マイコン講習
ー第7回ー

K.Miyauchi

2021.12.02

目次

- 電圧レベル
- マイコンで避けたい処理
- マイコンにおける変数のデータ長

本講習で使用するもの

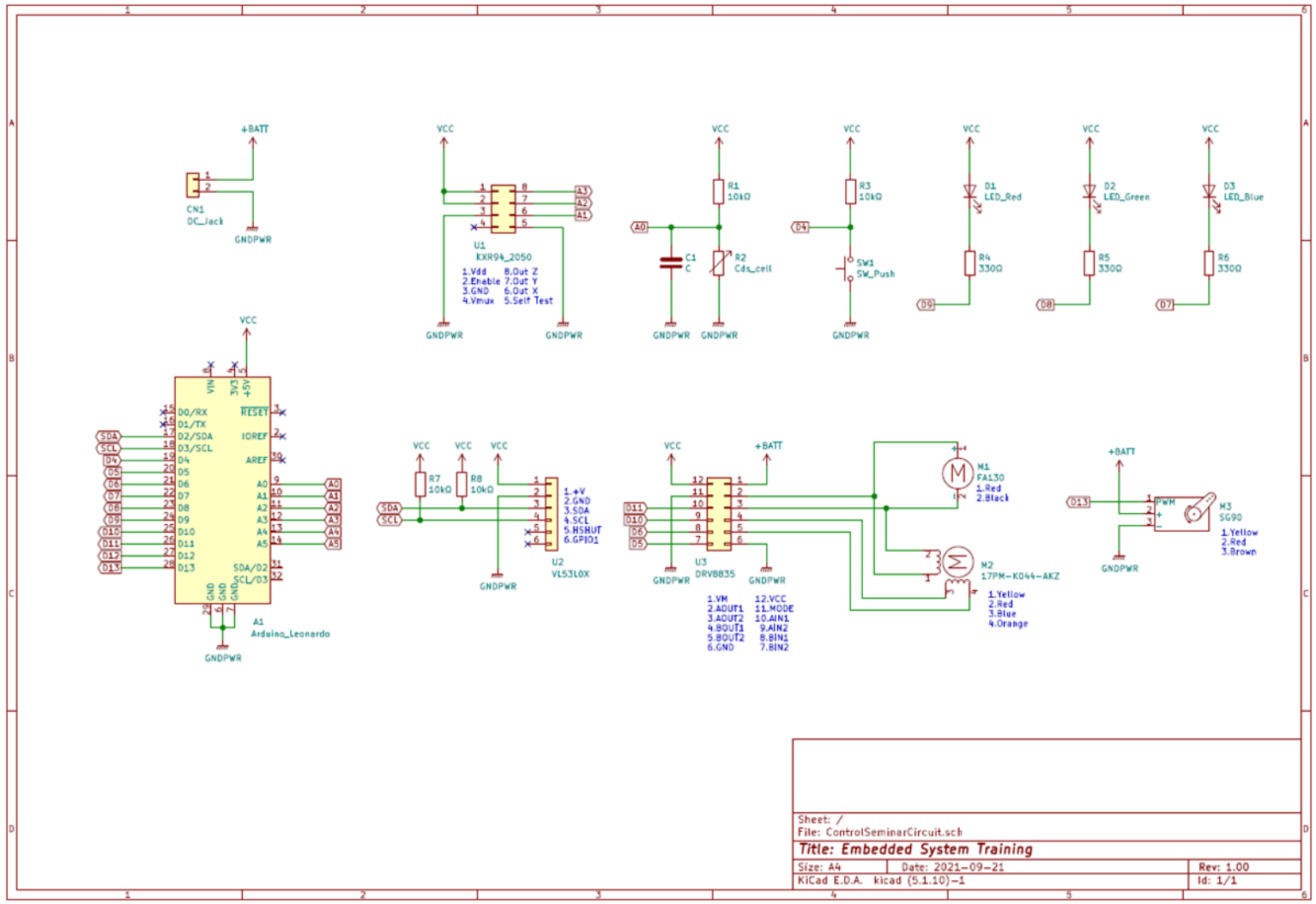
パソコン：Windows, Mac

開発環境：Arduino IDE

マイコン：Arduino Leonardo

その他，回路周りの物品

回路図


+BATT
CN1
DC_Jack
GNDPWR
VCC
U1
KXR94_2050
1.Vdd 8.Out Z
2.Enable 7.Out Y
3.GND 6.Out X
4.Vmux 5.Self Test
R1
10kΩ
C1
C
R2
Cds_cell
R3
10kΩ
SW1
SW_Push
D1
LED_Red
R4
330Ω
D2
LED_Green
R5
330Ω
D3
LED_Blue
R6
330Ω
A1
Arduino_Leonardo
R7
10kΩ
R8
10kΩ
U2
VL53L0X
1.+V
2.GND
3.SDA
4.SCL
5.NSHUT
6.GPIO1
U3
DRV8835
1.VM 12.VCC
2.AOUT1 11.MODE
3.AOUT2 10.AIN1
4.BOUT1 9.AIN2
5.BOUT2 8.BIN1
6.GND 7.BIN2
M1
FA130
1.Red
2.Black
M2
17PM-K044-AKZ
1.Yellow
2.Red
3.Blue
4.Orange
PWM
M3
SG90
1.Yellow
2.Red
3.Brown
Sheet: /
File: ControlSeminarCircuit.sch
Title: Embedded System Training
Size: A4
Date: 2021-09-21
Rev: 1.00
KiCad E.D.A. kicad (5.1.10)-1
Id: 1/1

電圧レベル

電圧レベルとは，マイコンの電源やデジタル信号において，マイコンやIC等が基準とする電圧のことである．

よく使われる電圧レベルとして，

- 3.3V
- 5V
- 12V

がある．Arduinoの電圧レベルは，5Vであるが，最近はやっているSTM32マイコンなどは，3.3Vであるため，使用するときには注意が必要である．

電圧レベル

ただし，マイコンによっては，自身のレベルより高い電圧の入力を許すものもある．このような機能のことをトレラントという．

これは，GPIOごとに機能の有無が異なるため，データシート等で確認が必要である．

マイコンで避けたい処理

マイコンは，PCと異なり，クロックがMHz単位とPCと比べて，10~1000倍近く処理速度が遅い．

また，保有している演算ユニットなどが異なるため，演算などは慎重に考えなければならない．

RAMについても，数百kBから数MBのものが多いため，変数の寿命などについて考慮する場面が多々ある．

今回は，特にマイコンで避けたい処理について紹介する．

マイコンで避けたい処理

「浮動小数点の演算」

　マイコンは浮動小数点用の演算器をもってないものが多く，整数用演算器で計算する．

浮動小数点の計算をした場合，
整数の演算にかかる時間に比べて，
数十倍の時間がかかる（私の経験的に）．

たとえ，浮動小数点用演算器（FPU）を搭載しているマイコンでも，
整数より時間が演算時間が長いので，できるだけ避ける．

対処方法：
整数で100倍で計算した後に100で割るみたいなことをする．

マイコンで避けたい処理

実際にどのくらい異なるのか確かめてみましょう.

```
#define CALC_TIMES 1000

void setup(){
  Serial.begin(9600);

  unsigned int startTime;
  unsigned int endTime;

  float a = 12.4141241231;
  float b = 1.24154123312;
  volatile float workf;
  startTime = micros();
  for(int i=0; i < CALC_TIMES; i++){
    workf += a/b;
  }
  endTime = micros();
  Serial.println(endTime - startTime);

  int A = 124141241231;
  int B = 124;
  volatile int worki;
  startTime = micros();
  for(int i=0; i < CALC_TIMES; i++){
    worki += A/B;
  }
  endTime = micros();
  Serial.println(endTime - startTime);
}

void loop(){

}
```

マイコンで避けたい処理

・micros()
起動してからの時間取得関数
返り値　　：起動してからの時間[us]

・millis()
起動してからの時間取得関数
返り値　　：起動してからの時間[ms]

マイコンで避けたい処理

「浮動小数点の演算」

```
// 避けたい例
float a=13.00;
float b=5;
float c = a/b;

// 対処例
int a=10000;       // 1000倍したのもの
int b=5;
int c=a/b/1000;    // 小数点以下は切り捨て
```

マイコンで避けたい処理

「メモリの動的確保」

　配列等を使用するときに，動的配列（C言語でいえばmalloc，C++でいえばvectorなど）の使用は控える．ものすごく、処理時間が長いです．使用しただけで，発熱により破損するマイコンもあるぐらい負荷が大きい．

対処方法：
静的配列でRAMの無駄遣いを覚悟の上で最大で使用する要素数だけ最初から用意する．

マイコンで避けたい処理

「メモリの動的確保」

```
// 避けたい例
vector<int> vec1(100,0);    // 配列宣言
vec1.resize(200);                // 要素数の変更

// 対処例
int lis[200];      // 最初に最大必要数を宣言する
```

マイコンで避けたい処理

「定数の変数での使用」

　マイコンはRAMサイズがとても小さい.
そのため，定数は変数での使用は避ける.

対処方法：
コードにそのまま定数をその都度書くのでもよいが，
とてもとても，めんどくさい，かつ，わからなくなる可能性が大のため，
C言語ではプリプロセッサ処理の#defineを用いた方法とconst修飾子を用いた方法がある.
C++では，constexpr修飾子を用いた方法があります.
（C++では，プリプロセッサ処理を避けましょうという風潮があるので，
#defineは基本的に使わない.）

マイコンで避けたい処理

「定数の変数での使用」

```
//避けたい例
int PinA=1;
int PinB=2;

// 対処例
#define PinA 1                  // マクロ定義による対処
const int PinB=2;               // 定数修飾子による対処
constexpr int PinC=3;           // コンパイル時定数による対処（C++）
```

マイコンで避けたい処理

「割り込み処理での過負荷処理」

　割り込み処理では，重い処理はやってはならない．
理由としては，
　・割り込みが終了する前に再度割り込みが発生する可能性がある
　・メイン処理が長い時間停止する

対処方法：
割り込み処理ありきで開発しない．
サブ的なものだと思ってシステム設計する．

マイコンにおける変数のデータ長

C言語において，int型であれば，4 byteなどといったように書籍などで書かれていることが多い.

しかし，使用する環境によっては，int型は2 byteであるときもある.

これは，int型の規定として，プロセッサのワード長に対応したbit数であるためである.

そのため，int型，long型，short型などの整数型においては，使用する前に，データサイズをsizeofなどを使用して，確認する必要がある.

マイコンにおける変数のデータ長

しかし，その都度，調べるのは手間がかかる．

そのため，以下のようなデータ型を扱うことが多い．
（特にマイコンでは多い）

型名	サイズ	型名	サイズ
int8_t	符号あり整数 8bit	int32_t	符号あり整数 32bit
uint8_t	符号なし整数 8bit	uint32_t	符号なし整数 32bit
int16_t	符号あり整数 16bit	int64_t	符号あり整数 64bit
uint16_t	符号なし整数 16bit	uint64_t	符号なし整数 64bit