# User Manual

## Part 1: Accessing the Pi

Since the Pi is mounted in the GLV box, SSHing into it is very advantageous for debugging code and accessing files. In order to do so, start by finding the pi's address.

**Finding the Pi's Address**

There are many ways to do so, but in my findings, the best way to do so is by first connecting a keyboard to the GLV.

If the GUI is fullscreen, start by clicking <alt><F4> to exit the program, then using the touchscreen, open terminal.

Once the terminal is open, type "hostname -I".

The terminal will return the current address of the Pi.

**SSHing into the Pi**

Once you have the address, you can SSH into the pi on any computer connected tot the same network by typing "ssh pi@address"

You will then be prompted for a password, which is "raspberry"

Once your'e in the pi, the program is located in a VSCADA folder on the Desktop

**Running the program**

Since we are using QT to create our GUI, there is a restriction in that we are unable to run the program from SSH. The tool we used to workaround this issue is called VNC Viewer. This allowed us mirror the pi's desktop on our personal machines.

Once you are on the desktop of the Pi, launch the terminal.

From terminal, change directories into the VSCADA folder by typing: "cd Desktop/VSCADA"

Then, type: "sudo python3 getCANData.py"

This will launch the UI on the GLV and and the dashboard. In order for it work, the USB display must be plugged in, as the buttons on the display are important to the operation of the program.

**Running Candump**

The PiCAN 2 came with an example C library that has simple functions such as candump and cansend (documentation for this library available on the PiCAN 2 User Manual online). The library is downloaded onto and Pi's desktop and the makefile has been run. In order to run candump, first get to the correct directory using the command:

"cd Desktop/can-"

Once in the directory, you run a full candump using the command:

"./candump can0"

To isolate one address from the CAN network, you can use a command like:

"./candump can0 | grep 0F4"

This will show you every time the packet address 0x0F4 shows up on the network.

# Part 2: Code Base

The code base for the Pi is on a public repository hosted on Github (http://github.com/watsongd/VSCADA). The code for receiving, processing, and displaying the CAN data is in the file "getCANData.py".

# Part 3: Communicating with Dashboard Display

The USB serial dashboard display requires serial communication and the manufacturer provided example code (Crystalfontz website). In order to communicate with the display using python, C functions were developed and imported into C. This is **NOT** the best way to do this. It is very convoluted and requires a serial connection to be established and disconnected. In the future, this serial communication should be done entirely in python and not using the example code.

**Buttons**

The dashboard display has 3 relavent buttons right now and three open buttons that could be programmed to do something.

Green Check Mark: Start recording – system will begin logging data to the data base if and only if this button is pressed. We suggest pushing it at the beginning of a run to ensure all data is captured. GLV display will indicate when SCADA is recording.

Red 'X': Stop recording – system will stop writing data to database after this button is pressed. Also exports the data files.

Right Arrow: Data recovery – recovers previous session data incase of catastophic failure

# Part 4: Program Details

**Database and Sessions**

The database currently has one table that holds all data from every session. A session is just an integer that increments everytime a recording session is finished whether its by pushing the stop recording button or by the GLV turning off. The sensor_id, sensorName, dataValue, time, system, pack, flagged, csv_out, session_id.

Sensor_id: a unique id that every sensor is assigned. You can see all of these values in the dictionary at the top of 'getCANdata.py'

sensorName: Name of sesnor. Helps with readability

dataValue: What is sent on the CAN network

time: Time entry was logged relative to start time

system: TSV, TSI, or MC

pack: pack number (1,2,3,4) or 0 if not a pack sensor

flagged: 1 if value is outside the given threshold in the config file

csv_out: determines if the data in the entry will be written to the csv. Comes from config file

session_id: Id of session

**Conifguration File**

The config file lists every sensor that is being sent over CAN. The config file can set thresholds for each sensor. If the user wants the car to drop out of drive mode if the data value is outside the thresholds that can be done in the config. If not the user will simply get a warning in the .log file. The entry will also be flagged if it lies outside the range. If the upper and lower thresholds are the same no check is performed.

The next two columns determine if the sensor data is placed in the database and if it is placed in the csv upon export.

Config files follow this format:

| sensor_id | lower_threshhold | upper_threshold | drop_out? | log_en | csv_en |
|---|---|---|---|---|---|

An extra column can be added to act as a comment to increase readability

Example:

| 13 | 2.7 | 3.6 | 0 | 1 | 1 |
|---|---|---|---|---|---|

This will flag data values outside 2.7 and 3.6. It will not drop out of drive mode. It will log the data and export it to the csv.

**Data Files**

The system produces data after the ‘Stop Recording’ button on the dashboard is pressed. The data is stored in the following format as a .csv file:

| sensor_id | Elapsed Time from Start | Data Value | Flagged? |
|---|---|---|---|

A single line from a data file might look like this:

| 117 | 2:35 | 3.5 | 0 |
|---|---|---|---|

The system exports two files at the end of every session. One file is called ‘car_data_all.csv’ and that contains all of the data from every session in the database. The other is called ‘car_data_session_{id}’ which just includes the data from the current session.

If the car shuts down unexpectedly for any reason the right button on the cockpit can be pressed and data from the previous session will be recovered and exported as a file called ‘car_data_recovery_{id}’. It can also be seen at anytime in the ‘car_data_all’ file after the next export.

**Flash drives and Files**

This program will work with any flash drive that has the two key files on it. The first is a file called ‘lafayetteSCADA.txt’ which is just an empty text file. This indicates to the system that a valid flash drive is present and csv files should be exported there. If that file is not present csv files will be stored on the desktop of the pi in a folder called ‘VSCADA_CSV_FILES’.

The second is ‘config.csv’ which tells the system to read the config file on the flash drive. IF ‘config.csv’ IS NOT PRESENT THE SYSTEM WILL USE THE CONFIG FILE FROM THE GIT REPO. This feature was added so that the configuration of the system could be changed quickly by pulling the flash drive, making changes, and restarting the program.

We recommend that you put a conservative and safe ‘config.csv’ on the repo as a fall back and leave it alone. Then if you want to change the configuration use the flash drive.

# Part 6: Graphic User Interface

**PyQt5 GUI**

The GUI of VSCADA system is built using PyQt5 toolkit. PyQt is a set of Python v2 and v3 bindings for the Qt application framework and runs on all platforms supported by Qt including Windows, OS X, Linux, iOS and Android. Since PyQt brings together the C++ cross-platform application framework and the cross-platform interpreted language Python, it is a convenient and flexible tool to create a UI program that can be integrated to another Python program (VSCADA). The GUI and other parts of VSCADA run in parallel as multiple threads. In this way, the UI program can be modified by multiple programmers in different laptops or platforms without affecting other parts of VSCADA.

The GUI of VSCADA will run only when PyQt related libraries are installed; otherwise, the built-in Qt commands in UI program cannot be executed. Libraries required are Python3, SIP, PyQt5. After Python3 is confirmed to be installed, we ran the following commands to install the required libraries:

sudo pip3 install SIP

sudo pip3 install pyqt5

**VSCADA Display**

The GUI display is divided into 4 sections. The VSCADA section displays state, session number and session time. A REC button in VSCADA section can be clicked and it can start or end recording vehicle data which will be exported as a file. MOTOR section displays speed, temperature and throttle voltage input of motor. TSI section displays IMD number, current drawn and throttle voltage output from TSI. The section that takes the bottom part of the screen is all for voltage, temperature and state information for 4 battery packs, motor controller and TSI. On the lower-right corner, there is log box which will display error message from the car including when drive mode is dropped out. On the left, there are 2 green label which shows the status of air and brake. The background color of the label is green when airs and brake is on, and red when they are off.