# SEUS Embedded Front-End Rev 0.1 Documentation

# Table of Contents

**System Design**……2

**Data Structure and Specifications of Transmitted Packets**……....3

**Headset Usage**……..……...4

**Smartphone Application Usage**……..6

**Changing Batteries**……....8
- **Changing PCB Battery**……..9
- **Changing Microphone Battery**……..11

**Programming the PCB**……....12
- **Programming the STM32f4**……....13
- **Programming the BMD-300 BLE Module**……...18

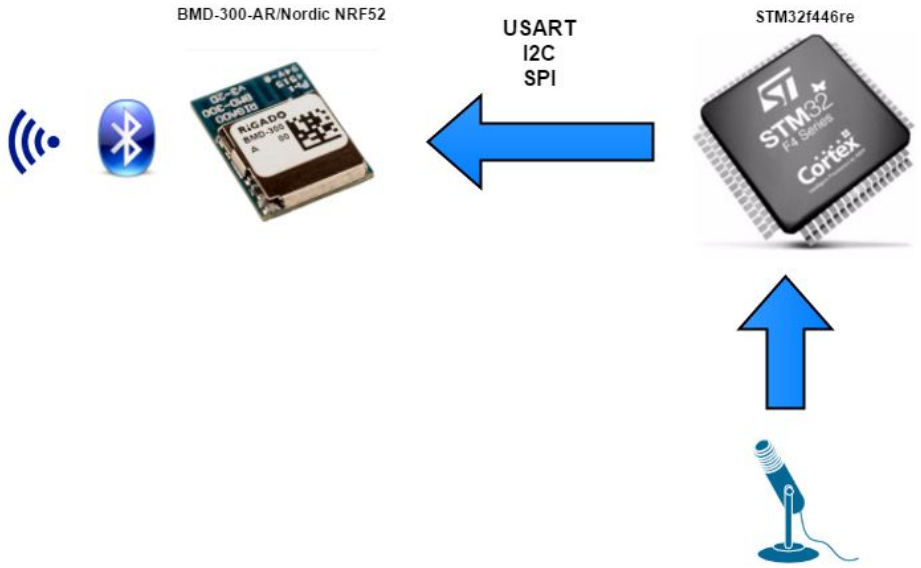
**Layout and Schematics**……..22

# System Design

The primary components of the SEUS embedded front-end system consists of:

- STM32f446re MCU
- BMD-300-AR Bluetooth 4.2 Low Energy Module

The current system flow is shown below:




The STM32f446re takes samples from the microphone inputs and calculates the desired features in determined windows (currently the features are: delay by cross-correlation, relative power, and zero crossing rate). The features are sent over to the BMD-300/NRF52 to send to the smartphone through Bluetooth. The following sections will describe the structure of the data/bytes transmitted from the PCB board to the smartphone application, as well as how to use the headset.

# Data Structure and Specifications of Transmitted Packets

The STM32f4 MCU takes in samples from the connected microphones, extracts features, and currently sends the features over USART to the BMD-300 Bluetooth module for transmission. The features are currently being extracted with the following specifications:

- Number of Microphones being sampled: 4 (the fifth one is fed directly into the phone)
- Sampling Rate: 32 kHz
- Sampling Precision: 8-bits
- Window Size: 100 ms
- Window Overlap: 50% (features are being calculated every 50 ms)

Once the BMD-300 BLE module is connected to the smartphone and receives the features from the STM32f4 it transmits the features to the smartphone in real-time. Each window of features are sent byte-by-byte in the following order:

1. Delimiter (5 bytes): SEUS#
2. Delay Microphone 2 (1 byte)
3. Delay Microphone 3 (1 byte)
4. Delay Microphone 4 (1 byte)
5. Relative Power Microphone 2 (8 bytes): Big Endian format
6. Relative Power Microphone 3 (8 bytes): Big Endian format
7. Relative Power Microphone 4 (8 bytes): Big Endian format
8. Zero-Crossing Rate Microphone 1 (2 bytes): Big Endian format
9. Zero-Crossing Rate Microphone 2 (2 bytes): Big Endian format
10. Zero-Crossing Rate Microphone 3 (2 bytes): Big Endian format
11. Zero-Crossing Rate Microphone 4 (2 bytes): Big Endian format

The delimiter currently used is “SEUS#”. The decimal value for the ASCII character “#” is 35, which represents the number of data bytes being sent over for each window. In other words, the delimiter is more accurately represented as “SEUS” + (number of bytes in the payload); in this current iteration, one window of features are represented as 35 bytes. Additionally, the multi-byte values are being sent in Big Endian format, meaning the most significant byte is sent first and the least significant byte is sent last; it is up to the smartphone application to reconstruct the features from the byte stream.

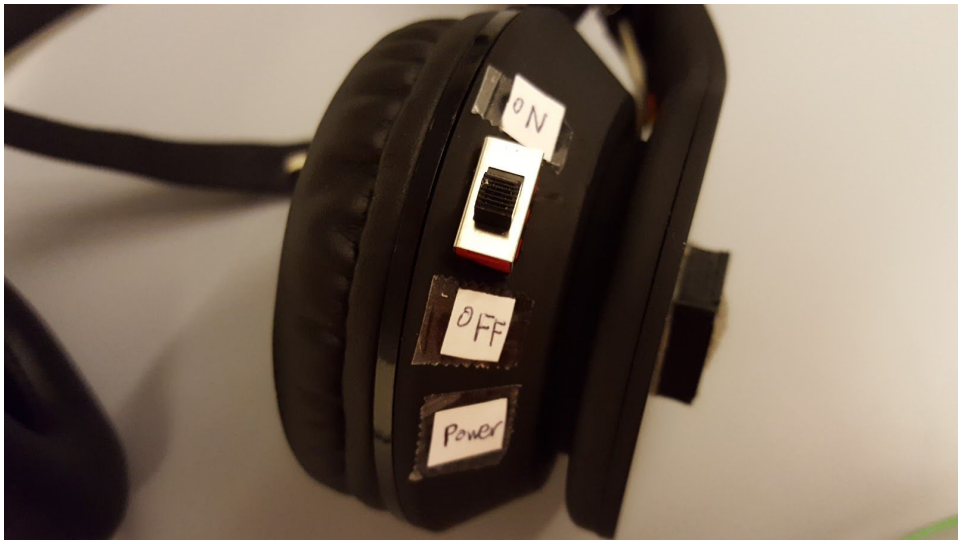
# Headset Usage

This section will go over how to use and maintain the SEUS headset. A full image of the headset is shown below.

What is important to notice are the labels attached to each ear of the headset. A close up of the labels are shown below.


Each ear of the headset has three labels that denote the position of the switch in order for the system to be on or off as well as what is stored in each ear and what each switch controls.

The side labeled “Power” contains the batteries that power the SEUS embedded-front end pcb. As shown in the image below, the system is turned off when the switch is pulled down and powered on when the switch is pulled up.



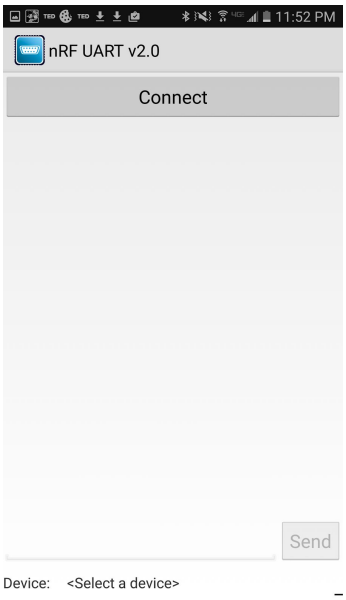

The side that is labeled “Mic + Board” contains the actual pcb itself, along with a separate coin cell battery that powers the four microphones. The switch on this ear turns the microphones on and off. As shown in the image below, the microphones are turned on when the switch is pulled low and turned off when the switch is turned high, which is the opposite from the batteries powering the board.

# Smartphone Application Usage

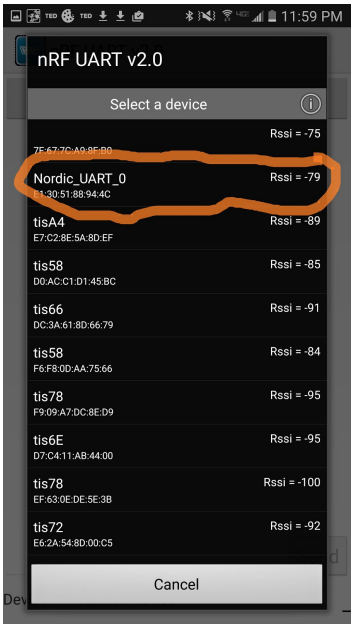
The smartphone application that reads and reconstructs the features received over BLE is a modified version of the “nRF UART App”. The file writing part is similar to what we did last time. The data recorded is stored in 4 files written to a folder called HEADDATA. One file stores all received data (each byte on a new line), the other stores the 3 delays of a window separated by “,” on a new line for every window. Similarly the power and zero crossings are also stored in separate files in a similar format.  You need to create a folder called HEADDATA before running the app. Use any file manager you want. The app, waits for the delimiter and then processes the sequence of bytes in the order in which the features are being sent.

We implemented a rudimentary way of separating and re-constructing the features from the bytes. Please feel free to use better methods of parsing the data. The delimiters and data format are mentioned above which you can use for the same.

When the application starts, the screen will look like the following:




Hit “Connect” and find **Nordic_UART_0**. Connect to “Nordic_UART_0, as shown below.

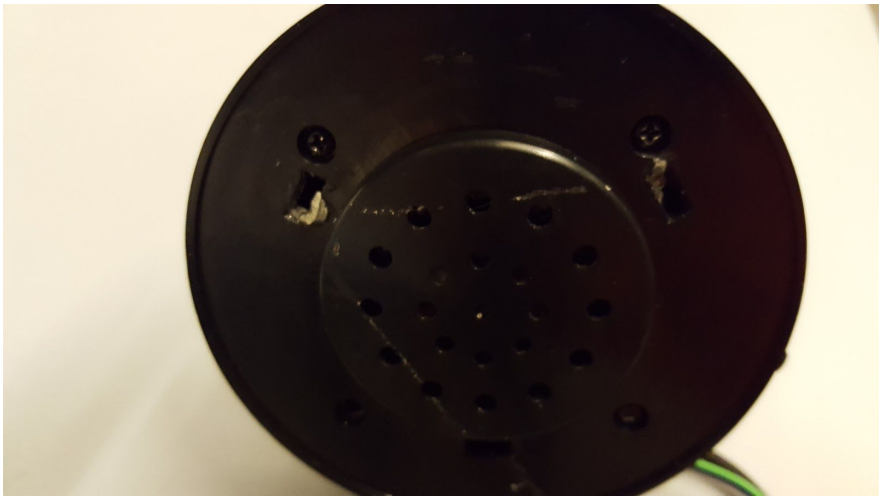



Once connected, the application should begin displaying strings or bytes of received data depending on the version of the application.

# Changing Batteries

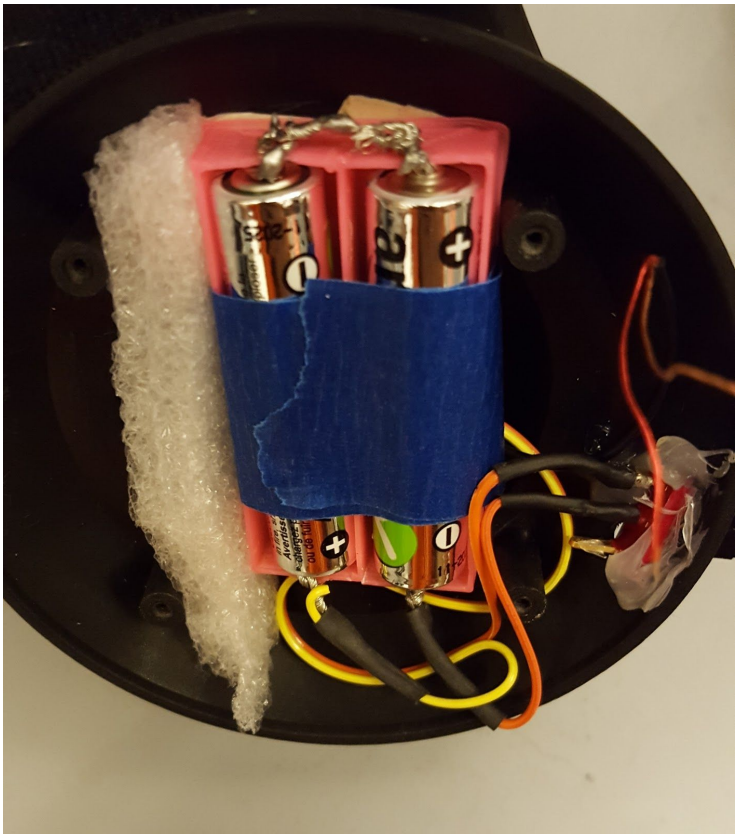
Changing the batteries of either the pcb or the microphones involves opening an ear of the headset. To open up the earphone remove the top layer padding, as shown below.

Once the padding has been removed, there is a cover, shown below, with up to four screws securing it to the headset. Remove the four screws to access the interior, where all of the SEUS hardware is stored.



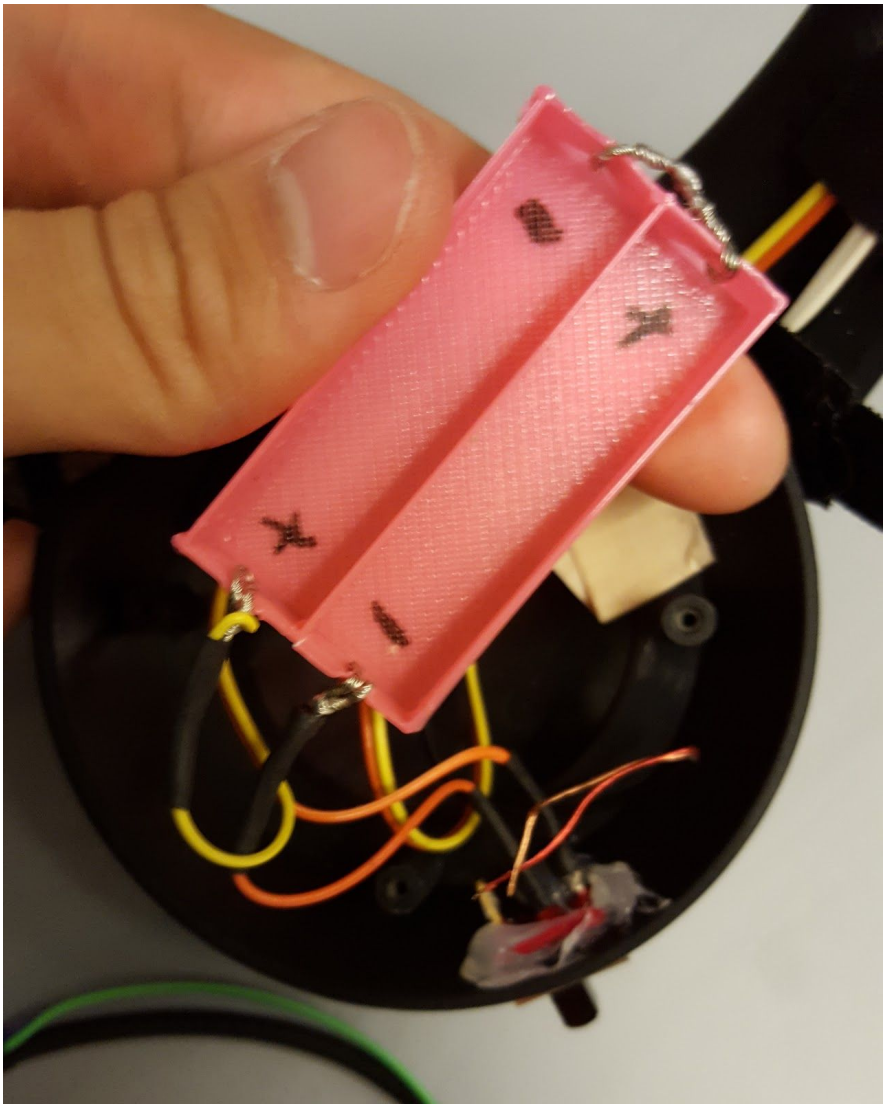
**Changing PCB Battery**

The battery that powers the pcb board is located in the side labeled “Power”. To replace the batteries, remove the cover to reveal a battery case as shown below.



Note that there is tape used to secure the batteries into the 3D printed holder. **Changing the batteries that power the board is a precarious process because the contact points are small and the battery holder is flimsy**. The holder is secured to the headset with double-sided tape on the bottom of the holder.

To change the batteries, carefully remove the tape holding the old batteries in place, and remove the batteries. Once the batteries are removed, new batteries may be placed into the holder with orientations written in the holder as shown below.

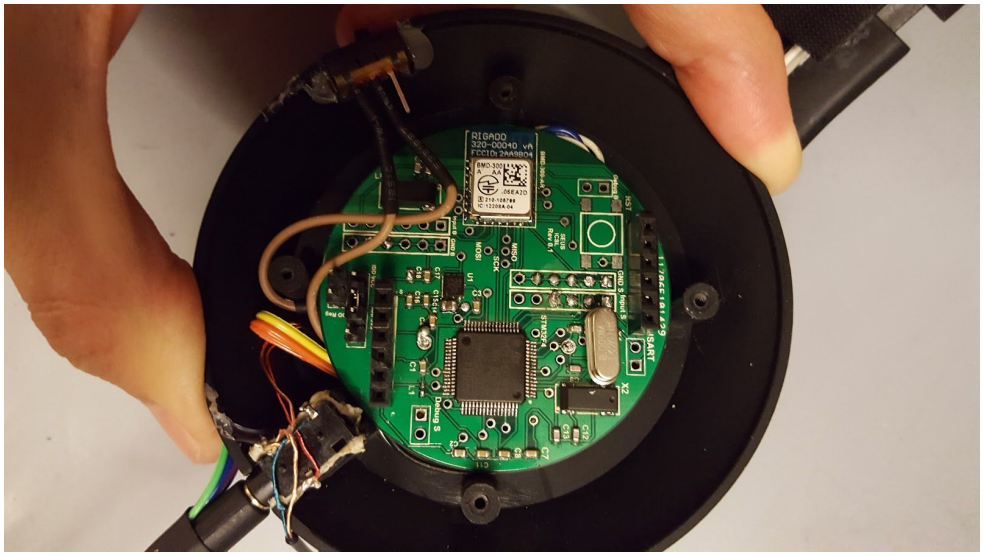


Once the new batteries are placed, **make sure to wrap the batteries into the holder tightly with tape**. The idea is to wrap the tape around holder and batteries so that the contact points of the holder fit tightly around the batteries and that the batteries are securely fastened in place. Once the batteries have been changed, if for some reason the smartphone cannot find the advertisement of the pcb, then the batteries were not properly and tightly placed into the headset.

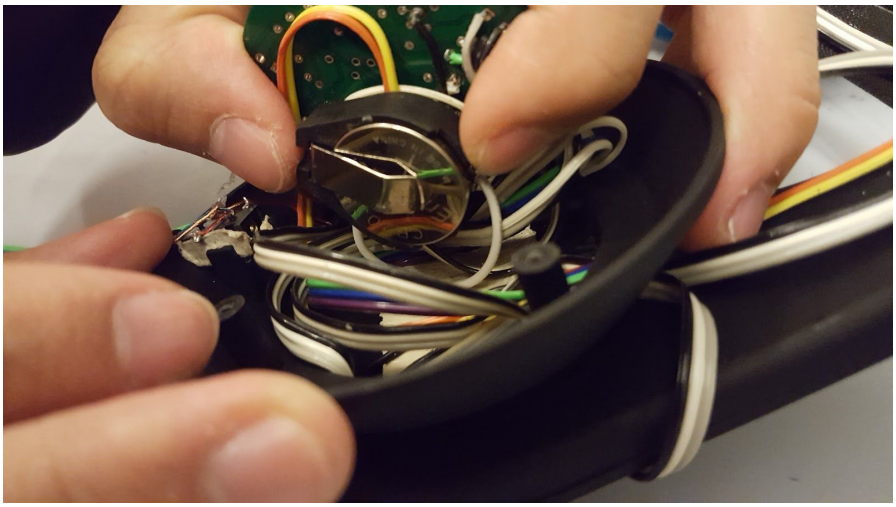
The batteries used to power the board are two triple AAA 1.5 V batteries. With this setup, it has been tested that the system is able to continuously calculate and transmit features for at least three hours.

**Changing Microphone Battery**

The battery for the microphone are stored on the same side of the headset as the pcb. To change the microphone battery, remove the cover of the side labeled “Mic + Board”. The inside is shown below.



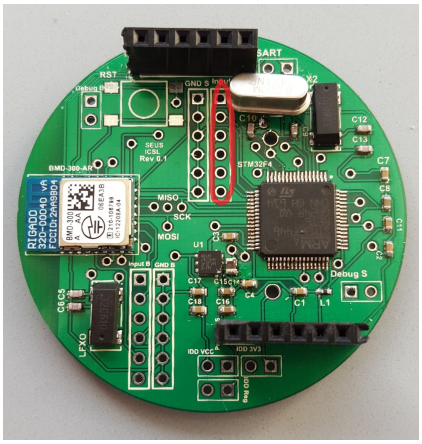
The battery for the microphones is underneath the board, so lift up the board to change the battery, as shown below. The battery used to power the microphone is a standard 20 mm 3V CR2032 coin cell.



Though there has not been any tests done to characterize how long the microphones can run off of one coin cell battery, it is expected that the battery of the microphones will last a much shorter time than the batteries powering the pcb.

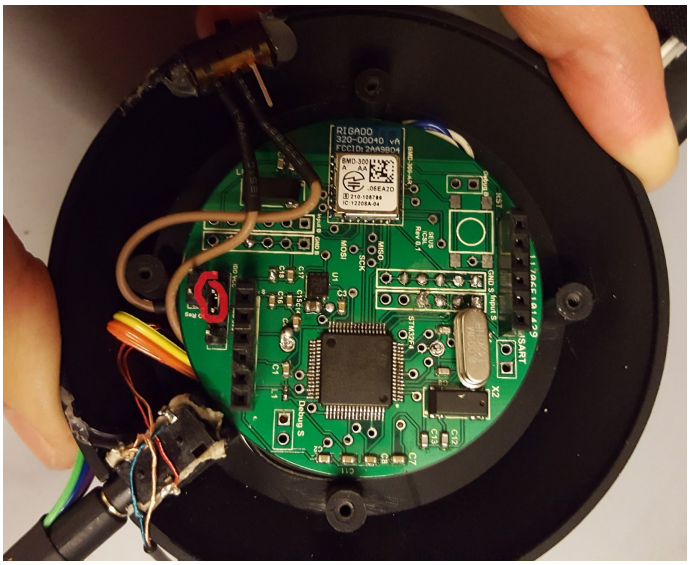
# Programming the PCB

There may come a time when the firmware of the modules on board may require updates. The following section details how to program both modules with provided .hex or .bin files. An image of the pcb is shown below. Additionally, the schematics and layout of the pcb are appended at the end of this document for reference.

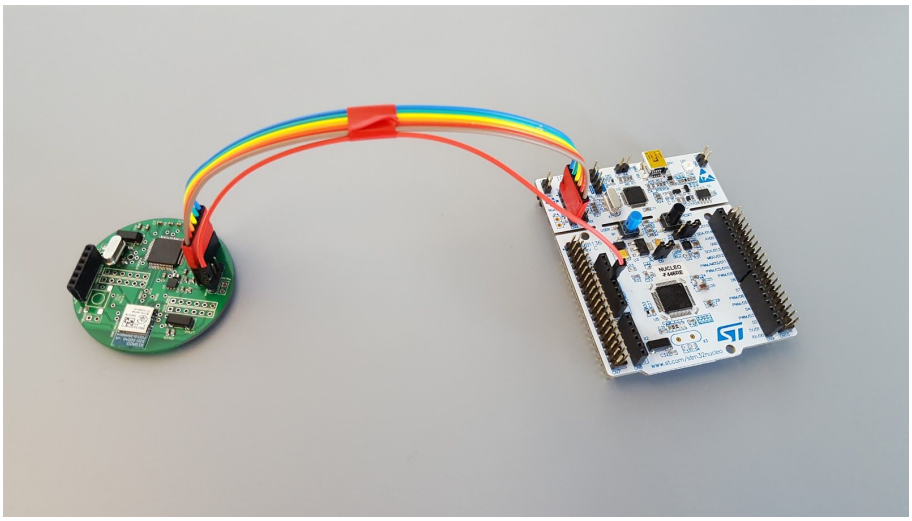


**Programming the STM32f4**

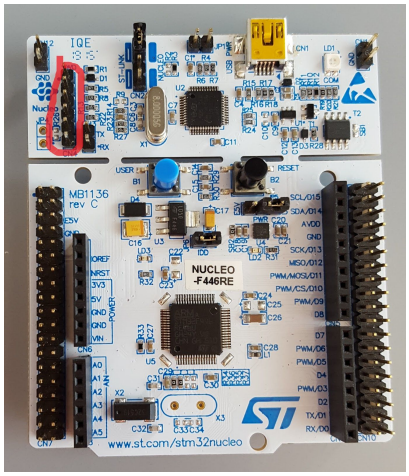
1. Download and install the ST-Link utility from the following link: http://www.st.com/content/st_com/en/products/embedded-software/development-tool-software/stsw-link004.html
2. Make sure the system and the development kit are powered off. The SEUS pcb comes with a jumper that connects a pin from the headers labeled “IDD 3V3” and a pin from the headers labeled “IDD VCC”. The jumper is shown and circled below. Make sure to take this jumper off before programming.



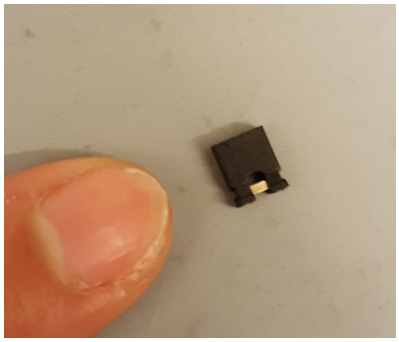
3. Connect the STM32f4 jumper from the development kit to the pcb as shown below. A detailed explanation of how to do this follows the image.



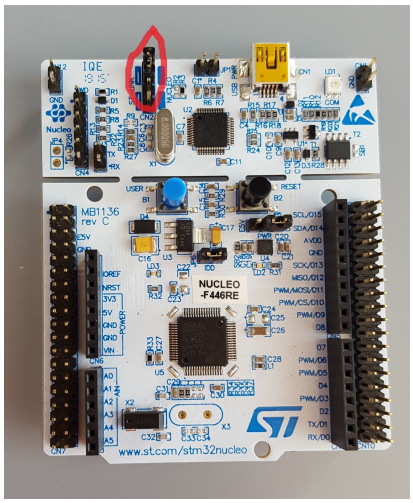
The side of the jumper that goes into the STM32f4 development kit have female headers and connect to header CN4 on the development kit. The CN4 header is circled below.



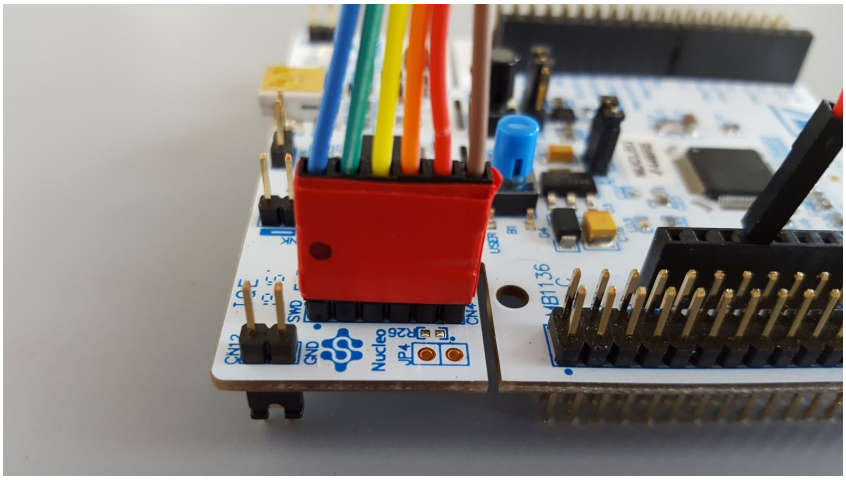

The STM32f4 development kit comes with jumpers that function to connect to adjacent headers on the board together. The jumper is shown below.



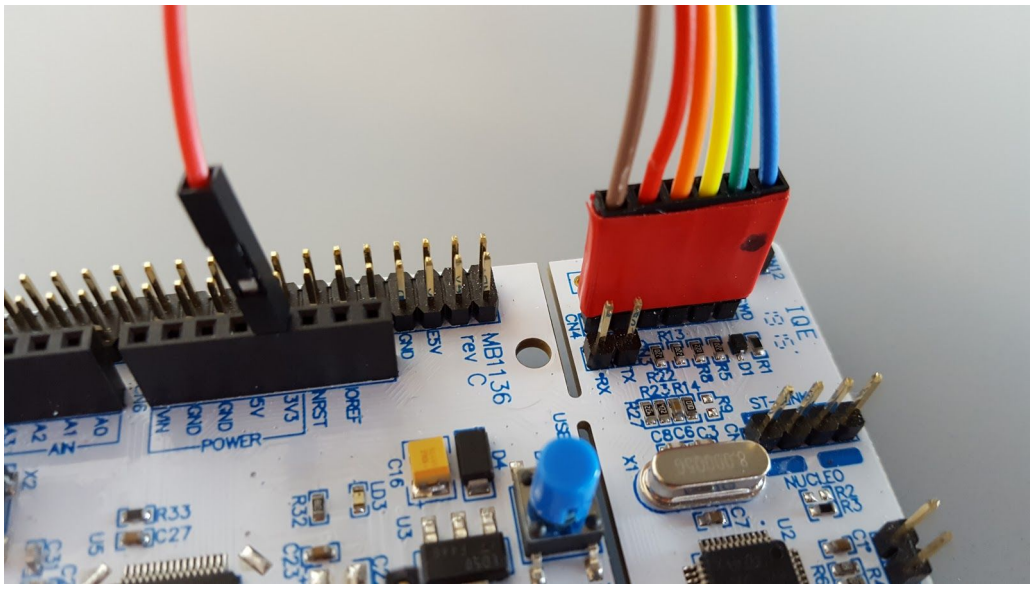
The development kit is shown below. The circled header, CN2, should be free from any jumpers that come with the board. By default, there are two jumpers on this header, so remove them



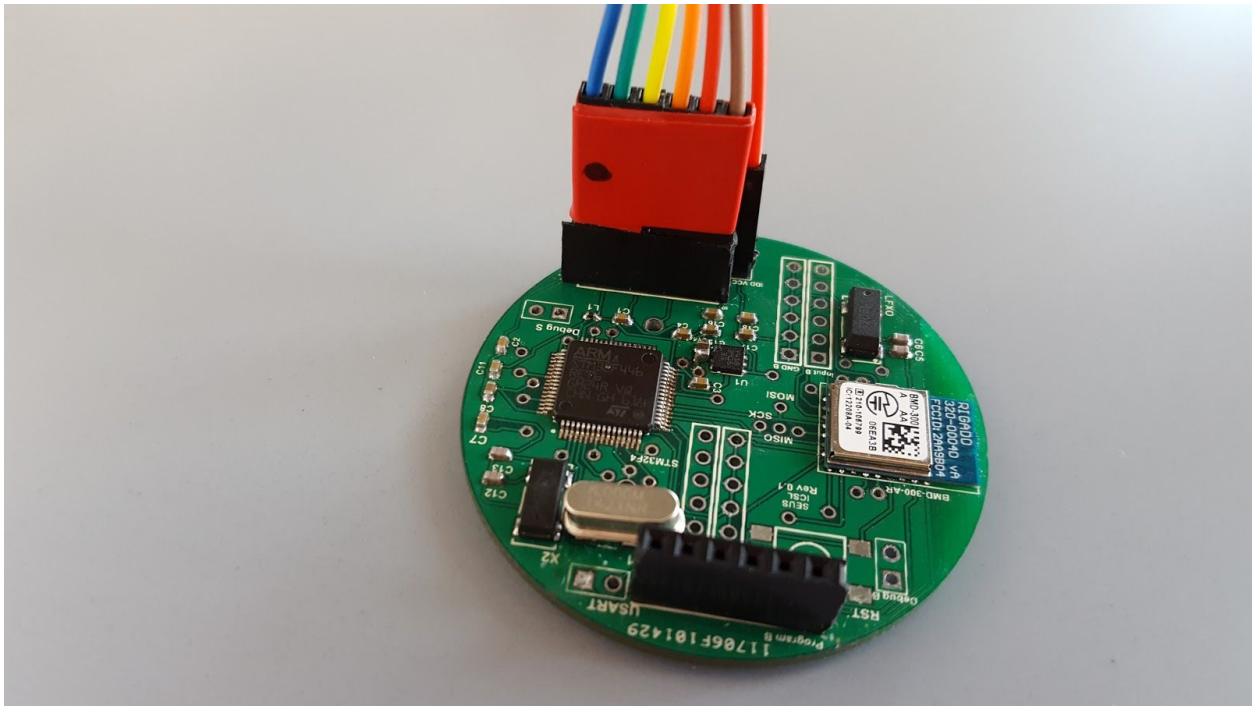

There is a dot on one edge of the jumper, use that to align with the header on the development kit as shown below.



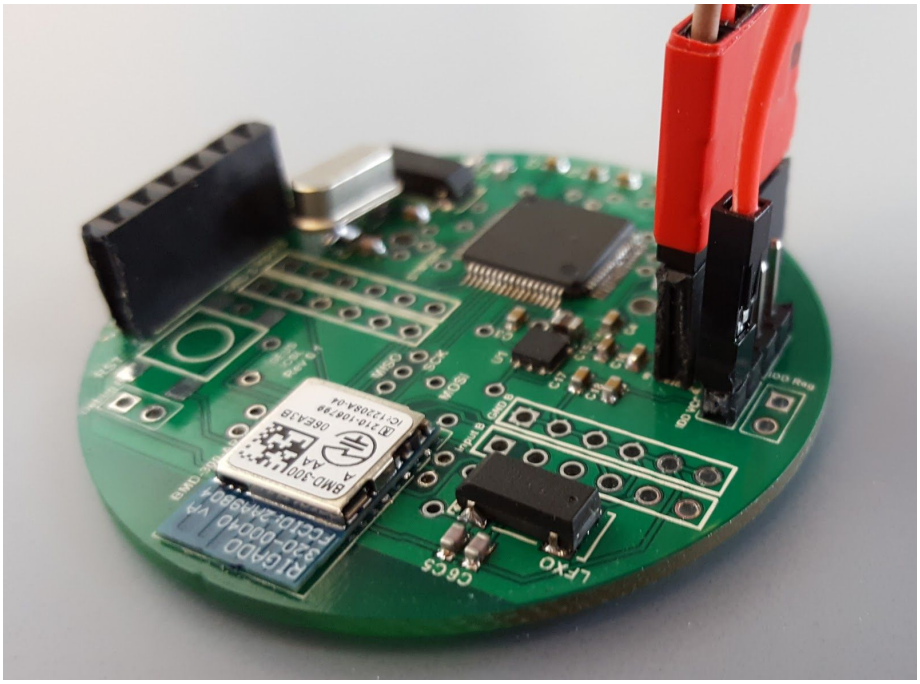
There is an additional red jumper that is not connected with the rest of the lines; this jumper goes into the “3V3” pin on the development kit as shown below.



Connect the other side of the jumpers to the female headers on the pcb labeled “Program S”. The end of the jumper with the dot goes into the header of “Program S” that is closest to the edge of the pcb, as shown below.



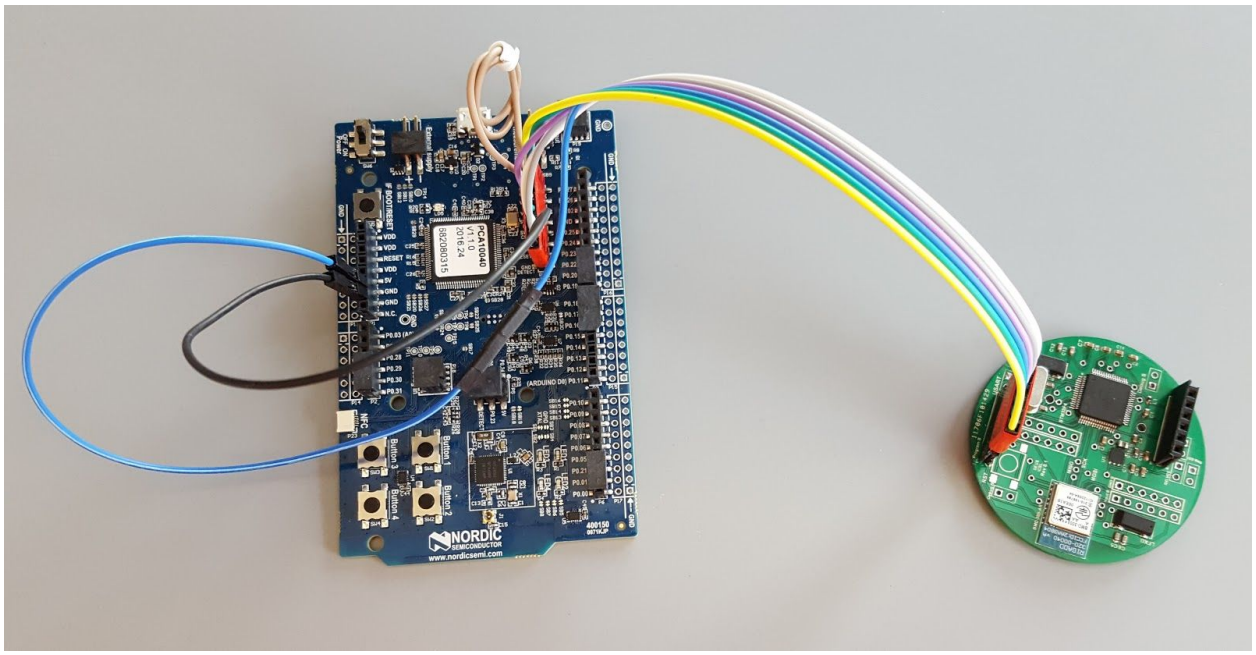
The extra red jumper that is not connected to the rest of the jumpers is used to power the board. Connect that to one of the headers labeled “IDD VCC”, as shown below.



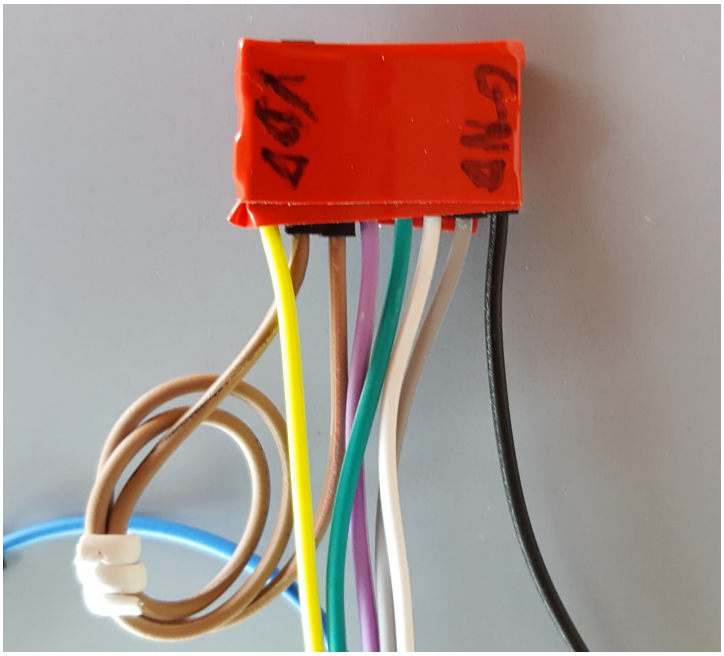
4. Now that the wiring is finished, connect the development kit to your computer using a mini-usb cable, and open up the ST-Link utility and go to “ST-LINK -> Firmware update”. In the pop-up window, click “Device Connect” and hit the “Yes” button to upgrade the firmware.
5. Click “Target->Connect” to connect to the target, and click “Target->Erase Chip” to clear the old firmware off of the STM32.
6. Go to “File->Open” and select the firmware file (.hex or .bin) that is to be flashed onto the chip.
7. Finally, go to “Target->Program & Verify”. Click “Start” in the pop-up window to flash the firmware. Once this is finished, the new program should be installed onto the STM32f4, make sure to replace the jumper that was removed from the pcb in step three before powering on the system.

**Programming the BMD-300 BLE Module**

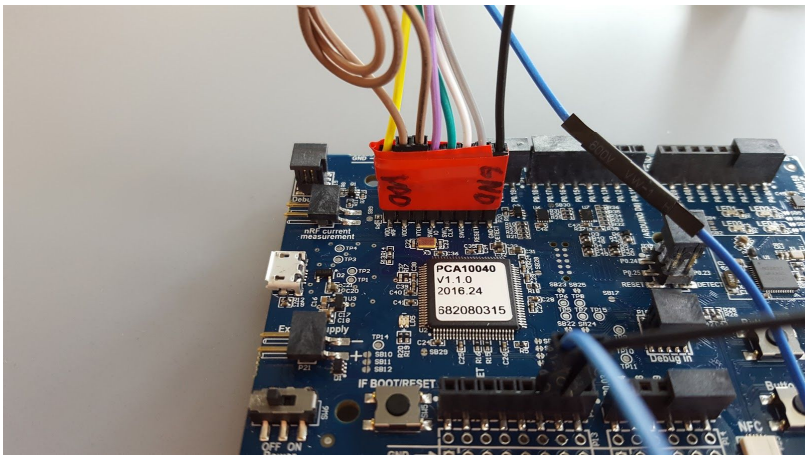
1. Download and install nRFgo Studio, nRF5x command line tools, and the j-link driver using the following links
    a. http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRFgo-Studio
    b. https://www.nordicsemi.com/eng/nordic/Products/nRF51822/nRF5x-Command-Line-Tools-Win32/33444
    c. https://www.segger.com/downloads/jlink
2. Make sure the pcb and the development kit are powered off and make sure the jumper connecting “IDD 3V3” and “IDD VCC” is removed just like when programming the STM32f4. Connect the Nordic nRF52 development kit to the pcb as shown below. A detailed explanation of how to do this will follow the image.



The end of the programming jumper that is connected to the development kit is labeled on one edge with “VDD” and the other edge with “GND”, as shown below.

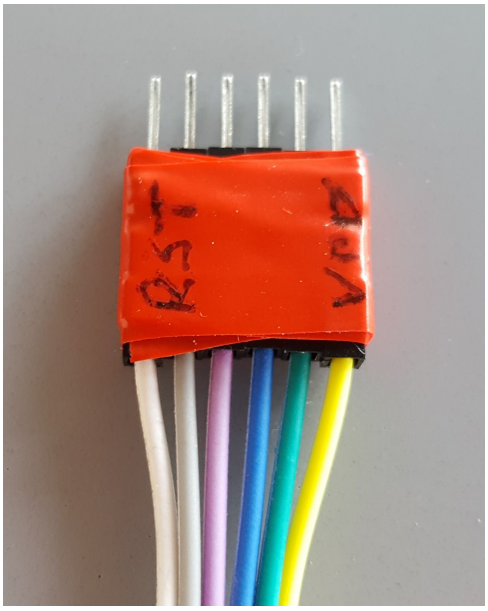



The edge labeled “VDD” is connected to the pin on the development kit labeled “VDD” as shown below.

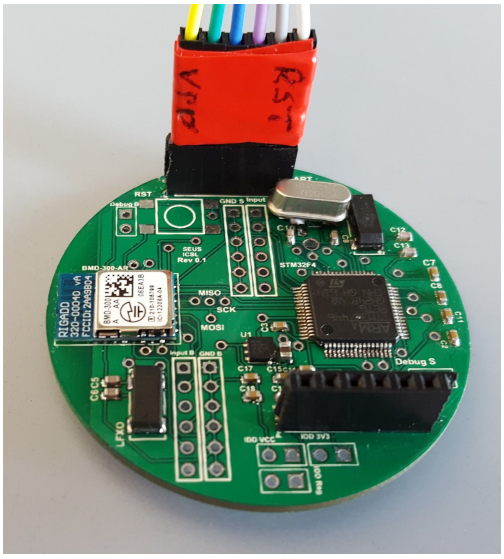



There are two other jumpers, one black and one blue, that are not connected with the rest of the jumpers wrapped in insulation tape. These pins are connected to any “GND” pin that is on the development kit. Two possible placements are shown in the image above.

The other end of the jumpers that connect to the pcb are shown below. One edge is labeled “VDD” and the other edge is labeled “RST”. This jumper is connected to the headers on the pcb labeled “Program B”.



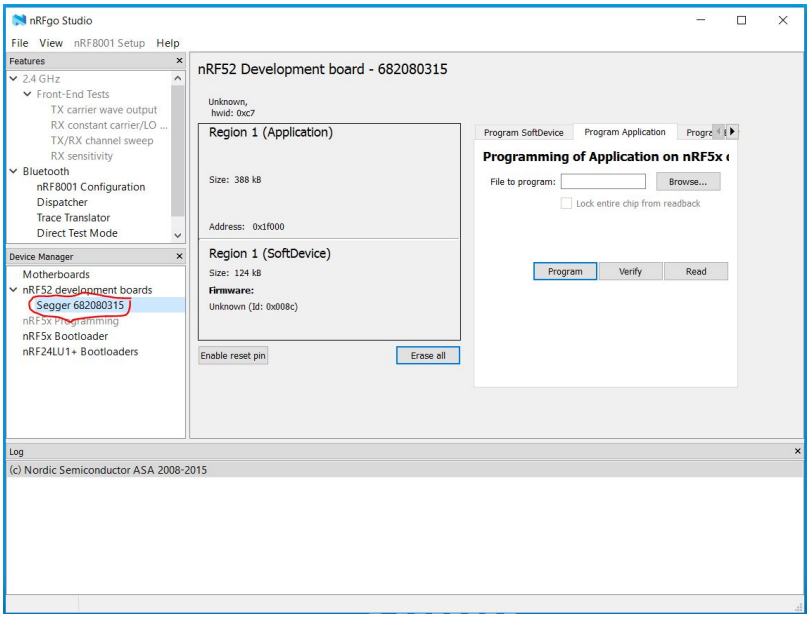

The side edge side of the jumpers that is labeled “VDD” is connected into the pin that is closest to the edge of the board, as shown below.



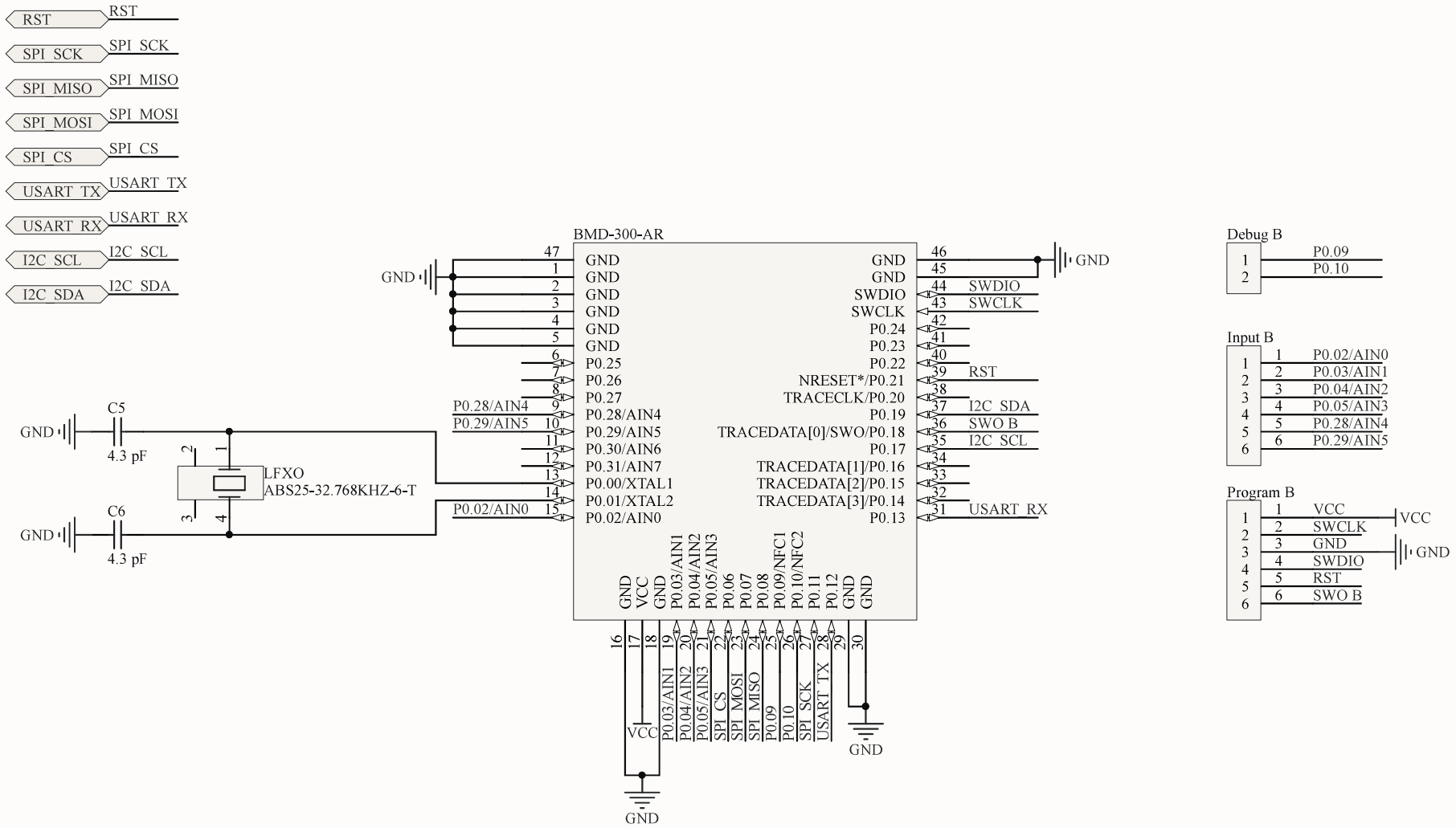

*Note: Programming the BLE module does not require anything to be connected to “IDD VCC” unlike programming the STM32f4

3. Now that the wiring is finished, connect development kit to the computer using a micro-usb cable.
4. Open a command prompt or terminal prompt and execute the command:
    a. “nrfjprog -f NRF52 --recover”
    b. This erases all of the Rigado firmware on the module. This step only has to be completed one time; after erasing this one time, there is no need to complete this step again.

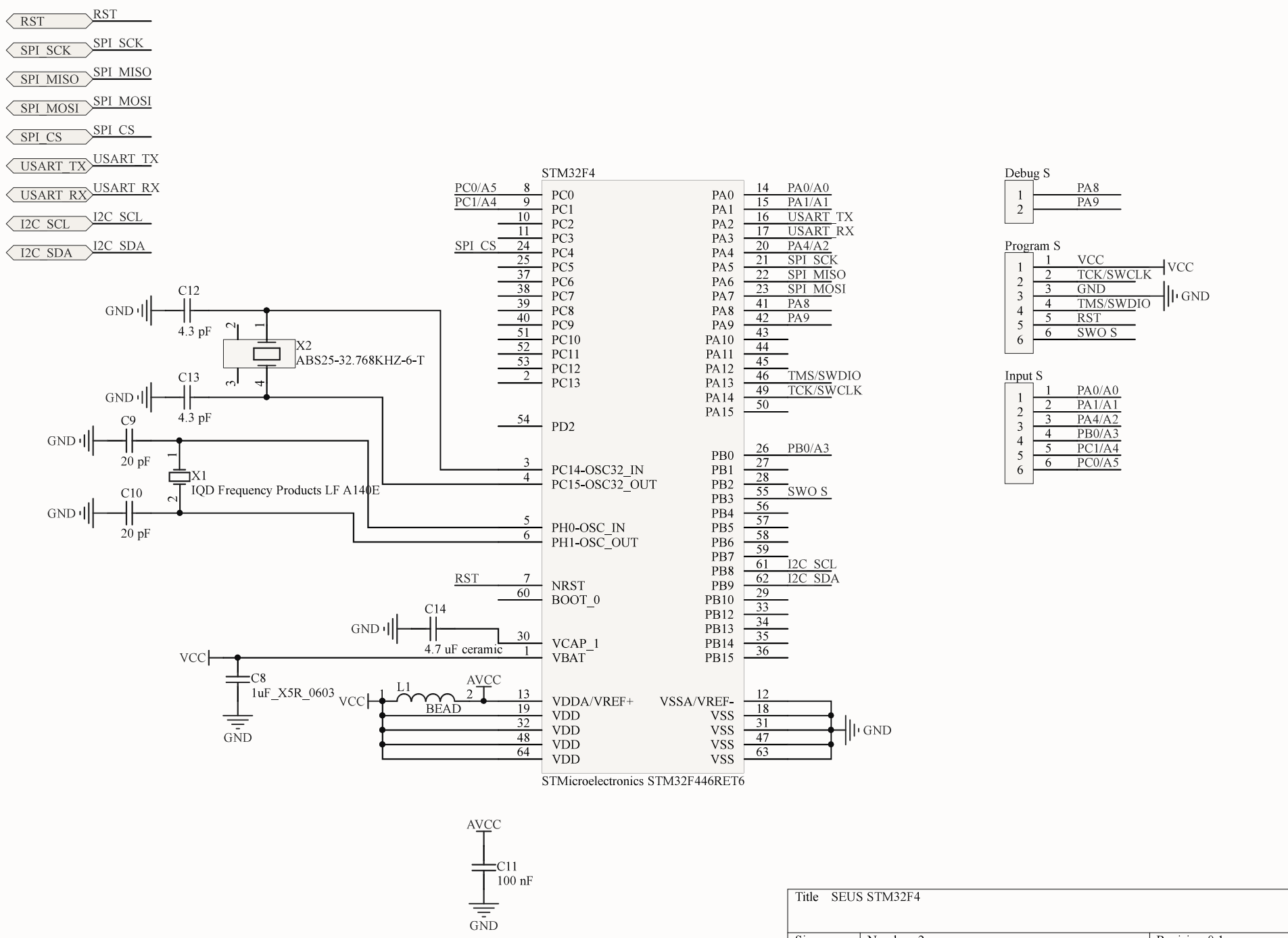
5. Open nRFgo Studio. The module on the pcb should be seen under “nRF52 development boards”, as shown below. Select the board that is to be programmed



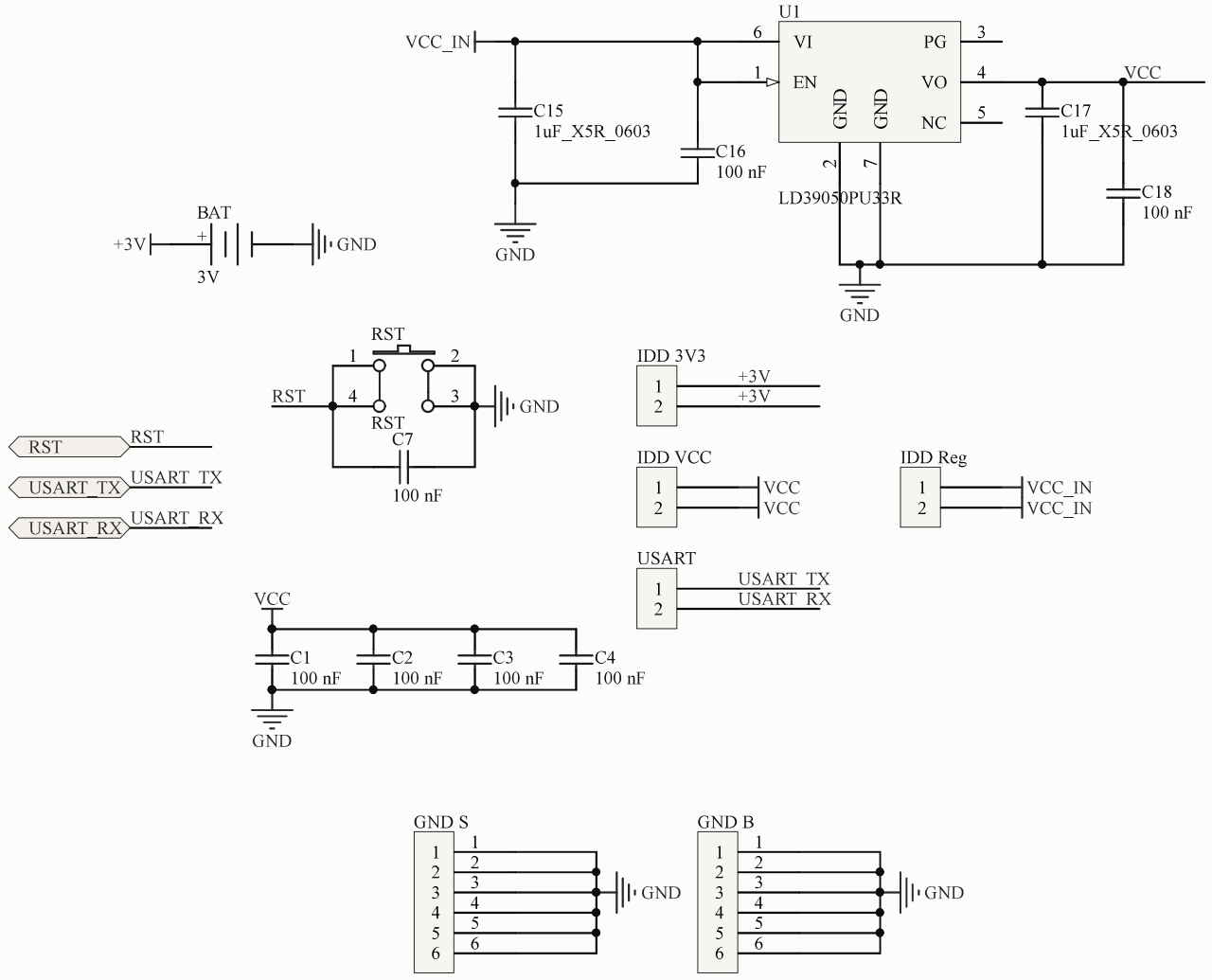

6. In the rightmost section of the interface, make sure the selected tab is “Program Application”. Choose the firmware to program onto the BLE module, and click “Program”.
7. Now that the BLE module is programmed, replace the jumper connecting “IDD 3V3” and “IDD VCC” together on the pcb and power the board back on.



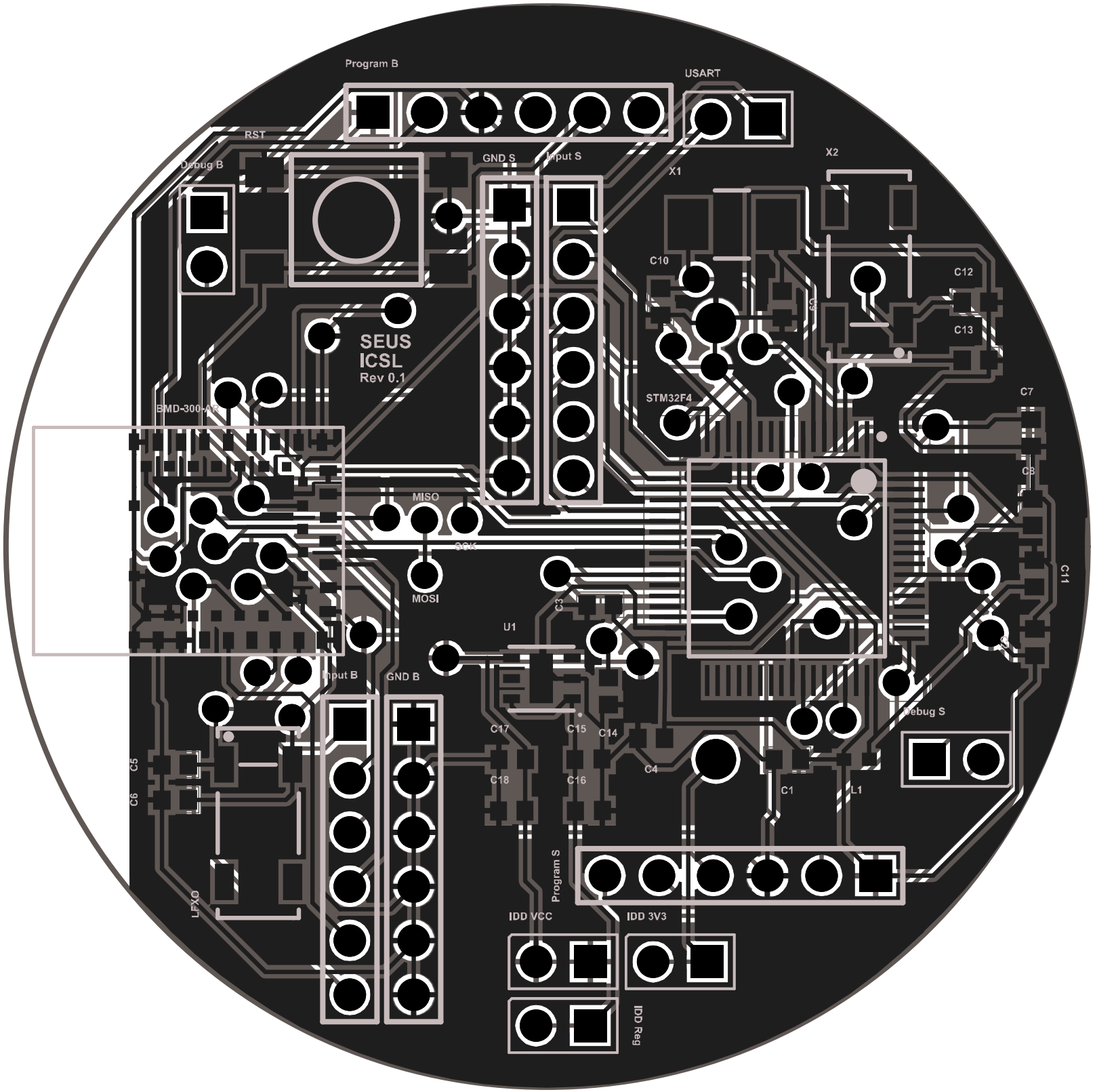

| | | |
|---|---|---|
| Title SEUS BMD | | |
| Size A | Number 1 | Revision 0.1 |
| Date: 9/17/2016 | | Sheet 1 of 3 |
| File: ICSL_PS_Combined_BMD.SchDoc | | Drawn By: Stephen Xia |




| | | |
|---|---|---|
| Title SEUS STM32F4 | | |
| Size A | Number 2 | Revision 0.1 |
| Date: 9/17/2016 | | Sheet 2 of 3 |
| File: ICSL_PS_Combined_STM.SchDoc | | Drawn By: Stephen Xia |




| | | |
|---|---|---|
| Title SEUS POWER + BREAKOUT | | |
| Size A | Number 3 | Revision 0.1 |
| Date: 9/17/2016 | | Sheet 3 of 3 |
| File: ICSL_PS_Combined_Shared.SchDoc | | Drawn By: Stephen Xia |


Program B
USART
RST
Debug B
GND S
Input S
X1
X2
C10
C12
C13
SEUS
ICSL
Rev 0.1
STM32F4
C7
BMD-300-A-R
C8
MISO
SPI
MOSI
C3
U1
C11
Input B
GND B
Debug S
C17
C15
C14
C5
C6
C18
C16
C4
C1
L1
Program S
IDD VCC
IDD 3V3
LFXO
IDD Reg

C11
C7
C8
C2
C12
C13
Debug S
X2
C9
L1
C1
USART
X1
C10
STM32F4
C4
C14
C15
C16
C3
U1
C17
C18
Program S
IDD 3V3
IDD Reg
IDD VCC
Input S
GND S
SCK
MISO
MOSI
GND B
Program B
SEUS
ICSL
Rev 0.1
Input B
RST
Debug B
BMD-300-AR
BMD-300
A YY
FCC ID:
2AA9B04 XXXXXX
IC: 12208A-04
LFXO
C5
C6