シングルタスク
プログラミング

本田　晋也
名古屋大学 大学院情報科学研究科
honda@ertl.jp

最終更新
2016/6/27

概要

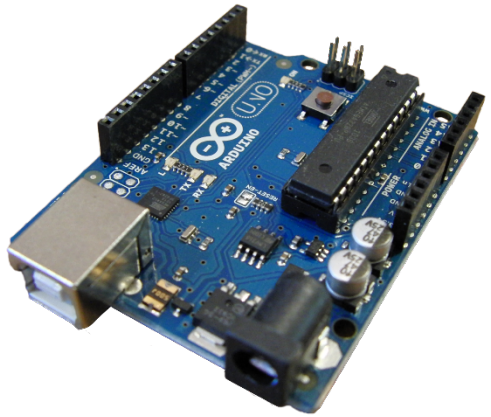
シングルタスクプログラミングについて学習する
ArduinoボードとArduino IDEを使用する

- アジェンダ
  - Arduinoとは
  - 開発環境のセットアップ
  - 単一機能のプログラム
  - 複合機能のプログラム

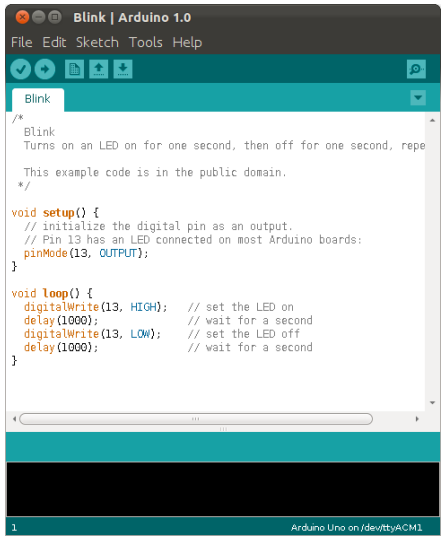
Arduinoとは

Arduino

- Makerの作品で広く使われているマイコンボードとIDEをセットにした環境
  - 安価で容易に使える
  - 書籍等の情報が豊富



- Arduinoボード
  - 各種マイコンを用いたボード（数十種類存在）
  - Arduino　Uno：最も一般的なArduinoボード
    - Atmel ATmega328P,　Flash 32KB, RAM 2KB
  - オープンハードウェアなためクローンのボードも存在



- ArduinoIDE
  - マルチプラットホームの開発環境
  - インストーラによりコンパイラ(GCC)やArduinoライブラリがインストールされる
  - ボタンを押すだけでコンパイルとボードへの書き込みが可能
  - デバッグ機能はない（printfデバッグ）
  - Arduinoライブラリを含む(ライセンスはGPL)

Arduino

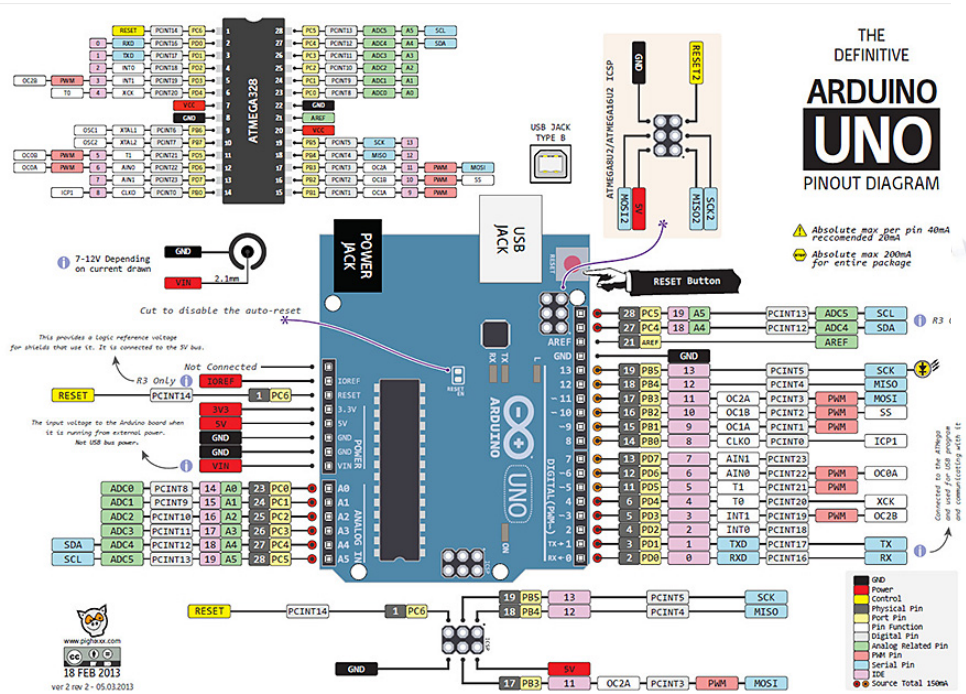
- Arduinoプログラミングモデル
  - setup()/loop()による容易なモデル
  - C++ベースの独自言語
    - プロトタイプ等は必要なし
  - Arduino IDEがC++に変換してコンパイル

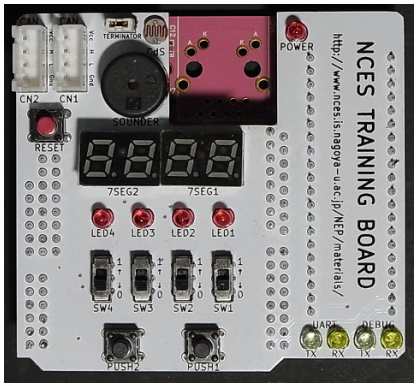
```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

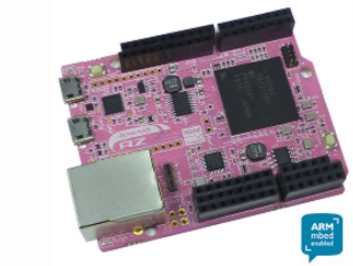
- Arduinoライブラリ
  - コアライブラリ
    - IO操作(GPIO, AD, SPI, I2C)，時間，文字列操作
  - Arduino準拠ライブラリ
    - コアライブラリの上で実現されたライブラリ
    - Arduino IDEに含まれるライブラリ
      - ✓SD，LCD，USB，Audio等のライブラリ
    - その他，センサーやシールドに含まれるライブラリ
      - ✓センサー値の変換，Wifiモジュール制御，IoTサービスへの接続

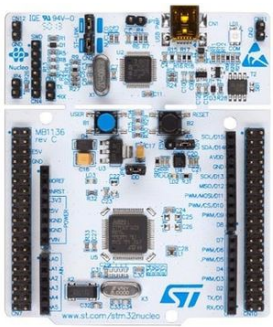
Arduino

- シールド
  - 拡張ピンに接続することによりハードウェア機能を拡張するボード
    - 物理的には拡張ボードにおけるデファクトとなっている
  - 電気的互換性がない場合があるので注意(5V or 3.3V)
  - Wifi, LCD, SD，ロボット，センサー





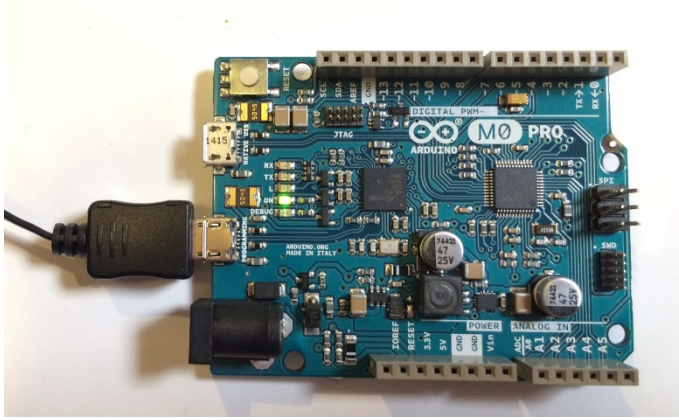




開発環境のセットアップ

必要な機材

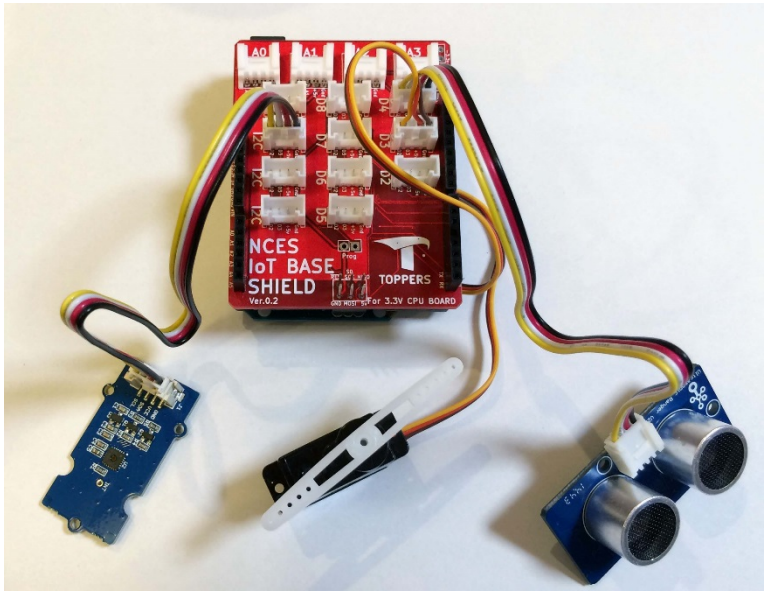
- ホストPC
  - Windows or Mac OS
  - 本チュートリアルではWindowsで説明
  - Linuxでも動作するはず

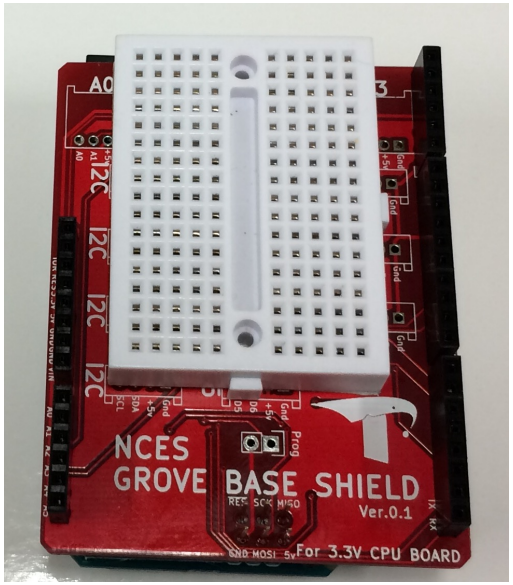


- Arduino M0 Pro
  - Cortex-M0+ 48MHz/ROM 256KB/RAM 32KB
  - デバッガ機能あり（EDBG(Atmel's Embedded Debugger)）
    - デバッガ機能なしのArduino M0 もあるが推奨しない
      - ✓Arduino UNOとの互換性が低いためプログラムの書き換えが必要
  - 一般的なArduinoボード(Arduino UNO)との違い
    - ARM(Cortex-M0)を搭載（UNO等はAVR）
    - IOが3.3V (UNO等は5V)

NCES IoTシールド

- IoT機器のプロト開発向けのシールド
  - ESP8266によるWifi機能(UART接続)
  - MicroSDスロット(SPI接続)
  - Grove Systemと互換のコネクタ（3.3V↔5V変換）
  - Groveのコネクタを実装しない場合はブレッドボードを置ける



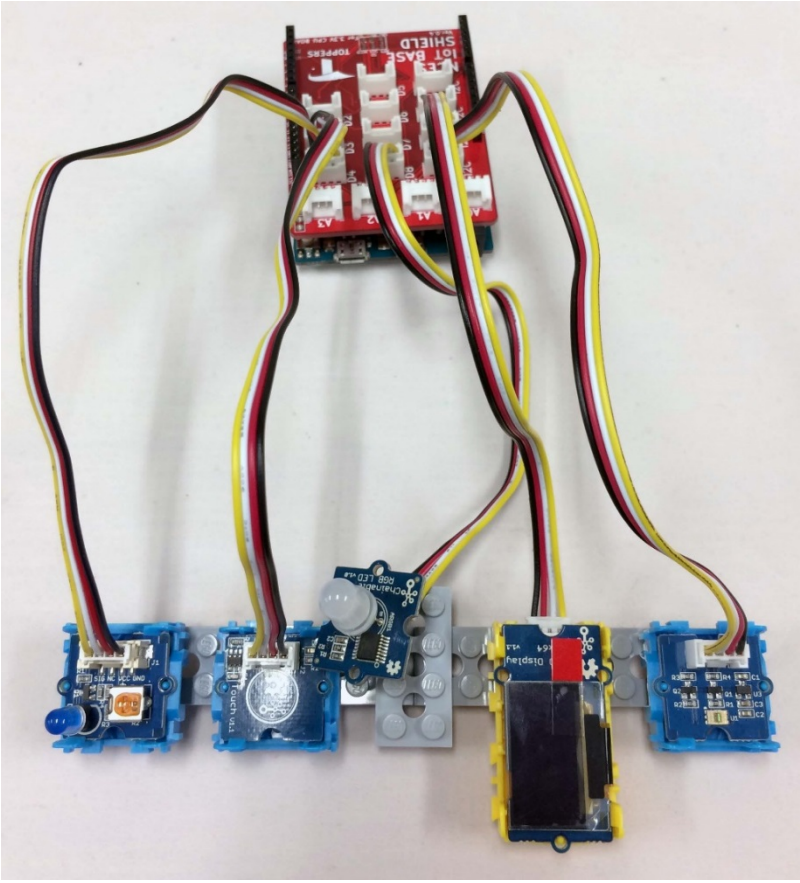


NCESIoTシールド: SD/Wifi/Grove System

- SD
  - Chip Selectをピン10とすることで使用することができる
- Wifi
  - ESP8266を使用しているのでWifiを使用するサンプルを動作させることが可能
- Grove System
  - 各種センサやアクチュエータを簡単に着脱可能なモジュール
  - 100種類以上のモジュールがリリースされている
  - 各GroveモジュールにはWikiページとArduinoライブラリが用意されている

NCES IoT Package

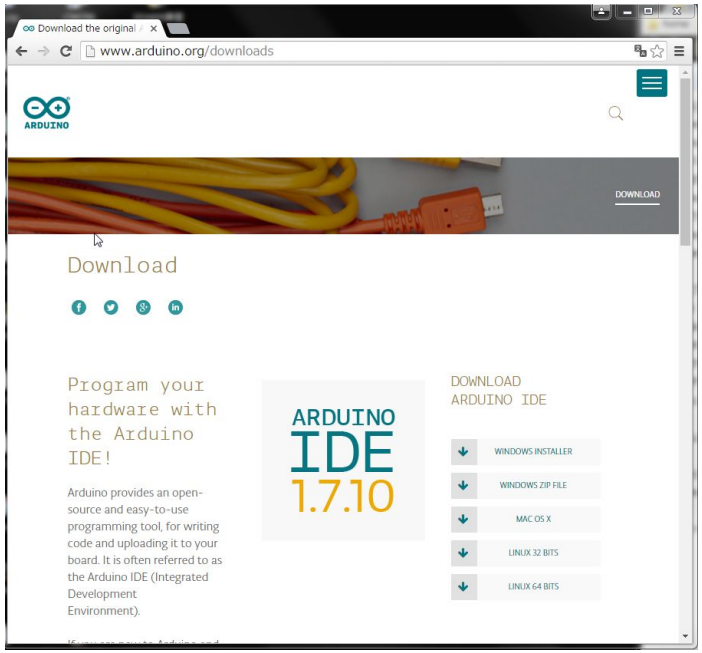
- Arduino M0 + NCS IoT ボード に各種センサーを追加したパッケージ
  - Grove – LED
    - 単色のLEDを1個搭載
  - Grove - Touch Sensor
    - タッチセンサーを1個搭載
  - Grove - Chainable RGB LED
    - 三原色の組み合わせで発光するLED
    - チェーン接続が可能
  - Grove OLED Display 0.96
    - 128×64の表示が可能な
ディスプレイ
  - Grove Digital Light Sensor
    - 可視光の明るさを取得可能なセンサ



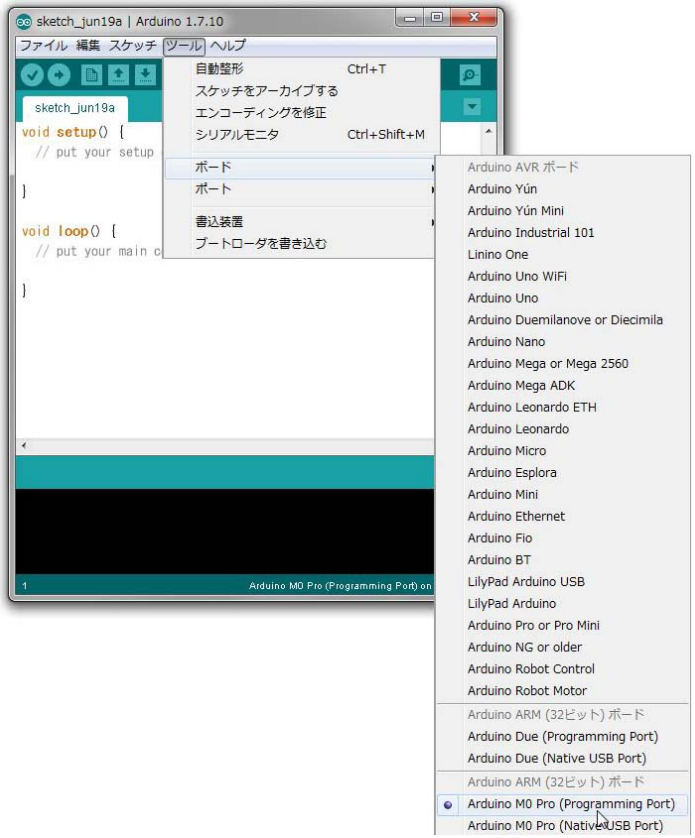
ツールのインストール

- Arduino IDE
  - Arduino.org(http://www.arduino.org/downloads）からダウンロード
    - 動作確認済みバージョン：1.7.10
  - !!Arduino.ccではないため注意!!
  - インストーラによりインストール
    - GCC/Make/GDB/OpenOCDがインストールされる
- ターミナルエミュレータ
  - Teraterm等をインストール

- 以下のツールはオプション
  - Atmal Studio
    - GUIによるデバッグ
    - 7.0で確認済み



Arduino IDEの設定と接続

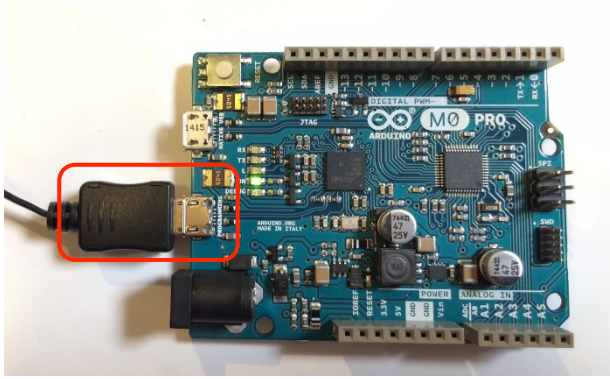


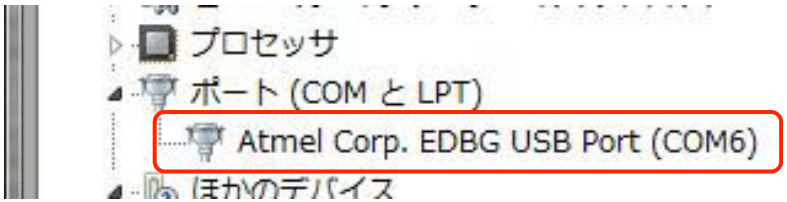
Arduino IDEの設定

- Arduino IDEを起動
- “ツール”→”ボード”→
  ”Arduino M0 Programming Port”を選択

ボードの接続

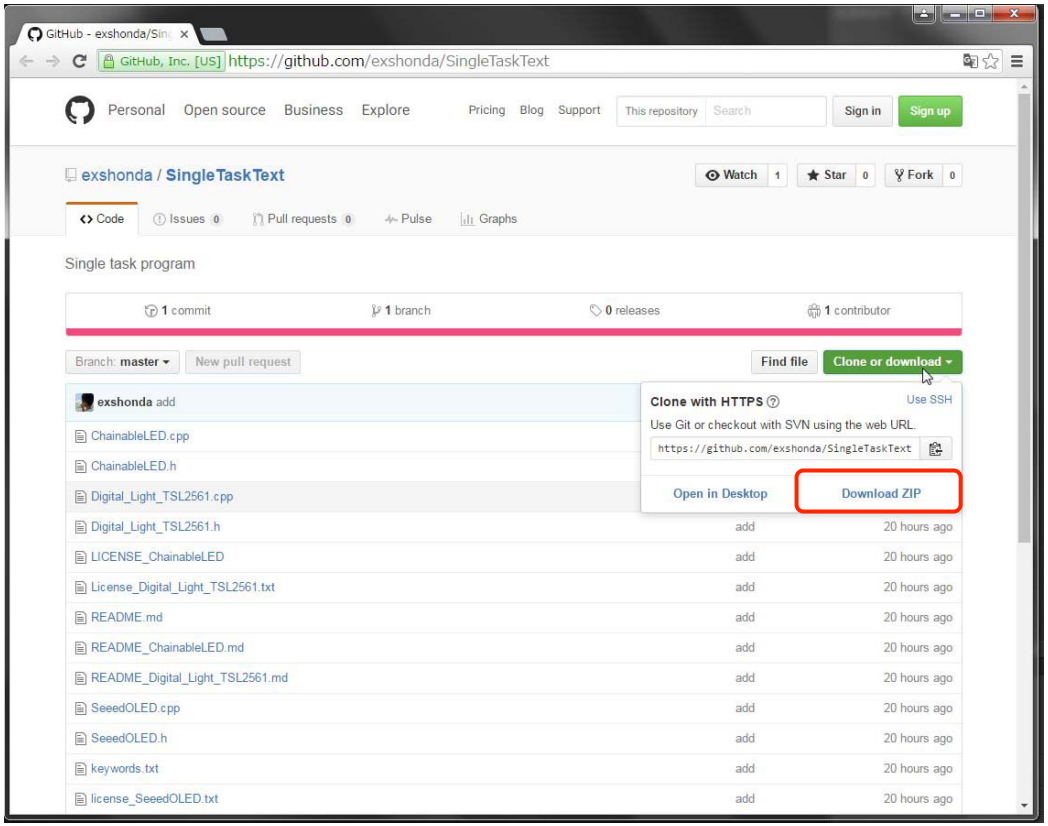
- M0の**PROGRAMMINGポート**とPCをUSBケーブルで接続
- ドライバがインストールされ，COMポートとEDBGが認識される
- COMポートの番号を確認して，Teraterm等で115200bpsで接続





ライブラリのインストール

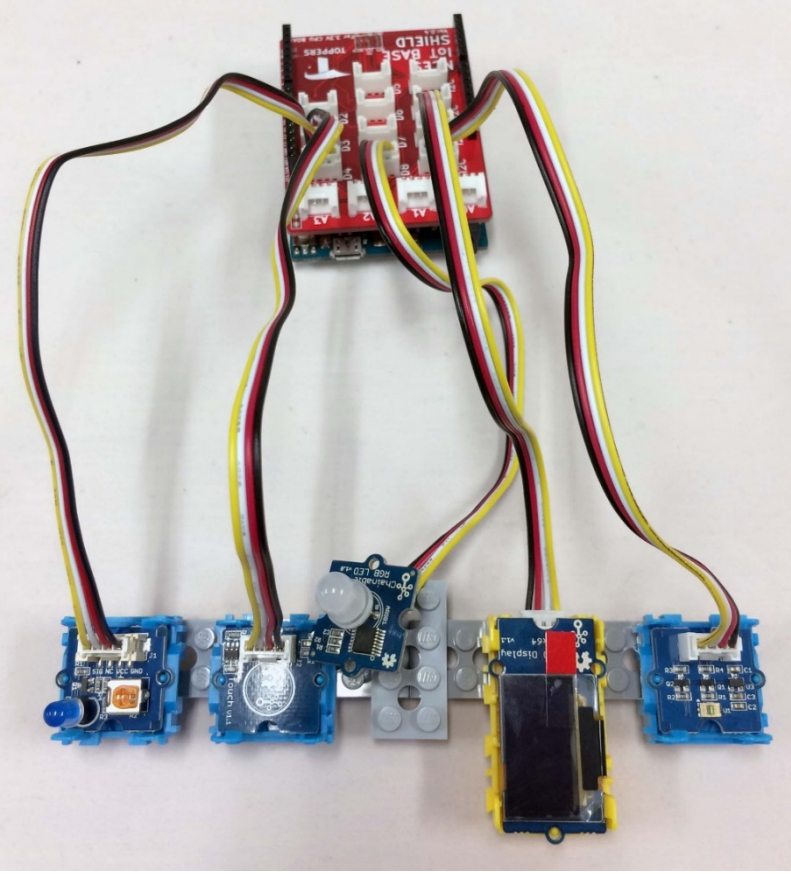
- GitHubにアクセス(https://github.com/exshonda/SingleTaskText)
- Clone or download ZIPからZIPファイルをダウンロード
- ダウンロードしたファイルを展開して，”マイドキュメント¥Arduino¥libraries に置いて，” SingleTaskText-master”から” SingleTaskText”に名称を変更



NCES IoT Package：ハードウェアセットアップ

- Grove – LED
  - D4に接続
- Grove - Touch Sensor
  - D3に接続
- Grove - Chainable RGB LED
  - D8に接続
- Grove OLED Display 0.96
  - I2Cに接続
- Grove Digital Light Sensor
  - I2Cに接続

- I2C
  - SDA/SCLの2線を用いた低速バス



プログラムの実行方法

- Arduino IDEを起動
- Arduino IDEのメニューから“ファイル”→”スケッチブックの例” → ”libraries” →”SingleTaskText” →”LedBlink”を選択
- メニューのアイコンの“マイコンボードへ書き込む”をクリック
- Grove LEDが点滅することを確認

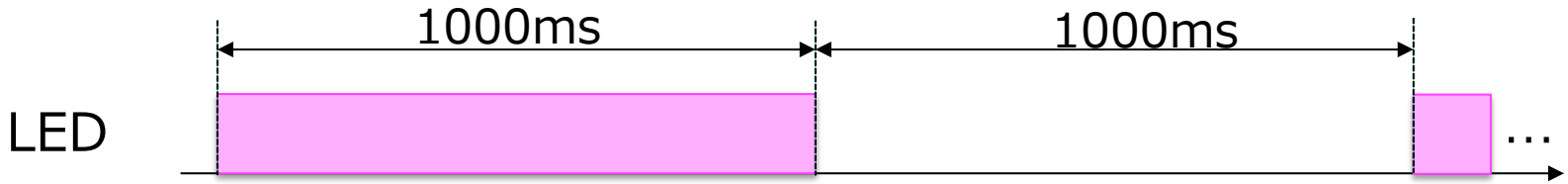
単一機能のプログラム

単一機能のプログラム

- 単一の機能のプログラムについて解説する
  - プログラム1(LEDBlink)
    - 1000m周期でLEDを点滅
  - プログラム2(CLEDBlink)
    - 250m周期でChange LEDの色を変更
  - プログラム3(TouchSense)
    - Touchセンサの検知によるLEDのON/OFF切り替え
  - プログラム4(LuxSense)
    - 光センサの値のOLEDへの出力

プログラム1(LEDBlink)

- 1000ms周期でLEDを点滅させる

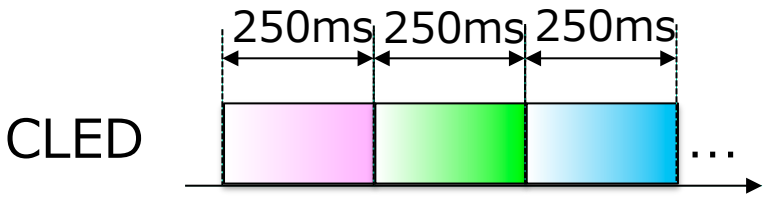


- プログラム
  - setup()
    - Grove LEDが接続されているポートを出力に初期化
  - loop()
    - Grove LEDが接続されているポートに1000ms毎にHIGHとLOWを出力

```
#define LED_PIN 4

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

void loop() {
  digitalWrite(LED_PIN, HIGH);
  delay(1000);
  digitalWrite(LED_PIN, LOW);
  delay(1000);
}
```

プログラム2(CLEDBlink)

- 250m周期でChange LEDの色を変更
  - Red, Green, Blueの順で点灯
  - 10ms毎に輝度を上げる
- プログラム
  - setup()：leds.init()により初期化
  - loop()：leds.setColorRGB()により出力
    - 輝度を上げるため10m周期のループを実行

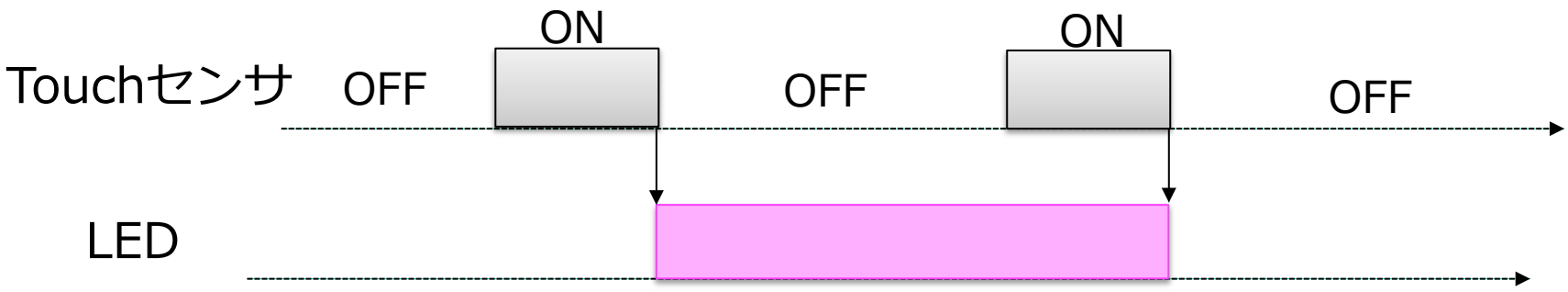



```
#include <ChainableLED.h>
#include <Wire.h>

#define NUM_LEDS 1
ChainableLED leds(8, 9, NUM_LEDS);

void setup() {
  leds.init();
}
```

```
void loop() {
  int i;
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, i*10, 0, 0);
    delay(10);
  }
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, 0, i*10, 0);
    delay(10);
  }
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, 0, 0, i*10);
    delay(10);
  }
}
```

プログラム3(TouchSense)

- Touchセンサが押されたらLEDのON/OFFを切り替える
  - Touchセンサは押されていない状態で'0',押された状態で'1'を読み込み込める
  - Touchセンサが押されたことをOFF-ON-OFFの検知で判定する



プログラム3(TouchSense)

- setup()
  - ピンの初期化とLEDの初期化
- loop()
  - TouchState：OFF-ON-OFFの検知のための変数
  - PreTouchValue：前のTouchセンサの状態

```
#define TOUCH_PIN 3
#define LED_PIN  4
int PreTouchValue = 0;
int TouchState = 0;
int LEDState = 0;
void setup() {
  pinMode(TOUCH_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
}
```

```
void loop() {
  int TouchValue = digitalRead(TOUCH_PIN);
  if ((PreTouchValue == 0) && (TouchValue == 1)
       && (TouchState = 0)) {
    TouchState = 1;
  }
  if ((PreTouchValue == 1) && (TouchValue == 0)
    && (TouchState = 1)) {
    TouchState = 0;
    if (LEDState == 0) {
      LEDState = 1;
      digitalWrite(LED_PIN, HIGH);
    }  else {
      LEDState = 0;
      digitalWrite(LED_PIN, LOW);
    }
  }
  PreTouchValue = TouchValue;
}
```

プログラム4(LuxSense)

- 光センサの値のOLEDへの出力
  - 光センサとOLED共にI2Cによりアクセス
    - アクセスに時間が必要であり割込みを使用する
  - 表示毎にカウンタをインクリメントして表示
- プログラム
  - OLEDCount：変数カウンタ

```
#include <Wire.h>
#include <Digital_Light_TSL2561.h>
#include <SeeedOLED.h>

void setup() {
  Wire.begin();
  TSL2561.init();
  SeeedOled.init();
  SeeedOled.deactivateScroll();
  SeeedOled.setNormalDisplay();
  SeeedOled.clearDisplay();
}
```

```
int OLEDCount = 0;
void loop() {
  int lux;
  lux = TSL2561.readVisibleLux();
  SeeedOled.setTextXY(0, 0);
  SeeedOled.putString("Digital Light");
  SeeedOled.setTextXY(1, 0);
  SeeedOled.putString("Sensor Value is");
  SeeedOled.setTextXY(2, 0);
  SeeedOled.putNumber(lux);
  SeeedOled.putString(" lux      ");
  SeeedOled.setTextXY(4, 0);
  SeeedOled.putString("Count is");
  SeeedOled.setTextXY(5, 0);
  SeeedOled.putNumber(OLEDCount++);
}
```

複合機能のプログラム

複合機能のプログラム

- 単一機能のプログラムを組み合わせることで複数の機能を持つプログラムを実現する

- プログラム5(LEDBlink_CLEDBlink)
  - プログラム1とプログラム2の組み合わせ

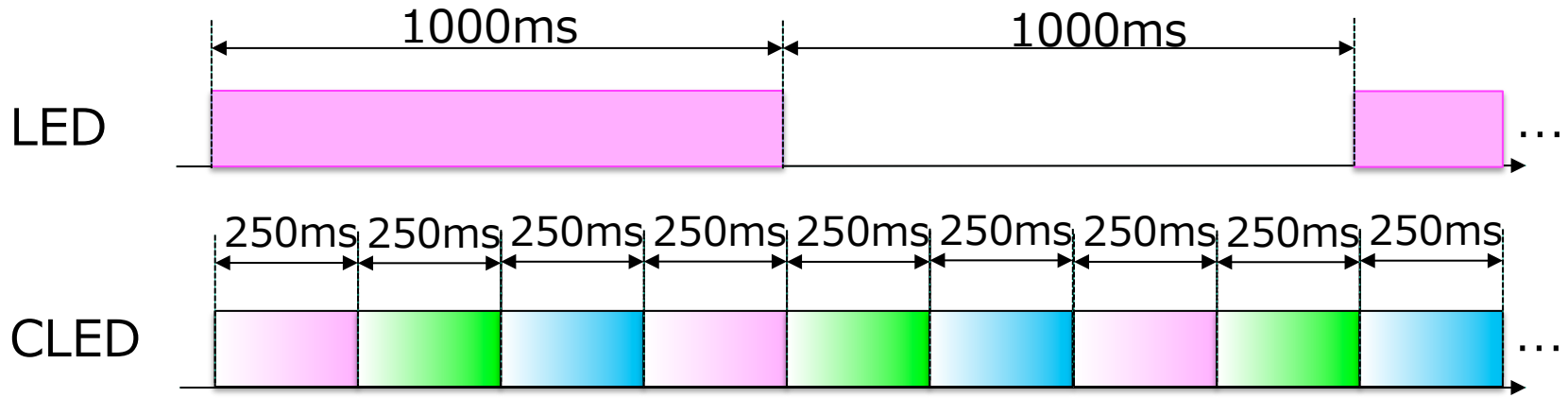
- プログラム6(TouchSense_LuxSense)
  - プログラム3とプログラム4の組み合わせ

- プログラム7(Interrupt)
  - プログラム6の割込み版
  - 処理をloop()と割込みハンドラに分ける

- プログラム8(Interrupt_Blink)
  - プログラム7 をベースにTouchセンサで検知するとLEDを2秒間点灯する

プログラム5(LEDBlink_CLEDBlink)

- プログラム1とプログラム2の組み合わせ
  - プログラム1
    - 1000m周期でLEDを点滅させる
  - プログラム2
    - 250m周期でChange LEDの色を変更
      - ✓Red, Green, Blueの順で点灯
      - ✓10ms毎に輝度を上げる



プログラム5：プログラム

- 別々のloop()処理を1つにまとめる
  - 周期の短い(CLED)方をベースにする(逆は機能を実現できない)
- LED処理は250m周期毎に呼び出されること前提に変更
  - 周期(led_count)とLED状態(led_stae)変数の追加
- loop()内でLED処理を呼び出す

```
int led_count;
int led_state;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  leds.init();
  led_count = 0;
  led_state = 0;
}
```

```
void led_check(){
  led_count++;
  if (led_count == 4) {
    led_count = 0;
    if (led_state == 0) {
      digitalWrite(LED_PIN, HIGH);
      led_state = 1;
    } else {
      digitalWrite(LED_PIN, LOW);
      led_state = 0;
    }
  }
}
```

```
void loop() {
  int i;
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, i*10, 0, 0);
    delay(10);
  }
  led_check();
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, 0, i*10, 0);
    delay(10);
  }
  led_check();
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, 0, 0, i*10);
    delay(10);
  }
  led_check();
}
```

プログラム5：問題点

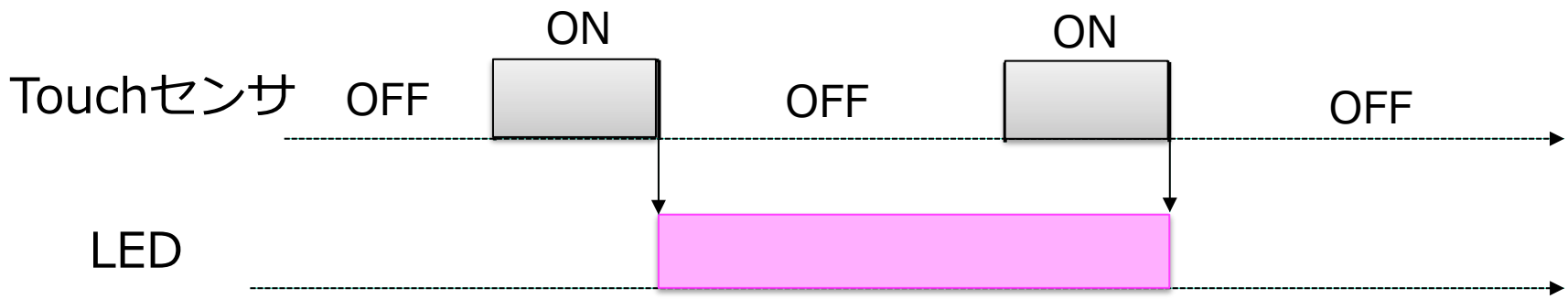
- 独立した処理のプログラムを関連付けて開発する必要がある
  - 時間的には独立していないためプログラム的に関連してしまう
    - CLEDのコードにLEDの実行呼び出しコードが入る
  - 片方の周期に変更があった場合，もう片方も変更が必要な場合が出てくる
    - 例）LEDの点滅周期を400msに変更

```
void led_check(){
  led_count++;
  if (led_count == 4) {
    led_count = 0;
    if (led_state == 0) {
      digitalWrite(LED_PIN, HIGH);
      led_state = 1;
    } else {
      digitalWrite(LED_PIN, LOW);
      led_state = 0;
    }
  }
}
```

```
void loop() {
  int i;
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, i*10, 0, 0);
    delay(10);
  }
  led_check();
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, 0, i*10, 0);
    delay(10);
  }
  led_check();
  for(i = 0; i < 25; i++){
    leds.setColorRGB(0, 0, 0, i*10);
    delay(10);
  }
  led_check();
}
```

プログラム6(TouchSense_LuxSense)

- プログラム3とプログラム4の組み合わせ
- プログラム3
  - Touchセンサが押されたらLEDのON/OFFを切り替える
    - Touchセンサは押されていない状態で'0',押された状態で'1'を読み込み込める
    - Touchセンサが押されたことをOFF-ON-OFFの検知で判定する



- プログラム4
  - 光センサの値のOLEDへの出力
    - 光センサはOLED共にI2Cによりアクセス
      - ✓アクセス時間が必要であり割込みを使用する
    - 表示毎にカウンタをインクリメントして表示

プログラム6：プログラム

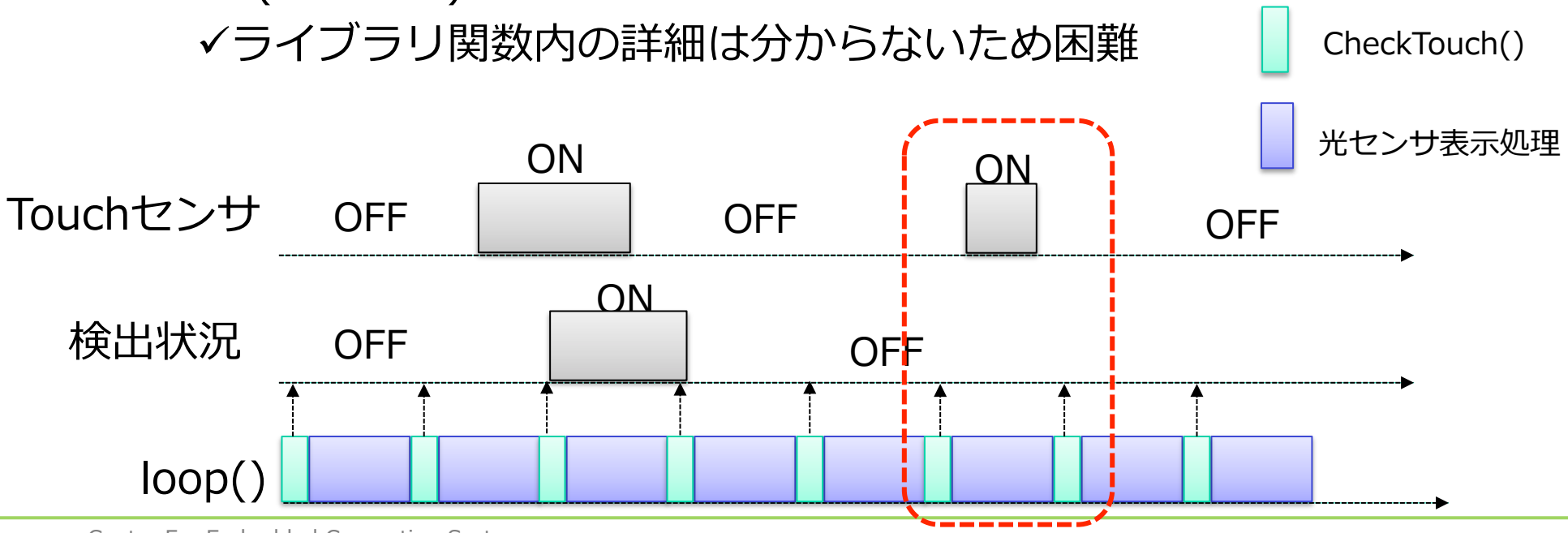
- プログラム5と同様にTouchセンサのセンシング処理を別関数として，loop()内でTouchセンサのセンシング処理を呼び出す

```
void CheckTouch() {
  int TouchValue = digitalRead(TOUCH_PIN);
  if ((PreTouchValue == 0) && (TouchValue == 1)
        && (TouchState = 0)) {
    TouchState = 1;
  }
  if ((PreTouchValue == 1) && (TouchValue == 0)
    && (TouchState = 1)) {
    TouchState = 0;
    if (LEDState == 0) {
      LEDState = 1;
      digitalWrite(LED_PIN, HIGH);
    }  else {
      LEDState = 0;
      digitalWrite(LED_PIN, LOW);
    }
  }
  PreTouchValue = TouchValue;
}
```

```
void loop() {
  int lux;
  CheckTouch();
  lux = TSL2561.readVisibleLux();
  SeeedOled.setTextXY(0, 0);
  SeeedOled.putString("Digital Light");
  SeeedOled.setTextXY(1, 0);
  SeeedOled.putString("Sensor Value is");
  SeeedOled.setTextXY(2, 0);
  SeeedOled.putNumber(lux);
  SeeedOled.putString(" lux      ");
  SeeedOled.setTextXY(4, 0);
  SeeedOled.putString("Count is");
  SeeedOled.setTextXY(5, 0);
  SeeedOled.putNumber(OLEDCount++);
}
```

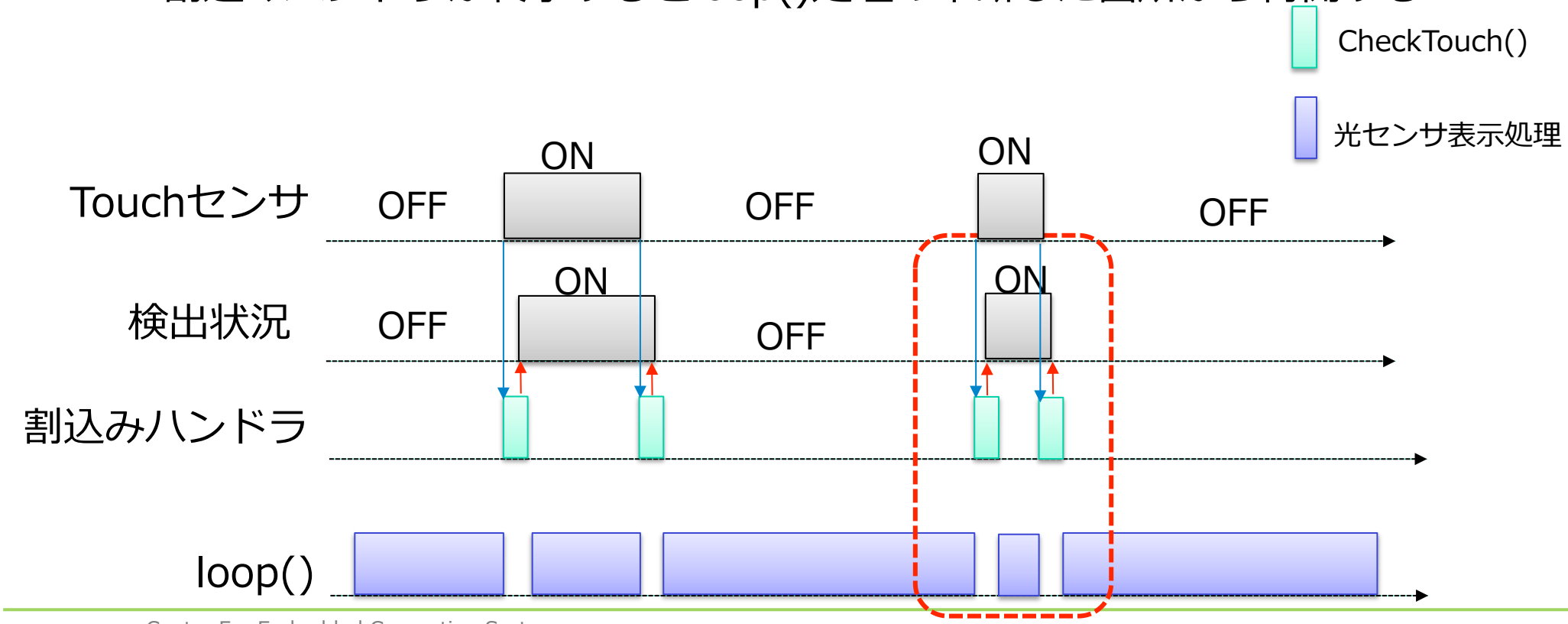
プログラム6：問題点

- TouchセンサのPUSHを検知しない場合がある
  - 光センサ表示処理に処理時間が必要なため，CheckTouch()の呼び出し間隔が長く，その間にOFF-ON-OFFがなされると検知出来ない
- 解決方法
  - CheckTouch()を呼び出す箇所を増やす
    - 単一の関数が長い場合はどうするか？
    - 性能(応答時間)の保証は可能か？
      - ✓ライブラリ関数内の詳細は分からないため困難



プログラム7(Interrupt)

- プログラム6の割込み版
  - 割込みハンドラでTouchセンサの検出を行う
  - Touchセンサの値が変化すると，loop()処理(光センサ表示処理)が中断されて，割込みハンドラ(別の関数)が実行される
  - 割込みハンドラが終了するとloop()処理の中断した箇所から再開する



プログラム7(Interrupt)：プログラム

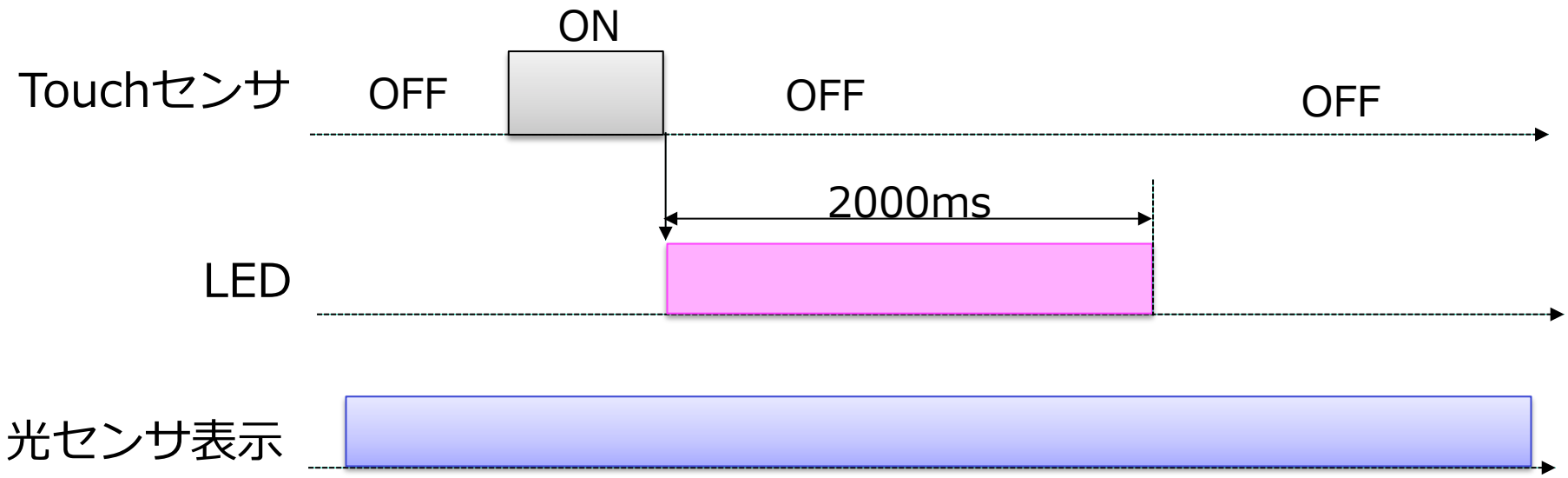
- CheckTouch()の内容は変更なし
- setup()
  - 割込みハンドラとして実行する関数を登録する
- loop()
  - CheckTouch()の呼び出しは行わない

```
void setup() {
  Wire.begin();
  TSL2561.init();
  SeeedOled.init();
  SeeedOled.deactivateScroll();
  SeeedOled.setNormalDisplay();
  SeeedOled.clearDisplay();

  pinMode(TOUCH_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  attachInterrupt(TOUCH_PIN, CheckTouch, CHANGE);
}
```

```
void loop() {
  int lux;
  lux = TSL2561.readVisibleLux();
  SeeedOled.setTextXY(0, 0);
  SeeedOled.putString("Digital Light");
  SeeedOled.setTextXY(1, 0);
  SeeedOled.putString("Sensor Value is");
  SeeedOled.setTextXY(2, 0);
  SeeedOled.putNumber(lux);
  SeeedOled.putString(" lux      ");
  SeeedOled.setTextXY(4, 0);
  SeeedOled.putString("Count is");
  SeeedOled.setTextXY(5, 0);
  SeeedOled.putNumber(OLEDCount++);
}
```

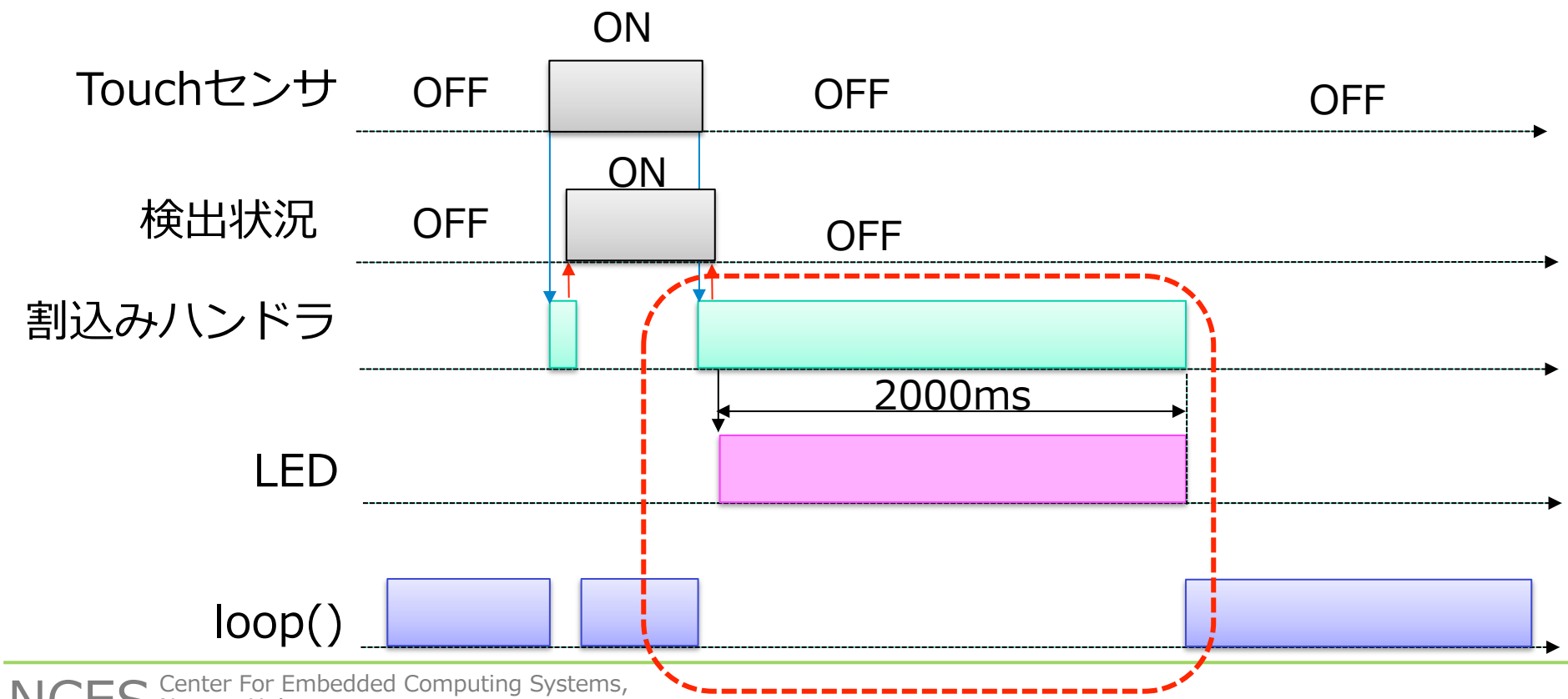
プログラム8

- プログラム7 をベースにTouchセンサでOff-On-Offを検知するとLEDを2秒間点灯する
- 光センサ表示処理も行う



プログラム8：プログラム

- 割込みハンドラではdelay()関数は使用出来ない
  - delay()関数内で割込みを使うため（割込みハンドラ実行中は割込み禁止）
  - for()で2秒程度の計時は可能だが，loop()処理が停止してしまう



プログラム8：プログラム

- 割込みハンドラで点灯しループ処理で消灯する方法
  - loop()関数は時間同期の処理でなく，ライブラリ関数の実行時間も不明なため実現は困難

```
void CheckTouch() {
  int TouchValue = digitalRead(TOUCH_PIN);
  if ((PreTouchValue == 0) && (TouchValue == 1)
      && (TouchState = 0)) {
    TouchState = 1;
  }
  if ((PreTouchValue == 1) && (TouchValue == 0)
    && (TouchState = 1)) {
    TouchState = 0;
    digitalWrite(LED_PIN, HIGH);
    for(volatile int i = 0; i < 0x400000; i++);
    digitalWrite(LED_PIN, LOW);
  }
  PreTouchValue = TouchValue;
}
```

```
void loop() {
  int lux;
  lux = TSL2561.readVisibleLux();
  SeeedOled.setTextXY(0, 0);
  SeeedOled.putString("Digital Light");
  SeeedOled.setTextXY(1, 0);
  SeeedOled.putString("Sensor Value is");
  SeeedOled.setTextXY(2, 0);
  SeeedOled.putNumber(lux);
  SeeedOled.putString(" lux      ");
  SeeedOled.setTextXY(4, 0);
  SeeedOled.putString("Count is");
  SeeedOled.setTextXY(5, 0);
  SeeedOled.putNumber(OLEDCount++);
}
```

まとめ

まとめ

- シングルタスクプログラムについて紹介

- 単一機能の実現
  - 基本的には記述可能

- 複合機能の実現
  - 記述可能な場合と不可能な場合がある
    - 可能な場合もプログラムの保守性が悪化

  - それぞれ時間に依存する処理の場合，お互いの実行タイミングを考慮したプログラミングが必要

  - 周期が短い処理と長い一連の処理を実行するのは不可能
    - 割込みハンドラによる解決
    - 割込み発生後に待ちを伴う処理の実現方法が課題