2012 年度(平成 24 年度)

ECPⅢ

Final Report

android アプリケーションによる 自動カーテン開閉システムの開発

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

リーダー　　：G109029 河合一平太

サブリーダー：G109035 黒澤祐喜

：G109011 岩本涼平

：G109063 藤原航平

：G109066 町田匠

：G108403 菖蒲澤宗成

提出日：2013 年 2 月 12 日

目次

概要 （藤原担当）

近年，スマートフォンの出荷台数は年々増加し，スマートフォン利用者も増加傾向にある．また，スマートフォンの普及に伴い，スマートフォンを利用して家電を操作するスマート家電となるものが登場してきている．

Android OS は，スマートフォンの中で日本での市場専有率が 1 位であり，日本のスマートフォンユーザーの多くは Android 端末を利用している．

本研究では，スマート家電と Android の双方に着目し，身近にあるものを Android 端末で操作することを Android OS の利用法の 1 つとして提案する．

豊かな生活を送るための一環として，規則正しい生活リズムに整えることに着目し，身近にあるカーテンを選択した．

研究を進めるに当たり，まず 3 つにチーム分けを行った．

1 つ目は，カーテンの自動開閉機構を考案・製作する機体班．2 つ目は，Android からの信号を受け取り, モータを制御する回路班. 3 つ目は, カーテンを操作するのに必要な Android アプリケーションを作成する Android 班である．

まず，機体班は，機構を考案し，試作機の製作に取り組んだ．回路班と Android 班は連係を取り Android 端末とマイクロコンピュータである Arduino との無線接続を Bluetooth を用いて取り組んだ．

それらを踏まえ，機体班は，本作機の設計・製作し，回路班は，電流センサを用いてモータの自動停止制御を行った．Android 班は，Bluetooth を用いたリモコン機能の他に，音声認識による操作，タイマー機能，インターネット経由での遠隔操作を実現した．

本研究で製作した Android アプリケーションによる自動カーテン開閉システムを Android OS の利用法の 1 つとして提案する．

第1章　　緒言　(藤原担当)

1.1　研究背景　(藤原担当)

1.1.1　　スマートフォンの普及　(藤原担当)

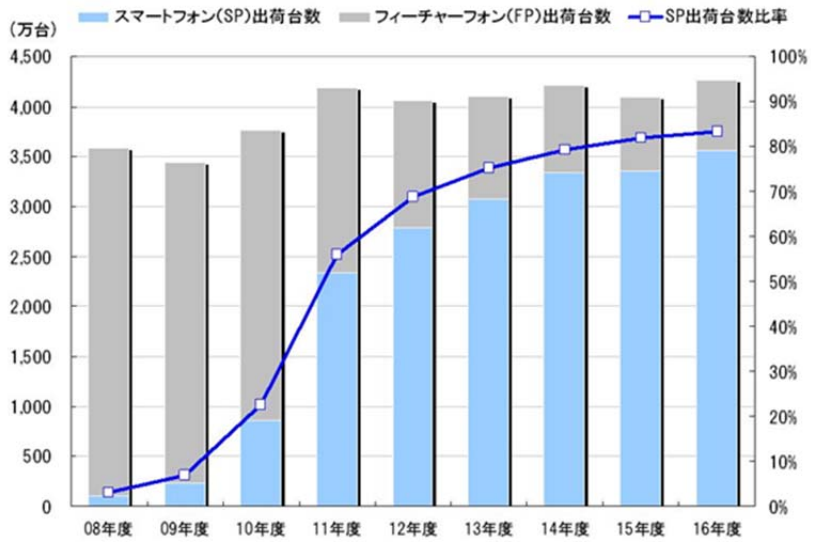


図 1-1 スマートフォン出荷台数の推移・予測

図 1-1 からスマートフォンの出荷台数が年々増加していることがわかる．またスマートフォン利用者が増加していく傾向がある [1].

2008 年から販売されているスマートフォンには，4 種類以上の OS が存在している．図 1-2 は，2011 年時点における日本で利用されている OS の市場占有率を示している．

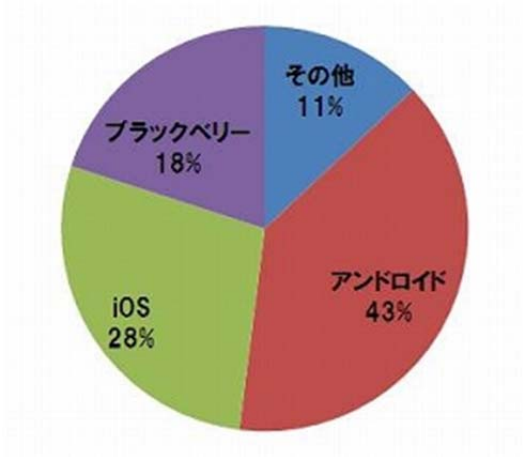


図 1-2 日本で使用されているスマートフォンの種類の内訳(2011 年度調査) [26]

1.1.2　　Android アプリケーション　(藤原担当)

Android アプリケーションとは　Android 端末で行える作業の目的別に機能を有するソフトウェアのことである.

オペレーティングシステム(OS)との対比で使われることがあり，Android や iOS 等に代表される．OS とは Android 端末の稼動に必要となるソフトウェアのことである．一方でアプリケーションはユーザーが作業目的に任意で Android 端末にインストールすることができ，アプリケーションを利用してゲームや表計算，画像編集などを手軽に行うことができる(図 1-3).

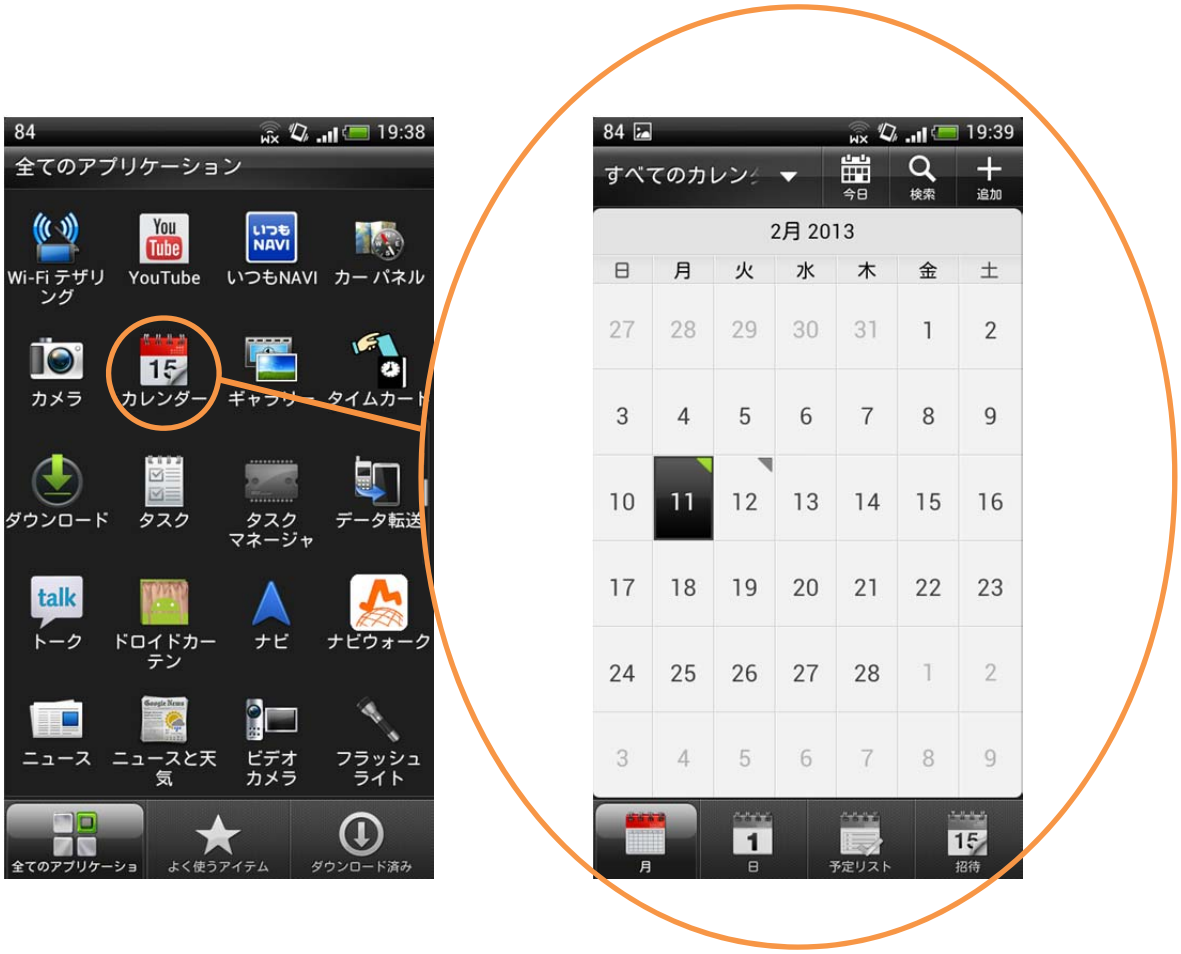


図 1-3 アプリケーション使用例

しかし近年，アプリケーションは Android 端末上だけではなく，身近にあるものと連携し，より利便性の高い製品が発売されている．その代表例としてスマート家電というものがある.

1.1.3　　スマート家電について　(藤原担当)

スマート家電とは，スマートフォンで家電を操作するものであり，近年ではスマート家電の需要が増えている．

スマート家電の果たす役割は，スマートフォンでインターネットを経由し遠隔操作することで，作業の時間の短縮化や手間の簡略化を実現する．近年では図 1．4 のように白物家電を中心に，スマートフォンとの連携で様々な機能が使用できる [2].

図 1-4 スマート家電

上記に記したスマート家電のうち，図 1-4 の 2 つのスマート家電について先行研究例を紹介する．

1.1.4　　ドラム洗濯乾燥機「NA-VX8200」Panasonic 株式会社製作(藤原担当)

図 1-5 の洗濯機は従来の機能の他に，タッチパネルやスマートフォンからインターネット利用した操作が可能である．アプリケーションから，使用する柔軟剤を選ぶことができ，必要な投入量の表示や，洗濯機や衣類の特徴を設定することで，それに適した洗濯を行うことができる [3].

この洗濯機は，スマートフォンを利用することで，衣料に応じて最適な洗い方を選ぶことができ，洗濯機を初めて使う人にもスマートフォンやタッチパネルの表示に従い，洗剤を投入という簡易的な操作が魅力である．

図 1-5 ドラム洗濯乾燥機「NA-VX8200」

1.1.5　　エアコン エアロボ X シリーズ CS-X403C2 パナソニック株式会社製作　(藤原担当)

図 1-6 のエアコンは人の活動量や間取り・日照などを検知し運転をコントロールするセンサの搭載や，外出先からスマートフォンで運転状況が確認でき電源の ON/OFF の切り替えができる [4].

このエアコンのメリットは外出先から指令を送ることで，帰宅前室内を快適な環境に整えることができる.

図 1-6 エアロボ X シリーズ CS-X403C2

1.1.6　　本研究で目指す製作物について　(藤原担当)

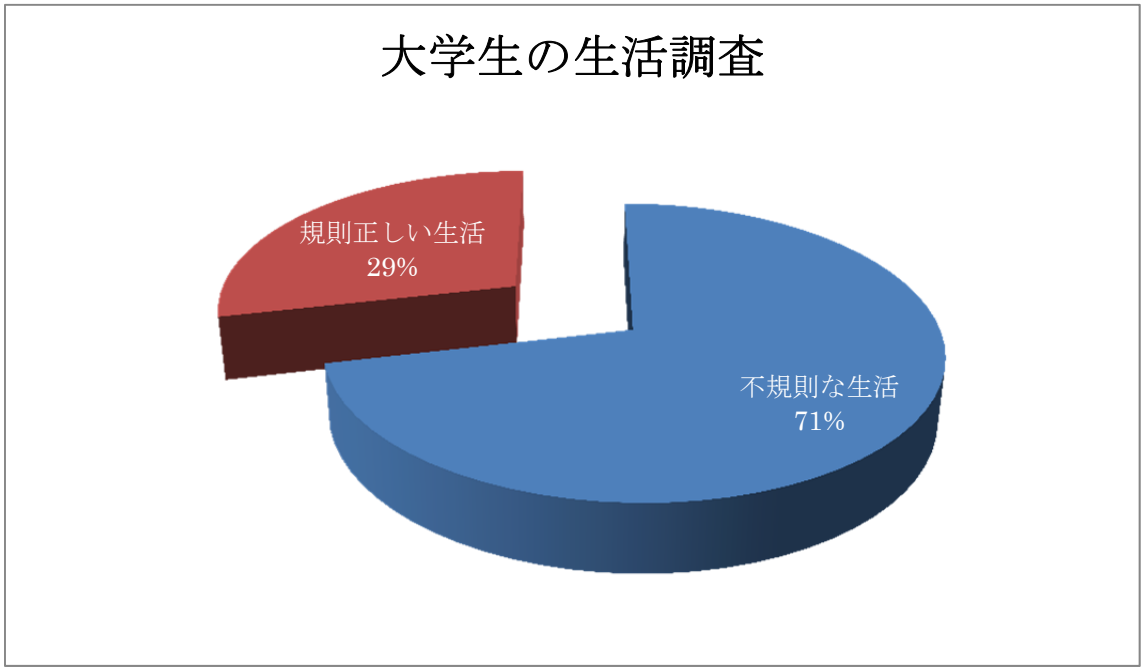


図 1-7 大学生の生活調査

前節のスマート家電は今後新たな利用の場を開拓できると考え，本研究ではスマート家電に学生独自の視点を組み合わせて検討した．その学生の視点から学生生活の身近な問題として不規則な生活をすることが多く見受けられることから（図 1-7），豊かな生活を送るための一環として，規則正しい生活リズムに整えることを目指し，本研究では規則正しい生活リズムにするために朝の起床に着目した．

朝の起床を補助する製品として一般的に目覚まし時計などがあるが，本研究は Android アプリケーションに目覚ましの役割を果たす新機能を加え，今までにない目覚ましシステムを製作することを考えた．以下は目標をまとめたものである．

・Android 端末を使い，目覚まし時計を製作する．

・今までにない目覚ましシステムである．

・規則正しい生活リズムに整える．

・スマート家電のように実際にものを操作する．

これらのことから本研究では図 1-8 のイメージのような自動カーテンの製作をすることを考えた．

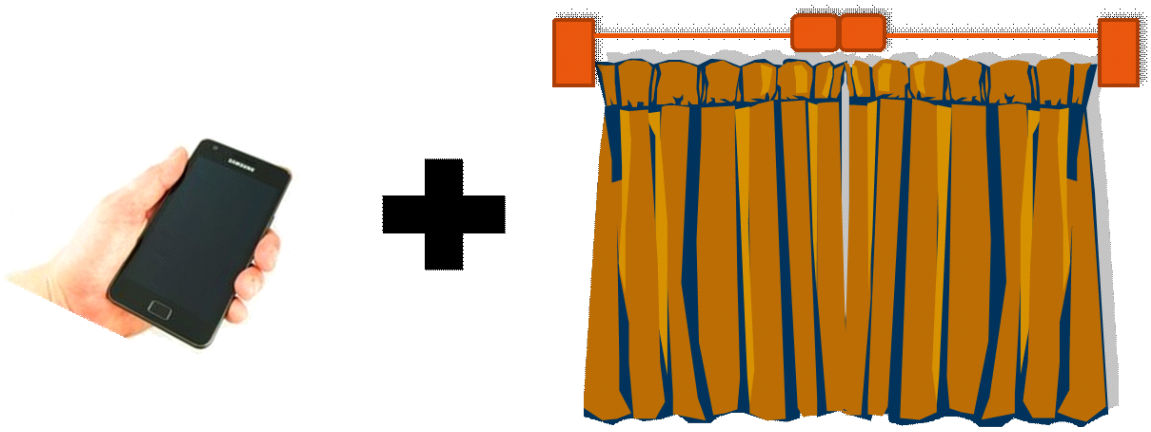


図 1-8 製作物のイメージ図

次節から本研究の太陽光のメリットをと既存の自動カーテンの問題点を説明する．

1.1.7　　太陽光　(藤原担当)

今までの目覚まし時計は音によって起床を促していたが，本研究では自動カーテンを用いることで太陽光によって起床を促すことを目指す.

太陽光を朝浴びることは生活リズムを整える重要な役割がある．地球の周期が 24 時間であるのに対して， 人間の体内時計は必ずしも 24 時間ではなく，人により 24～25 時間と言われている．1 日が 25 時間の人の場合，人間の自然な目覚めや寝入る時間は，日ごとに 1 時間ずつ遅れてゆくことになる．この体内時計の誤差をリセットするのが朝の太陽光である. そしてこれにより生体リズムを整える働きをする．しかし，現代は 24 時間社会となって生活パターンが不規則となり，太陽のリズムとは無関係の生活があたり前となっている昔ながらの日の出と共に起きて日の入りと共に眠るという生活パターンを踏襲しにくいことにより，体内時計がずれて生体リズムに異常をきたし，睡眠覚醒リズム障害[1]のような不調を感じている人が多く見られる．本研究では太陽光を利用して生活リズムを整えることが重要だと考える.

[1] 睡眠覚醒リズム障害・・・睡眠覚醒リズム障害とは昼間は活動して夜に眠るという規則正しいパターンを社会環境の変化などによって，この睡眠と覚醒のリズムが壊れてしまうこと.

1.1.8　　自動カーテンの背景と問題点　(藤原担当)

本研究ではスマート家電を参考にし，今までにない製品を考える．よって，現在ある自動カーテンにはどのような問題点があるかを以下で説明する．

図 1．9 のような現在販売されている自動カーテンは，カーテンの実寸や自宅の様々な周辺機器の配置を考える必要があるため設置が容易ではない．このことから次のことが問題であると考える [5].

・取り付け・取り外しの手間がかかる
・知識や資格が必要
・施工費など，カーテン以外の費用がかかる

これらの問題を解消するために，本研究では本来自動ではないカーテンレールに周辺機器を取り付け，Android 端末で操作する自動カーテンの開発を目指す.

図 1-9 シエルド 50

1.2　研究の目的 （藤原担当）

背景で述べたように，スマートフォンの普及に伴い，スマートフォンを使用して家電を操作する製品が増えてきている．このスマート家電を使って時間の有効活用や余計な手間を簡単にすることで，普段の生活が便利になりこれからますます流行になることが予想されている．

Android アプリケーションを開発する中で，学生独自の視点からスマートフォン利用者の利便性ある活用方法を考え，従来あるサービス及び機能の改善などのアイディアを提案することである．

これらのことから，本研究ではスマートフォンで家電を操作し生活を豊かにすることを目的とした．普段の生活を豊かにするための一環として，生活リズムを整えることを重視し，一日を過ごす上で大切である朝の起床を円滑にするためにスマートフォンを用いて自動カーテンを製作することが本研究の狙いである．

既存のカーテンの設置の手間やコストを解消するために，本研究では電動機能のみの後付けを可能にし，スマートハウスに倣ったスマートフォンでのカーテン操作を実現する．

そこで本研究の製品の目的は以下に記載する．

・既存のカーテンレールに後付け可能
・一人でも設置が可能
・自動・手動の開閉が可能であること
・長年使用できる製作物
・時間設定でカーテンの開閉が行うことが可能

1.3　本論文の構成 （藤原担当）

本論文の構成を以下に示す.

まず第 1 章では本研究に至る背景とその目的について述べる. 近年の Android OS やスマート家電の先行研究，本研究の方向性について示す.

第 2 章では本研究の概要について述べる．この章以降に続く製作物の必要性について示す.

第 3 章では製作した機体について述べる．機体の構造，設計図や使用した部品などについて詳細を示す.

第 4 章では製作した回路について述べる．回路図の設計，使用した部品やマイクロコンピュータの Arduino について詳細を示す.

第 5 章では製作した Android アプリケーションについて述べる．JAVA を用いてのプログラミング，多様な機能などについて詳細に示す.

第 6 章では機体，回路，Android を組み合わせて実際の環境下でテストした詳細について検証し，その結果を考察する.

第 7 章では本論文で得られた成果を要約し，結論とする.

第2章　　自動カーテン開閉システムの概要(町田担当)

2.1　自動カーテン開閉システムの構想　(町田担当)

本研究は Android 端末を用いて自動でカーテン開閉を行うことを目指している．**図 2-1** はそのイメージ図になる．左図のようにカーテンを開ける信号を Android 端末から送信すると，右図のようにカーテン側が受信してカーテンが開閉される．

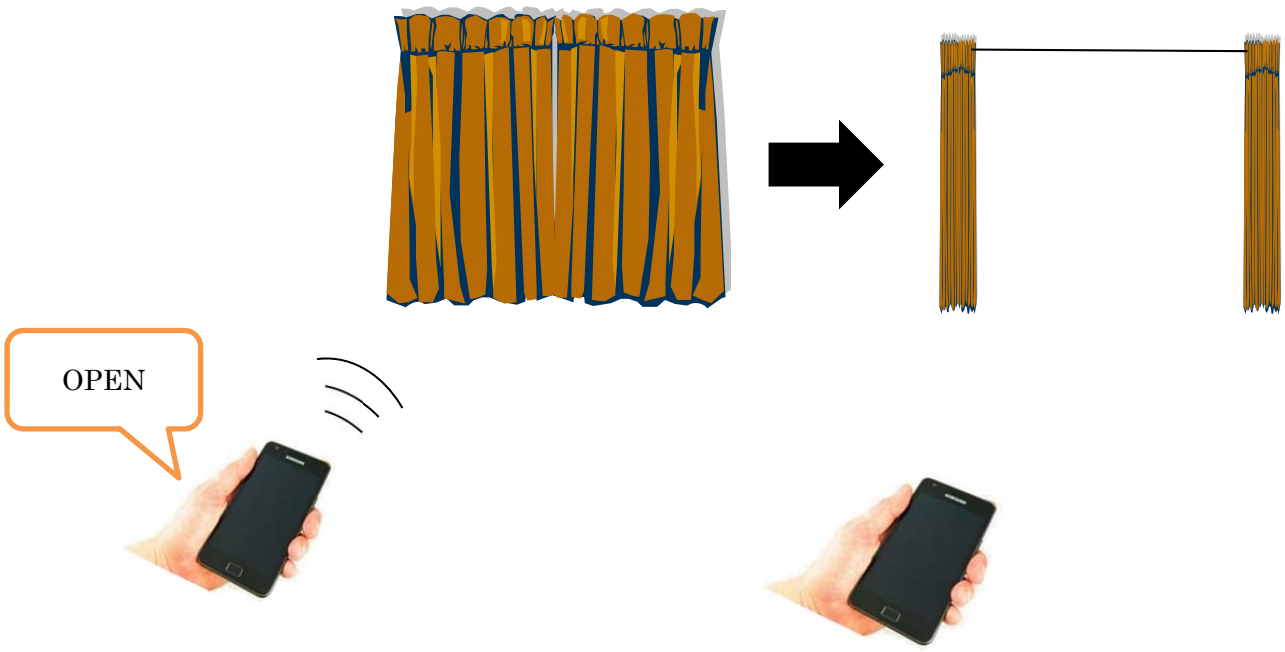


図 2-1 カーテン開閉の全体図

2.2　自動カーテン開閉システムの構造　(町田担当)

図 2-2 は自動カーテン開閉システムのより具体的な構造図を示している.

まず Android 端末からカーテン開閉の信号を無線で送信する．この無線接続には Bluetooth を使用する.

次に送信された信号はカーテンの機体（以下機体）に取り付けてある回路に送られ，その回路に搭載されている Bluetooth モジュールで受信する.

そして受信した信号は回路に搭載されているマイクロコンピュータで読み取られ，その結果モータに信号がられることで，モータが正転・反転をし，カーテンの開閉を実現する.

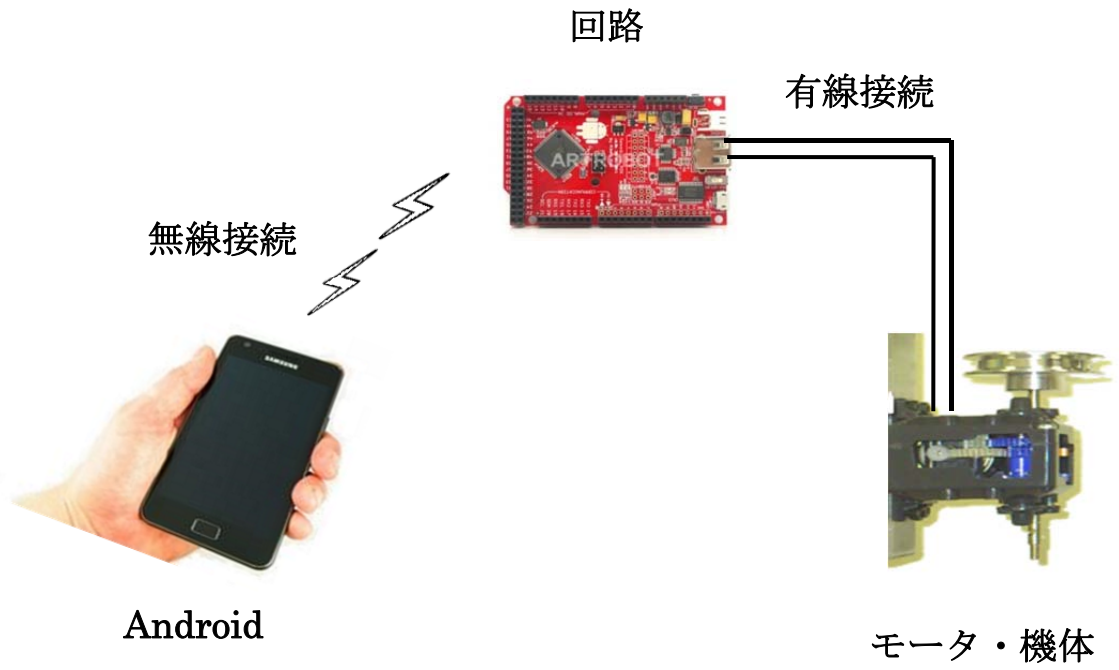


図 2-2 カーテン開閉の構造図

2.3　製作物の決定　(町田担当)

図 2.1 を受け，本研究において製作が必要だと判断した部分に個別で番号を振った図である．

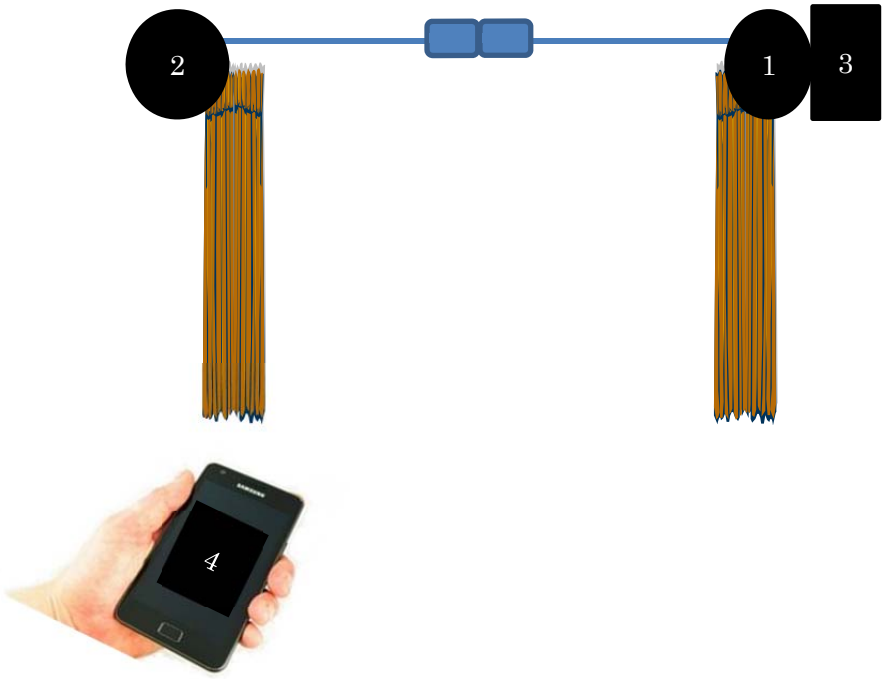


図 2-3 制作物の決定

1 は動力側の機体である．モータを含むカーテンの開閉機構を製作する．3 で製作した回路の搭載も目指す．

2 は動力機体から力を受ける受け手側の機体であり，これはカーテンを駆動する糸のテンションを常に維持する構造のために必要となる．

3 は回路であり，Android 端末からの信号をモータに伝える回路製作を行う．

4 は自動カーテン開閉システムのリモートコントロール部分であり，Android アプリケーションの製作をすることで実現する．

以上をまとめたのが表 2.1 である.

表 2.1 製作物の決定

番号	本研究で必要と考える製作物
1	動力側の機体/回路が搭載できる機構
2	受け手側の機体/ラインがたるまない機構
3	回路/C 言語を用いたプログラミング開発
4	Android アプリケーション/JAVA 言語を用いたプログラミング開発

次節からそれぞれについて詳しく解説していく.

第3章　機体（岩本，黒澤担当）

3.1　目標（岩本担当）

3.1.1　概要（岩本担当）

機体製作ではカーテン開閉のための機構を考案し，その設計の図面は CAD を用いて作成する．また，両開きのカーテンの構造を実現することを目指した．

3.1.2　動作原理（岩本担当）

本研究で考案した両開きのカーテンを開閉する仕組みを以下の図 3．1 に示す．カーテンの両側にプーリを取り付け，その片方のみにモータを取り付ける．その際，モータを含む方を動力部とし，一方を受け手部とする．

そしてラインを輪にし，カーテンを囲うようにプーリに取り付け，カーテンの中心部を下図の上側のラインと下側のラインにそれぞれに固定する．この構造により，モータを正回転させることで中央からカーテンが開閉する（図 3-1）

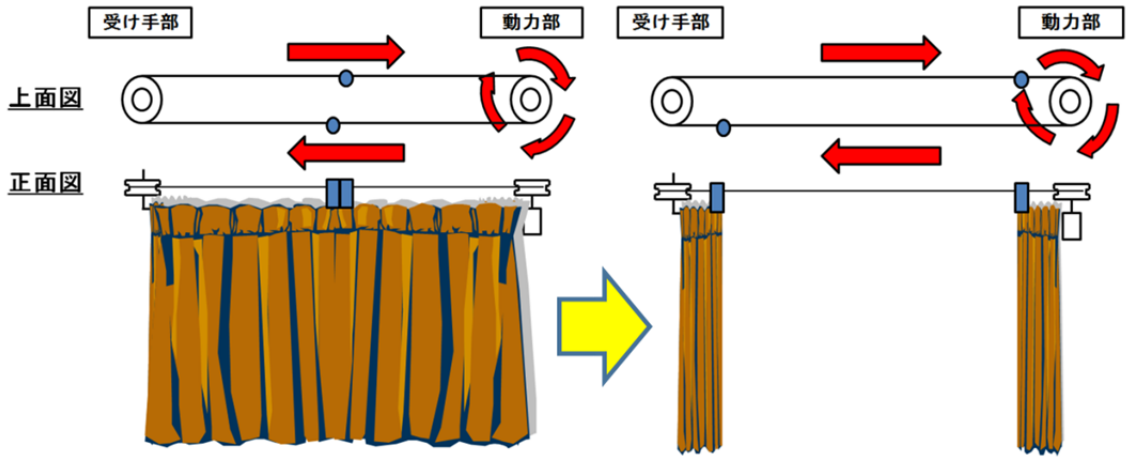


図 3-1 自動カーテンの開閉の仕組み

3.1.3　　制約条件・要求機能　(岩本担当)

本研究では，各家庭にあるカーテンやレールを活かし後付で自動カーテンにするために一般的なカーテンの取り付け環境を考慮し製作を行う．また，窓に近いため冬場などは結露も起こりやすい．それらを踏まえ以下に製作にあたっての制約条件を示す．

1. 大きさは，縦(150㎜)×横(30㎜) ×高さ(200㎜)以内に収める．
2. 天井を考え，カーテンの上を使用しない．
3. 長年使用できる材料を使用し製作する．
4. 手動でも開閉できる構造にする．

次に動力部の役割は，モータや回路を搭載する役割を担う．そして受け手部では，ラインの脱線防止のため，圧縮バネを用いた構造によりラインに常にテンションをかけ続ける役割を担っている．以上を踏まえ両開きカーテンの開閉システムを製作するにあたり，要求機能を以下に示す．

1. モータ付近に回路を取り付ける必要があるため，回路の取り付け方と取り付け位置を考え製作を行う．
2. カーテンレールと製作した機体との設置方法を考え製作する．
3. 脱線防止のため，ラインに常にテンションをかけ続けられる仕組みを考え製作する．

これらを考慮し製作に取り組んでいく．

3.2　開発方法　(黒澤担当)

本節では，自動カーテン開閉のための機構・機体の開発プロセスについて述べていく．本研究では，図 3-2 で示すフローチャート通りに開発を進めた．このフローチャートを順に説明する．

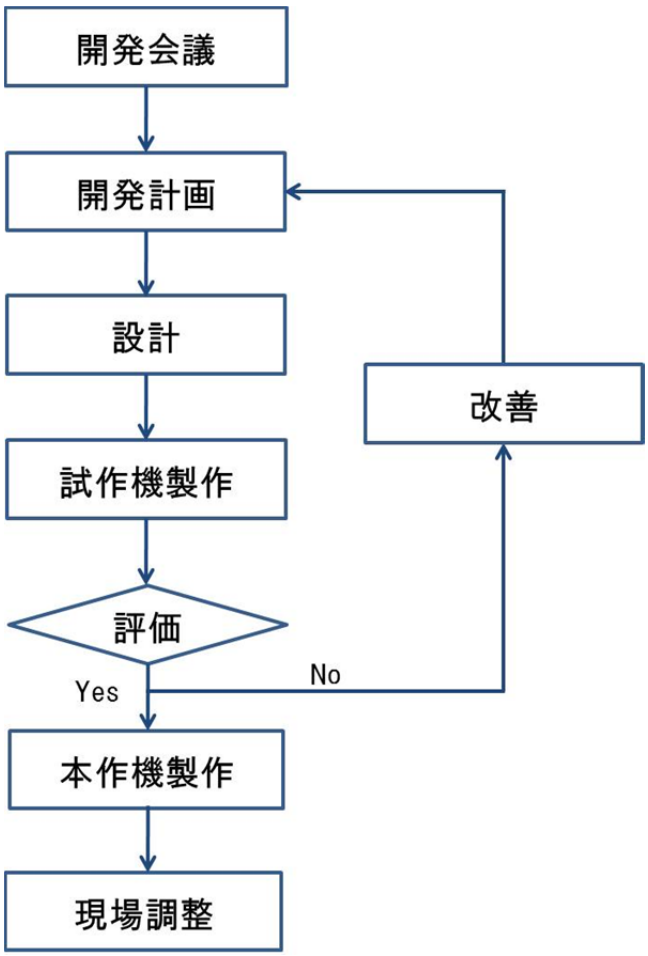


図 3-2 開発プロセスのフローチャート

まず，会議では，要求機能の案出，制約条件の設定，個人創案の提出などを行い，具体的な機体の構造・形状を決める．

次に，開発計画では機体の開発工程，日程など開発手順を決め，開発計画表を作成する．

設計では，まずポンチ絵を作成し，それを元に部品図面を作成する．部品図面が完成したら部品表を作り，規格部品の部品選定をし，設計を受けて試作機を製作する．

次に，試作機を評価し問題が生じた場合は，問題を列挙，整理しそれについての改善案を考案する．そして，開発計画に戻り，工程確認，設計，試作機の製作，評価を行い問題が解決するまでこの手順を繰り返す．問題がなければ本作機の製作を行う．最後に，組み立て，動作確認ができた本作機を実際に使用する環境に設置し動作確認を行う．本研究では，この手順に沿って機体の開発を行った．

3.2.1　　開発会議　(黒澤担当)

本研究では，まず要求機能の案出，制約条件の設定を行ったのち，週に 1 回個人創案の提出を行った．

創案の段階では，両開きのカーテンを開閉する仕組みを実現するため，モータとプーリを用いた仕組みを採用した．

モータを含む動力部と反対側に設置する受け手部に分け開発を行った．

図 3-3，図 3-4 が採用された創案図である．

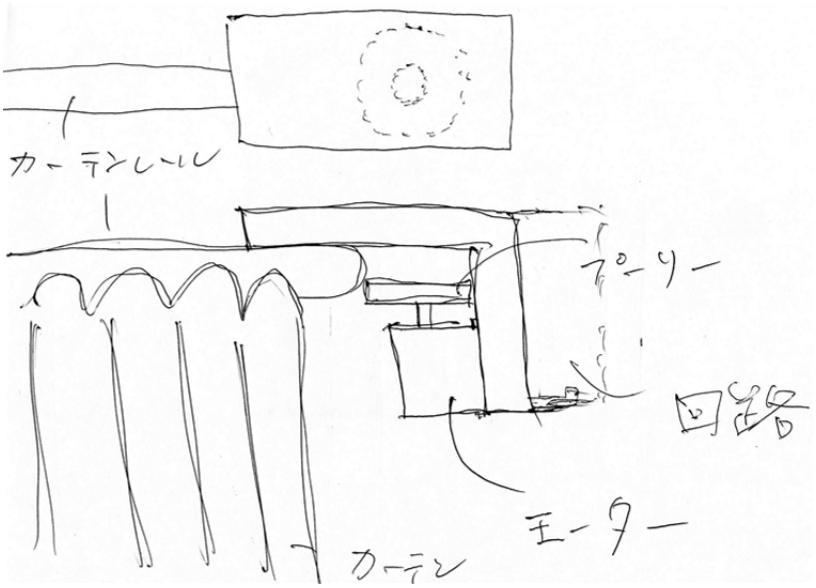


図 3-3 試作機動力部創案図

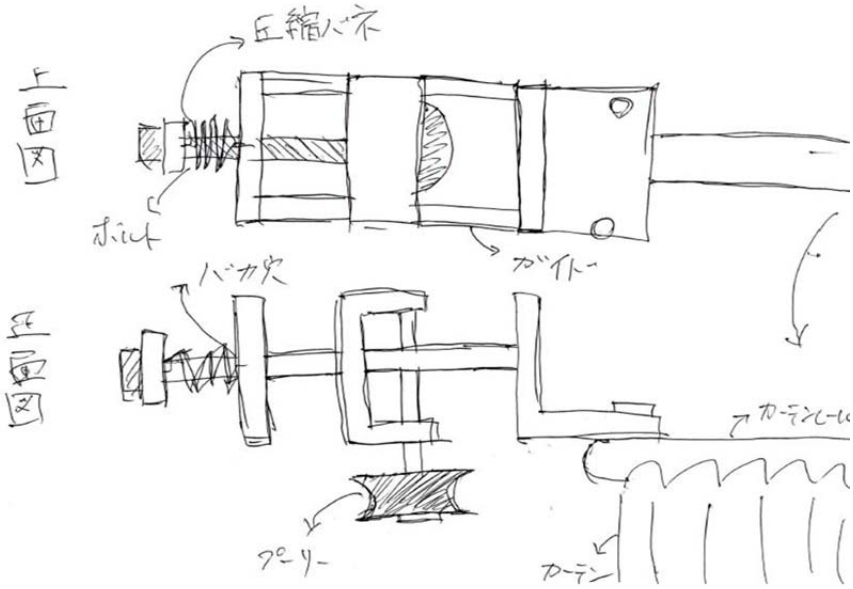


図 3-4 試作機受け手部創案図

3.2.2　　開発計画　(黒澤担当)

図 3-3，図 3-4 に示す創案を実現するための開発計画を立てた．

9 月から 10 月 2 週目までに設計し，10 月 2 週目からに試作機の製作に取り掛かり，11 月 1 週目から 2 週目にかけて評価，改善を行い，12 月に本作機の製作を行い，1 月に現場確認を行う計画を立てた．

表 3.1 が開発計画表となる．

表 3.1 開発計画表

項目＼月	9月				10月				11月				12月				1月			
週	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
試作機の設計	→				→															
試作機の製作					→				→											
評価									→											
改善									→											
本作機の設計											→		→							
本作機の製作													→				→			
現場確認																	→			

本研究は，この計画表に沿って開発を進めた

3.2.3　　試作機の設計　(黒澤担当)

まず，図 3-3，図 3-4 の創案図を受けて試作機のポンチ絵を作成した.

モータを含む動力部には要求機能として回路の搭載があるが，試作機の段階では回路がブレッドボード上で行われていたため，回路の搭載は本作機に回すことにした.

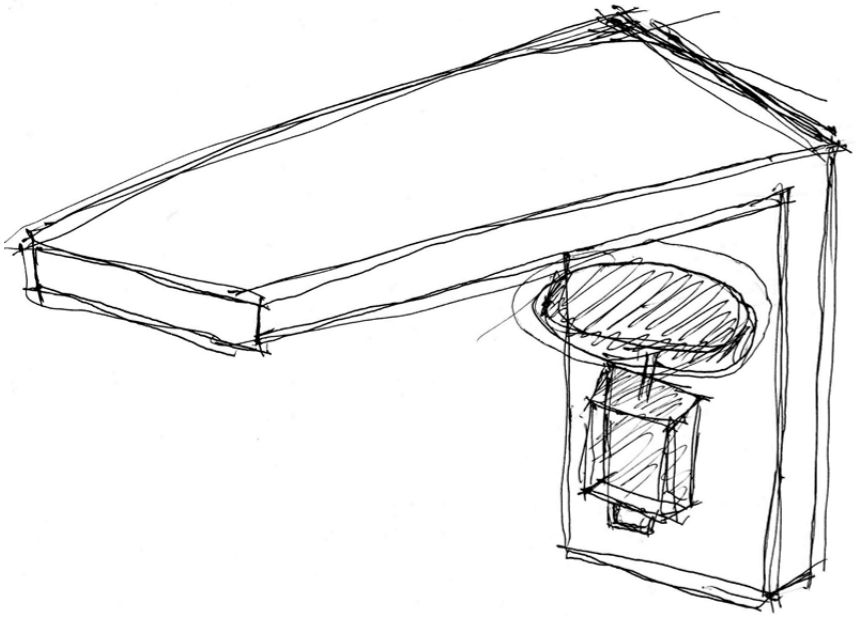


図 3-5 試作機動力部ポンチ絵

受け手部は基本的な構造は変えずに，受け手部創案図からプーリの固定方法を変更した.

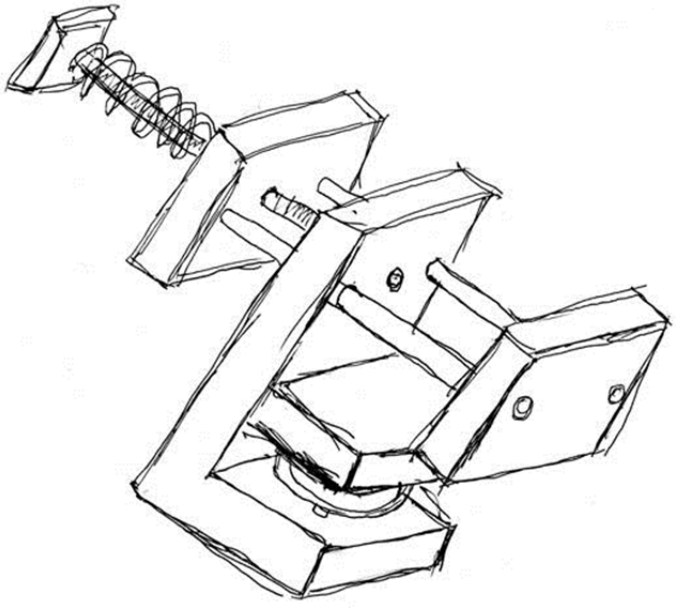


図 3-6 試作機受け手部ポンチ絵

次に，図 3-5・図 3-6 のポンチ絵を受けて各部の部品図面・組立図を Pro/E[2]を用いて作成した．

Pro/E を用いることで情報を共有することができ，また設計，アセンブリ，図面の修正を容易にでき，正確な図面を簡便に作成することができる．また，作成時間も短縮することができる．

図 3-7 が試作機の動力部の組立図となる．

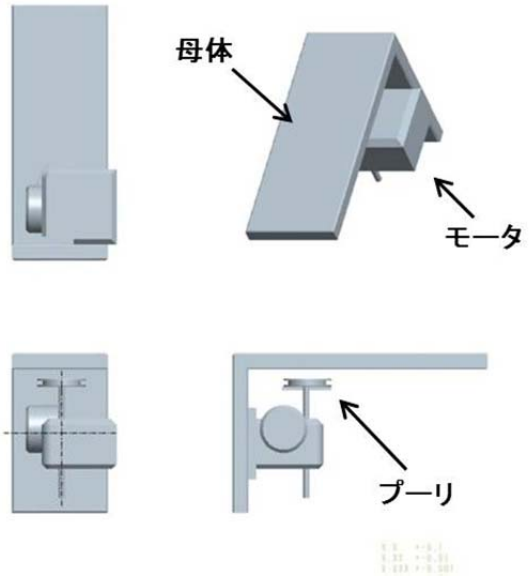


図 3-7 試作機動力部組立図

図 3-8 が試作機の動力部の母体の図面となる．

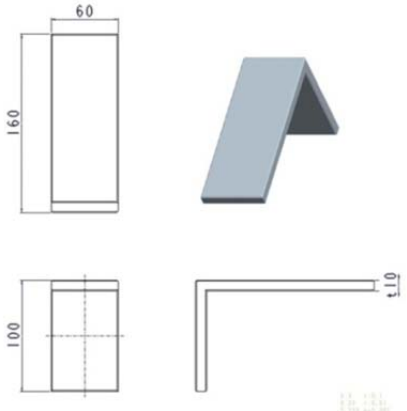


図 3-8 試作機動力部母体図面

[2] Pro/E・・・アメリカの PTC が発表をしているフィーチャーベースのパラトリックモデラーの３次元 CAD システムのソフトウェアである．

図 3-9 が試作機の受け手部の組立図となる．

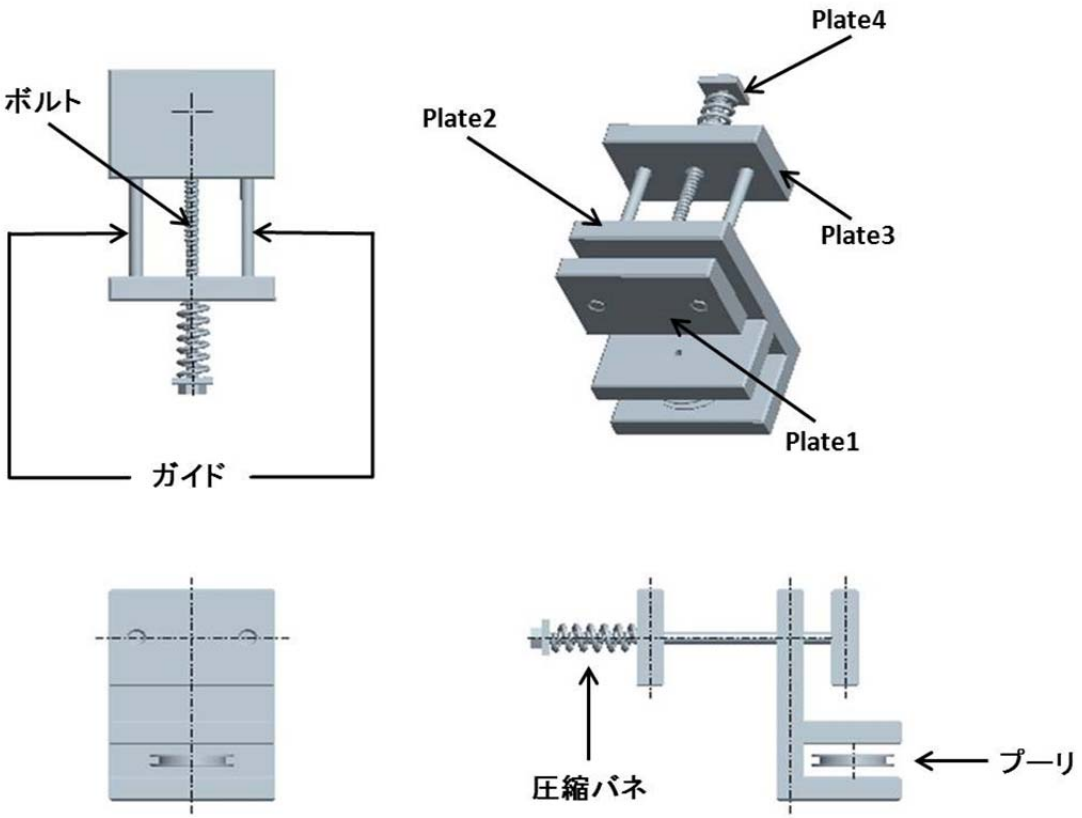


図 3-9 試作機受け手部組立図

次に，受け手部の各部の部品図面を掲載する．

図 3-10 が Plate1 の図面となる．この部品にガイドが固定され，またカーテンレールと固定される．

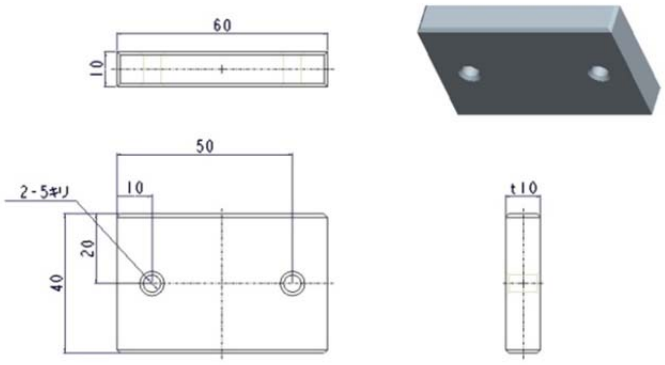


図 3-10 Plate1 図面

図 3-11 が Plate2 の図面となる．この部品にプーリが取り付けられる

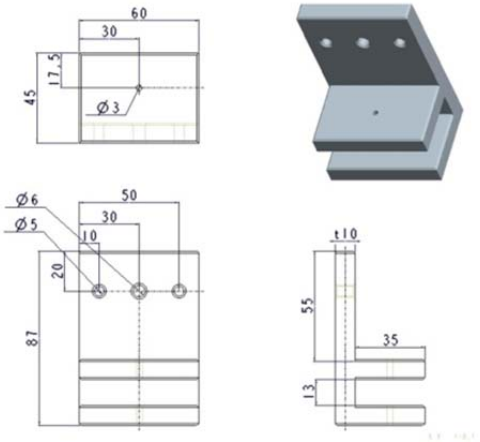


図 3-11 Plate2 図面

図 3-12 が Plate3 の図面となる．この部品にはガイドが固定される

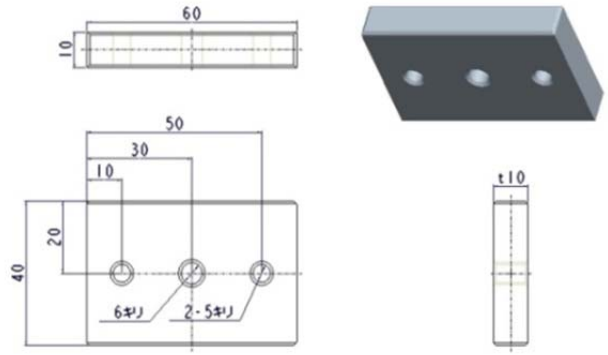


図 3-12 Plate3 図面

図 3-13 が Plate4 の図面となる．これは，圧縮バネの弾性力を受ける部品となる．

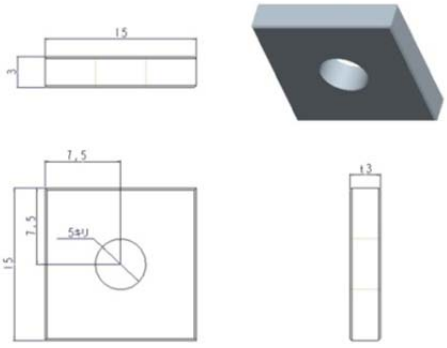
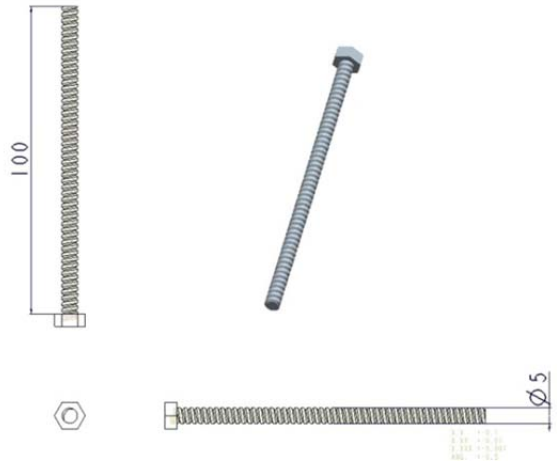
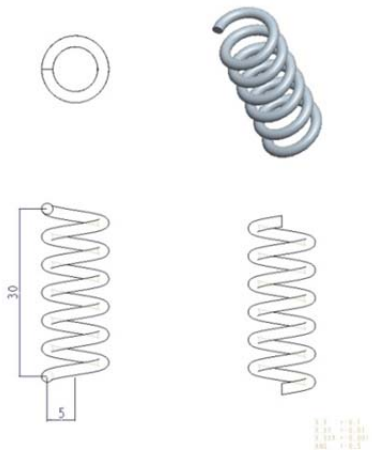


図 3-13 Plate4 図面

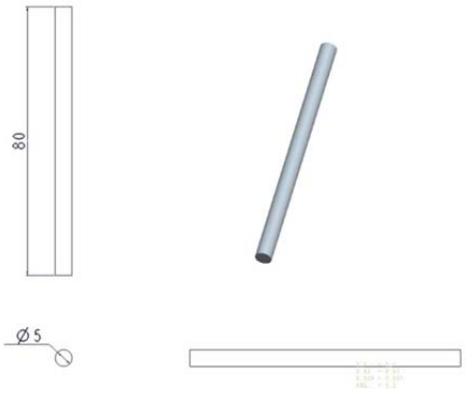
図 3-14 にその他の各種必要部品図面を示す.



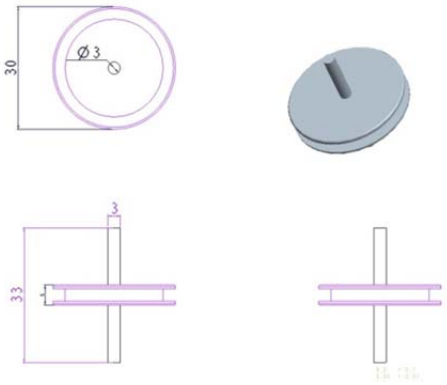
(a) ボルト図面



(b) 圧縮バネ図面



(c) ガイド図面



(d) プーリ図面

図 3-14 各種部品図面

次に，部品図面，組立図を受けて部品表を作成する．

試作機で使用する動力部の部品を表 3.2 に示し，受け手部の部品表を表 3.3 に示す．

表 3.2 試作機動力部部品表

部品名	個数	加工の要否
1．動力部 Plate1	1	要
2．モータ	1	否
3．プーリ（30mm）	1	否
4．ネジ	4	否

表 3.3 試作機受け手部部品表

部品名	個数	加工の要否
1．Plate1	1	要
2．Plate2	1	要
3．Plate3	1	要
4．Plate4	1	要
5．ボルト（5mm×100mm）	1	否
6．ガイド（5mm×80mm）	2	否
7．プーリ（30mm）	1	否
8．シャフト（3mm×40mm）	1	否
9．L 字板	2	否
10．ナット（5mm）	3	否
11．圧縮バネ（1．2mm×30mm）	1	否
12．ネジ	13	否
13．平座金	4	否
14．ナット（3mm）	4	否

要加工部品は，部品図を元に製作し，加工が不要な部品は，既製品から選定する．

モータは，TAMIYA の 4 速ウォームギヤボックス HE を使用した．これは，84:1，216:1，555．4:1，1428:1 と四段階にギヤ比を変えることができ，それぞれ 1071，1468，2306，2306(gf･cm)のトルクを出すことができる．3.3.2 で示す要求起動トルクを満たしているギヤ比 216:1 の 1468(gf･cm)で使用している(図 3-15)．

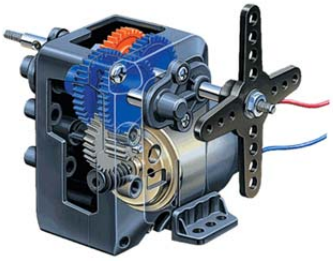


図 3-15 4 速ウォームギヤボックス

プーリは，動力部，受け手部共に TAMIYA のプーリ S セットの 30mm プーリを使用した(図 3-16)．

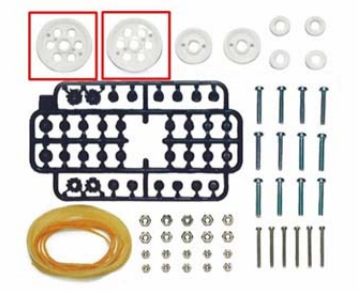


図 3-16 プーリ S セット

スプリングには，モリギン製の圧縮バネを 30mm に加工し，使用した(図 3-17)．

図 3-17 圧縮バネ

ガイドには，5mm×115mm のネジシャフトを用い，ネジ，座金等は寸法に見合ったものを適宜，使用した．

3.2.4　　試作機製作　(黒澤担当)

試作機の製作に使用した機材は下記の 3 点である.

1. (株) 高儀　スライド丸鋸　SM-100 (図 3-18)
2. 藤原産業株式会社　ラジアルボール盤　SDP-60000 (図 3-19)
3. (株) 高儀　電動卓上糸鋸盤　SS-301 (図 3-20)

図 3-18 スライド丸鋸

図 3-19 ラジアルボール盤

図 3-20 電動卓上糸鋸盤

試作機は，木材を加工して製作した.

試作機製作は，犬目校舎一号館の 3 階にある E.C.P 第二演習室を利用した.

図 3-21 が実際に製作した試作機になり，左側が受け手部で，右側が動力部となる.

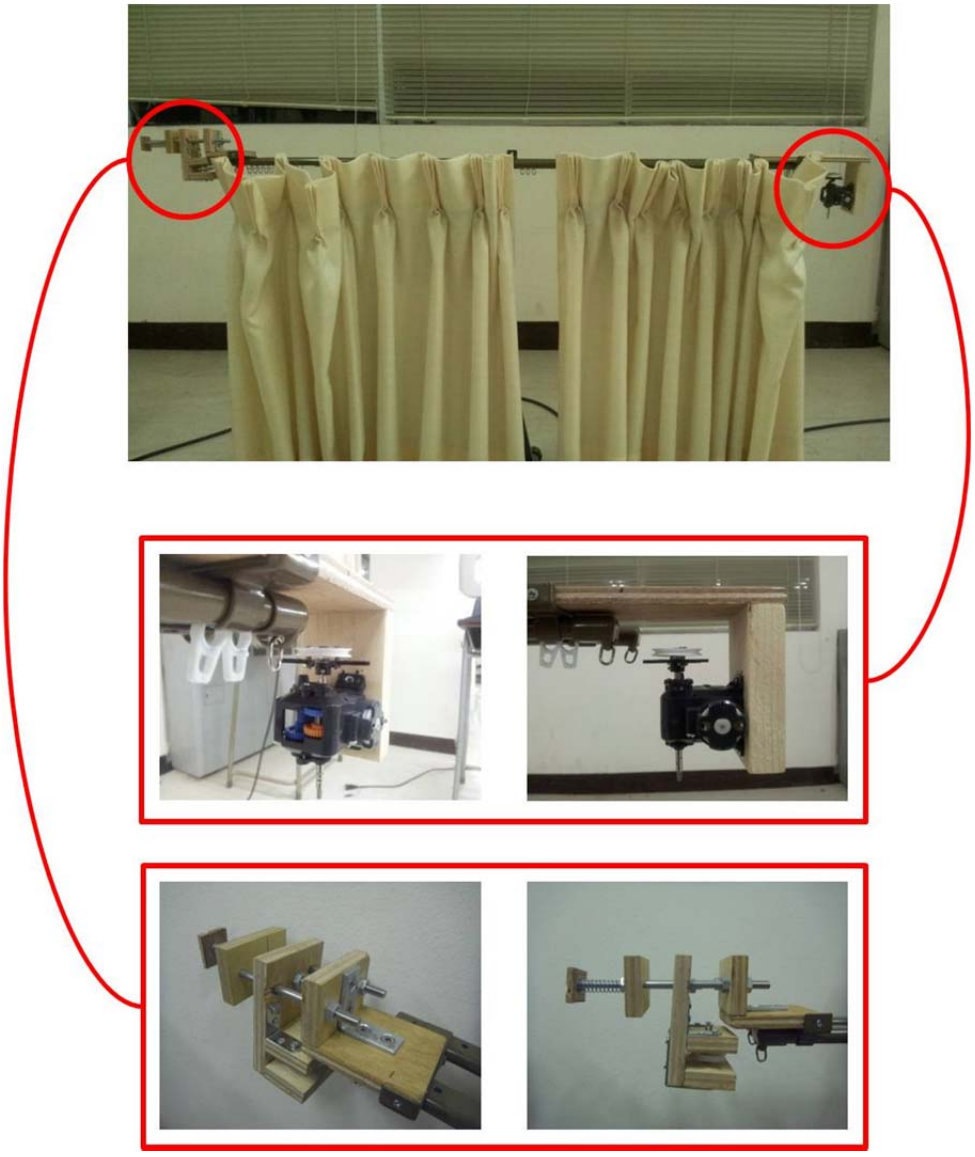


図 3-21 試作機

この試作機でカーテンを開閉できることが確認できた.

また，受け手部では，3.1.3 で示されているラインにテンションをかけ続けられる仕組みも実現できた.

3.2.5　　評価・改善　(黒澤担当)

先程の試作機を評価したところ，以下のような問題点が見つかった.

1. サイズが大きい(図 3-22).
2. 木材の材料特性上，湿気に弱く腐食してしまう.
3. 回路を搭載できていない(図 3-23).
4. 受け手部での力の釣り合いが同一作用線上にない(図 3-24).
5. 手動開閉ができない(図 3-25).

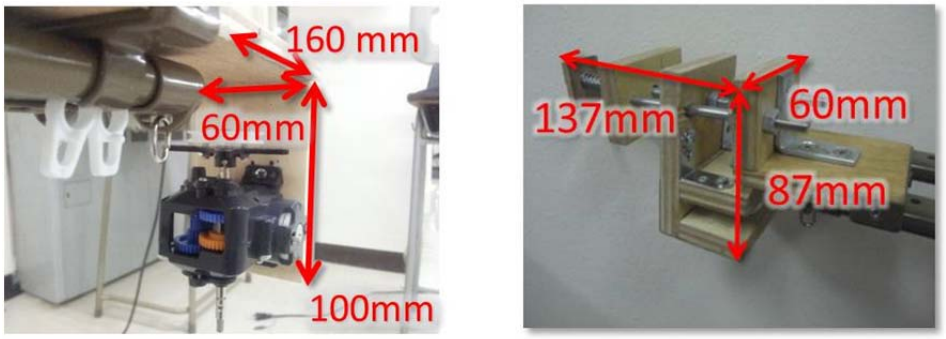


図 3-22 サイズが大きい

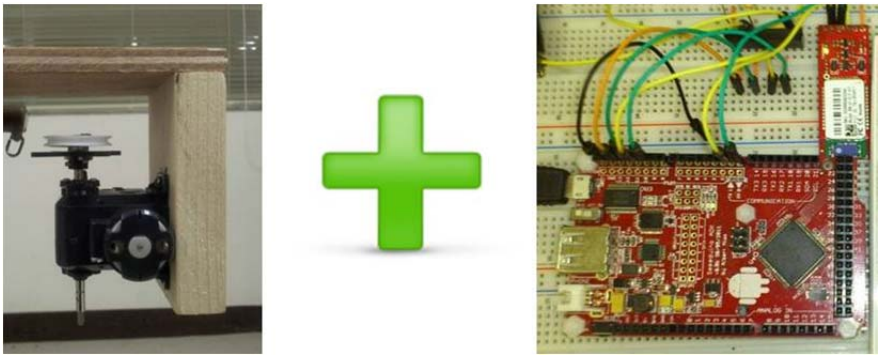


図 3-23 回路の搭載

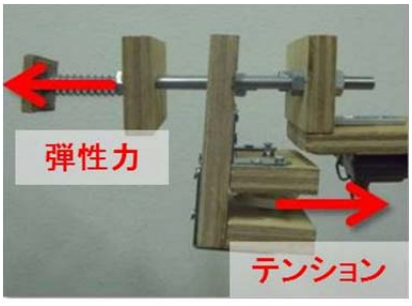


図 3-24 力の不釣合い

図 3-25 手動開閉が不可能

これらの問題を解決するために，本研究では以下に記載する改善案を案出した．

1. 犬目校舎一号館の 1 階にある E.C.P センターのワイヤー加工機を使用し，より精巧かつ緻密な加工を行うことでよりコンパクトな設計をする．
2. 金属の中でも比重が軽くて耐性が強く，加工のしやすいアルミを用いる．
3. 回路を搭載できるデザインを考案し，設計，製作を行う．
4. 力の釣り合いが取れるデザインを考案する．
5. 手動開閉が不可能な要因は受け手側の圧縮ばねの弾性力にあると考えられるため，弾性力を開放できる新しい部品を考案し，設計・製作を行う．

これらの改善案を踏まえ，本作機の製作にとりかかる．

3.3　理論値　(岩本担当)

3.3.1　　圧縮バネの弾性力　(岩本担当)

本章では受け手部に使用するバネの選定を行う．バネの選定方法は，まず必要な外径（内径）を決定し，そしておおよそ必要な自由長といくら変位したとき何 kg の荷重が必要かを決定する．ばねの弾性力はF = kxで求められる（図 3-26）.

ただしF[N]：弾性力，k[N/m]：ばね定数，x[m]：自然長からの距離，である．

設計した製作物に合わせバネを選定した結果, 必要な外径が12(mm), 自由長が32(mm)と分かった．次にバネ定数を求めていく1(cm)変位したとき17(kg)の荷重が必要であり以下の式に代入し計算する．

$$F = kx \qquad (3.\ 1)$$

$$\therefore k = 1.\ 7(\mathrm{N/mm})$$

以上より，バネ定数が1．7(N/mm)と求められた．それを基に以下のバネを購入し，使用した（表 3-4 購入したバネの詳細 ）.

表 3-4 購入したバネの詳細

規格/寸法等詳細情報	
外径(φmm)	12
線径(φmm)	1．2
総巻数(巻)	10
自由高さ(mm)	32
ばね定数(N/mm)	1．765
基準荷重時高さ(mm)	22
基準荷重(N)	17．65
許容荷重時高さ(mm)	15．5
許容荷重(N)	29．13

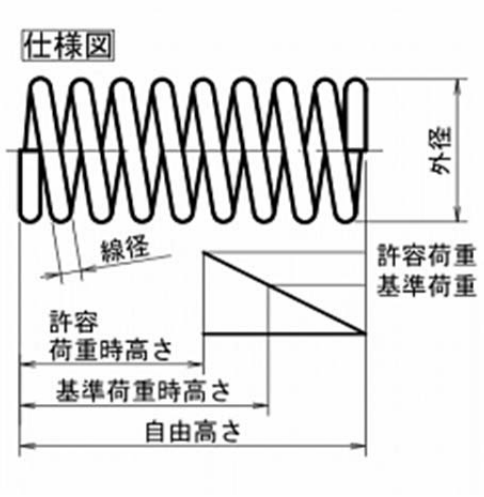


図 3-26 仕様図

3.3.2　　モータの必要トルク　(岩本担当)

本章では動力部に使用するモータのトルクに合わせて使用するギア比の選定を行う．トルクの計算は以下の式で求められる．ただし，Tm は総トルク，Ta は加速トルク，L は起動摩擦負荷トルクである．

$$Tm = Ta + L \tag{3．2}$$

次に加速トルク，起動摩擦負荷トルクは以下の式で求められる．ただし，Ta は加速トルク（kg．cm），J は負荷慣性モーメント（kg．cm^2），g は重力加速度（$980cm/sec^2$），f は動輪の等速運転速度(回転/sec)，t は加速期間の時間(sec)，W は物体の全重量（kg），μは摩擦係数（0．09），D は動輪の直径（cm）である．

$$Ta = \frac{J}{g} \times \frac{2\pi f}{t} \tag{3．3}$$

$$L = \frac{\mu WD}{4} \tag{3．4}$$

カーテンの引っ張る重さを W として計算すると以下になる．ただし，W は 4．9kg，　D は 4cm，　f は 0．78 回転/sec，　t は 0．1secである．

$$Ta = 9．8/980 \times 6．28 \times 0．78/0．1 = 0．49(kg．cm)$$
$$L = 0．09 \times 4．9 \times 4/4 = 0．441(kg．cm)$$

これから加速期間と等速期間のトルクは下記となる．

$$Tm（加速）= 0．49 + 0．441 = 931(g．cm)$$
$$Tm（等速）= 441(g．cm)$$

これより必要な起動トルクは，931（g．cm）とわかった．安全率を 1．5 倍を見て，1396．5（g．cm）以上と求められた．

以上を満たすものとして，今回 4 速ウォームギヤボックス HE のギア比 216：1（トルク 1468gf･cm）を使用した（図 3-27）．

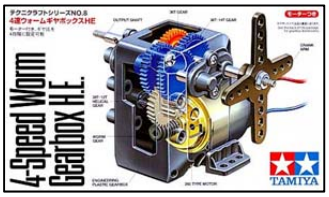


図 3-27 4 速ウォームギヤ HE

3.4　製作物の成果 （黒澤担当）

3.4.1　　本作機の設計 （黒澤担当）

まず，3.2.5 で案出した改善案を踏まえポンチ絵を作成した.

図 3-28 は本作機の動力部のポンチ絵となる．モータと回路の位置が近くなるようデザインした.

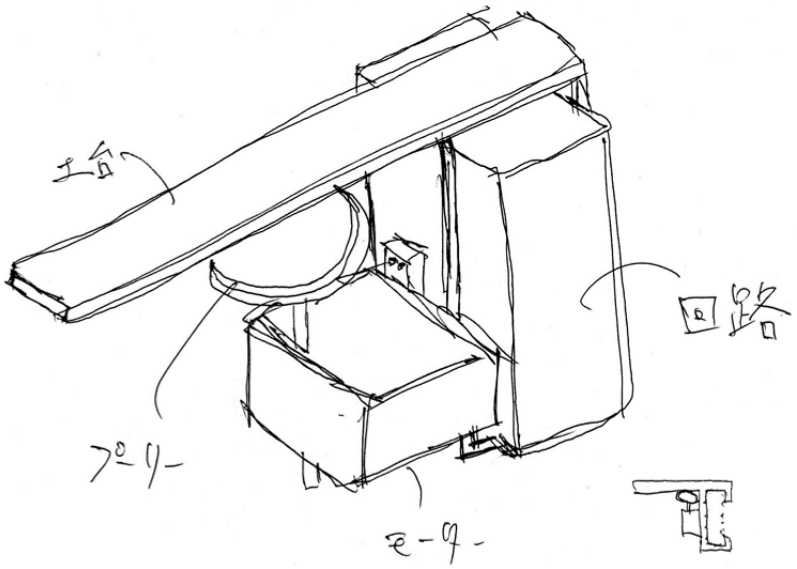


図 3-28 本作機動力部ポンチ絵

図 3-29 は本作機の受け手部のポンチ絵となる．試作機ではガイドにネジシャフトを用いたが，本作機ではガイドを箱型にすることで幅を縮小し，コンパクトな設計を目指した.

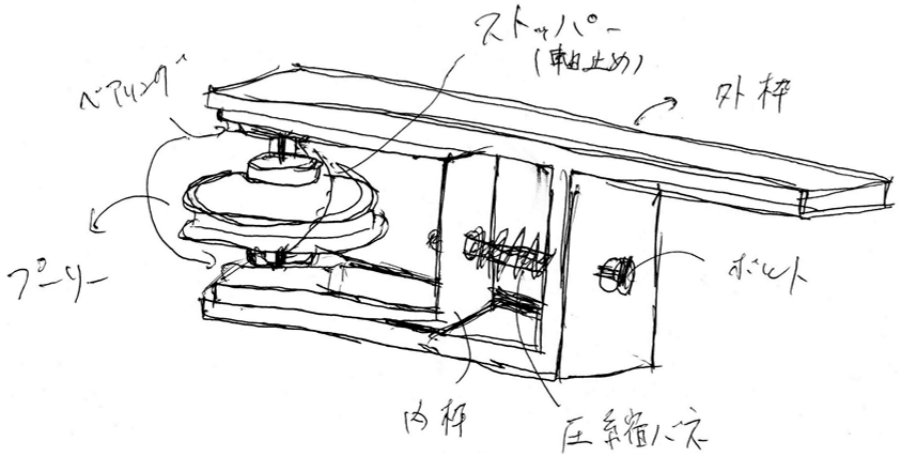


図 3-29 本作機受け手部ポンチ絵

次に，本作機動力部の組立図と部品図面を作成した(図 3-30)(図 3-31).

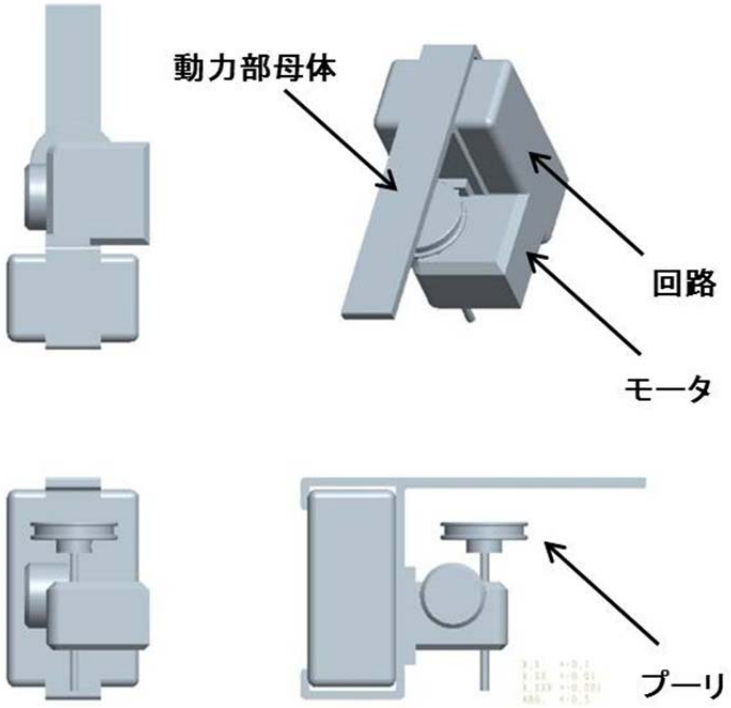


図 3-30 本作機動力部組立図

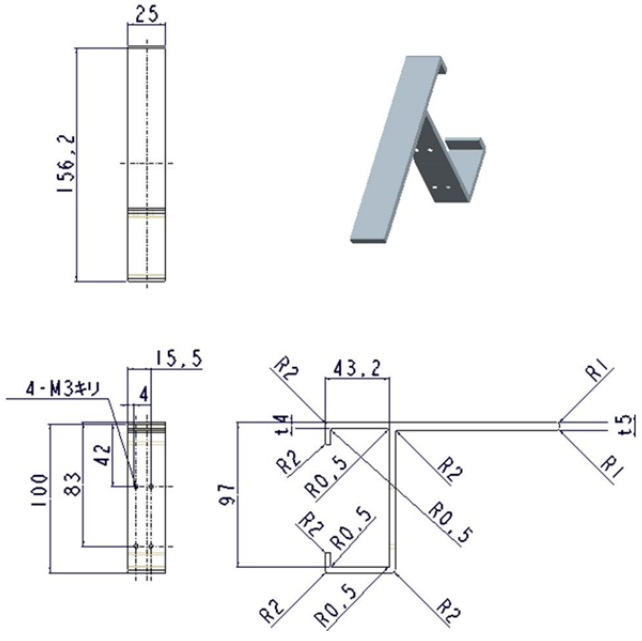


図 3-31 動力部母体図面

次に，本作機受け手部の組立図と部品図面を掲載する(図 3-32~図 3-35).

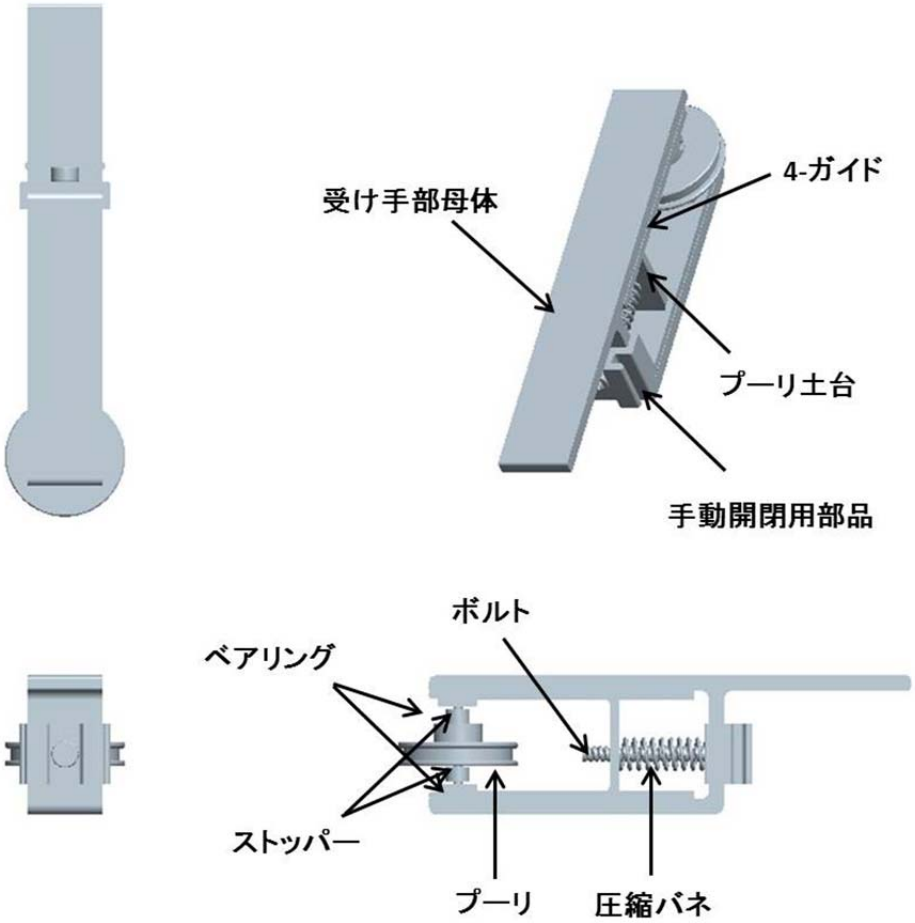


図 3-32 本作機受け手部組立図

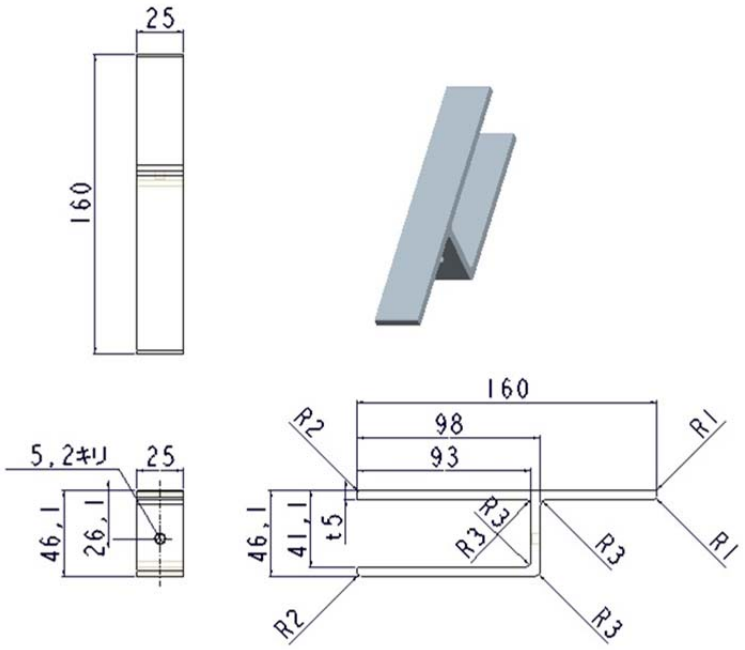


図 3-33 受け手部母体図面

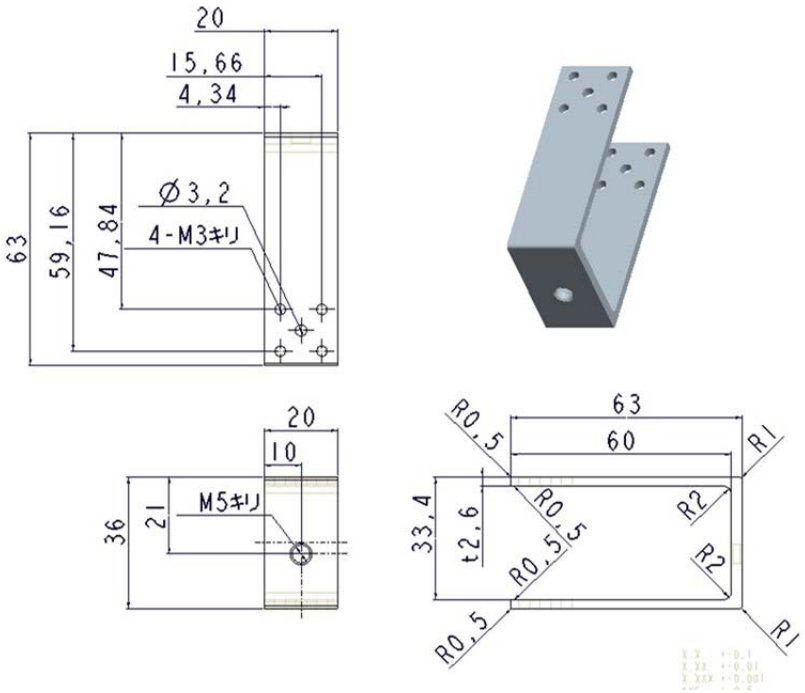
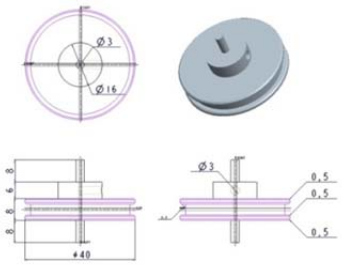
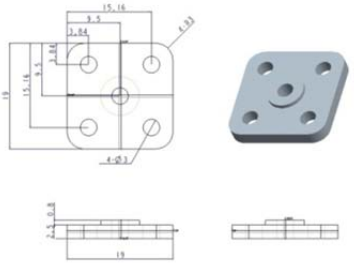


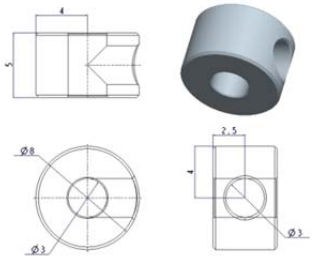
図 3-34 プーリ土台



(a プーリ図面



(b ベアリング図面



(c ストッパー図面

(d ガイド図面

図 3-35 各種部品図面

次に，組立図と部品図面を受けて部品表を作成する．

本作機で使用する動力部の部品表を表 3.5 に示し，受け手部の部品表を表 3.6 に示す．

表 3.5 本作機動力部部品表

部品名	個数	加工の要否
1．動力部母体	1	要
2．モータ	1	否
3．アルミプーリ(40mm)	1	否
4．ネジ	4	否

表 3.6 本作機受け手部部品表

部品名	個数	加工の要否
1．　受け手部母体	1	要
2．　プーリ土台	1	要
3．ガイド	4	要
4．ベアリング(M3)	2	否
5．ボルト（5mm×50mm）	1	否
6．シャフト（3mm×30mm）	2	否
7．アルミプーリ（40mm）	1	否
8．アルミストッパー（3mm）	2	否
9. 圧縮バネ（1. 2mm×30mm）	1	否
10．ネジ	8	否
11．平座金	8	否

この部品表を元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．

なお，圧縮バネとモータは，問題が無かったため試作機から引き続き同一の部品を使用する．

プーリには，レインボープロダクツのアルミプーリ 3mm×40mm を使用した．
試作機で使用したプラスチック製のプーリではカーテン開閉の回数を重ねていくとともに，変形し，たわみが出てしまったため，より強度のあるアルミ製のプーリを使用した．

また 40mm を使用したのは，径を大きくすることでラインの接地面積を増やしより滑りにくくするためである(図 3-36)．

図 3-36 アルミプーリ

ベアリングには，ORIGINALMIND 製のベアリング付きベアリングホルダ F683ZZZ を使用した(図 3-37)．

図 3-37 ベアリング

プーリの軸としてレインボープロダクツのステンレスシャフト 3mm×150mm を 3mm×30mm に加工し使用した．シャフトが腐食するとプーリの回転性が悪くなってしまうため，アルミより腐食に強いステンレスを使用した(図 3-38)．

図 3-38 ステンレスシャフト

プーリの軸の固定はレンボープロダクツ製の内径 3mm のアルミストッパーを用いた．これにより軸の上下運動を制止することができ，軸がぶれずプーリの安定した回転を実現することができる

図 3-39 アルミプーリ

3.4.2　　基本構造　(黒澤担当)

本項目では，受け手部の圧縮バネを用いた構造がどのように作用するのかについて記述する．

圧縮バネを用いた構造が必要な理由は，受け手部にはラインの脱線を防止するためである．ラインが伸びることで脱線してしまう問題をラインに常にテンションがかかっている状態を実現することで解決することを考えた．

受け手部母体とプーリ土台には，穴加工がされているが，受け手部母体にはバカ穴が開いており，プーリ土台にはネジ穴が開いている．

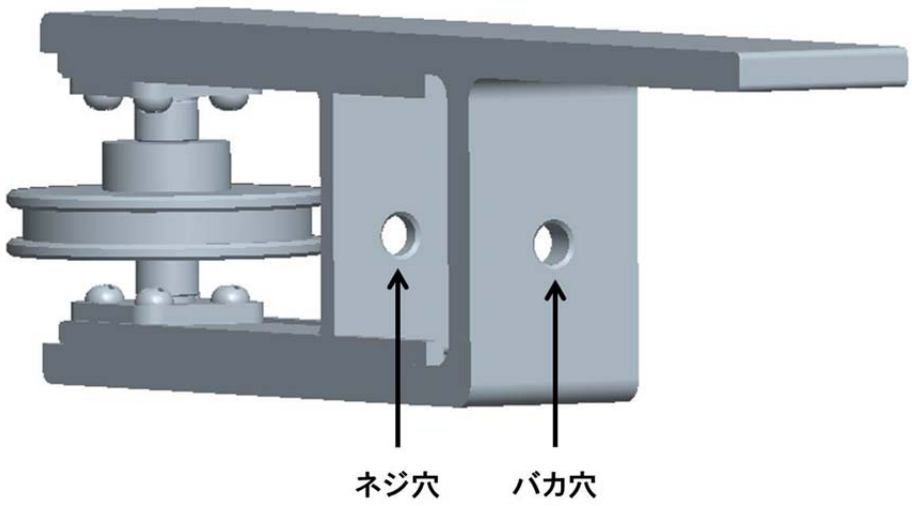


図 3-40 受け手部穴加工図

この加工によりラインからのテンションがかかっている時に図 3-41 のようにプーリ土台が引っ張られるが，圧縮バネの弾性力により押し返されるという構造になっている．

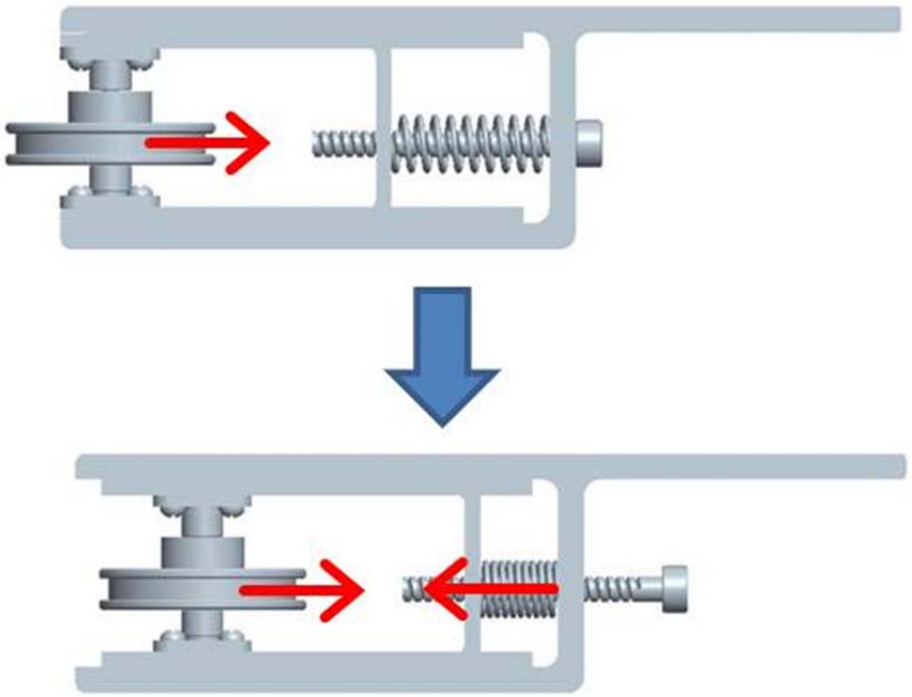


図 3-41 圧縮バネの弾性力作用図

3.4.3　　手動開閉の構造　(黒澤担当)

本項目では，試作機での手動開閉が不可能であった問題を改善するために新しく設計・製作した手動開閉用部品について記述していく．

図 3-42 が手動開閉用部品の図面となる．

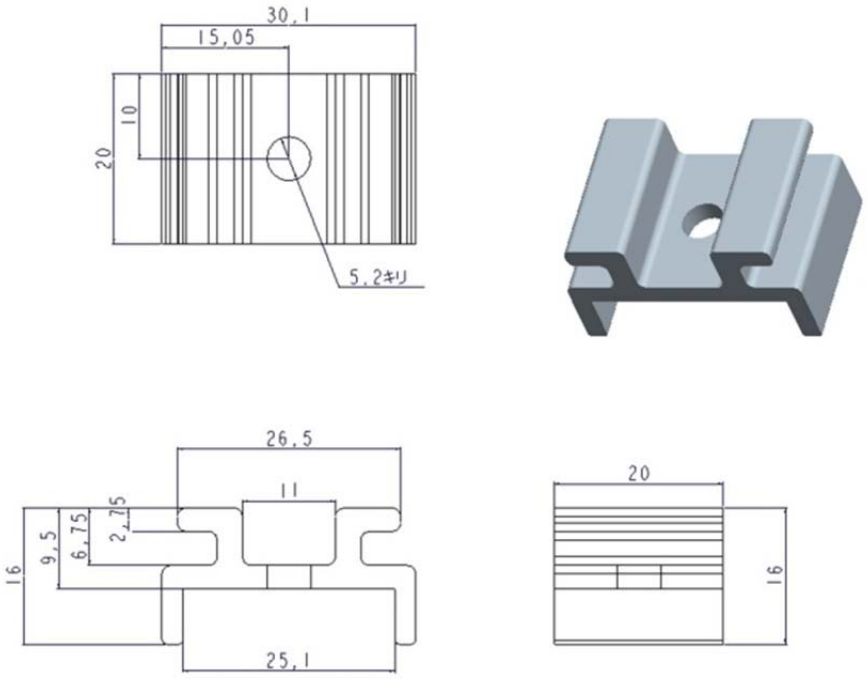


図 3-42 手動開閉部品図面

この部品を受け手部母体とボルトの間に図 3-43 のように取り付ける．

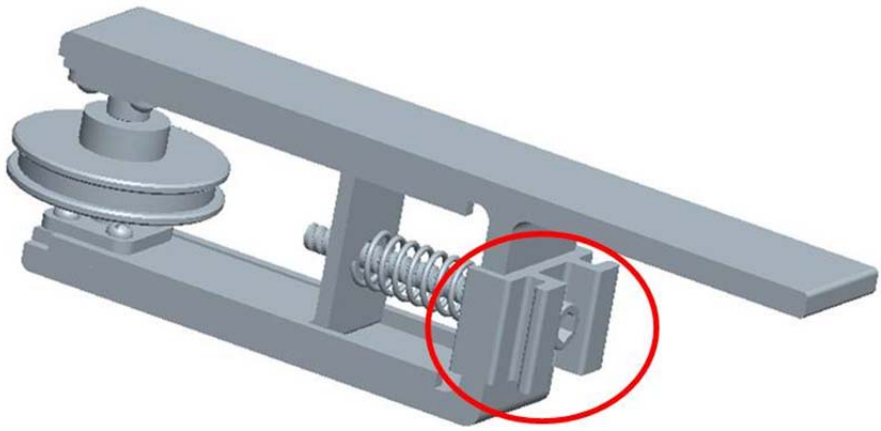


図 3-43 手動用部品の取り付け図

図 3-44 のように手動開閉用部品を引っ張り，縦にすると図 3-45 のようにプーリ土台が引っ張られ，6.5mm 前進することでラインがたわむ.

これによりラインによるテンションが開放され手動開閉が可能となる.

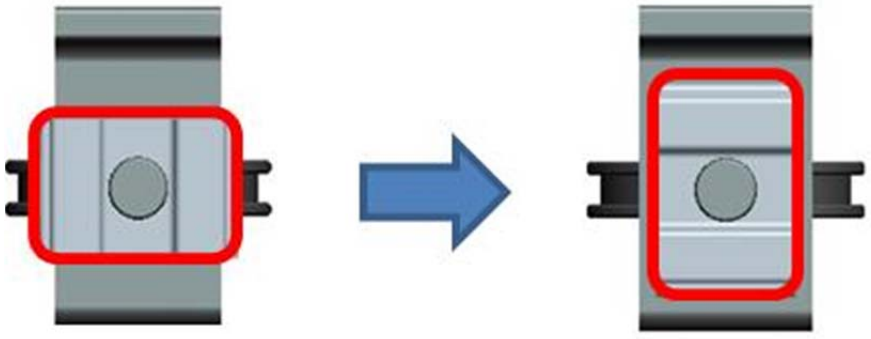


図 3-44 手動開閉の構造説明 1

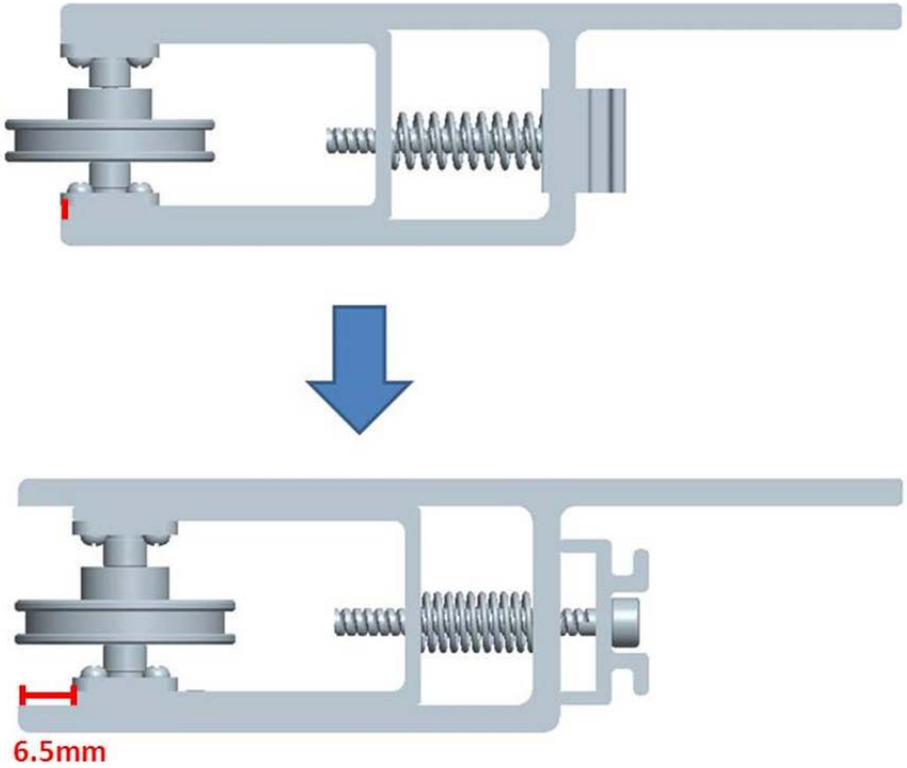
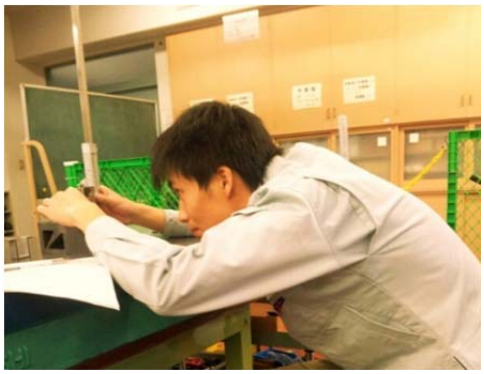


図 3-45 手動開閉の構造説明 2

3.4.4　　本作機の製作　(黒澤担当)

本作機の製作は，アルミ加工を行ったため，犬目校舎一号館の E.C.P.センターのワイヤー加工機を利用し，アルミ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で穴加工を行った(図 3-46)．

(a ワイヤー加工機



(b ハイトゲージによる罫書き

(c ボール盤による穴加工

図 3-46 本作機の製作

次に，実際に製作した本作機の写真を掲載する(図 3-47)．
右側がモータを含む動力部で，左側が受け手部となる．

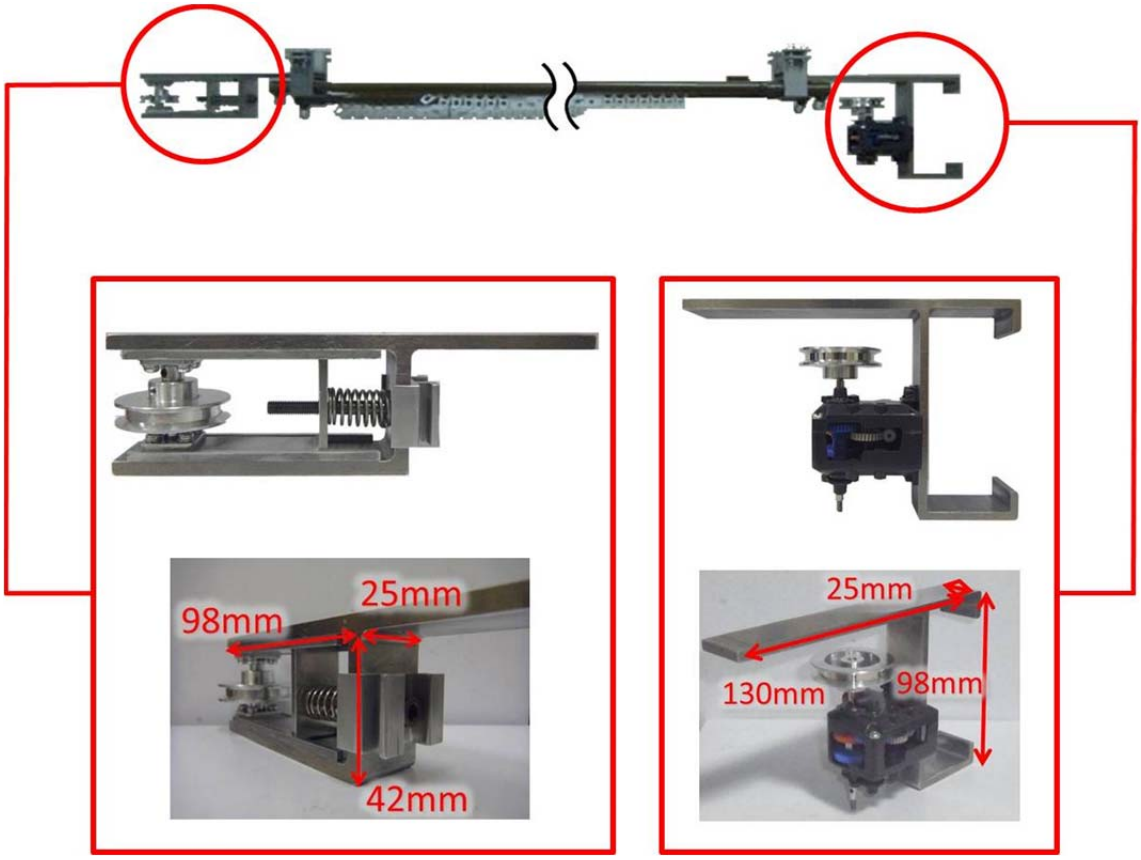


図 3-47 実物の本作機（左：受け手部，右：動力部）

この本作機でカーテン開閉の動作確認ができ，シンプルな構造かつコンパクトな設計・製作することができた．

また，動力部では回路の搭載も実現でき，受け手部では脱線防止機能，手動開閉機能も実現することができ，力の釣り合いも取ることができた．

これらのことから，試作機を評価した時に検出された問題点は全て改善する事ができたと考えられる．

3.4.5　　取り付け方法

ここでは本作機の設置についての解説を行う．まず図 3-48 にある組立パーツの 3 つを用意する．図にある 2 つの留め具を使用し，動力部と受け手をカーテンレールに留める．留め方はドライバーを使って 4 点のねじを回し，動力部や受け手とレールの幅に応じて固定を行う．

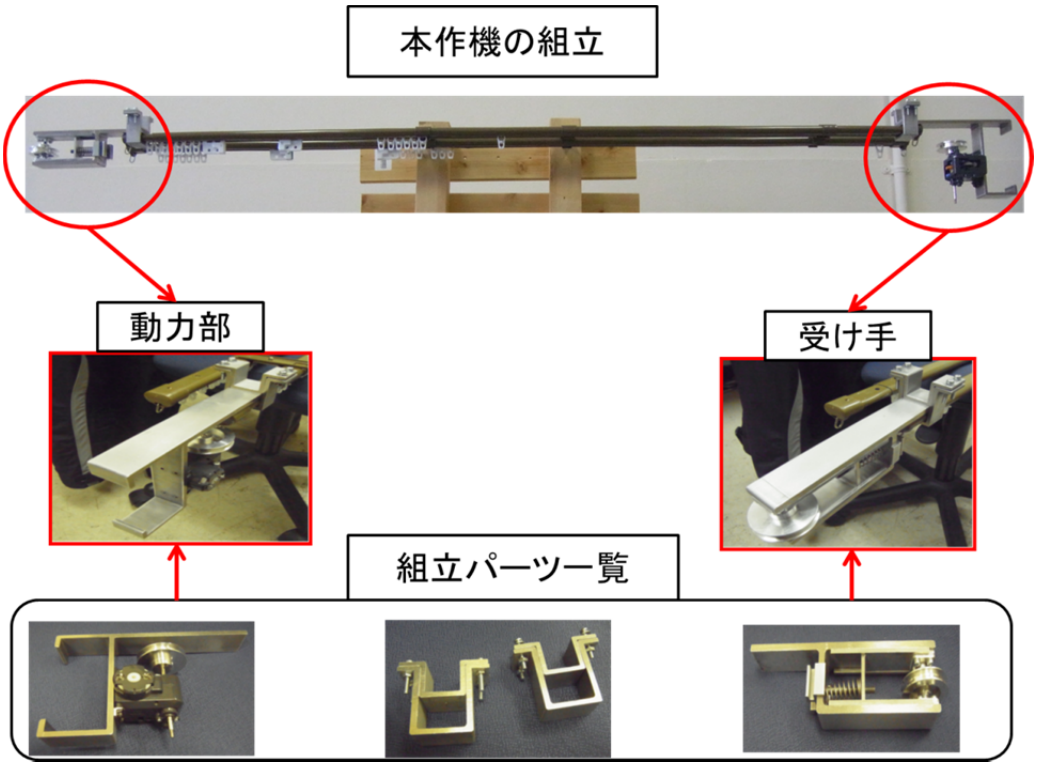


図 3-48 実機組み立てる手順の紹介

また本制作物の特徴は，既存のカーテンレールを取り外さずに設置が可能である点である．図 3-49 は研究室の近くにある 1 階と 2 階の間にある踊り場のレールに設置している様子である．一般家庭のカーテンレールであれば一人で簡単に取り付けることができる．図 3-50 は完成したカーテンであり，組み立てたものがカーテンで隠れていることから非常にコンパクトなものである．配線や回路が含まれた基板を載せることで，実機が完成する．

図 3-49 レールの取り付け

図 3-50 機体を搭載したカーテン

3.4.6　　考察　(黒澤担当)

シンプルな構造を考案でき，コンパクトな設計・製作することができた

また，要求機能や制約条件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.

第4章　回路（河合担当）

4.1　概要（河合担当）

回路はモータを電子制御することにより，カーテンを安全かつ確実に開閉する事を目的に制作する．また様々な要求機能に応じた制御を行う事で快適に使用できるような機能を組み込んでいく．

4.1.1　　要求機能（河合担当）

本製作での回路の要求機能は以下の通りである．

・モータの制御
・Android 端末との無線通信
・モータの自動停止制御
・基板への実装

このうち Android 端末との無線通信は Android 端末側が信号の送信，回路側が信号の受信部分を担当し協力して開発を進めていく．これらの要求機能については 4.2 において詳しく述べる．

4.1.2　　制約条件（河合担当）

本製作における制約条件は以下の通りである

・要求機能の実現
・電源は家庭用コンセントを使用する
・回路の大きさを縦 90mm，横 62mm，厚さ 26mm 以下にする
・機体の動力部に搭載する

このうち回路の大きさについては，回路を 90mm，62mm，26mm のケースに入れて機体の動力部に搭載するのでケースに合わせたサイズにする必要がある．

4.2　要求機能について　(河合担当)

本節では 4.1.2 で説明した要求機能について詳しく説明していく.

4.2.1　　モータ制御　(河合担当)

モータ制御には Arduino マイコン[3]により行う．カーテンはモータの回転によって開閉を行うので，モータの回転方向を変える正転・反転制御が要求される．　図 4-1 がそのイメージ図である.

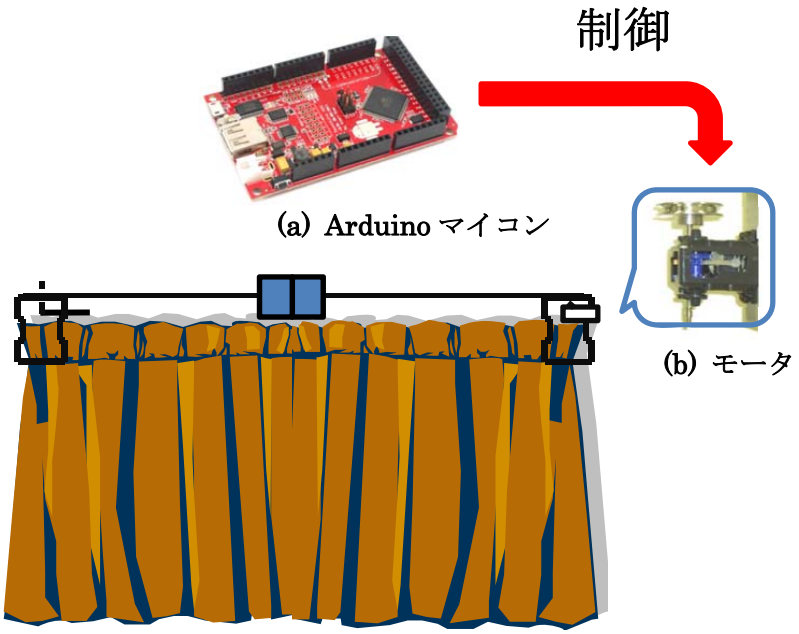


図 4-1　モータ制御のイメージ図

[3] マイクロコンピュータの略で CPU，メモリ，入出力回路等を一つの集積回路に格納したもの

4.2.2　　Android 端末との無線通信　(河合担当)

この機能は Android 端末から無線通信によりモータを動かしカーテンの開閉を行うというものである．本回路製作では Android 端末と Arduino マイコン間で近距離無線通信（以下 Bluetooth 通信）を行う． Bluetooth 通信を行うことで，カーテンから離れた場所でも Android 端末の操作によりカーテンの開閉を行うことが出来る．図 4-2 がそのイメージ図である．

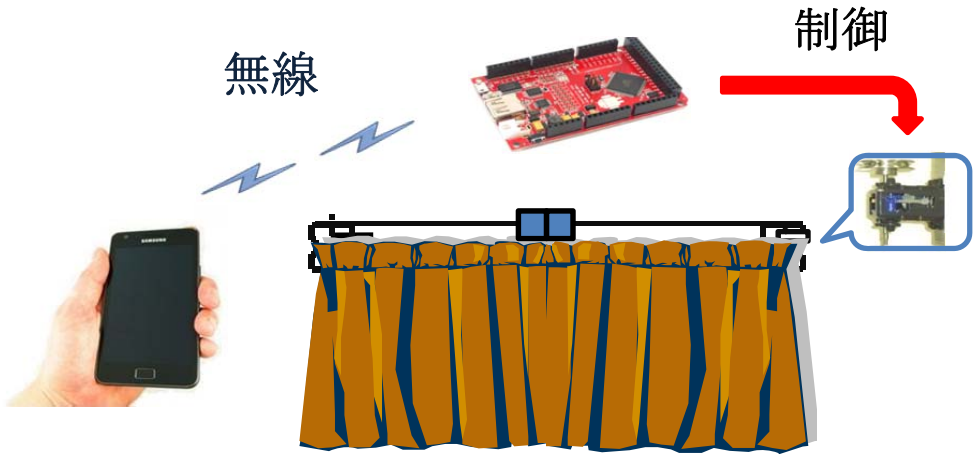
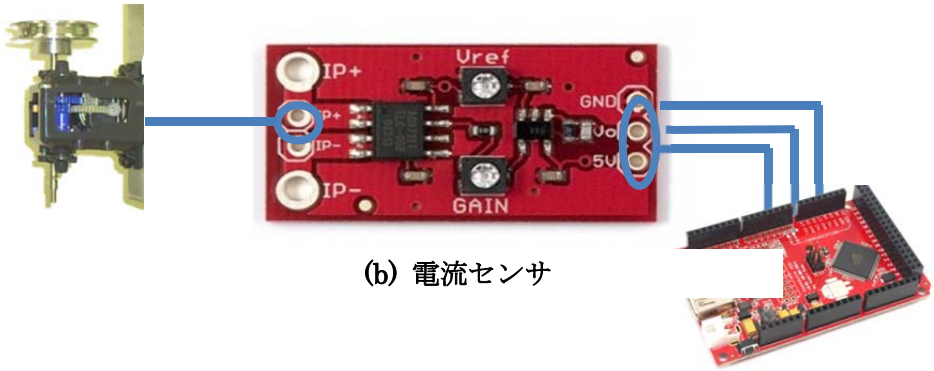


図 4-2　Android 端末との無線通信のイメージ図

4.2.3　　モータの自動停止制御　(河合担当)

モータの自動停止制御とは Android 端末からの停止信号が無くても自動的にモータを停めることである．そのためには電流センサを使用し，ある値以上の電流を検知したらモータを自動的に止めるものである．図 4-3 はイメージ図である．



(b) 電流センサ

図 4-3　モータの自動停止制御のイメージ図

4.3　開発方法　(河合担当)

回路開発の大まかな流れを図 4-4　フローチャートに示す．最初はブレッドボード[4]を使ってモータ制御，Bluetooth 通信，電流センサごとの回路，プログラムを製作していく．

次の段階は最初に作った 3 つの回路，プログラムをブレッドボード上で統合し動作確認を行う

最終段階として回路基板への実装を行っていく．基板が完成したら動作確認をする

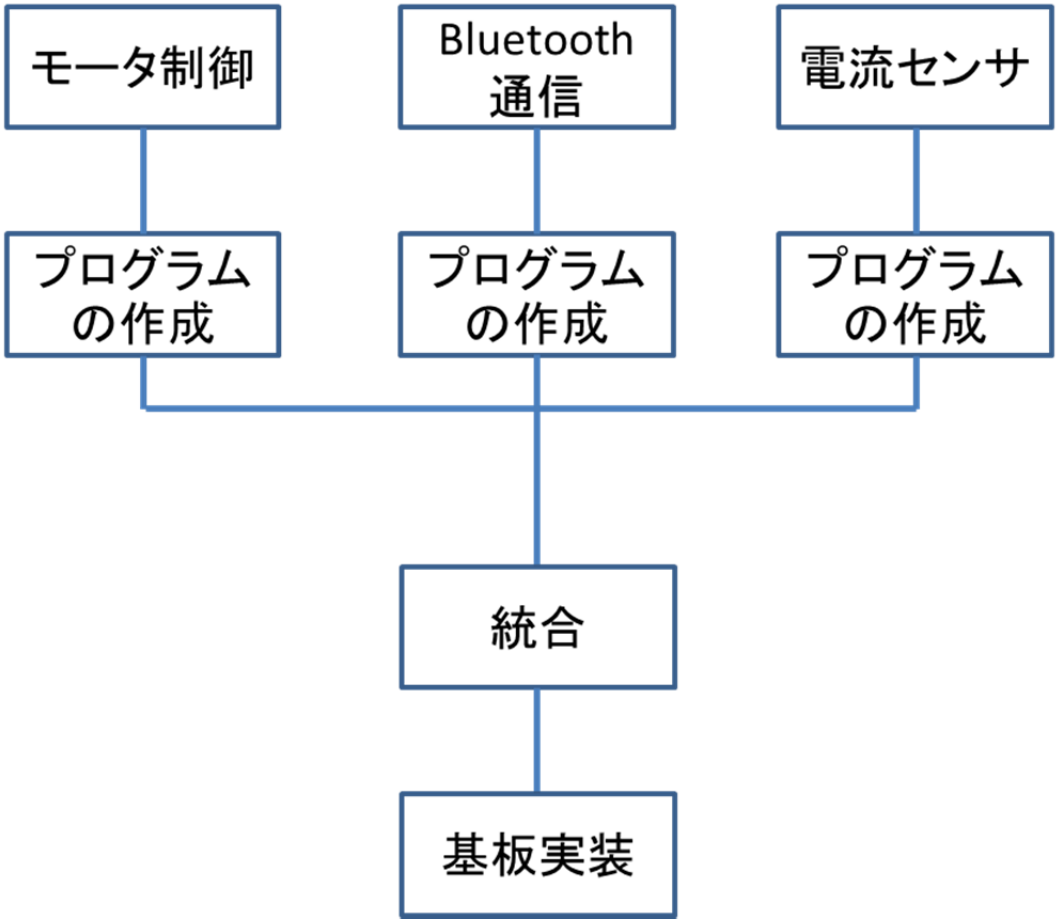


図 4-4　フローチャート

[4] 電子部品やジャンパ線をボードの穴に差し込むだけで電子回路を組むことが出来る基板

4.3.1　　使用部品　(河合担当)

本回路製作で使用した部品は表 4.1 のとおりである．表中にある 4 つのモジュールについては 4.3.2～4.3.5 で詳しく記述する．

表 4.1　使用部品

モジュール	電子部品	測定器	その他
Arduino マイコン	抵抗（10kΩ×2）	テスター	マイクロ USB ケーブル
Bluetooth モジュール	積層セラミックス コンデンサー（0．1μF）	オシロ スコープ	半田
電流センサ	ピンソケット	安定化電源	半田ごて
モータ ドライバ	ピンヘッダ		半田吸い取り機
	DC ジャッキ		メッキ線
	モータコネクタ		
	ユニバーサル基板		
	ブレッドボード		
	ジャンパー線		

4.3.2　　Arduino マイコン　(河合担当)

本回路製作で使用している Arduino マイコンは Seeed Studio 社の Seeduino ADK Main Board である(図 4-5 参照)．この Arduino マイコンは標準の Arduino のボードをベースに機能が拡張されており，16 個のアナログ入力ポート，14 個の PWM[5]出力，56 個のデジタル入出力ポートを持っている．またマイクロ USB コネクタの下にあるスイッチにより入出力ポートの電源を 5V か 3．3V のどちらかを選ぶことが出来る．この Arduino マイコンの動作電圧は 5V でマイクロ USB ケーブルを通じて供給されている．

図 4-5　Arduino マイコン

[5] パルス幅変調の事で，パルス信号を出力する時間の長さを変える事

4.3.3　　Bluetooth mate　(河合担当)

本回路製作で使用している Bluetooth モジュールは Spark Fun 社から製造されている Bluetooth mate(以下 Bluetooth モジュール)というものである. この Bluetooth モジュールを Android 端末と Arduino マイコンの無線通信用に用いる. Bluetooth モジュールの動作電圧は 5V である(図 4-6 参照).

図 4-6　Bluetooth mate

4.3.4　　電流センサ　(河合担当)

電流センサは Spark Fun 社から製造されている ACS712 という電流センサモジュールで，モータの自動停止制御に使用する．　電流センサの動作電圧は 5V で，電流センサに入力される電流の値によってアナログの電圧値が出力される．(図 4-7 参照).

図 4-7　電流センサ

4.3.5　　モータドライバ　(河合担当)

モータドライバはモータ駆動回路を搭載したICの事でモータを制御するのに必要である．今回使用しているモータドライバは東芝の「MP4212」という4V駆動のDCモータドライバでパワーMOS FET[6]を 4 つ搭載し，フルブリッジ回路を構成している．表 4-2 はモータドライバの各端子の役割について説明したものである．パワーMOSFETはゲート部分に電圧が印可されていない時はドレインとソースに電流が流れないが，電圧が印可されるとソース，ドレイン間に電流が流れるという特性がある(図 4-8 参照)．

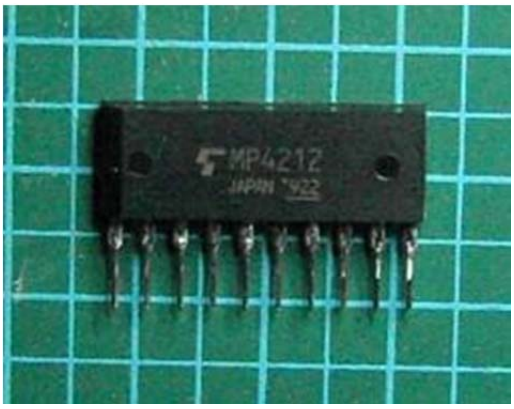


図 4-8　モータドライバ

表 4-2　端子について

モータドライバー	
ピン番号	信号
1	ソース
2	ゲート
3	ドレイン
4	ゲート
5	ドレイン
6	ゲート
7	ドレイン
8	ゲート
9	ドレイン
10	ソース

[6]大きな電力を扱える電界効果トランジスタで「ゲート」「ソース」「ドレイン」の 3 つの電極から構成されている

4.4　製作物の成果　(河合担当)

4.4.1　　モータ制御　(河合担当)

モータ制御では Arduino マイコンと DC モータを繋ぎ，DC モータの正転反転制御を行っている．図 4-9 はそのイメージ図である．

図 4-9　モータ制御のイメージ図

使用した部品は Arduino マイコンと DC モータ，モータドライバである． 図 4-11 はモータドライバの等価回路[7]である．回路との接続方法は Arduino マイコンの 13，12，11，10 の 4 つのデジタル出力端子とモータドライバのゲート端子 2，4，6，8，を繋げ，ドレイン端子の 3，7 と 5，9 をモータに接続する [6].

本回路では Arduino マイコンの電源にはマイクロ USB ケーブルを使用しデスクトップパソコンから 5V の電源を供給した．モータ回路の電源には安定化電源により 5Vの電圧を供給した．図 4-10 は回路図である．

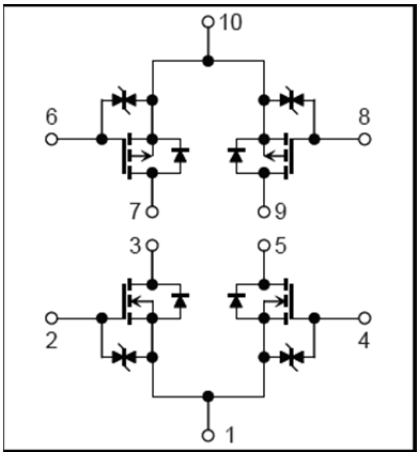


図 4-11　モータドライバの等価回路

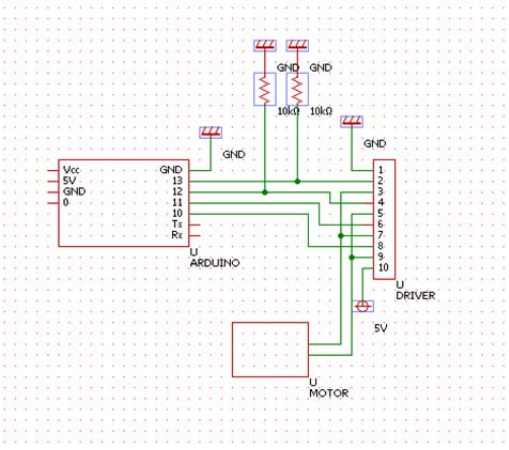


図 4-10　モータ制御の回路図

以下でモータ制御プログラムの解説をする

Arduino のデジタル出力端子の出力レベルを変えることによってモータの正転反転制御を行っている．モータドライバのゲート端子 2，4，6，8，と Arduino の 4 つのデジタル出力端子 13，12，11，10 の 4 つを接続させ，以下で初期設定として 13，12，11，10 を出力に設定する．

```
pinMode(13, OUTPUT);
pinMode(12, OUTPUT);
pinMode(11, OUTPUT);
pinMode(10, OUTPUT);
```

13，11 の出力が「HIGH」，12，10 の出力が「LOW」の時モータは正転回転をする

```
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(11, HIGH);
digitalWrite(10, LOW);
```

13，11 の出力が「LOW」，12，10 の出力が「HIGH」の時モータは反転回転をする

```
digitalWrite(13, LOW);
digitalWrite(12, HIGH);
digitalWrite(11, LOW);
digitalWrite(10, HIGH);
```

13，12 の出力が「LOW」，11，10 の出力が「HIGH」の時モータは停止する

```
digitalWrite(13, LOW);
digitalWrite(12, LOW);
digitalWrite(11, HIGH);
digitalWrite(10, HIGH);
```

図 4-12 が実際に製作した回路の写真である．また，モータ制御のプログラムの全体は付録に記載する

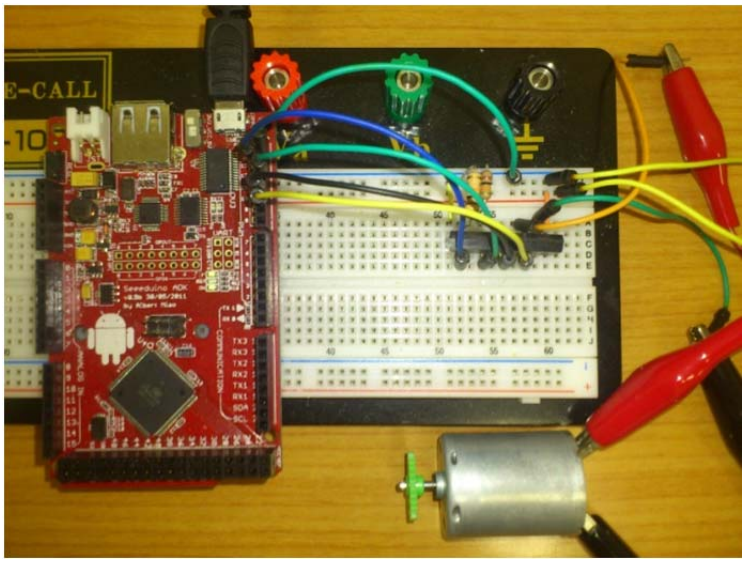


図 4-12　モータ制御回路

4.4.2　　Bluetooth 通信でのモータ制御　(河合担当)

本節では Android 端末から Bluetooth 通信を使ってモータ制御を行う．図 4-13 はイメージ図である．

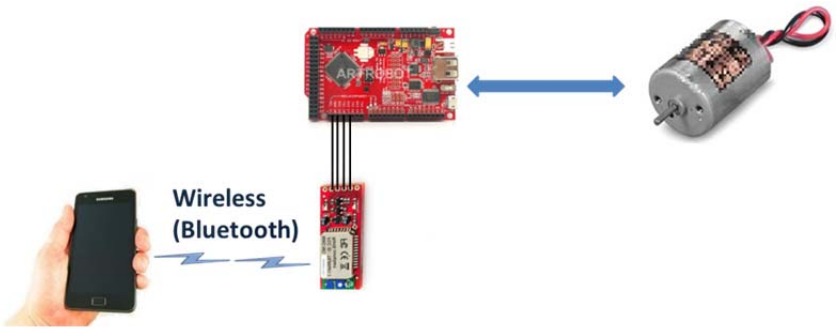


図 4-13　Bluetooth 通信でのモータ制御

使用した部品は Arduino マイコンと DC モータ，モータドライバ，　Bluetooth モジュールである．表 4.3 は Bluetooth mate の端子図である．Bluetooth mate のピン番号 2 の Rx 端子を Arduino マイコンの Tx 端子に，Bluetooth mate のピン番号 3 の Tx 端子を Arduino マイコンの Rx 端子につなげる事により，　Android 端末からの信号を Bluetooth mate で受信し，　Arduino マイコンに信号を伝えることが出来る［7］．その信号に基づき前節の方法でモータの正転反転を行う．Bluetooth mate の動作電圧は 5V である．図 4-14 は回路図である

表 4.3　Bluetooth mate の端子表

ピン番号	信号	意味
1	RTS-0	
2	RX-1	データ受信
3	TX-0	データ送信
4	Vcc	電源
5	CTS-1	
6	GND	グランド

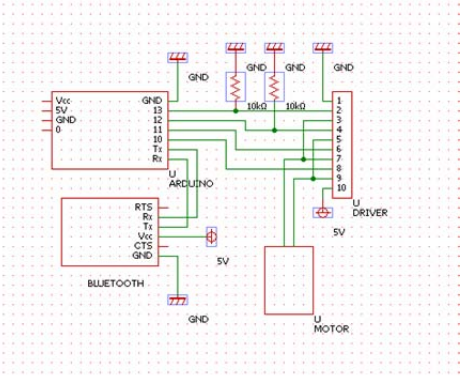


図 4-14　Bluetooth 通信でのモータ制御の回路図

以下で Bluetooth 通信でのモータ制御プログラムを解説する． Arduino 側が Bluetooth 受信を担当しているので，送信側である Android 端末側からのプログラムについて簡単に説明する. 詳しい説明は第 5 章で行う．Android 端末側には mButton, onButton, offButton の 3 つのボタンがあり，それぞれ「STOP」「OPEN」「CLOSE」を指している．それぞれのボタンを押すことによって“0”，“1”，“2”の信号が Arduino 側に送信される．

```
mButton.setOnClickListener(new OnClickListener(){
    public void onClick(View mView) {
        if (mView.equals(mButton)) {
            commandId = "0";

            //～省略～

onButton.setOnClickListener(new OnClickListener(){
    public void onClick(View mView) {
        if (mView.equals(onButton)) {
            commandId = "1";

            //～省略～

offButton.setOnClickListener(new OnClickListener(){
    public void onClick(View mView) {
        if (mView.equals(offButton)) {
            commandId = "2";
```

Android 端末から信号が届いているか Arduino 側の以下の部分で確認する．

```
if (Serial.available() > 0) {      //android端末から信号が来てたら実行
```

信号を受信していれば commandId という変数に信号を格納する［8］．

```
commandId = Serial.read();      //送信された信号をcommandIdに格納
```

commandId に格納された信号によりモータ制御を行う．

```
if(commandId == 48 /*'0'*/){      //android端末から"0"という信号が来たらモーター停止
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, HIGH);
  }else if (commandId == 49 /*'1'*/ ) {     //android端末から"1"という信号が来たらモーター正転
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, LOW);
  } else if(commandId == 50 /*'2'*/ ){      //android端末から"2"という信号が来たらモーター反転
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, HIGH);
  }
```

Android 端末から送信された信号は文字列の ASCII コードであり，それを Arduino マイコンによって命令を読み取っている．また Android 端末から何も信号が送信されていない時，モータは常に停止状態となる．Bluetooth 通信でのモータ制御プログラムは付録に記載する．

図 4-15 は実際の回路の写真である．

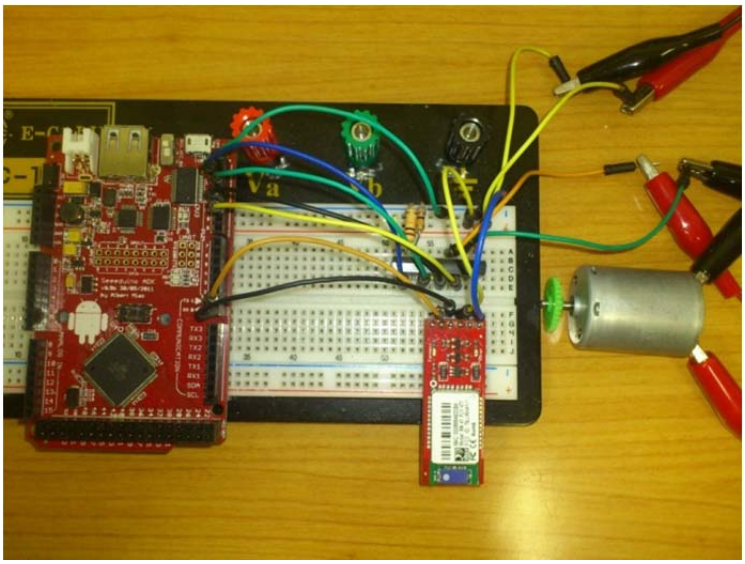


図 4-15　実際の回路

4.4.3　　モータの自動停止制御　(河合担当)

モータの自動停止制御とは Android 端末から「stop」のボタンを押さず電流センサを使って自動的にモータを止める制御である．図 4-16 はイメージ図である

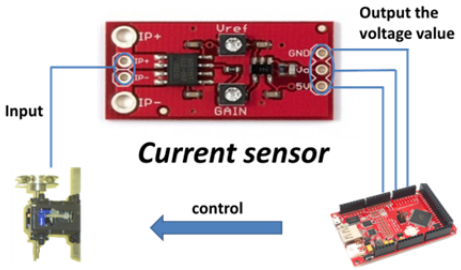


図 4-16　モータ自動停止制御のイメージ図

表 4-4 は電流センサの端子図である．電流センサの「Ip+」,「Ip−」端子にモータを繋げ電流を測定する．測定された電流量によって「Vout」端子から電圧値が出力される［9］．この値を Arduino マイコンに読み取らせることによってモータを自動停止させる．　図 4-17 が回路図である

表 4-4　電流センサの端子図

電流センサ		
ピン番号	信号	意味
1	5V	電源
2	Vout	電圧値出力
3	GND	グランド
4	Ip+	電流入力
5	Ip−	電流入力

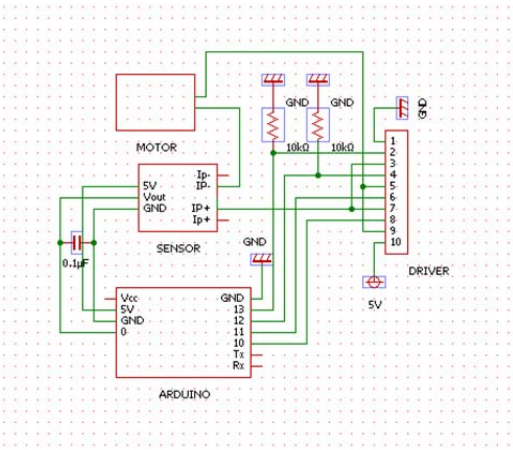


図 4-17　モータ自動亭制御の回路図

以下でモータを自動停止させるプログラムの解説を行う．

電流センサから出力される電圧値がある閾値を超え，それが一定期間継続したときモータを自動的に止めるというものである．モーターの動作開始時は瞬間的に大きな電流が流れることがあるが，そのような場合はモーターを停止しないこととする．

まず電流センサから読み取った電圧値を変数 CurrentValue1 に格納する．

```
CurrentValue1=  analogRead(0);    //通常測定の電流値
```

モーターが正転時（カーテンが開くとき）に CurrentValue1 は大きな値をとり，反転時（カーテンが閉じるとき）に小さな値をとる．そこで，以下のように停止判定を行う．

```
if( CurrentValue1>=650  // when opening
   || CurrentValue1<=310  // when closing
   ){
   exceed_count += 1;
 }
 if(exceed_count>10){
   curtainStop();
 }
```

10 回連続で閾値（650 と 310）を超えた場合，カーテンを停止する処理を行う．

もし上の条件を満たさなかったときは以下の処理に入り， Android 端末から送信された信号によってモータの制御を行う

```
switch(commandId){
 case 48:digitalWrite(LED1, LOW);
         digitalWrite(LED2, LOW);
         digitalWrite(LED3, HIGH);
         digitalWrite(LED4, HIGH);
         break;
 case 49:digitalWrite(LED1, HIGH);
         digitalWrite(LED2, LOW);
         digitalWrite(LED3, HIGH);
         digitalWrite(LED4, LOW);
         break;
 case 50:digitalWrite(LED1, LOW);
         digitalWrite(LED2, HIGH);
         digitalWrite(LED3, LOW);
         digitalWrite(LED4, HIGH);
         break;
```

モータの自動停止制御プログラムは付録に記載する.

図 4-18 は実際の回路の写真である.

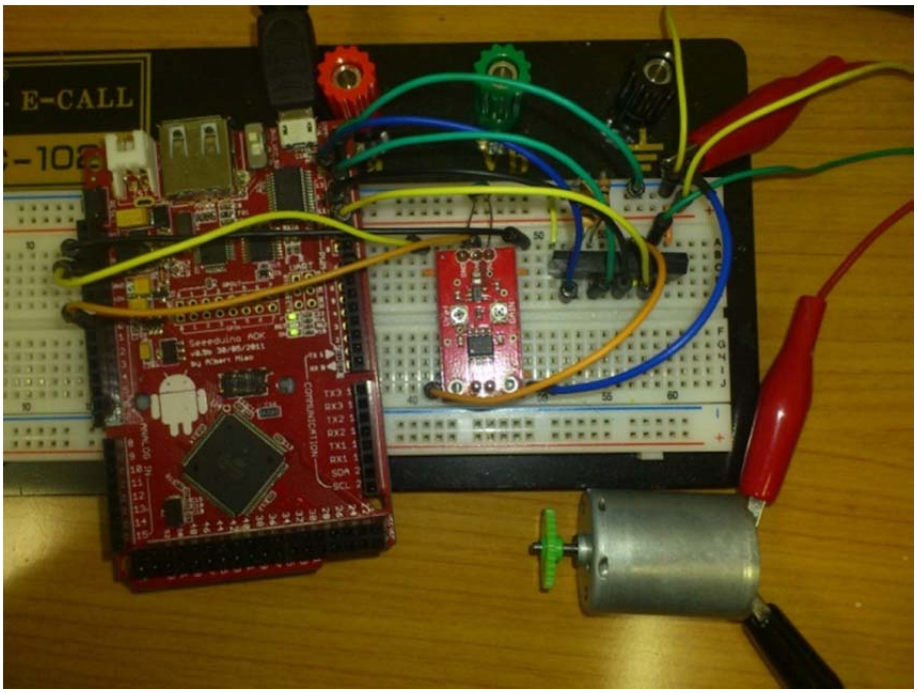


図 4-18 モータ自動停止制御回路

4.4.4　　電流センサからの出力値 (河合担当)

使用している電流センサは入力される電流の値によって出力する電圧が変化するという特徴を持っている．出力される電圧値は入力される電流に対して 185mV/A の感度で出力される．電流センサに電流が流れていない時は 2．5V が出力され，電流量によって電圧値が変わる． Arduino マイコンは 10 ビットの AD 変換機を 16 個持っているので 0～1024 の整数値で出力される．

4.4.5　　完成した回路　(河合担当)

図 4-19 はモータ， Bluetooth モジュール，電流センサのすべてを搭載した回路図であり，図 4-20 は回路の写真である．ここまではブレッドボードを用いて作っていたが，これは機体に搭載するためユニバーサル基板で製作した．それに伴い， Arduino マイコンの電源はパソコンからではなく 5 V の携帯用モバイルバッテリー，回路の電源は 5V の AC アダプタを家庭用コンセントから使用するようにした．モータとの接続もワニロクリップからコネクタに変更し機体にコンパクトに搭載できるよう作った．図 4-21 は Arduino と実装した回路である．

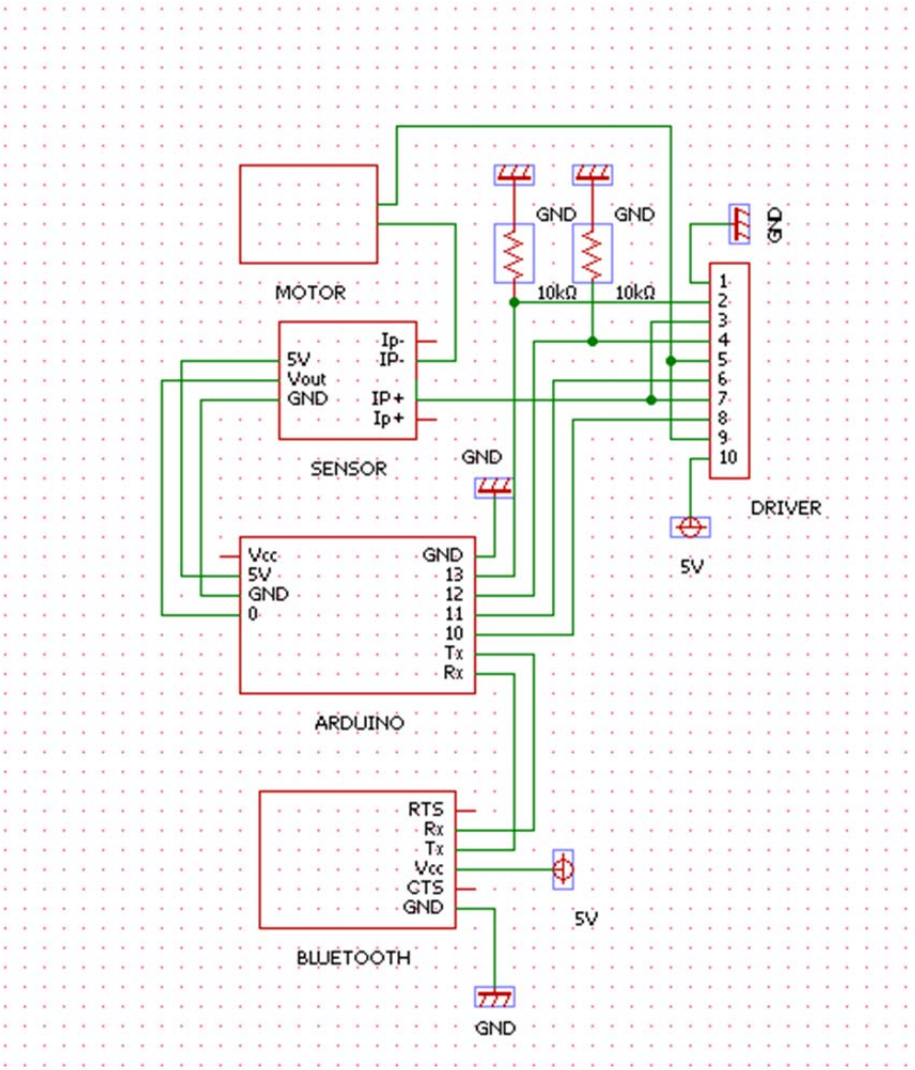


図 4-19　全体の回路図

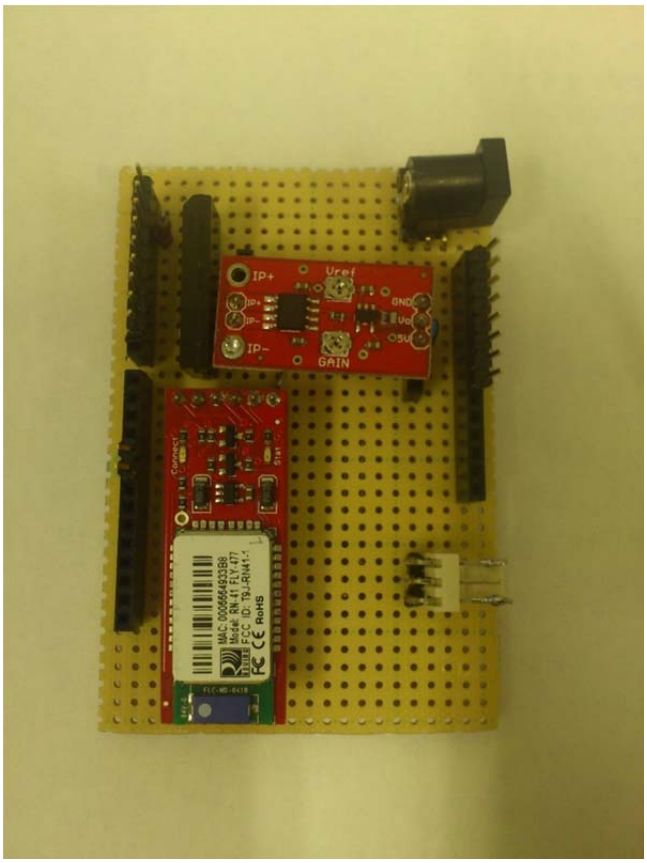


図 4-20　製作した回路

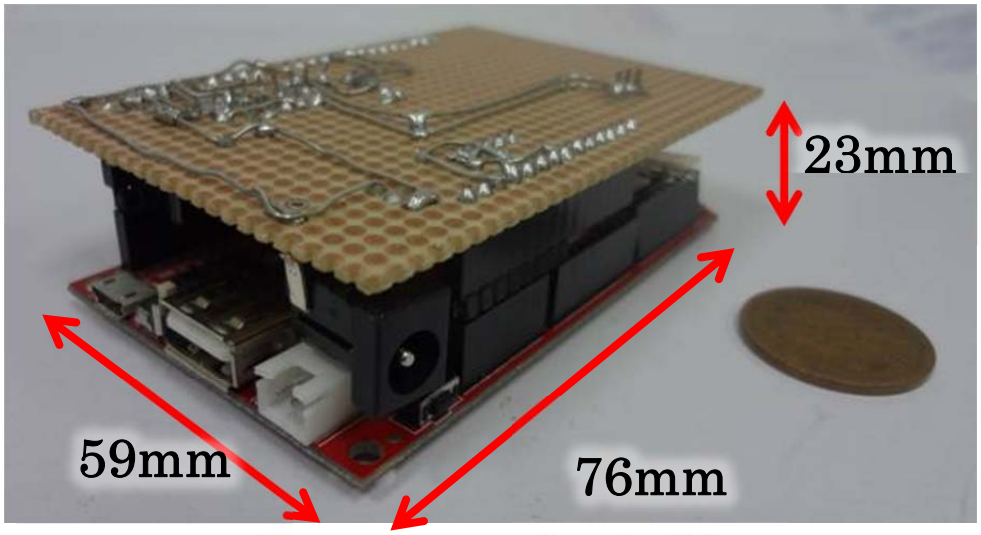


図 4-21　Arduino マイコンとの実装

4.4.6　　機体の搭載について　(河合担当)

図 4-21 で完成した回路を機体の動力部に搭載する．まずプラスティックの容器に回路を入れる(図 4-22 参照)．回路を入れるケースはプラスティックを用い，スポンジを用いることで回路にかかる振動を抑える．これに蓋を閉めて第 3 章で作ったモータの動力部に回路を搭載したのが図 4-23 である

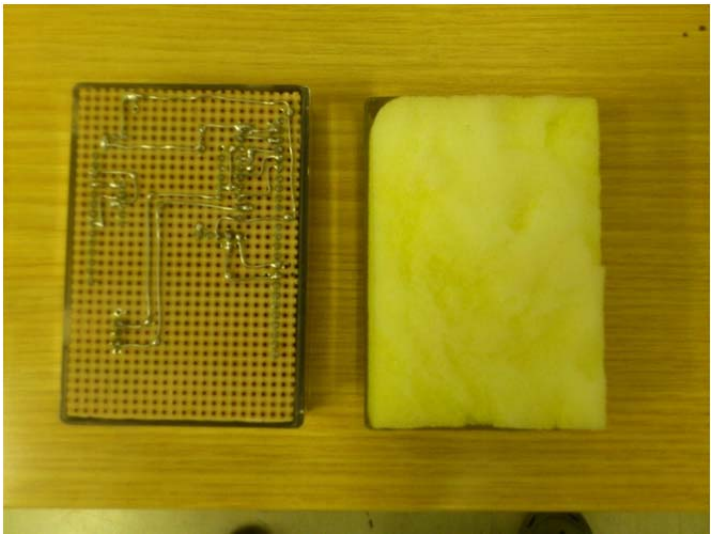


図 4-22　回路容器

.

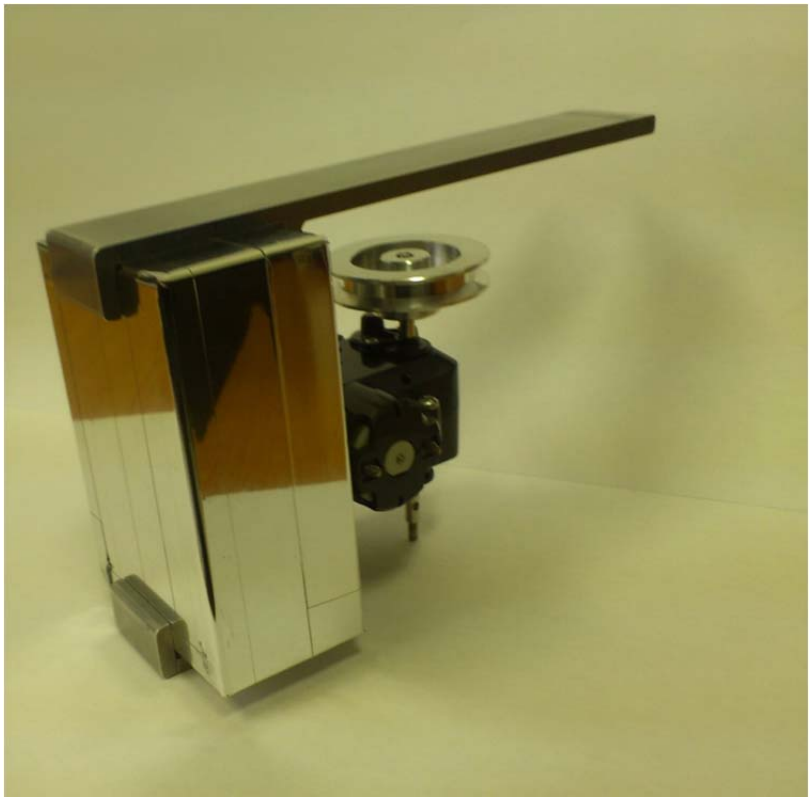


図 4-23　機体への搭載

第5章　Android application 製作（菖蒲澤担当）

5.1　アプリケーション作成 （菖蒲澤担当）

本研究のテーマは, Android アプリケーションによる自動カーテン開閉システムの開発であるが，本章では Android 班が担当した Android アプリケーションの部分について説明する.

5.1.1　機能案 （菖蒲澤担当）

カーテン開閉のアプリケーションに実装する機能は以下の通りである.

a. Bluetooth を利用した開閉操作…家電のリモコンのように無線でカーテンを開閉停止できる機能.

b. タイマーによるカーテンの自動開閉…アラームのように，あらかじめ設定した時間にカーテンを開閉する機能.

c. 音声認識によるカーテンの自動開閉…音声認識によりカーテンを開閉する機能.

d. 外出先などからの遠隔操作によるカーテンの自動開閉…外からインターネット経由でカーテンを開閉する機能.

5.2　機能について　（菖蒲澤担当）

この章では Android 班の作成したアプリケーションの機能について解説していく．リモコン機能，タイマー機能，音声認識機能，遠隔操作機能の 4 つを実装した．機能ごとに様々な特徴があり，カーテンの開閉がより便利かつ簡単に行えるようになっている．まず各機能を呼び出すために，HOME 画面を作成した．

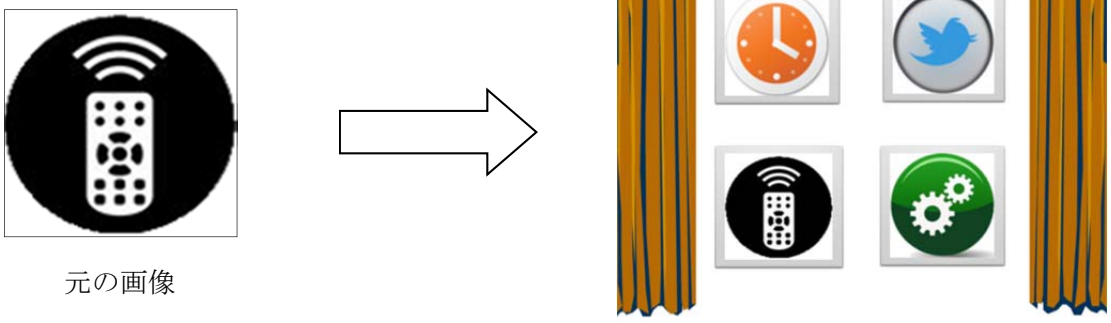
5.2.1　　HOME 画面　（菖蒲澤担当）

HOME 画面とは，アプリケーションを起動すると最初に呼び出される画面である(図 5-1)．画面の上部 4 つのボタンを押すことで各機能を呼び出すことができる．右下のボタンは「設定機能」であり，当初は細かいオプションの設定を行うことも考えたが，本アプリケーションには詳細な設定が必要ないので，機能の実装は見送った．

図 5-1　HOME 画面

5.2.2　　HOME 画面のデザイン　(菖蒲澤担当)

HOME 画面には，イメージボタンを使用している [12]．イメージボタンとは，あらかじめリソースフォルダに用意してある PNG 形式[8]の画像ファイルを利用してボタンとするものである(図 5-2)．イメージボタンを使用することで，文字だけではなく絵で機能を判断することでき，より直感的に使用したい機能を選べるのではないかと考え，使用している．



元の画像　　ボタン上に配置された画像

図 5-2　イメージボタン

[8] 画像の圧縮方法の一種．ファイルサイズは大きいが，画質が良い．

5.2.3　　画面の遷移　(菖蒲澤担当)

HOME 画面からイメージボタンを押すことで，各機能の画面に遷移する [13] [14]．画面が遷移したということは，プログラム上は activity が呼び出されたことを意味し，その activity が持つ別の機能を使用できるようになる．画面遷移は，後述する intent により実現する．

5.2.4　　Activity(アクティビティ)　(菖蒲澤担当)

Activity とは，Android アプリケーションの機能と画面を構成するクラスである．機能ごとにソースコードを作成し，AndroidManifest．xml に宣言することでプログラミングした機能が使えるようになる．本アプリケーションでは機能ごとに複数の Activity が存在している．

5.2.5　　AndroidManifest．xml　(菖蒲澤担当)

Android アプリケーションにはすべて，AndroidManifest．xml というファイルが存在する．新しいアクティビティを作成し，機能を使用したい場合，AndroidManifest．xml に宣言をしなければならず，機能によっては permission を追記する必要もある．

permission とは英語で許可や許諾といった意味で，許可を必要とする権限をあらかじめ AndroidManifest．xml に明示しておかなければない．本アプリケーションでも機能ごとにパーミッションの設定をしたものがあり，後述の機能別に記述していく．

5.3　リモコン機能　（菖蒲澤担当）

本アプリケーションの最も基本的で重要な機能であるリモコン機能とは，ボタンを押すことでカーテンの開閉停止を無線で行うことができる機能である．この機能をもとに他の機能を作成した．図 5-3 の赤い丸で囲った部分のボタンを押すことで，リモコン機能を呼び出すことができる．

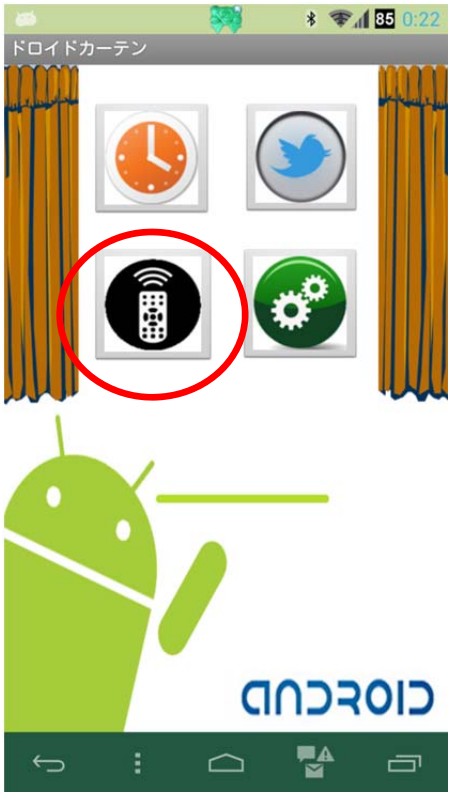


図 5-3　リモコン機能の呼び出し

5.3.1　　リモコン機能のデザイン　(菖蒲澤担当)

リモコン機能の画面には図 5-4 のようにボタンが 4 つあるが, 上部 3 つのボタン「OPEN」「STOP」「CLOSE」を使用する.「OPEN」または「CLOSE」のボタンは, 一度押すと半透明になって押せなくなるようにしており,「STOP」か反対の動きをするボタンしか押すことができず, それらを押すことによりもう一度すべてのボタンが押せるようになる [15].

これは, ユーザーに押したボタンを認識させるとともに, カーテンの動きをアプリケーションの画面上で判断できるようにするためである. ただし, カーテンが自動停止した場合は, 回路から状態を知らせる機能を実装していないため, ボタンは半透明のままである.

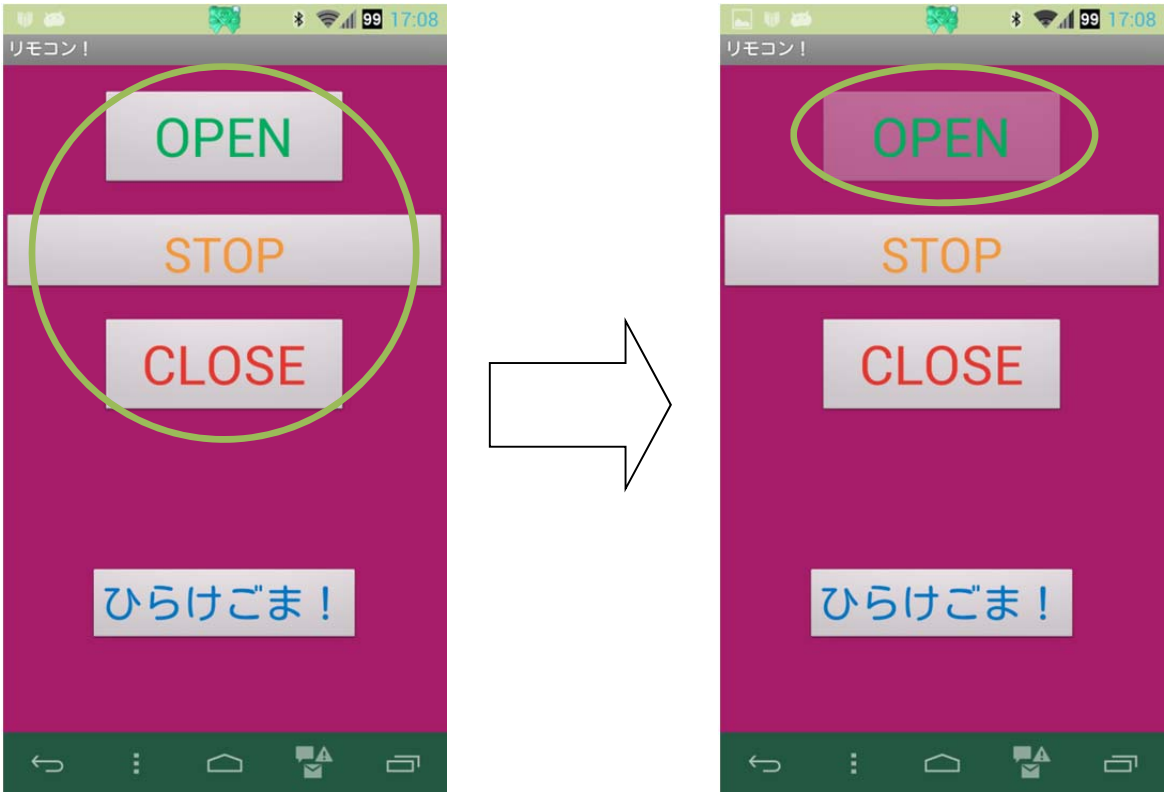


図 5-4　リモコン機能の画面

5.3.2　　リモコン機能の使い方　(菖蒲澤担当)

すべてボタンを押すことで機能が使用できる(図 5-5).

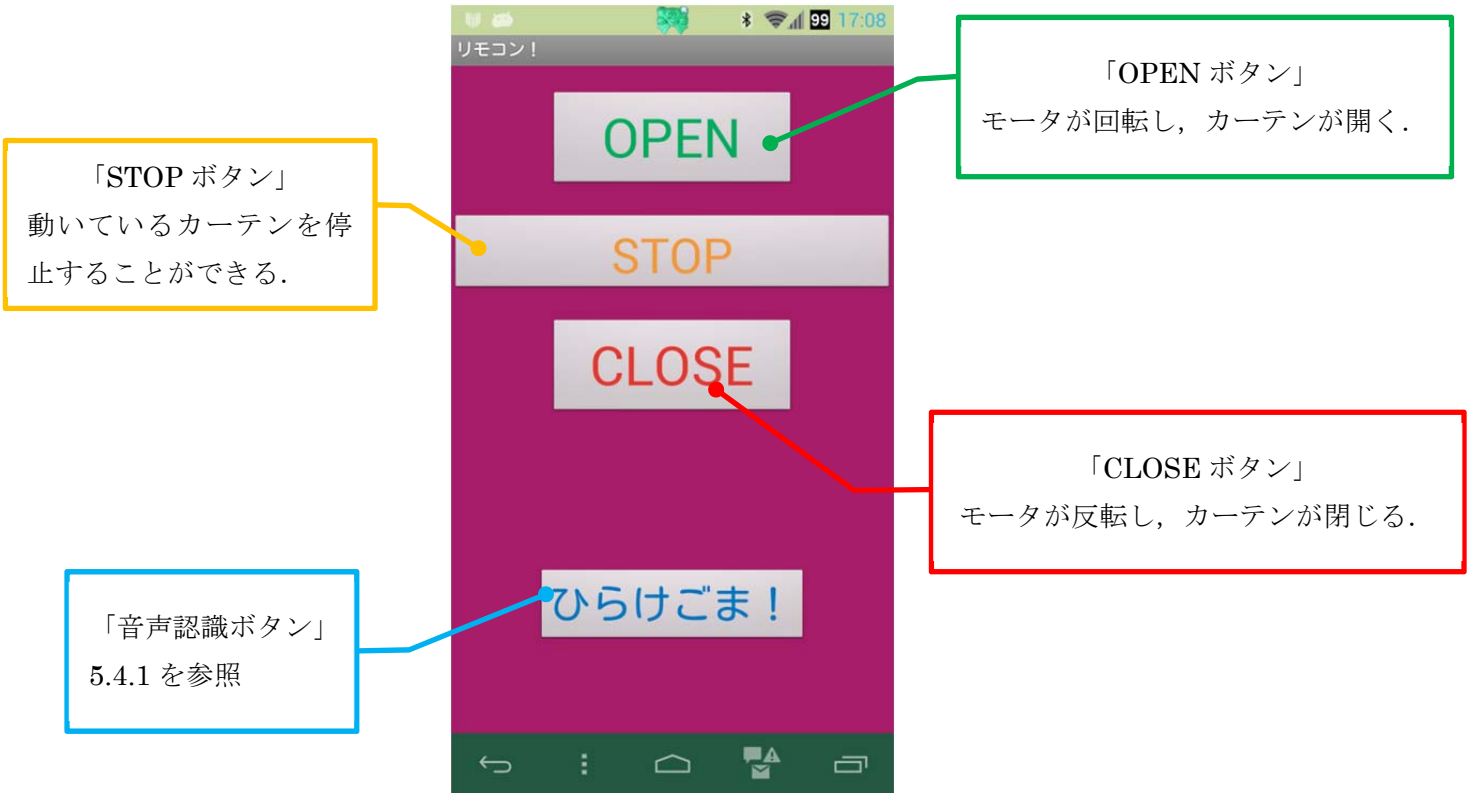


図 5-5　リモコン機能の使い方

5.3.3　　Bluetooth　（菖蒲澤担当）

リモコン機能を含み，作成したアプリケーションの中でカーテンを操作する機能にはすべて，Bluetooth という無線技術を利用している [16] [17].

Bluetooth とは Android とハードウェアを接続するための一つの方法である．本研究のカーテン自動開閉システムでは, 開閉システムの回路側に Bluetooth モジュールを使用し，Android アプリケーションとの無線接続を可能にしている．2013 年 2 月現在のスマートフォン事情では，ほぼ Bluetooth の機能が搭載されているので，ほとんどのスマートフォンで本研究のアプリケーションの機能を使用することができる．

5.3.4　　アプリケーションからのモータ制御　(菖蒲澤担当)

Android アプリケーションで Bluetooth を使用可能にするには, 図 5-6 のように permission を AndroidManifest．xml に追加する必要がある．

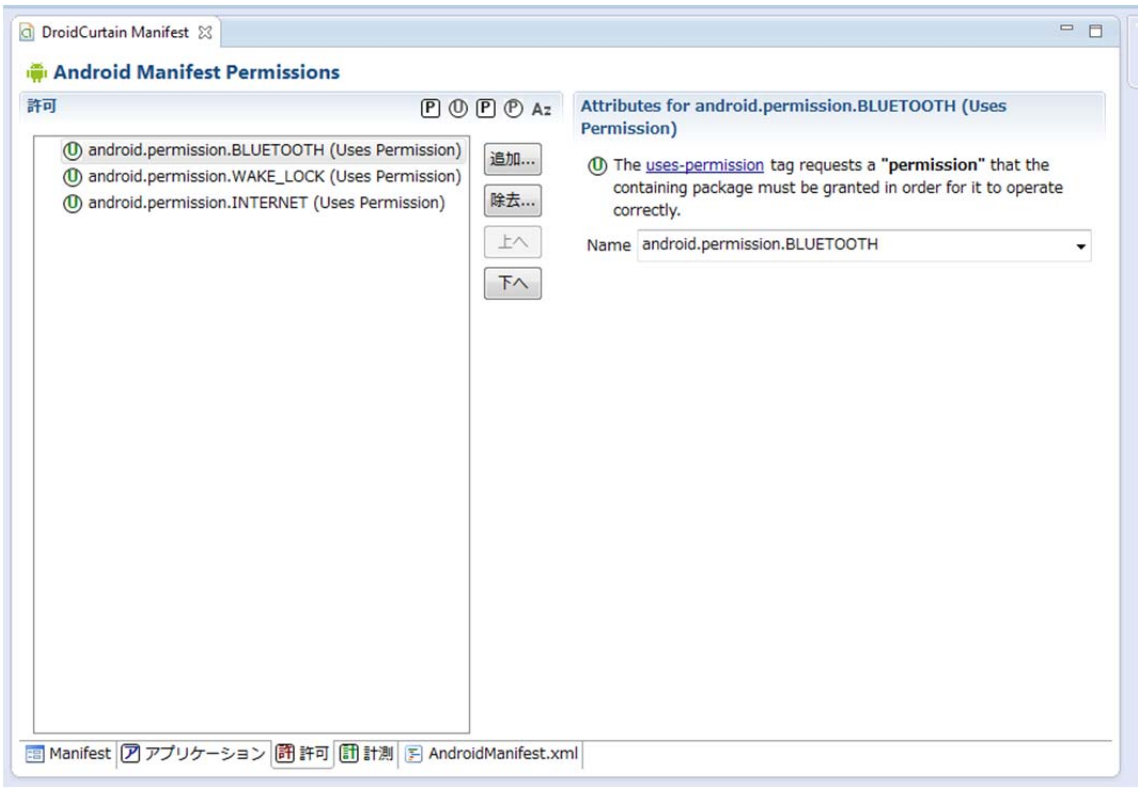


図 5-6　permission の宣言

用意した3つのボタンにそれぞれ開閉停止の機能を割り当てる.図 5-7 のプログラムで, mButton, onButton, offButton がそれぞれ, アプリケーション上のボタン「STOP」「OPEN」「CLOSE」を指している.「comandId」に値には, mButton がモータ停止の“0”, onButton がモータ回転の“1”, offButton がモータ反転の“2”と定義されており, ボタンを押すことで図 5-8 のプログラムによって Bluetooth を使って回路側のモジュールに命令を送信できる.

```
mButton. setOnClickListener(new OnClickListener(){
        public void onClick(View mView) {
                if (mView. equals(mButton)) {
                        commandId = "0";

        ～省略～

onButton. setOnClickListener(new OnClickListener(){
        public void onClick(View mView) {
                if (mView. equals(onButton)) {
                        commandId = "1";

        ～省略～

offButton. setOnClickListener(new OnClickListener(){
        public void onClick(View mView) {
                if (mView. equals(offButton)) {
                        commandId = "2";
        ～省略～
```

図 5-7　モータ制御のプログラム

```
        ～省略～

  public void run() {
        InputStream mmInStream = null;
        OutputStream mmOutputStream = null;
        try {
// 取得したデバイス名を使ってBluetoothでSocket接続
                mSocket = mDevice. createRfcommSocketToServiceRecord(MY_UUID);
                mSocket. connect();
                mmInStream = mSocket. getInputStream();
                mmOutputStream = mSocket. getOutputStream();
// InputStreamのバッファを格納
                byte[] buffer = new byte[1024];
// 取得したバッファのサイズを格納
                int bytes;
                while(isRunning){
                Log. d(TAG, "sending data");
                mmOutputStream. write(commandId. getBytes());

        ～省略～
```

図 5-8　Bluetooth による信号送信

5.4　音声認識機能　（菖蒲澤担当）

音声認識機能とは，人間などの音声を使ってカーテンの開閉停止をすることができる機能である．音声認識は Android1．5 から搭載されている機能である．この機能は「Google 音声検索」というアプリケーションを利用しているため，あらかじめ端末にインストールされている必要がある．

5.4.1　　音声認識機能の使い方　（菖蒲澤担当）

この機能は，リモコン機能と同じ画面を用いる．図 5-9 の画面下部の「ひらけごま！」というボタンを押すことで音声認識の機能を使用することができる．ボタンを押すと画面が変わり，そこにしゃべりかけることで，カーテンの開閉停止を行うことができる．

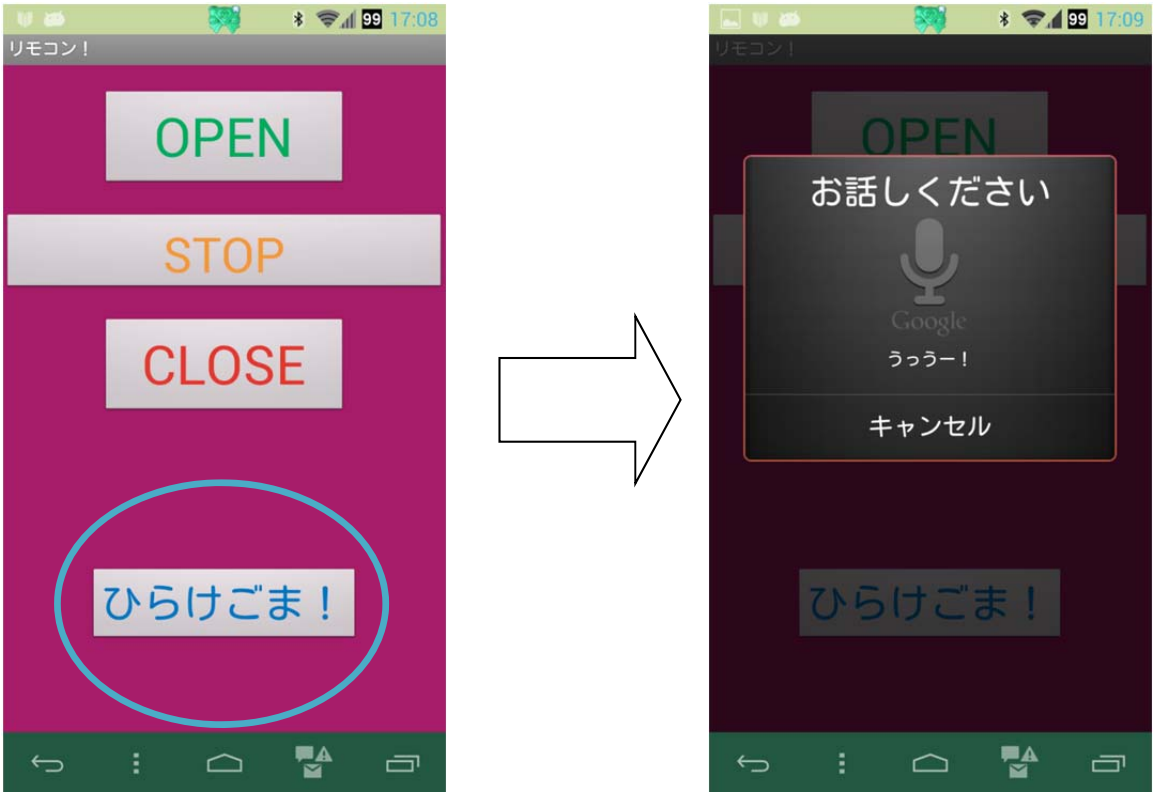


図 5-9　音声認識機能の画面

5.4.2　　音声検索　（菖蒲澤担当）

本研究のアプリケーション内音声認識機能には「Google 音声検索」という Google 社が開発したアプリケーションを利用している．

「Google 音声検索」は，新規でスマートフォンを購入するとあらかじめインストールされていることがほとんどであるため，インストールの方法などは割愛する．

入力した音声の認識は端末側ではなく，インターネットを使用し，サーバー側で認識をし，その結果を端末が受け取り判断するという仕組みである(図 5-10)．そのため，本アプリケーションの音声認識機能は端末が 3G や 4G などの携帯電話通信，Wi-Fi などのインターネット通信を利用可能な環境でないと使用することができない．

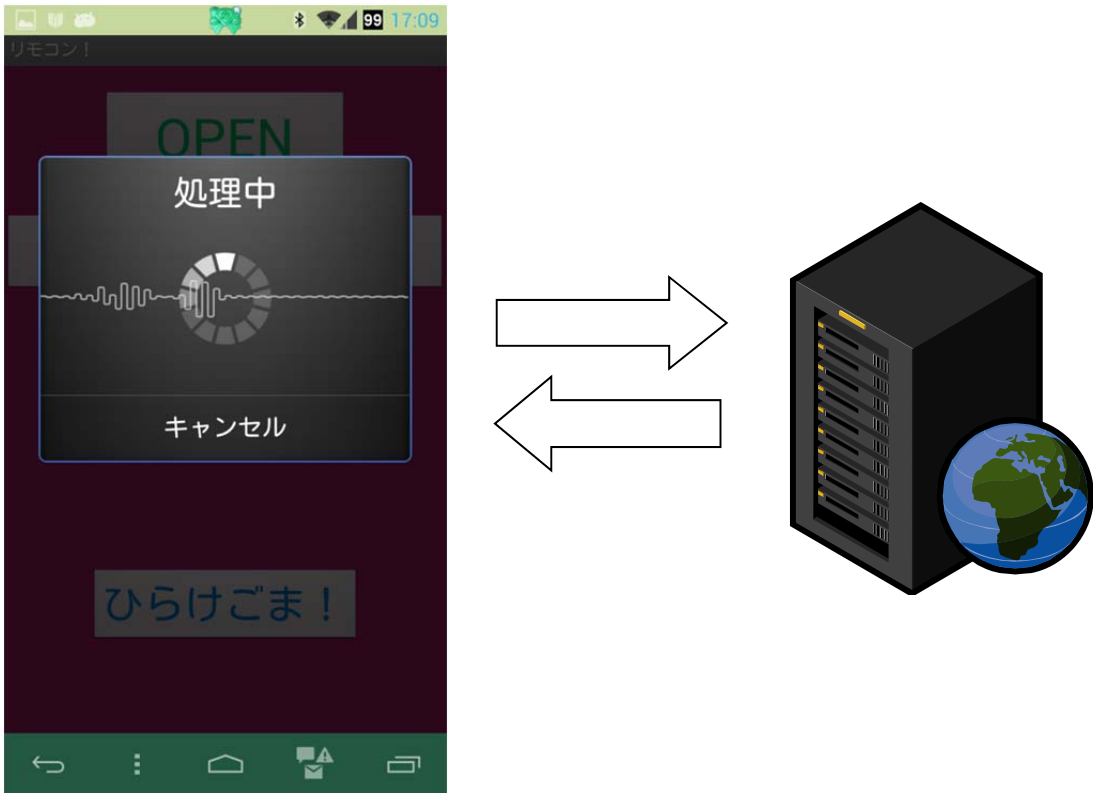


図 5-10　音声検索のイメージ

5.4.3　　結果の受け取り処理　(菖蒲澤担当)

本機能では，intent による「Google 音声検索」の呼び出し，サーバーから返ってきた結果の受け取り後の処理をプログラムした [18] [19].

まず「ひらけごま！」のボタンを押し，「Google 音声検索」を呼び出す(図 5-11)．その後端末に話しかけると，インターネットを経由して結果が返ってくる(図 5-12)．その返ってきた結果が，あらかじめ指定しておいたカーテン開閉停止に関係のある「ひらけ」「とじろ」「とまれ」などの文字列であった場合に，Bluetooth で回路に信号を送り，カーテンが開閉停止する仕組みである.

```
//音声検索
  vButton. setOnClickListener(new View. OnClickListener(){
        public void onClick(View v) {
                try {
                        Intent intent = new Intent(
                        RecognizerIntent. ACTION_RECOGNIZE_SPEECH);

                        intent. putExtra(
                        RecognizerIntent. EXTRA_LANGUAGE_MODEL,
                        RecognizerIntent. LANGUAGE_MODEL_FREE_FORM);

                        intent. putExtra(
                        RecognizerIntent. EXTRA_PROMPT, "うっうー！");

                ～省略～
```

図 5-11 音声検索を呼び出すプログラム

```
        ～省略～

protected void onActivityResult(int requestCode,  int resultCode,  Intent data)
{
// 自分が投げたインテントであれば応答する
        if (requestCode == REQUEST_CODE && resultCode == RESULT_OK) {
                String resultsString = "";

// 結果文字列リスト
                ArrayList<String> results = data. getStringArrayListExtra(
                RecognizerIntent. EXTRA_RESULTS);

                for (int i = 0; i< results. size(); i++) {
// ここでは，文字列が複数あった場合に結合しています
                resultsString += results. get(i);

        ～省略～
```

図 5-12 返ってきた結果を受け取るプログラム

5.4.4　　文字列の指定　(菖蒲澤担当)

カーテンの動作は，インターネットを経由し返ってきた結果をあらかじめ指定しておいた文字列(図 5-13)との比較・判断(図 5-14)で行われる．本機能はカーテンの開閉と停止が目的であるため，その動作に関係のある命令形の語句を複数指定した．

```
//文字列の指定
        ArrayList<String> Open = new ArrayList<String>();
        ArrayList<String> Close = new ArrayList<String>();
        ArrayList<String> Stop = new ArrayList<String>();

        Open. add("オープン");
        Open. add("ひらけ");
        Open. add("開け");
        Close. add("クローズ");
        Close. add("とじろ");
        Close. add("閉じろ");
        Stop. add("ストップ");
        Stop. add("とまれ");
        Stop. add("止まれ");
```

図 5-13 文字列を指定するプログラム

```
//返ってきた結果と文字列の比較・処理
        if(isOpen(resultsString)){
                commandId = "1";
                Log. d(TAG, "onButton pressed");
                Toast. makeText(this,  "開きます",  Toast. LENGTH_LONG). show();
                activateRecognize();

        }else if(isClose(resultsString)){
                commandId = "2";
                Log. d(TAG, "offButton pressed");
                Toast. makeText(this,  "閉まります",  Toast. LENGTH_LONG). show();
                activateRecognize();

        }else if(isStop(resultsString)){
                commandId = "0";
                Log. d(TAG, "mButton pressed");
                Toast. makeText(this,  "停止します",  Toast. LENGTH_LONG). show();
                activateRecognize();

        }else{
                Log. d(TAG, "once again");
                Toast. makeText(this,  "もう一度",  Toast. LENGTH_LONG). show();
                activateRecognize();
        }

        ～省略～
```

図 5-14 文字列との比較・処理のプログラム

5.4.5　　Intent(インテント)　(菖蒲澤担当)

Intent とは画面遷移やデータの呼び出し，受け渡しを行うために使う仕組みである．

本アプリケーションでは，明示的インテントという方法で，あらかじめ作成しておいた別機能のアクティビティの名前を直接指定し，呼び出すという方法をとっている．ここで呼び出されるアクティビティは，AndroidManifest．xml に登録してなければ呼び出すことができず，アプリケーションを使用することができなくなってしまうので注意が必要である．

5.5　タイマー機能　（菖蒲澤担当）

タイマー機能とは，任意で設定した時間にカーテンを自動開閉するための機能である．

赤い丸で囲った部分のボタンを押すことで，この機能を呼び出せる(図 5-15)．例えば，朝起きるときにカーテンを開くよう設定すれば起床時に日の光が差し込み目を覚ますという，音や振動ではない新たな目覚まし機能として使用できる．

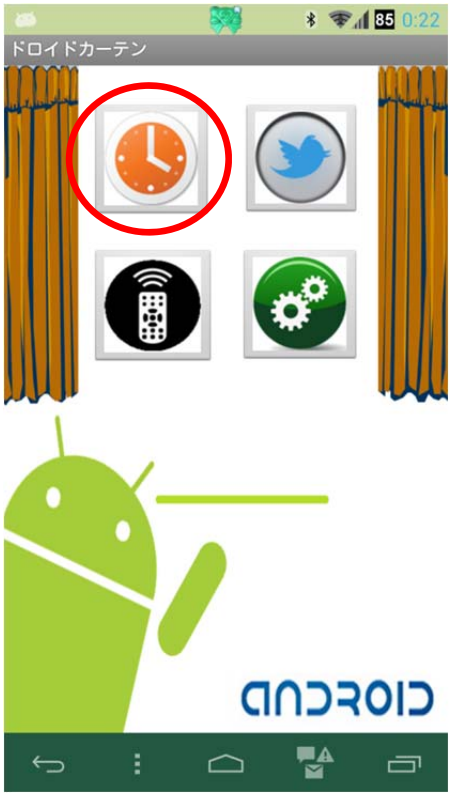


図 5-15 タイマー機能の呼び出し

5.5.1　　タイマー機能のデザイン　(菖蒲澤担当)

タイマー機能の画面は，画面上部の TimePicker と呼ばれる時間を選択するための部分と，中央部の 3 つのボタン「Open」「Close」「Cancel」から成り立っている(図 5-16).「Open」または「Close」のボタンは，一度押すと半透明になり押せなくなるようにしており，「Cancel」を押すことで，再び押せるようになる．そうすることで，現在タイマーが ON なのか OFF なのか判断できると考えている．

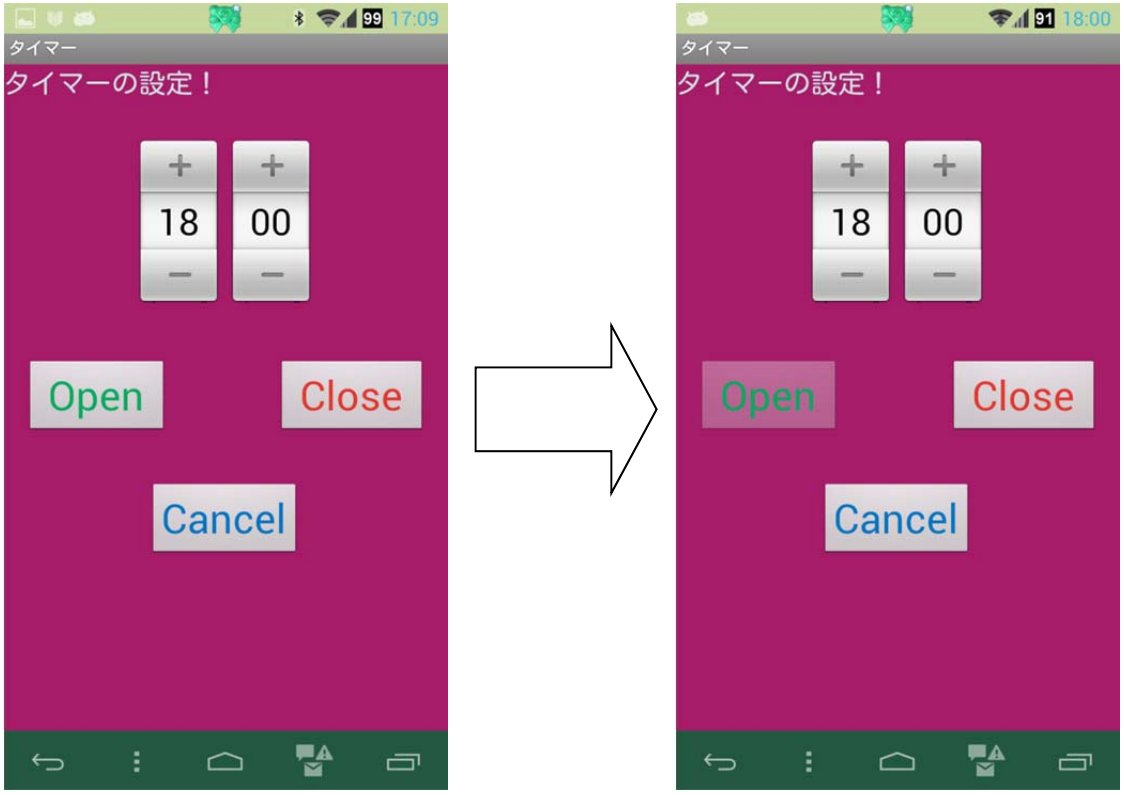


図 5-16　タイマー機能の画面

5.5.2　　タイマー機能の使い方　（菖蒲澤担当）

すべてボタンを押すことで機能が使用できる(図 5-17).

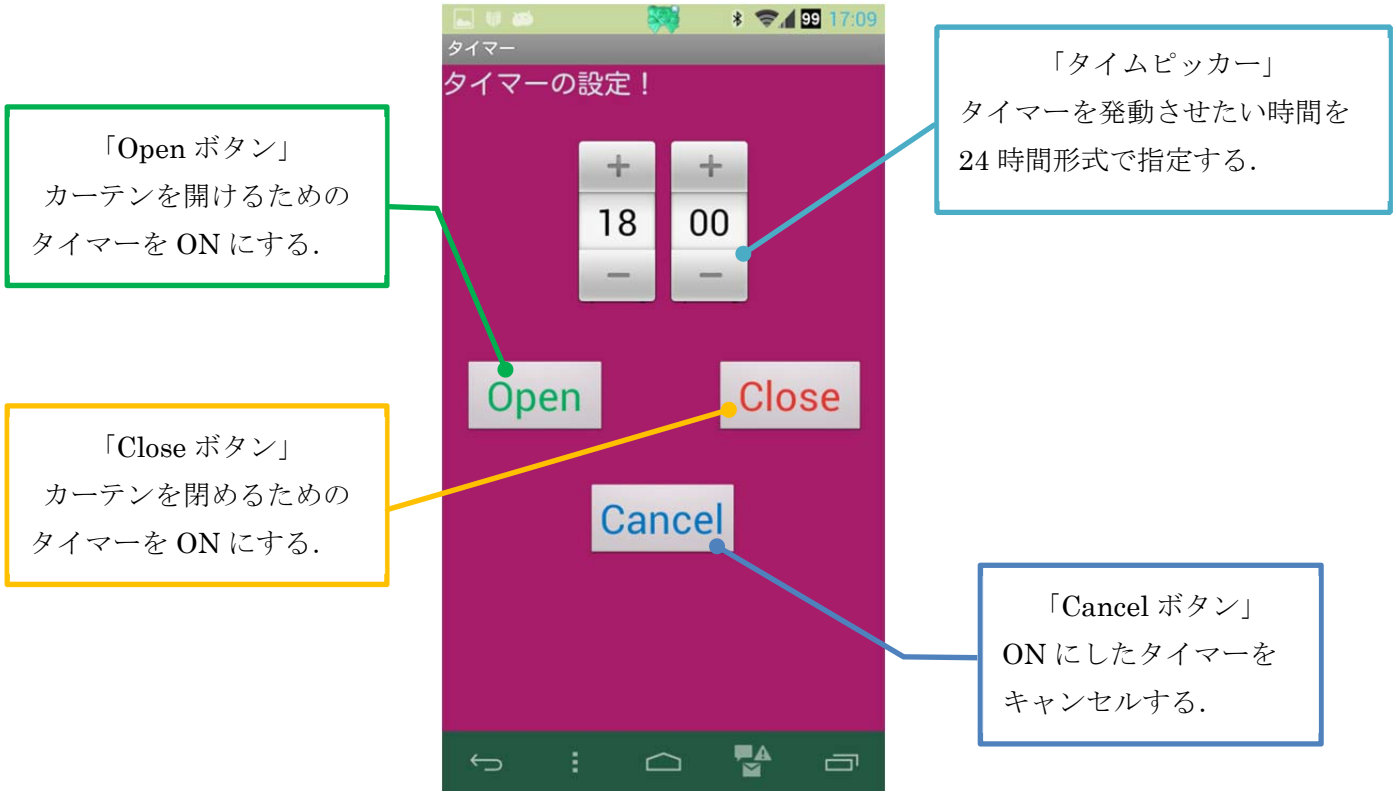


図 5-17 タイマー機能の使い方

5.5.3　　時間の取得　(菖蒲澤担当)

図 5-18 はアプリケーション内で現在時間を取得するためのプログラムである [20]．本機能は任意の時刻を設定してカーテンを開閉する機能なので，以下のプログラムのように，まずスマートフォンで設定されている時刻を取得する．その後，取得した時刻をアプリケーション画面上の TimePicker に表示させる．

```
// 現在時刻の取得
        calendar = Calendar.getInstance();
        calendar.setTimeInMillis(System.currentTimeMillis());
        Date date = calendar.getTime();
        cal_hour = date.getHours();
        cal_minute = date.getMinutes();

// TimePicker に反映
        timePicker = (TimePicker)findViewById(R.id.timePicker1);
        timePicker.setIs24HourView(true);
        timePicker.setCurrentHour(cal_hour);
        timePicker.setCurrentMinute(cal_minute);
```

図 5-18 時間の取得と TimePicker へ反映させるプログラム

5.5.4　　時間経過後の処理　(菖蒲澤担当)

取得した時刻から，任意設定した時刻になると設定していた開閉のタイマーが発動し，用意しておいた別の Activity が呼び出され，カーテンが開閉する（図 5-19）．この際，5.4.5 の画面遷移での記述でもあった，Intent という方法で機能を呼び出し開閉を行っている．

```
public void startAlarm() {
        alarmManager = (AlarmManager)getSystemService(Context. ALARM_SERVICE);
// TimePicker で選択された時刻を取得
        timePicker = (TimePicker)findViewById(R. id. timePicker1);
        cal_hour = timePicker. getCurrentHour();
        cal_minute = timePicker. getCurrentMinute();

        ～省略～

private PendingIntent getPendingIntent() {

        if(Open && !Close){
                Open1 = new Intent( getApplicationContext(), Timer_Open. class );
                Open2 = PendingIntent. getActivity(this, 0, Open1, PendingIntent.
                FLAG_UPDATE_CURRENT );
                result = Open2;
                Open = false;
        }

        if(Close && !Open){
                Close1 = new Intent( getApplicationContext(), Timer_Close.
                class );
                Close2 = PendingIntent. getActivity(this, 0, Close1,
                PendingIntent. FLAG_UPDATE_CURRENT );
                result = Close2;
                Close = false;
        }
        return result;
```

図 5-19 タイマー発動時のプログラム

5.6　遠隔操作機能　（菖蒲澤担当）

遠隔操作機能は，外出先などからインターネットを経由してカーテンを開閉することが目的の機能である．　赤い丸で囲った部分のボタンを押すことで機能を呼び出すことができる(図 5-20).

本機能は，「Android×Arduino でつくるクラウド連携デバイス　-Android ADK で電子工作をはじめよう！」を文献として参考にし，作成した [21] .

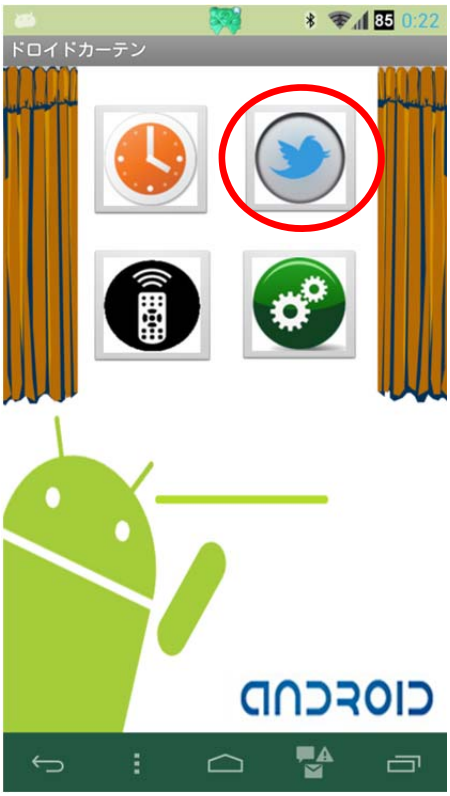


図 5-20 遠隔操作機能の呼び出し

5.6.1　　遠隔操作機能のデザイン　(菖蒲澤担当)

この機能の画面は，画面最上部に editText と呼ばれるキーワードを入力する部分，その下のボタン 2 つで遠隔操作によるカーテン開閉の待ち受け状態の ON/OFF を切り替えることができる(図 5-21)．ボタンはトグルボタンを利用し，待ち受け状態が ON なのか OFF なのかをわかりやすくした．

図 5-21　遠隔操作機能の画面

5.6.2　　遠隔操作機能の使い方　(菖蒲澤担当)

「キーワード」は，入力欄をタッチすることで入力できるようになる.「開ける」「閉める」は，ボタンを押すことで機能を使用できる(図 5-22)．本機能の遠隔操作には，後述する Twitter というインターネットを利用した情報サービスを利用する．そのため本機能を利用する場合，あらかじめ Twitter のアカウントを作成する必要がある.

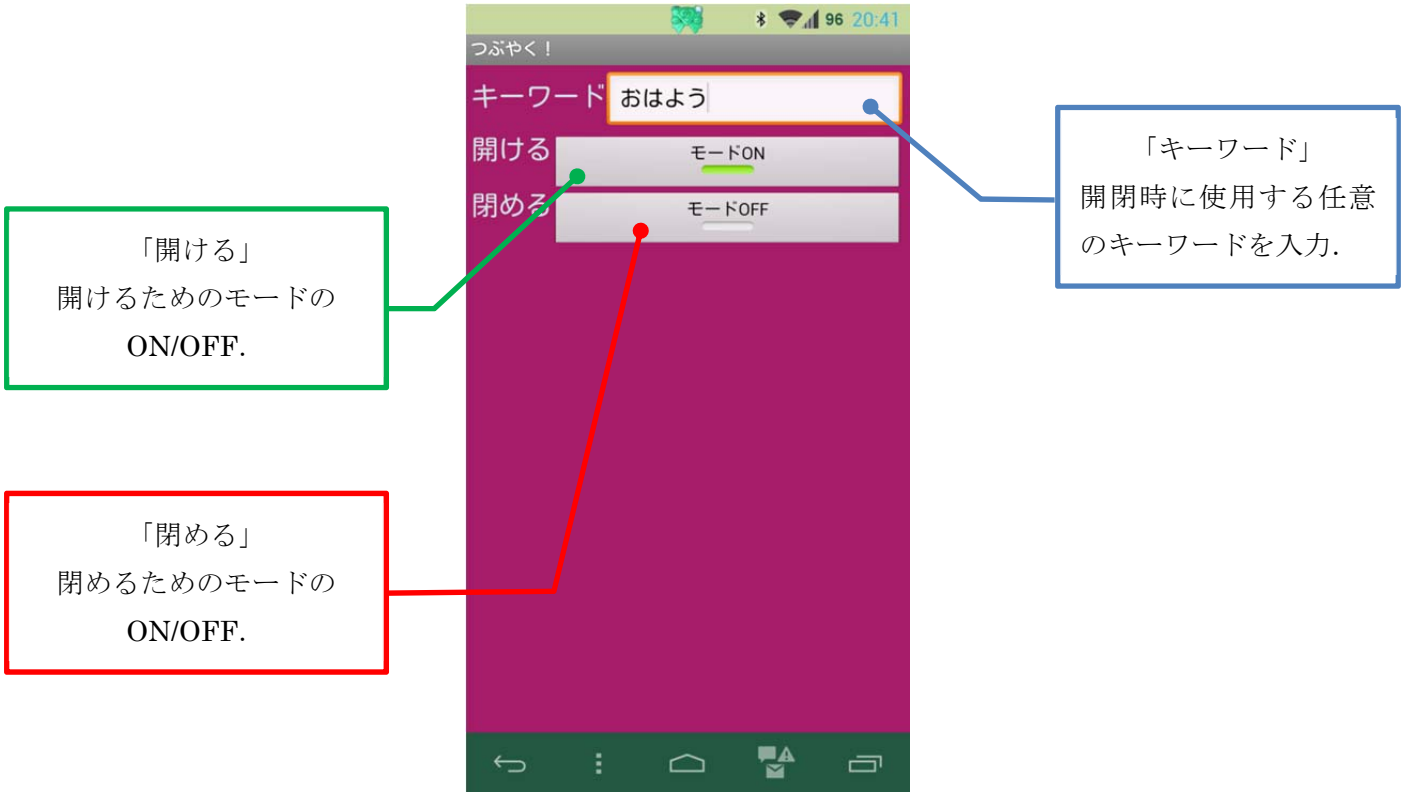


図 5-22 遠隔操作機能の使い方

5.6.3　　Twitter(ツイッター)　(菖蒲澤担当)

Twitter [22]とは，2006 年にアメリカのオブビアウス社(現 Twitter 社)が開始した，パソコンや携帯電話，スマートフォンから登録した自身のアカウントにログインし，「つぶやき」といわれる 140 文字以内の内容を投稿することができるというサービスである(図 5-23).

サービス利用にはインターネット通信が可能な環境が必要であり，投稿した内容は世界中の人々がみることができるシステムになっている．ログインした際，自分の投稿や，フォローしたユーザーの投稿がタイムラインとして表示される．気になる人や企業のアカウントをフォローすることで，様々な情報を得ることができる．

図 5-23　Twitter のロゴとタイムライン

5.6.4　　フォローと DM(ダイレクトメッセージ)　(菖蒲澤担当)

ツイッターでは，アカウントを持っている他のユーザーをフォローすることで，そのユーザーの投稿を自分のタイムラインに表示することができる．また，投稿の他にダイレクトメッセージ(以下 DM)という機能があり，電子メールのように使うことができる．本機能ではこの DM を利用して，カーテンの開閉を行う．DM を利用することより，第三者から投稿やメッセージのやり取りを見られる心配がないので，カーテンを閉める機能を防犯に役立てることもできる(図 5-24).

図 5-24　フォローと DM

5.6.5　　Twitter API　（菖蒲澤担当）

Twitter を自作のアプリケーションに利用する場合, Twitter API を利用するため Twitter の開発者として登録し，アプリケーションごとに ID を取得する必要がある．開発者向けのページ [23]より，作成するアプリケーションを登録することで CONSUMER_KEY と CONCUMER_SECRET が割り当てられ，それを本機能で使用する(図 5-25)．本機能は投稿と DM を利用するため，アクセスレベルで「Read, write, and direct messages」を選択する．

また，本機能はインターネット通信を行うため，AndroidManifest．xml でインターネット接続のパーミッションの設定をした．

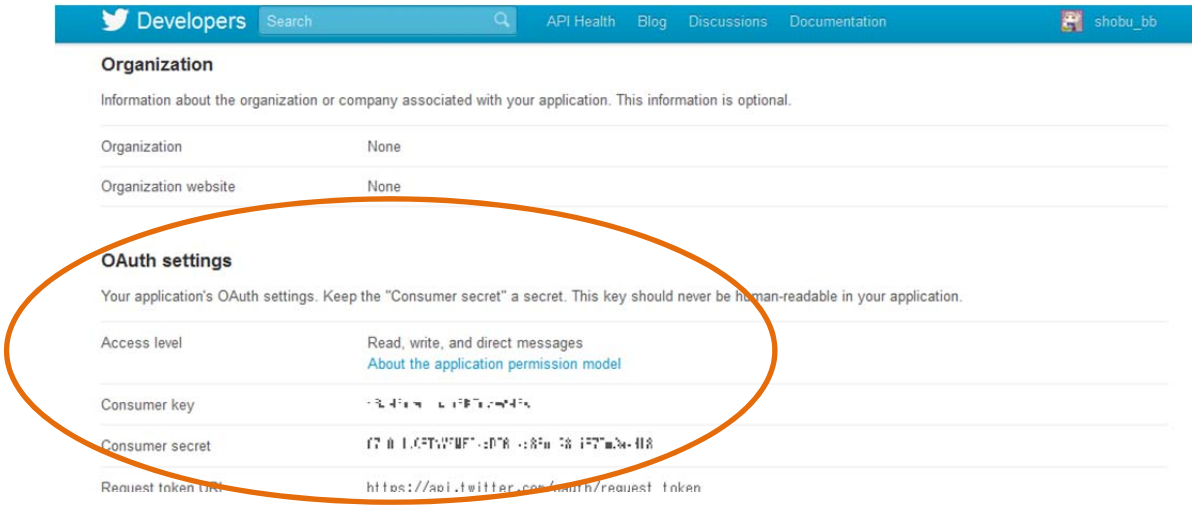


図 5-25　アクセスレベルの設定と KEY の取得

5.6.6　　Twitter 4J　（菖蒲澤担当）

Twitter を利用するためには API の利用登録のほかに，Ouath 認証[9]など複雑な処理が必要になる．そのため，今回は処理を簡略化するために「Twitter4J」というライブラリ[10]を利用した [24]．Twitter4j のダウンロードページより最新のライブラリをダウンロードし，その中のファイルを本アプリケーションで使用している(図 5-26)．2013 年 2 月現在では無料でダウンロードすることができる．

フォルダにコピーするだけでなく，ビルドパス[11]に追加をする必要もある(図 5-27)．そうすることによって Twitter4j が利用できるようになり，Ouath 認証の処理を簡略化することができる．

図 5-26 Twitter4J の利用

図 5-27 Twitter4J ビルドパス追加

9 インターネットでのアカウントなどの ID 認証の一種.
10 複数のプログラムをまとめたもの.
11 作成したプロジェクトに必要なライブラリ.

5.6.7　　待ち受け ON 時の処理　（菖蒲澤担当）

待ち受け状態を ON にすることで 図 5-28 のプログラムにより，入力したキーワードを含むつぶやきが投稿される．本アプリケーションは機能をわかりやすくするため，つぶやきが投稿されるようにしてあるが，防犯に役立てるために，投稿されずキーワードのみを登録し待ち受け状態を ON にすることが，プログラムの書き換えにより可能である．

```
openButton. setOnCheckedChangeListener(new OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView,  boolean
        isChecked) {
                if (isChecked) {
                if (isConnected()) {
                        Log. d(TAG, "already connected");
// Twitter認証済みの場合
                        createTwitterInstance();
                        mKeyword = mEditText. getText(). toString();
// 起こして欲しい時間とキーワードをツイート
        new Thread(new Runnable() {
        public void run() {
                try {
                        mTwitter. updateStatus("カーテンオープン待ち受け中！"
                        + "キーワードは「" + mEditText. getText() + "」");
                        } catch (TwitterException e) {
                        e. printStackTrace();
                }
        }
                        }). start();
                        Log. d(TAG, "オープンメンションwatch");
// streaming APIでmention watch開始
                        mTwitterStream. addListener(openUserStreamAdapter);
                        mTwitterStream. user();
                        Log. d(TAG, "オープンメンション終了");

        ～省略～
```

図 5-28 待ち受け ON 時のプログラム

5.6.8　　カーテンの開閉方法　(菖蒲澤担当)

Twitter 上で，本機能使用者のつぶやきに表記されているキーワードを含む返信，もしくはあらかじめ約束されたキーワードを DM で送信することで, 図 5-29 のプログラムによりカーテンが開閉する.

```
UserStreamAdapter openUserStreamAdapter = new UserStreamAdapter() {
        public void onStatus(Status status) {
                try {
                        String name = mTwitterStream. getScreenName();
                        String tweet = status. getText();
                        Log. v(TAG,  "@" + status. getUser(). getScreenName() + " :
                        " + tweet);
// 自分宛のメンションかつ指定したキーワードを含む場合
                if (tweet. startsWith("@" + name) && tweet. contains(mKeyword))
        {
                        openIntent();
                        }
                        } catch (IllegalStateException e) {
                                e. printStackTrace();
                        } catch (TwitterException e) {
                                e. printStackTrace();
                        }
                }
        public void onDirectMessage(DirectMessage directMessage){
                try{
                        String dm = directMessage. getText();
                        Log. v(TAG,  "dm: " + dm);
                        if (dm. contains(mKeyword)) {
                        openIntent();
                        }
                        }catch (IllegalStateException e) {
                        e. printStackTrace();
                        }
                }
        ～省略～
```

図 5-29 Twitter を利用したカーテンの開閉

第6章　実際のテスト　(町田担当)

6.1　テストの目的　(町田担当)

本研究で製作した機体・回路・Android アプリケーションを連携して動作させるテストを行った.

6.2　テストの目標　(町田担当)

・機体は，ラインが空転せずにカーテンを動かすことができるかを検証する.
・回路はカーテンの正転，反転，の切り替えと自動停止が機能するかを検証する.
・Android アプリケーションは音声認識やタイマー，遠隔操作機能が正常にするかを検証する.

研究全体の目標は一連の動作が正常に行われることを確かめる.

これらを以下の表 6.1 にまとめる.

表 6.1 目標の一覧表

名称	目標
機体	ラインが空転せずにカーテンを動かすことができるかを検証する.
回路	カーテンの正転，反転，の切り替えと自動停止が機能するかを検証する.
Android アプリケーション	音声認識やタイマー，遠隔操作機能が正常にするかを検証する.
研究全体	一連の動作が正常に行われるか

6.3　テスト方法　(町田担当)

テストは以下の手順で行う.

1 Android 端末とカーテンをテスト環境で示したとおり 4~5m ほど離れる

2 アプリケーション後 Bluetooth 接続したことを確認する.

3 各機能(音声認識・リモコン・タイマー・外出先からの遠隔操作)を使用してカーテンの OPEN・CLOSE の実験をする.

4 Android アプリケーションの各機能の実験をしてから回路の機能である自動停止機能を実験する.

5 アプリケーション終了したときに Bluetooth 接続が終了したことを確認する.

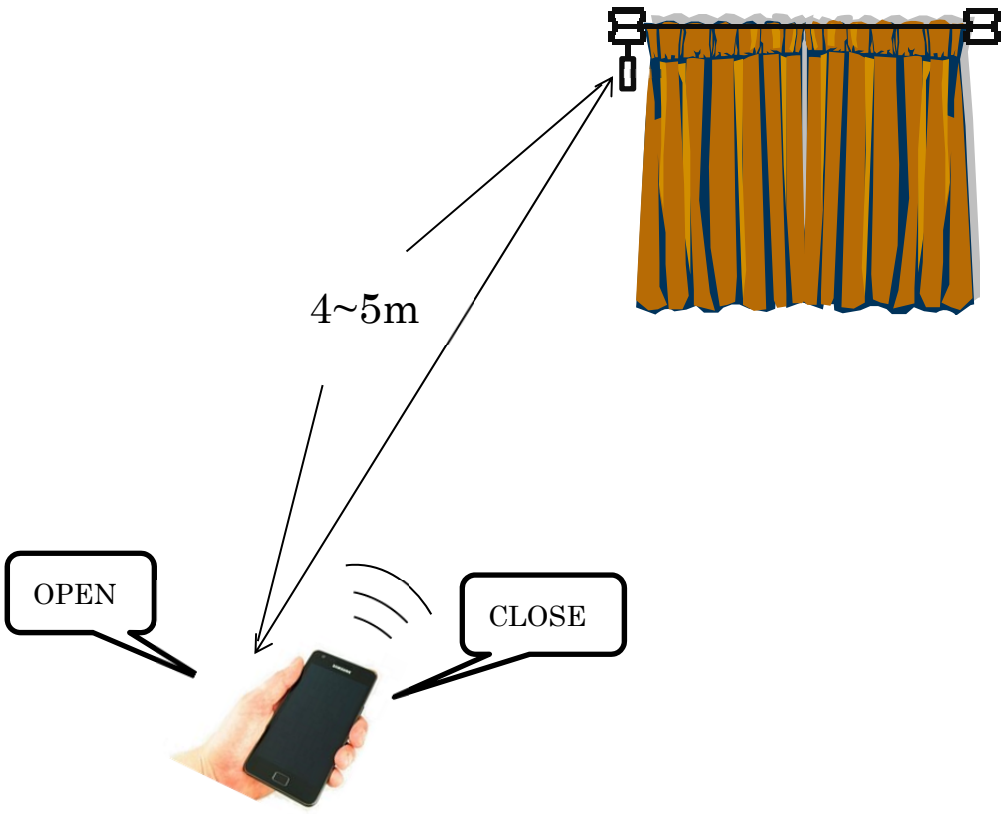


図 6-1 テストイメージ図

6.4　テスト環境　(町田担当)

テスト場所は犬目キャンパス二号館一階と二階の間に設置されているカーテンである．

実際のテストを行う環境は実際の部屋の大きさを考えて Android とカーテンの距離を 4~5m 程離れることにした．これにより多くの部屋を仮想できると考える．図 6-2 は実際にテストを行った場所の画像である．設置されているカーテンの詳しい数値は表 6.2 でまとめてあり，カーテンレールの詳しい数値は表 6．3 でまとめてある．

図 6-2 犬目キャンパス一階と二階の踊り場にあるカーテン

表 6.2 カーテン数値表

名称	数値
高さ	655mm
幅（二枚）	2040mm
重さ	700g

表 6．3 カーテンレール数値表

名称	数値
高さ	13mm
幅	17mm
長さ	2040mm

6.5　テスト結果　(町田担当)

機体はプーリが空転せずにラインを押し出すことに成功した．後付けで犬目キャンパス一階と二階の間の踊り場にあるカーテンレールに取り付けることに成功した．

回路は問題なく動作した．

Android アプリケーションは正常に動作した．

研究全体のテストの結果は機体・回路・Android アプリケーションが連動してカーテンの自動開閉が成功した．

6.6　テストの考察　(町田担当)

研究全体では機体，回路，Android アプリケーションの一連の動作は成功した．

機体は発生する恐れがあったプーリの空転も起こらずラインを送り出すことに成功した．

これにより機体の要求機能である「ラインの脱線防止」が証明された．また後付けにも成功した．

回路は動作するが回路自体に幾らか強度が足りないと考えられる．

Android アプリケーションは回路との連動しモータを動かす事が出来た．

今回のテストの結果や考察を踏まえ次節に結論を述べる．

第7章　結論 (町田担当)

7.1　目標との比較 (町田担当)

機体はシンプルな構造を考案し，コンパクトな設計・製作ができた．また要求機能である回路の搭載・脱線防止を実現することができた．しかし，製作して検証してみたところ，よりコンパクトな設計などが可能であると考えられる．今後，材料力学から必要最低限な材料強度の算出やよりコンパクトな設計を目指していけると考える．

回路は当初の要求機能として挙げたモータの制御，Android 端末との無線通信，モータの自動停止制御，基板への実装及び機体の搭載は達成することが出来た．しかしモータの自動停止制御の部分でモータが止まるまでの時間にばらつきがあることや，機体への実装という点では外から Bluetooth が繋がっているかがわかりづらいという問題も発生した．今後はこういった問題点を改善し，よりわかりやすい制御ができると考えている．

Android は要求機能である，Bluetooth を用いた開閉操作，タイマーによるカーテンの自動開閉，音声認識によるカーテンの自動開閉，外出先からの遠隔操作によるカーテンの自動開閉，の全てを実現できた．

7.2　総合評価 (町田担当)

本研究の目標でもある，「Android アプリケーションを用いて自動でカーテンを動かすこと」といった目的は達成できた．その為，本研究での目標であった「Android アプリケーションを用いて人々の暮らしを豊かにすること」ができ，後付けという機能で「今までにない自動カーテンシステム」も達成できたと考えられる．そして今回行ったテストでは実際使用する環境でテストした．そして予定していた全ての機能が問題なく動作した．これによって本研究での成果は実際に使用できるレベルだと判断できる．

しかし実際に商品として販売することを考えたとき，機体や回路のコストを考える必要がある．機体は使用している部材や大きさなどを強度や安全率などを考えながらコストを抑えることができ，回路も部品やシステムの簡略化ができると考える．

参考文献・URL

1. BLOGS 国内で来年にはスマートフォン普及率が過半か.
http://blogos.com/article/50893/.
2. 白物家電もクラウド連携、パナソニックが「スマート家電」6 機種を一斉投入.
http://blog.goo.ne.jp/kamiyadori-deep/e/264a3f5d3c30ac7926b02a7ffd8b0228.
3. ななめドラム洗濯乾燥機.
http://panasonic.jp/wash/d_w_desiccate/NA-VX8200/#hinban=NA-VX8200&recNum=p00&scrNum=block-01.
4. スマート家電シリーズ登場!「パナソニックはスマート家電へ」より.
http://denki-koji.info/blog/archives/1396.
5. タチカワ　電動カーテンレール「シエルド 50」新発売.
http://online.ibnewsnet.com/news/file_n/cn2012/cn120423-02.html.
6. Arduino と MP4212 を使ったモータドライバ.
http://d.hatena.ne.jp/nisuseteuryalus2/20110124/1295798111.
7. Arduino と Android を Bluetooth 連携.
https://sites.google.com/a/gclue.jp/android-docs-2009/bluetoothdearduinoto-lian-xie.
8. Arduino リファレンス. http://www.musashinodenpa.com/Arduino/ref/.
9. 電流センサ ACS712 の概要.
http://www.eleki-jack.com/ejackino/2010/03/no19-ejackinolcd11.html.
10. 電流測定を行う 2.
http://www.eleki-jack.com/ejackino/2010/03/no20-ejackinolcd13.html.
11. 電流測定を行う 3.
http://www.eleki-jack.com/ejackino/2010/03/no21-ejackinolcd14.html.
12. Android プログラマへの道 ～ Moonlight 明日香 ～.
http://wiki.livedoor.jp/moonlight_aska/d/%A5%A4%A5%E1%A1%BC%A5%B8%A5%DC%A5%BF%A5%F3%A4%F2%BA%EE%C0%AE%A4%B9%A4%EB.
13. Android プログラマへの道 ～ Moonlight 明日香 ～.
http://wiki.livedoor.jp/moonlight_aska/d/%B2%E8%CC%CC%A4%F2%C1%AB%B0%DC%A4%B9%A4%EB.
14. ど素人の Android アプリ開発入門. intent による画面遷移.
http://androidhacker.blog94.fc2.com/blog-entry-26.html.
15. HyperLandWeBlog. ボタン半透明化.
http://www.hlplus.jp/php/blog/index.php?UID=1303721721.

16. AndroDocs. Bluetooth 接続.
https://sites.google.com/a/gclue.jp/android-docs-2009/bluetoothdearduinoto-lian-xie.
17. 海の河童、デバイスで食う(目標). Bluetooth 接続.
http://umikappa.main.jp/20110708-148.
18. 組み込み Android な元気人を探せ！(3). 音声認識.
http://www.kumikomi.net/archives/2012/03/co09and3.php?page=4.
19. 浅草ギ研. 音声認識.
http://www.robotsfx.com/robot/robohow/RoboHow161/RoboHow161.html.
20. ぼっちな在宅プログラマーの開発覚書. タイマー機能.
http://ohwhsmm7.blog28.fc2.com/blog-entry-304.html.
21. 伊藤　元. Android×Arduino でつくるクラウド連携デバイス　-Android ADK で電子工作をはじめよう！-:インプレスジャパン, 2012.
22. Twitter. https://twitter.com/.
23. Twiiter Developers.　https://dev.twitter.com/.
24. Twitter4J.　http://twitter4j.org/ja/index.html.
25. BLOGS 国内で来年にはスマートフォン普及率が過半か.
http://blogos.com/article/50893/.
26. アイフォーン抜いたアンドロイド　日本でもスマートフォンの主流に.
http://www.j-cast.com/2011/10/13109752.html?p=all.
27. スマートハウス関連主要設備機器に関する調査結果 2012.
http://www.yano.co.jp/press/pdf/942.pdf.
28. 東京は朝の 6 時-朝活者は 19%.
http://blog.jma-net.jp/article/166645959.html.
29. 光害対策ガイドラインにおける光害の定義―環境省.
http://www.env.go.jp/air/life/m-syomei/04.pdf.
30. 地域照明環境計画策定マニュアル.： 環境庁大気保全局.
31. 光害 - Wikipedia より.
http://ja.wikipedia.org/wiki/%E5%85%89%E5%AE%B3.
32. 企業やご家庭の節電対策に　節電カーテン特集　カーテンくれないダイレクト.
http://www.e-kurenai.com/fs/curtain/c/setsuden.

謝辞　(藤原担当)

最後に，この場をお借りして 2 年間多大なご指導頂きました金丸隆志准教授，新井敏夫教授，開発にあたり親身に対応して頂いた山名徹教授，花野井利之様，研究における私たちの手本となり快く関わって頂いた修士の先輩方，切磋琢磨し互いに高め合った学友の皆様に深く感謝致します．駄文ではありますが，これを持って謝辞とさせていただきます．

2012 年度(平成 24 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

リーダー　　：G109029 河合一平太

サブリーダー：G109035 黒澤祐喜

：G109011 岩本涼平

：G109063 藤原航平

：G109066 町田匠

：G108403 菖蒲澤宗成

提出日：2013 年 2 月 12 日

目次

第1章　Arduino の基礎

1.1　Arduino での回路図・回路の写真及びソースプログラム

1.1.1　モータ制御

図 1-1 は本研究で製作したモータ制御の回路図である.

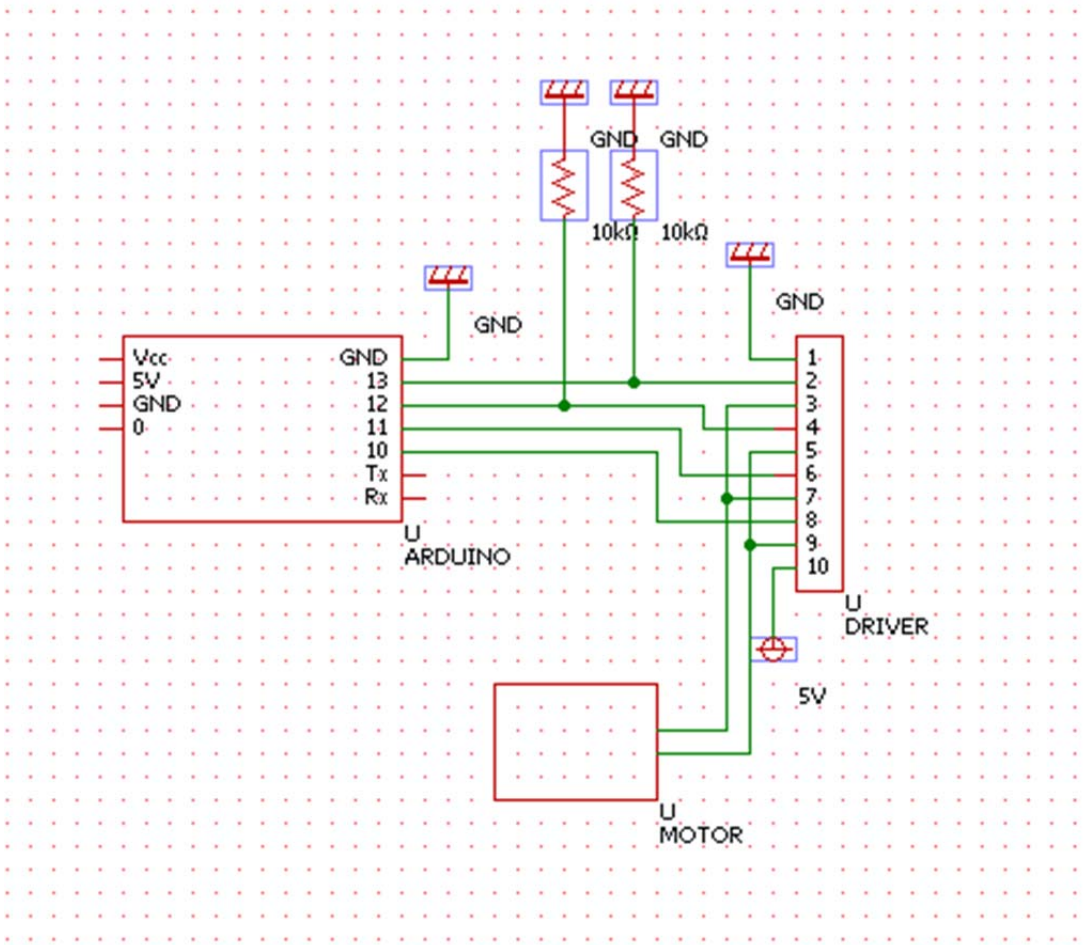


図 1-1　モータ制御の回路図

図 1-2 はブレッドボードに実装したモータ制御回路の写真である.

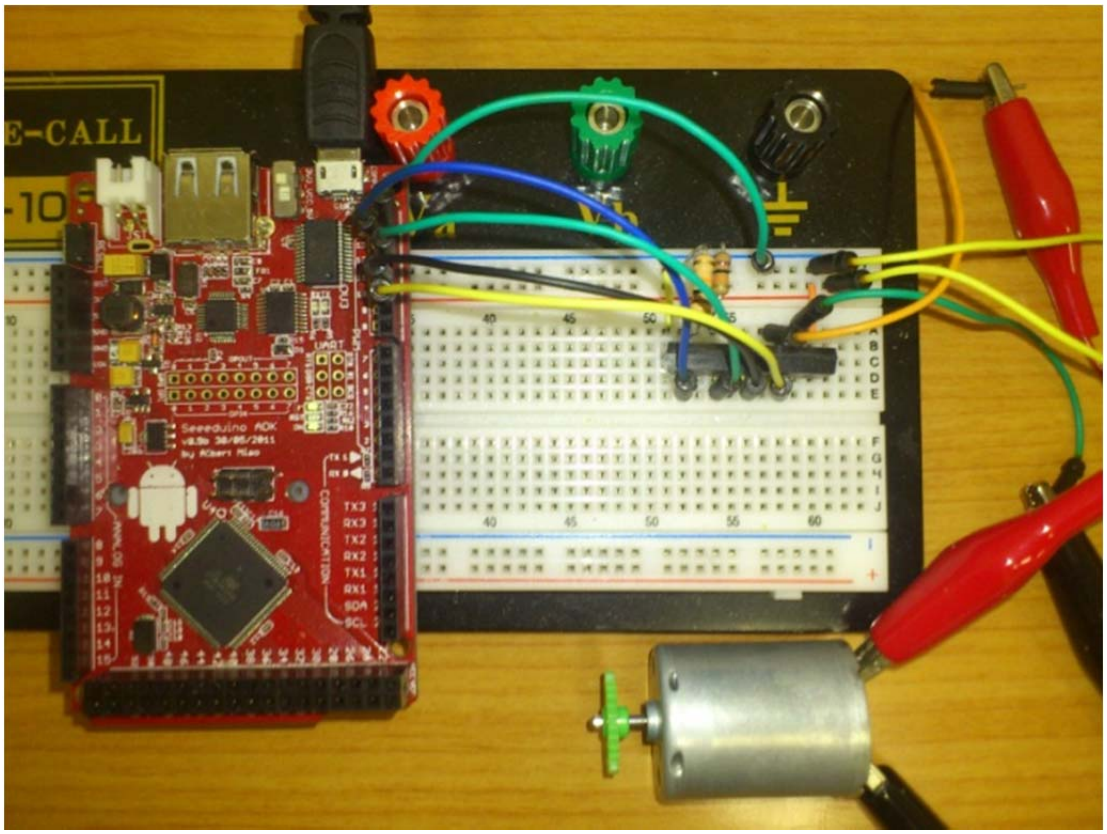


図 1-2 回路の写真

以下はモータ制御のプログラムである.

```
void setup() {
//4 つの端子を出力にする
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);

}

void loop() {
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  digitalWrite(10, LOW);
  delay(5000);
  digitalWrite(13, LOW);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  delay(5000);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);
  digitalWrite(10, HIGH);
  delay(5000);
  digitalWrite(13, LOW);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  delay(5000);
}
```

1.1.2 Bluetooth 通信によるモータ制御

図 1-3 は本研究で製作した Bluetooth 通信によるモータ制御の回路図である.

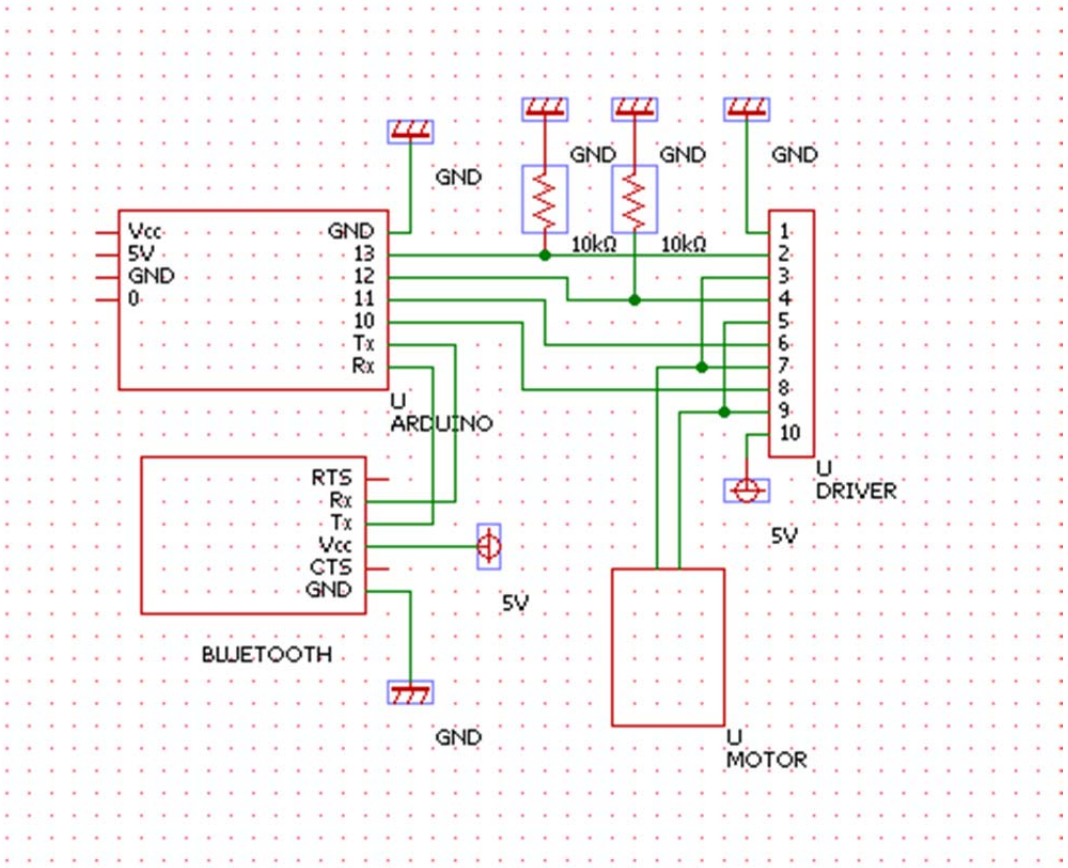


図 1-3 Bluetooth によるモータ制御の回路図

図 1-4 はブレッドボードに実装した Bluetooth 通信によるモータ制御回路の写真である.

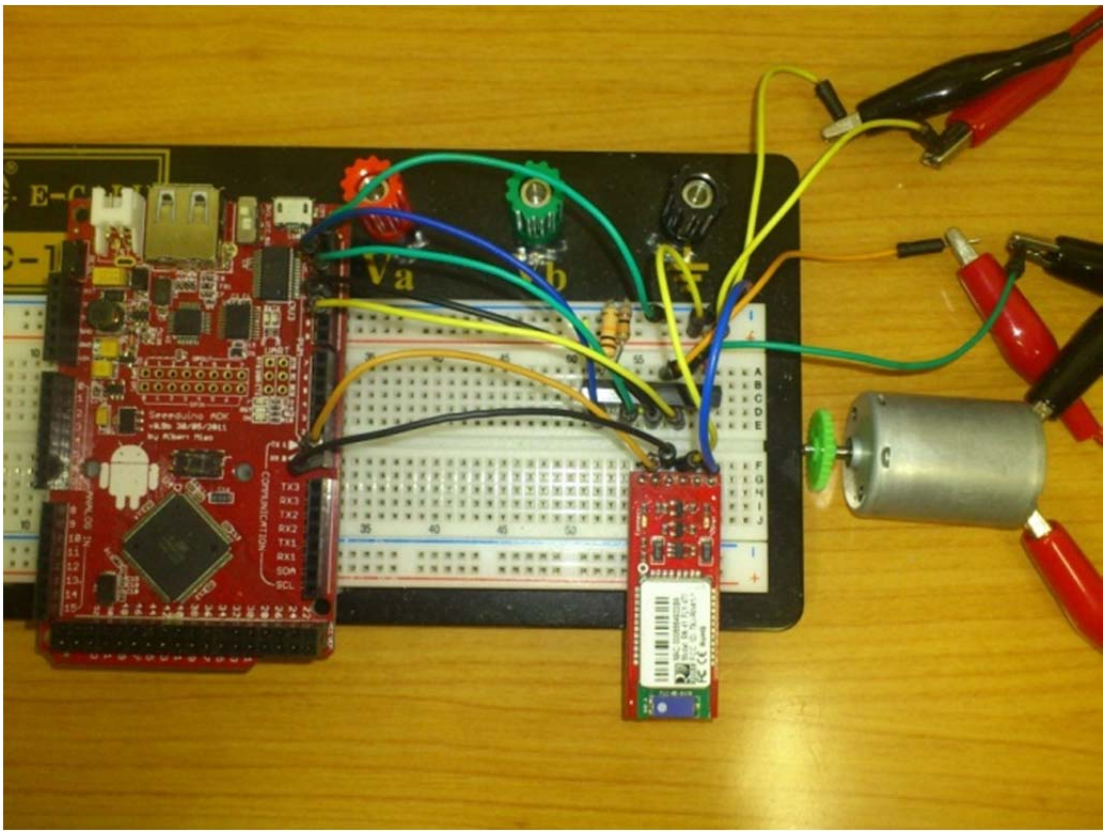


図 1-4 Bluetooth によるモータ制御の回路の写真

以下に Bluetooth 通信によるモータ制御のプログラムである.

```
#define   LED1   13 //
#define   LED2   12 //
#define   LED3   11 //
#define   LED4   10 //

int commandId;

//初期設定
void setup()
{
  Serial.begin(115200);

  pinMode(LED1, OUTPUT); //出力ポートにする
  pinMode(LED2, OUTPUT); //出力ポートにする
  pinMode(LED3, OUTPUT); //出力ポートにする
  pinMode(LED4, OUTPUT); //出力ポートにする

  //初期状態はモータ off
  digitalWrite(LED1, LOW);//LED を点灯する(6)
  digitalWrite(LED2, LOW);//LED を点灯する(6)
  digitalWrite(LED3, HIGH);//LED を点灯する(6)
  digitalWrite(LED4, HIGH);//LED を点灯する(6)

}

void loop(){
  if (Serial.available() > 0) {          //android 端末から信号が来てたら実行
       commandId = Serial.read();        //送信された信号を commandId という変数に代入
    Serial.println("0");

    if(commandId == 48 /*'0'*/){          //android から"0"という信号が来たら停止
         digitalWrite(LED1, LOW);
          digitalWrite(LED2, LOW);
          digitalWrite(LED3, HIGH);
```

```
            digitalWrite(LED4, HIGH);
        }else if (commandId == 49 /*'1'*/ ) {        //android から"1"という信号が来たら正転
            digitalWrite(LED1, HIGH);
            digitalWrite(LED2, LOW);
            digitalWrite(LED3, HIGH);
            digitalWrite(LED4, LOW);
        } else if(commandId == 50 /*'2'*/ ){         //android から"2"という信号が来たら反転
           digitalWrite(LED1, LOW);
            digitalWrite(LED2, HIGH);
            digitalWrite(LED3, LOW);
            digitalWrite(LED4, HIGH);
        }
    } else{                                          //android 端末から信号が無かったら実行
            digitalWrite(LED1, LOW);
            digitalWrite(LED2, LOW);
            digitalWrite(LED3, HIGH);
            digitalWrite(LED4, HIGH);
    }
    delay(100); //10 ミリ秒処理を停止(10 ミリ秒おきに loop()を繰り返す)
}
```

1.1.3　　完成した回路

図 1-5 は完成した回路図である.

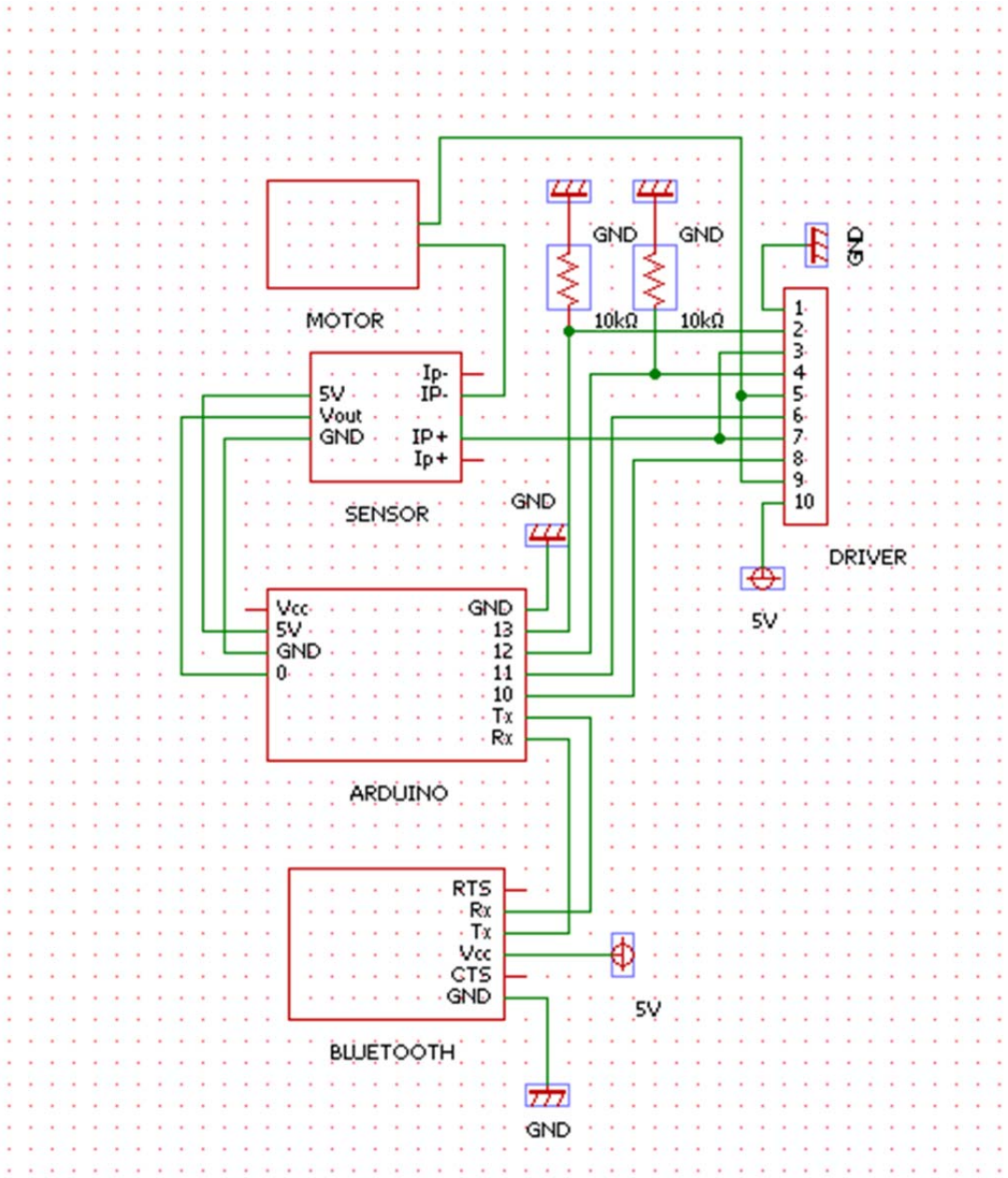


図 1-5 完成した回路図

図 1-6 は完成した回路の写真である．DC ジャックを使用することで家庭用コンセントからでも使用することが可能である．

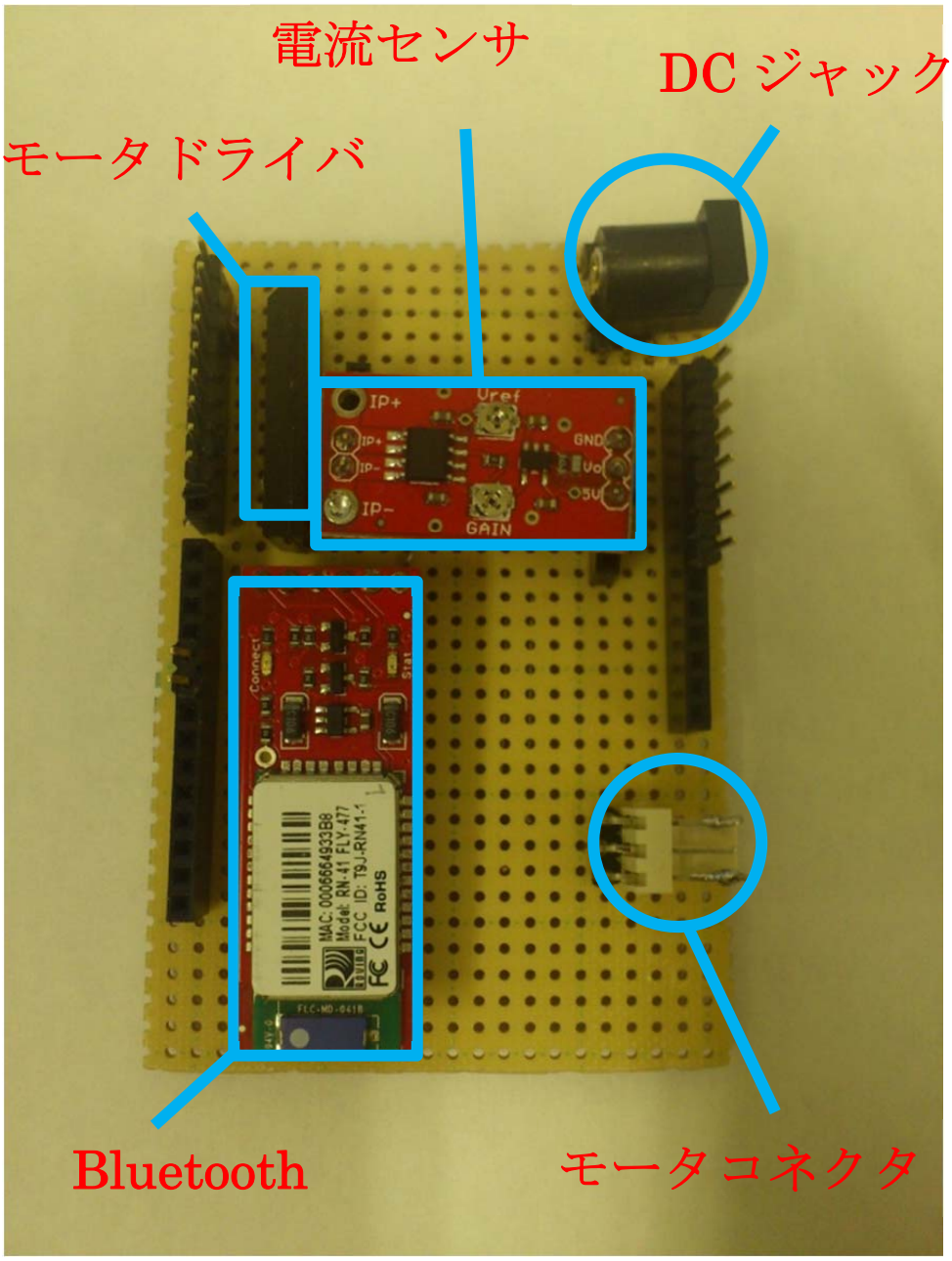


図 1-6 完成した回路

以下に Bluetooth 通信によるモータ制御のプログラムである.

```
#define   LED1   13
#define   LED2   12
#define   LED3   11
#define   LED4   10
#define CURTAIN_STOP 48
#define CURTAIN_OPEN 49
#define CURTAIN_CLOSE 50

// 1 なら自動停止あり、0 ならなし
#define auto_mode 1

int commandId = CURTAIN_STOP;
int CurrentValue1=0;   //電流値の読み込み 1
int exceed_count=0;

//初期設定
void setup(){
  Serial.begin(115200);
  pinMode(LED1, OUTPUT); //出力ポートにする
  pinMode(LED2, OUTPUT); //出力ポートにする
  pinMode(LED3, OUTPUT); //出力ポートにする
  pinMode(LED4, OUTPUT); //出力ポートにする

  //初期状態はモーターoff
  curtainStop();
}

void loop(){
  CurrentValue1=   analogRead(0);      //通常測定の電流値

  if(auto_mode){
    if(commandId == CURTAIN_OPEN || commandId == CURTAIN_CLOSE){
      while(commandId == CURTAIN_OPEN || commandId == CURTAIN_CLOSE){
        CurrentValue1=   analogRead(0);
```

```
        if( CurrentValue1>=650    // when opening
            || CurrentValue1<=310    // when closing
            ){
            exceed_count += 1;
          }
          if(exceed_count>10){
            curtainStop();
          }
          if (Serial.available() > 0){
            commandId = Serial.read();
          }
          Serial.println(CurrentValue1);
          delay(30);
        }
        exceed_count=0;
    }else{

        if (Serial.available() > 0){
          commandId = Serial.read();
        }
        Serial.println(CurrentValue1);
    }

  }else{
        if (Serial.available() > 0){
          commandId = Serial.read();
        }
        Serial.println(CurrentValue1);
  }

  switch(commandId){
    case CURTAIN_STOP:
      curtainStop();
      break;
    case CURTAIN_OPEN:
```

```
        curtainOpen();
        break;
      case CURTAIN_CLOSE:
        curtainClose();
        break;
      default:
        curtainStop();
        break;
  }

  delay(40);
}

void curtainStop(){
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
  digitalWrite(LED4, HIGH);
}

void curtainOpen(){
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
  digitalWrite(LED4, LOW);
}

void curtainClose(){
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, HIGH);
}
```

第2章　DroidCurtain ソースコード

2.1　Android ソースコード

以下に本研究で製作した Android アプリケーションで用いた JAVA プログラムのソースコードを記述する.

ソースコード	機能
AndroidManifest.xml	アプリの動作やパーミッションについて記述
res/layout/main.xml	画面のレイアウト
src/ ~/~java.	アプリの機能について記述
res/values/strings.xml	文字列や色などについて記述
ライブラリ	クラスファイルを圧縮してるもの

AndroidManifest.xml

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

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="15" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".Home"
            android:label="@string/title_activity_home" >
            <intent-filter>
```

```
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER"
/>
          </intent-filter>
       </activity>
       <activity
          android:name=".Timer"
          android:label="タイマー">
          </activity>
        <activity
          android:name=".Timer_Open"
          android:label="Opening...">
          </activity>
        <activity
          android:name=".Timer_Close"
          android:label="Closing...">
          </activity>
          <activity
          android:name=".Voice_Remote"
          android:label="リモコン！">
          </activity>
          <activity
          android:name=".AndroidGE"
          android:label="工学院大学GE学部パソコン部">
          </activity>
         <activity
          android:name=".SNS"
          android:label="つぶやく！">
       </activity>
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT"
/>
            <category android:name="android.intent.category.BROWSABLE"
/>
            <data android:scheme="oauth" android:host="t4jsample"/>
```

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

res/layout/main.xml

home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/home" >

    <ImageButton
        android:id="@+id/TimerButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="71dp"
        android:layout_marginTop="31dp"
        android:src="@drawable/timer2" />

    <ImageButton
        android:id="@+id/RemoteButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/TimerButton"
        android:layout_below="@+id/TimerButton"
        android:layout_marginTop="24dp"
        android:src="@drawable/remote2" />

    <ImageButton
        android:id="@+id/SettingButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/RemoteButton"
```

```
        android:layout_marginLeft="20dp"
        android:layout_toRightOf="@+id/RemoteButton"
        android:src="@drawable/setting2" />

    <ImageButton
        android:id="@+id/SnsButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/SettingButton"
        android:layout_alignTop="@+id/TimerButton"
        android:src="@drawable/sns2" />

</RelativeLayout>
```

sns.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"
    android:padding="5dp"
    android:background="#A61C69" >

    <TimePicker
        android:id="@+id/timePicker1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

    <LinearLayout
        android:id="@+id/formContainer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="キーワード"
            android:textColor="#D3E5E5"
            android:textAppearance="?android:attr/textAppearanceLarge" />

        <EditText
            android:id="@+id/editText1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="起きろ" >

            <requestFocus />
```

```
    </EditText>
</LinearLayout>

    <LinearLayout
    android:id="@+id/formContainer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="開ける"
        android:textColor="#D3E5E5"
        android:textAppearance="?android:attr/textAppearanceLarge" />

     <ToggleButton
    android:id="@+id/toggleButton1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textOff="つぶやく"
            android:textOn="つぶやきました" />
        <requestFocus />
</LinearLayout>

           <LinearLayout
    android:id="@+id/formContainer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="閉める"
        android:textColor="#D3E5E5"
```

```
            android:textAppearance="?android:attr/textAppearanceLarge" />

        <ToggleButton
        android:id="@+id/toggleButton2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textOff="つぶやく"
                android:textOn="つぶやきました" />
            <requestFocus />
    </LinearLayout>

</LinearLayout>
```

timer_close_.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#A61C69"
    >

    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="  STOP  "
        android:textColor="#E33830"
        android:textSize="15pt" />

        <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="27dp"
        android:text="カーテンが閉じていきます..."
        android:textColor="#D3E5E5"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

timer_open.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#A61C69"
    >

    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="  STOP  "
        android:textColor="#E33830"
        android:textSize="15pt" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="27dp"
        android:text="カーテンが開いていきます..."
        android:textColor="#D3E5E5"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

timer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#A61C69" >

    <Button
        android:id="@+id/open"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:layout_marginLeft="18dp"
        android:text=" Open "
        android:textColor="#01AC5E"
        android:textSize="15pt"/>

    <Button
        android:id="@+id/close"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/open"
        android:layout_alignBottom="@+id/open"
        android:layout_alignParentRight="true"
        android:layout_marginRight="16dp"
        android:text=" Close "
        android:textColor="#E33830"
        android:textSize="15pt" />

    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="タイマーの設定！"
        android:textColor="#D3E5E5"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TimePicker
        android:id="@+id/timePicker1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/text1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp" />

    <Button
        android:id="@+id/cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/open"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="38dp"
        android:text="Cancel"
        android:textColor="#0075C2"
        android:textSize="15pt" />

</RelativeLayout>
```

voice_remote.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#A61C69" >

    <Button
        android:id="@+id/open"
        android:layout_width="200dp"
        android:layout_height="80dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:text="OPEN"
        android:textSize="20pt"
        android:textColor="#01AC5E"/>

    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="65dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/open"
        android:layout_marginTop="20dp"
        android:text="STOP"
        android:textSize="18pt"
        android:textColor="#F3983B"/>

    <Button
        android:id="@+id/close"
        android:layout_width="200dp"
        android:layout_height="80dp"
```

```
        android:layout_alignLeft="@+id/open"
        android:layout_below="@+id/stop"
        android:layout_marginTop="20dp"
        android:text="CLOSE"
        android:textSize="20pt"
        android:textColor="#E33830"/>

    <Button
        android:id="@+id/voice"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="72dp"
        android:text="ひらけごま！"
        android:textSize="15pt"
        android:textColor="#0075C2"/>

</RelativeLayout>
```

values/strings.xml

strings.xml

```
<resources>

    <string name="app_name">DoroidCurtain</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_home">ドロイドカーテン</string>

</resources>
```

Styles.xml

```
<resources>

    <style name="AppTheme" parent="android:Theme.Light" />

</resources>
```

src/com.droidcurtain

Home.java

```
package com.doroidcurtain;


import android.os.Bundle;
import android.app.Activity;
//import android.view.Menu;
import android.view.View;
//import android.widget.Button;
import android.widget.ImageButton;
import android.view.View.OnClickListener;
import android.content.Intent;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.view.KeyEvent;

public class Home extends Activity {

        public ImageButton        imageButton1,
                                                imageButton2,
                                                imageButton3,
                                                imageButton4;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.home);

    imageButton1 = (ImageButton)findViewById(R.id.RemoteButton);
    imageButton2 = (ImageButton)findViewById(R.id.TimerButton);
    imageButton3 = (ImageButton)findViewById(R.id.SettingButton);
    imageButton4 = (ImageButton)findViewById(R.id.SnsButton);
```

```
imageButton1.setOnClickListener(new OnClickListener(){

    public void onClick(View v) {
            Intent intent = new Intent(Home.this, Voice_Remote.class);
            startActivity(intent);

    }

});

imageButton2.setOnClickListener(new OnClickListener(){

    public void onClick(View v) {
        Intent intent = new Intent(Home.this, Timer.class);
        startActivity(intent);

        }
    });
    imageButton4.setOnClickListener(new OnClickListener(){

    public void onClick(View v) {

            Intent intent = new Intent(Home.this, SNS.class);
        startActivity(intent);
                        }

    });
}    // BACK ボタンが押された時の処理
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if(keyCode==KeyEvent.KEYCODE_BACK){
        // アラートダイアログ
        showDialog(0);
        return true;
    }
    return false;
```

```
    }

    // アラートダイアログ
    @Override
    public Dialog onCreateDialog(int id) {
        switch (id) {

        case 0:
            //ダイアログの作成(AlertDialog.Builder)
            return new AlertDialog.Builder(Home.this)
            .setMessage("「アプリ」を終了しますか?")
            .setCancelable(false)
            // 「終了する」が押された時の処理
            .setPositiveButton("終了する", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    // アクティビティ消去
                    Home.this.finish();
                }
            })
            // 「終了しない」が押された時の処理
            .setNegativeButton("終了しない", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {

                }
            })
            .create();
        }
        return null;
    }

}
```

sns.java

```
package com.doroidcurtain;


import twitter4j.Status;
import twitter4j.Twitter;
import twitter4j.TwitterException;
import twitter4j.TwitterFactory;
import twitter4j.TwitterStream;
import twitter4j.TwitterStreamFactory;
import twitter4j.UserStreamAdapter;
import twitter4j.auth.AccessToken;
import twitter4j.auth.RequestToken;
import twitter4j.conf.Configuration;
import twitter4j.conf.ConfigurationBuilder;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.net.Uri;
import android.os.Bundle;
//import android.os.Handler;
import android.util.Log;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.TimePicker;
//import android.widget.Toast;
import android.widget.ToggleButton;


public class SNS extends Activity {
	static final String TAG = "SAlarm";

	// Twitter 上でアプリ登録した際に得られるアプリ固有値
```

```
static String CONSUMER_KEY = "与えられた KEY";
staticStringCONSUMER_SECRET = "与えられた KEY";

// Preference に保存するときのファイル名,キー名
static String PREFERENCE_NAME = "twitter_oauth";
static final String PREF_KEY_SECRET = "oauth_token_secret";
static final String PREF_KEY_TOKEN = "oauth_token";

// Twitter 認証後にコールバックされる URL
static final String CALLBACK_URL = "oauth://t4jsample";

// OAuth 認証のコールバック URL に含まれる OAuth 認証コードのパラメータ名
static final String IEXTRA_OAUTH_VERIFIER = "oauth_verifier";

private ToggleButton openButton;
private ToggleButton closeButton;
private EditText mEditText;
private TimePicker mTimePicker;

private static SharedPreferences mSharedPreferences;

private static Twitter mTwitter;
private static TwitterStream mTwitterStream;
private static RequestToken mRequestToken;
private String mKeyword;
private Intent openAlarm;
private Intent closeAlarm;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.sns);

   mEditText = (EditText)findViewById(R.id.editText1);
   mTimePicker = (TimePicker)findViewById(R.id.timePicker1);
   mTimePicker.setIs24HourView(true);
```

```
        openButton = (ToggleButton) findViewById(R.id.toggleButton1);
        closeButton = (ToggleButton) findViewById(R.id.toggleButton2);


    openButton.setOnCheckedChangeListener(new OnCheckedChangeListener() {

   public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
                if (isConnected()) {
        Log.d(TAG,"already connected");
                // Twitter 認証済みの場合

                createTwitterInstance();
                mKeyword = mEditText.getText().toString()
               // 起こして欲しい時間とキーワードをツイート
                new Thread(new Runnable() {
                public void run() {
        try {

        mTwitter.updateStatus(mTimePicker.getCurrentHour()        +        ":"        +
mTimePicker.getCurrentMinute()
                                + "に起こしてください！" + "　「" + mEditText.getText() + "」
" + "←のキーワードを使って,リプライ or メンションをください！" );//+
System.currentTimeMillis());
                                                                        }                catch
(TwitterException e) {
e.printStackTrace();
                                                                        }}}).start();
Log.d(TAG,"オープンメンション watch");
                        // streaming API で mention watch 開始

        mTwitterStream.addListener(openUserStreamAdapter);
        mTwitterStream.user();
        Log.d(TAG,"オープンメンション終了");
                                                } else {
                // Twitter 認証されていない場合は OAuth 認証を行う
```

```
        Log.d(TAG,"start askOAuth");
        new Thread(new Runnable() {

        public void run() {
        askOAuth();
        }}).start();
                                                }
                                        } else {
                    // ストリーミング API の停止,アラームの停止
        if (mTwitterStream != null) {
                                                mTwitterStream.shutdown();
                                                }
                                        }
                                }
                    });

        closeButton.setOnCheckedChangeListener(new OnCheckedChangeListener() {

        public void onCheckedChanged(CompoundButton buttonView, boolean
          isChecked) {
                if (isChecked) {
                                if (isConnected()) {
                    Log.d(TAG,"already connected");
                                // Twitter 認証済みの場合

                    createTwitterInstance();
                    mKeyword = mEditText.getText().toString();
                                // 起こして欲しい時間とキーワードをツイート
                                new Thread(new Runnable() {

public void run() {
        try {
        mTwitter.updateStatus(mTimePicker.getCurrentHour() + ":" +
mTimePicker.getCurrentMinute()

        + "にカーテンを閉めてほしいです！"
```

```
        + "　「" + mEditText.getText() + "」" + "←のキーワードを使って,リプライ or メンションをください！" );
                        //+ System.currentTimeMillis());
                                                                    } catch (TwitterException e) {

        e.printStackTrace();
                                                                    }
                                                                }
                                                            }).start();
                Log.d(TAG,"クローズメンション watch");
                    // streaming API で mention watch 開始

        mTwitterStream.addListener(closeUserStreamAdapter);
                                                            mTwitterStream.user();
                Log.d(TAG,"クローズメンション終了");
                                                    } else {
                 // Twitter 認証されていない場合は OAuth 認証を行う
                Log.d(TAG,"start askOAuth");
    new Thread(new Runnable() {

    public void run() {
                            askOAuth();
                            }
                                                            }).start();
                                                    }
                                            } else {
                 //  ストリーミング API の停止,アラームの停止
        if (mTwitterStream != null) {
                                                            mTwitterStream.shutdown();
                                                    }
                                                    //requestStopAlarm();
                                            }
                            }
                    });
```

```
        mSharedPreferences = getSharedPreferences(PREFERENCE_NAME,
MODE_PRIVATE);
                    // Twitter の認証画面からのコールバック Intent で呼ばれた時の処理
                    final Uri uri = getIntent().getData();
                    // コールバックに指定した URL から始まる URL で呼ばれたことを確認
                    if (uri != null && uri.toString().startsWith(CALLBACK_URL)) {
                              new Thread(new Runnable() {

          public void run() {
                                        // 認証コードを取得
            String verifier = uri.getQueryParameter(IEXTRA_OAUTH_VERIFIER);
                                        try {
             // トークンを取得し,トークン,シークレットを Preference に保存
                                                  AccessToken accessToken =
mTwitter.getOAuthAccessToken(mRequestToken, verifier);
                    Editor e = mSharedPreferences.edit();
                    e.putString(PREF_KEY_TOKEN, accessToken.getToken());
                    e.putString(PREF_KEY_SECRET, accessToken.getTokenSecret());
                                                  e.commit();
                                        } catch (Exception e) {
                                                  e.printStackTrace();
                                        }
                              }
                    }).start();
          }
        }

        /**
         * Twitter と接続済みか？
         *
         * @return
         */
        private boolean isConnected() {
                    return mSharedPreferences.getString(PREF_KEY_TOKEN, null) !=
null;
        }
```

```
        /**
         * OAuth 認証
         */
private void askOAuth() {
                // アプリに登録時に得られる
CONSUMER_KEY,CONSUMER_SECRET を元に
                // OAuth 認証用のを開く
        ConfigurationBuilder configurationBuilder = new ConfigurationBuilder();
                configurationBuilder.setOAuthConsumerKey(CONSUMER_KEY);

        configurationBuilder.setOAuthConsumerSecret(CONSUMER_SECRET);
                Configuration configuration = configurationBuilder.build();
                mTwitter = new TwitterFactory(configuration).getInstance();

                Log.d(TAG,"start askOAuth1");
                try {
                        Log.d(TAG,"start askOAuth4");
                        // コールバック URL を指定
        mRequestToken =mTwitter.getOAuthRequestToken(CALLBACK_URL);
                        Log.d(TAG,"start askOAuth2");
                        // 認証用 URL を開く
                this.startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse(mRequestToken.getAuthenticationURL())));
                } catch (TwitterException e) {
                        Log.d(TAG,"start askOAuth3");
                        e.printStackTrace();
                }
        }


        /**
         * Twitter API を使うためのインスタンスを生成する
         */
        private void createTwitterInstance() {
        String oauthAccessToken =
    mSharedPreferences.getString(PREF_KEY_TOKEN, "");
```

```
        String oAuthAccessTokenSecret =
    mSharedPreferences.getString(PREF_KEY_SECRET, "");

                ConfigurationBuilder confbuilder = new ConfigurationBuilder();
                Configuration conf = confbuilder

        .setOAuthConsumerKey(CONSUMER_KEY)

        .setOAuthConsumerSecret(CONSUMER_SECRET)

        .setOAuthAccessToken(oauthAccessToken)

        .setOAuthAccessTokenSecret(oAuthAccessTokenSecret)
                                                        .build();
                mTwitter = new TwitterFactory(conf).getInstance();
                mTwitterStream = new TwitterStreamFactory(conf).getInstance();
        }

        /**
         * StreamingAPI のリスナー
         */
        UserStreamAdapter openUserStreamAdapter = new UserStreamAdapter() {

                public void onStatus(Status status) {
                        try {
                                String name = mTwitterStream.getScreenName();
                                String tweet = status.getText();
                                Log.v(TAG, "@" + status.getUser().getScreenName()
+ " : " + tweet);
                // 自分宛のメンションかつ指定したキーワードを含む場合

         if (tweet.startsWith("@" + name) && tweet.contains(mKeyword)) {
                                        openIntent();
                                }
                        } catch (IllegalStateException e) {
                                e.printStackTrace();
```

```
                        } catch (TwitterException e) {
                                e.printStackTrace();
                        }
                }

        private void openIntent(){
                Log.v(TAG,"openIntent");
                openAlarm = new Intent(SNS.this , Timer_Open.class );
                startActivity(openAlarm);
        }

        };

UserStreamAdapter closeUserStreamAdapter = new UserStreamAdapter() {

    public void onStatus(Status status) {
                        try {
                                String name = mTwitterStream.getScreenName();
                                String tweet = status.getText();
                                Log.v(TAG,  "@"  +  status.getUser().getScreenName()
+ " : " + tweet);
        // 自分宛のメンションかつ指定したキーワードを含む場合

        if (tweet.startsWith("@" + name) && tweet.contains(mKeyword)) {
                                        closeIntent();
                                }
                        } catch (IllegalStateException e) {
                                e.printStackTrace();
                        } catch (TwitterException e) {
                                e.printStackTrace();
                        }
                }


        private void closeIntent(){
                Log.v(TAG,"closeIntent");
```

```
                closeAlarm = new Intent(SNS.this , Timer_Close.class );
                startActivity(closeAlarm);
        }

        };

}
```

Timer.java

```
package com.doroidcurtain;


import java.util.Calendar;
import java.util.Date;

import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TimePicker;

public class Timer extends Activity implements SharedPreferences.OnSharedPreferenceChangeListener {

	private AlarmManager alarmManager;
	private Calendar calendar;
	private TimePicker timePicker;

	private SharedPreferences prefs;

	private Intent Open1;
	private PendingIntent Open2;
	private Intent Close1;
	private PendingIntent Close2;
	private PendingIntent result;
	boolean Open = false;
```

```
 boolean Close = false;

 private int cal_hour;
 private int cal_minute;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.timer);

        // 現在時刻の取得
       calendar = Calendar.getInstance();
       calendar.setTimeInMillis(System.currentTimeMillis());
       Date date = calendar.getTime();
       cal_hour = date.getHours();
       cal_minute = date.getMinutes();

      final Button alarmOpenBtn = (Button)findViewById(R.id.open);
      final Button alarmCloseBtn = (Button)findViewById(R.id.close);
      final Button alarmCancelBtn = (Button)findViewById(R.id.cancel);


        // 保存された時刻を取得
        prefs = PreferenceManager.getDefaultSharedPreferences(this);
        prefs.registerOnSharedPreferenceChangeListener(this);
        getSharedPreferences();

        // TimePicker に反映
       timePicker = (TimePicker)findViewById(R.id.timePicker1);
       timePicker.setIs24HourView(true);
       timePicker.setCurrentHour(cal_hour);
       timePicker.setCurrentMinute(cal_minute);


//タイムピッカーを 1,2 とすれば,閉まるタイマーと開けるタイマーとできるんじゃね？
```

```
        alarmOpenBtn.setOnClickListener( new Button.OnClickListener() {
                public void onClick(View arg0) {
                    // TODO 自動生成されたメソッド・スタブ
                    Open = true;
                    alarmOpenBtn.setEnabled(false);
                    alarmCancelBtn.setEnabled(true);
                    // アラームの設定
                    startAlarm();
                }
        });

        alarmCloseBtn.setOnClickListener( new Button.OnClickListener() {
            public void onClick(View arg0) {
                // TODO 自動生成されたメソッド・スタブ
                Close = true;
                alarmCloseBtn.setEnabled(false);
                alarmCancelBtn.setEnabled(true);
                // アラームの設定
                startAlarm();
            }
    });

        alarmCancelBtn.setOnClickListener( new Button.OnClickListener() {
                public void onClick(View v) {
                    // TODO 自動生成されたメソッド・スタブ
                    alarmOpenBtn.setEnabled(true);
                    alarmCloseBtn.setEnabled(true);
                    // アラームの解除
                    stopAlarm();
                }
        });
}

public void startAlarm() {
        Log.d("AlarmTestActivity", "startAlarm()");
```

```
        alarmManager                                                    =
(AlarmManager)getSystemService(Context.ALARM_SERVICE);
        // 念のため現在の年月日と取得している時刻をカレンダーに設定
        calendar = Calendar.getInstance();
        calendar.setTimeInMillis(System.currentTimeMillis());
        Date date = calendar.getTime();
        date.setHours(cal_hour);
        date.setMinutes(cal_minute);
        calendar.setTimeInMillis(System.currentTimeMillis());

        // TimePicker で選択された時刻を取得
        timePicker = (TimePicker)findViewById(R.id.timePicker1);
        cal_hour = timePicker.getCurrentHour();
        cal_minute = timePicker.getCurrentMinute();

        // 取得した時刻をカレンダーに設定
        calendar.set(Calendar.HOUR_OF_DAY, cal_hour);
        calendar.set(Calendar.MINUTE, cal_minute );
        calendar.set(Calendar.SECOND, 0);
        calendar.set(Calendar.MILLISECOND, 0);

        // アラームに登録
        alarmManager.set(AlarmManager.RTC_WAKEUP,
                         calendar.getTimeInMillis(),
                         getPendingIntent());

        // 時刻を保存
        SharedPreferences.Editor editor = prefs.edit();
        editor.putInt( "cal_hour", cal_hour );
        editor.putInt( "cal_minute", cal_minute );
        editor.commit();
    }

    public void stopAlarm() {
        Log.d("AlarmTestActivity", "stopAlarm()");
```

```
            // 登録してあるアラームを解除
            alarmManager =
(AlarmManager)getSystemService(Context.ALARM_SERVICE);
            alarmManager.cancel(getPendingIntent());
    }

    private PendingIntent getPendingIntent() {
            // 起動するアプリケーションを登録

            if(Open && !Close){
           Open1 = new Intent( getApplicationContext(), Timer_Open.class );
           Open2 = PendingIntent.getActivity(this, 0, Open1,
                    PendingIntent.FLAG_UPDATE_CURRENT );
             result = Open2;
             Open = false;
            }
            if(Close && !Open){
                    Close1 = new Intent( getApplicationContext(), Timer_Close.class );
                Close2 = PendingIntent.getActivity(this, 0, Close1,
           PendingIntent.FLAG_UPDATE_CURRENT );
                    result = Close2;
                    Close = false;
            }
            return result;

    }


     public void onSharedPreferenceChanged(SharedPreferences arg0, String arg1) {
            // TODO 自動生成されたメソッド・スタブ
            getSharedPreferences();
     }

    private void getSharedPreferences() {
            // 保存されていた時刻を取得
           cal_hour = prefs.getInt("cal_hour", cal_hour);
```

```
            cal_minute = prefs.getInt("cal_minute", cal_minute);
        }
    }
```

Timer_open.java

```
package com.doroidcurtain;

import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;


import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.widget.CompoundButton.OnCheckedChangeListener;

public class Timer_Open extends Activity implements Runnable, OnClickListener{

        /**
         * TAG
         */
        private static final String TAG = "BT";

        /**
         * Bluetooth Adapter
         */
        private BluetoothAdapter mAdapter;
```

```
    /**
     * Bluetooth Devices
     */
    private BluetoothDevice mDevice;

    /**
     * Bluetooth UUID
     */
    private final UUID MY_UUID =
UUID.fromString( "00001101-0000-1000-8000-00805F9B34FB");

    /**
     * Device Name
     */
    private final String DEVICE_NAME = "FireFly-33B8";

    /**
     * Socket
     */
    private BluetoothSocket mSocket;

    /**
     * Thread
     */
    private Thread mThread;

    /**
     * Thread の状態を表す
     */
    private boolean isRunning;

    /**
     * Button
     */
    private Button mButton;
```

```
/**
 * Button
 */
private Button onButton;

/**
 * Button
 */
private Button offButton;

/**
 * Context
 */
private Context mContext;


/**
 * OnOff Flag
 */
private boolean onOff = false;

private String commandId = "1"; // デフォルトは静止だが正転に今回はした

@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.timer_open);

        mButton = (Button) findViewById(R.id.stop);
        mButton.setOnClickListener(this);

        mButton.setOnClickListener(new OnClickListener(){
        public void onClick(View mView) {

                if (mView.equals(mButton)) {
```

```
                                        //onOff = false;
                                        commandId = "0";
                                        Log.d(TAG,"mButton pressed");

                        }
                                }
                        });
        }
   public void onClick(View v) {
                                // TODO  自動生成されたメソッド・スタブ

                        }
{
                        // Bluetooth のデバイス名を取得
                        // デバイス名は,FireFly-BEXX になるため,XX の数字はデバイス毎に異
                        なる.
                        // DVICE_NAME でデバイス名を定義
                        mAdapter = BluetoothAdapter.getDefaultAdapter();
                        Set< BluetoothDevice > devices = mAdapter.getBondedDevices();
                        for ( BluetoothDevice device : devices ){
                                Log.i(TAG,"DEVICE:"+device.getName());
                                if(device.getName().equals(DEVICE_NAME)){
                                        mDevice = device;
                                }
                        }
                        // Thread を起動し,Bluetooth 接続
                        mThread = new Thread(this);
                        isRunning = true;
                        mThread.start();
        }

        @Override
        protected void onPause(){
                        super.onPause();

                        isRunning = false;
```

```
            try{
                    mSocket.close();
            }
            catch(Exception e){}
}

public void run() {
            InputStream mmInStream = null;
            OutputStream mmOutputStream = null;
            try {
    // 取得したデバイス名を使って Bluetooth で Socket 接続
            mSocket= mDevice.createRfcommSocketToServiceRecord(MY_UUID);
                    mSocket.connect();
                    mmInStream = mSocket.getInputStream();
                    mmOutputStream = mSocket.getOutputStream();

            // InputStream のバッファを格納
                    byte[] buffer = new byte[1024];
            // 取得したバッファのサイズを格納
                    int bytes;

                    while(isRunning){

                    Log.d(TAG,"sending data");

                    mmOutputStream.write(commandId.getBytes());

             // InputStream の読み込み
                    bytes = mmInStream.read(buffer);

           // String 型に変換
                    String readMsg = new String(buffer, 0, bytes);

           // null 以外なら表示
                    if(readMsg.trim() != null && !readMsg.trim().equals("")){
                                Log.i(TAG,"value="+readMsg.trim());
```

```
                    }else{
                // データがおかしいときはもう一度読む
                    bytes = mmInStream.read(buffer);
                // String 型に変換
                    readMsg = new String(buffer, 0, bytes);
                    if(readMsg.trim() != null && !readMsg.trim().equals("")){
                    Log.i(TAG,"value="+readMsg.trim());
                                }
                            }
                        }
                    }catch(Exception e){
                        Log.e(TAG,"error:"+e);
                        try{
                            mSocket.close();
                        }catch(Exception ee){}
                        isRunning = false;
                    }

}
}
```

Timer_close.java

```
package com.doroidcurtain;

import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;


import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.widget.CompoundButton.OnCheckedChangeListener;

public class Timer_Close extends Activity implements Runnable, OnClickListener{

	/**
	 * TAG
	 */
	private static final String TAG = "BT";

	/**
	 * Bluetooth Adapter
	 */
	private BluetoothAdapter mAdapter;
```

```
    /**
     * Bluetooth Devices
     */
    private BluetoothDevice mDevice;

    /**
     * Bluetooth UUID
     */
    private final UUID MY_UUID =
UUID.fromString( "00001101-0000-1000-8000-00805F9B34FB");

    /**
     * Device Name
     */
    private final String DEVICE_NAME = "FireFly-33B8";

    /**
     * Socket
     */
    private BluetoothSocket mSocket;

    /**
     * Thread
     */
    private Thread mThread;

    /**
     * Thread の状態を表す
     */
    private boolean isRunning;

    /**
     * Button
     */
    private Button mButton;
```

```
/**
 * Button
 */
private Button onButton;

/**
 * Button
 */
private Button offButton;

/**
 * Context
 */
private Context mContext;


/**
 * OnOff Flag
 */
private boolean onOff = false;

private String commandId = "2"; // デフォルトは静止だが反転に今回はした

@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.timer_close);

        mButton = (Button) findViewById(R.id.stop);
        mButton.setOnClickListener(this);

        mButton.setOnClickListener(new OnClickListener(){
 public void onClick(View mView) {
 if (mView.equals(mButton)) {
                        //onOff = false;
```

```
                                commandId = "0";
                                Log.d(TAG,"mButton pressed");


                        }
                                }
                        });
                }

                public void onClick(View v) {
                                // TODO 自動生成されたメソッド・スタブ

                        }
{
                        // Bluetooth のデバイス名を取得
                        // デバイス名は,FireFly-BEXX になるため,XX の数字はデバイス毎に異
なる.
                        // DVICE_NAME でデバイス名を定義
                        mAdapter = BluetoothAdapter.getDefaultAdapter();
                        Set< BluetoothDevice > devices = mAdapter.getBondedDevices();
                        for ( BluetoothDevice device : devices ){
                                Log.i(TAG,"DEVICE:"+device.getName());
                                if(device.getName().equals(DEVICE_NAME)){
                                        mDevice = device;
                                }
                        }

                        // Thread を起動し,Bluetooth 接続
                        mThread = new Thread(this);
                        isRunning = true;
                        mThread.start();
                }

                @Override
                protected void onPause(){
                        super.onPause();
```

```
            isRunning = false;
            try{
                        mSocket.close();
            }
            catch(Exception e){}
}

public void run() {
            InputStream mmInStream = null;
            OutputStream mmOutputStream = null;
            try {
                        // 取得したデバイス名を使って Bluetooth で Socket 接続
mSocket = mDevice.createRfcommSocketToServiceRecord(MY_UUID);
                        mSocket.connect();
                        mmInStream = mSocket.getInputStream();
                        mmOutputStream = mSocket.getOutputStream();

                        // InputStream のバッファを格納
                        byte[] buffer = new byte[1024];
                        // 取得したバッファのサイズを格納
                        int bytes;

while(isRunning){

                                    Log.d(TAG,"sending data");

                                    mmOutputStream.write(commandId.getBytes());

                                    // InputStream の読み込み
                                    bytes = mmInStream.read(buffer);

                                    // String 型に変換
String readMsg = new String(buffer, 0, bytes);

                        // null 以外なら表示
```

```
            if(readMsg.trim() != null && !readMsg.trim().equals("")){
                    Log.i(TAG,"value="+readMsg.trim());
                                    }else{
                     // データがおかしいときはもう一度読む
                                      bytes = mmInStream.read(buffer);
                                  // String 型に変換
                                    readMsg = new String(buffer, 0, bytes);
            if(readMsg.trim() != null && !readMsg.trim().equals("")){
                     Log.i(TAG,"value="+readMsg.trim());
                                                    }
                                    }
                                }


                    }catch(Exception e){
                            Log.e(TAG,"error:"+e);
                            try{
                                    mSocket.close();
                            }catch(Exception ee){}
                            isRunning = false;
                    }

}
}
```

音声認識

```
package com.doroidcurtain;

import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Set;
import java.util.UUID;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.content.Context;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.media.AudioManager;
import android.media.ToneGenerator;
import android.util.Log;
import android.view.View.OnClickListener;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;



public class Voice_Remote extends Activity implements Runnable, OnClickListener{

        /**
         * TAG
```

```
     */
    private static final String TAG = "BT";

    /**
     * Bluetooth Adapter
     */
    private BluetoothAdapter mAdapter;

    /**
     * Bluetooth Devices
     */
    private BluetoothDevice mDevice;

    /**
     * Bluetooth UUID
     */
    private final UUID MY_UUID =
UUID.fromString( "00001101-0000-1000-8000-00805F9B34FB");

    /**
     * Device Name
     */
    private final String DEVICE_NAME = "FireFly-33B8";

    /**
     * Socket
     */
    private BluetoothSocket mSocket;

    /**
     * Thread
     */
    private Thread mThread;

    /**
     * Thread の状態を表す
```

```
 */
private boolean isRunning;

/**
 * Button
 */
private Button mButton;

/**
 * Button
 */
private Button onButton;

/**
 * Button
 */
private Button vButton;

/**
 * Button
 */
private Button offButton;

/**
 * Context
 */
private Context mContext;


/**
 * OnOff Flag
 */
private boolean onOff = false;

private String commandId = "0"; // デフォルトは静止
```

```
ArrayList<String> Open = new ArrayList<String>();
ArrayList<String> Close = new ArrayList<String>();
ArrayList<String> Stop = new ArrayList<String>();

// = 0 の部分は,適当な値に変更してください（とりあえず試すには問題ないですが）
private static final int REQUEST_CODE = 0;
private void activateRecognize() {
}
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.voice_remote);
    Open.add("オープン");
    Open.add("ひらけ");
    Open.add("開け");
    Close.add("クローズ");
    Close.add("とじろ");
    Close.add("閉じろ");
    Stop.add("ストップ");
    Stop.add("とまれ");
    Stop.add("止まれ");

    // Bluetooth のデバイス名を取得
    // デバイス名は,FireFly-BEXX になるため,XX の数字はデバイス毎に異なる.
    // DVICE_NAME でデバイス名を定義
    mAdapter = BluetoothAdapter.getDefaultAdapter();
        Set< BluetoothDevice > devices = mAdapter.getBondedDevices();
        for ( BluetoothDevice device : devices ){
            Log.i(TAG,"DEVICE:"+device.getName());
            if(device.getName().equals(DEVICE_NAME)){
                mDevice = device;
            }
        }
        // Thread を起動し,Bluetooth 接続
        mThread = new Thread(this);
```

```
            isRunning = true;
            mThread.start();

            onButton = (Button) findViewById(R.id.open);
            onButton.setOnClickListener(this);

            offButton = (Button) findViewById(R.id.close);
            offButton.setOnClickListener(this);

            mButton = (Button) findViewById(R.id.stop);
            mButton.setOnClickListener(this);

 vButton = (Button) findViewById(R.id.voice);

 vButton.setOnClickListener(new View.OnClickListener(){
      public void onClick(View v) {
            try {
                  // インテント作成
                  Intent intent = new Intent(
RecognizerIntent.ACTION_RECOGNIZE_SPEECH);//ACTION_WEB_SEARCH
                  intent.putExtra(
                              RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                              RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
                  intent.putExtra(
                              RecognizerIntent.EXTRA_PROMPT,
                              "うっうー！"); // お好きな文字に変更できます

                  // インテント発行
                  startActivityForResult(intent, REQUEST_CODE);
            } catch (ActivityNotFoundException e) {
// このインテントに応答できるアクティビティがインストールされていない場合
                  Toast.makeText(Voice_Remote.this,
                 "ActivityNotFoundException", Toast.LENGTH_LONG).show();
            }
      }
 });
```

```
mButton.setOnClickListener(new OnClickListener(){
        public void onClick(View mView) {

                if (mView.equals(mButton)) {
                        //onOff = false;
                        commandId = "0";
                                Log.d(TAG,"mButton pressed");
                        onButton.setEnabled(true);
                        offButton.setEnabled(true);

        }
                }
        });

onButton.setOnClickListener(new OnClickListener(){
public void onClick(View mView) {

                        if (mView.equals(onButton)) {
                        //onOff = true;
                        commandId = "1";
                        Log.d(TAG,"onButton pressed");

                        onButton.setEnabled(false);
                        offButton.setEnabled(true);

            }
                                                }
                });

offButton.setOnClickListener(new OnClickListener(){
                public void onClick(View mView) {

                if (mView.equals(offButton)) {
                        //onOff = false;
                        commandId = "2";
                        Log.d(TAG,"offButton pressed");
```

```
if (offButton.getVisibility() == View.VISIBLE || onButton.getVisibility() ==
    View.INVISIBLE){
                        offButton.setEnabled(false);
                        onButton.setEnabled(true);
                            }
                                }
                    }
                                });
            }

            protected void onStop(){
                    super.onStop();

                    isRunning = false;
                    try{
                            mSocket.close();
                    }
                    catch(Exception e){}
            }

public void run() {
                    InputStream mmInStream = null;
                    OutputStream mmOutputStream = null;
                    try {
            // 取得したデバイス名を使って Bluetooth で Socket 接続
            mSocket = mDevice.createRfcommSocketToServiceRecord(MY_UUID);
                            mSocket.connect();
                            mmInStream = mSocket.getInputStream();
                            mmOutputStream = mSocket.getOutputStream();

                            // InputStream のバッファを格納
                            byte[] buffer = new byte[1024];
                            // 取得したバッファのサイズを格納
                            int bytes;
```

```
                    while(isRunning){

                        Log.d(TAG,"sending data");
                    mmOutputStream.write(commandId.getBytes());
                            // InputStream の読み込み
                        bytes = mmInStream.read(buffer);
                            // String 型に変換
                    String readMsg = new String(buffer, 0, bytes);
                        // null 以外なら表示
                    if(readMsg.trim() != null && !readMsg.trim().equals("")){
                        Log.i(TAG,"value="+readMsg.trim());
                                    }else{
                    // データがおかしいときはもう一度読む
                        bytes = mmInStream.read(buffer);
                        // String 型に変換
                        readMsg = new String(buffer, 0, bytes);
                    if(readMsg.trim() != null && !readMsg.trim().equals("")){

                Log.i(TAG,"value="+readMsg.trim());
                                                }
                                    }
                            }
                        }catch(Exception e){
                            Log.e(TAG,"error:"+e);
                            try{
                                    mSocket.close();
                            }catch(Exception ee){}
                            isRunning = false;
                        }
                    }
                    // アクティビティ終了時に呼び出される
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // 自分が投げたインテントであれば応答する
    if (requestCode == REQUEST_CODE && resultCode == RESULT_OK) {
        String resultsString = "";
```

```
        // 結果文字列リスト
        ArrayList<String> results = data.getStringArrayListExtra(
                RecognizerIntent.EXTRA_RESULTS);

      for (int i = 0; i< results.size(); i++) {
            // ここでは,文字列が複数あった場合に結合しています
            resultsString += results.get(i);
}
            if(isOpen(resultsString)){
            commandId = "1";
                    Log.d(TAG,"onButton pressed");
            Toast.makeText(this, "開きます", Toast.LENGTH_LONG).show();
            activateRecognize();

        }else if(isClose(resultsString)){
            commandId = "2";
                    Log.d(TAG,"offButton pressed");
            Toast.makeText(this, "閉まります", Toast.LENGTH_LONG).show();
            activateRecognize();

            }else if(isStop(resultsString)){
            commandId = "0";
                    Log.d(TAG,"mButton pressed");
            Toast.makeText(this, "停止します", Toast.LENGTH_LONG).show();
            activateRecognize();

            }else{
                    Log.d(TAG,"once again");
            Toast.makeText(this, "もう一度", Toast.LENGTH_LONG).show();
            activateRecognize();
            }

                    }

   super.onActivityResult(requestCode, resultCode, data);
```

```
    }
        public void onClick(View v) {
                // TODO 自動生成されたメソッド・スタブ

        }
          //音声認識で帰ってきた文字列を,事前に用意しておいた配列と比較するような
            //プログラム
         public boolean isOpen(String string){
              for(int i=0; i< Open.size(); i++){
                if(string.indexOf( Open.get(i)) != -1){
                  return true;
                }
              }
              return false;
            }

         public boolean isClose(String string){
              for(int i=0; i< Close.size(); i++){
                if(string.indexOf( Close.get(i)) != -1){
                  return true;
                }
              }
              return false;
            }

         public boolean isStop(String string){
              for(int i=0; i< Stop.size(); i++){
                if(string.indexOf( Stop.get(i)) != -1){
                  return true;
                }
              }
              return false;
            }
}
```

2.2　ライブラリ

ライブラリには twitter4j というライブラリを使用している.これはインターネットから無料で入手可能である.

2.3

Icon 一覧

Home.4.png

Ic_launcher_droidcurtain.png

timer2.png

Remote2.png

setting2.png

sns2.png

参考文献・URL

1. Google アンドロイド Android プログラミング入門. 出版地不明 ： 株式会社豆蔵.
2. android スマートフォンのホーム画面紹介.
http://d.hatena.ne.jp/itokoichi/20091204/p2.
3. Android で端末本体内のアプリを検索する方法.
http://appnavi.sonymobile.co.jp/pc/ag/index.php?page=powerpush&id=1869.
4. NAVER まとめ　android の主な特徴・仕様.
http://matome.naver.jp/odai/2129663748469157101.
5. はてなキーワード>アプリケーション.
http://d.hatena.ne.jp/keyword/%A5%A2%A5%D7%A5%EA%A5%B1%A1%BC%A5%B7%A5%E7%A5%F3.
6. スマホ所有率は 23.6%,android が iOS を逆転―インターネットコムより.
http://japan.internet.com/allnet/20120424/4.html.
7. HTC J Butterfly HTL21 初期インストール済みアプリケーション一覧.
http://sceneryandfish.withnotes.net/?p=1110.
8. iPhone と android のちょっとした大きな違い.
http://www.anlyznews.com/2010/06/iphoneandroid.html.
9. スマートフォンが抱えるセキュリティリスクとその対策.
http://trendy.nikkeibp.co.jp/article/column/20111114/1038628/.