レポート実験テーマ	制御工学実験　Ⅱ マイコン基礎・応用
所属・氏名 （共同実験者名は括弧内）	熊本高等専門学校　制御情報システム工学科 4 年 11 番　氏名 川口晃平
実験場所	5 号棟 3 階 ICT 活用アクティブラーニングルーム
実施日（第 1 週，第 2 週）	令和 3 年 7 月　6 日（火曜日），令和 3 年 7 月 20 日（火曜日）
レポート提出日	令和 3 年 7 月 19 日（月曜日）

評価項目 （A: 達成できている，B: 概ね達成できている，C: ほとんど達成できていない，D: 達成できていない）		自己評価 （A～D）	担当評価 （A～D）
実施評価	実験開始までに実験テキストや実験ノートを準備できており，事前課題がある場合は，それに取り組んでいた.	A	
	担当者による指示をよく聞き，不注意による無用な誤りなく安全に実験を行うことができた.	A	
	回路やプログラムを自分で作成し，グループワークの場合は自らの役割を全うするなど，課題に対して積極的に取り組むことができた.	A	
	与えられた課題を時間内に達成し，結果を正確に記録または出力できた.	A	
	使用器具の後片付けや実験場所の清掃をきちんと行った.	A	
レポート評価	章立ては適切であり，それぞれの章における記載内容は**自作のものである.** 引用がある場合は、その旨を明記している。	A	
	図・表の書き方は裏面の要領に準じており，**自作のものである.**（担当者が許可しない限り，指導書の図すら引用してはいけない）	A	
	使用器具や実験環境について，実験結果を再現するのに十分な情報を記載している.	A	
	課題に関する計測結果や出力結果を整理して記載し，結果に対する独自の考察を述べている.	A	
	研究課題に取り組み，適切な参考文献を基に答えを導き出している.	A	

※提出期限に w 週間遅れた場合，[レポート評価 = 遅れなしの評価 ×（1 − 0.1w）] とする. 4 週間を過ぎると再実験を行う. また，レポートの内容によっては，担当より再提出を求められることがある.

実施点 （50）	レポート点 （50）	**合計点** **（100）**

1 実験目的

これまでの実験では，マイコンである Arduino を使用して，基本的な機能の使い方について学習した．今回の実験では，マイコンの割り込み機能の使い方を習得する．割り込み機能を使用することで，ロボットの制御などをより正確に制御したり，より複雑なシステムを作成したりすることができる．

2 割り込みとは

2.1 割り込みとは

割り込みとは，現在実行している処理を中断し，他の処理を行うことである．Arduino で割り込みを行うと，現在実行している処理を中断し，他のプログラムを実行する．割り込んだ処理が終了すると中断していた処理を再開する．

delay()関数で時間を止めている場合や他の処理を行っている場合，センサからの値を読み取ることができない．例えば，移動式のロボットがあり，壁との衝突を検知するセンサがあったとする．この場合，delay()関数で時間を止めている時は壁の衝突を検知することができない．同様に，他の処理を実行しているときも壁との衝突を検知できない．割り込みを使用することで，いつでも壁との衝突を検知することができる．

2.2 外部割り込み

外部割り込みとは，Arduino のピンに接続されているスイッチやセンサなどに変化があった場合に割り込みを実行することである．

2.1 で述べたロボットにおいて，衝突を検知するセンサを Arduino に接続し外部割り込みに設定する．こうすることで，delay()関数で時間を止めたり，他の処理を実行しているときでも壁に衝突した瞬間に方向を転換するプログラムを実行できる．

Arduino で外部割り込みを使用する例を図 1，図 2 に示す．

1	`char ledPin = 10;`	LED を接続するピン
2		
3	`void setup() {`	
4	`  Serial.begin(9600);`	シリアル通信を開始
5	`  pinMode(2, INPUT);`	2 番ピンを入力に設定
6	`  pinMode(ledPin, OUTPUT);`	ledPin を出力に設定
7	`  attachInterrupt(0,interruptFunc,FALLING);`	2 番ピンの立ち下がりを検知した時に interrupFunc を実行
8	`}`	
9		
10	`void loop() {`	
11	`  digitalWrite(ledPin, HIGH);`	L チカ
12	`  delay(500);`	
13	`  digitalWrite(ledPin, LOW);`	

```
14      delay(500);
15  }
16
17  void interruptFunc(){
18      Serial.println("ボタンが押されました");      シリアルモニタに出力
19  }
```

図 1 外部割り込みのサンプルのプログラム

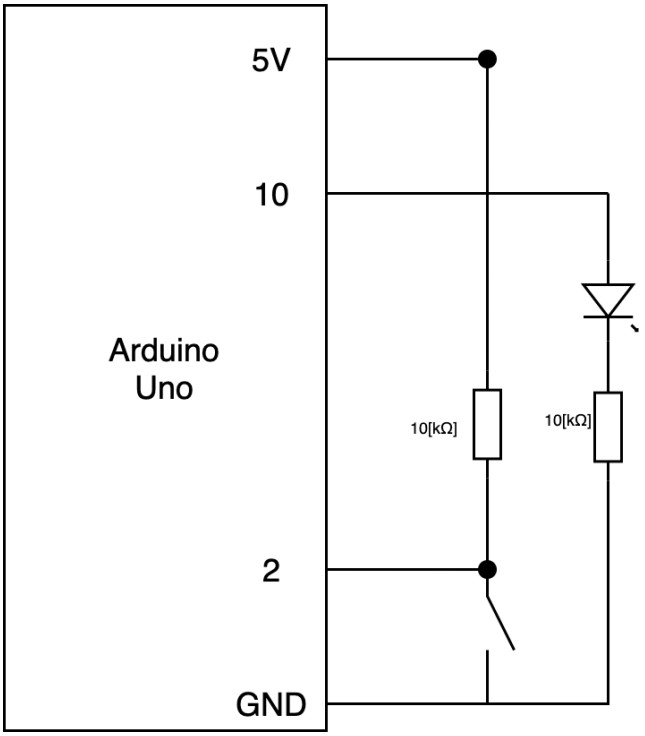

図 2 外部割り込みのサンプルの回路

図 1 の 7 行目で割り込みの設定を行なっている．attachInterrupt()関数の第 1 引数には割り込みに使用するピン番号指定し，Arduino Uno の場合，0 だと 2 番ピン，1 だと 3 番ピンを指定できる．第 2 引数には割り込みが発生した時に実行する関数を指定する．第 3 引数には割り込みを発生させるトリガを指定する．第 3 引数に指定できるパラメータを表 1 に示す．[1]

表 1 attachInterrupt()に指定できるパラメータ

LOW	ピンが LOW のとき
CHANGE	ピンの状態が変化したとき
RASING	ピンの状態が LOW から HIGH に変わったとき
FALLING	ピンの状態が HIGH から LOW に変わったとき

このプログラムを実行すると，スイッチを押すと外部割り込みが発生し，シリアルモニタ上に「ボタンが押されました」と表示される．

2.3 タイマー割り込み

タイマー割り込みとは，一定時間ごとに割り込みの処理を行うことである．

ロボットを直進させる場合，定期的に左右の回転速度を確認し，指定の回転速度に修正する必要がある．定期的に回転数を確認する処理をタイマー割り込みで指定することでこの処理を実現できる．

Arduino でタイマー割り込みを使用する例を図 3 に示す．

```
1  #include <MsTimer2.h>                    タイマー割り込み用のライブラリ
2
3  char ledPin = 13;                        LED を接続するピン
4
5  void setup() {
6    Serial.begin(9600);                    2 番ピンを入力に設定
7    pinMode(ledPin, OUTPUT);               ledPin を出力に設定
8    MsTimer2::set(1000, interruptFunc);    タイマー割り込みを設定
9    MsTimer2::start();                     タイマー割り込みを開始
10 }
11
12 void loop() {
13   digitalWrite(ledPin, HIGH);            L チカ
14   delay(500);
15   digitalWrite(ledPin, LOW);
16   delay(500);
17 }
18
19 void interruptFunc(){
20   Serial.println("1 秒経ちました");       シリアルモニタに出力
21 }
```

図 3 タイマー割り込みの例のプログラム

8 行目でタイマー割り込みの設定をしている．set()関数の第一引数には割り込みの間隔をミリ秒で指定し，第 2 引数には割り込み時に実行したい処理を指定する．

このプログラムを実行すると，1 秒おきに割り込みが発生し，「1 秒経ちました」と表示される．

3 実験方法

3.1 作成したシステム

以下の条件を満たす自動ブラインドシステムを作成する.

入力 : タクトスイッチ x1
　　光センサ x1

出力 : LEDx1
　　サーボモーターx1

1. 暗くなるとブラインドを開ける.
　　光センサを覆うとサーボモーターが正方向に回転する
2. 明るくなるとブラインドを閉める
　　光センサに光を当てるとサーボモーターが逆方向に回転する
3. ブラインドの制御周期は 100ms
4. タクトスイッチを押すと現在の明るさをシリアル通信で PC に送る
5. 本体は飾りになるように LED で L チカまたはぼんやり L チカしておく

3.2 使用した環境

使用した環境を表 2 に示す.

表 2 使用した環境

マイコン	Arduino Uno
開発環境	Arduino IDE 1.8.15

4 実験結果

作成した自動ブラインドシステムのプログラム，回路図，写真を図 4，図 5 に示す.

```
1  #include <Servo.h>                  サーボモーター制御用のライブラリ
2  #include <MsTimer2.h>               タイマー割り込み用のライブラリ
3  Servo blind;                        インスタンス化
4
5  int lightSensorPin = 0;             光センサーを接続するピン
6  int servomotorPin = 10;             サーボモーターを接続するピン
7  int ledPin = 6;                     LED を接続するピン
8  int buttonPin = 8;                  タクトスイッチを接続するピン
9  unsigned long endTime = 0;          前回の割り込みした時間
10
11 void setup()
12 {
13    pinMode(buttonPin, INPUT);
```

```
14    pinMode(ledPin, OUTPUT);
15
16    blind.attach(servomotorPin, 500, 2400);          サーボモーターをのピン，パルス幅を指定
17
18    MsTimer2::set(100, checkBrightness);             タイマー割り込みの設定
19    MsTimer2::start();                               タイマー割り込みを開始
20    attachInterrupt(0, printBrightness, FALLING);    スイッチの外部割り込みの設定
21
22    Serial.begin(9600);                              シリアル通信開始
23  }
24
25  void loop()
26  {
27    int i;
28    for (i = 0; i < 256; i+=i)                       じんわり点灯
29    {
30      analogWrite(ledPin, i);
31      delay(4);
32    }
33    for (i = 255; i >= 0; i-=1)                      じんわり消灯
34    {
35      analogWrite(ledPin, i) ;
36      delay(4);
37    }
38  }
39
40  void checkBrightness()                             明るさを表示する関数
41  {
42    int cds;
43
44    cds = analogRead(lightSensorPin);                電圧を測定
45
46    if (cds < 150)
47    {
48      moveBlind(1);                                  明るいときはブラインドを閉める
49    }
```

```
50    else
51    {
52        moveBlind(0);                    暗いときはブラインドを開ける
53    }
54  }
55
56  void moveBlind(int moveMode)           moveMode=1 でブラインドを開け，0 で閉める
57  {
58    if (moveMode == 0)
59    {
60      blind.write(180);                  180 度正方向に回転
61    }
62    else
63    {
64      blind.write(-180);                 180 度逆方向に回転
65    }
66  }
67
68  void printBrightness()                 明るさを表示する関数
69  {
70    int cds;
71    if((millis() - endTime) > 100){       チャタリング防止
72      cds = analogRead(lightSensorPin);  電圧を測定
73      Serial.println(cds);               明るさを表示する関数
74      endTime = millis();                現在時間を格納
75    }
76  }
```

図 4 自動ブラインドシステムのプログラム

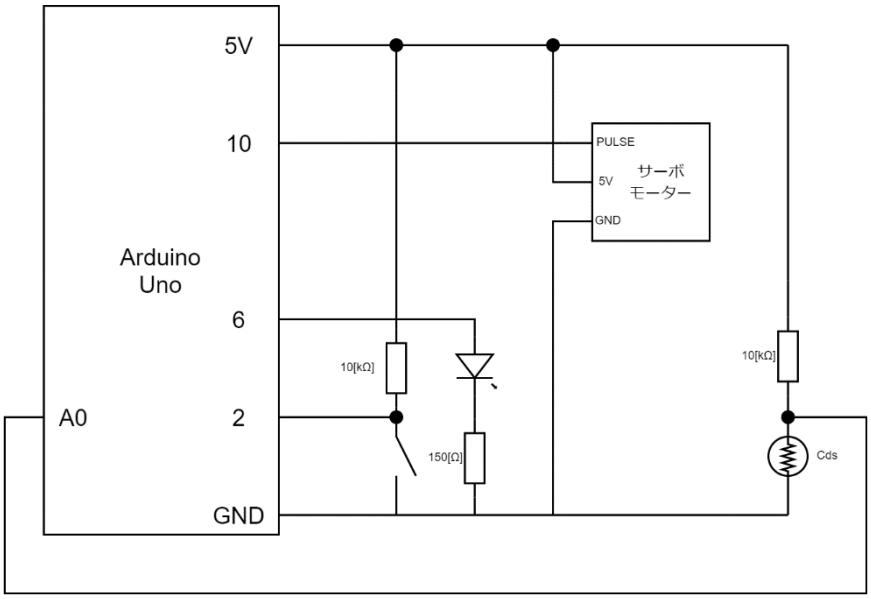

図 5 自動ブラインドシステムの回路図

今回，サーボモーターの制御には Servo ライブラリを使用した．図 4 の 16 行目でサーボモータの設定をしているが，attach()関数の第 1 引数にはサーボモーターの信号ピンの番号，第 2 引数にはサーボの角度が 0 度のときのパルス幅（マイクロ秒），第 3 引数にはサーボの角度が 180 度のときのパルス幅（マイクロ秒）を指定する．[2]

図 4 の 20 行目ではスイッチによる外部割り込みの設定を行っている．図 5 の回路図において，スイッチが開放されている場合 2 番ピンの電位は 5V である．スイッチが短絡されると抵抗で電圧降下が起き，2 番ピンの電位は 0V となる．つまり，スイッチを押すとピンの状態が HIGH から LOW に変化するので，図 4 の 20 行目の attachInterrupt()関数の第 3 引数には FALLING を指定している．

図 4 の 44 行目と 72 行目で A0 ピンの電圧値から明るさを求めている．光センサ（Cds）は明るいと抵抗値が低くなり，暗いと抵抗値が高くなるという特性を持っている．図 5 の回路図において，明るい状態だと Cds の抵抗値が低くなり，A0 ピンの電位は低くなる．一方暗い状態だと Cds の抵抗値は高くなり，A0 ピンの電位は高くなる．よって，analogRead()関数で取得した値が小さいと明るく，値が大きいと暗いことがわかる．

71 行目ではチャタリング防止の処理を行っているが，これについては 6 章の研究事項にて述べる．

システムの動作を簡単に解説する．常時 L チカを行っており，100 ミリ秒ごとに明るさを取得してその状態に応じてブラインドを開閉する．また，スイッチが押されると現在の明るさをシリアルモニタに表示するといった動作をしている．

5 考察

図 5 の明るさを読み取る部分は Cds と直列に抵抗を接続しているが，抵抗値は適切に決める必要がある．抵抗値が大きすぎた場合，明るさが変化しても電圧値はほとんど変化しないため明るさの変化に鈍感になってしまう．抵抗値が小さすぎた場合，明るさの変化には敏感になるが消費電力が増加してしまう．

抵抗と Cds は 5V ピンに接続しているため，常時電力を消費している．明るさを読み取るときにのみ電圧を印加すればいいので，5V ではなく番号のピンの接続し，読み取り時に 5V を出力することで消費電力を減らすことができるのではないか.

Arduino Uno において外部割り込みで使用することができるピンは通常 2 番ピン，3 番ピンである．これらの仕様は Arduino の公式サイトの仕様書にて確認することができる．プログラムでレジスタを直接編集することですべてのピンでピン変化割り込みを使用できる.

明るさの変化を確認するためのタイマー割り込みは 100 ミリ秒に設定しているが，リアルタイム性が重要なものではないのでもう少し制御周期を長くしても良いのではないか.

ブラインドを開閉するしきい値は 150 に設定しているが，値付近の明るさの場合，開閉を繰り返してしまう可能性がある．明るさが 140 以下になったらブラインドを閉め，160 以上になったらブラインドを開けるようにすることでこの問題が解決することができるのではないか.

6 研究事項

チャタリングとその対策について解説する.

図 5 の回路図において，理想的なスイッチを使用したとする．ピンの電圧はスイッチを押すと 0V，スイッチを離すと 5V となる．しかし，実際にはスイッチを押した直後や離した直後に 5V と 0V を短い間隔で繰り返している．この現象をチャタリングという．チャタリングが発生すると複数回スイッチが押されたと判定されるため，今回のシステムでは 1 回しかスイッチを押してないのに複数回明るさが表示されてしまう.

チャタリングの原因はスイッチの接点のバウンド，擦れなどによって発生する．これらの原因をなくすことは不可能に近いため，回路に素子を追加したり，プログラム側で対応したりすることが多い.

今回作成したシステムではプログラム側でチャタリングの対策を行った.

スイッチが押された場合，前回の外部割り込みから 100 ミリ秒以上経っている場合に明るさを表示するようにした．チャタリングは 100 ミリ秒以内に収まるため，このような設定にした.

millis()関数は Arduino ボードがプログラムの実行を開始した時から現在までの時間をミリ秒単位で取得する関数である．図 4 の 71 行目で現在時間と前回の外部割り込みを行った時間，つまり前回の外部割り込みから 100 ミリ秒以上経過しているかの判定をしている．もし 100 ミリ秒以上経過していたら明るさを表示し，処理を行った時間を変数に保持している.

7 感想

Arduino を使用してより利便性のあるシステムを作成することができてよかった．割り込み処理を習得したことにより，作成できるシステムの幅が大幅に広がった．なにか自分が困っていることを解決するシステムを作成してみようと思った.

参考文献

[1]Arduino Team, “attachInterrupt(interrupt, function, mode)”, Arduino 日本語リファレンス, http://www.musashinodenpa.com/arduino/ref/index.php?f=0&pos=3069 (参照日：2021 年 7 月 12 日)

[2]Arduino Team, “attach(pin)”, Arduino 日本語リファレンス, http://www.musashinodenpa.com/arduino/ref/index.php?f=1&pos=679 (参照日：2021 年 7 月 12 日)

[3] © Marutsuelec Co.,Ltd, “スイッチのチャタリングの概要 チャタリングを防止する方法”, マルツエレック株式会社 公式 HP, https://www.marutsu.co.jp/pc/static/large_order/1405_311_ph （参照日：2021 年 7 月 12 日）