1 演習の目的

実験を通して、距離センサ GP2Y0A21YK0F の使い方と仕組みを習得することを目的とする。

2 演習の使用部品

2.1 図 1 の電子部品 (距離センサ GP2Y0A21YK0F) を次のような点から調べなさい。

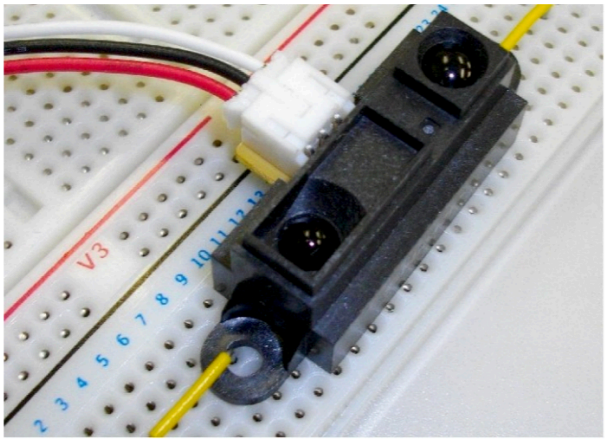

図 1: 距離センサ

2.1.1 どのような部品か

シャープの赤外線を使用した測距モジュールである。赤外線 LED と PSD(position sensitive detector)を使用して、非接触で距離を検出することができる[1] 。

2.1.2 どのような仕組みか

発光側の赤外光が物体に反射して受光すると距離に応じて出力電圧が変化する。この出力電圧で距離を検出することができる。測定距離を測る方法は PDS 方式を採用している。PSD 方式は三角測量の要領で距離を検出するもので 図 2 のように発光素子から出た光が対象物に当たって戻ってきた反射光を検出する。対象物が近いと PSD への光の入射角度は大きく、逆に遠いと入射角度は小さくなる。この角度の違いで出力電圧が変化するため、距離の情報を得ることができる[2] 。

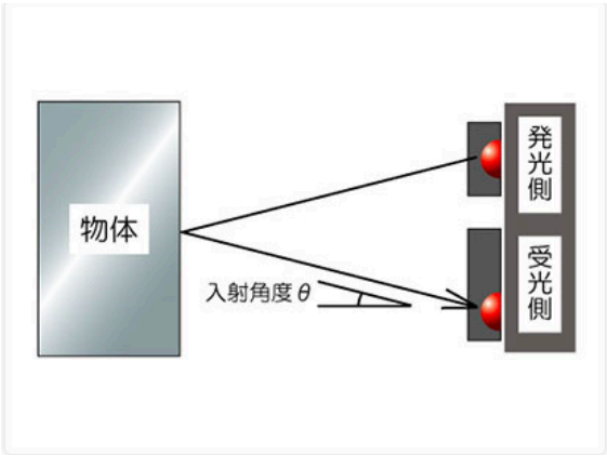

図 2: PSD 方式

2.1.3 どのような入力を取り扱うのか

距離センサーから出力された光を受光し、その出力電圧を取り扱う。 出力電圧は距離に応じて変化する。

2.1.4 入力に応じて出力がどう変化するのか（データシートや仕様書を参考に）

SHARP によると、以下のような特性がある[3] 。

図 3 をみると、38.3ms±9.6ms 毎に出力電圧を出力する。 最初のタイミングは測定をされていないため、不安定な値を出力されるが、2 回目以降は安定した値を出力する。

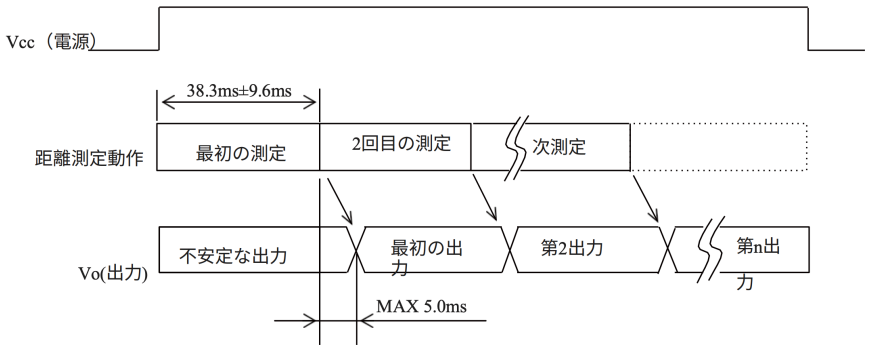

図 3: 距離センサタイミングチャート

出力電圧は距離に応じて変化する。距離が近いほど出力電圧は大きくなり、遠いほど小さくなる。距離と出力電圧の関係は 図 4 に示す。 グラフは比例関係ではなく、反比例な関係がある。 しかし、距離が近すぎるとうまく測定ができないため注意が必要である。

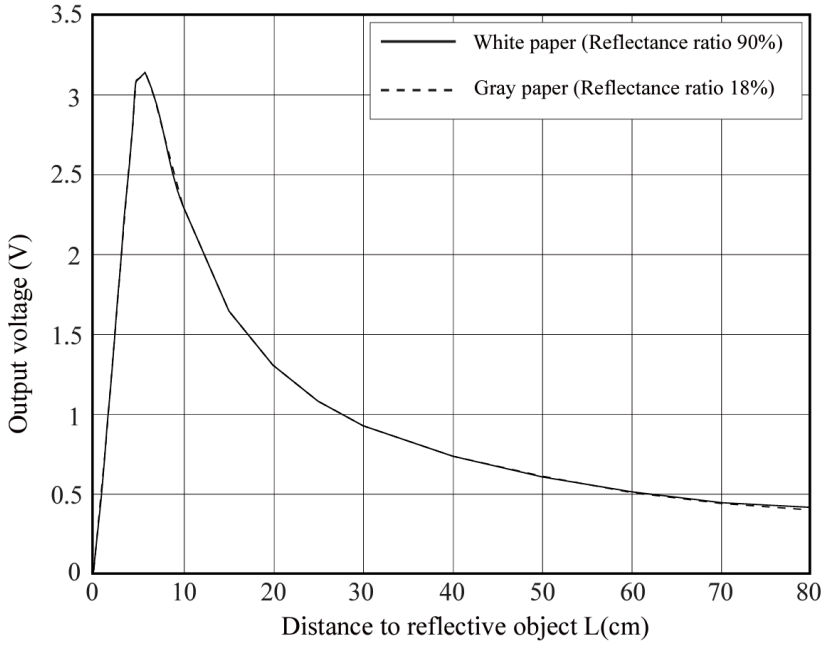

図 4: 距離センサ距離測定特性

2.1.5 どのようなピンアサイン（各ピンの役割）か

SHARP によると、以下のものを扱う[3] 。 図 5 をみると、以下のようなピンアサインがある。

1. 出力電圧
2. GND
3. Vcc（5V 電源電圧）

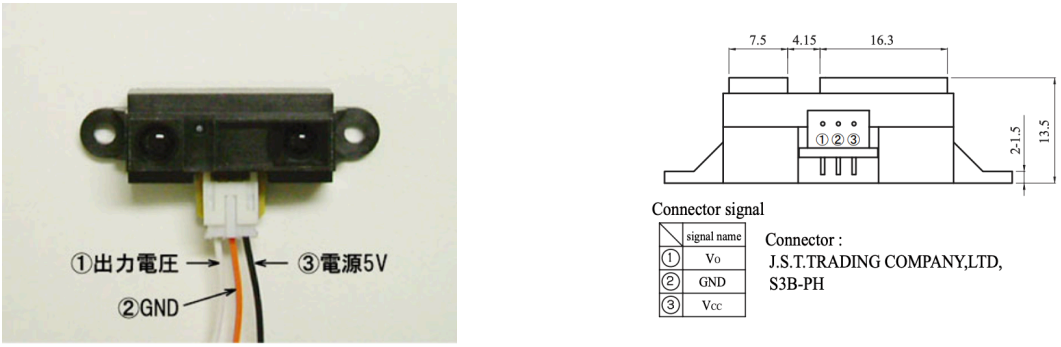

図 5: 距離センサーのピンアサイン

2.1.6 正しい動作の条件，範囲は何か

秋月電子によると、以下のような仕様がある [1] 。

- 電源電圧 min. : 4.5V
- 電源電圧 max. : 5.5V
- 測定距離 min. : 0.1m
- 測定距離 max. : 0.8m
- 測定方式 : 赤外線 PSD
- 測定項目 : 距離
- インターフェイス : アナログ
- 動作温度 min. : −10℃
- 動作温度 max. : 60℃
- 長辺 : 44.5mm
- 短辺 : 13.5mm
- 高さ : 18.9mm

3 課題内容

3.1 物体とセンサとの距離をはかる

3.1.1 実験その 1（部品動作の理解）距離センサからの距離が以下のような場合、アナログ入力はどのような値をとるか調べなさい。

- 卓上に上向きに置き，天井までに物体がないとき
- センサから 50cm のところに手をかざしたとき
- センサから 20cm のところに手をかざしたとき
- センサから 10cm のところに手をかざしたとき

回路図

図 6 は、実験 1 の回路図を示す。 距離センサをアナログ入力に接続して、距離センサと対象物との距離を測定する装置を作成する。

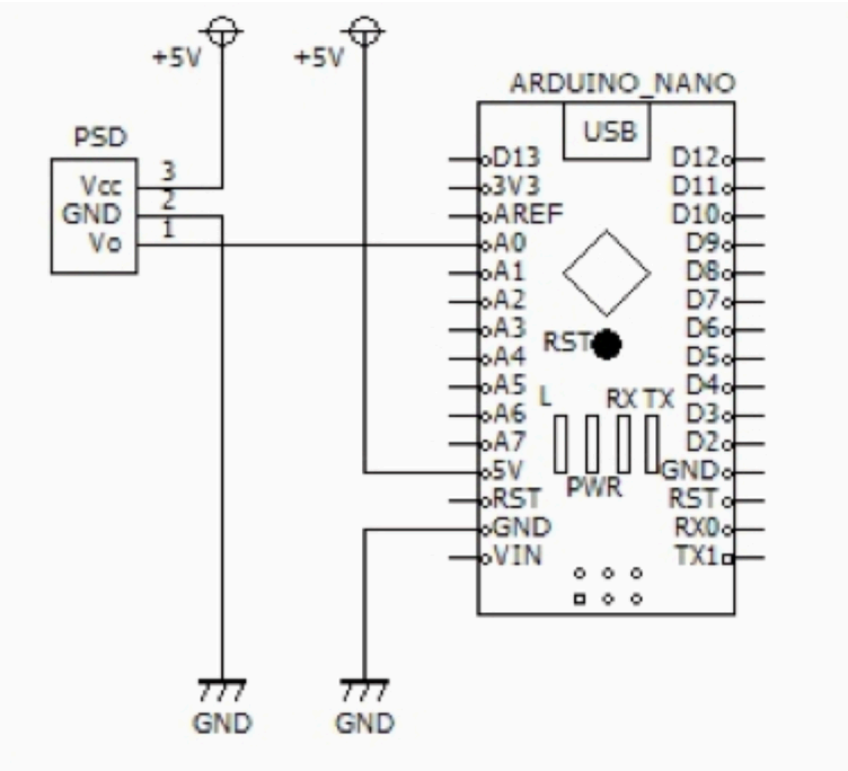

図 6: 実験 1 回路図

プログラム

リスト 1 は、実験 1 で使用したアナログ入力値を取得するソースコードを示している。

```
1  //プログラムに必要な変数の宣言及び定義
2  import processing.serial.*;
3  import cc.arduino.*;
4
5  Arduino arduino;
6  PFont myFont;
7
8  int usePin0 = 0; //Arduino A0 ピン
9
10 //Arduino 及びプログラムの初期設定
11 void setup(){
12     size(600, 250);
13     arduino = new Arduino(
14         this,
15         "/dev/cu.usbserial-14P54810"
16     );
17     myFont = loadFont("CourierNewPSMT-48.vlw");
18     textFont(myFont, 30);
19     frameRate(30);
20 }
21
22 // 入力値の格納用変数
23 int input0;
24
25 //プログラム本体（以下を繰り返し実行）
26 void draw(){
27     background(120);
28     input0 = arduino.analogRead(usePin0);
29
30     //入力値を表示
31     text("A0: " + input0, 50, 100);
32 }
```

リスト 1: 値の取得をするソースコード

- プログラムの概要

センサから ArduinoA0 ピンへの入力値をアナログ入力として読み込む。読み込んだ値を数値として表示する。

- プログラムの説明
  - 17 行目: プログラムに必要な変数の宣言および定義またはライブラリのインポートを行う。
  - 8 行目で ArduinoA0 ピンの使用を usePin0 = 0 として定義している。
  - 11–20 行目: Arduino およびプログラムの初期設定
    - 10 行目で画面表示に用いるウィンドウサイズを横 600px,縦 250px と定義している。
    - 11 行目で"/dev/cu.usbserial-14P54810″ のポートと 57600 の速度で通信する arduino インスタンスを生成する.
    - 14 行目でフレームレートを 30 としている.
  - 23–32 行目：プログラムの動作
    - 23 行目で入力用の変数 input0 を宣言している。
    - 27 行目で背景色を灰色に設定する。
    - 28 行目で ArduinoA0 ピンのアナログ入力を input0 に入れる。
    - 31 行目で input0 を数値として表示する.

結果

図 7 は、実験 1 の結果における、距離センサーの距離とマイコンへのアナログ入力値の関係を示している。 グラフによると、距離センサに物体を近づければ、マイコンのアナログ入力値が高くなる。

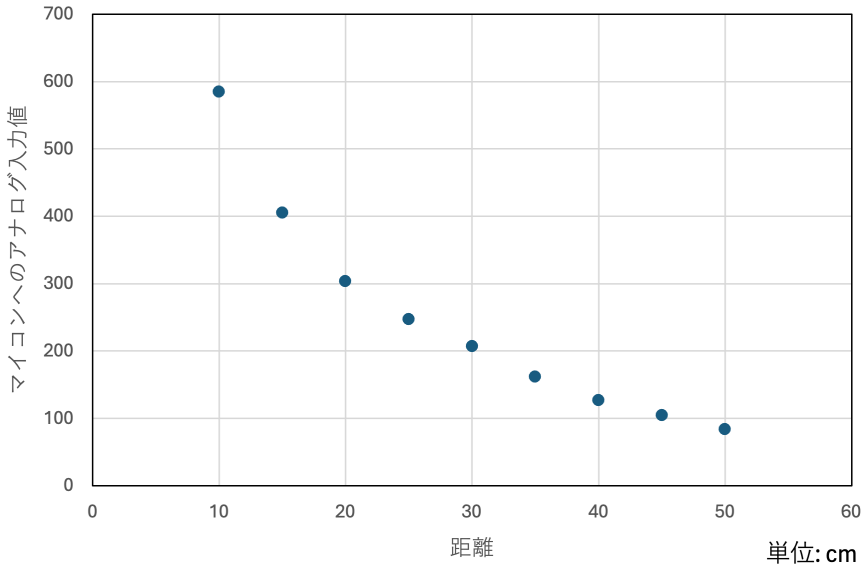

図 7: 距離センサ距離測定特性

考察

出力される値は距離と反比例の関係にある。 距離センサーは反比例の関係にあるため、アナログ入力値を用いて距離の測定ができる。距離が 10 20cm の間が一番感度が高く、他の部分に比べて細かい距離の変化を感知できる。

3.1.2 実験その 2 (動作可能範囲の確認)

プレレポートから距離センサには動作可能範囲があり、ある一定条件下では正しいアナログ値が得られないことが分かる。そこで動作不可能範囲についてもアナログ入力値を調べ、動作可能範囲、不可能範囲の両方を含むグラフを作成しなさい。

回路図・プログラム

装置、プログラムは実験その 1 と同じものを使用する。

結果

図 8 をみると、距離センサは近くなる程、値が増えていくはずだが、10cm より近づくと減っていく。 また、80cm より遠くなると値は同じになってしまう。 プレレポートで検索した情報と一致している。

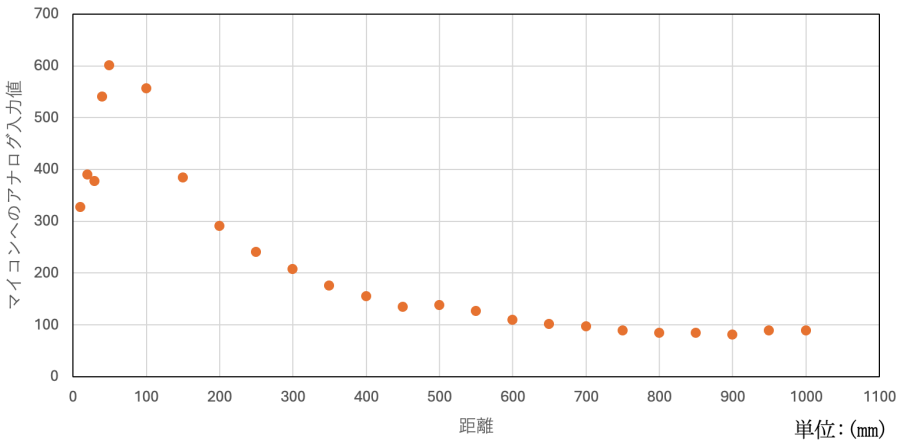

図 8: 距離センサ距離測定特性

考察

距離センサは、測定可能範囲が 50mm 800mm 程度である。この範囲外では正確な値が得られない。 理由としては、遠すぎると反射光が弱くなり、受講部に届かなくなるため、近すぎると発酵部と反射部の間で乱反射してしまい、正確な値が得られないためだと考えられる。測定可能範囲は 50mm 800mm 程度だと考えられる。

データシートとも比較しても、ほぼ一致しているため、正確な値が得られると考えられる。

3.1.3 発展その 1 (距離センサを用いた状態識別)

距離センサへの入力値により、ある 2 つの状態を「distant」もしくは「close」と識別して表示させなさい。また、ある 2 つの状態 (例えば，パソコンの全高より高いか低いか)」は自分で決め、これを実現する境界値と不感帯を導き出して使用しなさい。

回路図・プログラム

装置は実験その 1 と同じものを使用する。 リスト 2 は、発展その 1 で使用した距離センサを用いた状態識別するソースコードを示している。

```
//プログラムに必要な変数の宣言及び定義
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
PFont myFont;

int usePin0 = 0; //Arduino A0 ピン
int usePin1 = 3; //Arduino A1 ピン

//Arduino 及びプログラムの初期設定
void setup(){
    size(600, 250);
    arduino = new Arduino(
        this,
        "/dev/cu.usbserial-14P54810"
    );
    myFont = loadFont("CourierNewPSMT-48.vlw");
    textFont(myFont, 30);
    frameRate(30);
}

// 入力値の格納用変数
int input0;

// 不感帯の閾値
int closeDiv = 330;
int distantDiv = 270;

// 状態の格納用変数
String status = "";

//プログラム本体（以下を繰り返し実行）
void draw(){
    background(120);
    input0 = arduino.analogRead(usePin0);

    //入力値を表示
    text("A0: " + input0, 50, 100);

    // 不感帯の設定
    if(input0 > closeDiv){
        status = "close";
    }

    if(input0 < distantDiv){
        status = "distant";
    }

    // 状態の表示
    text("Status: " + status, 50, 150);
}
```

リスト 2: 距離センサを用いた状態識別するソースコード

- プログラムの概要

センサから ArduinoA0 ピンへの入力値をアナログ入力として読み込む。読み込んだ値を数値として表示し、その値によって状態を識別する。

- プログラムの説明
  - 1–9 行目: プログラムに必要な変数の宣言および定義またはライブラリのインポートを行う。
  - 8,9 行目で Arduino A0, D3 ピンの使用を usePin0 = 0 , usePin1 = 3 として定義している。
  - 12–21 行目: Arduino およびプログラムの初期設定
    - 10 行目で画面表示に用いるウィンドウサイズを横 600px,縦 250px と定義している。
    - 11 行目で"/dev/cu.usbserial-14P54810″ のポートと通信する arduino インスタンスを生成する.
    - 14 行目でフレームレートを 30 としている.
  - 23–52 行目：プログラムの動作
    - 24 行目で入力用の変数 input0 を宣言している。
    - 27,28 行目で閾値を設定している。
    - 30 行目で状態を格納する変数 status を宣言している。
    - 35 行目で背景色を灰色に設定する。
    - 36 行目で ArduinoA0 ピンのアナログ入力を input0 に入れる。
    - 39 行目で input0 を数値として表示する.
    - 42–48 行目で入力値によって状態を識別する。不感帯を作成し、その範囲内で状態を識別する。
    - 51 行目で状態を表示する。

結果

今回、MacBookAir のディスプレイの高さを閾値とし、閾値より高いか低いかで状態を識別する。 MacBookAir のディスプレイの高さより低いか高いかで推定を行う。としては、マイコンへのアナログ入力値 300 を目処としている。

また、ディスプレイの高さギリギリでだと推定が不安定になるため不感帯を儲ける。不感帯は 270–330 とする。

考察

不感帯をこれより狭めた状態で、センサと物体の距離を 20cm 前後を保つと、状態が安定しない。 逆に不感帯を大きくすると判定される距離が広くなり、測りたい距離を正確に測れなくなる。 そのため、今回不感帯の設定を 270-330 とし、距離センサと物体の距離を 20cm 前後を保っても、状態の安定を確認できた。

3.1.4 発展その 2（状態識別を用いた LED 点灯制御）

第 3 章で用いた LED を使い、一定距離以上近づくと、LED が点灯するようにしなさい. また、「一定距離」が何を指すかを自分で決め、これを実現する境界値と不感帯を導き出して使用しなさい。

回路図・プログラム

図 9 は、発展その 2 の回路図を示す。 装置は実験その 1 に LED を追加して使用する。

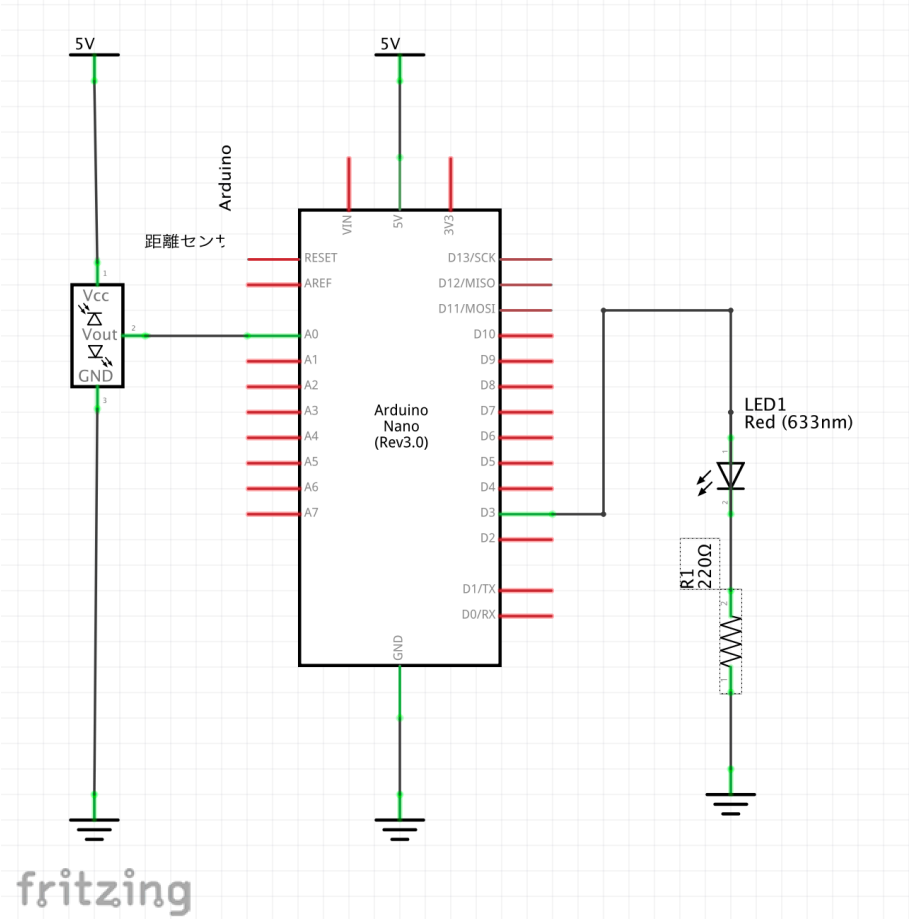

図 9: 実験 2 回路図

リスト 3 は、発展その 2 で使用した距離センサを用いた状態識別して LED 表示するソースコードを示している。 発展その 1 で作成したソースコードの 51 と 52 行目の間に挿入する。

```
// 状態に応じた処理
if(status.equals("close")){
    //close の時の処理
    arduino.analogWrite(usePin1, 255);
}else if(status.equals("distant")){
    //distant の時の処理
    arduino.analogWrite(usePin1, 0);
}
```

リスト 3: 距離センサを用いた状態識別して LED 表示するソースコード

- プログラムの概要

センサから ArduinoA0 ピンへの入力値をアナログ入力として読み込む。読み込んだ値を数値として表示し、その値によって状態を識別する。

- プログラムの説明
  - 2 行目で status が close か判定をする。
  - 4 行目で usePin1 に 5V を出力する。
  - 6 行目で status が distant か判定をする。
  - 8 行目で usePin1 に 0V を出力する。

結果

今回、MacBookAir のディスプレイの高さを閾値とし、閾値より高いか低いかで状態を識別する。 MacBookAir のディスプレイの高さより低いか高いかで推定を行う。としては、マイコンへのアナログ入力値 300 を目処としている。

また、ディスプレイの高さギリギリだと推定が不安定になるため不感帯を儲ける。不感帯は 270–330 とする。

考察

不感帯をこれより狭めた状態で、センサと物体の距離を 20cm 前後を保つと、状態が安定しない。 逆に不感帯を大きくすると測りたい距離を正確に測れなくなる。 LED を追加したので、センサと物体の距離が一定以上近づくと LED が点灯するようになった。それにより、ハードウェアのみで入力から出力までの処理を行えるようになった。

3.1.5 まとめ

今回の実験を通して、距離センサ GP2Y0A21YK0F の使い方と仕組みを習得できた。

距離センサは、反比例の関係にあるため、アナログ入力値を用いて距離の測定ができる。距離が 10 20cm の間が一番感度が高く、他の部分に比べて細かい距離の変化を感知できる。
また、測定可能範囲が 50mm 800mm 程度であると分かった。

距離センサを用いた状態識別を行うと、センサと物体の距離を 20cm 前後を保つ時、状態が安定しないことが分かった。 そのため、不感帯を用いて、センサと物体の距離を 20cm 前後を保っても、状態の安定を確認できた。

参考文献

[1] 秋月電子, 「シャープ測距モジュール GP2Y0A21YK」. 参照: 2024 年. [Online]. 入手先: https://akizukidenshi.com/catalog/g/g102551/

[2] 国立大学 55 工学系学部ホームページ運営事務局, 「赤外線距離センサーを使おう」. 参照: 2024 年. [Online]. 入手先: https://www.mirai-kougaku.jp/laboratory/pages/161221.php

[3] SHARP, 「Distance Measuring Sensor Unit Measuring distance: 10 to 80 cm Analog output type」. 参照: 2024 年. [Online]. 入手先: https://akizukidenshi.com/goodsaffix/gp2y0a21yk_e.pdf