[redacted]

[redacted] Harness Provisioning Tool Manual

13 November 2020

TABLE OF CONTENTS

REQUIREMENTS

- Windows computer with a copy of the provisioning software
    - Includes firmware and related firmware files
- Flash programmer to connect harness to computer serial port
- Harness with serial connection exposed
- WiFi SSID and password
- [redacted] login credentials for a user authorized to provision harnesses

APPLICATION OVERVIEW

This application is intended to provision the [redacted] Harness.

Three main tabs are at the top of the application. Begin Pre-assembly, Begin Post-assembly, and View Debug Log.

Pre-assembly

- Allows a technician to deprovision a previously provisioned harness, if desired
- Flashes firmware to the harness
- Places the harness in provisioning mode
- Allows a technician to input a harness serial number
- Runs basic tests on sensors
- Validates Bluetooth and Wi-Fi connectivity
- Logs harness serial port output for debugging purposes
- Sends harness serial information to the [redacted] API
- Erases test data
- Places the harness back into standard operating mode

Post-assembly

- Updates the [redacted] API to assign a size to the harness corresponding with a given serial number and harness size. **Note: This does not require a serial connection to the harness.**

Debug Log

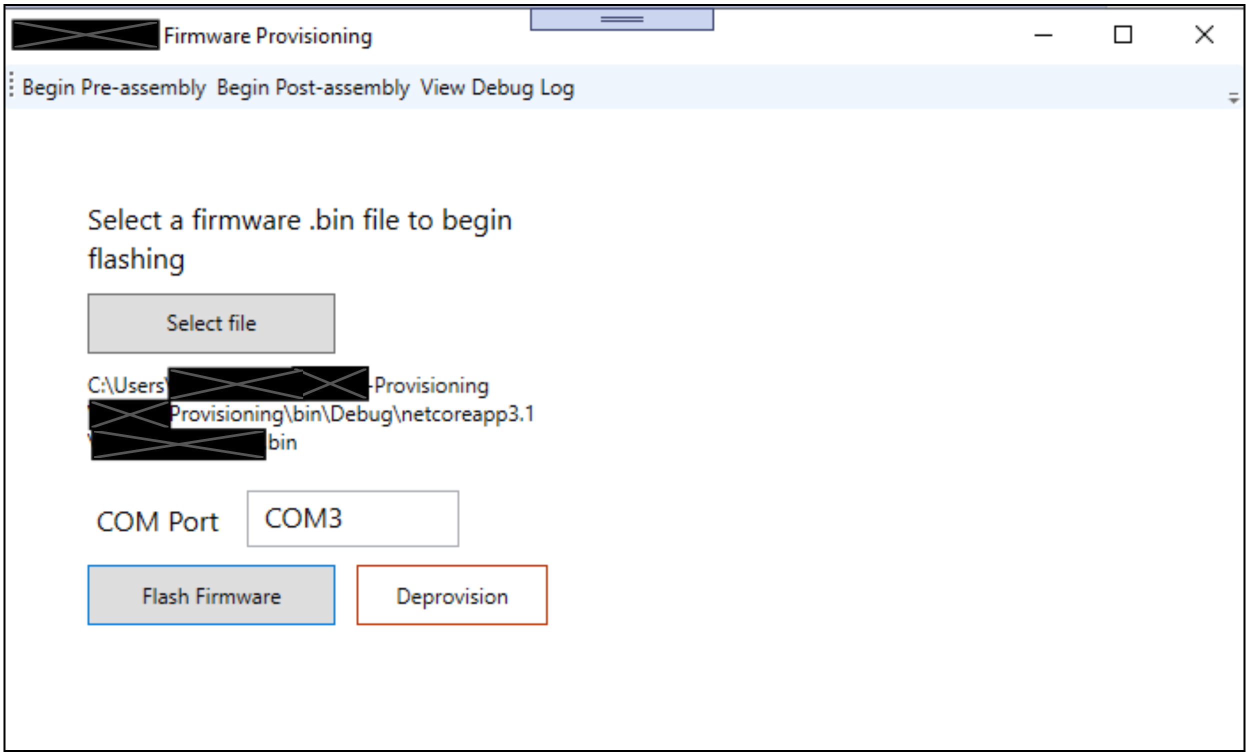
- Saves the harness serial output during provisioning to a text file for debugging purposes.

PROVISIONING APPLICATION SCREENS

Load Firmware

This is the screen that loads when you open the app. At the top of the screen are three options. The options are:

- Begin Pre-assembly
- Begin Post-assembly
- View Debug Log



Select a firmware .bin file to begin flashing

C:\Users\[redacted]Provisioning
[redacted]Provisioning\bin\Debug\netcoreapp3.1
[redacted]bin

COM Port: COM3

Fig. 1

When you first run the app, it will default to the Begin Pre-assembly view. (Fig. 1)

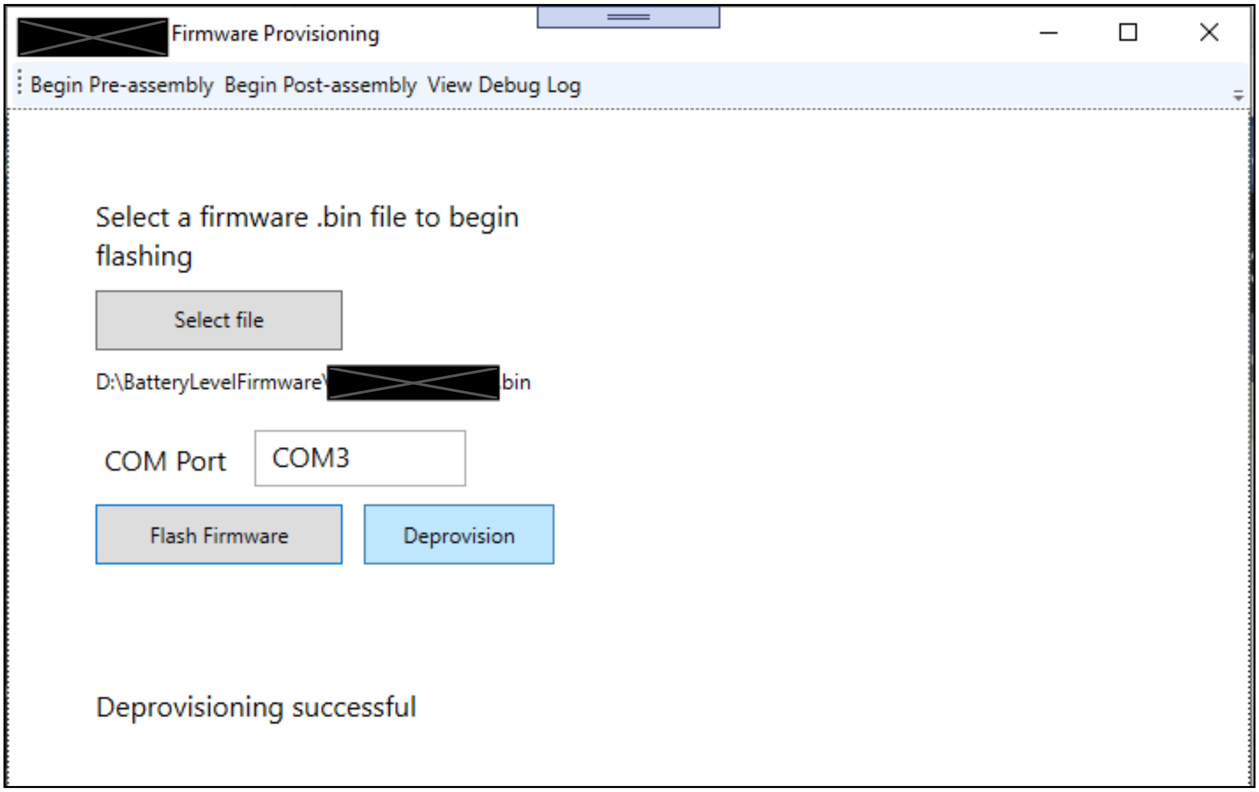
Steps On This Screen

If the harness has already been provisioned, press the Deprovision button.

There will be a series of messages that flash as the deprovisioning occurs. When it is done the screen should look like Fig. 2. After deprovisioning, you may reprovision by following the steps below.

If this is a new harness, or has previously been deprovisioned, follow the steps below.

1. The firmware binary file [redacted]**.bin** should be pre-selected. Check to see that the path to the file is correct. Clicking the **Select File** button launches a file picker; if necessary you can use this to select a new firmware binary.
2. Type in the name of the COM port that your flash programmer is plugged into on your Windows machine. This is often COM3 or COM4, but can be discovered by looking at Device Manager.
3. If this is a new harness, hit the Flash Firmware button. The application will go through several steps as it flashes the firmware.
4. If it is successful, the application will automatically advance to the next screen.
5. If the firmware flash fails, It should display output similar to Fig. 3 that should help identify the cause of the failure.



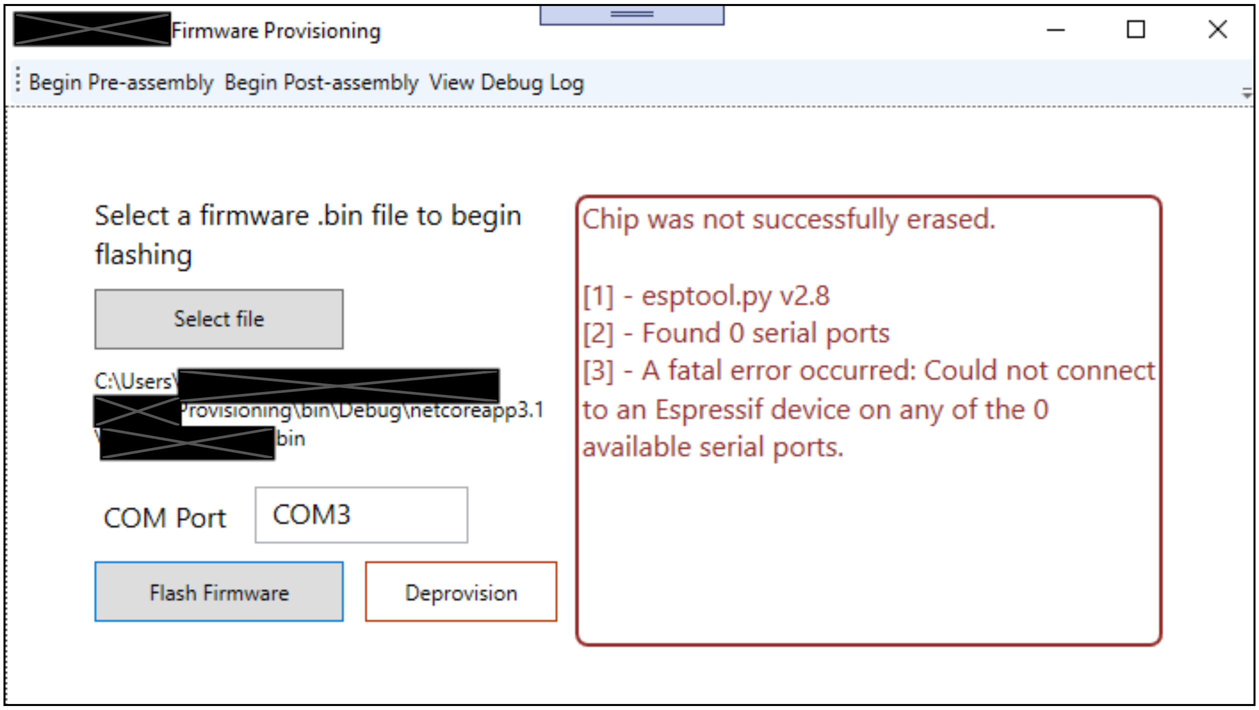
Select a firmware .bin file to begin flashing

D:\BatteryLevelFirmware\[redacted]bin

COM Port COM3

Deprovisioning successful

Fig. 2



Select a firmware .bin file to begin flashing

C:\Users\[redacted]
[redacted]rovisioning\bin\Debug\netcoreapp3.1
\[redacted]bin

COM Port: COM3

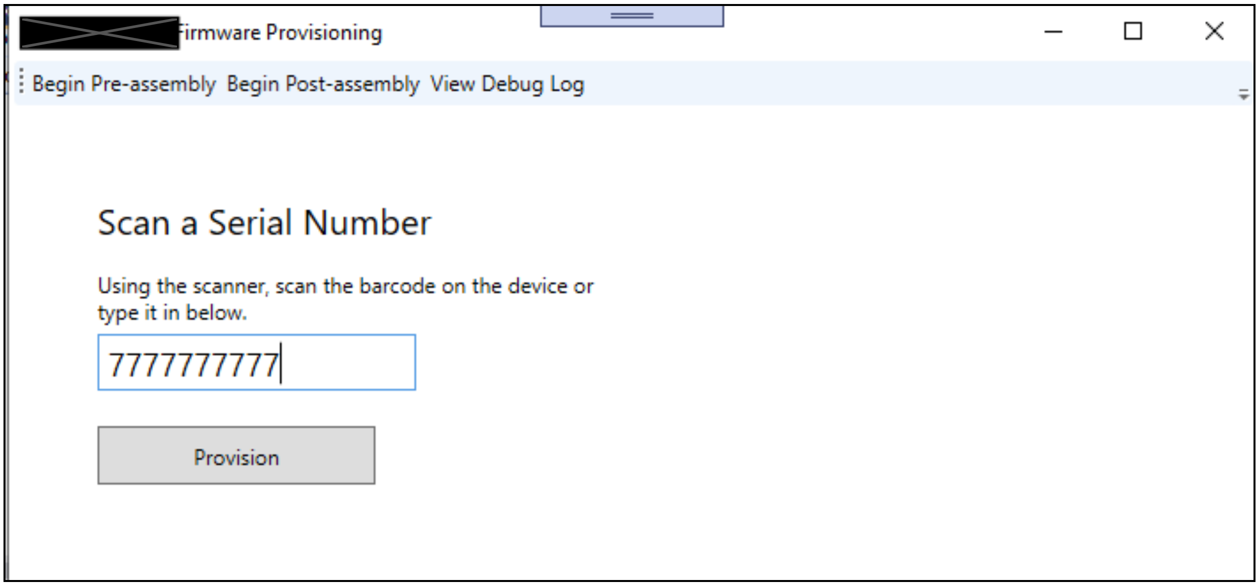
```
Chip was not successfully erased.

[1] - esptool.py v2.8
[2] - Found 0 serial ports
[3] - A fatal error occurred: Could not connect
to an Espressif device on any of the 0
available serial ports.
```

Fig. 3

Serial Number Input

Steps on This Screen

1. The technician must scan or type in a serial number. It must be 10 digits in length and should be the number from the board. (Fig. 4)



Scan a Serial Number

Using the scanner, scan the barcode on the device or type it in below.

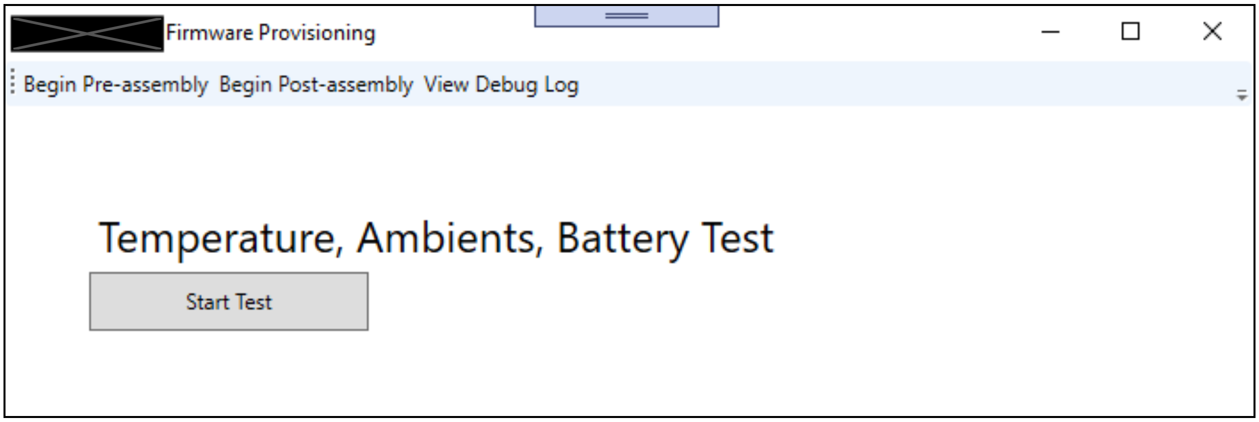
7777777777

Fig. 4

Test Sensors

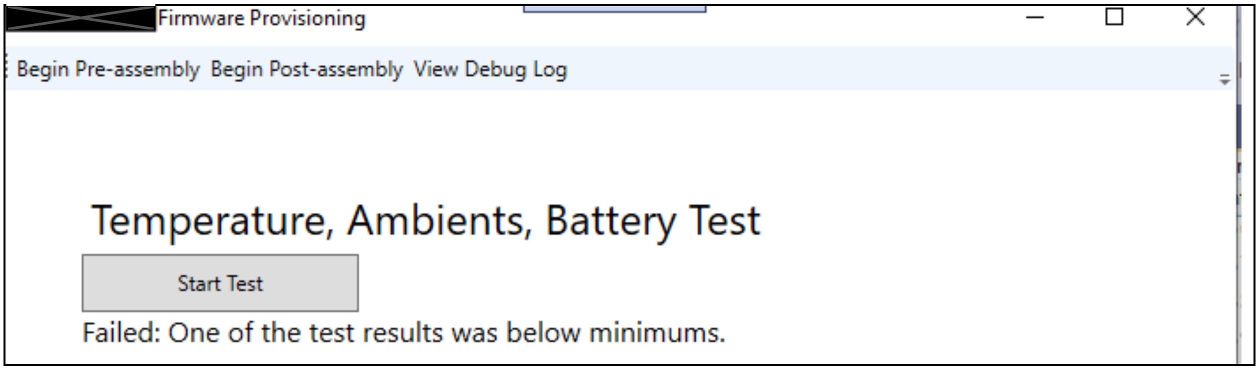
Steps on This Screen

1. The first test will verify the temperature sensors, ambient humidity and temperature sensors, and the battery read valid values. Click the **Start Test** button to begin. (Fig. 5)
2. Results display below the button. (Fig. 6)
3. The second test evaluates the respiratory, ECG and heart rate sensors and the accelerometer. It requires the harness to be flipped over 180 degrees before being run. Click the **Start Test** button to begin. (Fig. 7)
4. Results display below the button.
5. Both of these tests can be repeated upon failure, if required, by re-clicking the **Start Test** button.



Temperature, Ambients, Battery Test

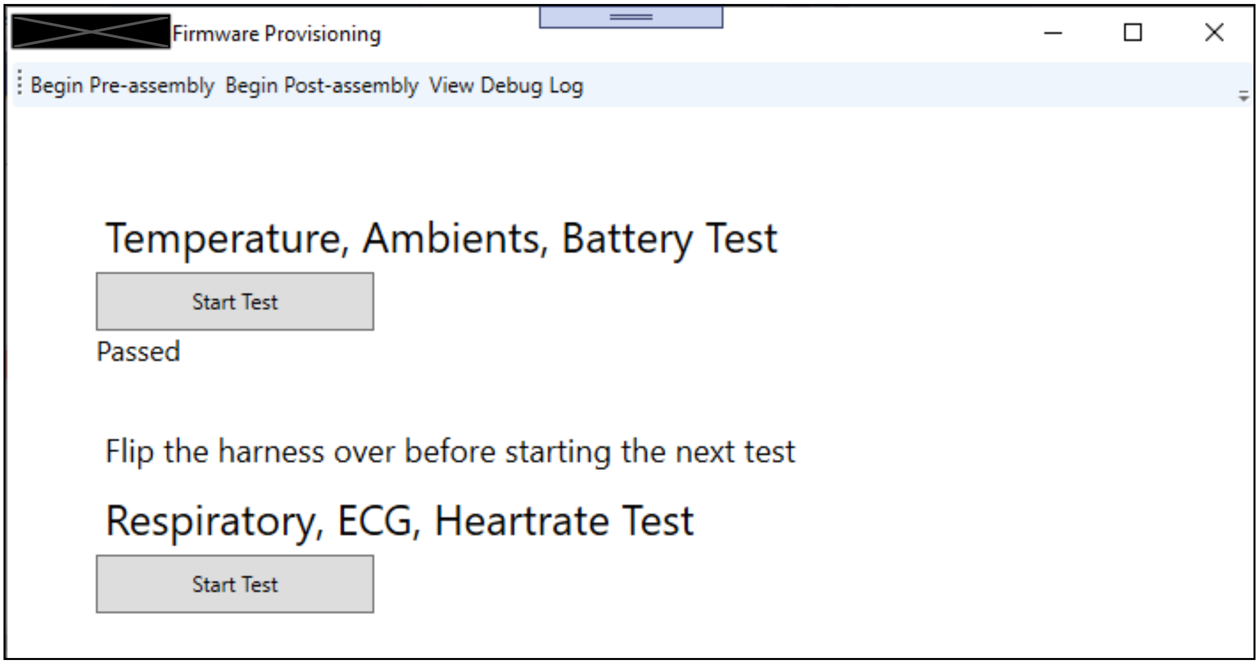
Fig. 5



Temperature, Ambients, Battery Test

Failed: One of the test results was below minimums.

Fig. 6



Temperature, Ambients, Battery Test

Passed

Flip the harness over before starting the next test

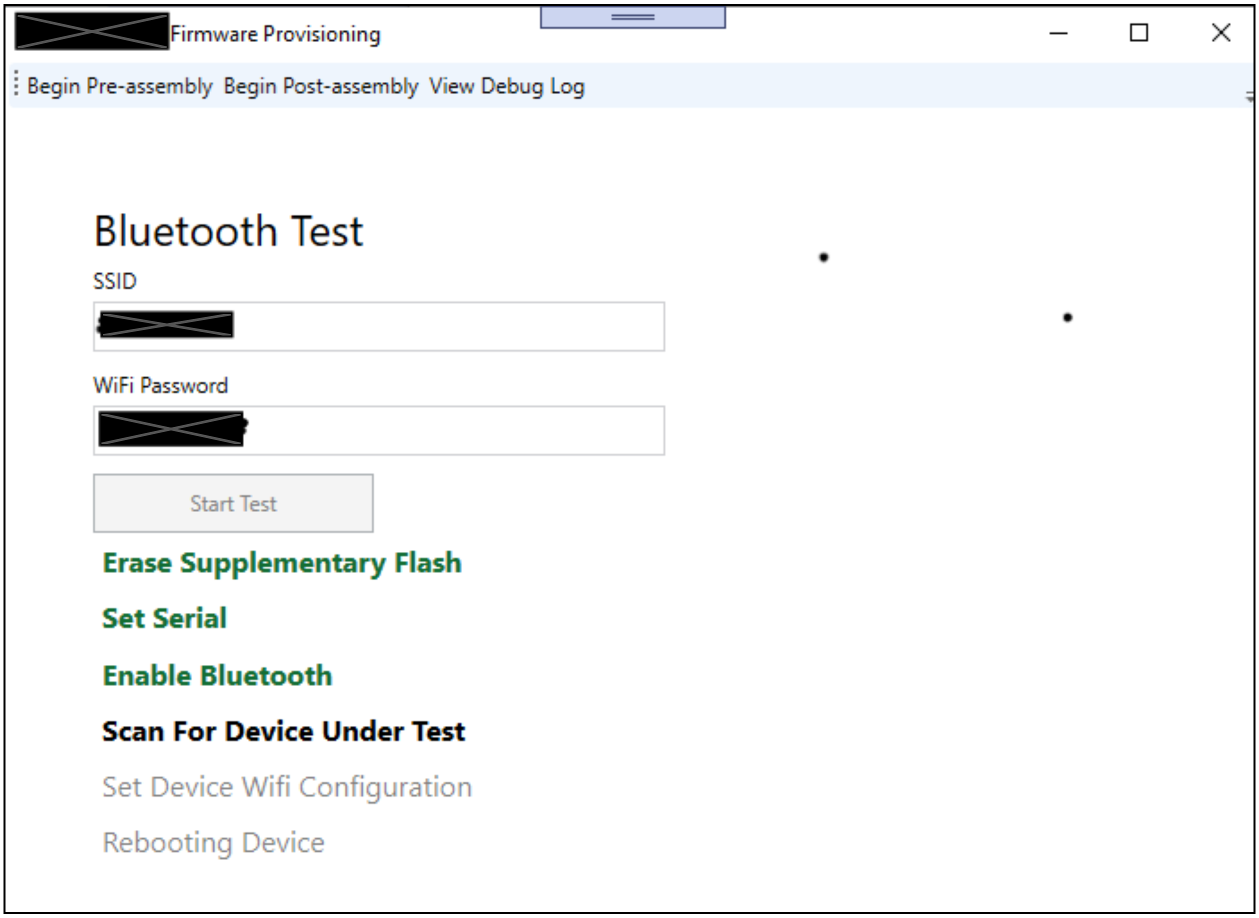
Respiratory, ECG, Heartrate Test

Fig. 7

Test Bluetooth/WiFi

Steps on This Screen

1. Enter a WiFi SSID and password for the network you want to use.
1. Click the **Start Test** button.
2. This test will run through a number of steps, and then eventually reboot the device. (Fig. 8)
3. If the test is successful and authentication credentials are not stored already, the user will be prompted to authenticate. (See Authentication).
4. After this test has succeeded the application will send harness information to the [redacted] API. If that harness serial number is a duplicate, the application will ask if this is intentional. If so, the user should hit **Retry.** (Fig. 9)



Bluetooth Test

SSID

[redacted]

WiFi Password

[redacted]

Erase Supplementary Flash

Set Serial

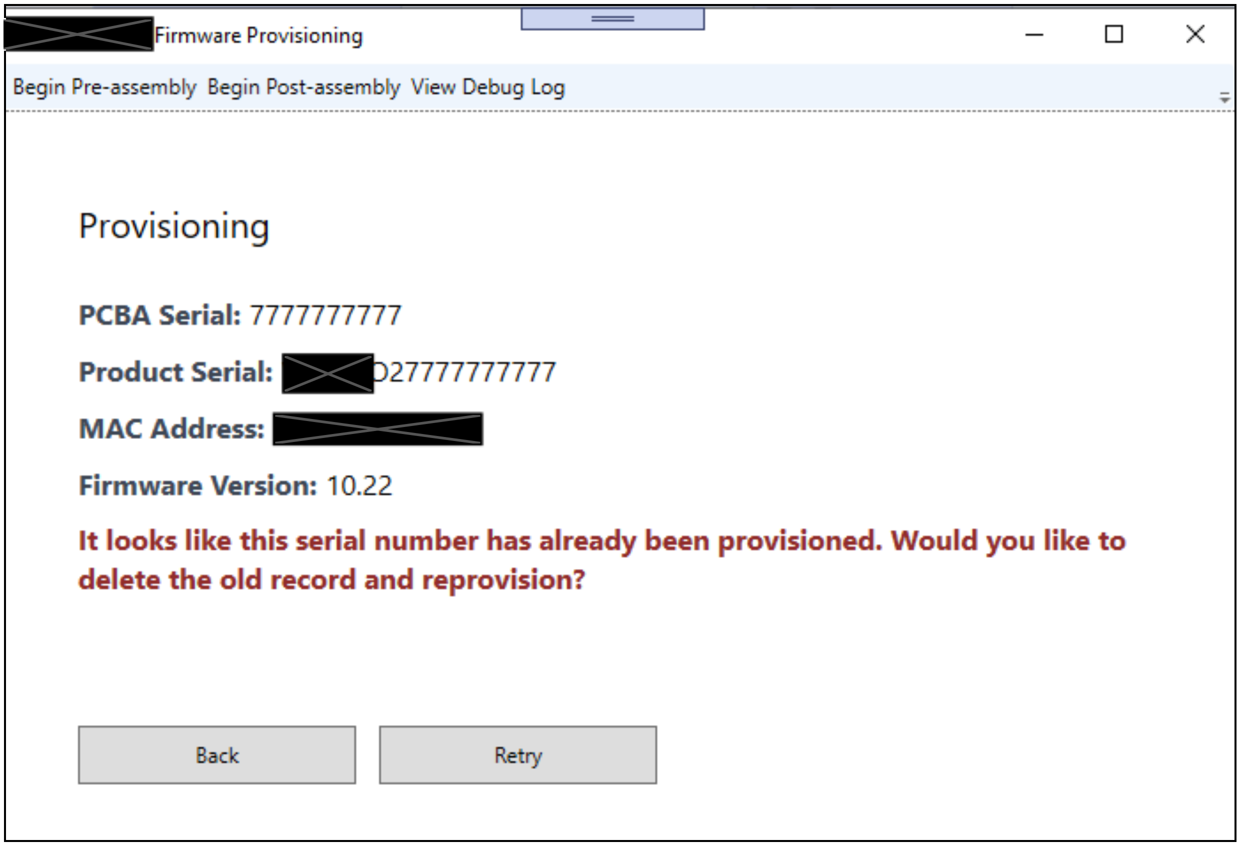
Enable Bluetooth

Scan For Device Under Test

Set Device Wifi Configuration

Rebooting Device

Fig. 8



Provisioning

PCBA Serial: 7777777777

Product Serial: [redacted]027777777777

MAC Address: [redacted]

Firmware Version: 10.22

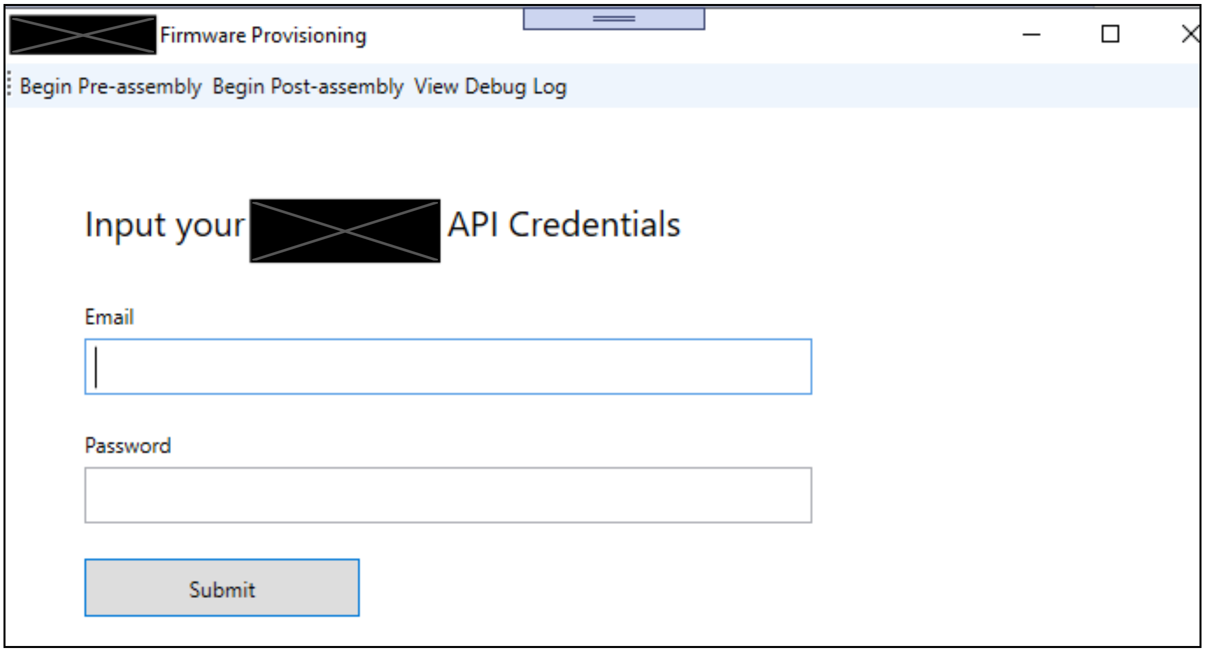
It looks like this serial number has already been provisioned. Would you like to delete the old record and reprovision?

Fig. 9

Authentication

Steps on This Screen

1. If necessary, the application will display an authentication screen. The user must type in a login and password that has been registered for provisioning harnesses on the [redacted] API. (Fig. 10)

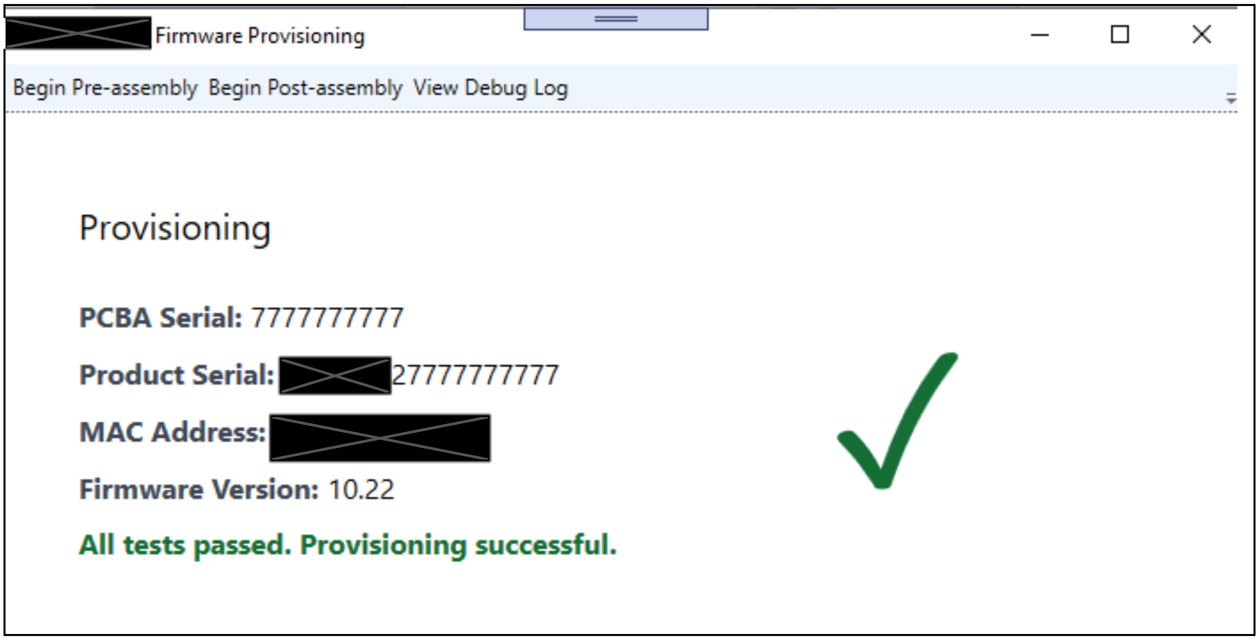


Input your [redacted] API Credentials

Fig. 10

Finish and Reboot

1. The app will automatically enter this step after the bluetooth test is complete and the application has posted the harness information to the API. The app will run through a WiFi test and remove test data. Provisioning is successful when the app displays a green checkmark. (Fig. 11)
2. If unsuccessful, the app will display an error on this view.



Provisioning

PCBA Serial: 7777777777

Product Serial: [redacted]27777777777

MAC Address: [redacted]

Firmware Version: 10.22

All tests passed. Provisioning successful.

Fig. 11

POST-PROVISIONING SCREEN

There is one step that must be completed after the harness has been put into a chassis. This is because the [redacted] API needs to know the harness size prior to sending harnesses to customers.

The harness does not need to be connected via serial port for this step.

The user must select the **Begin Post-assembly** tab at the top of the Provisioning application in order to enter this screen.

Set Harness Size

Steps on This Screen

1. Scan or enter the Product Serial Number. This should be a 16-digit serial beginning with VMVM02
2. Enter the harness size corresponding to the size of the chassis. (Fig 12)
3. Click the **Save** button.
4. If an incorrect serial number is entered or the application fails to record the size, the screen will display an error.

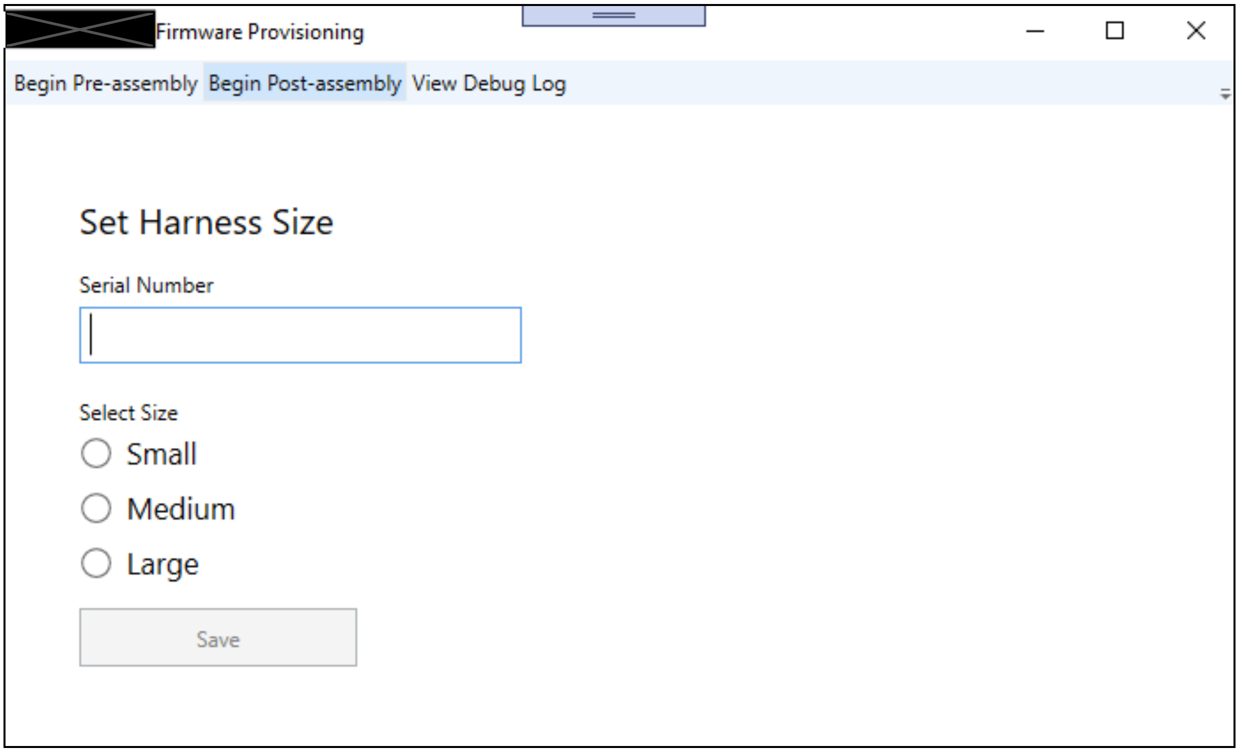


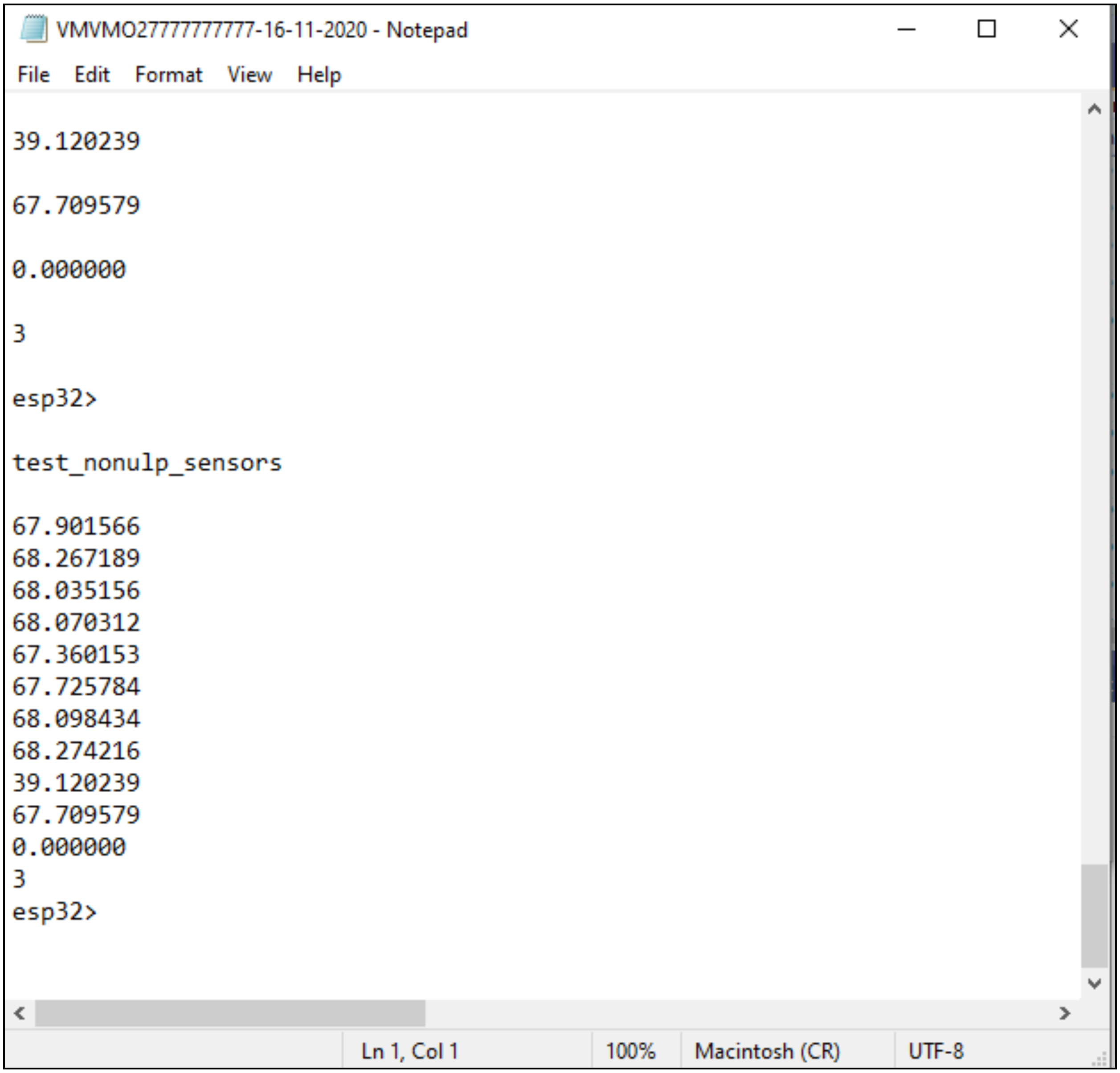
Fig. 12

DEBUG LOG

The debug log records the serial output from the harness. This log is saved in the application file while tests are running, and is deleted after provisioning is completed.

If a test fails or the application fails, the technician can open the debug log by clicking the **View Debug Log** button at the top of the application. They can send this log to [redacted] for support. This will be helpful when debugging issues.

A debug log could look something like the output in Fig 13.



```
39.120239

67.709579

0.000000

3

esp32>

test_nonulp_sensors

67.901566
68.267189
68.035156
68.070312
67.360153
67.725784
68.098434
68.274216
39.120239
67.709579
0.000000
3
esp32>
```

Fig. 13