最優秀賞

賞金 20 万円

葉-℃（はーと）グラフィー
～けろっぴの葉温観測～

独立行政法人国立高等専門学校機構
鳥羽商船高等専門学校

葉ー℃（はーと）グラフィー
～けろっぴの葉温観察～
報告書

平成 28 年 10 月

鳥羽商船高等専門学校

制御情報工学科

4 年　川邊　有紗

4 年　山口　龍也

4 年　世古口　英大

5 年　大北　悠人

1. 背景

人類は、はるか昔、農業の恩恵によって一定の場所で安定した生活ができるようになり、高度な文明を発達させました。そして農業は現在においても重要な産業の一つであることに変わりはありません。近年では観葉植物や家庭菜園の流行もあり、家庭で植物を育てることはごく当たり前に行われています。しかし、気温、日射量、降水量によって大きく影響される植物の育成は大変な作業です。その上、たとえ同じ種類であっても植物ごとの健康状態が異なるため、同じ気象環境であっても、**個々の植物が受ける影響は異なります**。特に植物にとって必要不可欠な水やりは大変重要で実は難しいのです。水やりの失敗は植物を枯らす直接の原因になりますし、うまくコントロールをすれば、甘い果実を実らせることができたり、植物をより大きく成長させることができるようになります。

このように植物の健康状態を管理する上で水やりは非常に重要です。ところが、**人の目で見ただけではどの植物が水を必要としているかが、簡単に判断できません**。もちろん、土壌水分量や太陽光量から植物育成のためのデータを測るセンサはあります。しかし、いずれも環境の状態を見ているだけなので**個々の植物のどれが水を必要としているかは直接わからないのです**。

1. **植物は光合成に伴って気孔から水や酸素と共に熱を放出していること。**
2. **水分が不足すると蒸散を防ぐために気孔を閉じ、熱を放出しなくなる。**

そこで、私達は植物の持つ以上の特徴に着目し、植物繊維を水に浸しラミネートして蒸散を行えない状態の葉温が高くなりやすい葉っぱのモデル（ダミーの葉っぱ(図 1)）の温度と植物の葉温を**比較観測することでひとつひとつの植物毎の健康状態を把握できるのではないかと考えました**。

葉温は植物の水分含有量と密接な関係があることが知られており、葉は暑さに弱いため内部の温度が上昇しすぎると異常をきたします。それを防ぐ為、気孔を開いて水分を蒸発させ、その時の気化熱で内部の温度を下げています。この時の葉温と蒸散量の関係を図２に示します。また、日中は植物に太陽光が照射されている為、太陽光により葉温が高くなり、光合成に利用される可視光線も熱になって失われる可能性が高くなるのでこれも葉温を上昇させる原因となってしまいます。このように様々な気象の変化によって変化する植物の水分含有量を、植物の葉温を計測することによって植物の健康状態を把握することが

でき、最適な水やりの量、タイミングを把握できると考えています

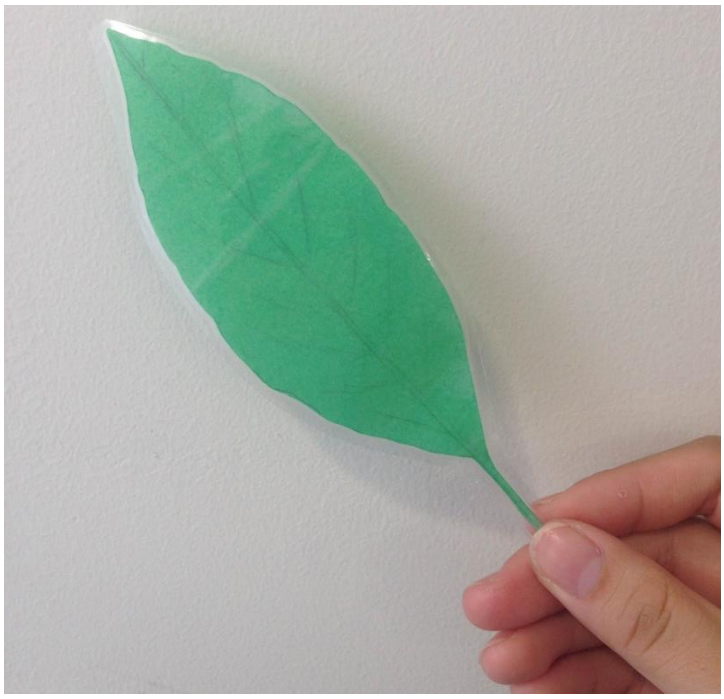


図１ ダミーの葉っぱ

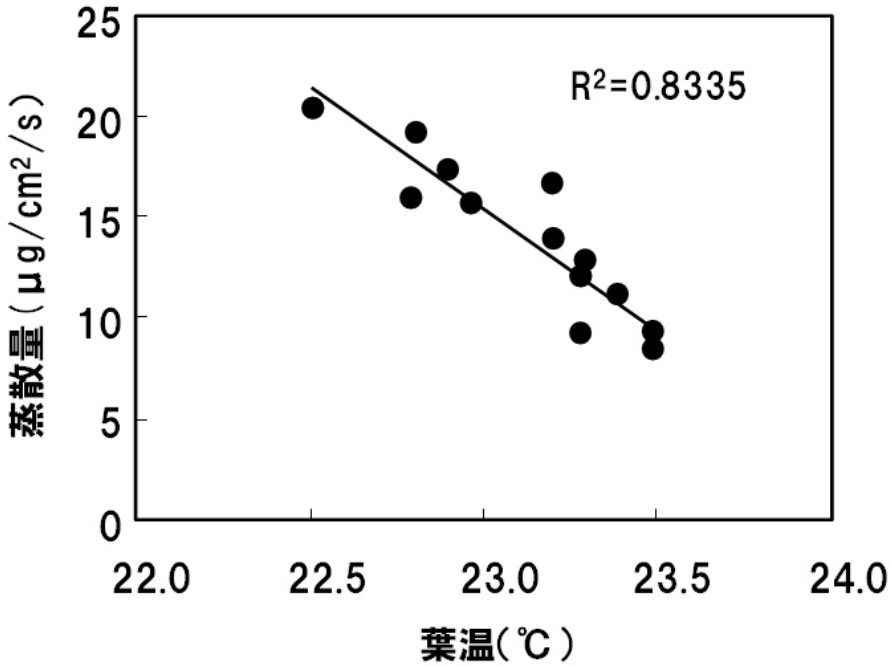


図２　葉温と蒸散量の関係[1]

2. 目的

葉-℃グラフィーは、普段人の目では観測することのできない植物の健康状態を今や殆どの人が使用しているスマートフォンを用いて観察できないかと考え、サーモグラフィーを搭載したアプリケーションを開発しました。本システムを用いて観測することにより、誰でも個々の植物の健康状態を把握することができ、植物が枯れるのを防ぐことが出来ます。また、果樹や野菜では甘い果実を得るために、与える水分量をできるだけ減らす栽培法が広く行われているので、家庭での使用だけでなく、果樹園や植物工場でも実用的に使用可能です。本システムを使用することにより身近な植物の水やりに悩むことがなくなり、さらには、農家の方の助けになることができればと思い本作品を製作しました。

3. システム

3.1 概要

葉-℃グラフィーはサーモグラフィーを用いて、植物を撮影し熱画像から植物の状態を観測するシステムです。図 3 にシステム概要を示します。

葉-℃グラフィー最大の特徴は気象条件だけでは分からない個々の植物の健康状態（水をどれだけ欲しがっているか）が撮影するだけで定量的に可視化できるということです。植物の種類や個体ごとによって適切な水やりの量は違いますが、葉温を見える化することにより水分ストレスを指数化し、植物の知識がない人でも植物の状態を容易に知ることが可能になります。(図 4)

図 3 システム構成図

図 4 水分ストレス指標表

（右にいくほど植物が水を欲しがっていることになります）

3.2機器説明

3.2.1 スマートフォン

今回は、iOS を使用し葉-℃グラフィーのアプリを開発しました。使用したスマートフォンのスペックを表 1 に示します。

表 1　スマートフォンのスペック情報

製品名	iPhone SE
寸法/重量	高さ:7.6mm/幅:58.6mm/厚さ:7.6mm/重量:113g
カラー	ローズゴールド
対応 OS	iOS 10
CPU	M9 モーションコプロセッサ(A9/1.85GHz)
メモリ	RAM:2GB
ディスプレイ	4 インチ Retina ディスプレイ(1136×640 ピクセル)
通信速度	4G LTE 経由最大 150Mbps Wi-Fi 経由最大 433Mbps
センサー	Touch ID 指紋認証/3 軸ジャイロ/加速度/近接/環境光

3.2.2 FLIR ONE for iOS

本機器ではサーモグラフィーとして、スマートフォンやタブレットを赤外線カメラに変える FLIR ONE を使用しました。FLIR ONE は、わずかな温度差を熱画像として表示することができ、非常に安価で高性能です。FLIR ONE の画像を図 5、スペックを表 2 に示します。

図 5　　FLIR ONE for iOS の画像

表 2　　FLIR ONE for iOS のスペック

製品名	FLIP ONE for iOS
デバイス互換性	Lightning コネクタ接続の iOS デバイス
カラー	ガンメタル
寸法/重量	高さ:18mm/幅：26mm/長さ：72mm
バッテリ	350mA/h
ヴィジブルカメラ	VGA
感受性	0.1℃の小単位で温度差を検出

4. 実験方法

4.1　実験 1

実験には植物の葉温を計測する為に、厚紙を葉の形に切り取り、水で湿らせラミネートし、ダミーの葉っぱを作ります。**このダミーの葉っぱは光合成活性がなく蒸散による温度の変化がないので、葉温を比較するときこの葉が基準になります**。ダミーの葉っぱを基準にする条件としては、蒸散をしていない本物の葉っぱと比較したときに葉温が近くなければなりません。そこで本物の葉っぱの裏側にワセリンを塗って蒸散ができない葉っぱに見立て、ダミーの葉っぱと比較する実験を行いました。

4.2 実験 2

植物に水をやると植物の水分含有量が増えて葉温が下がります。逆に水をやらないと水分含有量が減るのを防ぐために気孔が閉じ、葉温が上がります。この現象を FLIR ONE を使って確認しました。ダミーの葉っぱを実際に観測する生きた葉っぱと並べて、FLIR ONE で撮影をし、温度差を比較しました。観測する植物は 3 種類あり、実際に植物を育てるときと同じように、水やりのタイミングや気温、天気を考えた上で条件を変え、２時間毎に葉温を計測しました。

下記に条件と実際に観測を行った植物とその熱画像を図 6、図 7 に示します。

図 6 観測対象の葉っぱ(写真)

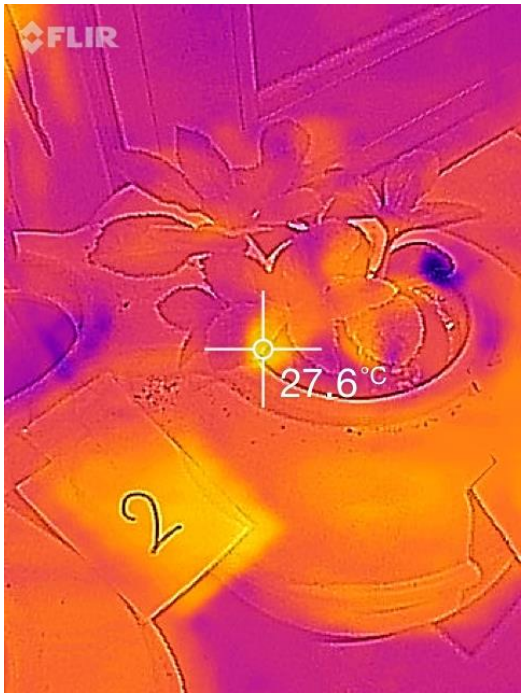


図 7 観測対象の葉っぱ(熱画像)

条件 1. １２時に水を与える。
条件 2. 葉っぱの温度がダミーの温度と同じになったら水を与える。
条件 3. 葉っぱの温度がダミーの温度より１度下回っている場合、水を与える。

5. 実験結果及び考察

5.1 実験結果 1

ワセリンを塗った葉っぱを FLIR ONE で撮影したものを図 8 に示します。

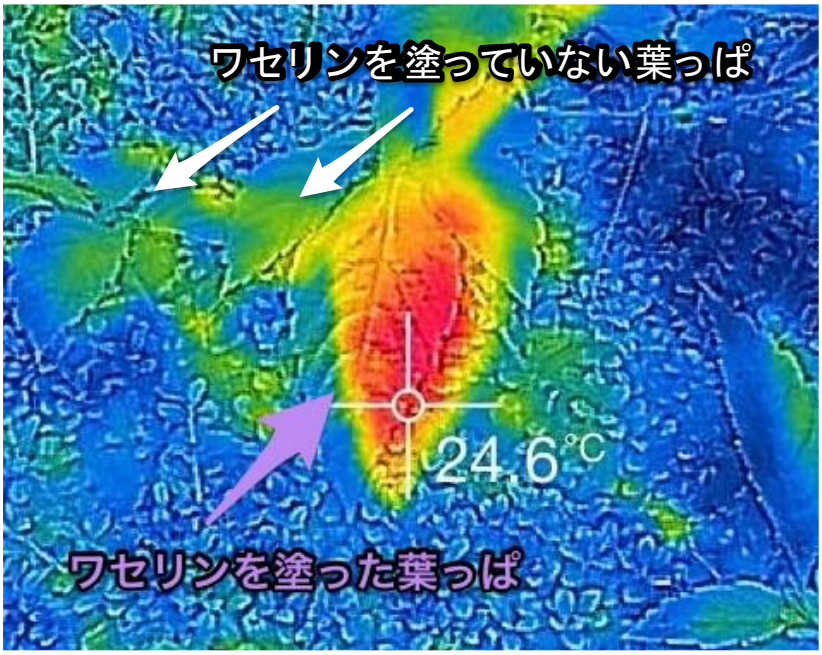


図 8 ワセリンを塗った葉っぱと塗っていない葉っぱ

図を見ると、ワセリンを塗ることによって気孔がふさがったことで蒸散を防ぎ、葉っぱの表面温度が上がっていることがわかります。この葉っぱとダミーの葉っぱを比較して FLIR ONE で撮影したものを図 9、図 10 に示します。

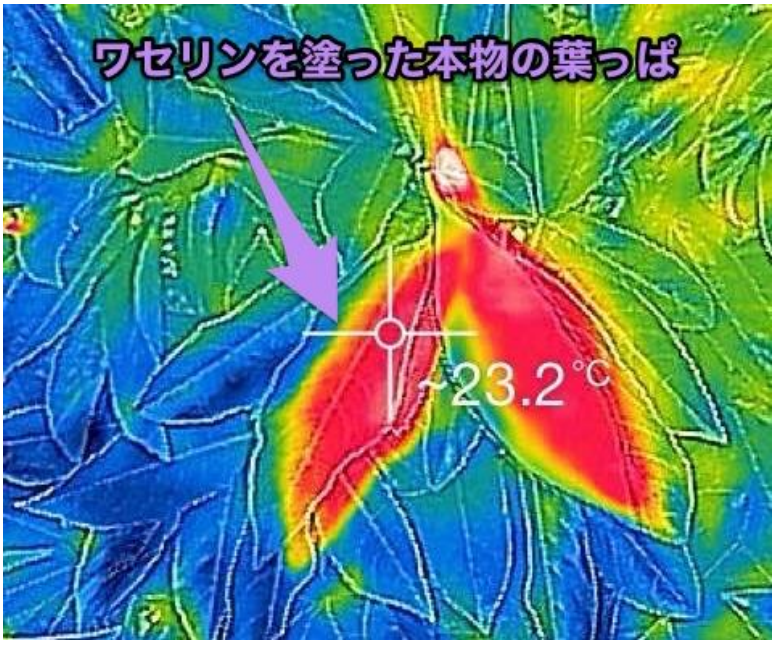


図 9　ワセリンを塗った本物の葉っぱの温度

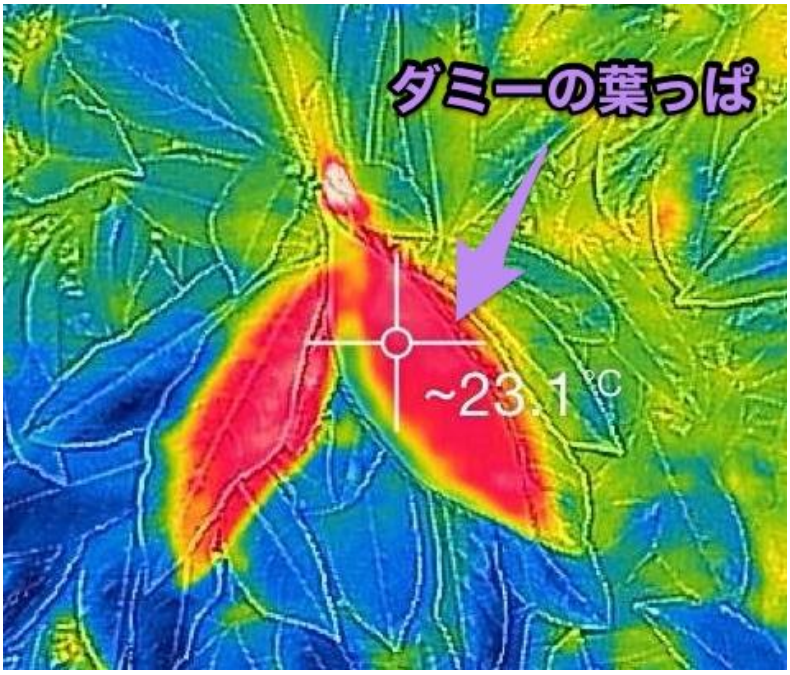


図 10　ダミーの葉っぱの温度

撮影したものを比較してみると、ほとんど同じ温度でした。

5.2 考察 1

実験結果から、ダミーの葉っぱとワセリンを塗った本物の葉っぱを比較すると、ほとんど同じ温度であったため、このダミーの葉っぱは葉温を比較するのに充分な基準になると言えます。

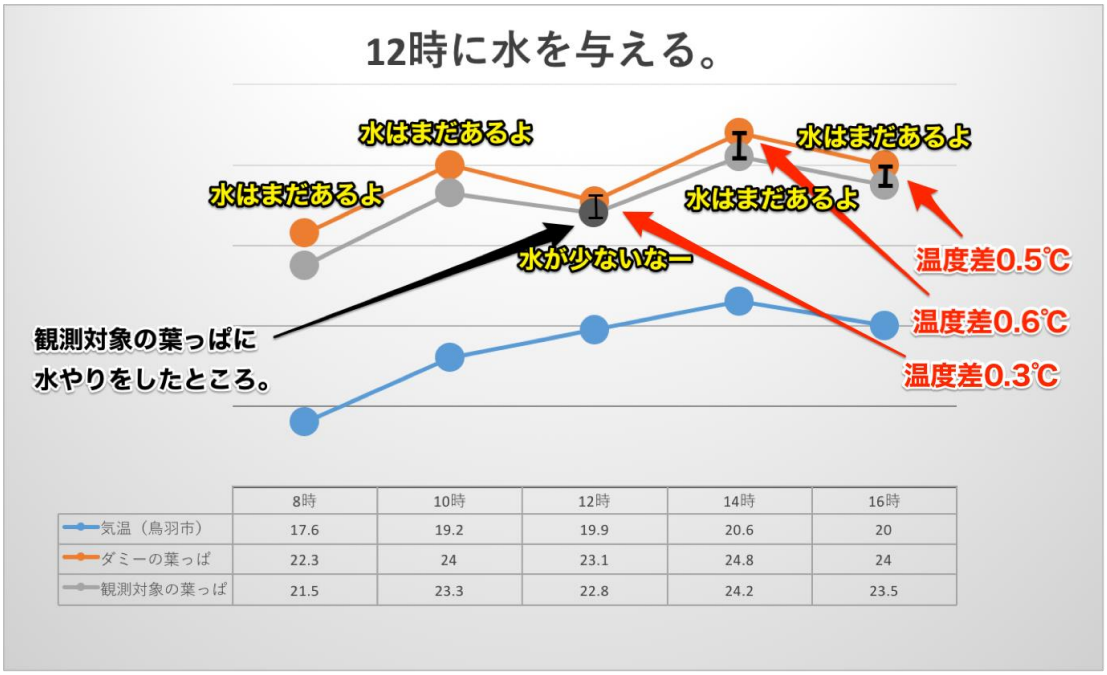
5.3 実験結果 2

次にそれぞれ種類が異なる植物を３つ用意し、それぞれの植物ごとに違う条件で葉温の温度差の比較実験を行いました。下記に実験結果を示します。

実験結果を図 11、図 12、図 13 に示します

5.3.1 条件 1 実験結果

① 12 時にはダミーとの温度が近く、水が不足していることがわかります。その時点で水やりをしてから２時間後、ダミーの葉っぱと比較対象の葉っぱの温度差が大きくなり、水が足りたことがわかります。



	8時	10時	12時	14時	16時
気温（鳥羽市）	17.6	19.2	19.9	20.6	20
ダミーの葉っぱ	22.3	24	23.1	24.8	24
観測対象の葉っぱ	21.5	23.3	22.8	24.2	23.5

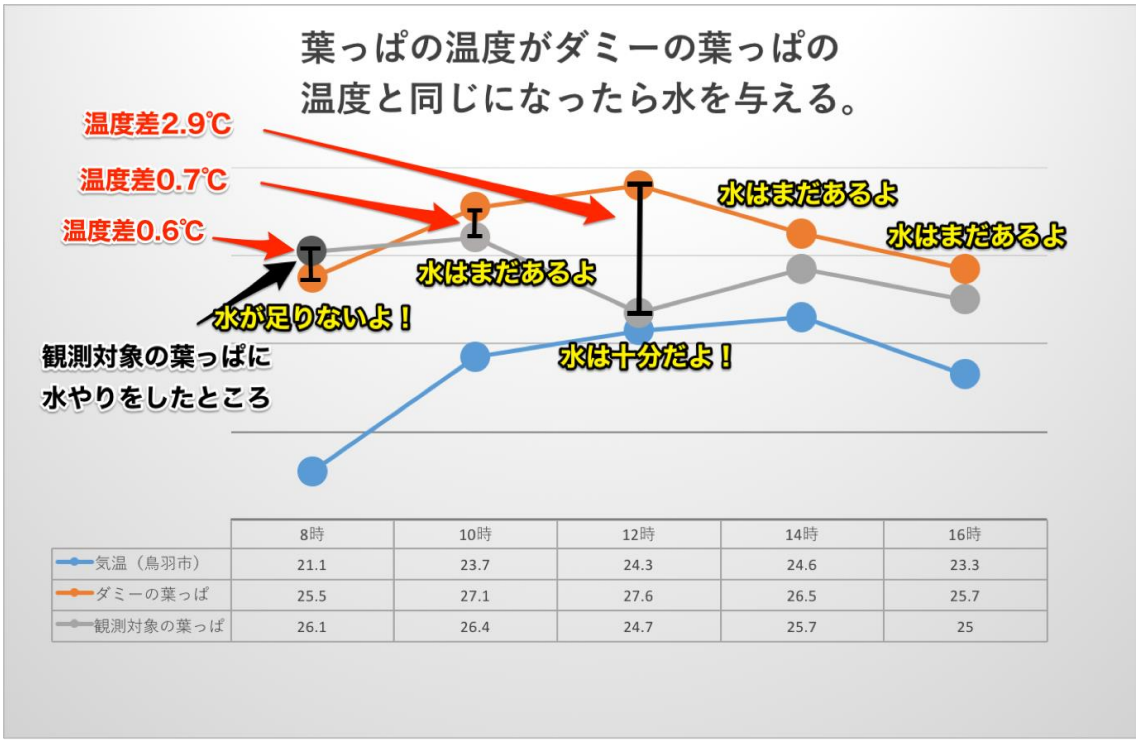
図 11　条件 1 結果

5.3.2 条件 2 実験結果

① 水やりをしてから２時間後、気温が上がったと同時にダミーの葉っぱの温度も上がりました。ダミーの葉っぱと比較対象の葉っぱの温度差は 0.7℃で温度差はあまりありませんでした。

② 水やりをしてから４時間後、ダミーの葉っぱと比較対象の葉っぱの温度差は 2.9℃でした。

③ 水やりをしてから、水分が十分になったあと植物の中で水が減っていく様子を捉えていると考えられます。



	8時	10時	12時	14時	16時
気温（鳥羽市）	21.1	23.7	24.3	24.6	23.3
ダミーの葉っぱ	25.5	27.1	27.6	26.5	25.7
観測対象の葉っぱ	26.1	26.4	24.7	25.7	25

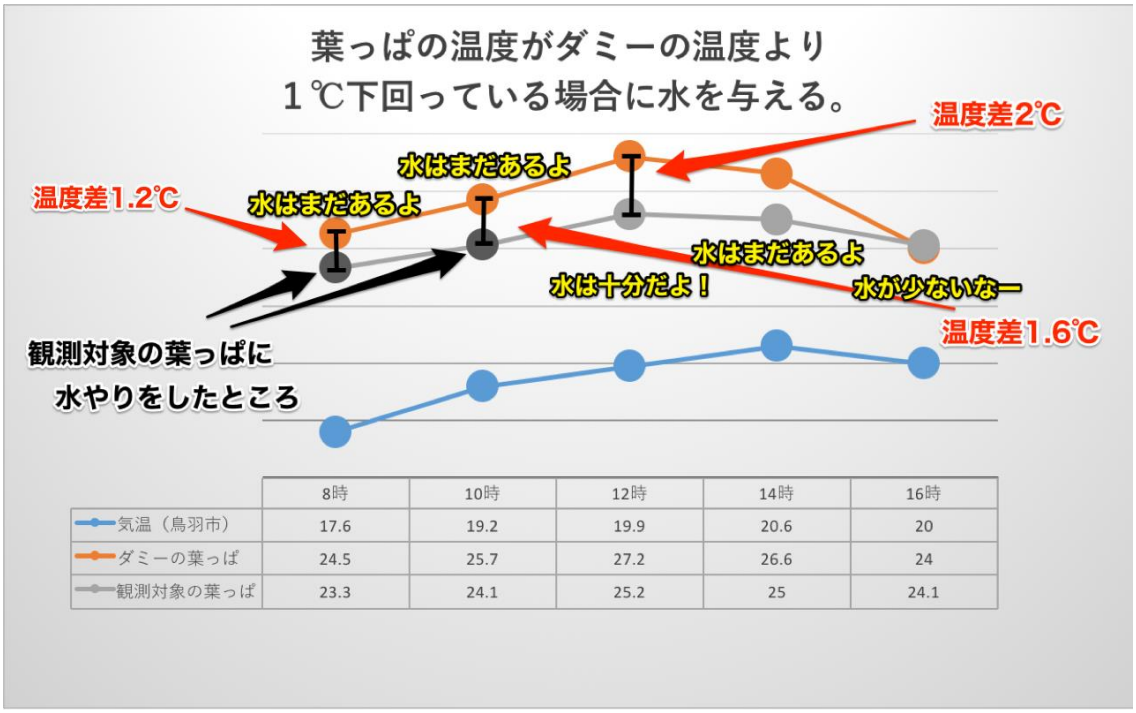
図 12 条件 2 結果

5.3.3 条件 3 実験結果

① 水やりをしてから 2 時間後、10 時の時点でダミーの葉っぱと比較対象の葉っぱの温度差は 1.6℃でした。水やりをする前と比べると、温度差が大きくなっていました。条件を飲んでいたので水をあげました。

② 水やりをしてから 2 時間後、12 時の時点でダミーの葉っぱと比較対象の葉っぱの温度差は 2℃でした。

③ この結果でも、水やり後の水分と植物の状態変化が連動していることがわかります。



	8時	10時	12時	14時	16時
気温（鳥羽市）	17.6	19.2	19.9	20.6	20
ダミーの葉っぱ	24.5	25.7	27.2	26.6	24
観測対象の葉っぱ	23.3	24.1	25.2	25	24.1

図 13 条件 3 結果

5.4 考察 2

鳥羽市の気温と比較したところ、気温が一番高くなる 12～14 時に葉温が一番高くなり、気温が低い 8 時～10 時は葉温が低いことから、葉温は温度と密接な関係があることが確認できました。本システムでは、WebAPI を通じて気象情報を取得し植物の状態判断に用いていますが、それが正しいことが示唆されました。

また、水やり後のダミーの葉温と比較対象の葉温に差がはっきり出ることが確認できました。実際に植物を育てるときと同じような条件下で比較したので、以上の結果から実際に家庭などで使用できると考えました。

6. 今後の展望

今後の展望として、本システムは家庭などの個人での使用を想定したシステムですが、大規模な農場（果樹園、植物工場）への導入も検討しています。**実際に三重県熊野市のミカン栽培を行っている果樹園からぜひ使用したいとの要望をいただいています**。また、葉温を観測することにより、カンキツ栽培における水管理にも使用可能であると考えています。カンキツの中でも温州ミカンの果実品質は樹体の水分状態に大きく左右され、適度な水分ストレスを与えることで糖度を向上させることが知られています。本システムを導入することによって、水やりの量を定量的に把握することができ、作物の品質向上に繋がると考えています。糖度と水ポテンシャルの関係を図 14 に示します。

しかし、春から夏の実が生長する時期にはたっぷりと水を与えれば良いのですが、実が成熟する晩秋から冬にかけては、収穫した翌年に樹木が衰弱してしまうので、与える水分を減らす必要があります。そういった細かい水分ストレスのコントロールも本システムを使用すれば可能になります。

最後に、私たちが作成したシステムを地域の大規模農場で導入して頂き、水やりの量を定量的に把握し水分ストレスを与えて、より美味しい果実を作るための支援を行っていき、全国へと広げていきたいと思います。

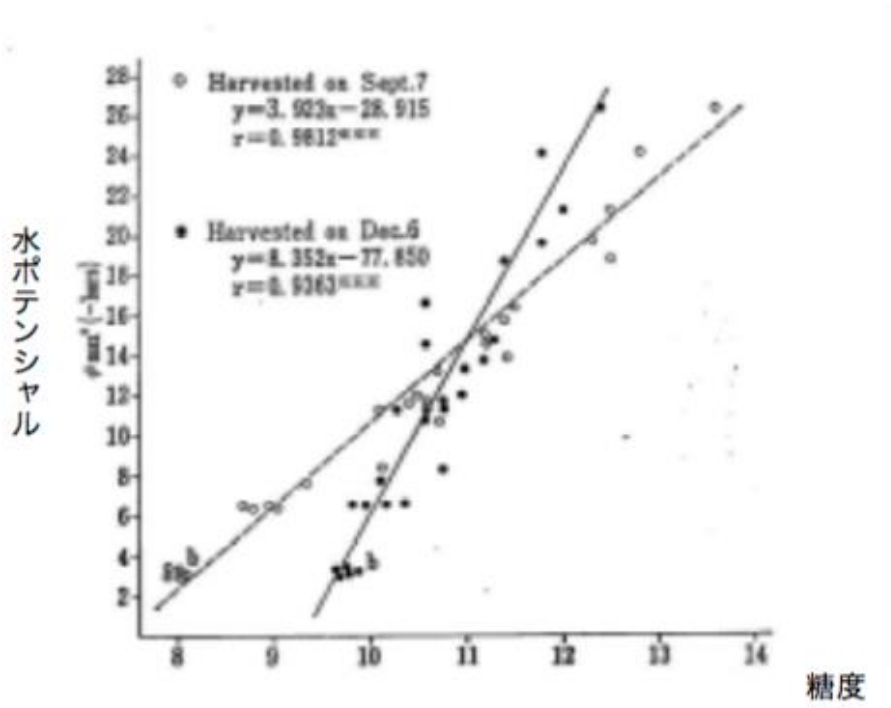


図 14 温州ミカンの糖度と水ポテンシャルの関係

<参考文献>

(1)astamuse
http://astamuse.com/ja/published/JP/No/2012161289 (2016 年 10 月 17 日)
(2)近畿中国四国農業研究センター総合研究第 2 チーム (2004)
果樹園、特にカンキツ園における水管理

優秀賞

賞金 10 万円

Starsphere Examine Satellite（成層圏観測衛星）「SES」

香川高専高等学校

高松キャンパス

Stratosphere Examine Satellite

(成層圏観測衛星)

「SES」

香川高等専門学校　宇宙開発研究同好(UKK)
メンバー　　新谷 悠真　　大西 哲
大島 風雅　　前川 雄壱
出原 寛大

指導教員　　村上 幸一

目次

1. 目的

僕たちが普段生活している地上の気温や湿度などの気象データはニュースやインターネットなどで毎日見ることができる。しかし、高高度(上空 10000m 以上)の気象データはあまり知る機会が無い。そこに注目して「空の気象はどのようなものなのか」、「地上の気象とは何が違うのか」、「雲の動きは実際にどのような動きをしているのか」など、空の気象の不思議や疑問を解決するため、バルーンを用いて空の気象を測定することにした。

2. これまでの高高度気象観測プロジェクト(Examine Project)

これまで僕たちは本コンテストにおいて、過去 3 回にわたり高高度気象観測機器を出展してきた。

① 第 2 回気象観測機器コンテスト「高層気象観測バルーン」

この時は、僕たちの先輩が初めて出場しバルーンの知識も少ない中、風船を 3 つ用いてビルの屋上から釣り竿で衛星を係留し、大気安定度を測定した。

しかし、低高度での測定だったため十分なデータが得られず実用性には欠けた。

図 2-1　高層気象観測バルーン

② 第 3 回気象観測機器コンテスト「Cloud Examine」

この回になりようやくラジオゾンデ用のバルーンを用いた気象観測を行った。高知県土佐清水市で放球を行う予定だったが、台風の影響や放球する寸前で機器のトラブルがあり係留実験だけになるという悔しい結果に終わった。

図 2-2　Cloud Examine

③ 第 4 回気象観測機器コンテスト「ガッスン from Examine Project」

昨年は測定機器のプログラムをさらに安定したものに変更、新たなセンサを追加して無事に放球することができた。しかし、機器を回収しなければデータ収得ができないのにもかかわらず、機器が回収できなかったため、一部の気象データを取得することができなかった。またもや、悔いの残る結果になってしまった。

図 2-3　ガッスン

3. ミッション

① 上空の気象を測定

温湿度センサや気圧センサを用いて、地上と上空の気象の違いを検証し考察する。

② 温室効果ガスの測定

今話題の地球温暖化の原因とも言われている温室効果ガスを二酸化炭素濃度センサと湿度センサを用いて測定し、温室効果ガスが地球(地表)に与える影響などを検証し考察する。

③ WEB サーバーへのアップロード

測定したデータをすべて宇宙開発研究同好会で使用している WEB サーバーにアップロードし、インターネットから気象データを取得できるようにする。

④ 機器の回収

GPS を搭載して位置情報を取得することにより､機器を回収できるようにする。たとえ回収できなくても同レベルのデータを取得できるようにする。

⑤ 低コストでの実現

なるべく既製品の測定器を使用せず、センサとマイコンを用いて機器を製作しプログラムを自作することで、低コストかつ安定した測定を実現する。

⑥ 上空から地上の撮影

これはボーナスミッションである。上空から地上を撮影し、上空 30000m から日本はどのように見えるのか、また雲はどのように見えるのかをタイムラプスで撮影し検証、考察する。

4. 使用機器

4.1　機器及びセンサ

① Raspberry Pi

この機器は気象データをサーバーにアップロードするために使用した、シングルボードコンピュータである。

図 4.1-1　Raspberry Pi

② Arduino

この機器は今回使用したセンサを制御するためのマイコンである。今回は 2 台使用した。

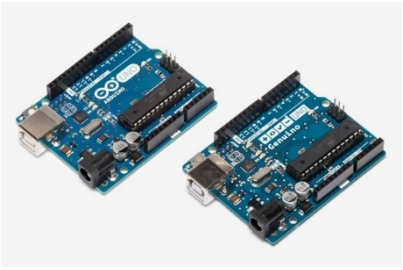


図 4.1-2　Arduino

③ XBee 及び Arduino 用シールド

これは Raspberry Pi と Arduino とのデータ通信に用いたものである。Arduino に接続した XBee は Router であり、後述する 3G－XBee モジュールに接続した XBee は Coordinator として使用した。

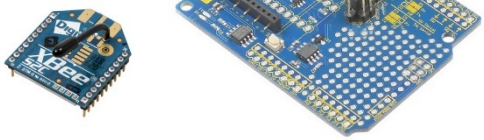


図 4.1-3 XBee

④　3G－XBee 通信モジュール

この機器は独自に開発した通信モジュールであり、Xbee で受け取ったデータを 3G 回線を用いて事前に指定したサーバーにアップロードするための機器である。この機器に搭載する XBee は Coordinator として使用する。

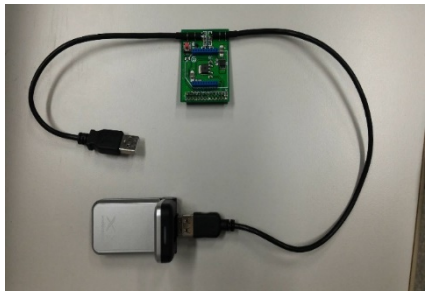


図 4.1-4　通信モジュール

⑤　GROVE－ベースシールド

このシールドはアナログ、ディジタル、I2C 通信に対応しており数多くのセンサを接続できる。今回は 2 台の Arduino に対して 1 台ずつ取り付けて使用しており、1 台には I2C 通信のセンサのみ、もう 1 台にはアナログ通信とディジタル通信のセンサを搭載した。

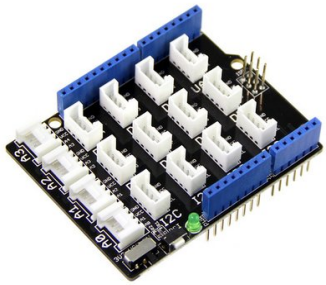


図 4.1-5　ベースシールド

⑥　BME280

このセンサは 1 つで温度、湿度、気圧を測定することができる。サンプルコードには気圧から高度を算出するプログラムが掲載されていたので参考にした。通信形式は I2C である。

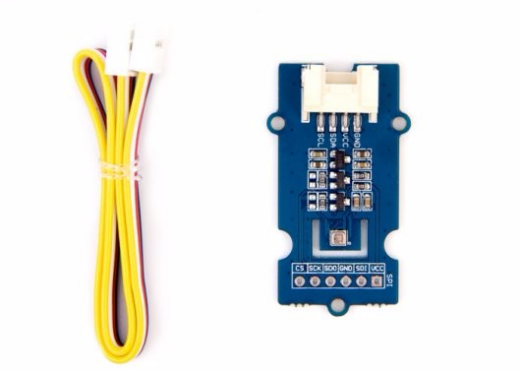


図 4.1-6　温湿度、気圧センサ

⑦ GROVE - ディジタル温度･湿度センサ Pro

このセンサは温度が-40～80 度まで測定できる Pro タイプの温湿度センサである。昨年は Pro モデルを使用せず低温測定ができなかったので、Pro モデルを使用した。

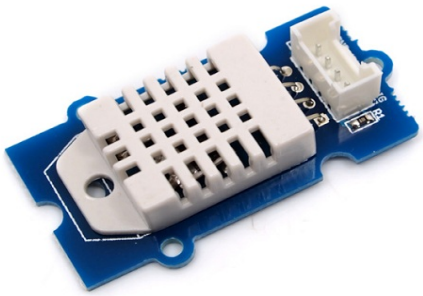


図 4.1-7　温湿度センサ

⑧ ML8511 紫外線センサ

これは紫外線強度を電圧降下により測定するセンサである。オゾン層通過前と通過後の数値を比較しオゾン層による紫外線の減少効果についての測定と考察を行う。

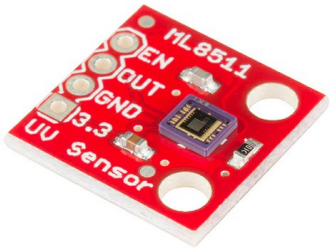


図 4.1-8　紫外線センサ

⑨ GROVE - CO_2 Sensor

これは二酸化炭素濃度センサである。紫外線と同じようにオゾン層通過前と通過後の数値を比較するほか、温室効果ガスの測定でも用いる。

図 4.1-9　CO_2 センサ

⑩ TR-313J　3G/GPS トラッカー

この機器は GPS であり、3G 回線を通じてインターネットに接続されているため、WEB ページから取得した GPS のデータを閲覧することができる。これは衛星の現在地情報の把握のために用いる。

また、この GPS トラッカーの位置を把握するために ezfinder(Google Map を使用した位置情報閲覧サイト)を利用した。

図 4.1-10　GPS

4.2　使用した機器の接続図

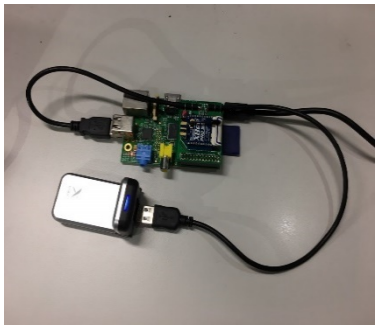


図 4.2-1
Raspberry Pi 及び 3G－XBee 通信モジュール

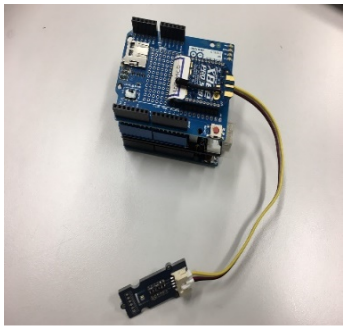


図 4.2-2
Arduino(BME280 を用いた気圧、温湿度)

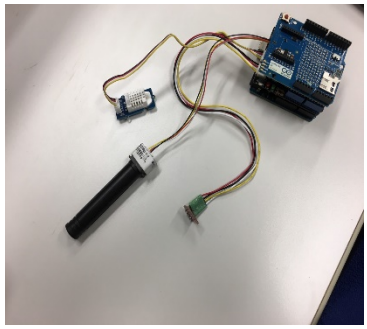


図 4.2-3
Arduino(紫外線センサ、CO2 センサ、温湿度センサ)

4.3　センサノードについて

センサノードの制御システムにはイタリアで教育用マイコンボードとして開発された Arduino を用いた。Arduino を用いた理由として、1 つ目に開発が手軽なことがあげられる。Arduino の統合開発環境(IDE)は Arduino の公式ページから無料でダウンロードすることができ、Arduino のプログラミング言語は C 言語や C++を基本としているため、簡単にプログラミングを行える。2 つ目の理由は Arduino でセンサノード開発を行っている人が非常に多いため、参考になるスケッチ(Arduino のプログラム)が mbed マイコンなどに比べて WEB ページ上に多数公開されているからである。

センサノードは、制御用のマイコンとして前述の Arduino と取得データを送信するための Xbee を使用している。またセンサは精度が良く価格の安い、Grove 製のセンサと Sparkfun 製のセンサを利用した。

センサノードは主に Arduino(図 4.3-1)を基本として、通信モジュールの Xbee(図 4.3-2-a)、センサを取り付けるためのベースシールド(図 4.3-3)、各種センサの 4 つから成り立っている。その構造を図 4.3-4 に示す。

図 4.3-1　Arduino の外観

図 4.3-2-a　XBee の外観

図 4.3-2-b　XBee 取付けシールド

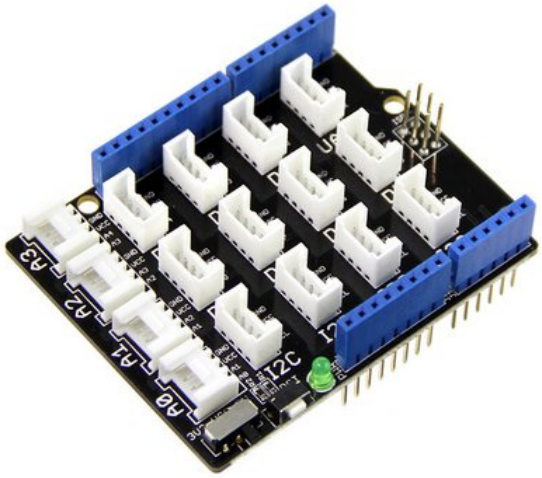


図 4.3-3　ベースシールドの外観

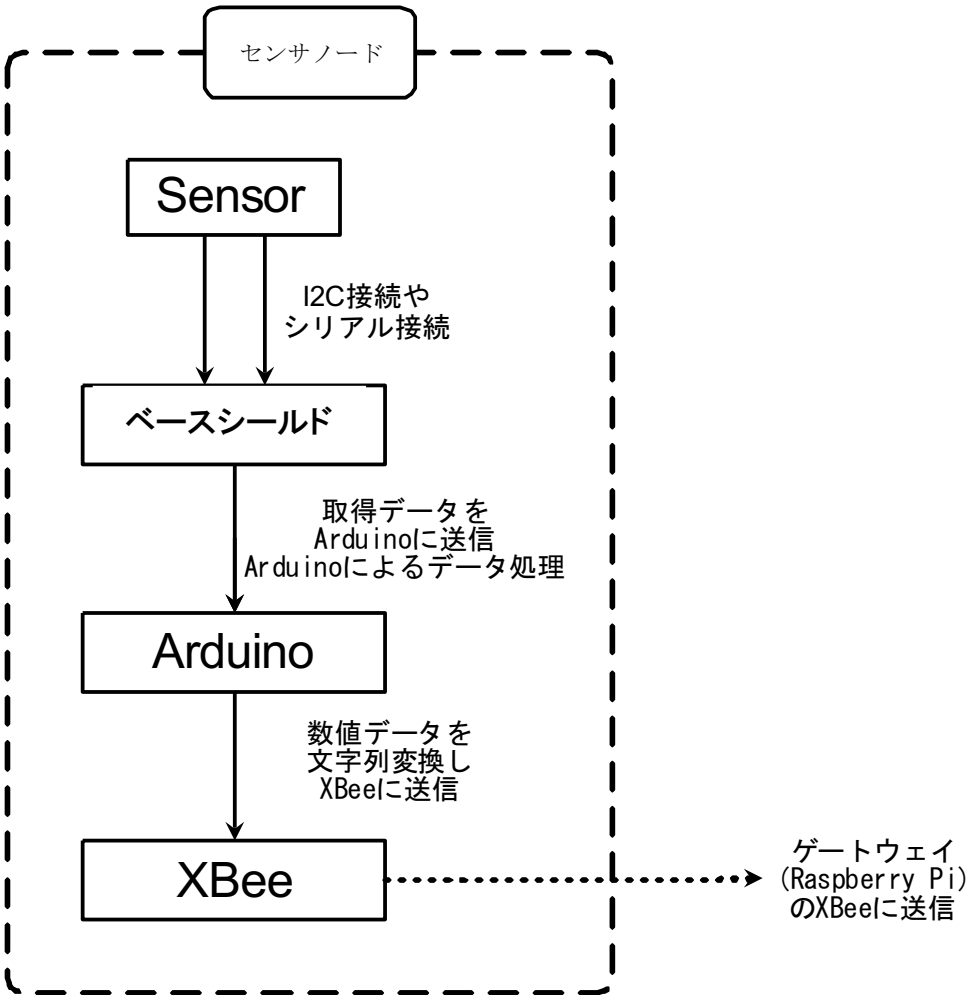


図 4.3-4　センサノードの構造

4.4　各種センサについて

センサノードには、アナログ温湿度センサ、I2C 温湿度・気圧センサ、二酸化炭素濃度センサ、紫外線強度センサを搭載した。センサノードは 2 つ準備し、1 つのセンサノードに I2C 温湿度・気圧センサを搭載し、もう 1 つのセンサノードにその他のセンサを取り付けた。センサノードを 2 つに分けた理由は、1 つ目に、温湿度センサはセンサの中でもっとも重要な気象データであるので、2 つのセンサノードに搭載することで冗長性を持たせようと考えたからである。2 つ目に I2C 温湿度・気圧センサと二酸化炭素濃度センサはどちらも I2C による接続を行っており、それぞれの I2C アドレスが同じであるため、1 つのセンサノードに両方のセンサを接続するとどちらかのセンサの動作が不安定になるためである。

4.5　アナログ温湿度センサ

アナログ温湿度センサは Grove 製の Grove - Temperature & Humidity Sensor Pro を使用した(図 4.5-1)。このセンサの精度は温度±0.5℃以内、湿度±2%以内と非常に高い。そしてアナログセンサはセンサ側で数値処理を行うため、Arduino にかかる負荷が小さいという利点もある。

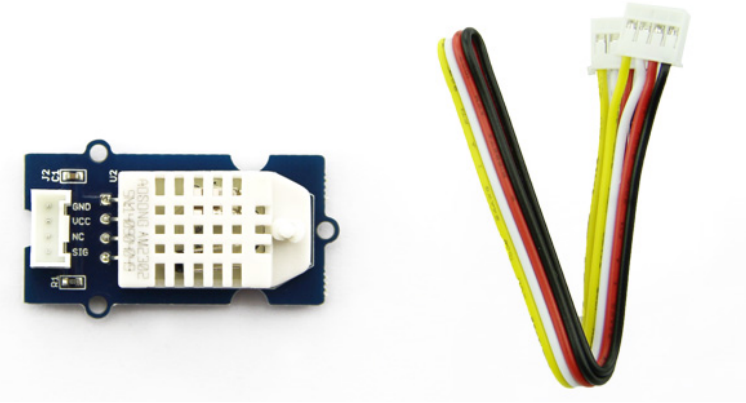


図 4.5-1　アナログ温湿度センサの外観

4.6　I2C 温湿度・気圧センサ

I2C 温湿度・気圧センサには、Grove 製の Grove - Temp&Humi&Barometer Sensor (BME280)を使用した(図 4.6-1)。このセンサの温度の精度は±1℃以内、湿度の精度は±3%以内、気圧の精度は±1hPa 以内と、温湿度に関してはアナログ温湿度センサに劣る。しかし気圧・温度・湿度の 3 情報を同時に取得できる数少ないセンサであること、比較的精度の高い BME280 を使用していることなどの理由からこのセンサを使用した。

I2C 温湿度・気圧センサは名前通り I2C(Inter Integrated Circuit)通信によりデータの送受信を行っている。SPI による通信も可能ではあったが、Arduino にセンサを接続するシールドに SPI 接続を行う端子がなく、またスケッチが複雑になるため使用しなかった。

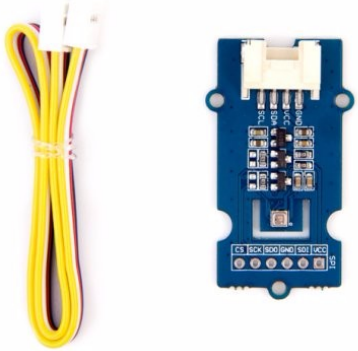


図 4.6-1　　I2C 温湿度・気圧センサの外観

4.7　二酸化炭素濃度センサ

二酸化炭素濃度センサには、Grove 製の Grove - CO2 Sensor を使用した(図 4.7-1)。このセンサはソフトウェアシリアルによる通信でデータの送受信を行っており、空気中の二酸化炭素濃度を精度±200PPM で取得する。またこのセンサにより気温の取得も可能であるが、センサのデータシートに気温の精度の記述がなかったため、このセンサによる気温の取得は行わなかった。

また、このセンサは電源を入れてから測定できるまでに 3 分ほど必要であるため、気球の打ち上げの少し前にセンサの電源を入れておく必要がある。

図 4.7-1　二酸化炭素濃度センサ

4.8　紫外線強度センサ

紫外線強度センサには Sparkfun 製の ML8511 UV Sensor を使用した(図 4.8-1)。このセンサはアナログ温湿度センサと同様にアナログ接続を行っている。また、出力電圧と紫外線強度が比例関係にあるので、出力電圧から紫外線強度を求めることができる。

図 4.8-1　紫外線強度センサの外観

4.9　Arduino のスケッチ

Arduino のスケッチ(動作プログラム)は、資料最終ページの付録に示してある。スケッチは前述の Arduino IDE でスケッチのコンパイルを行い、Arduino に書き込みを行った。センサで取得したデータは基本的に double 型や int 型などの数値型であるが、数値型データでは、XBee で数回に分けて送信する必要があり、また数値型データはゲートウェイで正常に処理できない。そこで一度文字列型に変換してすべてのデータを CR で区切り、1 つの長い文字列型変数に図 4.9-1 のように格納した後、XBee でゲートウェイに送信する方式をとっている。

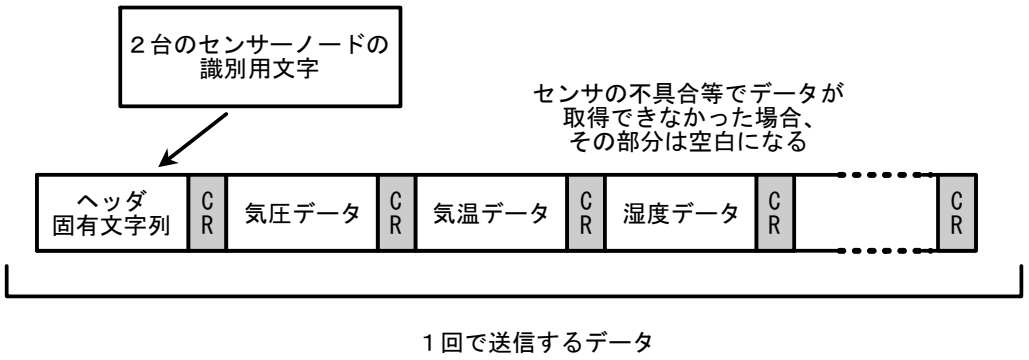


図 4.9-1　センサノードのデータ送信フォーマット

例えば、ヘッダ固有文字列が「S0013A20040ADA54D」、気圧が 1013.11 hPa、気温が 27.3 ℃、湿度が 60.0 %、二酸化炭素濃度が 678 PPM の時、センサノードからゲートウェイに送られるデータは、

“S0013A20040ADA54DCR1013.11CR27.3CR60.0CR678CR”となる。

観測間隔は本来短ければ短いほど正確なものになるが、ゲートウェイで処理しきれなくなる危険性も考慮して 1 分間に 1 回の計測を行うようにしている。またこの時、I2C 温湿度・気圧センサを取り付けたセンサノードと、それ以外のセンサを取り付けたセンサノードが同時にゲートウェイにデータを送信すると、データを処理しきれなくなりどちらかのデータが破損することがある。そのため 2 つのセンサノードがデータを送信する間隔を 30 秒間ずらすことで、ゲートウェイが処理しきれなくなってしまうことを防止している。

4.10　ゲートウェイ(Raspberry Pi)の概要

今回 Raspberry Pi で使用した OS は Raspbian をベースとし、ネットワークサーバー用に改良を加えたものであり、これを Arduino が取得したデータの送信先サーバーとして構築した。

さらに、図 4.10-1 の config ファイルでデータを送るサーバーの選択を行い、データの送信間隔や 3G エリア外に出たときの挙動などを決める。また、この config ファイルは Windows OS のテキストエディタで編集し、その際この OS は 3G 圏外に出ると再起動し続けるため、再起動させないプログラムを追加した。ネットワークに接続されていない間の Arduino から送られるデータは、Raspberry Pi の記憶領域に保存しておき再度ネットワークにつながったときに、まとめてデータが送られるようになっている。

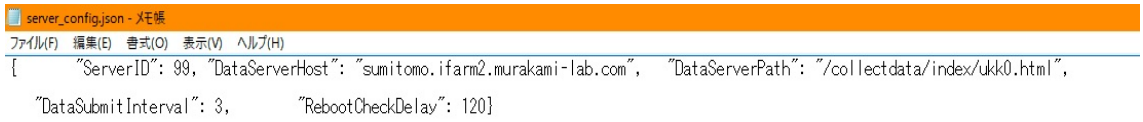


図 4.10-1　config ファイル

Raspberry Pi に搭載している通信モジュールは図 4.10-2 と図 4.10-3 の一体型のモジュールである。これは、Arduino のデータを WEB サーバーにアップロードするために開発した。

図 4.10-2 は Arduino との通信を行う XBee シールド、図 4.10-3 は WEB サーバーにアップするための 3G モバイルルーターである。モバイルルーターについては NTTdocomo の Xi(クロッシー)を使用しており、SIM カードを挿すだけで通信ができる。また、シールドに搭載されている赤いボタンはシャットダウンスイッチである。これを用いるとマウス及びキーボードを接続しなくても安全にシステムを停止させることができる。

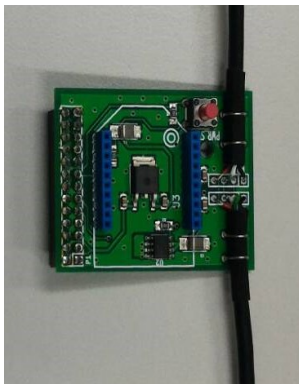


図 4.10-2　XBee シールド

図 4.10-3　モバイルルーター

4.11　WEB サーバーの概要

今回は、村上研究室にあるサーバーを使用した。このサーバーは観測データの生データを表示、サーバー内で生データを自動的に編集、CSV 形式で表示、サーバーに送られてきたデータの数やデータの破損状況を確認するといったデータ管理機能を持っている。

今回のデータ閲覧の URL と表示結果の例は以下のとおりである。

URL : 生データ： http://sumitomo.ifarm2.murakami-lab.com/ukk0/rawdata.txt
センサ：　http://sumitomo.ifarm2.murakami-lab.com/ukk0/sensordata.csv
サーバー：http://sumitomo.ifarm2.murakami-lab.com/ukk0/serverdata.csv

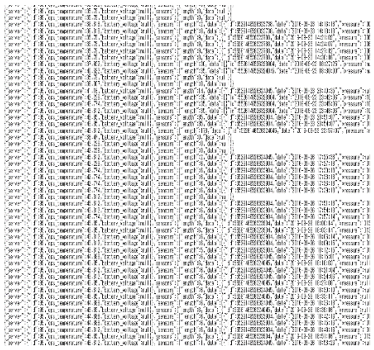


図 4.11-1　生データ

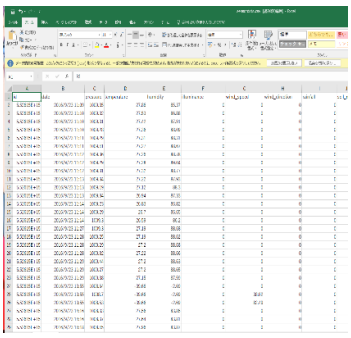


図 4.11-2　センサーデータの CSV ファイル

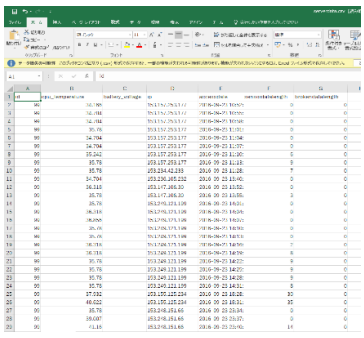


図 4.11-3　サーバーデータの CSV ファイル

表 4.11-1 と表 4.11-2 に、各 CSV ファイルで確認できるデータの一覧を示す。

表 4.11-1　センサーデータの CSV ファイルで確認できる項目一覧

項目	概要
id	センサノードの識別番号
date	気象データの取得日時
pressure	現在の気圧　単位は［hPa］
temperature	現在の気温　単位は［℃］
humidity	現在の湿度　単位は［%］
illuminance	現在の照度　今回は紫外線強度として使用
wind_speed	現在の風速　今回は使用しなかった
wind_direction	現在の風向　今回は使用しなかった
rainfall	現在の雨量　今回は使用しなかった
soil_moisture	現在の土壌水分　今回は使用しなかった
additional_sensor1~5	任意の追加センサ用　今回は二酸化炭素濃度センサ用に使用

表 4.11-2　サーバーデータの CSV ファイルで確認できる項目一覧

項目	概要
id	ゲートウェイの識別番号
cpu_temperature	CPU 温度
battery_voltage	ゲートウェイ（Raspberry Pi）の電源電圧
ip	ゲートウェイに割り振られた IP アドレス
accessdate	データの送信時刻
sensordatalength	センサーデータから送られてきたデータ数
brokendatalength	センサノードから送られたデータ数のうち、受信に失敗したデータ数

4.12　カメラ

この衛星には、撮影用としてスマートフォン(SAMSUNG 製　Galaxy SⅡ　LTE)を使用した。当初の予定では Raspberry Pi を使用する予定だったが、Raspberry Pi を使用しなかった理由として、スマホの方が安価であったこと、システム的に構築が簡単であったこと、画質が良かったことなどが挙げられる。

今回は定時撮影アプリで撮影した画像を OneDrive と Google Drive にアップロードした。2 つのクラウドサービスにアップロードした理由は 1 つのクラウドサービスだけでは、何らかのエラーが起こるとアップロード不能になるからである。これをふまえ、二重でアップロードすることでデータ取得の安定性を上げることができた。

図 4.12-1　 Galaxy SⅡ　LTE

4.13　ezfinder について

ezfinder は 7 ページの⑩で示した通り、GPS トラッカーの位置を把握するために用いたサイトである。無料版「ezfinder」と有料版「ezfinder business」の 2 種類があり、今回は無料版を使用した。以下に簡単な使用手順を示す。

サイトにログイン後、機種番号(IMEI)と電話番号を登録する。GPS 情報を受信すると、緯度、経度、高度をはじめとする位置情報や時刻などが表示される。(図 4.13-1)WEB ページから情報が得られるため、遠隔地でも衛星の状態が確認できる。

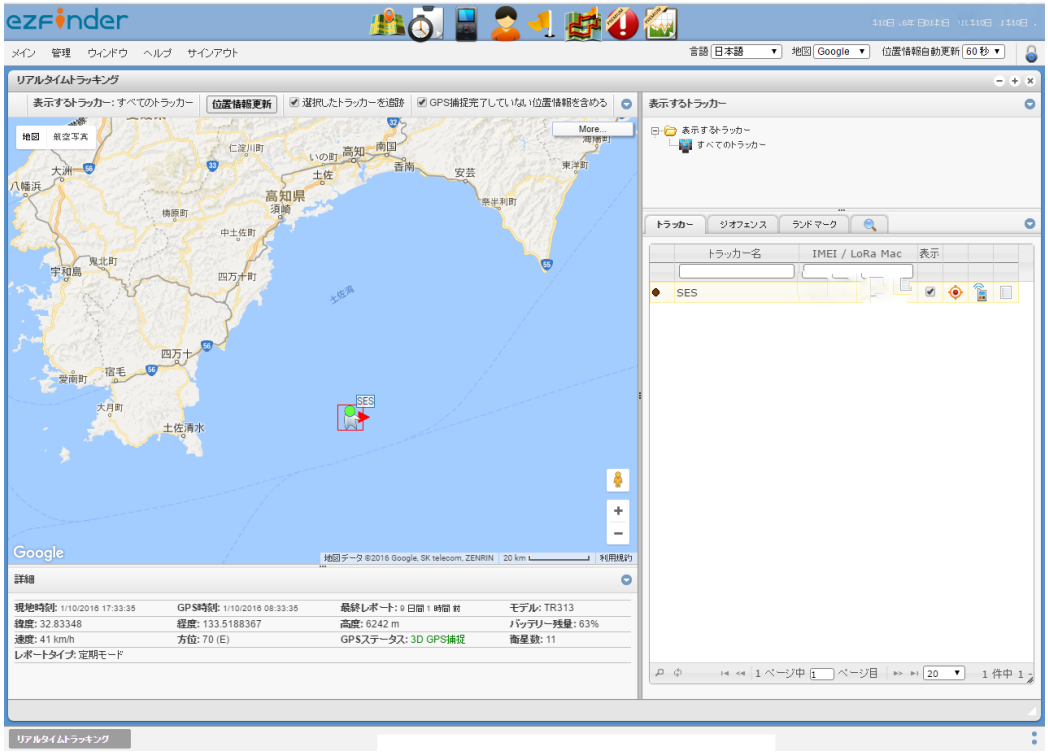


図 4.13-1　ezfinder サイト

5.　衛星全体の概要

5.1　キャリア

図 5.1-1 はセンサ積み込み時の衛星内部映像である。Arduino は動かないようにグルーガンで固定し、スマホは瞬間接着剤で固定した。Xi に関しては今回使用した機器の中で最も発熱し、その熱が Raspberry Pi などの機器に伝導すると熱暴走を起こして Raspberry Pi が動作を停止してしまうため、側面に固定した。

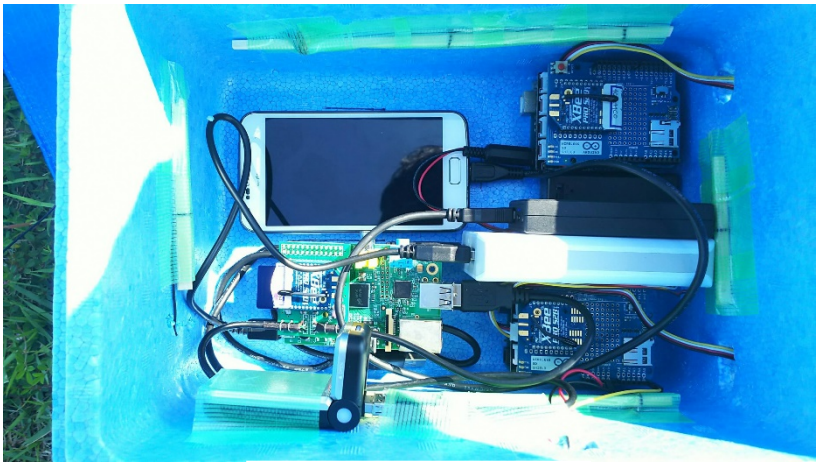


図 5.1-1　キャリア内部

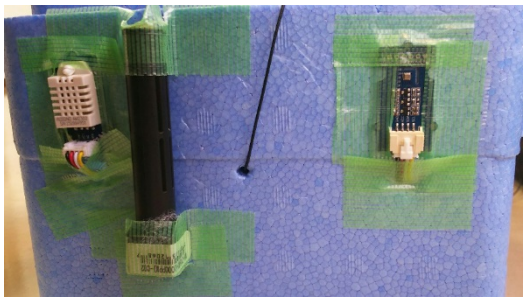


図 5.1-2　キャリア外部 1

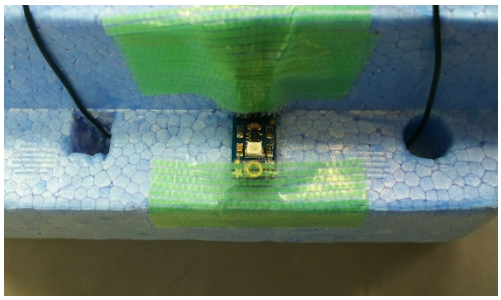


図 5.1-3　キャリア外部 2

5.2　パラシュート

今回、衛星に搭載したパラシュートは実際に我が校のヨット部で使用しているヨットの帆を再利用した。理由は素材が丈夫であるからだ。パラシュートの大きさは直径約 1.5m で、それを 8 本の釣り糸でキャリアに固定した。落下実験では時速 2.5m まで落下速度を抑えることができ、ヨットの帆を用いたパラシュートの性能を実証することが出来た。

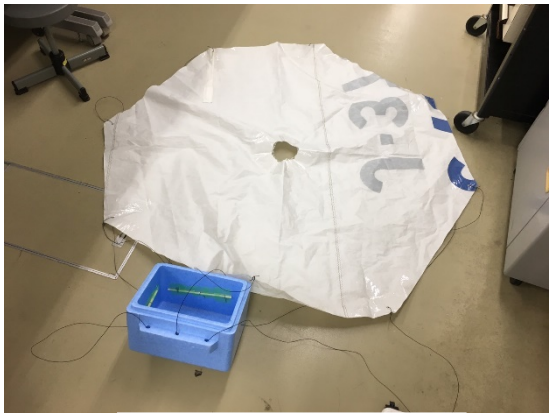


図 5.2-1　パラシュート

5.3　バルーン

使用したバルーンは直径約 1m 重さ 1kg ほどで、ヘリウムガスを 3000L 注入して放球を行った。参考として、ヘリウムガスは 1000L で 1kg のペイロードを浮かせることができる。今回はバルーン 1kg+キャリア 1.05kg で計 2.05kg となったため、余裕を持って 3000L 注入した。

図 5.3-1　バルーン

6.　気象データの閲覧方法

下図を用いて気象データの閲覧までの過程を記す。

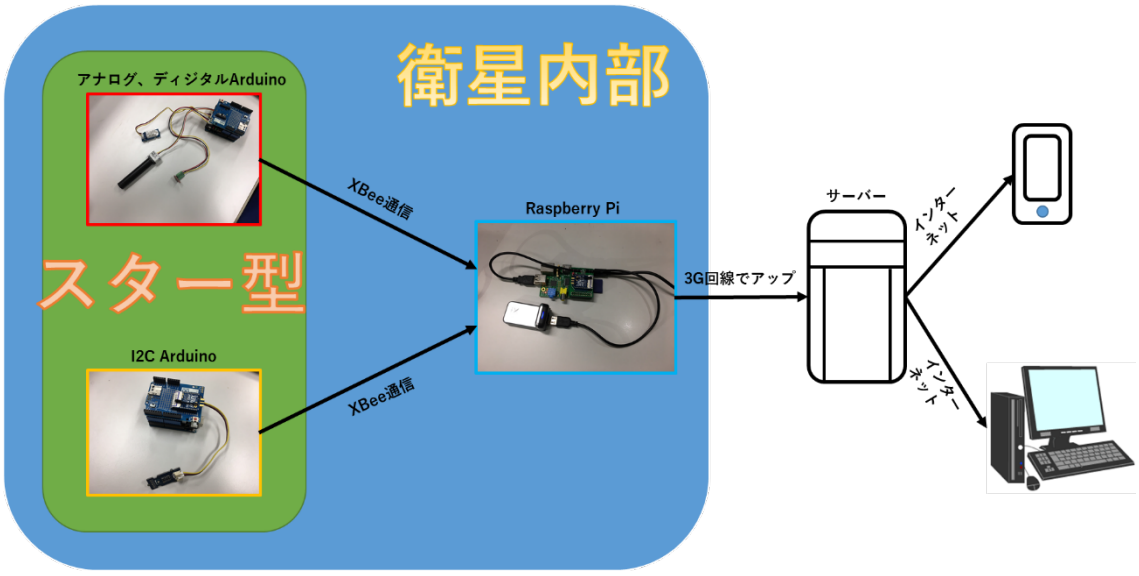


図 6-1　観測データ閲覧図

図 6-1 は Arduino で測定したデータをスマホなどの端末で閲覧するまでの過程を図に示したものである。今回衛星内部の通信は XBee を用いており、Router　2 台に対し Coordinator 1 台のスター型と呼ばれる通信方式を採用した。Arduino のデータを受け取った Raspberry Pi は 3G 回線を通じてサーバーにデータをアップロードする。そして、スマホやパソコンをはじめとするネットワーク機器からインターネット

を通じてサーバーにアクセスすることで、観測データを閲覧することができる。つまり、URL さえ知っていれば世界中の誰でも、どこからでもリアルタイムで閲覧することができるのである。

7. 測定方法

・放球地点：土佐清水総合公園(図 7-1)
(32.7879、132.9578)
・放球日時：10 月 1 日　15:30
・現地天気：晴れ・北西の風
・落下予想地点：(図 7-2)

落下地点(33.1468、133.498)
落下地点北 10km(33.2369、133.498)
落下地点南 10km(33.0567、133.498)
落下地点東 10km(33.1468、133.607)
落下地点西 10km(33.1468、133.388)

図 7-1　放球地点

図 7-2　落下予想地点

8. 測定結果

8.1　気温

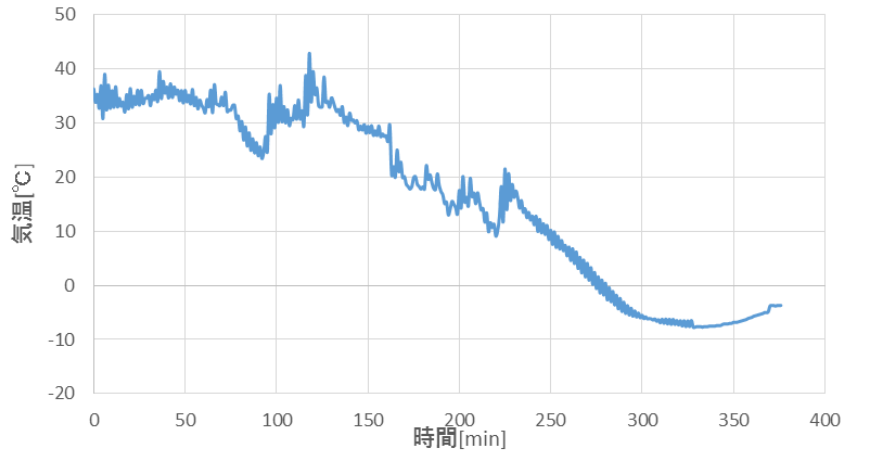


図 8.1-1　気温グラフ

[グラフの解説]

図 8.1-1 のグラフより 15:30 からセンサを動かし始めた。その時には 30℃以上の値を示していたことが読み取れる。しかし、時間が経つにつれて気温は急激に減少し 18:22 には最低温度の-7.4℃となった。また、今回は 2 つの温度センサで測定しており気温取得データ数は 377 個であった。

8.2 湿度

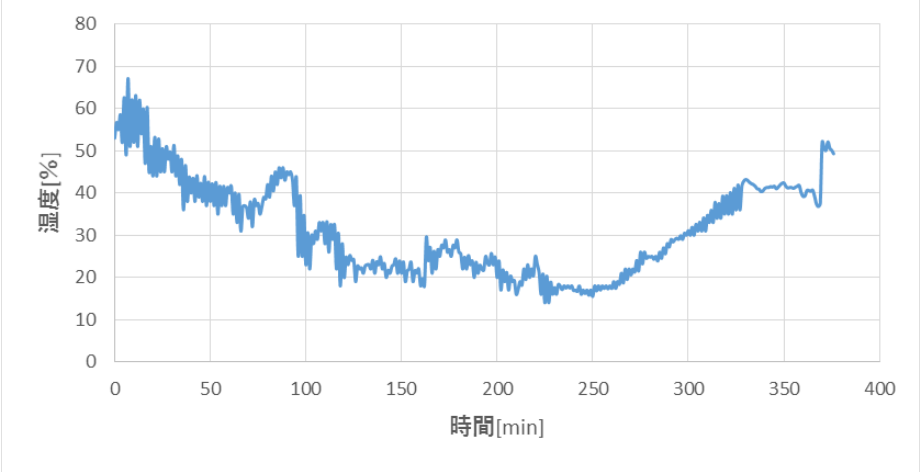


図 8.2-1 湿度グラフ

[グラフの解説]

図 8.2-1 のグラフより放球直後から、湿度が急激に下がっていることが分かる。打ち上げから 225 分の 14%が最低湿度でありそこを過ぎると再び湿度が上昇していることが分かる。

8.3 気圧

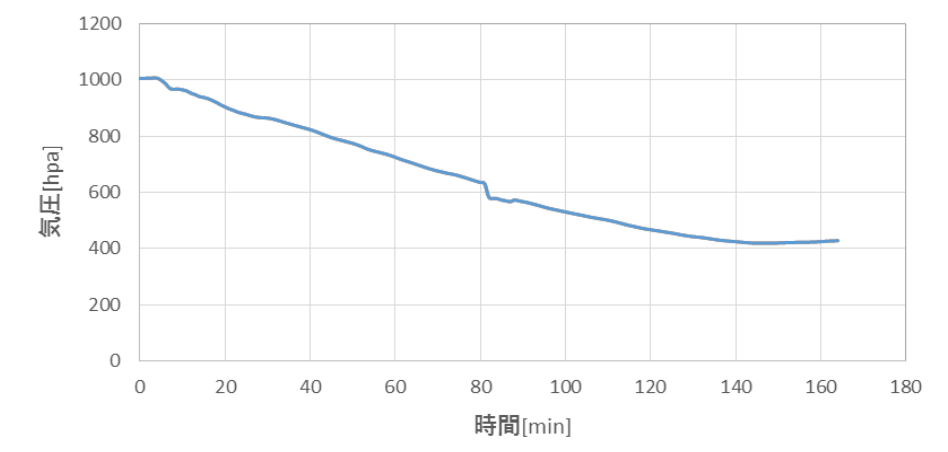


図 8.3-1 気圧グラフ

[グラフの解説]

図 8.3-1 のグラフより放球直後から、気圧が緩やかに低下していることが分かる。値はおよそ 160 分で途切れているためこのセンサの電源が切れてしまったと思われる。

8.4　高度

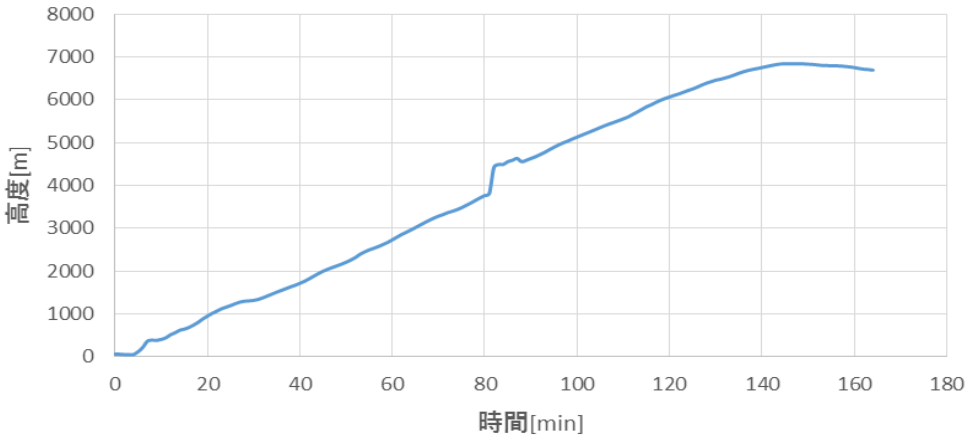


図 8.4-1　高度のグラフ

[グラフの解説]

図 8.4-1 のグラフより、放球 5 分後から緩やかに高度が上昇し始めていることが分かる。143 分後に測定最高高度 7000m に達した。また、80 分後付近では高度変化が激しくなっていることも分かる。また、この高度のグラフは気圧から算出している。以下に計算式を示す。

h:高度　p:気圧

$$h = \frac{1 - \left(\frac{p}{1.01325 \times 10^{5}}\right)^{\frac{1}{5.25588}}}{2.25577 \times 10^{-5}}$$

8.5 紫外線強度

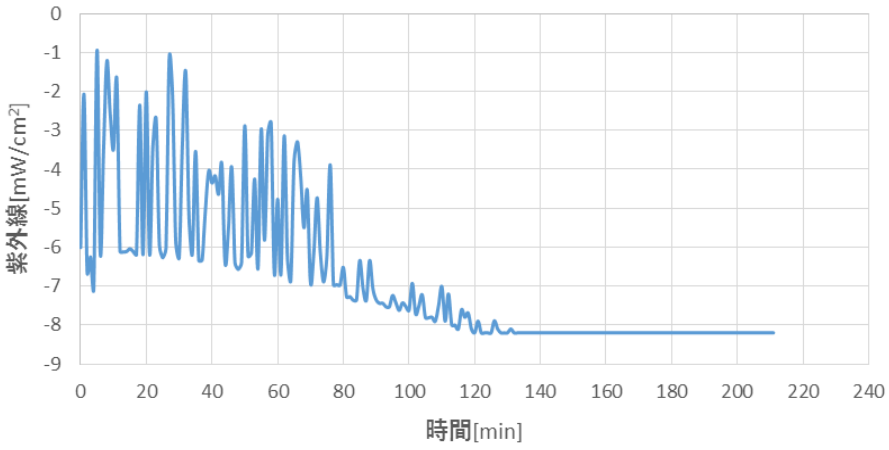


図 8.5-1　紫外線強度のグラフ

[グラフの解説]

図 8.5-1 のグラフより、放球直後から値が激しく変動していることが分かる。しかし、時間が経過していくにつれて、値の変動幅が小さくなり放球から 130 分経過したところで紫外線の値が 0[mW/cm²]となっている。また、紫外線強度の基準値はグラフより-8[mW/cm²]に設定する。

8.6　二酸化炭素濃度

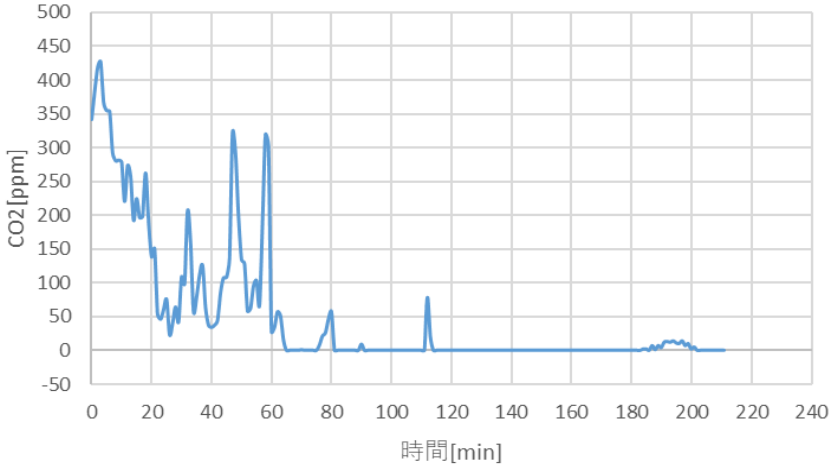


図 8.6-1　二酸化炭素濃度のグラフ

[グラフの解説]

図 8.6-1 のグラフより、放球直後から数分間は値が上昇しているが、その後は急激に値が減少していることが分かる。また、放球してから 50 分後と 60 分後にはそれぞれ値が急激に変動していることも分かる。

8.7　飛行経路

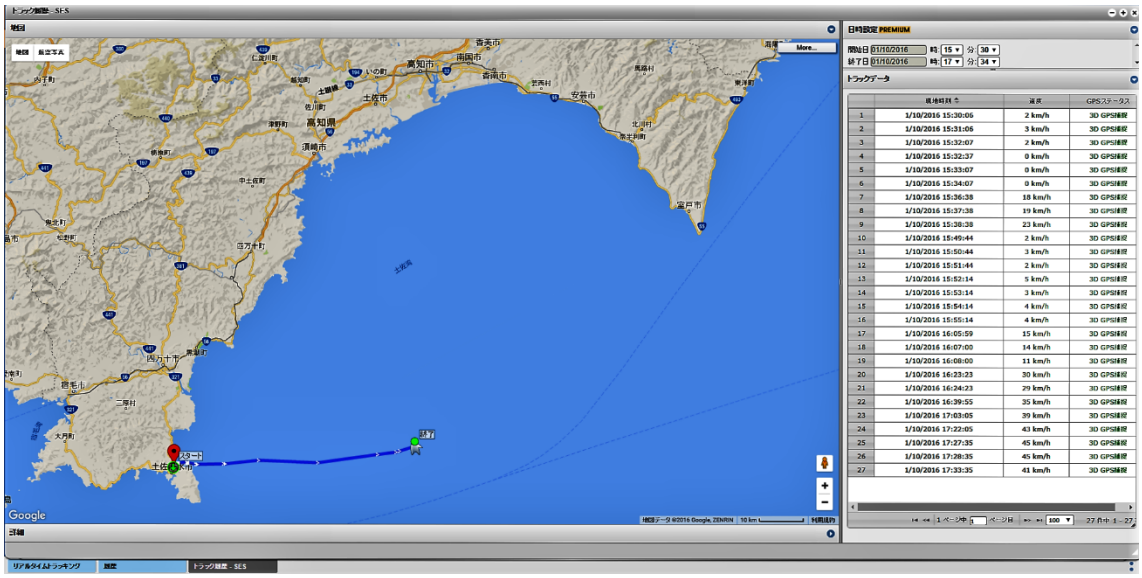


図 8.7-1　飛行経路

[図の解説]

図 8.7-1 より放球後ほぼ真東に進んでいることが分かる。今回は 17:34 でロストしてしまったため着水までのすべてのデータを得ることができなかった。データの軌道より室戸岬の南約 65km 付近に落下したと思われる。放球後徐々に速度を上げて 17:25 に、観測した中で最も速い時速 45km を記録した。

9. 考察

9.1 気温・湿度

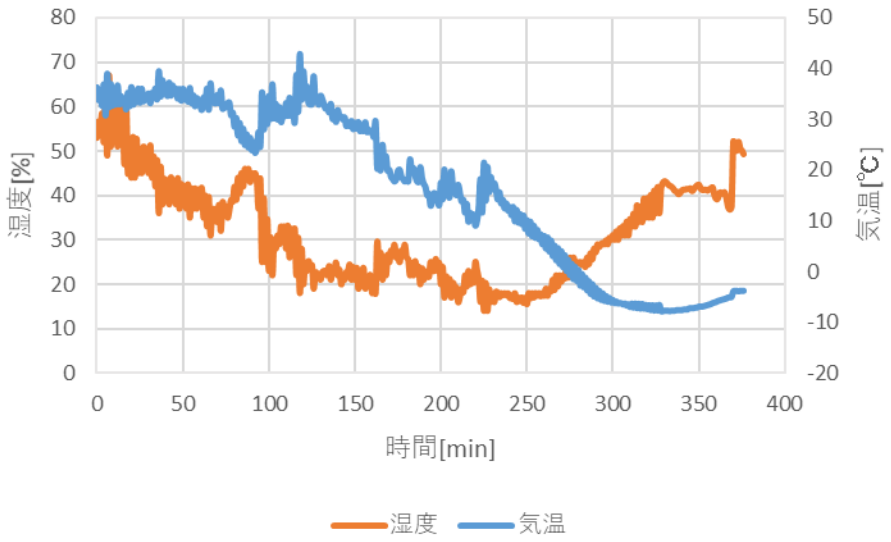


図 9.1-1 気温・湿度の推移

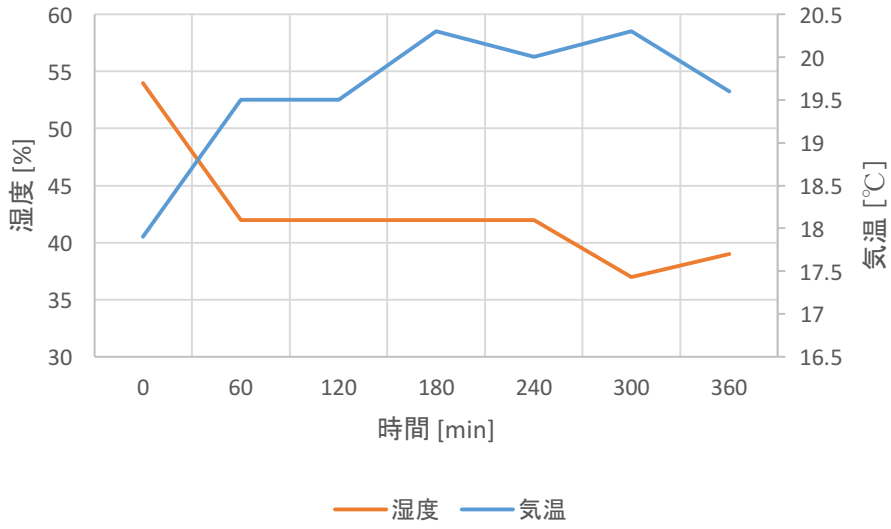


図 9.1-2 昨年度のデータ

図 9.1-1 より全体的に二つのグラフは対称の関係になっていることが分かる。このようになるのは空気中にほとんど均等に水蒸気があり、気温が高くなると飽和水蒸気量が増え相対的に湿度が減少するためである。逆に気温が低くなると飽和水蒸気量が減り相対的に湿度が高くなる。そのためグラフは対称の関係にあると考えられる。

また、図 9.1-2 は、昨年の気象観測機器コンテストで、「ヘイズカスーミー ～視界距離自動判定装置～」が測定した地上の気温・湿度の推移のグラフである。こ

のグラフを見ても、気温と湿度のグラフは対称の関係になっている。このことより、気温・湿度のデータが正確に取得できていることが分かり、上空でも地上でも気温と湿度の関係は変わらないことが分かる。

また、放球後250分のあたりで気温が上がり湿度が下がっていることが分かる。通常、地上で温湿度の測定を行うと夜は気温が低くなり、湿度は高くなるはずである。だが、この図を見ると逆の結果になっている。これは寒気の中にいた衛星が温暖前線を超えたため気温が高くなり湿度が減少したためだと考えられる。

さらに、全体的に温湿度の値が安定して推移していない。これは衛星が雲の中を通過したこと、及び、センサが日陰に入ってしまったことの 2 つの理由が考えられる。

9.2　二酸化炭素濃度・紫外線強度

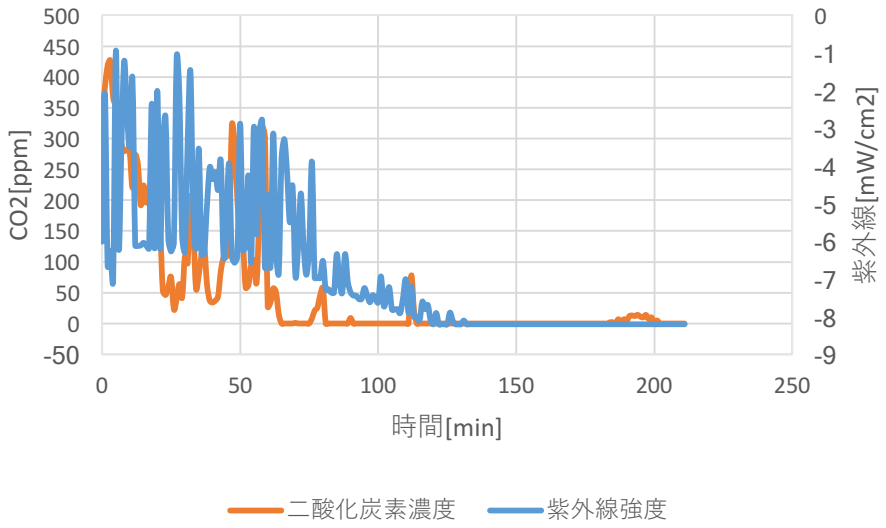


図 9.2-1　二酸化炭素濃度・紫外線強度の推移

図 9.2-1 より、2 つのグラフはほぼ比例の関係になっていることが分かる。また放球直後から緩やかに減少していることも分かる。

グラフ上の二酸化炭素濃度のデータに関しては、気体の分子量が関係している。空気の構成を窒素 80%酸素 20%と考えると以下の計算式より空気の分子量は約 28.8、二酸化炭素の分子量は 44 であることが分かる。したがって、空気の質量よりも二酸化炭素の質量が重いため高度が上昇するにつれて二酸化炭素濃度が減少していると考えられる。

紫外線は太陽から放出されるため 12～13 時の昼間に値が大きく上昇する。しかし、今回は 15 時 30 分に放球したため、大きな変化は見られなかった。また、

大きく値が低下しているのは日没が原因であると考えられる。それでも紫外線強度が 0 になることはなかったため、太陽が出ていなくも少なからず紫外線は存在することが分かった。

また今回使用した紫外線強度センサは電圧から強度を算出するため電源電圧の影響を受けやすく、さらに日陰の影響も受けるため安定した値を得ることができなかったと考える。

・計算式

O＝16　N=14　C＝12 より

$Air=O_2\times0.2+N_2\times0.8$

$=16\times2\times0.2+14\times2\times0.8$

$=32\times0.2+28\times0.8$

$=28.8[g/mol]$

$CO_2=C+O_2$

$=12+32$

$=44[g/mol]$

9.3　カメラ

図 9.3-1　放球直後

図 9.3-2　15 分後

図 9.3-3　30 分後

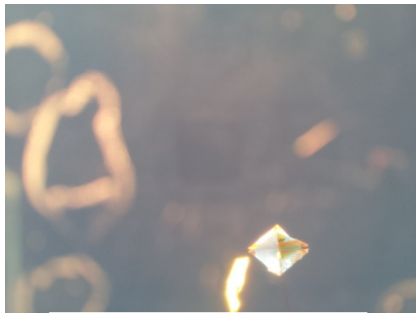


図 9.3-4　2 時間後

上の画像は気球に搭載したカメラが撮影したものである。放球直後は陸地が写っていたが、15 分後には海上だけが写っている。また、2 時間後にはレーダ

反射装置が太陽光を反射していることから日の入りの時間帯であると考える。

画像に偶然写っている気泡に注目する。時間が経つにつれて、放球直後は気泡がほとんど見えなかったが、2 時間後には気泡の輪郭をはっきりと確認することができた。また、気泡の大きさもだんだんと大きくなっていることが分かる。これは、バルーン上昇時の気圧の低下による気体の膨張であることが考えられる。放球場所の高度は海抜約 50m、2 時間後の高度は海抜約 6400m で気圧は 1/2 以下になっていた。したがって、気体は気圧が低下すると膨張すると考えられる。また、密閉状態であっても、同じく気体が膨張することが分かった。

10. 評価

今回僕たちは、3 章に示したように、6 つのミッションを掲げて機器の製作及び実験を行った。各々のミッションについて評価を行った。

① 「上空の気象を測定」について

上空の気圧・温湿度データ、特に温湿度データについては、2 つのセンサを用いた。30 秒間隔で測定したため、きわめて正確な値が取得できていた。よって、このミッションは成功したといえる。

また、今回の気球はバルーンの破裂後、落下中にも気象データを観測できるようにするため、減速機構(パラシュート)にも工夫を加えた。これまで僕たちが製作してきた衛星はパラシュートの素材としてブルーシートやゴミ袋を利用していた。しかしブルーシートやゴミ袋では軽量化こそできても、強度が低いため落下中に破れてしまい減速効果が十分に得られない危険性があった。これは万が一にも市街地に機器が落下したときの事を考えると大変危険なことである。そこでブルーシートに変わり、ヨットの帆を利用した。ヨットの帆は強度に優れ、軽量であるという利点があるため空中で破けてしまうという危険もない。これにより十分な減速効果が得られ、落下中も気象データの測定、上空の撮影などが行える。

② 「温室効果ガスの測定」について

温室効果ガスの原因である、水蒸気量(湿度)と二酸化炭素濃度を測定した。特に湿度に関しては、2 つのセンサを用いて冗長性を持たせていたため、きわめて正確な値が取得できていた。

③　「WEB サーバーへのアップロード」について

過去の気象観測機器コンテストでは、外注したサーバーを利用していたが、今年は本校の村上研究室のサーバーを利用し、自由度が高くまた安定性も大きく向上した。この結果、すべての気象データを重複なくサーバーにアップロードすることに成功した。

④　「機器の回収」について

打ち上げ日の気候や、事前に行った落下地点の予測などから、回収は困難であると判断した。しかし、今回のシステムはすべてのデータを WEB サーバーにアップロードしていたため、データが回収できないという事態は回避できた。またカメラで撮影した画像も、クラウドサービスを利用し WEB 上にアップロードしていたため、「回収ができなくても同レベルのデータを取得する」というさらに高度なミッションに成功したといえる。

⑤　「低コストでの実現」について

測定器は過去に僕達が作成した機器と違い、すべての測定器をセンサとマイコン制御によって構成することができた。これにより、機器の作製費を、大幅に削減することができた。また、既製品の測定器を使わなかったことで機器の軽量化に成功したため、バルーンに注入するヘリウムガスの量を、去年の 4000L から 3000L に減らすこともできた。

⑥　「上空からの地上の撮影」について

このミッションに関しては, ボーナスミッションであったため、実現までに非常に時間を要したが、スマートフォンを利用することでミッションを成功させることができた。スマートフォンは現在、SIM ロック解除の義務化や SIM フリースマートフォンの増加により、非常に安い価格で入手することができる。これにより、当初案であった Raspberry Pi を用いた空の撮影よりも低いコストで高高度撮影を行えた。またスマートフォンの強みとして、撮影した画像をクラウドシステムにアップロードし易いという点がある。この利点を最大限活かし、Raspberry Pi では実現困難であった複数のクラウドシステムへの画像の同時アップロードを行うことにも成功した。

11. 感想

今回のプロジェクトを行う際、昨年に実機を回収できなかったこともあり、すべてのデータを得るために、成功実績のある Weatherdoino を用いて一つの Arduino で測定することを検討した。しかし、Weatherduino と二酸化炭素センサのライブラリが競合してしまいプログラムを実行することができなかった。そこで、Arduino 2 台から Raspberry Pi にデータを送るスター型と呼ばれる通信方式を採用した。この通信方式は UNISEC の副理事長である日本大学の宮崎 康之 教授主催の HEPTA ハンズオン講座で学びそれを応用した。2 つのセンサノードから 30 秒ごとにデータを送ることで 2 台の Arduino 同士が競合することなく安定した通信を行うことができた。

写真の撮影は当初 Raspberry Pi を用いる予定だったが、3G 回線を抜けた際クラウドサービスへのアップロードができずプログラムが終了し再度 3G 回線に入ってもプログラムが実行されないという問題が発生したためスマートフォンでの撮影に切り替えた。スマートフォンで撮影することで、画像が自動でクラウドサービスにアップロードされ、尚且つ中古のスマートフォンを使用することで Raspberry Pi で撮影するよりもコストを抑えることができた。

最後に、今回は機器の回収というミッション以外は無事達成することができた。また、偶然にも HEPTA のハンズオン講座を受けられたことでプロジェクトを大きく前進させることもできた。そして、上空 30000m とはいかなかったものの 6000m までの気象データを詳細に測定することができた。来年以降は機器の回収もできるようにシミュレーションを重ねていきたいと思う。

12. Arduino スケッチ

・以下に今回使用した Arduino のスケッチを示す。

```
/*****************************************************
*  アナログ温湿度センサ，UV強度センサ，CO2濃度センサ用
*  センサ値を取得およびGatewayへの送信用
*  GatewayにはRaspberry Piを用いること
*  電源にはUSB給電，5V1Aを使用
*  実験及び打ち上げにはArduino Uno Rev3を使用
*
*  スケッチ作成日  2016年9月
*
*  参考にしたスケッチ
*  ・圃場センサ（センサノード）用スケッチ
*     第3回気象観測機器コンテスト（KOINOBORI Project）で使用
*  ・ヘイズカスーミー：気象データ取得用スケッチ
*     第4回気象観測機器コンテスト（ヘイズカスーミー）で使用
*  ・Grove wiki,Sparkfun社サイト
*****************************************************/


#include <Watchdog.h>
#include <Sleep.h>
#include "DHT.h"
#include <Streaming.h>
#include <PString.h>
#include <stdlib.h>
#include "Barometer.h"
#include <Wire.h>
#include <SoftwareSerial.h>


/*
温湿度センサはベースシールドA1,A2に接続
UV強度センサはベースシールドA0,A1に接続
CO2濃度センサはベースシールドD7,D8に接続
*/

//UVセンサ用
int UVOUT = A0;
int REF_3V3 = A1;
//UVセンサここまで

//CO2センサ用
#define DEBUG 0
const int pinRx = 8;
const int pinTx = 7;

SoftwareSerial sensor(pinTx,pinRx);
```

```
47
48  const unsigned char cmd_get_sensor[] =
49  {
50      0xff, 0x01, 0x86, 0x00, 0x00,
51      0x00, 0x00, 0x00, 0x79
52  };
53  unsigned char dataRevice[9];
54
55  //CO2センサ用ここまで
56
57
58  #define DHTPIN A1     // 温湿度はA1に接続
59  #define DHTTYPE DHT21   // DHT 21 (proモジュール用)(11で通常温湿度)
60
61  WatchdogClass WD = WatchdogClass(); //WDT
62  DHT dht(DHTPIN, DHTTYPE); //温室度ライブラリ用
63
64
65  void setup()
66  {
67   //serial begin(9600でシリアル通信開始)
68    Serial.begin(9600);
69    sensor.begin(9600);
70
71    pinMode(UVOUT, INPUT);
72    pinMode(REF_3V3, INPUT);
73
74    WD.systemResetEnable(false);
75    WD.enable(WatchdogClass::TimeOut8s);  // スリープ間隔を8秒に設定
76    dht.begin();                          //温湿度ライブラリ用
77
78  }
79
80  void loop ()
81  {
82    float tem,hum,pre,co2,alt;
83
84    //温湿度の取得
85    hum = dht.readHumidity();
86    tem = dht.readTemperature();
87    //温湿度の取得終わり
88
89
90    //UVセンサ値の取得
91
92    int uvLevel = averageAnalogRead(UVOUT);
```

```
93      int refLevel = averageAnalogRead(REF_3V3);
94
95
96      float outputVoltage = 3.3 / refLevel * uvLevel;
97
98      float uvIntensity = mapfloat(outputVoltage, 0.99, 2.8,
        0.0, 15.0);
99
100     //UVセンサ取得終わり
101
102
103     //CO2センサここから
104     byte data[9];
105     int i = 0;
106
107       for(i=0; i<sizeof(cmd_get_sensor); i++)
108       {
109           sensor.write(cmd_get_sensor[i]);
110       }
111       delay(10);
112
113       if(sensor.available())
114       {
115           while(sensor.available())
116           {
117               for(int i=0;i<9; i++)
118               {
119                   data[i] = sensor.read();
120               }
121           }
122       }
123
124       co2 = (int)data[2] * 256 + (int)data[3];
125
126       //CO2センサは，値の取得に失敗した場合，飛ばされます
127
128     //CO2センサここまで
129
130     char tems[20],hums[20],uvs[20],co2s[20]; //データ格納用文字列変数
131
132
133     //数値データを文字列に変換
134     dtostrf(tem,7,2,tems);
135     dtostrf(hum,7,2,hums);
136     dtostrf(uvIntensity,7,2,uvs);
137     dtostrf(co2,7,2,co2s);
```

```
138
139
140     //Gatewayへの送信用文字列
141     char buffer1[200];
142     PString str1(buffer1,sizeof(buffer1));
143
144
145     //文字列値をCRで区切り格納
146     str1 +="\r";//指定CR
147     str1 +="S\r";//ヘッダ固有文字列＋CR
148     str1 +="0013A200\r";//固有番号＋CR
149     str1 +="40ADA54D\r";//固有番号＋CR
150     str1 +="\r";//気圧CR
151     str1 +=tems;//温度
152     str1 +="\r";//温度CR
153     str1 +=hums;//湿度
154     str1 +="\r";//湿度CR
155     str1 +=uvs;//照度
156     str1 +="\r";//照度CR
157     str1 +="\r";//風速CR
158     str1 +="\r";//風向CR
159     str1 +="\r";//雨量CR
160     str1 +="\r";//土壌水分CR
161     str1 +=co2s; //追加センサ1【CO2】
162     str1 +="\r";//追加センサ１CR
163     str1 +="\r";//追加センサ２CR
164     str1 +="\r";//追加センサ３CR
165     str1 +="\r";//追加センサ４CR
166     str1 +="";//追加センサ5CR
167
168
169     //データをGatewayに送信
170     Serial.print(str1);
171
172
173     //1分間の待機
174     delay(1000*60UL);
175
176   }
177
178   //loop関数ここまで
179
180
181
182
183   //UVセンサ用関数
```

```
184
185  //Analogreadの平均値を戻り値とする関数
186  int averageAnalogRead(int pinToRead)
187  {
188    byte numberOfReadings = 8;
189    unsigned int runningValue = 0;
190
191    for(int x = 0 ; x < numberOfReadings ; x++)
192      runningValue += analogRead(pinToRead);
193    runningValue /= numberOfReadings;
194
195    return(runningValue);
196  }
197
198
199  //UV強度の算出
200  float mapfloat(float x, float in_min, float in_max, float
     out_min, float out_max)
201  {
202    return (x - in_min) * (out_max - out_min) / (in_max -
     in_min) + out_min;
203  }
204
205
```

```
/*****************************************************
*  BME280 気圧・温湿度センサ用
*  センサ値を取得およびGatewayへの送信用
*  GatewayにはRaspberry Piを用いること
*  電源には9V電池を使用
*  実験及び打ち上げにはArduino Uno Rev3を使用
*
*  スケッチ作成日  2016年8月
*
*  参考にしたスケッチ
*  ・Grove wiki
*****************************************************/


#include "Seeed_BME280.h"
#include <Wire.h>
#include <PString.h>

BME280 bme280;

void setup()
{
  Serial.begin(9600);
  bme280.init();
}

void loop()
{
  float pres;
  float tmp = bme280.getTemperature();//温度
  pres = bme280.getPressure();//気圧
  float pre=pres/100.0;
  float alt = bme280.calcAltitude(pres);//高度
  float hum = bme280.getHumidity();//湿度

  char tmpc[20],prec[20],altc[20],humc[20];

  dtostrf(tmp,7,2,tmpc);
  dtostrf(pre,7,2,prec);
  dtostrf(alt,7,2,altc);
  dtostrf(hum,7,2,humc);

  char buffer1[200];
  PString str1(buffer1,sizeof(buffer1));
```

```
45
46      str1 +="\r";//指定CR
47      str1 +="S\r";//ヘッダ固有文字列＋CR
48      str1 +="0013A200\r";//固有番号＋CR
49      str1 +="40ADA4C0\r";//固有番号＋CR
50      str1 +=prec;//気圧
51      str1 +="\r";//気圧CR
52      str1 +=tmpc;//温度
53      str1 +="\r";//温度CR
54      str1 +=humc;//湿度
55      str1 +="\r";//湿度CR
56      str1 +="\r";//照度CR
57      str1 +="\r";//風速CR
58      str1 +="\r";//風向CR
59      str1 +="\r";//雨量CR
60      str1 +="\r";//土壌水分CR
61      str1 +="\r";//追加センサ１CR
62      str1 +=altc;//追加センサ２(高度)
63      str1 +="\r";//追加センサ２CR
64      str1 +="\r";//追加センサ３CR
65      str1 +="\r";//追加センサ４CR
66      str1 +="";//追加センサ5CR
67
68       Serial.print(str1);
69
70      delay(1000*60UL);
71  }
```

13. 参考・引用資料

・Raspberry Pi：http://www.vesalia.de/e_Raspberry Pi.htm

・BME280：https://www.seeedstudio.com/Grove-TempHumiBarometer-Sensor-BME280-p-2653.html

・ベースシールド：https://www.switch-science.com/catalog/1293/

・XBee：https://www.switch-science.com/catalog/2611/

・Arduino ワイヤレスプロトシールド：https://www.switch-science.com/catalog/786/

・Arduino：https://www.arduino.cc/en/Main/ArduinoBoardUno#overview

・温湿度センサ：https://www.switch-science.com/catalog/819/

・紫外線センサ：https://www.switch-science.com/catalog/1697/

・CO2：https://www.seeedstudio.com/Grove-CO2-Sensor-p-1863.html

・3g トラッカー：http://www.trackers.jp/products_tr313j.html

・ezfinder：http://ezfinder.com.tw/

・Galaxy SⅡ LTE モデル
http://www.samsung.com/jp/consumer/mobilephone/smartphone/docomo/SGH-N034DANDCM

・Grove wiki,Sparkfun 社サイト：http://wiki.seeed.cc/Grove_System/

・2013ExamineProject「高層気象観測バルーン」(第 2 回気象観測機器コンテスト)

・2014ExamineProject「Cloud Examine」(第 3 回気象観測機器コンテスト)

・2015ExamineProject「ガッスン」(第 4 回気象観測機器コンテスト)

・2014 気象観測機器コンテスト「KOINOBORI PROJECT」

・2015 気象観測機器コンテスト「ヘイズカスーミー　～視界距離自動判定装置～」

優秀賞

賞金 10 万円

海塩観測器「塩見鳥」

長崎県立宇久高等学校

海塩観測機「塩見鳥（しおみどり）」

長崎県立宇久高等学校　１年　宮　崎　風　香
西　　有香子

１．はじめに

私たちが住む宇久島は，日本の西端にある長崎県のさらに西方海上にある五島列島最北端に位置します(図１～３).城ヶ岳(じょうがたけ)259m を最高点とする火山性地形がなだらかに広がる面積 24.9km^2 の火山島です.

図１．長崎県五島列島の位置

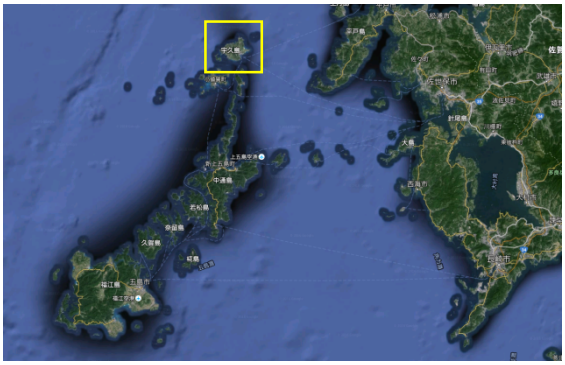


図２．五島列島の中の宇久島の位置

図３．宇久島

東海岸には砂丘があり，牧草地と砂丘を利用した無料のゴルフコースもあります．そのほかの海岸には，海岸段丘が発達しています.

島の人口は約 2100 人，かつては宇久町，合併後の現在は長崎県佐世保市に属しています．毎年 100 人のペースで人口が減少し続けています.宇久高校の全校生徒は２３名，私たち１年生は１１名です.

交通手段は船舶のみで，カーフェリーと高速船がそれぞれ１日２便ほどあり，宇久と佐世保を結んでいます.

このような島の生活は，海の日々の状況に大きく影響を受けます.

波の高さが３m程度になると高速船が欠航し，波の高さが４m程度でフェリーも欠航となります．船が欠航すると物資が届かなくなり，商店の棚から物がなくなりますので，天気予報などでは，波の高さなど海に関する情報を注意深く聞いています.

また，まわりを海に囲まれていますので，海からの風に含まれる塩類によって，塩害が生じます.

塩害とは，土壌中や空中の塩分によって，農作物や建造物,施設などが被害を受けることで,土壌中の塩分濃度が高くなることによる塩土害と，強い風によって海水の飛沫が吹き上げられて内陸に運ばれることによる潮風害に大別されています（日本大百科全書ニッポニカ）.宇久島では特に潮風害が発生しています.

島には多くの車や建物がありますが，それらは常に潮風にさらされており，腐食の被害を受けています．また，島では農業も盛んですが，作物もまた，潮風を受けることによる塩害を受けています．

塩害を防ぐには，塩分が付着した後，なるべく早期に水洗いをすることが大切ですが，島の水資源は限られていますので，毎日，すべての車や建物そして作物を水洗いするわけにもいきません．

そこで，私たちは水資源を保全しながら塩害を防ぐ，海塩観測機「塩見鳥」の作成し，島の生活の向上，ひいては塩害に苦しむ世界の人々に役立てていただくことを目指しました．

２．塩見鳥の機序

海水の中には，塩化ナトリウムや塩化マグネシウムそして硫酸ナトリウムなど，さまざまな塩類が溶け込んでおり，海水の飛沫が陸地の構造物や作物などに付着することで塩害が発生しています．そこで塩見鳥では，垂直に立てたポリプロピレンの板の表面を伝って常に蒸留水を流下させて空気中の塩類を溶け込ませ，その水溶液の電気伝導度を測定することで，空気中の塩類の存在量を推定します．電気伝導度は温度の関数ですので，水溶液の温度も測定し，25℃における電導度に換算して比較します．

測定の際に，流下面積を変化させたり，風にあてる期間を変化させたりすることにより，塩類を測定する感度を調整することができます．

また，風見鶏のように，その瞬間の値を得ることもできますが，ある期間の積算塩量を測定することもできます．これにより，付着塩類の濃度が一定以上に達した際に「塩害警報」を出すことができます．塩害警報をもとに効果的に洗浄を行うことで，塩害を防止するとともに，島で貴重な水資源の保全につなげることができそうです．

また，塩類は風に乗ってやってきますので，どの方向に面した建物の壁に塩類がたくさんつくかを測定できるようにしたいと考えました．

そのために，塩見鳥は，東西南北の４つの方角に法線ベクトルを向けた４平面についてそれぞれ蒸留水を流下させて，それぞれ独立に電気伝導度を測定しています．これにより，塩類の飛来が卓越する方向を知ることができます．

さらに，塩見鳥は見た目ですぐ塩類卓越方向がわかるように，電気伝導度の大きい方位をひよこの像（図４）が向くように設計します．

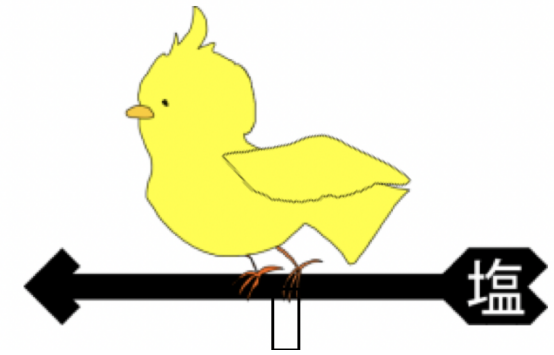


図４．ひよこ像

塩類卓越方位を知ることで，建物などの壁面を洗浄する際に，優先的に洗浄すべき場所を知ることができ，このことからも水資源を節約することができます．

付着塩類の濃度を，ひよこの羽の角度でも表現します．塩類が検出されていない場合を０度として，電気伝導度の上昇とともに羽の角度を立てていき，90 度の角度で塩害警報に相当するよう調整します（図５）．

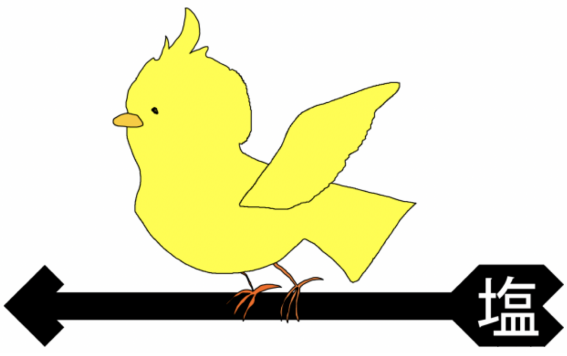


図５．塩害警報時

電気伝導度の測定およびひよこ像の回転そして羽の角度調整は，raspberryPi に接続したマイコン arduino を介して行います．電源は，内部にバッテリーを備え，発電はひよこ像の上に着けたソーラーパネルを使って行います．発電履歴は，日照時間や直達日射量の測定にも利用できます．

データは，RaspberryPi arduino と xBee チップを用いて無線通信で送信します．島内の複数箇所で測定し，得られたデータはサーバに蓄積します．塩害の発生が予想される場合，自動的にメールアドレス登録者にメールを送信し対策を促します．

また，ソーラーパネルの横には，植物を置き，塩害の発生を植物の反応で検出できるようにします．植木鉢には，土壌センサを設置し，土壌の含水量および塩類の濃度から，植物が置かれているシチュエーションを「植物の声」として予め自分（宮崎）の声を録音したメッセージをスピーカーから流して声で伝えます（表１）．

表１．植物の声

【水分飽和】
こんにちは
いつも世話してくれてありがとう
今日も元気だよ
一緒に遊ぼう
気持ちいいなー

【水分含有量　高】
お水ください
元気でない
かまってよー
水がほしいなあ
うてあえ！（宇久島の方言：かまって）

【水分含有量　中】
水がたりないよ！
おなかすいたなぁ・・・
さびしいなあ・・・
水をくれー
のどがかわいた

【水分含有量　低】
からからだよー
助けてー！
限界だよー！！
事件です！！
忘れないで

【水分含有量　極めて低い】
枯れちゃうよー
おなかぺこぺこ・・・
のどが砂漠状態・・・
危険です！！
誰かああ

【乾燥】
これまでありがとう
く，くるしい
もうダメ・・・
お別れだね・・・
ばいばい

【塩分濃度　極めて低い】
いい感じ！
今日も絶好調！
散歩に行きたいな
おっけー
ハッピー

【塩分濃度　低】
なんかしょっぱい
やばいかも
塩を感じる
海風反対
べたべたする

【塩分濃度　中】

まあまあ塩ある
しょっぱいぞ
やられるー
あぶないです
厳しいかも
苦しい

【塩分濃度　やや高い】

結構しおある
塩多すぎー
むりだー
しょっぱい！！
助けてください
塩対応やめてください

【塩分濃度　高】

危機的に塩ある
しょっぱくて枯れそうだよ！
何とかして
逃げたいよー
どこかへ非難させて！
ヘルプミー！！

【塩分濃度　極めて高い】

しぬ
耐えられない
つらいよう
やられたー
死んじゃう
枯れます

【乾燥あるいは高塩分から復帰したとき】

死ぬかと思った
ありがとう
三途の川でじいちゃんに会ってきたよ
生き返ったー
やっぱり君は最高だね

表１ここまで

それぞれの状況に応じたメッセージを５パターンほど準備し，ランダムに選択して流します．メッセージを流すタイミングは１０分間隔です．

３．実験方法

空気中の塩類のもととなる海水を宇久島すげ浜でサンプリングしました（図６図７）．

図６．すげ浜

図７．2L ペットボトルに３本分採取しました．

当日はとても風が強く，波も高い日でした．

得られた海水は，蒸発乾固させて濃度を測定すると，33 パーミルでした．

塩分濃度測定のための標準溶液を調整しました（図８）．

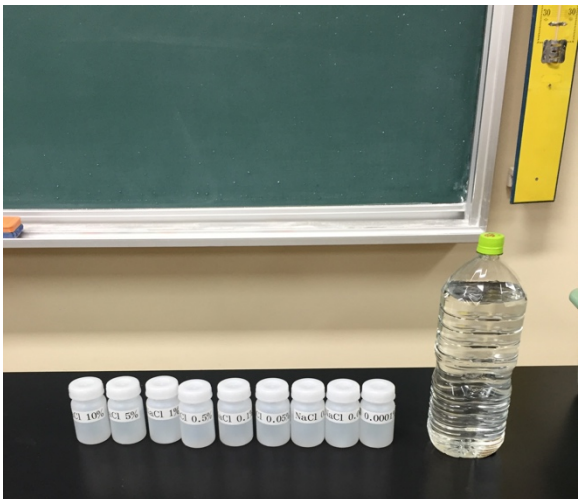


図８．濃度測定用の標準溶液

水溶液の塩分濃度は電気伝導度を測定することで得られます．今回は，Arduino で抵抗を測定し，抵抗の逆数である電気伝導度を算出しました．電極は固定ピッチのピンを流用しました（図９）．

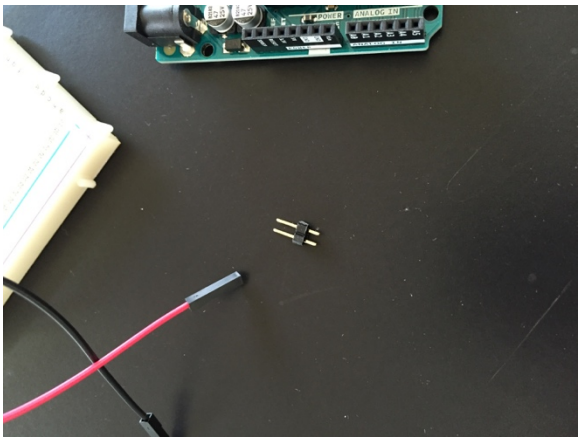


図９．電気伝導度測定用の電極

本来ならば，１m^{2}の電極板を１m 間隔でならべて測定すべきですが，標準溶液を測定して標準曲線を描いて校正することで，正確な測定ができると考えました．

塩分測定部は，ポリプロピレンの容器を切ってつくります（図 10）．ここに 100ml の水道水を入れて５V 駆動のポンプで常時循環させて，空気中の塩類を捕まえます．蒸留水ではなく水道水を使ったのは，塩類の検出が容易であることと運用が容易になることが理由です．蒸留装置がない場所でも，手軽に塩類測定ができます．

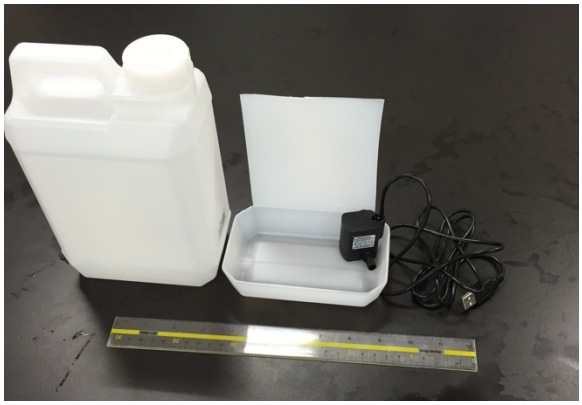


図 10．塩分濃度測定部

図 10 の測定器を４つつくり（図 11），４方向に並べて設置し，塩類飛来の卓越方向を調べます．

図 11．４つの測定部

ひよこは raspberryPi（図 12）にサーボモータを取り付けて動かします．像は発泡スチロールの板から切り出してつくりました（図 13）．

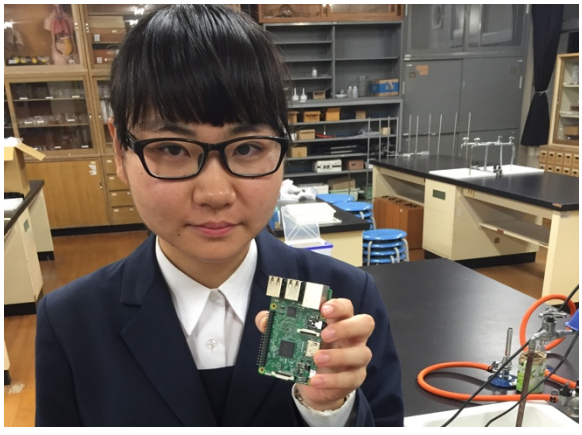


図 12．ラズベリーパイ

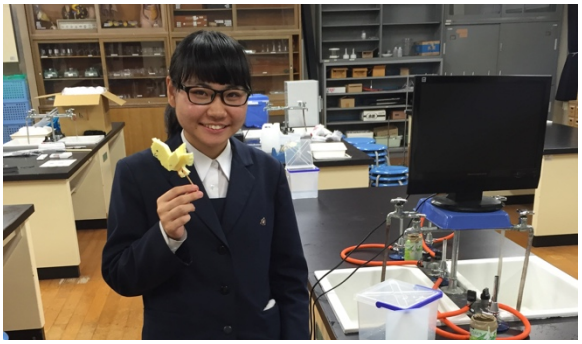


図 13.　ひよこ像

ソーラーパネルの横には,植物を置きます(図 14).

図 14.　塩松

塩害を検出するので，名前を「塩松」としました.「死を待つ」と同じ発音で縁起が悪いですが,塩害検出のために設置する意味を考えると，少々かわいそうですが，この名前にしました.

これらのパーツを組み上げると，塩見鳥ができます（図 15).

塩見鳥は高さ１mはほどで，イレクターパイプで組んであり，中央のケースの中にバッテリーやラズベリーパイ，arduino などを格納しています.

植物およびひよこの像，そして流下する測定液の組合せから，公共施設や幼児園などでも気軽に設置していただけるよう，デザイン面でも工夫しました．電源は，二次電池とソーラーパネルによる独立系，データ送信は無線式とし，半月程度はメンテナンスフリーのシステムの構築を目指しました.

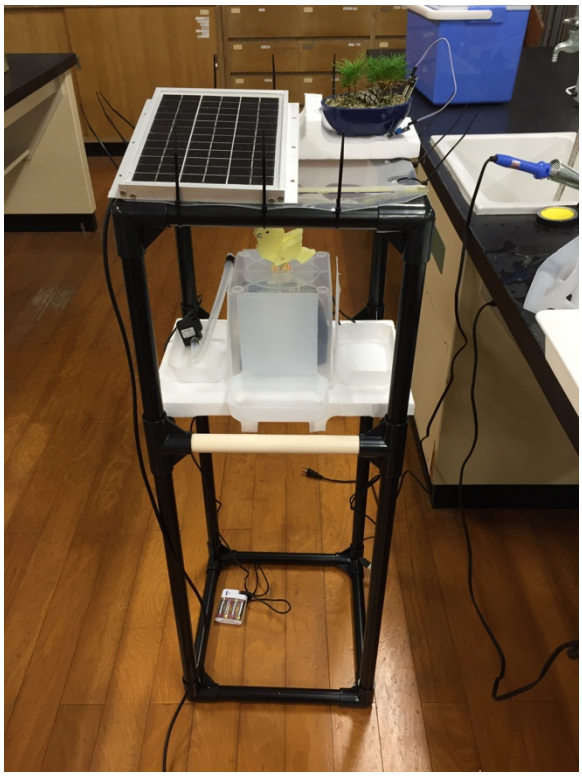


図 15.　塩見鳥

４．結果と考察

標準溶液の抵抗および電気伝導度を測定したところ，表２のような結果が得られました.

表２．測定結果

溶液・濃度	抵抗Ω・m	μS／cm
0.5	2.59	3859
0.1	4.54	2201
0.05	7.58	1319
0.01	19.7	507
0.001	29.4	340
0.0001	30.9	323
水道水	31.9	313

標準曲線を描くと，図 16 のようになります.

電気伝導度と塩分濃度はとても密接に関係しており，微量の塩類の添加でも電気伝導度は敏感に反応し値の上昇を見せることがわかりました．これにより，電気伝導度を測定することで

極微量の塩類の飛来を検出できることがわかりました.

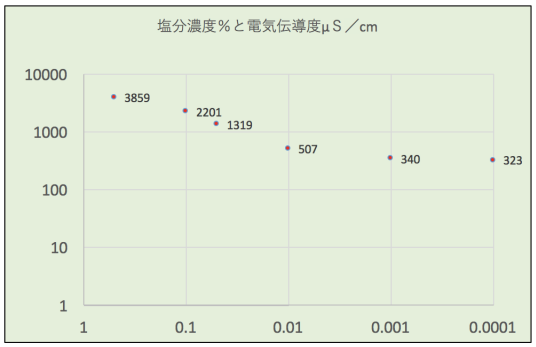


図 16. 塩分濃度　対　電気伝導度

５．まとめ

電気伝導度を測定測定することで，飛来塩類を極低濃度から検出できました，

離島である宇久島で空気中の塩類濃度測定を確立し，塩見鳥を完成させることで，日本や世界に多数ある離島での塩害防止に役立てたいという思いから本研究をはじめました．より内陸での大規模農業など，空気中の塩類に敏感な作物を扱っている方にも利用していただきたいです.

さらに，塩類による被害は人体にも及ぶ（ロシアで発生した塩類による視力障害など)ため，世界各地で空気中の塩類測定に活用していただきたいと考ます.

本研究を実施するにあたり，ＷＮＩ気象文化創造センターには，手厚い助成と発表の機会を賜りました．感謝申し上げます.