BLE MIDI Keyboard Project Report

Yiran Zhang, Renzhi Hao, Zhihao Deng
yiranzhang2023@u.northwestern.edu,renzhihao2022@u.northwestern.edu,zhihaodeng2022@u.northwestern.edu

ABSTRACT

This is a BLE MIDI Keyboard project report of Yiran, Renzhi, and Zhihao for the wireless protocol class at northwestern. In this paper, you will see the working result of how we make several buttons works like a wireless bluetooth MIDI keyboard and play music when detecting button pressed behavior.

1 INTRODUCTION

Nowadays, a lot more musical instrument companies and other music brands focus on making not just traditional instruments but electronic ones. Musicians and music producers thus are able to play different instruments easily with one smaller MIDI keyboard. The MIDI keyboard could be connected via cables to computers or other devices such as phones or tablets to transmit MIDI signals towards the softwares as they press the keys. But as the MIDI keyboard must use its own specific MIDI cable and port rather than common USB or type C cables, it's inconvenient to connect the MIDI keyboard to the devices. Compared to matching devices with relative cable or adapters, it's reasonable and practical to have a wireless MIDI keyboard. With the huge development on wireless technologies and more and more usage of bluetooth on the wearables in our daily life, more and more electronic instruments would like to implement the bluetooth settings in order to offer the best user experience. Inspired by this, our team is curious about how the BLE technique works for musical instruments and designed a BLE MIDI keyboard with buttons representing the keys.

2 BACKGROUND

Before stepping into the project and getting our hands dirty on it, we read some articles and related work to better understand the steps we need to work on and what we need in order to achieve our goal.

CS397/497, Spring 2022, Northwestern University
2022. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Figure 1: nRF52840 Board

2.1 nRF52840 Board [1]

In this project, we decided to use nRF52840 because it is the most accessible board for us and it supports the BLE protocol. It is built around the 32-bit ARM® Cortex™-M4 CPU with a floating point unit running at 64 MHz. It has NFC-A Tag for use in simplified pairing and payment solutions. The ARM TrustZone® CryptoCell cryptographic unit is included on-chip and brings an extensive range of cryptographic options that execute highly efficiently independent of the CPU. It has numerous digital peripherals and interfaces such as high speed SPI and QSPI for interfacing to external flash and displays, PDM and I2S for digital microphones and audio, and a full speed USB device for data transfer and power supply for battery recharging. [2]

2.2 MIDI protocol

MIDI represents for Musical Instrument Digital Interface, and it's a real-time communication protocol for real-time data transmission between hardware. Since all electronic instruments and computers communicated with MIDI signals, we thus searched on the MIDI Protocol to see what MIDI information is like and understand how we can package our own button's bit signal to the MIDI message. [3] From the protocol, we know that MIDI is a stream of message, and a message is commonly recognized as a sequence of bytes which uses 4 low bits as 0-127 in decimal for data bytes, and 4 most significant bits as 128-255 in decimal for command bytes. In command bytes, the 4 most significant

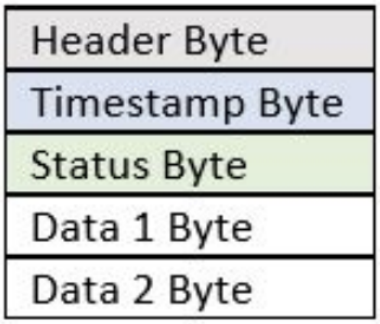

Figure 2: MIDI message sending format

[Header Byte,	Timestamp Byte,	Status Byte,	Data Byte 1,	Data Byte 2]
[0x80,	0x80,	0x90,	0x3C,	0x7F]

Figure 3: MIDI message example for middle C

bits are command part and the 4 least significant bits are channel part. MIDI protocol uses UART (Universal Asynchronous Receiver/Transmitter) for transmission simplex.

2.3 Bluetooth LE MIDI Specification

The BLE-MIDI specification has some modification to the MIDI protocol to fit the BLE format.[4] The stream of message are turn into packet based information with a minimum 7.5ms connection interval. So the timestamps are added to help interleave and deinterleave message in this time sensitive situation. The header is also need for the BLE package. The header A normal BLE MIDI message is consist of header, timestamp, status and data.(Figure 2) The header For an example of MIDI message sequence of turning a note on, take the middle C as example, the header and timestamp could be set as 0x80 for simplicity, as we will not care about the time bits in the Header and Timestamp and assume the time is 0. The command bytes is used as the turning note byte as 0b1001xxxx and the channel No.1 byte as 0bxxxx0000. The data byte 1 is the key of the note which is 0x3C for middle C, and the data byte 2 is the velocity the note is pressed and is the maximum as 0x7F in this example (Figure 3).

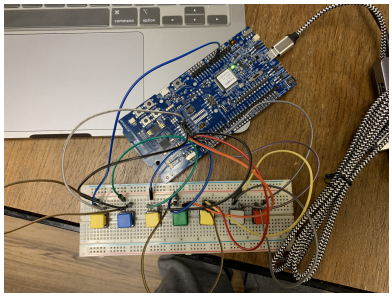

Figure 4: Circuit for buttons to nRF board

2.4 GATT for data over BLE

The GATT stands for Generic ATTribute Profile, and it establishes in detail how to exchange all profile and user data over Bluetooth LE connection between central and peripheral devices. The central device has several service and a service could have multiple characteristics, which contains descriptor for identity identification between devices, properties for permission and value as the data transmitted itself.

3 DESIGN AND IMPLEMENTATION

After researching enough about the background of how MIDI works, and how MIDI and BLE can be encoded and decoded, we detailized the steps and workflow we need to follow. It includes transferring button event to bit signal, encoding bit signal to BLE-MIDI package, transmit BLE-MIDI package, receive package from peripheral and software decoding package.

3.1 Button Event to Bit Signal

In this project, we wired up the seven buttons to the nRF52840DK board (Figure 4) to represent a full octave. The process of designing this part is shown as follow (Figure 5).

First we need to customize our devicetree configuration for nRF52840DK using an overlay file so that pin0's 11-15, 24, and 25 are wired corresponded with each button. Secondly, we initialized each button and necessary configuration when starting the board and test its connection. We then defined a callback function for button pressed and use it for further implement.

3.2 Bit Signal to BLE-MIDI Package

In this part, we demonstrate the tranfer between bit signal and MIDI data.

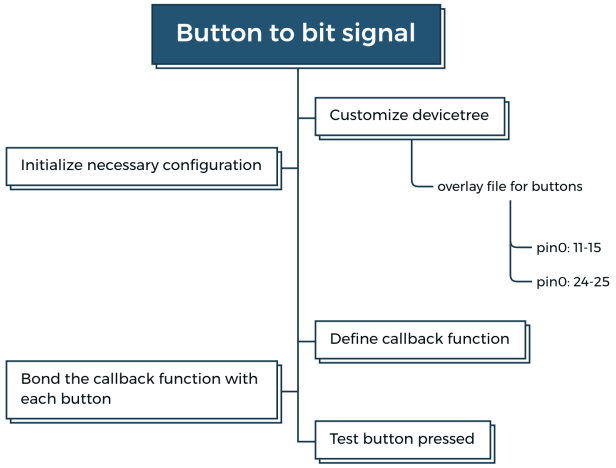

Figure 5: Process of button to bit signal

```
/*Example MIDI Data Package */
#define MIDI_Package_C_Note 0x80 0x80 0x90 0x3C 0x7F
//Header-0x80 Timestamp-0x80 Status-0x90 MIDI_NOTE_C3-0x3C MIDI_VELOCITY_MAX-0x7F
#define MIDI_Package_C_Chord 0x80 0x80 0x90 0x3C 0x7F 0x80 0x90 0x40 0x7F 0x44 0x7F //C, E and G Notes
//Header-0x80 Timestamp-0x80 Status-0x90 MIDI_NOTE_C3-0x3C MIDI_VELOCITY_MAX-0x7F
//Timestamp-0x80 Status-0x90 MIDI_NOTE_E3-0x40 MIDI_VELOCITY_MAX-0x7F MIDI_NOTE_G3-0x44 MIDI_VELOCITY_MAX-0x7F
```

Figure 6: Define MIDI Format as Macro

Before we illustrate the process, we need to define a struct of MIDI data. The MIDI message sending format (Figure 3) contains four parts: Header, Timestamp, Status, and Data. Within the data part we can define the midi note and midi velocity. The example MIDI message sending format is displayed in Figure 6.

Firstly, we define a Hex Byte Array named messages as the global variable and initialize it with { 0x80, 0x80, 0x90, 0X00, 0x7F }, which represent null in MIDI data. Then on the callback function, we detect the pin of the callback function with the button and modify the messages with corresponding value as different notes. After that, we were able to pass this to the function of the BLE-MIDI transmission module. When the next MIDI signal is detected, this process is repeated. The whole process of this part is shown on Figure 7.

3.3 BLE-MIDI Package Transmit

We follow the GATT format for topology of BLE-MIDI connection and data transmission. The MIDI service has one characteristic as the MIDI data, which are both defined by the unique 128bit UUIDs in MIDI protocol. A client, which is the central device such as the phones or computers having music software can subscribe to

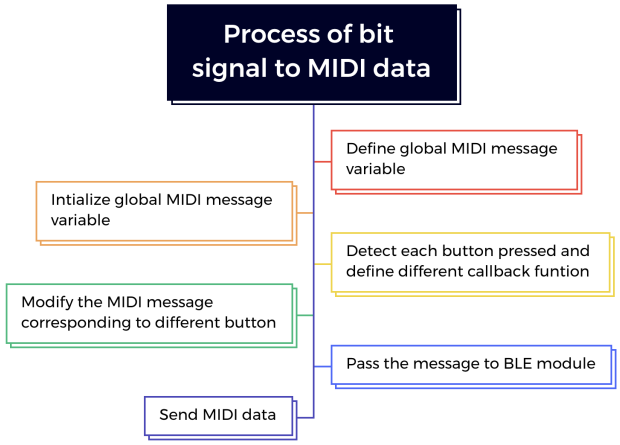

Figure 7: MIDI message format for sending

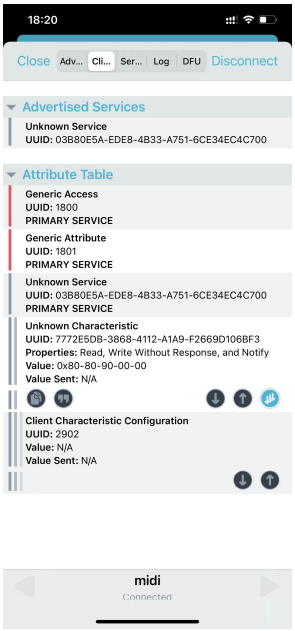

Figure 8: Notify Property Enabled

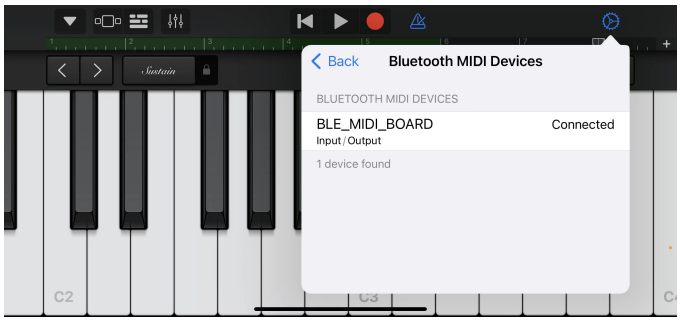

Figure 9: Connection on GarageBand

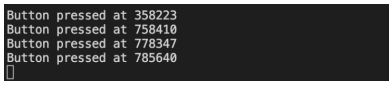

Figure 10: Button Detection Message

notifications from the MIDI data characteristic to receive BLE-MIDI package from the server, which is our BLE MIDI keyboard as the peripheral. The Bluetooth connection is implemented by Scan and Advertising procedure. When the power of nRF board is on, it start advertising the complete local name for Bluetooth scanning. On the device when opening the music software such as GarageBand, it start scanning the device and send scan request to the nRF board. The nRF board then send the MIDI UUID as the scan response to match successfully. After the matching, the pair connection is established.(Figure 8)

3.4 Peripheral Receive Package

By enabling the notification property on the central side as phone or computer, the CCCD (Client Characteristic Configuration Descriptor) as 2902 is defined for the flag of notification, and the BLE MIDI board can send MIDI message as notification through the MIDI data characteristic value intentionally without reading operation from central side every time.(Figure 9)

3.5 Software Decoding Package

The software such as GarageBand and FLStudio could detect MIDI Service and connect after verification. After connection, it could automatically subscribe and decode the qualified MIDI package and generate note relatively.

4 RESULT EVALUATION

4.1 Button Detection Debugging

In the button detection procedure, we test it by printing out a button pressed message when we pressed any buttons.(Figure) Therefore, we are able to see when the button being pressed and thus it successfully detect the button bit signal of the current state of buttons. (Figure 8)

We got MIDI data packing ready for byte array so that we can later package that into BLE version to transfer this MIDI message to the phones or tablets.

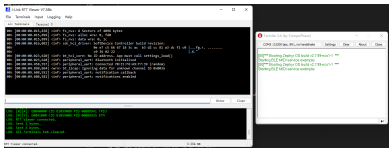

Figure 11: JTT and Termite Terminal

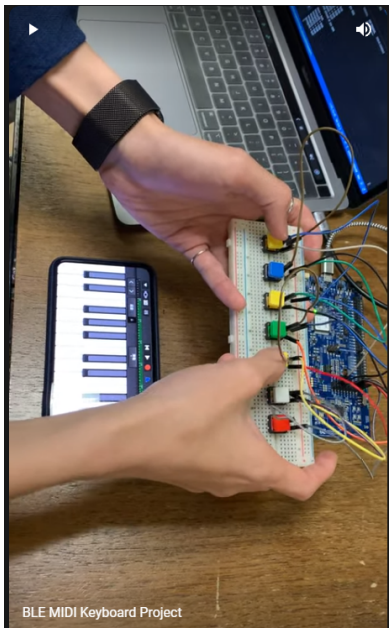

Figure 12: Playing Real-time Demo

4.2 Data Sent Debugging

We then got Board-BLE-Software connected. Thus, the nRF Connect Mobile application is able to connect Garageband and program to send and receive data from it.(Figure) On the JTT viewer, we can see it is connected at line [00:00:03.075,714] and we can send data through UART port to test the transmission. Then, we can send MIDI data hex array a note and play the corresponding note when the GarageBand receive this array.

4.3 Video and Code Links

The video for playing Twinkle Twinkle Little Star using our BLE MIDI Keyboard: https://youtube.com/shorts/U-aYi7iQOs4?feature=share (Figure 12)

The GitHub repository: https://github.com/haorenzhi/BLE_MIDI_Keyboard

REFERENCES

[1] N. Semiconductor, "nrf52840 product specification," *Nordic Semiconductor. Version*, vol. 1, 2018.

[2] "nRF52840 system on chip." https://www.nordicsemi.com/Products/nRF52840. Accessed: 2022-06-07.

[3] "Midi communication protocol." https://www.midi.org/specifications/midi-2-0-specifications/ump-protocol-specifications. Accessed: 2022-06-07.

[4] "Bluetooth le midi specification." https://www.midi.org/specifications-old/item/bluetooth-le-midi. Accessed: 2022-06-07.