2023 年度　第 3 版

JSPS 科研費「ひらめき☆ときめきサイエンス」助成事業

IoT を活用した自動計測およびモバイル環境測定の実践

鹿児島大学　理学部　化学プログラム
神﨑　亮

Internet of Things

近年，「IoT」という言葉を聞くことが多くなりました．IoT は，Internet of Things，日本語では「モノのインターネット」と呼ばれます．24 時間コンセントにつながれ，常に起動している家電製品（例えばテレビは一般的に消している間も電力は供給されリモコンからの入力を待っています）をインターネットに接続することで，離れている場所でも情報のやり取りや制御ができるようなシステムです．例えば外出先から自宅のエアコンにアクセスし，室温を知ったり電源を帰宅前に入れておいたりすることができるようになると便利ですね．これを実現するには，小さなネットワーク機器を組み込む必要があります．IoT は，これらが小型化され，また安価で供給されるようになったこと，および自宅では WiFi，外出先では広域通信網が整備され，双方（機器とアクセスする人）がネットワークを常時使える状況が確立されたことで，急速に広まりました．

化学と IoT

我々化学者も，この恩恵に与る機会が多くなってきました．実験室では，時間がかかるような測定を自動化し，無人で動作させることも多いですが，例えば自宅でもデータを監視できると，不測の事態にも対応しやすいですし，任意のタイミングで次のステップに移るような操作もできます．研究現場ではギリギリの条件設定をすることも多いですから，測定を仕掛けた次の日の朝に「実験室に入るまで測定が成功したかどうか分からない」ようなドキドキする状況が避けられます．

また，遠隔地の環境測定にも役立ちます．例えば「1 時間に 1 回，1 カ月間の温度変化を調べる」ような観測は，これまで「データロガー」と呼ばれるものを測定箇所に置き，測定終了後に回収していました．これでは回収するまでデータが得られませんし，機器に不良が生じてもその時にしか分かりませんでした．しかしネットワークにデータを送信すれば，データは随時得られますし，電源さえあれば半永久的に観測を続けることができます．もちろん，極めて到達が難しいような観測対象など，以前からそのような方法も採られていた場面もありますが，今ではそれが安価で小さいデバイスで可能になってきました．

SoC (System on Chip)

IoT は，SoC により実現されます．SoC は「System on Chip」のことで，以前は「マイコン」などと呼ばれていたものの発展と思って良いでしょう．コンピュータ（計算機）の要素である CPU，メモリー，インターフェイスを，1 つの IC に組み込んだものです．広義にはスマートフォンやタブレットの心臓部も SoC ですが，特定の機種向けに専用に設計されたものばかりでなく，多様な用途に向けた，汎用的な SoC もあります．今回使うのは，そのような SoC の 1 つ，ESP32 と呼ばれる製品です．

Espressif ESP32

ESP32 は，Espressif Systems（上海）が開発した SoC で，本体は 1 cm 角よりも小さい IC ですが，この中に CPU，メモリー，インターフェイス（WiFi，Bluetooth，デジタル IO，各種センサー，各種通信機能）が内蔵されています．計算能力は，単純計算では現在主流の PC 用の CPU の数百分の 1（MIPS 単位）ですが，多くの計測や通信には十分です．実際には，外部にプログラム保存用のメモリーや WiFi アンテナなどを内蔵した基盤に載せられた ESP-WROOM-32 という状態か，これを USB 接続により PC からプログラム書き込みなどを容易にできるようにした DevKitC（ESP32 開発ボードとも呼ばれます）という状態で販売されます．ESP-WROOM-32 で 600 円程度，開発ボードで 1,500 円程度です．今回用いるのは，この開発ボードです．

ESP32 を動作させるには

ESP32 は基本的に IC です．電力を供給すれば動作しますが，その動作をプログラムすることができるというものです．「開発ボード」には，電力供給およびプログラムの書き込みを USB 経由で提供する機能が備わっています．デバッグや動作の追跡も，USB 接続した PC 上で行います．動作することが確認できたら，PC と切り離して，乾電池などの電力で作動させることもできます．

プログラムを書き込む方法も色々ありますが，情報が多く手っ取り早いのは，「Arduino IDE」を使う方法です．Arduino も，ESP32 と同様，SoC のシリーズです．（性能や価格の面では後発の ESP32 と比べると見劣りしますが，解説情報やキットなどは充実しています）．Arduino IDE では「Arduino 言語」が使われますが，基本的には C 言語の拡張のようですので，C 言語に慣れている人は無理なくプログラミングできるでしょう．

ただし今回は時間や装置の都合上，プログラムは書き込んだ状態で使ってもらいます．実習に使った ESP32 開発ボードは，希望するなら持ち帰って良いですので，自宅でプログラミングを挑戦してみてください．プログラムを書き込む方法は，資料の最後に簡単に解説していますが，インターネットで探した方が詳しく丁寧だと思います．

ESP32 は何ができるのか？

ESP32 開発ボードをよく見ると，2 列に計 38 本のピンが並んでおり，それぞれのピンに番号が振ってあります．数字以外の記号が割り当てられているピンもあります．抜けている番号もあります．数字が書いてあるピンは「GPIO」（General Purpose Input/Output）と呼ばれていて，ESP32 から任意のタイミングで仮想的なスイッチを ON・OFF して外部に接続した LED を点灯したり，外部に押しボタンスイッチを接続してその状態を読み取ったりすることができます．

ON・OFF の切り替えや読み取りは高速に動作することができるため，外部との通信に使うことができます．今回は，このことを利用して，外部センサーの制御とデータ取得を実現しています．また，別の IC と通信して液晶ディスプレイを制御したり，PC と通信することも可能です．

ESP32 の起動

ESP32 開発ボードにシルク印刷されているピン番号のうち，「GND」と「3V3」の間に規定の電圧（3.0～3.6 V）を供給する必要があります．GND（グラウンド）ピンは 2 本ありますが共通です．ESP32 起動中は「GND」と「3V3」に電圧がかかっていますから，これを周辺 IC の電源として使うこともできます．

ESP32 開発ボードを PC と USB 接続している場合は，USB からの供給電源を使うことができます．ただし USB の電源は 5 V ですから，開発ボードにはこれを 3.3 V に降圧する IC が搭載されています．今回は，アルカリ乾電池 3 本を「5V」ピンに接続（プラス極，マイナス極は GND に接続）します．最初の状態でおおよそ 4.5 V ですが，3.3 V の電圧を供給することができます．

起動には，「EN」（Enable）ピンを「3V3」と接続します．そうすると，保存してあるプログラムを読み込み，ESP32 が動作を開始します．これを「GND」と接続すると動作は停止し，再度「3V3」と接続すると最初から動作を開始します．つまりリセットがかかります．開発ボードでは，既にこのような回路が作ってあり，ESP32 の「EN」ボタンを押すと「EN」と「GND」が接触されます（つまり事実上リセットボタンとして機能します）．一方，GPIO 0 ピンを「GND」と接続したまま起動（「EN」と「3V3」を接続）すると，プログラムを外部から受信するモードになります．このことを利用して，PC で準備したプログラムを内蔵のメモリーに保存することができます．（これは「Boot」ボタンとして実装されていますが，開発ボードと「Arduino IDE」を使う場合は自動で制御されるので気にする必要はありません）

実験１：電解質の溶解熱の測定

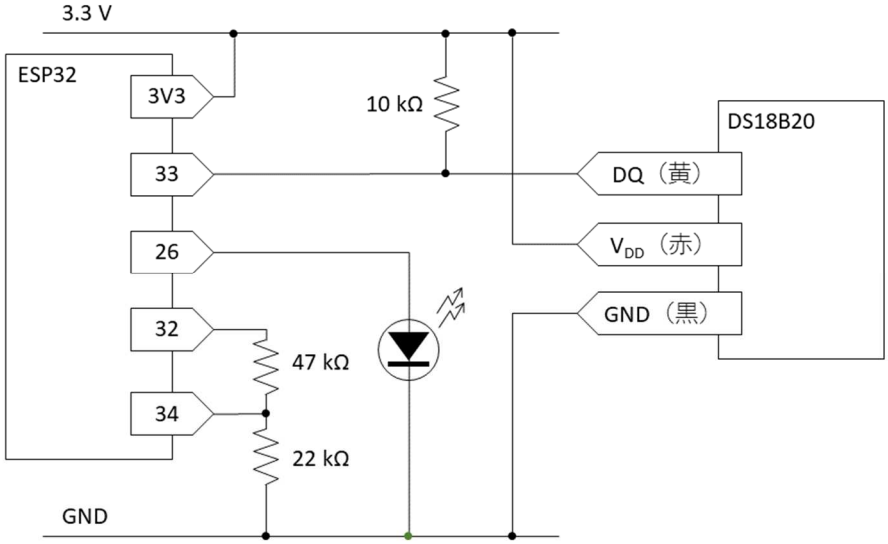
はじめに

試薬：塩化カルシウム，塩化ナトリウム，硝酸アンモニウム
器具：ビーカー，撹拌棒，温度計を接続した ESP32

実験 1 では，溶解熱測定の実習をします．3 種類の電解質を水に溶解し，その温度変化（経時変化）を測定することで，溶解熱を見積もります．温度変化をモニターするだけであれば，温度計を使っても良いですし，リアルタイムでデジタル表示する温度計も市販されていますが，今回は ESP32 を WiFi アクセスポイントおよびウェブサーバーとして作動させ，スマートフォンでデータを表示します．このことで，まずは ESP32 の性能を体験してください．

回路図と組み立て

温度計側部は DS18B20 と呼ばれるデジタル温度センサーIC がステンレス管に格納されています．この IC は，「1-Wire」と呼ばれる通信形式で通信し，測定した温度情報を伝えます．黒は GND（マイナス極），赤は電源プラス極（V_{DD}，3.0〜5.5 V），黄は信号線（DQ）です．これを，以下の回路図を参考に，ESP32（開発ボード，以下単に ESP32 と表記します）と接続します．



1．ブレッドボードに ESP32 を挿します．回路図通りに接続されてさえいれば，配置は任意ですが，ESP32 の位置が悪いと組み立てにくいので，ESP32 はブレッドボードの電源ライン（黒と赤の線が書いてある）を手前に置いき，右上隅より 1 段下に（最上段を 1 行空ける），USB コネクタを外側（右側）にして挿してください．
2．DS18B20 をブレッドボード左上の適当な箇所に挿し，GND（黒）と V_{DD}（赤）の間に

3.3 V の電圧をかけると DS18B20 が起動します．測定命令や温度データは DQ（黄）と ESP32 の「33」ピンを使った通信によって遣り取りされます．通信にどのピンを使うかは，プログラム内で指定することができます．

※ DQ と V_{DD}（または電源の赤ライン）を 10 kΩ 抵抗で接続する必要がある場合もあります．これはプルアップ抵抗と呼ばれるもので，回路がオープンなときに DQ の電圧レベルを 3.3 V 近くに保つためのものです．

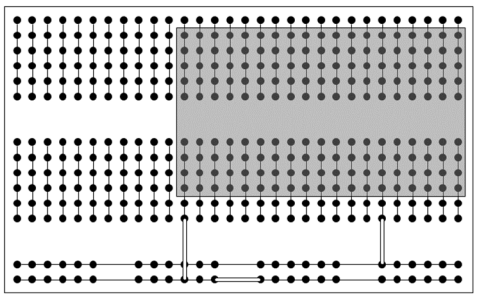
3．22 kΩ および 47 kΩ 抵抗を，回路図の通りになるように接続します．これは，供給電圧を測定するためのものです．「34」ピンを電圧計として動作させ，「32」ピンの出力を測定します．ただし 3.3 V を直接測定すると値が振り切れるので，2 つの抵抗で分圧しています．高校物理の範疇ですが，測定される電圧は 22/(22+47)となるので，プログラム内ではその逆数を掛けて出力するようにしています．また，そうすると常に電流が流れている状態になり，電池の消費が早くなってしまうので，測定の直前に「32」ピンを ON にします．「32」ピンは電源（「3V3」ピン）の電圧より低く出るとの情報もありますが，ほぼ同じ電圧が出力されるようです．

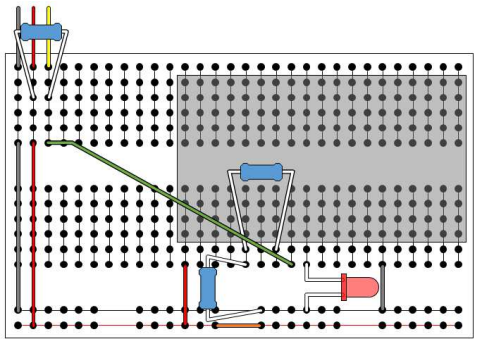
4．電池のプラス（赤いリード線）とマイナス（黒いリード線）を，それぞれ ESP32 の「GND」および「5 V」ピンに接続し，電源スイッチを ON にします．すると ESP32 に電力が供給され，起動します．

乾電池が消耗し，出力が低下すると，ESP32 を起動できなくなります．「32」ピンの電圧が 3 V を下回ったら，レギュレータを通さずに，電池の赤いリード線を直接「3V3」ピンと接続しましょう．このことによって，「3V3」ピンから直接，ESP32 に電力が供給されます．ただし，最初の状態（> 4.5 V）を直接「3V3」ピンと接続すると，電圧超過により ESP32 を破損する可能性があります．

ブレッドボードの使い方

今回，回路の組み立てに使う，格子状に穴がある板をブレッドボードと呼びます．穴の間隔は 2.54 mm（0.1 インチ）で，多くの IC や電子部品がこの間隔に合わせられています．穴の中には接点があり，素子を挿し込んだり抜いたりできます．接点間は図のように接続されています．ここに素子を差し込んで接続させ，回路を作ります．図の右上の影部分に，ESP32 開発ボードを挿します．下の 2 本は電源用で，黒ラインはマイナス極，赤ラインはプラス極です．今回のブレッドボードは赤ラインが分断されていて，ジャンパーワイヤで接続されています．これを外して，例えば左側は 3.3 V，右側は 5 V といった使い分けもできます．このような場合でも GND は共通で使います．開発ボードには GND のピンが 2 本ありますが，どちらを使っても構いません．（電流量が多くなるような使い方では，2 本とも黒ラインと接続した方が良い場合もあります）





左上のスペースに DS18B20 を挿し，ジャンパー線で GND，電源（赤ライン），および信号（33 番ピン）に接続します．10 kΩ 抵抗も，指定の接続になるように挿します．22 kΩ および 47 kΩ 抵抗，および LED は，場所が少なめですが，ESP32 の下のスペースで完結できます．一般的に，マイナス極は黒，プラス極（電源）は赤で配線しますが，今回は 1 色 1 本ずつですので，低彩度色（白や灰色）はマイナス（GND），赤，黄色系統はプラス，青や緑などは信号線など区別すると混乱しにくいでしょう．

温度測定

ESP32 が起動すると，WiFi アクセスポイントとして動作します．そこで，スマートフォンで ESP32 に接続します．SSID は「hirameki_esp32_****（****は ESP32 ごとに異なる 4 桁の 16 進数）」です．パスワードはかかっていません．接続すると，「信頼できないネットワーク」などの警告が出る場合があります．また「インターネット未接続」などと表示され，インターネットには接続できなくなります．しかし，ESP32 とスマートフォンのみからなるネットワークが出来上がっています．その状態で，ウェブブラウザ（Safari や Google Chrome）で「192.168.0.1」にアクセスしてください（実際には，http://192.168.0.1/）．このとき，ESP32 はウェブサーバーとして機能しています．アクセスがあると，DS18B20 で温度を測定し，その結果（および電源電圧と経過時間）を表示します．この間，26 番ピンに接続した LED が点灯するでしょう．このウェブページには，10 秒後に再読み込みするような命令が書き込まれているので，ウェブページは 10 秒おきに自動更新され，その度ごとに温度が測定されます．

実験操作

実験は，まず 100 mL ビーカーに蒸留水 50 mL を入れ，DS18B20 を浸し，撹拌します．10 秒おきに温度を測定し画面を更新するので，これを読んでグラフ用紙にプロットします．温度が安定したら，撹拌しながら，塩化カルシウム 5 g を加えます（撹拌し続けます）．3 分ほど観測したら測定をやめます．水は廃液溜めに棄て，ビーカーを蒸留水で洗ったのち，塩化ナトリウム，および硝酸アンモニウムで同じ実験を繰り返します．

解説

硝酸アンモニウムは「寒剤」として使われます．水を加えると吸熱して周囲の温度を下げるため，これを使った冷却パックが市販されています（これと尿素の粉末が混合されています）．一方，塩化ナトリウムも「寒剤」として使われることがあります．小・中学校の実験で，氷水の温度を低下するのに使ったことがあるかもしれません．しかし塩化ナトリウムを溶解しただけだと，硝酸アンモニウムほどには温度は下がりません．塩化ナトリウムによって水温が下がるのは，塩化ナトリウムによる凝固点降下によるものであり，実際に水温を低下しているのは共存している氷です．一方，塩化カルシウムは融雪剤や除湿剤として使われます．塩化カルシウムは発熱的に溶解しますから，少しでも溶けるとその熱でさらに氷を溶かします．凝固点降下によって，やはり水溶液の温度は低下しますが，再凍結しにくくなりますので，「凍結防止剤」としても作用します．（鹿児島では出番は少ないですけどね！）

なぜ吸熱的な電解質と発熱的な電解質があるのでしょうか？「溶解」も，1 つの重要な化学変化であり，結晶状態から水溶液中で電離した状態に変化します．電離状態では，単にバラバラになっているだけでなく，陽イオンと陰イオンの双方が，周囲の水分子と相互作用して（水和と呼ばれます）います．溶解熱とは，この反応の反応熱であり，反応前（結晶）と反応後（水和イオン）の結合状態の差が，反応熱に関わってきます．一般的に，2 価イオンはより強く水和されます．結果として，塩化カルシウムが電離する時に，より発熱的となります．一方，硝酸は塩化物イオンよりも水和が弱いので，電離反応に熱を奪われることになったのだと説明されます．しかし，電解質の結晶のイオン結合を切断してバラバラにする（電離する）にはかなりのエネルギーの供給が必要（吸熱的）である一方で，同じくらい水和によるエネルギー的な安定化（発熱的）も大きいため，収支としての溶解熱を予測するのは困難です．ここでは，水和エネルギーがイオン結合（結晶の凝集エネルギー）に匹敵するほど大きいのだということに注目してください．

なお，それぞれの物質の式量と溶解熱は以下の通りです．塩化カルシウム：111 g/mol，81 kJ/mol，塩化ナトリウム：58.4 g/mol，−4 kJ/mol，硝酸アンモニウム：80.0 g/mol，26 kJ/mol（化学便覧「溶解エンタルピー」より）．

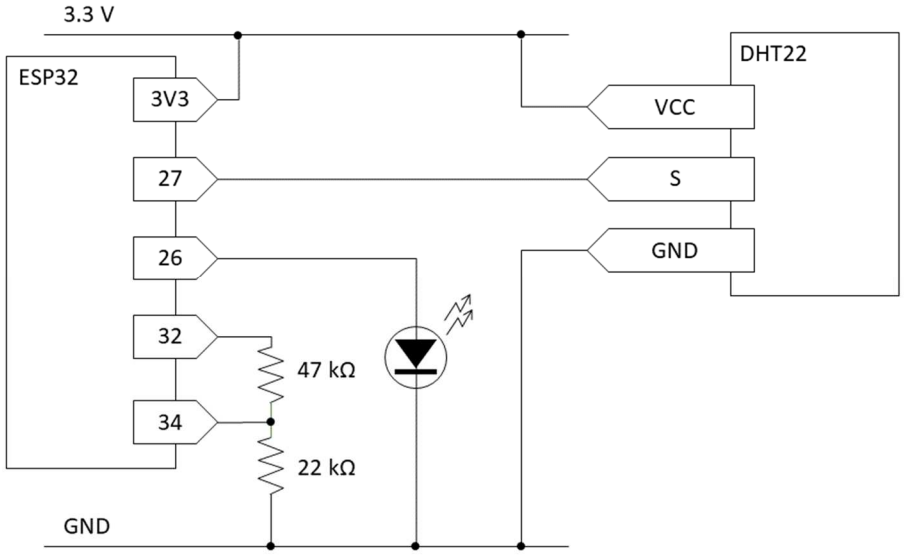
実験２：モバイル環境測定

はじめに

実験 2 では，屋外で環境（気温，湿度，および気圧）を測定します．さらにその情報をクラウドサービスにアップロードします．このサービスにブラウザでアクセスすると，スマートフォンやタブレット上でグラフ化されたデータを表示してくれます．乾電池で動作しますから，これをサンプリングしたい箇所に置いておくと，電源の続く限り，データを送信し続けてくれることになります．

回路図と組み立て

今回使うセンサーは，DHT22 と呼ばれるセンサーです．駆動電圧は 3.3～5.5 V となっており，ESP32 の入力電源を共用できます．ESP32 との通信は 27 番ピンを使うようにプログラムされています．なお，LED（GPIO 26）および電池電圧測定部分は，実験 1 の場合と同様です．



1．ブレッドボードに ESP32 開発ボードと LED，22 kΩ および 47 kΩ 抵抗を接続します．「34」ピンで，「32」ピンの電圧測定も行います．これらは，実験 1 で接続したままの状態で使うことができます．

2．DHT22 をブレッドボード左上の適当な箇所に挿し，GND と電源のマイナス極（黒），VCC と電源のプラス極（赤），ESP32 の「27」ピンと DHT22 の「S」ピンを接続します．

3．電池のマイナス極（黒いリード線）を黒ラインに，プラス極（赤いリード線）を ESP32 の「5V」ピンに接続します．正しく接続されたら，電源スイッチを ON にします．LED が点灯したのち（1 秒弱）消灯したら，正しく起動しています．

動作・データ転送

今回のデータは,「Ambient」というクラウドサービスに転送します．これにより，測定データをスマートフォンで図示・確認できます.

ESP32 は起動したら WiFi ルーターに接続します．この時点で，ESP32 はインターネットに接続されました．WiFi ルーターに接続できることが確認できたら，30 秒に 1 回，BME280 で気温・湿度・気圧を測定し，Ambient に送ります（Ambient は 1 日で 3000 点までとなっているので，1 点当たり 30 秒弱）．Ambient にブラウザでアクセスすると，送られてきたデータをグラフで表示されます. 自分のスマートフォンで見てみてください.

「Ambient」https://ambidata.io/
メールアドレスは hirameki_ambient_**@envchem.sci.kagoshima-u.ac.jp
（**は 01 または 02）
パスワード・チャネルは当日お知らせします.

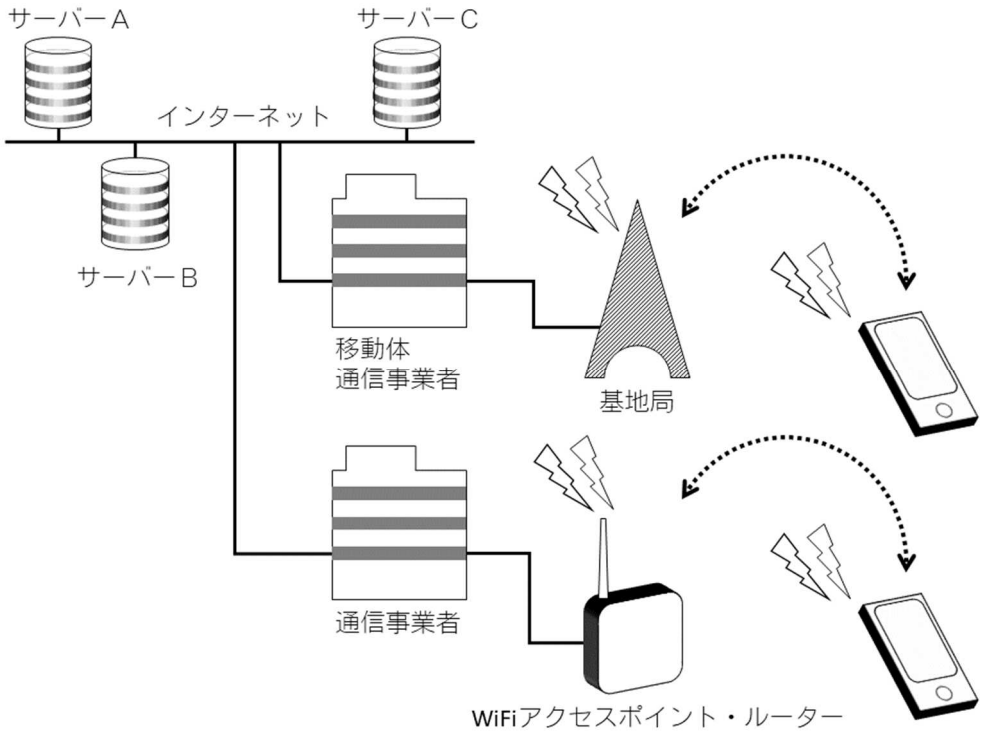
ログインしたら，自分のチャネル番号を開いてください．データがグラフになって表示されると思います．グラフ上部の「チャート…▼」をクリックすると，データ表示範囲などを変更することができます.

屋外へ出て, 気温や湿度の変化が反映されるか, 確かめてみましょう. ただし今回は, セキュリティの都合上，PC のモバイルホットスポット機能を使ってインターネット接続を提供していますので，WiFi の届く範囲内でしか行動できません.

ESP32 を使う

ESP32 のネットワーク機能

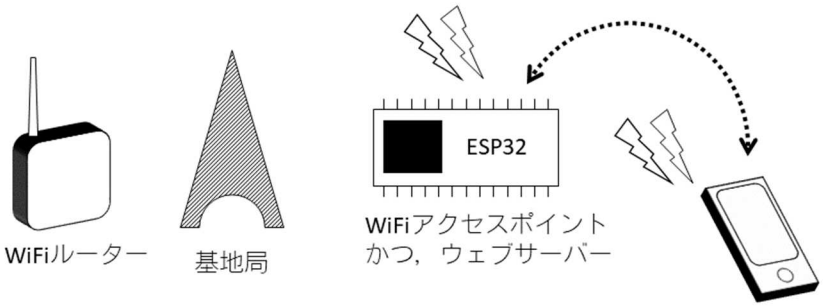
ESP32 のネットワーク機能を見る前に，まずインターネットの仕組みをおさらいしましょう．スマートフォンでネットニュースや Google 検索，ブログなどを閲覧するときの通信経路は，下に示すようになっています．



移動体通信事業者とはドコモや au などのことで，街中のあちこちに通信基地局を設置し，スマートフォン（携帯電話）を捕捉し通信します．いわゆる「ギガを使う」通信では，その基地局を通してインターネットに接続しています．また，自宅内に有線インターネット接続環境があれば，WiFi ルーターを設置し，これを通じて接続することもできます．WiFi ルーターは，WiFi アクセスポイント（WiFi 端末と無線通信してネットワークを形成する）とルーター（このネットワークをさらに別のネットワークと接続する）を兼ね備えた機器です．

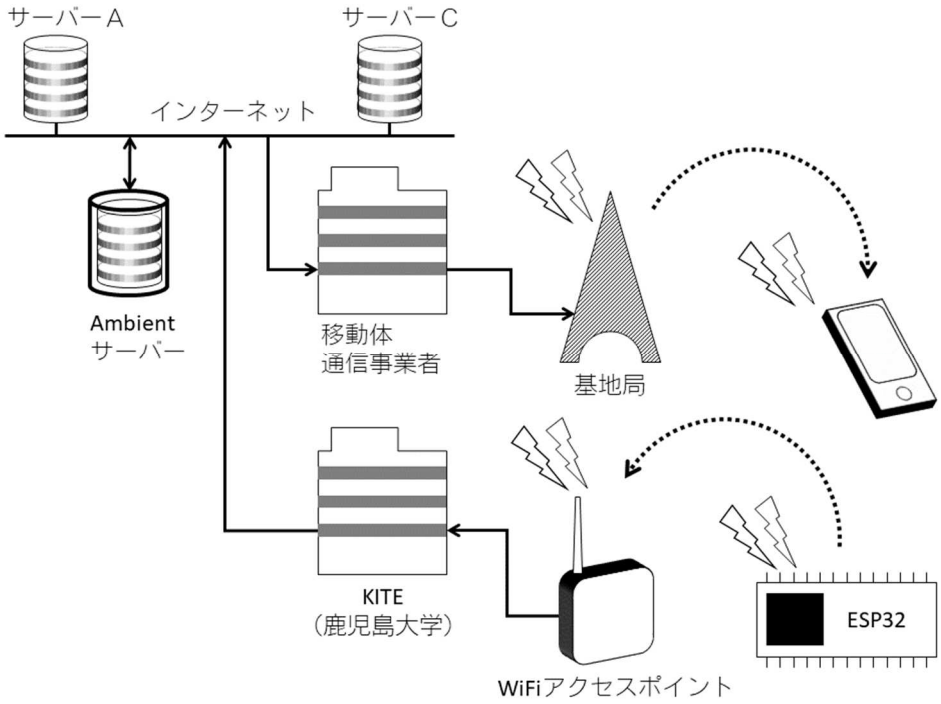
インターネットの先に，目的のサービスを提供するサーバーが接続されています．ブラウザ（ウェブブラウザ，Microsoft Edge や FireFox，Google Chlome，Safari など）は，指示されたサーバーにアクセスしてデータをダウンロードし，表示しています．

さて実験 1 では，ESP32 に，WiFi アクセスポイントとしての機能を持たせていますが，ルーター機能は提供していませんので，ESP32 とスマートフォンとの 2 台だけの隔離されたネットワークが形成されることになります．



この間，基地局や WiFi ルーターとの接続は制限されますので，普段アクセスしているウェブサイトや LINE などのインターネット通信は遮断されます（電話は通じます）．しかし，ESP32 がウェブサーバーとしての機能も果たすため，ESP32 だけには，アクセスすることができます．ESP32 は，アクセスがあったら，温度を計測し，そのデータをウェブページと同じ形式（html）にして応答します．これで，スマートフォンのブラウザでデータを見ることができるのです．

実験 2 では，インターネットを使った通信が発生します．ESP32 は，大学内のアクセスポイント経由で大学内ネットワーク，その先のインターネットに接続されています．気温・湿度・気圧を測定したら，そのデータをインターネット経由で「Ambient」のサーバーに送ります．一方，スマートフォンは，契約している移動体通信事業者の通信網を通して，「Ambient」のサーバーからデータをダウンロードして表示します．図では，データの流れを矢印で模式的に示しています．



ESP32 の開発環境

ESP32 にプログラムを書き込むには，「Arduino IDE」というソフトウェアを使うのが一般的ですので，ここではその流れを簡単に説明します．より詳細・丁寧な説明は，インターネットで「esp32 arduino ide」などで検索してください．

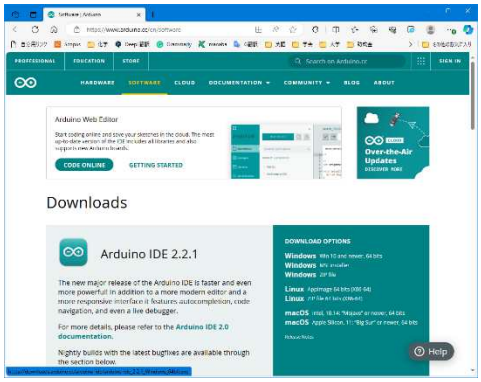
1．Arduino IDE のインストール

まず，以下のサイトから，適切なものをダウンロードします．（Windows の場合，

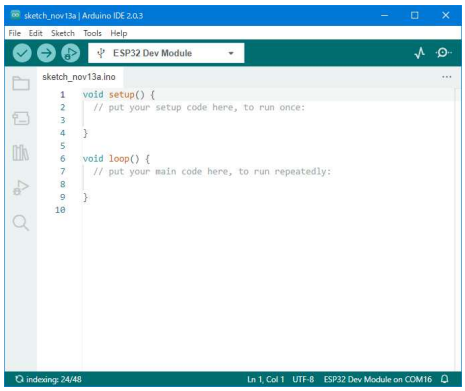
https://www.arduino.cc/ （Arduino のサイト）

https://www.arduino.cc/en/software （上記サイト → SOFTWARE）

2023 年の時点で，バージョンは 2.2.1，「Windows」の最上段をクリックして入手できるファイルがインストーラーです．「JUST DOWNLOAD」を選んでダウンロード・インストールしてください．日本語表示が良ければ，メニューバー（上段）の File → Preferences... を選択し，「Language:」で「日本語」を選択しましょう．



Arduino IDE ダウンロード

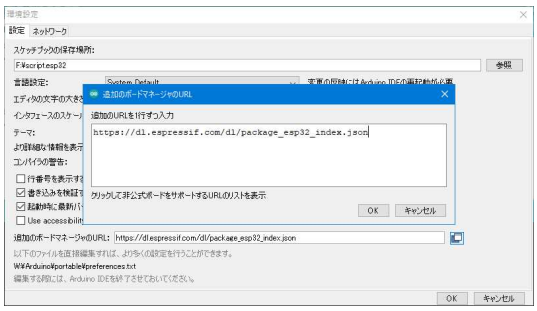


Arduino IDE 起動画面

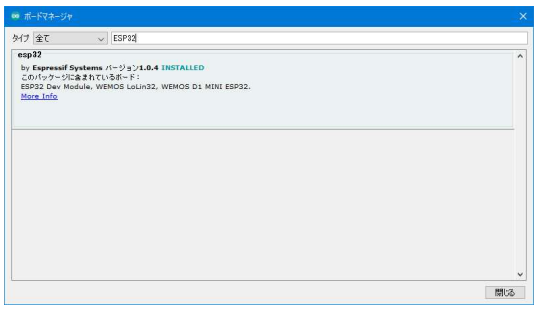
2．Arduino IDE の ESP32 への対応

Arduino IDE が起動したら，ファイル → 環境設定で，「追加のボードマネージャの URL」の右のボタンをクリックし，「追加のボードマネージャの URL」ウィンドウで，以下の URL を指定します．

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



Arduino IDE 環境設定



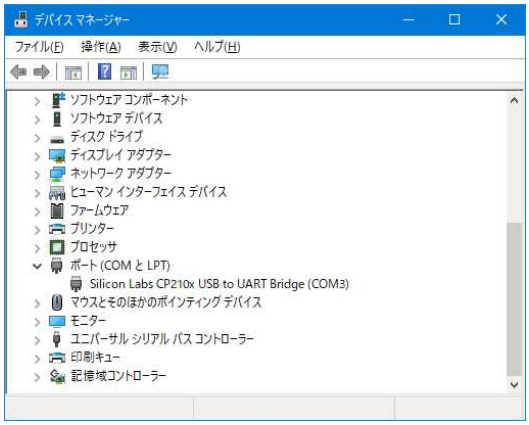
ボードマネージャ画面

「OK」ボタンをクリックして戻ったのち，メニューの[ツール] → [ボード：...] → [ボードマネージャ]の検索欄で「ESP32」と入力すると（検索結果が表れるのに時間がかかるかもしれません），「**esp32 by Espressif Systems**」というのが見つかりますので，[インストール] ボタンでインストールします．インストールできたら，メニューの[ツール] → [ボード] → [esp32] → [ESP32 Dev Module] を選択します．

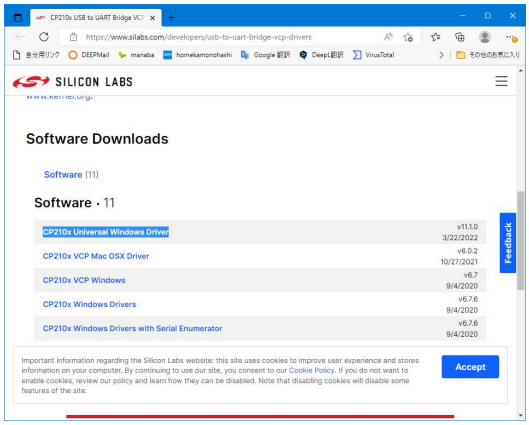
3．ESP32 開発ボードを Windows に認識させる

次に，ESP32 開発ボードを USB に接続します．データ通信ができる USB Micro B ケーブルを使って接続して下さい．以下，Windows の場合を説明します：最初に接続する時に，ドライバ（周辺機器を制御するソフトウェア）をインストールしようとします．デバイスマネージャー（ファイル名を指定して実行 → devmgmt.msc）の「ポート(COM と LPT)」のところに「Silicon Labs CP210x USB to UART Bridge」が出ていれば，正常に動作しています．同じ行に，「COM」に続いてポート番号が表示されていますので，それを控えておいてください．もしこのデバイスが「ほかのデバイス」にリストアップされていれば，ドライバがインストールされていません．右クリック→ドライバの更新で，正しいドライバを設定します．ドライバは，ダウンロードした Arduino IDE の「Drivers」に入っているようです．また，以下の Silicon Labs のウェブサイトから「CP210x Universal Windows Driver」をダウンロードすることもできます．デバイスマネージャーで，正しく認識されていることを確認し，ポート番号を控えてください．

https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers



デバイスマネージャー



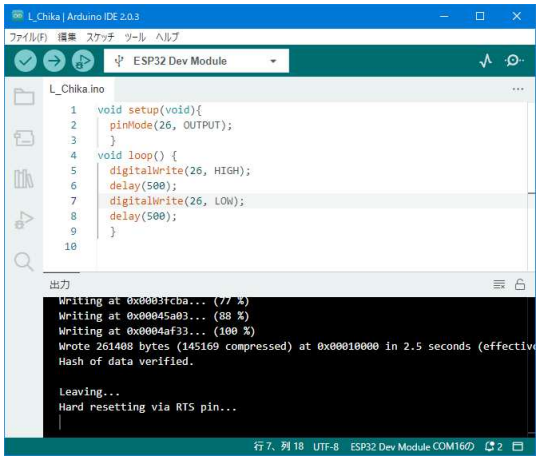
Silicon Labs のドライバダウンロードページ

Arduino IDE に戻り，[ツール] → [シリアルポート：...] → [COM...] から，接続されているポート番号を選択してください．これでプログラムを書き込む準備ができました．Arduino IDE は，Mac 版，Linux 版もあるようです．

4．いわゆる「L チカ」をやってみよう

これで開発環境の出来上がりです．ESP32 の「26」ピンに LED の長い足（実習と同じ

位置),「GND」ピンに短い足を接続し，Arduino IDE で，右図のプログラムを書き込みます．書き込んだら，転送（スケッチ → 書き込み 選択，または画面の上の方にある右向き矢印 をクリックすると，自動的にコンパイル（ESP32 が認識できる形式に変換）して ESP32 に書き込みし，実行を開始します．Arduino IDE の下半分に,「Leaving…」「Hard resetting via RTS pin…」と表示されたら，正しく書き込まれ，実行されています．1 秒間に 1 回，LED が点滅しましたか？このような，とりあえず ESP32 のピンのはたらきを確かめる小さなプログラムは,「L チカ」と呼ばれています．ウェブ検索すると多くの情報があるので，上手くいかない場合は調べてみましょう．



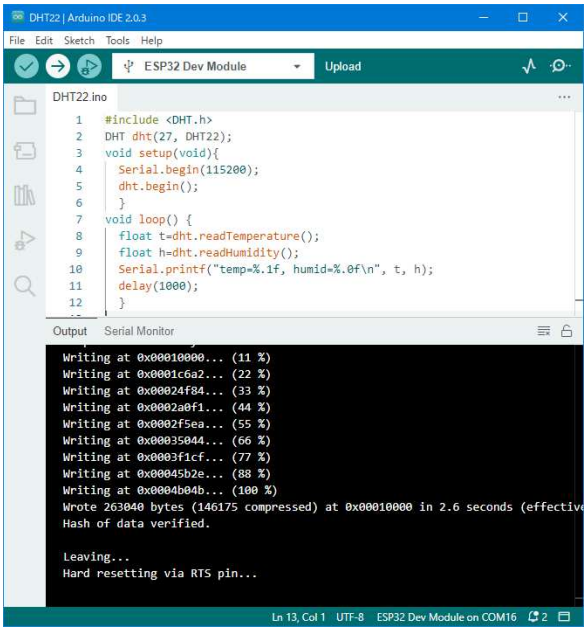
5．必要なライブラリを追加する

最後に，DHT22 を使う準備をします．Arduino IDE の スケッチ → ライブラリをインクルード → ライブラリを管理… を選択すると「ライブラリマネージャ」ウィンドウが開くので，以下のライブラリを検索（テキストボックスに入力）してインストールします（マウスカーソルを乗せると インストール ボタンが表示されるので，クリックする）．

- **DHT sensor library by Adafruit**
- **Adafruit Unified Sensor** by **Adafruit**

（DHT sensor をインストールしようとすると，一緒にインストールすることを促されます）

DHT22 を，実習の通りに接続し，以下のプログラムを書き込んで，実行してみましょう．ツール → シリアルモニタを選択すると，DHT22 から読み込んだ温湿度情報を出力します．DHT22 本体を温めたりすると温度表示が上昇しますか？もし，正しく書き込まれているにもかかわらず，うまく作動していないようでしたら，[EN]ボタンを押してリセットしてみてください．



プログラムの内容については，ここでは説明しませんが，何となく意味は分かるのではないでしょうか．1 行目・2 行目は，DHT22 を使うための準備です．3～6 行目は，ESP32 起動時に 1 回だけ呼

ばれる部分で，初期化などを行います．7〜12 行目は，ESP32 起動中に常に実行される部分です．「delay」の括弧内はミリ秒指定です．ここでは，8・9 行目で DHT22 から温度情報を読み取り，10 行目で「シリアルモニタ」に出力，11 行目で 1 秒待ってから，同じ処理を繰り返します．

```
 1:  #include <DHT.h>
 2:  DHT dht(27, DHT22);
 3:  void setup() {
 4:      Serial.begin(115200);
 5:      dht.begin();
 6:      }
 7:  void loop() {
 8:      float t= t=dht.readTemperature();
 9:      float h=dht.readHumidity();
10:      Serial.printf("temp=%.1f, humid=%.0f\n", t, h);
11:      delay(1000);
12:      }
```

Ambient を使ってデータを送信する場合も，比較的容易にできます．以下のライブラリが必要です．（インストールする手順は DHT22 と同じ）

Ambient ESP32 ESP8266 lib by **Ambient Data**

また，この場合あらかじめ Ambient のチャネルを作っておく必要があります（実習で使ったものは将来削除しますので使わないでください）．「Ambient ESP32」など検索すると，詳しい説明が見つかると思います．

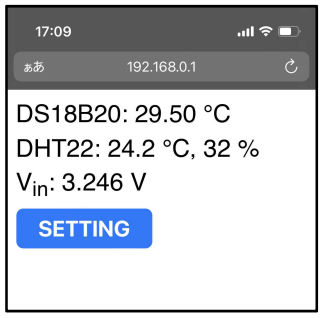
持ち帰った ESP32 をそのまま使う

本日の実習の終了時に，希望すれば ESP32 を持ち帰っても構いません．開発環境を整えて，自分でプログラムを書き込んでも構いませんが，単独でも機能するようなプログラムを書き込んだうえでお渡しします．ここでは，その利用法を説明します．

注 1：説明の通りに進まなかった場合の対策にはネットワークに関するある程度の知識が必要で，また設定にはご自宅の状況などの情報が必要となりますので，質問されても対応するのは困難と予想されます．誠に申し訳ありませんが，**このプログラムの利用法に関する質問はご遠慮ください**.

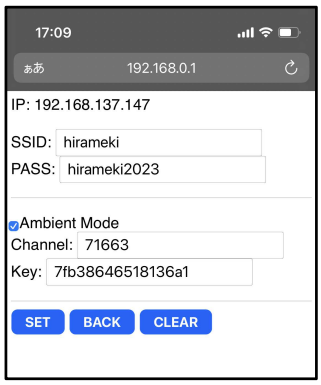
注 2：ESP32 はネットワーク通信の能力を持っていますから，セキュリティリスクがあります．特に無線機器は，物理的に離れた場所からの攻撃を可能にします．**いかなる場合であっても，セキュリティに関する責任は負えません．**使わないときは電源を入れないなど，気を付けて運用してください．

初期モード



ESP32 を起動する（USB で PC や USB 電源と接続するか，実習のように乾電池を接続する）と，LED が 5~10 秒ほど光ったのちに，早い点滅となります．このとき，「hirameki_esp32_****」（****は実習と同じ固有の 4 桁）という ssid で接続を待ち受けます．これが初期モードです．パスワードは当日お伝えします．ESP32 に WiFi 接続し，ウェブブラウザで http://192.168.0.1/ にアクセスすると，DHT22 のデータが表示されます．

WiFi モード



自宅に WiFi ネットワークがある場合は，SETTING ボタンをクリックして表示された画面で自宅の ssid とパスワードを入力し SET をクリックします．そのあと，[EN]ボタンを押して（手動で）リセットしてください．再起動後，LED が点灯し，自宅の WiFi に接続すると消灯します．通常は数秒内に接続できますが，やはり 10 秒ほど光るようでしたら，再度リセットしてみてください．

自宅の WiFi に接続できていれば，「SETTING」画面で「192.168.」から始まる IP アドレスが表示されます．この IP アドレスで，同じネットワークに接続された他の PC やスマートフォンからアクセスすることができます．

Ambient モード

「SETTING」画面で,「□Ambient」をチェックし，チャネルとライトキーを入力すると，自宅の WiFi を経由して，測定データを定期的（30 秒おき）に Ambient サーバーに送信するモードになります．Ambient のチャネルは，Ambient ウェブサイトであらかじめ作成しておく必要があります．チャネル ID およびライトキーは，Ambient でユーザー登録してログインし，「チャネルを作る」から新規作成することができます．

https://ambidata.io/

設定完了したら（手動で）リセットしてください．その後，30 秒ごとに LED が点灯します．（Ambient にデータ送信するためには，WiFi への接続が必要です）

Ambient モードでは，ネットワークからの接続を受け付けません．通常の WiFi モードに戻すためには，リセット後すぐに（WiFi に接続するために LED が光っている間に）「Boot」ボタンを押し続けてください．うまく WiFi モードにもどる準備ができたら，LED が 2 回点滅しますので，そのあと再度リセットしてください．

なお WiFi の情報は ESP32 に残っていますので，破棄・譲渡する際は，「SETTING」画面で CLEAR ボタンを押すと消去すると良いでしょう．また，このあとリセットすると，初期モードで起動します．