Test document – Track Controller Software

This document will show the test cases passed for the Track Controller (Software). First, the code for testing is shown below: (you can find it on github under “trackControllerTests.py” in the tests folder in our project directory).

programmerUI.py | UI.py | TrackControllerSoftware.py M | trackControllerTest.py M | PLC_redline.txt | launch.py M

train_system > tests > trackControllerTest.py > TrainControllerTests > test_lights_redline

```python
import sys
import os
import unittest


sys.path.append(os.path.dirname(os.path.dirname(__file__)))

from TrackControllerSoftware.TrackControllerSoftware import Track_Controller
from signals.trackcontroller_signals import TrackCTCSignals
from signals.track_signals import TrackSignals

class TrainControllerTests(unittest.TestCase):

    # test that programmer can change the outputs manually
    def test_manual_outputs(self):
        tc = Track_Controller(ctc_signals=None, track_signals=None, test=True)
        tc.set_traffic_lights(2,4,0,1)  # set RED lights ON on block 5 on blue line
        tc.set_switch_positions(False,76,2,1)   # move switch at block 77 (green line) to LEFT
        tc.set_crossings(True,1,46,1)  # set crossings ON in red line (block 47)
        self.assertEqual(tc.get_traffic_lights()[2][0][4],True,"Traffic light on block 5, blue line isn't set to RED")
        self.assertEqual(tc.get_switch_positions()[2][76],False, "Switch at block 77 (green line) not set to LEFT")
        self.assertEqual(tc.get_crossings()[1][46],True,"Crossings not set to ON on red line")

    #test that plc computes the correct outputs under specific values for track occupancy on red line
    def test_switches_redline(self):
        tc = Track_Controller(ctc_signals=None, track_signals=None, test=True, testUI=False)
        # check switches
        tc.set_track_occupancy(True,15)  # train on block 1 on red line
        tc.voter_redline("train_system\TrackControllerSoftware\PLC_redline.txt")  #this also checks that PLC is uploaded
        self.assertEqual(tc.get_switch_positions()[1][15],False,"Switch doesn't switch to left when train is on block 1")

    def test_lights_redline(self):
        # check traffic lights
        tc = Track_Controller(ctc_signals=None, track_signals=None, test=True, testUI=False)
        tc.set_track_occupancy(True,31)   # train on block 17 on red line
        tc.voter_redline("train_system\TrackControllerSoftware\PLC_redline.txt")
        self.assertEqual(tc.get_traffic_lights()[0][1][0],False,"Lights are GREEN on block 1 when they should be RED because train is on block between 16 and 52 (red line)")
        self.assertEqual(tc.get_traffic_lights()[1][1][0],False,"Lights are YELLOW on block 1 when they should be RED because train is on block between 16 and 52 (red line)")
        self.assertEqual(tc.get_traffic_lights()[2][1][0],True,"Lights are not RED as they should be (red line)")

    def test_crossings_redline(self):
```

programmerUI.py | UI.py | TrackControllerSoftware.py M | trackControllerTest.py M | PLC_redline.txt | launch.py M

train_system > tests > trackControllerTest.py > TrainControllerTests > test_lights_redline

```python
    def test_crossings_redline(self):
        tc = Track_Controller(ctc_signals=None, track_signals=None, test=True, testUI=False)
        # check crossings
        tc.set_track_occupancy(True,60) # train on block 46 on red line, so crossins should be ON on block 47
        tc.voter_redline("train_system\TrackControllerSoftware\PLC_redline.txt")
        self.assertEqual(tc.get_crossings()[1][46],True,"Crossings should be ON, they are not")

    def test_faults_redline(self):
        tc = Track_Controller(ctc_signals=None, track_signals=None, test=True, testUI=False)
        # check if track fault detected and if train stops (commanded speed set to 0)
        tc.set_commanded_speed(43,32,1) # set initial commanded speed for train on block 33 to 43 mph
        tc.set_track_occupancy(True,47) # have a train on block 33
        tc.set_broken_rail(True,48)  # set broken rail on block 34
        self.assertEqual(tc.get_commanded_speed()[1][32],0,"Commanded Speed for train on block 33 isn't set to 0 when broken rail on block 34")


    #test that plc computes the correct outputs under specific values for track occupancy on green line
    def test_switches_lights_greenline(self):
        tc = Track_Controller(ctc_signals=None, track_signals=None, test=True, testUI=False)
        # check switches/lights
        tc.voter_greenline("train_system\TrackControllerSoftware\PLC_greenline.txt")
        self.assertEqual(tc.get_switch_positions()[2][84],False,"Default switch on block 85 is not set to LEFT") # check switch position before we add any train (default position) should be LEFT
        tc.set_track_occupancy(True,190)  # train on block 100 on green line
        tc._track_occupancy[167:176]=[False]*9  # blocks 77 to 85 are unoccupied
        tc.voter_greenline("train_system\TrackControllerSoftware\PLC_greenline.txt")
        self.assertEqual(tc.get_switch_positions()[2][84],True,"Switch didn't turn to right when train is on block 100")
        tc.set_track_occupancy(True,174) # train on block 84
        # so now since there is a train at 84 and a train on 100. The train on 100 should wait until train on 84 gets to 86 (gets in the loop)
        # before proceeding. So switch on 85 should still be set to left to let train on block 84 go through.
        # also, lights on block 100 should be red, and the ones on block 99 yellow. Which we will test here.
        tc.voter_greenline("train_system\TrackControllerSoftware\PLC_greenline.txt")
        self.assertEqual(tc.get_switch_positions()[2][84],False,"Switch didn't stay set to LEFT")
        self.assertEqual(tc.get_traffic_lights()[2][2][99],True,"Light on block 100 isn't RED")
        self.assertEqual(tc.get_traffic_lights()[1][2][98],True,"Light on block 99 isn't YELLOW")
```

```
75
76      def test_crossings_greenline(self):
77          tc = Track_Controller(ctc_signals=None, track_signals=None, test=True, testUI=False)
78          # check crossings
79          tc.set_track_occupancy(True,108)  # train on block 18, so crossings on 19 should turn ON
80          tc.voter_greenline("train_system\TrackControllerSoftware\PLC_greenline.txt")
81          self.assertEqual(tc.get_crossings()[2][18],True,"Crossings didn't turn ON on block 19 when train in on block 18")
82
83      def test_faults_greenline(self):
84          tc = Track_Controller(ctc_signals=None, track_signals=None, test=True, testUI=False)
85          # check if track fault detected and if train stops (commanded speed set to 0)
86          tc.set_commanded_speed(43,63,2) # set initial commanded speed for the train on block 64 as 43 mph
87          tc.set_broken_rail(True,155) # broken rail on block 65
88          tc.set_track_occupancy(True,154) # train on block 64
89          self.assertEqual(tc.get_commanded_speed()[2][63],0,"Commanded speed for train on block 64 isn't set to 0 when track fault on block 65")
90
91
92  if __name__ == "__main__":
93      unittest.main()
94
```

All test cases passed:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

ResourceWarning: Enable tracemalloc to get the object allocation traceback
.......
----------------------------------------------------------------------
Ran 8 tests in 0.035s

OK
PS C:\Users\RAYAN\OneDrive\Desktop\ECE1140>
```