

Product Test Plan

Vehicle Safety and Security Monitor (VSSM)

Author: Che Morrow

Date: 4/15/2021

I. INFORMATION REQUIRED FOR EXECUTION

PURPOSE

SCOPE

RESPONSIBILITIES

REFERENCES

DEFINITIONS

EQUIPMENT/SUPPLIES

PRECAUTIONS & WARNINGS

II. TESTING FEATURES & FUNCTIONS

TESTING FEATURES & FUNCTIONS

III. OTHER TESTING

INSTALLATION REQUIREMENTS / POWER SUPPLY REQUIREMENTS

I. INFORMATION REQUIRED FOR EXECUTION

A. Purpose

The purpose of this test protocol is to verify the full and complete operation of the VSSM.

B. Scope

This test protocol should be executed to verify that each principal feature and function performs within specification called out by the engineering requirements document. In addition, other necessary specifications shall be tested such as specific and necessary user, installation and power requirements.

The testing called out in this protocol is subject exclusively to those selected specifications provided for the VSSM 1.0. It shall not include any specification for future versions of the VSSM.

C. Responsibilities

It is the responsibility of the assigned test engineer to execute all tests included herein to the best of their ability. If necessary, seek additional assistance to execute tasks.

D. References

Voice.py:

Python-SoundDevice

<https://python-sounddevice.readthedocs.io/en/0.4.1/>

Vosk

<https://alphacephei.com/vosk/>

PiCamera

<https://picamera.readthedocs.io/en/release-1.13/>

GPS_Upload.py:

GPSD

<https://gpsd.gitlab.io/gpsd/>

PyTZ

<https://pypi.org/project/pytz/>

DateUtil

<https://dateutil.readthedocs.io/en/stable/>

PyUSB

<https://pypi.org/project/pyusb/>

OBD_Upload.py

Python-OBD

<https://python-obd.readthedocs.io/en/latest/>

OBD_Upload.py/Voice.py/GPS_Upload.py:

Requests

<https://pypi.org/project/requests/>

Website:

Maps API

<https://developers.google.com/maps/documentation/javascript/overview>

E. Definitions

HDMI: High-Definition Multimedia Interface

USB: Universal Serial Bus

UPS: Uninterruptible Power Supply

DTC: Diagnostic Trouble Codes

GPS: Global Positioning System

ISO: International Standards of Organization

F. Equipment/Supplies

- 1) Monitor with HDMI input
- 2) USB Keyboard and Mouse
- 3) Microphone included with VSSM
- 4) Lab bench power supply
- 5) GPS enabled device
- 6) ~5 GiB of test files
- 7) Device able to receive SMS messages

G. Precautions & Warnings

This test plan contains certain warnings and cautions designed to alert the test engineer while performing tests. The following illustrates each of these messages and how to recognize them.

WARNING: <message>

The “WARNING: Message” alerts the user about safety issues that are of the highest importance, such as possible injury to the operator.

CAUTION: <message>

The “CAUTION: Message” alerts the user to issues concerning possible damage to the equipment or that can lead to erroneous test results

II TESTING FEATURES AND FUNCTIONS

Customer Need	Functional Element	Physical Element	“Software” Element	Test Number
User can find location of vehicle.	Determining Vehicle Position	VK-162 GPS Dongle	Location Processing/Upload Script (Python)	1.1,1.2
Users can find vehicle location and obtain DTCs through the website	Internet Connection	SixFab 4G Base Hat	Any Script Sending Data to Website	2.1
The device records a video of the cabin if the user gives command.	Security Recordings	Pi Camera Module v2	Video recording script (Python), video upload script (Python)	3.1,3.2,3.3,3.4,3.5
User can read DTCs using vehicle’s OBD-II port.	Car Diagnostics	ELM327 OBD-II Adapter	OBD Data Acquisition Script (Python)	4.1,4.2
Users can find vehicle location and obtain DTCs through the website	Delivering Data to Users	Local Windows Server	Website (HTML, PHP, Javascript)	5.1
Day to day use does not require special knowledge. Works with vehicle OBD-II port, power.	Powering Device with Vehicle	MoPower UPS	UPS Firmware	6.1,6.2
Users can give command to contact EMS.	Contact with Emergency Services	Quectel EC25 Mini PCIe 4G/LTE Module	AT Command Script	7.1
The user interacts through voice commands	Retrieving User Voice Data	Bluetooth Lavalier Microphone	Bluetooth Configuration	8.1
The user interacts	Processing User		Vosk API	9.1,9.2

through voice commands	Voice Data		(Python)	
Day to day use does not require special knowledge.	Startup Processes and Exception Handling	Raspberry Pi 3 B+	“Main” Program (Python)	10.1,10.2

1. VK-162 GPS Dongle, Location Processing/Upload Script (Python)

1.1 – Determining Locational Accuracy

We can use a commercial GPS service to make sure the position determined by the VSSM is accurate.

Equipment Required:

Cell phone or similar GPS enabled device.

Input:

VSSM, powered through vehicle or bench supply.

Output:

Location uploaded to the website located at <http://vssm.ddns.net/map.php>.

Pass Criteria:

Location determined by the VSSM should be within 200 ft. of the location determined by a correctly operating test device. This distance is accurate enough to find the vehicle within the average American city block.

Fail Criteria:

Location on website is farther than 200 ft. away from the location determined by the test device.

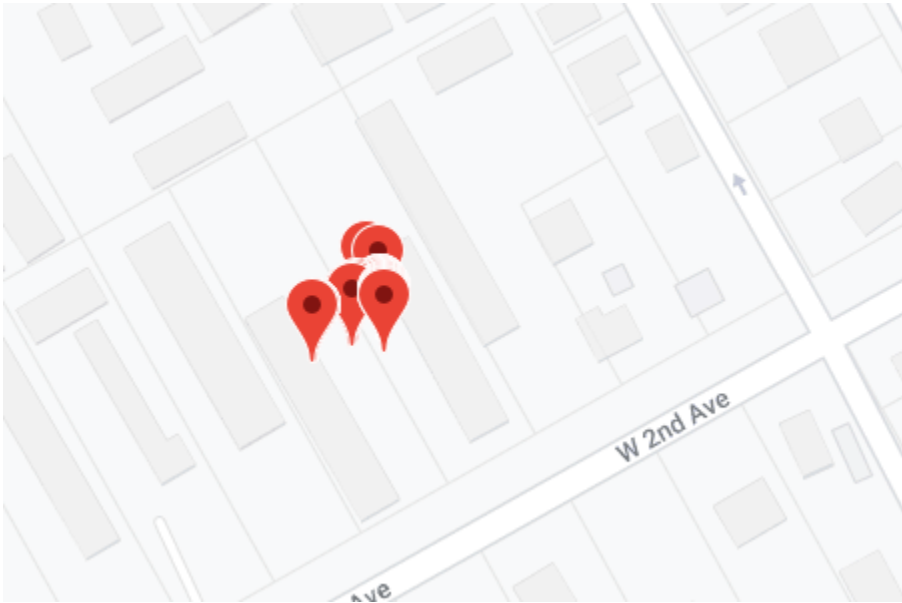
Test Procedure:

If operating within the vehicle, just connect the VSSM to the vehicle and wait for the information to be sent to the website. Then, compare the location of the pin to the location determined by the commercial service.

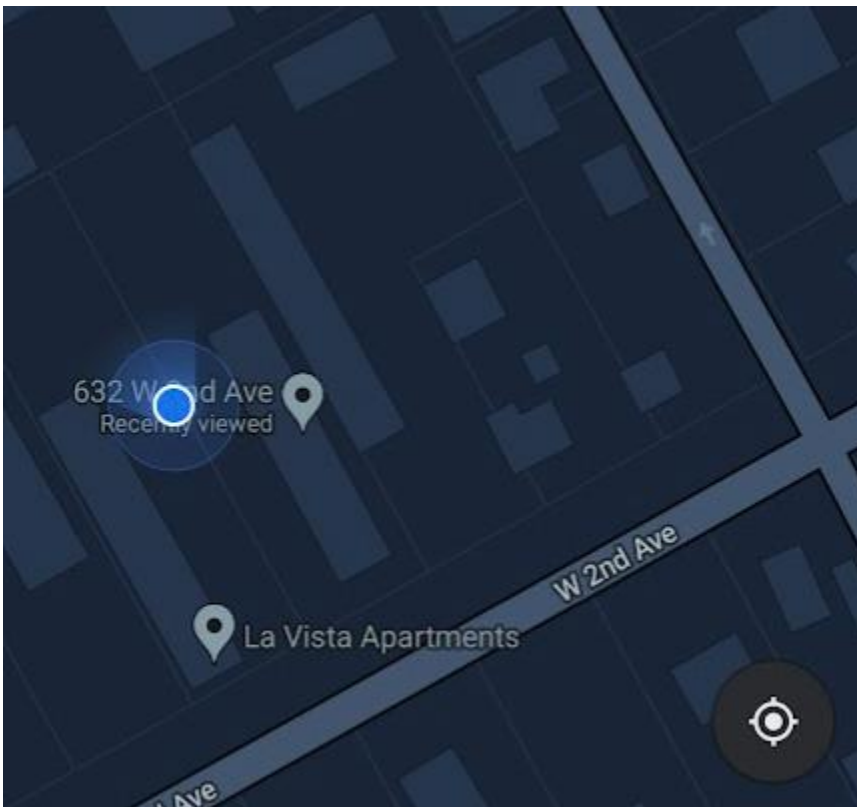
If working from a lab bench, make sure that the GPS device is close to a window. The rest of the process is the same.

Test Results:

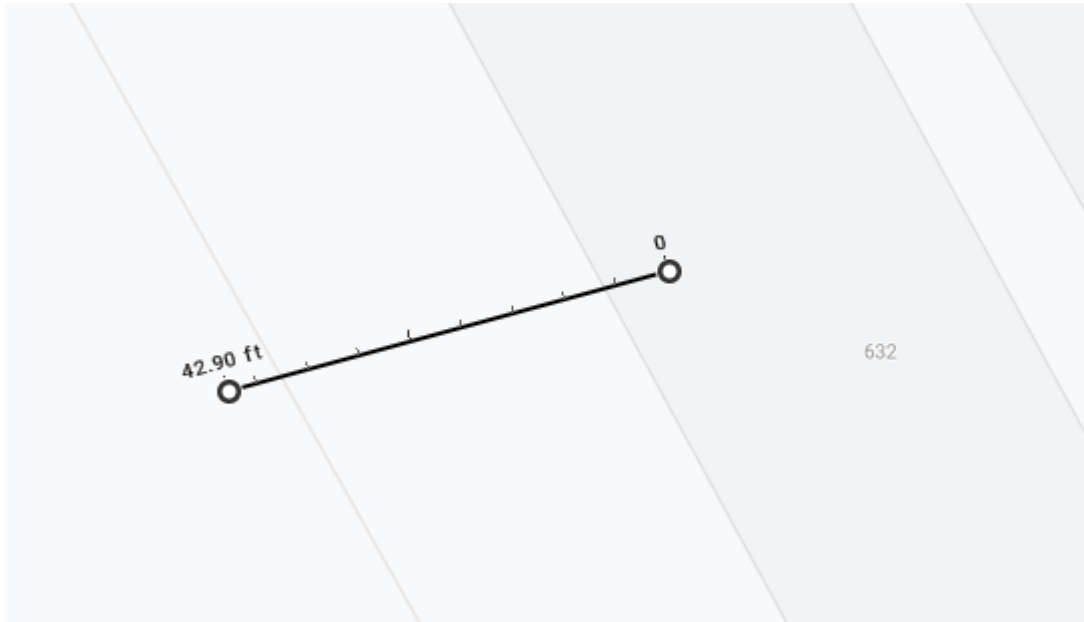
My test was taken at my lab bench with the GPS device placed near a window. After waiting for the data to be uploaded, I captured an image of the pin to compare to a commercial service. I waited long enough for multiple points to be added to the database, and these results were left on the “4-20-2021” entry on the website.



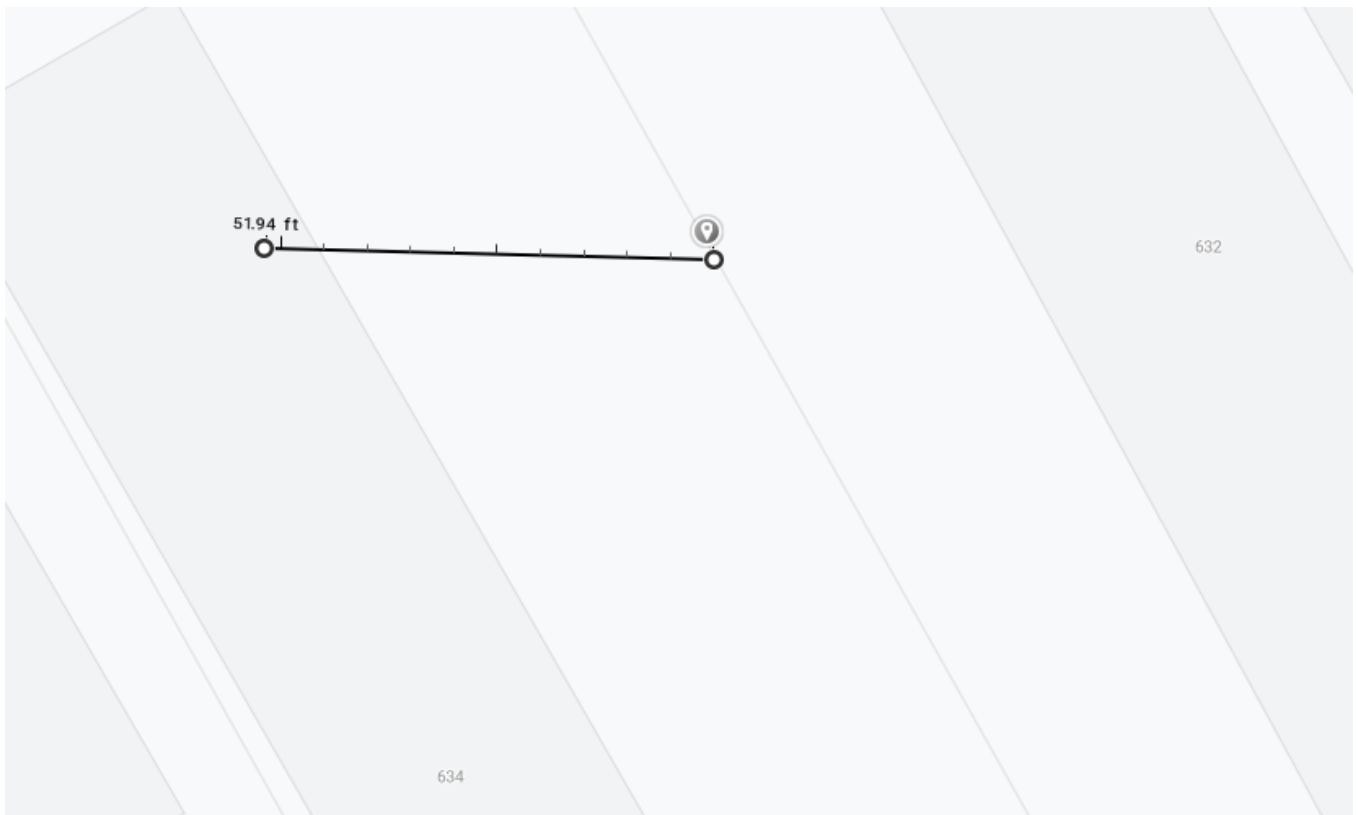
To verify I used the Google Maps app on a cell phone to find my location.



Using the measure tool, we can see that my actual location was almost 50 ft. from what my cell phone was able to determine.



Measured from the point my phone determined, the worst performance of the GPS unit was about the same at a little over 50 ft.



Test Analysis:

The GPS unit showed a high enough accuracy to determine which parking lot the car was located in. This is the precision expected from a low-cost GPS device and is perfectly reasonable for our application.

1.2 – Hardware Error Detection

The GPS script cannot determine the vehicle's location unless the device is connected to the Raspberry Pi's USB. Therefore, the script should return an error if the device is not connected.

Equipment Required:

Monitor and keyboard/mouse connected to Pi.

Input:

Removed GPS device.

Output:

Script failure reported by program.

Pass Criteria:

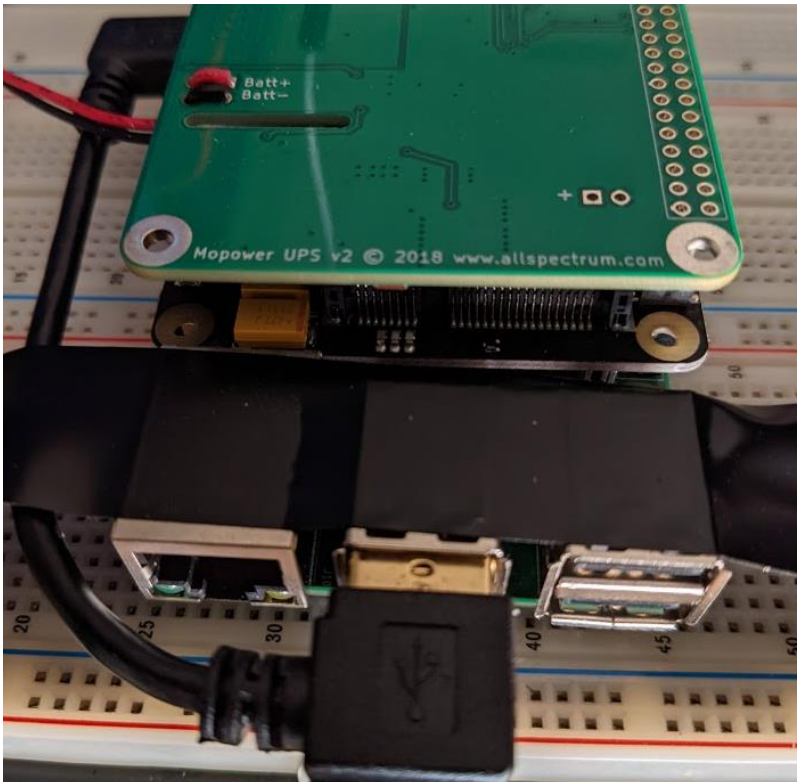
The GPS script should not attempt to determine the user's location without the GPS device in the USB port. In addition, the script should report this error in some way.

Fail Criteria:

GPS script continues to operate without the GPS device.

Test Procedure:

Remove the GPS device from the lower right USB port. Here is a picture of the device after the GPS is removed.



Then, attempt to launch the GPS script. This is best done at a lab bench in order to see the results of the script.

Test Results:

Launching the script after removing the GPS device results in the message “GPS device not connected” and the exit of the script. This exit is reported to the driver program, vssm.py.

Test Analysis:

Exiting the script is the best result for a missing GPS device. Since the user is not expected to have the knowledge or desire to fix hardware issues the best course is to prevent the script from starting to alert a technician to the issue.

2. SixFab 4G Base Hat, Any Script Sending Data to Website

2.1 – Connection to the Website

The VSSM needs to be connected to the website in order to post location and DTC data as well as upload recording at user request.

Equipment Required:

Monitor and keyboard/mouse connected to Pi.

Input:

Terminal commands.

Output:

Percentage of packets lost and website homepage.

Pass Criteria:

In order to pass the Pi must have 0% packet loss using the 4G base hat to connect to the internet and the website displays.

Fail Criteria:

Greater than 0% packet loss or website not displayed.

Test Procedure:

After booting up the VSSM, make sure that you are not connected to any Wi-Fi networks by accessing the drop-down menu in the top right. Select the option to “Turn Off Wi-Fi”. Now, open a terminal and type the command “ping www.google.com”. Press Ctrl-C after enough data has been sent and view the percentage packet loss.

In order to test website connectivity, we cannot use the ping command. This is because the hosting provider blocks ICMP packets. Instead, use the command “xdg-open http://vssm.ddns.net”. This should cause the website homepage to be displayed in the default browser (Chromium).

Test Results:

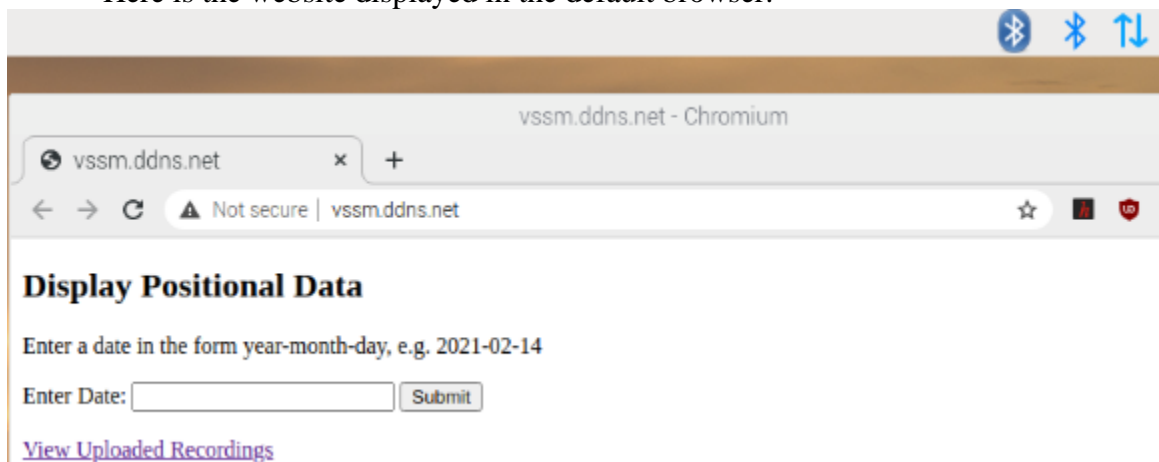
Here are the ping results when attempting to reach google.com.

```

pi@raspberrypi:~$ ping www.google.com
PING www.google.com (172.217.164.132) 56(84) bytes of data:
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=1 ttl=109 time=175 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=2 ttl=109 time=146 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=3 ttl=109 time=133 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=4 ttl=109 time=141 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=5 ttl=109 time=140 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=6 ttl=109 time=147 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=7 ttl=109 time=147 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=8 ttl=109 time=138 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=9 ttl=109 time=187 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=10 ttl=109 time=146 ms
64 bytes from iad30s24-in-f4.1e100.net (172.217.164.132): icmp_seq=11 ttl=109 time=145 ms
^C
--- www.google.com ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 23ms
rtt min/avg/max/mdev = 133.496/149.683/186.904/15.461 ms

```

Here is the website displayed in the default browser.



Test Analysis:

Internet connectivity was achieved on startup and the test is passed.

3. Pi Camera Module v2, Video recording script (Python), video upload script (Python)

3.1 – Recording Videos to Local Storage (Day Time)

The VSSM uses the Pi Camera v2 in order to record videos at the request of the user. To best test this, the standalone scripts (record.py and upload.py) can be run instead of the usual way of accessing them through the voice commands.

Equipment Required:

Monitor and keyboard/mouse to connect to the Pi.

Input:

Standalone scripts located in /home/pi/ directory. Terminal commands.

Output:

Video (normal ISO) saved to /home/pi/recordings/ directory.

Pass Criteria:

Upon running the script, a new video is saved to the correct directory with a timestamp burned into the video. The same timestamp should be used as the filename.

Fail Criteria:

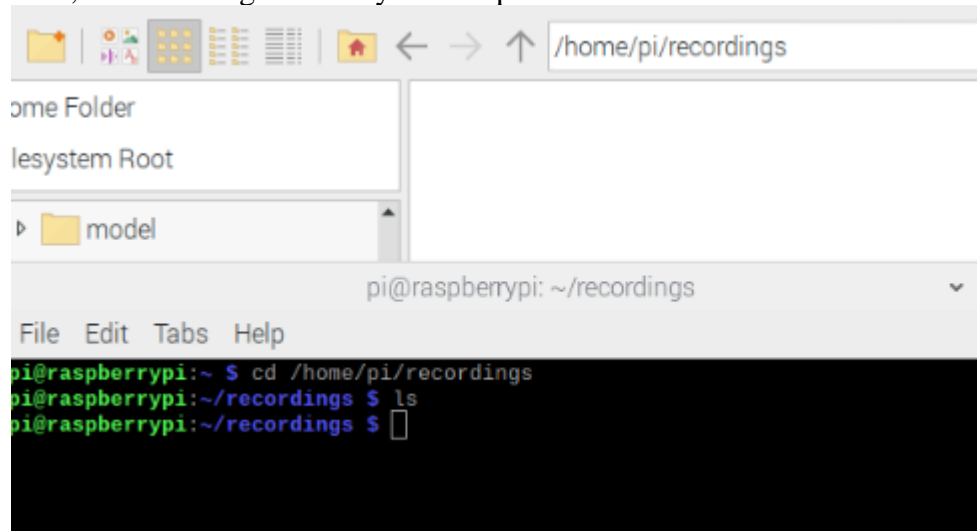
No video saved or saved to the wrong directory.

Test Procedure:

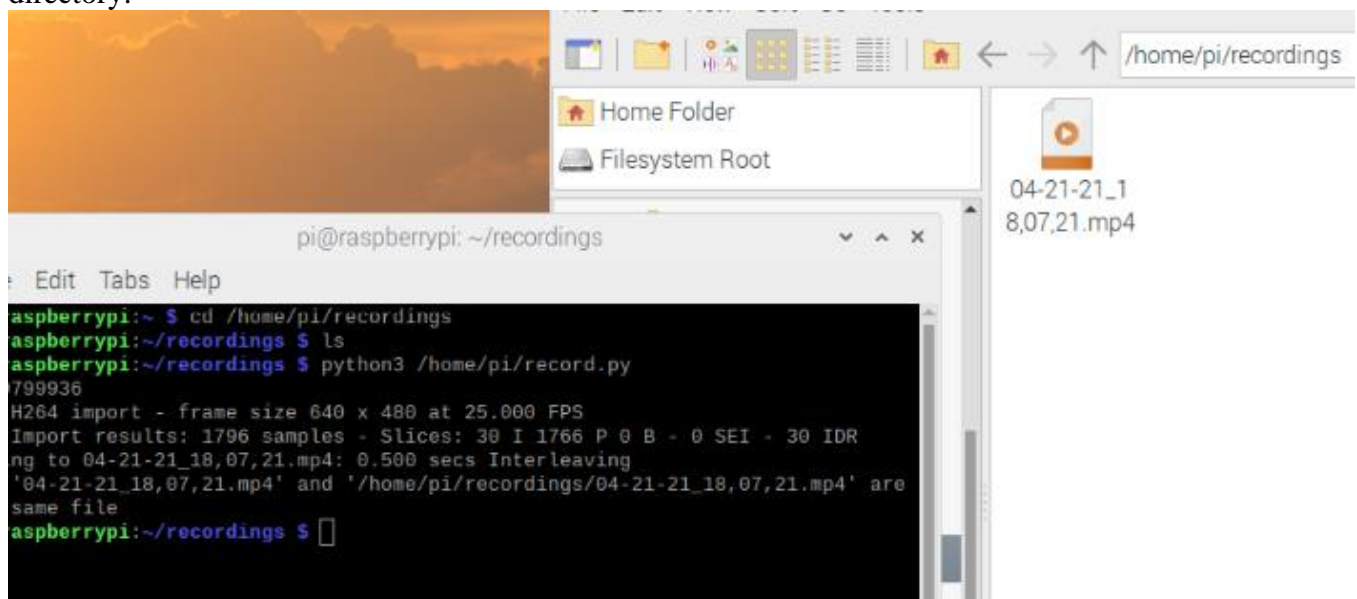
Note the current time and date. To avoid conflicts with 3.2 make sure the time is 07:00 – 20:00. Open a terminal and enter the command “python3 /home/pi/record.py”. After the script finishes, navigate to the /home/pi/recordings/ directory either in the terminal or using a file manager. Determine if the filename matches the time you entered the command. Launch the video and compare the burned stamp to the filename.

Test Results:

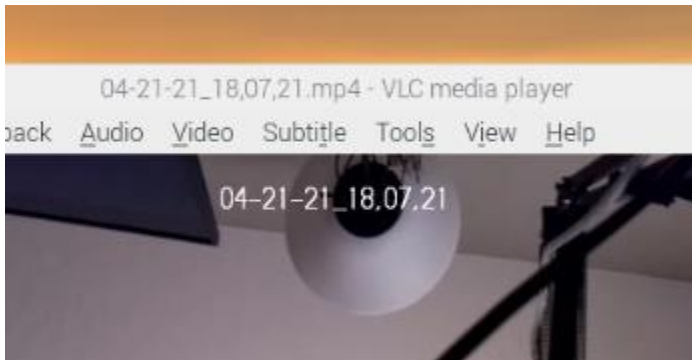
First, the recordings directory was emptied. The result is shown below.



After running the script, I saw the terminal output stating the available space, the conversion to mp4, and the removal of the original file. Here are these outputs as well as the resulting file in the directory.



Finally, here is screenshot of the video showing the timestamp.



Test Analysis:

The script recorded and saved the video correctly.

3.2 – Recording Videos to Local Storage (Night Time)

The VSSM uses the Pi Camera v2 in order to record videos at the request of the user. To best test this, the standalone scripts (record.py and upload.py) can be run instead of the usual way of accessing them through the voice commands.

Equipment Required:

Monitor and keyboard/mouse to connect to the Pi.

Input:

Standalone scripts located in /home/pi/ directory. Terminal commands.

Output:

Video (800 ISO) saved to /home/pi/recordings/ directory.

Pass Criteria:

Upon running the script, a new video is saved to the correct directory with a timestamp burned into the video. The same timestamp should be used as the filename. Subject of video should also be visible.

Fail Criteria:

No video saved or saved to the wrong directory or video too dark.

Test Procedure:

Note the current time and date. Make sure it is 20:00 – 06:00. Open a terminal and enter the command “python3 /home/pi/record.py”. After the script finishes, navigate to the /home/pi/recordings/ directory either in the terminal or using a file manager. Determine if the filename matches the time you entered the command. Launch the video and compare the burned stamp to the filename.

Test Results:

Results mirrored 3.1 except for the video. Here is a screenshot of the video with the boosted ISO.

04-21-21_23,53,20

Test Analysis:

As you can see, the test was a failure. Boosting the ISO to the max was not enough to film in darkness, although headlights or street lights may help somewhat. The only real solution would be increasing the quality of camera to have better night capability.

3.3 – Dealing with low available space

Continually saving videos to local storage will eventually eliminate free space on the disk. In order to continue operating without user assistance, the VSSM should remove the oldest videos whenever space is at a critical point and a recording is requested.

Equipment Required:

Monitor and keyboard/mouse connected to Pi. Also, ~5 GiB of test files copied to the Pi's disk.

Input:

Low available space and a recording request, made by standalone scripts located in /home/pi/ directory.

Output:

Additional space created by removing the oldest available files, until a certain amount of space is made.

Pass Criteria:

The record script operates even though there is low disk space. Continually running the record script will not push available space to critical levels since the oldest file would be removed.

Fail Criteria:

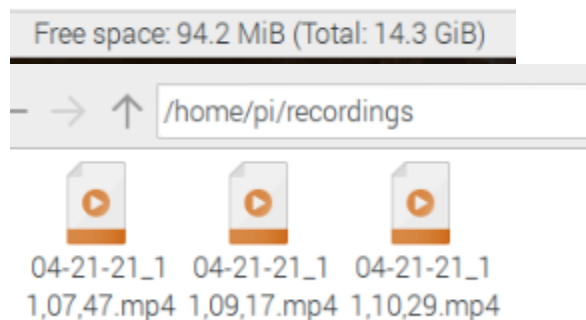
The recording is not created due to low disk space or anything but the oldest recording is removed.

Test Procedure:

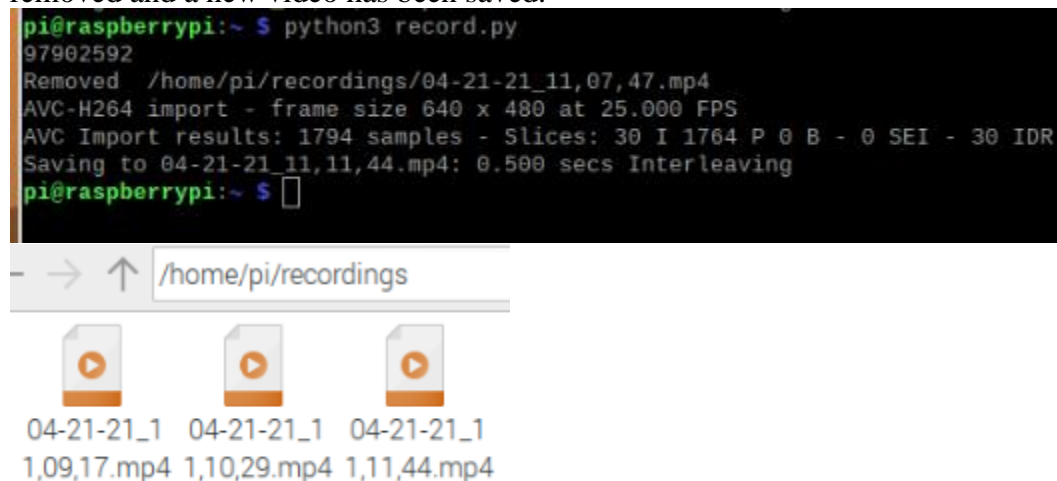
Copy over or generate enough data to fill the available disk space to around 100MiB. Take note of the oldest video saved in the /home/pi/recordings/ directory. Run the script using the command “python3 /home/pi/record.py” in a terminal. Continue to do this until you receive a message showing “Removed” followed by the filename. Compare this to the file you determined to be the oldest and verify that they match.

Test Results:

First, I copied over some DAT files from an unrelated program on another computer. These files, along with some calls to the record script, pushed the available space to under 100 MiB. Here is the state just before the trigger point. Note that the oldest video is labeled “04-21-21_11,07,47.mp4”.



Here is the result of running the script one more time. The video mentioned previously has been removed and a new video has been saved.



Test Analysis:

The record function was able to remove the oldest recording in order to save another video. This ensures that the user will never run into errors saving to a full disk and passes the test.

3.4 – Uploading Videos to Website

The VSSM should be able to upload the last recorded video at the request of the user. This should be tested after 2.1.

Equipment Required:

Monitor and keyboard/mouse connected to Pi.

Input:

Script located at /home/pi/upload.py.

Output:

Terminal output of script, link on website.

Pass Criteria:

Script outputs “File Uploaded” to terminal. Video is displayed on website located at <http://vssm.ddns.net/download.php>.

Fail Criteria:

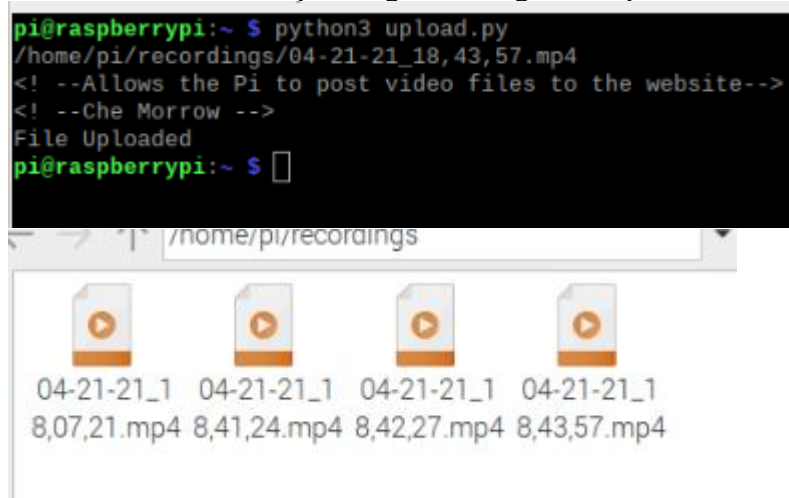
Script fails and does not display the correct result. No link is available on website.

Test Procedure:

Make sure there are videos in the directory to be uploaded. Take note of the newest video saved to the directory. Run the upload script using the command “python3 /home/pi/upload.py”. Verify that the terminal output matches the newest recording and that it displays “File Uploaded”. Next, navigate to the website and select the link that states “View Uploaded Recordings”. Verify the newest entry (top of the list) matches the correct video.

Test Results:

Here is the script being run along with a picture of the available files.



Here is the website after the script was run.

List of Available Recordings

[04-21-21_18,43,57.mp4](#)

[04-11-21_18,28,30.mp4](#)

[04-11-21_18,21,55.mp4](#)

[03-22-21_11,39,08.mp4](#)

Test Analysis:

The correct video was posted to the website and made available for download.

3.5 – Hardware Error Detection

The recording functionality of the VSSM relies on the Pi Camera Module v2. If this device is not connected, the record script should not run.

CAUTION: <Removing the camera module requires the removal of the power supply and 4G hats. This has the possibility of damaging the header pins and should be handled cautiously.>

Equipment Required:

Monitor and keyboard/mouse connected to Pi.

Input:

Removal of the camera module.

Output:

Script fails and exits.

Pass Criteria:

Script fails and does not continue.

Fail Criteria:

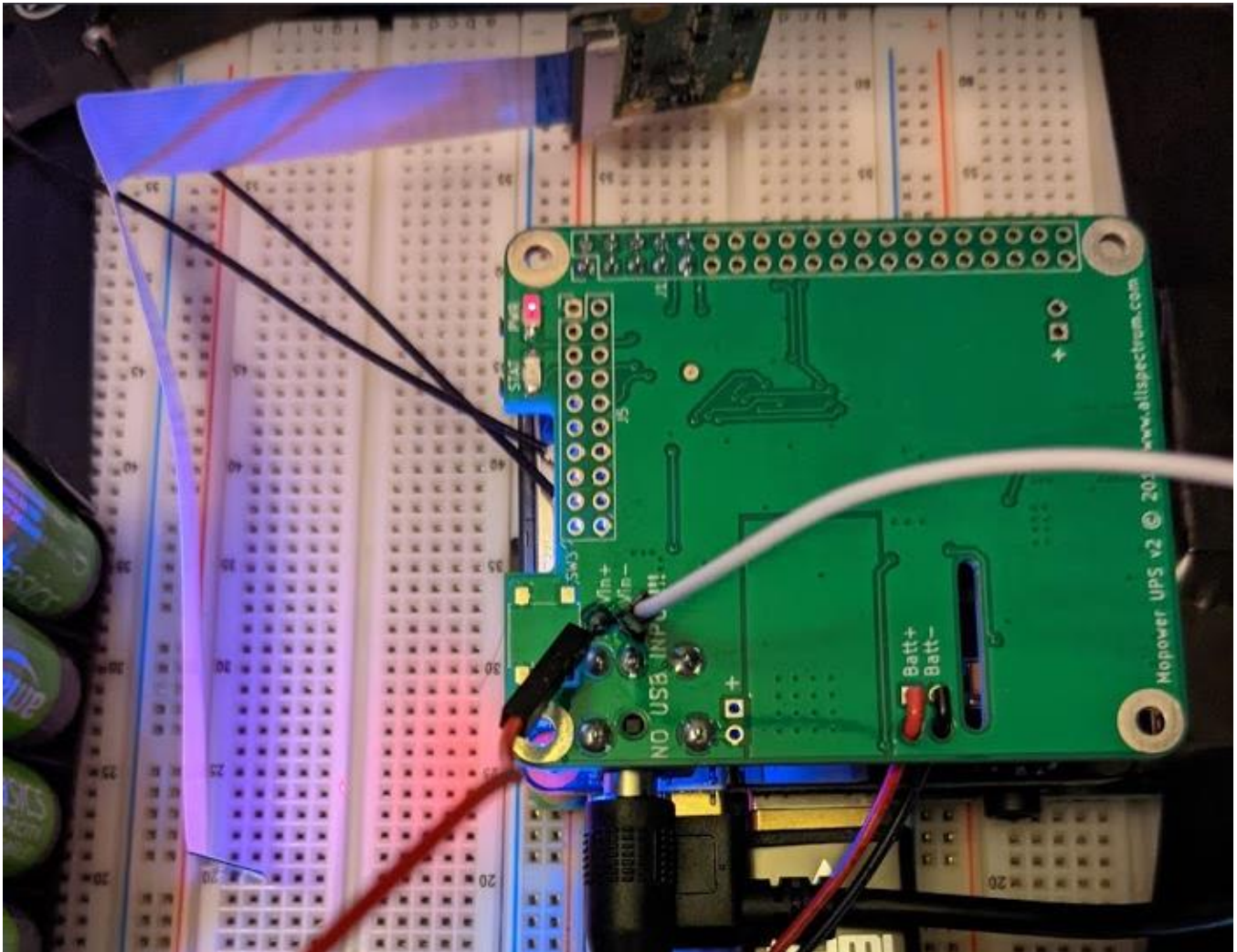
Script ignores missing hardware and continues “quietly”.

Test Procedure:

Test 3.1 and 3.2 should provide proof that the record script functions normally before this test is attempted. First remove the power supply board from the header pins. Then, disconnect the 4G base hat from the USB port and also remove from the header pins. Finally, locate the port that the camera module is connected to. Pull up on the top piece of the port and withdraw the ribbon cable. Next, reverse these steps to attach the 4G base hat and the power supply board. Power up the Pi and attempt to repeat test 3.1 or 3.2.

Test Results:

Attempting to run the script without the camera connected results in an error that stops the script. Here is the Pi without the camera.



The record script fails without the camera connected.

Test Analysis:

The script exited due to connection errors, which was the desired outcome.

4. ELM327 OBD-II Adapter, OBD Data Acquisition Script (Python)

4.1 – Testing Connection to Vehicle

The VSSM should be able to connect to a vehicle in order to communicate. This test should take place after 4.3. If the test vehicle has a check engine light 4.1 and 4.2 should take place at the same time.

Equipment Required:

A vehicle with an OBD-II port.

Input:

Connection to the test vehicle.

Output:

RPM at engine start displayed on website.

Pass Criteria:

Data is retrieved from vehicle is POSTed to the website.

Fail Criteria:

No data is retrieved from vehicle or POSTed to the website.

Test Procedure:

Connect the VSSM to the test vehicle. Turn on the engine and wait for the website to update.

Test Results:



Diagnostic Trouble Codes Detected:

RPM at start: 954.5 revolutions_per_minute

Test Analysis:

This data was retrieved from the connection to the vehicle. There is unfortunately no way to compare the results other than to eyeball the RPM gauge in the dashboard.

4.2 – Reading Codes

The VSSM should be able to let users know what triggered a DTC in the event of a check engine light. This test should take place after 4.3.

Equipment Required:

A vehicle with a check engine light displayed.

Input:

A connection to the test vehicle.

Output:

DTC is POSTed to the website.

Pass Criteria:

The VSSM should detect a DTC pending in the vehicle.

Fail Criteria:

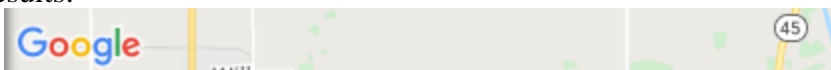
The VSSM cannot detect any codes in the vehicle.

Test Procedure:

Connect the VSSM to the test vehicle. Turn on the engine and wait for the website to update.

Shut off the engine and then disconnect the Pi from the vehicle.

Test Results:



Diagnostic Trouble Codes Detected:

RPM at start: 954.5 revolutions_per_minute

P0430, Catalyst System Efficiency Below Threshold

[Return To Homepage](#)

Test Analysis:

The VSSM was able to detect this code from the vehicle.

4.3 – Uploading OBD-II Data to Website

Data obtained from the vehicle should be POSTed to the website and placed below the Maps API. This test should be completed before 4.1 and 4.2 to facilitate testing while in a vehicle.

Equipment Required:

Monitor and keyboard/mouse.

Input:

Small change to OBD_Upload.py.

Output:

Data displayed on website.

Pass Criteria:

Website displays a series of test codes.

Fail Criteria:

Data does not POST correctly to website.

Test Procedure:

Open up the file OBD_Upload.py. Add the following lines before the if(running) on line 40.

```
day = date.today()
```

```
payload = { "day" : day, "code" : "Test Codes Follow, Disregard" }
```

Test Results:

Here is the website after my example POST using the request library.



Diagnostic Trouble Codes Detected:

Test Codes Follow, Disregard

Test Analysis:

This command show we are able to successfully send data to the website to be stored in the OBD database.

5. Local Windows Server, Website (HTML, PHP, Javascript)

5.1 – Exception Handling for POST Requests

Website should not store incorrect data to the website. In this example we will attempt to post random data to the GPS upload link.

Equipment Required:

Monitor and keyboard/mouse.

Input:

Python example script “bad_upload.py”.

Output:

Nothing should change. Current location data should display correctly.

Pass Criteria:

Map should continue to load positional data, database should not contain erroneous data.

Fail Criteria:

Maps API fails to load or data stored in database.

Test Procedure:

Run the script using the command “python3 bad_upload.py”.

Test Results:

The script completed but the website loaded the map without any issues. Here is the map, zoomed out so we can see no pins placed in odd locations.



Test Analysis:

The website and the database configuration mainly provide defense against accidental bad posts. Protecting the databases against outside manipulation is not in the scope of this project.

6. MoPower UPS, UPS Firmware

6.1 – Startup and Shutdown Voltage Ranges

Since the VSSM should operate without user assistance, it requires the ability to start and shut down with the vehicle. Although the OBD-II connector is always connected to the battery, the voltage varies with the running of the engine. The power supply board is set to measure these changes and respond accordingly.

Equipment Required:

Lab bench power supply with variable voltage output.

CAUTION: <While the current to the power supply board is limited by a polyamp fuse, the tester should not exceed 15 V input to the device. Also, make sure to follow the documentation of your chosen bench power supply.>

Input:

Specific voltages over given time frames.

Output:

Power supply is set to turn on at 13.1 V for 3 seconds and set to turn off at 11.5 V for 10 seconds.

Pass Criteria:

At the specified voltages and after the given time frame the VSSM starts or shuts down.

Fail Criteria:

The VSSM does not obey the ranges set in the power supply.

Test Procedure:

Connect the VSSM to your power supply and set the voltage to 13.1 V. Wait 3 seconds and document the response. Next, lower the voltage below 11.5 V and wait 10 seconds for the response.

Test Results:

The VSSM started at 13.1 V and then shut off at 11.5 V for 10 seconds.

Test Analysis:

The VSSM passed the test and will turn off once the battery reaches this level. Using data gathered from the car this should happen within 10 minutes of the vehicle's engine turning off.

6.2 – Status Light Messages

The power supply board has a status LED that is used to give a small amount of feedback to the user. It blinks once when the main program starts, blinks rapidly while recording a video, and blinks an SOS when attempting to contact emergency services.

Equipment Required:

Not Applicable.

Input:

Power to the VSSM through bench supply or OBD-II port.

Output:

Status LED should blink once when the VSSM is started.

Pass Criteria:

Status LED winks.

Fail Criteria:

No output to the status LED.

Test Procedure:

This will occur automatically on startup. Simply provide power to the device and wait a few seconds for boot.

Test Results:

LED blinked as expected once the device powered on.

Test Analysis:

Test was successful. Take note that this test will also occur naturally during 3.1, 3.2, and 7.1.

7. Quectel EC25 Mini PCIe 4G/LTE Module, AT Command Script

7.1 – Sending SMS Message

Description of requirement.

Equipment Required:

Monitor and keyboard/mouse connected to the Pi. Device proven to accept SMS messages.

Input:

Script to call contact function based on voice command, voice.py.

Output:

Status light and SMS message.

Pass Criteria:

SMS message received at destination and status light should blink SOS.

Fail Criteria:

SMS message not received at destination or status light not changed.

Test Procedure:

This should take place after 8.1 and 9.1 are tested and passed. Launch the main script (vssm.py) and wait for the voice subprocess to be launched. Speak the command “vehicle emergency contact”. Wait for verification of the SMS message by verifying the status light of the power supply board and the receipt of the message on the test device.

Test Results:

The status light blinks an SOS but message never received.

Test Analysis:

Since the message was never received this test was a failure. This is likely due to a limitation of the device’s phone plan but will require more research.

8. Bluetooth Lavalier Microphone, Bluetooth Configuration

8.1 – Connection to Microphone

Connection to the Bluetooth microphone is essential to process user commands. The VSSM should automatically connect to the microphone and set it as the default input device due to OS configuration changes.

Equipment Required:

Monitor and keyboard/mouse connected to Pi. Bluetooth lavalier included with VSSM.

Input:

Power button on microphone and BlueZ dropdown menu.

Output:

Device status updates without additional inputs.

Pass Criteria:

Pi automatically connects to microphone after it is powered on.

Fail Criteria:

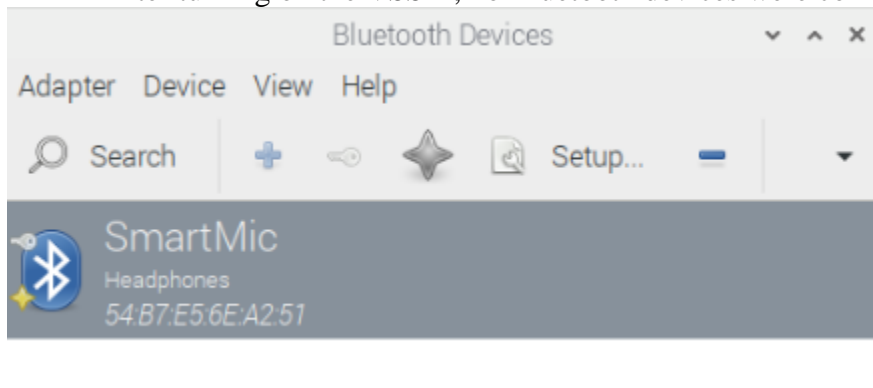
User needs to give some input in order to pair the device.

Test Procedure:

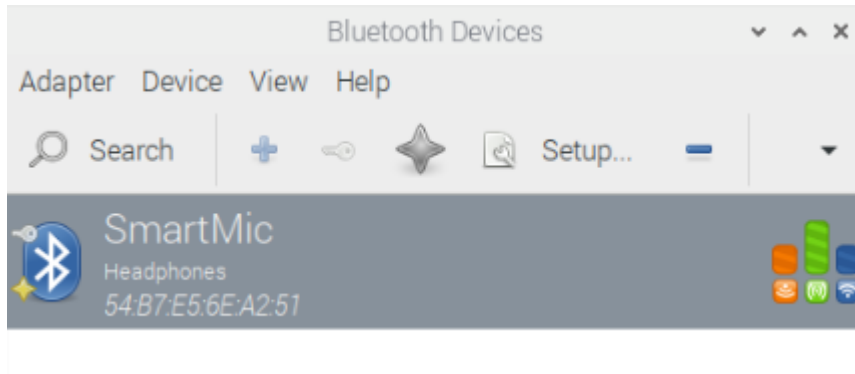
Power on the VSSM with the Bluetooth microphone off. Take note of the status of connected devices by selecting “Devices” from the Bluetooth menu at the top right. Now, power on the microphone. Note the new status of the device in the menu.

Test Results:

After turning on the VSSM, no Bluetooth devices were connected. Here is the starting menu.



Next, I powered on the microphone and verified the device status in the menu.



Test Analysis:

The microphone connected without additional input, passing the test.

9. Vosk API (Python)

9.1 – Voice Commands Call Needed Functions

When the user gives a command, the VSSM should execute the related script. To best test this, we can modify the main program to execute only the voice script and verify the output. Must take place after tests 3.1, 3.2, 3.4, and 7.1.

Equipment Required:

Monitor and keyboard/mouse connected to Pi.

Input:

Changes to main program and execution of main script.

Output:

Terminal output as well as results of tests 3.1, 3.2, 3.4, and 7.1.

Pass Criteria:

Command “vehicle begin recording” mirrors test results of 3.1 or 3.2.

Command “vehicle transmit video” mirrors test results of 3.4.

Command “vehicle emergency contact” mirrors test results of 7.1.

Fail Criteria:

Any command does not have the same outcome as the related test.

Test Procedure:

First thing is to comment out lines 155 and 156 of the vssm.py script. This ensures a clean output to verify our tests. Next, launch the script using the command “python3 /home/pi/vssm.py”. Speak the phrases one at a time and verify the output after each instruction.

Test Results:

The voice commands were given in the order listed in the pass criteria. Here are the output results of each command.


```

pi@raspberrypi:~ $ python3 vssm.py
VSSM PID is: 1105
MORSE=E
MORSE=
Voice PID not found
Voice PID is: 1136
Microphone connected
Voice Running
LOG (VoskAPI:ReadDataFiles():model.cc:194) Decoding params beam=10 max-acti
LOG (VoskAPI:ReadDataFiles():model.cc:197) Silence phones 1:2:3:4:5:6:7:8:9
LOG (VoskAPI:RemoveOrphanNodes():nnet-nnet.cc:948) Removed 0 orphan nodes.
LOG (VoskAPI:RemoveOrphanComponents():nnet-nnet.cc:847) Removing 0 orphan c
LOG (VoskAPI:CompileLooped():nnet-compile-looped.cc:345) Spent 0.266516 sec
LOG (VoskAPI:ReadDataFiles():model.cc:221) Loading i-vector extractor from
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:183) Computing deriv
actor
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:204) Done.
LOG (VoskAPI:ReadDataFiles():model.cc:251) Loading HCL and G from model/gr
t
LOG (VoskAPI:ReadDataFiles():model.cc:273) Loading winfo model/graph/phones
Recording Video
MORSE=E
5150871552
AVC-H264 import - frame size 640 x 480 at 25.000 FPS
AVC Import results: 144 samples - Slices: 3 I 141 P 0 B - 0 SEI - 3 IDR
Saving to 04-21-21_20,08,18.mp4: 0.500 secs Interleaving
MORSE=
Uploading Video
<! --Allows the Pi to post video files to the website-->
<! --Che Morrow -->
File Uploaded
Contacting EMS
MORSE=SOS
MORSE=

```

Test Analysis:

Each command resulted in the pass conditions of their related test, except for 7.1 which failed in the same manner.

9.2 – Voice Range Testing

The adaptability of the voice recognition software is outside the scope of this project. However, a small test can be done to at least show a cursory effort to make the device universal.

Equipment Required:

Monitor and keyboard/mouse. Bluetooth microphone included with VSSM.

Input:

A voice dissimilar to your own.

Output:

Terminal output of script.

Pass Criteria:

Output must match 9.1.

Fail Criteria:

Output does not match 9.1.

Test Procedure:

Repeat the process used in 9.1, except use your assistant to issue the needed commands.

Test Results:

I used Kayla Mitchell to issue the commands in my place. Here are the results.

```
Microphone connected
Voice Running
LOG (VoskAPI:ReadDataFiles():model.cc:194) Decoding params beam=10 max-a
LOG (VoskAPI:ReadDataFiles():model.cc:197) Silence phones 1:2:3:4:5:6:7:4
LOG (VoskAPI:RemoveOrphanNodes():nnet-nnet.cc:948) Removed 0 orphan nodes
LOG (VoskAPI:RemoveOrphanComponents():nnet-nnet.cc:847) Removing 0 orphan
LOG (VoskAPI:CompileLooped():nnet-compile-looped.cc:345) Spent 0.26106 s
LOG (VoskAPI:ReadDataFiles():model.cc:221) Loading i-vector extractor fr
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:183) Computing de
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:204) Done.
LOG (VoskAPI:ReadDataFiles():model.cc:251) Loading HCL and G from model/
LOG (VoskAPI:ReadDataFiles():model.cc:273) Loading winfo model/graph/pho
Uploading Video
<! --Allows the Pi to post video files to the website-->
<! --Che Morrow -->
File Uploaded
Recording Video
MORSE=E
5142474752
AVC-H264 import - frame size 640 x 480 at 25.000 FPS
AVC Import results: 144 samples - Slices: 3 I 141 P 0 B - 0 SEI - 3 IDR
Saving to 04-21-21_22,04,19.mp4: 0.500 secs Interleaving
MORSE=
Contacting EMS
MORSE=SOS
MORSE=
```

Test Analysis:

These results mirror the outcome from 9.1, although she did them in a different order.

10. Raspberry Pi 3 B+, “Main” Program (Python)

10.1 – Program Begins on Startup

This test is nearly the same as 6.2, however our pass criteria will be different. When the VSSM is started, it needs to launch the main program “vssm.py”.

Equipment Required:

Monitor and keyboard/mouse connected to Pi.

Input:

Terminal command “ps aux”.

Output:

Terminal output.

Pass Criteria:

Four programs should be running on startup; vssm.py, voice.py, GPS_Upload.py, and OBD_Upload.py.

Fail Criteria:

These programs are not running or the VSSM does not start.

Test Procedure:

Start the VSSM normally using the bench power supply. Open a terminal and enter the command “ps aux”.

Test Results:

All processes were started successfully with the VSSM.

Test Analysis:

The main program is set to start at boot using the SYSTEMD method, launch the needed programs and passing our tests. This means the process will also automatically restart if any unforeseen exceptions occur.

10.2 – Exception Handling

Although an effort was made to prevent exceptions from occurring in the scripts, there are some errors we cannot protect against, e.g. hardware disconnection. Therefore, programs may need to be restarted if they exit unexpectedly. In addition, we can have the Pi restart completely when programs have failed a specified amount of time, which may solve unexpected hardware issues.

Equipment Required:

Monitor and keyboard/mouse connected to Pi.

Input:

Running main script with lines 155 and 156 commented out, microphone disconnected.

Output:

Terminal output.

Pass Criteria:

Main program continually restarts failed process.

Fail Criteria:

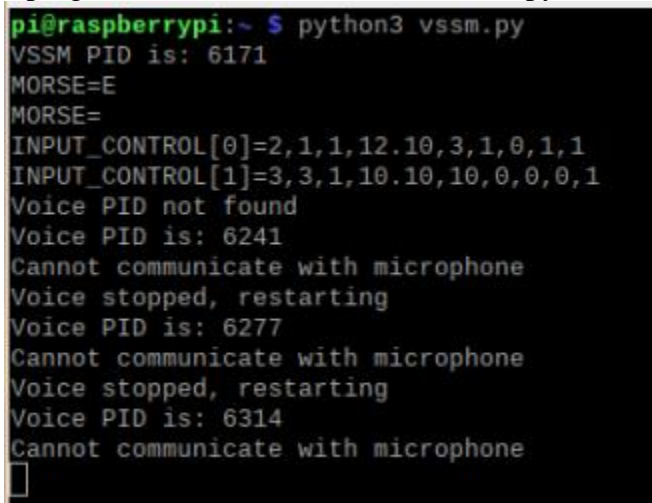
Main program ignores failed process.

Test Procedure:

Comment out lines 155 and 156 in the main program “vssm.py” for simplicity. Then, run the script using the command “python3 vssm.py”. Take note of the output.

Test Results:

The voice program exits when a microphone is not detected. As you can see from these results, the main program detects this and restarts voice.py.



```
pi@raspberrypi:~ $ python3 vssm.py
VSSM PID is: 6171
MORSE=E
MORSE=
INPUT_CONTROL[0]=2,1,1,12.10,3,1,0,1,1
INPUT_CONTROL[1]=3,3,1,10.10,10,0,0,0,1
Voice PID not found
Voice PID is: 6241
Cannot communicate with microphone
Voice stopped, restarting
Voice PID is: 6277
Cannot communicate with microphone
Voice stopped, restarting
Voice PID is: 6314
Cannot communicate with microphone

```

After trying ten times, the program gives up and sends the reboot command to the Pi.

Test Analysis:

This operates exactly as we expected, and handles exits from all of the scripts using signal().

III. OTHER TESTING

INSTALLATION REQUIREMENTS/ POWER SUPPLY REQUIREMENTS

The VSSM has been tested in three different vehicles, with three different makes and models. The device is expected to perform in any vehicle using the OBD-II standard. The user must connect the OBD-II adapter to their vehicle and the device should perform as expected. If not, make sure a qualified technician has completed the tests detailed above.