システム実験
基礎実験 3

6119019056 山口力也

2019/05/10 日提出

1　電圧, ディジタル値との関係

演習 2.3.1 の結果を報告し, その結果から AD 変換結果 (ディジタル値) と電圧のグラフを作成せよ.

- AD 変換結果 (ディジタル値)-A_0 の計算上の電圧 V_o[V]
- AD 変換結果 (ディジタル値)-AD 変換結果から求めた電圧 [V]

以下図 1, 図 2 に結果のグラフを示す.

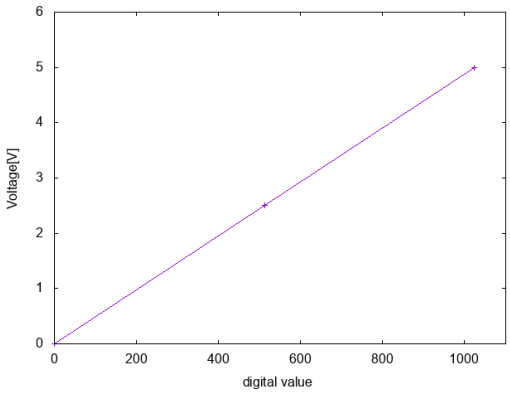

図 1: 演習 2.3.1 の結果 (理論値)

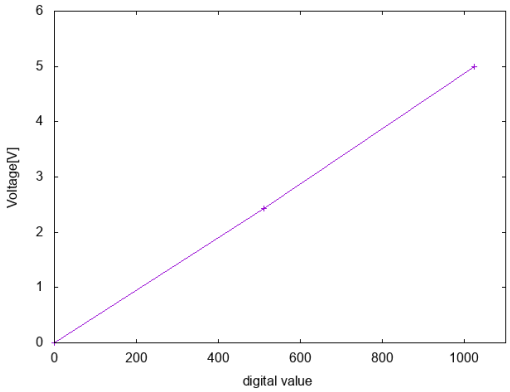

図 2: 演習 2.3.1 の結果 (測定値)

2　半固定抵抗による LED の点灯・消灯

課題 2.3.1 において実装したブレッドボード配線図およびプログラムを報告せよ. シリアルモニタで確認した値と LED の点灯・消灯の状況を報告せよ. さらにプログラム作成時に工夫した点を記せ. また, 抵抗値を調整することで, なぜ図 2.40 の A_0 の電圧が変化するのか,A_0 における電圧を図 2.38(b) を参考にして考察せよ.

以下図 3 にブレッドボード配線図を示す.

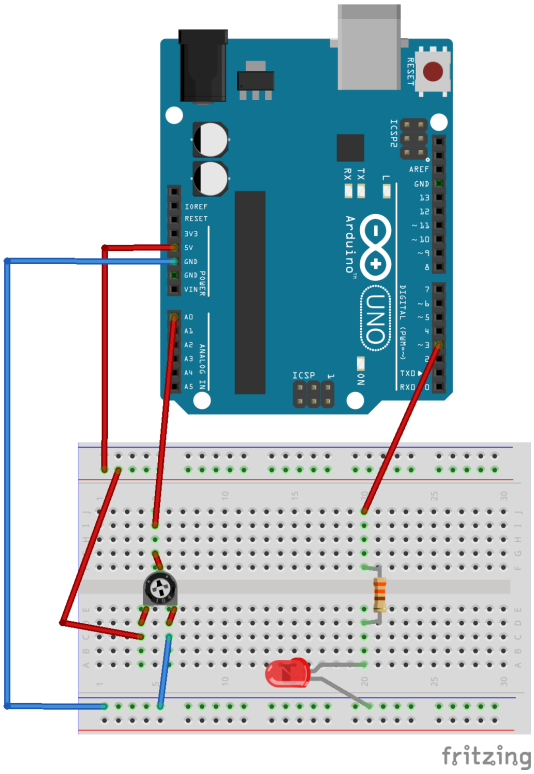

図 3: 課題 2.3.1 の配線図

以下ソースコード 1 にプログラムを示す.

ソースコード 1: 課題 2.3.1

```
1  const int LED_PIN = 3;//LED_PIN を定義
2  void setup() {
3    Serial.begin(9600);
4    // シリアル通信を 9600kbps で初期化
5    pinMode(LED_PIN,OUTPUT);
6    //LED_PIN を出力に設定
7  }
8  void loop() {
9    int sensorValue = analogRead(A0);//
         A0 ピンの AD 変換結果を取得する.
10   float vo = sensorValue*(5.0/1024.0);//
         sensorValue の値を電圧値に変換
11   Serial.println(vo);//システムモニタにvo を表示
12   delay(1); // 安定用
13   if(vo>2.50){//2.5V 以上なら
```

```
14      digitalWrite(LED_PIN,HIGH);//LED 点灯
15    }
16    else{//2.5V 以下なら
17      digitalWrite(LED_PIN,LOW);//LED 消灯
18    }
19  }
```

シリアルモニタで確認した時

プログラム作成時に工夫した点は特にないが,LED_PIN を定義してわかりやすくした点である. 閾値の 2.50 を定義して閾値の変更を容易にするともう少し良いプログラムになったと思う. 抵抗値を調整することで電圧が変化するのはオームの法則より, 抵抗に流れる電流が変化し, 電圧が変化するためである.

3　照度センサによる LED の点灯・消灯

課題 2.3.2 において実装したブレッドボード配線図, プログラムおよびプログラムの実行結果を報告せよ. また, プログラム作成において工夫した点を記せ. 演習 2.3.3 で作成したプログラムによる, シリアルモニタで確認した照度の最大値および最小値を報告せよ. 作成したプログラムによる, 明るさの変化と LED の点灯・消灯の状況を報告せよ.

以下図 4 にブレッドボード配線図を示す.

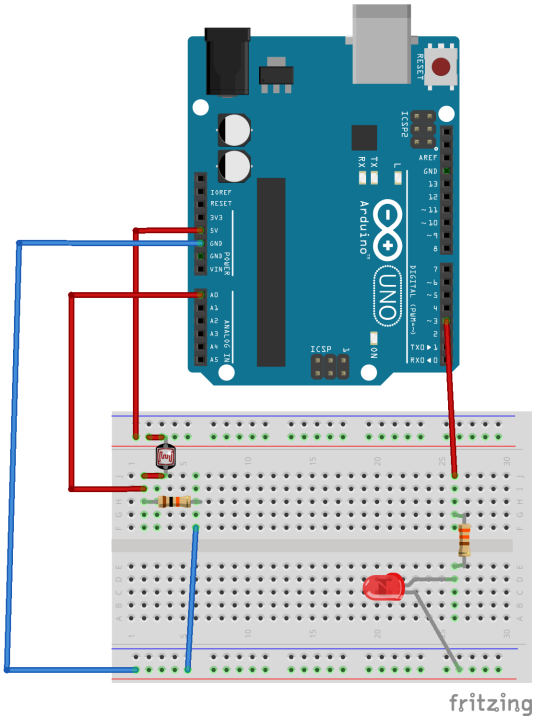

図 4: 課題 2.3.2 の配線図

また, 以下ソースコード 2 にプログラムを示す.

ソースコード 2: 課題 2.3.2

```
20 const int LED_PIN = 3; //LED_PIN を 3 と定義
21 void setup() {
22   Serial.begin(9600);
23   // シリアル通信を 9600kbps で初期化
24   pinMode(LED_PIN,OUTPUT);
25   //LED_PIN を出力に設定
26 }
27 void loop() {
28   int sensorValue = analogRead(A0);//
         A0 ピンの AD 変換結果を取得する.
29   float vo = sensorValue*(5.0/1024.0);//デジタル値を電圧値に
         変換
30   float L =222*vo; //電圧を照度値に変換
31   Serial.println(L); //照度値をシステムモニタに表示
32   digitalWrite(LED_PIN,lux_threshold(L)); //
         LED_PIN ポートに threshold の戻り値を出力
33   delay(1); //安定用
```

```
34 }
35
36 int lux_threshold(float lux){ //中間値用
37   int th_val = 0;
38   if(lux > 1.08){//中間値 (1.08)より大きいなら
39     th_val = HIGH; //LED 点灯
40   }
41   else{ //そうでないなら
42     th_val = LOW; //LED 消灯
43   }
44   return th_val; //th_val を返す
45 }
```

プログラムで工夫した点はデジタル値を電圧値, 照度値に変換する時に一気にやるのではなく, 一度電圧値に変換してから照度値に変換することでわかりやすく, また他のシステムでも使いやすくしたところである. ただ,5.0/1024.0 や, 関数 lux_threshold の中間値の 1.08 なども定義すると状況が変わっても容易に変更できたと思う. シリアルモニタで確認した照度値の最大値は 2.17, 最小値は 0 だった.LED は明るい場合点灯し, 暗い場合消灯した.

4 ディジタル PWM による LED の明るさ調整

演習 2.3.4 で実装したブレッドボード配線図, プログラムおよび演習結果 (表) および analogWrite の引数と 2 つの LED の明るさを見比べた結果を報告せよ. また,PWM 制御によりなぜ LED の明るさが変化するのか, その理由を考察せよ.

以下図 5 にブレッドボード配線図を示す.

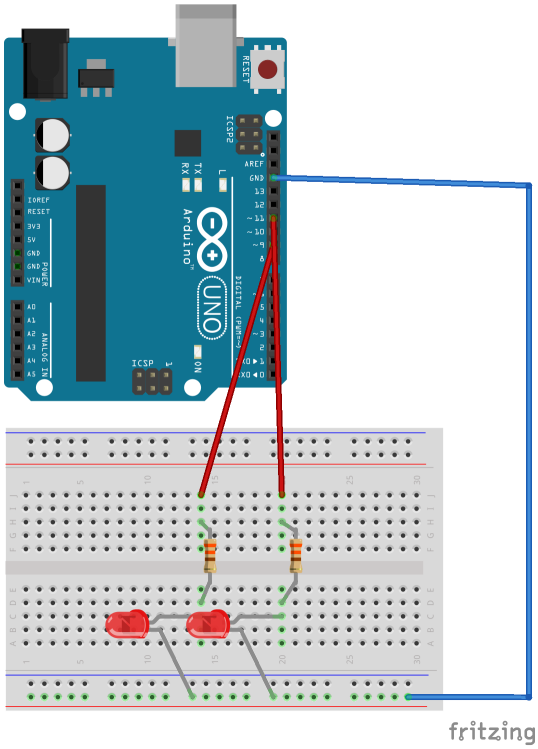

図 5: 演習 2.3.4 の配線図

また, 以下ソースコード code:enshu2-3-4 にプログラムを示す.

ソースコード 3: 演習 2.3.4

```
46 //それぞれのポート番号を定義
47 const int LED_RED_PIN = 9;
48 const int LED_YEL_PIN = 11;
49 void setup() {
50   Serial.begin(9600);
51   // シリアル通信を 9600kbps で初期化
52   pinMode(LED_RED_PIN,OUTPUT);
53   pinMode(LED_YEL_PIN,OUTPUT);
54   //LED のポートをそれぞれ出力に設定
55 }
56 void loop() {
57     analogWrite(LED_RED_PIN,64);//赤色に 25%出力
58     analogWrite(LED_YEL_PIN,192);//黄色に 75%出力
59     delay(10);//安定用
60   }
61 }
```

analogWrite の引数を大きくすると LED の明るさは明るくなる.PWM 制御により LED の明るさが変化するのは, デューティ比により見かけの電圧が変化するためである. 電圧が変化すると LED に流れる電流が変化するので LED の明るさも変化する. 以下表 1 に結果の表を示す.

表 1: 演習 2.3.4 の結果

デューティ比 [%]	analogWrite の引数
0	0
25	64
50	128
75	192
100	255

5　ディジタル PWM による LED の明るさ調整 2

課題 2.3.3 で実装した, ブレッドボード配線図, プログラムおよび実験結果を報告せよ. また, プログラムで工夫した点を記せ.

以下図 6 にブレッドボード配線図を示す.

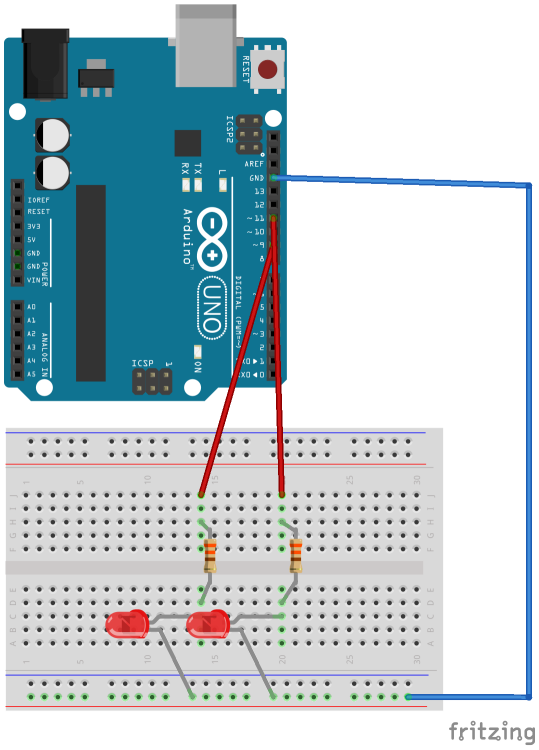

図 6: 課題 2.3.3 の配線図

また, 以下ソースコード 4 にプログラムを示す.

ソースコード 4: 課題 2.3.3

```
62 //それぞれのポート番号を定義
63 const int LED_RED_PIN = 9;
64 const int LED_YEL_PIN = 11;
65 void setup() {
66   Serial.begin(9600);
67   // シリアル通信を 9600kbps で初期化
68   pinMode(LED_RED_PIN,OUTPUT);
69   pinMode(LED_YEL_PIN,OUTPUT);
70   //LED のポートをそれぞれ出力に設定
71 }
72 void loop() {
73   for(int i = 0; i <= 256; i+=64){
74     if(i==256){//256は引数にならないので
75       i--;
76     }
77     analogWrite(LED_RED_PIN,i);//それぞれにi を出力
```

```
78      analogWrite(LED_YEL_PIN,i);//それぞれにi を出力
79      delay(1000);//1秒待つ
80    }
81  }
```

実験結果としては 1 秒ごとに LED の明るさが変化した. プログラムで工夫した点は for 文の中で analogWrite は 256 を引数に取らないので if 文をつけて条件化して 256 を排除したところである.

6　発展課題 2.3.1

発展課題 2.3.1 で実装したブレッドボード配線図, プログラムおよび実験結果を報告せよ. また, 圧電ブザーが鳴る原理について調査せよ.

以下図 7 にブレッドボード配線図を示す.

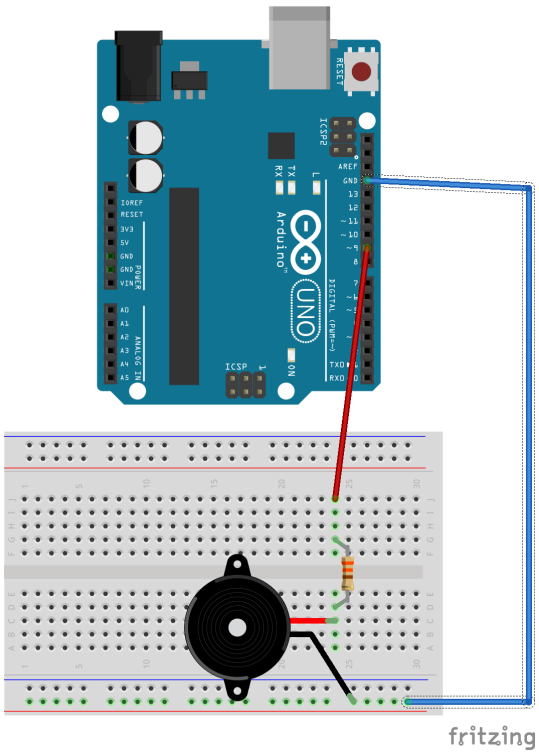

図 7: 発展課題 2.3.1 の配線図

また, 以下ソースコード 5 にプログラムを示す.

ソースコード 5: 発展課題 2.3.1

```
82 //値を定義
83 const int buzzer = 9;
84 void setup() {
85   Serial.begin(9600);
86   // シリアル通信を 9600kbps で初期化
87   pinMode(buzzer,OUTPUT);
88   //buzzer を出力に設定
89 }
90 void loop() {
91   for(int i = 0; i <= 256; i+=64){
92     if(i==256){//256は引数にならないので
93       i--;
94     }
95     analogWrite(buzzer,i);//buzzer に出力
96     delay(1000);//1秒待つ
97   }
98 }
```

結果としては段階的に圧電ブザーの音が高くなった. 圧電ブザーは, 電圧をかけると振動する板と, その振動版に金属が貼り合わされた構造になっている. 電圧をかけると, 圧電セラミックスが伸びて金属板が曲がり,OFF にすると形が戻る. この動作を繰り返すことで振動し音が鳴る.PWM 機能を用いて ON と OFF の割合を変更することで周波数が変化し, 音の高さが変わる.

7 発展課題 2.3.2

発展課題 2.3.2 で実装したブレッドボード配線図, プログラムおよび実験結果を報告せよ. またプログラムで工夫した点を記せ.

以下図 8 にブレッドボード配線図を示す.

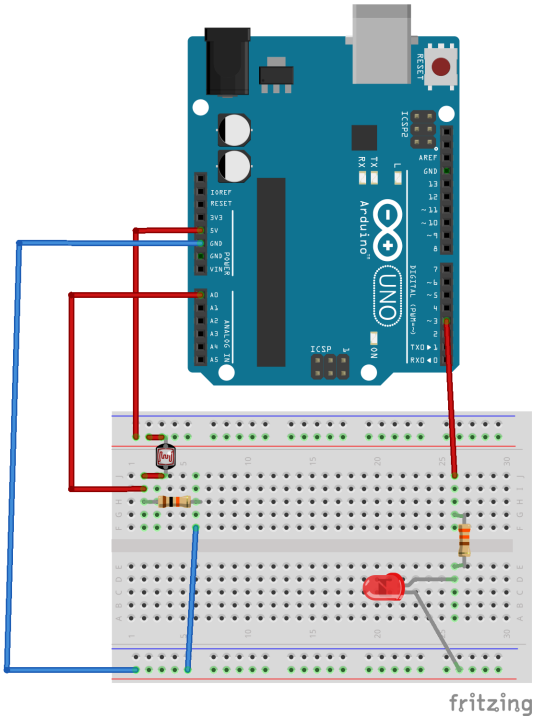

図 8: 発展課題 2.3.2 の配線図

また, 以下ソースコード 6 にプログラムを示す.

ソースコード 6: 発展課題 2.3.2

```
99  //値を定義
100 const int LED_PIN = 9;
101 void setup() {
102   Serial.begin(9600);
103   // シリアル通信を 9600kbps で初期化
104   pinMode(LED_PIN,OUTPUT);
105   //LED を出力に設定
106 }
107 void loop() {
108   int sensorValue = analogRead(A0);//
          A0 ピンの AD 変換結果を取得する.
109   float vo = sensorValue*(5.0/1024.0);//デジタル値を電圧値に
          変換
110   float L =222*vo;//電圧を照度値に変換
111   Serial.println(L);//照度値をシステムモニタに表示
112   analogWrite(LED_PIN,convert(L,0.0,2.17));//
          LED_PIN ポートに convert の戻り値を出力
```

```
113   delay(1); //安定用
114 }
115
116 int convert(float lux,float lux_min,float lux_max){
117   float out = lux/(lux_min - lux_max) *255;
118   //照度値をPWM の出力に変換
119   return out;//out を返す
120 }
```

プログラムで工夫した点は, convert の部分を関数化して使いやすくしたところである.

8 発展課題 2.3.3

発展課題 2.3.3 で実装したブレッドボード配線図, プログラムおよび実験結果を報告せよ.2 つの PWM の変化方法により,LED の変化状態がどのように違ったか報告し, なぜ違いが生じているのか考察せよ. さらに, プログラムで工夫した点を記せ.

以下図 9 にブレッドボード配線図を示す.

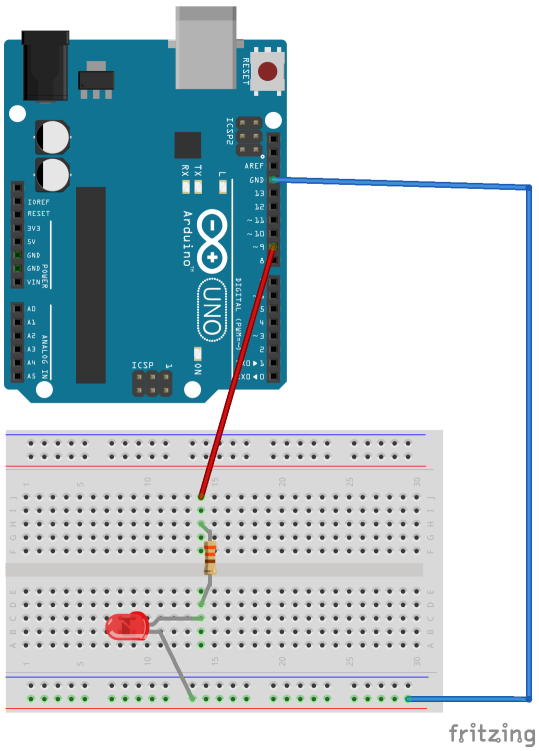

図 9: 発展課題 2.3.3 の配線図

また, 以下ソースコード 7 にノコギリ波の, ソースコード 8 に三角波のプログラムを示す.

ソースコード 7: 発展課題 2.3.3 ノコギリ波

```
121 //値を定義
122 const int LED_PIN = 9;
123 void setup() {
124   Serial.begin(9600);
125   // シリアル通信を 9600kbps で初期化
126   pinMode(LED_PIN,OUTPUT);
127   //LED_PIN を出力に設定
128 }
129 void loop() {
130   for(int i = 0; i < 256 ; i++){
131     analogWrite(LED_PIN,i);//LED_PIN に i を出力
132     delay(1000/255);//255回で 1秒待つ
133   }
134 }
```

ソースコード 8: 発展課題 2.3.3 三角波

```
135 //値を定義
136 const int LED_PIN = 9;
137 void setup() {
138   Serial.begin(9600);
139   // シリアル通信を 9600kbps で初期化
140   pinMode(LED_PIN,OUTPUT);
141   //LED_PIN を出力に設定
142 }
143 void loop() {
144   for(int i = 0; i <=204 ; i++){//80パーセントは 204
145      analogWrite(LED_PIN,i);//i を LED_PIN に出力
146      delay(1000/204);//204回で 1秒
147   }
148   for(int j = 204; j>0 ; j--){
149     analogWrite(LED_PIN,j);//i を LED_PIN に出力
150     delay(1000/204);//204回で 1秒待つ
151   }
152 }
```

ノコギリ波では,LED は徐々に明るくなったあと一定の明るさまで到達すると一気に明るさが落ち最初の状態となり, 三角波では,LED は徐々に明るくなったあと一定の明るさまで到達するとその後徐々に最初の明るさまで暗くなっていった. これらの違いは PWM の変化方法が違うからである. ノコギリ波と

三角波のグラフを見れば一目瞭然だが, ノコギリ派はある一定の値に達した後最初の値に戻るが, 三角波はある一定の値に達した後は上昇した傾きと対照な傾きで値が下降していく．プログラムで工夫した点は三角波の部分についてif 文二つでわかりやすく簡潔にかいたことである.