専攻分野の名称	工学
専攻の区分	電気電子工学

テーマ名：

手袋型デバイスを用いた自然的インターフェースの研究

氏名：高田　崚介

目 次

第1章　はじめに

現在,携帯用通信端末としてスマートフォンが普及しており,それに伴って様々な操作手法が提案されている. その中で,主に用いられているのはタッチパネルを用いたタッチジェスチャーと,マイクを用いた音声操作である. しかし,タッチパネルや音声操作を用いる手法には欠点が存在する. 例えば液晶一体型のタッチパネルを用いるには,端末を手に持って小さな画面を見ながら操作しなくてはならない. このことから,外出先で歩行している際などの運用は事故につながり易く危険である. 実際に昨年5月,スマートフォンを操作していた小学5年生の男児が誤ってホームに転落し,怪我をするという事故なども起こっている[1]. また音声操作は,周りの雑音が多い場所や音声を発するのが適切ではない場面など,使用が難しい場所が容易に想定される.

そこで,本研究ではこれらの問題を解決する入力用インターフェースとしてデータグローブに着目した. データグローブとは,曲げセンサや慣性計測用センサを搭載した,手勢検出用手袋のことであり,1980年代に「エアギター」を実現するために開発されてから,主にバーチャルリアリティ（仮想現実感）の分野での入力用インターフェースとして注目される様になったデバイスである[2]. 本研究では,このデータグローブを用いて指文字などのハンドジェスチャーを検出し,スマートフォンの操作に用いることでタッチパネルや音声操作などで存在する問題を解決する.

本研究では実際にデータグローブを作製した. 作製したデータグローブでは,曲げセンサを用いて指の曲げを測定し,慣性計測用センサを用いて手の回転や移動を検出し,Bluetooth通信を用いてスマートフォンにデータを送信した. スマートフォン側では受信したデータを基にハンドジェスチャーの検出を行った. デバイスを作製した結果,逐次的に指文字の「あ」「い」「う」「え」「お」や他のジェスチャーの検出を行うことができた. 本論文ではまず,第2章でシステム全体の構成や検出するジェスチャー,検出手法について,次に,第3章では作製したデータグローブの構成,検出の手法について解説し,第4章で今回の研究についてまとめる.

第2章　システム

2.1　システムの構成

2.1.1　ジェスチャー検出システムの全体構成

図2.1にデータグローブとスマートフォンを用いたジェスチャー検出システムの全体の構成を示す. データグローブには, 指の曲げ具合を検出するための抵抗式の曲げセンサを5指分, 手の回転や移動を検出するためのジャイロ・加速度センサを搭載した慣性計測用センサが1つ, また, それらのセンサの値を処理するためのコンピュータとして, Arduino Pro Microを1台搭載している. また, これらのセンサで検出したデータをスマートフォンに送信するモジュールとして,Bluetooth通信モジュールを1台搭載している. スマートフォンはAndroid OS搭載のものを採用した.

次に簡単な動作説明をする. まず, ユーザーはデータグローブを手に装着し, データグローブ上に搭載されたセンサを用いてユーザーの手勢を検出する. 検出したセンサの値を一度, データグローブ上のマイコンで処理し,Bluetooth通信を用いてスマートフォンへと送信する. スマートフォンでは, 受信したデータを解析し, ジェスチャーの判別を行う. また, 判別したジェスチャーに基づくアプリケーションの制御も併せて行うことで, ユーザーは任意のジェスチャー入力によるアプリケーションの制御を行うことができる.

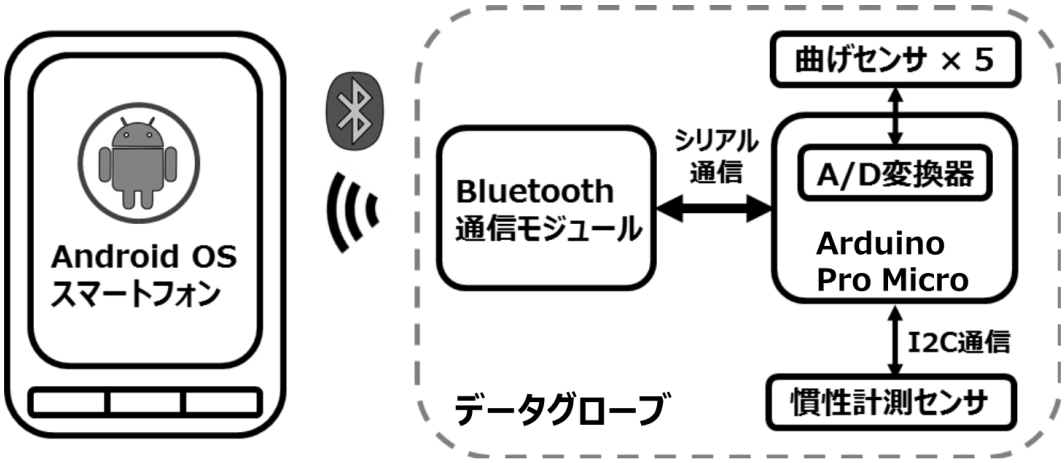


図2.1 ジェスチャー検出システムの全体構成

2.2　検出するジェスチャーについて

2.2.1　指文字

指文字とは, 片手のみで言語を表現することができる視覚言語の一つである [3]. 50 音を表現することができるので, 主に手話の補助として用いられることが多い. また, 手話と違い地域ごとにおけるいわゆる方言が無く, 統一されたジェスチャーを用いるため広く使用されているという特徴がある. 図 2.2 に 50 音の内の「あ」「い」「う」「え」「お」の例を示す.

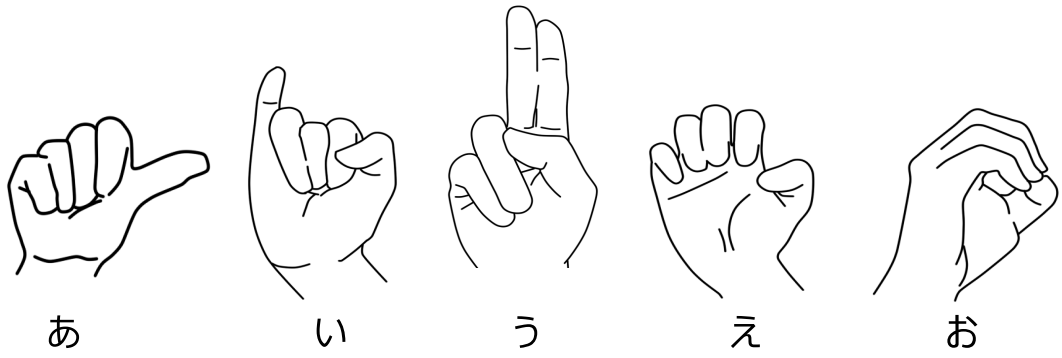


図 2.2 指文字の例

本研究では, 片手のみの手袋型デバイスで検出できるジェスチャーとして指文字の検出を行った.

指文字の多くは指の曲げのみで表現することができるジェスチャーであるが, 一部の文字では手の回転や移動などを用いる必要がある. 例えば図 2.3 の様に, 濁点を表現する際は, 清音の形を保ったまま手を右へ移動しなければならない. これらのジェスチャーにおいては指の曲げに加えて, 手の回転や移動などの手勢の検出が必要となる. 本研究では, 曲げセンサのみで検出できない指文字を, 慣性計測用センサを用いて検出することとする.

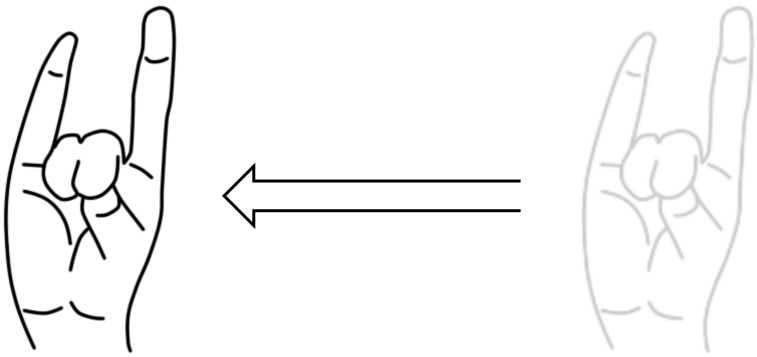


図 2.3 指文字”ぎ”の例

2.3 指の曲げの検出手法

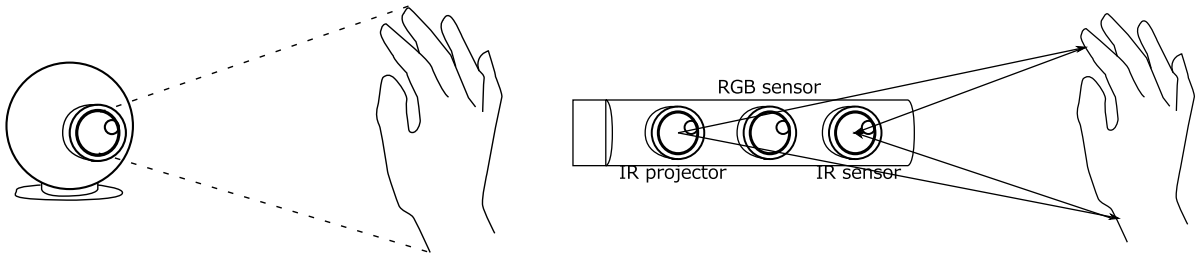
手勢の検出を行う方法として大別すると画像処理を用いる手法と,センサを用いる手法の2つが挙げられる.本研究ではひずみゲージを用いた手勢の検出を行った.以下に,それぞれの手法の検討と,ひずみゲージを選択した理由について記載する.

2.3.1 画像処理を用いた手法

この手法ではカメラセンサから送られてきたイメージを基に,人体の手の動きや指の動きを検出する.一般的に特徴抽出やパターンマッチングを行い,手や指がどのような状態にあるかを検出し,それらの時間的変化を元にジェスチャー検出を行う手法である.一台のカメラのみで検出することができる利点があるが,精度を上げるのが難しく,計算処理に時間がかかり易いといった欠点がある.また,カメラと手の間に一定以上の距離を設ける必要があり,例えば外出時に用いることを想定すると,カメラを取り付ける位置が問題となり,その解決手法として首からネックストラップを用いて吊り下げて使用する手法が挙げられる[4].しかし,カメラの視野角に収まるように手を前方に挙げる必要があり,常に使用するのは難しい.

2.3.2 3D深度センサを用いた手法

3D深度センサとは,単純なRGB入力だけではなくセンサから被写体までの距離も測定するセンサのことである.3D深度センサとしては,Microsoft社が提供する「Kinect」や,Primesense社が提供する「Xtion」が有名である.画像処理を用いた手法に比べて,精度が高いという利点があるが,一般的に3D深度センサは2台以上のイメージセンサを間隔を開けて配置する必要があるため,通常のカメラセンサに比べて大型になるが,肩の上に載せて利用する手法がある[5].また,コストも高い.



[1] RGBカメラを用いる手法　　[2] 3D深度センサを用いる手法

図2.4 画像処理を用いて検出する手法

2.3.3 機械的手法

スライド式可変抵抗や, 回転式可変抵抗とワイヤやばねを用いて指の曲げに応じて抵抗値が変化する機構を作製し, それを用いる手法である. 指の先端から伸ばしたワイヤをロータリエンコーダが付いた滑車機構に接続し, 指の曲げ伸ばしを行った際の回転数を計測する手法もある. 一般的にコストが安くてすむ利点があるが, 機構が大型化しやすい欠点がある.

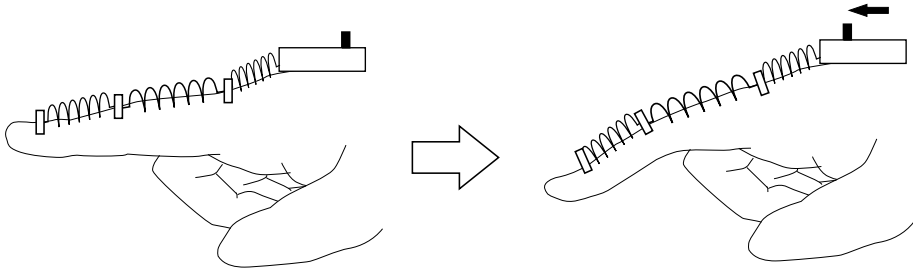


図 2.5 機械的機構を用いた手法

2.3.4 光ファイバを用いた手法

手袋の指に沿って配した光ファイバの光透過率が, 指を曲げることにより変化することを利用して, ファイバの一端から LED(発光ダイオード) で入力した光を, 他端に配したフォトトランジスタで測定し, その光量から指の曲げ具合を測定するという手法である.

この際, 光透過率を任意に設定するために微小なスリットを光ファイバに入れる場合が多い. 他の物理的センサと違い, 指の曲げだけでなく, 指の開きの検出まで行うことができるが, 一般的にコストが高い.

また, あまり実用化されておらず, センサが市販されていない.

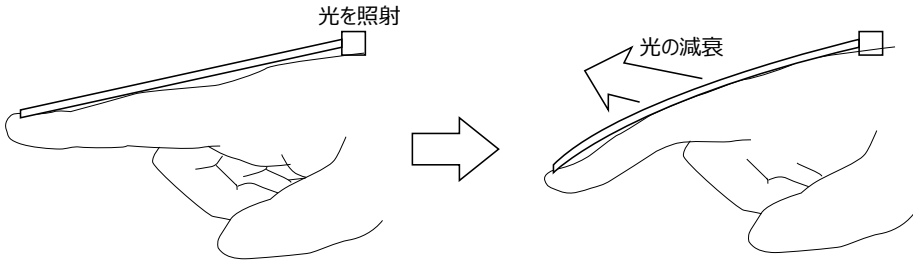


図 2.6 光ファイバを用いた手法

2.3.5　ひずみゲージを用いた手法

図 2.7 に用いた指の曲げ伸ばし時のひずみゲージの差異について示す. 手袋の指に沿ってひずみゲージを貼り付け, 指の曲げに応じて変化する抵抗値を読み取る手法である. 一般的に検出が容易で, 比較的安価なため最もよく用いられる手法である.

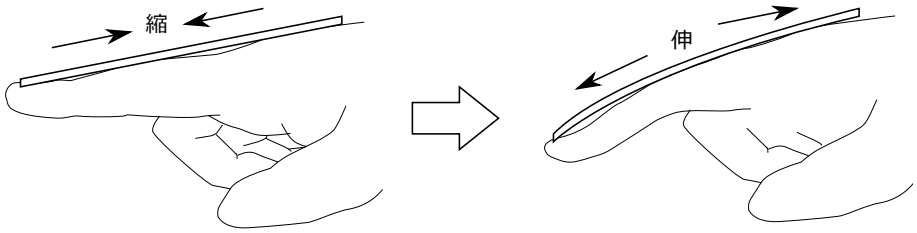


図 2.7 曲げセンサを用いた手法

曲げ具合の検出には図 2.8 のような分圧回路を用いる.

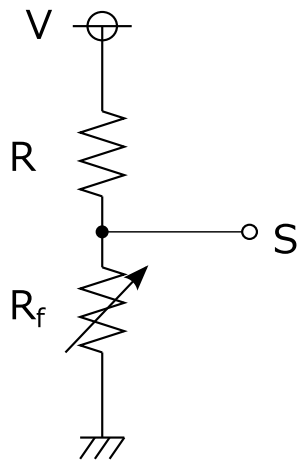


図 2.8 ひずみゲージを用いたセンサ回路

この時, センサの端子 S には以下の式で表される電位が生じる

$$V_S = \frac{R_f}{R + R_f} \times V \tag{2.1}$$

この分圧の式からわかるとおり,R_f の値が上昇すれば S の電圧が上昇する. この電位を計測することで曲げ度を検出する.

2.3.6 検出方法の比較

表 2.1 に, 今回挙げた指の曲げを検出する手法についてまとめた.

表 2.1 指の曲げを検出する手法の比較

検出手法	外出時における設置の容易性	コスト
画像処理を用いた手法	難	高
3D 深度センサを用いた手法	難	高
機械的手法	中 (大型化しやすい)	低
光ファイバを用いた手法	易	高
ひずみゲージを用いた手法	易	低

本研究では外出先で用いることを考え, 人体から一定以上離れた場所に置かなければならないカメラを用いた手法ではなく, 物理的センサを用いて直接指の曲げを検出する方法を選択した. また, 物理的センサを用いた手法の中でも最も扱いが容易でコストが低いひずみゲージを用いた手法を選択した.

第3章　システムの解説

3.1　データグローブの構成

図 3.1 に実際のデータグローブの写真と, その説明の図を示す. データグローブの Wifi カメラを含めた総重量は約 400g である. 装着したままでも静電容量式のタッチパネル端末を使用できるよう, 導電繊維を編みこんだ手袋を用いている.5 つの曲げセンサは 5 本の指に沿う様に面ファスナーを用いて貼り付けており, 調整のための脱着が可能になっている. また Bluetooth 通信モジュール, 慣性計測用センサ,Arduino などの各種モジュールは手の甲側に面ファスナーを用いて貼り付けてあるブレッドボード上に接続されており, それらの間はジャンパ線を用いて配線している. Wifi カメラとリチウムイオンバッテリー (1000mAh) は腕に巻いてあるアームバンド上に面ファスナーを用いて貼り付けてある. これは手袋側に設置すると, 手袋の重量が重くなり, ユーザーの動きを阻害してしまうためである. また, 人差し指と親指の指先の腹側には導電繊維を編みこんで作製したスイッチを取り付けてある, この二つの電極の導通を検出することで, 写真撮影ジェスチャーをとった際などのシャッター動作検出を補助している.

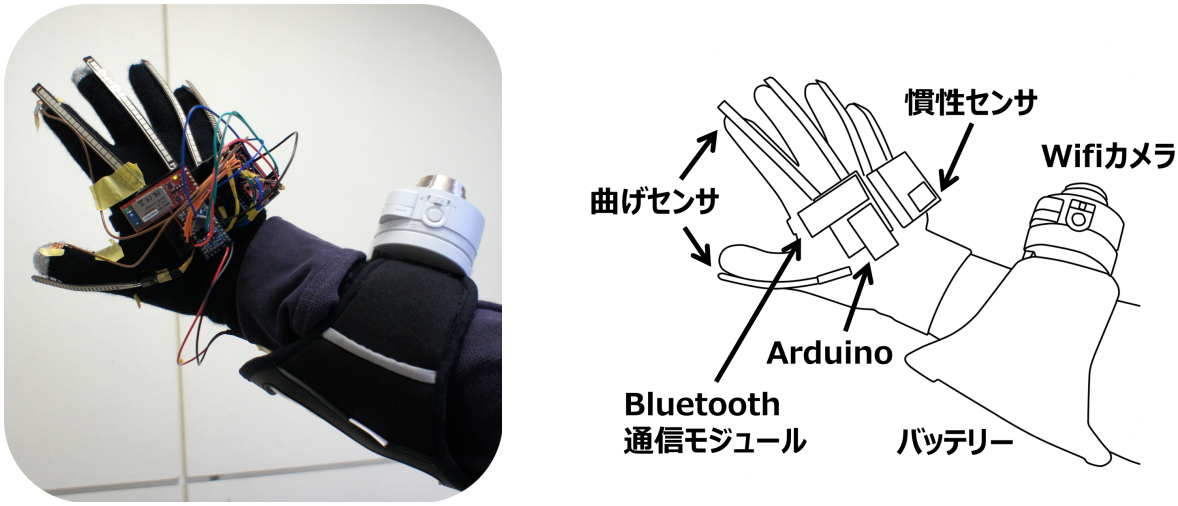


図 3.1 作製したデータグローブの写真とその配置

3.2 ハードウェアの構成

図 3.2 にハードウェア全体の構成を示す.

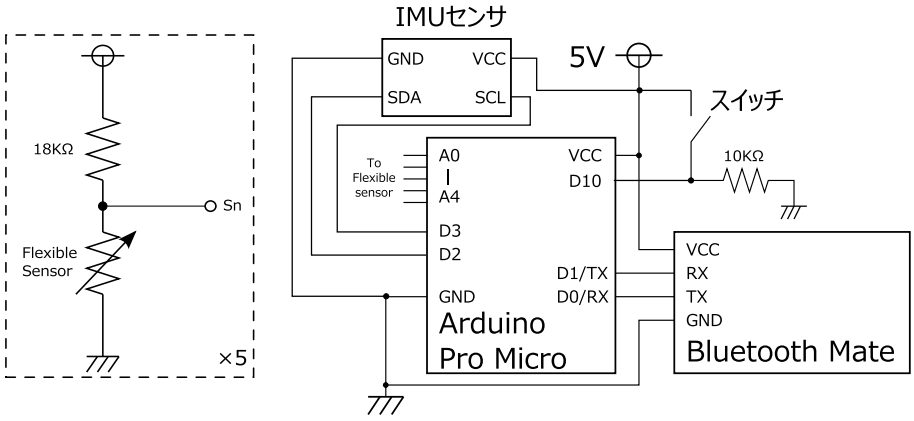


図 3.2 ハードウェアの構成

1) ひずみゲージ部

可変抵抗であるひずみゲージと $18k\Omega$ の抵抗の分圧器になっている.

ひずみゲージの抵抗値に比例して端子電圧が変化する.

2) 慣性計測用センサ

Sparkfun 製 IMU センサを使用. ジャイロセンサとして ITG-3200, 加速度センサとして ADXL345 を搭載している. 非常に小型なモジュールになっており, このセンサを用いてユーザーの手の回転や移動を検出する.

3) スイッチ

親指と人差し指に導電繊維を編みこんで作製した電極をスイッチとし, プルアップ回路を構成しており, 導通すると,Arduino 側の入力が HIGH になるようになっている.

4) Arduino Pro Micro

ひずみゲージ部の端子電圧を AD 変換している. このとき, 端子電圧が 0-5V を 0-1023 のデジタルデータに変換している.

慣性計測用センサとは I2C 通信を用いて接続している.

また,BluetoothMate とシリアル接続している.

5) BluetoothMate

Arduino Pro Micro とシリアル接続している. 内部に Bluetooth 通信モジュールが入っており,MAC アドレスも割り当てられているため, AndroidOS スマートフォンはこの素子と同期を行うことで Arduino からデータを受信することができるようになっている.

3.3 ソフトウェアの構成

3.3.1 データの送受信

図 3.3 に計測したデータをスマートフォンに送信する際の全体のフローチャートを示す. 以下に, データ送受信の際の全体の流れについて説明する.

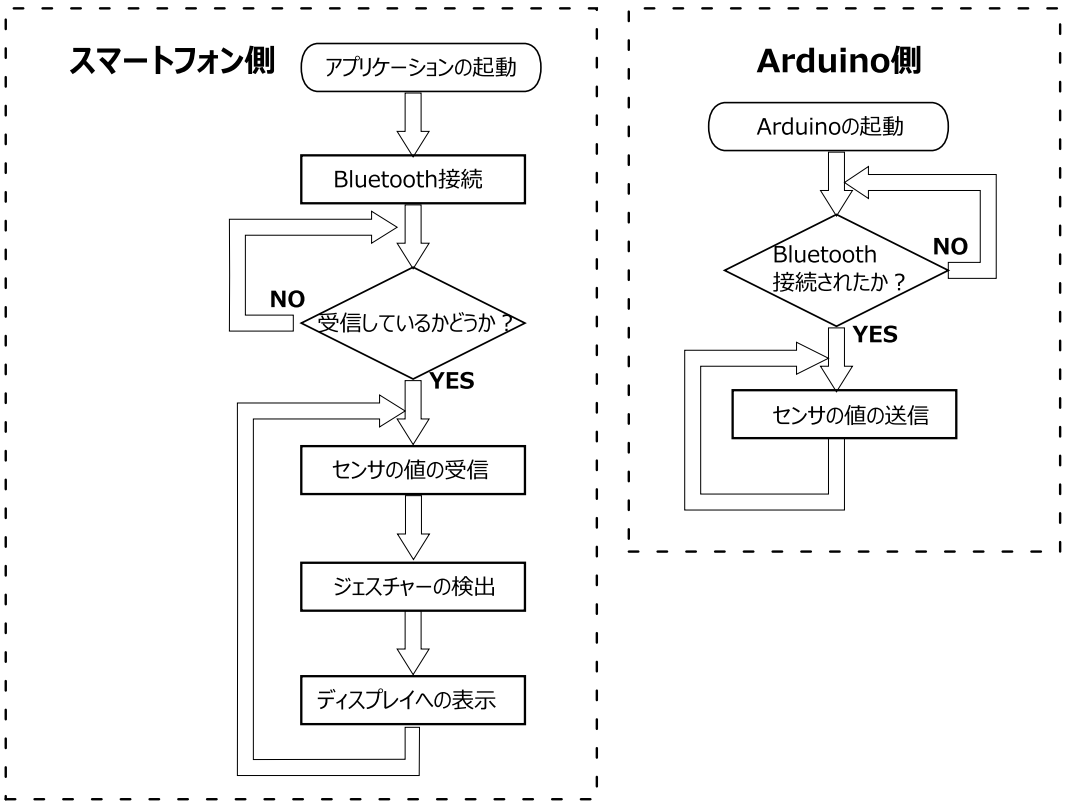


図 3.3 データ送受信のフローチャート

1) スマートフォン側でジェスチャー検出用のアプリケーションを起動する.

2) Arduino に電源を接続し, 給電を開始する.

3) スマートフォン側から Arduino 側へ Bluetooth 接続設定を行う.

4) Arduino 側は接続完了後は読み取ったセンサの値を,Bluetooth 通信を用いてスマートフォンへと送信し続ける.

5) スマートフォン側は受け取ったデータを元に, ジェスチャーの検出を行う.

以上の手順で, ジェスチャーの検出を行う.

主にジェスチャー検出のためにデータを元に計算を行うのはスマートフォン側で行い, Arduino はデータの送信用に用いている.

3.3.2 Arduino 側の処理

図 3.4 に Arduino 側のソフトウェアのフローチャートを示す.

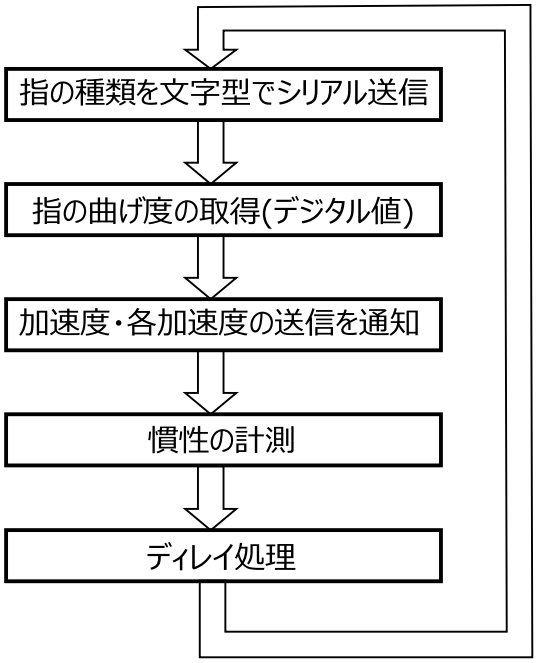


図 3.4 Arduino 側のデータ送信のフローチャート

本デバイスにおいて,Arduino はひずみゲージの出力値の AD 変換, 慣性計測用センサのデータの取得, スイッチの導通の検出, およびそれらの観測データの Bluetooth 通信を用いたスマートフォンへの送信にのみ用いている. AD 変換等は Arduino 内部に組み込まれた AD コンバータで自動的に行うため, ソフトウェア側での制御は不要である. この処理で, ひずみゲージの分圧回路によって出力値の 0-5V を 0 から 1023 の離散値に（以後 0-1023 と表記）AD 変換している. また, 慣性計測用センサと Arduino は I2C 通信でデータのやり取りをしている. Bluetooth 通信はシリアル通信用の SPP プロファイルを使用し, 通信レートは 115200bps を選択した. なお,Bluetooth 通信を実際に行うとスマートフォンが受信中である際に,Arduino が続きを送信しようとしてデータが崩れてしまうことがあるため, データ送信の間に 30mS ほどのディレイ処理を挟んである.

3.3.3　スマートフォン側の処理

図 3.5 にスマートフォン側のソフトウェアのフローチャートを示す.

ソフトウェア起動から, ジェスチャーを検出するまでの流れは大まかに図 3.5 の通りである.

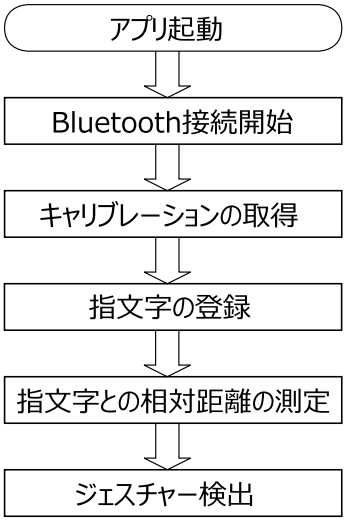


図 3.5 スマートフォン側ソフトウェアのフローチャート

以下に手順を示し, 解説する.

1) Bluetooth 接続

Bluetooth 接続では, データグローブに搭載された Bluetooth モジュールの MAC アドレスを指定して接続を行う.

2) キャリブレーションの取得

キャリブレーションの取得では, データグローブを装着したユーザーに 7 秒間, 指の曲げ伸ばしを行ってもらい, その際の指の曲げによる曲げ値の最大値と最小値の計測を行う. このデータを図 3.6 の線形変換を用いて指の曲げ値を 0-255 の 256 段階の離散値に変換することで, ユーザーの手の大小や環境によらない曲げ具合の検出が可能になる.

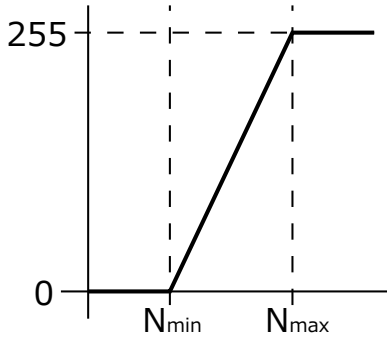


図 3.6 指の曲げ値線形変換

3) 指文字の登録

キャリブレーション取得後,ユーザーに指を曲げて指文字を作ってもらい,その際の曲げ値をスマートフォンに記憶させる.この登録されたデータを用いてジェスチャーの判別を行う.

4) 慣性計測

慣性計測用センサとI2C通信を用いて,加速度と角加速度を検出する.この加速度を積分することで,どの程度ユーザーが手を回転・移動させたかの検出を行う.また,加速度等の信号には誤差があり,それを積分して得られるデータのずれが常に広がるため,定期的にデータの初期化を行う.

5) ジェスチャー検出

線形変換した指の曲げ値と登録した指文字との間の距離を,各指の曲げ値の差を用いて,ユークリッド距離を計算する.

$$\text{距離} = \sqrt{\sum_{N=\text{親指}}^{\text{小指}} (\text{曲げ値}_N - \text{ジェスチャーの設定値}_N)} \tag{3.1}$$

この時の距離が一定値（しきい値）以下ならジェスチャーの検出を行う.

また,一部の指文字においてはこの曲げ値から求まる距離以外にも,現在デバイスが移動しているか,振られているか等を慣性計測用センサで取得した値から求めて,検出条件として用いる場合もある.

3.3.4　アプリケーションの UI

図 3.7 にスマートフォン側のジェスチャー検出用アプリケーションの UI を示す.

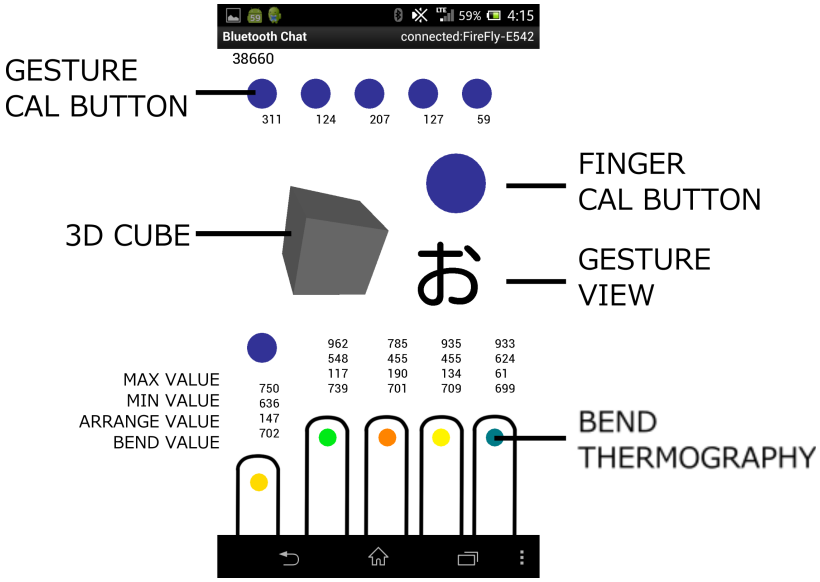


図 3.7 アプリケーション UI の解説

以下にアプリケーションの操作画面における各表示やボタンについての説明を述べる.

- GESTURE CAL BUTTON

  キャリブレーションの取得が終了した後, ジェスチャーをとった際の, 各指の曲げ値を登録するのに用いる.

  デフォルトでは 5 つのボタンは左から「あ」,「い」,「う」,「え」,「お」の指文字に対応したボタンとなっている. 本研究ではこの 5 つのジェスチャーの判別を行った.

- FINGER CAL BUTTON

  指の曲げ値の最大値と最小値を得るためのキャリブレーションボタンである. このボタンを押してから 7 秒間ボタンが緑色に変化するため, その間に手を握る・開くを繰り返し, 指の曲げ値の最大値と最小値を記憶させる.

- GESTURE VIEW

  現在, 検出されているジェスチャーを表示する.

- 3D CUBE

  現在, どの程度手が握られているか, 回転しているかを 3D モデルを用いて, 視覚的にわかりやすく表示している.

- VALUE
  - MAX VALUE

    FINGER CAL BUTTON を押して取得した際の指の曲げの最大値を表示する.
  - MIN VALUE

    FINGER CAL BUTTON を押して取得した際の指の曲げの最小値を表示する.
  - ARRANGE VALUE

    指の曲げ値を 0-1023 から 0-255 に線形変換した後のデータを表示する.
  - BEND VALUE

    Arduino から送られてきた 0-1023 のデータをそのまま表示する.
- BEND THERMOGRAPHY

  指の曲げ度 (ARRANGE VALUE) のデータを元に, 曲げの強度に応じた色を表示する. また色の設定はサーモグラフィを参考にした. 表示される色は図 3.8 のとおりである.

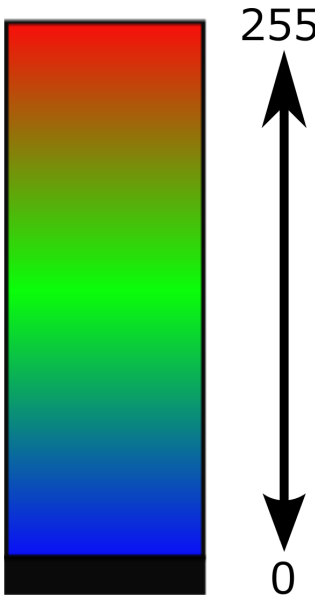


図 3.8 曲げ度に応じたサーモ色一覧

第4章　結果とまとめ

4.1　ジェスチャーの判別精度

GROVE システムの指文字検出の際の測定にかかる時間や, 精度について検証した.
検証方法を以下に示す.

1) ジェスチャー判別のしきい値の設定を行う.

2) キャリブレーションの取得を行う.

3) その後, 指文字を表現した際の各指の曲げ値を登録する.

4) 手をパーの状態に開いて, 各指の曲げ値が安定するまで待機する.

5) その後, 評価用に設定したボタンを押してから指文字を表現した.

6) ボタンを押した後, 指文字の判断が完了した時点で, ボタンを押してから経過した時間を表示し, 記録する. なお, この際はじめに検出された指文字が現在表現している指文字と違った場合は, ×と記載しカッコ内に検出された指文字を記入した.

7) (5)-(6) を各指文字 (「あ」,「い」,「う」,「え」,「お」) についてそれぞれ 5 回ずつ行った.

上記の手順をしきい値 70 と 100 の場合で, それぞれ (2)-(7) を 2 回ずつ行った.
なお, しきい値はジェスチャーと完全一致で 0, 最大で実測 300 程度の範囲を取る値の中から設定した値である. 結果を以下の表 4.1 に示す.

表 4.1 GROVE システムの検出精度

	しきい値 70		しきい値 100	
	正解率	応答時間 [秒]	正解率	応答時間 [秒]
あ	100%	2.14	100%	1.55
い	100%	3.04	100%	0.68
う	100%	1.30	100%	1.09
え	90%	2.46	90%	1.21
お	60%	2.07	50%	1.43

表 4.1 のデータから,「お」の検出精度が著しく低いことが分かる. これは, 手を開いた状態から「お」の手を閉じた状態に状態が遷移する過程で間違って「え」が検出されるためである. これを解決する手段として, 指の曲げ度の変化を微分することで傾きを検出し, 傾きが安定した所のみでジェスチャーの検出を行う等が考えられる.

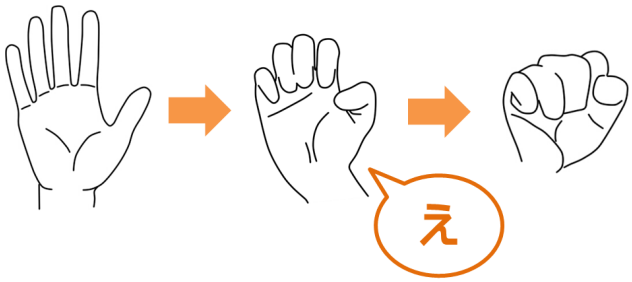


図 4.1 指文字「お」の誤検出の仕組み

4.2 まとめ

データグローブを作製し, 実際のジェスチャーの判別精度などの評価を行い, 得られた結果を以下に示す.

- 指の曲げの検出を行う手法について検討し, コストや目的の観点から, 抵抗式のひずみゲージを用いる手法を選択し, 指の曲げの検出, 手の回転や移動の検出が可能なデータグローブの作製に成功した.
- データグローブで検出し,Bluetooth 通信を用いてスマートフォンに送信したデータを基にジェスチャーの検出を行ったところ, 単純なユークリッド距離によるジェスチャーの判定のみでは, 似たようなジェスチャーにおいて誤検出が生じる. また, その際のジェスチャー検出にかかる時間は約 1〜2 秒程度であった.
- 今回の研究では, キャリブレーション時に取得した指の曲げ値の最大値と最小値を用いて線形変換を行った, 指の曲げ度合いのデータを用いたが, 指文字の「あ」,「い」,「う」等, 互いに似通っておらず, 簡単な指の曲げのみのジェスチャーであれば十分検出することができた. しかし, 指文字の「え」や「お」などの似通ったジェスチャーでは誤検出が多く, 問題となった. 今後, アルゴリズムを改良することで認識精度を高めていく予定である.
- デバイスには指文字の濁点などを検出するための慣性計測用センサを搭載している. 今後はこのセンサを用いて検出したデータを基に, さらに多くの種類のジェスチャーを検出する予定である.

参考文献

[1] 日本経済新聞:“小学生がホームから転落,あわや事故,JR 四ッ谷駅”, “http://www.nikkei.com/article/DGXNASDG27056_X20C13A5CC1000/”, 2013 年 5 月 28 日,2014 年 8 月 29 日閲覧

[2] 黒田　知宏, 田畑　慶人, 後藤　忠敏, 生田　祐樹, 對馬　英二 :
「手形を認識するデータグローブ StrinGlove」, 電子情報通信学会技術研究報告.PRMU, パターン認識・メディア理解 107(427),pp.123-124(2–8)

[3] 独立行政法人　国立特別支援教育総合研究所 : “手話の指導”, “http://www.nise.go.jp/cms/13.884.44.175.html”,2014 年 2 月 13 日閲覧

[4] Pranav Mistry,Pattie Maes: “SixthSense: a wearable gestural interface”, ACM SIGGRAPH ASIA 2009 Sketches Article No. 11(SIGGRAPH ASIA 2009)

[5] Chris Harrison,Hrvoje Benko,Andrew D.Wilson: “OmniTouch:Wearable Multitouch Interaction Everywhere”, UIST ’11 Proceedings of the 24th annual ACM symposium on User interface software and technology, pp.441-450(2011)