(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 617 111 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.03.2020 Bulletin 2020/10

(51) Int Cl.:
B66B 1/34 *(2006.01)*

(21) Application number: **18191709.7**

(22) Date of filing: **30.08.2018**

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
KH MA MD TN

(71) Applicant: **Inventio AG**
6052 Hergiswil (CH)

(72) Inventors:
- **ARNOLD, Daniel**
  6462 Seedorf (CH)
- **FREY, David**
  6004 Luzern (CH)
- **SAGER, Matthias**
  6403 Küssnacht am Rigi (CH)
- **BÜRGISSER, Patrick**
  6344 Meierskappel (CH)

(54) DOWNLOADING UPDATES FOR CONTROL SOFTWARE ON PASSENGER TRANSPORT SYSTEMS

(57) The present invention relates to an update module (100), a control system and a passenger transport system with an update module (100) as well as to a method for updating control software for a passenger transport system (EI). At least two instances (I_1, I_2) of a control application (A) are provided in parallel, wherein at any time point one of the at least two instances (I_1, I_2) is active and is used for control of the passenger transport system (EL) while the other one of the at least two instances (I_1, I_2) is passive and is not used for control but instead is used for being updated. The method comprises the following steps:
- Providing a software update (P);
- Updating the passive instance (pI) of the control application (A);
- Issuing a trigger signal (t) by the passive instance (pI) if the update is completed;
- Automatically switching from the active instance (aI) to the passive instance (pI) for taking over control of the passenger transport system (EL), in case the trigger signal (t) is issued.

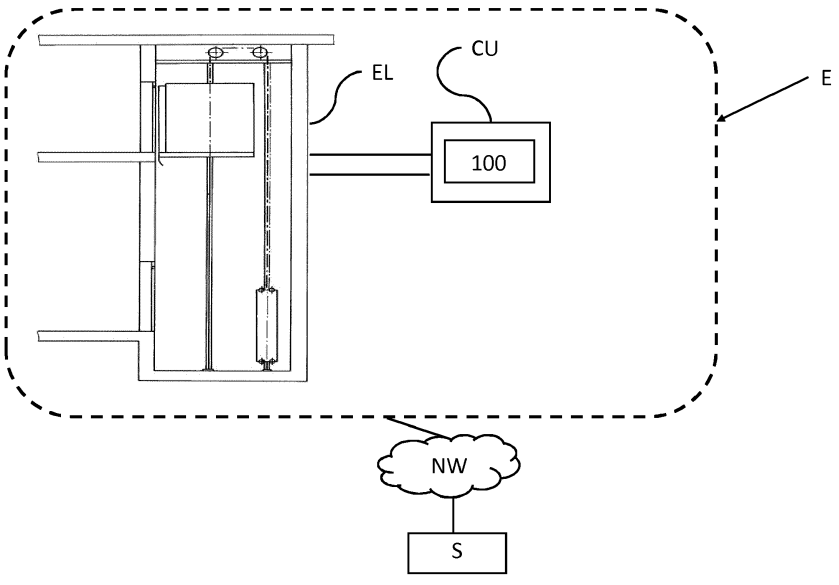

Fig. 1

EP 3 617 111 A1

Description

[0001] The present invention relates to updating control software for passenger transport systems, like elevator and/or escalator systems.

[0002] In known elevator and/or escalator systems, updating software requires a manual interaction with a service technician. In particular, a service technician needs to take the respective system out of service during the time period which is necessary to complete an update of the control software and to reach a fully operational state again.

[0003] WO 2016/180484 discloses a method to allow a service technician to update safety related software in a passenger transport system. This document shows to load an update in a local download memory and to only execute the update in case a service technician has activated a local switch in order to make sure that a service person is attendant at the passenger transport system. Executing the update requires rebooting the local software system as a whole or at least in parts.

[0004] Thus, in known systems there is a drawback in that the transport system is not available to the customer during the update time period. Another disadvantage is that a service technician is required for initiating a local software update.

[0005] Therefore, it is an object of the present invention to improve availability of the passenger transport system also during time periods in which updates of control software are executed. Further, the update process should be automated without impeding safety of the system so that it is not required to initiate any manual steps by a service technician.

[0006] According to a first aspect, this object is achieved by a method for updating control software for a passenger transport system, like elevators and escalators. At least two instances of a control application are provided in parallel, wherein at any specific point in time one of the at least two instances acts as master and is active (with respect to control) and is thus used for control of the passenger transport system while the other one of the at least two instances is passive and is not used for control (and thus is inactive with respect to the control) but instead is used for being updated. The method comprises:

- Providing, in particular downloading a software update (for example via an external interface for update of the control software);
- Updating the passive instance of the control application (i.e. the instance which is currently not used for control of the passenger transport system), so that the passenger transport system may remain in service;
- Issuing a trigger signal by the passive instance if the update is completed and thus representing that the control application instance is in an operational state again;
- Automatically switching from the active instance, i.e. the instance, being currently actively used to control the transport system, to the passive instance for the control of the passenger transport system, if a trigger signal is issued and/or correctly received.

[0007] The present invention provides an option to flexibly upload software patches for control of passenger transport systems, like elevators and escalators with continuous operability and without impairing safety. The patches may be deployed at any point of time and the transport system is available for customers also in this time period. Due to the automatic execution, it is not necessary to provide any interaction with a service technician. This saves costs and improves availability.

[0008] According to a preferred embodiment, there is an automatic continuous switching between the at least two instances, being used for control of the passenger transport system.

[0009] In a preferred embodiment, each of the patches is provided on every instance of the control application. This means that after the inactive instance is used to load the patch, while the active instance is used to control the transport system, a mirroring is executed. This means that the formerly active and now inactive instance is also provided with the same (mirrored) patch version as the other instance. This has the technical effect that redundant control may be provided. In case one instance becomes corrupted, the other one may immediately take over control, because the latest update versions are always loaded on both of the two instances. With this, availability of the system may be assured and improved. In this embodiment, in particular after switching from the active instance to the passive instance of the control application, updating the formerly active instance is automatically triggered, in particular with the same update which is currently used on the formerly passive instance. This improves security as a redundant instance of the control application may be provided and instantly used.

[0010] In an alternative but also preferred embodiment, the method may also be operated in a single-load mode. In the single-load mode, the patches are only loaded on one instance of the application. Thus, the subsequent patch numbers or versions are loaded in an alternating manner on the first and second instance, respectively. This improves efficiency and saves processing resources for redundant loading of patches on both of the instances.

[0011] According to another embodiment, updating the passive instance comprises to reset and start up from the provided software update and to synchronize with the other instance (previously being the active instance and acting as a master instance) concerning I/O-states, service transitions, parameters settings and/or other configurations.

[0012] According to another preferred embodiment, the method steps (providing, updating, issuing and/or switching) are executed iteratively for successive up-

dates or patches.

[0013] According to another preferred embodiment, updating control software is executed as a hidden service and fully autonomously without any user interaction. This has the effect that not service operator needs to be provided in the field to deploy a new software update.

[0014] According to another preferred embodiment, the update is usable and will become operational without rebooting the software system. This improves availability of the elevator or transport system.

[0015] According to another preferred embodiment, updating control software is executed without interruption of the passenger transport system. This improves availability of the elevator or transport system, too.

[0016] In another aspect the invention relates to an update module for updating control software for a passenger transport system, wherein at least two instances of a control application are provided in parallel, wherein at any point in time one of the at least two instances is active and is used for control of the passenger transport system while the other one of the at least two instances is passive and is not used for control but instead is used for being updated. The update module comprises:

- A patch interface (e.g. to a central server) for providing a software update;
- A central processing unit which is adapted to update the passive instance of the control application;
- A sensor element, which is adapted to issue a trigger signal if the update on the passive instance is completed; the sensor element may be provided in the central processing unit;
- Initiating the central processing unit to automatically switch from the active instance to the passive instance for taking over control of the passenger transport system, in case the trigger signal is issued.

[0017] According to a preferred embodiment, the update module is adapted to execute the method as described above.

[0018] In another aspect the invention relates to a passenger transport system with an update module as mentioned above.

[0019] In another aspect the invention relates to an update system for a passenger transport system with:

- A central server, providing updates;
- A set of passenger transport systems, wherein all or a selected group of passenger transport systems comprises an update module, as mentioned above and
- A network connection between each of the passenger transport systems and the central server.

[0020] The system may additionally comprise a field control device which is responsible for controlling and updating a group of passenger transport systems. This has the advantage of being able to uniformly control a set of passenger transport systems for example in one building. Then, the server may provide a patch to be downloaded on the field control device, acting as intermediary node, to control the local update modules of the passenger transport systems.

[0021] Further, the update module is adapted to perform all steps as claimed in connection with the corresponding method which is to be performed in the corresponding module. The module is preferably provided as hardware module and/or as software module. The module may be embedded in a computing environment with a processing unit or a microprocessor and memory instances and a bus system may serve as interface with external devices.

[0022] With other words, a control software platform can run two instances of a software control application in parallel. One instance is running as operating master, a second instance is running passively. Due to this architecture, each of the instances "knows" its state (being active or passive). Both of the instances notice the presence of a new patch or update. The passive one is adapted to automatically initiate a self-reset for the purpose of being updated.

[0023] When a new software patch/version is downloaded into the local memory, the update process executes as follows:

- Both applications notice the presence of a new software patch. As explained above, the formerly passive instance independently resets and starts up from new software (new config reset only allowed for and executed by passive instance) and synchronizes with the master instance concerning I/O states, service transitions, parameter settings etc.

- Once the updated instance reaches fully operational state ("Normal" operation) it takes over master operation (internally driven by who has the newer software version number).

- The former master instance immediately becomes passive and updates afterwards in the same manner, eventually starting up to full operational state. However, accidental takeover of operation is prevented by the information of both instances having the same software versions.

[0024] This automatic update process may be controlled and/or executed by the platform or by the respective operating system. In an exemplary preferred embodiment, RTEMS- and/or Linux-based control systems are used. RMTEMS is an abbreviation for Real-Time Executive for Multiprocessor Systems and is a is a real-time operating system (RTOS) designed for embedded systems. In case of a Linux-system it is possible to update the base system. In case of RTEMS usually only the application will be updated.

[0025] As an advantage, no additional or special sys-

tem resources need to be provided. It is only necessary to provide sufficient memory and CPU resources to run two full blown instances of controller software applications.

[0026] The main benefit of the proposed solution is a software update procedure that does not decrease the elevators availability. This in turn, increases customer satisfaction and reduces maintenance costs. By having the elevator(s) in continuous operation and executing invisible updates, the elevator system remains available to the outside without any interruption. Silent software update creates optimal customer benefit due to seamless service.

[0027] Moreover, the maintenance staff becomes much more flexible in their choice of the time slot to do the update. With the increased connectivity built in modern elevator systems, some updates could be initiated remotely. The time required to stay at the installation is expected to decrease. All these factors add to saving time and money for both the customer and the maintenance institution.

[0028] In an emerging environment of interconnected devices that goes along with an increased importance of IT security, it is crucial to be able to react with minimum delay to revealed vulnerabilities. With a remotely controlled seamless software update, corrections and mitigations can be rolled out in the field more quickly and with less impact on the customer.

[0029] This is a competitive advantage itself which contributes to both customer security and availability.

[0030] In the following, a short definition of terms used within this application is given.

[0031] The control application serves to control the passenger transport system. The control application is a computer program which is in data exchange with actors and/or electronically controllable modules of the transport system. The control application is executed on a software platform and may be controlled by an operating system. The control application and/or the update module are preferably provided as embedded system. Generally, the properties of an embedded system interacting with the electrical and mechanical parts of the passenger transport system are low power consumption, small size, robust and rugged operating ranges, and low per-unit cost. Typically, the control application has no user interface. I a preferred embodiment, the control application may be adapted to notice, store and provide its state and in particular the update version being currently used. The state may be provided as a message to a user interface and/or to the server.

[0032] An instance of the application is to be construed as the creation of a realized application execution. Each time a program runs, it is an instance of that program. Thus, the application is executed twice, in a passive and in an active instance, wherein both of the instances are used for different purposes: one is used for updating while the other is used for controlling the elevator system. In an embodiment, each program instance will generally load in a separate runtime environment that contains the program's details, states etc. that are to be managed during program execution. If another, second instance of this program is executed, the second instance runs alongside the first instance, so each can be quit and managed and controlled independently.

[0033] The passenger transport system may be an elevator, escalator, a moving walkway or any other kind of people conveyer system.

[0034] The term software update is construed to be a synonym to the term patch. The patch may be a binary file or an application program code or a part thereof (e.g. only comprising the amendments to the former version). The software update is initiated by the server only. Usually, there is no data sent form the control application to the server, informing the server which kind of update should be loaded. The server is the sole instance, which decides on the updated to be used. Therefore, the patch interface may be adapted to be one directional, namely for receiving the patch.

[0035] Providing the software update means to download it via the patch interface and/or may comprise to unpack (if the update is provided in a packed format), to decrypt (if the update is provided in encrypted form) and/or to store the update locally. Moreover, the compatibility with the control application may be checked automatically. In case of non-compliance an error signal may be issued and reported to the server.

[0036] The trigger signal is a digital signal, which may be provided as flag or binary electronic signal in order to reflect completion of an update procedure.

[0037] Switching refers to an electronic action. Switching may be done by the software platform or operating system. Switching refers to define the master (instance) which is responsible for control of the elevator.

[0038] The network for server interaction may be based on a wireless data transmission. In a first embodiment, an encrypted protocol, for example SSL/TLS, may be used. In a second embodiment, an unencrypted protocol may be used, like HTTP, IMAP or POP3.

[0039] The update procedure may be provided as background service without any manual support or interaction. Updates are loaded as a hidden service without interruption of the elevator system. The update procedure may be based on a script.

[0040] The sensor element is a functional element and serves to detect the completion of the update procedure. The sensor element may be part of the control application. The sensor element also be part of the central processing unit and/or the software platform. The sensor element may be implemented as a software function and the result may binary (flag, indicating completion or still-running process).

[0041] The method may be implemented as a computer program. Thus, the invention is furthermore embodied in a computer program loadable into a processing unit of a control unit for a passenger transport system, the computer program comprising code adapted to perform the

steps of a method for updating control software as described above when processed by the processing unit. The computer program may be stored in a computer readable memory.

Brief Description of the Drawings

[0042] In the following, the invention will further be described with reference to exemplary embodiments illustrated in the figures, in which:

Fig. 1 schematically shows an infrastructure of an update module in a context of a passenger transport system according to a preferred embodiment of the invention;

Fig. 2 is a block diagram of an update module according to a preferred embodiment of the invention;

Fig. 3 shows an interaction diagram with interactions and message transfer between the respective computing instances according to a preferred embodiment of the invention;

Fig. 4 represents a flow chart of a method according to a preferred embodiment of the invention.

Detailed Description

[0043] In the following description, for purposes of explanation and not limitation, specific details are set forth, such as particular passenger transport systems or computing environments and communication standards etc., in order to provide a thorough understanding of the current invention. It will be apparent to one skilled in the art that the current invention may be practiced in other embodiments that depart from these specific details. For example, the skilled person will appreciate that the current invention may be practiced with any system architecture, for example comprising further field devices. As another example, the invention may also be implemented in any local device, having the respective control interfaces for controlling the passenger transport system, like a mobile electronic device. Further, the invention may be practiced with any other network or bus system.

[0044] In summary, it is proposed to provide a technique for providing software patches for control software for passenger transport systems, like elevators and/or escalators, as a hidden service without interruption of the transport system and without requiring any manual interaction.

[0045] **Fig. 1** shows in a schematic overview an elevator system E with an elevator EL which is in data connection with an update module 100. In a preferred embodiment, the update module 100 is integrated into the elevator E and the data connection is a wired connection, e.g. bus system. Alternatively, it may be a wireless connection and the update module 100 may be provided as a separate instance, for example locally at the site of the elevator system E but apart from the elevator EL. Typically, the update module 100 is embedded in an electronic device which acts as control unit CU and comprises a processing entity (microcontroller, processor) for data processing. The update module 100 serves for updating control software for the elevator EL or any other passenger transport system. The control unit CU is adapted to controlling the elevator EL. Accordingly, control signals are transferred between the control unit CU and respective actors in the elevator EL. The control unit CU may be implemented in hardware and a control application A may be run thereon for executing control functions. The control application A is required to be updated. The update or patch P is provided in the form of digital code from a central server S, which is in data connection with the field system to control the elevator E. The data connection is a wireless connection (over the air - OTA) or may be internet-based and may be based e.g. on a http protocol. Other protocols may also be used, like for example TCO/IP or others like CANOpen, DCP or proprietary protocols.

[0046] **Fig. 2** shows the update module 100 in more detail. Generally, the update module 100 receives the software patch P for updating the control application A via a patch interface PI. The patch P may be a binary file. After receiving the patch P on the update module 100, it may be stored in a memory MEM. A computing runtime environment or a software update platform with a central processing unit CPU is provided for running at least two instances I of the control application A in parallel. An operating system will control the at least two instances so that at any point in time only one of the at least two instances is used for active control. Control is executed by transferring control signals via a control interface CI to the elevator EL. The central processing unit CPU additionally comprises a sensor element 101, which is adapted for monitoring the update load process. In case the update on the inactive instance is completed, a trigger signal t is issued in order to inform that the respective instance to take over control is to be switched.

[0047] In particular, the instance which is active and is (actively) used for control of the elevator EL is defined as active instance aI of the control application A. The other one of the at least two instances, which is passive and is not used for control but instead is used for being updated, is defined as passive instance pI.

[0048] Thus, in a preferred embodiment, there is exactly one instance active (the active instance aI) while the other one is passive. The passive instance pI is used for being updated. The roles of the instances (active/passive) continuously alternate during the course of operation with several successive updates. For example, for a first patch P_1 a first instance aI_1 is active, whereas for a second patch P_2 a second instance AI_2 is active, which was the passive one during executing the first patch P_1. For a subsequent third patch P_3 the first instance aI_1 will become active again and so on.

[0049] **Fig. 3** shows a sequence diagram for explaining the signal and message transfer between the respective computing entities.

[0050] The server S issues and provides the patches P. For this purpose, the server S may rely on a pre-defined update scheme, so that e.g. updates are provided at pre-defined time intervals or only for a set of elevator systems EL or in a configurable manner. The server interacts with the update module 100. The update module, in particular, is equipped with the memory MEM, the two instances I_1, I_2 of the control applications A and a computing environment with an operating system, in the figures depicted with CPU. Other entities may also be part of the update module 100, but are not further mentioned here, for the sake of clarity. The elevator EL is controlled by one of the two instances I_1, I_2, and in particular with the very instance currently being active.

[0051] In the example shown in Fig. 3, a first patch P1 is provided on the server S and is transferred to the memory MEM. As it is the second instance I_2 which is currently actively controlling the elevator EL, the first patch P_1 is deployed on the first instance I_1, being currently passive. After the loading of the first patch P_1 is completed, the first instance I_1 issues a trigger signal t in order to inform that the update procedure on the first instance I_1 is fully completed. This is used by the operating system or by another control instance in the update module 100 to switch between the two instances, so that now, the first instance I_1 becomes active and the second instance I_2 becomes inactive. Control of the elevator EL is taken over by the first instance I_1.

[0052] If a second patch P_2 is to be processed, it will be provided locally on the memory MEM. The second instance I_2 will load this second update P_2, because it is the one which is currently inactive. After the download and update procedure is completed and the second instance I_2 is in fully operational state again for taking over control, the trigger signal t will be issued in order to trigger a switch again between the two instances I_1, I_2, so that the second instance I_2 will become active and take over control of the elevator system EL.

[0053] As can be seen in Fig. 3 on the right side in the context of the elevator system EL, during the first time segment, the elevator EL is controlled by the second instance I_2, based on the former patch, depicted in Fig. 3 with P_0. After having updated the local system with the first patch P_1, the elevator EL is controlled by the first instance I_1, based on the first patch P_1. After having updated the local system with the second patch P_2, the elevator EL is controlled by the second instance I_2, based on the second patch P_2 and so on. It can be seen that the sequences of patches are used to control the elevator EL successively.

[0054] In the embodiment shown in Fig. 3, the two instances I_1, I_2 are only loaded with every second patch (n -> n+2 -> n+4 ->..., wherein n refers to the patch version or number). Thus, the second instance I_2 for example, is provided with patch P_0 and P_2, whereas the first instance I_1 is provided with the first patch P_1 and a third patch P_3 (not explicitly shown in Fig. 3.) and so on and so forth. This has the advantage that additional updates on the other instances may be skipped. This saves local processing resources.

[0055] In another preferred embodiment, it is assured that each of the patches is provided and loaded on all of the instances I (i.e. n -> n+1 -> n+2 -> n+3 -> n+4 ->..). In this embodiment, for example, the second instance I_2 will also be provided with the first patch P_1. This loading will be executed in the phase during which the second instance is in the inactive state. This embodiment has the advantage that security may be improved, as both instances I do process the same patch version. Thus, in case of a failure, e.g. during a breakdown or malfunction of one instance, the respective other instance may take over control and may act as redundancy fall back control instance.

[0056] **Fig. 4** is a flow chart of the update method according to a preferred embodiment. After Start, in step S1 a (new) patch P is loaded on the local update module 100. In step S2 the received update patch P is stored in the local memory MEM. In step S3, the inactive instance is selected and used for processing and loading the patch P. The update process is monitored and after end of the update procedure, the inactive instance issues a trigger signal t in step S4. The trigger signal t may e.g.be provided to the operating system in order to trigger a switch between the instances in step S5, so that the active instance aI becomes inactive and in turn the inactive one will become the active one. Step 6 should reflect a control of the elevator system EL by the active instance. After this, the method may end or may continue (if no further patches are received, the state remains stable) or may be re-iterated by branching back to step 1, if new patches are to be received.

[0057] In state of the art systems, for a controller software update the corresponding elevator EL must be taken out of service and marked as being under maintenance. The consequence is a service outage of usually several hours. With the seamless and hidden software update according to the proposed solution, the elevator remains fully in service and is therefore capable of serving calls. The user will not notice any difference.

[0058] The elevator system continuously monitors states of input devices. Since operability remains unchanged during the software update this process is not interrupted. Therefore, the system is still able to recognize activation of critical inputs for e.g. emergency modes.

[0059] In case of an activation of such an emergency input the update actions will be canceled immediately. For safety reasons, Emergency Operation Modes of any kind (Fire, Hospital, Earthquake etc.) do not allow neither manual nor automatic software updates. Therefore, the invention provides an internal locking mechanism, which will actively prevent update attempts or requests in case an emergency mode has been detected.

[0060] The internal locking mechanism leads to an additional and significant advantage over today's situation. Presently, an elevator taken out of service for update is not available to execute tasks in sudden Emergency Mode. An elevator under silent software update according to this solution - which will be canceled in these situations - is still operable and may support building evacuation etc. This internal locking mechanism is in particular of relevance for elevator environments being critical, e.g. hospitals.

[0061] As a further advantage, the hidden or silent control software update relates to all kind of updates (patches, releases, new versions etc.).

[0062] A preferred embodiment relates to Linux-based elevator control systems. The proposed method may thus be used as Linux "live patch" mechanism. This approach allows complete transparency of operational availability both from internal technical and customer point of view.

[0063] While the instant invention has been described in relation to its preferred embodiments, it is to be understood that this description is for illustrative purposes only. Accordingly, it is intended that the invention be limited only by the scope of the claims appended hereto.

Claims

1. Method for updating control software for a passenger transport system (EL), wherein at least two instances (I_1, I_2) of a control application (A) are provided in parallel, wherein at any time point one of the at least two instances (I_1, I_2) is active and is used for control of the passenger transport system (EL) while the other one of the at least two instances (I_1, I_2) is passive and is not used for control but instead is used for being updated, the method comprises:

   - Providing (S1) a software update (P);
   - Updating (S3) the passive instance (pI) of the control application (A);
   - Issuing (S4) a trigger signal (t) by the passive instance (pI) if the update is completed;
   - Automatically switching (S5) from the active instance (aI) to the passive instance (pI) for taking over control of the passenger transport system (EL), in case the trigger signal (t) is issued.

2. Method according to claim 1, wherein there is an automatic continuous switching between the at least two instances (I_1, I_2), being used for control of the passenger transport system (EL).

3. Method according to any of the preceding claims, wherein updating (S3) the passive instance (pI) comprises to reset and start up from the provided updated software and to synchronize with the active instance (aI), which has acted as a master instance before concerning I/O-states, service transitions, parameters settings and/or other configurations.

4. Method according to any of the preceding claims, wherein the method steps are executed iteratively for successive software updates (P).

5. Method according to any of the preceding claims, wherein updating control software is executed as a hidden service and fully autonomously without any user interaction.

6. Method according to any of the preceding claims, wherein the update is used without rebooting the software system.

7. Method according to any of the preceding claims, wherein updating control software is executed without interruption of the passenger transport system (EL).

8. Method according to any of the preceding claims, wherein after switching from the active instance (aI) to the passive instance (pI) of the control application (A), updating the formerly active instance with the same update which is currently used is triggered automatically.

9. Method according to any of the preceding claims, wherein the method comprises: providing an internal locking mechanism, which actively and automatically prevents update requests in case an emergency mode has been detected.

10. Update module (100) for updating control software for a passenger transport system (EL), wherein at least two instances (I_1, I_2) of a control application (A) are provided in parallel, wherein at any point in time one of the at least two instances (I_1, I_2) is active and is used for control of the passenger transport system (EL) while the other one of the at least two instances (I_1, I_2) is passive and is not used for control but instead is used for being updated, wherein the update module (100) comprises:

    - A patch interface (PI) for providing a software update (P);
    - A central processing unit (CPU) which is adapted to trigger updating the passive instance (pI) of the control application (A);
    - A sensor element (101), which is adapted to issue a trigger signal (t) if the update on the passive instance (pI) is completed;
    - Initiating the central processing unit (CPU) to automatically switch from the active instance (aI) to the passive instance (pI) for taking over control of the passenger transport system (EL), in case the trigger signal (t) is issued.

11. Passenger transport system (EL) with an update module (100) according to the directly preceding claim.

12. Update system for a passenger transport system (EL), comprising:

- A central server (S), providing updates (P);
- At least one passenger transport system (EL) with an update module (100) according to the claim above;
- A network connection between the at least one passenger transport system (EL) and the central server (S).

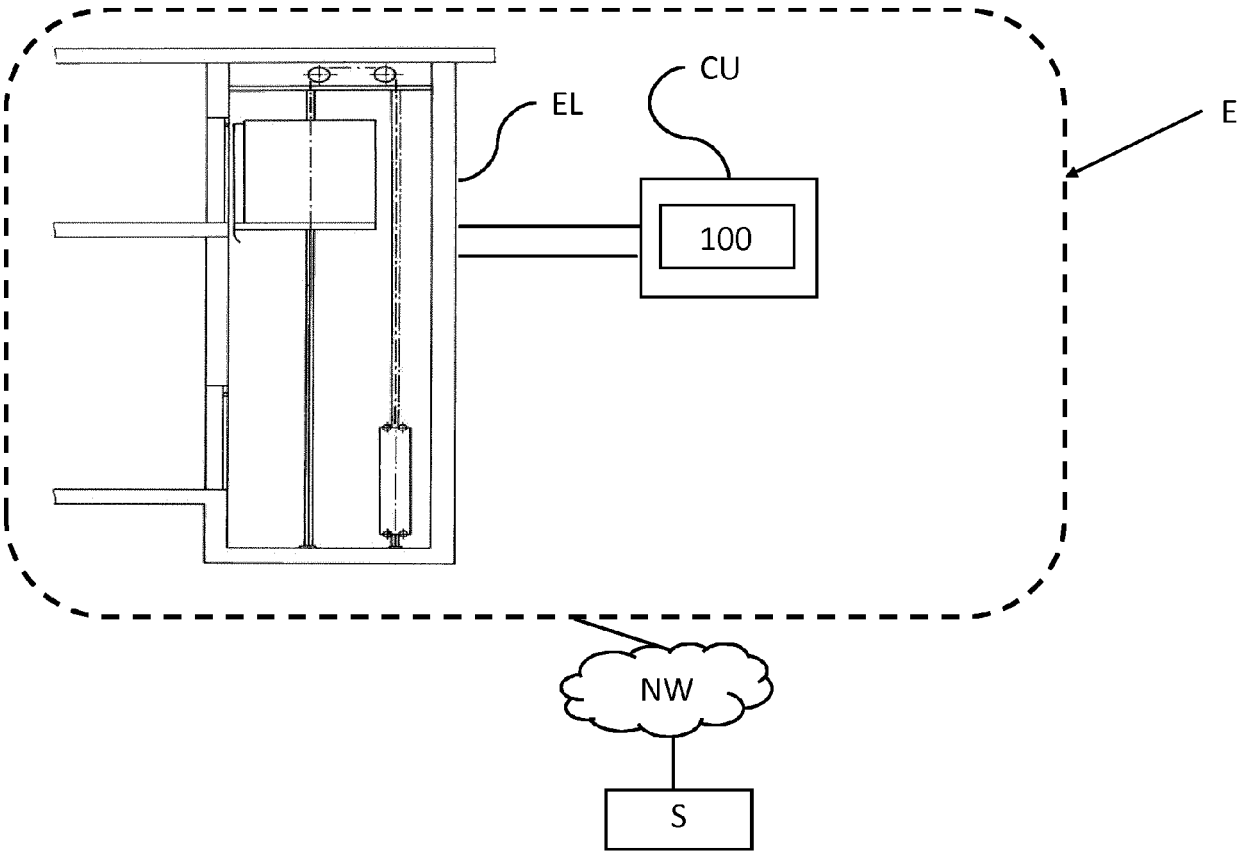

Fig. 1

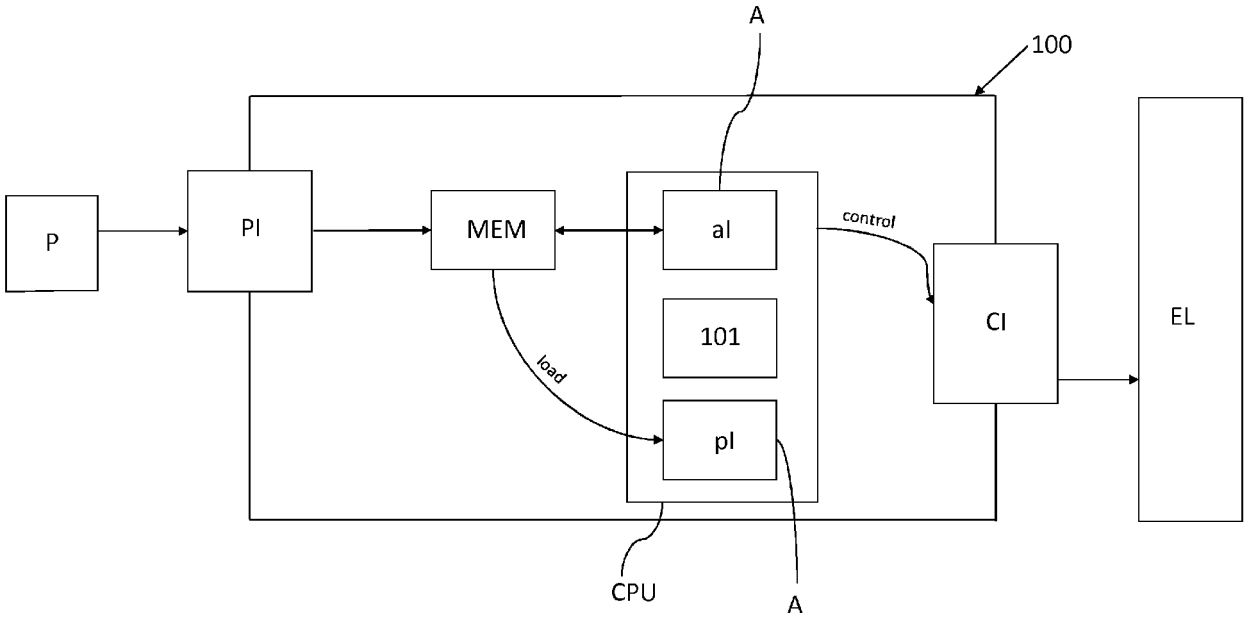

Fig. 2

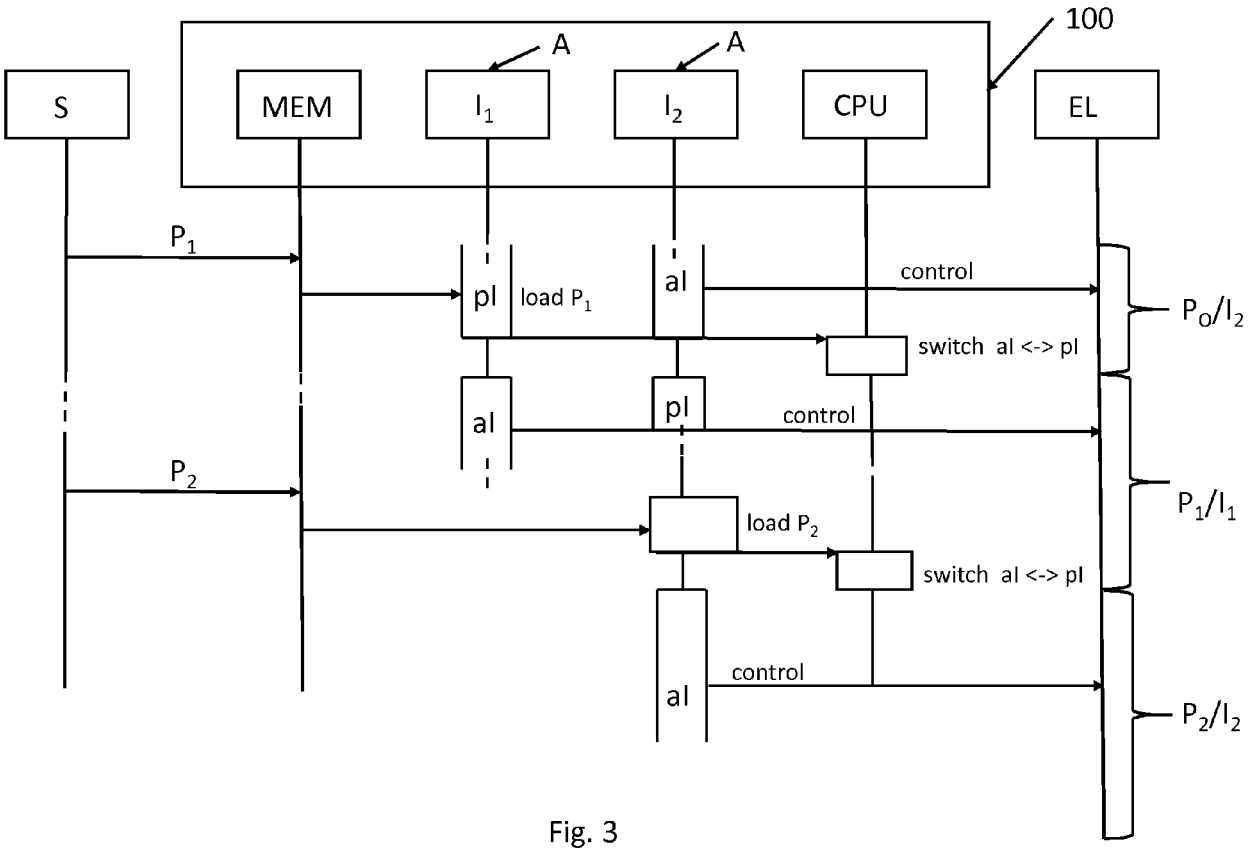
100
A
A
S
MEM
I_1
I_2
CPU
EL
P_1
pl
load P_1
al
control
P_O/I_2
switch al <-> pl
al
pl
control
P_2
load P_2
P_1/I_1
switch al <-> pl
al
control
P_2/I_2

Fig. 3

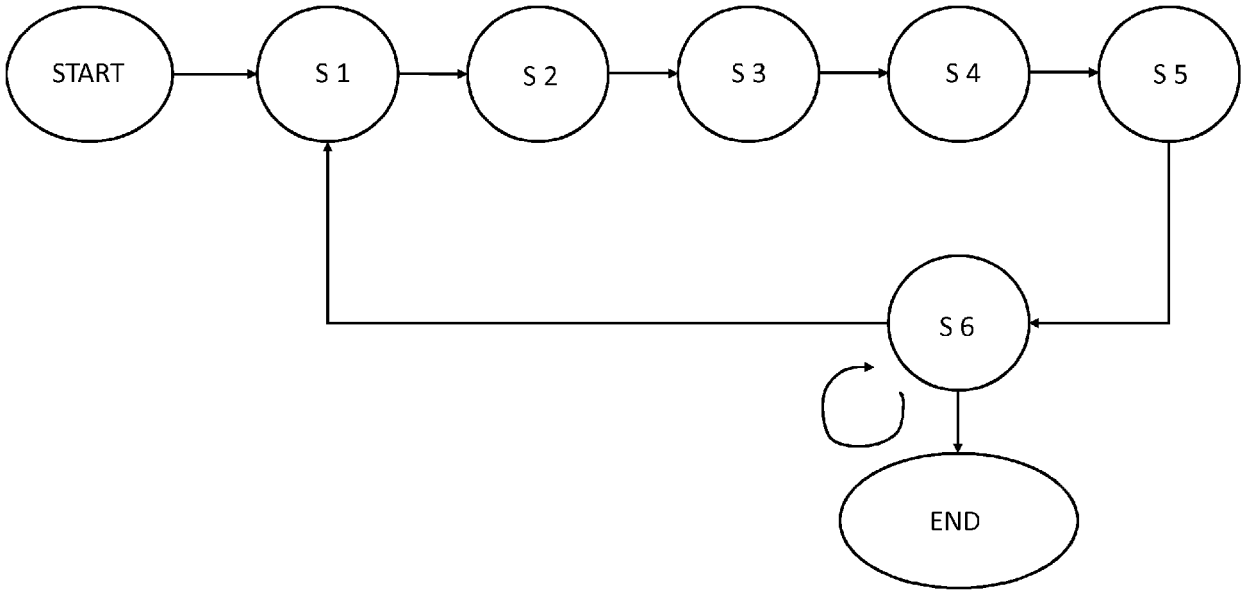

Fig. 4

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 18 19 1709

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	CN 104 973 466 A (GUANGZHOU ROPEN TECHNOLOGY DEV) 14 October 2015 (2015-10-14) * paragraph [0010] - paragraph [0013] * * paragraph [0015] - paragraph [0018] * * paragraph [0031] - paragraph [0035] *	1-12	INV. B66B1/34
A,D	WO 2016/180484 A1 (OTIS ELEVATOR CO [US]; KIRCHHOFF FRANK [DE]) 17 November 2016 (2016-11-17) * the whole document *	1-12	
A	JP 2014 218326 A (MITSUBISHI ELEC BUILDING TECHN) 20 November 2014 (2014-11-20) * paragraph [0009] - paragraph [0014] *	1-12	
A	JP 2002 020052 A (TOSHIBA CORP) 23 January 2002 (2002-01-23) * paragraph [0020] - paragraph [0026] *	1-12	
			TECHNICAL FIELDS SEARCHED (IPC) B66B

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
The Hague	5 March 2019	Szován, Levente

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 18 19 1709

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

05-03-2019

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
CN 104973466 A	14-10-2015	NONE	
WO 2016180484 A1	17-11-2016	CN 107636607 A	26-01-2018
		EP 3295263 A1	21-03-2018
		KR 20180005690 A	16-01-2018
		US 2018157482 A1	07-06-2018
		WO 2016180484 A1	17-11-2016
JP 2014218326 A	20-11-2014	NONE	
JP 2002020052 A	23-01-2002	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- WO 2016180484 A **[0003]**