**EMBEDDED SYSTEMS PROJECT REPORT**

**Processor-in-the-loop Implementation of Master-Slave Battery Management Systems**

**E-PiCo 2022-2024**

**Guillermo Israel Buenfil Solis**

**Gustavo Gomez Casanova**

**Musa Matthew**

**Bramantio Yuwono**

**Umar Zia Syed**

****

****

**March 2024**

1. **System Overview**

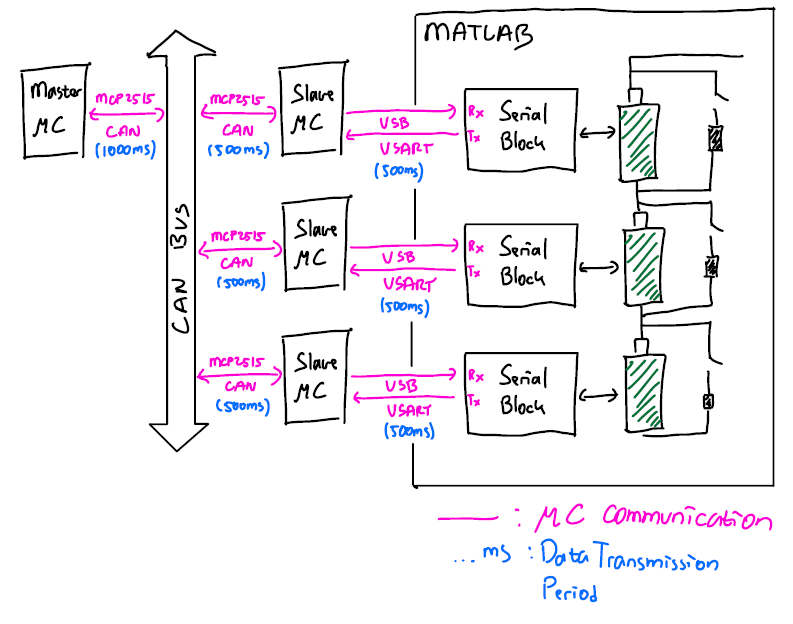


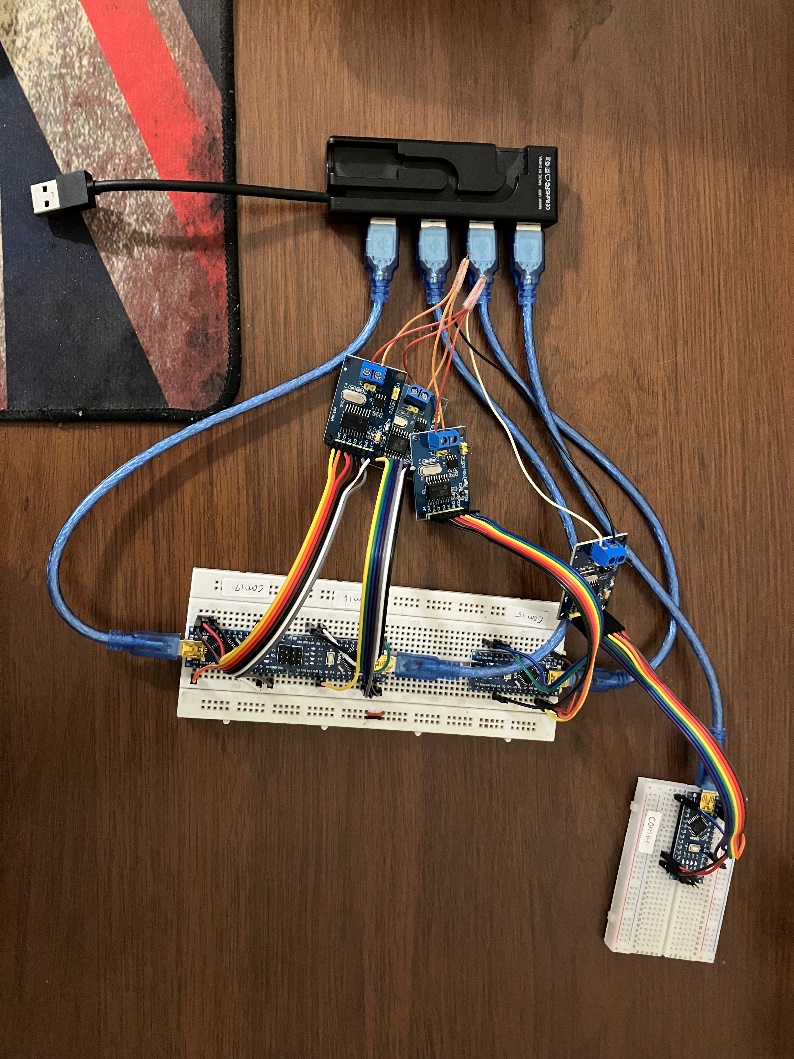
Figure 1 Overall System

The system's main purpose is to implement the voltage-based balancing controller without needing actual batteries connected to the battery management system. With this proposed method, the battery balancing algorithm can be verified without the risk of harming the actual batteries.

The battery cells and balancing circuit models are emulated on Simulink. Then, the models are interfaced with the battery management system utilising serial blocks. Serial blocks enable the data exchange between the controller implemented on Arduino Nano and the models on Simulink.

The Master-Slave BMS topologies are chosen for this project. In this topology, BMS slaves only read the battery parameters from the sensors (or in this case, the serial block) and pass the data to the BMS Master via CAN bus. Subsequentially, the BMS Master receives the battery parameters from all BMS Slaves and processes the data using the balancing algorithm to decide which cell requires to be balanced. Moreover, the BMS Master transmits the balancing command to each of the BMS Slaves. Based on the command from the BMS Master, BMS Slaves apply the passive balancing action (let the battery drain the charge through the resistor).

For this project, the transmission period for communication between the model and BMS Slaves, BMS Slaves to CAN bus, and the BMS Master to BMS Slaves via CAN bus are set to 500ms, 500ms, and 1s respectively. The effect of the transmission period on the balancing performance is exempted for this project.



BMS Slaves

BMS Master

Figure 2 Real Implementation of BMS Master-Slave PIL

The implementation of the Master-Slaves BMS is shown in the figure above. All BMS modules are implemented on Arduino Nano. To enable CAN communication, MCP2515 is connected to each of the BMS modules. The connection between BMS Slaves and the models on Simulink is realized with USB to the USB hub. The USB connection between the BMS Master and the USB hub is made only for debugging purposes, not related to the system's main functionalities.

1. **Implementation Details**
   1. **Battery and Passive Balancing Models on Simulink  
      <Gustavo>**

Contents to put here:

* + - Picture of simulink model
    - Battery model explanation
    - Battery model parameters and its explanation
    - Balancing circuit model explanation
    - Relation between the measured parameters in the model and real implementation
  1. **PIL Communication: Matlab Serial Block and Arduino Program  
     <Memo>**Contents to put here:
     + Picture of simulink model
     + Serial block implementation explanation
     + Connection between the battery and balancing circuit models with serial block
     + Interface between Simulink and Arduino
     + Arduino Serial receiving and sending functions: readFromMatlab() and writeToMatlab()
  2. **Battery Management Master Algorithm Explanation  
     <Umar>**

Contents to put here:

* + - Overview of BMS Master basic functionalities
    - BMS Master pin connection (to MCP2515) explanation
    - BMS Master parameters explanation
    - Details about transmitted CAN data by BMS Master
    - Flowchart of BMS Master Algorithm -> make it concise not too detailed   
      (please make it using [yEd Graph Editor](https://www.yworks.com/products/yed))
    - Algorithm explanation (including the balancing algorithm)
  1. **Battery Management Slave Algorithm Explanation  
     <Musa>**

Contents to put here:

* + - Overview of BMS Slave basic functionalities
    - BMS Slave pin connection (to MCP2515) explanation
    - BMS Slave parameters explanation
    - Details about transmitted CAN data by BMS Slave
    - Flowchart of BMS Slave Algorithm -> make it concise not too detailed   
      (please make it using [yEd Graph Editor](https://www.yworks.com/products/yed))
    - Algorithm explanation

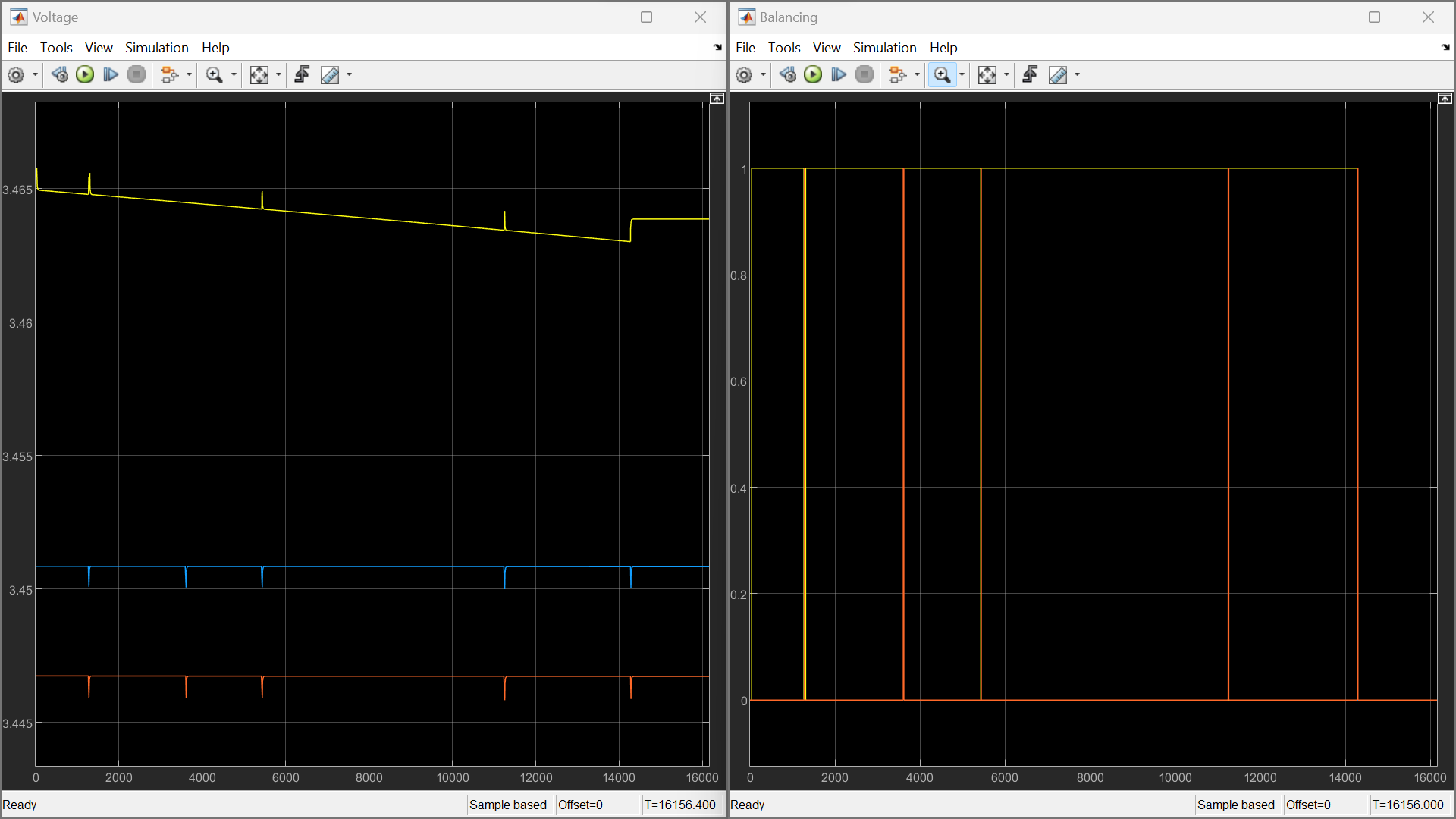
1. **Result and Observation**

Figure 3 Implementation Result: Battery Voltage(left) and Balancing Command(right)

The test is performed while the battery is in an idle state (no charging or discharging) to ensure the observation of the balancing algorithm performance is not affected by the battery charging/discharging process. Initially, the SOC for each cell differed to 85% for cell 1 (yellow), 55% for cell 2 (blue), and 50% for cell 3 (red).

Generally, it can be observed that the balancing algorithm works well anticipating the voltage imbalance between three cells until the third cell voltage is reduced to the voltage level imbalance allowable range (5mv between each cell and the mean voltage of all three cell voltages).

However, in some cases, there are sudden drops in cell voltage likely due to the sudden change in the balancing command for cell 3. There are three possible causes of this: the misprocessing by the balancing algorithm, serial/CAN communication delay, and battery model abnormalities. Regarding this, further investigation is required.

1. **Conclusion**

From this project, it can be concluded into these points:

* + - The implemented processor-in-the-loop battery management system with master-slave topology for passive balancing can be emulated utilizing battery cell and balancing circuit models on Simulink connected to BMS modules via serial communication.
    - Generally, the balancing algorithm on the BMS Master can identify which cells require to be balanced and send the command to the BMS Slaves. It can stop the balancing process also when the imbalance conditions are not fulfilled anymore.
    - During testing, it is observed that there are sudden voltage drops in some cases. Further investigation on this is required to solve this issue.
    - Further improvement for future works: SoC-based balancing algorithm, incorporating temperature into the battery model, investigation about the effect of CAN/serial transmission period on balancing performance, cell operating condition monitoring function.