December 17, 2021

Smart Door Lock

Boas Meier, Niklas Tanner, Lukas Mettler, Tobias Heller
HSLU, Informatik Department, Switzerland
boas.meier@stud.hslu.ch, niklas.tanner@stud.hslu.ch,
lukas.mettler@hslu.ch, tobias.heller@stud.hslu.ch

Code:

https://gitlab.enterpriselab.ch/IoT-I_BA_IOT/iot-i_ba_iot_h21/i-ba-iot-h2101/group01

© September 24, 2021 by Prof. Dr. Angela Nicoara / IoT Systems & Software Research / HSLU Informatik, Switzerland

Abstract

Nowadays more and more devices get connected to the cloud. The connectivity to the cloud not only increases the customer value but also reduces the maintenance costs for the operator. This work proposes a proof-of-concept for a scalable smart door lock solution with a comprehensive architecture based on the state-of-the-art IoT messaging protocol MQTT. The smart door look provides a smart card support for locally unlocking the door over RFID. Furthermore, the App allows remote monitoring, locking, and unlocking over a telegram bot or a web interface. A building equipped with the smart door lock consists of multiple Arduinos – one for each door lock – which are connected over SPI to a RFID Card Reader. Further the building contains one Raspberry Pi to which all the Arduinos are wirelessly connected to, and which acts as a gateway to the cloud. The cloud consists of a REST API, a Key-Value Database, a Frontend and an Elastic Stack for centralized logging and monitoring.

Contents

1. Introduction

"You've forgotten your keys and now you're standing in front of the locked front door. All your roommates are out of the house, and you have no other choice than waiting for the arrival of one of them. What a pity! Especially when you have to change your clothes for the very important job interview."

With the Smart Door Lock you will never have to wait at the doorsteps of your own flat. It allows remote opening of the door. It is not only a convenient way of opening a door with doorbell functionality but also provides additional security features. If someone is strolling around in front of your door a built-in movement sensor will detect this and alert you on your mobile phone. If this person happened to be a burglar who breaks into your house there will be an alarm triggered and optionally the police is being informed.

From a technical standpoint this is achieved with an Arduino Controller which has an NFC Reader, a movement sensor, a door sensor, and various status LEDs connected. Multiple Arduinos in a building are continuously pushing the state, the events, and logs to the gateway over MQTT.

The gateway then analyses the data and performs the correct action. For example, logs are pushed to the logging tool Elastic Stack where they get visualized. In case of state changes these will be forwarded to the backend, where the data is stored in a Key-Value Database and available through a REST-API. If there is an event like someone is ringing the doorbell, the gateway will play a sound and forward the event to the backend. The backend will then forward this action to the mobile phone of the user where there is a message prompted to open the door.

2. Related Work

In this section are other approaches for smart door locks described and analyzed.

Blockchain based Smart Door Lock system

The paper Blockchain bases Smart Door Lock system (Han et al., 2017) proposes a Smart Door Lock which guarantees authentication, non-repudiation and data integrity. Further it proposes an in- and outdoor intrusion detection algorithm based on PIR and ultrasonic sensors, which broadcasts detected intrusions to the blockchain network. The principle of the intrusion detection is more advanced than to one used in this work but in general very similar.

NFC based Smart Door Lock system

The paper of Jose Pacheco et al. proposes a low-cost NFC Door Lock for a smart Home System (Pacheco & Miranda, 2020). Therefore, an Arduino Mega Board in combination with an PN532 NFC module is used. The door is visualized with software written in Unity. The hardware used in the paper is mostly the same as the one used in this work. Whereas the Arduino Mega Board has more power and memory than the Arduino Uno Wifi Rev 2 used in this project. The paper proposes only simple lock and unlock functionality. Therefore, the Arduino Mega seems an overkill. For this use case a smaller microcontroller would be a good fit. For example, an ARM Cortex M0+ which costs less than 1.- CHF[1].

[1] ARM Cortex M0+: https://www.reichelt.com/ch/de/raspberry-pi-rp2040-arm-cortex-m0--rpi-rp2040-p306486.html?PROVID=2808&gclid=Cj0KCQiA15yNBhDTARIsAGnwe0VdK8LDbG4Gkut0p0rDRzPXUgEf4-xtC1fsCFsZxDe1kOfInJ3MFsEaAjpXEALw_wcB (2021-12-01)

3. System Design and Implementation

This section covers the system and software architecture and introduces the individual components with their dependencies.

3.1 System Overview

The presented system is working in a distributed way. On the direct control level for a particular door there is a MCU with sensors and actuators installed. For each building all the door locks are connected to an edge device which is working as a gateway. All the gateways for the different buildings are then connected to a common cloud. In the cloud the data is further aggregated, presented, and stored (see Figure 1).

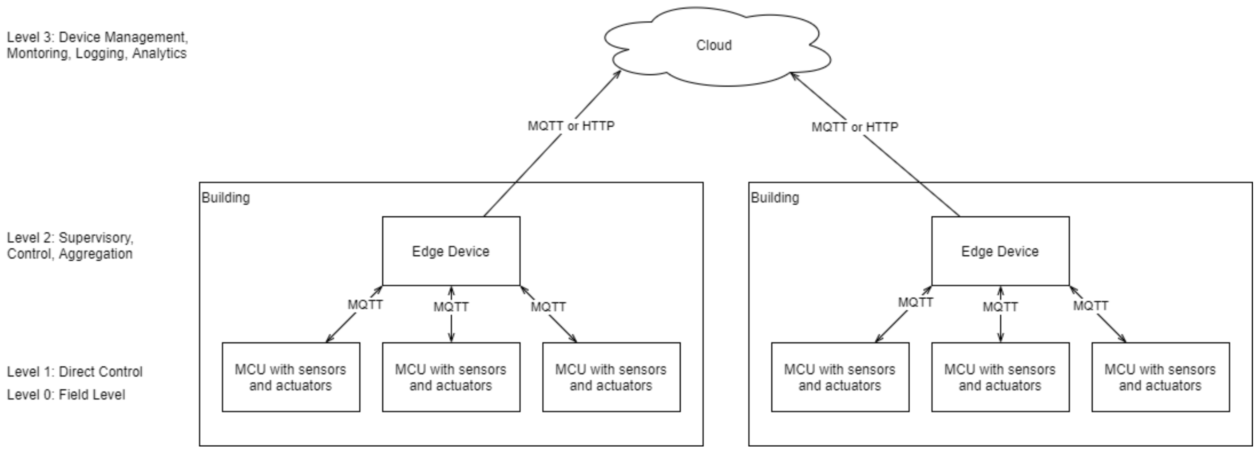

Level 3: Device Management, Montoring, Logging, Analytics
Level 2: Supervisory, Control, Aggregation
Level 1: Direct Control
Level 0: Field Level
Cloud
MQTT or HTTP
MQTT or HTTP
Building
Edge Device
MQTT
MQTT
MQTT
MCU with sensors and actuators
MCU with sensors and actuators
MCU with sensors and actuators
Building
Edge Device
MQTT
MQTT
MQTT
MCU with sensors and actuators
MCU with sensors and actuators
MCU with sensors and actuators

Figure 1: System Overview

3.2 System Architecture

The system architecture is divided in five levels as shown in Figure 2: System Architecture. The following Table 1 lists all the levels and their description.

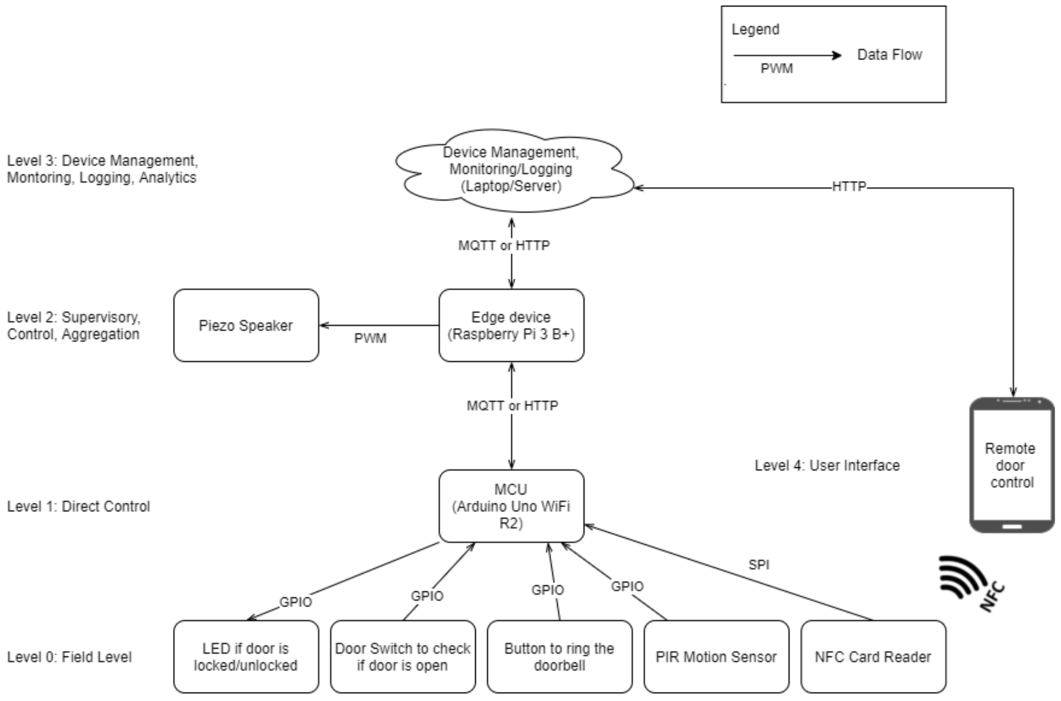

Legend
Data Flow
PWM
Level 3: Device Management, Montoring, Logging, Analytics
Device Management, Monitoring/Logging (Laptop/Server)
HTTP
MQTT or HTTP
Level 2: Supervisory, Control, Aggregation
Piezo Speaker
PWM
Edge device (Raspberry Pi 3 B+)
MQTT or HTTP
Level 4: User Interface
Remote door control
Level 1: Direct Control
MCU (Arduino Uno WiFi R2)
GPIO
GPIO
GPIO
GPIO
SPI
NFC
Level 0: Field Level
LED if door is locked/unlocked
Door Switch to check if door is open
Button to ring the doorbell
PIR Motion Sensor
NFC Card Reader

Figure 2: System Architecture

Level	**Description**
Level 0 - Field	Consists of all the sensors and actuators.
Level 1 - Direct Control	Contains MCUs which deliver the needed real time capabilities for controlling the field level. The sensors and actuators are controlled over GPIO and SPI. SPI is chosen because of its speed, the short distance and because only one card reader is connected.
Level 2 - Supervisory and Aggregation	Acts as a single point of entry to the building. This comes with various security benefits for example there is only one connection to the cloud which needs to be monitored. Controls all the MCUs, aggregates data and is backed with more processing power and storage than level 1. Communication with MCUs is over MQTT because of its small code footprint and low network bandwidth.
Level 3 - Device Management	Manages all the devices installed on the different sites. Centralizes all the logs and monitors the status of the different sites. Offers a REST API for commanding the devices.
Level 4 - User Interface	User interface built upon the REST API.

Table 1: System Architecture Level Overview

3.3 Software Architecture Layers & Modules

On the Arduino there is the single `DoorControllerFirmware` running, which interacts with the sensors and the `EdgeAgent` via MQTT. The `EdgeAgent` communicates with the `DoorLockCloudAPI` via MQTT and stores its own logs and the logs of the Arduino on the disk. These files are read by `Filebeat`, which pushes those logs via HTTP into Elasticsearch. `Elasticsearch` therefore stores the indexed log messages and `Kibana` visualizes the data stored within `Elasticsearch`. The `DoorlockFrontend` is an UI built upon the `DoorlockCloudApi`. The `DoorlockApp` is composed of a Telegram Integration and the possibility to access the `DoorlockFrontend` via the browser.

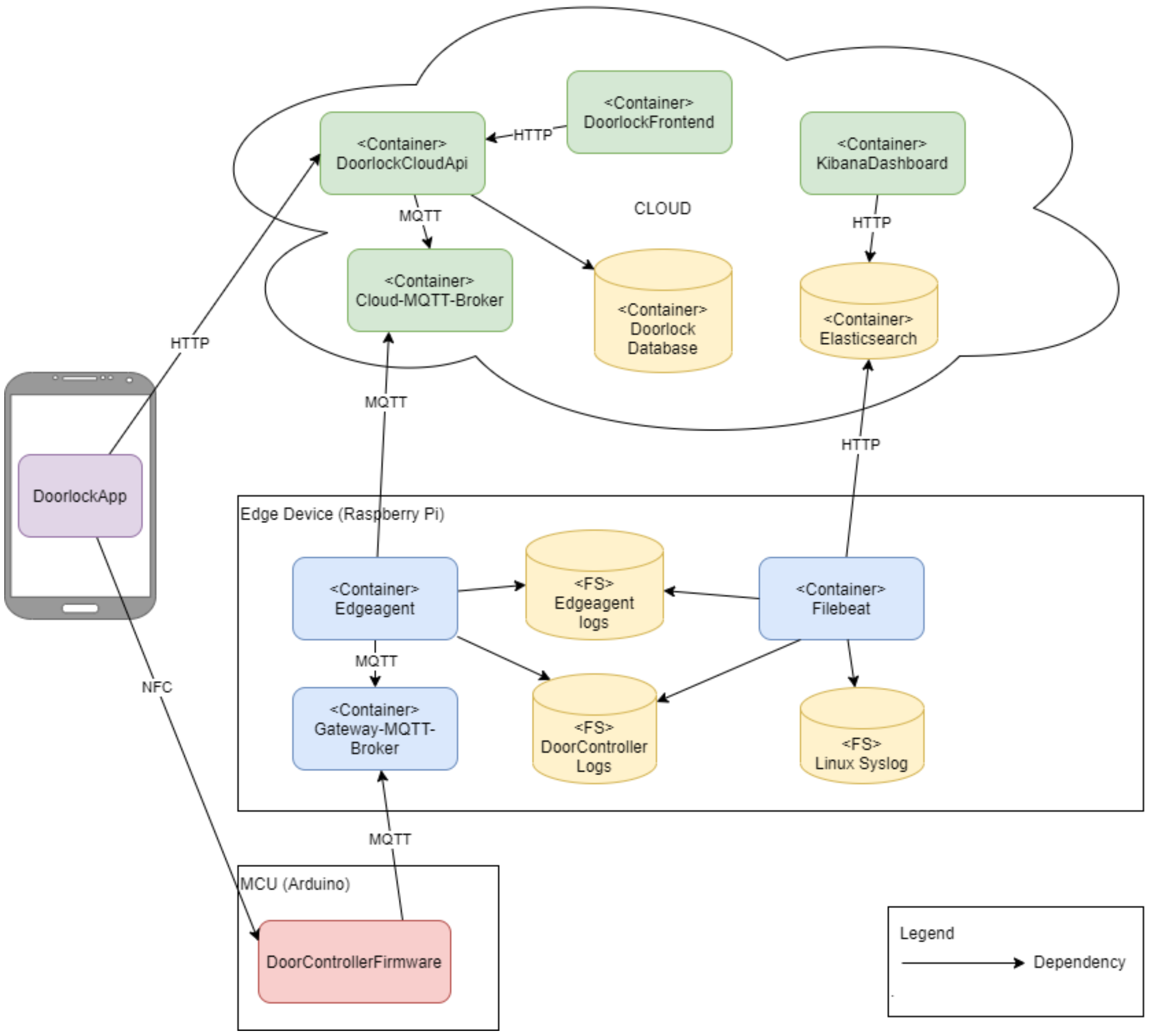


Figure 3: Smart Door Lock Software Architecture

Component	**Description**	**Technology**	**Environment**
DoorControllerFirmware	Handles sensors and actuators connected to the MCU (Arduino) and communicates with Edge Device (Raspberry Pi).	C / C++	MCU (Arduino)
Edgeagent	Implements HTTP Client and MQTT Client to communicate with cloud and MCU (Arduino). Delivers commands from cloud to specific MCU (Arduino).	Python	Edge Device (Raspberry Pi)
Gateway-MQTT-Broker	Is an MQTT-Broker only accessible from within the local network of the site and manages topics of the MCU and the Edge Device.	EMQX	Edge Device (Raspberry Pi)
Filebeat	Log shipper service which aggregates logs and sends them to Elasticsearch database.	Golang	Edge Device (Raspberry Pi)
Cloud-MQTT-Broker	Is accessible from the internet and manages topics of the Edge Device and the Cloud.	EMQX	Cloud (Laptop)
DoorlockCloudApi	MQTT Interface for the Edge Devices to establish a connection to the cloud and the REST-API for customers to control a door lock.	Python / Flask	Cloud (Laptop)
DoorlockDatabase	Stores information about the devices and their events and actions.	Redis	Cloud (Laptop)
ElasticsearchDatabase	Database to store and query log messages.	Java	Cloud (Laptop)
KibanaDashboard	GUI for Elasticsearch.	TypeScript / JavaScript	Cloud (Laptop)
DoorlockFrontend	GUI for interacting with theDoorlockCloudApi.	JS-Framework	Cloud (Laptop)
DoorlockApp	Receives push messages and emulates an NFC card for unlocking the door locally.	Telegram Push	Mobile

Table 2: Smart Door Lock component overview

A detailed description of all the public interfaces of the components can be found in the section Detailed definitions of public interfaces.

3.4 System Implementation / Functional Software Architecture

These following major operations are supported by the system. Depending on the complexity of the operation a different number of components are involved.

#	Operation	Triggered by	Comments
1	Local unlock	Swipe of RFID Card	-
2	Remote unlock	Call to REST API	-
3	Remote lock	Call to REST API	Is a suboperation of #4
4	Doorbell ring and mobile phone unlock	Pressing of the doorbell	-
5	Intrusion detection	Door sensor gets opened without unlock	-
6	Suspicious activity detection	Movement detected by sensor with no subsequent unlock	Decision whether it is a suspicious activity or not is done on the edge-agent
7	Monitoring and Logging	Continuously by each component	-

Table 3: Use Cases of Smart Door Lock

As an example, two of the most used operations are described in the following paragraphs. All the other operations work in a very similar manner.

3.4.1. Local unlock operation

One of the base operations is the local unlock (Figure 4). In this operation a user presents his RFID Card to the card reader. Depending on if this card has access, the door will be unlocked or not and the respective LED (green or red) will light up. 10 seconds after an unlock, the door lock will automatically lock again. This state changes are propagated via the Edge Agent to the Cloud API.

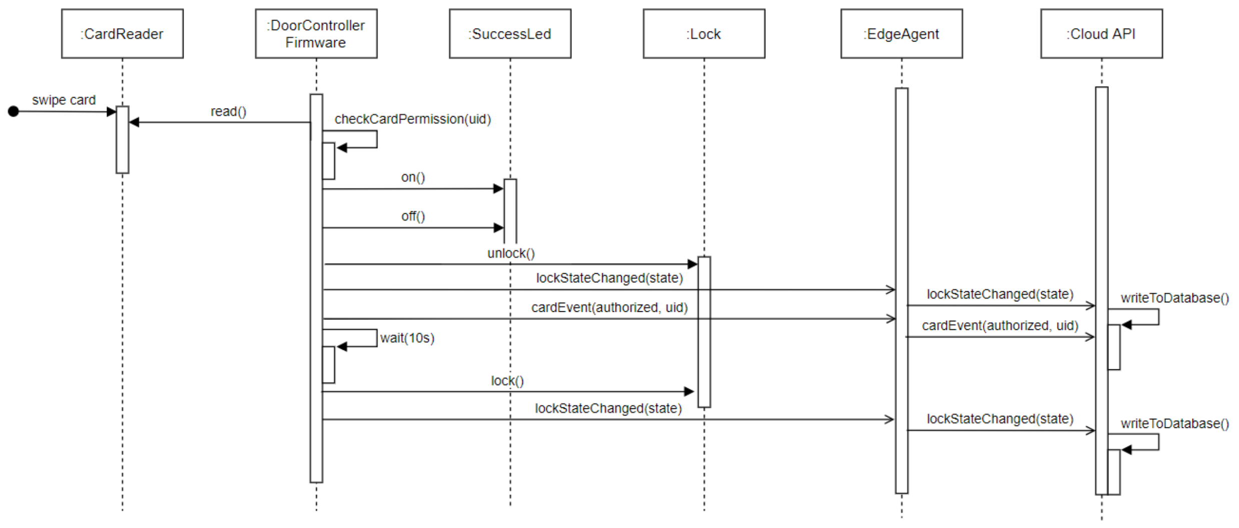


Figure 4: Sequence diagram of local unlock operation

3.4.2. Doorbell ring and mobile phone unlock operation

One of the most important operations is the doorbell function (Figure 5). A visitor can push the doorbell and the speaker will play a ring sound. The event will be forwarded to the resident's mobile phone where he can unlock the door. If the unlock action is triggered this will cause the Arduino to unlock the door.

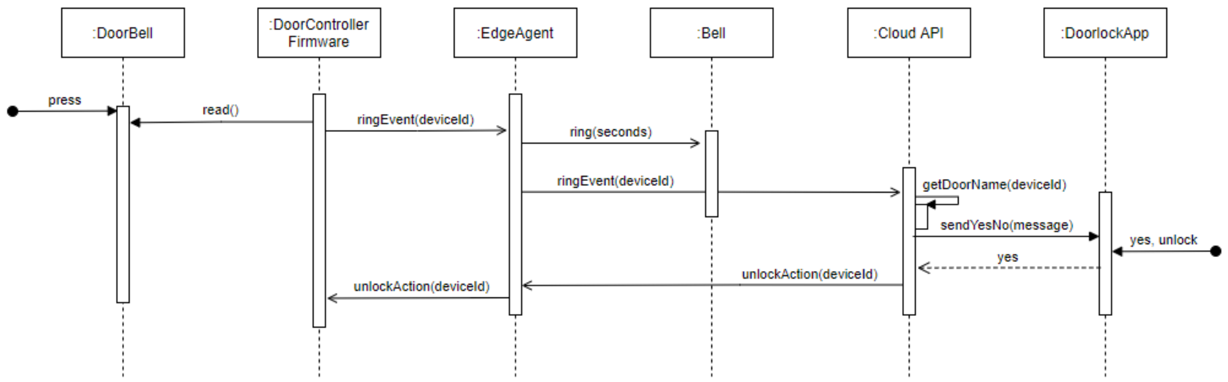


Figure 5: Sequence diagram of doorbell ring and mobile phone unlock operation

Remarks: The subsequent unlock, lock and updating of the lock state is not shown due to simplicity.

4. Evaluation/Experiments/Results/Discussion

The following Table 4 shows all the performed system tests to verify the operations (see Table 3) of the Smart Door Lock.

#	Testcase	Expected behaviour	Actual behaviour
		Local unlock operation	
1.1.1.	Present authorized NFC tag to card reader.	Door unlocks and green led acknowledges authorized NFC tag.	Door unlocks and green led acknowledges authorized NFC tag.
1.1.2.	Open the door.	Door opened shown in UI.	Door opened shown in UI.
1.1.3.	Shut the door.	Door closed shown in UI.	Door closed shown in UI.
1.2	Present unauthorized NFC tag to card reader	Door stays locked and red led acknowledges unauthorized NFC tag.	Door stays locked and red led acknowledges unauthorized NFC tag.
		Remote unlock operation	
2.1	Press unlock button of UI	Door lock unlocks.	Door lock unlocks.
		Remote lock operation	
3.1	Press lock button of UI	Door lock locks.	Door lock locks.
		Doorbell ring and mobile phone unlock operation	
4.1	Press doorbell button of Arduino and unlock door over telegram app on mobile phone.	Doorbell rings and door is unlocked as soon as telegram app approves unlock.	Doorbell rings and door is unlocked as soon as telegram app approves unlock.
		Intrusion detection	
5.1	Open door despite it is locked	Gateway sounds an alarm. User is notified on mobile.	Gateway sounds an alarm. User is notified on mobile.
		Suspicious activity detection	
6.1	1 min movement in front of door without lock or unlock requests	User is notified on mobile.	User is notified on mobile.
		Monitoring and logging	
7.1	Check Kibana	Logs of backend, gateway and door controller are visible.	Logs of backend, gateway and door controller are visible.

Table 4: Test protocol of Smart Door Lock

Tests successfully passed on 17th of December 2021. (Boas Meier, Tobias Heller)

5. Application(s)

The presented system allows multiple applications. One could be a simple door lock where no physical key is needed but instead a RFID Card or a smartphone is sufficient.

Additional applications are in the field of a doorbell functionality which also allows unlocking the door remotely without opening the door physically from inside. The most advanced use case for the Smart Door Lock is related to security. Two main features make this system especially useful in this case. First there is an intrusion detection where a door sensor is detecting a break in. The system afterwards sends a push notification to the user and an alarm sound is played to scare off the potential burglar.

To prevent this from happening there is a second functionality which is called "suspicious activity detection". A motion detector checks what is happening in front of the door. If the custom algorithm is detecting suspicious movements or repeated swiping of an unauthorized card a notification is sent to the user.

The lock and door states can be checked from all over the world over the web interface, just an internet connection is needed. There is no need to lend the keys to somebody, the locks can always be opened over the web interface.

6. Conclusion

In this project the team has been able to develop a working prototype of a Smart Door Lock with additional security features. Multiple hardware components like an Arduino Uno, Raspberry Pi, Motion sensors and more were selected and implemented. Additionally, these parts were connected through intelligent software using the MQTT protocol and monitored with the Elastic Stack. The provided functionality was evaluated with successful experiments.

If the project were to be continued additional effort would be used to create a better mobile app, add a camera as a security feature and introduce an identity and access management system. Furthermore, additional investments in the security architecture and a concept for zero-touch deployment models would improve the existing prototype.

7. Contributions / Acknowledgments

In the beginning the whole team was working on the idea generation for a project. After a few ideas were presented, the team decided on the smart door lock project.

After that all the members of the team contributed to the software architecture and worked on the project proposal. As described in 8.1 Team and Roles the components were distributed to the team members. In the following paragraphs the contributions of each team member are described.

Boas Meier

He was mainly responsible for the evaluation and wiring of the hardware as well as for the implementation of the firmware in C/C++. Furthermore, he supported Tobias Heller in the development of the edge agent. Other than that, he was the main driver of the system design.

Niklas Tanner

He worked on the mobile notification system (telegram bot) and made a Python-implementation which has been used within the backend. He also evaluated and developed the Angular frontend from scratch displaying the doors and related event-information using the backends' REST-API.

Lukas Mettler

He worked mainly on the logging and monitoring components. This consists entirely of the Elastic Stack. This includes a total of four components. Two log shippers (Filebeat), one configured on the edge device and one in the cloud. Elasticsearch for processing and indexing the logs and Kibana for visualization.

Tobias Heller

He was mainly working on the backend and gateway implementation. He evaluated the database system and built the backend REST-API in Python. Additionally, together with Boas Meier he worked on the definitions and the implementation of the messaging system with MQTT. Here and there he helped Niklas with the telegram bot, mainly for the integration into the backend.

8. Major Milestones & Deliverables

8.1 Team and Roles

Since we did work in an agile manner, we didn't have work packages but instead used single tasks which got distributed and were managed on Gitlab [2]. The following table shows which team members contributed mostly to a certain module.

TEAM	PROJECT MODULES	CONTRIBUTORS
GROUP 1	DoorControllerFirmware	Boas Meier
	Edgeagent	Boas Meier, Tobias Heller
	Filebeat	Lukas Mettler
	DoorlockCloudApi	Tobias Heller
	DoorlockDatabase	Tobias Heller
	ElasticSearchDatabase	Lukas Mettler
	KibanaDashboard	Lukas Mettler
	DoorlockFrontend	Niklas Tanner
	DoorlockApp	Niklas Tanner, Tobias Heller

Table 5: Project modules and the main contributors

8.2 Project Planning, Timelines, Milestones & Deliverables

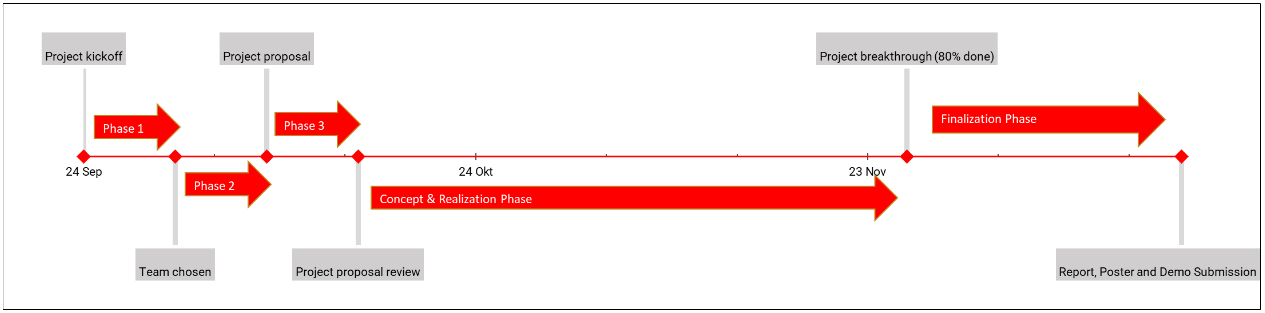


Figure 6: Timeline and Milestones

Date	Milestone	Deliverable	Responsible
24.09.2021	Project kickoff	-	
01.10.2021	Team chosen	Mail with team members	Boas Meier
08.10.2021	Project proposal	Project proposal document	All
15.10.2021	Project proposal review	Project proposal feedback	Angela Nicoara
26.11.2021	Project breakthrough (80% done)	80% of code and report done	All
26.12.2021	Report and Demo Submission	Final report and code	All

Table 6: Milestones and the deliverables

[2] Tasks in Gitlab: https://gitlab.enterpriselab.ch/IoT-I_BA_IOT/iot-i_ba_iot_h21/i-ba-iot-h2101/group01/-/issues?scope=all&state=all

10. References / Biography

10.1. References

Han, D., Kim, H., & Jang, J. (2017). Blockchain based smart door lock system. *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, 1165–1167. https://doi.org/10.1109/ICTC.2017.8190886

Pacheco, J., & Miranda, K. (2020). Design of a low-cost NFC Door Lock for a Smart Home System. *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 1–5. https://doi.org/10.1109/IEMTRONICS51293.2020.9216409

10.2. List of figures

10.2. Table directory

Appendix

11.1. Additional documentation of modules

11.1.1. DoorController

Figure 7 shows the schematics of the DoorController and the Gateway. The description of the different components is in Table 7.

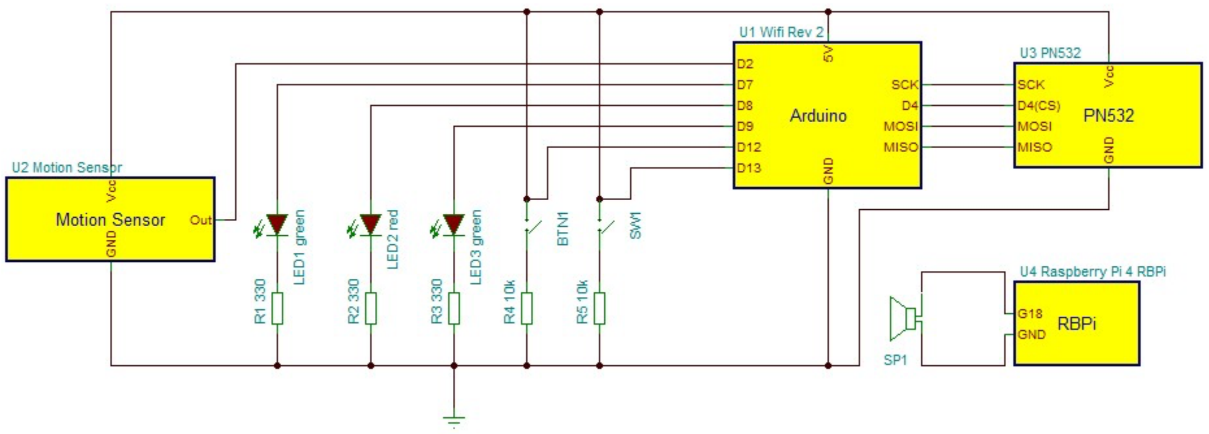


Figure 7: Schematics of DoorController

Identifier	**Description**
LED1 green	Doorlock mock. On if the door is locked.
LED2 red	On for 1s if NFC tag isn't valid and access isn't granted.
LED3 green	On for 1s if NFC tag is valid and access is granted.
BTN1	Doorbell. Normally closed. Triggers on falling edge.
SW1	Doorswitch. Triggers on rising and falling edge.
U1	Arduino Uno Wifi Rev 2.
U2	Motion Sensor.
U3	PN532 breakout board.
U4	Raspberry Pi 4.
SP1	Piezo speaker. Rings when doorbell is pressed and acts as an alarm.

Table 7: Circuit component description

11.1.2. Edge Agent

The following figure shows the inner workings of the EdgeAgent Code.

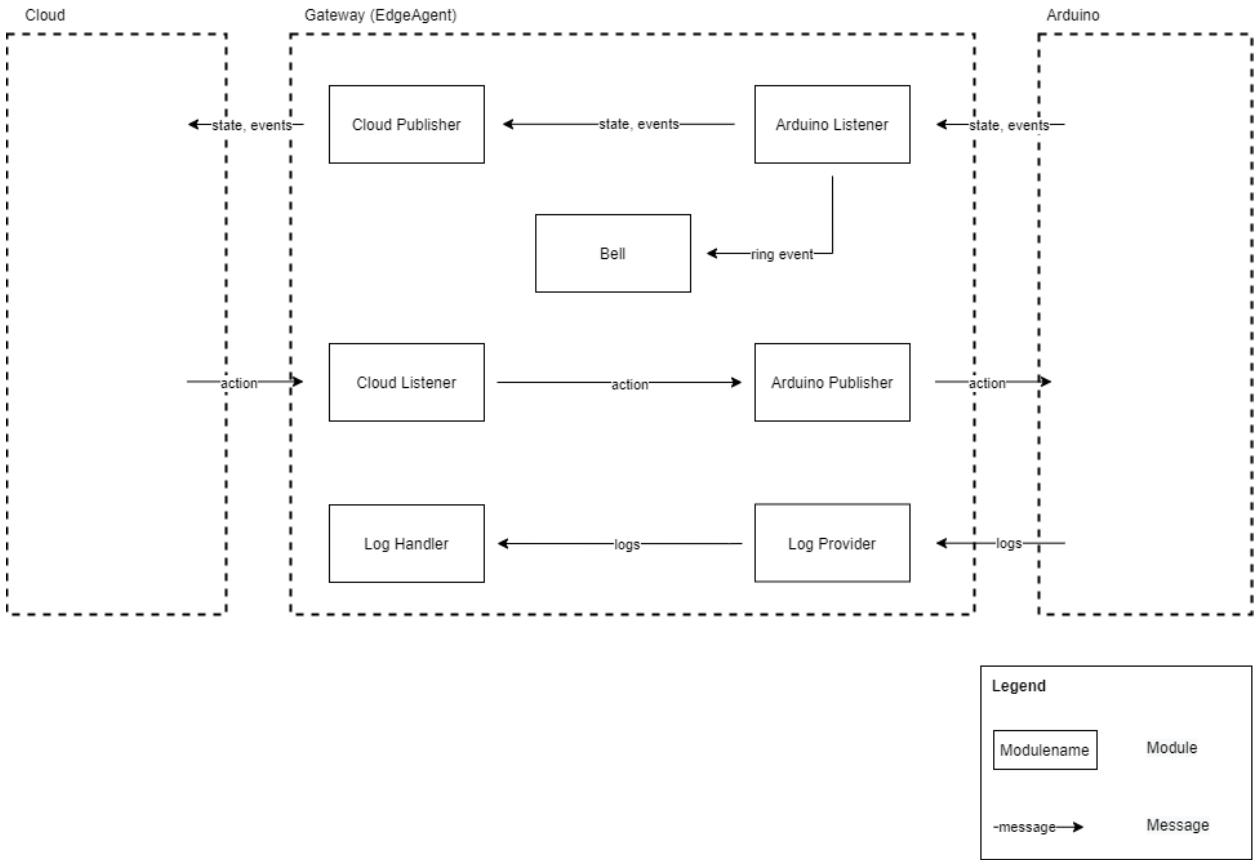


Figure 8: SW-Architecture EdgeAgent

11.1.3. DoorLockFrontend

The frontend is provided by a simple Angular WebApp. It contains three modules: App-Component, Doorlist-Component and Doorlock-Service. The App-Component represents a simple webpage, without any further content, except the Doorlist-Component. This sends GET or POST requests to the backend to receive a list of doors or to send actions for a single door.

A GET request could look like this: http://localhost:5004/doorlocks/iotlab

The backend then returns a list of doors as a JSON.

For each door another GET request is being done, to retrieve the device's specific event and action entries. These entries represent different events, for example "intrusion". These events are displayed by an inner table within the main table. It can be accessed by clicking on the target device.

Additionally, the GUI allows for execution of actions like "unlock" or "lock".

A POST request could look like this: http://localhost:5004/doorlocks/iotlab/1/action

The 1 in the URL is the id of the door, where the action should be executed. The body of a request to unlock the door could look like this:

```
{
  "action": "unlock"
}
```

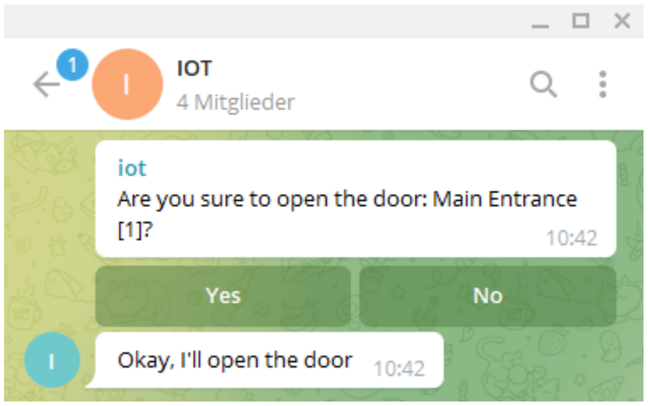
If an error occurs and an error-code is replied, a small notification-window shows up, stating that an error occurred. The error-details are also logged in the console of the browser.

Otherwise, the list refreshes itself every 5 seconds to stay up to date. Buttons are displayed depending on the current state of the referring door. If the door is locked the button shows "Unlock" and otherwise "Lock" is displayed. The POST-request also changes accordingly.

11.1.4. DoorLockApp

The DoorLockApp is implemented using a Telegram Bot. Telegram is a messenger app developed by Telegram Messenger Inc. and Telegram FZ LLC. It provides the possibility to create and develop custom bots. These bots can be created using the “BotFather” and are accessed by a token.
In this project the Python package “python-telegram-bot” is being used. It provides a variety of features including simple sending and reading messages up to handling whole conversations with states.

The bot is being triggered via the Cloud API if e.g., someone is ringing and sends a message into a group chat, asking if it should open the door. The message contains a custom keyboard with the two buttons “yes” and “no” which contain callback-information. This callback is used to determine further steps within the telegram bot. E.g., if you press “yes” the bot responds with a confirmation that he’ll open the door.


IOT
4 Mitglieder
iot
Are you sure to open the door: Main Entrance [1]? 10:42

Okay, I'll open the door 10:42

Figure 9: Telegram Bot example

Only one telegram-bot at a time can run. Otherwise, a conflict occurs, because only one bot can pull updates from the telegram servers at the same time. The bot also must run in a separate thread. But the python-package “python-telegram-bot” takes care of this problem as it performs as a wrapper and outsources the listening process automatically to another thread.

11.2. Detailed definitions of public interfaces

DoorControllerFirmware – Public Interfaces	
Interface	**Description**
Provides	
CardReaderHMI	Provides a green and red LED to indicate whether access is granted or not.
MQTT Doorlock Action	Provides an MQTT Interface for executing actions like unlock and lock of the doorlock.
Uses	
EdgeAgent	Used for transmitting state, logs, and events to the gateway.

Table 8: Public interfaces of DoorControllerFirmware

EdgeAgent – Public Interfaces	
Interface	**Description**
Provides	
MQTT Doorlock Action	Provides an MQTT Interface for executing actions like unlock and lock of a door lock for the provided site.
MQTT Telemetry	Stores telemetry like logs.
MQTT Doorlock State	Provides an MQTT Interface for updating the state of a door lock.
MQTT Doorlock Event	Provides an MQTT Interface for handling events of a door lock like ring, intrusion.
Uses	
DoorLockCloudAPI	Used for transmitting state and events to the cloud.
DoorControllerFirmware	Used for transmitting actions and receiving events and state updates from the door controller devices.

Table 9: Public interfaces of EdgeAgent

FileBeat – Public Interfaces	
Interface	**Description**
Provides	
Logs collection and shipping	Collects all Logfiles defined in the configuration and sends them to the selected output. Inputs: Logfiles from directory /var/log/* Dockerlogs of all running containers Output: elasticsearch
Uses	
Edgeagent logs	Filesystem and Docker logs of the Edgeagent.
DoorController logs	Filesystem logs of the DoorController.
Linux Filesystem logs	Filesystem logs for monitoring the Edge Device.

Table 10: Public interfaces of Filebeat

DoorLockCloudAPI – Public Interfaces	
Interface	**Description**
Provides	
Doorlock State API	Reads the state of the door locks.
Doorlock Action API	Allows the execution of actions e.g., unlock or lock for door locks.
Doorlock Name API	Allows the updating and creating of names for door locks.
MQTT Doorlock State	Provides an MQTT Interface for updating the state of a door lock
MQTT Doorlock Event	Provides an MQTT Interface for handling events of a door lock like ring, intrusion.
Uses	
Telegram API	Used for transmitting messages to the phone via telegram.
Edgeagent	Used for receiving the actions to the gateway.
DoorLock Database	Used for storing the state of the door locks.

Table 11: Public interfaces of DoorLockCloudApi

DoorLockDatabase– Public Interfaces	
Interface	**Description**
Provides	
Doorlock Lock State	Store and read the state of the lock.
Doorlock Door State	Store and read the state of the door.
Doorlock Name	Store and read the name of the door lock.
Uses	
-	-

Table 12: Public interfaces of DoorLockDatabase

ElasticSearchDatabase – Public Interfaces	
Interface	**Description**
Provides	
Indexed logs from all log shippers	indexing makes it possible to run queries against the data and you can use aggregations to retrieve summaries of the data.
Uses	
Filebeat	Filebeat collects the raw data and sends it to elasticsearch.

Table 13: Public interfaces of ElasticSearchDatabase

KibanaDashboard – Public Interfaces	
Interface	**Description**
Provides	
Log visualization	Kibana provides a user interface where all log data can be visualized, dashboards can be created, and the Elastic Stack can be managed.
Uses	
Elasticsearch	Takes the indexed logs from elasticsearch.

Table 14: Public interfaces of KibanaDashboard

DoorLockFrontend – Public Interfaces	
Interface	**Description**
Provides	
UI	Visual representation of door locks and enables the execution of the actions unlock and lock. For each door all passed events such as "intrusion" can be checked.
Uses	
DoorLockCloudAPI	Used for receiving the door locks and transmitting the actions.

Table 15: Public interfaces of DoorLockFrontend

DoorLockApp – Public Interfaces	
Interface	**Description**
Provides	
Telegram API	Used for transmitting messages to the phone via telegram.
RFID Emulator	Used to emulate RFID Cards to unlock the door locally.
Uses	
DoorLockCloudAPI	Used for receiving events and triggering actions.

Table 16: Public interfaces of DoorLockApp

11.3. Detailed definitions of modules

11.3.1. DoorControllerFirmware

MQTT Interfaces for DoorControllerFirmware

Message	MQTT-Topic	Payload	Example	Comments
Action				
Unlock Door Action	gateway/[deviceId]/action/unlock	-	-	Unlocks the specific door
Lock Door Action	gateway/[deviceId]/action/lock	-	-	Locks the specific door

11.3.2. EdgeAgent

MQTT Interfaces for Edge Agent

Message	MQTT-Topic	Payload	Example	Comments
Event				
Intrusion Event	gateway/[deviceId]/event/intrusion	-	-	Gets sent if door switch gets opened without unlock.
Movement	gateway/[device]/event/movement	-	-	Gets sent if someone is standing in front of the door.
Ring Event	gateway/[deviceId]/event/ring	-	-	Unlocks the specific door
Card Event	gateway/[deviceId]/event/card	{ “authorized”: “value” “uid” : “value” }	{ “authorized” : false, “uid”: “01 02 03 04 05” }	Gets sent if a card is swiped in front of the card reader. Possible values for authorized (true, false)
State				
Door State	gateway/[deviceid]/state/door	{ “doorState”: “value”	{ “doorState”: “open” }	Gets sent if the status of the Door changes.

		}		Possible values: “open”, “closed”
Lock State	gateway/[deviceid]/state/lock	{ “lockState”: “value” }	{ “lockState”: “locked” }	Gets sent if the status of the lock changes. Possible values: “locked”, “unlocked”
		Telemetry		
Log Message	gateway/[deviceid]/telemetry/logs	{ “timestamp”: “value”, “lvl”: “value”, “deviceid”: ”value”, “msg”: “value” }	{ “timestamp”: “2021-10-18T20:47:53”, “lvl”: “INFO”, “deviceid”: ”1234”, “msg”: “Motion detected” }	Gets sent if something gets logged in the firmware.

11.3.3. Doorlock Cloud API

REST Interfaces

The following APIs are available through http on the Port 5004.

Function	**Method**	**Endpoint**	**Return**	**Example**
Doorlock State API				
All doorlocks for Site.	GET	/doorlocks/[siteId]	List of all doorlocks for this siteId.	GET /doorlocks/iotlab [{ "deviceId": "1", "doorState": "open", "lockState": "unlocked", "name": "Main Entrance [1]" }, { "deviceId": "2", "doorState": "closed", "lockState": "locked", "name": "Office [2]" }]
One Doorlock with siteId and deviceId.	GET	/doorlocks/[siteId]/[deviceId]	One doorlock object.	/doorlocks/iotlab/1 { "doorState": "open", "lockState": "unlocked", "name": "Main Entrance [1]" }
Doorlock Action API				
Execute an action for a doorlock. [«unlock», «lock»]	POST	/doorlocks/[siteId]/[deviceId]/action	200 or 404	/doorlocks/iotlab/1/action { “action”: “unlock” }
Get all actions for this doorlock.	GET	/doorlocks/[siteId]/[deviceId]/action	200 or 404	[{ "action": "unlock", "datetime": "2021-11-19T09:30:06.003832", "message": "", "source": "api" }, { "action": "lock", "datetime": "2021-11-19T10:34:27.570701", "message": "",

				"source": "api" }]
Doorlock Event API				

Get all events for this doorlock.	GET	/doorlocks/[siteId]/[deviceId]/event	200 or 404	[
				{
				"datetime": "2021-11-22T09:21:39.891565",
				"event": "suspiciousactivity",
				"message": ""
				},
				{
				"datetime": "2021-11-22T09:21:49.155834",
				"event": "intrusion",
				"message": ""
				}
				]

Doorlock Name API				

Updates or creates the name of a doorlock.	PUT	/doorlocks/[siteId]/[deviceId]	200 or 400	/doorlocks/iotlab/1
				{
				“name”: “Main Entrance”
				}

MQTT Interfaces

Message	MQTT-Topic	Payload	Example	Comments
MQTT Doorlock Event				
Intrusion Event	[siteId]/doorlocks/[deviceid]/event/intrusion	-	-	Gets sent if door switch gets opened without unlock.
Suspicious Activity Event	[siteId]/doorlocks/[deviceid]/event/ suspiciousactivity	-	-	Gets sent if someone is standing in front of the door for too long.
Ring Event	[siteId]/doorlocks/[deviceid]/event/ring	-	-	Gets sent if someone rings the doorbell.
Movement	[siteId]/doorlocks/[device]/event/movement	-	-	Gets sent if someone is standing in front of the door.
Card Event	[siteId]/doorlocks/[deviceId]/event/card	{ “authorized”: “value” “uid” : “value” }	{ “authorized”: false, “uid”: “01 02 03 04 05” }	Gets sent if a card is swiped in front of the card reader. Possible values for authorized (true, false)
MQTT Doorlock State				
Door State	[siteId]/doorlocks/[deviceid]/state/door	{ “doorState” : “value” }	{ “doorState”: “open” }	Gets sent if the status of the Door changes. Possible values: “open”, “closed”
Lock State	[siteId]/doorlocks/[deviceid]/state/lock	{ “lockState”: “value”	{ “lockState” : “locked”	Gets sent if the status

		}	}	of the lock changes. Possible values: “locked”, “unlocked”

11.3.4. DoorlockDatabase

This data is stored in the Key-Value Database Redis.

Data	**Key**	**Value**	**Example**	**Comments**
		Doorlock Door State		
Door State	[siteId]:doorlocks:[deviceid]:state:door	{ “doorState” : “value” }	{ “doorState ”: “open” }	Possible values: “open”, “closed”
		Doorlock Lock State		
Lock State	[siteId]:doorlocks:[deviceid]:state:lock	{ “lockState”: “value” }	{ “lockState” : “locked” }	Possible values: “locked”, “unlocked”
		Doorlock Name		
Name	[siteId]:doorlocks:[deviceid]:name	{ “name”: “value” }	{ “name”: “Main Entrance” }	Any String possible.
		Doorlock Action		
Action	[siteId]:doorlocks:[deviceid]:action:[times tamp]	{ "action": "value", "datetime": "isostring", "message": "", "source": "value" }	{ "action": "unlock", "datetime" : "2021-11-19T09:30:0 6.003832", "message": "", "source": "api" }	Possible values for source: “api”, “telegram”, “unknown.
		Doorlock Event		

Event	[siteId]:doorlocks:[deviceid]:event:[timestamp]	{ "event": "value", "datetime": "isostring", "message": "", }	{ "event": "intrusion", "datetime": "2021-11-19T09:30:06.003832", "message": "" }	-