SCC.131: Digital Systems
Introduction to the micro:bit architecture

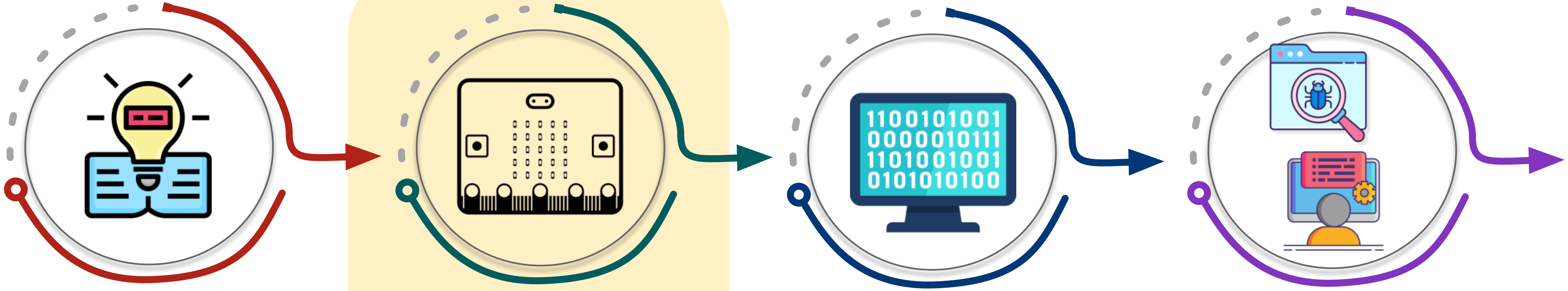
Ioannis Chatzigeorgiou

(i.chatzigeorgiou@lancaster.ac.uk)

Summary of the last lecture

- You met Joe and Steve, two of the inventors of micro:bit!
- Learnt about the initial requirements, e.g.:
  - A more tactile/tangible approach for teaching/learning about computing.
  - Easy to use, easy to engage with.
  - No installation, no setup, no internet.
- Found out differences between micro:bit and arduino / raspberry pi.
- Followed a demo in MakeCode using block coding.
- Given a very high-level overview of the architecture of micro:bit.

Reminder: SCC.131 organization



1. *Computer Architecture*
2. *Micro:bit Architecture*
3. ARM Assembly
4. Systems Architecture

Weeks 7 to 12

Topics covered in Weeks 7 to 12

Lancaster University

Week 7

- Architecting a digital system for global impact
- Introduction to the micro:bit architecture

Week 8

- Introduction to C/C++ CODAL (Parts 1 and 2)

Physical Computing

Week 9

- Compiler, assembler, linker and loader
- The C preprocessor

Week 10

- Debugging
- Revision and discussion of lab tasks

Digital Systems

Topics covered in Weeks 7 to 12

Week 11a	- The micro:bit radio module
Week 11b	- Memory layout (x86-64)
Week 12	- Memory layout (nRF52833 for micro:bit) - Build automation

Physical Computing

Digital Systems

Glossary

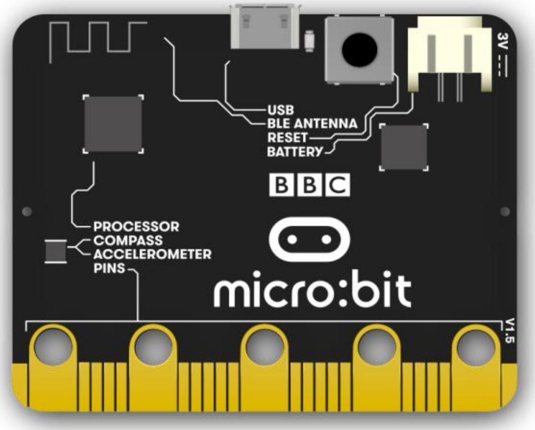
- **Runtime code/software**[1]: Software platform that provides an environment for executing user code. Serves as an abstraction layer that developers can use to write software.
- **Microcontroller**[2]: Often referred to as a processor, for simplicity, or a microcontroller unit (MCU). A compact integrated circuit equipped with one or more central processing units (CPUs) and memory (flash and static RAM). Similar to a system on a chip (SoC).
- **Embedded system**[3]: An electronic product that comprises a microcontroller or multiple microcontrollers executing software instructions stored on a memory module to perform an essential function.

[1] https://tech.microbit.org/software/runtime/
[2] https://en.wikipedia.org/wiki/Microcontroller
[3] https://www.trentonsystems.com/en-gb/blog/what-are-embedded-systems

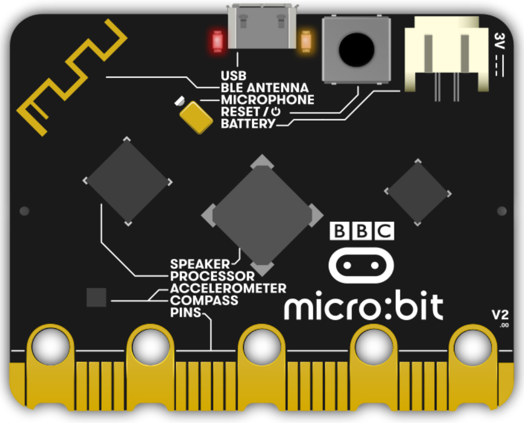
The BBC micro:bit

- Built on BBC’s legacy with the **BBC Micro** for computing in education in 1981-1984.
- Open-source embedded system based on ARM microcontrollers.
- Micro:bit **v1** was announced in 2015.
- Micro:bit **v2** was announced in 2020.
- Version numbers are printed on the rear of the board, e.g., v2.21, v2.00, v1.5, etc.
- What version is yours?

BBC Microcomputer



BBC micro:bit v1.5

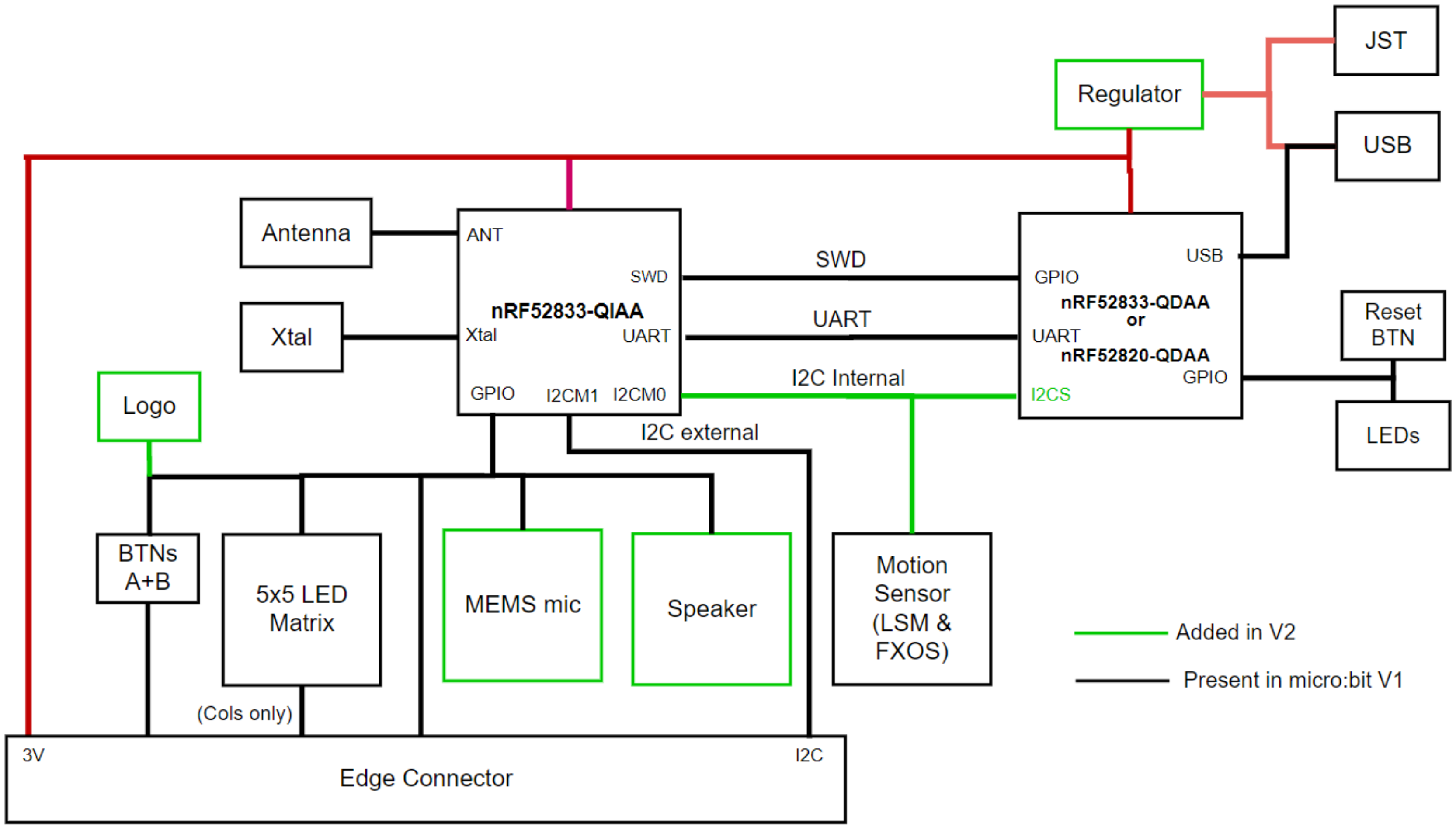


BBC micro:bit v2.00

Main differences between versions

micro:bit	v1	v2.2X
Target/App. MCU	16 MHz 32-bit ARM Cortex-M0	**64 MHz** 32-bit ARM Cortex-M4
Flash memory	256 KB	**512 KB**
Static RAM	16 KB	**128 KB**
Interface MCU	48 MHz ARM Cortex-M0+	**64 MHz ARM Cortex-M4F**
Extras	3-axis accelerometer and magnetometer (compass), and temperature sensor. LEDs can sense light.	The same as v1 as well as **microphone** with LED indicator, **speaker**, **touch sensor button**.

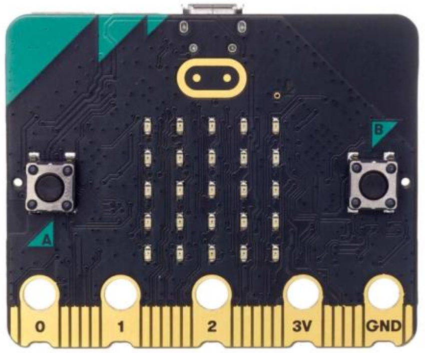
Hardware



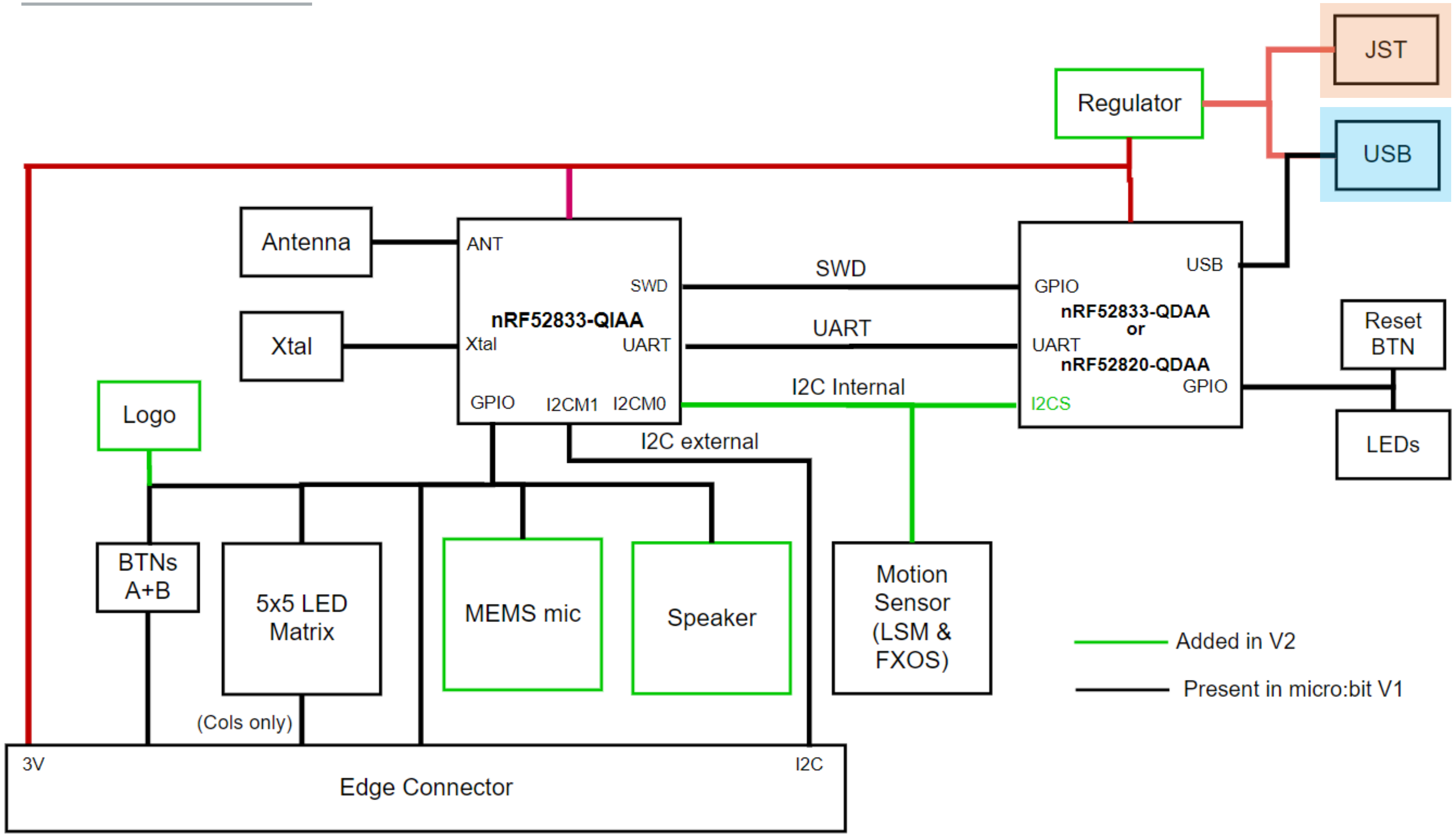
JST
Regulator
USB
Antenna
ANT
SWD
nRF52833-QIAA
Xtal
UART
GPIO
I2CM1
I2CM0
SWD
UART
I2C Internal
I2C external
Xtal
Logo
USB
GPIO
nRF52833-QDAA
or
UART
nRF52820-QDAA
GPIO
I2CS
Reset BTN
LEDs
BTNs A+B
5x5 LED Matrix
(Cols only)
MEMS mic
Speaker
Motion Sensor (LSM & FXOS)
3V
Edge Connector
I2C
Added in V2
Present in micro:bit V1

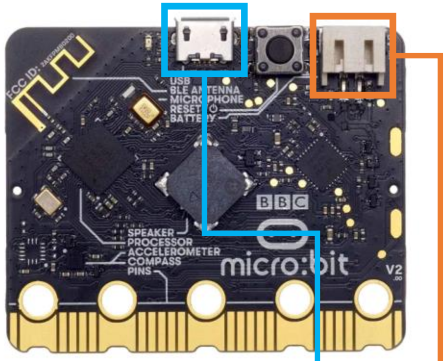
Rear side

Front side



Hardware





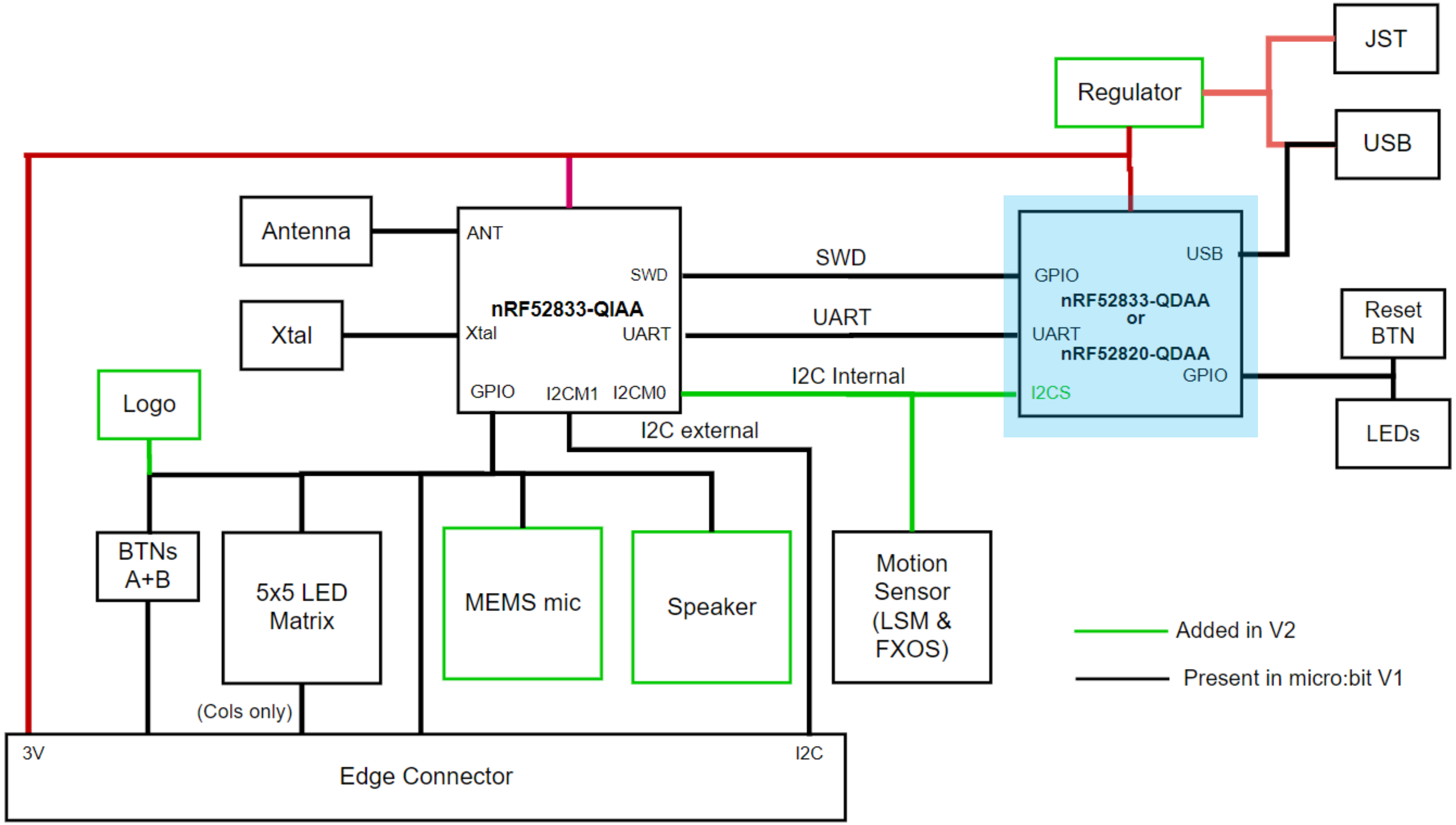
Micro USB

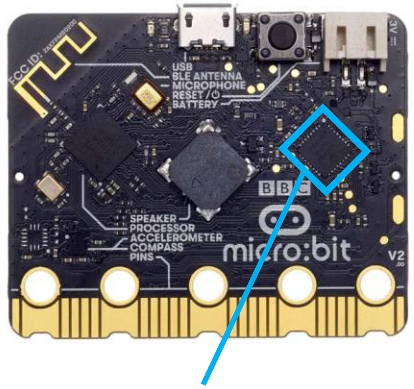
Streams data to and from the micro:bit and provides power.

JST connector

Power supply (3V battery).

Hardware

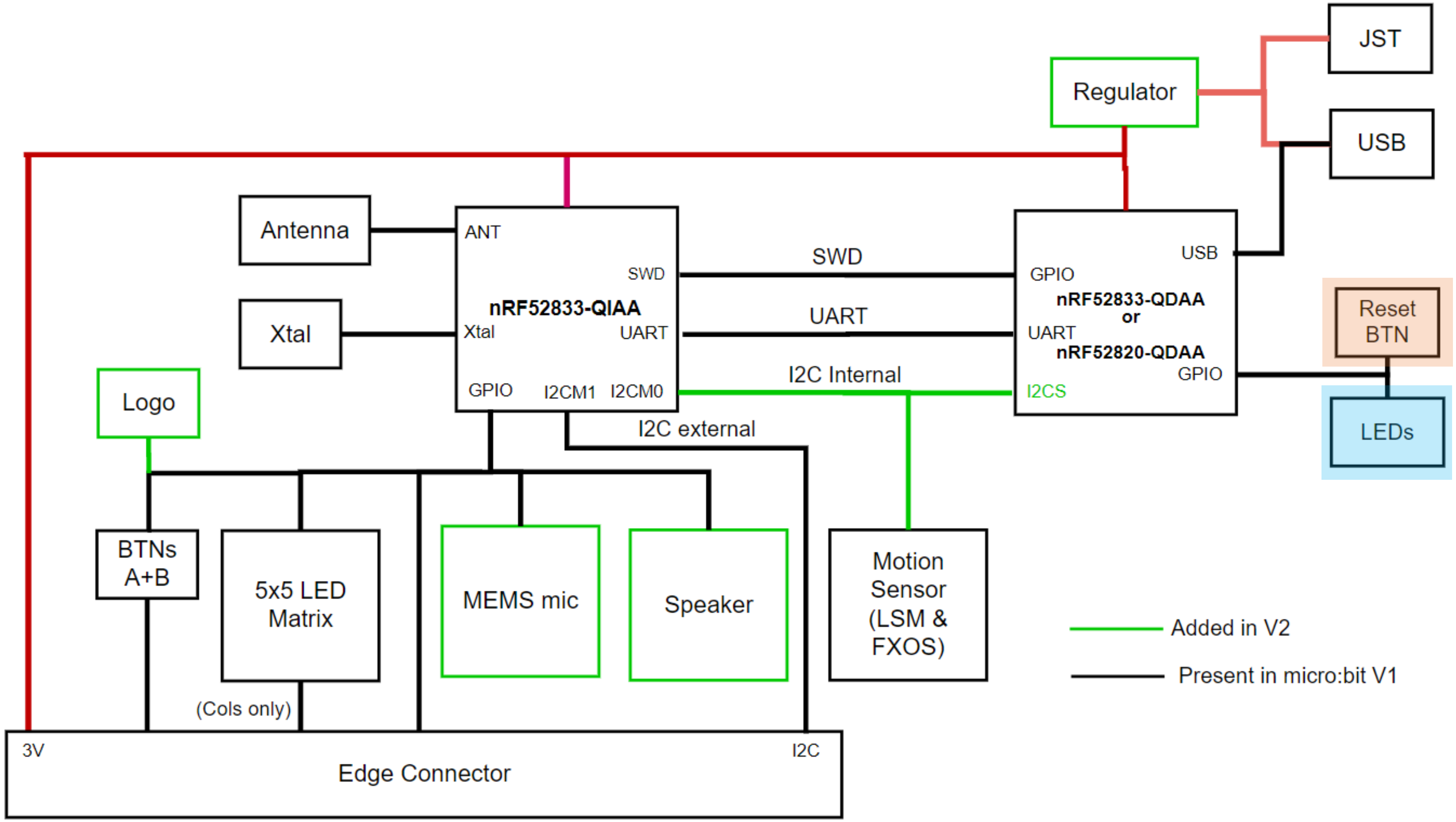


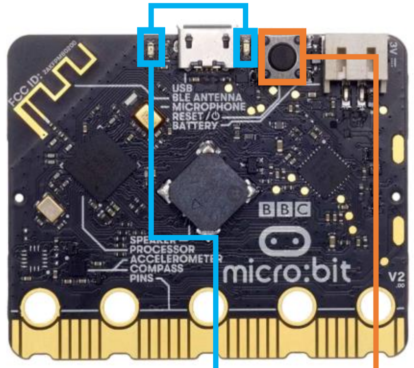


ARM Cortex-M4F processor

Interface that handles USB connection. Used for flashing code and Tx/Rx data to/from connected devices.

Hardware





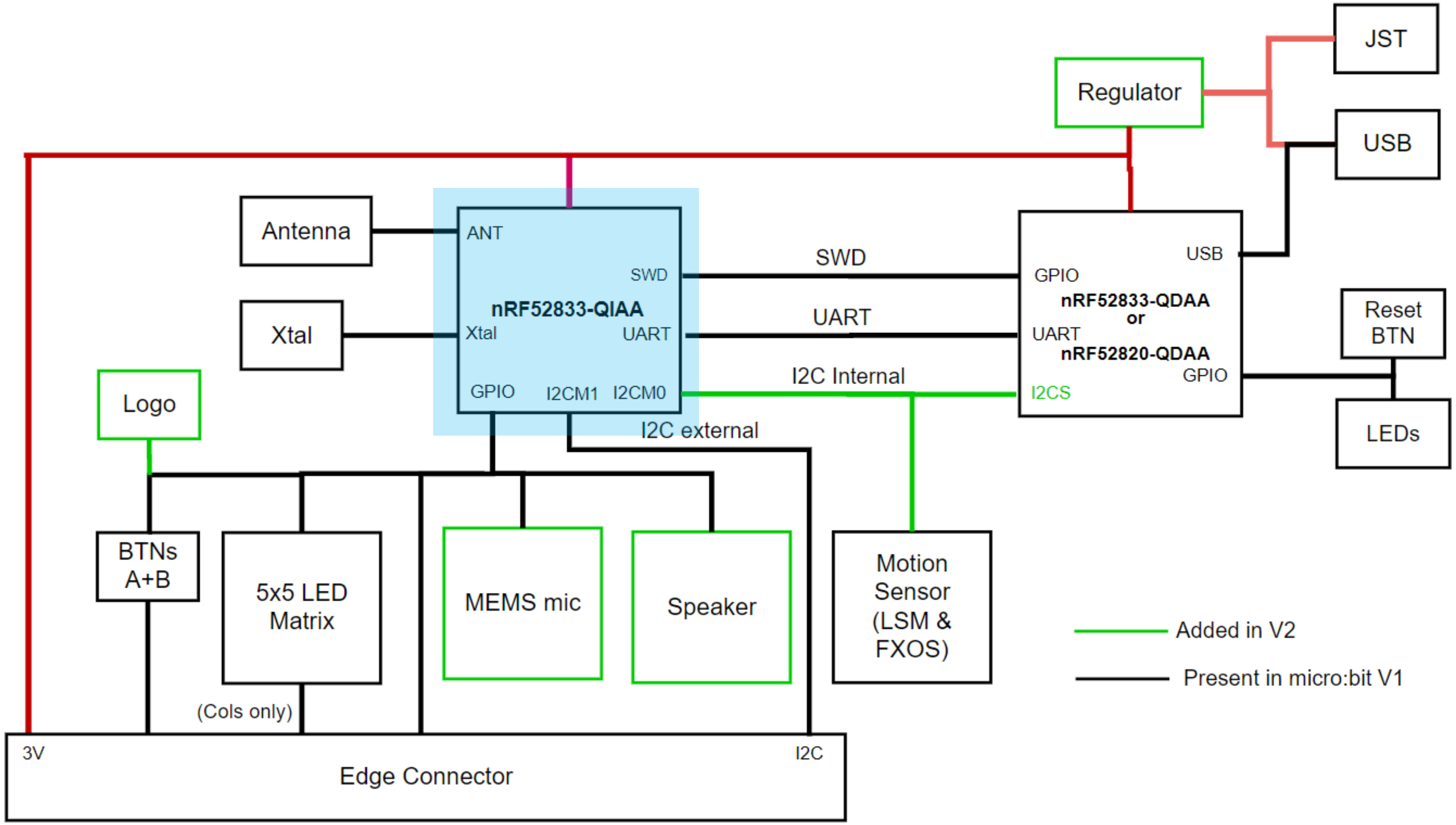
LEDs

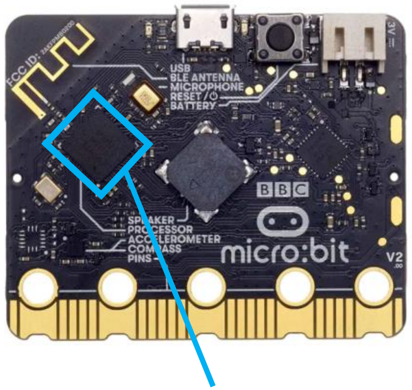
Power indicator (L) and USB activity (R).

Reset button

Restart code, sleep/wake.

Hardware

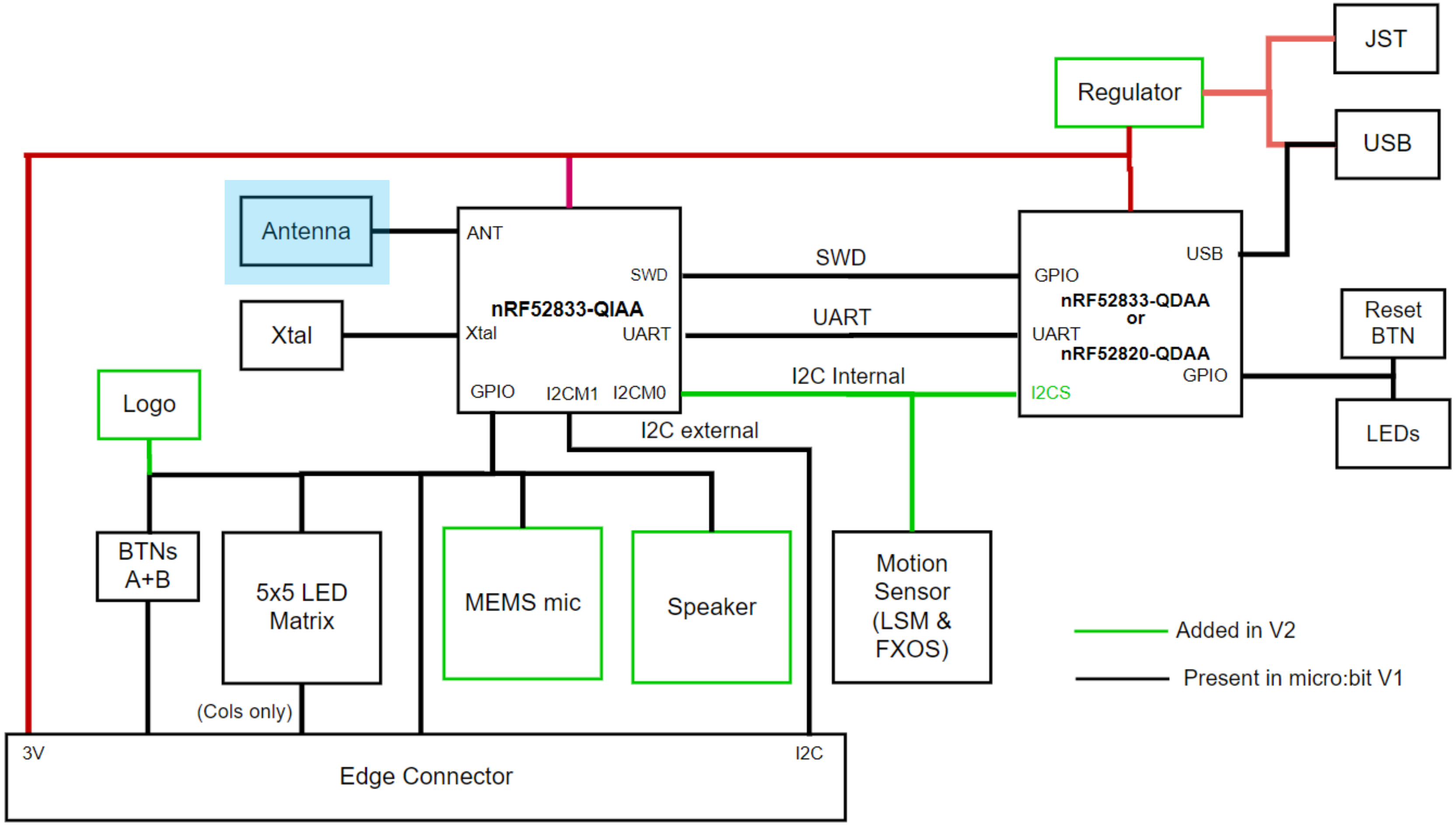


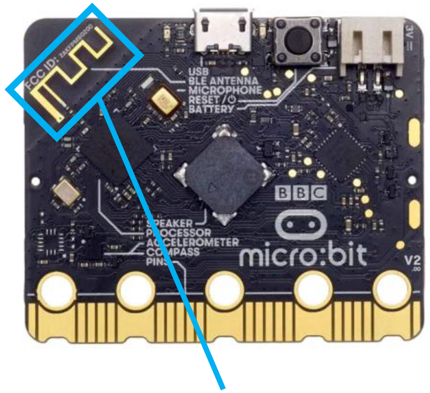


ARM Cortex-M4 32-bit processor

User code, runtime code and Bluetooth stack run from flash memory. Custom radio capabilities. Integrated temperature sensor.

Hardware



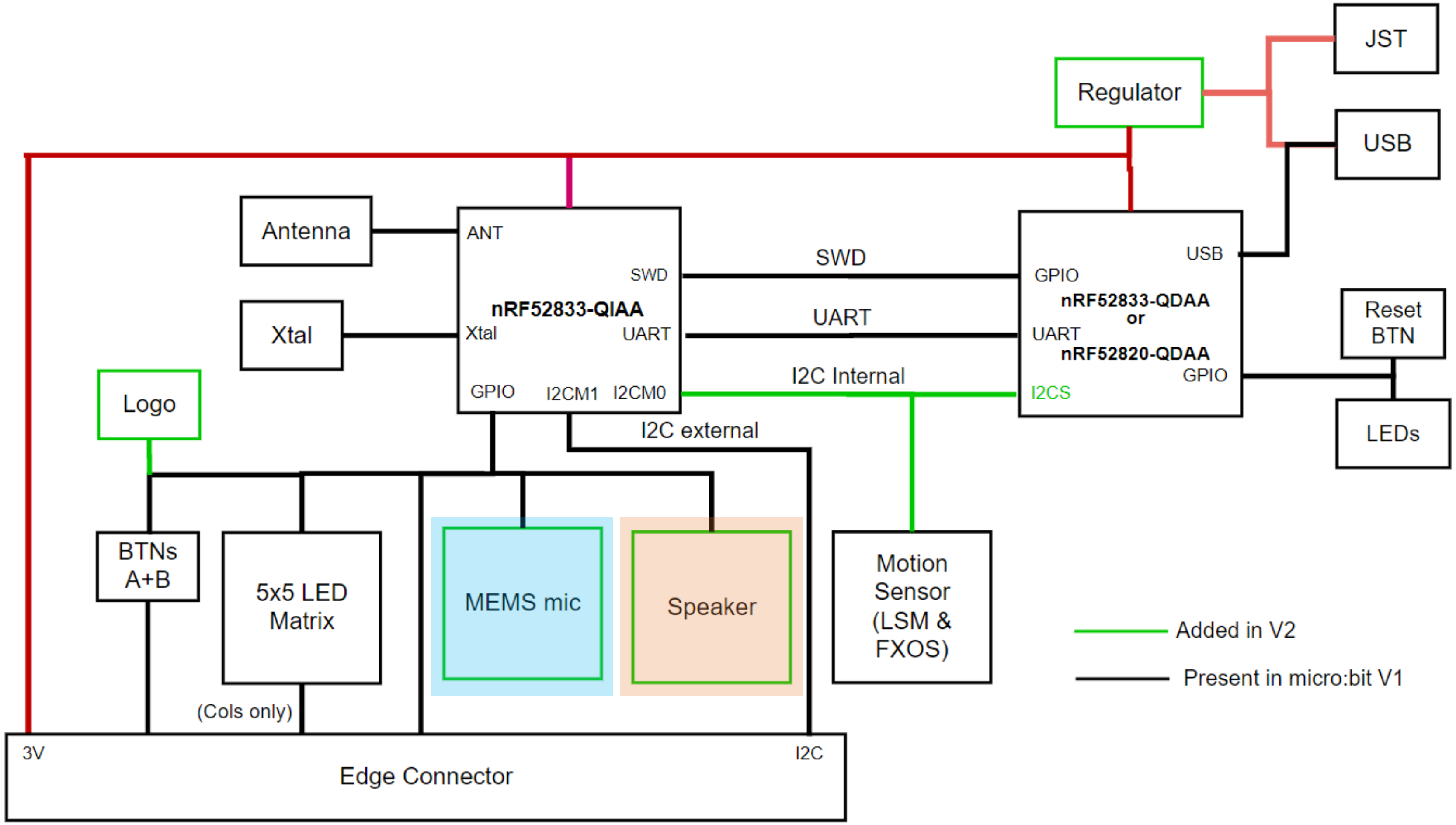


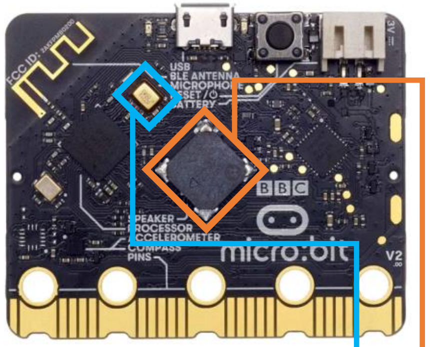
Bluetooth low energy (BLE) antenna

Bluetooth: micro:bit (Peripheral) talks with Central devices only.

Radio: micro:bit devices can talk to each other.

Hardware





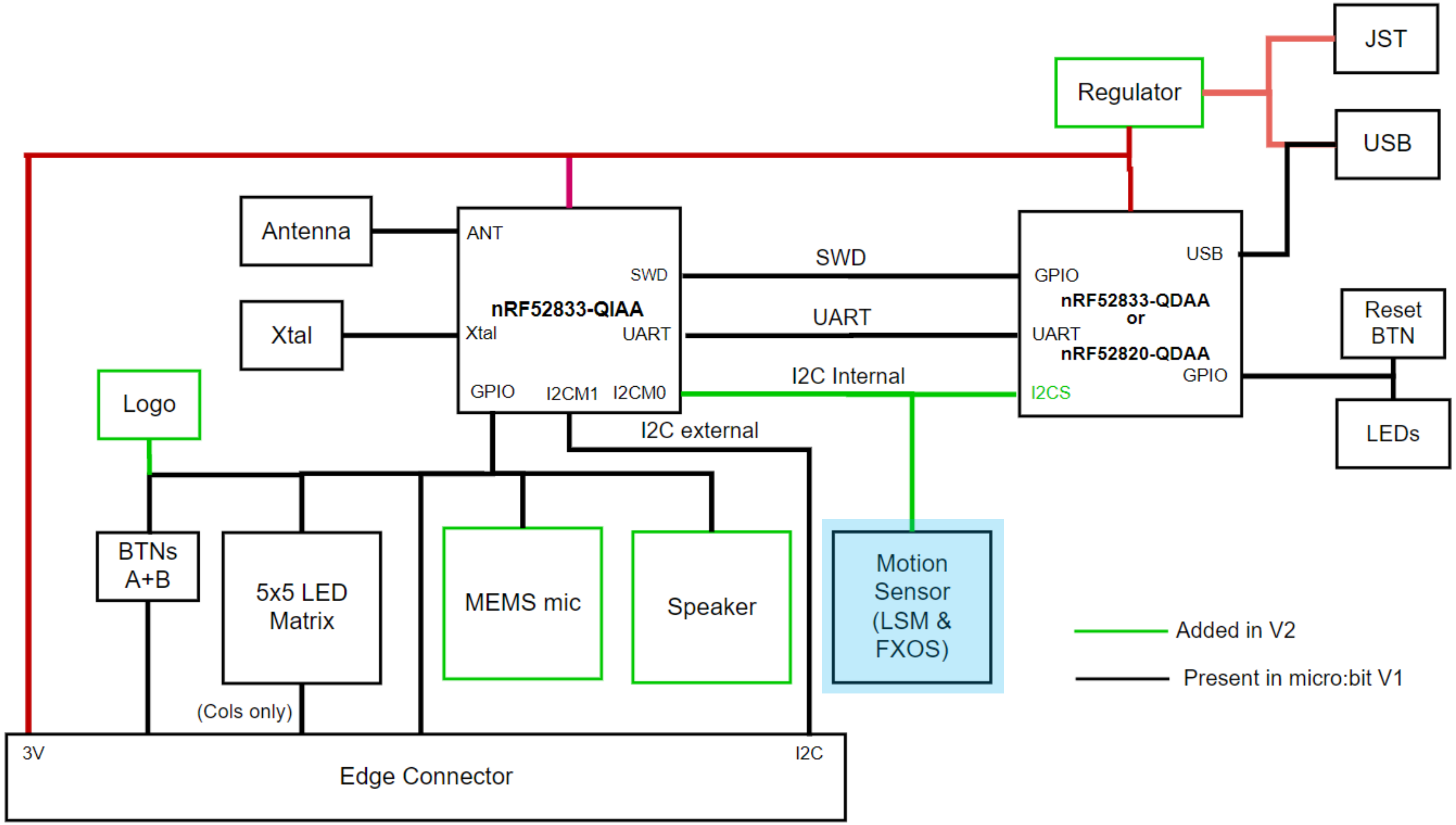
Microphone

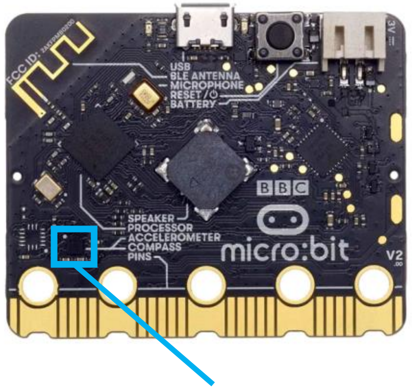
Micro-electro-mechanical system (MEMS).

Speaker

Sound also on General Purpose Input/Output (GPIO) pins.

Hardware

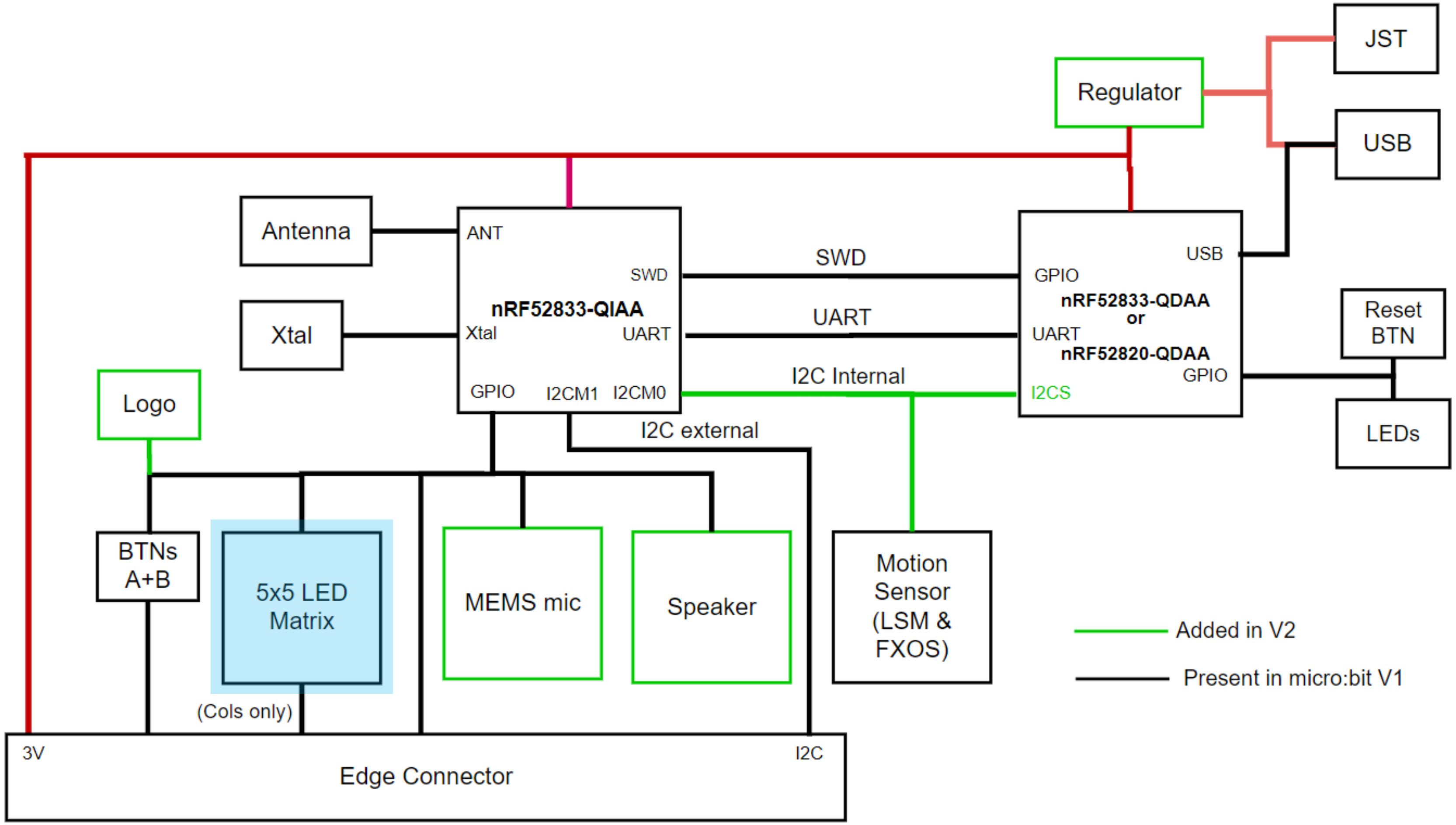




Motion sensor

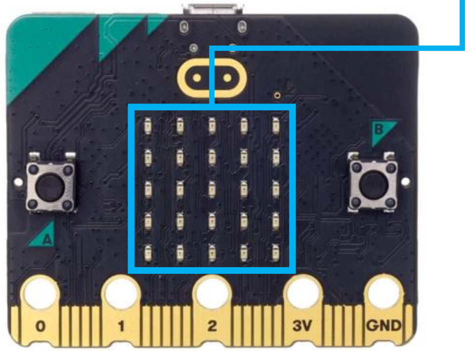
Accelerometer and magnetometer. Footprint for two sensors (LSM & FXOS) but only one is placed.

Hardware

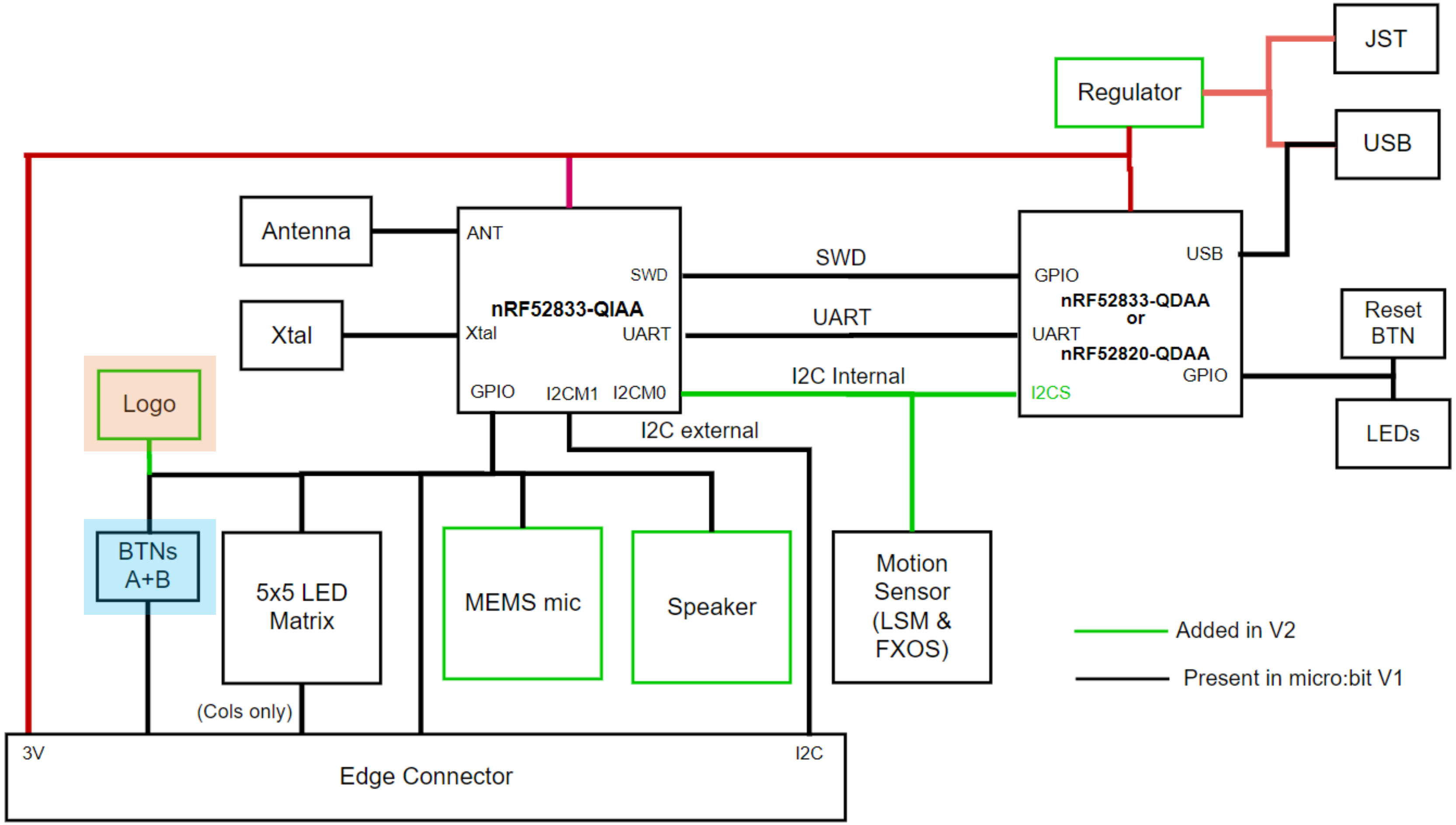


Display (25 red LEDs) refreshed by runtime software at high speed. Senses ambient light. The columns of the matrix are connected to GPIO pins.

5×5 LED matrix



Hardware

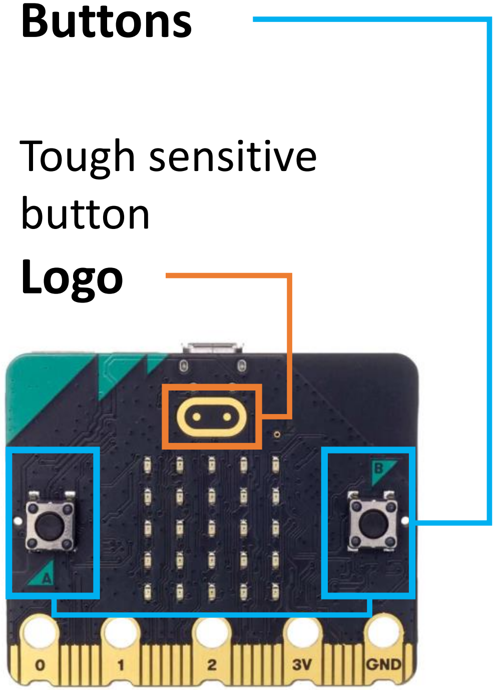


Detects A, B and A+B pressed together.

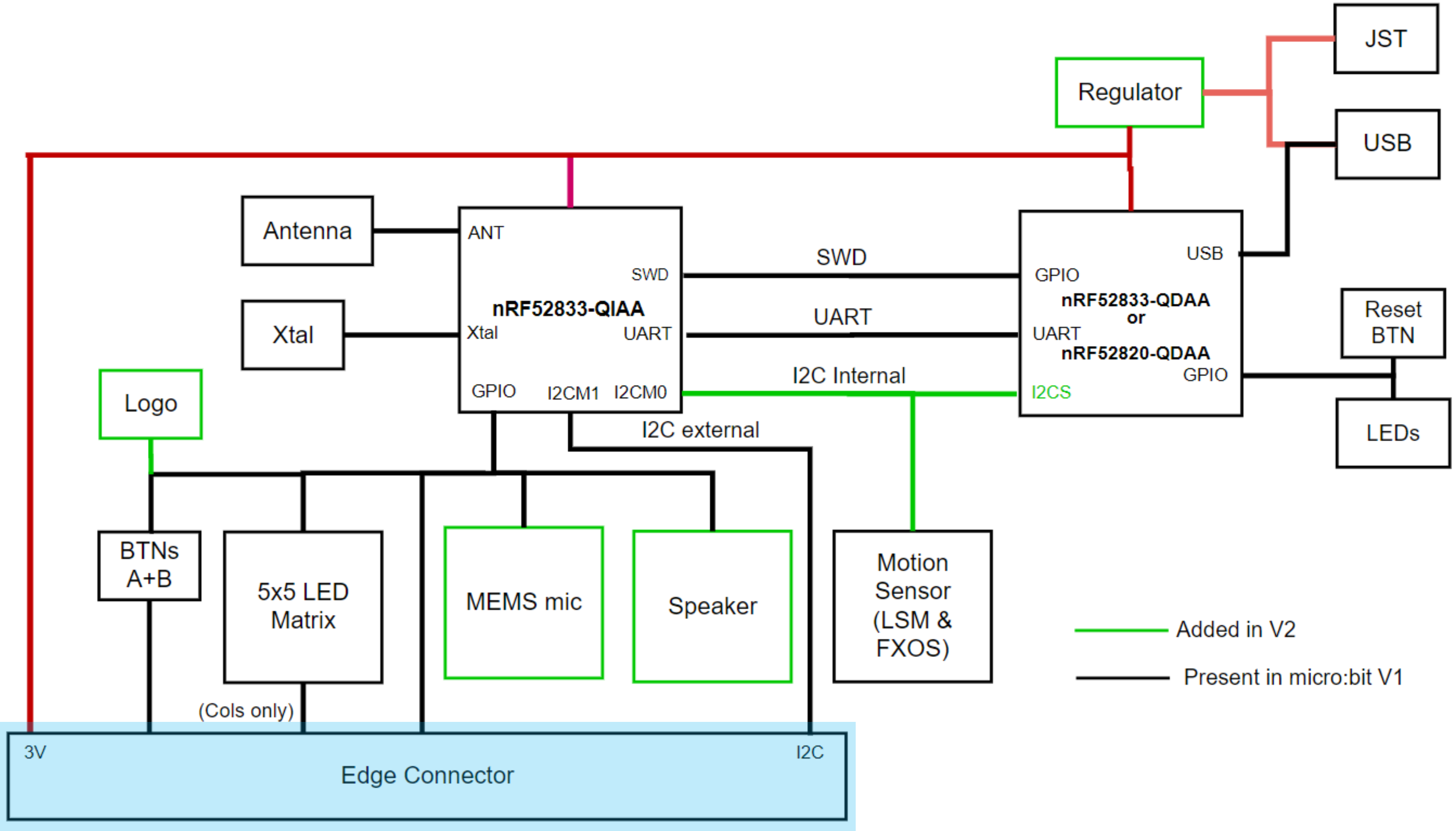
Buttons

Tough sensitive button

Logo

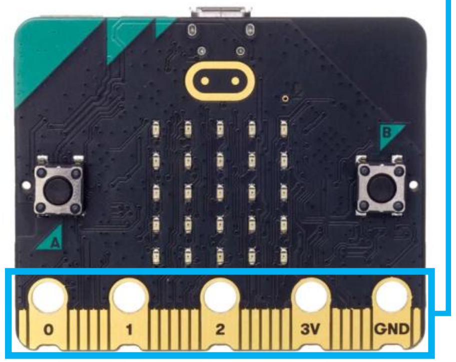


Hardware

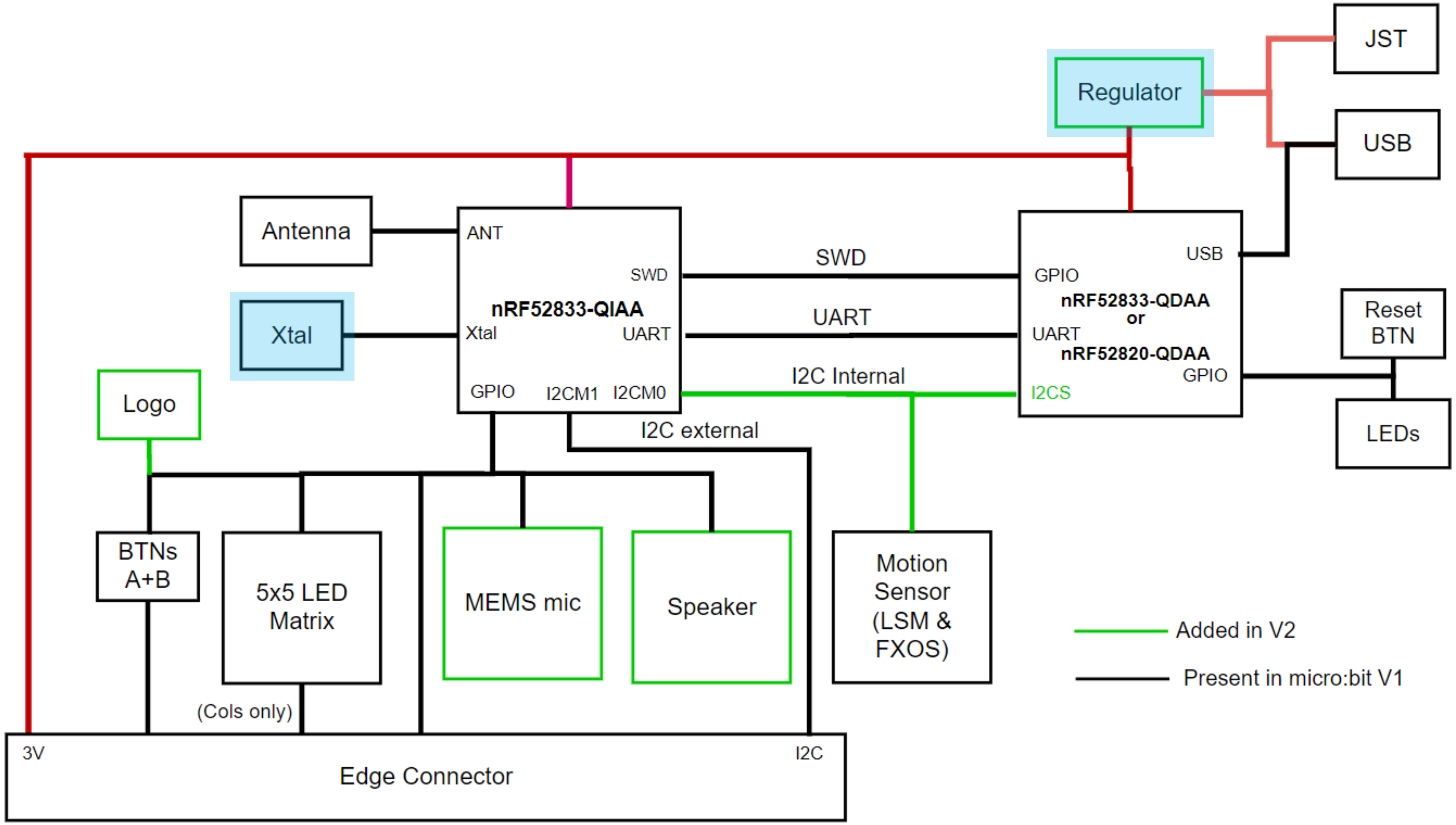


20 pins/strips and 5 rings for connection with external components using crocodile clips or banana plugs, respectively.

Edge connector



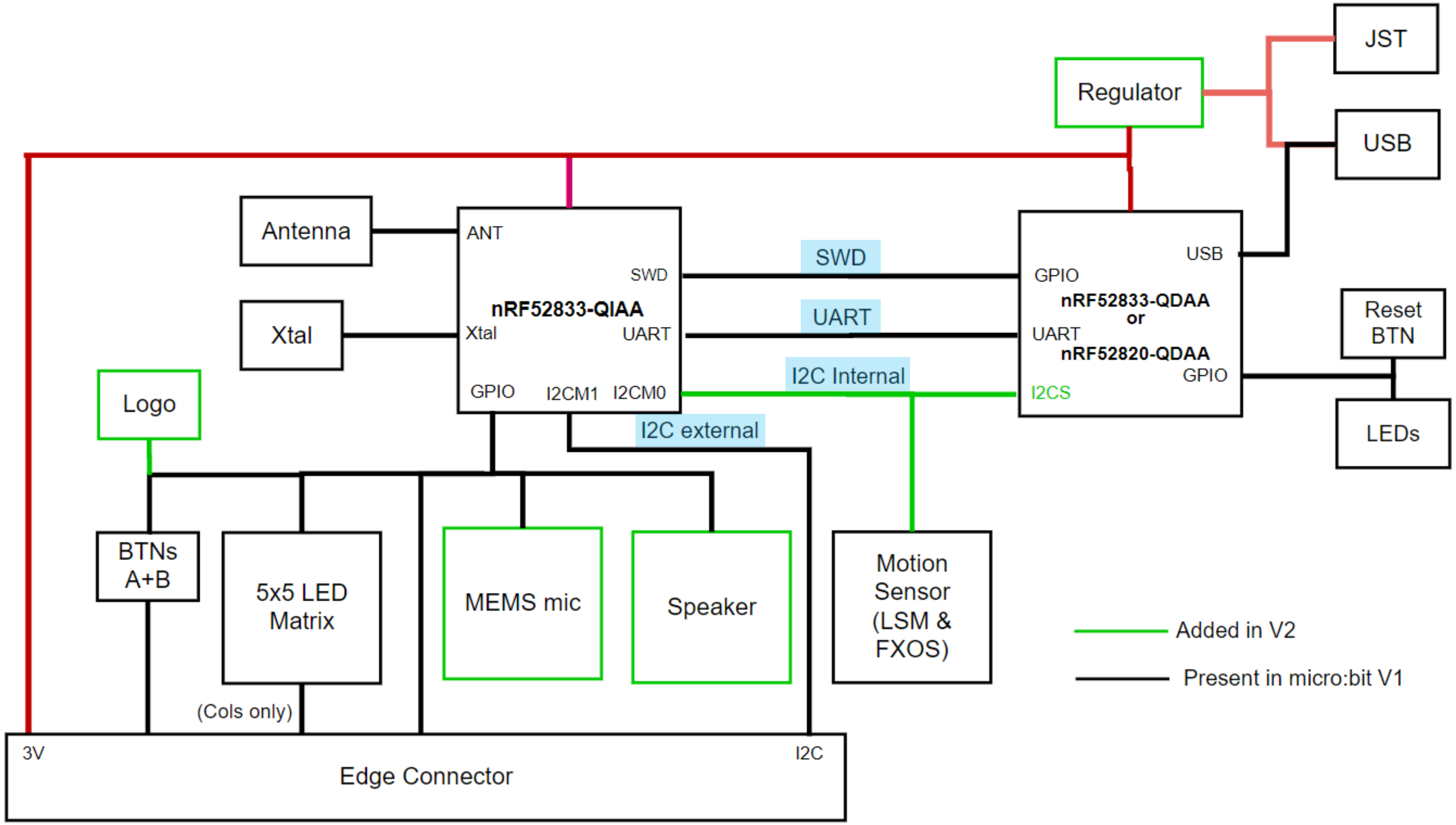
Hardware



Regulator: Dedicated on-board regulator that steps down voltage to 3.3V, suitable for powering the micro:bit.

Xtal: Electronic *crystal* oscillator that provides the clock signal.

Hardware

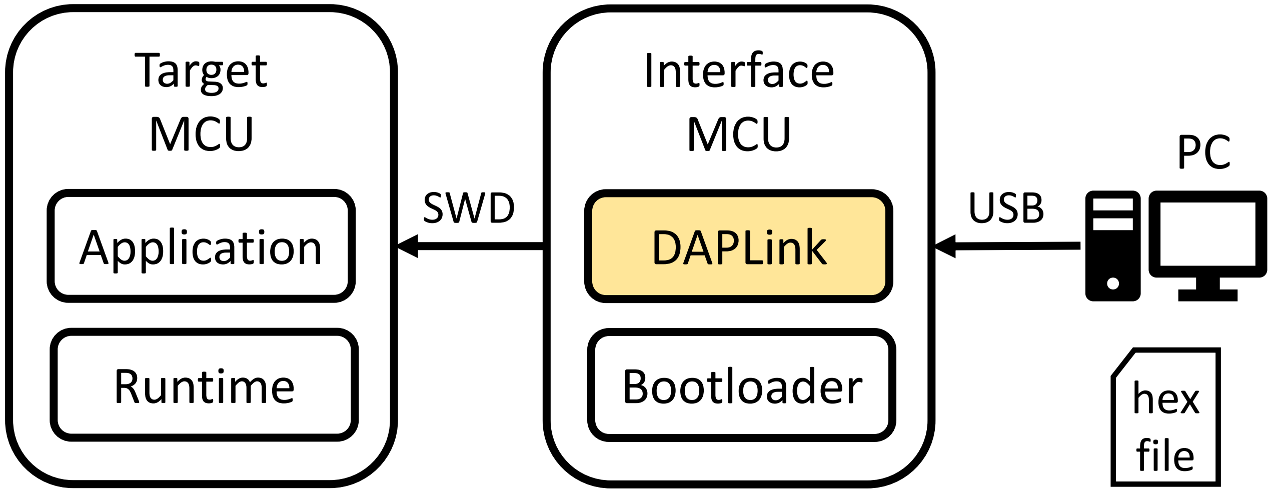


SWD: Serial wire debug for programming the target MCU.

UART: Universal asynchronous receiver-transmitter for exchanging data with the device connected to the USB.

I^{2}C: Inter-integrated circuit bus that allows the main component (target MCU) to communicate with secondary components.

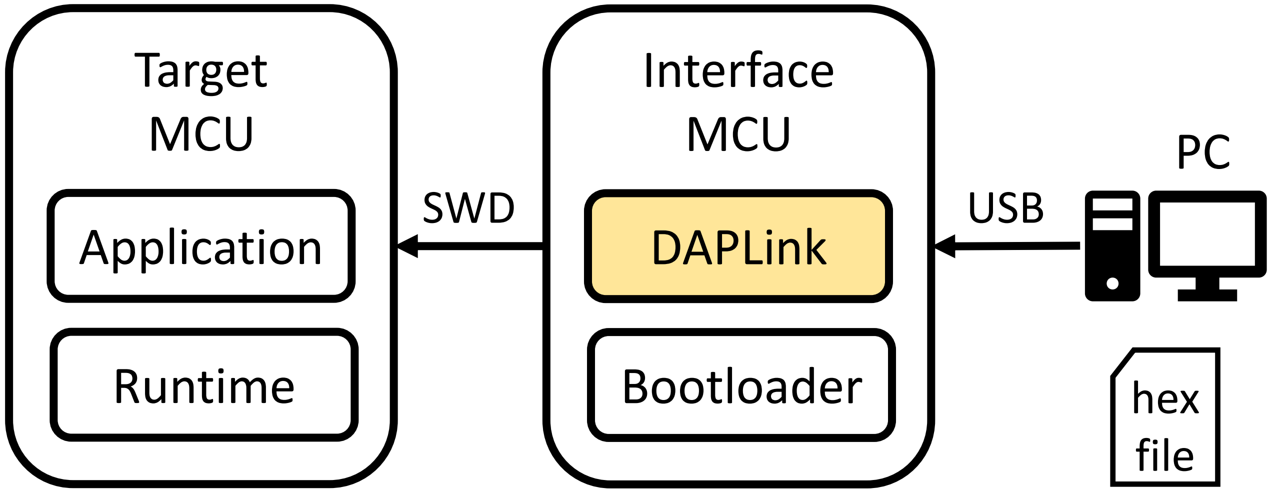
Firmware (DAPLink)



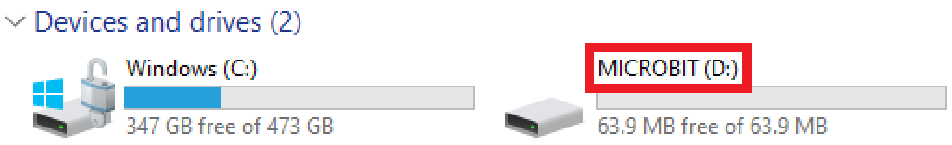
DAPLink:

- Open-source interface firmware that creates a 'bridge' between PC and SWD.
- The micro:bit presents itself as a USB disk.
- Enables drag-and-drop programming without installing drivers.
- Compatible with Windows, MacOS and Linux.

Firmware (DAPLink)



Interface mode



Devices and drives (2)

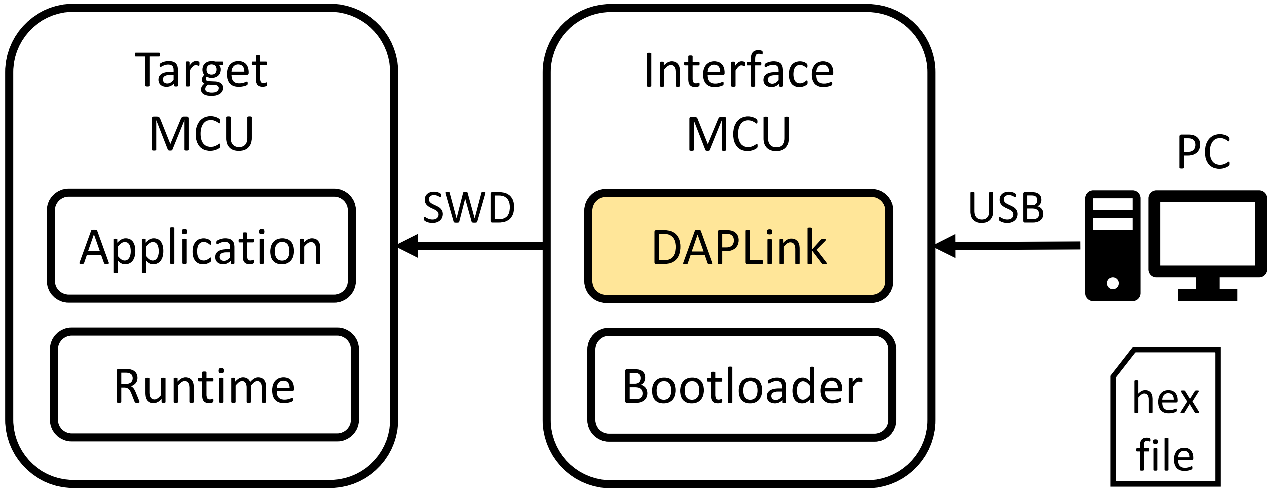
Windows (C:)
347 GB free of 473 GB

MICROBIT (D:)
63.9 MB free of 63.9 MB

DAPLink:

- *Interface mode*: The hex file dropped onto the ‘USB disk’ is written into the **target** MCU flash. The name of the USB disk is `MICROBIT`.

Firmware (DAPLink)

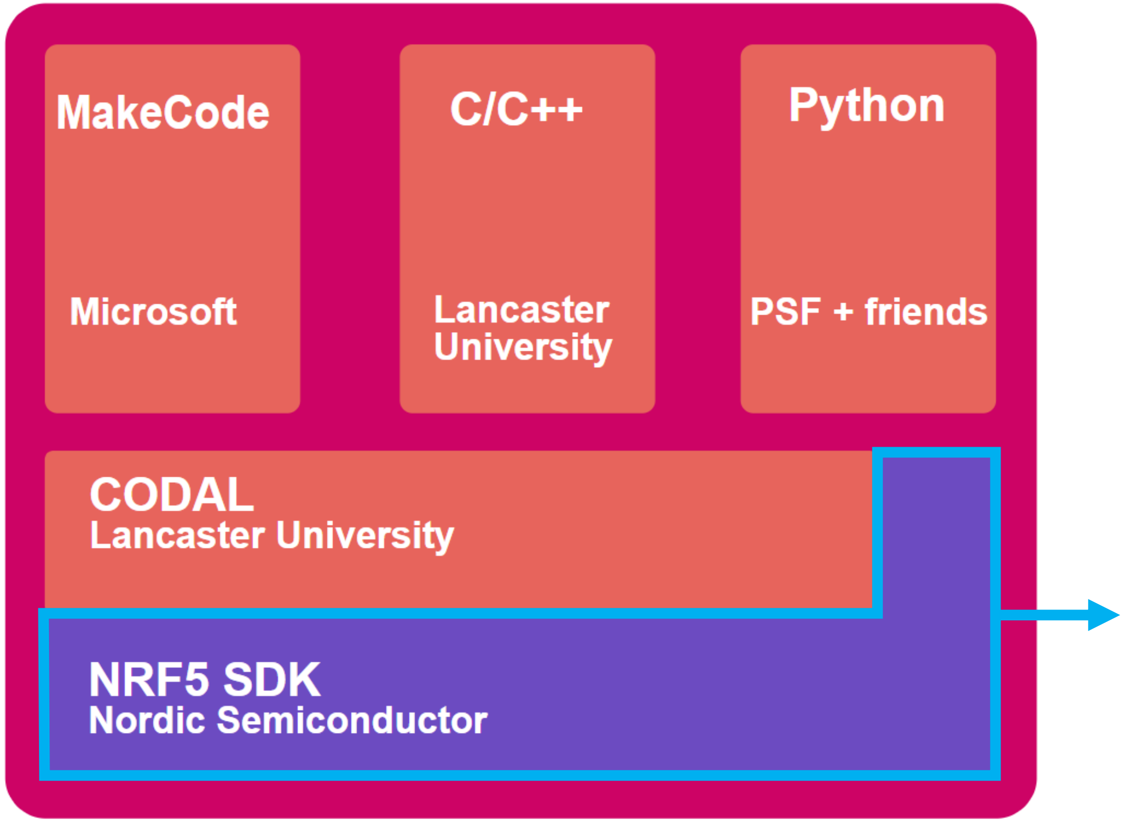


DAPLink:

- *Bootloader mode*: The hex file dropped onto the ‘USB disk’ is written into the **interface** MCU flash and updates the version of DAPLink. The name of the USB disk is `MAINTENANCE`.

To activate the bootloader mode, hold the reset button and plug the USB cable into the micro:bit and PC.

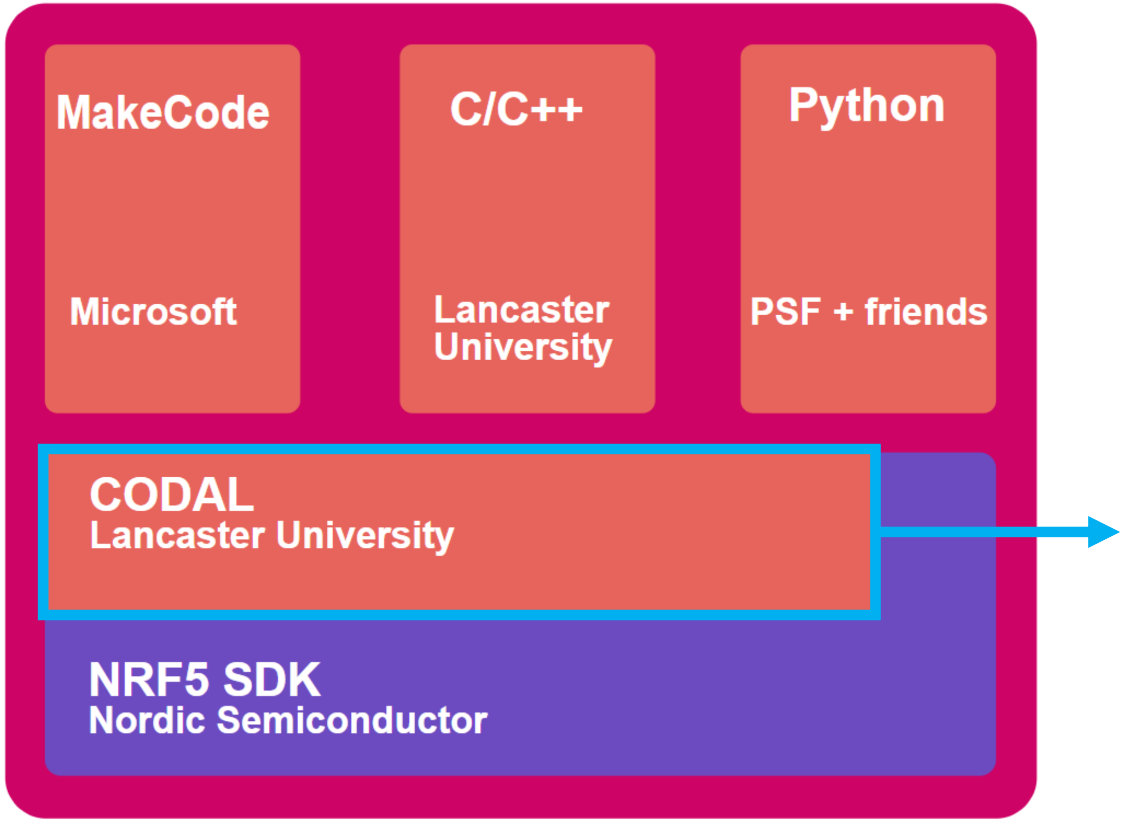
Software



nRF5 Software Development Kit:

- Rich development environment for nRF51/nRF52 series MCUs (Nordic).
- Provides hardware abstraction.
- Includes drivers, libraries, examples of peripherals and radio protocols.

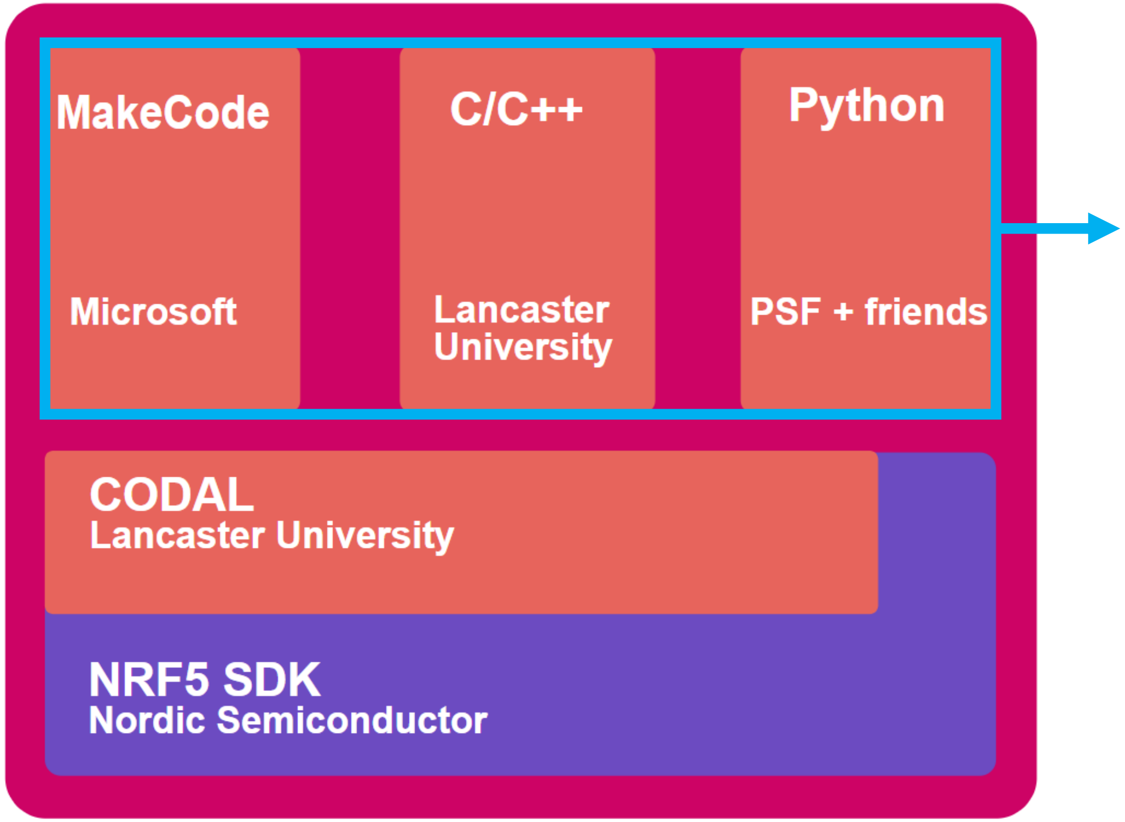
Software



Component Oriented Device Abstraction Layer:

- This is the micro:bit runtime software.
- Written in C/C++.
- Abstracts hardware components as software components represented by C++ classes.
- Offers eventing subsystem for mapping asynchronous events to event handlers.

Software



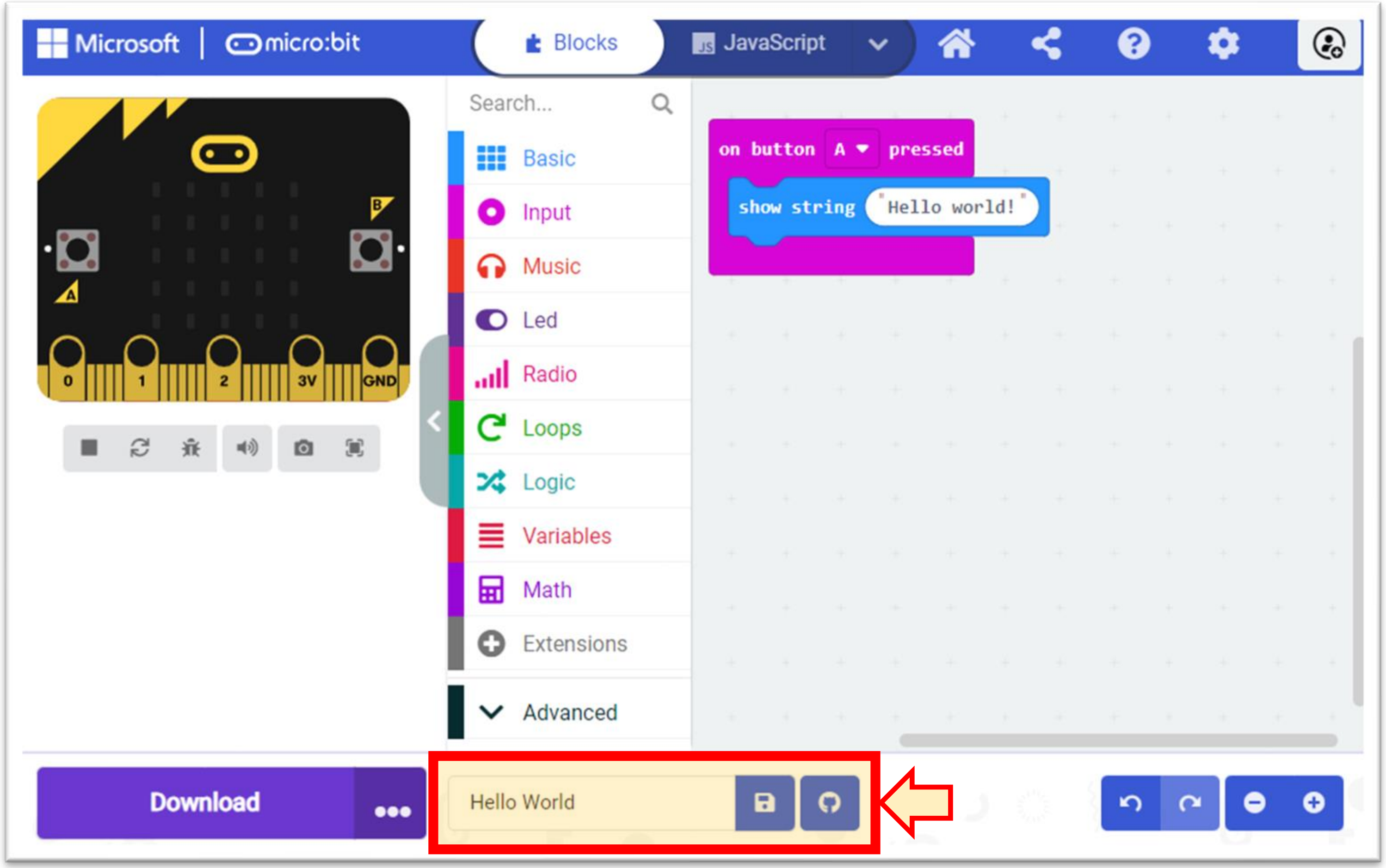
Programming languages

- **Interpreted languages** (e.g., Python): User code and interpreter are copied into the target MCU. Allow users to program the micro:bit 'live' over USB.
- **Compiled languages** (e.g., C/C++): User code is compiled to ARM assembly.
- **Editors** (e.g., MakeCode): High-level programming using blocks.

Microsoft MakeCode

- Barriers still exist for inexperienced users of embedded devices.
- MakeCode is a **web app** that provides non-expert programmers with an Integrated Development Environment (IDE) for embedded systems.
- MakeCode uses **drag-and-drop** visual **block-based** programming with in-browser **compilation** and device **simulation** before transferring programs (hex files) to the physical device.
- Programs can be authored/translated in TypeScript (superset of Javascript) and Python. Blocks and Python are converted to TypeScript before being compiled to lower-level languages.

Microsoft MakeCode

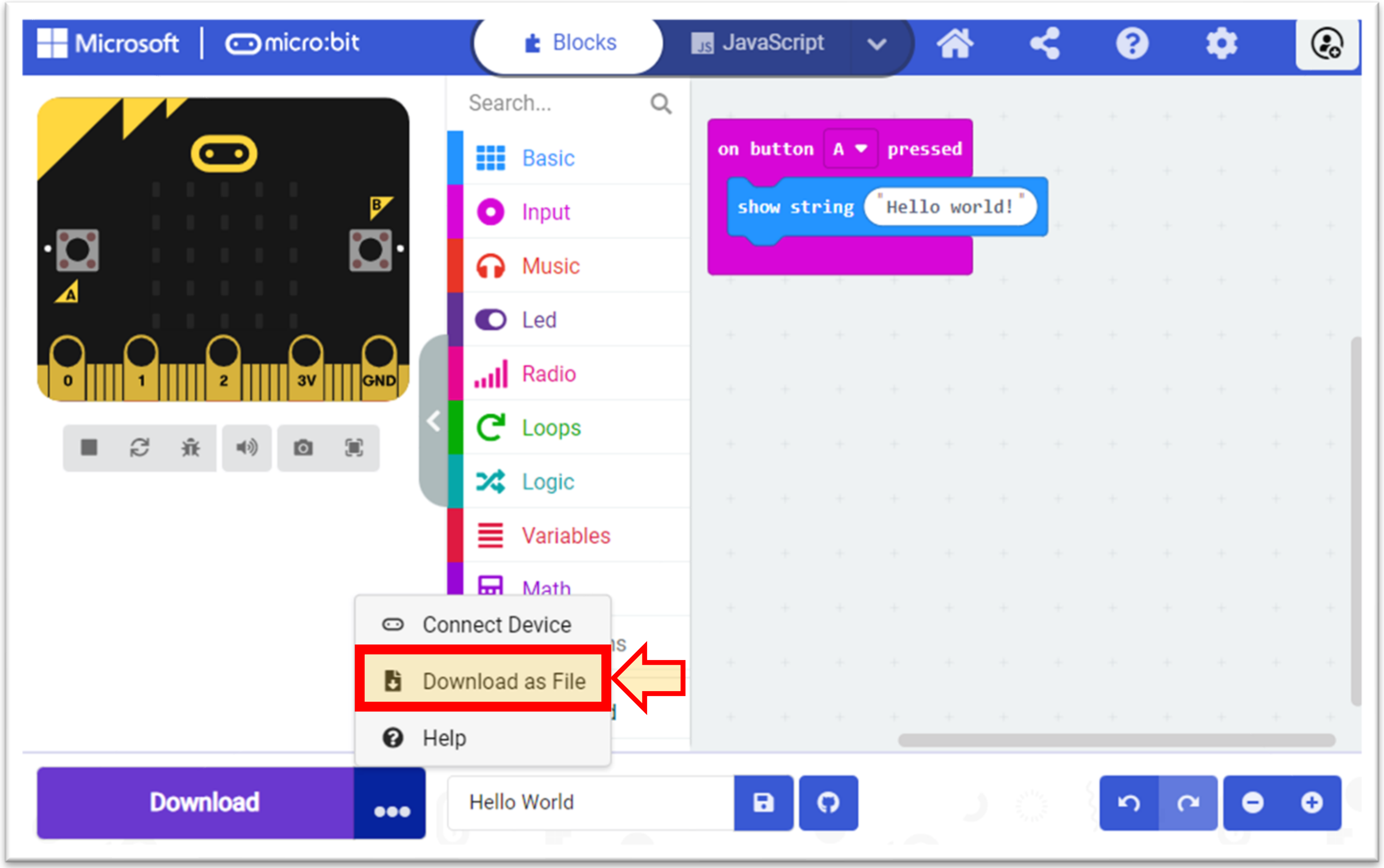


```
on button A pressed
    show string "Hello world!"
```

For example, go to: https://makecode.microbit.org/ and create a new project with title **Hello World**.

Copy the **blocks** shown in the screenshot.

Microsoft MakeCode



```
on button A pressed
    show string "Hello world!"
```

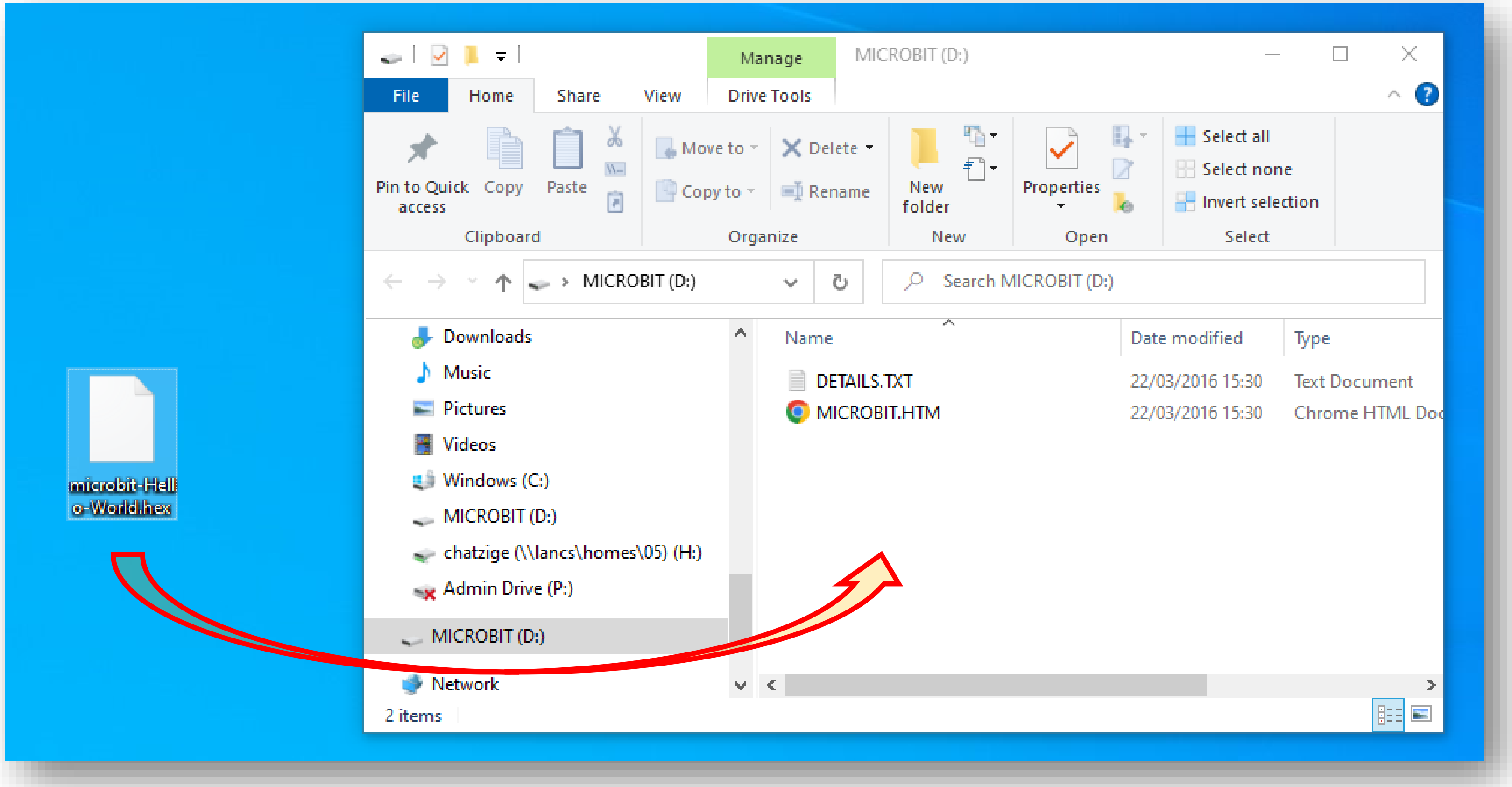
Click on "…"

and select

"Download as File"

to download the **hex** file `microbit-Hello-World.hex`

Microsoft MakeCode



Name	Date modified	Type
DETAILS.TXT	22/03/2016 15:30	Text Document
MICROBIT.HTM	22/03/2016 15:30	Chrome HTML Doc

- Use a **USB cable** to connect micro:bit to your PC.
- Move the downloaded hex file from your PC to the **USB disk** called MICROBIT.

Microsoft MakeCode example

Demonstration of code development using MakeCode

Summary

Today we learnt:

- What the **key hardware components** of micro:bit are and how they interact.
- What the functions of the **target MCU** and the **interface MCU** are.
- What the role of the **interface firmware** (DAPLink) is.
- How the **runtime environment** (CODAL) facilitates code development.
- What the pros and cons of compiled code and interpreted code are when it is flashed to the micro:bit.
- How micro:bit can be programmed using **MakeCode**.

Resources

- Hardware overview: https://tech.microbit.org/hardware
- DAPLink USB interface:
  - https://tech.microbit.org/software/daplink-interface
  - https://microbit.org/get-started/user-guide/firmware
- Software overview: https://tech.microbit.org/software
- CODAL runtime: https://tech.microbit.org/software/runtime
- MakeCode:
  - https://tech.microbit.org/software/makecode
  - https://makecode.microbit.org/