２０１３年度（平成２５年度）

ECPⅢ

Android 観光望遠鏡の開発

～レンズと Android を組み合わせ、

次世代に向けて何ができるか～

Final Report

チーフアドバイザー：金丸　隆志 准教授

サブアドバイザー：新井　敏夫 教授

チームメンバー

G1-10007　今井一智

G1-10014　小野将弘

G1-10020　紀伊皓介

G1-10050　濵内　瞬

G1-10075　吉野晃平

G1-09030　河端和也

目次

第 1 章　　諸言　（小野担当）

1.1　　研究背景　（小野担当）

1.1.1　　研究テーマ

私たちは、K 社提供の “レンズと Android を組み合わせ、次世代に向けて何ができるか” というテーマに取り組む。この問いに対し、私たちは、レンズと Android の組み合わせにより、レンズの需要を増やすような新製品を創り出すことを目指すことにした。

1.1.2　　Android について

本研究にはレンズと Android という大きな 2 つの柱があるが、ここではその 1 つの Android について説明する。Android とは多くのスマートフォンなどに搭載されている OS(Operating System)であり、パソコンでいえば Windows がそれに相当する。

1.1.3　　スマートフォンの普及率

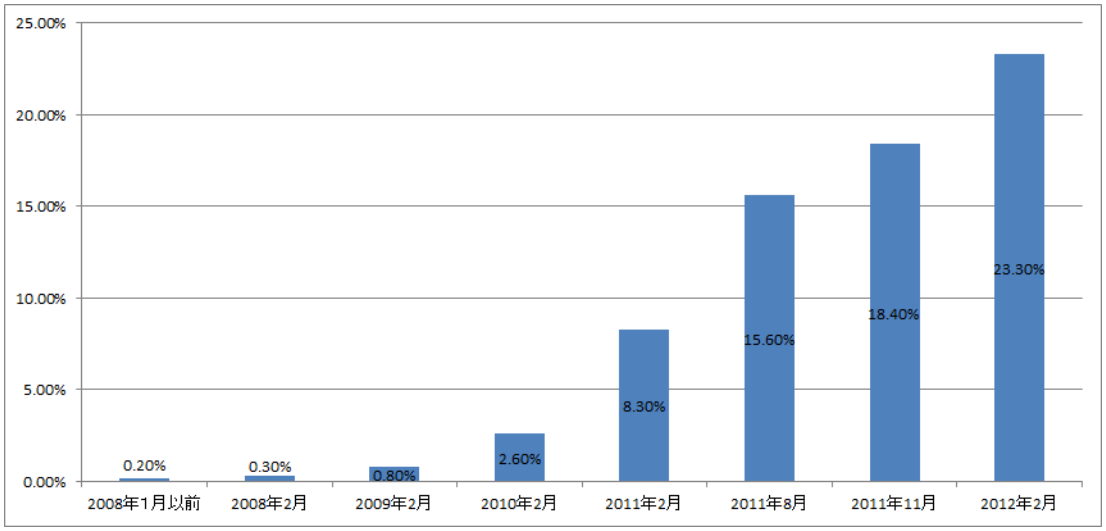


図 1-1 スマートフォンの普及率[1]

図 1-1 は、全国 15～69 歳の一般生活者を対象に行った日本国内におけるスマートフォンの普及率の推移の調査結果を表したグラフであり、2008 年 1 月以前は 0.2%だった普及率が、2012 年 2 月には 23.3%に急増している。また、2013 年においては、国内普及率が 28.2%に達したという調査結果もある。

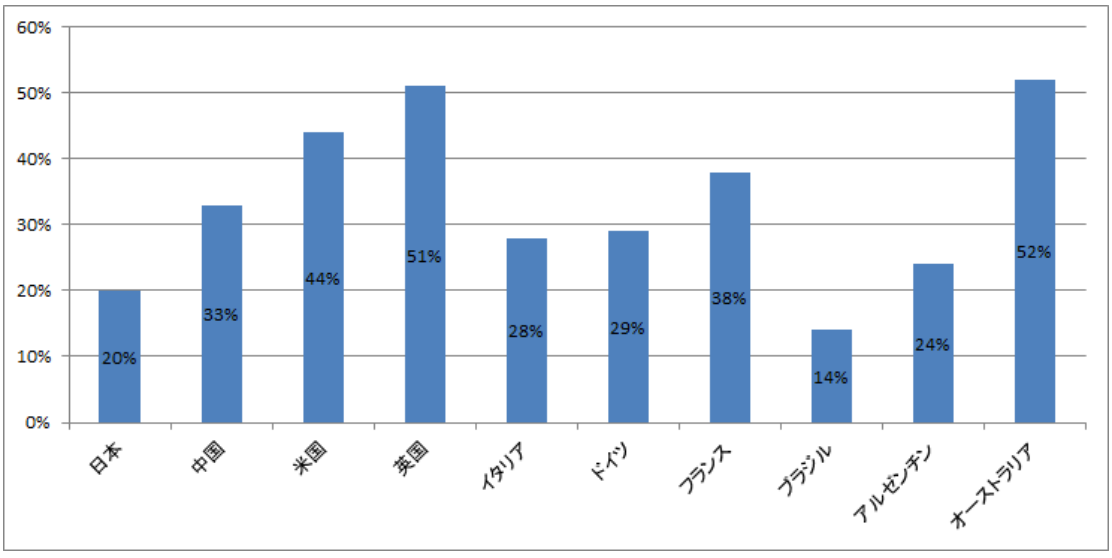


図 1-2 2012 年世界各国のスマートフォン普及率[2]

さらに世界へ目を向けてみるとスマートフォンは世界的に広く普及しているのがわかる。図 1-2 は世界各国の 16~64 歳のオンラインユーザー1000 人を対象に行った世界各国のスマートフォン普及率の調査結果を表したものである。これを見ると日本は世界的には、まだまだ普及が進んでいないことがわかる。英国やオーストラリアは 50%を超えており、世界的にこれ程普及が進んでいるということは、今後の日本の普及率にも十分伸び代があるということを示していると考えられる。

1.1.4　　スマートフォン OS のシェア

これまで、スマートフォンの普及率について述べてきたが、ここではスマートフォンの OS について触れる。代表的なものを挙げると、本研究で取り扱う Android (google)の他に、iPhone (Apple)、Windows Phone (Microsoft)などが挙げられる。

*カッコ内はスマートフォン契約数

図 1-3 国内でのスマートフォン OS 別契約数シェアの予測[3]

図 1-3 は MM 総研[4]が携帯電話利用者に対するアンケート及び出荷統計データなどに基づき、国内におけるスマートフォン OS 別契約数シェアの予測したデータである。元々シェア全体を大きく 2 分していた Android と iOS だが、近年では Android が iOS を上回っているのがわかる。また、スマートフォン向けの OS の世界シェアは、2013 年度 Android が約 80%に到達したという Strategy Analytics による調査結果[5]も報告されている。このように Android は国内のみならず世界的にも広く普及していることがわかる。

1.1.5　　レンズについて

この研究のもう一つの柱がレンズである。私たちはレンズの新しい可能性を生み出し、レンズの需要を増やさなければならない。本研究では、スマートフォンなどに搭載されている内蔵カメラを使用したのでは需要を増やしたとは言いにくいと判断し、外部カメラを接続する方式をとることにした。

レンズは光を発散させたり、集束させたりする特性もち、様々な場所と用途で使用されている。カメラやメガネなどに始まり、研究や医療現場、意外なところではスキャナーなどにも使われており、私たちの生活を支えてくれている。レンズは形や材料などで分類でき、一般的にレンズといわれてすぐ浮かんでくるのが、虫眼鏡などに使われている球面レンズ(図 1-4 左側)だろう。一方で非球面レンズ(図 1-4 右側)というのも存在する。非球面レンズは球面レンズで生じる収差と呼ばれる焦点のズレが生じないという特長がある。収差が生じてしまうと、レンズを複数枚重ねるなどして収差を取り除く必要があるが、非球面レンズはその必要がないため、使用するレンズの枚数を減らすことができ、現在では広くカメラなどに使用されるようになっている。その他に様々な形の異形レンズも存在し、様々な用途で使用されている。またレンズはガラスから作られるのが一般的だが、プラスチックで作られたものも存在する。プラスチックはガラスに比べ成形が容易で、非球面レンズや異形レンズに多く使われていて、レンズユニットの小型・軽量化ができ、デジタルカメラやスキャナーなどに使われている。

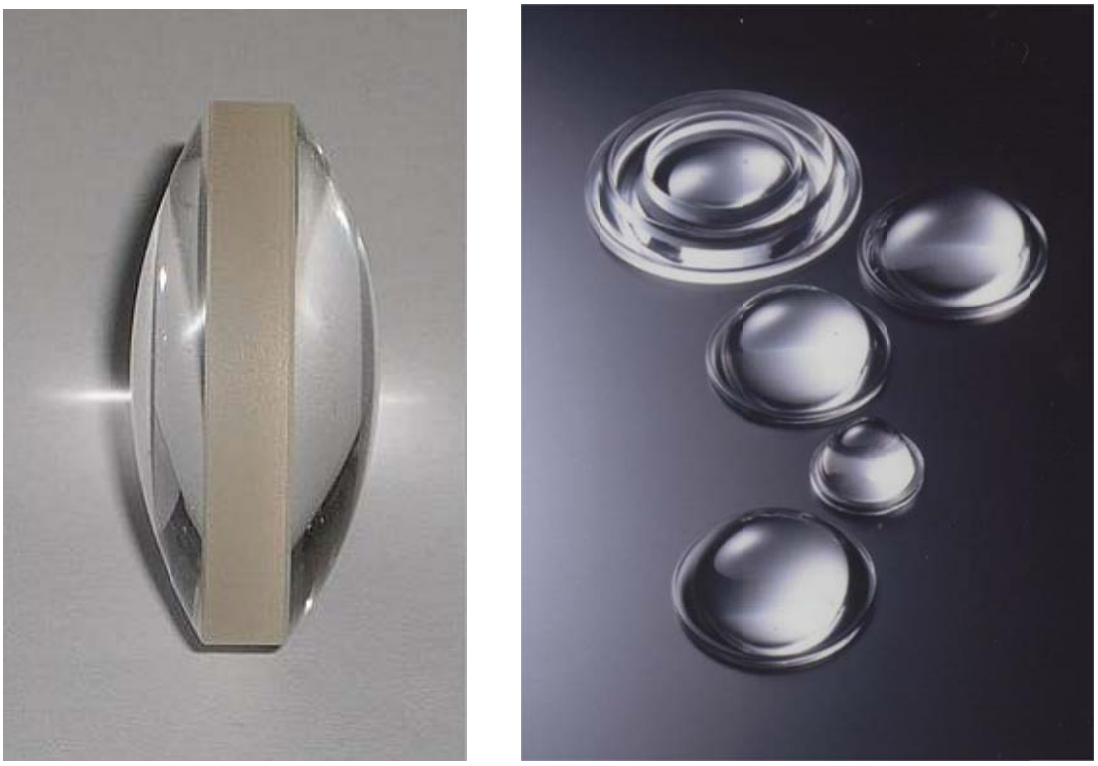


図 1-4 球面レンズ(左)と非球面レンズ(右)[6]

1.2　研究の概要　（小野担当）

1.2.1　製作物の決定

私たちはレンズの新しい可能性として、観光望遠鏡に目を付けた。もちろん観光望遠鏡にはレンズが使用されている。昨今、観光望遠鏡の設置台数は減少傾向にあると考えられ、事実 2012 年 5 月に開業したスカイツリーには観光望遠鏡は設置されていないうえ、東京タワーでも 2008 年に観光望遠鏡が撤去された。その理由を考えると、観光望遠鏡の設置により、展望スペースが損なわれてしまうこと、また大きな鉄の塊により、景観が損なわれることなどが挙げられる。

私たちは、こういった従来の観光望遠鏡のデメリットを克服した、新たな観光望遠鏡のコンセプトを生み出すことを目指していく。

従来型の観光望遠鏡が減っているということはすなわち、その分レンズの需要が減っているということであるが、私たちの新しい観光望遠鏡により、レンズの需要を増やせると考えている。

1.2.2　　研究の方向性

私たちは、新たな観光用の望遠鏡を開発していくこととした。ここでは、その新たな観光用の望遠鏡の構想について説明する。

私たちが目指したのは、普及目覚ましい Android 端末とレンズを組み合わせた新しい望遠鏡である。具体的には、展望台のスペースと景観を損なわぬよう、観光施設の外側に取り付けた望遠鏡を、観光施設内の Android 端末で操作できるというものである(図 1-5)。Android 端末で操作できる動作は、上下左右運動及び、ズーム機能などの基本機能の他、望遠鏡で見ている景色をそのまま端末に保存できるなど、電子端末ならではの付加価値を付けることも考える。

また、本研究において使用する Android 端末は Acer 製のタブレット、ICONIA Tab A500(図 1-6)とする。スマートフォンではなくタブレットを選択した理由は、タブレット端末のほうが画面のサイズが大きく、研究時や発表時などに有利と考えたからである。

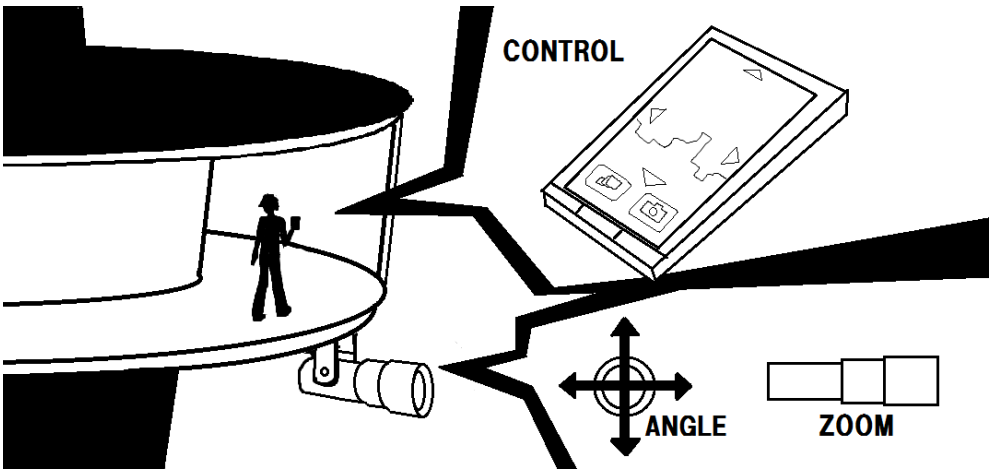


図 1-5 完成イメージ図

図 1-6 ICONIA Tab A500

1.2.3　　望遠カメラ

図 1-7 提供して頂いた望遠カメラ

図 1-7 は K 社より提供された望遠カメラである。USB で PC 等に接続でき、カメラ本体に 3 箇所ある回転部においてそれぞれ、ズーム・フォーカス・明暗の調節ができるようになっている。このカメラの使用を前提として研究を進めていく事とする。

1.2.4　研究の進行について

私たちは研究を進めていくうえで、まず大きく 2 つにチーム分けをした。

１つ目は機構班である。機構班は、この研究の柱の 1 つである望遠カメラをマウントし、さらにそのカメラを上下左右運動させる機構の設計と製造が主な役割である。

もう 1 つがプログラミング班である。こちらは、実際に Android タブレットを操作することで、望遠カメラを動かせるようにすることと、電子端末ならではの付加価値を付けることが主な目的である。プログラミング班は 3 つの役割に分かれ、1 人は望遠カメラを操作するのに必要なアプリケーションの開発、1 人はカメラの駆動に必要なモータなどを制御する回路の構築、もう 1 人は付加価値の開発の役割を担う。

最後に、本研究における製作物の名称を Android 観光望遠鏡とすることとした。

第 2 章　　機構　　(紀伊担当)

2.1　　目標　　(紀伊担当)

2.1.1　　概要

本章では新しい観光望遠鏡のために製作した機構について解説する。

機構製作では、望遠カメラのズーム・フォーカス・明暗を調節でき、上下左右方向に操作することのできる機構の作製を目指した。

2.1.2　　要求機能・制約条件

望遠鏡を自在に操作するために、本研究で作成する望遠カメラに必要な機能は下記の通りである。

1. カメラを上下左右運動（首振り）させる。
2. ズーム・フォーカス・明暗の調節をさせる。

次に機構を動作させるにあたり、モータや回路を取り付ける必要がある。さらに想定する取り付け位置も考慮すると、下記の事が制約される。

1. モータや回路のスペースを考慮したうえで機構の設計ができる。
2. 望遠鏡を支えることができる。
3. 上から吊るすかたちで設置できる。

これらを考慮し製作に取り組んでいく。

2.1.3　　動作原理

本研究で製作する Android タブレット上で操作できる望遠カメラの仕組みを簡単に説明する。

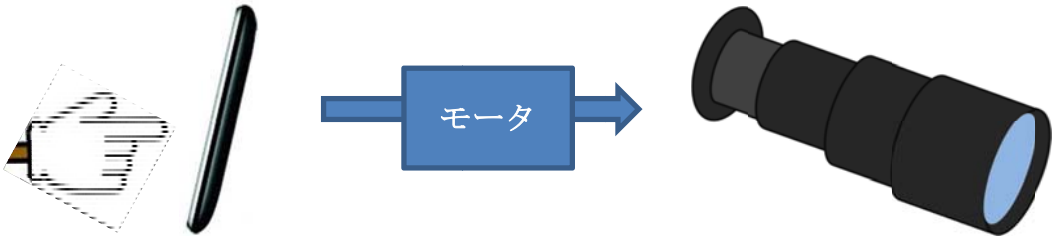


図 2-1 動作原理

Android タブレット上で望遠鏡の操作を行うということは、図 2-1 のように Android タブレットをタッチすることで、プログラムを通してモータを回転させて望遠カメラの必要な箇所を調節できるようにするというものである。機構班としては、その信号を受け取って必要な動作を実現するための機構を作製していく。

2.2　機構の作製　(紀伊担当)

2.2.1　機構のアイデア

2.1.2 の要求機能・制約条件を満たす機構の仕組みについて、機構班全体で様々な案を出した。

その結果、モータによってズーム・フォーカス・明暗の 3 個所に加え、上下左右方向に動かすことが出来ることを前提に、1 つの大まかなイメージを立てた。

図 2-2 が採用されたイメージである。

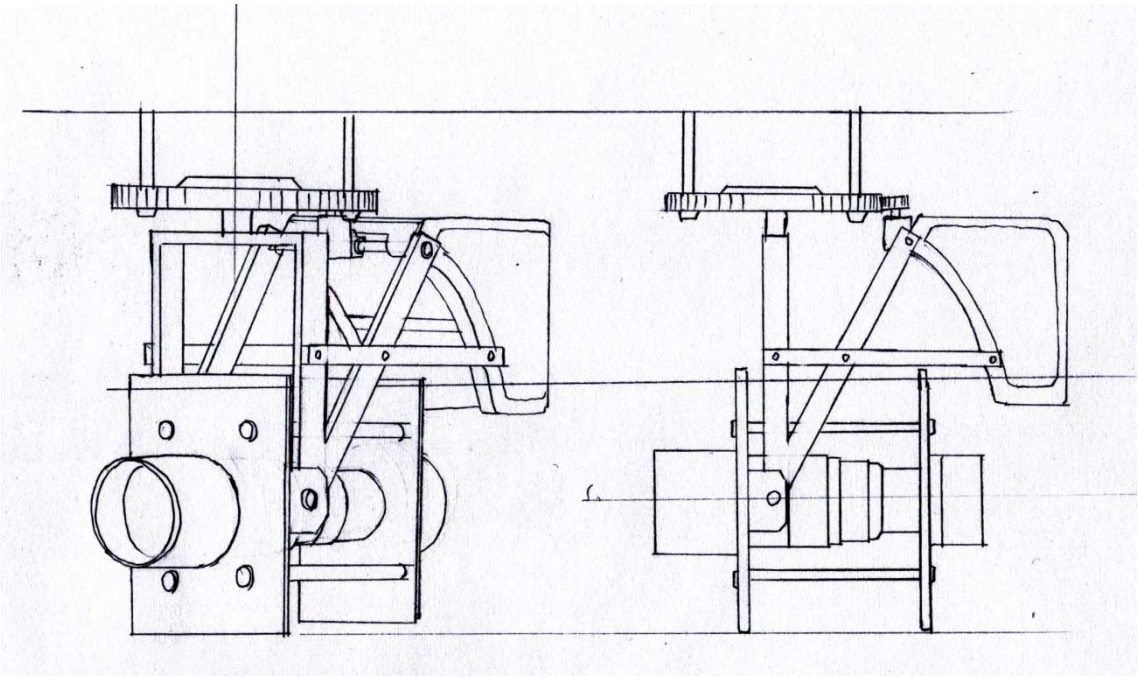


図 2-2 イメージ図

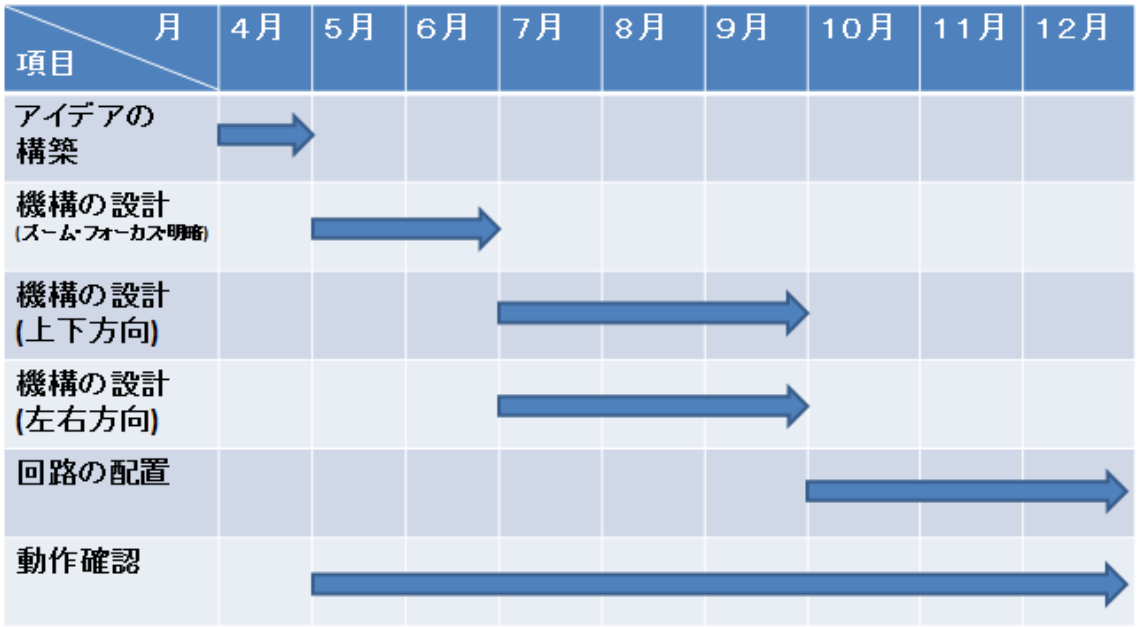
2.2.2　　計画

イメージ図をもとに、計画を立てた。表 2-1 が機構を作る上での計画表となる。

まずは 4 月に機構のアイデアについて班員で話し合い、どういう仕組みでどのように動くのかを検討した。次に 5 月から 6 月にかけて望遠鏡のズーム・フォーカス・明暗の調節機構の設計から加工を行い、7 月から 9 月にかけて望遠鏡の上下左右調節の設計・加工を行った。最後に 10 月から 12 月にかけて回路のレイアウトを行うという計画を立てた。

また、各工程ごとに動作確認を行い、改善点などを見ることも行った。

表 2-1 計画表



2.2.3　　機体の設計（ズーム・フォーカス・明暗）

カメラ周りの調節機構は、望遠鏡のズーム・フォーカス・明暗の 3 箇所をモータで動かすというものである。

図 2-3 リエゾン提供の望遠鏡

リエゾンから提供された望遠鏡は、図 2-3 で示す部分がそれぞれ回転し、調節を行えるようになっている。私たちは、対象となる 3 つの円筒にフレキラックを巻き付け、モータに取り付けたギヤと噛み合わせることで回転を行わせることにした。

そこで図 2-2 のイメージ図と図 2-3 の使用する望遠鏡を受けて、各部品図面を 3 次元 CAD ソフトウェアである SolidWorks を用いて作成した。SolidWorks を用いることで設計、アセンブリ、図面の修正を容易にでき、また正確な図面を簡単に作成することができるので作成時間も短縮することができる。

図 2-4 がズーム・フォーカス・明暗を調節するための機構に必要な部品を望遠鏡に取り付けた図である。

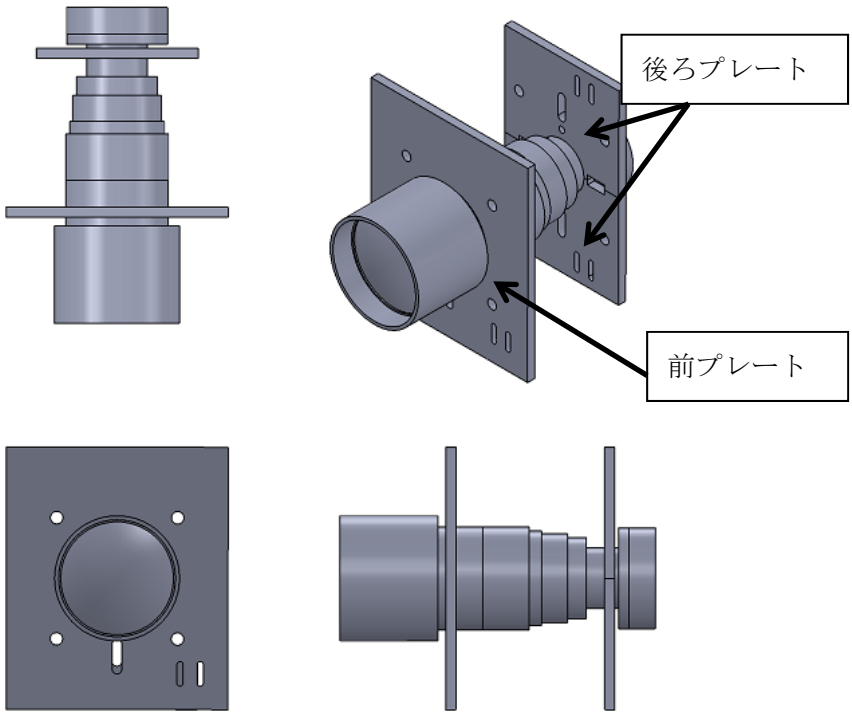


図 2-4 加工部全体像

機構の作製にあたって、必要な部品の設計図を示す。

図 2-5 が前プレートの図面である。

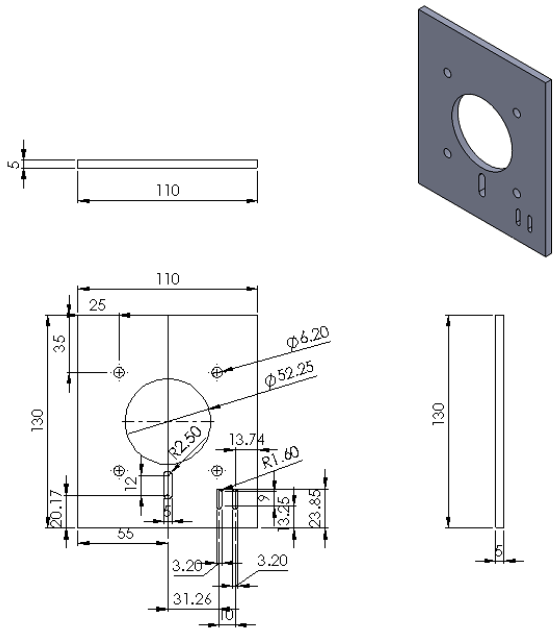


図 2-5 前プレート

図 2-6 が後ろプレートの図面である。

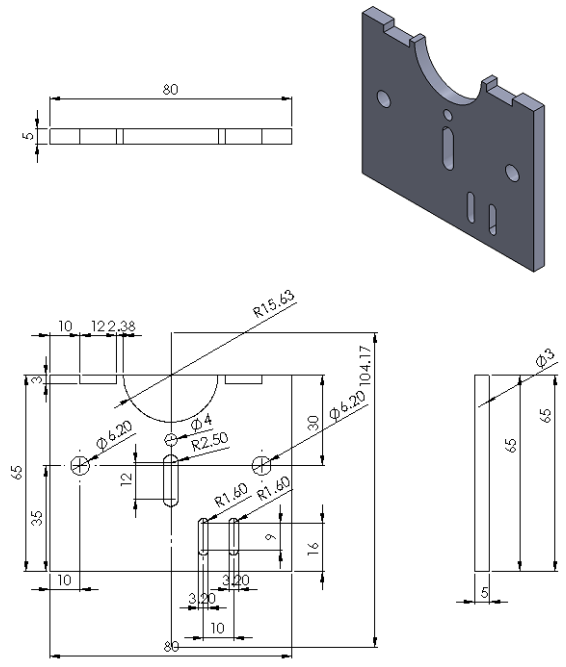


図 2-6 後ろプレート

後ろプレートは図 2-4 の全体像からも分かる通り、同じもの 2 枚で 1 組とする。

また、使用するモータとしては、図 2-7 のタミヤの 4 速ウォームギヤボックス HE を選択した。4 速ウォームギヤボックス HE は回転数が低く、大きなトルクを発揮するのが特徴であり、4 種のギヤ比が選べ、最も低速の 1428.2：1 では 1 分間に出力シャフトが約 7 回転する。今回はこの 1428.2：1、トルク 2306gf.cm で使用する。

図 2-7 タミヤ 4 速ウォームギヤボックス HE [7]

また、プレートとモータを接続するために、図 2-8 に示すカメダデンキの補助金具 LKHK を使用する。

図 2-8 補助金具 LKHK [8]

さらに、対象となる 3 箇所の可動部には図 2-9 で示す小原歯車工業（KHK）の DR 成形フレキラックを使用する。フレキラックとは、変形自在で、幅広い用途が可能なギアラックであり、これにギヤを取り付けたモータを噛み合わせることで回転を行う仕組みである。

図 2-9 DR 成形フレキラック [9]

図 2-10 が加工した前プレートと 2 枚の後ろプレートにモータ等を取り付けた全体像である。

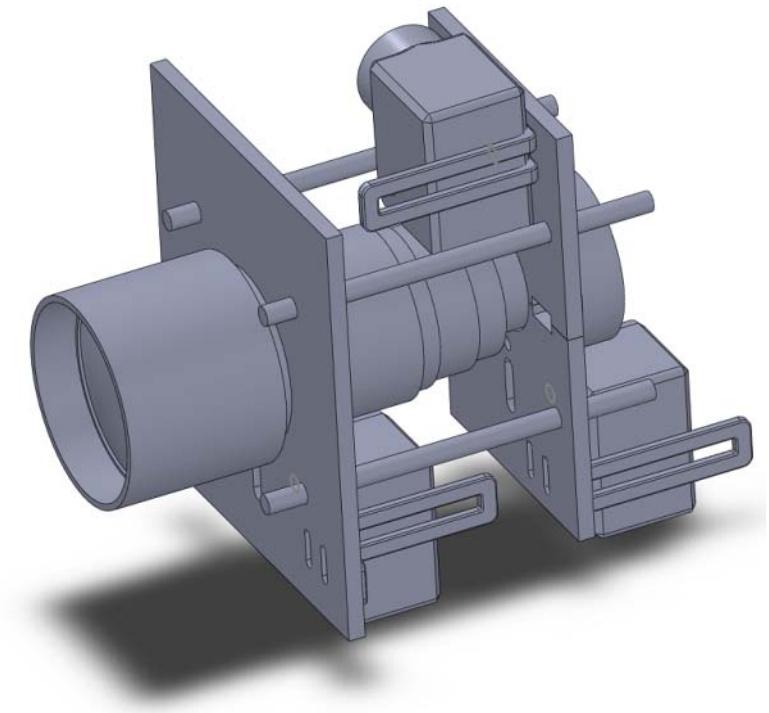


図 2-10 全体像（ズーム・フォーカス・明暗）

望遠鏡のズーム・フォーカス・明暗の調節部に取り付けたフレキラックに噛み合わせて回転させるために、モータの軸に図 2-11～図 2-14 に示すピニオンギヤを取り付ける。レインボープロダクツのピニオンを用いる。これらのピニオンギヤは、フレキラックのモジュールと歯の大きさが全て等しいので使用可能である。

図 2-11 ピニオンギヤ

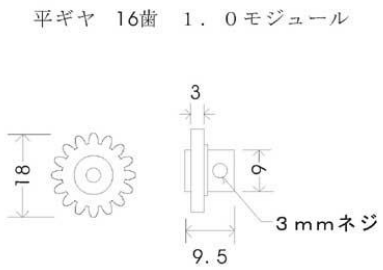


図 2-12 平ギヤ 16 歯詳細［10］

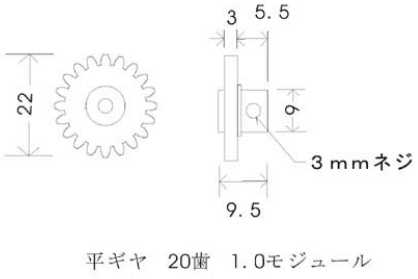


図 2-13 平ギヤ 20 歯詳細［11］

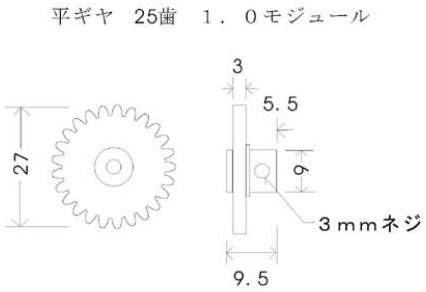


図 2-14 平ギヤ 25 歯詳細［12］

次に、使用した部品の表が表 2-2 である。
これらが示すのは望遠鏡のズーム・フォーカス・明暗を調節する機構に必要な部品である。

表 2-2 ズーム・フォーカス・明暗調節機構部品表

部品	個数	加工の要否
1. 前プレート	1	要
2. 後ろプレート	2	要
3. モータ	3	否
4. ピニオンギヤ	3	否
5. ラック	3	要
6. 寸切りボルト（M6）	4	要
7. ナット（M6）	16	否
8. 平座金	16	否
9. バネ座金	16	否
10. L字金具	3	否
11. ネジ (M3)	12	否
12. ナット（M3）	12	否

図 2-15 M6 ステンレス寸切りボルト [13]

前プレートと後ろプレートを支える 4 本の寸切りボルトは、前プレートと後ろプレートを十分に固定するのに必要な長さで切断した。

この部品表をもとに加工が必要な部品は製作し、加工が不要な部品は既存製品を使用する。

次に、組み立て法について解説する。

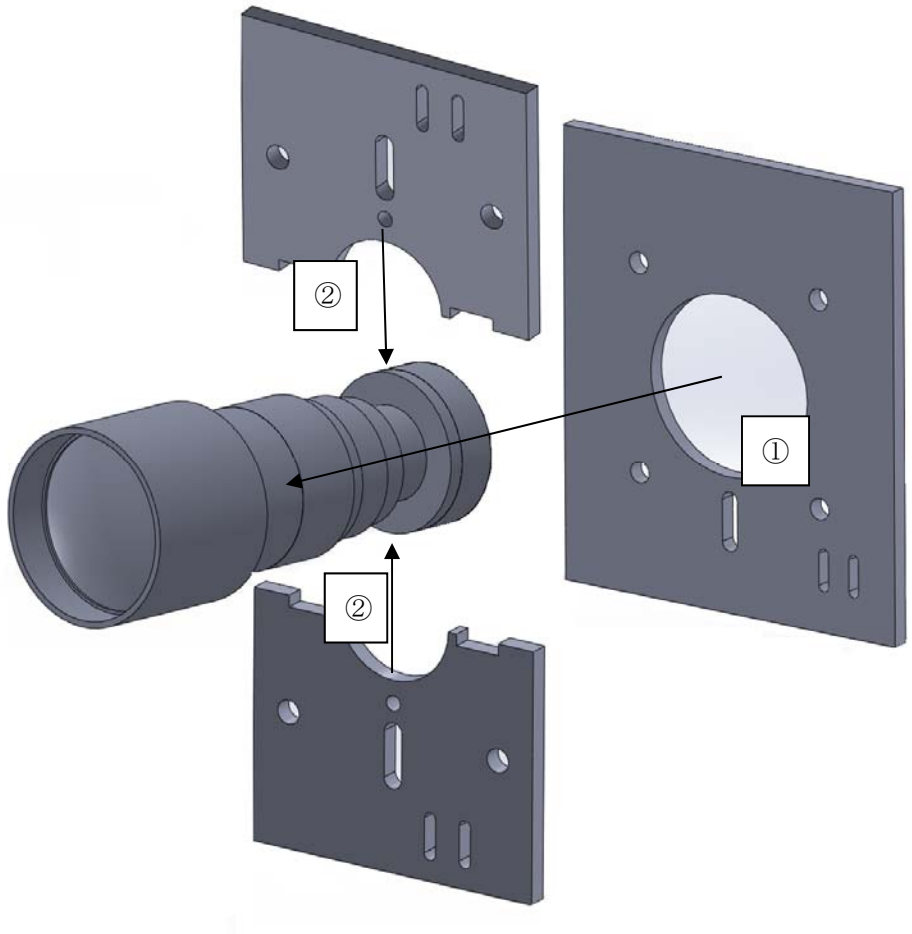


図 2-16 プレート取り付けイメージ

図 2-16 の様にしてカメラにプレートを取り付ける。前プレートはカメラの接眼部からはめ込み、後ろプレートはカメラのくびれに上下から挟み込む形で取り付ける。

また、その際に望遠鏡にあらかじめついている基板等を止めるためのネジとプレートとを共締めする。その後、前後のプレートに寸切りボルトを通し、ワッシャー・バネ座金・ナットを挟みこむ形で固定する（番号は取り付け順序を示す）。

完成写真（ズーム・フォーカス・明暗）

図 2-5、図 2-6 の図面をもとにして、金属加工を行った。機体の製作は犬目校舎 1 号館の E.C.P センターのワイヤー加工機を使用し、滑川軽銅のアルミ切板厚さ 5mm を部品ごとに切り出した。次に、切り出した部品にハイトゲージにより罫書きを行い、ポンチを打ってからボール盤によって穴加工を行った。場合によってはその穴をもとにもう一度ワイヤー加工機を使用した。

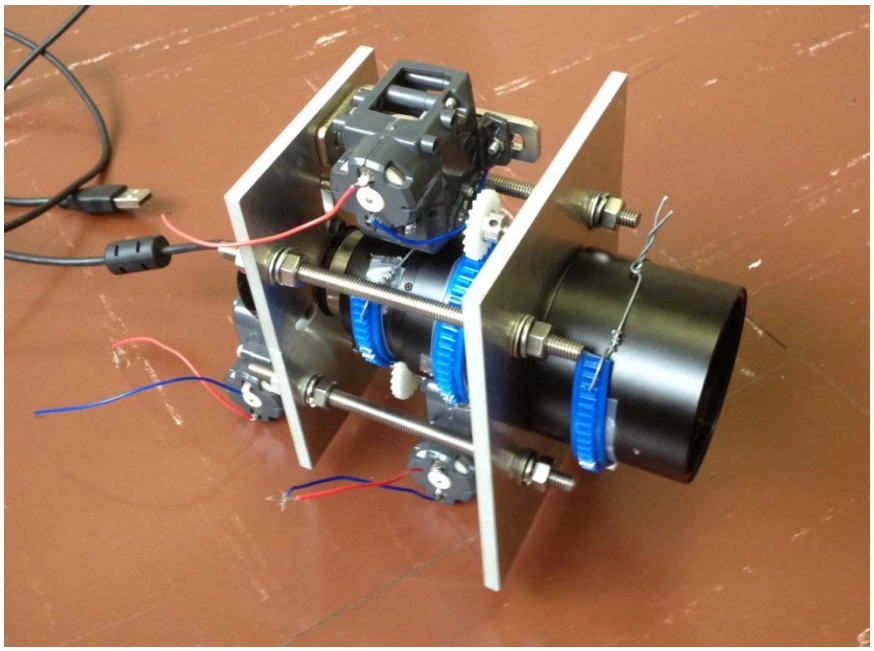


図 2-17 ズーム・フォーカス・明暗調節機構

図 2-17 の様にプレート等を設置し、モータを適切な位置に取り付けた。望遠鏡に取り付けてあるフレキラックは望遠鏡を傷つけないため、またラックの滑り防止のためにシリコンゴムを挟んだのち針金で縛り、固定した。なお、シリコンゴムは後に強力両面テープに変更した。この点については 2.2.6 参照。

動作確認

思い通りに組み上がった。ギヤとフレキラックは適切に噛み合い、モータを回転させることによって望遠鏡の可動部を調節することができた。

改善点

望遠鏡に取り付けたフレキラックが思いのほか固く、取り付けに時間を要した。また、モータの動作に伴い位置が変わってしまうことがあった。シリコンゴムを挟み、なおかつフレキラックを密着させ針金で固定しているが、ここの仕組みも再検討の余地がある。

また、カメラの回転が限界を超えても回転させようとした場合、カメラやモータに負担がかかることでモータとフレキラックがうまく噛み合わなくなることがあった。故障の原因となるため、回転限界を感知するとそれ以上回転させないような仕組みが必要である。

2.2.4　　機体の設計（上下左右方向運動）

次に、上下左右方向運動の機構の設計について解説する。ズーム・フォーカス・明暗を調節するための機構では 4 本の寸切りボルトが、前プレートと後ろプレートを固定している。上下左右に動かす仕組みを構築する上で、この 4 本のボルトをうまく利用することにした。またその際、望遠鏡を上下左右に十分に動かせることを第 1 に考える。

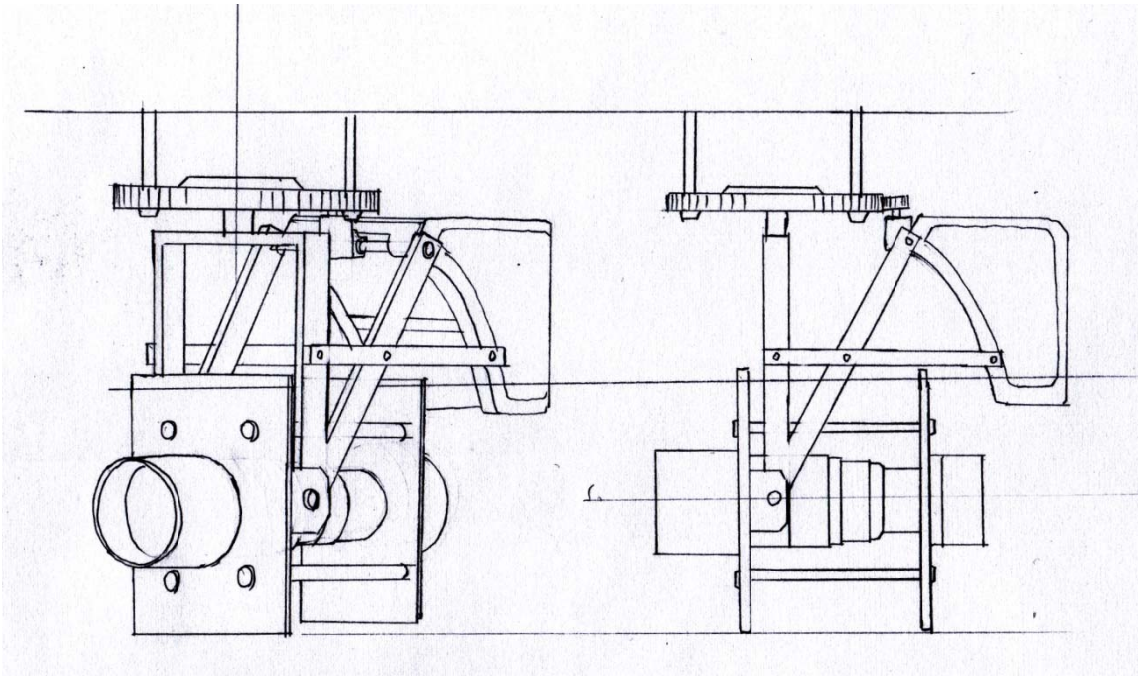


図 2-18 イメージ図

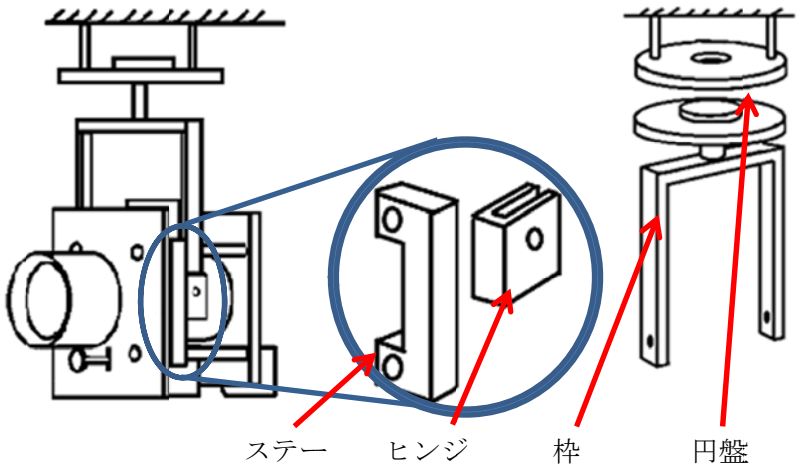


図 2-19 上下左右方向運動パーツイメージ

当初のイメージをもとに、図 2-19 のような形で寸切りボルトを上下に固定するステーのようなパーツに、ヒンジとなるパーツを組み合わせてカメラの左右に取り付ける。この部分を中心に上下回転するように枠をはめ、あらかじめ天井に固定した円盤にぶら下がるような形で固定する。モータの取り付けに関しては後に説明する。

図 2-20 および図 2-21 が上下左右運動のための機構で、加工が必要な部分を組み立てたものである。

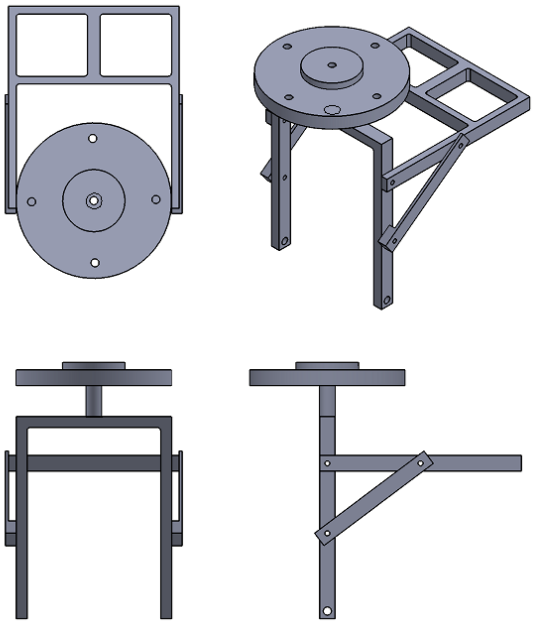


図 2-20 上下左右機構

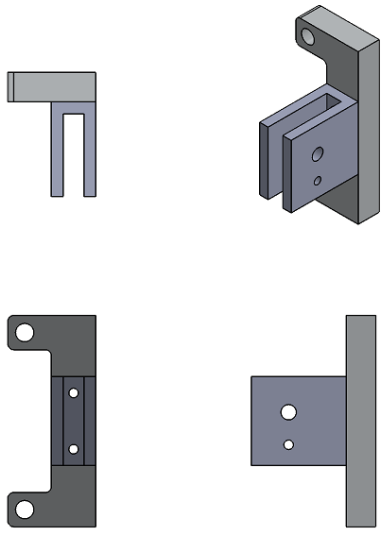


図 2-21 ヒンジ

図 2-22 がズーム・フォーカス・明暗の機構に上下左右の機構を取り付けた全体像である。

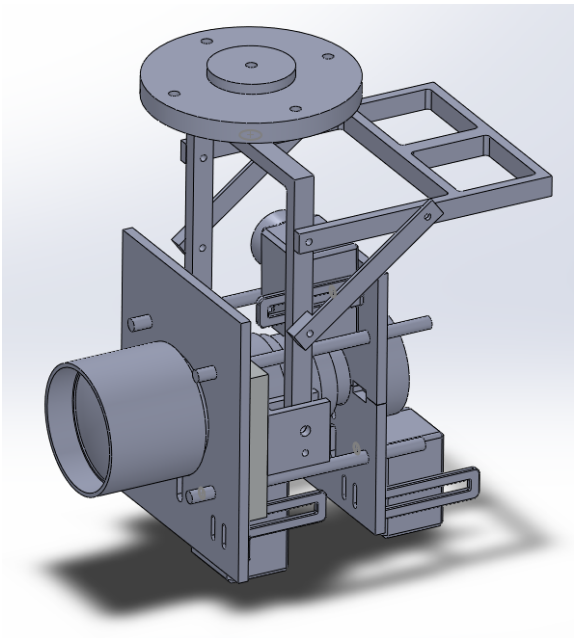


図 2-22 全体像

機構の作製にあたって、必要な部品の設計図を図 2-23～図 2-30 に示す。

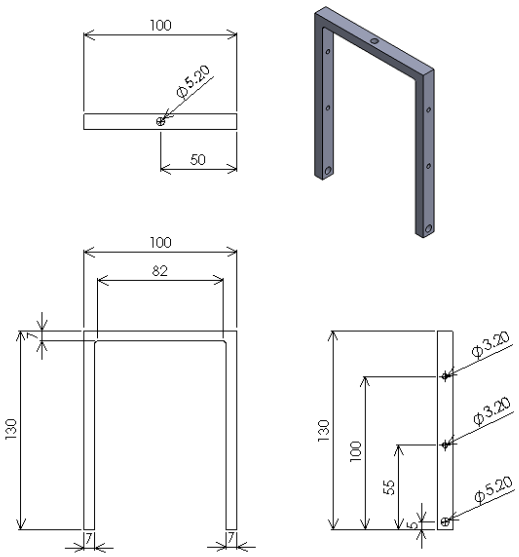


図 2-23 柱

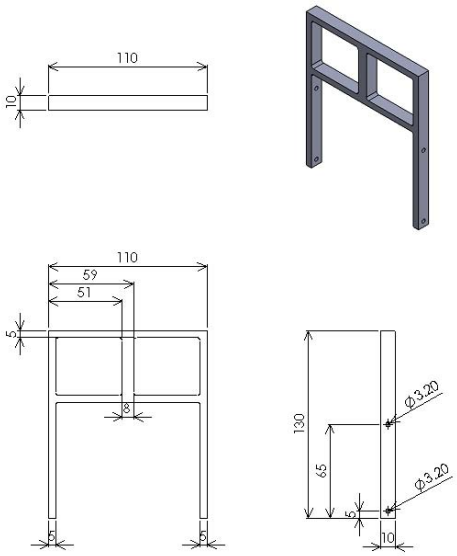
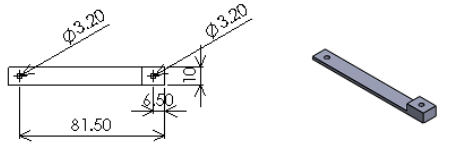
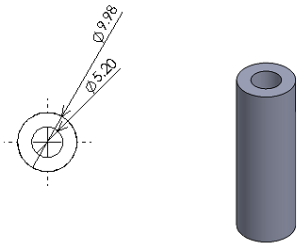


図 2-24 回路スペース





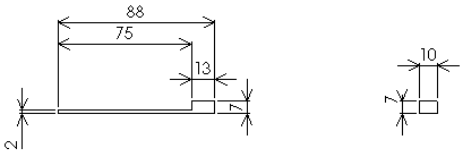


図 2-25 支え

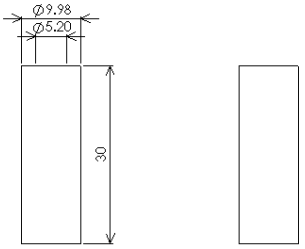


図 2-26 支柱

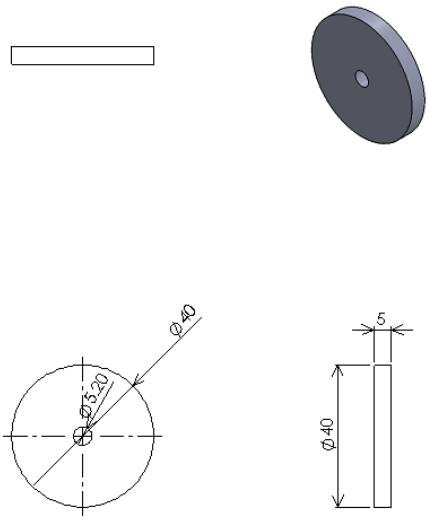


図 2-27 留め具

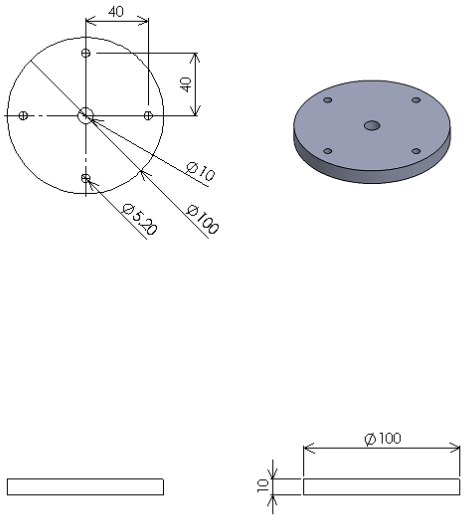


図 2-28 回転盤

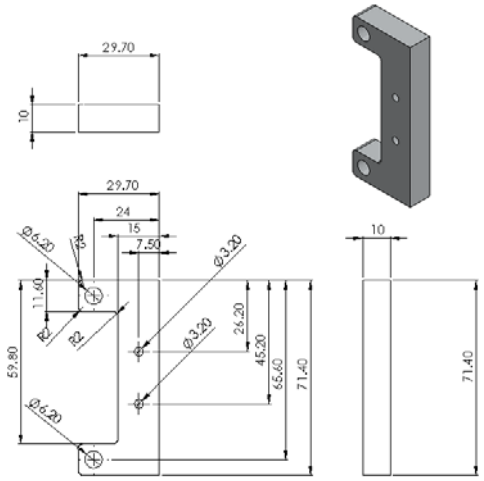


図 2-29 ステー

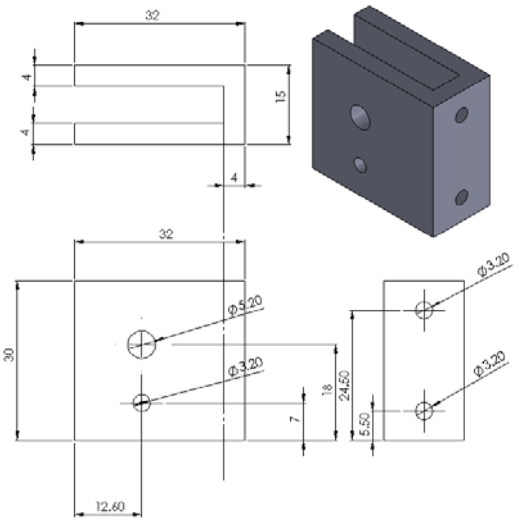


図 2-30 ヒンジ

2.2.5　モータについて

望遠鏡のズーム・フォーカス・明暗を調節する機構に組み合わせるかたちで取り付けられるよう、上下左右方向に動作させることのできる機構の設計を行った。その際、使用するモータとしては、上下方向にサーボモータ KRS-788HV ICS Red Version[14]、左右方向に、高トルクの DC モータであるタミヤギヤードモータ 380K300[15]を採用した。この望遠鏡は高層建築の軒下などの外部に設置することを考えているため、左右方向に関しては 180 度、上下方向に関しては下に見下ろす形で 90 度あれば十分としたためである。

上下方向については T=重心×距離×角度の公式に当てはめて選定した。
望遠鏡のズーム・フォーカス・明暗の機構は重量が約 1.2kg である。また、使用するサーボモータには後に述べるようにブラケットを装着させた。それにより、重心までの距離が約 2.5cm となる。また角度を 90 度としていることから、公式に当てはめると、

T=重心×距離×角度

$T=1.2\times2.5\times\sin90$

T=3 (kg・cm)

よって使用するサーボモータ KRS-788HV ICS Red Version（トルク 10.0kgf・cm）で十分とした。

以下で左右方向と上下方向のモータの取り付け法についてそれぞれ詳しく述べる。

・左右方向

望遠鏡を左右方向に回転させる仕組みとして、フレキラックとピニオンの関係を用いた。図 2-31 の回転盤の周りにズームの箇所でも用いたフレキラックを巻き付け、そこにピニオンを取り付けたモータをかみ合わせることで回転を行う。ピニオンはフレキラックに合わせるために、専用のものを使用した。また、フレキラックは接着のしやすさから、回転盤の周りに 360 度巻き付けた。

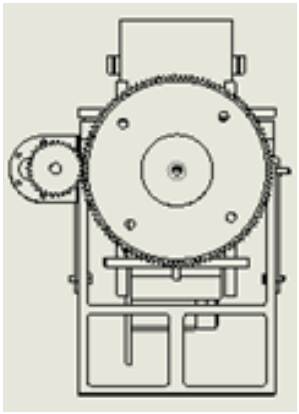


図 2-31 左右方向機構

図 2-32 DR 成形フレキラック

図 2-33 SSDR DR 専用ピニオン [16]

また、使用するモータはタミヤのギヤードモータ 380K300 であり、機構に搭載するにあたって専用のマウントも購入した。これらを組み合わせて左右の機構を作製する。

図 2-34 タミヤギヤードモータ 380K300 [15]

図 2-35 ギヤードモータ用マウント[17]

・上下方向

上下方向は望遠鏡のサイドにサーボモータを取り付けることで回転を行う。また、モータには KONDO の KRS-788HV ICS Red Version を選定した。このモータは最大動作角度が 180 度であるが、望遠鏡は高層建築の軒下などの外部に設置することを考えているため、下に見下ろす形で 90 度あれば十分であるとして設置をした。

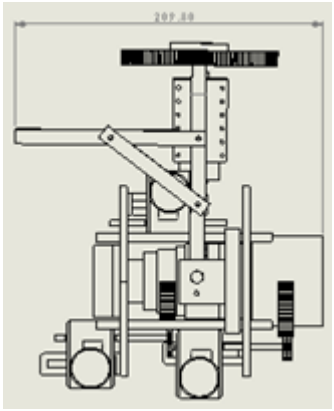


図 2-36 上下方向機構

図 2-37 KRS-788HV ICS Red Version [14]

また、機構に搭載するにあたって、このモータにあうサーボブラケットもあわせて購入した。これらを組み合わせて上下の機構を作製する。

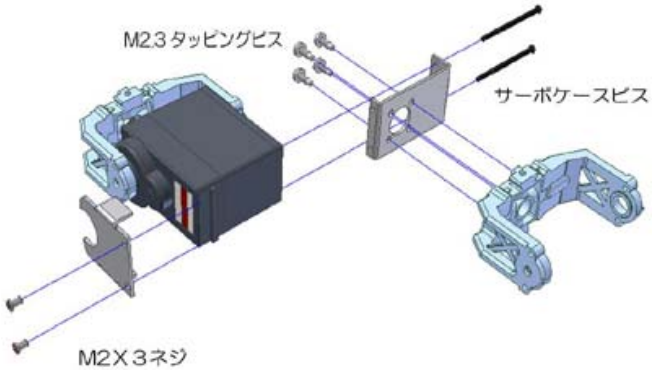


図 2-38 SC-78X-A(ブラケットタイプ A) [18]

次に、部品表を作成する。機構部、上下方向、左右方向の 3 つに分けて記す。この部品表をもとに要加工部品は製作し、加工が不要な部品は既存製品の選定を行う。

表 2-3 機構部部品表

部品	個数	加工の要否
1. 柱	1	要
2. 回路スペース	1	要
3. 支え	1	要
4. 支柱	1	要
5. 留め具	1	要
6. 回転盤	1	要
7. ステー	1	要
8. ヒンジ	1	要
9. ボルト（M5）	7	否
10. ボルト（M3）	6	否
11. ナット（M5）	7	否
12. ナット（M3）	6	否

表 2-4 上下方向部品表

部品	個数	加工の要否
1. モータ	1	否
2. ブラケット	1	否
3. ボルト（M2）	2	否
4. ボルト（M3）	3	否
5. ナット（M2）	2	否
6. ナット（M3）	3	否
7. L 字金具	1	要

表 2-5 左右方向部品表

部品	個数	加工の要否
モータ	1	否
マウント	1	要
ギヤ	1	否
ボルト（M3）	3	否
ナット（M3）	3	否
バネ座金	3	否
ラック	1	要

次に、上下左右方向に動作可能な機構の組み立て法について解説する。

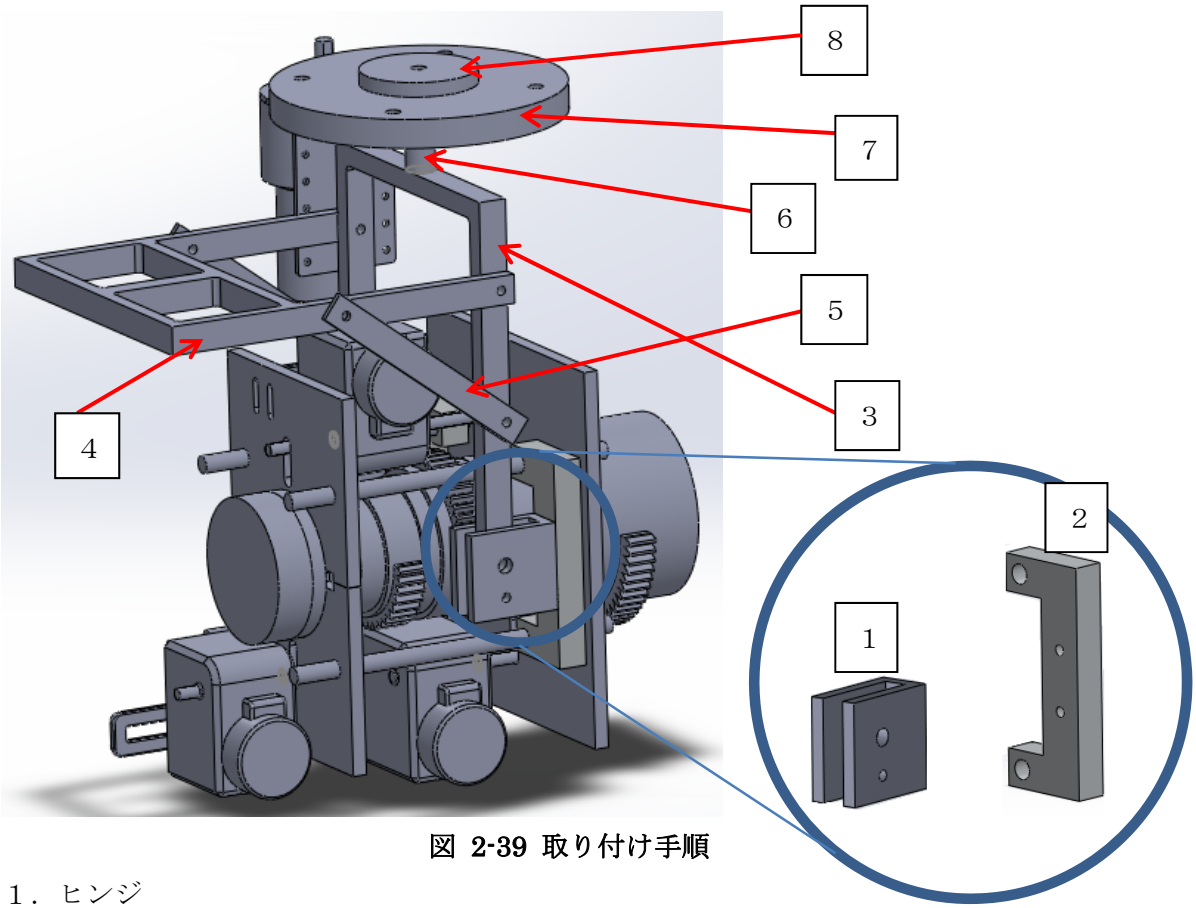


図 2-39 取り付け手順

1．ヒンジ
2．ステー
3．柱
4．回路スペース
5．支え
6．支柱
7．回転盤
8．留め具

まず 1 に 2 を取り付け、それを左図の寸切りボルトの部分に差し込んで固定する。これを左右合わせて 2 つ取り付ける。次にその固定した部分に 3、4、5 の順で枠組みを取り付ける。そのあと 6 を取り付け、7 をそこに差し込む。そのあと 8 を 6 にボルトでしっかりと固定する。最後に 7 にあいた 4 つの穴にボルトを通し、天井に取り付け固定する。

また、各部品をつなぐためにはボルトとナットを使用した。1、2 と 3 をつなぐ中心の部分、6 と 8 をつなぐ部分には M5、それ以外の部分には M3 を使用した。

完成写真（上下左右）

上下左右の機構についても、ズーム・フォーカス・明暗の場合と同様に E.C.P センターにて金属加工を行った。

図 2-40 上下左右機構

図 2-40 は研究室の棚を天井に見立てて仮止めしているものである。

動作確認

思い通りに組み上がった。望遠鏡を満足に支えることができ、なおかつギヤとフレキラックは適切に噛み合い、モータを回転させることによって可動部を調節することができた。

改善点

機構的な動作は特に問題なかったが、ここにプログラム等で使用する配線を設置するとさらに複雑になるため、分かりやすい配置が必要である。これについては後に述べる。

2.2.6　　機体の設計　（停止機能・回路）

・停止機能

2.2.3 のズーム・フォーカス・明暗の改善点に挙げた通り、カメラの回転が限界を超えてもなお回転をさせようとした場合、カメラやモータに負担がかかることでモータがラックから脱落してしまった。故障の原因となるため、回転限界を感知すると、それ以上回転させないような仕組みが必要である。

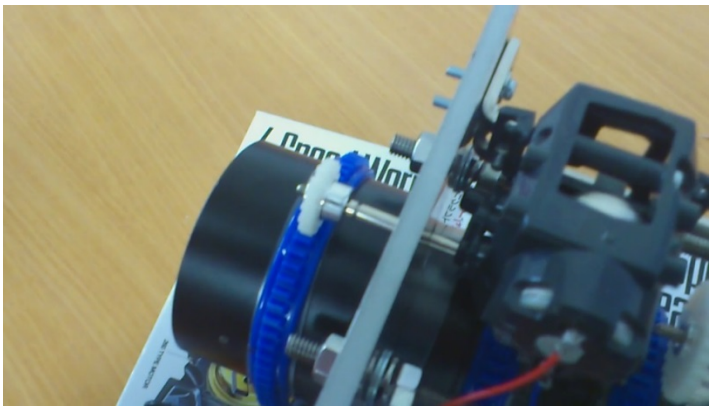


図 2-41 ギヤ接合部

それを防ぐために、マイクロスイッチを用いた緊急停止機能を持たせることにした。簡単に動作の仕組みを説明すると、マイクロスイッチが押された場合にのみ、それ以上同じ方向に回転することを防ぐものである。

マイクロスイッチを取り付ける前に、カメラの各部位に巻き付けてあるフレキラックをカメラに改めてしっかり固定し、なるべくズレが起こらないようにした。これまでシリコンゴムを挟んでいたものを、より粘着度の高い強力両面テープに変えた。

使用したマイクロスイッチはコンパクトで使用箇所に制限の少ない、神明電機の MQS-54-5L マイクロ SW を使用した。ズーム・フォーカス・明暗の 3 箇所にそれぞれ 2 個ずつ、計 6 個のマイクロスイッチを設置した

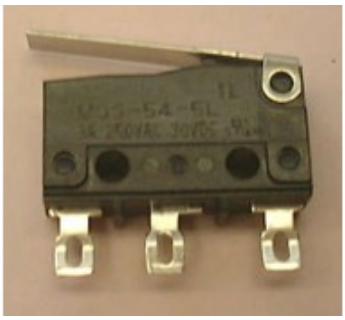


図 2-42 神明電機 MQS-54-5L マイクロ SW [19]

マイクロスイッチは、カメラの回転限界の手前でスイッチが押されるような場所を選んで設置し、カメラに巻き付けてあるフレキラックが接触してスイッチが押されることで作動するようにする。そのため、カメラを固定する前後のプレート上に瞬間接着剤で取り付ける。その際、手間を省くため導線をあらかじめ半田付けしてから設置した。

マイクロスイッチを取り付けた位置は以下の通りである。前部から、フォーカス・ズーム・明暗の調節部である。設置する際、箇所によってはマイクロスイッチ同士が近接しているので、互いの導線が触れないように熱圧縮チューブを巻いて対応した。

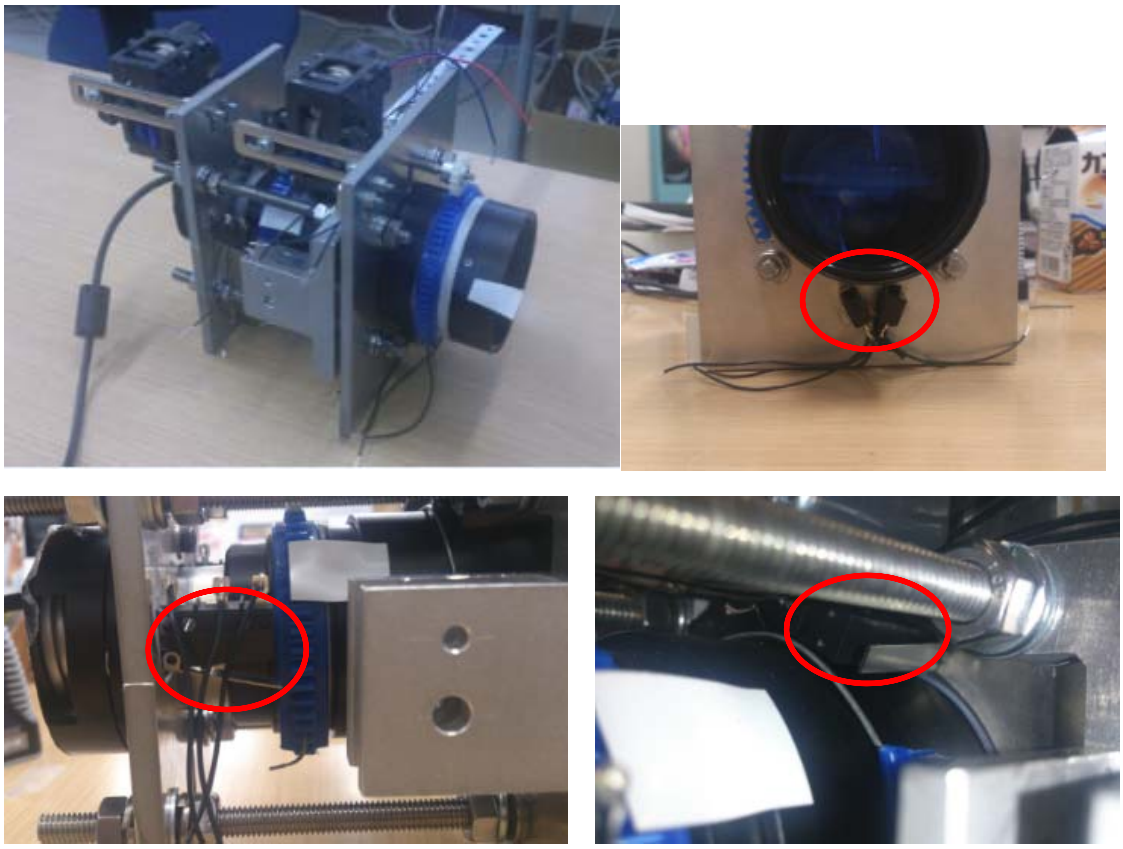


図 2-43 マイクロスイッチの設置

動作確認

全ての位置で適切にスイッチが押されることを確認した。

改善点

プログラム班が実験をしたところ、スイッチを押してから止まるまでに少々時間がかかり、スイッチに負担がかかりマイクロスイッチが脱落してしまう箇所があった。
ラグを考えてスイッチを余裕のある位置へ移動した。また、プログラム班にはモータをより瞬時に停止できるような仕組みの構築を提案した。詳細は 3 章 5 項目に記す。

・回路

完成した機構には、多くの導線が混在している。これらをきれいに配置し、回路にまとめる必要がある。また、カメラは上下左右に動作するため、導線が絡まることを避けなければならない。

まず、図 2-45 の黒丸で示した回路スペースに回路を取り付けていく。

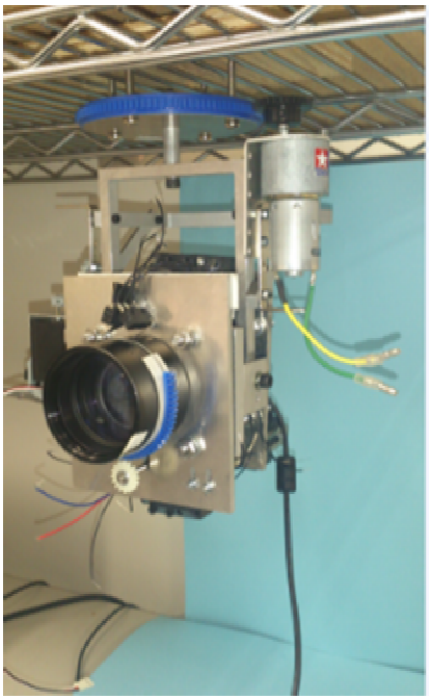


図 2-44 全体像

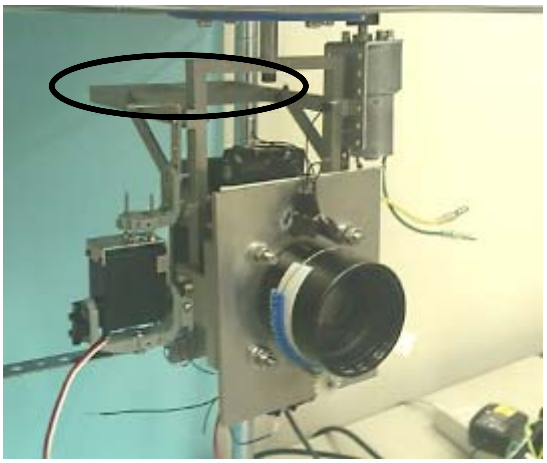


図 2-45 回路スペース表示

作製した回路は望遠鏡のズーム・フォーカス・明暗の操作と上下左右運動、そしてマイクロスイッチの接触検出のための回路である。

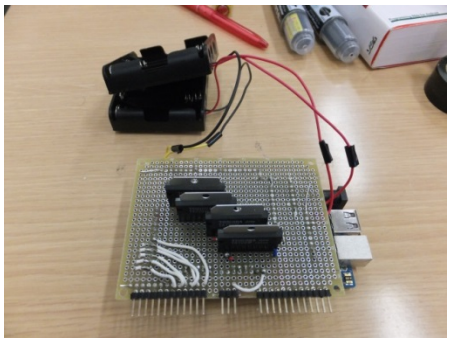


図 2-46 回路写真

図 2-46 はプログラム班が作製した回路である。Arduino ADK の上にモータやマイクロスイッチを制御する基板がのっており、乾電池による電力供給が出来るようになっている。

作製した回路を保護し、まとめるためにプラスチックケースの中にまとめることにした。

このケースの中に、基板や Arduino、電源を設置しカメラの上部に取り付ける。

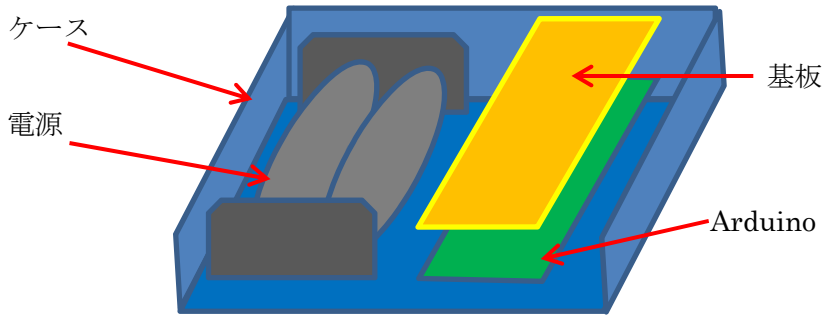


図 2-47 回路ケースイメージ図

配線については、ケースに収める方向を確定した後、ケースに適切な穴をあけ、そこから機構部へと接続する。その際、コネクターを用いて回路部とモータやスイッチ類との接続を簡単にした。図 2-48 は回路ボックスの内部である。右下にあるのが基板で、その上に Arduino とモータの電源を配置している。

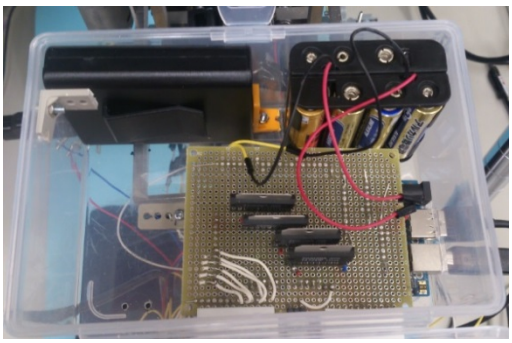


図 2-48 回路ボックス内部

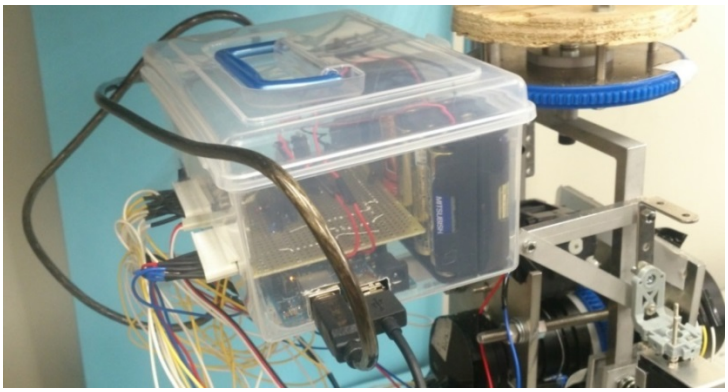


図 2-49 回路ボックス全体写真

図 2-49 は機構に回路ボックスを取り付けたところである。Arduino の電源、望遠カメラの差込口、モータやマイクロスイッチへつながる配線の部分に穴が開いており、簡単に差し込むことができる。普段はふたを閉め、作業を行う際には簡単に開けることができる。

2.3　　製作物の成果　　(紀伊担当)

2.3.1　　動作確認

図 2-50 は完成した全体の写真である。

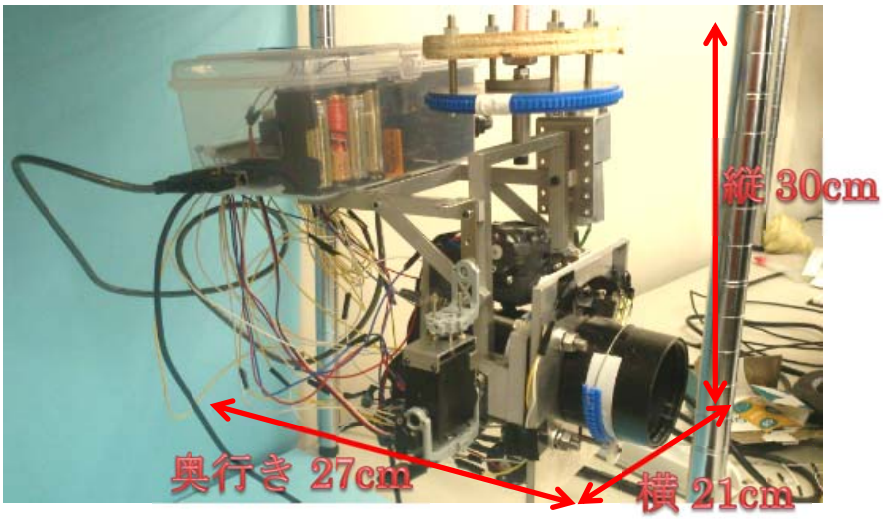


図 2-50 完成写真

また、完成した機体の概要は以下の表に示すとおりである。

表 2-6 大きさと重さ

縦	30cm
横	21cm
奥行き	27cm
全重量	2.84kg

これで必要とされるすべての機能を機構に搭載することが完了したので、プログラム班と協力し、機構の動作確認を行った。確認するのはズーム・フォーカス・明暗の調節・上下左右運動・マイクロスイッチの動作である。

動作確認

回路に負担がかかる事なく正常に作動させることができた。

改善点

電池を複数個使用しているため重量があり、サイズの点からボックスを若干後方に設置したことでバランスに影響があった。

電源をコンセント方式に変えることでスペース、重量を縮小できる。それによりボックスのサイズもより小さい物へと変更できる可能性がある。

2.3.2　　電源の変更

・ボックスの改良

2.3.1 の改善点でも述べたように、回路の部分には電池を複数個使用しており、重量、サイズ共に大きくなってしまっていた（図 2-51）。それを解消するために、後で述べるように、プログラム班が電源をコンセント方式に変更した。それに合わせて機構班では回路を収めていたボックスの改良を行った（図 2-52）。

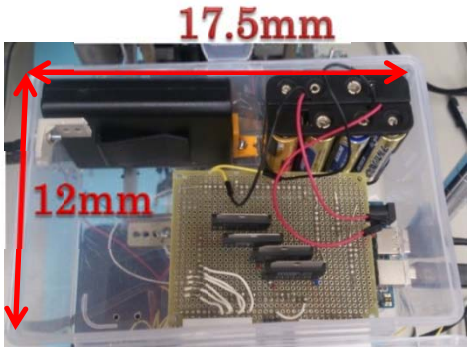


図 2-51 回路ボックス内部

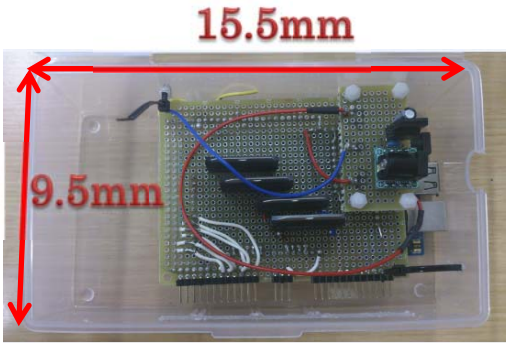


図 2-52 回路ボックス（改良版）内部

これにより、これまでボックスの中に収められていた電池が必要なくなり、回路ボックスの重量を減らすことができるとともに、バランスの問題に関しても解決することができた。また、変更したボックスの左側の空いたスペースには、ビニールテープといった応急処置のための用具をいれておける。

2.3.3　　考察

予定通りすべての工程を終えることができた。

また、制約条件や要求機能をすべて満たすことができたので、目標としていた製作物ができたといえる。

第 3 章　　回路とそのプログラム

(今井担当)

3.1　　目標　　(今井担当)

3.1.1　　概要

本章では、私達が製作した Android 観光望遠鏡を動かすための回路及びマイコンを使用したプログラムについて解説をする。

3.1.2　　機能

本研究で製作する Android 観光望遠鏡に必要な制御機能は下記の通りである。

1. 望遠カメラの上下左右動作をモータで実現させる。
2. ズーム・フォーカス・明暗の調節をモータで実現させる。
3. 安全装置として、機構を破壊しないためのスイッチを回路で実現させる。
4. 基板へ実装する。

これらの機能を実現することを目指し回路・プログラム製作に取り組む。

3.2　　モータ制御　　(今井担当)

3.2.1　　Arduino の使用

機構の制御に Arduino というシステムを使用している。Arduino は AVR マイコン、入出力ポート基板などで構成されるシステムのことを指すが、マイコンとは何なのか、簡単に説明する。

マイコンとはマイクロコントローラーの略称であり、コンピュータシステムをひとつの集積回路に組み込んだものである。CPU コア、メモリ、タイマー、入出力部を含み、プログラムを実行することができる。

Android 観光望遠鏡を製作するコンセプトの中に、景色を Android タブレットに映すという目的がある。そのために外部にカメラを設置する必要があり、屋内で有線あるいは無線で Android タブレットを操作するのである。そこで、PC のように大きく、またコストがかかる物を使用するのではなく、カメラと一緒に設置することができるほど小さく、コストも安価でプログラムを用いて命令をすることができるマイコンを使用することにした。

本研究ではマイコンを含む基板として Arduino ADK を利用することにした(図 3-1)。それは、プログラム初心者でも比較的簡単にプログラムを作ることができるからである。

図 3-1 Arduino ADK

3.2.2　　Arduino を使用して DC モータを動かす

機構班との相談の結果、ズーム・フォーカス・明暗の 3 箇所に左右運動を加えた 4 箇所に DC モータを使用し、有線で Android タブレットから操作することに決めた。そのためにタブレットから Arduino へ命令を送り、Arduino が送られた命令をもとに DC モータを動かすプログラムを製作した。

ここで、DC モータを使用することにした理由に触れる。DC モータとは、固定子に永久磁石を使い、回転子（電機子）にコイルを使って構成されたもので、電機子に流れる電流の向きを切り替えることで磁力の反発、吸引の力で回転力を生成させるものである。身近なもので言えば、ミニ四駆に使われているモータのことである。

図 3-2 DC モータ[20]

一般に DC モータは回転の制御がし易く、制御用モータとして非常に優れた特性を持っているといわれている。以下に具体的に優れている部分を挙げる。

1. 起動トルクが大きい
2. 入力電流に対し出力トルクが直線的に比例しかつ出力効率が良い
3. 低価格

上記のことから、制御のし易さと効率の良さ、コスト面でも優秀なことがわかる。

逆に、DC モータの欠点として、構造上ブラシとコミュテータによる機械式接点があることによって寿命があることや、マイコン制御をしようとするとノイズが走るという点があげられる。

製作物を作るうえで、効率やコスト面、機構が単純な動きで済むことやデメリットの少なさを考慮した末、DC モータの選択に至った。

3.2.3　　モータドライバについて

DC モータを回すために、モータドライバという IC を使用した。DC モータは電流を流すと回転するが、電流の方向で正回転・逆回転が決まってしまう。その電流の ON/OFF や流れる方向をマイコンから制御するための IC がモータドライバである。使用したモータドライバは、図 3-3 の Toshiba 製　TA7291P である。

図 3-3 モータドライバ[21]

このモータドライバのブロック図は図 3-4 である。

ブロック図

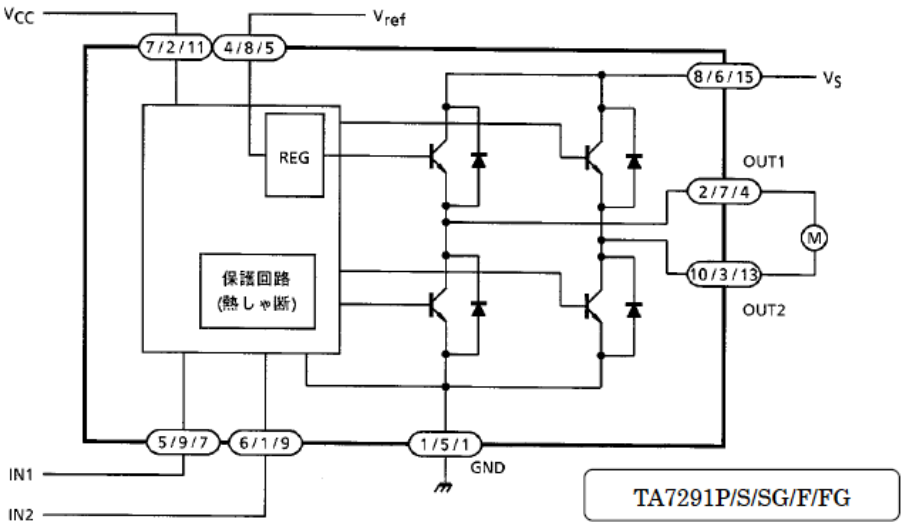
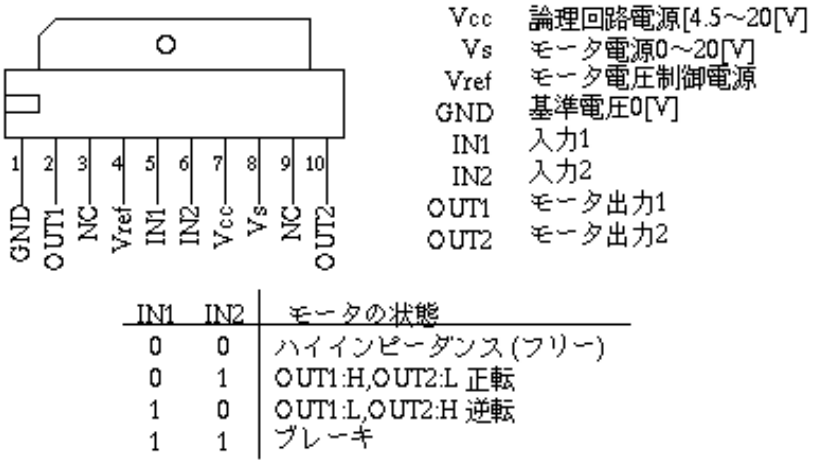


図 3-4 モータドライバブロック図[22]

このブロック図を元に回路を組み、Arduino で制御した。OUT1,OUT2 にはモータを接続し、IN1,IN2 には Arduino からの命令ピン(デジタル入力)を接続する。また、Vcc にはモータドライバ用の電源として電池ボックスを 1 つ接続する。Vs には、モータ用の電源として電池ボックスを 1 つ接続している。Vref はモータ電圧制御電源なので今回使用していない。具体的なピン配置と機能は図 3-5 である。



IN1	IN2	モータの状態
0	0	ハイインピーダンス(フリー)
0	1	OUT1:H,OUT2:L 正転
1	0	OUT1:L,OUT2:H 逆転
1	1	ブレーキ

図 3-5 ピンの配置と機能[23]

図 3-5 を見ても分かる通り、入力 IN1 と IN2 の組み合わせでモータの回転方向及び静止とブレーキを切り替えている。

3.2.4　　Arduino とサーボモータの接続とモータの特徴

Andoroid 観光望遠鏡の上下運動にはサーボモータを用いる。サーボモータと DC モータは性質が異なる。一般的に使われている DC モータは電流を流すとその電流によって回転数が変わるがこのサーボモータは位置指令によって回転角度が決まる性質がある。仕組みは図 3-6 の通りである。サーボモータのケース内に位置アンプと速度アンプがあり電圧に対して決まった角度をとるようになっている。

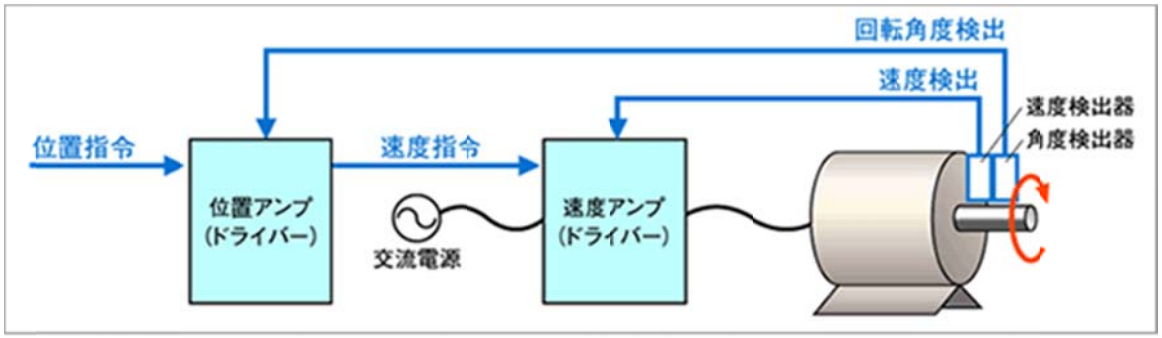


図 3-6 サーボモータの原理[24]

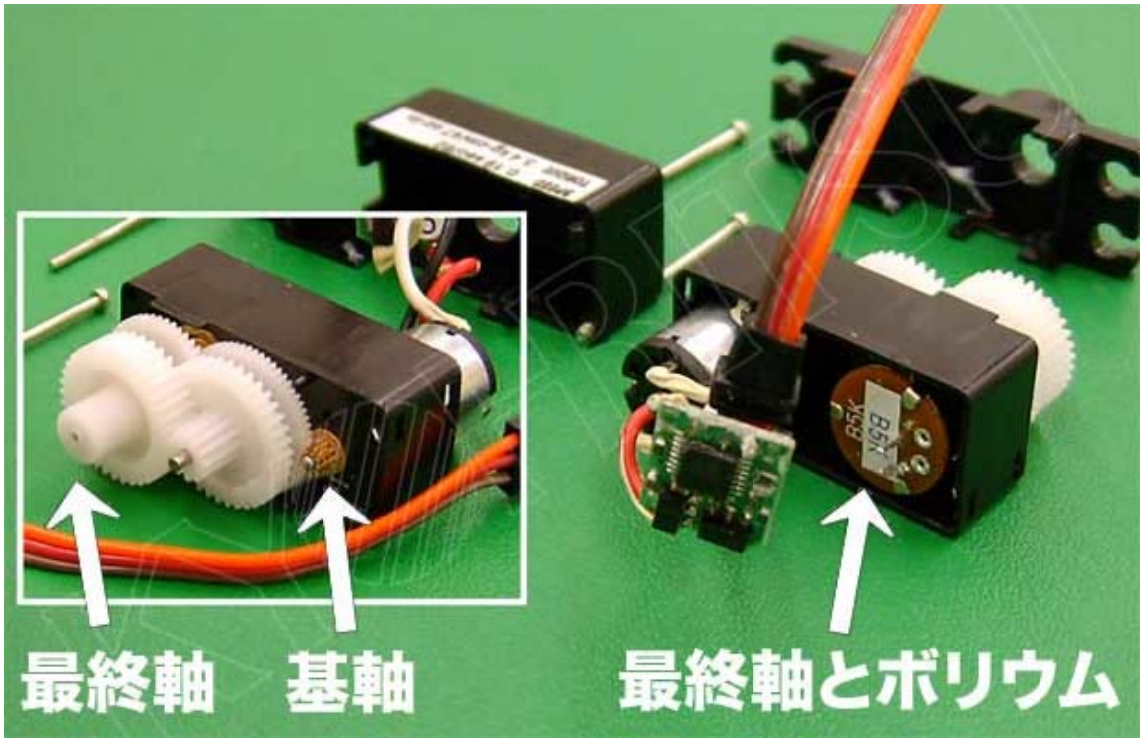


図 3-7 サーボモータの構造[25]

3.2.5　　PWM 制御

パルス信号と PMW 制御について(サーボモータ)

サーボモータは PWM 制御という手法でパルス信号により制御される。PWM とは Pulse Width Modulation の略であり、名前の通りパルス信号を利用している。パルス信号とは電気で情報を伝える方法であり、電圧や電流が継続する時間で情報を伝える。単純にここでは ON/OFF を繰り返す信号だと考えてもらえればよい。

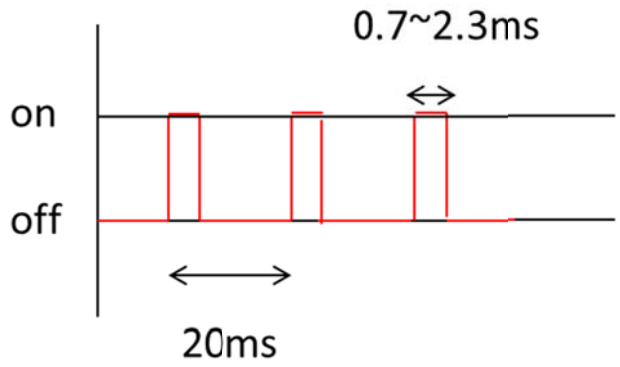


図 3-8 PWM を用いたパルス信号の波

具体的に言うと、パルスの幅を長くしたり、短くしたりしてモータの角度を制御している。

3.3　プログラム　(今井担当)

3.3.1　Arduino と DC モータを 1 つ動かすプログラム

Android 観光望遠鏡の製作にあたって、動作の確認のために DC モータを 1 つ動かすプログラムを製作した。そのプログラムを下記に載せる。

```
#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>

#define DC1_PWM 3

#define LOW (0)
#define HIGH (1)
//Android アプリ側の Accessory_filter.xml 内の属性と一致させる
AndroidAccessory acc("AndroidCamera",   //第一引数:組織名（manufacturer 属性と一致）
        "DC1",                          //第２引数：モデル名（model 属性と一致）
        "AdkLedPwm - Arduino USB Host",    //第３引数：ダイアログ表示メッセージ
        "1.0",                          //第４引数：バージョン（version 属性と一致）
        "http://accessories.android.com/",//第５引数：ジャンプ先 URL
        "0000000012345678");            //第６引数：シリアルナンバー

void setup()  //最初に一度だけ実行
{
  Serial.begin(9600);
  Serial.print("¥r¥nStart");
  pinMode(DC1_PWM,OUTPUT);  //DC1(PWM)用ピンを出力ポートにする
  //USB ホスト機能を有効にする
  acc.powerOn();
  pinMode(6,OUTPUT); //信号用ピン
  pinMode(7,OUTPUT); //信号用ピン
}

void loop()
{
```

```
  byte msg[1];
  if  (acc.isConnected())  {   //Android を起動・接続する命令を送る
    //communicate with Android application
    int len = acc.read(msg, sizeof(msg), 1); //ADK 接続から読み込み       ----------(1)

    if(len>0){                        ----------(2)

     if(msg[0]==0){                ---------- (3)
      digitalWrite(1,LOW);
      digitalWrite(2,HIGH);   //0 で正回転
     }else if(msg[0]==255){          ----------(3)
      digitalWrite(1,HIGH);
      digitalWrite(2,LOW);    //255 で反転
     }else{                              ----------(3)
      digitalWrite(1,LOW);
      digitalWrite(2,LOW);    //128 で静止
     }
    }
  }
}
```

上記のプログラムはタッチ操作のためのプログラムである。ボタンをタッチすることで Android 側からの命令を受けて、Arduino でデジタル出力の切り替えとアナログ出力を操作する。次に、Android 側から sendCommand 命令を受け取り、モータを動かす。

メッセージを受け取るところでは(1)で 1byte の msg を受け取り、(2)で受け取ったデータ長が 1byte 以上ならば命令を実行するというプログラムである。(3)では受けとったメッセージが 0、255、それ以外で、正回転・反転・静止を分けている。

3.3.2　Arduino とサーボモータを動かすプログラム

DC モータを動かすプログラムに引き続き、上下動作のためのサーボモータを動作させるプログラムを製作した。

```
#include <Servo.h>
#include <Max3421e.h>
#include <Usb.h>

//Android アプリ側の Accessory_filter.xml 内の属性と一致させる
AndroidAccessory acc("AndroidCamera",   //第一引数:組織名（manufacturer 属性と一致）
        "DC1",                          //第２引数：モデル名（model 属性と一致）
        "AdkServoPwm - Arduino USB Host",    //第３引数：ダイアログ表示メッセージ
        "1.0",                          //第４引数：バージョン（version 属性と一致）
        "http://accessories.android.com/",//第５引数：ジャンプ先 URL
        "0000000012345678");            //第６引数：シリアルナンバー
void setup()  //最初に一度だけ実行
{
  Serial.begin(115200);
  Serial.print("¥r¥nStart");
 //USB ホスト機能を有効にする
  acc.powerOn();
  servo.attach(3);

}
void loop(){
  byte msg[1];
  if  (acc.isConnected())  {  //Android を起動・接続する命令を送る
    //communicate with Android application
    int len = acc.read(msg, sizeof(msg), 1); //ADK 接続から読み込み

    int val;
    if (len > 0){
  //アナログ出力
  analogWrite(3,msg[0]); //0~255
  //0.02 秒ループにする
  delay(20);
```

```
    }
  }
}
```

上記のプログラムは、Android タブレットから送られる 0~255 のコマンドデータを元に、シークバーを使用してサーボモータを動かすことができる。最終的には、DC モータのプログラムとサーボモータのプログラムを合わせて、複数のモータを Android と Arduino で制御していく。

3.3.3　　ズーム・フォーカス・明暗部分の DC モータ

これまで述べてきたように、Android 観光望遠鏡のズーム・フォーカス・明暗の調整と左右移動に DC モータ 4 個、上下運動にサーボモータ 1 個で、合計 5 つのモータを使用することになる。これらのモータを制御するための Arduino 用プログラムについては、3.3.5 で解説する。

ズーム・フォーカス・明暗の 3 箇所を Android タブレット上で動かすシステムを製作した。ここで使用するのは 3 箇所すべて DC モータである。

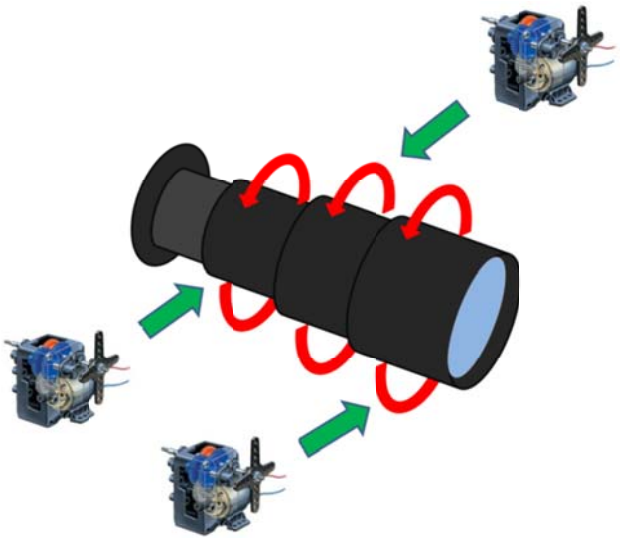


図 3-9 ズーム・フォーカス・明暗を調節するイメージ図

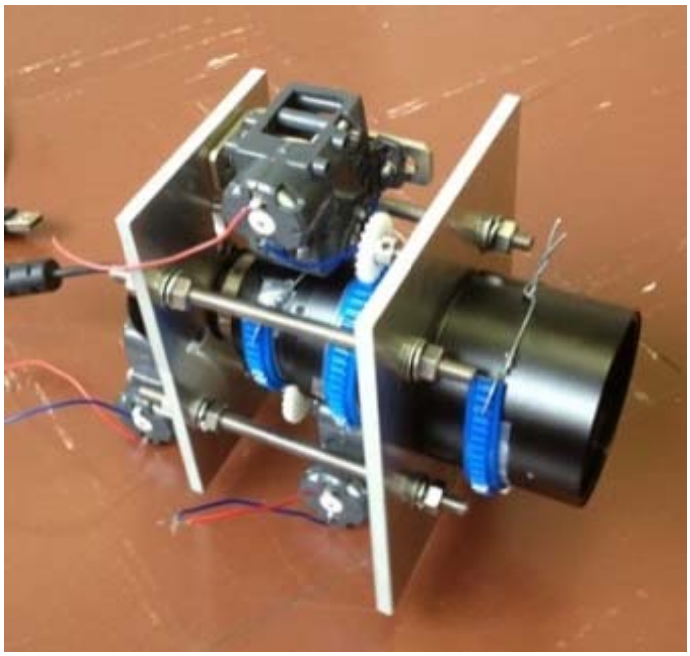


図 3-10 ズーム・フォーカス・明暗を調節するためのモータの配置

3.3.4　　上下左右の首ふり動作

フォーカスなどの 3 箇所に加え、上下にサーボモータ、左右に DC モータを 1 つずつ取り付け動かす。

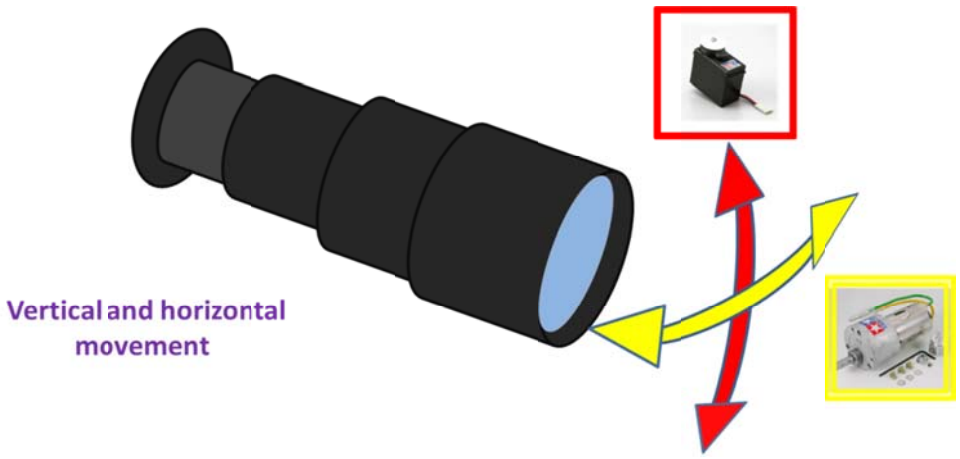


図 3-11 上下左右動作イメージ図

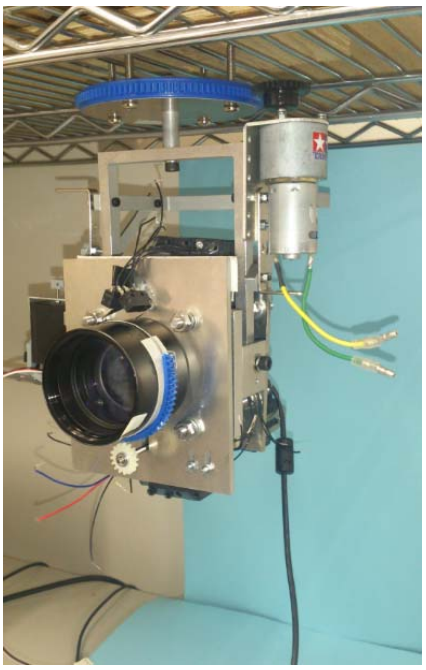


図 3-12 上下左右の動作モータ取り付け後の様子

今回は計 5 箇所のモータを動かすことになるが、送る byte 数を 2byte に増やさなければならない。理由は、1byte 目をモータの指定に使い、2byte 目をサーボモータの角度指定及び DC モータの回転方向指定に使用するためである。それに伴い、DC モータの命令も左右で 2 ついるところを、1 つにまとめた。

3.3.5　　5 箇所のモータを動かすプログラム

3.3.3 及び 3.3.4 で説明したモータを全て動かすプログラムを以下にまとめた。このプログラムについての解説を載せる。

```
#include <Max3421e.h> //ADK を利用するための 3 つのライブラリを読み込む
#include <Usb.h>
#include <AndroidAccessory.h>
int incomingByte0;          // a variable to read incoming serial data into
int incomingByte1;
#include <Servo.h>
Servo servo1;//

//Android アプリ側の accesory_filter.xml 内の属性と一致させる(2)
AndroidAccessory acc("AndroidCamera",     //第 1 引数:組織名（manufacturer 属性と一致）
        "DC1",                            //第 2 引数:モデル名（model 属性と一致）
        "DC1 - Arduino USB Host",   //第 3 引数:ダイアログ表示メッセージ
        "1.0",                                  //第 4 引数:バージョン（version 属性と一致）
        "http://accessories.android.com/", //第 5 引数:ジャンプ先 URL
        "0000000012345678");                  //第 6 引数:シリアル番号

void setup()   //最初に一度だけ実行される部分
{
  Serial.begin(9600);
  Serial.print("¥r¥nStart");
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(22, OUTPUT);
  pinMode(23, OUTPUT);
  pinMode(24,INPUT);
  digitalWrite(25,HIGH);
  servo1.attach(13);
  acc.powerOn();
```

```
}

void loop()   //繰り返し実行される部分
{
  byte msg[2];//Android から受け取るデータ

  if (acc.isConnected()) {   //Android を起動・接続する命令を送る
    //communicate with Android application
    int len = acc.read(msg, sizeof(msg), 2); //ADK 接続から読み込み

    if (len > 0) { //読み込んだデータがあれば処理する
      if (msg[0] == 0x1){ //独自プロトコルのコマンド・コマンドが 0x1 ならば処理する
       if (msg[1] == 0x1){
          digitalWrite(2, HIGH);
          digitalWrite(3, LOW);
        }
       if (msg[1] == 0x2){
          digitalWrite(3, HIGH);
          digitalWrite(2, LOW);
        }
       }
        else if(msg[0]==0x2){
       if (msg[1] == 0x1){
            digitalWrite(8, HIGH);
            digitalWrite(9, LOW);
        }
       if (msg[1] == 0x2){
            digitalWrite(8, LOW);
            digitalWrite(9, HIGH);
        }
       }
        else if(msg[0]==0x3){
         if (msg[1] == 0x1){
          digitalWrite(6, LOW);
          digitalWrite(7, HIGH);
```

```
        }
        if (msg[1] == 0x2){
          digitalWrite(6, HIGH);
          digitalWrite(7, LOW);
        }
      }
      else if(msg[0]==0x5){
        if (msg[1] == 0x1){
          digitalWrite(22, HIGH);
          digitalWrite(23, LOW);
        }
        if (msg[1] == 0x2){
          digitalWrite(22, LOW);
          digitalWrite(23, HIGH);
        }
      }
      else if(msg[0]==0x4){
          int i=map(msg[1],0,255,0,180);
          servo1.write(i);
        }
      else {
      digitalWrite(2, LOW);//LED を消灯する(6)
      digitalWrite(3, LOW);
      digitalWrite(8, LOW);
      digitalWrite(9, LOW);
      digitalWrite(6, LOW);
      digitalWrite(7, LOW);
      digitalWrite(22, LOW);
      digitalWrite(23, LOW);
    }
  }
}
```

上記のプログラムの解説をする。まず、Command 1byte と value　1byte の 2byte データを受け取る。Command 部分では 0x1~0x3 でズーム・フォーカス・明暗に使用している DC モータを指定、0x5 で左右の DC モータを指定する。使用するモータを指定した後に、value で正転か反転の指定をする。プログラム上では、0x1 で正転、0x2 で反転である。Command が 0x4 の部分ではサーボモータの動作する角度決めをしている。

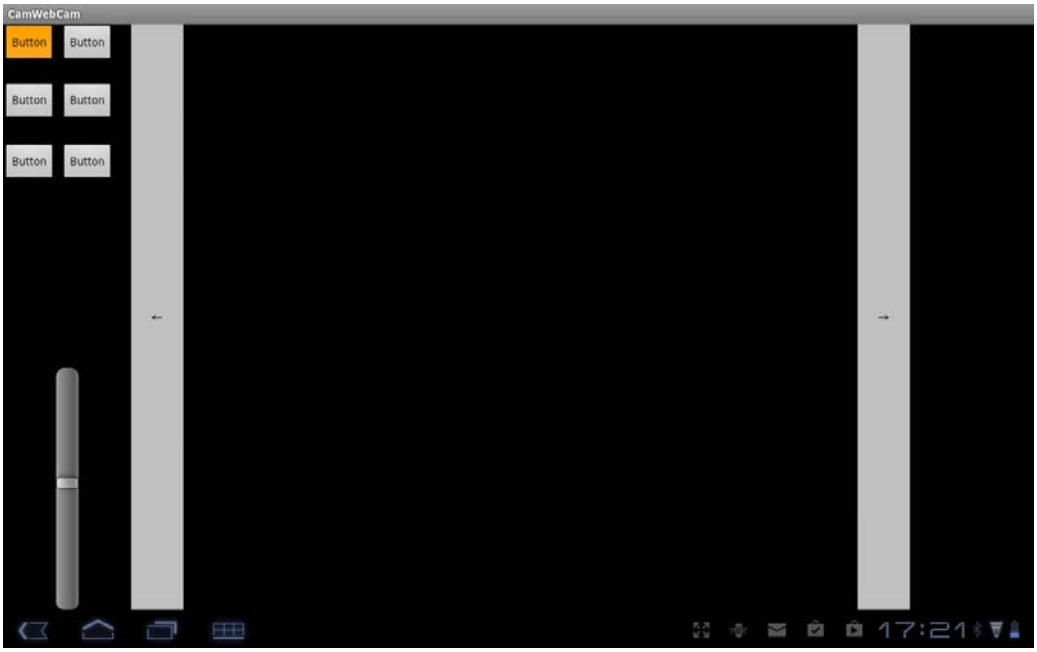


図 3-13 5 箇所動作時のタブレット画面

このプログラムにより、全てのモータを動かすことに成功した。これらを機構に取り付け、カメラを動かす。

3.4　　安全装置(プログラム)　　(今井担当)

これまで述べてきたように、本システムでは 4 箇所に DC モータを用いるが、DC モータは命令を与えている間回転し続けてしまい、回し続けると機構の破壊を招いてしまうことがある。そこで、機構の安全性を考えた制御システムが必要になる。そのために、いくつかのセンサを試し、本研究に適している物を見つけていく。

3.4.1　　センサ(制御用)

図 3-14 赤外線センサ[26]

赤外線センサは、赤外線を投光部から信号光として発射した後、検出物体によって反射する光を受光部で検出し、遮光される光量の変化を受光部で検知、出力信号を得るものである。

図 3-15 マイクロスイッチ[27]

マイクロスイッチは、物体が金属部分に触れたとき、機械的にスイッチが入る機械式のスイッチである。機械的な信号を電気的な信号に変換し、接点の動きでオン・オフの切り替えが可能である。主に緊急停止、シャットダウン用に使われる。物体を検出するセンサとしては、機構が単純でわかりやすいため扱いやすい。

上記で挙げたマイクロスイッチと赤外線センサについて、実際に Arduino を用いて使用可能かどうか、また、私達の研究に適したセンサはどちらかを実験した。

3.4.2　　実験

3.4.2.1　　赤外線(フォトインタラプタ)で物体を検知する

まず、Arduino を用いて赤外線で物体を検知する実験を行ってみた。今回の実験で使用した物は、反射型フォトインタラプタ「RPR-220」(ROHM 製)である。反射型であるため、赤外線とフォトトランジスタはそれぞれ上面に向けて配置されている。赤外線 LED は常時点灯しており、フォトインタラプタの上面に反射物があると、赤外線が反射され、その反射光がフォトトランジスタに当たり ON の状態になる。回路は ON/OFF の切り替えで LED が点灯/消灯するように組む。

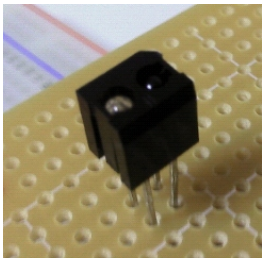


図 3-16 反射型フォトインタラプタ[28]

Arduino テスト用のプログラムは以下の通りである。

```
// フォトインタラプタテスト
#define Sensor 2 // フォトインタラプタのピン
#define LED 13// Arduino 基板上の LED
void setup(){
  // I/O 設定
  pinMode(Sensor,INPUT);
  pinMode(LED,OUTPUT);

  digitalWrite(LED,LOW); //LED 消灯
}
void loop(){
  //フォトインタラプタの信号が HIGH なら LED に HIGH を出力
  digitalWrite(LED,digitalRead(Sensor));
}
```

実験結果

赤外線センサは物体を検出することができ、任意の時(スイッチ切り替え時)に LED を点灯/消灯させることに成功した。反応もよく、物体を検知する範囲が想定より広かったが、外の明るさによって反応する範囲が変わったり、物体の影に反応してしまったりなど安定性が低かった。

3.4.2.2　　マイクロスイッチで物体を検知する

次にマイクロスイッチで物体を検知する実験を行った。今回の実験で使用したマイクロスイッチは「MQS-54-5L」(神明)である。1 回路 2 接点になっており、物体を検出(スイッチを押す)ことでスイッチの ON/OFF が切り替わる。スイッチが ON(押されている)時に LED が点灯、スイッチが OFF(押されていない)時は消灯となるように回路を組んだ。

図 3-17 マイクロスイッチ[19]

Arduino テスト用プログラムは以下

```
void setup() {

　　pinMode(2,INPUT) ;　　//スイッチに接続ピンをデジタル入力に設定
　　pinMode(13,OUTPUT) ;　//ＬＥＤに接続ピンをデジタル出力に設定
}
void loop() {
　　if (digitalRead(2) == HIGH) {　　//スイッチの状態を調べる
　　　　digitalWrite(13,HIGH) ;　　//スイッチが押されているなら LED を点灯
　　} else {
　　　　digitalWrite(13,LOW) ;　　//スイッチが押されていないなら LED を消灯
　　}
 }
```

実験結果

マイクロスイッチも物体を検出することができ、任意の時に LED を点灯/消灯させることに成功した。特に、マイクロスイッチは想定通りの動きであり、スイッチが触れている間だけ OFF になるという機能を完璧にこなしていた。

3.4.3　考察

上記の実験結果を基に、本研究でどちらを使用すべきか考えた。今回は機構の安全性を考えたシステムのため、確実性が第 1 優先となる。したがって、安定性が低い赤外線センサは廃案とし、マイクロスイッチを使用していくことに決定した。

3.4.4　マイクロスイッチで DC モータを止める

マイクロスイッチで LED の点灯/消灯するプログラムと回路を基に、DC モータをスイッチの切り替えで止めるシステムのサンプルを作った。使用しているのは DC モータ 1 つにマイクロスイッチ 2 個(実際に使用するときのモータ 1 箇所分)である。また、今回は可変抵抗器を使用してモータを回している。プログラムは以下の通りである。

```
void setup(){
  pinMode(4,OUTPUT); //信号用ピン
  pinMode(5,OUTPUT); //信号用ピン
  pinMode(6,INPUT) ;
  pinMode(7,INPUT) ;
}

void loop(){
  //アナログ入力:0 番ピンの値を 2 で割る
  int val=analogRead(0)/2; //0~511 の値にする

  //静止／正転／逆転の状態に分けてプログラムする

  if(val>=255 && val<=256){ //静止:255~256
    //LOW,LOW でデジタル出力
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);

  }else if(val>256){          //正転:257~511
    //HIGH,LOW でデジタル出力
    digitalWrite(4,HIGH);
    digitalWrite(5,LOW);

  }else if(val<255){          //逆転:0~254
    //LOW,HIGH でデジタル出力
    digitalWrite(4,LOW);
    digitalWrite(5,HIGH);
  }
  if(digitalRead(6) == HIGH){ ----------(1)
    digitalWrite(4,LOW);
```

```
    digitalWrite(5,LOW);
  }
 if(digitalRead(7) == HIGH){ ----------(2)
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);
  }
}
```

上記のプログラムを用いることにより、Arduino に接続して、スイッチを切り替えることで強制的にモータの回転を止める事に成功した。以下プログラムについて解説する。電圧(val)の値によって DC モータの正転・逆転・静止をコントロールする。マイクロスイッチは(1)と(2)の部分にそれぞれ 1 つずつ使用し、マイクロスイッチが押されたときにストップの命令を出す。実際にはこのシステムを基に 3 セット作り、機構を破壊しないようにする。

3.4.5　マイクロスイッチの応用

マイクロスイッチのスイッチによりで DC モータを止める事は成功した。これを「DC モータの動きを可動範囲内に制限する」ように変更するため。回路でモータを止めてから反対側にしか回らないようにした。その、プログラムを組み、ズーム・フォーカス・明暗の 3 部分についてマイクロスイッチに適用した。以下でそのプログラムを載せ解説する。

```
if (msg[1] == 0x1 && motorMoving1!=0 ){ // マイクロスイッチ 1 の方向なら止める
   motorMoving1 = 0;
   digitalWrite(2, LOW);
   digitalWrite(3, LOW);
}                                          ----------(1)
if (msg[1] == 0x2){ // マイクロスイッチ 1 の方向でなければ動かす
   motorMoving1 = 2;
   digitalWrite(2, LOW);
   digitalWrite(3, HIGH);                  ----------(2)
}
```

上記のプログラムを解説する。これは 0x1 方向にあるマイクロスイッチ 1 が押された時に実行されるコードである。その方向に DC モータを動かそうとしても、(1)の if 文で DC モータが静止する。(2)のように反対側にモータが回るボタンを押したときのみモータが動作する。このようにすることでモータの可動範囲を制限することが可能となる。

3.5　　レスポンス(操作性)の向上　　(今井担当)

モータを動かすプログラムの完成後、様々な人にシステムを評価してもらった。その中の意見に「使用者の体感と映像にずれがある」というものがあった。ボタンを押し、ピントが合ったら指を離す。単純な動作だが、実際に Android 観光望遠鏡で体感してみるとなかなか難しいというのがわかった。ボタンを押すまではいいのだが、指を離してからもピント調節が動き続けるためである(図 3-18)。

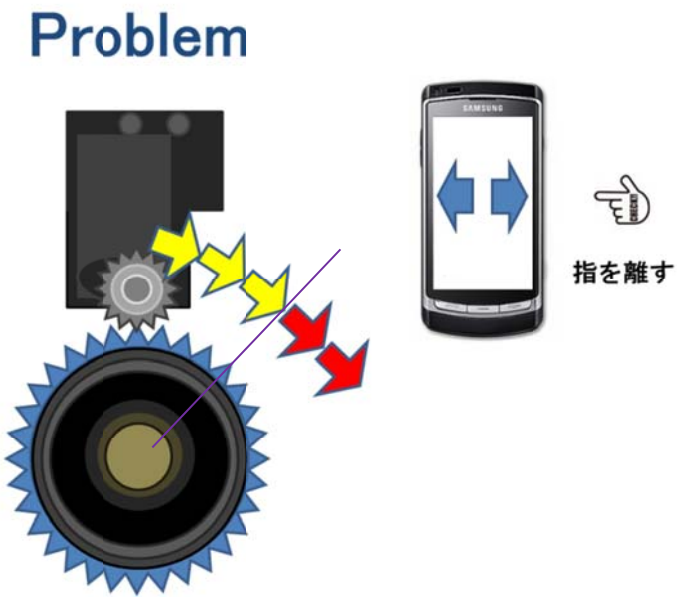


図 3-18 指を離してもモータは回り続ける

この体感のずれの原因を調査した結果、DC モータのトルクが小さいためストップの命令を出しても、慣性によりモータが回り続けてしまうことが原因だとわかった。そこで、モータドライバのブレーキ機能を使用したが、それでも改善はみられなかった。

この問題の解決策として、モータ停止の際に「静止」命令ではなく「静止」→「逆転」→「静止」と命令を送ることにした。その結果、ギアと映像ともに今までより遥かに正確に静止させることが可能になった。以下で修正前と修正後のプログラムを比較として載せ、解説する。

1. 修正前のモータブレーキ命令部分のプログラム

```
digitalWrite(2, LOW);
digitalWrite(3, LOW); ----------(1)
```

2. 修正後のモータブレーキ命令部分のプログラム

```
else{ // 静止命令  msg[0] = 9
      if(motorMoving1 != 0){ //正転・反転→静止ならば
            //LOW,LOW でデジタル出力
        digitalWrite(2,LOW);
        digitalWrite(3,LOW); ----------(2)
        delay(20);

      if(motorMoving1 == 1){//正転→静止ならば
        digitalWrite(2,LOW);
        digitalWrite(3,HIGH); ----------(3)
         delay(130);
        }
      if(motorMoving1 == 2){//反転→静止ならば
        digitalWrite(2,HIGH);
        digitalWrite(3,LOW); ----------(3)
         delay(130);
        }
       }
         digitalWrite(2,LOW);
         digitalWrite(3,LOW); ----------(4)
       }
```

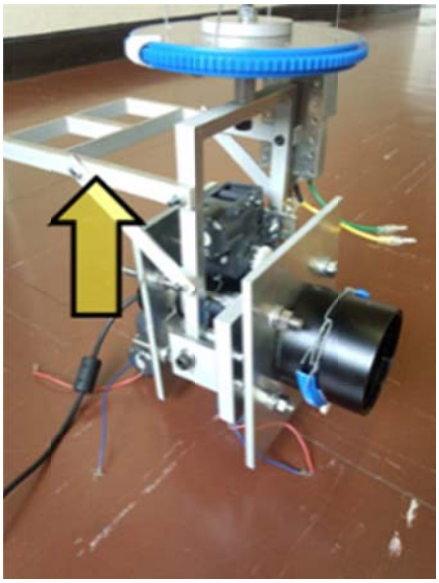
上記のプログラムについて解説する。今までのプログラムでは、DC モータが(1)で静止するだけのプログラムだった。修正後のプログラムでは、回転しているモータがマイクロスイッチに触れたとき、(2)でまず静止し、次に(3)で反転、最後にもう一度(4)で静止するようになっている。このプログラムより体感のずれはかなり軽減され、操作性を向上させることができた。

なお、ここまでのプログラムをまとめた最終的なプログラムは別紙付録に記載する。

3.6　　基板への実装　　(今井担当)

3.6.1　　ユニバーサル基板への回路の実装

実際に屋外で望遠カメラを使用することを想定した時、省スペース化や配線が絡まらないことに配慮しなくてはならない。そこで、図 3-19(右)のユニバーサル基板上に回路を実装することにした。既に機構部分で回路を乗せるスペースは製作してあるため、そのスペースに収まるよう製作する。



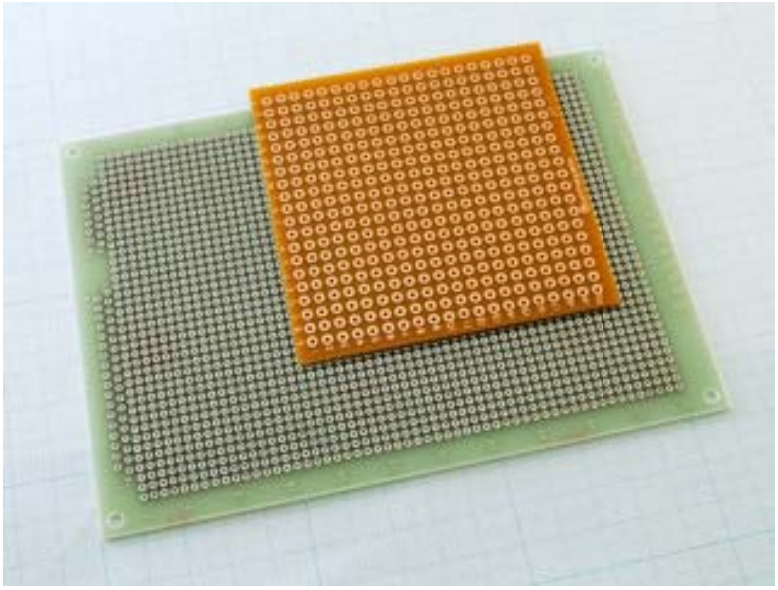


図 3-19 基板を乗せるスペース(左)と使用するユニバーサル基板(右)[29]

回路を基板に実装した結果が図 3-20 である。

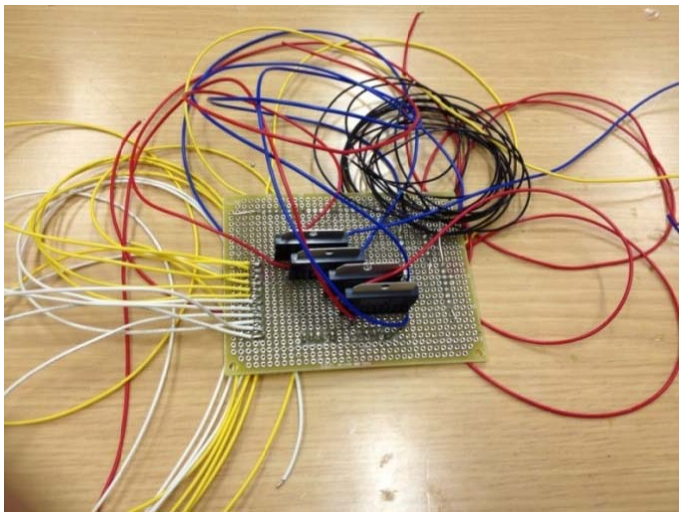


図 3-20 基板プロトタイプ

この段階では、導線が全て基板から伸びているので動作確認しかできない。そこで、この基板にコネクタを使用する。導線は全てカメラの方から伸ばし、基板にはコネクタを挿すだけで使用可能にした。実際に製作したのが図 3-21 である。

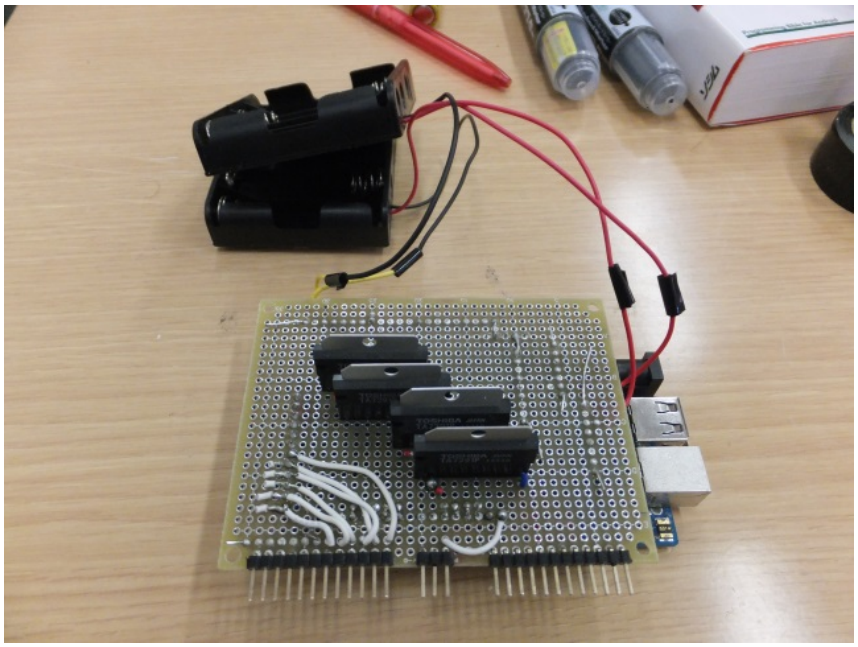


図 3-21 基板

図 3-21 の基板を Android 観光望遠鏡に乗せた様子は図 3-22 のようになる。

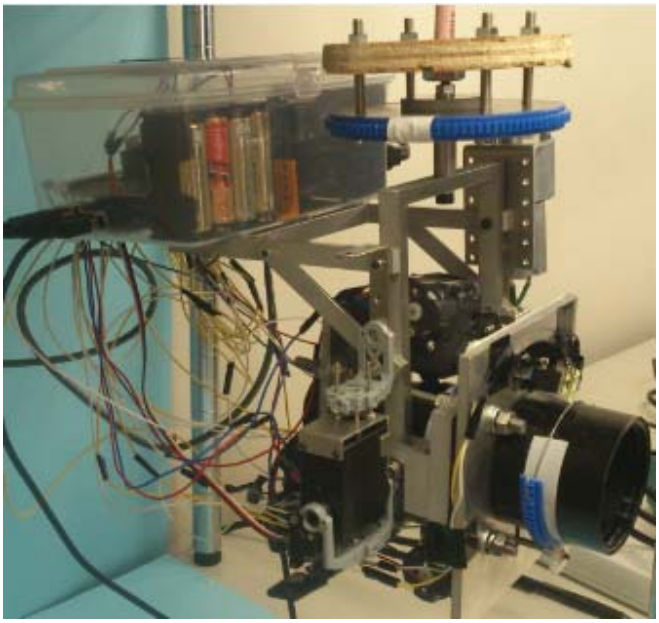


図 3-22 製作物に基板を乗せた様子

3.6.2　回路の接続

完成したプログラムを使用して、動作確認を行った(詳しくは 5 章で説明する)。ここでは、その時に使用した回路の接続について詳しく説明する

図 3-21 で使用している図がその回路にあたる。まず電源の供給だが、Arduino の電源に電池ボックスを 1 つ使用した。また、ズーム・フォーカス・明暗の 3 箇所に設置されているモータドライバの Vs と Vcc に電池ボックスを 1 つ使用した。左右運動の DC モータには必要電圧の関係で DC モータ 1 つに対し電池ボックスを 1 つ使用した。サーボモータは Arduino から電源を供給している。ここまでで、この回路を動作させるために計 3 つの電池ボックスを使用している。

次に使用しているピンとそれに対応する回路を表 3-1 に、電池ボックスと電源供給部分を表 3-2 にまとめた。

表 3-1 Arduino ピンと回路対応表

Arduino ピン番号	回路
2~5,15~18	DC モータの INPUT
22,24,26,28,30,32	マイクロスイッチの INPUT ピン
34,36,38,40,42,44	マイクロスイッチの GND ピン
7	サーボモータの命令ピン
5V 出力ピン(Arduino)	全てのモータドライバの Vcc サーボモータの電源
GND ピン	モータドライバ・サーボモータの GND マイクロスイッチの GND

表 3-2 電池ボックスを使用している部分

電池ボックス	電源供給部分
電池ボックス 1(単 3 電池×4)	Arduino 本体の電源(USB 接続)
電池ボックス 2(単 3 電池×4)	ズーム・フォーカス・明暗部分の Vs
電池ボックス 3(単 3 電池×4)	左右運動部分の Vs

表 3-1 の通りに回路と Arduino ピンを接続し、実際に動作実験を行った。

3.6.3　　電源の変更

既に述べたように、Arduino 及びモータの電源に、電池ボックスを複数使用している。しかし、実際の使いやすさや望遠カメラの総重量を考えると、この状況は好ましくない。そこで、電池ボックスを全て取り除き、まとめてコンセントの方から電源を供給できるように回路に手を加えた。

3.6.3.1　　3 端子レギュレータの使用

電源には安定した供給が必要なため、電源レギュレータ IC 等使用して安定した電圧にする必要がある。そこで、本研究では 3 端子レギュレータを使用することにした。使用したレギュレータは図 3-23 の Toshiba 製 TA4805S である。

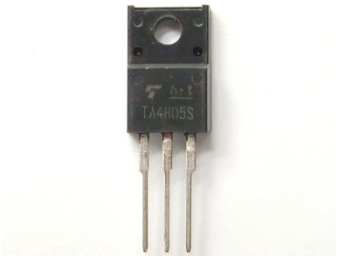


図 3-23 3 端子レギュレータ[30]

この 3 端子レギュレータの回路図は図 3-24 である。

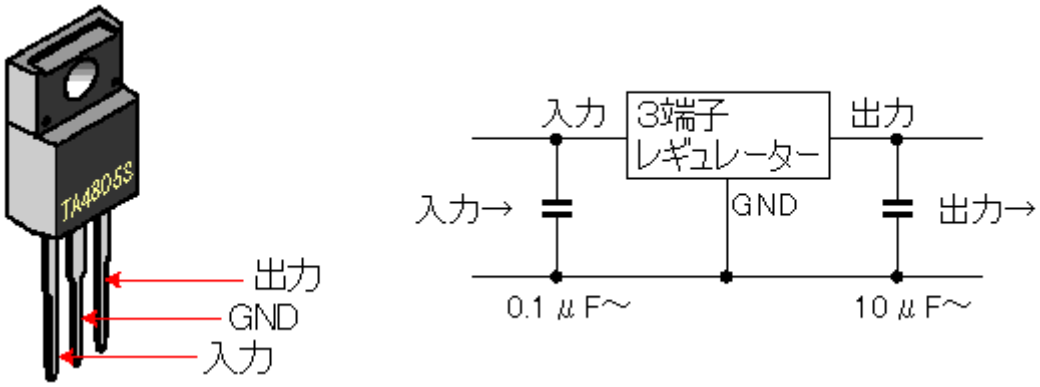


図 3-24 3 端子レギュレータ簡易回路図[31]

本研究では AC アダプタの出力が 12V の物を使用する。従って、図 3-24 の入力部は 12V となる。また、3 端子レギュレータからの出力は 5V となる。GND は回路の GND に接続する。

3.6.3.2　　コンセント電源の実装

本研究で使用していた回路に、コンセント電源を加える。電源は 1 つにまとめることができるので、使用していた電池ボックスは全て取り外した。また、レギュレータと DC ジャックを新たに回路に組み込んだ結果、表 3-2 の電源部分は表 3-3 のようになる。なお、モータに使用しているピンは変わらない。

表 3-3 コンセント実装後の電源対応表

電源から	電源供給部分
12V 出力	Arduino 本体の電源(Vin ピン)
5V 出力	各 DC モータの Vs
5V 出力(Arduino)	全てのモータドライバの Vcc サーボモータの電源

実際に電源実装後の様子が図 3-25 である。

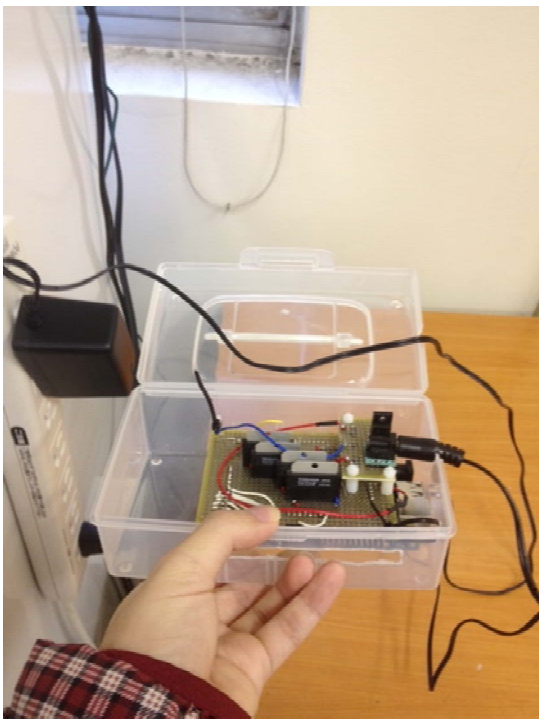


図 3-25 電源実装後の回路

電源をボックスからコンセントに変えたことで、扱いやすさの向上及び総重量の軽減を可能とした。また、安定した電源の供給という目的を果たした。

第 4 章　　Android　アプリケーション

（濵内担当）

本章では Android タブレットで使用するアプリケーションの作成について説明する。

4.1　　概要　　（濵内担当）

4.1.1　　目的

カメラアプリケーションの目的は、Arduino に信号を送りモータを動かすことで利用者の見たい映像を Android タブレットに表示することである。図 4-1 はシステムの簡略図である。

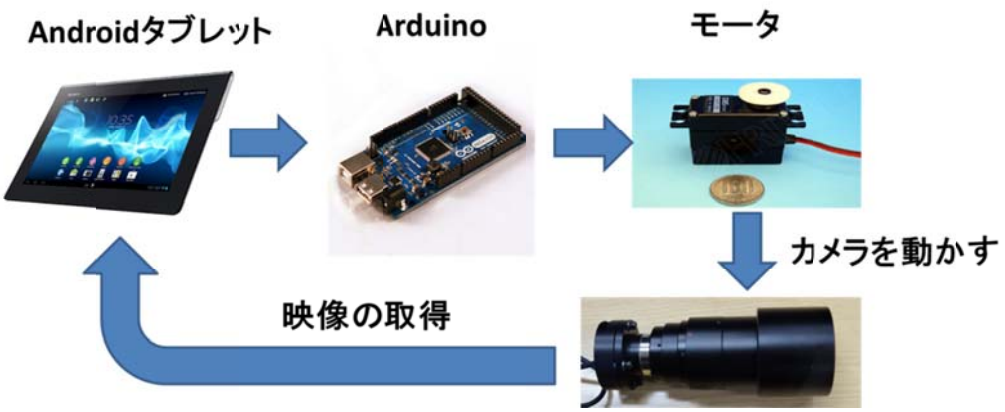


図 4-1 システム簡略図[32]

4.1.2　　必要な機能

アプリケーションに必要な機能は以下の通りである。

1.望遠カメラの映像をタブレットに映す。

2.ズーム・フォーカス・明暗を制御する DC モータを動かすための命令を Arduino に送る。

3.左右方向を制御する DC モータを動かすための命令を送る。

4.上下方向を制御するサーボモータを動かすための命令を送る。

今回望遠カメラの映像を映すアプリケーションはインターネットで公開されているものをベースとして用いる。アプリケーション名は SimpleWebCam であり、これに命令送信機能を追加していく(SimpleWebCam の URL は参考文献[33]に記載)。図 4-2 がそのアプリケーションの実行画面である。

図 4-2 SimpleWebCam

4.2　Arduinoとの通信　（濵内担当）

4.2.1　概要

本アプリケーションではマイクロUSB接続によりAndroidタブレットからArduinoに命令を送り、Android観光望遠鏡を操作する。通信方式はAndroid2.3.4および3.1以降で追加されたADK(Android Open Accessory Development Kit)を用いる。ADKとはAndroid向け周辺機器を誰でも開発できる仕組みと、その開発に必要なハードウェア、ソフトウェア一式のことである[34]。

機構に使われているモータは4.1.2で説明したDCモータ4つ、サーボモータ1つである。この5つのモータを制御するためにArduinoへ命令を送るプログラムを作る。

開発環境としてeclipse用いてJava言語で開発を行った。

4.2.2　Arduinoとの通信を行うプログラム

ADKを用いてArduinoと通信をするためには図4-3の黒枠のコードをAndroid Manifestに加える必要がある。Android ManifestとはAndroidアプリケーションを実行するにあたって必要な情報を記載、宣言する場所である。

更に図4-4にあるaccessory_filterをxmlフォルダに生成し、黒枠で囲ってあるmanufacturer・model・versionをArduinoのコードと同じものにする。さらに図4-5に書かれているコードをActivityに書く必要がある。Activityはアプリケーションの1画面に相当する。

*TKCameraWebCam Manifest

```
<application
    android:icon="@drawable/icon"
    android:label="@string/app_name" >
    <uses-library android:name="com.android.future.usb.accessory" />

    <activity
        android:name=".Main"
        android:label="@string/app_name"
        android:screenOrientation="landscape" >
        <intent-filter>
            <action android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <meta-data
            android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
            android:resource="@xml/accessory_filter" />
    </activity>
    <activity android:name="CameraPreview" >
    </activity>
    <activity android:name="JoystickView" >
    </activity>
    <activity android:name="ServoControl" >
    </activity>

    <uses-library android:name="com.android.future.usb.accessory" />

    <activity android:name="MainActivity" >
    </activity>
    <activity android:name="AdkLedOnOffProjActivity" >
    </activity>
</application>
```

図 4-3 Android Manifest

accessory_filter.xml

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
    <!-- ADKボードのファームウェアで指定した名前と一致させる -->
    <usb-accessory manufacturer="AndroidCamera" model="DC1" version="1.0" />

</resources>
```

図 4-4 accessory_filter

Main.java

```
    private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            // TODO 自動生成されたメソッド・スタブ
            String action = intent.getAction();
            if (ACTION_USB_PERMISSION.equals(action)) {
                synchronized (this) {
                    // Intent からアクセサリを取得
                    UsbAccessory accessory = (UsbAccessory) intent
                            .getParcelableExtra(UsbManager.EXTRA_PERMISSION_GRANTED);

                    // パーミッションがあるかチェック
                    if (intent.getBooleanExtra(
                            UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                        // 接続を開く
                        openAccessory(accessory);
                    } else {
                        Log.d(TAG, "permission denied for accessory "
                                + accessory);
                    }
                    mPermissionRequestPending = false;
                }
            } else if (UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
                // Intent からアクセサリを取得
                UsbAccessory accessory = (UsbAccessory) intent
                        .getParcelableExtra(UsbManager.EXTRA_PERMISSION_GRANTED);

                if (accessory != null && accessory.equals(mAccessory)) {
                    // 接続を閉じる
                    closeAccessory();
                }
            }
        }
    };
```

図 4-5 Activity

4.2.3　　DC モータを動かす

4.2.2 で Arduino との通信が可能になったため、DC モータを動かすための信号を Arduino へ送るプログラムを作成する。手順として SimpleWebCam とは別のアプリケーションでモータの制御を行い、5 つのモータ全ての動作を確認した後、SimpleWebCam と組み合わせる。初めに 1 つのモータを動かせるようにし、その後 3 つのモータを動かせるようにする。まずボタンを 2 つ作り、このボタンを押すことで正転、反転を制御できるようにする。図 4-6 が 1 つの DC モータを制御するための試作アプリケーションである。

図 4-6 1 つの DC モータを制御するためのアプリケーション

図 4-6 の left のボタンが押された時、図 4-7 のように回路に sendCommand(0)という命令が実行されモータが回転するようにし、同様に right のボタンが押された時に sendCommand (255)という命令が実行されるようにする。またボタンを離した時に sendCommand(128)という信号を送り停止させる。回路での処理のプログラムは 3.3.1 で説明している。

Main1Activity.java

```
        mleft.setOnTouchListener(new OnTouchListener() {
            public boolean onTouch(View v, MotionEvent event) {
                int action = event.getAction();
                byte command = (byte) 128;
                switch (action) {
                case MotionEvent.ACTION_DOWN: // 押しはじめ時
                    // comand = (byte)ox1;
                    // value = (byte)o;
                    // sendcomamand(comand, value);
                    command = (byte) 0;
                    sendCommand(command);
                    break;
                case MotionEvent.ACTION_UP: // 離した時
                    command = (byte) 128;
                    sendCommand(command);
                    break;
                }
                return true;
            }
        });

        mright = (Button) findViewById(R.id.right);

        mright.setOnTouchListener(new OnTouchListener() {
            public boolean onTouch(View v, MotionEvent event) {
                int action = event.getAction();
                byte command = (byte) 128;
                switch (action) {
                case MotionEvent.ACTION_DOWN: // 押しはじめ時
                    command = (byte) 255;
                    sendCommand(command);
                    break;
                case MotionEvent.ACTION_UP: // 離した時
                    command = (byte) 128;
                    sendCommand(command);
                    break;
                }
                return true;
            }
        });
```

図 4-7 DC モータ コード

次にズーム・フォーカス・明暗を制御するための 3 つの DC モータを動作させるために、ボタンを 6 つもつプログラムを作成する。そのボタンは図 4-8 である。

プログラム上で分かりやすいよう、モータ 1 つ 1 つに 0x1~0x3 と名前をつけた。また、モータの正転を 0x1、反転を 0x2 とした。例を挙げると sendCommand(0x1,0x2)を実行したとき、モータ 0x1 が反転する。また sendCommand(0x9,0x9)を実行した時全てのモータが停止するようにした。回路での処理は 3.3.5 で説明している。図 4-9 はこのプログラムのうちモータ 1 つに対応する部分である。なお、ボタンを押している間、一定間隔で命令が送られ続けるよう、タイマー機能を用いている。以上でズーム・フォーカス・明暗の 3 つの DC モータを制御できるようになった。

図 4-8 3 つの DC モータを動かすための試作アプリケーション

Main.java

```
        a.setOnTouchListener(new OnTouchListener() {
            public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                    timer_a = new Timer();
                    timer_a.scheduleAtFixedRate(new TimerTask() {
                        @Override
                        public void run() {
                            byte command = (byte) 0x1;
                            byte value = (byte) 0x1;
                            sendCommand(command, value);
                        }
                    }, 0, INTERVAL_PERIOD);

                    break;
                case MotionEvent.ACTION_MOVE:
                    break;
                case MotionEvent.ACTION_UP:
                    if (timer_a != null) {
                        timer_a.cancel();
                    }
                    command = (byte) 0x9;
                    value = (byte) 0x9;
                    sendCommand(command, value);

                    break;
                case MotionEvent.ACTION_CANCEL:

                    break;
                }

                return false;
            }
        });
```

図 4-9 3 つの DC モータを動かすためのコード。1 つのモータに対応する部分のみ示す。

4.2.4　　サーボモータを動かす

次に上下方向に動作させるためにサーボモータを制御する。サーボモータは DC モータと信号の種類が異なるので、ボタンを押している間 0~255 の値を送り続けることで角度調節を行うことを試みた。これに 4.2.3 と同様の方法で左右を制御するためのボタンとプログラムを追加し、SimpleWebCam と組み合わせたのが図 4-10 である。

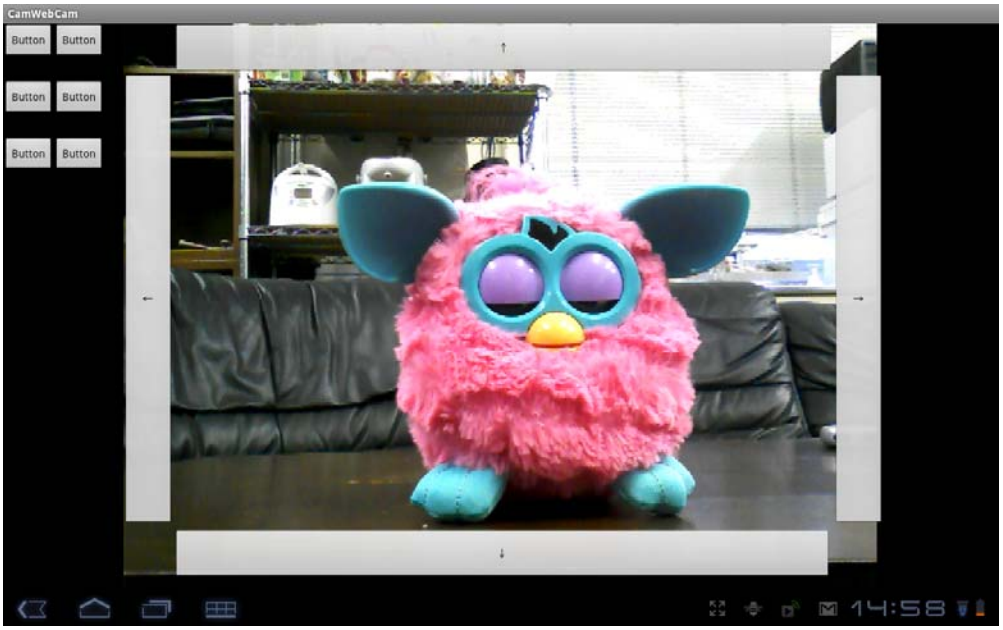


図 4-10 Camera アプリケーション

図 4-10 のアプリケーションですべての動作を確認することができた。しかしアプリケーション上で上下のボタンを押した時、信号を送り続けることが原因で回路での処理が間に合わず、動作しない時があった。そこで角度の数値を送るシークバーに変えたところ解決できた。そのアプリケーションの図が図 4-11、プログラムが図 4-12 である。

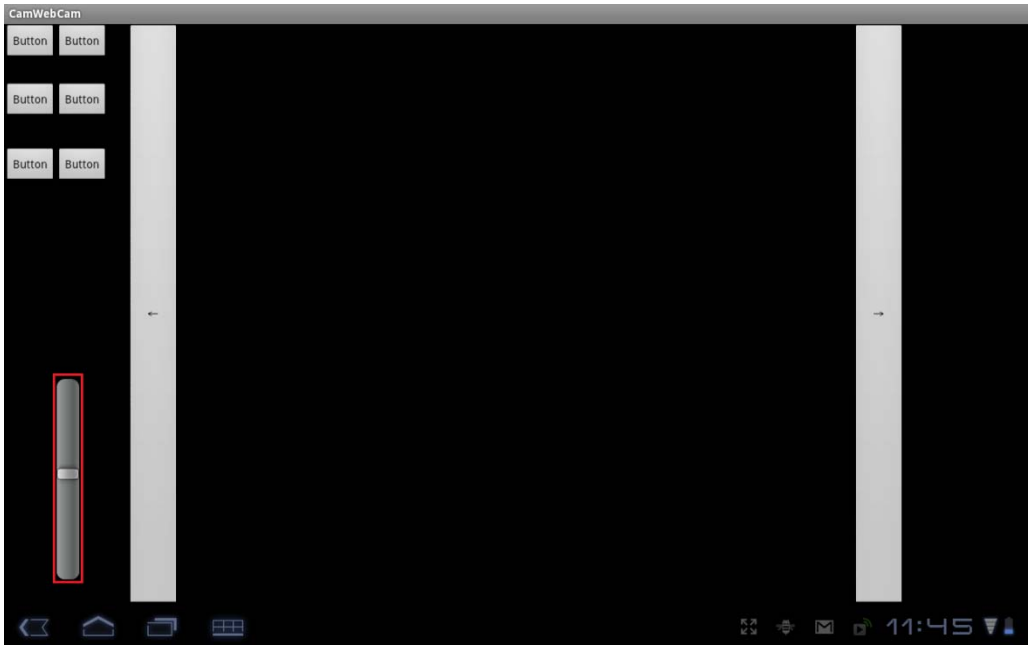


図 4-11 シークバー

Main.java

```
        mSeekBar.setOnSeekBarChangeListener(new VerticalSeekBar.OnSeekBarChangeListener() {

            public void onProgressChanged(VerticalSeekBar seekBar,
                    int progress, boolean fromUser) {
                // TODO 自動生成されたメソッド・スタブ
                if (progress > 0 & progress < 255) {
                    byte command = (byte) 0x4;
                    // valueで値を決める
                    byte value = (byte) progress;
                    // データをsendCommandに送る

                    sendCommand(command, value);
                } else {
                }
            }

            public void onStartTrackingTouch(VerticalSeekBar seekBar) {
                // TODO 自動生成されたメソッド・スタブ

            }

            public void onStopTrackingTouch(VerticalSeekBar seekBar) {
                // TODO 自動生成されたメソッド・スタブ

            }
        });
```

図 4-12 シークバー コード

4.3　アプリケーションのレイアウト　(濵内担当)

4.3.1　左右ボタン

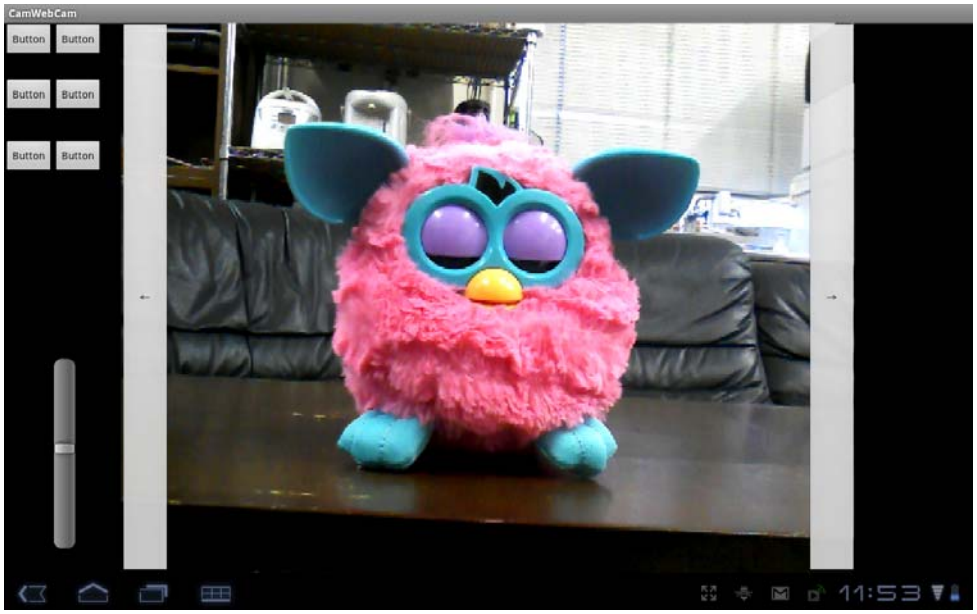


図 4-13 左右ボタン

ここからは使い手にとって使いやすいアプリケーションにするためのレイアウトについて考える。

これまでに作成したアプリケーションは図 4-13 のように左右のボタンが邪魔で映像の範囲が制限されている。そこで、この左右のボタンを透明にした。res フォルダの drawable にボタンの色を指定する xml ファイルを作成することで透明にできる。図 4-14 がその color_stateful.xml である。8 桁の数字で色を指定することができ、上位 7,8 桁目で透明度を指定することが出来る。完全な透明にしてしまうとボタンがどこにあるのか分からなくなってしまうので、うっすら見える程度に透明にすることを考え、透明度の数値は 0x30 とした。またこのファイルでボタンを押した時の色も指定することが出来るのでボタンを押した時はオレンジ色(#FF8C00)になるように設定した。

*color_stateful.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:state_pressed="true"><color android:color="#FF8C00" />
    </item>
    <item android:state_pressed="false"><color android:color="#30C0C0C0" />
    </item>

</selector>
```

図 4-14 左右のボタンを透明にするためのコード

この color_stateful.xml を設定し、左右ボタンを透明にしたのが図 4-15 である。

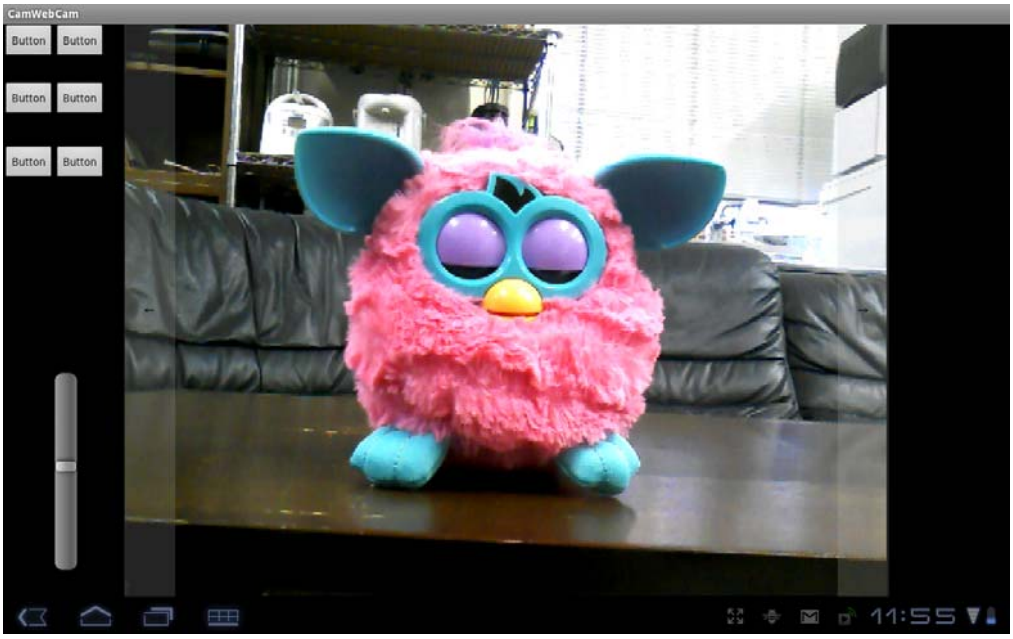


図 4-15 左右ボタンを透明にした様子

4.3.2　　上下シークバー

左右のボタンは見える映像の内側にボタンを設置し制御するのに対し上下のボタンはその範囲外にあり、さらにボタンではなくシークバーを用いることに違和感があったので、シークバーを見える映像の範囲に設置し透明にすることを考えた。まず画面をタッチすることで上下を制御できるよう、左右のボタンに被らないように拡大した。その図 4-16 である。

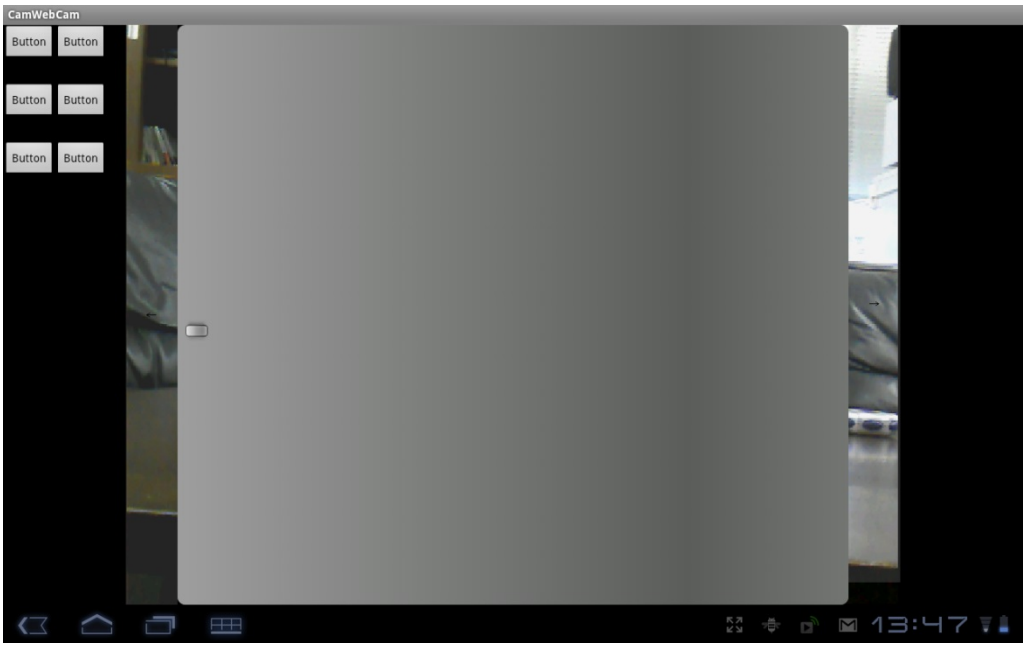


図 4-16 シークバーの拡大

その後 4.3.1 で説明した左右ボタンようの xml ファイルと同じように透明化の設定を seekbar.xml に記して drawable-mdpi に作成した。図 4-17 がその図である。

*seekbar.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >

    <item android:id="@android:id/background">
        <shape>
            <corners android:radius="10dip" />

            <gradient
                android:angle="270"
                android:centerColor="#005a5d5a"
                android:centerX="0.75"
                android:endColor="#00747674"
                android:startColor="#009d9e9d" />
        </shape>
    </item>
    <item android:id="@android:id/progress">
        <clip
            android:clipOrientation="vertical"
            android:gravity="bottom" >
            <shape>
                <corners android:radius="10dip" />

                <gradient
                    android:angle="270"
                    android:centerColor="#005a5d5a"
                    android:centerX="0.75"
                    android:endColor="#00747674"
                    android:startColor="#009d9e9d" />
            </shape>
        </clip>
    </item>

</layer-list>
```

図 4-17 シークバーを透明にするためのコード

4.3.1 と同様に seekbar.xml ファイルを作成したシークバーに参照させ透明にすることが出来た。シークバーを透明にした様子が図 4-18 である。

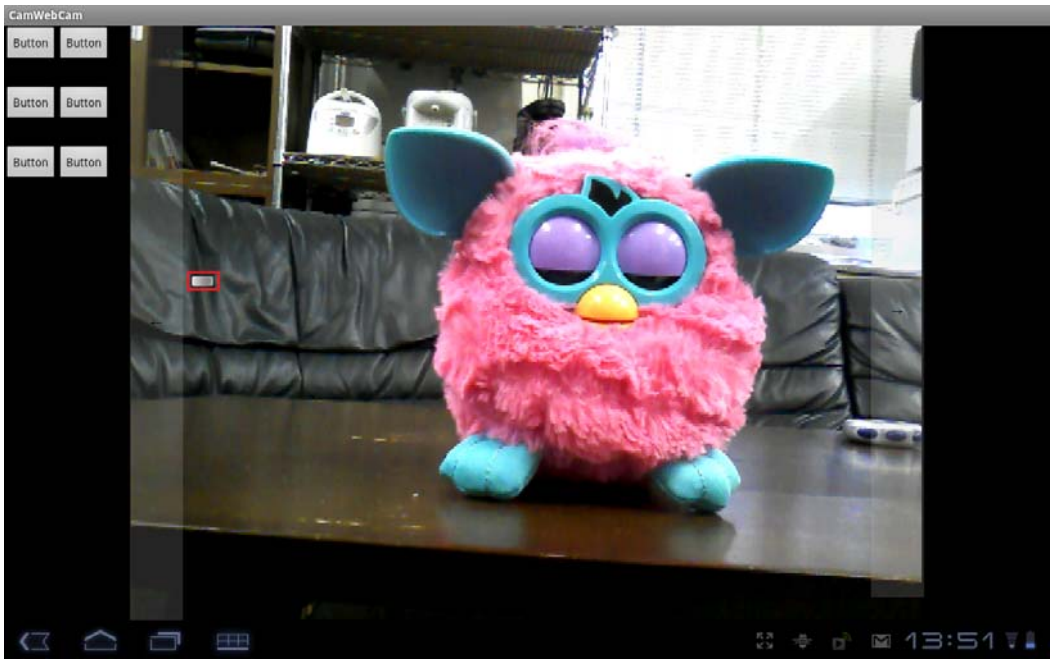


図 4-18 シークバーを透明にした様子

しかし図 4-18 の赤枠で囲ってある所のようにタブを透明にすることができなかった。原因はシークバーはプログラムで作成するのに対し、タブは画像を使っていることだった。そこでタブをプログラムで作成し透明にした。作成したタブが図 4-19 で、透明にした様子が図 4-20 である。

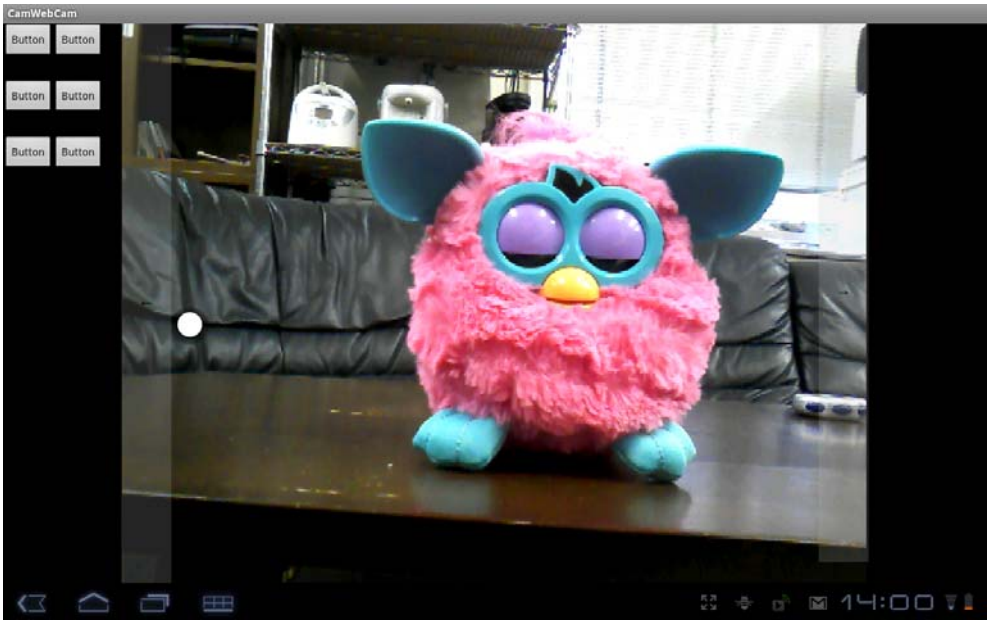


図 4-19 タブの作成

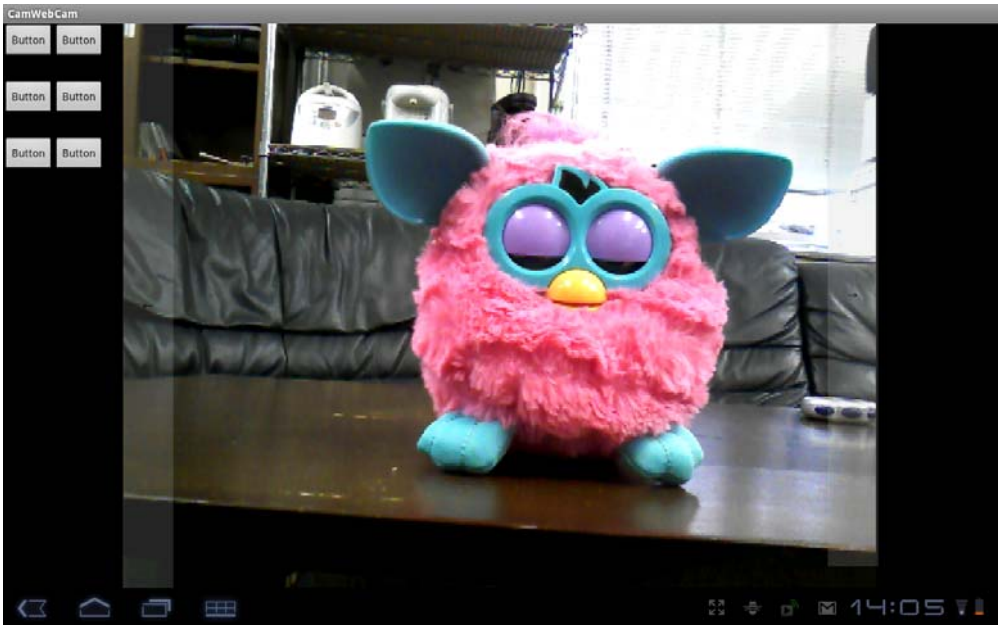


図 4-20 タブの透明

4.3.3　　配置

最後に、ズーム・フォーカス・明暗を制御するためのボタンのテキストをそれぞれ分かりやすく「近」「遠」、「ピント」、「明」「暗」に変え、大きくした。その実行画面が図 4-21 である。

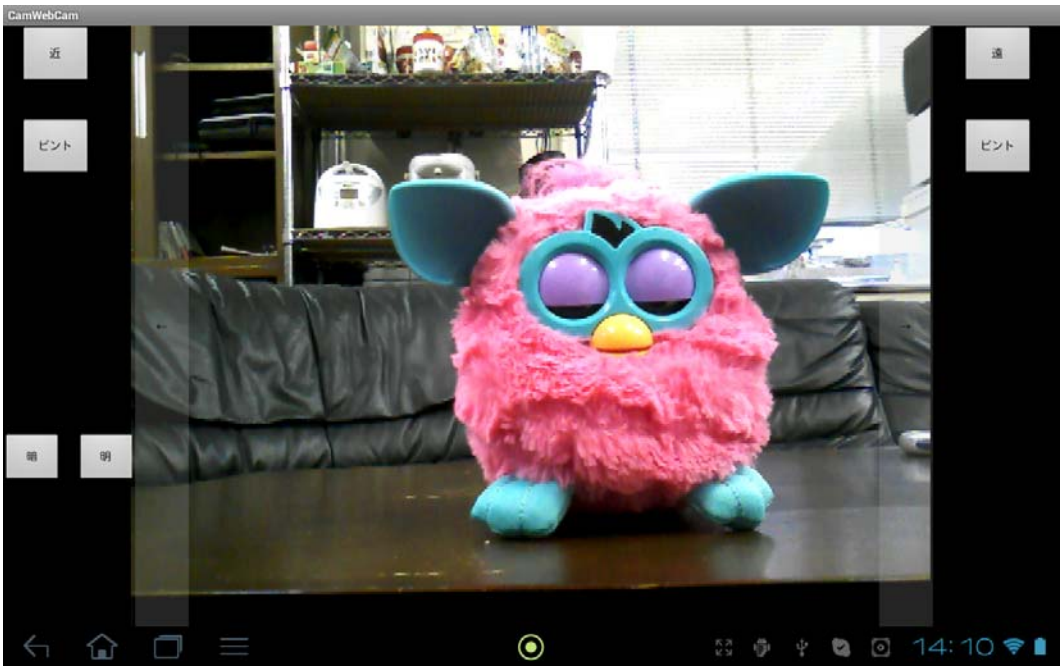


図 4-21 完成アプリケーション

第 5 章　　動作確認　（吉野担当）

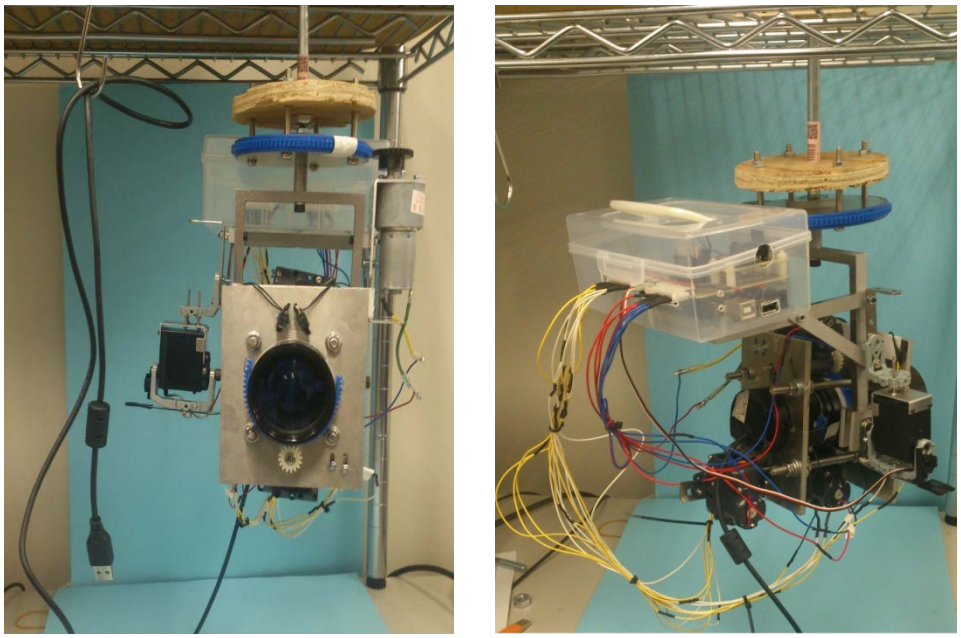
5.1　　動作確認の目的　　（吉野担当）

5.1.1　　概要

本章では実際に Android 観光望遠鏡を設置して動作確認を行うことで、本研究の要求機能が満たされていることをチェックする。

5.1.2　　動作確認の準備

Android 観光望遠鏡の回路の配線は仮のものであり、これを屋外や天井に設置することは、壁に取り付け用のボルト穴を開けることや配線の防水性が低い等の問題があるため、屋外の壁や天井ではなく研究室の棚にとりつけチェックを行うことにした。実際に設置した Android 観光望遠鏡の表側及び裏側からの写真が図 5-1 である。



前　　後

図 5-1 Android 観光望遠鏡の設置図

5.2　　動作確認の評価方法　　（吉野担当）

5.2.1　　評価目標

Android 観光望遠鏡の評価は機構・Arduino・Android、の３つの班の要求機能をもとに行うことにする。

要求機能を表 5-1 にまとめる。

表 5-1 要求機能一覧

製作した班	要求機能
機構	宙に下げてモータを稼働させても機体が壊れない
	フレキラックとピニオンが空転せずにカメラを操作できる
Arduino	モータの正転、反転、停止が動作する
	安全装置が動作する
	カメラが移動しても配線が絡まらない
Android	タブレットでカメラを操作できる
	タブレットで映像を確認できる

5.2.2　　動作確認のロケーション

動作確認は研究室内で行うことにし、被写体として目立ちやすいタカラトミー社製ファービーを選択した(図 5-2)。被写体をカメラからおよそ５m 離れたところにおいた状態で動作確認を行う。

図 5-2 被写体（タカラトミー社製ファービー）

5.2.3　　動作確認の手順

動作確認は以下の手順で行う

1. Android 観光望遠鏡をスチールラックに取り付け、Android タブレットと 2 本の USB ケーブルで接続する。
2. Android タブレットのアプリケーションを起動して映像が映ることを確認する。
3. 上下左右の操作をして、被写体を中央に捉える。
4. ズーム、フォーカス、明暗を操作して被写体をより鮮明に捉える。また安全装置が作動してモータが停止することを確認する。
5. Android 観光望遠鏡を大きく回転させ、Android 観光望遠鏡の機構及び配線に異常がないか確認する。

5.3　　動作確認の結果　　（吉野担当）

動作確認の結果をAndroid観光望遠鏡とタブレットの図と共に5.2.3の手順に沿って示す。

1.　　Android 観光望遠鏡を研究室の棚に取り付け、タブレットと接続した様子が図 5-3 である。この状態で水平方向のフレキラックとピニオンが空転しないのを確認した。

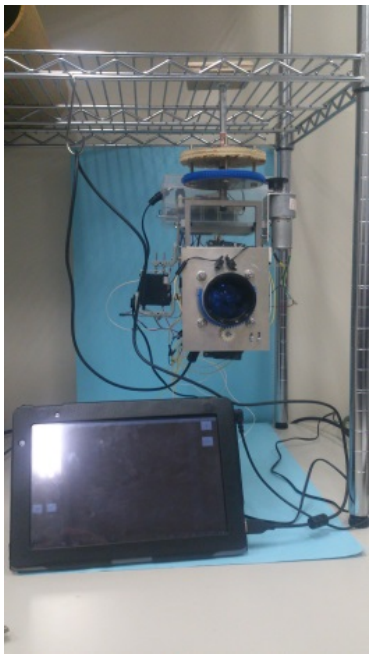


図 5-3 Android 観光望遠鏡とタブレットの全体図

2.　　アプリケーションを起動して Android 観光望遠鏡の映像がタブレット上で映し、Android 観光望遠鏡とタブレットの接続が安定していることを確認した(図 5-4)。

図 5-4 カメラから転送された映像をタブレット上で見ている状態

3. Android 観光望遠鏡を上下左右方向に操作して被写体を画面中央にもってくることができた。図 5-5 は Android 観光望遠鏡が被写体を捉えている時の図である。

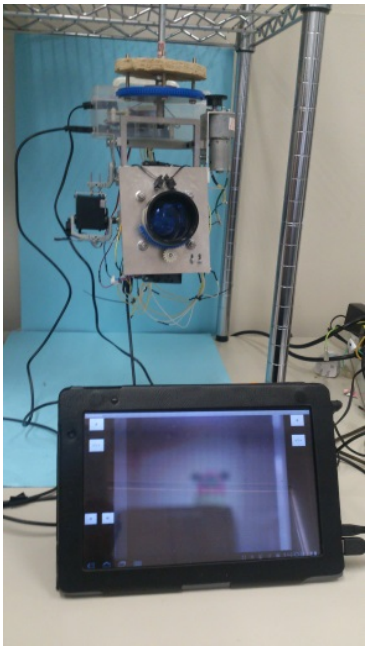


図 5-5 被写体に視点を合わせた状態

4. 手順 3 で画面中央にもってきた被写体をズーム等の機能を使って調節した。図 5-6 は調節をした後の図である。またマイクロスイッチを用いた安全機構の動作確認も同時に行った。

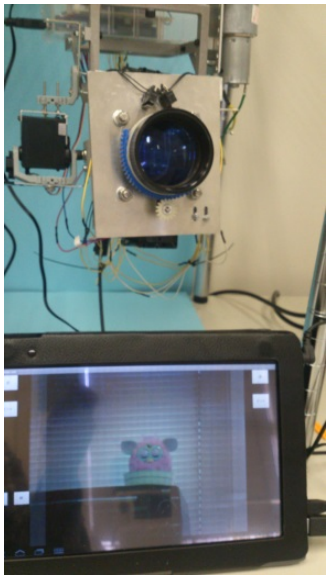


図 5-6 被写体にフォーカスを合わせた状態

5.　手順 4 の状態から Android 観光望遠鏡を左右に 180 度 回転させて機構と配線に異常がないことを確認した。図 5-7 はその時の図である。

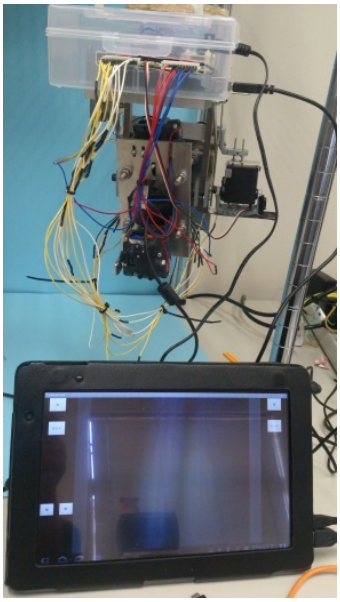


図 5-7 機体を左右方向に 180 度回転した状態

5.4　　考察　　（吉野担当）

動作確認の結果から表 5-1 に示した機構、Arduino、Android に関する要求機能が全て満たされていることが確認できた。

当初懸念していたフレキラックとピニオンの空転は起こらず、安全機構であるマイクロスイッチの動作も確実だった。

なお、本章の実験は全て電池で行ったが、これを 2.3.2 や 3.6.3 の章で触れたようにコンセントからの電源に変更したところ Arduino は問題なく動作した。

しかし、カメラの向きを変更するときの速さが左右方向と比べ、上下方向がやや早く、なれるまですこし操作しにくいと感じることがあった。

第 6 章　　付加価値　　(河端担当)

6.1　　概要　　(河端担当)

6.1.1　　付加価値の提案

本章では Android 観光望遠鏡に追加する付加価値を提案することを目的とする。Android 観光望遠鏡はカメラ映像および命令が有線接続になっている。これらの接続を無線化することを提案する。

6.1.2　　無線化の利点

図 6-1 は 2 章から 4 章で作成した Android 観光望遠鏡が、2 本の USB ケーブルと Android タブレットが接続されているところを示している。カメラからの映像は USB ケーブルで Android タブレットに送られ、Android タブレットからモータへの指令はマイクロ USB ケーブルから回路へ送信される。

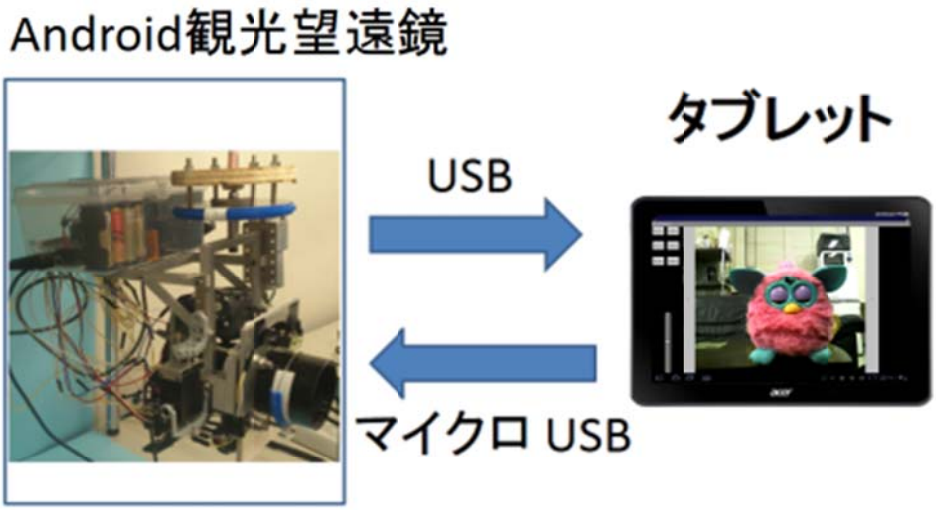


図 6-1 Android 観光望遠鏡システム

一方、本章で提案する無線化のイメージを示したのが図 6-2 である。Android 観光望遠鏡からの映像と、Android タブレットからのモータへの命令が無線化されていることを示している。

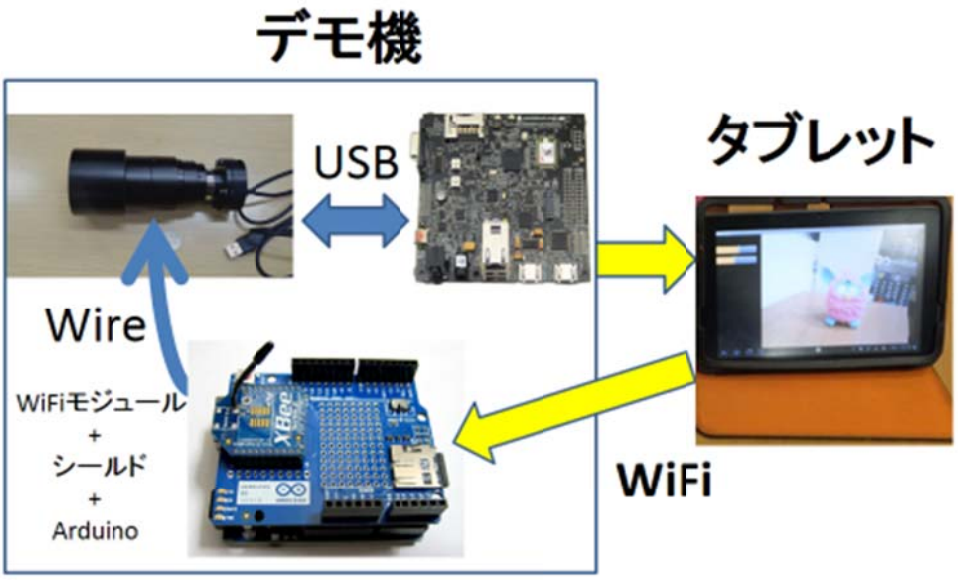


図 6-2 映像と命令を無線化したイメージ(黄色い矢印が無線通信を示している)

無線化にする利点は図 6-3(左)の 2 本の USB ケーブルが図 6-3(右)でなくなることであり、それにより利用者が持ちやすいシステムとなる。複数人で回して見られることに加え、USB ポートが付いていないスマートフォンでも使用ができるため、より利用の幅が広がるだろう。

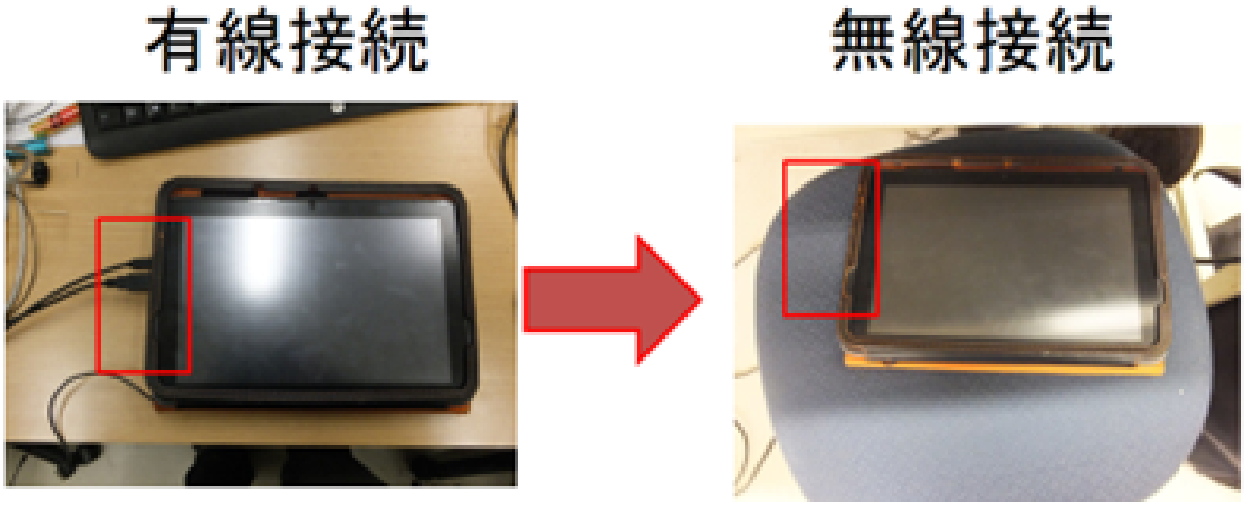


図 6-3 有線接続と無線接続

6.2　　無線化の仕組み　　(河端担当)

6.2.1　　通信手段

候補になった無線通信は 2 つあり、Bluetooth 通信と WiFi 通信である。Bluetooth は 1 対 1 を目的とした接続に優れ、WiFi は複数に接続できることに優れている。今回は複数人での接続を得意とする WiFi 接続を使った。

6.2.2　　ルーターの役割

WiFi 接続に必要な物に図 6-4 のルーターがある。ルーターの役割は IP アドレスを端末に振り分けることである。IP アドレスとは、ネットワーク上の機器を識別するために指定するネットワーク層における識別用の番号である。例として図 6-4 にルーターが複数台の PC に IP アドレスを設定している様子を示した。

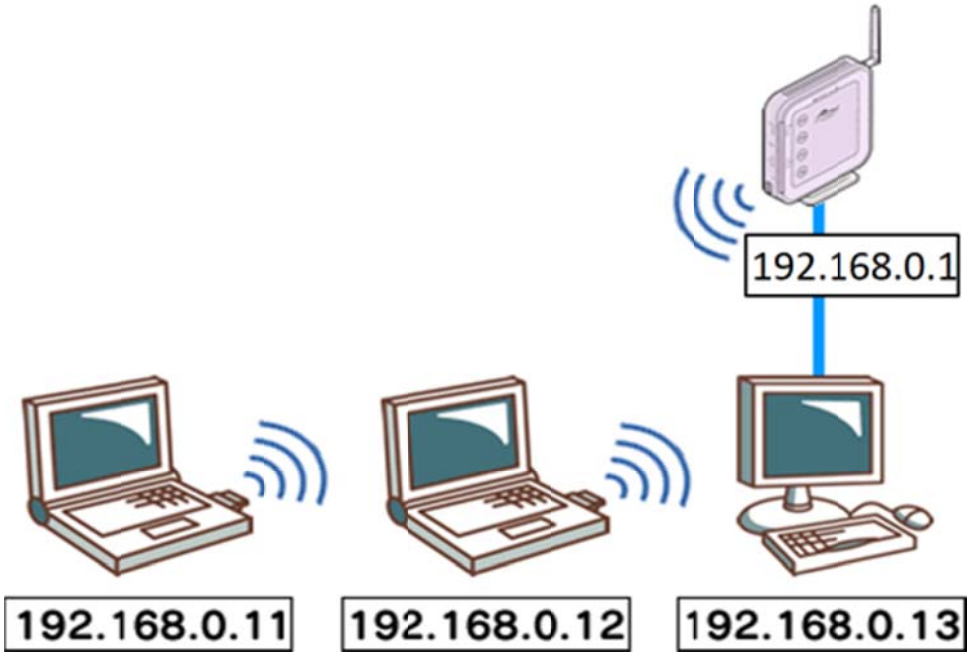


図 6-4 ルーターによる IP アドレスの振り分け[35]

6.2.3　　ルーターの種類

図 6-5(左)が室内用据え置きルーター, 図 6-5(右)は外出用小型ルーターである。小型ルーターは図 6-4 の機能である IP アドレス振り分けを外出先でも行うことができる。

図 6-5 2 種類のルーター[36][37]

本研究ではどちらでも使うことができるが、図 6-6 のように持ち運びができ、持ち運びのたびに IP アドレスを設定しなくてよいので、小型ルーターで本研究内のデモ機に使用した。

図 6-6 小型ルーターの機能イメージ[38]

6.3　　USB カメラの映像を無線化　　（河端担当）

6.3.1　　USB カメラの映像を Android タブレットに送る方法

USB カメラの映像を Android タブレットに送信する方法として、図 6-7 にあるように USB カメラと PandaBoard という小型 PC を USB 接続し、PandaBoard に入っているアプリケーションを使い USB カメラの映像を公開するという手法を用いる。公開された映像は Android タブレットからルーターを通して見ることができる。ここで重要なのは映像が PandaBoard のアプリケーションによって公開されているのでインターネット経由で見ることができるのは勿論、インターネットを介さなくても IP アドレスが同じルーターから与えられていれば同じ映像を見ることができる。

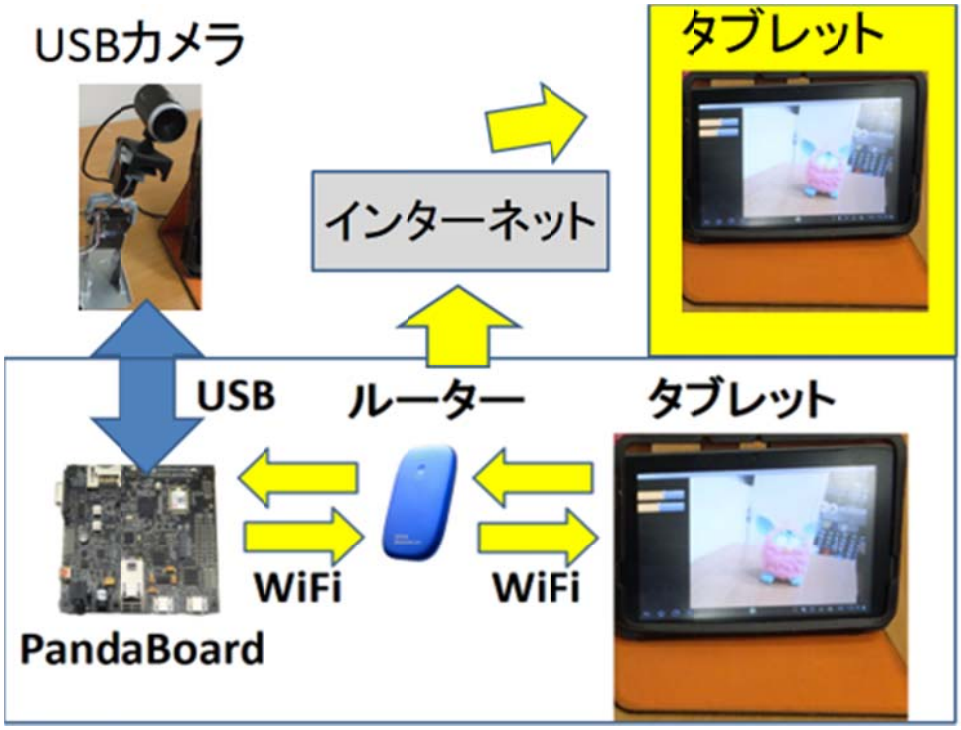


図 6-7 映像を Android タブレットに送信するイメージ

6.3.2　PandaBoard について

図 6-7 に使用している PandaBoard とは Texas Instruments 社が出している低消費電力、低コストのワンボードコンピュータである。ワンボードコンピュータとはむき出しの一枚（ワン）のプリント基板（ボード）の上に、電子部品と最低限の入出力装置を付けただけの極めて簡素なコンピュータである。図 6-8 にあるように SDHD カードに OS（Android、Linux など）を IP アドレスストールすることで、PC として用いることができる。図 6-8 のように、5V 電源、USB ポートから PC の周辺機器、及びにモニターを接続して用いる。

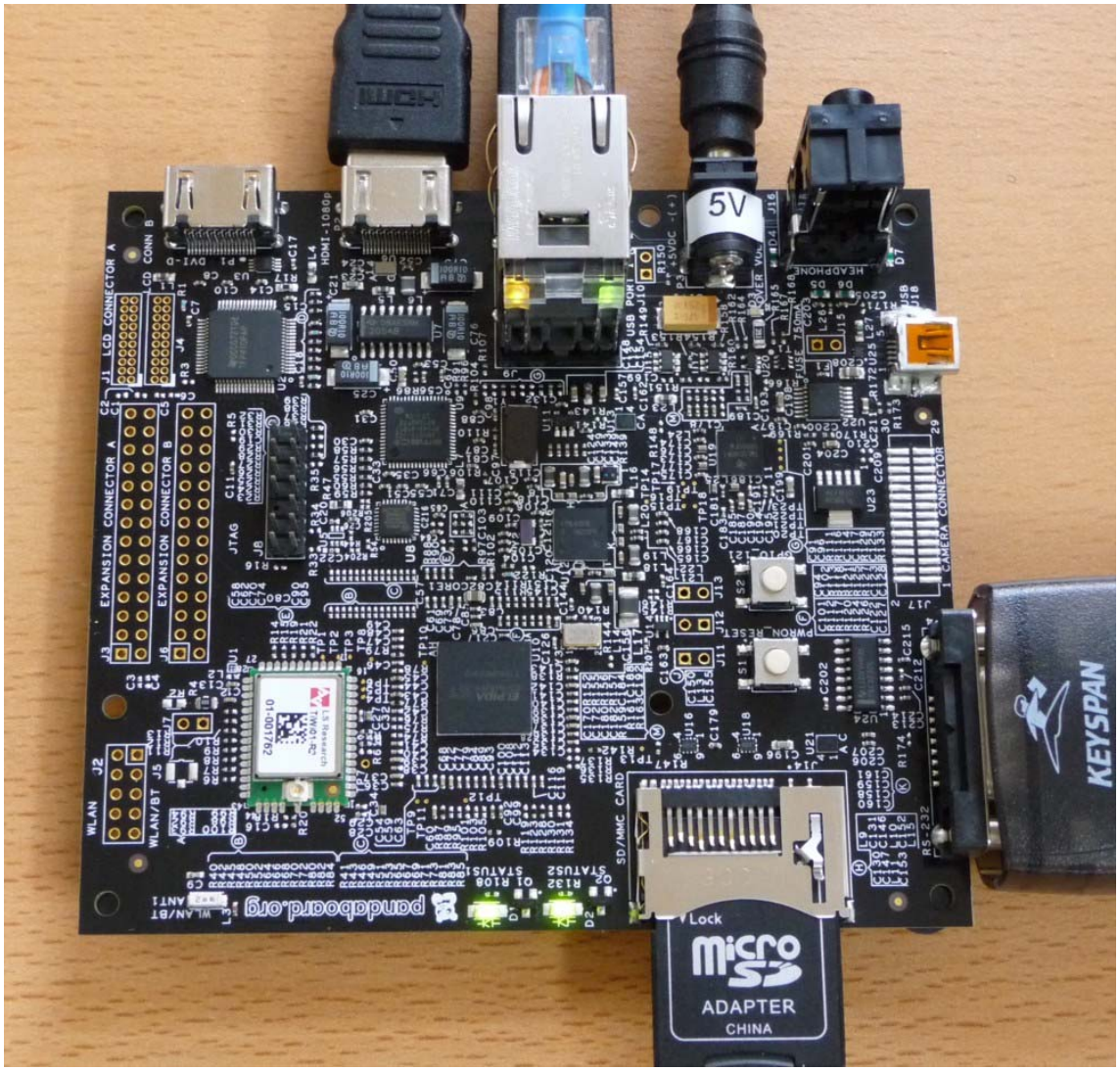


図 6-8 PandaBoard と周辺機器の接続[39]

6.3.3　PandaBoard へ書き込み

本章では SDHD カードに OS を書き込む方法を解説する。本来このような OS を書き込む場合、Linux OS の PC が好ましいが Windows で簡単に書き込む方法があったのでこちらの方法で書き込んだ。まず PandaBoard のサイトで OS をダウンロードする。次に SDHD カード書き込みソフト(opensolaris-liveusb-creator http://devzone.sites.pid0.org/OpenSolaris/opensolaris-liveusb-creator)を利用して直接書き込む。このソフトは無料で使用できる。次に SDHD カードを図 6-8 のように接続して電源をつけると PandaBoard が起動する。

次に無線映像を送る Linux PC のアプリケーションをインストールする。今回使用した PC アプリケーションは USB カメラの映像を配信するために作られた MjpgStreamer である。

Linux では起動画面でコマンド画面を起動し、黒い画面がでてきてコマンドが打てるようになる。次に以下のコマンドを打つことでソースがダウンロードできる。

```
# svn co https://mjpg-streamer.svn.sourceforge.net/svnroot/mjpg-streamer
```

これを PC にインストールするには以下のコマンドを打つ。

```
# cd mjpg-streamer/mjpg-streamer/
# make
```

これでアプリケーションが使用可能になったので USB カメラを PandaBoard に接続し、以下のコマンドを打ち込むことで MjpgStreamer を起動する。

```
# ./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -y" -o "./output_http.so -w ./www -p 8080 -c hoge:hoge"
```

(# ./mjpg_streamer -i ″入力側 -d 使用するデバイス″ -o ″出力側 -w Web 表示用のファイル置場 -p ポート -c ユーザ名:パスワード″)

確認として PandaBoard 上のブラウザで「http://PC の IP アドレス:8080」を指定すると図 6-9 の画面がでてきて USB カメラの映像を見ることができる。

図 6-9 PandaBoard 上での MjpgStreamer 起動画面[40]

最後に MjpgStreamer の映像を Android タブレットから見るのに、Bitbucket から SimpleMjpegView[41]をダウンロードする。Eclipse にインポートし、Android タブレットにアプリケーションを入れる。アプリケーションを起動して設定をすることで図 6-10 のように Android タブレットで無線化された映像を見ることができた。

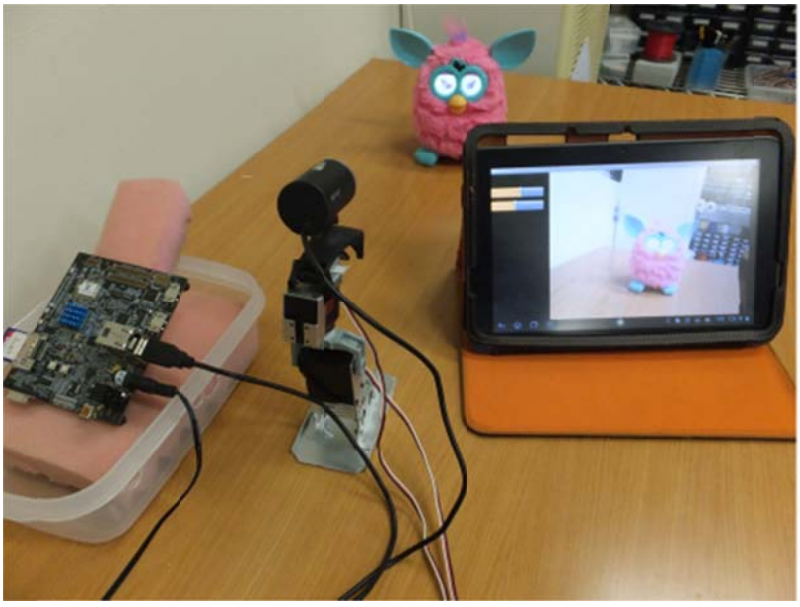


図 6-10 Android タブレットでの確認様子

6.4　Arduino への命令の無線化　（河端担当）

6.4.1　無線で命令するイメージ

付加価値班では Android タブレットからの命令を Arduino に送るため、図 6-11 にあるように Android アプリケーションと次章で解説する XbeeWiFi との間で WiFi 接続を行う。Android タブレットの命令がルーターを介して、XbeeWiFi に伝えられる。その命令はシールドを通じて Arduino に送られ、Arduino プログラムがその命令に合わせてモータを動かすという仕組みである。

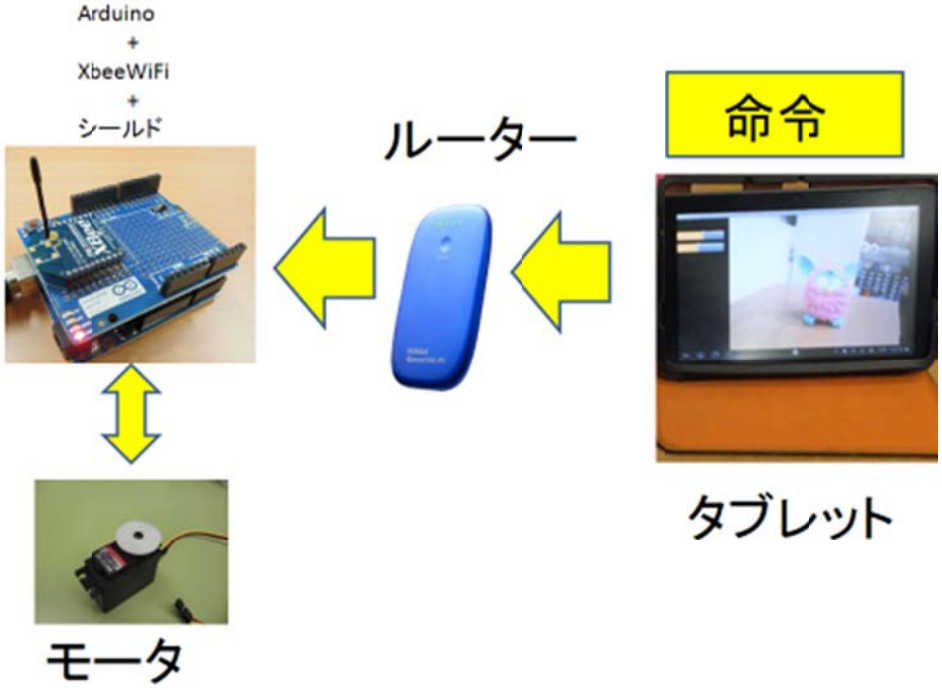


図 6-11 Android から Arduino への命令が無線化されるイメージ

6.4.2　　WiFi モジュールについて

図 6-11 の WiFi 接続に必要不可欠のものが図 6-12 にある WiFi モジュールである。このモジュールはディジ･インターナショナル社製 Zigbee 規格に沿った無線モジュールであり、正式名称は XbeeWiFi 型番 XB24-WFPIT-001 である。

図 6-12 XbeeWiFi XB24-WFPIT-001[42]

Zigbee 規格とはセンサーネットワークを主目的とする近距離無線通信規格の一つであり、このモジュールの仕様としてはデータ転送速度 20Kbps-250kbps の間である。ただし、使用する無線周波数帯によって異なり、2.4GHz では 250Kbps である。電波法の関係から日本国内で利用できるのは ISM として開放されている 2.4GHz 帯を用いた仕様のみである。

このモジュールは基板に組み込み可能であるが、本研究では回路を作らなくてよい Arduino 用シールドを用いている。

6.4.3　XbeeWiFi の設定

XbeeWiFi で WiFi 接続をするために必要な設定は、接続先の IP アドレスとルーターのパスワード、モード設定(今回はルーターを使用する設定)である。それ以外はデフォルトでよい。

それらの設定を行うのに Windows 版の X-CTU[43]を使い設定した。図 6-13 にある Scan をクリックすると接続可能なルーターが出てくるので、自分が使用するルーターを選ぶ。接続の前にパスワードを入力する。Read をクリックすると SSID(ルーターに割り当ている ID)と MY(XbeeWiFi に割り当てられている IP アドレス)に ID が入る。あとは DL をクリックしてから(接続したい Android に設定されている IP アドレス)入力し、Write をクリックすると書き込みが始まり、終われば設定が完了する。

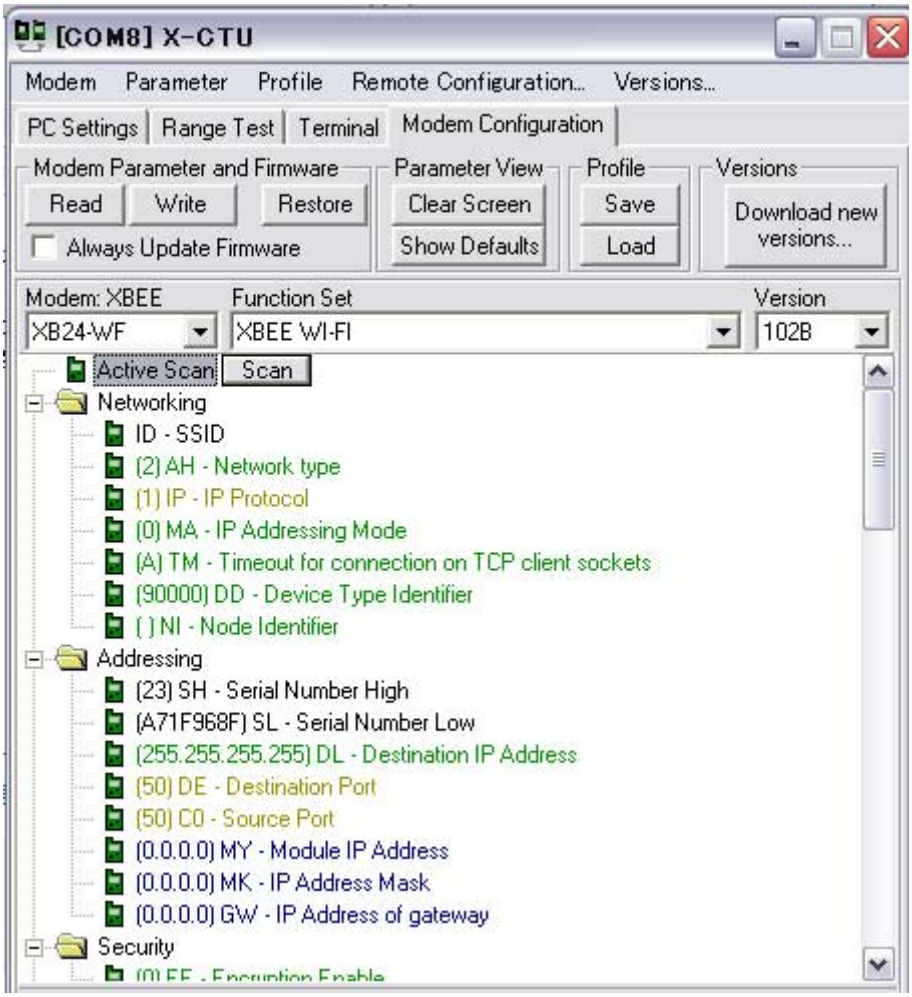


図 6-13 X-CTU 設定画面[44]

これで XbeeWiFi の設定が完了する。

6.4.4　　モータの無線化プログラム

Android タブレットと XbeeWiFi の無線通信には Socket 通信を用いる。Socket 通信とは図 6-14 のように端末同士の郵便のやり取りのようなものである。この例えを用いて説明すると、誰かに手紙送るとき、郵便番号と住所が事前にわかっていることで、手紙が相手先に届けられる。この郵便番号と住所を Android タブレットと XbeeWiFi に置き換えると IP アドレスとポート番号(コンピュータがデータ通信を行う際に通信先のプログラムを特定するための番号)がそれにあたる。

図 6-14 Socket イメージ

6.4.5　　起動テスト

ここでは、Android と XbeeWiFi との間で Socket 通信するテストを行う。実行するのは、Android 側のプログラムから Socket 通信で文字を送信するプログラムである。Arduino では送られた文字が H なら LED を点灯し、L なら LED を消灯する。

```
//Arduino　LED 点滅のサンプル
const int ledPin = 13;
int incomingByte;
void setup() {
  Serial.begin(9600);
   pinMode(ledPin, OUTPUT);
}
void loop() {
   if (Serial.available() > 0) {
       incomingByte = Serial.read();
    // H が来たら点灯
    if (incomingByte == 'H') {
       digitalWrite(ledPin, HIGH);
    }
    //L が送られたら、消灯
    if (incomingByte == 'L') {
       digitalWrite(ledPin, LOW);
    }
  }
}
```

```
//Android SocketEX(一部抜粋)

private final static String IP="192.168.1.102";//★変更必須
//接続
    private void connect(String ip,int port) {
        int size;
        String str;
        byte[] w=new byte[1024];
        try {
            //ソケット接続(2)
            addText("接続中");
            socket=new Socket(ip,port);
            in =socket.getInputStream();
            out=socket.getOutputStream();
            addText("接続完了");

            //受信ループ(3)
            while (socket!=null && socket.isConnected()) {
                //データの受信(4)
                size=in.read(w);
                if (size<=0) continue;
                str=new String(w,0,size,"UTF-8");

                //ラベルへの文字列追加
                addText(str);
            }
        } catch (Exception e) {
            addText("通信失敗しました");
        }
    }
  public void onClick(View v) {
        //スレッッドの生成
        Thread thread=new Thread(new Runnable(){public void run(){
            error=false;
            try {
                //データの送信(5)
```

```
            if (socket!=null && socket.isConnected()) {
                byte[] w=edtSend.getText().toString().getBytes("UTF8");
                out.write(w);
                out.flush();
            }
        } catch (Exception e) {
            error=true;
        }
        //ハンドラの生成
```

図 6-15 が実行様子である。H を送信したとき LED が緑色に点灯し、L を送信すると LED の消灯を確認できた。

図 6-15 サンプル実行画面

6.5　　デモ機　　（河端担当）

6.5.1　　検証方法について

デモ機を用いて映像と命令の無線化を検証した。デモ機はカメラを上下左右動かせるように 2 個のサーボモータを用い、その上に USB カメラを付けたものである。このデモ機のサーボモータを Arduino に接続し、USB カメラを PandaBoard に USB 接続した。小型ルーターを用いて PandaBoard、Android タブレットとシールド付 XbeeWiFi を無線接続した。

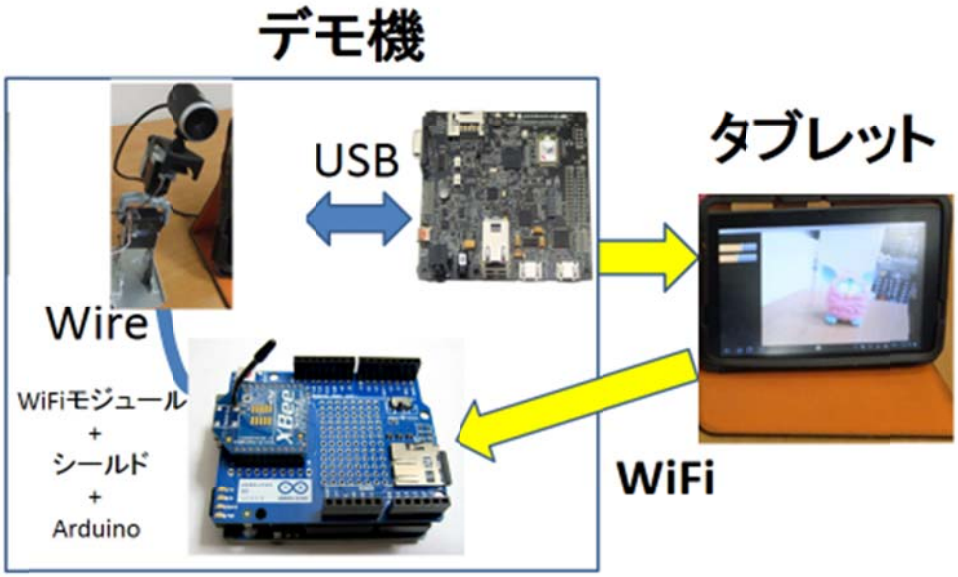


図 6-16 デモ機の全体図(黄色い矢印は無線通信している)

6.5.2　サンプルプログラムの変更点

サンプルのプログラムでは 1byte の命令を送っていたが、デモ機を動かすのに 2byte のデータを送信するものに変える必要がある。理由は 1byte 目の命令で対象のモータを決め、2byte 目の命令で角度を決めるためである。デモ機のプログラムには Arduino 側で 0～255 の値を 0～180°に変換してサーボモータを制御している。Arduino では 1 つ目の命令が来るのを待っているプログラムを 2 つの命令がくるまで待たせるように変えた。そこを変えないと 1 つ目の命令でコードが読み込まれてしまい、2 つ目の命令が読み込まれず、2 つ目の数値がでたらめな数値を検出したためである。以下がそのコードである。

//Arduino デモ機

```
#include <Servo.h>
Servo myservo1;
Servo myservo2;
// the pin that the LED is attached to
int incomingByte0;          // a variable to read incoming serial data into
int incomingByte1;
void setup() {
  // initialize serial communication:
  Serial.begin(9600);
myservo1.attach(8);
 myservo2.attach(9);
}
void loop() {
 if (Serial.available() > 1) {
    incomingByte0 = Serial.read();
     incomingByte1 = Serial.read();
    if (incomingByte0 == 'b') {
    val = map(incomingByte1, 0, 1023, 0, 180);
  myservo1.write(val);
    }
    if (incomingByte0 == 'c') {
    val = map(incomingByte1, 0, 1023, 0, 180);
  myservo2.write(val);
    }
  }
}
```

```
                    //JAVAデモ機  SocketEX.java
        sk1.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
                @Override
                public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromTouch) {
                        byte command = (byte) b;
                        byte value    = (byte)progress;
                    sendcommand(command,value);
            }
```

6.5.3　　実験

PandaBoard に電源、USB カメラを接続し、Arduino には XbeeWiFi とシールドを接続する。図 6-17 のように Android タブレットから映像と命令を送ることができるかを検証するのが実験の目的である。

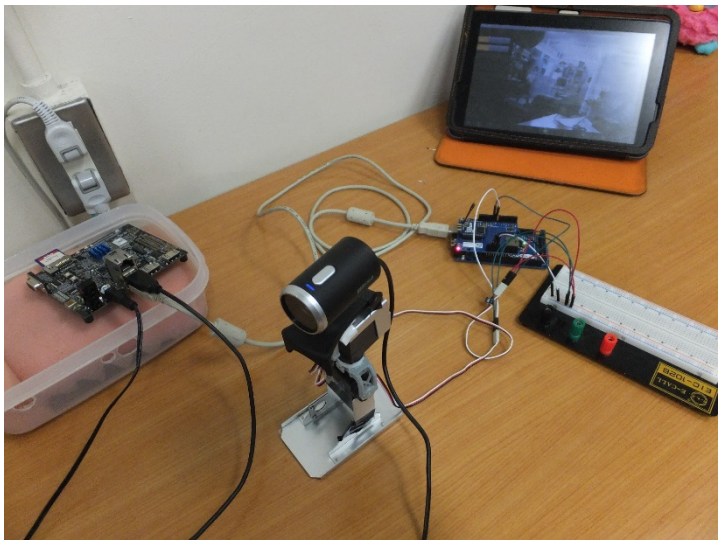


図 6-17 デモ機全体図

6.5.4　　実験結果及び結論

まず、図 6-18 のように Android タブレットに USB カメラの映像が写っていることが確認できた。Android タブレットの左赤枠の半分黄色のバーがある。バーの色の部分が多いほど 2 つ目の命令で送られる数値が高くなり、サーボモータが制御できているところが確認できた。これで USB カメラの映像と操作ができることが確認できた。しかし、動作後の USB カメラの映像が Android タブレットに映るまで 1 秒前後のラグがあった。

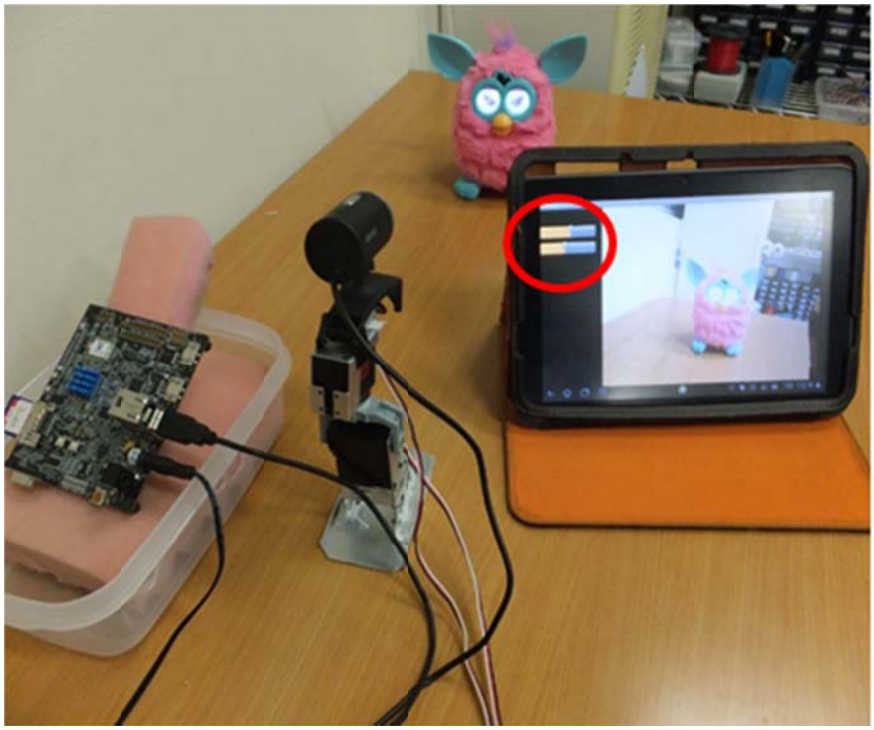


図 6-18 デモ機実行画面

ラグについての考察はルーターの通信速度が遅いからではないかと考察している。小型ルーターの通信速度は 8.71 Mbps で室内用据え置きルーターでは 94.68Mbps である。実際に室内用据え置きルーターで実験してみたところ 0.5 秒前後にラグが小さくなった。

結論として、Android 観光望遠鏡の映像と命令を無線化できることが技術的に確認できた。

6.6　　デモ機を踏まえた改良機

6.6.1　　デモ機の問題点

映像と命令を無線で Android 観光望遠鏡に伝えるデモ機を作成したが、下記のようにいくつかの問題があった。

1. 映像に 0.5 秒から 1 秒程度のラグがあること
2. PandaBoard と Arduino 上の XbeeWiFi とにそれぞれ別の IP アドレスが振られるため、設定やプログラミングが煩雑となること

1.のラグの問題については、既に述べたルーターの性能の問題の他、ソフトウェアに起因する問題も考えられ、ソフトウェアの改良によりさらに短くすることは可能であると考えられる。一方、2.の問題は現状のハードウェア構成では解決することができない。さらに、この構成を用いると

- XbeeWiFi のシールドと干渉するため、3 章で作成した制御回路を用いることができない

という問題も起こる。

そこで、ここでは XbeeWiFi の使用をやめ、新しい構成での Anroid 観光望遠鏡の無線化法を紹介する。

6.6.2　　改良機の構成

改良機の構成は図 6-19 の通りである。

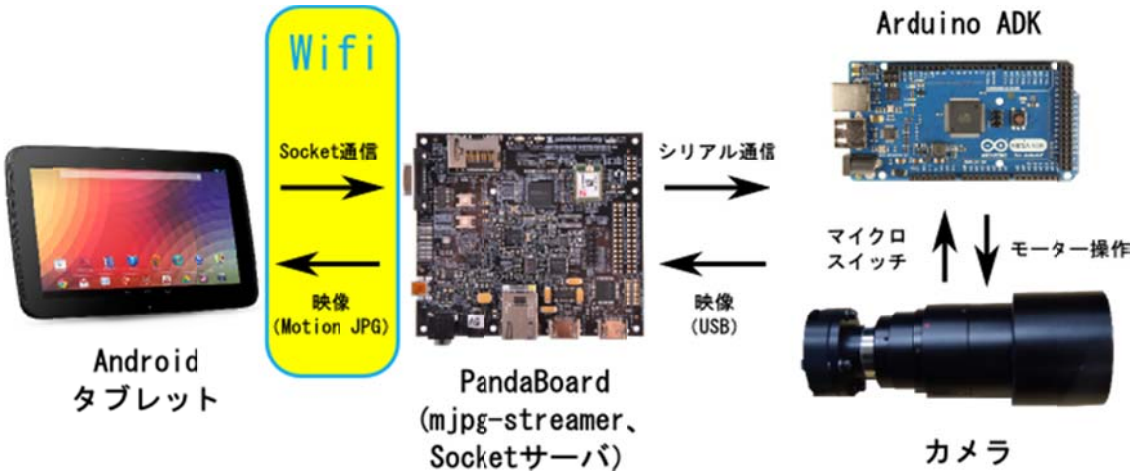


図 6-19 改良機の構成

この改良機の特徴をまとめると下記のようになる。

- Android タブレットが通信する相手は PandaBoard のみである
- 図 4-1 のシステム構成における Android タブレットと望遠鏡システムの間に PandaBoard を挿入した形になっており、既存の回路をそのまま使うことができる
- Android タブレットからの命令は Socket 通信によって PandaBoard に伝えられる。PandaBoard はそれを USB 接続された Arduino ADK に対してシリアル通信で伝える。なお、既存回路との互換性のために Arduino ADK を用いているが、ADK 機能は使わず、Arduino Mega として用いている。Socket 通信とシリアル通信を行うプログラムは C 言語で記述されている
- 映像配信部はデモ機と同じく mjpg-streamer と SimpleMjpegView[41]を用いる

この改良機を実装した様子が図 6-20 である。3 章で作成した回路がそのまま使われていることがわかる。

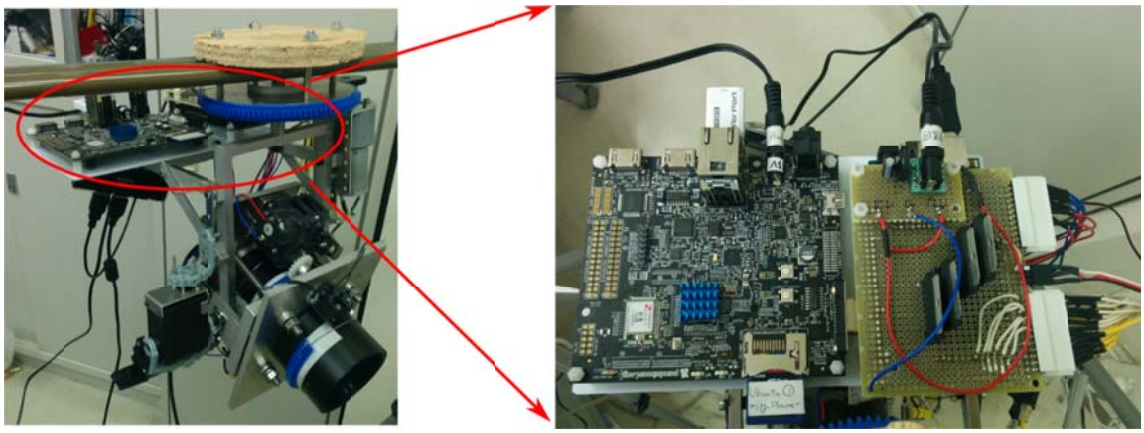


図 6-20 改良機の実装

6.6.3 改良機の動作

改良機をテストしたところ、良好に動作した。映像のラグは有線と同等というわけにはいかないが、0.5 秒よりは短く、不自然さを感じない程度には低減された。これはソフトウェアを全体的に見なおしたことが原因であろう。

この改良機を 2014 年 3 月 21 日に秋葉原で行われる Android Bazaar and Conference 2014 Spring にて「Android 観光望遠鏡」として展示する。

第 7 章　　結論　　（吉野担当）

7.1　　目標と Android 観光望遠鏡の比較　　（吉野担当）

7.1.1　　目標

Android 観光望遠鏡の要求機能は、Android タブレットで望遠カメラのズーム・フォーカス・明暗を調節でき、さらに望遠カメラの向きを上下左右に操作できることである。

7.1.2　　Android 観光望遠鏡の評価

機構班は望遠カメラのズーム・フォーカス・明暗を調節でき、上下左右方向に操作することのできる機構を製作した。さらに安全機構としてマイクロスイッチを取り付け、回路や電源を入れるための電池ボックスを作製した。

回路班はズーム・フォーカス・明暗の調節及び上下左右方向を操作のため DC モータ 4 つ及びサーボモータ 1 つの 2 種類のモータを制御するプログラムを作製した。安全機構として Android 観光望遠鏡にマイクロスイッチを取り付け、フレキラックがマイクロスイッチに触れた瞬間 DC モータを止めるような、機構を壊さないためのシステムを作った。また、操作性の向上として DC モータを停止する際のラグを小さくするためにモータ停止の際に「静止」だけではなく「静止」→「逆転」→「静止」と命令を送ることでより正確に停止できるようにした。動作確認を行うために基板を作製し Android 観光望遠鏡の配線を行い、さらに Arduino 及びモータの電源を電池からコンセントに変更することで総重量の軽減と安定した電源の供給を可能とした。

Android 班は Android タブレットに望遠カメラの映像を表示しながら、機構を動かすためのモータを操作できるアプリケーションを開発した。望遠カメラの映像を Android タブレットの画面中央に配置し、フォーカス等を操作するボタンを映像の外側に、左右操作のためのボタンを内側左右に、上下操作用のシークバーを映像部分全体に配置した。

7.1.3　　付加価値班の結果

付加価値班は Android 観光望遠鏡の付加価値案として無線化を提案した。通信種類として 1 つの望遠カメラと複数の Android タブレットという接続を想定して WiFi 通信を選択している。望遠カメラからの映像を PandaBoard 上のアプリケーションを使用して、Android タブレットに送っている。また Android タブレットの命令をモータに送るには XbeeWiFi を用いた。小型の USB カメラを用いてデモ機を製作し、命令と映像の無線化のテストを行った。結果としてモータの動作は安定したが、映像のほうは小型ルーターでは 1.0 秒、室内用据え置きルーターでは 0.5 秒ほどのラグがでた。これはテストに使用したルーターの通信速度が原因だと考えられる。さらに、このデモ機を踏まえて改良機を作成し動作を確認したところ、ラグの問題は解決した。

7.2　　総合評価　　（吉野担当）

研究テーマである「レンズと Android を組み合わせ、次世代に向けて何ができるか」について、レンズの需要を増やすために普及著しい Android タブレットと需要の減ってきた観光望遠鏡を組み合わせ、従来の観光望遠鏡を置き換える Android 観光望遠鏡を提案し、製作した。この目的のために私たちは機構班、Arduino 班、Android 班、付加価値班の４つに分かれて活動した。製作した Android 観光望遠鏡の動作確認を行い、要求機能を十分に満たしていると判断した。これにより、ユーザーにとって使いやすいレイアウトの Android 観光望遠鏡を製作し、レンズの新しい需要を提示することができた。

第 8 章　　参考文献・URL

[1]スマートフォン普及動向調査

http://www.d2c.co.jp/news/2012/20120418-1340.html

[2]OUR MOBILE PLANET

http://www.thinkwithgoogle.com/mobileplanet/ja/downloads

[3]マイナビニュース

http://news.mynavi.jp/news/2013/10/10/189

[4]MM 総研

http://www.m2ri.jp

[5]Strategy Analytics による調査結果

http://blogs.strategyanalytics.com/WSS/post/2014/01/29/Android-Captured-79-Share-of-Global-Smartphone-Shipments-in-2013.aspx

[6]球面レンズと非球面レンズ

wikipedia

http://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:BiconvexLens.jpg

日本ニューフーズル

http://www.nihon-new-nozzle.co.jp/mt/mt-search.cgi?tag=%E9%9D%9E%E7%90%83%E9%9D%A2%E3%83%AC%E3%83%B3%E3%82%BA&IncludeBlogs=6&IncludeBlogs=6

[7]TAMIYA　4 速ウォームギヤボックス HE

http://www.tamiya.com/japan/products/72008_4speedwormgear

[8]カメダデンキ　補助金具 LK、HK

http://www.kameden.com

[9]KHK STOCK GEARS　DR 成形フレキラックシリーズ一覧

http://www.khkgears.co.jp/khkweb/search/sunpou.do?indexCode=30&lang=ja&referrer=series

[10]RAINBOW PRODUCTS　平ギヤ(MOD1.0)　16 歯

http://www.powers-rainbow.com/cgi-bin/tkxcgi/shop/goods_detail.cgi?CategoryID=000011&GoodsID=00000143

[11]RAINBOW PRODUCTS　平ギヤ(MOD1.0)　20 歯

http://www.powers-rainbow.com/cgi-bin/tkxcgi/shop/goods_detail.cgi?CategoryID=000011&GoodsID=00000144

[12]RAINBOW PRODUCTS　平ギヤ(MOD1.0)　25 歯
http://www.powers-rainbow.com/cgi-bin/tkxcgi/shop/goods_detail.cgi?CategoryID=000011&GoodsID=00000145
[13]楽天 ICIBA　ステンレス寸切りボルト
http://item.rakuten.co.jp/nejiya/longscrew_285x006
[14]楽天 ICIBA　KSR-788HV ICS Red Version
http://item.rakuten.co.jp/grass-road/866743
[15]TAMIYA SHOP ONLINE　AO-8025 タミヤギヤードモータ 380-300
http://tamiyashop.jp/shop/product_info.php?products_id=89865
[16]KHK STOCK GEARS　SSDR　DR 専用ピニオンシリーズ一覧
http://www.khkgears.co.jp/khkweb/search/sunpou.do?indexCode=111&referrer=series&sic=1&lang=ja
[17]TAMIYA SHOP ONLINE　AO-8055 ギヤードモータ用マウント
http://tamiyashop.jp/shop/product_info.php?products_id=89926
[18]Robot Shop Technologia WebShop　SC-78X-A（ブラケットタイプ A）
http://www.technologia.co.jp/59_3636.html
[19]せんごくネット通販　神明電機 MQS-54-5L マイクロ SW
http://www.sengoku.co.jp/mod/sgk_cart/detail.php?code=2A2R-85M8
[20]DC モータの基礎
http://www.picfun.com/motor01.html
[21]秋月電子通商　モータードライバー　TA7291P(2 個入)
http://akizukidenshi.com/catalog/g/gI-02001
[22]YAHOO! JAPAN ブログ　TA7291P を読むーワイルドミニ四駆 RC
http://blogs.yahoo.co.jp/gozubonzoku/4088504.html
[23]H8/3664　制御で自走模型
http://www.ei.fukui-nct.ac.jp/~t-saitoh/exp/h8/basecar.html
[24]ヘイシンロボディスペン　サーサーボモータについて
http://www.robo-dispenser.com/compass/compass07.html
[25]ワンダーキットプロダクツ　サーボモータって何
http://wonderkit.kyohritsu.com/roid/servo.html
[26,27]フォトセンサで物体の存在を検出する
http://www.geocities.jp/zattouka/GarageHouse/micon/Arduino/PS/PS.htm
[28]nanoblog　RPR-220 反射型フォトセンサを使ってみる
http://nanoappli.com/blog/archives/5051
[29]N.Kojima　ユニバーサル基板の作り方
http://homepage1.nifty.com/x6/elecmake/universal.htm

[30] hishiyasan のホームページへようこそ 大阪出張所
http://hishiyasan.com/ElectroD.html
[31]デジタルアラームクロック
http://www.zea.jp/audio/clock/clock_02.htm
[32]GAPSIS　Xperia Tablet S
http://www.gapsis.jp/2012/10/xperia-tablet-s-trouble.html
[33]Bitbucket SimpleWebCam
https://bitbucket.org/neuralassembly/simplewebcam
[34]伊藤　元(2012)『Android×Arduino でつくるクラウド連携デバイス』　株式会社インプレスジャパン
[35]無線 LAN ルーター 画面で見るマニュアル
http://www.iodata.jp/lib/manual/wn-gdn_us2_20091029/htm/tcp-1.htm
[36]ゲーム好きの PS 系ブログ 雑記
http://gamesukiblog.blog40.fc2.com/blog-entry-122.html
[37]BIGLOBE WiMAX データ端末詳細
http://office.biglobe.ne.jp/mobile/wimax/device02.html
[38]VAIO Duo 11 徹底レビュー
http://satouchi.com/vaio/duo/svd1121aj-customize.html
[39]PandaBoard
http://en.opensuse.org/HCL:PandaBoard
[40]ウェブカメラでストリーミング
http://cubic9.com/Devel/%C5%C5%BB%D2%B9%A9%BA%EE/RaspberryPi/%A5%A6%A5%A7%A5%D6%A5%AB%A5%E1%A5%E9%A4%C7%A5%B9%A5%C8%A5%EA%A1%BC%A5%DF%A5%F3%A5%B0/
[41] SimpleMjpegView
https://bitbucket.org/neuralassembly/simplemjpegview
[42] XBee grows up
http://www.engadget.com/2011/07/29/xbee-grows-up-delivers-wifi-to-diyers-and-arduino-enthusiasts
[43] XCTU
http://www.digi.com/support/productdetail?pid=3352
[44]XBee WIFI　酔っぱらいおじさんＷＥＢ事始め
http://trhk.exblog.jp/15794379

謝辞

最後に、この場をお借りして本研究を進めるにあたり、望遠カメラの提供等のご支援いただいた K 社、並びに 2 年間様々なご指導を頂きました金丸隆志准教授、新井敏夫教授、機構の設計や金属の加工の相談に快く関わって頂いた花野井利之様、研究全体に対して多くの助言を頂いた修士の先輩方、協力していただいた皆様への心からの感謝の気持ちと御礼を申し上げたく、謝辞にかえさせていただきます。

２０１３年度（平成２５年度）

ECPⅢ

Final Report

Androidアプリケーション&回路班付録

チーフアドバイザー：金丸　隆志　准教授

サブアドバイザー：新井　敏夫　教授

チームメンバー

G1-10007　今井一智

G1-10014　小野将弘

G1-10020　紀伊皓介

G1-10050　濵内　瞬

G1-10075　吉野晃平

G1-09030　河端和也

目次

第 1 章　　最終的な Arduino のプログラム

本章では、実際に本研究で使用した最終的なプログラム(Arduino)を以下に載せる。

```
#include <Max3421e.h> //ADK を利用するための 3 つのライブラリを読み込む
#include <Usb.h>
#include <AndroidAccessory.h>
//int incomingByte0;          // a variable to read incoming serial data into
//int incomingByte1;
#include <Servo.h>
Servo servo1;//
int motorMoving1 = 0;
int motorMoving2 = 0;
int motorMoving3 = 0;
//Android アプリ側の accesory_filter.xml 内の属性と一致させる(2)
AndroidAccessory acc("Dorobook,Socym",     //第 1 引数:組織名（manufacturer 属性と一致）
"DC1",                                     //第 2 引数:モデル名（model 属性と一致）
"DC1 - Arduino USB Host",   //第 3 引数:ダイアログ表示メッセージ
"1.0",                                          //第 4 引数:バージョン（version 属性と一致）
"http://accessories.android.com/", //第 5 引数:ジャンプ先 URL
"0000000012345678");                        //第 6 引数:シリアル番号

void setup()   //最初に一度だけ実行される部分
{
  Serial.begin(9600);
  Serial.print("¥r¥nStart");

  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(15, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(17, OUTPUT);
  pinMode(18, OUTPUT);
  pinMode(22,INPUT);
```

```
  pinMode(24,INPUT);
  pinMode(26,INPUT);
  pinMode(28,INPUT);
  pinMode(30,INPUT);
  pinMode(32,INPUT);
  // pull-up for inputs
  digitalWrite(22, HIGH);
  digitalWrite(24, HIGH);
  digitalWrite(26, HIGH);
  digitalWrite(28, HIGH);
  digitalWrite(30, HIGH);
  digitalWrite(32, HIGH);

  servo1.attach(7);
  acc.powerOn();
}

//moltorMoving 1:回転
//                2:回転
//                0:静止

void loop()   //繰り返し実行される部分
{
  byte msg[2];//Android から受け取るデータ

  if (acc.isConnected()) {   //Android を起動・接続する命令を送る
    //communicate with Android application
    int len = acc.read(msg, sizeof(msg), 2); //ADK 接続から読み込み

    if (len > 0) { //読み込んだデータがあれば処理する
      if(msg[0]==0x1){ // motorID = 1

        int mswitch1_state = digitalRead(22);
        int mswitch2_state = digitalRead(24);

        if (mswitch1_state == 0){ // マイクロスイッチ 1 が押された
```

```
  if (msg[1] == 0x1 && motorMoving1!=0 ){ // マイクロスイッチ 1 の方向なら止める
    motorMoving1 = 0;
    digitalWrite(2,LOW);
    digitalWrite(3,LOW);
    delay(20);
    digitalWrite(2,LOW);
    digitalWrite(3,HIGH);
    delay(130);
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    Serial.println("s");
  }
  if (msg[1] == 0x2){ // マイクロスイッチ 1 の方向でなければ動かす
    motorMoving1 = 2;
    digitalWrite(2, LOW);
    digitalWrite(3, HIGH);
  }
  Serial.println("V");
}
else if(msg[1] == 0x1){ // マイクロスイッチ 1 押されてなくてその方向への移動
  //HIGH,LOW でデジタル出力
  motorMoving1 = 1;
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  Serial.println("a");
}

if (mswitch2_state == 0){   // マイクロスイッチ 2 が押された

  if (msg[1] == 0x1){ // マイクロスイッチ 2 の方向でなければ動かす
    motorMoving1 = 1;
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    Serial.println("S");
  }
```

```
    if (msg[1] == 0x2 && motorMoving1!=0 ){// マイクロスイッチ 2 の方向なら止める
      motorMoving1 = 0;
      digitalWrite(2,LOW);
      digitalWrite(3,LOW);
      delay(20);
      digitalWrite(2,HIGH);
      digitalWrite(3,LOW);
      delay(130);
      digitalWrite(2, LOW);
      digitalWrite(3, LOW);
    }
    Serial.println("v");
  }
  else if(msg[1] == 0x2){// マイクロスイッチ 2 押されてなくてその方向への移動
    motorMoving1 = 2;
    //LOW,HIGH でデジタル出力
    digitalWrite(2,LOW);
    digitalWrite(3,HIGH);
    Serial.println("b");
  }


}
else if(msg[0]==0x2){// motorID = 2

    int mswitch1_state = digitalRead(26);
  int mswitch2_state = digitalRead(28);

  if (mswitch1_state == 0){// マイクロスイッチ 1 が押された

    if (msg[1] == 0x1 &&   motorMoving2!=0 ){// マイクロスイッチ 1 の方向なら止める
      motorMoving2 = 0;
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
      delay(20);
      digitalWrite(4,LOW);
```

```
      digitalWrite(5,HIGH);
      delay(130);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);
      Serial.println("h");
    }
    if (msg[1] == 0x2){// マイクロスイッチ 1 の方向でなければ動かす
      motorMoving2 = 2;
      digitalWrite(4, LOW);
      digitalWrite(5, HIGH);
    }
    Serial.println("k");
  }
  else if(msg[1] == 0x1){// マイクロスイッチ 1 押されてなくてその方向への移動
    //HIGH,LOW でデジタル出力
    motorMoving2 = 1;
    digitalWrite(4,HIGH);
    digitalWrite(5,LOW);
    Serial.println("c");
  }

  if (mswitch2_state == 0){   // マイクロスイッチ 2 が押された

    if (msg[1] == 0x1){// マイクロスイッチ 2 の方向でなければ動かす
      motorMoving2 = 1;
      digitalWrite(4, HIGH);
      digitalWrite(5, LOW);
      Serial.println("i");
    }
    if (msg[1] == 0x2 && motorMoving2!=0){// マイクロスイッチ 2 の方向なら止める
      motorMoving2 = 0;
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
      delay(20);
      digitalWrite(4,HIGH);
      digitalWrite(5,LOW);
```

```
        delay(130);
        digitalWrite(4, LOW);
        digitalWrite(5, LOW);
      }
      Serial.println("p");
    }
    else if(msg[1] == 0x2){// マイクロスイッチ 2 押されてなくてその方向への移動
      motorMoving2 = 2;
      //LOW,HIGH でデジタル出力
      digitalWrite(4,LOW);
      digitalWrite(5,HIGH);
      Serial.println("t");
    }

}
else if(msg[0]==0x3){ // motorID = 3

      int mswitch1_state = digitalRead(30);
    int mswitch2_state = digitalRead(32);

    if (mswitch1_state == 0){// マイクロスイッチ 1 が押された

      if (msg[1] == 0x1 && motorMoving3!=0){// マイクロスイッチ 1 の方向なら止める
        motorMoving3 = 0;
        digitalWrite(16,LOW);
        digitalWrite(15,LOW);
        delay(20);
        digitalWrite(16,LOW);
        digitalWrite(15,HIGH);
        delay(130);
        digitalWrite(16, LOW);
        digitalWrite(15, LOW);
        Serial.println("h");
      }
      if (msg[1] == 0x2){// マイクロスイッチ 1 の方向でなければ動かす
        motorMoving3 = 2;
```

```
      digitalWrite(16, LOW);
      digitalWrite(15, HIGH);
    }
    Serial.println("k");
}
else if(msg[1] == 0x1){// マイクロスイッチ 1 押されてなくてその方向への移動
    //HIGH,LOW でデジタル出力
    motorMoving3 = 1;
    digitalWrite(16,HIGH);
    digitalWrite(15,LOW);
    Serial.println("L");
}

if (mswitch2_state == 0){   // マイクロスイッチ 2 が押された

    if (msg[1] == 0x1){// マイクロスイッチ 2 の方向でなければ動かす
      motorMoving3 = 1;
      digitalWrite(16, HIGH);
      digitalWrite(15, LOW);
      Serial.println("i");
    }
    if (msg[1] == 0x2 && motorMoving3!=0){// マイクロスイッチ 2 の方向なら止める
      motorMoving3 = 0;
      digitalWrite(16,LOW);
      digitalWrite(15,LOW);
      delay(20);
      digitalWrite(16,HIGH);
      digitalWrite(15,LOW);
      delay(130);
      digitalWrite(16, LOW);
      digitalWrite(15, LOW);
    }
    Serial.println("p");
}
else if(msg[1] == 0x2){// マイクロスイッチ 2 押されてなくてその方向への移動
    motorMoving3 = 2;
```

```
      //LOW,HIGH でデジタル出力
      digitalWrite(16,LOW);
      digitalWrite(15,HIGH);
      Serial.println("t");
    }

}
else if(msg[0]==0x4){
   int i=map(msg[1],0,255,60,120);
   servo1.write(i);
}
else if(msg[0]==0x5){
   if (msg[1] == 0x1){
      digitalWrite(17, HIGH);
      digitalWrite(18, LOW);
      Serial.println("g");
   }
   if (msg[1] == 0x2){
      digitalWrite(17, LOW);
      digitalWrite(18, HIGH);
      Serial.println("h");
   }
}
else{ //  静止命令  msg[0] = 9

   if(motorMoving1 != 0){ //正転・反転→静止ならば
      //LOW,LOW でデジタル出力
      digitalWrite(2,LOW);
      digitalWrite(3,LOW);
      delay(20);

      if(motorMoving1 == 1){//正転→静止ならば
         digitalWrite(2,LOW);
         digitalWrite(3,HIGH);
         delay(130);
      }
```

```
    if(motorMoving1 == 2){//反転→静止ならば
      digitalWrite(2,HIGH);
      digitalWrite(3,LOW);
      delay(130);
    }
  }
  else if(motorMoving2 != 0){
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);
    delay(20);

    if(motorMoving2 == 1){//正転→静止ならば
      digitalWrite(4,LOW);
      digitalWrite(5,HIGH);
      delay(130);
    }
    if(motorMoving2 == 2){//反転→静止ならば
      digitalWrite(4,HIGH);
      digitalWrite(5,LOW);
      delay(130);
    }
  }
  else if(motorMoving3 != 0){
    digitalWrite(16,LOW);
    digitalWrite(15,LOW);
    delay(20);
    if(motorMoving3 == 1){//正転→静止ならば
      digitalWrite(16,LOW);
      digitalWrite(15,HIGH);
      delay(130);
    }
    if(motorMoving3 == 2){//反転→静止ならば
      digitalWrite(16,HIGH);
      digitalWrite(15,LOW);
      delay(130);
    }
```

```
            }
            motorMoving1 = 0;
            motorMoving2 = 0;
            motorMoving3 = 0;
            //LOW,LOW でデジタル出力
            digitalWrite(2,LOW);
            digitalWrite(3,LOW);
            digitalWrite(4,LOW);
            digitalWrite(5,LOW);
            digitalWrite(15,LOW);
            digitalWrite(16,LOW);
            digitalWrite(17,LOW);
            digitalWrite(18,LOW);
            Serial.println("e");
          }//msg[0]
        }// len>0
    }//acc.isConnected()

    delay(10);
}
```

第 2 章　　最終的な Android アプリケーションのプログラム

本章では最終的にできた Android アプリケーションのプログラムを載せる。

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tk.camerawebcam"
    android:versionCode="1"
    android:versionName="1.0" >

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <uses-library android:name="com.android.future.usb.accessory" />

        <activity
            android:name=".Main"
            android:label="@string/app_name"
            android:screenOrientation="landscape" >
            <intent-filter>
                <action
        android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <meta-data
                android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
                android:resource="@xml/accessory_filter" />
        </activity>
        <activity android:name="CameraPreview" >
        </activity>
```

```
        <activity android:name="JoystickView" >
        </activity>
        <activity android:name="ServoControl" >
        </activity>

        <uses-library android:name="com.android.future.usb.accessory" />

        <activity android:name="MainActivity" >
        </activity>
        <activity android:name="AdkLedOnOffProjActivity" >
        </activity>
    </application>

    <uses-sdk
        android:minSdkVersion="13"
        android:targetSdkVersion="8" >
    </uses-sdk>

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>

</manifest>
```

Accessory_filter.xml

```
<resources>
    <!-- ADK ボードのファームウェアで指定した名前と一致させる  -->
    <usb-accessory manufacturer="AndroidCamera" model="DC1" version="1.0" />
</resources>
```

Color_stateful.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"><color android:color="#FF8C00" />
    </item>
```

```
    <item android:state_pressed="false"><color android:color="#30C0C0C0" />
    </item>
</selector>
```

Seekbar.xml

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@android:id/background">
        <shape>
            <corners android:radius="10dip" />
            <gradient
                android:angle="270"
                android:centerColor="#005a5d5a"
                android:centerX="0.75"
                android:endColor="#00747674"
                android:startColor="#009d9e9d" />
        </shape>
    </item>
    <item android:id="@android:id/progress">
        <clip
            android:clipOrientation="vertical"
            android:gravity="bottom" >
            <shape>
                <corners android:radius="10dip" />
                <gradient
                    android:angle="270"
                    android:centerColor="#005a5d5a"
                    android:centerX="0.75"
                    android:endColor="#00747674"
                    android:startColor="#009d9e9d" />
            </shape>
        </clip>
    </item>
</layer-list>
```

Progress_vertical.xml

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <item android:id="@android:id/background" >
        <shape>
            <corners android:radius="10dip" />
            <gradient
                    android:startColor="#009d9e9d"
                    android:centerColor="#005a5d5a"
                    android:centerX="0.75"
                    android:endColor="#00747674"
                    android:angle="0"
            />
        </shape>
    </item>
    <item android:id="@android:id/progress" >
        <clip android:clipOrientation="vertical" android:gravity="bottom" >
            <shape>
                <corners android:radius="10dip" />
                <gradient
                    android:startColor="#009d9e9d"
                    android:centerColor="#005a5d5a"
                    android:centerX="0.75"
                    android:endColor="#00747674"
                    android:angle="0"
                />
            </shape>
        </clip>
    </item>
</layer-list>
```

cam.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
    <FrameLayout
        android:id="@+id/camera_preview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" >
    </FrameLayout>
    <RelativeLayout
        android:id="@+id/relativeLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="72dp"
        android:layout_marginTop="88dp" >
    </RelativeLayout>
    <RelativeLayout
        android:id="@+id/relativeLayout2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true" >
        <Button
            android:id="@+id/button8"
            android:layout_width="wrap_content"
            android:layout_height="60dp"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:layout_marginRight="154dp"
            android:background="@drawable/color_stateful"
            android:text="→"
            android:width="65dp" />
        <Button
            android:id="@+id/button5"
            android:layout_width="65dp"
            android:layout_height="45dp"
```

```
        android:layout_alignBaseline="@+id/button6"
        android:layout_alignBottom="@+id/button6"
        android:layout_alignParentLeft="true"
        android:text="Button" />
    <Button
        android:id="@+id/button1"
        android:layout_width="65dp"
        android:layout_height="45dp"
        android:layout_above="@+id/button4"
        android:layout_alignParentLeft="true"
        android:text="Button" />
    <Button
        android:id="@+id/button2"
        android:layout_width="65dp"
        android:layout_height="45dp"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/button3"
        android:text="Button" />
    <Button
        android:id="@+id/button10"
        android:layout_width="wrap_content"
        android:layout_height="60dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="24dp"
        android:layout_toRightOf="@+id/button2"
        android:background="@drawable/color_stateful"
        android:text="←"
        android:width="65dp" />
    <Button
        android:id="@+id/button3"
        android:layout_width="65dp"
        android:layout_height="45dp"
        android:layout_below="@+id/button1"
        android:layout_marginTop="28dp"
        android:layout_toLeftOf="@+id/button6"
```

```
            android:text="Button" />
        <Button
            android:id="@+id/button6"
            android:layout_width="65dp"
            android:layout_height="45dp"
            android:layout_alignLeft="@+id/vseekWaist"
            android:layout_below="@+id/button4"
            android:layout_marginTop="36dp"
            android:text="Button" />
        <Button
            android:id="@+id/button4"
            android:layout_width="65dp"
            android:layout_height="45dp"
            android:layout_below="@+id/button2"
            android:layout_marginTop="28dp"
            android:layout_toLeftOf="@+id/button10"
            android:text="Button" />
        <com.tokaracamara.android.verticalslidevar.VerticalSeekBar
            android:id="@+id/vseekWaist"
            android:layout_width="30dp"
            android:layout_height="250dp"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="76dp"
            android:layout_toRightOf="@+id/button5"
            android:fitsSystemWindows="false"
            android:max="255"
            android:minHeight="20dip"
            android:progress="133"
            android:progressDrawable="@drawable/progress_vertical"
            android:thumb="@drawable/seek_thumb" />
    </RelativeLayout>
</RelativeLayout>
```

Main.java

```
import java.io.FileDescriptor;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Timer;
import java.util.TimerTask;
import com.tokaracamara.android.verticalslidevar.VerticalSeekBar;
import com.tk.camerawebcam.R;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.hardware.usb.UsbAccessory;
import android.hardware.usb.UsbManager;
import android.os.Bundle;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.SeekBar;

public class Main extends Activity {
	CameraPreview cp;
	FrameLayout preview;
	private static final String TAG = "Socym.Adk";
	private static final String ACTION_USB_PERMISSION =
	"jp.co.dorobook.DC1.action.USB_PERMISSION";

	private PendingIntent mPermissionIntent;
	private boolean mPermissionRequestPending;
	private UsbManager mUsbManager;
```

```
private UsbAccessory mAccessory;
final int INTERVAL_PERIOD = 100;
Timer timer_a = null;
Timer timer_b = null;
Timer timer_c = null;
Timer timer_d = null;
Timer timer_e = null;
Timer timer_f = null;
Timer timer_left = null;
Timer timer_right = null;

private VerticalSeekBar mSeekBar;

long time = 0;
ParcelFileDescriptor mFileDescriptor;
FileOutputStream mOutputStream;

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {

 @Override
 public void onReceive(Context context, Intent intent) {
            // TODO 自動生成されたメソッド・スタブ
            String action = intent.getAction();
            if (ACTION_USB_PERMISSION.equals(action)) {
                        synchronized (this) {
                                    // Intent からアクセサリを取得
                                    UsbAccessory accessory = (UsbAccessory) intent

 .getParcelableExtra(UsbManager.EXTRA_PERMISSION_GRANTED);

                                    // パーミッションがあるかチェック
                                    if (intent.getBooleanExtra(

 UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                                                // 接続を開く
                                                openAccessory(accessory);
```

```
                        } else {
                                Log.d(TAG, "permission denied for accessory "
                                        + accessory);
                        }
                        mPermissionRequestPending = false;
                }
        } else if (UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
                // Intent からアクセサリを取得
                UsbAccessory accessory = (UsbAccessory) intent

  .getParcelableExtra(UsbManager.EXTRA_PERMISSION_GRANTED);

                if (accessory != null && accessory.equals(mAccessory)) {
                        // 接続を閉じる
                        closeAccessory();
                }
        }
  }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.a);
  cp = new CameraPreview(this);
  preview = (FrameLayout) findViewById(R.id.camera_preview);
  preview.addView(cp);
  mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);

  // オレオレパーミッション用 Broadcast Intent
  mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(
                ACTION_USB_PERMISSION), 0);

  // オレオレパーミッション Intent とアクセサリが取り外されたときの Intent を登録
  IntentFilter filter = new IntentFilter();
  filter.addAction(ACTION_USB_PERMISSION);
```

```
filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
registerReceiver(mUsbReceiver, filter);
Button a = (Button) findViewById(R.id.button1);
Button b = (Button) findViewById(R.id.button2);
Button c = (Button) findViewById(R.id.button3);
Button d = (Button) findViewById(R.id.button4);
Button e = (Button) findViewById(R.id.button5);
Button f = (Button) findViewById(R.id.button6);

Button right = (Button) findViewById(R.id.button8);
Button left = (Button) findViewById(R.id.button10);
mSeekBar = (VerticalSeekBar) findViewById(R.id.vseekWaist);

a.setOnTouchListener(new OnTouchListener() {
        public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                        timer_a = new Timer();
                        timer_a.scheduleAtFixedRate(new TimerTask() {
                                @Override
                                public void run() {
                                        byte command = (byte) 0x1;
                                        byte value = (byte) 0x1;
                                        sendCommand(command, value);
                                }
                        }, 0, INTERVAL_PERIOD);

                        break;
                case MotionEvent.ACTION_MOVE:
                        break;
                case MotionEvent.ACTION_UP:
                        if (timer_a != null) {
```

```
                        timer_a.cancel();
                    }
                    command = (byte) 0x9;
                    value = (byte) 0x9;
                    sendCommand(command, value);

                    break;
                case MotionEvent.ACTION_CANCEL:

                    break;
                }

                return false;
            }
        });

        b.setOnTouchListener(new OnTouchListener() {
            public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                    timer_b = new Timer();
                    timer_b.scheduleAtFixedRate(new TimerTask() {
                        @Override
                        public void run() {
                            byte command = (byte) 0x1;
                            byte value = (byte) 0x2;
                            sendCommand(command, value);
                        }
                    }, 0, INTERVAL_PERIOD);

                    break;
                case MotionEvent.ACTION_MOVE:
```

```
                    break;
                case MotionEvent.ACTION_UP:
                    if (timer_b != null) {
                        timer_b.cancel();
                    }
                    command = (byte) 0x9;
                    value = (byte) 0x9;
                    sendCommand(command, value);

                    break;
                case MotionEvent.ACTION_CANCEL:

                    break;
                }

                return false;
            }
        });

        c.setOnTouchListener(new OnTouchListener() {
            public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                    timer_c = new Timer();
                    timer_c.scheduleAtFixedRate(new TimerTask() {
                        @Override
                        public void run() {
                            byte command = (byte) 0x2;
                            byte value = (byte) 0x1;
                            sendCommand(command, value);
                        }
                    }, 0, INTERVAL_PERIOD);
```

```
                    break;
                case MotionEvent.ACTION_MOVE:
                    break;
                case MotionEvent.ACTION_UP:
                    if (timer_c != null) {
                        timer_c.cancel();
                    }
                    command = (byte) 0x9;
                    value = (byte) 0x9;
                    sendCommand(command, value);

                    break;
                case MotionEvent.ACTION_CANCEL:

                    break;
                }

                return false;
            }
});

d.setOnTouchListener(new OnTouchListener() {
            public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                    timer_d = new Timer();
                    timer_d.scheduleAtFixedRate(new TimerTask() {
                        @Override
                        public void run() {
                            byte command = (byte) 0x2;
                            byte value = (byte) 0x2;
```

```
                                        sendCommand(command, value);
                                }
                        }, 0, INTERVAL_PERIOD);

                        break;
                case MotionEvent.ACTION_MOVE:
                        break;
                case MotionEvent.ACTION_UP:
                        if (timer_d != null) {
                                timer_d.cancel();
                        }
                        command = (byte) 0x9;
                        value = (byte) 0x9;
                        sendCommand(command, value);

                        break;
                case MotionEvent.ACTION_CANCEL:

                        break;
                }

                return false;
        }
});

e.setOnTouchListener(new OnTouchListener() {
        public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                        timer_e = new Timer();
                        timer_e.scheduleAtFixedRate(new TimerTask() {
                                @Override
```

```
                                        public void run() {
                                                byte command = (byte) 0x3;
                                                byte value = (byte) 0x1;
                                                sendCommand(command, value);
                                        }
                                }, 0, INTERVAL_PERIOD);

                                break;
                        case MotionEvent.ACTION_MOVE:
                                break;
                        case MotionEvent.ACTION_UP:
                                if (timer_e != null) {
                                        timer_e.cancel();
                                }
                                command = (byte) 0x9;
                                value = (byte) 0x9;
                                sendCommand(command, value);

                                break;
                        case MotionEvent.ACTION_CANCEL:

                                break;
                        }

                        return false;
                }
        });

        f.setOnTouchListener(new OnTouchListener() {
                public boolean onTouch(View v, final MotionEvent event) {

                        byte command;
                        byte value;
                        switch (event.getAction() & MotionEvent.ACTION_MASK) {

                        case MotionEvent.ACTION_DOWN:
```

```
                        timer_f = new Timer();
                        timer_f.scheduleAtFixedRate(new TimerTask() {
                                @Override
                                public void run() {
                                        byte command = (byte) 0x3;
                                        byte value = (byte) 0x2;
                                        sendCommand(command, value);
                                }
                        }, 0, INTERVAL_PERIOD);

                        break;
                case MotionEvent.ACTION_MOVE:
                        break;
                case MotionEvent.ACTION_UP:
                        if (timer_f != null) {
                                timer_f.cancel();
                        }
                        command = (byte) 0x9;
                        value = (byte) 0x9;
                        sendCommand(command, value);

                        break;
                case MotionEvent.ACTION_CANCEL:

                        break;
                }

                return false;
        }
  });
  // SeekBar でサーボモーターを動かす
  mSeekBar.setOnSeekBarChangeListener(new
VerticalSeekBar.OnSeekBarChangeListener() {

        public void onProgressChanged(VerticalSeekBar seekBar,
                        int progress, boolean fromUser) {
```

```
                // TODO 自動生成されたメソッド・スタブ
                if (progress > 0 & progress < 255) {
                        byte command = (byte) 0x4;
                        // value で値を決める
                        byte value = (byte) progress;
                        // データを sendCommand に送る

                        sendCommand(command, value);
                } else {
                }
        }

        public void onStartTrackingTouch(VerticalSeekBar seekBar) {
                // TODO 自動生成されたメソッド・スタブ

        }

        public void onStopTrackingTouch(VerticalSeekBar seekBar) {
                // TODO 自動生成されたメソッド・スタブ

        }
});

right.setOnTouchListener(new OnTouchListener() {
        public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                        timer_right = new Timer();
                        timer_right.scheduleAtFixedRate(new TimerTask() {
                                @Override
                                public void run() {
                                        byte command = (byte) 0x5;
```

```
                                        byte value = (byte) 0x1;
                                        sendCommand(command, value);
                                }
                        }, 0, INTERVAL_PERIOD);

                        break;
                case MotionEvent.ACTION_MOVE:
                        break;
                case MotionEvent.ACTION_UP:
                        if (timer_right != null) {
                                timer_right.cancel();
                        }
                        command = (byte) 0x9;
                        value = (byte) 0x9;
                        sendCommand(command, value);

                        break;
                case MotionEvent.ACTION_CANCEL:

                        break;
                }

                return false;
        }
});

left.setOnTouchListener(new OnTouchListener() {
        public boolean onTouch(View v, final MotionEvent event) {

                byte command;
                byte value;
                switch (event.getAction() & MotionEvent.ACTION_MASK) {

                case MotionEvent.ACTION_DOWN:
                        timer_left = new Timer();
                        timer_left.scheduleAtFixedRate(new TimerTask() {
```

```
                                        @Override
                                        public void run() {
                                                byte command = (byte) 0x5;
                                                byte value = (byte) 0x2;
                                                sendCommand(command, value);
                                        }
                                }, 0, INTERVAL_PERIOD);

                                break;
                        case MotionEvent.ACTION_MOVE:
                                break;
                        case MotionEvent.ACTION_UP:
                                if (timer_left != null) {
                                        timer_left.cancel();
                                }
                                command = (byte) 0x9;
                                value = (byte) 0x9;
                                sendCommand(command, value);

                                break;
                        case MotionEvent.ACTION_CANCEL:

                                break;
                        }

                        return false;
                }
  });

}

@Override
public void onResume() {
  super.onResume();

  // すでにストリームが開かれていた場合は何もしない
```

```
 if (mOutputStream != null) {
            return;
 }

 // 接続されている USB アクセサリのリストを取得
 UsbAccessory[] accessories = mUsbManager.getAccessoryList();

 // 接続されている USB アクセサリが無ければ null、１つ以上あれば最初の１つを取得
 UsbAccessory accessory = (accessories == null ? null : accessories[0]);

 if (accessory != null) {
            // USB アクセサリにアクセスするパーミッションがあるか確認
            if (mUsbManager.hasPermission(accessory)) {
                        // USB アクセサリへのアクセスを開始
                        openAccessory(accessory);
            } else {
                        Log.d(TAG, "not hasPermission");
                        // パーミッションの要求
                        synchronized (mUsbReceiver) {
                                    if (!mPermissionRequestPending) {
                                                mUsbManager.requestPermission(accessory,
                                                            mPermissionIntent);
                                                mPermissionRequestPending = true;
                                    }
                        }
            }
 } else {
            // USB アクセサリが接続されていないか、認識できていない
            Log.d(TAG, "mAccessory is null");
 }
}

@Override
public void onPause() {
 super.onPause();
 closeAccessory();
```

```
}

@Override
public void onDestroy() {
  unregisterReceiver(mUsbReceiver);
  super.onDestroy();
}

private void openAccessory(UsbAccessory accessory) {
  mFileDescriptor = mUsbManager.openAccessory(accessory);

  if (mFileDescriptor != null) {
                mAccessory = accessory;
                FileDescriptor fd = mFileDescriptor.getFileDescriptor();
                mOutputStream = new FileOutputStream(fd);

                Log.d(TAG, "accessory opened");

  } else {
                Log.d(TAG, "accessory open fail");

  }
}

private void closeAccessory() {
  try {
                if (mFileDescriptor != null) {
                                mFileDescriptor.close();
                }
  } catch (IOException e) {
  } finally {
                mFileDescriptor = null;
                mAccessory = null;
  }
}
```

```
public void sendCommand(byte command, byte value) {

  {

        byte[] buffer = new byte[2];

        buffer[0] = command;
        buffer[1] = value;
        Log.d("test", "rTouch");
        if (mOutputStream != null) {
              try {
                    mOutputStream.write(buffer);
              } catch (IOException e) {
                    Log.e(TAG, "write faled", e);
              }
        }
  }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  super.onCreateOptionsMenu(menu);
  menu.add(1, 0, 0, R.string.save_name);

  return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

  switch (item.getGroupId()) {

  case 1:
        cp.takeStillPicAll();
        return true;
```

```
            default:
                    return true;
            }
        }

}
```

第3章　　付加価値の最終的なプログラム

本章では付加価値として提案した無線化のプログラムを載せる。

Arduino サンプル Physical Pixel

```
const int ledPin = 13; // the pin that the LED is attached to
int incomingByte;      // a variable to read incoming serial data into
void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}
void loop() {
  // see if there's incoming serial data:
  if (Serial.available() > 0) {
    // read the oldest byte in the serial buffer:
    incomingByte = Serial.read();
    // if it's a capital H (ASCII 72), turn on the LED:
    if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
    }
    // if it's an L (ASCII 76) turn off the LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

Android サンプル SocketEx.java

```
package net.npaka.socketex;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

//ソケット通信
public class SocketEx extends Activity
    implements View.OnClickListener {
    private final static String BR=System.getProperty("line.separator");
    private final static int WC=LinearLayout.LayoutParams.WRAP_CONTENT;
    private final static int MP=LinearLayout.LayoutParams.MATCH_PARENT;
    private String meg;
    //IP アドレスの指定(1)
    private final static String IP="192.168.1.102";//★変更必須

    private TextView lblReceive;//受信ラベル
    private EditText edtSend;    //送信エディットテキスト
    private Button    btnSend;    //送信ボタン

    private Socket          socket; //ソケット
    private InputStream   in;       //入力ストリーム
    private OutputStream out;      //出力ストリーム
```

```
private boolean          error;   //エラー

private final Handler handler=new Handler();//ハンドラ

//アクティビティ起動時に呼ばれる
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    requestWindowFeature(Window.FEATURE_NO_TITLE);

    //レイアウトの生成
    LinearLayout layout=new LinearLayout(this);
    layout.setBackgroundColor(Color.rgb(255,255,255));
    layout.setOrientation(LinearLayout.VERTICAL);
    setContentView(layout);

    //送信エディットテキストの生成
    edtSend=new EditText(this);
    edtSend.setId(2);
    edtSend.setText("",TextView.BufferType.NORMAL);
    edtSend.setLayoutParams(new LinearLayout.LayoutParams(MP,WC));
    layout.addView(edtSend);

    //送信ボタンの生成
    btnSend=new Button(this);
    btnSend.setText("送信");
    btnSend.setOnClickListener(this);
    btnSend.setLayoutParams(new LinearLayout.LayoutParams(WC,WC));
    layout.addView(btnSend);

    //受信ラベルの生成
    lblReceive=new TextView(this);
    lblReceive.setId(1);
    lblReceive.setText("");
    lblReceive.setTextSize(16.0f);
    lblReceive.setTextColor(Color.rgb(0,0,0));
```

```
        lblReceive.setLayoutParams(new LinearLayout.LayoutParams(MP,WC));
        layout.addView(lblReceive);
    }

    //アクティビティ開始時に呼ばれる
    @Override
    public void onStart() {
        super.onStart();

        //スレッドの生成
        Thread thread=new Thread(){
            public void run() {
                try {
                    connect(IP,8080);
                } catch (Exception e) {
                }
            }
        };
        thread.start();
    }

    //アクティビティの停止時に呼ばれる
    @Override
    public void onStop() {
        super.onStop();
        disconnect();
    }

    //受信テキストの追加
    private void addText(final String text) {
        //ハンドラの生成
        handler.post(new Runnable(){
            public void run() {
                lblReceive.setText(text+BR+
                    lblReceive.getText());
            }
```

```
        });
    }

    //接続
    private void connect(String ip,int port) {
        int size;
        String str;
        byte[] w=new byte[1024];
        try {
            //ソケット接続(2)
            addText("接続中");
            socket=new Socket(ip,port);
            in =socket.getInputStream();
            out=socket.getOutputStream();
            addText("接続完了");

            //受信ループ(3)
            while (socket!=null && socket.isConnected()) {
                //データの受信(4)
                size=in.read(w);
                if (size<=0) continue;
                str=new String(w,0,size,"UTF-8");

                //ラベルへの文字列追加
                addText(str);
            }
        } catch (Exception e) {
            addText("通信失敗しました");
        }
    }

    //切断
    private void disconnect() {
        try {
            socket.close();
            socket=null;
```

```
        } catch (Exception e) {
        }
    }

    //ボタンクリックイベントの処理
    public void onClick(View v) {
        //スレッッドの生成
        Thread thread=new Thread(new Runnable(){public void run(){
            error=false;
            try {
                //データの送信(5)
                if (socket!=null && socket.isConnected()) {
                    // byte[] w=edtSend.getText().toString().getBytes("UTF8");
//                  byte [] bytes = buffer.getBytes("UTF-8");
                    byte[] buffer=new byte[2];
                    String ne = new String(buffer, "UTF-8");
                    byte [] bytes = ne.getBytes("UTF-8");
                    //byte[] w=edtSend.getText().toString().getBytes("UTF8");
                    Log.d(meg, ne);
                      out.write(bytes);
                      out.flush();
                }
            } catch (Exception e) {
                error=true;
            }
            //ハンドラの生成
            handler.post(new Runnable(){public void run() {
                if (!error) {
                    edtSend.setText("",TextView.BufferType.NORMAL);
                } else {
                    addText("通信失敗しました");
                }
            }});
        }});
        thread.start();
    }
```

```
}
```

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.npaka.socketex"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="17" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:screenOrientation="portrait">
        <activity
            android:label="@string/app_name"
            android:name=".SocketEx" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
```

```
            </intent-filter>
        </activity>
    </application>

    <uses-permission android:name="android.permission.INTERNET"/>

</manifest>
```

Arduino デモ機

```
#include <Servo.h>
Servo myservo1;
Servo myservo2;
int incomingByte0;          // a variable to read incoming serial data into
int incomingByte1;
void setup() {
   // initialize serial communication:
   Serial.begin(9600);
 myservo1.attach(8);
 myservo2.attach(9);
}

void loop() {
   if (Serial.available() > 1) {
      incomingByte0 = Serial.read();
       incomingByte1 = Serial.read();
      if (incomingByte0 == 'b') {
      val = map(incomingByte1, 0, 1023, 0, 180);        // scale it to use it with the servo (value between 0
and 180)
   myservo1.write(val);
      }
       if (incomingByte0 == 'c') {
      val = map(incomingByte1, 0, 1023, 0, 180);        // scale it to use it with the servo (value between 0
and 180)
   myservo2.write(val);
      }
   }
}
```

Android デモ機 SocketEx.java

```
package com.camera.simplemjpeg;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;

//ソケット通信
public class SocketEx extends Activity	{
	private final static String BR=System.getProperty("line.separator");
	private final static int WC=LinearLayout.LayoutParams.WRAP_CONTENT;
	private final static int MP=LinearLayout.LayoutParams.MATCH_PARENT;

	//IP アドレスの指定(1)
	private final static String IP="192.168.1.102";//★変更必須

	private final static String IP1="192.168.1.100";//★変更必須
	private TextView lblReceive;//受信ラベル
	private EditText edtSend;	//送信エディットテキスト
	private Button	bt1;	//送信ボタン
	private Button	bt2;
```

```
private RelativeLayout k;
private Socket           socket; //ソケット
private InputStream    in;       //入力ストリーム
private OutputStream out;      //出力ストリーム
private boolean          error;   //エラー
private SeekBar sk1;
private SeekBar sk2;
private final Handler handler=new Handler();//ハンドラ
int p=100;
long timee=0;
//アクティビティ起動時に呼ばれる
@Override
public void onCreate(Bundle bundle) {
      super.onCreate(bundle);
      setContentView(R.layout.main);
      //レイアウトの生成
        k=(RelativeLayout)findViewById(R.id.k);
      sk1=(SeekBar)findViewById(R.id.seekBar1);
      sk2=(SeekBar)findViewById(R.id.seekBar2);
      //受信ラベルの生成
      lblReceive=new TextView(this);
      lblReceive.setId(1);
      k.addView(lblReceive);
}

//アクティビティ開始時に呼ばれる
@Override
public void onStart() {
      super.onStart();
      Thread thread1=new Thread(){
            public void run() {
                  try {
                        connect(IP1,8080);
                     connect(IP,8080);
                  } catch (Exception e) {
                        }
```

```
            }
        };
        thread1.start();
    }
    //アクティビティの停止時に呼ばれる
    @Override
    public void onStop() {
        super.onStop();
        disconnect();
    }

    //受信テキストの追加
    private void addText(final String text) {
        //ハンドラの生成
        handler.post(new Runnable(){
            public void run() {
                lblReceive.setText(text+BR+
                    lblReceive.getText());
            }
       });



        sk1.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {

                @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromTouch) {
                                byte command = (byte) b;
                                byte value   = (byte)progress;
                sendcommand(command,value);
            }

                        @Override
                        public void onStartTrackingTouch(SeekBar seekBar) {
                                // TODO 自動生成されたメソッド・スタブ
                        }
```

```
                @Override
                public void onStopTrackingTouch(SeekBar seekBar) {
                        // TODO 自動生成されたメソッド・スタブ
                }
    });


    sk2.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {

      @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromTouch) {
                byte command = (byte) c;
                byte value   = (byte)progress;
            sendcommand(command,value);
        }

                @Override
                public void onStartTrackingTouch(SeekBar arg0) {
                        // TODO 自動生成されたメソッド・スタブ

                }

                @Override
                public void onStopTrackingTouch(SeekBar seekBar) {
                        // TODO 自動生成されたメソッド・スタブ
                }
                });
}

//接続
private void connect(String ip,int port) {
    int size0;
     int size1;
     int ret =0;
     String str;
```

```
        byte[] w=new byte[16384];
        try {
            //ソケット接続(2)
            addText("接続中");
            socket=new Socket(ip,port);
            //socket=new Socket(ip,port);
            in =socket.getInputStream();
            out=socket.getOutputStream();
            addText("接続完了");
            //受信ループ(3)

        } catch (Exception e) {
            addText("通信失敗しました");
        }
    }

    //切断
    private void disconnect() {
        try {
            socket.close();
            socket=null;
        } catch (Exception e) {
        }
    }


      private void sendcommand(final byte command,final byte value) {
                // TODO 自動生成されたメソッド・スタブ
                 Thread thread=new Thread(new Runnable(){
                        public void run(){

                        byte[]buffer = new byte[3];
                        buffer[0]=command;
                        buffer[1]=value;
                        buffer[2]='.';
```

```
                        try {
                            out.write(buffer);
                            out.flush();
                        } catch (Exception e) {
                                error=true;
                        }
                         }
                    });
                    thread.start();
            }
              private void add(final int text) {
              //ハンドラの生成
              handler.post(new Runnable(){
                    public void run() {
                        lblReceive.setText(text+BR+
                            lblReceive.getText());
                    }
              });
}}
```

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <RelativeLayout
        android:id="@+id/k"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <SeekBar
            android:id="@+id/seekBar1"
            android:max="255"
            android:layout_width="250dp"
            android:layout_height="50dp"
            android:layout_alignParentLeft="true"
            android:layout_marginTop="66dp" />

        <SeekBar
            android:id="@+id/seekBar2"
            android:max="255"
            android:layout_width="250dp"
            android:layout_height="50dp"
            android:layout_alignParentLeft="true"
            android:layout_below="@+id/seekBar1"
            android:layout_marginTop="14dp" />

        <com.camera.simplemjpeg.MjpegView
            android:id="@+id/mjpegView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
```

```
            android:layout_alignParentTop="true"
            android:layout_toRightOf="@+id/seekBar1" />

    </RelativeLayout>

</LinearLayout>
```

Androidmanifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.camera.simplemjpeg"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8"    />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="true">
        <activity
            android:name=".MjpegActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".SettingsActivity"
            android:label="@string/settings"
            android:windowSoftInputMode="stateHidden|adjustResize">
        </activity>
    </application>

</manifest>
```