Introduction to the Micro:bit

Lab IoT

Philipp H. Kindt

Assistant Professorship for Pervasive Computing Systems (PCS)
TU Chemnitz

November 11, 2021

The BBC Micro:bit

- Initially designed by the Brithish Broadcasting Corperation (BBC) for educational purposes, i.e., teaching children how to program.
- Built upon the commonly used Nordic nRF52833 SoC.
- In addition to the nRF52 SoC, the following hardware is on the board. Programmer/debugger (CMSIS-DAP), 5x5 LED matrix, speaker, microphone, accelerometer, 2 pushbuttons, 1 capacitive button, expansion connector, USB jack,...
- Various extensions available.
- In this course, we "abuse" this platform to study the low-level programming of embedded systems and IoT nodes, which is not the intended purpose of this board.

A First Look on the Micro:bit

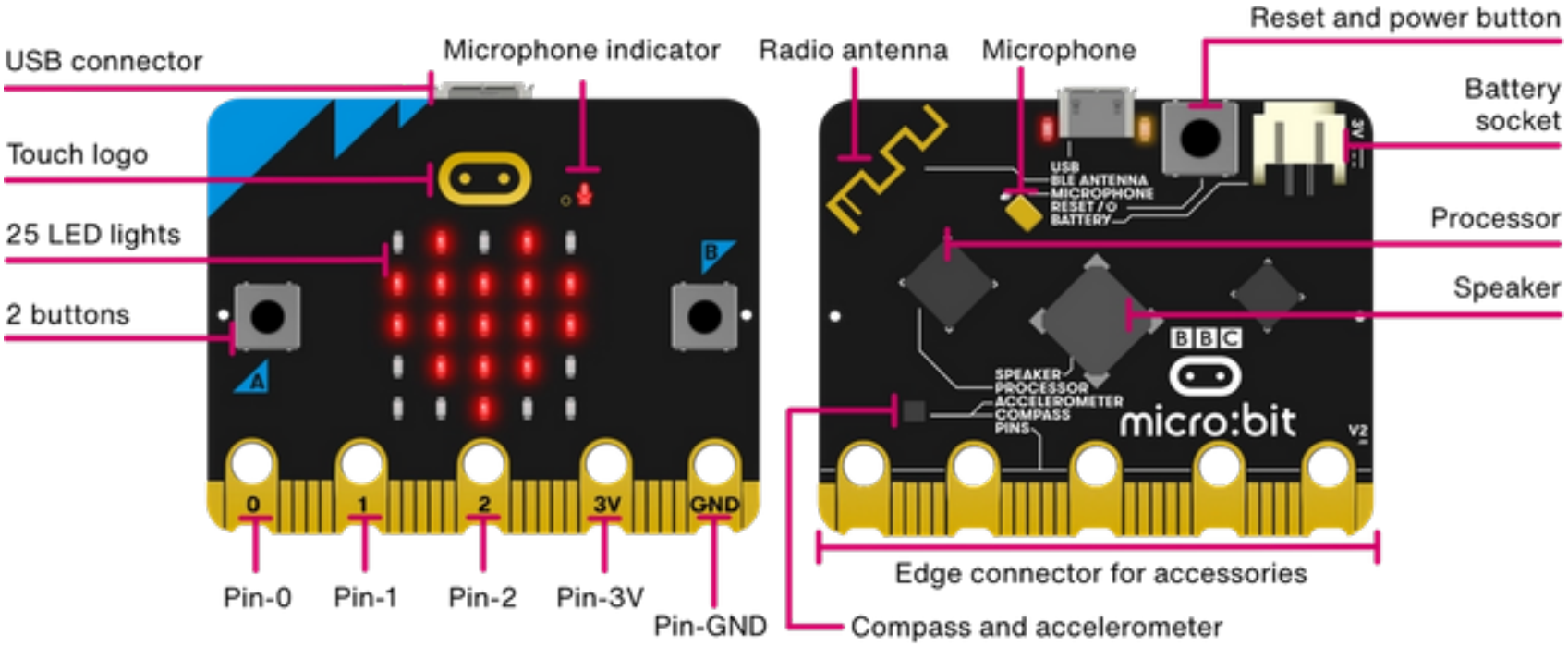
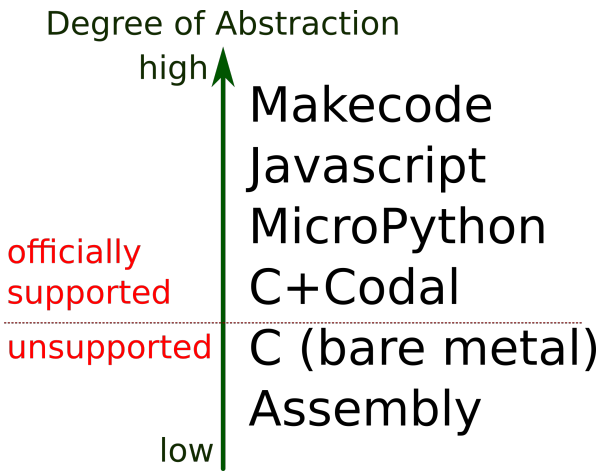


Image source:Micro:bit Educational Foundation, `https://microbit.org/get-started/user-guide/overview/`

The nRF52833 SoC

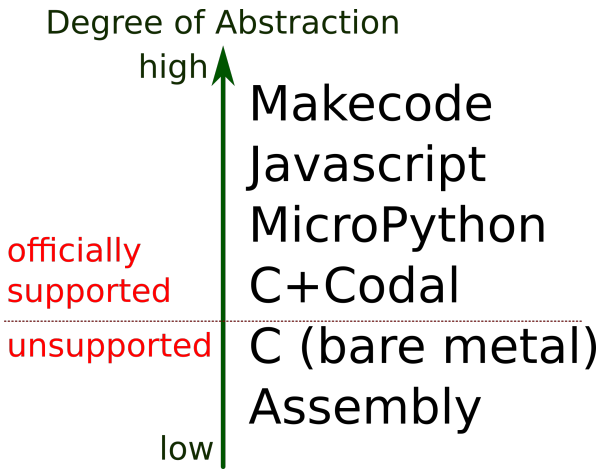
- Full featured ARM Cortex M4 CPU with a 64 MHz clock frequency.
- 512 kB of flash (i.e., non-volatile memory), 128 kB of RAM (i.e., volatile memory).
- Features an USB controller, SPI, UART, I2C, PWM, I2S, NFC, etc.
- 1.7B to 5.5.V supply voltage, integrated DC-DC converter.
- On-chip 2.4 GHz radio designed for Bluetooth 5 and IEEE 802.15.4.
- Intended for applications such as smart homes, fitness, wearables, localization, asset tracking, etc.
- Costs about 3.5 EUR / 3.5 USD (including VAT).

6 Different Ways of Programming the Micro:bit



- **Graphical:** Using Makecode (`https://makecode.microbit.org`)
- **JavaScript:** MakeCode can convert between graphical blocks and JS (`https://makecode.microbit.org`)
- **Python:** The Micro:bit can run a MicroPython interpreter. An online editor is available under `https://python.microbit.org/v/2`).

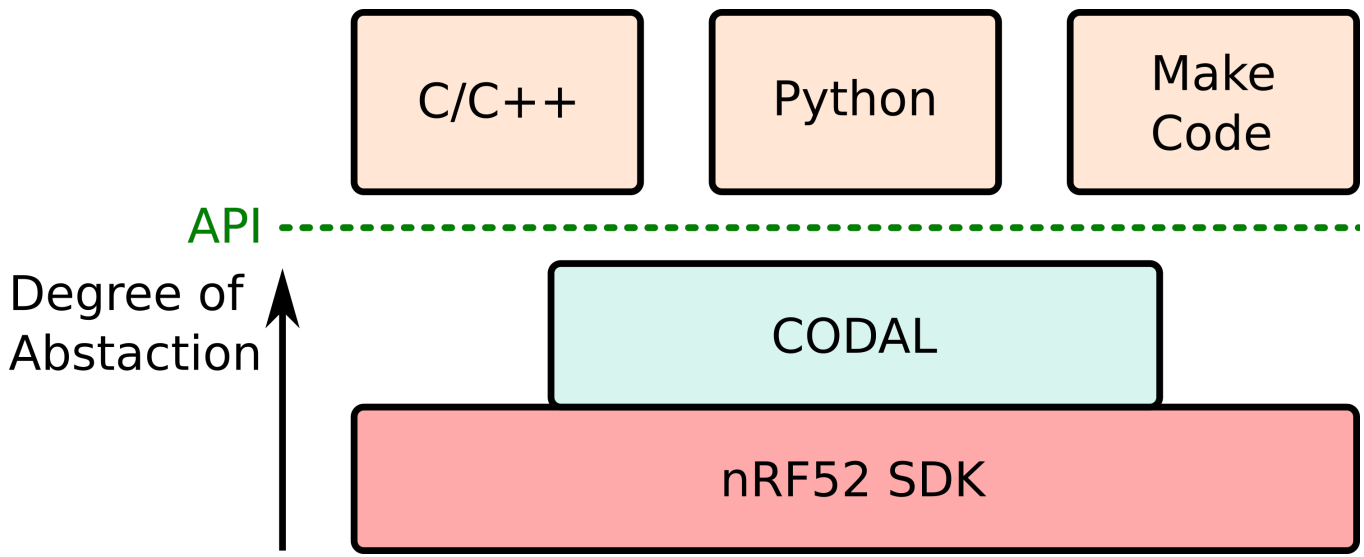
6 Different Ways of Programming the Micro:bit (2)



- **C + Codal**: Codal is a high-level hardware abstraction layer that provides convenient functions, i.e., for creating a scrolling text with one line of code. Contains a small operating system (scheduler, etc.).
- **C + nRF52 SDK**: The nRF52 SDK provides a lower-level, bare-metal API to access (essentially) all functionalities of the chip. It does not support off-chip peripherals on the Micro:bit board.
- **Assembly**: The most low-level way...

CODAL

- CODAL is a C-library, which can be used to program the Micro:bit in C and C++ with low effort.
- It has been developed by Lancester University
- It forms the foundation underlying the Micro:bit ports of Makecode + MicroPython



CODAL Example

Sample code shipped by Lancester University:

```
#include "MicroBit.h"

MicroBit uBit;

int
main()
{
    uBit.init();

    while(1)
        uBit.display.scroll("HELLO␣WORLD!");
}
```

nRF52 SDK

- Comprehensive hardware driver/abstraction C-library by Nordic Semiconductors
- Significantly more comprehensive, complex and low-level than CODAL
- CODAL itself builds upon the nRF52 SDK
- The nRF52 SDK provides different levels of abstraction for controlling the same functionality, e.g., C-Macros for accessing the registers of the UART vs. convenience functions that control an entire UART transmission.
- Documentation: `https://infocenter.nordicsemi.com/topic/struct_sdk/struct/sdk_nrf5_latest.html`

Programming High- vs Low Level

- Programming on different layers of abstractions has different properties.

Property	Low-Level	High-Level
Development Effort	higher	lower
Insights	more	less
Universality	higher	lower
Perfromance	potentially higher	depends
Risk of Bugs	higher	lower
Effort to Setup Toolchain	higher	lower

This Lab

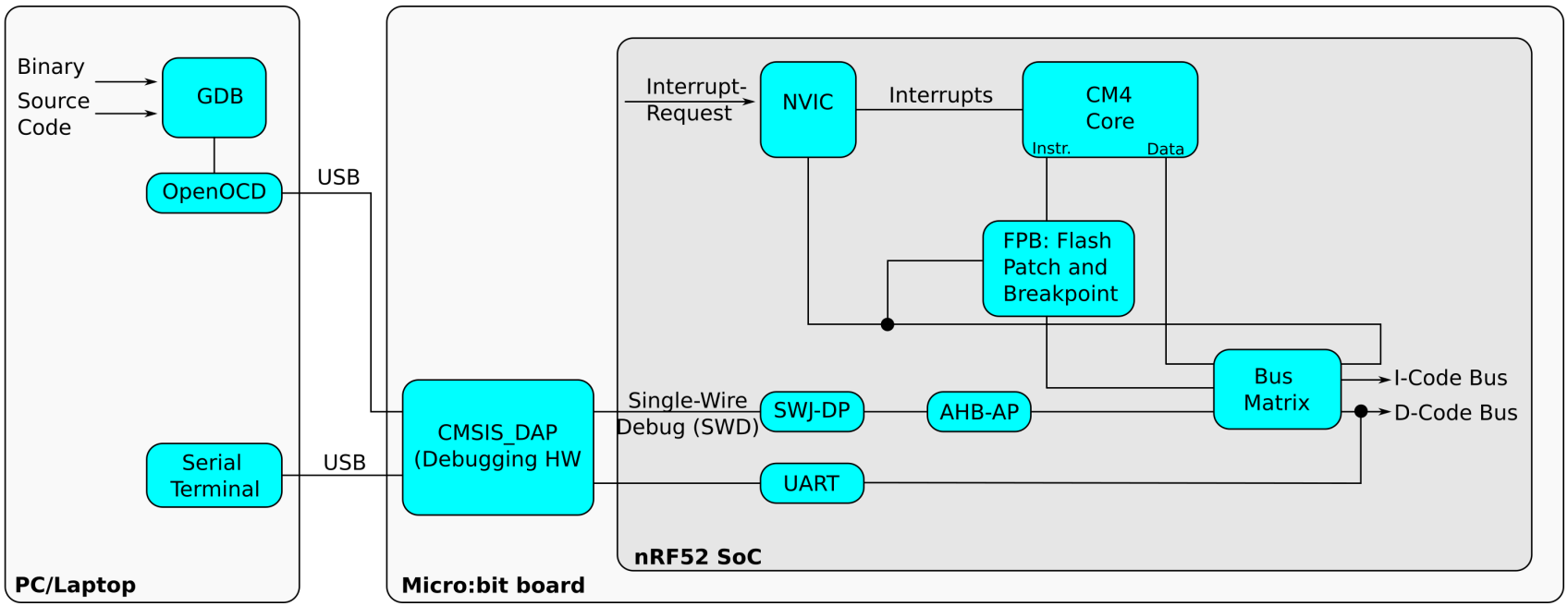
- In this lab, we aim to maximize the insight and hence program the nRF52 SoC using C on bare metal, as well as assembly.
- We use the nRF52 SDK, but avoid the "higher" layers of abstractions provided by it (e.g.., the convenience functions).
- When developing software, e.g., in a company, the usual way to be productive is using available abstractions, while being able to do modifications or extensions on a lower level.
- Also using abstractions often nevertheless requires a good understanding of the underlying system.

HW/SW Tools for Programming the Microbit (2)

- On the PC, the *GNU Debugger (GDB)* provides functionality to flash and debug any program code.
- While GDB is generic for different architectures and microcontrollers, *OpenOCD* provides device-specific access to the programmer hardware (which is often called *debugger*, *JTAG/SWD adapter* or *probe*). Besides accessing it through GDB, OpenOCD can also be used directly, e.g., for flashing.
- *CMSIS-DAP* is a debugger, which is connected to the PC via USB. It "translates" the data received via USB into the *Serial-Wire Debug (SWD)* interface.
- *SWJ-DP*, *AHB-AP* and the *FPB* are hardware blocks that interpret the data received over SWD and trigger the appropriate actions on the SoC.
- CMSIS-DAP also provides a UART-to-USB converter. Hence, the program running on the nRF52 SoC can communicate with programs on the PC, e.g., a serial terminal. Like this, functions such as `printf()` are supported.

HW/SW Tools for Programming the Micro:bit (2)

The following blocks to program and debug the microcontroller are distributed on your PC/laptop, the Micro:bit board and the nRF52 SoC.



Resources

- The material provided in this course provides most of the necessary information. Nevertheless, you might need to obtain some information from additional resources.
- We will use the nRF52 SDK in our experiments to control the nRF52 SoC. All functions and macros are listed in its documentation (see next slides).
- The nRF52 SDK essentially provides macros for register addresses. Utilizing them requires a good understanding of the hardware itself. The *nRF52 product specification* contains a detailed description of the hardware.
- Some (very few) functionality, e.g., timers and functions to access the radio, stem from an open-source project called *blessed*. These functions are not documented, but can be easily learned from examining the blessed source code.
- Assembly programming requires an understanding of the ARM architecture, for which we list suitable documents in the next slides.

Resources (2)

The following documents might be helpful for this lab.

ARM Architecture and Assembly (essential):

1. ARM, Limited. Thumb 16-bit Instruction Set Quick Reference Card, `https://developer.arm.com/documentation/qrc0006/e`
2. ARM Limited. ARM v7-M Architecture Reference Manual, 2014, `https://developer.arm.com/documentation/ddi0403/eb/`
3. ARM Limited. ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition, 2011, `https://developer.arm.com/documentation/ddi0406/latest`
4. ARM Limited. Cortex-M4 Technical Reference Manual, Revision r0p1, 2020, `https://developer.arm.com/documentation/100166/0001/`
5. ARM Limited. Procedure Call Standard for the Arm Architecture. Release 2020Q2, 2020, `https://developer.arm.com/documentation/ihi0042/latest`

Resources (3)

ARM Architecture and Assembly (further reading):

1. The University of South Wales, Digital Systems Laboratory: An Introduction to the GNU Assembly. `http://www.cse.unsw.edu.au/~cs3221/labs/assembler-intro.pdf`
2. Joseph Yiu. The Definitive Guide to the ARM Cortex-M0, Elsevier 2011, ISBN: 978-0-12-385477-3
3. Vincent Mahout. Assembly Language Programming: ARM Cortex-M3, ISTE Ltd/ John Wiley & Sons, Inc., 2012, ISBN 978-1-84821-329-6
4. Joseph Yiu. ARM Cortex-M for Beginners An overview of the ARM Cortex-M processor family and comparison. ARM Limited, 2017

Resources (4)

nRF52 SoC:

1. Nordic Semiconductors ASA: nRF52833 Product Specification v1.4, 2020, `https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps_nrf52833%2Fkeyfeatures_html5.html`
2. Nordic Semiconductors ASA: nRF5 SDK v17.0.2 Documentation, 2020, `https://infocenter.nordicsemi.com/index.jsp?topic=%2Fsdk_nrf5_v17.0.2%2Findex.html`
3. Paulo Borges: Bluetooth Low Energy Software Stack for Embedded Devices (Blessed), `https://github.com/pauloborges/blessed`