~~Lab 3~~: BLE Software

Introduction

The purpose of today's lab is to show you how to write software that uses BLE communication and run it on the nRF52840DK. We'll focus on modifying existing application examples to meet our goals, rather than trying to create them from scratch.

To enable this, we'll have to do yet more toolchain installation, which can be finicky and confusing at times, but once it works, you'll have the capability to write your own programs for the nRF52840DK, which will likely be useful for the final project.

Let me know if you run into problems and I will help you debug!

Goals

- Set up nRF development toolchain on your system
- Send and receive BLE Advertisements
- Create some simple BLE services on a device for Connections

Equipment

- Computer
- nRF52840DK + USB cable
- Smartphone (optional)

Partners

- ~~This lab should be done in small groups. At least two are needed, but up to four students would be fine.~~
- N/A: see submission below

Submission

- Nothing! Given the current week of the quarter, I'd rather have you all spend time on your projects.
- If you're doing BLE stuff for a project with the nRF52840DK, this will be useful. If not, you don't have to go through these steps at all.

1. Setting up a programming environment

This is the hard part. Getting an environment set up is kind of a pain. Advance warning: for those of you on M1 (ARM) MacOS, it's even more painful.

Three different approaches (in order of how much I recommend them) pick ONE of these:

- **VSCode** (straight-up not supported on M1 MacOS right now)
  This is relatively new still, but seems to work great on Windows, Intel MacOS, and Linux.

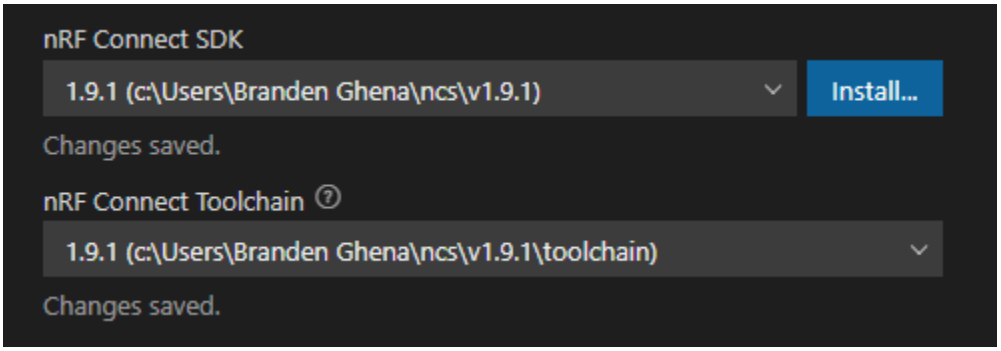
- **Segger Embedded Studio**
  This is the classic way of handling this stuff. Also works pretty well, although it is certainly not the world's best IDE. This has successfully worked on M1 MacOS.

- **Command Line**
  You can set up Linux (maybe in a VM) or use MacOS or use WSL and just install the build system yourself.

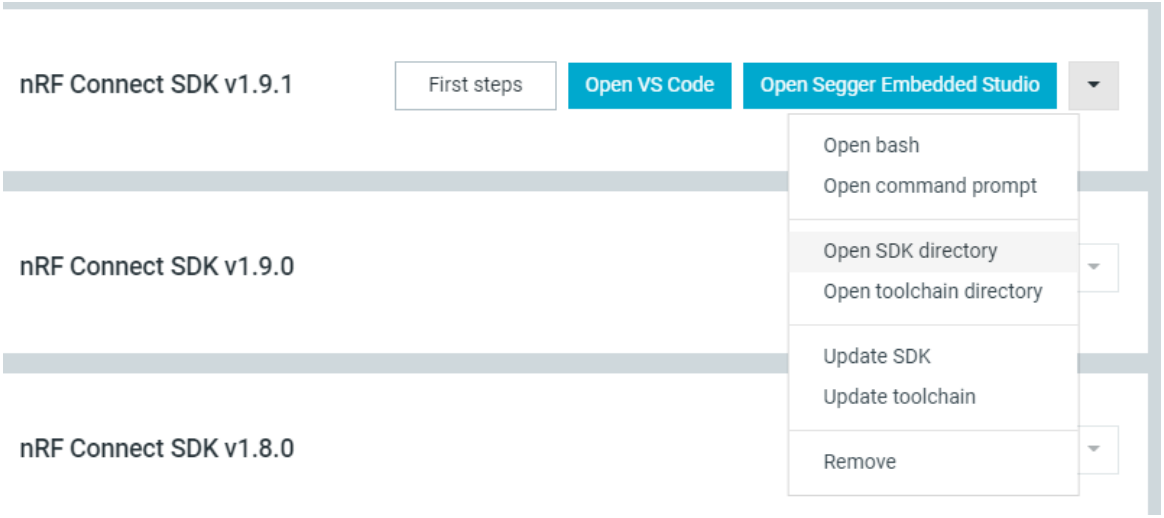
1.1. VSCode

- Follow the instructions here: https://nrfconnect.github.io/vscode-nrf-connect/connect/install.html
    - Toolchain Manager is an app inside of nRF Connect
    - **Important:** the folder where you install the toolchain MUST NOT have spaces anywhere in the path to it. So probably make a folder in your root directory called “nrf_toolchain” or something like that. Using your home directory could fail if your username has a space in it! (like mine did on Windows)
    - You’ll have to restart nRF Connect after installing VSCode for it to find it

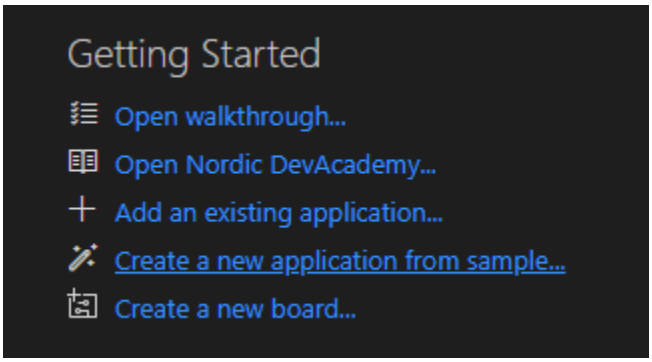
- When opening VSCode from Toolchain Manager, you should see that the home screen says “nRF Connect for VS Code”. If it didn’t find “nRF Connect SDK” and “nRF Connect Toolchain”, you’ll need to browse to the paths.
    - When I set mine up on Windows, it looked like:



    - If you need them, Toolchain Manager can show you what the paths are. The down arrow on the right has an option to open the directories.

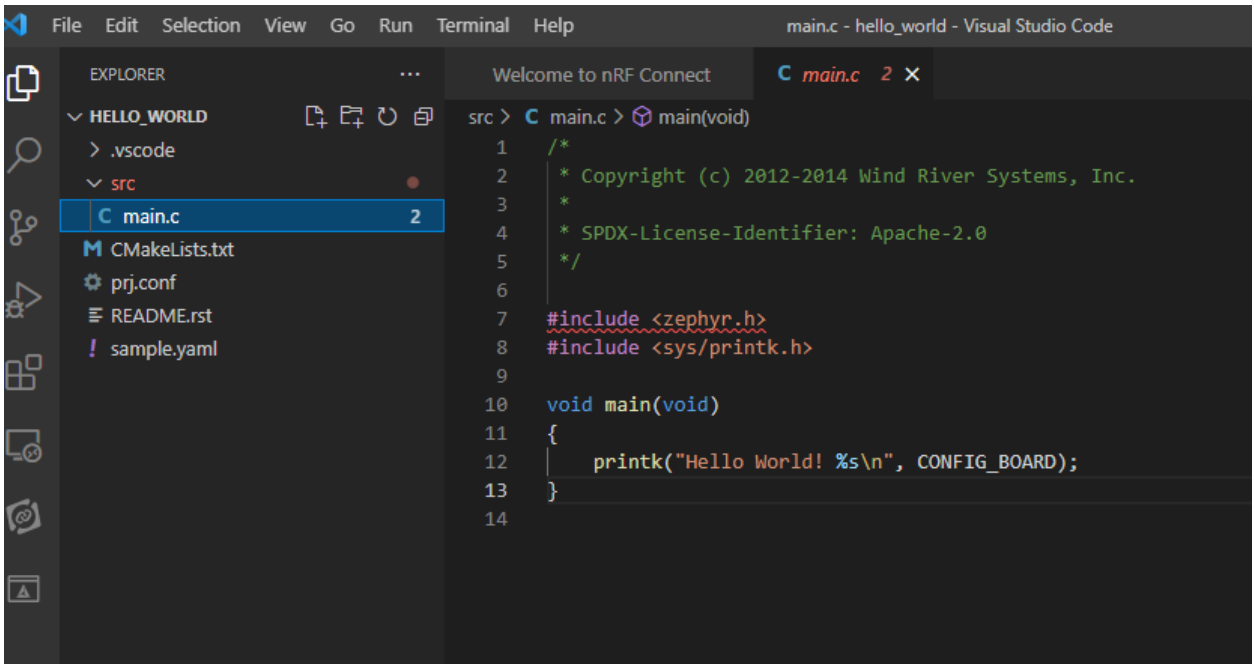


- Create a new example application.
    - You should now be able to select “Create a new application from sample…” from that home screen.



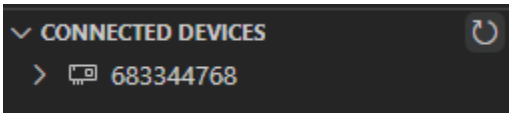
    - Choose the “Freestanding” radio button
    - You will have to choose an “Application Location” for it to place your applications in. **Important:** this folder MUST NOT have spaces anywhere in the path to it. So probably make a folder in your root directory called “nrf_apps” or something like that.
    - For “Application template” choose “zephyr/samples/hello_world”

- You should now have the project open. That should get you something like this:



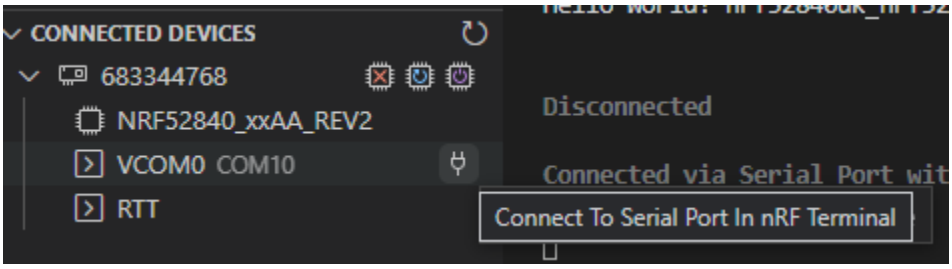
- Create Build Configurations for your new application
    - On the far left of VSCode, one of the installed apps will be “nRF Connect”, click it
    - Then under “Applications”/”hello_world”, click “No build configurations”
    - Under Board, choose “nRF52840dk_nrf52840” and click “Build Configuration”
    - That’ll take a hot minute to run

- Flash the application
  - Connect your nRF52840DK to the computer
  - On the far left of VSCode, choose the “nRF Connect” app
    Now, the left window will have some sections: “Welcome”, “Applications”, etc.
  - Under “Connected Devices”, make sure a device is there
    - You may have to click the “Refresh symbol”



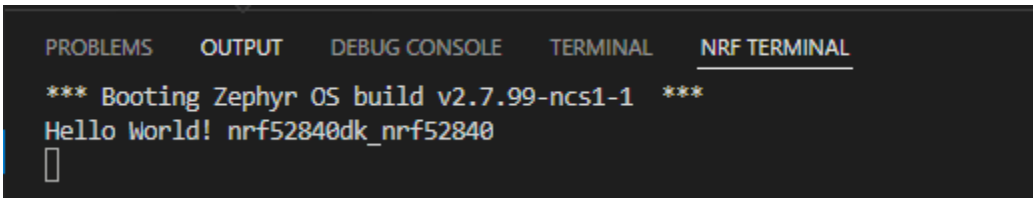
  - Then under “Actions”, click Flash
  - That will take a second to run

- Open a terminal to the board
  - On the far left side of VSCode, choose the “nRF Connect” app
  - Under the “Connected Devices” section, choose the dropdown on your board
  - There should be some kind of serial/virtual COM device under there



  - Click the little plug thing, which will open a “Choose Configuration” at the top of VSCode
  - “115200 8n1 rstcts:off” is correct. Hit enter

  - Now a serial terminal is open, hit the “IF Boot/Reset” button on the nRF52840DK to restart it and display text



    - If it doesn’t work, try flashing the code again

1.2. Segger Embedded Studio (SES)

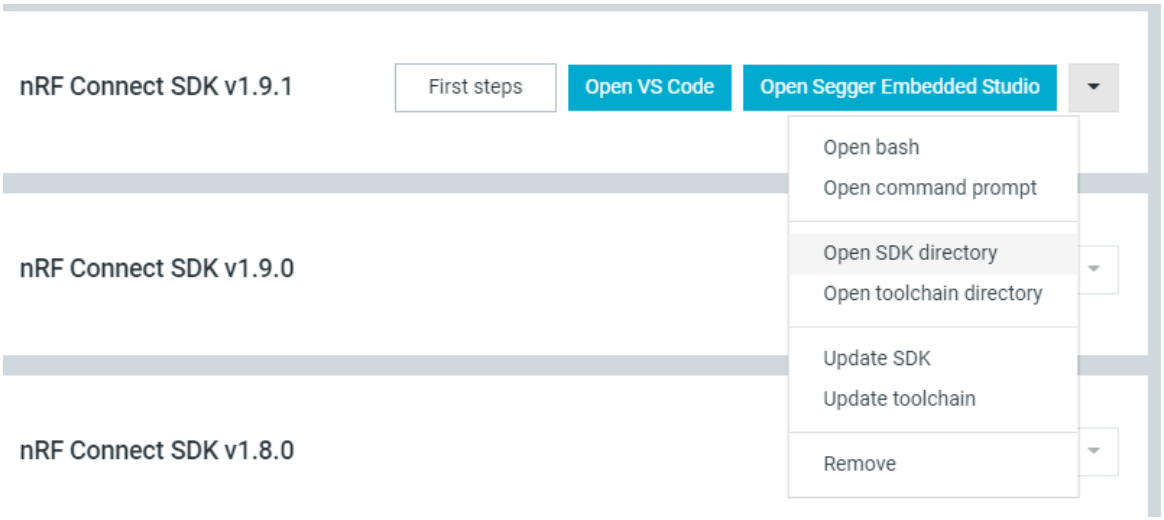
- First, install Segger Embedded Studio: https://www.segger.com/downloads/embedded-studio/
    - You want “Embedded Studio for ARM”

- Then follow the instructions here to use Toolchain Manager: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/gs_assistant.html
    - **Important:** the folder where you install the toolchain MUST NOT have spaces anywhere in the path to it. So probably make a folder in your root directory called “nrf_toolchain” or something like that. Using your home directory could fail if your username has a space in it! (like mine did on Windows)
    - As the last step, click the “Open Segger Embedded Studio” button. You’ll have to restart nRF Connect after installing Segger Embedded Studio for it to find it

- Create a new example application.
    - The “nRF Connect Options” window might pop-up automatically. If it doesn’t, you can find it in File->”Open nRF Connect SDK Project…”

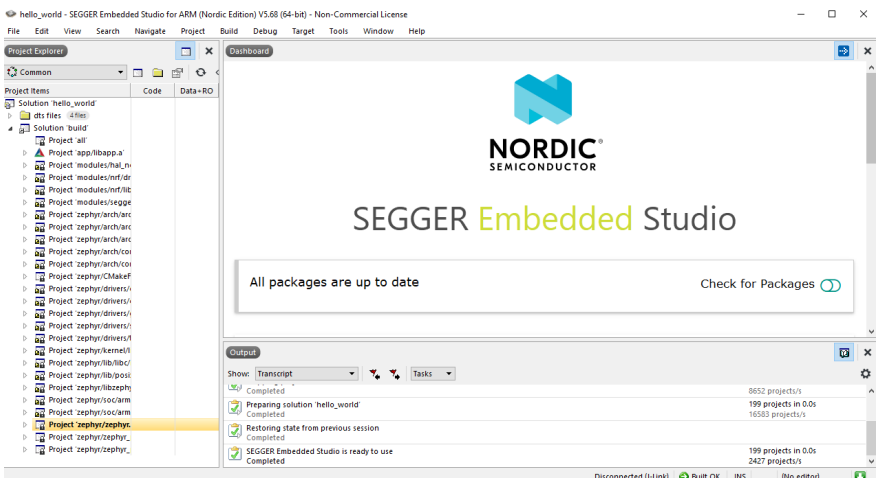
    - The “nRF Connect SDK Release” should be set to the path to your SDK installation.

- If you need them, Toolchain Manager can show you what the paths are. The down arrow on the right has an option to open the directories.



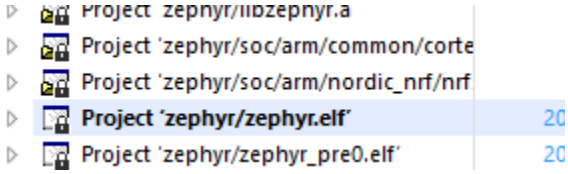
  - The “Projects:” dropdown has a list of example projects. Under the “Getting Started | Zephyr” heading, you want to select “hello_world”.
  - The “Board Name:” dropdown has a list of supported hardware. You MUST select “nRF52840dk_nrf52840”
  - The “Build Directory:” is automatically generated and should be fine. You can always choose a different location if you want.
    - **IMPORTANT:** make sure that path doesn’t have spaces anywhere in it, or everything will break
  - Click OK to create a new project.

- You should now have the project open. That should look like this:



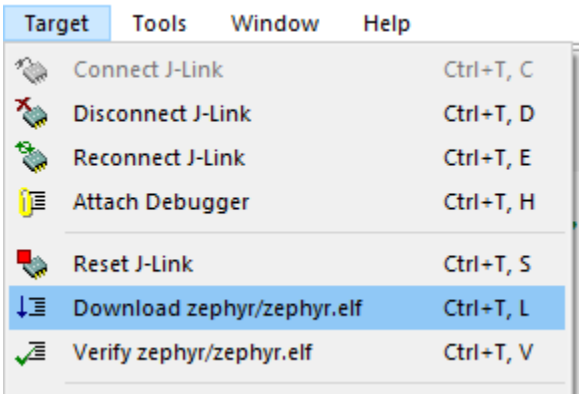
- The C code for the project can be found under the “Project Explorer” pane on the left. Select the down arrow on “Project app/libapp.a”, then on “C_COMPILER__app_”.

- Build the project
  - In the menubar at top, select Build->”Build Solution”
  - That should make some compilation steps appear for several seconds in the bottom “Output” pane. They should all come up as “OK”.
  - If you get an error about no executable, make sure you double click “Project zephyr/zephyr.elf” in the left “Project Explorer” pane. It should be bolded.



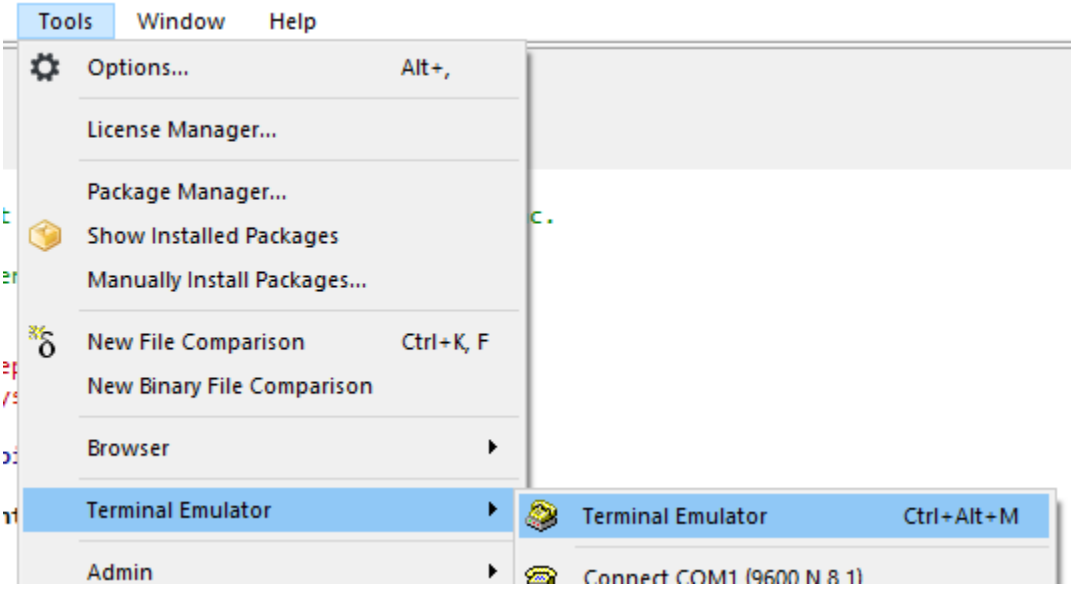
- Connect a board and flash the application
  - Plug in your board over USB
  - To connect, in the menubar at top, select Target->”Connect JLink”
    - If you get a pop-up about selecting an Emulator, that means it couldn’t find your board. Unplug/replug? Maybe something else is connected to it?
    - If it works, nothing will pop up, but the bottom of the IDE will say “CortexM4 on J-Link”

- To flash, in the menubar at top, select Target->"Download zephyr/zephyr.elf"

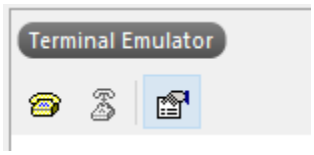


  - That should make a pop-up with a loading bar that fills with green pretty quickly. When it's finished, the code is on your board!
  - If it doesn't work, try disconnecting and reconnecting. Also try unplug/replug on your board. These things can be finicky.

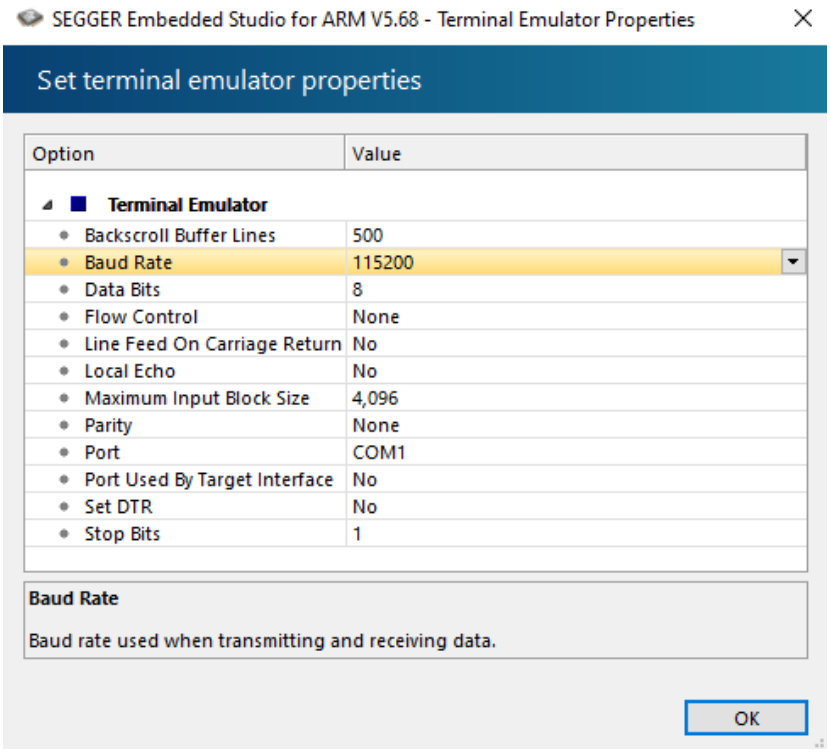
- Open a terminal to the board
  - In the menubar at top, select Tools->"Terminal Emulator"->"Terminal Emulator"



  - That should open a new bottom pane called "Terminal Emulator"
  - Open the Terminal Emulator properties which are the icon of a little hand pointing at a card.

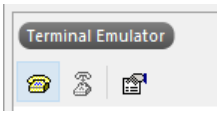


- Edit the terminal properties
  - Set “Baud Rate” to 115200
  - Set “Port” to the serial port for your board
    - On Windows likely COM1
    - On MacOS something like `/dev/tty.usbmodemxxxx` where the x’s might be numbers
      - You can figure it out by opening a terminal and running `ls /dev/tty*`

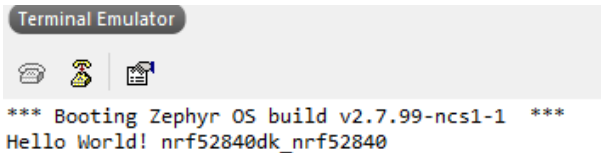


  - Click OK

- Open the terminal to your board by clicking the old-as-heck telephone button.



- Hit the reset button on your board labeled “IF BOOT/RESET” to restart it and display output



  - If it doesn’t work, try flashing the code again.

1.3. Command Line

- If you want some instructions for creating a Virtual Machine, here are some:
  Making an Ubuntu VM
    - This is very unlikely to work on M1 MacOS as many VMs don't support M1 and would need an ARM version of Ubuntu even if they did, which might not support all of the applications we need.

- Follow the instructions here:
  https://docs.zephyrproject.org/latest/develop/getting_started/index.html
    - I haven't run through these myself, so I don't have more specific instructions. Let me know if you run into any problems though, and I can assist!

TASK: None. Continue to the next section.

2. Good example applications

- Zephyr is an embedded operating system that Nordic uses for many example applications.
- Documentation
    - General: https://docs.zephyrproject.org/latest/index.html
    - Bluetooth: https://docs.zephyrproject.org/latest/connectivity/bluetooth/index.html

- Example applications
  All of these can be created in VSCode or SES via the nRF SDK Connect app creation method as detailed above. In VSCode, they are under “zephyr/samples/”. In SES they are just mixed in with the other applications, and not even in alphabetical order.
    - Hello World [“hello_world”]
        - Display some simple text
        - We already had you load this one!
        - https://docs.zephyrproject.org/2.7.0/samples/hello_world/README.html
    - Blinky [“blinky”]
        - Blink an LED on and off
        - https://docs.zephyrproject.org/2.7.0/samples/basic/blinky/README.html
    - Button [“button”]
        - Display a message when a button is pushed
        - https://docs.zephyrproject.org/2.7.0/samples/basic/button/README.html
    - Bluetooth Beacon [“beacon”]
        - Send advertisements with URLs
        - https://docs.zephyrproject.org/2.7.0/samples/bluetooth/beacon/README.html
    - Bluetooth Scan & Advertise [“scan_adv”]
        - Scan for advertisements while also advertising
        - Good starting point for mesh network stuff!
        - https://docs.zephyrproject.org/2.7.0/samples/bluetooth/scan_adv/README.html
    - Bluetooth Central [“central”]
        - Scan for and connect to other BLE peripherals
        - https://docs.zephyrproject.org/2.7.0/samples/bluetooth/central_multilink/README.html
    - Bluetooth Peripheral HIDS [“peripheral_hids”]
        - Generic mouse human-interface-device over BLE
        - With services and characteristics for connections
        - https://docs.zephyrproject.org/2.7.0/samples/bluetooth/peripheral_hids/README.html

TASK: Nothing to submit. These are just for you to use for projects.