カンヨウちゃん

香川高等専門学校　3 年　桑島雄介

泉凜太郎

中村駿平

樫裕翔

宮下大空

指導教員　村上幸一

目次

1. 製作背景と目的

2020 年から世界中に流行している新型コロナウイルスの影響により外出しなくなり家に居ることが多くなった. コロナ渦で家に居る時でも癒しを求めて育てやすい**観葉植物**が注目され始めた. 育てやすい観葉植物ではあるが水をあげ,適切な温湿度にしなければ枯れてしまう. そこで観葉植物が注目されている今,**誰でも簡単に少しでも長く育てる**ために温湿度・気温をスマートフォンに知らせ,自動で水やりをする装置を作りたいと思った.

2. 使用機器

(1) Raspberry Pi/Zero

Raspberry Pi はイギリスのラズベリーパイ財団が開発したシングルボードコンピュータである. 今回は RPZ-IR-Sensor と WayinTop 自動水やり装置,静電容量式土壌水分センサーを使用するために利用した.

図 2.1 Raspberry Pi/Zero

(2) RPZ-IR-Sensor

動作をプログラミング可能な,Raspberry Pi 用ホーム IoT 拡張ボードである. ケーブル経由で BME280 を接続することで温度,湿度,気圧を測定することができる.

図 2.2 RPZ-IR-Sensor

(3) WayinTop 自動水やり装置

土壌湿度センサーと小型ウォーターポンプ,ホース,水の入ったバケツ,リレーモジュール,AD コンバータを組み合わせて土が乾いたら水を出す装置である.

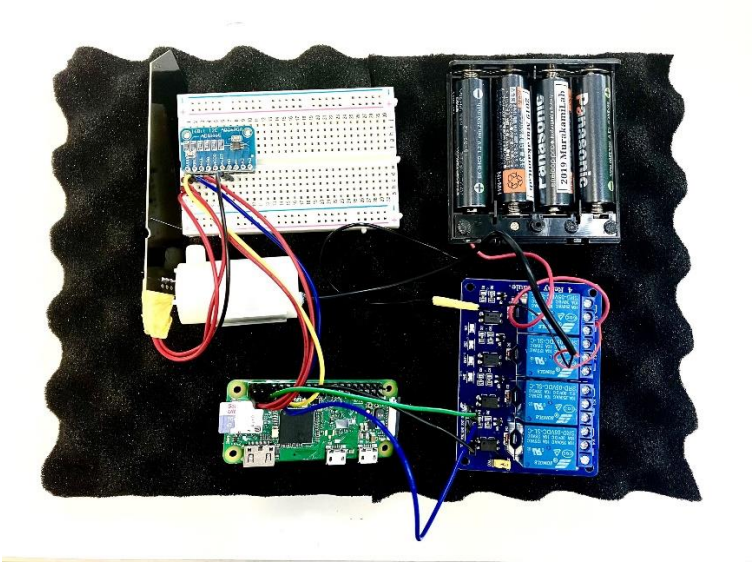

図 2.3 WayinTop 自動水やり装置

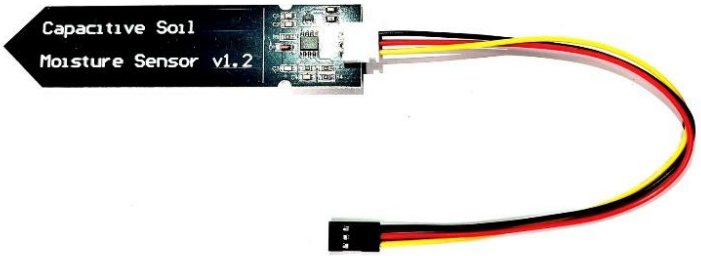

図 2.4 静電容量式土壌水分センサー

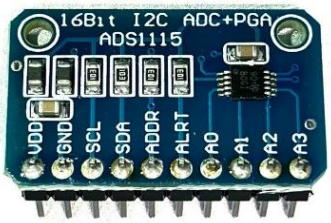

図 2.5 AD コンバータ

(4) サンスベリア,ガジュマル,クロトン

香川高専(高松キャンパス)の近くにある西村ジョイで購入した観葉植物である. 温湿度・気圧を測定して,水やりを行うために使用した.

図 2.6 観葉植物

各観葉植物の特徴

4.1 サンスベリア

・空気浄化の効果が高いキジカクシ科の植物である.

置き場所: 明るく,日陰で暖かい場所.

水やり: 乾燥には強く,過湿を嫌うため適度に水やりを行う.

4.2 カジュマル

・独特な形をした太い幹でクワ科イチジク属の植物である.

置き場所: 直射日光を避けた日光がよく当たる場所.

水やり: 高温多湿に強い性質を持っているため冬以外は土の表面が乾きだしていたら水を与える.

4.3 クロトン

・カラフルな葉色と葉形の変化が魅力なトウダイグサ科の植物である.

置き場所: 日当たりの良い場所.

水やり: 一年を通して霧吹きで水を与える. 生育期の5～10月は土の表面が乾きだしたら水をよく与える.

(5) MOUSE PC

プログラムを作成するため使用した.

3. システムの概要

「カンヨウちゃん」は適切な温湿度ではない時に LINE に通知され,また植物の水分量を測定し,その水分量に応じて自動的に水やりをすることができる.

各センサーを動作させるために Tera Term を使用した.

(1) 温湿度・気圧

Raspberry Pi と BME280 を接続するために I2C を使用した. I2C とは Raspberry Pi とセンサーなどの IC との間で通信するための規格のことである. 図 3.1 に I2C を使用したセンサーとの通信の概要を示す.

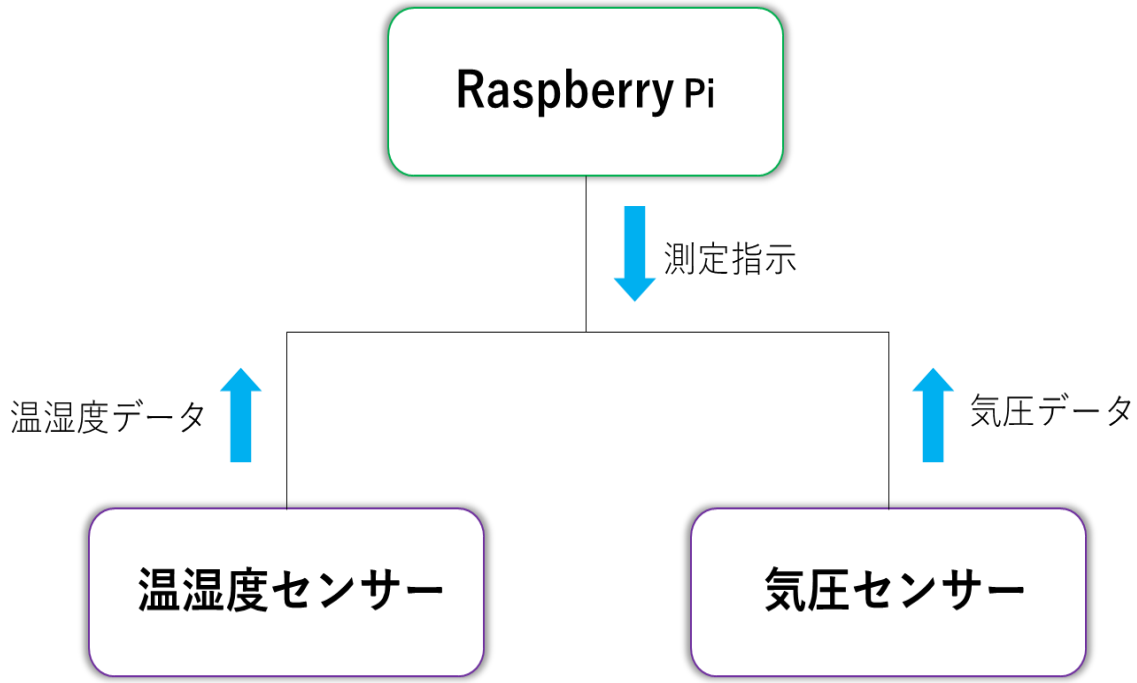

図 3.1 I2C を使用したセンサーとの通信

図 3.2 に作成したプログラムを示す. このプログラムはセンサーから受け取った気象データをもとに各植物に適した育成環境になっているかを判断し,問題がある場合スマートフォンに通知を送るプログラムである. スマートフォンへの通知には LINE Notify を使用した. LINE Notify とは LINE が提供する公式アカウント「LINE Notify」から通知を受信する事ができるサービスである.

プログラムの詳しい内容を解説する. 使用したモジュールは cgsensor,time,requests である. まず 7 行目では,変数 name に標準入力により植物の名前を代入している. 植物ごとに適した環境が異なるので測定する植物を区別するためである.

次に while 文内のプログラムを解説する. While には引数として True を渡すことでプログラムを終了するまで while 文内の内容を繰り返すようにしている. まず,def 文で read,read2,read3 という関数を作成した. read は戻り値としてセンサで測定した温度を,read2 は湿度を,read3 は気圧を返す関数である. プログラム

が動いているかの確認として read 関数の戻り値を標準出力するようにしている．次に変数 a,b,c にそれぞれ read の戻り値,read2 の戻り値,read3 の戻り値を代入している．次に,def 文で main 関数を作成した. main 関数は受けとった気象データに応じた通知をスマートフォンに送る関数である．次に main 関数を動かし,timesleep 関数で 15 秒間プログラムを停止することで 15 秒間隔での気象データの測定を行っている．

main 関数の詳細を以下に示す. main 関数では,まず,url 変数に LINE Notify の url を代入し,token 変数にラインから送られたトークン名を代入している．そして辞書型である headers の value の値として token 変数を使用している．次に,if 文によって name 変数の値に応じて動かすプログラムを選択している．ここでは観測対象がカジュマルの場合を例に解説する．カジュマルの場合は,まず if 文によって a 変数の値,つまり温度が適正温度(20~30 度)から外れているかを判断する．外れていた場合は通知する文字列を message 変数に代入し,message 変数を辞書型である payload の value の値として代入する．そして requests.post メソッドのオプションとして url 変数と辞書型の headers,payload を使用して通知を送っている．

```
import cgsensor
import time
#coding:UTF-8
import requests

print('植物名を入力してください。->')
name = input()

while True:
        def read():
                bme280=cgsensor.BME280(i2c_addr=0x76)
                bme280.forced()
                return bme280.temperature
        def read2():
                bme280=cgsensor.BME280(i2c_addr=0x76)
                bme280.forced()
                return bme280.humidity
        def read3():
                bme280=cgsensor.BME280(i2c_addr=0x76)
                bme280.forced()
                return bme280.pressure

        if __name__=='__main__':
                print(read())
                a = read()
                b = read2()
                c = read3()
        def main():
                url = "https://notify-api.line.me/api/notify"
                token = "pSd75JPWc9gbzZcrcpt4I12UBwqBzjPv06MgnMwu9UI"
                headers = {"Authorization" : "Bearer "+ token}
                #message = str(a) + '度' + str(b) + '%' + str(c) + 'hPa'
                #payload = {"message" :  message}
                #r = requests.post(url ,headers = headers ,params=payload)
```

```
        if name == 'カジュマル':
                if a>30 or 20>a:

                        if a>30:
                                message ='温度がカジュマルの適正より'+ str(a-30) + '度高いわよ'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)
                        if 20>a:
                                message = '温度がカジュマルの適正より' + str(20-a) + '度低いわよ'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)

                if 70>b:
                        message ='湿度がカジュマルの適正より'+ str(70-b) + '%低いわよ'
                        payload = {"message" :  message}
                        r = requests.post(url ,headers = headers ,params=payload)

#               if __name__ == '__main__':
#                       main()
#                       time.sleep(15)
```

```
        elif name == 'サンスベリア':
                if a>25 or 20>a:
                        if a>25:
                                message ='温度がサンスベリアの適正より'+ str(a-25) + '度高いよ'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)
                        if 20>a:
                                message = '温度がサンスベリアの適正より' + str(20-a) + '度低いね'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)

                if b>60 or 50>b:
                        if a>60:
                                message ='湿度がサンスベリアの適正より'+ str(a-60) + '%高いよ'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)
                        if 50>a:
                                message = '湿度がサンスベリアの適正より' + str(50-a) + '%低いね'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)


#               if __name__ == '__main__':
#                       main()
```

```
        elif name == 'クロトン':
                if 10>a:
                        message = '温度がクロトンの適正より' + str(10-a) + '度低いです'
                        payload = {"message" :  message}
                        r = requests.post(url ,headers = headers ,params=payload)

                if b>80 or 40>b:

                        if b>80:
                                message ='湿度がクロトンの適正より'+ str(b-80) + '%高いです'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)
                        if 40>b:
                                message = '湿度がクロトンの適正より' + str(40-b) + '%低いです'
                                payload = {"message" :  message}
                                r = requests.post(url ,headers = headers ,params=payload)



if __name__ == '__main__':
        main()
        time.sleep(15)
```

図 3.2 プログラム 1

(2) 水やり

図 3.3 に配線図を示す. WayinTop 自動水やり装置のセットだけでは Raspberry Pi と接続することができなかったため,AD コンバータを購入した. AD コンバータでセンサーから取得したアナログ値をデジタル値に変換して,ラズパイで読み取る.

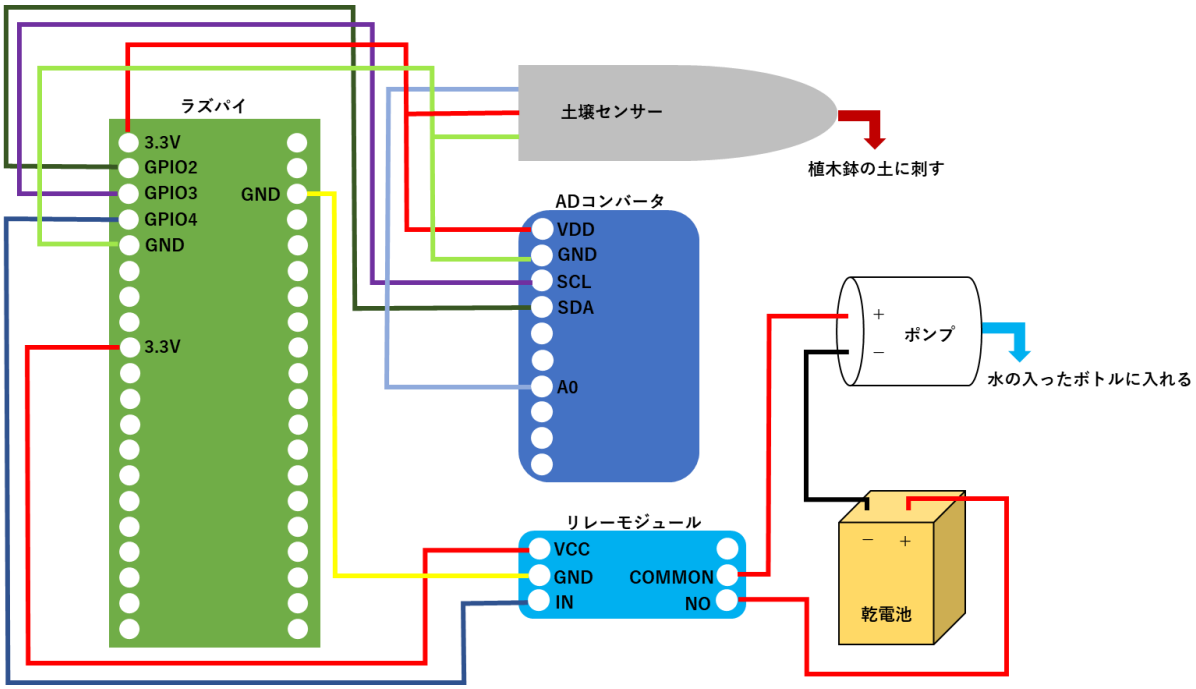

図 3.3 配線図

図 3.4 に作成したプログラムを示す. プログラムで設定した値を超えると数秒間水が出て,値以下の場合は水が止まる. 予め決めた値を超えるとずっと水が出続けるのではなく,値を超えると 1 秒間だけ出ていったん止まる. したがって,植木鉢から水が溢れ出す心配はない.

```
#!/usr/bin/python3

import RPi.GPIO as GPIO
import time
import Adafruit_ADS1x15
import math

adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
SENSOR_THRESHOLD = 15000

PIN = 4

values = [0] * 100

def setup():
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(PIN, GPIO.OUT)
        GPIO.output(PIN, GPIO.LOW)

def loop():
        while True:
                maxValue = 0
                # read some times to avoid pulse value
                for i in range(100):
                        values[i] = adc.read_adc(0, gain = GAIN)
                        if values[i] > maxValue:
                                maxValue = values[i]
                print("maxValue == " + str(maxValue))
                # if dry
                if (maxValue) > SENSOR_THRESHOLD:
                        GPIO.output(PIN, GPIO.LOW)
                        time.sleep(1.0)
                        GPIO.output(PIN,GPIO.HIGH)
                        print("Pump is ON")
                # if not dry
                else:
```

```
                        GPIO.output(PIN, GPIO.HIGH)
                        print("Pump is OFF")
                time.sleep(10.0)

def destroy():
        GPIO.setup(PIN, GPIO.IN)
        GPIO.cleanup()

if __name__ == '__main__':
        setup()
        try:
                loop()
        except KeyboardInterrupt:
                destroy()
```

図 3.4 プログラム 2

4. 結果

(1) 温湿度・気圧

図 4.1,4.2,4.3 に LINE で取得したデータを示す. 図 4.4 に測定中の写真を示す.

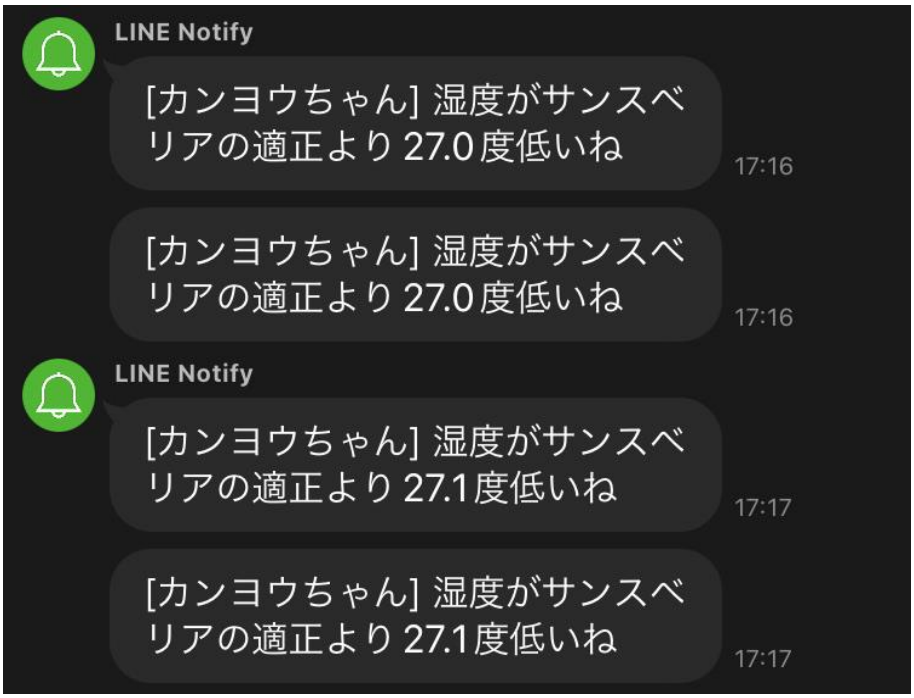

図 4.1 サンスベリアデータ

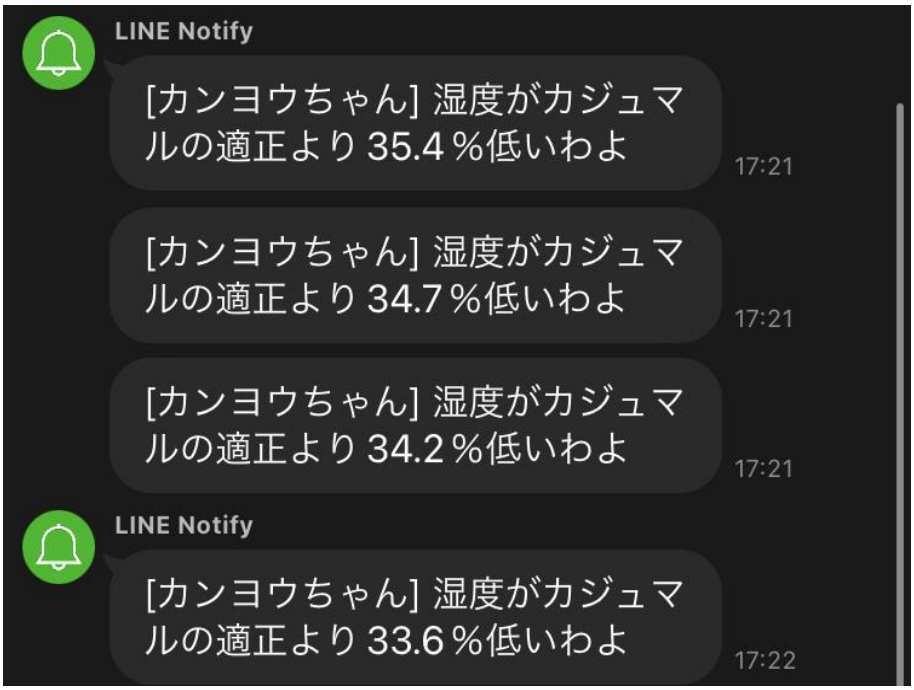

図 4.2 カジュマルデータ

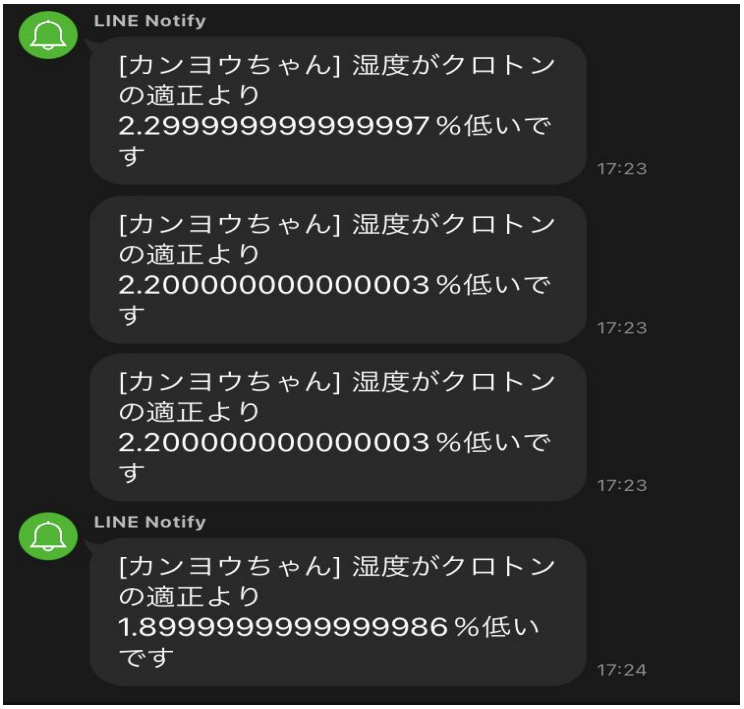

図 4.3 クロトン

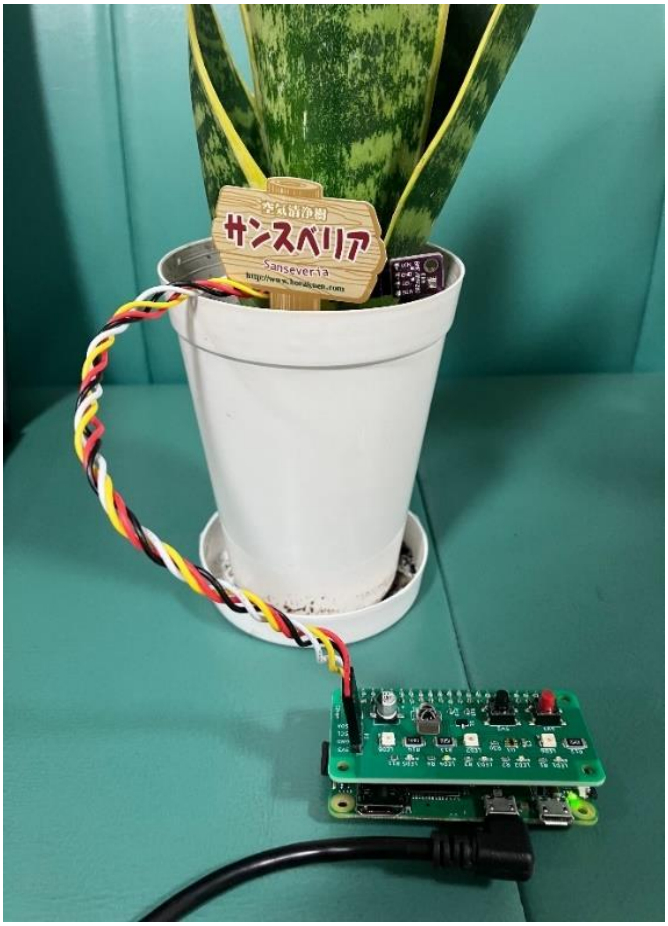

図 4.4 測定中の様子

(2) 水やり

図 4.5 に土壌センサーの値と水やりが行われている実行結果を示す.

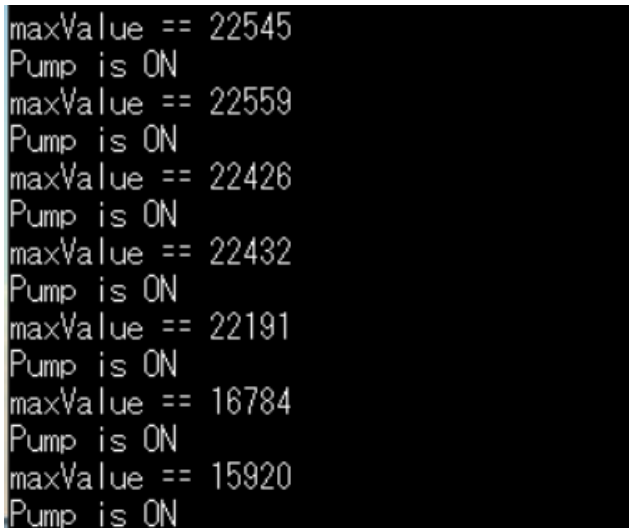

図 4.5 水やり実行結果

5. 考察

(1) 温湿度・気圧

表 5.1 に各植物の生育適温湿度を示す.表 5.1 の植物の生育適温湿度ではなかった場合に送られたメッセージが図 4.1,4.2,4.3 のようにスマートフォンに表示される. ですが, 今回作成した図 3.2 のプログラムでは様々な種類のある観葉植物全てに対応するのは非効率である. この機器の完成形としては,観葉植物ごとのデータをサーバーなどで管理して逐一データを参照できるようにすればいいと考える.

表 5.1 各植物の生育適温湿度

サンスベリア	20～25℃	50～60%
カジュマル	20～30℃	70%以上
クロトン	10℃以上	40～80%

(2) 水やり

図 4.5 から maxValue が土壌センサーの値を表示していて,15000 以上だったためポンプが ON になって水が流れている. 私たちは水やりの基準として土壌センサーの値を 15000 に設定した. 図 3.4 のプログラムで 15000 以上になると自動的に水やりが行われて,15000 以下になると水が出なくなる. 3 のシステム概要で述べた通り,15000 以上になると永遠に水が流れるのではなく 1 秒間出て水が止まる. 植木鉢から水があふれだすことがないので,簡単に観葉植物を育てられると考えられる.

6. 感想

今回は温湿度・気圧センサーと土壌センサーを別々に測定した．そのため次回からは両センサーを一体化させて一つの機器にしたいと思う．

7. 参考資料

(1) RPZ-IR-Sensor
https://www.indoorcorgielec.com/products/rpz-ir-sensor/
(2) WayinTop 自動水やり装置
https://linuxfun.org/2021/06/25/raspberry-pi-watering-system/
(3) 植物
https://www.vivahome.co.jp/green_life/hanatomidori/7gatu/01sunsuberia/default.htm
https://www.hyponex.co.jp/plantia/study/16422
https://www.oretanyamabokori.com/gaju-ookikunaran/
https://www.shuminoengei.jp/m-pc/a-page_p_detail/target_plant_code-126
http://www.yasashi.info/ku_00009g.htm