

BlueScriptハンズオン 事前準備編

講師：望月 文香

TA：田中 基暉

目次

- Windows 事前準備 (p.3 ~)
- macOS 事前準備 (p.13 ~)
- Linux 事前準備 (p.21 ~)

※ この事前準備では、Wi-Fi環境が整っていることを前提としています。

Windows 事前準備

Windows上での環境構築では、Chocolateyの使用を推奨します。

Chocolateyのインストール

- 以降のソフトウェアをインストールするために、Chocolateyをインストールします。
 - Chocolateyをインストール済み、もしくは以降のソフトウェアをChocolateyなしでインストールしたい場合は飛ばしてください。



ACTION

1. 管理者権限で、コマンドプロンプトを開いてください。
2. コマンドプロンプトで以下のコマンドを実行してください。

(<https://docs.chocolatey.org/en-us/choco/setup/#install-with-cmdexe> より)

```
$ @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "[System.Net.ServicePointManager]::SecurityProtocol = 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ choco --version
```

Python3のインストール

- Chocolateyを使用してPython3をインストールします。
 - Python3 (v3.8以上) がインストール済みの場合は飛ばしてください。



ACTION

管理者権限で開いたコマンドプロンプトで以下のコマンドを実行してください。

```
$ choco install python -y
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ python --version
```

Gitのインストール

- Chocolateyを使用してGitをインストールします。
 - インストール済みの場合は飛ばしてください。



ACTION

管理者権限で開いたコマンドプロンプトで以下のコマンドを実行してください。

```
$ choco install git -y
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ git --version
```

Node.jsとnpmのインストール

- Chocolateyを使用してNode.jsとnpmをインストールします。
 - Node.js (v20以上)がインストール済みの場合は飛ばしてください。



ACTION

管理者権限で開いたコマンドプロンプトで以下のコマンドを実行してください。

```
$ choco install nodejs --version="24.19.0" -y
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ node --version
```

Visual C++ Build Environmentのインストール

- Chocolateyを使用してVisual C++ Build Environment 2022をインストールします。



ACTION

管理者権限で開いたコマンドプロンプトで以下のコマンドを実行してください。

```
$ choco install visualstudio2022-workload-vctools -y
```



NOTE

公式サイトからインストールする場合は、必ずVisual C++ Build Environment 2022か2019をインストールするようにしてください。

また、セットアップ過程でDesktop development with C++を選択するようにしてください。

mingw32-makeのインストール

- Chocolateyを使用してmingw32-makeをインストールします。
 - mingw32-makeかmakeがインストール済みの場合は飛ばしてください。



ACTION

管理者権限で開いたコマンドプロンプトで以下のコマンドを実行してください。

```
$ choco install mingw -y
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ mingw32-make --version
```

BlueScriptのCLIをインストール

- npmを使用してBlueScriptのCLIをインストールします。



ACTION

ターミナルで以下のコマンドを実行してください。

```
$ npm install -g @bscript/cli
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ bscript -v
```

ESP32の環境をセットアップ

- BlueScriptのCLIを使用してESP32用の環境をセットアップします。



ACTION

コマンドプロンプトで以下のコマンドを実行してください。

少し時間がかかります。気長に待ってください。

```
$ bscript board setup esp32
```



CHECK

以下のコマンドでesp32の環境がセットアップされていることを確認してください。

```
$ bscript board list
Available boards:
-- esp32  - set up
```

準備完了

ここまでで、Windowsの事前準備は完了です 🎉

お疲れ様でした。

セッションでお会いできるのを楽しみにしています！

macOS 事前準備

Homebrewのインストール

- Homebrewをインストールします。
 - Homebrewがインストール済みの場合は飛ばしてください。



ACTION

ターミナルで以下のコマンドを実行してください。

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ brew --version
```

Node.jsとnpmのインストール

- Homebrewを使用してNode.jsとnpmをインストールします。
 - Node.js (v20以上)がインストール済みの場合は飛ばしてください。



ACTION

ターミナルで以下のコマンドを実行してください。

```
$ brew install node@24
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ node --version
```

makeのインストール

- makeをインストールします。
 - makeがインストール済みの場合は飛ばしてください。



ACTION

ターミナルで以下のコマンドを実行してください。

```
$ xcode-select --install
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ make --version
```

PythonとGitの確認

- Python (v3.8以上)とGitがインストールされていることを確認してください。



CHECK

1. 以下のコマンドでPython3 (v3.8以上) がインストールされていることを確認してください。

```
$ python3 --version  
Python 3.9.21
```

2. 以下のコマンドでGitがインストールされていることを確認してください。

```
$ git --version  
git version 2.39.2 (Apple Git-143)
```

BlueScriptのCLIをインストール

- npmを使用してBlueScriptのCLIをインストールします。



ACTION

ターミナルで以下のコマンドを実行してください。

```
$ npm install -g @bscript/cli
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ bscript -v
```

ESP32の環境をセットアップ

- BlueScriptのCLIを使用してESP32用の環境をセットアップします。



ACTION

ターミナルで以下のコマンドを実行してください。

少し時間がかかります。気長に待ってください。

```
$ bscript board setup esp32
```



CHECK

以下のコマンドでesp32の環境がセットアップされていることを確認してください。

```
$ bscript board list
Available boards:
-- esp32   - set up
```

準備完了

ここまでで、macOSの事前準備は完了です 🎉

お疲れ様でした。

セッションでお会いできるのを楽しみにしています！

Linux 事前準備

build-essential のインストール

- build-essential をインストールします。
 - make がインストール済みの場合は飛ばしてください。



ACTION

ターミナルで以下のコマンドを実行してください。

ディストリビューションに合わせて、適宜コマンドを変更してください。

```
$ sudo apt install build-essential
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ make --version
```

Node.jsとnpmのインストール

- nvmを使用してNode.jsとnpmをインストールします。
 - Node.js (v20以上)がインストール済みの場合は飛ばしてください。



ACTION

ターミナルで以下のコマンドを順番に実行してください。

```
# nvmをダウンロードしてインストールする
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.6/install.sh | bash
$ \. "$HOME/.nvm/nvm.sh" # シェルを再起動する代わりに実行
$ nvm install 24 # Node.jsをダウンロードしてインストールする
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ node --version
```

BlueScriptのCLIをインストール

- npmを使用してBlueScriptのCLIをインストールします。



ACTION

ターミナルで以下のコマンドを実行してください。

```
$ sudo npm install -g @bscript/cli
```



CHECK

以下のコマンドでバージョンが確認できれば成功です。

```
$ bscript -v
```

ESP32の環境をセットアップ

- BlueScriptのCLIを使用してESP32用の環境をセットアップします。



ACTION

ターミナルで以下のコマンドを実行してください。

少し時間がかかります。気長に待ってください。

```
$ bscript board setup esp32
```



CHECK

以下のコマンドでesp32の環境がセットアップされていることを確認してください。

```
$ bscript board list
Available boards:
-- esp32   - set up
```

準備完了

ここまでで、Linuxの事前準備は完了です 🎉

お疲れ様でした。

セッションでお会いできるのを楽しみにしています！

BlueScript ハンズオン！

講師：望月 文香

TA：田中 基暉

本日のスケジュール

- BlueScriptの紹介 (10分)
- ハンズオン (45分)
- Q&A (10分)
- アンケート (5分)

こんなお悩みありませんか？

- 思いついたアイデアをすぐに試したい！
- ハードウェアの動きを見ながらパラメータを調整したい！
- 高速・省メモリも諦めたくない！

そんな時は、 **BlueScript** !

マイコン開発を、もっと身近に、もっと自由に

書きやすい

- TypeScript風の構文
- cも埋め込める

試しやすい

- 対話的な開発環境
- CLIで全て完結
- Bluetooth経由で開発可能

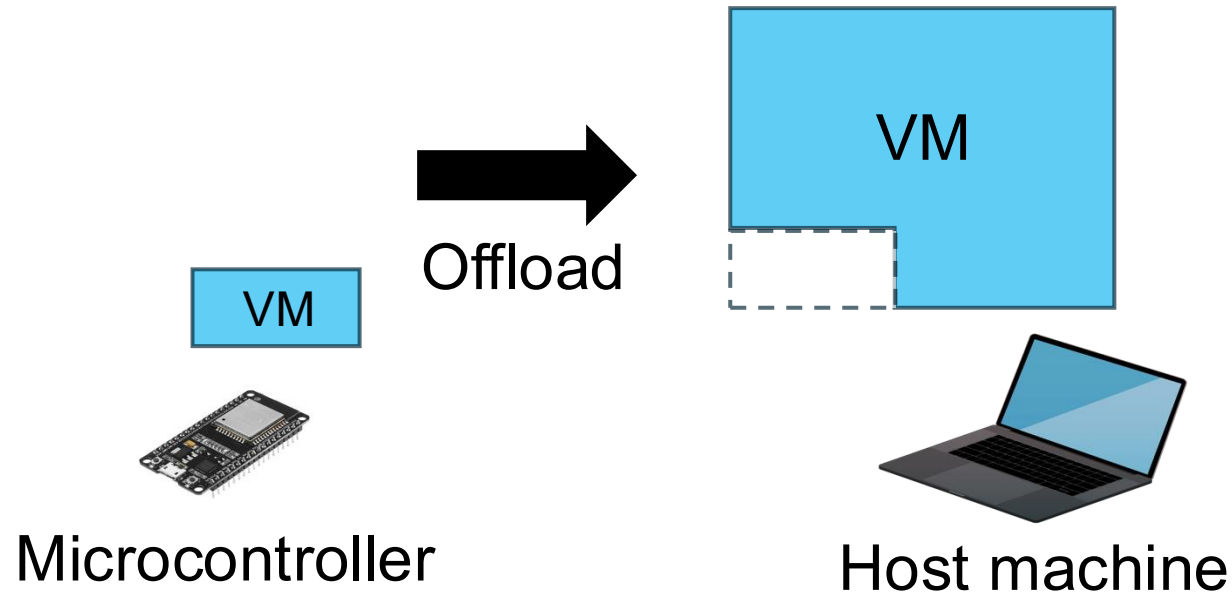
速い

- 少ないメモリ消費量
- MicroPythonに比べて20倍の実行速度 [1]

[1] ベンチマーク S. Marr, DLS'16 における比較

省メモリ・高速実行とリッチな開発環境を両立する仕組み

- BlueScriptは非集約型仮想マシンというVMのモデルを採用し、**できるだけ多くのVMコンポーネントをホストマシンにオフロード**しています。
 - 非集約型仮想マシンとは、複数マシンにVMのコンポーネントが配置されるVMモデルです。



それでは、ハンズオンを
始めていきましょう

今日作るもの

- LEDを青色に点滅させます。



グループを作しましょう



ACTION

- 隣の人と2～3人のグループを作成してください。
 - できれば、環境構築ができている人が各グループにいるようにしてください。
 - グループに一つ、M5Stack Fireとケーブルを配ります。



CHECK

1. グループはできましたか？軽く自己紹介をしましょう。
2. 本日必要なものは以下になります。全て揃っていることを確認してください。
 - Windows / macOS / Linux のいずれかが動作するPC
 - M5Stack Fire
 - PCとM5Stack Fireを繋ぐデータ書き込み用ケーブル

Step 0: 事前準備

事前準備スライドを参考に、BlueScriptのCLIをインストールし、esp32の環境をセットアップしてください。



CHECK

1. 以下のコマンドでBlueScriptのCLIがインストールされていることを確認してください。

```
$ bscript --version  
2.1.0
```

2. 以下のコマンドでesp32の環境がセットアップされていることを確認してください。

```
$ bscript board list  
Available boards:  
-- esp32 - set up
```

Step 1: ランタイムの書き込み

BlueScriptのランタイムをM5Stack Fireに書き込みます。



ACTION

1. M5Stack FireとPCをケーブルで繋がします。



ケーブル
を挿す

2. 以下のコマンドを実行してください。

<device-name> はM5Stack Fireの箱に付箋で書かれているものを使用してください。(例：B01)

```
$ bscript board flash-runtime esp32 -d "<device-name>"
```



NOTE

- ポートを選択するように促されます。(1a86:55d4) が末尾についているものを選択してください。
- 適切なポートが現れなければリセットボタンを2回押してみてください。

ここからはBluetooth経由で開発します。

以降のプログラムの変更や実行はBluetooth経由で行います。



NOTE

USBケーブルを外して開発が出来ます。
ただし、充電が少ない場合はM5Stack FireをPCまたは電源に接続してください。

Step 2: プロジェクトの作成

BlueScriptのプロジェクトを作成します。



ACTION

以下のコマンドを実行してください。

```
$ bscript project create hands-on
```



CHECK

ディレクトリが作成されていれば成功です。
プロジェクトは以下のような構成になっています。

```
.
├── bsconfig.json  #プロジェクトの設定ファイル
└── src
    └── index.bs   #エントリファイル
```

Step 3: Hello, BlueScript!

最初のプログラムを実行してみましょう。



ACTION

1. プロジェクトのディレクトリに移動します

```
$ cd hands-on
```

2. プロジェクトを好きなエディタで開いて、src/index.bsに以下が書かれていることを確認してください。

```
console.log("Hello world!");
```

3. 以下のコマンドでプログラムを実行します。

<device-name> はランタイムの書き込みに使用したのと同じものです。(例 : B01)

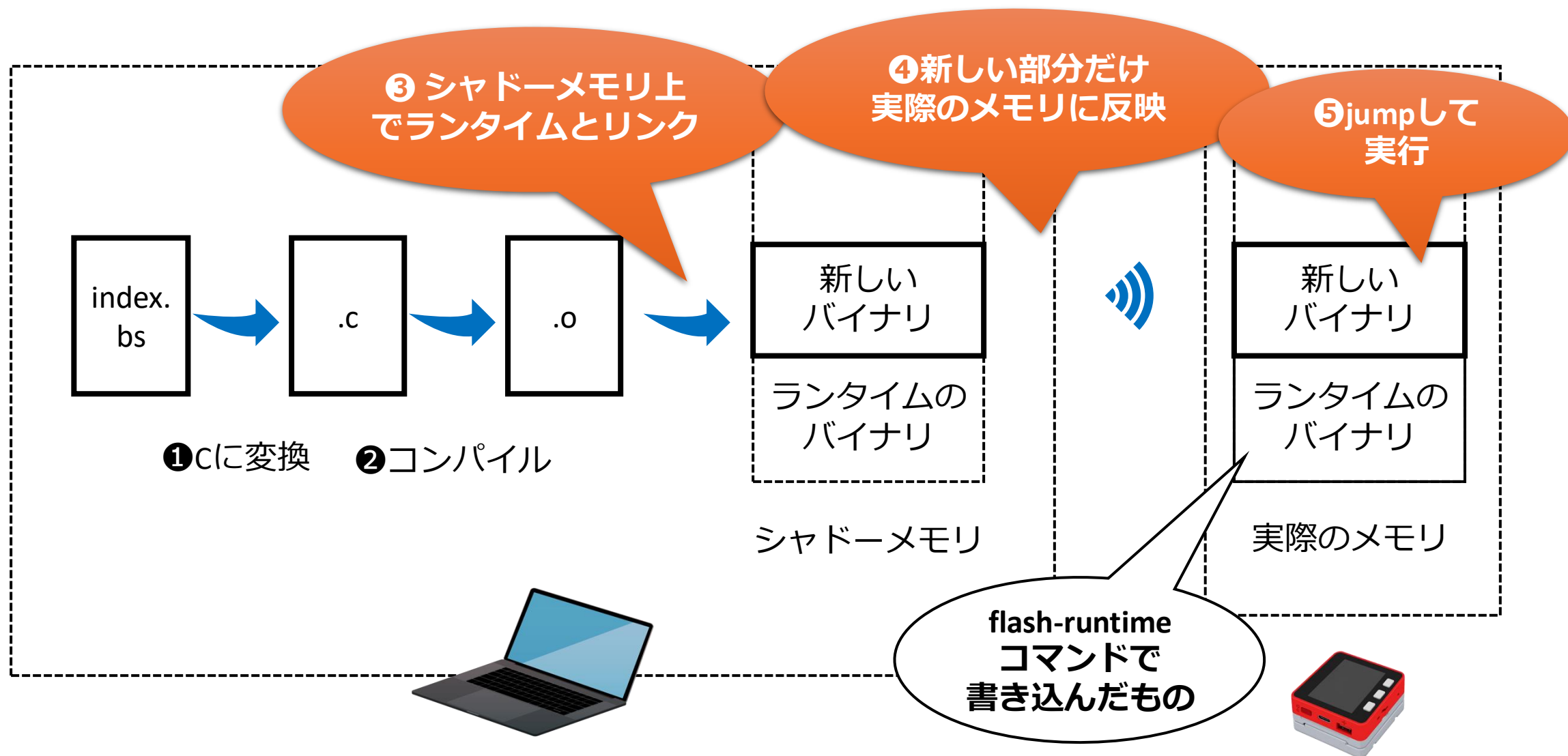
```
$ bscript project run -d "<device-name>"
```



CHECK

ターミナルに"Hello world!"が表示されれば成功です。

runコマンドで何が起きたのか？



LEDを光らせる

Step 4: LEDパッケージのインストール

はじめに、LEDパッケージをインストールします。



ACTION

以下のコマンドを実行してください。

```
$ bscript project install https://github.com/bluescript-lang/pkg-neopixel-esp32.git
```



NOTE

依存ライブラリも自動でインストールされます。

Step 5: LEDを光らせるプログラムを書く



ACTION

以下のプログラムをsrc/index.bsに書き込んでください。

```
import { NeoPixel, Colors } from "neopixel"; // パッケージをインポート

const strip = new NeoPixel(15, 10);           // LEDを初期化
strip.setBrightness(50);                      // 明るさをセット

function setBlue() { // 全てのLEDを青色にする関数
  for (let i = 0; i < 10; i++) {
    strip.setPixelColorHex(i, Colors.Blue);
  }
}

for (let i = 0; i < 5; i++) { // LEDを5回点滅させる
  console.log("LED off");
  strip.clear();              // 全てのLEDを消す
  strip.show();               // 操作を反映
  time.delay(500);            // 500 ms 待つ
  console.log("LED on");
  setBlue();                  // 全てのLEDを青色にする
  strip.show();               // 操作を反映
  time.delay(500);
}
```

Step 6:実行！

プログラムを実行してみましょう。



ACTION

以下のコマンドを実行してください。

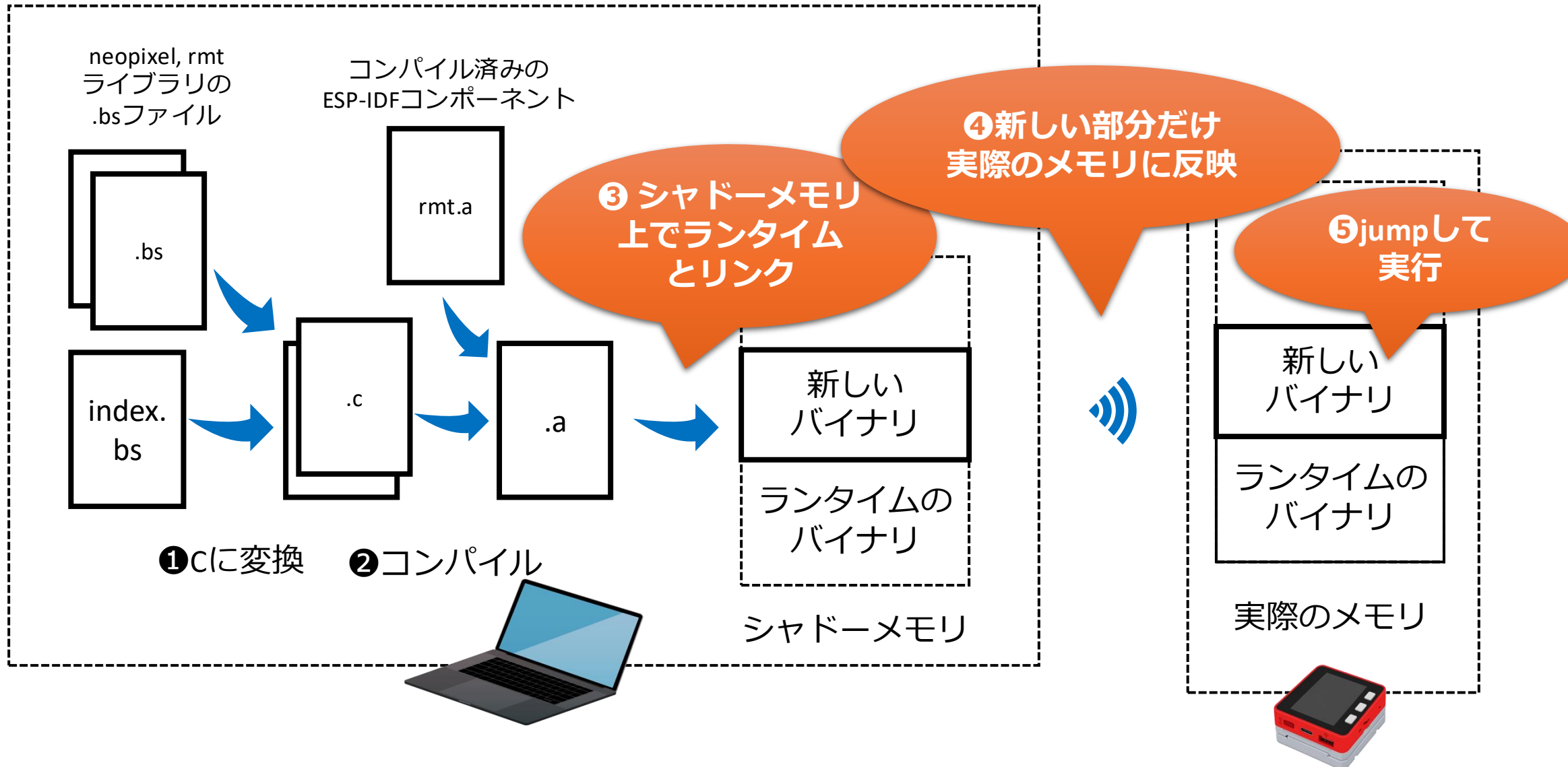
```
$ bscript project run -d "<device-name>"
```



CHECK

M5Stack FireのLEDが青く点滅したら成功です。

何が起こったのか？



対話環境を使ってみる

BlueScriptの対話環境

BlueScriptはブラウザ上のNotebook環境から対話的にプログラムを追加することができます。

The screenshot displays the BlueScript interface, which is a web-based notebook environment. It features a dark header bar with the text "BlueScript". The main area contains three code cells, each with a copy icon on the left. The first cell contains the code `console.log("LED on"); led.write(PinLevel.High);` and shows execution times of 355.28 ms for compile, 85.81 ms for loading, and 120.73 ms for execution. The second cell contains `console.log("LED off"); led.write(PinLevel.Low);` with 300.36 ms for compile, 52.37 ms for loading, and 61.32 ms for execution. The third cell contains a function definition for `blink` and shows 302.79 ms for compile, 47.55 ms for loading, and 61.63 ms for execution. Below the code cells is a final cell with `blink(5);`. On the right side, there is a log area showing "LED on" and "LED off".

BlueScript

プログラム

実行ボタン

ログ

実行時間

```
console.log("LED on");
led.write(PinLevel.High);
| compile: 355.28 ms | loading: 85.81 ms | execution: 120.73 ms |

console.log("LED off");
led.write(PinLevel.Low);
| compile: 300.36 ms | loading: 52.37 ms | execution: 61.32 ms |

function blink(n: integer) {
  for (let i = 0; i < n; i++) {
    led.write(PinLevel.High);
    time.delay(1000);
    led.write(PinLevel.Low);
    time.delay(1000);
  }
}
| compile: 302.79 ms | loading: 47.55 ms | execution: 61.63 ms |

blink(5);
```

LED on
LED off

Step 7: マイコンをリセット



ACTION

電源につながっている場合は一度電源を抜き、リセットボタンを素早く2回押して、マイコンの電源を切ってください。その後、もう一度リセットボタンを押して電源を入れてください。

リセットボタン



CHECK

LEDが消えていることを確認してください。

Step 8: 対話環境を呼び出す

対話環境を呼び出してみましょう。



ACTION

以下のコマンドを実行してください。

```
$ bscript project run -d "<device-name>" --with-notebook
```



CHECK

プログラムの実行後、対話環境がブラウザで開かれれば成功です。
開かなければ、ブラウザで <http://localhost:3000> にアクセスしてみてください。

Step 9: 対話環境でプログラムを実行

対話環境でプログラムを書いて実行してみましょう。



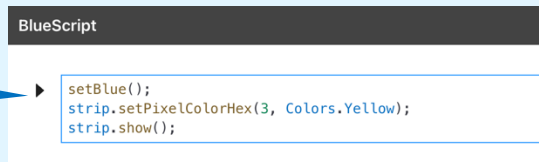
ACTION

1. 以下のプログラムを最初のセルに書き込んでください。

```
strip.setPixelColorHex(3, Colors.Yellow);  
strip.show();
```

2. セルの横の実行ボタンを押して実行してください。

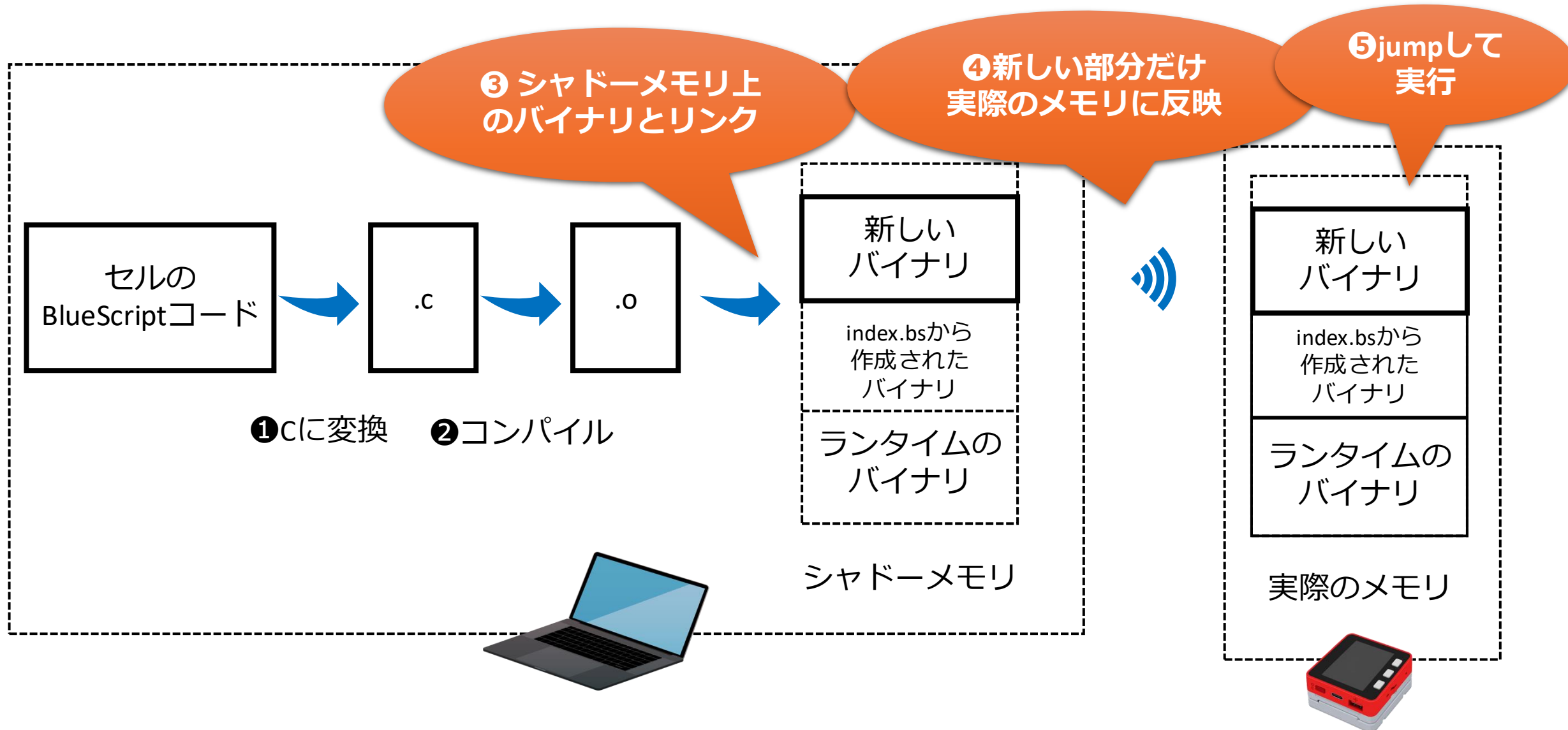
Click!



CHECK

4番目のLEDが黄色に、それ以外が青に光れば成功です。

何が起こったのか？



発展課題

時間が余りましたか？

対話環境を使って、試しにLEDを虹色に光らせてみましょう。

BlueScriptとLEDライブラリの仕様はこちらです。

- BlueScript : <https://csg-tokyo.github.io/bluescript/docs/reference/language/intro>
- LEDライブラリ : <https://github.com/bluescript-lang/pkg-neopixel-esp32>

まとめ

他にもこんなことができます

- 割り込み処理を記述する
- BlueScriptのプログラムにCを埋め込む
- ホストマシン上でBlueScriptを実行する

詳しくは、Webサイトをご覧ください！



<https://csg-tokyo.github.io/bluescript/>

BlueScript仲間を募集中です！

BlueScriptと一緒に育ててくれる方を探しています。

使ってみる

手元のマイコンで動かして、感想や困ったことを教えてください。

作ってみる

新しいボードへの対応やライブラリ、サンプルを追加してみませんか？

広めてみる

作品・ブログなどでBlueScriptを紹介してください。

まずは「使ってみた」だけでも大歓迎です！

Q & A

質問を考えている間に、BlueScriptを応援していただけると嬉しいです。

★GitHubでスター



<https://github.com/csg-tokyo/bluescript>

📢 Xをフォロー



<https://x.com/BlueScriptLang>

ご参加ありがとうございました！

- アンケートへの回答をお願いいたします。



<https://forms.gle/Zo3LHrGJkix2x2bQA>