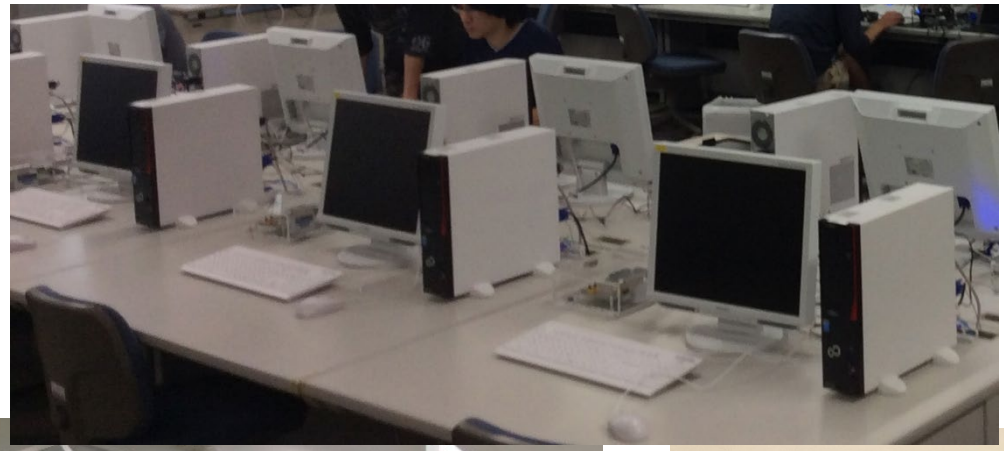
RITSUMEIKAN

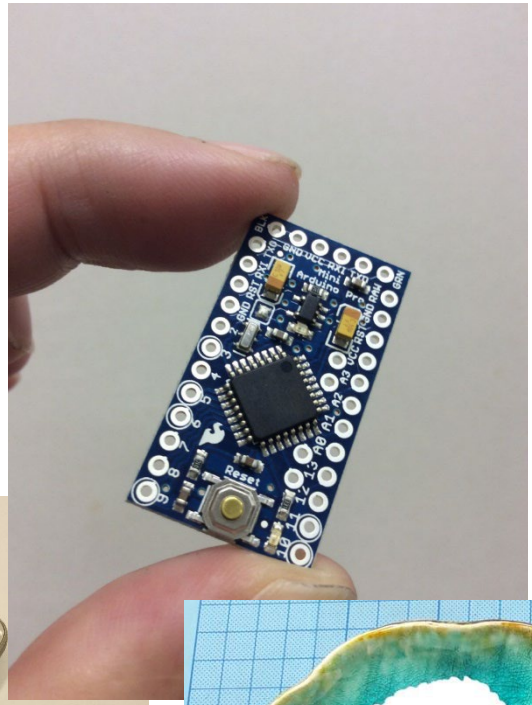
システム設計CAD
マイコンプログラミング編
(1) 概要

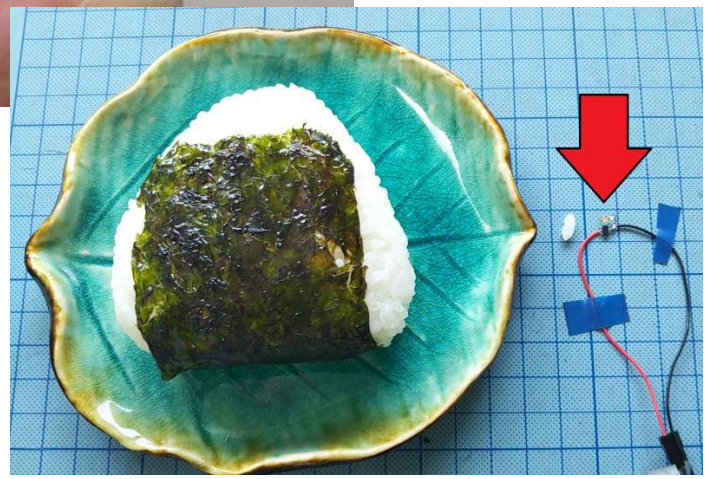
立命館大学 理工学部 電子情報工学科
泉　知論　　田中　亜実

http://www.ritsumei.ac.jp/se/re/izumilab/lecture/23cad/

大きなコンピューター　小さなコンピューター







小さいけれど　コンピューター

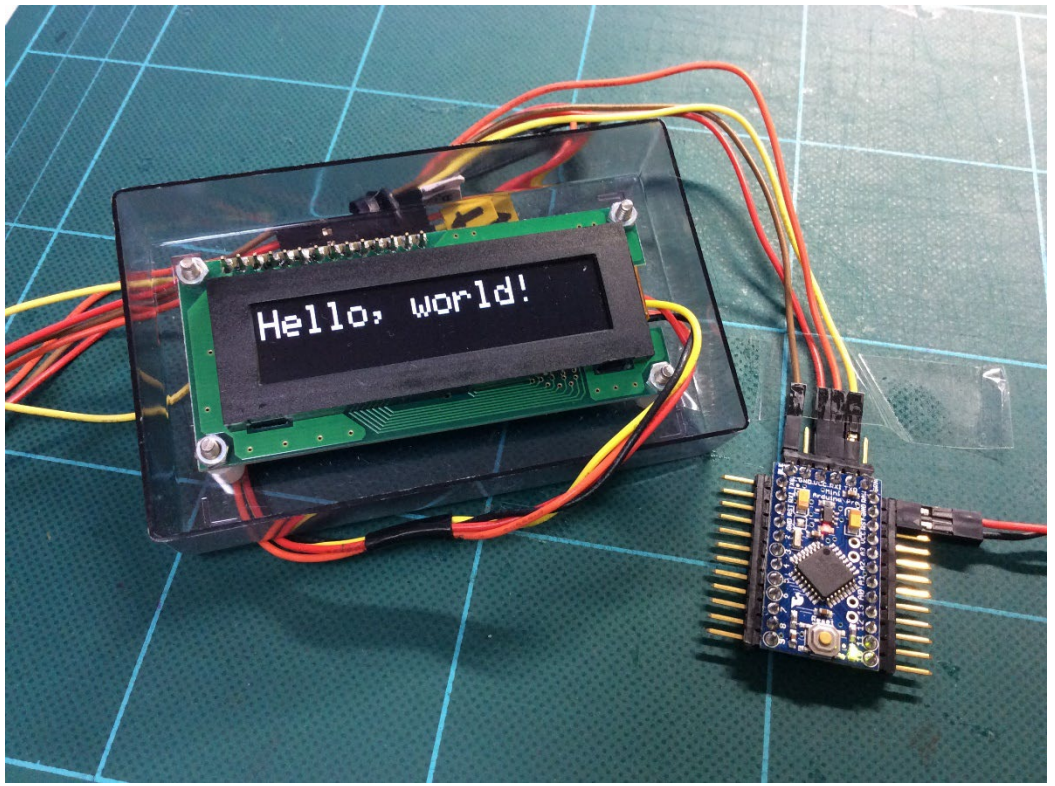
hello | Arduino 1.6.8

ファイル　編集　スケッチ　ツール　ヘルプ

hello §

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

void loop() {
  Serial.println("Hello, world!");
  delay(1000);
}
```



米粒より小さいけれど コンピューター

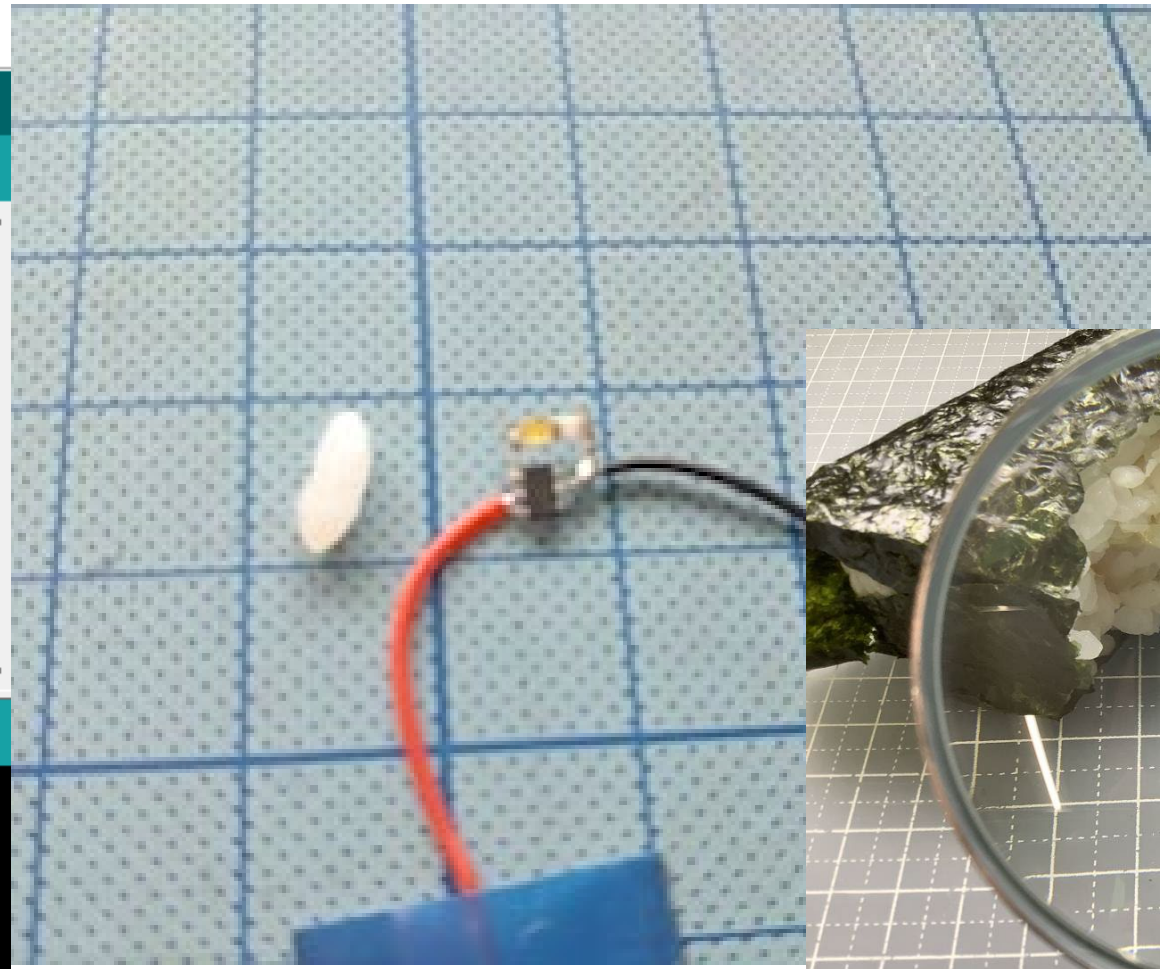
led | Arduino 1.8.9

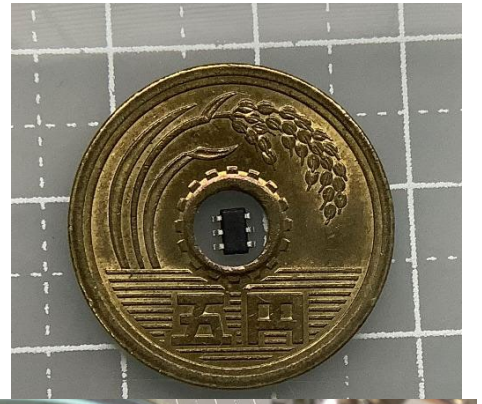
ファイル 編集 スケッチ ツール ヘルプ

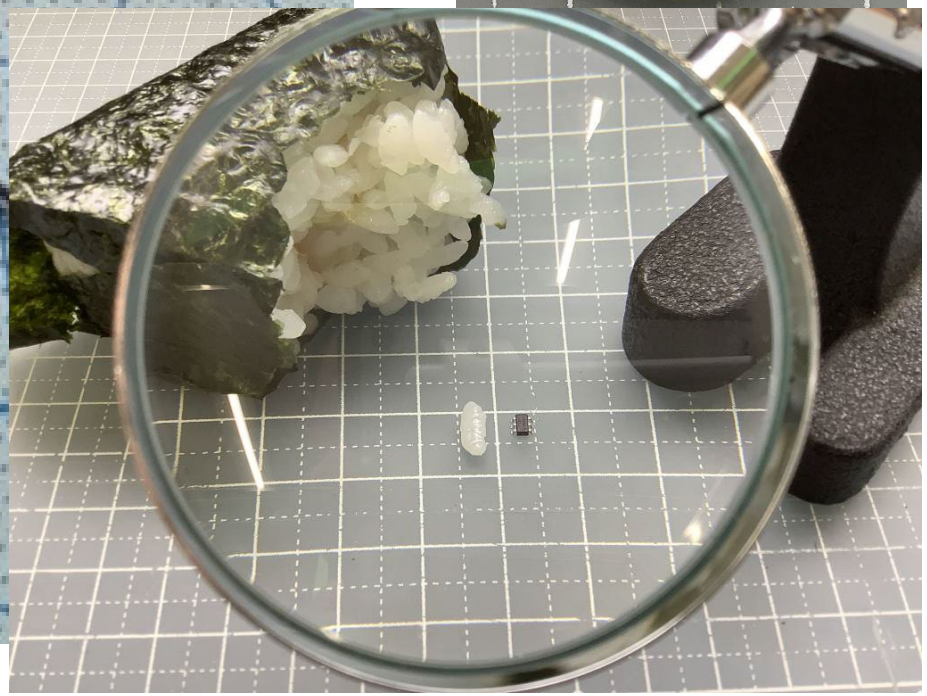
led

```
1 void setup() {
2   pinMode(PB2,OUTPUT);
3 }
4 void loop() {
5   digitalWrite(PB2,HIGH);
6   delay(500);
7   digitalWrite(PB2,LOW);
8   delay(500);
9 }
```

COM16のATtiny10 (bitDuino10), 1MHz(Internal)







「マイコン」と組込みシステム

- 高性能は良いことか？
  - カーナビに地球気候シミュレーションができるだけの計算能力は必要か？
  - テレビのリモコンに高精細画像処理能力は必要か？
- 大きさ、重さ、消費E
- 価格
- <u>必要充分な性能</u>

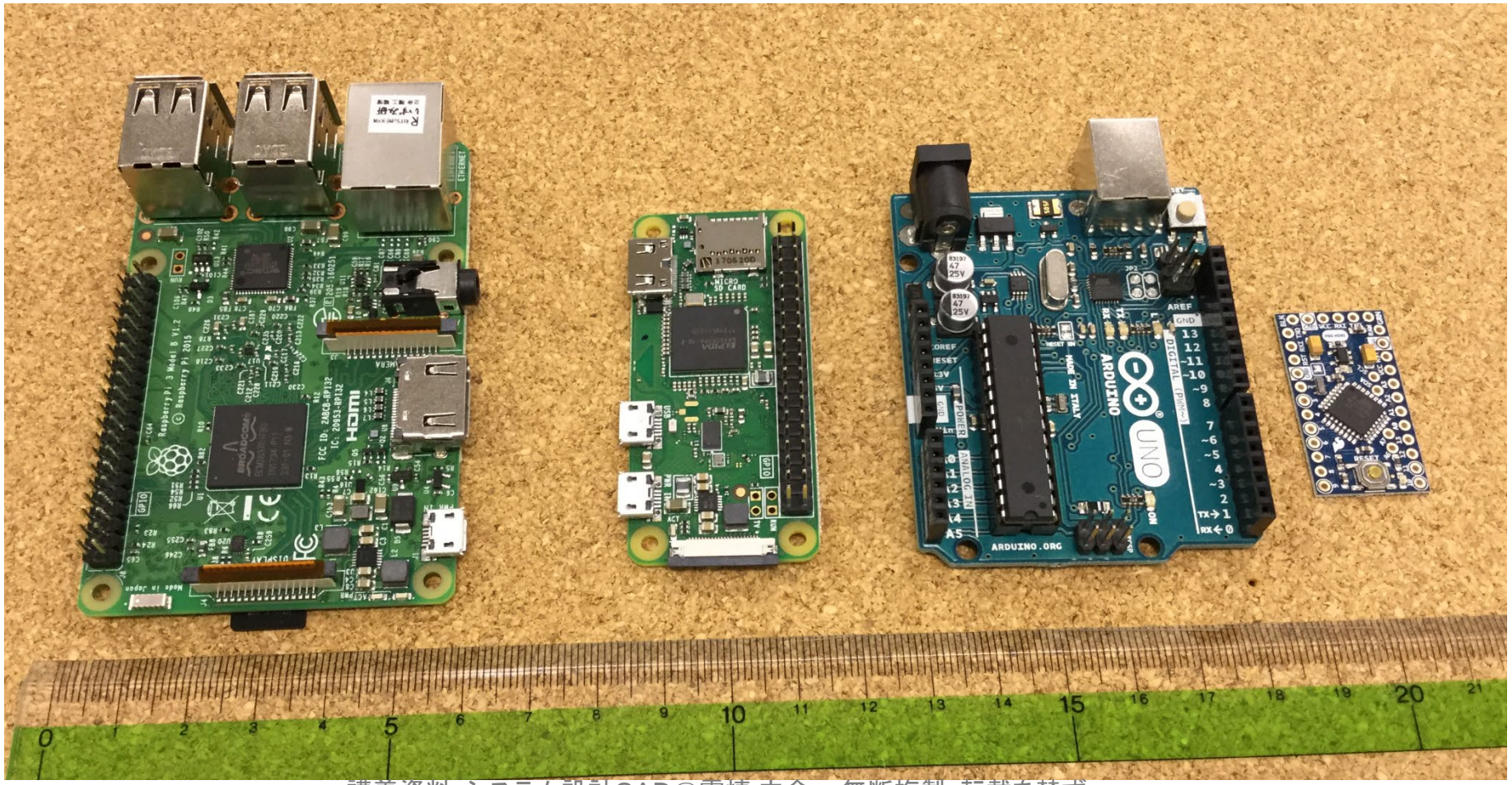
- マイコン＝？
  - micro controller
  - micro computer
  - micro processor

マイコンと性能

- Processor
  - 演算ビット幅、動作速度、アドレス幅
  - 命令セット、動作モード、割込、例外
- Memory System
  - 容量、バス速度
  - Cache, MMU
- I/O peripherals
  - GPIO, TIMER, COUNTER, ADC, DAC, PWM
  - UART, I2C, SPI, USB, SD, Ether, …
  - PCIe
  - 内蔵 v.s. 外付
- Operating System
  - Unix系/Windows系/Apple系…
  - 組込み linux
  - RTOS
  - baremetal, standalone
- Middleware
  - Android, ROS, ….

広く普及しているマイコンボードの例

Raspberry Pi　　Arduino



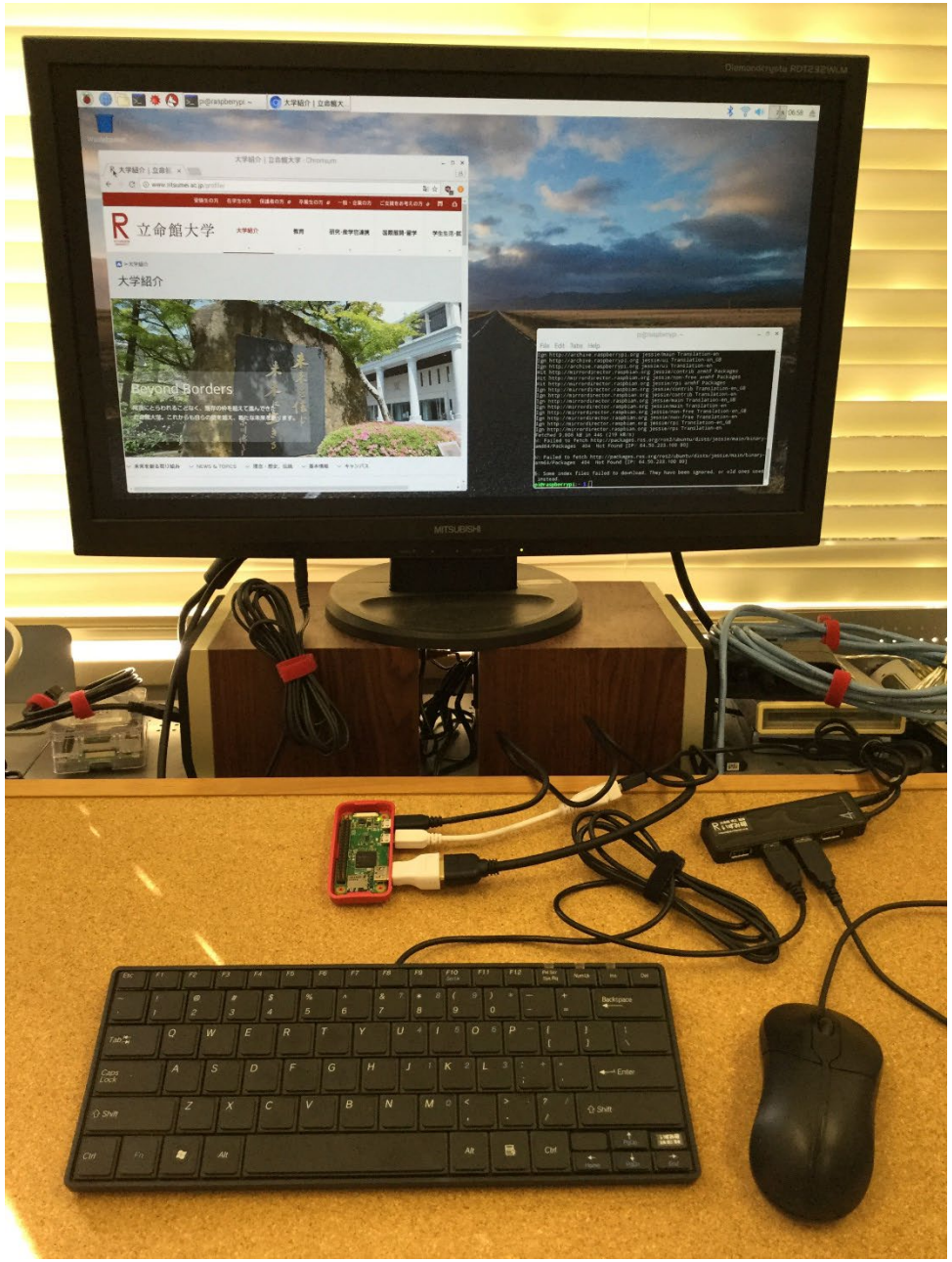
Raspberry Pi

- ARM Coretex-A53 Quad Core等
- 32bit or 64bit, 数百M～1GHz以上
- +GPU
- 数百M～数GB DDR2 memory
- 数W～十数W
- USB, HDMI, SD
- GPIO
- Linux, Windows 等

スマホ、カーナビなどと同じくらいのレベル

Raspberry Pi！

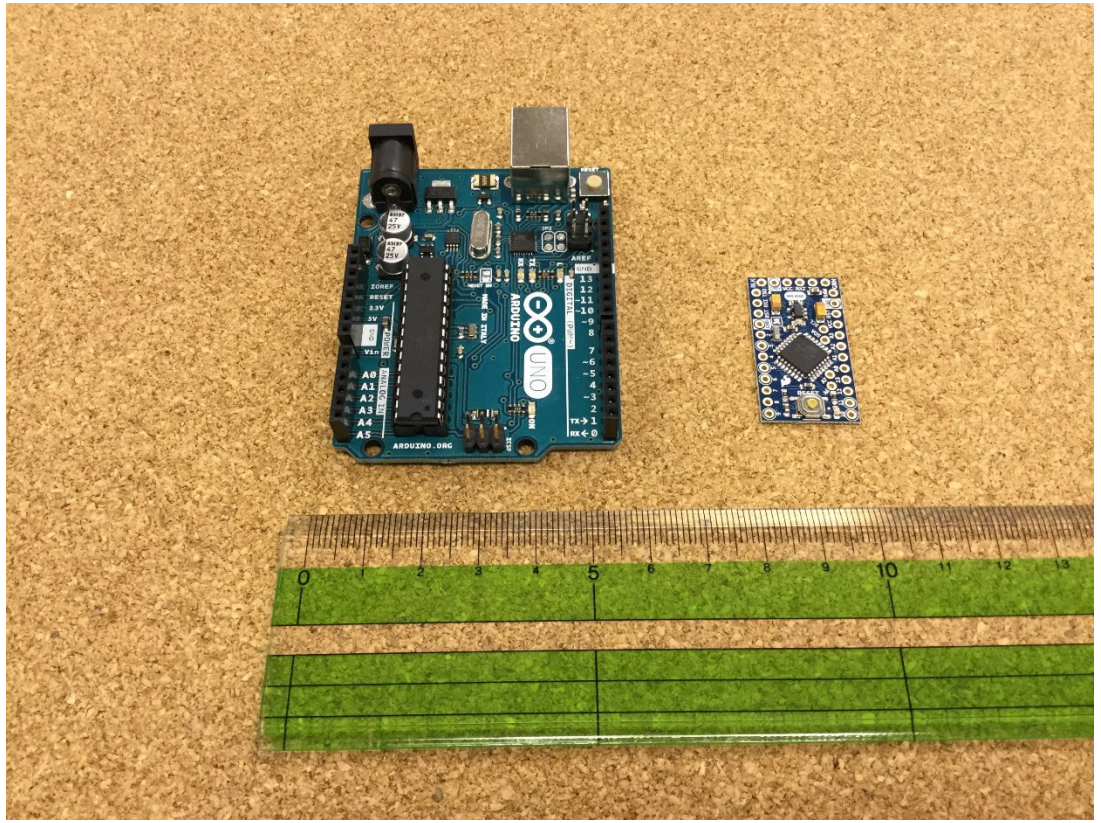
こんなに小さくても
モニタをつないで (HDMI)
キーボードとマウスをつないで (USB)
ネットワークにつなぐと (WiFi)
普通にコンピューターに見える



Arduino

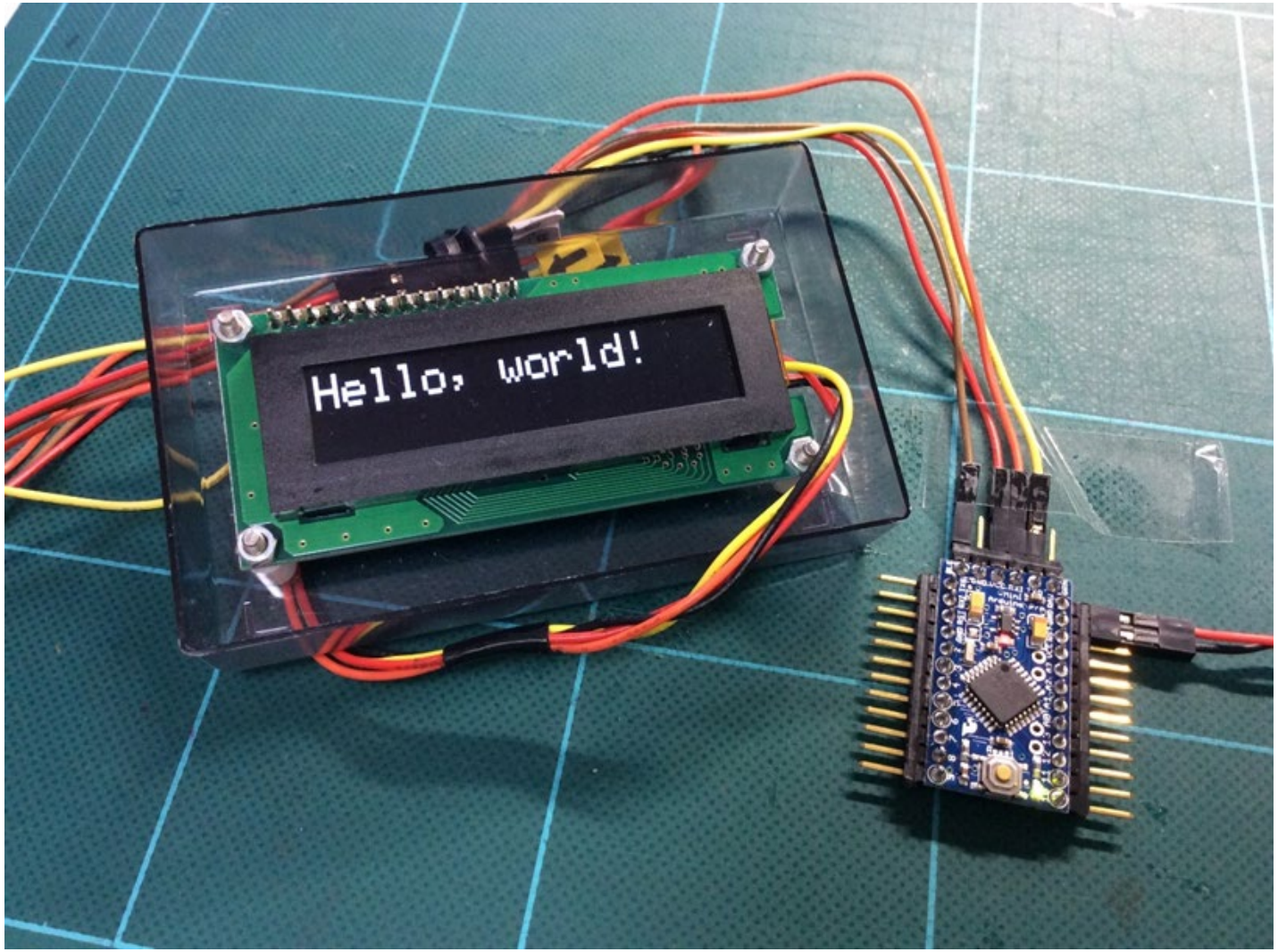
- AVR ATmega328等
- 8bit～, 8MHz～
- 数十kB 不揮発, 数kB SRAM
- 数mW～
- GPIO, ADC, PWM, UART, SPI, …
- 全てが1チップ！
- Arduino IDE (≒baremetal)

家電のリモコン、ロボットの各関節などと同じくらいのレベル

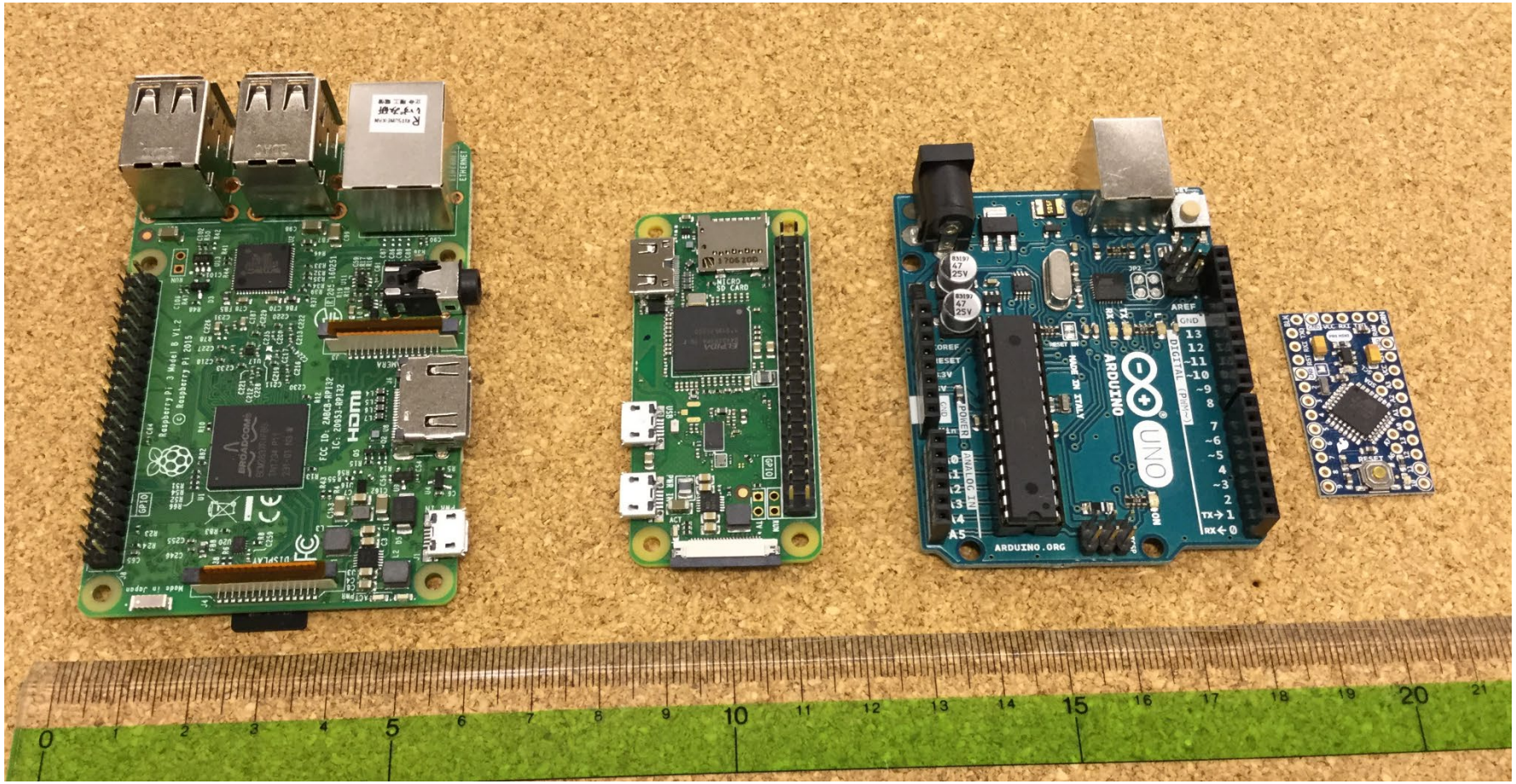


Arduino!

不愛想だけど
Hello, world! するので
コンピューターらしい



? Raspberry Pi v.s. Arduino ?



? Raspberry Pi v.s. Arduino ?

性能の差は電池の差でもある

性能∝電池の大きさ･価格

? Raspberry Pi v.s. Arduino ?

今回はこちら

- Raspberry Pi
  - ARM Coretex-A53 Quad Core等
  - 32bit or 64bit, 数百M～1GHz以上
  - +GPU
  - 数百M～1GB DDR2 memory
  - 数W～十数W
  - USB, HDMI, SD
  - GPIO
  - Linux, Windows 等

スマホ、カーナビなどと同じくらいのレベル

- Arduino
  - AVR ATmega328等
  - 8bit～, 8MHz～
  - 数十kB 不揮発, 数kB SRAM
  - 数mW～
  - GPIO, ADC, PWM, UART, SPI, …
  - 全てが1チップ！
  - Arduino IDE (≒baremetal)

家電のリモコン、ロボットの各関節などと同じくらいのレベル

データシートを読もう！

ATmega328 datasheet ［検索］

ATmega328P

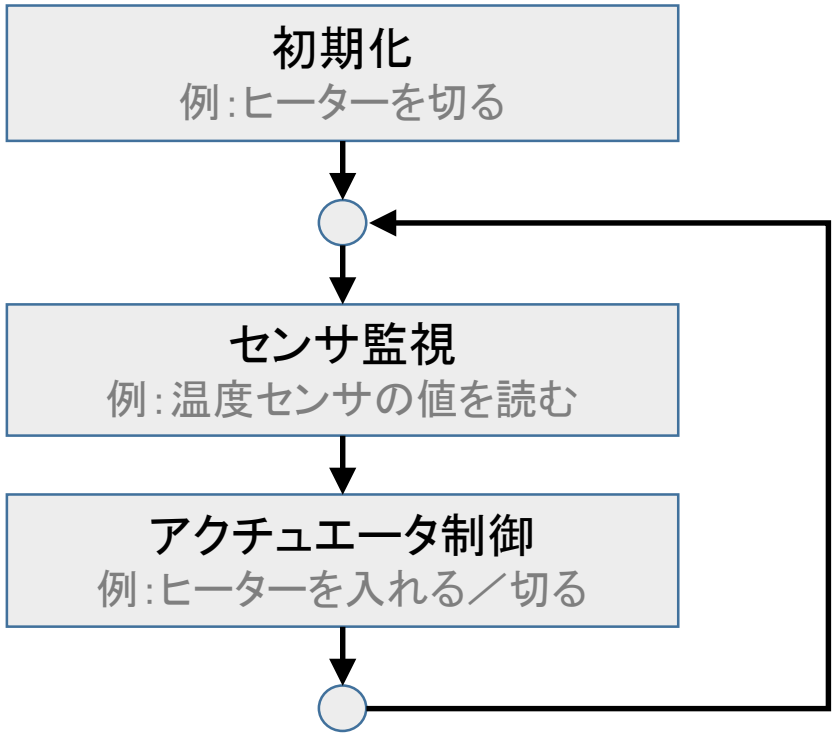
- 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash
- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
- 32 x 8 general purpose working registers
- Up to 16MIPS throughput at 16MHz
- On-chip 2-cycle multiplier
- 32K bytes of in-system self-programmable flash program memory
- 1Kbytes EEPROM, 2Kbytes internal SRAM
- Two 8-bit Timer/Counters, One 16-bit Timer/Counter
- Six PWM channels
- 8-channel 10-bit ADC
- Temperature measurement
- USART, SPI, I2C
- Programmable watchdog timer
- On-chip analog comparator
- Interrupt and wake-up on pin change
- Internal calibrated oscillator
- External and internal interrupt sources
- Six sleep modes

システム設計CAD
マイコンプログラミング編
(2) Arduinoプログラミング基礎

立命館大学 理工学部 電子情報工学科
泉　知論　　田中　亜実

制御用プログラムの基本形

- 対象をずっと監視して制御する



- Arduino IDE では↓雛形に従う

```
void setup() {
    /* 初期化処理 */
}


void loop() {
    /* ループ内の処理 */


}
```

Arduino のプログラムの開発と実行

- Arduino IDE を起動
- プログラムを書く
- プログラムを保存
- プログラムをコンパイル
- Arduinoを接続
- Arduinoに書き込む
- Arduino上で実行

✓開発用のPCが必要（クロスコンパイル）

✓プログラムが書き込んであれば、電源を入れると自動実行する（PCが無くても Arduino 単体で実行可能）

✓詳細は自身でマニュアルを参照すべし（ヘルプ→リファレンス）

演習：何はともあれ「Lチカ」

Arduinoだけでできる！
（ブレッドボード不要）

Arduino上のLEDを点滅させる。LEDは D13 ピンに接続されている。周期は1秒（点灯500msec、消灯500msec）とする。

- Arduino IDE を起動
- ファイル→新規ファイル
- プログラムを作成
- 名前を付けて保存
  - 演習保存用のドライブか持参USBメモリに保存すること
  - 付けた名前のフォルダが作成される
- （✔）検証（コンパイル）

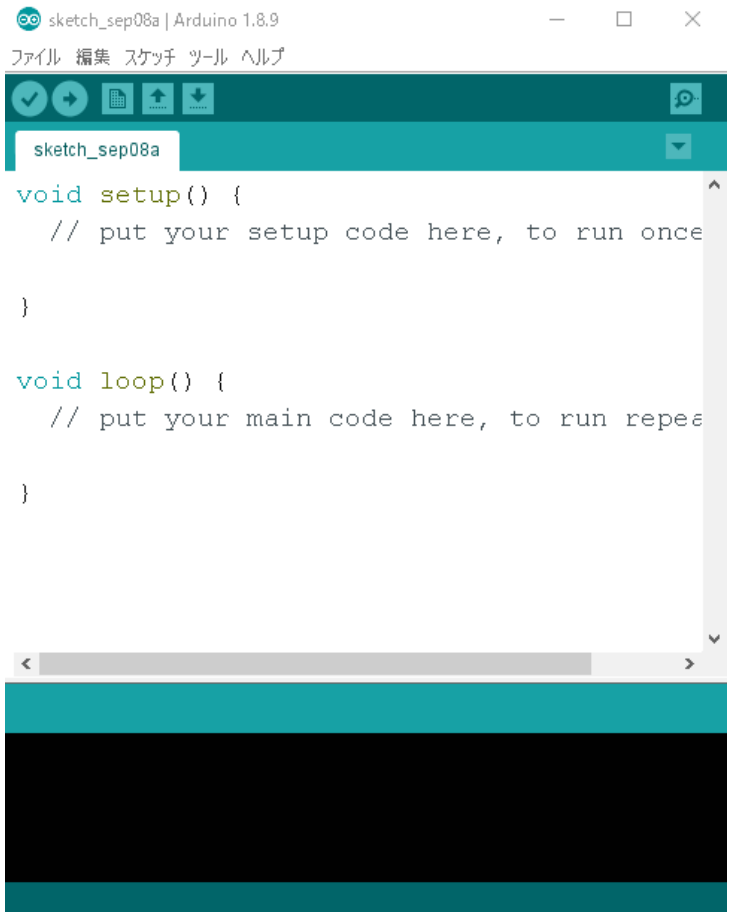
```
#define LEDPIN 13
#define PERIOD 1000

void setup() {
  pinMode( LEDPIN, OUTPUT );
}

void loop() {
  digitalWrite( LEDPIN, HIGH );
  delay( PERIOD / 2 );
  digitalWrite( LEDPIN, LOW );
  delay( PERIOD / 2 );
}
```

接続と起動

- ArduinoとPCをUSBケーブルで接続する
- Arduino IDE を起動する



新規プログラムの作成と保存

- 「ファイル」→「新規ファイル」で新規プログラムファイルを作成する
- 「ファイル」→「名前を付けて保存」で保存したい場所・名前でいったん保存する

プログラムの作成

- プログラムを作成する

led0 | Arduino 1.8.9

ファイル 編集 スケッチ ツール ヘルプ

led0 §

```
#define LED0 13      /* D13 pin */
#define PERIOD 1000 /* 1000msec */

void setup() {
  pinMode(LED0,OUTPUT);
}

void loop() {
  digitalWrite(LED0,HIGH);
  delay(PERIOD/2);
  digitalWrite(LED0,LOW);
  delay(PERIOD/2);
}
```

保存しました。

7 COM10のArduino/Genuino Uno

ボードの設定

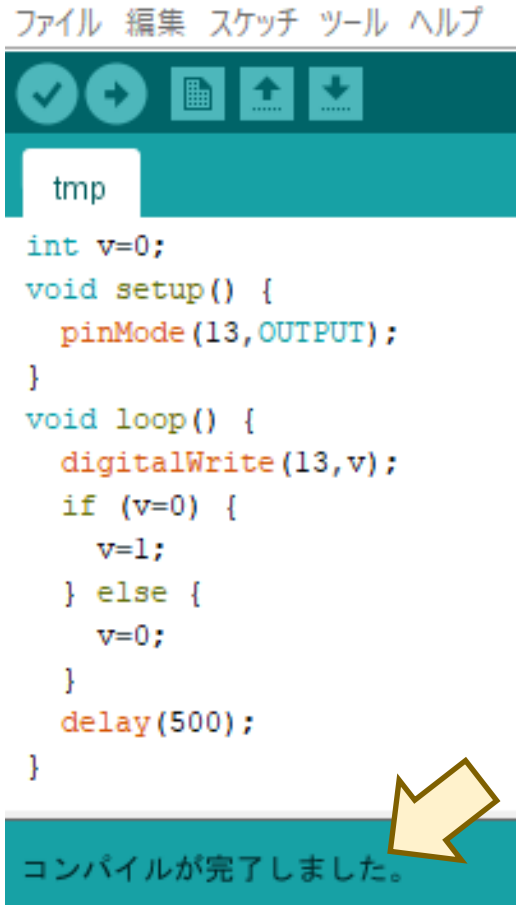
- 「ツール」→「ボード」で“Arduino/Genuino Uno”を選択
- 「ツール」→「シリアルポート」で“Arduino/Genuino Uno”と示されているポートを選択

コンパイル、書込、実行

- (✔) 検証（コンパイル）
- (➡) マイコンボードに書き込む

- １秒周期の点滅動作を確認する

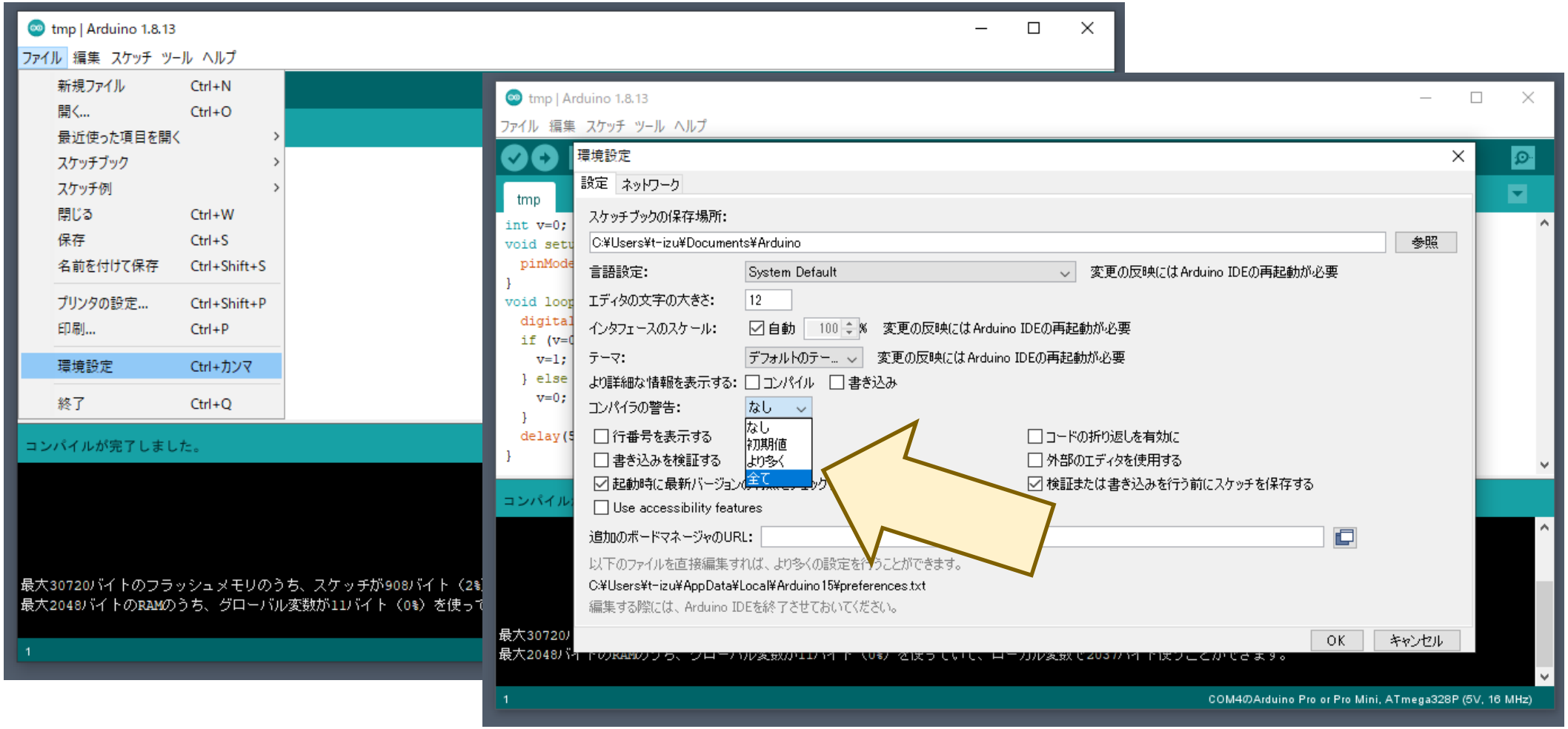
コンパイラの警告メッセージについて



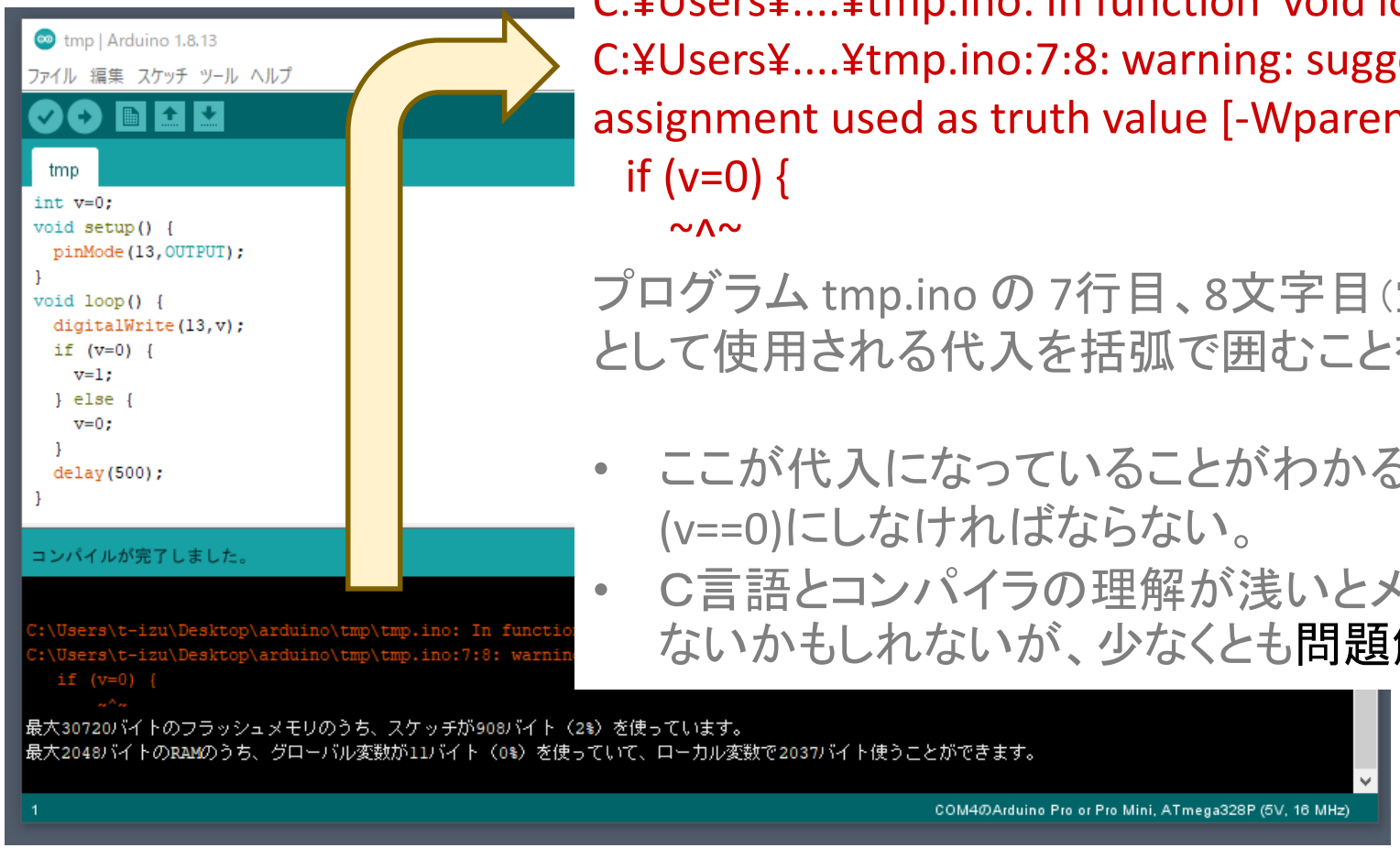
- LED点滅のつもりだけど、点滅しない
- コンパイルしてもエラーはない
- どこが間違っているかわかる？

- 初期状態では警告は出さないようになっている
- プログラミングに**自信がない人こそ**警告を読むべし！

警告は全て出す！


tmp | Arduino 1.8.13
ファイル 編集 スケッチ ツール ヘルプ
新規ファイル Ctrl+N
開く... Ctrl+O
最近使った項目を開く
スケッチブック
スケッチ例
閉じる Ctrl+W
保存 Ctrl+S
名前を付けて保存 Ctrl+Shift+S
プリンタの設定... Ctrl+Shift+P
印刷... Ctrl+P
環境設定 Ctrl+カンマ
終了 Ctrl+Q
コンパイルが完了しました。
最大30720バイトのフラッシュメモリのうち、スケッチが908バイト（2%）
最大2048バイトのRAMのうち、グローバル変数が11バイト（0%）を使って
環境設定
設定 ネットワーク
スケッチブックの保存場所:
C:¥Users¥t-izu¥Documents¥Arduino
参照
言語設定:
System Default
変更の反映にはArduino IDEの再起動が必要
エディタの文字の大きさ:
12
インタフェースのスケール:
自動
100
%
変更の反映にはArduino IDEの再起動が必要
テーマ:
デフォルトのテー...
変更の反映にはArduino IDEの再起動が必要
より詳細な情報を表示する:
コンパイル
書き込み
コンパイラの警告:
なし
なし
初期値
より多く
全て
行番号を表示する
書き込みを検証する
起動時に最新バージョンの
Use accessibility features
コードの折り返しを有効に
外部のエディタを使用する
検証または書き込みを行う前にスケッチを保存する
追加のボードマネージャのURL:
以下のファイルを直接編集すれば、より多くの設定を行うことができます。
C:¥Users¥t-izu¥AppData¥Local¥Arduino15¥preferences.txt
編集する際には、Arduino IDEを終了させておいてください。
OK
キャンセル
COM4のArduino Pro or Pro Mini, ATmega328P (5V, 16 MHz)

警告を読む！



C:¥Users¥....¥tmp.ino: In function 'void loop()':
C:¥Users¥....¥tmp.ino:7:8: warning: suggest parentheses around assignment used as truth value [-Wparentheses]
  if (v=0) {
     ~^~

プログラム tmp.ino の 7行目、8文字目（空白も数えて）：警告：真理値として使用される代入を括弧で囲むことを提案する

- ここが代入になっていることがわかる。代入(v=0)ではなく比較(v==0)にしなければならない。
- C言語とコンパイラの理解が浅いとメッセージの意味はわからないかもしれないが、少なくとも**問題解決のヒント**にはなる。

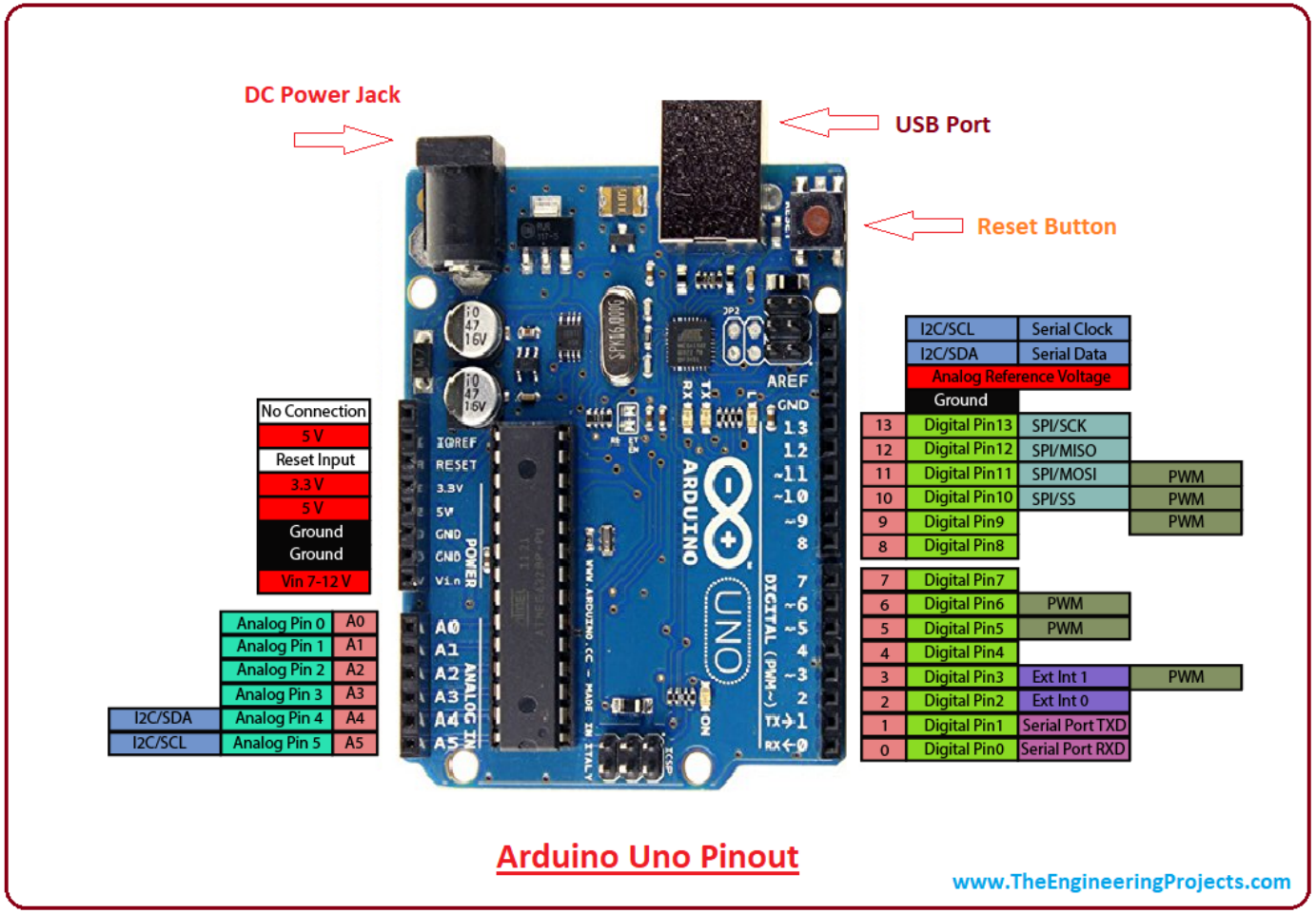
システム設計CAD
マイコンプログラミング編
(3) マイコン入出力回路の原理

立命館大学 理工学部 電子情報工学科

泉　知論　　田中　亜実

LEDだけ？　自分で追加できる！

- Arduino（というかこのマイコンチップ）は、様々な入出力がつなげられるように設計されている
- Analog 6pin, Digital 14pin, +α
- ピン数が限られる（価格）ため、様々な機能がピンを共有している
- 詳細はマニュアル参照



マイコンの入出力ピン

- 電位で値を表す
- 電流はほぼゼロ

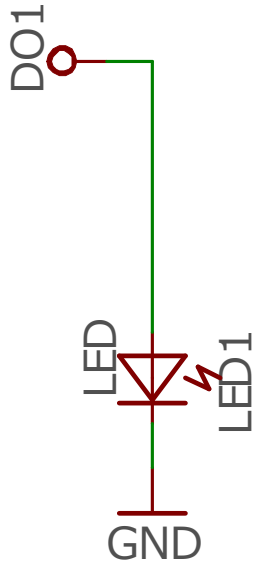
「電圧」と「電位」の違いを理解していますよね？

原則、流れない／流さない

- デジタル出力
  - 0 → GND (0V)
  - 1 → VCC (5V または 3.3V)
- デジタル入力
  - 0 ← GND ($\frac{GND+VCC}{2}$ V を十分下回る)
  - 1 ← VCC ($\frac{GND+VCC}{2}$ V を十分上回る)

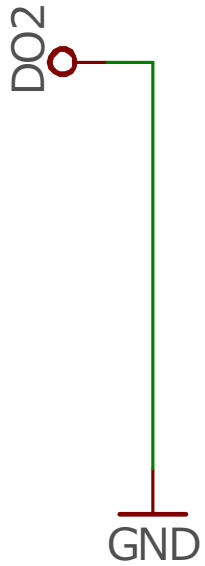
- アナログ入力
  - GNDからVCCの間の電位を
  - 0 から 2^N-1 のデジタル値に変換する
- アナログ出力
  - デジタル値を電位に変換(基本)
  - あるいは…(後述)

デジタル出力回路～直結



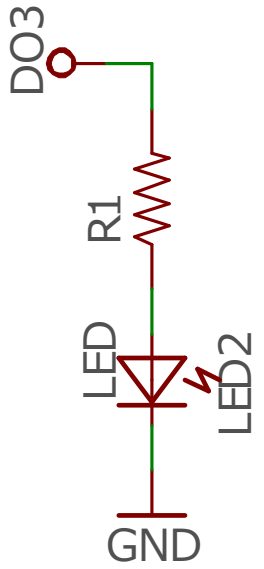
- D out に直結
- タフなマイコンで頑丈なLEDなら
  これで光るかも知れないけど…

デジタル出力直結の問題



- これは明らかにダメ
- 出力をHigh(5Vや3.3V)にしたら電流が流れ過ぎる
- 壊れるかも
- 前ページの直結は大丈夫？
- 出力端子の電圧や内部抵抗、接続したデバイスの抵抗や電圧降下しだい

デジタル出力回路～電流制限抵抗付き

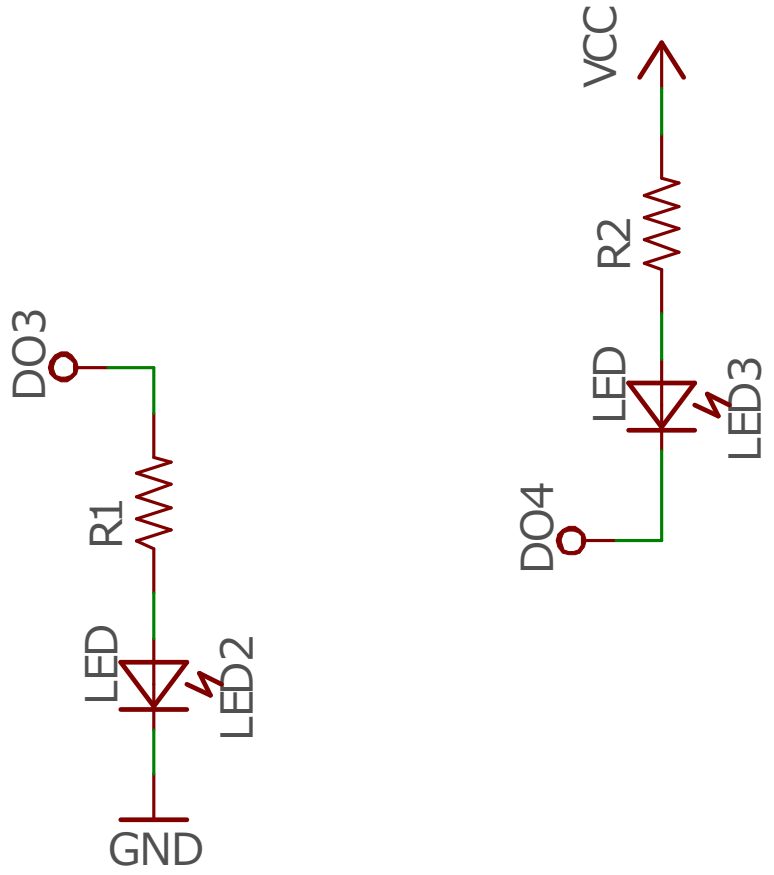


- 電流を適切に制限すべき
- 抵抗を挟めばよい
- 適切な抵抗値は端子の出力電圧や接続素子の電圧降下から算出する

  例：LEDの電圧降下が2.0Vで、推奨電流が10mA、出力電圧が5.0Vなら

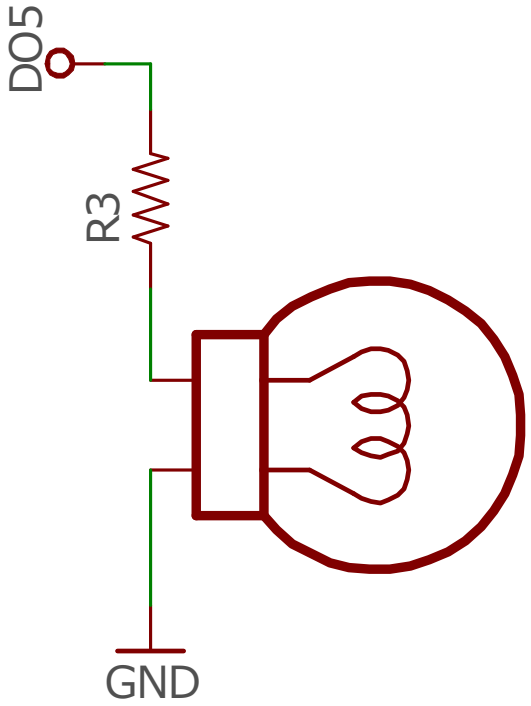
  $R = \frac{5.0\mathrm{V} - 2.0\mathrm{V}}{10\mathrm{mA}} = 300\Omega$ くらい

デジタル出力回路～直接駆動



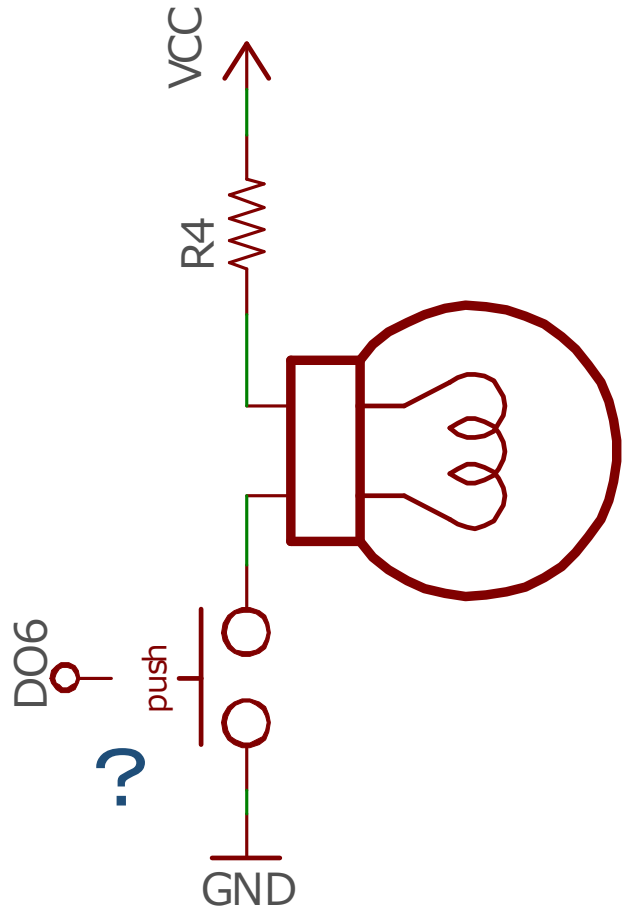
- マイコンでLEDを直接駆動する
- 正論理・負論理
  - 左：1 (High) 出力で光る
  - 右：0 (Low) 出力で光る
- 負論理が定番
  - 電源をマイコン外から取れる
- タフなマイコンならこれでOK…だけど…

デジタル出力～直接駆動？できない？



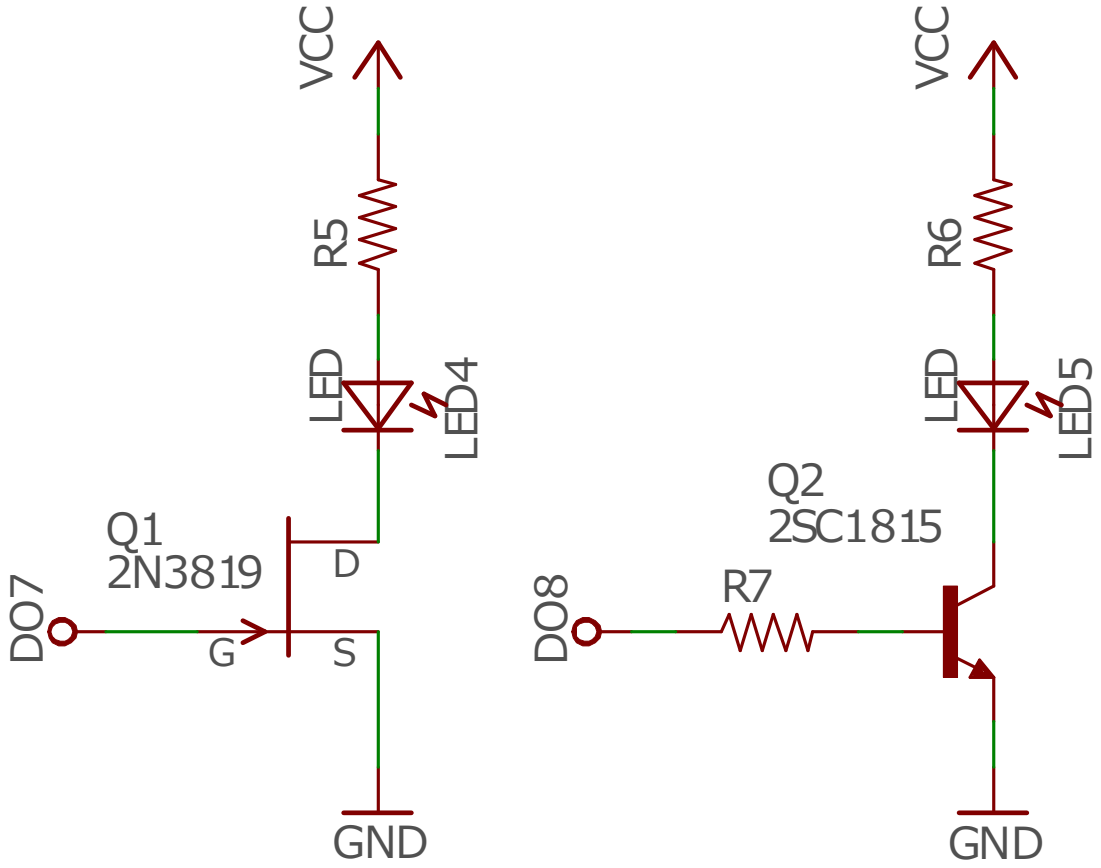
- これで光るか？
- 入出力端子に流せる電流には限度がある
- マイコンの仕様を超えた電流は流せない
- 端子の仕様に依らず駆動するには…

デジタル出力～間接駆動？



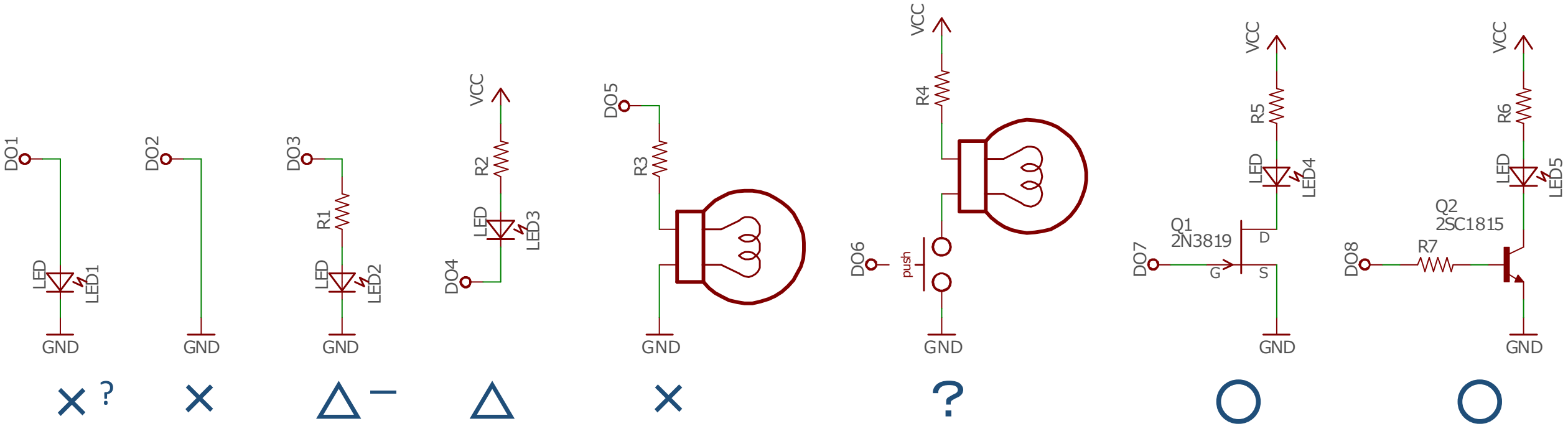
- マイコンの電流で駆動するのではなく、スイッチを介して駆動する
  - 対象は別の電源ラインで駆動
  - スイッチで制御
  - マイコンはスイッチを駆動
- スイッチ？
  - リレー
  - トランジスタ

デジタル出力回路～トランジスタで駆動

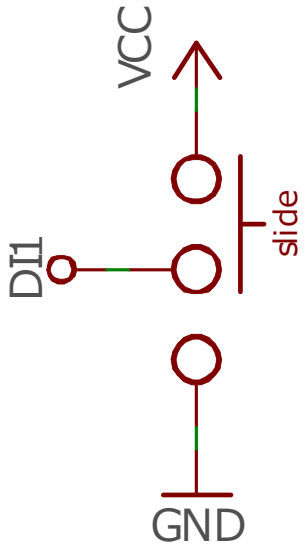


- トランジスタを介して対象を駆動
- 対象に応じてトランジスタや周辺素子を選んで回路を構成
  - FET, bi-polar 等
  - バイポーラの場合は電流調整の抵抗(R7)が必要
  - 必要なら2段にする

結論：デジタル出力回路の構成例

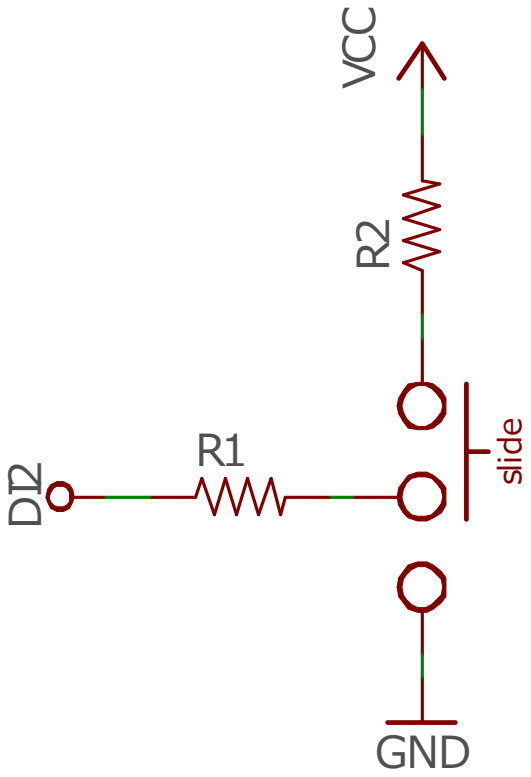


デジタル入力回路～スライドスイッチ直結



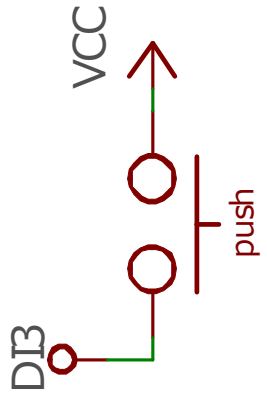
- スライドで接続を切り替える
  - 1 (High, VCC)
  - 0 (Low, GND)
- 入力端子には電流はほとんど流れない
- 素直に回路を組んだらこうなる
- リスクを考えると…
  - 端子が出力に設定されたら…
  - スイッチがショートしたら…
  - ピン配置を間違えたら…

デジタル入力回路～スライドスイッチ



- 安全のため、電流制限の抵抗を入れておく
- 入力端子には電流はほとんど流れないので、抵抗の影響はほぼなし

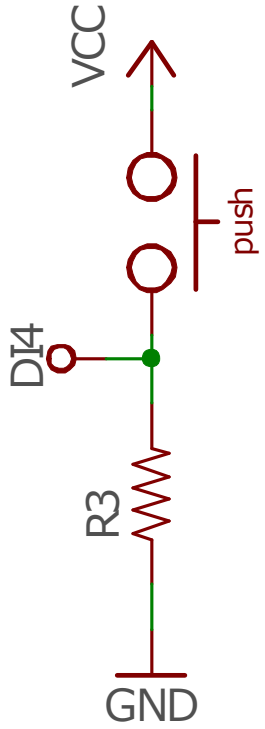
デジタル入力回路～プッシュスイッチ直結



- 押した状態で 1(High, VCC)
- 放した状態では？？？？
  いわゆる「浮いた」状態

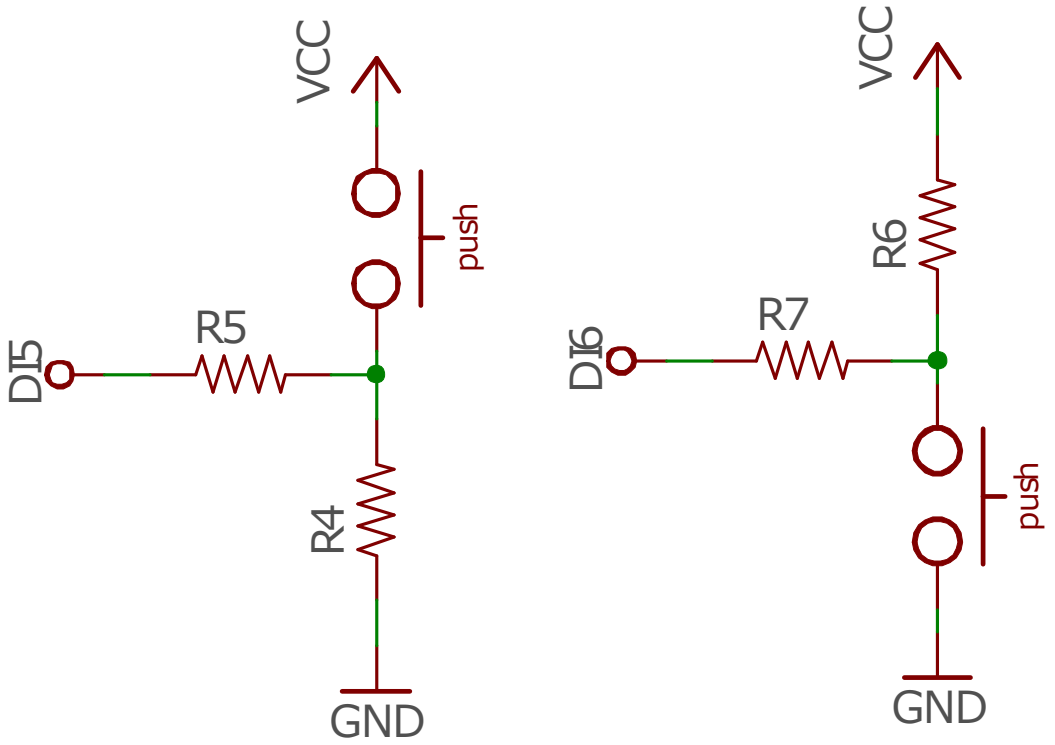
- この回路ではダメ

デジタル入力回路～プルダウン付き



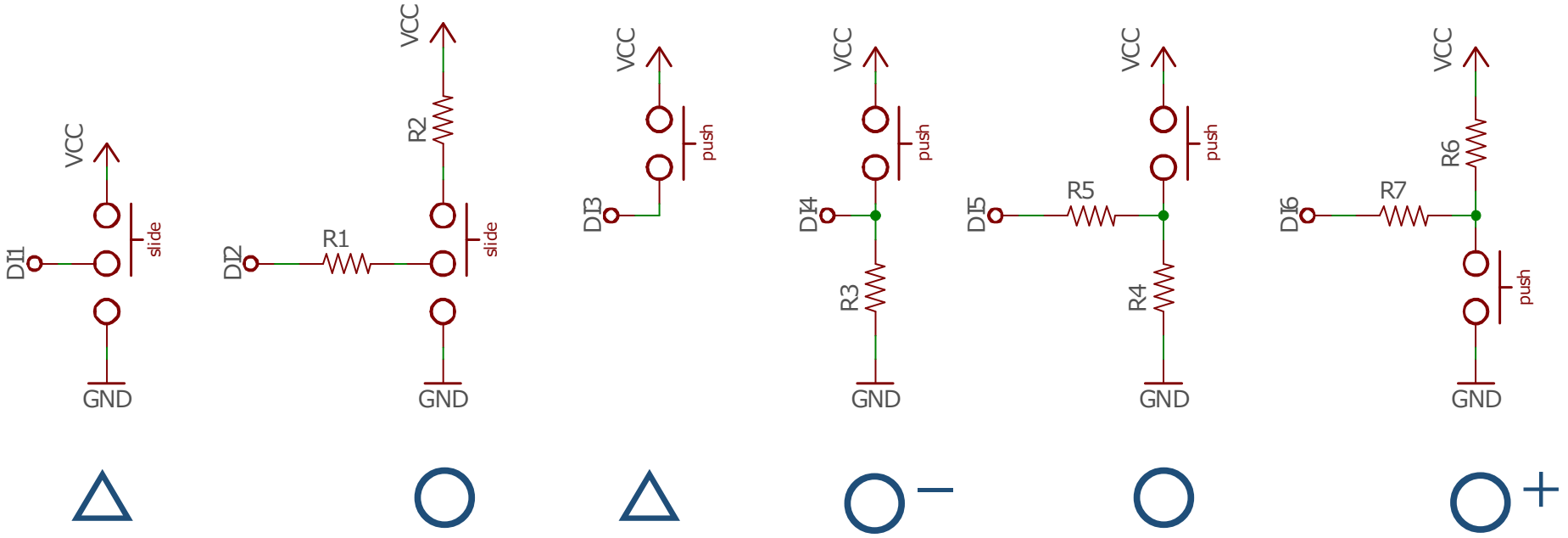
- スイッチを放した状態でGNDに落とす
- 直結すると押したときに短絡してしまう→抵抗を入れる
  - 押…R3の上端の電位=VCC
  - 放…入力電流≒0なのでR3の電圧降下≒0つまり電位はGND
- 回路やプログラムに手違いがなければこれでいいけど…

デジタル入力回路～プッシュスイッチ用

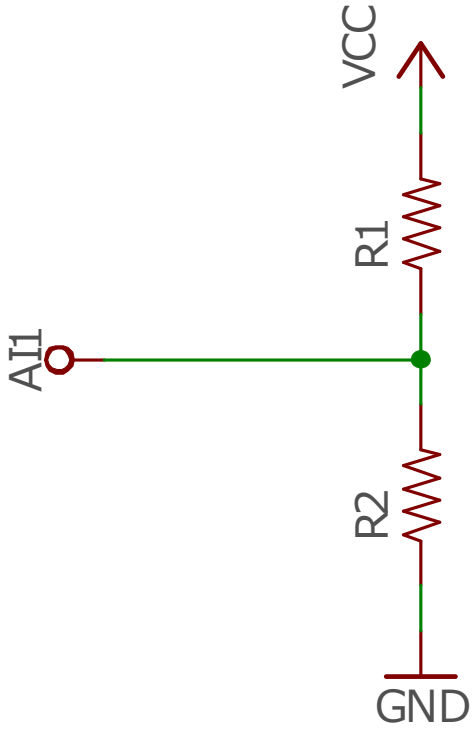


- 安全のため、入力端子との間に抵抗を挟む
- 正論理・負論理
  - 右：プルダウン付き、押したら1
  - 左：プルアップ付き、押したら0
- 負論理・プルアップが定番
  - スイッチ側にVCCが要らない
- マイコンによってはプルアップ抵抗を内蔵していて選択的に使用できる
  - スイッチ1個の直結でも使える

結論：デジタル入力回路の構成例

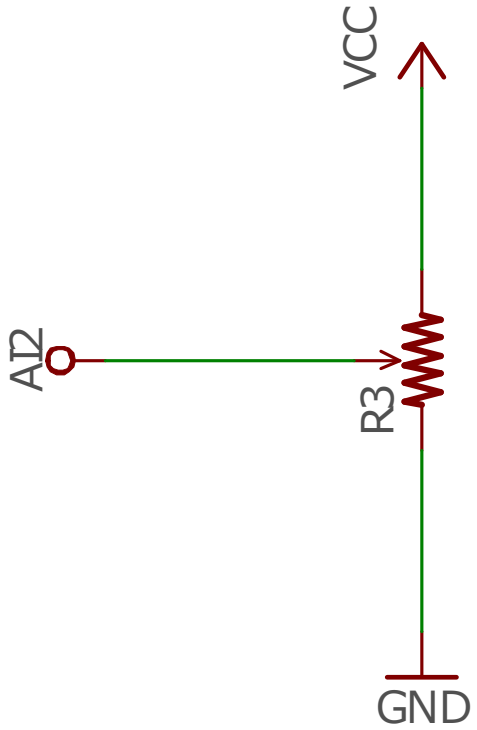


アナログ入力回路～基本構造



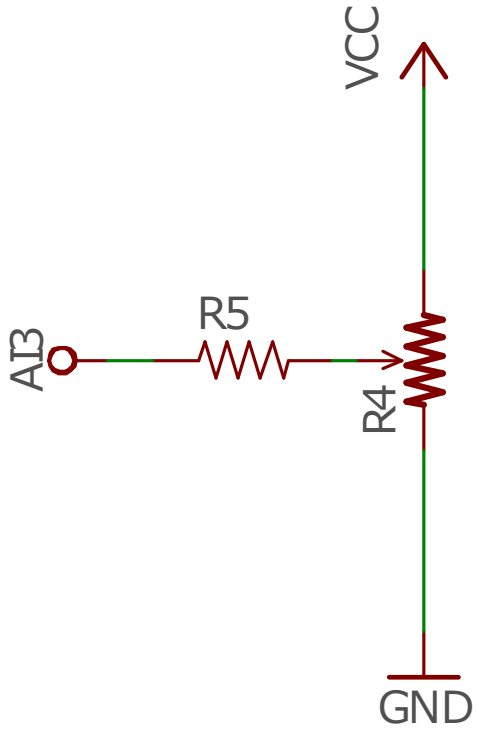
- GNDと基準電位（ここではVCC）の間の電位を入力する
- 抵抗で分圧
- $V_{\mathrm{in}} = \frac{R_2}{R_1+R_2} V_{\mathrm{cc}}$
- R1 や R2 が変わると上式に従って入力電位が変化する

アナログ入力回路～ボリューム直結



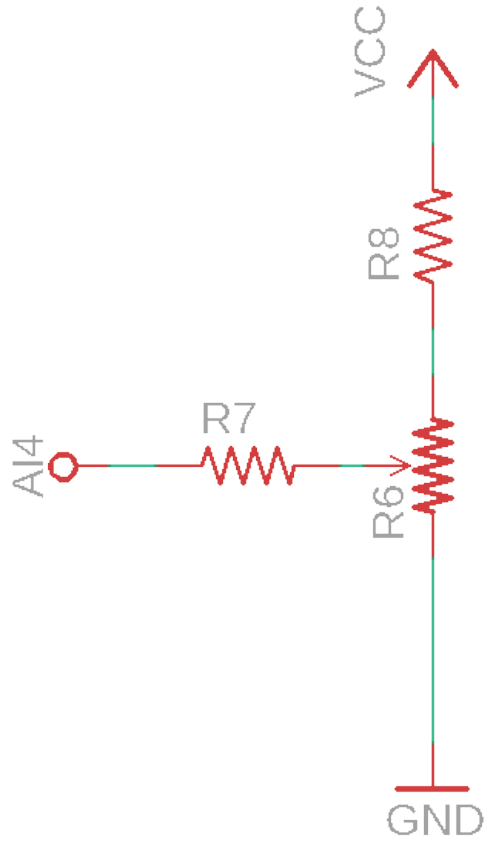
- 抵抗の両端の端子
- 途中で可変の端子
- 可変抵抗、ボリューム、ポテンショメータ、半固定抵抗、ロータリーエンコーダ
- 回路やプログラムに手違いがなければこれでいいけど…

アナログ入力回路



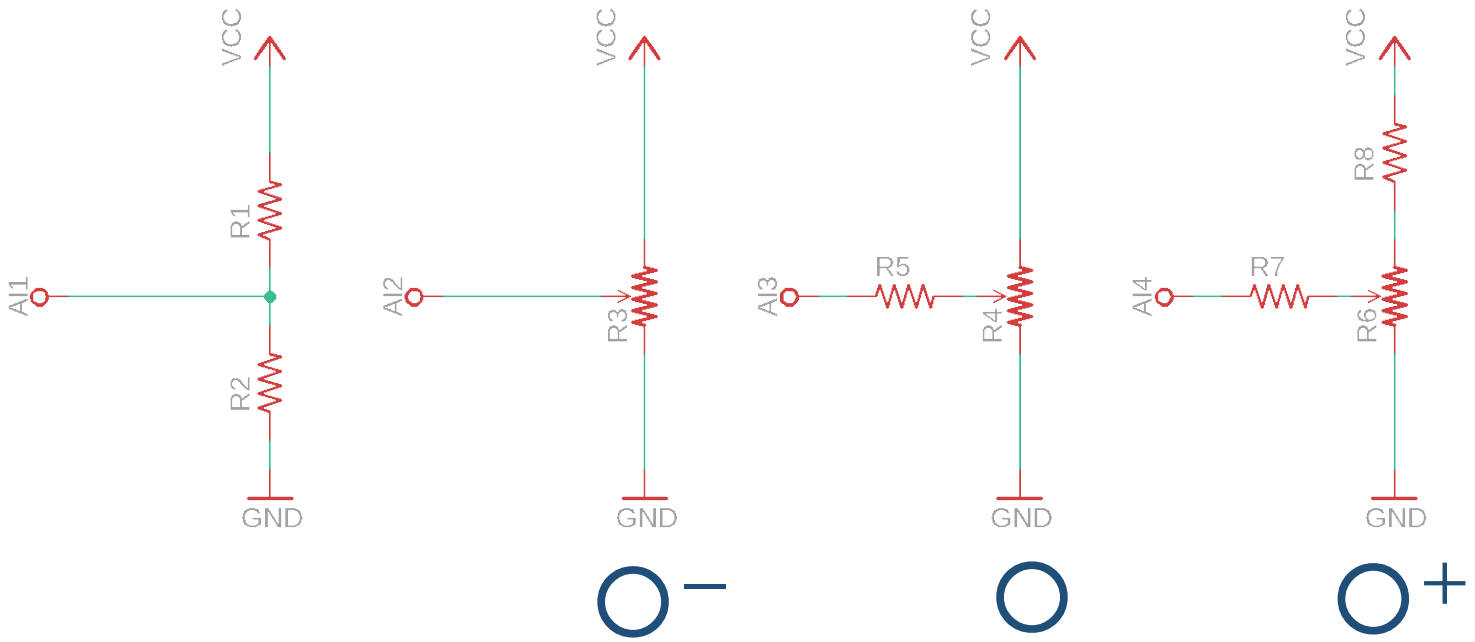
- 安全のため、入力端子との間に抵抗をはさむ
- 必要に応じて、ボルテージフォロア回路、増幅回路などを挟む

アナログ入力回路～ミス防止



- さらに**安全のため**、VCCとの間に抵抗をはさむ
- 他の入出力回路も全てVCCとの接続は必ず**抵抗に統一**することで、ブレッドボード配線の**電源短絡ミスを防ぐ**

結論：アナログ入力回路の構成例



システム設計CAD
マイコンプログラミング編
(4) マイコン入出力回路の実装

立命館大学 理工学部 電子情報工学科

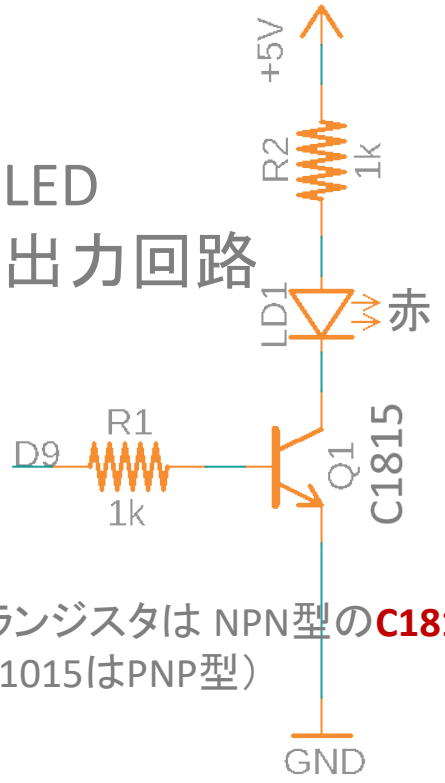
泉　知論　　田中　亜実

入出力回路の追加 …基本編（必須）

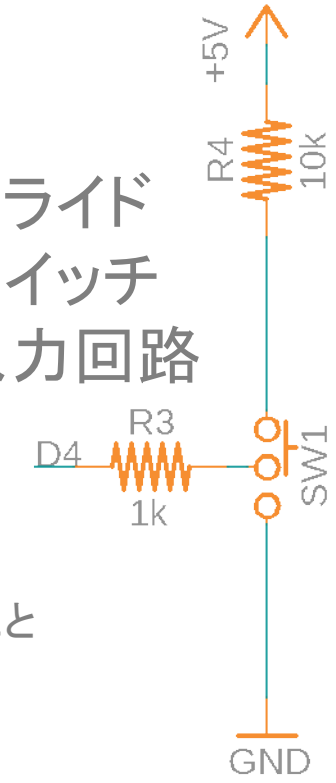
ブレッドボードにこれらの回路を実装しArduinoに接続する。
※もちろん、自作プリント基板を使っても良い

それぞれArduino の D9, D4, D2, A0 ピンに接続すること。
ふたつめをつくってもよい。その場合それぞれ D8, D5, D3, A2に接続する。

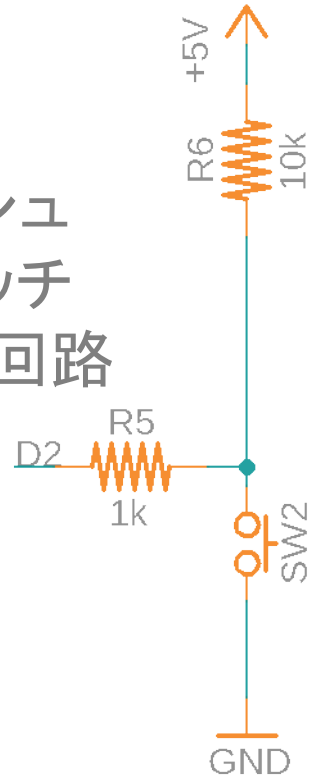
LED
出力回路



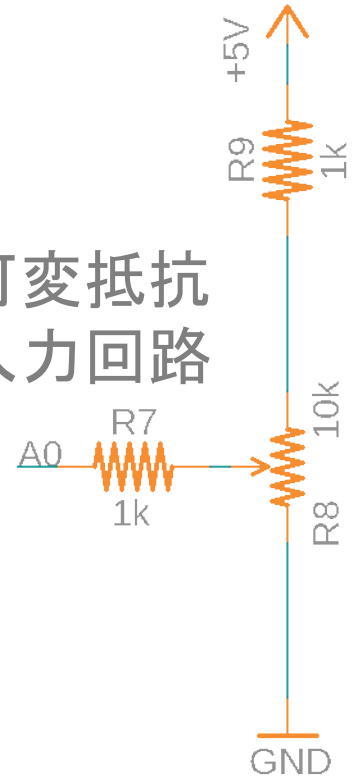
スライド
スイッチ
入力回路



プッシュ
スイッチ
入力回路

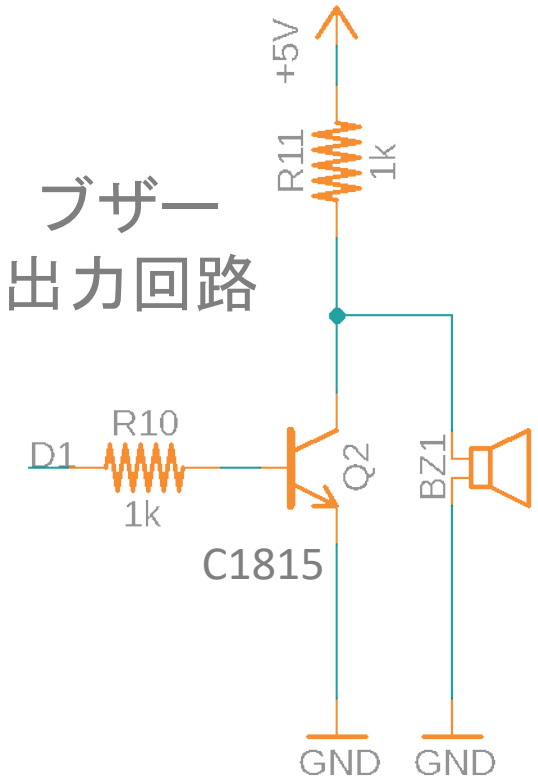


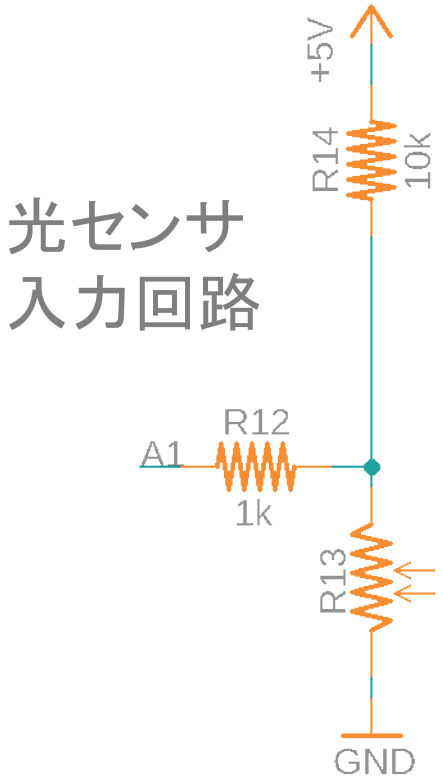
可変抵抗
入力回路



トランジスタは NPN型の**C1815**を使うこと
（A1015はPNP型）

入出力回路の追加…発展編（任意）





基本編ができて余裕がある人は、こちらの回路も追加する。
それぞれArduino の D1, A1 ピンに接続すること。

ブレッドボードの使い方

- 実験で使いましたよね？
- 自信が無い人は復習・自学自習すること

ブレッドボード 使い方 ［検索］

- 参考資料

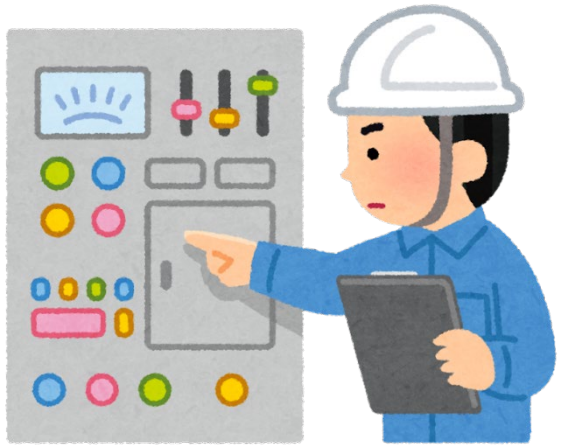
  サンハヤト「ブレッドボードの使い方」
  https://www.sunhayato.co.jp/dcms_media/other/howto_breadboard.pdf
  サンハヤト「電子工作ブログ～ブレッドボードの使い方」
  https://www.sunhayato.co.jp/blog/2015/03/04/7

！注意！

- 間違った回路を接続するとPCやArduinoが壊れる
- PC接続や電源ONの前に短絡チェックをする！

  できれば Vin-GND, 5V-GND, 3.3V-GND の抵抗値を測る

  少なくとも 5Vの接続と抵抗、GNDの接続を指さし確認する！
- 自信がなければ教員・TAの確認を受けること！
- 変だと思ったらすぐにUSBケーブルを抜くこと！

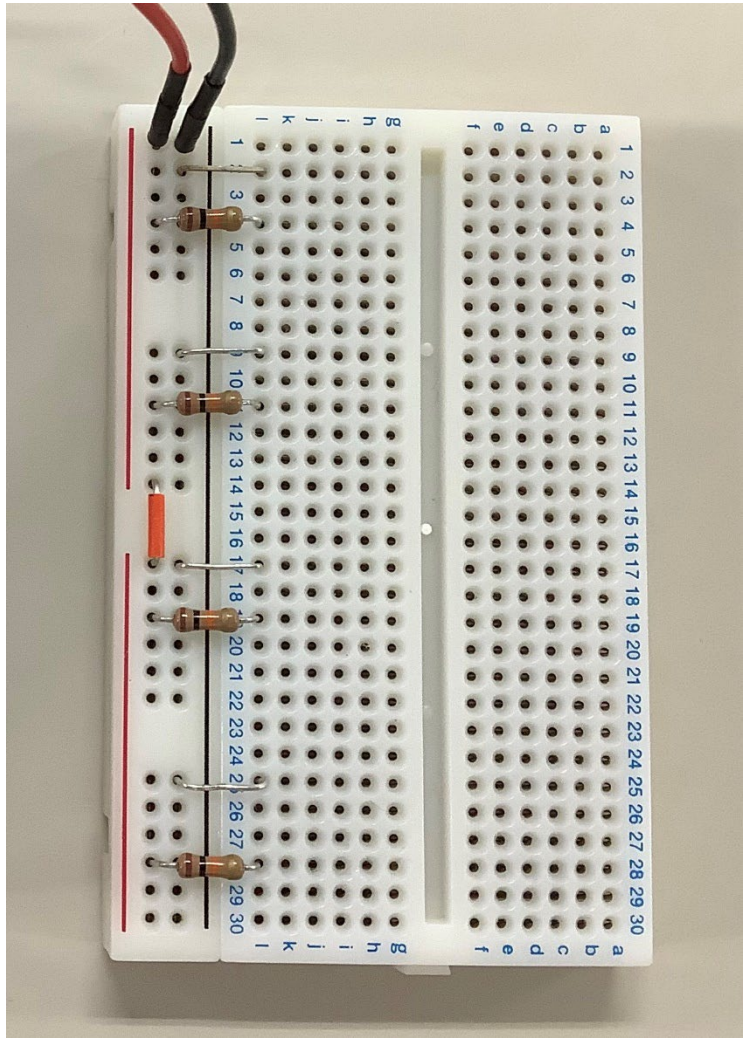


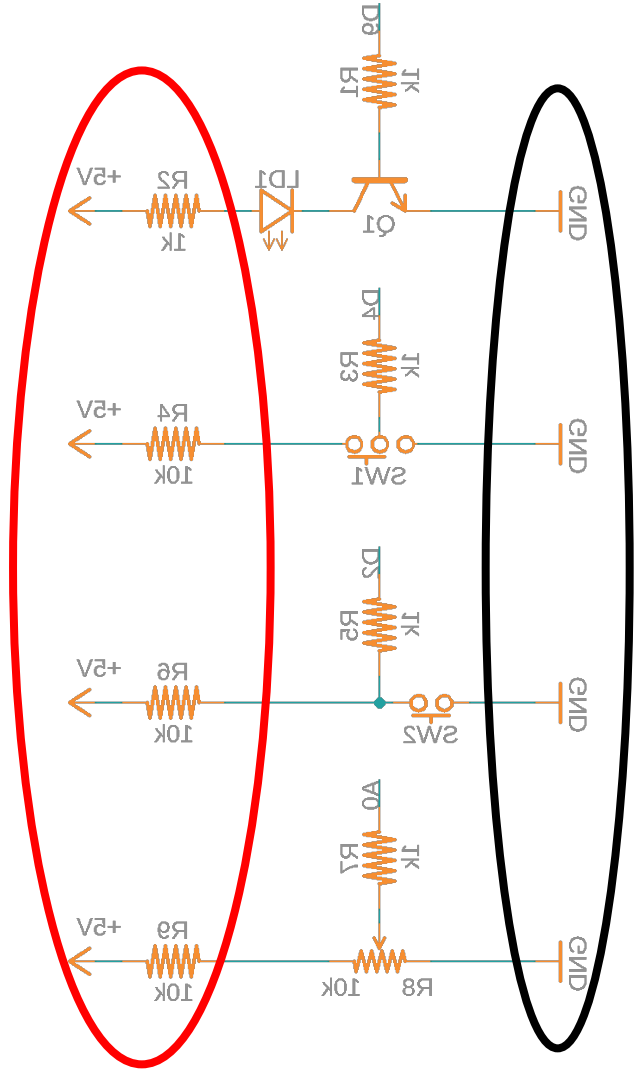
ミスを防ぐ・被害を最小限に抑える工夫

赤ワイヤ は ＋ (5V, 3.3V)
黒ワイヤ は ー (GND)
だけに使うこと

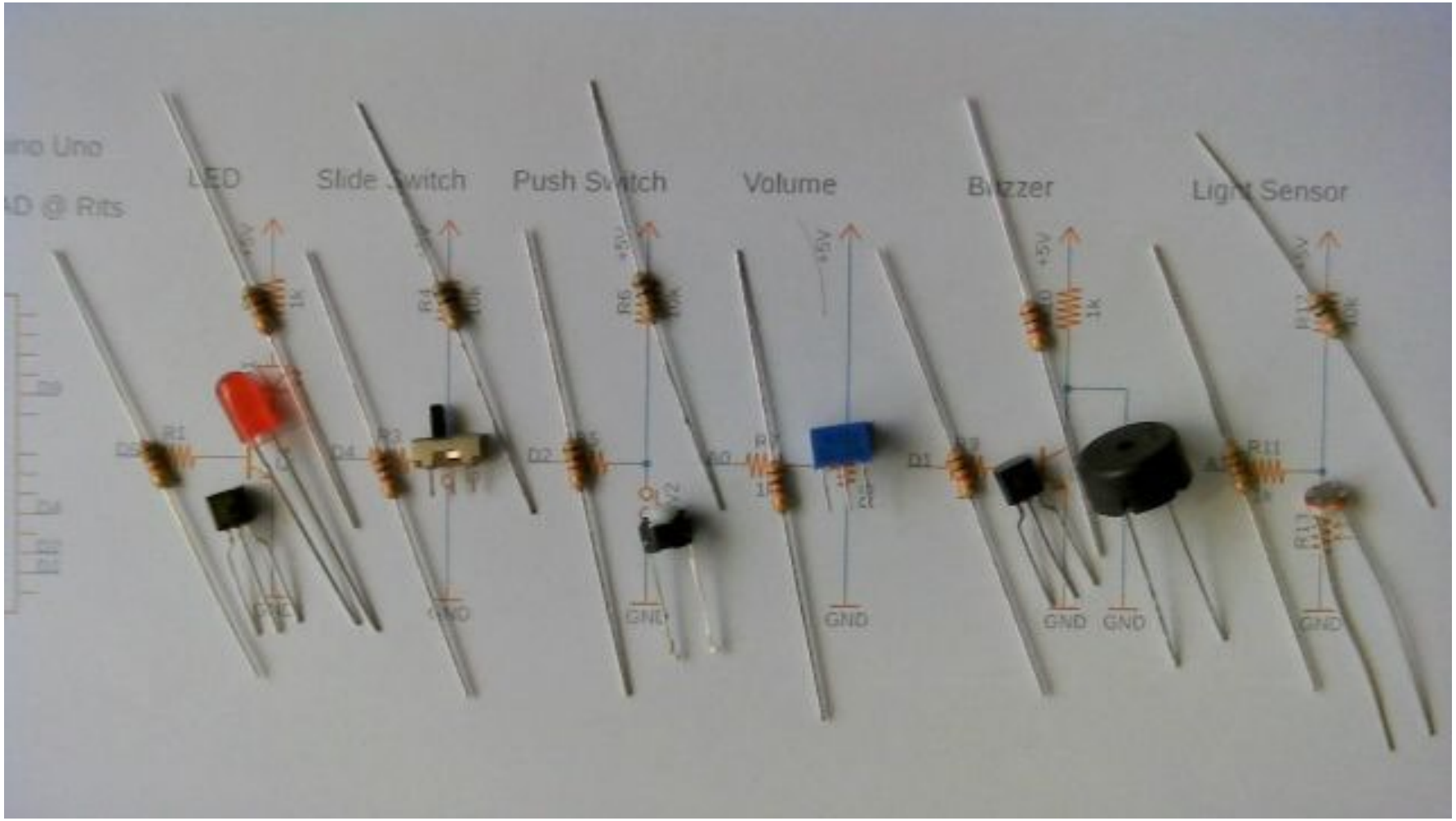
＋からは必ず抵抗で引き出す

ミスの中でも被害が大きいのは電源のショート（短絡、直結）です。このルールを守ってよく確認して、自信がないなら教員・TAに見せて、進めてください。
＋ラインの接続をすべて抵抗にして電源の短絡をなくせば、他を多少ミスしたとしても大きな破損にはなりません。

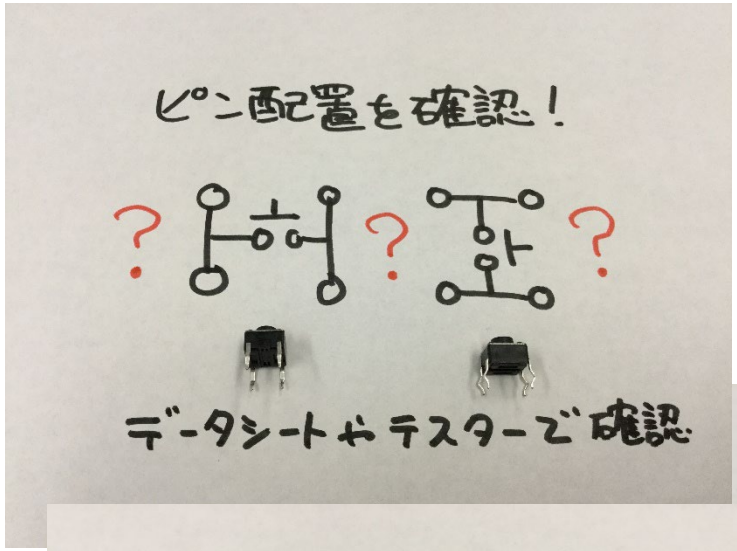




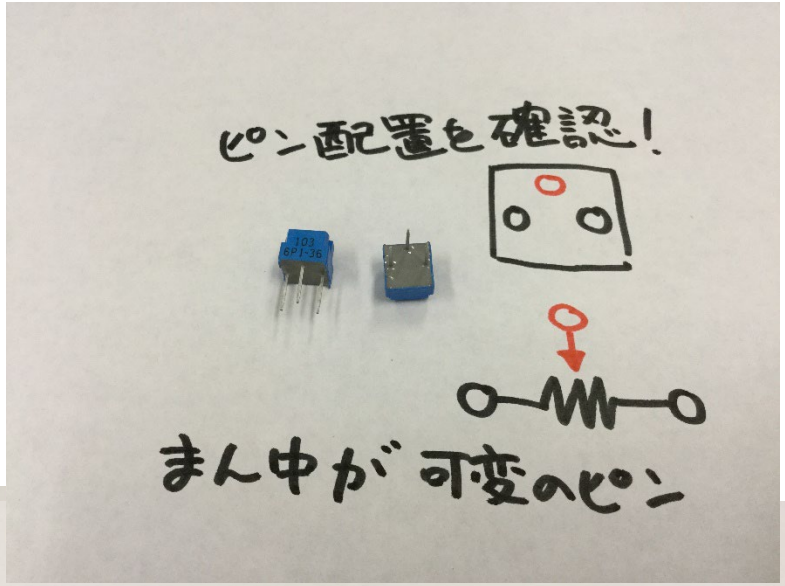
回路図と実際の部品

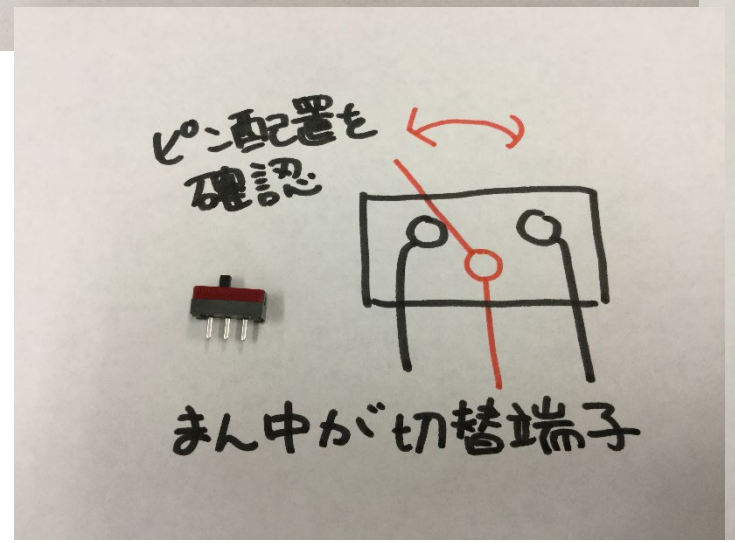

LED
Slide Switch
Push Switch
Volume
Buzzer
Light Sensor
+5V
GND
1K

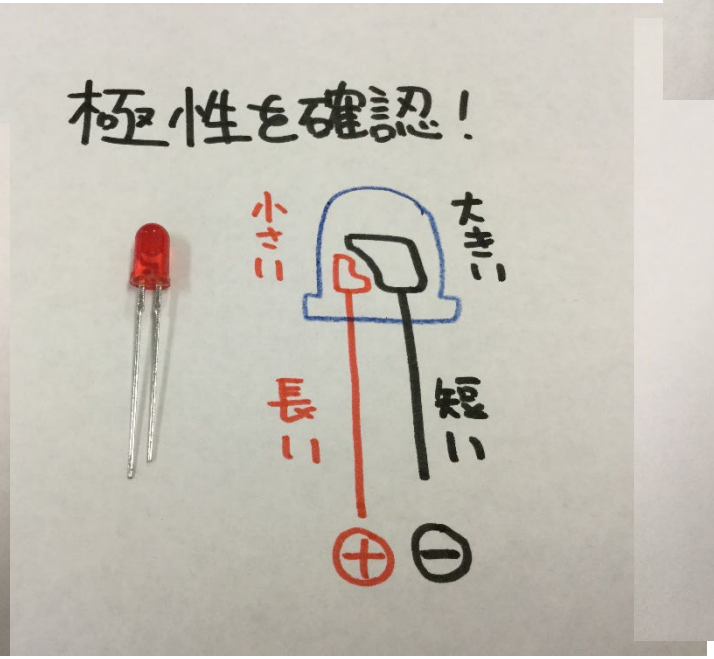
ピン配置に注意

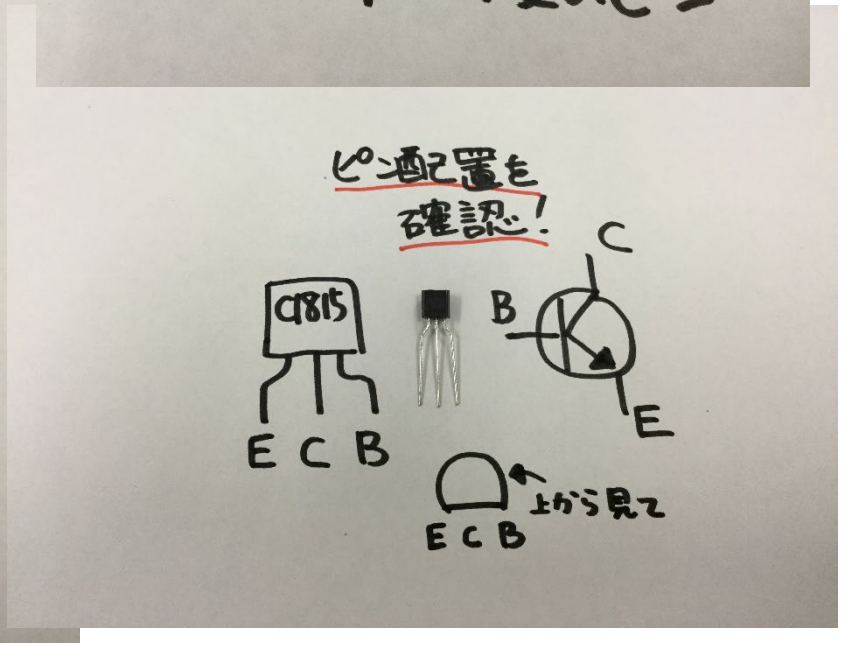


データシートや
実物で確認すること！



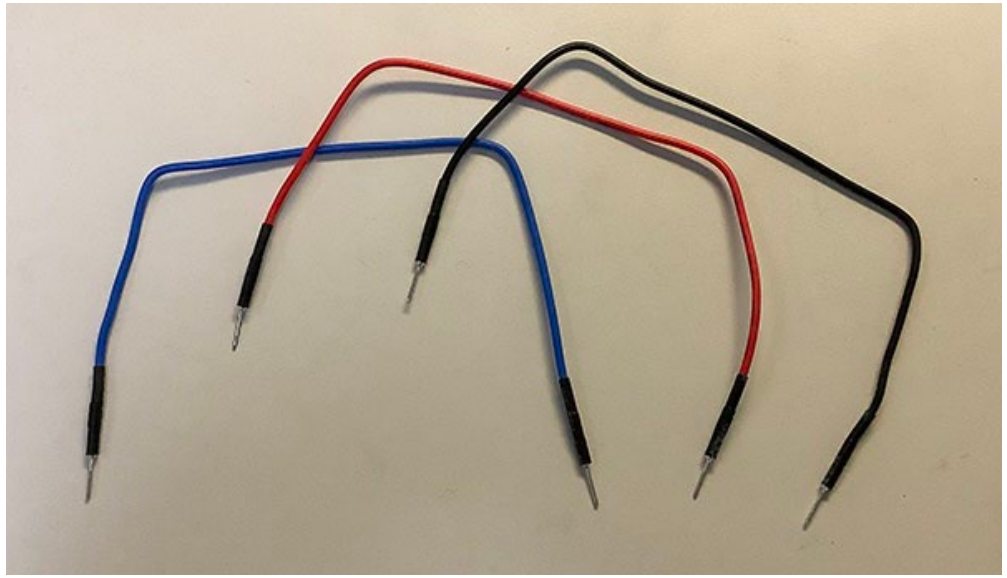






部品の不足・破損について…

- 不足している部品があったらTA・教員に連絡すること。補充します。
- 部品が破損している場合もある。怪しいと思ったらTA・教員に相談する。
- 特にこの↓ワイヤは断線しやすい。



自分が設計した基板を使いたい！


AIOL
自分の基板と
部品をもって
AIOL に Go!

システム設計CAD
マイコンプログラミング編
(5) プログラミング

立命館大学 理工学部 電子情報工学科
泉　知論　　田中　亜実

演習1　基本入出力の動作確認

- 次の仕様を実現せよ
- スライドスイッチをONにしたらLEDが点灯する
- スライドスイッチをOFFにしたらLEDが消灯する

ヒント～使用する関数

```
pinMode()
digitalRead()
digitalWrite()
delay()
```

演習2　基本入出力の動作確認

- 次の仕様を実現せよ
- スライドスイッチをONにしたらLEDが**点滅**する
- スライドスイッチをOFFにしたらLEDが消灯する
- 点滅周期は2秒程度にする

ヒント～使用する関数

```
pinMode()
digitalRead()
digitalWrite()
delay()
```

Arduino Reference

LANGUAGE
FUNCTIONS
VARIABLES
STRUCTURE
LIBRARIES
GLOSSARY

The Arduino Reference text is licensed under a Creative Commons Attribution-Share Alike 3.0 License.

Find anything that can be improved? Suggest corrections and new documentation via GitHub.

Doubts on how to use Github? Learn everything you need to know in this tutorial.

Language Reference

Arduino programming language can be divided in three main parts: functions, values (variables and constants), and structure.

FUNCTIONS

For controlling the Arduino board and performing computations.

Digital I/O
digitalRead()
digitalWrite()
pinMode()

Analog I/O

Characters
isAlpha()
isAlphaNumeric()
isAscii()
isControl()
isDigit()

Arduinno Reference ［検索］

「教わる」ではなく「学ぶ」こと！
解説、マニュアル、リファレンス、
仕様書、データシートなどを
自ら探し求めて読むべし！

課題1　状態 v.s. 変化

- 次の仕様を実現せよ
- プッシュスイッチを押すとLEDの点滅／消灯が切り替わる
- 注意
  - プッシュスイッチ押下判定
    - 直前の状態を覚えておいて
    - 直前の値と現在の値で判定
  - chattering (bouncing) 現象について

ヒント～使用する関数

```
pinMode()
digitalRead()
digitalWrite()
delay()
```

RITSUMEIKAN

Analog

連続値

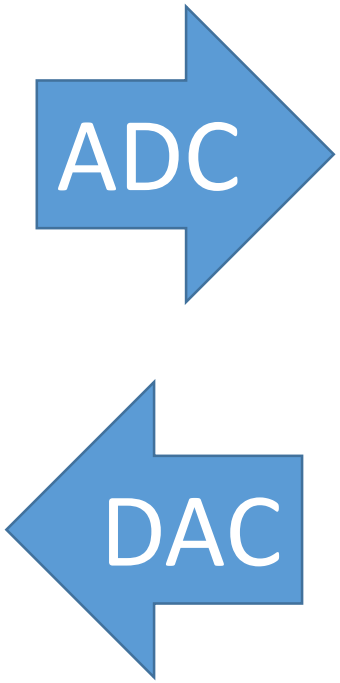
電子回路では

電圧で表現

例

0V … 5V

-12V … +12V



Digital

離散値

コンピューター内では

二進数で表現

例

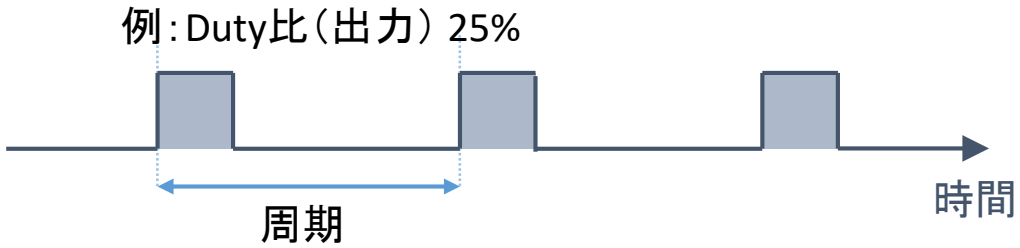
8bit 符号なし 0…255

10bit 符号なし 0…1023

16bit 符号付き -32768…+32767

デジタル出力でアナログ値を表現～PWM

- Pulse Width Modulation
- パルス幅変調
- 矩形波を出力
- 0と1の時間比率で中間値を表現
- 電圧のかわりに時間で量子化
- 駆動対象に応じて周期を設定する



- ATmegaでは、内部タイマカウンタを使用して周期信号を発生
- 使用可能なピンは決まっている
- カウンタは多用途で共有しているため、共存できない機能あり
- ライブラリ関数

  `analogWrite()`

演習3　PWM出力の動作確認

- 次の仕様を実現せよ
- スライドスイッチとプッシュスイッチの組合せで、LEDの明るさが以下のように変わる
  - ON-ON … 100%
  - ON-OFF … 66%程度
  - OFF-ON … 33%程度
  - OFF-OFF … 0%（消灯）

ヒント～使用する関数

```
pinMode()
digitalRead()
analogWrite()
delay()
```

アナログ入力～ADC

- Analog-to-Digital Converter
- アナログ／デジタル変換器
- アナログ値（電圧）を読み取って数値に変換して入力する
- 性能の指標
  - 分解能（数値のビット数）
  - 速度（1秒当たりの変換回数）

- ライブラリ関数
  `analogRead()`

演習4　アナログ入力の動作確認

- 追加した入出力を用いて、次の仕様を実現せよ
- 可変抵抗の回転角度に応じて、LEDの明るさが変わる
  - 左いっぱいにまわすと0%
  - 右いっぱいにまわすと100%
  - 中間では回転角に応じた明るさ

ヒント～使用する関数

```
pinMode()
analogRead()
analogWrite()
delay()
```

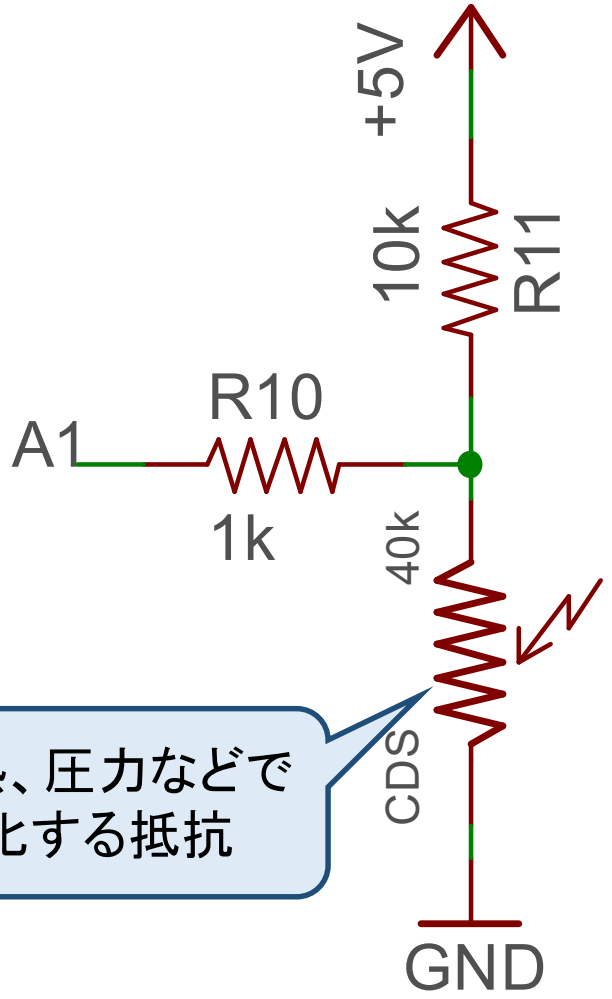
（演習4～発展）

- 可変抵抗（回転センサ）の代わりに光センサや熱センサを接続する
  - 光で抵抗値が変化…CdSセル
  - 熱で抵抗値が変化…サーミスタ、熱電対

  など
- 可変抵抗の値の範囲に応じて固定抵抗値(R11)を決める

  （可変側の中央程度）
- 範囲は0～5Vよりも狭くなることに注意

  （プログラム側で補正が必要）



Serial ほか便利なライブラリ群

- デバグのため値を表示したい？
- シリアル通信でできる
- known as UART, COM, Serial
- Arduino では D0, D1 を使う
  - D0 … RX (Receive, 受信)
  - D1 … TX (Transmit, 送信)
- USBでPCと接続時はPCのCOM port と接続される

※この演習の回路ではBZとピンを共有

- Arduino IDE では標準ライブラリ群に含まれている
- Serial ライブラリ
  - `Serial.begin()`
  - `Serial.println()` ほか
- See Arduino Reference
- PCで入出力するには Arduino IDE ツールの〔ツール〕→〔シリアルモニタ〕を開く

補足：Serial の設定について

※（）内は本演習推奨値

- Baud Rate (9600 bps)
  通信速度
- Data Bits (8bit)
  データひとつ（一文字）のビット数
- Parity (none)
  ビットエラー検出方法
- Stop Bits (1 bit)
  一文字の後にあけるビット幅
- Flow Control (none)
  データ流量（中断・再開）制御方法

端末設定（表示と入力）

- ローカル・エコー
  自分で入力した文字を表示するか否か
- 改行コード
  LF/CR+LF/CR
- 文字コード、漢字コード
  US-ASCII/JIS
  JIS/SJIS/EUC/UTF

演習５　Serial の動作確認

- 追加した入出力を用いて、次の仕様を実現せよ
- １秒毎にスライドスイッチ、プッシュスイッチ、ボリュームの値をSerial でPC上に表示させる

ヒント～使用する関数

```
pinMode()
Serial.begin()
digitalRead()
analogRead()
Serial.print()
Serial.println()
delay()
```

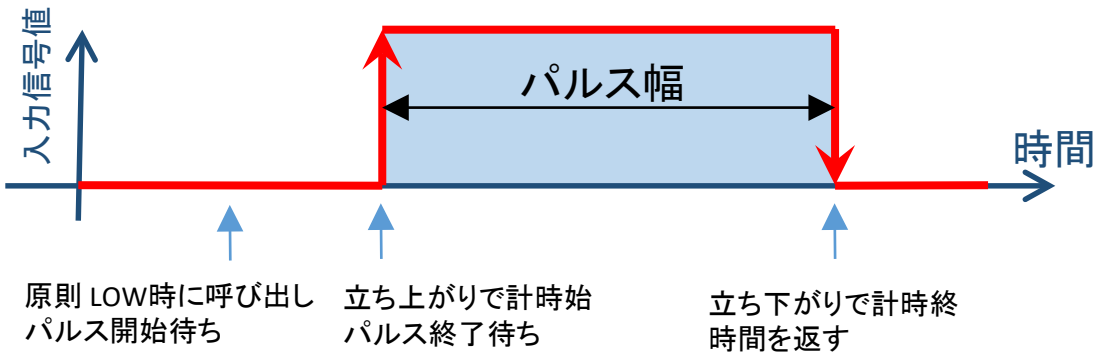
デジタル入力でアナログ値を表現

- PWMの逆
- 矩形波（パルス）を入力
- 1の時間の長さで中間値を表現

- ATmegaでは、内部タイマカウンタを使用してパルス長を計測
- ライブラリ関数

  `pulseIn()`

例: `pulseIn(Pin, HIGH)` の動作タイミング



課題2　pulseIn() & analogWrite()

- 次の仕様を実現せよ
- プッシュスイッチを1度押すとLED の明るさが変わる
- 押してから放すまでの時間によりLEDの明るさが変わる。
  - 0秒では0%
  - 1秒につき10%程度上昇
  - 10秒以上では100%
- 60秒でタイムアウトして消灯

ヒント～使用する関数

```
pinMode()
digitalRead()
pulseIn()
analogWrite()
delay()
```

（課題2～発展）

- プッシュスイッチのかわりに超音波距離センサを使う。
- 距離に応じて明るさが変わる。
- 超音波の反射波の時間をpulseIn()で測る。

HC-SR04 ［検索］

イベント検知とポーリング(polling)

- イベント
  - スイッチが押された
  - 赤外線光を検知した
  - 設定した時間になった
  - など
- イベントに関連して一連の処理が実行される(タスク)
- イベントの検知→処理

- Polling … 定期的に信号を読み出して確認
- 読出周期≒反応速度
- イベントと関連タスクが複数あると複雑に影響し合う
  - 読出周期が長くなる
  - 読出周期が不安定になる
  - プログラムが複雑になる

割込み (interrupt)

- 予めイベントの条件と処理（割込ハンドラ, 割込サービスルーチン）を設定しておく
- イベントが発生すると、自動的に実行中のプログラムを中断し、割込ハンドラの処理を開始する
- 処理が終わると中断していた処理に戻る
- 中断・再開された処理はそのことを感知する必要はない
- 割込みハンドラと普通の定期実行ルーチンをうまく組み合わせる

例：音声再生の停止ボタン

- 通常の処理として音声を再生しておく
- 割込みイベント：停止ボタン押下の入力信号
- 割込みハンドラ：再生停止の指示

例：ゲームの画像表示ルーチン

- 通常の処理としてゲームの進行
- 割込みイベント：タイマ
- 割込みハンドラ：フレームの書き換え

などなど

割込みサービスルーチンと登録

- 割込み時に呼び出される関数
- 割込みサービスルーチン
  (Interrupt Service Routine) または割込みハンドラ (Interrupt Hander) と呼ばれる
- 定形の呼び出し方

  Arduino IDE では↓の形

  ```
  void exampleISR(){ … }
  ```

✓細々した制約あり→マニュアル参照

- setup()内で割込みの発生条件と発生時に呼び出す関数を設定しておく

`attachInterrupt(`n`,`ISR`,`$mode$`)`

n … 割込番号（D2ピンは 0）

ISR … ISRの関数名

$mode$ … 発生条件

RISING, FALLING, etc.

- 以後は自動で呼び出される

volatile 変数

- これ↓は普通は無限ループ

```
int flag;
void loop() {
    flag = 1;
    while (flag) {
        digitalWrite(PinLED, LOW);
    }
    digitalWrite(PinLED, HIGH);
    delay(1000);
}
```

- コンパイラはwhile開始時にflagを読み出し、以後その値を使用する

- 割込みで flag=0 にしたら、それでwhileループが終わる
- コンパイラに指示する

  volatile int flag;

  ➢volatile = 変わりやすい
- これで毎回flagを読み出すようにコンパイルされる

演習6　割込みの動作確認

- 次の仕様を実現せよ
- プッシュボタンを押すとLEDが1秒間点灯する
- 前ページのソースコードに割込みを追加する
- プッシュボタンを押すと割込みが発生し、割込みサービスルーチンで flag=0 にする

✓ while 内でプッシュボタンのチェックすれば実現できるが、ここではそれはしない。割込みの練習なので、割込みを使う。

ヒント～使用する関数・機能

`attachInterrupt()`

ISR (Interrupt Service Routine)

`volatile` 変数

課題３　割込み

- 次の仕様を実現せよ。
- 通常処理として、10秒毎にボリュームの値と割込み回数をシリアルに出力する。
- プッシュボタンを押すと割込みによりLEDの点灯／消灯を切り替える。また、割込み回数を＋1する。

✓ チャタリングや多重割込みへの対策はしなくてもよい

ヒント～使用する関数・機能

`attachInterrupt()`

ISR (Interrupt Service Routine)

`volatile` 変数

補足：作法と常識と非常識

- 業界が変われば常識も変わる
- 制御用マイコンは非力
  機械（マイコン）に優しく
  人（プログラマ）に厳しく
- 一般的なPCのプログラミングとは異なるテクニックが必要

- グローバル変数を積極的に使用
- 大きな局所配列は厳禁
- 関数に分けることに消極的
- 再帰は避ける
- volatile 変数
- いざとなったらアセンブラも使う

システム設計CAD
マイコンプログラミング編
(6) レポート課題

立命館大学 理工学部 電子情報工学科
泉　知論　　田中　亜実

レポート3～マイコンプログラミング

- 課題3または課題2または課題1についてまとめる
  ※完成していない場合は、できたところまでの内容をまとめてください
- 以下のファイルをひとつの**zipファイル**にまとめて別途指示する日時までに manaba+R にアップロードすること
- zipファイル名はRAINBOW IDとする…例 ri0123ab.zip
- レポート文 (MS Word または PDF)
  - 講義名、レポートタイトル、学生証番号、氏名、提出年月日
  - 設計の目的と設計物（何を設計するか）の説明、設計環境の概説、設計結果（どのようなものができたか）の説明、まとめ（成果物、修得できたこと、反省点、感想）、参考文献について記述すること
  - 外付け回路の回路図、ブレッドボードやプリント基板の写真、プログラムコードなどを入れて説明すること
- 設計ファイル
  - プログラム *.ino