



Using μ T-Kernel 3.0 on micro:bit

June 16, 2026

Personal Media Corporation

Agenda

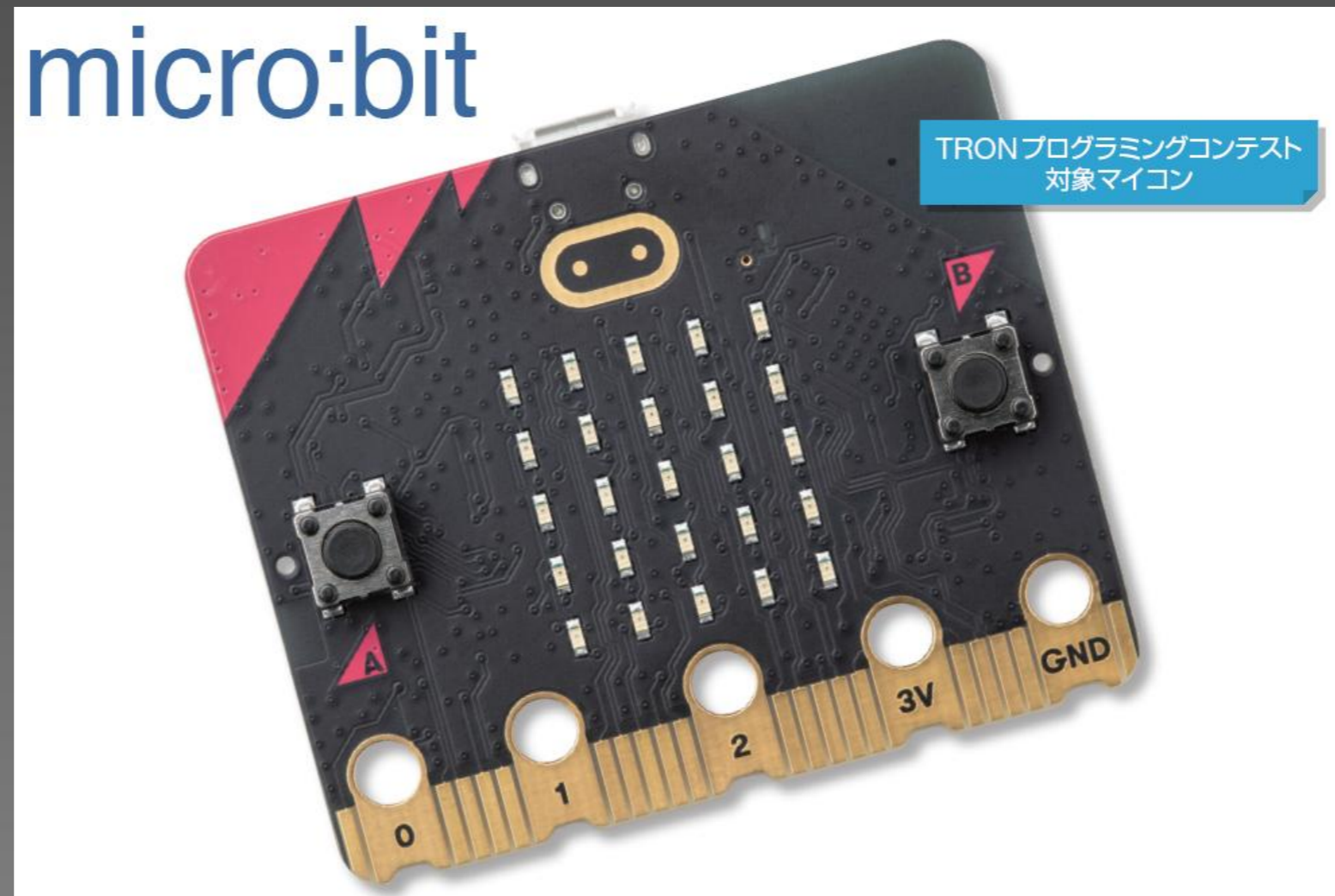
- Introduction to micro:bit
- Running μ T-Kernel 3.0 on micro:bit
- Development Environment
- Software Development with Eclipse
- Using Peripheral Devices
- Tips for contest entry and AI utilization
- Q & A



Introduction to micro:bit

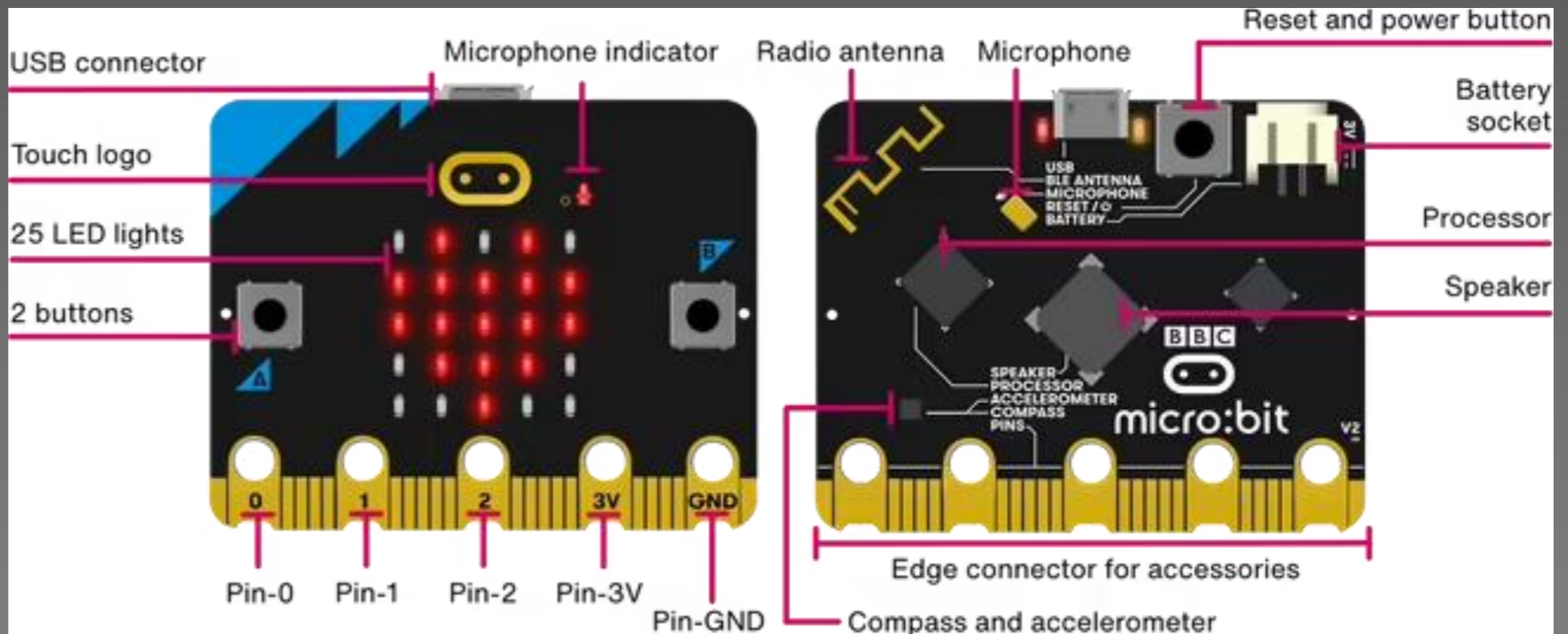
Introduction to micro:bit

- Mini-sized educational board computer, developed by BBC
- CPU : Nordic nRF52833 (Arm Cortex-M4 core)



Introduction to micro:bit

- Many peripheral devices on micro:bit
 - Display : 5 * 5 matrix LED / single LED
 - Sensors : acceleration / illuminance / touch / temperature
 - Sound : speaker / microphone
 - Communication : USB / UART (serial) / BLE (Bluetooth Low Energy)
 - Push button switches
 - Connector pins



Introduction to micro:bit

- Standard software development is by Microsoft MakeCode





Running μ T-Kernel 3.0 on micro:bit

Running μ T-Kernel 3.0 on micro:bit

- Choice 1: “IoT edge node Practice Kit for micro:bit”
- Choice 2: Articles on TRONWARE magazine
- Choice 3: Same articles available on website
 - Now available on Personal Media Corporation website
 - <https://www.t-engine4u.com/info/mbit/index.html>
- Same RTOS and Cross Development Environment
 - RTOS: μ T-Kernel 3.0 of TRON Forum, ported to micro:bit
 - Cross Development Environment: Eclipse

Running μ T-Kernel 3.0 on micro:bit

- Choice 1: "IoT edge node Practice Kit for micro:bit"
http://www.t-engine4u.com/products/ioten_prackit.html

PERSONAL MEDIA CORP. サイトマップ お問い合わせ 検索

www.t-engine4u.com Home ソリューション 製品 セミナー 情報/書籍 サポート English

Home >> 製品一覧 >> T-Kernel用教材セット >> IoTエッジノード実践キット/micro:bit

micro:bitを使ってリアルタイムOSの動作を学ぶ学習キット

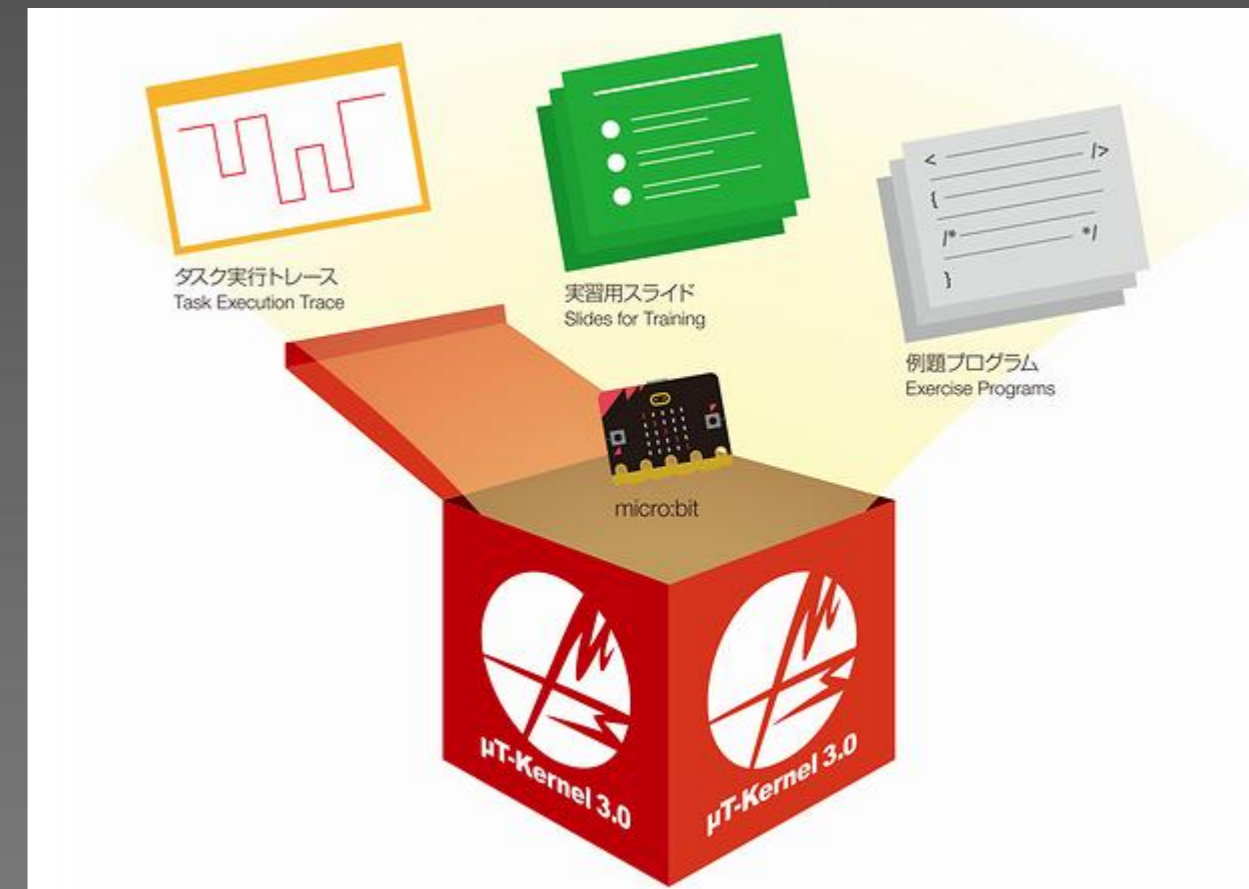
IoTエッジノード実践キット micro:bit

タスクのスケジューリングとは?

セマフォによる排他制御とは?

製品一覧

- T-Kernel(リアルタイムOS)
- T-Kernel用デバイスドライバ
- T-Kernel用ミドルウェア・アプリケーション
- T-Kernel用開発ツール
- T-Kernel用教材セット
- IoTエッジノード実践キット/micro:bit**
- IoT-Engine教育&実習パッケージ
- μ T-Kernel 3.0教育&実習パッケージ
- T-Kernel搭載 組込み向け/教育向けボード



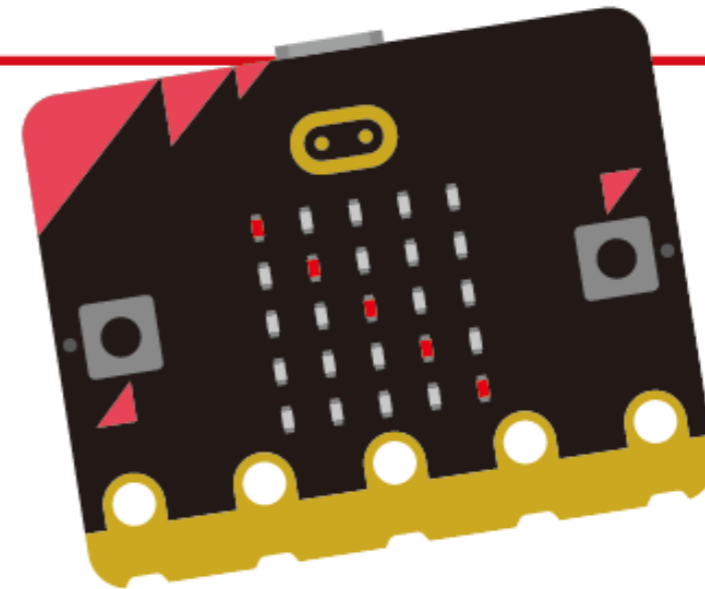
Running μ T-Kernel 3.0 on micro:bit

- Choice 2: articles on TRONWARE magazine

連載

 micro:bit で μ T-Kernel 3.0 を動かそう

micro:bit で μ T-Kernel 3.0 を動かそう



[第8回] ドレミファ音階の再生とティック時間

トロンフォーラム T3 WG

本連載では、小学生向けのプログラミング教育などに使われているBBC micro:bit (以下「micro:bit」)の上で動くようになった μ T-Kernel 3.0をご紹介している。連載第8回の本号では、micro:bitから音を出してみよう。 μ T-Kernel 3.0の持つ時間関連の機能をいろいろ試しながら、ドレミファの音階を再生するところまで進めていく。

Running μ T-Kernel 3.0 on micro:bit

Choice 3: same articles available on website

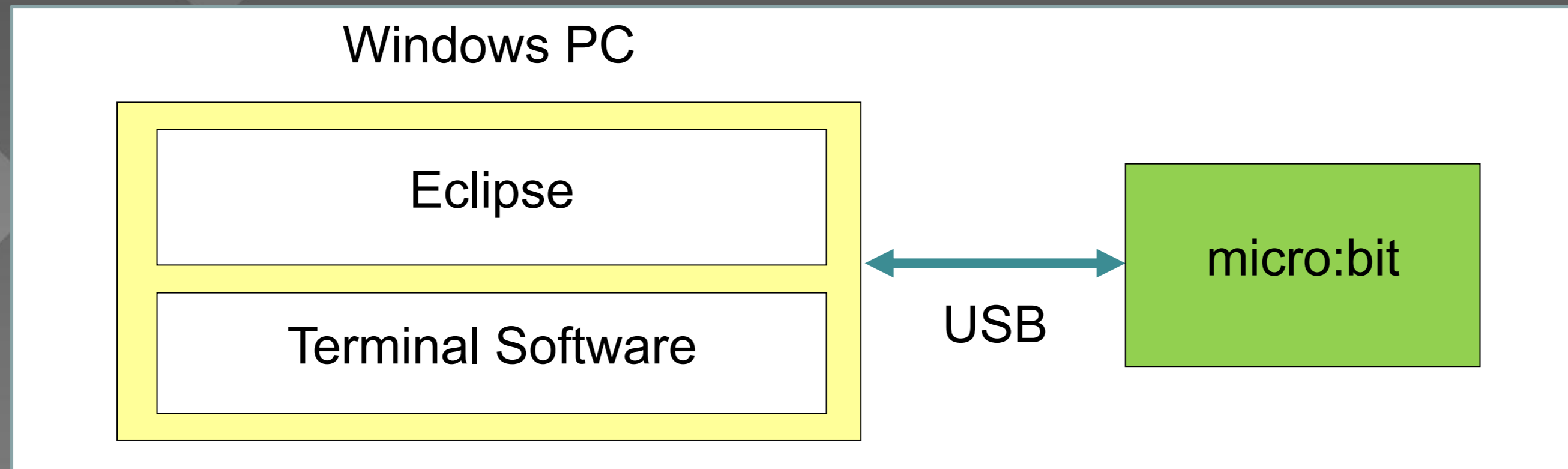




Development Environment for μ T-Kernel 3.0 on micro:bit

Development environment for μ T-Kernel 3.0

- Cross development by Eclipse
 - Edit, compile, build, run and debug your software
 - Character I/O on terminal software via USB serial



Development environment for μ T-Kernel 3.0

- C language programming for developing your applications
 - MakeCode is not available
- Programming for peripheral devices
 - Make device drivers, or access hardware directly by application
 - Articles on TRONWARE cover the following peripheral devices
 - ✓ LED matrix, push button switches
 - ✓ Speaker, PWM (=Pulse Width Modulation), A/D conversion
 - ✓ I2C, acceleration sensor
 - ✓ Serial communication between two micro:bit boards



Software Development with Eclipse

Software Development with Eclipse

- [Step-1] Installing development tools for μ T-Kernel 3.0
 - (1) Install Eclipse
 - (2) Install GNU Arm Embedded Toolchain
 - (3) Install xPack Windows Build Tools
 - (4) Install Python and pyOCD
- [Step-2] Import μ T-Kernel 3.0 and compile
 - (5) Get and expand source code of μ T-Kernel 3.0 for micro:bit
 - (6) Make a project of μ T-Kernel 3.0 on Eclipse
 - (7) Compile and build on Eclipse

Software Development with Eclipse

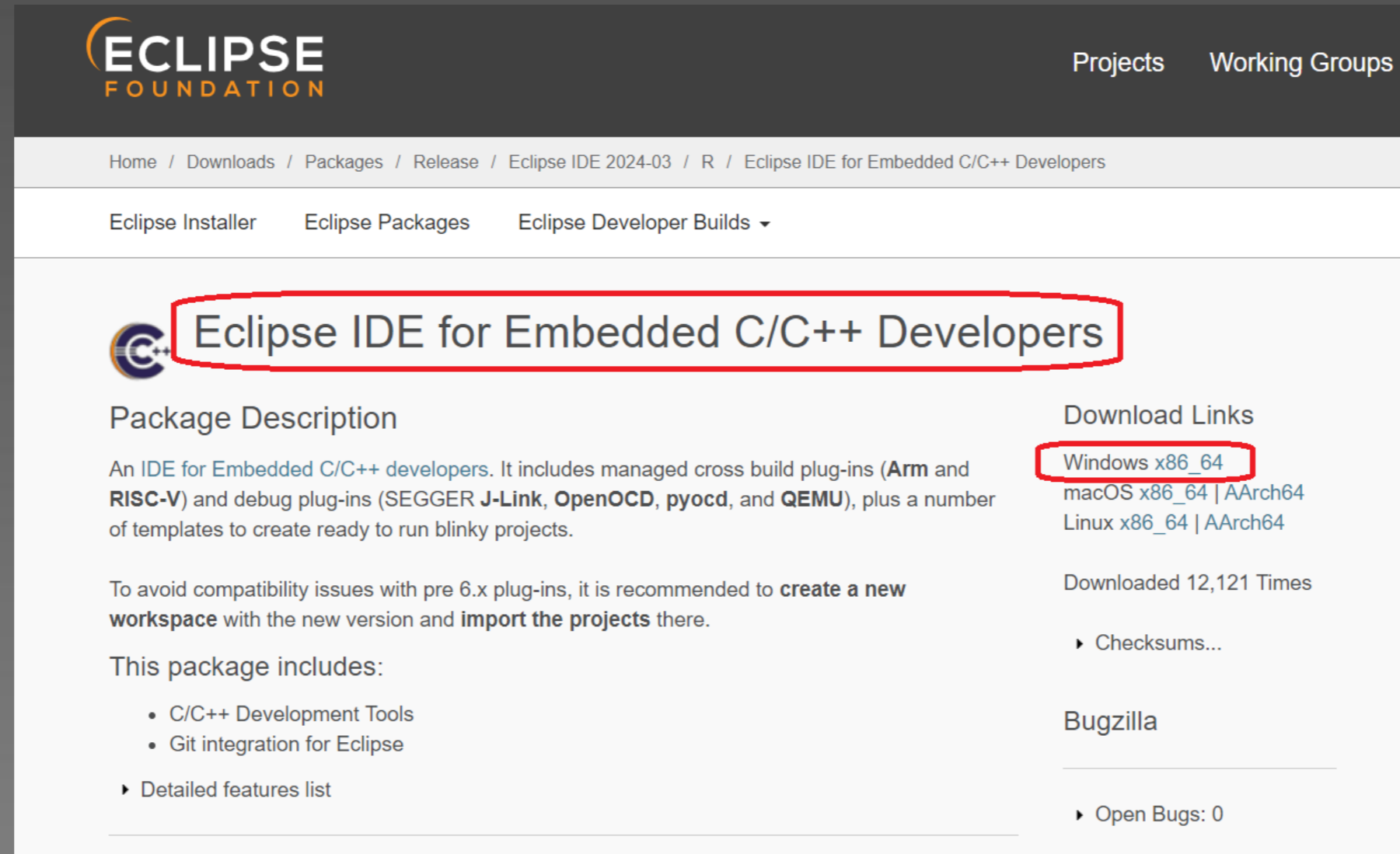
■ Development tools

Tool	Description	URL
Eclipse IDE for Embedded C/C++ Developers	Integrated development environment to edit source code, compile, debug etc.	https://www.eclipse.org/downloads/packages/release/2024-03/r/eclipse-ide-embedded-cc-developers
GNU Arm Embedded Toolchain	GNU C compiler and related tools for embedded Arm processors	https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads
xPack Windows Build Tools	Build tools running on Windows	https://github.com/xpack-dev-tools/windows-build-tools-xpack/releases
Python	Interpreter for programming language Python	https://www.python.org/downloads/
pyOCD	Debug tool for Arm processors running on Python	(Install in Python)
μT-Kernel 3.0 for micro:bit	Source code package (ZIP archive with password)	https://www.personal-media.co.jp/book/tw/tw_index/362.html

Application Development with Eclipse

■ (1) Install Eclipse

- Integrated development environment to edit source code, compile, debug etc.



The screenshot shows the Eclipse Foundation website page for "Eclipse IDE for Embedded C/C++ Developers". The page title is "Eclipse IDE for Embedded C/C++ Developers", which is circled in red. The page includes a "Package Description" section, a "Download Links" section, and a "Bugzilla" section. The "Download Links" section lists "Windows x86_64", "macOS x86_64 | AArch64", and "Linux x86_64 | AArch64", with "Windows x86_64" circled in red. The "Bugzilla" section shows "Open Bugs: 0".

ECLIPSE FOUNDATION Projects Working Groups

Home / Downloads / Packages / Release / Eclipse IDE 2024-03 / R / Eclipse IDE for Embedded C/C++ Developers

Eclipse Installer Eclipse Packages Eclipse Developer Builds ▾

Eclipse IDE for Embedded C/C++ Developers

Package Description

An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (**Arm** and **RISC-V**) and debug plug-ins (SEgger **J-Link**, **OpenOCD**, **pyocd**, and **QEMU**), plus a number of templates to create ready to run blinky projects.

To avoid compatibility issues with pre 6.x plug-ins, it is recommended to **create a new workspace** with the new version and **import the projects** there.

This package includes:

- C/C++ Development Tools
- Git integration for Eclipse

▶ Detailed features list

Download Links

- Windows x86_64
- macOS x86_64 | AArch64
- Linux x86_64 | AArch64

Downloaded 12,121 Times

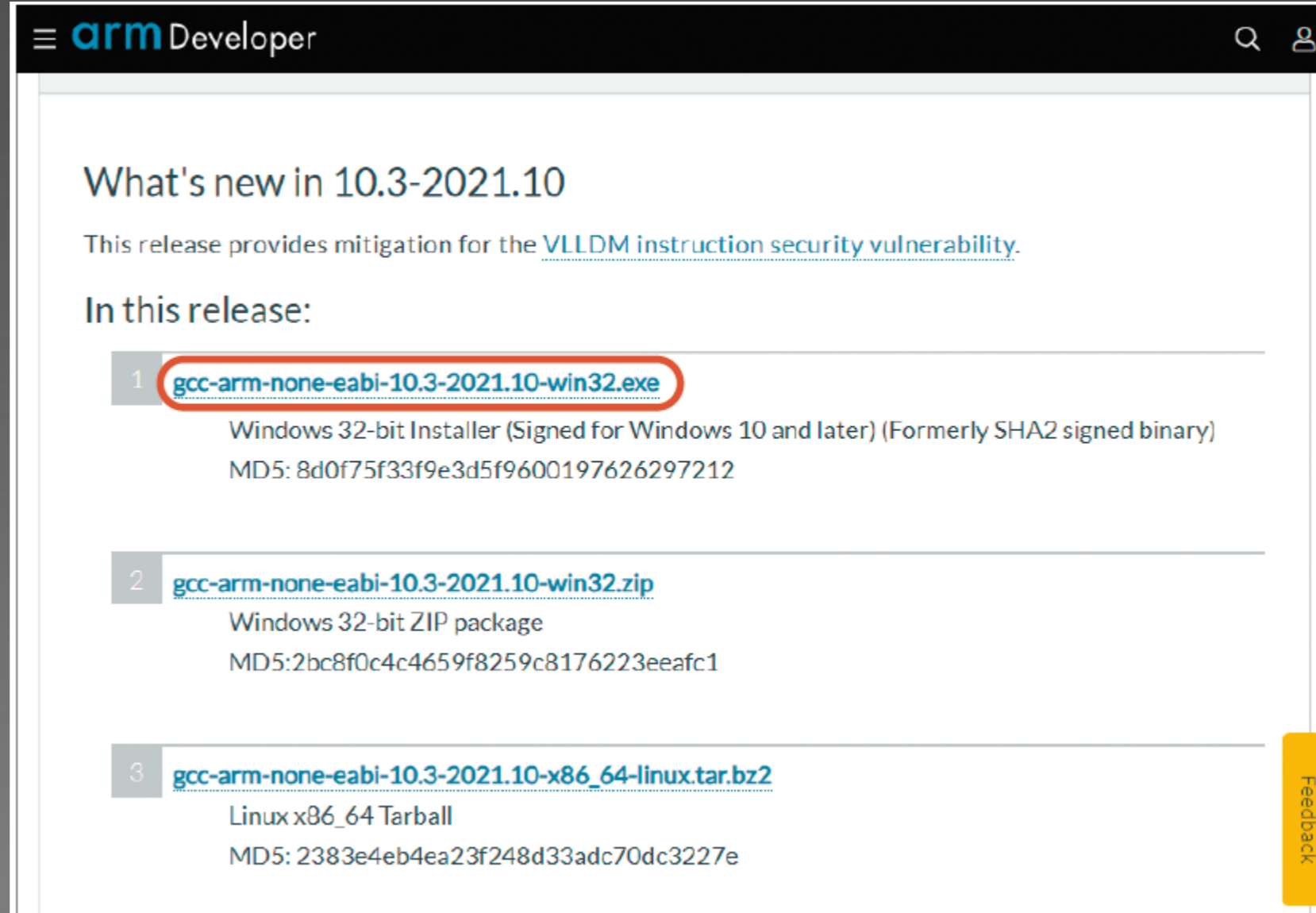
▶ Checksums...

Bugzilla

▶ Open Bugs: 0

Software Development with Eclipse

- (2) Install GNU Arm Embedded Toolchain
 - GNU C compiler and related tools for embedded Arm processors



arm Developer

What's new in 10.3-2021.10

This release provides mitigation for the [VLLDM instruction security vulnerability](#).

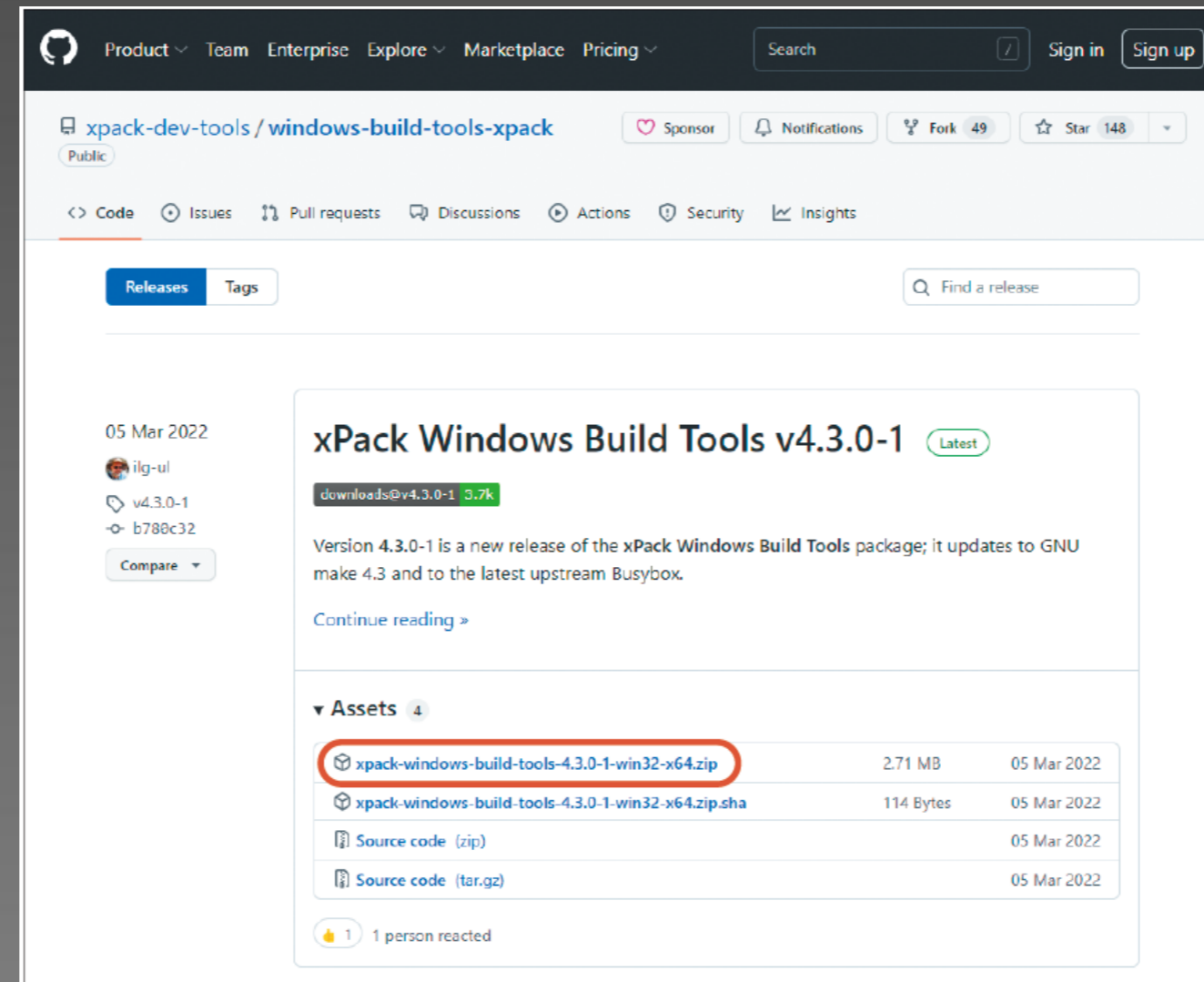
In this release:

- [gcc-arm-none-eabi-10.3-2021.10-win32.exe](#)
Windows 32-bit Installer (Signed for Windows 10 and later) (Formerly SHA2 signed binary)
MD5: 8d0f75f33f9e3d5f9600197626297212
- [gcc-arm-none-eabi-10.3-2021.10-win32.zip](#)
Windows 32-bit ZIP package
MD5: 2bc8f0c4c4659f8259c8176223eeafc1
- [gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2](#)
Linux x86_64 Tarball
MD5: 2383e4eb4ea23f248d33adc70dc3227e

Feedback

Software Development with Eclipse

- (3) Install xPack Windows Build Tools
 - Build tools running on Windows



The screenshot shows the GitHub repository page for `xpack-dev-tools/windows-build-tools-xpack`. The page is in the "Releases" tab, displaying the latest release, `xPack Windows Build Tools v4.3.0-1`, which was published on 05 Mar 2022. The release description states: "Version 4.3.0-1 is a new release of the xPack Windows Build Tools package; it updates to GNU make 4.3 and to the latest upstream Busybox." The "Assets" section lists four files: `xpack-windows-build-tools-4.3.0-1-win32-x64.zip` (2.71 MB), `xpack-windows-build-tools-4.3.0-1-win32-x64.zip.sha` (114 Bytes), `Source code (zip)`, and `Source code (tar.gz)`. The `xpack-windows-build-tools-4.3.0-1-win32-x64.zip` asset is highlighted with a red circle.

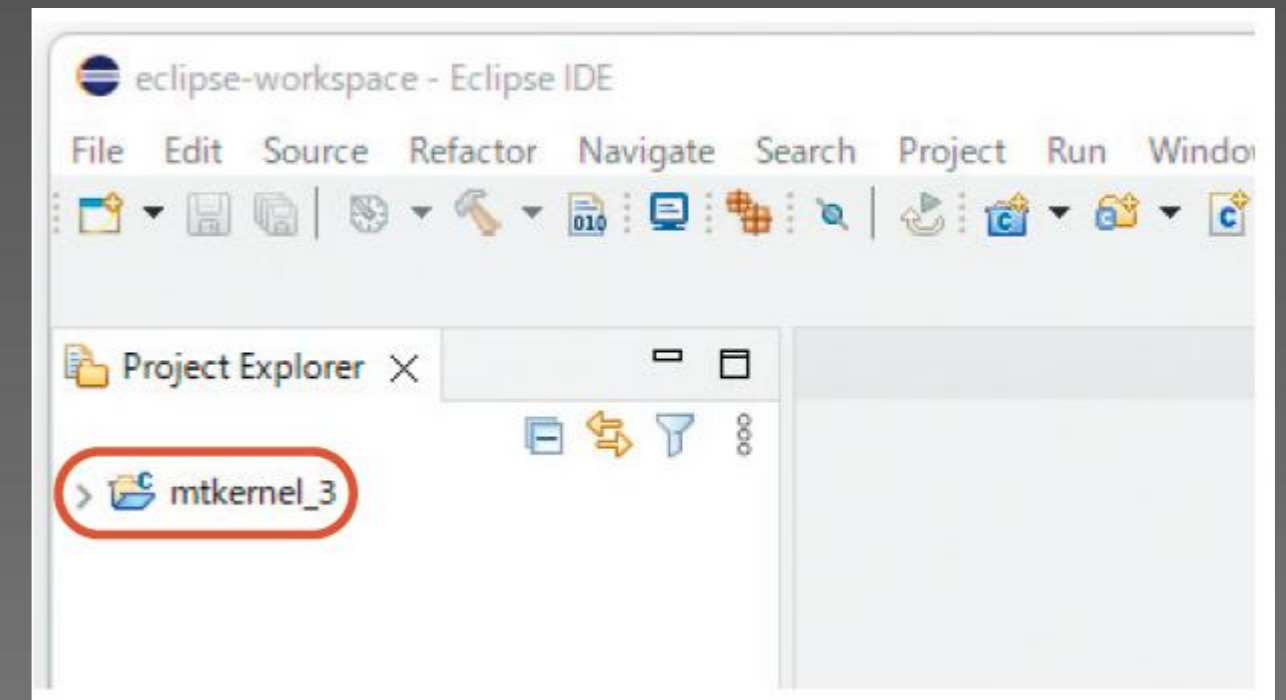
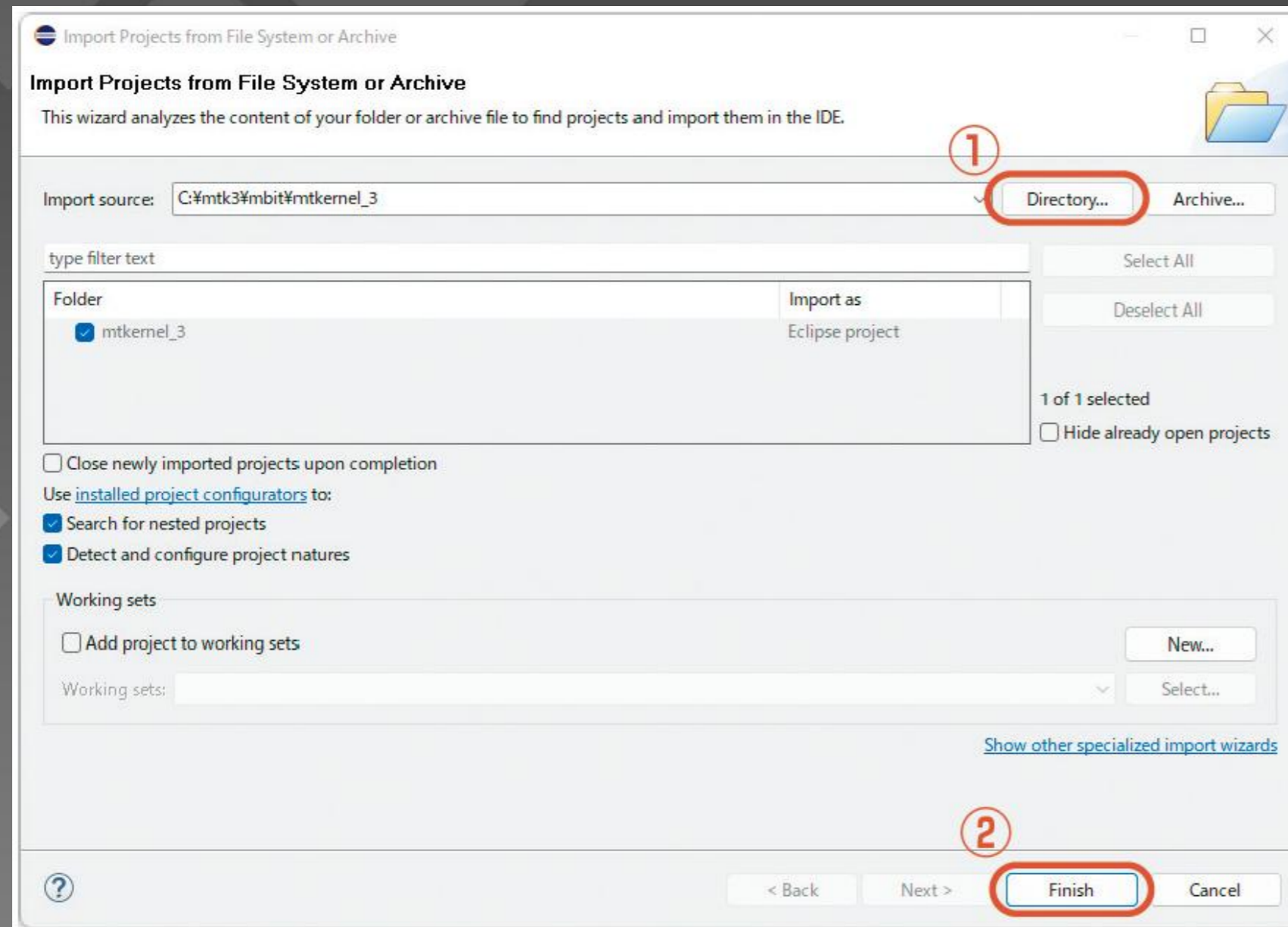
Software Development with Eclipse

- (4) Install Python and pyOCD
 - Debug tool for ARM processors running on Python
 - “python -mpip install -U pyocd” after installing Python



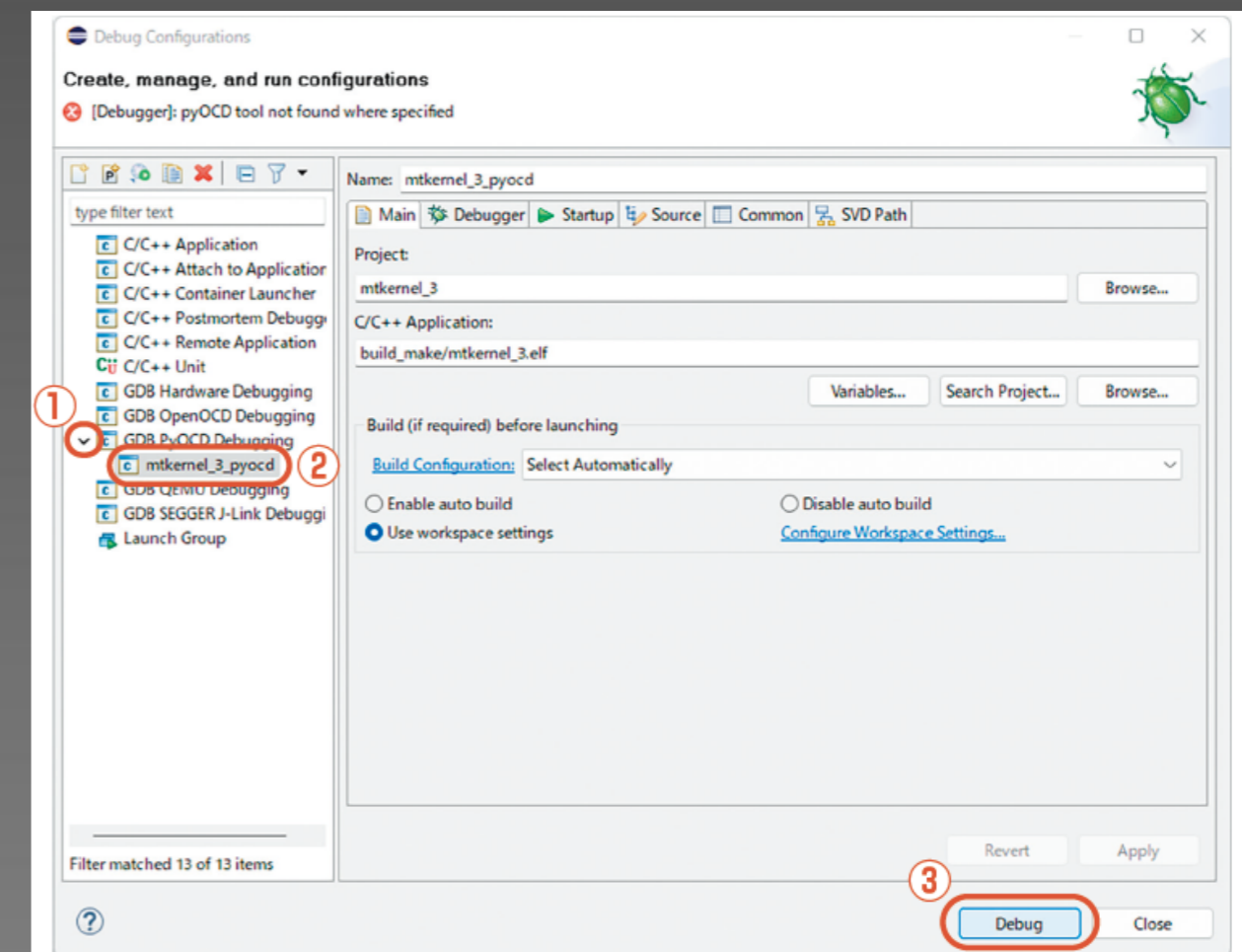
Software Development with Eclipse

- [Step-2] Import μ T-Kernel 3.0 source code and compile



Software Development with Eclipse

- [Step-3] transfer μ T-Kernel 3.0 to micro:bit and run on it
 - (8) Erase Flash ROM of micro:bit
 - (9) Start terminal software and connect it to micro:bit
 - (10) Eclipse configuration for debug
 - (11) Running on micro:bit

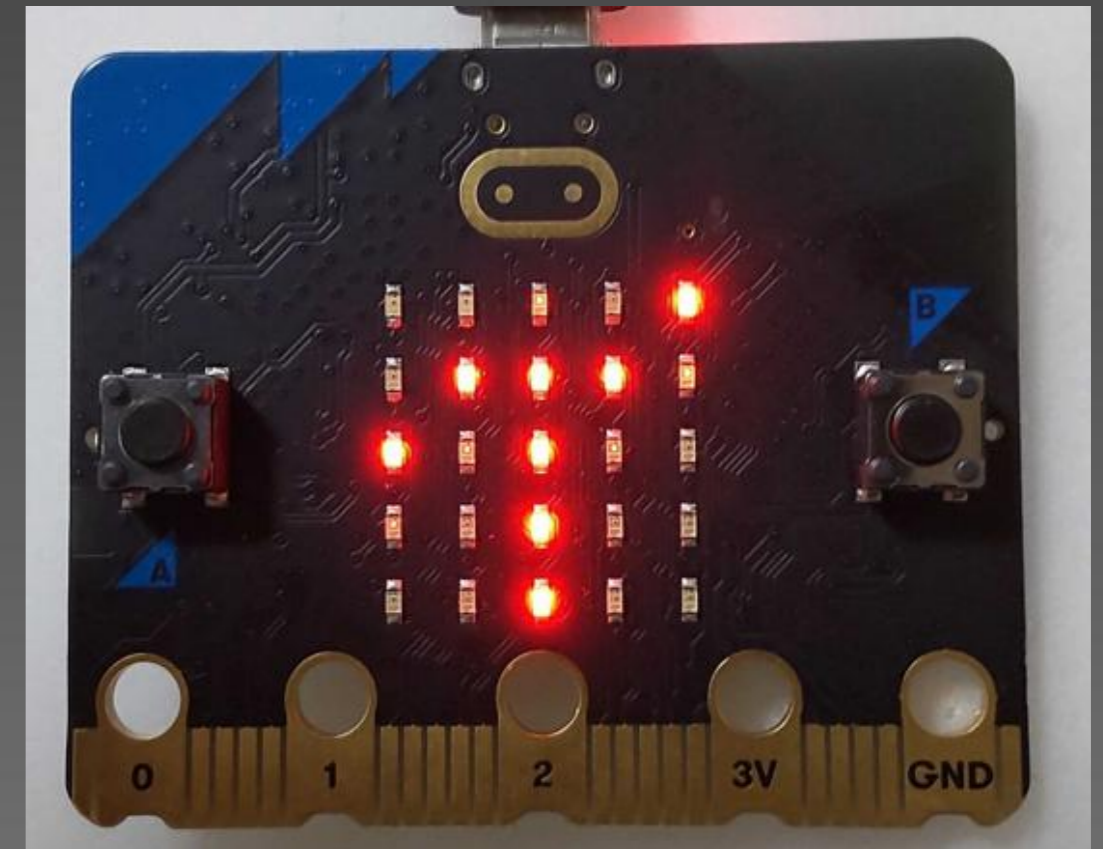




Using Peripheral Devices

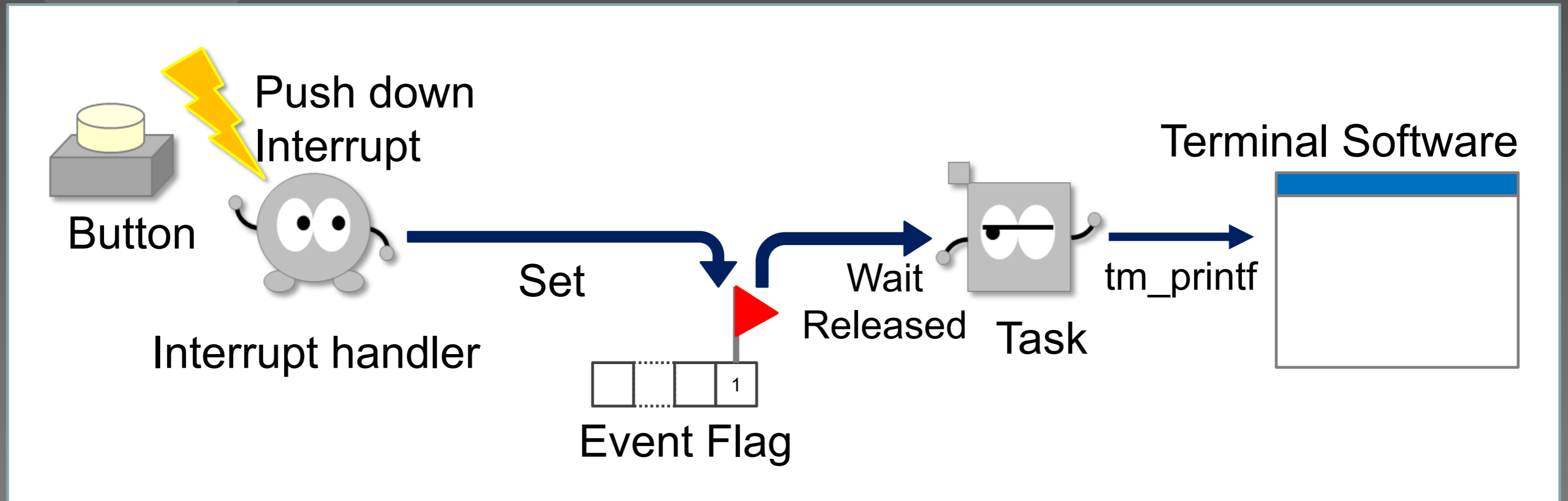
Using Peripheral Devices

- (1) GPIO and push button switch interrupt
- (2) Put “イ” pattern on LED matrix
- (3) Music and dimming LED by PWM
- (4) A/D conversion and joystick
- (5) I2C and acceleration sensor
- (6) UART comm. between 2 boards



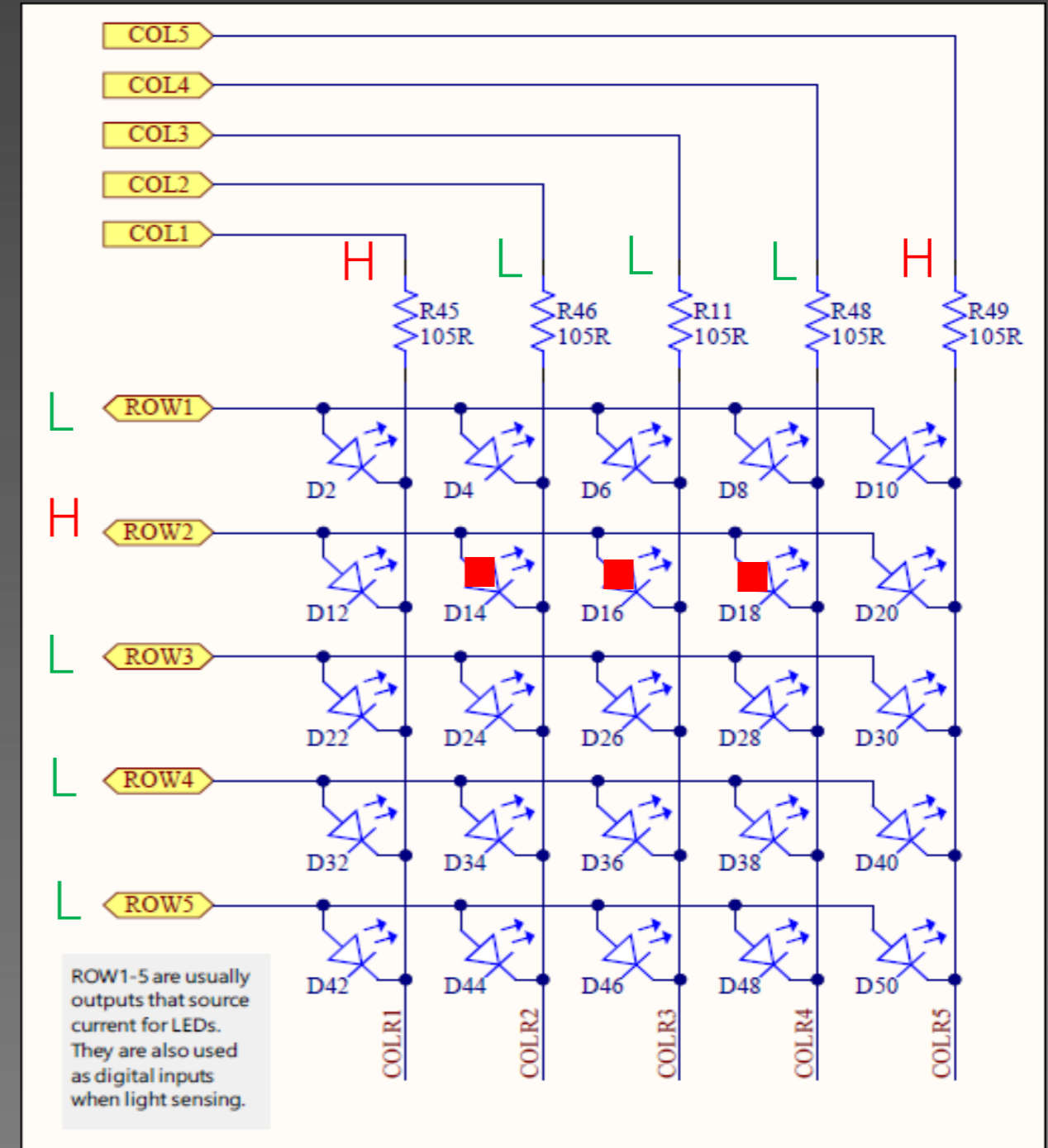
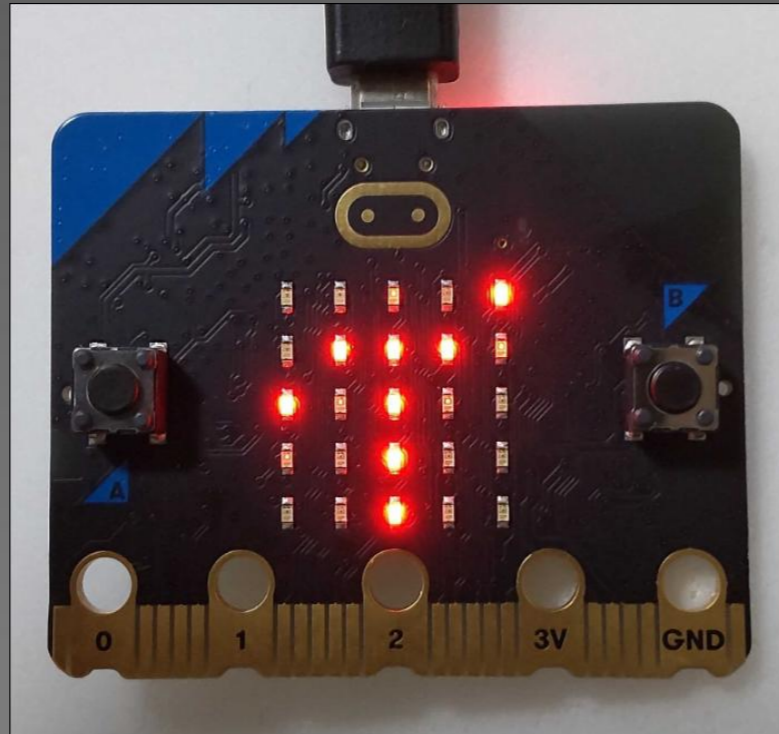
Using Peripheral Devices

- (1) GPIO and push button switch interrupt
 - Push button → interrupt → set event flag → task wait released → print



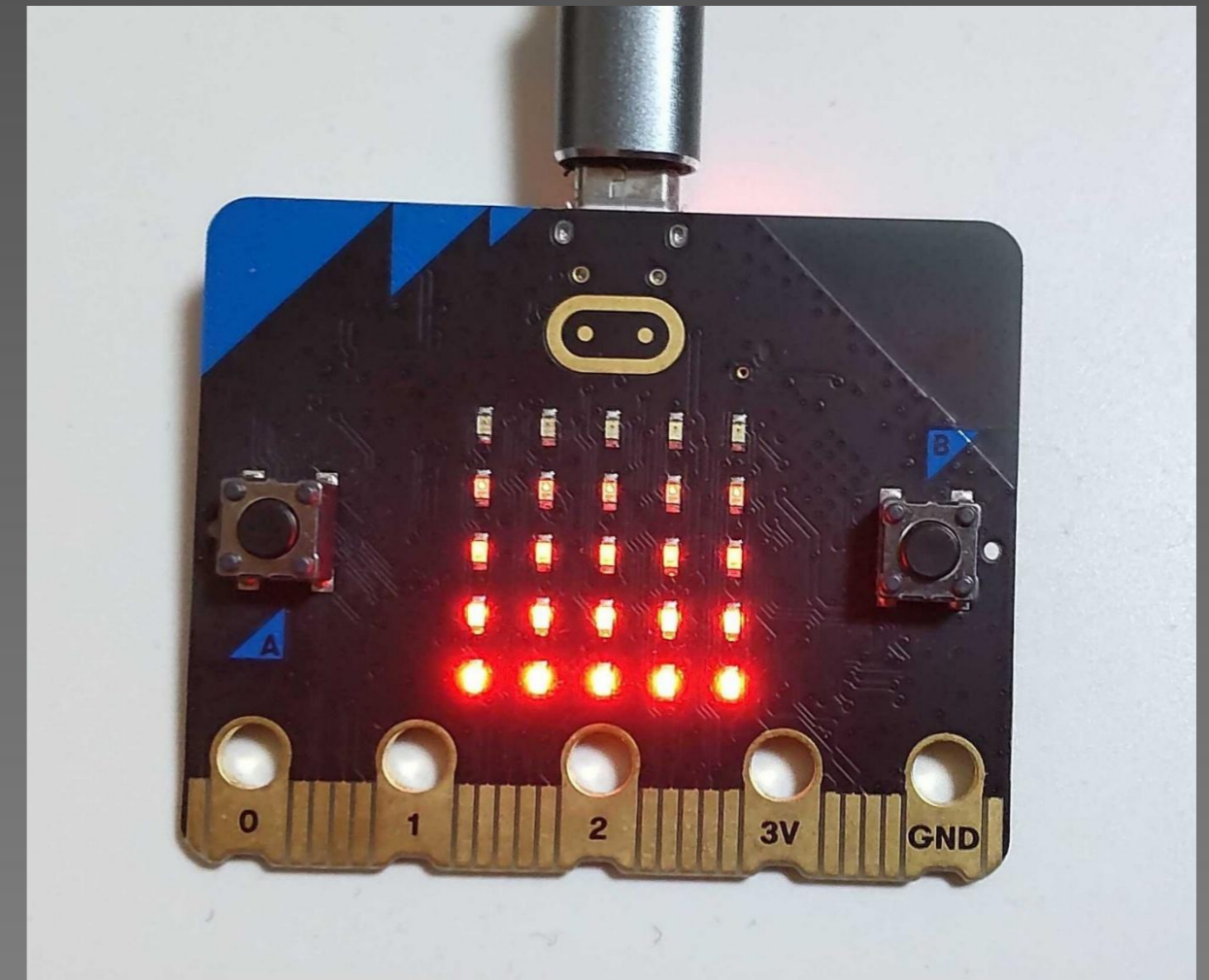
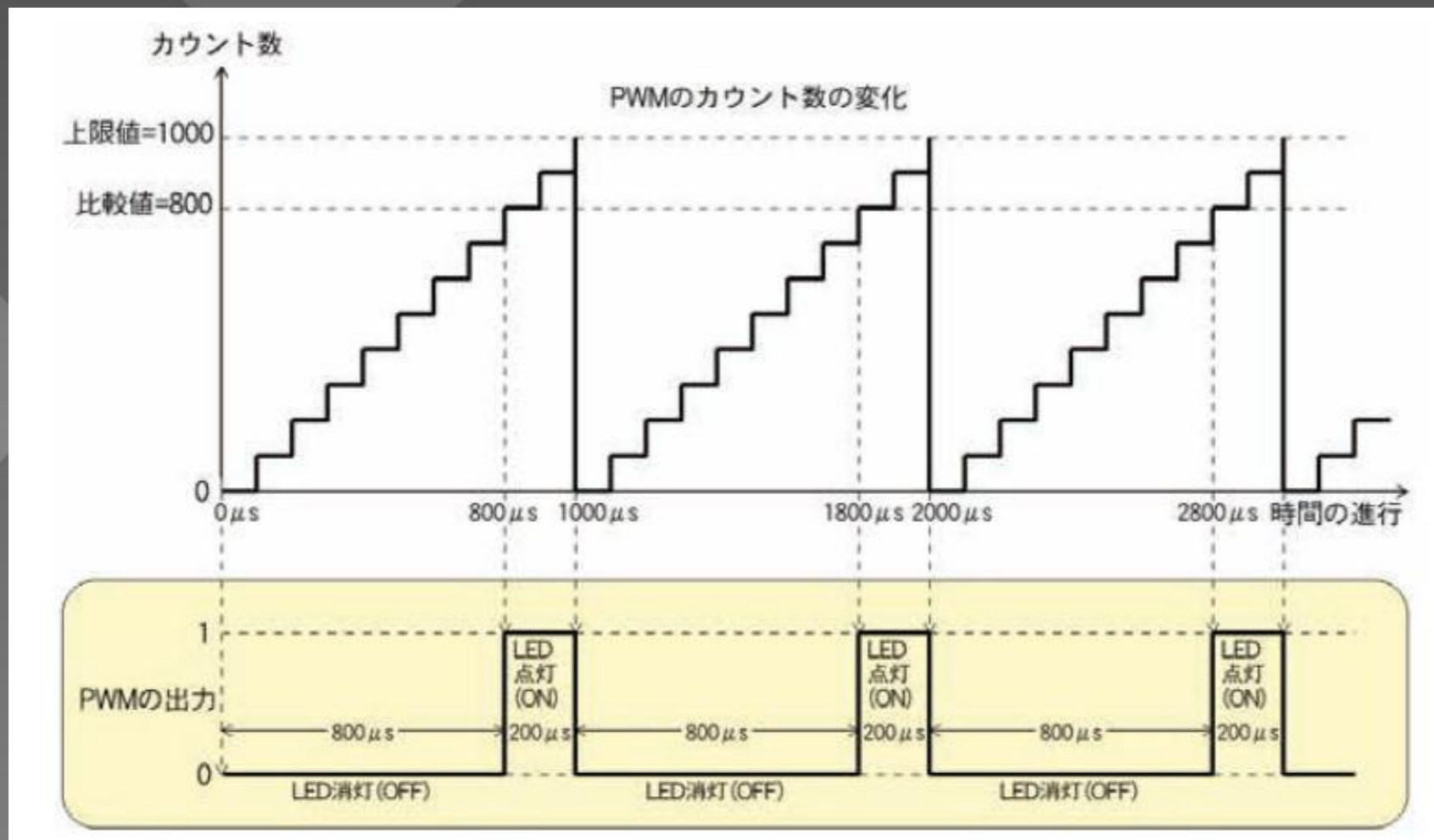
Using Peripheral Devices

- (2) Put pattern “1” on LED matrix
 - LED=ON \Leftrightarrow Row=High & Col=Low
 - Dynamic drive to put any patterns
 - 10 bits but 2^{25} patterns
 - ✓ (“1” was a test pattern of early TV in 1926)



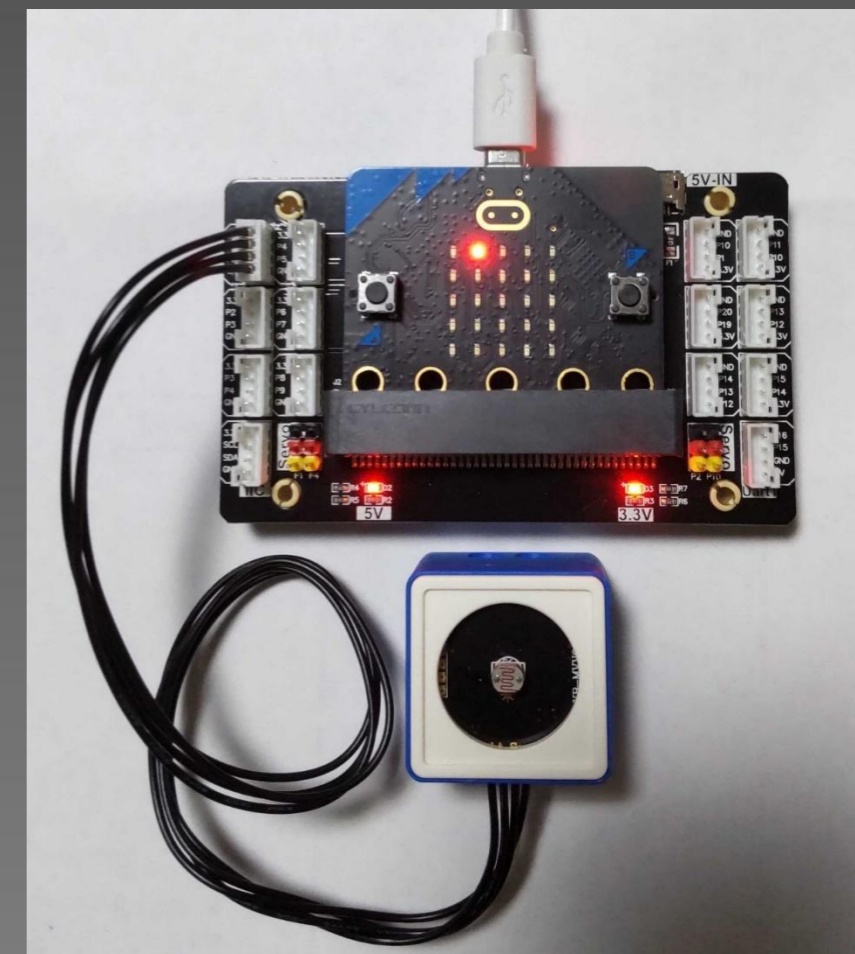
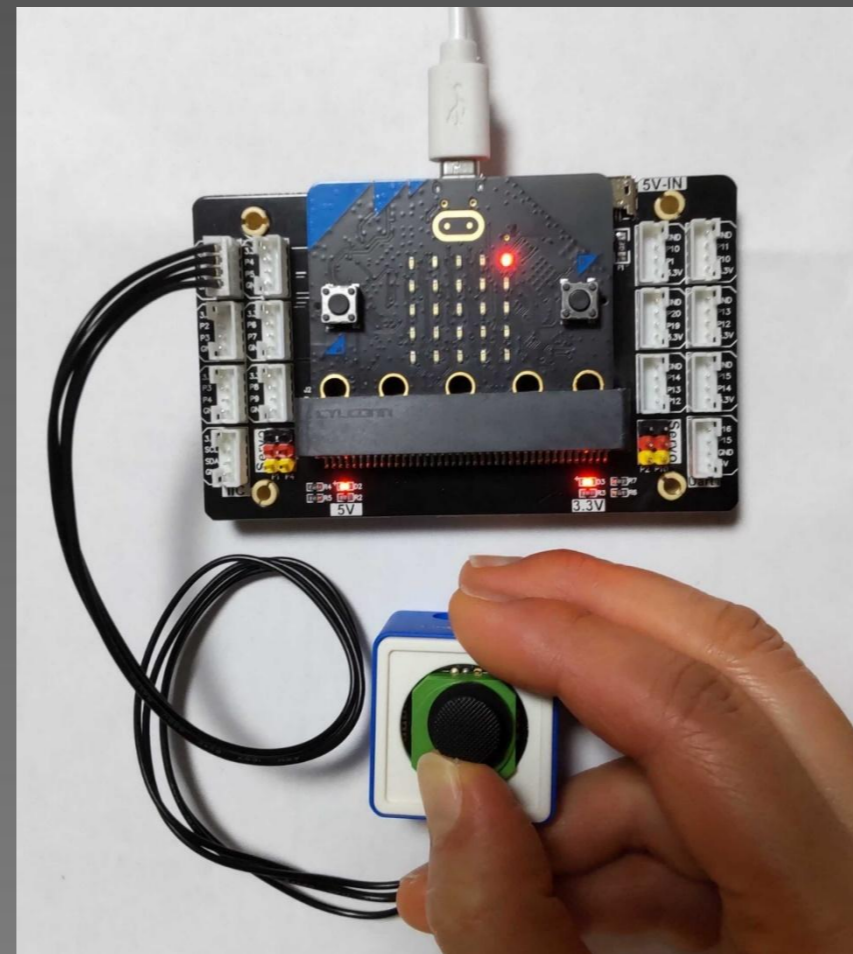
Using Peripheral Devices

- (3) Play music & dimming LED by PWM: Pulse Width Modulation
 - Example 1: OFF 800 μ s, ON 200 μ s \rightarrow Freq. 1000Hz / Duty ratio 20%
 - Example 2: OFF 100 μ s, ON 400 μ s \rightarrow Freq. 2000Hz / Duty ratio 80%



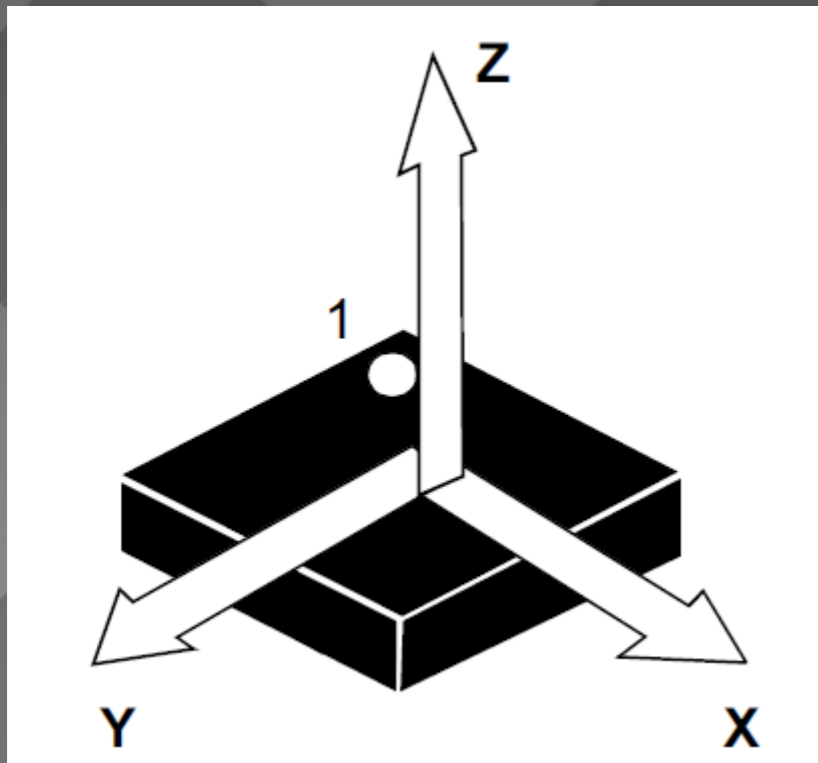
Using Peripheral Devices

- (4) A/D conversion and joy stick
 - Joystick included in "World of Module Sensor Kit" by iftiny:
 - ✓ <https://store.iftiny.com/products/yahboom-world-of-module-programmable-sensor-kit-for-microbit-v2>



Using Peripheral Devices

■ (5) I2C and acceleration sensor



```
microT-Kernel Version 3.00
```

```
WHO_AM_I_A = 0x33
```

```
CTRL_REG1_A = 0x57
```

```
Acc: x,y,z= -10, -6, 237
```

```
Acc: x,y,z= -8, -4, 247
```

```
Acc: x,y,z= -14, -11, 240
```

```
Acc: x,y,z= 36, 2, 241
```

```
Acc: x,y,z= 97, 3, 224
```

```
Acc: x,y,z= 127, 17, 208
```

```
Acc: x,y,z= 144, 6, 201
```

```
Acc: x,y,z= 11, 12, 259
```

```
Acc: x,y,z=-112, -5, 214
```

```
Acc: x,y,z=-165, 2, 192
```

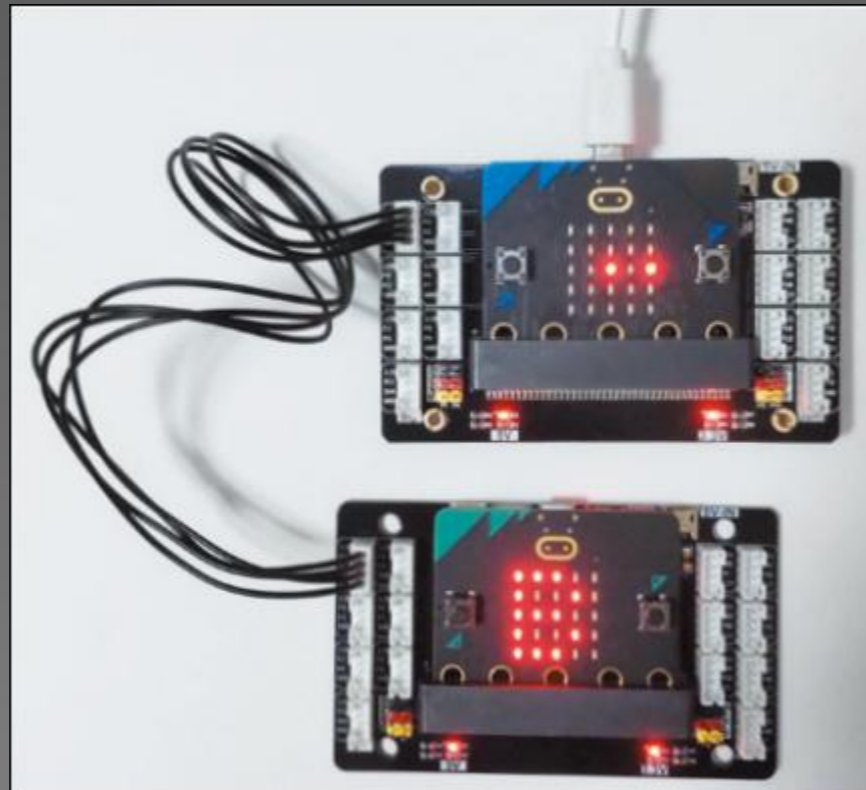
← ボードが水平 (micro:bit の文字が上)

← ボードの左側を下に傾ける

← ボードの右側を下に傾ける

Using Peripheral Devices

- (6) UART communication between micro:bit A and B
 - micro:bit A is programmed by μ T-Kernel 3.0
 - micro:bit B can be programmed by MakeCode



Using Peripheral Devices

- Sample Programs in articles on TRONWARE magazine and website
- (1) GPIO and push button switch interrupt (VOL.201)
 - <https://www.t-engine4u.com/info/mbit/5.html>, <https://www.t-engine4u.com/info/mbit/6.html>
- (2) Put pattern “イ” on LED matrix (VOL.202)
 - <https://www.t-engine4u.com/info/mbit/7.html>
- (3) Play music and dimming LED by PWM (VOL.205)
 - <https://www.t-engine4u.com/info/mbit/10.html>
- (4) A/D conversion and joystick (VOL.206)
 - <https://www.t-engine4u.com/info/mbit/11.html>
- (5) I2C and acceleration sensor (VOL.207)
 - <https://www.t-engine4u.com/info/mbit/12.html>
- (6) UART communication between two micro:bit boards (VOL.208)
 - <https://www.t-engine4u.com/info/mbit/13.html>

Using generative AI to program micro:bit + μT-Kernel

- Generative AI can write program source code by inputting some prompts
- Prompt example
 - Write a program to put pattern “イ” for μT-Kernel 3.0 on micro:bit
- See TRONWARE VOL.217



2025 TRON Symposium — TRONSHOW —

● 最初の部分はまったく正しい。GPIOのアドレス定義も正しい。

● nRF51822はmicro:bit V1.5のCPUであり、現在入手できるV2.2ではnRF52833。nRF51822はmicro:bit V1.5のCPUであり、現在入手できるV2.2ではnRF52833。

```

#include <tk/kernel.h>
...
/* GPIOレジスタのベースアドレス (nRF51822) */
#define GPIO_BASE 0x50000000
#define GPIO_OUT (*(volatile unsigned int *) (GPIO_BASE + 0x004))
#define GPIO_OUTSET (*(volatile unsigned int *) (GPIO_BASE + 0x008))
#define GPIO_OUTCLR (*(volatile unsigned int *) (GPIO_BASE + 0x00C))
#define GPIO_DIRSET (*(volatile unsigned int *) (GPIO_BASE + 0x018))
        
```

● LEDマトリックスのピン配置で、行(Row)は正しいが、列(Column)は間違っている。

● また、COL4のみGPIO P0ではなくP1を用いるため、GPIOアドレスの定義も別が必要。

LEDマトリックスのピン配置	GPIOのポート番号とピン番号	LEDマトリックスのライン番号	GPIOのポート番号とピン番号
COL1	P0.28	ROW1	P0.21
COL2	P0.11	ROW2	P0.22
COL3	P0.31	ROW3	P0.15
COL4	P1.05	ROW4	P0.24
COL5	P0.30	ROW5	P0.19

● 「イ」の文字のドットパターンが不自然。特に難しいところではないと思うが...

```

/* 「イ」のドットパターン (5x5マトリックス) */
/* 各行のビットパターン: ビット4が左端、ビット0が右端 */
static unsigned char ledptn_1[LED_ROWS] = {
    0b00100, // 1行目: ●●●●● (中央に縦線)
    0b00100, // 2行目: ●●●●●
    0b00100, // 3行目: ●●●●●
    0b00100, // 4行目: ●●●●●
    0b01110, // 5行目: ●●●●● (下部を広く)
};
        
```

図3 AIの生成したC言語プログラム

● 「幻覚」のAPIと危険なコード

GPIOの設定などは正しく、一見動きそうに見えます。しかし、詳しく見ていくと問題がありました。まず、GPIOのピン番号の定義です。micro:bitのハードウェア仕様では、LEDに接続されているGPIOのピン番号が不規則に割り当てられているのですが、AIは列(Column)方向のピン番号を「4、5、6、7、8」といった順番で定義してしまいました。これではLEDは点灯しません。また、「イ」のドットパターンも正しく実現できませんでした。そこで課題を変えて、「1秒ごとに『○』と『×』を交互に表示して」と指示しました。

こうした、LEDのドットパターンは正しく定義されました。また、1秒ごとに表示を切り替えるために、μT-Kernelの「周期ハンドラ」を使うという正しいアプローチを取ってきました(図4)。

これを実際の開発環境(Eclipse)に入ると、エラーが出ました。原因は、AIが「物理タイマ」を使おうとして、「tk_cre_ptmr」という架空のAPI(システムコール)を勝手に作って呼び出していたからです。周期ハンドラのAPI名からの類推で捏造してしまったのです。さらに致命的だったのは、AIが周期ハンドラの中で「tk_wai_sem」(セマフォ待ち)のAPIを使っていたことです。周期ハンドラの中で待ち状態に入るシステムコールを使うことはできないのですが、こういったリアルタイムOSの基本的なルールを、AIは理解できていませんでした。

● 対話的な修正で動作に成功

そこで、「物理タイマのAPIは間違っています」「周期ハンドラの中で待ち状態に入るシステムコールは使えません」と指摘し、正しい仕様書やウェブサイトの記事を参照させました。するとAIは素直に反省し、コードを修正してきました(図5)。ピン番号の定義

● 「○」と「×」のビットパターンは正しく定義されている。

```

/* 「○」のドットパターン (5x5マトリックス) */
static const unsigned char ledptn_circle[LED_ROWS] = {
    0b01100, // 1行目: ●●●●●
    0b01000, // 2行目: ●●●●●
    0b10001, // 3行目: ●●●●●
    0b00001, // 4行目: ●●●●●
    0b10001, // 5行目: ●●●●●
};

/* 「×」のドットパターン (5x5マトリックス) */
static const unsigned char ledptn_cross[LED_ROWS] = {
    0b01001, // 1行目: ●●●●●
    0b01000, // 2行目: ●●●●●
    0b00001, // 3行目: ●●●●●
    0b10001, // 4行目: ●●●●●
    0b10001, // 5行目: ●●●●●
};
        
```

● Eclipseに入ると、残念ながらエラーのマーク

● 物理タイマにはDefinePhysicalTimerHandlerを使う必要がある

- しかし、周期ハンドラのtk_cre_cycからの類推で、tk_cre_ptmrやtk_sta_ptmrといった架空のシステムコールを呼んでいる
- 正しくはDefinePhysicalTimerHandlerとStartPhysicalTimerを使う必要がある
- パラメータの与え方も違う
- このため、プログラムのコンパイルやリンクがエラーになる

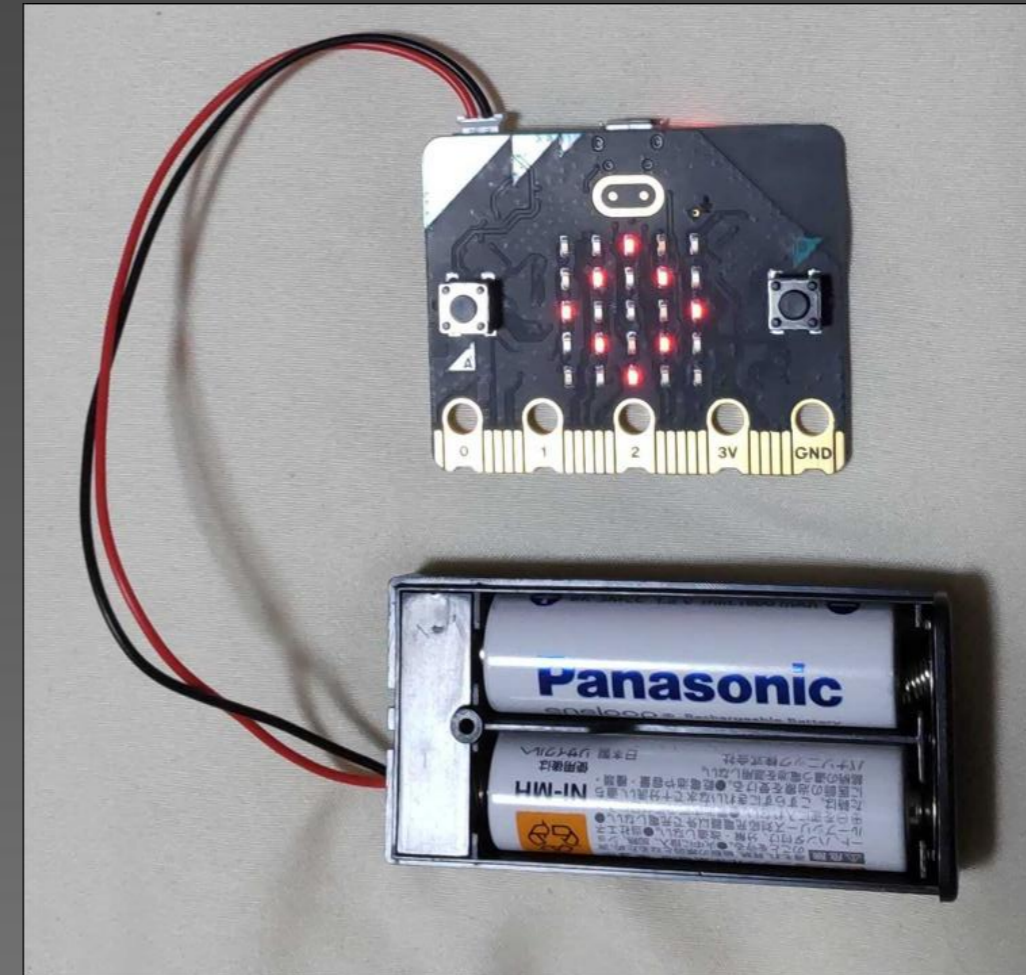
```

T_CPTMR cptmr;
/* 物理タイマ生成 (2ms間隔でデマチック点灯) */
cptmr.ptmrattr = TA_HUNG;
cptmr.ptmrmode = TA_CYCLIC;
cptmr.cycctm = 2000; // 2ms (マイクロ秒単位)
cptmr.ptmrhdr = led_switch_row_hdr;
cptmr.extint = NULL;
ptmr_id = tk_cre_ptmr(&cptmr);
        
```

図4 AIの生成したC言語プログラム(2)

Tips for contest entry and AI utilization

- Take advantage of micro:bit features
 - Mini-sized micro:bit board
 - Battery-powered portability
 - Many peripheral sensors
- Example: mini-sized music instrument
 - (3) Play music by PWM
 - (5) Acceleration sensor
- AI utilization
 - Very small AI framework or machine learning running on micro:bit
 - Connect cloud AI to micro:bit
 - Using AI for development tools
- Your new ideas are welcome



Q & A

- Your questions are welcome



www.tron.org
www.t-engine4u.com