Guzin Gunduz

Charging System for a Vehicle with Multiple Drivelines

1. Introduction

In this work, the aim is to demonstrate my experience in model-based software development, skills in Simulink, and understanding of Electric Vehicle technologies. This Simulink model is a driveline selection system for charging in electric vehicles that have more than one driveline.

Conventional EV charging protocols (for example CCS, widely accepted protocol in Charging systems), supports charging of only one driveline (thus one battery pack). In this work I focused on a problem: What if a vehicle has more than one driveline? For example, vehicle designers might choose to use multiple drivelines for applications that need more power capacity. And they can use existing drivelines in the same vehicle to quickly develop a vehicle. In these cases, charging will be a major problem and I focused on some parts of this issue in this project.

My software acts as a charging controller to charge multiple drivelines in a vehicle simultaneously. More specifically, this software is a driveline selector. In Ccs charging, multiple drivelines can be charged at the same time if they are using AC charging. But if they are using DC charging, multiple drivelines cannot be charged simultaneously because each driveline will require different levels of Voltage and Current in DC charging. So, we need to charge them one by one. Selecting which driveline to charge and deciding when to switch to the other drivelines for charging or deciding when to stop charging is an important decision of this process. This model handles these decisions.

2. Modeling

To be able to develop this software based on a model, I started by creating a simple mathematical model of a battery (In Figure2.1), which will be used in our model.

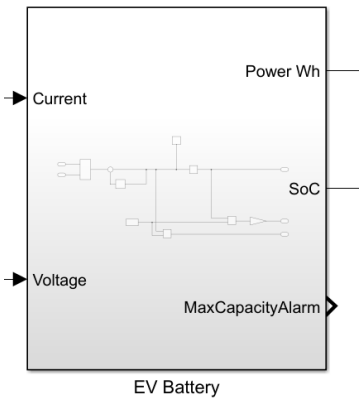

Figure 2.1: eV Battery

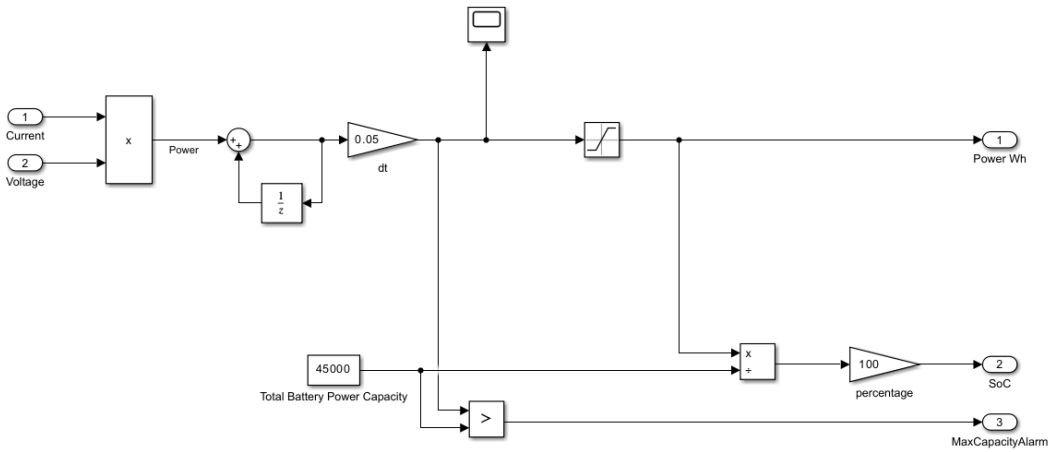

Figure 2.2: The Model of the Battery

This is the simple eV battery model and the second picture (Figure2.2) shows the modeled battery subsystem. Inputs of my battery model are Voltage(V) and Current(A). Inside the battery model, instantaneous power (Voltage x Current) is accumulated to mimic the battery charging behavior. I decided the battery to have 45kWh capacity, which is usual in many EVs. The three outputs of the battery model are the SoC level of the battery, the "Fully Charged" alarm, and the stored power of the battery. At this point, the battery model is completed, and all the adjustments are done.

For this work, I placed 3 batteries in my model for a vehicle with 3 drivelines. Finally, I put unit delays in my battery outputs, so I can test it in the same model (cause and effect needs to be in order) with my charging controller software.

Now, I had to come up with a charging strategy for these batteries. I decided to first charge the battery with the lowest SoC level until it increases to %2 more than the previously highest charged battery. Then repeat this logic again and again until all the batteries are charged. This way the SoC levels of each battery will be closer to each other, and this way, we will not focus on a single battery and heat it a lot. That is another advantage of my algorithm. The number, %2, can be changed for more optimization (like the temperature of the batteries)

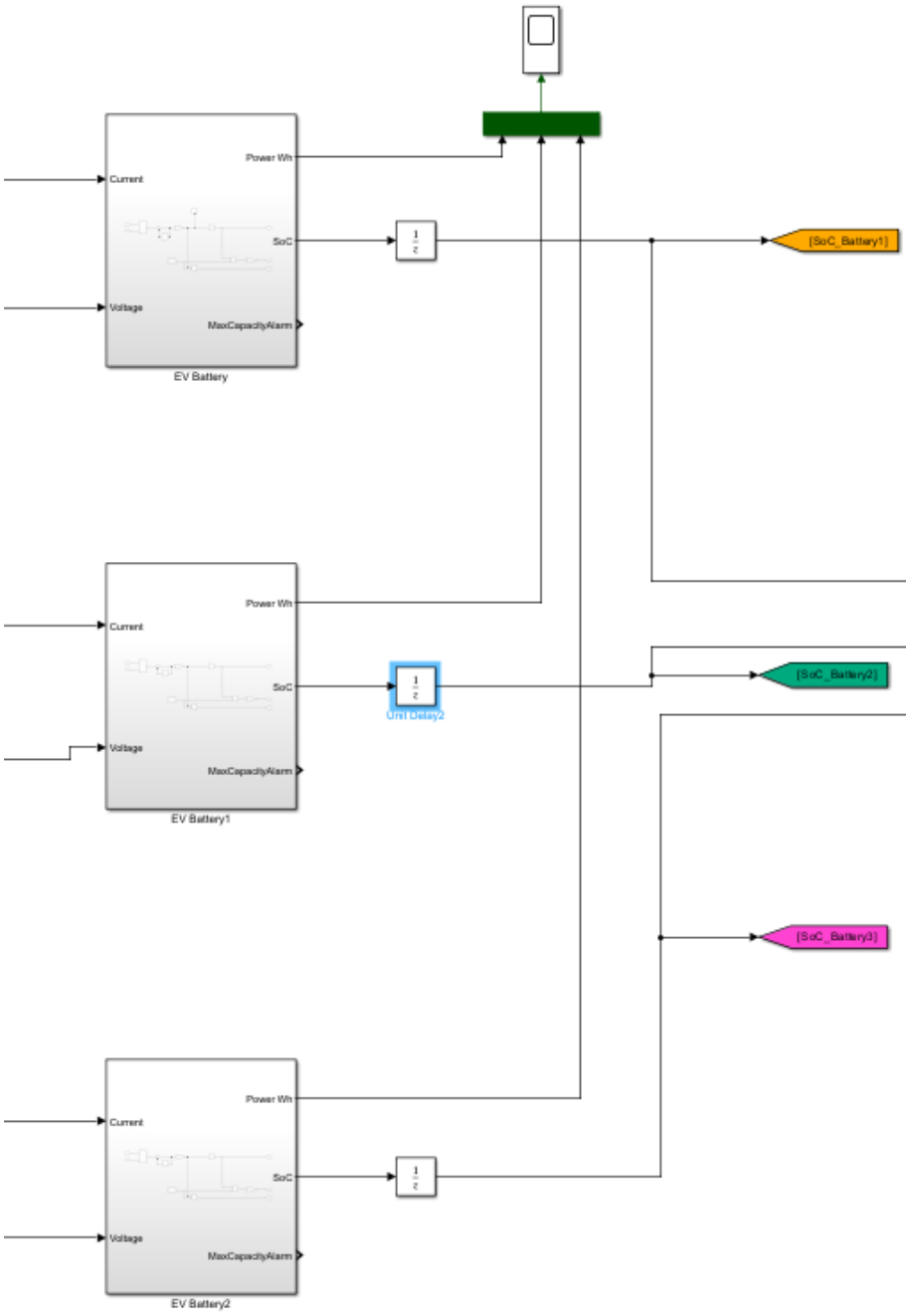

Figure 2.3: Shows the 3 Batteries in the test bench and Unit Delay Blocks

For the next step, I wrote a short function (in Figure 2.4 and 2.5) using a MATLAB Function block located in the user-defined blocks section of the Simulink Library. This function is developed to identify and select the driveline (battery) with the lowest SoC level. The output of the function is connected to a Simulink GoTo Tag, named as LowestBatteryID, and used through the model. This signal will be the main selector signal. Peripherals of this function block has summation blocks to select the drivelines when their soc levels are %2 higher than the other drivelines' batteries. This calculation runs continuously, so we know soc level of each battery in each time step and we can switch them whenever they reach the necessary levels.

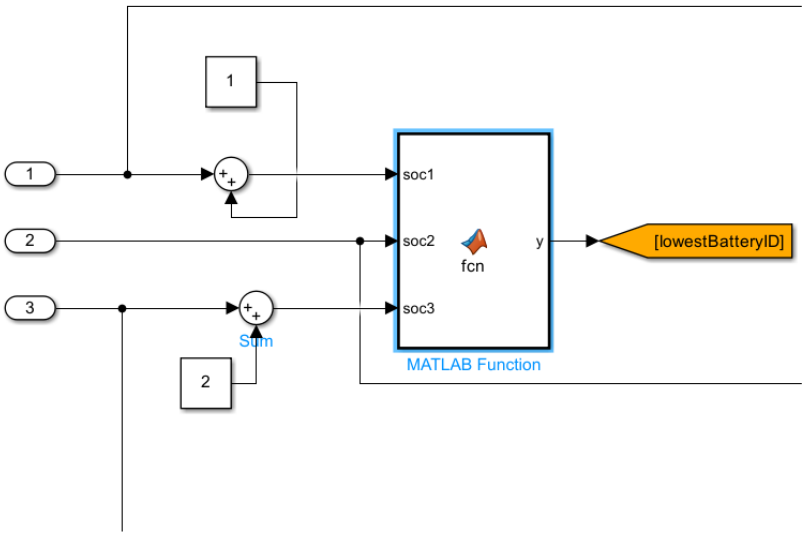

Figure 2.4: a MATLAB Function Block

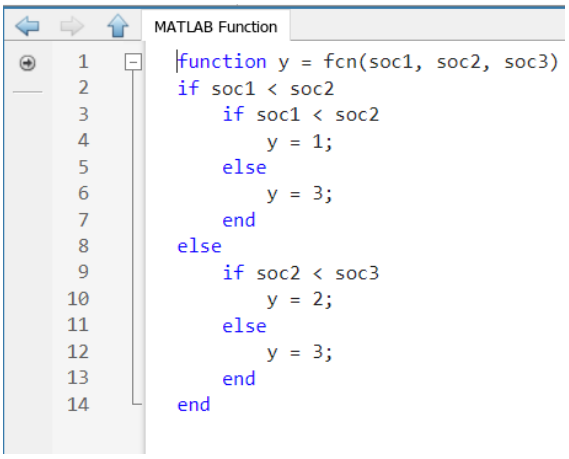

Figure 2.5: MATLAB Function to detect the battery with the lowest SoC level

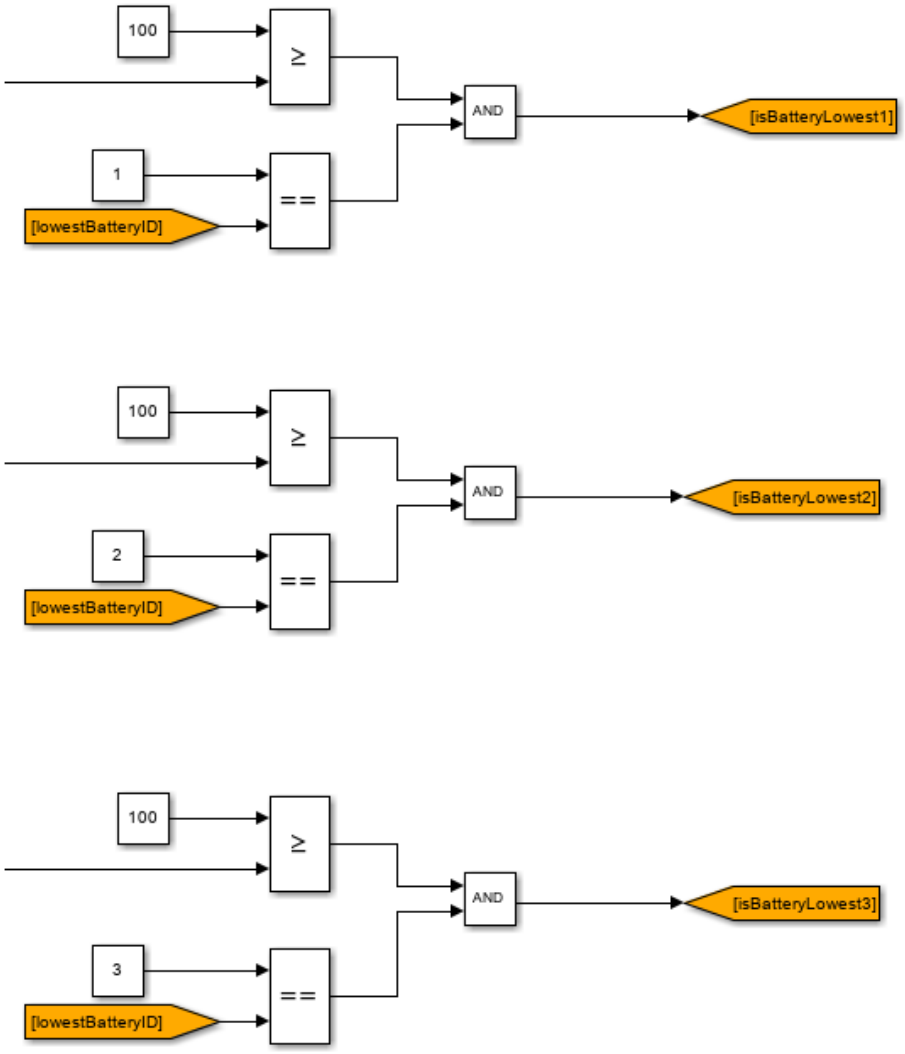

Figure 2.6: Relational Blocks and Logical Operators to construct each batteries state signal

In this part of the model (Figure 2.6), I compared the batteryIDs with the LowestBatteryID, which is the output of the previous Matlab Function block. And each battery's SoC levels are compared with %100, in order to have information if they are fully charged or not. These 2 relational operators' outputs are input of the AND block for each battery. So, we have battery state signals, called isBatteryLowest<N>, for each battery. If a battery is not the lowest charged battery, or it is fully charged, the controller will not provide current and voltage to them. Then the battery with the lowest SoC level will be charged or none of the batteries will be charged if they are all fully charged.

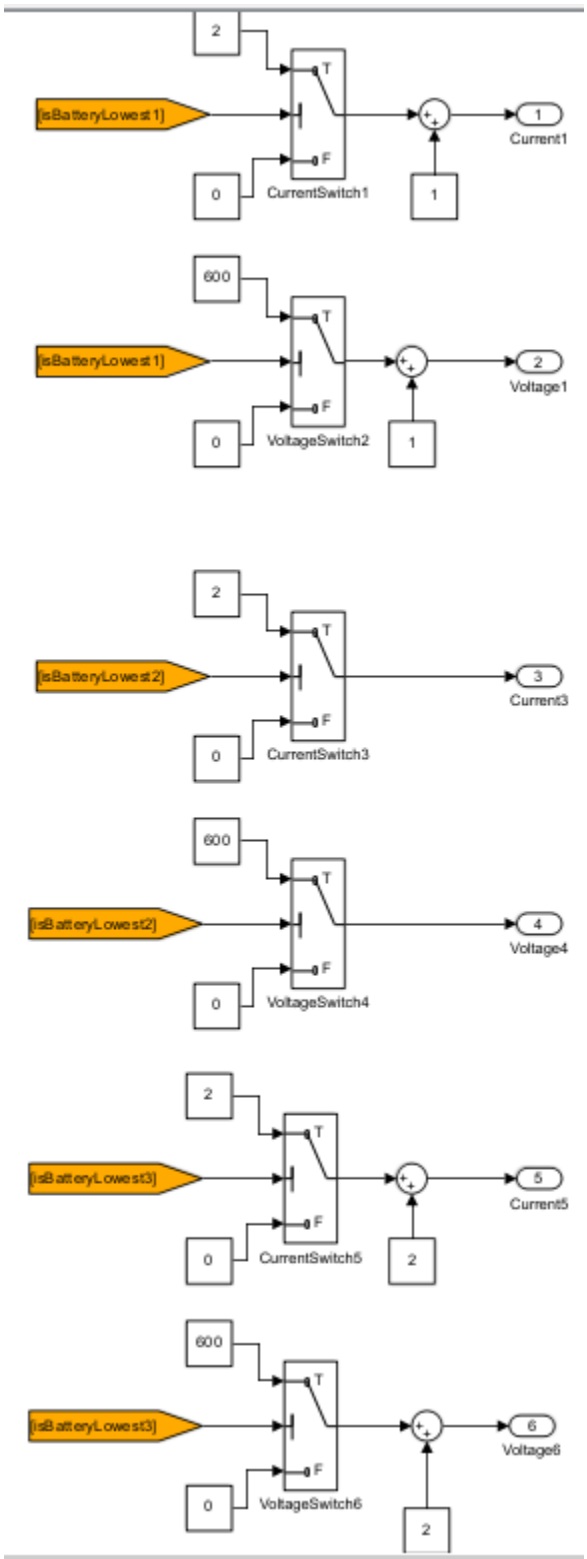

Figure 2.7: Charging Switches controlled by the battery state signals.

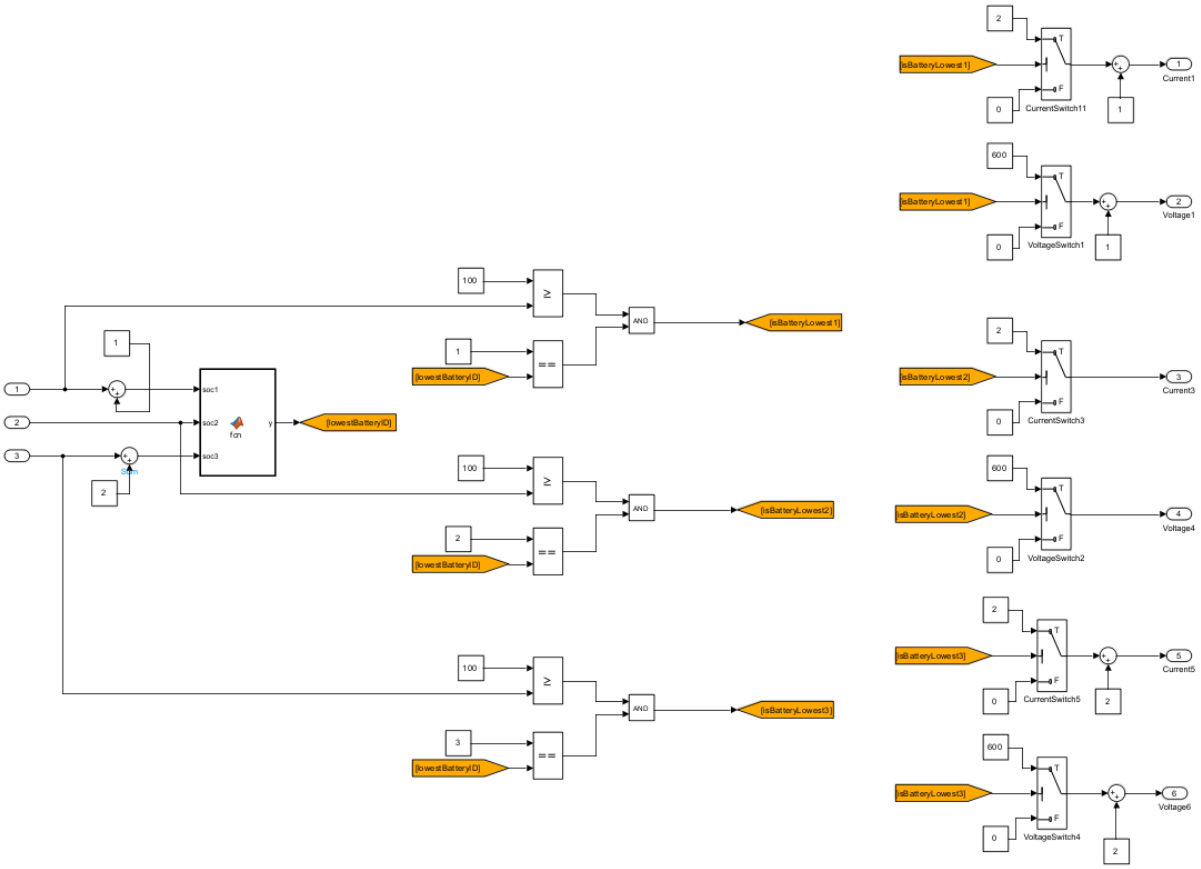

Figure 2.8: A Complete View of the Charging Controller Subsystem

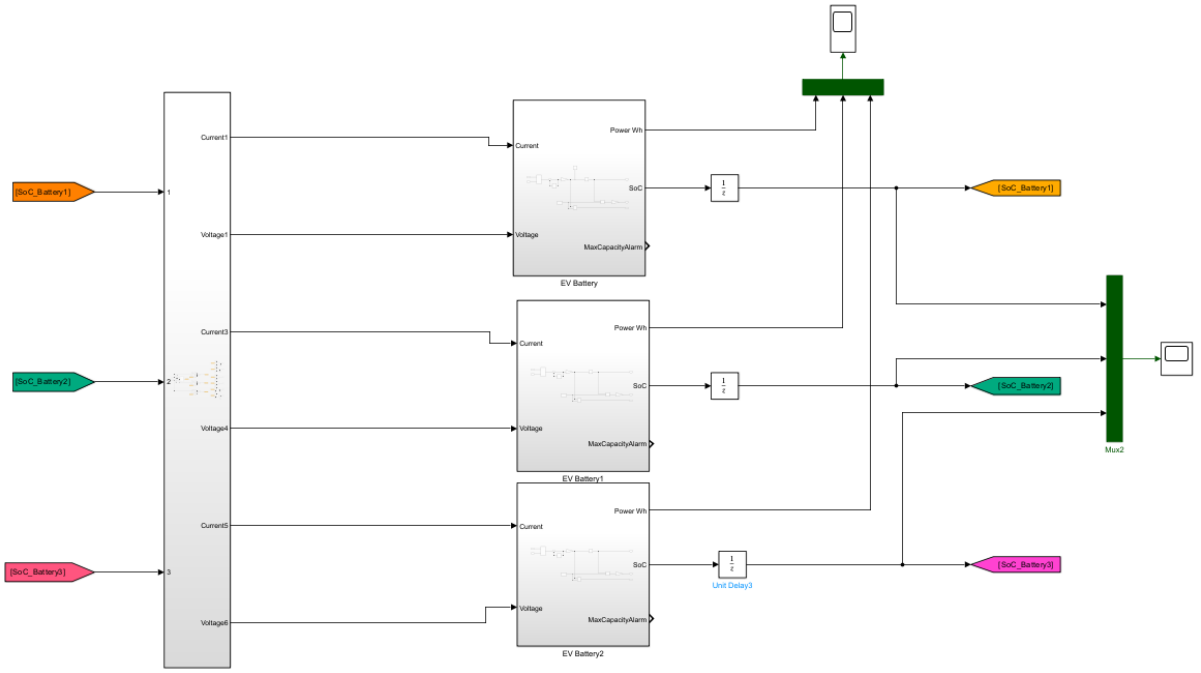

Figure 2.9: Complete View of The Simulink Model

3. Test and Results

After the development is completed, parameters are tuned, and the model is verified by running the controller and the battery models in the same Simulink model. In the next section, the results of the charging controller can be seen by looking at the SoC levels and power graphs of each battery.

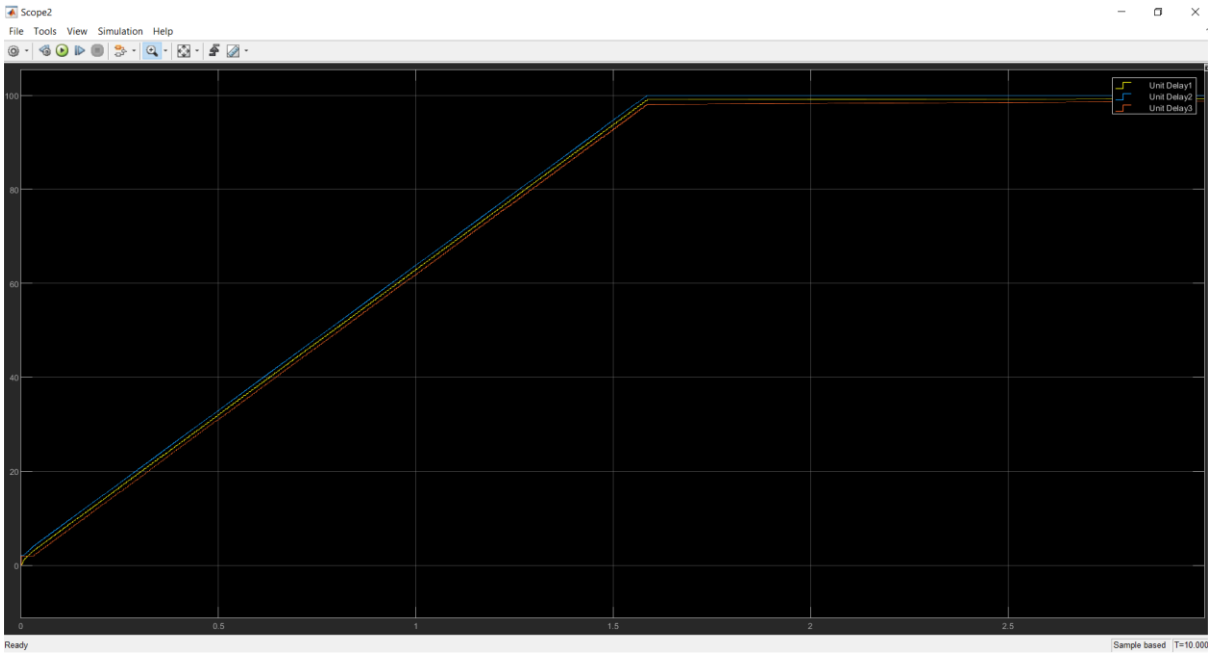

Figure 3.10: SoC Levels of Each Battery

In Figure 3.10, SoC level signals of each battery can be seen. Signals rise together until they are fully charged, and they are charging one by one until one of them has charged %2 more than the second one. They are not simultaneously charged but their selection makes their soc levels look like they were charged simultaneously. This behavior can be seen more clearly in Figure 3.11.

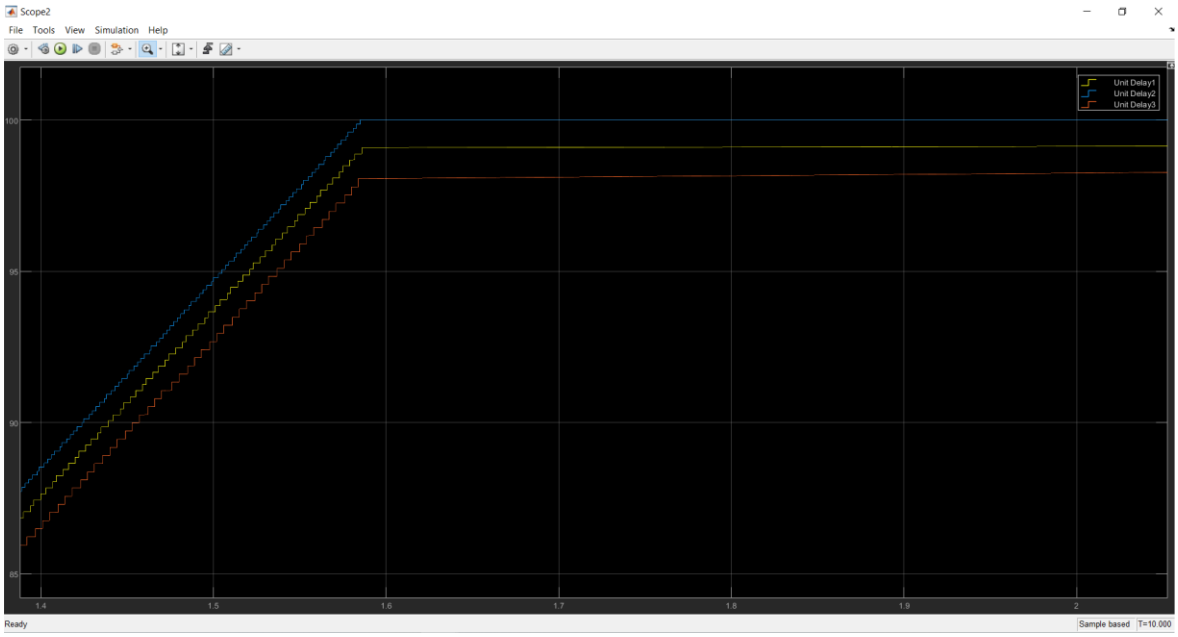

Figure 3.11: Detailed SoC Levels - SoC_1(yellow), SoC_2 (blue), SoC_3(red)

4. Summary

I designed an EV battery DC charging selection software for vehicles with multiple drivelines In Simulink. It is a model-based software development process, so I first started development with modeling the batteries. A simple mathematical battery model is used. Then I started the development of the controller software. I decided on a charging strategy and implemented that strategy in Simulink. And finally, the verification of the software is completed by providing the graphs of each battery's SoC and Power levels.

For the next steps, there are some areas to improve. For the battery models, temperature can be included. So, we can increase the temperature with a simple mathematical model again, and as temperature increases, we can slow down the charging speed.

Also, we can include contactors of the Dc charging in the model. So instead of providing current to the batteries directly, we can open or close the contactors of each batteries switch unit.