インターフェース回路設計と IO制御プログラミング

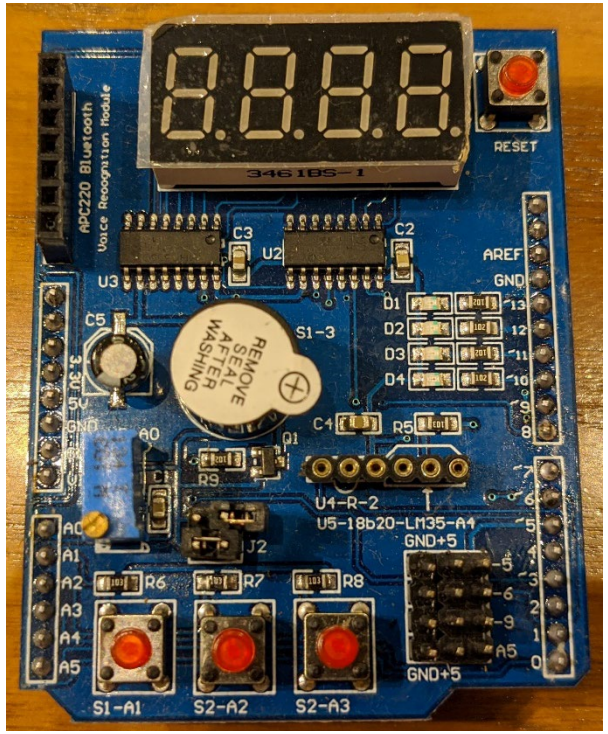
ポリテクセンター熊本

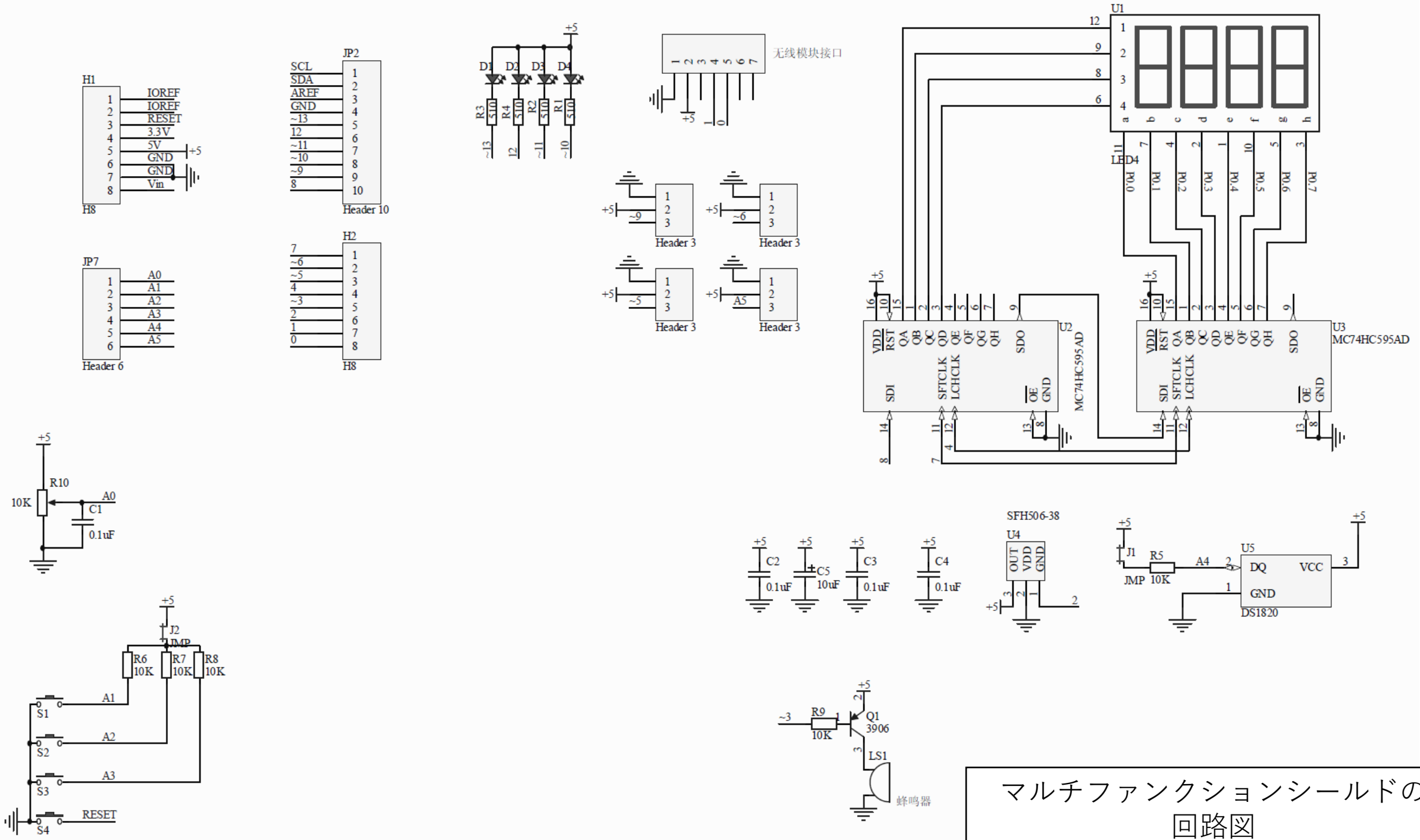
目的・目標

- 目的
  - マイコンと周辺回路がどう繋がるのか？を学ぶ
    - アナログ回路，デジタル回路とマイコンの結びつき
  - ビジュアルプログラミングを通じてプログラミングの基礎を得る
- 目標
  - LED，スイッチ回路を理解して制御できる
  - 7SEG LED回路を理解して制御できる
    - スタティック点灯，ダイナミック点灯を実現する
  - プログラミングの基礎を身に着ける
    - 逐次・分岐・繰り返し
    - 値と変数の取り扱い

Multi Function Shield

- 中国国内で教育用途で開発された多機能シールド
  - Arduino Unoで使用する事を前提とされている
  - 非常に安価
    - 1000円以下
- 搭載されている機能
  - LED　× 4
  - プッシュスイッチ　× 3
  - ボリューム　× 1
  - ブザー　× 1
  - 7セグLED(4桁)　× 1





マルチファンクションシールドの回路図

IO表

Arduino				MFシールド		メモ
IO		ADC	PWM			
0	RXD			-		
1	TXD			-		
2	INT0			*	メス	
3	INT1		○	ブザー		
4				7SEG	LCHCLK	
5			○	*	オス	
6			○	*	オス	
7				7SEG	SFTCLK	
8				7SEG	SDI	
9			○	*	オス	
10			○	LED4		
11			○	LED3		
12				LED2		
13				LED1		
A0		○		ボリューム		
A1		○		スイッチ1		J2を外せばフリー
A2		○		スイッチ2		J2を外せばフリー
A3		○		スイッチ3		J2を外せばフリー
A4	I2C	○		*	メス	J2を外せばフリー
A5	I2C	○		*	オス	

＊ … 空きピン

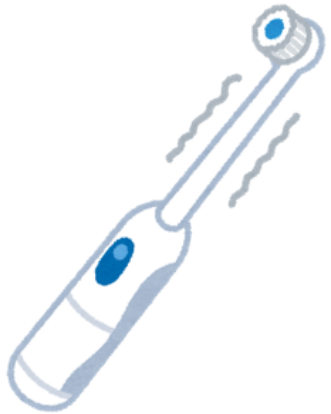
ブロック図を書いてみよう

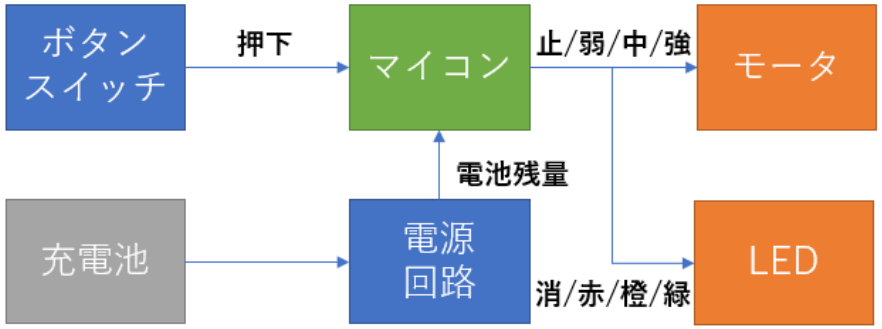
- システム設計・ハードウェア設計を行う場合，いきなり回路図を書く事はしません。
- 抽象化したブロック図を書くことでハードウェア全体をイメージしやすくします。
  - 主に表現するモノ
    - マイコン
    - センサ
    - アクチュエータ
    - ディスプレイ　　など
  - 必要あればコネクタなども

マイコンと入出力機器を表現する事がポイント

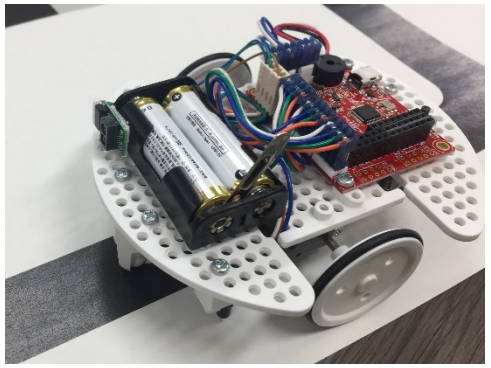
ブロック図の例

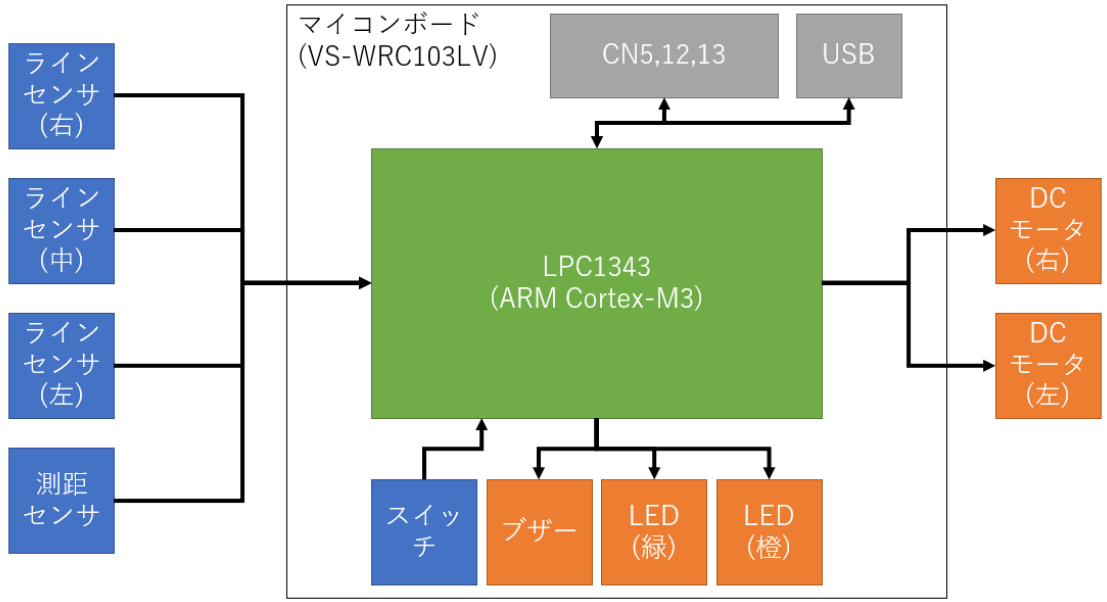
電動歯ブラシ





ライントレースカー

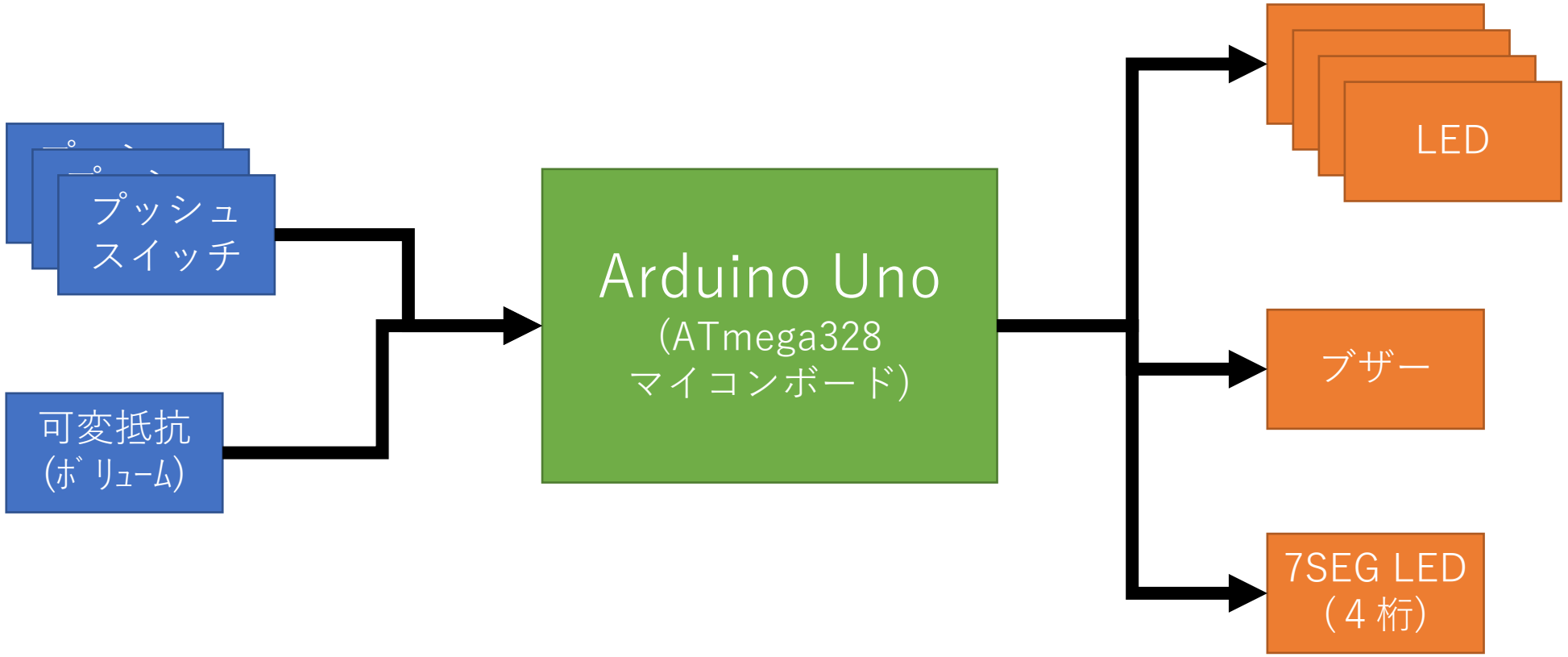




Arduino+Multi Function Sheildのブロック図

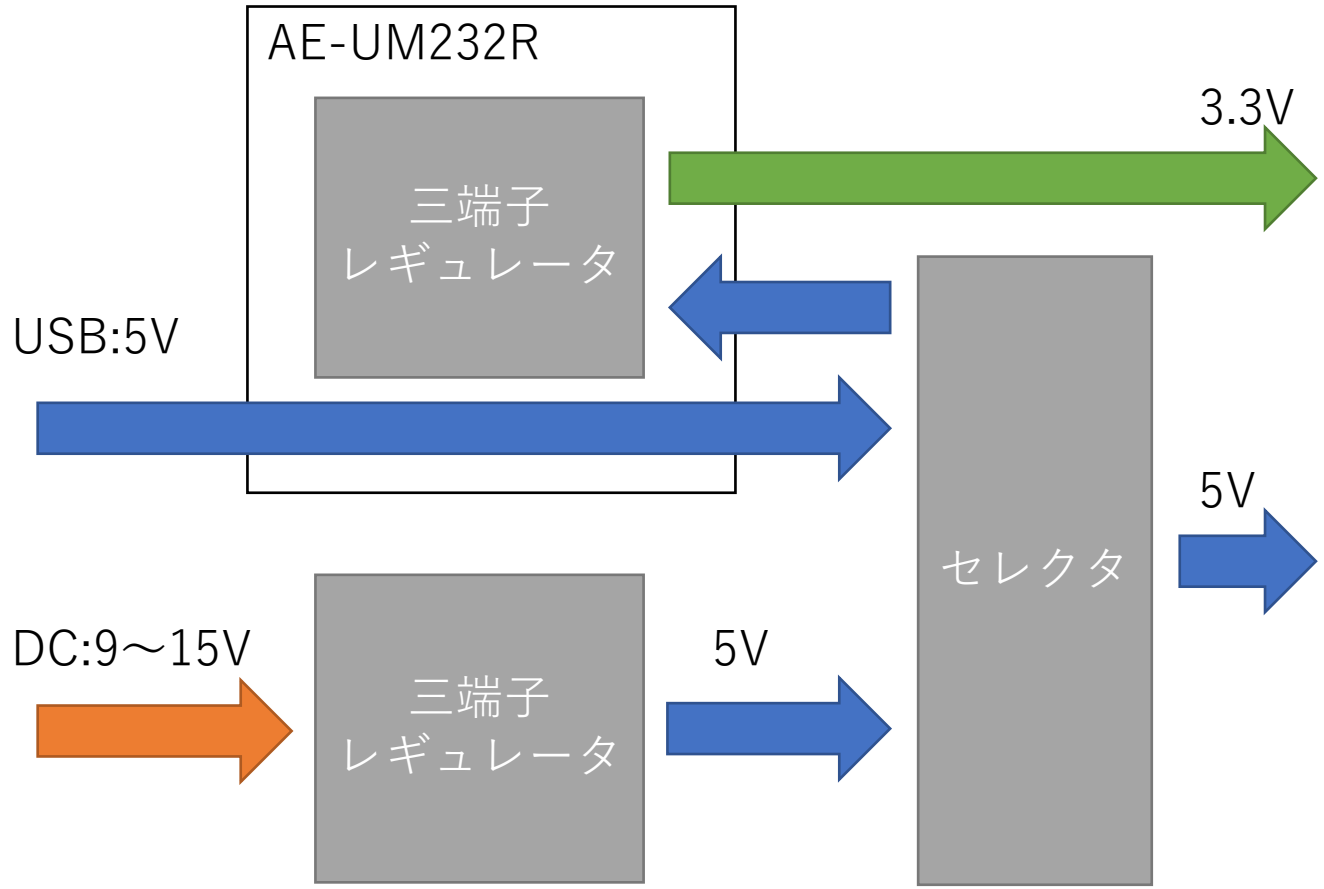
Arduino Uno
(ATmega328
マイコンボード)

(解答例)


プッシュ
スイッチ
可変抵抗
(ボリューム)
Arduino Uno
(ATmega328
マイコンボード)
LED
ブザー
7SEG LED
(4桁)

電源系統のブロック図も書いておくと良い

- 近年の組込みシステムは，**5V駆動のみ**とは限りません
  - 1つのシステムの中に複数の系統の電源が入っている事もザラです.
  - 使いたい電子部品によっては，5Vでは動かず「3.3Vのみ動作」というものも多くあります
  - その為に，電源系統のブロック図も書いておくと，ハードウェアを検討・設計する際に便利です

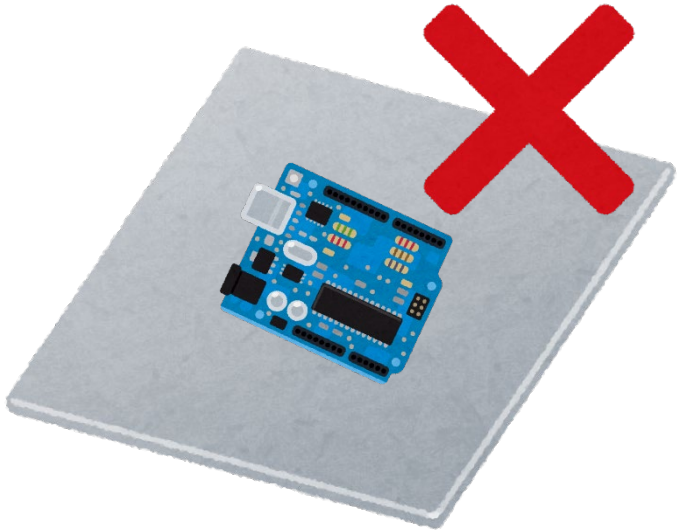


- 三端子レギュレータとは
  - 入力電圧を降圧して，一定の電圧を出力する電源IC
  - 降圧した分のエネルギーは熱エネルギーに変換する
  - 安定した電源共有が可能であり，多用される電源ICの1つ
- 3.3V系と5V系の2つの電源ラインあり
- 最大供給電流
  - 三端子レギュレータに依存する
  - USB:5V駆動時
    - 5V　… 接続したUSBポートに依存する
    - 3.3V　… 最大50mA
  - DC:10V駆動時
    - 5V　… 最大1.5A
    - 3.3V　… 最大50mA

実習を始める前に

モノを壊さない為に…

- 壊さない為に以下の事に気を付けましょう！
  1. **水気のあるものは遠ざける**
  2. **金属面(PCの天板など)に基板を置かない**
  3. **回路を組み替える時にはUSBケーブルを抜く**



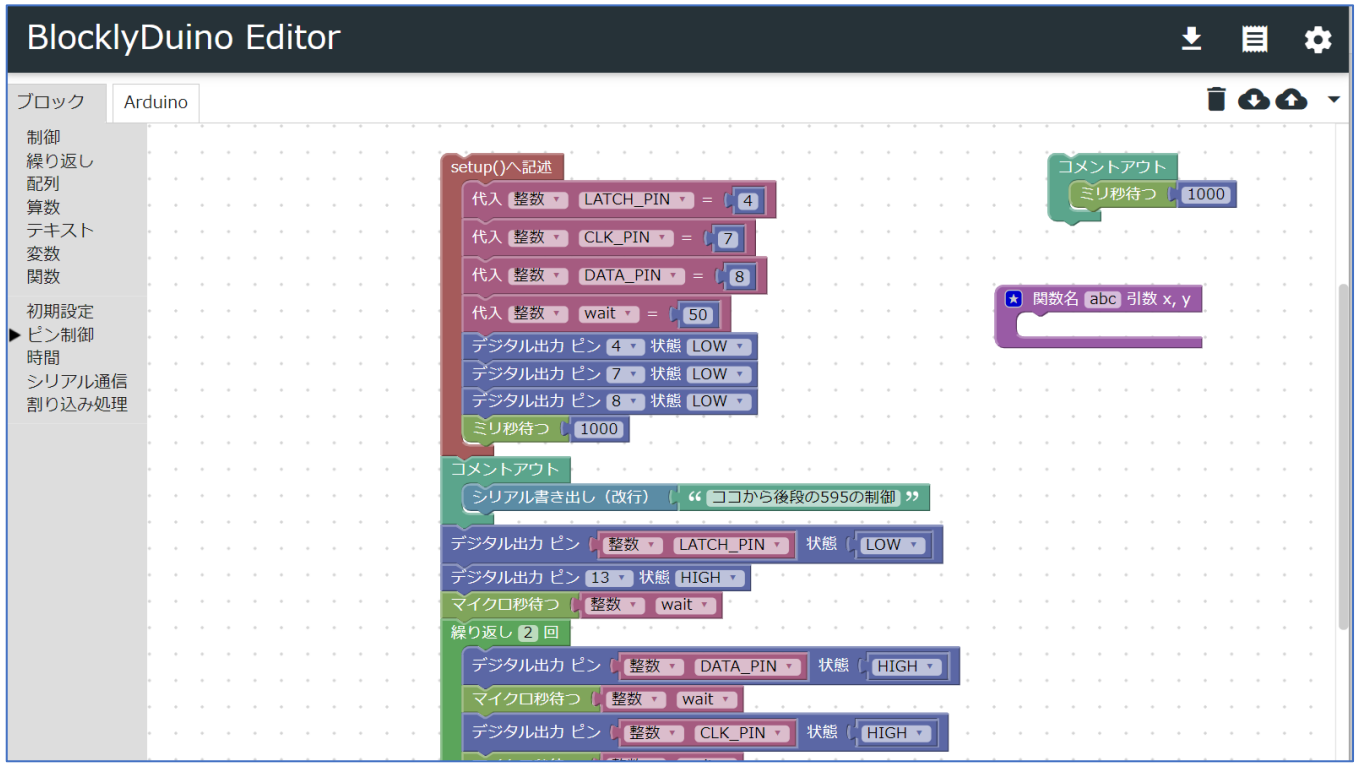
開発環境

Arduino IDE

- Arduino系のマイコンボードを開発する際に使用する統合開発環境

- 機能
  - エディタ
  - プログラムの書き込み
  - コンパイラ
  - ライブラリ管理　など

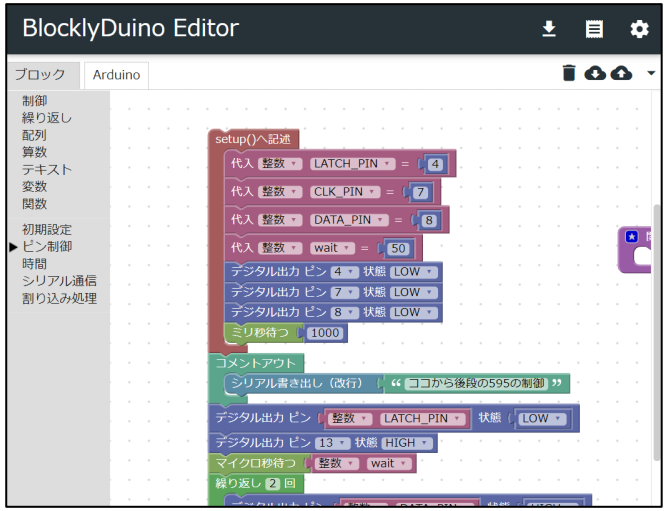
Bloklyduino



https://10.10.140.225/BDE/

- Arduino向けのビジュアルプログラミング環境
- ブロックプログラミングで視覚的に分かりやすいプログラミングが可能
  - ブロックからコードを自動生成する

Arduino IDEとBlockduinoの関係



Blocklyduino

コピー
&
ペースト

Arduino IDE

Write

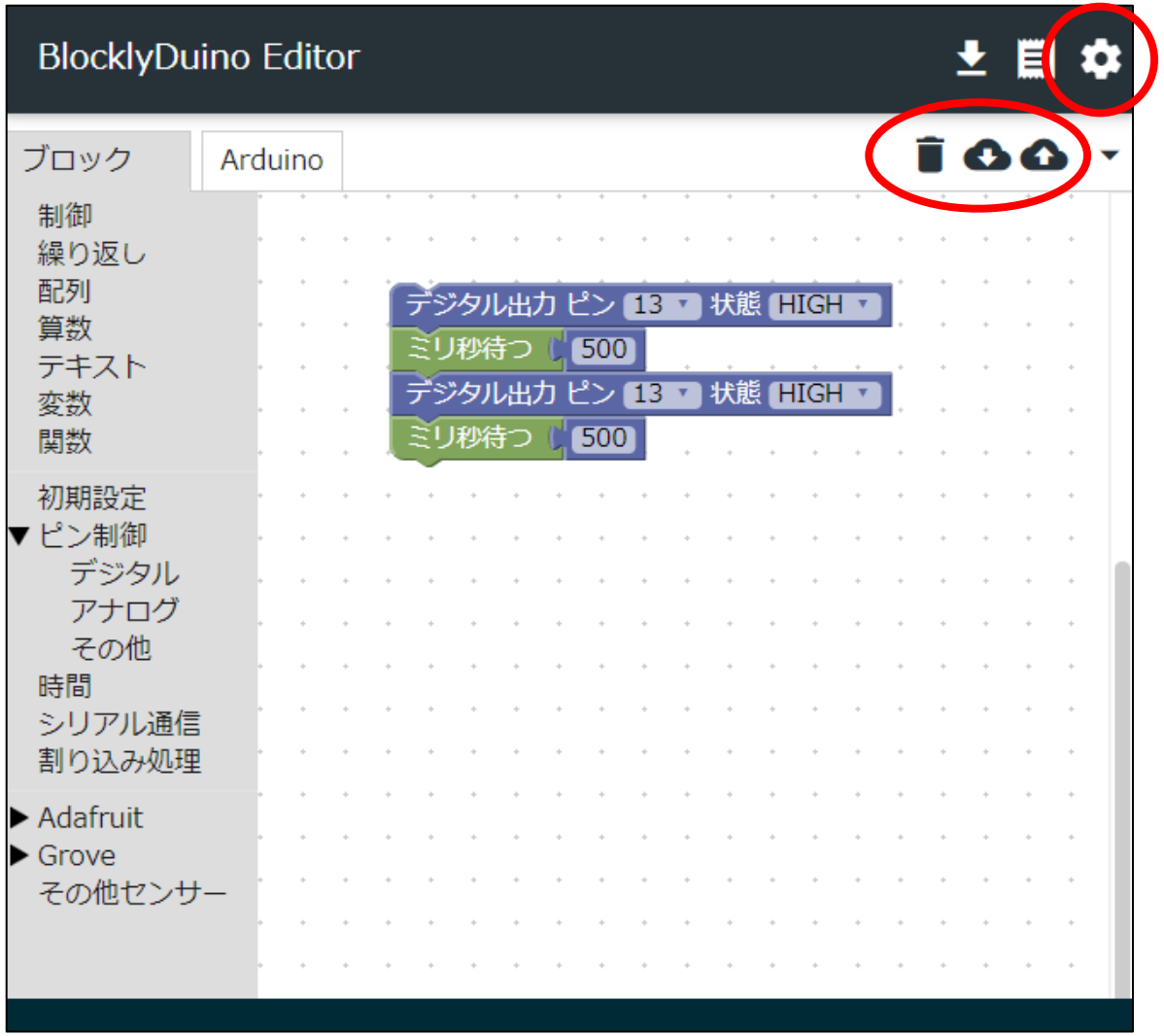
マイコンボード

プログラミング

- **コンパイル**
- **プログラムの書き込み**

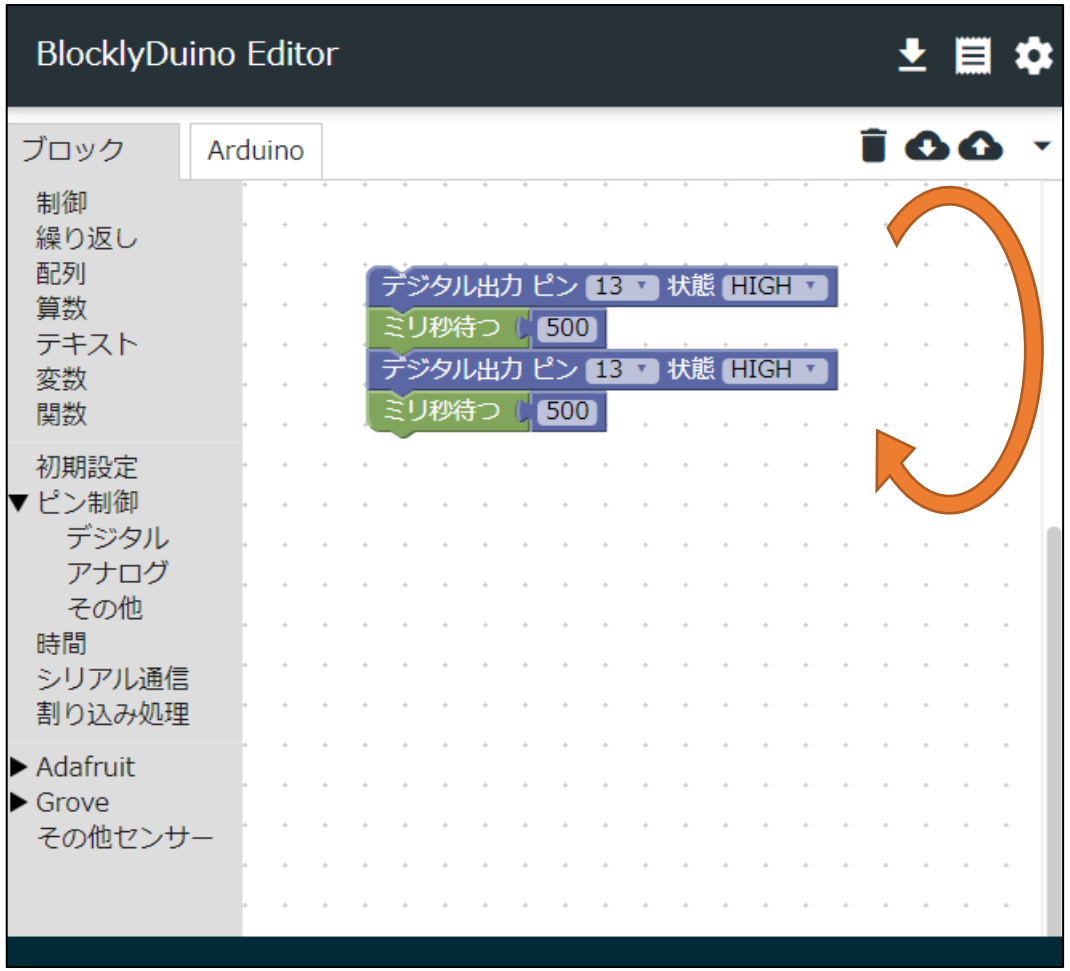
L チ カ

Blocklyduinoの使い方



- ゴミ箱
  - コードを全て消す
- ダウンロード
  - ブロックコードの保存
- アップロード
  - ブロックコードの読み込み

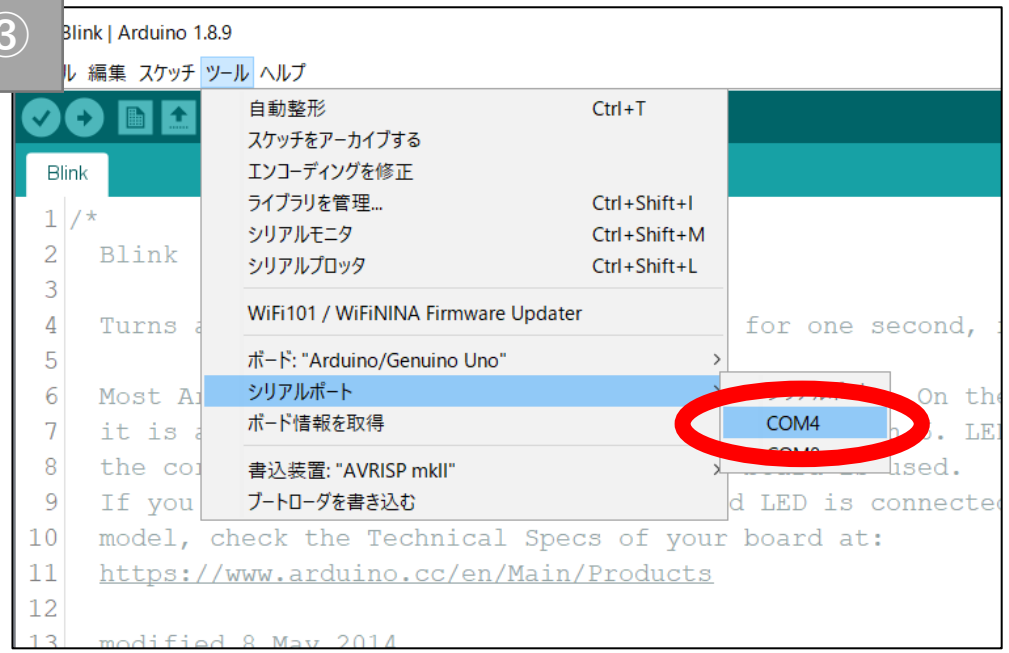
Lチカ ＝ 組込みの世界のHello World!!



- ファイル名
  - 01_blink
- Lチカするコード
  - 13pinに繋がる1秒周期で赤LEDが点滅

シリアルポートを選択して書き込み

③



③シリアルポートを選択する

ツール＞シリアルポート＞COM**X**

④

④書き込みボタンを押す

① IO出力

GPIOとは

- GPIO
  - General-purpose input/output"の略
  - 日本語で汎用入出力

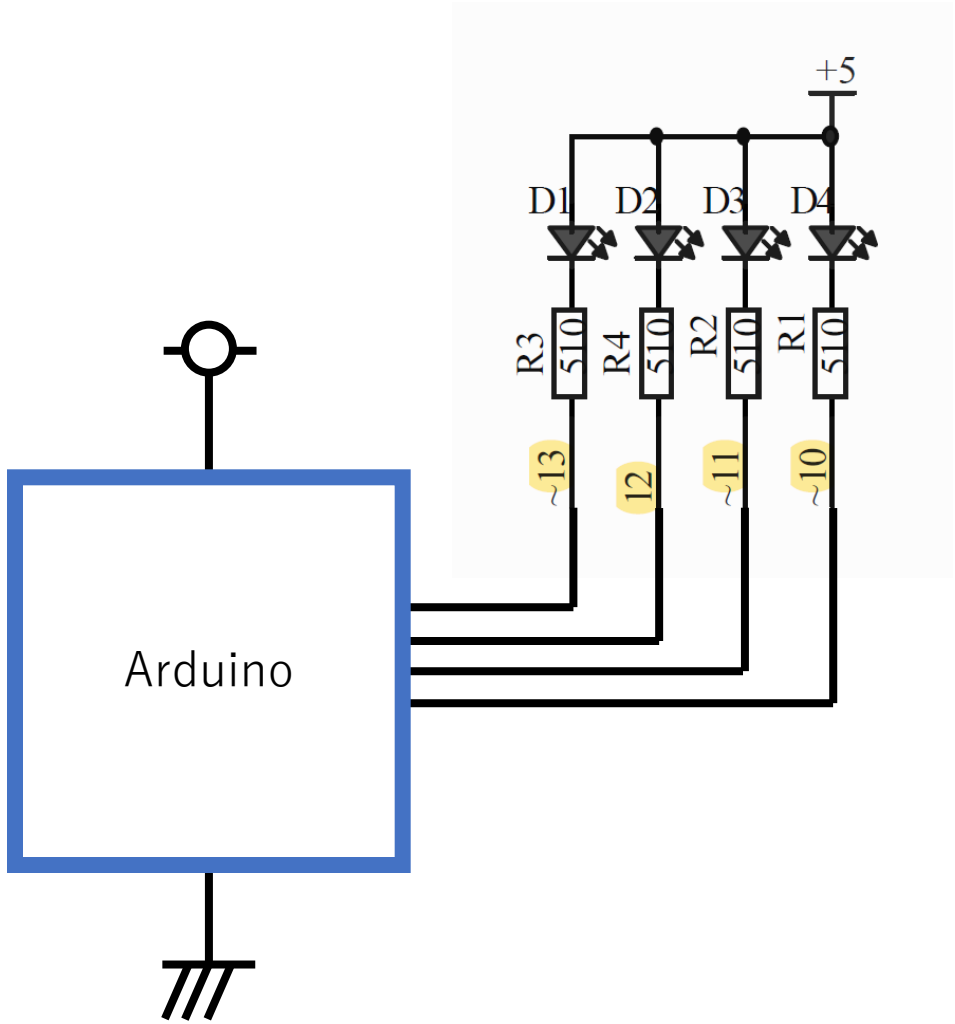
- ArduinoのIOポートで入出力を行う
  - Ouputの場合　…　LEDの点灯/消灯、モータのON/OFFなど
  - Inputの場合　…　スイッチのON/OFF入力など

- Arduino UNOの場合
  - Pin0～13, A0～A5の18ピンを利用可能

GPIO出力で制御できるものの事例

- LED
- 赤外線LED
- 7セグメントLED
- ブザー
- モータ

LEDを制御する



- Arduinoの10～13ピンに割り当て
  - シンク電流による駆動
- Arduino(ATmega328)のシンク電流は？

練習

1. 4つのLED(D1～D4)に対して全点灯→全消灯を繰り返す制御
2. 4つのLED(D1 ～D4)を交互に点灯させる
    1. D1, D3をオン，D2, D4をオフ
3. 4つのLED(D1 ～D4)を上から下へ1つずつ点灯させる．最後に全消灯
4. 4つのLED(D1 ～D4)を上から下へ1つずつ点灯させる．(常時1つだけ点灯の状態)
5. 4つのLED(D1 ～D4)を下から上へ1つずつ点灯させる．(常時1つだけ点灯の状態)

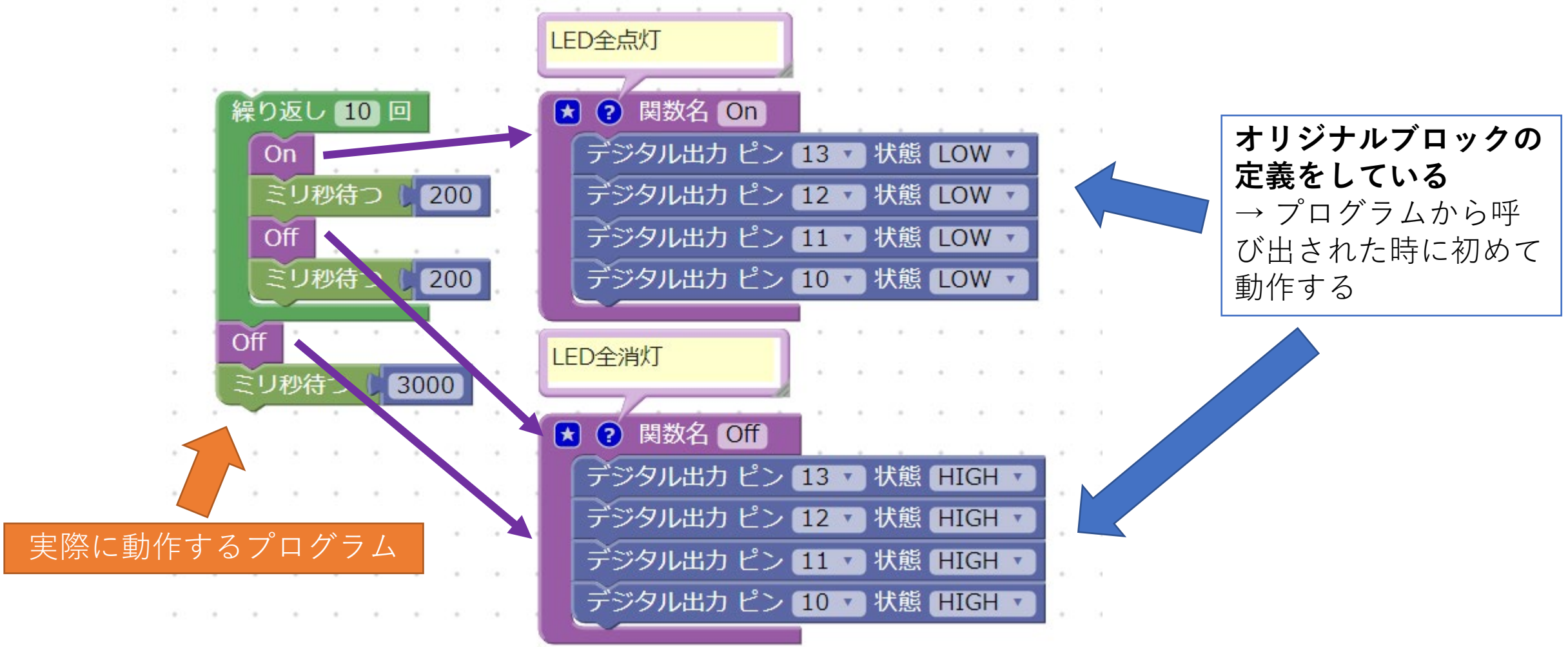
練習②

6. 上から下→下から上を繰り返す
7. 上から下→下から上→全点灯→全消灯→全点灯→全消灯→全点灯→全消灯
8. 「上から下」を3回繰り返し後，「下から上」を2回繰り返す

「関数」の活用

- 何度も同じ処理を記述するのは非効率的

- 何度も呼び出される同じ処理をまとめて、オリジナルブロックを作成するできる機能
- オリジナルブロックを都度呼び出す事で，記述量が格段に減る
  - 加えて，処理の変更時に変更する量を格段に減らす事が可能に


LED全点灯
関数名 On
デジタル出力 ピン 13 状態 LOW
デジタル出力 ピン 12 状態 LOW
デジタル出力 ピン 11 状態 LOW
デジタル出力 ピン 10 状態 LOW
LED全消灯
関数名 Off
デジタル出力 ピン 13 状態 HIGH
デジタル出力 ピン 12 状態 HIGH
デジタル出力 ピン 11 状態 HIGH
デジタル出力 ピン 10 状態 HIGH
繰り返し 10 回
On
ミリ秒待つ 200
Off
ミリ秒待つ 200
Off
ミリ秒待つ 3000
実際に動作するプログラム
オリジナルブロックの定義をしている
→ プログラムから呼び出された時に初めて動作する

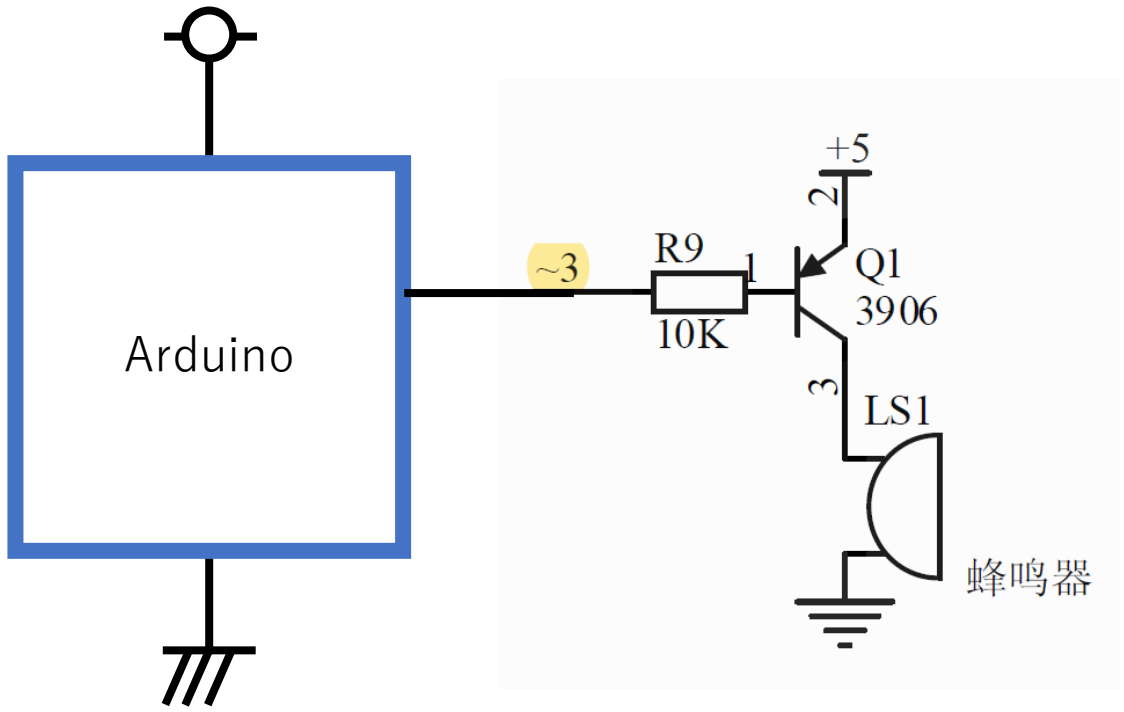
「繰り返し」

- プログラミングの基本要素「逐次」「分岐」「繰り返し」のうちの１つ

- 活用の場面
  - **同じ処理を決められた回数，繰り返して実行する**
    - 例）LEDを5回点滅させる
  - **同じ処理を特定の条件を満たすまで，繰り返して実行する**
    - 例）ボタンが押されている間，LEDを点滅させる

- 参考：
  - 現状のプログラムは**無条件に**繰り返して実行されている状態

ブザーを制御する



- Arduinoの3ピンに割り当て
  - HIGHにすると，ブザーが＿＿＿＿＿
  - LOWにすると，ブザーが＿＿＿＿＿
- サンプルコード

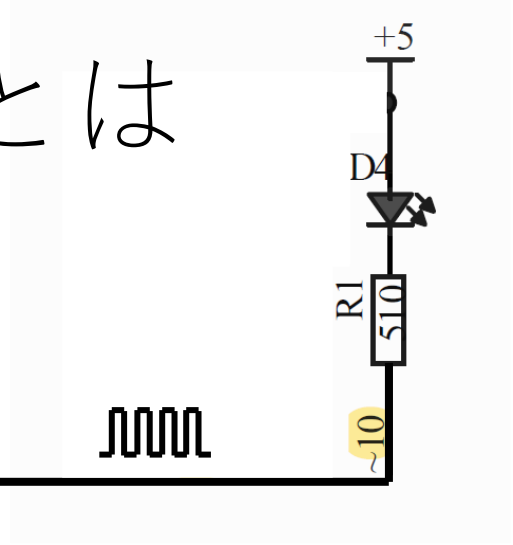
- **setup()への記述**
  - 電源投入時に1度だけ実行される処理
  - 初期化用途などに利用される

練習

1. ブザーを0.2秒間鳴らした後に，2.8秒間停止する処理を実現しましょう
2. １の課題に加えて，ブザーが鳴動している間はLED(D4)が点灯するプログラムを作成しましょう
3. ２の課題に加えて，LED(D1)を1秒周期で点灯するプログラムを作成しましょう

①-2 PWM制御

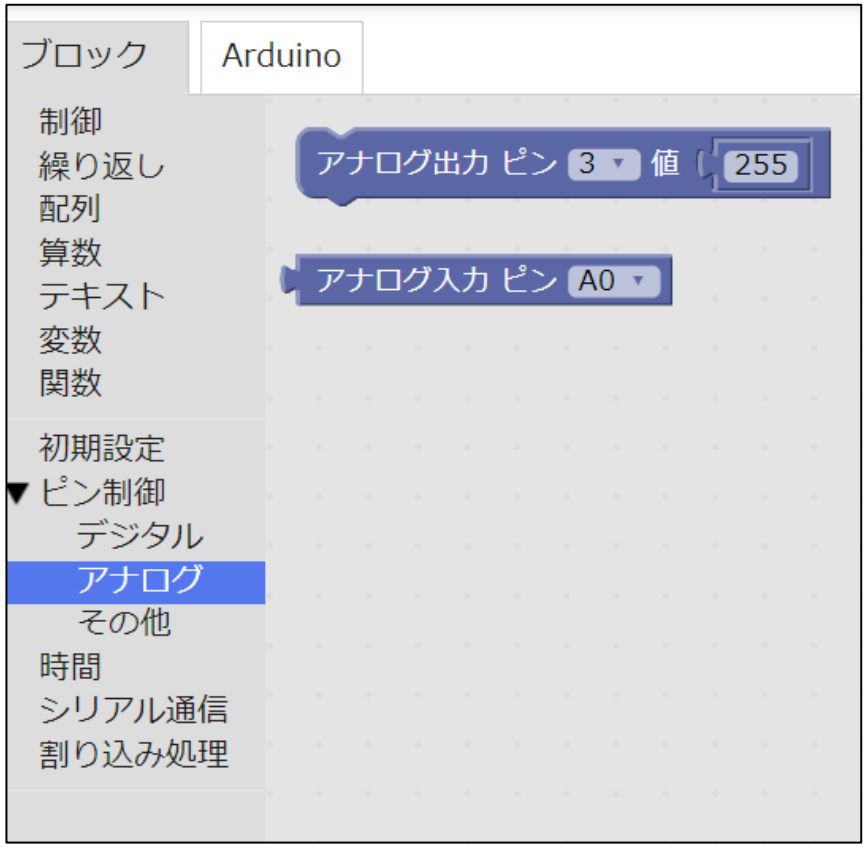
PWM制御とは



- Pulse Width Moduration**の略**
  - 制御対象に流す電流を細かくON/OFFさせてTotalで流れる電流量を調整することで**LEDの輝度**や**モータの回転速度**の変化が生じる
- Duty比とは
  - ON時間とOFF時間の比
- 同じ周期でDuty比の増減で制御を行う

アナログ出力(PWM)ブロックを使用する

- Arduino UNOには一部のピンにPWM機能が割付らえている
  - D3,5,6,9,10,11ピンの合計６ピン



値：0～255

値が0の時　　… 0V出力(Duty比：0%)
値が255の時　… 5V出力(Duty比：100%)

例：LEDの明るさ段階的に変化させる

- 10ピンをPWMピンとして利用し，LED4(D4)の明るさを段階的に変更するプログラムを作成する

- 確認
  - LEDの光は段階的に明るくなった？暗くなった？
  - この動作になる理由は？
- 練習
  - LED3(D3)にLED4とは逆の動作の点灯をさせましょう

② IO入力

GPIOとは

- GPIO
  - General-purpose input/output"の略
  - 日本語で汎用入出力

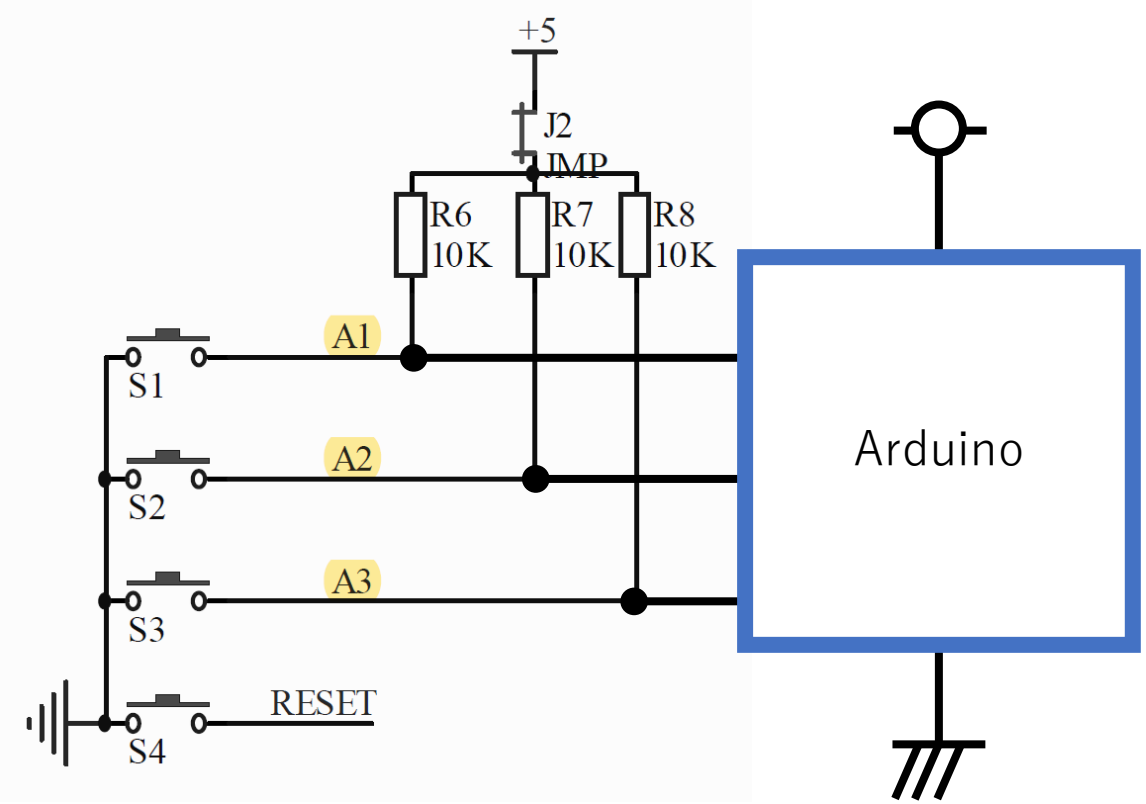
- ArduinoのIOポートで入出力を行う
  - Ouputの場合　…　LEDの点灯/消灯、モータのON/OFFなど
  - Inputの場合　…　スイッチのON/OFF入力など

- Arduino UNOの場合
  - Pin0～13, A0～A5の18ピンを利用可能

GPIO入力で制御できるものの事例

- スイッチ
  - ボタンスイッチ(プッシュスイッチ)
  - トグルスイッチ
- ドアの開閉センサ

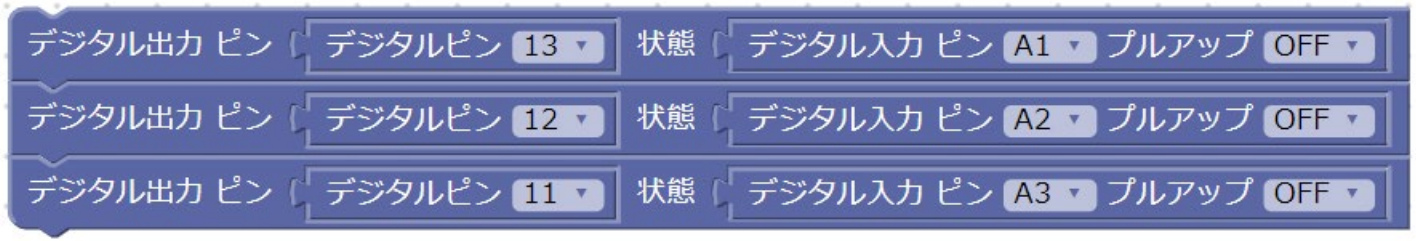
プッシュスイッチ(ボタン)を制御する



- ArduinoのA1,A2,A3ピンに割り当て
  - ボタン押下　　：＿＿＿＿＿＿
  - ボタン未押下　：＿＿＿＿＿＿

- RESETボタンはマイコンのRESETピンに接続
  - マイコンを再起動させる

例１：ボタンとLED制御



- ファイル名：button1.xml
- 動作
  - ボタンを押下したらLEDを点灯する
  - ボタンを離したらLEDを消灯する

例２：ボタンとLED制御

- ファイル名：button2.xml
- 動作
  - **例１（button1.xml）と同じ**
  - ボタンを押下したらLEDを点灯する
  - ボタンを離したらLEDを消灯する

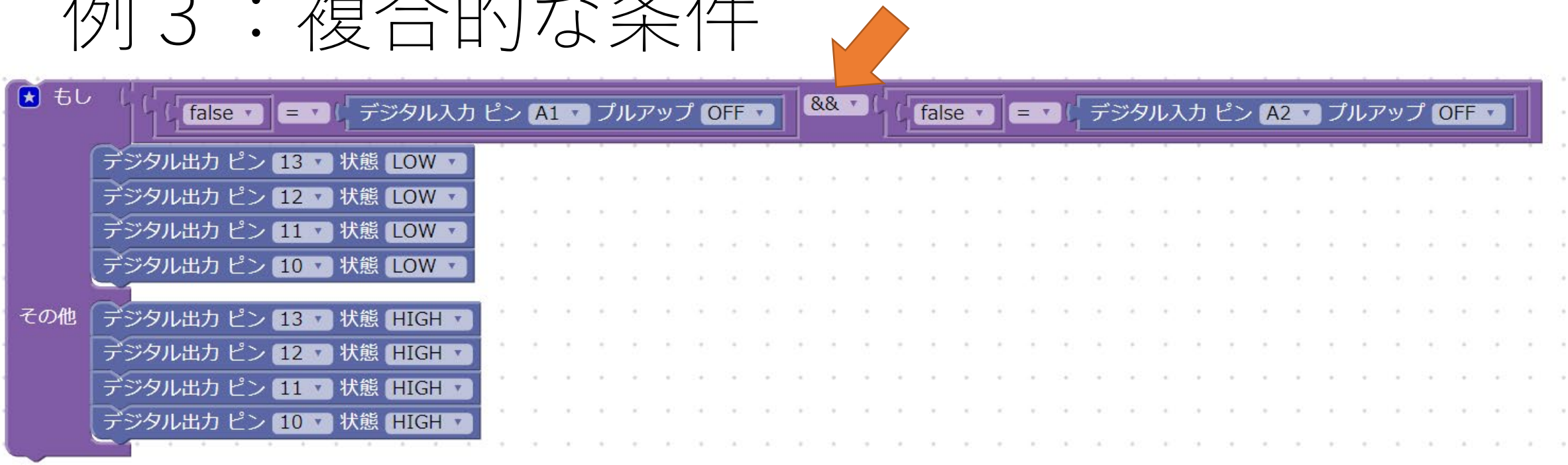
- 試してみよう！
  - 論理を反転させ，以下の動作にする
    - ボタンを押下したら**LEDを消灯**する
    - ボタンを離したら**LEDを点灯**する

「分岐」

- プログラミングの基本要素「逐次」「分岐」「繰り返し」のうちの１つ

- 活用の場面
  - **ある条件を満たしたら，特定の処理を実行する**
    - 例１）ボタンを押したらブザーを鳴らす
    - 例２）暗くなったらLEDを点灯する
    - 例３）ボタンAとボタンBを押したらLEDを全て点灯する
    - 例３）温度が20 °C未満の時は，青LEDを点灯する
温度が20°C以上，30°C未満の時は，緑LEDを点灯する
温度が30°C以上の時は，赤LEDを点灯する

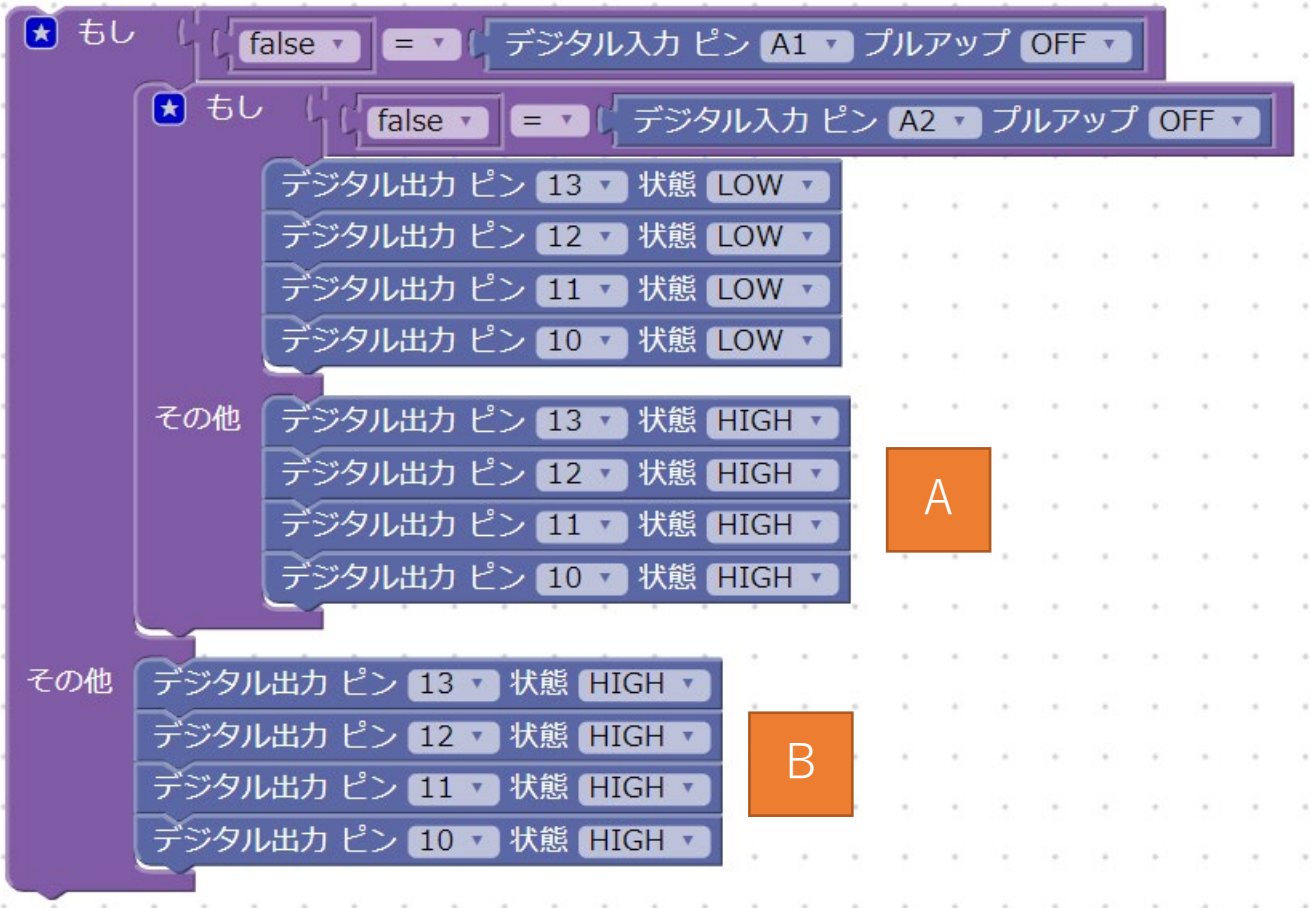
例３：複合的な条件



- ファイル名：button3.xml
- 動作
  - ボタン(S1)とボタン(S2)を両方とも押下したらLEDを全点灯する
  - それ以外の条件ではLEDを全消灯する

- 試してみよう！
  - **&&**を||に変更すると，どのような動作になりますか？

例３：複合的な条件(別解)



- 左図のような記述も可能
  - 条件分岐を**”入れ子”**にする考え方
- ただし，AとBの記述が冗長になる為，好ましくない

練習

ファイル名：button_ren01_XX.xml

1～4までは個別にプログラムを作りましょう！

1. ボタン(S1)を押している間，LED(D1～D4)が点灯する．ボタン(S1)を離すとLED(D1～D4)が消灯する
2. ボタン(S2)を押している間，ブザーが鳴る
3. ボタン(S3)を押している間，LED(D1～D4)が100msec周期で点滅する．
4. ボタン(S1)とボタン(S3)を同時に押している間，1sec周期で点滅する．
5. 1～4までの動作を全て1つのプログラムに集約する

発展練習①

ファイル名：button_ren2.xml

6. ボタン(S1)とボタン(S2)を同時に押している間，以下の動作を実現する
   - 4つのLED(D1 ～D4)を**上から下**へ1つずつ点灯させる．(常時1つだけ点灯の状態)
7. ボタン(S2)とボタン(S3)を同時に押している間，以下の動作を実現する
   - 4つのLED(D1 ～D4)を**下から上**へ1つずつ点灯させる．(常時1つだけ点灯の状態)
8. 全てのボタン(S1,S2,S3) を同時に押している間，4つのLEDを1sec周期で交互に点灯させる

---

9. 例３と同じ動作を&&ではなく，||で実現する

ファイル名：button_ren3.xml

発展練習②

ファイル名：button_ren4.xml

1. ボタン(S1)のON/OFFによりLEDアニメーションを制御する
    1. ON　　：4つのLED(D1 ～D4)を**上から下**へ1つずつ点灯させる．(常時1つだけ点灯の状態)
    2. OFF　　：4つのLED(D1 ～D4)を**下から上**へ1つずつ点灯させる．(常時1つだけ点灯の状態)

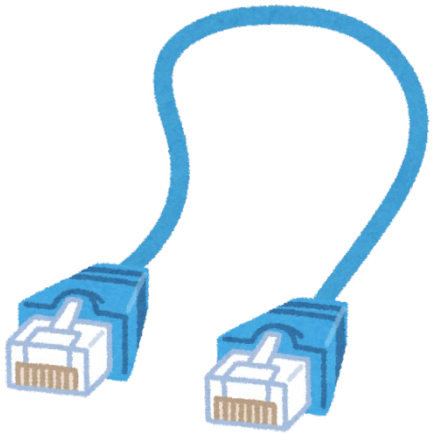
2. ボタン(S3) のON/OFFにより、LEDアニメーションの開始・停止をする電源ボタンの機能を実装する

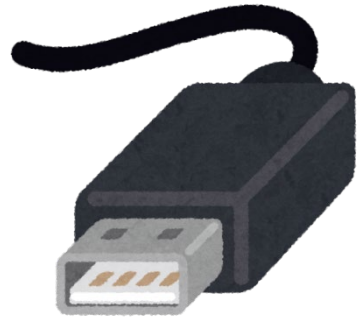
3. 各ボタンの入力により、LEDアニメーションの動作が変化するプログラムを作成する

	S1	S2	S3
ON	LEDアニメーション (上→下)	点滅周期を 500msec	LEDアニメーション の開始
OFF	LEDアニメーション (下→上)	点滅周期を 100sec	LEDアニメーション の停止

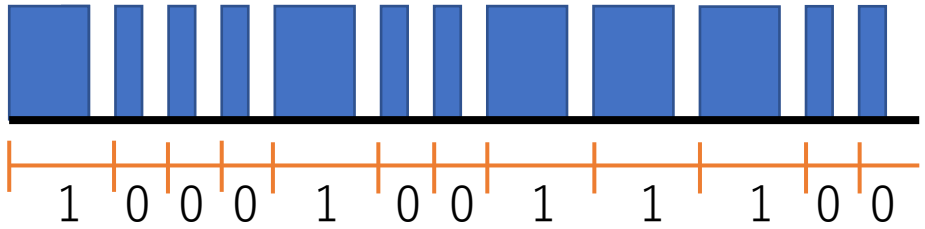
③-0 シリアル通信

シリアル通信とは





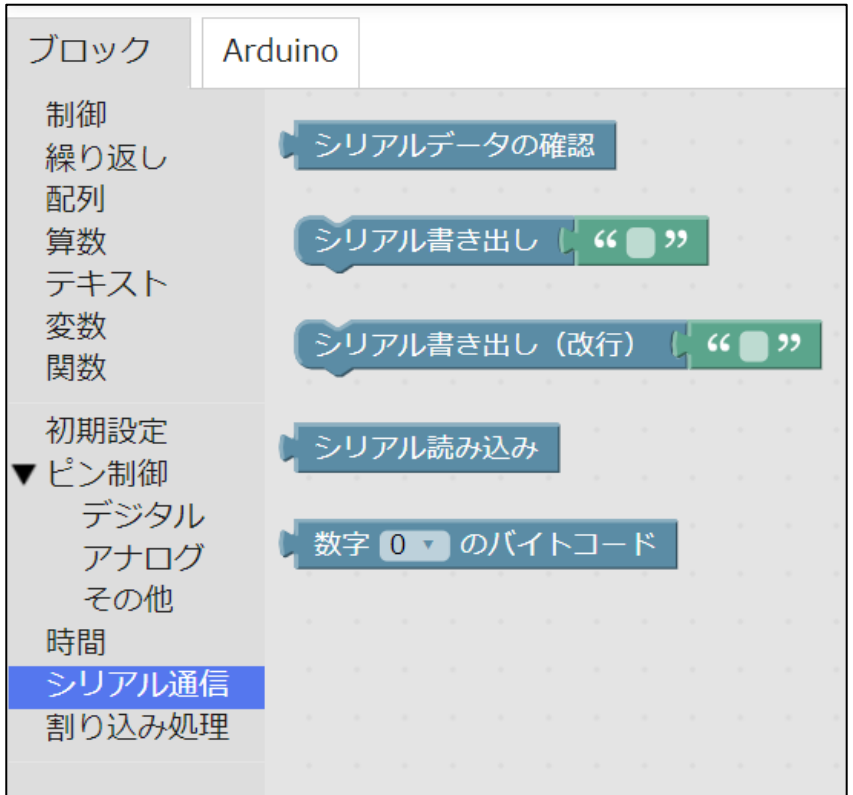
- シリアル通信には主に2つの意味で使われる
- 広義のシリアル通信
  - １本の信号線に対してデータを直列に送信する通信のこと



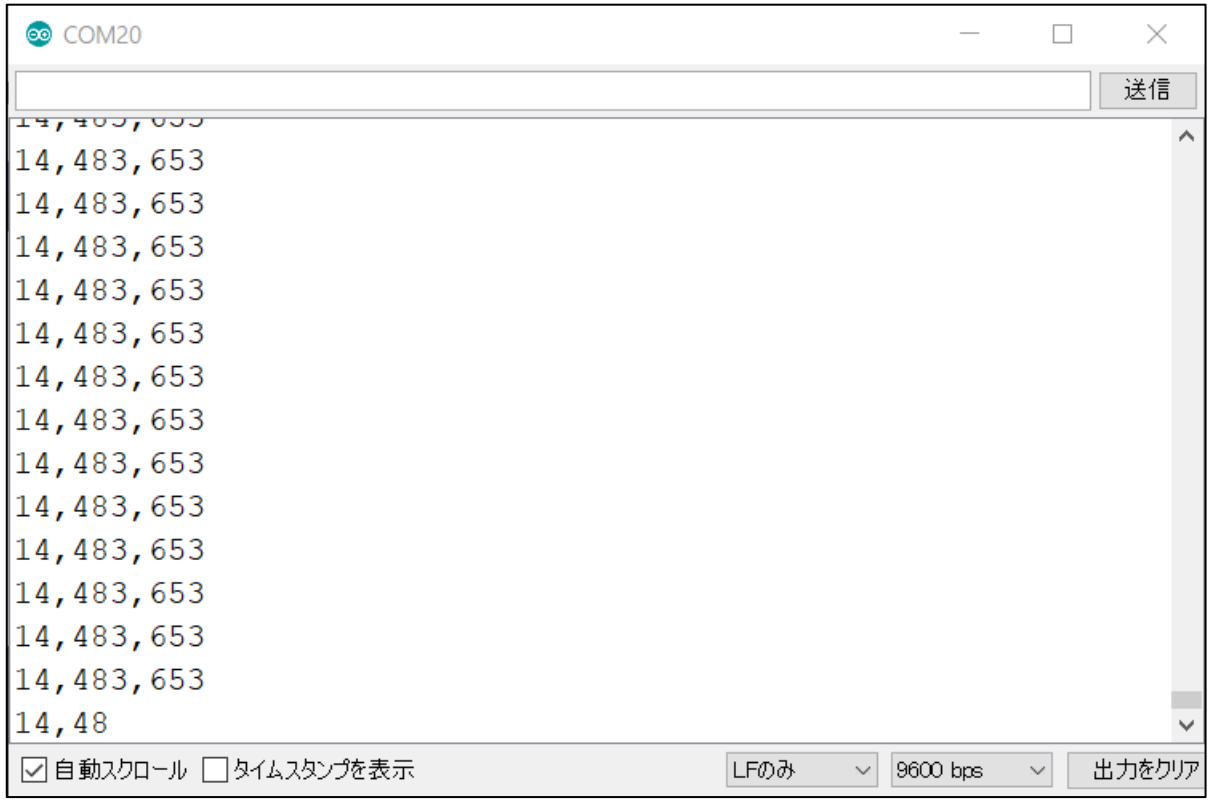
  - 例：USB, Ethernet, RS-232, MIDI, SerialATA, UART, SPI, I2Cなど
- 狭義のシリアル通信
  - RS232C通信ないし、UARTによる通信のこと
  - FAや組込み業界のエンジニアの中では一般的な呼び方

例１：文字を表示する

- シリアル通信を用いて，マイコンからPCへデータ転送を行うプログラム
  - 1000msecに一度，”hello”という文字列データを送信する



シリアルモニタ



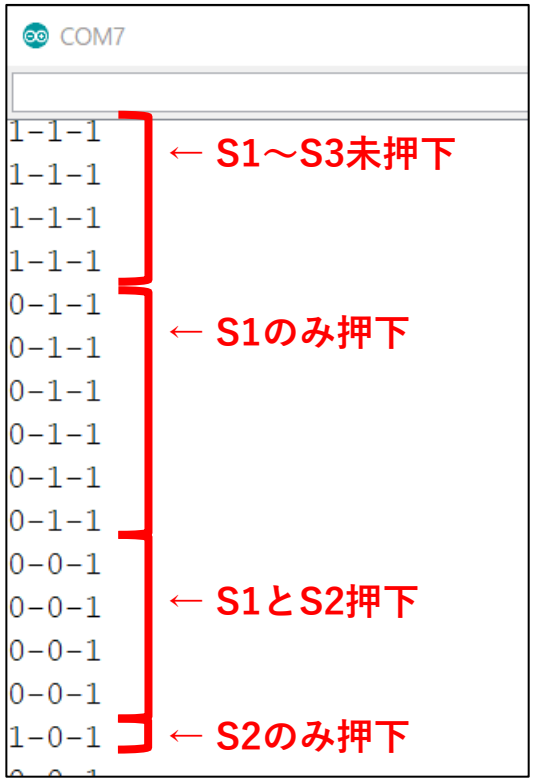
- シリアルモニタ
  - Arduinoから送信されたシリアル通信の内容を文字列で表示する

例２：ボタン(S1)の状態を表示する

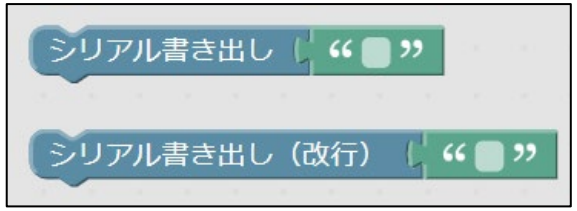
- シリアル通信を用いて，マイコンからPCへデータ転送を行うプログラム②
  - ボタン(S1)が押されていれば，文字”0”を送信.
  - ボタン(S1)が押されていなければ，文字”1” を送信.

練習

1. ボタン(S2)の状態を表示してみましょう
2. ボタン(S1～S3)の状態を表示してみましょう
   - 表示イメージ：

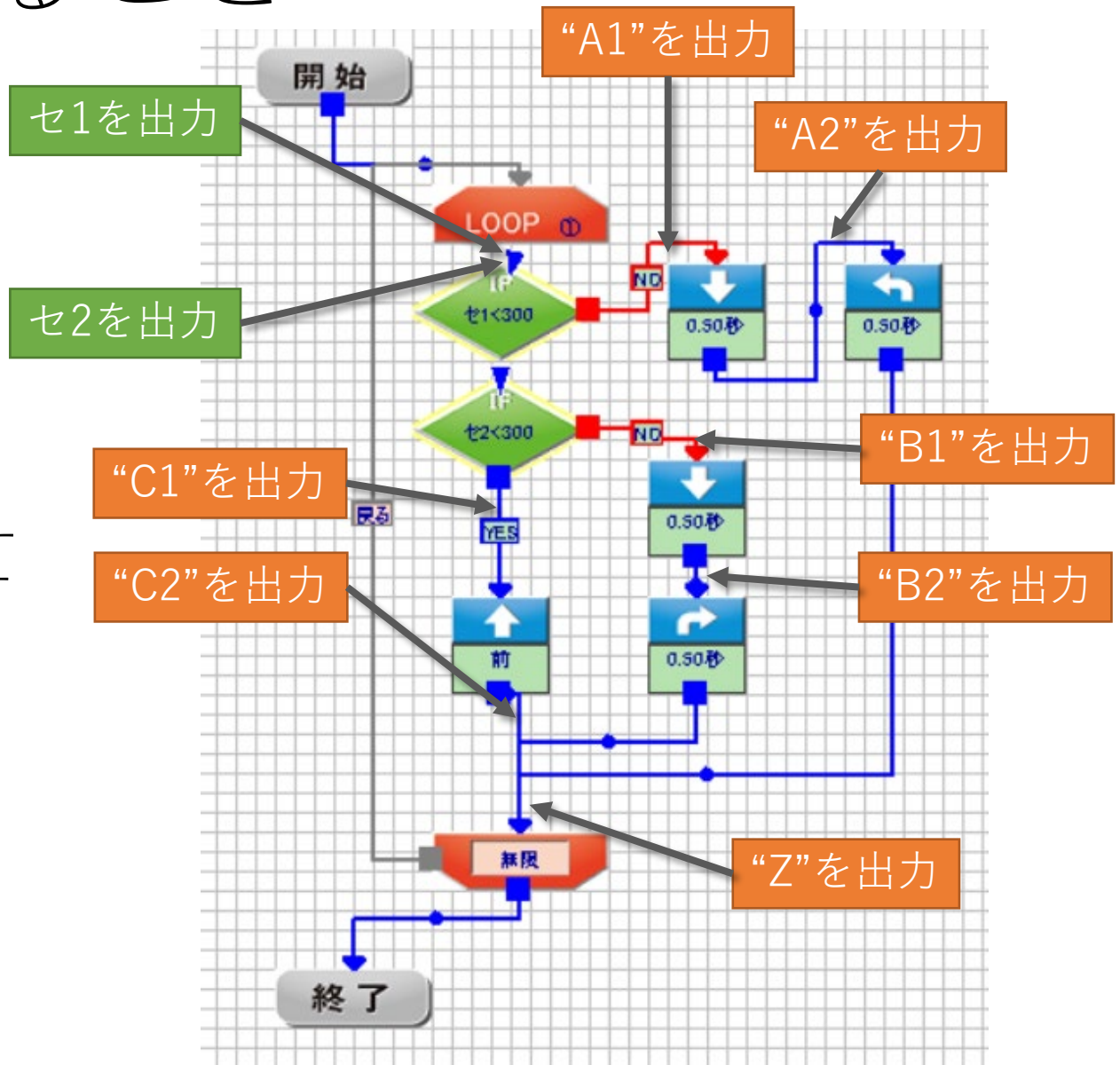


ヒント：「シリアル書き出し」を使いましょう



printfデバッグで出来ること

- 変数の中身を確認する
  - 例えば、ADから取得したラインセンサ値、距離センサ値の確認
  - カウンタの値の確認
  - 変化しやすい値は、その値によって処理が分岐することが主である
- 内部フローを確認する
  - 特定のif文の中に入ったら'a'を表示する、if文に入らなかったら'z'を表示する。など
  - 自身の作成したプログラムが意図したタイミングで分岐や繰り返しをしているかを判断



③-1 ADコンバータ

はじめに

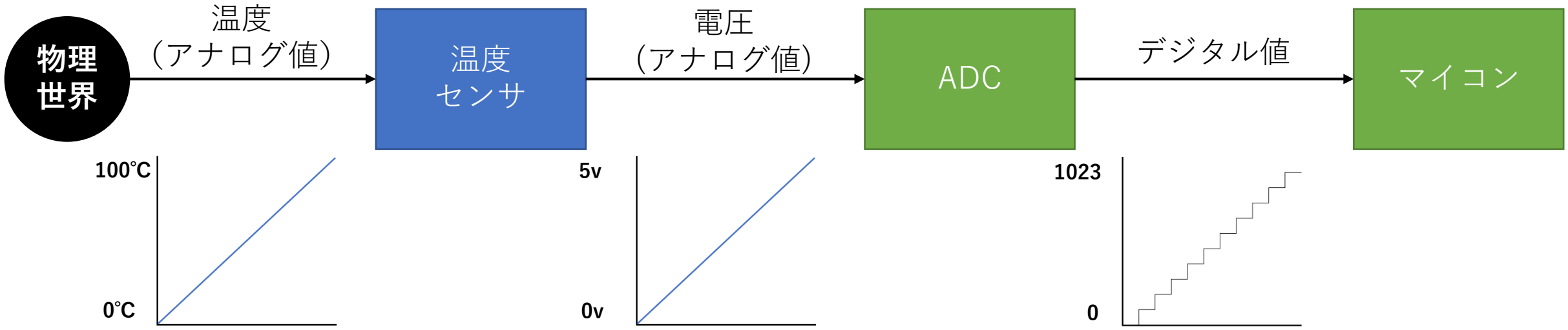
- ADコンバータとは、アナログ量をデジタル量に変換する機能です。多くのマイコンに内蔵されています。
- 物理世界のほとんどの物理量はアナログ量であり、温度・湿度・光量・音量・加速度など全て連続的なアナログ値になります。
- 組込み製品ではこれらのアナログ量を扱うことが多々ありますが、GPIOではHIGH / LOW の2値しか取得できない…
  - センサを扱うのであれば，より細かく値を取得したい！

ADコンバータを活用する！！

AD入力で制御できるものの事例

- 照度センサ
- 温度センサ
- 距離センサ
- 可変抵抗器
- 加速度センサ　ほか

アナログ値がデジタル値に変換されるまでの流れ



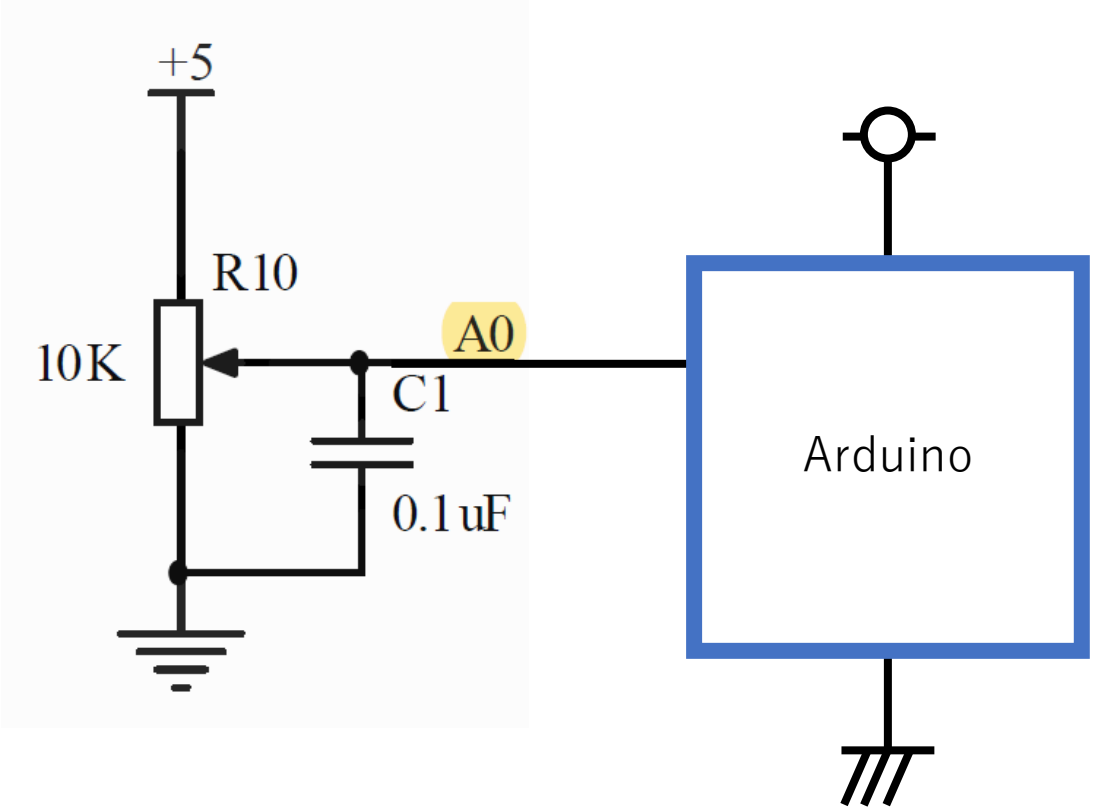
- センサ
  - アナログ量を電圧に変換するモノ
  - 変換できるアナログ量、電圧はセンサによって様々
  - 変換出力は電圧だけではなく、電流や抵抗値のことも
    - この場合、ADCとの間にI→VやR→Vの変換回路を入れる
- ADコンバータ(ADC)
  - 電圧をデジタル値に変換するモノ
    - **今回の場合、0～5vの入力電圧を0～1023のデジタル値の出力に変換する**
  - 分解能が高ければ高い程、より細かくデジタル値で表現できる

図の事例の場合、マイコンが…
　ADCから0を取得したら0℃
　ADCから1023を取得したら100℃
ということになる

分解能について

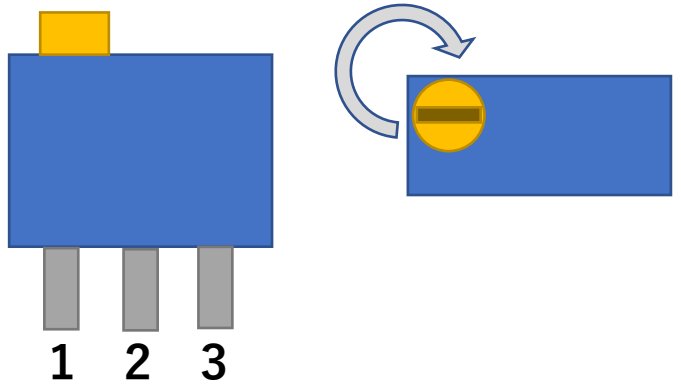
- ADコンバータの性能を指す。分解能が高い程、より細かくアナログ値をデジタル値に変換できる。
- 通常、ビット数で表され、8bit, 10bit, 12bit, 16bit…というように表現される。
- このビット数は、デジタル量で表現できる大きさを表す。
  - 仮に2bitのADコンバータであれば、4段階に分けてアナログ量を表現できる
  - 温度センサであれば、「冷たい・ぬるい・暖かい・熱い」の四段階程度
- 分解能が高ければより細かい正確な測定ができるのか？
  - 高度な測定に必要な要素の1つであるが、他にも必要とされる要素がある為、分解能1つでは高度な測定は実現できない

ボリューム(多回転型)を制御する



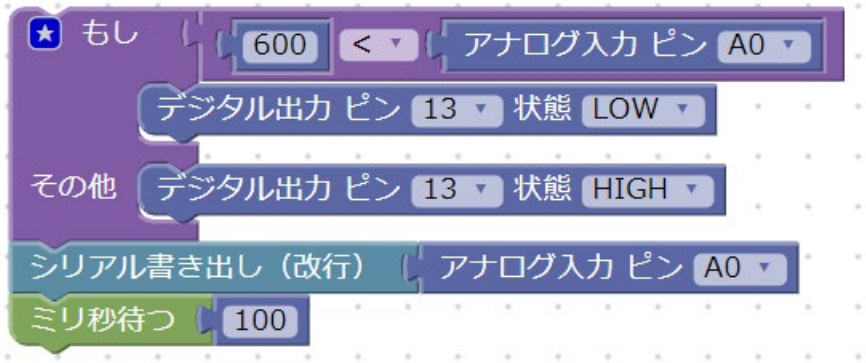
総和はボリュームの型番によって変わります
100Ω, 1kΩ, 10kΩ, 100kΩなどがあり

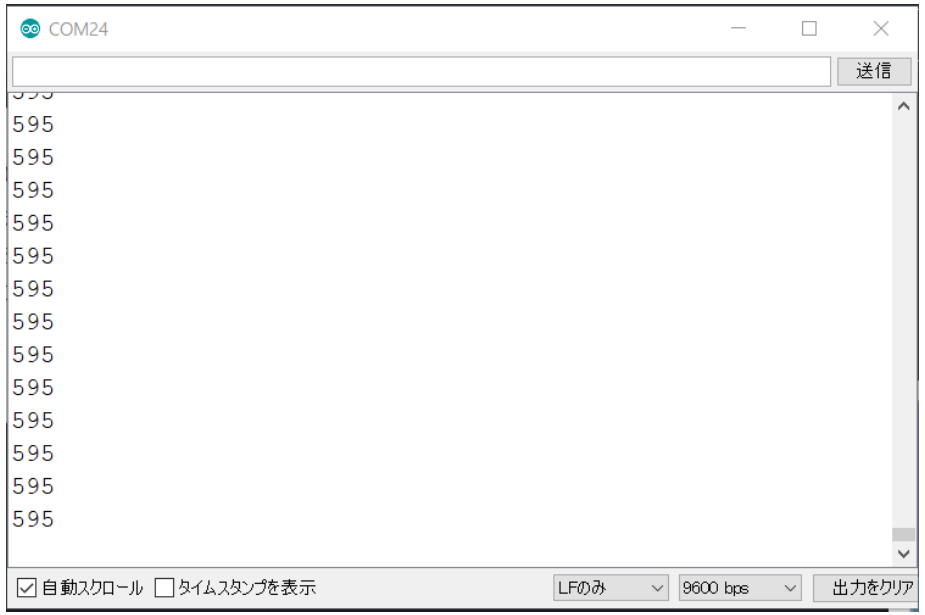
- ArduinoのA0に割り当て
  - 半時計周りに回す：電圧が小さくなる
  - 時計回りに回す　：電圧が大きくなる



ツマミ	1-2間	2-3間
反時計回りにX回，回す	10kΩ	0Ω
0回転	0.5kΩ	0.5kΩ
時計回りにX回，回す	0Ω	10kΩ

例：ボリューム制御

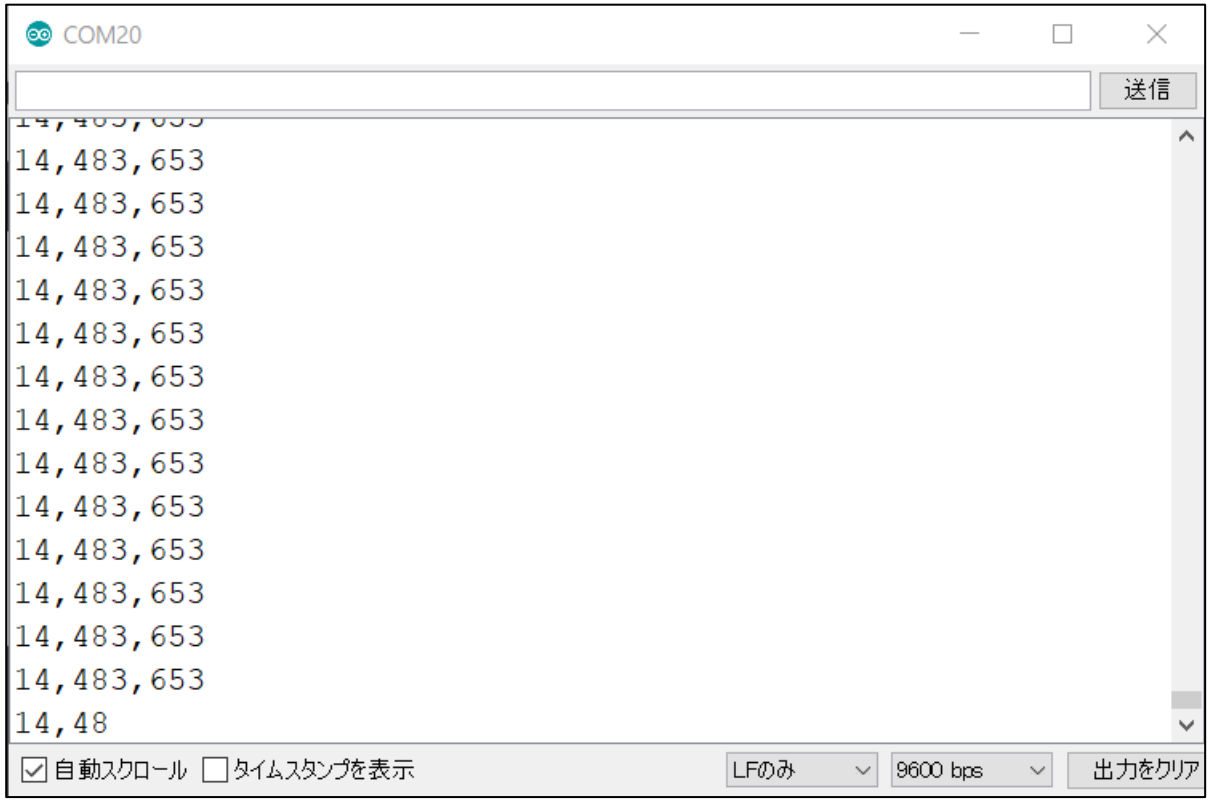


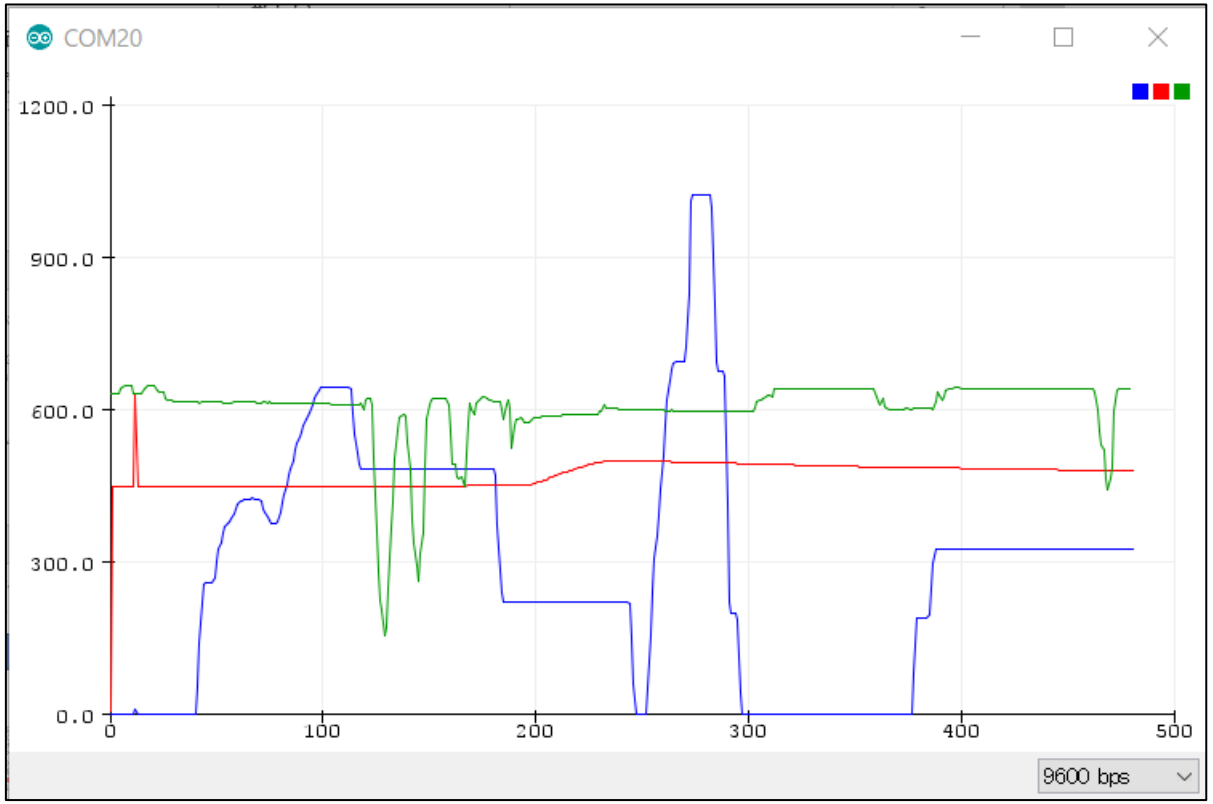


- ファイル名：adc.xml
- 動作
  - ボリュームから出力される電圧値を取得・AD変換を行い，ADカウント値をシリアル通信で出力する
  - ADカウント値が，600を超えたらLED(D1)を点灯する

- 比較演算子
  - X＝Y　　… XとYが同じ値
  - X≠Y　　… XとYが同じ値
  - X<Y　　… YはXより大きい
  - X≦Y　　… YはX以上
  - X>Y　　… XはY未満
  - X≧Y　　… XはY以下

シリアルモニタとシリアルプロッタ





- シリアルモニタ
  - Arduinoから送信されたシリアル通信の内容を文字列で表示する
- シリアルプロッタ
  - Arduinoから送信されたカンマ区切りのデータをグラフにプロットして表示する

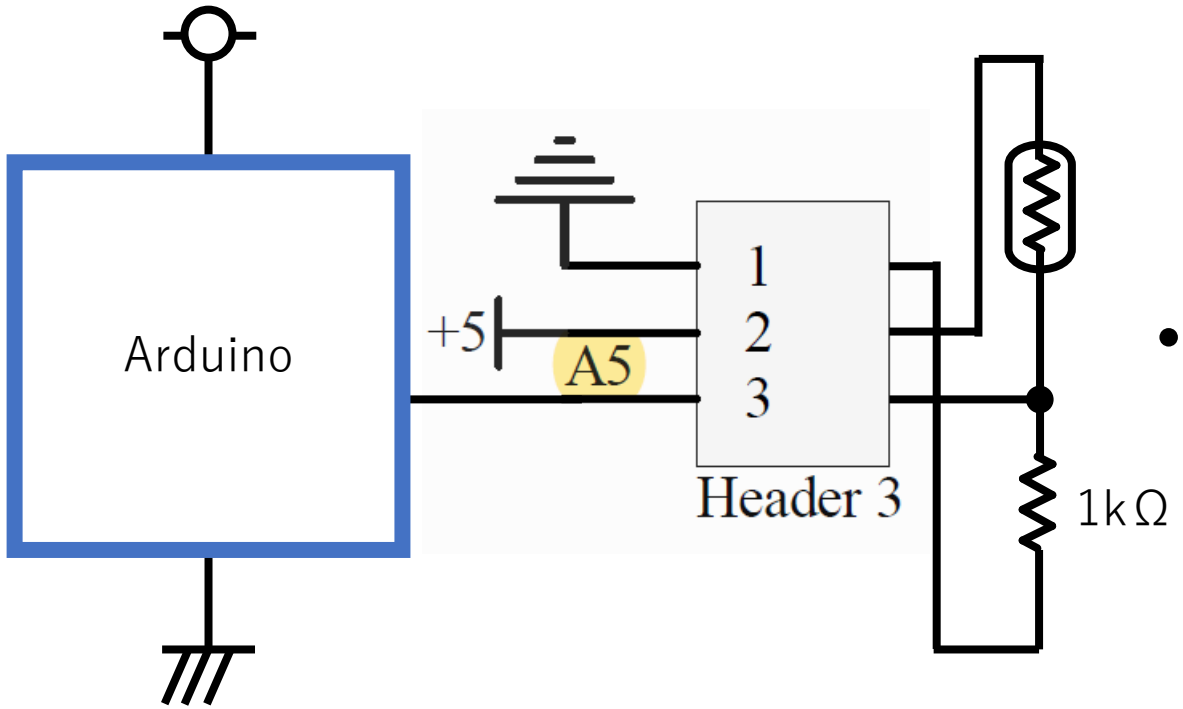
練習

ファイル名：adc_ren1.xml

1. ボリュームのADカウント値が，600を超えたらLED(D1)を点灯する．下回ったらLED(D1)を消灯する．
2. ボリュームのADカウント値が，610を超えたらLED(D2)を点灯する．下回ったらLED(D2)を消灯する．
3. ボリュームのADカウント値が，620を超えたらLED(D3)を点灯する．下回ったらLED(D3)を消灯する．
4. ボリュームのADカウント値が，630を超えたらLED(D4)を点灯する．下回ったらLED(D4)を消灯する．

③-2 ADコンバータ
(光センサ)

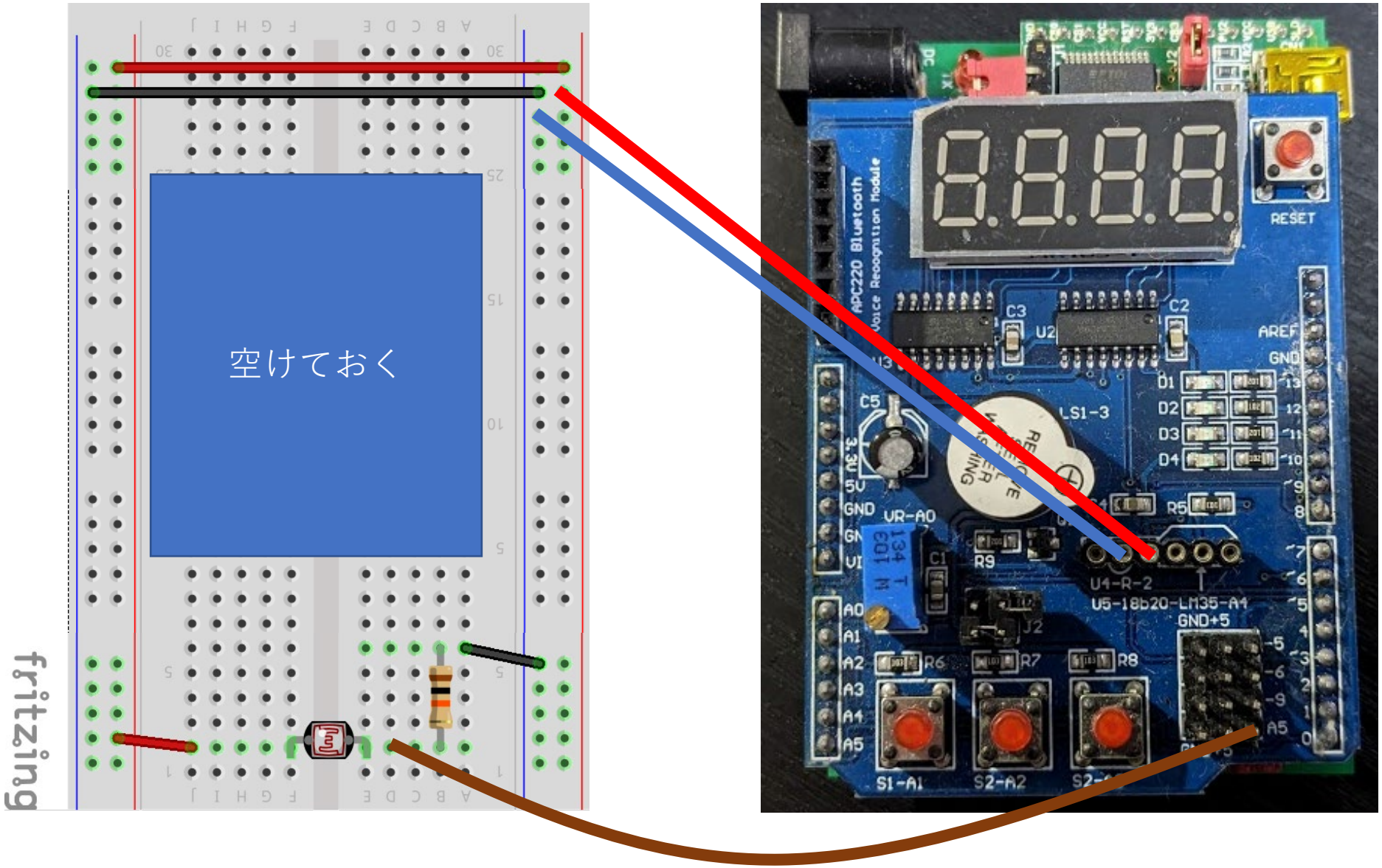
光センサを制御する



- CdS
  - 暗いと抵抗値は＿＿＿＿＿＿
  - 明るいと抵抗値は＿＿＿＿＿＿
- ArduinoのA5に割り当て
  - 暗い　　… 電圧が＿＿＿＿＿＿なる
  - 明るい　… 電圧が＿＿＿＿＿＿なる

回路拡張：要ブレッドボード

実態配線図・例


空けておく
fritzing
APC220 Bluetooth
Voice Recognition Module
RESET
LS1-3
VR-A0
U4-R-2
U5-18b20-LM35-A1
GND+5
S1-A1
S2-A2

練習

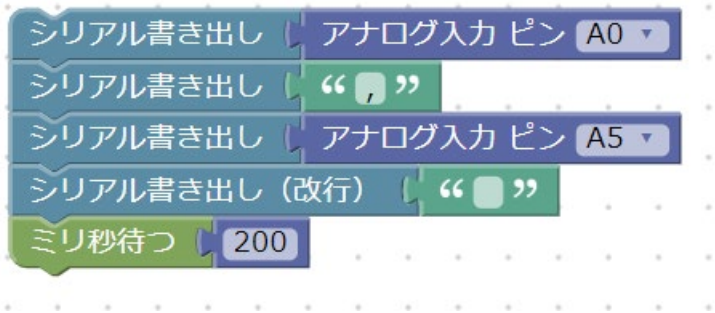
ファイル名：adc_ren2.xml

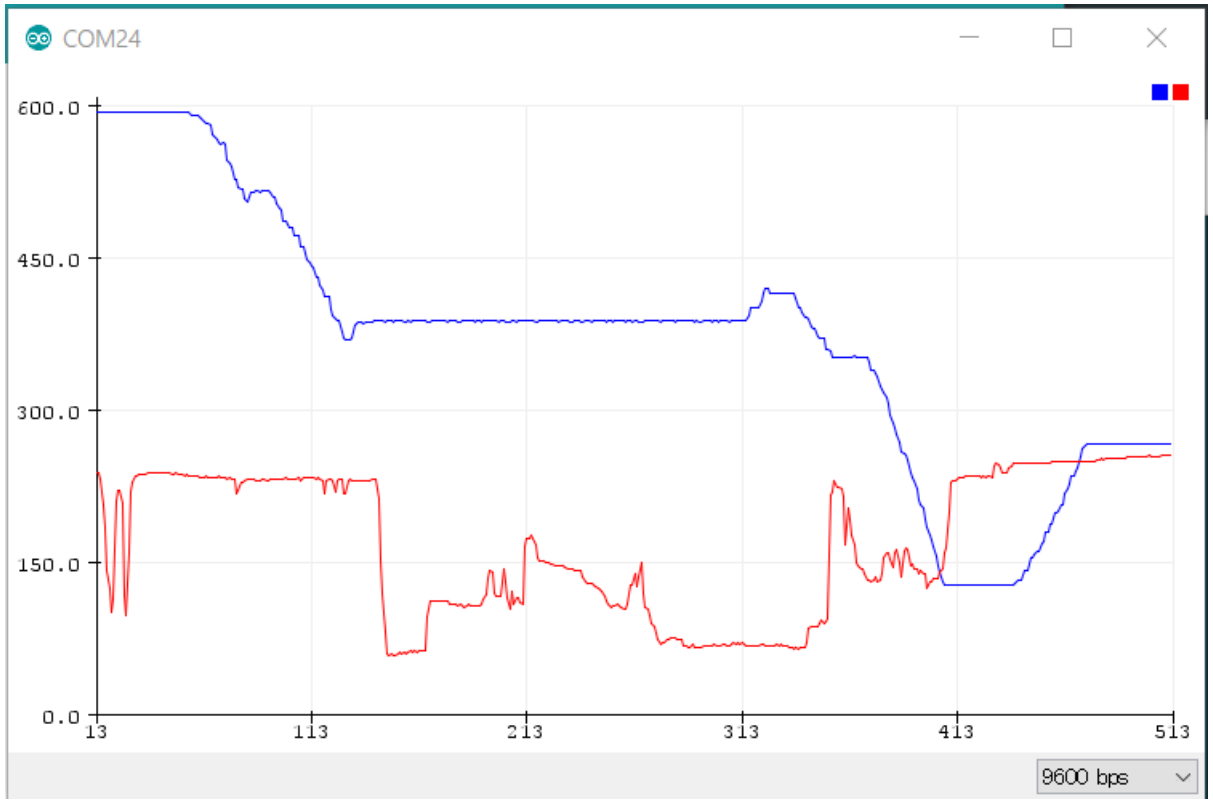
1. 回路をブレッドボードに実装しましょう！
   - 回路作成
   - 図面への消込
   - ショートチェック

2. ボリュームのプログラム例を改変し，光センサの値をシリアル通信で表示するプログラムを作成しましょう
3. 閾値を改変し，暗くなったらLED(D2)が点灯するように変更しましょう
4. 1,2で作ったプログラムとボリュームのプログラム例を1つのプログラムにしてまとめましょう

シリアルプロッタで2つの値を表示

- シリアル通信でPCに出力する際に，カンマ区切りにする事で複数の値をグラフ表示する事が可能です



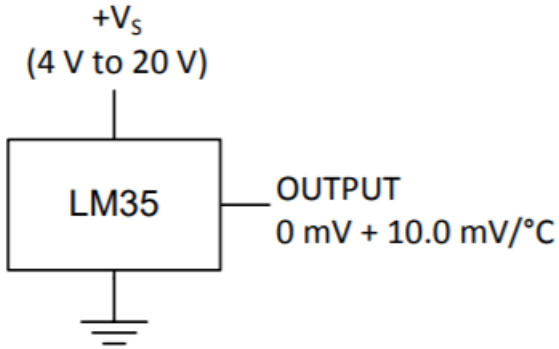


③-3 ADコンバータ
(温度センサ)

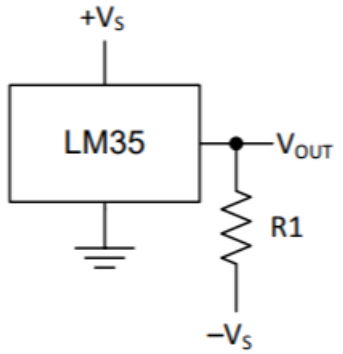
温度センサを制御する

- 温度センサ(LM35)
  - 2～150°Cの温度測定が可能なセンサ
    - （回路を工夫すると，-55～150°Cで測定可）
  - 測定誤差は±0.5°C
  - 出力はリニアに変化する
    - 0mV+10.0mv/°Cでの変化

基本的な摂氏温度センサ
(2℃～150℃)

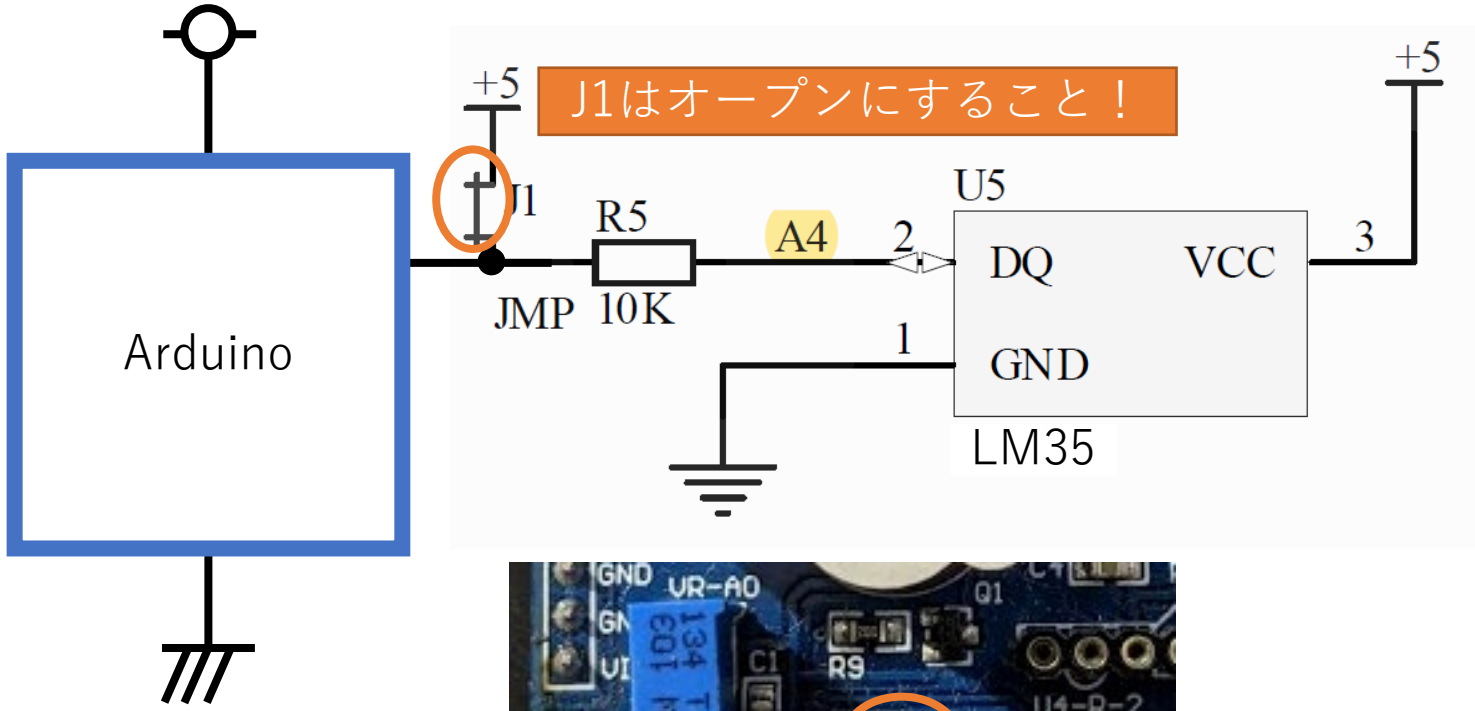


全範囲の摂氏温度センサ

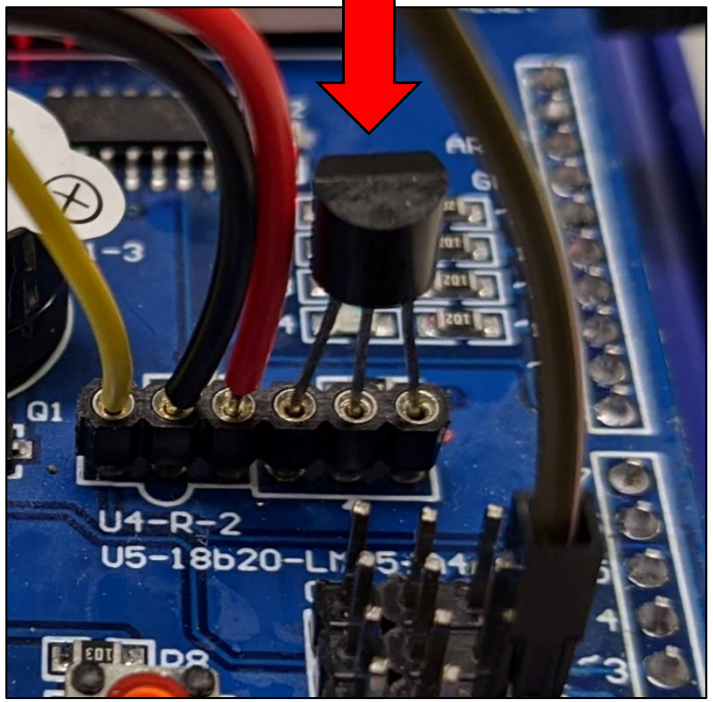


$R_1 = -V_S$ / 50μAを選択
150℃においてV_{OUT} = 1500mV
25℃においてV_{OUT} = 250mV
–55℃においてV_{OUT} = –550mV

温度センサの実装



取付向き注意！！
故障の可能性大



練習

ファイル名：adc_ren3.xml

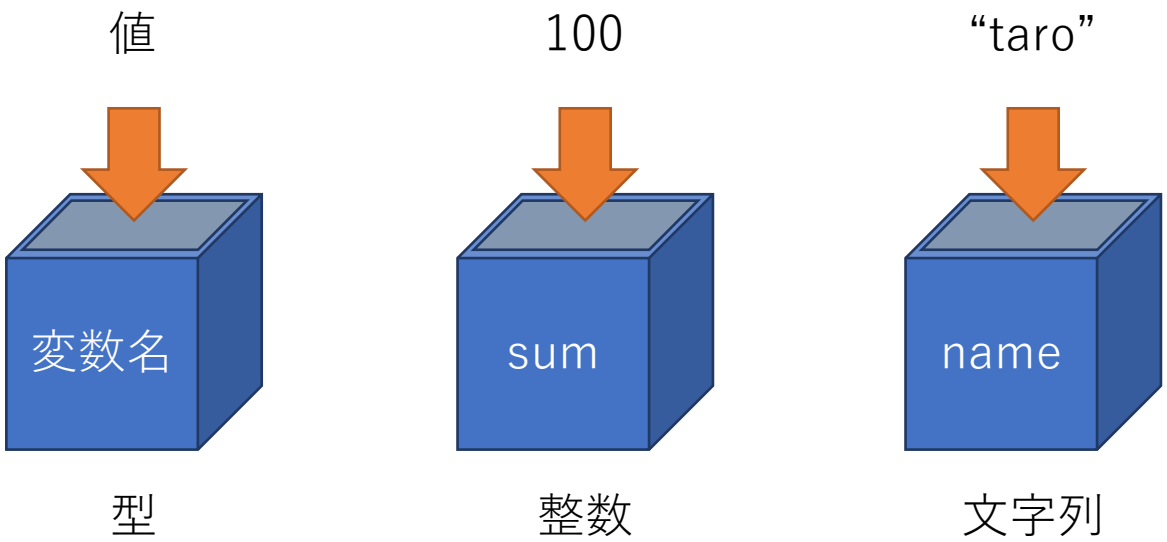
1. 回路をブレッドボードに実装しましょう！
   - 回路作成
   - 図面への消込
   - ショートチェック

2. ボリュームのプログラム例を改変し，温度センサの値をシリアル通信で表示するプログラムを作成しましょう

何もしない時	
ドライヤーで５秒間あてた時	

「変数」とは

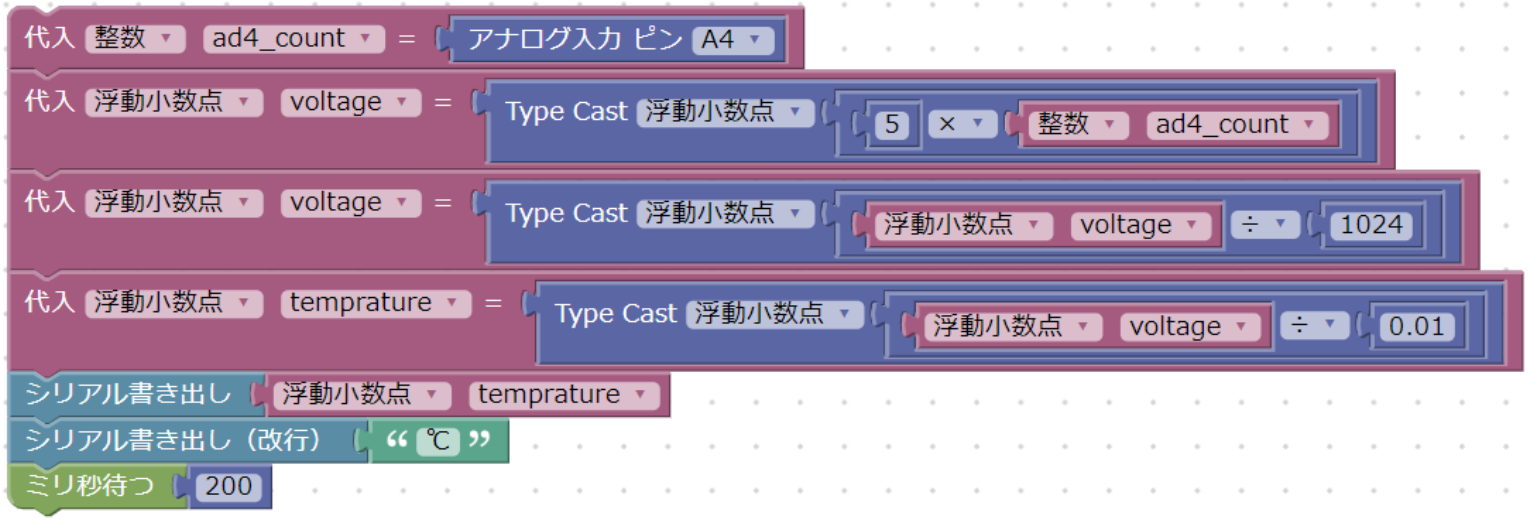
- プログラミングの基本概念の1つ
- 変数は数値や文字などの「**値**」を一時的に保管しておく「箱」であり，使用する際には「**型**」**の指定**と「**変数名**」**の指定**が必須である



よく使う「型」と「値」の事例

型名	用途	値の例	扱える値の下限	扱える値の上限
整数	整数の値を扱う	123	-32,768	32,767
ロング	より大きな整数の値を扱う	123456	-2,147,483,648	2,147,483,647
浮動小数点	小数点以下を含む値を扱う(勿論，整数も可)	123.45	$-3.4028235*10^{38}$	$3.4028235*10^{38}$
真偽	True/falseの二値を扱う	true	-	-
文字	1文字を扱う	‘a’	-	-
テキスト	複数文字を扱う	“abc”	-	-

例：温度の計測および算出

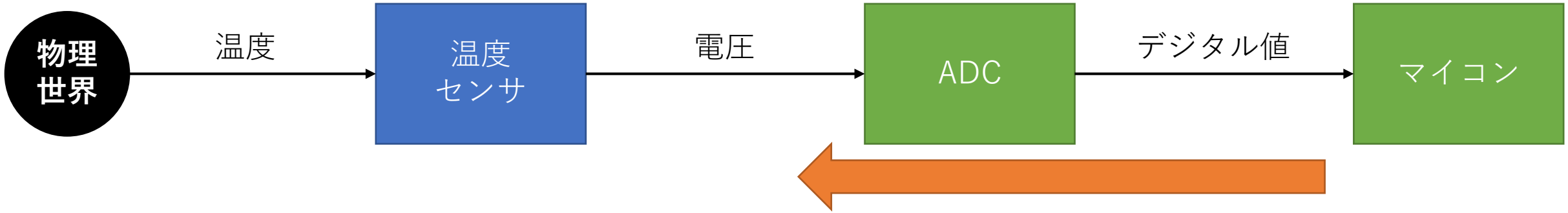


- ファイル名
  - adc_temp.xml
- 動作
  - AD4からの値を取得後，ADカウント値から電圧値を算出
    その後，電圧値から温度を算出し，シリアル通信で出力する

温度=(5v * ADカウント値) / 1024 / 0.01[V]

センサによる温度の算出方法①

デジタル→電圧



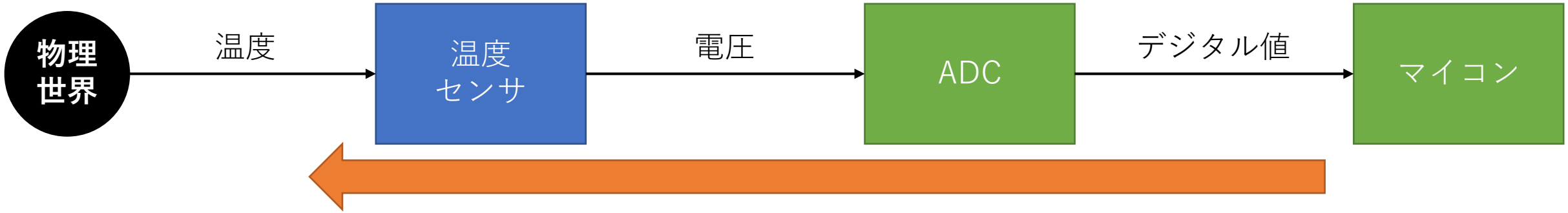
- マイコンが取得した「デジタル値」から「温度」を求める為には、まずは「電圧」を算出します
- デジタル値→電圧の変換は以下の式で求められます
  - ADCは0～5vの入力を、0～1023のデジタル値に変換
  - 入力電圧を**Vin**, 出力されたデジタル値を**d**とすると…

    Vin : 5v = d : 1024

    → 5v * d = Vin * 1024 → **Vin = 5v * d / 1024**

センサによる温度算出方法②

電圧→温度

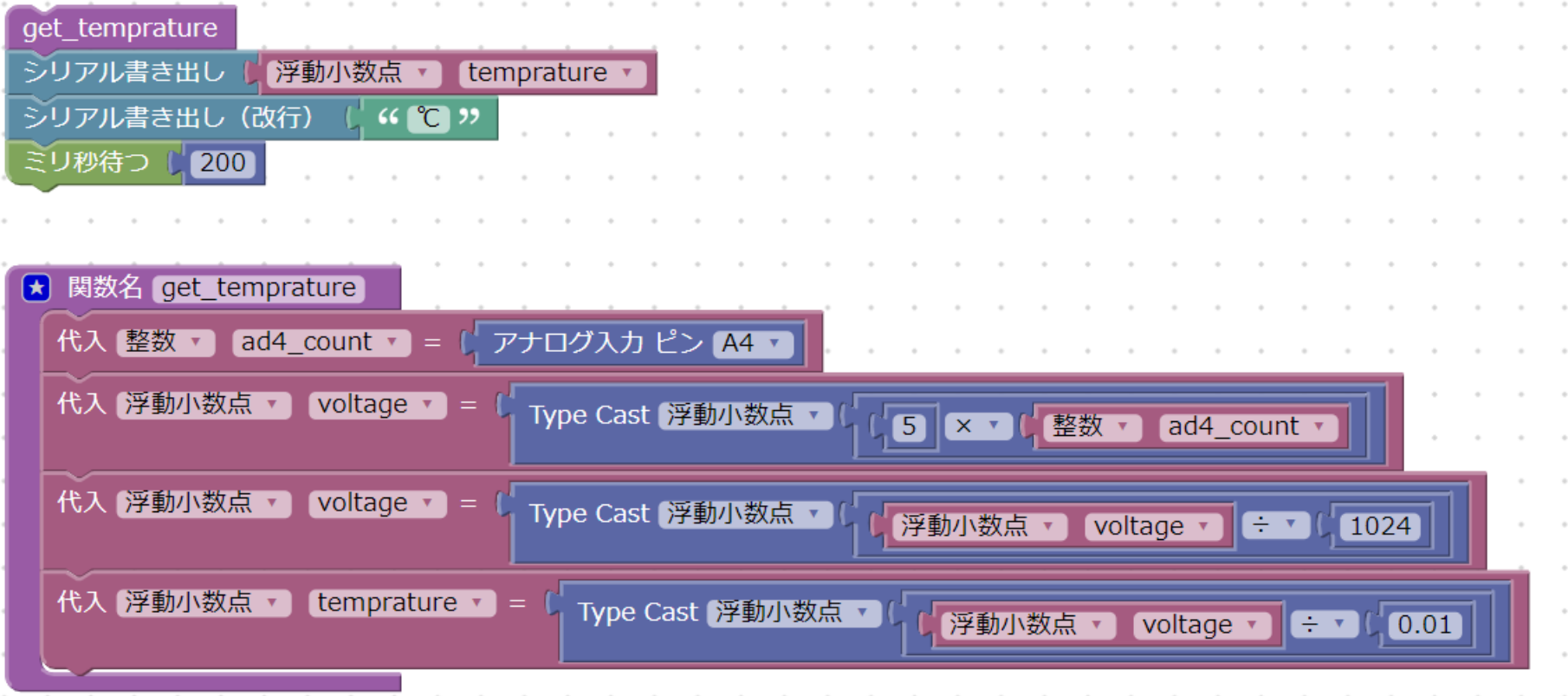


- 今回使用するLM35と呼ばれるセンサは2°C～+150°Cまでの温度範囲を検出できる温度センサIC
- 出力電圧は摂氏温度にリニアに比例(+100mV/°C)
- 温度と電圧の関係は以下の算出式から求められる
  - Vo = (+10mV/°C * T°C)　　(…温度からの電圧算出)
  - T = Vo / +10mV　　(…電圧からの温度算出)

デジタル値と物理量

- ラインセンサやボリュームなどは正確な物理量の算出は出来ない
  - ラインセンサ：光量からライン検知
  - ボリューム　：回転角度の検出
  - 変化量から何かを判断したい場合、この手のセンサは扱われる
- 温度センサや測距センサは、デジタル値→電圧→物理量まで算出が可能
- このように2種類あるので、設計する上で必要なセンサがどちらなのか捉えておくこと

温度取得の処理を関数化



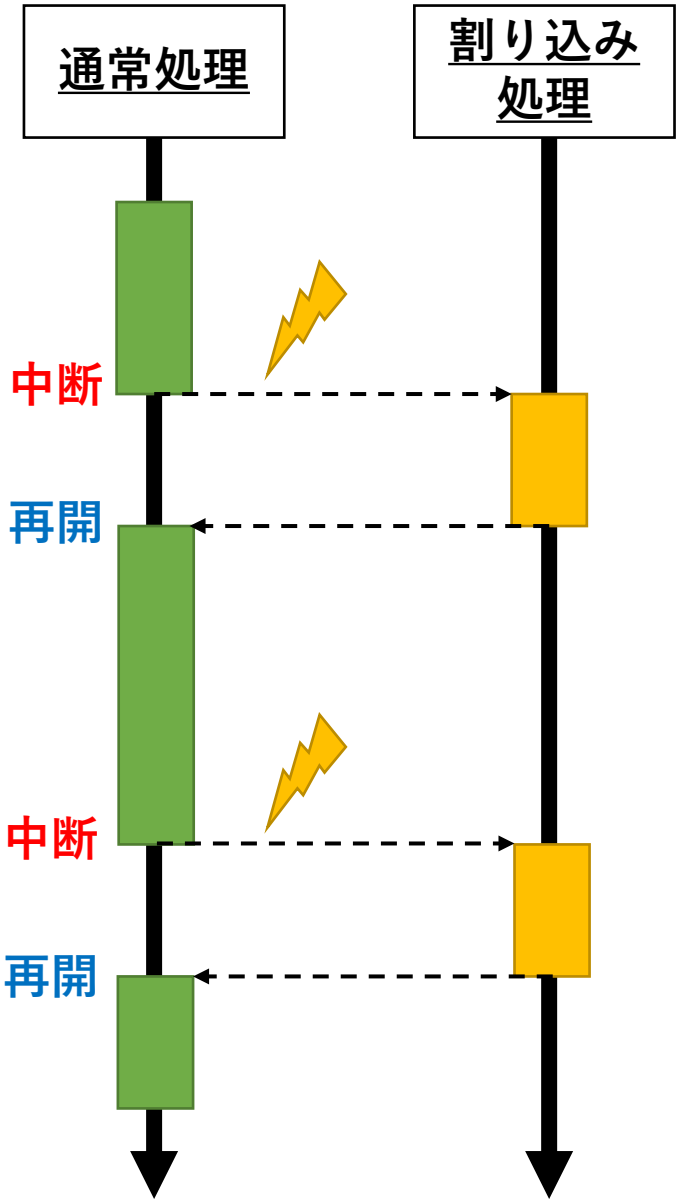
- GetTemprature関数実行後，temprature変数を参照すると中に温度データが格納されている

④ 割込み制御

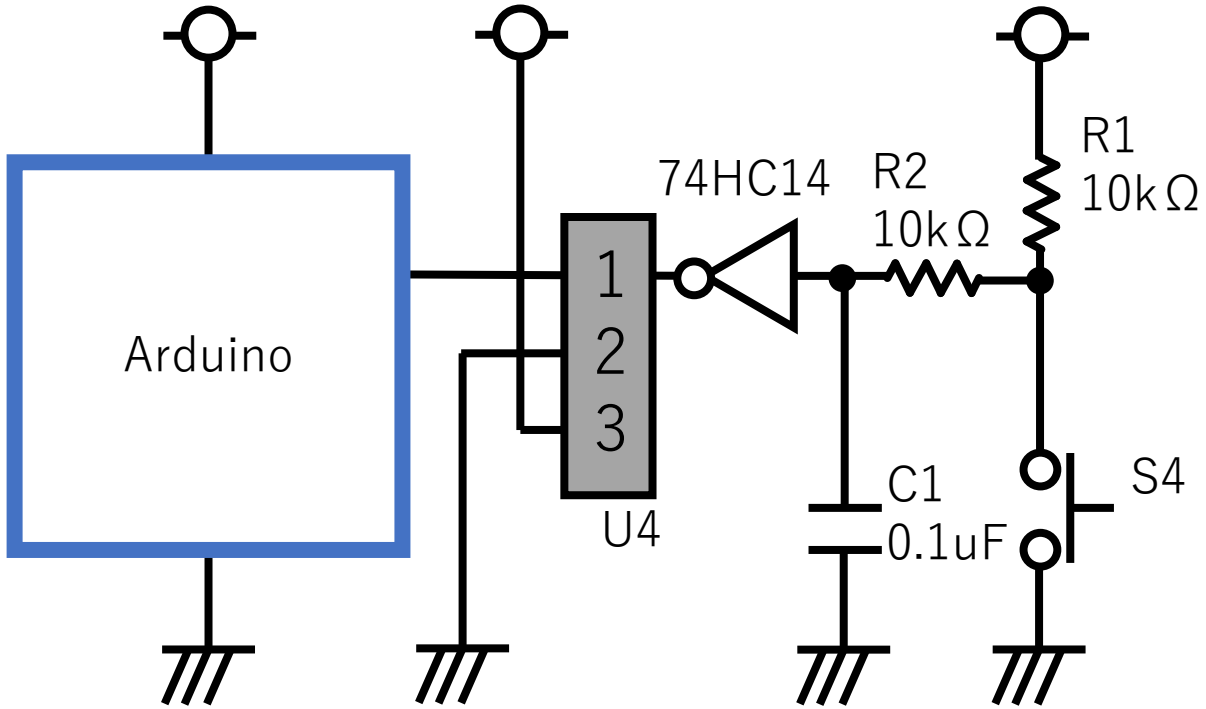
割り込みとは？

- 特定の入力等をきっかけに現在行っている処理を中断して，特定の処理を実行できる仕組み

- イメージ
  - 自動車の走行処理の実行中
  - 事故が発生し，急停止の信号をマイコンが受け取り
  - 自動車の走行処理を中断，マイコンからエアバッグの作動信号を出力
  - エアバッグが開く



ボタンの押下回数を取得する



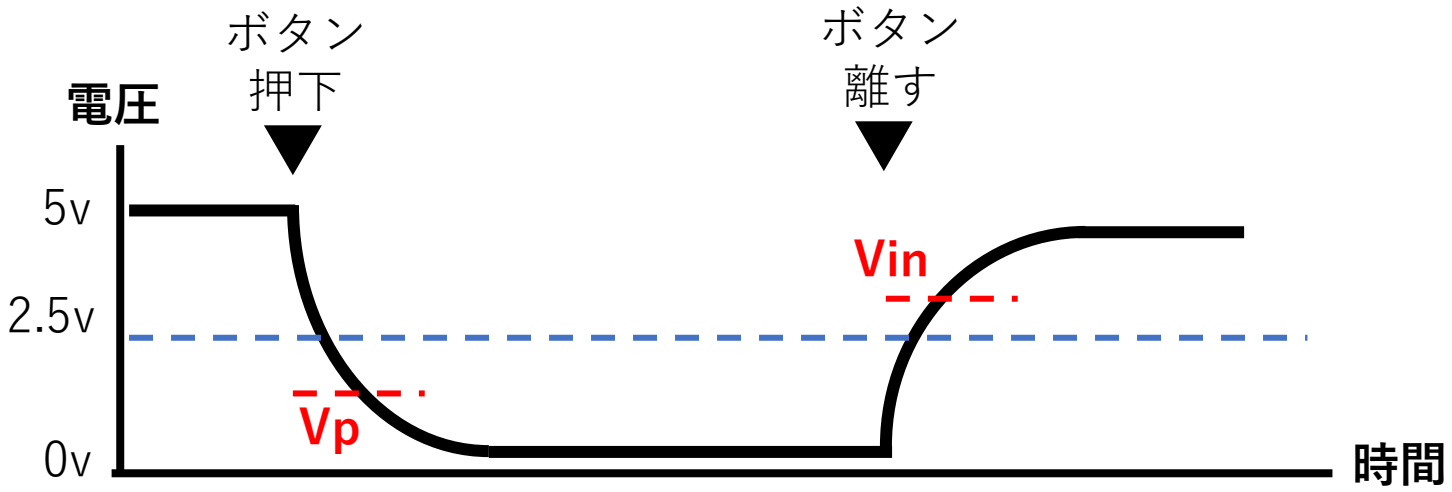
- チャタリングとは
  - 機械式スイッチ(トグルスイッチやボタンスイッチなど)を押す／離す時に発生する接点のバウンドのこと

- CRによるチャタリング防止回路
  - ボタンスイッチ向けの防止回路(左図)

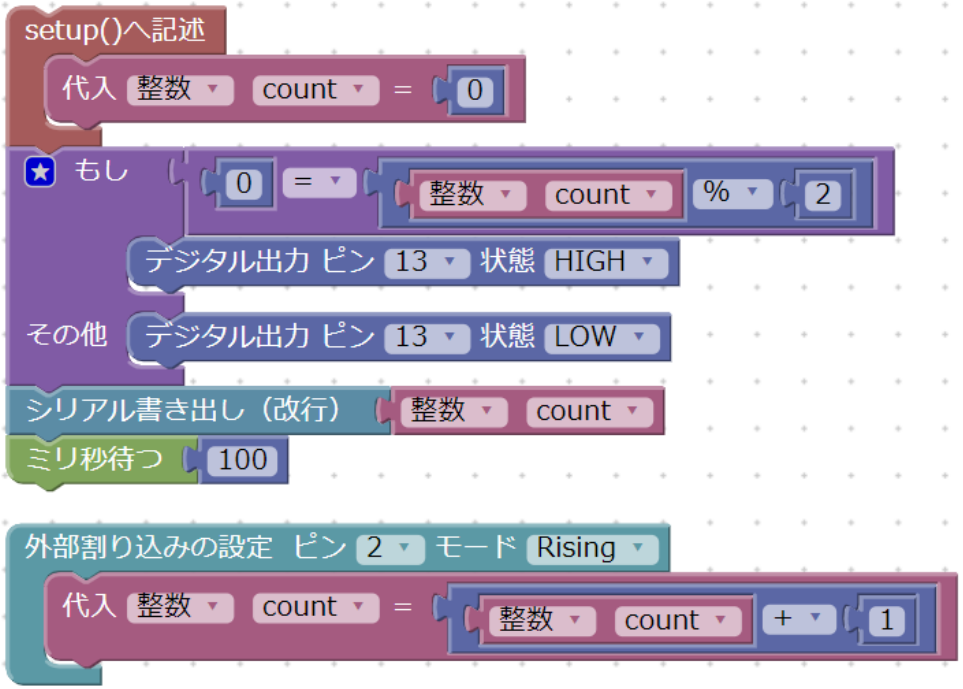
- Arduinoの2pinに割り当て
  - ボタン押下　　：＿＿＿＿＿
  - ボタン未押下　：＿＿＿＿＿

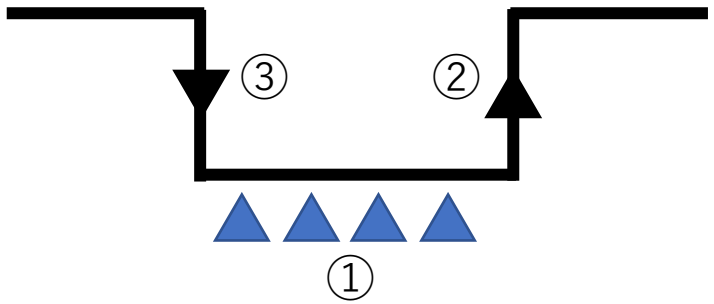
シュミットトリガ(74HC14)

- シュミットトリガ(74HC14)
  - NOTに似ているがNOTではない
  - 入力電圧の変化に対して出力状態がヒステリシスをもって変化する
- Vcc=4.5V Ta=25°Cの時は，Vp=2.7V，Vin=1.6Vで動作



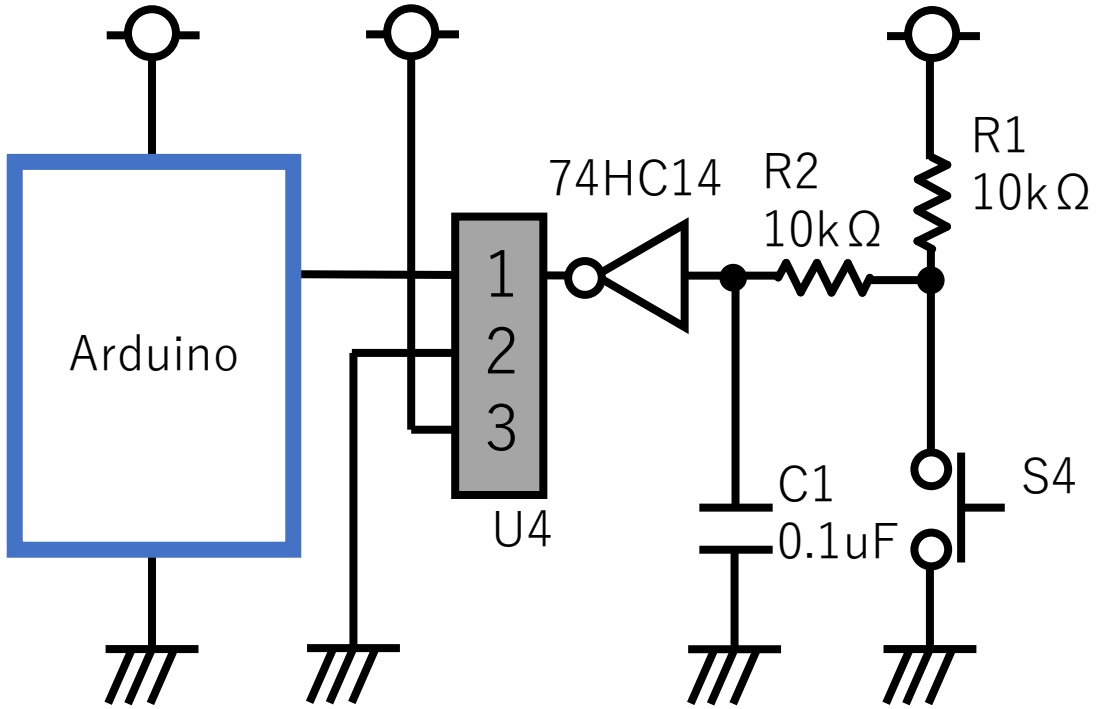
例：ボタンの押下回数の取得



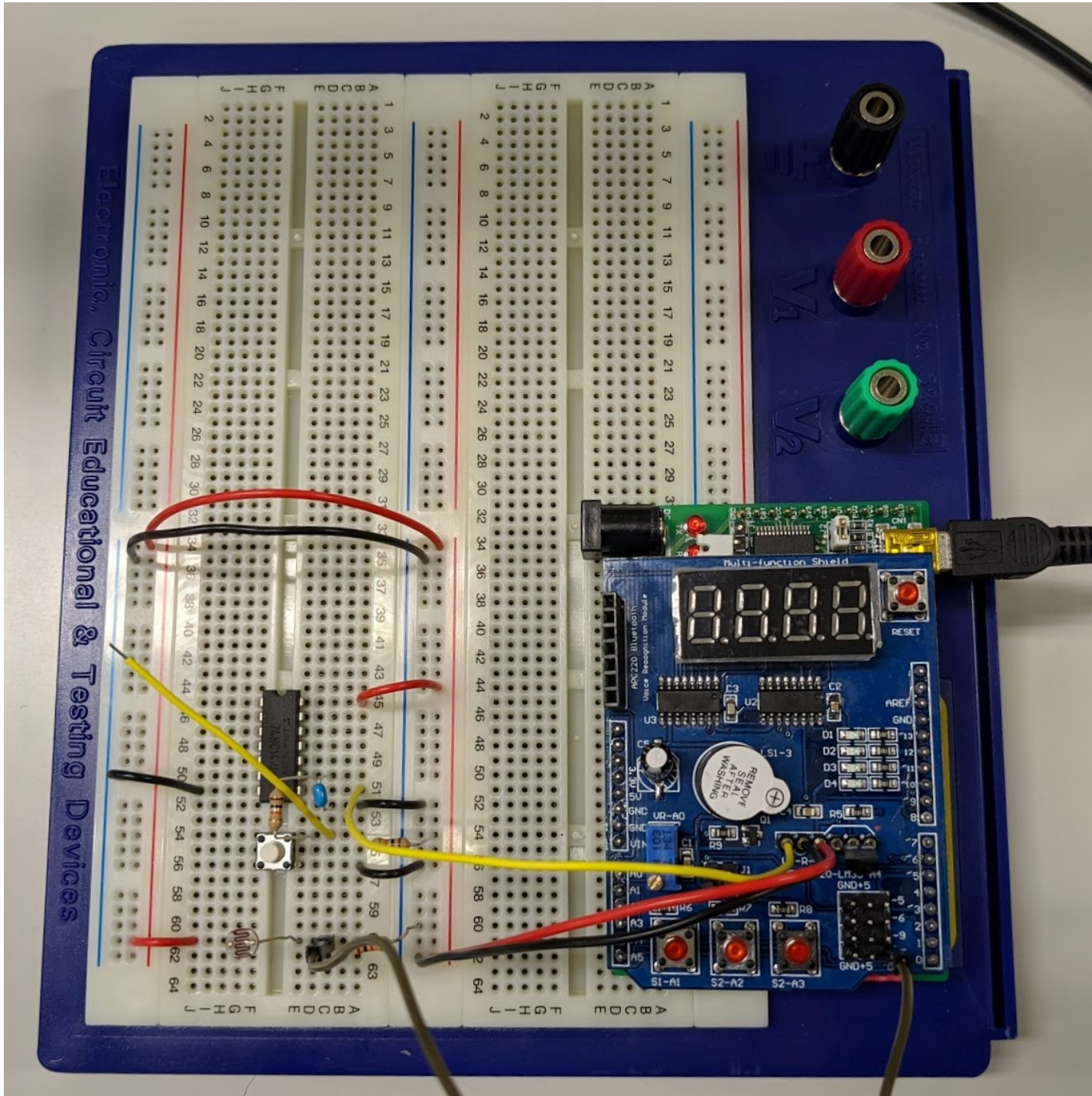


- ファイル名：interrupt.xml
- 動作
  - ボタン(S4)を押した回数をカウントし，シリアル通信で表示する
  - ボタン(S4) を1度押すとLED(D1)を点灯，もう一度押すと消灯する
    - いわゆる「オルタネイトスイッチ」
- 割り込みのモード
  - Low / Rising / Falling / Change
    ①　②　③　②，③

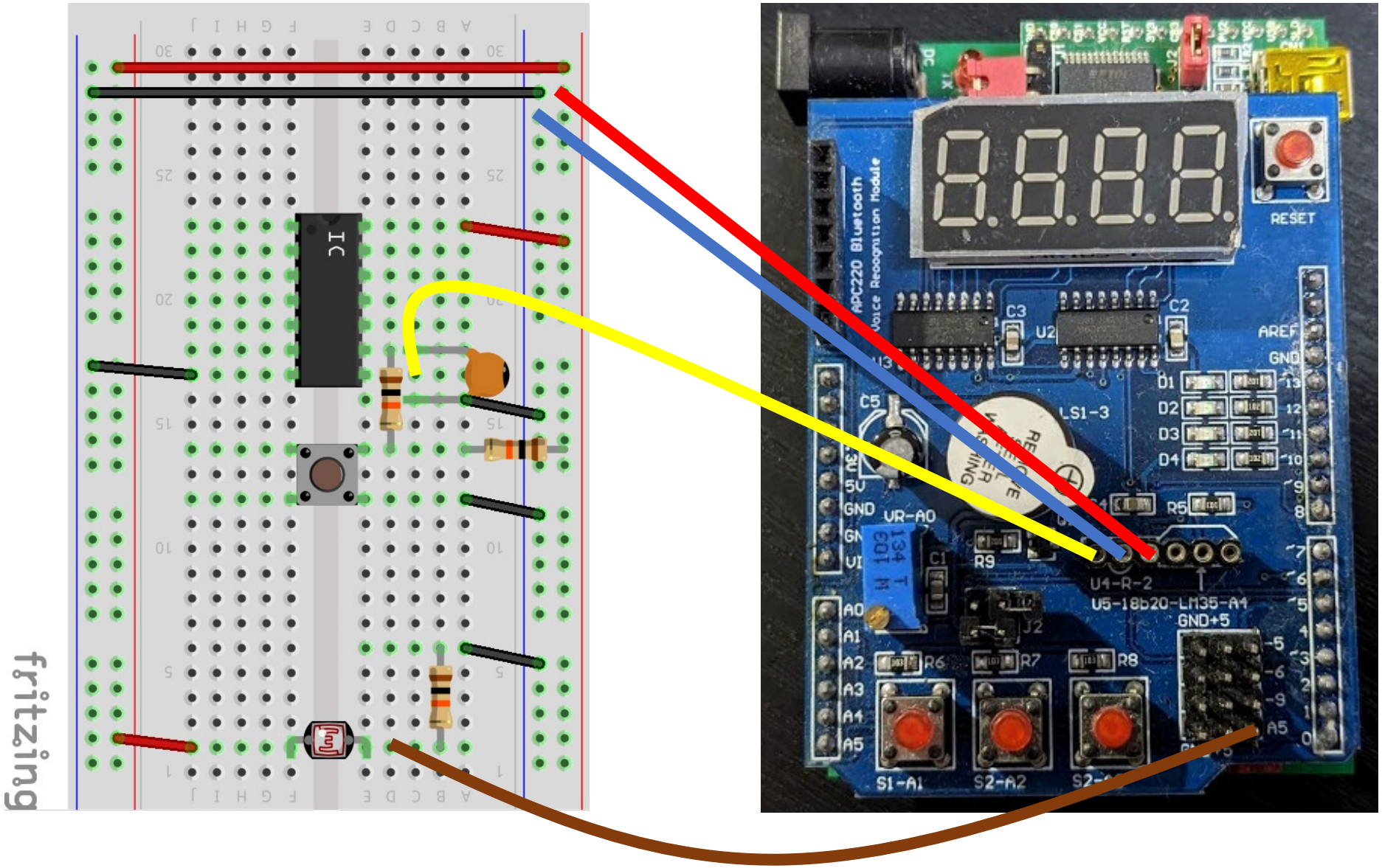
配線イメージ例



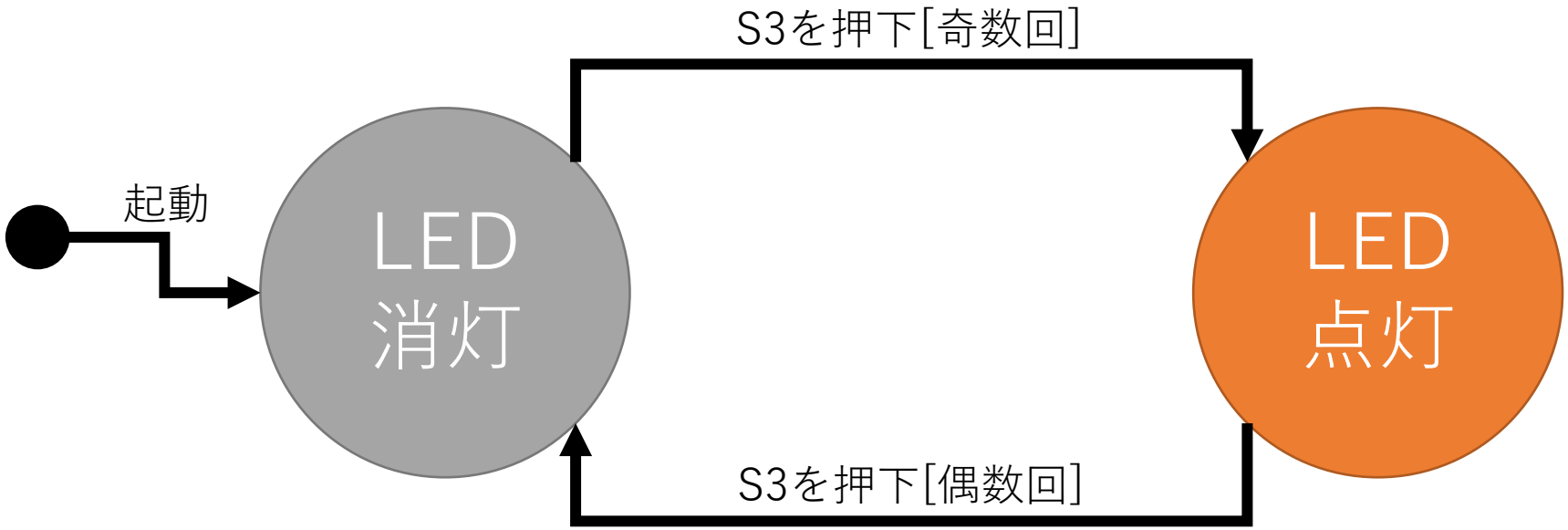
回路拡張：要ブレッドボード



実態配線図・例


fritzing
IC
APC220 Bluetooth
Voice Recognition Module
RESET
C3
U2
C2
AREF
GND
D1
D2
D3
D4
C5
LS1-3
5V
GND
VR-A0
R5
R9
U4-R-2
U5-18b20-LM35-A1
GND+5
A0
A1
A2
A3
A4
A5
R6
R7
R8
S1-A1
S2-A2

状態遷移図



- プログラムやシステムの状態を図示化して表し，プログラム全体の見通しを良くする為のもの
- 状態と状態を矢印で結び，何をキッカケで状態を遷移するのか？矢印に要因(トリガー)を記述する

練習：モード切替

ファイル名：interrupt_ren1.xml

1. ボタン(S4)を押すごとにモードを以下のように切り替えてプログラムの動作を変更してください
   - ０回目　：LEDの全消灯
   - １回目　：LEDの全点灯
   - ２回目　：LEDの交互点滅
   - ３回目　：LEDアニメーション(上から下)
   - ４回目　：LEDアニメーション(下から上)
   - ５回目　：　０回目に戻る

2. 上記のプログラムの状態遷移図を書いてみましょう

発展：モード切替②

ファイル名：interrupt_ren2.xml

1. ボタン(S4)を押すごとにモードを以下のように切り替えてプログラムの動作を変更してください
   - ０回目　：LEDの全消灯
   - １回目　：LED(D1)のみ点灯　＋　ボリューム値のシリアル出力
   - ２回目　：LED(D2)のみ点灯　＋　光センサ値のシリアル出力
   - ３回目　：LED(D3)のみ点灯　＋　温度のシリアル出力
   - ４回目　：　０回目に戻る

2. 上記のプログラムの状態遷移図を書いてみましょう

オマケ

ファイル名：interrupt_ren3.xml

1. ボタン(S4)を押した回数をLED(D1～D4)に２進数で表示しましょう
2. １に追加．ボタン(S4)を押したら，「ピッ」と音が鳴るようにしてみましょう

【ヒント】

LEDが4つであるので，0～15までを表示が可能
仮に「11」を表示しようとした場合…

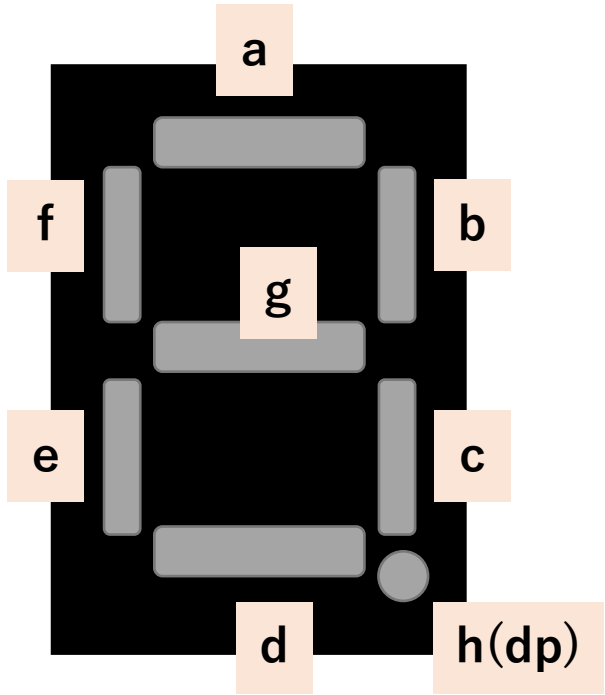
11/8の結果が0? 1? → 1ならLED(D4)を点灯
11%8の結果を変数aに格納
a/4の結果が0? 1? → 1ならLED(D3)を点灯
a%4の結果を変数aに格納
a/2の結果が0? 1? → 1ならLED(D2)を点灯
a%2の結果が0? 1? → 1ならLED(D1)を点灯

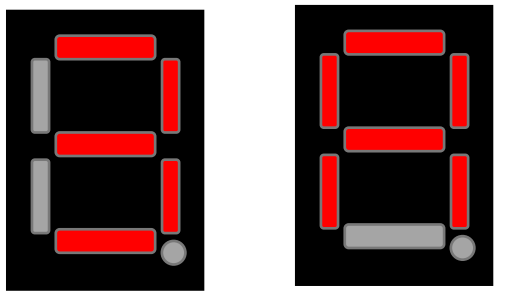
⑤ 7セグメントLED
(スタティック点灯)

7セグメントLEDは

- LEDを7個or8個組み込んだ電子部品
  - 各セグメントにLEDが1つ組み込まれる
  - LEDを光らせる位置を工夫することで数字を表現する
    - 例）a,b,c,d,gセグメントを点灯すると’3’を表現できる
    - 工夫すればアルファベットも可
      - 7セグメントLED+フォントで検索！！
  - 多くの場合，小数点のセグメントを含み8個のLEDが内蔵される

- 今回は4桁の7セグメントLEDを使用する

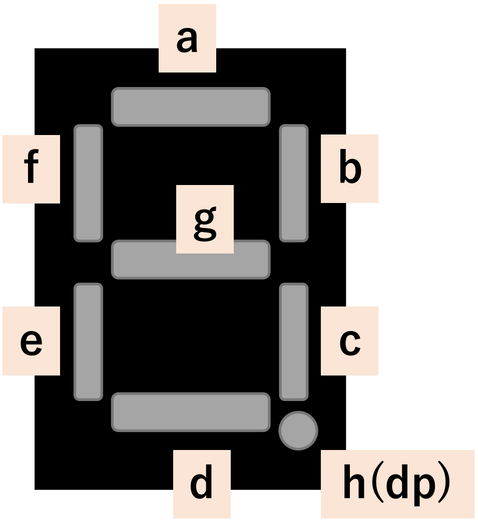


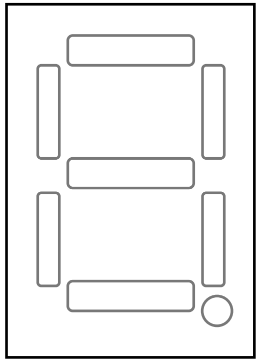


‘3’の例

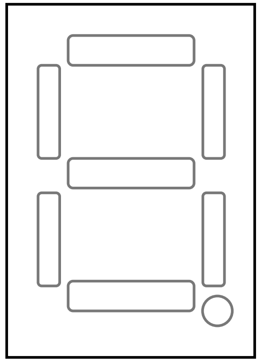
‘A’の例

0～9を表示する為には？

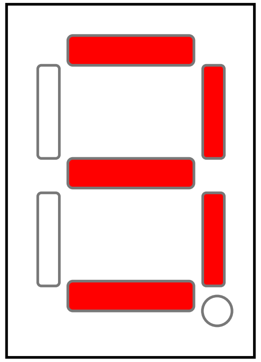




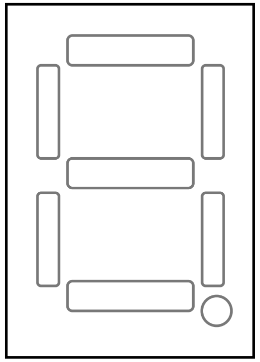
a, b, c, d,
e, f, g, h



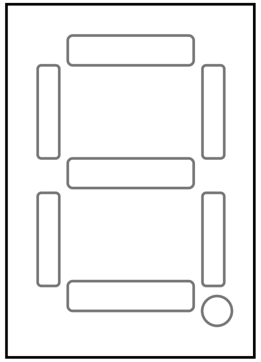
a, b, c, d,
e, f, g, h



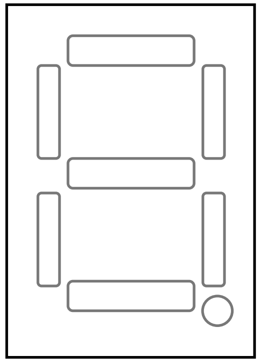
a, b, c, d,
e, f, g, h



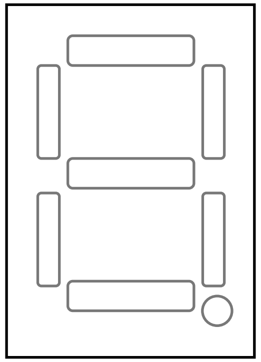
a, b, c, d,
e, f, g, h



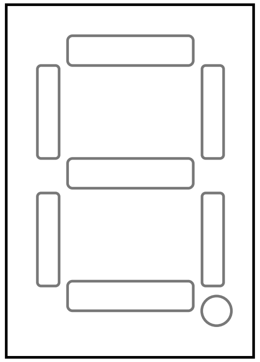
a, b, c, d,
e, f, g, h



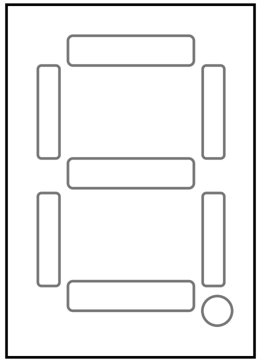
a, b, c, d,
e, f, g, h



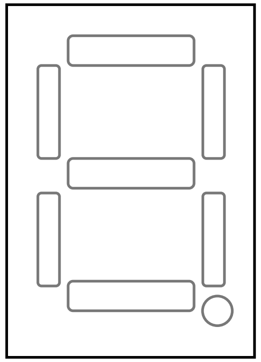
a, b, c, d,
e, f, g, h



a, b, c, d,
e, f, g, h

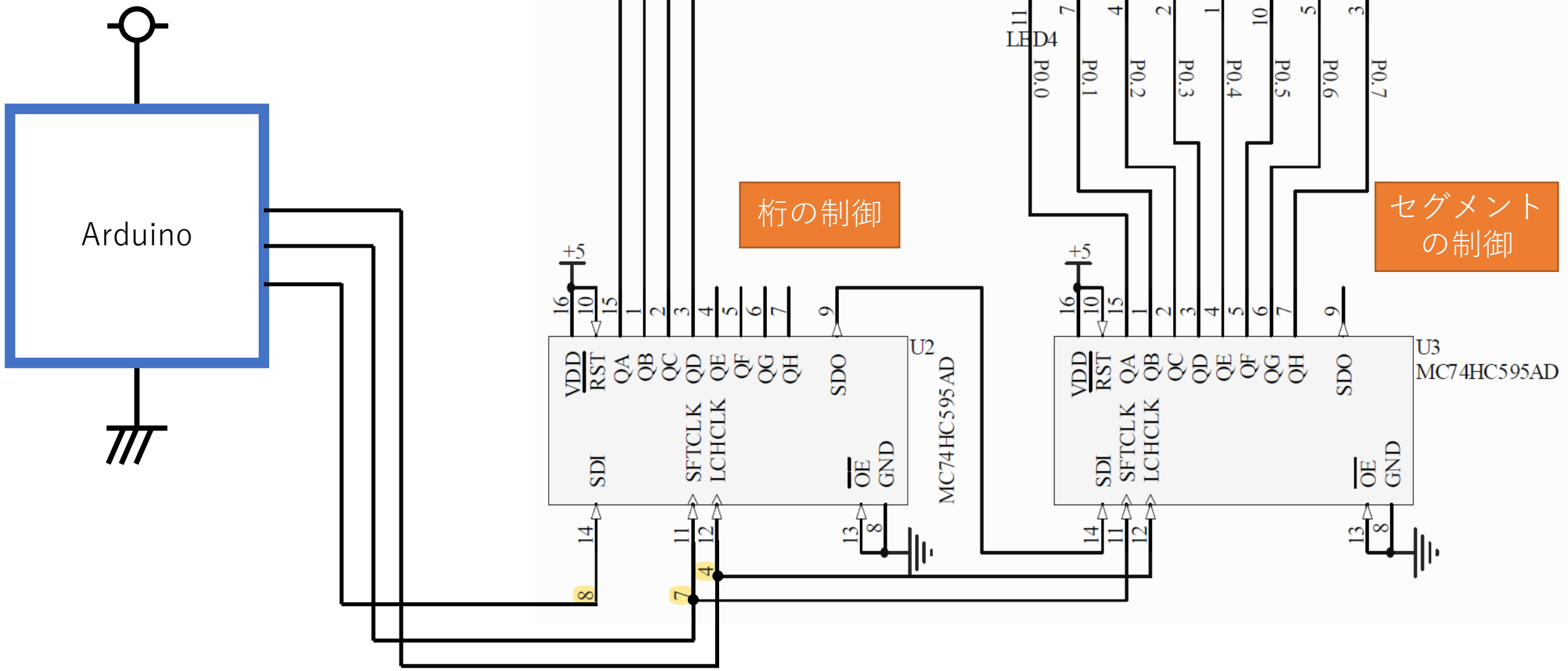


a, b, c, d,
e, f, g, h



a, b, c, d,
e, f, g, h

7セグメントLEDを制御する



16bitシフトレジスタの割当

- 8bit + 8bitシフトレジスタの各ビットは以下のような割当となっている

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
桁の制御 (制御対象には1)				未使用				セグメントの制御 (制御対象には0)							
4桁目	3桁目	2桁目	1桁目	-	-	-	-	a	b	c	d	e	f	g	h (DP)

【例】「1桁目」に「3」を表示したい場合

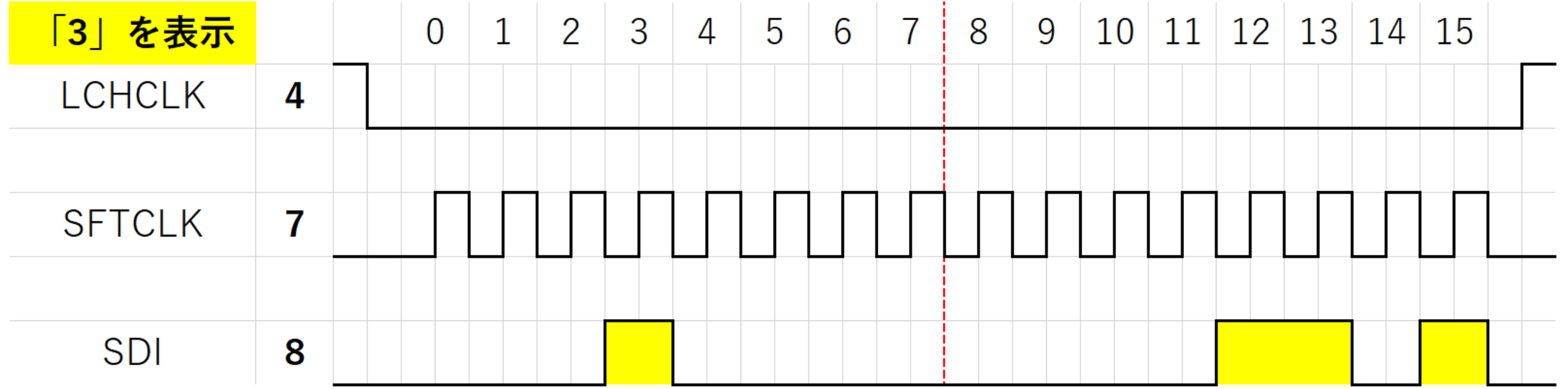
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4桁目	3桁目	2桁目	1桁目	-	-	-	-	a	b	c	d	e	f	g	h (DP)
0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1

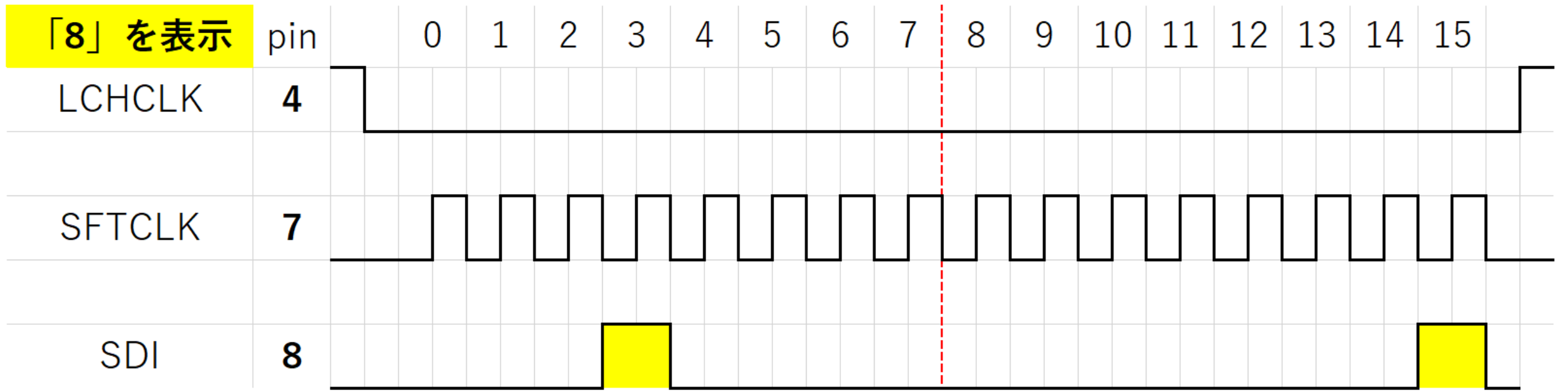
シフトレジスタに格納する値を考える

制御対象		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
桁	セグ	桁の制御 (制御対象には1)				未使用				セグメントの制御 (制御対象には0)							
		4桁目	3桁目	2桁目	1桁目	-	-	-	-	a	b	c	d	e	f	g	h (DP)
1	3	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1
1	8																
1	0																
1	1																
1	5																
2	0																
3	1																
4	5																

メモ用

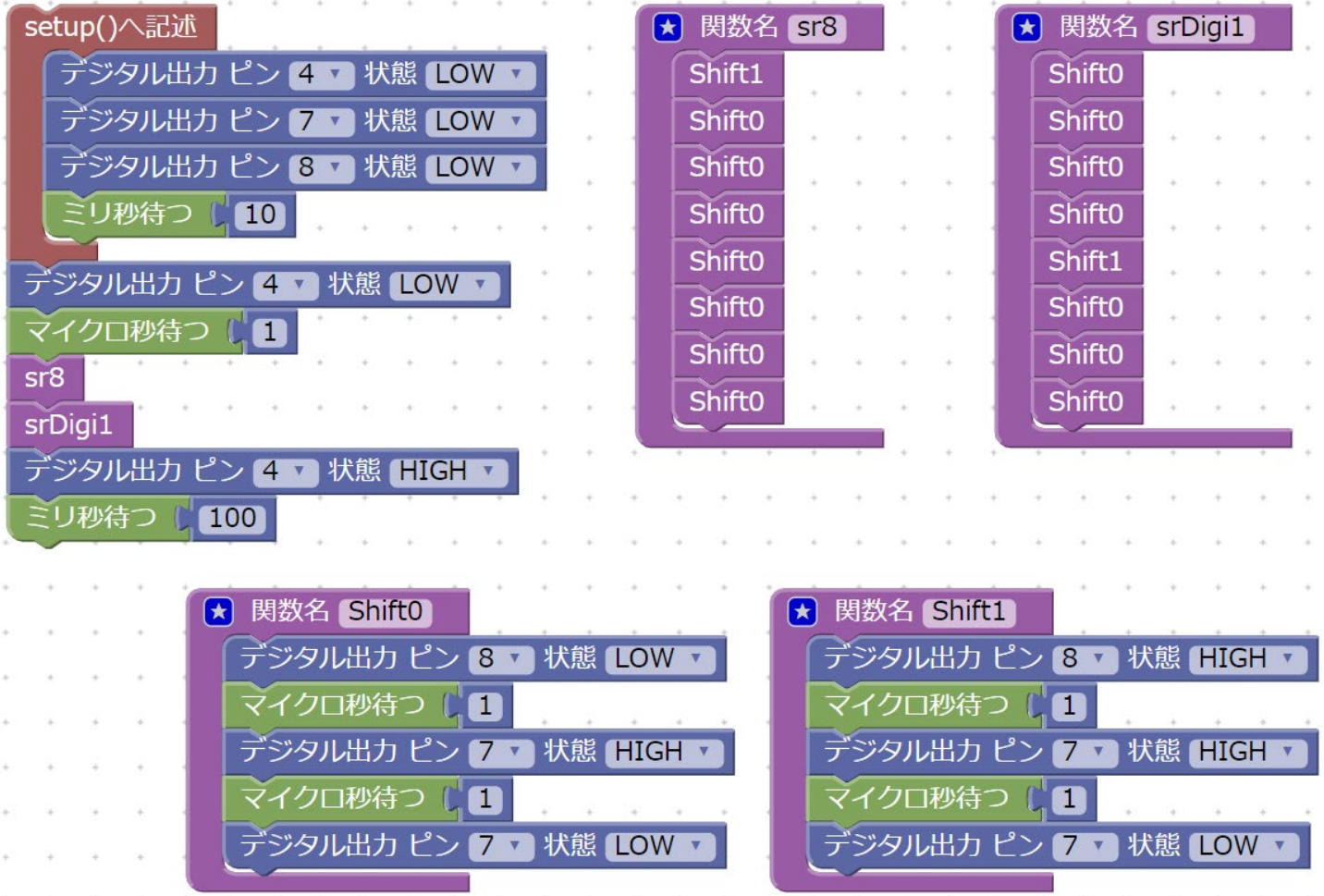
制御対象		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
桁	セグ	桁の制御 (制御対象には1)				未使用				セグメントの制御 (制御対象には0)							
		4桁目	3桁目	2桁目	1桁目	-	-	-	-	a	b	c	d	e	f	g	h (DP)
1	3	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1


「3」を表示
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
LCHCLK 4
SFTCLK 7
SDI 8


「8」を表示
pin
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
LCHCLK 4
SFTCLK 7
SDI 8

ソースコード配布

例１：7セグLEDに「8」を表示する



- ファイル名：_7seg.xml
- 動作
  - 7セグLEDの1桁目に8を表示する
- プログラム解説
  - Shift0, Shift1ブロック
    - シフトレジスタへ0を送る，1を送るブロック
  - sr8ブロック
    - 表示する数字(8)を指定する
  - srDigi1ブロック
    - 表示する桁数を指定する

練習

ファイル名：_7seg_ren1.xml

1. 1桁目に0を表示する
2. 1桁目に1を表示する
3. 1桁目に5を表示する
4. 2桁目に0を表示する
5. 3桁目に1を表示する
6. 4桁目に5を表示する

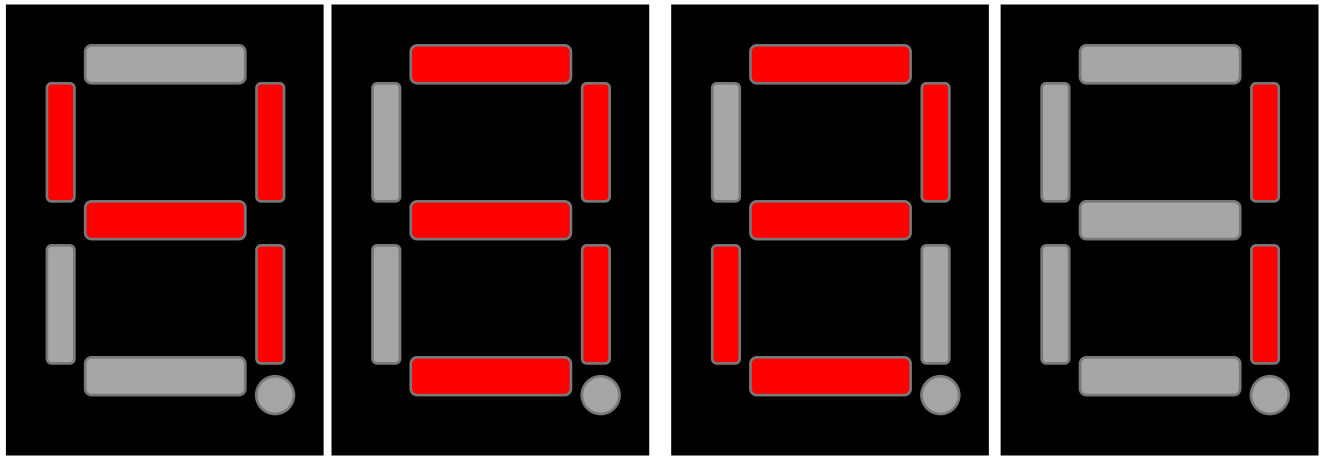
発展練習

ファイル名：_7seg_ren2.xml

1. 1,2桁目に8を表示する
2. 1,3桁目に0を表示する
3. 1～4桁目に1を表示する
4. 1～4桁目をブランク(全消灯)にする
5. 7セグLEDに対して，0から9までをカウントアップして表示する(1秒周期)
6. オマケ：文字を表示するシリーズ
    1. 1桁目にAを表示する
    2. 1桁目にEを表示する
    3. 1桁目にHを表示する

⑦ 7セグメントLED
(ダイナミック点灯)

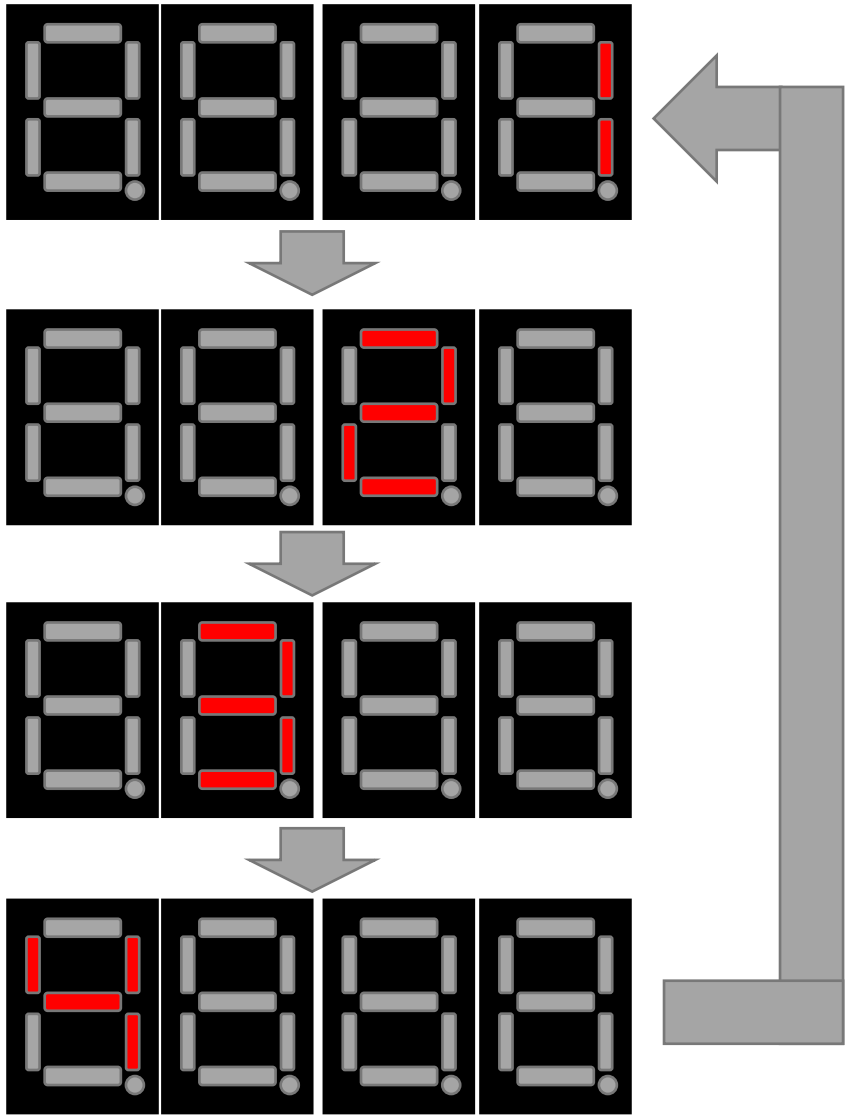
4桁にそれぞれ別の数字を表示したい



現状の回路ではすべて同じ数字が表示されてしまう

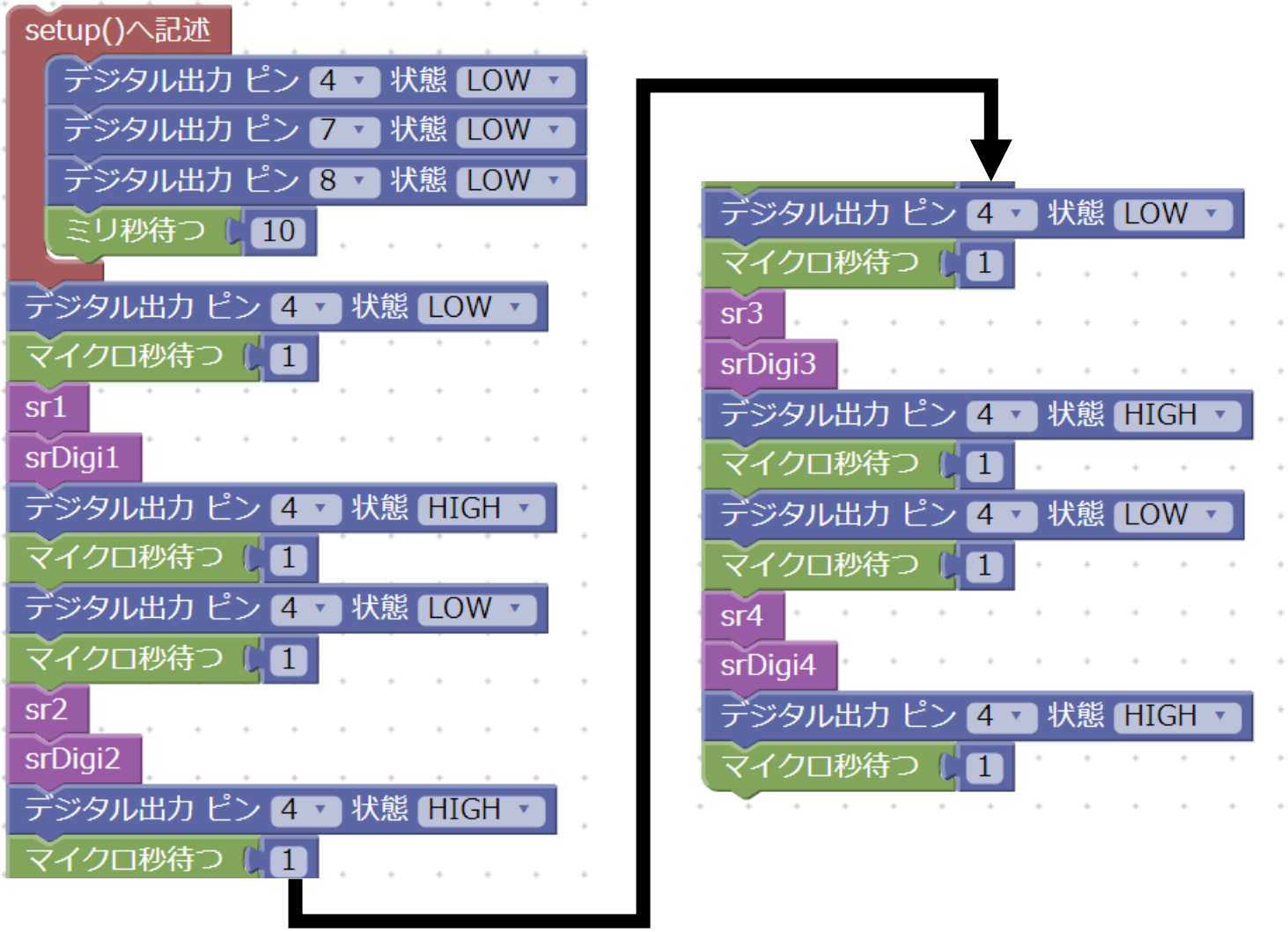
各桁を1つずつ切り替えながら制御すれば可能

ダイナミック点灯



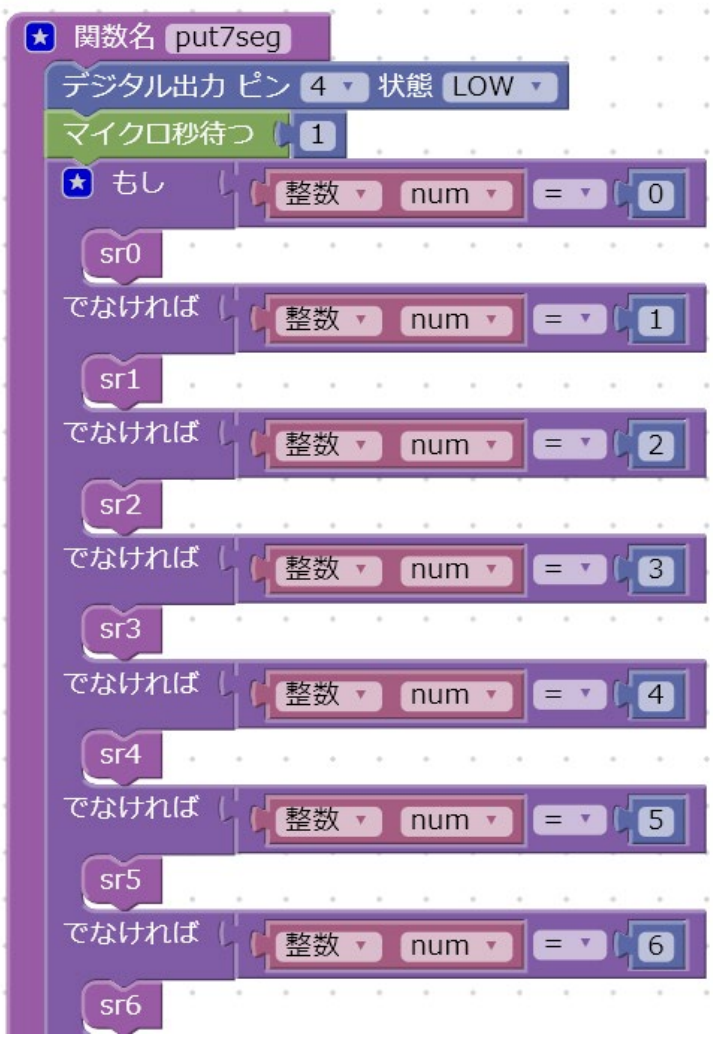
- 1～4桁目を1つずつ切り替えながら，それぞれの値を表示する
- これを，高速に繰り返す事であたかも常時点灯している状態のように見える
  - いわゆる残像効果
  - ただし，LEDの輝度は若干下がる

例１：7セグLEDに「4321」を表示する



- ファイル名：_7segD.xml
- 動作
  - 7セグLEDに対してダイナミック点灯で4321を表示する

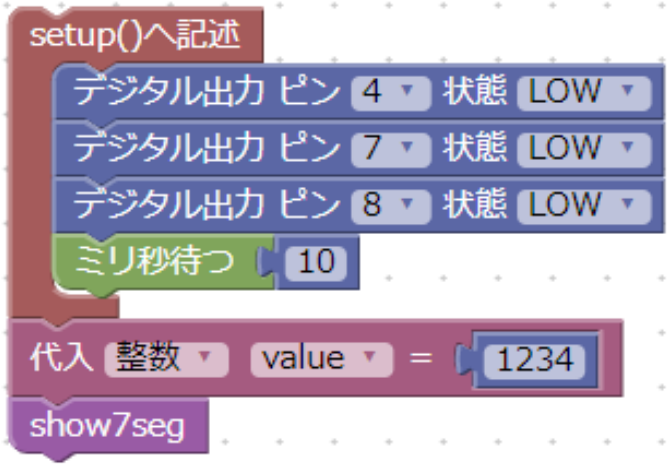
関数化して再利用性を高める

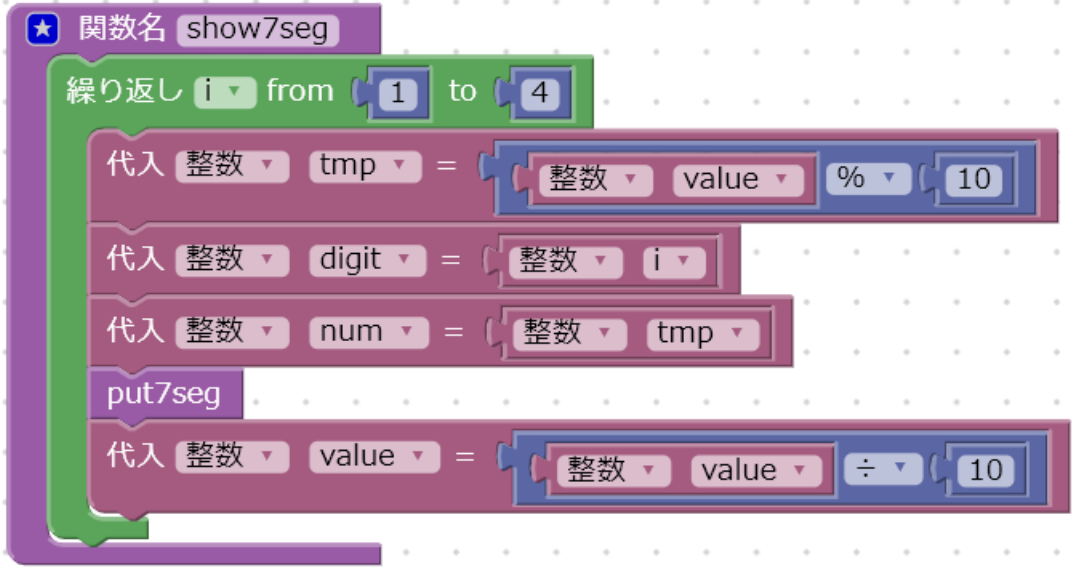


- put7seg関数
  - 指定の桁数に対して，任意の数字を表示できる
  - 入力の変数
    - digit … 桁
    - num … 数字

ソースコード配布

例２：関数を活用して表示





- ファイル名：_7segD2.xml
- show7seg関数
  - 任意の整数値(四桁まで)を7セグLEDに表示できる関数
  - 入力の変数
    - Value　　… 7セグに表示する値
- メインの処理
  - 7セグLEDに1234を表示する

練習

ファイル名：_7segD_ren1.xml

1. 光センサの値を7セグLEDに表示する
2. ボタン(S4)を押した回数を7セグLEDに表示する

発展練習

ファイル名：_7segD_ren2.xml

1. (概ね)15秒カウンタの作成
   - 0000からスタートして，0014までカウントアップして表示する
   - 0015に達したら0000に戻る
2. (概ね)60秒カウンタの作成
3. ストップウォッチの作成
   - 時間は1秒単位で7セグLEDに表示する
   - ボタン(S1)を押下中のみ時間計測する
   - ボタン(S1)を離すと，計測結果が残ること
   - ボタン(S3)を押すと，計測結果が0クリアされる
4. ストップウォッチの作成②
   - 先ほど作成したストップウォッチの機能のうち，ボタン(S1, S3)を使用せずボタン(S4)のみで実現する
   - 仕様から自分で考えましょう
   - 状態遷移図も書いてみましょう

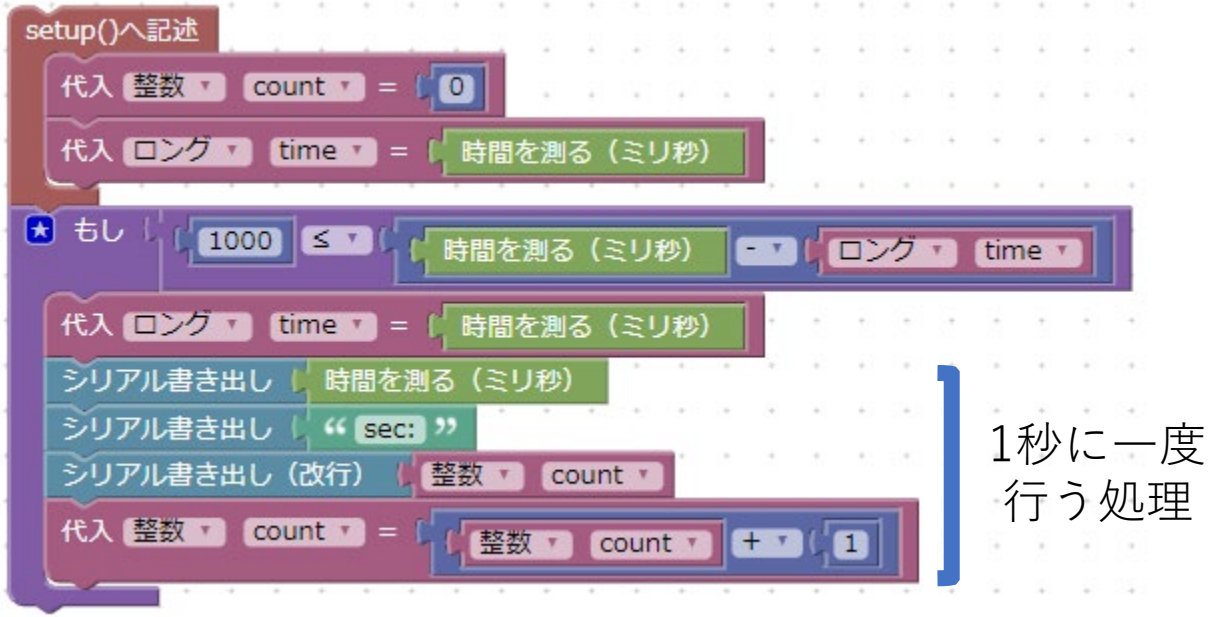
ソースコード配布

ヒント：時間を測るブロックの活用

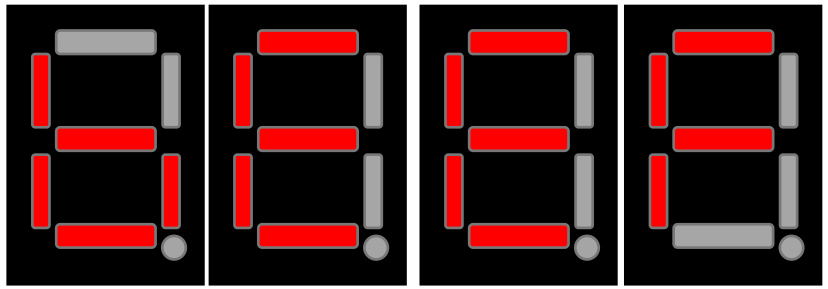
- ダイナミック点灯を行う場合，プログラム内で「待ちブロック」を使用すると表示が乱れてしまう
- 「時間を測るブロック」を活用する事で，経過時間を計測することができる
  - time.xml

例①：まずはコレから試してみよう！

例②：「1秒」おきに特定の処理をする事例



発展練習②

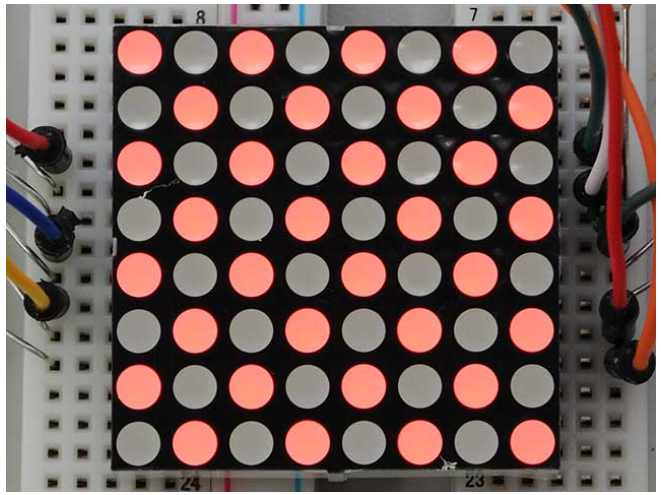


- 文字を表示してみようシリーズ
  1. 7セグLEDに，”bEEF”と表示する
  2. 7セグLEDに，”CAFE”と表示する

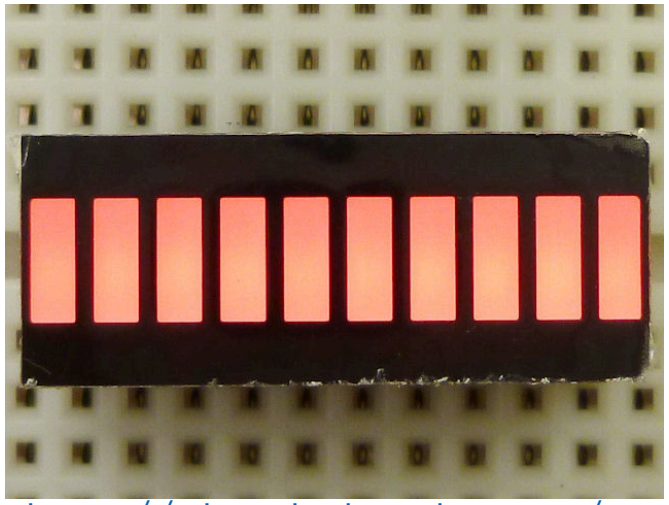
- ヒント：「7セグ　フォント」で検索してみましょう！

[Appendix]ダイナミック点灯の活用先

- 7SEG LED以外にもマトリックスLEDやバー型のLEDでも同様にダイナミック点灯による制御が可能です



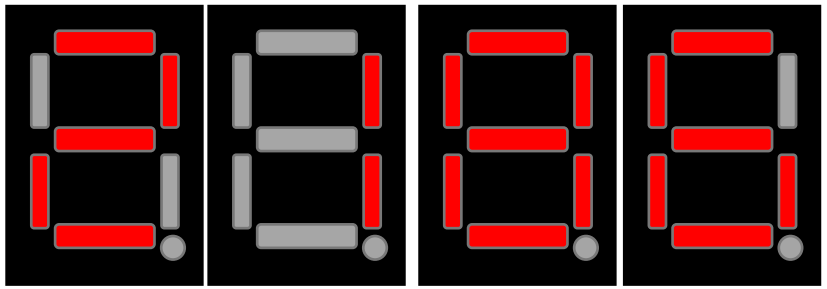
http://akizukidenshi.com/catalog/g/gI-05163/



http://akizukidenshi.com/catalog/g/gI-04290/

⑧温度計

仕様

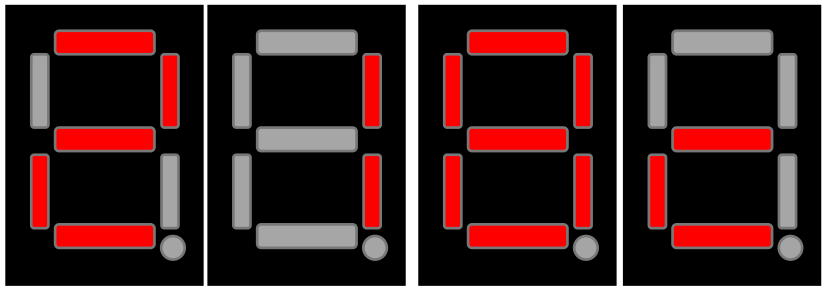


- 上記の図のようなイメージで温度を表示する
  - 例：21.86°Cの場合の表示例
  - 小数点(DP)は表示しない

- 追加仕様
  - 30.0°Cを超えたらアラート表示としてLED(D1)を点灯させる
  - 35.0°Cを超えたらアラート表示としてLEDを全点灯＋ブザーONする

- 追加仕様②
  - ボタン(S1)を押下している間は，温度に変わって明るさを表示する

発展仕様



- 追加仕様③
  - 温度表示を次のように変更する
    - 1桁目には単位(℃)として，「c」を表示する
    - 2～4桁目に，小数点以下第1位までの温度を表示する
    - 今回，小数点(DP)の表示は考えない
  - 上記の図は，21.8℃を表示した時の例

- 追加仕様④
  - ボタン(S4)を押下した回数でモードを変更する
    - 0回目：温度表示モード　＋　LED(D1)のみ点灯
    - 1回目：明るさ表示モード＋　LED(D2)のみ点灯
    - 2回目：カウントアップ　＋　LED(D3)のみ点灯

まとめ

今回，学んだこと

- マイコンと回路のインターフェース
  - LED制御
  - スイッチ制御
  - センサ制御(ADコンバータ)
  - 7セグメントLED制御

- プログラミングの基礎の基礎
  - 逐次・分岐・繰り返し
  - 値の種類と取り扱い
  - 変数
  - 関数

の概念

この先にあるもの

- ハードウェア主体の学習はココまでで一区切り

- この先からは「プログラミング」が主体
  - [2ヶ月目] C言語プログラミング
  - [3ヶ月目] マイコンプログラミング

引用元・参照元

- 参考元・引用元
  - Multi-function Shield関連
    - Using an Arduino Multi-function Shield
      - https://www.cohesivecomputing.co.uk/hackatronics/arduino-multi-function-shield/
    - 回路図
      - http://aitendo3.sakura.ne.jp/aitendo_data/product_img/binbo/shield/GK-SLD1.sch.pdf
  - マルツ電波>スイッチのチャタリングの概要
    - https://www.marutsu.co.jp/pc/static/large_order/1405_311_ph

画像利用・謝辞

- 画像利用
  - いらすとや
    - https://www.irasutoya.com/
  - 秋月電子
    - http://akizukidenshi.com/catalog/default.aspx

- 謝辞
  - BlocklyDuinoを日本語化され公開された岡田 裕行氏に感謝します
    - https://github.com/makewitharduino/Online-BlocklyDuinoEditor