TriMark e-ASK UWB Digital Key System

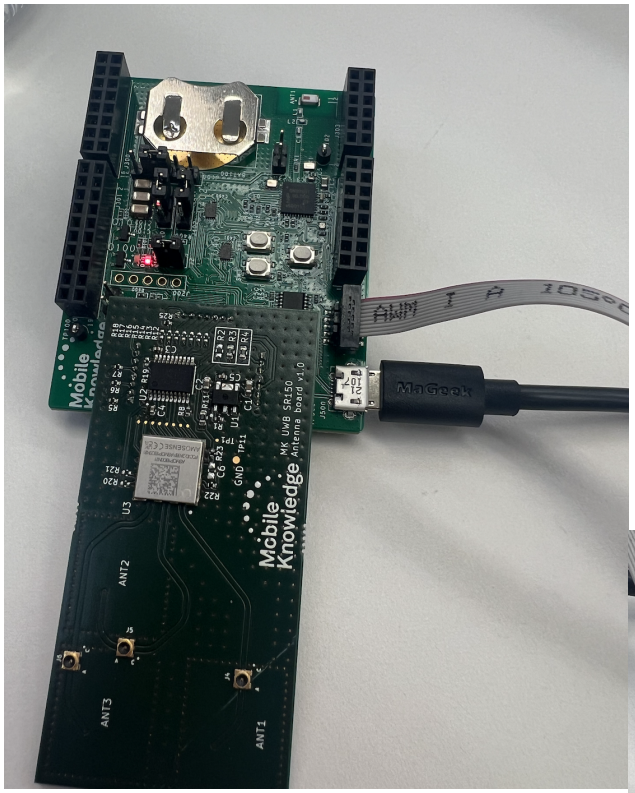
Iowa State Senior Design Team 9

Table of Contents

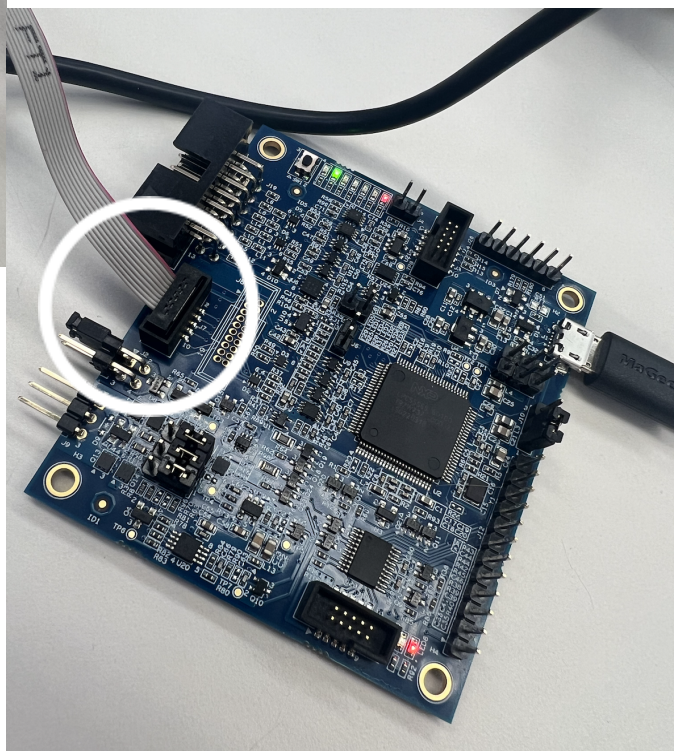
How To: Set-Up NXP MCU-Link Pro Debug Probe & Flash Memory

Instructions on connecting the hardware and flashing code are in more detail at this link: S2 - Module 2: MK UWB Kit hands-on (timestamp: 9:15). The debugger that was given to us is a different version, therefore some of the ports are in different spots. The pictures below show how to set up for our hardware version.

Anchor Setup:



NXP Debugger Setup:
(jumper 2 and debugging cable circled)



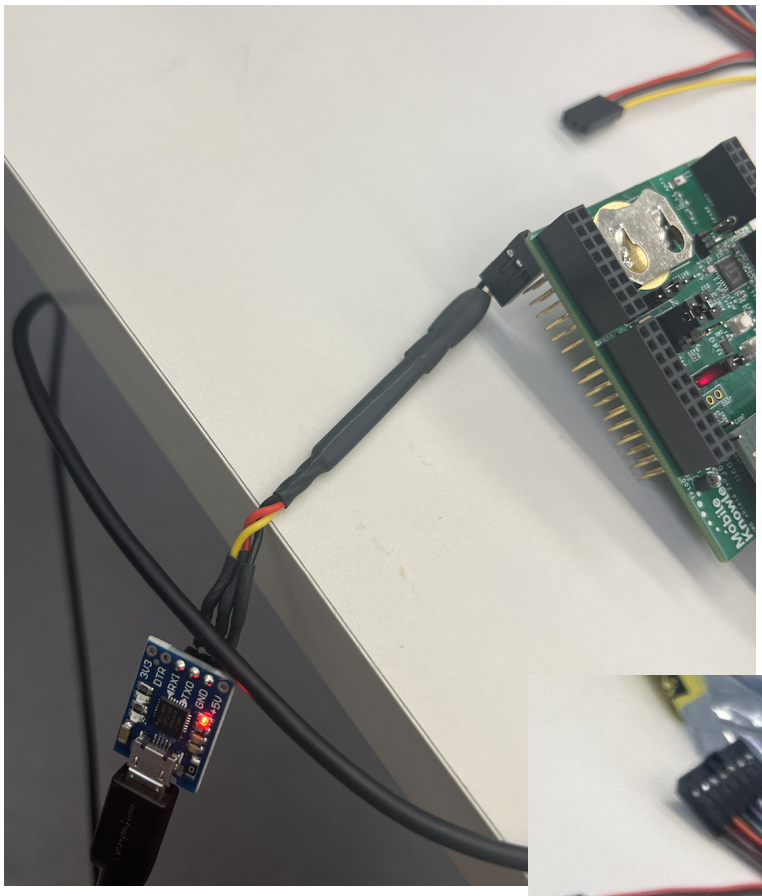
Flashing Instructions:
- MCUXpresso IDE is needed.
- Follow all instructions from video, but switch build configuration to UWB Shield 2 SR150
- For Mobile Edition, you will need to download the necessary SDK into MCUXpresso. This SDK can be found at this link: https://mcuxpresso.nxp.com/en/select.
    - select QN9090DK6 board
    - include FreeRTOS and Wireless BLE components

How To: Read Logs From NXP UART Interface

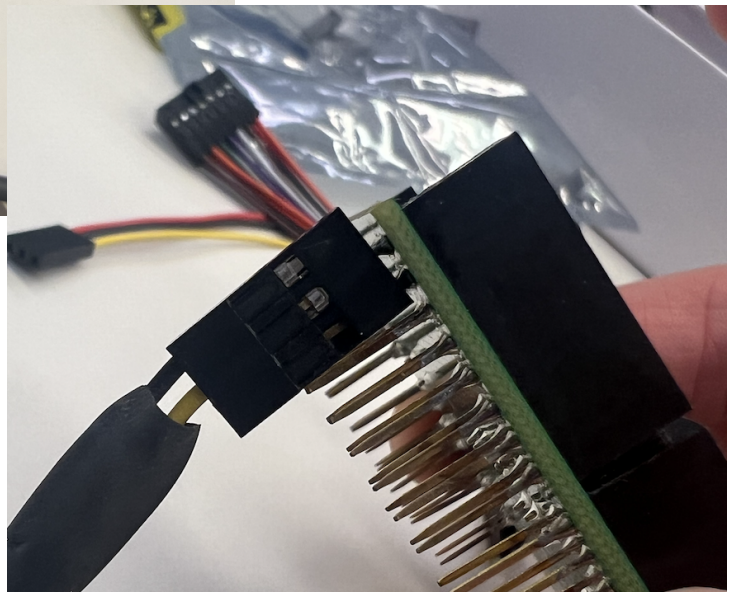
Using a serial port reader, configure the following settings:

- Set COM to the port of the FTDI cable
- Baud rate: 115200
- Data size: 8
- Parity: none

Connect the RX port FTDI cable to the TX port of the MK UWB Shield 2 shown in the following pictures:



(yellow cord is RX on FTDI cable)



UART Standards

UART Communication

MK has existing UART code that handles sending messages, created to interface with the PCShell. Here are a few notes on this:

- Baud Rate: 115200
- Data Size: 8
- Parity: none

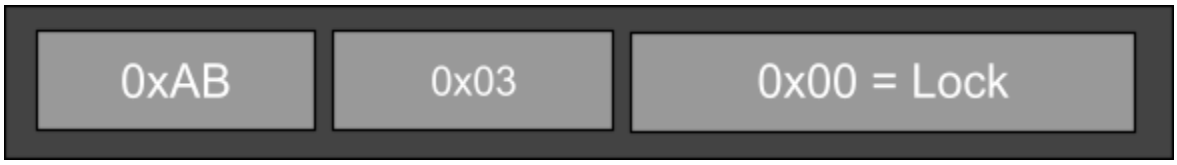
TLV: Type, Length, Value

The MK code samples use TLV's to encapsulate the UART messages sent to the PCShell. For the application that we are using, we will use the suggested values that Mobile Knowledge went over during the training videos.

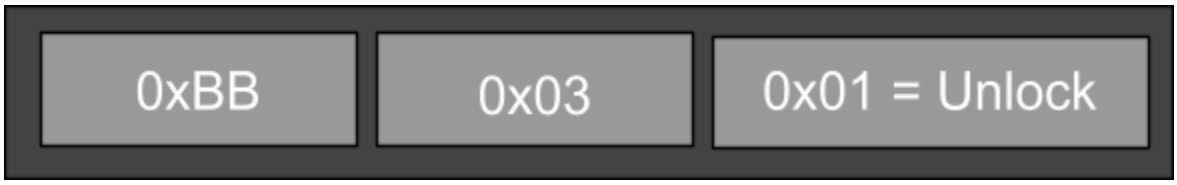
In order to send messages with distance information, we need to account for the maximum BLE range as well as the MAC address of every phone. The maximum BLE range is 100 meters, and the MK kit measures to centimeter accuracy, meaning we will need 15 bits (rounded up to 2 bytes) to store the maximum length. A UUID address contains 16 bytes of data. In addition to these pieces of data, we will send a type specifier, which is 1 byte, along with length values, which is also 1 byte. Overall, the message size will be a maximum of 20 bytes.

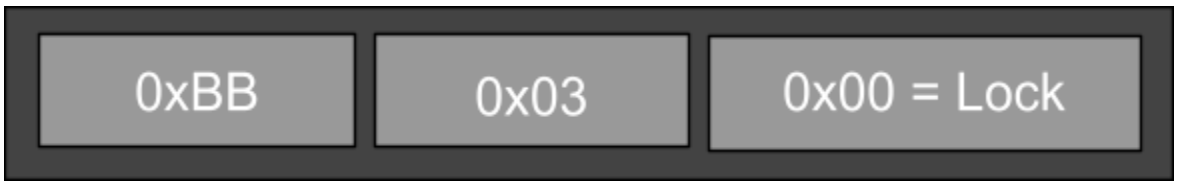
Distance Message

Unlock/Lock Status Messages

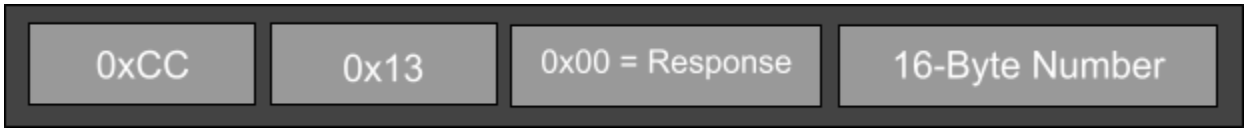


Unlock/Lock Request Messages





Random Number Request Message



UUID Authorization Message

UUID Authorization Response

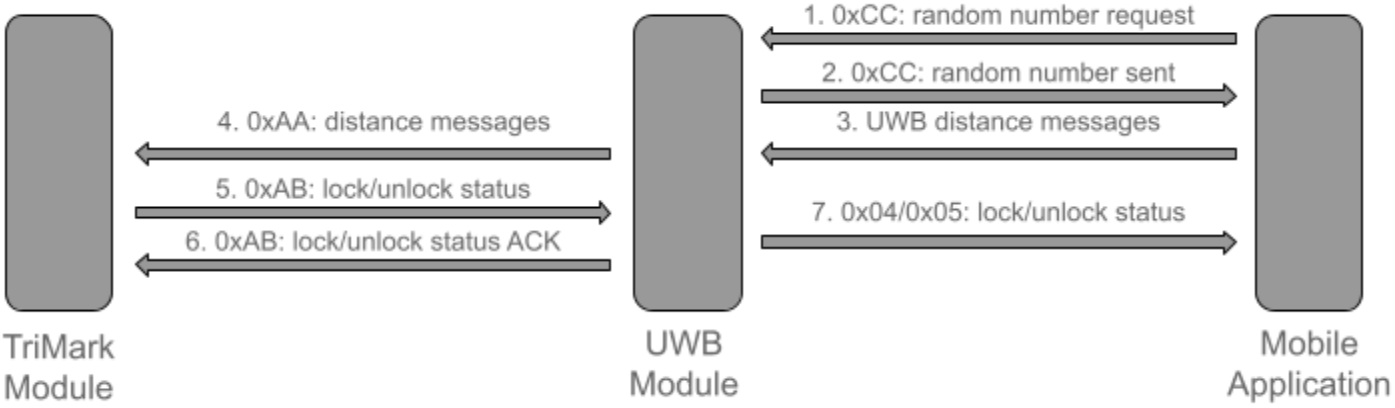
UUID Pairing Message

UUID Pairing Response

Vehicle Nickname Response (UUID Authorization Positive Response)

Message Flow - Use Cases

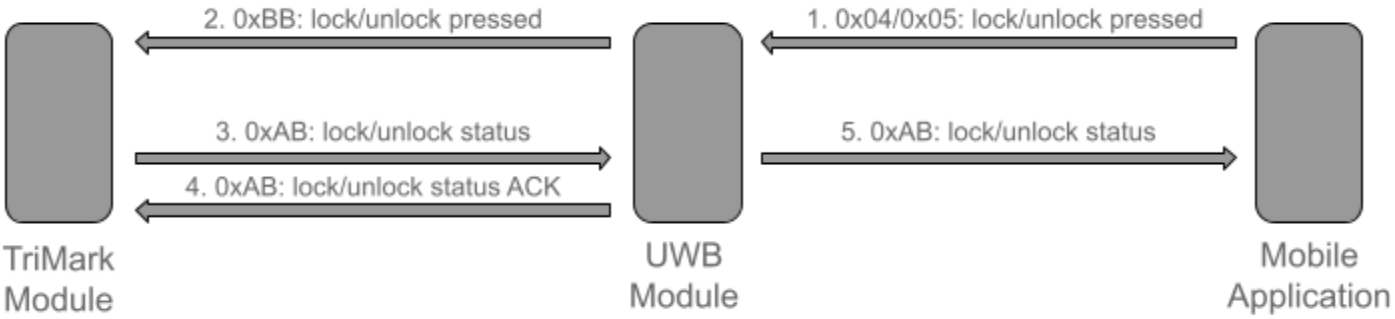
Phone Enters/Exits Lock/Unlock Range



1. **Phone:** Establish BLE & UWB connection & request random number from UWB Module
    a. Type: 0xCC
2. **UWB Module:** Send random number
    a. Type: 0xCC
3. **Phone:** Establish BLE & UWB connection and begin sending distance messages.
    a. Type: no type - done implicitly
4. **UWB Module:** Continuously sending distance messages

   a. Type: 0xAA
5. **TriMark Module:** When distance is within unlock range/outside of lock range, locks/unlocks and sends status.
   a. Type: 0xAB
6. **UWB Module:** Lock/unlock status received, sends ACK of receival of this message.
   a. Type: 0xAB
7. **UWB Module:** Lock/Unlock status sent to mobile phone to display.
   a. Type: 0x04/0x05

Manual Unlock/Lock



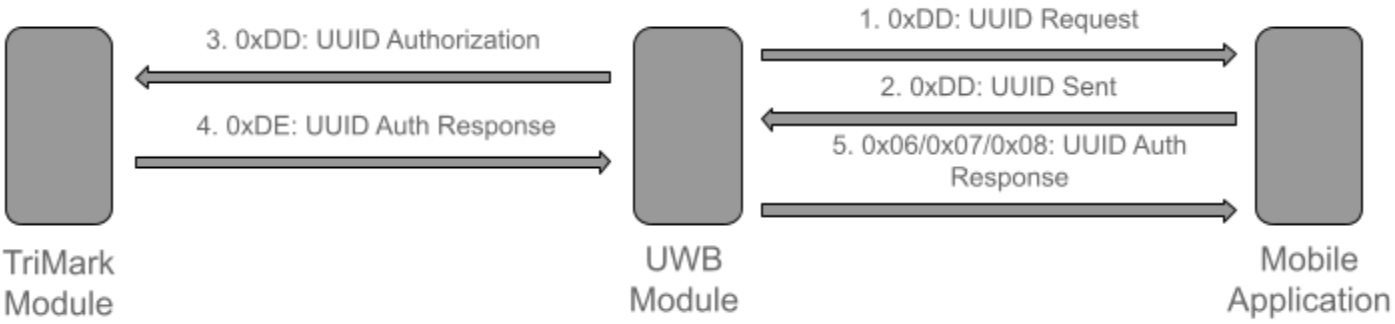
1. **Phone:** Lock or Unlock button is pressed
   a. Type: 0x04/0x05
2. **UWB Module**: Forward lock or unlock pressed message to TriMark module
   a. Type: 0xBB
3. **TriMark Module**: Lock/unlock button press received, action performed, sends status message to UWB module
   a. Type 0xAB
4. **UWB Module:** Lock/unlock status received, sends ACK of receival of this message.
   a. Type: 0xAB
5. **UWB Module:** Lock/Unlock status sent to mobile phone to display.
   a. Type: 0xAB

Push-To-Start

1. **TriMark Module:** Should be the only module involved in this scenario.
   a. Module checks whether a registered phone is within Push-To-Start distance, executing the start or ignoring the request as needed.

Initial UUID Authentication

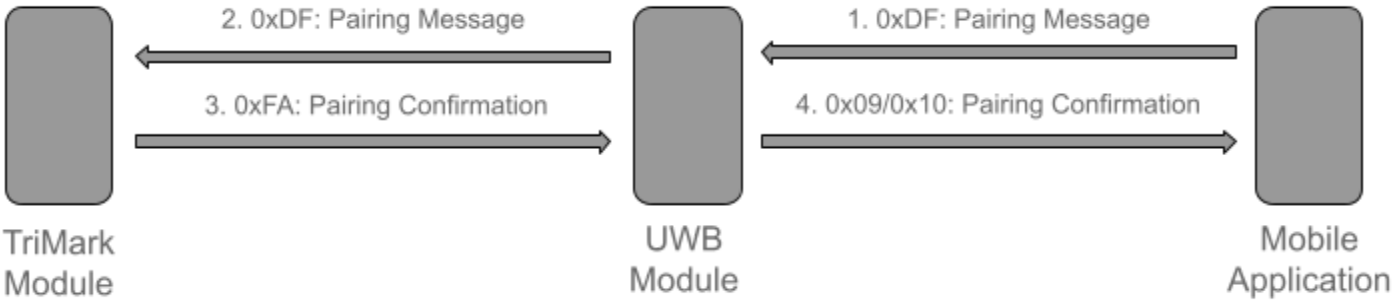
Upon connection of bluetooth between a UWB module and an application, the UUID of the application needs to be checked once with the authorized UUIDs stored in EEPROM.



1. **UWB Module:** BLE connection is established, UUID request sent to phone
    b. Type: 0xDD
2. **Phone:** UUID is sent to UWB module
    a. Type: 0xDD
3. **UWB Module**: UUID is received and forwarded to TriMark module
    a. Type: 0xDD
4. **TriMark Module**: Checks UUID in EEPROM, sends positive or negative response
    a. Type 0xDE/0xFB(if accepted)
5. **UWB Module**: Forwards positive/negative response to phone
    a. Type: 0x06/0x07/0x08
6. **Phone**: Either continues bluetooth/UWB session on positive response, stops connection on negative response, or displays that the TriMark module is in pairing mode

Initial Pairing Procedure

Upon initial UUID authentication, when the TriMark module is in pairing mode, a button can be pressed in order to pair the phone and enter the UUID into EEPROM.



1. **Phone:** Pairing mode pop-up button is pressed, sends UUID to UWB Module.

    c. Type: 0xDF
2. **UWB Module:** UUID is sent to UWB module
    a. Type: 0xDF
3. **TriMark Module**: UUID is received and added to the EEPROM. Confirmation sent to UWB Module
    a. Type: 0xDF
4. **Phone**: Forwards confirmation to phone, where message is displayed
    a. Type 0x09/0x10

Message Encryption

For this application, we will be using the AES-128 standard for encryption and decryption. Most of these source files have come from existing TriMark encryption, but we will add our own key and final encryption step.

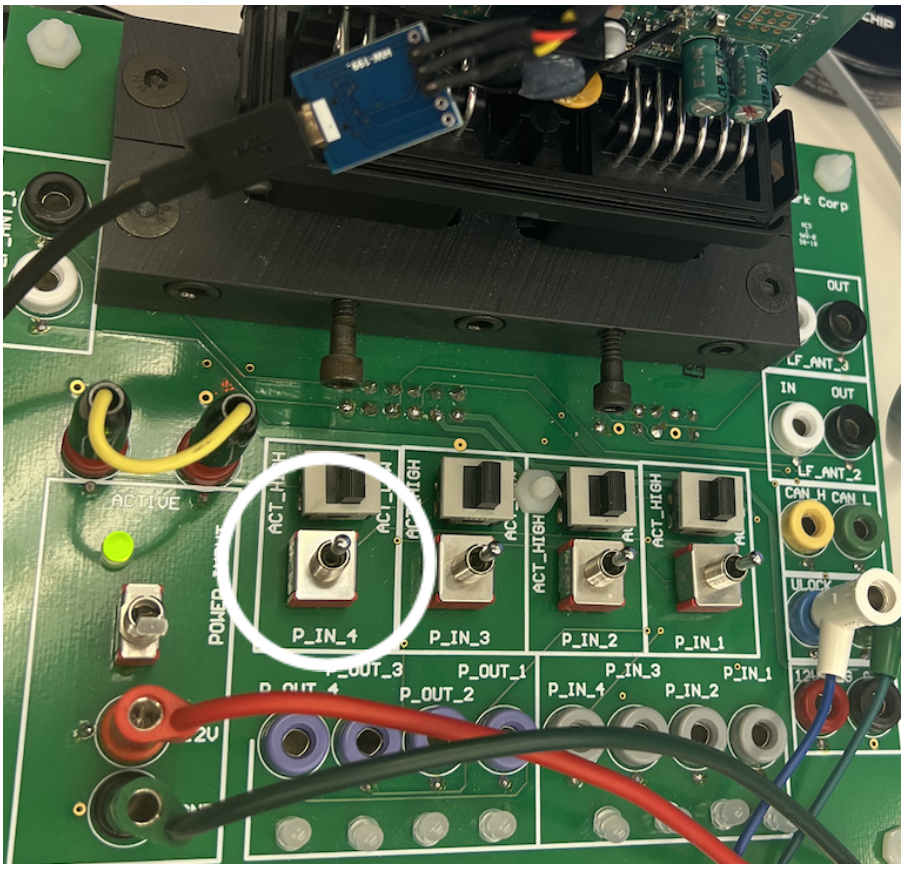
Encryption Key

The 16-byte encryption key that we will use is as follows:

0x3975AE2F 9D9264CA BFF26430 A4286DFA

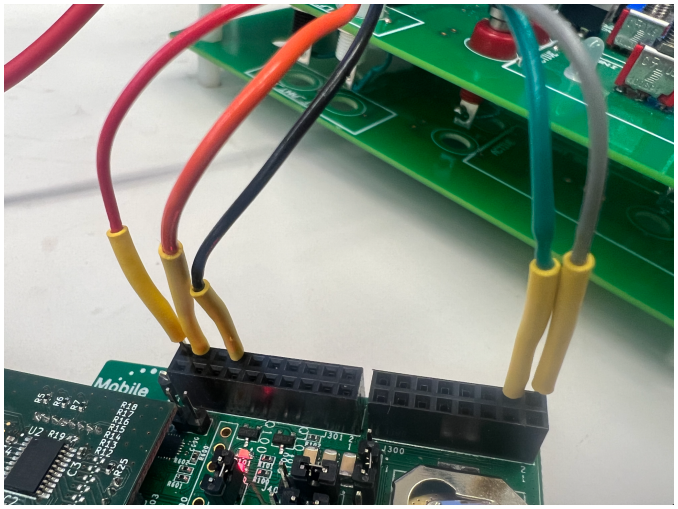
Initial Pairing Procedure

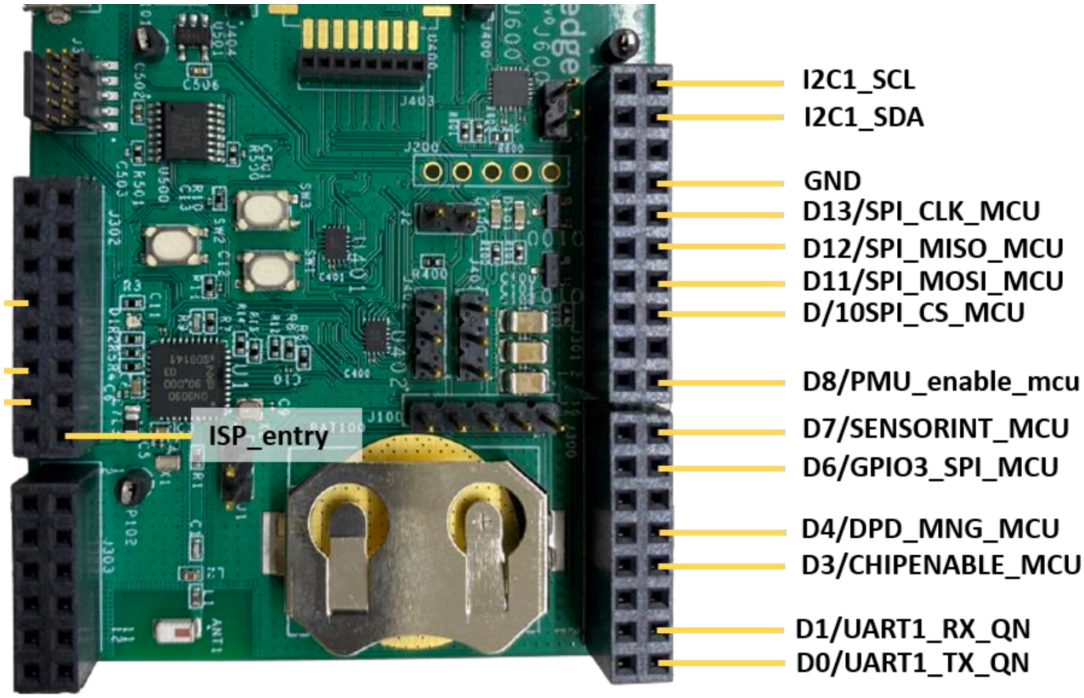
1. **TriMark Module**: Initiate Pairing Mode
    a. Pulse P_IN_4 3 times consecutively within 5 seconds
    b. Enter Pairing Mode & Lock Actuator signal
    c. Pairing Mode will time out after 60 seconds of no input or messages from phone
2. **Phone:** Must try to connect to bluetooth, press “Pair Phone” button when prompted
    a. Single unlock pulse when phone is added successfully
    b. When leaving pairing mode, unlock pulse
    c. After each phone is added, Pairing Mode is exited, and the sequence must be repeated to add another phone.



UWB Board -> TriMark Board Connection Guide

I2C1_SCL : Red Wire (CTS Input for UWB, CTS Output for TriMark)
I2C1_SDA : Orange Wire (CTS Output for UWB, CTS Input for TriMark)
GND : Black Wire (self-explanatory)
D1/UART1_RX_QN : Green Wire (UART RX for UWB, UART TX for TriMark)
D0/UART1_TX_QN : Gray Wire (UART TX for UWB, UART RX for TriMark)





Bluetooth in the Background

https://developer.apple.com/forums/thread/707775
https://developer.apple.com/videos/play/wwdc2022/10008/?time=1073
https://www.wwdcnotes.com/notes/wwdc22/10008/

Background sessions

Until now, when the app transitions to the background, or when the user locks the screen on iOS and watchOS, any running `NISession`s are suspended until the application returns to the foreground.

In iOS 16, we can connect to devices via BLE and then run a NISession in the background:

```
var niSession = NISession()

func runBackgroundSession(accessoryData: Data, peripheral: CBPeripheral) {
  // Provide the accessory's UWB configuration data, and its Bluetooth peer identifier
  let peerIdentifier = peripheral.identifier
  let config = NINearbyAccessoryConfiguration(
    accessoryData: accessoryData,
    bluetoothPeerIdentifier: peerIdentifier
  )
  session.run(config)
}
```

This helps create "hands-free" experiences, like having music start playing as soon as the user walks into a room.

This background mode requires the `Nearby Interaction` string in the `UIBackgroundModes` array in your app's `Info.plist`