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
`ThrottlePedal Position_1`	-	Analog Input	$p_1(v) = \frac{1}{5} * v$ $p_1 = \frac{1}{5} * \frac{5}{2^{10} - 1} * ADC_{out}$	0 (0V)	1 (5V)

ThrottlePedal Position_2	-	Analog Input	$p_2(v) = \frac{1}{5} * v$ $p_2 = \frac{1}{5} * \frac{5}{2^{10}-1} * ADC_{out}$	0 (0V)	1 (5V)
ThrottlePedal Position_3	-	Analog Input	$p_3(v) = \frac{1}{5} * v$ $p_3 = \frac{1}{5} * \frac{5}{2^{10}-1} * ADC_{out}$	0 (0V)	1 (5V)
BrakePedal Pressed	-	Analog Input	$b(v) = \begin{cases} 0, & v = 0V \\ 1, & v = 5V \end{cases}$	0 (0V)	1 (5V)
Automatic Transmission SelectorState	Enum. {P, R, N, D, B}	Analog Input	$t(v) = \begin{cases} 0\ (P), & 0V\ \leq v < 1V \\ 1\ (R), & 1V\ \leq v < 2V \\ 2\ (N), & 2V\ \leq v < 3V \\ 3\ (D), & 3V\ \leq v < 4V \\ 4\ (B), & 4V\ \leq v \leq 5V \end{cases}$ $v = \frac{5}{2^{10}-1} * ADC_{out}$	0 (0V)	4 (5V)
Vehicle_Speed _available	-	Digital Input	$5V \rightarrow available,$ $0V\ \rightarrow not\ available$	0V	5V
ATSelector State_ Availability	-	Digital Input	$5V \rightarrow available,$ $0V\ \rightarrow not\ available$	0V	5V
Vehicle_Speed _km_h	Km/h	Analog Input	$s(v) = \frac{300}{5} * v - 60\ km/h$ $s = \frac{300}{5} * \frac{5}{2^{10}-1} * ADC_{out} - 60$	-60 km/h (0V)	240 km/h (5V)
TorqueRequest _Nm[1]	Nm	Digital Output (PWM)	$D_{\%} = \frac{100}{255} * uint8(\frac{255}{80} * \lvert torque_{req} \rvert)$	0 Nm (0%)	80 Nm (100%)
TorqueRequest _negative[1]	-	Digital Output	$torque_{req}\ < 0\ \rightarrow true$ $torque_{req}\ \geq 0\ \rightarrow false$	0V	5V
Automatic Transmission State	Enum. {P, R, N, D, B}	Digital Output (PWM)	$D_{\%} = \begin{cases} 0\%, & Park \\ 25\%, & Reverse \\ 50\%, & Neutral \\ 75\%, & Drive \\ 100\%, & Brake \end{cases}$	0 (P)	4 (B)
Vehicle_Speed _Availability	-	Digital Output	$5V \rightarrow available,$ $0V\ \rightarrow not\ available$	0V	5V
Pedal_ position_read ing_warning	-	Digital Output	$5V \rightarrow warning\ ON,$ $0V\ \rightarrow warning\ OFF$	0V	5V

[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.

Pedal_ position_reading_error	-	Digital Output	$5V \rightarrow error\ ON,$ $0V\ \rightarrow error\ OFF$	0V	5V
B_Mode_ available	-	Digital Output	$5V \rightarrow available,$ $0V\ \rightarrow not\ available$	0V	5V
Validated_ ThrottlePedal	-	Digital Output (PWM)	$D_{\%} = \frac{100}{255} * uint8(255 * Valid\_Throttle)$	0 (0%)	1 (100%)
ATSelector State_ Available	-	Digital Output	$5V \rightarrow available,$ $0V\ \rightarrow not\ available$	0V	5V

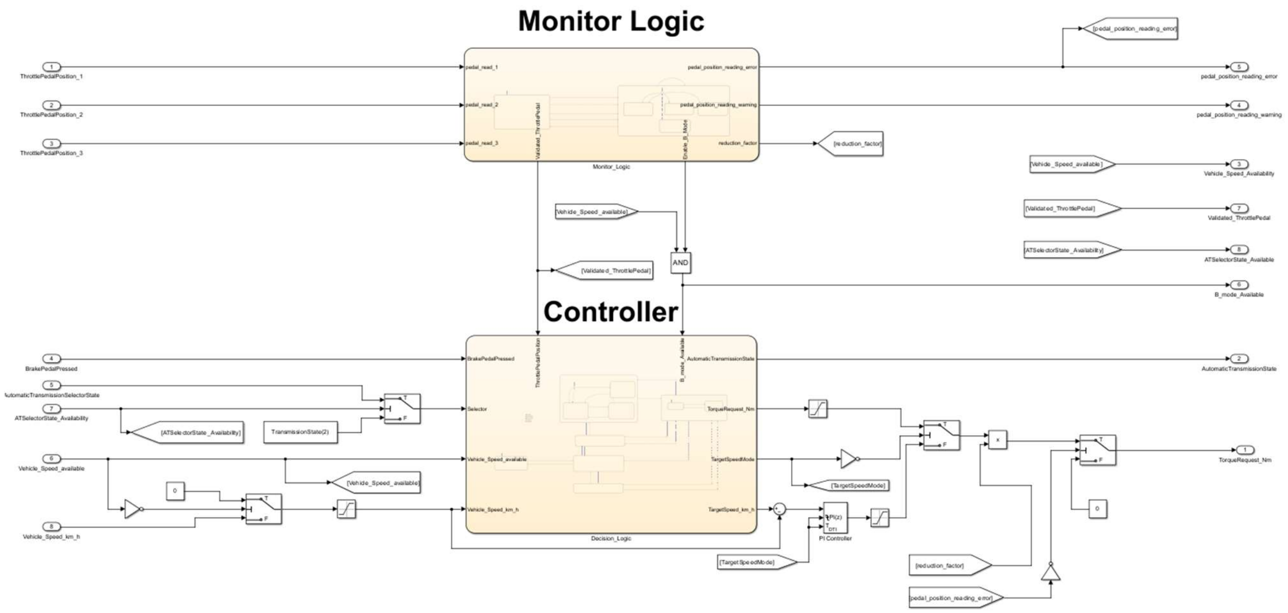


Figure 1 Screenshot of the controller model ready for the code generation.

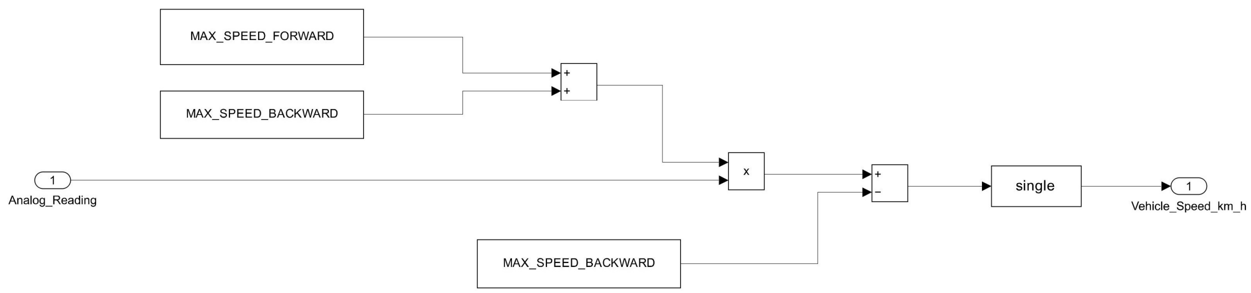


Figure 2 Conversion formula for the vehicle speed data implemented on Simulink.

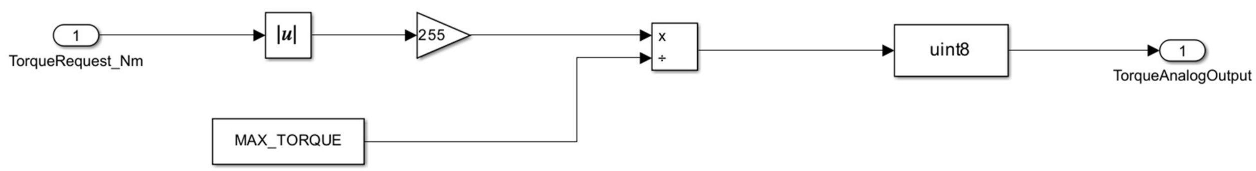


Figure 3 Conversion formula for the torque request implemented on Simulink.

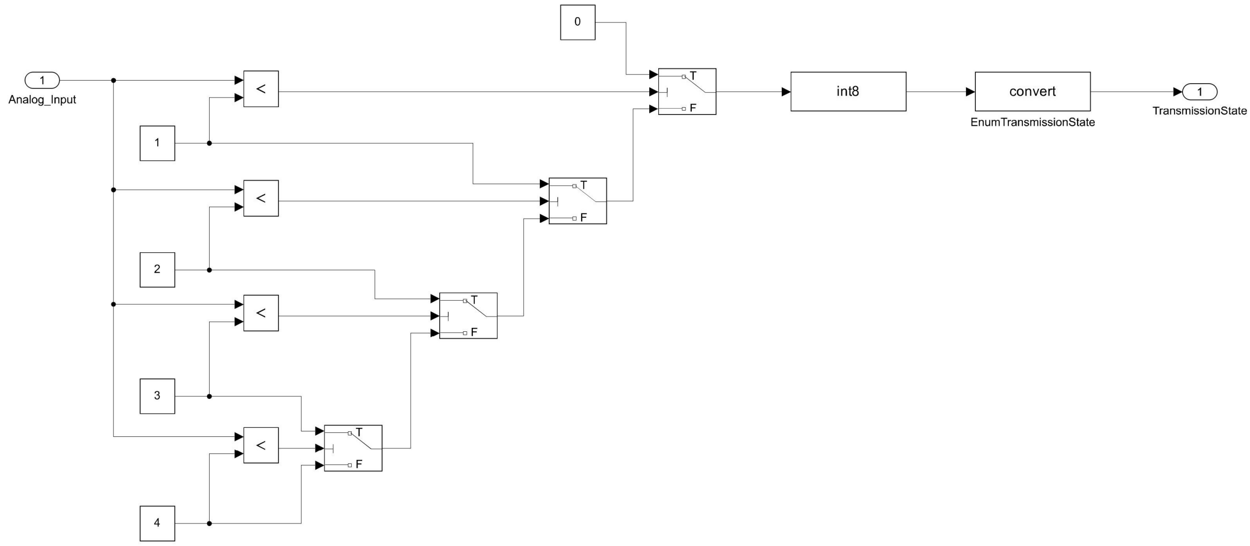


Figure 4 Conversion formula for the automatic transmission selector state input implemented on Simulink.

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.

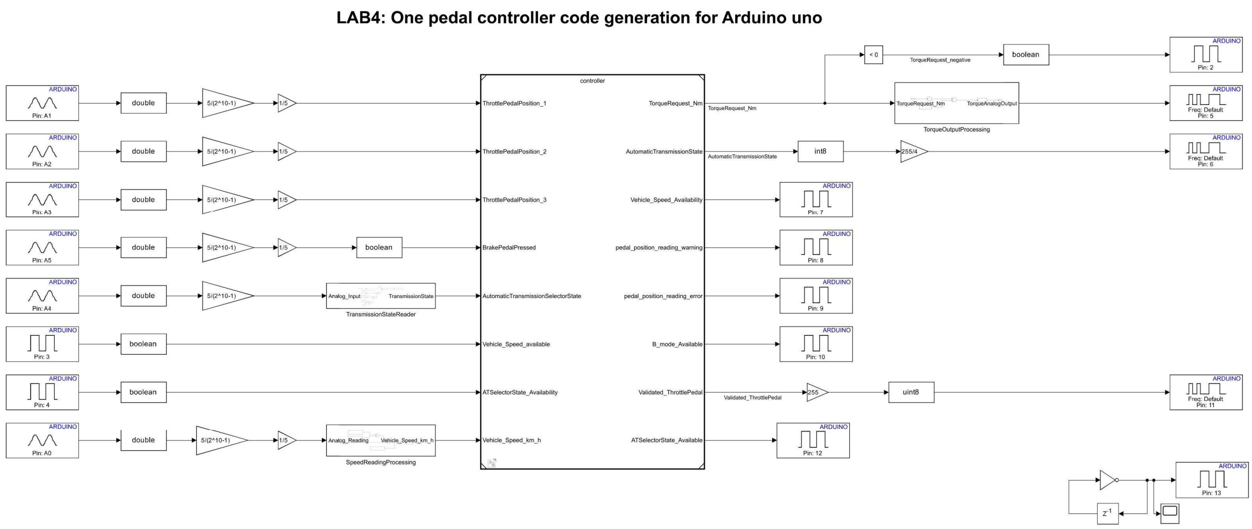


Figure 2 Screenshot of the controller model instrumented with the blocks of the Support Package for Arduino Hardware.

Harness

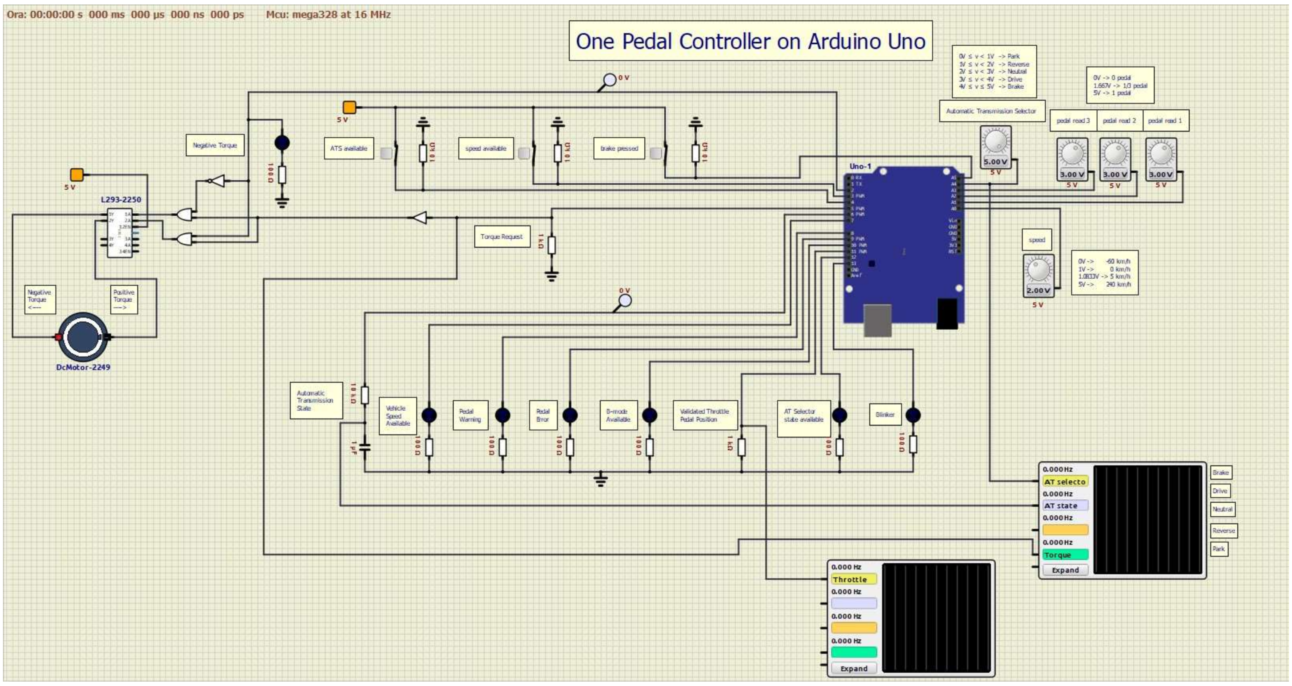
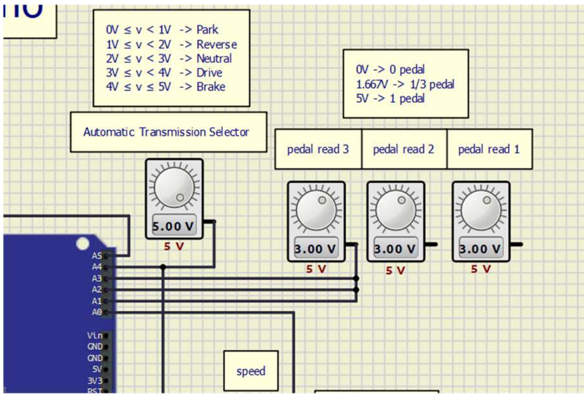


Figure 3 Screenshot of the harness implemented in SimulIDE.

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:



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.

test single pedal																
		Input						expected output (final)								
test number	test name	Pedal Position	Automatic Transmission Selector	ATS available	speed available	brake pressed	Vehicle speed	Automatic trasmission state	Vehicle Speed Available	pedal warning	pedal error	B-mode available	Validated throttle pedal position	AT selector state available	negative torque flag	abs torque request
1	Drive Mode test	3V (60%)	3.5V (Drive)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	Drive	TRUE	OFF	OFF	TRUE	60%	TRUE	OFF	60% of maxtorque
2	no Drive if the speed is <-5 km/h	3V (60%)	3.5V (Drive)	HIGH	HIGH	first HIGH then LOW	0V -60 km/h	Neutral	TRUE	OFF	OFF	TRUE	60%	TRUE	OFF	0
3	Reverse Mode test	3V (60%)	1.5V (Reverse)	HIGH	HIGH	first HIGH then LOW	1V 0km/h	Reverse	TRUE	OFF	OFF	TRUE	60%	TRUE	ON	60% of maxtorque reverse
4	no Reverse if the speed is >5 km/h	3V (60%)	1.5V (Reverse)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	Neutral	TRUE	OFF	OFF	TRUE	60%	TRUE	OFF	0
5	Brake Mode Test	3V (60%)	5V (B-mode)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	B-mode	TRUE	OFF	OFF	TRUE	60%	TRUE	OFF	40% of maxtorque
6	Brake Mode regenerative Test	first 3V (60%) then 0V (0%)	5V (B-mode)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	B-mode	TRUE	OFF	OFF	TRUE	first 60% then 0%	TRUE	ON	100% of maxtorque (negative)
7	Brake Mode regenerative to car stopped test	first 3V (60%) then 0V (0%)	5V (B-mode)	HIGH	HIGH	first HIGH then LOW	2V 60km/h to 1V 0km/h	B-mode	TRUE	OFF	OFF	TRUE	first 60% then 0%	TRUE	ON	100% of maxtorque till 1.011V speed where PI controller takes control
8	0 torque in Neutral	3V (60%)	2.5V (Neutral)	HIGH	HIGH	first HIGH then LOW	1V 0km/h	Neutral	TRUE	OFF	OFF	TRUE	60%	TRUE	OFF	0
9	0 torque in Park	3V (60%)	0V (Park)	HIGH	HIGH	LOW	1V 0km/h	Park	TRUE	OFF	OFF	TRUE	60%	TRUE	OFF	0
10	no torque when hydraulic brake on	3V (60%)	3.5V (Drive)	HIGH	HIGH	first HIGH then LOW than HIGH again	2V 60km/h	Drive	TRUE	OFF	OFF	TRUE	60%	TRUE	OFF	0
11	PI controller to make the car move after a stop	0V (0%)	3.5V (Drive)	HIGH	HIGH	first HIGH then LOW	1V 0km/h	Drive	TRUE	OFF	OFF	TRUE	0%	TRUE	OFF	100% of maxtorque
12	PI controller to make the car move after a stop (Reverse)	0V (0%)	1.5V (Reverse)	HIGH	HIGH	first HIGH then LOW	1V 0km/h	Reverse	TRUE	OFF	OFF	TRUE	0%	TRUE	OFF	100% of maxtorque reverse
13	no B mode when speed is not available	3V (60%)	5V (B-mode)	HIGH	LOW	first HIGH then LOW	2V 60km/h	Drive	FALSE	OFF	OFF	FALSE	60%	TRUE	OFF	60% of maxtorque
14	no PI controller when speed is not available	0V (0%)	3.5V (Drive)	HIGH	LOW	first HIGH then LOW	1V 0km/h	Drive	FALSE	OFF	OFF	FALSE	0%	TRUE	OFF	0
15	Neutral when ATS not available	3V (60%)	5V (B-mode)	LOW	HIGH	first HIGH then LOW	2V 60km/h	Neutral	TRUE	OFF	OFF	TRUE	60%	FALSE	OFF	0

test 3 pedal																		
		Input								expected output (final)								
test number	test name	Pedal1	Pedal2	Pedal3	Automatic Transmission Selector	ATS available	speed available	brake pressed	Vehicle speed	Automatic trasmission state	Vehicle Speed Available	pedal warning	pedal error	B-mode available	Validated throttle pedal position	AT selector state available	negative torque flag	abs torque request
16	pedal warning	3V	3V	first 3V then 2V	3.5V (Drive)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	Drive	TRUE	ON	OFF	FALSE	60%	TRUE	LOW	42% (60%*0.7) of maxtorque
17	pedal error	3V	first 3V then 1V	first 3V then 2V	3.5V (Drive)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	Drive	TRUE	OFF	ON	FALSE	-	TRUE	LOW	0
18	from warning to error	3V	first 3V then 1V	first 3V then 2V	3.5V (Drive)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	Drive	TRUE	First ON then OFF	First OFF then ON	FALSE	-	TRUE	LOW	42% of maxtorque then 0
19	no b-mode with warning	3V	3V	first 3V then 2V	4.5V (B-mode)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	Drive	TRUE	ON	OFF	FALSE	60%	TRUE	LOW	42% (60%*0.7) of maxtorque
20	warning turn off after 20 seconds	3V	3V	first 2V then 3V	4.5V (B-mode)	HIGH	HIGH	first HIGH then LOW	2V 60km/h	Drive then B-Mode	TRUE	ON to OFF after 20 s	OFF	first FALSE then TRUE	60%	TRUE	LOW	42% (60%*0.7) then 40% when in B-Mode

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.