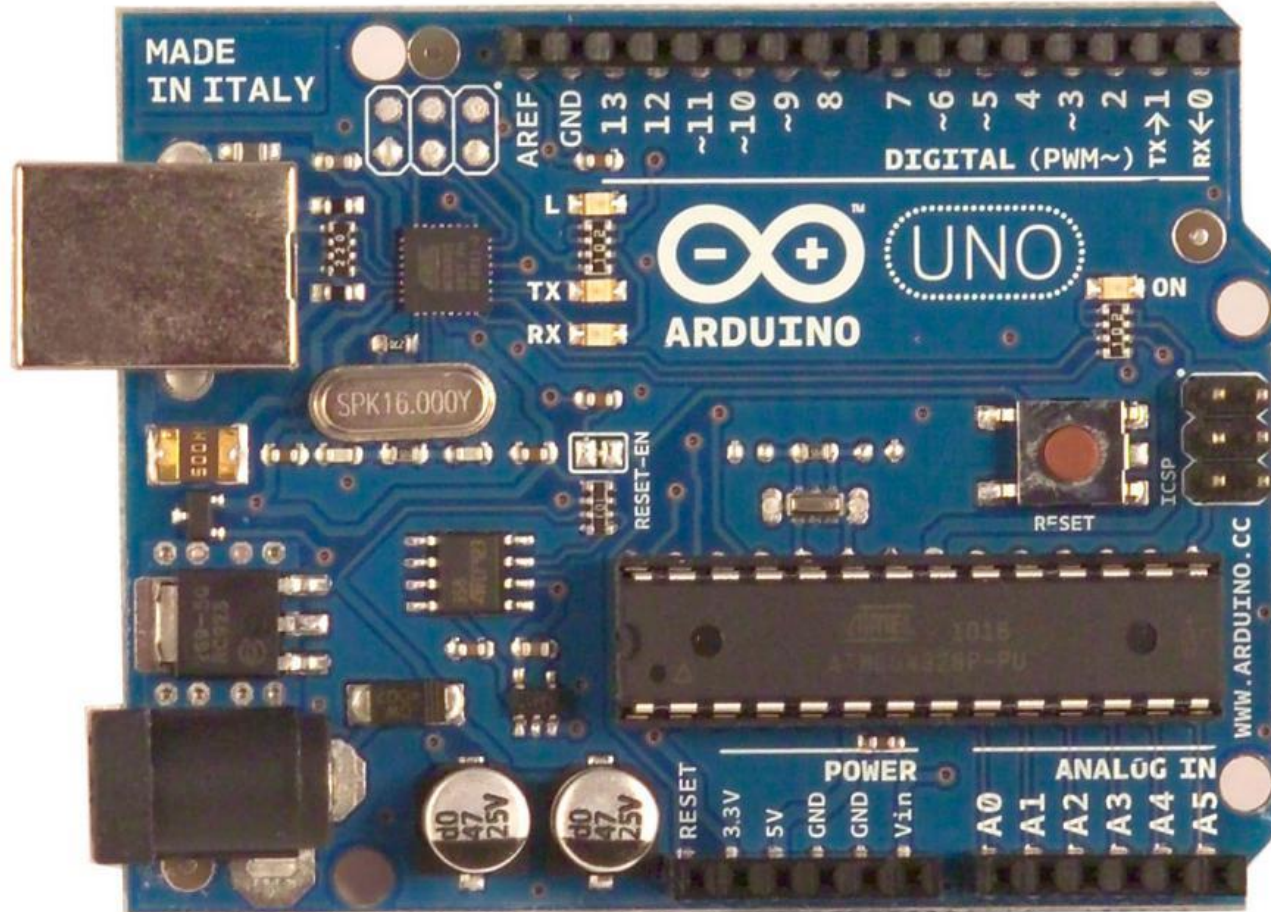


Arduinoの I / Oポート

- ・14本(+6本)のデジタル入出力ポート
- ・6本のアナログ入力ポート



It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.

Arduinoシミュレータ

各自のデスクトップのプログラムフォルダーにArduinoシミュレータUnoArduSimがあるので、ダブルクリックして実行する。また、UnoArduSimは
<https://www.sites.google.com/site/unoardusim/services>
で配布されている。(インストールせずに実行ファイルのみで実行可能)

使用上の参考サイト→arduino日本語リファレンス
<http://www.musashinodenpa.com/arduino/ref/>

また、文法はC++に準拠しているので、必要に応じて、調べること

C++とPythonの比較

	C++, C	Python
言語の特徴	コンパイル言語	インタプリタ言語
長所	<ul style="list-style-type: none">• 実行速度が速い• 機械語の実行形式になっているので、小さなコンピュータで実行可能	<ul style="list-style-type: none">• コンパイルの必要が無いため、待たずに実行できる• 多少のエラーは融通がきく• 文字列の操作が得意
短所	<ul style="list-style-type: none">• 規模によってはコンパイルに時間がかかる• エラーがあるとコンパイルできない• 文字列の操作が比較的苦手	<ul style="list-style-type: none">• C++に比べて遅い• 実行するコンピュータにはPythonインタプリタを実装している必要があるので、小さなコンピュータでは実行できない

マイクロコンピュータの記述言語としてはCまたはC++が使われることが多い

C++とPythonの文法構造の比較

C++

```
int x,y,z;
if (x > 500)
{
    y=x+100;
    Serial.println(y);
}
z=100;
```

- 変数の型宣言が必要
- if文の論理式は丸括弧の中に入れる
- 論理式がtrueの時に実行される実行単位は波括弧の中に入れる（インデントは動作とは無関係）

Python

```
if x > 500:
    y=x+100
    print(y)
z=100
```

- 変数の型宣言が不要
- if文の論理式はセミコロンの前までとする
- 論理式がtrueの時に実行される実行単位はインデントされている部分とする

プログラムの基本構造

実行部は波括弧の中に入れる

```
int ButtonPin = 3;  
void setup() 文の最後はセミコロン  
{  
    beginSerial(9600);  
    pinMode(buttonPin, INPUT);  
}  
void xyz(){  
    int abc; ローカル変数の定義  
}  
void loop()  
{  
    xyz(); // 実行したいプログラム  
} //以降はコメント
```

グローバル変数の定義

初期設定プログラム
(起動後、1度だけ実行される)

関数の定義
(機能を小さな単位で分割することが望ましい)

実行プログラム本体
(ループし続ける)

UnoArduSimの使い方

プログラムファイルの読み込み・編集・保存

ファイルを開くのにはダブルクリックではなく、これを使うこと。(ダブルクリックではArduino IDEが立ち上がる)

The screenshot shows the UnoArduSim v1.6 interface. The 'File' menu is open, and three yellow callout boxes with red arrows point to specific menu items:

- プログラムファイルの読み込み** (Load INO or PDE Prog..) with keyboard shortcut CTRL+L.
- プログラムファイルの編集** (Edit/View) with keyboard shortcut CTRL+E.
- プログラムファイルの保存** (Save As) with a note: (次の課題で使うこともあるので、必ず保存すること) (It may also be used in the next assignment, so be sure to save it).

The code editor shows the following code:

```
count=0;
}

void loop()
{
  count=count+1;
  delay(100);
}

//the "int main()" below is IMPLICIT in Arduino
//but is shown here EXPLICITLY by UnoArduSim
int main()
{
  setup();
  while(true)
  {
```

The interface also displays a breadboard simulation with three resistors labeled 'R=1K' and an Arduino Uno board. The I/O Values Scaler is visible on the right side.

UnoArduSimの使い方（編集・コンパイル）

ファイルメニューでEdit/Viewを選ぶと以下のwindowが開く。

プログラムファイルの編集画面およびコンパイル

The screenshot displays the UnoArduSim IDE interface. On the left is a code editor with the following C++ code:

```
/* Use File->Load Prog to
   load a different Program
*/

const int Plus_pin=4;
const int Minus_pin=5;

const int Plus_time=20;
const int Minus_time=80;

void setup()
{
  pinMode(Plus_pin, OUTPUT);
  pinMode(Minus_pin, OUTPUT);
}

void loop()
{
  digitalWrite(Minus_pin, HIGH);
  delay(Minus_time);
  digitalWrite(Minus_pin, LOW);

  digitalWrite(Plus_pin, HIGH);
  delay(Plus_time);
  digitalWrite(Plus_pin, LOW);
}
```

In the center, a yellow box contains the text: **この画面でプログラムファイルを編集** (Edit program files on this screen).

On the right, the 'BUILT-IN's' list is visible, showing various data types and constants. A yellow box highlights the 'Adopt' button, with an arrow pointing to it from the text: **プログラムファイルの実行形式への書き換え** (Conversion of program file to execution format).

UnoArduSimの使い方(エラーの表示)

“Adopt”を押しても、エラーがあると実行できない

エラーを起こしている文が強調されることもある

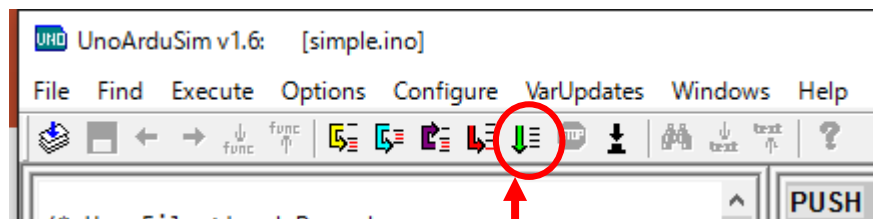
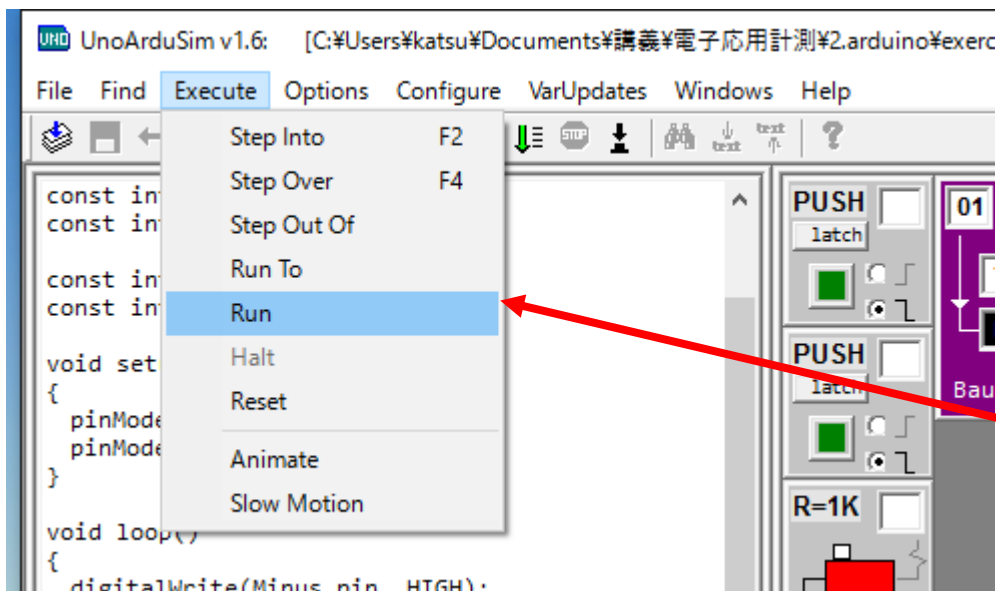
```
void loop()
{
  digitalWrite(Minus_pin, HIGH);
  delay(Minus_time);
  digitalWrite(Minus_pin, LOW);

  digitalWrite(Plus_pin, HIGH);
  delay(Plus_time);
  digitalWrite(Plus_pin, LOW);
}
```

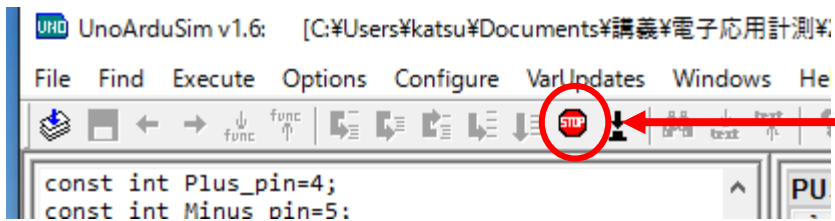
PARSE ERROR: can't find expected close bracket

文法エラー時(PARSE ERROR)には、ここにエラーの内容が表示される

UnoArduSimの使い方(実行方法)



プルダウンメニューから"Run"を選ぶか(左)、実行アイコン(上)を選んで実行させる。

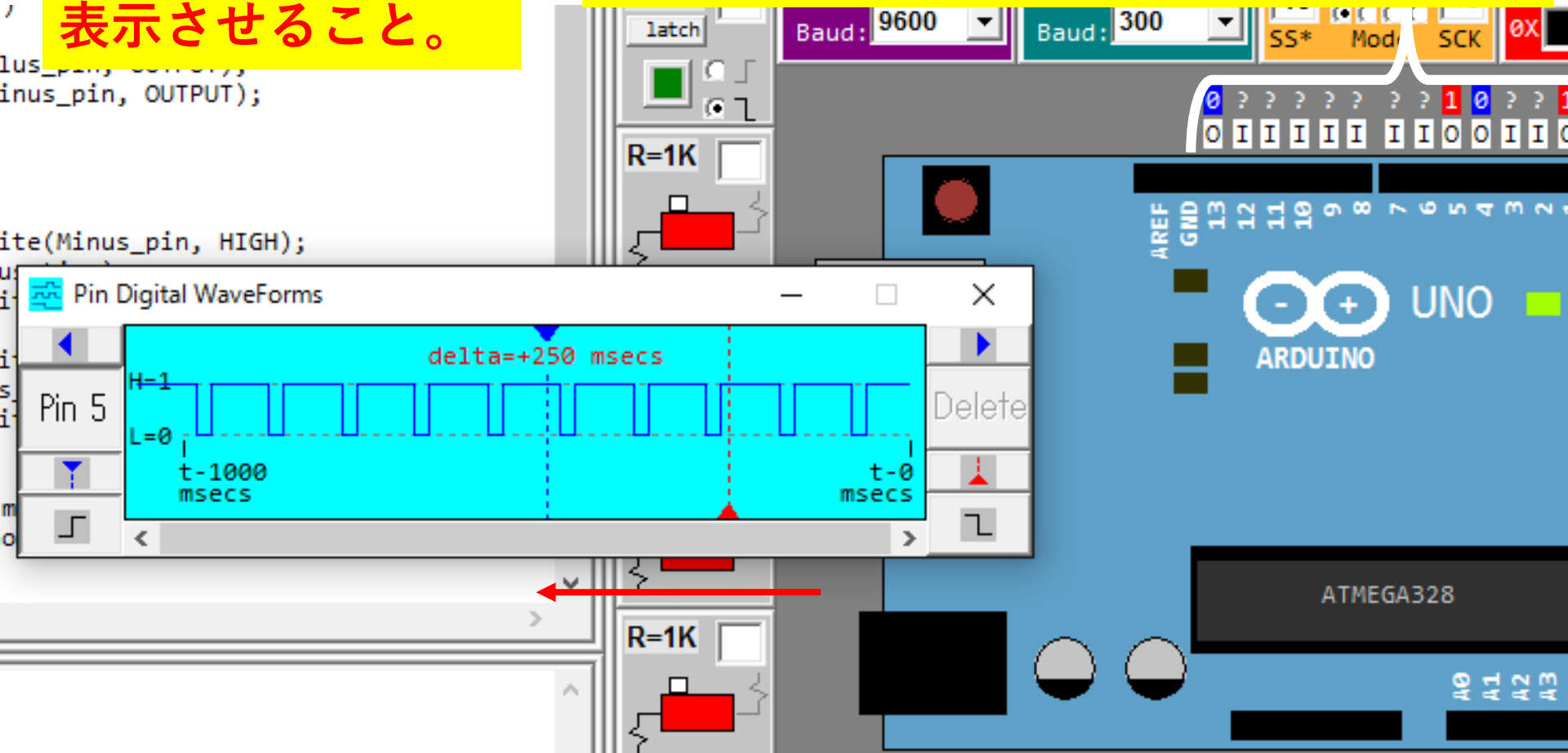


途中で実行をストップするときにはストップアイコンを押す

UnoArduSimの使い方（実行結果の表示法）

演習提出時にはピンの出力結果をオシロの出力として表示させること。

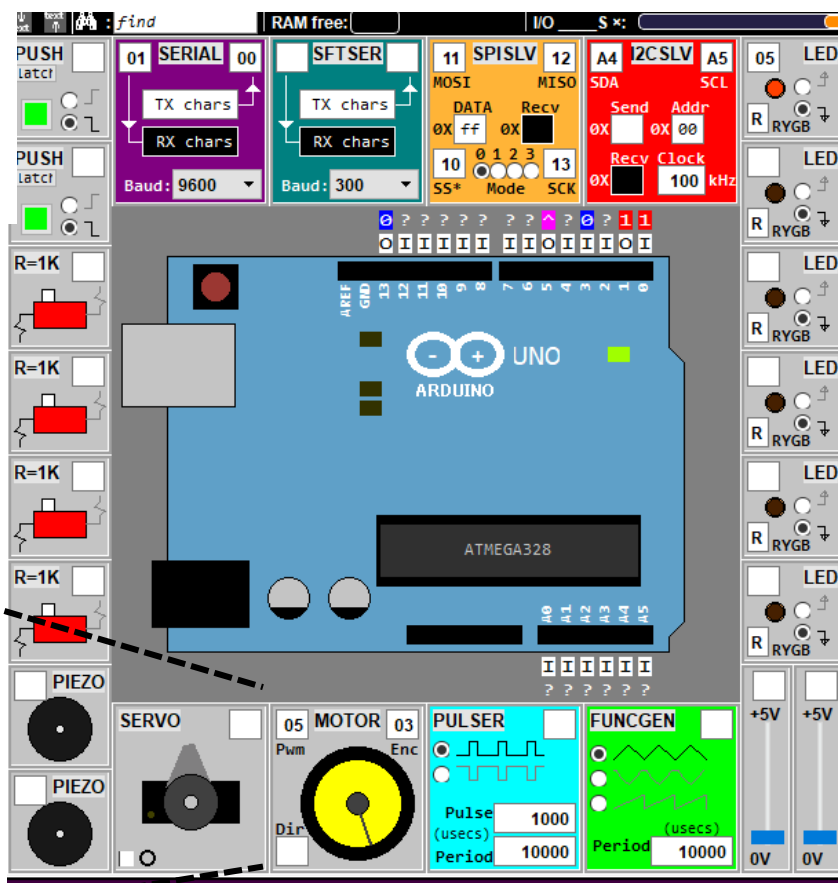
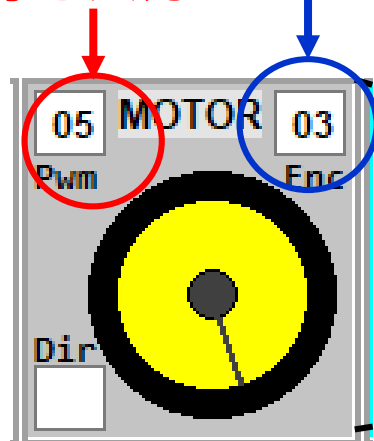
ピンモード（I:INPUT, O:OUTPUT）を表す。ここをクリックすることにより、左側のオシロ出力が表示される。



モーター、LEDの接続方法

Wheel Encoder:ここから1回転あたり8パルスの出力が出てくるので、適当なピンに接続し、その信号を見る

ここにPWM出力のピン番号を入力



ここにLEDに接続するピン番号を入力



PWMピンの出力を観測し、マウスのホイールを使って時間軸を拡大し、デューティー比が変化していることを確かめる。

スイッチの使い方

スイッチの種類

このスイッチの
接続ピン番号

The image shows an Arduino IDE interface with the following components:

- Code Editor:** Contains C++ code for a digital write and read operation. The code includes `digitalWrite(Minus_pin, LOW);`, `digitalWrite(Plus_pin, HIGH);`, `delay(plus_time);`, `digitalWrite(Plus_pin, LOW);`, and a `loop()` function that reads a button state and generates a pulse.
- Breadboard:** A simulated breadboard with a push button labeled 'PUSH' and a red square representing the button's state. The button is connected to a pin labeled '08'. A 'latch' attribute is shown below the button label.
- Digital Waveform:** A graph showing digital signals for Pin 4 and Pin 5. The signals are square waves with a period of 250ms and a pulse width of 100ms. The graph is labeled 'Pin Digital WaveForms' and includes time markers 't-1000 msec' and 't-0 msec'.

Red circles and arrows highlight the 'PUSH' button component, its 'latch' attribute, and the pin number '08'.

スイッチ（押すと状態が変わる）

Pushスイッチの設定

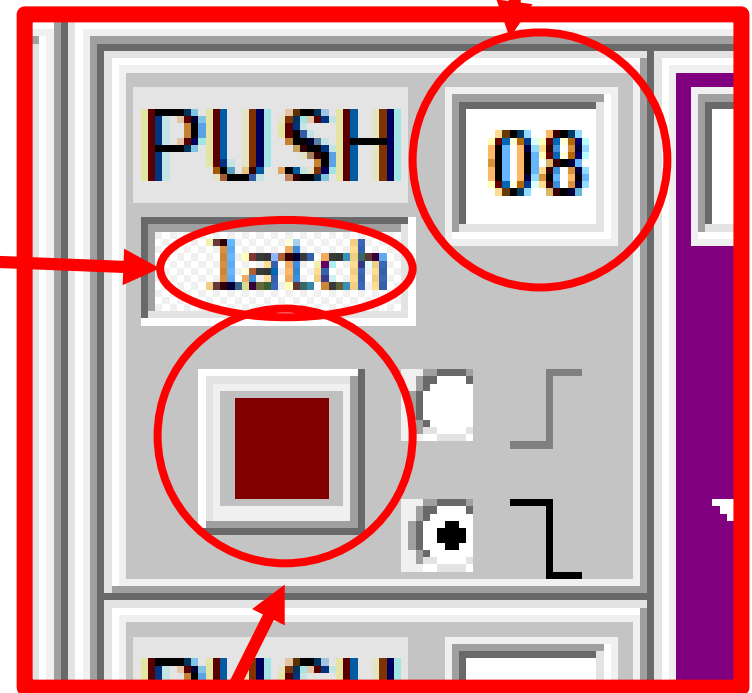
このスイッチの接続ピン番号（キーボードから入力）

スイッチの種類

Latch:押し込むことにより状態を変えられる

latchなし：押しているときだけOn (LOW)になる

latchあり：押す毎にon(LOW)とoff(HIGH)が切り替わる



スイッチ（押すと状態が変わる）