MBSD Assignment #4 A.Y. 2024/25

Purposes

- Integrate the one-pedal controller into a simulated Arduino Uno[1] microcontroller (µC), resorting to SimulIDE[2].
- Interact with the µC through its digital and analog interfaces.

Instructions

For instructions on how to use SimulIDE and the Platform Support Packages, follow the instructions provided by Prof. Violante in the lectures.

The delivery shall contain:

- The controller Simulink model is used to generate the firmware binary file (plus all the accompanying files needed to make it possible to generate the code again, like .m files containing initializations)
- The firmware binary file to be loaded into the simulated Arduino in SimulIDE
- The SimulIDE project file.
- The PDF or Microsoft Word version of the report.

Is available an example based on a Tank level controller in the folder

The deliverable has to be provided as a . ZIP file up to **June 8th at 23:59. It shall also contain a brief report explaining the** integration process using the following template. It is sufficient that only one of the group members uploads it.

[1] Arduino Uno Board Anatomy https://docs.arduino.cc/tutorials/uno-rev3/BoardAnatomy, last visited on 14/05/2025.

[2] SimulIDE, https://www.simulide.com/p/home.html, last visited on 14/05/2025.

Model-Based Software Design, A.Y. 2024/25

Assignment 4 Report

Components of the working group (max 2 people)

- Amirhossein Ayanmanesh Motlaghmofrad, s323874

I/O interfaces

[Please describe the I/O interfaces to interact with the one pedal controller through the electrical interfaces of the Arduino]

ADC pins of the Arduino board is used as the input interfaces, in Arduino Uno, these pins are A0 to A5. Here, the aim is to simulate the input signals through analogue voltages that assume values between 0 and 5. Then, the digital value of the input signal is multiplied by 5/(2^10-1) to convert the digital value to volt. From here, according to the expected value of each signal, this value is multiplied by a gain and is casted accordingly.

As BrakePedalPressed signal, it is expected to obtain a Boolean values signal, therefore, the output of ADC is first multiplied by 5/(2^10-1), then it is multiplied by 1/5 to map 0-5 into 0-1, then the output is rounded and casted to Boolean.

As ThrottlePedalPosition1,2,3, we need single value between 0 and 1; therefore, simply the voltage value of ADC is multiplied by 1/5, and is converted to single data type.

For transmissionMode, it is required that ADC output is mapped between 0-4, rounded and then casted to data type enum: TransmissionState.

For the velocity the voltage equivalent of ADC is mapped between -60-240, this mapping is done by a linear mapping. That is, 0 analogue voltage in input is mapped to -60 Km/h and the voltage 5 is mapped to 240 Km/h.

For the output interface, the Digital outputs of the Arduino platform support package is used.

As to Torque request_Nm, a PWM output is used in order to represent the magnitude of the torque requested. For doing so, the value of the torque requested should be mapped from 0-255. In order to do so, -80 is mapped to 0 and 80 to 255. Therefore, 0 torque correspond to 50% PWM, and -80 means the output would be 0.

The TransmissionState is divided into 5 pins, each of which corresponding to one transmission mode; and for Warning and Fault signals, just a simple digital output is used.

[Screenshot of the controller model ready for the code generation]

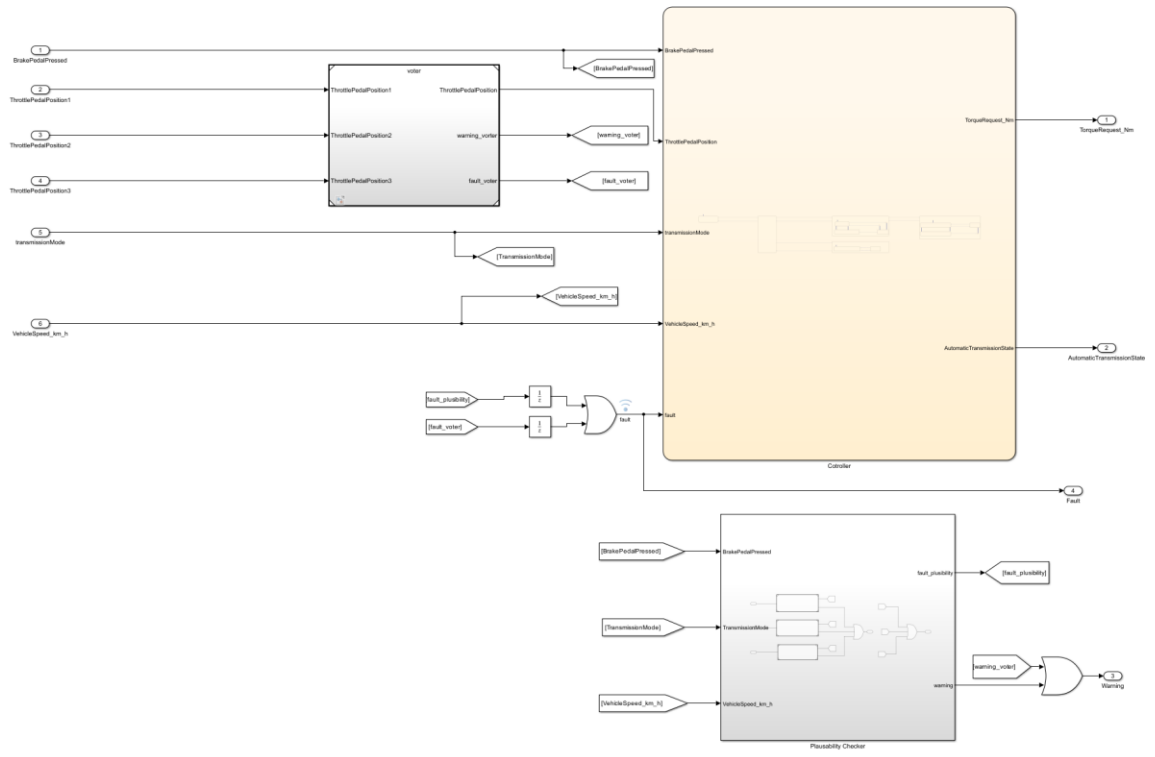

Figure 01: The Simulink scheme of the controller for conde generatio; with respect to the previous lab, some plausibility checks are removed in order to make the software suitable for stimuli can be tested in simulIDE.

Name	**Unit**	**Type**[3]	**Conversion formulas**	**Min**[4]	**Max**
`BrakePedalPressed`	-	AI	ADC/5	0	1
`ThrottlePedalPosition1`	V/5	AI	ADC/5	0	1
`ThrottlePedalPosition2`	V/5	AI	ADC/5	0	1
`ThrottlePedalPosition3`	V/5	AI	ADC/5	0	1
`transmissionMode`	-	AI	ADC*4/5	0	4
`VehicleSpeed_km_h`	Km/h	AI	ADC*60 - 60	-60	240
`TorqueRequest_Nm`	N*m	DO (PWM)	PWM = 127.5 + 127.5*x/80	-80	80
`AutomaticTransmissionState`	-	DO (5 pin)	-	0	4
`Warning`	-	DO	-	0	1
`Fault`	-	DO	-	0	1

[3] Digital Input (DI), Digital Output (DO), Analog Input (AI).
For AIs, provide the conversion formula from input voltage to the measurement unit data (indicating also how to perform the conversion from the raw reading of the ADC).
[4] The Min/Max values that can be handled due to the conversion formula shall be expressed in the measurement unit specified in the Unit column.

Code generation for Arduino

[Please describe the steps to generate the Arduino firmware using the Simulink Support Package for Arduino Hardware]

The solver used is Fixed-step with fixed-step 0.1 second. In the hardware implementation, Arduino Uno is selected and Build action is set to Build, where the generated code later is used in the SimulIDE. In the Code Generation, as System target file, ert.tlc is chosen and the language C with standard C89/C90 (ANSI) is used. For the tool chain Arduino AVR is chosen. In the interface setting of Code Generation, Nonreusable function is used, since the aim is full executable and not algorithm export for this purpose. Further, the check mark of continuous time is removed. Then, the code generation toolbox of Simulink is used in order to generate the full executable export.

[screenshot of the controller model instrumented with the blocks of the Support Package for Arduino Hardware]

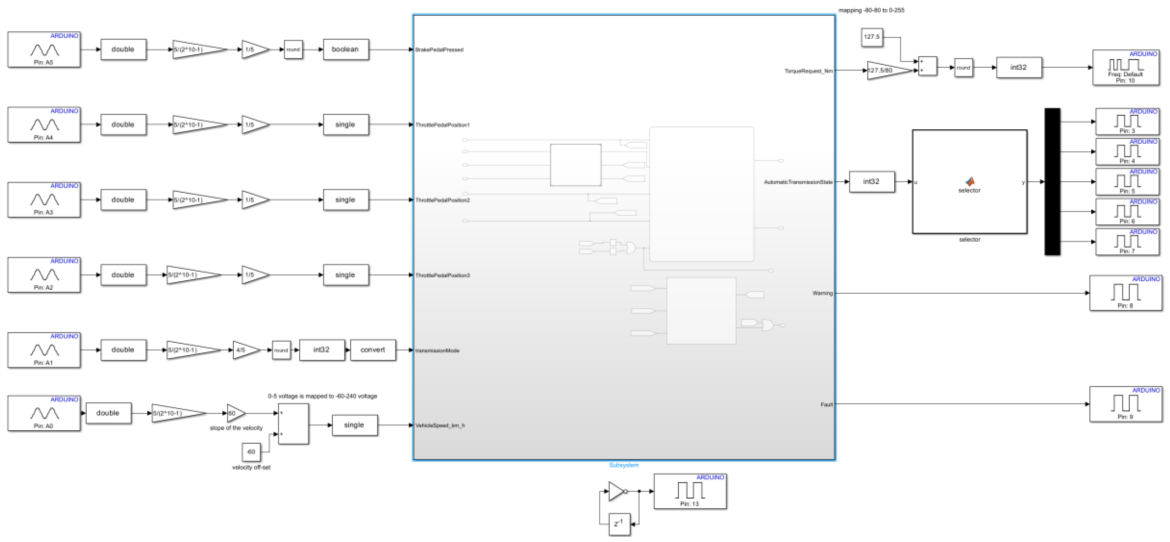

Figure 02: the Simulink scheme used for the code generation

Harness

[screenshot of the harness implemented in SimulIDE]

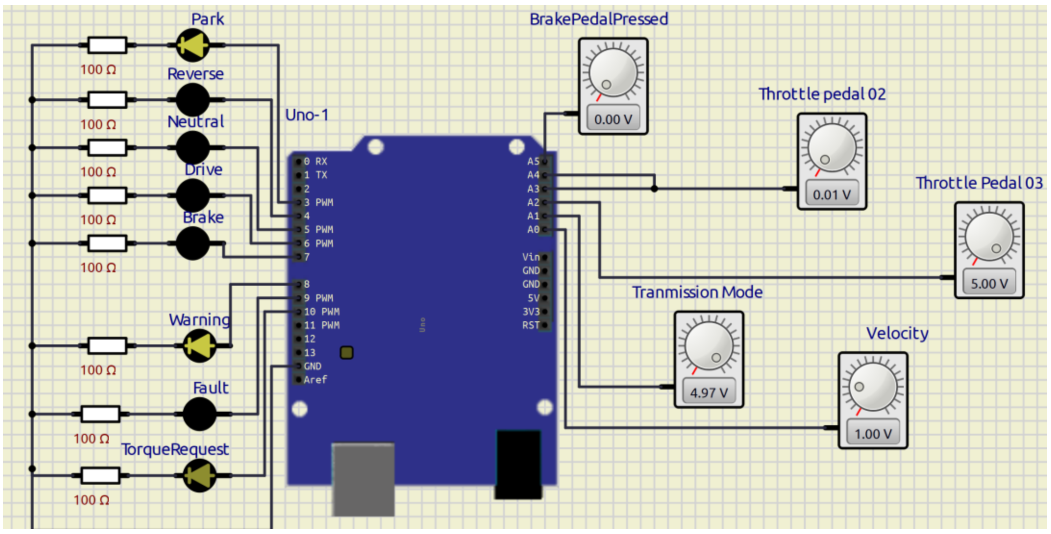

Figure 03: a) The harness used in SimulIDE, here one voltage source is used for throttle pedal signals 1 and 2, and one is used as outlier; for checking the functionality of the voter. Here Torque request is at 50% PWM which means the torque requested is 0, and as for the velocity, 1 volt means 0 velocity.

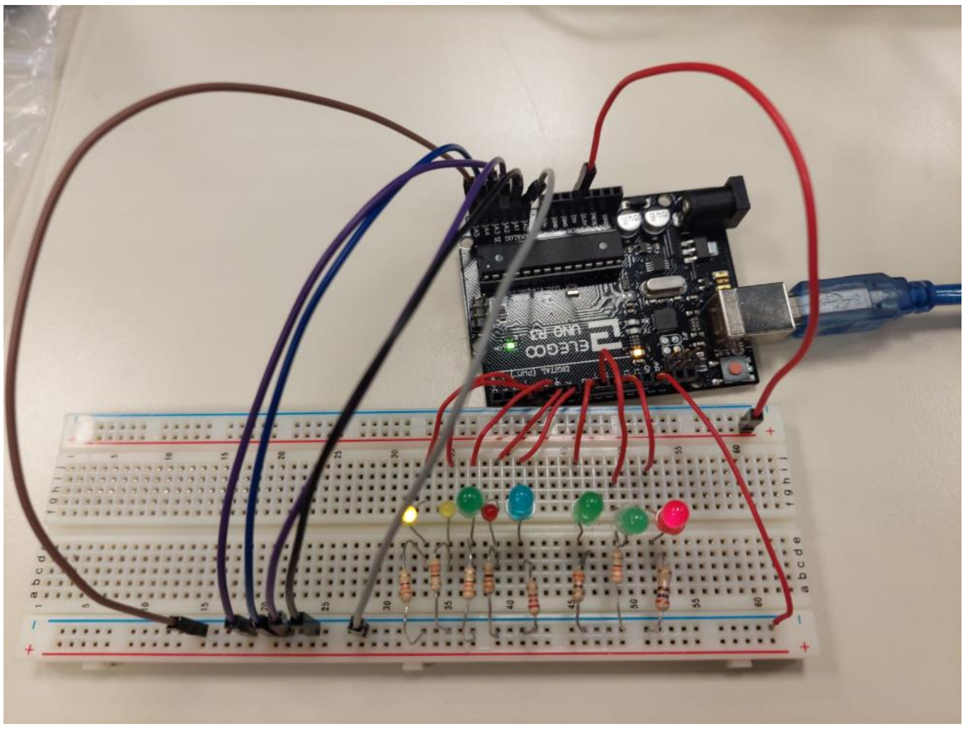

Figure 03: Arduino implementatito; since a variable voltagae sourse is not available, for input signals, 5V and GND pins of the Arduino are used.

Test stimuli

[Provide a brief description of the stimuli set chosen to test the controller functionality] Due to the limitations of the SimulIDE, provide stimuli to check the function of the one pedal controller without the whole physical model

Testing the controller chart:

To begin with, the functionality of the chart should be checked, so that we make sure that when all the signals are correct the controller have the expected functionality. To do so, all the 8 transition condition between the states of the controller was tested, two of which are described in the following.

Transition from Drive to Brake mode:
In the following figure, it can be seen that the transition to Brake mode does not happen before the ThrottlePedalPressed signal passes 1/3 of its course.

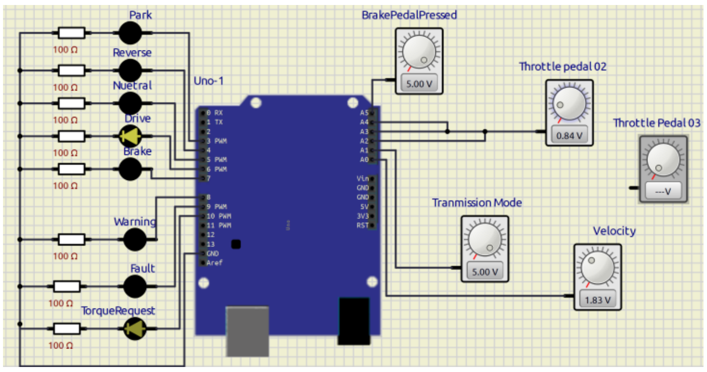

Figure 05: a) The transmission mode is set to Brake mode, but since the throttle pedal is in the first 1/3 of its course, the tranmissionState is on the Drive mode; simulation.

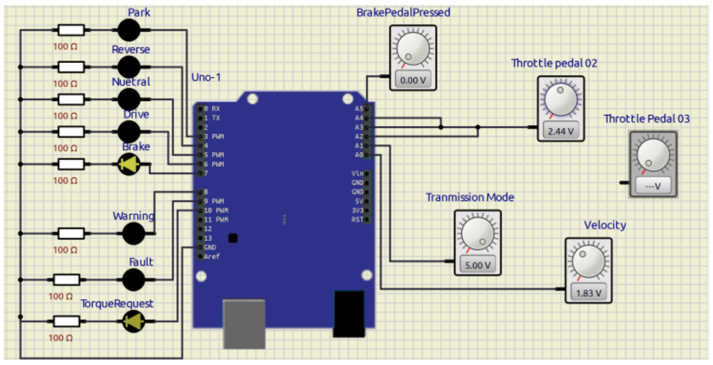

Figure 05: b) once the throttle pedal passes 1/3 of its course, which corresponds to the accreleration regione for the braking mode, the transmissionState switches to Brake mode; simulation.

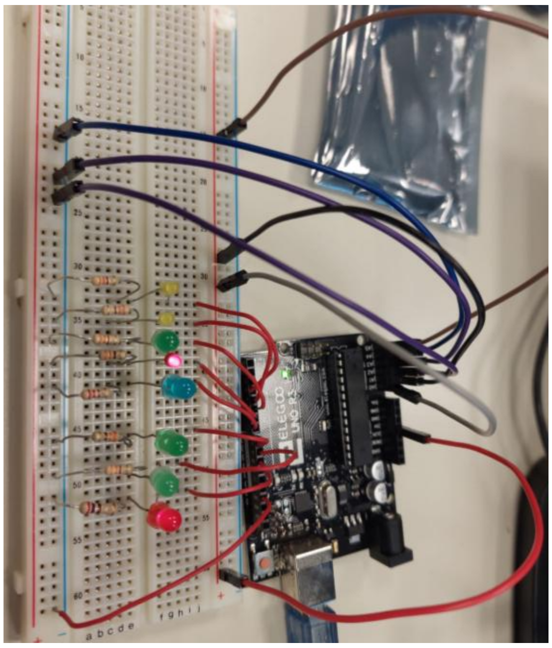

Figure 05: c) The transmission mode is set to Brake mode, but since the throttle pedal is in the first 1/3 of its course, the tranmissionState is on the Drive mode; Arduino implementation.

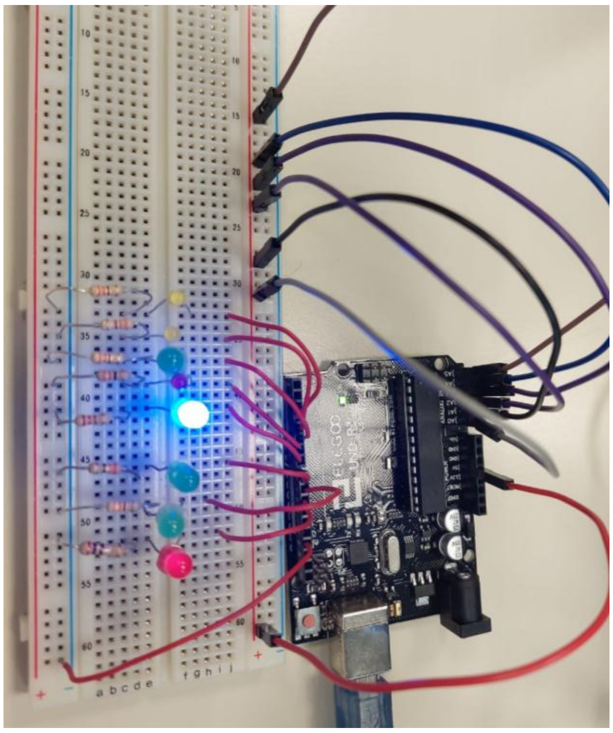

Figure 05: d) once the throttle pedal passes 1/3 of its course, which corresponds to the accreleration regione for the braking mode, the transmissionState switches to Brake mode; Arduino Implementation.

Transition from Drive to Reverse mode:
It can be seen that the transition from Drive to Reverse first passes Neutral. Further, it can be seen that this transition does not occure when the velocity of the vehicle is more than 5 km/h.

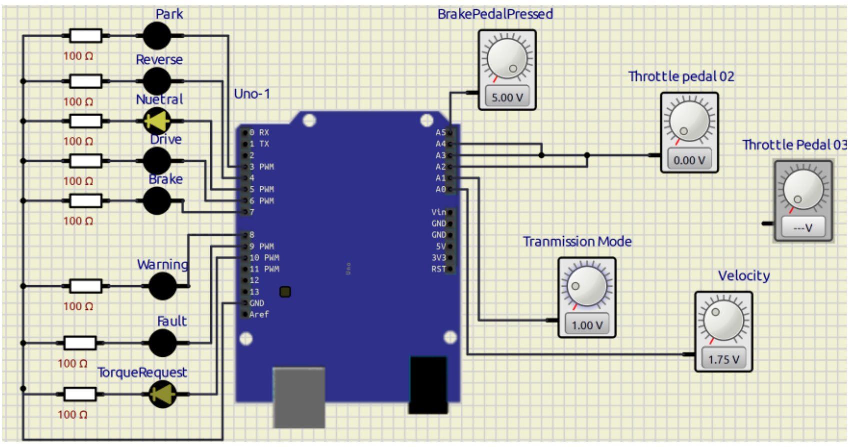

Figure 06: a) The transmission mode is set to Reverse mode, but since the velocity is more than 5 Km/h, the tranmissionState does not switch to the Reverse mode.

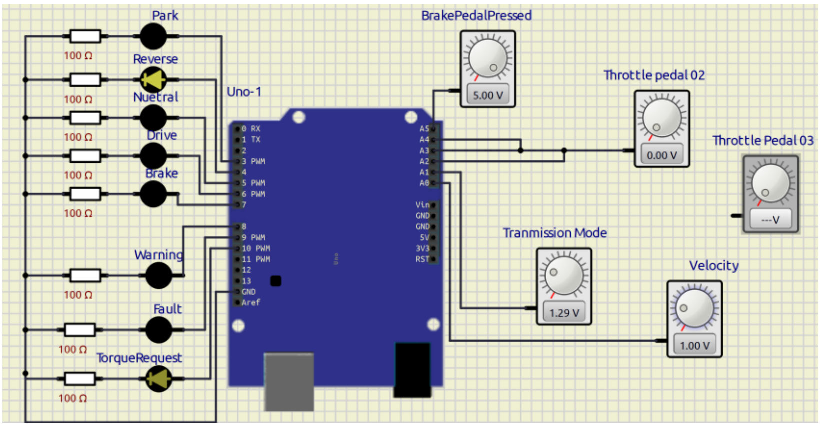

Figure 06: b) once the velocity becomes lower than 5 Km/h, the transition occurs.

Further, it can be seen that the transition to neutral can occure at any moment without any condition.

Due to lack of variable voltage sourse, this test was not performed on thet Arduino.

Testing transition between substates:
Further, the subtrasitions between the substates of the Brake, Drive, and Reverse mode are tested. Having tested substates, we are confident that the chart of the controller works as expected. Hereunder, some transitions between the substates of Brake mode are discussed.

When the transmission state changes to Brake, the default substate is acceleration region, and in this region it can be seen that by increasing the value of the throttle pedal, the PWM signal assume a larger percentage, and thereby the TorqueRequest LED becomes brighter.

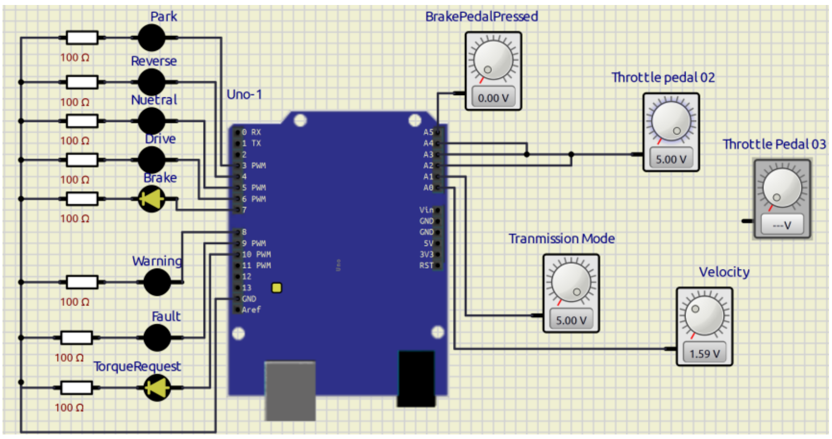

Figure 07: a) It can be seen that in the acceleration region of the Brake mode, the TorqueRequest LED becomes brighter.

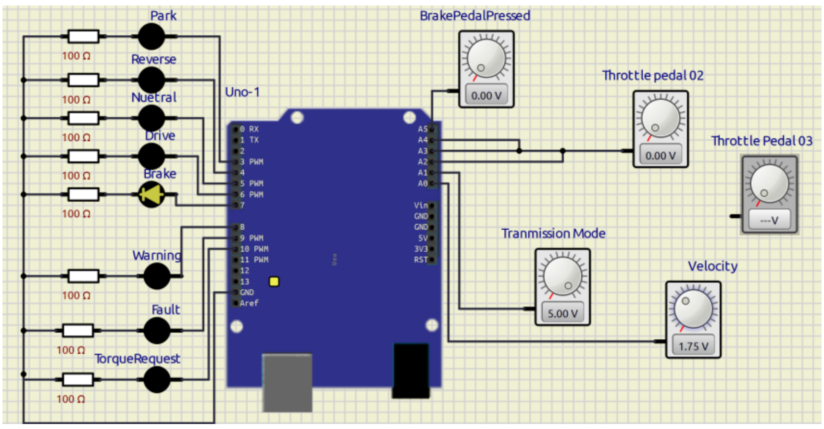

Figure 07: b) it can be seen that once the throttlepedal enter the first 1/3 of its course when the velocity is large enough, PWM assumes 0%, and hence TorqueRequest LED turns off.

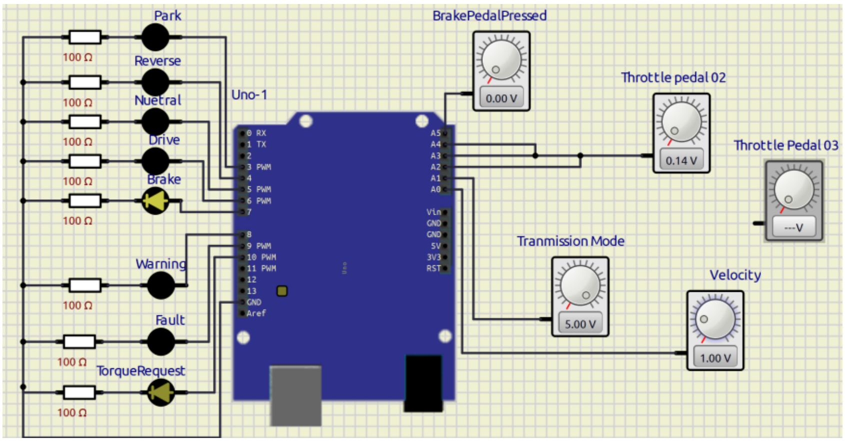

Figure 07: c) once the velocity becomes lower than 0.5 Km/h, substate change to ZeroTorque substate to prevent the vehicle from moving in the reverse direction. 0 torque correspond to 50% PWM, for this reason the LED is dim.

Further, the transition to ZeroTorque substate of the driving mode is represented, where when the brak pedal is pressed the LED becomes dim again.

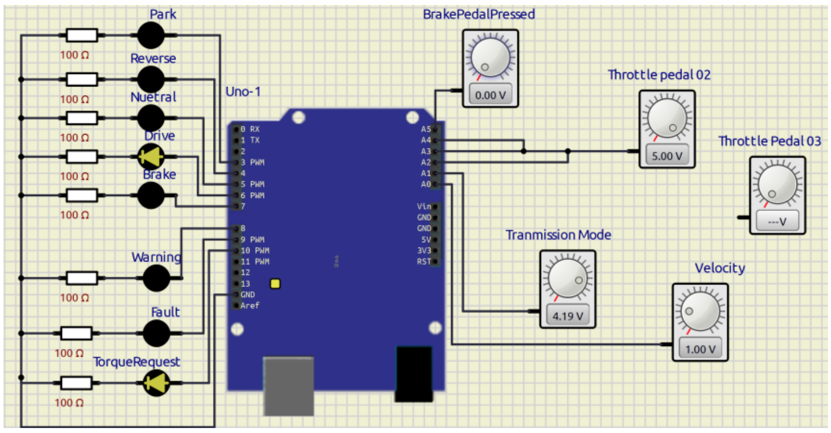

Figure 08: a) In the driving mode, the TorqueRequest LED becomes brighter corresponding the the position of the throttle pedal.

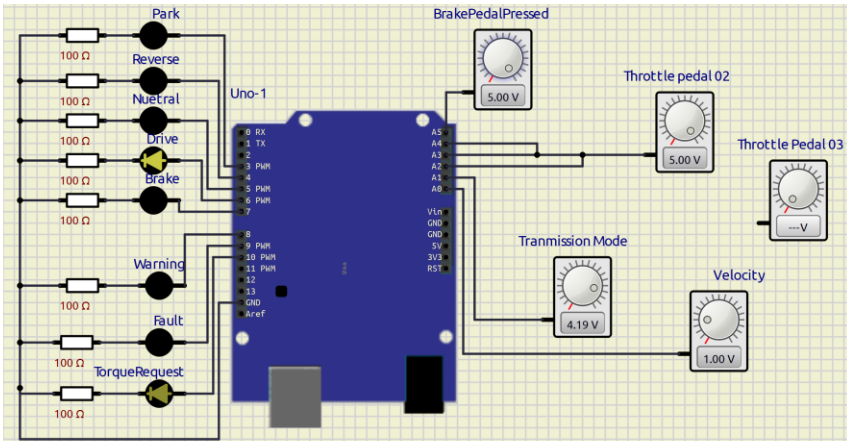

Figure 08: b) However, once the Brake pedal is pressed, regardless of the position of the throttle pedal, the TorqueRequest becomes 0, or PWM 50%.

Voter functionality:

Here, it is tested that if one of the sensors are outlier, the voter is able to distinguish the outlier sensor and average the two incoming signal from the functioning sensors.

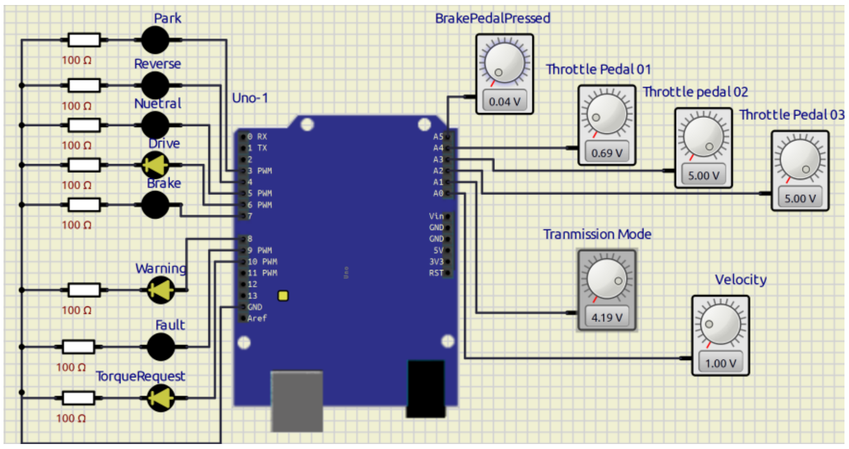

Figure 09: a) It can be seen that with one outlier, the controller works as expected, while the warning LED turns on; Here, the first sensor is the outlier.

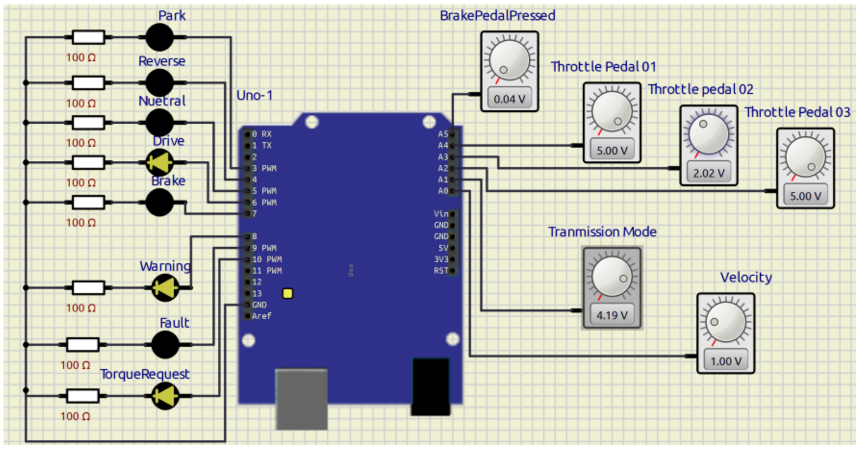

Figure 09: b) It can be seen that with one outlier, the controller works as expected, while the warning LED turns on; Here, the second sensor is the outlier.

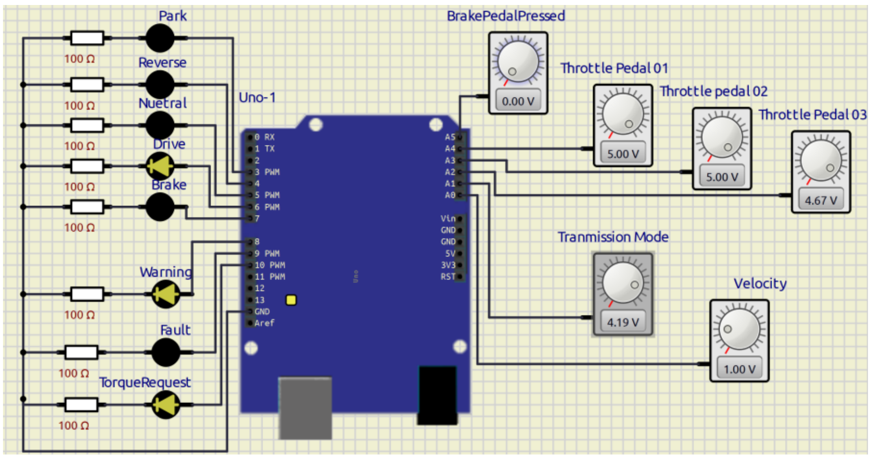

Figure 09: c) It can be seen that with one outlier, the controller works as expected, while the warning LED turns on; Here, the third sensor is the outlier.

The following pictures of the test done on the Arduino hardware to check the functionality of the voter. In this case, while three sensors are connected to 5V pin, warning signal is off, and while one of them is not connected, the LED turns on.

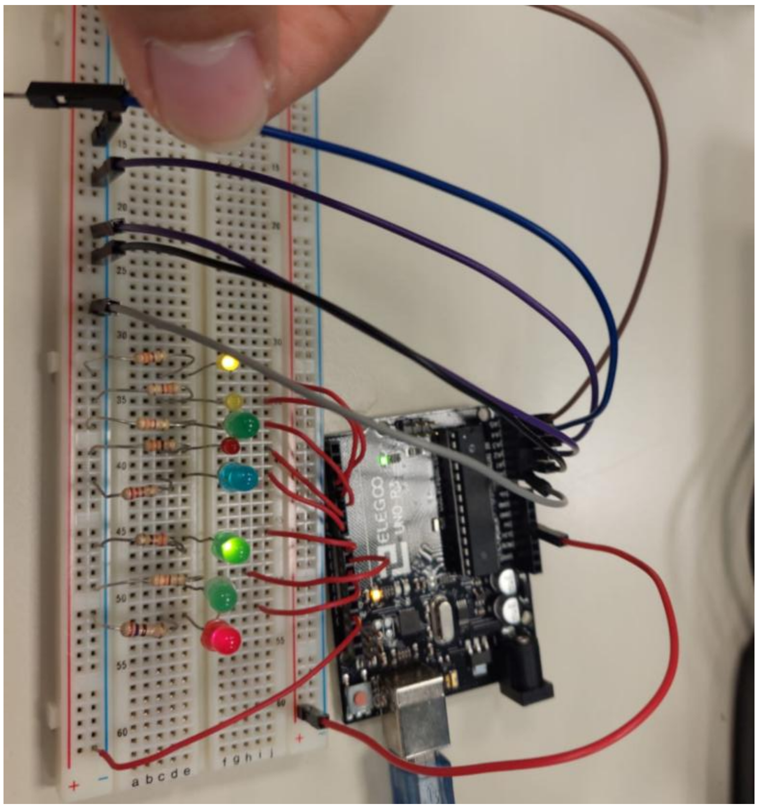
Figure 10: a) It can be seen that with one outlier, the controller works as expected, while the warning LED turns on; Here, the first sensor is the outlier.

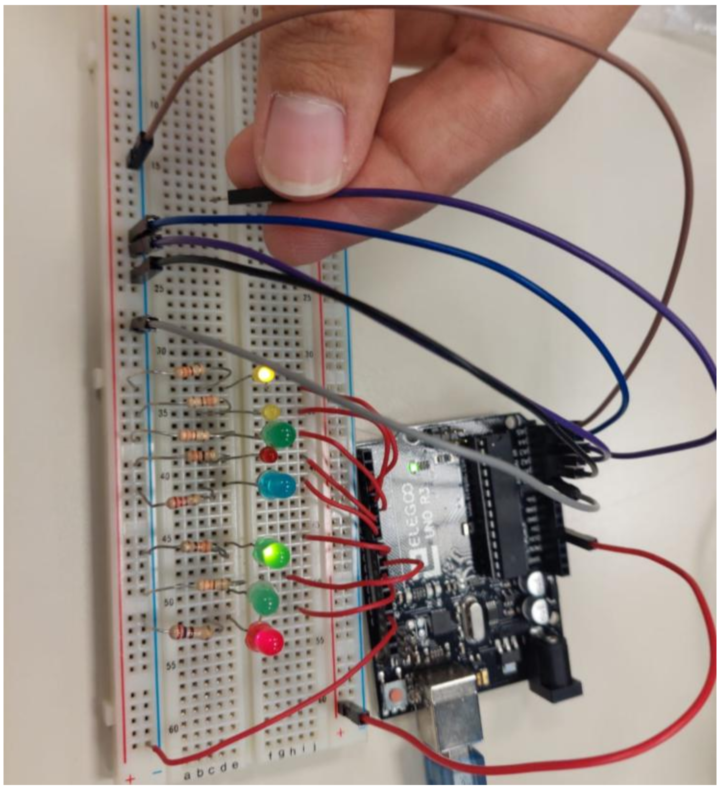
Figure 10: b) It can be seen that with one outlier, the controller works as expected, while the warning LED turns on; Here, the second sensor is the outlier.

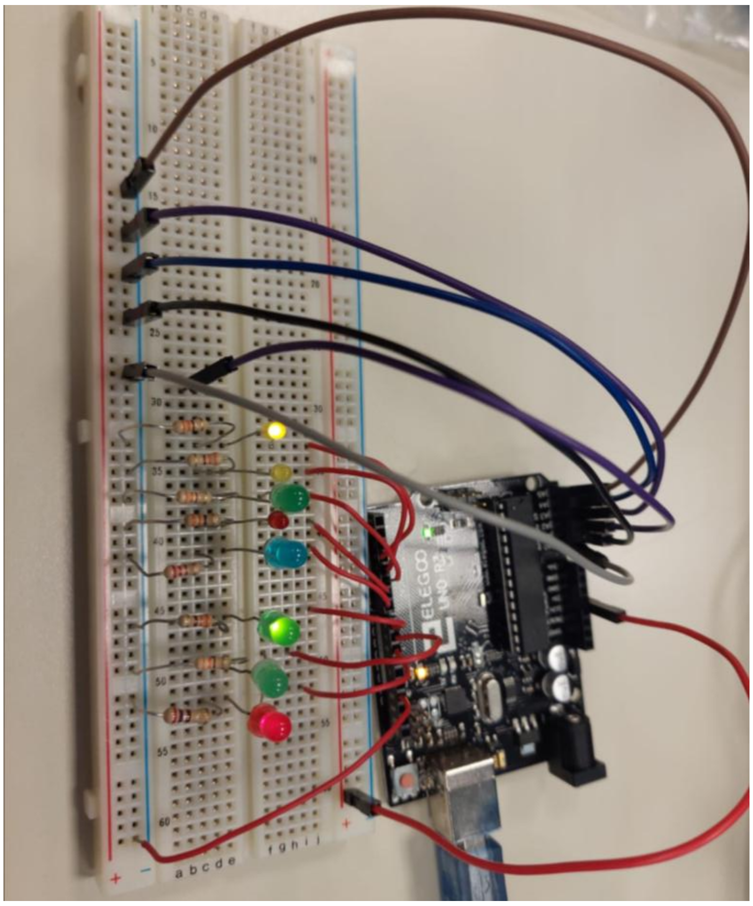

Figure 10: c) It can be seen that with one outlier, the controller works as expected, while the warning LED turns on; Here, the third sensor is the outlier.

In this part, it is represented that if all the sensors send signals different from one another, the transmission state changes to Neutral and the fault LED is turned on.

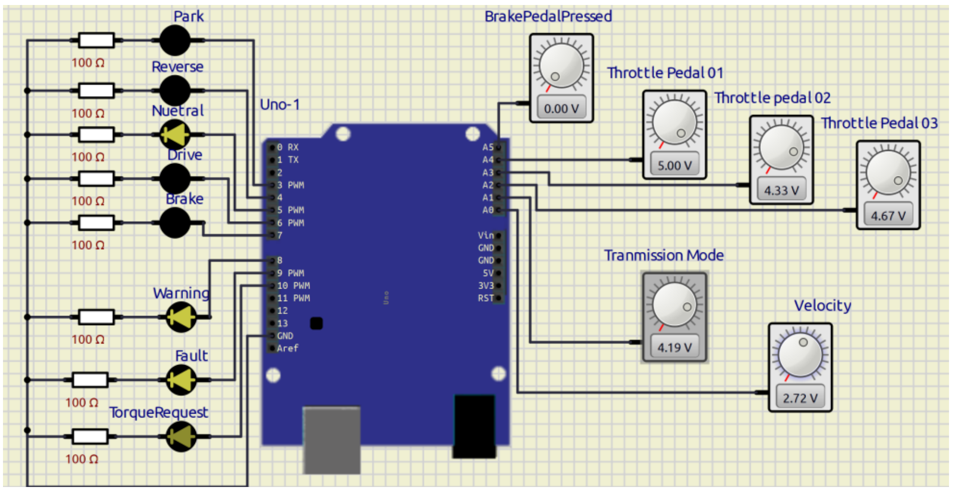

Figure 11: a) It can be seen that with sensors sending different signals, the transmission state changes to Neutral and the Warning and Fault signals are also turned on to inform the driver; simulation

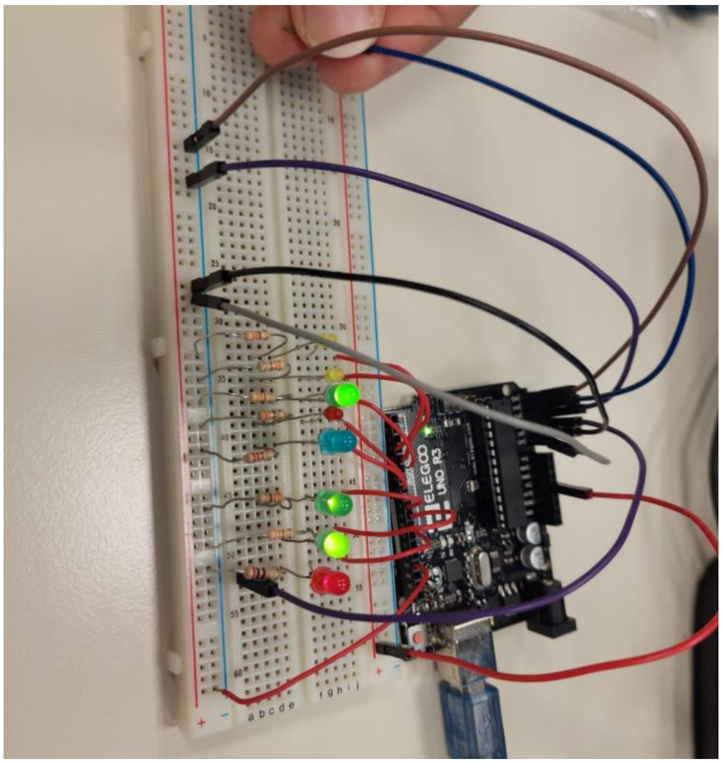

Figure 11: b) It can be seen that with sensors sending different signals, the transmission state changes to Neutral and the Warning and Fault signals are also turned on to inform the driver; Arduino impolementation.