

POLITECNICO DI TORINO

DIMEAS

Dipartimento di Ingegneria Meccanica ed Aerospaziale

Thesis submitted for the master's degree in
Mechanical Engineering



AUTOSAR compliant SoftWare Component development to acquire and filter analog signals

Supervisors:

Massimo **SORLI**
Politecnico di Torino

Roberta **RUBIN**
FCA spa Italy

Roberto **SOBRITO**
FCA spa Italy

Angelo **AMENTA**
FCA spa Italy

Fabio **TULLIO**
FCA spa Italy

Candidate:

Gaetano Mirko **SORRENTINO**

*A te, amore mio, che sei la persona che mi ha
sostenuto lungo tutto il cammino e assieme a me
ha patito mille sofferenze.
A te che mi hai saputo far rialzare dopo ogni
caduta e non hai mai chiesto nulla in cambio.*

*Alla mia famiglia per il sostegno e il coraggio di
aver creduto nel mio sogno*

Acknowledgments

I would like to thank my Academic Tutor professor Massimo Sorli for the support and the valuable advices.

I would like to express my greatest gratitude to my Company Tutors for the wonderful experience. I am extremely grateful to have worked with such an efficient and kind team. They were always available for any issues and they helped me in any situations. So, thank you Roberta Rubin for your patience and for all the efforts made to carry out this splendid project. Thank you, Roberto Sobrito and Angelo Amenta, for the funny but above all educational sessions spent teaching me AUTOSAR in every facet. Thank you, Fabio Tullio, for the support.

Table of contents

1. Abstract	6
2. Introduction	6
3. Key selector	8
3.1. Component description	8
3.2. Mechanical model	13
3.2.1. Mechanical model equations	13
3.2.2. Mechanical model in Matlab/Simulink	18
3.2.3. Mechanical model simulation	27
3.3. Equivalent electric circuit	34
3.4. Electrical model in PSpice	37
3.4.1. PSpice	37
3.4.2. PSpice circuit (first simulation)	39
3.4.3. Key selector signal (first simulation)	42
3.4.4. PSpice circuit (second simulation)	43
3.4.5. Key selector signal (second simulation)	46
3.4.6. Montecarlo simulation	47
4. Algorithm description	51
4.1. General algorithm overview	51
4.2. Input signals quantization	51
4.2.1. Key selector status examples	55
4.3. Algorithm structures and variables	58
4.4. Validation time for read values	59
4.5. Algorithm logic	60
5. Algorithm model AUTOSAR compliant	61
5.1. Autosar overview	61
5.2. Benefits and disadvantages using AUTOSAR	64
5.3. SWC development	65
5.4. SWC toolchain	66
5.5. Model description	68
5.6. Threshold runnable	68
5.6.1. Main runnable	70
5.6.2. Model validation	73
6. Simulation	79

7. C-code generation	88
8. Conclusion and future trends	89
9. Acronyms	90
10. Bibliography	90
11. Table of figures	91
12. List of tables.....	94

1. Abstract

The key selector is the part of a car that lets the user to command the engine ignition or the powering of several other components.

The thesis work product is the Matlab/Simulink model of the algorithm that recognizes the mechanical position of a resistive coded key selector based on the corresponding electrical signals. First, the behaviour of the mechanical key rotation is simulated in Simulink. Secondly, the key selector electrical signals are studied and an equivalent electrical circuit in PSpice¹ is realized to obtain the simulated voltage signals acquired by the ECU (Electronic Control Unit). Then, based on the Model Based Design approach and following the principles of the AUTOSAR² architecture, the algorithm model is implemented.

2. Introduction

The technology progress has led to ever greater interactions between different sectors.

Currently it is important to have knowledge both in the mechanical and electrical/electronic fields and it is fundamental to know how to make both worlds interact. Today, many vehicle mechanical subsystems are controlled by one or more ECUs as the result of years of research and studies in the automotive sector.

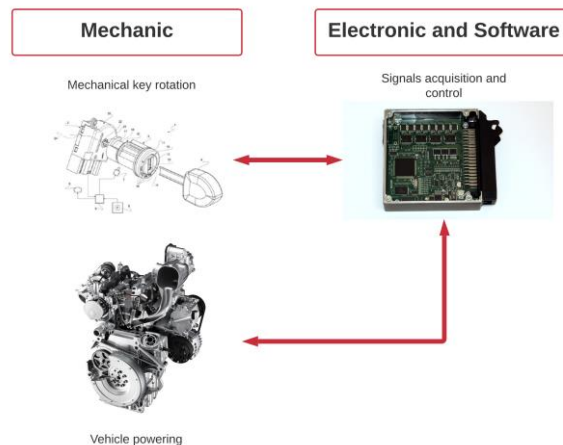


Figure 1: Example of interaction between mechanical and electronic

¹ For further information consult the paragraph *PSpice*

² For further information consult the paragraph *AUTOSAR*

The ECU reads values from several sensors and it computes the data using a software that is stored in the internal memory. It provides instructions to various electrical or mechanical actuators based on processed data. Many ECUs are interconnected and each of them performs the allocated functions. They are placed in different locations and the communication among them relies on suitable communication buses to exchange important information. ECU performances are driven by software algorithms that are developed to match the given requirements.

Software are developed following well-defined architectural structures and standardized guidelines. One of the most common standardized architectures in the automotive field is AUTOSAR.

The focus of the thesis is a key selector that represents the interaction between mechanical and electrical/electronic fields.



Figure 2: Key selector (<https://www.gppaudio.it/ilx-702d-500x-e-ine-w710d-500x/>)

The traditional key selector commands directly the relays that open/close the supply of the circuits that allow the engine on, while the component studied in the thesis is acquired by an ECU that elaborates the electrical signals, commands the actuators and distributes the key selector position information through the communication bus. The analysed key selector is a resistive coded key selector; it means that the mechanical key rotation causes the output signals voltage levels variation. The signals are acquired and elaborated by the software that recognizes:

- the different key mechanical positions;
- some failure conditions.

3. Key selector

The car key selector is a mechatronic subsystem because it is a combination of mechanical, electrical parts and software.

A model of the electrical and mechanical key selector behaviour is developed to produce signals (amplitude and duration) to test the software algorithm. First, a Simulink³ model is developed to simulate the mechanical key rotation in order to obtain the holding time per angle position. Secondly, a PSpice electrical circuit is developed to simulate the voltage value signals.

3.1. Component description

The component is a resistive coded key selector with four mechanical positions: **Ignition lock**, **Accessory**, **Run** and **Start**.

The studied component is composed of a cylinder that is solid to a cam. The cam is coupled to a wheel with an equal number of teeth. The wheel is responsible for the sliding contacts rotation. The rotation angle of the key and of the sliding contacts is the same thanks to the coupling ratio of 1 to 1 and to the efficiency $\eta \approx 1^4$. Every angular position (α) corresponds to an equivalent resistance (R_{eq}) through a transfer function. The key selector structure should be considered as a black box in which the input is the key angular position and the output is the circuit equivalent resistance.

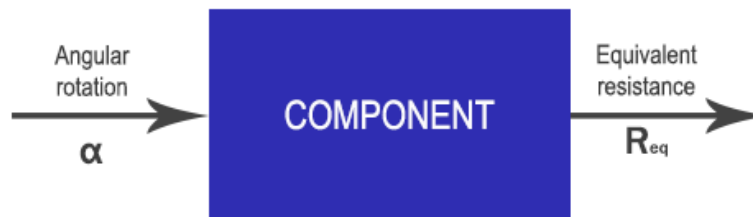


Figure 3: Transfer function

³ For further information consult the paragraph "SWC toolchain"

⁴ For further information consult the paragraph "Mechanical model Equations"

The key selector is hard wired connected to the vehicle battery, that provides the supply to the circuit, and to the ECU that acquires the output signals and that provides the ground connection.

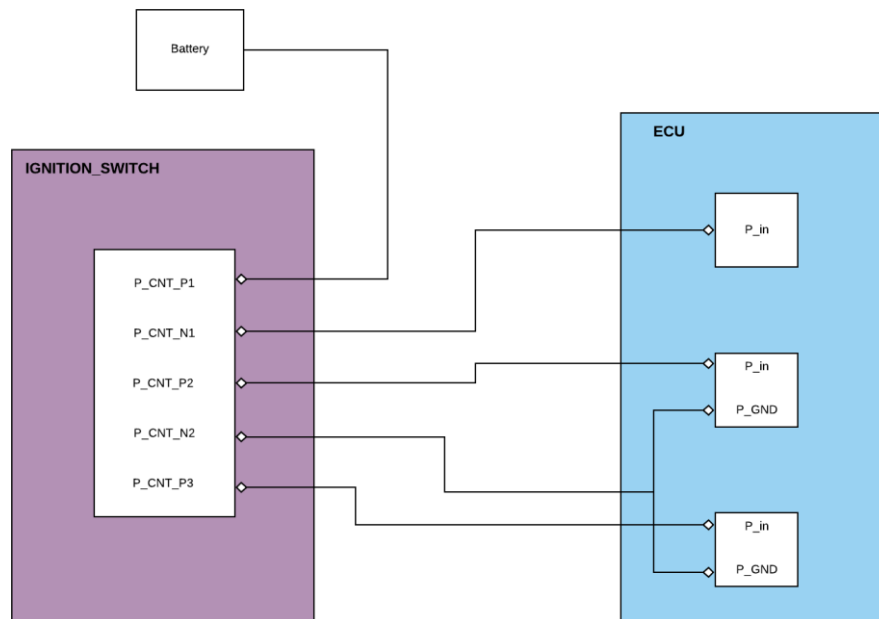


Figure 4: Wire diagram

In Figure 4 it is how the key selector is connected to the ECU. The pin *P_CNT_P1* is connected to the battery voltage through a PTC⁵ resistance. The *P_CNT_N1* is connected to a pull-down resistance connected to ground. The signal is an analog signal that assumes two voltage values.

⁵The PTC resistance that is used in the key selector circuit on pin 1. The PTC is a current-limiting device and it is used as replacements for fuses to protect the circuit. The current that passes through the resistance generates a small amount of heating. If the current is large enough, it generates more heat than the device can lose to its surroundings and this causes its resistance to increase. This self-reinforcing effect limits the current that can pass through. Often the PTC thermistor is used because it is simple, reliable and inexpensive.

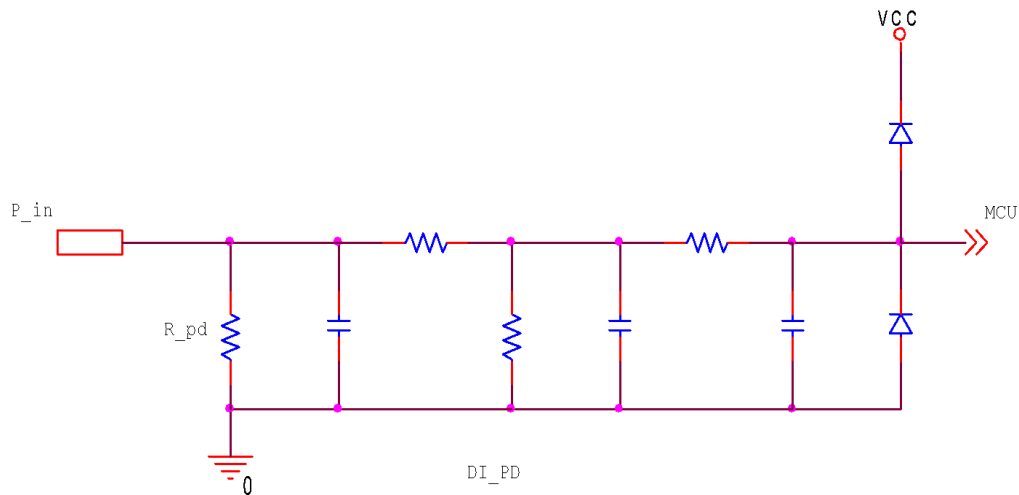


Figure 5: Analog input pull-down ECU interface

The pin P_CNT_P2 and P_CNT_P3 are connected to a pull-up resistance supplied by a voltage regulator (V_a). The P_CNT_N2 is connected to an internal ECU ground to avoid voltage ground shift problems in the acquisition of P_CNT_P2 and P_CNT_P3 . The signals are analog signals that assume different voltage levels.

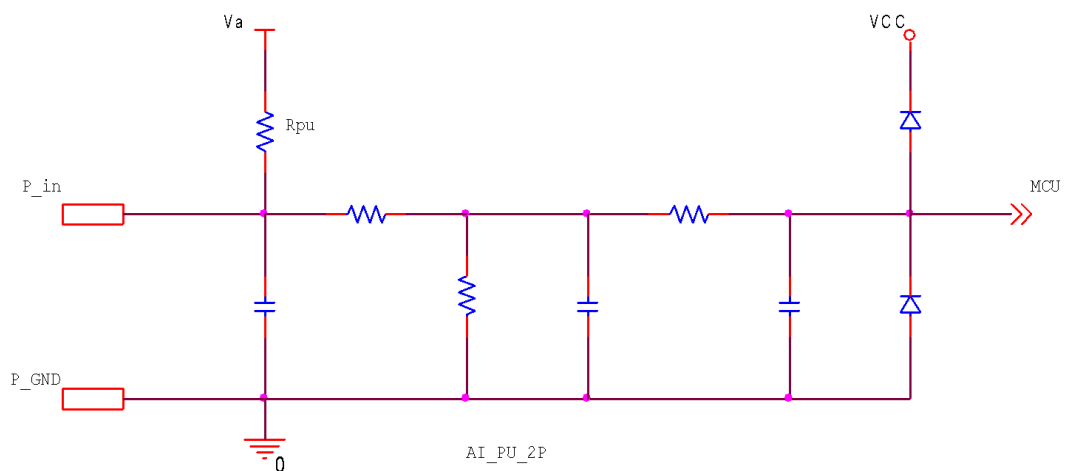


Figure 6: Analog input - Pull-up ECU interface

The **key selector switch** is composed of low intensity sliding contacts with two golden multi-fingers interfaces. The multi-fingers perform a rotation equal to the key angle imposed by the user.

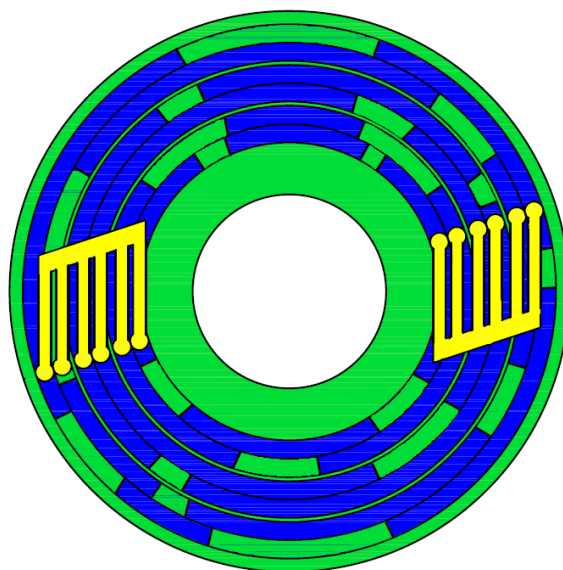
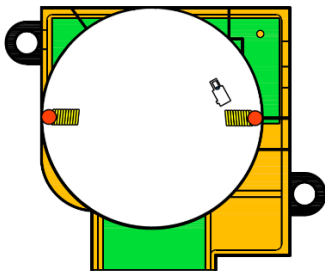


Figure 7: Picture of a key selector for illustration purpose only

The equivalent impedance values per PIN change depending on the angular rotation (clockwise and anticlockwise) of sliding contacts: this causes three different voltage values per key angle sector. The electric or mechanical faults could cause voltage levels different than the nominal condition.

Angle position	Description	User
<p>0°</p>  <p>Figure 8: Angle position 0°</p>	<p>IGNITION LOCK Position (stable)</p> <p>The angle position of 0° imposes the circuit equivalent resistance that determines the voltage values assigned to the Ignition Lock position.</p>	<p>The user rotates the key selector to the ignition lock position to turn off the vehicle.</p>

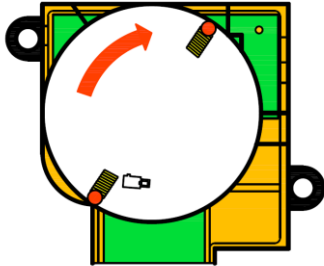
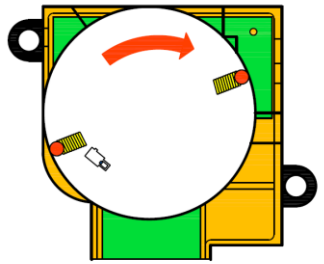
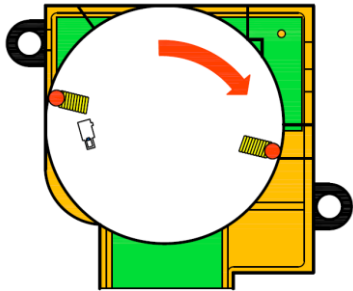
<p style="text-align: center;">70°</p>  <p style="text-align: center;"><i>Figure 9: Angle position 70°</i></p>	<p style="text-align: center;">ACC Position (stable)</p> <p>The angle position of 70° imposes the circuit equivalent resistance that determines the voltage values assigned to the Accessory position.</p>	<p>The user rotates the key selector to the Accessory position to use radio, windshield wipers and other accessories while the engine is off.</p>
<p style="text-align: center;">105°</p>  <p style="text-align: center;"><i>Figure 10: Angle position 105°</i></p>	<p style="text-align: center;">RUN Position (stable)</p> <p>The angle position of 105° imposes the circuit equivalent resistance that determines the voltage values assigned to the Run position.</p>	<p>The use rotates the key to the Run position to operate accessories and use the instrumental panel. This position is also used by the qualified operators to look at the car and run diagnostics.</p>
<p style="text-align: center;">140°</p>  <p style="text-align: center;"><i>Figure 11: Angle position 140°</i></p>	<p style="text-align: center;">START Position (temporary)</p> <p>The angle position of 140° imposes the circuit equivalent resistance that determines the voltage values assigned to the Start position.</p>	<p>The user rotates the key selector to the Start position keeping the foot on the brake pedal (on the clutch if the car is with a manual transmission) to start the engine. When the engine starts and the user releases the key, the key selector switch automatically returns to the RUN position.</p>

Table 1: Angle description and user functionalities

The major part of the key positions is stable except the *START* position; an elastic spring works in order to return the key from *START* to *RUN*.

3.2. Mechanical model

A mechanical model is usually defined in order to analyse the mechanical behaviour of systems.

The key selector angular position is obtained starting from the torque applied by the user on the key. The user couple applied in the simulation is derived from mechanical studies done by the component supplier.

If the angular position is known, the holding time of every key position is calculable from the angular speed.

3.2.1. Mechanical model equations

There are two parts of the key selector implied in the angular rotation that are responsible for the multi fingers angular position (α): the cylinder and the rotational switch.

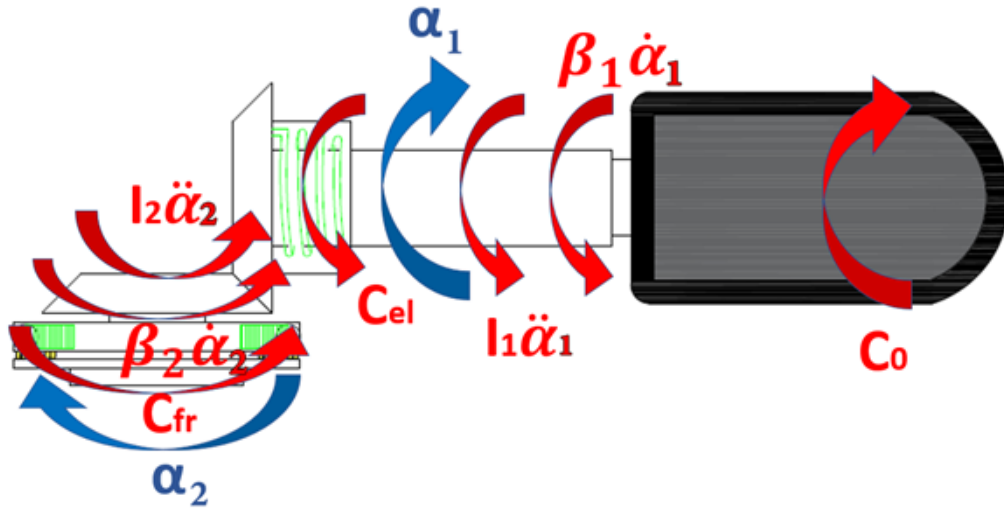


Figure 12: Torques applied on key selector

C_0 is the user torque applied on the key that causes the rotation of the coded cylinder.

$I_1 \ddot{\alpha}_1$ and $I_2 \ddot{\alpha}_2$ are the moment of inertia for the cylinder and for the body with multi fingers.

$\beta_1 \dot{\alpha}_1$ and $\beta_2 \dot{\alpha}_2$ are the damping torques.

C_{el} is the elastic torque due to an angular spring that is active from RUN to START without a preload.

C_{fr} is the friction torque due to the contact between the steel ball and the cover (material: MINLON 11C 140 BK086). The balls are used to maintain the angle

when there isn't an applied torque and to give to the user the feeling of the different key positions.

The gear wheels coupling is always characterized by a transmission ratio and its efficiency.

$$\tau = \frac{\dot{\alpha}_2}{\dot{\alpha}_1} = \frac{\alpha_2}{\alpha_1} \Rightarrow \dot{\alpha}_2 = \tau \dot{\alpha}_1; \alpha_2 = \tau \alpha_1 \quad (1)$$

$$\eta = \frac{P_{out}}{P_{in}} = \frac{C_{tr2} \dot{\alpha}_2}{C_{tr1} \dot{\alpha}_1} \Rightarrow C_{tr1} = C_{tr2} \cdot \frac{\tau}{\eta} \quad (2)$$

1 is used to refer to the cylinder and 2 is used to refer to the switch.

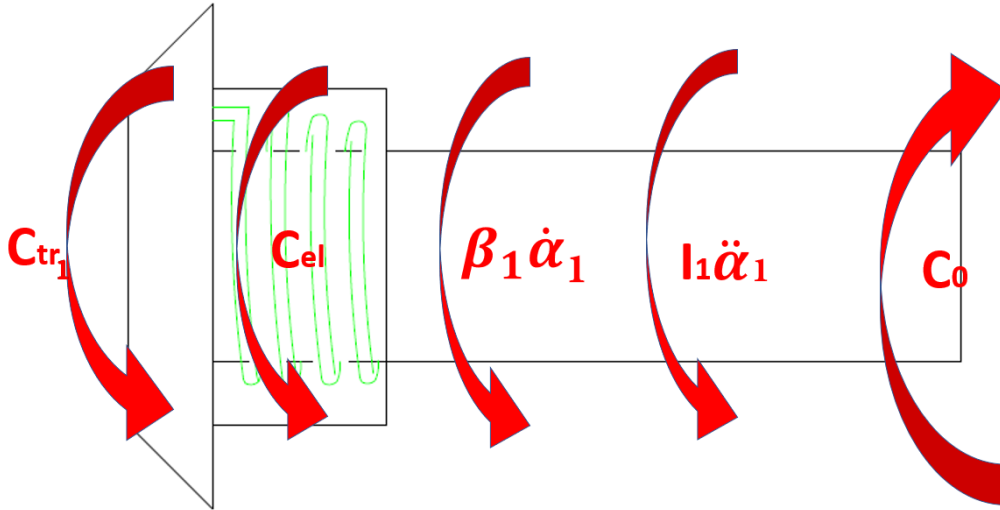


Figure 13: Free body diagram of the cylinder

The rotation equilibrium in the body diagram of the cylinder is:

$$C_0 - C_{tr1} = C_{el} + I_1 \ddot{\alpha}_1 + \beta_1 \dot{\alpha}_1 \quad (3)$$

where C_{tr} is the transmission torque and the elastic torque is:

$$C_{el} = k_\theta \cdot (\alpha_1 - \alpha_0) \quad (4)$$

with $\alpha_0 = 1.8326 \text{ rad}$ (105°) that corresponds to the RUN position.

The key selector is an under-damped system and it means that the damping factor is $0 < \zeta < 1$. A damping factor $\zeta = 0.2$ is considered in this case. The damping coefficient for the cylinder is calculated as

$$\beta_1 = \zeta \cdot 2\sqrt{I_1 \cdot k_\theta} \quad (5)$$

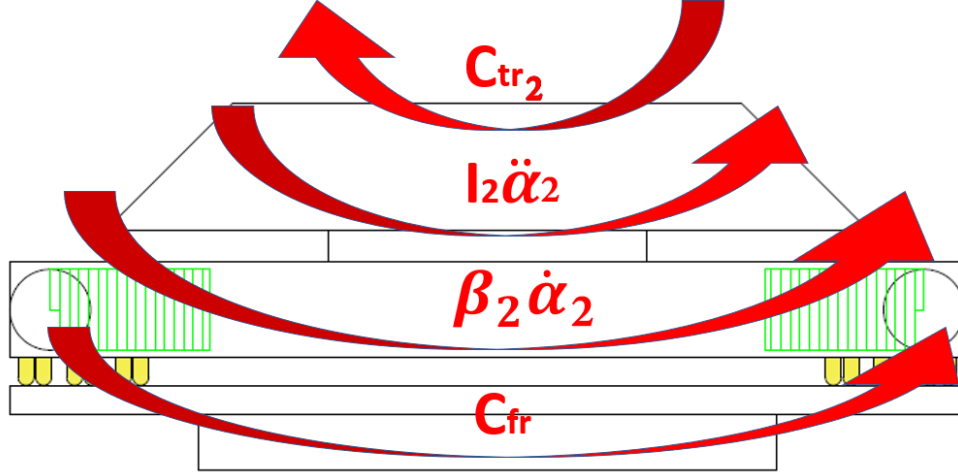


Figure 14: Free body diagram of the switch

The equilibrium in the body diagram of the rotational is:

$$C_{tr_2} = I_2 \ddot{\alpha}_2 + \beta_2 \dot{\alpha}_2 + C_{fr} \quad (6)$$

The damping effect for the switch is negligible and so

$$\beta_2 \dot{\alpha}_2 \approx 0 \quad (7)$$

To obtain the friction torque is necessary to consider the spring effect on the steel balls.

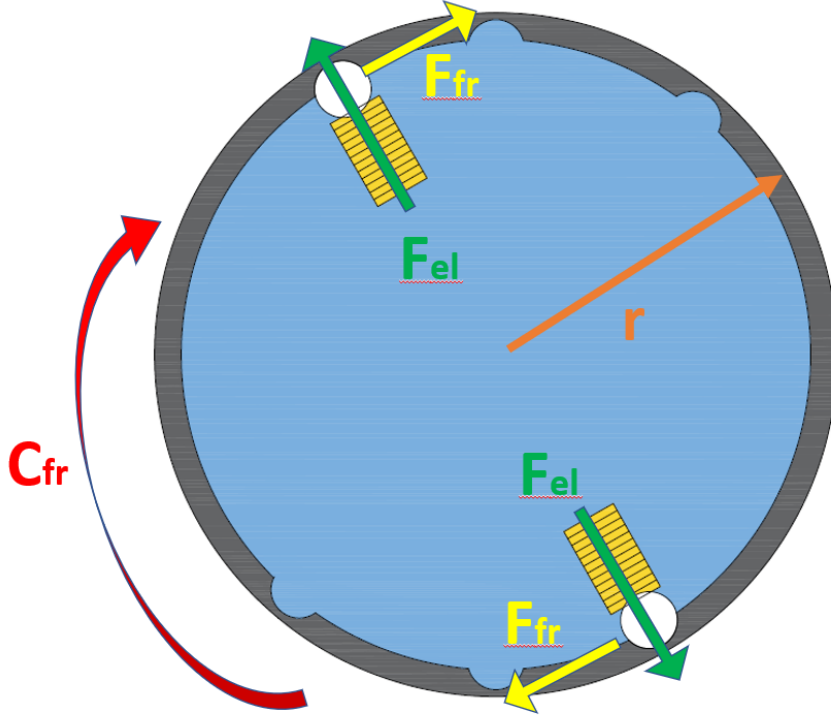


Figure 15: Torques applied on rotational

The springs apply elastic forces that cause friction.

$$F_{el} = k \cdot (l - l_0) + F_{el_0} \quad (8)$$

where l_0 is the initial length of the spring and F_{el_0} is the preload.

$$F_{fr} = \mu \cdot F_{el} \quad (9)$$

where μ is the friction coefficient and its value is 0.38 for the contact between steel and a polyamide.

The friction couple is:

$$C_{fr} = 2 \cdot F_{fr} \cdot r + C_{fr_{add}} \quad (10)$$

where $C_{fr_{add}}$ is the additional friction torque generated by the position markers.

Considering the equation (3) and (6)

$$C_{tr_2} = I_2 \ddot{\alpha}_1 \tau + \beta_2 \dot{\alpha}_1 \tau + C_{fr} \quad (11)$$

Inserting in the equation (1) and (2) is obtained

$$C_{tr_1} = \frac{I_2 \tau^2}{\eta} \ddot{\alpha}_1 + \frac{\beta_2 \tau^2}{\eta} \dot{\alpha}_1 + C_{fr} \cdot \frac{\tau}{\eta} \quad (12)$$

and so

$$C_0 - C_{el} - I_1 \ddot{\alpha}_1 - \beta_1 \dot{\alpha}_1 - \frac{I_2}{\eta} \cdot \tau^2 \ddot{\alpha}_1 - \frac{\beta_2}{\eta} \cdot \tau^2 \dot{\alpha}_1 - C_{fr} \cdot \frac{\tau}{\eta} = 0$$

$$\Rightarrow \left(I_1 + \frac{I_2 \tau^2}{\eta} \right) \ddot{\alpha}_1 + \left(\beta_1 + \frac{\beta_2 \tau^2}{\eta} \right) \dot{\alpha}_1 + k_\theta \alpha = C_0 + k_\theta \alpha_0 - C_{fr} \cdot \frac{\tau}{\eta} \quad (13)$$

Considering $I_{eq} = I_1 + \frac{I_2 \tau^2}{\eta}$ and $\beta_{eq} = \beta_1 + \frac{\beta_2 \tau^2}{\eta}$ it is obtained:

$$I_{eq} \ddot{\alpha}_1 + \beta_{eq} \dot{\alpha}_1 + k_\theta \alpha = C_0 + k_\theta \alpha_0 - C_{fr} \cdot \frac{\tau}{\eta} \quad (14)$$

Both the bodies are considered approximately as a full cylinder in order to calculate the inertia:

$$I_1 = \frac{1}{2} \cdot M_1 \cdot R_1^2 \text{ for the cylinder}$$

$$I_2 = \frac{1}{2} \cdot M_2 \cdot R_2^2 \text{ for the rotational}$$

M is the mass and R is the body ray.

The final equation considered for the acceleration calculus is:

$$\ddot{\alpha}_1 = \frac{\left(C_0 - C_{fr} \cdot \frac{\tau}{\eta} \right) - \beta_{eq} \cdot \dot{\alpha}_1 - k_\theta \cdot (\alpha_1 - \alpha_0)}{I_{eq}} \quad (15)$$

The Figure 16 is the high-level view of the key selector mechanical model to obtain the angular position and the angular speed vs time with the equations of the previous paragraph.

The *user torque* is applied only clockwise. It is imported with a signal builder.

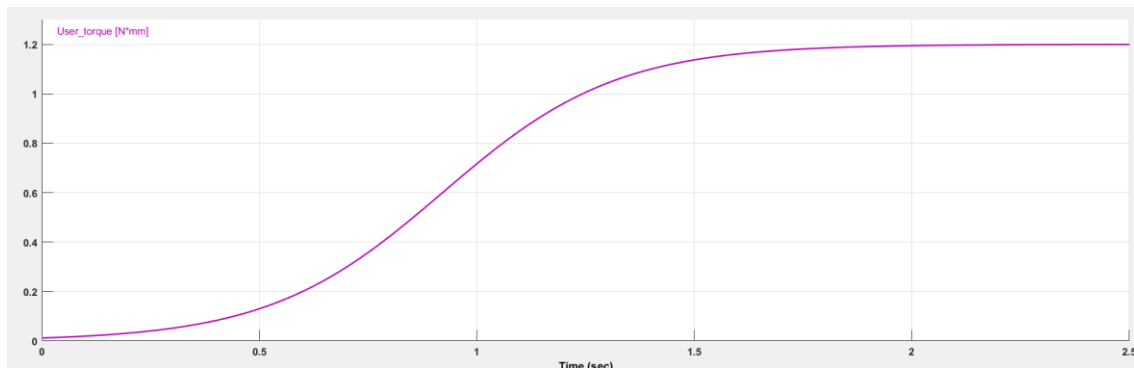


Figure 18: User torque clockwise

The testing user torque is a sort of ramp with a smooth at the start and the end. This torque is used only to test the key selector behaviour.

For this test the user torque is applied for 2.5 s and then the key is released. After releasing the rotational spring turns the angular position from Start to Run.

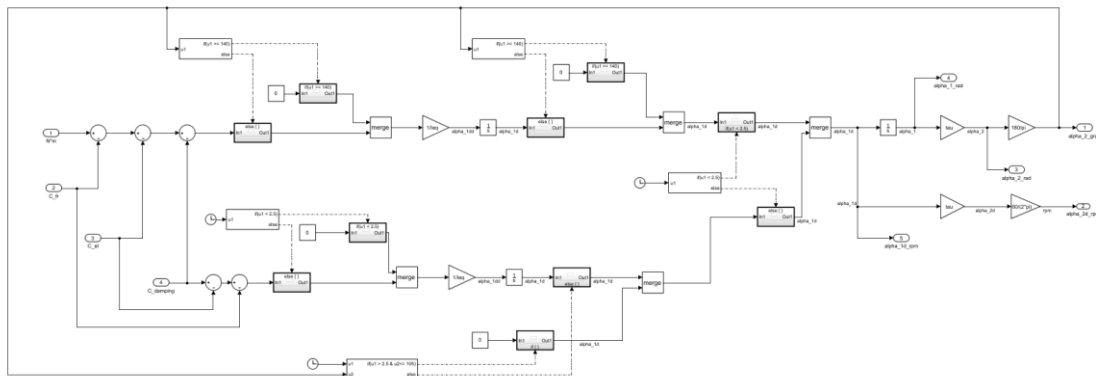


Figure 19: Structure of the main logic

The Figure 19 is the main part of the mechanical model.

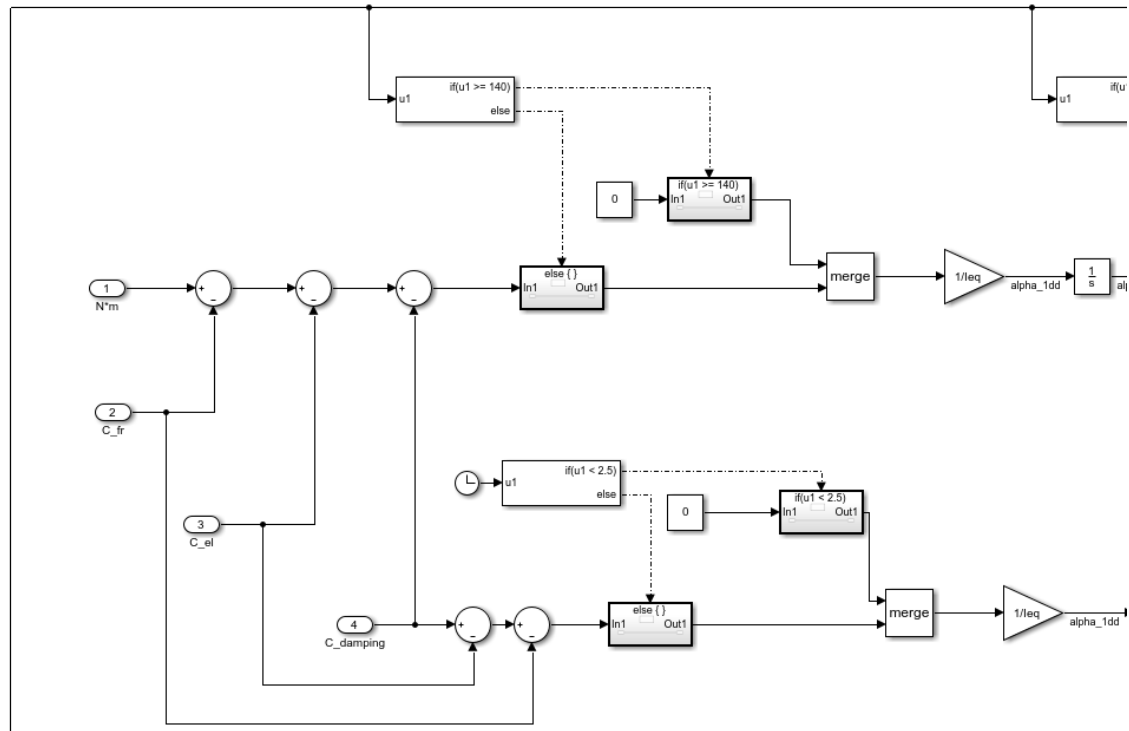


Figure 20: Torque implementation (left part)

Torques are implemented in order to obtain the acceleration: this result is calculated dividing the torques by the system inertia with the equations explained in the previous paragraph. The upper part is dedicated to the key rotation clockwise while the lower part is dedicated to the return from Start to Run.

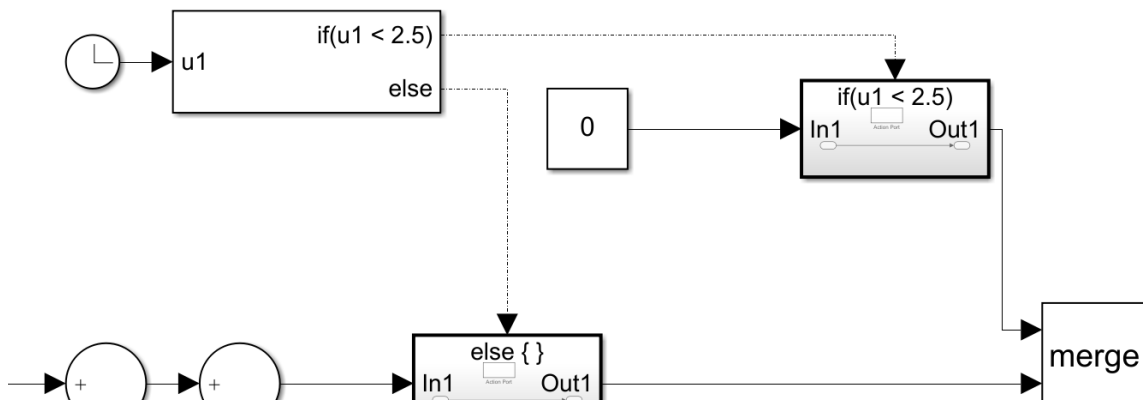


Figure 21: if block return activation (lower part)

The *if block* in the Figure 21 is used to activate the counterclockwise part of the logic after 2.5 s that corresponds to the moment in which the user leaves the key.

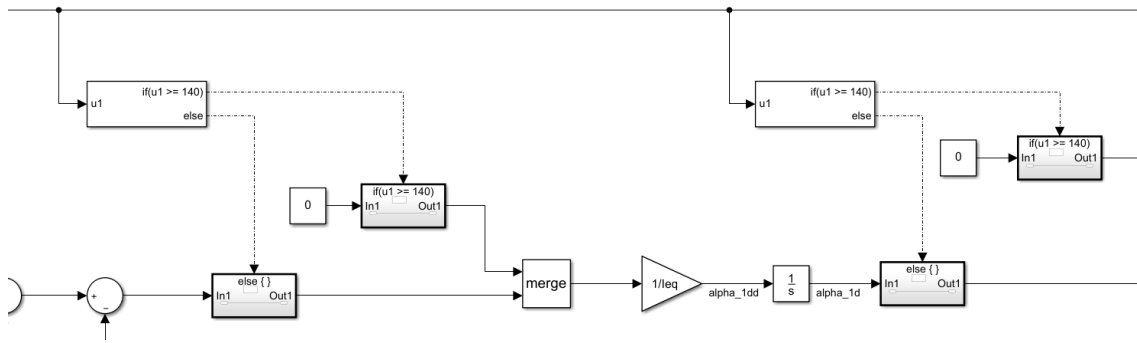


Figure 22: if block for the physical block for the clockwise part (upper part)

The *if blocks* in the Figure 22 are used to simulate the physical block at the end of the key rotation. The resulting torque and the angular speed are equal to 0 when the angle is 140° .

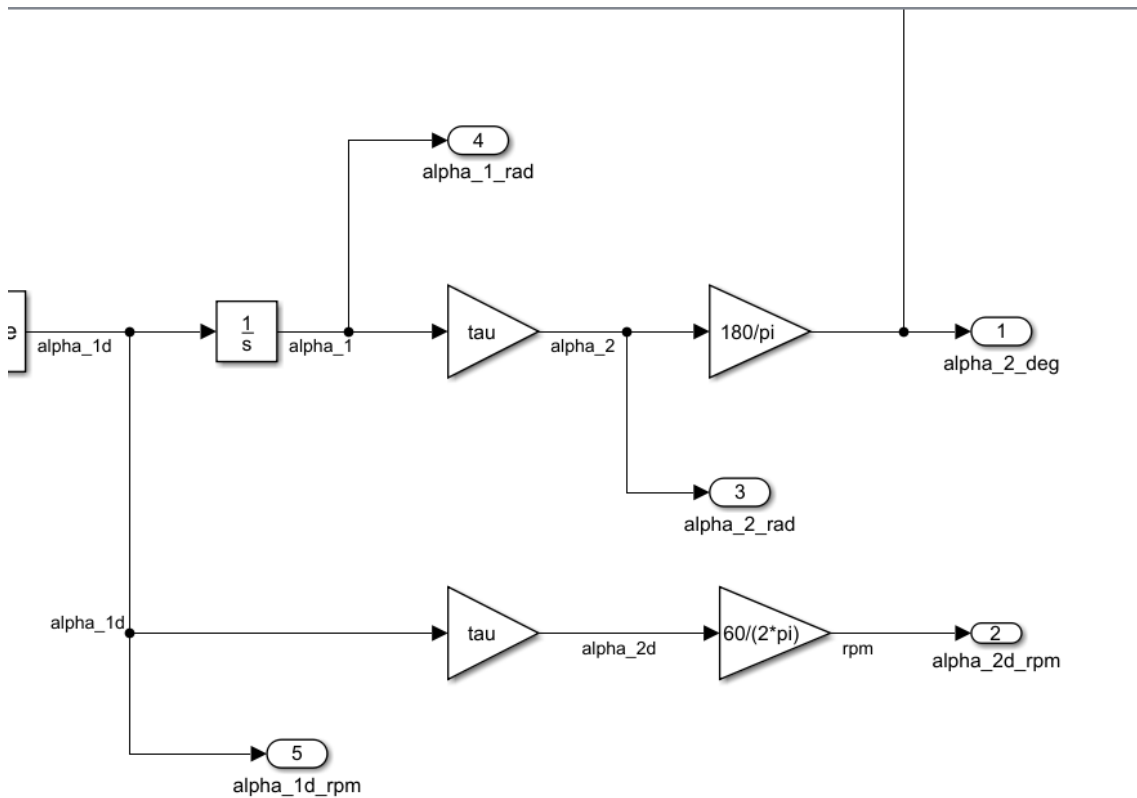


Figure 23: Angle conversion (right part)

The Figure 23 is the part of the main logic dedicated to the angle conversion: from angular position of the cylinder (α_1) to the multi fingers angular position in degrees (α_2_{deg}).

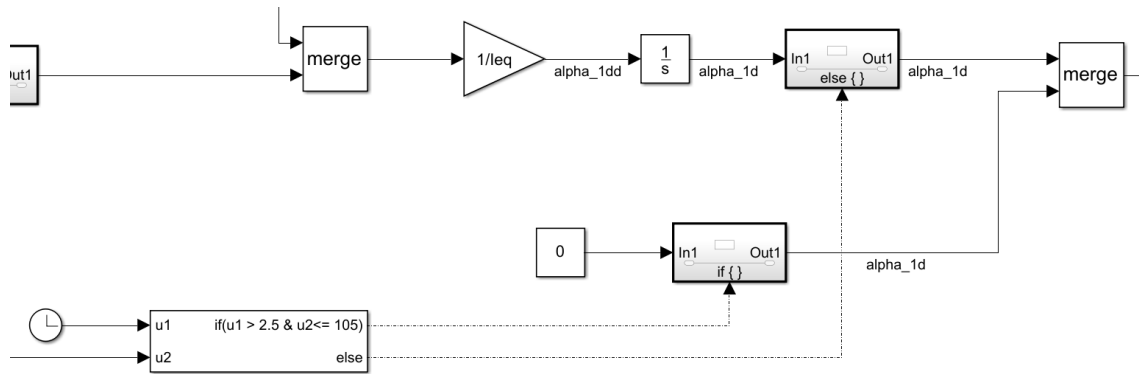


Figure 24: *if* block for the physical block for the counterclockwise part (lower part)

The *if* block in the figure ?? is used to simulate the physical block for the counterclockwise key rotation. The angular speed is 0 rad/s until $time > 2.5\text{ s} \ \& \ \alpha_{2} < 105^{\circ}$.

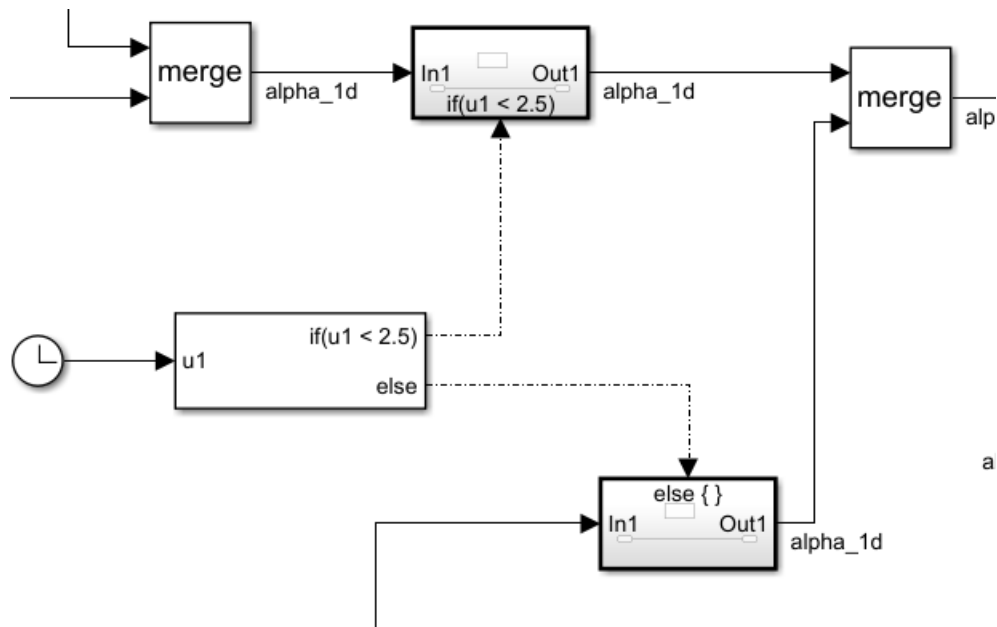


Figure 25: *if* block for the rotation direction

The Figure 25 is the *if* block used to select the clockwise rotation part of the model or the counterclockwise one. It is based on the key releasing time.

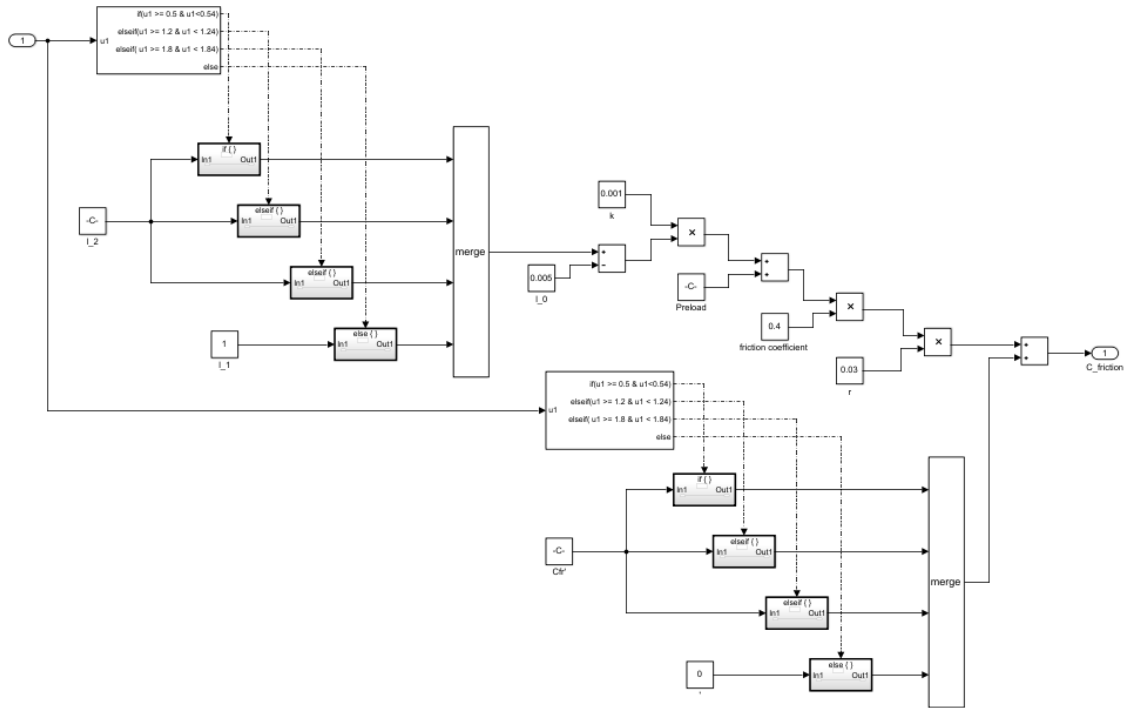


Figure 26: Simulink model for the friction torque

The Figure 26 is the friction torque block.

The length of the spring in the Figure 15 is almost the same except when the angular steel balls pass through the position markers.

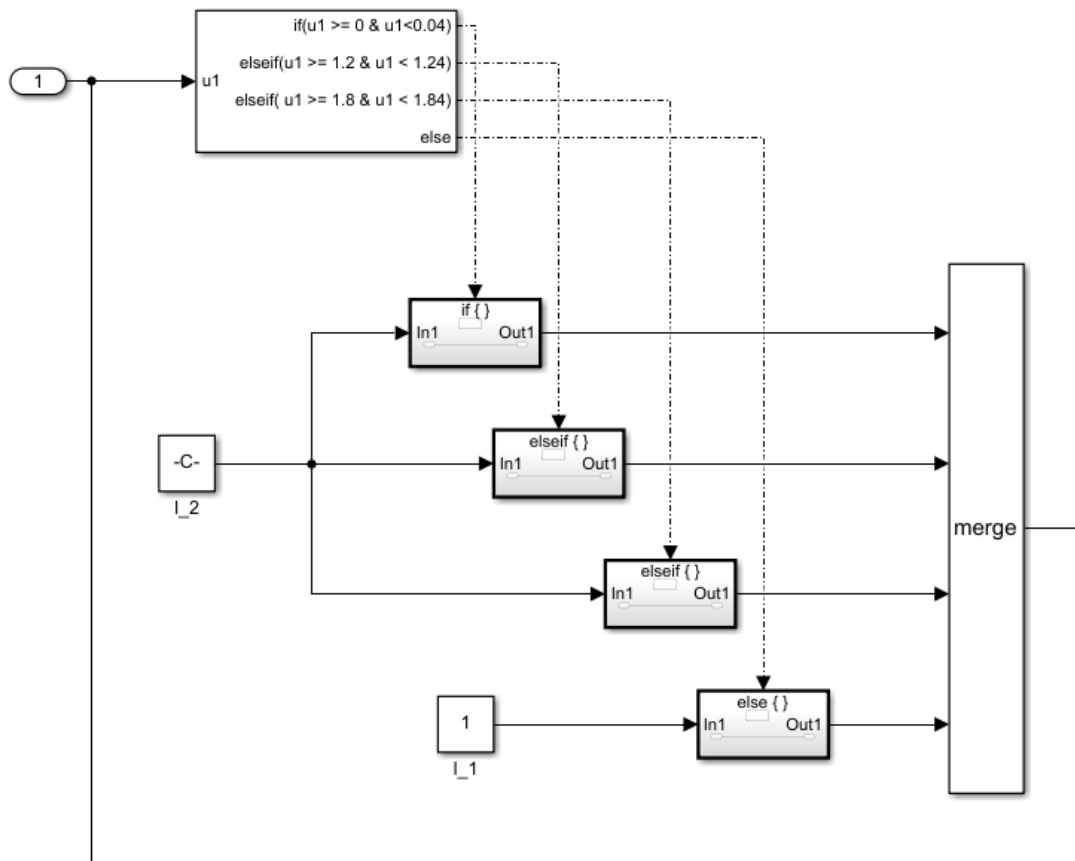


Figure 27: If block for the spring length variation

The *if block* in the Figure 27 is used for the spring length variation during the rotation.

$$\begin{aligned} \text{if } 0 < \text{angle} < 0.04 \text{ rad} \mid 1.2 < \text{angle} < 1.24 \mid 1.8 < \text{angle} < 1.84 \Rightarrow l = l_2 \\ \text{else} \Rightarrow l = l_1 \end{aligned}$$

The radiant values correspond to the position Ignition lock, Accessory and Run. There isn't the position marker in the Start position. The *merge block* sums its inputs.

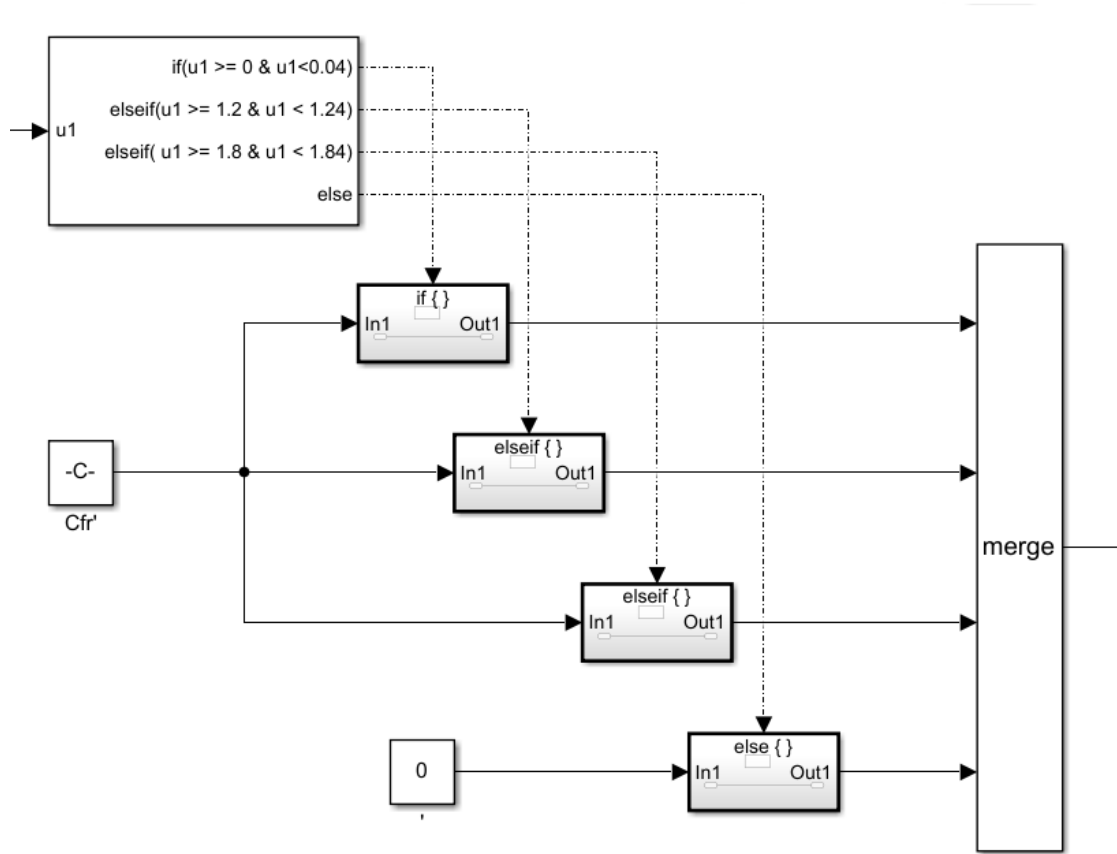


Figure 28: If block for the spring length variation

The *if block* in the Figure 28 is used to activate the additional friction torque only when the key angular position is Ignition lock, Accessory and Run. The *merge block* sums its inputs.

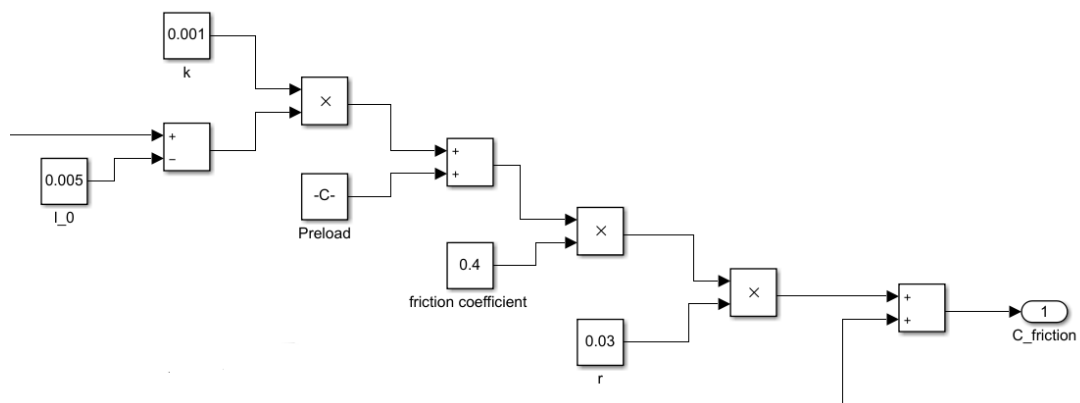


Figure 29: Main part of the friction torque model

The model in the Figure 29 is the transposition of the equations (8),(9) and (10).

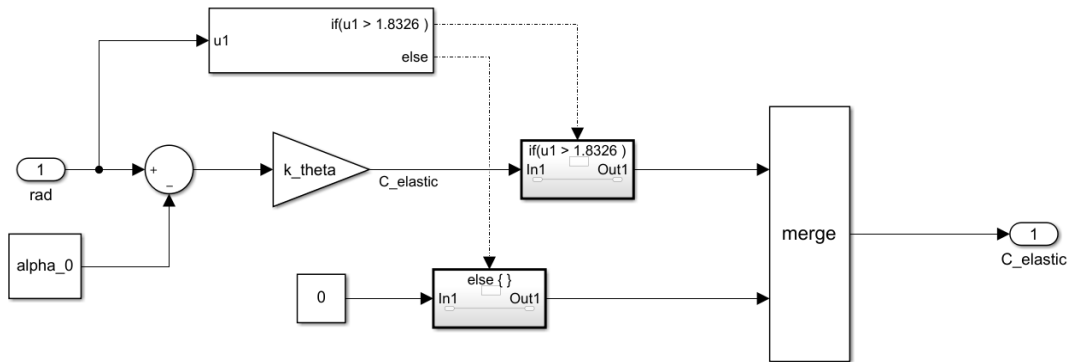


Figure 30: Simulink model for the elastic torque

The Figure 30 is the elastic torque block.

An *if block* is used to activate the elastic torque only from Run to Start (expressed in radians):

if angle > 1.8326° ⇒ the elastic torque is active

if angle < 1.8326° ⇒ the elastic torque is 0

The *merge block* sums its inputs. The rest of the structure is the transposition of the equation (4).

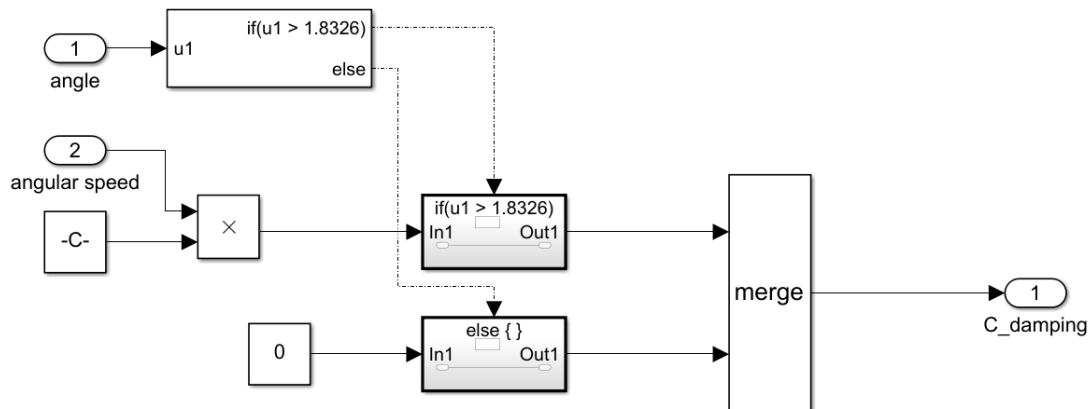


Figure 31: Damping torque model

The Figure 31 is the inside of the damping torque block. The damping torque is activated only from Run to Start in consequence of the elastic torque.

3.2.3. Mechanical model simulation

```
MechanicalModel2.m  Input.m  +
1 - tau = 1;
2 - eta = 0.98;
3 - m1 = 0.25; % [m]
4 - m2 = 0.1; % [m]
5 - r1 = 0.03; % [m]
6 - r2 = 0.03; % [m]
7 - k_theta = 0.0005; % [N*m/rad]
8 - zita = 0.2;
9 - alpha_0 = 1.8326; % [rad]
10 - I1 = (m1*(r1)^2)/2; % [kg*m^2]
11 - I2 = (m2*(r2)^2)/2; % [kg*m^2]
12 - beta1 = 2*zita*(I1*k_theta)^(1/2); % [(kg*m^2)/(rad*s)]
13 - beta2 = 0;
14 - Ieq = (I1+(I2*(tau)^2)/eta); % [kg*m^2]
15 - betaeq = (beta1+(beta2*(tau)^2)/eta); % [(kg*m^2)/(rad*s)]
16
```

Figure 32: Damping torque model

The Figure 32 is the program used to initialise the mechanical model. There are parameters such as the masses, geometrical dimensions, coefficients and inertia.

The wheel's teeth number is the same and so the transmission ratio is 1. The efficiency of the system is approximately 0.98.

The friction torque additional contribution at 70° isn't considered for the simulation of the configuration without Accessory.

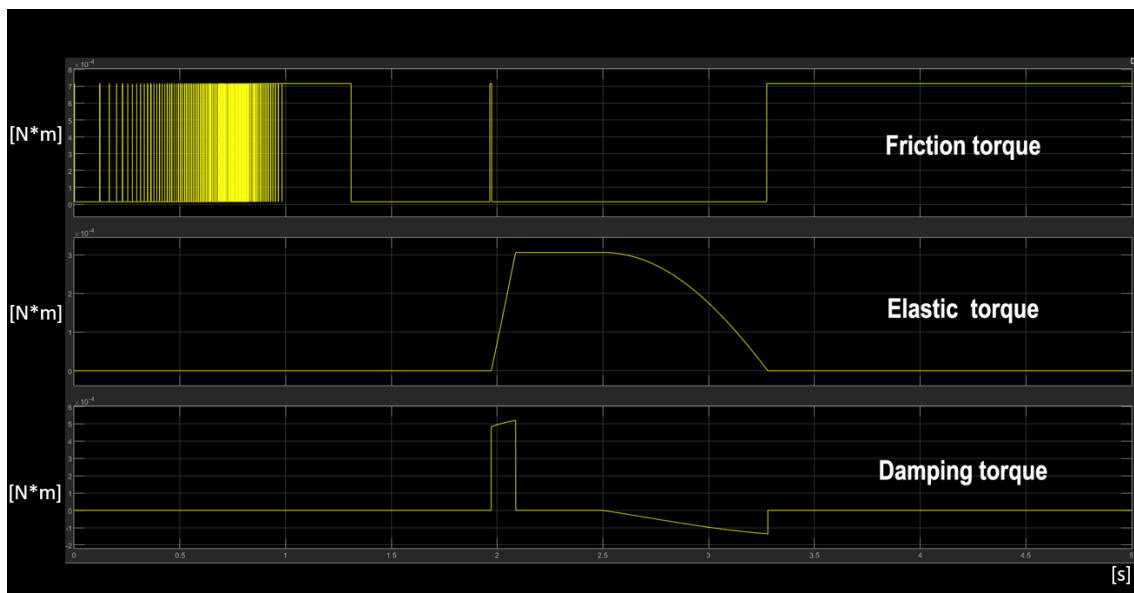


Figure 33: From the top to the bottom: friction torque, elastic torque and damping torque (configuration without Accessory)

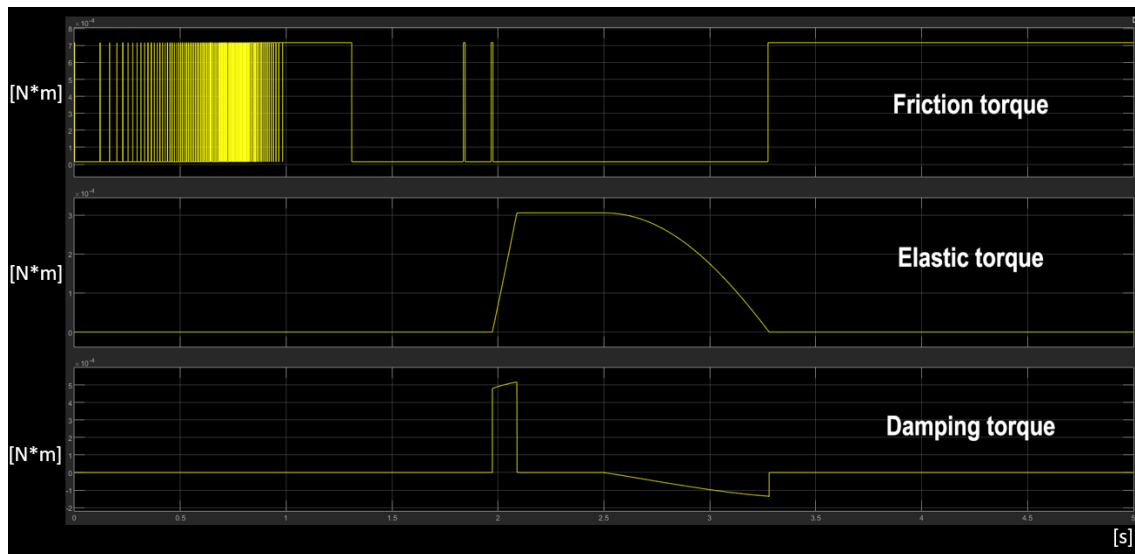


Figure 34: From the top to the bottom: friction torque, elastic torque and damping torque (configuration with Accessory)

The friction torque is the result of two components well-described in the previous paragraph. The spikes corresponding to the additional torque due to the position marking.

The elastic torque is activated starting from the Run position to the Start position. Its value decreases as it gets closer to the Run position.

The damping torque acts only when the elastic torque is active and the angular speed has a positive value.

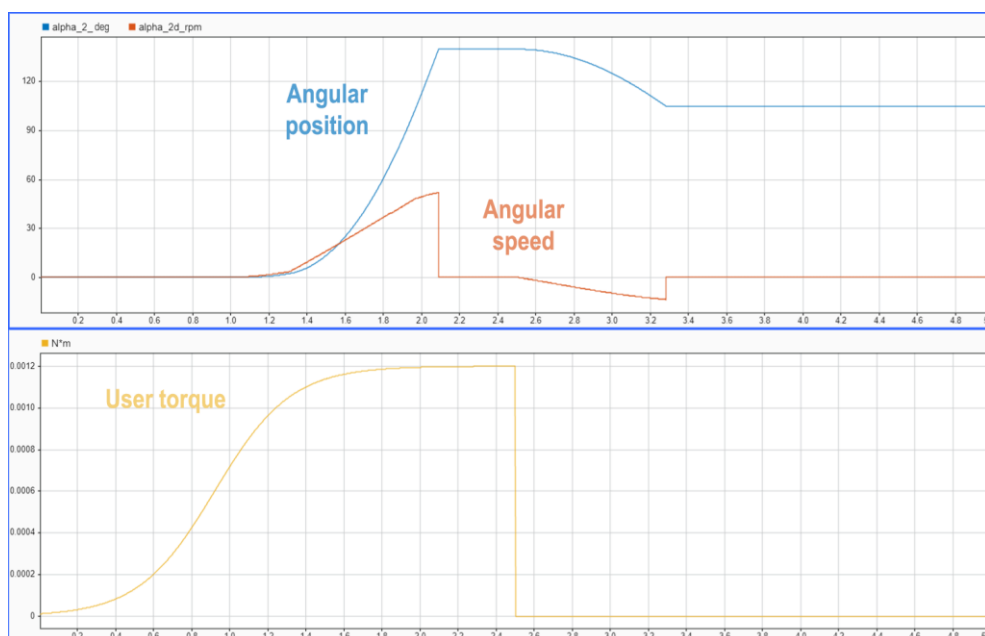


Figure 35: The angle position (degree vs time), the angular speed (rpm vs time) and the user torque (N*m) in the configuration without Accessory

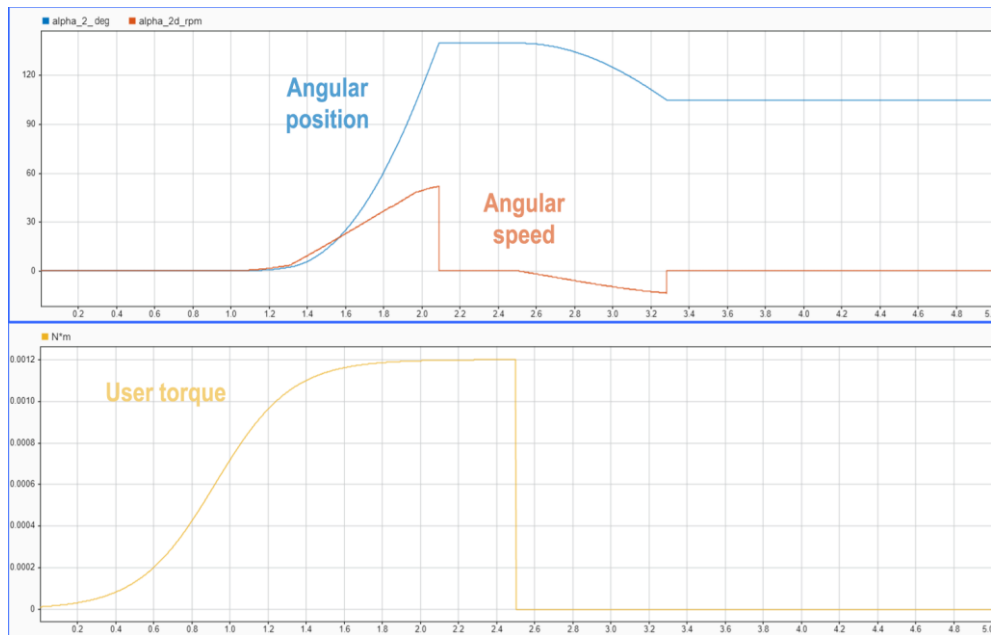


Figure 36: The angle position (degree vs time), the angular speed (rpm vs time) and the user torque (N*m) in the configuration with Accessory

The Figure 35 and Figure 36 is a comparison between the angle position, the angular speed and the applied user torque. The angular rotation starts when the user torque value is as big as required to win the friction torque. There is a slowing of speed growth when the angular spring starts to work (Run position).

The elastic torque is responsible for the rotation from Start to Run when the key is released (time = 2.5 s).

Starting from the analysis of the graphs, the transition time from a position to another is the information to obtain for the electrical simulation.

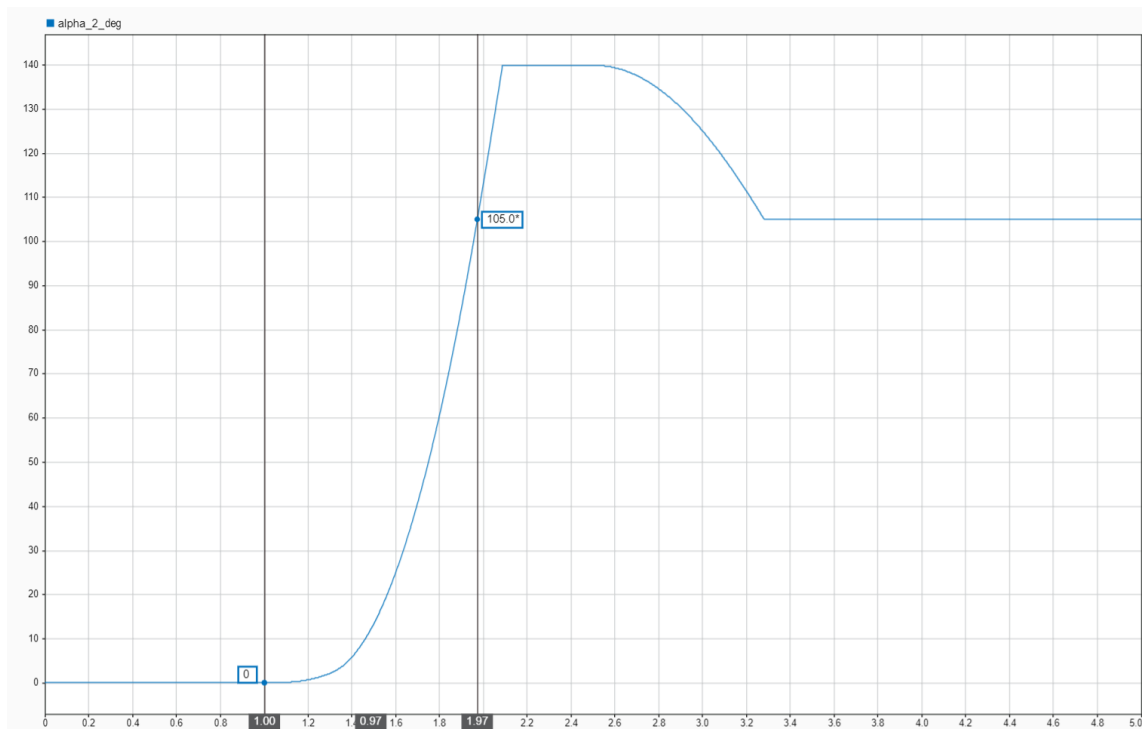


Figure 37: Transient time from Ignition lock to Run (configuration without Accessory)

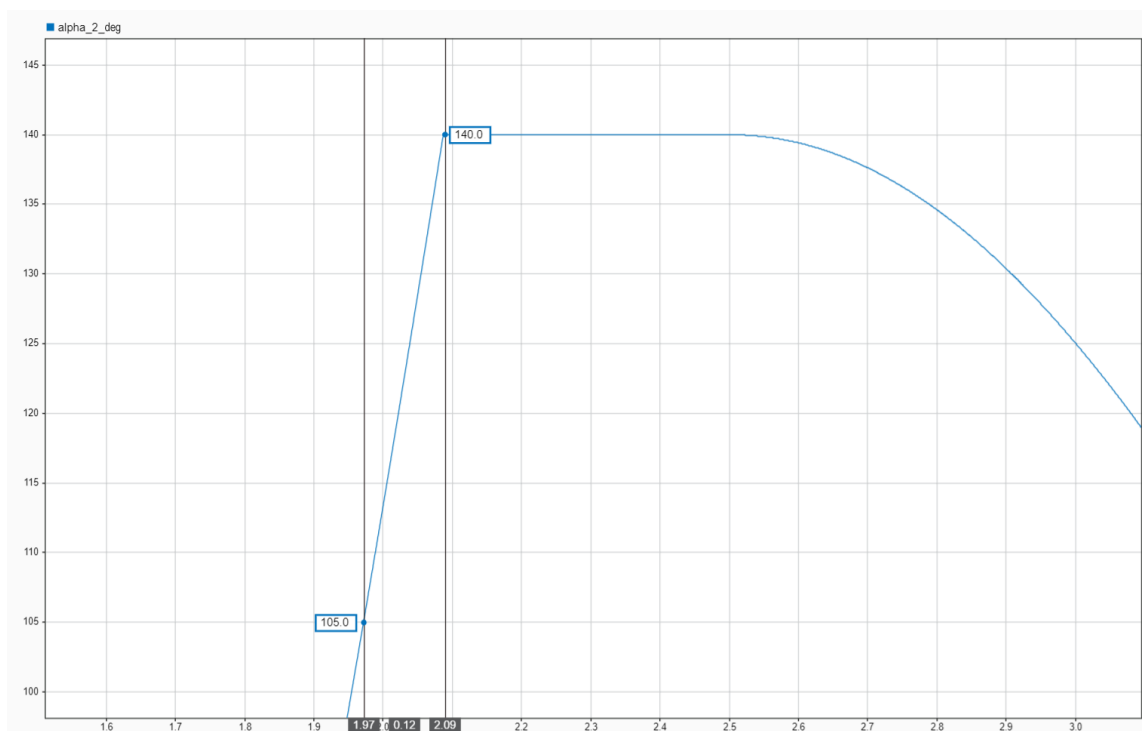


Figure 38: Transient time from Run to Start (configuration without Accessory)

In the following table there are the holding times for each key position with the configuration without Accessory.

Key position transition	Transient time
From Ignition lock to Run	T1 = 1.97 s
From Run to Start	T3 = 0.12 s
Hold in Start	T4 = 0.41 s (key released at 2.5 s)

Table 2: Holding time without Accessory position

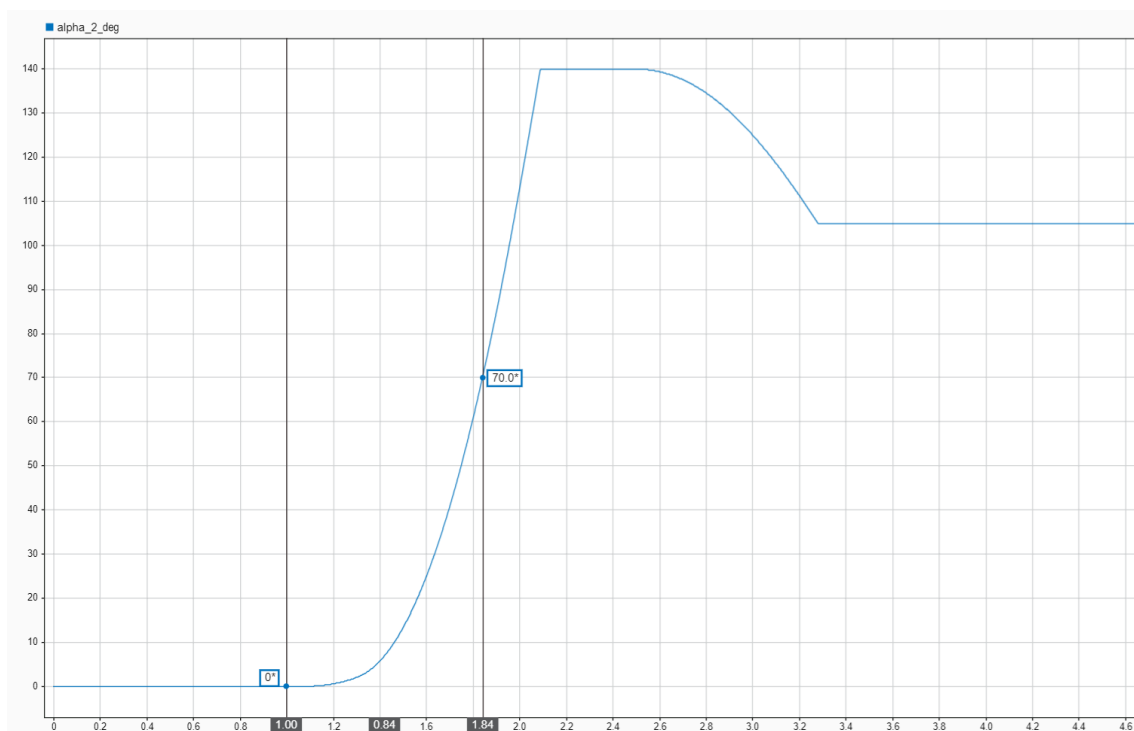


Figure 39: Transient time from Ignition lock to Accessory (configuration with Accessory)

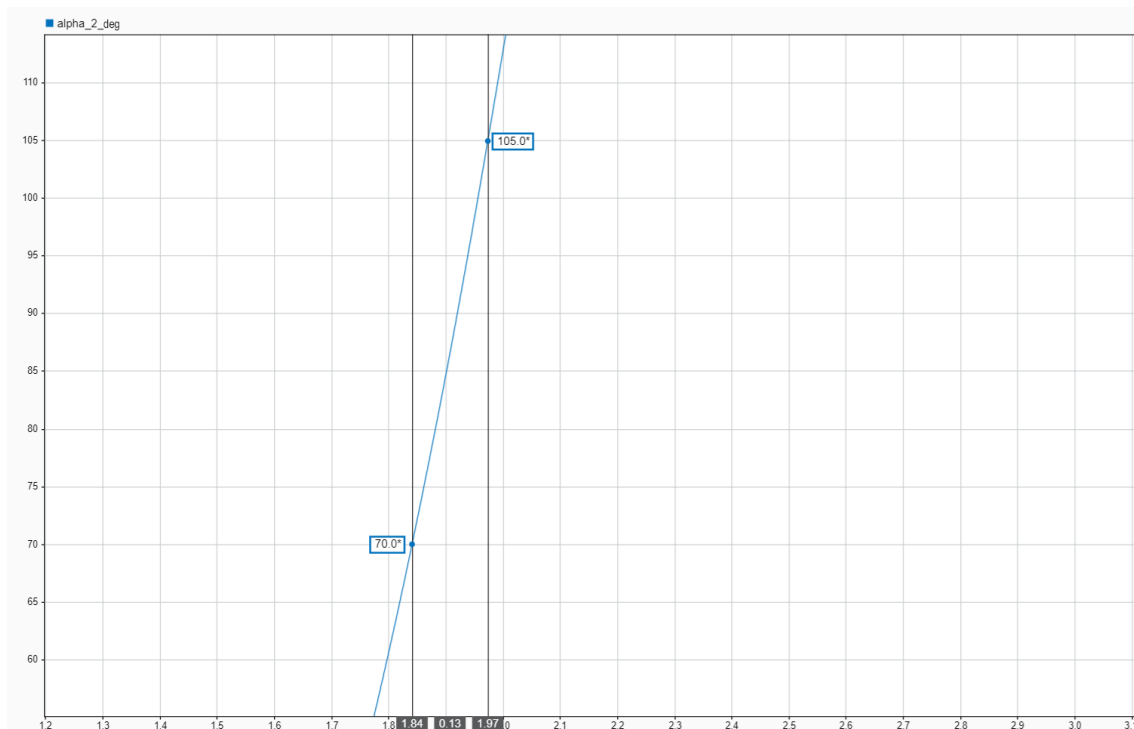


Figure 40: Transient time from Accessory to Run (configuration with Accessory)

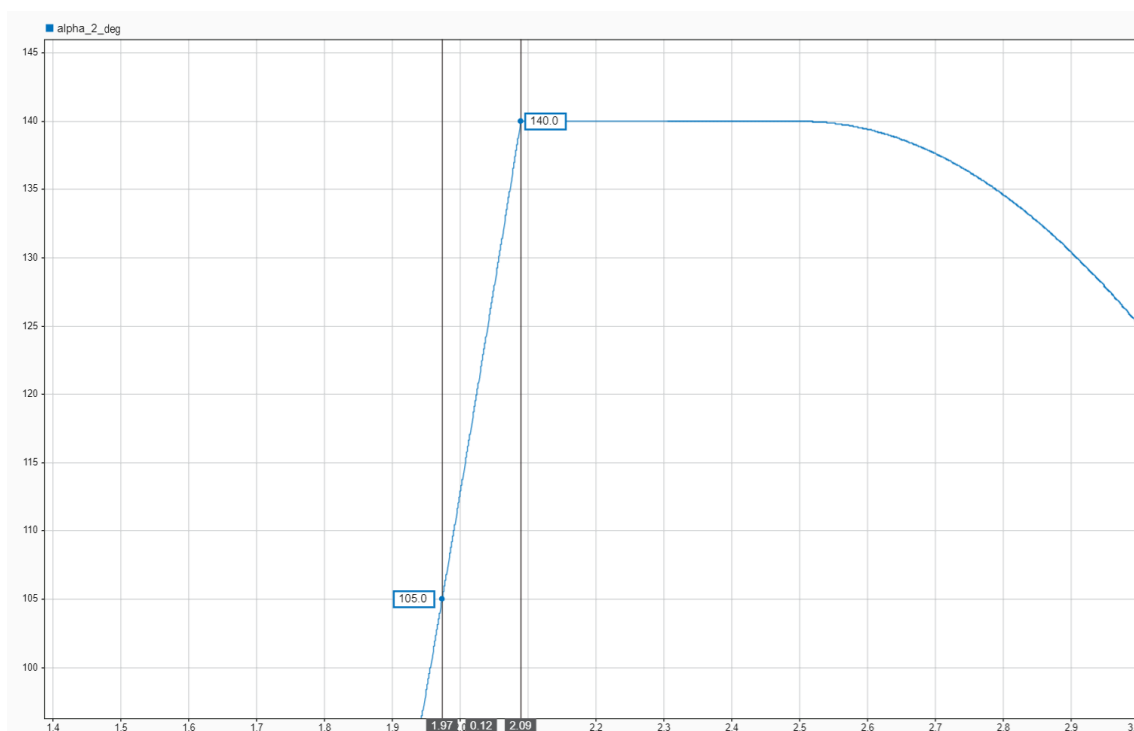


Figure 41: Transient time from Run to Start (configuration with Accessory)

In the following table there are the holding times for each key position with the configuration with Accessory.

Key position transition	Transient time
From Ignition lock to Accessory	T1_A = 1.84 s
From Accessory to Run	T2_A = 0.13 s
From Run to Start	T3_A = 0.12 s
Hold in Start	T4_A = 0.41 s (key released at 2.5 s)

Figure 42: Holding time with Accessory position

The transient time values are used as input in the electrical simulation in PSpice described in the following paragraphs.

3.3. Equivalent electric circuit

The key selector equivalent electric circuit is basically a voltage divider. The voltage divider is a circuit with two or more passive components connected in series in order to divide the applied voltage based on their value.

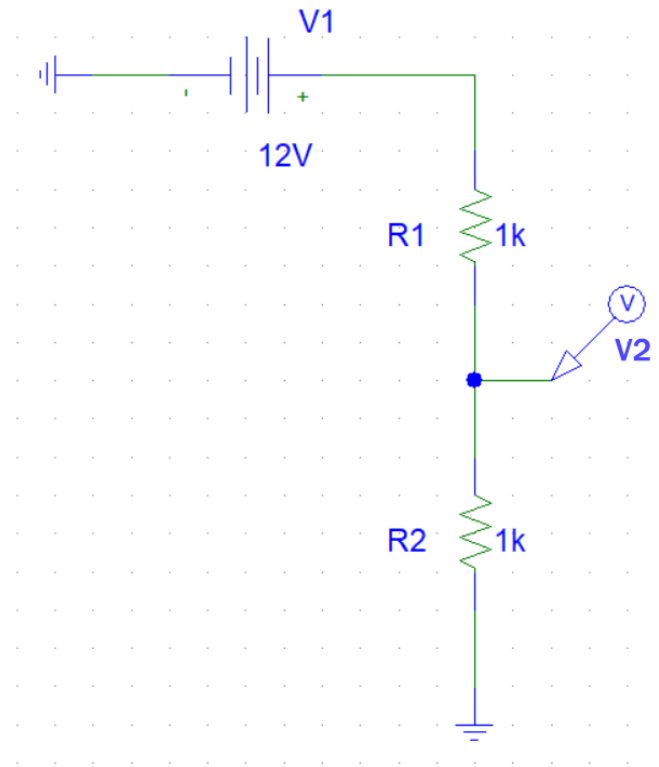


Figure 43: Voltage divider (example realized with PSpice)

Referring to the Figure 43 the transfer function is:

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2} \quad (16)$$

The following figure is the key selector equivalent electrical circuit.

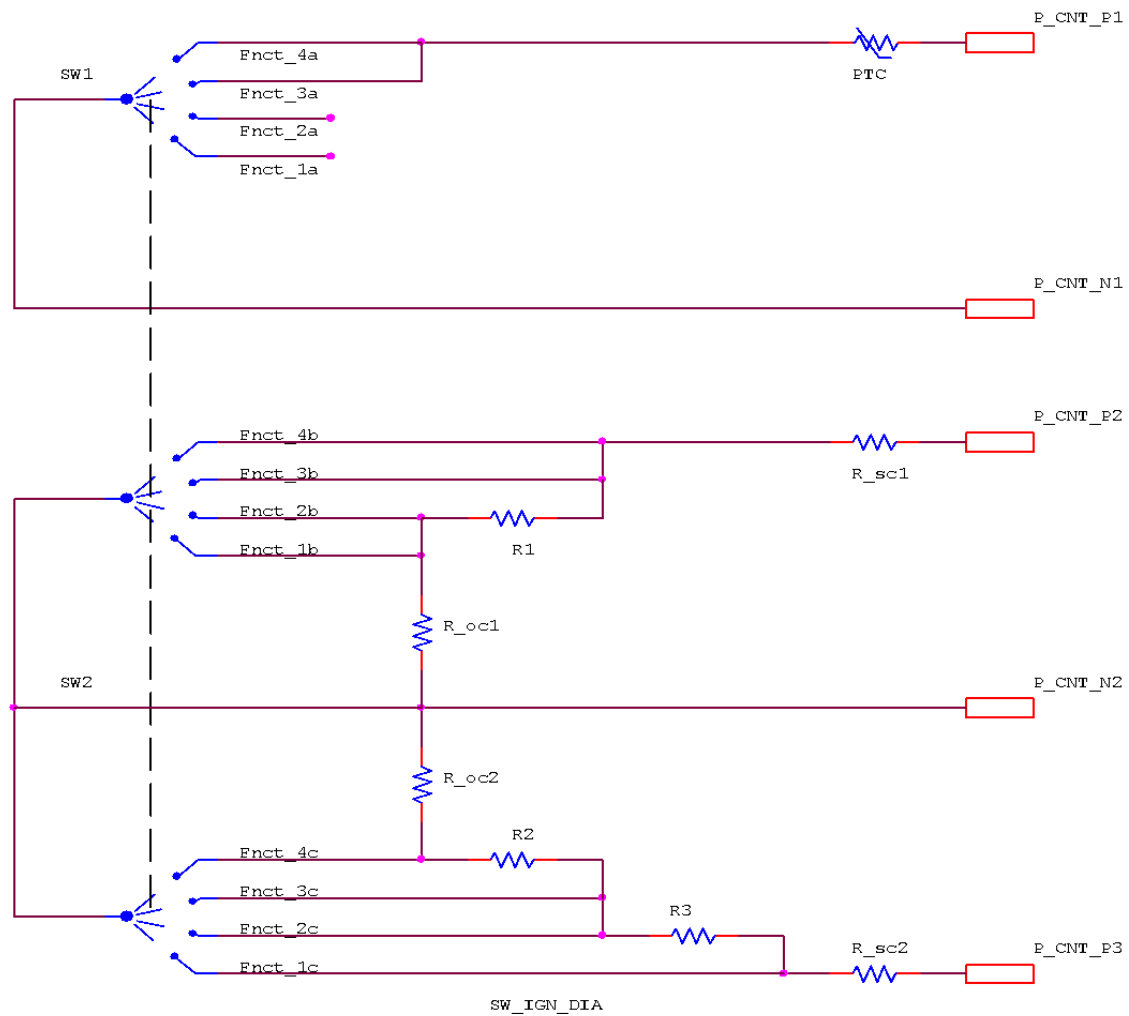


Figure 44: Key Selector (key selector switch) circuit

The sliding contacts are modelled through three multi-position switches; every switch position corresponds to a key position and the switch position changes accordingly to the key angle-position imposed by the user through the mechanical rotation.

Key position	Switches position
Ignition Lock	Fnct_1* (stable)
Accessory	Fnct_2* (stable)
Run	Fnct_3* (stable)
Start	Fnct_4* (temporary)

Table 3: Switches position vs key position

The Fnc2_* is absent in the configuration without the Accessory position.

Every switch position imposes a different voltage level at the ECU input pins (Pin1, Pin2 and Pin3). The results obtained are reported in the following table:

Nominal Key Position	Pin 1 [v]	Pin 2 [V]	Pin 3 [V]
NO_KEY	Directly connected to ground	$\frac{(R1 + Rsc1 + Roc1) \cdot Va}{R1 + Rsc1 + Roc1 + Rpu2}$	$\frac{(R2 + R3 + Rsc2 + Roc2) \cdot Va}{R2 + R3 + Rsc2 + Roc2 + Rpu3}$
IGN_LK	Directly connected to ground	$\frac{(R1 + Rsc1) \cdot Va}{R1 + Rsc1 + Rpu2}$	$\frac{Rsc2 \cdot Va}{Rsc2 + Rpu3}$
ACC	Directly connected to ground	$\left(\frac{R1 + Rsc1}{R1 + Rsc1 + Rpu} \right) \cdot Va$	$\left(\frac{Rsc2 + R3}{Rsc2 + R3 + Rpu} \right) \cdot Va$
RUN	Directly connected to the supply voltage	$\left(\frac{Rsc1}{Rsc1 + Rpu} \right) \cdot Va$	$\left(\frac{Rsc2 + R3}{Rsc2 + R3 + Rpu} \right) \cdot Va$
START	Directly connected to the supply voltage	$\left(\frac{Rsc1}{Rsc1 + Rpu} \right) \cdot Va$	$\left(\frac{Rsc2 + R3 + R2}{Rsc2 + R3 + R2 + Rpu} \right) \cdot Va$

Table 4: Equivalent impedance and pin voltage per position

	Voltage value read by the ECU (nominal condition)		
Key selector position	V_P_CNT_N1	V_P_CNT_P2	V_P_CNT_P3
NO_KEY	0 V	3.66 V	3.756 V
IGN_LK	0 V	2.133 V	1.019 V
ACC	0 V	2.133 V	1.725 V
RUN	12 V	1.019 V	1.725 V
START	12 V	1.019 V	2.518 V

Table 5: Pin voltage value per position

The output of the equivalent circuit is a set of three voltage values measured at the three different pins. These values are the input of the ECU and they are called:

- *IGN_IgnitionSW_K115* (voltage value measured on pin 1);
- *IGN_IgnitionSW_MainSignal* (voltage value measured on pin 2);
- *IGN_IgnitionSW_PlausibilitySignal* (voltage value measured on pin 3).

3.4. Electrical model in PSpice

3.4.1. PSpice

SPICE (Simulation Program with Integrated Circuit Emphasis) is a software for the circuit simulation.

PSpice is a commercial version of SPICE with software for the design of electrical circuits (*Schematics* and *Capture*) and for viewing results.

The model obtained with the using of this software is only an approximation of the real behaviour of the keyselector. For the realization of the circuit model solved by SPICE, it is considered the voltage as the difference between the two corresponding potential nodes.



Figure 45: PSpice logo

In this study, "Schematics" is used to design the circuit with different components that are available in the library. After connecting all the components, their characteristic values are set.

When there are errors in the connections of the circuit, the software detects them. The errors are reported in a different window where their typology and their location are shown to solve them quickly.

In *Schematics* it's possible to set different types of simulation, as the *transient simulation* that is used in this study for the generation of the electrical signals.

The *transient simulation* computes the circuit's response as a time function. It stops after a set time.

For the realization of the equivalent circuit they are used the following PSpice library elements:


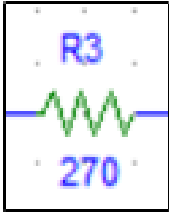
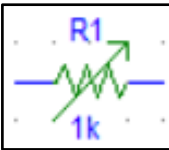
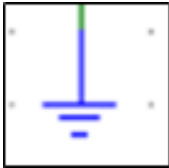
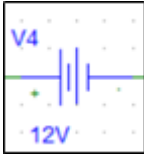
 <p>Figure 46: Switch symbol</p>	<p>The switch allows the connection between two parts of the circuit in which it is. There are two types of switch: normally open or normally closed. The first one is initially open, but it closes after a certain time (t_{Close}), connecting the two parts of the circuit. The second one is closed, but it opens after a certain time (t_{Open}), disconnecting the two parts of the circuit.</p>
 <p>Figure 47: Resistance symbol</p>	<p>The resistance is a component type that opposes the passage of current. Ohms (Ω) indicate the value of the resistance.</p>
 <p>Figure 48: Variable resistance symbol</p>	<p>The variable resistance is a resistor that changes its value over time, it could happen in a mechanical way (as with a potentiometer) or in a parametric way (as with PTC).</p>
 <p>Figure 49: Ground symbol</p>	<p>The ground usually identifies the part of the circuit which has the minor potential. The terminal generally corresponds to the negative pole of the battery.</p>
 <p>Figure 50: DC voltage source symbol</p>	<p>The DC voltage source is an ideal part of the circuit that can maintain a fixed voltage value.</p>

Table 6: PSpice library elements used for the electrical model

3.4.2. PSpice circuit (first simulation)

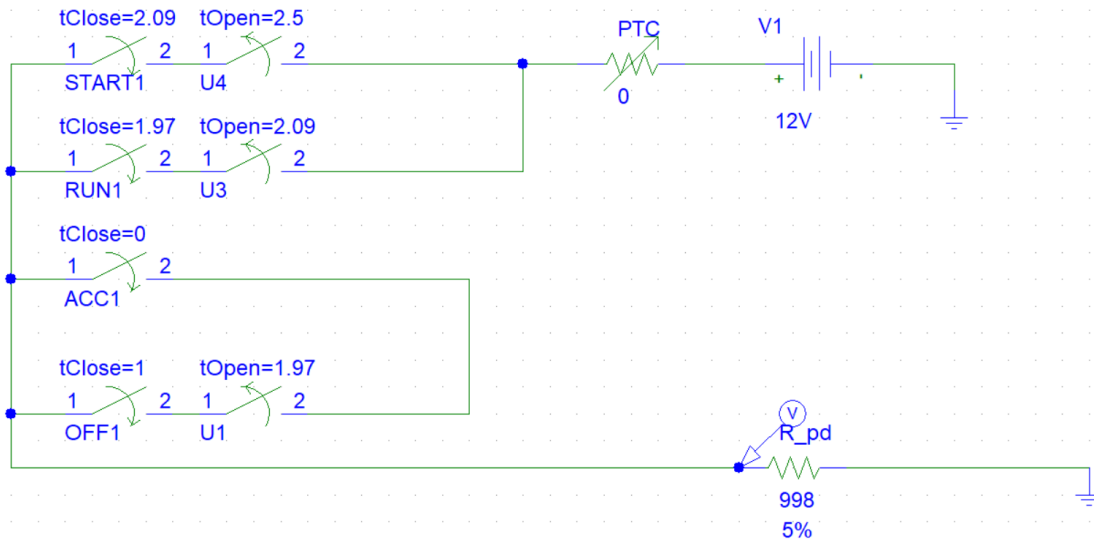


Figure 51: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (without Accessory)

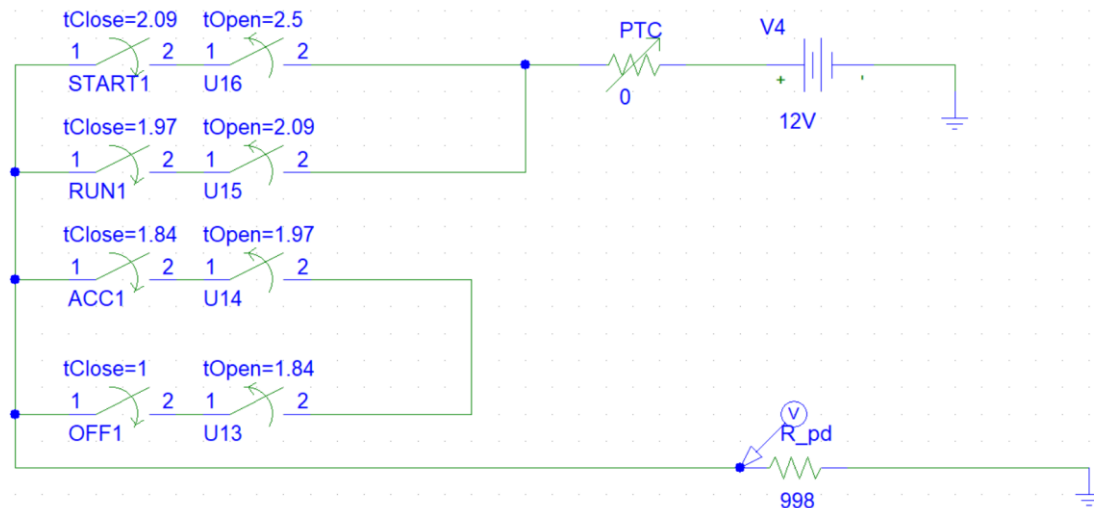


Figure 52: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (with Accessory)

The multi-position switch is simulated through parallel sets of two normally open and normally closed switches in series; they are visible on the left side of the Figure 51 and Figure 52.

The normally closed switches and the normally open one in series allow to simulate the multi fingers behaviour. The opening and closing times are set to simulate the voltage values vs time during the transition time from a key position to the other. The opening and closing times values are the ones resulting from the mechanical model simulation.

The Pin1 voltage is the voltage drop across the resistance R_{pd} . The Pin1

voltage can assume only two values: 0V (ground) and 12 V (battery).⁶

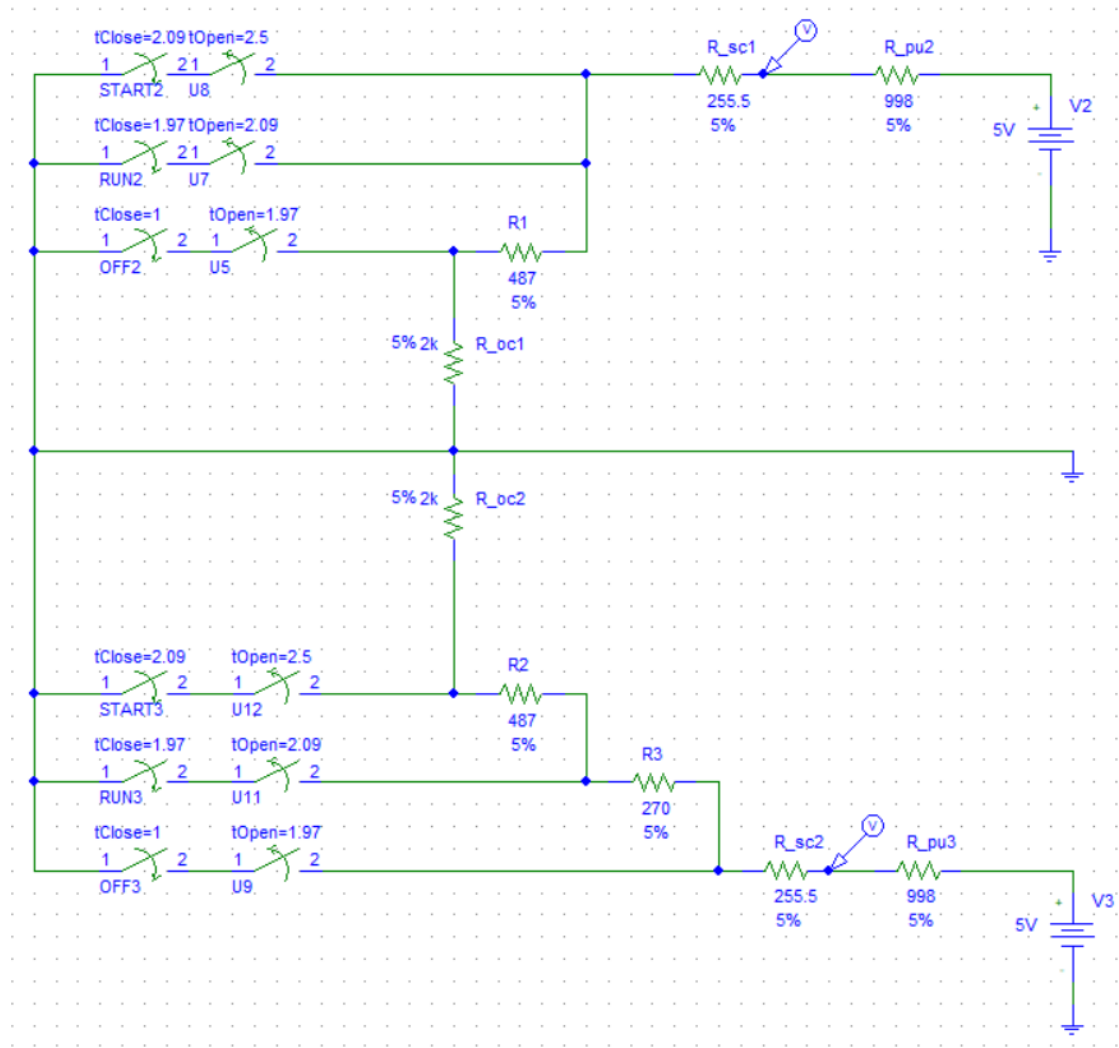


Figure 53: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (without Accessory position)

⁶ Refer to paragraph Equivalent electric circuit

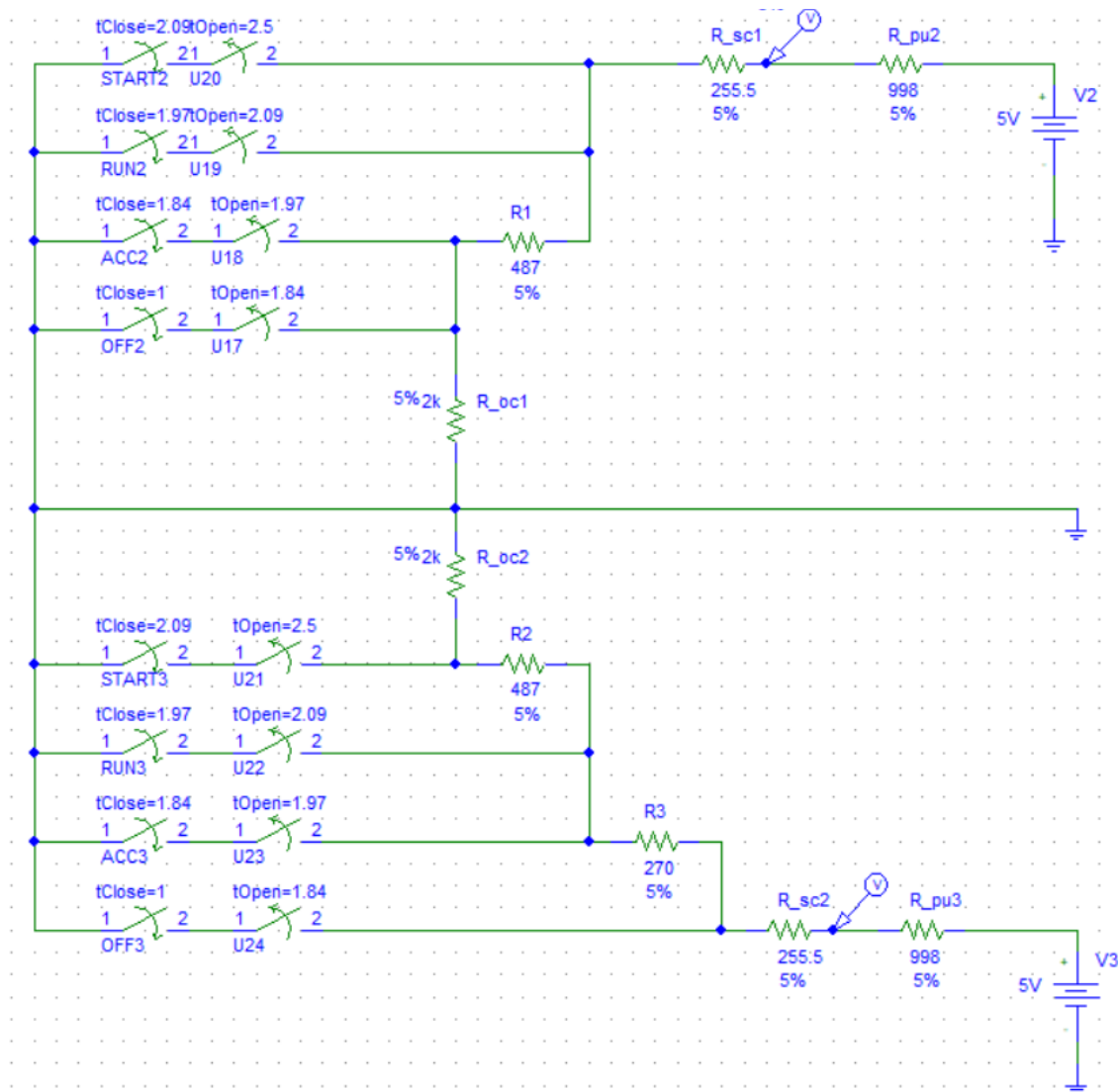


Figure 54: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (with Accessory position)

The Figure 53 and Figure 54 are the circuit parts corresponding to the Pin2 and the Pin3 in both configurations. For these pins, the circuit is connected to voltage regulators that generate a voltage of 5 V.

The Pin2/Pin3 voltage is the voltage drop across the equivalent resistive impedance between the Pin2/Pin3 and the ground.

The Pin2/Pin3 voltage can assume different values.⁷

⁷ Refer to the paragraph Equivalent electric circuit.

3.4.3. Key selector signal (first simulation)

The first *transient simulation* result is the ideal three pin voltages; they are approximately a square wave. The voltage values change when the key position changes. The pins voltage values are exactly the ones calculated in the paragraph “Equivalent electric circuit”. The voltage values stay at a specific value for the time defined in the paragraph “Mechanical model simulation”⁸.

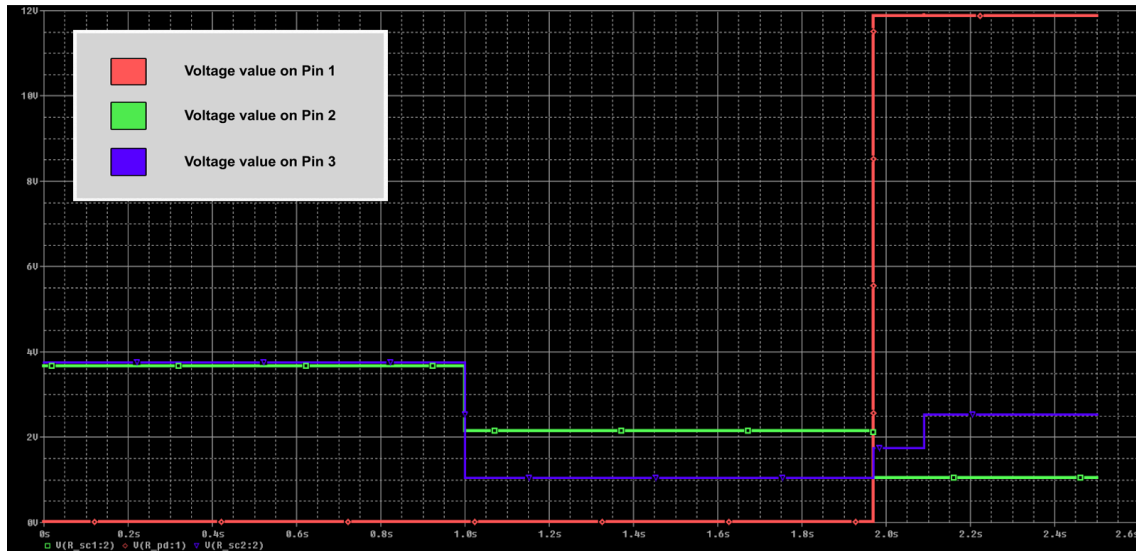


Figure 55: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration without Accessory

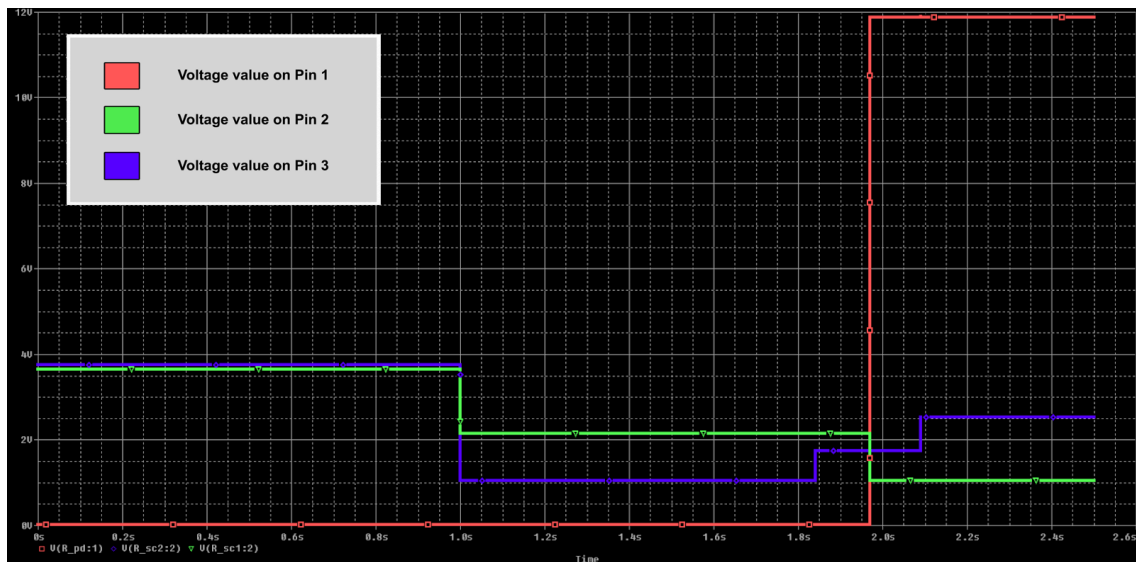


Figure 56: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration with Accessory

⁸ Refer to tables Table 2: Holding time without Accessory position and Table 5.

3.4.4. PSpice circuit (second simulation)

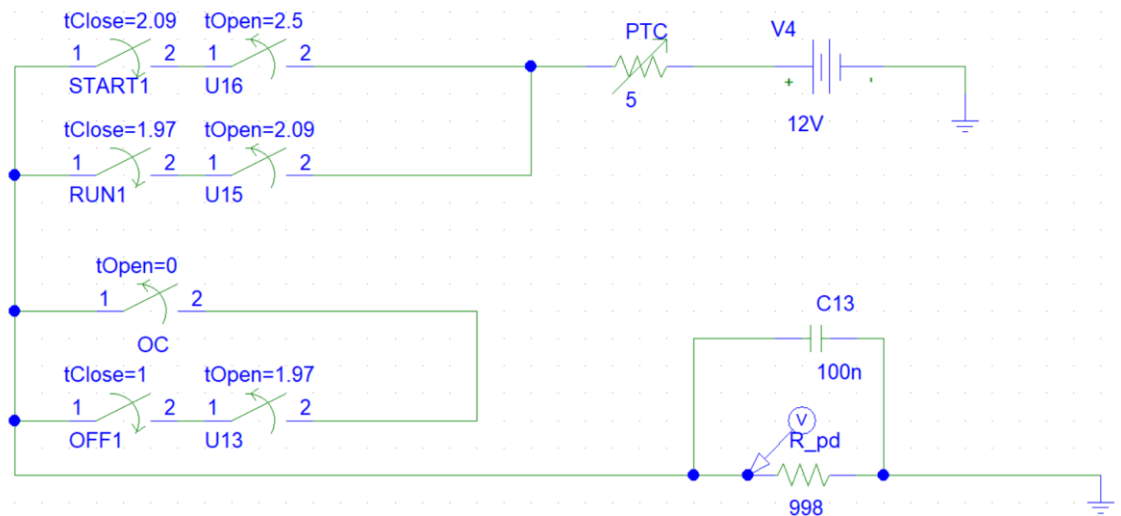


Figure 57: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (without Accessory position)

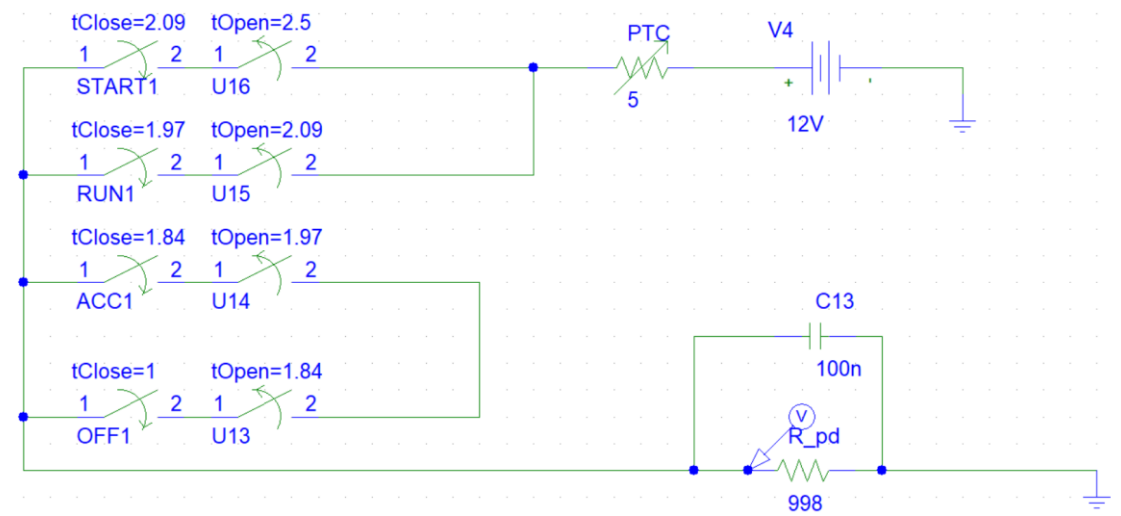


Figure 58: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (with Accessory)

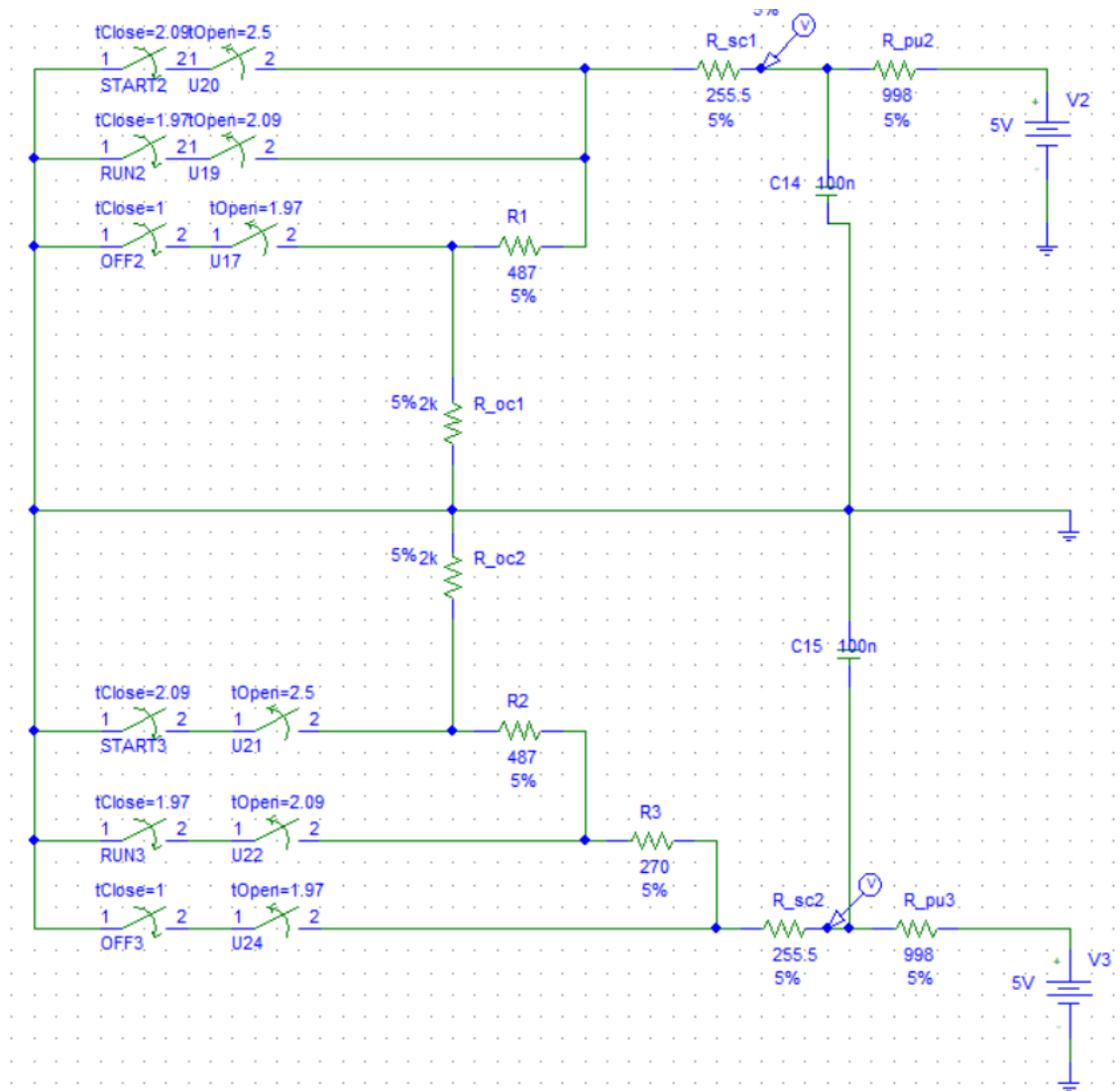


Figure 59: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (without Accessory position)

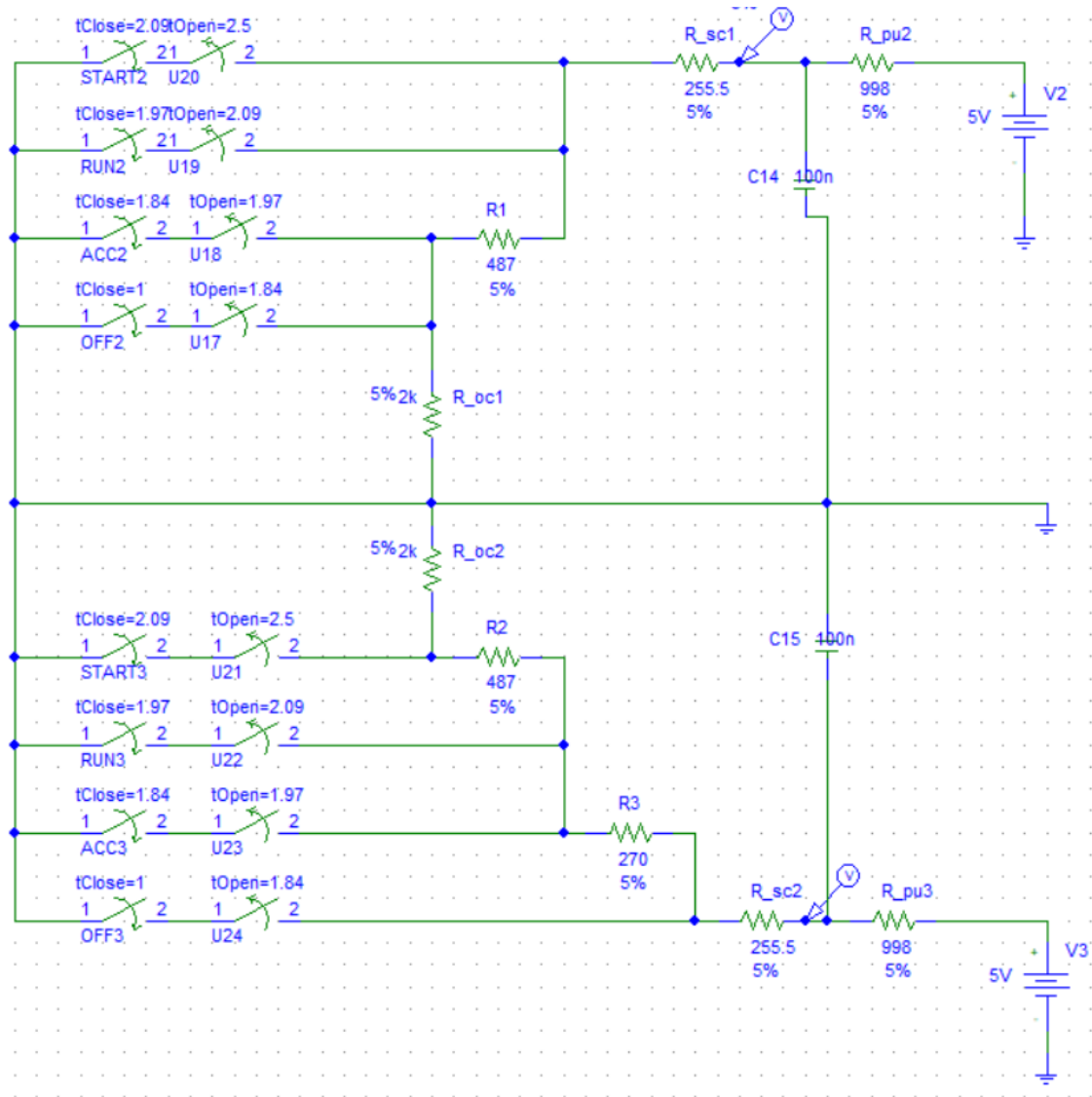


Figure 60: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (with Accessory position)

Then the circuit is changed to have more realistic signals:

A **capacitor** is added in parallel to the resistance to simulate the capacitive effect of the not ideal switches. The effect is a slower input voltage profile.

- All the contacts are made by materials with their own resistance effect that usually increases during the use. The switch **resistance** (10 Ω total) is set.
- Add a **transition time** for the normally open switch to simulate the rise and fall time of a real component.
- Add **PTC resistance** value that is used to protect the circuit.
- **10% resistor tolerance** is set.

3.4.5. Key selector signal (second simulation)

The second *transient simulation* result is a more realistic three pin voltages; some differences are visible in this configuration.

The switches transition time causes the spikes visible for every signal switch.

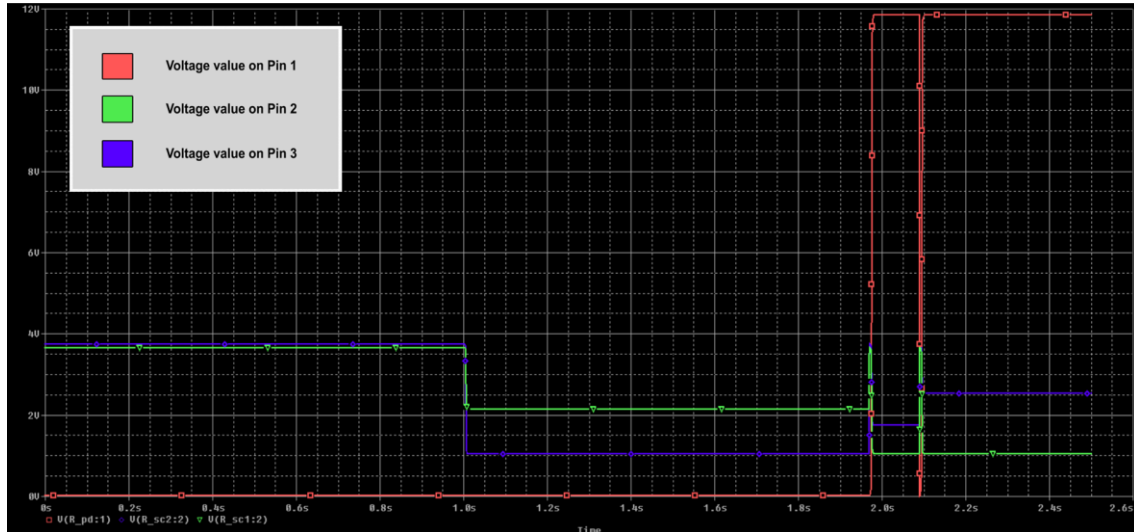


Figure 61: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration without Accessory



Figure 62: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration with Accessory

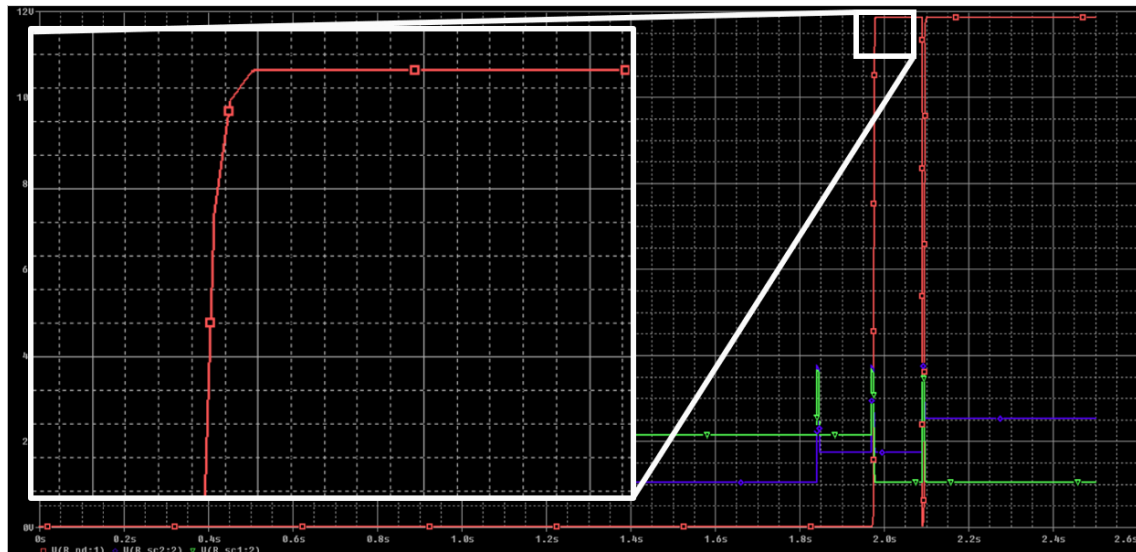


Figure 63: Particular of Pin 1 voltage signal in the configuration without Accessory

The capacitor's effect causes a smooth profile when each signal passes from a voltage level to another.

3.4.6. Montecarlo simulation

In *Schematics* it is possible to use the *Montecarlo simulation - Worst Case* that predicts the behaviour of a circuit statistically when components values are varied within their tolerance range, in this case the resistances. It is possible to set the *max* or the *min* of the function. In the simulations, the *max* and the *min* values are compared with the nominal one. The following figures report some of the results of the simulation.

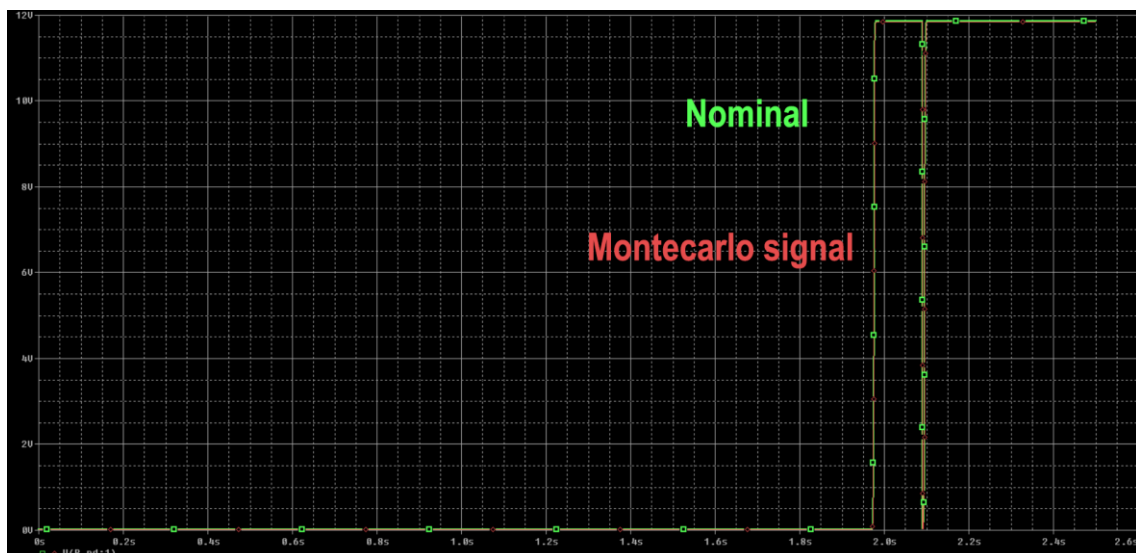


Figure 64: Montecarlo simulation (Min) of the voltage value on pin 1 without Accessory

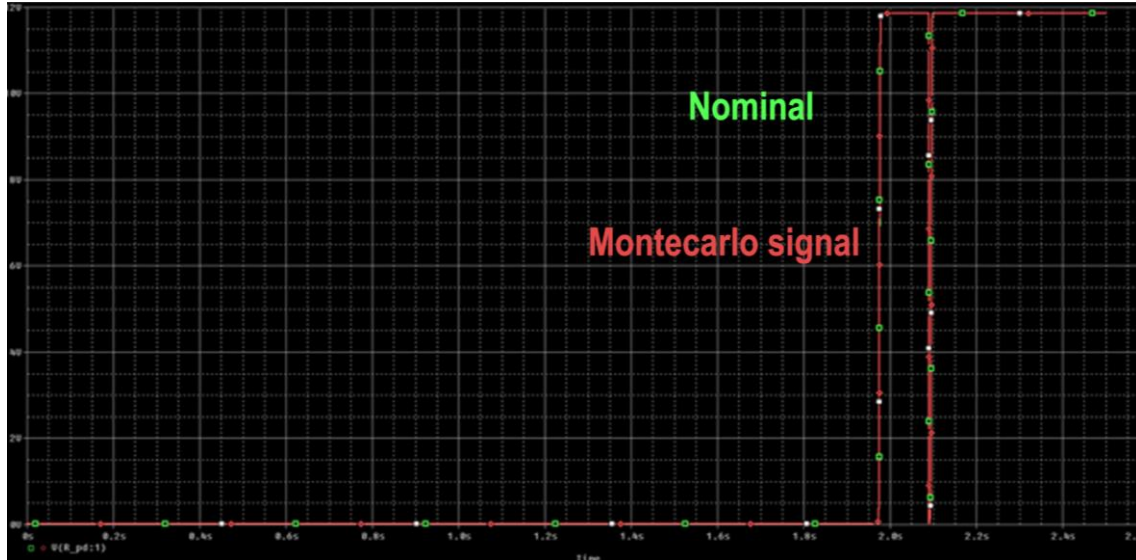


Figure 65: Montecarlo simulation (Max) of the voltage value on pin 1 with Accessory

The difference among the voltage levels min, nominal, max is not relevant in this case because the resistance tolerance has a minimum effect on the pin 1 voltage for both configurations.

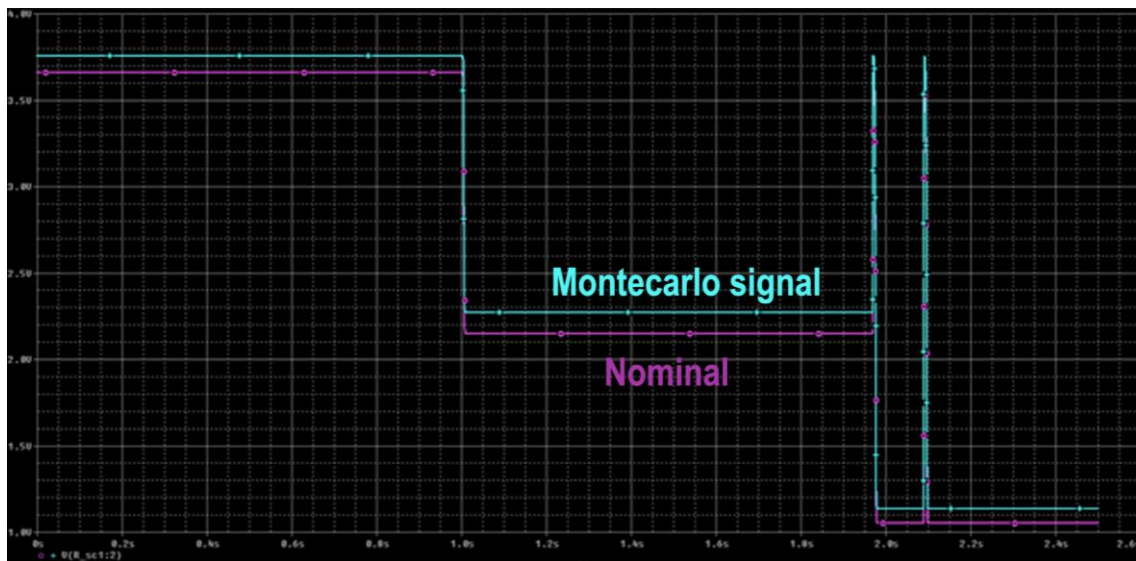


Figure 66: Montecarlo simulation (Max) of the voltage value on pin 2 without Accessory

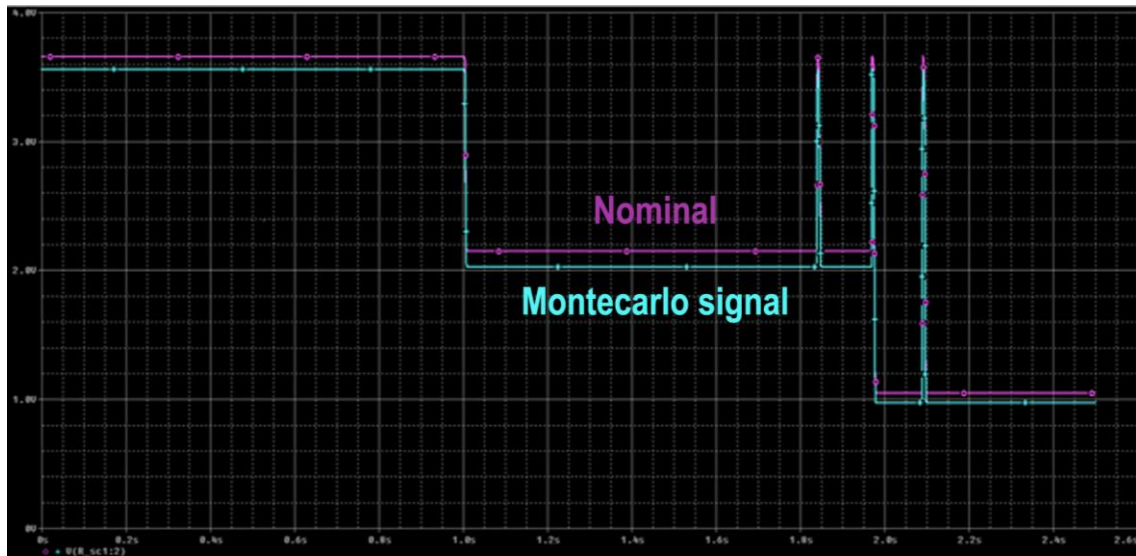


Figure 67: Montecarlo simulation (Min) of the voltage value on pin 2 with Accessory

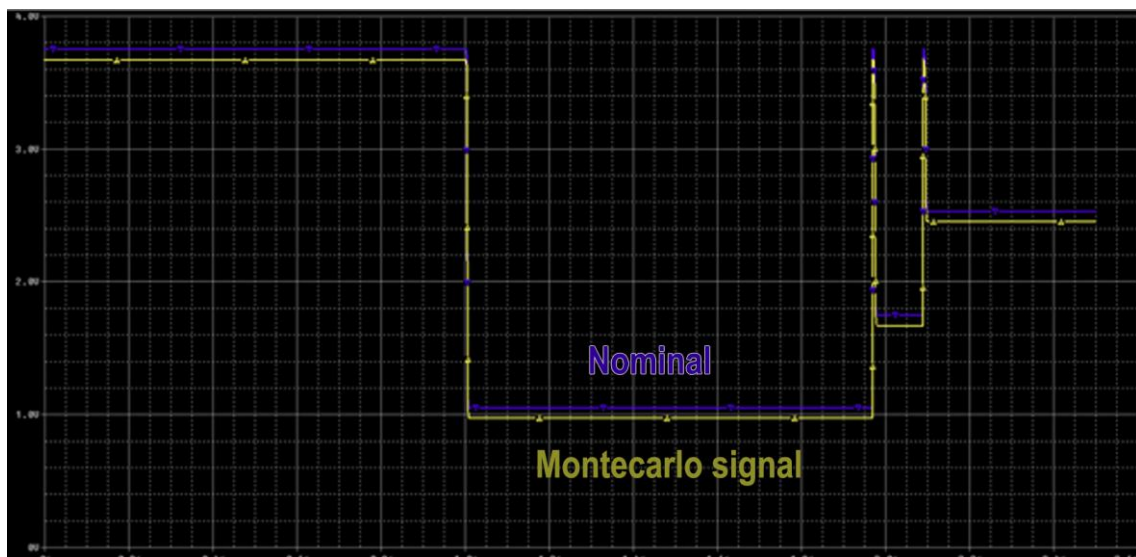


Figure 68: Montecarlo simulation (Min) of the voltage value on pin 3 without Accessory

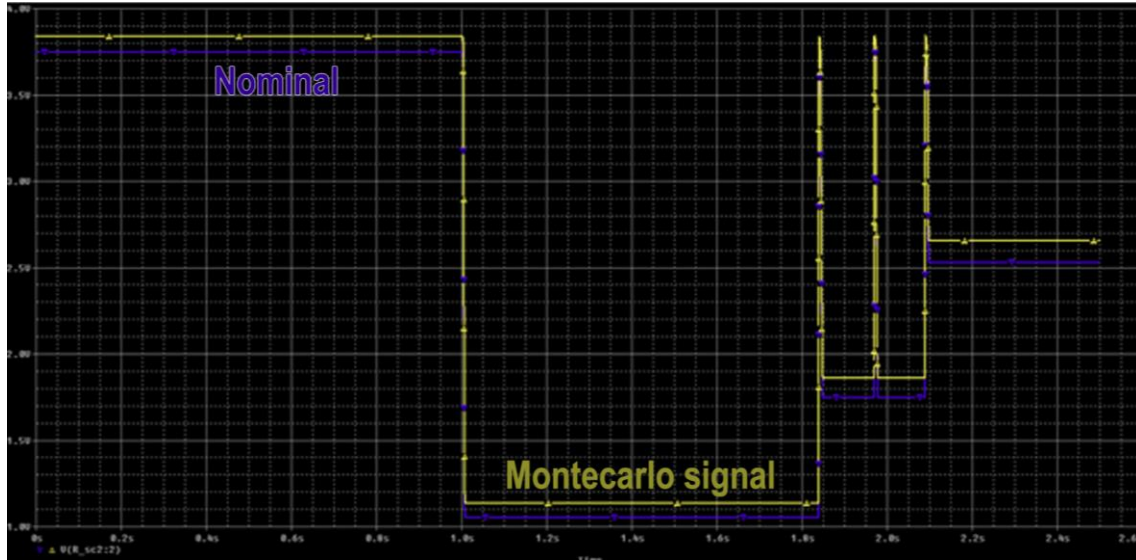


Figure 69: Montecarlo simulation (Max) of the voltage value on pin 3 with Accessory

The effect of the resistance tolerance is more evident on the pin 2 and the pin 3; the max value is clearly higher than the nominal one while the min value is clearly lower than the nominal one.

In the thesis, we choose to stimulate the model with the *min* function of the voltage signals in the configuration without Accessory and the *max* function of the voltage signals in the configuration with Accessory.

4. Algorithm description

4.1. General algorithm overview

The algorithm recognizes the key positions and the failure conditions.

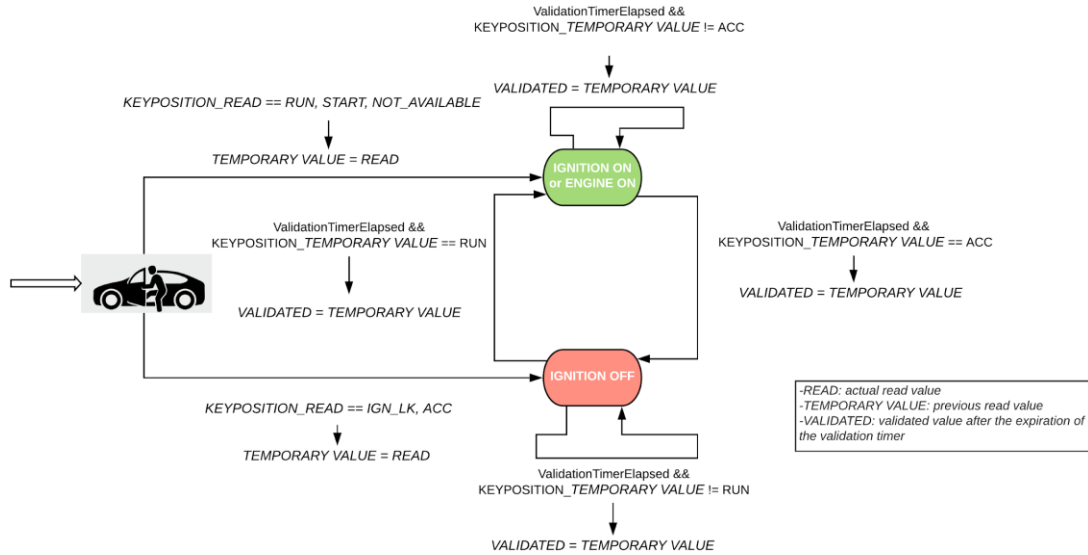


Figure 70: Algorithm description

The Figure 70 is the high-level description of the algorithm. The key selector voltage values are read and stored in a variable to be validated. When the validation time elapses the new key position is assigned as the final value. The transition from Ignition On / Engine On state to Ignition Off state and vice versa are driven by the validated value.

4.2. Input signals quantization

The input signals are analog signals; these analog values are mapped into a set of discrete finite values.

The quantization is done by setting a threshold between the voltage levels assigned to the key selector positions.

If the two voltage levels are x and y , the threshold is set:

$$Threshold = \frac{(x^+ + y^-)}{2}$$

- x^+ is the max value from Montecarlo simulation;
- y^- is the min value from Montecarlo simulation.

The ECU detects the signals from the pins and quantizes them according to the thresholds.

The input P1 signal range is from 0 V to 12 V.

This analog voltage signal is quantized into two values: *LOW* and *HIGH*.

If the voltage V is lower or equal than the threshold V_{P1_Th} then the input P1 is assigned the value *LOW*.

If the voltage V is higher than the threshold V_{P1_Th} then the input P1 is assigned the value *HIGH*.

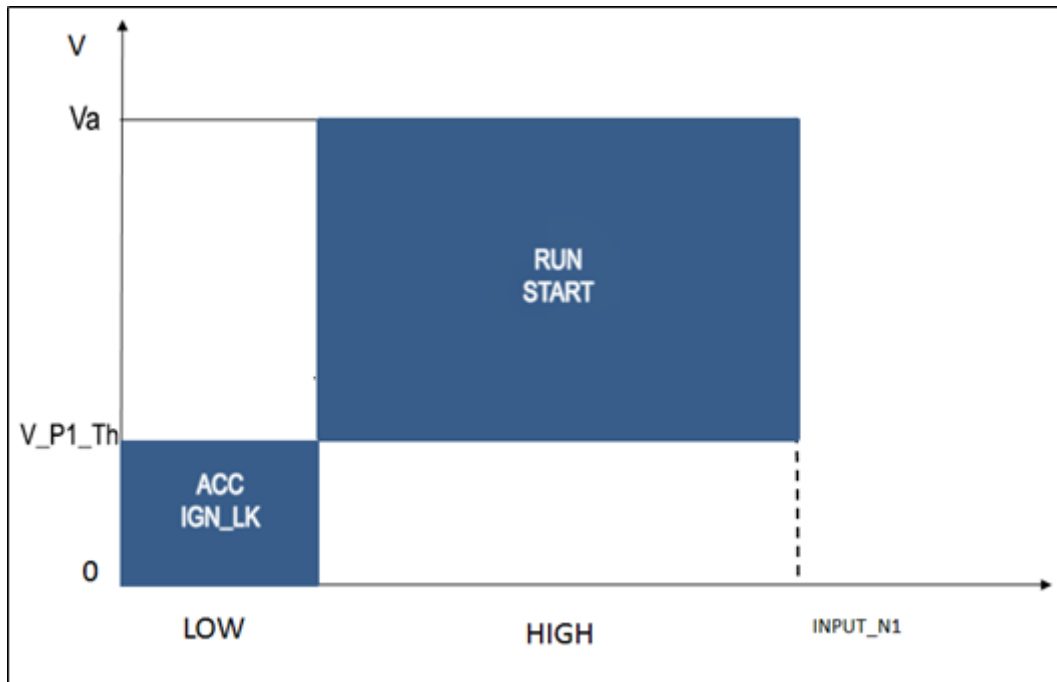


Figure 71: INPUT 1 value vs voltage value at pin 1

The value is *LOW* if the key selector is in *IGN_LK* or in *ACC* (where the *Accessory* position is present) without failure conditions.

The value is *HIGH* if the key selector is in *RUN* position or in *START* position without failure conditions.

The input P2 and P3 signal range is from 0 V to 5 V.

The input P2 analog voltage signal is quantized into four values: SctoGND, A, B, OC/SctoVbat.

If the voltage V is lower or equal than the threshold V_P2_Th1 then the input P2 is assigned the value SctoGND.

If the voltage V is lower or equal than the threshold V_P2_Th2 and higher than the threshold V_P2_Th1 the input P2 is assigned the value A.

If the voltage V is lower or equal than the threshold V_P2_Th3 and higher than the threshold V_P2_Th2 the input P2 is assigned the value B.

If the voltage V is higher than the threshold V_P2_Th3 then the input P2 is assigned the value OC/SctoVbat.

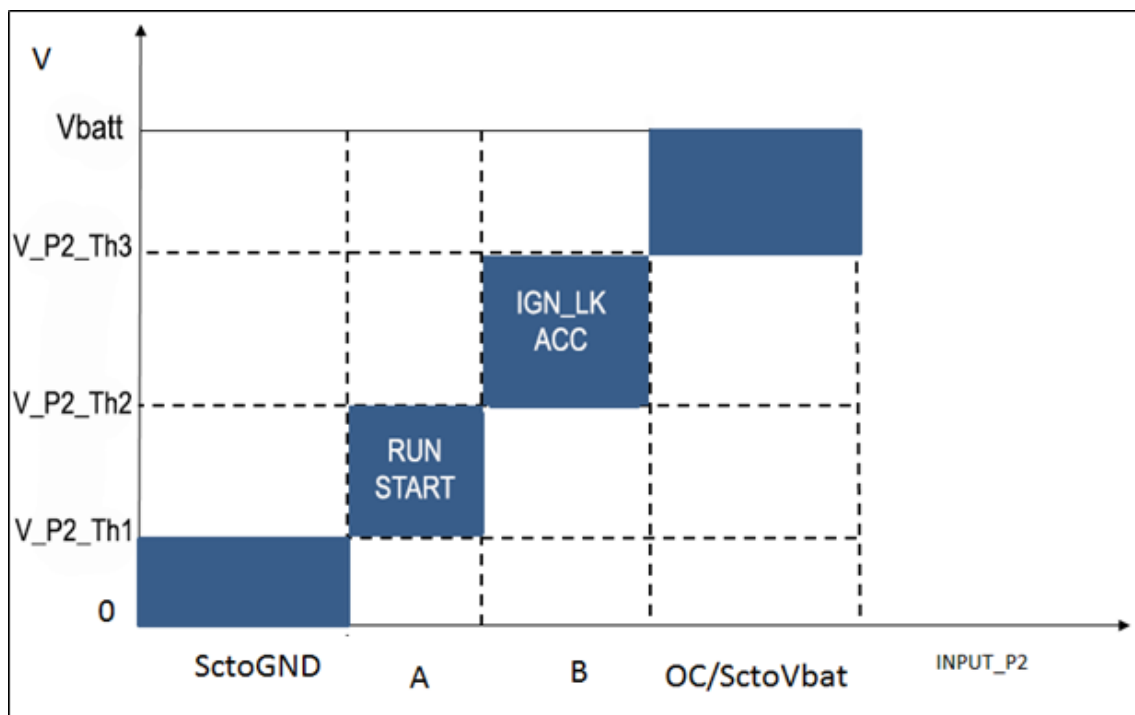


Figure 72: INPUT 2 value vs voltage value at pin 2

The value is *SctoGND* if the ECU detects the short circuit to ground condition in the pin 2.

The value is *A* if the key selector is in *RUN* position or *START* position without failure conditions.

The value is *B* if the key selector is in *IGN_LK* position or *ACC* (where the Accessory position is implemented) without failure conditions.

The value is *OC/SctoVbat* if the ECU detects the open circuit condition or the short circuit to VBat condition in pin 2.

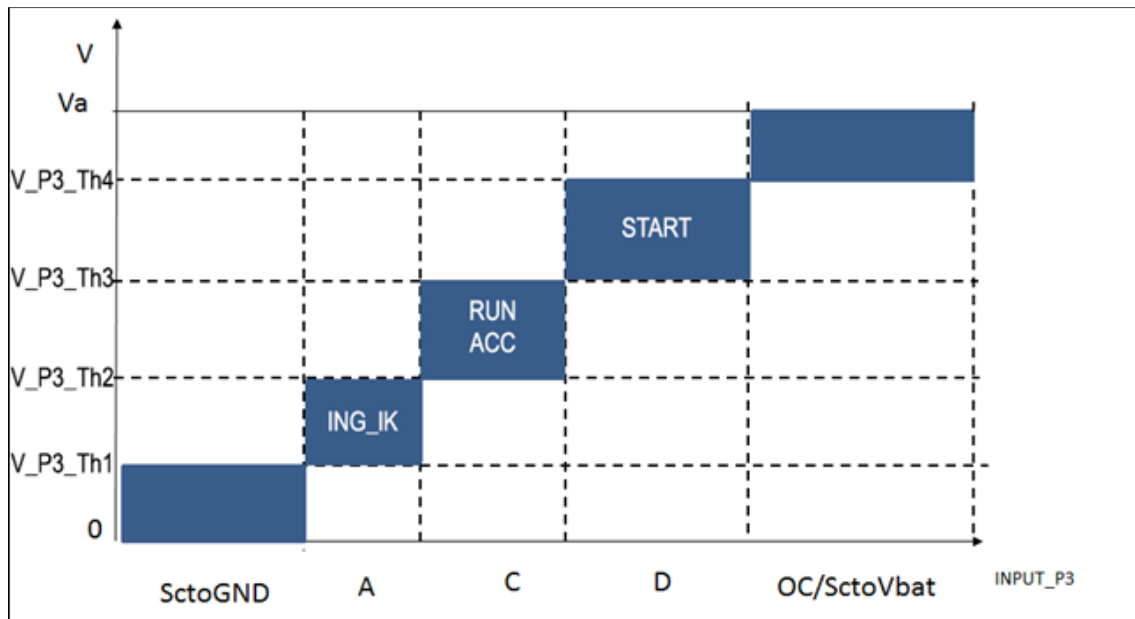


Figure 73: INPUT 3 value vs voltage value at pin 3

According to Figure 27 the input P3 can assume the values *SctoGND*, *A*, *C*, *D*, *OC/SctoVbat*.

The value is *SctoGND* if the ECU detects the short circuit to ground condition in pin 3.

The value is *A* if the key selector is in *IGN_LK* position without failure conditions.

The value is *C* if the key selector is in *RUN* position or *ACC* (where the *Accessory* position is implemented) without failure conditions.

The value is *D* if the key selector is in *START* position without failure conditions.

The value is *OC/SctoVbat* if the ECU detects the open circuit condition or the short circuit to battery condition in the pin 3.

The nominal condition is the use case without any failure conditions

	Voltage value read by the ECU (nominal condition)		
Key selector position set by the ECU (nominal condition)	INPUT_N1	INPUT_P2	INPUT_P3
IGN_LK	LOW	B	A
ACC	LOW	B	C
RUN	HIGH	A	C
START	HIGH	A	D

Table 7: Pin quantized voltage value per position

The key selector position is set to *NOT_AVAILABLE* when the key position is not detected by the algorithm or the key is not inserted by the user.

The algorithm sets the key position when at least two of the three read voltage values are congruent to the set of values expected in one of the nominal conditions.

If the key is not inserted by the user the voltage values P2 and P3 are the ones NO_KEY described in the Table 4. These values are not distinguished from the open circuit / short to battery condition according to the hypothesis of this study.

4.2.1. Key selector status examples

In the following figures some examples are reported for the configuration with Accessory position to highlight:

- how the hardware signals voltage values *IGN_IgnitionSW_KI15* (pin 1), *IGN_IgnitionSW_MainSignal* (pin 2), *IGN_IgnitionSW_PlausibilitySignal* (pin 3) change if the key selector position changes from *IGN_LK* position to *ACC* position or *START* position or *RUN* position;
- the key selector positions and failures that should be recognized by the ECU according to the input signals value.

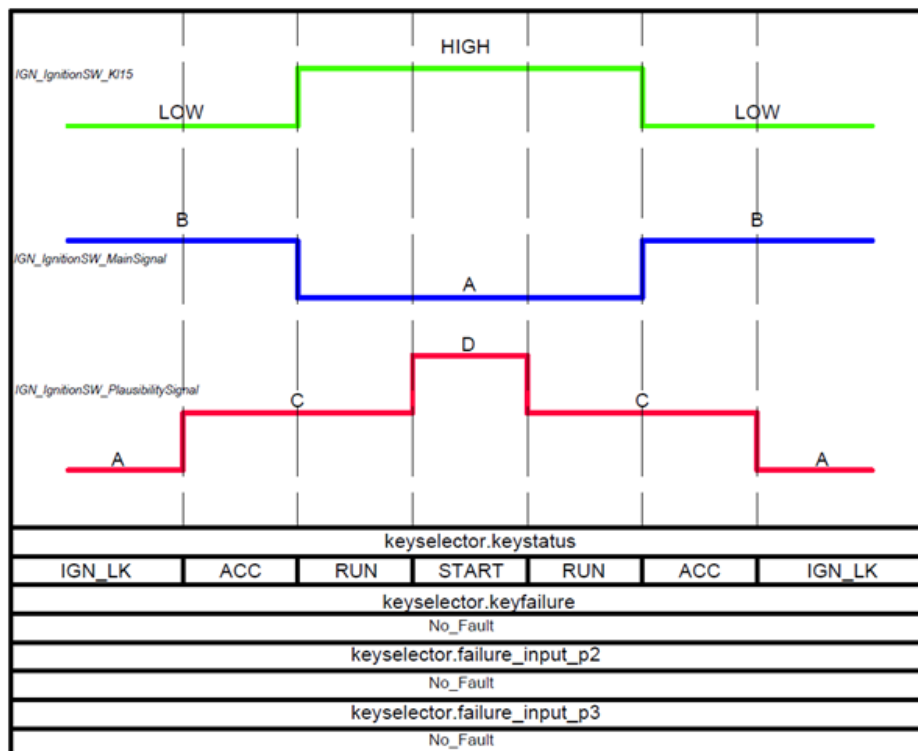


Figure 74: Typical

The algorithm should filter the input signals so that the key positions or the failure conditions are recognized even if there is a micro interruption.

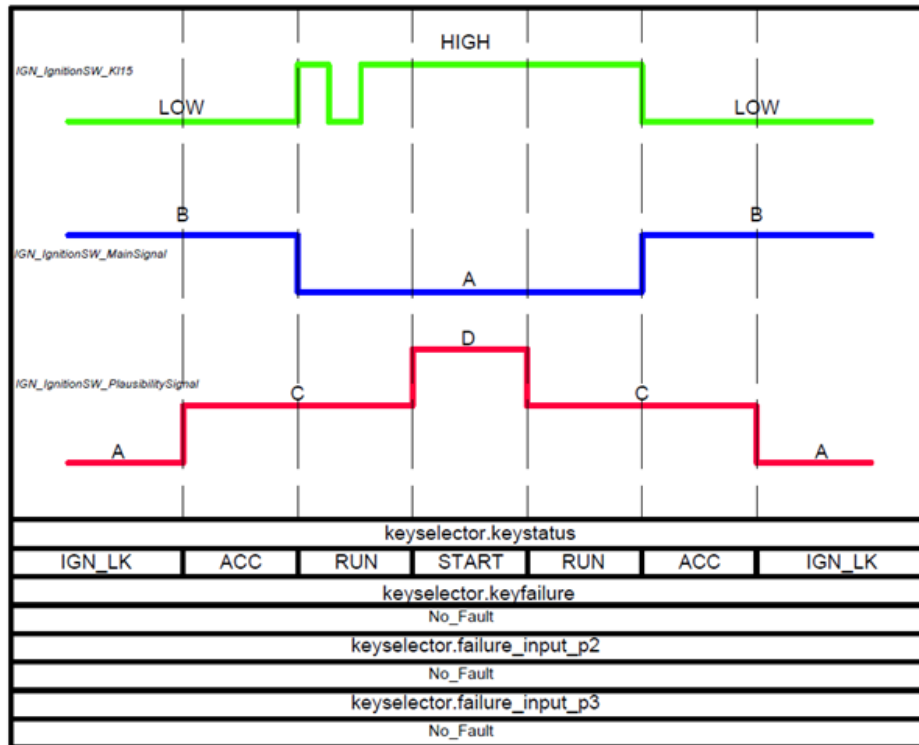


Figure 75: IGN_IgnitionSW_KL15 is disturbed for a short time

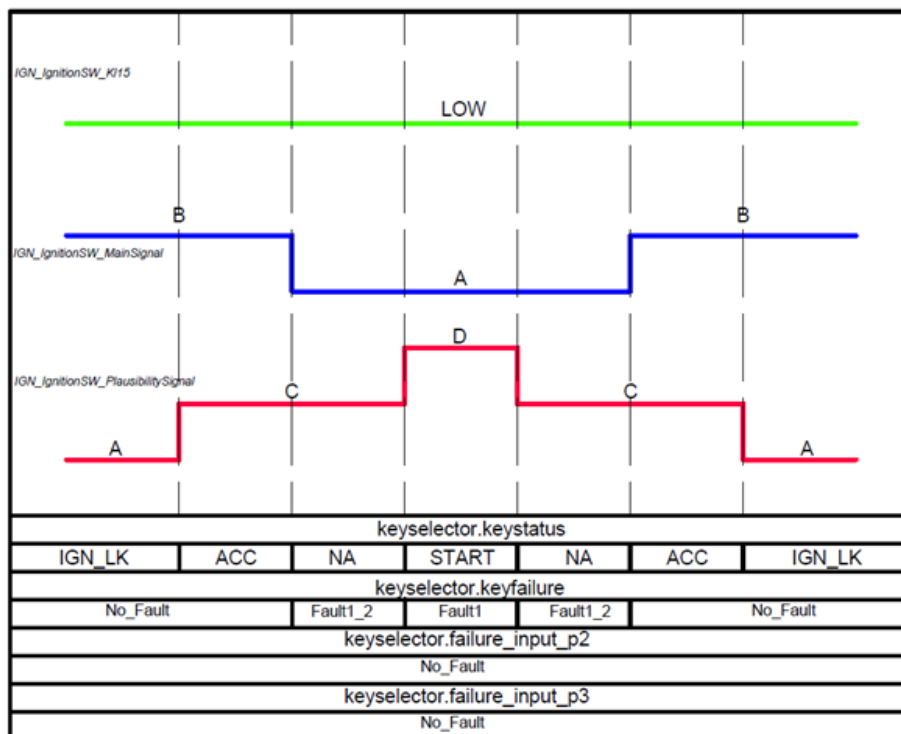


Figure 76: IGN_IgnitionSW_KL15 is SctoGND

In Figure 76 the pin1 short circuit to ground condition cannot be detected, but the key positions are recognized because the other two signals assume nominal values and they are coherent.

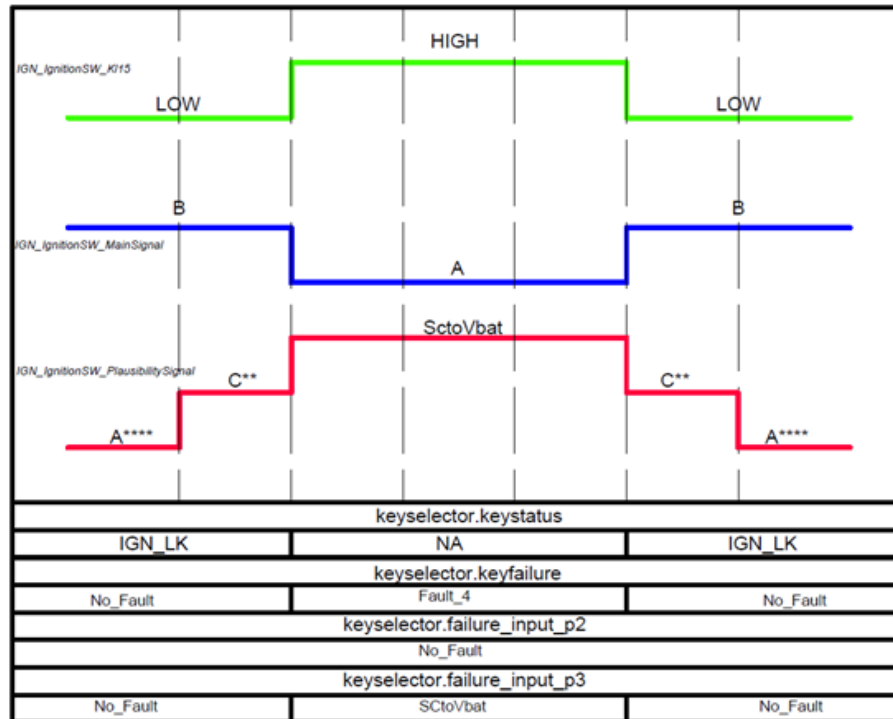


Figure 77: IGN_IgnitionSW_KL15 and IGN_IgnitionSW_PlausibilitySignal are shorted

The ECU detects the key status equal to *IGN_LK*, but the next position is set to *NOT_AVAILABLE* because it cannot be identified. In the figure 11 the pin 1 and pin 3 are shorted.

*A***** identifies the voltage value at the pin *P_CNT_P3* if the signals *IGN_IgnitionSW_KL15* and *IGN_IgnitionSW_PlausibilitySignal* are shorted.

*C*** identifies the voltage value at the pin *P_CNT_P3* if the signals *IGN_IgnitionSW_KL15* and *IGN_IgnitionSW_PlausibilitySignal* are shorted.

*A***** could be detected by the ECU as *A* or *SCtoGND* assigned to the variable *INPUT_P3*. It depends on the voltage threshold *V_P3_Th1*.

*C*** could be detected by the ECU as *A* or *C* assigned to the variable *INPUT_P3*. It depends on the voltage threshold *V_P3_Th2*.

In this use case the ECU recognizes only the *IGN_LK* position.

4.3. Algorithm structures and variables

Three different structures are declared to manage the quantized voltage values in the algorithm implementation.

The structure contains four variables:

- *KeySelectorPosition*: the key position and the values are IGN_LK, RUN, START, NOT_AVAILABLE⁹;
- *FaultInfo*: the general failure condition and the values are in the Table 8;
- *Fault_P2*: the failure condition on pin 2 and the values are in the Table 9;
- *Fault_P3*: the failure condition on pin 3 and the values are in the Table 9.

Fault_1	The value on pin 1 is not one of the possible values in nominal conditions, while pin 2 and pin 3 are possible nominal values
Fault_2	The value on pin 2 is not one of the possible values in nominal conditions, while pin 1 and pin 3 are possible nominal values
Fault_3	The value on pin 3 is not one of the possible values in nominal conditions, while pin 1 and pin 2 are possible nominal values
Fault_4	An electric fault: short to ground, short to battery or open circuit
Fault_5	Two electric faults or a combination of the previous faults

Table 8: key failure conditions

SCtoGND	The pin is shorted to ground
SCtoVbat	The pin is shorted to the battery voltage or there is an open circuit

Table 9: Pin 2 and Pin 3 failure conditions

The structures are:

- *keyselector_input_read*: the structure contains the un-debounced key status and the un-debounced key selector failures read by the ECU;
- *keyselector_input_stored*: the structure contains the saved key status under validation and the saved key selector failures under validation;
- *keyselector*: the structure contains the key selector validated status and the key selector validated failures.

⁹ the value is NOT AVAILABLE when the algorithm can't identify correctly the key position or the key is not inserted by the user.

4.4. Validation time for read values

To validate a new key selector value or a new key selector failure condition the signals should stay unchanged for the debounced time to filter very quick variations. In the algorithm there are set two parameters as described in the following table:

Parameter	Description	Value
KeyPositionTimer	Time to validate the new input status if one of the following positions is recognized <i>START</i> , <i>ACC</i> , <i>RUN</i> , <i>IGN_LK</i> without failure conditions	50 ms (adjustable)
FailureTimer	Time to validate the new input status if a failure condition is recognized	200 ms (adjustable)

Table 5: Validation timer

The timer should be set if the *keyselector_input_read* value is different from the already validated values contained in the *keyselector* structure and different from the value that is being validated in the *keyselector_input_stored* structure.

The timer should be reset if the *keyselector_input_read* value is equal to the already validated values contained in the *keyselector* structure but it is different from the value that is being validated in the *keyselector_input_stored* structure.

For instance, the timer is reset if the validated key position is C, A is the validating key position value and C is read newly.

4.5. Algorithm logic

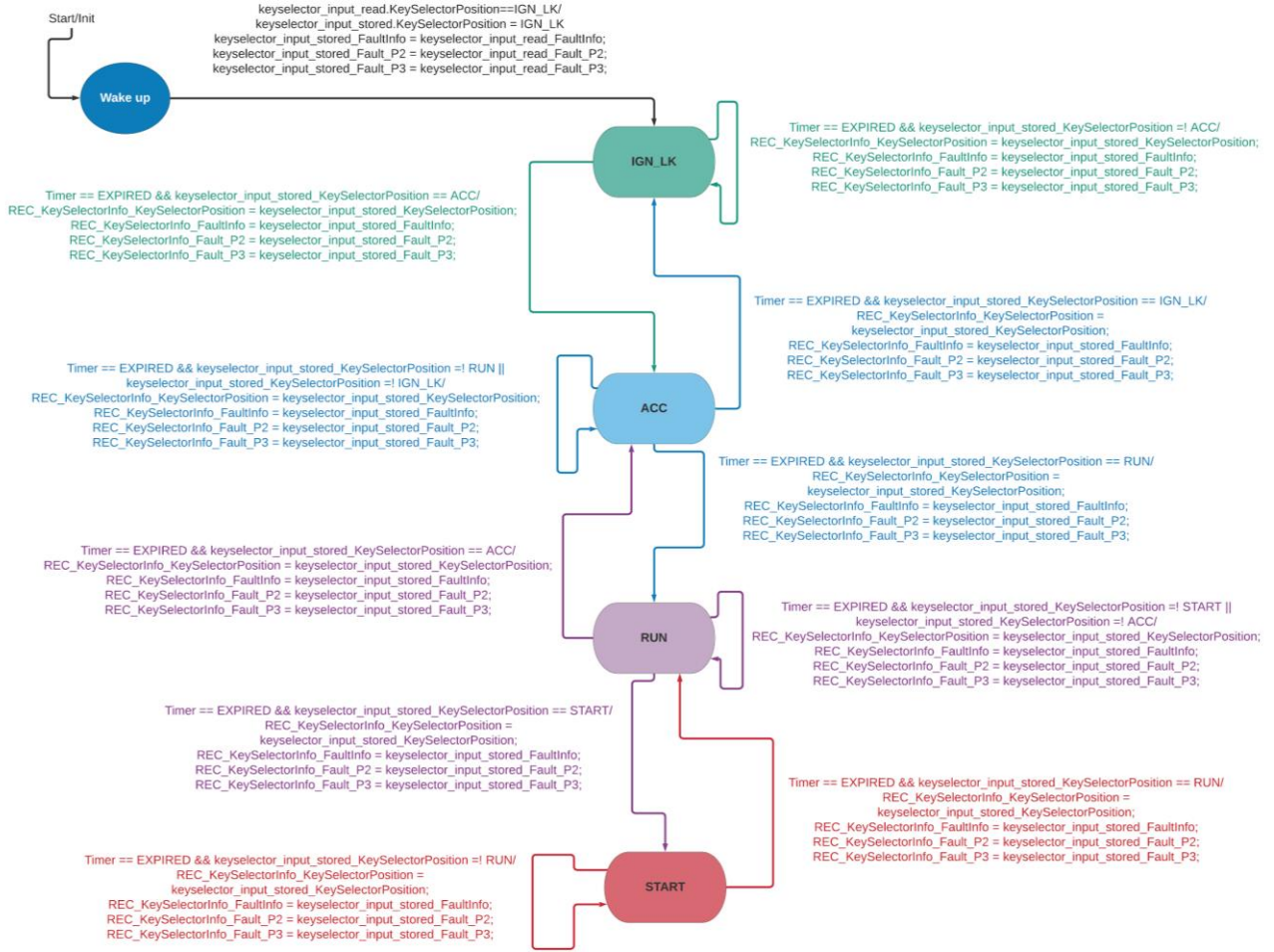


Figure 78: Algorithm main logic flowchart

The *keyselector_input_read* structure is initialized and the first key position assigned is Ignition lock with no faults.

After the input sampling period (5 ms), the *keyselector_input_read* structure assumes the read value that is stored in the *keyselector_input_stored*.

The variables in the *keyselector_input_stored* structure are validated when the timer elapses.

5. Algorithm model AUTOSAR compliant

5.1. Autosar overview

Autosar (AUTomotive Open System ARchitecture) is a software architecture for the development of standardized automotive systems. It pursues the objective of creating and establishing an open and standardized software architecture for automotive Electronic Control Units (ECUs). The functional software is described with *SoftWare Components* (SWCs) and each of these presents specific functionalities. The integration of different SWCs determines the final software for the possible application. The management of the components is made easier thanks to the standardization required by Autosar. The AUTOSAR architecture makes it possible to reuse the same components for different automotive applications which makes easier and faster the realization of new projects.



Figure 79: AUTOSAR logo

In the SoftWare Components, it is not necessary to insert information about the communicating component. Key information is the input ports, the output ports and the type of the exchanged data. SWCs communicate with the rest of the SoftWare and with themselves through the RTE (RunTime Environment) layer. The SWCs output goes to the *virtual functional bus* (VFB) which directs the information to the requesting part of the software. This is possible thanks to the standard language of Autosar.

Software architecture is based on three levels:

- Application layer (SWC)
- Presentation layer (RTE)
- Basic Software Modules (BSW)

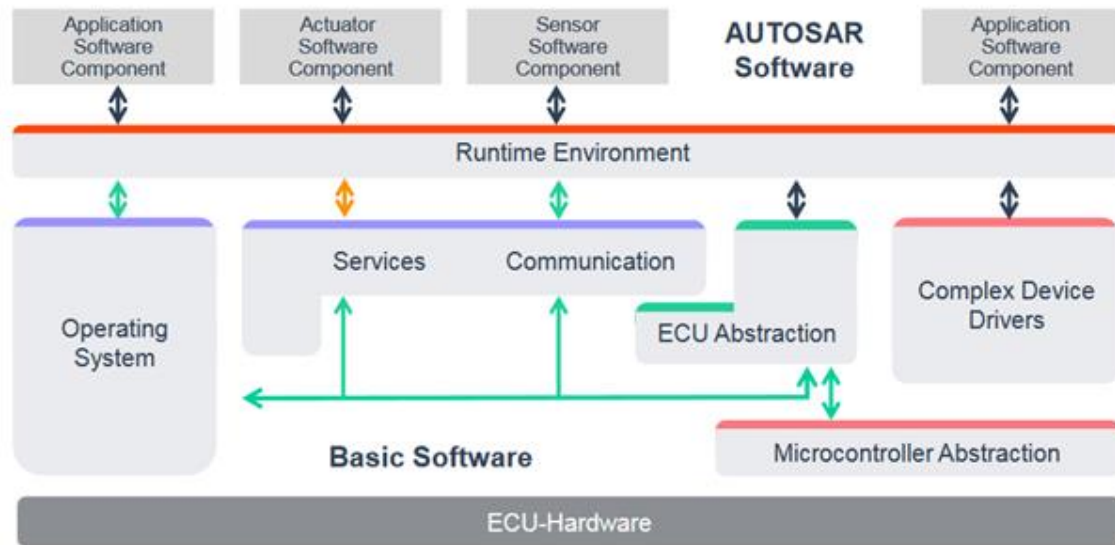


Figure 80: AUTOSAR architecture structure

The BSW is the standardized software layer which provides the infrastructure and services for execution of Software Components. BSW includes:

- System services
- I/O services
- Communication and Network management
- NVRAM management
- Operating System (OS)
- Microcontroller abstraction
- Complex Device Drivers, etc.

The BSW aggregates all the functionalities that are used by the applications. The *application layer* is made of interconnected Autosar SWCs and represents the SW application.

In particular, we have different types of SoftWare Component in the Application Layer:

- Application Software Component
- Sensor/Actuator Software Component

The Application SoftWare Component doesn't depend on the ECU and the location. Typically, it contains the main functionalities logics.

The Sensor/Actuator AUTOSAR Software Component is independent of the ECU on which it is mapped but is dependent on a specific sensor and/or actuator for which it is designed and therefore has strong relationship to local signals.

It has been decided to locate the Sensor/Actuator SW Components above the RTE for integration reasons (standardized interface implementation and interface description).

Because of their strong interaction with raw local signals they are not relocatable in most cases. Thus, because of performance issues, such components will need to run on the ECU to which the sensor/actuator is physically connected.

It is often necessary to have more than one SWC to make the overall SW application in real cases.

The SWC is composed of well-defined ports through which it can interact with other components. A port is used to provide or request information (data, operations calls, etc.). A complete Autosar SWC must have:

- A complete and formal Software Component Description which specifies how the infrastructure must be configured for the component.
- An implementation of the component.

Autosar supports three main ports interfaces:

- *Sender-Receiver*: a sender sends information to one or more receivers or one receiver gets information from one or more senders;
- *Client-Server*: several clients request a set of operations (Get, Set, etc.) to a server which provides them the requested service;
- *Calibration*: static communication pattern that allows modules to access static calibration parameters.

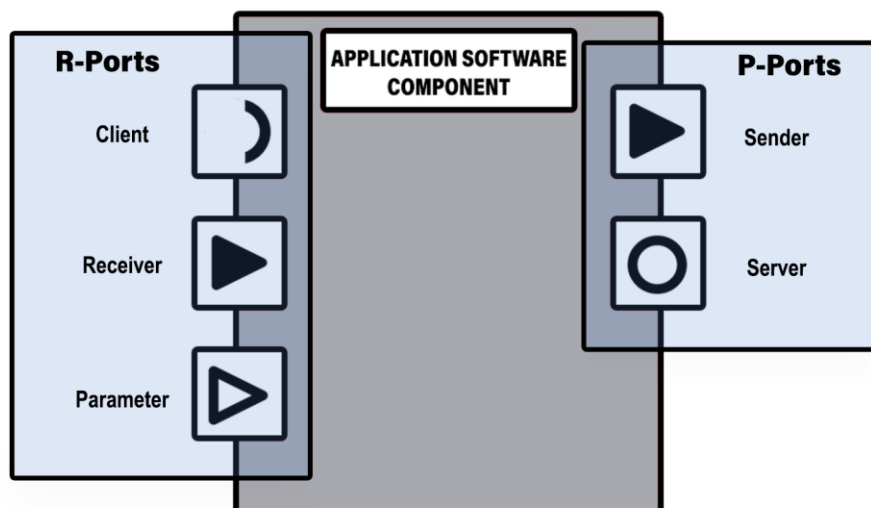


Figure 81: AUTOSAR standard port-icons of ports interfaces

For every port interface:

- *PPort* (ProvidePort): refers to ports that send information to others;
- *RPort* (ReceivePort): refers to ports that get information from others.

The components are developed against the *Virtual Functional Bus* without a direct dependency on ECUs and communication busses. *Virtual Functional Bus* (VFB) is the most abstract level where components are described as data types and interfaces, ports and connections between them. Autosar SWCs are implemented independently from the underlying hardware to achieve the relocatability. This entails that components must not call directly the Operating System or the communication Hardware, therefore components can be deployed (integration process) to ECUs very late in the development process.

The implementation of the Autosar VFB is the RTE and it acts as a system level communication center for information exchange. The RTE manages the connections between the basic software and the application layer (it is visible in Figure 80 where the indicators represent the connections). The RTE also manages the communication between the different application software components which can't communicate directly with each other.

Runnable Entities (one or more) are the schedulable parts of an Atomic Software Component which can be executed independently from the others: they are sequences of instructions that can be started by the RTE and mapped to different tasks.

It is also possible to configure specific parameters with the calibration ports mechanism in order to make the system versatile.

5.2. Benefits and disadvantages using AUTOSAR

Main Autosar benefits are:

- Use of common interfaces
- Design flexibility
- Integration of Application SoftWare Component vs Basic SoftWare level easier
- Reduction of SW development costs
- Interoperability
- SWC reuse for different HardWare platforms

Autosar standard is used by more than 180 organizations worldwide in the automotive domain who “speak the same standardized language”.

On the other hand, Autosar complexity causes a cost to teach people how to use it. The suggestion of most users is to make Autosar more tool oriented to reduce

the general complexity. Using AUTOSAR there is an increment of system resources required (for example the memory occupation).

5.3. SWC development

With reference to AUTOSAR architecture structure (the Figure 80) the current studied system can be located at *Application level* categorized as a *sensor component* due to its dependency on specific voltage signals. We can apply the architectural model usually used at the application level (Application SoftWare Component prototype).

In AUTOSAR each SWC is described with a SWC template. In this study, the SWC is defined in all its aspects using a SWC Configuration (SWCC) template, in Excel format, that contains:

- General characteristics: name, manufacturer, etc;
- Input/Output properties: SendPorts, ReceivePorts, Interfaces and details (such as DataType, InvalidValue, InitialValue);
- Runnable entities with trigger events, port access, etc;
- Required HW resources: ECU timer, scheduling and memory;

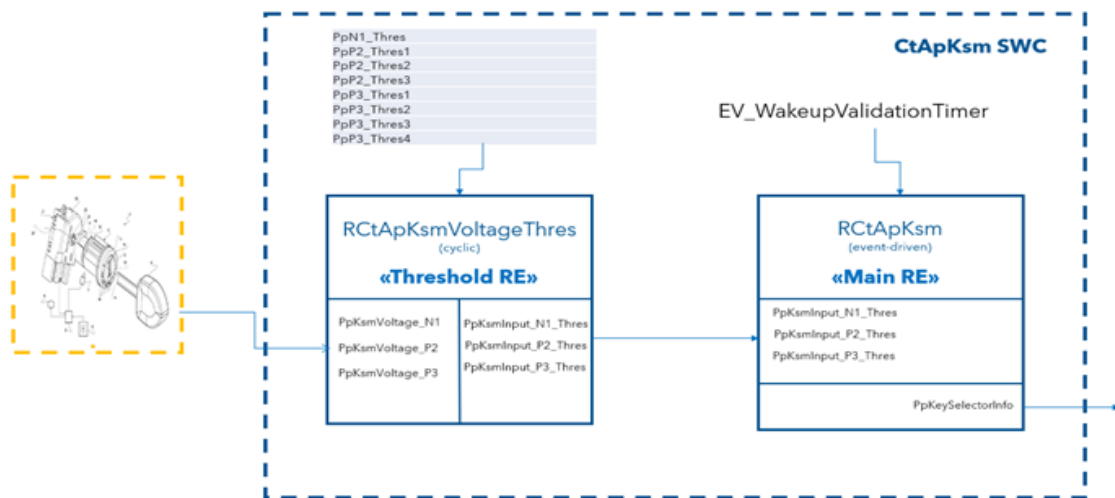


Figure 82: Software architecture scheme

Referring to the Figure 82 the current study is developed on a unique SoftWare Component called *CtApKsm*. It is composed of two *runnables*:

- *Threshold RE (RCTApKsmVoltageThres)* deals with the discretization of the signals that come from the Key selector pins.
- *Main RE (RCTApKsm)* processes quantized signals coming from the threshold runnable to provide the key position or the situation *NOT_AVAILABLE* and the failure conditions.

The *Threshold runnable* is *cyclic* and it means that the code is executed at every established period of time defined in the SWCC.

The main runnable is *event-driven*, so the code is executed only when one of the well-defined events in the SWCC occurs.

In the SWCC it is described every port, every interface and all the AUTOSAR characteristics of the elements (DataType, port type, timer set operation, etc.). The arxml language is the formal language used to describe the SWC. In this case the SWCC Excel format is used to make the reading easier and starting from its contents arxml format and subsequent API (Application Program Interface) are then automatically generated to interact with the RTE level for the integration in the rest of the system (ECU).

The inputs of the thresholds runnable are the analog signals evaluated on the pins and the outputs are the quantized values based on thresholds.

The inputs of the main runnable are the output of the previous one and its output is a record that contains the key position, the general fault information, the fault information on pin 2 and the fault information on pin 3.

5.4. SWC toolchain

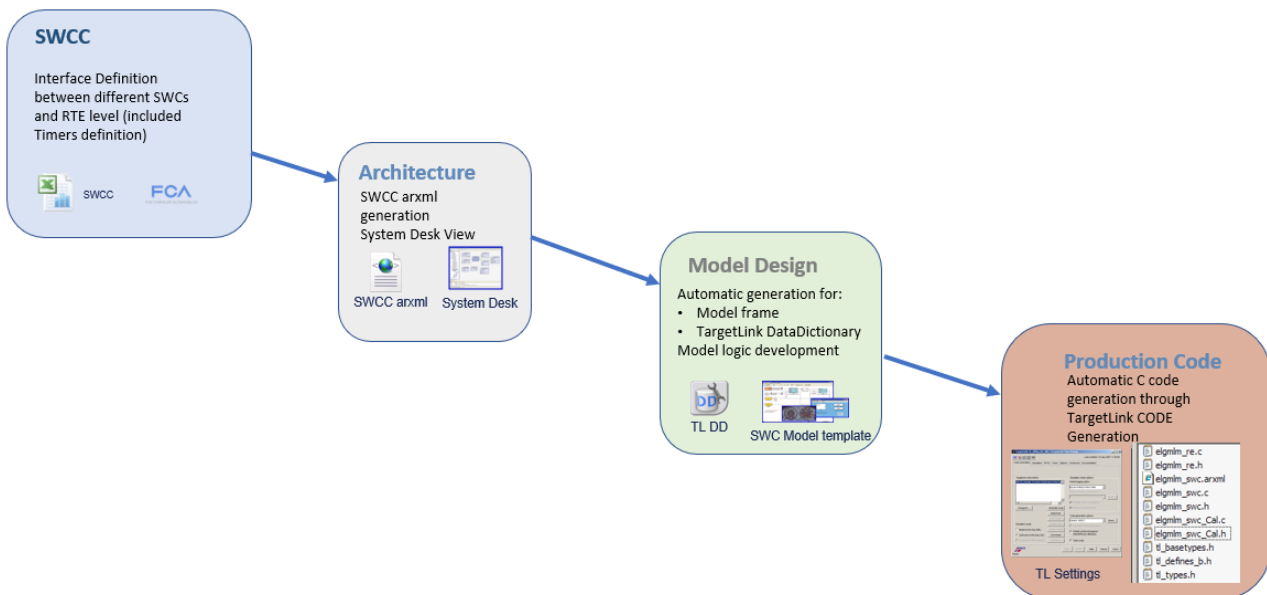


Figure 83: SWC toolchain

The tools used for the development of the SWC are:

- Excel (SWCC);
- System Desk (Architecture);
- Matlab, Simulink and Stateflow (Model Design);
- Targetlink (Production Code).

Excel is a spreadsheet with functions for calculation, graphing tools, pivot tables and a macro programming language. In this case it is used to describe the SWC Configuration (SWCC) aspects into a document.

SystemDesk is a system architecture tool that provides sophisticated and extensive support for modeling AUTOSAR architectures and systems for application software.

Matlab, *Simulink* and *Stateflow* are used to design and test the model¹⁰. *Simulink* is a MATLAB-based graphical programming environment for modeling, simulating and analysing multi-domain dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. The *Stateflow* uses a graphical language with state transition diagrams, flow charts, state transition tables and truth tables. It is used to describe the logic of MATLAB algorithms and *Simulink* models after the receiving of input signals, events or time-based conditions. The model and sequential decision logic can be implemented as a block within a *Simulink* model.

TargetLink is a software to generate automatically production code based on *Simulink/Stateflow* graphical development environment. It is used to generate the C-code of the functional logic developed.

TargetLink converts existing *Simulink* models to work on.

The C code can be generated with ANSI C code, optimized fixed- or floating-point code for AUTOSAR platforms. The code configuration options are versatile to ensure the adaptability with all the processors constraints. There is a central data container (called the *Data Dictionary*) that is used for the management of the information related to the code generation.

The Figure 83 explains the process flow of the SWC development. Starting from the SWCC filled with the SWC characteristics, the SystemDesk project and the related SWC arxml files are created through a proprietary macro. Then the macro allows in an automatic way the Simulink model frame generation and the data dictionary creation used by Targetlink. The functional logic is developed inside the model frame and finally the automatic code generation is performed.

¹⁰ For further information consult the paragraph "Model description"

5.5. Model description

The algorithm is implemented by Model Based Design (in particular *Simulink* and *Stateflow*) in order to carry out the simulation and to generate the C-code.

5.6. Threshold runnable

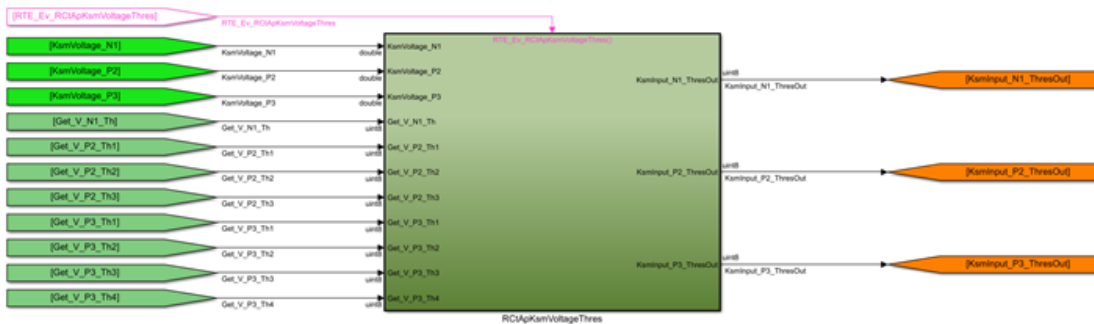


Figure 84: General view of the thresholds runnable

On the left side of the Figure 84 there are the inputs of the thresholds runnable:

- analog voltage values measured on the pins;
- threshold values used to quantize the analog voltage signals.

Threshold values are managed as inputs because they are defined as tunable parameters in order to make the model suitable for both the key selector configuration (with or without Accessory position).

The outputs of this runnable are the quantized voltage values of each pin.



Figure 85: General view of the Input_P3 logic

Figure 85 is a general view of pin 3 quantization logic. A similar one is created for the other two pins but with a different number of threshold values.

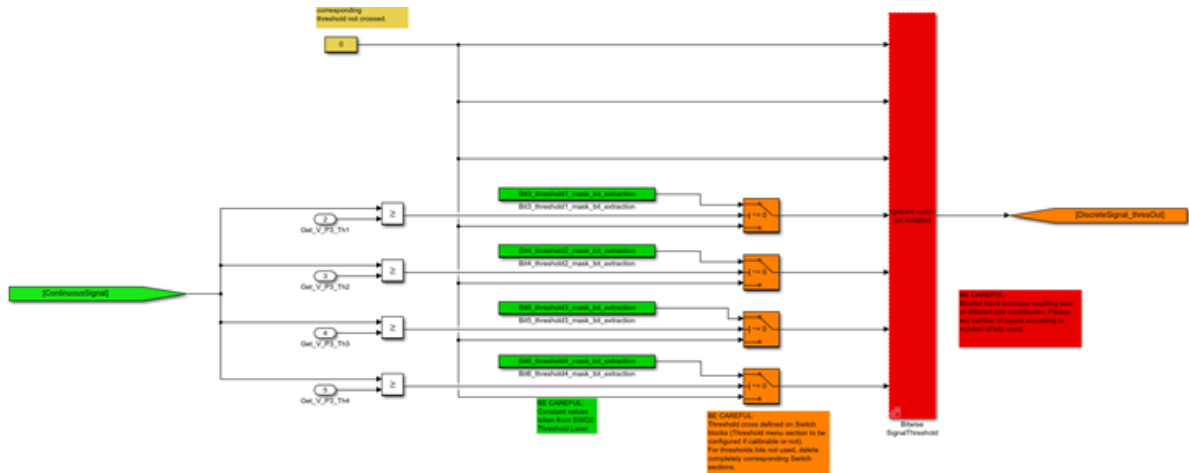


Figure 86: Logic pin 3 quantization

Figure 86 is the logic of the input on pin 3 to quantize the analog voltage signals. This logic is similar to the logic of the other two pins. The continuous signals are managed with a standardized bit mask based on unsigned integer numbers stored with 8 bits. The first 3 bits are used for service information (doesn't used in this context) while the others obtain information from the comparison with the threshold values (one or more).

For the input on the pin 1:

$$\begin{aligned} \text{Continuous value} \geq \text{Threshold value} &\rightarrow \text{quantized value} = 8 \\ \text{else} &\rightarrow \text{quantized value} = 0 \end{aligned}$$

For the input on the pin 2:

$$\begin{aligned} \text{Continuous value} \geq \text{Threshold value 3} &\rightarrow \text{quantized value} = 56 \\ \text{Continuous value} \geq \text{Threshold value 2} &\rightarrow \text{quantized value} = 24 \\ \text{Continuous value} \geq \text{Threshold value 1} &\rightarrow \text{quantized value} = 8 \\ \text{else} &\rightarrow \text{quantized value} = 0 \end{aligned}$$

For the input on the pin 3:

$$\begin{aligned} \text{Continuous value} \geq \text{Threshold value 4} &\rightarrow \text{quantized value} = 120 \\ \text{Continuous value} \geq \text{Threshold value 3} &\rightarrow \text{quantized value} = 56 \\ \text{Continuous value} \geq \text{Threshold value 2} &\rightarrow \text{quantized value} = 24 \end{aligned}$$

Continuous value \geq Threshold value 1 \rightarrow quantized value = 8
else \rightarrow quantized value = 0

For instance, in the case of figure 34 if the voltage is equal to 0 V then the bit mask would be Bin 00000000 (decimal value 0), if the voltage major than the threshold value 2 then the bit mask would be Bin 00011000 (decimal value 24).

5.6.1. Main runnable

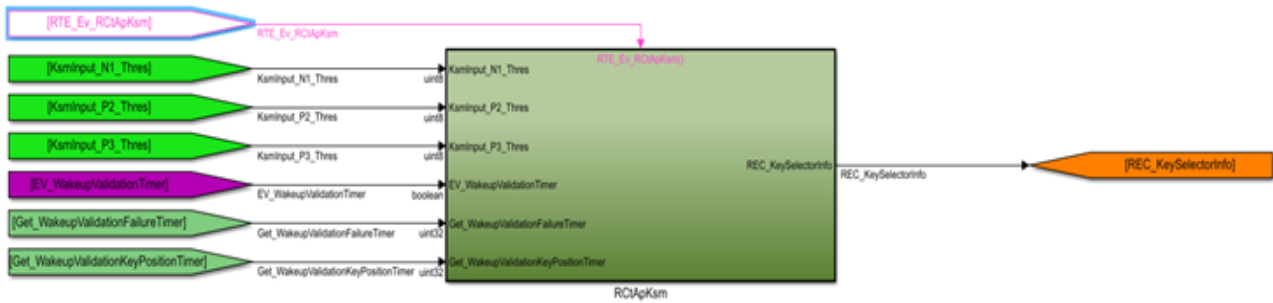


Figure 87: General view of the main runnable

On the left side of the Figure 87 there are the inputs of the main runnable:

- quantized values of the voltage which comes from the thresholds runnable;
- expiration event of the timer for the key position validation¹¹;
- parameters to define the validation timer duration based on the debounce logic.

The output of this runnable is a *record* (structure) which contains 4 components:

- key position;
- general failure situation;
- electric failure on pin 2;
- electric failure on pin 3¹².

¹¹ For further information consult the paragraph "Validation time for read values"

¹² For further information consult the paragraph "Algorithm structures and variables"

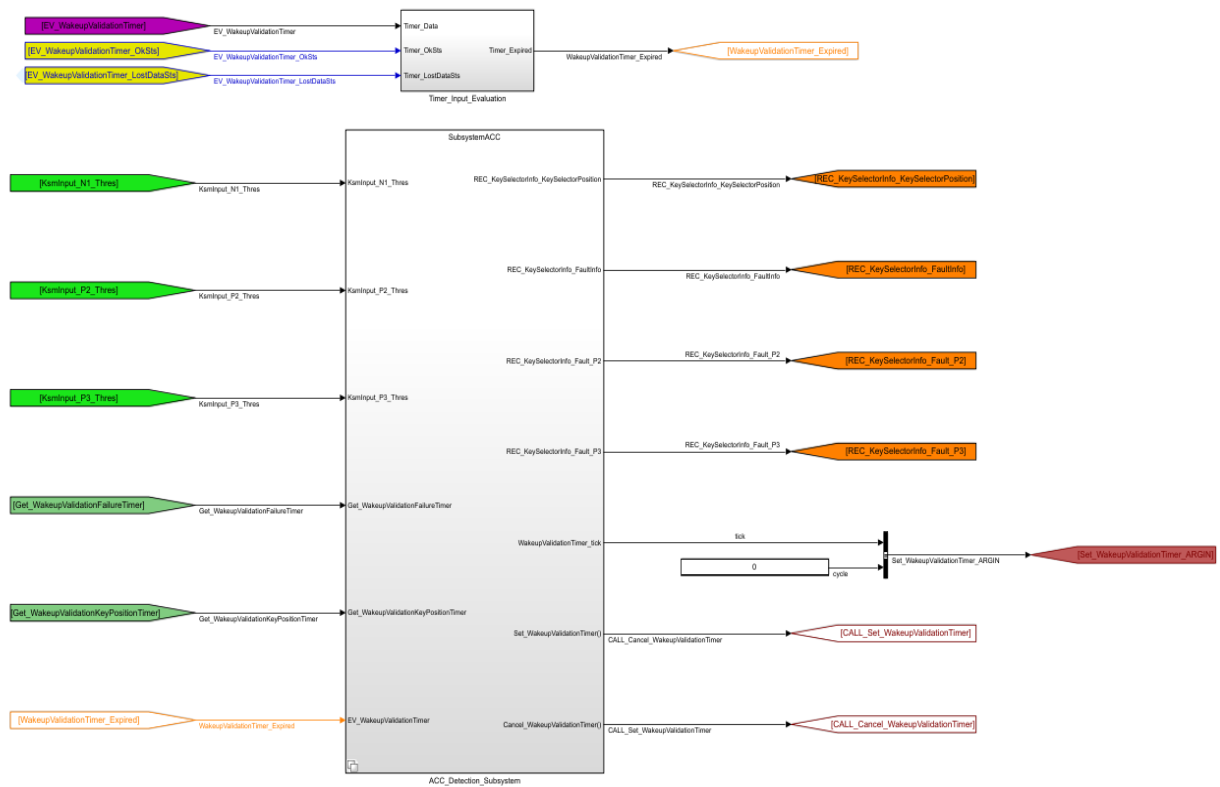


Figure 88: General view of the detection block

Two different *variant subsystems* are developed in the model in order to make it suitable for both the key selector configurations:

- with *Accessory*;
- without *Accessory*.

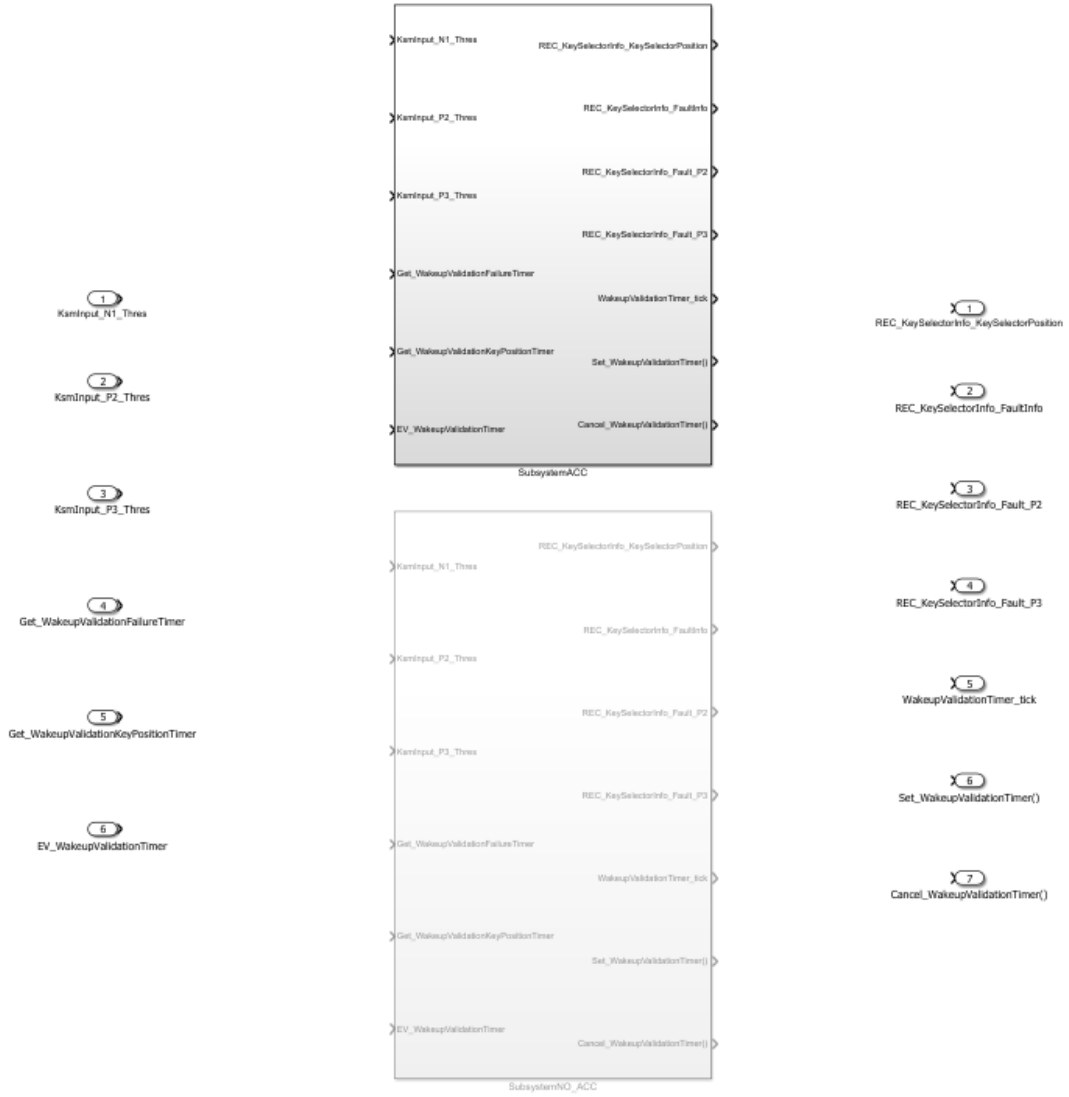


Figure 89: ACC and NO_ACC variant subsystem

One or the other subsystem is activated depending on the *Accessory* position presence related to the value of the variable *RCtApKsm_ACC_Presence*.

$$RCtApKsm_ACC_Presence == 0 \Rightarrow NO\ ACCESSORY$$

$$RCtApKsm_ACC_Presence == 1 \Rightarrow ACCESSORY$$

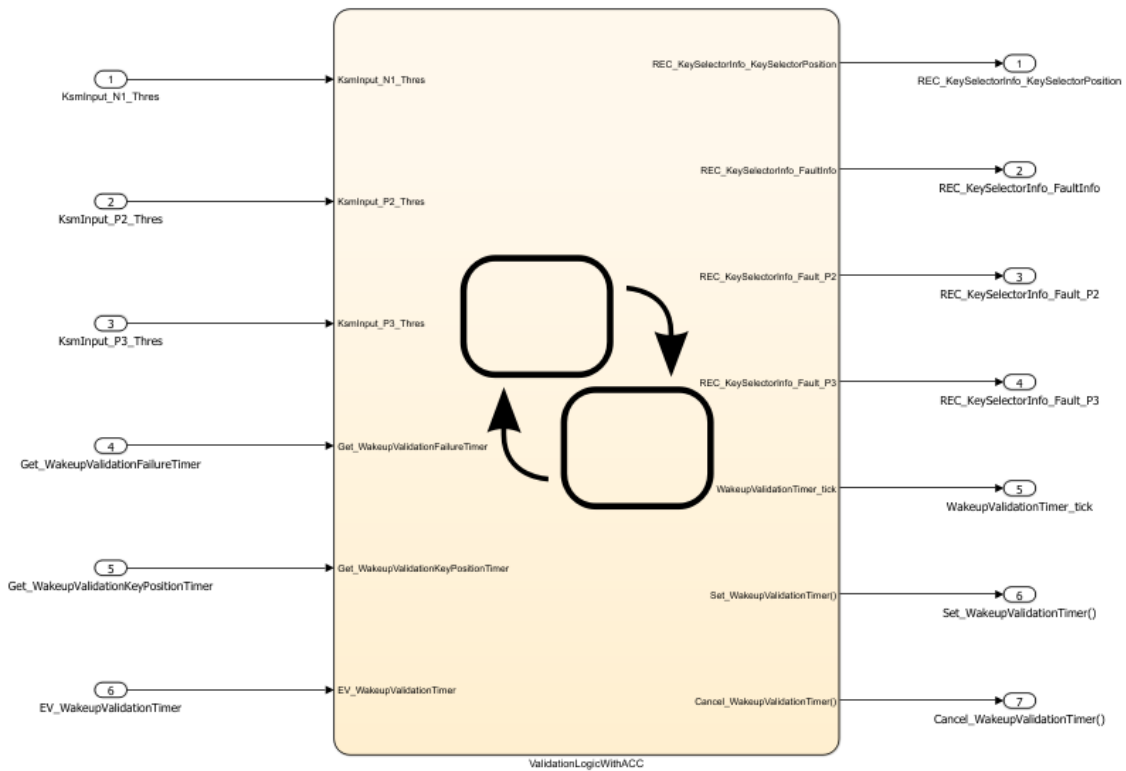


Figure 90: Acquisition logic stateflow

Stateflow is used here for the acquisition and debouncing logic of the voltage signals¹³.

5.6.2. Model validation

For testing the model logic, a **dashboard** is created in Simulink. In this study, it is used to:

- compare analog and quantized voltage values;
- show the outputs of the acquisition logic;
- adjust some variables for the selected key selector configuration.

¹³ For further information about the developed logic consult the paragraph “Algorithm logic”

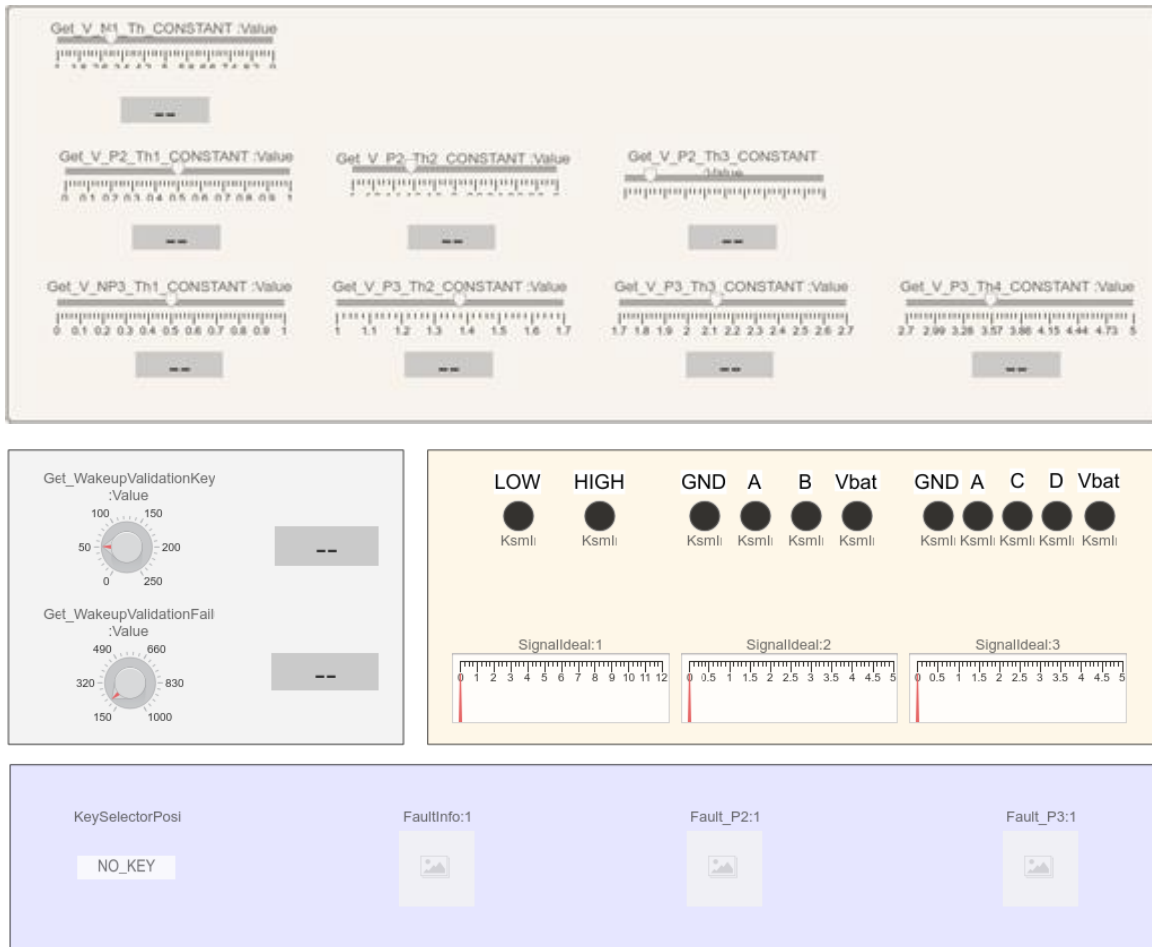


Figure 91: Dashboard of the key selector model

The **sliders** on the top of the dashboard are used to set the threshold values.

The **knobs** in the grey box are referred to the validation timers.

The **linear gauges** in the yellow box are used to display the values of the analog signals.

The **lamps** are used to show the quantized signal values. A lamp lights up when the corresponding threshold is exceeded.

The last box contains graphical indicators related to the final output of the model (the structure *keyselector*).

Other 3 knobs are used to change the analog voltage values on each pin in order to test the *Simulink* model.



Figure 92: Knobs used to test the Simulink model

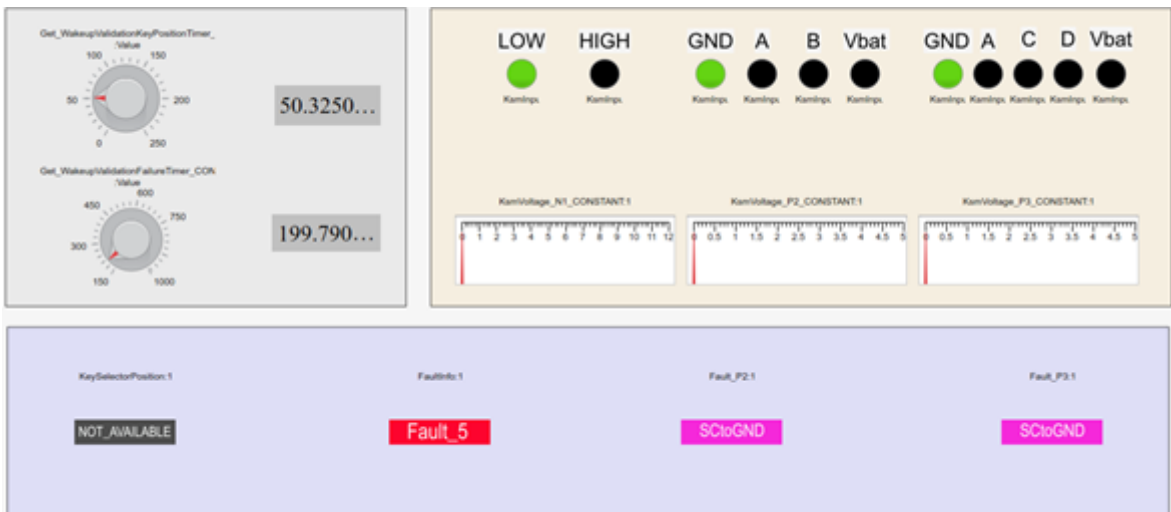


Figure 93: Result of a model test

For instance, if the knobs are set as in the Figure 92 the analog values are all 0 V. The detected key position is NOT_AVAILABLE and the general fault is *Fault_5* with the *SCtoGND* for pin 2 and 3.

The following figures are print screens of the simulation that use the PSpice electrical signal simulation as input for the implemented model (nominal voltage signals in the configuration without Accessory).

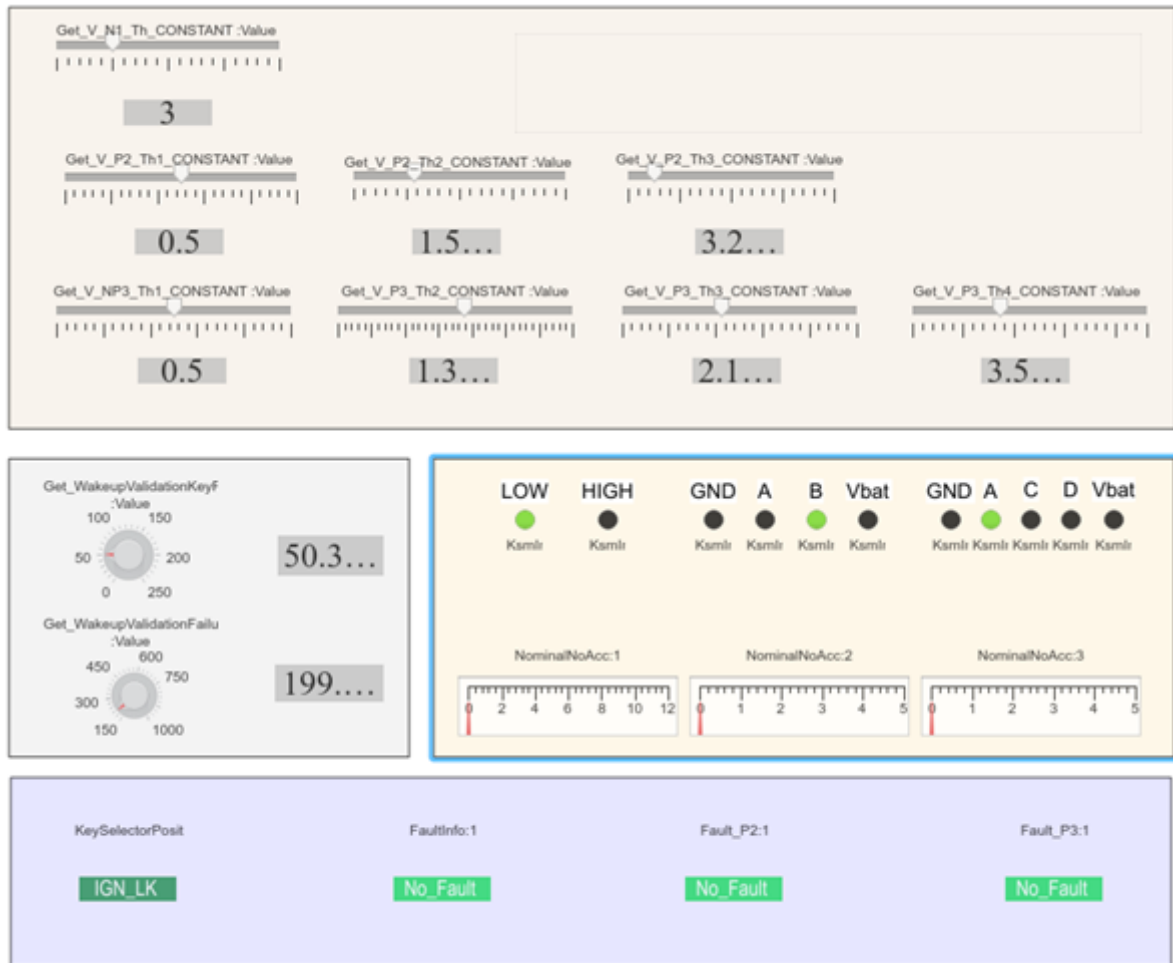


Figure 94: Print screen simulation (IGN_LK status)

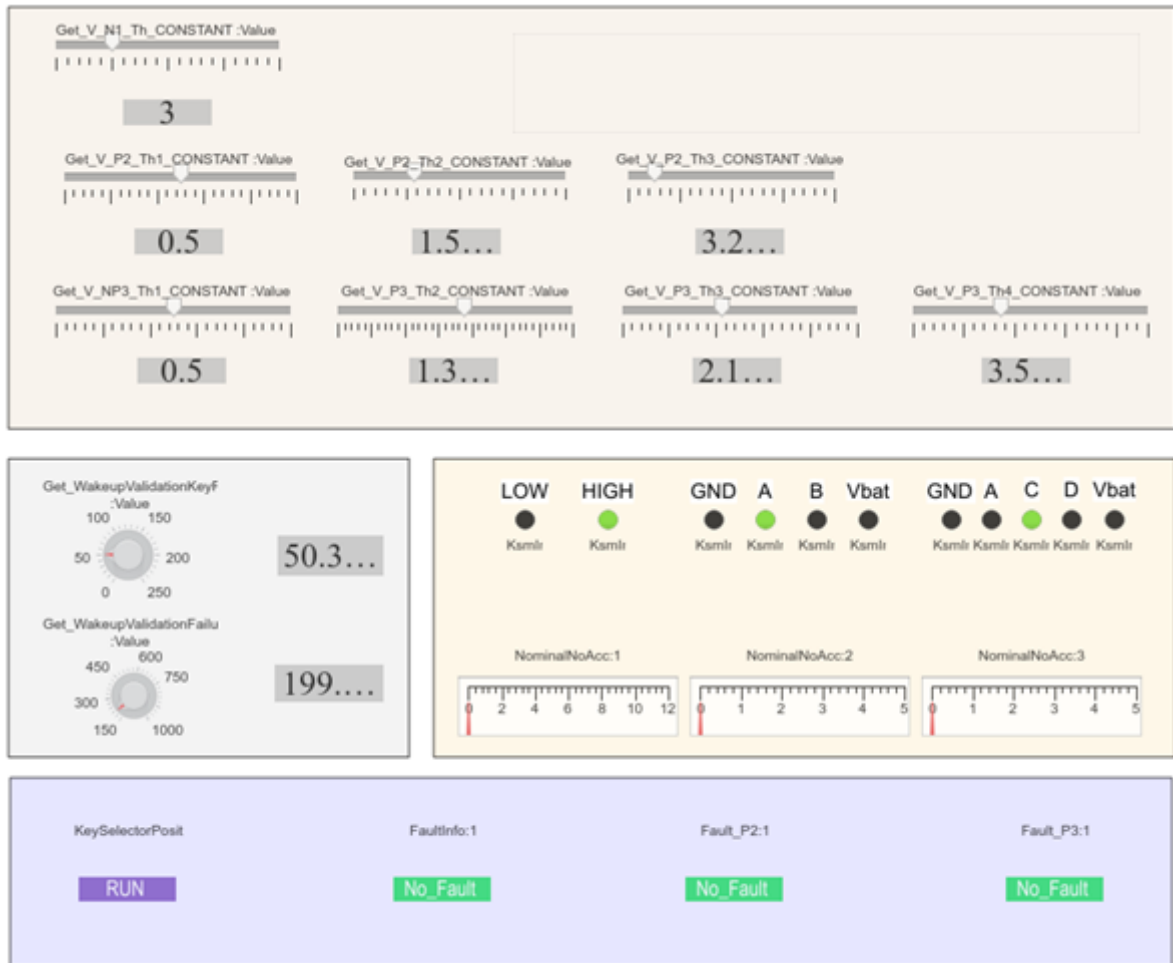


Figure 95: Print screen simulation (RUN status)

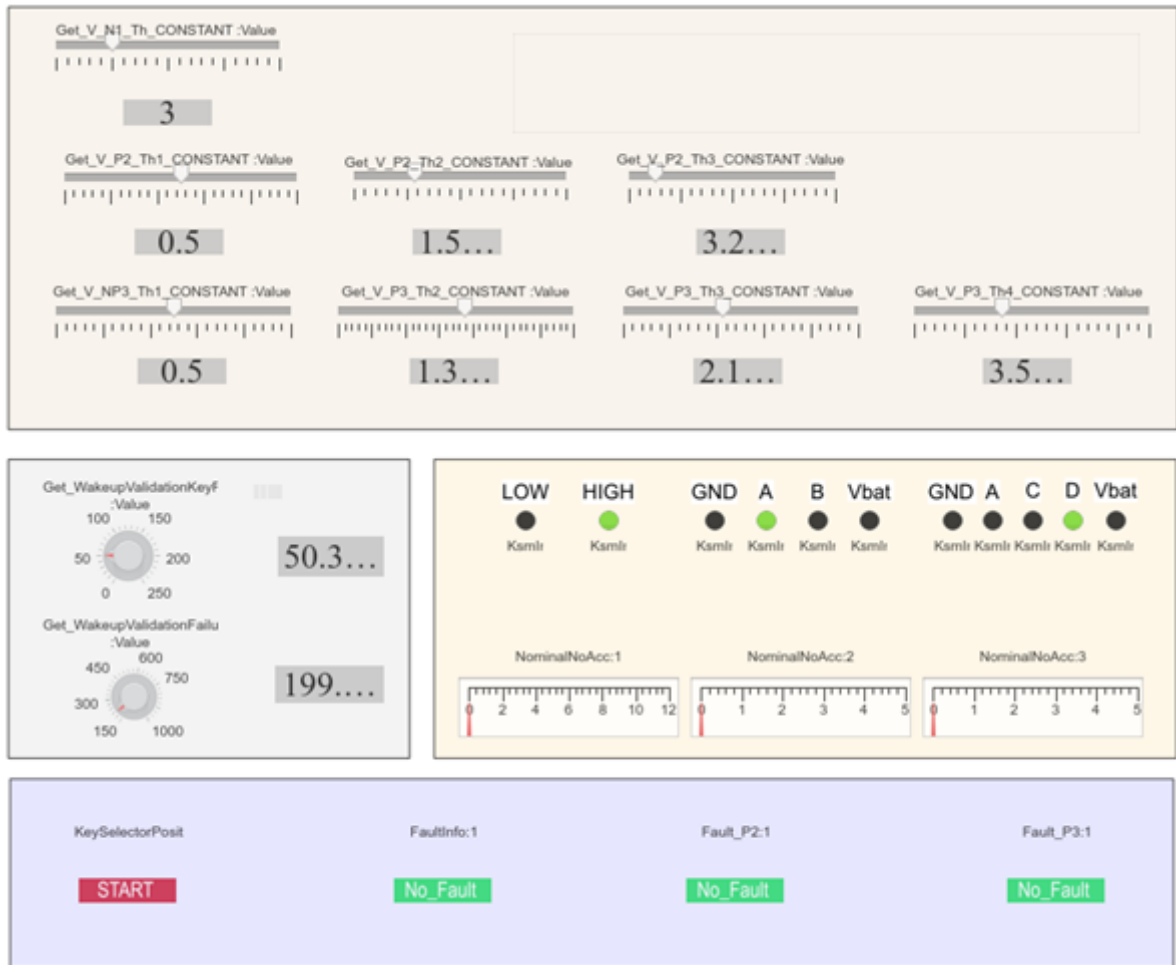


Figure 96: Print screen simulation (START status)

6. Simulation

Here below are reported some of the signals used to test the *Matlab/Simulink* model; the signals are the result of the PSpice simulation. The data (voltage, time) are exported in an Excel file and a *Signal Builder* makes them as input for the model.

Using the nominal signals in the configuration without Accessory, the result is in the following figures.

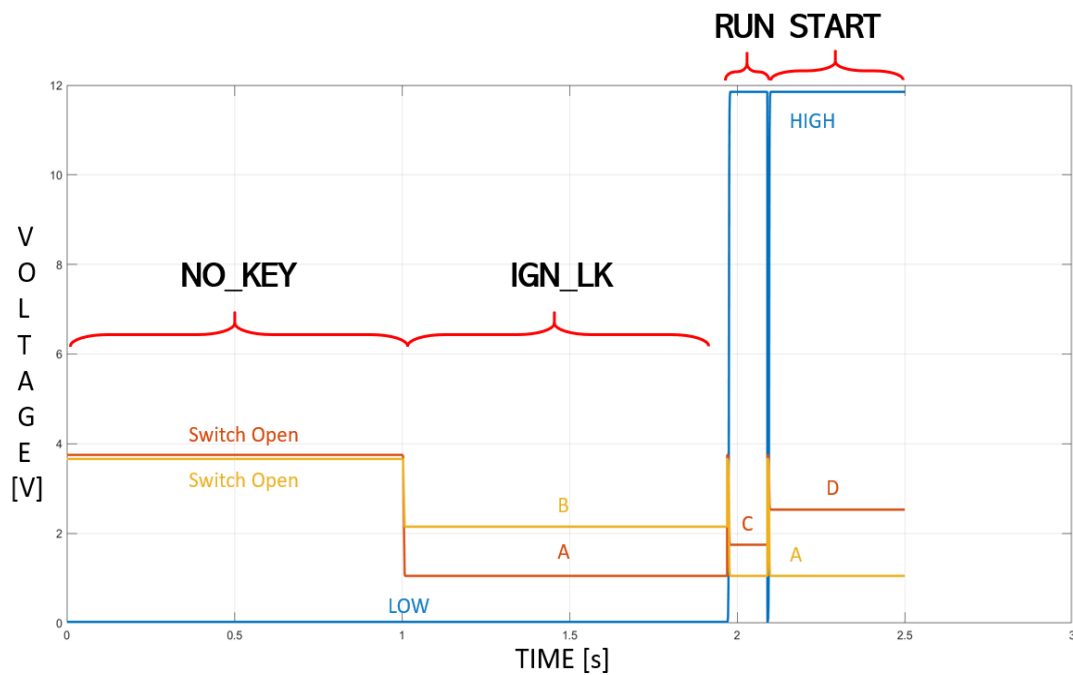


Figure 97: PSpice Nominal Pins voltage signals (configuration without Accessory)

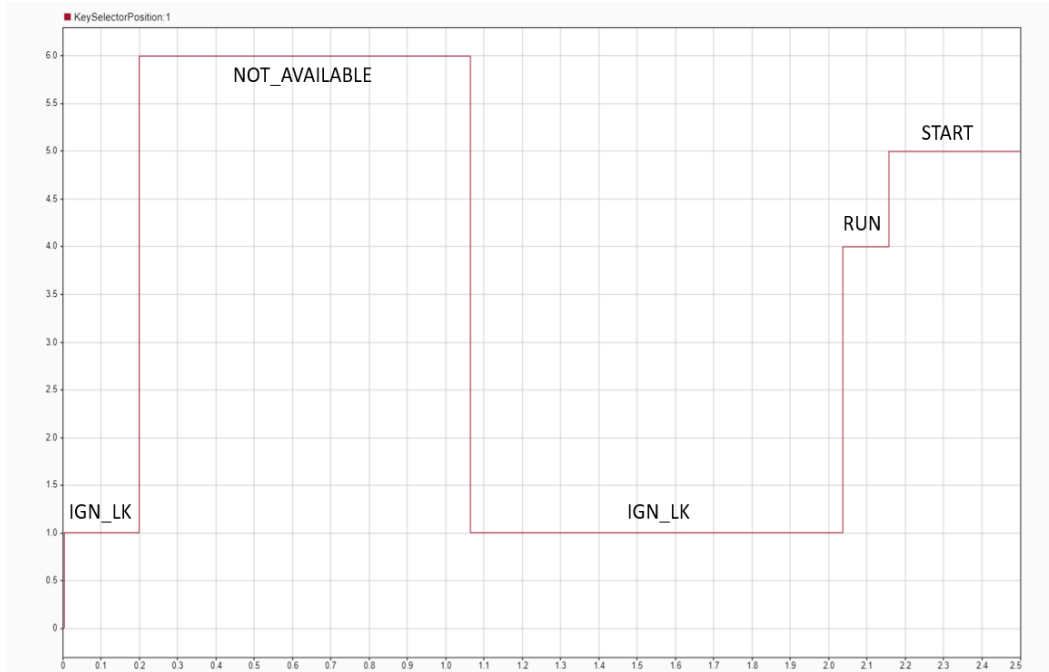


Figure 98: Key position detected with nominal pins voltage signals (configuration without Accessory)

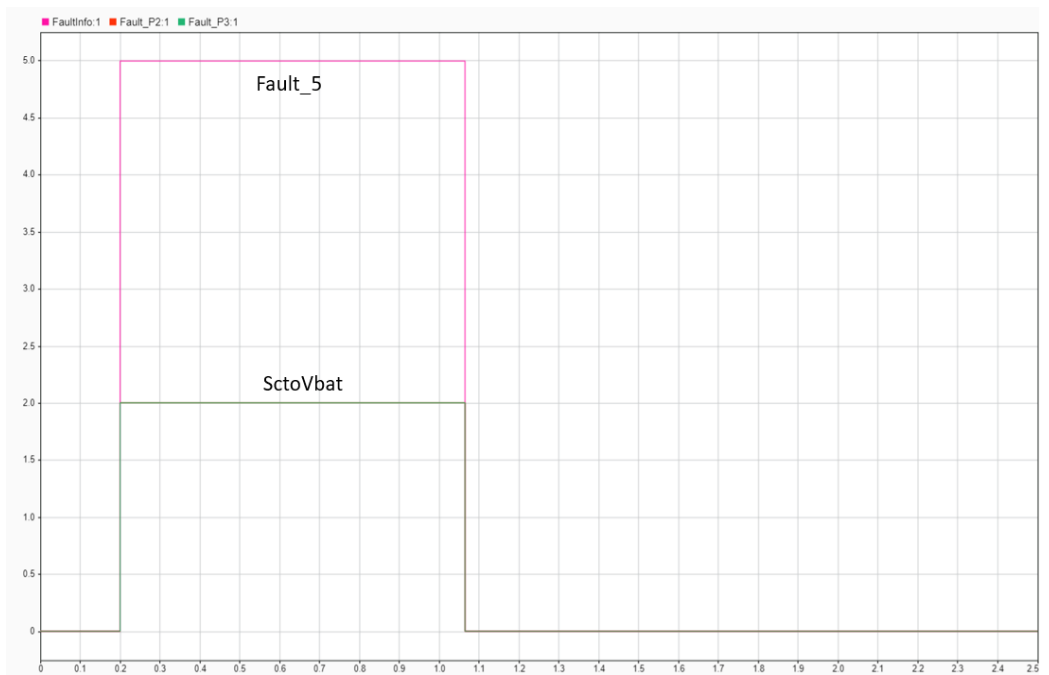


Figure 99: Faults detected with nominal pins voltage signals (configuration without Accessory)

As per hypothesis explained in the paragraph “Input signal quantization”, the key not inserted by the user is recognized as a SctoVbat/OC condition for the pins P2 and P3; so the algorithm sets the Fault_5 and the key position is set to NOT_AVAILABLE.

Using the nominal signals in the configuration with Accessory, the result is the following figure.

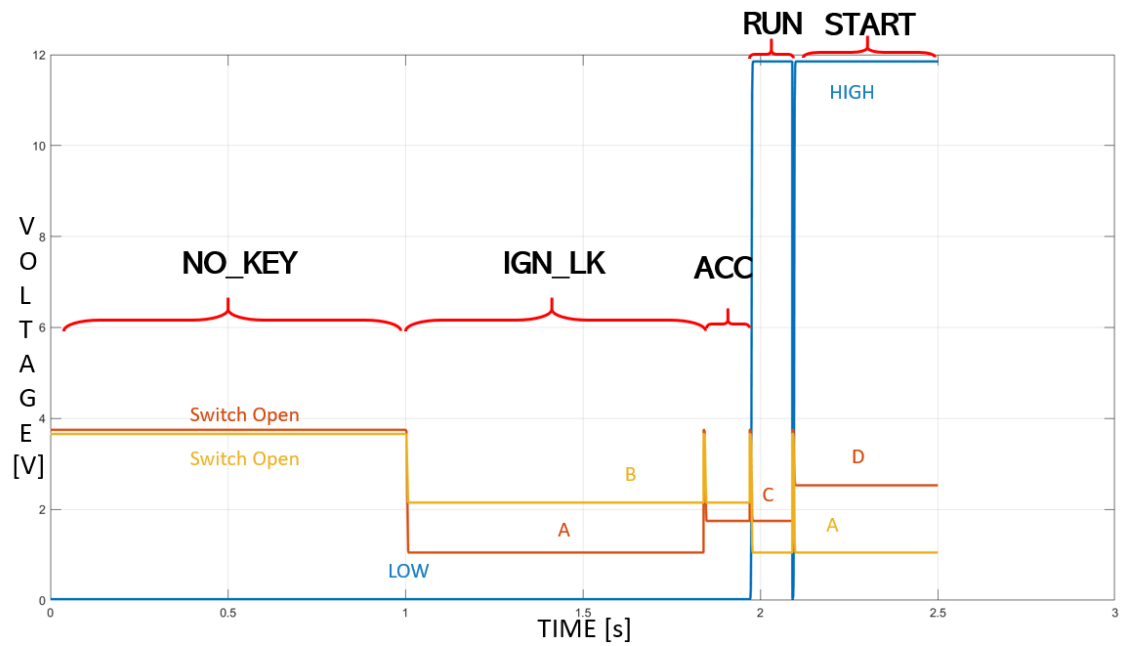


Figure 100: PSpice Nominal Pins voltage signals (configuration with Accessory)

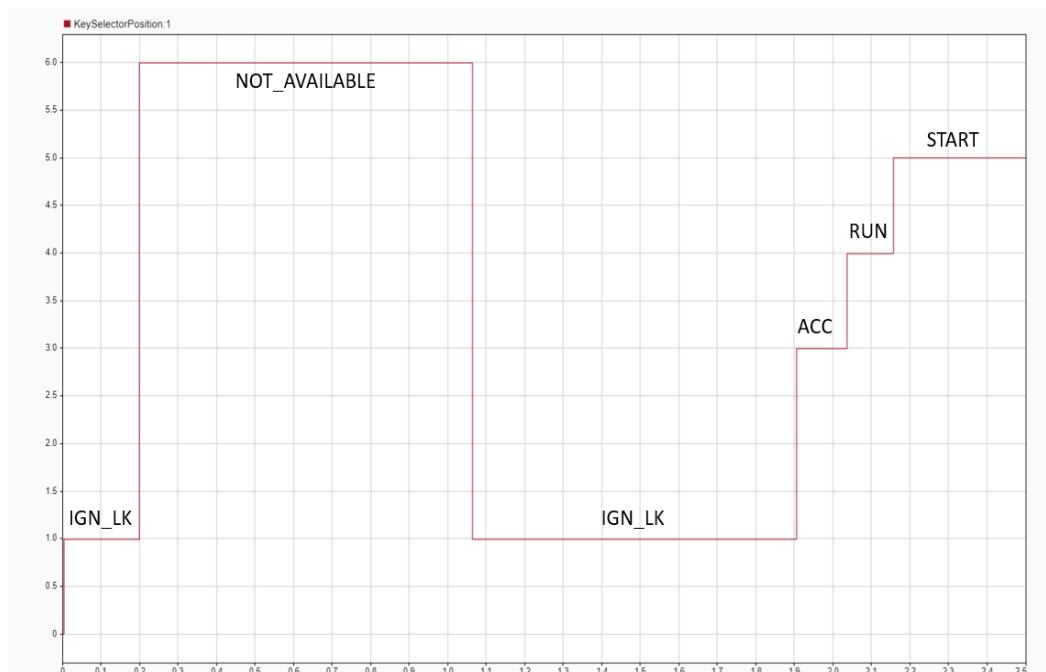


Figure 101: Key position detected with nominal pins voltage signals (configuration with Accessory)

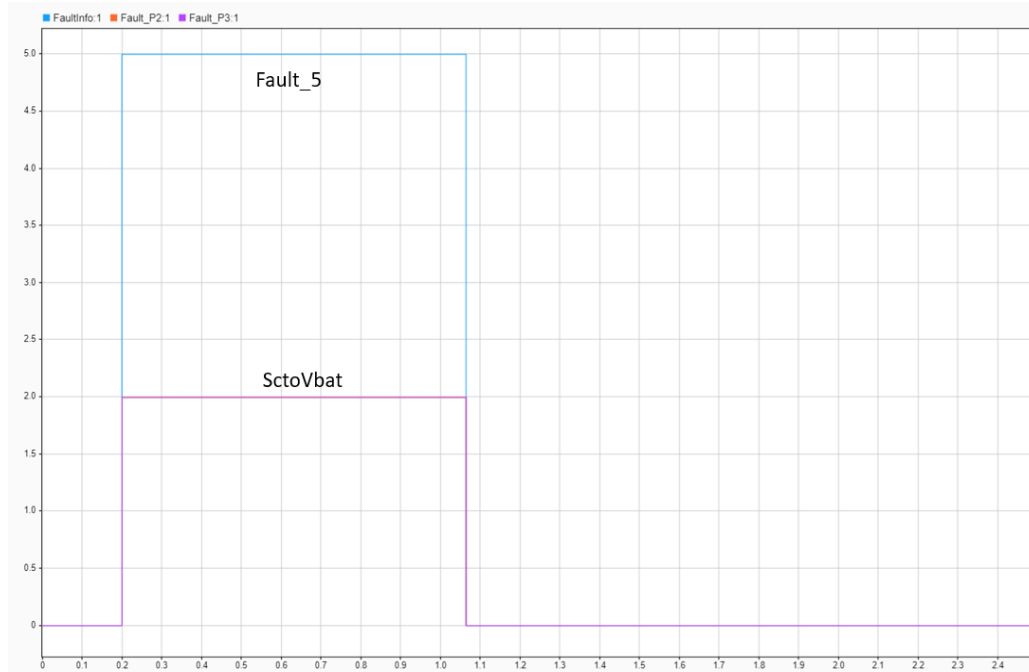


Figure 102: Faults detected with nominal pins voltage signals (configuration with Accessory)

Using the Montecarlo signals (Min) in the configuration without Accessory, the result is the following figure.

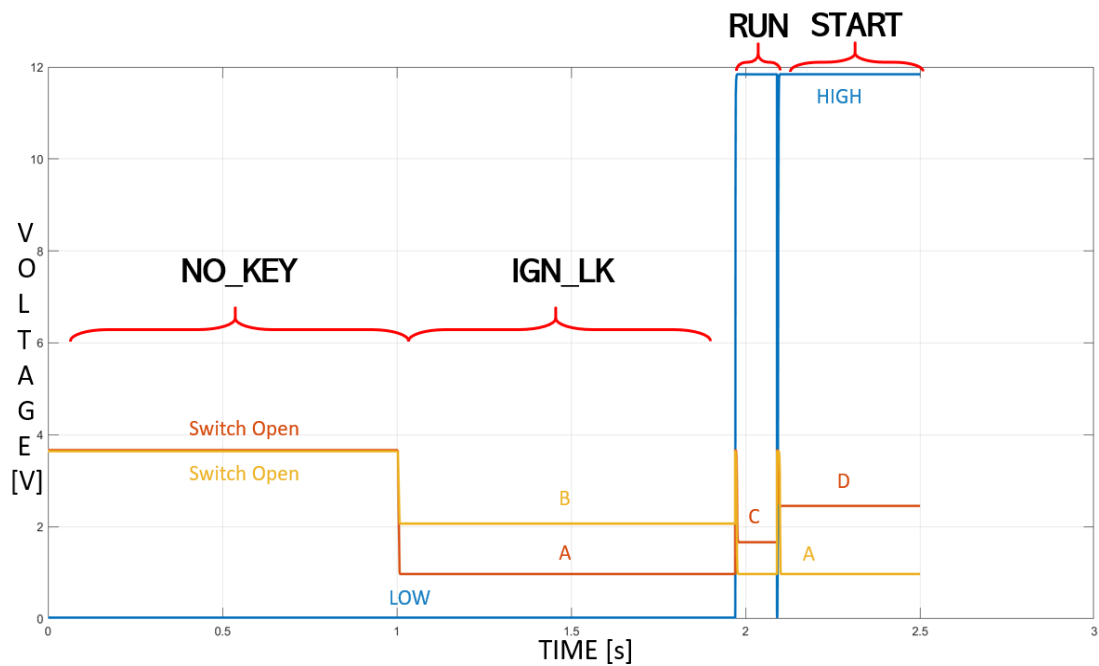


Figure 103: PSpice Montecarlo Pins voltage signals Min (configuration without Accessory)

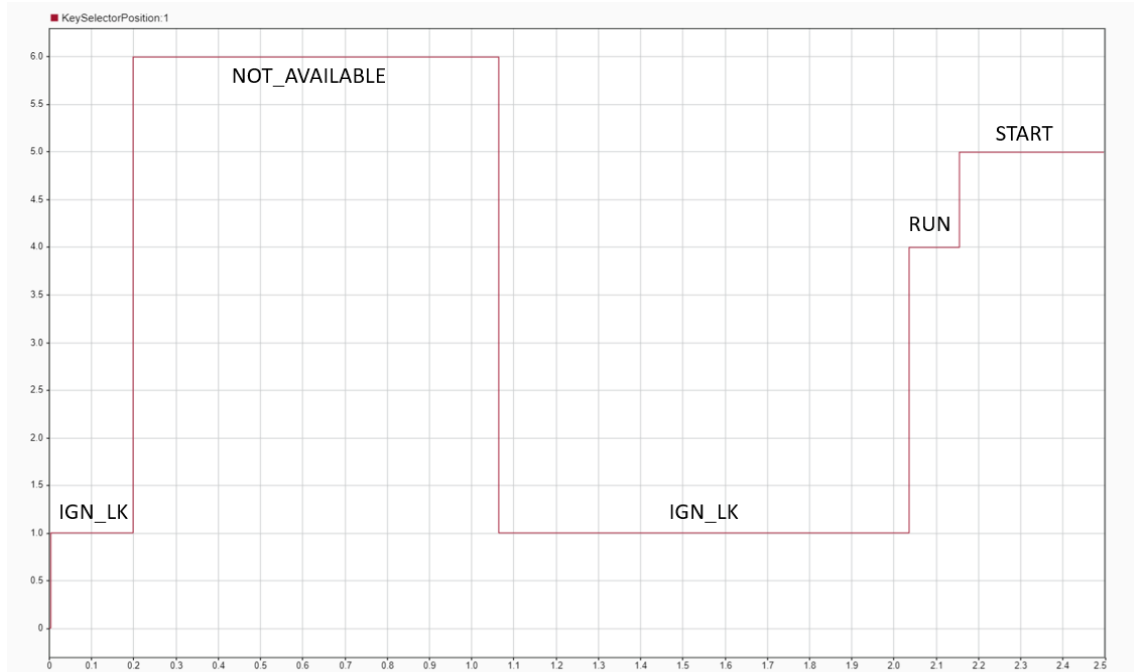


Figure 104: Key position detected with Montecarlo pins voltage signals (configuration without Accessory)

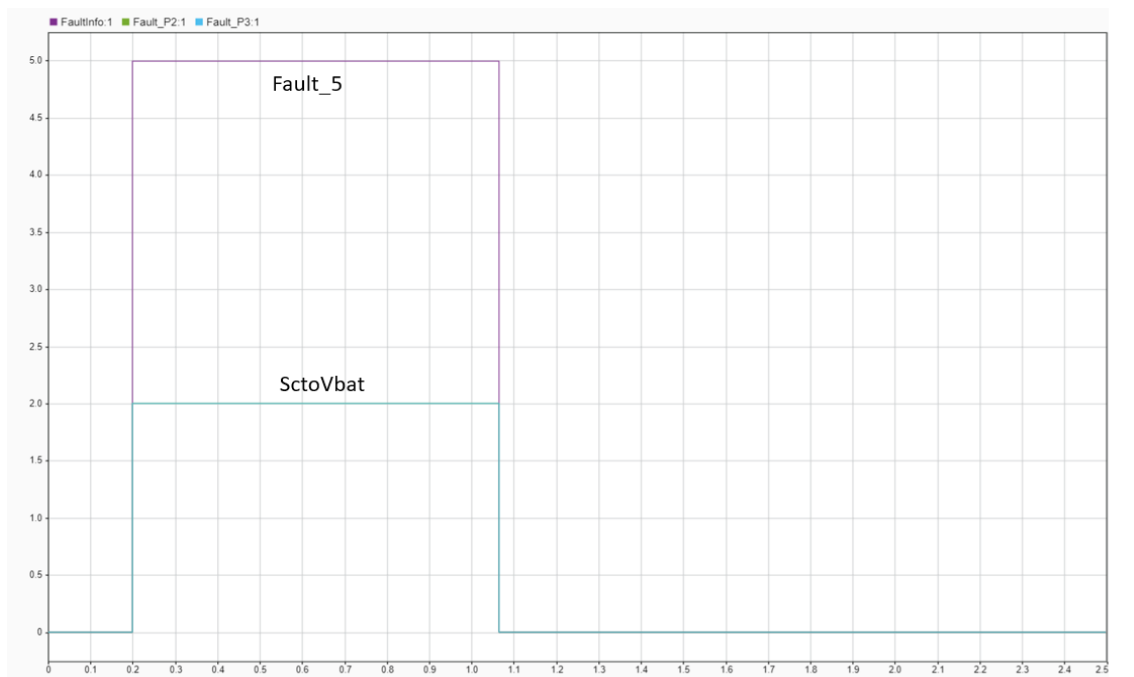


Figure 105: Faults detected with Montecarlo pins voltage signals (configuration without Accessory)

Using the Montecarlo signals (Max) in the configuration without Accessory, the result is the following figure.

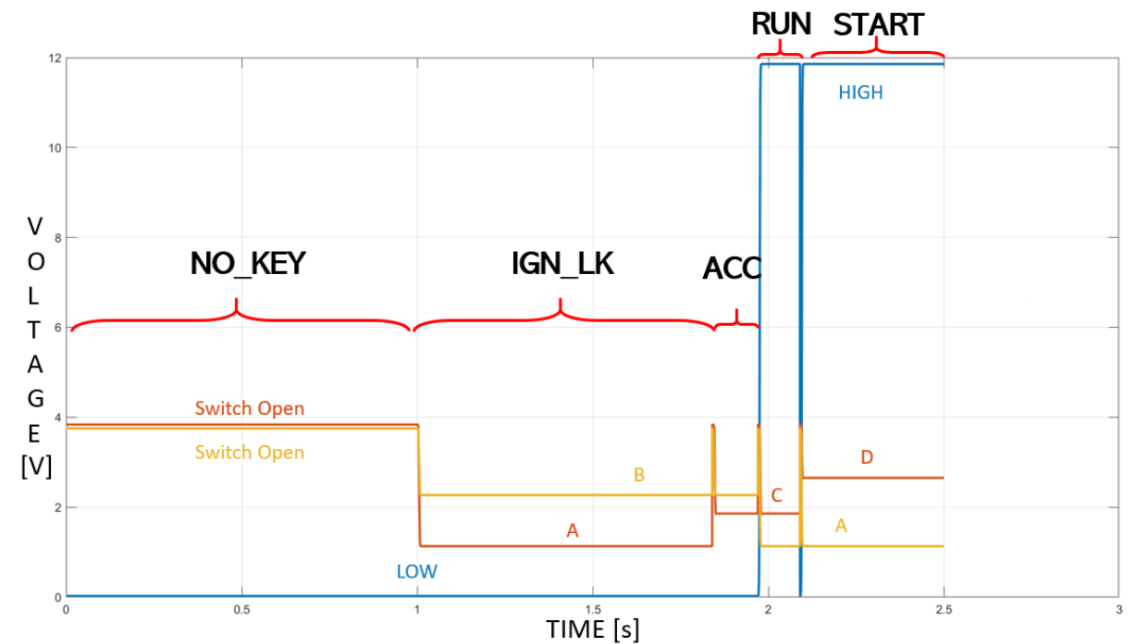


Figure 106: PSpice Montecarlo Pins voltage signals Max (configuration with Accessory)

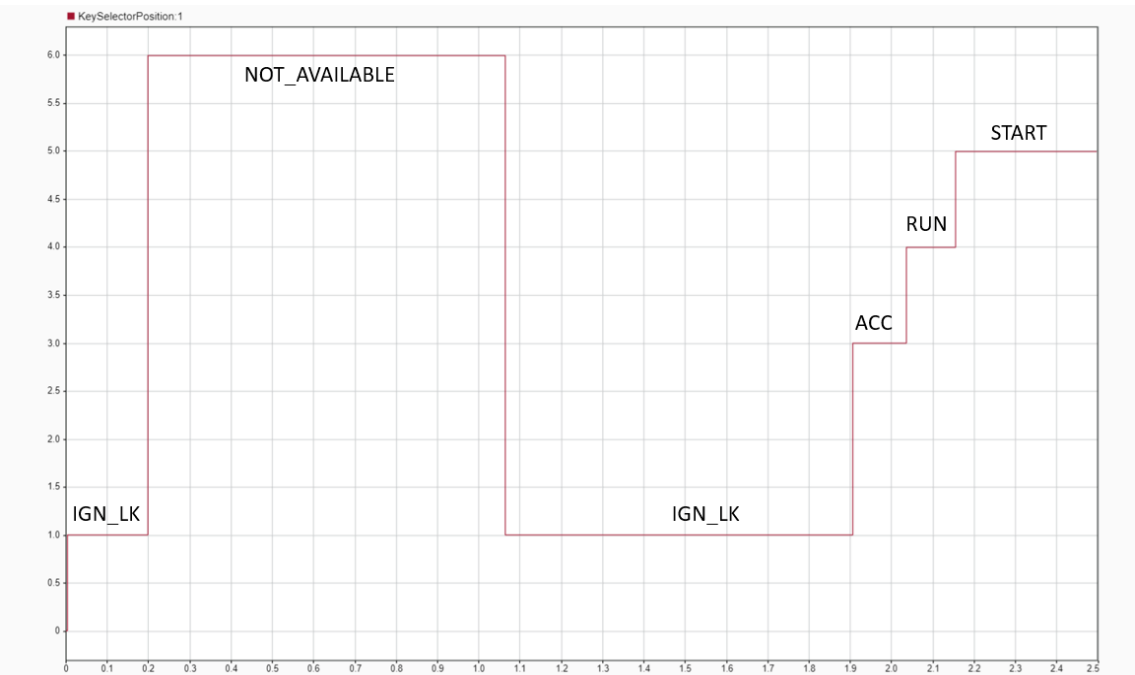


Figure 107: Key position detected with Montecarlo pins voltage signals (configuration with Accessory)

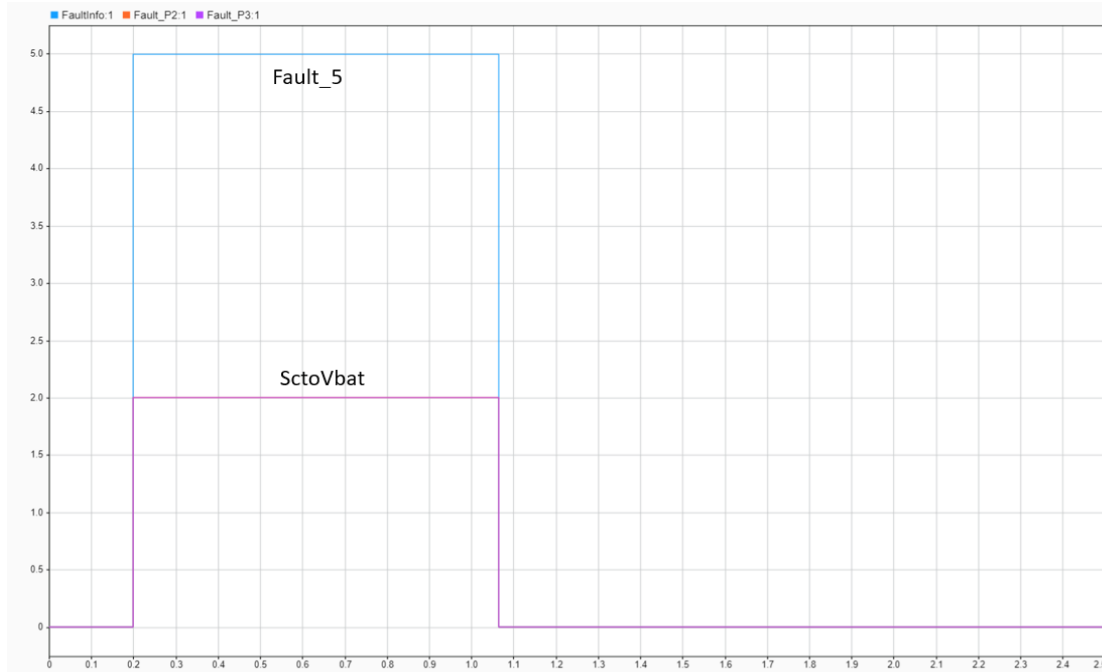


Figure 108: Faults detected with Montecarlo pins voltage signals (configuration with Accessory)

All the figures are the key position evolution over time; the x axis is the time and the y axis is the value assigned of the recognized key positions (IGB_LK =1, ACC = 3, RUN = 4, START = 5, NOT AVAILABLE = 6). The signal is initially set at *IGN_LK* because it is the value of the initialization. After 200 ms (debouncing time of a failure condition), a NOT_AVAILABLE situation is detected. In this case the analog values correspond to the NO_KEY values but the algorithm logic includes it in the NOT_AVAILABLE status.

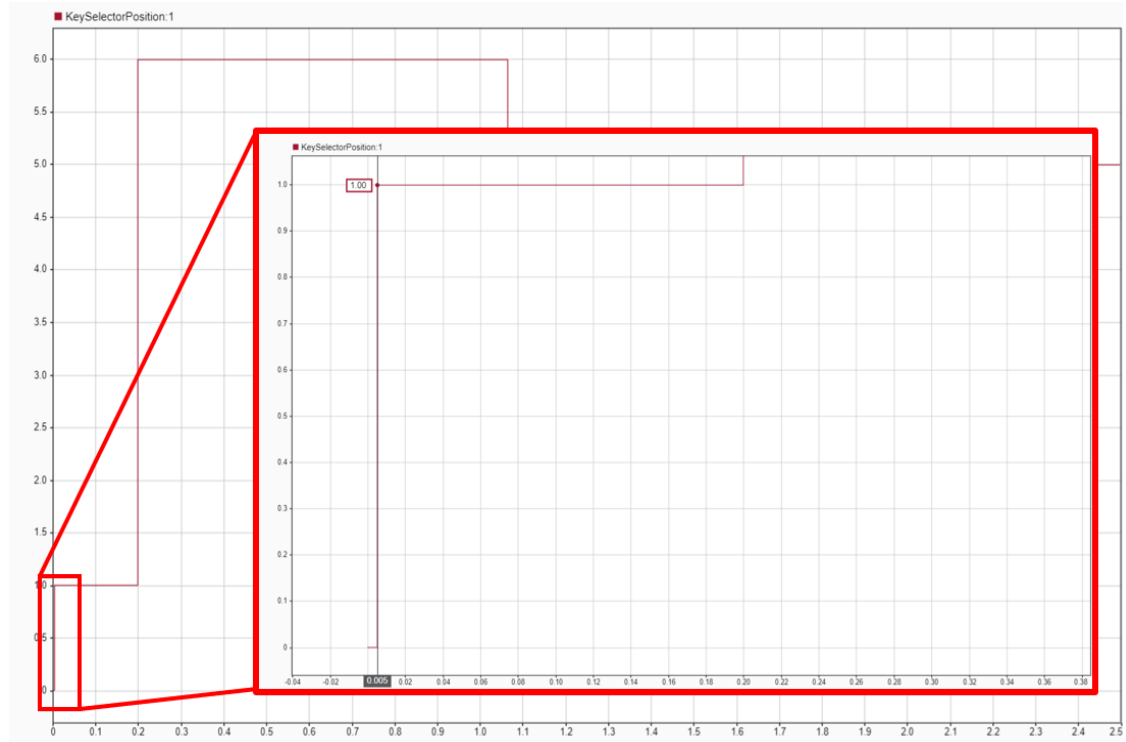


Figure 109: Zoom of figure 107

Zooming on the first part of each key selector position plot there is an initialization delay due to the sampling rate of 5 ms.



Figure 110: Analog and detected values comparison

Comparing the analog voltage value evolution over time and the key position detected, in every case there is a delay of 55 ms for every position change. The

delay is caused by the debounce timer (50 ms) plus the sampling rate (5 ms) to make the new key position value available.

In the figures that report failure conditions the x axis is the time and the y axis is the value assigned to general Fault info (No_Fault = 0, Fault_1 = 1, Fault_2 = 2, Fault_3 = 3, Fault_4 = 4, Fault_5 = 5) or to failure conditions on pin 2 and 3 (No_Fault = 0, SCtoGND = 1, SCtoVbat_2 = 2). A fault condition is recognized in correspondence of the NO_KEY position because the voltage value when the key is not inserted by the user is detected as a SctoVbat according to the hypothesis done in this analysis.

7. C-code generation

The C-code is automatically generated from the model and a part of it is shown as an example of the Main.

```
switch (KsmInput_P3_Thres) {
  case 8: {
    /* it corresponds to the value HIGH */
    if (KsmInput_N1_Thres == 8) {
      switch (KsmInput_P2_Thres) {
        case 24: {
          CtApKsm_CtApKsm1_keyselector_input_read_keystatus = IGN_LK;
          CtApKsm_CtApKsm1_keyselector_input_read_keyfailure = Fault_1;
          CtApKsm_CtApKsm1_keyselector_input_read_failure_input_p2 = No_Fault;
          CtApKsm_CtApKsm1_keyselector_input_read_failure_input_p3 = No_Fault;
          break;
        }
        case 8: {
          CtApKsm_CtApKsm1_keyselector_input_read_keystatus = Ksm_Not_Available;
          CtApKsm_CtApKsm1_keyselector_input_read_keyfailure = Fault_3;
          CtApKsm_CtApKsm1_keyselector_input_read_failure_input_p2 = No_Fault;
          CtApKsm_CtApKsm1_keyselector_input_read_failure_input_p3 = No_Fault;
          break;
        }
      }
    }
  }
}

[...]
```

Switch cases and innered *if-then-else* statement to identify all the possible threshold input combination and producing the related output value

An additional constraint has been taken into consideration during code generation: avoiding the using of floating data in order to be compliant with several ECU microcontrollers used in the automotive field.

This aspect has required some scaling operations inside the code. For instance, the threshold evaluation:

```
if (KsmVoltage_P3 >= ((uint8) (Volt_am_Offset + ((uint8) (Par_f * Volt_am_LSB)))) {
```

```
    /* Switch:
```

```
CtApKsm_frame/CtApKsm/RCtApKsmVoltageThres/LOGIC/8Bits_Threshold_Logic_Block1/Switc
```

```
    h4
```

```
    Variable 'SCTApKsm45_Switch4' replaced by 'Aux_U8' */
```

```
Aux_U8 = Bit3_threshold1_mask_bit_extraction;
```

8. Conclusion and future trends

The final goal of this thesis is to release the model of an algorithm that recognizes the key positions and the possible faults with an appropriate accuracy. The simulation results confirm the goal is reached applying the same methods, process and tool used to generate the code suitable for production.

The thesis covers the development of a mechanical model to simulate the key selector angular speed in order to obtain the transient time of every key position. The results of the mechanical model are used in the electrical simulation to generate some analog voltage signals that are used to test the algorithm.

From mechanical and electrical models, we obtain signals in advance of the final part availability. It allows to test the algorithm with signals that are closer to the real behaviour than the ones that a test engineer can create as stimulus.

The simulation highlights that in the following use case, the key position sequence needs to be evaluated:

when the user gets into the car and the user does not insert the key, the key position is set at IGN_LK and then at NOT_AVAILABLE (refer to the figures from Figure 98 to Figure 107Figure 108) until the user inserts the key and the algorithm sets the IGN_LK value.

It should be investigated what the effect is in the application layer:

- what happens in the vehicle?
- what are the consequences for the user?
- is a fault stored?

An improvement should be to detect when the key is not inserted by the user adding another threshold to distinguish the NO_KEY value from the OC/SctoVbat value.

The approach to create a virtual model of the final components in advance is a good choice. It is an effective way to detect in the early stage of the development if the implementation or the requirements are correct and if the system has the behaviour expected by the customer.

9. Acronyms

- ❖ **ECU** Electronic Control Unit
- ❖ **API** Application Program Interfaces
- ❖ **AUTOSAR** Automotive Open System Architecture
- ❖ **SWC** SoftWare Component
- ❖ **RTE** RunTime Environment
- ❖ **VFB** Virtual Functional Bus
- ❖ **BSW** Basic SoftWare
- ❖ **OS** Operating System
- ❖ **IGN_LK** Ignition Lock
- ❖ **ACC** Accessory
- ❖ **RE** Runnable Entity

10. Bibliography

- [1] Ignition positions (Key Access): <http://www.buiclub.com/info-2530.html>
- [2] EMEA_SWF_Autosar_Application
- [3] EMEA_SWF_ModelBasedDesign
- [4] SWC_partitioning_with_Init_and_Threshold_Res
- [5] Classic platform: <https://www.autosar.org/standards/classic-platform/>

11. Table of figures

Figure 1: Example of interaction between mechanical and electronic	6
Figure 2: Key selector (https://www.gapaudio.it/ilx-702d-500x-e-ine-w710d-500x/).....	7
Figure 3: Transfer function	8
Figure 4: Wire diagram	9
Figure 5: Analog input pull-down ECU interface	10
Figure 6: Analog input - Pull-up ECU interface	10
Figure 7: Picture of a key selector for illustration purpose only.....	11
Figure 8: Angle position 0°	11
Figure 9: Angle position 70°	12
Figure 10: Angle position 105°	12
Figure 11: Angle position 140°	12
Figure 12: Torques applied on key selector	13
Figure 13: Free body diagram of the cylinder	14
Figure 14: Free body diagram of the switch	15
Figure 15: Torques applied on rotational.....	16
Figure 16: Key selector mechanical model.....	18
Figure 17: User torque	18
Figure 18: User torque clockwise	19
Figure 19: Structure of the main logic	19
Figure 20: Torque implementation (left part)	20
Figure 21: if block return activation (lower part).....	20
Figure 22: if block for the physical block for the clockwise part (upper part)	21
Figure 23: Angle conversion (right part)	21
Figure 24: if block for the physical block for the counterclockwise part (lower part).....	22
Figure 25: if block for the rotation direction	22
Figure 26: Simulink model for the friction torque	23
Figure 27: If block for the spring length variation.....	24
Figure 28: If block for the spring length variation.....	25
Figure 29: Main part of the friction torque model	25
Figure 30: Simulink model for the elastic torque.....	26
Figure 31: Damping torque model.....	26
Figure 32: Damping torque model.....	27
Figure 33: From the top to the bottom: friction torque, elastic torque and damping torque (configuration without Accessory)	27
Figure 34: From the top to the bottom: friction torque, elastic torque and damping torque (configuration with Accessory)	28
Figure 35: The angle position (degree vs time), the angular speed (rpm vs time) and the user torque (N*m) in the configuration without Accessory	28
Figure 36: The angle position (degree vs time), the angular speed (rpm vs time) and the user torque (N*m) in the configuration with Accessory	29
Figure 37: Transient time from Ignition lock to Run (configuration without Accessory)	30
Figure 38: Transient time from Run to Start (configuration without Accessory)	30
Figure 39: Transient time from Ignition lock to Accessory (configuration with Accessory)	31

Figure 40: Transient time from Accessory to Run (configuration with Accessory)	32
Figure 41: Transient time from Run to Start (configuration with Accessory)	32
Figure 42: Holding time with Accessory position	33
Figure 43: Voltage divider (example realized with PSpice).....	34
Figure 44: Key Selector (key selector switch) circuit	35
Figure 45: PSpice logo.....	37
Figure 46: Switch symbol	38
Figure 47: Resistance symbol.....	38
Figure 48: Variable resistance symbol	38
Figure 49: Ground symbol.....	38
Figure 50: DC voltage source symbol	38
Figure 51: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (without Accessory).....	39
Figure 52: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (with Accessory)	39
Figure 53: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (without Accessory position).....	40
Figure 54: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (with Accessory position).....	41
Figure 55: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration without Accessory.....	42
Figure 56: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration with Accessory.....	42
Figure 57: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (without Accessory position)	43
Figure 58: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 1 (with Accessory)	43
Figure 59: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (without Accessory position).....	44
Figure 60: Equivalent electrical circuit of key selector switch realized in PSpice relating to Pin 2 and 3 (with Accessory position).....	45
Figure 61: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration without Accessory.....	46
Figure 62: Simulated signals on pin 1 (red), pin 2 (green) and pin 3 (blue) in the configuration with Accessory.....	46
Figure 63: Particular of Pin 1 voltage signal in the configuration without Accessory	47
Figure 64: Montecarlo simulation (Min) of the voltage value on pin 1 without Accessory	47
Figure 65: Montecarlo simulation (Max) of the voltage value on pin 1 with Accessory. 48	
Figure 66: Montecarlo simulation (Max) of the voltage value on pin 2 without Accessory	48
Figure 67: Montecarlo simulation (Min) of the voltage value on pin 2 with Accessory..	49
Figure 68: Montecarlo simulation (Min) of the voltage value on pin 3 without Accessory	49
Figure 69: Montecarlo simulation (Max) of the voltage value on pin 3 with Accessory. 50	
Figure 70: Algorithm description	51
Figure 71: INPUT 1 value vs voltage value at pin 1.....	52
Figure 72: INPUT 2 value vs voltage value at pin 2.....	53

Figure 73: INPUT 3 value vs voltage value at pin 3.....	54
Figure 74: Typical	55
Figure 75: IGN_IgnitionSW_KL15 is disturbed for a short time	56
Figure 76: IGN_IgnitionSW_KL15 is SCtoGND.....	56
Figure 77: IGN_IgnitionSW_KL15 and IGN_IgnitionSW_PlausibilitySignal are shorted	57
Figure 78: Algorithm main logic flowchart.....	60
Figure 79: AUTOSAR logo.....	61
Figure 80: AUTOSAR architecture structure	62
Figure 81: AUTOSAR standard port-icons of ports interfaces	63
Figure 82: Software architecture scheme.....	65
Figure 83: SWC toolchain	66
Figure 84: General view of the thresholds runnable	68
Figure 85: General view of the Input_P3 logic.....	68
Figure 86: Logic pin 3 quantization	69
Figure 87: General view of the main runnable.....	70
Figure 88: General view of the detection block.....	71
Figure 89: ACC and NO_ACC variant subsystem	72
Figure 90: Acquisition logic stateflow	73
Figure 91: Dashboard of the key selector model	74
Figure 92: Knobs used to test the Simulink model	75
Figure 93: Result of a model test	75
Figure 94: Print screen simulation (IGN_LK status)	76
Figure 95: Print screen simulation (RUN status)	77
Figure 96: Print screen simulation (START status)	78
Figure 97: PSpice Nominal Pins voltage signals (configuration without Accessory)	79
Figure 98: Key position detected with nominal pins voltage signals (configuration without Accessory).....	80
Figure 99: Faults detected with nominal pins voltage signals (configuration without Accessory).....	80
Figure 100: PSpice Nominal Pins voltage signals (configuration with Accessory)	81
Figure 101: Key position detected with nominal pins voltage signals (configuration with Accessory).....	81
Figure 102: Faults detected with nominal pins voltage signals (configuration with Accessory).....	82
Figure 103: PSpice Montecarlo Pins voltage signals Min (configuration without Accessory).....	82
Figure 104: Key position detected with Montecarlo pins voltage signals (configuration without Accessory).....	83
Figure 105: Faults detected with Montecarlo pins voltage signals (configuration without Accessory).....	83
Figure 106: PSpice Montecarlo Pins voltage signals Max (configuration with Accessory)	84
Figure 107: Key position detected with Montecarlo pins voltage signals (configuration with Accessory).....	84
Figure 108: Faults detected with Montecarlo pins voltage signals (configuration with Accessory).....	85
Figure 109: Zoom of figure 107	86

Figure 110: Analog and detected values comparison.....	86
--	----

12. List of tables

Table 1: Angle description and user functionalities	12
Table 2: Holding time without Accessory position	31
Table 3: Switches position vs key position	35
Table 4: Equivalent impedance and pin voltage per position	36
Table 5: Pin voltage value per position.....	36
Table 6: PSpice library elements used for the electrical model	38
Table 7: Pin quantized voltage value per position	54
Table 8: key failure conditions	58
Table 9: Pin 2 and Pin 3 failure conditions	58