公立はこだて未来大学 2023 年度 システム情報科学実習
グループ報告書

Future University Hakodate 2023 Systems Information Science Practice

Group Report

プロジェクト名

Interaction Elements - 『未来を形作る部品』を作ろう

Project Name

Interaction Elements - Creating Elements for Future

グループ名

グループ A, B, C

Group Name

Group A, B, C

プロジェクト番号/Project No.

5

プロジェクトリーダ/Project Leader

田中琳仁　Rihito Tanaka

グループリーダ/Group Leader

工藤翔太　Shota Kudo
米原孝星　Kosei Yonehara
森重龍　Ryo Morishige

グループメンバ/Group Member

田中琳仁　Rihito Tanaka
米原孝星　Kosei Yonehara
井上芽依　Mei Inoue
板垣智也　Tomoya Itagaki
工藤翔太　Shota Kudo
樫木海生　Kai Kashiki
島田麻飛　Ashahi Shimada
冨田夏央　Natsuo Tomita
村上太一　Taichi Murakami
大塚創生　Sousei Otsuka
菅英寛　Hidehiro Suga
橘孝則　Takanori Tachibana
森重龍　Ryo Morishige

指導教員

安井重哉　塚田浩二　伊藤精英

Advisor

Shigeya Yasui　Koji Tsukada　Kiyohide Ito

提出日

2024 年 1 月 17 日

Date of Submission

January 17, 2024

概要

日本語の概要を書く。Interaction Elements とは、人が外界の環境 (身の回りの実世界や、コンピュータの中の仮想世界など) と相互行為を行う際に用いる要素のことである。例えば、人が動かすことで電気がつく照明のスイッチや、マウスなどの人がクリックすることでコンピュータの中のボタンが押され通信するなどの行為を可能にする部品のことを指す。本プロジェクトは、今までにはなかった、未来を形作る Interaction Elements を制作することを目的としている。今年度は、最終成果物としてこすり出しを楽しむペン「Copypen」、風を心地よく可視化するブラインド「BLWIND」、全方向の雨から守る傘「雨守り」を制作した。3 人から 6 人で構成されるチームを 3 チーム作り、グループ同士相互交流を積極的に行いながら制作を行った。

（※文責: 大塚創生）

Abstract

Abstract in English. Interaction Elements are elements that people use to interact with the external environment (the real world around them, the virtual world inside a computer, etc.). For example, a light switch that turns on when a person moves it, or a mouse that enables a person to click a button on a computer and communicate with it. This project aims to create Interaction Elements that have never existed before and that will shape the future. This year, as final products, we created "Copypen," a pen to enjoy rubbing out, "BLWIND," a blind to visualize wind comfortably, and "Amamori," an umbrella to protect from rain in all directions. 3 teams consisting of 3 to 6 members per team name were conducted, and production was conducted while actively engaging in mutual exchange.

（※文責: 大塚創生）

目次

第 1 章　導入

1.1　プロジェクトの目標・目的

本プロジェクトは、未来を形作る Interaction Elements を作ることを目的としている。Interaction Elements とは、人が外界の環境（身の回りの実世界や、コンピュータの中の仮想世界など）とインタラクションを行う際に用いる構成要素のことである。一例として、照明のスイッチやドアノブがあり、身近には様々な Interaction Elements が身近に存在する。本プロジェクトでは人間の五感を利用し、今までにない面白い体験を与える Interaction Elements を制作する。今年度は、こすり出しを楽しむペン「Copypen」、風を心地よく可視化するブラインド「BLWIND」、全方向の雨から守る傘「雨守り」を制作した。

（※文責: 冨田夏央）

1.2　組織

プロジェクト構成員は学生 13 名と教員 3 名である。意見を積極的に交換できる環境を構築するために、プロジェクト全体のリーダー・副リーダーを設定した上で、プロジェクトマネージャー・グループリーダー・議事録係・Web 制作班などを設定した。 実際に Element 制作を行うのは 3 グループであり、「Copypen」を制作した 6 人で構成される A グループ、「BLWIND」を制作した 4 人で構成される B グループ、「雨守り」を制作した 3 人で構成される C グループである。これらのグループは、プロジェクトメンバーの全員がいずれか 1 つのグループに所属している。制作のグループとは別に、中間発表会・成果発表会で用いる Web サイト・ポスター・動画・発表資料などを制作・実演するグループに分かれ、それぞれのメンバーが役割を見つけ、役割に沿って協力し制作を行った。このグループについては、自身がやりたいと思えるグループに集まり、所属を決定した。 また、本プロジェクトメンバーは、今までにない面白い体験を与える Interaction Elements を制作するという考えの下活動している。また、1 チームのエンジニアが他チームの制作物に携わることや、議事録係が全体の制作風景や過程を記録したり、フォトグラファーが制作物の撮影を行うなど、グループを跨いだ活動を行った。以下では、私たちがプロジェクト活動を行った手順について記述する。

（※文責: 冨田夏央）

1.3　プロジェクト活動の手順

手順 1 ついついやってしまうこと図鑑の作成

Interaction Elements を制作するにあたり、アイデア出しのため、日常生活に潜む「心地いいこと」「ついついやってしまうこと」をフィールドワークで集めて「ついついやってしまうこと図鑑」を作成した。「ついついやってしまうこと図鑑」は行為、五感、対象、気づきの視点に分けて分類・分析を行った。図 1.1 のような 100 種類を超える多くの「ついついやってしまうこと」が集まっ

た。これにより、各メンバーの興味のあることを視覚化し、分類することが出来るようになった。

図 1.1　ついついやってしまうこと図鑑

（※文責: 冨田夏央）

手順 2 図鑑を KJ 法で分類、それを元にアイデアを考える

手順 1 で出した断片的なアイデアや情報を分類ごとに効率的に整理するため、集まったついついやってしまうことを図 1.2 のように付箋を使用してそれぞれ KJ 法で分類した。今回は、視覚や聴覚などの五感や「心地よい」、「不安や欲求からなる行為」などの特徴ごとで分類を行った。さらに、図 1.3 のように分類したジャンルに注目して作りたいものやあったらいいなと感じるアイデアを考えた。さらに指導教員から多くの論文などを教えていただいた。一例を [1.2-1] に示した。

図 1.2　ついついやってしまうこと図鑑

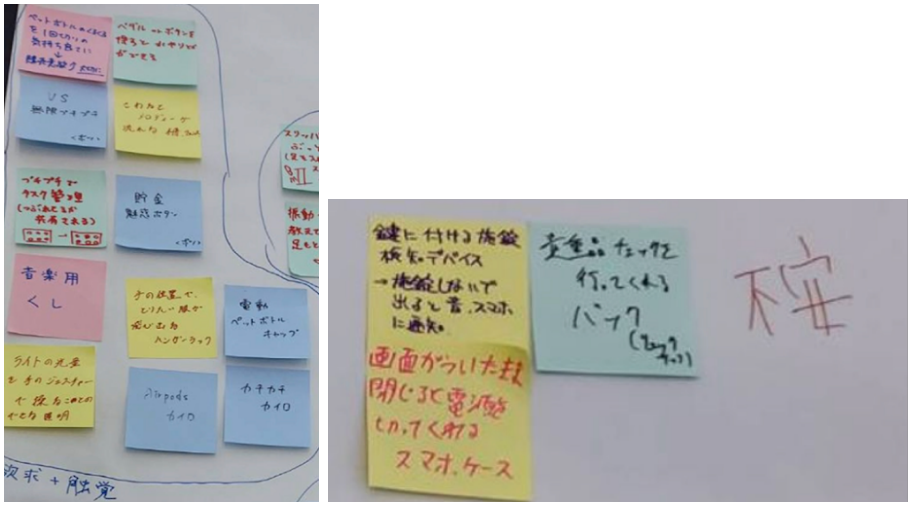

図 1.3　ネタ帳

（※文責: 冨田夏央）

手順 3 アイデアを KJ 法で分類し、五感を中心とした分野で絞り込む

図 1.4 のように模造紙とアイデアを書いた付箋を用いて、五感と特徴ごとに集めたアイデアをさらに KJ 法で分類した。この際、KJ 法では、似ている要素を持つもので分類を行い、五感を中心とした分野で絞り込んだ。

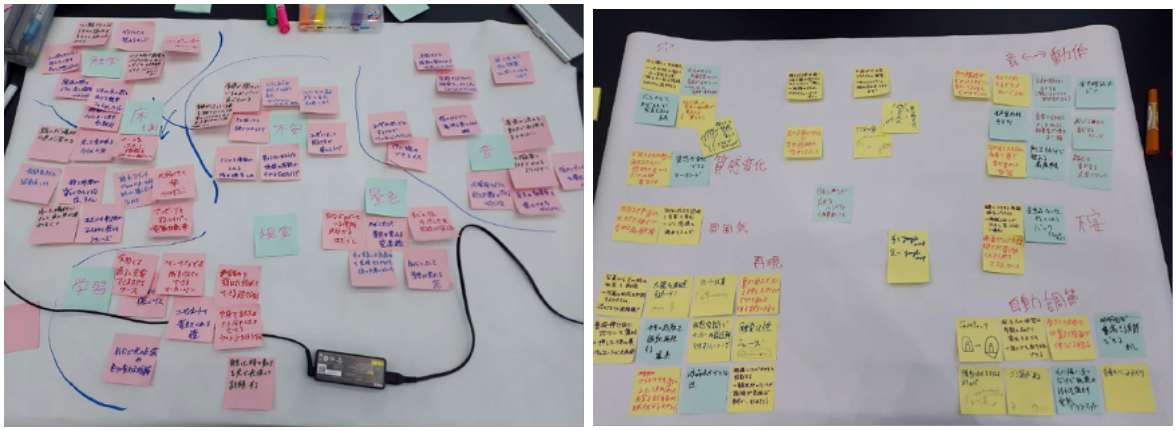

図 1.4　アイデアの分類後

（※文責: 冨田夏央）

手順 4 3 つのグループに分かれる

五感を中心とした分野に分類したあと、特に面白いと思ったアイデア、作りたい分野にそれぞれが集まり、３つのグループに分かれた。ここで、触覚に興味を持ったグループを A、触覚と触覚・音に興味を持ったグループを B、音に興味を持ったグループを C とした。こうしたグループ分けを経て、各メンバーが同じ目的に向かって作業することが出来た。

（※文責: 冨田夏央）

手順 5 グループごとに具体的なアイデア出し

制作をスムーズに進めるため、各グループごとに実際に制作する Element の方向性やアイデアを出した。そのアイデアを教授を交え全員で交流会を行い、様々な視点からアイデアを精査した。これにより、具体的なイメージをグループ内で共有できるようになった。

（※文責: 冨田夏央）

手順 6 コンセプトシートの作成

各グループごとに Element のコンセプトを決定するコンセプトシートを作成した。このシートを作成する上で、Element の機能、期待される効果、その対象やイメージスケッチを記載するなど、フォーマットを決定した。これにより、Element の制作イメージをグループ内で共有し、解像度を高めることが出来た。

（※文責: 冨田夏央）

手順 7 グループごと Element の制作（中間発表会まで）

3 つのグループに分かれたあと、中間発表までの目標をそれぞれの Element の動作イメージが伝わるようにすることして、グループごとに活動を行った。グループ内の役割分担としてリーダー、PM を設定し、グループごとに Notion でスケジュールを共有しながら計画的に活動できるように工夫した。また、毎週水曜日には学生だけでのグループごとの進捗報告を行い、毎週金曜日には先生を含めた進捗報告を行った。先生を含めた進捗報告での技術や進め方などについての意見をもとに PM を中心にスケジュールを調整しながら、制作を進めた。進捗報告を定期的に行うことで、グループ内からは出なかった考え方や異なる視点を得ることができた。

（※文責: 冨田夏央）

手順 8 中間発表資料の作成

中間発表に向けて、発表スライド、コンセプト動画、ポスターを作成した。各グループの発表の原稿や Element の作成班の他にスライドとポスターを含むデザイン班、動画班で分け、PM がこれを統括した。発表スライドとポスターは Adobe Illustrator を用いて作成し、動画は Adobe Premiere Pro、Adobe AfterEffects を用いて作成した。また、スライドや動画では情報をさらに伝わりやすくするために 3DCG アニメーション作成ソフトの Blender、 動画編集ソフトの Aviutl などのソフトも使用した。また、プロジェクトの認知度の向上や全体でのイメージの統一のためにプロジェクト全体のロゴの作成（図 1.5）も行った。

図 1.5　中間発表時のロゴ

（※文責: 冨田夏央）

手順 9 グループごと Element の制作（成果発表会まで）

前期の反省を活かし、中間発表会を通して発生した様々な課題を解決しつつ、各 Element を改善するため活動を続けた。前期と同様に PM を中心にスケジュールを調整しながら、完成目標に到達出来るよう、発表リハーサルに向けて制作を進めた。最終的には、完璧にスケジュール通りにとはいかなかったが、各 Element は成果発表会までに概ね完成できた。

（※文責: 冨田夏央）

手順 10 成果発表資料の作成

成果発表会に向けて、発表スライド、コンセプト動画、Web サイト、ポスターを作成した。中間発表会の資料作成（手順 8）同様、デザイン班、動画班、Web 班で分け、PM がこれを統括した。ポスターは Adobe Illustrator を用いて作成し、動画は Adobe Premiere Pro、Adobe AfterEffects を用いて作成した。前期とは違い、発表スライドは Figma でフィードバックをしながら作成した。また、スライドや動画では情報をさらに伝わりやすくするために 3DCG アニメーション作成ソフト Blender も使用した。これらの発表資料は、中間発表時点での資料よりも詳細に、分かりやすくなるよう作成した。また、評価シート用の QR コードを記載したスタンドの制作、web を自由に閲覧できるブースなどを設営し、発表をより良いものとできるように工夫を行った。学生からのフィードバックからも分かる通り、成果発表会に向けて作成した様々な資料により各 Element の特徴やメリットを具体的に伝えることが出来ていた。

（※文責: 冨田夏央）

第 2 章　各 Element 制作

2.1　Element.01 「Copypen」

2.1.1　目標・目的

本 Element の目的はペンにインタラクションを持たせることにより、ペンを動かすことが心地よいと思えるような体験と、画像を浮かび上がらせる魔法のような体験をユーザに提供することである。本グループは触覚に関する Element 制作に興味を持ったメンバーで構成されている。触覚に働きかける Element についてアイデア出しを行った結果、画像を擦ってスキャンしてこすり出しの要領で書き出せるペンを着想した。こすり出し (フロッタージュ) とは、表面に凹凸がある物体に紙をのせて鉛筆などでこすると、凹凸が模様となって写される絵画技法のひとつである。メンバー同士でのアイデア出しの際、こすり出しの凹凸に鉛筆が当たる感覚が面白く心地よいという意見が挙げられ、メンバー全員でそのことを確認した。また、凹凸のある物体が無くても好きなイラストや画像をこすり出すことができれば、魔法のような体験ができるのではないかと考えた。つまり、「好きな画像をスキャンしてデータとして保存し、ユーザが紙にペンをこするだけでその画像を書き出せるペン」が発案された。このペンにはこすり出しの心地よさや、こするだけでイラストが描画できる面白さに加えて、ペンさえあれば好きなイラストをいつでもどこでも描画できる便利さなどにも期待ができた。これらによりコピー&ペーストができるペン「Copypen」とプロダクト名をつけ、本グループの制作目標として定めて活動を開始した。

（※文責: 板垣智也）

2.1.2　Element の制作手順・方法

方針の決定

本グループのメンバーは人間の五感である「触覚」に着目し、触覚を介したインタラクションを備えている Element の制作を行いたいと考えているメンバーで構成されている。プロジェクトのグループ活動開始当初は、凹凸の上でも平面の上かのように描くことのできる「触覚キャンセリングペン」の制作の話が出ていた。他にも、ぬるぬるとした面をぬるぬるを感じずに描くことのできるペンや、硬さや柔らかさを感じずに描くことのできるペン、圧力により色と濃さを制御するペン、天地無用を守る箱、転がらないコップのような案も出ていた。最終的な案として、画像の濃淡を解析してその濃淡に合わせて凹凸を作り出す平面型デバイスと、本 Element である Copypen が残った。平面型デバイスでは、運用するにあたってデバイスと紙とペンの 3 つの要素が必要であるのに比べて、ペン型デバイスであれば、デバイスと紙の 2 つの要素だけが必要である。ペンだけで実装機能を完結させることができるという利便性から、グループメンバー間での話し合いの後、ペン型デバイスである Copypen が採択された。名前についても「擦り刷り」や「印刷えんぴつ」のような案が多数出ていたが、メンバー間の投票により「Copypen」と決定し、制作を開始した。

グループメンバーは計 6 人いる。半分の 3 人ずつに分けて、それぞれ外装班と機能班に分かれることで効率的な制作を目指した。しかし、明確に外装班・機能班に分かれているわけではなく、そ

れぞれの班で困ったことがあったり、人員が足りない時には班の枠組みを飛び越えて活動をしていた。そのため、各班の進捗に差はなく、順序良く制作過程を踏むことに繋がった。

以下では、制作過程で用いたセンサやマイコンについて、グループで触れた順番に解説をしていく。

（※文責: 樫木海生）

Copypen に必要なセンサを学ぶ (5/12〜6/1)

M5StickC

中間発表時点まではマイコンとして M5StickC を使用していた。なぜこのマイコンを使用していたかというと、最初に「Copypen」を作ろうとしていた段階で、プロジェクトスペースに最初からあったからというのもあったが、非常に小さくコンパクトな点が、ペンである以上はできるだけ小さいものにしたいという需要とマッチしていたためである。しかし中間発表後に、2 つ以上のセンサを接続したり、読み取り部分に使用するマウスを接続するための拡張パーツを接続するのが困難な点や、マイコンが小さくても結果的にはセンサや配線の部分で大きくなってしまうという結論になったため、マイコンを M5Stack Gray に変更することとなった。

（※文責: 村上太一）

Copypen 本体作成（中間発表）(6/1〜7/16)

M5StickC Plus

中間発表時点まではマイコンとして M5StickC を使用していたが、もう一つのマイコンとして、M5StickC より新しい機種である M5StickC Plus の利用も視野に入れていた。プロジェクトとして用いることができる機材の中にあったので、運用することはできた。しかし、いくつかの理由から採用されたのは M5StickC であった。確かに、M5StickC より M5StickC Plus の方が性能は良く、ディスプレイも大きく、それに加えて中古品ではなく新品を用いることもできたが、M5StickC と M5StickC Plus ではプログラムコードが微妙に異なっているためにプログラムの改変をする必要があった。プロジェクトが始まって、序盤から M5StickC を利用していた GroupA にとっては、M5StickC Plus に機種を変更するメリットが感じられず、逆にデメリットを大きく感じたため、利用を断念した。これは Element の制作には入らないが、M5StickC Plus は実は中間発表・成果発表のときに利用されていた。Copypen 自体にこのマイコンが含まれているわけではないのだが、GroupA の発表スライドをめくる際のリモコンとして活用されていた。

（※文責: 樫木海生）

M5StickC Servo Hat

中間発表以前の、M5StickC を使っていた当初から、「Copypen」にはサーボモータを使っていたので、M5StickC Servo Hat は M5StickC に対応したサーボモータであり、最も簡単に実装、接続できるサーボモータであると考えていたため採用されていた。しかし、M5Stack Gray を使うことになったためそもそも使用することが困難になったほか、接続できたとしても、M5StickC Servo Hat は接続に配線を使っていないため、マイコンとサーボモータの位置を離して配置することができなかったため、適切ではないということになり、使用をやめた。

（※文責: 村上太一）

超音波センサ HC-SR04

座標を取得するセンサとして中間発表以前に使用していたセンサである。任意の壁を作成し、その壁との距離を測るという形で座標を取得していたのだが、M5StickC に３つ以上のセンサを接続するためには複雑な手順を必要とし、結果的に自分たちでは１つの M5StickC に 3 つのセンサを接続することはできなかった。そのため、２つのマイコンを用いて超音波センサを制御しようとしたのだがラグが起こりまともに動かすことができなかったうえ、超音波センサを２つ以上同時に使おうとすると超音波が干渉してしまって距離を測れなくなってしまう現象が発生した。そのため、中間発表の時点では超音波センサは１つで使用していたのだが、座標を１方向からしか取得することができなかったため、中間発表後に、２方向を１つのセンサで取得できる実質上位互換となるマウスが使用できるようになったことで、超音波センサを使う案はなくなった。

（※文責: 村上太一）

モックアップの作成

内部構造の設計を行いやすくするために、本格的な外装設計を行う前に、ダンボールで外装を設計した。内部構造に M5StickC、超音波センサ、サーボモータ及びペンを出し引きする機能を格納でき、かつペンとして握ることができるよう極力小さいサイズにするため、ダンボールを縦 17cm、横 4cm の長方形に切り取ったものを 4 枚用意した。それらを透明セロテープで繋ぎ合わせて角柱の形にして、角柱の形が崩れてしまわないよう対辺に三角形の支えをダンボールで作成するなどの工夫を施した。

（※文責: 工藤翔太）

MDF でプロトタイプ制作

プロトタイプの作成には、M5StickC、M5StickC Servo Hat、超音波センサ HC-SR04 を用いた。内部に取り付ける各パーツの配置を決定したことによって、外装のサイズを決定した後、5mm の MDF をレーザーカッターで加工することによりプロトタイプを作成した。前述したモックアップに倣って角柱の形状とし、切り取る長方形の大きさも同じサイズとした。しかし、ダンボールと比較すると MDF の方が厚みがあるため、内寸が 5mm ほど狭くなってしまい、さらにサイズの調整が必要になった。また、前面には超音波センサの金属ケースの部分のみを露出できるよう直径 1.5cm の穴を２箇所開けた。さらに、M5StickC が水平方向に移動しないよう両側面を支えるパーツを制作するなど、精度が高くなるための工夫を行った。

（※文責: 工藤翔太）

最終発表会に向けた Copypen 制作 (9/10〜)

マイクロサーボ SG-90

中間発表を経て、GroupA のメンバーでの話し合うと、以前まで制作に用いていた M5StickC 用の Servo Hat を用いたままだと自由度が小さいために理想のペンの動きが追求できないという結

果に至った。また、それに伴いマイコンの変更も視野に入れていたため、Servo Hat では運用が難しいと考えた。そこで、配線部分が長いために自由度が高く、比較的運用しやすいマイクロサーボ SG-90 を制作に用いることにした。このサーボモータに変更したことにより、外装はこのサーボモータに合わせた設計にしなくてはならなくなった。3D プリンタでのパーツ作成も SG-90 の大きさ・配置に合わせた再設計を行った。さらに、プログラムについても変更を余儀なくされた。しかし、プロジェクト開始当初、M5StickC や Arduino Uno で SG-90 を動作させていたときのコードが残っていたり、そもそも動作させることが容易であったりという理由からプログラムの変更についてはさほど影響はなかった。結果としては、変更したことにより自由度が高くなったために、新たにラックアンドピニオン機構の採用を検討することになり、理想のペン先の動きを実現することが可能となった。変更過程では移行テストや再設計・再印刷等の作業を強いられたが、最終的な成果に繋がるものであった。

（※文責: 樫木海生）

カラーセンサ、フォトリフレクタ

イラストを読み取る機構を制作するにあたり、色を読み取るセンサとしてカラーセンサを使用していた。カラーセンサのメリットとして、Grove 端子によって M5StickC と接続できる点があげられる。そのため、安定したデータの受信や組み立てのしやすさ、配線数を少なくできるなどの利点が多かった。また、RGB 値がとれるため正しい色が取れているかのテストも簡単にできた。しかし、デメリットとしてイラストの 1 ドットを大きくしなければ、正確に読み取れないというものがあった。カラーセンサは色と色の境界を読み取ろうとすると、その 2 つを混ぜた色がデータとして送信される。色の混ざらない 1 ドットの大きさを計測したところ、およそ 10mm 四方であり、グループとしては許容できない大きさであった。また、床に接地するサイズも大きく、ペンの太さを考え直す必要があった。そのため、使用するセンサをカラーセンサからフォトリフレクタに変更した。フォトリフレクタは 1 ドットがおよそ 8mm 四方で、接地面もカラーセンサより大幅に小さい。ユニバーサル基板を用いた配線が必要なことや固定がしにくいデメリットがあるものの、フォトリフレクタを読み取る機構として採用することにした。

（※文責: 板垣智也）

フォトリフレクタとその配線

フォトリフレクタはダイオードから赤外線を放出し、反射した光をフォトトランジスタにより受け取ることで出力電流を変化させるセンサである。例えば、物体が近くにあれば赤外線がすぐに反射され、強い赤外線を受け取る。逆に物体が遠ければ、弱い赤外線を受け取る。「Copypen」はイラストの明暗を、この仕組みを利用して判別している。黒色は光を吸収する性質があるため、至近距離で赤外線を当てても返ってくる量は少ない。白色は光を反射するため、受け取る赤外線は強くなる。つまり、受け取った赤外線の強さによって、明暗を判別している。また白と黒の 2 値ではなく、1000 以上の細かい値でとれるため、受光のブレにも対応することができた。フォトリフレクタを M5StickC や M5Stack Gray で使用するときには、ブレッドボードにフォトリフレクタや抵抗などを挿して使用していた。しかし、Copypen の外装にブレッドボードを入れるスペースがなく、ブレッドボードを用いないでフォトリフレクタを使えるようにすることを求められた。この問題を解決するために、ユニバーサル基板での基盤作成に取り組んだ。工房職員の方に教わりなが

ら、基盤の設計やはんだ付け、空中配線などを駆使してフォトリフレクタとマイコンとのブレッドボードを用いない配線を作り上げた。

（※文責: 板垣智也）

3D プリンタでのパーツ作成

まず、Copypen の書き込み機能を実装するにあたって、ペン先を押引する機構が必要だったため、はじめはプロトタイプとして M5StickC の Servo Hat に鉛筆の芯を固定する簡易的なパーツを作成し、中間発表まで使用していた。このパーツと機構には、数えきれない程の問題があったため、成果発表会に向けパーツの改善を開始した。その問題については後述していく。次の段階では、新しく採用した SG-90 というサーボモータでピニオン（歯車）を回転させて、平板状の棒に歯切りをしたラックを直線に動作させる「ラック＆ピニオン」というパーツを 3D プリンタで作成した。この段階ではペンを押引させるという当初の目的は果たせていたが、ペン先の摩擦によるブレや小型化、ペンの固定が不十分など様々な問題点があった。また、「こすり出し」をコンセプトにしていて鉛筆の芯をペン先に採用していたため、ペン先の摩耗により書き込めなくなるという重大な問題もあった。それを踏まえて、次に作成したパーツは、ラック内部にバネとペン先を組み込んだボールペンのような役割を果たすものだった。このパーツはペン先の摩擦によるブレは多少改善しており、バネを組み込んでいたので鉛筆の先が摩耗してもある程度は書き込み続けることが出来ていた。しかし、この機構は非常に脆く、不安定であったためにいまだ改善の余地があった。また、この段階でサーボモータが動作の影響で動いてしまっていたため、本体にネジで固定できるパーツも作成していた。この時点で第一に解決しなければいけない問題は、「ペンの固定不十分によるブレ」であった。これを解決するため、3D プリンタによるペンプロッタに目を付けた。これは 3D プリンタのノズル部分にペンを固定するパーツを付けて、3D プリンタに正確に文字やイラストを書かせるというものである。このペンを固定するパーツというものが今回の目的に沿っていたため、参考にして最終的なパーツを作成した。この最終的なパーツというのは、ネジで抑え込むことでペンを完全に固定し、ペンの摩擦は輪ゴムの反発で抑える機構であった。このパーツをピニオン（歯車）で上下させることで今までの問題を概ね解決した機構を完成させることが出来た。そしてこのパーツはペンの種類を自由に変更することも出来た。太くて見やすいサインペンや筆ペンなど様々なペンで実験したが、本来のコンセプトである「こすり出し」のイメージに近く、書き心地が良いことから最終的には鉛筆を使用することになった。また、サーボモータに取り付けたピニオン（歯車）が非常に不安定であったため、直接取り付けるのではなく、サーボモータ付属のサーボホーンにはめ込むことで固定出来るパーツに更新した。この際の接着は当初ホットボンドで行っていたが、数回破損したため、プラスチック用の接着剤で固定することになった。そして、何度も Copypen 本体の内寸が変更されていたため、成果発表会の直前に最終的な内寸に合ったサーボモータの固定パーツを作成した。ここまでのパーツ更新で当初の目的であった、ブレのない安定した書き込み機構を実装することが出来た。これらすべてのパーツのモデリングは、Fusion360 を用いて行った。また、3D プリンターでの印刷の際、充填率や品質を向上させることができるため、本番で組み込むパーツや動作に大きく影響するパーツは時間をかけて印刷し、やすりで余分な部分を削ったりすることで、より精度を高められるよう工夫した。

（※文責: 冨田夏央）

マウスの固定

座標取得に用いるマウスセンサを本体底面に固定するパーツも 3D プリンタで作成した。このパーツはマウスセンサに直接取り付け、片側の壁にはネジ、もう片方には穴にさして固定する物だった。マウスセンサは 1mm 浮いていても、本体底面から飛び出ていても正常に動作しなくなるほど精密な物であったため、このパーツのモデリングは非常に緻密な計算が必要であった。このパーツを実装したことで、マウスが地面に接地せず位置を取得出来ないという問題と固定不十分によるブレが大きいという問題が解決された。しかし、このパーツは設計上非常に脆く、1 度破損したが、成果発表会までに時間が無かったため、そのままのモデルを最後まで使用した。また、このパーツも本体内寸に合わせて設計されていたパーツのため、成果発表会直前に調整を行い、再印刷した。

（※文責: 板垣智也）

M5Stack Core2

M5Stack シリーズの第 2 世代目。性能だけで見るなら実際に「Copypen」に使われた M5Stack Gray よりも高性能。また、M5Stack Gray は絶版であるため手に入りにくいことを考えても一見するとこちらを使わない理由は無いように見える。しかし「Copypen」の座標取得のために使用していたマウスを接続するための拡張パーツである USB モジュールを接続することができず涙をのんだ。USB 接続もしくは Bluetooth 接続でマウスの接続を実現できていたら使用される未来もあったかもしれないが、いろいろ試しても実現できなかったため、このマイコンは我々に扱いきれるスペックではなかったということかもしれない。

（※文責: 村上太一）

M5Stack Gray

中間発表後に選ばれたマイコン。M5StickC よりも大きいがセンサを 2 つ以上つなげることができる。M5Stack Core2 の項目にも書いたが、M5Stack Gray を使用した理由は、マウスを接続できる USB モジュールが Core2 には使用できず Gray には使用可能であったからである。サイズを上げたことによって全体のサイズが大きくなってしまうと思ったが、センサや配線の兼ね合いによって結局サイズは大きくなってしまっていたため、大きな問題にはならなかった。画面を「Copypen」の動作の補助を表示させるのに使っているほか、ボタンにも機能を割り当てていたため、画面とボタンを搭載しているマイコンであるメリットを生かし切ることもできたと思う。そのため、「Copypen」はこのマイコンでなくてはならなかったといえる。

（※文責: 村上太一）

補正アルゴリズム

読み取り時のデータにノイズがあったり、データの欠けがあったりするという問題をデータの補正アルゴリズムを用いることによって解決した。また、メディアンフィルタや膨張、縮小アルゴリズムでも補正ができるか試したがうまくいかなかった。また、補正アルゴリズムの詳しい説明に関しては、「2.2.3 プログラムの解説」で記述する。

（※文責: 島田麻飛）

マイコン変更後の外装制作

使用するマイコンを M5StickC から M5Stack Gray に変更したため、外装の内寸を大きく変更する必要があった。M5StickC は縦横 2cm × 4cm ほどの大きさだったため、縦向きに取り付けることができれば面幅を 2cm ほどで抑えることができたが、M5Stack Gray は縦横 5.5cm × 5.5cm の大きさがあったため、地面に垂直な方向に取り付けることを前提とすると最小でも 5.5cm の面幅が必要となる。

そのため、面幅が 5.5cm になった時、ペンとして許容できるかどうかを検証するため、MDF で外装を制作する前に、MDF と同じ 5.5mm の厚みのあるハレパネを用いてモックアップを再度作成した。検証の結果、ペンとしては非常に大きなサイズとはなったものの、片手で握ることが可能であったため、この大きさで内寸を変えずに制作することを決定した。

また前提として、制作するエレメントがペンであったため、六角形である方がイメージに則るのではないかという意見が同時に出てきたが、面幅 5.5cm の六角柱を作るとなると、片手で握ることがほぼ不可能となり、かつ MDF で六角柱を作ることはそもそも難しいと工房の指導教員からアドバイスを頂いたため、面幅 5.5cm の四角柱で MDF プロトタイプを設計することにした。面同士が接着剤無しでも垂直方向に結合できるよう、各面の接着辺に切り込みを入れ、レーザーカッターで 0.05mm の補正を入れる工夫を施した。

また、前面には M5Stack の液晶及びボタン部分が露出できるよう、面に 5.5cm × 5.5cm の穴を開けた。さらに、3D プリンタで作成したサーボモータの固定パーツや、マウスポインタの固定パーツを取り付ける穴を開けた。

前述した、マイコン変更前の MDF プロトタイプを用いて内部構造を設計していた際、面同士の接着が透明のセロテープであったため、固定がしっかりされておらず、精度がブレてしまうという問題があった。そこで、面同士をネジで固定するパーツを制作することで解決可能であるという助言を工房の指導教員に頂き、同じ 5.5mm の MDF を用いて固定パーツを作成した。

（※文責: 工藤翔太）

Blender を用いた内部構造の設計

外装のサイズを決定し、MDF でプロトタイプを作成した後、内部にあるフォトリフレクタ・サーボモータ・マウスポインタ・ラックアンドピニオン及びラックに接着する鉛筆の芯がどのように配置されるかを共有するため、Blender を用いて内部構造を設計した。設計した内寸に各パーツが収まるかどうかを把握するため、M5Stack を含む、内部に含まれるパーツを実寸大で設計した。

設計の結果、M5Stack・サーボモータ及びラックアンドピニオンなどのパーツは配置可能であることが分かったが、M5Stack を充電する際に必要となるバッテリーを配置することが難しいということが分かった。そのため、5mm 厚の MDF で作成した外装の形状を一部変更し、M5Stack の側面が露出するように Copypen の側面に適切なサイズの穴をくり抜いた。このことから、Copypen を制作する上で Blender による内部構造の設計は十分に効果があった。

また、設計を進めていくうちに、Blender で内部構造を理解しやすく表現することができれば、成果発表の際に観客にも見てもらうことでより理解を深めてもらうことができると考え、視覚的に理解しやすいようにテクスチャを調整し、より実体のパーツのビジュアルに近づける工夫を施し

た。Blender 上に机と紙を配置し、実際に GroupA のロゴをペーストしたような画像を出力した。

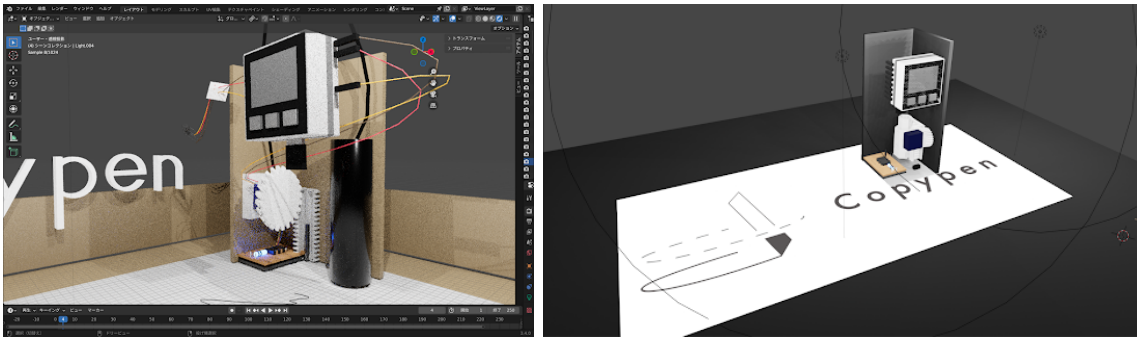

図 2.1　blender で製作した Copypen のイメージ図

（※文責: 工藤翔太）

MDF の塗装

本 Element の制作段階で、シンプルなビジュアルにすることを決めていたため、Copypen 本体の外装を白色に塗装することにした。レーザーカッターを用いて MDF を指定のサイズに切断した後、より綺麗な白色に仕上げるため、いくつかの工夫をした。まず、切断線についたヤニがあると綺麗に塗装することができないため、細かめの紙ヤスリを用いてヤニを取り除いた。次に、ムラなく均等に着色できるよう、ホワイトスプレーをする前にグレーのサーフェイサーを 2~3 回吹きかけた。その後、ホワイトスプレーを 3~4 回全体に吹きかけ、外装を綺麗な白色にすることができた。

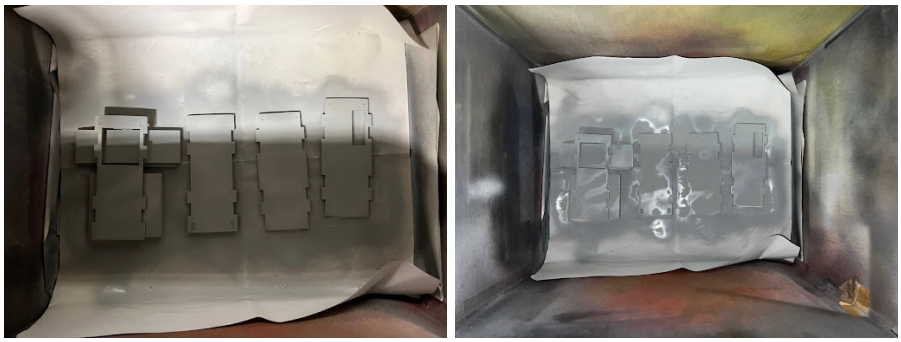

図 2.2　blender で製作した Copypen のイメージ図

（※文責: 工藤翔太）

白アクリルへの変更

塗装した MDF を使用して内部構造を設計していると、「内部に各パーツを全て収めることができるかどうか厳しい」「木材に近い素材であるため重い」「白く塗装した部分がだんだん禿げていく」などの問題が発生した。これらの問題を解決するために、素材を 5mm の MDF から 3mm の白アクリルに変更した。これにより、重さがかなり軽量化され、薄いため内寸に余裕ができ、塗装の劣化も防ぐことができた。

（※文責: 工藤翔太）

展示用 Copypen の制作

成果発表会で内部構造をより視覚的に理解しやすく説明するため、白アクリルで作成したプロトタイプとは別に、展示用の Copypen を作成した。外装の形状及び内部の構造は全て白アクリルで作成したものと同じ状態にした。また、白アクリルで作成したプロトタイプが正常に動作しなかったり、デモ実施時に故障してしまった時のために、透明アクリルのプロトタイプでも同じ動作ができるようにした。

（※文責: 工藤翔太）

「Copypen」のロゴデザイン

「Copypen」のロゴは、Adobe Illustrator で作成した。このロゴは、「Copypen」の Copy の”C”、Pen・Paste の”P”をかたどった四角形のパーツで構成されている。上項で説明したロゴアニメーションの作成のために、直線のパスで描かれている。こうしたことで、Pen の機能である書き込みを再現しやすくなった。また、全体図は∞ (無限) の形を描くようデザインされている。これは、Copy & Paste が無限に出来る、という意味が込められている。正方形を組み合わせた形にすることで、Copypen 本体の正方形の形のイメージを落とし込んでいる。

（※文責: 冨田夏央）

2.1.3 プログラムの解説

本 Element では PlatformIO for VisualStudioCode と ArduinoIDE を使用して開発を進めた。ここでは、それらのプログラムについての解説をする。最終的なコードは末尾に付録として記載する。

以下に PlatformIO for VisualStudioCode の環境構築、使用したライブラリを示す。

（※文責: 樫木海生）

```
[env:m5stack-grey]
platform = espressif32
board = m5stack-grey
framework = arduino
lib_deps =
        m5stack/M5Stack@^0.4.6
        felis/USB-Host-Shield-20@^1.6.0
        madhephaestus/ESP32Servo@^1.0.0
```

今回の制作では、M5Stack Gray をマイコンとして選んだため、board には m5stack-grey を選んだ。また、M5Stack 用の USB モジュールを用いるため、USB-Host-Shield のライブラリを使用している。マイクロサーボも用いているため、ESP32Servo のライブラリを使用している。

（※文責: 樫木海生）

```
#include <M5Stack.h>
```

```
#include <Usb.h>
#include <hidboot.h>
#include <hiduniversal.h>
#include <usbhub.h>
#include "M5Mouse.h"
```

今回の制作には M5Stack や USB モジュール、USB マウスを用いているため様々なライブラリを include している。M5Mouse.h は PlatformIO for VisualStudioCode の既存ライブラリに無かったため別でプログラムを用意して include している。

（※文責: 樫木海生）

```
#define WindowWidthSize 10000
#define WindowHeightSize 10000
#define ArraySize 80
```

我々のグループで実験を繰り返し行った結果、Copypen の可動範囲としてマウスから得られる位置情報を 10,000、Copypen の記憶領域を 80 × 80 とした。

（※文責: 樫木海生）

```
int PIN = G2;
int mode = 1;
int StaPotX = 160, StaPotY = 120;

int count = 0;
int flag = 0;
int HoseiCount = 0;

int Canvas[ArraySize][ArraySize] = {};
int Copy[ArraySize][ArraySize] = {};
```

今回の制作に用いている変数である。count は後述する読み取り機能の補助である補正プログラムで用いている。flag は後述する書き込みモードで M5Stack Gray の画面に図形を描画する際に用いている。HoseiCount は後述する補正プログラムでの 2 段階補正の切り替えに用いている。また、配列 Canvas は 80 × 80 の大きさであり、読み取り機能で読み取った情報を保存しておくための配列である。配列 Copy は Canvas に保存されている情報をさらに保存しておくための配列である。

（※文責: 樫木海生）

```
void draw()
{
  ledcWrite(PWM_CH, 5000);
}
void stop()
{
  ledcWrite(PWM_CH, 7000);
```

```
8 }
```

名前を draw()、stop() としてペンの制御を行う関数を宣言している。ledcWrite() はマイクロサーボ SG-90 を制御するプログラムである。通常、デジタル信号しか出すことができないが、細かくオンオフを繰り返すことで平均としてアナログ出力を出している。今回 ledcWrite() の GPIO として PWM_CH を用いている。サーボの動作に伴ってペンの出し引きをしている。

（※文責: 樫木海生）

マウスポインタ

```
1  void Mouse_Pointer(int PotDataX, int PotDataY)
2  {
3    if (0 < (StaPotX + PotDataX) && (StaPotX + PotDataX) <= WindowWidthSize)
4      StaPotX = (StaPotX + PotDataX);
5    else if ((StaPotX + PotDataX) <= 0)
6      StaPotX = 0;
7    else
8      StaPotX = WindowWidthSize - 1;
9    if (0 < (StaPotY + PotDataY) && (StaPotY + PotDataY) <= WindowHeightSize)
10     StaPotY = (StaPotY + PotDataY);
11   else if ((StaPotY + PotDataY) <= 0)
12     StaPotY = 0;
13   else
14     StaPotY = WindowHeightSize - 1;
15 }
```

ここのプログラムではマウスから得た座標のデータの変化量を基に、マウスカーソルの現在位置を計算している。関数の引数として PotDataX、PotDataY を用いており、今回のプログラムの場合、PotDataX、PotDataY にはそれぞれ include している”M5Mouse.h”の mou_px、mou_py が指定されることになる。”M5Mouse.h”内の mou_px、mou_py はマウスから得る座標のデータの変化量である。それらの変数と宣言した StaPotX、StaPotY とを加減算することで、StaPotX、StaPotY をマウスの現在位置としている。ここで、M5Stack Gray の画面に描画する際の都合上、taPotX、StaPotY が取り得る値の範囲は WindowWidthSize 及び WindowHeightSize で制限されている。

（※文責: 樫木海生）

待機モード

ここでは待機モードについて解説する。

```
1 int locX = StaPotX / 125, locY = StaPotY / 125; // 1.6cm = 16mm = 1,000
```

locX、locY は Mouse_Pointer() で得た StaPotX、StaPotY を 125 という値で割ることにより、配列の添字を取得している。125 という値は我々のグループでマウスポインタについての実験を繰り返し行った結果、今回の配列の添字を取得するのに適した数字であったために用いた。

（※文責: 樫木海生）

```
M5.Lcd.setTextDatum(1);
M5.Lcd.drawString("Copypen", M5.Lcd.width() / 2, M5.Lcd.height() / 2 - 10, 4);
```

待機中の画面に Element の名前を表示するために、Lcd 表示を操作する関数である M5.Lcd.setTextDatum()、M5.Lcd.drawString() を用いて画面の中央付近に”Copypen”という文字を表示している。

（※文責: 樫木海生）

```
if (M5.BtnA.isPressed())
{
    M5.Lcd.fillScreen(BLACK);
    StaPotX = 0;
    StaPotY = 0;
    Serial.println("yomikomi_kaishi");
    mode = 2;
    delay(500);
}
```

Copypen には 3 つのモードを搭載している。そのモードの変更を行うために M5Stack Gray のボタン操作を用いている。待機モード上で M5Stack Gray のボタン A を押した際は、読み込みモードに移るようにした。その手順を以下に示す。まず前の画面を黒く塗りつぶす。その後、StaPotX と StaPotY を 0 にする。つまりは、マウスの座標を初期化して、モニター画面の左上にマウスの座標が来るようにしている。次に、読み込みモードに移ったことが分かるように”yomikomi_kaishi”とシリアルモニタ上に表示する。次に、mode を 2 に指定する。これにより、モードの変更が行われる。最後に delay(500) を挟んで待機モードを終了する。

（※文責: 樫木海生）

読み込みモード

ここでは読み込みモードについて解説する。

```
int val = analogRead(PIN);
val = 256 - map(val, 2700, 4095, 0, 255);
```

この文ではフォトリフレクタで取得した値を val という変数に代入している。関数 map() を用いることで、フォトリフレクタが取得する 2,700~4,095 という範囲の値を 0〜255 に矯正している。さらに、その値と 256 の差を取ることにより、フォトリフレクタで得た値を擬似的にグレースケールに変換している。val は 0〜256 の値を取るが、値が小さい方が黒色、大きい方が白色としている。

（※文責: 樫木海生）

```
M5.Lcd.drawLine(39, 0, 39, 240, WHITE);
M5.Lcd.drawLine(281, 0, 281, 240, WHITE);
```

読み取った絵の情報を画面に表示する際の補助線を引いている。M5StackGray は画面の縦横比が

3:4 であるため、正方形の情報を表示しようとすると画面の端に余白ができてしまう。余白のバランスを調整するために画面の中央から均等な距離に縦線を引いている。

（※文責: 樫木海生）

```
Canvas[locY][locX] = val;
```

配列 Canvas にフォトリフレクタで読み取った値を格納している。locX と locY を配列の添字に指定することで、val を対応している x 座標、y 座標に格納することが可能になっている。

（※文責: 樫木海生）

```
if (Canvas[locY][locX] < 150)
{
    M5.Lcd.fillRect(locX * 3 + 40, locY * 3, 3, 3, RED);
}
else
{
    M5.Lcd.fillRect(locX * 3 + 40, locY * 3, 3, 3, WHITE);
}
```

値を読み取ると同時に M5Stack Gray の画面にも読み取った値に合わせて表示を行っている。読み取った値が黒色に近いなら赤色、白色に近いなら白色でその読み取った座標に合わせて M5Stack Gray の画面にもその座標の位置に描画を行う。これにより、こすって絵を読み取ると同時に M5Stack Gray の画面に描画されるので視覚的なフィードバックを得ることが可能になっている。

（※文責: 樫木海生）

```
if (M5.BtnA.isPressed()) // 書き込みモード
{
    M5.Lcd.fillScreen(BLACK);
    StaPotX = 0;
    StaPotY = 0;
    Serial.println("kakikomi_kaishi");
    mode = 3;
    delay(500);
}
```

読み込みモード上で M5Stack Gray のボタン A を押した際は、書き込みモードに移るようにした。その手順を以下に示す。まず前の画面を黒く塗りつぶす。その後、StaPotX と StaPotY を 0 にする。次に、書き込みモードに移ったことが分かるように”kakikomi_kaishi”とシリアルモニタ上に表示する。次に、mode を 3 に指定する。これにより、モードの変更が行われる。最後に delay(500) を挟んで読み込みモードを終了する。

（※文責: 樫木海生）

```
if (M5.BtnB.isPressed())
{
    StaPotX = 0;
```

```
    StaPotY = 0;
    delay(500);
}
```

M5StackGray のボタン B を押した際は、マウスの座標を初期位置に戻すことができる。これにより、途中で読み取りがずれてしまった場合に読み取りをやり直すことができる。最後に delay(500) を挟んでいる。

（※文責: 樫木海生）

```
if (M5.BtnC.isPressed())
{ // 保存
    for (int t = 0; t < ArraySize; t++)
    {
      for (int s = 0; s < ArraySize; s++)
      {
        Copy[t][s] = Canvas[t][s];
      }
    }
    M5.Lcd.fillScreen(BLACK);
    Serial.println("hozon_shimashita");
    delay(500);
}
```

M5Stack Gray のボタン C を押した際は、宣言している t と s の値が二重の for 文内で変化していき、配列 Canvas の要素全てを見ることができる。つまり、Canvas に保存されている全要素を配列 Copy にコピーすることができる。読み取った好きな絵をユーザ自身の操作によって保存することが可能である。保存した後は前の画面の黒く塗りつぶし、シリアルモニタ上にも”hozon_shimashita”と表示をすることで保存行為を行ったことが視覚的に分かりやすいようにした。最後に delay(500) を挟んでいる。

（※文責: 樫木海生）

```
if (Canvas[locY][locX] < 150)
{
    draw();
}
else
{
    stop();
}
```

読み込みモードで配列 Canvas に格納した値を読み込みの方式と同じく locX、locY を用いて対応している座標上で、その配列に格納されている val 値に合った draw() と stop() を呼び出している。今回は白色と黒色の境目を val でいう 150 としているため、val が 150 未満であると draw() が呼び出され、150 以上であると stop() が呼び出されるようになっている。つまり、読み込んだ色が黒に近ければ書き込みのときにはペンを出す draw() を呼び出し、逆に読み込んだ色が白に近け

れば書き込みの時にはペンをしまう stop() を呼び出している。

（※文責: 樫木海生）

```
if (flag == 0)
{
    for (int i = 0; i < ArraySize; i++)
    {
        for (int j = 0; j < ArraySize; j++)
        {
            if (Canvas[i][j] < 150)
            {
                M5.Lcd.fillRect(j * 3 + 40, i * 3, 3, 3, RED);
            }
        }
    }
    for (int i = 0; i < ArraySize; i++)
    {
        for (int j = 0; j < ArraySize; j++)
        {
            if (Canvas[i][j] >= 150)
            {
                M5.Lcd.fillRect(j * 3 + 40, i * 3, 3, 3, WHITE);
            }
        }
    }
    flag = 1;
}
```

読み込みモードでは読み取った軌跡を画面に表示していたが、その軌跡をドット状に描画しているのがここのプログラムである。Canvas の要素を for 文で全て見ていき、150 の境目で区分けして赤色と白色で画面に描画している。ここで、変数 flag によって if 文が制御されているが、後述する Copypen の現在位置を M5Stack Gray の画面に描画するプログラムにおいて不都合が生じてしまうためである。常に画面に読み取り軌跡が描画され続けると、現在位置が見えなくなってしまうため、flag という変数で書き込みモードに移行した時に 1 度だけ描画するようにしている。

（※文責: 樫木海生）

```
if (Canvas[locY][locX] < 150)
{
    M5.Lcd.fillRect(locX * 3 + 40, locY * 3, 3, 3, 0x02E9);
}
else
{
    M5.Lcd.fillRect(locX * 3 + 40, locY * 3, 3, 3, BLUE);
}
```

ここのプログラムでは Copypen の現在位置を M5Stack Gray の画面に表示している。書き込み時にどこまで書き込んだのかを可視化するために、白色と赤色とは違う色を用いての描画としている。

（※文責: 樫木海生）

```
if (M5.BtnA.isPressed())
{
  M5.Lcd.fillScreen(BLACK);
  StaPotX = 0;
  StaPotY = 0;
  flag = 0;
  stop();
  for (int i = 0; i < ArraySize; i++)
  {
    for (int j = 0; j < ArraySize; j++)
    {
      Canvas[i][j] = 190;
    }
  }
  Serial.println("taikishimasu");
  mode = 1;
  delay(500);
}
```

書き込みモード上で M5StackGray のボタン A を押した際は、待機モードに移るようにした。その手順を以下に示す。まず前の画面を黒く塗りつぶす。その後、StaPotX と StaPotY を 0 にする。また、flag も 0 に初期化する。次に stop() でサーボを初期位置に戻す。これにより、サーボが出ている状態で読み取ることがないようにする。次に、配列 Canvas に 190（基準値）を格納する。次に、待機モードに戻ったことが分かるように”taikishimasu”とシリアルモニタ上に表示する。次に、mode を 1 に指定する。最後に delay(500) を挟んで書き込みモードを終了する。

（※文責: 島田麻飛）

```
if (M5.BtnB.isPressed())
{
  if (HoseiCount == 0)
  {
    for (int t = 0; t < 16; t++)
    {
      for (int s = 0; s < 16; s++)
      {
        count = 0;
        for (int i = 0 + 5 * t; i < 5 + 5 * t; i++)
        {
          for (int j = 0 + 5 * s; j < 5 + 5 * s; j++)
          {
```

```
14              if (Canvas[i][j] < 150)
15              {
16                count++;
17              }
18            }
19          }
20          if (count >= 4)
21          {
22            for (int a = 0 + 5 * t; a < 5 + 5 * t; a++)
23            {
24              for (int b = 0 + 5 * s; b < 5 + 5 * s; b++)
25              {
26                Canvas[a][b] = 0;
27              }
28            }
29          }
30        }
31      }
32      HoseiCount++;
33    }
34    else
35    {
36      for (int t = 0; t < 16; t++)
37      {
38        for (int s = 0; s < 16; s++)
39        {
40          count = 0;
41          for (int i = 0 + 5 * t; i < 5 + 5 * t; i++)
42          {
43            for (int j = 0 + 5 * s; j < 5 + 5 * s; j++)
44            {
45              if (Canvas[i][j] < 150)
46              {
47                count++;
48              }
49            }
50          }
51          if (count <= 3)
52          {
53            for (int a = 0 + 5 * t; a < 5 + 5 * t; a++)
54            {
55              for (int b = 0 + 5 * s; b < 5 + 5 * s; b++)
56              {
57                Canvas[a][b] = 255; // 白
58              }
```

```
59              }
60            }
61          }
62        }
63        HoseiCount = 0;
64      }
65      flag = 0;
66      delay(500);
67    }
```

ボタン B を押すのが一回目の場合、HoseiCount == 0 の if 文に遷移する。このコードは縦横 5 × 5 の 25 マスの配列分ごと精査するコードになっている。配列 Canvas の値の中身が 150 未満なら count++ をして count の値を＋ 1 した。その後、count の値の中身が 4 以上なら 5 × 5 の 25 マスの全ての配列の値を 0 にする。要するに、 25 マスの内 4 マス以上黒が配列に入っていたら、25 マス全て黒にするというコードである。その後、HoseiCount++ をして HoseiCount の値を＋ 1 する。

ボタン B を押すのが 2 回目の場合、HoseiCount の値が 1 なので、HoseiCount == 1 の if 文に遷移する。上記と同じように、縦横 5 × 5 の 25 マスの配列分ごと精査をし、Canvas 配列の値の中身が 150 未満なら count++ をして count の値を＋ 1 した。その後、count の値の中身が 3 以下なら 5 × 5 の 25 マスの全ての配列の値を 255 にする。要するに、25 マスの内、黒が 3 マス以下の場合 25 マス全て白にするというコードである。その後 HoseiCount の値を 0 にする。

その次に、flag の値を 0 にする。

上記のような補正のアルゴリズムを行うことでイラストの補間と、ノイズの除去を行うことができる。

また、他にもメディアンフィルタを試した。メディアンフィルタとは、画像のノイズを除去する手法の 1 つで、ある画素を、周りの画素の濃度の中央値に変換することである。メディアンフィルタは、画像を平滑化することなく、画像のエッジ部分をそのまま残してノイズが除去できるという特徴がある。メディアンフィルタでは、3 × 3 や 5 × 5 のような正方形の領域の中心の画素を、周りの画素の濃度の中央値に変換する。例えば、3 × 3 の画素が「2、1、3、1、8、4、3、1、2」と並んでいる場合、小さい順に整列して「1、1、1、2、2、3、3、4、8」とする。中央値は 5 番目の「2」になるため、「8」の画素の濃度を「2」に変換する。

しかし、このメディアンフィルタでの補正アルゴリズムではノイズの除去はできても、イラストの補間ができないため採用しなかった。

他にも、膨張、収縮処理を試した。「膨張」と「収縮」は、画像処理や信号処理などの分野で使用される操作である。これらの操作は、主に画像中の特定の領域を強調したり、ノイズを取り除いたりするために利用される。以下にそれぞれの操作について簡単に説明する。

膨張（Dilation）: 膨張は、画像中の物体や特定の領域を拡大する操作である。この操作では、各画素の周りにある近傍領域を調べ、その中で最大の値を選択して元の画素の値とする。これにより、物体や構造が強調され、小さな穴が埋められたり、細い構造が太くなったりする。ノイズの影響を減少させる効果もある。

収縮（Erosion）: 収縮は、画像中の物体や特定の領域を収縮させる操作である。この操作では、各画素の周りにある近傍領域を調べ、その中で最小の値を選択して元の画素の値とする。これにより、物体や構造が収縮され、小さな突起が取り除かれたり、穴が拡大されたりする。ノイズを取り

除く効果がある。

これらの操作は、ノイズの除去や物体の検出、形状の修正など、さまざまな画像処理のタスクで使用される。また、膨張と収縮を組み合わせたり、複数回繰り返すことで、さらなる効果を得ることができる。加えて、この補正もメディアンフィルタと同様に、望み通りの補正結果を得ることができなかったので採用しなかった。

（※文責: 島田麻飛）

```
if (M5.BtnC.isPressed())
{
  for (int t = 0; t < ArraySize; t++)
  {
    for (int s = 0; s < ArraySize; s++)
    {
      Canvas[t][s] = Copy[t][s];
    }
  }
  flag = 0;
  delay(500);
}
```

M5StackGray のボタン C を押した際は、配列 Canvas に読み取りモード時に保存した配列 Copy の中身を渡す。その後、flag の値を 0 にする。最後に delay(500) を挟んでいる。

（※文責: 樫木海生）

setup

ここでは setup 内のプログラムについて解説する。

```
for (int i = 0; i < ArraySize; i++)
{
  for (int j = 0; j < ArraySize; j++)
  {
    Canvas[i][j] = 190;
  }
}
```

プログラム実行後、最初に配列 Canvas の全要素に 190 という基準値を格納している。190 という値はフォトリフレクタの読み取り実験を重ねて行っていった結果得られた val の値として適当な値である。

（※文責: 樫木海生）

```
Serial.println("M5USB_Demo Start...");
if (Usb.Init() == -1)
{
  Serial.println("USB Host Init Error");
```

```
5   }
6   HidMouse.SetReportParser(0, (HIDReportParser *)&Prs);
```

ここでは我々のグループが制作に用いた M5Stack 用の USB モジュールが接続することができずにエラーが起きた時の例外処理用の文である。シリアルモニタに「M5USB_Demo Start…」という表示が出た後、「USB Host Init Error」が出た場合、USB モジュールが上手く接続できなかったことになる。

（※文責: 樫木海生）

loop

ここでは loop 内のプログラムについて解説する。

```
1   if (Usb.getUsbTaskState() == USB_STATE_RUNNING)
2   {
3     Mouse_Pointer(mou_px, mou_py);
4     mou_px = 0;
5     mou_py = 0;
6
7
8     if (mode == 1)
9     {
10      taiki();
11    }
12    else if (mode == 2)
13    {
14      yomikomi();
15    }
16    else if (mode == 3)
17    {
18      kakikomi();
19    }
20  }
```

USB モジュールが上手く接続され USB ハブに USB 接続があった場合、マウスポインタを扱っている関数である Mouse_Pointer() が呼び出される。そして、mode によって待機モードか読み込みモードか書き込みモードに割り当てられることになる。mode が 1 に指定されていると、if 文で mode が 1 のときの分岐に飛び、待機モードを呼び出すことになる。mode はプログラム実行時には 1 に初期化されているので、Copypen を起動すると待機モードから始まる。mode が 2 に指定されていると、if 文で mode が 2 のときの分岐に飛び、読み込みモードを呼び出すことになる。mode が 3 に指定されていると、if 文で mode が 3 のときの分岐に飛び、書き込みモードを呼び出すことになる。

（※文責: 樫木海生）

以上が本 Element である Copypen の制作で用いたプログラムの解説である。

（※文責: 樫木海生）

2.2 Element.02「BLWIND」

2.2.1 目標・目的

本 Element の目的は、実際の外の風をデータとして取得し、ブラインドのスラットを波のように動かすことで視覚的に心地よさを生み出すことである。また、生活における問題を楽しく、心地よく解決することに着目した。考案したアイデアとして「つい待ってしまうことが楽しくなるタイル」、「風で直観的にナビゲーションできるウエアラブルデバイス」、「視覚的にワクワクできるようなブラインド」などが挙げられた。

そこで、「ついついやってしまうこと図鑑」を制作している中での気づきの一つである「つい心地よく風を感じられる場所に行ってしまうこと」に着目し、アイデア出しでの「風で直観的にナビゲーションできるウエアラブルデバイス」、「視覚的にワクワクできるようなブラインド」をヒントに、「視覚的に外の風の心地良さを表現するブラインド」を制作した。

この Element の目標は、外界との Interaction の増幅である。人間は風の情報から、さまざまな Interaction を発生させている。この Element を見ることで、風を体で直接感じなくても風の心地よさや外界の開放感を感じることができる。さらに、風の情報を可視化し、外界を感じることで、面白い体験ができるようにすることが本 Element の目標である。

（※文責: 米原孝星）

前期の活動方針

まず、前期の活動方針として、制作するモノのコンセプトを固めることに方針を定めた。私たちは、フィールドワークの中で見つけた、体で感じる風の心地よさと、視覚で感じる動きの心地よさを組み合わせたプロダクトを制作することにした。プロジェクト全体を通して、一貫するようにコンセプトワークを行った。そこで、制作するプロダクトのコンセプトを「外の風を心地よく可視化させる」こととした。

そこで、私たちは、屋外と屋内の境界を繋ぐ建材として、縦羽根を持つブラインドに着目した。屋外の風に合わせて、ブラインドの羽根をなめらかに動かすことで、屋内からも風の心地よさを視覚的に感じることができると考えた。風の向きや強さを検出し、ブラインドの羽根の動きで心地よく可視化できるシステム「BLWIND」を制作することとした。

次に、コンセプトから具現化させるために、加工しやすいダンボールやスチレンボードを用いることで、素早くモックアップを制作することとした。また、Blender 内でも設計することで、コーディングを行う前に、スラットがパタパタ動くようなモーション設計を行うことができ、グループ内でイメージを共有することができた。実際に、モックアップを制作することで、プロダクト実現時の設計問題を洗い出すことを前期の活動方針とした。

（※文責: 米原孝星）

2.2.2 Element の制作手順・方法

プロトタイプ制作

私たちは、はじめに外の風を可視化するための方法について考えた。外の風を可視化させるための方法として、モーショングラフィックスをヒントに「パタパタと回転させる動き」に着目した。

そこで、屋外と屋内の境界線にあるブラインドを回転させることで外の風を可視化することをコンセプトとし、本 Element を制作した。ブラインドのスラット 1 枚 1 枚を Arduino を用いてサーボモータの回転角度、動作間隔を制御することで、サーボモータに接合されたスラットの一連の動きで波のような動きを創出し、外の風を可視化させることを実現できる。

スラットを回転させるのに必要なサーボモータを制御するプログラムを作成するのと同時並行で、ダンボールやスチレンボードを用いたモックアップを制作することで、すぐに実際の動きを確認できるように制作した。

つぎに、実際に想定しているサイズをグループメンバー間で共有することで、実物の大きさを幅 690mm・高さ 600mm の窓枠、幅 50mm・高さ 500mm のスラットを 10 枚制作することで決定した。実際の大きさが決定してから、3DCG アニメーション作成ソフト「Blender」を用いて、回転角度・動作間隔のシミュレーションを行い、グループメンバー間での動作イメージの共有を行った。これによりリアルでどのような動きが心地よいと感じるのかを即座に確認することが可能となった。

そして、プロトタイプは素早く加工することができ、ある程度の強度が担保されるスチレンボードを使用した。スチレンボードにサーボモータを埋め込むためにレーザカッターを使用して切り出し、窓枠とスチレンボードの接続部分は 3D プリンタを用いてクリップのような部品を制作することで、将来的にスラットを他の素材に変更しやすくなるように工夫を行った。(図 2.3)

また、サーボモータを制御するプログラムは、前述した Blender によるシミュレーションをモックアップに素早く反映するための工夫を行った。Arduino 内のパラメータを Blender 内と互換性を高めるようなプログラムを実現することで、Blender で想定した動きを同じようにリアルモックで反映することを可能にした。

リアルでモックアップを制作し、3D モデルでシミュレーションを行い、互換性を高めるプログラムを用意することで、風を心地よく可視化するための開発環境を整備することができた。

図 2.3　スチレンボードで製作した中間発表会時点でのプロトタイプ

（※文責: 米原孝星）

プロトタイプの改良

中間発表での質疑応答や、前期の活動を通じて、この Element の課題が 2 つ挙げられた。1 つ目は、風を取得するセンサを制作することである。私たちは、中間発表以前には、風を取得するセンサを実現することができていなかった。そのため、中間発表会ではサーボモータ内の変数を直接編集することで、動きを実現していた。

本 Element の展望としては、実際の風を取得し、心地よく反映させることである。そのため、風の情報を取得するようなセンサを実現することが求められる。前期の段階で、想定していた風の取得方法は、風車のようなものに加速度センサを取り付け、風力・風向を取得する方法であった。また、それ以外にも風鈴のように音で風を取得するセンサも考えられた。これらの方法で風を実際に取得できるセンサの実現を目指すこととした。

2 つ目は、どのような縦羽根のスラットが動きが心地よく感じる動きなのかを明確にすることである。前期の段階では、Blender によるシミュレーションでは風の動きを完全に表現できるようなパターンを制作できていなかった。そのため、多くのモーションパターンを試作し、どのようなスラットの動きから心地よさを感じるのか、評価実験を行うことで科学的な根拠を持った制作を心がけた。また、ブラインド自体の素材によって、与える印象にどのように関係するのかも検討していく必要がある。

よって、後期の活動方針として、前期で挙げられた問題点を解決しつつ、巧みなプロダクトの質感設計・風のモーション設計に尽力することとした。

（※文責: 米原孝星）

モーション設計

風のモーション設計

実際のプロダクトにプログラムを反映する前に、グループメンバー間でイメージを共有するために 3DCG アニメーション制作ソフト「Blender」を用いて、シミュレーションを行った。今回のプロダクトでは、外の風の風速・風向を心地よく可視化することを目指しているので、グループメンバー間で「心地よく感じるのか」により着目し、シュミレーションを行った。

まず、Blender 内で実際のプロダクトのモデリングを行った。実際の Element 本体のスケールが一致するように、緻密にモデリングを行った。

次に、モデリングしたスラットを Z 軸方向に回転させ、キーポイントを打つことで、アニメーションを制作した (最初の角度 0 度でキーポイントを打つ→そのスラットの到達目標角度、到達時間のフレーム数でキーポイントを打つ→最初の角度 0 度に戻るように同じフレーム数でキーポイントを打つ)。しかし、ここでのキーポイントの打点作業は全て手動で実施していた。そのため、1 枚のスラットに対して、3 つのキーポイントを打つ必要がある。よって、16 枚のスラットを手動で打つためには、合計で 48 個のキーポイントを打つ必要があった。これは、パターン数を増やし、あらゆる動きを制作する上では、非常に効率の悪い作業であった。

そこで、私たちは、コンピュータグラフィックスの講義内で紹介された、「スクリプトでアニメーションを制作」を実現することを目指した。これは、Blender 内での「スクリプト作成」という機能を使うことで、python コードでアニメーションの作成が可能となる。この機能を活用することで、手動で 48 個のキーフレームを計算しながら打っていた作業が圧倒的に短縮化される。1 パターンあたりの作業が手動から自動に変わることによって、作業効率が向上し、より多くのパター

ンのアニメーションを制作することが可能となった。

ここで、Blender 内で実現した python コードについて解説する。まず、モデリングしたオブジェクトを準備する。(ここで、アニメーションを制作する必要があるのは、スラットの部分のみなので、スラットのオブジェクトには、番号がわかるように cube.XX と設定しておく。※ XX には、01〜16 まで対応した数を割り振るものとする。)

また、サーボモータが制御できるパラメータとして、「スラット間の動作間隔」、「スラットの回転速度」、「スラット回転時の最大角度」、「スラット間の減衰量」の 4 種類とした。このパラメータ 4 種類を制御することで、以下のモーションパターン 4 つのアニメーションを制作した。

- パターン A 角度変化・動作間隔どちらもだんだん小さく
- パターン B 角度変化・動作間隔どちらも変化しない
- パターン C 角度変化だんだん小さく・動作間隔変化しない
- パターン D 動作間隔だんだん小さく・角度変化変化しない

パターン A では、python コード内の for 文で、interval = 31 - 2 * (i - 1) とし、その変数をスラットが回転するフレーム数に割り当てることで、1 枚目のスラットから離れれば離れるほど、2 フレームずつ減少させていくこととした。これによって、動作間隔をだんだん小さくすることを python コード内で実現している。また、MAX = float(90) と定義する。ここでの 90 度という角度の定義は、縦羽根のスラットの明暗がはっきりと認知できる最大角度である。この 90 度を for 文の中で、devived = float(90 / 16 * (i - 1)) を定義し、float$_r adians = MAX - devived$ をすることによって、90 度を 16 等分した値ずつ角度変化させることを実現している。これによって、*python* コード内で角度変化がだんだん小さくなることを実現している。

次にパターン B について解説する。python コード内の for 文で、interaval = 31 とした。この interval の変数をスラットが回転するフレーム数に割り当てる。これは、パターン A と違い、1 枚目〜16 枚目まで 31 フレームという同間隔で動作する。これによって、動作間隔を一定とすることを python コード内で実現している。また、MAX = float(90) と定義する。ここでは、パターン A での devived の変数を無効化とした。これらをすることによって、90 度という一定の回転角度を変化させないことを python コード内で実現している。

これらのプログラムコードから、部分的に無効化することによって、パターン A からパターン D の 4 パターンのモーションを python コードの中で実現できる。

印象調査

風の心地よさを伝えるのに適したスラットの動きを調査するため、Blender で作成した CG アニメーションを用いて予備的な印象調査を行った。評価方法として、順位評価と評価グリッド法を使用した。対象者は大学生 9 名とし、図 2.8 に示す 4 種類のアニメーションを提示した。これらは実機で表現可能なスラットの動作範囲を考慮して設計され、スラット毎の動作角度と、スラット間の動作間隔が異なる。各被験者は、4 種類の動きを心地いいと感じた順で序列をつけ、その理由をできるだけ詳細に回答することとした。

順位評価において最も多く 1 位評価されたのはパターン B(5 票) であり、次点はパターン C(4 票) であった。評価グリッド法を用いて理由を分析したところ、心地よい動きのキーワードとして、「自然」と「一定」が見られた。「パターン B では，等間隔・同角度で回転している『一定な感じ』

が心地よい」、「パターン C は，スラット毎の動作角度がだんだん減衰していくことで『自然な感じ』がして、自分が持っている風の印象とあっていた」といった意見が挙げられた。

よって、パターン C を元にして、サーボモータの制御コードに落とし込むことにした。制御可能なパラメータは、「スラット間の動作間隔」、「スラットの回転速度」、「スラット回転時の最大角度」とした。スラットが回転する最大角度は風速に応じて 10〜90 度の範囲に割り当てた上で、対象のスラットから離れる程一定ずつ減衰させた。風速が速いときは、スラットの最大角度・回転速度を大きくすることで、強い風の加速感を表現した。逆に風速が弱いときは、スラットの最大角度・回転速度を小さくすることで、おだやかな風を表現した。一方、スラット間の動作間隔は全て一定とした。このようにして、風の強弱を感じさせつつ、表現としての心地よさを保てるように工夫した（図 2.9 下）。

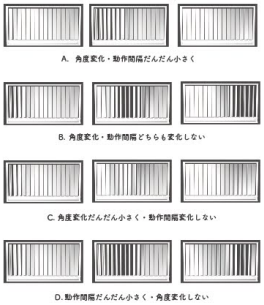

図 2.4　印象調査での 4 種類の動き

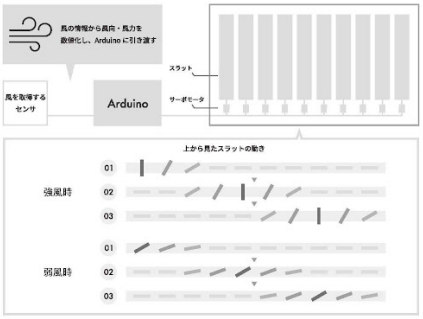

図 2.5　システム動作の流れ

（※文責: 米原孝星）

筐体・機構

プロダクトの外装の仕様変更

中間発表に制作していたプロトタイプは、サイズが幅 690mm 高さ 600mm であり、スラットが 10 枚並んだものであった。しかし、プロトタイプの動作では、横に流れる風を表現するにはスラットの枚数が不十分だと感じたため、スラットの枚数を 16 枚に増やした新しいモデルの制作をすることを決定した。16 枚という数は、Arduino Uno で使用可能なサーボモータ用のモータードライバ PCA9685 で制御できる最大の数であったためである。また、16 枚のスラットを並べた際にできる長方形の比率を、人が美しいと感じる黄金比である 1:1.618 とすることで、よりスラットの動作の心地よさを引き出す工夫をした。中間発表時に制作していたプロトタイプでは、サーボモータ

をプロダクトの上部に吊るす形で設置していたが、重心が上がることによりプロダクトが不安定になることや、サーボモータの配線の延長が必要になるという問題点があった。そのため、サーボモータと配線を格納する箱をプロダクトの下部に配置する仕様に変更することで、これらの問題を解決できるのではないかと考えた。

（※文責: 大塚創生）

プロダクトの素材

中間発表時に制作していたプロトタイプは、加工のしやすさからプロダクトの素材にスチレンボードを使用していた。しかし、強度が不十分でありフレームが歪んでしまったため、想定していたスラットの動作を実現することが難しく、見た目も悪いという問題点が挙げられた。そのため、新しくプロダクトに使用する素材の検討を行った。素材を決定する上で、フレームの強度や、加工のしやすさ、3D プリンタのパーツと組み合わせることのできる拡張性、見た目の美しさなどを考慮しながら検討した。新しい素材としてアルミフレームを骨組みとし、その上からアクリル板を貼り合わせる形で制作することを決定した。アルミフレームはミスミの 5 シリーズ 20mm 角、1 列溝アルミフレームを使用した。ミスミのアルミフレームを使用することで、課題であったフレームの剛性を解決できるほか、フレーム内に専用のナットを入れることができるので、3D プリンタで作成するパーツと組み合わせることができ拡張性に優れている。また、アルミフレームの上から白いマット素材のアクリル版を貼り合わせることで、プロダクトの見た目を美しくするほか、視覚的なノイズを減らしスラットの動きに注目させるという狙いがある。

（※文責: 大塚創生）

3D プリンタでのパーツの制作

「Fusion360」という 3DCAD ソフトを用いて、プロダクトの設計をする上で必要となるパーツを主に 4 つ制作した（図 2.4）。積層型の 3D プリンタを使用して制作を行なったが、パーツ自体小さいものも多かったため、寸法の精度を出すことに苦労した。どのパーツも精度を必要とするものが多く、パーツを設計する上で、0.1mm 単位での微調整が必要となる場合が多く、何度も実物と確認しながら制作した。

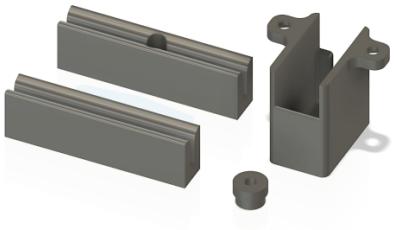

図 2.6　3D プリンタで制作したパーツ

（※文責: 大塚創生）

プロダクトの全体設計

「Fusion360」という 3DCAD ソフトを用いて、上記で記述した 3D プリンタで制作するパーツの設計と並行してプロダクトの全体設計を進めた。3D プリンタで制作したパーツのデータを読み込み、CAD 上でパーツを組み合わせながらプロダクト全体を設計することにより、寸法ミスや改善点の発見をしながら進めていくことができた。フレーム／スラット共に、背景との輝度差がわかりやすいように無光沢の白色素材を使用した。注目させたいスラットの動きに目線を集中させる目的のもと、視覚的なノイズを最小限にする設計を心掛けた。例えば、サーボモータや配線を収納し、ユーザから見えなくする工夫（図 2.5）や、スラットの上下と左右の隙間を統一すること (図 2.6)、3D プリンタで制作したパーツを全て研磨した上で白色に塗装することなどである。スラットには、サーボモータでしなやかに動かせるように、軽量で加工が容易な低発泡塩ビ板を使用した。

図 2.7　内部の収納スペース

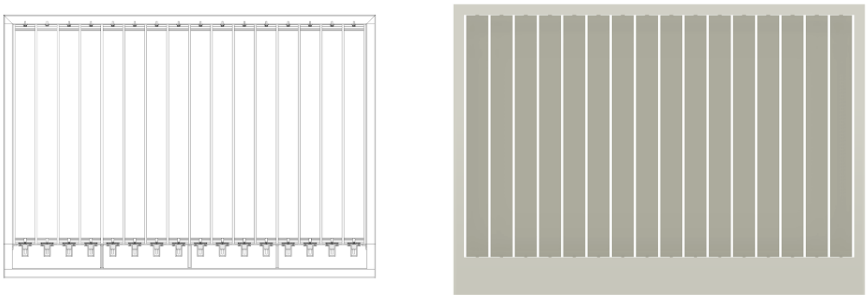

図 2.8　スラットの上下と左右の隙間を統一

（※文責: 大塚創生）

風センサ

プロペラ

風の向きと速度を計測するために、ロータリーエンコーダ（REL18-100BP）と、自作プロペラを組み合わせた風力センサを制作した。風力センサの制御には Arduino Nano を使用し、サーボモータの制御で使っている Arduino Uno と I2C 通信することで、データを送信している。「Fusion360」という 3DCAD ソフトを用いて、風力センサのモデリングを行い、3D プリンタで制作した（図 2.7)。センサの下部には Arduino Nano が収納できるようなケースを設計した。I2C 通信するためのジャンパワイヤを白くすることで、プロダクト全体の統一感を出した。また、プロ

ダクト本体のデザインと馴染むよう、角張ったデザインを心がけ制作した。

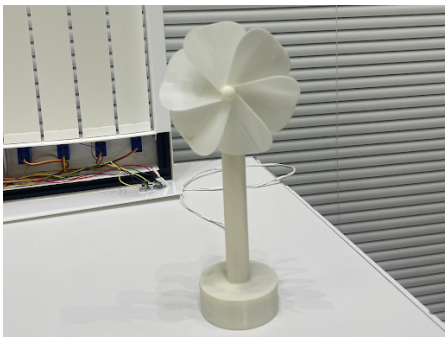

図 2.9　風力センサ

（※文責: 大塚創生）

風の取得方法

風の取得方法として、ロータリーエンコーダとプロペラを組み合わせた風センサを使用した。ロータリーエンコーダは小型で軽量の REL18-100BP を使用し、プロペラは 3D プリンタで作成した自作のものを使用した。プロペラに風を受けることで、軸と連結されたエンコーダが回転する。これにより風から取得したプロペラの回転角度と方向をサーボモータの制御に使用した。

サーボモータでスラットを制御する際に表現できる動きに制限があったため、今回ロータリエンコーダで取得する情報は加速度と回転角度、方向のみとし、制御には回転角度と方向を使用した。

前期のプロトタイプ時点では、風を取得するセンサを作成することができていなかったためサーボモータの角度を制御することでスラットの動きを表現していた。しかし、後期ではプロペラとロータリーエンコーダを用いた風センサを使用することで取得した風をもとにリアルタイムで心地よい動きを表現できるようになった。

（※文責: 井上芽依）

モータ

使用するモータ

BLWIND のスラット 16 枚の角度を制御するために、各スラットの下部にサーボモータを設置した。前期に作成したプロトタイプの段階で、すでに動きのシミュレーションを行っていたため、0 度から 90 度の範囲でスラットを動かすことや、表現したい動きが大まかに決まっていた。前期では、メンバーで想定していた動きはまだ再現できていなかったが、イメージは前期と変わらなかったためプロダクトを作り直す際も同様に SG90 を使用することとした。スラットは一枚一枚動くタイミングが異なり、角度や速度を細かく制御したいと考えていたため、0 度から 90 度の範囲で角度を指定することができ、速度の指定ができる SG90 が動きの面で最も適しているのではないかと考えた。また、制作の面でも BLWIND は初期のプロトタイプの段階から 10 個から 16 個のスラット数を想定していたため、軽量で電力消費が少なく、安価である SG90 が適していた。後期に作成したプロダクトではスラットの枚数を 10 枚から 16 枚に増やしたため、サーボモータも 16 個使用した。

（※文責: 井上芽依）

制御用のマイコン

制御用のマイコンとしてプロペラとロータリーエンコーダを組み合わせた風センサには Arduino Nano、サーボモータには Arduino Uno を使用した。

まず、風センサに Arduino Nano を用いたのはプロダクト制作の際に配線部分をできるだけ小さく、コンパクトにしたいと考えたからである。風センサには自作のプロペラとロータリーエンコーダを使用している。前期に作成していたプロトタイプの段階から、サーボモータの数が多く、BLWIND 本体のスラット制御部分に配線が集中することがわかっていたため、風センサはできるだけコンパクトに配置できるようにしたいと考えていた。そのため、Arduino の中では小さく、軽量で自作のプロペラの土台の中に収まるサイズのマイコン Arduino Nano を用いることとした。

サーボモータは前期のプロトタイプ時と同様に Arduino Uno を用いて制御した。

（※文責: 井上芽依）

モータドライバ

サーボモータを制御するにあたって、10 から 16 個のサーボモータを制御するには Arduino Uno だけでは PWM のピンが少ないことや電力が足りず、サーボモータの挙動が不安定になってしまうことから 16 個の PWM ピンを持ち、最大 16 個のサーボモータを安定して制御することができるモータドライバ PCA9685 も使用した。モータドライバを使用したことで、一台の Arduino から 16 個のサーボモータを安定して制御できるようになった。

（※文責: 井上芽依）

2.2.3　プログラムの解説

本 Element では、Arduino Uno と Arduino Nano を使用して開発を進めた。ここでは、それらのプログラムについて解説する。

風の取得

ここでは、風を取得し、サーボモータに取得した風の情報を送信するまでのコードについて解説する。ロータリーエンコーダは REL18-100BP を使用し、3D プリンタで自作したプロペラの軸部分に設置している。

まず、使用するライブラリは以下の通りである。

```
1 #include <Arduino.h>
2 #include <Wire.h>
```

風センサとサーボモータは I2C 通信を用いてデータの送受信を行っているため、Wire.h ライブラリを使用する。ただし、ここではサーボモータ側を Slave としている。次に、以下ではそれぞれの変数の初期値を定義している。

```
1 const int encoderPinA = 2;
2 const int encoderPinB = 3;
3 int previousEncoderStateA = LOW;
4 int encoderStateA;
```

```
int previousEncoderPosition = 0;
float encoderPosition = 0;
```

ここではロータリーエンコーダを Arduino Nano に接続する際のピン番号を指定している。使用しているロータリーエンコーダはインクリメンタル形であり、A 相と B 相の出力タイミングから回転方向を取得し、回転による A 相と B 相の状態の変化からパルスの数をカウントすることで回転量を計算し角度を取得する。ここでは、エンコーダの A 相のピンを 2、B 相のピンを 3 として設定したあと、エンコーダの状態を格納する変数を定義している。

次に、この初期値設定では、時間を管理して一定の間隔ごとに速度を計算するための初期値を指定している。

```
unsigned long previousMillis = 0;
const int interval = 500;
```

interval は速度を計算する間隔のことを指しており、0.5 秒ごとに速度が計算される。

以下は、この後の処理で用いる初期値を定義している。

```
int tmpAngle;
int angle;
float speed = 0;
int rotate;
```

int 型の変数である angle と rotate はそれぞれ I2C 通信で Master へ送信する変数である。angle は角度を定義し、rotate は回転方向を指定している。それぞれの変数はこれらのように定義した。次に setup() と loop() 内の処理について説明する。

まず、setup() は以下である。

```
void setup() {
  Serial.begin(9600);

  pinMode(encoderPinA, INPUT_PULLUP);
  pinMode(encoderPinB, INPUT_PULLUP);

  Wire.begin(8);
  Wire.begin();
  Wire.onRequest(DataRequest);
  Wire.onReceive(DataReceive);
}
```

ここでは、主に I2C 通信を行う際の関数を指定している。Master をサーボモータとして、Master からデータの要求があった時に呼び出す関数を DataRequest、データを受信した時に呼び出す関数を DataReceive とした。

次に、loop() は以下である。

```
void loop() {
  encoderStateA = digitalRead(encoderPinA);

  if (encoderStateA != previousEncoderStateA) { ...^^e2^^91^^a0
        if (digitalRead(encoderPinB) != encoderStateA) {
```

```
6         encoderPosition--;
7         rotate = 1;
8         } else {
9         encoderPosition++;
10        rotate = 0;
11        }
12
13        unsigned long currentMillis = millis();
14        if (currentMillis - previousMillis >= interval) {...^^e2^^91^^a1
15        speed = (encoderPosition - previousEncoderPosition) / (float)interval * 500.0;
16        Serial.print("speed: ");
17        Serial.println(speed);
18        previousMillis = currentMillis;
19        previousEncoderPosition = encoderPosition;
20
21        tmpAngle = map(abs(speed), 0, 800, 0, 90);
22        if(tmpAngle > 90) {
23        angle = 90;
24        }else if(tmpAngle < 0) {
25        angle = 0;
26        }else{
27        angle = tmpAngle;
28        }
29
30        Serial.print("angle: ");
31        Serial.println(angle);
32        Serial.print("encoderPosition: ");
33        Serial.println(encoderPosition);
34        Serial.print("rotate: ");
35        Serial.println(rotate);
36        Wire.beginTransmission(8);
37        Wire.write(angle);
38        Wire.write(rotate);
39
40        Wire.endTransmission();
41        }
42  }
43  previousEncoderStateA = encoderStateA;
44 }
```

loop() 内ではエンコーダの状態変化を監視し、一定の時間間隔ごとに速度や角度を計算して Master デバイスに送信する役割を果たしている。まず、最初の encoderStateA = digitalRead(encoderPinA); ではエンコーダの A 相の状態を読み取り、encoderStateA に格納する。次に、の if 文を用いて、エンコーダ A 相の状態が変化した場合の処理を指定する。エンコーダ A 相の状態が変化した時に、B 相の状態を確認し、エンコーダの回転方向を判定する。判定結果に応じ

て、encoderPotion を更新し、rotate も 0 か 1 を指定するよう定義する。

次にの if 文を用いて、一定の時間間隔ごとに速度を計算する条件を設定する。速度をもとに Master に送信する角度を決定する。ただし、サーボモータを 0 度から 90 度の範囲のみで動くように指定しているため、送信する角度が 90 度よりも大きくなった場合は角度を 90 度とし、0 度よりも小さくなった場合は 0 度にあらかじめ指定することで Master であるサーボモータにデータを送信した際に想定していない動きをすることがないようにした。

最後に、想定している動きを実装していく際に、コードの調整を行いやすくするために Master に送信するデータを中心にシリアルモニタに値を表示するよう指定した。プロダクト制作にあたって、風を心地よく可視化するために 3DCG シミュレーションソフト Blender で作成したシミュレーションを使用した。メンバーで決めた動きシミュレーションをもとにコードで再現することを目的としてコードを作成していたため、シリアルモニタに軸となる送信データを表示することはコードの修正に役立った。

次に、風センサでは最後となる、I2C 通信時に使用したデータ送受信の関数について説明する。コードは以下である。

```
1  void DataRequest() {
2    Send_data = angle;
3    Wire.write(Send_data);
4   // Wire.write(previousEncoderPosition);
5    Wire.write(rotate);
6    Serial.print("Send_data >> ");
7    Serial.println(Send_data);
8  }
9  void DataReceive() {
10   while (Wire.available()) {
11         Get_data = Wire.read();
12         Serial.println(" ");
13         Serial.print("Get_data << ");
14         Serial.println(Get_data);
15         Send_data = Get_data;
16   }
17 }
```

データを送信するときの関数を DataRequest()、データを受信するときの関数を DataReceive() とした。これらの処理もコードの修正を容易に行えるようにするために、送受信するデータをシリアルモニタに表示するようにした。

以上が風を取得し、サーボモータに取得した風の情報を送信するまでの風センサのコードである。これらの処理により、ロータリーエンコーダとプロペラを組み合わせた風センサを使用して風を取得し、I2C 通信を用いてデータを送信することでサーボモータを制御している。

（※文責: 井上芽依）

スラットの制御

ここでは、スラットの制御に使用したサーボモータのコードについて述べる。16 枚の各スラットの角度を制御するために、sg90 を 16 個使用した。また、16 個のサーボモータを一台の Arduino

から制御するために、モータドライバ（PCA9685）を使用し、制御用のマイコンとして Arduino Uno を使用した。中間発表時のプロトタイプでは、スラットの枚数が 10 枚であり、風センサとのデータの送受信も行っていなかった。後期ではプロダクト設計にあたってスラットの枚数が 10 枚から 16 枚に変化し、風センサとの I2C 通信を行うことでデータの送受信をできるようにしなければいけなかったため、設計に合わせてコードの修正を行った。

ここからはコードについて述べる。

まず、使用するライブラリは以下の通りである。

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
```

後期でのコード修正の初期段階ではライブラリとしてモータドライバ（PCA9685）の公式ライブラリを使用していた。しかし、制作に移り、サーボモータを設置したあと、サーボモータの初期値がパルス幅によって異なるため角度を 0 度に指定していたとしてもスラットごとに角度が異なってみえることがわかった。つまり、16 枚のスラットの角度を全て 0 度に指定していたとしても、全てのスラットが一直線に並んでいるように見えなかった。BLWIND のスラットはプロペラから風を取得したときのみ動くようにしたかったので、0 度のときの角度がサーボモータによって異なる部分を改善する必要があった。このように、サーボモータごとに 0 度の位置が異なる場合、サーボモータの位置を角度ではなくパルス幅で指定する必要がある。しかし、モータドライバのライブラリでは、パルス幅を使用して角度を指定することができない。そのため、当初のモータドライバを使用したライブラリから、サーボモータのパルス幅を指定することができる PWM サーボドライバー Adafruit$_{P}WMServoDriver.h$ ライブラリに変更した。このライブラリは、$AdafruitPWM$ サーボドライバーボードを制御するための $Arduino$ 用ライブラリで、$Arduino$ マイクロコントローラーが $I2C$ 通信を介して PWM サーボドライバーボードを制御し、複数のサーボモータを同時に駆動することが可能になる。ライブラリを変更したことで、角度で指定していた初期値をパルス幅で指定できるようになったため、$BLWIND$ を正面から見た時にスラットが一直線に見えるよう定義できるようになった。

また、風を取得する際と同様に、I2C 通信でデータの送受信を行うためのライブラリも使用した。I2C 通信では、サーボモータを Master としてデータの送受信を行った。

次に、以下ではそれぞれの変数の初期値を定義している。

```
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x7f);

int pin[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };
int servoMin[] = { 120, 130, 130, 86, 118, 118, 118, 130, 118, 105, 108,
120, 100,120, 130, 125 };
int servoMax[] = { 600, 600, 600, 600, 600, 600, 600, 600, 600, 600, 600,
600, 600, 600, 600, 600 };

int qua = 16;
int dly = 250;
int minAngle = 0;
long maxAngle[] = { 90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90,
90, 90, 90, 90 };
bool activeFlags[] = { true, true, true, true, true, true, true, true,
true, true, true, true, true, true, true, true };
```

```
int pos[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
long increment[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
int updateInterval = 10;
unsigned long lastUpdate = 0;
unsigned long lastFlagUpdate[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0 };
unsigned long lastWireUpdate = 0;
int data1 = 0;
int data2 = 0;
int countFlag[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
int countServo = 1;
unsigned long lastCSUpdate = 0;
int tmpdirec = 1;
```

制御パラメータとして、各モータの動作角度（最小、最大)、動作間隔、動作速度等を配列として用意した。特に、こだわった点として角度を取得していない時にスラットを一直線に配置するために全てのサーボモータの初期角度をパルス幅で定義した点である。初期値として全てのサーボモータの最大パルス幅と最小パルス幅を統一して定義することも可能だったが、16 個のサーボモータはそれぞれパルスの幅が異なっていて位置が異なるため、最大パルス幅と最小パルス幅をそれぞれ servoMax[]、servoMin[] として配列で定義した。最大パルス幅は配列の中でも同じ値で統一したが、初期値となる最小パルス幅は実際にコードを動かしながら正面から見た時に該当する要素番号のスラットが真っ直ぐになるように値を調整した。そのため、最小パルス幅はサーボモータによって異なり、86 から 130 の範囲で定義した。これにより、角度を取得していない時、BLWIND を正面から見るとスラットが全て真っ直ぐになるようになった。

さらに各スラットの同期を取りつつなめらかに制御するために角度の増分を指定する変数 increment、各サーボモータの状態を示す変数 activeFlags なども配列で定義した。

BLWIND のスラットは同時に動くのではなく、一枚一枚角度や動作開始時間をずらして波のような動きを表現する必要があったため、変数もサーボモータごとに調整がしやすいように工夫して定義した。

次に、setup() は以下である。

```
void setup() {
  Wire.begin();
  Serial.begin(9600);

  pwm.begin();
  pwm.setPWMFreq(50);

  int initialPosition = 0;
  for (int i = 0; i < 16; i++) {
        pos[i] = initialPosition;
        pwm.setPWM(pin[i], servoMin[i], map(pos[i], 0, 180, servoMin[i],
 servoMax[i]));
  }
}
```

ここでは、I2C 通信とシリアル通信の初期設定のあとに、PWM サーボドライバーの初期設定を行った。ここで、PWM の周波数は初期値として 50Hz とした。さらに、プロペラが回転しておらず、風を取得していないときの初期位置を０と定義し、１６個のサーボモータの初期位置が全て０の位置になるように map 関数を使用してサーボモータの初期位置を PWM 信号の範囲（最大パルス幅と最小パルス幅の範囲）に変換して指定した。この処理により、風を取得していないときの角度を指定し、スラットが正面から見て一直線になるようにした。

次に、スラットを制御する上での最も重要な処理となる loop() 内の処理について説明する。loop() は以下である。ただし、loop() 内で行なっている処理は多いため、いくつかに分けて説明する。

```
1  void loop() {
2    if ((millis() - lastWireUpdate) > 250) {
3          Wire.requestFrom(8, 2);
4          Serial.print("data1: ");
5          Serial.println(data1);
6          Serial.print("data2: ");
7          Serial.println(data2);
8
9          while (Wire.available()) {
10         data1 = Wire.read();
11         data2 = Wire.read();
12         lastWireUpdate = millis();
13         }
14   }
```

まずここまでの部分では、主に I2C 通信からのデータの受信を行っている。風センサから送信される二つの angle と rotate は角度と回転方向と定義した。これらの二つのデータを受け取り、それぞれ data1、data2 としてシリアルモニタに表示する。シリアルモニタに表示することで風の取得時と同様にコードの修正を行いやすくした。

次に以下のコードである。

```
1  if ((millis() - lastCSUpdate) > dly * countServo && countServo < 16) {
2          countServo++;
3    }
```

この部分では、一定の条件のもとでサーボモータの角度を更新する処理を行なっている。条件を前回のサーボモータの更新から一定時間が経過し、dly*countServo ミリ秒以上経過している場合かつ countServo が 16 よりも小さいときとした。条件を満たしたとき、countServo の値は 1 増加する。これにより、16 個のサーボモータ全体に順番に角度の更新処理が行われるようにした。

次に以下のコードについて説明する。

```
1  if ((millis() - lastUpdate) > updateInterval) {
2          lastUpdate = millis();
3
4          for (int i = 0; i < countServo; i++) {
5          if (tmpdirec == 0) {
6          if (activeFlags[15 - i] == true) {
```

```
7         pos[15 - i] += increment[15 - i];
8         pwm.setPWM(pin[15 - i], pos[15 - i], map(pos[15 - i], 0, 180,
9        servoMin[15 - i], servoMax[15 - i]));
10        }
11
12        if (pos[15 - i] >= maxAngle[i] || pos[15 - i] <= minAngle) {
13        increment[15 - i] = -increment[15 - i];
14        pos[15 - i] += increment[15 - i];
15        countFlag[15 - i]++;
16
17        if (countFlag[15 - i] == 2) {
18                activeFlags[15 - i] = false;
19                lastFlagUpdate[15 - i] = millis();
20                countFlag[15 - i] = 0;
21
22                if ((15 - i) == 0) {
23                while (millis() - lastFlagUpdate[15] < 2000) {
24                }
25                long tmp = millis();
26                long tmpangle = data1;
27                dly = map(tmpangle, 90, 10, 100, 200);
28                updateInterval = map(tmpangle, 90, 10, 5, 15);
29                for (int j = 0; j < 16; j++) {
30                activeFlags[j] = true;
31                lastFlagUpdate[j] = tmp;
32                maxAngle[j] = tmpangle - ((tmpangle / 16) * (j + 1)) + 10;
33                }
34                countServo = 1;
35                lastCSUpdate = tmp;
36                tmpdirec = data2;
37                }
38        }
39        }
40        }
41        if (tmpdirec == 1) {
42        if (activeFlags[i] == true) {
43        pos[i] += increment[i];
44        pwm.setPWM(pin[i], pos[i], map(pos[i], 0, 180, servoMin[i],
45        servoMax[i]));
46        }
47
48        if (pos[i] >= maxAngle[i] || pos[i] <= minAngle) {
49        increment[i] = -increment[i];
50        pos[i] += increment[i];
51        countFlag[i]++;
```

```
52
53          if (countFlag[i] == 2) {
54                  activeFlags[i] = false;
55                  lastFlagUpdate[i] = millis();
56                  countFlag[i] = 0;
57
58                  if (i == 15) {
59                  while (millis() - lastFlagUpdate[15] < 2000) {
60                  }
61                  long tmp = millis();
62                  long tmpangle = data1;
63                  dly = map(tmpangle, 90, 10, 100, 200);
64                  updateInterval = map(tmpangle, 90, 10, 5, 15);
65                  for (int j = 0; j < 16; j++) {
66                  activeFlags[j] = true;
67                  lastFlagUpdate[j];
68                  }
69          }
70          }
71          }
72          }
73    }
74  }
```

この部分では、一定の間隔で各サーボモータの角度を更新する処理を行なっている。サーボモータの動きは tmpdirec によって制御されていて、tmpdirec が 0 の場合と 1 の場合とで処理が分かれている。これはプロペラの回転方向に応じてスラットが動く向きを変更するためである。

また、delay を使用すると 16 個のサーボモータそれぞれを異なるタイミングで動かすことができないため、ここでは millis() を使用して制御した。

さらに、モータの角度を目標値まで一度に動かさず、順番に少量ずつ変化させるようにするために幾つかの for 文を用いてサーボモータを制御した。回転方向に関わらず、取得した角度を 10〜90 度の範囲に割り当てた上で、対象のスラットから離れる程一定ずつ減衰させた。角度が大きい時を風速が速いときとみなし、速い時にスラットの最大角度・回転速度を大きくすることで、強い風の加速感を表現した。逆に風速が弱いときは、スラットの最大角度・回転速度を小さくすることで、おだやかな風を表現した。一方、スラット間の動作間隔は全て一定とした。このようにして、風の強弱を感じさせつつ、表現としての心地よさを保てるようなコードとなるようにした。

（※文責: 井上芽依）

2.3 Element.03「雨守り」

2.3.1 目標・目的

本 Element の目的は、日常生活において身近なものである傘を機械的な要素を用いることで風を受けて動作させ、また持ち手を握る強さで動作させるという 2 つのインタラクティブな要素を与

えることである。これによって、「これまで人が行っていた物事を捉え直し、他の機構で代替することで、その行動を楽にする」という新しい体験を生み出すことである。新しい体験を生む Element を制作するにあたって、まずどのような視点で考えるべきかを定める必要があると考え、様々な視点でアイデア出しを行った。グループ内での案出や、プロジェクト全体で制作した「ついついやってしまうこと図鑑」などを通して、本グループは音に関する事象に注目した。加えて、「ついついやってしまうこと図鑑」で出された雨の方向に反応する傘という案に着目した。そこから、雨の方向を検知するためには風向と風力の検知が必要ということが分かった。風で雨の向きが変わった際に手の動作のみだと防くことや雨の音のみを検知すること、光や振動でも適切な風向と風力を検知することが困難という点に気づいた。そこで雨が降ってくる方向となる風向きは角度の検知に特化したロータリーセンサで自動検知することと、防ぎ具合に直結する傘の傾きは人間の傘の握り具合という人的要因で調節できるようにすることで、雨の中で傘をさすことを楽にすることができる 2 つのインタラクティブな要素を傘に与えることとなった。

本 Element の目標は「全方向の雨から利用者を守る傘」というコンセプトをもとに、風にあおられる雨をインタラクティブに防ぐことで風向きを見ながら対応する労力を減らし、風の強い日の雨を外出の障害としないようにすることである

（※文責: 森重龍）

2.3.2　制作過程

前期の方針

本グループでは雨と傘に着目し、風向きを計算して雨を完全に防ぐ傘である「雨守り」の制作を行った。この Element の制作を始める際に、どのように傘を傾けるか、風向きはどのように検知するかをメンバー内で話し合った。まず、傘を傾ける際に必要なセンサとアクチュエータには何があるのかをまとめ、設計を検討した。結果、基本的な機構としては、サーボモータを 2 つ用いて縦に重ね、上側を前後方向に倒れるサーボモータ、下側を地面と平行に回転するサーボモータとすることで、全方向へ傘を傾けることができるといった物を考案した。

しかし、2 か月間という中間発表会までの短い期間内に目に見える形で実現する事は容易ではないと感じた。そのため、入手が容易なものや既にあるものでプロトタイプを制作して方向性を定めることが最重要課題であると考えた。プロトタイプ上でサーボモータを用いる以外に実現すべき基本的な機構として、風向きを検知するセンサを考案する必要があった。風向きセンサの案として、光を利用して物体の有無や位置を検出する小型の光センサであるフォトインタラプタを利用した風量によってセンサに入る光の量が変化するものや、風の音を利用してマイクから入る風音の大きさで感知する案などが出されたが、どれも制作が難しかったり、精度が低かったり、大型であったりといった問題があった。

（※文責: 森重龍）

中間発表用プロトタイプ制作

検討を基に M5StickC を利用した傾きセンサや、以前プロジェクト内で使われていたサーボモータなどのすぐに入手できるものを活用し、Element としての動きが想像しやすい、限定的な動きを行うプロトタイプを制作した。このプロトタイプでは M5StickC の傾きセンサを利用して内部

のセンサから傾き具合を検知、そのデータ Arduino へ送信することで Arduino 上で計算を行い、サーボモータで動作させるという雨守りの基本的な動作を実現した。

（※文責: 森重龍）

後期の方針

中間発表後、プロトタイプをもとに発表で尋ねられた質問や Google フォームで集められたフィードバックを用いて、現状の問題点と実寸大サイズの制作における設計を考案した。

はじめに問題点として、M5StickC を利用した傾きセンサでは正確性に欠けるという点が挙げられた。実際を想定した際に、歩いた際に受ける振動や歩くことで発生する向かい風など外界の影響を多く受けてしまうため、風力と風向を正確にデータ化することが困難であることが分かった。そのため、再度風向センサの考案をする必要があることが分かった。次に、実寸大サイズにおける重量問題である。中間発表後の段階で、通常の傘の重量に加えてモータやセンサ類の重量も加わる想定となっていたため、重量が非常に重くなってしまう状態であった。そのため、後期の制作に入る際には使用する素材を吟味して制作する必要があることが分かった。

これらの問題点を踏まえて、後期での制作ではサーボモータの配置や手持ち位置などを実際に作ってみて試行錯誤していくという形になった。

後期以降の制作では、主に動作班と外装班の 2 つに分かれて制作を行った。これは得意な分野ごとに作業を分担することで、制作の効率化を図った。

（※文責: 森重龍）

モータ

動作班では、大きく分けて「雨守りのサーボモータ動作の確立」と「風向センサの設計・実装」を行った。はじめに、中間発表後に発注していたサーボモータを用いた動作の確立を行った。中間発表で使用したサーボモータでは、実寸大サイズの傘を動かすために十分なパワーを持ち合わせていなかったため、安定して動作を行えるものに交換することとなった。新しいサーボモータではパワーの向上に加えて正確な角度指定ができるため、傘の動作をより安定したものにできるようになったが、角度指定できる範囲が 0 度から 180 度までと制限されてしまったため、プログラムに修正が行われた。具体的には、風向に合わせて水平方向動作可動域 180 度と傾き動作の傾く方向の兼ね合いを可能にするプログラムを作成した。これによって、2 つのサーボモータの 180 度可動域内での全方向対応を可能にすることができた。

このプログラムができたことによって、想定していた雨守りの動きが実現可能な状態にすることができたため、実際に傘を取り付けて動作のフィードバックを行った。そこで、サーボモータの動作が予想以上に持ち手に対して反動が大きいことが分かったため、サーボモータの動作速度を減少させるようプログラム調整することで反動の軽減を図った。その結果として動作の反動を軽減することができ、利用者にストレスを与える要因を減らすことができた。これをもって、基本的な雨守りの動作を実現するサーボモータの動作は確立することができた。

（※文責: 森重龍）

センサ

次に動作班では、センサについて中間発表後に実寸大サイズの風向センサに使用するセンサの再選定を行った。様々なセンサがある中で、振動センサや光センサなどが挙げられたが動作に合うセンサはこの時点では見つけることができず、風向センサ自体の考案も定着状態だった。

考案を続けることで、ボールを使用した風向センサという案が考案された。これは、薄いすり鉢状の土台と小さく軽いボール、フォトリフレクタを使用したものとなっており、土台の中で風を受けて転がるボールをフォトリフレクタで検知することで風向を見出すというものだったが、これでは風向を検知することはできるが風力が検知できないという点に加えてボールで風向センサを作るのは難しくあまりに複雑なものであったため、より単純にとる方法を考えたほうが良いという結果となった。

この結果を受けて、風向センサのみで風向と風力を取ることは難しいと判断したため風光センサを風向のみ取るものとして扱い、追加で利用者が握る持ち手部分に圧力式のボタンを設置することで、外界とのインタラクティブ要素に加えて人的インタラクティブ要素を追加する方針となった。

また、風向センサはロータリーセンサという角度センサと風向計の要素を使用した。風向計の風向きに向いた際に回った角度をロータリーセンサで検知することで、風向を検知するという仕組みとなっている。ロータリーセンサは、他のセンサに比べて角度を取ることに特化しているため、今回の風向センサとしての役割に最適だと考えた。

風向計では 3DCAD ソフトである Fusion360 を使用して 3D モデルを作製し、3D プリンターを使用して作成した。この 2 つを組み合わせることでシンプルな構造にすることが可能となり、機能としてわかりやすいものにすることができた。

これらを組み合わせることで雨守りの動作が確立した。機能としては風向センサで風向を検知してマイコンの Arduino を経由し水平方向のサーボモータを動作させ、圧力ボタンで握り具合を検知して Arduino を経由し傾き動作のサーボモータを動作させるという仕組みになった。検知する対象を分けたことによって単純化することでき、人的要素を加えることで Element としての機能性を向上することができた。これをもって雨守りの動作全体が完成となった。

（※文責: 森重龍）

機構と外装

外装班では、はじめに大まかな設計と使用する素材の考案から入った。

まず設計段階で傘を一から作るのは時間がかかると判断したため、市販の傘を加工して目的の動作に合ったパーツ作製と素材からの軽量化を図った。傘に関してはホームセンターで販売されている折り畳み傘を購入した。通常の傘と折り畳み傘の比較に関して、通常の傘の重量だと全体重量的に過重量になってしまうため、小さくなってはしまうが重量や機能性を重視して折り畳み傘を選択した。傘は傾けることができるように上部の広がる部分から中部までを使用し、下部はのこぎりを使用して切断することで長さを調整した。

また、傘のパイプは初期状態から歪んでいたため加工に不向きであったので、傘の中部のパイプを少し短くし、外径 5mm のアルミ棒をパイプに接続することで長さを伸ばし、後のアルミパイプとの接続加工をしやすくした。次に傘の中部から柄の部分は、金属系でも軽い分類になるアルミパイプを使用した。アルミパイプは加工がしやすく強度も十分なものであったため今回の素材に適していると判断し使用した。

これらの素材を集めた後、2 方向の動作の前に傘の傾きの 1 方向のみの動作を確認するため、傾きのみのプロトタイプの作製を開始した。

始めに、アルミパイプを使用する分だけのこぎりで切り取り、サーボモータを基準に上部パイプと下部パイプの作製を行いつつ、サーボモータに接続できるようにネジ穴開けやパイプ加工を行った。接続部分やサーボモータの固定には 8mm 厚のアクリル板を使用した。アクリル板は、他の加工製品に比べて強度もある上に加工しやすいため今回の制作で多用した素材の 1 つである。また実際に傘を使用する為に、上部パイプ部分に加工を加えた。使用したアルミパイプが内径約 14mm であるのに比べて傘のパイプ部分は外径約 7mm であったため、パイプに通すには隙間が空きすぎていた。

ここでスペースを埋めるパーツを Medium Density Fiberboard（以下 MDF）で作製した。MDF を外径 16mm で円形加工しつつ中央に 5mm の穴をあけて加工し、できるだけ多く作製しつなげることで、スペースを埋める役割と接続部分の固定の役割を同時に持たせることができ、それらを使用することで 1 方向のプロトタイプの作製が完了した。

このプロトタイプを受けて作製後のフィードバックを行った。サーボモータとアルミパイプとの接続部分に隙間があるため動作にがたつきが生まれてしまっていることに加えて、サーボモータの動作部分付近の加工に問題があることが分かった。また現状の傘の上部が傘を閉じることができなかったため、サーボモータと傘との連結に使用するパイプの再選定が必要となっていた。他にも問題点が多く出たが、実際に傘が傾けることができることを確認できたので、解決案を模索しつつ本制作に入った。

本制作に入る前にプロトタイプの際に発生した問題に対し、複数の解決案を考案した。

はじめに、動作面で危惧されていた傘本体に関しては基本的にプロトタイプのものを流用し、動作範囲の調性のため長さの調節を行った。これによって守る範囲は少し狭まってしまったが、使用時の周りへの影響と動作を鑑みた結果として妥当な対応だった。

次に動作部分では、本来の目標である全方位の雨を防ぐためには動作部分であるサーボモータを 1 つから 2 つに増加させる必要があったため、本体設計の見直しを行った。案としてサーボモータ 2 つを 1 か所にまとめるか、離れた位置に設置することで重量の分散を試みるかの 2 案が考案された。作製にかかる時間や難易度の兼ね合いを考慮した結果、試しにサーボモータを 1 か所にまとめた案で制作を行い、その際に致命的な問題点があった場合に分散させる案を起用するという形となった。

またサーボモータを 1 か所にまとめる際に使用する素材として、プロトタイプ作製で使用したアクリル板を使用して連結パーツを作製するという方針になった。ここのパーツ設計にて動作の安定性を向上させるように作製するように計画を立てた。

最後に傘を閉じることができない問題に対して、サーボモータと傘本体との連結部分に使用していたアルミパイプを細いものに変更することで傘が閉じられることが判明したため、外径 16mm のアルミパイプから外径 7mm の銅パイプに変更することで問題解決を図った。しかし銅パイプに変更することで、傘本体に装着しているアルミ棒を銅パイプに通した際の内側の隙間が大幅に小さくなったため、プロトタイプで使用していた MDF では強度が確保することができず加工が難しくなってしまったため使用できなくなった。隙間埋めの代案については、傘側のアルミ棒にテープを巻き付けることとなった。

これらの解決案を一度設計段階で取り入れられるように設計図の再構築を行った。設計では、雨守りの外装イメージを作製し、使用する各パーツの採寸を記録した。そこから各部分で作製しなければいけないパーツをまとめて設計を行った。また 2 つのサーボモータの連結パーツ設計を行う

際に、グラフィックデザインツールである Adobe Illustrator を使用した。また、ここの設計段階において Arduino やモータドライバ等の収納については考慮していなかった。理由としては、グループにてまず Element としての動作の確立を最優先にするべきだと考えたからである。

設計がある程度完成した段階で、本格的に外装制作に取り掛かった。まず初めに行ったのはアルミパイプの加工である。この段階では風向センサの取り付け位置が決まってなかったため、加工の際には設計した長さより少し長めに行うことでイレギュラーに対処できるようにした。

次に手持ち部分のアルミパイプとサーボモータの連結部分の作製である。プロトタイプの問題点で挙げられていた動作のがたつきを解消する為に、プロトタイプで作製したものより複雑な設計を行った。具体的には持ち手部分に繋がるアルミパイプとサーボモータの連結部分をアクリル板で作製し、留め具として使用した M3 型ネジの入れ方を工夫することで、過度な動きに対してもがたつきを生まない連結パーツを作製することができた。

ここでパーツ作製にあたって重要となる工夫点として 2 つ挙げられる。まず 1 つ目に、アクリルパーツ同士のつなぎ方である。単にパーツ同士を重ねてネジで止めるだけ終わってしまうと、ネジ穴付近のパーツの境目の強度に問題が生まれてしまう。その点を改善するため、パーツの形状に工夫を加え隣り合うパーツ同士の穴と穴に合う凸部分を作りパーツ同士の干渉力を大きくした。またアクリル板を加工する際には学内工房のレーザーカッターを使用した。

パーツの穴や凸工夫を受けて、2 つ目に使用する留め具の使用方法を変更した。設計段階ではM3 ネジを多用する予定だったが、留め具部分のアクリルパーツを特殊な形状にすることで 1 か所 留めるために使用するのが M3 ネジ 1 本と同規格ナット 1 個で可能になった。

次の作業では、上部に設置する 2 つのサーボモータ間の接続部分の作製を行った。ここではじめに行ったのは、レーザーカッターで加工したアクリル板に傘の動きに耐えられるようなサーボモータの接続を行うということだった。作製していた設計図ではアクリル板とサーボモータをうまく繋げることができないことがわかり、有識者に助言をいただきながら再考案を行った。その結果として、サーボモータ本体に備わっているネジ穴を有効利用できるようアクリル板加工を行うことで問題を解決することができた。

また 2 つのサーボモータはそれぞれ目的とする動きが違う。下のサーボモータは水平方向動作のためにサーボモータを縦にして固定するため縦型のパーツを作製し、上のサーボモータは傾き動作のため横に倒した状態で使用するため横向きに作製した。この問題を解決することで、2 つのサーボモータ間の接続部分の作製に入ることができた。

接続部分の作製にあたって、サーボモータ購入の際に付属されていたジョイントパーツを使用した。ジョイントパーツを使用することで、ジョイントパーツを 1 から作るより加工に自由度を持たせることができた。使用したジョイントパーツは本体の駆動部分と繋げるパーツで、パーツ本体にネジ穴が開いているため基本的に繋げたい対象に同じ規格のネジ穴をあけることで接続は容易にできるものとなっていた。そのため、前段階で傾き用サーボモータに取り付けたアクリルパーツに同じ規格のネジ穴をあけることで繋げることができた。ここで繋ぎ目部分の強度について危惧されていたが、ジョイントパーツとアクリルの強度やネジの具合を鑑みて問題ないと判断した。その後サーボモータと傘本体との連結部分に使用していたアルミパイプを銅パイプに変更し、傘の開閉が可能な状態にした。これによって傘としての機能を残しつつ、Element としての要素を加えるという形をとることができた。

次に水平方向動作用サーボモータを動作させるのに使用した風向センサの設置を行った。風向センサは傘の下部に設置する設計であったため、アルミパイプと繋ぎ合わせるために折り畳み傘の手持ち部分を加工して使用した。

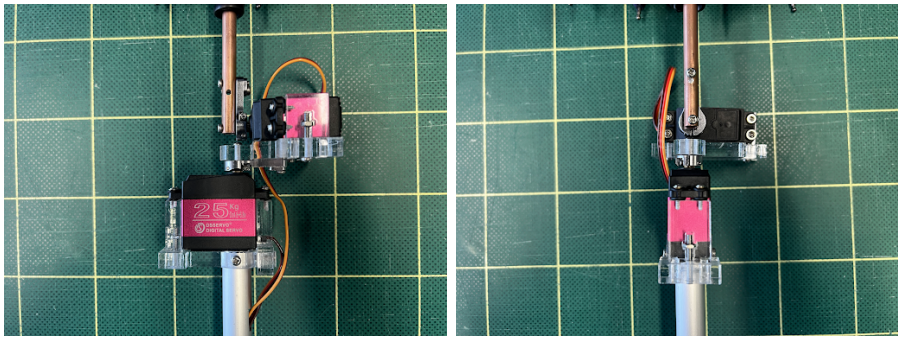

図 2.10　本体サーボモータ連結部分

折り畳み傘のパーツを使用した理由は大きく 2 つあった。1 つ目に、傘としてのイメージを残すためである。今回の雨守りは日常生活での使用を想定していたため、傘のパーツを使用することで視覚的にも傘の Element であることをわかりやすくすることができると考えた。2 つ目に風向センサの拡張子としての扱いやすさである。このパーツは配線の経路確保や加工がしやすいなどの利点が多く存在していたため使用することとした。

取り付ける際に風向センサの向いている向きとデータで表されている向きが一致するように固定する位置を調整し、取り付け後は、加工パーツとアルミパイプの繋ぎ目に M3 ネジを通すことで固定をした。

続いて傾き動作用サーボモータを動作させるのに使用した圧力ボタンの設置を行った。設置開始時ではアルミパイプに直接貼り付ける予定であったが、直接貼り付けてしまうと圧力がうまくとることができず持ちにくいことが判明したため、追加で手持ち部分のパーツの考案を行った。その結果、スイッチを装着できるような手持ちパーツを 3D プリンターで作製するという方針となった。

ここで 3DCAD ソフトである Fusion360 を使用して手持ちパーツの 3D モデルを作製してプリントした。また圧力操作がしやすくなるように、シリコン製の柔らかいボタンにすることでより押し具合が調節しやすいものとなった。ここまでの制作で雨守りとしてのおおまかな外装が完成とな

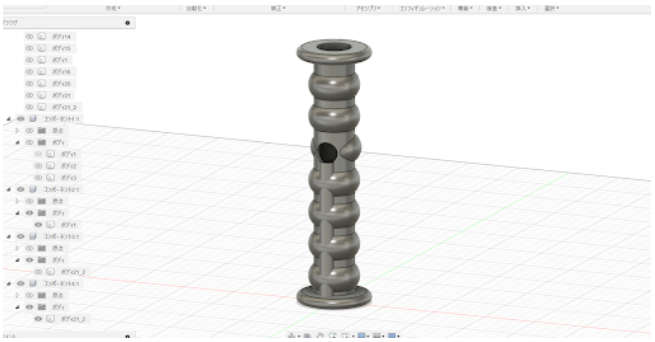

図 2.11　Fusion360 上での手持ちパーツの作製

り、予定していた動作が可能な状態となっていた。しかし、動作に使用しているマイコンやブレッドボード等は収納しておらず、電力供給もコンセントから DC ケーブルを通して行っていた。グループで話し合った結果、やはり傘というものとして電力等も独立させた方が良いということとなり、マイコンやブレッドボード、電力供給用バッテリーを収納するケースを作製することとなった。

ケースの素材として乳白色の厚さ 2mm アクリル板を使用し、設置場所はグリップと風向センサの間に設置した。形状は縦長のボックス型となっており縦横 75mm 高さ 135mm となった。各面の端に凹凸をつけた組み込み式の形状を使用した。またマイコンの位置が固定になったため、配線

もケースに集中するよう配線経路を新しく作った。特に上部に設置されてあるサーボモータの配線はアルミパイプの側面に穴をあけることで外に出さずに配線を下まで持ってくることができた。

この時点で全体としての外装制作は終了し、ここからは細かい修正を行った。まずマイコンなどを収納しているケースの中の整理を行った。収納内容としてマイコン、ブレッドボード、サーボモータ用ドライバ、バッテリーとなっており、これらをケース内に収納するとなると配置を工夫しなければ入らなかった。ブレッドボードの配線を変え、各要素の固定場所を考慮することで上手く収納することができた。

その後、風光センサと圧力ボタンの感度確認や動作班との全体通しての動作確認をもって雨守りの外装が完成となった。

（※文責: 森重龍）

2.3.3　ソフトウェア解説

本 Element に用いたプログラムについての解説を行う。なお、プログラムは末尾に付録として記載を行っている。

要求された要件

プログラム本文の解説の前に、プログラムを組むにあたって求められた要件について解説する。

求められた要件は大まかに

- モータの安定した動作の確立
- モータの回転の反動の軽減
- ２つのサーボモータを同時に独立して動かす

という 3 つの点に分けることができる。

まず、一つ目の要件について、本 Element は比較的力の強いサーボモータを 2 個用いる必要があった。しかし、Arduino 単体では低出力のサーボモータ 1 個をかろうじて動かすことのできる程度の電力しか供給できず、モータを安定して動かすことができないといった問題が発生した。サーボモータを動かす方法として一般的なものは、方形派の周波数を固定したうえで、電圧のパルスが高い時間の割合を変えることで電力を制御する「PWM 制御」と呼ばれる方法だが、Arduino には標準で、 この手間のかかる PWM 制御の計算を自動で行うライブラリ「Servo ライブラリ」が搭載されている。これは、関数に引数として、モータの制御信号を送るためのデジタルピンの番号と、具体的な角度の値を渡すことで、手軽にその角度までモータを回すことができるライブラリである。

安定を考えていなかった当初はこのライブラリの使用を検討したが、様々な理由から用いることができなかった。理由として、Arduino が提供できる 5V 電源は約 40mA 程度の電流しか無いためである。本 Element に用いたサーボモータの場合、モータ 1 台の場合は動くものの、モータ 2 台を動かす際に電力不足が発生し、痙攣のような動作を起こしたり、一切動作しない問題が発生してしまったりと、不安定な挙動を示した。これは、Servo ライブラリでサーボモータを制御しようとすると、自動的に最高速で動作してしまい、容易に電力不足を引き起こしてしまうためである。

また、他の理由として、細かな調整が行えない事が挙げられる。Servo ライブラリは指定された角度から計算された PWM 信号を生成し、指定したピンへの送信を行うことはできるが、具体

的な回転速度や、加減速といったパラメーターの細かな調整は行うことができない。この問題は、Servo ライブラリを拡張した varServo ライブラリという速度の指定が行えるライブラリが存在していたが、第三者が制作したものであり、公式でサポートされたものではなかったこと、ユーザによるサポートが終了していたこと、このライブラリを用いた電力不足の解決方法では同時に動かせなかったり、非常に低速で動かさなければならず、応答性に欠けることから、こちらも使用を見送り、最終的に後述するサーボドライバで解決を図った。

次に、二つ目の要件について、モータが回転・停止を急速に行った際、慣性の法則から傘本体の運動に腕が引っ張られてしまうという問題があった。このモータの反動を和らげる方法はいくつかあるが、Arduino のみでできる方法としてはモータの速度を下げる事が挙げられる。これは前述した varServo ライブラリでも実現可能だったが、これを用いない方法として一般的なものは、一度に目的の角度まで動かさずに、現在の角度から目的の角度までの動作を一定数に分割し、順番にモータへ指示を行う間にプログラムで待機を挟む事で一時停止を非常に細かく行い、モータの動作速度を下げる方法が存在する。

この手法を用いるにあたって、当初はプログラムの待機を delay 関数を用いて行っていた。しかし、これを用いるとモータは 1 台ずつしか動かすことができなかった。これは、後述する 3 つ目の「同時に 2 つのモータを動かす」という要件を満たさないこととなる。

プログラムの待機に用いていた delay 関数とは、Arduino に標準で搭載されている関数で、引数として与えた値をミリ秒単位として解釈し、その時間の間は全てのプログラムの動作を止めるという関数である。しかし、この delay 関数で指定した時間中は Arduino 全体の動作が止まり、プログラムが全て停止してしまうため、モータが目的の角度へ到達しても停止の指示や次の目的角度の指示が行われずに回り続けるなどの問題が起きる。特に、今回のように反応の速さが必要であったり、二つのサーボモータを制御するといった、並列な処理を行う制作物には不向きである。そのため、delay 関数を用いずにモータの動作速度を下げ、モータの回転の反動を発生しづらくすることで軽減する必要が生じた。

三つ目の要件として、二つのサーボモータを同時に動かす事が挙げられる。サーボモータを同時に動かせない場合、一瞬雨から守ることができない瞬間が生まれてしまい、濡れてしまう瞬間ができてしまう。そのため、速度を落としつつサーボモータを同時に動かす処理を実装する必要があった。しかし、前述したとおり、delay を用いて動作を分割するとモータを一つずつしか動かすことができず、これに代わる他の方法で動作を分割しなければならなかった。

これらの要件があるという前提を基に、プログラム本文の解説を行う。

（※文責: 田中琳仁）

本文解説

```
1 #include "PCA9685.h"
2 #include <Wire.h>
```

まず、1 行目の include 文だが、これは、電力不足を解消するために使用した、”PCA9685”と呼ばれるサーボドライバのライブラリである。これは Element.02「BLWIND」にも用いられた、大量のサーボモータを制御することができる部品の一つである。本来の用途は様々な機器をつなぐ用途で使われるにもかかわらず、Arduino には 12 個しかないデジタルピンを拡張し、サーボモータ専用のデジタルピンを増設、サーボモータのデジタルピンの占有率を下げる目的で用いられる事が一般的である。今回の場合、モーター以外の他の機器は各種センサー数個しか使わないため、デジ

タルピン自体は足りていたが、大量のサーボモータを接続できる性質上、外部電源からモータへ電力を安定供給することができるという機能があり、これを用いることで外部電源を安全かつ容易に取り入れることができるため、電力不足の解決策として採用した。2 行目の include 文はサーボドライバと I2C 通信と呼ばれる規格の通信を行い、データをやり取りする際に必要なライブラリを導入するコードである。I2C 通信を用いることで、Arduino から PCA9685 へモータの動作を指示し、サーボモータを制御するという流れである。

```
1 #define ROTATE 10
2 #define LEAN 11
```

4 行目、5 行目の define 文はモータの制御に用いるピンの変更を容易にするための定義文である。ここで ROTATE と LEAN という文字列をピン番号として定義しておくことで、今後ハード側で変更を行った際にプログラムの編集を行いやすくする意図がある。元々はそれだけの用途だったが、角度計算を行う関数の簡略化のため、計算方法を変える分岐条件としても使用できることが分かったため、2 つのモータを区別しなければならない場面でも使用している。

7 行目、8 行目も同様の理由から定義を行った。こちらはモータの動作を分割する数と分割した動作の間で待機する時間を定義している。可能であれば、後述する分割数を格納する配列と同期するような設計を行いたかったが、時間の都合から困難であると判断したため、この二つの同期は手動で行うこととした。

```
1 int beforeTime = 0;
2 float rotateProcess[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
3 static int beforeRotate = 0;
4 float leanProcess[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
5 static int beforeLean = 0;
6 boolean overMotor = false;
7
8 ServoDriver servo;
9
10 boolean debugOn = true;
```

10 行目から 15 行目はプログラム内で用いる変数の宣言を行っている。

Arduino はプログラムファイル全体の記述順にほとんど関わらず、一度プログラム全体を読み込んだ後に loop 関数内をループする動作を行う。また、関数内で宣言された変数はローカル変数となり、関数外で使う事ができない。これらの仕様から、使用するためにはグローバル変数にする必要があったためここで宣言を行い、グローバル変数としている。

17 行目は 1,2 行目で include したライブラリからサーボモータを動作させる為に宣言を行っている。

19 行目は 10 行目から 15 行目で宣言している変数と同じ理由で debugOn 変数を宣言しているが、デバッグに関わる物なので、あえて他の宣言から改行で離すことでわかりやすく宣言する意図がある。

```
1 void setup() {
2   Wire.begin();
3   Serial.begin(9600);
4   servo.init(0x7f);
5 }
```

21 行目からは setup 関数がある。この setup 関数は Arduino が起動した際に、loop 関数の前に動かす関数であり、各種値の初期化を行うことができる。グローバル変数の宣言とは異なり、全体を読み込んだ後に動作する関数であるため、ライブラリを使用する初期化などは原則ここで行うこととなる。ただし、ここで変数を宣言すると、setup 関数内でのみしか使えないローカル変数となってしまうため注意が必要である。

22 行目はドライバとの I2C 通信の確立を行っている。23 行目はシリアル通信の確立だが、22 行目で行っている、モータの制御に必要な I2C 通信とは異なり、こちらはデバッグデータを PC へ送信するための一文であるため、本番環境では通常不要なものである。本来であれば、debug 変数の値に応じてシリアル通信を起動するかどうかを変えたほうが良いが、関数内の処理が煩雑になることや、シリアル通信をしていても本番で特に問題が起きていなかったため、そのまま本番でもシリアル通信が起動するようにしている。

24 行目は I2C 通信に用いるアドレスの指定を行っている。ここで異なるアドレスを複数指定すると、使用できるアドレスであれば、複数のドライバを運用することができる。ただし、今回はサーボモータドライバ以外は使用しないため、アドレスは 1 つしか指定していない。

25 行目は変数 beforeTime の初期化を行っている。これは、後述する while を用いた動作分割に用いる変数で、プログラムを開始してからの経過時間を示す millis 関数の返り値を 100 で割った値が格納されている。100 で割っている理由として、millis 関数はプログラムを開始してからの経過時間をミリ秒単位で返すため、int 型では容易にオーバーフローを起こしてしまうためである。オーバーフローを防ぐ手段として、long 型を用いたり、一定値を超えた際に初期化を行うなどの方法が挙げられるが、オーバーフローを起こすほど長時間稼働させたことがほとんどなかったこと、初めてオーバーフローを起こして動作に異常が生じ、その原因を突き止めた時期が成果発表直前の時期とかなり遅い時期であったことから、簡易な方法でオーバーフローを防ぐ方針を取った。

```
1  void loop() {
2    debug(debugOn);
3    beforeTime = millis() / 100;
4    while (millis() / 100 - beforeTime < 1) {
5      ;
6    }
7    calcRad(analogRead(A1), LEAN);
8    lean();
9    while (millis() / 100 - beforeTime < 1) {
10     ;
11   }
12   calcRad(analogRead(A1), LEAN);
13   lean();
14   while (millis() / 100 - beforeTime < 1) {
15     ;
16   }
17   calcRad(analogRead(A0), ROTATE);
18   calcRad(analogRead(A1), LEAN);
19   doMotor();
20 }
```

27 行目からは loop 関数が始まっている。前述したとおり、この関数は while や for を用いずとも、

電源が投入され、明示的に止められない限り中身を無限ループするため、基本的にこの関数に書かれた内容で常に動いていると解釈できる。

28 行目は 104 行目の debug 関数を、前述の debugOn 変数の値を渡して呼び出している。debugOn 変数に false の値が格納されている場合、debug 関数は特に何も行わずに関数の動作を終了する。debugOn 変数が true の場合はその時点での各種センサの値と稼働時間を 1000 で割った値をシリアル通信で print する。この内容は、シリアルモニタ等で確認することができるため、デバッグする際に変数の値を確認する機能として動いている。

30 行目から 32 行目は millis 関数の返り値を 100 で割った値と、最後にモータが動いた時間、あるいは Arduino 全体が動作を開始した時間を比較して、一定値以下の間は無限ループするというプログラムである。これが、3 つ目の要件を満たしつつ 2 つ目の要件を満たすことができる、delay 関数を用いないプログラムの待機方法である。全体を通して、delay 関数を用いて待機する部分はこれに置き換わっている。プログラム全体から delay 関数を取り除くことで、プログラムの誤動作や停止を防いでいる。

33 行目は 44 行目の calcRad 関数にセンサの値と動作させるモータのピン番号を渡している。calcRad 関数とは、センサの値から角度を計算する部分を担う、本グループで開発した関数である。計算部分を関数として分けた理由として、loop 内に計算式を記述すると冗長になってしまい、管理が困難になってしまうことが挙げられる。他の一般的なソフトウェア開発と異なり、Arduino は単一の動作ファイルで稼働するため、複数ファイルに処理を分けるといった管理ができない。そこで、関数としてまとめて loop 外に置くことで、メインで動くプログラムを簡略化し、大きな変更があった際に変更を容易にする意図がある。結果として、意図したものではないが、一連の処理を一行の変更のみでオンオフすることができるようになったため、前述したオーバーフローの問題を解決する際に大いに役立った。

```
1  void calcRad(int sensorValue, int type) {
2    int sensor;
3    float temp;
4    switch (type) {
5      case ROTATE:
6        sensor = map(sensorValue, 0, 1023, 0, 360);
7        if (sensor > 180) {
8          sensor = sensor - 180;
9          overMotor = true;
10       } else {
11         overMotor = false;
12       }
13       temp = (sensor - beforeRotate) / PROCESS;
14       for (int i = 0; i < PROCESS; i++) {
15         rotateProcess[i] = (beforeRotate + temp * (i + 1));
16       }
17       beforeRotate = sensor;
18       break;
19     case LEAN:
20       if (overMotor) {
21         sensor = map(sensorValue, 1023, 800, 0, -90)+90;
```

```
22        } else {
23          sensor = map(sensorValue, 1023, 800, 0, 90)+90;
24        }
25        temp = (sensor - beforeLean) / PROCESS;
26
27        for (int i = 0; i < PROCESS; i++) {
28          leanProcess[i] = (beforeLean + temp * (i + 1));
29        }
30        beforeLean = sensor;
31        break;
32    }
33  }
```

calcRad 関数について解説を行う。前述の通り、引数として受け取ったセンサの値とピン番号を用いて処理を行うが、後者のピン番号については、特にモータに対して動作の支持を行わないため、本来必要ではない。しかし、二つのモータの角度計算を１つの関数にまとめようと試みた結果、ピン番号が識別に便利であり、コードを書く際にわかりやすかったことから、単なる識別番号として用いている。

45 行目、46 行目で宣言している変数は計算の結果を一時的に格納しておく変数で、この関数内でのみ使用する。47 行目で、識別番号としてのピン番号を switch 文で判定し、分岐を行っている。

49 行目からが ROTATE、つまり回転を行う下側のモータについての角度計算、62 行目からが LEAN、つまり、傘の傾きを担う上側のモータについての角度計算をそれぞれ行っている。

まず、ピン番号が ROTATE だった場合についてだが、最初に map 関数を用いて、センサの値を角度の値へマッピングしている。map 関数とは、特定の範囲を持つ変数の値を、別の範囲として変換する関数である。例えば、1〜4 の範囲を持つ変数を map 関数を用いて 1〜2 の範囲として変換すると、1 は 1、4 は 2、2.5 は 1.5 として変換される。本来は float 型で小数点以下も出るが、今回は int 型の変数に格納しているため、小数点以下は切り捨てられている。

50 行目では変換された値が 180 度以上であった場合に分岐を行っている。これは、使用しているモータが約 180 度までしか動かないためである。この 180 度までしか動かないという問題は、全方位の雨から守るというコンセプトにおいて解決しなければならなかった。解決策として、この個所のプログラムで 180 度を超えた角度の値から 180 を減算し、overMotor という変数を true に切り替えている。overMotor 変数は、62 行目からの傾きの角度計算に用いる変数であるため、傾きの角度計算の項で解説を行う。

その後のプログラムでは、56 行目から 58 行目までの間で、現在の角度から計算で導いた角度の間の分割角度を配列に格納している。まず、temp 変数に現在の角度と計算で導いた角度の差を等分した値を代入する。次に、現在の角度に temp 変数を配列の番号分足し合わせた角度の値を for 文を用いて配列に代入していく。このプロセスを踏むことで、配列の順番でモータを待機時間を挟みつつ動かすだけで、動作速度を制限したモータの回転を実現できる。

その後、現在の角度を示す変数に、計算後の変数を代入し、関数を終える。

この一連の流れで、次回の回転で使う角度の並びと、次回の計算で使う現在の角度の値を計算し、設定することができる。

識別番号が LEAN だった場合についてだが、計算のプロセスについてはほとんど変わらない。ただし、モータの可動域が 0 度から 180 度である都合から、90 度をデフォルトの角度として設定

するため、map 関数で値を変換するときに、90 を加算している。

また、ROTATE の際の計算で設定されていた、overMotor 変数による分岐をここで行う。overMotor 変数は、下側のモータについての角度計算の項で触れた通り回転角度が 180 度以上かどうかを記録している変数である。この変数に true が格納されている場合、傾く方向を反対にすることで、360 度の範囲から守ることができる。

その後の分割プロセスは下側のモータの角度計算時と同一のものである。

```
1  void lean() {
2    for (int i = 0; i < PROCESS; i++) {
3      beforeTime = millis() / 100;
4      while (millis() / 100 - beforeTime < WAITTIME) {
5        ;
6      }
7      servo.setAngle(LEAN, leanProcess[i]);
8    }
9  }
10 void doMotor() {
11   for (int i = 0; i < PROCESS; i++) {
12     beforeTime = millis() / 100;
13     while (millis() / 100 - beforeTime < WAITTIME) {
14       ;
15     }
16     servo.setAngle(LEAN, leanProcess[i]);
17     servo.setAngle(ROTATE, rotateProcess[i]);
18   }
19 }
```

その下にある lean 関数、doMotor 関数は、calcRad 関数で計算された角度をモータへ反映するための関数である。loop 関数でも用いた、while 文での動作分割を行いつつ、現在角度から目的の角度まで分割された角度を順番にモータへ書き込んでいく処理を行っている。

lean 関数と doMotor 関数の違いは、下側のモータを動かすかどうかであり、lean 関数の場合は傾きを動かす上側のモータのみ動作を行う。二つに分けた理由は、傾きは高い頻度で更新する必要があったためである。

下側のモータで行う回転は、あまり高い頻度で動かすと、腕に負担がかかってしまう他、風によって動いたセンサのデータから行う動作であったため、3 度程度の少量の角度が変わっただけですぐに動作してしまうという場面を避けたいと考えた。そのため、傾きを高い頻度で更新するために上側のモータだけ動かす場面を設け、下側のモータを動かすのは 2 回に 1 回、つまり約 2 秒で 1 回に抑えようと考えた。結果として、毎回二つのモータが動くことによって電力不足が起きたり、腕に過度な負担がかかるといった問題は解消された。

総じて、処理自体は簡単なものであり、最終的な成果物としては比較的簡素なプログラムだったが、事実としてモータの同期や電力不足の解消が主な問題であり、要求されたソフトウェアの動作そのものは特段難しいものではなかった。しかし、パラメータや動作はハードウェア側に変更があるたびに変更する必要があったため、なるべくハードコーディングを避け、関数を用いる、グローバル変数を用いる、define でマジックナンバーを避けるといったコーディングを徹底することで柔軟な対応ができたため、、細かい調整を行うことに注力することができた。

繰り返し行う処理はなるべくまとめるようにコーディングを行ったが、while を用いた待機は関数にまとめると他の処理が割り込めてしまうためにそのまま書いていたり、calcRad 関数をいちいち loop からモータ動作系の関数の前に呼んでしまっていたりといった全体の改善点はあり、それらをプロジェクト期間内に修正できていれば更に読みやすいコードになっていたと考えるが、動作自体は問題なくできており、モータを制御するという観点の 3 つの要件は達成していると考える。

（※文責: 田中琳仁）

第 3 章　中間発表

3.1　Element.01 「Copypen」

3.1.1　中間発表会を受けて

中間発表に向けて、まず、必要な機構の洗い出し、ペンのプロトタイプの作成、発表資料作成を行った。Copypen に必要な機能は主に 3 つあり、「読み取り」「書き込み」「位置情報の取得」であったが、中間発表の時点では、最低限の機能を有したプロトタイプを作成するため、「書き込み」と「位置情報の取得」を優先的に制作した。「書き込み」では、芯を出すサーボモータを使用し、サーボモータとペンを接合する部品を 3D プリンターで制作した。「位置情報の取得」では、超音波センサを用いて、任意の壁との距離を読み取る方法を採用し、壁が付いたフィールドをペンとは別に作成した。そして、すべてのセンサは M5StickC というマイコンを使用して制御した。また、中間発表用プロトタイプの外装は、MDF を加工して作成した。大きさは縦 60mm、横 60mm、高さ 175mm となった。

（※文責: 村上太一）

3.1.2　中間発表会でのフィードバックを受けて

中間発表会の時点では、2 つの課題と 1 つの改善点があった。課題の 1 つ目は、読み取りの機能が存在しないことであった。そのため、元の情報を読み込ませておかないと書き込みが行えないため、柔軟性が非常に低かった。課題の 2 つ目は、距離を 1 方向しか計測できないことである。超音波センサを使って距離を取っていたが、超音波センサは 2 つ以上つけると音波が共存できず距離を測ることができなくなってしまったので、超音波センサでは 1 方向しか計測することができなかった。そして改善点とは、芯を出す機構に関するものである。この時点では、サーボモータを用いて芯の出し引きを行っていたが、動作させる方向によっては芯が滑らかに動かず良い書き心地とは言えなかった。そのため、書き心地が良くなるような改善を行いたいと考えていた。

中間発表会でのフィードバック及び質問でいくつかの課題点が挙げられた。以下は、フィードバックで寄せられたフィードバックと質問、それに対する回答である。

F. インタラクションエレメンツとは何かをもっと詳しく説明するといいと思いました

F. 構造上あのサイズになるのはわかるが、製品化するとした時もっと小型でポップな見た目になるとさらに良くなると思いました。

F. 今後の展望も説明があり、心地いい素材の選定ようにすでにいろんな素材を買っていると言っていたので最終発表が楽しみだなあと思いました

Q. どこがインタラクションなのか。

A. ユーザが手を動かして、コピーする体験を得ること。

Q. スキャンする機構はどう作る予定なのか。
A. カラーセンサやフォトリフレクタで得たデータを変換する。

Q. ペンタブレットなどと比べてどこが便利なのか。
A. 便利さを求めて作られたものではなく、楽しい体験を出来ることを重要視している。

これらのフィードバック及び質問により、自分たちの中でも統一されていないことがあることに気付いたため、本 Element に対する、各メンバーの思考を共有した。読み取りの機構に関して、カラーセンサとフォトリフレクタはどちらのほうがより優れているのかについて話し合った。「Copypen」にはまず、白黒で書き込むことができればまずは完成といっていいだろうということになり、「Copypen」ではフォトリフレクタを使うことになった。また、超音波センサはやはり自分たちの中では機能として不足しているということになり、そこで現在のマウスを使用して位置情報を取得する方法を使うことになった。しかし、取り入れるのは困難であったフィードバックもあった。サイズを小さくできたほうがよいという意見に対して、グループメンバー全員がそれを実現したいとは思っていたが、配線やセンサ、マイコンなどの兼ね合いでどうしても大きくするしかなかった。ただ、デザインに工夫を凝らすことはできるということになったので、最終発表では素材やデザインに工夫を凝らした。

（※文責: 村上太一）

3.2 Element.02「BLWIND」

3.2.1 中間発表会を受けて

中間発表会に向け、私たちのグループではプロトタイプの製作、コンセプトムービーの制作、そして発表用スライドの作成を行った。

プロトタイプは、主にサーボモータを活用してスラットを制御する部分を実装した。この実装では、Arduino を使用して 10 個のサーボモータの角度と動作間隔を細かく調整し、10 枚のスラットが連続的に動くことで波のような動きを実現した。プロトタイプの仕様は、外枠が 600mm × 690mm、スラットが 500mm × 50mm である。これらの寸法は、スチレンボードを使用することで加工のしやすさを考慮している。スチレンボードはレーザーカッターを利用して加工した。また、サーボモータとスラットを接続する部品は 3D プリンタを用いて製作した。これにより、機構の組み立てがスムーズに行え、安定性を確保でき、さらにスラットの素材をさまざまに変えることができるようになった。このプロトタイプによって、本 Element 実現時の設計問題を洗い出すことができた。

コンセプトムービーでは、プロトタイプの映像と 3DCG モデルを組み合わせ、シミュレーション映像を作成した。この映像を通じて、当 Element のコンセプトや完成時の利用イメージをわかりやすく伝えることを目指した。

発表用スライドでは、インフォグラフィックスや動画を利用して、プロジェクトのコンセプト、動作の仕組み、そして制作のプロセスについて詳細に伝えることを意図した。これにより、聴衆に対して分かりやすく情報を提供し、プロジェクトの魅力や成果を効果的に伝えることを目指した。

（※文責: 菅英寛）

3.2.2　中間発表会でのフィードバックを受けて

以下は中間発表での質問とそれに対する回答である。

Q: ブラインドなのに光を遮るものではないのか。
A: 窓辺にあって部屋の内外の境界であることに着目し、風をスラットの連続した動きで表現することが目的であるため、ブラインドの光を遮るという機能は重要視していない。

Q: 実際にはどのくらいのサイズを想定しているのか。
A: 一般的な窓くらいのサイズを想定している。

Q: サーボモータの駆動音については対策を考えているのか。
A: まだ考えていないため今後対策を考えていく。

これらのフィードバックを受け、今後私たちが実現したいことを含めて、中間発表会後の展望を定めた。まずは、次のステップとして、風の情報を取得するセンサ部の実装が挙げられる。本 Element の目標は、実際の風の情報をリアルタイムで取得し、それに基づいてスラットの角度や動作間隔、動作速度などを制御することによって、風をスラットの連続する動きで表現することである。そのため、この風の情報を取得するセンサ部の作成が優先度が高いタスクとして挙げられた。案として、風見鶏のような風向計と風車のような風速計を組み合わせて風向・風速を計測するもの、二つの風車のような風速計をそれぞれ反対の向きにつけることで、二つの風速計の計測値の差から風向を計測するものなどが挙げられた。これらのアイデアの中から、プロトタイピングを行い、風向・風速の計測が効果的にできる方法を採用することに決定した。

次に、心地よい動きとはどのような動きか、どのような動きが心地よく感じる動きなのかを明確にし、心地よい動きをきちんと定義することに焦点を当てた。中間発表会の時点では、心地よい動きとはどのような動きなのかが定まっていなかった。この問題に対処するため、3DCG を用いたシミュレーションを利用して、さまざまな動きのパターンを作成し、それを用いた評価実験を行うなどして、科学的な根拠を持って心地よい動きを定める方針で決定した。この時点では具体的な調査方法までは決定せず、まずは風の情報に基づいて動くプロトタイプを完成させることを優先することにした。

プロトタイプに関しては、中間発表会の時点のプロトタイプでは、スチレンボードの強度が不足していて、サーボモータの重さによって撓んでしまい、構造が歪んでしまうという問題が発生していた。成果物のクオリティを高めるためには、この強度不足の問題を解決する必要があった。そのため、強度不足の解決策として、強度を確保するために設計をし直す方針で決定した。具体的には、ミスミの 5 シリーズ 20mm 角、1 列溝アルミフレームを利用して強度を保ち、その外側を化粧板で覆うような設計に変更することで、強度とデザインの両立を図ることを計画した。同時に、中間発表会の時点のプロトタイプではサーボモータを吊るす形で上に設置していたが、サーボモータの重みで構造が歪んでしまうという強度問題に加え、重心が高く不安定になったり、配線の延長が必要になったりするなどの問題が発生していたため、サーボモータを下部に配置するようにした。また、中間発表会の時点のプロトタイプでは 10 枚であったスラットを 16 枚に増やすことにした。

16 枚にした理由としては、枚数が多い方が動きをより連続して見せられ、Arduino Uno で使用可能なサーボモータ用のモータードライバ PCA9685 で制御できる最大の数が 16 枚であったからである。さらに、設計の見直しとともに、スラットやフレームを囲う素材についても、動きへの注目を妨げる視覚的なノイズにならないようなものを検討し選定することにした。例えばスラットの素材は、角度変化がわかりやすいように、明暗の変化がわかりやすい白い素材で、不必要な光の反射が起こらないよう表面がマットなものを採用することにした。またアルミフレームを囲う化粧板も、スラットの素材に揃えて白く表面がマットなものを選ぶことにした。そのほかにも、サーボモータの動力を確保する電源と Arduino の電源を共有すると Arduino の動作が不安定になってしまうため、電源を別に分けるような設計にするなどの問題解決も計画した。

さらに、プログラムについてもさまざまな改善をする必要があった。中間発表会の時点のプログラムでは、サーボモータを一つずつ独立して制御することしかできなかった。そのため、連続感のない動きしか実現できず、目標である連続した動きで風を表現するということを達成するには不適切なプログラムであった。さらに、Blender でシミュレーションしていた動きを完全に再現することができていないという課題もあった。これら課題を解決するためには、サーボモータを同時並行で制御できるようなプログラムに改良する必要があった。そのためには、delay() 関数の使用を避け、millis() 関数を活用して、擬似的なマルチタスキングを実現するプログラムに改良するという目標を立てた。以上の展望と課題に対する解決策を考慮し、今後やるべきことをリストアップし、後期の活動計画を立てた。

（※文責: 菅英寛）

3.3 Element.03「雨守り」

3.3.1 中間発表会を受けて

本グループでは、音を使用した Element を制作したいメンバーで集まり、音に反応するインタラクティブな作品について考えた。全体のメンバーが制作した「ついついやってしまうこと図鑑」や、それに基づいた Element アイデアの KJ 法を参考に、グループメンバー内で作るものについて話し合いを行った。また、話し合いでは、グループ内で Element アイデアを出すために新たに KJ 法を行った。KJ 法以外にも、チャットアプリ内にグループメンバーが思い付いた Element アイデアを書き溜めていった。こうして集まった Element アイデアについて、プロジェクト担当教員に相談し、アイデアの面白さや先進性について精査を行ってもらった。こうした考案を繰り返したが、なかなか納得のいくアイデアを思いつくことができなかったため、グループ内でもう一度、最初に行った全体での「ついついやってしまうこと図鑑」や、それに基づいた KJ 法に立ち返ってみることにした。そして、その KJ 法の中の「雨の方向に傘が向く」という案に注目し、それを参考に「雨守り」という Element を考案した。

次に、雨守りの機構についての考案を行った。雨守りの制作を行ううえで重要になる部分は 2 つある。雨が降る方向を検知する方法と、その方向に傘を傾ける方法である。傘を傾ける方法については、早い段階で機構が決まった。傘を傾けるために、サーボモータを 2 つ使用する方法を採用した。その方法とは、水平方向のサーボモータと、垂直方向のサーボモータを連結する方法である。この 2 軸のサーボモータにより、傘の 360° の回転を実現した。一方で、雨の方向を検知する方法の考案は難航した。雨の方向を検知するということは、つまり風の方向を検知するということである。風向を検知するセンサの案は、グループ内で様々な考案が為された。例えば、光センサを利用

する案があった。その案とは、光センサを内蔵した箱があり、この箱の横の 4 面はカーテンのような軽い布で仕切られている。カーテンが風であおられ、箱の中に光が入ってきた際に、光センサが光量を検知し、風量を測る。この光センサとカーテンの仕組みを 4 面、または 8 面に設置することで、風向を検知するという案である。しかし、雨が降る状況では雲が多く、外が暗いので光量の差が測りづらい可能性があるため、廃案となった。また、他には音センサを利用する案があった。この案では、傘の雨を受け止める面の内側に音センサを貼り付け、面に当たる雨の音量の差を計算することで、雨が降っている方向を測る。しかし、傘が傾いたときに雨の当たり具合も変化してしまい、雨の音量の差が上手く測れないのではないかという懸念に加え、音センサの防水面での心配もあり、この案は廃案となった。そして、最終的には M5StickC というマイコンと、それに内蔵されてある角度センサを利用することにした。この角度センサを使用した案は、M5StickC を内蔵できる十字型の羽を作り、それを傘の柄から糸で垂らすというものである。十字型の羽が風にあおられた際、そのあおられた角度を M5StickC 内蔵の角度センサが測ることができる。この風向センサの案から、全体の機構を考案した。その機構は、風センサで計測した角度データを Arduino というマイコンに送信し、サーボモータの回転を計算をして制御するという機構である。この機構をもとに、プロトタイプの制作を始めた。プロトタイプでは、竹ひごや厚紙で制作した小さい傘の模型を使用し、傾きの制御の検証を行った。サーボモータは、水平方向のサーボモータのジョイント部分に垂直方向のサーボモータをテープで止めた。その垂直方向のサーボモータの駆動部分に傘の模型を接着し、想定している雨守りの動作を表現した。M5StickC に取り付ける十字型の羽は、発泡スチロールで制作した。M5StickC と Arduino、サーボモータの動作機構は、M5StickC の傾きを Arduino が計算し、そのデータをもとにサーボモータが回転し、傾くように制御した。

中間発表に向けて、M5StickC の風センサ案を採用したプロトタイプや、CG モデリングを使用したシミュレーション動画、発表資料やポスター、プロモーション動画を制作した。プロトタイプは上記の物を使用し、M5StickC による雨守りの動作を見せることとした。CG モデリングを使用したシミュレーション動画には、CG モデリングソフト「Blender」を使用した。プロトタイプの実演だけでは想像しづらい実寸大の雨守りの動作を、シミュレーション動画によって補完することにした。シミュレーション動画では、雨が降っているフィールドを用意し、その中で実寸大の傘を使用した雨守りの動作を表現した。雨が降る方向の傾きを変えることで、風が吹いている方向を表現し、その方向に準じて傘の柄から垂れ下がっている風センサを傾けた。その風センサの傾きに応じて、傘の中棒の中間地点にある 2 つのサーボモータを動かし、傘を傾けるようにアニメーションを制作した。発表資料の作成には Adobe Illustrator を使用した。発表資料には、スライド、ロゴ、写真、シミュレーション動画、インフォグラフィックスを使用した。このとき、雨守りのグループは人数が少なく、作業時間の確保が厳しいため、他 Element 制作グループメンバーの協力を得て、資料を作成した。ロゴは Adobe Illustrator で作成し、傾いている様子の傘をシンプルに表現した。写真には、行った KJ 法の写真や、制作したプロトタイプの写真、プロトタイプを制作している様子の写真、想定している実寸大の雨守りの 3D モデルの画像を使用した。インフォグラフィックスの画像素材には Blender で制作した 3D モデルを使用した。画像素材は、想定する雨守りの全体像、風センサ、連結した 2 つのサーボモータを用意した。この画像素材を使い、雨守りの機構や動作、センサが取得したデータの流れをインフォグラフィックスとして作成した。ポスターには、Interaction Elements 全体の紹介部分と、各グループの Element 紹介部分があった。我々グループは、Element 紹介部分に必要な雨守りのロゴ、写真、インフォグラフィックス、コンセプトや使用方法の説明、機構や仕組みの説明の文章とそれの英訳を用意し、ポスター制作担当に渡した。プロモーション動画には、実際に撮影した映像を使用した。この際、全グループで、実際に使用し

ている場面を想定したものを制作するというコンセプトのもと動画制作をした。そこで、雨守りグループのプロモーション動画には、「オズの魔法使い」というプロトタイプ技法を採用した。この技法は、実際には仕組みがまだできていないプロダクトを、完成しているように人力で動きを見せかけるという手法である。我々は風センサが傾いている様子の映像と傘が傾いている様子の映像を用意した。傘が傾いている映像に風センサの映像をワイプで表示し、同時に傾けさせることで、まるで風センサによって傘が傾いているように見せかけるプロモーション動画を制作した。

（※文責: 橘孝則）

3.3.2　中間発表会でのフィードバックを受けて

中間発表を経て、雨守りに頂いたフィードバックは「車椅子を対象にしたものであれば、利便性は良さそうだと思った。使用者を限定するのはどうか」「重さを改善するための策として、背負い型という話をしていたが、リュックを普段利用する方であれば使いづらいかもしれない」「傘が傾いたときに、傘の橋が真上に来た時に水滴が落ちるのではないか」「雨が風であおられた時、空気抵抗などを考慮した、現実での雨の横方向の速度を分析する必要があるのではないか」などがあった。

まず、車椅子の案についてグループで話し合った。確かに使用者を絞り込んだ状況を想定すると、雨守りの「傘が自動的に傾く」というコンセプトにさらに意味や説得力を持たせられるという意見が挙がった。しかし、車椅子に装着した際、傘が傾いたときに重心が変わり、車椅子が倒れてしまうのではないかという懸念や、車椅子走行時に他の歩行者に当たってしまうのではないかという懸念も挙げられた。このように、車椅子に関する知識が足りていないため、制作期間を鑑みた結果、使用者の限定については考慮しない方向でまとまった。背負い型の案については、背負い型雨守りとリュックを同時に背負うと、邪魔になってしまうというフィードバックが寄せられた。その点については、以前からグループ内でも話し合われていたことであったため、この問題点に関しては、まず制作を進めた後に改善を行うことにした。傘が傾いたときに雨が滴る可能性については、傘の雨を防ぐ箇所の形状についてグループ内でいくつか案が挙がった。傘の上部を後方に長くする案や、全体的に広くする案などが出たが、傘を傾ける際に問題が発生するため難航した。風にあおられた雨の、実際の傾き具合については、ひとまず風の強さのみで測り、傘を傾けることとした。

（※文責: 橘孝則）

3.4　中間発表資料・ポスター制作について

デザインコンセプト

スライド・ポスター・動画のデザインは、統一感を出すため、白と黒を中心としたモノトーンなデザインとした。本プロジェクトでは、今までにはない新しく、面白い Interaction Elements を制作することを目的としているので、新しく生み出すことをイメージして洗練された白と黒のシンプルなデザインを意識した。

（※文責: 冨田夏央）

3.4.1 発表資料制作

プレゼンテーション資料制作

発表スライドはデザインコンセプトに合わせて、InteractionElements のイメージを崩さないようなシンプルかつ伝わりやすいデザインを心掛け制作した。グループごとの紹介と全体の紹介でそれぞれ異なる魅力を伝えられるように内容を工夫し、アイデア出しの過程を矢印のようなフロー図で作成したり、画像を複数枚使用したりして誰でもわかるようなデザインを意識した。制作には Adobe Illustrator を使用し、ガイドを用いてフォントや画像、ロゴなどの位置を細かく調整し、体裁を整えた。また、余白を大きくとることでみやすくシンプルなデザインを心掛けた。さらにポスター・動画と全てのフォントを統一することで発表全体で統一感がでるよう工夫した。

（※文責: 井上芽依）

動画制作

動画制作では各グループのコンセプト動画の制作を行った。動画に用いる素材の撮影は全て学内で行った。それぞれのグループで動画素材を撮影し、グループの動画編集担当がコンセプト動画を制作した。その後、できた動画を結合し 1 つの動画とした。動画の編集と BGM の追加には Adobe Premiere Pro・AviUtl、モーショングラフィックスの作成には After Effects、特殊な図形素材の制作には Adobe Illustrator、アニメーションの制作には Blender を用いた。また、各グループの動画で統一感を出すために、字幕のフォントや BGM の雰囲気を揃えた。動画内容には最終的に目指している制作物の動作と使用例を組み込むことで統一感を持たせた。

（※文責: 樫木海生）

3.4.2 ポスター

ポスターは Adobe Illustrator で制作した。タイトルや文章のフォントを統一することで、デザインのコンセプトをポスター内で統一した。また、Interaction Elements のロゴをタイトルの左隣に配置することで強調させ、プロジェクトのイメージ付けがされるように意識した。文章は全て日本語と英語のバイリンガルとし、多くの人に伝わるポスターを心がけた。また、可能な限り線や装飾は使わず、文章や情報のブロックをラインに合わせてレイアウトするグリッドレイアウトを意識することで、整理され見やすいポスターを目指した。しかし、各 Element で、紹介部分の文量に差ができてしまったため、グリッドレイアウトを意識しきれない部分も見受けられた。各 Element の紹介部分では、情報のブロックを縦にレイアウトし、構図に変化をつける工夫をすることで、一番注目してもらいたい Element の紹介部分に目が向くようなポスターを目指した。各 Element の紹介部分では、概要と仕組みについての詳細を説明した。概要と仕組みについての説明を分けることで、それぞれの内容が簡潔に伝わるような工夫をした。仕組みの説明の文では、インフォグラフィックスを用いることで、構造についての理解が深まるようにした。本文は、和文フォントを游ゴシック体、欧文フォントを Futura として合成フォントを作り使用していたが、和欧混色の組み合わせが自然ではなく、統一感がないという改善点が見受けられた。ポスターに使用したプロダクトの写真では、ライティングや撮影状況の違いから、各 Element の背景の明るさに違いがでてしまっていたため、ポスター全体のまとまりが損なわれていたという改善点が見受けられた。最終発

表で使用するポスターでは、見受けられたいくつかの改善点に気をつけて制作していきたい。

図 3.1　中間発表時のポスター

（※文責: 大塚創生）

3.5　中間発表方法について

3.5.1　プレゼンテーション資料

プレゼンテーション資料は、オンライン上でプロジェクトメンバーが相互に編集しやすい環境を構築するために google slide の共有機能を用いて作成した。全体で発表するスライドと各グループで発表するスライドを作成し、全体発表でプロジェクトの概要を説明した後、各グループでより細かいエレメントの詳細を発表した。あらかじめフォントなどのデザインルールを共有しておき、それに沿って作成した。

ポスターと共通のデザインルールのもと、スライドを作成したため、発表会では統一感のある空間を作り出すことができた。また、全体で発表するスライドの右下にロゴを入れたレイアウトとなった。

全体スライドの構成

もくじ

全体発表は約 5 分間のプレゼンテーションのため、初めに全体の概要を示すもくじを作成した。もくじでは、当日説明する内容について大きく触れ、順を追って説明していくことを伝えるページとした。

メンバー

プロジェクトメンバーが学生 13 名、担当教員 3 名という大所帯のプロジェクトのため、個人の詳細なプロフィールはなく、学生と教員という見出しのもと、氏名を列挙した。

Interaction Elements とは

作成した Element の紹介に入る前に、まず Interaction Elements とは何かを伝えるページを作成した。身近で分かりやすいドアノブ・電気のスイッチを例に挙げた。また、説明的な文章として、１文程度で Interaction Elements の概要を説明した。

制作プロセス -図鑑の作成-

実際に Element の作成に入る前に、実際にフィールドワークを行うことによって身の回りにある心地良いことをまとめた「ついついやってしまうこと図鑑」と称した図鑑の紹介を行った。ただ、図鑑を載せるだけでなく、図鑑の内容にはどのようなものがあるかを１文で簡易的に紹介した。

制作プロセス -KJ 法によるマッピング-

作成した図鑑から、KJ 法によるマッピングを行って、アイデア出ししたことを紹介するページを作成した。実際にマッピングを行った時の写真を取り入れることによって、視覚的にどのようにアイデア出しをしたのか理解しやすいページとした。

制作プロセス -アイデア出し-

KJ 法によるマッピングをもとに、五感や特徴などからアイデア出しを行ったことを説明した。全体のアイデア出しの風景をまとめた時の写真と具体的なアイデアをアップで撮影した写真を２枚掲載することで、アイデア出しの内容が理解しやすいページとした。

提案プロダクト

KJ 法によるアイデアのマッピングから派生して各プロダクトを着想したということを説明するために、特に大切にした五感を強調し、各 Element の紹介をした。

全体スライドのグループによる Element の紹介

全グループで構成を統一し、各エレメントのコンセプトとそれぞれのアイデア出しのプロセスを紹介した。着目した五感からどのように現在のエレメントに落とし込んだのかを段階的に説明した。

個別スライドのグループによる Element の紹介

全グループで構成を統一し、Element の概要→コンセプト→今後の展望という構成で行った。各グループが中間報告時までに完成している Element の写真や資料を組み込むことで、これからの展望を見せる Element の紹介とした。

デモ体験

各グループが個別スライドによる発表を終えた後、それぞれのタイミングでデモ体験を行った。聴衆に中間発表時点でのエレメントの動作を見せることで、進捗がどの程度であるか、また今後の展望を視覚的にも理解してもらうことができた。

ポスター

Interaction Elements とは何であるか・制作スケジュール・各エレメントの説明およびインフォグラフィックスを掲載したポスターを発表ブースに展示した。

（※文責: 工藤翔太）

3.5.2 プレゼンテーション方法

ロジェクト学習の発表会はオープンスペースで行われることから、他のプロジェクトに発表内容をかき消されたり、ぼやけてしまわないための自分達らしい発表会を意識したプレゼンテーションを心がけて行った。声量面ではマイクなどの用意はできなかったため、発表者 1 人 1 人が、発表ブース全体に届くような声量、滑舌を意識することで聞き取りやすく感じられる発表とした。

（※文責: 工藤翔太）

第 4 章　成果発表会

4.1　Element.01 「Copypen」

4.1.1　成果発表会に向けて

私たちは成果発表会に向けて様々な活動を行った。まず、Element Copypen の最終調整のための制作を行った。これには多くの時間と労力が注がれ、最終的には素晴らしい結果を得ることができた。中間発表会の時点では、Copypen は位置の取得に赤外線センサと超音波センサを用いており、これには壁を伴った特殊なフィールドが必要であった。しかし最終発表会までに、マウスを用いた新しい位置の取得方法を取り入れたことで、Copypen は特殊なフィールドを必要とせず、独立して使用することが可能になった。また、中間発表で挙げられた、芯が出すぎて Copypen 本体が浮いてしまうという反省点を生かし、芯の出力機構を改良した。輪ゴムを用いてボールペンの中のばねのような役割を持たせることで Copypen 本体が浮かなくなるようになっている。また、ロゴの制作も行った。Copypen という文字の C と P を組み合わせることで、対称性を持ったシンプルなロゴを作成した。このロゴは、認知的負荷が少なく、視聴者にとっても親しみやすいものになった。デザインの微調整や機能の最適化など、細かい部分にも時間と努力を注いだ。私たちの目標は、視覚的な魅力を高めるだけでなく、ブランドのメッセージや価値観を伝えることであった。そして、それを達成するために、ロゴのデザインにおいて各要素のバランスを考え抜いた。

総じて、私たちのロゴは視覚的に魅力的で意味のあるデザインとなった。これにより、私たちのブランドの存在感と認知度を高めることができた。結果、私たちのチームは、デザインの力を最大限に活用し、視覚的な引き付けを強化することに成功したといえるだろう。

さらに、プロモーション動画の作成も行った。映像の撮影では、ホワイトバックと照明を使用し、また Element を撮影する角度にもこだわった。また、動画内での編集にも配慮し、視聴者が退屈しないように角度を変えたカットを挟む工夫を行った。プロモーション動画の効果を最大限に引き出すために、音楽の選曲にも力を入れた。楽曲の選択には、視聴者の感情に響く曲を厳選した。さらに、音楽と映像のシンクロを図るために、編集作業で細かいタイミングの調整を行なった。このような努力が実を結び、見栄えの良い動画を作成することができた。

撮影内容は、Copypen がひらがなの「た」のイラストをこする動きでコピーし、こする動きでペーストする様子を撮影した。ひらがなの「た」を使用した理由は、その文字が縦と横の垂直、水平な線でシンプルに構成されており、Copypen の性能を視聴者に伝えるのに適していると考えたからである。他の文字や形状では、Copypen の性能や操作方法を的確に伝えることができない可能性があった。ひらがなの「た」は、そのシンプルさと明瞭さから、視聴者に Copypen の使いやすさや高性能さを効果的に伝えることができると考えられる。

さらに、インフォグラフィックスの作成も行った。インフォグラフィックは Copypen の中身が分かりやすいように、できるだけシンプルに描いた。これらのグラフィックスは、発表時に使用する動画や発表資料、ポスター、Web サイトなどに活用された。

最後に、発表資料の作成についてである。Figma を使用してスライド資料を作成した。最初は Illustrator を検討していたが、Figma の共同編集機能が優れていることから、Figma を採用することにした。ただし、一部の図形が Figma では作成できなかったため、途中で Illustrator も併用

する必要があった。

以上が私たちの活動の概要である。成果発表会に向けて、様々な面で努力を重ねたが、その結果、素晴らしい成果を得ることができた。

（※文責: 島田麻飛）

4.1.2　成果発表会でのフィードバックを受けて

成果発表会での質問及びフィードバックを受けて、今後の改善点が明確になった。まず、フィードバックでは、以下の内容が挙げられた。以下は発表に関するフィードバックである。F は送られてきたフィードバック、A はそれに対する返答を表す。

F : 説明中に噛んでしまったり、言葉が出てこなくて止まってしまうことがそこそこあったので、もう少し練習した方がいいと思います。 内容はとても良かったです。

F : 全体のに向けた概要発表が少し長いかなと感じた

F : 全体的にシンプルなスライドで読みやすく綺麗にまとまっていたと思います。 中間発表と同じものだと思いますが、グループ活動の概要をスライドと言葉だけで伝えるだけではなく、やはり動画で動かしている様を簡単に知れるのが良かったです。 アトリエだったので聞こえないというわけではないのですが、もう少し声量があると発表の質がより高まると思いました。 雨守りの話を聞いていたのですが、センサの部分の話を「動作の流れ」のページではなく別のページのときにセンサの話があったので、センサの部分はセンサのスライドで話をしているともっと分かりやすくなると思いました。

F : Copypen を触らせてもらった。 十円玉を鉛筆でうつすことを例にあげられていたときに、「楽しい」とおっしゃっていたが、どの部分が楽しいにつながっているかをもっと知りたかった。私は十円玉の凹凸をなぞる部分が楽しいと思い、プロトタイプでは鉛筆が動いている感覚が手に伝わって、精度に拘らず楽しかったです。

F : 流れがきれいでわかりやすい発表 物がしっかりとできているのが高評価 早くデモがみたい気分になって発表が長く感じた。

以下はプロジェクトの目的と成果に関するフィードバックである。

F : 荒さが生まれるのがこすり出しらしさが出て個人的にはいいと感じた ただモニターを見るに読み取り時とこすり出し時に太さが違う＝それぞれで 1 回ごとに下に下げる適正量が変わってしまうように感じたので、ここを改善するとより使いやすくなるのではないでしょうか（この指摘が間違ってなかったらだけど……）

F : Copypen データのコピーが難しそうだなと思った。センスが必要

F：補正アルゴリズムをさまざまな物を試しており、その中で、マス目を取る方法が最も良い結果を出しているのが面白いと感じた

F：途中から参加したのに丁寧に説明してくれて助かりました。

F：Copypen (?) の説明を聞きました。デモ可能な状態まで制作をがんばったのは評価できると思います。ただ、フロッタージュのメタファというのは、若干無理があると思いました。言われなければちょっとわかりません。タンジブル UI の分野では、けっこう昔から現実世界のコピペデバイス（ブラシとか）は提案されてきているので、それらとの差別化を意識した説明なのかもしれませんが。 他の 2 つのデバイスの説明は聞けませんでしたが、それぞれ面白いと思いました。

以下は成果発表会での質問とそれに対する回答である。

Q：読み取りデータの補正アルゴリズムについてである。メディアンフィルタを試すのはどうか。
A：メディアンフィルタについては、実際に試してみたが補正がうまくいかなかった。ノイズの除去に関しては、メディアンフィルタは効果を発揮したが、データが足りないところの補間に関しては効果を発揮しなかった。また、他にも膨張、縮小のアルゴリズムを試してみたが同様にうまくいかなかった。

Q：最終発表のデモでひらがなの「た」のイラストを用いていたが、他のイラストでも描くことは可能なのか。
A：可能である。他にも、ひらがなの「え」のイラストやおにぎりのイラスト、矢印のイラストなどを用意していたが、デモの時間の関係上披露しなかった。

（※文責: 島田麻飛）

4.2 Element.02 「BLWIND」

4.2.1 成果発表会に向けて

私たちは成果発表会に向けて、Element 制作に加え、プロモーション動画の作成、インフォグラフィックスおよび発表用スライドの作成などを行った。Element 制作において、ハードウェアの面では、まず Fusion360 を利用して 3D の設計図を詳細に作成した。設計段階では、動きに注目するのを妨げる、視覚的なノイズになる要素をできるだけ排除するように行い、4 辺の外枠とスラットの隙間を揃えたり、中間発表の時点のプロトタイプでは外から見えていたサーボモータやそれに伴う配線を囲って隠すような設計にしたりした。次に、設計図をもとに作成を進めた。材料は骨組みにミスミの 5 シリーズ 20mm 角、1 列溝アルミフレームを用い、その周りを白のアクリル板で囲った。アクリル板は表面を研磨した上で塗装し、表面に無光沢の加工を施した。スラットの素材にはサーボモータでしなやかに動かせるように、加工の難易度も考慮して白の低発泡塩ビ板を用いた。また、安定して動作させるために、3 D プリンタを用いてサーボモータとスラットの接続部品、空回りキャップを作成し、これも研磨した上で白で塗装することによって、表面のディテールや色味を他の素材と揃えることで視覚的なノイズを最小限に抑えるようにした。また、風の情報を

取得するロータリーエンコーダを搭載する風車についても 3D プリンタで作成した。センサの下部には Arduino Nano が収納できるようなケースを制作し、また通信のためのジャンパワイヤを白くすることで、プロダクト全体の統一感を出し、視覚的なノイズを少なくするようにした。

ソフトウェアでは、スレーブは、ロータリーエンコーダの回転方向から風向を、また一定時間の角度変化から風速を取得し、角度に変換した値をマスタに送信するようプログラムした。また、サーボモータを制御するマスタは millis() 関数を利用して、短い周期でサーボモータの制御を繰り返すことで、複数のスラットが同時並行に独立して動くように見せる擬似マルチタスキングを実装した。スレーブから送信される角度を最大角度とし、少しずつ回転角度が減少するようにプログラムしたり、風が強ければ強いほど、サーボモータの動作間隔が短く、サーボモータの駆動が早くなるようにしたりして、風の強さを自然に動きに反映するよう工夫した。

スラットの一連の動きは、3DCG によるシミュレーション映像を作成し、それを用いて順位評価と評価グリッド法による質的分析によって決定した。この調査はプロジェクトメンバー 12 人に対して行われた。パターン A：角度変化・動作間隔どちらもだんだん小さく、パターン B：角度変化・動作間隔どちらも変化しない、パターン C：角度変化だんだん小さく・動作間隔変化しない、パターン D：動作間隔だんだん小さく・角度変化変化しないという 4 パターンの動きを作成し、順位評価での平均順位が高いパターン C の動きをベースに、評価グリッド法からわかった心地よい動きの要素である「自然」と「一定」を取り入れた動きを心地よい動きと定義し、実装した。

プロモーション動画においては、本 Element がどのような動作をするのか、またどのような提供価値があるのかを効果的に伝えられるように、実際の動作と利用イメージを撮影して編集した。インフォグラフィックスや発表スライドは、Element の動作の仕組みはもちろん、制作するうえでこだわった点や、どういったプロセスで制作したかなども伝わるように工夫して作成した。

（※文責: 菅英寛）

4.2.2　成果発表会でのフィードバックを受けて

以下は成果発表会での質問とそれに対する回答である。

Q. なぜブラインドを選んだのか

A. ブラインドは窓辺に設置するインテリアで、窓辺は屋外と屋内の境界であるため。屋内外の境界線に設置されるブラインドに風を表現する機能を導入することで、その境界線を溶かし、屋内に新たな開放感を与えるという提供価値が生まれることが期待できると考えた。

Q. 心地よい動きを決定する調査をプロジェクトメンバー内で行ったが、それでは規模が小さいのではないか。

A. 今回のプロジェクトでは、時間的な制約により調査対象をプロジェクトメンバーに限定せざるを得なかった。そのため、統計的な根拠を求めるにはデータ数が不足していると認識している。ただし、評価グリッド法を用いた質的な分析によってどのような要素が心地よさに繋がるのかの意見や評価を調査し、それに基づいて動きを決定する根拠を得ているため、確かに調査の規模は小さいが、科学的な根拠として成り立つと考えている。今後の展望として、調査対象をプロジェクトメンバーに限定せず、より多くの人を対象にした調査を行い、より多様な視点や思考を反映させ、本 Element の動きの根拠がより説得力のあるものにしていきたいと考えている。

Q. サーボモータの動作音が大きく感じるが、何か対策はしなかったのか。

A. 本 Element の目標は、風を心地よい動きで可視化することである。私たちはユーザに対して、動きに注目し、その心地よさを感じてほしいため、動作音については現段階では考慮しておらず、重要視していない。中間発表会の時点のプロトタイプと比較すると、サーボモータがアクリル板によって覆われているため、静音にはなっている。今後の展望として、静音性に関する改善策を検討していく。

これらのフィードバックを受けて、本 Element の展望、今後の課題についてまとめる。まず課題として、動作音が大きい問題と、評価実験の規模が小さいという問題が挙げられる。動作音が大きい問題については、静音サーボを導入して動作音を小さくしたり、動作時に風の音や風鈴の音を流して聴覚的にも心地よさを提供したりといった改善案が考えられる。これにより、ユーザが Element を利用した際に、動作の不快な音が軽減され、より視覚的な心地よさに集中してもらえたり、聴覚的な心地よさも相まってよりユーザ体験をよくできると考える。また、評価実験の規模が小さいという問題に関しては、プロジェクトメンバーだけでなくもっと対象者を増やして統計的に意味のある規模で評価実験を行いたいと考える。さらに、現在風力センサとサーボモータはそれぞれ別のマイコンで制御しているが、有線接続になっているため、システムの設置場所がやや限定されている。そのため、今後はマイコン同士を無線接続可能にすることで、設置の自由度を高めていきたいと考える。

（※文責: 菅英寛）

4.3 Element.03 「雨守り」

4.3.1 成果発表会に向けて

成果発表に向け、我々は Element「雨守り」の実寸大の制作に取り掛かった。我々はまず、中間発表の段階において雨守りが抱えている課題をまとめた。課題は、自由な角度のモーター制御、実寸大プロダクト制作時の問題、風センサの形の見直し、Arduino やバッテリーの設置場所、手持ち型か背負い型かの決定などが挙げられた。特に、実寸大プロダクト制作時の問題には、雨漏りの重量や、傘の上半分を動かせるだけのモータの確保、重さに耐えられるモータの機構、バッテリーなどの問題があった。

雨守りの制作にあたり、まずは雨守りに適した小さく軽い傘の確保を行った。実際にホームセンターへ行き、傘の選定を行い、検討を行った。その結果、傘の上部が広く、中棒が丈夫で、加工しやすく、軽く、コンパクトな折り畳み傘に決定した。また、雨守りの制御に用いるモータについての思索を行った。傘の上部はある程度の重量があり、既に所持していたサーボモータではトルクが足りず、支えきれないのではないかという懸念があった。そのため、かなりのトルクがあり、角度指定もしやすいステッピングモータを使用する案も挙げられたが、ステッピングモータ自体がかなりの重量があり、雨守りが重くなってしまうため却下した。そして、従来通りサーボモータでの制作を行うことにし、傘の上部を支えられるだけのトルクを持つサーボモータを通販で購入した。次に、風センサについての考案を行った。中間発表で採用した M5StickC の角度センサによる 2 台のサーボモータの制御案を見直した。そして、一つのセンサで水平方向と垂直方向の制御を行うのではなく、それぞれの方向に別のセンサを用いて制御を行うことにした。その後、グループ内

で話し合い、水平方向の新しい風向センサ案を考案した。まず、振動センサを使用した案についての検証を行った。その案とは、傘の内側に振動センサを取り付け、雨が当たる振動の強さの差で雨が降っている方向を検知するという案である。この振動センサを購入し、実際に検証したところ、少しの振動であまりにも大きな値を返してしまった。そのため、この案は不採用となった。次に、フォトリフレクタを使用した案についての検証を行った。その案とは、まず、4 面を風が通るネットで覆ったすり鉢状の箱を用意する。そのすり鉢の端の四隅にフォトリフレクタを設置し、白くて軽いボールを箱の中に入れる。そのボールが風にあおられ、すり鉢の坂を上り、端に来たらフォトリフレクタがそのボールを認識し、風が吹いている方向を検知するという仕組みである。この案の検証では、まず一隅だけのプロトタイプを作成し、軽い緩衝材の玉を使用して検証を行った。この検証では、うまく動いたように見えた。しかし、「この案は風を検知するためには複雑すぎるのではないか」という意見を教員からいただき、我々はこの案を却下してシンプルな案を考えることにした。そして、最終的にはロータリーセンサを使用した案を採用した。ロータリーセンサとは、リング状のセンサであり、輪の中に棒を通して回すと、その角度を検知するセンサである。この案は、風向計の軸にロータリーセンサをはめるというシンプルな構造の案である。ロータリーセンサによって、風向計によって風の向きをシンプルに測ることに成功した。次に、垂直方向の制御に必要なセンサについてグループ内で話し合った。教員からいただいた「人の動きを増強するインタラクティブな傘」というアドバイスをもとに、傘を握る強さで傘の傾きを制御するという案を採用した。そのために、圧力センサを傘の持ち手部分に設置することにした。次に、バッテリについての課題解消に取りかかった。雨守りの開発初期は、PC に USB ケーブルをつなぎ、それを電源として使用していた。しかし、雨守りは傘であるため、自由に持ち運べるような電源にする必要があった。そこで、電源として電池を使用したが、電力不足によってサーボモータが動作不良を起こしてしまうことがあった。そのため、電源をモバイルバッテリーに変更し、動作の安定性を確保した。

グループを外装班とセンサ班に分け、分担して作業を行った。外装班では、傘とアルミパイプの接続を行った。傘の中棒とアルミ棒を接続し、アルミ棒とアルミパイプを 2 つのサーボモータを挟んで接続する設計で制作を行った。傘の中棒の切断や、中棒の空洞に丁度はまるよう、ステンレス棒のやすり掛けなどの加工を行った。また、ステンレスパイプも適切な長さになるように切断を行った。サーボモータの接合にはアクリル板を使用した。アクリル板の加工にはレーザーカッターを使用し、目的の形に切り出した。そのアクリル板とアルミパイプにネジ穴を開けてネジで接合し、アクリル板とサーボモータネジで接合した。また、水平方向の制御を行うサーボモータのジョイント部分にもネジ穴を開けたアクリル板を接合し、そこに垂直方向の制御を行うサーボモータをネジで接合した。垂直方向のジョイント部分は、ネジによってアルミ棒との接合を行った。しかし、耐久性に不安があるという理由から、のちにアルミ棒を銅製の棒に変更した。さらに、マイコンやバッテリを格納し、雨守りをコードレスで持ち運ぶためのプラスチックケースを作成した。このケースは、雨守りの持ち手部分の下に設置した。センサ班では、センサとサーボモータの動作のプログラミングと、センサパーツの作成を行った。動作プログラミングでは、風向センサに取り付けられているロータリーセンサから送信される角度データを用いて、水平方向の駆動をするサーボモータを制御するプログラムを書いた。また、グリップ部分に設置されている圧力センサから送信される圧力データを用いて、垂直方向の駆動をするサーボモータを制御するプログラムを書いた。特に、サーボモータを制御するプログラムは、雨守りの回転速度や傾き速度を制御する部分であるため、重要である。速い速度で動作をすると、使用者が雨守りに引っ張られてしまう。しかし、遅い速度で動作をすると、使用者が雨に濡れてしまう。雨守りの動作が丁度良い速度になるように、グループ内で検討を重ねた。次に、センサパーツの作成を行った。センサパーツは、風の方向を検

知するために使用する風向計と、圧力センサを押しやすくするためのグリップ、圧力センサに使用したシリコン製のボタンがある。まず、風向計は 3D プリンタを使用して作成した。その際、3D モデルは Autodesk Fusion 360 という 3D モデリングソフトを使用して作成した。風向計の棒部分がロータリーセンサに丁度通るような調整を行ったが、ロータリーセンサの小さな穴に通るような細い突起は 3D プリンターで印刷することができなかった。そのため、丁度通るようにやすりで削ったネジを、ロータリーセンサに通す細い突起に使用した。こうしてできた風向センサは、雨守りの下部に設置された。次に、グリップを 3D プリンタで作成した。グリップは、アルミパイプにハマるようなサイズで印刷した。グリップには、アルミパイプに張り付けている圧力センサを押すための穴を開けている。穴にはシリコン製のボタンを装着し、圧力センサを押しやすくした。このシリコン製のボタンは、初期に考えていた「プチプチを利用した Element」の部品を流用したものである。グリップは雨守りを持ちやすくするほか、ボタンの位置によって雨守りの前後の向きを明確にする。

成果発表に向けて、実寸大の雨守りや発表資料、ポスター、Web サイト、雨守りを使用したプロモーション動画を制作した。発表資料作成には Figma を使用した。資料には、雨守りの写真や使用している様子の gif 画像を使用した。内容は、雨守りのコンセプトや、それに至るまでの考案プロセス、雨守りの使い方や機構についてを説明した。また、インフォグラフィックスには、Blender で作成した 3D モデルの画像や、モータが動いている様子の gif 画像を使用した。インフォグラフィックスでは、雨守り全体の機構や、風向センサやスイッチの圧力センサから Arduino での計算処理、そしてサーボモータの制御までの流れを表現した。ポスターはインフォグラフィックスやロゴを、ポスター制作担当に渡して制作してもらった。Web サイトは、インフォグラフィックスやロゴ、雨守りのコンセプト文章、撮影した雨守りの写真、雨守り制作作業中の写真を、Web サイト制作担当に渡し、我々のグループの紹介ページを作成してもらった。雨守りを使用したプロモーション動画は、制作した雨漏りを実際に使用している様子を撮影し、動画制作担当へ渡して制作してもらった。雨守りが回転したり傾いたりしている様子と、センサ部分にクローズアップした映像をワイプにした動画を制作した。

（※文責: 橘孝則）

4.3.2　成果発表会でのフィードバックを受けて

成果発表会を経て、雨守りに頂いたフィードバックは「さらにコンパクトにしてほしい」「雨の方向に動いてくれる傘は画期的である」「雨守りというネーミングが雨漏りしそうで面白い」「デモで正しく動作していなかったのがもったいない」「傘の柄が曲がるのが新しい。精度が上がれば実際に使えそう」「さらに改良が加われば、手で持たなくても自動で雨から守れる完全防御傘が実現できそう」「デザインや小型化、軽量化ができるとより良い」「完成したのは素晴らしいと思う。しかし、中間発表で軽量化や防水、傘が傾いた際の注意点などを期待していたが、その部分が変わらなかった」などがあった。

雨守りのコンパクト化については、傘を短く折りたたむ設計にできなかったため指摘されたと考える。雨守りの構造上、中棒の中間点に回転や傾けるためのモータが必要になるため、コンパクトな設計にするには新たな工夫が必要になるだろう。軽量化については、グループ内でも課題として挙がっていた。アルミパイプや銅棒などの傘本体の重さに加え、マイコンや電池、モバイルバッテリーなどの重さも加わり、全体的に重くなってしまった。傘として使うには、片手で軽く持てるぐ

らいの重量である必要がある。しかし、雨守りは片手で持てはするものの、長時間持つには適さない重量になってしまった。パーツの素材やバッテリの重さなどを見直すことで、軽量化できると考える。防水についても、グループ内で課題として挙げられていた。特にサーボモータ部分がむき出しになってしまっているため、駆動部分をラバーなどで覆うことができれば、改善できると思われる。発表会のデモでは、度々動作不良を起こしてしまった。最初のうちのデモでは問題なく動いていたが、デモを重ねるうちに途中で動作しなくなってしまった。動作不良の理由は、マイコンケースの中の配線が抜けてしまったためであり、配線の繋がりを補強をすることで改善が可能であると思われる。傘の傾きによって雨が滴る問題については、いくつか改善案が挙げられていたが、どの案も課題があった。そのため、この問題を解決するためには、傘の形状や動作など、新たな工夫が必要になると思われる。しかし、雨守りについての好評も多かった。特に、実際にセンサに対応して傘が回転したり、傾いたりしていたことが好評であった。この評価を見れば、プロダクト制作は十分成功したといえるだろう。

（※文責: 橘孝則）

4.4 Web サイト・ポスター・動画制作

4.4.1 Web サイト制作

最終報告会では、Interaction Elements2023 のプロダクトなどを紹介する Web サイトを作成した。有志でメンバーを集め、「デザイン班」と「コード班」に分かれ作業を行った。

（※文責: 板垣智也）

Web デザイン

Web サイトのデザインは、我々が IE のイメージとして持っていた「シンプル」を意識して作成した。発表スライドやポスター、議事録などから掲載する資料を選別し、少なすぎず多すぎない情報量を目指した。進捗や変更内容の共有を円滑に行うため、デザインは Figma 上で行った。Web サイト全体のフォントは、Windows・macOS 両方で同じものを使用したいため、Note Sans JP を採用した。また色もグレースケールのみを使用し、発表スライドやポスターに揃えた。作成したページは、メインページ・IE についてのページ・プロダクト紹介ページ (3 グループ分) の 5 ページである。メインページでは、IE のロゴ、IE について、各プロダクトのビジュアル (画像とロゴ)、メンバー・先生紹介を掲載した。IE についてのページでは、IE の説明、IE2023 のロゴの説明を掲載した。プロダクト紹介ページでは、各グループのプロダクトロゴ、紹介動画、コンセプト、機構、制作プロセス、活動中の写真を掲載した。紹介動画は成果報告会で作成したもの、機構はポスターに掲載したインフォグラフィック、制作プロセスはスライドで使用したもの、活動写真は議事録係が撮影していた活動写真から抜粋して使用した。各ページにはヘッダーを置くことでどのページにも遷移しやすくした。

（※文責: 板垣智也）

Web コーディング

コーディングは HTML と CSS を用いて行った。複数人でコーディングを行うため、バージョン管理システムの GitHub を利用することにした。しかし、使用したことがあるメンバーが一人のみであったため、そのメンバーを講師とした GitHub 勉強会を行った。その後、各自 GitHub を利用してオンラインで作業した。Figma で決めたデザインを参考に、フォントや文字サイズ、写真サイズなどを指定していった。また、画面サイズに応じて配置を変えるレスポンシブ化を施し、スマートフォンでも見やすいよう工夫した。スマートフォン用デザインは Figma では想定していなかったが、横 3 列に並べていたものを 1 列にすることで対応させた。各グループのロゴや動画、インフォグラフィックなどは未完成のものがあったため、仮の画像を入れるなどして対応し、完成次第入れ替えていった。複数人でコーディングしたため、CSS がメンバーにより多少ズレがあった。そのため、メインページの CSS に合わせて修正した。製作期間はおよそ 1 週間を要した。

（※文責: 板垣智也）

サーバー

公開は GitHub を使用していたこともあり、GitHub Pages を使用した。個人のアカウントで製作していたため、新たに「interactionelements2023」という Organization を作成し、このアカウントで公開した。

（※文責: 板垣智也）

4.4.2　ポスター制作

ポスターは Adobe Illustrator で制作した。最終発表のポスターは、中間発表のポスターをベースに制作を進めた。中間と同じく、タイトルや文章のフォントを統一することで、デザインのコンセプトをポスター内で統一した。また、文章は全て日本語と英語のバイリンガルとし、多くの人に伝わるポスターを心がけた。また、Interaction Elements のロゴを、タイトルの左隣に配置したりポスターの左上にあしらうことで強調させ、プロジェクトのイメージ付けがされるように意識した。中間発表のポスターの本文のフォントは、和文フォントを游ゴシック体、欧文フォントを Futura として合成フォントを作り使用していたが、最終発表のポスターでは和文フォントを游ゴシック体、欧文フォントを Helvetica Neue とすることで、和欧混色した際に自然で統一感のある印象を与えられるよう改善した。最終発表のポスターでも、中間発表のポスターと同様にグリッドレイアウトを意識し、中間発表ではばらつきがあった各 Element の紹介部分の文量をできる限り揃えることで、より整理され見やすいポスターを目指した。中間発表のポスターでは、文字と背景のコントラストが強いと感じたため、グレーのトーンを落としコントラストを調整することで、読みやすさを向上した。ポスターに使用したプロダクトの写真では、ライティングや撮影状況を全体で統一し、背景の明るさを揃えることで、ポスター全体にまとまりを出した。

（※文責: 大塚創生）

図 4.1　成果発表会のポスター

4.4.3　動画制作

撮影

動画の制作にあたり、各 Element のコンセプト動画を制作するため、撮影を行った。撮影場所はグループごとに異なる。撮影機材は以下のとおりである。

- カメラ：CANON EOS M5
- レンズ：SIGMA 30mm F1.4 DC HSM

これらの機材を使用して各 Element を撮影した。プロジェクトの白を基調としたイメージに合わせ撮影を行った。撮影時には、 Element を綺麗に魅せるために動画に映りこむ影をできる限り無くし Element の配置やライトの照らし方を試行錯誤した。コンセプト動画は Element を実際に使用しているシーンや搭載されている機能の動作が伝わるような動画を目的として制作を行った。また、撮影する前に動画の構成を考え、必要な素材を検討してから撮影を行ったために、動画編集の際に必要な素材を効率的に手に入れることができた。

（※文責: 樫木海生）

編集

上記の方法で撮影した動画素材を用いてグループごとの Element のコンセプト動画を制作した。今回の制作では編集ソフトとして各 Element のロゴのアニメーションを制作するために Adobe After Effects、動画素材の切り抜き及びエフェクト付加をするために Adobe Premiere Pro を用いた。

（※文責: 樫木海生）

ロゴアニメーション編集

まず初めに、ロゴのアニメーションの制作を行った。このアニメーションを制作する上でロゴをただ単調に動かすのではなく、各 Element の動きを抽象的に表現するようなアニメーションを制作することを目標として制作を進めた。また、ロゴアニメーションの時間については視覚的に負荷がかからない程度である時間とした。制作に用いたエフェクトは Adobe After Effects に機能として備わっているもので、主にブラシアニメーション、トランスフォームでの変形・回転である。それぞれの Element に合わせて用いるエフェクトは変更した。

まず、GroupA の Copypen のロゴアニメーションには、パスのトリミングとイージーイーズを用いた。パスのトリミングというエフェクトは、直線で書かれた素材を端から表示していくアニメーションで、これらのエフェクトを用いることで、Copypen の書き込みのイメージを表現すると共に、イージーイーズにより単調な印象を消すことが出来た。

つぎに、 GroupB の BLWIND のロゴアニメーションにはブラシアニメーションを用いた。このエフェクトを用いると、ロゴが端から徐々にブラシで描いたかのような軌跡で現れるようなアニメーションを生成できる。これにより、 GroupB の Element の雰囲気に合う流れる風を視覚化したようなアニメーションを制作した。また、こちらもイージーイーズを用いることで、直線的に見えていたアニメーションが緩急のある動きに変化し、よりイメージに合った仕上がりとなった。

最後に、GroupC の雨守りのロゴアニメーションには、図形の縦横比と角度の変更、イージーイーズを使用した。傘に見立てたパーツを縦横比の変更で動かすことで、まるで傘が開いたかのように表現した。ただ横方向の縦横比を 0% から 100% に変更しただけでは、傘が開いているようには見えなかったため、縦方向の縦横比も 105% から 100% に変更することでよりリアルなアニメーションに仕上がった。傘の軸部分のアニメーションは、角度を変更することで雨守りの傘が自動で折り曲がるような表現をした。また、これらの制御にイージーイーズを用いることで、動き始めは素早く、終わりはゆっくりと見えるようになった。これにより、動きが単調で無くなり、よりダイナミックな表現が可能になった。

ロゴの背景には各 Element の写真を表示し、その映像をはじめと終わりに表示した。この際、はじめに表示する画像は Element の全貌が見える写真、最後に表示する画像は特徴が見えやすい画像にすることで、単調さを無くし、見ている人に飽きさせない編集を行った。また、画像の上に直接ロゴを表示すると視認性が悪かったため、画像とロゴの間に不透明度を 60% に調整した白色の図形を表示することで、見やすくなるよう解決した。ロゴのアニメーションははじめのみ表示し、終わりには表示し無かった。これは、アニメーションを 2 回表示したバージョンの動画と比較して見たところ、はじめのみ表示した方が見やすかったためである。

（※文責: 冨田夏央）

動画編集

各 Element の動作を確認することのできる動画を編集した。 各グループの動画の長さはおおよそで統一した。Element の動きが良く表現できているシーンを多数抽出しカットして動画の素材とした。GroupA が制作した Element はペンであった。ペンの見た目を接写で映すシーン、実際に使用した時の手元のシーン、搭載されている機能を紹介するシーンを主に使用した。GroupB が制作した Element はブラインドであった。正面から映し、ブラインドのスラットの 2 パターンの動きを紹介するシーン、実際に使用しているイメージのシーンを主に使用した。GroupC が制作した Element は傘であった。傘全体を映すシーン、2 つの機能を紹介するシーンを主に使用した。カットしたシーンとシーンの繋ぎにはディゾルブ効果を用いることで繋ぎ目に不自然がない多視点からの映像を制作した。1 つのシーンの時間が長すぎる時には早送りをして細かな秒数調整をした。カットではなく早送りという手法を用いたのは、カット編集を施すとシーンから次のシーンに映る際に突発的になってしまい、繋ぎ目が不自然になってしまうためである。そのため、早送りでの編集にすることで 1 シーンの最初から最後までを見せることができ、自然な流れとなる。また、GroupC の動画のシーンでは Element の形上、手元と動作部分を一度に映像に収めることが難しかったため、ワイプとして手元の動きを切り出してそれに応じた可動部分の動画と組み合わせることで動作が視覚的に上手く伝わるような映像とする工夫を施した。

図 4.2　実際の動画の一部

次に、計 3 つの音楽を選定して映像に当てはめた。使用した音楽は各 Element のイメージに合わせたものにした。また、Element の紹介を効果的に行うため、落ち着いた BGM を選定した。音楽の長さについても各グループで長さを統一するため、ある程度の長さがあり、途中でカットをしても違和感のないような BGM を選んだ。

次に、編集した動画の冒頭に別で制作したロゴアニメーションの映像を差し込んだ。このようにすることで、どの Element の紹介動画かを動画冒頭で伝えることが可能である。また、動画の最後にも背景として制作物、前面にロゴを表示する映像を差し込んだ。映像全体として制作物とロゴのシーンから始まり、同じように制作物とロゴのシーンで終えることで、プロジェクト全体として統一感のある各グループの動画とした。

最後に、それぞれ編集した全ての動画を一つの動画としてまとめた。グループの動画とグループの動画の繋ぎにはホワイトアウトを用いたフェードのエフェクトを付けた。プロジェクトの白を基調としたイメージに合っていて、シンプルにまとまる白色での効果を選んだ。全体動画の長さは最終的に 2 分 38 秒になった。全体動画をプロジェクトメンバーで確認し、フォントの統一や、イメージの不一致などの問題を解決させ、動画を完成とした。

（※文責: 樫木海生）

4.5　最終発表方法

4.5.1　プレゼンテーション用スライド資料

中間発表時同様、デザイン統括担当のメンバーからフォントの種類や、大きさ、色調などのデザインルールを共有してもらい、それに沿って作成した。中間発表時に Google スライドを利用して作成したが、他のグループのスライド制作の進捗が見にくかったり、使用できるフォントの幅が狭く、より効率的に編集しやすい環境を構築するために Figma を利用して行うことで、中間発表時の問題点の改善を行った。また、中間発表時より、プレゼンテーションの構成を変更し、制作プロセスの紹介を全体発表ではなく各グループの発表の際に紹介するといった構成にした。

スライドの構成

もくじ

約 10 分間のプレゼンテーションのため、全体の概要を示すもくじを作成した。中間発表と全体的な構成を練り直したため、それに伴ったページ構成とした。

Interaction Elements とは

中間発表時に作成した Interaction Elements とは何かを伝えるページを作成した。中間発表では私たちのプロジェクトの目的を口頭で紹介していたが、最終発表ではスライドに文章として掲載することで、より説明として伝わりやすくなった。

制作プロセス -図鑑の作成-

中間発表では、図鑑の要素の写真だけをスライドに掲載していたが、それに加えて図鑑の全要素を収めた引きの写真を掲載することで、視覚的にどのくらいの数が集まったのか理解しやすくなった。

制作プロセス -KJ 法によるマッピング-

作成した図鑑から、KJ 法によるマッピングを行って、アイデア出ししたことを紹介するページを作成した。「五感を軸に KJ 法を用いることは厳密には誤っている」という教授の助言のもと、スライドには「特徴をもとにグルーピングを行った」という表現に切り替えた。

グループによる Element の簡略化した紹介

中間発表同様、全グループで構成を統一し、Element の概要→コンセプト→機構という構成で行った。中間発表時では、各グループが 4 分程度の紹介とし説明を行ったが、成果発表会では、各グループ 6 分程度の紹介とすることで、グループ別に分かれた後に、詳細な説明や、質問をしてもらえる時間を設けられるようにした。

個別スライドのグループによる Element の紹介

全グループで構成を統一し、Element の概要→コンセプト→デモという構成で行った。各グループが中間報告時までに完成している Element の写真や資料を組み込むことで、これからの展望を

見せる Element の紹介とした。

（※文責: 工藤翔太）

4.5.2　プレゼンテーション方法

プロジェクト学習の成果発表会は、中間発表会同様にオープンスペースで行われることから、他のプロジェクトに発表内容をかき消されることや、ぼやけてしまうことを防ぐために自分達らしい発表会を意識したプレゼンテーションを心がけた。また、中間発表会を行った場所より狭いスペースでプレゼンテーションを行ったため、他のグループに声が干渉しないよう注意しながら発表を行った。

（※文責: 工藤翔太）

4.5.3　成果発表会の全体フィードバック

各グループの成果発表会でのフィードバックは前述しているため、4.5.3 では全体でのフィードバックを受けた反省点・改善点・評価を記述する。

発表技術について

「非常に聞きやすい発表でした」「流れが綺麗で分かりやすい発表でした」「しっかり身振り手振りを使い、具体的な例を出して説明してくださったので分かりやすかった」「全体説明において、短時間でプロジェクトの概要全体を説明できていて良いと思いました」などのアンケート結果を全体としてもらうことができた。これによって課題としていた、自分達らしい発表会を意識したプレゼンテーションを行うことができていたと考えられる。特に「理解しやすかった」「聞き取りやすかった」「プロジェクト活動について理解できた」というフィードバックの抜粋から、伝わりやすいプレゼンテーションができていたことが考えられる。数値的にも、全体評価が約 8.65 と高い評価をもらうことができた。

発表内容について

「制作プロセスがしっかりしていたのが印象的でした」「目的・成果共にわかりやすく面白いものでした」「作った理由が順当なものであった」「社会的に役立つ研究になると思った」「技術と発想をしっかり融合していた」など、発表内容についても技術を除いた部分で十分な評価を得られていると考えられるアンケート結果をもらうことができた。これは、全体の発表を各グループの説明は大まかに、詳細な説明については各グループのブースに移動してもらうという形式をとったことで、必要な説明を選出することができたと言える。対照に、目的の完成度に達成できなかった Element があり、それについて不十分に感じられたというようなアンケート結果もいくつかあり、全体の発表としては十分であると感じられたが、各グループの結果としては、もう少しエレメントの動作の精度を上げることができた部分が存在するのではないかと考えられる。

（※文責: 工藤翔太）

第 5 章　おわりに

5.1　グループのまとめ

5.1.1　Element.01 「Copypen」

本グループは触覚に関する Element 制作を希望するメンバーで結成され、数多くの案出しを行った。初期は、でこぼこな地面でも問題なく書ける「触覚キャンセリングペン」というコンセプトを採用しており、開発に着手していた。しかし、「触覚キャンセリングペン」の実現がかなり難しいことが判明し、コンセプトを変えなければいけない状態に陥った。再び案出しを行うこととしたが、すでにペンに関する調査は行っていたので、インタラクションのあるペンの案を出すこととした。案出しは 2 回目だったこともあり、時間をかけすぎずにこすり出しとコピー＆ペーストを足し合わせた「Copypen」を考案することができた。

前期は簡単なプロトタイプを作成することを目標に活動した。まず必要となる機構を洗い出し、大きく「読み取り」「書き取り」「位置情報」の 3 つに分けた。中間発表までに「読み取り」の機能を作ることは難しいと考え、まずは「書き取り」「位置情報」の二つを機能として持つプロトタイプ作成を始めた。「書き取り」機構では、サーボモータを用いたラック＆ピニオンを採用し、ラックにペンを接着して芯を出す予定だった。しかし、手作業で製作していたために安定性がなかった。そこでサーボモータに直接ペンを付けることで、安定性の問題を解決した。「位置情報」の機構は、超音波センサを使用して壁との距離を測ることで位置をとる方式を使用した。専用の壁付きの台が必要となるものの安定した位置が計測できるため採用した。また、超音波センサとサーボモータを制御するためのマイコンとして、できるだけ小さいプロトタイプにしたいため小型の M5StickC を使用した。プログラムは超音波センサの距離に合わせて、サーボを芯が少しだけ出るように回すというものであった。

後期は、完成に向けて「読み取り」機構の作成と中間発表までのプロトタイプの改善を主に行っていった。「読み取り」機構は M5StickC に接続が容易な Grove 端子を持つカラーセンサを使用していた。しかし、接地面が大きいなどの問題からフォトリフレクタにセンサを変更した。フォトリフレクタはブレッドボードを必要としていたため、サイズが大きくなるとして使用を断念していたが、空中配線やユニバーサル基板を駆使してブレッドボードを使わずにマイコンと接続を可能にした。プロトタイプの改善として、M5StickC から M5Stack Gray への変更、超音波センサからマウスへの変更、ラック＆ピニオン機構の再検討などが挙げられる。これらの改善により、機構の拡充や精度・安定性の向上につながった。また、レーザーカッターや 3D プリンタを使うことで、セロハンテープで組み立てていたものが、ネジで組み立てられるようになり見栄えもかなり良くなった。プログラム面も大幅に変更が加わった。フォトリフレクタからのデータ処理、マウスでの位置の取得、イラストデータの保存機能などの追加を行った。特に、イラスト全てを人の手で読み取ることは難しいため、取得できていないデータを周りのデータから推測して埋める補正には力を入れた。

成果報告会では、「た」や「え」などのひらがなを 1 辺 10mm のドット絵で用意し、実際にそれをコピー＆ペーストするデモを見せた。また、発表を聞きに来てくださった方にも、体験してもらった。普段通りの精度は出せなかったものの、Copypen のコンセプトなどを伝えることはでき

た思う。

「Copypen」制作の反省点として、ブラッシュアップが出来なかったことが挙げられる。センサや機構、プログラムがすべて使える状態に揃ったのが、成果報告会の 2 週間ほど前で精度の向上や使用感の改善などに時間を使うことができなかった。また、目標に「こすり出しの感触の気持ちよさを体験してもらう」とあったが、こすり出しのガタガタ感を Copypen では再現することができなかった。これらは外装班とプログラム班での情報共有が上手くいかなかったことや、センサや機構の決定が遅れたことによる時間の浪費が問題であった。メンバー 6 人での情報共有手段や優先度の決め方などを、最初から定めておくべきだったと考える。しかし時間が足りないながらも、目標にしていた動作ができるよう制作でき、制作過程も満足のいくものであったと考えている。

「Copypen」制作は、複数人でのプロジェクトの進め方やモノづくりの方法などの技術面もしかり、とりあえずやってみることやメンバーを信頼するなどの精神面もしかり、学ぶことがとても多かった。「Copypen」を先生方に提案した際、「大変そう」「道のりは長い」との評価をいただき、実際何から始めたらよいか分からない状態でのスタートだった。そこから、先生や工房職員の方の力を借りながら知識や技術を身に着け、成果報告会では納得のいく Element を制作することができた。紆余曲折を含めた「Copypen」制作の経験は、これからの私たちに多大な影響を与えるものだろう。この経験を今後の活動に生かしていきたい。

（※文責: 板垣智也）

5.1.2 Element.02「BLWIND」

本グループでは、アイデアのヒントやインスピレーションを得るために行ったフィールドワークの中で、風の心地よさと連続する動きの心地よさに着目し、風を心地よい動きで可視化する Element の制作を目標に活動してきた。本 Element は「BLWIND」と名付けられ、風を視覚的に感じるという新たな体験の創出と、屋内外の境界線を溶かし新たな開放感を与えるという提供価値の創出を目的とした。中間発表会までは、スチレンボードを用いたプロトタイプを作成し、10 枚のスラットの動きによって波のような動きを表現できることを確認した。中間発表会後は、Fusion360 を用いて詳細な設計図を作成し、プロトタイプを作成した。スラットの一連の動きは、Blender を用いて作成した 3DCG シミュレーション映像を用いて順位評価および評価グリッド法による質的分析を行い、決定した。決定した動きを反映するプログラムは、millis() 関数を利用して短い周期でサーボモータの制御を繰り返すことで、複数のスラットが同時並行に独立して動くように見せる擬似マルチタスキングを実装した。また、プロモーション映像や発表用スライドなどを作成し、成果発表会でこれまで私たちが行なってきた活動が少しでも伝えられるよう努力した。成果発表会のフィードバックでは、実際に窓がない部屋に置いてみたいというコメントもあり、本 Element の目的である風を視覚的に感じるという新たな体験の創出と、屋内外の境界線を溶かし新たな開放感を与えるという体験価値の提供は概ね達成できていると考える。プロジェクト遂行にあたっては、プロジェクトマネージャーを中心に、グループメンバーそれぞれがタスクを積極的かつ協働的に進めていたため、プロジェクトマネジメント面でも満足なものになったと感じる。今回この Element の制作に必要だった技術的な知識に加え、プロジェクト遂行の仕方であったり、自分たちの活動の内容を第三者に伝える方法であったりなど、このプロジェクトで学んだことは今後のグループメンバーそれぞれの活動において大いに役立つ有意義なものになるだろう。

（※文責: 菅英寛）

5.1.3 Element.03「雨守り」

本グループは、最初は音に関する Element を作成したいメンバーで集まって構成された。しかし、グループ内で様々な案を考えては、廃案にすることを繰り返し、前期の半ばまでは制作プロダクトが決まっていなかった。その背景から、音に絞らず、もっと自由に広く案を考えることに方針を変更した。また、プロジェクト全体で制作したついついやってしまうこと図鑑や、それに基づいた KJ 法など、初めに立ち返るということを行った。そうして「雨の方向に自動的に向く傘」という案に注目し、「雨守り」の制作に取り掛かった。

中間発表会まであまり時間が無かったが、グループメンバーを分担し、効率的に作業を行った。その結果、雨守りの駆動部分をほとんど実現するほどのプロトタイプを完成させることができた。また、3D モデルによるシミュレーション動画によって、想定している雨守りの実寸大を映像で再現することができた。これらによって、中間発表会では、雨守りの動作の仕組みをプロトタイプの実演で説明することができ、シミュレーション動画によって我々が想定している雨守りがどのようなイメージなのかを伝えることができた。中間発表会のフィードバックから、背負い型が邪魔になってしまう問題や、雨守りが傾いた際の水滴問題など、様々な課題が浮き彫りになった。

成果発表会までは、雨守りの設計から制作、動作プログラミングなどを行った。まず、雨守りに使う傘のために、ホームセンターへ足を運び、傘などの選定を行った。そして、軽く加工しやすい折り畳み傘を調達した。また、M5StickC を使った風センサは廃案としたため、新たなセンサ案についての考案も行った。いくつか出た案の中から、水平方向の動作には風向計とロータリーセンサを用いた風向センサ、垂直方向の動作には押す強さを検知する圧力センサを用いる案を採用した。バッテリについても改善を行った。最初は電池を利用したが、電力が足りず動作不良を起こしてしまった。そこで、モバイルバッテリーに変更することで改善された。雨守りの制作を行うにあたり、加工班とセンサ班に分かれた。加工班では、折り畳み傘の加工や、アルミパイプの加工、アクリル板を使用したサーボモータ連結の設計や制作など、多くの加工を行った。その結果、実寸大の雨守りを実現した。これにより、完成度の高い実物が伴う成果物を作ることができた。また、センサ班では、センサのデータ制御やサーボモータの動作制御、配線などの電子工作を行った。その結果、実寸大の雨守りがセンサに対応して回転したり、傾いたりする動作を実現した。これにより、成果発表会の雨守りの実演では、使用者に面白い体験をさせることができた。成果発表会を通して、重量や防水の問題など、雨守りにはまだ残っている課題がある。しかし、雨守りという成果物を作るまでには、様々な思索、制作を行い、実物が出来上がっていく過程は良い経験となった。また、発表会では、使用者に面白い体験や感動を提供することができた。その経験は、グループメンバー一人一人の糧となり、今後においても役立つだろう。

（※文責: 橘孝則）

5.2 プロジェクトのまとめ

本プロジェクトでは、最初に全体でアイデア出しを行った後、それぞれが興味を持った分野ごとに 3 つのグループに分かれ、各グループで制作を行った。その結果、最終成果物として、こすり出しの気持ちよさに着目し、ペン一つで読み取りから書き出しまでを行うことができる「Copypen」、風の心地よさに着目して、風の流れを気持ちよく可視化表現する「BLWIND」、雨の音から発想を得て、風に反応して風で曲がった雨を的確に防ぐ「雨守り」の 3 つを最終成果物として提出を

行った。

制作物の違いからグループに分かれた作業をメインとして行ったが、共通する機構についての意見交換や部品の情報交換はグループ問わずに積極的に行い、意見交換の場として設けた毎週末の進捗報告会やリハーサルを通して、グループ内外問わず、プロジェクト全体で一体感を持って活動できた。

当初思い描いていた内容は、部品の都合や設計上の問題などで実現が困難であったり、度重なる仕様変更や方針転換など、様々な問題に直面し、グループごとに達成度に差異が出たが、最終的にはプロジェクトの目的である「未来を形作る部品」の制作を達成し、今までにない Interaction Elements を創出することができたと考える。

（※文責: 工藤翔太）

付録 A　中間発表で使用した発表スライド

中間発表で制作した発表スライドは以下の URL を参照。スライドの一部を図 A.1 で示した 。https://docs.google.com/presentation/d/1sTQL4BAhvCYGKbWDdXblMaeDlpN2A12N/edit?usp=drive_link\&ouid=115850785478201232549\&rtpof=true\&sd=true

図 A.1　中間発表スライド一部

付録 B　成果報告会で使用した発表スライド

成果発表会で制作した発表スライドは以下の URL を参照。スライドの一部を図 B.1 で示した。https://drive.google.com/file/d/1wWlWu_yQX1MCnxlToxvmpzzbhf7GPjPf/view?usp=drive_link

図 B.1　成果発表会スライド一部

付録 C　成果報告会に向けて作成した Web サイト

成果報告会で制作した Web サイトを掲載する。Web サイト全体は以下の URL を参照。
https://interactionelements2023.github.io/InteractionElements.guthub.io/

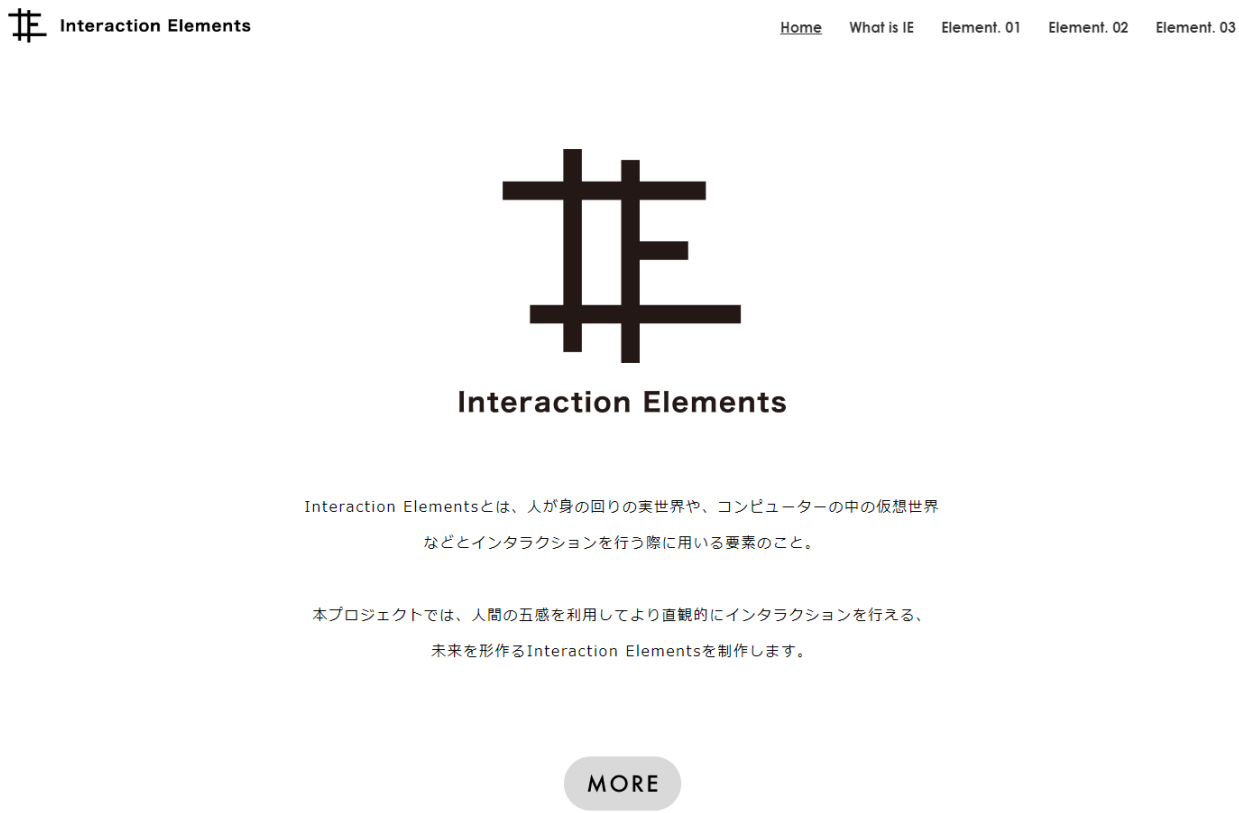

図 C.1　作成した Web の一部

参考文献

[1.2-1]

[1] Shogo Fukushima: 笑い増幅器 2008-Flatters : Laugh enhancement system, Shogo Fukushima. 2022 年 1 月 25 日. http://shogofukushima.com/?p=10 (アクセス日: 2023/07/14)

[2] 齋藤 星輝, 塚田 浩二. 足音から歩行をデザインする靴の提案. インタラクション 2021 論文集, インタラクティブ発表（プレミアム発表）, 2B01, pp.345-349, 2021-03.

[3] 渡邊 淳司, 安藤 英由樹, 朝原 佳昭, 杉本 麻樹, 前田 太郎: 靴型インタフェースによる歩行ナビゲーションシステムの研究. 情報処理学会論文誌. 2005, Vol46, No5, p.1354-1362.https://ipsj.ixsq.nii.ac.jp/ej/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=10643&item_no=1&page_id=13&block_id=8

[4] 山岡 潤一, 筧 康明: “dePENd: ボールペンの強磁性を利用した手描き補助システム”, 情報処理学会論文誌, 55 巻 4 号, pp.1237–1245(2014.4).

[5] 中川 久倫, 伊藤 弘大, 藤田 和之, 岸 楓馬, 福島 力也, 伊藤 雄一: エクスカキバー：ビジュアル・サウンドエフェクトを用いた筆記支援, 情報処理学会インタラクション. pp.641-644, http://www.interaction-ipsj.org/proceedings/2022/data/pdf/5D03.pdf

[6] 宮下 芳明, 小坂 崇之, 服部 進実: 没入型三次元風覚ディスプレイのためのコンテンツ開発 (「アート&エンタテインメント」特集). 2007, Vol12, No3, p.315-321,https://www.jstage.jst.go.jp/article/tvrsj/12/3/12_KJ00007499111/_article/-char/ja/