VISA SensorTile Tutorial

Author: Stephen Sheldon

Introduction:

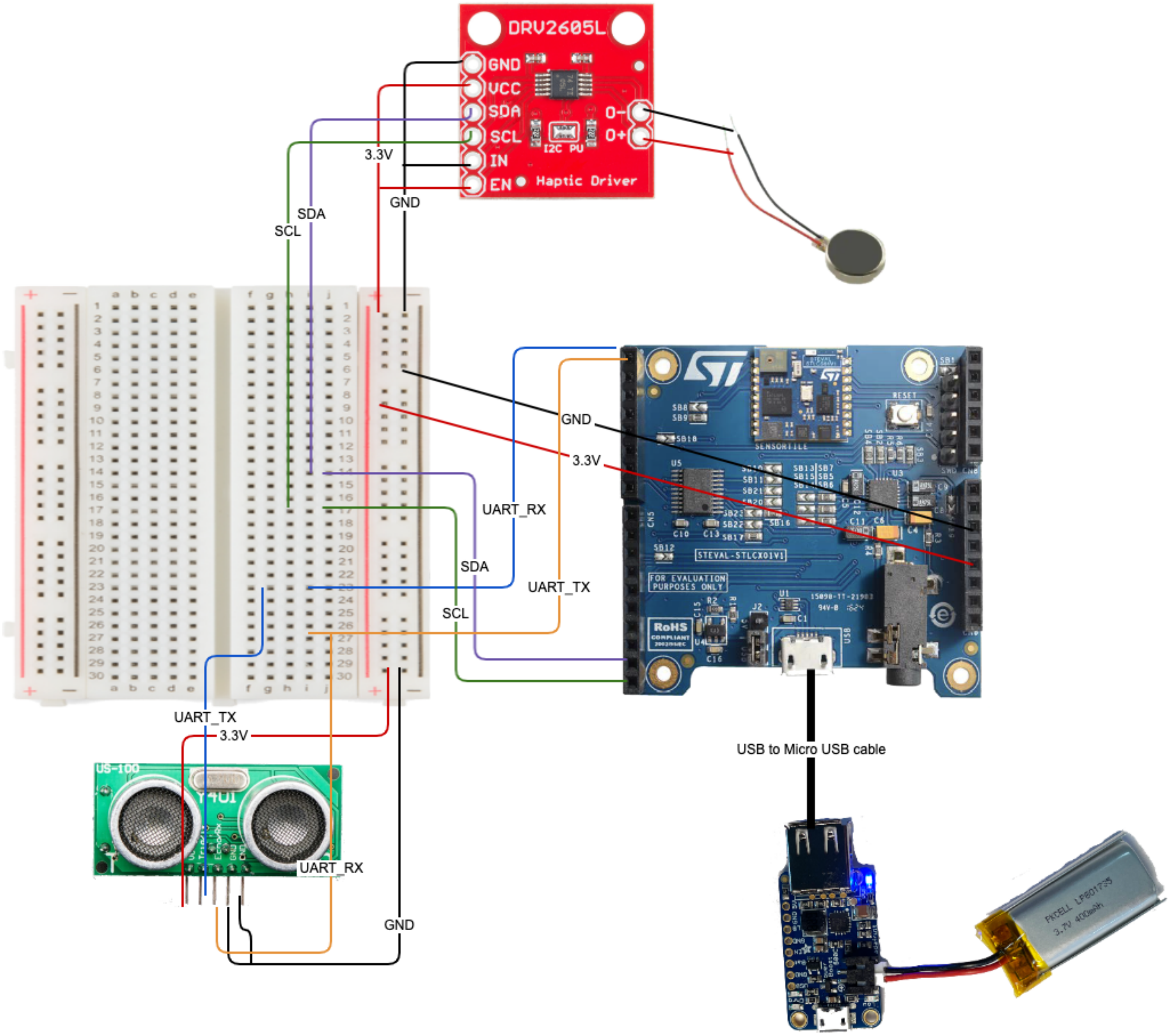
This tutorial guides you on setting up the VISA SensorTile project.

Parts:

- Sparkfun Haptic Motor Driver – DRV2605L (https://www.sparkfun.com/products/14538)
- Vibration Motor – B1034.FL45-00-015 (https://www.sparkfun.com/products/8449)
- STM SensorTile with SensorTile Cradle Expansion Board - STEVAL_STLKT01V1 (https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html)
- US-100 Ultrasonic Distance Sensor (https://www.adafruit.com/product/4019)
- Adafruit PowerBoost 500 + Charger (https://www.adafruit.com/product/1944)

Setting up the circuit:

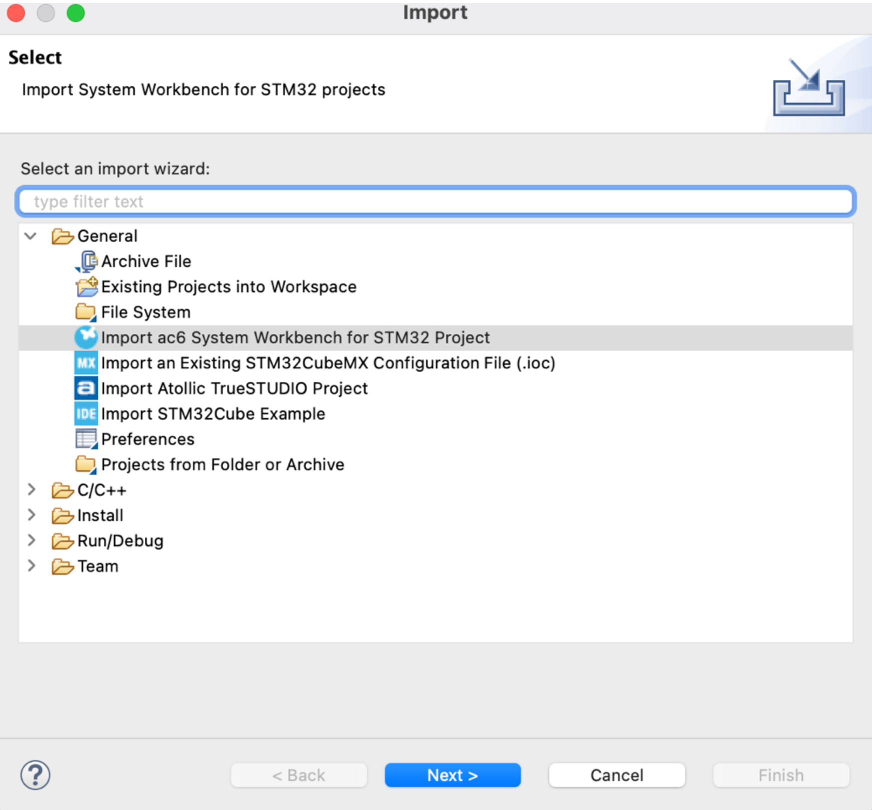
Utilize the following diagram and schematic to help setup the circuit.



Importing the VISA SensorTile project:

1. Download the VISA SensorTile Source code

2. Inside STM32CubeIDE Go to File → Import → import ac6 System Workbench for STM32 Project then click “next.”



Import

Select

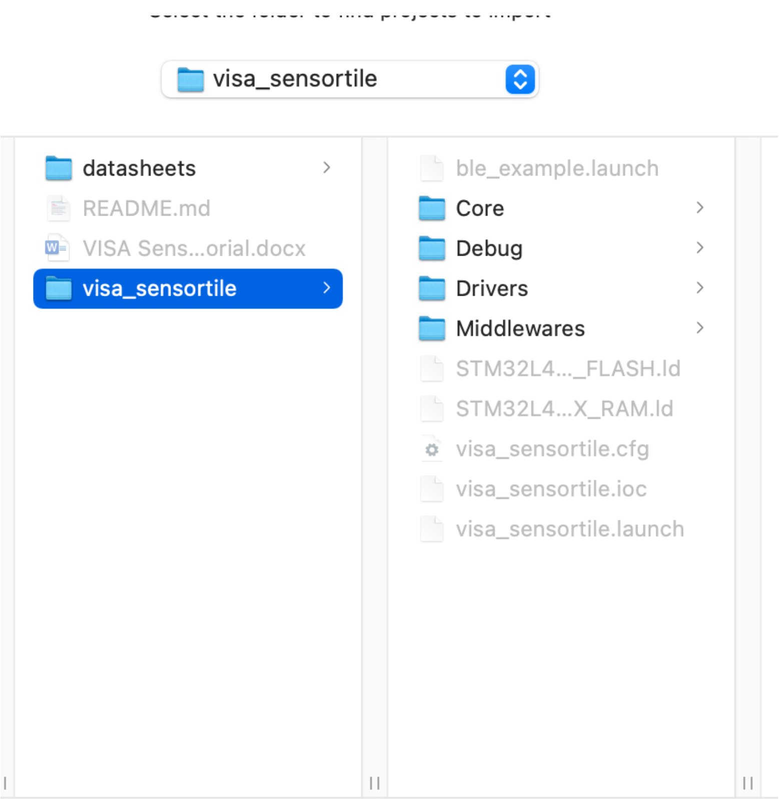
Import System Workbench for STM32 projects

Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Import ac6 System Workbench for STM32 Project
  - Import an Existing STM32CubeMX Configuration File (.ioc)
  - Import Atollic TrueSTUDIO Project
  - Import STM32Cube Example
  - Preferences
  - Projects from Folder or Archive
- C/C++
- Install
- Run/Debug
- Team

3. In the next screen you will select the visa_sensortile project folder from your download.



Select the folder to find projects to import

visa_sensortile

- datasheets
- README.md
- VISA Sens...orial.docx
- visa_sensortile

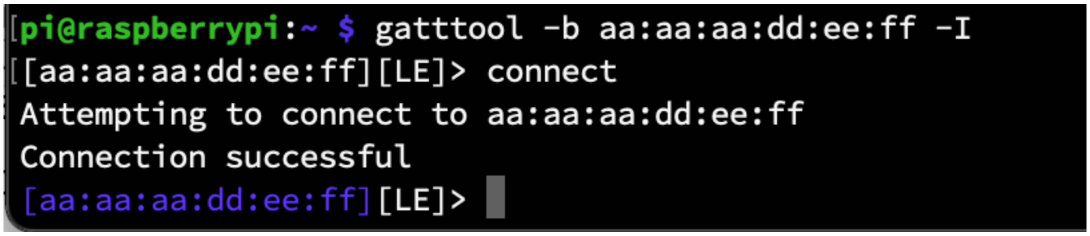
- ble_example.launch
- Core
- Debug
- Drivers
- Middlewares
- STM32L4..._FLASH.ld
- STM32L4...X_RAM.ld
- visa_sensortile.cfg
- visa_sensortile.ioc
- visa_sensortile.launch

4. Click the “Finish” button. Your visa_sensortile project is now imported and ready to be build and run.

Running the VISA SensorTile project:

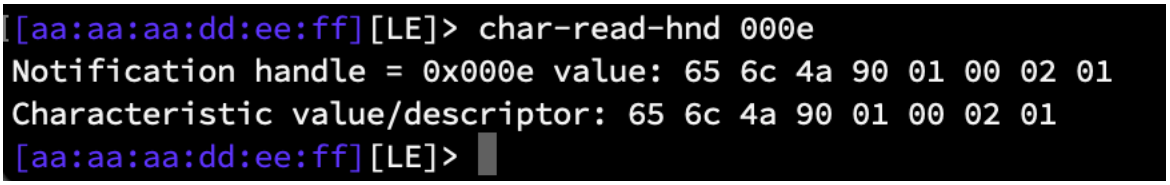
For this portion I am using a Raspberry Pi 4 to interact with the VISA SensorTile over BLE.

1. Run the VISA SensorTile code.
2. On your BLE client device connect to the SensorTile’s mac address which is **aa:aa:aa:dd:ee:ff**.



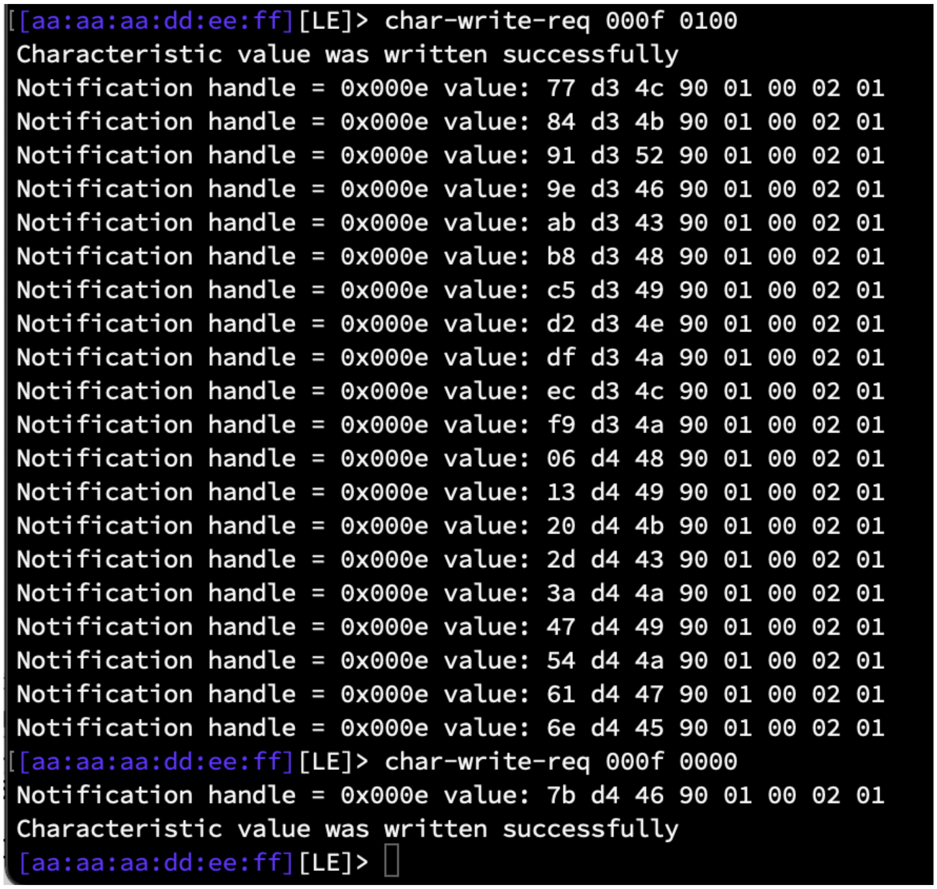
```
pi@raspberrypi:~ $ gatttool -b aa:aa:aa:dd:ee:ff -I
[aa:aa:aa:dd:ee:ff][LE]> connect
Attempting to connect to aa:aa:aa:dd:ee:ff
Connection successful
[aa:aa:aa:dd:ee:ff][LE]>
```

3. To read data from the distance sensor you can issue the following command: **char-read-hnd 000e**. The last two bytes returned will be the distance in little endian order. So, for example, in the below screenshot you have a distance of 0x0102 which is 258 millimeters when converted to decimal.



```
[aa:aa:aa:dd:ee:ff][LE]> char-read-hnd 000e
Notification handle = 0x000e value: 65 6c 4a 90 01 00 02 01
Characteristic value/descriptor: 65 6c 4a 90 01 00 02 01
[aa:aa:aa:dd:ee:ff][LE]>
```

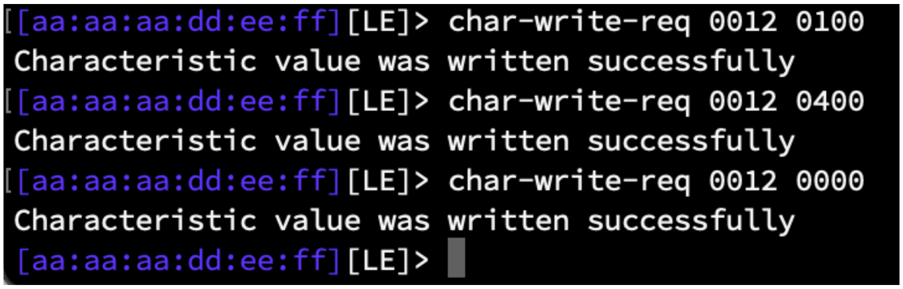
4. To stream distance sensor data over BLE you will need to modify the characteristic value of handle 0x000f as seen below. You can write 0100 to turn the stream on and 000 to turn it off.



```
[[aa:aa:aa:dd:ee:ff][LE]> char-write-req 000f 0100
Characteristic value was written successfully
Notification handle = 0x000e value: 77 d3 4c 90 01 00 02 01
Notification handle = 0x000e value: 84 d3 4b 90 01 00 02 01
Notification handle = 0x000e value: 91 d3 52 90 01 00 02 01
Notification handle = 0x000e value: 9e d3 46 90 01 00 02 01
Notification handle = 0x000e value: ab d3 43 90 01 00 02 01
Notification handle = 0x000e value: b8 d3 48 90 01 00 02 01
Notification handle = 0x000e value: c5 d3 49 90 01 00 02 01
Notification handle = 0x000e value: d2 d3 4e 90 01 00 02 01
Notification handle = 0x000e value: df d3 4a 90 01 00 02 01
Notification handle = 0x000e value: ec d3 4c 90 01 00 02 01
Notification handle = 0x000e value: f9 d3 4a 90 01 00 02 01
Notification handle = 0x000e value: 06 d4 48 90 01 00 02 01
Notification handle = 0x000e value: 13 d4 49 90 01 00 02 01
Notification handle = 0x000e value: 20 d4 4b 90 01 00 02 01
Notification handle = 0x000e value: 2d d4 43 90 01 00 02 01
Notification handle = 0x000e value: 3a d4 4a 90 01 00 02 01
Notification handle = 0x000e value: 47 d4 49 90 01 00 02 01
Notification handle = 0x000e value: 54 d4 4a 90 01 00 02 01
Notification handle = 0x000e value: 61 d4 47 90 01 00 02 01
Notification handle = 0x000e value: 6e d4 45 90 01 00 02 01
[[aa:aa:aa:dd:ee:ff][LE]> char-write-req 000f 0000
Notification handle = 0x000e value: 7b d4 46 90 01 00 02 01
Characteristic value was written successfully
[aa:aa:aa:dd:ee:ff][LE]>
```

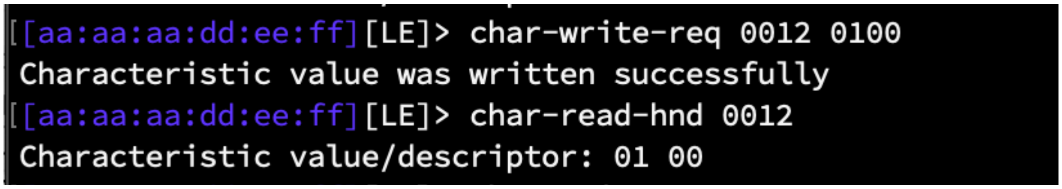
5. The haptic feedback can be controlled by modifying the characteristic value of handle 0x0012. This can take one of the following values. When you change the characteristic value, the haptic motor will give a vibration pattern unique to each hand movement.

Characteristic Value	Hand Movement
0000	Turn off haptic feedback
0100	Move hand right
0200	Move hand left
0300	Move hand up
0400	Move hand down
0500	Move hand forward
0600	Move hand backward



```
[[aa:aa:aa:dd:ee:ff][LE]> char-write-req 0012 0100
Characteristic value was written successfully
[[aa:aa:aa:dd:ee:ff][LE]> char-write-req 0012 0400
Characteristic value was written successfully
[[aa:aa:aa:dd:ee:ff][LE]> char-write-req 0012 0000
Characteristic value was written successfully
[aa:aa:aa:dd:ee:ff][LE]>
```

6. If you want to see what the current characteristic value is for the haptic feedback handle you can issue the following command: **char-read-hnd 0012**. You will receive the current characteristic value returned in little endian format.



```
[[aa:aa:aa:dd:ee:ff][LE]> char-write-req 0012 0100
Characteristic value was written successfully
[[aa:aa:aa:dd:ee:ff][LE]> char-read-hnd 0012
Characteristic value/descriptor: 01 00
```