令和４年度卒業研究報告書

M5StickCPlus による自動水やり機能付き プランターの作成

情報技術科　高橋　佳輝

指導教員　　佐々木　建

目次

第1章　はじめに

私は、1年次に学んだArduinoの授業やインターネットの掲載記事をみてIoTについて興味を持った。自分でもワンボードマイコンを用いてIoTの形で何か自動化できないかと思い模索していたところ、M5Stackシリーズで取り組んでいる事例を目にした。そこで私は、今まで具体的に使ったことのないRaspberry PiやM5Stackで何か事例等がないかと書籍やインターネットから情報を模索していたところ、M5StackシリーズのM5Stick C Plusを使った事例があり、使用方法等を参考にしながら何か形にしたいと考え、卒業研究で取り組むことに決めた。

過去の卒業研究でもM5Stackを用いたテーマがあったが、そのテーマで取り組んだ研究はほとんど完成しており、踏襲して行うに至らなかった。テーマについては、色々と模索をした結果、スマート農業などでも応用が出来そうなテーマでもある「自動水やり」をテーマに取り組みをすることを考え、プランターでのサイズに絞り、土壌の湿り気の測定結果をセンサで測定し、スマートフォン（またはパソコン）でデータを受け取り、何かしらスマートフォン（またはパソコン）上で操作ができる、言わばM5stick C Plusとの通信機能の中でやり取りできる研究にしたいと考え、M5Stackで使用する水分推定センサを用いて土壌の水分量のデータを題材にして、この研究に取り組んでみたいと考えた。

研究イメージは頭にあったものの、実機の入手に時間がかかったため、スケジュール管理は徹底しなければならないことと、参考文献についても早い段階で入手できるようインターネットなどを通して情報を得て、担当教員と相談をしながら進めていった。研究の詳細については次章以降で述べる。

第 2 章　開発概要

本研究では、M5Stack シリーズの M5Stick C Plus を使い M5stack 用水分推定センサ付きポンプユニット(M5STACK-U101、以下、「ポンプユニット」と呼ぶ)により土壌の水分量を計測し、一定値を上回る（乾いていると判断する）と散水する。水を出した状況を知らせるために、LINE を通してスマホに通知を行う仕様とした。

二つ目の機能としてはプランターの現状をリアルタイムでウェブページにより映像を確認できるような仕様にした。研究当初の計画では予定には入れていなかった部分であるが、同じ M5Stack シリーズの M5StickV の AI.Camera を使えば容易にリンクできるものと思っていたものの、Camera の納期に時間がかかり入手までに時間がかかってしまったため、M5Stick C Plus との連携について詳しく追求することができなかった。従って、リアルタイムの映像の確認については、Raspberry Pi を使用する仕様に変更した。

三つ目の機能としては、スマホやパソコンの Web 上からボタンをクリックすることにより、手動での散水機能を遠隔操作にて行えるようにした。

最後四つ目の機能としては、取得した土壌の湿り気数値のグラフ化をする仕様である。

アンビエントデーター株式会社の IoT クラウドサービス「Ambient」を活用してプランター内の土壌の水分量の推移をグラフ化により可視化した。

以上のように自動機能部分と手動機能部分及び実際のプランターの設計の 3 つの柱で研究を行った。

図 1　実際の作成物

第 3 章　開発環境

3.1 開発環境についての概要

今回の研究で作成したシステム構成図を図 3.1 に開発環境を表 3.1.1 表 3.1.2 に示す

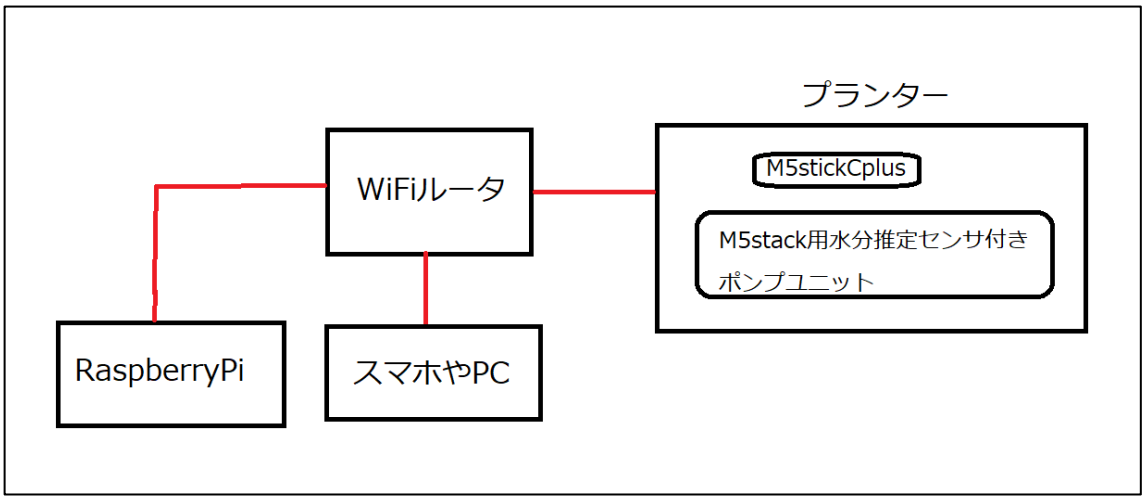

図 3.1.1 システム構成図

OS	Windows10
使用言語	Micropython,
主な使用機材	M5stickCplus 1.1　（M5Stack Technology Co., Ltd.） M5stack 用水分推定センサ付きポンプユニット （M5STACK-U101: M5Stack Technology Co., Ltd.） RaspberryPi 4 model b（4GB）
使用サービス	Google スプレッドシート(データ書き込み用) LINENotify　IFTTT Ambient　M5flow

表 3.1.1 マイコン開発環境

OS	Raspberry Pi OS Bullseye
使用言語	HTML,css,JavaScript
web サーバーアプリケーション	nginx
接続機器	UCAM-C0220FBNBK(web カメラ)

表 3.1.2 RaspberryPi 開発環境

Raspberry Pi は状況確認用の Web カメラとして使用したが、Raspberry Pi の特性を活かし、同時にデータ確認ページの Web サーバとしても運用できる仕様にすることにした。

今回の研究では Web サーバーアプリケーションに Nginx（エンジンエックス：Modern Hire Co., Ltd）を使用した。Nginx とは、無料でソースコードが紹介されているオープンソースの Web サーバをいう。特徴は以下のような特徴がある。

① 処理性能に重きが置かれている

② 高い並行処理性能を持つ

③ メモリーの使用量を抑制できる

また、プランターへの実装についてもポンプユニットをプランターのどの位置に配置するか、散水用に組み上げた水をどう散水するか等の問題も、展示用のプランターにするためには様々な工夫をしなければならないことがわかった。ポンプユニットに搭載されている散水用のホースの口が１つしかないため、最初はプランターの一部にしか水が行き渡らない問題が発生した。１本のホースから３本位に分岐できないかと調べたところ、分岐をさせるための部品をインターネット等で見つけ、それを使用することにより３箇所での散水が可能となった。このことによりプランター全面が同じような水分量になることをふまえ、１箇所での水分量測定で適正な値を得ることができるようになった。

3.2 使用機材

3.2.1　M5stickCplus

M5Stick C Plus とは、M5Stack テクノロジー社のマイコンボードで M5stick C を大画面(1.14 インチ TFT 液晶画面)にしたアップグレード製品で ESP32-PICO-D4 を搭載し、Bluetooth や Wi-Fi にも対応し通信が可能になっている。Arduino IDE や Platform IO、ビジュアルプログラミング(M5flow)、Micro Python など様々な開発環境で開発が可能なマイコンになっている。Grove 端子がついており他の M5Stack の製品ユニットと接続することで様々な動作が可能になっている。今回は、M5Stack 用水分推定センサ付きポンプユニットというものを用いて水分量値により自動で散水すること。また、計測した水分量を通信技術によりスマホで確認できること。スマホ上から手動でも散水できること。更に受け取った水分量データからグラフを作成し可視化したことなどの機能を搭載した。

M5Stick C Plus の主なスペックについては次ページの表 3.2.1 で示す。

図 3.2.1　M5stickCplus

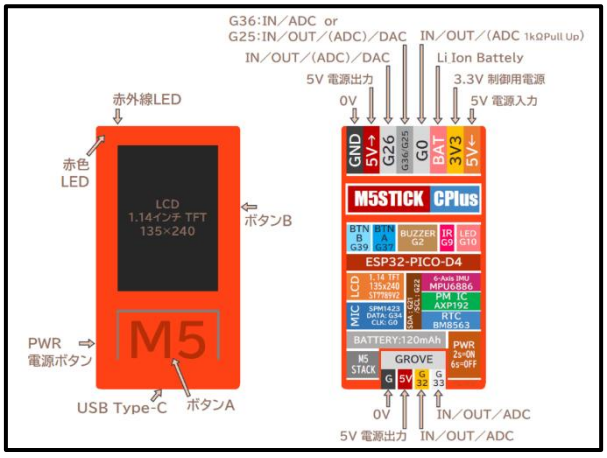

図 3.2.2 ポート・装備品一覧

TFT コントローラー	ST7789
液晶サイズ	1.14 inch
画面サイズ	135 × 240
ブザー	有
バッテリー	120mAh @ 3.7V
PIN ポート	G0, G25/G36, G26, G32, G33

表 3.2.1 M5StickCPlus スペック

3.2.1 M5Stack 用水分推定センサ付きポンプユニット

M5Stack 用水分推定センサ付きポンプユニットとは M5Stack 製品専用の水分推定センサとポンプがついているユニットになっている。接続には Grove ポートを用いて接続し、センサの部分を土壌に刺して水分の計測を行いポンプ部分で散水する。

図 3.2.3 ユニットの画像

図 3.2.4 ユニットの接続の様子

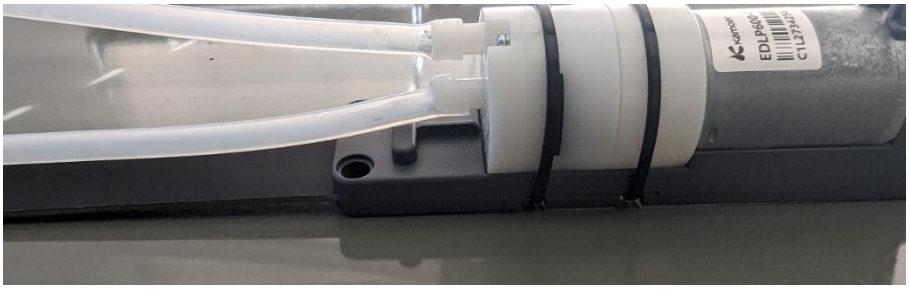

図 3.2.5 ユニットの拡大画像

図 3.2.5 のポンプ部分を見ると、外側（天井側）のパイプで水をくみ上げ、散水するときは内側（センサー側）のパイプで散水する。前述の通り、散水は 1 本のパイプが標準になっているためプランター全体に水が行き渡るようにするためには工夫が必要であったため、今回の研究では 1 本から 3 本に分岐をさせる部品を使い、プランター全体に水が行き渡るよう工夫した。1 箇所の散水による取得データと 3 箇所から散水した取得データとでは値の信憑性が異なったと思われるのでこの工夫は研究の中では必要性が高かった。

また、失敗した点は土入の水を汲み上げてしまい、汲み上げポンプ自体の目詰まりをさせ、作動できなくした点である。分解をしてクリーニングをしたら動くようになったが注意が必要である。水がないときにポンプを動かすことも故障の原因になると考えられるため使用は避けるようにする。

3.2.2 RaspberryPi

今回は監視カメラとウェブページの実装の際に RaspberryPi を使用した。

RaspberryPi とは、イギリスのラズベリーパイ財団が教育用に開発した安価なシングルボードコンピューターです。特徴として電源、ストレージ、入力装置、OS を用意すれば本格的なコンピューターを作ることができ、多彩なインターフェースが標準搭載のため拡張性が高いのが挙げられる。WiFi や Bluetooth の接続も可能である。

種類豊富であるが今回は RaspberryPi4 model b を使用した。RaspberryPi4 model b はデュアルディスプレイに対応し USB3.0 があり現在の RaspberryPi の中では最高クラスのスペックを誇る。

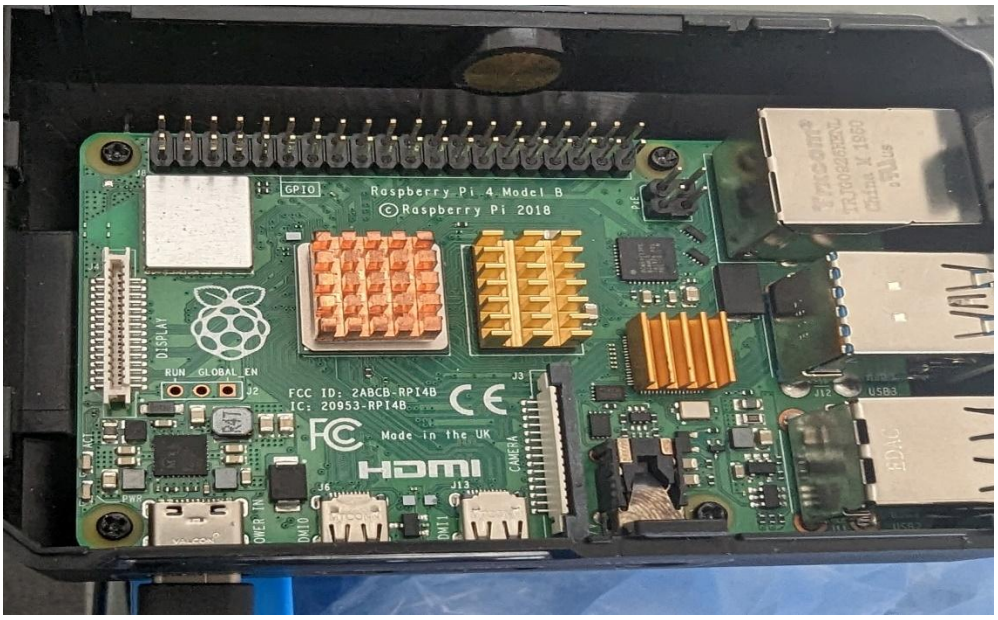

図 3.4.1 実際の RaspberryPi4 model b

今回実際に使用した RaspberryPi4 model b のスペックを以下の通り示す。

CPU		Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC
メモリー		4GB
接続コネクター	USB	・USB3.0 Standard A コネクター×2 ・USB2.0 Standard A コネクター×2
	有線 LAN	RJ-45×1：IEEE802.3i（10BASE-T）、IEEE802.3u（100BASE-TX）、IEEE802.3ab（1000BASE-T）
	その他	・HDMI 出力(マイクロ)×2 ・microSD カードスロット×1 ・3.5mm ジャック（オーディオ/コンポジットビデオ出力） ・Camera interface （CSI） ・Display interface （DSI） ・40 ピン GPIO
Bluetooth		Bluetooth5.0
無線 LAN		IEEE802.11b、IEEE802.11g、IEEE802.11n、IEEE802.11ac （2.4GHz 帯/5GHz 帯対応）
電源		DC 5V

表 3.4.1 RaspberryPi4modelb スペック

3.2.3　監視カメラ用ウェブカメラ(UCAM-C0220FBNBK)

今回使用するウェブカメラである。スペックは以下の通りである。
RaspberryPi に接続して監視カメラ機能を実装させる。
監視カメラに使用するソフトは別途記述する。

カメラ インターフェース	USB2.0(タイプ A オス)
画素数	200 万画素
最大解像度	1600 × 1200 ピクセル
最大 フレームレート	~640px×480px → 30fps 1600px×1200px → 4fps

表 3.2.2 ウェブカメラスペック

図 3.2.6 実際のカメラ

3.3 UIflow について

3.3.1　概要

「UIFlow」は、M5Stack 社が開発した「M5Stack シリーズ」のための開発環境である。 「ビジュアルプログラミング」が特徴で、ブロックを組み合わせてプログラムを作成することや microPyhon でコード書いたりすることができる。ブラウザベースでの開発と Desktop 版で開発との 2 種類があり、内部的には MicroPython で動作する。
初期設定の際には burner というもので初期化を行い、M5StickCPlus のモニターに出される
API キーを入力して設定完了で簡単に行うことができる。

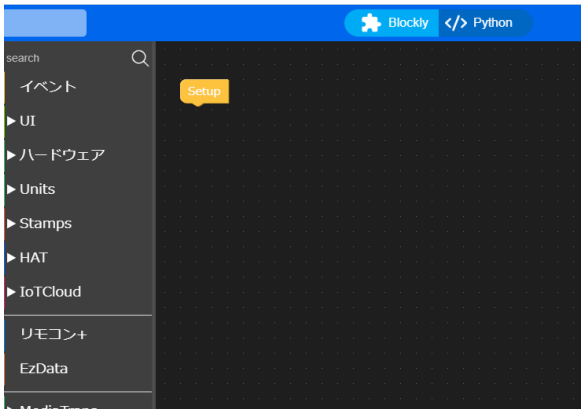

図 3.3.1Blocky での開発画面

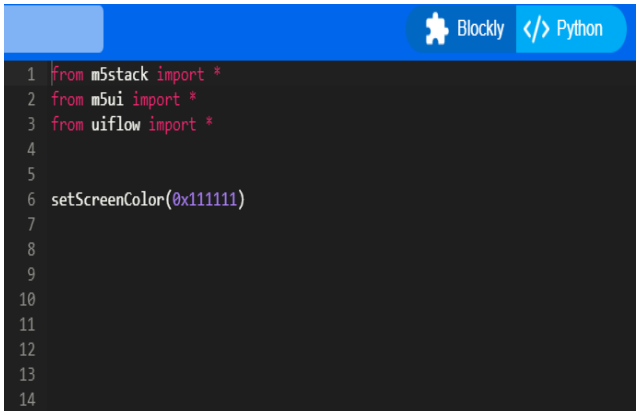

図 3.3.2MicroPython での開発画面

3.3.2　MicroPython

MicroPython は、Python 3 というプログラミング言語をベースに開発されたもので、マイクロコントローラーという制限された環境で動作することを目的として作られた。最小限の Python 標準ライブラリーと MicroPython 固有のライブラリーに加えて、MicroPython が動作するマイクロコントローラーに合わせて、そのハードウェアを操作する固有のライブラリーが用意されている点が優れている。また、uiflow だけでなく VScode でも利用できる。

3.3.3　M5Burner 及び M5Flow のインストール手順

M5Burner 及び M5Flow のインストール手順について示す。

M5Burner 及び UIFlow は https://docs.m5stack.com/en/download よりダウンロードする。

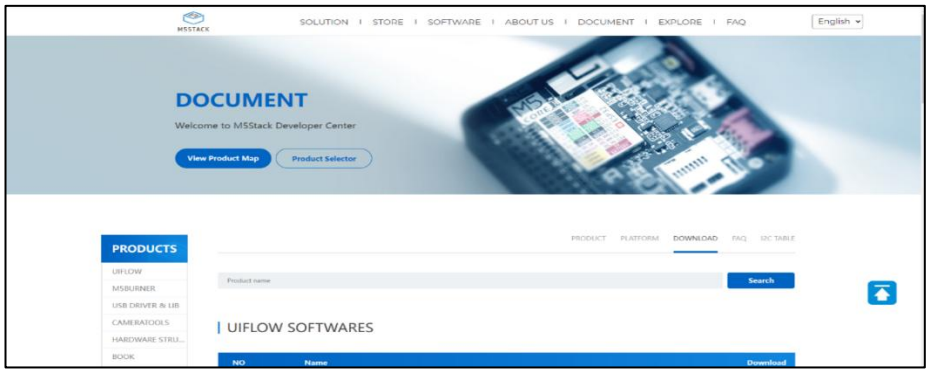

図 3.3.1 ダウンロードページ画面

3.3.4　M５Burner ダウンロード

（1）　M5Burner をダウンロード

一番上の「M5Burner Win10 x64 v3.0」　を選択し、ダウンロードする。

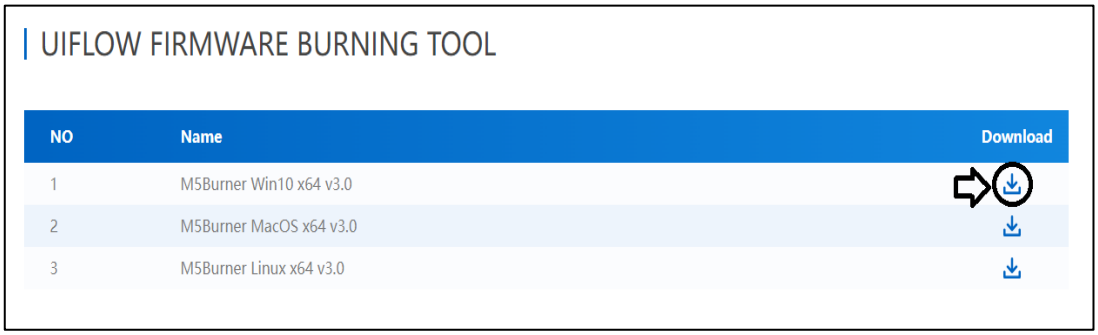

図 3.3.2M5Burner ダウンロード画面

「M5Burner-v3-beta-win-x64.zip」形式ファイルがダウンロードされる。

（2） M5Burner を起動

ダウンロードした圧縮形式のファイルを解凍し、解凍したフォルダの中に「M5Burner.exe」があるのでダブルクリックをして起動する。
この時 M5StickCPlus とパソコンを usb typeC ケーブルで接続しておく。

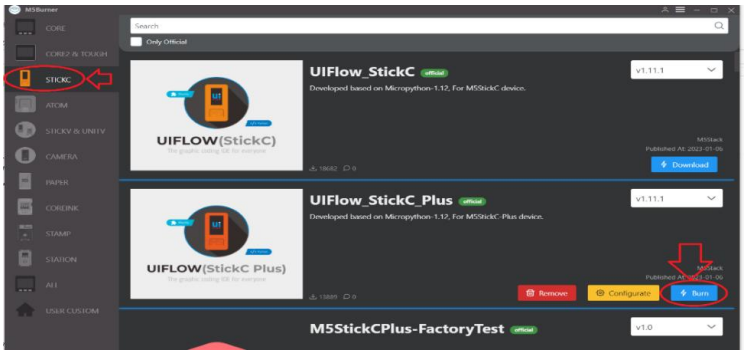

図 3.3.3M5Burner 画面

（3） 「StickC」を選択し、「UIFlow_StickC_Plus」の「Burn」をクリック

Burn を行うことにより、初期化 M5Stick C Plus とパソコンとの関係が開発環境に登録される。
Wi-Fi 経由により UIFlow で書いたプログラムがスムーズに M5Stick C Plus に書き込まれるようになる。この設定をする際には Wi-Fi の環境が必要になる。

3.3.5　UIFlow のダウンロードから設定

（1）　UIFlow のダウンロード

UIFLOWS SOFTWARES から「Desktop IDE Win10 x64 (update is terminated)」 を選択し、ダウンロードする。
一番上にある「UIFlow Web IDE」はウェブブラウザ版の UIFlow のリンクになっている。

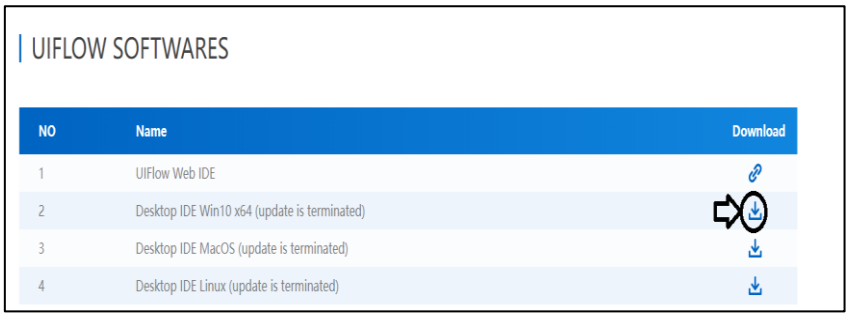

図 3.3.2M5Burner ダウンロード画面

「UIFlow-Desktop-IDE.zip」形式ファイルがダウンロードされる。

（2）　UIFlow-Desktop-IDE の起動

zip ファイルを解凍するとフォルダ内に「UIFlow-Desktop-IDE.exe」の

実行ファイルがあるのでダブルクリックで起動する。起動すると図 3.3.3 の通りになる。

USB ケーブルで M5StickCPlus とパソコンを接続すると利用可能になる。

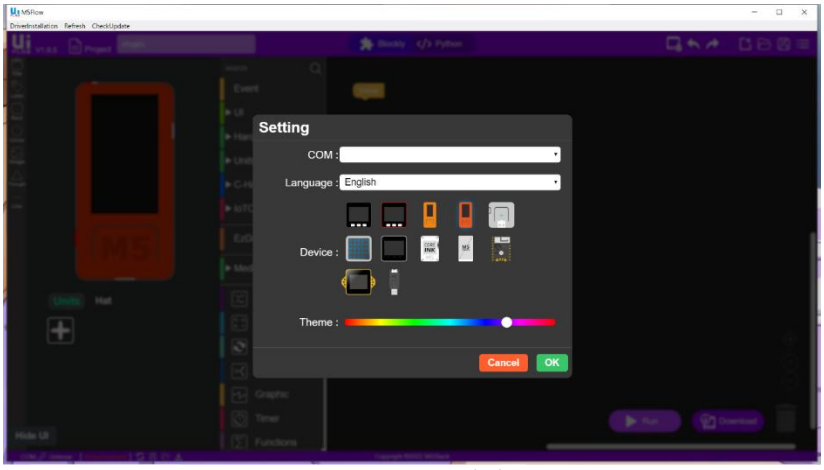

図 3.3.3uiflow 画面

第 4 章　作成

4.1 基本機能の実装

4.1.1　M5StickCPlus の動作確認

MicroPython も M5Stick C Plus も今まで見たことも触れたこともなかったため、様々な動作確認を実験的に行った。
授業で習った言語の記述とも違うため、文法的なものを理解するまでに時間がかかった。

(1)　画面に「Hello M5」の表示

```
from m5stack import *
from m5ui import *
from uiflow import *


label0 = M5TextBox(47, 41, "Text", lcd.FONT_Default, 0xFFFFFF, rotate=0)
label0.setText('Hello M5')
```

コード 4.1　文字表示

(2)　LED の点灯と消灯

```
from m5stack import *
from m5ui import *
from uiflow import *


M5Led.on()
    wait_ms(8)
```

コード 4.2 LED

(3) スピーカー

Speaker の第一引数が周波数、第二引数は秒数を入れている次のコードは装備されている

スピーカーより電子音が鳴ることを確認するものである。

ピーピー

```
from m5stack import *
from m5ui import *
from uiflow import *


speaker.tone(1800, 500)
    wait_ms(3)
```

コード 4.3 スピーカー

(4) ボタン制御

M5Stick C Plus には 2 つのボタンが装備されており、中央のボタンが A、右のボタンが B となっている。次のコードは A ボタンを押すと LED が点灯するかどうかを確認するものである。

```
from m5stack import *
from m5ui import *
from uiflow import *


if btnA.wasPressed():
M5Led.on()
wait_ms(3)
```

ボタン A

コード 4.4 ボタン

4.1.2　水分推定センサ付きポンプユニットの制御

まず、M5Stack 用推定センサ付きポンプユニットが正常に動くかどうか確認するために、
次のコードで確認した。

```
from m5stack import *

from m5ui import *

from uiflow import *

import unit

Watering_0 = unit.get(unit.WATERING, unit.PORTA)

if btnA.wasPressed():

Watering_0.set_pump_status(1)

if btnA.wasRelease():

Watering_0.set_pump_status(0)
```

コード 4.5 水流し（ボタン）

上記コードを実行することにより、「A ボタン」を押すとペットボトル側の水を汲み上げ、
テストプランター側に散水することが確認できた。
このことよりポンプユニットがコード（プログラム）により正常に動くことを確認した。

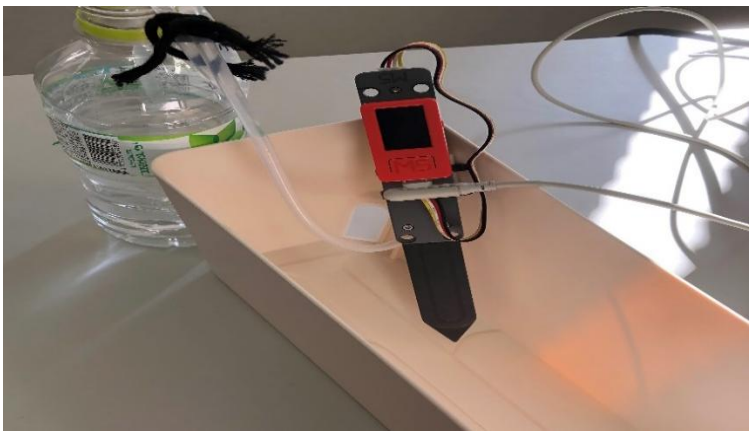

図 4.1.1 実験の様子

次に、推定センサより水分量が測定できるかどうかの確認を行った。前ページの図 4.1.1 で行った実験と併せて行った。次のコードにより推定センサが水分量を測定し、M5Stick C Plus の画面に測定値を返すか確認を行った。

```
from m5stack import *

from m5ui import *

from uiflow import *

import unit


Watering_0 = unit.get(unit.WATERING, unit.PORTA)

label0 = M5TextBox(47, 41, "Text", lcd.FONT_Default, 0xFFFFFF, rotate=0)

label0.setText(str(Watering_0.get_adc_value()))
```

コード 4.6　水分量

上記のコードを実行することにより、測定した水分量を M5Stick C Plus の画面に映し出したことを確認した。以上、ポンプユニットの動作確認は検証した。

4.1.3　散水のための閾値の決定

本研究でプランターに実際に散水するための基準（閾値）を決めるために、事前に実際に使う土壌で散水前と散水後の水分量を測定した。

＜測定結果＞

・プランターに土壌を入れたて時の数値　→　測定値 2000～2050 g/m^3

・散水後の土壌の数値　→　測定値 1800～1850 g/m^3

以上の結果より、平均値である 1900 g/m^3 を閾値とすることに決めた。閾値以上になった場合にポンプユニットから散水するコードで本研究を進めることとした。しかし、散水する箇所、ポンプユニットを設置する場所により、測定値が変わることが予測されたため、改めてプランターでのユニット等の配置を検討した。その詳細については後述する。

図 4.1.3 土壌が乾いている状態

図 4.1.4 土壌に水を流した後

4.1.4　自動散水機能の実装

前述した 4.1.2、4.1.3 の結果等を踏まえて自動散水機能を実装した。次のコードにより閾値以上になると自動で散水するかどうかを確認した。

```
from m5stack import *

from m5ui import *

from uiflow import *

import unit



label0 = M5TextBox(45, 103, "label0", lcd.FONT_DejaVu18, 0xFFFFFF, rotate=0)

label0.setText(str(Watering_0.get_adc_value()))

if (Watering_0.get_adc_value()) >= 1900:    //閾値との比較

      Watering_0.set_pump_status(1)

        setScreenColor(0x00cccc)
```

コード 4.7 自動みずやり

＜主なコードの説明＞

`Watering_0.get_adc_value()`は推定センサから得た水分値のことを指す。`if` 文により閾値 `1900` により散水するかしないかを振り分けている。

`Watering_0.set_pump_status()`は、ポンプユニットの制御部分で値が「1」の場合は水が出て、値が「0」になると止まる。

4.2 スマホやパソコンへの通知機能の実装

4.2.1 LINENotify の概要

LINE Notify(ライン ノティファイ：https://notify-bot.line.me/ja/)とは、LINE と外部での Web サービスやアプリを連携し、ユーザがカスタマイズされた好みの情報をプッシュ通知として受け取ることのできる機能のことをいう。

LINE のアカウントを持っている人なら誰でも活用できる。API も公開されており、活用の幅が広い。連携できるサービスとして IFTTT(イフト)、GIT Hub(ギットハブ)、Mackerel(マカレル)などがある。本研究では LINE Notify と IFTTT を連携させて水分量の通知やスマホ（パソコン）側から手動での散水を行えるようにした。

図 4.2.1LINENotify トップページ

4.2.2　IFTTT の概要

IFTTT（イフト：https://ifttt.com/）とは異なるソーシャルメディア（例えば、Twitter や Facebook、Gmail、Instagram、Evernote、Dropbox など）やプラットフォームを連携させる Web サービスのことをいい、各種 SNS や Google ドライブといったプラットフォームの仲介役を担い、新しいサービスの展開を図る。

IFTTT という名称は「IF This Then That」の頭文字をとっており、あるツールで特定の動作をしたときに別のツールで不随した動作を自動で行うなどをするのが IFTTT の主な機能である。IFTTT では起動させる条件を「トリガー」といい、それによって自動的に行われる動作を「アクション」という。また、IFTTT で既に提供されているサービスのことを「レシピ」と呼んでいる。

使用例としては、「スマートフォンの位置情報サービス（ロケーションサービス）とメールを組み合わせれば、乗換駅に着いたら、自動的に家族にメールを送るといったことが簡単に実現できる。」（参照：https://atmarkit.itmedia.co.jp/ait/articles/1711/22/news031.html）

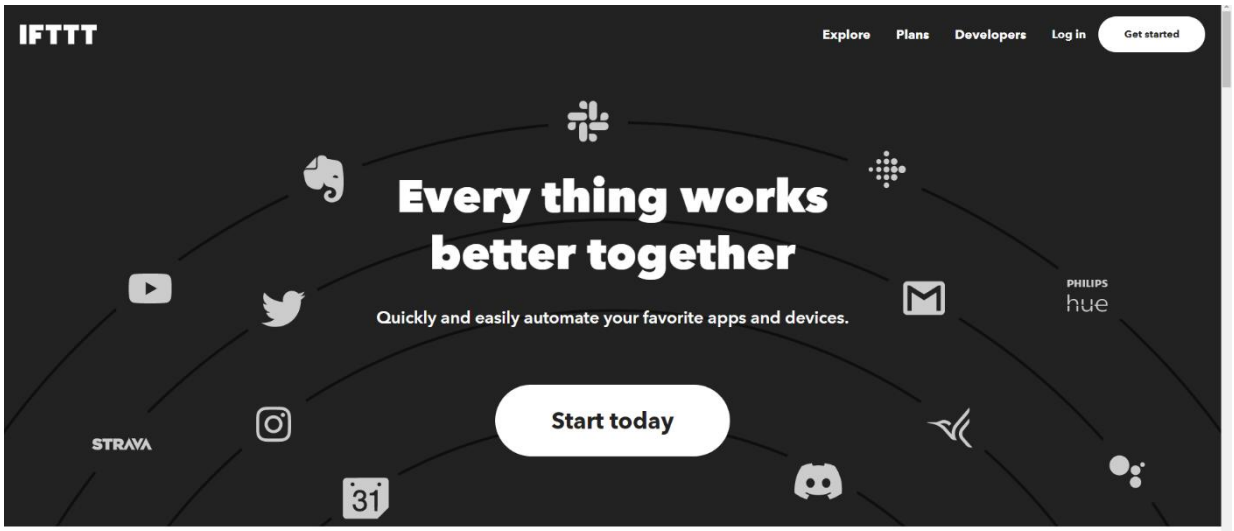

図 4.2.2IFTTT トップページ

4.2.3　IFTTT と LINENotify の連携

IFTTT の使い方及び LINE Notify との連携を以下に述べる。後述するアプセットというものはプロジェクトの作成と同様の意味を持つ。IFTTT は無料版の場合アプセットは 5 つまでしか作成できない制限があるため使用するためには注意が必要である。その手順を次に示す。

(1)　アカウントの作成及びログイン

画面右上にある「Log in」ボタンからログインする。アカウントを持っていない場合はアカウントを作成しなければならない。今回は学校のメールアドレスを使用してアカウントを作成した。次回からのログインは「Log in」ボタンを押すだけでログインすることができる。

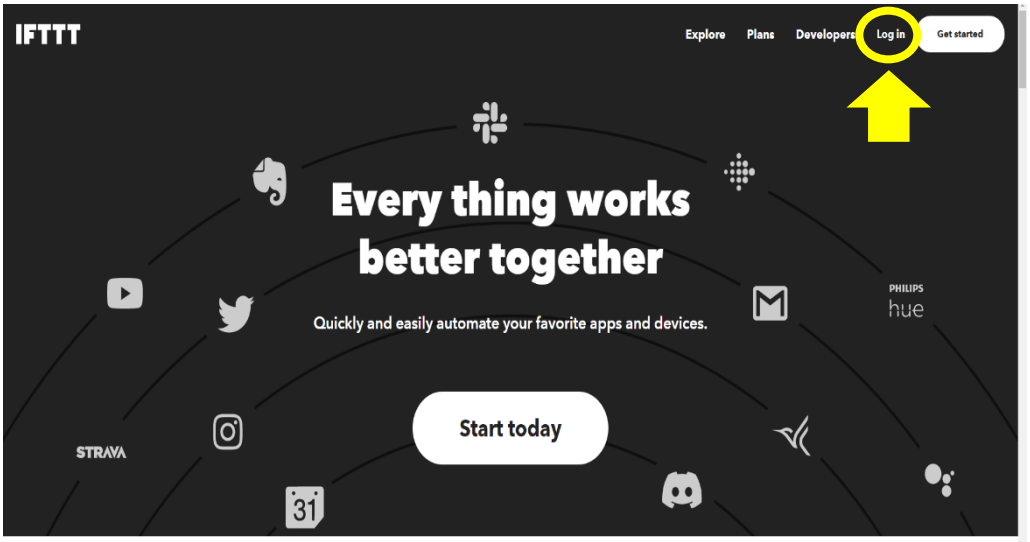

図 4.2.2IFTTT トップページ

(2)　アプレット（レシピ）の作成

前ページの手順(1)でログインが完了したら次の画面に進む。
右上にある「Create」というボタンをクリックすると、アプレットを作成することができる。
アプレットの作成の際には「トリガー」と「アクション」を設定する。

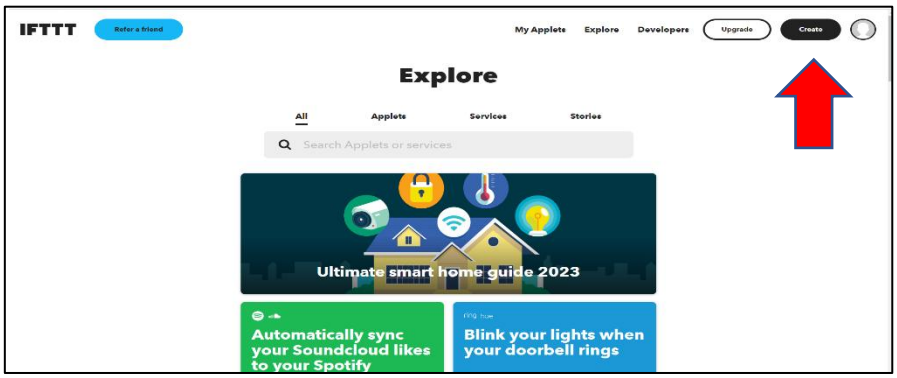

図 2.2.3 トリガー設定

(3)　トリガーの設定

トリガーには「Webhook」を指定する。Webhook とは「Web コールバック」や「HTTP プッシュ API」などとも呼ばれ、あるアプリケーションから別のアプリケーションに対して、リアルタイムな情報提供を実現するための仕組みことをいう。
LINE は Webhook に対応しているサービスなのでトリガーとして使用した。

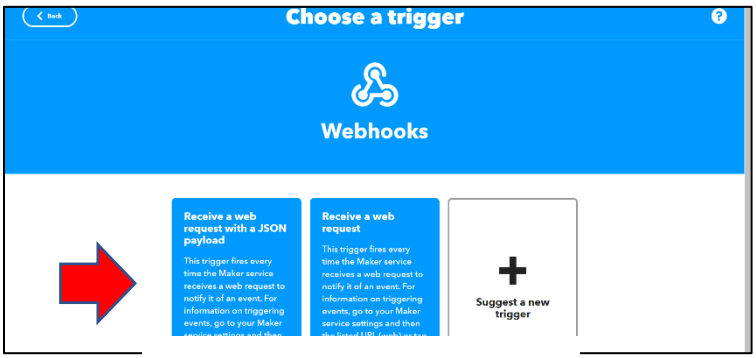

図 4.2.4　トリガー設定 2

(4)　アクションの設定

アクションは LINE を選択します。検索欄に「LINE」と打つと出てくる。
LINE では送りたいメッセージを設定する。
メッセージ以外にも設定したデータを送ることができる。

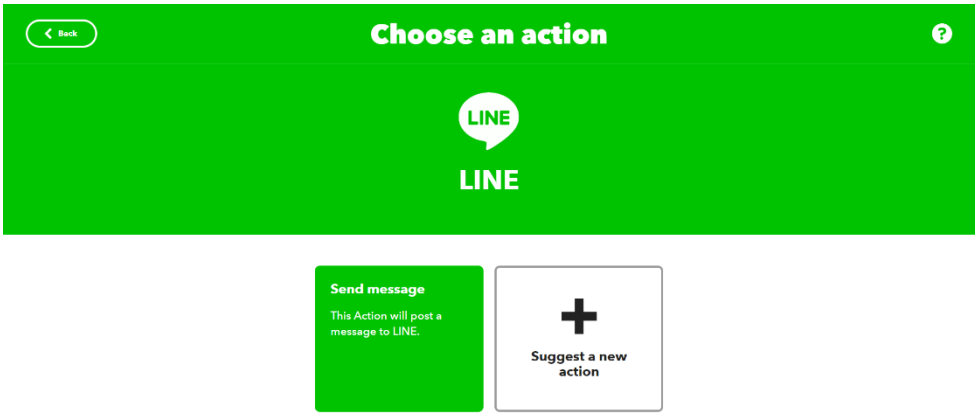

図 4.2.5 アクション設定

設定できる項目等として上から「送信先」、「受信者」、「テキストメッセージ[HTML 文も書ける]」及び「画像 URL」である。

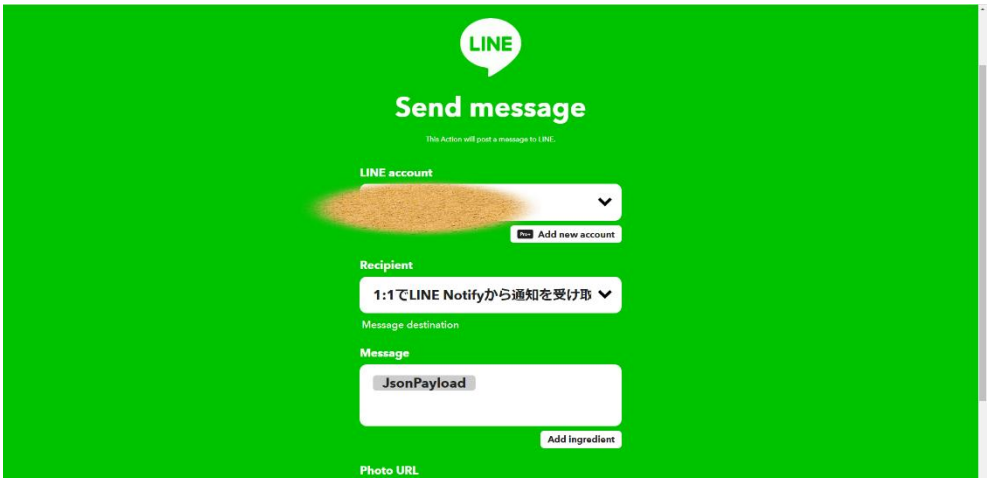

図 4.2.6 アクション設定

(5)　実装

下の画像は「My Applet」の画面である。
先ほど設定した Applet を見ることができ、設定を変更するときは変更したいサービスの
ロゴを選択する。

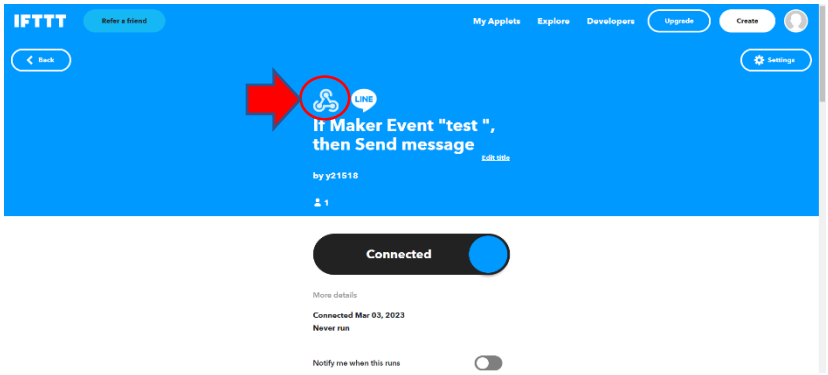

図 4.2.7 マイアプレット

実際に通知を実装させたい場合には webhook の Documentation を選択し
その中の URL をコピーしてコードに埋め込めることで実装できる。

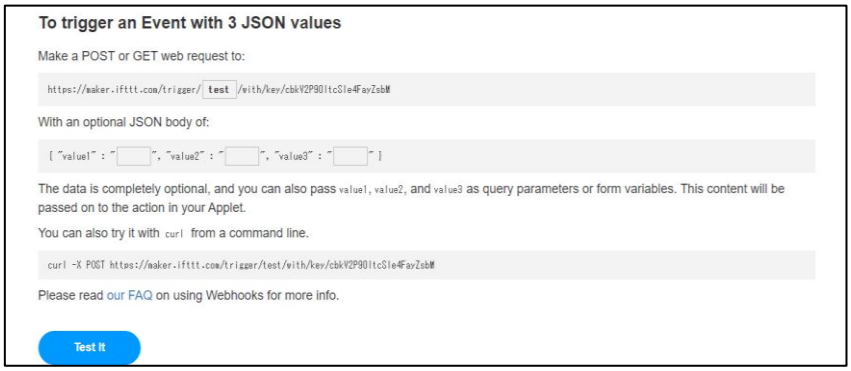

図 4.2.8 Documentation 画面

(6)　MyApplet(レシピ)と散水機能とのリン

次のコードが通知を実装させたときのコードであり、図 4.2.9 は LINE でメッセージを受け取った様子の画像である。M5Stick C Plus が Wi-Fi に接続する必要があるため、接続する処理を加え、http 通信により IFTTT から LINE に通知が届くようにした。

```
～　略　～
Watering_0 = unit.get(unit.WATERING, unit.PORTA)
wifiCfg.doConnect('CIS_A000', '0000000000')
  if (Watering_0.get_adc_value()) >= 1900:
    Watering_0.set_pump_status(1)
    try:
req=urequests.request(method='POST',url='https://maker.ifttt.com/trigger/watertest/with/key/cbkV2P90ltcSle4FayZsbM',json={},headers={})
    setScreenColor(0x00cccc)
  else:
    Watering_0.set_pump_status(0)
```

コード 4.7 自動水やり通知付き

図 4.2.9 通知画面

上記の処理では水分値を測定し、一定値(水分値 1900)を上回る場合はポンプを作動させ土壌に水を流し、水分値が正常である場合は、ポンプを止める動作を行っている。
ポンプを作動させ水を流すとき HTTP リクエスト(POST)を行い、IFTTT のトリガーを起動させて連携している LINENotify で設定しているメッセージを送るようにしている。

4.3 追加実装機能

4.3.1　水分値のグラフ化

前述の通り、土壌の水分量を推定センサにより数値を測定し、測定値をスマホやパソコンに送り表示させることができた。今回の研究で水分量を数値だけで表示しても、変化がわかりにくいと考え、データを Google スプレッドシート上に CSV 形式で蓄積をしていき、そのデータを用いグラフで表現することに仕様を追加した。そのために Ambient（アンビエント：https://ambidata.io/docs/gettingstarted/）というサービスを利用した。

Ambient は細かな初期設定をしなくても、送ったデータでリアルタイムにグラフ化をして可視化する機能を持っている。

使い方として「チャネルを作る」をクリックして名前を設定するのみで使用が可能である。
その後はチャネル ID とライトキーを用いてグラフにデータを送るようにして使っていく。

図 4.3.1 チャネルの作成

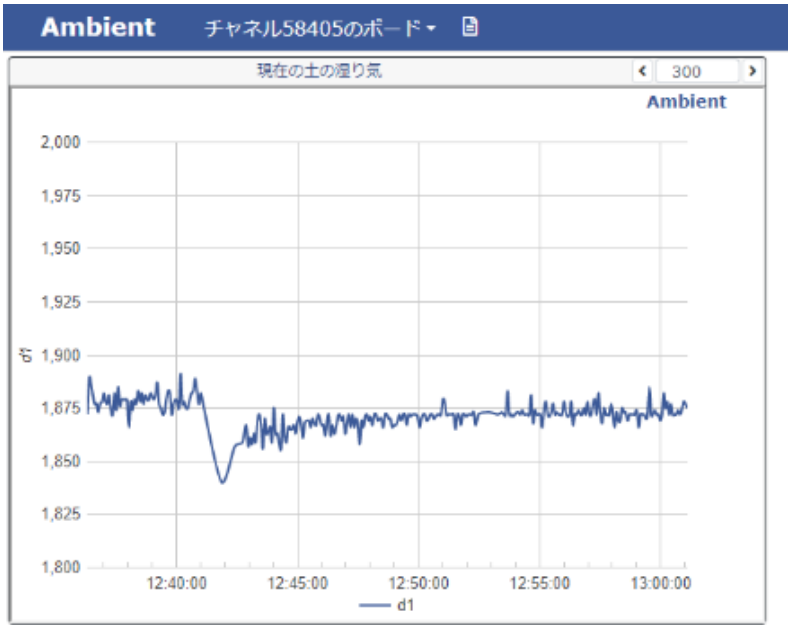

図 4.3.2 実際のグラフ一部

```
～ 略 ～
    def ambientSend(_channelId, _writeKey, _data):
        if isinstance(_data, list):
            __d = _data
        else:
  __d = [_data]
        try:
            req = urequests.request(method='POST', url=(('http://ambidata.io/api/v2/channels/'
+ str(_channelId))) + '/dataarray'),json={'writeKey':_writeKey,'data':__d}, headers={'Content-
Type':'application/json'})
            _ret = True
        except:
            _ret = False
        return _ret

    ambientSend(58405, '12b178398728391a', {'d1':(Watering_0.get_adc_value())})
```

コード 4.8 ambient へ水分量送信

＜主なコードの説明＞

上の処理では ambient に水分値を送る動作をしている。

先ほどの IFTTT と同じように HTTP リクエスト(POST)でデータの送信を行い自分のチャネルに水分値が送られるようにしている。Isinstance 関数では、データ型の判別を行い今回の場合 list 型かどうかを判定しており、list 型でないとき list 型に変換するようにしている。Ambient に送ったデータはサイトで自動的にグラフにされる。

4.3.2　遠隔操作

遠隔操作は M5Stack シリーズの UIFlow に搭載されている機能である。ボタンに割り当てすることにより様々な動作や表示を行うことができる。本研究では実装機能は表 4.3.1 の通りに割り当てた。

ボタン 1	電源オフ
ボタン 2	散水
ボタン 3	スピーカー
ラベル	水分値の表示

表 4.3 実装機能一覧

```
def button_1_callback():
    global Watering_0
    axp.powerOff()
def button_2_callback():
    global Watering_0
    Watering_0.set_pump_status(1)
def button_3_callback():
    global Watering_0
    for count in range(10):
        speaker.tone(1800, 500)
        wait_ms(3)
def label_1_callback():
    global Watering_0
    return Watering_0.get_adc_value()
```

コード 4.9 遠隔操作

図 4.3.3 遠隔操作画面

第 5 章　RaspberryPi について

日本ではラズパイと略されることもある Raspberry Pi（ラズベリーパイ）は、イギリスのラズベリーパイ財団によって開発されました。2012 年に販売が開始され、世界累計出荷台数は 4,500 万台を突破している。

ラズベリーパイは、教育用コンピューターとして開発されたため、価格はリーズナブルである。本研究では Raspberry Pi 4 model B を使用した。

5.1 初期設定

5.1.1　OS のインストール

Raspberry Pi OS は Raspberry Pi Foundation が公式にサポートしている OS であり、Raspberry Pi 用の最も標準的な OS である。Debian がベースになっており、Python が標準でインストールされているのが特徴の OS である。インストール手順は以下に示す。

（1）　Raspberry Pi Imager のインストール

RaspberryPiOS をインストールするために Imager を用いて行った。
Imager を用いて microSD カードをフォーマットして RaspberryPiOS をインストールする。
インストールした microSD カードを RaspberryPi を挿入することにより OS として機能する。
Raspberry Pi 公式ページ(https://www.raspberrypi.com/software/)より Imager をインストールする。

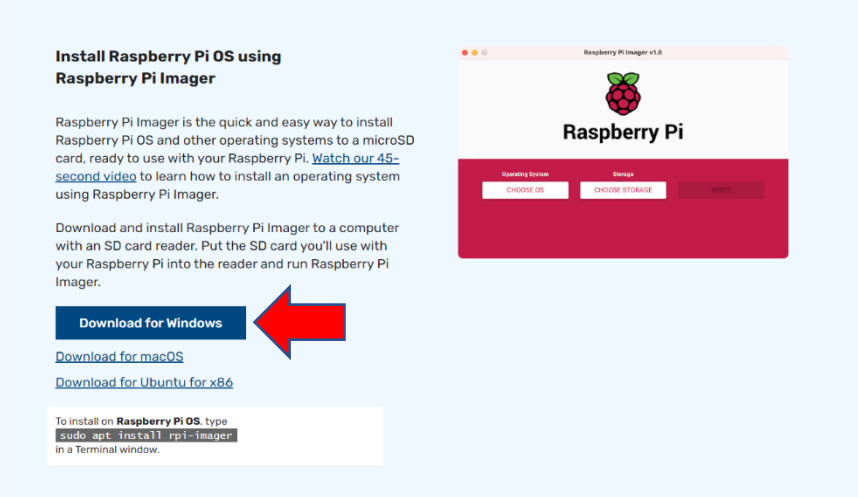

図 5.1.1Imager インストール画面

(2) OS のインストール

手順(1)でダウンロードした実行ファイル（2023 年 3 月 8 日時点：imager_1.7.4.exe）ファイルをダブルクリックし、Raspberry Pi Imager を立ち上げる。micro SD カードを用意し、パソコンに接続されたカードリーダに挿入する。本研究では ADATA 製の micro SDHC/SDXC UHS-I Class10 32GB を使用した。micro SD カードは対応していない種類があるため注意が必要である。

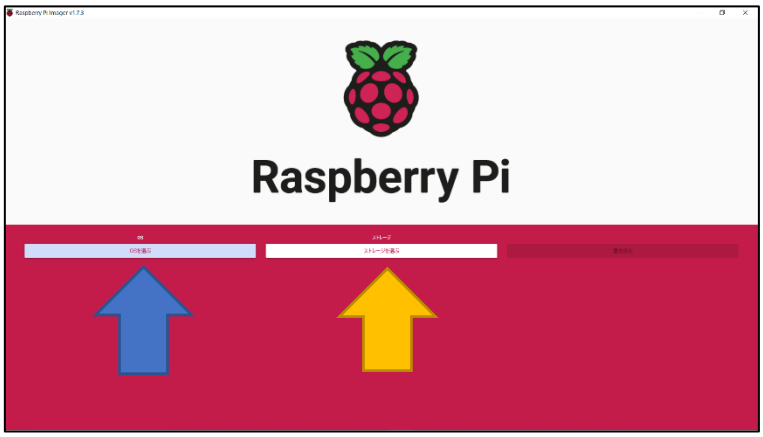

図 5.1.2Imager 画面

図 5.1.2 の青矢印の部分（「OS を選ぶ」）をクリックすると、図 5.1.3 のような選択画面が表示される。本研究では画面一番上の「Raspberry Pi OS 32bit」を選択した。

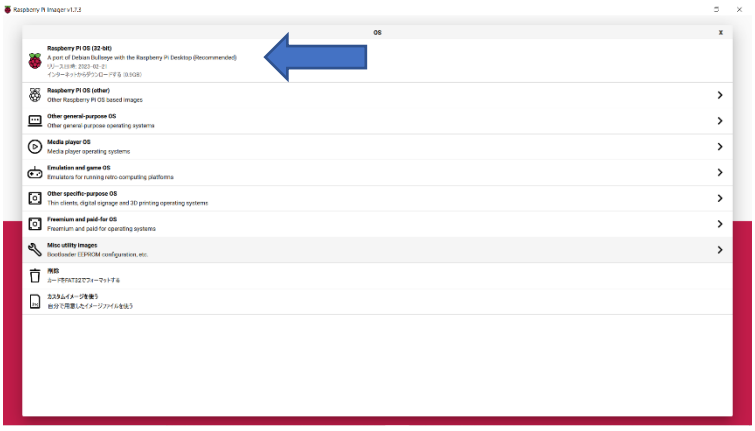

図 5.1.2Imager OS の選択画面

次に、ストレージを選ぶをクリックするとインストール可能なストレージが表示される。複数のストレージを接続していると複数表示されるので注意が必要である。

図5.1.3Imager SD カードの選択画面

最後は「書き込み」ボタンが有効になるので、「書き込み」ボタンをクリックすると SD カードがフォーマットされ、Raspberry Pi OS が書き込まれる。終了すると「終了メッセージ」が表示される。

(3) Raspberry Pi の初期設定

Raspberry Pi 4 本体に microSD カード（専用スロットがある）を挿入して、電源（USB（TypeC））と microHDMI とディスプレイ（HDMI ポート付き）とキーボード及びマウスを接続し、起動する。本研究では初期設定として以下の項目を設定した。

1．言語（日本語）

2．ユーザー名、パスワード

3. WiFi

以上の項目を設定することにより利用可能になる。

5.2 監視カメラ（Web カメラ）の実装

前述の通り、本研究の当初では計画になかったリアルタイムの画像を確認する機能であったが、スマホやパソコン上でもリアルタイム映像を確認したいと思い機能を追加した。M5Stack シリーズのカメラ M5StickV であれば簡単に接続できると考え購入手続きを踏んでいただいたが、納期に時間がかかってしまったため、M5Stick C Plus と M5Stick V との関係を検証する時間が足りず、仕様を変更して Raspberry Pi 4 上で使えるカメラとして、市販の Web カメラを使用することにした。タイムラグはあるものの、思った以上に鮮明なり画像をスマホ、パソコンの画面に映し出すことができたことは満足している。

カメラ機能を実装させるためには「Motion」というソフトを利用した。Motion は「動き」を検知して、静止画や動画を撮影できるソフトウェアであり、本研究で使用することとした。
次に実装までの手順を表す。

(1) Motion のインストール

RaspberryPi のターミナルを起動し、以下のコマンドを入力してインストールする
ターミナルは Linux 系と同じなのである程度のコマンドは授業で習っていたため
違和感は余りなかった。

```
Sudo apt install  –y motion
```

(2) 設定ファイルの変更

Motion のオプション設定ファイルは/etc/motion/motion.conf にある。
まず、インストールする位置までのフォルダに移動する。

```
cd /etc/motion
```

次に Motion.conf のファイルを vi コマンドで編集する
この時管理者権限が必要のため sudo コマンドが必要になる。

```
sudo vi motion.conf
```

設定ファイルの変更点として以下がある。

1. deamon

2. stream_localhost

ローカルホストをオフにしないと外部からアクセスができないため設定が必要である。

```
daemon on → deamon off

stream_localhost on → stream_localhost off
```

起動や停止についてのコマンドは次の通りである。

```
起動

sudo motion

停止(motion のプロセス終了)

sudo pkill motion または sudo service motion stop

状態の確認

ps –def | grep motion
```

これらの設定を行うことで図 5.2.1 の画像のように Web カメラが動作する。
Raspberry Pi、M5Stick C plus と同じ Wi-Fi 環境で接続していれば Google Chrome 等のブラウザ上で確認することができる。

本研究の場合は IP:ポート番号/0/stream の URL で確認することができる。常時リアルタイム映像を流している状態にあるので、一定時間の動画保存の場合は別途に設定が必要であるが、本研究では取り扱っていない。

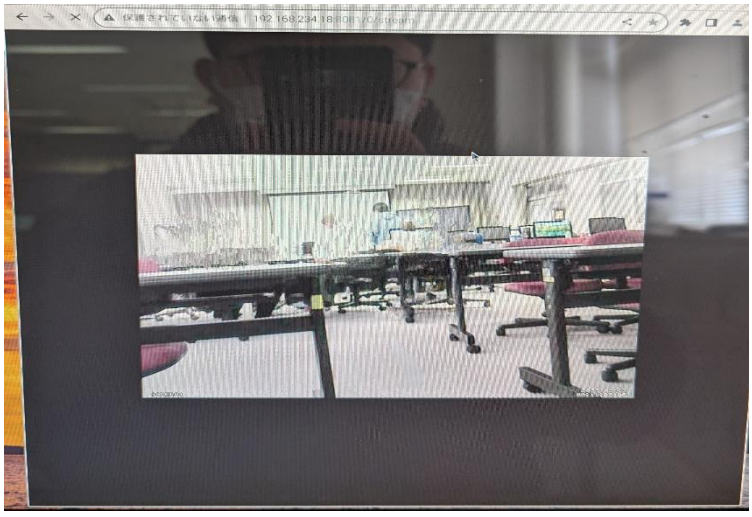

図 5.2.1 監視カメラ動作確認

(3) 自動起動設定

最後に Raspberry Pi 4 が起動したときに Motion が動くように設定する。そのために/etc/rc.local の設定で変更を行った。起動処理が終わる直前で motion を動作させるように設定を変更させる。

```
motion &

exit 0
```

この設定を行うことにより、ターミナルでコマンドを打たなくとも、自動で Motion が起動する。大きな利点として Raspberry Pi に電源を供給するだけで監視カメラ(Web カメラ)が実装できる。

5.3 ウェブサーバーの立ち上げ

本研究では Raspberry Pi 4 で Web カメラを実装させるとともに Web サーバとして運用する仕様にした。Web サーバアプリケーションに Nginx（エンジンエックス：Modern Hire Co.Ltd：https://www.nginx.co.jp/）を使用した。

Nginx とは、無料でソースコードが紹介されているオープンソースのウェブサーバで、処理速度が高いことやメモリ消費量を抑えやすいことが特徴の Web サーバである。主に静的コンテンツの処理に優れているため、今回作成したウェブページ程度であれば Nginx の方が向いていると考える。Nginx のインストール手順は次に示す。

(1) 次のコマンドでインストールする。

```
sudo apt install nginx
```

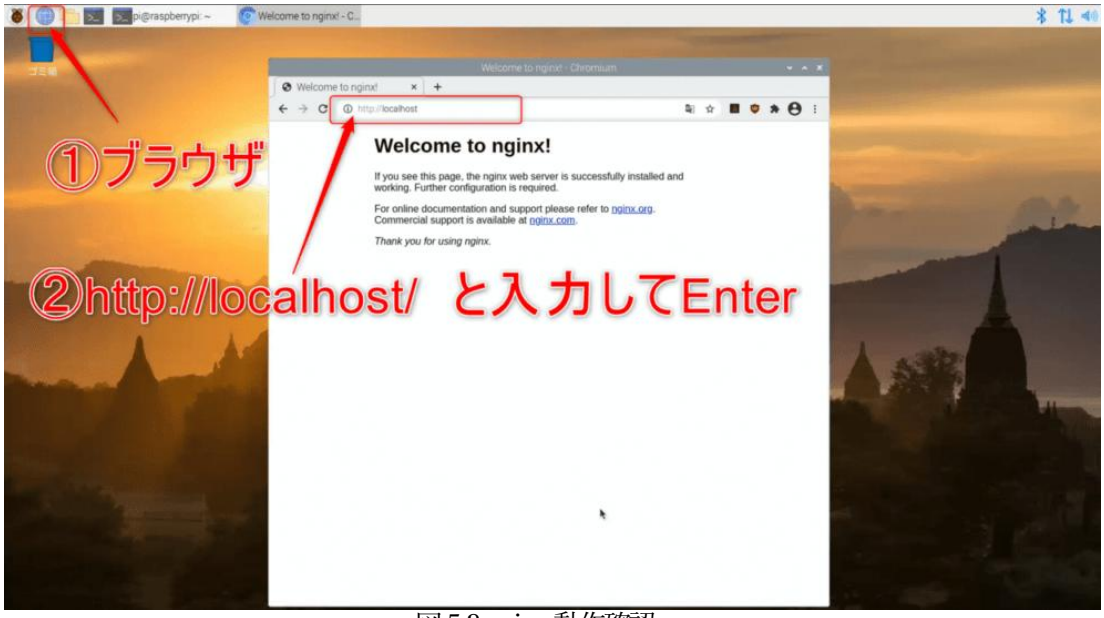

図 5.3 nginx 動作確認

(2) 事前に VSCode で作ったウェブページをドキュメントルートである/var/www/html に貼り付けると作ったページを表示させることができる。

(3) IP アドレスを固定することで同じ URL でアクセスできるようにできるが、テストの段階で正常に動作しなくなることがあったため、本研究では行っていない。

第 6 章　データ確認ページについて

6.1 確認ページの構成

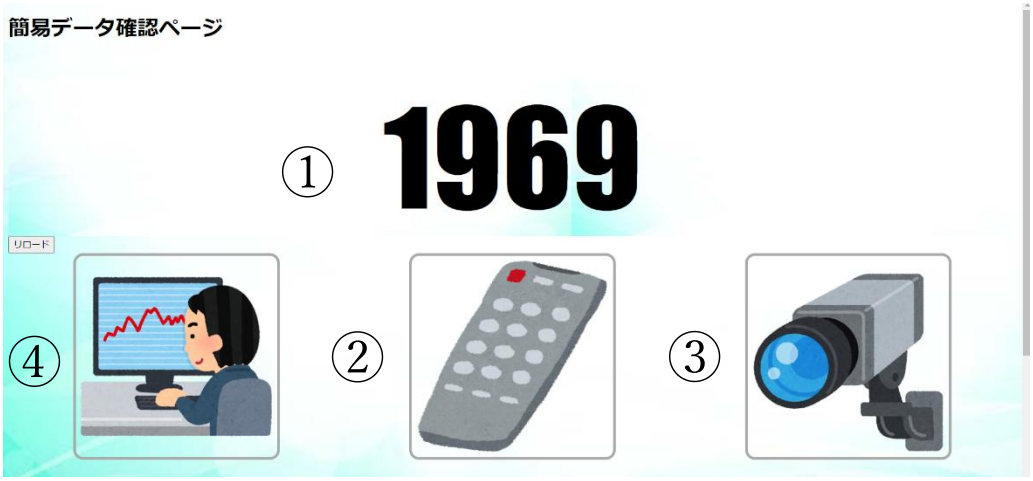

図 6.1 実際のパソコン側確認ページ

＜機能一覧＞

① 水分値の表示

M5Stick C Plus からポンプユニットで測定した水分量を Google スプレッドシートに送信し、スプレッドシートから JavaScript 経由でデータの表示をしている。

② グラフの確認ボタン

第 4 章 4.3.1 で説明した ambient にアクセスできるようになっている。

③ 遠隔操作ボタン

第 4 章 4.3.2 で説明した遠隔操作ページにアクセスできるようになっている。

④ 監視カメラボタン

第 5 章 5.2 で説明したカメラのストリームにアクセスができるようになっている。

6.2 水分値の表示について

6.2.1　スプレッドシートの設定

Web ページに直接水分量が表示できるようにするために、Google スプレッドシート（以下、「スプレッドシート」という。）にデータを格納して JavaScript でセル内データを取得して HTML に反映させるやり方で行った。そのためにスプレッドシートの拡張機能である Google Apps Script(以下、「GAS」という)を利用した。

GAS は、ひとことで言えば Google が提供する各種サービスの自動化／連携を行うためのローコード開発ツールである。GAS を使うと、Gmail やカレンダー、Google スプレッドシート、Google ドライブなど、Google が提供する様々なサービス上で処理を自動化したり、
複数のサービスを連携させたりすることができる。書き方は JavaScript に似ている。

利用するためには、次の図 6.2.1 の拡張機能から「Apps Script」選択する。

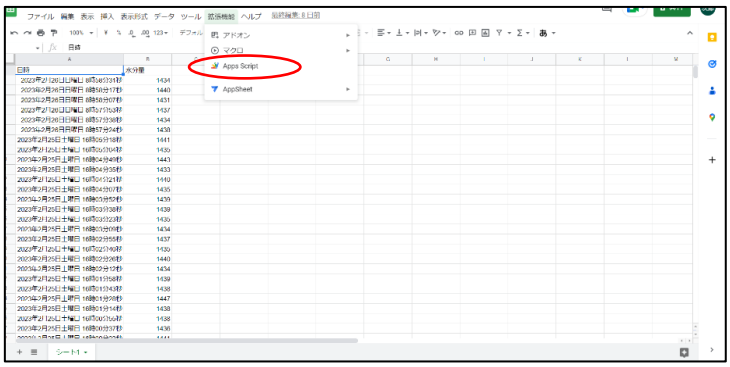

図 6.2.1GAS 導入仕方

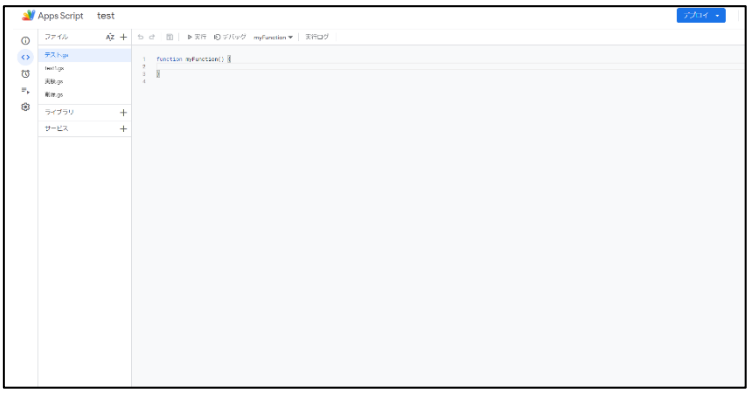

図 6.2.2GAS 初期画面

水分量をスプレッドシートに書き込むコードは次に示す。

```
function doPost(e) {
var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('シート 1');
var params = JSON.parse(e.postData.getDataAsString());
var status = params.status;

// データをシートに追加
sheet.insertRows(2,1);
sheet.getRange(2, 1).setValue(new Date());       // 受信日時を記録
sheet.getRange(2, 2).setValue(status);           // 測定値を抜き出してを記録
  }
```

コード 6.2.1 スプレッドシート書き込み

＜主なコードの説明＞

本研究の場合（2, 1）に書き込まれた日時を入れ、（2, 2）には水分量を入れるようになっている。（コード中、注釈が付いている箇所）

SpreadsheetApp.getActiveSpreadsheet().getSheetByName でシートの取得を行っている。JSON.parse では文字列を JSON として解析し、値やオブジェクトを構築できるものであり、

内容には M5Stick C Plus から送られてきたデータが含まれている。データの中の水分量を抜き出し、表示を行っている。sheet.getRange().setValue()は、シートのセルに書き込む処理を担っており、getRange で列と行を指定 setValue で書き込む内容を指定することができる。

スプレッドシートに書き込むコードは次に示す

```
function getData() {
  const spreadsheet = SpreadsheetApp.getActiveSpreadsheet()
  const sheet = spreadsheet.getActiveSheet()
  const range = sheet.getDataRange().getValues()
  const values = range
  const data = values.map(row => {
    let col = 0
    return {
      id: row[col++],
      dataw: row[col++],
    }
  })
  console.log(data)
  return data
}
```

コード 6.2.2 スプレッドシート書き込み

<主なコードの説明>

スプレッドシートとセルの全データを取得し、A 列のデータを id、B 列のデータを dataw としてコンソール上に表示させている。

sheet.getDataRange().getValues()では、シート上にデータが存在する範囲の取得をする。
そのため、送られてきた水分量のデータのすべてが対象として取られる。

次に JSON としてレスポンスする処理のコードを示す。

```
function doGet() {
  const data = getData()
  const response = ContentService.createTextOutput()
  response.setMimeType(MimeType.JSON)
  response.setContent(JSON.stringify(data))
  return response
}
```

コード 6.2.3 JSON としてレスポンスする処理

＜主なコードの説明＞

前段階で取得したデータを JSON 形式としてレスポンスするようにしている。これら 2 つの関数を一つのファイルにまとめてデプロイを行う仕様である。

デプロイとは「作成したアプリケーションの公開をすること」を指す。運用環境に配置・展開して実用に供することも指す場合がある。 GAS で、デプロイを行うと URL が生成され、 その URL にアクセスすると、GAS で作成されたアプリケーションの利用が可能になる。作成手順は以下に示す。

(1) 新しいデプロイの作成を押しデプロイの作成を開始

GAS の右上の「デプロイ」を選択すると 3 つの項目のメニューが現れる。

その中の「新しいデプロイ」を選択する。下のデプロイ管理では作成したデプロイの設定の変更を行うことや作成された URL の再表示等が行うことができ、テストでは、作成したスクリプトのテストを行うことができる。

図 6.2.3 ウェブアプリ作成 1

(2) 種類の選択とアプリにアクセスできるユーザの指定

公開するものの設定を行う。種類としてウェブアプリ、API、アドオン、ライブラリがあり、本研究はウェブアプリとして公開した。

次に、アクセスできるユーザの設定を行った。デフォルトでは、自分のみになっているが対象を全ての人に変更した。全ての人にしておかないと M5Stick C Plus からのデータをシートに書き込めず、後述する JavaScript においてもデータを取り出しができないため、この設定は必要となる。

しかし、このアクセスユーザの設定は学校の Google アカウントでは管理権限で許可されておらず、使用することができないため、別途 Google アカウント（捨てアカウント）で作業する必要がある。

設定が完了したらデプロイボタンを押し、Web アプリの URL を作成する。こうして発行された Web アプリの URL を利用して Web ページに数値を反映させる。

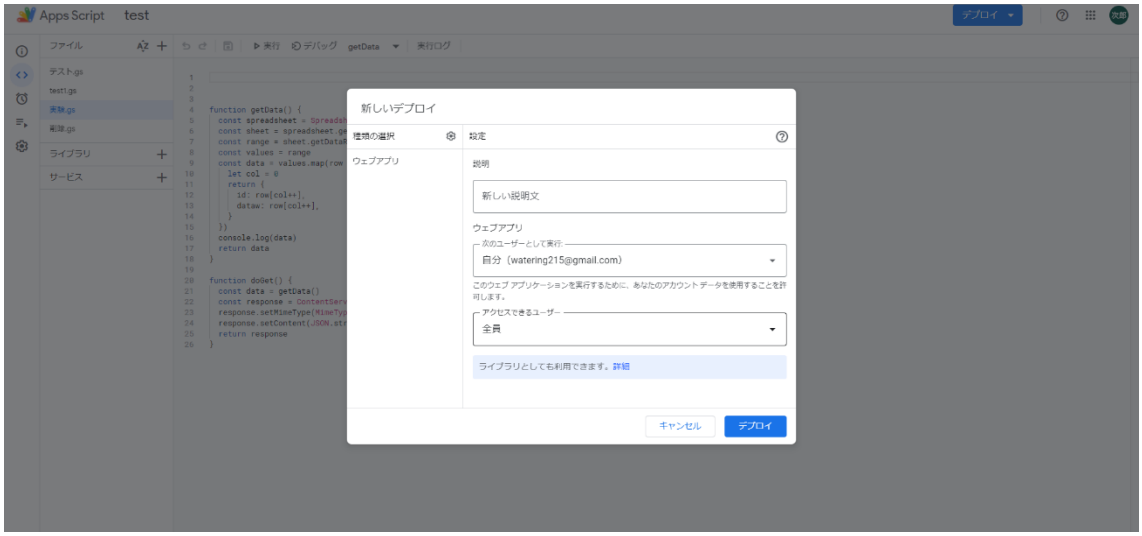

図 6.2.4 ウェブアプリ作成 2

6.2.2　JavaScript でデータの取得

第 6 章 6.2.1 で作成した Web アプリを用いて水分量を取得し、HTML に反映するようにした。

本研究では fetch メソッドを使用した。

Fetch メソッドは非同期通信で HTTP リクエストを発行し、指定された URL のリソースを取得する。この fetch メソッドを用いて第 6 章 6.2.1 でまとめたデータを取り出し、最新の値であるセル(2,2)の値だけを取り出すようにしている。次のコードは本研究で作成した水分値の表示である。

```
const url = 'https://script.google.com/macros/s/abcd……..'

fetch(url)
  .then((res) => res.json())
  .then((data) => {
    const a = document.createElement('a')
  a.textContent =  data[1].dataw

  document.getElementById('data_w').appendChild(a)
  console.log(data[1])
    })
```

コード 6.2.4 水分値の表示

＜主なコードの説明＞

URL のアプリケーションでまとめられているデータを取り出し、新しく作成する。

要素にデータの中にある水分値だけを取りだすようにしている。Data[1].dataw と指定することによって data[1](一番新しい値)の dataw(水分値)だけを取り出すようにしている。document.getElementById('data_w').appendChild(a)では、id が data_w に子要素 a を追加する処理を行っている

第 7 章　考察

今回の場合小規模な自動水やり機だったが M5StickCPlus を使えば大規模にもできると考えた。実際に運用する場合は、水分値を測定する位置や水やりの仕方の改善が必要になってくると考えた。今回は行えなかったが、実際に植物を植えて動作するか、水やりを行って育つのかを検証をしていくべきだと感じた。

本研究で作成した自動水やり機能付きプランターだったが、当初は M5StickCPlus で自動水やりと LINE 通知がある便利なプランターというシンプルな作品を作る計画であったが作業を行っていくうちに機能が物足りないと感じ、機能が増えていき最終的な形になった。不必要な機能はあまりなく便利なものができたと考えている。自動水やり、プッシュ通知、グラフの作成などの機能がその例である。

初めに中間発表まで作成したプランターでは、基本的な機能（自動水やりと LINE 通知）は完成していた状態ではあったが私自身はもの足りないと感じ、機能を実装していった。卒業研究発表会までには主に既存の機能に追加する形で私自身が納得するような作品作りに取り組んだ。その結果が遠隔操作、グラフの作成、監視カメラ、ウェブページの実装である。グラフ作成では、ambient というサービスを利用してグラフを作成した。しかし、作成後にスプレッドシートでグラフを作成するべきだったと感じた。ウェブページの作成時に利用した GAS でグラフを作成してそのグラフデータが送ることができることが分かったためである。今回は、ambient でよいが次回同じようなモノを作るときには、スプレッドシートでグラフ作成を行うやり方を採用したと思った。

監視カメラは、本来であれば M5StickV で行う予定であったが時間が足りず Raspberry Pi で実装することになったが監視カメラだけでなく Web カメラの機能だけでなく作成した
ウェブページのサーバーとして利用することができ結果よかったと感じている。
実装させるとすると送るデータの量を減らすと実装できると考えている。
産技短までには、実際に使用できるまで仕上げを行った、結果として本来作成したい作品とは少し違う形ではあったがよい作品を作り上げることができた。監視カメラのみ M5StickCPlus で実装できなかったため研究が引き継がれるような場合動作するようになればよいと考えている。

第8章　終わりに

本研究の当初の目的の通り自動みずやり機能、通信機能（通知や水分値データ送信）を
用いたプランターを予定どおり作成することができた。卒業研究の期間を通して計画通りに研究が進むことができたと考えている。機能の実装がうまくいかず妥協して別の形で実装した機能もあったがそれがよい方向に進んだこともあった。様々な試行錯誤があってできた作品は私自身が納得できるようになってよかったと感じている。

本研究で使用した技術やサービスは別の形でも利用できると考えているので自分で今後作成する際には再度利用して生きたい。LINE Notify の通知機能も IFTTT の自動送信機能などは日常生活の中でも使えると考えているので活用していきたい。

また、今回作成したプランターはまだ、スマートとは言えない見た目のため再度作成する際は見た目もよくしながら作成していきたい。Raspberry Pi で実装させた監視カメラを
M5Stick V で実装させることができれば規模を縮小できるのではないかと考えている。

今回の研究ではできなかったが、M5StickCPlus で複数ユニットの同時制御を行うことができれば規模の縮小ができるとも考えている。

そしてバラバラになっていた機能をなるべく一つに集約して作成していくべきだとも考えている。研究を進めていく中でスプレッドシートを使う機会があり、使用するとできることが多く一つのサービスで集約することに気づき始めから使えばよかったと考えることがあった。しかし、使用したことのないサービスを利用する機会となってよかったと考えている。

本研究では、いままでの授業で学んだことや新しく学ぶことの両方をふんだんに使うことができたと考えている。授業で学んだ通信やウェブページの作成、スクリプトの作成と新しく学んだ Micro Python を生かすことのできた研究だと考えている。

本研究では、ハードウェア、ソフトウェアの知識、技術を用いることになり、学べる事が多かったと感じたので趣味で同じようなものを作成するときや仕事に生かしていき今回行った研究を活用していきたいと考えました。

参考文献

参考サイト

「トマト栽培をスマート化しよう」

https://tkrel.com/iot-planter

「LINE Notify を利用して UIFlow のプログラムで LINE に通知を送る」

https://qiita.com/youtoy/items/76586479c2d4c5893c5b#ifttt%E3%82%92%E4%BD%BF%E3%82%8F%E3%81%AA%E3%81%84%E5%AE%9F%E8%A3%85%E3%81%AE%E4%BE%8B

「M5StickC で小型環境センサ端末を作る」

https://ambidata.io/samples/m5stack/m5sitckc/#:~:text=Wi%2DFi%20%E3%81%AB%E6%8E%A5%E7%B6%9A%E3%81%97%E3%81%A6%E3%80%81%E3%83%87%E3%83%BC%E3%82%BF%E3%82%92%E3%82%AF%E3%83%A9%E3%82%A6%E3%83%89%E3%81%AB%E9%80%81%E4%BF%A1%E3%81%99%E3%82%8B

「UnitV と M5StickC で動画ストリーミングを堪能」

https://homemadegarbage.com/ai13

【ラズベリーパイ】監視カメラの作り方｜Python でカメラモジュールを自在に操作

https://sozorablog.com/camera_shooting/

ラズベリーパイで Web サーバーを動かす（nginx 編）

https://www.fabshop.jp/webserver-nginx/

Raspberry Pi で Motion を使って監視カメラを作ってみよう

https://ponkichi.blog/motion/

M5Stack による自動給水機の作成

https://zenn.dev/tototo/articles/4fbf581b33e984

Google スプレッドシートを GAS と JavaScript で取得してサイトに表示する方法

https://ayumitk.com/ja/blog/fetch-and-display-google-sheets-data-on-your-website-with-gas-and-javascript/#google%E3%82%B9%E3%83%97%E3%83%AC%E3%83%83%E3%83%89%E3%82%B7%E3%83%BC%E3%83%88%E3%82%92%E4%BD%9C%E6%88%90

GAS の利用　～Web アプリでスプレッドシートの値を表示する

https://digita-l-ocal.tech/?p=521

【M5Stack でデータ解析】M5Stack で取得したデータを、Google スプレットシートへ書き込む

https://knt60345blog.com/m5stack-googlespreadsheet/#toc10

参考書籍

「M5Stack & M5Stick C ではじめる IoT 入門」

株式会社アイエンター 高馬 宏典 著

出版社：シーアンドアール研究所