Protocol Validation Homework Assignment: Movable Patient Support for an MRI Scanner

Denis Mazzucato, Francesco Parolini
Vrije Universiteit Amsterdam

Contents

1 Requirements **2**
1.1 Safety properties . . . 2
1.2 Liveness properties . . . 2

2 Interactions List **3**
2.1 Vertical and Horizontal motor . . . 4
2.2 Bed position . . . 4
2.3 Calibration mode . . . 4
2.4 Console buttons . . . 5
2.5 Dock unit . . . 5
2.6 Emergency mode . . . 5

3 Requirements-interactions mapping **6**
3.1 Safety properties . . . 6
3.2 Liveness properties . . . 9

4 Architecture **13**
4.1 Console . . . 14
4.2 Output . . . 14
4.3 Sensors . . . 15
4.4 Processor . . . 16
4.5 Behavior . . . 16
4.5.1 Initial state . . . 16
4.5.2 Emergency mode . . . 17
4.5.3 Movement above the standard height . . . 17

5 Implementation **17**
5.1 Representation of the state . . . 17
5.2 Representation of the position . . . 18
5.3 Calibration . . . 18

1 Requirements

Here we state the global requirements that we have identified either explicitly or implicitly in the text. We divide the requirements in two categories: *Safety* and *Liveness* and for the sake of clarity we assign a code to each requirement.

1.1 Safety properties

- **SF01**: If the emergency mode is triggered, no motorized movement can occur.
- **SF02**: If one motor is on, the corresponding brake must not be applied.
- **SF03**: If the MPSP is not in the rightmost position, the undock message should never be sent to the scanner.
- **SF04**: If the MPSP is at the standard height and docked and the button up is pressed, then that button must be released before the inward movement is commenced.
- **SF05**: If the MPSP is docked and calibrated, the bed cannot be moved above the standard height.
- **SF06**: If the MPSP is docked and not at the rightmost position, no motorized vertical movement can occur.
- **SF07**: If the MPSP is uncalibrated or undocked and either the up or down button is pressed, the bed can move only up and down.
- **SF08**: If the MPSP is undocked, the bed must be in the rightmost position.
- **SF09**: If the MPSP is undocked, the horizontal brake must be applied.
- **SF10**: The MPSP can not move under the lowermost position, above the uppermost position and to the left of the leftmost position.
- **SF11**: If the vertical motor is off, the vertical break must be applied.

1.2 Liveness properties

- **LV01**: The system never ends up in a state where it cannot perform any action (*Deadlock freeness*).
- **LV02**: If the emergency mode is triggered, the horizontal brake, if applied, must be released, so that the medical staff can drag the patient outside the scanner.
- **LV03**: If the emergency mode is activated, the resume button must trigger the normal mode.

- **LV04**: If the MPSP is docked, the stop button must trigger emergency mode.
- **LV05**: If the MPSP is docked and at the rightmost position, the undock button must send one undock message to the scanner.
- **LV06**: If the MPSP is undocked, the reset button must forget the standard height.
- **LV07**: If the MPSP is docked and at the rightmost position, the reset button sets the standard height.
- **LV08**: If the MPSP is docked and at the standard height and at the rightmost position and the button up is pressed, the bed moves into the scanner.
- **LV09**: If the MPSP is docked and not at the rightmost position and the button down is pressed, the bed moves to the left.
- **LV10**: If the MPSP is above the lowermost position and outside of the scanner and the button down is pressed, the bed moves downwards.
- **LV11**: If the MPSP is undocked and below the uppermost position and the button up is pressed, the bed moves upwards.
- **LV12**: If the MPSP is docked, calibrated and below the standard height and the button up is pressed, the bed moves upwards.
- **LV13**: If the MPSP is docked, uncalibrated and below the uppermost position and the button up is pressed, the bed moves upwards.

2 Interactions List

In this section we list all the interactions of the control system with the outside world, grouped according to the behavior module.

2.1 Vertical and Horizontal motor

Name	*Behavior*
`motorLeft`	The horizontal motor M2 is turned on and makes an inward movement
`motorRight`	The horizontal motor M2 is turned on and makes an outward movement
`motorUp`	The vertical motor M1 is turned on and makes an upward movement
`motorDown`	The vertical motor M1 is turned on and makes an downward movement
`horizontalMotorOff`	The horizontal motor M2 is turned off
`verticalMotorOff`	The vertical motor M1 is turned off
`applyHorizontalBrake`	Apply the horizontal brake
`applyVerticalBrake`	Apply the vertical brake
`releaseHorizontalBrake`	Release the horizontal brake
`releaseVerticalBrake`	Release the vertical brake

2.2 Bed position

Name	*Behavior*
`leftmostReached`	Indicates that the bed is completely inside the scanner
`rightmostReached`	Indicates that the bed is at rightmost position
`uppermostReached`	Indicates that the bed is at the highest position
`lowermostReached`	Indicates that the bed is at the lowest position
`standardHeightReached`	Indicates that the bed is at the standard height, if possible
`standardHeightPassed`	Indicates that the bed has passed the standard height without stopping

2.3 Calibration mode

Name	*Behavior*
`resetStandardHeight`	Reset the standard height
`setStandardHeight`	Set a new standard height

2.4 Console buttons

Name	*Behavior*
`tapStop`	The Stop button is pressed and immediately released
`pressStop`	The Stop button is pressed
`releaseStop`	The Stop button is released
`tapResume`	The Resume button is pressed and immediately released
`pressResume`	The Resume button is pressed
`releaseResume`	The Resume button is released
`tapUndock`	The Undock button is pressed and immediately released
`pressUndock`	The Undock button is pressed
`releaseUndock`	The Undock button is released
`tapUp`	The Up button is pressed and immediately released
`pressUp`	The Up button is pressed
`releaseUp`	The Up button is released
`tapDown`	The Down button is pressed and immediately released
`pressDown`	The Down button is pressed
`releaseDown`	The Down button is released
`tapReset`	The Reset button is pressed and immediately released
`pressReset`	The Reset button is pressed
`releaseReset`	The Reset button is released

2.5 Dock unit

Name	*Behavior*
`dock`	Indicates that the platform has been docked with the scanner
`undock`	Unlock the platform from the docking unit

2.6 Emergency mode

Name	*Behavior*
`emergencyMode`	Indicates that the system is in emergency mode
`normalMode`	Indicates that the system has returned to the normal operating mode

3 Requirements-interactions mapping

Here we translate the global requirements in terms of action sequences. Some cases can be quite complicated, while their corresponding *mu-calculus* formulas are compact and quite clear. Sometimes to make the translations more readable we add parenthesis to clarify the meaning. Some cases have multiple conditions that should hold simultaneously.

3.1 Safety properties

- **SF01**: If `emergencyMode` not followed by `normalMode` occurred, then no `motorLeft`, `motorRight`, `motorUp` or `motorDown` can occur.
- **SF02**:
  - If `applyVerticalBrake` not followed by `releaseVerticalBrake` occurred, then no `motorUp` or `motorDown` can occur.
  - If `applyHorizontalBrake` not followed by `releaseHorizontalBrake` occurred, then no `motorLeft` or `motorRight` can occur.
  - If `motorLeft` or `motorRight` not followed by `horizontalMotorOff` occurred, then no `applyHorizontalBrake` can occur.
  - If `motorUp` or `motorDown` not followed by `verticalMotorOff` occurred, then no `applyVerticalBrake` can occur.
- **SF03**: If `motorLeft` not followed by `rightmostReached` occurred, then `undock` can not occur.
- **SF04**:
  - *This is the case in which the MPSP gets docked and after that reaches the standard height.*
    If `dock` not followed by `undock` occurred, subsequently followed by `standardHeightReached` not followed by (`undock`, `motorDown` or `motorUp`), subsequently followed by `pressUp` but not by `releaseUp`, then no `motorLeft` can occur.
  - *This is the case in which the MPSP first reaches the standard height and then gets docked and not moved or re-calibrated.*
    If `standardHeightReached` not followed by (`resetStandardHeight`, `motorUp` or `motorDown`) occurred and is subsequently followed by `dock` not followed by (`undock`, `motorUp` or `motorDown`), and is then followed by `pressUp` not followed by `releaseUp`, then no `motorLeft` can occur.
- **SF05**:
  - *In this case first the MPSP gets docked and then the standard height is reached.*

If `dock` not followed by `undock` occurred, followed by `standardHeightReached` not followed by (`motorDown` or `undock`), then no `motorUp` can occur.

- *This is the complementary case: first the MPSP reaches the standard height and then it gets docked and not moved or re-calibrated.*
  If `standardHeightReached` not followed by (`resetStandardHeight`, `motorUp` or `motorDown`) occurred, followed by `dock` not followed by (`motorDown` or `undock`), then no `motorUp` can occur.

- **SF06**: If `dock` not followed by `undock` occurred and then `motorLeft` not followed by `rightmostReached` occurred, then no `motorUp` or `motorDown` can occur.

- **SF07**:
  - *This represents the case in which the MPSP has never been calibrated.*
    If no `setStandardHeight` occurred and then `tapDown`,`tapUp` `pressDown` or `pressUp` not followed by `setStandardHeight` occurred, then no `motorLeft` or `motorRight` can occur.
  - *This is the case in which the MPSP has been calibrated and subsequently uncalibrated.*
    If `resetStandardHeight` not followed by `setStandardHeight` occurred, subsequently followed by `tapDown` `tapUp` `pressDown` or `pressUp` not followed by `setStandardHeight` then no `motorLeft` or `motorRight` can occur.
  - *This represents the case in which the MPSP has never been docked.*
    If no `dock` action occurred and `tapDown` `tapUp` `pressDown` or `pressUp` occurred not followed by `dock`, then no `motorLeft` or `motorRight` can occur.
  - *This is the case in which the MPSP has been docked and subsequently undocked.*
    If `undock` not followed by `dock` occurred, subsequently followed by `tapDown`,`tapUp` `pressDown` or `pressUp` not followed by `dock`, then no `motorLeft` or `motorRight` can occur.

- **SF08**:
  - *This represents the case in which the MPSP has never been docked (and so it is at the rightmost position).*
    If no `dock` action occurred, then no `motorLeft` can occur.
  - *This represents the case in which the MPSP gets docked: if one left movement occurs, then the bed can not get undocked until one* ***`rightmostReached`*** *occurs. This implies that if it gets subsequently undocked it will be at the rightmost position.*
    If `dock` not followed by `undock` occurred, and it is subsequently followed by `motorLeft` not followed by `rightmostReached` then no `undock` can occur.

- *This, together with the previous condition of this requirement, implies that while the MPSP is undocked it must always be at the rightmost position, as when it is undocked it begins at the rightmost, and while undocked, it can not move.*
  If `undock` not followed by `dock` action occurred, then no `motorLeft` can occur.

- **SF09**:
  - *This represents the case in which the MPSP has never been docked (and then the horizontal brake is applied).*
    If no `dock` action occurred, then no `releaseHorizontalBrake` can occur.
  - *This represents the case in which the MPSP gets docked: if one release of the horizontal brake occurs, then the bed can't get undocked until one "applyHorizontalBrake" occurs. This implies that if it gets subsequently undocked the horizontal brake will be applied.*
    If `dock` not followed by `undock` occurred, and it is subsequently followed by `releaseHorizontalBrake` not followed by `applyHorizontalBrake`, then no `undock` can occur.
  - *This, together with the previous condition of this requirement, implies that while the MPSP is undocked the horizontal brake must always be applied, as when it gets undocked it is applied, and while undocked, it can't be released.*
    If `undock` not followed by `dock` action occurred, then no `releaseHorizontalBrake` can occur.

- **SF10**:
  - If `uppermostReached` not followed by `motorDown` occurred, then no `motorUp` can occur.
  - If `lowermostReached` not followed by `motorUp` occurred, then no `motorDown` can occur.
  - If `leftmostReached` not followed by `motorRight` occurred, then no `motorLeft` can occur.

- **SF10**:
  - If `uppermostReached` not followed by `motorDown` occurred, then no `motorUp` can occur.
  - If `lowermostReached` not followed by `motorUp` occurred, then no `motorDown` can occur.
  - If `leftmostReached` not followed by `motorRight` occurred, then no `motorLeft` can occur.

- **SF11**:

- If `verticalMotorOff` occurred, then the next action must be `applyVerticalBrake`.
- If `releaseVerticalBrake` occurred, then the next action must be `motorUp` or `motorDown`.

Important note: normally this requirement would be (informally) implemented as "every trace that contains `verticalMotorOff`, must contain `applyVerticalBrake` before", but we noticed that it is not possible to satisfy this and requirement **SF02** at the same time. This is due to the fact that either (the motor is turned off and after that the brake is applied [**SF02**]) or (the brake is applied and then the motor is turned off [**SF11**]) can hold. In order to be able to satisfy both requirements we decided to implement **SF11** as specified.

3.2 Liveness properties

- **LV01**: For any sequence of actions, one subsequent action can always be performed *(Deadlock freeness)*.
- **LV02**: Each `leftmostReached` that is not followed by `motorRight` or `emergencyMode` actions, and that is then followed by `emergencyMode`must eventually be followed by `releaseHorizontalBrake`.
- **LV03**: Each `emergencyMode` that is not followed by `normalMode`, but is subsequently followed by `pressResume` or `tapResume` must be eventually followed by `normalMode`.
- **LV04**: Each `dock` that is not followed by `undock`, but that is followed by `tapStop` or `pressStop` must be eventually be followed by `emergencyMode`.
- **LV05**:
  - *This is the case in which no left movement occurs.*
    Each `dock` that is not followed by `undock` or `motorLeft`, but that is followed by `tapUndock` or `pressUndock`, must be eventually followed by `undock`.
  - *This is the case in which left movement occurs.*
    Each `dock` that is not followed by `undock` or `motorLeft`, that is followed by `rightmostReached` not followed by `undock` or `motorLeft`, that is subsequently followed by `tapUndock` or `pressUndock`, must eventually be followed by `undock`.
- **LV06**: Each `setStandardHeight` which is not followed by (`resetStandardHeight` or `emergencyMode`) and that is subsequently followed by `undock` not followed by `dock` or `resetStandardHeight`, and that is then followed by `tapReset` or `pressReset` must be eventually followed by `resetStandardHeight`.

- **LV07**:
  - *This is the case in which no left movement occurs.*
    Each `dock` that is not followed by `undock` or `motorLeft`, but that is then followed by `tapReset` or `pressReset` must be eventually followed by `setStandardHeight`.
  - *This is the case in which left movement occurs.*
    Each `dock` that is not followed by `undock`, but that is followed by `rightmostReached` not followed by `undock` or `motorLeft`, that is subsequently followed by `tapReset` or `pressReset`, must be eventually followed by `setStandardHeight`.
- **LV08**:
  - *This is the case in which the standard height has been reached before docking the MPSP.*
    Each `dock` that is not followed by `undock`, but that is followed by `standardHeightReached` not followed by (`tapUp`, `pressUp`, `motorDown` or `undock`), and that is subsequently followed by `tapUp` or `pressUp` must be eventually followed by `motorLeft`.
  - *This is the case in which the standard height is reached after docking the MPSP.*
    Each `setStandardHeight` that is not followed by (`motorDown`, `motorUp` or `resetStandardHeight`), but that is followed by `dock` not followed by (`tapUp`, `pressUp`, `motorDown` or `undock`), and that is subsequently followed by `tapUp` or `pressUp`, must be eventually followed by `motorLeft`.
- **LV09**: *This is the case in which the MPSP is not in the lowermost position.*
  Each `motorLeft` that is not followed by `rightmostReached` that is subsequently followed by `tapDown` or `pressDown` must be eventually followed by `motorRight`.
- **LV10**: Each `motorUp` not followed by (`lowermostReached`, `motorLeft` or `emergencyMode`), that is followed by `tapDown` or `pressDown`, must be eventually followed by `motorDown`.
- **LV11**: From this point on, the properties we describe are quite complicated and many sub-cases are needed. It is not easy to clearly express some of them in natural language, and for this reason we try to explain them adding a short description of the case that we are considering.
  - *The uppermost has never been reached and the MPSP has never been docked.*
    If no `dock` or `uppermostReached` occurred and `pressUp` or `tapUp` occurred, then `motorUp` must eventually occur.

- *The MPSP has never been docked but the uppermost has been reached.*
  If `uppermostReached` not preceded or followed by `dock` action occurred, followed by `motorDown`(to be sure that the MPSP is at least one step under the uppermost) but not by `dock` or `uppermostReached`, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
- *The MPSP has been docked but has never reached the uppermost position.*
  If `undock` not preceded by `uppermostReached` or `emergencyMode` and also not followed by `dock` or `uppermostReached` occurred, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur. In is the case emergency mode cannot be triggered: this is due to the fact that in emergency mode the pressing of button up is ignored.
- *The MPSP has been docked and has reached the uppermost position after that.*
  If `undock` not followed by `dock` action occurred, followed by `uppermostReached` but not by `dock` action, and subsequently followed by `motorDown`, but not by a subsequent `dock` or `uppermostReached` if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
- *The uppermost has been reached before getting docked and the bed moves down after getting undocked.*
  If `uppermostReached` not followed by `motorDown` occurred, followed by `undock` but not by by `dock`, if it is then followed by `motorDown` not followed by `dock` or `uppermostReached` if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
- *The uppermost has been reached before getting docked and the bed move down while docked.*
  If `uppermostReached` occurred in a trace before `motorDown`, and then no `uppermostReached` or `emergencyMode` occurred, followed then by `undock` not followed by `dock` or `uppermostReached`, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.

- **LV12**:
  - *The MPSP gets docked, reaches the standard height and then moves down.*
    If `dock` not followed by `undock` occurred, followed then by `standardHeightReached` not followed by `undock`, and subsequently followed by a `motorDown` but not by `standardHeightReached` or `undock`, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
  - *The MPSP reaches the standard height, gets docked and after that moves down.*
    If `standardHeightReached` not followed by `resetStandardHeight`

occurred, subsequently followed by `dock` but not by `undock`, if after that a `motorDown` not followed by `standardHeightReached` or `undock` occurred, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.

- **LV13**: This formula is by far the most complicated one, due to the fact the we have to keep track of each possible way to get uncalibrated, docked and under the uppermost position in all possible ways.

  - *The MPSP has never been calibrated and has never reached the standard height.*
    If no `setStandardHeight` or `uppermostReached` occurred before `dock` and no `setStandardHeight`, `undock` or `uppermostReached` after it, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
  - *The MPSP has never been calibrated, reaches then uppermost position, gets docked and then moves down.*
    If no `setStandardHeight` occurred before `uppermostReached`, then no `motorDown` or `setStandardHeight` occurred before `dock`, and then no `undock` or `setStandardHeight` occurred before `motorDown`, and then no `undock` or `setStandardHeight` or `uppermostReached` occurred , if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
  - *The MPSP has never been calibrated, gets docked, reaches the uppermost position and then moves down.*
    If no `setStandardHeight` occurred before `dock`, and it is not followed by `undock` or `setStandardHeight`, if then `uppermostReached` occurred not followed by `undock` or `setStandardHeight`, if after that `motorDown` occurred not followed by `undock`, `setStandardHeight` or `uppermostReached` , if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
  - *The MPSP reaches the uppermost position, moves down, gets uncalibrated and then gets docked.*
    If `uppermostReached` appeared in a trace before `motorDown` not followed by `uppermostReached`, if then `resetStandardHeight`occurred not followed by `setStandardHeight` or `uppermostReached`, if then `dock` occurred not followed by `undock`, `setStandardHeight` or `uppermostReached` action , if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
  - *The MPSP reaches the uppermost position, gets uncalibrated, moves down and then gets docked.*
    If `uppermostReached` appeared in a trace before `resetStandardHeight` not followed by `setStandardHeight`, if then `motorDown` occurred not followed by `setStandardHeight` or `uppermostReached`,

and it is subsequently followed by `dock` but not by `undock`, `setStandardHeight` or `uppermostReached` , if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.

- *The MPSP gets uncalibrated, reaches the uppermost position, moves down and then gets docked.*
  If `resetStandardHeight` occurred not followed by `setStandardHeight`, if it is then followed by `uppermostReached` but not by `setStandardHeight`, if then `motorDown` occurred not followed by `setStandardHeight` or `uppermostReached`, if then `dock` occurred not followed by `undock`, `setStandardHeight` or `uppermostReached`, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
- *The MPSP gets uncalibrated, reaches the uppermost position, gets docked and then moves down.*
  If `resetStandardHeight` not followed by `setStandardHeight` occurred, if it is followed by `uppermostReached` but not by `setStandardHeight`, if it is subsequently followed by `dock` but not by `undock` or `setStandardHeight`, if it is then followed by `motorDown` but not by `setStandardHeight`, `uppermostReached` or `emergencyMode`, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.
- *The MPSP gets uncalibrated, gets docked, reaches the uppermost and then moves down.*
  If `resetStandardHeight` not followed by `setStandardHeight` occurred, if it is followed by `dock` but not by `undock` or `setStandardHeight`,if it is followed by `uppermostReached` but not by `undock` or `setStandardHeight`, if it is followed by `motorDown` but not by `setStandardHeight`, `uppermostReached` or `emergencyMode`, if `pressUp` or `pressDown` occurred, then `motorUp` must eventually occur.

Note that in the translation to *mu-calculus* formulas, living properties that include the word "eventually" are translated in the (general) form:

$$[...]\mu X.(< T > T \wedge [\neg a]X)$$

To indicate that in a finite number of steps action a will happen. The first atom, $< T > T$, is omitted in the specification of the property because the requirement of *Deadlock freeness* holds.

4 Architecture

We describe here the architecture of the control system that we have designed. As specified in the assignment it is comprised of three separate controllers: one for inputs from the console (named `Console`), one for inputs from the sensors (named `Sensors`), and one for outputs to the motors and the brakes (named

`Output`). Furthermore, we included another controller to orchestrate the correct functioning of the others: one central processor (named `Processor`). As general convention in the specification we use the prefix `r_` to indicate receive actions and `s_` for send actions.

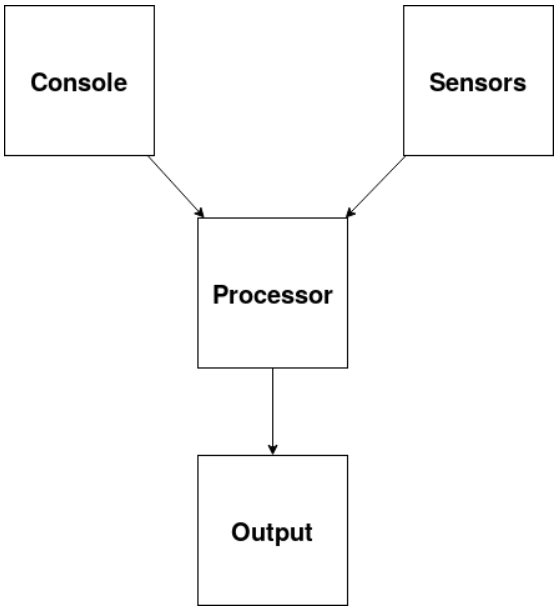

Figure 1: Overall system architecture

4.1 Console

The console has 6 different buttons: `Up`, `Down`, `Stop`, `Reset`, `Undock` and `Resume`, as requested. They can be both pressed and subsequently released or they can be tapped (which is an atomic action). When a button is pressed, released or tapped the corresponding send action is sent to the processor, which will compute the input.

4.2 Output

The output controller accepts inputs from the processor through a secure channel and simply executes the requests. It is composed of five sub-components: the horizontal brake, the vertical brake, the horizontal motor, the vertical motor and the undock controller. The brakes can be released or applied, while the motors can move in two directions (up/down or left/right) or they can be turned off. The undock controller can be used to undock the MPSP. There is no "intelligence" in this output module, as all of the complexities to satisfy the requirements are computed in the processor. For example, if the processor sends

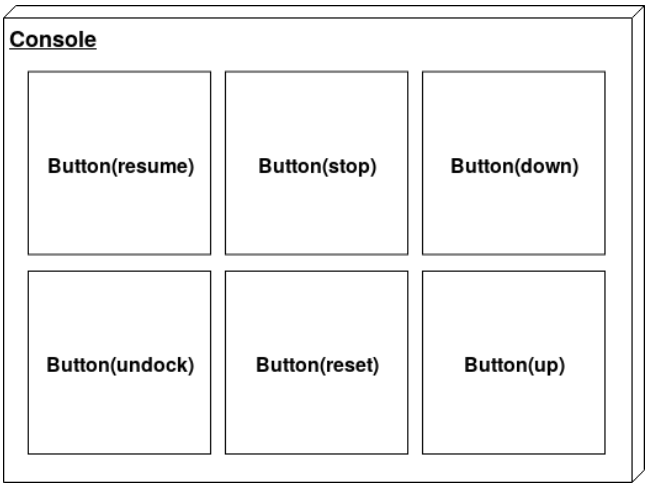

Figure 2: Console controller

the request to apply the vertical brake and to move up at the same time, the motor will overheat. Of course this won't happen, as the processor is designed to satisfy the requirements.

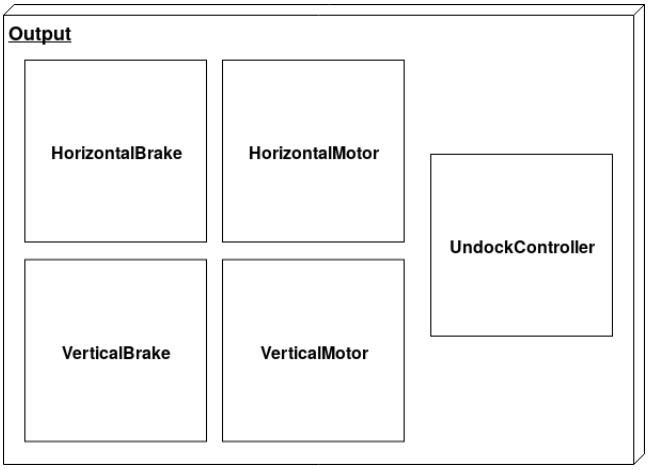

Figure 3: Output controller

4.3 Sensors

The sensors accept input from the physical world and communicate them to the processor. This module is composed of three sub-modules: the horizontal position sensors, the vertical position sensors and the dock sensor. The first

can send the input rightmost or leftmost reached, while the second can send uppermost, lowermost or standard height reached and the last can send dock. It is important to note that this module does not recognize if the movement is done mechanically or manually, and for this reason the processor will have to infer if a position message is due to motorized movement or not.

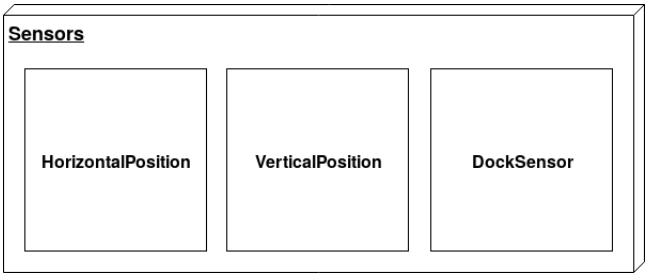

Figure 4: Sensors controller

4.4 Processor

This controller keeps track of the current state of the system through one state variable. It accepts inputs from the console and the sensors and, after processing them it sends the proper output to the brakes, the motors or the undock controller. It models the behavior requested in the assignment, and takes care of the complexities related to satisfying the requirements (e.g. not applying the vertical brake while the vertical motor is moving).

4.5 Behavior

The system is designed to work as described in the assignment, but sometimes we had the freedom make our design decisions and not repeat the whole behavior of the system as specified in the assignment. In this section, we describe the complex behavior of the MPSP and the autonomous decisions we made during the design of our system.

4.5.1 Initial state

As it was not specified, we have decided that the initial state of the MPSP is:

- Undocked
- Uncalibrated
- Vertical motor off
- Horizontal motor off
- Vertical brake applied

- Horizontal brake applied
- In normal mode
- At the lowermost position

This represents the scenario of a new MPSP delivered to the hospital from the producer.

4.5.2 Emergency mode

The emergency mode can be activated only when the MPSP is docked. If the emergency mode is triggered, we ensure that the horizontal brake is not applied to allow medical staff to manually remove locate the patient outside of the scanner. We thought that it is easier in the case of emergency, e.g. heart attack, to directly relocate the patients into the emergency section without removing them from the MPSP. For this reason, we decided that after triggering the emergency mode, and after reaching the rightmost position, the MPSP gets automatically undocked. If the bed is already at the rightmost position, MPSP gets undocked. In case of the emergency mode, the bed reaches the rightmost position. Then the horizontal brakes is applied to prevent the patient from falling down. Pressing or tapping the resume button puts the MPSP back to normal operating mode. This means that the MPSP can be docked, calibrated, etc. While in emergency mode, no motorized movement can occur.

4.5.3 Movement above the standard height

As it is required that *"While the MPSP is docked and calibrated, the bed cannot be moved above the standard height"*, we decided that if the MPSP gets docked while calibrated and above the standard height, if the button up is pressed or tapped, nothing happens. Pressing the down button is the only actionb that lets the bed move. It is possible to have the bed above the standard height and docked for example calibrating it, undocking it, while undocked pressing up and then docking the MPSP.

5 Implementation

Although it was not requested, we want to point out some implementation details of the mCRL2 specification.

5.1 Representation of the state

While it was possible to store each state variable (e.g. state of the brake, position etc) as a parameter of the processor we decided to use one single variable of type `ProcessorState`, grouping all the variables in one single type. This has the advantage to produce more compact and clear mCRL2 specification. To extract some variables from the state, we have implemented many utility functions, for example `isAboveStandardHeight` or `isMovingUp`.

5.2 Representation of the position

The sensors can only inform the processor about certain points in the space (e.g. the lowermost position). The MPSP deduce the actual position also from the messages sent to the motors. For example, if one action `lowermostReached` occurred and then one `motorDown` is sent to the motor, then the processor deduces that the position is somewhere between the uppermost and the lowermost position.

We don't need two separate variables for the vertical and the horizontal position, because if the MPSP has moved to the left, then it will be at the standard height. For this reason we have eight possible values for the position: `leftmost`, `horizontalBetween`, `lowermost`, `uppermost`, `verticalBetween`, `standardHeight`, `aboveStandardHeight` and `belowStandardHeight`. Also, we wanted to keep track if the MPSP is above or under the standard height when it is calibrated; therefore, we included the last two values. The processor is able to determine whether it is correct to use either `aboveStandardHeight` or `verticalBetween` when moving up from the lowermost position, because it has the information if the bed is calibrated or not.

5.3 Calibration

We had trouble dealing with the fact that the MPSP could be calibrated at the uppermost position. In fact, if the MPSP got subsequently undocked and the button reset was pressed at the standard height, there was no correct way to update the position in the state: the `uppermost` was not correct if the MPSP was calibrated under the uppermost position, and `horizontalbetween` was not correct if the MPSP was calibrated at the uppermost position. We had the same problem for the lowermost position. For this reason we decided to include in the calibration variable the fact that the MPSP can be calibrated at the uppermost or at the lowermost position. The calibration is not a boolean value, but it can have four possible values: `calibratedInBetween`, `calibratedUppermost`, `calibratedLowermost` and `uncalibrated`. In this way the processor is capable to correctly track the position of the MPSP.