SCADA User Manual

ECE 492 - Spring 2019

Latest Revision: 4/16/2019
Prepared by: Samuel Mwaura

Abstract

This document is intended for use as a user manual for the SCADA system. It may only prove useful after familiarizing yourself with the contents of the ‘SCADA Maintenance Manual’, unless you already have a working environment set up.

Table of Contents

Preliminary System Setup

This section is meant to ensure that you have the framework needed to run the SCADA software. To set the program up for running, please follow the following procedure:

Program Setup & Sanity Check

a. Create a new folder (I'll call it *SCADA_folder* in this document) on the Raspbian filesystem. Preferably put this folder in the home directory
b. Transfer the built SCADA file into *SCADA_folder*
c. Copy all the files in the folder *aux_files* and place its contents in *SCADA_*folder
d. Create a configuration file as specified in the next section and place it in *SCADA_folder*
e. As a sanity check, ensure you have CAN, I2C and GPIO utilities installed as stated in the maintenance manual

Configuration

To configure the SCADA software, a configuration file named config.xml must be created (XML format compliant) and placed into *SCADA_folder*. This section is a guide on how to write a custom configuration file that's runnable through SCADA.

WARNING: If there's an XML-related syntax error in your file, SCADA will only read through the file up to where the error is located. What comes after an error is not processed by SCADA.

The XML format is more human readable with the <tag>value</tag> format than most other file formats. The tags and value characteristics are outlined below:

Mode

This configures the mode in which the SCADA will be launched. There are two possible values: RUN - launches SCADA and actively listens/samples for configured sensors; and TEST - runs pseudo-random numbers through the data collection framework to test the system's build integrity. An example is shown on Figure 1.

GPIO Sampling Rate

This specifies the sampling rate of the GPIO interface (all GPIO sensors) in milliseconds. An example is shown on Figure 1.

USB 7204 Sampling Rate

This specifies the sampling rate of the USB 7204 interface (all USB 7204 sensors) in milliseconds. An example is shown on Figure 1.

CANbus Bitrate

This specifies the bitrate for the CANbus interface. An example is shown on Figure 1. Since CAN is set up as a listening port, there's no sampling rate. CAN sync signals can be set up to request data from CAN and CANOpen nodes.

```
<mode>RUN</mode>
<gpiorate>1000</gpiorate>
<usb7204rate>1000</usb7204rate>
<canrate>125000</canrate>
```

Figure 1

Group

This specifies a display grouping of sensors. The sensors listed will be displayed together under one group name. Available child nodes include:

i. **name** - Name of the group
ii. **sensorid** - Id's of sensors belonging to this group

See example on Figure 2

```
<group>
    <name>GLV</name>
    <sensorid>11</sensorid>
    <sensorid>12</sensorid>
    <sensorid>13</sensorid>
    <sensorid>14</sensorid>
    <sensorid>15</sensorid>
    <sensorid>16</sensorid>
</group>
```

Figure 2

Boot Sequence

This configures a set of commands that are sent by SCADA on boot. Available child nodes include:

i. bootcan - CAN command to be sent. Also has child nodes:
    - address - CAN address

- data - data to be sent

ii. booti2c - I2C command to be sent on boot. This command contains an address (7-bit) as per i2c specifications.

See example on Figure 3.

```
<bootsequence>
    <bootcan>
        <address>20</address>
        <data>23452</data>
    </bootcan>
    <bootcan>
                <address>20</address>
                <data>23542</data>
        </bootcan>
    <bootcan>
                <address>20</address>
                <data>234552</data>
        </bootcan>
    <booti2c>2572544</booti2c>
    <booti2c>2565</booti2c>
    <booti2c>523</booti2c>
    <booti2c>260</booti2c>
    <booti2c>770</booti2c>
</bootsequence>
```

Figure 3

Sync

These nodes configure commands that are sent periodically by the SCADA system. Available child nodes include:

i. cansync - CAN commands to be sent periodically. Its child nodes include:
   - address - CAN address to be used
   - data - data to be sent
   - bytes - size of data in bytes
   - ratems - rate in milliseconds

ii. i2csync - I2C commands to be sent periodically. Its child nodes include:
   - address - I2C address to be used
   - data - data to be sent
   - ratems - rate in milliseconds

iii. gpiosync - GPIO pins to be set periodically. Its child nodes include:
   - pin - GPIO pin to be set
   - data - value to be set on pin
   - ratems - rate in milliseconds

See example on Figure 4

```
<sync>
    <cansync>
        <address>128</address>
        <data>0</data>
        <bytes>1</bytes>
        <ratems>1000</ratems>
    </cansync>
    <i2csync>
        <address>49</address>
        <data>0</data>
        <ratems>1000</ratems>
    </i2csync>
    <gpiosync>
        <pin>21</pin>
        <data>1</data>
        <ratems>1000</ratems>
    </gpiosync>
</sync>
```

Figure 4

System Controls

This configures on-screen controls that can be used to send custom data. There are 3 control interface options: buttons, sliders and text input fields. Child nodes for system controls include:

i. name - name of the control
ii. type - type of control as stated above
iii. minrange (for slider) - minimum value of slider range
iv. maxrange (for slider) - maximum value of slider range
v. multiplier (optional) - multiplier for slider value. Default is 1
vi. usbchannel (optional) - channel to send data if configured
vii. primaddress (optional) - CAN address if configured for CAN
viii. auxaddress (required for CAN) - section of data to affect
ix. offset (required for CAN) - size of data to affect (in bits). Values allowed : 1-64
x. gpiopin (required for GPIO) - pin to affect
xi. pressvalue (for button) - value sent when button is pressed
xii. releasevalue (for button) - value sent when button is released

See example on Figure 5

```
<systemcontrols>
    <control>
        <name>Valve</name>
        <type>slider</type>
        <minrange>0</minrange>
        <maxrange>100</maxrange>
        <multiplier>0.05</multiplier>
        <usbchannel>0</usbchannel>
    </control>
    <control>
        <name>Brake</name>
        <type>button</type>
        <primaddress>301</primaddress>
        <auxaddress>60</auxaddress>
        <offset>1</offset>
        <pressvalue>1</pressvalue>
        <releasevalue>0</releasevalue>
    </control>
    <control>
        <name>MC Tests</name>
        <type>textfield</type>
        <primaddress>301</primaddress>
        <auxaddress>52</auxaddress>
        <offset>8</offset>
    </control>
</systemcontrols>
```

Figure 5

System Statuses

This configures system statuses to be displayed on the screen in binary form i.e. active/inactive, true/false, high/low. This is displayed as a button on the screen that changes color depending on a true/false status. Statuses are currently only configured to be received over CAN.
Child nodes include:

i. name - name of status
ii. primaddress - primary CAN address
iii. auxaddress - section of data to check
iv. offset - size of data being checked
v. value - value to check for. Status is ‘activated’ if the checked data matches this value

See example on Figure 6

```
<systemstatus>
    <name>Brake</name>
    <primaddress>300</primaddress>
    <auxaddress>61</auxaddress>
    <offset>1</offset>
    <value>1</value>
</systemstatus>
```

Figure 6

Responses

These configure responses to incoming data. Each sensor can have a configured reaction for when its value exceeds its upper threshold, lower threshold, or is within the thresholds. Responses are currently meant to only send/effect CAN and GPIO data. Child nodes include:

i. id - Response identifier
ii. description - brief text on what the response does
iii. primaddress - primary CAN address
iv. auxaddress - section of data to send/effect
v. offset - size of data to send/effect
vi. canval - value to send over CAN
vii. gpiopin - GPIO pin to change/effect
viii. gpioval - GPIO value to write

See example on Figure 7

```
<responses>
    <response>
        <id>0</id>
        <description>Ignore Event</description>
    </response>
    <response>
        <id>1</id>
        <description>Writing to GPIO pin</description>
        <gpiopin>20</gpiopin>
        <gpioval>1</gpioval>
    </response>
    <response>
        <id>2</id>
        <description>Writing to CAN and GPIO</description>
        <gpiopin>4</gpiopin>
        <primaddress>123</primaddress>
        <auxaddress>0</auxaddress>
        <offset>32</offset>
        <gpioval>1</gpioval>
        <canval>345</canval>
    </response>
    <response>
        <id>3</id>
        <description>Writing to CAN</description>
        <primaddress>345</primaddress>
        <auxaddress>0</auxaddress>
        <offset>32</offset>
        <canval>33</canval>
    </response>
    <response>
        <id>4</id>
        <description>SCADA Relay Closed</description>
        <gpiopin>21</gpiopin>
        <gpioval>1</gpioval>
    </response>
</responses>
```

Figure 7

State Machines

Configures system state machines. Various states are specified as well as all the conditions that affect the state machine. State machines are currently only configured to be received through CAN. Note that both the state(s) and condition must share a common base address. Child nodes include:

i. name - name of state machine
ii. primaddress - main CAN address to check
iii. auxaddress - section of data to check for state values
iv. offset - size of data to check for state values
v. state - specifies a state in the state machine
    - name - name of state
    - value - value required to switch to this state
vi. Condition - specifies a condition affecting the state machine. Available child nodes include:
    - name - name of condition
    - auxaddress - section of data to check for condition
    - offset - size of data to check for condition

See example on Figure 8

```
<statemachine>
    <name> Drive States </name>
    <primaddress>1024</primaddress>
    <auxaddress>48</auxaddress>
    <offset>8</offset>
    <state>
        <name>Init</name>
        <value>0</value>
    </state>
    <state>
        <name>Idle</name>
        <value>1</value>
    </state>
    <condition>
        <name>Throttle Pressed</name>
        <auxaddress>0</auxaddress>
        <offset>1</offset>
    </condition>
    <condition>
        <name>Throttle Implausibility</name>
        <auxaddress>0</auxaddress>
        <offset>2</offset>
    </condition>
</statemachine>
```

Figure 8

Record Windows

This specifies an automatic data recording trigger that makes data files available once data collection stops. Child nodes include:

i. triggersensor - sensor that's checked for thresholds to start/stop data recording
ii. startvalue - value beyond which data collection starts
iii. stopvalue - value below which data collection stops
iv. saveprefix - prefix used for data files created by this trigger
v. savepath - path for saving data files created by this trigger
vi. sensors - specifies sensors whose data is recorded by this trigger. Child nodes include:
    - sensorid - id of sensor to be recorded

See example on Figure 9

```
<recordwindow>
    <triggersensor>13</triggersensor>
    <startvalue>0</startvalue>
    <stopvalue>1</stopvalue>
    <saveprefix>test</saveprefix>
    <savepath>./savedsessions/</savepath>
    <sensors>
        <sensorid>11</sensorid>
        <sensorid>14</sensorid>
        <sensorid>15</sensorid>
    </sensors>
</recordwindow>
```

Figure 9

Sensors

This configures individual sensors that are recognized by the system. Its child nodes include:

i. id - sensor identifier
ii. name -sensor name
iii. primaddress - primary CAN address
iv. auxaddress - secondary CAN address
v. offset - size of data specified. See example of CAN sensor on Figure 10

```
<sensor>
    <id>3</id>
    <name>Brake Voltage</name>
    <primaddress>897</primaddress>
    <auxaddress>48</auxaddress>
    <offset>16</offset>
    <checkrate>6000</checkrate>
    <unit>V</unit>
    <main>1</main>
    <minimum>-1</minimum>
    <maximum>100</maximum>
    <minresponse>0</minresponse>
    <normresponse>0</normresponse>
    <maxresponse>0</maxresponse>
    <calmultiplier>0.001</calmultiplier>
</sensor>
```

Figure 10

vi. i2caddress - i2c sensor address

vii. i2creadpointer - i2c pointer for sensor data

viii. i2cconfigdata - i2c command to send to sensor to configure it

ix. i2cdatafield - size of data to be read

x. i2creaddelayus - delay allowed for i2c data to be read in microseconds. See example of i2c sensor on Figure 11

```
<sensor>
    <id>14</id>
    <name>GLVTemp</name>
    <i2caddress>16</i2caddress>
    <i2creadpointer>64</i2creadpointer>
    <i2cconfigdata>523</i2cconfigdata>
    <i2cconfigdata>260</i2cconfigdata>
    <i2cconfigdata>770</i2cconfigdata>
    <i2cdatafield>12</i2cdatafield>
    <i2creaddelayus>20</i2creaddelayus>
    <checkrate>5000</checkrate>
    <main>1</main>
    <unit>C</unit>
    <minimum>20</minimum>
    <maximum>50</maximum>
    <minresponse>0</minresponse>
    <maxresponse>0</maxresponse>
    <calconstant>-283.97</calconstant>
    <calmultiplier>0.37679</calmultiplier>
</sensor>
```

Figure 11

xi. gpiopin - gpio pin to read for GPIO sensor. See example of GPIO sensor on Figure 12

```
<sensor>
    <id>13</id>
    <name>F2 Signal</name>
    <gpiopin>26</gpiopin>
    <checkrate>5500</checkrate>
    <main>1</main>
    <minimum>-1</minimum>
    <maximum>2</maximum>
    <minresponse>0</minresponse>
    <maxresponse>0</maxresponse>
    <calmultiplier>1</calmultiplier>
</sensor>
```

Figure 12

xii. usbchannel - usb channel to read for a USB7204 sensor. See example of USB7204 sensor on Figure 13.

```
<sensor>
    <id>17</id>
    <name>Strain Gauge</name>
    <checkrate>5000</checkrate>
    <main>1</main>
    <minimum>0</minimum>
    <maximum>40</maximum>
    <unit>ftLbs</unit>
    <calmultiplier>75.018</calmultiplier>
    <calconstant>107.33</calconstant>
    <usbchannel>1</usbchannel>
</sensor>
```

Figure 13

xiii. checkrate - rate at which sensor is checked to ensure that data collection is still active. If data is not received within the specified time, a blank field is shown on the screen. Specified in ms

xiv. unit - unit of measurement

xv. endianness - specifies whether values are received in little-endian or big-endian formats. ‘0’ indicates little endian and big endian otherwise

xvi. main - specifies whether the sensor will be shown on the main screen or only on the subsystem data screen. The value ‘1’ enables viewing on the main screen, and ‘0’ disables this feature.

xvii. minimum - lower sensor threshold

xviii. maximum - upper sensor threshold

xix. minresponse - response id for the response triggered when sensor value goes below the ‘minimum’ value specified above

xx. maxresponse - response id for the response triggered when sensor value goes above the ‘maximum’ value specified above

xxi. normresponse - response id for the response triggered when sensor value is between the specified ‘maximum’ and ‘minimum’

xxii. calmultiplier - shorthand calibration value that is multiplied to the raw data value. This operation is done before addition

xxiii. calconstant - shorthand calibration value that is added to the raw data value or added to the result after the raw data value is multiplied by the ‘calmultiplier’.

xxiv. calpoly - specifies a calibration polynomial. Coefficients are stated from the constant, to the first, second, third... coefficient, in that order. See example on Figure 14

```
<sensor>
    <id>9</id>
    <name>Flow Rate</name>
    <primaddress>1024</primaddress>
    <auxaddress>40</auxaddress>
    <offset>8</offset>
    <checkrate>4000</checkrate>
    <main>1</main>
    <unit>L/m</unit>
    <minimum>0</minimum>
    <maximum>100</maximum>
    <minresponse>0</minresponse>
    <maxresponse>0</maxresponse>
    <calmultiplier>1</calmultiplier>
    <calpoly>
        <coef>0.0605</coef>
        <coef>0.03314</coef>
    </calpoly>
</sensor>
```

Figure 14

Running the Program

Once the configuration file has been prepared, the program is ready to run!
Open a terminal window and change directory to *SCADA_folder*. Run the program using:

```
sudo ./VSCADA
```

Besides the configured displayable parameters, please note the presence of the following features:

Interface Reset Buttons

There exists two purple buttons named ‘CAN’ and ‘USB7204’. The CAN reset button resets the CANbus interface by setting can down then up again at the configured bitrate. The USB7204 button resets its interface by destroying its active objects and creating new ones to establish a new connection to a connected device.

Exit Button

On the screen, there’s a bold red button with an ‘X’ printed on it (typical cancel/close button). This button exits the program.

System Log Tab

The SCADA user interface is organized into tabs. The main tab that's shown on boot shows the 'main' sensors configured on each sensor group. The next (already present) tab is the system log tab which shows important messages, errors and warnings generated by the system. The number of messages (scrollable) is restricted to 100 lines. This is to prevent memory leakage over time.

Group Detail Data Pages

On the main page, each group of sensors is displayed under a button which has the group name printed on it. Clicking on this button will open up a window that displays both the 'main' and the 'auxiliary' sensors of each group. Recall that the displaying of a sensor on the main page is determined by its *main* attribute, as specified in the configuration file.

System Outlook