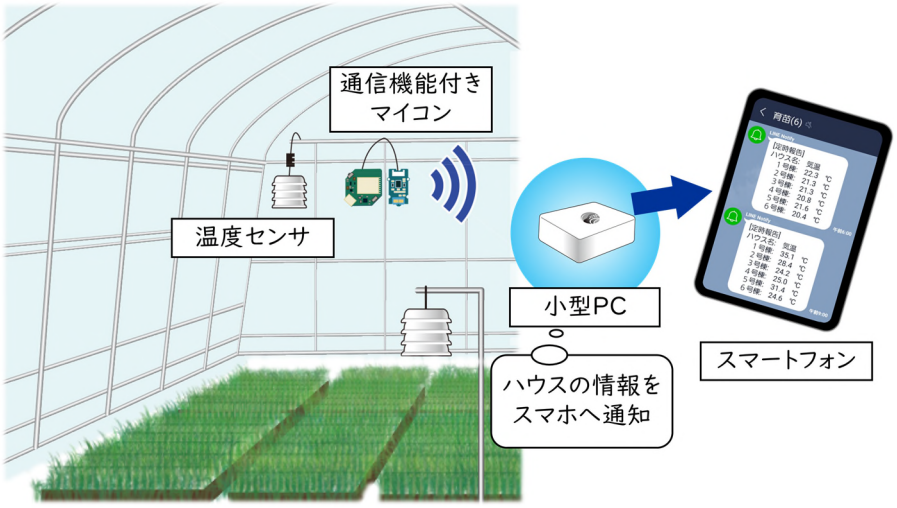
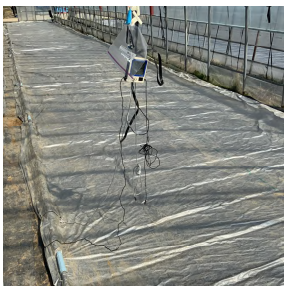
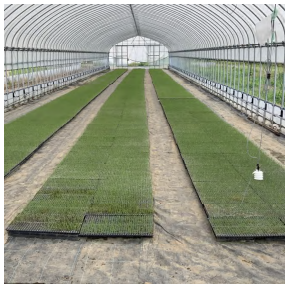
安価かつ簡便に
ハウスの遠隔監視に使える
IoT 機器「通い農業支援システム」
製作マニュアル







農研機構
NARO 国立研究開発法人 農業・食品産業技術総合研究機構

農研機構　東北農業研究センター

目次

免責事項

■本マニュアルを用いて作成する「通い農業支援システム」、「配布プログラム」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、一切の責任を負わない。
■マニュアル記載の配布プログラムの商業的な利用又は配布は、農研機構に連絡する。
■農研機構は、配布プログラムに関して不具合やエラーや障害が生じないことを保証しない。
■農研機構は、配布プログラムに欠陥があると判明した場合、訂正や補修する義務を負わない。

はじめに

本マニュアルは「通い農業支援システム」の製作方法を説明したものです。

本システムはモジュール化された簡便な IoT 機器を利用して、離れたハウスの温度等をスマホで遠隔監視できるようにするハウス遠隔監視システムです。

ハウスごとに通信機能付きマイコンと温度センサを設置して遠隔監視システムを構築するには、IoT に関する知識や WEB サーバの構築またはクラウドサービスの契約が必要であり、容易には作成できません。そこで、データ取得用・メッセージ通知用の Web API*を利用することで、(初歩的なプログラミングの知識など) 一定の知識があれば利用できる、ハウス温度等の遠隔監視を「安価かつ簡便に」実現するシステムを提案しています。

本マニュアルでは「IoT 温度計の設定」、「スマホのメッセージアプリの設定」、「小型パソコン (Raspberry Pi) でのプログラム」「プログラムの自動実行」「現場での設置方法」「導入事例」について説明しており、「温度等のデータを定期通知」、「しきい値を超えた際の警報通知」、「日平均値・最大値・最小値の通知」、「グラフでの通知」といった機能を持つシステムを作成できます。

*HTTP からはじまる URL を用いてサーバ側のアプリケーションを実行する仕組み

通い農業支援システムの目的

被災地では大規模水田輪作やハウス栽培による営農再開が先行しています。
しかしながら、震災被災地の営農再開地域では以下の問題があります。

■少ない担い手で大規模化・分散化した農地・ハウスを管理している
■水稲育苗では既存ハウスに一時的に利用できる遠隔監視システムが求められている
■育苗と圃場作業が同時期で労働負荷が大きい
■見まわりなど生産管理の省力化対策が求められている
■遠隔監視システムは有用だが、1年中使用することが前提の装置は過剰投資となる
■ハウスを新たに建設し、灌水や側窓開閉を自動化したが遠隔監視機能はついていない

本マニュアルで紹介する「通い農業支援システム」を導入することで、遠隔地からスマートフォンで温度、湿度、土壌水分などを定期的に確認でき、取得データは平均値やグラフで確認できるため、見回りの管理の負荷を軽減できるなど、これらの問題の解決に役立ちます。

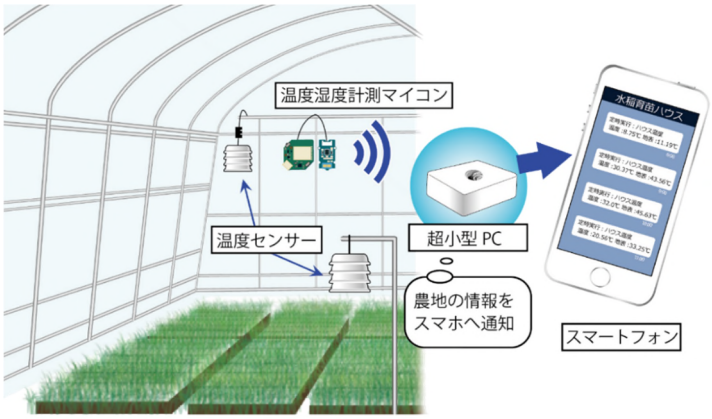


図1 通い農業支援システムのイメージ

通い農業支援システムのコスト

1. 通い農業支援システムの構成

通い農業支援システムは以下の3つから構成されています。

(1) マイコン (Seeed 社製 Wio Node*) およびセンサ
(2) データの取得や通知を行う小型パソコン Raspberry Pi
(3) メッセージアプリ (本マニュアルでは LINE を例にしています)

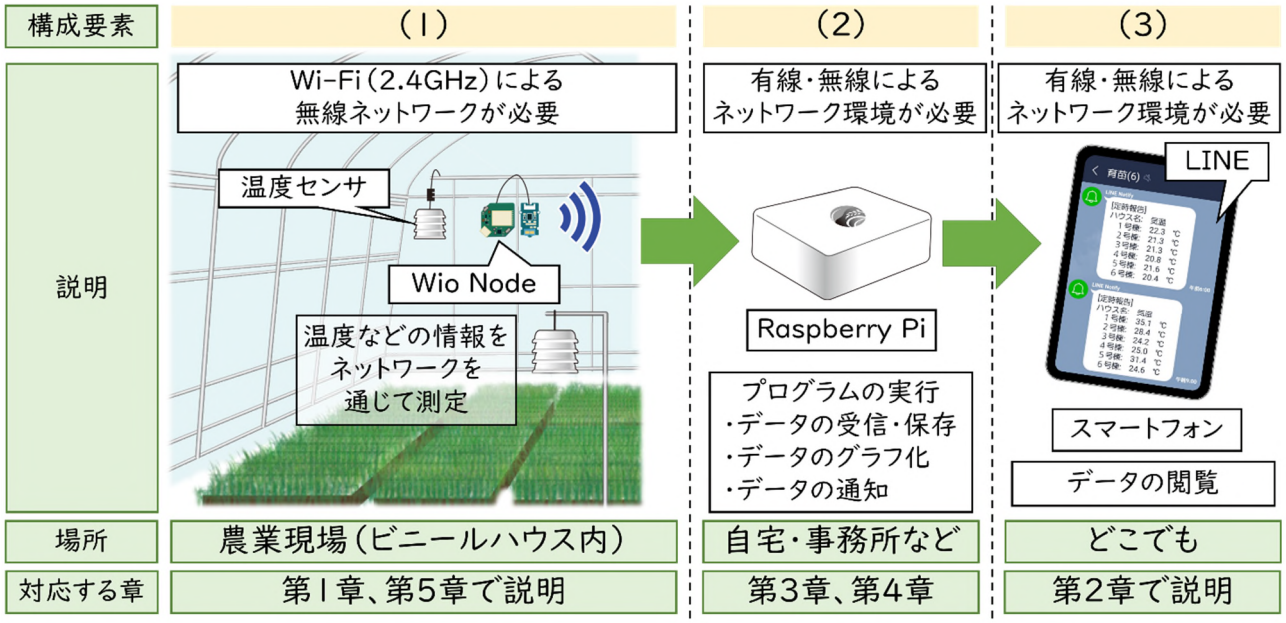


図2 通い農業支援システムの構成要素とその役割

設置場所ごとに必要となる資材について簡単に説明していきます。なお、詳しい設置方法は第 5 章で説明しますので、使用する材料については適宜第 5 章を確認してください。今回はハウス1棟から 6 棟を上限に温度センサを 1 つずつ設置するパターンについて説明します。

ハウス内には (1) の Wio Node や温度センサを設置する必要があります。Wio Node の設置には電源が必要です。100V 電源から一般的なスマートフォンの充電器に使われる USB-AC アダプタを使って、USB ケーブルで接続します。100V 電源が無い場合は太陽光発電システムを設置する必要がありますが、これについてはイニシャルコストの試算をしません。

データの取得や保存、グラフ化などといった処理を行う (2) の Raspberry Pi はインターネットを通じてデータを取得・通知します。そのため、ビニールハウス内やその周りに設置する必要は無く、インターネットが利用可能な自宅や事務所などに設置します。マウスやキーボードなどの周辺機器は新品で揃える必要が無いため、コスト試算には含めません。本体基板とケースのみを想定しています。

データの閲覧にはメッセージの送受信に必要な Web API を備えるメッセージアプリであれば使

*「ワイオノード」と呼びます

用できます。今回はLINE社のメッセージアプリ「LINE」を例にしますのでコスト試算には含めません。

2. 通い農業支援システムのイニシャルコスト（1 棟から 6 棟の場合）

ハウス1棟から 6 棟を上限に通い農業支援システムを導入した際のイニシャルコストは 1 棟では約 2 万円、上限の 6 棟の場合は 1 棟あたり約 1 万円となります。表 1 の黄色の網掛け部分は 1 棟ごとに必要となる資材となります。

表1 通い農業支援システムの導入コスト試算

	ハウスの棟数	1棟	3棟	6棟
（1）	マイコン（Wio Node）	1,300	3,900	7,800
	防水温度センサ	1,000	3,000	6,000
	電源用USBケーブル	110	330	660
	USB延長ケーブル	500	1,500	3,000
	100V電源延長コード他*	5,000	15,000	30,000
	USB ACアダプタ	1,000	3,000	6,000
	Wi-Fiルータ**	5,000	5,000	5,000
（2）	Raspberry Pi 3B+***	6,000	6,000	6,000
	合計	¥19,910	¥37,730	¥64,460
	1棟あたりの費用	¥19,910	¥12,577	¥10,743

*100V 電源延長コード他の内訳は、「100V 電源延長コード（1,500 円）」、延長コードを入れて USB AC アダプタを保護する「コンテナ（1,000 円）」、「コンセント差し込み型の漏電ブレーカ（2,500 円）」を想定しています。

**Wi-Fi ルータは 100V で動作する市場流通品のホームルータ（例えば Speed Wi-Fi Home L01s 等）やモバイルルータを想定しています。

***Raspberry Pi 3B+は本体のみを想定しています。周辺機器一式が整っているスターターキットであれば約 1 万円となります。

自宅の前にビニールハウスがある場合は、自宅の Wi-Fi ネットワークが利用できることがあります。その場合は Wi-Fi ルータは不要となります。

既に自動灌水装置などを導入しているハウスであれば、ハウス内に配電盤があることが多いです。その場合はコンテナや 100V 延長コードを利用せず、配電盤内に USB-AC アダプタを設置することで 100V 電源延長コード等を省略することができます。また、配電盤のブレーカが漏電ブレーカであれば、コンセント差し込み型の漏電ブレーカを省略することが可能です。

また、100V 電源が無い場合、太陽光発電システム（第 5 章で説明）を作成することで運用が可能ですが、冬季間を除いて 1 年間使用できるものを作る場合、少なくとも約 3 万円必要となりますので、今後ハウスに自動側窓開閉装置や自動かん水装置などを検討しているのであれば、100V 電源を設置する工事とどちらが安価かを検討する必要があります。

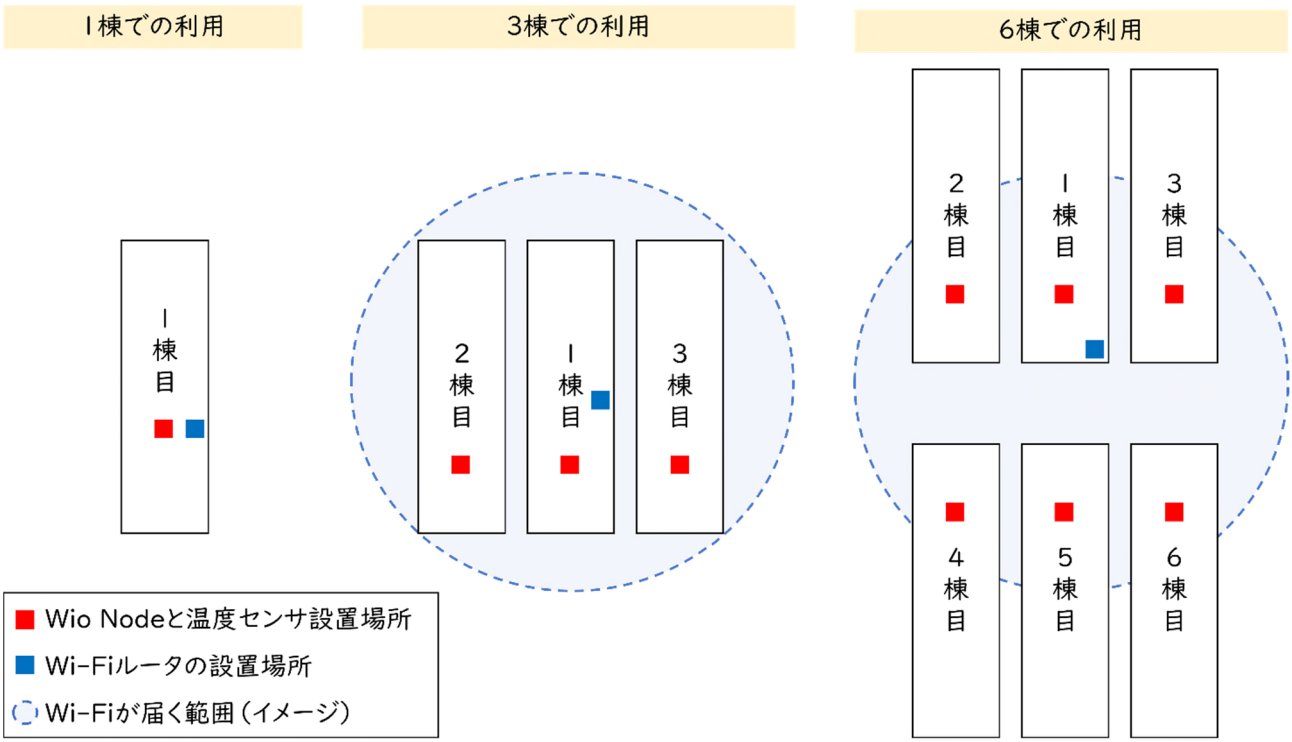


図3 通い農業支援システム導入のイメージ

複数棟で利用する際は、Wi-Fi ルータの設置場所を工夫してできる限り1台の Wi-Fi ルータで無線ネットワークの範囲を広くすることがコツです。ただし、通常の Wi-Fi ルータは同時接続台数が 10 台（ホームルータでは最大 20 台ほど）ですので、温度以外にも他に測定したい項目がある、あるいは多点で計測したい場合は Wio Node の数が増えるため複数の Wi-Fi ルータが必要となることに注意してください。また、市販の Wi-Fi 中継器を用いて無線ネットワークの範囲を広げることが可能ですが、Wi-Fi 中継器自体も接続数にカウントされますので注意が必要です。

Wi-Fi ルータは製品によって異なりますが、Wi-Fi の電波はハウス内に設置した場合は直線距離で約 30~40m ほど届きます。用意した Wi-Fi ルータでどこまで電波が届くかは事前に確認する必要があります。今回 6 棟で使用する場合は、中央のハウスの入り口に設置し、他の隣接するハウスでは電波が届く範囲に Wio Node を設置することを前提としています。

3. 使用に伴うランニングコスト

通い農業支援システムで使用するインターネット回線は高速かつ大容量である必要はありません。そのため、格安 SIM と呼ばれる月額 1,000 円でデータ容量 3GB 程度の物を選択すれば十分に利用できます。使用する Wi-Fi ルータで利用可能な格安 SIM を選択しなければいけないことに注意が必要です。

4. 通い農業支援システムの技術的な仕組み

ハウスの遠隔監視システムを製作するには、「マイコンから無線でデータを取得する仕組み」を作り、「データの取得」と「データの閲覧」機能を備える必要がありますが、簡単ではありません。

この仕組みを実現するためには、マイコン本体を「Web サーバ」とよばれるものにし、データを蓄積して、インターネットを通じてマイコン本体にアクセスしてデータを確認する方法があります。パソコンをハウス内に置いて、外部からそのパソコンにアクセスするイメージになります。また別の方法は、マイコンからデータを貯めるためのクラウドへ自動でデータをアップロードする方法です。これはプログラミングの知識のある方であれば作成できますが、いずれの方法もマイコンにプログラムを書き込む必要があり、簡単ではありません。

そこで、「通い農業支援システム」では「データの取得を簡易に行える Web API が用意されたモジュール化された IoT 機器（本マニュアルでは Wio Node）」を使用して作ります。データ取得用の Web API が用意されていると、図 4 のような仕組みで簡単にデータを取得できます。

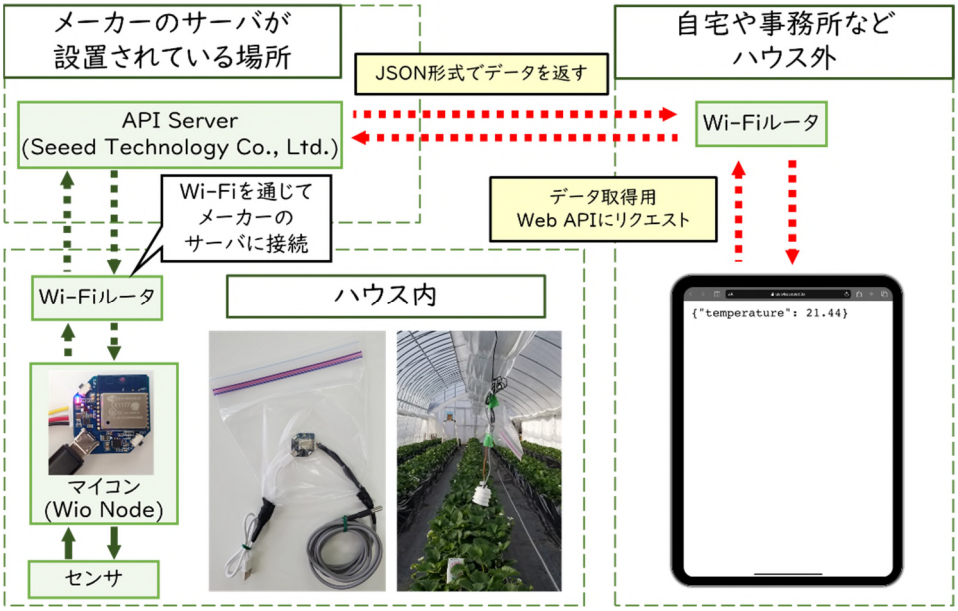


図4 Web API を利用したデータ取得の仕組み

スマートフォンからデータ取得用の Web API にアクセス（図 4 中の赤い点線）すると、自宅や事務所の Wi-Fi を通じてメーカーのサーバにデータを取得するよう指示（リクエスト）を出します。メーカーAPI サーバはハウス内に設置した Wio Node のセンサデータを取得し、スマートフォンにデータを返します。

このように Wio Node と温度センサは前述のように数千円と、市販のインターネットにつながる 1～2 万円する温度計と比べ安価です。そのため、ハウス遠隔監視システムを安価に作成することが可能となります。

*Wio Node は Seeed Technology Co,. Ltd の製品で、データ取得用 Web API が無償で利用可能です。

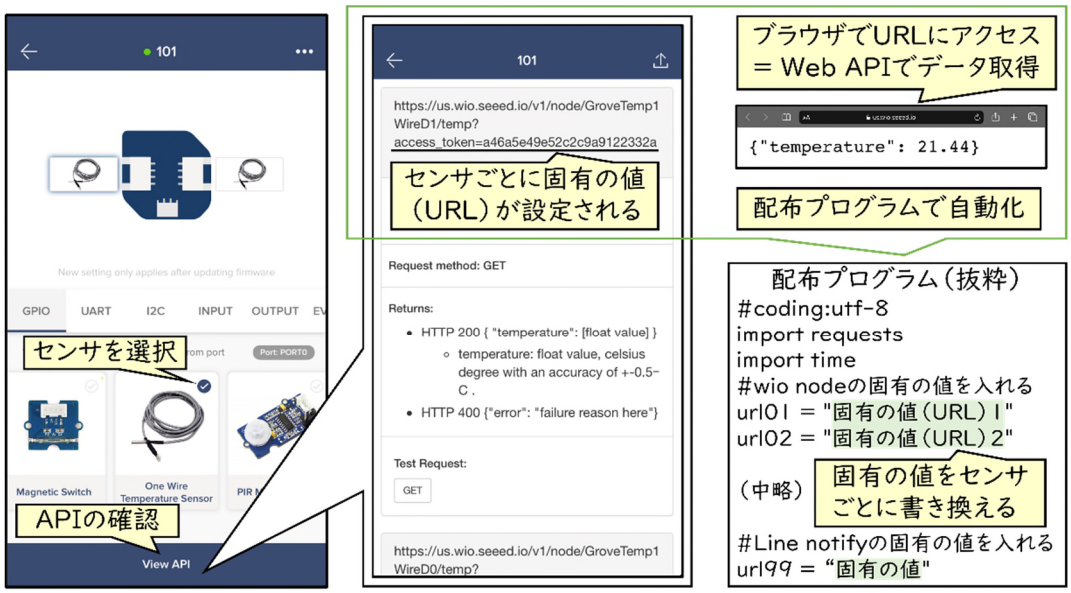


図5 データ取得用 Web API のアクセストークンの設定方法と自動化方法

さて、データ取得用の Web API にアクセスする仕組みを図 4 では赤い点線で示しましたが、実際には図5のような設定が必要です。第 1 章で説明しますが、Wio Node ではスマートフォン上のアプリでマイコンを設定し、使用するセンサを決めることができます。使用するセンサを決めると、センサごとにアクセストークン*が設定されます。Wio Node の場合は、「https://から始まる URL にアクセストークンが含まれた文字列」（固有の値）が設定されます。この固有の値をブラウザにアクセスすることでデータが取得可能です。

しかし、これでは手動でしかデータを知ることができません。そこでこの操作を自動化して、スマートフォンに通知するプログラムを「通い農業支援システム」では用意しています。配布プログラム中では、センサごとに設定される「固有の値（URL）」を書き換えることで自動実行できるようになります。なお、配布プログラムの入手方法や利用については第 3 章で詳しく説明します。

ここまで「データ取得用の Web API が用意されてる機器は、データの取得が比較的簡便に行える」という「データの取得」機能について説明をしてきましたが、ここからは「データを閲覧する」機能をどのように実現するかについて説明します。

Web サーバやクラウドを利用したハウス遠隔監視システムは、ブラウザで直接 Web サーバにアクセスするか、専用のアプリケーションを利用してデータを閲覧します。この方法でも、いつでもどこでもデータを確認することはできますが、普段使っているメッセージアプリでデータを確認することができれば、より簡便に利用することが可能です。

例えば「データを確認して、従業員に指示を出す」といったことをする際に、「専用のアプリを開く」⇒「メッセージアプリで従業員に連絡」といった流れよりは「メッセージアプリでデータを確認」⇒「メッセージアプリで従業員に連絡」の流れの方が簡便です。そのため、「通い農業支援システム」ではデータをメッセージアプリに通知することにしています。

*ユーザーを判別する固有の値

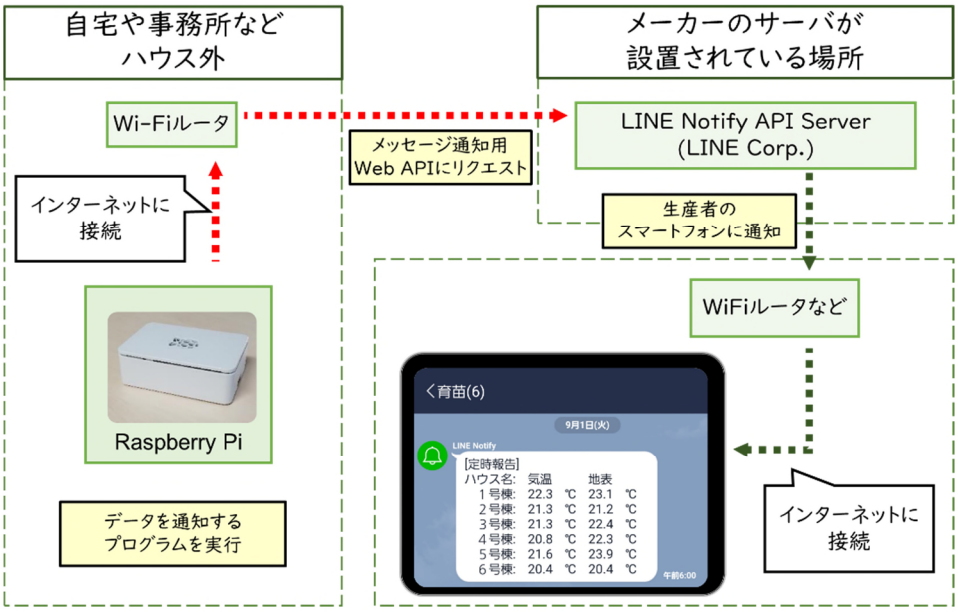


図6 メッセージ通知の仕組み

メッセージアプリにはメッセージ通知用の Web API が利用されているものを使用する必要があります。そのため、今回は LINE を利用することとしています。Raspberry Pi でデータを通知するためのプログラムを実行すると、LINE 社のメッセージ通知用の API サーバにデータを渡します（赤い点線）（図 6）。これだけでスマートフォンにデータを通知することができます。*

配布プログラム（抜粋）

```
（中略）

#LINE notify のURL
#LINE notify のトークン（通知先に応じて変更すること）

url99 = "https://notify-api.line.me/api/notify"
token = '通知するLINEグループのアクセストークン'

#LINEに通知するメッセージを記入 ''は文字列のこと
message  = 'ハウスの情報\n '
#message += を使うと通知メッセージを増やせる
#センサ名、データ番号、単位を書く
message += '温度1:'+str(data1)+'℃'+'\n'
message += '温度2:'+str(data2)+'℃'+'\n'

#LINEに通知するための3行
payload = {'message' : message}
headers = {'Authorization' : 'Bearer '+ token,}
r = requests.post(url99,data=payload,headers=headers)
```

アクセストークン

通知するメッセージ

Web APIサーバに
アクセストークンと
メッセージを渡す

図7 配布プログラム中のメッセージ通知部分の例

ただし、LINEのどのグループに（あるいはどのユーザーに）通知するかを設定する必要があります。これには、Wio Node と同様に、LINE のメッセージ通知サービスである「LINE notify」を利用し、アクセストークンを発行する必要があります。アクセストークンの発行は第2章で説明しますので割愛しますが、LINE 社の Web API サーバに「通知するメッセージ」と「アクセストークン」を渡せば指定したグループにメッセージを通知することが可能です（図 7）。

*メッセージ通知用の Web API が用意されているアプリケーションは LINE や Slack があります。第3者が個人情報にアクセスできる状態となっていた事例もあることに留意して利用してください（p.98参照）。

ここまで、「通い農業支援システム」の「データ取得」と「データ通知」について説明しました。

しかし、例えば「10 分おきの温度」が知りたいのか、それとも「平均温度」や「グラフ」などデータを処理することで傾向を知りたいのかは生産者によって異なります。生産者にとって「利用しやすい形」でデータを通知する必要があります。そこで「通い農業支援システム」ではデータの取得、保存と通知を行う「データ通知プログラム」と、10 分おきに保存したデータ（瞬時値）を「平均値」や「グラフ」に処理してから通知する「グラフ通知プログラム」を用意しています（図 8）。

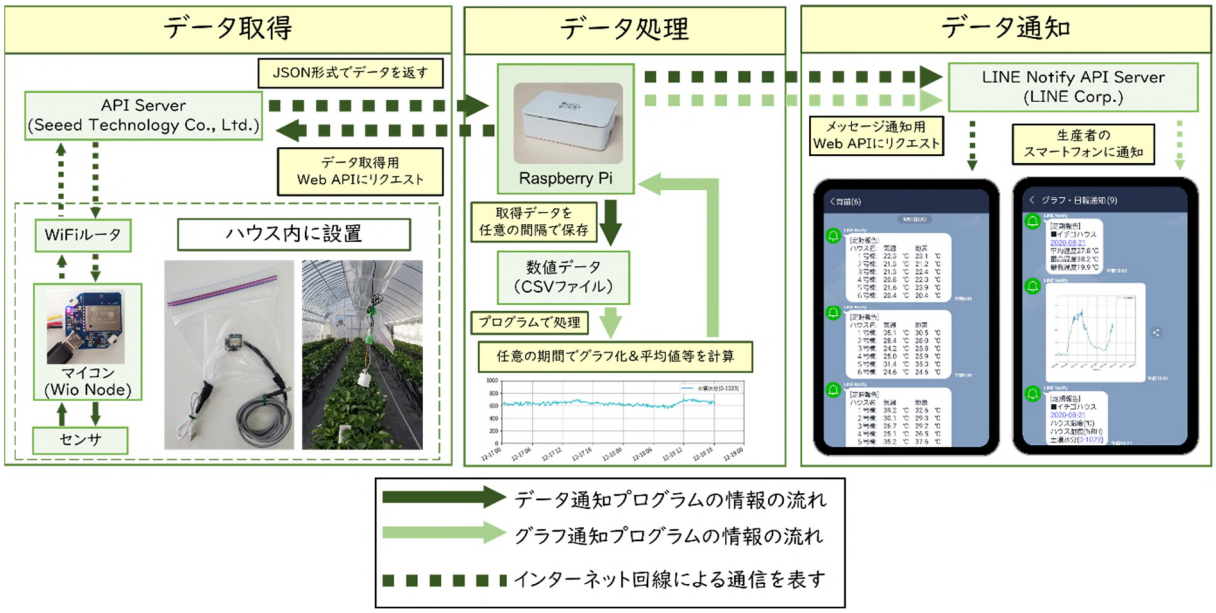


図8 「通い農業支援システム」の仕組み

なお、プログラムの実行タイミングは、Raspberry Pi OS に付属する「プログラムを任意のタイミングで自動実行する機能」である「crontab」を利用して設定しますが、第4章で説明するためここでは説明を割愛します。

「通い農業支援システム」の技術的な説明は以上となりますが、配布プログラムは Python と呼ばれるプログラム言語で書かれています。Python はインターネットを介してデータをやり取りしたり、データを処理してグラフを作成したりすることが比較的簡単です。そのため、多くの入門書やインターネット上に解説が書かれています。本マニュアル中で説明に用いている用語や、プログラムの書き方についてわからない点があれば、まずはインターネット上で検索し調べてみることをお勧めします。

なお、最近のインターネットにつながるスマート機器はデータ取得用の Web API が公開されているものが増えてきています。本システムでは農業用途に利用できるセンサの豊富さと、本体が安価であるため Wio Node を用いていますが、Web API を後述するような JSON 形式でデータを取得できる機器であれば、他の機器からデータを取得することも技術的には可能です。ただし、本システムでは対象としておりませんので、ご了承ください。また、配布プログラムはβ版（プロトタイプ）であり、Raspberry Pi の OS のアップデートや、メーカーの Web API サーバの仕様変更による不具合には対応いたしかねますので、ご留意ください。

第 1 章 IoT 温度計の製作

簡便な IoT 機器を活用した「通い農業支援システム」は、
以下の3つを使って、「ハウスの温度を計って、スマートフォンに通知」します。
そのうち、1 の IoT 温度計の部分が第 1 章となります。

1.温度センサ付きのマイコン（IoT 温度計）を組み立てて設定をする
2.スマートフォンのメッセージアプリ（LINE など）の設定をする
3. 1と2を自動実行する配布プログラムを小型 PC（Raspberry Pi）で動かす

1. 材料を確認しましょう

Wio Node

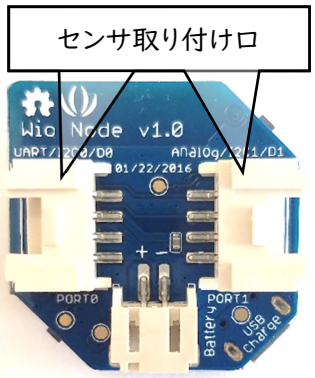

温度センサ


micro USB ケーブル


USB-AC アダプタ

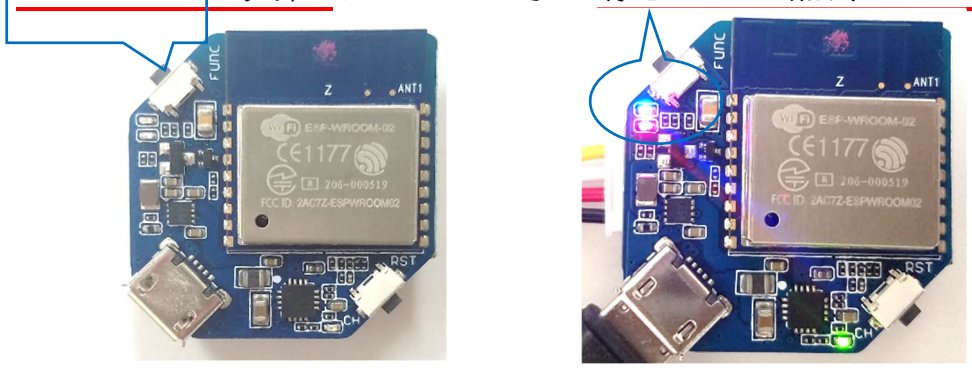
2. 温度計を組み立てましょう

センサと micro USB ケーブルを Wio Node に接続します。

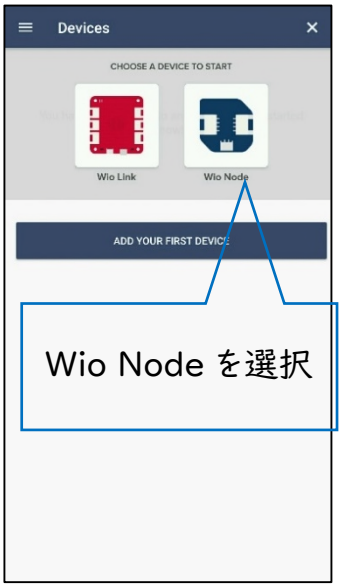


3. 温度計をスマホと接続しましょう（先にソフトをインストールしておくとスムーズです）

イ. func ボタン を長押しするとボタン近くの青色 LED の点滅がゆっくりになります



ロ. 専用ソフト「Wio」(iOS は「Wio Link」)をインストールし、スマホに機器を登録します。Android 版*を例に説明します。（Google Play や App Store からインストールします）

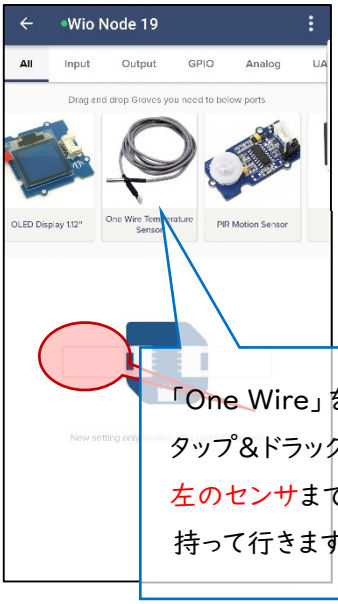


（注意）NEXT を押す前に･･･

スマホの設定を確認
1.GPS を ON
2.モバイルネットワークを OFF
3.WiFi を ON にして,
SSID： 接続先 Wi-Fi
Pass ： 接続先 Wi-Fi の
パスワードに接続

*「Wio」ver.2.5.2 / Android 10
「Wio Link」ver.1.5.6 / iOS 14.3 動作確認済

ハ. 温度センサを機器に登録します。



Update Firmware に
失敗した場合は、
この手順を
もう一度行ってください。

【設定をする際のポイントと注意】

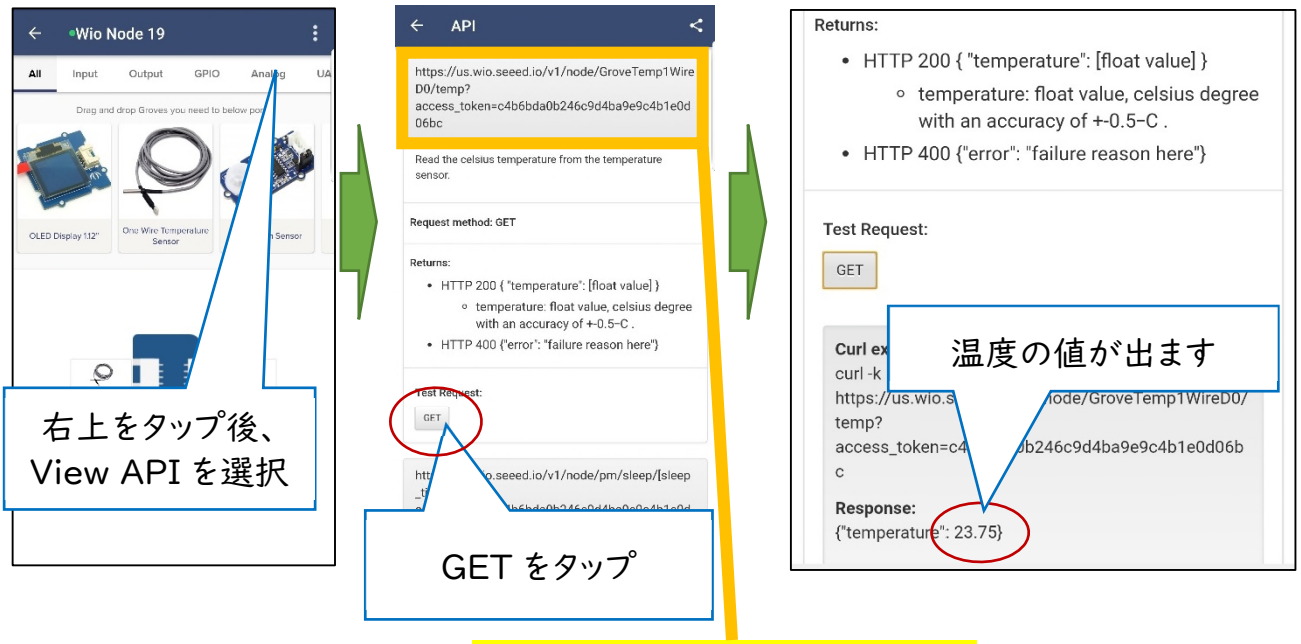
- iOS 版では設定画面や設定手順が一部異なりますが、同様の設定ができます。Android 版の手順を開始する前に、接続用の Wi-Fi と接続しておく必要があります。「接続用の Wi-Fi と接続」⇒「Setup Your Wio Node まで進める」⇒「iOS の設定画面を開いて、Wio Node の SSID と接続する」⇒「アプリに戻って設定を進める」の順に進めてください。残りは Android 版とほぼ同様です。

- 複数の Wio Node を設定する際は 1 台ずつ行い、設定が完了した Wio Node は一度電源を切ってから次の Wio Node の設定を開始してください。
（Wi-Fi ルータへの接続数が増えると回線が混雑して設定に失敗します）

- 接続する Wi-Fi は 2.4GHz の SSID と接続してください。5GHz の SSID とは接続できません。

- 一部のスマートフォンでは設定できません。また、OS の更新がされた場合は、Wio Node 設定アプリのアップデートがされるまで設定できなくなることがあります。
（Wio Node の Func ボタンを押すことで、Wio Node が設定用の電波を発し、Wio Node の SSID を選択し接続することで、Wio Node とスマートフォンを接続しています。一部のスマートフォンは接続した SSID でインターネット通信できない場合、自動的に接続を切断してしまうため設定できません。）

4. 温度を計ってみましょう



View API の画面に出てくる https://us/wio.seeed.io/····· の URL が重要です。この URL をアクセストークン*と呼びます。

*ユーザーを識別する固有の値のこと

この URL をスマートフォンのブラウザでアクセスすると、以下のように表示が出ます。

ブラウザでアクセスすることで、メーカーのサーバ（Web API サーバ）を介してデータを取得しています。この操作を自動化するためにプログラムを作成する必要があります。

（本マニュアルではすぐに使える配布プログラムのほか、修正に必要な知識を得るためのサンプルプログラムを掲載しています。）

5.【重要】温度計のアクセストークンを保存する

API 画面右上に「共有ボタン」があるので、そこをタップすると View API の画面にアクセスできる URL をコピーできます。コピーして保存しておきましょう。紙に書き留めてもよいですが、「スマホのメールで自分に送る」「メモアプリ」に保存するなどすると便利です。しかし、この URL を知られると他の人にもアクセスされてしまうので、保管方法には十分気を付けてください。

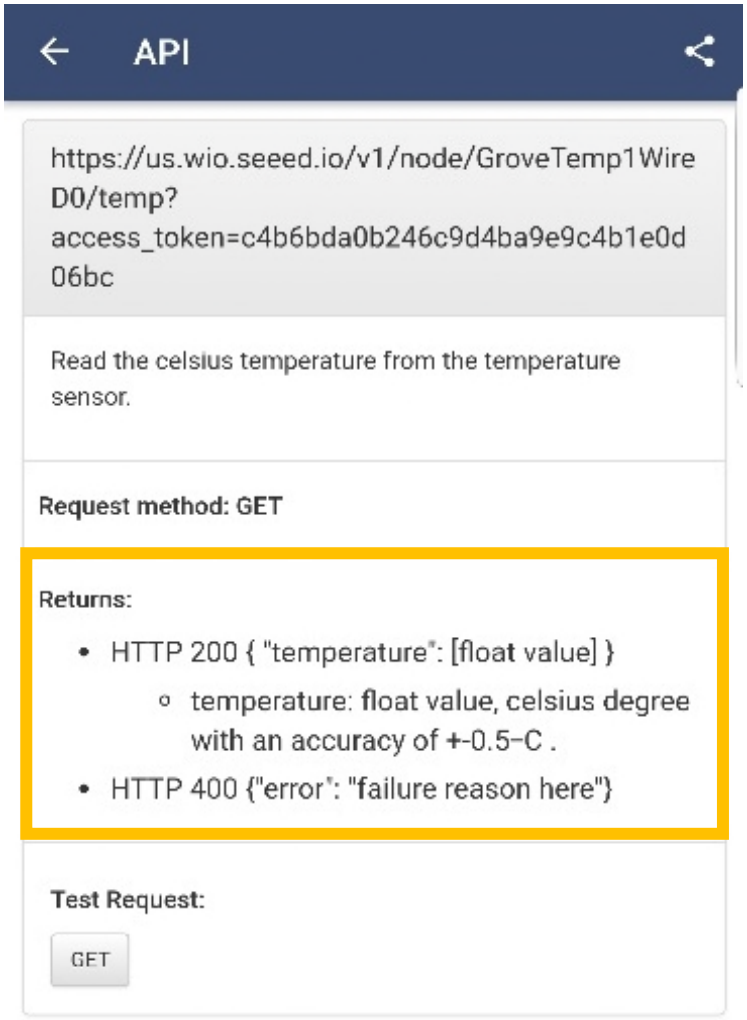
6.【応用】他のセンサをつないだときの注意

p.11 で温度を測定した際に、以下のように表示されました。
これは「JSON」と呼ばれる「名前」と「データ」の組み合わせの表示形式です。

{”temperature”:23.75}
Key value
「データの名前のこと」「データ」

第 3 章で作成するプログラムでは、この Key が重要になります。今回は温度を測定するため、”temperature”しか使いませんが、他のセンサを使う場合はそれぞれのセンサに応じた Key が表示されます。たとえば、水分センサであれば”moisture”と表示されるため、水分センサを使う際はプログラム上で moisture”と表記する必要があります。

下図の Returns:という場所を見てみましょう。
これは http://us から始まるアクセストークンにアクセスした際に表示される値です。
HTTP 200 {“temperature”:[float value]} と書いてあり、key がわかります。



「HTTP 200」は
正常にアクセスできた際の表示

[float value]は数字のことです
※プログラムでは文字と数字を
区別します

「HTTP 400」は
アクセスできなかった際の表示

第２章 スマホのメッセージアプリの設定

第2章では、スマホにデータを通知するためメッセージアプリを連携させるための準備をしていきます。今回はメッセージアプリ「LINE」を例に、機能の１つである「LINE Notify」の設定を行っていきます。

１.【スマホで】事前準備

通知に使用したいLINEのアカウント（メールアドレス）とパスワードを確認しておきます。
※ スマートフォンの LINE アプリ上で PC でのログイン許可を行いましょう

2.【スマホで】通知先の設定

通知したいスマホのLINEアプリでグループを作っておきます。

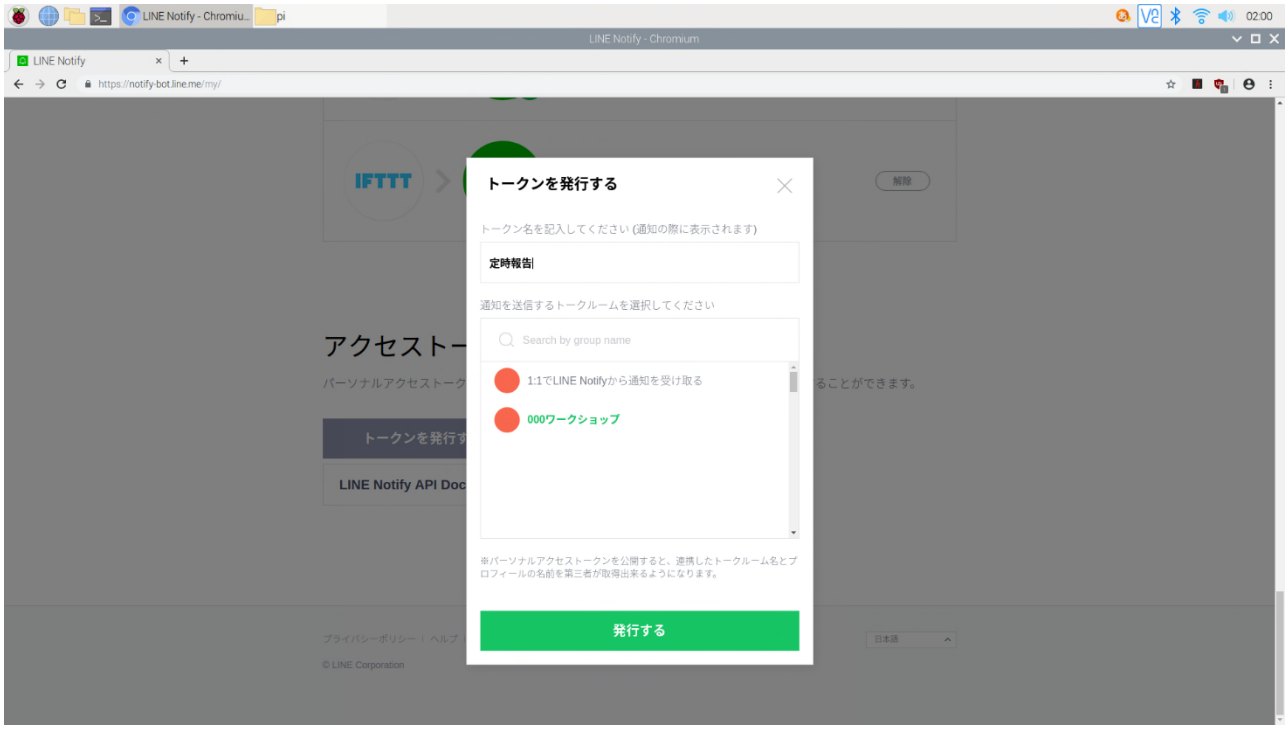
3.【Raspberry Pi で】LINE Notify にアクセスして、トークンを発行する

イ. Raspberry Pi のブラウザを開き、https://notify-bot.line.me/ja/ にアクセスし、メールアドレスとパスワードでログインします。
※ログインできない場合は、スマートフォンでブラウザアプリを開き、「PC 版サイト」を閲覧する設定にして同じ操作を行ってみてください。
（注意！）複数回ログインに失敗するとしばらくログインできなくなるようです。LINE アプリの設定については本マニュアルでは説明いたしませんので、ご了承ください。

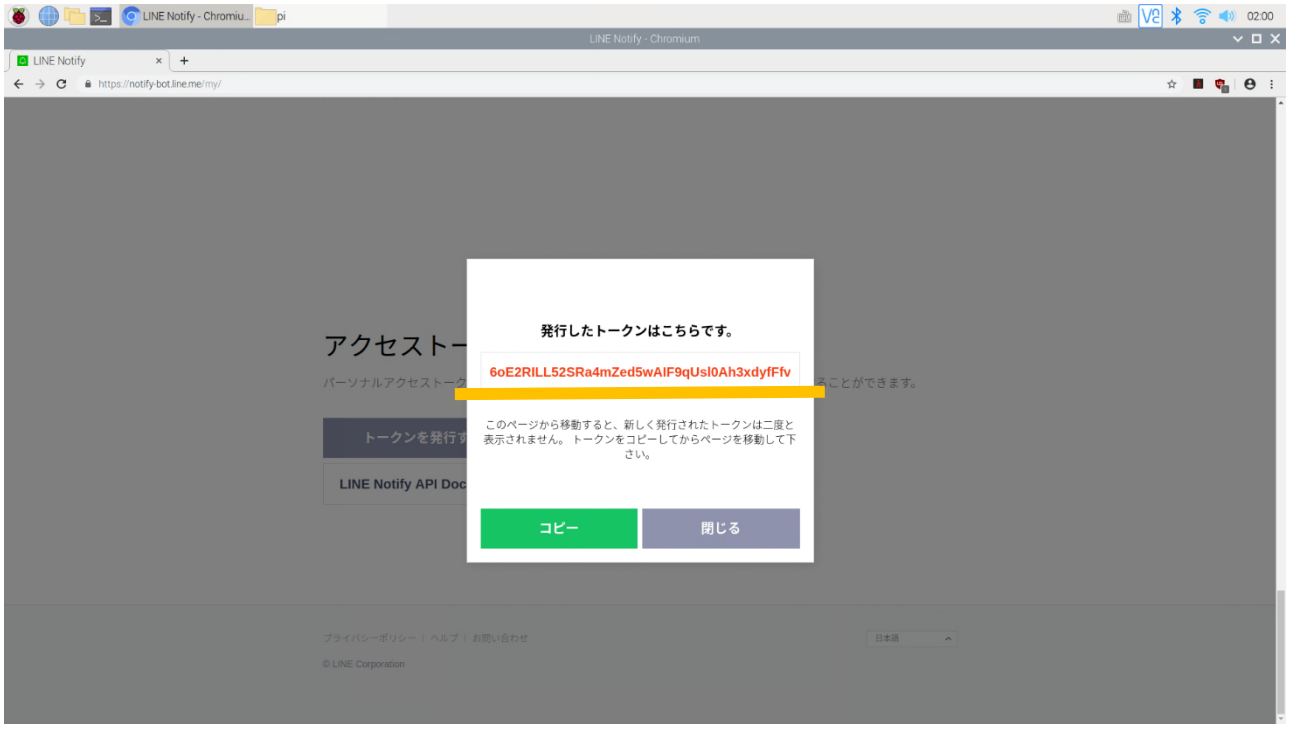
ロ. 下の図のような Line Notify のページが開きます。右上をクリックして、
マイページにアクセスします

ハ. 下の方にアクセストークン発行とあるので、「トークンを発行する」をクリックします

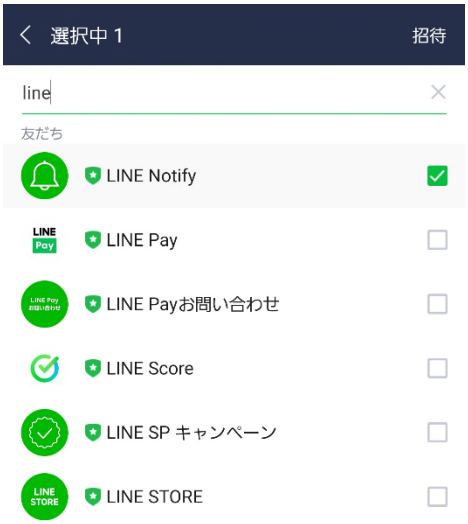
ニ. 通知したいグループを選び、「トークン名」には「定時報告」と書き込み、
「発行する」を押しましょう
※トークン名はデータを通知する際に表示されます。



ホ. 【重要】アクセストークンが発行されます。
コピーボタンを押し、wio node のアクセストークンと同様にしっかり保存しましょう。
1度しか表示されないので、必ず「テキストファイル」などに一時保存しましょう



4.【スマホで】通知したい LINE グループに LINE Notify を招待する



LINE グループのトークから招待する必要があります。招待が終わればこれで本章は終了です。

なお、Line Notify のアクセストークンを再発行したい場合、本章の内容を初めからもう一度行えば再発行可能です。

（参考）スマートフォンで Line notify のトークンを設定する方法

「3.【Raspberry Pi で】LINE Notify にアクセスして、トークンを発行する」と同じ操作をスマートフォンで行います。スマートフォンのブラウザ（Chrome など）の設定画面で、「PC 版サイト」を表示する項目をオンにすると、スマートフォン向けサイトでは設定することができないアクセストークンの発行が可能です。ただし、「PC 版サイト」を表示する設定は、ページを移動すると「オフ」になってしまうので注意してください。

第３章 小型パソコン（Raspberry Pi）でのプログラム

第3章では、第１章で設定した IoT 温度計と、第2章で設定したメッセージアプリを連携させるためのプログラムを作っていきます。

プログラムは小型パソコン（Raspberry Pi）上で作成し定期実行させます。Raspberry Pi は、初めから python というプログラミング言語が使用できる環境がインストールされており、プログラムの自動実行に必要な第4章で使用する crontab が使用できます。

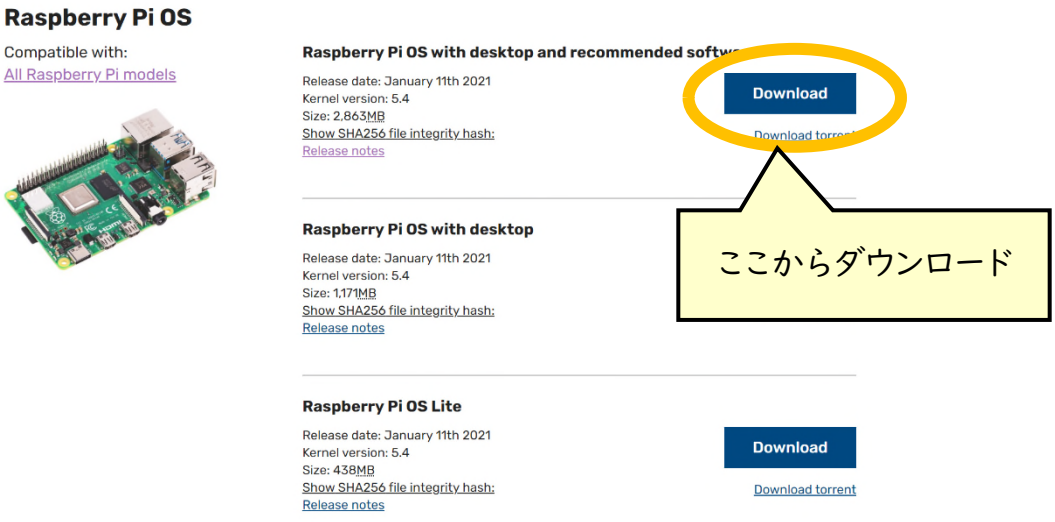
まずは簡単に Raspberry Pi の設定から説明していき、その後それぞれのプログラムを動かしていきます。Raspberry Pi の詳しい OS インストール方法などについては市販の入門書等を参照してください。また、既に OS が書き込み済みの microSD カードが付属するものを購入した場合はこの操作は必要ありません。今回は何も書き込まれていない未使用の microSD を使用した場合の手順を説明します。なお、本手順はマニュアル作成時のものであることをご了承ください。

１. Raspberry Pi の設定（windows パソコンでの操作）

Raspberry Pi Imager と呼ばれる公式ソフトウェアを利用した方法を説明します。

イ. OS ファイルをダウンロードする

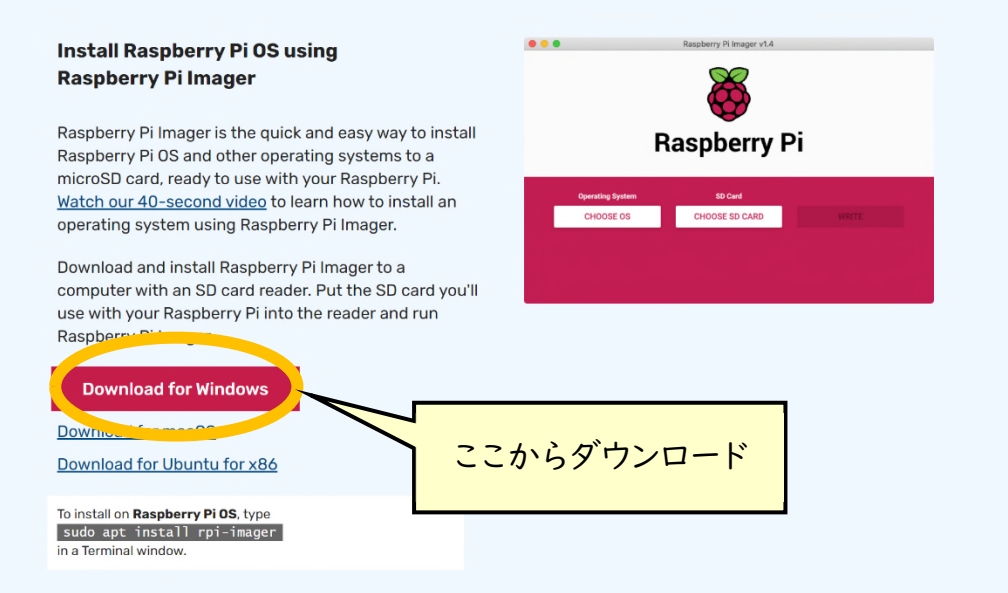
ラズベリーパイ財団のホームページのダウンロードページ（以下 URL）にアクセスする
https://www.raspberrypi.org/software/operating-systems/
「Raspberry Pi OS with desktop and recommended software」をダウンロードするために「Download」をクリックします。
ファイルが大きいため、時間がかかります。注意してください。



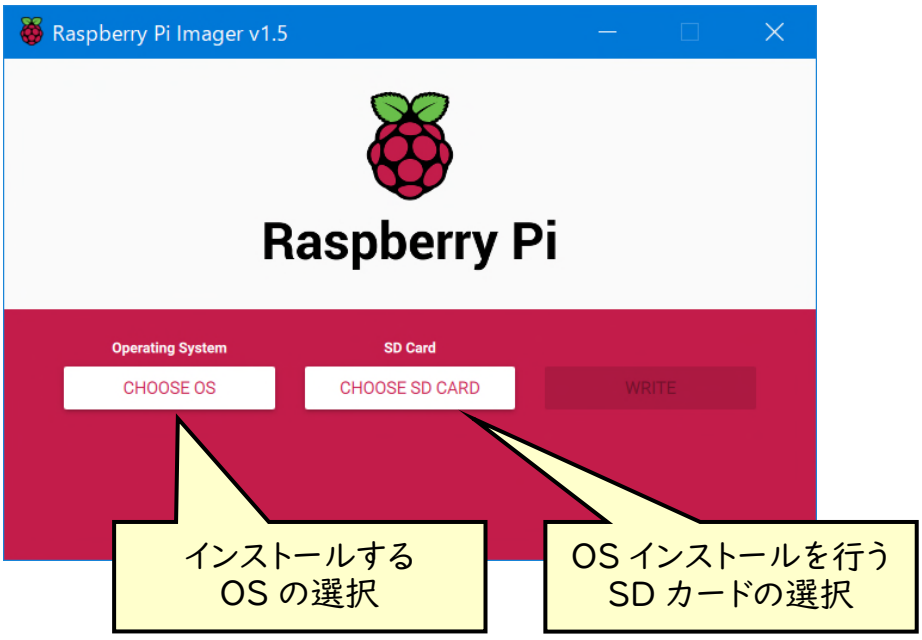
引用：https://www.raspberrypi.org/software/operating-systems/ より

ロ. OS ファイルをダウンロードしたら、zip ファイルのままで解凍する必要はありません。

ハ. 次に OS をインストールするための公式ソフトウェア「Raspberry Pi Imager」をダウンロードします。
ラズベリーパイ財団のホームページのダウンロードページ（以下 URL）にアクセスします。https://www.raspberrypi.org/software/
Windows 版を例に説明します。「Download for windows」をクリックしてダウンロードを開始してください。「imager_1.5.exe」というファイルがダウンロードされます。
今回はバージョン 1.5 を用いた場合の説明を行います。



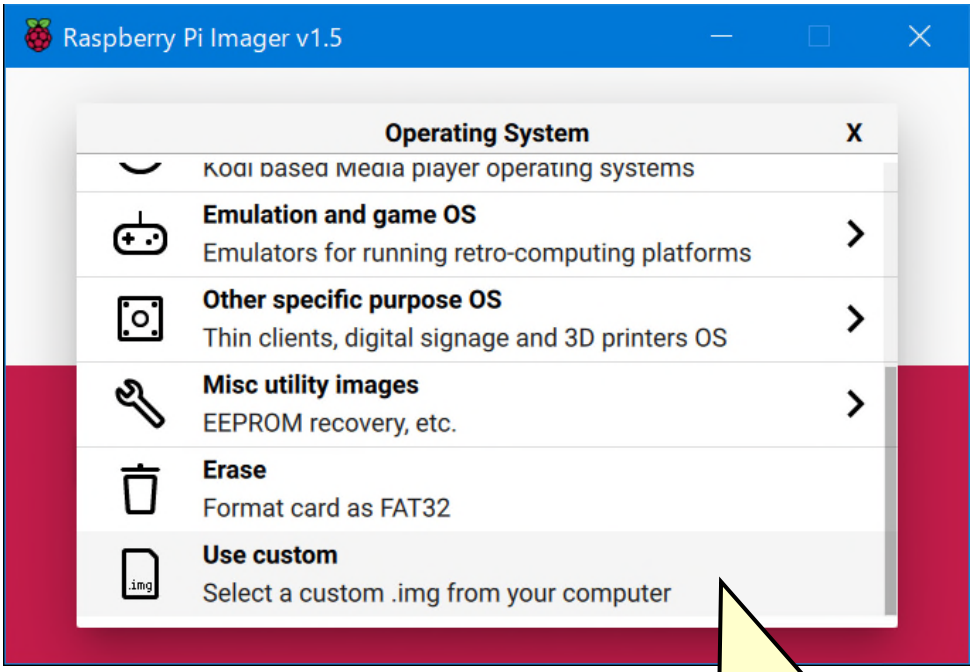
ニ.「imager_1.5.exe」から「Raspberry Pi Imager」をインストールします。インストール後、Raspberry Pi Imager を実行します。書き込みする microSD カードの選択とインストールする OS を選択してから OS のインストールを行います。

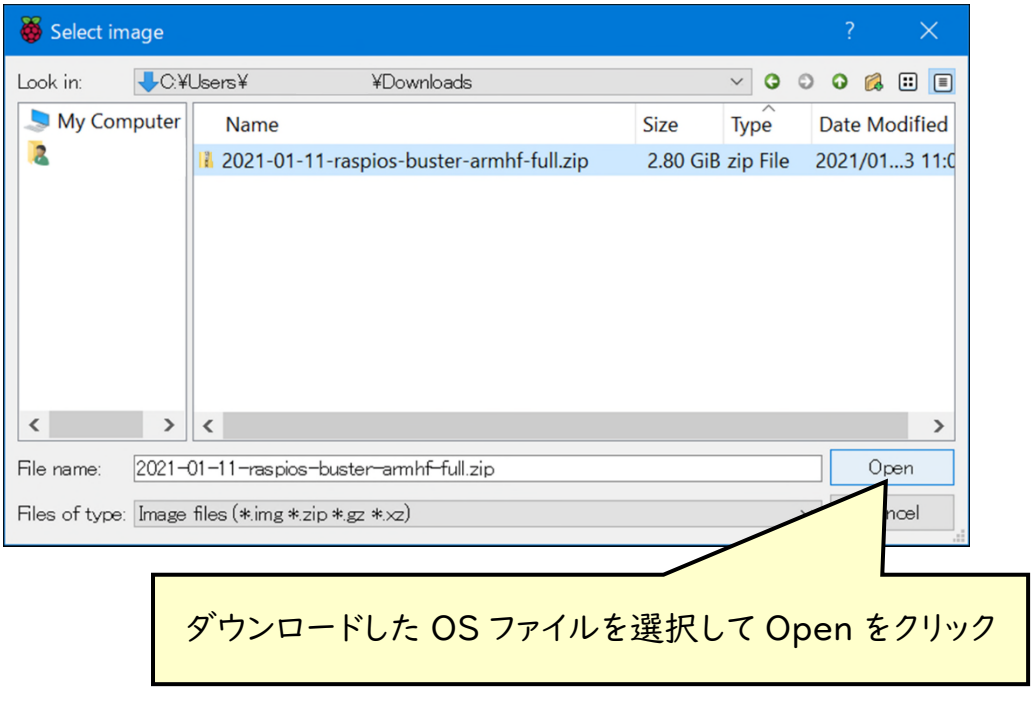


ホ. インストールする OS の選択を行います。

「Use custom」を選択し、ダウンロードした OS ファイル「20xx-xx-xx-raspios-buster-armhf-full.zip」を選択します。

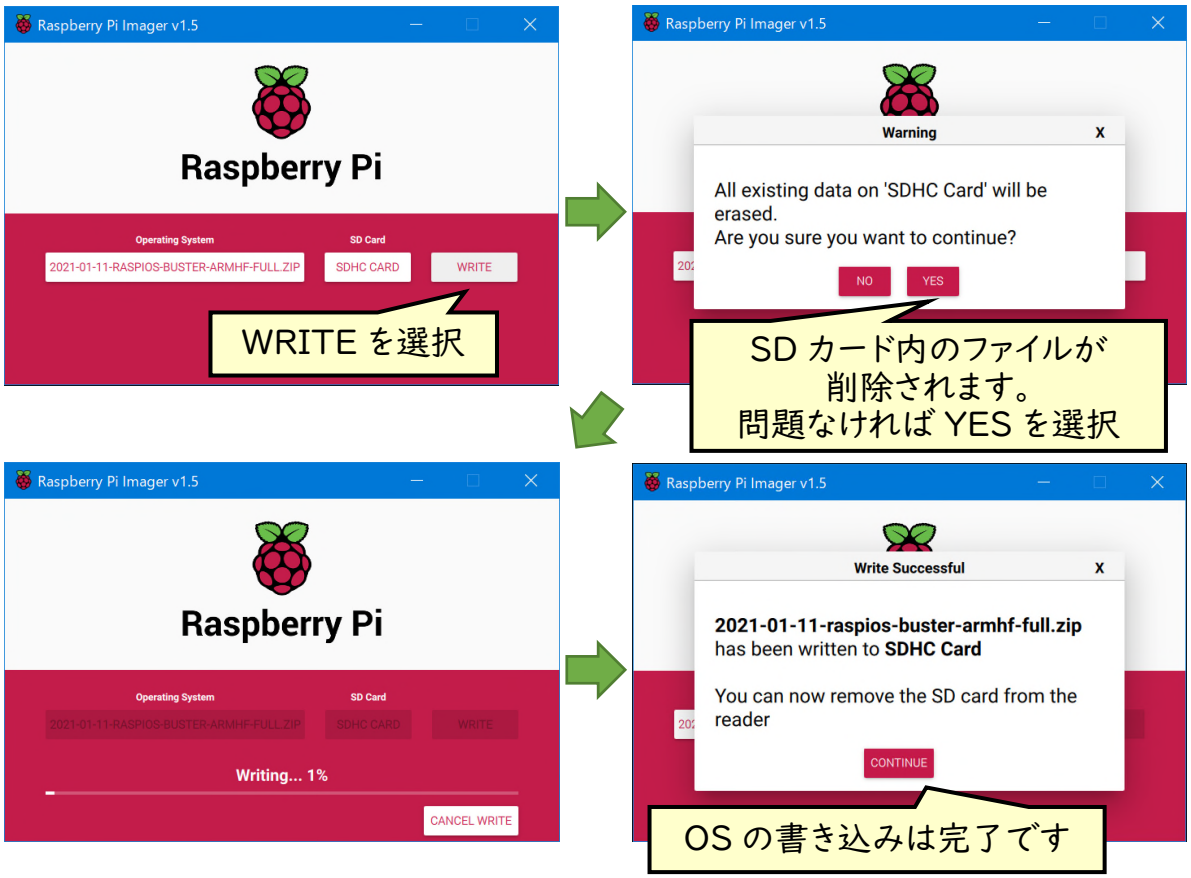
なお、「Raspberry Pi OS(other)」⇒「Raspberry Pi OS Full(32bit)」を選択することでもインストールが可能ですが、OS ファイルをダウンロードしながらインストールすることとなるため、長い時間がかかるほか、通信環境等の問題で失敗した場合は途中から再実行できないため、今回は OS ファイルを事前にダウンロードする方法で行っています。





へ. インストールする SD カードの選択を行います。
　事前に使用する microSD カードを使用できるようマイクロ SD カードリーダーに接続しておきます。

ト. OS ファイルと SD カードの選択が終わりましたので、WRITE を選択し、OS ファイルの書き込みを開始します。

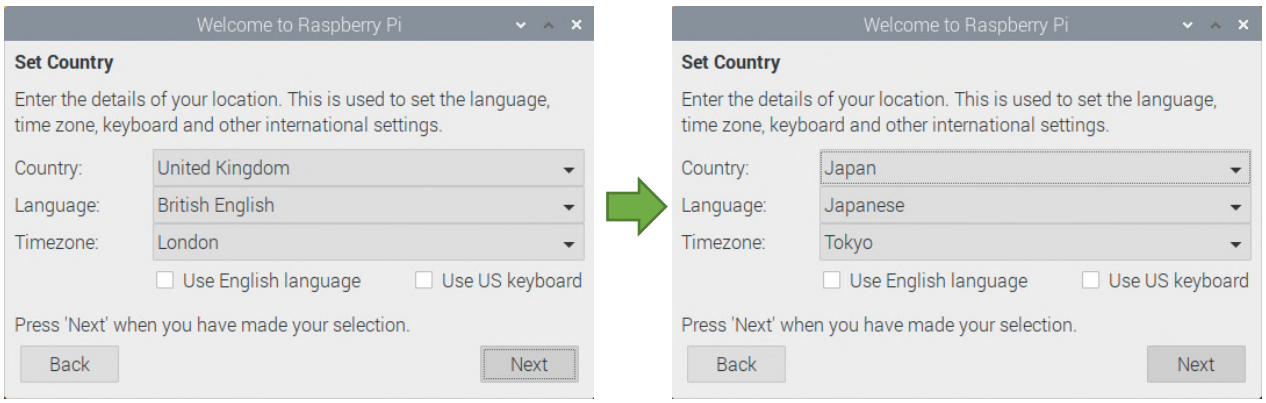


2. Raspberry Pi の設定（Raspberry Pi での操作）

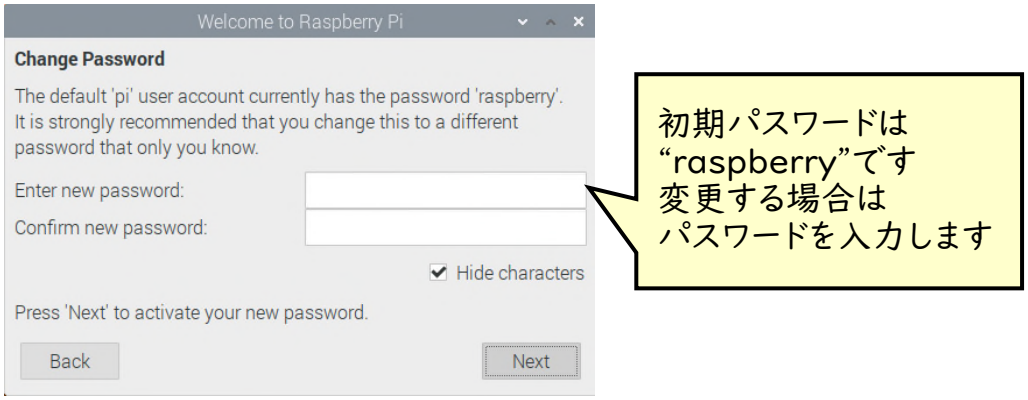
OS をインストールした microSD を Raspberry Pi にセットし、電源用の USB ケーブルをつなぎ Raspberry Pi に電源を入れます。Raspberry Pi には液晶モニタ、キーボードおよびマウスが接続された状態にしてください。

イ．初期設定画面が開かれていますので、以下の画面の Next をクリックして、画面に表示される指示に従って初期設定を進めていきます。

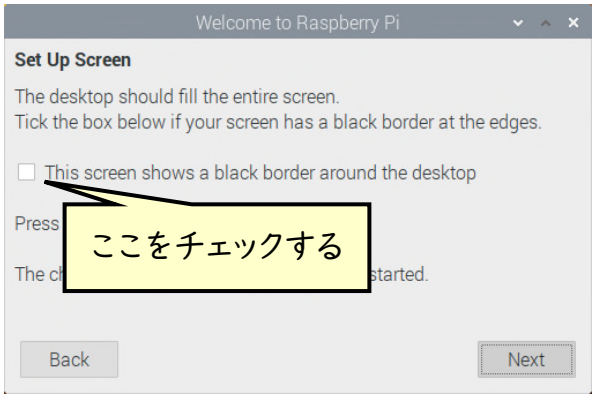
ロ．日本語での表示を行うために、「Country」を「Japan」に変更します。



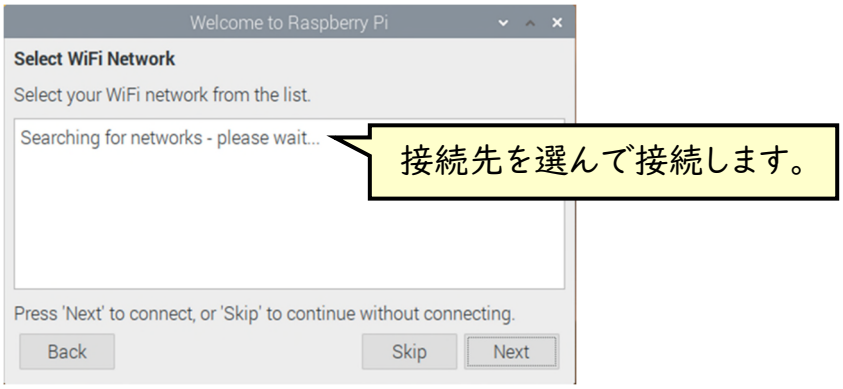
ハ．必要に応じてパスワードを変更します。



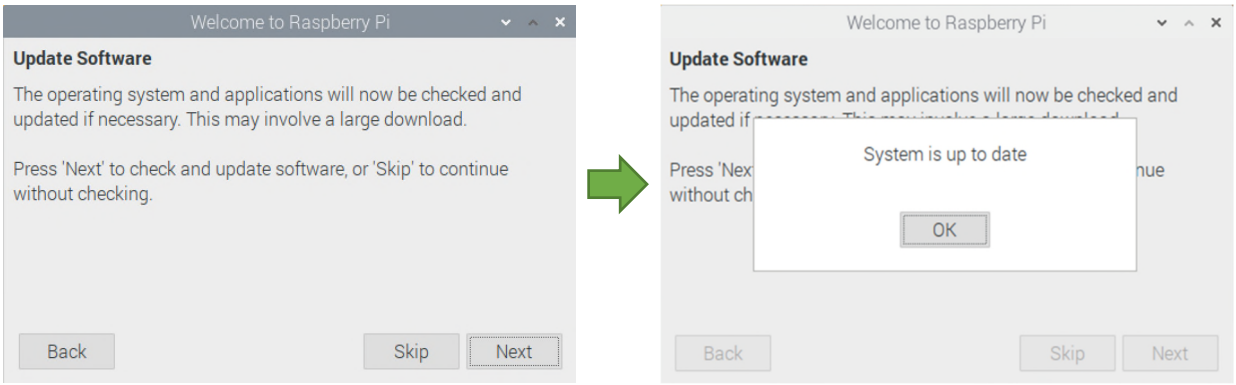
ニ．表示画面に黒い枠がある場合はチェックボックスを入れて進めてください。



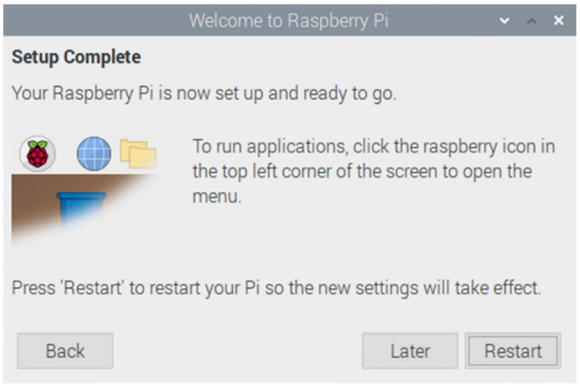
ホ．Wi-Fi に接続します。



へ．ソフトウェアの更新を行います。ここで更新することで日本語入力ができるようになるので、必ず実行してください。アップデートに成功すると右の画面になります。



ト．“Restart”をクリックして再起動すると日本語表示、入力ができるようになります。

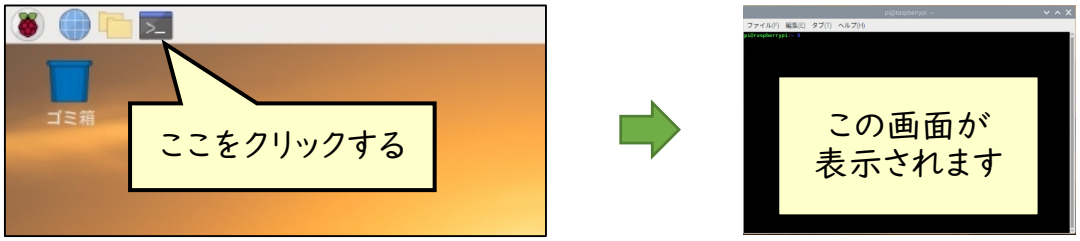


再度初期設定を行いたい場合、
また日本語入力できない場合は、
初期設定画面から再度ソフトウェア
の更新を行う必要があります。
LXterminal(ターミナル)を使用して、
sudo piwiz
と入力して実行してください。
※LXterminal の説明は次ページ

3. グラフ通知プログラムを使用する際の追加設定

ここまでの設定で本マニュアル上のプログラムはほぼ使用できますが、グラフ通知プログラムを利用する際は以下の設定が必要です。必要に応じて設定を行ってください。

イ． LXterminal（ターミナル）を使用して、グラフ通知プログラムに必要なモジュール*（プログラミング言語 Python の機能を拡張するライブラリ）を 3 つインストールします。「NumPy」「pandas」「matplotlib」と呼ばれるものです。以下の手順に従ってください。



ロ. ターミナルに次のコマンド（下線部）を打ち込んで Enter を押してください

pi@raspberrypi: ~ $ sudo pip3 install numpy

```
pi@raspberrypi:~ $ sudo pip3 install numpy
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: numpy in /usr/lib/python3/dist-packages (1.16.2)
```

本マニュアルに沿って設定していると既にインストールされているとの表示が出ます。NumPyの version は 1.20.1 の利用を想定していますが、pandas をインストールする際にアップデートされます。

ハ. ターミナルに次のコマンド（下線部）を打ち込んで Enter を押してください

pi@raspberrypi: ~ $ sudo pip3 install pandas

```
pi@raspberrypi:~ $ sudo pip3 install pandas
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting pandas
  Downloading https://www.piwheels.org/simple/pandas/pandas-1.2.2-cp37-cp37m-linux_armv7l.whl (32.3MB)
    100% |████████████████████████████████| 32.3MB 14kB/s
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.7/dist-packages (from pandas) (1.20.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/lib/python3/dist-packages (from pandas) (2.7.3)
Collecting pytz>=2017.3 (from pandas)
  Downloading https://files.pythonhosted.org/packages/70/94/784178ca5dd892a98f113cdd923372024dc04b8d40abe77ca76b5fb9
    100% |████████████████████████████████| 512kB 333kB/s
Installing collected packages: pytz, pandas
Successfully installed pandas-1.2.2 pytz-2021.1
```

インストールに成功すると上記の表示になります。

* Python 3.7.3、Numpy 1.20.1、Pandas 1.2.2、 Maplotlib 3.3.4 で動作を確認済

ニ. ターミナルに次のコマンド（下線部）を打ち込んで Enter を押してください

pi@raspberrypi: ~ $ sudo pip3 install matplotlib

```
pi@raspberrypi:~ $ sudo pip3 install matplotlib
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: matplotlib in /usr/lib/python3/dist-packages (3.0.2)
```

既にインストールされているとの表記が出ますが、matplotlib の version が 3.0.2 となっています。本マニュアルでは 3.3.2 以上が必要ですのでアップデートを行います。version が古いままだとグラフ通知時に日本語での表示ができません。

pi@raspberrypi: ~ $ sudo pip3 install -U matplotlib

```
pi@raspberrypi:~ $ sudo pip3 install -U matplotlib
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting matplotlib
  Downloading https://www.piwheels.org/simple/matplotlib/matplotlib-3.3.4-cp37-cp37m-linux_armv7l.whl (11.5MB)
    100% |████████████████████████████████| 11.5MB 38kB/s
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in /usr/lib/python3/dist-packages (from matplotl
Requirement already satisfied, skipping upgrade: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages (from matplo
Requirement already satisfied, skipping upgrade: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /usr/lib/python3/dist-
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in /usr/lib/python3/dist-packages (from matpl
Collecting pillow>=6.2.0 (from matplotlib)
  Downloading https://www.piwheels.org/simple/pillow/Pillow-8.1.0-cp37-cp37m-linux_armv7l.whl (1.3MB)
    100% |████████████████████████████████| 1.3MB 162kB/s
Requirement already satisfied, skipping upgrade: cycler>=0.10 in /usr/lib/python3/dist-packages (from matplotlib) (
Installing collected packages: pillow, matplotlib
  Found existing installation: Pillow 5.4.1
    Not uninstalling pillow at /usr/lib/python3/dist-packages, outside environment /usr
    Can't uninstall 'Pillow'. No files were found to uninstall.
  Found existing installation: matplotlib 3.0.2
    Not uninstalling matplotlib at /usr/lib/python3/dist-packages, outside environment /usr
    Can't uninstall 'matplotlib'. No files were found to uninstall.
Successfully installed matplotlib-3.3.4 pillow-8.1.0
```

アップデートに成功して、3.3.4 となったという表示が出ています。

ホ. このままですと numpy がインストールされているのに使用できない（インポートできない）状態になっています（2021 年 3 月現在）。そのため、以下のコマンドを実行し、使用できるようにします。

pi@raspberrypi: ~ $ sudo apt-get install libatlas-base-dev

```
pi@raspberrypi:~ $ sudo apt-get install libatlas-base-dev
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
以下の追加パッケージがインストールされます:
  libatlas3-base
提案パッケージ:
  libatlas-doc liblapack-doc
以下のパッケージが新たにインストールされます:
  libatlas-base-dev libatlas3-base
アップグレード: 0 個、新規インストール: 2 個、削除: 0 個、保留: 0 個。
5,365 kB のアーカイブを取得する必要があります。
この操作後に追加で 32.1 MB のディスク容量が消費されます。
続行しますか? [Y/n] y
取得:1 http://ftp.tsukuba.wide.ad.jp/Linux/raspbian/raspbian buster/main armhf libatlas3-base armhf 3.10.3-8+rpi1 [
取得:2 http://ftp.tsukuba.wide.ad.jp/Linux/raspbian/raspbian buster/main armhf libatlas-base-dev armhf 3.10.3-8+rpi
5,365 kB を 4秒 で取得しました (1,214 kB/s)
以前に未選択のパッケージ libatlas3-base:armhf を選択しています。
(データベースを読み込んでいます ... 現在 167042 個のファイルとディレクトリがインストールされています。)
.../libatlas3-base_3.10.3-8+rpi1_armhf.deb を展開する準備をしています ...
libatlas3-base:armhf (3.10.3-8+rpi1) を展開しています...
以前に未選択のパッケージ libatlas-base-dev:armhf を選択しています。
.../libatlas-base-dev_3.10.3-8+rpi1_armhf.deb を展開する準備をしています ...
libatlas-base-dev:armhf (3.10.3-8+rpi1) を展開しています...
libatlas3-base:armhf (3.10.3-8+rpi1) を設定しています ...
update-alternatives: /usr/lib/arm-linux-gnueabihf/libblas.so.3 (libblas.so.3-arm-linux-gnueabihf) を提供するために
```

へ. ここまででラズベリーパイの設定は終了ですが、必要に応じて OS のアップデートを以下の操作を参考に行ってください。

1.アップデートがあるか確認する。ターミナルで以下のコマンドを打つ。

sudo apt-get update

2. アップデートをインストールする。Lx Terminal で以下のコマンドを打つ。

sudo apt-get upgrade

場合によっては以下を実行

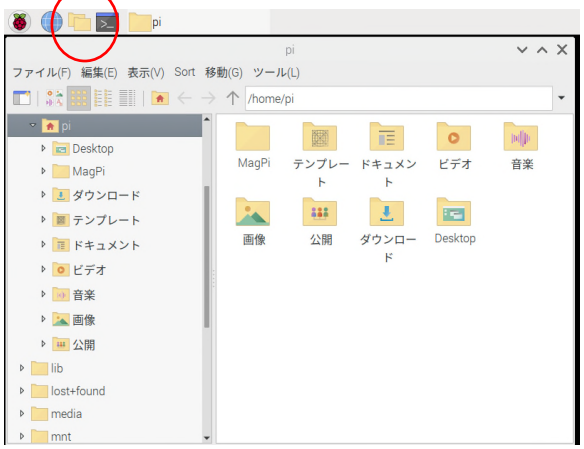
sudo apt autoremove

4. プログラム言語 Python を使う準備

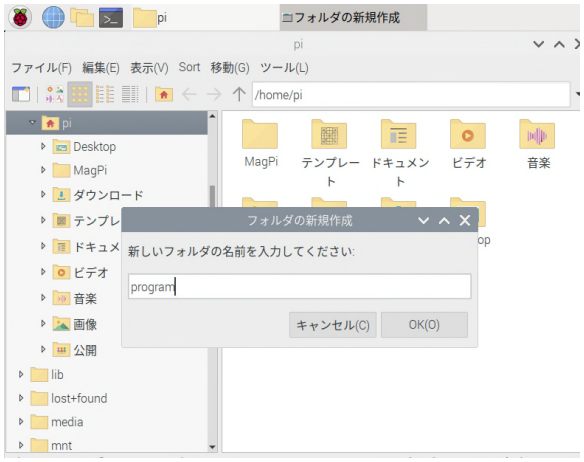
イ. プログラムを入れるフォルダを作りましょう

Raspberry Pi （の OS である「Raspberry Pi OS」）では、通常使うフォルダは、home フォルダ内の pi フォルダの中に作られます。つまり、 /home/pi/ の中に作られます。

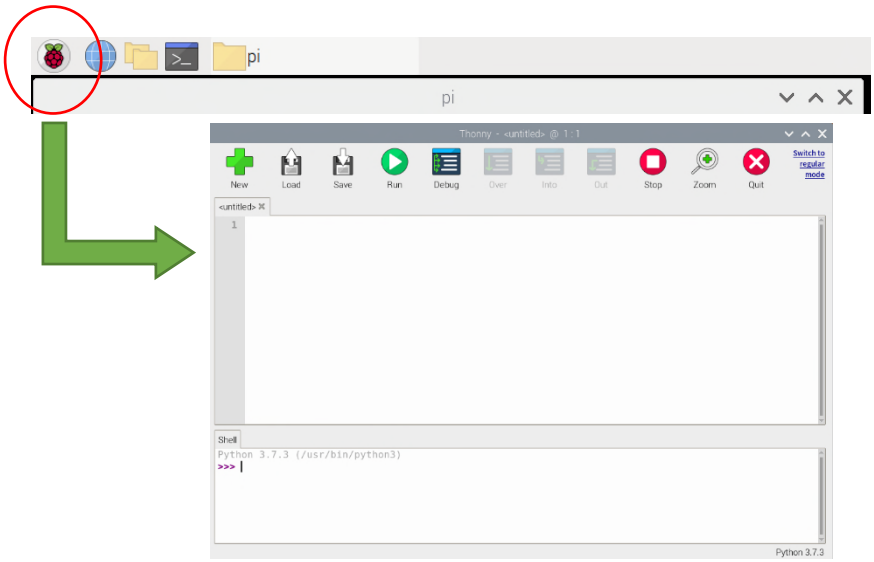
以下の画面のフォルダをクリックすると pi フォルダが表示されます。



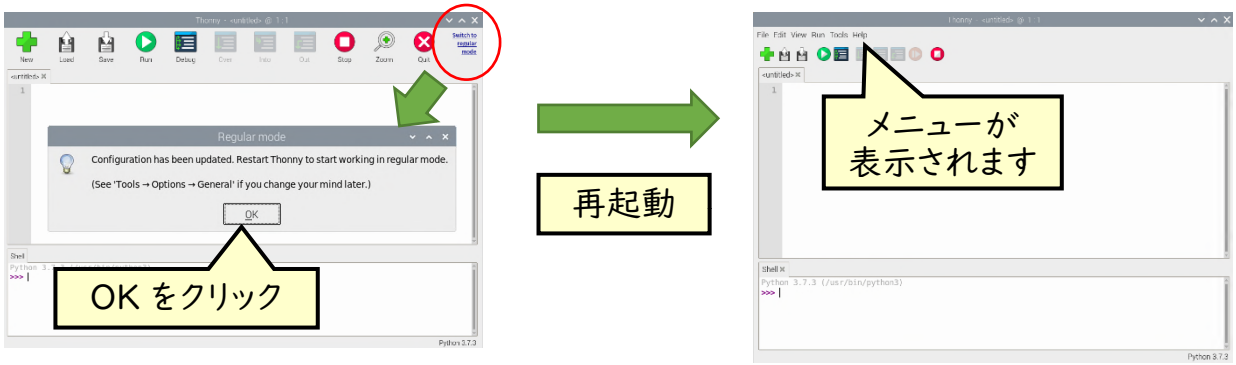
このフォルダの中にプログラムを入れる「program」フォルダを作りましょう。
右クリックの新規作成（または「New Folder」を選択）から作ることができます。



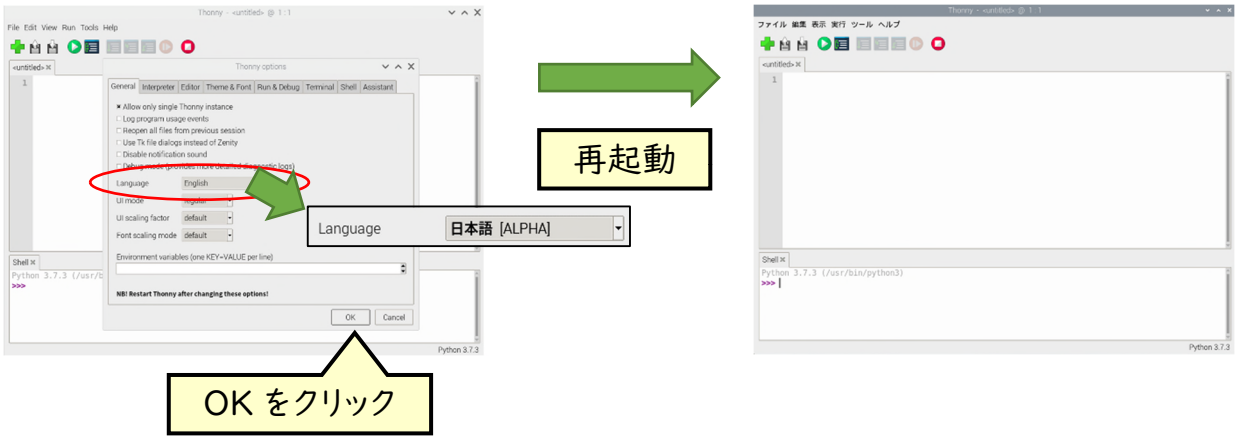
ロ. Python を書くためのエディタ（ソフト）を立ち上げてみましょう左上のラズベリー（図の赤い丸）をクリックし、プログラム⇒Thonny Python IDE をクリックすると下の画面が出てきます。このままでも使用できますが、日本語での表示ができるように設定していきましょう。



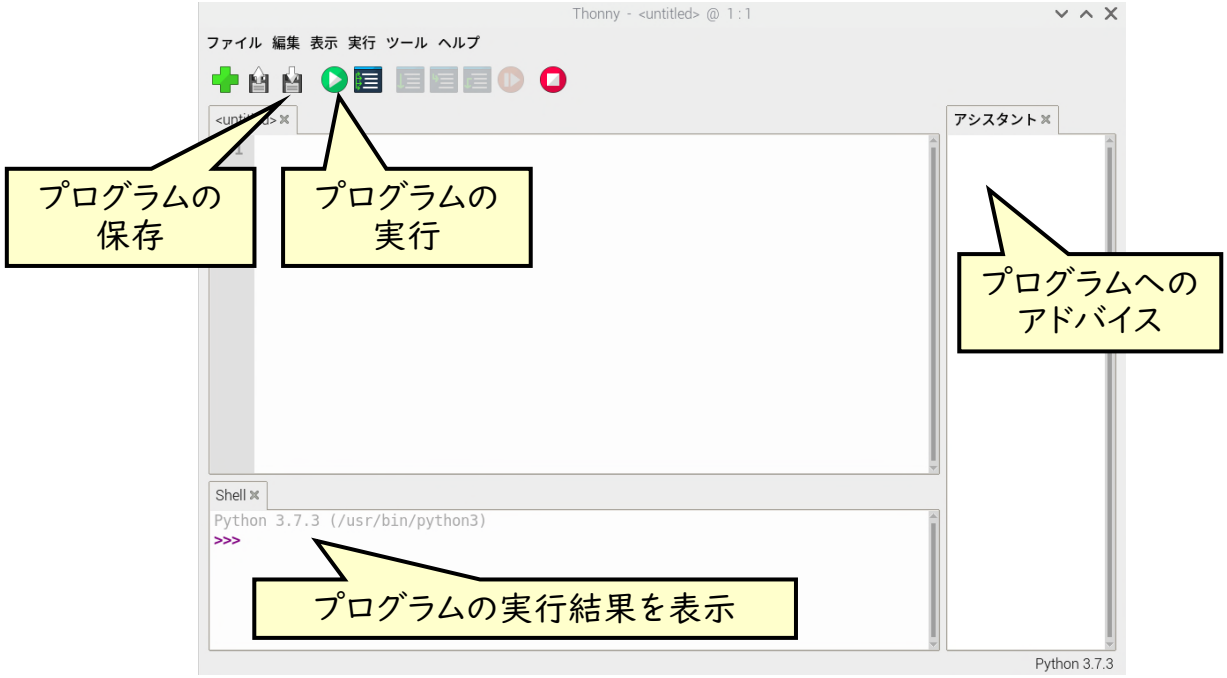
ハ. まず、右上にある「Switch to regular mode」をクリックした後、OK を選択後、Thonny（トニー）を終了し、もう一度起動してください。起動すると右の画面になります。



ニ.「Tool」から「Options」を選択すると、左図の Option 画面が出ますので、「Language」を「日本語」に変更し、OK を選択した後、Thonny を再起動してください。



ホ. 無事日本語設定が出ましたが、プログラムを作成し実行した際に、間違っている箇所などを教えてくれる「アシスタント」を表示しておきます。「表示」から「アシスタント」をクリックすると、以下の画面になります。これで Python のプログラムを書く設定は完了です。

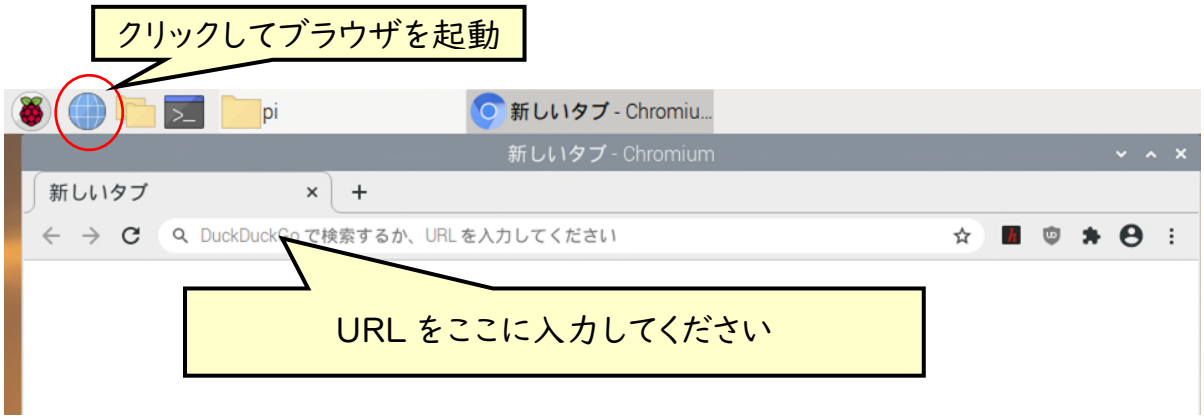


5. 配布プログラムのダウンロード

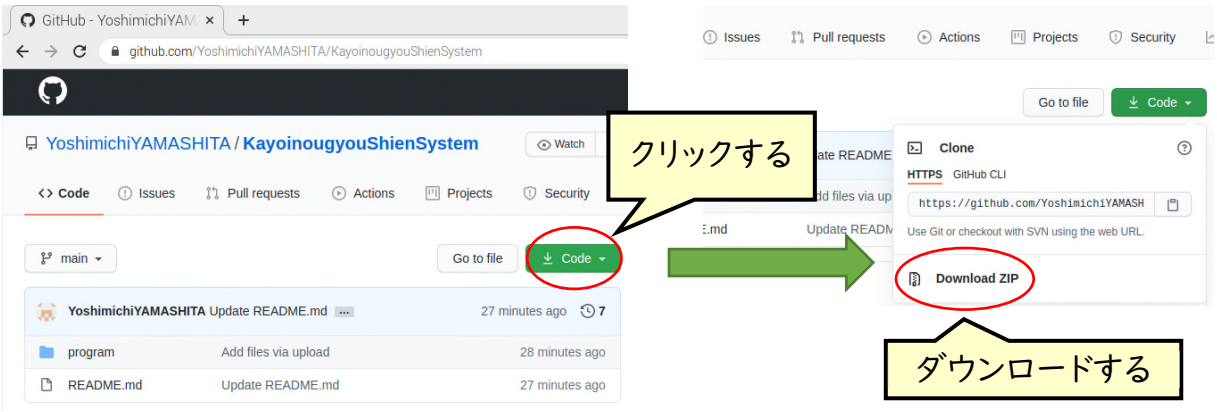
マニュアル内のプログラムを Thonny で書いて使用することもできますが、通い農業支援システムでは配布プログラムをダウンロードして使用できます。
以下の手順に沿って、プログラムをダウンロードしましょう。

イ. ブラウザ（Chromium）を開いて、以下の URL にアクセスします。

https://github.com/YoshimichiYAMASHITA/KayoinougyouShienSystem



ロ. GitHub と呼ばれるサイトからプログラムをダウンロードできます。「↓ Code」をクリックして、「Download ZIP」を選択してダウンロードしてください。



ハ. pi フォルダ中の Downloads フォルダにプログラムがダウンロードされます。
ダウンロードしたファイル「KayoinougyoShienSystem-main.zip」を右クリックして、「ここでファイルを展開」を選択すると圧縮されたファイルが解凍されます。

ニ. 解凍したフォルダには以下のファイルが入っていることを確認してください。
　これら以外もいくつか資料が付属していますので、適宜確認し利用してください。
【 KayoinougyoShienSystem-main フォルダ】

・program（フォルダ）

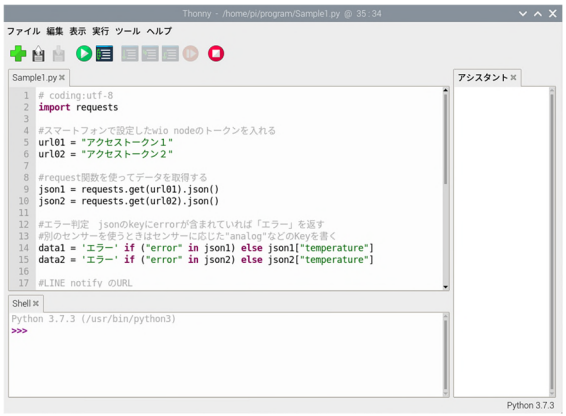
program1.py	…	プログラム①「定期通知プログラム」
program2.py	…	プログラム②「警報通知プログラム」
program3.py	…	プログラム③「定期通知プログラム」（再実行あり）
program4.py	…	プログラム④「データ保存プログラム」

　これらの4つのプログラムは「通い農業支援システム」を構成する機能の一部です。このため、本システムを利用するためのチュートリアルを目的としています。しかし、これらのプログラム単体で運用することも可能です。時間が無く、通い農業支援システムをいち早く作成したい場合は、少なくともプログラム①だけを作成して利用方法を学んで下さい。

Kayoi_data10.py	…	「データ通知プログラム」 10 分おきにデータを（通知）・保存する
Kayoi_daily_report.py	…	「グラフ通知プログラム」 平均値、最大・最小値、グラフを通知する

　「通い農業支援システム」はこれらの2つのプログラム「データ通知プログラム」と「グラフ通知プログラム」で動かすことを前提としています。データ通知プログラムで 10 分ごとにデータを取得し、通知と保存を行います。グラフ通知プログラムはデータ通知プログラムで保存したデータから、前日の平均値・最大値・最小値とグラフを通知します。なお、この2つのプログラムは第6章で説明します。

ホ. これらのプログラムを「4. プログラム言語 Python を使う準備」で作成した「program」フォルダに移動させてください。その後、program1.py をダブルクリックして開くと、以下のような画面が開きます。次ページの説明を読んで、このプログラムを利用してみましょう。



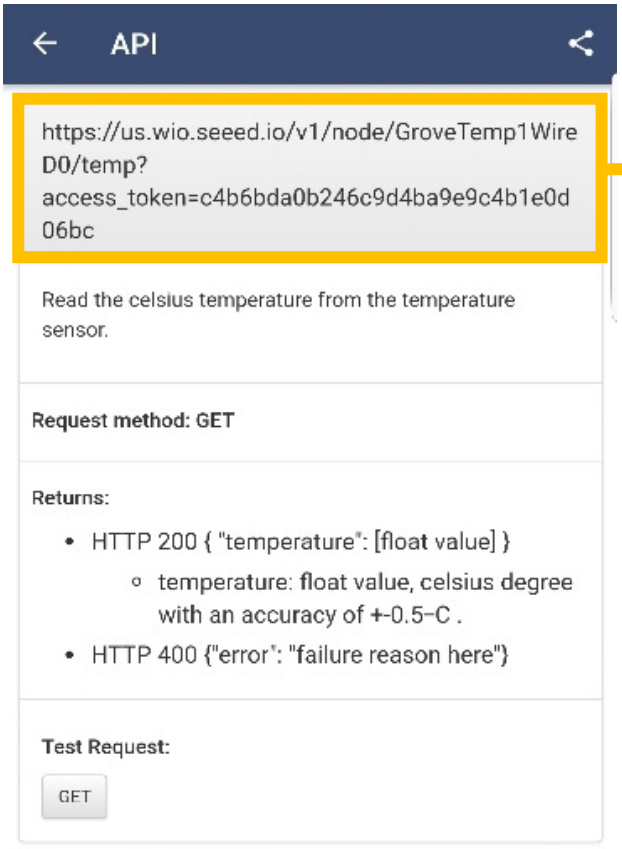
6. プログラム①:「定期通知プログラム」温度を決まった時刻に通知

program1.py を開いて、プログラムを作成します。第1章、第2章で設定したアクセストークンを利用します。以下のプログラムの灰色の網掛け部分を変更します。

5~6行目はアクセストークンを変更
21行目は通知するLINEグループのアクセストークンに変更
24 行目、27~28 行目のメッセージ、センサ名、単位は必要があれば変更

```
1  # coding:utf-8
2  import requests
3
4  #スマートフォンで設定した wio node のトークンを入れる
5  url01 = "アクセストークン1"
6  url02 = "アクセストークン2"
7
8  #request 関数を使ってデータを取得する
9  json1 = requests.get(url01).json()
10 json2 = requests.get(url02).json()
11
12 #エラー判定 json の key に error が含まれていれば「エラー」を返す
13 #別のセンサを使うときはセンサに応じた"analog"などの Key を書く
14 data1 = 'エラー' if ("error" in json1) else json1["temperature"]
15 data2 = 'エラー' if ("error" in json2) else json2["temperature"]
16
17 #LINE notify の URL
18 #LINE notify のトークン(通知先に応じて変更すること)
19
20 url99 = "https://notify-api.line.me/api/notify"
21 token = '通知する LINE グループのアクセストークン'
22
23 #LINE に通知するメッセージを記入 ''は文字列のこと
24 message  = 'ハウスの情報\n '
25 #message += を使うと通知メッセージを増やせる
26 #センサ名、データ番号、単位を書く
27 message += '温度1:'+str(data1)+'℃'+'\n'
28 message += '温度2:'+str(data2)+'℃'
29
30 #LINE に通知するための3行
31 payload = {'message' : message}
32 headers = {'Authorization' : 'Bearer '+ token,}
33 r = requests.post(url99,data=payload,headers=headers)
34
```

イ. 5~6 行目のアクセストークンは第 1 章の p.15 で説明しました。



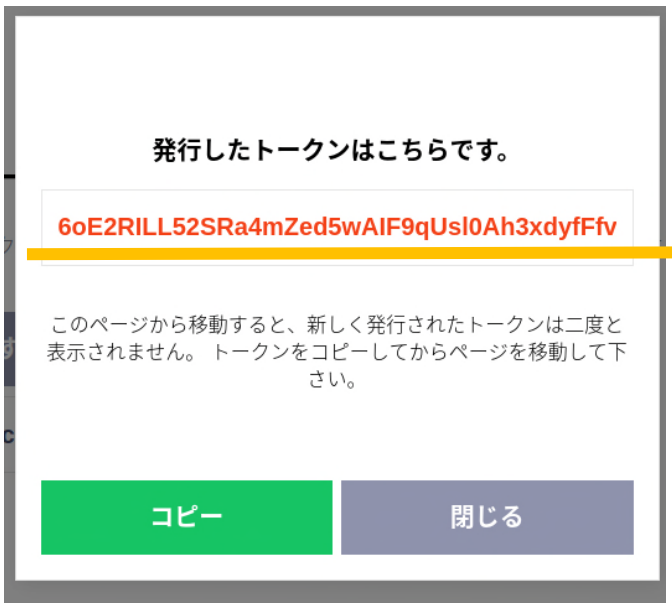
Wio Node のアプリを立ち上げ、
View API で出てくるこの URL が
「アクセストークン」になります。

View API のこの画面には
p.6 で説明した通り、PC やスマホの
ブラウザ上でアクセスできます。
Raspberry Pi のブラウザに、
View API の URL を打ち込んで、
アクセストークンを確認し、
コピーしてプログラムに貼り付けましょう。

なお、
アクセストークンをブラウザに打ち込むと、
GETボタンを押したときと同じ表示が
ブラウザ上に表示されます。

第 1 章では温度センサを 1 つしか設定しませんでした。そのため、アクセストークンは今回 1 つだけとなります。そのため、url01、url02 ともに同じアクセストークンをコピー&ペーストしてみましょう。

ロ. 21 行目のアクセストークンは第2章の p.19 で説明しました。通知したい LINE グループのアクセストークンを設定した際、画面に表示されるものを使います。

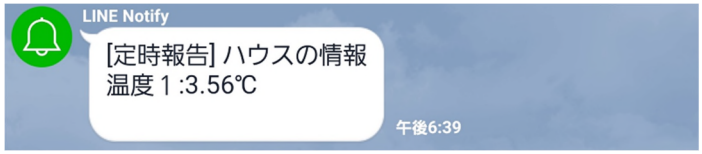


LINEグループのアクセストークン

ハ.プログラムを実行してみましょう。問題なければ設定した LINE グループに通知が来るはずです。もしも来なければアクセストークンのコピー&ペーストに失敗していないか確認して、Thonnyの右側にあるアシスタントをチェックして修正してみてください。

温度１、温度2に同じ値が通知されました。Wio Node に温度センサを2つ接続している場合や、ハウス2棟に温度センサを １ つつけた Wio Node をそれぞれに設置している場合はこのプログラムで問題ありません。しかし、今回は温度センサが１つだけですので、温度２を表示させる必要はありません。

ニ.温度２を通知しないように、温度2に関係する行を実行しないようにプログラムを修正します。実行しない行は「＃」シャープをつけることでコメント行に変えることができます。6行目、10行目、15行目、28行目の頭に「＃」をつけてからプログラムを実行してください。



温度１だけが通知されましたか？ 通知されない場合はアシスタントを確認してください。よくある間違いとして「＃」が半角ではなく、全角の「＃」シャープになっていることがあります。プログラムは半角文字で書くので、半角で書くようにしてください。

ホ.次は「24 行目、27～28 行目」の「メッセージ」、「センサ名」、「単位」の部分を変更してみましょう。’ ’や” ”で囲うことで「文字列」として表示することができます。これらを消さないように注意して名称を変えて試してみましょう。また、\n は改行を表しています。キーボードの「¥」を押すことで「\」（バックスラッシュ）を入力できます。

program1.py

```
1  # coding:utf-8
2  import requests
3
4  #スマートフォンで設定したwio nodeのトークンを入れる
5  url01 = "https://us.wio.seeed.io/v1/node/GroveTemp1WireD0/temp?access_token=[redacted]"
6  #url02 = "https://us.wio.seeed.io/v1/node/GroveTemp1WireD0/temp?access_token=[redacted]"
7
8  #request関数を使ってデータを取得する
9  json1 = requests.get(url01).json()
10 #json2 = requests.get(url02).json()
11
12 #エラー判定　jsonのkeyにerrorが含まれていれば「エラー」を返す
13 #別のセンサーを使うときはセンサーに応じた"analog"などのKeyを書く
14 data1 = 'エラー' if ("error" in json1) else json1["temperature"]
15 #data2 = 'エラー' if ("error" in json2) else json2["temperature"]
16
17 #LINE notify のURL
18 #LINE notify のトークン（通知先に応じて変更すること）
19
20 url99="https://notify-api.line.me/api/notify"
21 token='[redacted]'
22
23 #LINEに通知するメッセージを記入 ''は文字列のこと
24 message  = 'メッセージを変えました\n'
25 #message += を使うと通知メッセージを増やせる
26 #センサー名、データ番号、単位を書く
27 message += 'ハウスの温度:'+str(data1)+'℃'+'\n'
28 #message += '地温２:'+str(data2)+'℃'
29
30 #LINEに通知するための３行
31 payload = {'message' : message}
32 headers = {'Authorization' : 'Bearer '+ token,}
33 r = requests.post(url99,data=payload,headers=headers)
```

アシスタント

The code in program1.py looks good.

If it is not working as it should, then consider using some general debugging techniques.

Was it helpful or confusing?

Shell

```
>>> %Run program1.py
>>> %Run program1.py
>>>
```

へ.次は通知するセンサを増やしてみましょう。

6 行目の次の行に、
「url03 = "アクセストークン3"」を、
10 行目の次の行に、
「json3 = requests.get(url03).json()」を、
15 行目の次の行に、
「data3 = 'エラー' if ("error" in json3) else json3["temperature"]」を、
28 行目の最後に
「+'\n'」を追加し、その次の行に
「message += '温度 3:'+str(data3)+'℃'」を追加しましょう。

１つ１つ入力する必要はありません、前の行からコピー&ペーストして、必要な場所だけ修正しましょう。その後はこれまで通り、灰色網掛け部分を修正してみましょう。アクセストークン3はアクセストークン1と同じものにしてから実行すると、以下のような通知が来るはずです。

program1.py

```
# coding:utf-8
import requests

#スマートフォンで設定したwio nodeのトークンを入れる
url01 = "https://us.wio.seeed.io/v1/node/GroveTemp1WireD0/temp?access_token=[redacted]"
url02 = "https://us.wio.seeed.io/v1/node/GroveTemp1WireD0/temp?access_token=[redacted]"
url03 = "https://us.wio.seeed.io/v1/node/GroveTemp1WireD0/temp?access_token=[redacted]"

#request関数を使ってデータを取得する
json1 = requests.get(url01).json()
json2 = requests.get(url02).json()
json3 = requests.get(url03).json()

#エラー判定　jsonのkeyにerrorが含まれていれば「エラー」を返す
#別のセンサーを使うときはセンサーに応じた"analog"などのKeyを書く
data1 = 'エラー' if ("error" in json1) else json1["temperature"]
data2 = 'エラー' if ("error" in json2) else json2["temperature"]
data3 = 'エラー' if ("error" in json3) else json3["temperature"]

#LINE notify のURL
#LINE notify のトークン（通知先に応じて変更すること）

url99="https://notify-api.line.me/api/notify"
token='[redacted]'

#LINEに通知するメッセージを記入 ''は文字列のこと
message  = 'ハウスの情報\n'
#message += を使うと通知メッセージを増やせる
#センサー名、データ番号、単位を書く
message += '温度１:'+str(data1)+'℃'+'\n'
message += '温度２:'+str(data2)+'℃'+'\n'
message += '温度３:'+str(data3)+'℃'
```

アシスタント

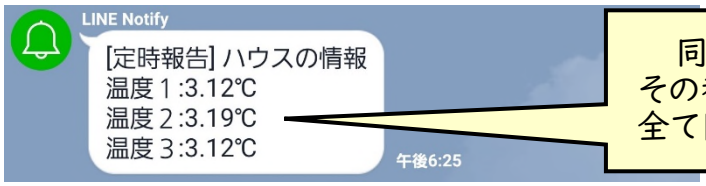
The code in program1.py looks good.

If it is not working as it should, then consider using some general debugging techniques.

Was it helpful or confusing?

Shell

```
>>> %Run program1.py
>>> %Run program1.py
>>>
```



同じデータを通知していますが、
その都度データを取得しているので、
全て同じ値にならないこともあります。

このように「アクセストークンの URL の行」、「データを取得しエラー判定を行う行」、「通知メッセージの行」の3つのセットを増やしたり減らしたりすることで通知するデータの数を変えることができます。

ト.温度センサ以外を使用する時は、対応する key に変更する必要があります。例えば土壌水分センサを利用した場合は、以下のように key は"moisture"となるので、14～15 行目の"temperature"は"moisture"と変更しなくてはいけないことに注意してください。

Key は temperature

Key は moisture

```
#エラー判定　jsonのkeyにerrorが含まれていれば「エラー」を返す
#別のセンサーを使うときはセンサーに応じた"analog"などのKeyを書く
data1 = 'エラー' if ("error" in json1) else json1["temperature"]
data2 = 'エラー' if ("error" in json2) else json2["temperature"]
data3 = 'エラー' if ("error" in json3) else json3["temperature"]
```

ここを変更する必要があります

7. プログラム②:「警報通知プログラム」決まった温度以上・以下になった時に通知

```
 1  # coding:utf-8
 2  import requests
 3
 4  #スマートフォンで設定した wio node のトークンを入れる
 5  url01 = "アクセストークン1"
 6
 7  #request 関数を使ってデータを取得する
 8  json1 = requests.get(url01).json()
 9
10  #"temperature"のデータが取れたときに data に温度を入れる
11  data1 = json1["temperature"]
12
13  #LINE notify の URL
14  #LINE notify のトークン(通知先に応じて変更すること)
15  url99 = "https://notify-api.line.me/api/notify"
16  token = '通知する LINE グループのアクセストークン'
17
18  #高温用メッセージ
19  message1 = '高温注意!:'+str(data1)+'℃'
20  payload1 = {'message' : message1}
21
22  #低温用メッセージ
23  message2 = '低温注意!:'+str(data1)+'℃'
24  payload2 = {'message' : message2}
25
26  headers = {'Authorization' : 'Bearer '+ token,}
27
28  #警報を出したいときの温度を設定する
29  if data1 > 20.0:#高温用:この温度より「高い」と通知する
30      r = requests.post(url99,data=payload1,headers=headers)
31
32  if data1 < 10.0:#低温用:この温度より「低い」と通知する
33      r = requests.post(url99,data=payload2,headers=headers)
34
```

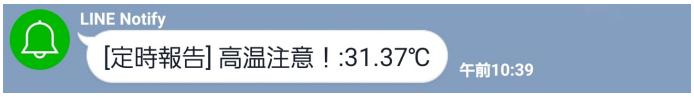
修正しなければいけない箇所は灰色の網掛け部分

05~06行目のアクセストークンを変更

21行目は通知するLINEグループのアクセストークンに変更

29、32行目の設定温度は自由に変えてみましょう

program2.py を開いて、プログラムを作成します。1から 17 行目までは先ほどのプログラム①と同様です。18 行目から 33 行目が設定した温度以上、以下になった際にメッセージを通知するために追加された部分です。プログラムの灰色の網掛け部分を変更して、実行してみてください。高温注意を出したい場合は、温度センサを握って実行してみてください。例えば、以下のような通知が来れば完成です。



リアルタイムでデータ計測して通知を出すのではなく、「データを取得したタイミング」で、設定したしきい値を上回っているか、あるいは下回っているかで警報通知するかを判断しています。そのため、警報プログラムを利用する際は、「定期通知」のプログラム①と、「警報通知」のプログラム②を両方実行して、警報通知だけは短い間隔で実行するように設定する必要があります。また、通知する LINE グループのアクセストークンは「定期通知」と「警報通知」でそれぞれ発行し、通知を分けることをお勧めします。

たとえば、以下のように設定します。

プログラム①「定期通知」・・・1時間おきに実行
⇒1時間おきに通知
プログラム②「警報通知」・・・5分おきに実行
⇒5 分おきにデータを取得し、異常値の時だけ通知

※自動実行間隔の設定は次の章で説明しますが、crontab には以下の通り記述します。

```
00 * * * * sudo python /home/pi/program/program1.py
*/5 * * * * sudo python /home/pi/program/program2.py
```

なお、温度データ取得に失敗した際は警報通知が行われませんので、プログラム①「定期通知」を行うことで、温度が問題なく取得できる状況にあることを確認して下さい。

また、高温時、低温時の通知のどちらかだけを使用したい場合は、29～30 行目、あるいは 31～32 行目に#をつけることでコメント行に変更してください。

8.【応用】プログラム③：センサが3つ以上あり、エラー時に再取得する

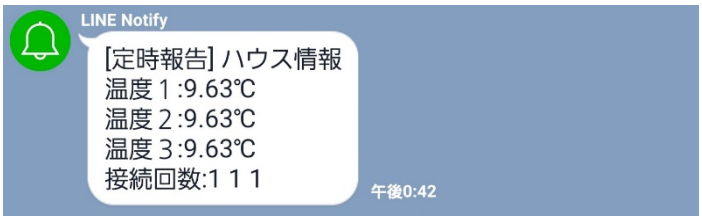
```
# coding:utf-8
import requests
import time

#スマートフォンで設定した wio node のトークンを入れる
url01 = "アクセストークン1"
url02 = "アクセストークン2"
url03 = "アクセストークン3"

#request 関数と for 関数を使って最大3回  5秒おきにデータを取得する
CONNECTION_RETRY = 3
INTERVAL_TIME = 5

#data1
#1<= X < 4 なので 1 2 3 回試行する
for i1 in range(1, CONNECTION_RETRY+1):
    json1 = requests.get(url01).json()
    if ("error" in json1) :
        data1 = 'エラー'
        print('data 1 error!')
        time.sleep(INTERVAL_TIME)
    else:
        data1 = json1["temperature"]
        break

#data2
for i2 in range(1, CONNECTION_RETRY+1):
    json2 = requests.get(url02).json()
    if ("error" in json2) :
        data2 = 'エラー'
        print('data 2 error!')
        time.sleep(INTERVAL_TIME)
    else:
        data2 = json2["temperature"]
        break

#data3
for i3 in range(1, CONNECTION_RETRY+1):
    json3 = requests.get(url03).json()
```

```
40      if ("error" in json3) :
41          data3 = 'エラー'
42          print('data 3 error!')
43          time.sleep(INTERVAL_TIME)
44      else:
45          data3 = json3["temperature"]
46          break
47
48  #LINE notify の URL
49  #LINE notify のトークン（通知先に応じて変更すること）
50
51  url99 = "https://notify-api.line.me/api/notify"
52  token = '通知する LINE グループのアクセストークン'
53
54  #LINE に通知するメッセージを記入
55  message  = 'ハウス情報\n'
56  #'センサ名:'+str(data 番号)+'単位'+'¥n'  センサ名、データ番号、単位を書く
57  message += '温度1:'+str(data1)+'℃'+'\n'
58  message += '温度2:'+str(data2)+'℃'+'\n'
59  message += '温度3:'+str(data3)+'℃'
60  #必要に応じて接続回数を通知する
61  message += '\n'+'接続回数:'+str(i1)+' '+str(i2)+' '+str(i3)
62
63  payload = {'message' : message}
64  headers = {'Authorization' : 'Bearer '+ token}
65
66  r = requests.post(url99, data=payload, headers=headers)
67
```

修正箇所は灰色の網掛け部分

6～8 行目のアクセストークン
23,34,45 行目の”key”は使用するセンサに応じて変更
52 行目の通知するLINEグループのアクセストークンに変更
55 行目の通知するメッセージを必要に応じて変更
57～59 行目のセンサ名、単位を変更

program3.py を開いて、プログラムを作成します。10 から 46 行目まではプログラム①の 8 行目から 15 行目までと同じ働き（データ取得）を行っています。データを取得使用したが、回線の混雑などがあった場合に再実行するようになっています。なお、再実行の間隔と実行回数は 11-12 行目で設定できますが、通常はこの設定で問題なく動作します。網掛け部分をこれまでのプログラム同様に修正すると、以下の通知が来ます。



接続回数は、データを取得しようとした回数です。通常は 1 か 2 ですが、電源が抜けていたり、Wi-Fi ルータの電源が切れていたりすると 3 になります。便利な機能ですが、必要としない場合は 61 行目の頭に「#」をつけてコメントに変えてください。

次にプログラム中に出てくる「for」について説明します。「for」は繰り返し処理を行うときに使いますが、詳しい使い方については、市販の入門書等に任せて、ここでは「センサの数を増やしたとき」、「センサの種類を変えたとき」にどのように修正すれば良いかを解説します。以下のようにセンサの数を増やした際は黄色網掛けの数字を増やしてください。センサの種類を変えた際は灰色の網掛けの文字列（key）を変更してください。

```
#data1
for i1 in range(1, CONNECTION_RETRY+1):
    json1 = requests.get(url01).json()
    if ("error" in json1) :
        data1 = 'エラー'
        print('data 1 error!')
        time.sleep(INTERVAL_TIME)
    else:
        data1 = json1["temperature"]
        break
```

コピー&ペーストして数字を変更

```
#data2
for i2 in range(1, CONNECTION_RETRY+1):
    json2 = requests.get(url02).json()
    if ("error" in json2) :
        data2 ='エラー'
        print('data 2 error!')
        time.sleep(INTERVAL_TIME)
    else:
        data2 = json2["temperature"]
        break
```

数字を変更

水分センサなら moisture に変更

9.【応用】プログラム④:3つ以上のデータを csv 形式で保存する方法

```
# coding:utf-8
import requests
import os

#スマートフォンで設定した wio node のトークンを入れる
url01 = "アクセストークン1"
url02 = "アクセストークン2"
url03 = "アクセストークン3"
url04 = "アクセストークン4"
url05 = "アクセストークン5"
url06 = "アクセストークン6"

#request 関数を使ってデータを取得する
json1 = requests.get(url01).json()
json2 = requests.get(url02).json()
json3 = requests.get(url03).json()
json4 = requests.get(url04).json()
json5 = requests.get(url05).json()
json6 = requests.get(url06).json()

#エラー判定 json の key にエラーが含まれていれば「空白」を返す
#別のセンサを使うときはセンサに応じた"analog"などの Key を書く
data1 = '' if ("error" in json1) else json1["temperature"]
data2 = '' if ("error" in json2) else json2["temperature"]
data3 = '' if ("error" in json3) else json3["temperature"]
data4 = '' if ("error" in json4) else json4["temperature"]
data5 = '' if ("error" in json5) else json5["temperature"]
data6 = '' if ("error" in json6) else json6["temperature"]

#LINE notify の URL
#LINE notify のトークン(通知先に応じて変更すること)

url99 = "https://notify-api.line.me/api/notify"
token = '通知する LINE グループのアクセストークン'

#LINE に通知するメッセージを記入
message  = 'ハウス情報\n'
#'センサ名:'+str(data 番号)+'単位'+'¥n' センサ名、データ番号、単位を書く
message += '温度1:'+str(data1)+'単位'+'\n'
```

```
40 message += '温度2:'+str(data2)+'単位'+'\n'
41 message += '温度3:'+str(data3)+'単位'+'\n'
42 message += '温度4:'+str(data4)+'単位'+'\n'
43 message += '温度5:'+str(data5)+'単位'+'\n'
44 message += '温度6:'+str(data6)+'単位'
45
46 payload = {'message' : message}
47 headers = {'Authorization' : 'Bearer '+ token}
48
49 r = requests.post(url99, data=payload, headers=headers)
50
51 #csv の保存のために時刻を取得
52 timestamp = 'date +%F" %H:%M:%S"'
53 current_time = os.popen(timestamp).readline().strip()
54
55 #csv で保存するデータを組み合わせる
56 data_set    = str(current_time)
57 data_set += ',' + str(data1)
58 data_set += ',' + str(data2)
59 data_set += ',' + str(data3)
60 data_set += ',' + str(data4)
61 data_set += ',' + str(data5)
62 data_set += ',' + str(data6)
63 data_set += '¥n'
64
65 #/home/pi/data.txt というファイルにデータを保存する
66 fout = open('/home/pi/data.txt','at')
67 fout.write(data_set)
68 fout.close()
69
```

修正箇所は灰色の網掛け部分

6～11行目のアクセストークン

23～28行目の”key”は使用するセンサに応じて変更

34行目の通知するLINEグループのアクセストークンに変更

37 行目の通知するメッセージを必要に応じて変更

39～44行目のセンサ名、単位を変更

66 行目 data.txt を必要に応じて別の名前に変更

program4.py を開いて、プログラムを作成します。プログラム③のようにエラー時に繰り返し実行をするようにはなっていません。必要に応じて、13-28 行目をプログラム③で使用した for 部分に置き換えてください。

ここでは取得したデータを保存する部分である 51 行目から 68 行目について説明します。はじめに、「時刻の取得」について説明します。以下の 51-53 行目で時刻を取得しています。

```
#csv の保存のために時刻を取得
timestamp = 'date +%F" %H:%M:%S"'
current_time = os.popen(timestamp).readline().strip()
```

%H は「時」、
%M は「分」、
%S は「秒」を示しています。

次に、55-63 行目を説明します。保存するデータ数を増やしたいとき減らしたいときはこれらの行を変更します。必要のない行（例えばセンサの数が3つなら data4 から data6 の行）は削除、あるいは「#」をつけてコメントに変更します。

```
#csv で保存するデータを組み合わせる
data_set  = str(current_time)
data_set += ',' + str(data1)
data_set += ',' + str(data2)
data_set += ',' + str(data3)
data_set += ',' + str(data4)
data_set += ',' + str(data5)
data_set += ',' + str(data6)
data_set += '\n'
```

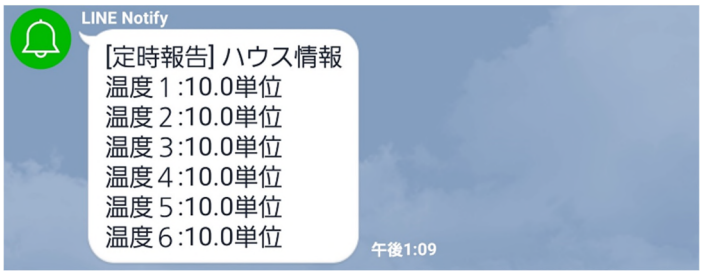
7つ目のセンサをつけた時は次の行に
data_set += ',' + str(data7)
を追加

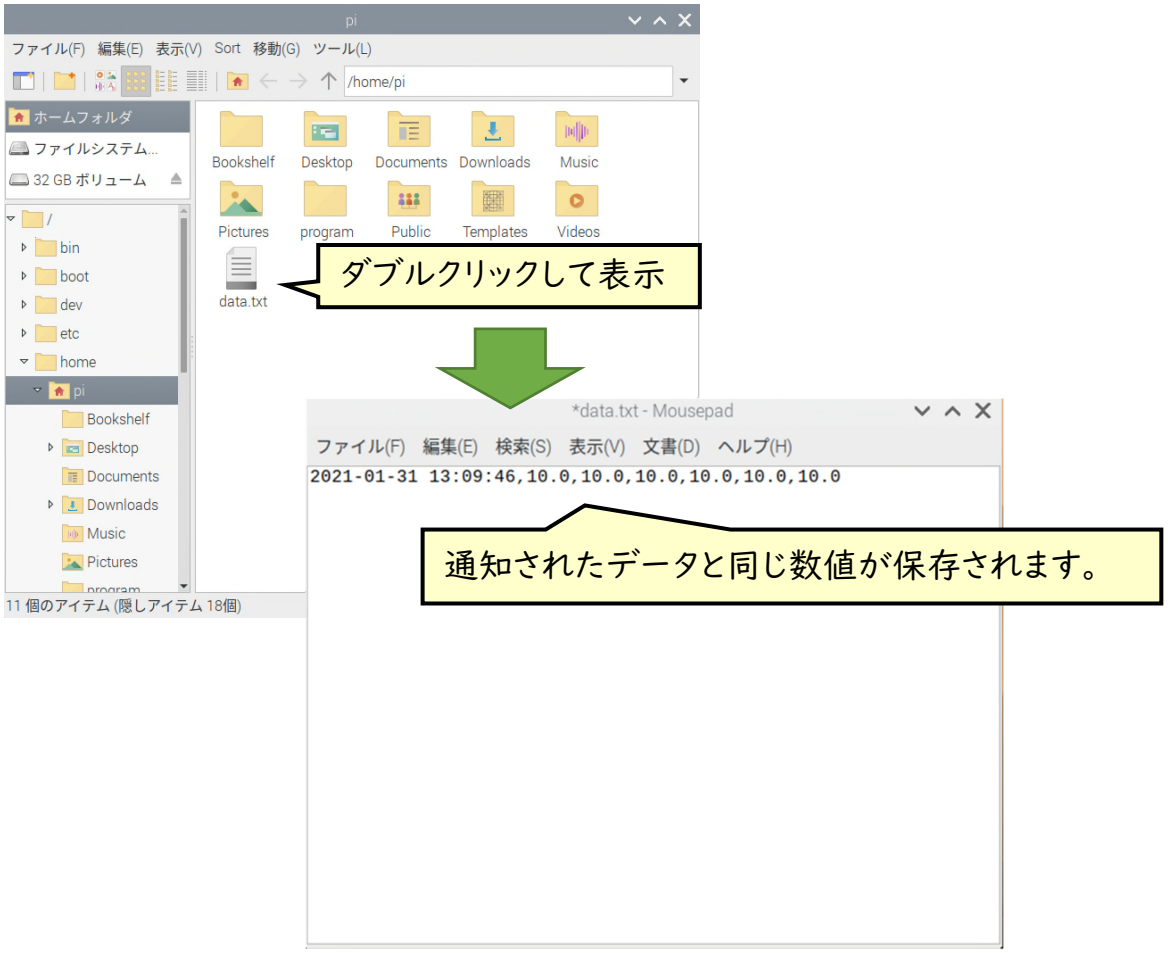
次に、65-68 行目を説明します。データを保存するファイル名は灰色の網掛け部分になります。このままだと、保存する先は「pi」フォルダ内で、「data.txt」という名前のファイルになります。

```
#/home/pi/data.txt というファイルにデータを保存する
fout = open('/home/pi/data.txt','at')
fout.write(data_set)
fout.close()
```

データの取得間隔によって名前を変えると良いでしょう。たとえば、
10 分間隔なら「data10.txt」
60 分間隔なら「data60.txt」など

プログラムを実行すると、以下のような通知の他に、「pi」フォルダ内に「data.txt」と呼ばれるファイルが保存されているはずです。ダブルクリックして表示すると以下のようになります。





第4章で説明する crontab を用いて自動実行させるときは、以下のように記述してみてください。

●10 分間隔でデータを保存したいとき

```
*/10 * * * * sudo python3 /home/pi/program/program4.py
```

●1 時間おき（60 分間隔）でデータを保存したいとき

```
00 * * * * sudo python3 /home/pi/program/program4.py
```

第４章　プログラムの自動実行

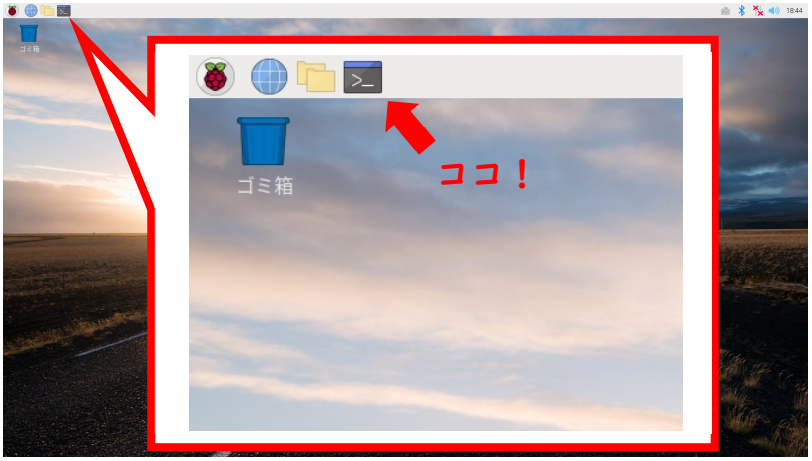
ラズベリーパイには、crontab（クロンタブ）と呼ばれる、プログラムを決めた時間に実行してくれる機能が搭載されています。これを使って、先ほど作成したプログラムを自分の好きなタイミングで動かすことができます。たとえば １ 時間ごとや、早朝の温度が上がる時間の6時から 18時までなど自由に通知間隔を設定できます。

設定方法は次の 1.～3.の手順です。

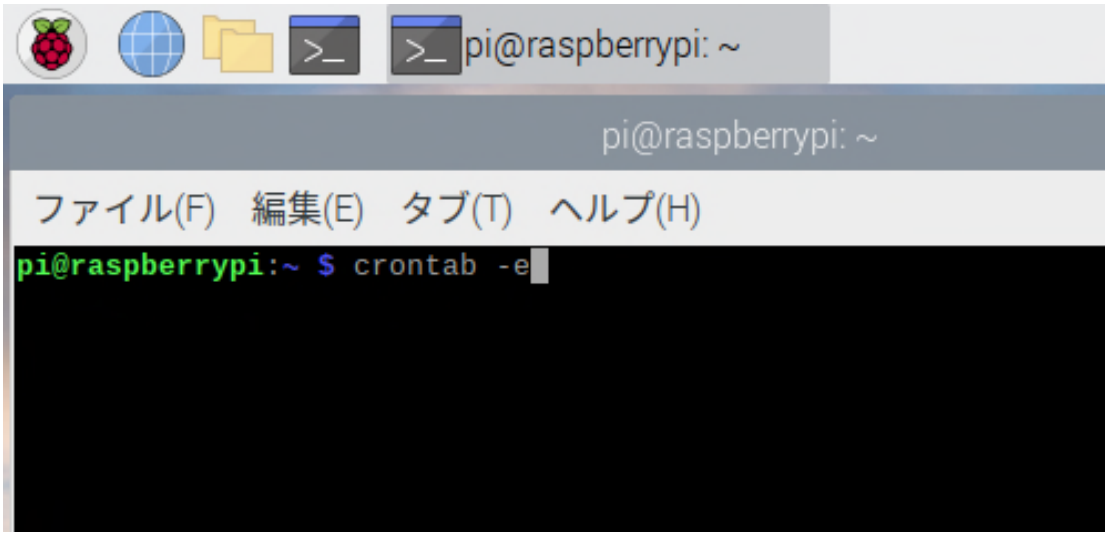
1. ターミナルから crontab を開く
2. crontab の初期設定をする。
3. crontab を使って通知間隔の設定を行う。

1. ターミナルから crontab を開く

イ. デスクトップの左上、ファイルのアイコンの隣にある黒い四角（LX terminal）をクリックしてください。

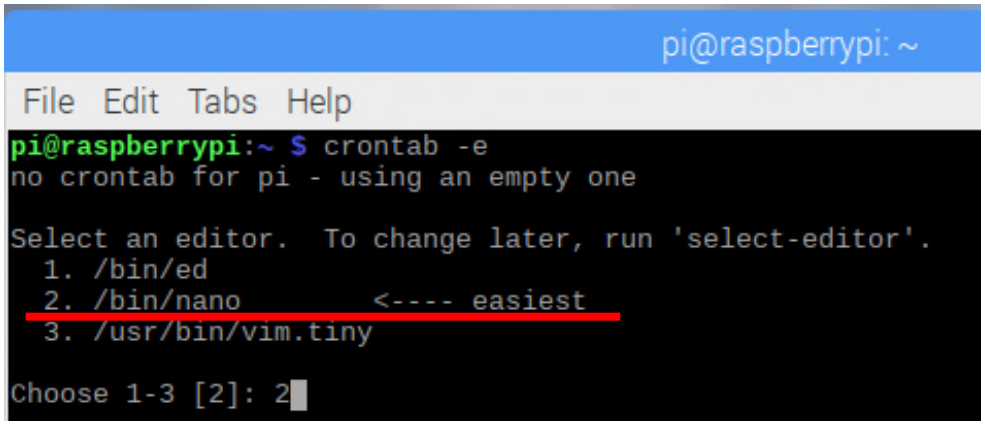


ロ. この画面のように、 crontab -e と打って、Enter を押してください。

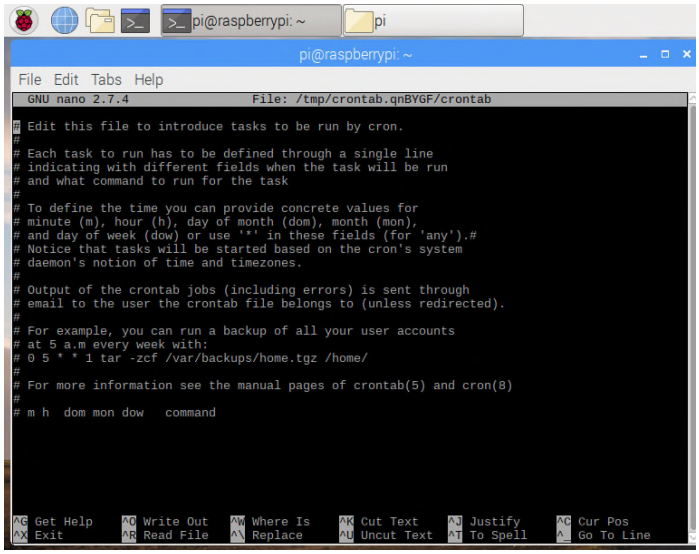


2. crontab の初期設定をする。（2回目以降は初期設定画面が出てきません）

イ. /bin/nano（と呼ばれるエディタ）を使います。画面の表示をよく見て、該当する番号を打ち Enter を押してください。（この画面の場合は「2」です。）

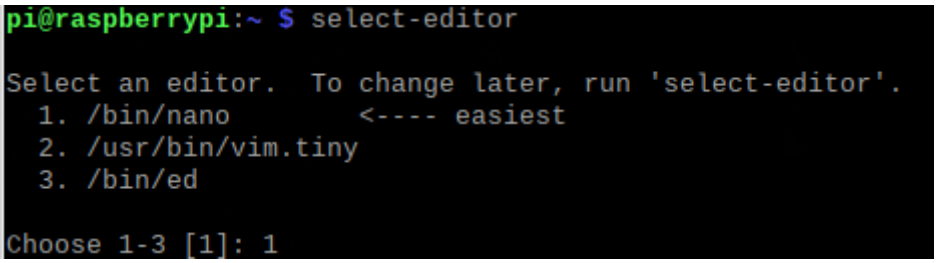


ロ. この画面になりましたら初期設定完了です。次に通知間隔の設定に移ります。
Ctrl を押しながら X を押すと終了し、元の画面に戻ります。



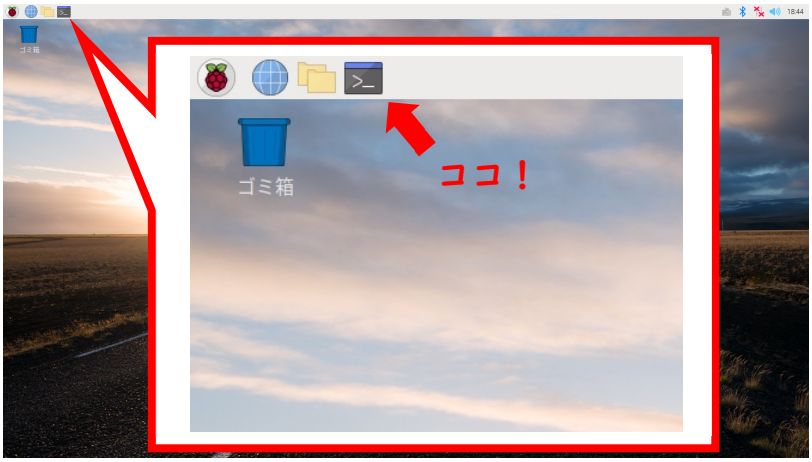
ハ. 誤って「イ」の操作で /bin/nano 以外を選択してしまった場合は、ターミナルを一度終了した後、ターミナルを再度起動して、以下の下線部のコマンドを実行して /bin/nano を選択してください。

pi@raspberrypi: ~ $ select-editor

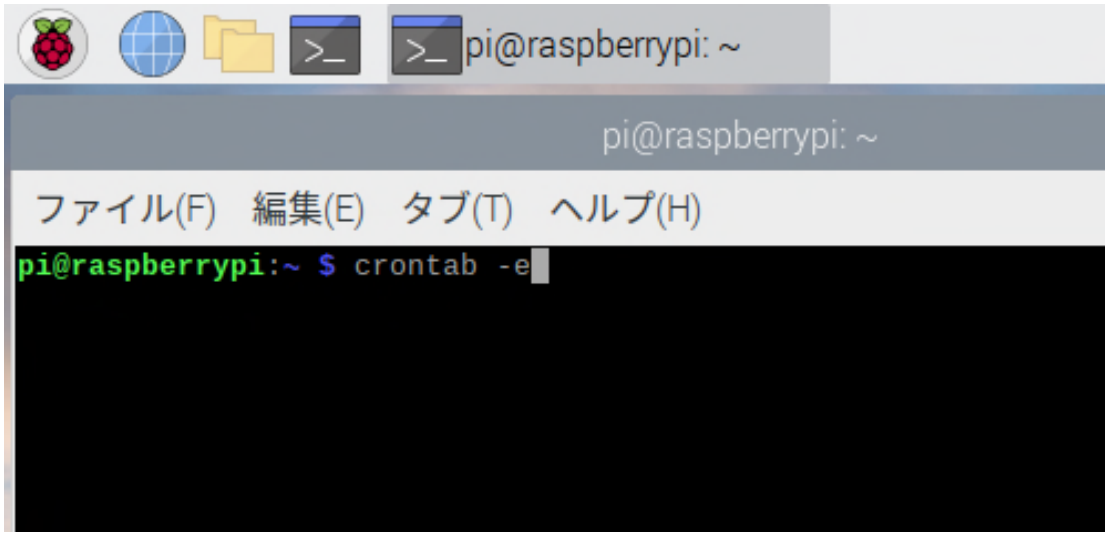


3. crontab を使って通知間隔の設定を行う。

イ. 手順 1 と同様に、デスクトップの左上、ファイルのアイコンの隣にある黒い四角（LXterminal）をクリックしてください。
（手順 2 に続けて行う場合はこの手順を飛ばします）



ロ. この画面のように、 crontab -e と打って、Enter を押してください。



ハ.この画面が出たら、□で囲んだ部分に次の文を書きます。

```
*/2 * * * * sudo python3 /home/pi/program/test.py
```

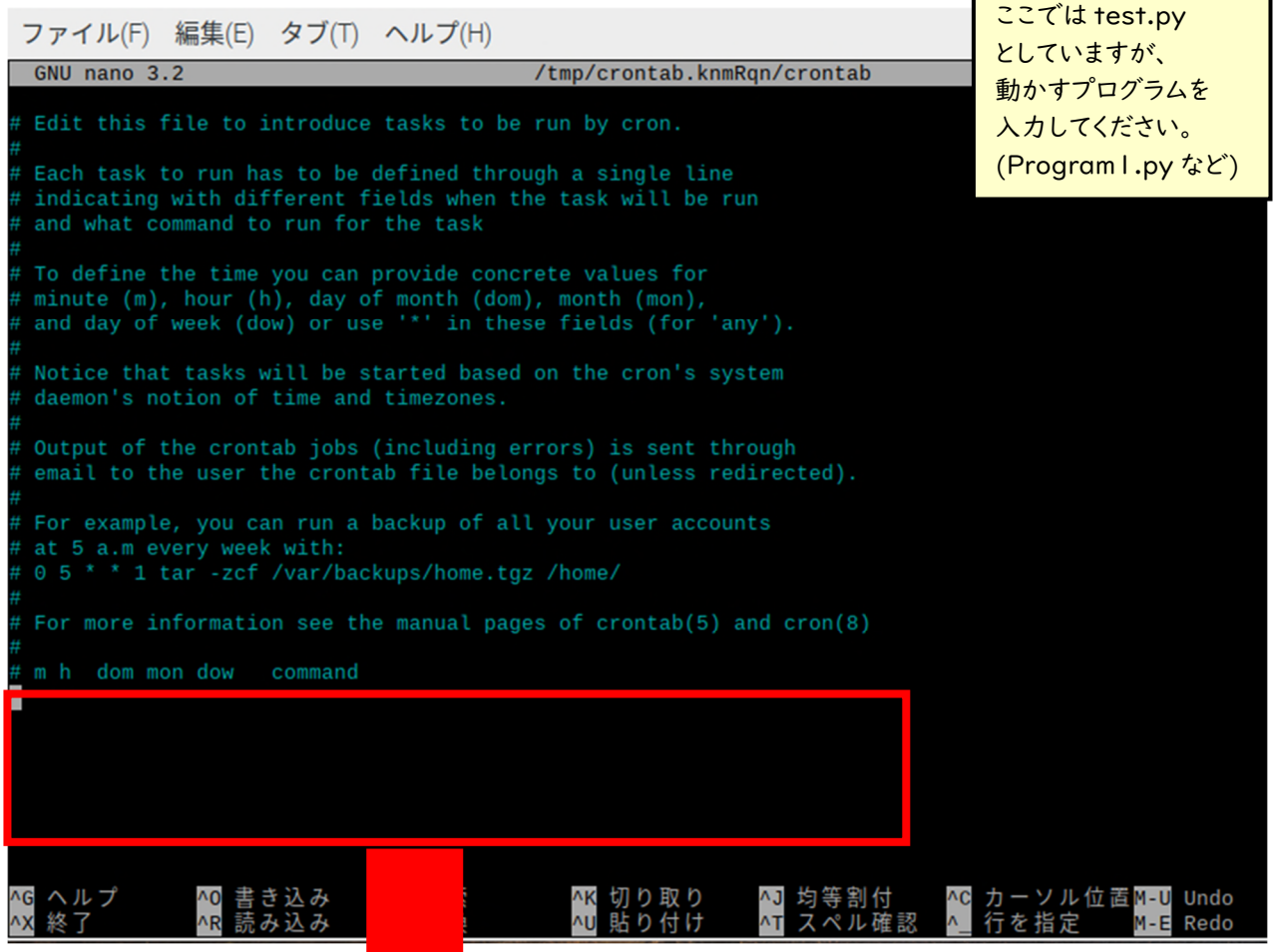
2 分間隔で実行

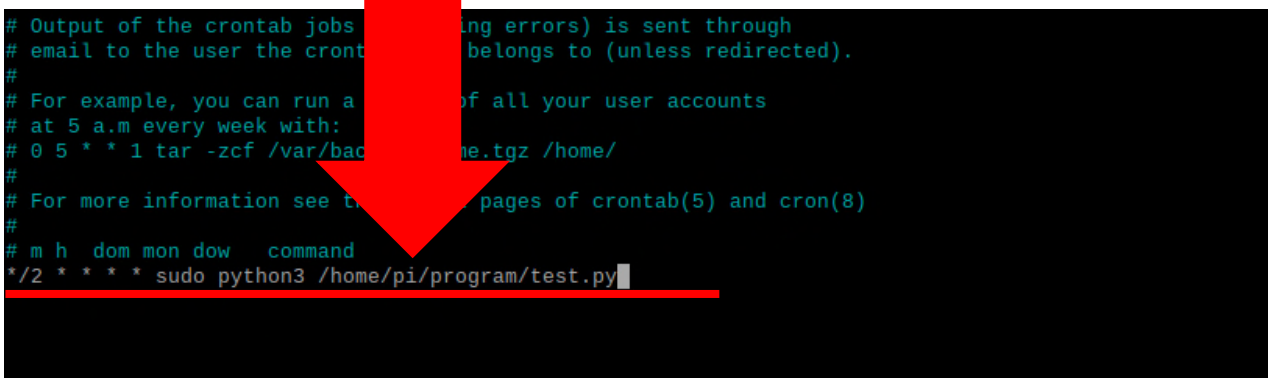
プログラムは Python を使う

プログラムのあるところ program はフォルダ名

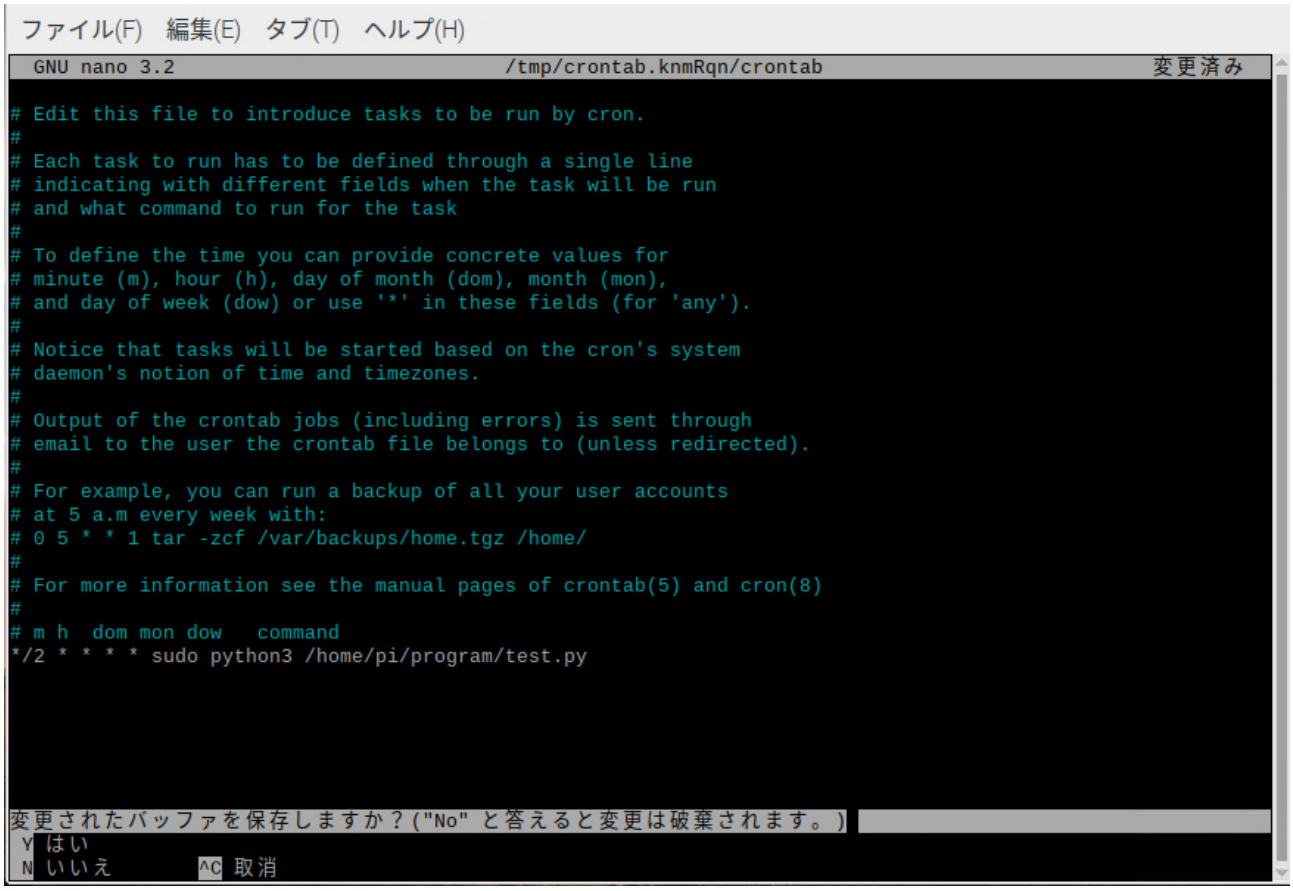
ファイル名

ここでは test.py としていますが、動かすプログラムを入力してください。(Program1.py など)



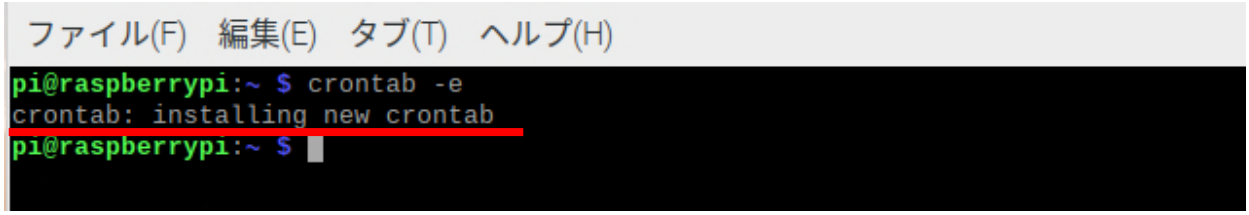


ニ. 書き込んだあとはキーボードの「Ctrl」キーを押しながら X を押してください。
保存するか聞かれるので「y」を押しましょう。その次は「Enter」を押してください。
2 分間隔で通知が来ます。来ない場合はもう一度やり直してみてください。



キーボードで「y」を押します

キーボードで「Enter」を押して保存します



この画面が出れば保存となります。

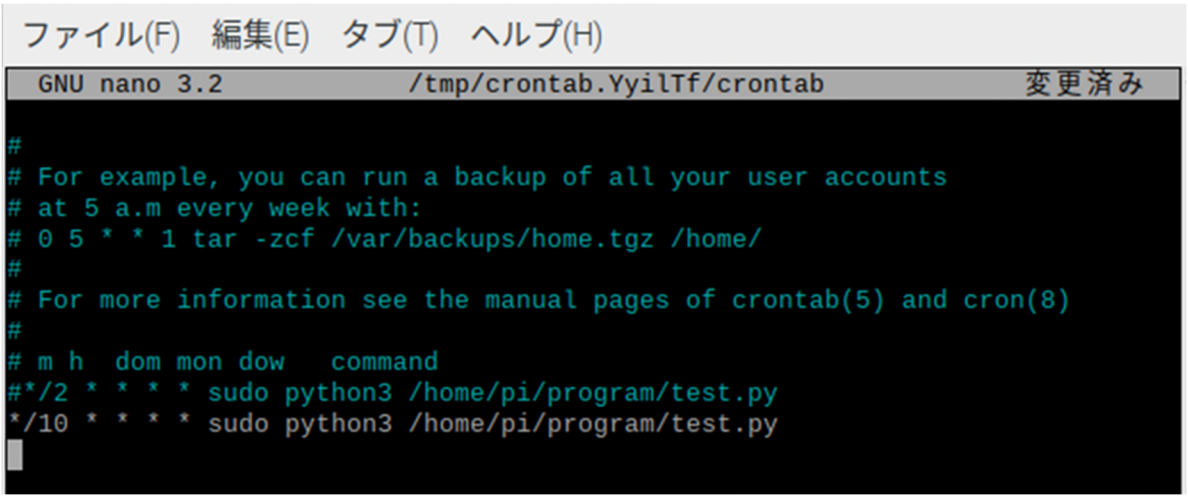
ホ. 指定した LINE グループに通知が来たのを確認できたら、
下記の設定例を参考に通知間隔を変えてみましょう。
その前に今作成した設定は使わないので、先頭に#「シャープ」をつけましょう。
#をつけることでその設定は動かなくなります。

```
#*/2 * * * * sudo python3 /home/pi/program/test.py
```

2 分間隔で実行

プログラムは Python を使う

プログラムのあるところ program はフォルダ名



また動かしたいときは#を消してください。
crontab を設定したときは必ず Ctrl + X で終了し、忘れずに保存しましょう。

〔crontab の設定例〕

例 1 ）10 分おき
*/10 * * * * sudo python3 以下略

例 2 ） 1 時間おき
00 * * * * sudo python3 以下略

例 3 ）6-18 時まで 10 分おき
*/10 6-18 * * * sudo python3 以下略

例 4 ）6 時、9 時、10 時、12 時、13 時、14 時、15 時、22 時
00 6,9,10,12,13,14,15,16,22 * * * sudo python3 以下略

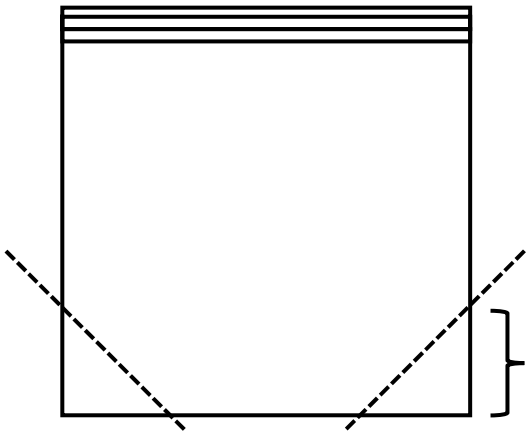
第５章 現場での設置方法

本章では現場に実際に設置する際の方法や使用できるセンサについて説明します。

１．マイコンの防水方法

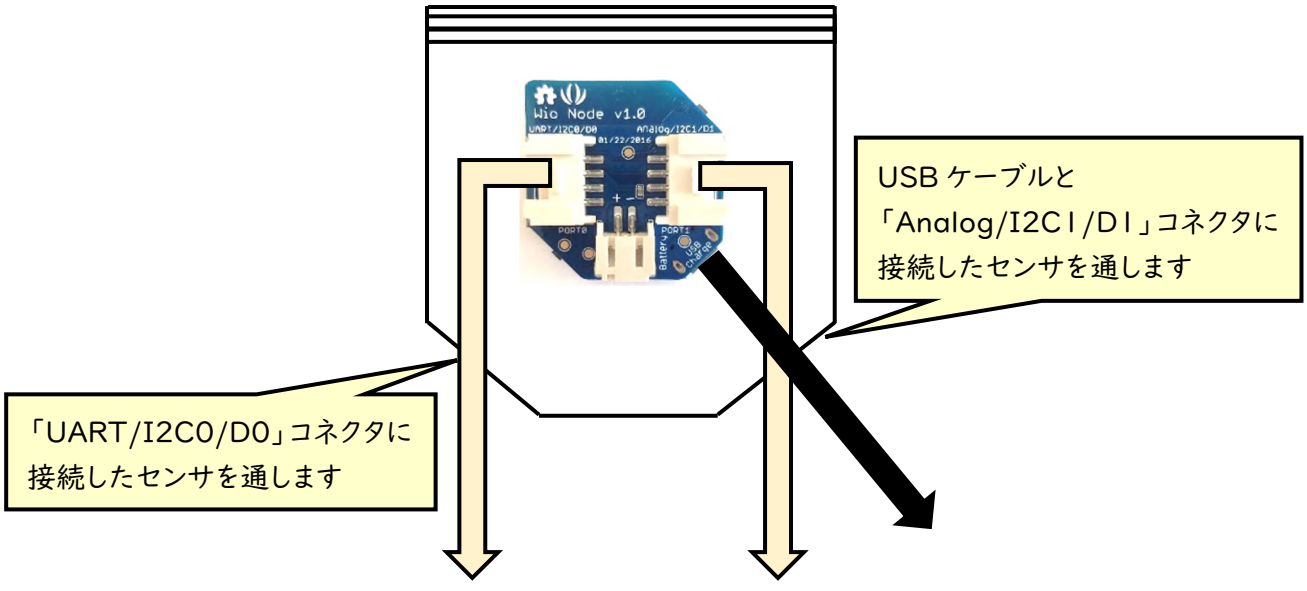
●必要な材料
[簡易かつ短期間（半年ほど）使用する場合]
・ジッパーバッグ（市販の M サイズ以上をオススメします）
・ビニールテープ又は自己融着テープ*
*日東シンコー、自己融着性ブチルゴムテープ No.１５ など

イ．まずはじめにジッパーバッグを用意して、袋の隅をカットします。



通すセンサによりますが、
１.5cm もあれば十分です。

ロ.その後、袋の中に電源用の USB ケーブルとセンサをつなげた Wio Node を入れ、カットして穴をあけた部分から USB ケーブルとセンサを通します。



ハ．センサを通した後は、センサ部と袋の隅を「ビニールテープ」もしくは「自己融着テープ」で巻けば完成です。マイコンのコネクタ側を下にし、センサを１つだけつけた場合は以下の写真（左側）になります。

ミスト灌水など、ミストがかかるような場所には設置しないよう、高い位置から吊るすなどしてください。また、袋ごと地面に置くと中に水が入ることがあります。ハウス内に設置する際はマイカ線などを利用して、できるだけ吊るすようにしてください。

●必要な材料

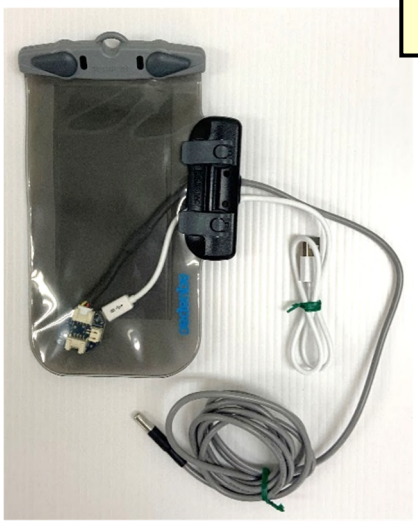
[長期間（１年中ハウス内に設置して）使用する場合]

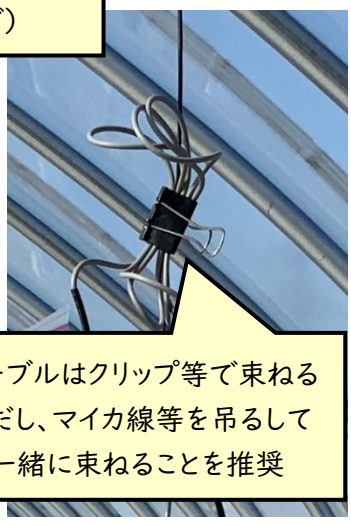
ジッパーバッグによる簡易な防水袋を作り変えればよいですが、作り変えるのが面倒であったり、高価でも長期間使用したい場合は、市販の防水バッグを利用してください。

・市販防水バッグ*

*Aquapac Wire Through Case S (TC Clip 付き) 型番 548

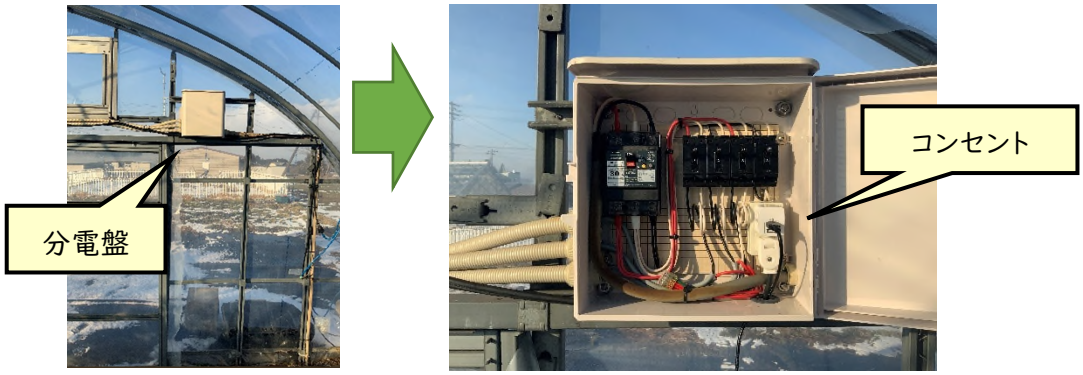
市場価格約 7,000 円





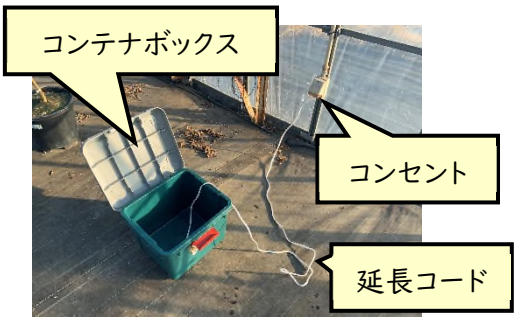
2. 100V 電源からの設置方法

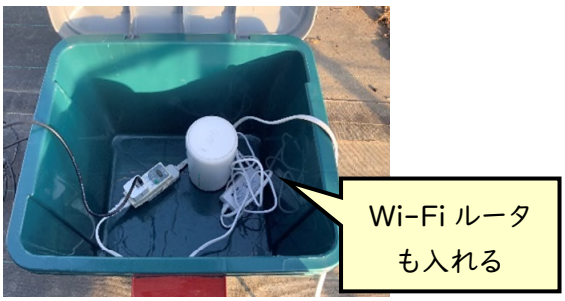
イ. 【ハウス内の分電盤内などにコンセントがあり、その周辺に設置したい場合】
分電盤内のコンセントに USB-AC アダプタを差し込み使用します。

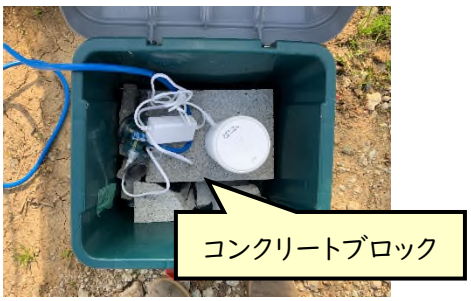


ロ.【ハウス内のコンセントから測定したい場所が遠い場合】
100V 電源延長ケーブルで延長します。電源ケーブルを地面に置いてしまうと、灌水時に漏電する危険性があるため、電源ケーブルはコンテナ（RVBOX など）内に設置します。コンテナ内にはコンクリートブロックなど適当な重りを入れて動かないようにします。

●必要な材料
・コンテナボックス*
・100V 電源延長ケーブル（電工ドラムなどでも可）
・漏電ブレーカ**
・コンクリートブロック等
* アイリスオーヤマ 収納 BOX RVBOX400 など（約 1,500 円）
**テンパールビリビリガード プラグ型漏電遮断器 など（約 2,500 円）







可能であれば万が一の漏電を防ぐため、コンセントには漏電ブレーカを設置してください。使用しない箇所はビニールテープで塞ぐか、予算が許せば防水延長コードや、普通のプラグを防水プラグ化するプラグカッパーなどを利用してください。

※【参考】100V電源はハウス内に無いけれども、ハウス隣の機械庫にはある、といった場合に、一時的にハウス内まで電工ドラム等で引き込む場合は以下のように行います。

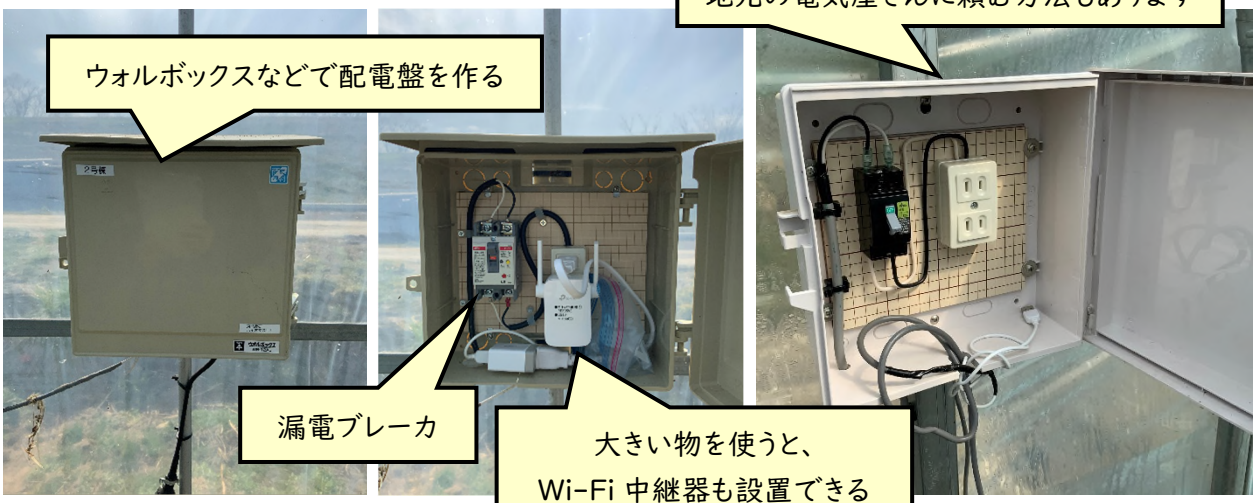
ハウスまで電工ドラムで引き込む

ハウス近くにあるコンセント

電工ドラムの電線は火災防止のため、すべて引き出しておく

各ハウスへ

多くのハウスに設置する場合は、地元の電気屋さんに頼む方法もあります



ウォルボックスなどで配電盤を作る

漏電ブレーカ

大きい物を使うと、Wi-Fi 中継器も設置できる

RV ボックスからの延長コードを自作して、ウォルボックス内に設置する方法もあります

ハウス近くの 100V 電源からハウスまで引き込む際に、電線（電工ドラムのキャプタイヤケーブル）の上を軽トラなどが通過する場合は、電線が断線して漏電事故が起きることを防ぐため、ケーブルプロテクタの中にキャプタイヤケーブルを通すなどしてください。

3. 電源から設置場所が遠い場合の設置方法（USB ケーブルの延長について）

イ.【USB 延長ケーブルで延長する】

●必要な材料

・USB 延長ケーブル（5m）*

・ビニールテープ

*エレコム U2C-JE50BK [エコ USB2.0 延長ケーブル 5m]など（約 500 円）

Wio Node に接続する USB ケーブル（microB）と接続してビニールテープを巻いて絶縁します。USB 延長ケーブルで延長する際は 2 個までとすることを推奨します。長くなればなるほど電圧が降下し、Wio Node が動作しません。（電圧降下は、電線の太さが細ければ細いほど、電線が長ければ長いほど起こります。そのため、細い電線を使った USB ケーブルでは延長できません。延長する際は Wio Node に接続する USB ケーブル（microB）は急速充電対応と書いてあるものを使用することを推奨します。）

ロ.（参考）【USB 延長ケーブルを自作する】

電子工作が得意な方向けの参考情報です。

●必要な材料

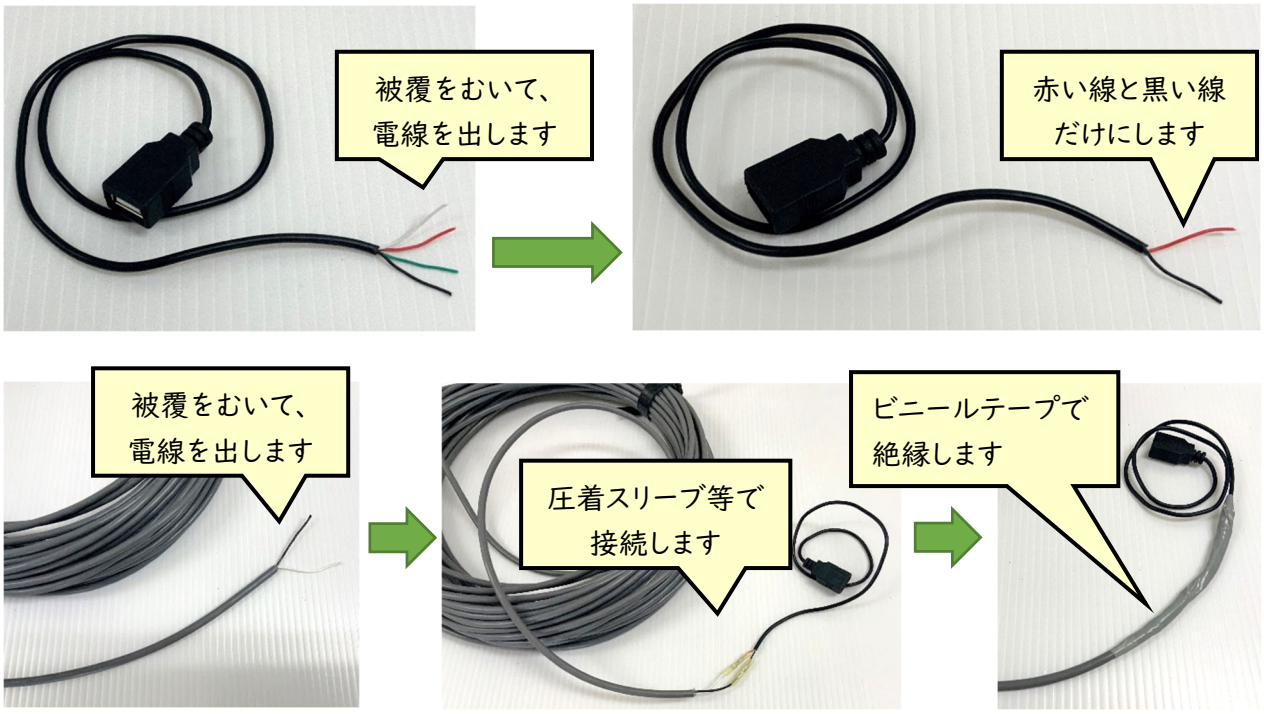
・USB ケーブル（microB）または USB 延長ケーブル

・シールド線（2線 0.5sq）*

・ビニールテープ、熱収縮チューブ、圧着スリーブなど

* 富士電線工業 MVVS 0.5sq×2 芯 ケーブルなど 100m 市場価格 約 8,000 円
1m 単位でも流通していますが、100m 巻が安価です。

100m 巻を
4 等分して、
25m にしたもの



太い電線を利用することで、電圧降下を抑えた電源用のUSB延長ケーブル（最大25mほど）を作ることができます。USB ケーブル内には4つの電線がありますが、赤い線（5V）と黒い線(GND)だけ使用します（必ずテスターで確認して下さい）。USB ケーブルをカットして、シールド線を用いてつなぎ延長してください。その際はハンダ付けか圧着スリーブを用いて、ショートしないように熱収縮チューブによる絶縁処理だけでなく、ビニールテープを用いて再度絶縁処理をするようにしてください。短い USB 延長ケーブルではなく、急速充電対応のUSBケーブルを用いると電線が太いため作りやすいです。うまく電線の被覆をはがせない、あるいは圧着スリーブをかしめても電線が抜けてしまう場合は太い電線の USB ケーブルに変えることを検討してください。

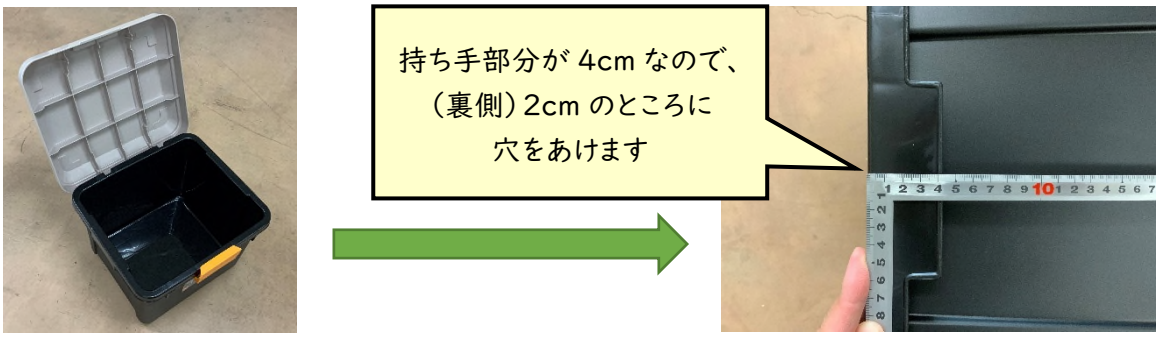
4.【参考】100V 電源が無い場合に太陽光発電システムを自作する

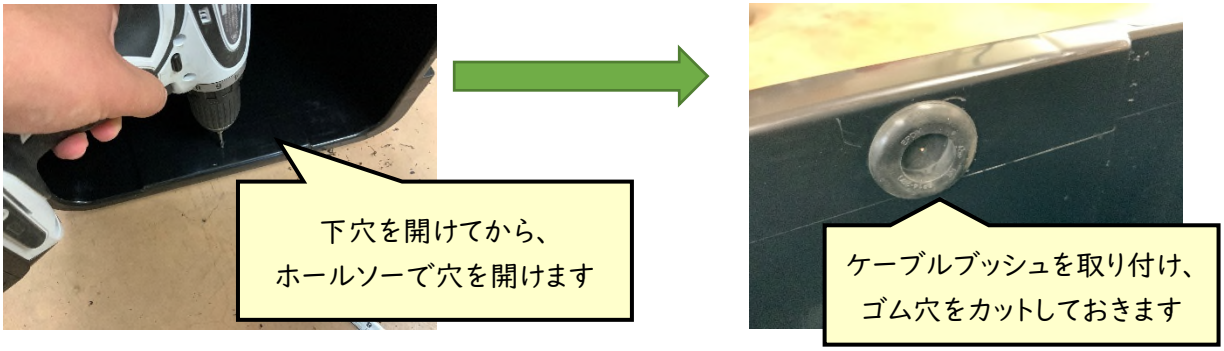
ハウス内やその近くに 100V 電源がない場合は、自動車用のバッテリーとソーラーパネルを利用して太陽光発電システムを作り、電源とすることができます。モバイル Wi-Fi ルータ（NEC Aterm MR05LN など）と Wio Node2 つを使用することを前提に、積雪の心配がない 4 月から 11 月まで利用することを想定しています。また、Wio Node の代わりに市販のネットワークカメラ（PLANEX スマカメ）等を接続しても構いません。

●必要な材料（約 3 万円）
・ソーラーパネル（50W）(1)
・バッテリー（12V36Ah）(2)
・充放電コントローラ(3)
・シガーソケットUSB(4)
・自動車用メスソケット(5)

・ウォルボックス×2(6)
・MC4コネクタ付きケーブル(7)
・コンテナボックス(8)
・その他（ビニールテープや圧着端子、スピーカーケーブルなどの電線）

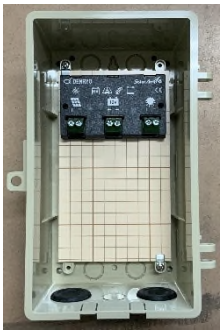
（1）JAsolar AT-MA50A ソーラーパネル 単結晶 50W 約 7,500 円
（2）LONG 12V36Ah ディープサイクルバッテリ（U1-36NE） 約 9,000 円
（3）電菱 太陽電池充放電コントローラ Solar Amp B SA-BA10 約 3,000 円
（4）BUFFALO USB カーチャージャー BSMPS2401P2BK 約 700 円
（5）槌屋ヤック 車用 ソケット分配器 コードタイプ PZ-724 など 約 600 円
（6）未来工業 ウォルボックス WB－1AJ 約 1100 円
（7）蓄電システム.com ソーラーパネル接続ケーブル 2SQ など 約 3500 円
（8）アイリスオーヤマ RV ボックス 400 約 1,500 円

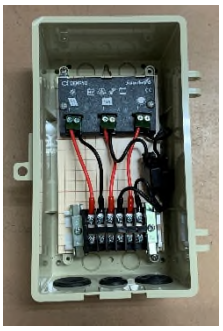


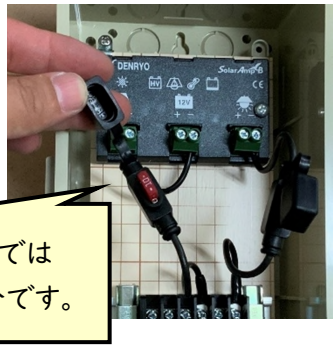


RVボックス内にはバッテリーや充放電コントローラを設置し、充放電コントローラからは「ソーラーパネル用のケーブル」とバッテリーから 12V 電源を取り出す「メスソケット用の電源ケーブル」を通す必要があります。コンテナのサイド部分の内側に穴をあけます。ホールソーなどで 2cm 程の穴をあけます。ケーブルブッシュがあればそれを付けますが、ない場合は安価なエアコンパテで代用してください。水が浸入しないようにすることが目的ですので、必ずしもきれいに作る必要はありません。ホールソーの扱いに慣れていない場合は、必ずしも丸い穴である必要はありませんので、ドリルで穴を開けてからニッパーなどで穴を広げてください。

充放電コントローラを入れるケースを作ります。ウォルボックスの下部のケーブルを通せるようにします。（ケーブルブッシュがあれば使用します）

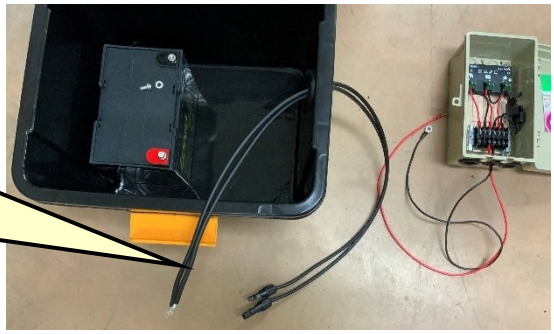






ウォルボックス内に充放電コントローラをねじ止めします。電線の取り回しを楽にするため、なるべく上に取り付けます。もし端子台があればウォルボックス内に取り付けます。充放電コントローラのバッテリー側と、負荷側のマイナス極は「ヒューズ」を取り付ける必要があります。ヒューズは自動車用のカーヒューズで構いません。端子台を設置する場合は「平型」か「ミニ平型用」のものを使用すると良いでしょう。端子台を設置しない場合はスペースに余裕があるので、ガラスヒューズ用のものでもかまいません。入手しやすいものを利用してください。なお、制作時には充放電コントローラの説明書をよく読み、指示に従ってください。

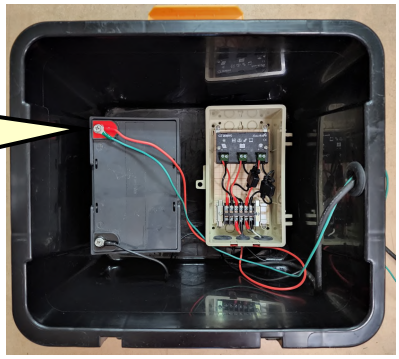
ソーラーパネル用ケーブルは通しておきます。MC4 コネクタ側はボックスの外側になります。



RV ボックスの中にバッテリーを置き、充放電コントローラから接続できるようケーブルを準備します。ケーブルはホームセンターで販売している 2.0SQ のスピーカー線等で問題ありません。ただし、芯径は 2.0SQ のものか、少なくとも 1.25SQ のものを使用してください。なお、充放電コントローラを設置したウォルボックスはバッテリーの隣に設置するので、スピーカー線は 50cm もあれば十分でしょう。メンテナンス性を確保する場合は RV ボックスの外から出して地面におけるように 1m 弱の長さでもかまいません。

使用時には
地面に打ち込みます

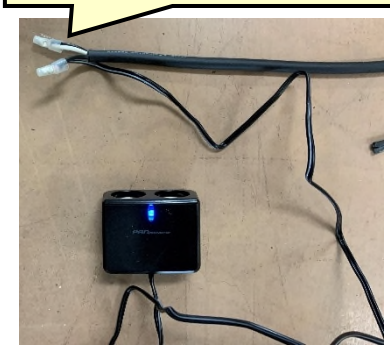
バッテリーに接続し、
プラス極側には
アース線を取り付けます



充放電コントローラとバッテリーを接続します。感電しないよう注意して作業を行ってください。アース線*はホームセンターや電気屋などで入手しやすいものを使用してください。
*ELPA アース棒 W型 3m など

ソケットと2芯の電線
を用意します。

絶縁被覆付閉端接続子など
で接続し、ビニールテープで
絶縁します。

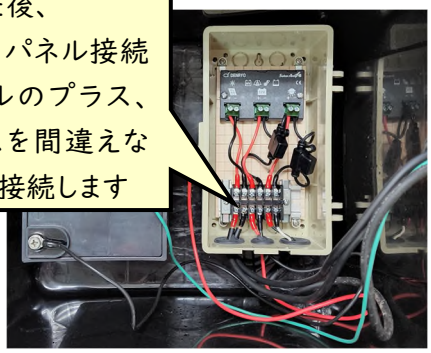


ウォルボックス内への設置は、
超強力両面テープなどで行
い、ケーブルの固定はケーブ
ルクリップを使うと簡単です。

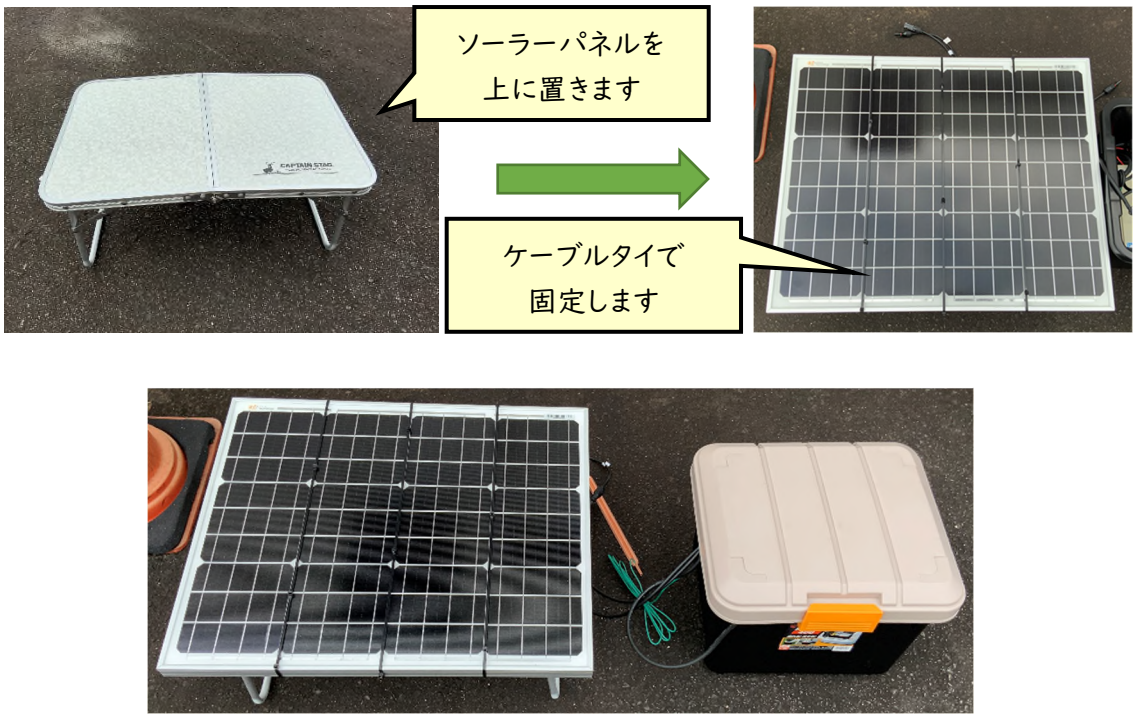
負荷側を作成します。ソケット分配器にスピーカー線やビニルキャプタイヤコードなどを接続し、ウォルボックス内に設置します。その後、充放電コントローラに接続してください。必ずテスターでプラスマイナスが逆になっていないかを確かめてください。逆になった状態で、USBカーチャージャーを接続すると壊れますので注意してください。

ソーラーパネル側に
プラス、マイナスが
書いてあります。

確認した後、
ソーラーパネル接続
ケーブルのプラス、
マイナスを間違えな
いように接続します



ソーラーパネル用ケーブルを用いて、ソーラーパネルと接続します。ソーラーパネル側のコネクタにプラス極とマイナス極が書いてあるので、それに合わせてソーラーパネル用のケーブルを充放電コントローラに接続します。



完成図

ソーラーパネルはRVボックスの上に置き、ケーブルタイや被覆ビニール線で固定するか、キャンプ用のテーブルなどの上に置いて、同様にケーブルタイなどで固定しても構いません。負荷用のウォルボックスはハウス内に設置するか、RVボックス内に入れて使用してください。ソーラーパネルは南向きに設置し、雑草で覆われないように注意してください。

5. 温度センサの設置方法

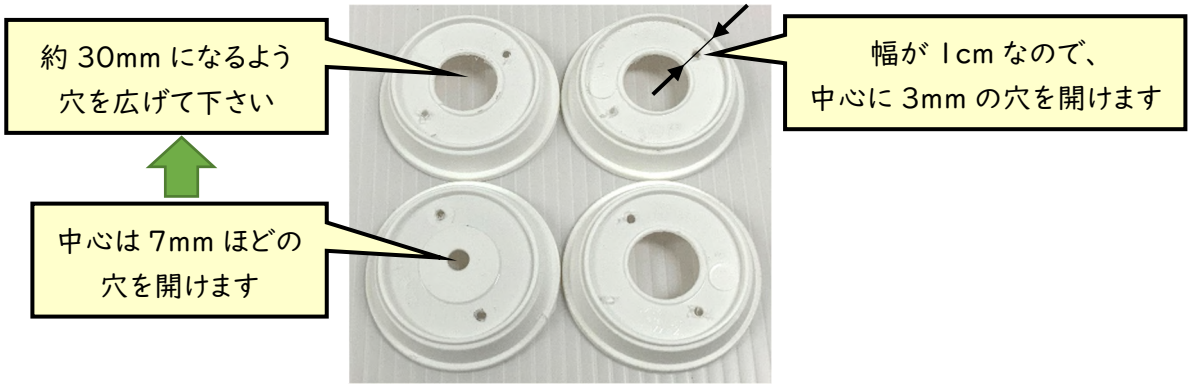
ハウス内の気温を測定するためには、センサに日射を遮るための日よけが必要になります。「ラジエーションシールド」で検索すると市販品が出てきますが、自分でも作成できます。作成方法はインターネット上でも手に入りますが、ここでは Wio Node と接続する温度センサ、温湿度センサを設置するための簡易なものを製作する方法を紹介します。

第1章で使用した
防水温度センサ
One Wire Temperature sensor

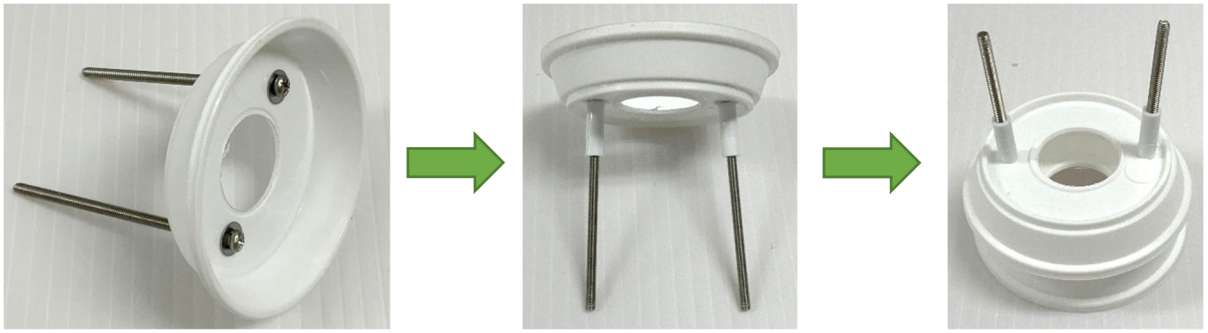
温湿度センサ
Grove 温湿度センサ（SHT31）
Grove 温湿度センサ（SHT35）

●必要な材料
・鉢皿*×4
・M3 ステンレスねじ 60mm
・M3 ステンレスナット×2（または×4）
・M3 ステンレスワッシャー×2
・M3 ステンレススプリングワッシャー×2（または×4）
・アイロンビーズ** 直径 5mm ×26 個
・テープ（養生テープやビニールテープ）
*大和プラスチック 鉢皿サルーン 1 号 φ65×H15 ホワイト 市場価格 30 円
**ウィルトンブランズ パーラービーズ 5001 単色 しろ 市場価格 200 円

本来はプラスペーサーを利用しますが、プラスペーサーは高価なため「アイロンビーズ」で代用しています。

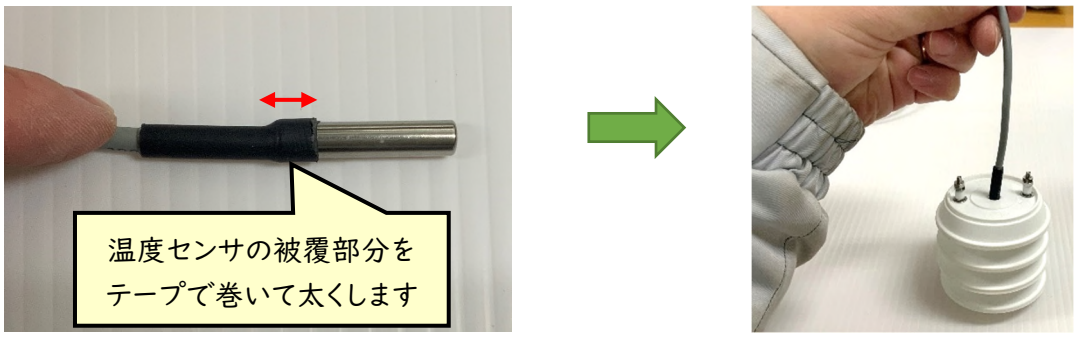


鉢皿を加工します。鉢皿全てに、温度センサを通すため穴を開けます。中心に 7mm ほどの穴を開けてください。1 枚はそのままにし、3枚はホールソーなどで穴を広げて約 30mm の穴にします。ホールソーが無い場合はニッパーで大まかに穴を広げ、調整はカッターナイフ等で行ってください。ステンレスねじを通す穴を開けるには 3mm のドリルで穴を開けます。4枚すべてに行ってください。

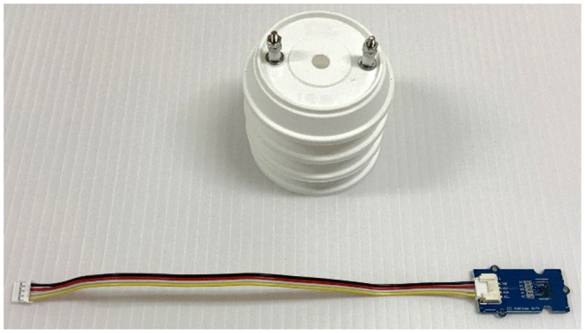


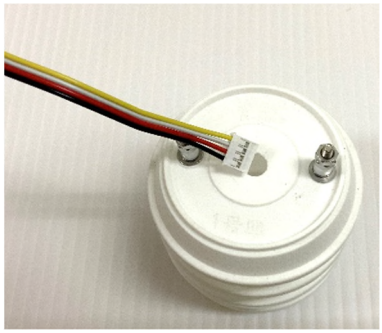
3cmの穴を開けた鉢皿の下からネジを通します。ネジにはスプリングワッシャー（無くてもかまいません）とワッシャーを入れてから鉢皿に通します。ネジにアイロンビーズ3つずつ通してスペーサーとします。その後鉢皿を通し、またアイロンビーズを3つずつ通してスペーサーとします。一番上の鉢皿には中心に 7mm の穴を開けたものが来るようにしてください。

最上段の鉢皿をつけたのち、ワッシャー、スプリングワッシャー（無ければ省略可）、アイロンビーズ、ナットの順で取り付け、ナットを回してしめつけていきます。スプリングワッシャーがある場合はスプリングワッシャーがつぶれるまでしめます。無い場合は右図のようにナット2から3個分取り付けられる程度にしめます。ナットが外れるのが心配な場合は右図のようにダブルナットにしてください。

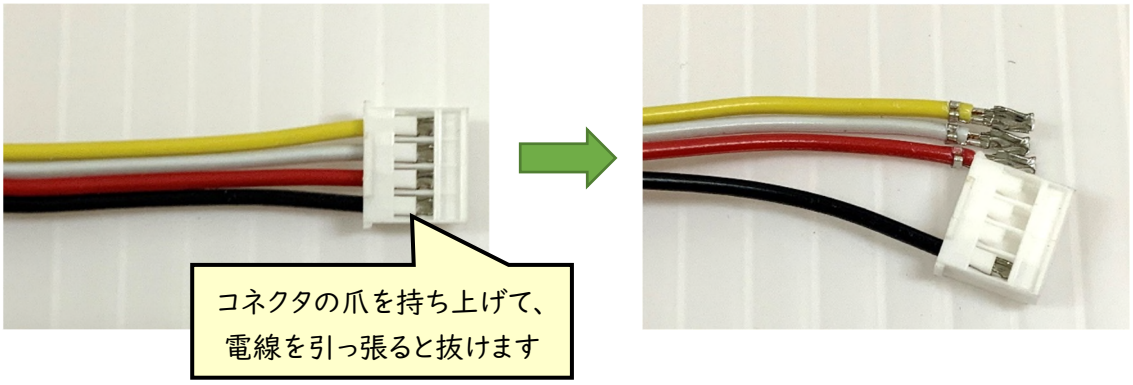


温度センサを穴に通した後、温度センサが抜けないようにします。温度センサの被覆部分を養生テープやビニールテープで 2 から 3 周分巻きます。右図のように持ち上げても外れなければ完成です。

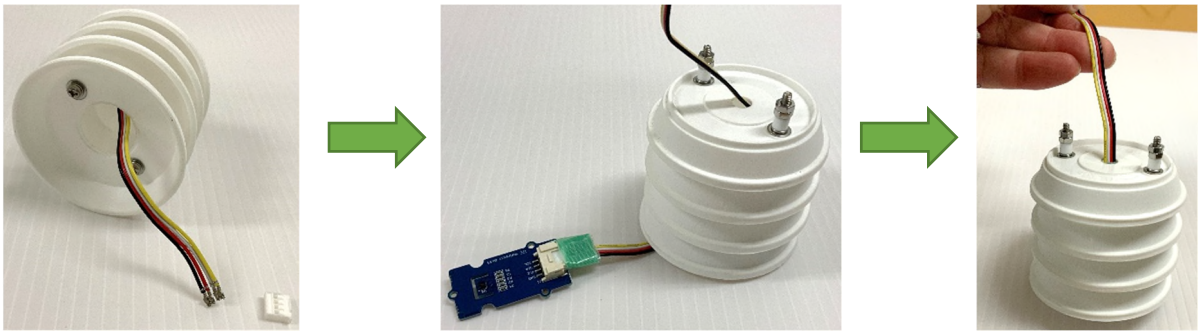




温湿度センサを使用する場合について説明します。温湿度センサはこのままの状態では、ケーブル部分を通すことができないため、コネクタを一度外す必要があります。



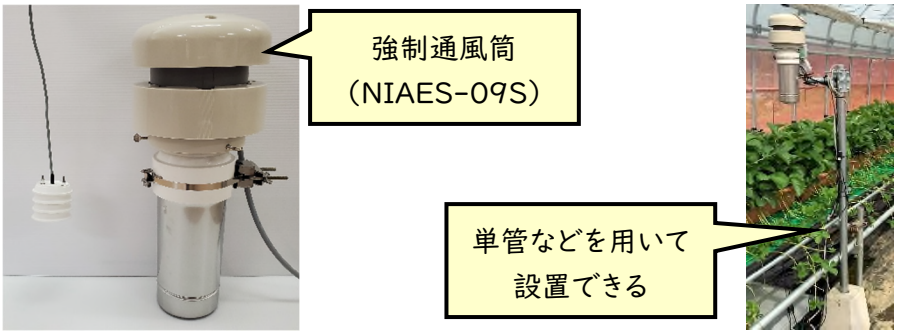
コネクタを外すには、爪を持ち上げる必要があります。カッターナイフでも行うことができますが、コネクタを破壊してしまう恐れがあるため、精密マイナスドライバなどで行ってください。



コネクタを外して、日よけに通した後はコネクタを再度取り付けて、温湿度センサを取り付けてください。その際、コネクタ部を養生テープ等で巻いておき、日よけから落ちないようにします。右図のように持ち上げても外れなければ完成です。

接続用のケーブルが短いため、長いケーブルを購入して使用することを考えるかもしれませんが、ここで紹介した温度センサ、温湿度センサは「I2C」と呼ばれる通信方式のセンサであるため、ケーブルを長くするだけでは通信ができなくなってしまいます。少々短いですが、付属してくる長さのケーブルを使用してください。

ここでは、簡易な日よけを作成する方法を紹介しましたが、強制通風筒と呼ばれるタイプのラジエーションシールドを利用すると、より精度よく測定することができます。強制通風筒も多くの自作方法がありますが、材料費 2 万円以下で作成できる「NIAES-09S」は、センサの設置方法が簡単で、作成や設置方法等の詳しい資料*が公開されています。



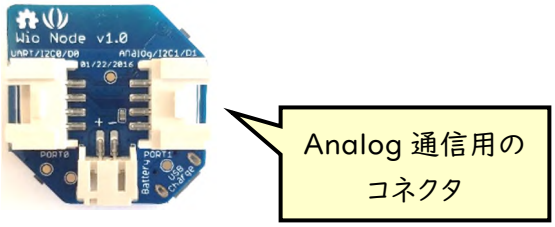
*農業気象学会 の HP でダウンロード可能です。
連載講座「栽培環境における気温の観測技法と利用」
（4）NIAES-09S 改型強制通風筒の製作法（福岡峰彦ら）
http://agrmet.jp/wordpress/wp-content/uploads/2019-A-2.pdf
連載講座「栽培環境における気温の観測技法と利用」
（5）観測用マストの建て方（福岡峰彦）
http://agrmet.jp/wordpress/wp-content/uploads/2019-A-3.pdf

【温度センサ使用の注意】

温度センサを利用する際には、これまで使用していた温度センサとの差を確認してから使用してください。なお、防水温度センサ（one wire temperature sensor ）や温度センサ（Grove 温湿度センサ（SHT31））の温度は±1℃の精度で測定できることを確認しておりますが、基板がむき出しのため、水をかけてしまったなどの原因で壊れたり、明らかにおかしい値が出るようになった場合は、新たにセンサを購入して交換してください。

温度センサの精度は重要です。ハウスで使用後、事務所などに回収し、しばらく使用しないで保管していたものを再度次年度に利用する際は、新しい温度センサと比較して問題なく使用できるかを確認して下さい。防水温度センサが複数ある場合は、バケツに水を入れてその中に全ての温度センサを同時に入れて、それぞれの温度センサの値の差を見ると確認が簡単です。

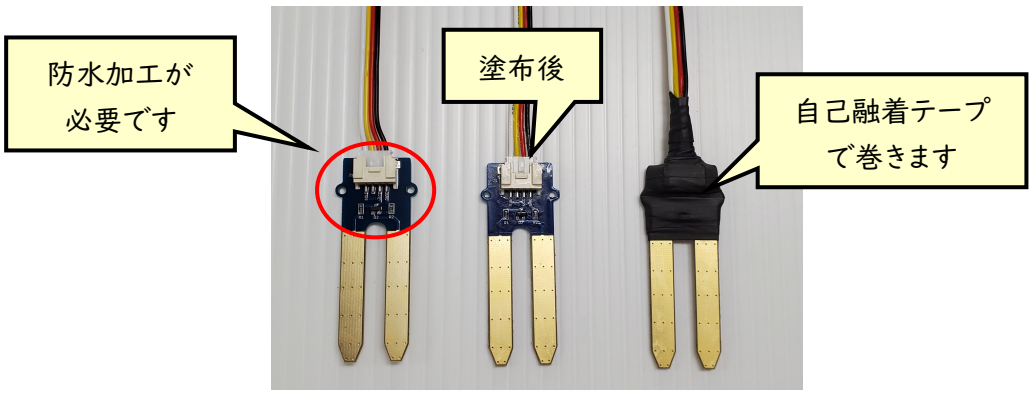
6. 土壌水分センサの設置と使用方法



Wio Node には 2 つコネクタがありますが、土壌水分センサ*に利用できるものは Analog と書かれたコネクタ側のみとなりますので注意してください。Analog での通信のため、ケーブルは延長可能です。右図の写真のような延長ケーブル**を使用できます。

*GROVE - 水分センサ 市場価格 約 500 円

**M5Stack 用 GROVE 互換ケーブル 200 cm（1 個入り） 市場価格 約 500 円



（1） 未処理
（2） ゴムスプレーを塗布
（3） 自己融着テープで防水

土壌水分センサは基板部分がむき出しのため、防水加工を施す必要があります。最も良い方法は、基板部分以外を養生テープでマスキングして、RTV シリコーンスプレーなどの基盤塗布用のゴムスプレーを塗布してから、自己融着テープを巻きます。RTV シリコーンスプレーが手に入らない場合は自己融着テープだけでもかまいません。

第６章 第５章までのまとめ-温湿度と土壌水分をハウスで測定する例-

本章ではこれまでのまとめとして、ハウスの温度（、湿度と地温、土壌水分）を測定するシステムを作成していきます。P.33 でダウンロードした「データ通知プログラム」、「グラフ通知プログラム」、「警報通知プログラム」を使用します。

〔本章で作成する遠隔監視システムの仕様〕
（1）. ハウスの温度、（湿度、地温、土壌水分）を 1 時間毎に定期通知する
（2）. ハウスの温度が 30℃を超えた時に警報通知する。
（3）. ハウスの温度について、前日の「平均温度」と「最高、最低温度」に加えて、
前日の 0 時から通知する日の 24 時までの範囲のグラフを 12 時に通知する
（4）. 測定したデータは 10 分毎に保存する。

1. Wio Node とセンサの準備（第 1 章を参照）

ハウスの温度、湿度、地温、土壌水分を測定します。Wio Node と各センサを準備します。
●必要な材料
・Wio Node × 2 個 … それぞれを「Wio Node 001」「Wio Node 002」とします。
・温湿度センサ×1個 … Grove 温湿度センサ（SHT31 または SHT35）
・防水温度センサ×1個 … One wire Temperature sensor
・土壌水分センサ×1個 … Grove 土壌水分センサ
※ 入手先は第 8 章に掲載しております。少なくとも温湿度センサか温度センサのどちらか 1 つあれば本章のシステムを作成できます。

第 1 章を参考に、Wio Node とセンサを設定して、アクセストークンを取得します。

必要な材料全てが揃っている場合

【Wio Node 001】
・温湿度センサ
温度 key : temperature
url01 = “アクセストークン1”
湿度 key : humidity
url02 = “アクセストークン2”

【Wio Node 002】
・温度センサ
地温 key : temperature
url03 = “アクセストークン3”
・土壌水分センサ
土壌水分 key : moisture
url04 = “アクセストークン4”

温度センサのみある場合

【Wio Node 001】
・温度センサ
温度 key : temperature
url01 = “アクセストークン1”

Wio Node に
番号を書くと便利です

2. 通知する LINE グループを準備する

以下の3つのグループを作り、トークンを発行します。

(1) 001 ハウス情報（定期通知）　… 1 時間毎に通知する
(2) 002 ハウス情報（日報・グラフ通知）… 毎日 12 時に平均温度やグラフを通知する
(3) 003 ハウス情報（警報通知）　… ハウス温度が 30℃を超えたら通知する

3. プログラムの準備を行う

以下の 4 つのプログラムを作成します。

(1) Kayoi_data10.py　… 10 分間隔のデータ保存用。通知は行わない。

(2) Kayoi_data60.py　… 60 分（1 時間）間隔で「001 ハウス情報（定期通知）」へ通知する。

(3) Kayoi_daily_report.py … (1) で保存した 10 分間隔データ(data10.txt)から平均温度と最高、最低温度に加えてグラフを作成し、「002 ハウス情報（日報・グラフ通知）」へ通知する。

(4) Kayoi_alert.py　… 5 分毎に温度をチェックし、ハウス温度が 30℃を超えた際に「003 ハウス情報（警報通知）」へ通知する。

※データを1時間毎ではなく、10 分毎に通知する場合は (2) は必要ありません。また、警報通知を必要としない場合は (4) は必要ありません。

p.33 でダウンロードした「Kayoi_data10.py」は、これを元に「Kayoi_data60.py」を作成するため、事前にコピーを作ってください。次に「program3.py」をコピーして「Kayoi_alert.py」とファイル名を変更します。

4.「データ通知プログラム」を作成する

Kayoi_data10.py を開いて、必要な箇所を修正します。以下のプログラムの灰色の網掛け部分を適宜修正します。1 行に収まらない場合などは一部表記を省略、改変しています。

10,13,16,18 行目 対応するアクセストークンに変更
119 行目 通知する LINE グループのアクセストークンに変更

温度センサ1つで使用する場合などは p.75 から解説します。

```
1  # coding:utf-8
2  import requests #Web API にアクセスするために用いる
3  import time #時刻の取得に用いる
4  import os #CSV ファイルの保存に用いる
5  from decimal import Decimal, ROUND_HALF_UP #四捨五入に用いる
6
7  #スマートフォンで設定した wio node のトークンを入れる
8  #【Wio Node 001】
9  #  温湿度センサ-温度  key:temperature
10 url01 = "アクセストークン1"
11 #"""
12 #  温湿度センサ-湿度  key:humidity
13 url02 = "アクセストークン2"
14 #【Wio Node 002】
15 #  温度センサ-地温  key:temperature
16 url03 = "アクセストークン3"
17 #  土壌水分センサ-土壌水分  key:moisture
18 url04 = "アクセストークン4"
19 #"""
20
21 #request 関数と for 関数を使って最大 3 回  5 秒おきにデータを取得する
22 CONNECTION_RETRY = 3 #データ取得を行う最大試行数を設定する
23 INTERVAL_TIME = 5 #データ取得失敗時に何秒待つかを設定する
24
25 #data1（json1 の後には設定したセンサにあわせて「key キー」を記入する）
26 #1<= X < 4  なので  3  回試行する
27 for i1 in range(1, CONNECTION_RETRY+1):
28     json1 = requests.get(url01).json()
29     if ("error" in json1) :
30         if("Node is offline" in json1["error"]):
31             data1 = ' OFF'#マイコンか WiFi の電源がオフ
32             break
```

```
33            elif("timeout" in json1["error"]):
34                data1 = 'TIME'#マイコン再起動中か WiFi の調子が悪いです
35                time.sleep(INTERVAL_TIME)
36            elif("METHOD" in json1["error"]):
37                data1 = 'FWUP'#Wio アプリのセンサが正しいかを確認してください
38                break
39            elif("Unknown" in json1["error"]):
40                data1 = ' UNK'#センサが壊れているか、センサが抜けています
41                time.sleep(INTERVAL_TIME)
42            else:
43                data1 = 'FAIL'#データ取得に失敗しました。
44                time.sleep(INTERVAL_TIME)
45        else:
46            data1 = (Decimal(json1["temperature"]).quantize(Decimal('0.1')
              ,rounding=ROUND_HALF_UP))#データを四捨五入します
47            break
48
49    #"""
50    for i2 in range(1, CONNECTION_RETRY+1):
51        json2 = requests.get(url02).json()
52        if ("error" in json2) :
53            if("Node is offline" in json2["error"]):
54                data2 = ' OFF'#略
55                break
56            elif("timeout" in json2["error"]):
57                data2 = 'TIME'#略
58                time.sleep(INTERVAL_TIME)
59            elif("METHOD" in json2["error"]):
60                data2 = 'FWUP'#略
61                break
62            elif("Unknown" in json2["error"]):
63                data2 = ' UNK'#略
64                time.sleep(INTERVAL_TIME)
65            else:
66                data2 = 'FAIL'#略
67                time.sleep(INTERVAL_TIME)
68        else:
69            data2 = (Decimal(json2["humidity"]).quantize(Decimal('0.1')
              ,rounding=ROUND_HALF_UP))#データを四捨五入します
70            break
71
```

```
72  for i3 in range(1, CONNECTION_RETRY+1):
73      json3 = requests.get(url03).json()
74      if ("error" in json3) :
75          if("Node is offline" in json3["error"]):
76              data3 = ' OFF'#略
77              break
78          elif("timeout" in json3["error"]):
79              data3 = 'TIME'#略
80              time.sleep(INTERVAL_TIME)
81          elif("METHOD" in json3["error"]):
82              data3 = 'FWUP'#略
83              break
84          elif("Unknown" in json3["error"]):
85              data3 = ' UNK'#略
86              time.sleep(INTERVAL_TIME)
87          else:
88              data3 = 'FAIL'#略
89              time.sleep(INTERVAL_TIME)
90      else:
91          data3=(Decimal(json3["temperature"]).quantize(Decimal('0.1')
            ,rounding=ROUND_HALF_UP))#データを四捨五入します
92          break
93
94  for i4 in range(1, CONNECTION_RETRY+1):
95      json4 = requests.get(url04).json()
96      if ("error" in json4) :
97          if("Node is offline" in json4["error"]):
98              data4 = ' OFF'#略
99              break
100         elif("timeout" in json4["error"]):
101             data4 = 'TIME'#略
102             time.sleep(INTERVAL_TIME)
103         elif("METHOD" in json4["error"]):
104             data4 = 'FWUP'#略
105             break
106         elif("Unknown" in json4["error"]):
107             data4 = ' UNK'#略
108             time.sleep(INTERVAL_TIME)
109         else:
110             data4 = 'FAIL'#略
111             time.sleep(INTERVAL_TIME)
```

```
112         else:
113             data4=(Decimal(json4["moisture"]).quantize(Decimal('1')
                ,rounding=ROUND_HALF_UP))#データを四捨五入します
114             break
115 #"""
116 
117 #LINE notify の URL
118 url99 = "https://notify-api.line.me/api/notify"
119 token = '(1)001 ハウス情報(定期通知)のトークン' #LINE notify のトークン
120 
121 #"""
122 #LINE に通知するメッセージを作る
123 message  = '\n '
124 #'センサ名:'+str(data 番号)+'単位'+'\n' センサ名、データ番号、単位を書く
125 message += '■ハウス環境'+'\n'
126 message += '  温度:'+str(data1)+' ℃'+'\n'
127 #"""
128 message += '  湿度:'+str(data2)+' %RH'+'\n'
129 message += '  地温:'+str(data3)+' ℃'+'\n'
130 message += '土壌水分:'+str(data4)+''
131 #"""
132 
133 #LINE グループに通知を行う
134 payload = {'message' : message}
135 headers = {'Authorization' : 'Bearer '+ token}
136 r = requests.post(url99, data=payload, headers=headers)
137 #"""
138 
139 #CSV で保存する
140 #CSV の保存のために時刻を取得
141 timestamp = 'date +%F" %H:%M"'
142 current_time = os.popen(timestamp).readline().strip()
143 #CSV で保存するデータを組み合わせる
144 data_set  = str(current_time)
145 data_set += ',' + str(data1)
146 #"""
147 data_set += ',' + str(data2)
148 data_set += ',' + str(data3)
149 data_set += ',' + str(data4)
150 #"""
151 data_set += '\n'
```

```
152
153 #/home/pi/data.txt というファイルにデータを保存する
154 fout = open('/home/pi/data10.txt','at')
155 fout.write(data_set)
156 fout.close()
157
```

現在の設定ですと、
「10 分間隔のデータを保存し、通知を行う」というプログラムになっています。」
p.70 で説明した以下のプログラムを完成するには、通知を行わないようにする必要があります。
(1)Kayoi_data10.py　　… 10 分間隔のデータ保存用。通知は行わない。
p.69 で用意したセンサが「温度センサ」だけだった場合なども含めて、次ページの修正チャートを確認しながら、以下のパターンに応じて修正を行ってください。

【パターン1:10 分間隔のデータを保存し、通知を行う」】
そのままで修正なし。通知間隔が 10 分で問題ない場合はこれで終了です。

【パターン 2:10 分間隔のデータを保存し、通知は行わない】
目的の設定です。以下の作業を行って、(1)Kayoi_data10.py を完成してください。
121 行目 「#"""」から 「#」を削除し、「"""」に変更します。
127 行目 「#"""」を削除します。
131 行目 「#"""」を削除します。
これで(1)Kayoi_data10.py は完成です。おつかれさまでした。
※ 「#」は後ろの文字列がコメントになる記号です。
「"""」は次の行からコメントになる記号です。コメント行を終わらせるには、
「"""」を再度書くことでコメント行を終わらせることができます。

【パターン 3:温度センサが1つだけの場合で、通知を行う】
11 行目 「#"""」から 「#」を削除し、「"""」に変更します。
49 行目 「#"""」から 「#」を削除し、「"""」に変更します。
127 行目 「#"""」から 「#」を削除し、「"""」に変更します。
146 行目 「#"""」から 「#」を削除し、「"""」に変更します。

【パターン 4:温度センサが1つだけの場合で、通知を行わない】
11 行目 「#"""」から 「#」を削除し、「"""」に変更します。
49 行目 「#"""」から 「#」を削除し、「"""」に変更します。
121 行目 「#"""」から 「#」を削除し、「"""」に変更します。
127 行目 「#"""」を削除します。
131 行目 「#"""」を削除します。
146 行目 「#"""」から 「#」を削除し、「"""」に変更します。

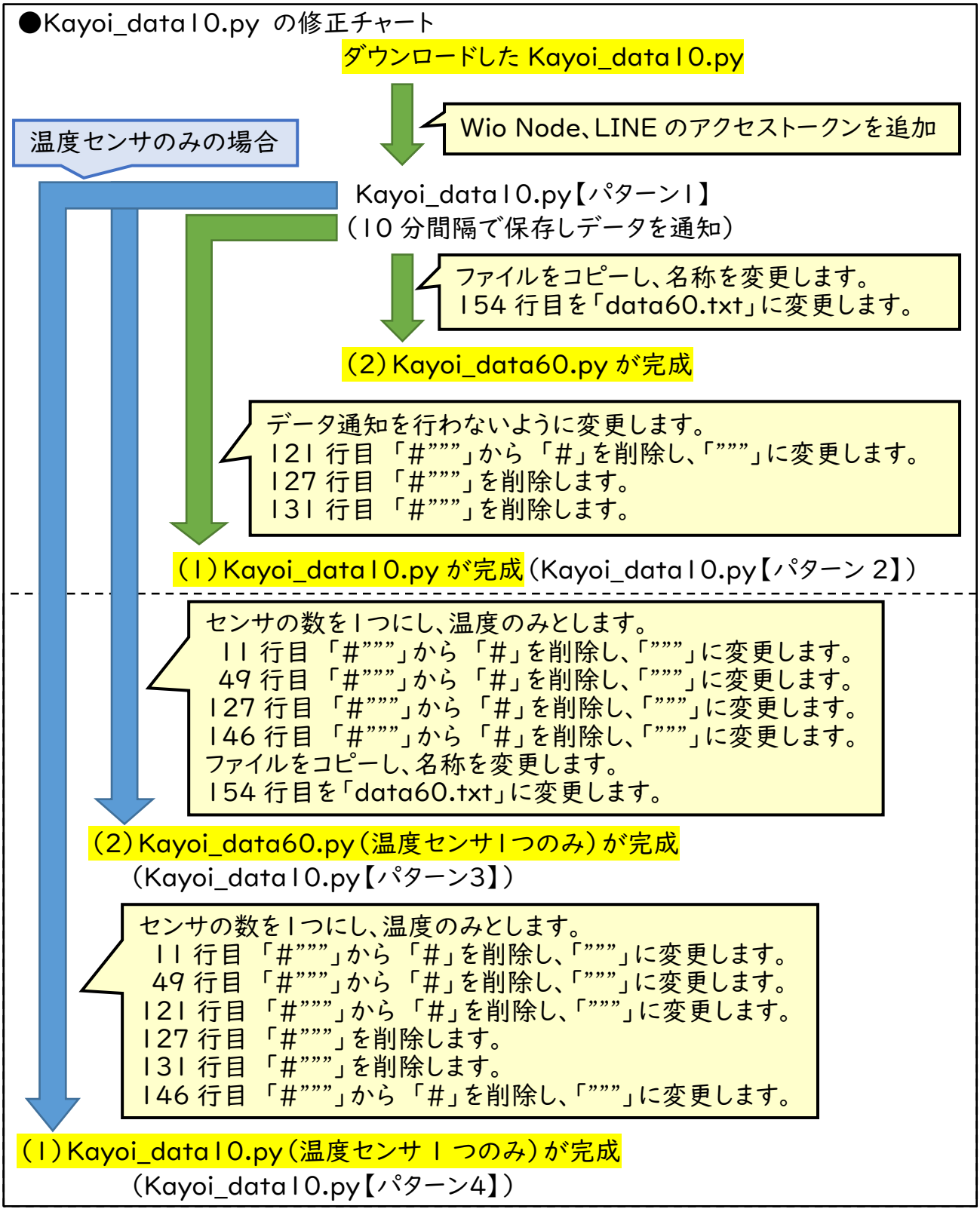
(2) Kayoi_data60.py…60 分（1 時間）間隔で「001 ハウス情報（定期通知）」に通知を作成します。

前ページのパターン1を修正します。

154 行目 fout = open('/home/pi/data10.txt','at') の

「data10.txt」を「data60.txt」に変更

Kayoi_data10.py をコピーし、ファイル名を「Kayoi_data60.py」に変更し完成。



次にプログラムを活用する上で役に立つ事項を2つ説明します。

●エラー原因の通知（「プログラム③」の応用部分）

以下のプログラムはプログラム③のエラー時に再実行するプログラムを、現場で Wio Node 等のメンテナンスをしやすいように変更したものです。データを取得時に「エラー」となった場合、そのエラーの原因から通知するメッセージを変えるように変更したものです。

```
#data l
for i l in range( l , CONNECTION_RETRY+ l ):
    json l = requests.get(url0 l ).json()
    if ("error" in json l ) :
        data l = 'エラー'
        print('data l error!')
        time.sleep(INTERVAL_TIME)
    else:
        data l = json l ["temperature"]
        break
```

データ取得時に
key が"error"の際、
返される値に応じて、
メッセージを変えています。

```
#data l（json l の後には設定したセンサにあわせて「key キー」を記入する）
for i l in range( l , CONNECTION_RETRY+ l ):
    json l = requests.get(url0 l ).json()
    if ("error" in json l ) :
        if("Node is offline" in json l ["error"]):
            data l = ' OFF'#マイコンか WiFi の電源がオフ
            break
        elif("timeout" in json l ["error"]):
            data l = 'TIME'#マイコン再起動中か WiFi の調子が悪いです
            time.sleep(INTERVAL_TIME)
        elif("METHOD" in json l ["error"]):
            data l = 'FWUP'#Wio アプリのセンサが正しいかを確認してください
            break
        elif("Unknown" in json l ["error"]):
            data l = ' UNK'#センサが壊れているか、センサが抜けています
            time.sleep(INTERVAL_TIME)
        else:
            data l = 'FAIL'#データ取得に失敗しました
            time.sleep(INTERVAL_TIME)
    else:
        data l = (Decimal(json l ["temperature"]).quantize(Decimal('0. l ')
        ,rounding=ROUND_HALF_UP))#データを四捨五入します
        break
```

「Wio Node の電源がオフになっている場合」や、「Wio Node のスマートフォンアプリ上で設定したセンサと異なるセンサが接続されている、あるいはセンサが抜けている場合」は、原因が明らかなため、再実行する必要がありません。そのため、この2つがエラー原因だった場合は再実行しないようにしています。しかし、状況によってはセンサが接続されているのに FWUP と通知される場合があります。その場合は以下のプログラムのうち、break を削除すると、この場合でもデータ取得エラー時でも再実行するようになります。

```
        elif("METHOD" in json_l["error"]):
            data_l = 'FWUP'#Wio アプリのセンサが正しいかを確認してください
            break
```

これらのエラーメッセージが必要無い場合は、Kayoi_data_10.py の該当部分を、プログラム③の該当部分に書き換えて使用してください。

●データを四捨五入する

データを取得した際、温度センサは小数第2位までの値が表示されることがあります。しかし、実際には小数第2位まで必要な場合は少ないことと、通知時に数字の数がバラバラだと判断しにくいので、データを四捨五入して表示しています。

たとえば、22℃前後の値の際、以下の2つのパターンのように表示されると読みにくいので、小数第1位までの値に変更します。

22.3℃
22.24℃

22.3℃
22.2℃

```
                                    1 にする⇒小数第1位を四捨五入して表示
                                    0.1 にする⇒小数第 2 位を四捨五入して表示
                                    0.01 にする⇒小数第 3 位を四捨五入して表示
data_l = (Decimal(json_l["temperature"]).quantize(Decimal('0.1')
,rounding=ROUND_HALF_UP))#データを四捨五入します
```

※実際のプログラム上では2行にせず、以下のように1行になります。

```
data_l = (Decimal(json_l["temperature"]).quantize(Decimal('0.1'),rounding=ROUND_HALF_UP))#データを四捨五入します
```

●その他

このプログラムでは 4 つのセンサまではこのプログラムでできます。Key や四捨五入の設定、メッセージ通知時の単位を修正して使用してください。また、センサを増やす場合は p.37、p.43 も参考に修正してください。

5.「グラフ通知プログラム」を作成する

Kayoi_daily_report.py を開いて、必要な箇所を修正します。プログラムの灰色の網掛け部分を修正します。1 行に収まらない場合などは一部表記を省略、改変しています。

147 行目　通知する LINE グループのアクセストークンに変更
（(2)002 ハウス情報（日報・グラフ通知）用のアクセストークン)

温度センサ 1 つで使用する場合などは p.85 から解説します。

```
1  # coding: utf-8
2  import pandas as pd #pandas をインポートする
3  import matplotlib.pyplot as plt #グラフを描画するのに用いる
4  import matplotlib as mpl #フォント名を指定するためにインポート
5  from datetime import date,timedelta #日付を扱うための標準ライブラリ
6  import requests
7  import datetime
8  from decimal import Decimal, ROUND_HALF_UP #四捨五入に用いる
9
10 #  読み込む csv ファイル名（絶対パスを推奨）
11 csv_file = "/home/pi/data10.txt"
12
13 # data10.txt にはヘッダーが無いので、列に名前をつける
14 col01 = "温度"
15 #"""
16 col02 = "湿度"
17 col03 = "地温"
18 col04 = "土壌水分"
19 #"""
20 colums_name = ["datetime"]
21 colums_name.append(col01)
22 #"""
23 colums_name.append(col02)
24 colums_name.append(col03)
25 colums_name.append(col04)
26 #"""
27
28 # csv_file をデータフレーム型で読み込む
29 df = pd.read_csv(csv_file, names = (colums_name),  index_col =
   "datetime", parse_dates = True, encoding = "UTF8")
30
31 #  データが取れなかった場合の警告があるところを NaN にする
```

```
df = df.replace(' OFF', 'NaN')
df = df.replace('TIME', 'NaN')
df = df.replace('FWUP', 'NaN')
df = df.replace(' UNK', 'NaN')
df = df.replace('FAIL', 'NaN')
# NaN を含む行を除外する
df = df.dropna(how = "any")
#  データが文字列になっているので float に変更
df = df.astype(float)

#  基準日を算出する（通常はプログラム実行日）
d = datetime.date.today()
#  日時を指定する方法
#d = date(2020,12,30)

# 【基準日を含めた合計 48 時間を出力する】
d1 = d + timedelta(days = -1)#  基準日の前日を算出する
d2 = d + timedelta(days = +1)#  基準日の翌日を算出する

#matplotlib のフォントを日本語にする
mpl.rcParams['font.family'] = "Noto Sans CJK JP"

#全ての項目をグラフ化&画像として保存
#グラフ1：ハウス温度（1 つのグラフで）
df.plot.line( y = ["温度"], subplots = False, grid = True, figsize=(8,8), xlim = [d1,d2], ylim = [0,40], colormap = "Set1")
plt.savefig("graph1.png")

#"""
#グラフ2：ハウス湿度（1 つのグラフで）
df.plot.line( y = ["湿度"], subplots = False, grid = True, figsize=(8,8), xlim = [d1,d2], ylim = [0,100], colormap = "Set1")
plt.savefig("graph2.png")

#グラフ3：地温（1 つのグラフで）
df.plot.line( y = ["地温"], subplots = False, grid = True, figsize=(8,8), xlim = [d1,d2], ylim = [0,40], colormap = "Set1")
plt.savefig("graph3.png")

#グラフ4：土壌水分（1 つのグラフで）
df.plot.line( y = ["土壌水分"], subplots = False, grid = True, figsize=(8,8),
```

```
    xlim = [d1,d2], ylim = [0,1023], colormap = "Set1")
70  plt.savefig("graph4.png")
71
72  #グラフ 5:ハウス温度、ハウス湿度、地温、土壌水分(それぞれのグラフを 1 枚で)
73  df.plot.line( y = ["温度","湿度","地温","土壌水分"], subplots = True, grid =
    True, figsize=(8,8), xlim = [d1,d2], colormap = "Set1")
74  plt.savefig("graph5.png")
75
76  #グラフ 6:ハウス温度、地温(1つのグラフで)
77  df.plot.line( y = ["温度","地温"], subplots = False, grid = True,
    figsize=(8,8), xlim = [d1,d2], ylim = [0,40], colormap = "Set1")
78  plt.savefig("graph6.png")
79  #"""
80
81  # DatetimeIndex の秒を切り捨て floor
82  print(df.index)
83  df.index = df.index.floor("min")
84  print(df.index)
85
86  # 【統計量を計算する(通知日から基準日までの区間)】
87  d_td = d
88  d_ld = d_td + timedelta(days = -1) # 基準日(通知日 1 日前とする)
89
90  # df.index の中身は timestamp 型であり、通知日で設定した date 型との比較では
    怒られるので、timestamp 型に変換する
91  d_td = pd.to_datetime(d_td)
92  d_ld = pd.to_datetime(d_ld)
93
94  #【統計量を計算する】
95  #平均、最高、最低を算出する。日平均気温は 1 時~24 時までの毎正時の平均値なの
    で、60min データを df3 とする
96  #24 時(00:00:00)のデータを計算に入れるために全体を 10 分前にずらす(8/10
    00:00:00 -> 8/9 23:50:00)
97  df2 = df
98  df3 = df
99  df2= df[(df.index >= d_ld) & (df.index <= d_td)]
100 df2.index = df2.index + timedelta(minutes = -10)
101
102 # 指定した範囲で df を抽出(置換)
103 df2= df2[(df2.index >= d_ld) & (df2.index <= d_td)]#10min ずらしたことで
    出てくる前後+1 日を削除
```

```
# 10 分データから最大値・最小値を算出する
df_max = df2.resample("1D").max()
df_min = df2.resample("1D").min()

#【平均値を算出するために 60min データを作成する】
df3 = df[(df.index >= d_ld) & (df.index <= d_td)]
df3 = df3.asfreq('1H') #60min データを作成
#  全体を 1 時間前にずらす（例:8/10 00:00:00 -> 8/9 23:00:00）
df3.index = df3.index + timedelta(hours = -1)
#  指定した範囲で df を抽出（置換）
df3 = df3[(df3.index >= d_ld) & (df3.index <= d_td)]

print(df3)

# 60 分データから日平均値を算出する
df_mean = df3.resample("1D").mean()

print(df_mean)

#  全データ（np_mean[日付 index, データ項目 column]の2次元配列）
np_mean = df_mean.values
np_max = df_max.values
np_min = df_min.values

print(np_mean)

#  前日のみ抽出（インデックス-1 は最後の意味）
np_mean_yd = np_mean[-1]
np_max_yd = np_max[-1]
np_min_yd = np_min[-1]

#  変数に入れる(col1 を指定したい。0 から始まるので、1 列目のハウス内温度 1 は 0,2 列目の･･･は 1 のようになる）
#ハウスの温度
house_temp_mean_1 = Decimal(np_mean[-1, 0]).quantize(Decimal('0.1'), rounding = ROUND_HALF_UP)
house_temp_max_1 = np_max[-1, 0]
house_temp_min_1 = np_min[-1, 0]

#  日付を index から取得する
index = df_mean.index.date
```

```
143 index_str = df_mean.index.strftime('%Y-%m-%d')
144
145 # LINE notify の URL
146 url99 = "https://notify-api.line.me/api/notify"
147 token = '(2)002 ハウス情報(日報・グラフ通知)用のアクセストークン'
148
149 # LINE グループに通知を行う(1枚目の画像を送る)
150 message = '\n'
151 message+= "■ハウス温度" + '\n'
152 message+= str(index[-1]) + '\n'
153 message+= '平均温度' + str(house_temp_mean_1) + " ℃" + '\n'
154 message+= '最高温度' + str(house_temp_max_1) + " ℃" + '\n'
155 message+= '最低温度' + str(house_temp_min_1) + " ℃"
156
157 payload = {'message' : message}
158 headers = {'Authorization' : 'Bearer '+ token}
159 files = {"imageFile":open("graph1.png","rb")}
160 r = requests.post(url99, data=payload, headers=headers, files = files)
161
162 #"""
163 # LINE グループに通知を行う(2枚目の画像を送る)
164 message = '\n'
165 message+= "■ハウス湿度" + '\n'
166 message+= str(index[-1])
167
168 payload = {'message' : message}
169 headers = {'Authorization' : 'Bearer '+ token}
170 files = {"imageFile":open("graph2.png","rb")}
171 r = requests.post(url99, data=payload, headers=headers, files = files)
172
173 # LINE グループに通知を行う(3枚目の画像を送る)
174 message = '\n'
175 message+= "■地温" +'\n'
176 message+= str(index[-1])
177
178 payload = {'message' : message}
179 headers = {'Authorization' : 'Bearer '+ token}
180 files = {"imageFile":open("graph3.png","rb")}
181 r = requests.post(url99, data=payload, headers=headers, files = files)
182
183 # LINE グループに通知を行う(4枚目の画像を送る)
```

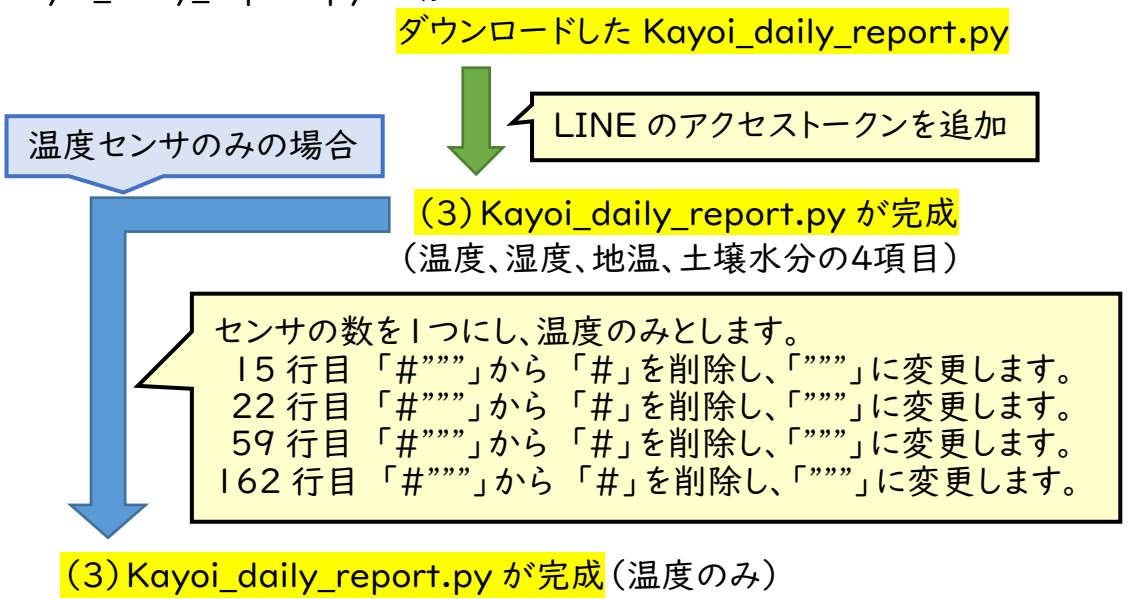
```
message = '\n'
message+= "■土壌水分" + '\n'
message+= str(index[-1])

payload = {'message' : message}
headers = {'Authorization' : 'Bearer '+ token}
files = {"imageFile":open("graph4.png","rb")}
r = requests.post(url99, data=payload, headers=headers, files = files)

# LINE グループに通知を行う(5枚目の画像を送る)
message = '\n'
message+= "ハウス情報まとめ" + '\n'
message+= str(index[-1])

payload = {'message' : message}
headers = {'Authorization' : 'Bearer '+ token}
files = {"imageFile":open("graph5.png","rb")}
r = requests.post(url99, data=payload, headers=headers, files = files)

# LINE グループに通知を行う(6枚目の画像を送る)
message = '\n'
message+= "ハウス温度と地温" + '\n'
message+= str(index[-1])

payload = {'message' : message}
headers = {'Authorization' : 'Bearer '+ token}
files = {"imageFile":open("graph6.png","rb")}
r = requests.post(url99, data=payload, headers=headers, files = files)
#"""

```

これで、(1)Kayoi_data10.py を実行することで作られる data10.txt から、平均温度や最高、最低温度に加えてグラフを通知するプログラム(3)Kayoi_daily_report.py が作成できました。p.69で用意したセンサが「温度センサ」だけだった場合は、次ページの修正チャートを確認しながら、以下のパターンに応じて修正を行ってください。

【パターン1:必要なセンサが全てそろっている場合】
そのままで修正なし。(3)Kayoi_daily_report.py は完成です。

【パターン2:温度センサ1つの場合】
温度センサ以外の部分を全てコメントに変えます
15 行目 「#"""」から 「#」を削除し、「"""」に変更します。
22 行目 「#"""」から 「#」を削除し、「"""」に変更します。
59 行目 「#"""」から 「#」を削除し、「"""」に変更します。
162 行目 「#"""」から 「#」を削除し、「"""」に変更します。

●Kayoi_daily_report.py の修正チャート



次にプログラムを活用する上で役に立つ事項を説明します。

●グラフの設定を変更する
温度のグラフを通知している、55-57 行目を例に説明します。

```
df.plot.line( y = ["温度"], subplots = False, grid = True, figsize=(8,8),
xlim = [d1,d2], ylim = [0,40], colormap = "Set1")
plt.savefig("graph1.png")
```

イ. 表示するデータを選択する
14-18 行目で設定した、それぞれのデータ名を、y=["データ名"]の中に書き込みます。
今回は「温度」なので、y=["温度"]としています。たとえば、温度と地温を1つのグラフに表示する場合は、y=["温度","地温"]とします。

ロ. グラフの表示方法を変える
複数の項目 (たとえば、温度と地温など) を表示する際に、1つのグラフの中に表示するか、X 軸 (時間軸) を同じにして複数のグラフを並べて表示するかを変更できます。
subplots = False … 1 つのグラフで表示する
subplots = True … 複数のグラフを並べて表示する

ハ. Y 軸の範囲を変更する

今回は「ylim ＝ [0,40]」と表示されています。これは Y 軸を 0 から 40 の間に設定していることとなります。たとえば、冬の間にハウス内で果樹などの永年性作物を越冬させたい場合は、氷点下になることもあります。その場合は「ylim ＝ [-10,30]」とすると、Y 軸を-10 から 30 の間に設定することができます。

土壌水分の場合は、ylim ＝ [0,1023]としていますが、これは土壌水分センサが水分に応じて 0 から 1023 の間に値を通知するからです。

ニ. グラフの線の色を変更したい

「colormap ＝ "Set1"」を変更することで、線の色を変更することができます。

Set1,Set2,Set3,tab10 などがありますので、変更して見て下さい。また、以下の URL に選択できるカラーマップについて詳しく書いてあります。英語で書かれていますが、カラーチャートとともに表示されているので、いくつか試してみることをお勧めします。

https://matplotlib.org/3.3.4/tutorials/colors/colormaps.html

ホ. 通知したグラフの画像を保存したい

plt.savefig("graph1.png")と書いてありますが、graph1.png がグラフの画像です。これは「グラフ通知プログラム」を実行したフォルダに保存されます。今回は📁homeフォルダ内の📁pi フォルダの📁program フォルダ内に「グラフ通知プログラム」である「kayoi_daily_report.py」を置いているため、program フォルダ内にグラフ画像は保存されます。

●平均温度の値をより正確に計算したい

Kayoi_data10.py では、プログラムの後半（141-142 行目）に時刻を取得し、データを保存しています。そのため、crontab で指定した実行時刻よりも少し遅れることがあり、遅れた場合の値は除外されたまま計算されてしまいます。そのため、プログラムの実行と同時に時刻を取得するように変更（たとえば 141-142 行目の内容を 6 行目に変更）することで、より正確に平均温度を計算することができます。

6.「警報通知プログラム」を作成する

program2.py を用いて、(4) Kayoi_alert.py を作成します。P.39 で説明した内容を参考に修正します。まず、program3.py を Kayoi_alert.py に名前を変更します。

5行目のアクセストークンを「温度センサ用のアクセストークン」に変更
16 行目は通知する「(3) 003 ハウス情報 (警報通知)」のトークンに変更
29 行目の設定温度を 30.0 に変更し、32、33 行目に「#」をつけコメントに変更

```
1  # coding:utf-8
2  import requests
3
4  #スマートフォンで設定した wio node のトークンを入れる
5  url01 = "アクセストークン1"
6
7  #request 関数を使ってデータを取得する
8  json1 = requests.get(url01).json()
9
10 #"temperature"のデータが取れたときに data に温度を入れる
11 data1 = json1["temperature"]
12
13 #LINE notify の URL
14 #LINE notify のトークン(通知先に応じて変更すること)
15 url99 = "https://notify-api.line.me/api/notify"
16 token = '通知する LINE グループのアクセストークン'
17
18 #高温用メッセージ
19 message1 = '高温注意!:'+str(data1)+'℃'
20 payload1 = {'message' : message1}
21
22 #低温用メッセージ
23 message2 = '低温注意!:'+str(data1)+'℃'
24 payload2 = {'message' : message2}
25
26 headers = {'Authorization' : 'Bearer '+ token,}
27
28 #警報を出したいときの温度を設定する
29 if data1 > 20.0:#高温用:この温度より「高い」と通知する
30     r = requests.post(url99,data=payload1,headers=headers)
31
32 if data1 < 10.0:#低温用:この温度より「低い」と通知する
33     r = requests.post(url99,data=payload2,headers=headers)
34
```

これで（4）Kayoi_alert.py は完成ですが、データ取得時にエラーだった場合に再実行するように修正する場合は、7 行目から 11 行目を削除し、Program3.pyの 14-24 行目をコピー＆ペーストしてください。

```
#request 関数を使ってデータを取得する
json1 = requests.get(url01).json()

#"temperature"のデータが取れたときに data に温度を入れる
data1 = json1["temperature"]
```

データ取得時にエラーであった場合、
再実行するように変更

```
#data1
#1<=X<4 なので 1 2 3 回試行する
for i1 in range(1, CONNECTION_RETRY+1):
    json1 = requests.get(url01).json()
    if ("error" in json1) :
        data1 = 'エラー'
#         print('data 1 error!')
        time.sleep(INTERVAL_TIME)
    else:
        data1 = json1["temperature"]
        break
```

7. プログラムを定期実行できるように crontab を設定する

p.70 で取り決めた設定で各プログラムを実行するようにします。

●LINEグループの設定

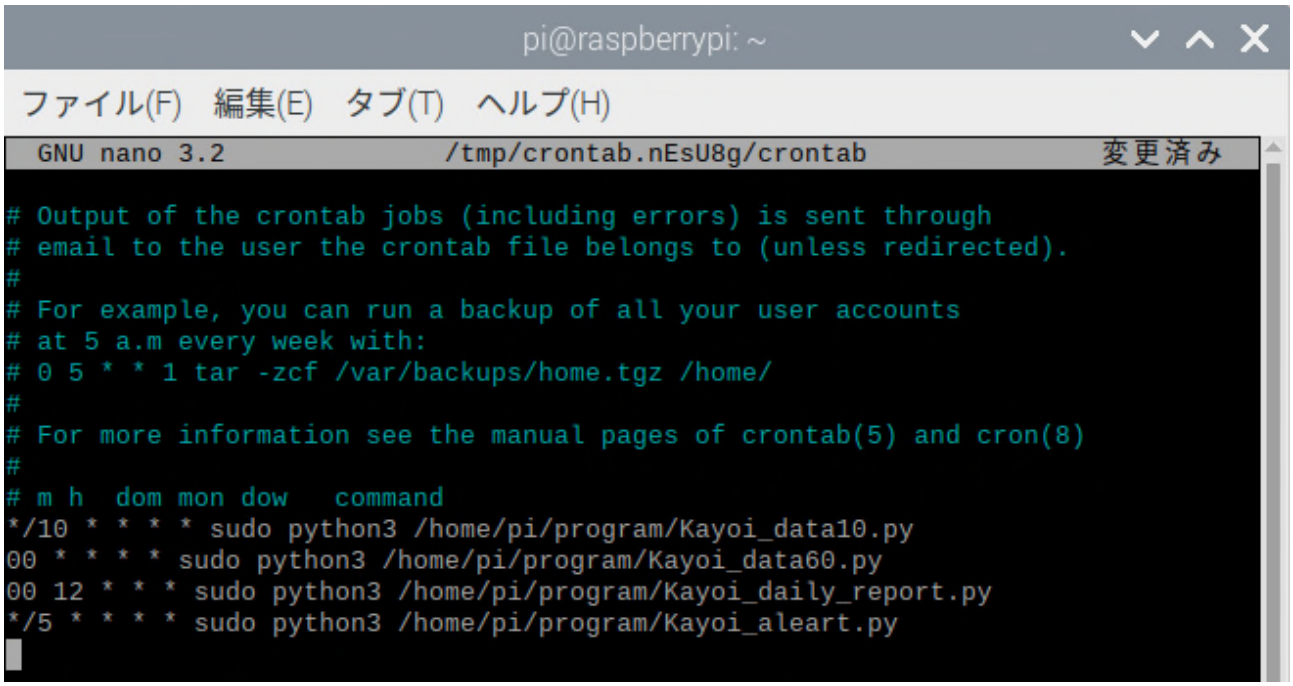
(1) 001 ハウス情報(定期通知)　　… 1 時間毎に通知する
(2) 002 ハウス情報(日報・グラフ通知) … 毎日 12 時に平均温度やグラフを通知する
(3) 003 ハウス情報(警報通知)　　… ハウス温度が 30℃を超えたら通知する

●プログラムの設定

(1) Kayoi_data10.py　　… 10 分間隔のデータ保存用。通知は行わない。
(2) Kayoi_data60.py　　… 60 分(1 時間)間隔で「001 ハウス情報(定期通知)」へ通知する。
(3) Kayoi_daily_report.py … (1)で保存した 10 分間隔データ(data10.txt)から平均温度と最高、最低温度に加えてグラフを作成し、「002 ハウス情報(日報・グラフ通知)」へ通知する。
(4) Kayoi_alert.py　　… 5 分毎に温度をチェックし、ハウス温度が 30℃を超えた際に「003 ハウス情報(警報通知)」へ通知する。

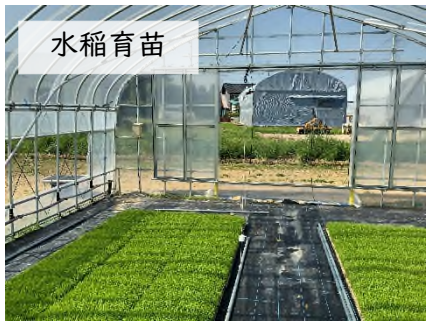
ターミナルで crontab -e を実行し、crontab を以下のように設定します。

```
*/10 * * * * sudo python3 /home/pi/program/Kayoi_data10.py
00 * * * * sudo python3 /home/pi/program/Kayoi_data60.py
00 12 * * * sudo python3 /home/pi/program/Kayoi_daily_report.py
*/5 * * * * sudo python3 /home/pi/program/Kayoi_aleart.py
```



第７章 現場での使用事例

１．水稲育苗ハウスでの水稲育苗・タマネギ育苗の事例



南相馬市小高区の農業法人「（株）飯崎生産組合」の水谷社長

生産者の説明

平成 22 年に小高区の担い手組織として飯崎生産組合を設立。震災後は組合員とともに小高区に通いながら、農地の保全管理などに尽力し、営農再開に向けた準備を行ってきた。

平成 28 年の避難指示解除後、水稲、大豆、タマネギ、カスミソウの栽培を再開し、経営体強化のため、平成 31 年に法人化し代表に就任。所有するハウス 6 棟に「通い農業支援システム」を導入し、水稲やタマネギ育苗時のハウス管理を行っている。

生産者の声（導入の目的とその効果）

管理する耕地面積が広いため、耕地からハウスまで農機で移動するには多くの時間が必要です。これまで、育苗ハウスの管理（側窓開閉・灌水）に行き来するのは作業効率が良いとはいえませんでした。また、ハウス管理が必要かを判断できるハウス遠隔監視システムが必要と考えていました。

「通い農業支援システム」を導入することで、定期的に温度やグラフ、画像が手元で得られ、ハウスの状況がよくわかります。どこにいてもハウス管理の判断ができるので、作業計画が立てやすくなりました。苗の様子をデータから判断し、灌水するようなこともできるので、直接ハウスに行く回数が減りました。特に、農作業が忙しいときほど役に立っています。

2. 水稲育苗施設（ハウス 14 棟、催芽機）の事例

岩手県奥州市の農業法人「（有）ピース」の家子社長と IT 担当の藤原さん

生産者の説明

水稲や大豆を 100ha 超（受託含む）で栽培する大規模な農業法人。水稲苗は自社利用のほか、地域の生産者に販売しており、年間約 1.5 万箱を生産している。

水稲育苗施設（水稲育苗ハウス 14 棟、催芽機2機）に「通い農業支援システム」を導入。催芽機内の温度、プール育苗を行っているハウスの温度、プール水温を得て、水稲育苗の微妙な温度管理に利用している。

生産者の声（導入の目的とその効果）

育苗に用いる催芽機は蒸気加温方式で自社開発しました。ボイラやヒーターで催芽機内の温度を制御して芽出しや催芽を行いますが、安定した品質の苗を生産するには、ボイラやヒーターの稼働状況を常に監視する必要があります。以前はチームを作って輪番で夜間の見回りを行いました。車で往復 1 時間かけて確認作業を行う社員もおり、見回りは大きな負担でした。また、ハウス 1 棟ごとに市販の温度センサをつけて確認作業を行うことも大変な負担でした。

「通い農業支援システム」で遠隔監視ができるようになり、夜間の見回りが家にいてもできるようになりました。ハウス温度をスマートフォンでいつでも確認できるようになったことにより、ハウスの急な温度上昇などにも育苗担当チーム全員が対応できるようになりました。

3. イチゴ育苗ハウス・花きハウスの事例

川俣町山木屋地区生産者の谷口豪樹さん

生産者の説明

震災後、義父の農業を手伝ったことをきっかけに就農。地域の先導的な花き生産者の一人。川俣町山木屋地区で熱帯植物のアンスリウム（復興の花「かわまたアンスリウム」）を栽培。また、ヒマワリやストック、カラーなどの花きのほか、イチゴの苗生産を行っている。

アンスリウムハウス、花きハウス、イチゴ育苗ハウスに通い農業支援システムを導入し、ハウス内の温度のほか、培地の土壌水分なども通知し管理に用いている。

生産者の声（導入の目的とその効果）

アンスリウム栽培は温度管理が重要なため、ハウス内の温度環境は自動で制御されています。しかし、暖房などの稼働状況を離れていても確認する方法はありませんでした。アンスリウムの品質維持のため、何度もハウス内の温度を確認に行き来する必要がありました。

「通い農業支援システム」を活用し、スマートフォンでハウスの状況を確認できるばかりか、実際にハウスにいなくても、遠くから作業の指示をすることができるようになりました。また、地震で暖房が停止するような突発的な状況にも自宅に居ながら、すぐに状況を確認し適切な対応をすることができるようになりました。温度に敏感な花の調整作業も、作業スペースの温度をグラフで確認できるため安心して作業ができ、品質維持に非常に役立っています。

4. イチゴハウスの事例

群馬県昭和村の農業法人「（有）農園星ノ環」の星野社長

生産者の説明

平成 17 年に開拓農家3代目の星野社長が設立。レタス、小松菜、ほうれん草などの高原野菜を約 18ha 生産している。また、5 年前から新規作物としてイチゴ栽培に取り組み始めた。

イチゴ栽培ハウスに「通い農業支援システム」を導入し、ハウスの温湿度や培地温度、土壌水分、CO_2濃度を測定してスマートフォンに通知し管理を行っている。

生産者の声（導入の目的とその効果）

新たにイチゴ栽培を始めるかどうかを検討していました。まずは小規模な生産から始めましたが、他の作物の作業もあり効率的な運営には、ハウスのモニタリングシステムなどの技術の必要性を感じていました。施設栽培の担当社員と技能実習生で管理を行っていますが、はじめはポカポカしてきたら側窓を開けるなど体感に頼った曖昧な栽培管理でした。

高価な市販品の購入は困難でしたが、安価な「通い農業支援システム」を知り、すぐに導入を行いました。温度センサが無かった頃の管理とは異なり、今では管理に携わるスタッフ全員がいつでもスマートフォンで温度を数値で確認でき、データに基づいた明確な管理の指示ができるようになり、とても役に立っています。

施設栽培担当も在宅時にデータを遠隔で確認できるため、自宅から管理作業の指示だけでなく、現地設置センサのメンテナンスの指示などを出しています。

5．マッシュルームハウスの事例

栽培の様子

マッシュルームハウス

岩手県八幡平市　ジオファーム八幡平の船橋代表

生産者の説明

馬を中心とした持続可能な仕組みづくりを行っているジオファーム八幡平の代表。現役引退した競走馬からの馬ふんを用いた培地でマッシュルームを生産し、生産後の培地からは良質な馬ふん堆肥を生産している。

マッシュルーム栽培では培地の温度を指標に管理作業を行うが、この温度指標を安定して得るために「通い農業支援システム」を導入した。安価に複数箇所の温度データを得ている。

生産者の声（導入の目的とその効果）

馬ふん×マッシュルーム×地熱など、馬を中心とした新たな循環の形を実践しています。マッシュルーム生産時には馬厩肥自体が発酵するため、温度管理が極めて重要となります。生育ステージに応じたハウス内の温度設定をどうするかといった管理指標を作るには、1 棟あたり 4 段ある栽培棚2つの床温を測定するために何度も足を運ぶ必要があり、なかなかデータに基づいた管理指標を作ることができずにいました。

「通い農業支援システム」の導入で、遠隔監視できるようになってからは満足のいく温度管理ができるようになりました。今後は湿度のコントロールに向けて活用する予定です。また、自社の馬ふんを関東の馬厩肥生産施設に毎週トラックで運んでいますが、長距離運搬中も休憩時間にはハウスの床温をスマートフォンで確認できるので、余計な心配をせずに輸送でき安心できています。

6．福島における水稲育苗ハウスを活用した香酸カンキツ栽培の事例

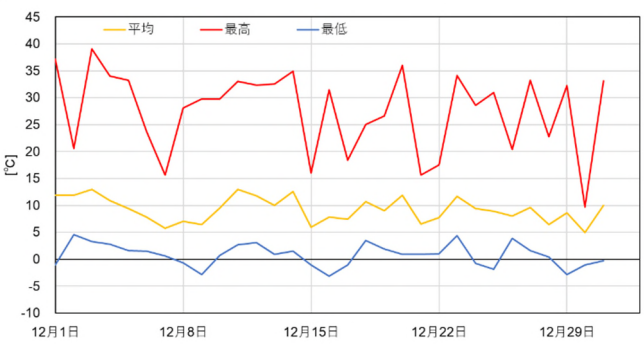
事例の説明

福島の被災地の営農再開地域では、大規模な水稲生産や花きなどの施設園芸を中心に営農再開しています。営農を促進するためには、魅力的な新規作物が必要です。そこで、通い農業支援システムを利用して、水稲育苗ハウスを活用し「スダチ」や「カボス」といった香酸カンキツのポット栽培を行いました。

使っていない時期の育苗ハウスを利用して香酸カンキツを越冬させます

育苗ハウスから搬出した後は、露地で栽培を行います

通い農業支援システムの活用方法



南相馬市小高区ハウス内温度（2019 年 12 月）

冬季の寒害回避や、急に暖かくなり新芽が吹くようなことにならないよう、温度管理が重要です。
そのために「通い農業支援システム」が有用です。

第８章 使用する部品と入手先について

本マニュアルで使用した部品と入手先の一例
通販サイト「Amazon」や、IoT に関する部品を多く販売している通販サイト
「スイッチサイエンス」「マルツオンライン」、「共立エレショップ」などで購入可能です。
なお、URL は 2021 年 3 月にアクセスして確認したものです。

・マイコン　Wio Node
https://www.switch-science.com/catalog/2799/
https://www.amazon.co.jp/dp/B01HT25304/

・温度センサ one wire temperature センサ
https://www.marutsu.co.jp/pc/i/829201/

※熱電対を使用されたい方は以下のものもあります。
・熱電対モジュール
https://www.switch-science.com/catalog/5345/
https://www.switch-science.com/catalog/4102/
ご自身で使用される熱電対を購入される必要がありますが、
センサの長さを自分で延長できるため便利です。

・温湿度センサ GROVE - 温湿度センサ（SHT31） v1.0
・温湿度センサ GROVE - I2C 高精度温湿度センサ（SHT35）
https://www.switch-science.com/catalog/2853/
https://www.switch-science.com/catalog/5337/

・土壌水分センサ
https://www.switch-science.com/catalog/814/
使用条件によりますが、耐久性は約 1～2 ヵ月です。
使用の際は交換用に予備を用意しておくと安心です。

・照度センサ
https://www.switch-science.com/catalog/5345/
測定レンジを確認して用途に応じて使用してください。

・CO_2センサ
https://www.switch-science.com/catalog/5345/
測定レンジが狭いので、使用には注意が必要です。

・USB ケーブル　3m,1.8m など
https://www.amazon.co.jp/dp/B071S5NTDR/

https://www.amazon.co.jp/dp/B01MUJOA5E/
100 円ショップなどの安価なケーブルであっても、
急速充電対応のものは電線径が大きいです。

・**USB 延長ケーブル 5m**
https://www.yodobashi.com/product/100000001001680205/
ハウス内で使用する際は、2 つ繋いで 10m にして使用することは可能です。
接続部はビニールテープを巻いて、漏電しないようにしてください。
3 つ繋いで 15m 以上の延長については電圧降下が起きるため推奨しません。

・**USB-AC アダプタ**
https://www.amazon.co.jp/dp/B07DHP4QZ8/
100 円ショップなどではなく、ある程度の価格で良質な性能を持った製品がおすすめです。

・**漏電遮断機**
https://www.amazon.co.jp/dp/B01LMWDA60
ウォッシュレット機能付き便座とセットでよく使用される漏電遮断器です。
現場で万が一漏電した際に遮断してくれる装置です。使用を強く推奨します。

・**RV ボックス**
https://www.amazon.co.jp/dp/B000XBLYVA
ハウス内に設置する際に、
USB-AC アダプタやモバイル WiFi ルータなどを入れておくために使います。
転倒防止のため、中にコンクリートブロックなどを入れておくことを推奨します。

・**小型 PC（Raspberry Pi）**
https://www.switch-science.com/catalog/3880/
https://www.physical-computing.jp/product/1336
ケースやケーブル、microSD が付属するものから本体だけのものもあります。

注意

本資料を用いて作成した「通い農業支援システム」は、自己の責任において製作・利用する遠隔監視システムです。本システムを用いて生じた故障又は損害等に関しては一切の責任を負いかねますのでご了承ください。

また、メッセージ通知用の Web API が用意されているアプリケーションには LINE や Slack があります。2021 年 3 月に明らかになった LINE 利用者の個人情報が中国の関連会社で閲覧可能となっていた問題を受け、行政機関などで LINE の使用を禁止する動きがみられます。メッセージアプリの利用に際しては、サービス提供会社のプライバシーポリシーを十分に確認するとともに、自身の組織の情報セキュリティ管理方針に基づいて利用を判断下さい。

謝辞

本マニュアルは農林水産省委託事業「原発事故からの復興のための放射性物質対策に関する実証研究委託事業」による研究の成果です。

著者・発行者

2021年3月

「安価かつ簡便にハウスの遠隔監視に使える IoT 機器「通い農業支援システム」製作マニュアル」

著者　山下善道、稲葉修武、内藤裕貴、星典宏、金井源太

発行者　農研機構　東北農業研究センター

問い合わせ先

国立研究開発法人農業・食品産業技術総合研究機構
東北農業研究センター　地域戦略部研究推進室広報チーム
〒020-0198 岩手県盛岡市下厨川字赤平 4
TEL：019-643-3414
MAIL: www-tohoku@naro.affrc.go.jp
http://www.naro.affrc.go.jp/laboratory/tarc/