Model Railroad System

2.2.1

Generated by Doxygen 1.8.17

Chapter 1

Examples using OpenLCB / LCC

This documents examples using the Model Railroad System to interface with model railroad layouts using LCC. Most of the examples will be making use of the Raspberry Pi with one or more HATs.

The first example, outlined in Crossing Interchange Controlled by a Raspberry Pi with HATs and LCC, is for an interchange between two railroad lines that cross at a level crossing. It uses a Raspberry Pi to manage a pair of turnouts that implement the interchange between the two rail lines and handle the signaling around the turnouts and around the crossing.

Chapter 2

Crossing Interchange Controlled by a Raspberry Pi with HATs and LCC

This example illustrates how one might implement the control aspects of a simple interchange between two rail lines that meet at a level crossing. The layout module is shown here:

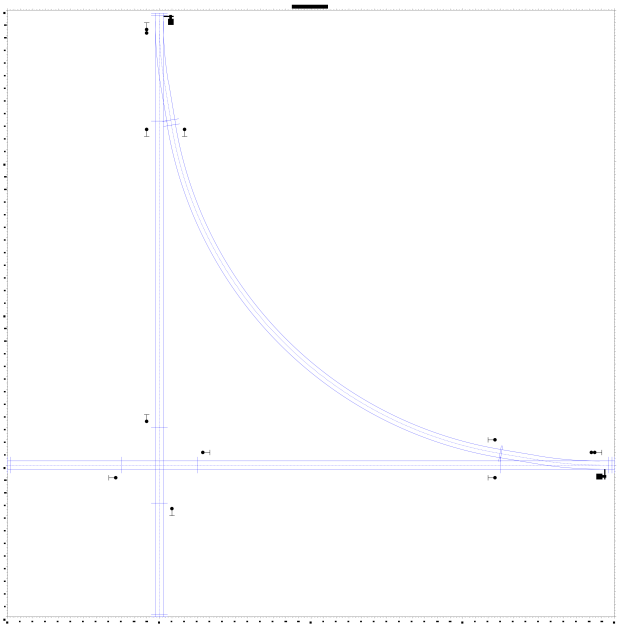

Figure 2.1 Crossing Interchange Layout

1

[1] There is an XTrkCAD file, named `CrossingInterchange.xtc`, and a PDF file, named `CrossingInterchange.pdf`, in the examples distribution directory.

2.1 Hardware used.

We will control the two turnouts with a SMCSenseHAT. This board uses 4 GPIO pins on the Raspberry Pi, two to set the motors and two to sense the state of the switch motors.

Signaling will be with 8 three color single head signals and with 2 3 over 2 double head signals at the point entrance to the turnouts. There is a total of twelve heads and we will use three QuadSignalCA HATs and common anode LED signals.

We will be sensing occupancy of 8 blocks:

1. **OS1** Turnout SW1, at the northern entrance of the north-south rail line.
2. **OS2** Turnout SW2, at the eastern entrance of the east-west rail line.
3. **Crossing OS** The level crossing itself.
4. **Interchange** The interchange track connecting between the two turnouts.
5. **Main One** The east-west mainline between SW2 and the Crossing.
6. **Main Two** The north-south mainline between SW1 and the Crossing.
7. **Main One West** The east-west mainline west of the Crossing.
8. **Main Two South** The north-south mainline south of the Crossing.

We can use a pair of Circuits 4 Tracks Quad Occupancy detector boards, with their outputs wired to 8 of the Raspberry Pi's GPIO pins.

We will have a total of 5 HAT boards:

- 1 **SMCSenseHAT** to manage the two turnouts.
- 1 **Adafruit Perma-Proto HAT** to connect the occupancy detector boards.
- 3 **QuadSignalCA HATs** to light the 12 signal heads.

The **SMCSenseHAT** is hardwired to use Wiring Pi pins 0, 1, 2, and 3 (BCM pins 17, 18, 27, and 22). The PiGPIO XML file included with this example, maps Motor 1 (pin 0 for motor control and pin 2 for point sense) to SW1 and motor 2 (pin 1 for motor control and pin 3 for point sense) to SW2.

The **SMCSenseHAT** is connected as shown here:

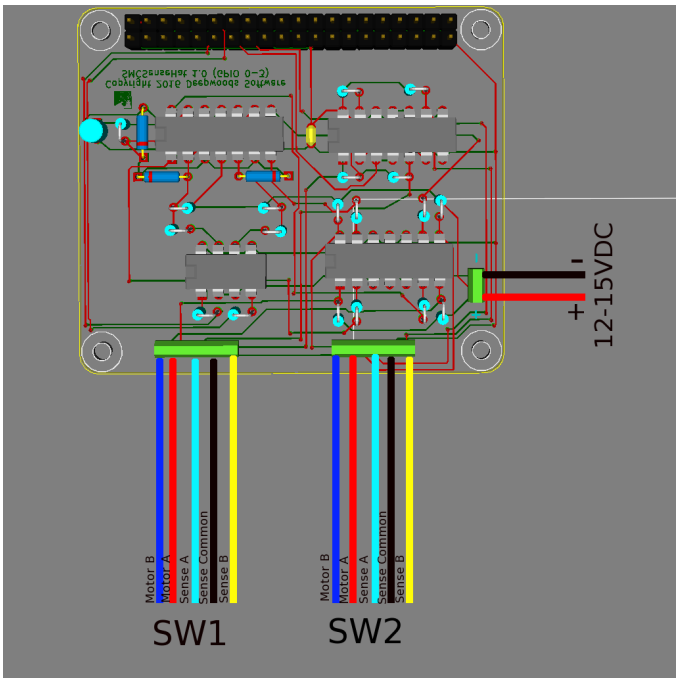

Figure 2.2 Connecting the SMCSenseHAT

The **Adafruit Perma-Proto HAT** is wired as shown below. The only components installed on this board are a pair of 6 position screw terminals. It is possible to use a board that simply has screw terminals for the GPIO pins (like the Adafruit Pi-EzConnect Terminal Block Breakout HAT) instead of wiring up a board like this. The PiGPIO XML file included with this example maps the GPIO pins like this:

- WPi 4 (BCM 23) to OS2
- WPi 5 (BCM 24) to Main One
- WPi 6 (BCM 25) to Crossing OS
- WPi 7 (BCM 4) to OS1
- WPi 21 (BCM 5) to Interchange
- WPi 22 (BCM 6) to Main Two
- WPi 23 (BCM 13) to Main One West
- WPi 26 (BCM 12) to Main Two South

This is how the GPIOs are wired:

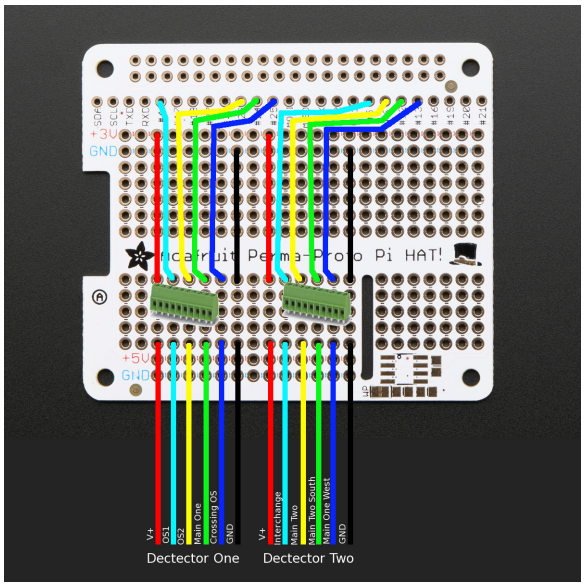

Figure 2.3 Wiring and connecting the Adafruit Perma-Proto HAT.

The QuadSignal XML files included with this example maps the signals like this:

- First board (i2c address 0):
  - S1S H1 and H2
  - S1MN H3
  - S1IN H4
- Second board (i2c address 1):
  - S2ME H1
  - S2W H2 and H3
  - S2IE H4
- Third board (i2c address 2):
  - S3W H1
  - S3N H2
  - S3E H3
  - S3S H4

This is how the **QuadSignalCA HATs** are wired (be sure to take note of the address jumpers):

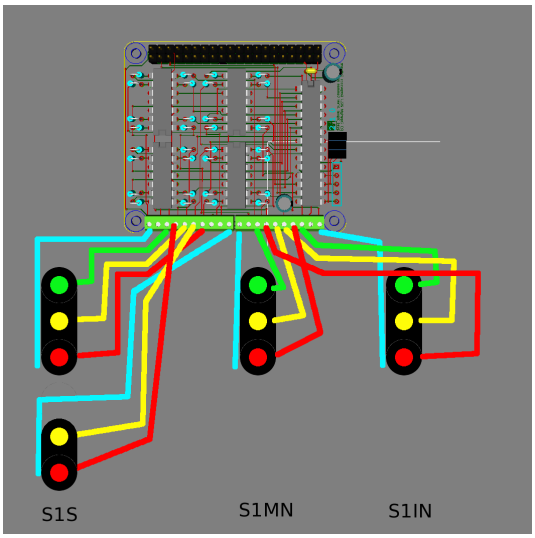

Figure 2.4 Connecting the signals at Control Point 1.

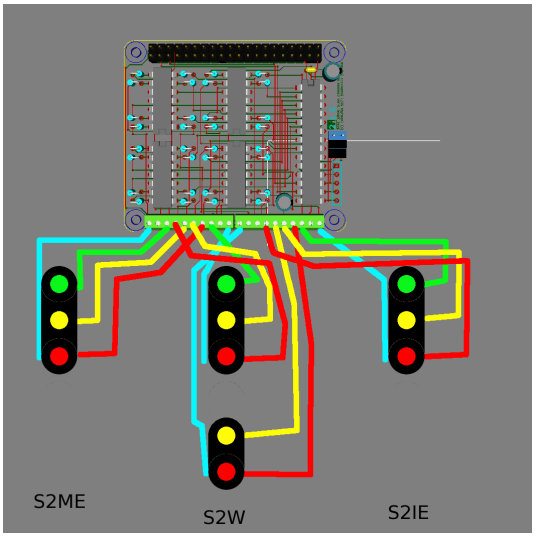

Figure 2.5 Connecting the signals at Control Point 2.

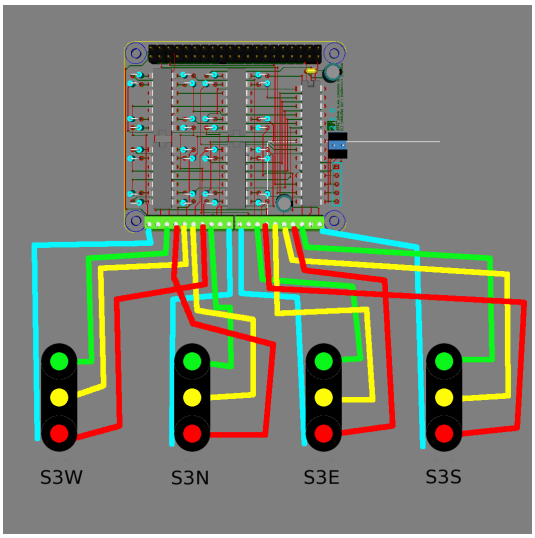

Figure 2.6 Connecting the signals at Control Point 3.

2.2 Software used.

This example uses these 3 OpenLCB daemons:

1. OpenLCB_QuadSignal (three instances, one for each QuadSignalDriverCA hat).
2. OpenLCB_PiGPIO (for both the SMCSenseHat and the detector inputs)
3. OpenLCB_Logic (to implement the signaling logic)

This example also uses a CTC panel created by the Dispatcher program.

Each deamon instance's configuration is taken from an XML file (included in this example's distribution directory) and the CTC panel exists as a Tcl/Tk source file (which is also included in this example's distribution directory).