Carberry Pi Documentation

Ryan McHugh

April 2020

Carberry Pi

```
     11111111
    11   11   1
 111     11    1111
1111111111111111111111
1111001111111111110011111
  100001         100001
    00             00
```

0.1 Outline of this Document

1 Introduction

- *Carberry Pi* is an automotive application of a mini-computer in the car. As the quintessential project for my undergraduate studies, this concept provides a deep-dive into an area of future interest.

2 Hardware

Carberry Pi requires a few tools of the trade.

Namely:

- Raspberry Pi (this project uses a Raspberry Pi 3 model B)
- Professional Grade OBDII Cable
- Raspberry Pi Touchscreen
- DS3231 RTC IC (Real Time Clock)

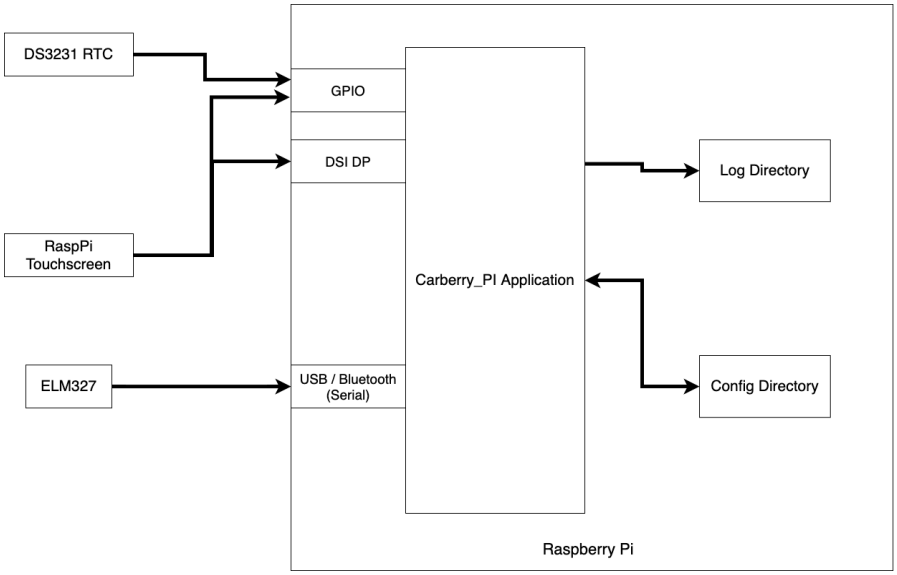

Figure 2.1: Hardware Overview

3 Carberry Pi Software

Dashboard

Displays a speedometer, tachometer, and coolant temperature gauge.

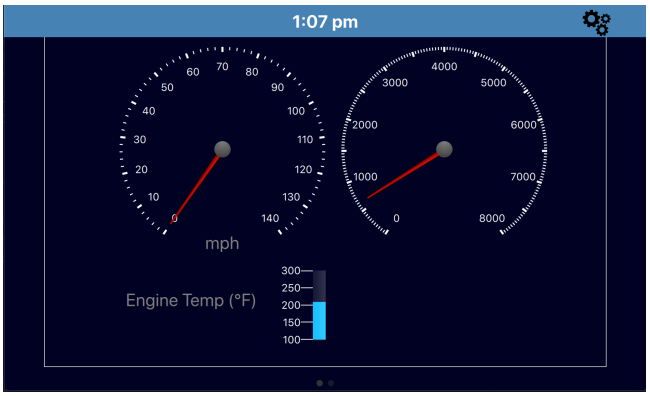

Figure 3.1: Dashboard

Diagnostics

Displays a list of all data points provided by the Car Computer.

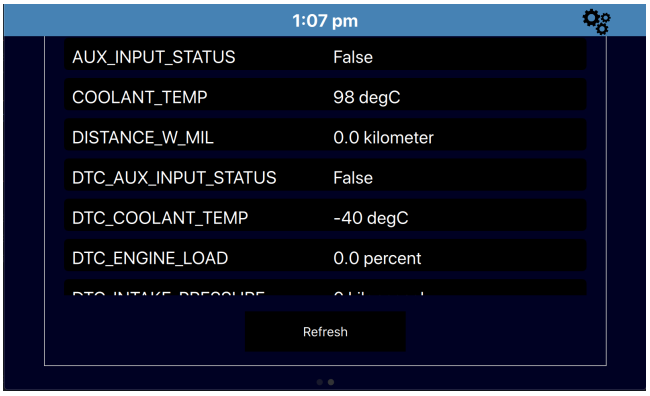

Figure 3.2: Diagnostics

Configuration

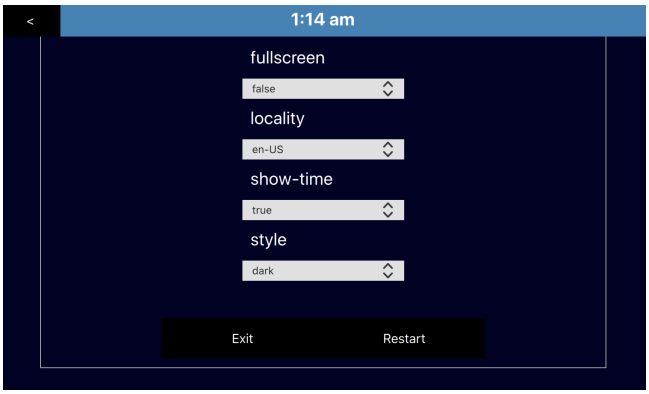

Figure 3.3: Configuration

Manage the settings of the application.

- locality: Region-based conversion of units (main dashboard only)
- fullscreen: Toggle for fullscreen on startup (only works on
- style: Manages the overall theme of the application (toggle between light and dark modes)
- time: Show/Hide the time in the header bar

Application must be restarted for some preference changes to take effect.

Architecture

Toolkit

- Backend: Python
  - Utilizes python-obd library for OBD information
- Frontend: PyQt (Qt-Quick) + QML | Javascript

Interface Architecture

- Dynamic loading allows react.js style module instantiation and destruction
  - Each component is loaded into a *view* as a separate entity
  - These components can then be pushed/popped onto or from the main*stackview*

  - A separate script (javascript) manages the creation/destruction of the *back* button

- Time
  - The time is based on the RTC (Real Time Clock) of the Raspberry Pi itself.
  - As such, changing the locality has no effect on the time value.

Directory Structure | File Enumeration

- *documentation*: Stores the source files and compiled containers for this documentation

- *src*: Contains the Project Source files
  - *items*: reusable _custom_ QML items
  - *js*: JavaScript scripts (primarily for object creation and destruction)
  - *log*: storage for log output *(YYYY-MM-DD)*
  - *partials*: QML partials (snippets)
  - *resources*: assets | icons

Runtime Overview *run.py* lasts the runtime of the program. It initializes the GUI through the pyqt application engine. Then the front end initializes communication with *run.qml*. In the meantime, *run.py* (backend) begins syncing between the car and the application. This data is then read into a dictionary object, then passed to *context.py* - a middleman between the backend and frontend. The GUI is then updated, and the process of *read, update, display* is repeated through the runtime of the program.

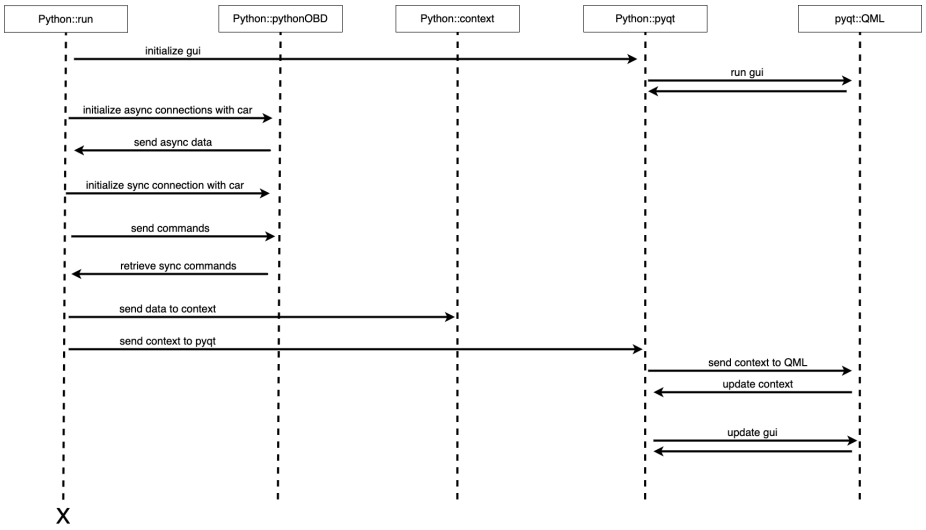

Figure 3.4: Sequence Diagram

4 Connecting the Pieces

4.1 Step 1: Gather the Equipment

Figure 4.1: Step 1

4.2 Step 2: Getting Connection

- Screw the raspberry pi into the board posts at the rear of the screen.
- Connect the displayport cable from the screen to the Pi (the white ribbon cable)
- Connect the pins from the GPIO pins to the screen as pictured in *Figure 4.2.* [pins 1,2 –> pins 1,5]

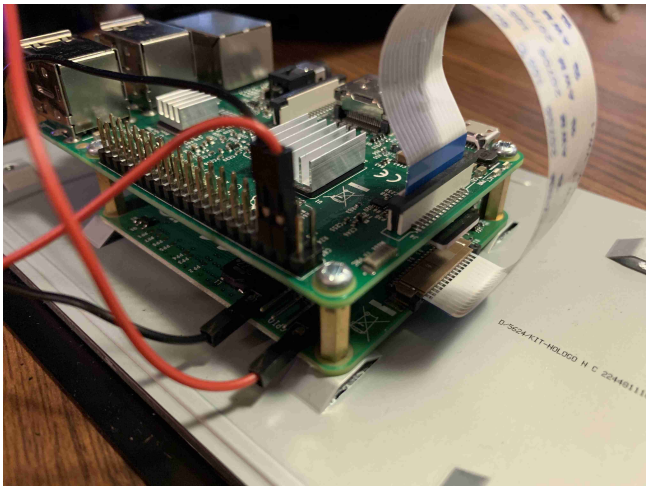

Figure 4.2: Step 2

4.3 Step 3: RTC Module

- Attach the RTC module to the first four GPIO pins parallel to the previously-connected cables.

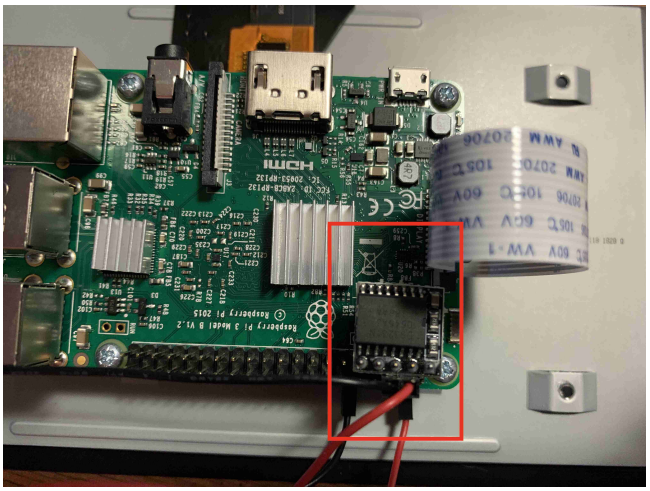

Figure 4.3: Step 3

4.4 Step 4: Screwing It All Together

- Line up the back casing with the raspberry pi.
- Place the screws into each of the four highlighted holes in *Figure 4.4*
- Attach the back cover

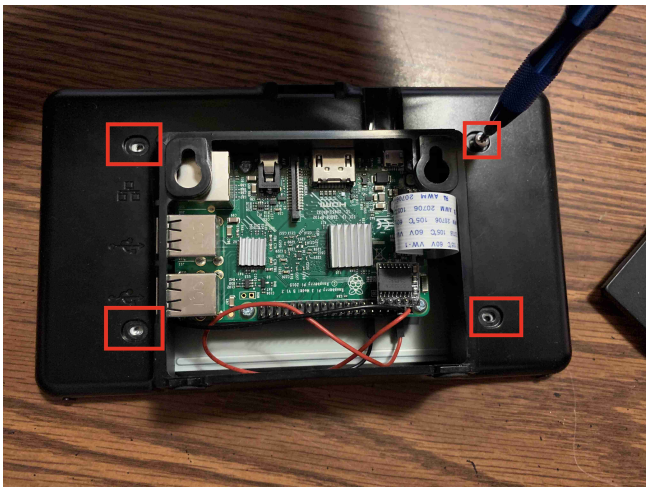

Figure 4.4: Step 4-1

Figure 4.5: Step 4-2

4.5 Final Product

Figure 4.6: Final Product

5 Getting Up and Running

Recommended OS: DietPi The *DietPi* (debian-based) operating system distribution acts as a lightweight desktop environment for running GUIs on the Pi.

Of Course, you may run this application on another operating system of your choosing.

Recommended DE: LXDE This project uses *LXDE*. It is a lightweight desktop environment that suits the limited hardware of the Raspberry Pi wonderfully.

The use of another desktop environment will require appending a command that executes the *start_carberry.sh* script to the startup file of the respective DE.

Note: The RTC Module requires the installation of a module-specific driver

Installation

1. Clone the repo from `https://github.com/brohemz/carberry-pi`
2. Run *install.sh* in the */* folder as SuperUser
    - Note: The *autostart* functionality of the installation script requires LXDE.
3. In order to ensure proper *autostart* functionality, restart the computer now.
4. Run the application *start_carberry.sh* from *src* folder.

The application should now launch to the main dashboard.