MBSD – Parking Pawl Example

Jacopo Sini

# Purposes

* Design a controller and a plant model for a *parking pawl* device of an automatic transmission.
* Generate the code of the controller by Simulink Embedded Coder.

The *parking pawl* is a pin, inserted inside the final transmission gear tooths, to lock the wheels when the vehicle is parked.

It has to be inserted when the gear selector is set to the Park (P) and removed in all the other positions.

# Parking pawl description

A bidirectional direct current motor moves the parking pawl, moving it forward (to insert it, locking the gear) or backward (to remove it, freeing the gear).

## External physical interfaces

|  |  |  |
| --- | --- | --- |
| **Name** | **Direction** | **Type** |
| **Enable\_DI** | Input | Discrete |
| **Insert\_DI** | Input | Discrete |
| **Extract\_DI** | Input | Discrete |
| **Pos\_mm** | Output | Analog |
| **Current\_A** | Output | Analog |
| **Inserted\_Pos\_Switch\_DO** | Output | Discrete |
| **Extracted\_Pos\_Switch\_DO** | Output | Discrete |

## Description of the whole system

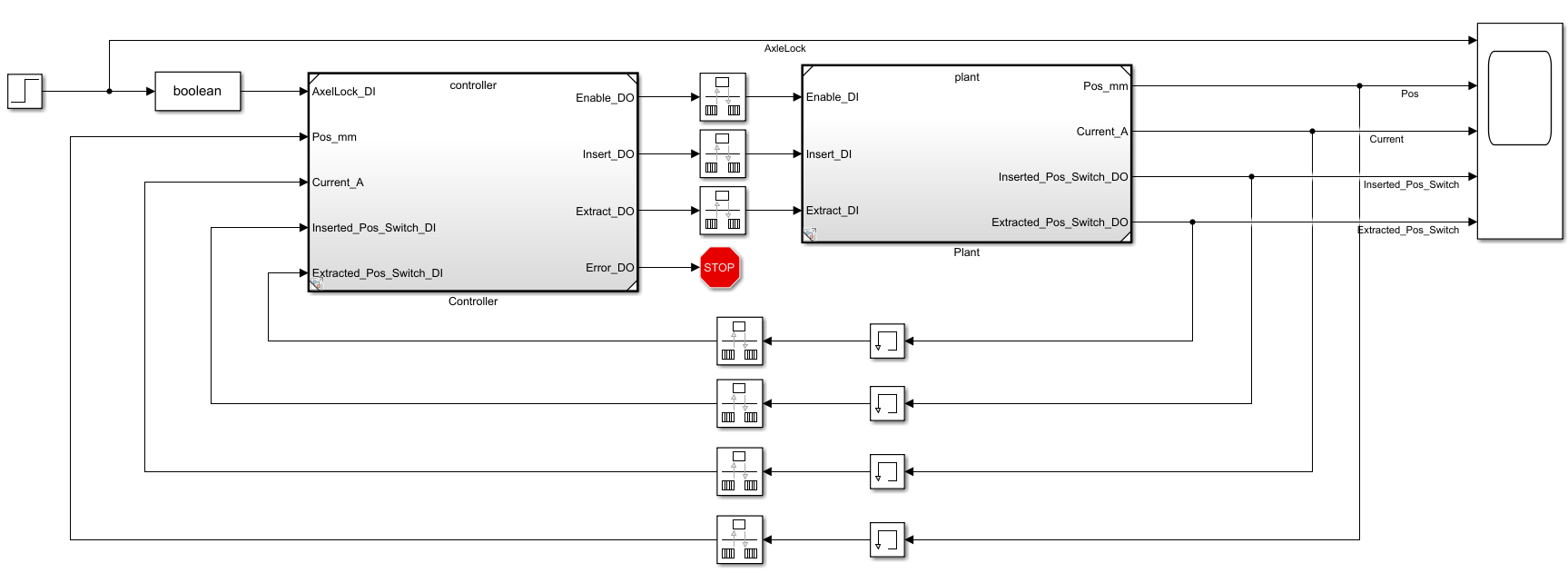


Figure 1 The Park pawl (plant) and its controller with all the needed I/O interfaces

The parking pawl device works as follows.

* When Enable\_DI is false, it stops the motor regardless of all the other inputs.
* When Enable\_DI is true, while Insert\_DI and Extract\_DI signals are both false, it stops the motor.
* When the Enable\_DI and Insert\_DI signals are both true, the pawl moves in the direction to insert it into the gear. When it reaches the final position, it closes the inserted position switch making the signal Inserted\_Pos\_Switch\_DOtrue.
* When the Enable\_DI and Extract\_DI signals are both true, the pawl moves in the direction to insert it into the gear. When it reaches the final position, it closes the removed position switch making the signal Extracted\_Pos\_Switch\_DOtrue.
* When Enable\_DI, Insert\_DI, and Extract\_DI signals are true, it stops the motor.

The parking pawl subsystem can find itself in three different states, depending on position switches states[[1]](#footnote-1):

Extracted\_Pos\_Switch\_DO == false

Extracted\_Pos\_Switch\_DO == true

Inserted\_Pos\_Switch\_DO == true

Inserted\_Pos\_Switch\_DO == false

Some assumption:

* A condition with both position switches Inserted\_Pos\_Switch\_DO, and Extracted\_Pos\_Switch\_DOequals to true is impossible from the mechanical point of view; hence it indicates a failure of one or both the switches.
* The position inserted corresponds to Pos\_mm = 0 ± 1 mm, while the position extracted to Pos\_mm = 12 ± 1mm. The controller can use this measurement only to perform plausibility checks.
* The motor moves the pin at a speed of about vf= = 3 mm/s in both the directions, absorbing a current Im (nominal) = 1 A. Its short-circuit current is equal to Im (short circuit) = 4 A, while the current protection shall intervene when a current Im> MAX\_CURR\_A = 1.5 A is measured for more than 500 ms.
* The controller shall stop the motor when it engages a position switch.
* The plant model must be able to simulate the short circuit current when the pin reaches the end positions Pos\_mm = 0 and Pos\_mm = 12, locking the motor.
* The simulation of a stuck condition of the pin in a random position is possible, but not simulated by the plant model. In any case, it has to be kept into account in the controller design (at least protecting the motor from overheating).

## Hints

A simplified model of the parking pawl can be described by the equation

with the output of the integrator saturated between 0 and 12 mm.

The current absorbed by the motor can be computed, in Amperes, with a very simplified DC motor model, as:

with and |. When the motor is not supplied, of course, .

# Controller SW Unit specifications

This software unit shall be implemented as a periodic task with a period of 10 ms (100 Hz)

## Interfaces

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Unit\*** | **Type** | **Data Type** | **Dimension** | **Min** | **Max** |
| AxleLock\_DI | N/A | Input | boolean | 1x1 | 0 *(false)* | 1 *(true)* |
| Pos\_mm | mm | Input | real32 | 1x1 | -2.0 | 14.0 |
| Current\_A | A | Input | real32 | 1x1 | -2.0 | 10.0 |
| Inserted\_Pos\_Switch\_DI | N/A | Input | boolean | 1x1 | 0 *(false)* | 1 *(true)* |
| Extracted\_Pos\_Switch\_DI | N/A | Input | boolean | 1x1 | 0 *(false)* | 1 *(true)* |
| Enable\_DO | N/A | Output | boolean | 1x1 | 0 *(false)* | 1 *(true)* |
| Insert\_DO | N/A | Output | boolean | 1x1 | 0 *(false)* | 1 *(true)* |
| Extract\_DO | N/A | Output | boolean | 1x1 | 0 *(false)* | 1 *(true)* |
| Error\_DO | N/A | Output | boolean | 1x1 | 0 *(false)* | 1 *(true)* |

## Use cases

|  |  |
| --- | --- |
| **Use Case ID** | **Description in natural language** |
| TMS\_UC1  *Extraction of the park pawl* | **When AxleLock\_DI is false**, and the system is yet in the required state (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == false), no further actions are required.  **When AxleLock\_DI is false**, and the system is in the opposite state w.r.t. the required one (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == true), it is necessary to enable and run the motor in the extraction direction.  **When AxleLock\_DI is false**, and the system is in an undefined state (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == false), it is necessary to enable and run the motor in the extraction direction.  **When AxleLock\_DI is false**, and the system is in a faulty state (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == true), it is necessary to raise an error signal to the upper level software. |
| **Requirement ID** | **Description in mathematical notation** |
| TMS\_UC1\_REQ1 | IF (AxleLock\_DI == false) AND (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == false) THEN  Enable\_DO = false;  Insert\_DO = false;  Extract\_DO = false;  Error\_DO = false;  ELSE IF (AxleLock\_DI == false) AND (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == true) THEN  Enable\_DO = true;  Insert\_DO = true;  Extract\_DO = false;  Error\_DO = false;  ELSE IF (AxleLock\_DI == false) AND (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == false) THEN  Enable\_DO = true;  Insert\_DO = true;  Extract\_DO = false;  Error\_DO = false;  ELSE IF (AxleLock\_DI == false) AND (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == true) THEN  Enable\_DO = false;  Insert\_DO = false;  Extract\_DO = false;  Error\_DO = true;  END |

|  |  |
| --- | --- |
| **Use Case ID** | **Description in natural language** |
| TMS\_UC2  *Insertion of the park pawl* | **When AxleLock\_DI is true**, and the system is yet in the required state (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == true), no further actions are required.  **When AxleLock\_DI is false**, and the system is in the opposite state w.r.t. the required one (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == false), it is necessary to enable and run the motor in the insertion direction.  **When AxleLock\_DI is false**, and the system is in an undefined state (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == false), it is necessary to enable and run the motor in the insertion direction.  **When AxleLock\_DI is false**, and the system is in a faulty state (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == true), it is necessary to raise an error signal to the upper level software. |
| **Requirement ID** | **Description in mathematical notation** |
| TMS\_UC2\_REQ1 | IF (AxleLock\_DI == true) AND (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == true) THEN  Enable\_DO = false;  Insert\_DO = false;  Extract\_DO = false;  Error\_DO = false;  ELSE IF (AxleLock\_DI == true) AND (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == false) THEN  Enable\_DO = true;  Insert\_DO = false;  Extract\_DO = true;  Error\_DO = false;  ELSE IF (AxleLock\_DI == true) AND (Extracted\_Pos\_Switch\_DI == false) AND (Inserted\_Pos\_Switch\_DI == false) THEN  Enable\_DO = true;  Insert\_DO = false;  Extract\_DO = true;  Error\_DO = false;  ELSE IF (AxleLock\_DI == true) AND (Extracted\_Pos\_Switch\_DI == true) AND (Inserted\_Pos\_Switch\_DI == true) THEN  Enable\_DO = false;  Insert\_DO = false;  Extract\_DO = false;  Error\_DO = true;  END |

### Overcurrent protection

|  |  |
| --- | --- |
| **Use Case ID** | **Description in natural language** |
| TMS\_UC3  *Overcurrent protection* | **When Enable\_DO is true**, and the motor is moving, hence Insert\_DO is true *xor*[[2]](#footnote-2) Extract\_DO is true, the controller shall monitor that the current not exceed the maximum tolerable value MAX\_CURR\_A for more than 500 ms. |
| **Requirement ID** | **Description in mathematical notation[[3]](#footnote-3)** |
| TMS\_UC2\_REQ1 | IF (Enable\_DO == true) AND  (Current\_A > MAX\_CURR\_A and LastedTime < 50)  LastedTime ++;  ELSE IF (Enable\_DO == true) AND  (Current\_A > MAX\_CURR\_A and LastedTime >= 50)  Enable\_DO = false;  ~~Insert\_DO = false;~~  ~~Extract\_DO = false;~~  Error\_DO = true;  ELSE  LastedTime = 0;  END |

1. It does not manage all the possible situations, like when both position switches equal to true. [↑](#footnote-ref-1)
2. The *or* case is managed by the hardware, that shuts down the motor. [↑](#footnote-ref-2)
3. Under the assumption that the controller runs at 100 Hz. [↑](#footnote-ref-3)