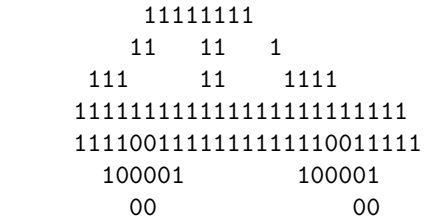
Carberry Pi Documentation

Ryan McHugh

April 2020

(1,0)300

Carberry Pi



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)

3 Carberry Pi Software

Dashboard

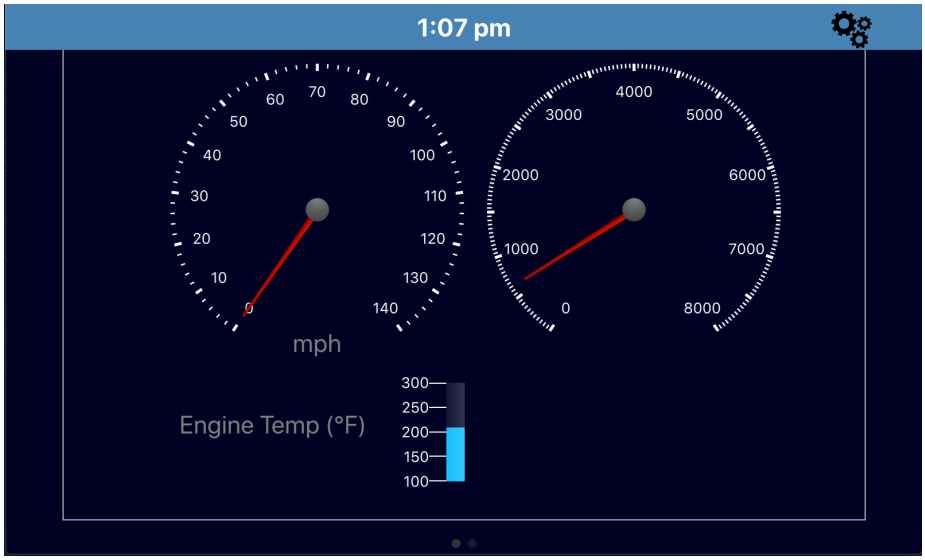


Figure 1: Dashboard

Diagnostics

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

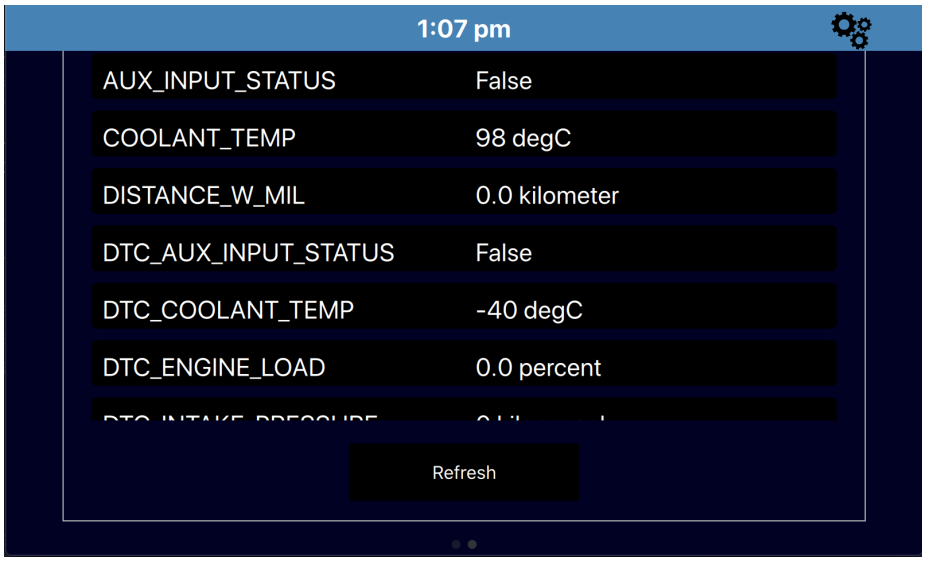


Figure 2: Diagnostics

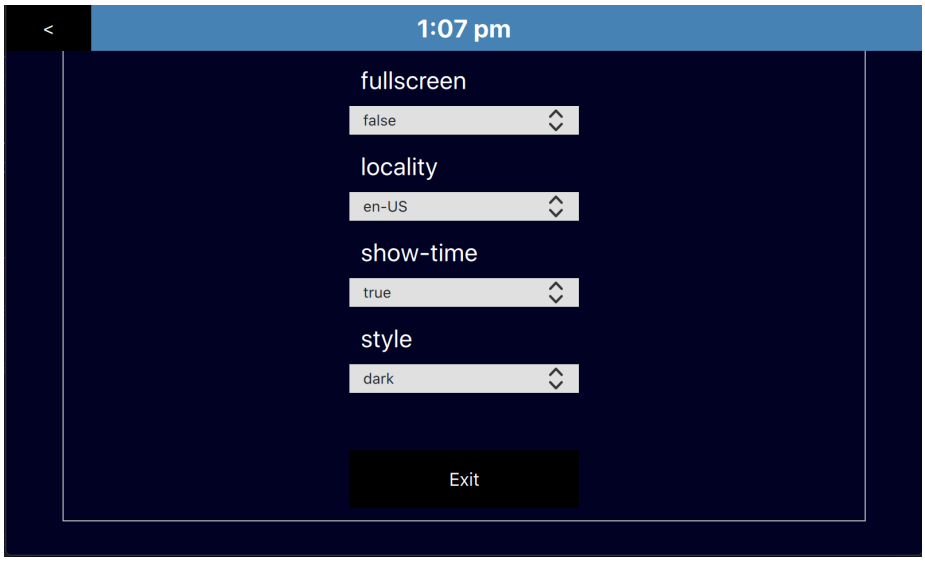


Figure 3: Configuration

- style: Manages the overall theme of the application (toggle between light and dark modes)
- time: Show/Hide the time in the header bar

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

4 Connecting the Pieces

// tutorial with picture layout of connecting each component

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.

Installation

1. Clone the repo from https://github.com/brohemz/carberry-pi
2. Run *install.sh* in the *src* 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.

You should now see the main dashboard.