# Project Design Document (PDD)

## Vehicle Safety and Security Monitor (VSSM)

**Author: Che Morrow**
**Date: 4/22/2021**

## 1.0 Architectural Design

### **1.0.**1**.**

- Determine vehicle position
- Provide mobile internet connection
- Record security videos
- Obtain vehicle diagnostic trouble codes
- Deliver data to users
- Device must be powered by vehicle
- Contact with Emergency Services
- Retrieve user voice data
- Process user voice data
- Startup processes and exception handling

### **1.0.**2.

| Customer Need | Functional Element | Physical Element | “Software” Element |
|---|---|---|---|
| User can find location of vehicle. | Determining Vehicle Position | VK-162 GPS Dongle | Location Processing/Upload Script (Python) |
| Users can find vehicle location and obtain DTCs through the website | Internet Connection | SixFab 4G Base Hat | Any Script Sending Data to Website |
| 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) |
| User can read DTCs using | Car Diagnostics | ELM327 OBD-II Adapter | OBD Data Acquisition Script |

| | | | |
|---|---|---|---|
| vehicle's OBD-II port. | | | (Python) |
| Users can find vehicle location and obtain DTCs through the website | Delivering Data to Users | Local Windows Server | Website (HTML, PHP, Javascript) |
| Day to day use does not require special knowledge. Work s with vehicle OBD-II port, power. | Powering Device with Vehicle | MoPower UPS | UPS Firmware |
| Users can give command to contact EMS. | Contact with Emergency Services | Quectel EC25 Mini PCIe 4G/LTE Module | AT Command Script |
| The user interacts through voice commands | Retrieving User Voice Data | Bluetooth Lavalier Microphone | Bluetooth Configuration |
| The user interacts through voice commands | Processing User Voice Data | | Vosk API (Python) |
| Day to day use does not require special knowledge. | Startup Processes and Exception Handling | Raspberry Pi 3 B+ | "Main" Program (Python) |

**2.0 Hardware Design**

VK-162 GPS Dongle [1][8]:

2.0.1.1. Receives wireless signals from at least 4 GPS satellites.

2.0.1.2. Solves navigational equations using times and positions of satellites to determine current location. Makes data available as latitude and longitude.

2.0.1.3. USB (serial) connection to Raspberry Pi 3B+. Latitude ranges from -90 to 90, longitude ranges from -180 to 180, and time ranges from 00:00:00 to 23:59:59.

SixFab 4G Base Hat [2][8]:

2.0.2.1. Mini PCIe connection to Quectel EC25 Mini PCIe 4G/LTE Module. USB (serial) connection to the Raspberry Pi 3B+.

2.0.2.2. Bridges the connection to the Quectel EC25 Modem, providing a simple connection to the 4G/LTE cellular network.

2.0.2.3. Mini PCIe connection to Quectel EC25 Mini PCIe 4G/LTE Module. USB (Serial) connection to the Raspberry Pi 3B+.

Pi Camera Module v2 [3][8]:

2.0.3.1. Sony IMX219 8-megapixel sensor. Connects to Camera Serial Interface (CSI) on Raspberry Pi 3B+ over ribbon cable.

2.0.3.2. Converts light coming into sensor into a data stream that can be processed and saved as a video.

2.0.3.3. Connects to Camera Serial Interface (CSI) on Raspberry Pi 3B+ over ribbon cable.

ELM327 OBD-II Adapter [4][8]:

2.0.4.1. Data Link Connector (DLC) located in vehicles manufactured after 1996. USB (serial) connection to Raspberry Pi 3 B+.

2.0.4.2. Converts SAE J1850 PWM, SAE J1850 VPW, ISO 9141-2, ISO 14230 KWP2000, or ISO 15765 CAN to serial over USB signal for data processing on the Raspberry Pi 3 B+. The exact communication protocol differs between vehicle makes and models.

2.0.4.3. USB (serial) connection to Raspberry Pi 3 B+. Data Link Connector (DLC) located in vehicles manufactured after 1996.

MoPower UPS [5][8]:

2.0.6.1. UART connection to Raspberry Pi 3 B+ over header pins. Vin connected to pin 16 of vehicle DLC. Voltage input ranges from 11-30 VDC. Input current limited by a 5 A automotive (ATM) fuse.

2.0.6.2. Measures input voltage of DLC and sends power to the Raspberry Pi 3 B+ when voltage reaches 13.1 V for 2 seconds. Sends shutdown instruction to Pi when voltage drops to 11.5 V for at least 10 seconds. Also, receives firmware instructions over UART connection.

2.0.6.3. UART connection to Raspberry Pi 3 B+ over header pins. Output voltage limited to 5 VDC. Output current limited by a 2.5 A polyfused.

Quectel EC25 Mini PCIe 4G/LTE Module [6][9]:

2.0.7.1. Mini PCIe connection to SixFab 4G Base Hat. LTE Main & Diversity u.FL Antenna 100 mm.

2.0.7.2. Provides connection to cellular towers within a network using LTE antenna.

Communicates with the 4G Base Hat in order to simplify data usage of the Raspberry Pi 3 B+. Also accepts AT commands from Pi/Base Hat at baudrate 115200.

2.0.7.3. Mini PCIe connection to SixFab 4G Base Hat. LTE Main & Diversity u.FL Antenna 100 mm.

Bluetooth Lavalier Microphone [7][8]:

2.0.8.1. 6mm Electret Condenser Microphone (ECM).

2.0.8.2. Converts audio input from ECM to 48Khz-16 bit Advanced Audio Coding (AAC).

2.0.8.3. Bluetooth 4.2 connection to Raspberry Pi 3 B+.

**2.1 Theory of operation**

The VSSM is housed near the vehicle's DLC in order to constantly maintain a connection. While the car is not running the UPS prevents battery drain by keeping the device from booting up. When the UPS detects a large voltage increase (13.1 V for 2 seconds), it is assumed to be the vehicle starting. This causes the UPS to send a startup signal to the Raspberry Pi 3 B+. Upon starting, a process is set to trigger the execution of a main driver program that manages our subprocesses. Firmware and basic drivers control most of the hardware initialization, including the data connection through the SixFab 4G Base Hat and Quectel EC25. This driver program initializes three subprocesses:

Voice: This process constantly parses audio detected by the Bluetooth microphone looking for spoken key phrases. These phrases allow the user to trigger video recordings, upload the latest recording to the website, and contact emergency services.

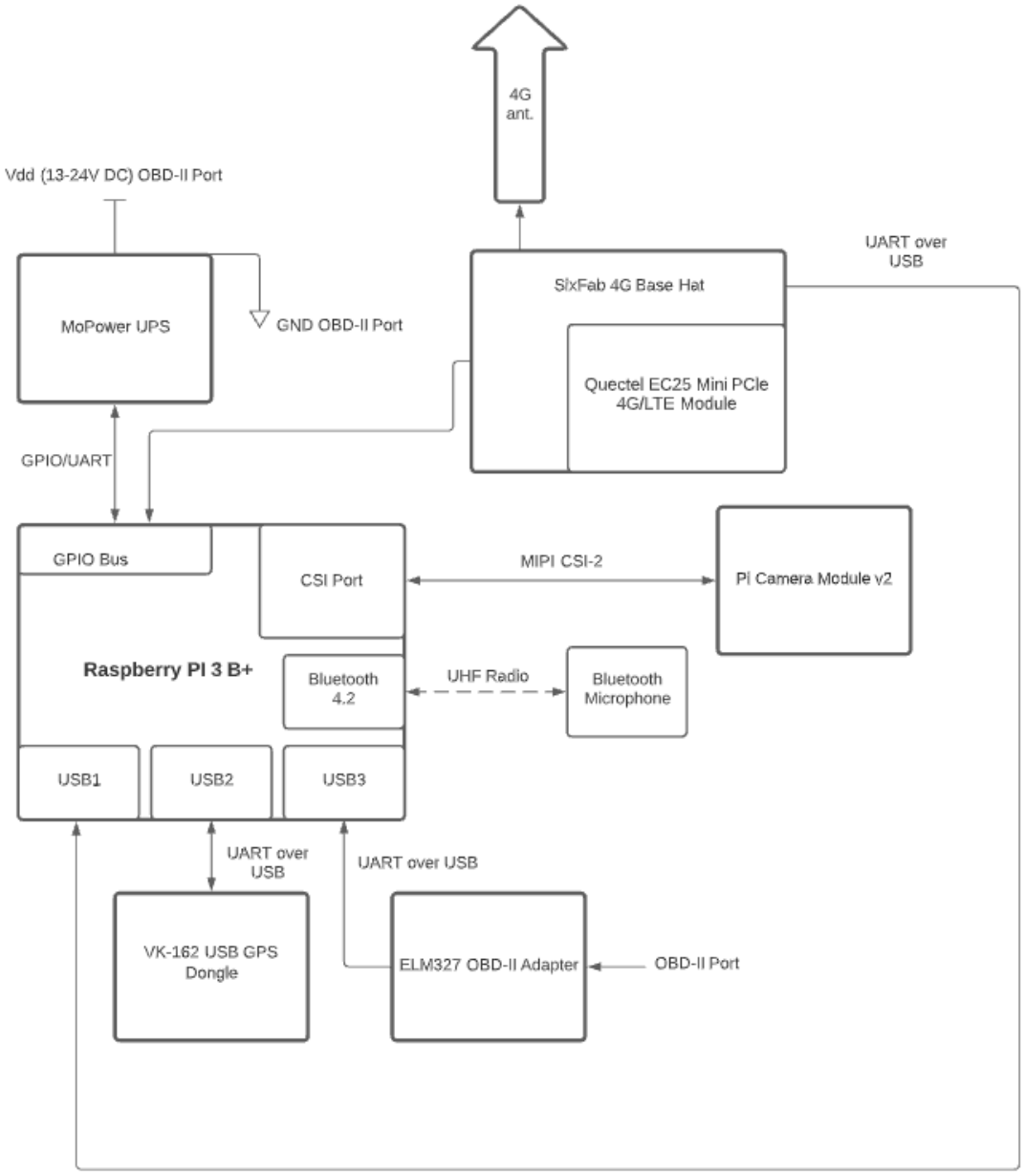
GPS: This process obtains the current location of the vehicle through the GPS dongle and then converts and uploads this data to the website and the related database. The website then uses the JavaScript Maps API to plot these points so users can follow the travel of the vehicle. This happens consistently while the vehicle is in operation.

OBD: This process attempts to pulls DTCs from the vehicle's OBD-II port using most supported communication methods. Since a fully functioning vehicle may not have any codes stored the connection is proven by requesting and receiving a non-zero RPM measurement through this connection. Once this data is obtained, it is sent to the website and added to the related database.

Website: The website has two options on the landing page. The first is a date entry that will allow users to view that day's travel locations as well as any DTCs triggered while driving. Second is a page listing all available videos that the user may download, sorted newest to oldest.

Once the user has arrived at their destination and turned the engine off, the VSSM will continue to operate until the battery drops to a specific level (11.5 V for 10 seconds). Once this occurs, the UPS sends a shutdown signal to the Pi.

## 2.2 Schematics




## 3.0 Software Design

Determining Vehicle Position [13][14][15][16]:

3.0.1.1. Serial data retrieved from the GPS module using the GPSD library.

3.0.1.2. Creates a gpsd object with reporting mode enabled. Checks class of gpsd.next(), should be TPV, if it is we can use getattr() to get latitude, longitude, and a time stamp. Before we send this data to the website the time stamp is converted into PST and then separated into time and day. All of this data is then stored into an array and returned to the while loop for

POSTing.

3.0.1.3. Latitude, longitude, time, and day are POSTed to the website using the Requests library.

Uploading videos to website [18]:

3.0.2.1. Video file stored in the /home/pi/recordings/ directory.

3.0.2.2. Sorts the recordings directory to file the newest file. Then, open()s the file in binary read mode and then POSTs the request. Also, prints the return of the request for error detection.

3.0.2.3. POST request containing video file.

Security Video Recording [12][20]:

3.0.3.1. Data stream from Pi Camera Module v2.

3.0.3.2. First step is to make sure the disk has space to record a video. We can retrieve the available space using shutil.diskusage() on the path. If the space is too low, find the oldest video using sort() and remove it. Continue this until there is space to record. To record a video, we create a PiCamera object and set the desired resolution, framerate, and burn the time stamp into the video. PiCamera() saves the video as an h264 file type, so we use MP4box to convert the file to mp4.

3.0.3.3. An mp4 video saved in the recording directory.

Obtaining Vehicle DTCs [17][18]:

3.0.4.1. Serial data retrieved from OBD-II adapter.

3.0.4.2. A connection to the car is formed using OBD() without any passed setting. This sets the connection to auto-retrieve the baud rate and port that the car uses. To make sure the connection is formed, we send a query containing the RPM command. If this returns a value, we know the engine is running and we're connected. Next, we query the vehicle periodically for any DTCs pending. If any are found, we POST them to the website.

3.0.4.3. POST request containing DTCs.

Displaying Data to Users [18][19]:

3.0.6.1. HTTP GET requests from users. Users also POST a date to search for.

3.0.6.2. Data from the databases is retrieved using statement handles executed as if they were a mySQL command entered directly into the console. Something similar is done with the Maps API, but first that data is converted into an XML file since the API requires it. The file is formatted with a series of fwrites() to mirror the required structure into a file. This file is then sent to the Maps API, which is based off of the example given in the documentation, with

some small changed to the layout of the map and the data given to the pins.

3.0.6.3. Data stored in the mySQL locational database converted to pins and placed on Google Map. Data stored in OBD database displayed as a list. Videos available for download are also listed as links.

UPS Firmware Modifications [5]:

3.0.7.1. None.

3.0.7.2. We can send commands to the UPS by calling a command to the terminal. For example, to flash an SOS on the status LED we simply call "/home/pi/mopower/mpcmd MORSE=SOS". These instructions can be found on the MoPower website.

3.0.7.3. Terminal command to MoPower Python wrapper.

Processing Voice Input [10][11]:

3.0.8.1. Audio input from Bluetooth microphone.

3.0.8.2. First, we retrieve a list of sound devices from the Pi. Testing has shown that our Bluetooth microphone is always device 11. Next, we run our model through the Vosk API. This model is small and completely offline, limiting the resources needed to detect speech. We want to create an input stream using the default sample rate and our device information. Using this stream and our model we can then proceed to build a "recognizer" using Kaldi and Vosk. Now, we take audio from the queue, run it through the recognizer, and if the waveform is acceptable, we can determine the words spoken. Finally, we can compare these words to set strings and call the functions accordingly.

3.0.8.3. Calls functions depending on detected key phrases.

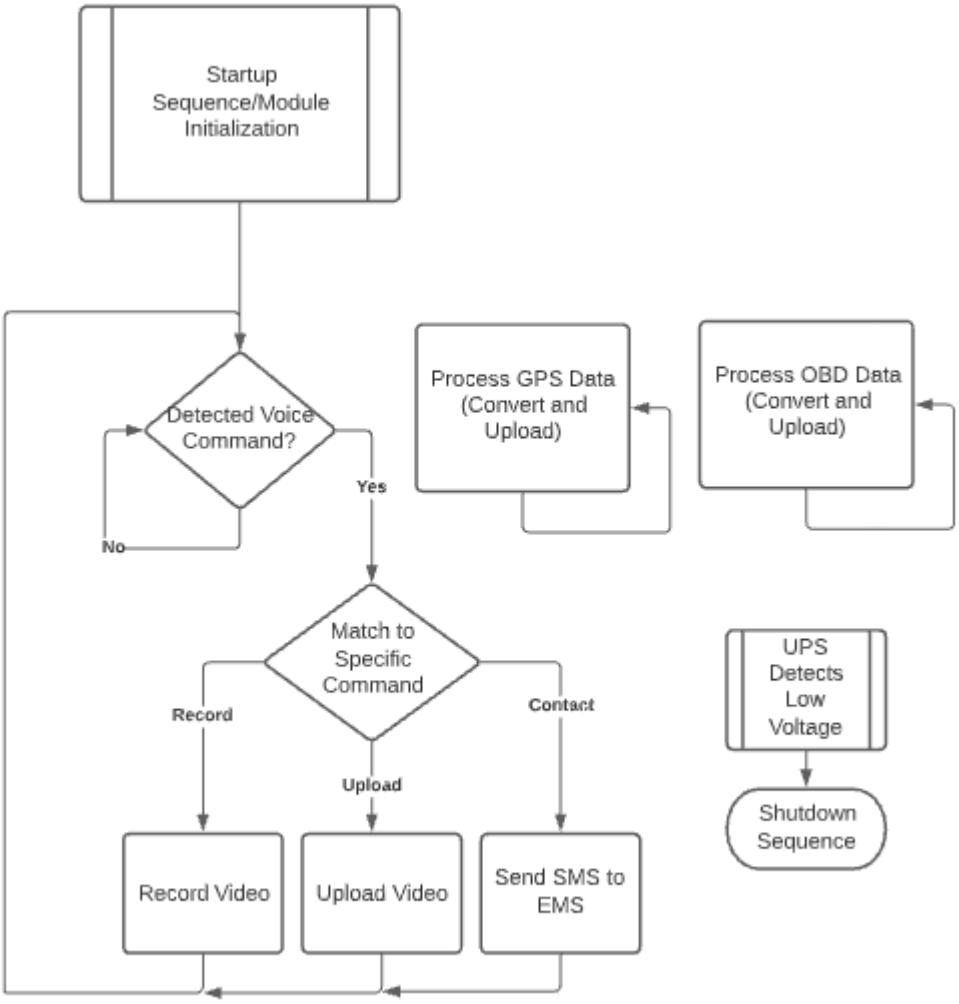
Startup and Exception Handling:

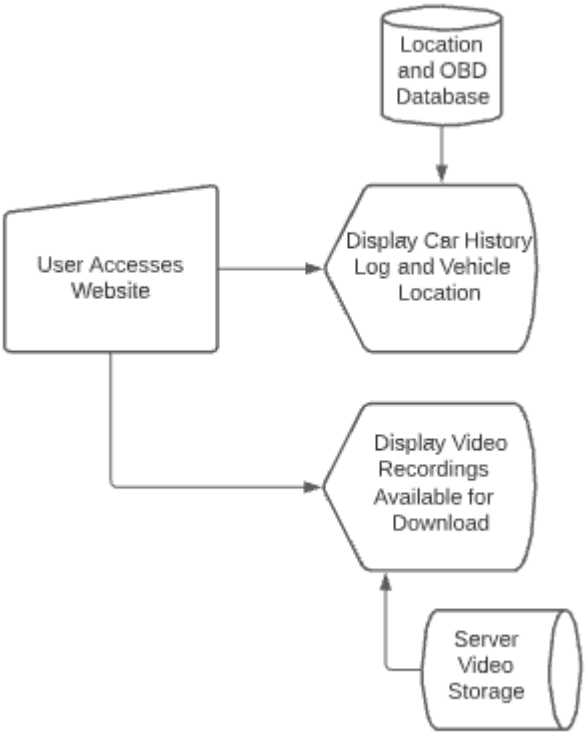
3.0.9.1. Program IDs stored in a JSON file.

3.0.9.2. We can use signals() to determine if a Python script is running if we have its program ID. First, our driver program creates a JSON object with entries for all the individual programs. It then calls all of these scripts using an ampersand argument to background the process. Then, it periodically polls the program to make sure they are still running. If they are not, the script is run again and an error counter increments. If ten errors are encountered there is assumed to be a hardware issue and a reboot command is executed.

3.0.9.3. Forked Python scripts and possibly a reboot command.

### 3.1 Software Inter-Relationships Block Diagram






### 3.2 Pseudo code of critical software elements

#### 3.2.1 VSSM.py

```
check program IDs in file
    if voice sub process is not running
        start it
    if GPS sub process is not running
        start it
    if OBD sub process is not running
        start it

while VSSM is running:
    go to check voice function
    go to check GPS function
    go to check OBD function
    if error count >= 10
        restart VSSM

check * function:
    if the sub process is not running:
        start sub process
        increment error count
```

### 3.2.2 Voice.py

```
save program ID to file

if microphone is not detected
    exit

create VOSK object using Kaldi and Sound Device

while VSSM is running:
    get data from VOSK
    if the data is acceptable
        if it matches a command
            go to matching function

record function
    if disk is full
        remove files
    if it is night time
        record video, boost iso
    otherwise
        record video
    convert to mp4 and save it

upload function
    find newest video in directory
    POST to website using requests

contact function
    flash SOS on UPS LED
    send AT command for SMS
```

### 3.2.3 GPS_Upload.py

```
save program ID to file

initialize GPSD

if GPS device is not found
    exit

while VSSM is running:
    get position data from GPS device
    convert data to PST
    POST to website using requests
```

3.2.4 OBD_Upload.py

```
save program ID to file

request vehicle engine RPM

if vehicle is unresponsive
    exit

while VSSM is running:
    get DTCs from vehicle
    if there are DTCs
        POST to website using requests
```

3.2.5 Upload Pages (Website)

```
if the request is a POST on a database page
    parse data into database friendly statement handles
    use PHP data object to connect to database
    execute statement handle

if the request is a POST on the recording page
    try to receive video and move it to download directory
    if it was successful
        return file uploaded
    otherwise
        error
```

3.2.6 Google Maps JavaScript API (Website)

```
Index:
user provides data to look up on home page
    iterate through database entries
        format entries as XML
    send request and redirect user to map page

Map Page:
if request is GET
    call map initialization using API key

    create a new map centered on Chico

    for each entry in the XML file:
        Use ID and Time to create pin
        Give pin a location using latitude and longitude
        Give the pin to the map object

create statement handle to retrieve DTC data
use PHP data object to connect to database
execute statement handle, store in array
while the array has entries:
    display an entry
```

### 4.0 Glossary

| Word or Acronym | Meaning in This Context |
|---|---|
| GPS | Global Positioning System |
| OBD | On-Board Diagnostics |
| HTML | Hyper-Text Markup Language |
| PHP | PHP: Hypertext Preprocessor |
| DTC | Diagnostic Trouble Code |
| UPS | Uninterruptible Power Supply |
| EMS | Emergency Medical Services |
| API | Application Programming Interface |
| USB | Universal Serial Bus |
| UART | Universal Asynchronous Recevier/Transmitter |

| | |
|---|---|
| Pi | Raspberry Pi 3 B+ |
| UHF | Ultra-High Frequency |
| GND | Ground |
| RPM | Rotations per Minute |
| XML | Extensible Markup Language |
| JSON | JavaScript Object Notation |

**5.0 Appendix** – Reference documents and hardware interface details

| No. | URL |
|---|---|
| 1 | https://www.pishop.us/product/gps-antenna-vk-162/ |
| 2 | https://sixfab.com/product/raspberry-pi-base-hat-3g-4g-lte-minipcie-cards/ |
| 3 | https://www.raspberrypi.org/products/camera-module-v2/ |
| 4 | https://www.amazon.com/Forscan-Scanner-ELMconfig-FoCCCus-Diagnostic/dp/B07MQ8GHG3/ref=sr_1_7?dchild=1&keywords=elm327+usb&qid=1619120371&sr=8-7 |
| 5 | http://www.allspectrum.com/mopower/ |
| 6 | https://sixfab.com/product/quectel-ec25-mini-pcie-4g-lte-module/ |
| 7 | https://www.amazon.com/gp/product/B08CC2BL4X/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1 |
| 8 | https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ |
| 9 | https://sixfab.com/product/lte-main-diversity-gnss-triple-port-u-fl-antenna-100mm/ |
| 10 | https://python-sounddevice.readthedocs.io/en/0.4.1/ |
| 11 | https://alphacephei.com/vosk/ |
| 12 | https://picamera.readthedocs.io/en/release-1.13/ |
| 13 | https://gpsd.gitlab.io/gpsd/ |
| 14 | https://pypi.org/project/pytz/ |
| 15 | https://dateutil.readthedocs.io/en/stable/ |
| 16 | https://pypi.org/project/pyusb/ |
| 17 | https://python-obd.readthedocs.io/en/latest/ |
| 18 | https://pypi.org/project/requests/ |
| 19 | https://developers.google.com/maps/documentation/javascript/overview |
| 20 | https://github.com/gpac/gpac/wiki/MP4Box |