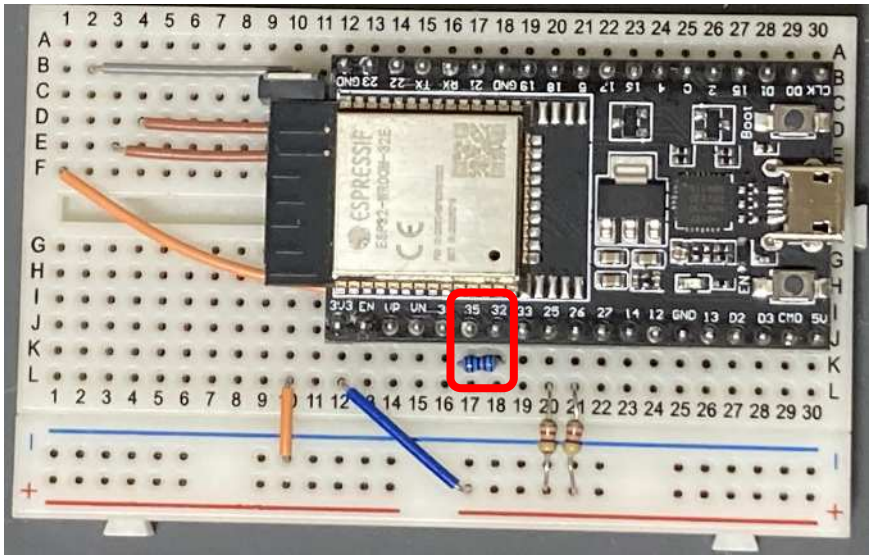
以上で EC 測定用の簡易電極の作成は完了です。ホットボンドでしっかりと固定できていれば、ナットは不要なので取り外しておきます。（ナットは位置決め用の仮の固定です。つけたままにしておくと測定結果に影響することや、設置後に外れてしまう可能性があります。）

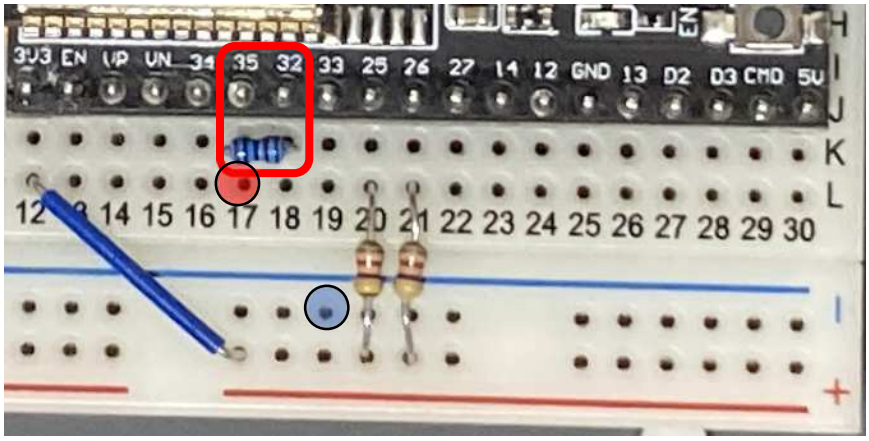
測定用電極の準備ができたら、ESP32 の任意の GPIO（今回は 35）に一方を、もう一方を GND のラインに接続します。写真では赤と黒の二色のケーブルを使用していますが、この電極については、向きは関係ありません。

電極の他に、任意の GPIO（今回は 32）と先の電極を接続した GPIO（今回は 35）の間に 510ωの抵抗を接続します。

この時 510ω抵抗は電極との比較で EC を割り出すための基準値となるので、精度の高い金属皮膜抵抗を用いるのが望ましいです。なお、通常の農業で使用する EC は 2.5dS/m 以下なので問題はありませんが、より高濃度の EC を測定するときは精度を上げるために抵抗値を低いものに変更する必要があります。この際、抵抗値に合わせて改めて EC の標準液での補正が必要となります。

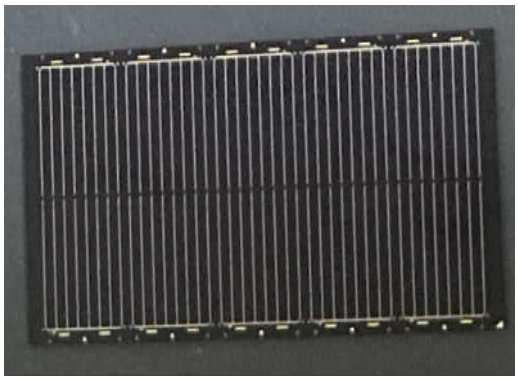
また、EC は温度によっても影響を受けるため、より正確な測定結果が必要なときは、地温センサーを併設して、測定時の温度を記録するようにして、25℃時の数値に補正します。



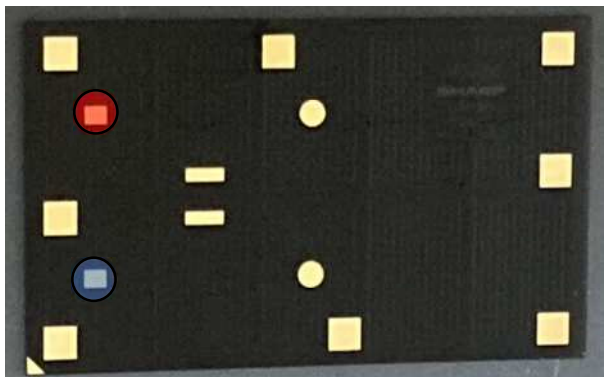


- ESP32 の GOIO32 と GPIO35（ブレッドボードの K-18 と K-17）を 510ω抵抗でつなぐ（分圧用の既知の抵抗）
- 簡易 EC センサーのケーブルを GPIO35（ブレッドボードの L17）とマイナスのラインにつなぐ。（ケーブルの向きはどちらでもよい）

○簡易日射センサー（ソーラーパネル）



ソーラーパネル表面



ソーラーパネル裏面

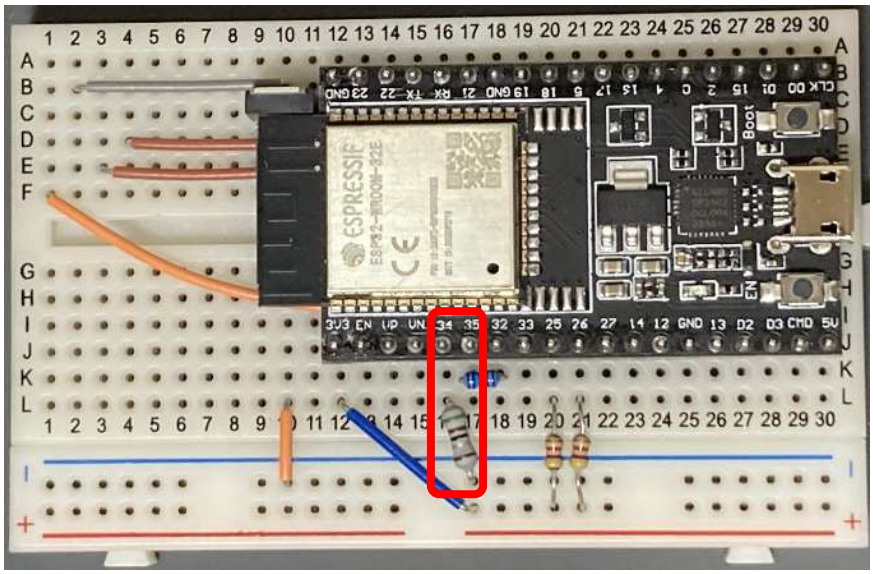
日射量とソーラーパネルが発電する電圧との間には強い相関関係があることから、ソーラーネルの発電電圧を測定することで、日射量を簡易的に評価することができます。

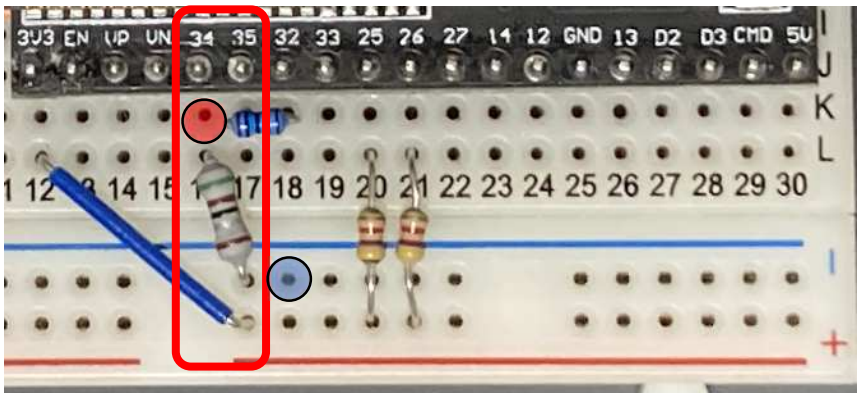
ただし、小型のソーラーパネルは少量に日光で発電量が上限に達してしまうため、抵抗器を使って出力される電圧を下げることで、出力を測定範囲内に収める必要があります。

材料のリストに上がっているソーラーパネルはケーブルが付属していないので、２芯のケーブルをソーラーパネルの裏面にあるプラスの電極とマイナスの電極にはんだ付けする必要があります。電極の配置についてはソーラーパネルに付属するマニュアルを確認してください。

このソーラーパネルは表面は樹脂によるコーティングがされていますが、裏面の電極はむき出しなので、ホットボンドやラッカー塗料、マニキュアなどを使って電極を保護してください。（マニキュアは塗膜の強度もある程度高く、乾燥も早いため電極の保護に扱いやすいです。）

ソーラーパネルのプラスの電極を任意の GPIO（今回は 34）に接続し、マイナスの電極を GND のラインに接続します。また、プラス電極を接続した GPIO と GND のラインの間に 22ω（22ｋωではない）を接続することで、発電した電圧を減衰します。この時、抵抗が熱を発生するので、容量の大きい１W の抵抗を使用する必要があります。抵抗をつけると、室内の光ではほとんど出力しなくなります。





・ESP32 の GPIO34（L-16）とマイナスのラインを 22ω抵抗（1W）でつなぐ
・ソーラーパネルのプラス端子を ESP32 の GPIO34
・ソーラーパネルのマイナス端子をマイナスのラインにつなぐ

　ソーラーパネルを設置していると、パネル面の汚れや白化等が起きるので、定期的にパネルを確認して、パネル面が白濁した場合や、晴天時でも測定値が大きく下がるようなことがあれば、交換する必要があります。

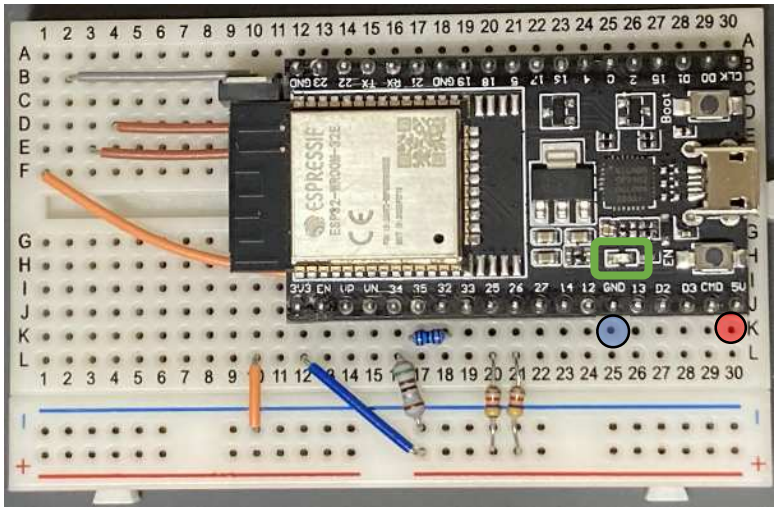
○電源用電池ボックス

各センサーの準備ができたら、電源用の電池ボックスを接続してスイッチを入れることで動作が開始されるのを確認してください。

電源に使用する単三乾電池×４本の電池ボックスには、購入時点で赤と黒のケーブルが付いていますが、末端はコネクタになっていません。これまでと同様に末端部分をコネクタに加工して使用します。

電池ボックスにスイッチがついているので、使用するときだけ電源を入れることができます。

電池ボックスのプラス端子（赤いケーブル）を ESP32devkitC の５V の端子に、マイナス端子（黒いケーブル）を ESP32devkitC の GND にそれぞれ接続します。この時、電源の端子はセンサー等を接続していたブレッドボードの手前の電源のエリアではなく、ESP32DevkitC の５V、GND につないでください。（手前のエリアは mosfet を使用してスイッチで切り替えているため、ここに電源を接続することはできません）



正しく接続できていれば、電池ボックスのスイッチを入れると、ESP32DevkitC の LED（緑色の枠の部分）が点灯します。ここまでの組み立てに問題が無ければ、SD カードにデータが追記されるので、PC 等で SD カードのデータを確認してください。

以上で装置の作成は終了です。

○測定結果の読み方

記録されるデータは、SD カードに csv 形式で測定毎に新しい行で記録されます。
csv 形式のファイルは、表計算ソフトなどで一覧表として表示することが可能です。
マイクロソフト社の office エクセル等を使用することで、図のようにデータを確認することができます。

	A	B	C	D	E	F	G
1	DAYTIME	temp	humidty	SR	g－temp	g-moist	EC
2	2024/5/31 0:00	20	77.6	0	19.06	1100	0
3	2024/5/31 0:30	20.1	76.3	0	19	1098	0
4	2024/5/31 1:00	19.9	83.9	0	19	1101	0
5	2024/5/31 1:29	19.7	86	0	18.94	1104	0
6	2024/5/31 1:59	19.1	91.8	0	18.94	1105	0
7	2024/5/31 2:29	18.6	87.9	0	18.94	1104	0
8	2024/5/31 2:59	17.5	97.3	0	18.87	1099	0
9	2024/5/31 3:28	17.6	98.8	0	18.87	1102	0

SD カードに記録されるデータの例

測定装置で出力されるデータはセンサーごとに異なるデータになります。

・AM2302（DHT22）センサーでは、測定した温度(℃)と湿度(%)が直接出力されます。
・ds18b20 では、センサーのプローブ部分の温度(℃)が直接出力されます。
・10HS 土壌水分センサーではセンサーが出力する電圧値として mV が出力されます。
10HS 土壌水分センサーでは、空気中等水分が 0%に近い条件では 300～400 程度の、センサーを水没させる等 100%に近い状態では 1200 程度の数値を出力します。
・日射センサーとして使用するソーラーパネルからは、ソーラーパネルの発電した電圧値が mV で出力されます。ソーラーパネルが発電する電圧は日射量（MJ）と比例する数値になります。
・簡易 EC センサーで出力される値は、mS ですが、電極の長さや大きさ、電極間の距離

によって補正が必要なため、一般的に使用される dS/m（mS/cm）に変換するためには EC 用の標準液を簡易 EC センサーで測定し、出力値と標準濃度から補正を行う必要があります。

各センサーが正常に動作することを確認するために、センサー部分を手で握る、水につける、光を当てる等しながら、SD カードに記録されるデータを確認、もしくは USB で PC と接続して、シリアルモニタに表示されるデータを確認してください。

ステップ6

プログラムの調整（完成）

ステップ５までで、各センサーのデータを SD カードに時間毎に記録する装置が完成します。ステップ４で登録した動作確認用のサンプルプログラムでは、測定間隔を 10 秒おきに設定しているため、実際に使用する場合はプログラムを一部調整して、改めてプログラムの書き込みを行う必要があります。

使用するプログラムは、ステップ４で使用したものを引き続き使用します。

ファイル名：sample_ESP32_aichinososhi.ino

左上のファイルのタブから、開くを選んで、ダウンロードしたサンプルプログラムを選択する。

プログラムが開いたら、動作間隔を決定する内容を修正します。

○スリープ時間の修正

```
22  //ディープスリープ設定
23  #define uS_TO_S_FACTOR 1000000ULL  //ディープスリープの指定はマイクロ秒で扱いにくいので秒単位に変換
24  #define TIME_TO_SLEEP  10 //スリープ時間を設定（秒単位）
25  RTC_DATA_ATTR int bootCount = 0; //動作回数をカウントし、初回のみの動作を指定するためにbootCountを用意
```

スリープ時間を修正する部位は、上記の２４行目、

#define TIME_TO_SLEEP 10 //スリープ時間を設定（秒単位）

の部分です。

数字の 10 がスリープ時間の秒数を決めていますので、例えば 1 時間おきに動作させる場合は、この数字を 3600 に変更します。数字は必ず半角で入力するようにしてください。スリープ時間を長くするほど、消費電力が下がるため、電池寿命が長くなります。

○作成するファイル名の変更

サンプルプログラムではSD カードに作成するファイル名を「test1.csv」としています。複数の装置を使用する場合は、ファイル名を変更した方が整理しやすいため、ファイル名を決定しているプログラムを修正します。

```
30  //SDカード用ライブラリ(ダウンロード不要)
31  #include "FS.h"
32  #include "SD.h"
33  #include "SPI.h"//WROOM
34  const char* fname1 = "/test1.csv";  //ＳＤカードのルートにtest1.csvを作成
35  //const char* fname2 = "/test2.txt";  //ＳＤカードのルートにtest2.txtを作成
36  File f1, f2;  //ファイルｆ１、ｆ２を用意、ｆ２は拡張用
37  enum { sd_sck = 19,sd_miso = 21,sd_mosi = 5,sd_ss = 17 };
38  //SDカードの2番ピンにSS17、3番ピンにmosi5、4番ピンに電源＋、5番ピンにSCK19、6番ピンに電源－、7番ピンにmiso21
```

ファイル名を決定している部位は、上記の３４行目

const char* fname1 = ″/test1.csv″;

の部分です。

“”の内側に記載された、/test1.csv がファイル名になるので、「test1」の部分を任意の名称に変更してください。この名称は半角英数字のみ対応可能です。「/」と「.csv」は変更しないようにしてください。

○RTC の取得時刻の修正

サンプルプログラムでは RTC は本体が再起動するたびに、現在時刻をインクルードしたタイミングの時刻にリセットします。これはRTCに現在時刻を登録するプログラムが、起動するたびに実行されることが原因なので、動作確認で RTC に現在時刻を記録した後はこのプログラムを削除する必要があります。

```
100     rtc.begin();
101   //rtc.adjustはPCからインクルードしたときのデータのみが反映される
102       rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));//インクルード時のPCの現時刻を使用
103     // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));//任意の時刻を使用
```

上記 102 行目の

rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));//インクルード時の PC の現時刻を使用

の部分で RTC が現在時刻を上書きしているので、この部分が動作しないように次のように変更します。

```
100     rtc.begin();
101   //rtc.adjustはPCからインクルードしたときのデータのみが反映される
102     //  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));//インクルード時のPCの現時刻を使用
103     // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));//任意の時刻を使用
```

102 行目の最初に//を追加しました。

//以下に続く文はコメント（メモ）として扱われるため、プログラムが実行されません。（コメントアウト）

これによって RTC の現在時刻の上書きはされなくなります。RTC の時間を再設定するためには、もう一度

rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));//インクルード時の PC の現時刻を使用

を有効化します。（インクルードした後でもう一度コメントアウトして上書きしないようにします。）

プログラムの修正が終わったら、SD カードに記録されるデータに問題がないことを確認してください。

実際に畑等で使用する際は、本体をタッパーや電工ボックス、ウォルボックス等に入れて使用します。容器には電動ドリル等で穴をあけるか、ボックスについているケーブル用の穴などを使ってセンサー類のケーブルを引き出せるようにします。

開口部からの浸水が考えられる場合は、防水パテやエアコンパテ、ヒートボンドやレジン樹脂などを使用して防水加工してください。

センサーの接続部分、電極の露出している部分などは多少の湿気であれば問題ありませんが、水がかかる、水没する等が生じると故障します。装置が完成して動作することが確認できたら、補強を兼ねてホットボンドやシリコンスプレー（シリコンオイルではない）を使用して防水対策を行ってください。スプレーを使用する場合は、センサーの測定部位にかからないように、養生テープ等で測定部位を覆っておく必要があります。

参考：サンプルプログラムの内容

サンプルプログラムの内容

サンプルプログラムは拡張子が ArduinoIDE 用の.ino になっているため、内容確認のためにこちらに転記しています。

実際の操作は ArduinoIDE 上で行ってください。

```
////////////////////////////////////////////////////////////////////////////////////////////////////
/*このスケッチは愛知県農業総合試験場において制作した、栽培環境モニタリング装置のためのサンプルスケッチです。
本スケッチによる機器の故障をはじめとしたその他トラブルに対して責任を負いません。

このスケッチでは、実際に装置を動作しデータの記録を行います。
点が変更されています。

測定装置本体について

ESP32DevkitC を使用し、マイクロ SD カードリーダー、RTC(DS1307）、mosfetk4017 を使用した測定装置本体を使用します。
測定用センサーとして、
・温湿度：AM2302（DHT22）センサー
・温度（地温・水温）：ds18b20 センサー（防水プローブ付き）
・土壌水分：METER 社　10HS センサー(静電容量式)
・EC：簡易 EC センサー（自作）
・日射：簡易日射センサー（自作）
を接続し、各データをマイクロ SD カードに任意の時間間隔（ディープスリープで設定した時間）で記録します。


*////////////////////////////////////////////////////////////////////////////////////////////////////

//ディープスリープ設定
#define uS_TO_S_FACTOR 1000000ULL  //ディープスリープの指定はマイクロ秒で扱いにくいので秒単位に変換
#define TIME_TO_SLEEP  10 //スリープ時間を設定（秒単位）
```

RTC_DATA_ATTR int bootCount = 0; //動作回数をカウントし、初回のみの動作を指定するために bootCount を用意

//mosfet 設定
#define mosfet 16

//SD カード用ライブラリ(ダウンロード不要)
#include "FS.h"
#include "SD.h"
#include "SPI.h"//WROOM
const char* fname1 = "/test1.csv"; //ＳＤカードのルートに test1.csv を作成
//const char* fname2 = "/test2.txt"; //ＳＤカードのルートに test2.txt を作成
File f1, f2; //ファイルｆ１、ｆ２を用意、ｆ２は拡張用
enum { sd_sck = 19,sd_miso = 21,sd_mosi = 5,sd_ss = 17 };
//SD カードの２番ピンに SS17、３番ピンに mosi5、４番ピンに電源＋、５番ピンに SCK19、６番ピンに電源ー、７番ピンに miso21

//RTC 用ライブラリ
#include "RTClib.h"//RTC 用ライブラリ（ライブラリマネージャーから取得）
#include "Wire.h"//I2C 通信用ライブラリ（任意のピンで RTC を起動するため）
RTC_DS1307 rtc;//1307 ユニットは5V を要求する

//AM2302(DHT 用)ライブラリ
#include "DHT.h"//（ライブラリマネージャーから取得）
#define DHTPIN 25//DHT22 の通信用に GPIO25 を使用
　　//GPIO25 と 3.3V の間に 4.7kω 抵抗をつなげること（プルアップ）
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

//ds18b20 用ライブラリ
#include <OneWire.h>
#include <DallasTemperature.h>//（ライブラリマネージャーから取得）
#define ONE_WIRE_BUS 26//ds18b20 センサーの通信用に GPIO26 を使用
　　//GPIO26 と 3.3V の間に 4.7kω 抵抗をつなげること（プルアップ）
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

```
//日射用設定
#define SRmVPin 34//簡易日射計として使用するソーラーパネルのプラス端子を GPIO34 に接続する。
                  //ソーラーパネルのマイナス端子はブレッドボード上の GND に配線する
                  //ソーラーパネルのプラス端子を接続した ESP32 の GPIO34 と、GND の間にまたがるように 5ω 抵
抗を接続する。

//EC 用設定
int R1= 510;//分圧用抵抗、電極に合わせて調整する
int Ra=22;  //本体の内部抵抗値、arduino Ra=25; esp32 Ra=22
int ECPin= 35;    //INPUT 分圧された電圧を読み取る。電極の一方を接続。もう一方は GND に接続。
int ECPower =32;  //OUTPUT 短時間 3.3V を出力するために必要。デジタルでもいいはず
                  //ECPin と ECPower との間に R1（510ω）を接続して R1 と測定対象で分圧する
float K=3;//セル定数：電極の距離、面積によって決まる。ここで校正する。
float soilEC=0;
float raw= 0;
float Vin= 3.3;  //ESP32 の基準電圧が 3.3V、Arduino なら 5V
float Vdrop= 0;
float Rc= 0;
float buffer=0;

//土壌水分用設定
#define soilmoistPin 33//土壌水分センサーの通信用に GPIO33 を使用する。



void setup() {
bootCount++;  // 起動回数カウントアップ
pinMode(mosfet,OUTPUT);
pinMode(SRmVPin, INPUT);
pinMode(ECPin,INPUT);
pinMode(ECPower,OUTPUT);
  R1=(R1+Ra);//分圧用抵抗と内部抵抗を合計

digitalWrite(mosfet,HIGH);//電源 on、これ以降各センサーと SD カード、RTC に電源が入る
delay(1000);//各センサー動作待ち
```

```
  Serial.begin(115200);
  dht.begin();
  sensors.begin();
    //Wire は前の数字が SDA、後の数字が SCL
  Wire.begin(22,23);//SDA(22),SCL(23)の順に入力する
  rtc.begin();
//rtc.adjust は PC からインクルードしたときのデータのみが反映される
  //  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));//インクルード時の PC の現時刻を使用
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));//任意の時刻を使用
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  sensors.requestTemperatures(); // Send the command to get temperatures
  float soiltemp = sensors.getTempCByIndex(0);

  int soilmoist = analogRead(soilmoistPin);

  int SRmV = analogRead(SRmVPin);

  digitalWrite(ECPower,HIGH);
  raw= analogRead(ECPin);
  delay(2);//esp32 の計算速度が速いので 2 ミリ秒待機:長いほど出力値は小さくなるが安定？
  raw= analogRead(ECPin);//静電容量の影響を抑えて再度計測
  digitalWrite(ECPower,LOW);
  Vdrop=(Vin*raw)/4096.0; //arduino では 4096→1024 に変更する。
  Rc=(Vdrop*R1)/(Vin-Vdrop);
  Rc=Rc-Ra; //acounting for Digital Pin Resitance
  float soilEC = 1000/(Rc*K);//セル定数が大きくなるほど分母が大きくなる。


  SPI.end();                              //SPI 通信を一度初期化
  SPI.begin(sd_sck, sd_miso, sd_mosi, sd_ss);  //指定した SPI のピンで通信を開始
  SD.begin(sd_ss, SPI, 24000000, "/sd");       //SD カードライブラリを始動、通信速度 24Mh z (CPU サイク
ル 80Mhz の 30%)
```

```
 f1 = SD.open(fname1, FILE_APPEND)；// ｆ１ファイルに fname1 を定義して追記形式でオープン、（write は
1 回上書き形式）
 if (bootCount == 1) {//bootCount が 1 の時だけ、項目名を記入する
 f1.println(F("date_time,Humid,Temp,soilTemp,soilmoist,SR,EC"))；// ｆ１、ｃｓｖ最上段に項目を記入
 }
   DateTime now = rtc.now();//現在時刻を取得し SD カードへ記入
   f1.print(now.year(), DEC);
   f1.print('/');
   f1.print(now.month(), DEC);
   f1.print('/');
   f1.print(now.day(), DEC);
   f1.print(" ");
   f1.print(now.hour(), DEC);
   f1.print(':');
   f1.print(now.minute(), DEC);
   f1.print(':');
   f1.print(now.second(), DEC);
   f1.print(',');

   f1.print(h);//AM2302 の湿度の値を入力
   f1.print(',');
   f1.print(t);//AM2302 の温度の値を入力
   f1.print(',');

   f1.print(soiltemp);//地温系 ds18b20 の測定温度を入力
   f1.print(',');

   f1.print(soilmoist);//土壌水分計 10HS の出力値（mV）を入力
   f1.print(',');

   f1.print(SRmV);//日射計として使用したソーラーパネルの出力値（mV）を入力
   f1.print(',');

   f1.print(soilEC);//簡易 EC センサーの出力値を入力
   f1.println(',');
```

```
  f1.close();  //各ファイルを閉じる

digitalWrite(mosfet,LOW);//電源 off
delay(1000);


//動作確認用シリアルモニタ表示
Serial.println("測定時刻は　");
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

Serial.println("温湿度は　");
    Serial.print(h);//AM2302 の湿度の値を入力
Serial.print("% ");
    Serial.print(t);//AM2302 の温度の値を入力
Serial.println("℃");

Serial.println("地温は　");
    Serial.print(soiltemp);//地温系 ds18b20 の測定温度を入力
Serial.println("℃");

Serial.println("土壌水分は　");
    Serial.print(soilmoist);//土壌水分計 10HS の出力値（mV）を入力
Serial.println("mV");

Serial.println("日射量は　");
```

```
    Serial.print(SRmV);//日射計として使用したソーラーパネルの出力値（mV）を入力
Serial.println("mV");

Serial.println("EC は　");
    Serial.println(soilEC);//簡易 EC センサーの出力値を入力


Serial.println("動作終了");
//ディープスリープ実行
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
  esp_deep_sleep_start();//スリープモードへ移行
}

void loop() {
}
```

以上