1 ASSUMPTIONS

We made several assumptions to model the project.

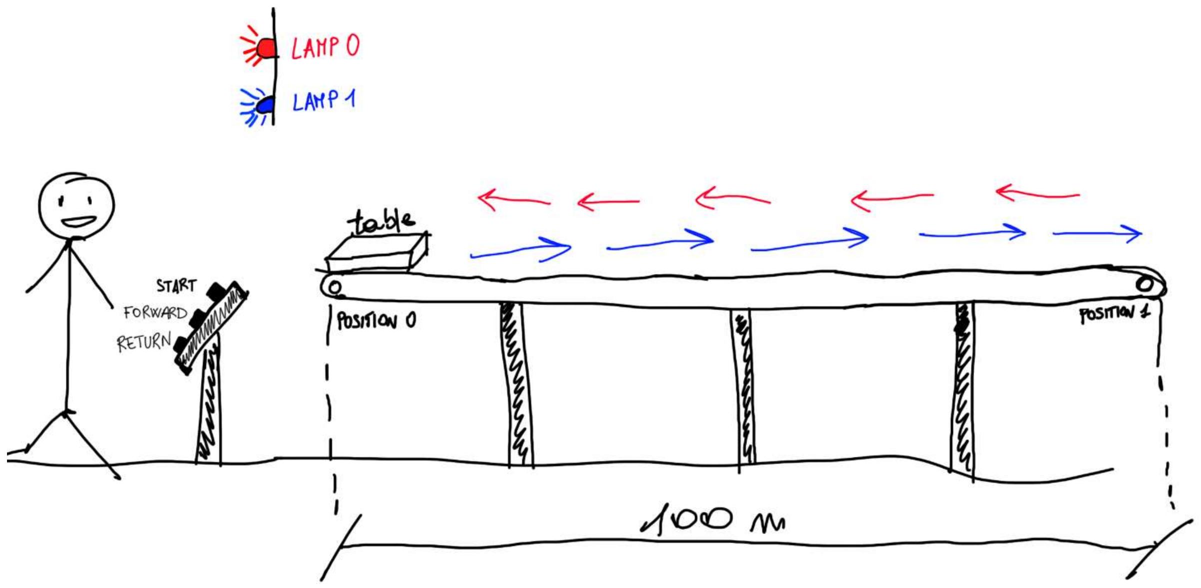
First of all we assume having two buttons, which can be pressed by a human operator: reverse button and forward button. We also assumed that the table, while its moving, the path cannot be modified when someone presses a button. For Example: if the table is moving from position 0 to position 1 and if someone press reverse, the table will keep moving towards position 1, so no change of action takes place.

We assume that a human is in front of the dashboard, located in position 0, the Dashboard consists of:

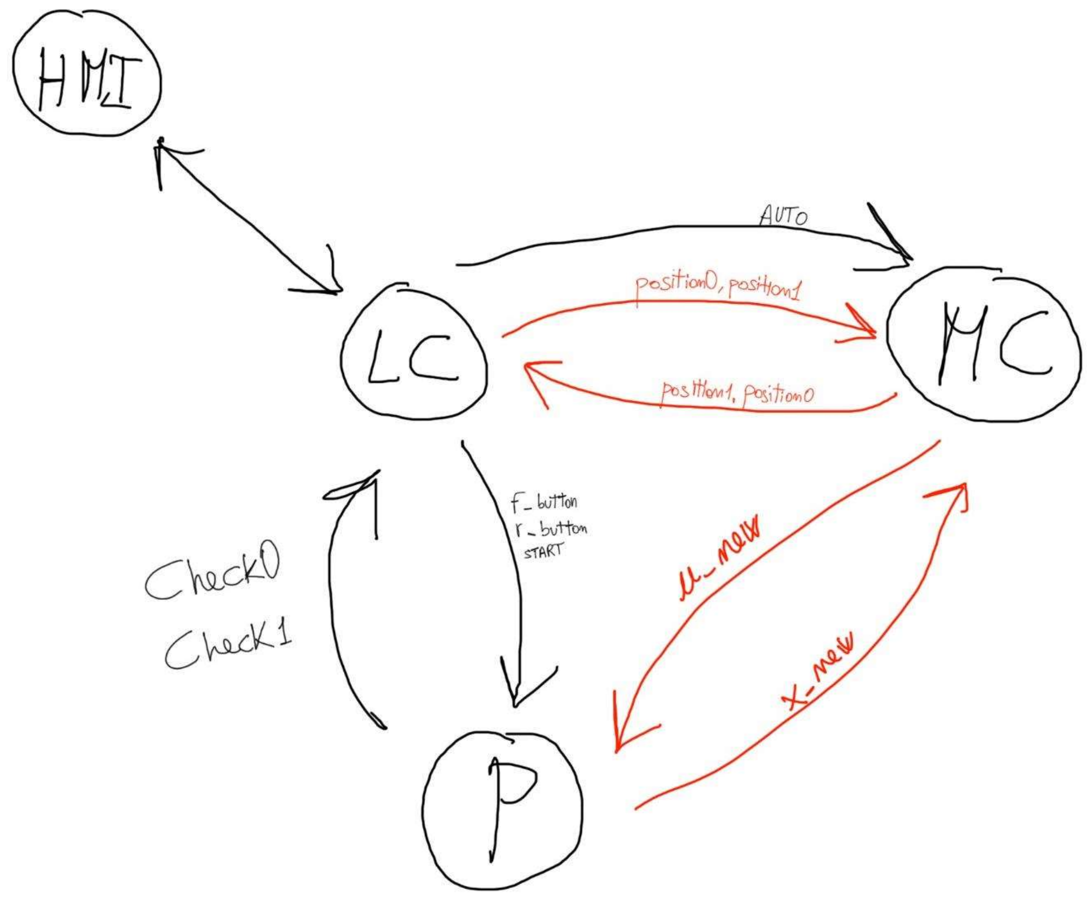
- Physical Start button
- Physical Return (Reverse) button, once pressed the table will move from position 1 to position 0;
- Physical Forward button, once pressed the second one the table will move from prosition 0 to position 1;
- a LED called “Lamp0”, which tell us that the table is still in position 0;
- a LED called “Lamp1”, which tell us that the table is still in position 1;

We also assume that the path is 100 meters long. Also the control signal is limited between -1 and 1.

In this figure we draw the above assumptions:



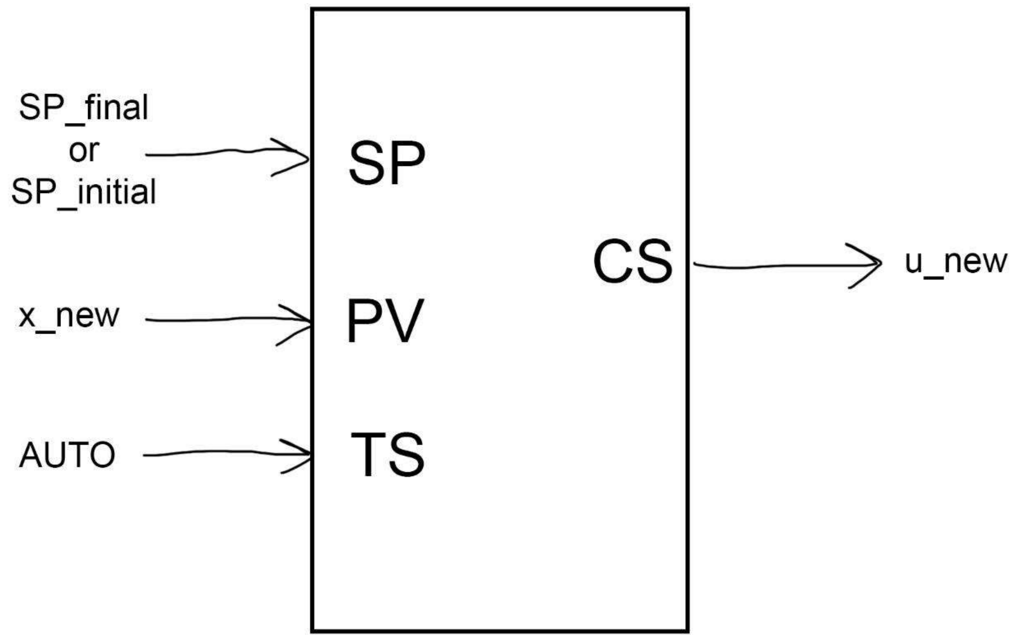
2 SUBSYSTEMS AND SIGNALS

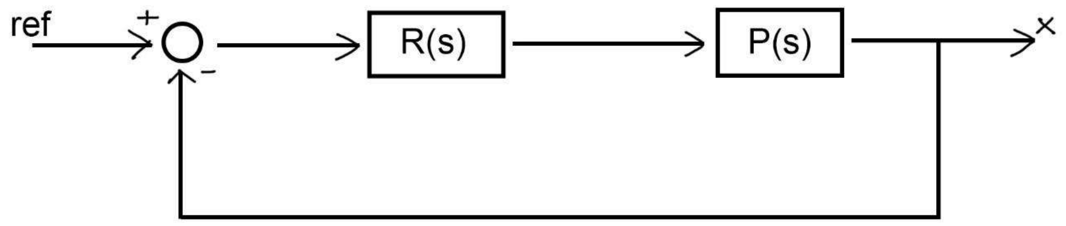


Position1 is SP_final

Position0 is SP_initial

3 MODULATION CONTROL

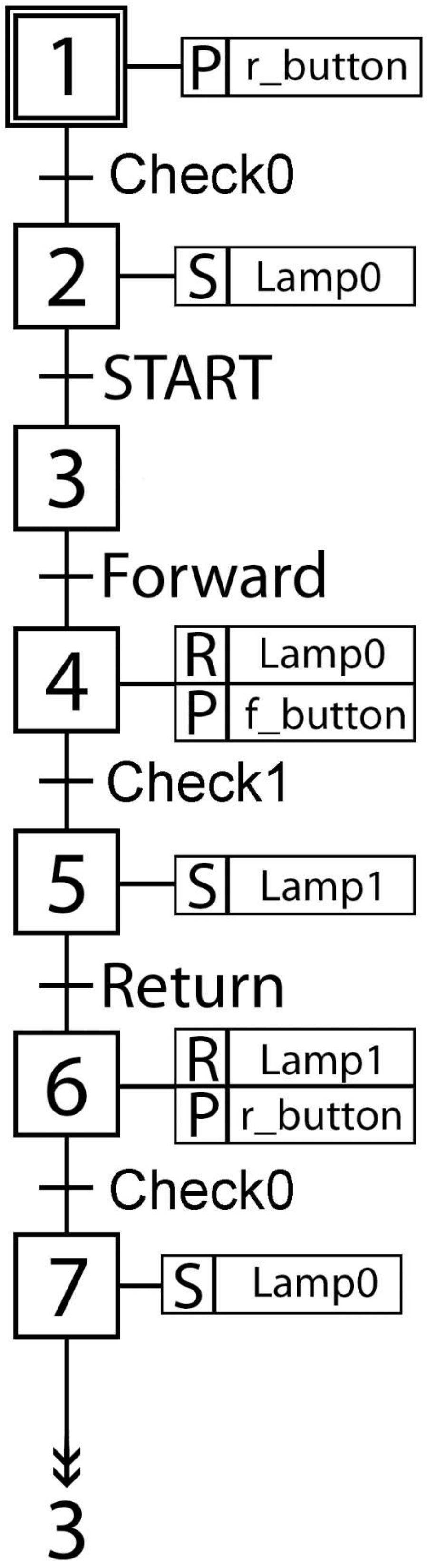




Ref: SP_final or SP_inital depending on forward or reverse pressed.

In the Chapter 5 we'll show R(s).

4 LOGIC CONTROL



Corresponding to each row you can understand the phase and transition description.

PHASE 1: Wherever the table is, the pulse r_button is sent to bring table at initial position.

TRANSITION: When the table reaches position 0, check0 is verified (with a certain tolerance) and we go to phase 2.

PHASE 2: Lamp0 is ON.

TRANSITION: Check if human press the physical START button.

PHASE 3: Do nothing.

TRANSITION: Check if the human has pressed physical forward button.

PHASE 4: Lamp0 is switched off and send a pulse signal f_button.

TRANSITION: When the table reaches position 1, check1 is verified (with a certain tolerance) and we go to phase 5

PHASE 5: Lamp1 is switched on.

TRANSITION: Check if the human has pressed physical return (reverse) button.

PHASE 6: Lamp1 is switched off and send a pulse signal r_button.

TRANSITION: When the table reaches position 0, check0 is verified (with a certain tolerance) and we go to phase 7.

PHASE 7: Lamp0 is switched on.

Go back to PHASE 3.

5 TUNING CONTROLLER

We have to handle the following transfer function:

$$P(s) = \frac{10}{s\,(1 + 0.2s)}$$

First of all, we tried to tune a PI controller. But in this way we didn’t reach a good solution: in any possible gain values; with a reasonable settling time, there were a lot of oscillation with a too big overshoot.

We decided to use a PD controller with a filter. In this way we have reached a good result, as we’ll show later.

Our controller is of the form as given below:

$$C_{PD}(s) = K_P\left(1 + \frac{T_D\, s}{\tau_D\, s + 1}\right)$$

We can simplify it into the the form as given below:

$$R(s) = K\,\frac{s+z}{s+p}$$

Specifications:

- 4% overshoot;
- 1 second settling time when the error has reached 1%.

The formulas are applied

$$\xi = \sqrt{\frac{\ln(0.04)}{\pi^2 + (\ln(0.04))^2}} = 0.72$$

$$T_s = \frac{4.6}{\xi\,\omega_n} \;\Rightarrow\; \omega_n = \frac{4.6}{1 * 0.72} = 6.39$$

Desired Poles:

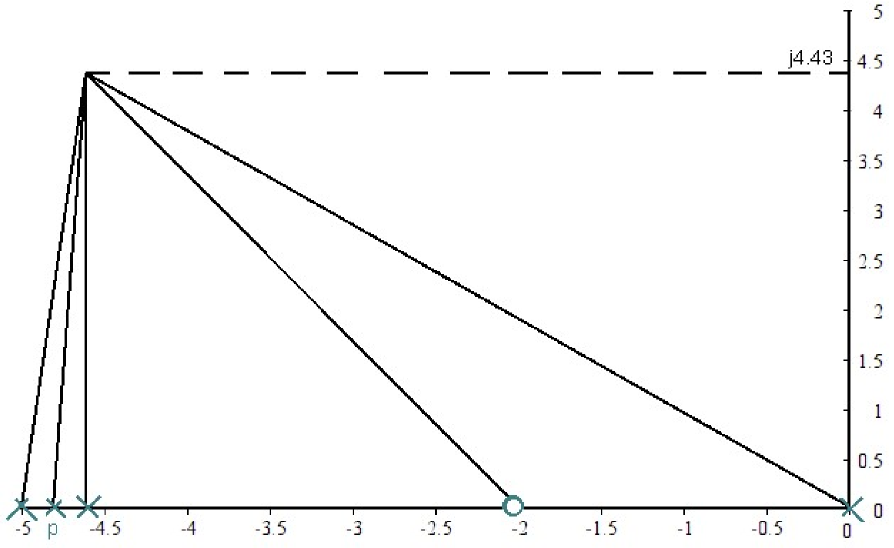
$$-\xi\,\omega_n \pm j\,\omega_n\sqrt{1-\xi^2} = -4.6 \pm j\,4.43$$

Denominator of our closed loop system with compensator is:

$$1 + \frac{K\,(s+z)}{s\,(s+p)\,(1+0.2s)}$$

$$\angle\frac{s+z}{s(s+p)(1+0.2s)} = \pm 180°\,(2k+1) \qquad where\ k = 0,1,2,\ldots$$

Assuming zero is at -2:



$$[180 - \tan^{-1}(\frac{4.43}{2.6})] - \theta_p - \tan^{-1}\left(\frac{4.43}{0.4}\right) - [180 - \tan^{-1}(\frac{4.43}{4.6})] = -180$$

Solving for θ_p:

$$\theta_p = 78.12°$$

$$\tan^{-1}\left(\frac{4.43}{p-4.6}\right) = 78.12°$$

$$p = 5.531$$

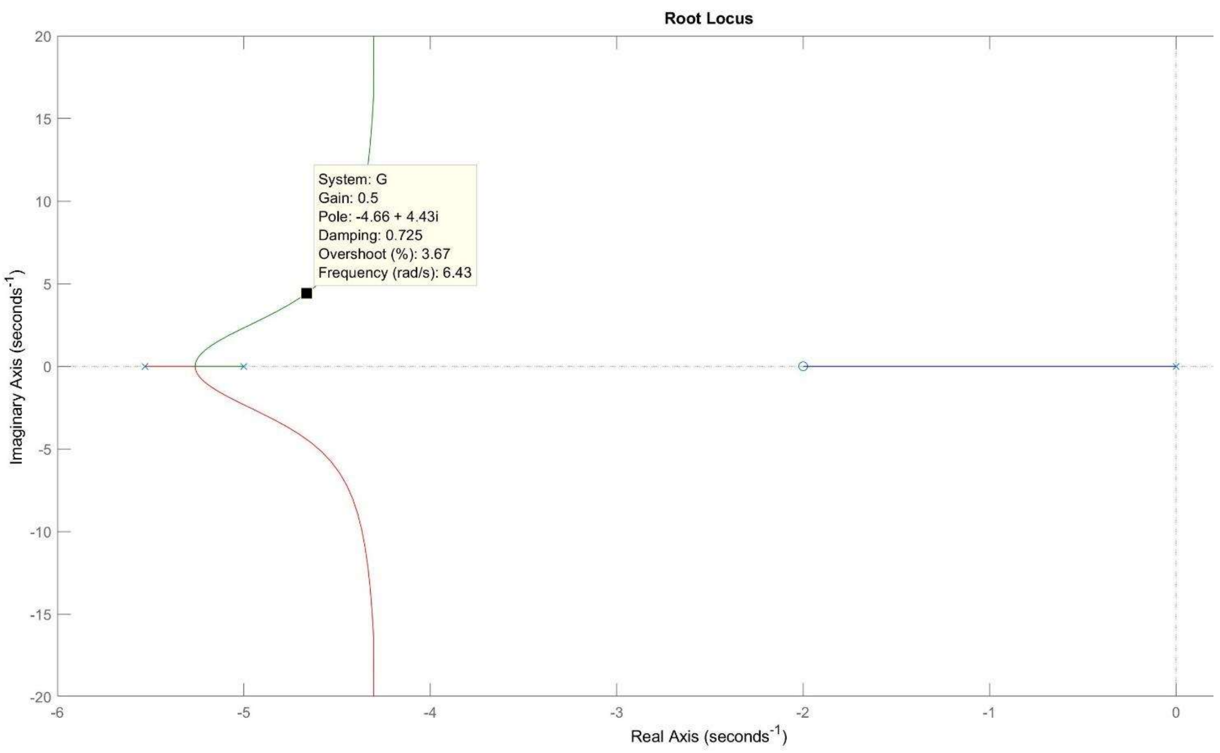
$$z = 2$$

Now, we can write:

$$R(s) = K\,\frac{s+2}{s+5.531}$$

Using Matlab rootlocus command we found the proper K:

$$K = 0.5$$



System: G
Gain: 0.5
Pole: -4.66 + 4.43i
Damping: 0.725
Overshoot (%): 3.67
Frequency (rad/s): 6.43

6 SIMULATION AND RESULTS

Putting the results obtained above in SciLab and using our code we got some results as we can see from the plots below.

From Figure 1, we can see that the controller is tuned properly and we have a very small overshoot, as we desired, and the settling time is also good. The overshoot is less than 4%, it satisfies our requirment.

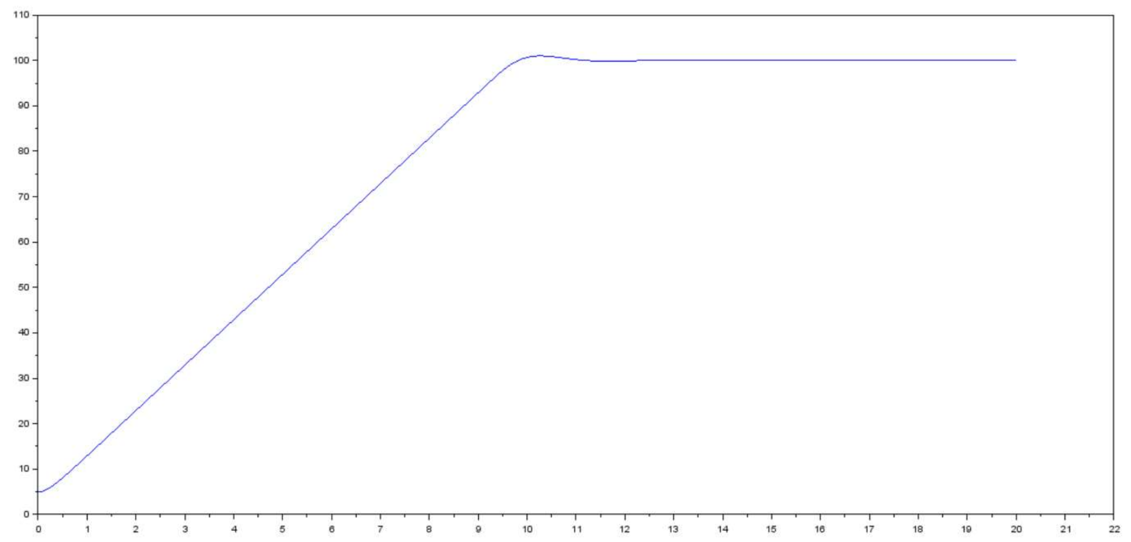


Figure 1. Setpoint=100 and initial position not at 0. Plot of only plant dynamics with compensator.

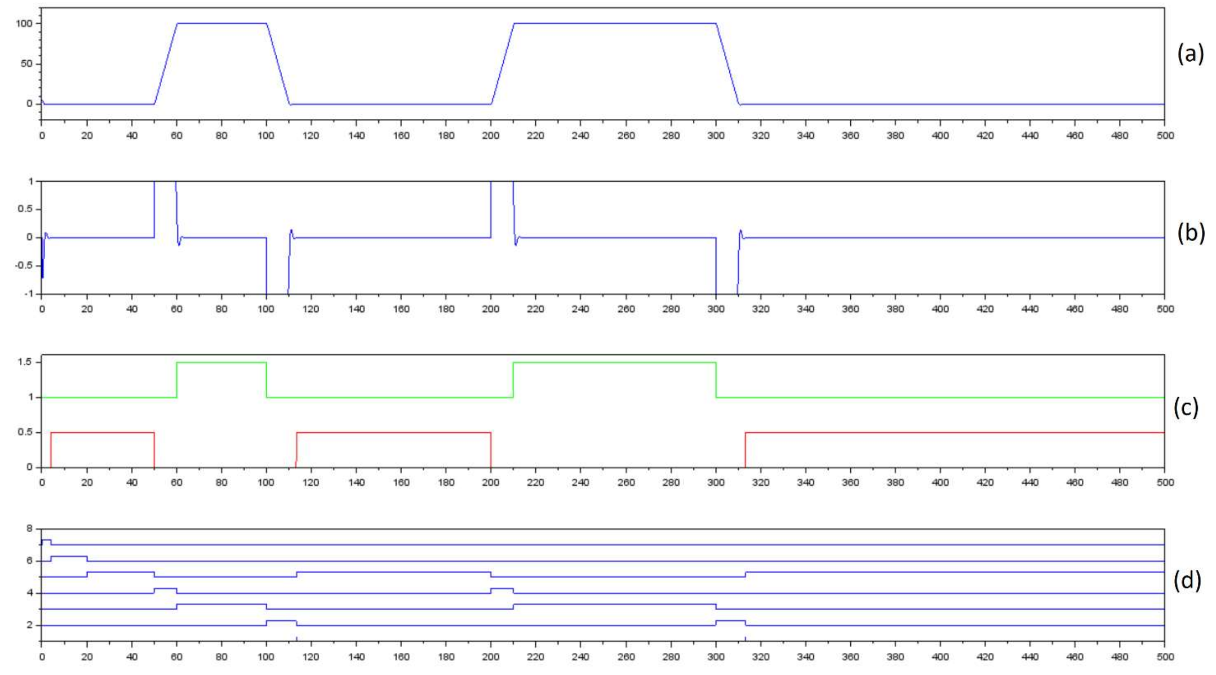


Figure 2.

Figure 2(a) represents the evolution of position x with respect to time.

Figure 2(b) represents the evolution of control signal u with respect to time.

Figure 2(c) represents the evolution of lamp0 (red) and lamp1 (green) with respet to time.

Figure 2(d) represents the evoltuion of each phase of SFC with respect to time.

Looking at figure 2, initially the table is assumed to be at x=5. Then pulse r_button is sent and the table will go back to position 0 (Lamp0 is ON). The person presses START physical button at t=20s. Now at t=50s the FORWARD physical button is pressed and the table starts to go to position 1 (Lamp0 is OFF). Once the table reaches the position 1 at about t=60s, then Lamp1 is ON. Then the human presses RETURN physical button and the table starts to move to position0 (Lamp1 is OFF). Once the table reaches position0 the Lamp0 is ON. You can see pulses in figure 2 (c), that indicates the duration the table has spent in either position 0 or position 1. Similarly the human presses FORWARD button at t=200s, and RETURN at t=300s, and we get similar desired response.

7 CONCLUSIONS

The graphs in the above figures satisfies our requirements, this proves that our controller is tuned properly. Futhermore, in our code in order to simulate the plant dynamics $\dot{x} = f(x, u)$ we used an explicit method. We understood the importance of simulation which we would have taken for granted if we used something like Simulink.

It has been really important to use SFC in order to make any phase of the process clearer with respect to any other approach. For example, if we would have used a naive approach of only “if,else” statements, our solution would be extremely complicated and we would make mistakes easily. SFC makes sure the signals are being used, we are proceeding in a sequential manner and verifying at any point is easy. Futhermore, the controller gains can be easily put into an embedded controller to work with in real life.