SCADA Maintenance Manual

ECE 492 - Spring 2019

Latest Revision: 04/16/2019
Prepared by: Zian Geng
Revised by: Samuel Mwaura

Abstract

This document is intended for provide information about the components used in the SCADA system, their setup and functionality.

Table of Contents

Introduction

The SCADA is the Supervisory Control and Data Acquisition System used for the vehicle. The SCADA for 2019 team is able to communicate with cooling system, TSI, GLV, motor controller and Huff Box in Dyno room. The data will be displayed on the Raspberry Pi and saved in an SQLite database as per configuration. This document by itself will not provide adequate information to run the program. Please refer to the user manual on our website to successfully run the program.

Software

Setting up Cross-Compilation Platform and Installation

Since we use the Qt platform to develop SCADA on a Linux Fedora platform, we need to cross compile the program to enable it to run on the Raspberry Pi. Prior to cross compilation, the steps outlined in the link: *https://wiki.qt.io/Raspberry_Pi_Beginners_Guide* need to be performed. After this, compiling the program on Qt is as usual (pressing the build button). To deploy this program, go to the build directory on the Linux Fedora machine and run the command (replace PI-IP_ADDRESS with actual pi IP address : type hostname -I on the Raspberry Pi):

scp ./VSCADA pi@PI-IP_ADDRESS:~

Also do:

scp /PATH/TO/config.xml pi@PI-IP_ADDRESS:~

to move the configuration file to the Pi's home address. An example config file can be found on the project website.

Programming Language: C++

We use the C++ code for most of the software functions in SCADA. It allows as to have more control over the functionality and design of the SCADA. In addition, it helps us to communicate with other subsystems in a more generic where we can easily configure a new sensor for SCADA through C++ Code. The C++ Code now has the gpio interface, can interface, configuration file interface, huff box interface and subsystem thread which includes all the data necessary for each subsystem. In order to run C++ program on the Raspberry, we will program the C++ code with QT and use cross compiling to clone our program to the raspberry. We highly recommend continuing using C++ in the future design.

Configuration

We use xml file as configuration file for the SCADA. The configuration contains all the constant used for each subsystem and state machine of the car, include the calibration constants, the CAN bus address, the sensor names, the threshold, etc. By using an xml, we can easily access and control all the setting of the subsystems in a generic way so any addition to the SCADA will be easy.

Data Capture and Storage

Data is selectively captured and stored. This is determined by the ‘recordwindow’ section of the xml configuration file. This section configures data to be captured only under certain conditions, also specified in the configuration file (please refer to the ‘recordwindow’ section of the configuration file for more detail). Once data is captured, it is stored in csv files to a path and filename prefix specified in the configuration file. If the program is closed while collection is in progress, data collection will be halted and the captured data will also be stored to the specified path. Besides the specified prefix, there will be a session number appended to each file to keep track of each recording session. This also prevents overwriting of data files.

Hardware

Raspberry Pi 3B

We are using the same Raspberry Pi used in the previous years. With the cross compiling, it can successfully run our C++ program with lower lag compare to the Python code used previously. Beside the CAN bus interface, we used I2C to read data from GLV sensor, and use gpio interface for SCADA relay and fault message transmission. In addition, we also connected the Raspberry Pi with the Huff box for Dyno room test. The Raspberry Pi has supported all the test and functions we planned without any problem.

Real Time Clock

A real time clock for Raspberry Pi is connected through an external board to the Raspberry Pi. It will count the real time with its own battery source and send the real time to SCADA at the begin of each reboot once it is set through terminal. The RTC helps us to save data in real time stamp.

PiCAN2

The PiCan2 board is used as can interrupt which is the same board we used in previous year. It works fun with the CAN interface we write in C++ code. It supports the can communication with the TSI board without any problem. And we are still using can dump for this year.

Rack Display

The Rack Display is connected to the Raspberry Pi to perform test in Dyno Room. There is not any problem with the monitor functions of the display. It helps us to perform test on Huff box and all the other subsystems further in dyno room.