SmT(シント)

松江工業高等専門学校 情報工学科
青笹　誓也

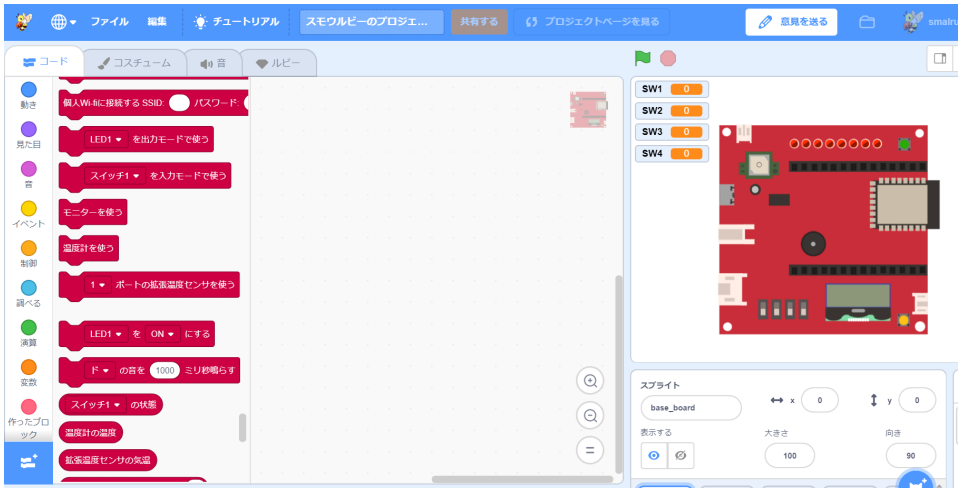
今日の流れ

1. SmTの紹介
2. SmTを使おう
3. LEDを光らせよう
4. 拡張センサから温度をとりサーバへ送ろう
5. mruby/cのソースコードを書いてみよう

SmTの紹介

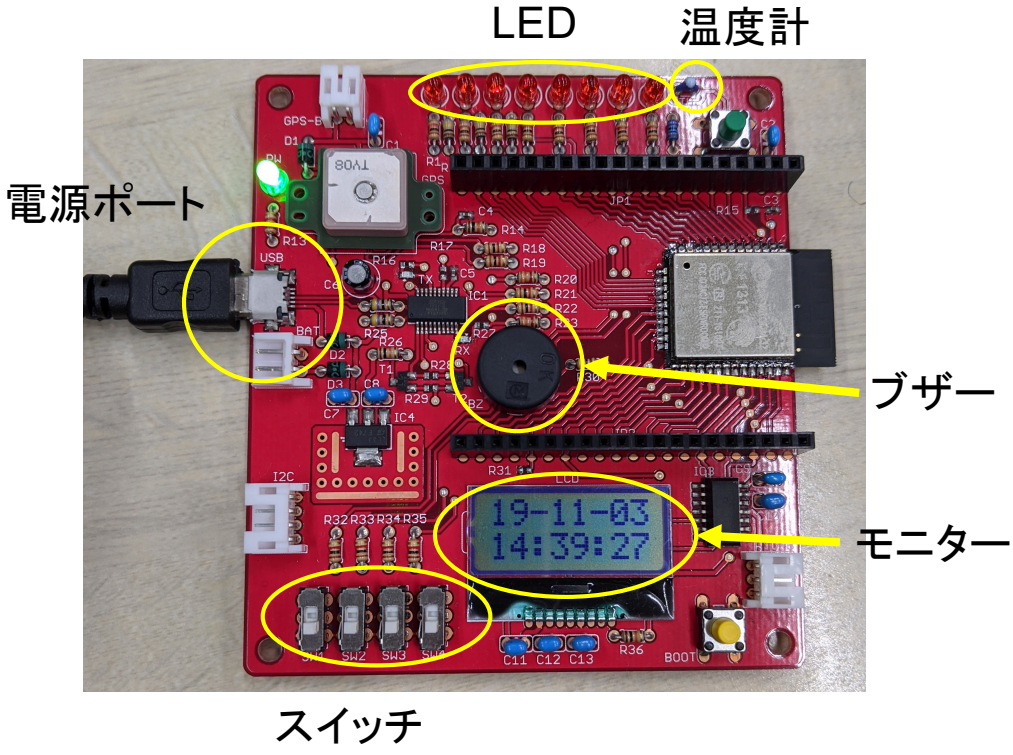
・SmT(シント)

➡ スモウルビーをつかってマイクロコンピュータ(マイコン)を動かせるようにしたもの



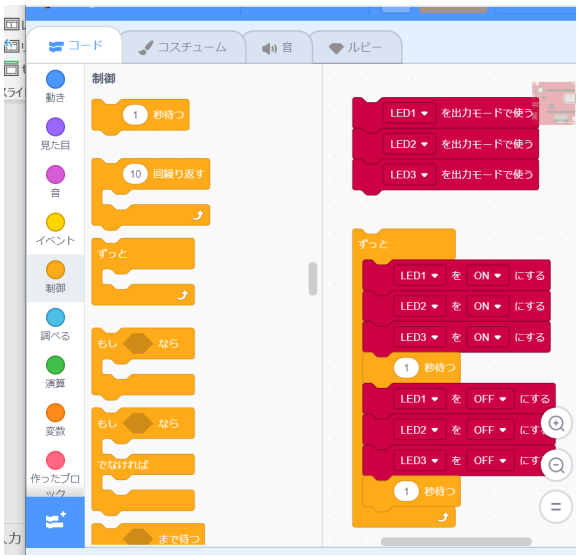
SmTの紹介

・マイコン(SmT用ボード)



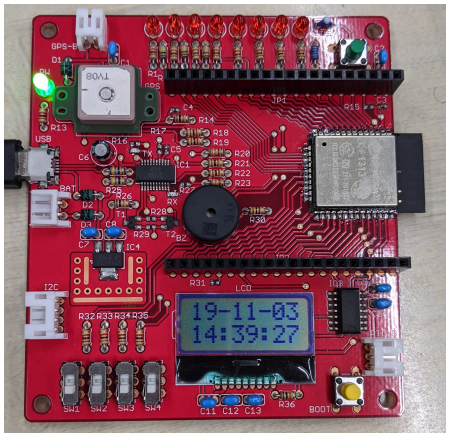
SmTの紹介

パソコンを使って
SmTでプログラムをつくる



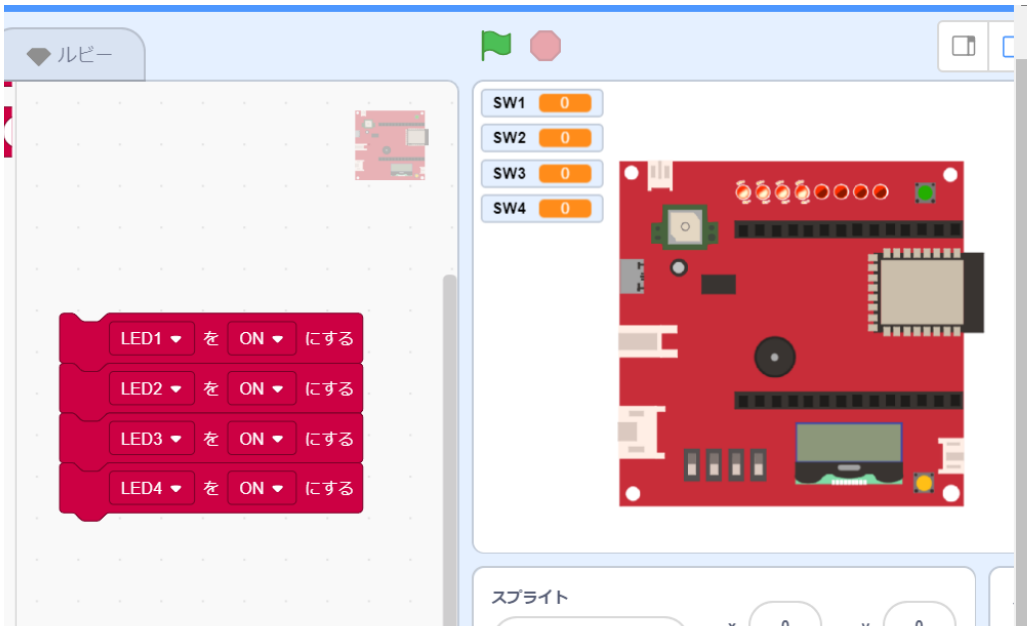
送る

SmTでつくった
プログラムが
マイコンで動く



SmTの紹介

マイコンがなくてもシミュレーターで
プログラムの確認ができる

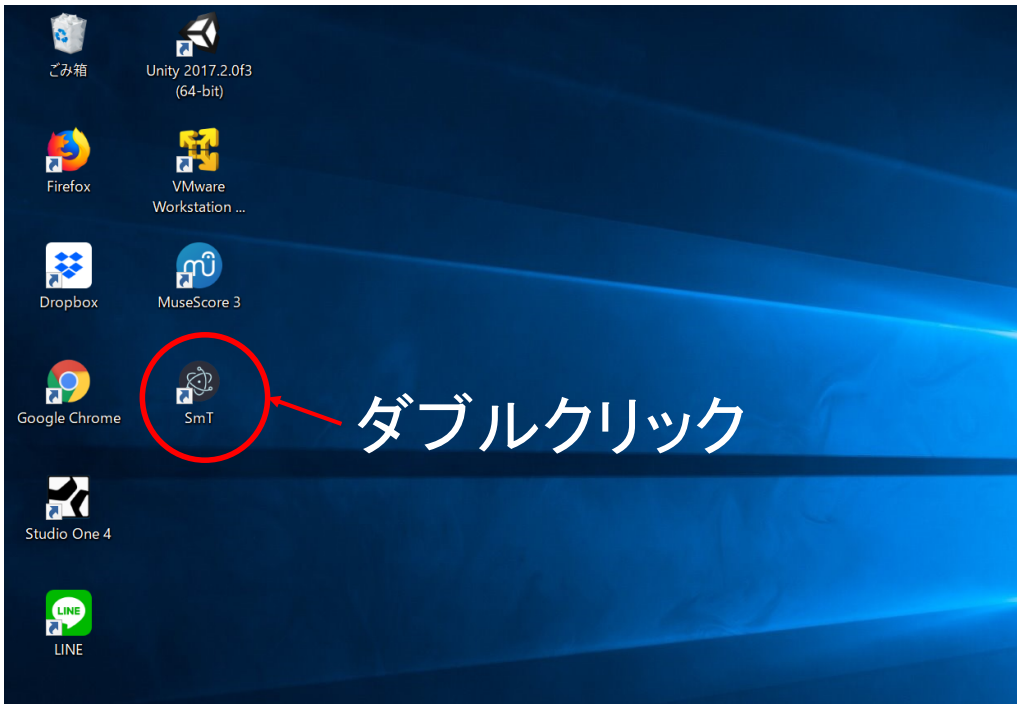


今日の流れ

1. SmTの紹介
2. SmTを使おう
3. LEDを光らせよう
4. 拡張センサから温度をとりサーバへ送ろう
5. mruby/cのソースコードを書いてみよう

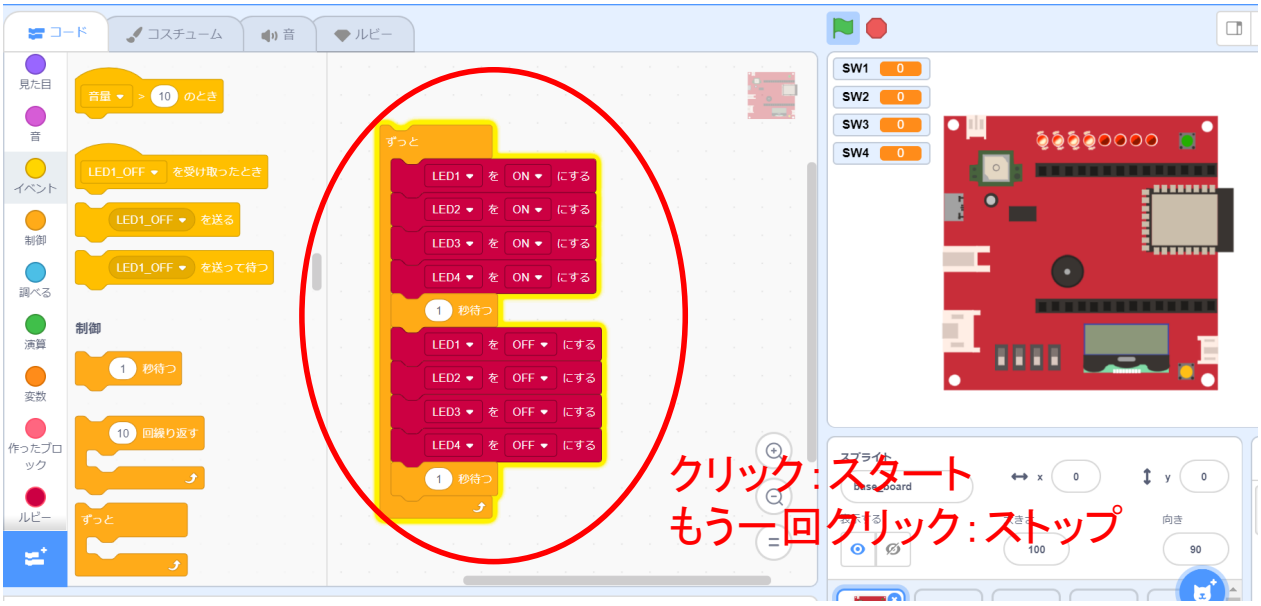
SmTを使おう

起動しよう！



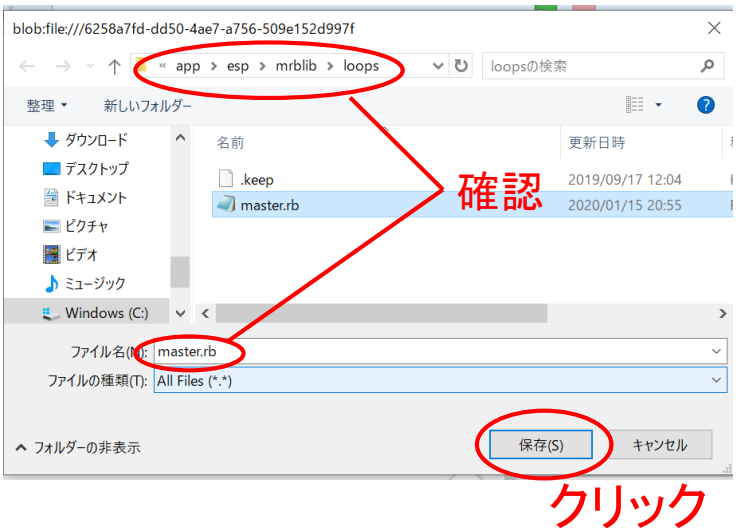
SmTを使おう

- SmTの操作方法はスモウルビーとおなじ
- ブロックをクリックするとシミュレーターが動き，もう一度クリックすると止まる



SmTを使おう

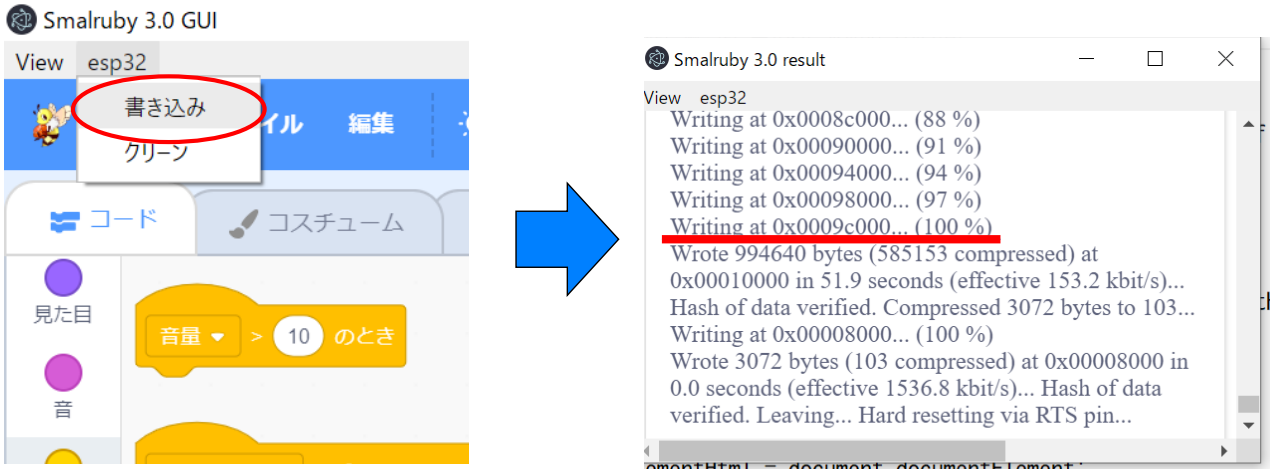
- C:¥SmT-win32-x64¥resouces¥app¥esp¥mrblib¥loops フォルダの中にルビーをmaster.rbという名前になっているのを確認してからプログラムを保存
- 保存を押すと上書きしますか？と聞かれるのではいを選ぶ



SmTを使おう

・プログラムをボードに送るにはメニューバーの
esp32→書き込みのボタンを押す

➡ 結果の画面が表示され，100%になれば完了
(時間が結構かかる)



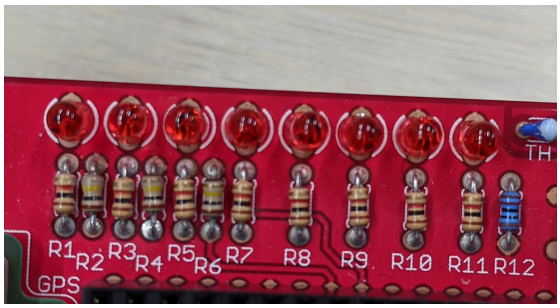
今日の流れ

1. SmTの紹介
2. SmTを使おう
3. LEDを光らせよう
4. 拡張センサから温度をとりサーバへ送ろう
5. mruby/cのソースコードを書いてみよう

LEDを光らせよう

- LEDを光らせるには「光らせたいLEDを出力モードで使う」ブロックと「光らせたいLEDをONにする」ブロックを使う

LED1を1秒ごとにチカチカさせるプログラム

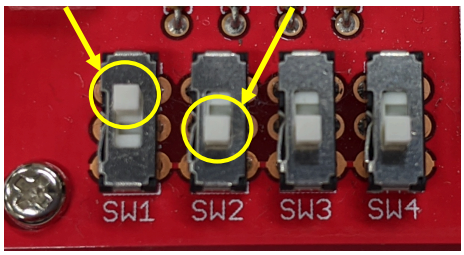


LED1 ⟶ LED8

LEDを光らせよう

- LEDと同じように「スイッチを使う」ブロックを使う
- 「スイッチの状態」ブロックはONの時に1，OFFの時に0を返す

ON：1　　OFF：0

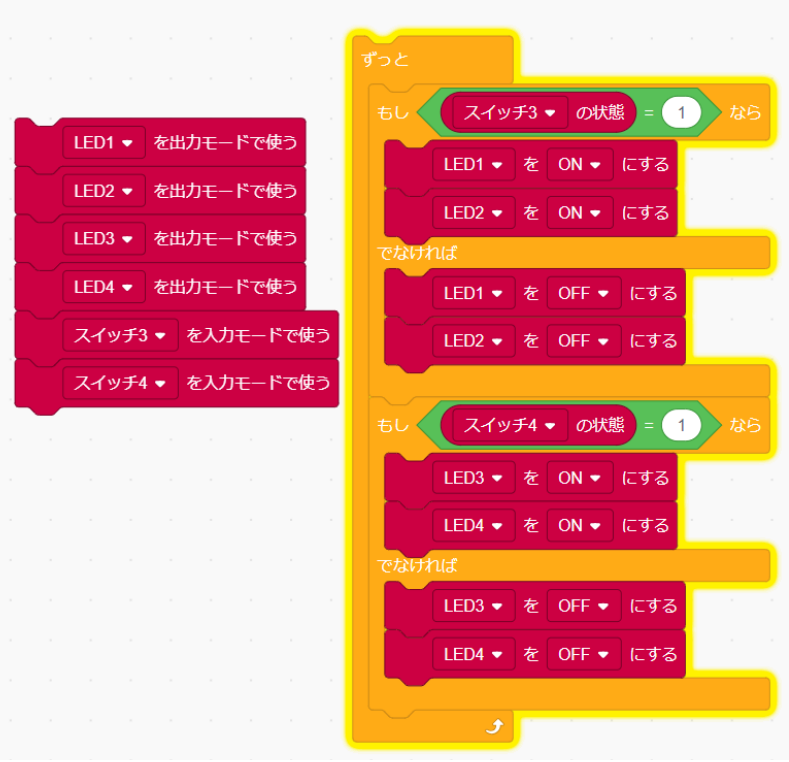


スイッチ1 ⟶ スイッチ4

LEDを光らせよう

スイッチ3がONの時,LED1,2を
スイッチ4がONの時,LED3,4を
点灯させるプログラムを作ろう
(OFFの時は消灯)

LEDを光らせよう

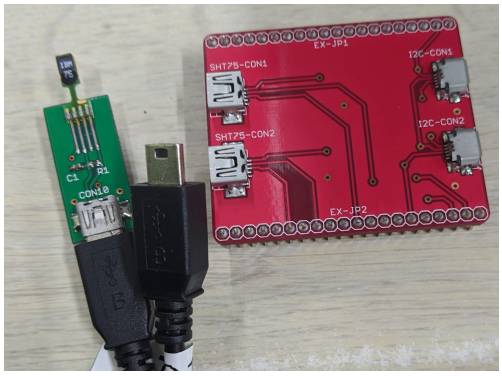


今日の流れ

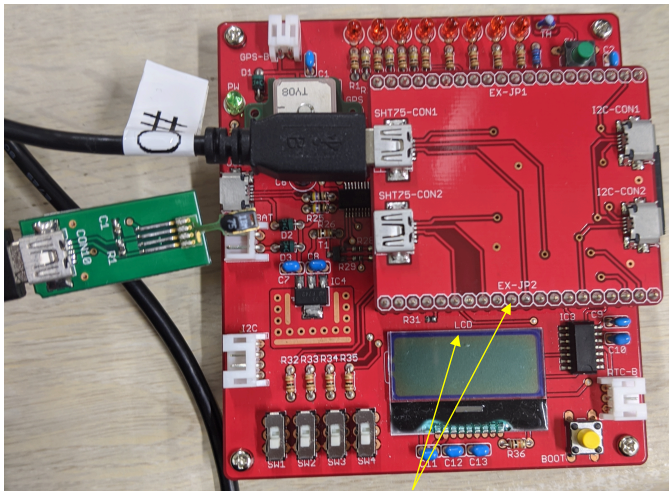
1. SmTの紹介
2. SmTを使おう
3. LEDを光らせよう
4. 拡張センサから温度をとりサーバへ送ろう
5. mruby/cのソースコードを書いてみよう

拡張センサから温度をとりサーバへ送ろう

・拡張ボードと拡張温度センサをマイコンに取り付ける



拡張ボードと拡張センサ

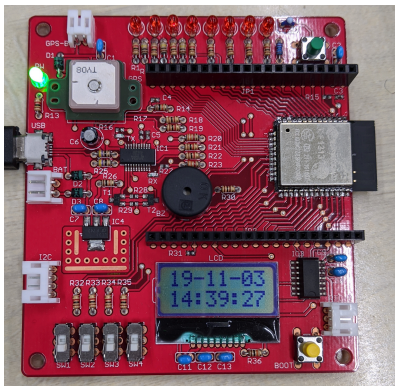


取付方向に注意!!
文字の方向が実験基盤と同じになるように取り付ける

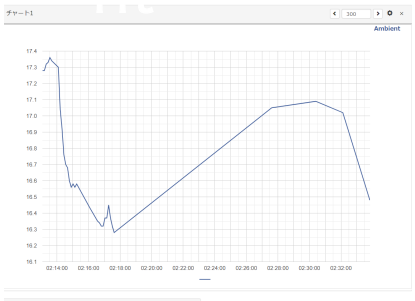
拡張センサから温度をとりサーバへ送ろう

Wi-Fi

Wi-fiで学内ネットワークに接続



Ambie



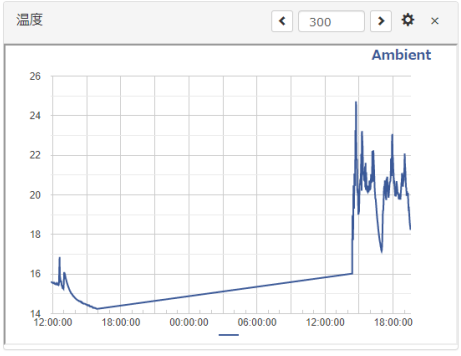
センサから温度と湿度を取得

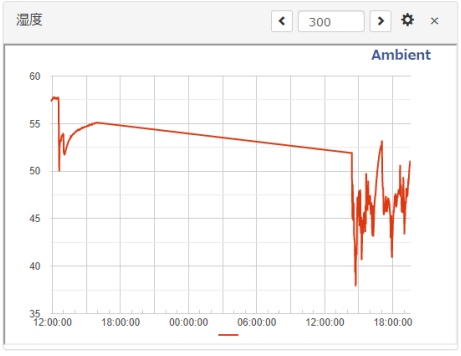
HTTP(ネットワークのルール)でAmbientというサービスにデータを送信

拡張センサから温度をとりサーバへ送ろう

Ambie とは？

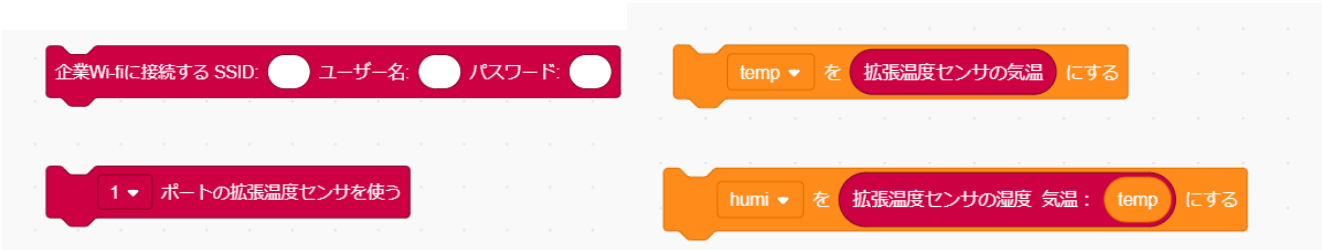
- OSSのグラフ描画ツール
- ユーザ登録後チャネルを作ることで利用可能
- チャネルID,リード(ライト)キー の情報をもつデータをHTTPで送ると自動でグラフを作成してくれる





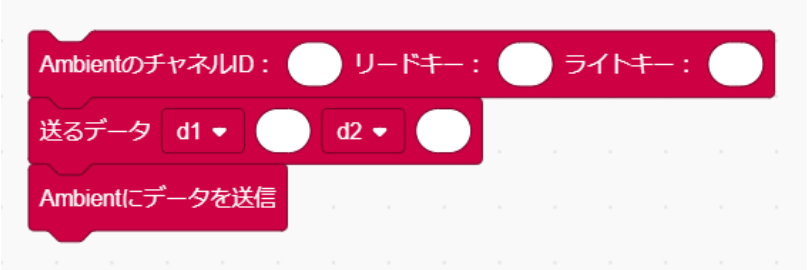
拡張センサから温度をとりサーバへ送ろう

- 「企業Wi-fiに接続する」ブロックで学内Wi-fiに接続できる
- 「拡張温度センサを使う」ブロックでポートを指定して温度センサを使う
- 拡張温度センサは湿度も計測できるが気温のデータが必要になる



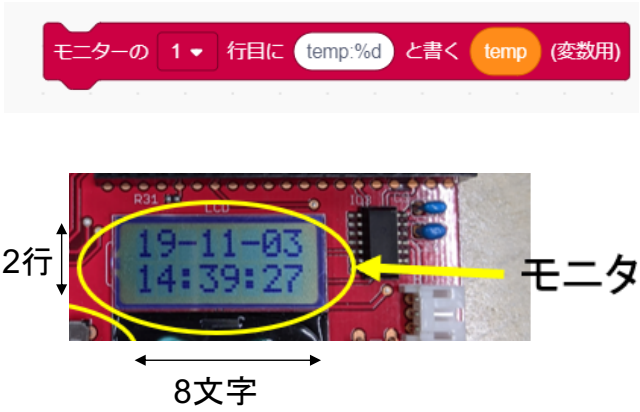
拡張センサから温度をとりサーバへ送ろう

- Ambientにデータを送るにはチャネルID,リードキー,ライトキーを指定する必要がある
- Ambientは一つのチャネルにd1~d8の8個のデータを格納できるので「送るデータ」ブロックで指定する
- データの指定が終わったら「データを送信」ブロックでAmbientに送信する



拡張センサから温度をとりサーバへ送ろう

- LEDと同じように「モニターを使う」ブロックで使える
- モニターは「モニターの何行目に書く」ブロックでアルファベットだけを書くことができる
- 変数を使うときはC言語と同じように書ける



拡張センサから温度をとりサーバへ送ろう

拡張温度センサから気温,湿度を取得し
モニターの1行目にtemp:〇〇
モニターの2行目にhumi:〇〇
と毎秒表示し,そのデータをAmbientに
30秒ごとに送信するプログラムを作ろう

Ambientのパラメータ

チャネルID：17575

リードキー：b0be70026092fb5a

ライトキー：75c3d34cd511e0e6

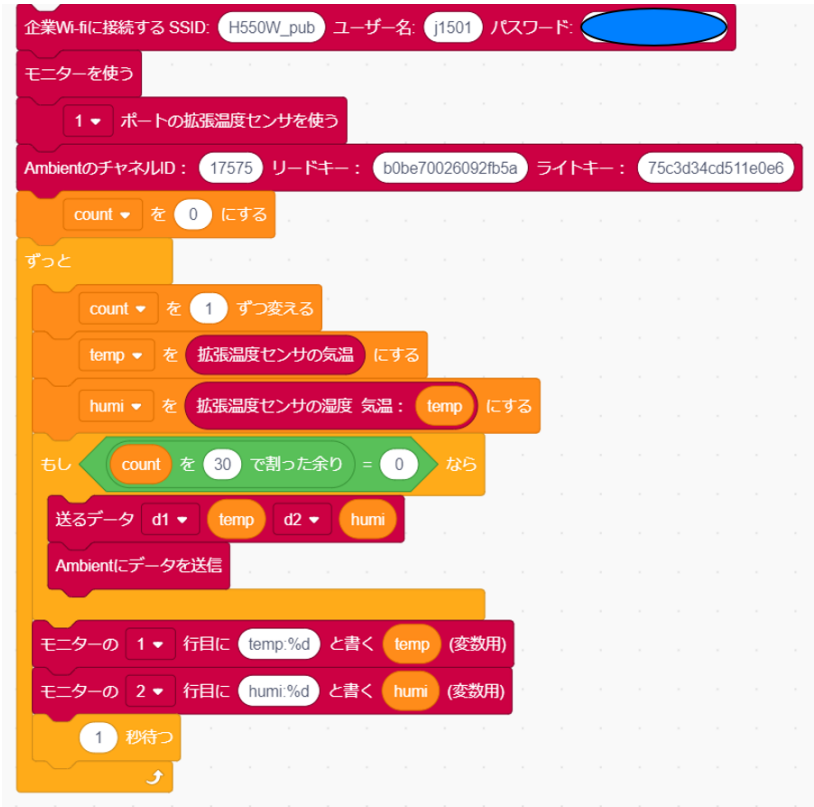
データ番号

あべくん：d3(温度),d4(湿度)

そうくん：d5(温度),d6(湿度)

金崎くん：d7(温度),d8(湿度)

拡張センサから温度をとりサーバへ送ろう



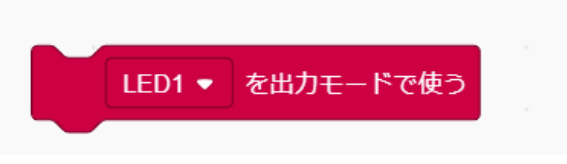
今日の流れ

1. SmTの紹介
2. SmTを使おう
3. LEDを光らせよう
4. 拡張センサから温度をとりサーバへ送ろう
5. mruby/cのソースコードを書いてみよう

mruby/cのソースコードを書いてみよう

・SmTはブロックからmruby/cのソースコードに変換を行ってプログラムを動かしていた

・3,4で作成したブロックプログラムをmruby/cの生のソースコードで書いてみよう



gpio_init_output(13)

ブロックからmruby/cのコードに変換

mruby/cのソースコードを書いてみよう

- mruby/cのプログラムの書き方は基本的にRubyとほとんど同じ
- C言語とは違い#includeやmain関数等も必要ない
- 変数の型宣言は不要で「変数名 = 値」というようにすぐ使える
- 条件分岐は「if 条件文　処理 end (else 処理 end)」となる
- くり返し処理は「while 条件文 処理 end」となる
- 待つ処理は「sleep(秒数)」となる

mruby/cのソースコードを書いてみよう

- 3のプログラムに必要な命令

gpio_init_output(ポート番号)
  - ポート番号のLEDを初期化する

gpio_init_input(ポート番号)
  - ポート番号のスイッチを初期化する

gpio_set_level(ポート番号,設定する状態)
  - 状態を1にするとポート番号のLEDをつけ，0にすると消す

gpio_get_level(ポート番号)
  - ポート番号のスイッチの状態を取得する
    スイッチがONのとき1，OFFのとき0を返す

mruby/cのソースコードを書いてみよう

・LEDとスイッチのポート番号表

器具	ポート番号
LED1	13
LED2	12
LED3	14
LED4	27
LED5	26
LED6	25
LED7	33
LED8	32
スイッチ1	34
スイッチ2	35
スイッチ3	18
スイッチ4	19

mruby/cのソースコードを書いてみよう

・LED1を1秒おきに点滅させるサンプルコード

```
gpio_init_output(13)
while true
    gpio_set_level(13,1)
    sleep(1)
    gpio_set_level(13,0)
    sleep(1)
end
```

・スイッチ3がONなら1秒待つサンプルコード

```
gpio_init_input(18)
while true
    if gpio_get_level(18) == 1
        sleep(1)
    end
end
```

mruby/cのソースコードを書いてみよう

```
gpio_init_output(13)
gpio_init_output(12)
gpio_init_output(14)
gpio_init_output(27)
gpio_init_input(18)
gpio_init_input(19)
while true
  if gpio_get_level(18) == 1
    gpio_set_level(13,1)
    gpio_set_level(12,1)
else
      gpio_set_level(13,0)
    gpio_set_level(12,0)
  end
  if gpio_get_level(19) == 1
    gpio_set_level(14,1)
    gpio_set_level(27,1)
  else
    gpio_set_level(14,0)
    gpio_set_level(27,0)
  end
end
```

mruby/cのソースコードを書いてみよう

・4のプログラムに必要なサンプル

- Wi-fiに接続する

```
initialize_wifi(0,SSID,ユーザ名,パスワード)
```

- モニターの初期化

```
i2c = GpioTest.new(22, 21)
i2c.i2c_init
i2c.lcd_init
```

- 拡張温度センサの初期化

```
sht = GpioTest.new(2,4)
sht.sht_init
```

mruby/cのソースコードを書いてみよう

・4のプログラムに必要なサンプル

➢ Ambientの設定

```
ambient_client_id = “チャネルID"
ambient_read_key = “リードキー"
ambient_write_key = “ライトキー"
url = "http://ambidata.io/api/v2/channels/#{ambient_
      client_id}/data“
```

➢ 気温と湿度の取得

```
temp = sht.sht_get_temp / 100.0
humi = sht.sht_get_humi(temp)
```

mruby/cのソースコードを書いてみよう

・4のプログラムに必要なサンプル

➢ Ambientのデータをセットする

```
data = "{
¥"writeKey¥": ¥"#{ambient_write_key}¥",
¥“d1¥”: #{気温},
¥“d2¥”: #{湿度}
}".tr("¥n", "")
```

➢ モニターに文字列を書き込む

```
i2c.lcd_write(0x00, [ 0x01, 0x80 ] )        //1行目に書く
i2c.lcd_write(0x40, sprintf(“文字列"))
i2c.lcd_write(0x00, [ 0x80 + 0x40 ] )       //2行目に書く
i2c.lcd_write(0x40, sprintf(“%d”,変数))
```

mruby/cのソースコードを書いてみよう

・4のプログラムに必要なサンプル

➢ Ambientにデータを送信する

```
connected = check_network_status()
if connected
  http_client_init(url)
  http_client_set_header("Content-Type",
                         "application/json")
  http_client_set_header("Connection", "close")
  http_client_set_post_field(data)
  get_http_response()
  http_client_cleanup()
end
```

mruby/cのソースコードを書いてみよう

```
initialize_wifi(0,"H550W_pub","j1501","")
i2c = GpioTest.new(22, 21)
i2c.i2c_init
i2c.lcd_init
sht = GpioTest.new(2,4)
sht.sht_init
ambient_client_id = "17575"
ambient_read_key = "b0be70026092fb5a"
ambient_write_key = "75c3d34cd511e0e6"
url = "http://ambidata.io/api/v2/channels/#{ambie
nt_client_id}/data"
$count = 0
while true do
  $count += 1
  $temp = sht.sht_get_temp / 100.0
  $humi = sht.sht_get_humi($temp)
  if $count % 30 == 0
    data = "{
    ¥"writeKey¥": ¥"#{ambient_write_key}¥",
    ¥"d1¥": #{$temp},
    ¥"d2¥": #{$humi}
    }".tr("¥n", "")
    connected = check_network_status()
    if connected
      http_client_init(url)
      http_client_set_header("Content-
                    Type", "application/json")
      http_client_set_header("Connection", "close")
      http_client_set_post_field(data)
      get_http_response()
      http_client_cleanup()
    end
  end
  i2c.lcd_write(0x00, [ 0x01, 0x80 ] )
  i2c.lcd_write(0x40, sprintf("temp:%d",$temp))
  i2c.lcd_write(0x00, [ 0x80 + 0x40 ] )
  i2c.lcd_write(0x40, sprintf("humi:%d",$humi))
  sleep(1)
end
```