MBSD Lab #4 A.Y. 2022/23

# Purposes

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

# Instructions

For instruction on how to use SimulIDE and the Platform Support Packages follow the instruction provided by Prof. Violante in the lecture of Thursday 11th May 2023.

The delivery shall contains:

* The controller Simulink model used to generate the firmware binary file (plus all the accompanying files needed to make it possible to generate again the code, 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.

It 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 11th at 23:59.** It shall also contain a brief report explaining integration process using the following template. It is sufficient that only one of the group members uploads it.

# Model-Based Software Design, A.Y. 2022/23

# Laboratory 4 Report

## Components of the working group (max 2 people)

* Simone Bergadano, S303053
* Pietro Vignini, S317465

# I/O interfaces

To interact with the single-pedal controller we used the available pins of the Arduino UNO board as follows:

* For analog input signals, such as the three readings for the pedal position and the vehicle speed, we used the analog input pins, converting for each case the ADC output (0 - 1023) into suitable ranges.
* For boolean input signals we used a digital input pin. The only exception is the information on the brake pedal pressed, which we supplied via analog input and then converted to boolean due to lack of digital pins.
* To provide the information about the automatic transmission selector state we used an analog input pin, dividing the input voltage range into 5 equal intervals, and making each one correspond to a selector position.
* We used digital output pins to handle digital output signals, such as the pedal errors, the availability information, and the negative torque flag.
* For the analog output signals, which are the torque request, the validated throttle pedal position, and the automatic transmission state, we used digital output pins with pulse-width modulation (PWM), after properly converting the output ranges into the duty cycle range (0-255).

In the case of the automatic transmission state we then used a low-pass filter on simulIDE to “convert” the signal to a DC voltage with module proportional to the duty cycle.

The information about the I/O interfaces are collected in the Table below, which also reports the conversion formulas. The Figures below show the the controller model ready for the code generation and how we implemented some of the conversion formulas on Simulink.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Unit** | **Type** | **Conversion formulas** | **Min** | **Max** |
| ThrottlePedalPosition\_1 | - | Analog Input |  | 0 (0V) | 1 (5V) |
| ThrottlePedalPosition\_2 | - | Analog Input |  | 0 (0V) | 1 (5V) |
| ThrottlePedalPosition\_3 | - | Analog Input |  | 0 (0V) | 1 (5V) |
| BrakePedal  Pressed | - | Analog Input |  | 0 (0V) | 1 (5V) |
| Automatic Transmission SelectorState | Enum.  {P, R, N, D, B} | Analog Input |  | 0 (0V) | 4 (5V) |
| Vehicle\_Speed\_available | - | Digital Input |  | 0V | 5V |
| ATSelector  State\_  Availability | - | Digital Input |  | 0V | 5V |
| Vehicle\_Speed\_km\_h | Km/h | Analog Input |  | -60 km/h (0V) | 240 km/h (5V) |
| TorqueRequest\_Nm[[1]](#footnote-2) | Nm | Digital Output (PWM) |  | 0 Nm (0%) | 80 Nm  (100%) |
| TorqueRequest\_negative1 | - | Digital Output |  | 0V | 5V |
| Automatic  Transmission  State | Enum.  {P, R, N, D, B} | Digital Output (PWM) |  | 0 (P) | 4 (B) |
| Vehicle\_Speed\_Availability | - | Digital Output |  | 0V | 5V |
| Pedal\_  position\_reading\_warning | - | Digital Output |  | 0V | 5V |
| Pedal\_  position\_reading\_error | - | Digital Output |  | 0V | 5V |
| B\_Mode\_  available | - | Digital Output |  | 0V | 5V |
| Validated\_  ThrottlePedal | - | Digital Output (PWM) |  | 0 (0%) | 1 (100%) |
| ATSelector  State\_  Available | - | Digital Output |  | 0V | 5V |

*Immagine che contiene testo, diagramma, linea, Piano

Descrizione generata automaticamente*

Figure 1 Screenshot of the controller model ready for the code generation.

*Immagine che contiene diagramma, schizzo, Disegno tecnico, disegno

Descrizione generata automaticamente*

Figure 2 Conversion formula for the vehicle speed data implemented on Simulink. Immagine che contiene testo, diagramma, linea, Carattere

Descrizione generata automaticamente

Figure 3 Conversion formula for the torque request implemented on Simulink.

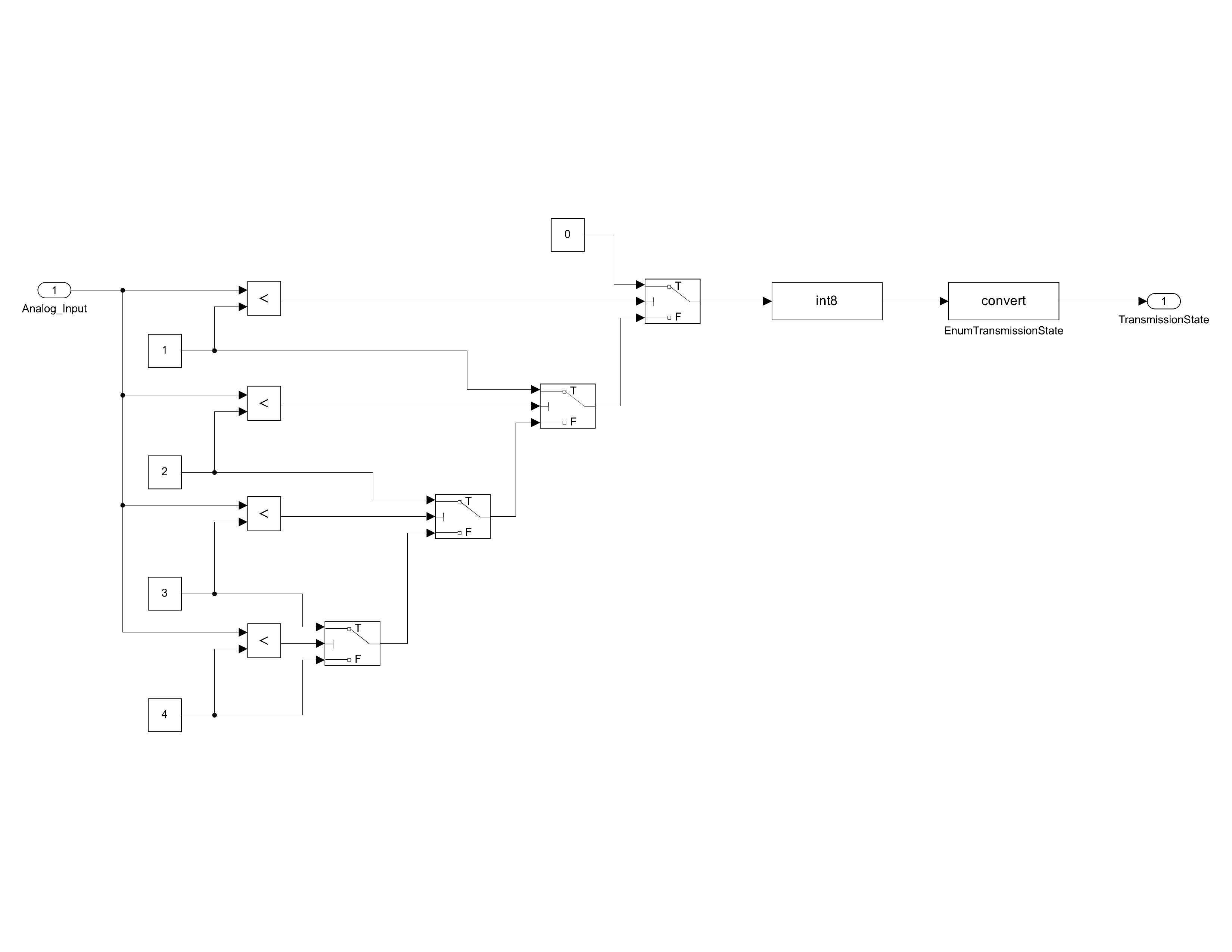
**

Figure 4 Conversion formula for the automatic transmission selector state input implemented on Simulink.

# Code generation for Arduino

To generate the Arduino firmware, we started by adding the block from the Simulink Support Package for Arduino Hardware for each of the input and output pins that we decided to use to interact with the one pedal controller. The result can be seen in Figure 5.

Then, we set the Simulink solver to discrete time fixed step with an integration step of 0.1 seconds, and we chose Arduino Uno as Hardware board in the Hardware implementation settings. In the Code generation settings, we chose the Faster Runs as Build configuration.

Finally, we generated the firmware for the Arduino board (.hex file). We then loaded it in SimulIDE.

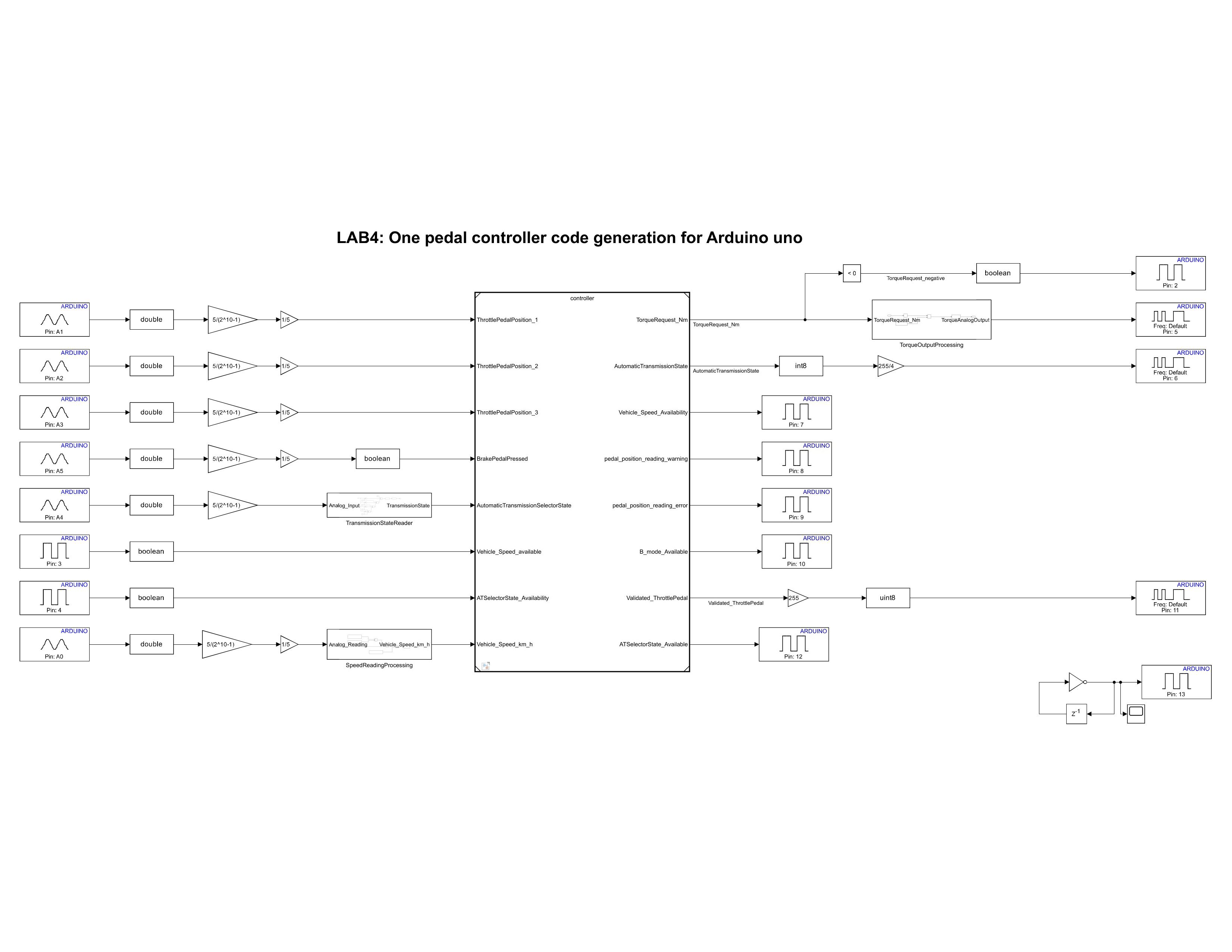


Figure 2 Screenshot of the controller model instrumented with the blocks of the Support Package for Arduino Hardware.

# Harness

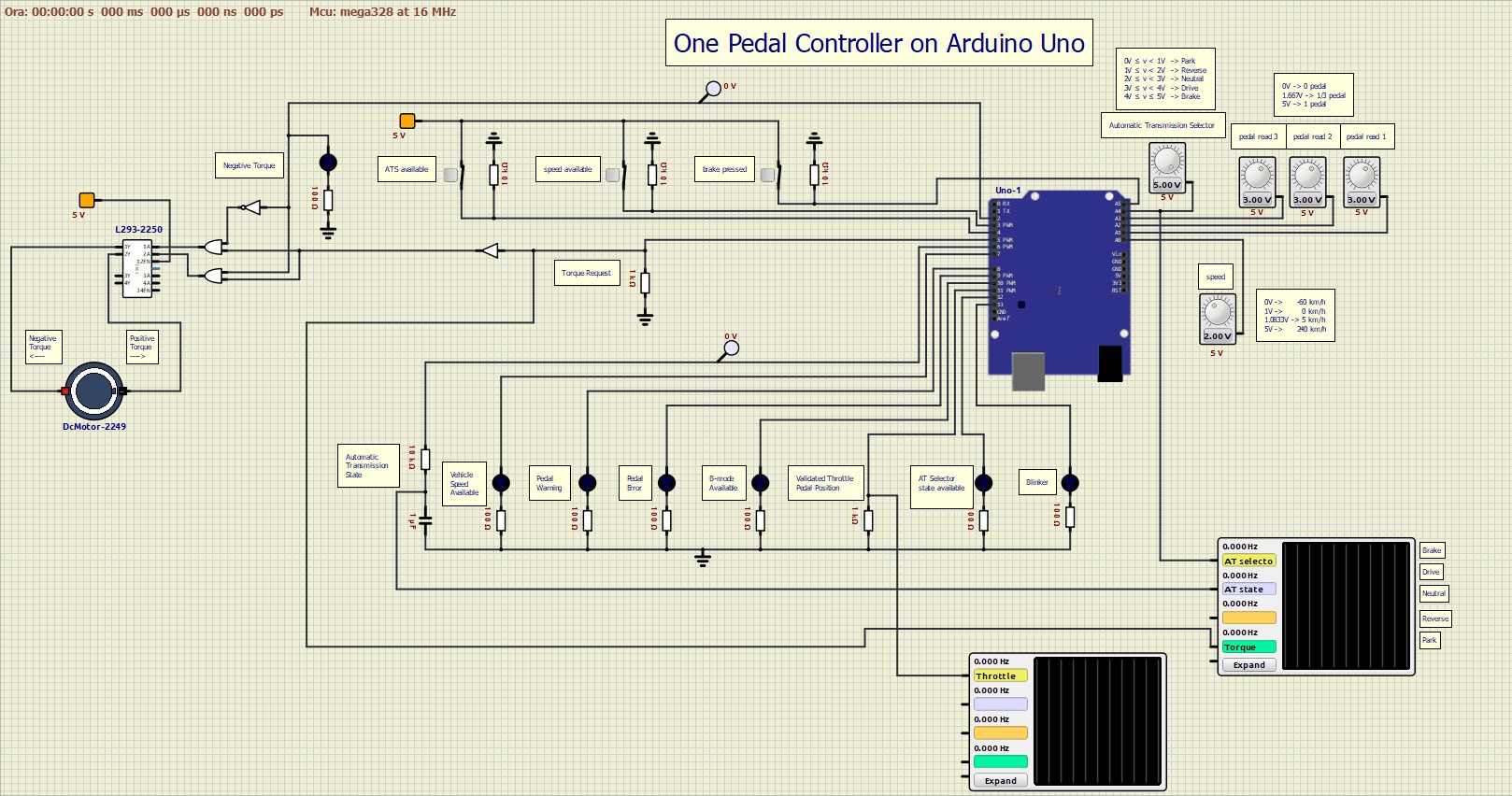
**

Figure 3 Screenshot of the harness implemented in SimulIDE.

## Test stimuli

In the two tables below, we reported some of the tests that we performed to evaluate the controller functionality. We divided the tests into two groups: in the first one we assessed the correct behavior of the controller when there are no pedal position reading errors. Here we tested all the driving modes and the response of the system when the vehicle speed and the automatic transmission selector state information are not available.

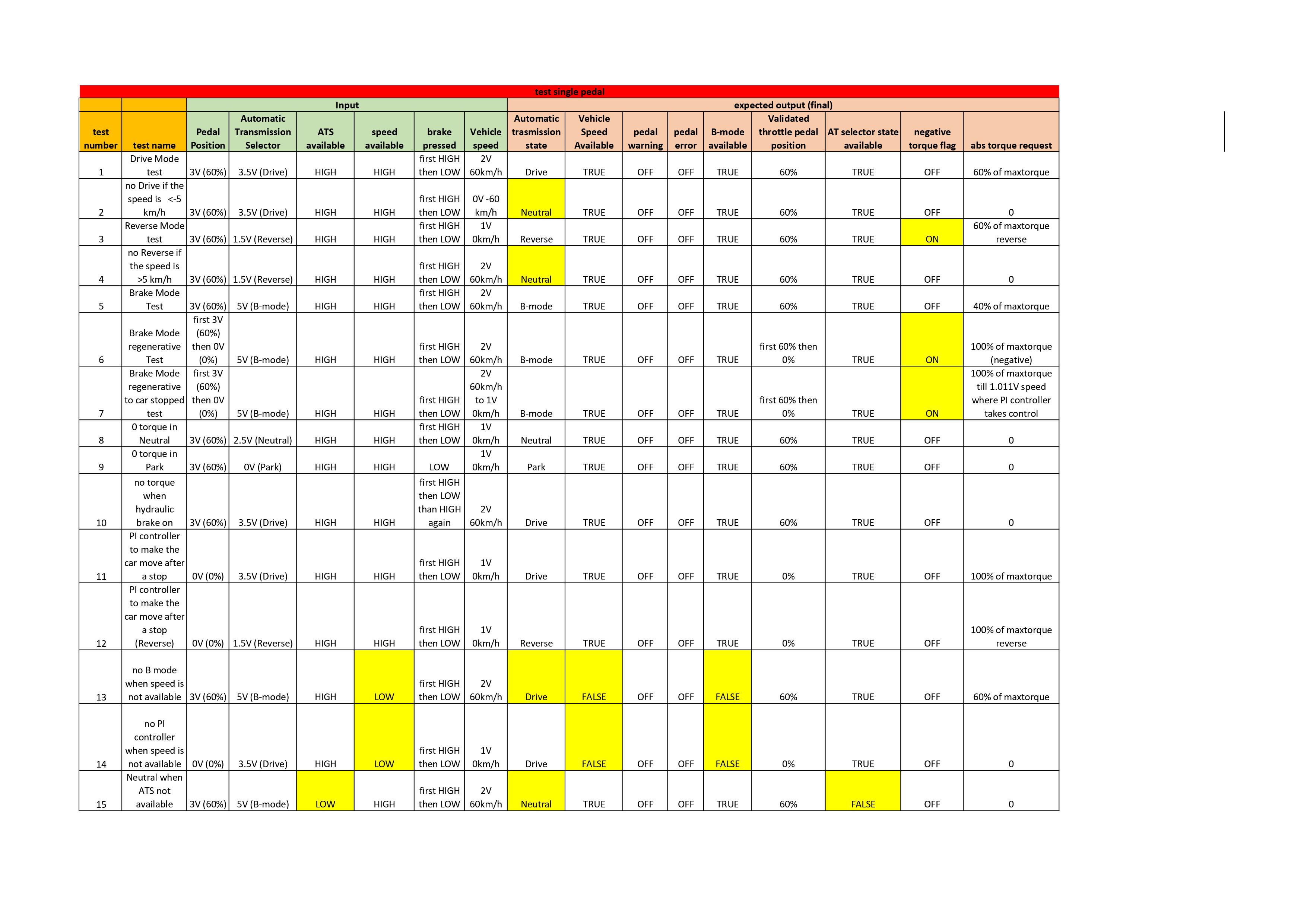
To be able to have the same pedal reading on the three analog input pins we used just one voltage source for all three, as follow:

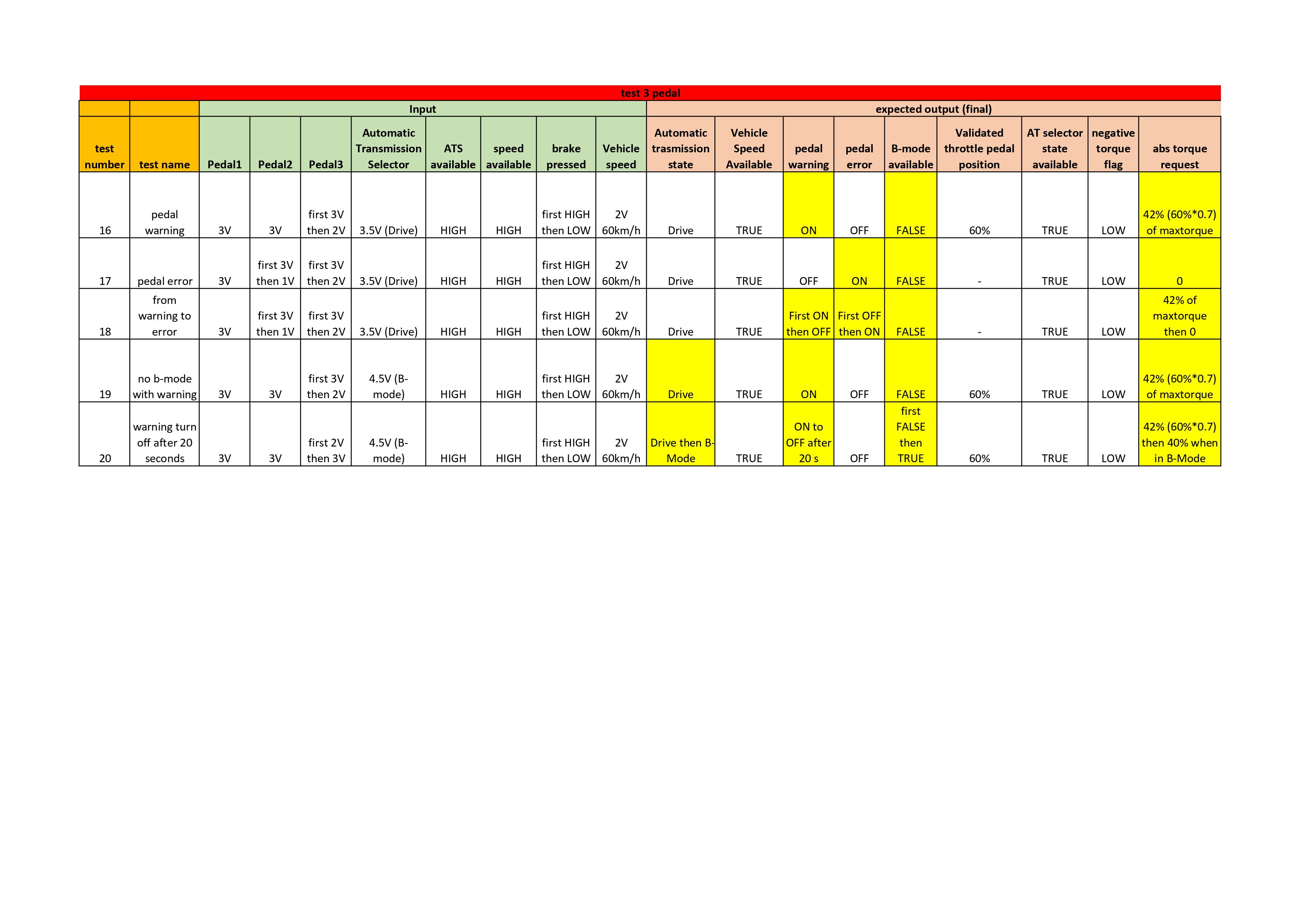
Immagine che contiene testo, schermata, diagramma, Rettangolo

Descrizione generata automaticamente

In the second group of tests we evaluated the behavior of the system in case of one or more pedal positions reading mismatches. For these tests we used the configuration shown in Figure 5 to have three separate voltage sources.

All tests in both tables were passed successfully.





NB: We have noticed that a bug showing the maximum torque request in situations where it should be zero can sometimes occur when running the circuit on SimulIDE. When it happens, restarting the program usually fixes the problem. We have tested this bug thoroughly and we are certain that it is not a controller problem, but something related to SimulIDE, even more considering that restarting the application makes everything work fine.

1. To better control the motor, we decided to divide the torque request into two components: the absolute value, which is communicated through a PWM digital pin, and the sign, which instead is a Boolean variable and is communicated with a digital pin. [↑](#footnote-ref-2)