Model-Based Software Design, A.Y. 2023/24

Laboratory 4 Report

Components of the working group (max 2 people)

- Matteo Gravagnone, s319634
- Danilo Guglielmi, s318083

I/O interfaces

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

Four analog inputs and one digital input have been selected for the input interfaces:

- *BrakePedalPressed:* digital input as it can assume either TRUE (so 5V as electrical tension) or FALSE as Boolean values.

The following inputs are analog because either they assume continuous values or a single value among several ones, like for the states.
At the beginning, a common sequence of blocks is given: the RAW measurement is casted into a double variable to preserve accuracy for the next conversion into an equivalent voltage readout. This value shall then be scaled, taking into account the range of possible values and the reference voltage, to obtain the desired controller inputs.

- *ThrottlePedalPosition*: the input voltage is scaled to a floating point number in the range [0, 1] by applying a gain factor. At the end, we added a cast into single to give it to the controller.
- *ThrottlePedalPositionRedundancy:* same as for *ThrottlePedalPosition*¸ it is a redundant value used in one of the safety mechanisms implemented in Lab3.
- *AutomaticTransmissionSelectorState*: the input voltage is scaled to a floating point number in the range [0, 4] by applying a gain factor. At the end, we added a cast into integer and then one to the enum class to give it to the controller.
- *VehicleSpeed_km_h*: the input voltage is scaled to a floating point number in the range [-60, 240] by applying a gain factor and then subtracting the minimum (negative) value. At the end, we added a cast into single to give it to the controller.

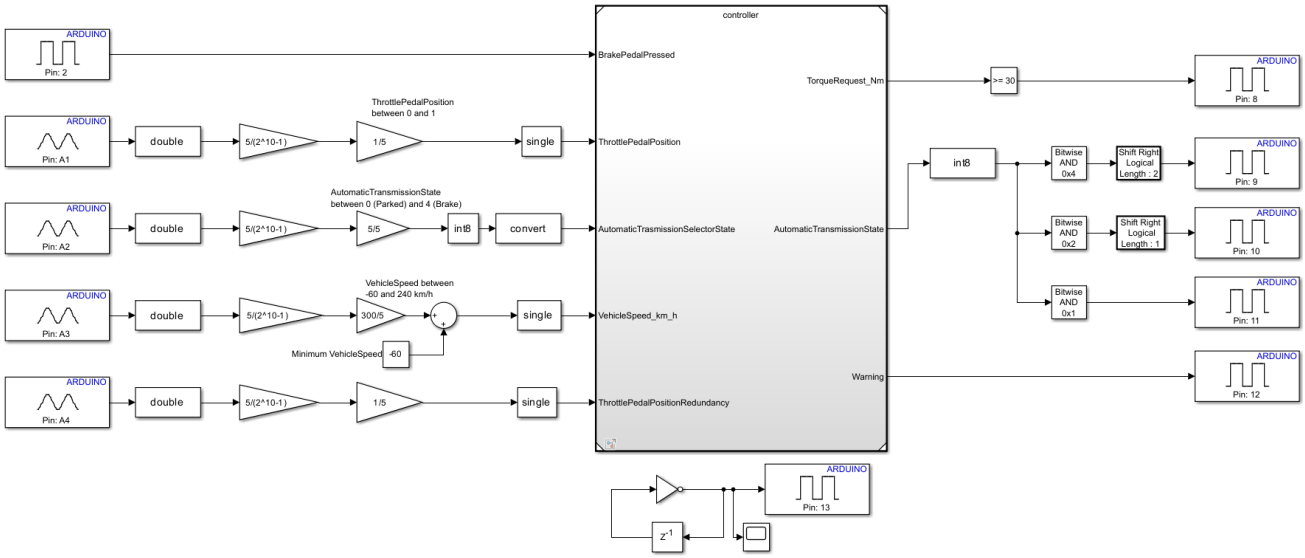
Five digital output pins have been selected for output interfaces:

- *TorqueRequest_Nm:* a threshold value of 30 Nm has been chosen and a 1 is obtained, by means of a comparison block, when the requested torque is greater than the threshold, otherwise 0. Let's note that this is a simplification that does not give as output values of the actual torque, as an high number of pins should be used.
- *AutomaticTransmissionState:* firstly, a cast is needed to convert from enum type to integer (a value from 0 to 4). Then, we implemented a mechanism to obtain the

binary representation of the integer by means of “Bitwise AND” and appropriate “Shift Right Logical” blocks.
The pin 9 represents the Most Significant Bit and the pin 11 the LSB.

- *Warning*: connected to pin 12, it is used to check the value of the warning flags implemented in the safety mechanisms of Lab3. It has high value when at least one condition is met.

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



Name	Unit	Type[3]	Conversion formulas	Min[4]	Max
BrakePedalPressed	Unitless	Digital Input	-	0	1
ThrottlePedalPosition	Unitless	Analog Input	$RAWin * \frac{5}{2^{10}-1} * \frac{1}{5}$	0	1
AutomaticTransmissionSelectorState	Unitless	Analog Input	$RAWin * \frac{5}{2^{10}-1} * \frac{5}{5}$	0	4
VehicleSpeed_km_h	Km/h	Analog Input	$(RAWin * \frac{5}{2^{10}-1} * \frac{300}{5}) - 60$	-60 km/h	240 km/h
ThrottlePedalPositionRedundancy	Unitless	Analog Input	$RAWin * \frac{5}{2^{10}-1} * \frac{1}{5}$	0	1
TorqueRequest_Nm	N*m	Digital Output	-	-80 Nm	80 Nm
AutomaticTransmissionState	Unitless	Digital Output	-	0	4

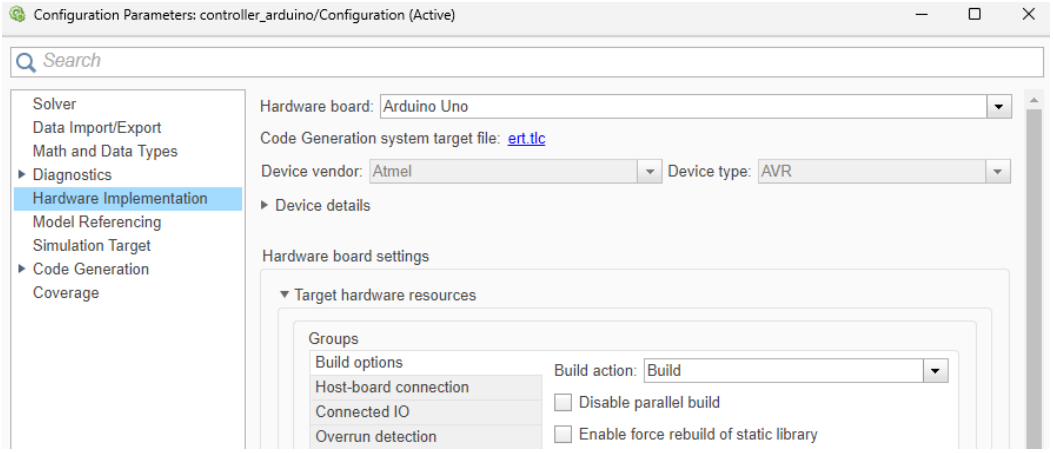
[3] Digital Input (DI), Digital Output (DO), Analog Input (AI).
For AIs, provide the conversion formula from input voltage to the measurement unit data (indicating also how to perform the conversion from the raw reading of the ADC).
[4] The Min/Max values that can be handled due to the conversion formula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

Code generation for Arduino

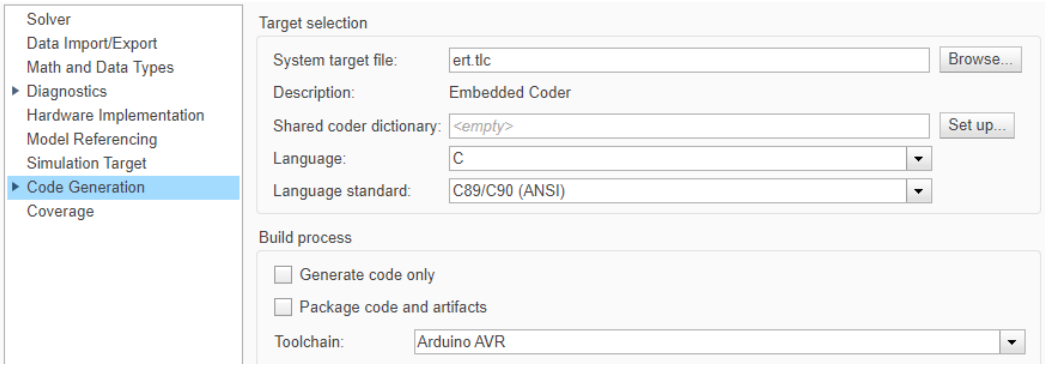
[Please describe the steps to generate the Arduino firmware using the Simulink Support Package for Arduino Hardware]

A prerequisite for this procedure is the proper installation of Simulink Support Package for Arduino Hardware, Embedded Coder, and any necessary dependencies. These shall appear in the Installed section of the Add-On Manager.
Before generating the executable firmware, some *Model Settings* have to be checked for both our "*controller_arduino*" and "*controller*" Simulink models.
In the Solver settings we chose "*Inf*" as Stop time, *Fixed-step* with *discrete solver (no continuous states)* and a proper step size was chosen.

In the "*Hardware Implementation*" section we have to choose as *Hardware Board* "*Arduino Uno*" and, in the Target hardware resources, the *Build* action, under Build options, was chosen as "*Build*" because, in this case, we are not trying to directly load the firmware on a connected board.

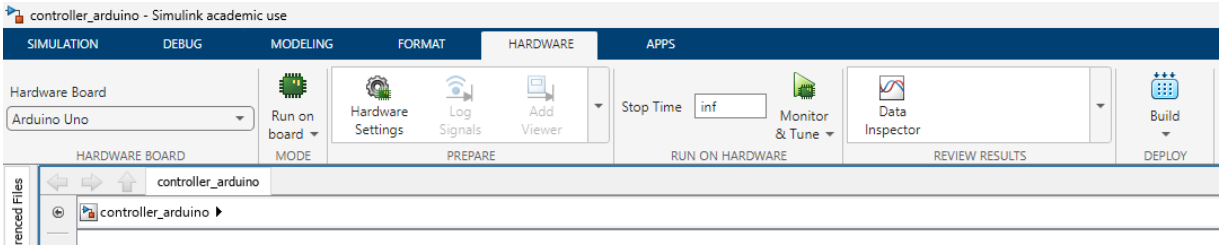


At last, in the Code Generation section, the System target file is "*ert.tlc*" and the description, if the following add-on is correctly installed, should report "*Embedded Coder*".



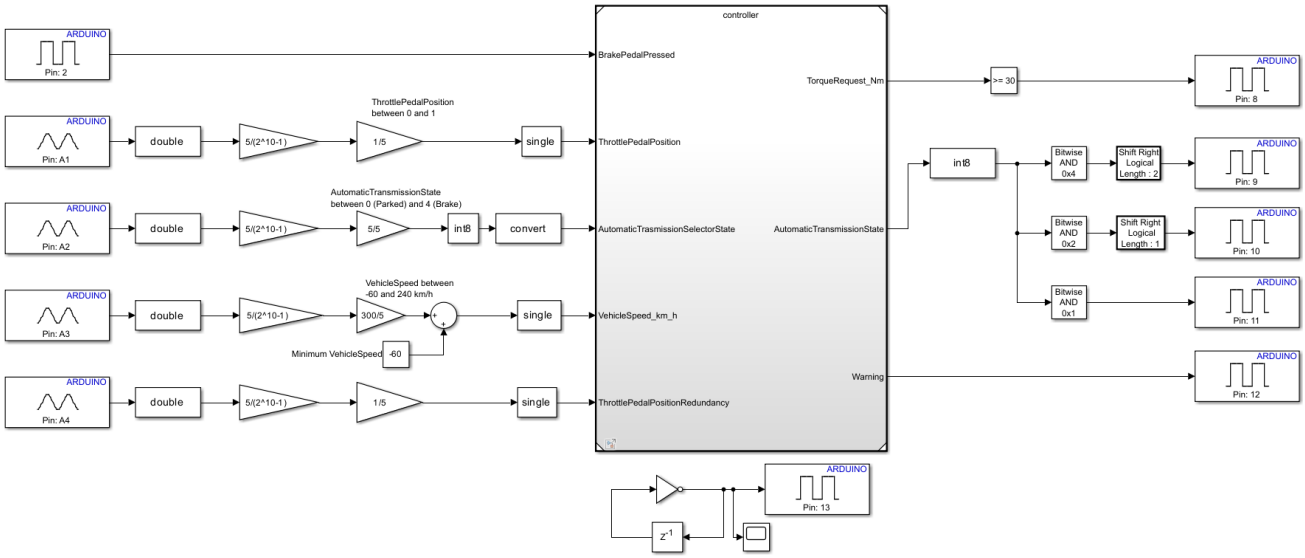
As our controller relies on some definitions given from the *init_fn.m* Matlab script, we need to open and execute this file.

At this point we can actually generate the firmware by going, with the "*controller_arduino*" model opened, to the Hardware section on top and, under Deploy, by clicking on the little arrow, we make sure to select "*Build*" instead of "*Build, Deploy & Start*".



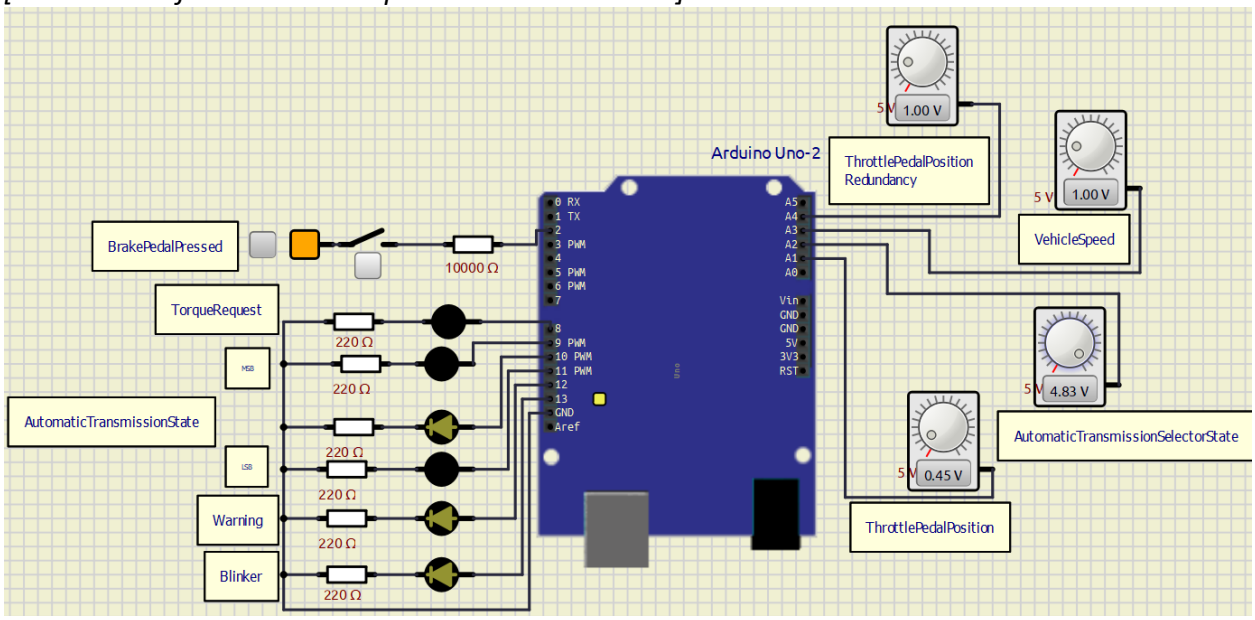
If the steps are correctly executed, a new folder and some controller_arduino files will be in the current folder. As, in this case, we are interested in the "fully executable" approach, we can ignore the new folder and focus on the .hex file, which is our firmware ready to be deployed on simulIde.

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

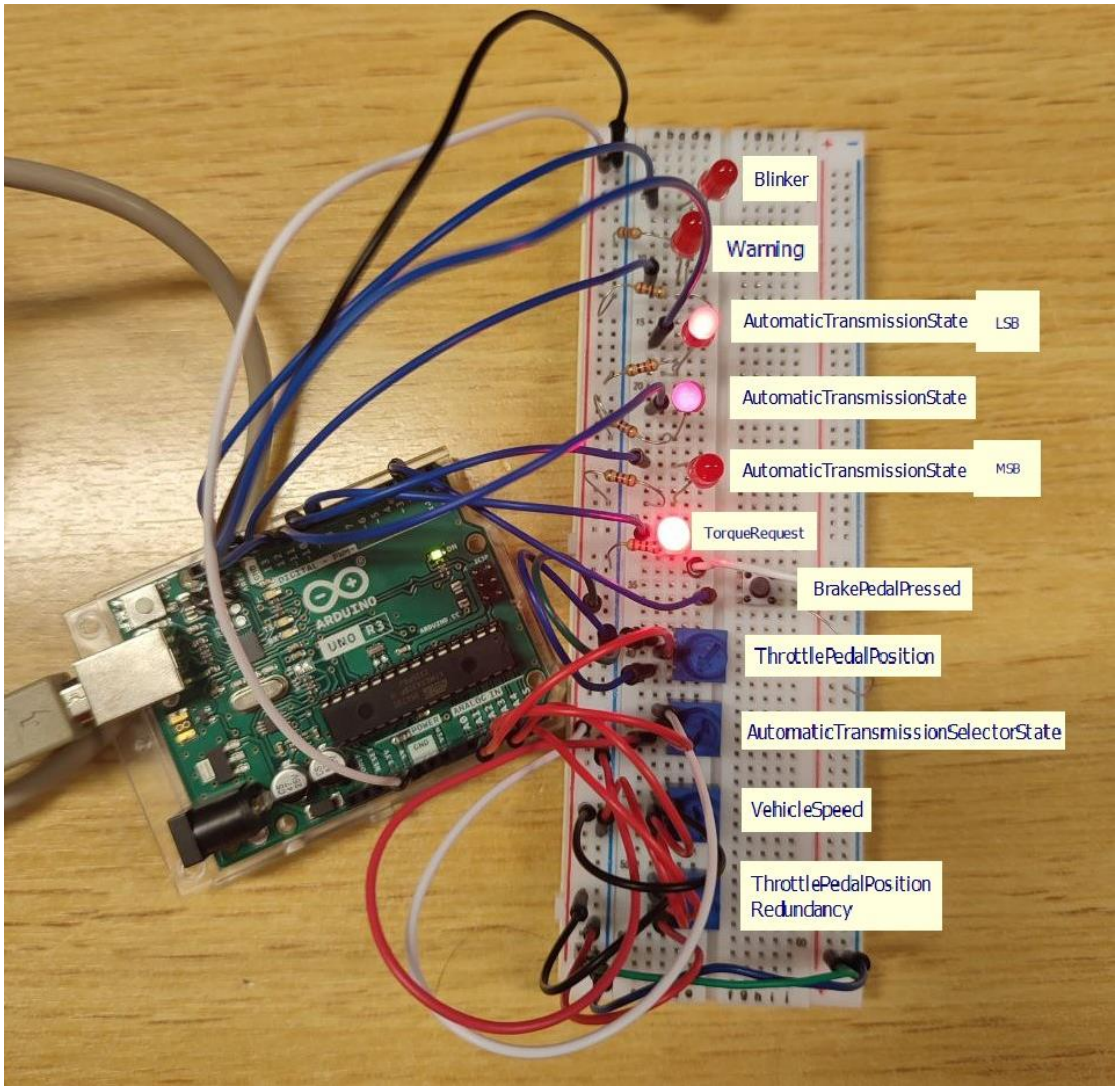


Harness

[screenshot of the harness implemented in SimulIDE]



Harness with Arduino board:

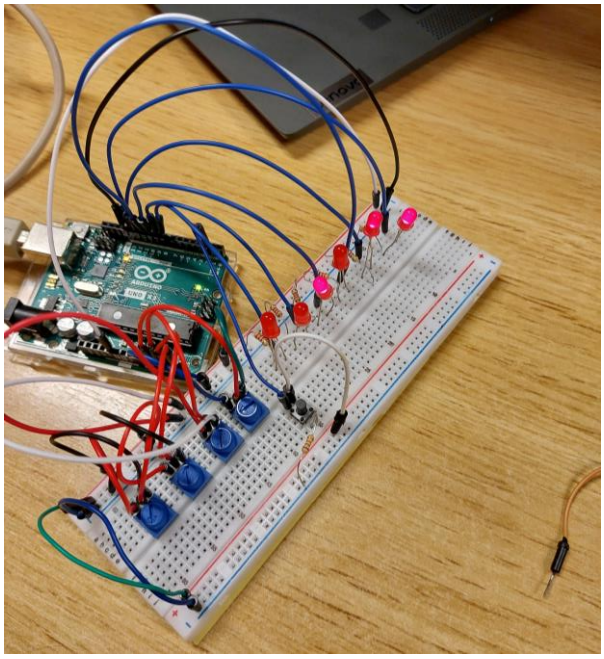


Test stimuli

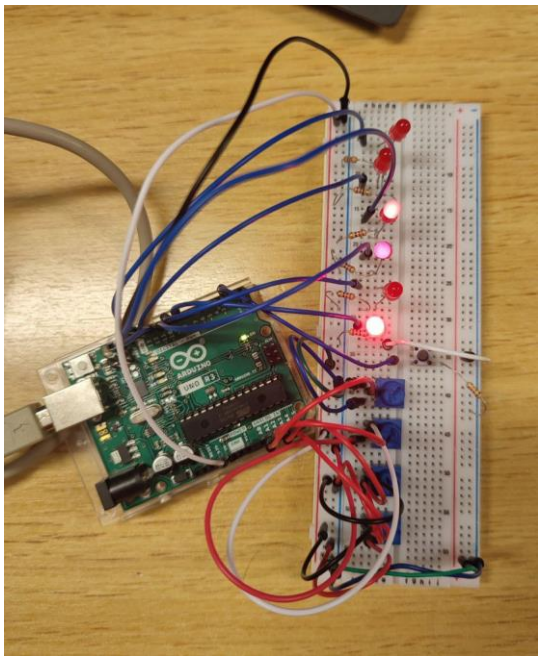
[Provide a brief description of the stimuli set chosen to test the controller functionality] Due to the limitations of the SimulIDE, provide stimuli to check the function of the one pedal controller without the whole physical model

Tests will be reported according to the following scheme:
`(PedalPosition, SelectorState, VehicleSpeed, PedalPositionRedundancy, BrakePedalPressed)->(Warning, TransmissionState[2], TransmissionState[1], TransmissionState[0], TorqueRequest)`

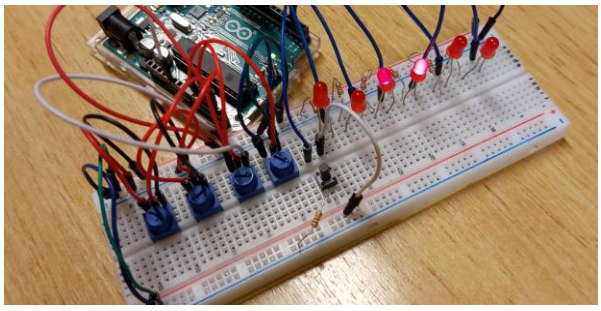
T1: (0.5, Park, 0, 1, false) -> (1,0,1,0,0)
PASSED



T2: (0.8, Drive, 120, 0.8, true) -> (0,1,1,0,1)
PASSED



T3: (0.2, Drive, 30, 0.2, true) -> (0,1,1,0,0)
PASSED



T4: (0.5, Brake, 40, 0.5, true) -> (0,0,0,1,0)
PASSED