

Ciocîrtău Cosmin

Master 2023

Water Dispenser

Requirements

1. The water dispenser can output cold water or hot water. The hot water is heated on the spot (somehow).
2. The Simulink model has the following inputs and outputs:

Inputs:

* Water button (boolean)
* HotWater button (boolean)
* SelfTest button (boolean)
* Pour Value (number, 2 to 3) (Additional)
* Water level sensor (number, 0 to 1000 ml)
* Water temperature sensor (number, 0 to 100 degrees Celsius)

Outputs:

* + Activate Water Heater (boolean)
  + Activate Water Pouring (boolean)
  + Machine Status (integer):
* 0 = IDLE
* 1 = WORKING
* 2 = NO\_WATER
* 3 = HEATER\_FAULT
* 4 = POURING\_FAULT
* 5 = TESTING... (Additional)

1. The process is as follows:

• When pouring normal water:

– Start pouring water when Water=TRUE (i.e. user presses button)

– Stop when Water=FALSE

• When pouring hot water:

– When HowWater=TRUE (i.e. user presses button), activate the water heater

and wait for 500 milliseconds. Don’t pour any water yet.

– Only afterwards start pouring water

– Stop when HotWater=FALSE

1. All buttons must be debounced both ways, with a time duration of 0.2 seconds.
2. There is a separate self-test mode, activated via the SelfTest button. The procedure

is as follows:

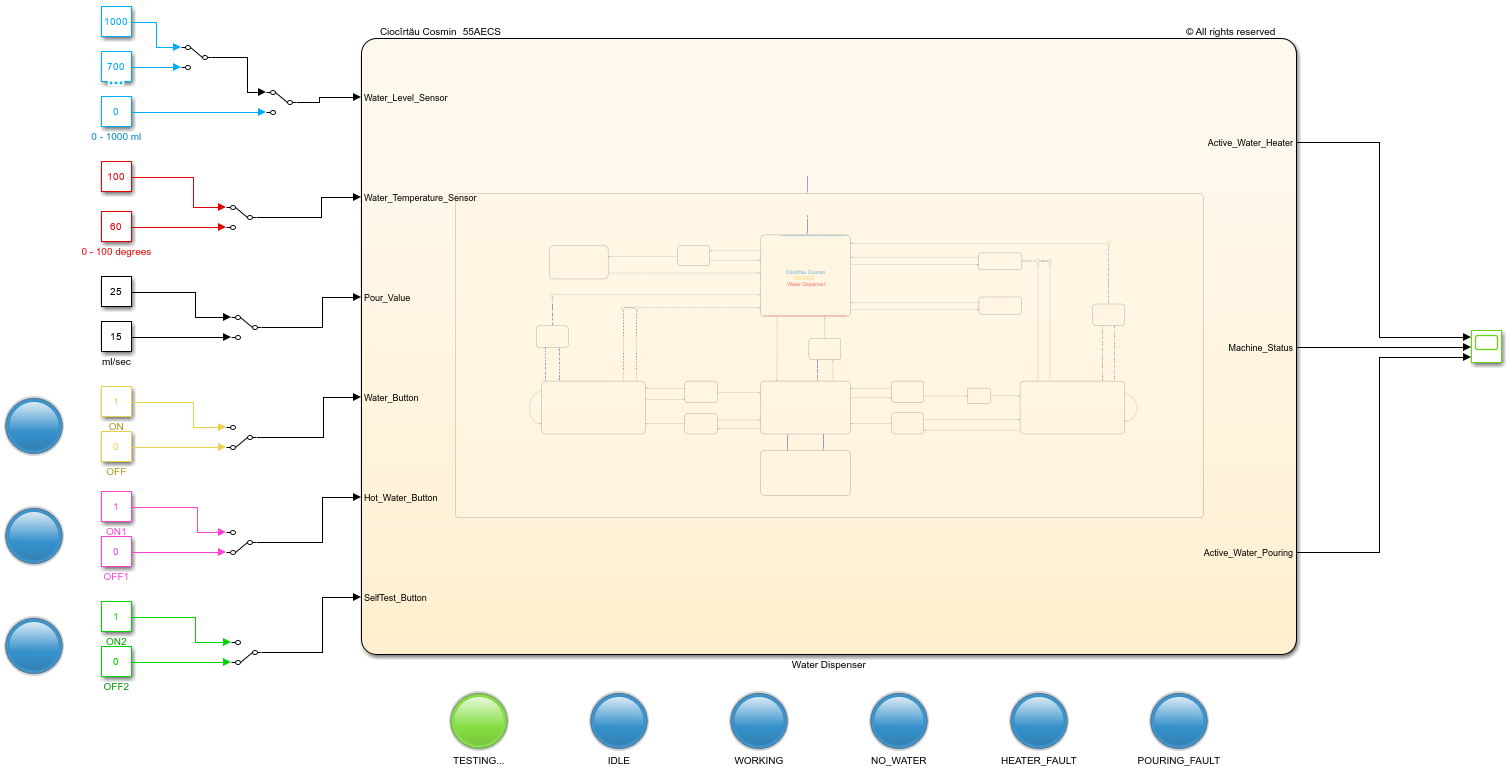
* + Start heating water. If the temperature doesn’t reach 99 degrees in 20 seconds, there is a heater error. The error must be signalled by setting Status = HEATER FAULT for at least 10 seconds.
  + Start pouring water. If the water level doesn’t drop by 50ml in 2 seconds, the pouring mechanism is blocked (i.e. limestone). The error must be signalled by setting Status = POURING FAULT for at least 10 seconds.

1. Use parameters from Matlab for all values you consider necessary (e.g. duration of

times etc.). Our customer may want to adjust the parameters at any time.

1. Test your state machine (use one/multiple separate test models if necessary).
2. Additional, I added from me requirements for WORKING and NO\_WATER modes. I implemented an IDLE block which means an ultra power saving mode. I implemented LEDs to see in real-time the status of the machine (Working, Fault, Testing). I implemented a system which counting how much water is pouring (tells us when we have no water remaining in the container of the machine).

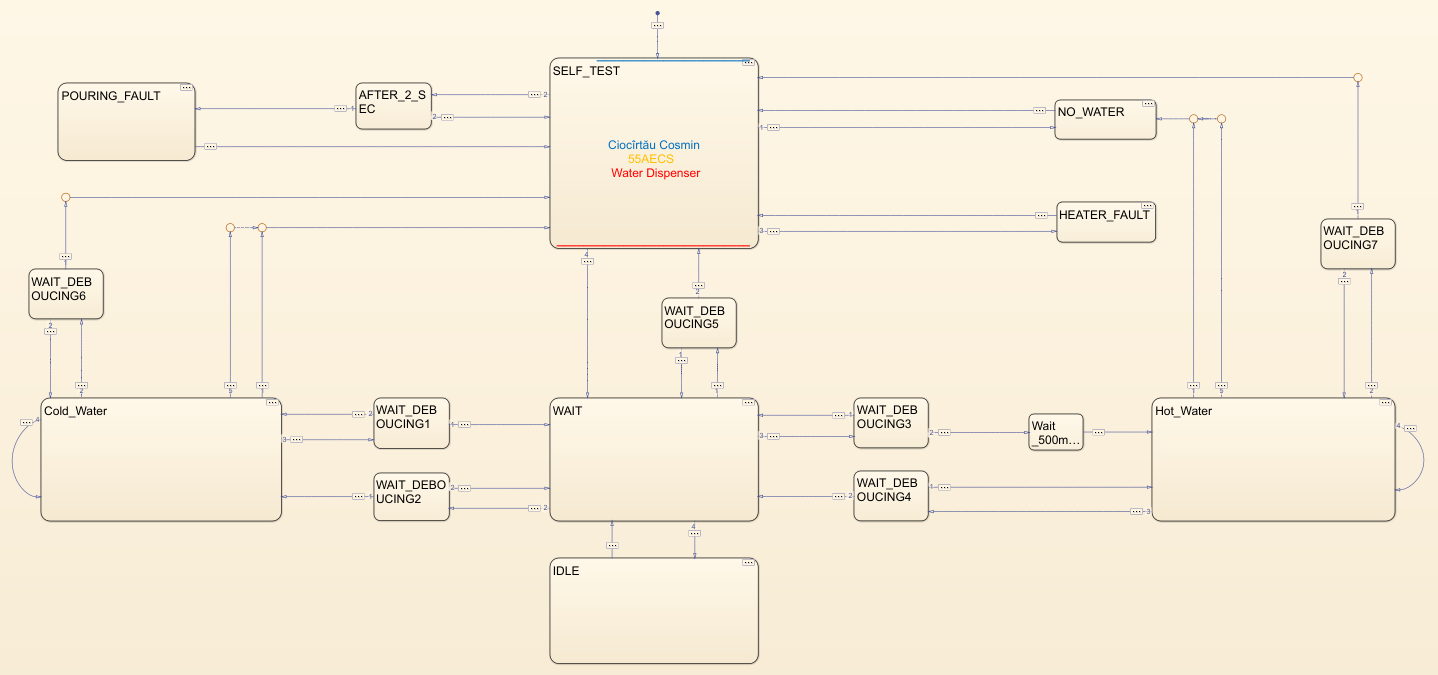
Main design



Description:

* We have 6 inputs and for any of them I used switches which allow me to change the value for any input very easy. For boolean inputs I used 1 or 0. For integer I used different values (e.g. at Water\_Level\_Sensor I can have any value between 0 and 1000).
* We have 3 outputs which will be viewed by scope. Active Water Heater will be a boolean signal (0/1), machine status will be viewed like a digital signal that grows and decreases in steps because we can have only 6 integer values (0, 1, 2, 3, 4, 5).
* The LEDs from the left side are used only for buttons from input. They tell us if the button from the right part is ON (1) or OFF (0). The LEDs from bottom side tell us the status of the machine, which can be Testing, Idle, Working, No\_Water, Heater\_Fault and Pouring\_Fault. Blue color means that the LED is OFF.

Finite State Machine



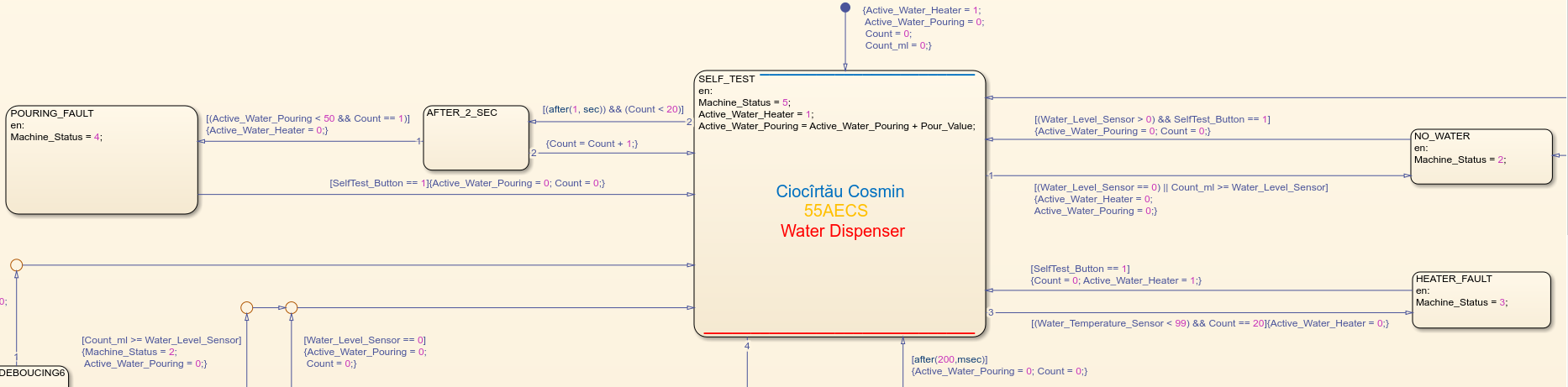
Our Finite State Machine is built from 5 main blocks:

1. Self\_Test;
2. Wait;
3. Cold\_Water;
4. Hot\_Water;
5. Idle.

Description:

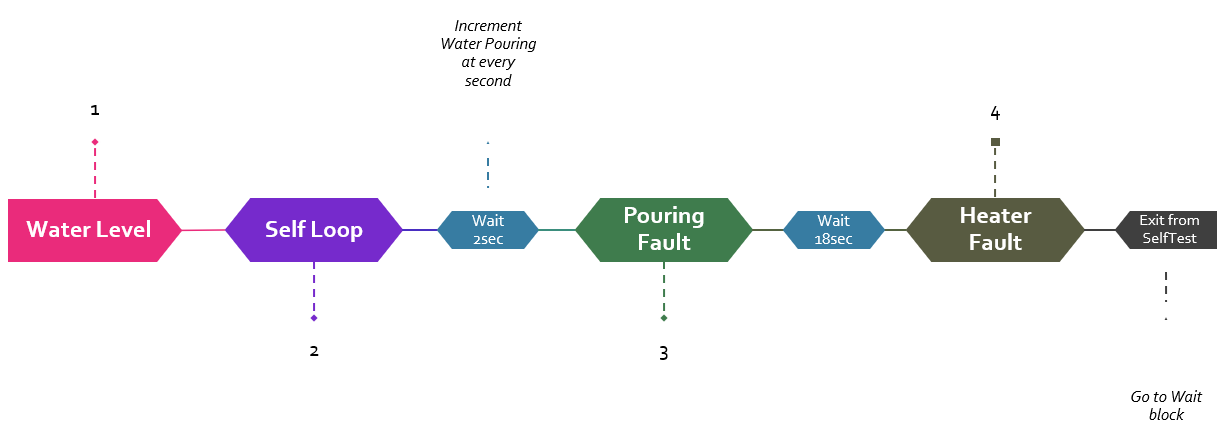
* SelfTest is highest priority block because it can be accessed from any state you are. It represent the block which verify all the functionality of the machine (e.g. Water level, Heater functionality).
* Wait is the block where the machine waits for next command from user; this can be a requirement for cold water or for hot water.
* Cold\_Water is the block which allows the water to pouring while we have water enough.
* Hot\_Water is a block as cold water but this time the machine pouring hot water.
* Idle is an ultra power saving energy mode which (in real life) can stop all LEDs to save more energy.

How it works



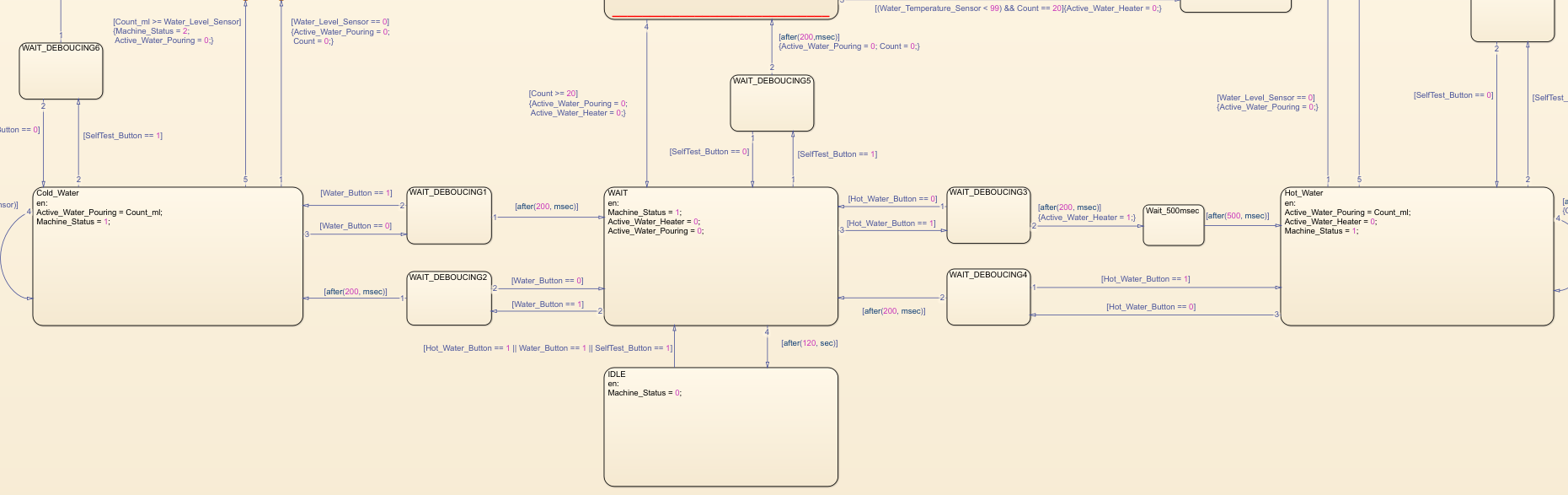
Initially, we turn on the Active\_Water\_Heater, Active\_Water\_Pouring and initialize the Count and Count\_ml with 0. Count is use to increment the Active\_Water\_Pouring at every second with Pour\_Value (input). Count\_ml is used to count how much water has droped from the beggining, to know and verify if we have still remaining water in container.

Remember execution order from the next chart:



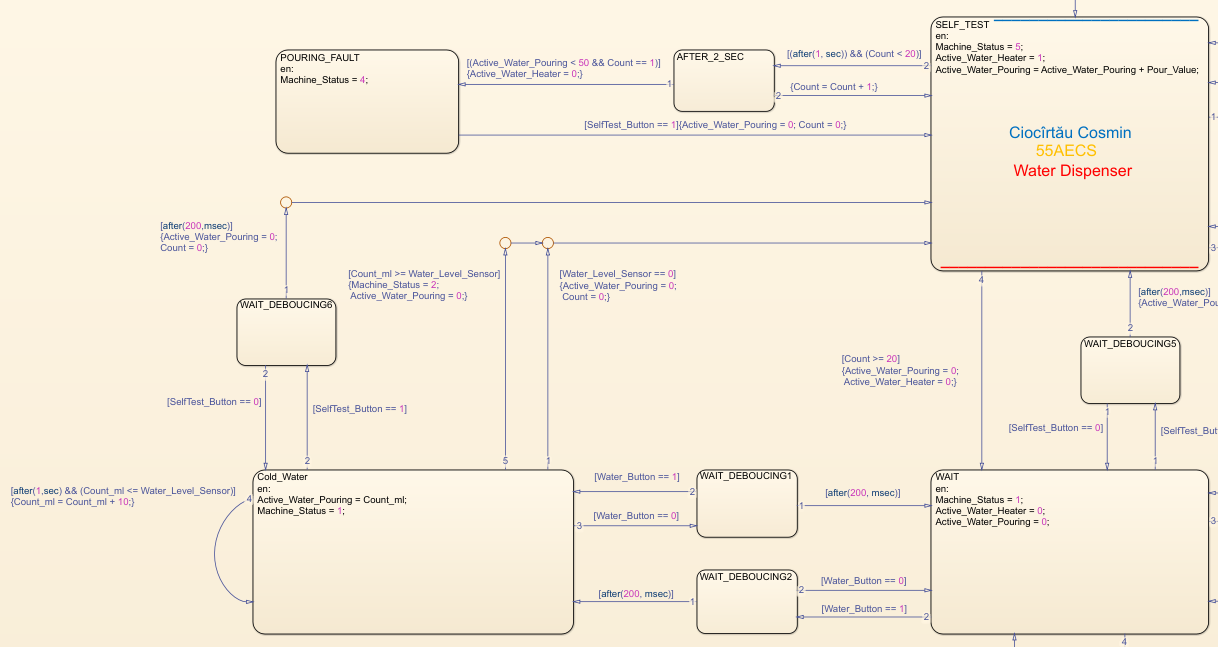
In SELF\_TEST we verify first time the water level, because without water a water dispenser is useless even if the rest of functionality works fine. Second, we enter into a self loop for 20 seconds because, according to requirement from beggining, we need to verify if our machine can reach 99 degrees in 20 seconds. But this will be verified later. After 2 seconds maded in self loop, we verify if our system can drop 50ml water in 2 seconds. If the system doesn’t drop 50ml in 2 seconds, the pouring mechanism is blocked. Then, we verify if the temperature can reach 99 degrees in 20 seconds, if not, there is a heater error.

If the system pass all test, he goes into Wait block.



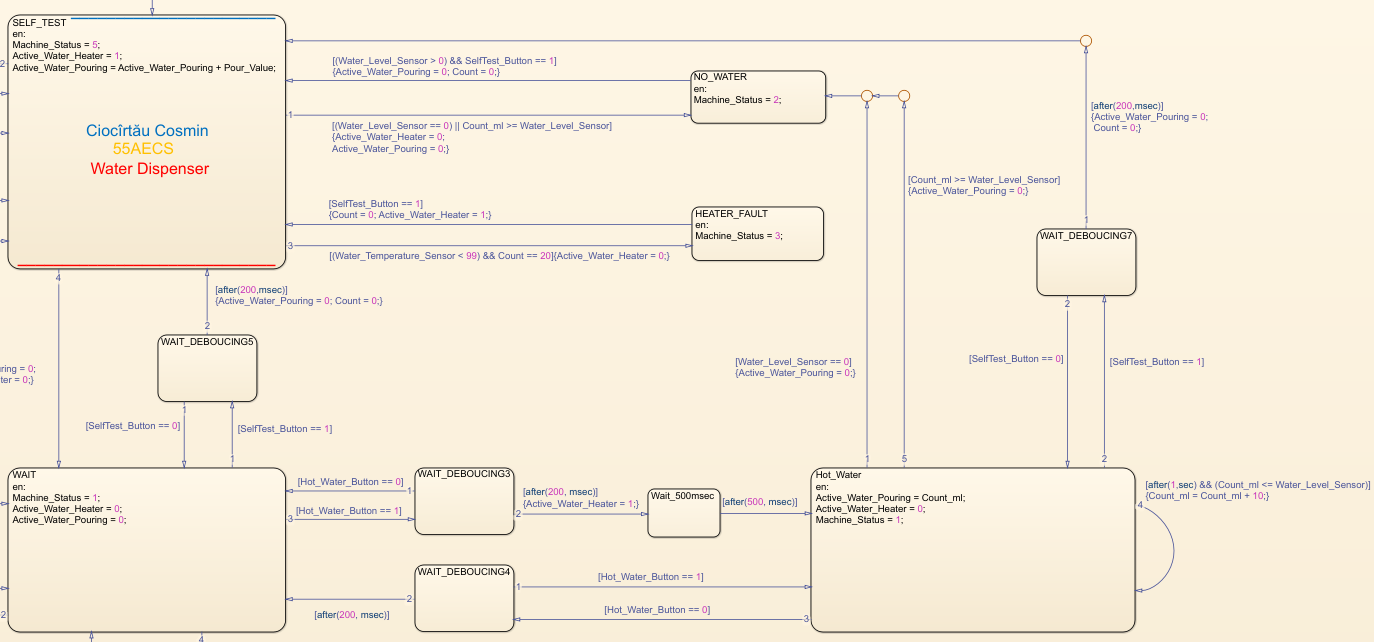
In Wait block, we do nothing, just wait for a next command, which can be a require for Cold\_Water, Hot\_Water, Idle mode or even to go back to SelfTest block.

If the Water\_button is activated still after 200ms (deboucing), the next state where the machine goes in is Cold\_Water.



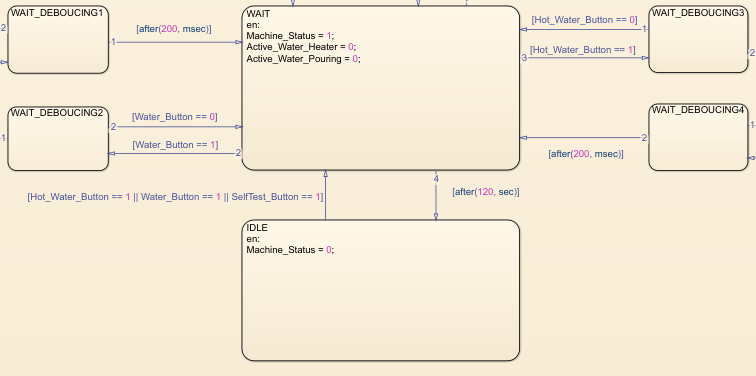
How long we are here, we’ll count how much water is droping. I used a value neither small nor large, so I supposed that the machine will drop 10ml/sec water. We have 3 options to exit from this state: if we have not water remaining in container (Count\_ml >= Water\_Level\_Sensor), if we press SelfTest button and require for a new test of functionality, or if the Water\_button is no longer pressed.

Back to the Wait block, we can also go to the Hot\_Water if we press Hot\_Water\_Button. If the Hot\_Water\_Button is activated still after 200ms (deboucing), we activate Active\_Water\_Heater for 0.5 seconds and then the next state where the machine goes in is Hot\_Water.



This state repeat the functionality of Cold\_Water and the biggest difference is that for this state we use in plus the water heater. One observation is that Count\_ml is further incremented and takes into account its previous value. For example, if we drop some other into Cold\_Water block, and the Count\_ml is now 230ml, the next time when the machine goes into Cold\_Water or Hot\_Water will continue to count from 230ml forward.

Back to the Wait block again, we can go also to Idle block, which represent a ultra power saving mode.

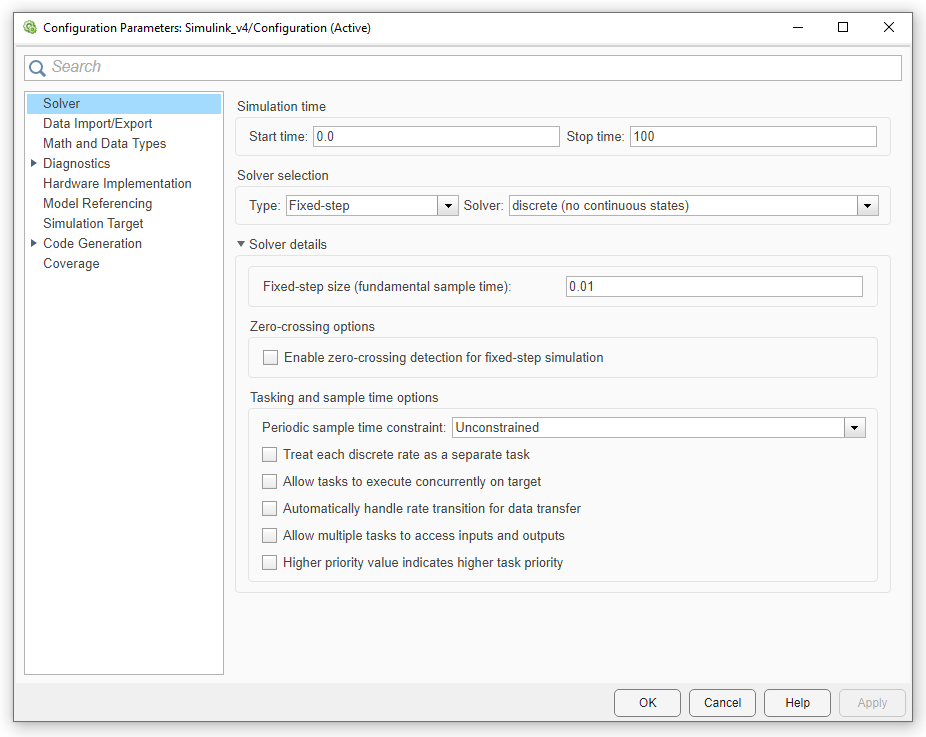


The exit from here is very simple because all we need to do is to press any button from the machine. In real life this block can represent that the machine will turn off all LEDs and extra functionality after some time, while the machine is not used. The machine will go to Idle if at least 120 seconds this is no used.

Observation: From any state we are, we can go anytime we want into SelfTest block to verify again the functionality of the FSM. This happen every time we press SelfTest\_Button from input.

Simulation

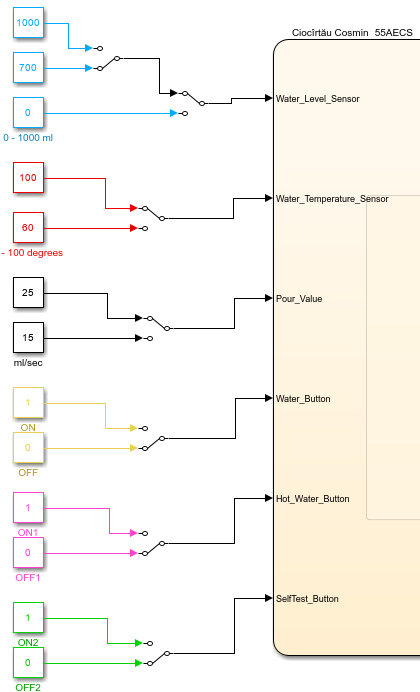
Before simulation, we set next configuration for our model:



I set fixed-step size 0.01, because according to requirements we need to debounce the buttons at 200ms, and for example step size 1 or 0.5 is not properly. The program analyze the signals faster at 0.01 than 1 or 0.5.

In the graphs the signals are delayed with 0.01, exactly the step size.

Let’s supose that we have these 6 inputs:



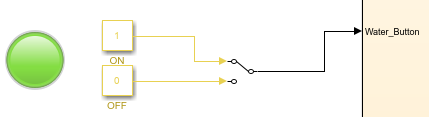
For the moment remember this: we have enough water, and the system meet all requirements to work well. The simulations looks like that (if you can’t see properly, please enlarge the images):



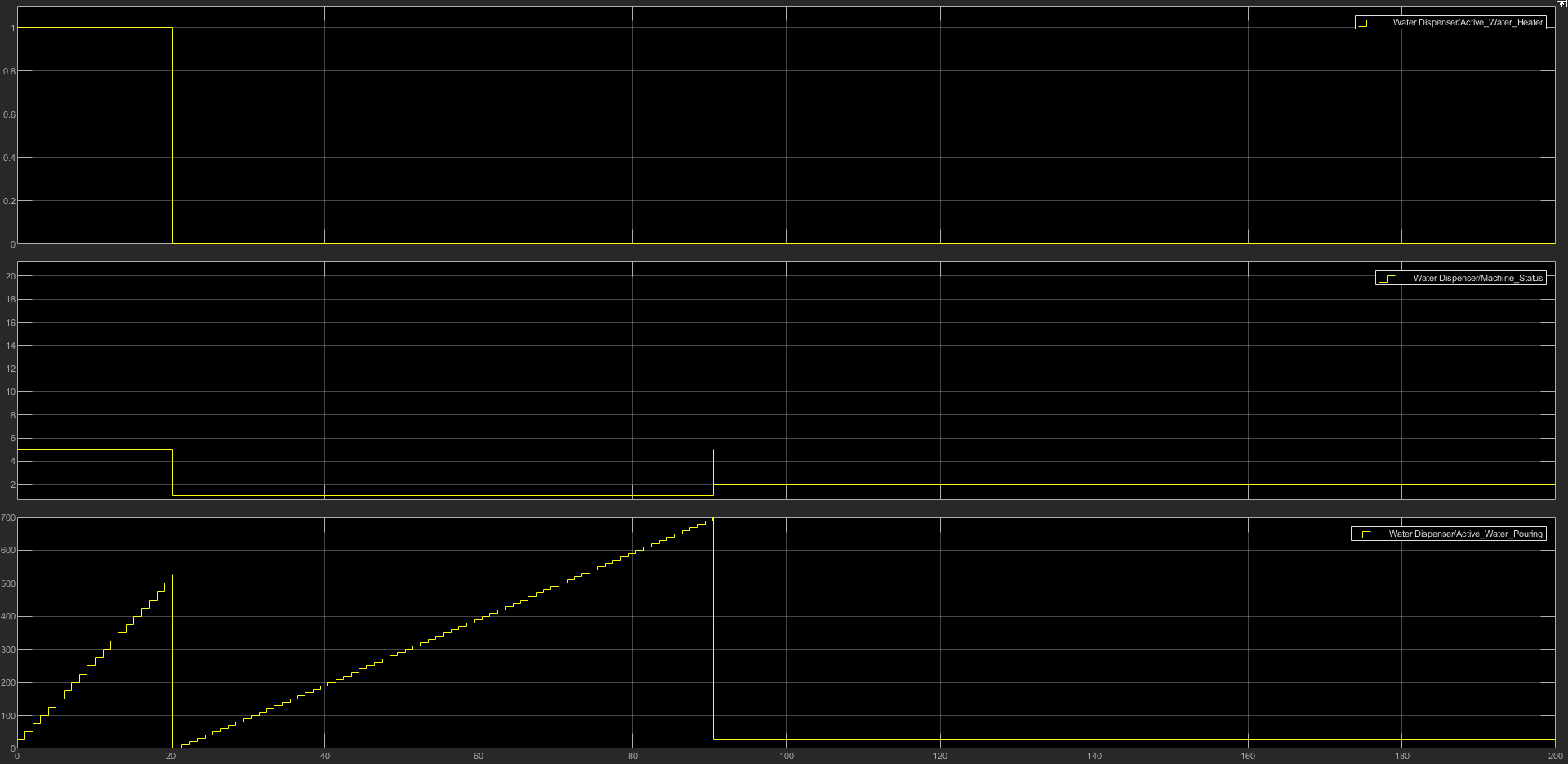
We can observe that the Active\_Water Heater is ON for 20 seconds, Machine\_Status has value 5 (Testing) for first 20 seconds but then has value 1 (Work), Active\_Water\_Pouring increase with 25 ml/sec till second 20. We can observe that imediately after 20 seconds (20 represent the and of the test), all outputs are changed.

* Water\_Button = ON

Now, from Wait state, let’s supose that we activate Water\_Button.



Let’s see how look the outputs:

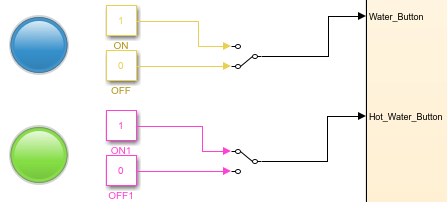


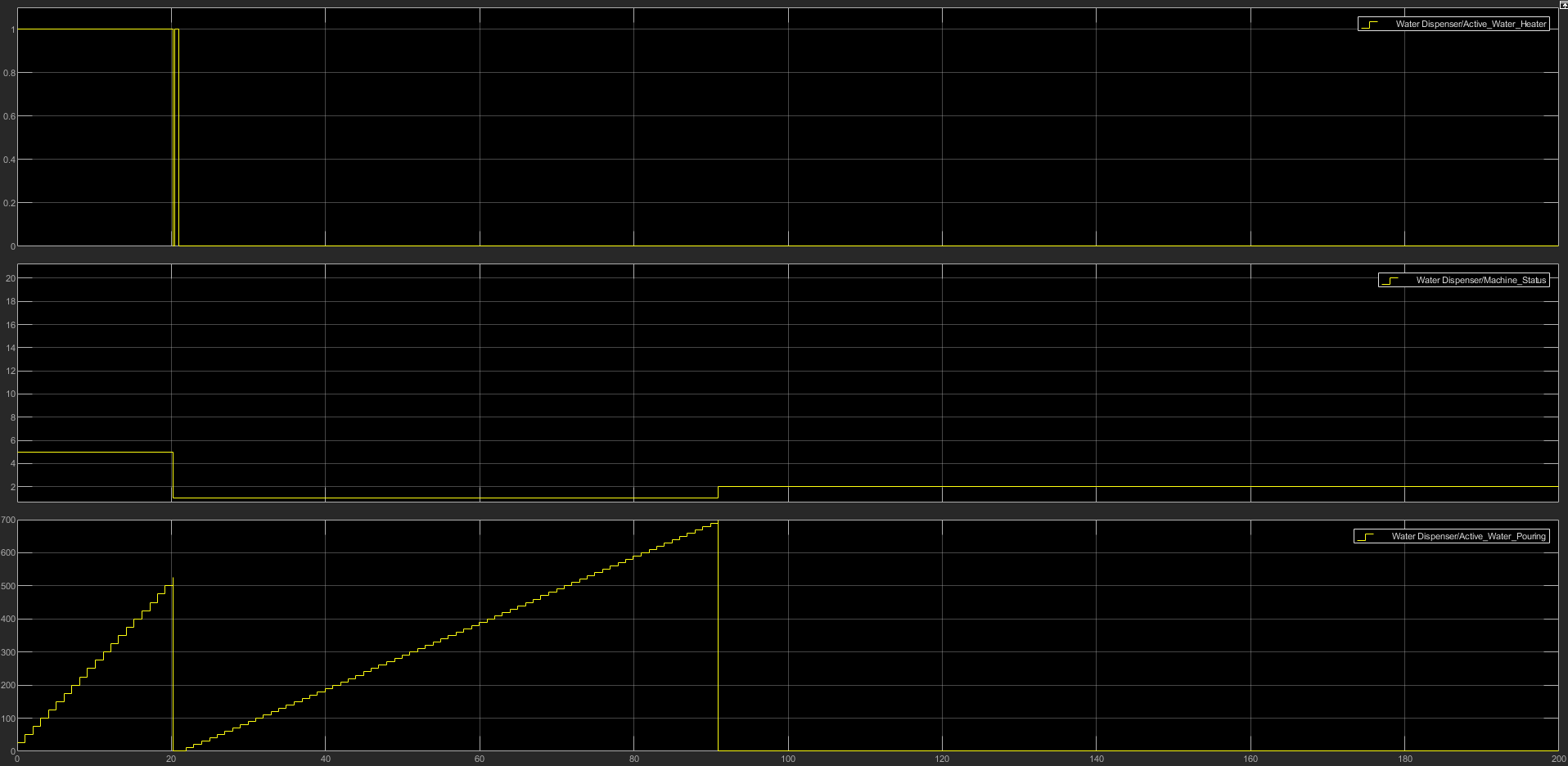
We can see that imediately after checking all functionality from SelfTest, the machine allow water to pouring. At second 90 we can see that Active\_Water\_Pouring has stop, because we have no water left. At the same moment we can see that the Machine\_Status has changed the value from 1 (Working) to 5 (Testing ⬄ SelfTest) and then to 2 (No\_Water). This happens because Cold\_Water state is not directly connected to No\_Water state, he go through SelfTest state first and then to No\_Water.



* Hot\_Water\_Button = ON

Now, from Wait state, let’s supose that we deactivate Water\_Button and activate Hot\_Water\_Button:



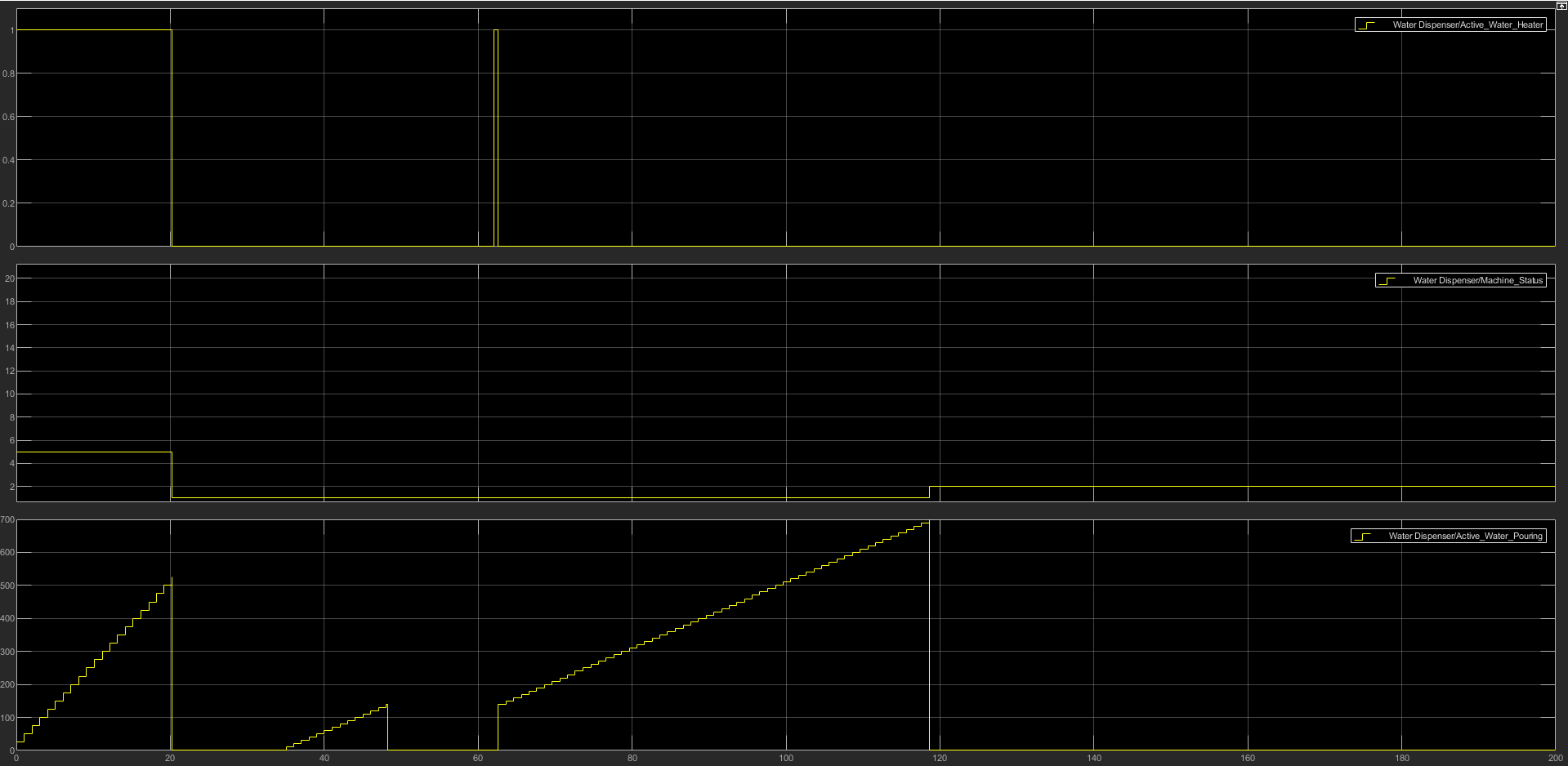


One difference between Cold\_Water and Hot\_Water is that here we activate Active\_Water\_Heater twice, once from SelfTest (for functionality of the system) and once for 500ms (to heat the water) before entry in Hot\_Water.

Another difference is that here Hot\_Water is directly connected to No\_Water state, and that is why Machine\_Status go from 1 (Working) to 2 (No\_Water) and not from 1 to 5 and then to 2.



* Water\_Button = ON || Hot\_Water\_Button = ON (not in the same time)

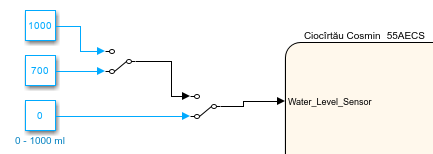


We can observe that:

1. From second 0 to 20: the system test the functionality of the system.
2. From second 20 to 47: the Water\_Button is activated and the machine drop 150ml cold water. Machine\_Status has been changed from 5 (Testing) to 1 (Working).
3. From 47 to 73 sec: the Water\_Button is deactivated and the machine to nothing.
4. From 73 to 117 sec: Hot\_Water\_Button has been required. Active\_Water\_Heater has been activated for 500ms. After this 500ms, the system allow to pouring untill we have no water left.
5. From second 117 to end: Machine\_Status has changed from 1 (Working) to 2 (No\_Water).



* No Water (from the beggining)



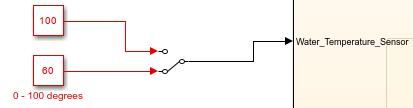
The outputs look like this:

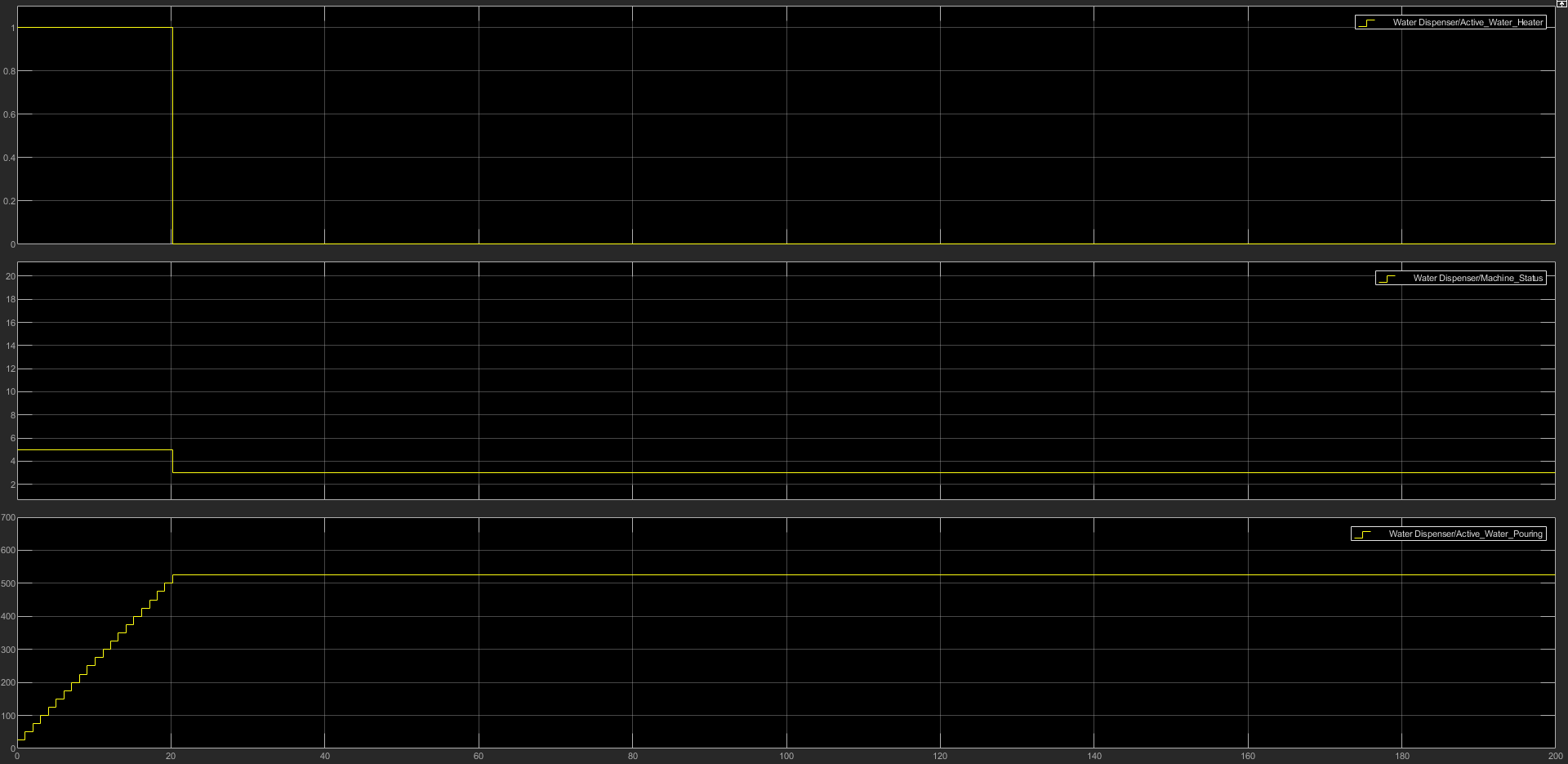


Exactly! All are on 0. If we don’t have water is not needed to test another functionality. Because, as I said at the beggining, a water dispenser is useless without water.



* Heater Fault





We can see that in the first 20 seconds the system keep Active\_Water\_Heater ON and try to heat the water. But imediately after these seconds the system see that he can’t reach 99 degrees in this time and he turn OFF Active\_Water\_Heater and the Machine\_Status goes from 5 (Testing) to 3 (Heater\_Fault).



* Pouring Fault

Let’s see what’s happen if the system can’t drop at least 50ml in 2 seconds and he drop only 30ml in 2 seconds:



The system see that he couldn’t drop at least 50ml after 2 seconds and he know that there is a problem with the pouring mechanism. After 2 seconds Active\_Water\_Heater is deactivated, Machine\_Status goes from 5 (Testing) to 4 (Pouring\_Fault) and Active\_Water\_Pouring is also deactivated.



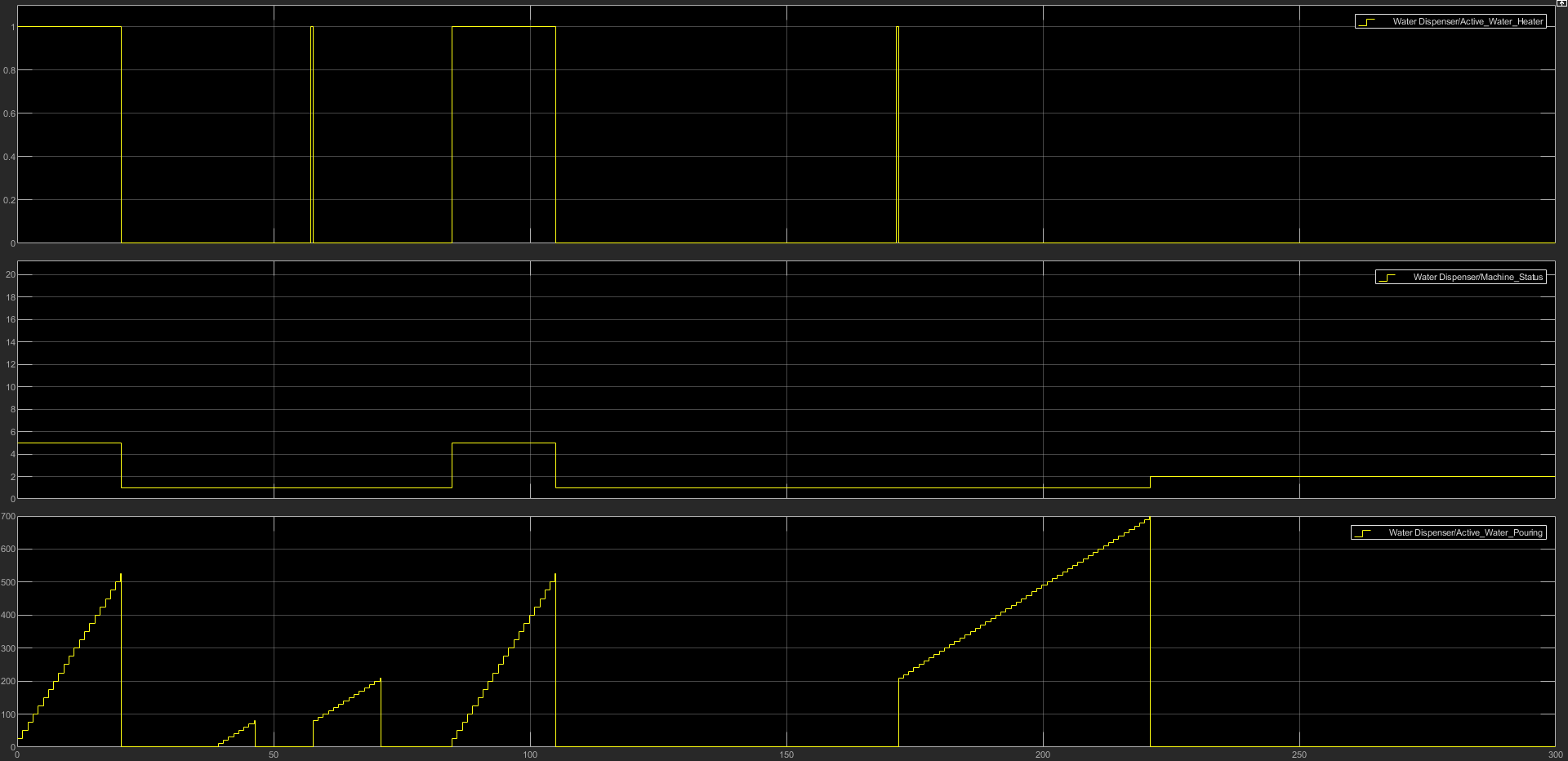
* IDLE state



The system enter into ultra power saving mode after 120 seconds of inactivity. Machine\_Status has been changed from 1 (Working) to 0 (Idle).



* More combination



Let’s describe what happens above:

1. From second 0 to 20: The system start to test the functionality of the machine.
2. From second 20 to 40: The Machine\_Status has been changed from 5 (Testing) to 1 (Working).
3. From second 40 to 46: The Active\_Water\_Pouring is activated.
4. From second 46 to 59: The system is in Wait state.
5. From second 59 to 59.5: The Active\_Water\_Heater is activated ⬄ require for hot water.
6. From second 59.5 to 70: The Active\_Water\_Pouring is activated.
7. From second 70 to 80: The system is in Wait state.
8. From second 85 to 105: The SelfTest\_Button has pressed and the system start to test the functionality of the machine again. The Machine\_Status has been changed from 1 (Working) to 5 (Testing).
9. From second 105 to 170: The system is in Wait state.
10. From second 170 to 170.5: The Active\_Water\_Heater is activated ⬄ require for hot water.
11. From second 170.5 to 220: The Active\_Water\_Pouring is activated.
12. From second 220 to final: Active\_Water\_Pouring become 0 because we have no longer water. The Machine\_Status has been changed from 1 (Working) to 2 (No\_Water).

GitHub

In case that you want to simulate or see in more detail the project, I will let you a link to the GitHub project below. Please notice that I will keep the project public till the end of the semester.

https://github.com/Cosmin45/Matlab