Learning PLC Control Code from archived data

Abstract

PLCs are widely used for industrial automation. PLCs are complemented with historians, which has the job to archive data for monitoring and troubleshooting. Our aim is to learn PLC control logic from the archived data. The reverse engineering of deducing logic from data can help in many ways. Data can show hidden relationships among different signals that a programmer might not have utilized for programming. In another situation, if a hacker gets access to the historical archived data, he can launch targeted attacks without triggering any alarm in the system. Learning logic from archived data can also help in troubleshooting any problem during normal plant operation. A problem in normal operation of the plant can significantly change data trend that can be captured in the logic.

, .

1 Introduction

PLCs are backbone of industrial automation. They consist of CPUs, peripheral inputs and outputs. Sometimes there is a historian that archives signal trends on a real time basis. Since PLCs automate critical plant processes, their abnormal behaviour can impact the process significantly and even lead to catastrophic damage. The sensor status or PLC signal trends are captured by historian for the purpose of troubleshooting the problem during a breakdown in plant operation.

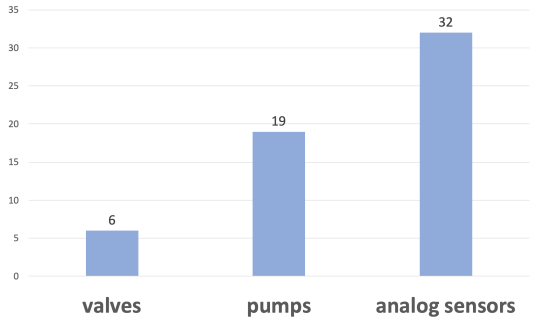

Figure 1: Types of Signals captured by SWAT

We analyzed SWAT dataset, which consists of signal history for 496800 seconds. There are 48 signals of all types such as actuators, transmitters and switches. We analysed the data to find the number of states of each signal. For instance the pump P-101 has two states (state: 0, which refers to OFF state of the pump, while state 1, which refers to the ON state of the pump). The valves show three states: state 0 for OFF state, state 1 for intermediate state (that is when the valve is trying to switch ON or OFF) and state 2 for ON state. To convert the three state values to two state system, we imputed the state 1 with the state in the next time step. For instance if the valve is transitioning from state 0(ON state) to state 2, we impute the state 1 with 2. This makes the valves a binary signal.

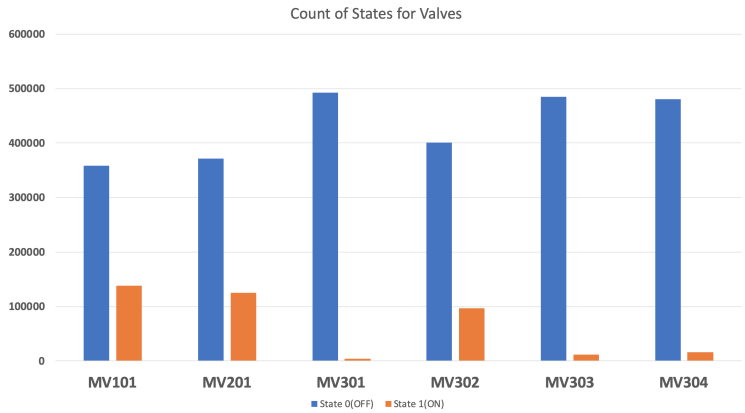

Figure 2: Frequency of States of different Valves

Figure 2 and 3 plot the frequency of states for each pump and valve. It is noticeable that some pumps, for instance P102, P201, P202, P204, P206, P401, P403, P404, P502, P601 and 'P603 exhibit no change in state. Thus, it is difficult to learn control logic of these pumps. Also in real world, some pumps might be a stand-by of another pump. We are yet to explore a method to figure out which pump is a stand-by pump just by analysing the historical trend. Similarly some valves are in same state for more than 96% of the time as depicted in figure 4. It can be challenging to utilize machine learning tools to model such biased data. Thus we utilized innovative methods to model these signals.

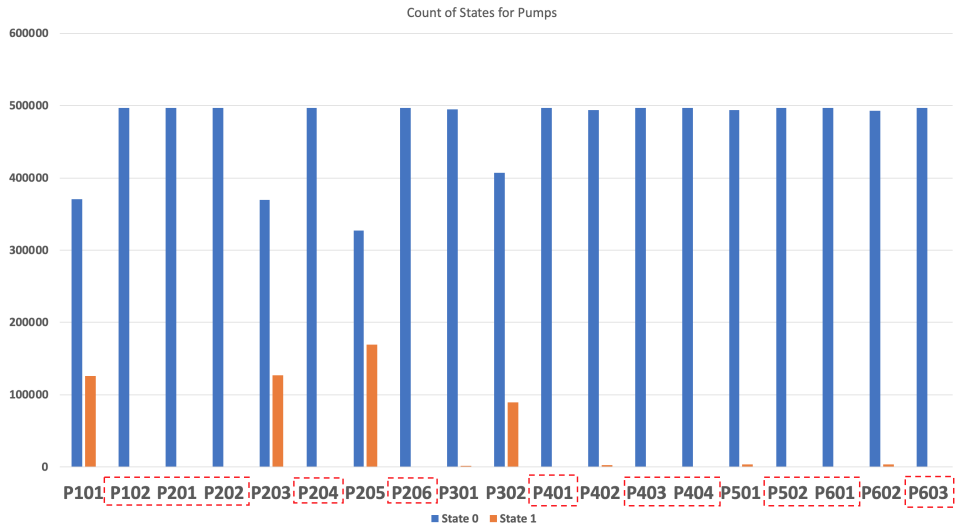

Figure 3: Frequency of States of different Pumps

Signals	State 0(OFF)	State 1(ON)	Percentage_time_OFF
MV101	358885	137915	72.23933172
MV201	371524	125276	74.78341385
MV301	492391	4409	99.11252013
MV302	400441	96359	80.60406602
MV303	484578	12222	97.53985507
MV304	481019	15781	96.82347021
P101	371040	125760	74.68599034
P102	496800	0	100
P201	496800	0	100
P202	496800	0	100
P203	370097	126703	74.49617552
P204	496800	0	100
P205	327682	169118	65.95853462
P206	496800	0	100
P301	494829	1971	99.60326087
P302	407510	89290	82.02697262
P401	496800	0	100
P402	493614	3186	99.35869565
P403	496800	0	100
P404	496800	0	100
P501	493462	3338	99.32809984
P502	496800	0	100
P601	496800	0	100
P602	492868	3932	99.20853462
P603	496800	0	100

Figure 4: Table depicting count of states and percentage time in OFF state for all Binary Signals

2 Statistical Approach

The relationship between different signals can be established by either statistical analysis or by learning a model from the data. We evaluated different approaches to detect the hidden relationship between signals. For statistical analysis, we computed correlation to measure similarity in binary signals. We computed Matthew's correlation between all binary signals. We got a clear relationship between signals shown in figure 5.

Sl.No	Signal 1	Signal 2	Correlation Coefficient
1	MV201	P101	0.997423516
2	MV201	P203	0.992398778
3	MV201	P205	0.808249785
4	MV301	P602	0.943901002
5	MV302	P302	0.922175989
6	P101	P203	0.992774605
7	P101	P205	0.808540181
8	P203	P205	0.814445158
9	P402	P501	0.976816175
10	P501	UV401	0.977587317

Figure 5: Signals which exhibit high correlation

We verified the high correlation between signals with the actual control logic to see whether there is true relationship. We found that correlation gave a clear picture of either direct relationship or secondary relationship. For instance in case of MV201 and P203 (figure 5 Sl.No2), there is a direct relationship as noted in the Control Logic Manual, figure 6. We tried to find similar relationship

between signals with high correlation coefficient. The figure 5 shows that Matthew's correlation was able to identify relationship between 6 signals (identified by yellow Sl.Nos). In 4 signal pairs, it showed high correlation but there was no PLC code for these pairs. But high correlation definitely indicate a secondary relationship. By primary relationship, we mean that there exists a PLC code which contribute to the observed relationship between the data. Whereas by secondary relationship, we mean that even though there is no documented PLC code which depicts relationship between both signals, the fairly high correlation indicates that both signals operate simultaneously because they are linked to a common signal.

Signals	MV101	MV201	MV301	MV302	MV303	MV304	P101	P102	P201	P202	P203	P204	P205	P206	P301	P302	P401	P402	P403	P404	P501	P502	P601	P602	P603	UV401
MV101	1	0.23987336	0.05703065	-0.1656302	0.09569381	0.00112544	0.23832319	0	0	0	0.2465251	0	0.18240099	0	0.03912405	-0.1955749	0	0.07030493	0	0	0.06995084	0	0	0.0538476	0	0.07017209
MV201	0.23987336	1	0.04970962	0.51677977	0.07250191	-0.0378372	0.99742352	0	0	0	0.99239878	0	0.80824978	0	0.0366485	0.49597245	0	0.07714925	0	0	0.07689713	0	0	0.04710572	0	0.07701601
MV301	0.05703065	0.04970962	1	-0.1929026	0.56854042	-0.0171396	0.049414	0	0	0	0.05014767	0	0.03215328	0	-0.0059722	-0.2021542	0	0.00760229	0	0	0.00778272	0	0	0.943901	0	0.00760829
MV302	-0.1656302	0.51677977	-0.1929026	1	-0.323752	-0.3582388	0.52048922	0	0	0	0.5147836	0	0.42496891	0	0.02747713	0.92217599	0	0.13303728	0	0	0.13276716	0	0	-0.182081	0	0.13299933
MV303	0.09569381	0.07250191	0.56854042	-0.323752	1	0.41601767	0.07231701	0	0	0	0.07327789	0	0.04786733	0	-0.0100232	-0.334575	0	0.01275905	0	0	0.01306188	0	0	0.56167565	0	0.01276913
MV304	0.00112544	-0.0378372	-0.0171396	-0.3582388	0.41601767	1	-0.0374359	0	0	0	-0.0365024	0	-0.0327463	0	-0.0033978	-0.0920593	0	0.00865653	0	0	0.0091368	0	0	-0.0161781	0	0.0086726
P101	0.23832319	0.99742352	0.049414	0.52048922	0.07231701	-0.0374359	1	0	0	0	0.9927746	0	0.80854018	0	0.03674317	0.49992157	0	0.09223926	0	0	0.09168819	0	0	0.04677558	0	0.09209409
P102	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P201	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P202	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P203	0.2465251	0.99239878	0.05014767	0.5147836	0.07327789	-0.0365024	0.9927746	0	0	0	1	0	0.81444516	0	0.03692763	0.49429613	0	0.09218318	0	0	0.09161324	0	0	0.04751851	0	0.09203768
P204	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P205	0.18240099	0.80824978	0.03215328	0.42496891	0.04786733	-0.0327463	0.80854018	0	0	0	0.81444516	0	1	0	-0.0546039	0.42254977	0	0.08335991	0	0	0.08624961	0	0	0.03036981	0	0.08336401
P206	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P301	0.03912405	0.0366485	-0.0059722	0.02747713	-0.0100232	-0.0033978	0.036743 17	0	0	0	0.03692763	0	-0.0546039	0	1	-0.134829	0	-0.2035215	0	0	-0.2292007	0	0	-0.0056371	0	-0.2045575
P302	-0.1955749	0.49597245	-0.2021542	0.92217599	-0.334575	-0.0920593	0.49992157	0	0	0	0.49429613	0	0.42254977	0	-0.134829	1	0	0.17163145	0	0	0.17570496	0	0	-0.1908135	0	0.17176694
P401	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P402	0.07030493	0.07714925	0.00760229	0.13303728	0.01275905	0.00865653	0.09223926	0	0	0	0.09218318	0	0.08335991	0	-0.2035215	0.17163145	0	1	0	0	0.97681618	0	0	0.0071758	0	0.99921118
P403	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P404	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P501	0.06995084	0.07689713	0.00778272	0.13276716	0.01306188	0.0091368	0.09168819	0	0	0	0.09161324	0	0.08624961	0	-0.2292007	0.17570496	0	0.97681618	0	0	1	0	0	0.0734612	0	0.97758732
P502	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P601	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P602	0.0538476	0.04710572	0.943901	-0.182081	0.56167565	-0.0161781	0.04677558	0	0	0	0.04751851	0	0.03036981	0	-0.0056371	-0.1908135	0	0.0071758	0	0	0.0074612	0	0	1	0	0.00718147
P603	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
UV401	0.07017209	0.07701601	0.00760829	0.13299933	0.01276913	0.0086726	0.09209409	0	0	0	0.09203768	0	0.08336401	0	-0.2045575	0.17176694	0	0.99921118	0	0	0.97758732	0	0	0.00718147	0	1

Figure 6: Types of Signals captured by SWAT

3 Machine Learning Approach

Our primary objective is to learn the rules that data obey. By learning rules we mean to learn the PLC codes which guide the sensor or actuator operation. Previously, we deployed statistical tools such as correlation coefficient to find out the rules or relationship. This could not give us any relationship between analog and digital signal. We thus ventured into Decision Trees. The objective of learning decision tree was to understand rules that govern the performance of analog and digital signal. We used sklearn package in python. The sklearn by default uses gini index to compute nodes in the decision tree. We compared the learned decision tree with the PLC code and found out

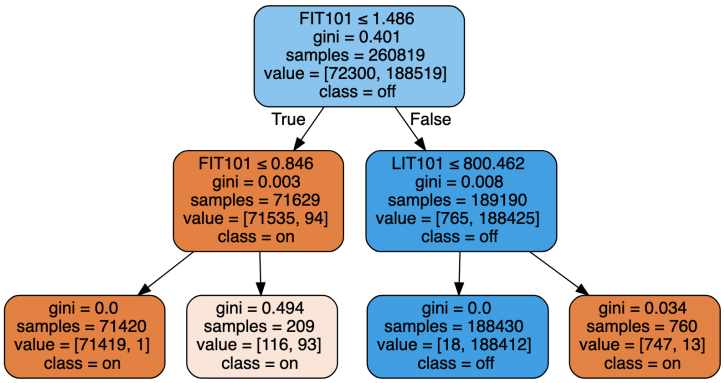

Figure 7: Decision Tree for MV101

some differences. The decision tree could not detect precise thresholds for splitting the nodes. For instance, the threshold selected for right node of decision tree shown in figure 8(LIT $\leq$ 999.924) only represents High setpoint. Thus the decision tree is not able to capture the low setpoint (LIT $\geq$ 800) which is shown in the control code, figure 9.

We even increased the depth of the learned decision tree, but the tree was unable to capture the low and high setpoints. To deal with this issue, we tried to learn decision tree by augmenting the features. We needed some method to detect the thresholds for low and high setpoints. We conceived of a method of detecting simultaneous transition.

We implemented our decision tree using the equations shown, which basically uses the concept of information gain and entropy.

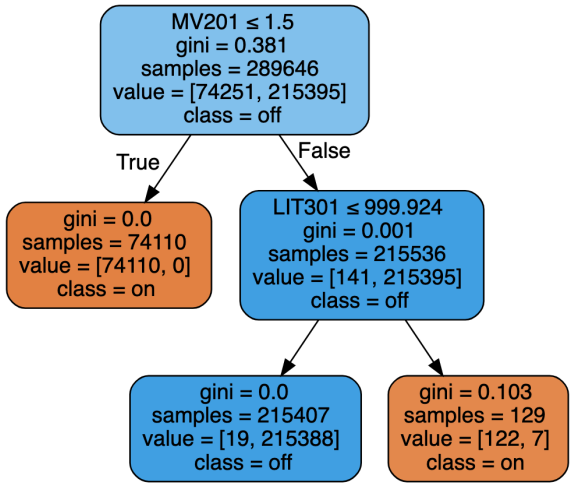

Figure 8: Decision Tree for P101 with depth 2

2) Raw water pump (P-101 & P-102) shall be interlocked with
   a. One duty, one standby
   b. To interlocked with UF feedwater tank level (LIT-201)
      i. Low Setpoint: 800mm ➔ Pump P-101/ P-102 START
      ii. High Setpoint: 1000mm ➔ Pump P-101/ P-102 STOP
   c. To interlocked with Feedwater flowmeter (FIT-201)
      i. Low Setpoint: 0.5m³/h ➔ Pump P-101/ P-102 STOP

Figure 9: PLC Control Code for P101

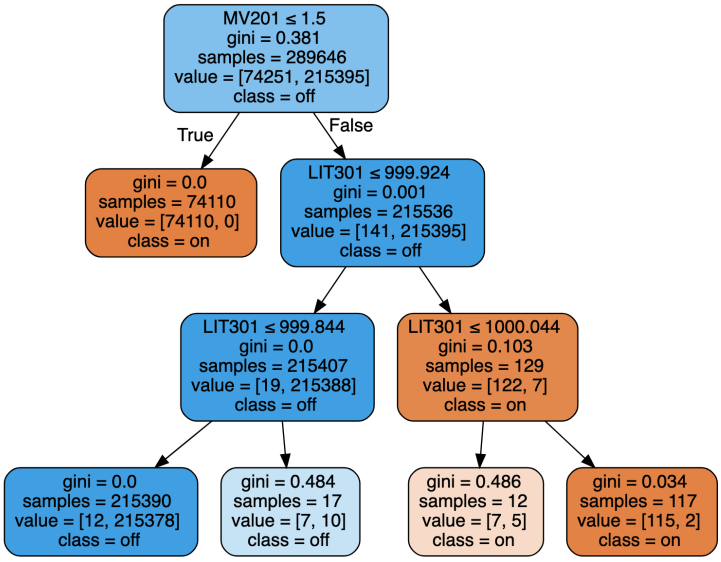

Figure 10: Decision Tree for P101 with depth 3

$$Entropy\, H(Y)\, of\, a\, random\, variable\, Y\, before\, split$$

$$H(Y) = -\sum_{i=1}^{m} P(Y = y_i) log_2 P(Y = y_i) \tag{1}$$

$$Entropy\, of\, Y\, after\, split$$

$$H(Y \mid X) = -\sum_{j=1}^{k} P(X = x_j) \sum_{i=1}^{m} P(Y = y_i | X = x_j) log_2 P(Y = y_i | X = x_j) \tag{2}$$

$$Information\, Gain\, or\, Mutual\, Information\, of\, X\, and\, Y$$

$$I(Y, X) = H(Y) - H(Y \mid X) \tag{3}$$

After decision trees, we employed Random Forest to learn the relationship. In our case Random Forest also suffered from the same deficiency. It could not take into account the sequence of operation and time dependency of data.

```
FEATURE IMPORTANCE:  [9.90757754e-01 1.57589677e-07 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
6.02998498e-03 0.00000000e+00 0.00000000e+00 3.21210345e-03
0.00000000e+00]

TREE: 0
0 NODE: if feature[11] < 1.5 then next=1 else next=4
1 NODE: if feature[0] < 542.4927368164062 then next=2 else next=3
2 LEAF: return class=1
3 LEAF: return class=0
4 NODE: if feature[1] < 1.5 then next=5 else next=6
5 LEAF: return class=1
6 LEAF: return class=1
TREE: 1
0 NODE: if feature[11] < 1.5 then next=1 else next=4
1 NODE: if feature[0] < 543.0226440429688 then next=2 else next=3
2 LEAF: return class=1
3 LEAF: return class=0
4 NODE: if feature[0] < 530.40283203125 then next=5 else next=6
5 LEAF: return class=1
6 LEAF: return class=1
TREE: 2
0 NODE: if feature[8] < 1.5 then next=1 else next=4
1 NODE: if feature[0] < 250.25595092773438 then next=2 else next=3
2 LEAF: return class=0
3 LEAF: return class=1
4 NODE: if feature[0] < 543.3170471191406 then next=5 else next=6
5 LEAF: return class=1
6 LEAF: return class=0
```

Figure 11: Random Forest Rules for P101 with number of estimators = 3, and max. depth = 2

4 Learning Thresholds from Transitions

The Matthew's correlation cannot detect relationship between signals when there is a time delay in the response of one signal. For instance suppose the pump P101 starts after 10 minutes when the valve MV101 starts. Although some paper discussed using Pearson correlation for time series data, we calculated transitions (change in state of the signal) to find out these timed relationships. The logic behind calculating simultaneous transitions is that if two signals show relationship then in a fixed time period both would show similar transition. While learning the decision tree, we witnessed that it could not identify the low and high setpoints. Thus we superimpose the digital signal over the analog signal to find out the crossover points. If the analog and digital signal have relationship the crossover point will not vary over time.

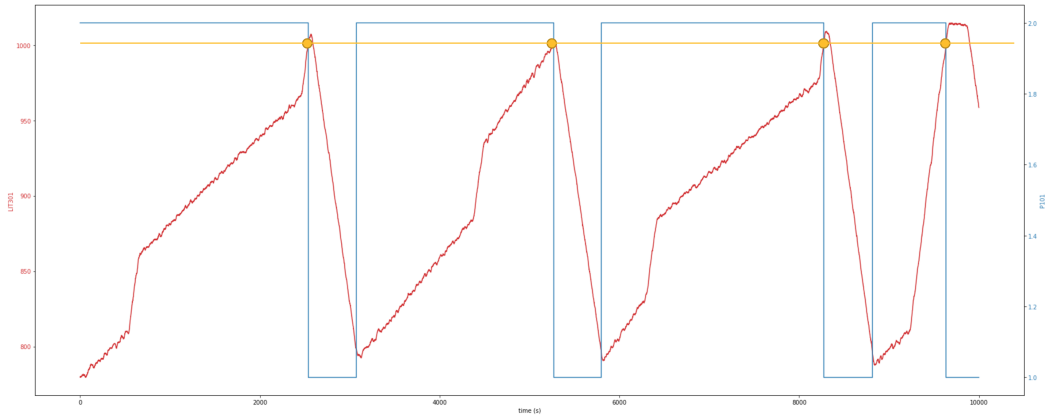

Figure 12: Superimposing LIT101 and P101. The yellow dots indicate similar crossover that is time when P101 switches from ON to OFF state

In the figure 11, we can see that when P101 was superimposed over LIT101, the value of LIT101 when P101 changed its state from ON state to OFF state did not change over time (All yellow dots lie on the same line).

Algorithm 1 Crossover point from Transition

Result: low and high setpoints
previous state=0
lowstate=0
highstate=1
lowstate value=[]
highstate value=[]
while *rows in dataframe* **do**
  if *current state value of digital signal ==lowstate AND previous state value of digital signal==highstate* **then**
    highstate value = value of analog signal
  end
  if *current state value of digital signal ==highstate AND previous state value of digital signal==lowstate* **then**
    lowstate value = value of analog signal
  end
  previous state value of digital signal=current state value of digital signal
end

```
LIT301  On CrossOver:  790.5376653225806  Percent_Similar_Crossover:  98.3739837398374 %
LIT301  Off CrossOver:  1000.2330741935482  Percent_Similar_Crossover:  98.3739837398374 %
LIT101  On CrossOver:  805.5147056451613  Percent_Similar_Crossover:  95.9349593495935 %
LIT101  Off CrossOver:  531.8045935483871  Percent_Similar_Crossover:  83.73983739837398 %
FIT201  On CrossOver:  0.0  Percent_Similar_Crossover:  98.3739837398374 %
FIT201  Off CrossOver:  2.4463587177419353  Percent_Similar_Crossover:  98.3739837398374 %
```

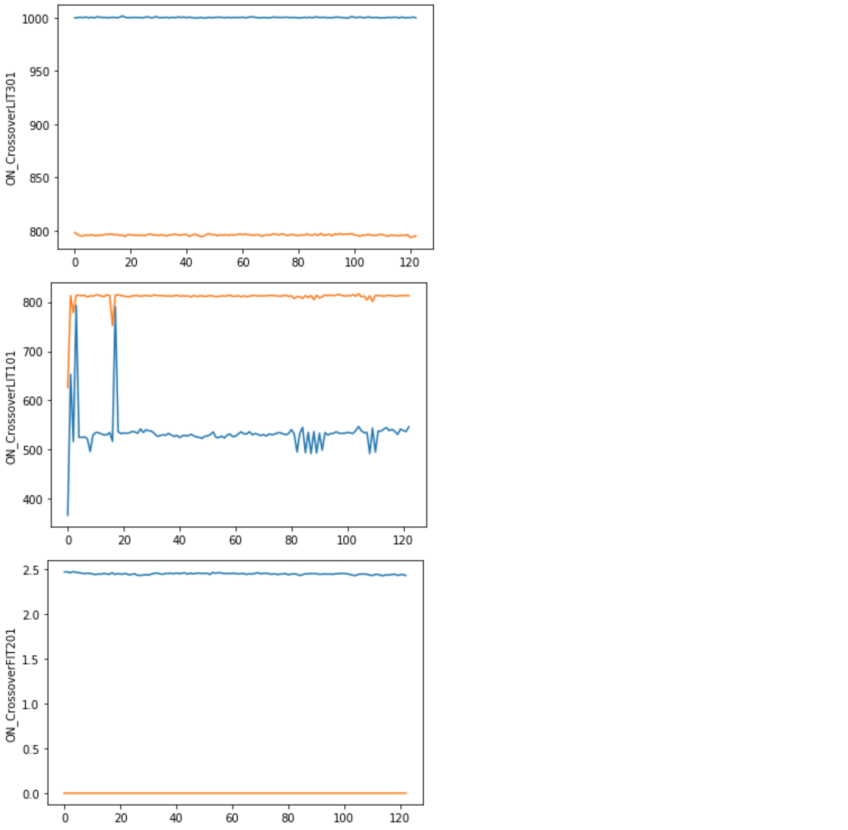

Figure 13: Low and High Setpoint trend for P101 and LIT301, LIT101, FIT201.

Figure 13 clearly shows that we could get precise crossover points (Low and High setpoints) by mapping values of FIT201 and LIT301 during state transition of P101.