



Expertise
and insight
for the future

Internet of Things Project

The Door – Technical Documentation

January 2022

Contents

1	Introduction	1
1.1	Installation	2
2	3D miniature model	2
2.1	Components	2
3	Frontend	4
3.1	Login page	4
3.2	Main page	4
3.3	History page	4
4	Backend	5
4.1	Authentication	5
4.2	Database	5
4.3	Lock states	6
4.4	MQTT	7
5	Embedded	8
6	Conclusion	9

1 Introduction

The Door is an Internet of Things project, which provides its users with a door control system. This system regulates the different lock states of the door via a web interface. Depending on the lock state, the door can be opened with assigned NFC cards, through the detection of motion or not at all. Furthermore, it is possible to open the door manually by clicking a button in the web interface. Also, the web interface offers a history page. The history page displays information when and by whom a NFC card was used to open the door, as well as a documentation of the changed log modes of all users.

In the project, the system consists of a microcontroller (Arduino Uno), a servo motor, an Ethernet Shield Module, two sensors (a passive infrared motion and a Grove NFC), a server and a web interface. Together they build a miniature door control system. The microcontroller controls the servo motor, which opens and closes the door, as well as the sensors. Moreover, the microcontroller has a connection to the server, which is hosting the web interface, making it possible to set the different lock states or to open the door manually.



This technical documentation provides information about frontend, backend and the embedded part of the system. It however does not include any descriptions of how to use the web interface. This information can be found in the User Manual.

1.1 Installation

This project is based on Go which needs to be installed to run the server.

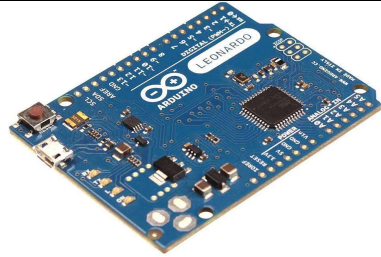

Moreover Fiber, MQTT, ArangoDB and Protobuffer packages are also required to run the server. For more detailed information please look at the `go.mod` file.

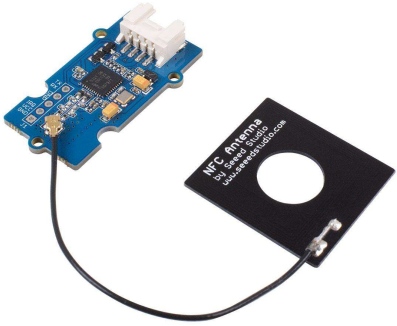




To sign in on the web interface, the credentials ('User', 'User') can be used.

2 3D miniature model

The door control system model was designed using the "Fusion 360" software. Afterwards the door and the platform were printed with grey and blue PLA plastic using Metropolia's 3D printer. The holes for the components were either cut or not printed. The model has got 3 LEDs; a green one, a yellow one and a red one. They are used to indicate the different door modes. Green for "UNLOCKED", yellow for "SOFT" and red for the "HARD" mode.

2.1 Components

Arduino Uno	
5V servo motor	

Grove NFC sensor	 <p>A blue Grove NFC sensor module with a black circular antenna and a white connector.</p>
Ethernet Shield Module W5100	 <p>A blue Ethernet Shield Module W5100 with a silver Ethernet port and a black USB connector.</p>
Grove (MAXREFDES131#)	 <p>A green Grove (MAXREFDES131#) module with a black connector and a small black component.</p>
Passive infrared motion sensor	 <p>A blue passive infrared motion sensor module with a white sensor lens and a white connector.</p>
LED's	 <p>Three small LEDs in red, yellow, and green colors.</p>

3 Frontend

The web interface was created using HTML, a Bulma framework, a bit of custom CSS as well as JavaScript. The design as well as its functionality is kept simple to enhance the usability. The website can be divided into three main parts: login page, main page and history page.

3.1 Login page

The login page displays the name of the website as well as the logo. Here, the user can enter his*her credentials to enter the web interface. With the `userDataHandler()` function the user's name and the current lock state of the main page are displayed on the main page.

```
func userDataHandler(c *fiber.Ctx) error {
    user := c.Locals(key: "user").(*jwt.Token)
    claims := user.Claims.(jwt.MapClaims)
    name := claims["name"].(string)

    s := "{\"name\":\"" + name + "\", \"mode\":\"" + tcpPacketOut.GetLockStatus().String() + "\"}"
    return c.SendString(s)
}
```

3.2 Main page

The main page offers the actual functionality of the web interface. The user can change the lock mode of the door and open the door manually. The picture of the lock also displays the current state of the door. On the top of the website there is a navigation bar, to easily lock out or get to the history page.

3.3 History page

The history page provides two tables. The left table is the `DOOR_HISTORY` and the right one the `LOCK_HISTORY` table. The `keycardHistoryHandler()` and the `lockHistoryHandler()` functions return json packages, so the data can be shown in the tables of the website.

4 Backend

The server's functionality mainly is found in the `networking.go` file. It hosts the database, the static content (HTML, CSS, JS) and also enables the MQTT connection. Go is used as programming language and Fiber is used to manage the routing.

4.1 Authentication

The access to the web interface is restricted. It is for logged-in users only. Therefore, the routes, which the server handles besides the `/login`, require the client to authenticate before the server provides any information. So, the routes need to provide the generated token, which is the key to access the website.

```
// JWT Middleware
app.Use(jwtware.New(jwtware.Config{
    SigningKey: []byte("secret"),
}))

/*      ---    RESTRICTED APIS    ---    */
app.Get( path: "/getInitData", userDataHandler)
app.Put( path: "/manualOpen", forceOpen)
app.Put( path: "/updateLock/:lockMode", updateLock)
app.Get( path: "/statistics/modeChanged", lockHistoryHandler)
app.Get( path: "/statistics/keycardUsed", keycardHistoryHandler)
```

The following functionalities are implemented in the `loginHandler()` function. When a user connects to the website, he*she is asked to enter his*her username and password. The user's credentials are sent to the server using a POST request. If the username is found in the database, the users' password gets hashed. This hashed value then gets compared to the users' hashed password, which is stored in the database. A token, which is generated for each login activity, is sent to the client, if both hashed values are the same. Every time, the user wants to interact with the website, the token is sent and compared with the tokens saved on the server.

4.2 Database

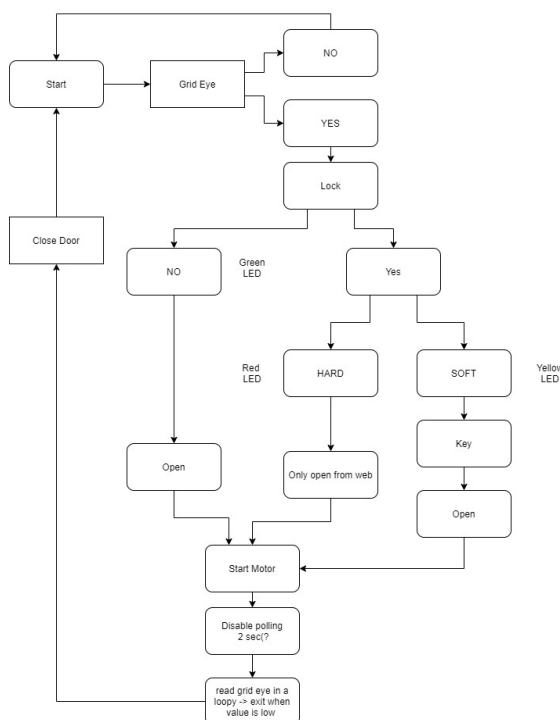
The server uses an ArangoDB database to store the necessary data. In the `database.go` file code connected to the database can be found. The database contains four tables: `DOOR_HISTORY`, `DOOR_LOGIN`, `DOOR_RFID` and `LOCK_HISTORY`.

DOOR_HISTORY stores information about who (which NFC card) opened the door and when it is done. DOOR_LOGIN stores the users of the website and DOOR_RFID the NFC card numbers. LOCK_HISTORY stores information about which website user changed the door lock mode, which mode is active following the change and when this was done.

```
// Create a collection for door history
userCol, err = db.Collection(ctc nil, name: "DOOR_HISTORY")
if err != nil {
    fmt.Println(err, "creating new...")
    ctx := context.Background()
    options := &driver.CreateCollectionOptions{ /* ... */ }
    userCol, err = db.CreateCollection(ctx, name: "DOOR_HISTORY", options)
    if err != nil {
        fmt.Println(err)
    }
}
```

4.3 Lock states

The door control system can be set to three different lock modes: "UNLOCKED": 1, "SOFT": 2, and "HARD": 3. In the "UNLOCKED" mode the door opens when the passive infrared motion sensor detects motion. There are no further restrictions in this mode. "SOFT" mode requires an NFC card to be able to open the door. When the door mode is set to "HARD", it can only be opened by clicking a button in the web interface.



The user is able to change the lock modes in the web interface. This functionality is implemented in the `updateLock()` function. The changed lock mode is received through the URL. The parameter is then converted from a string to an integer. It also gets the name of the user for the `LOCK_HISTORY` table. Furthermore, a door request is sent to the microcontroller in order to open the door of the model.

The `forceOpen()` function sets the tcp package to `Door_Request_Approved` and sends it to `handleMQTTout()`. This tells the embedded device to open the door.

```
func forceOpen(c *fiber.Ctx) error {
    tcpPacketOut.LockStatus = Door_Request_HARD // Just making sure
    tcpPacketOut.DoorRequest = Door_Request_APPROVED
    handleMQTTout()
    fmt.Println(a... "door manually opened")
    tcpPacketOut.DoorRequest = Door_Request_NO_REQUEST
    return c.SendStatus(status: 200)
}
```

4.4 MQTT

MQTT handles the communication between the microcontroller and the server. Both can publish and subscribe to a specific topic to transport or get their data. The topic "door/request" deals with door lock commands and the topic "door/RFID" is for the NFC card number.

`MQTT_OUT.go` implements the data transport from the server to the microcontroller. The protobuf data, which is needed for the door request is sent to the microcontroller. There are different door requests `Door_Request_LOCK_NOT_SET`, `Door_Request_UNLOCKED`, `Door_Request_SOFT` and `Door_Request_HARD`.

`MQTT_IN.go` implements the data transport from the microcontroller to the server. The server is listening to incoming NFC card packages from the MQTT server. If the server finds a package, meaning someone used an NFC card to open the door, it encrypts it and looks if the database stores that specific NFC card. If the server finds this value, this value as well as the time when the card was read, are put into the `DOOR_RFID` table. Afterwards the lock state is set to `Door_Request_APPROVED`, because the package was found.

Otherwise, the lock state would be set to `Door_Request_DISAPPROVED`. The respective door request is then sent to the MQTT server, so the microcontroller knows, if it should open the door or not.

5 Embedded

The microcontroller is responsible for controlling the different sensors (NFC, passive infrared motion), the servo motor and the LEDs. To enable this, different libraries were used. `Servo.h` handles for example how the servo motor is moving the door, the `ledstatusHandler()` function sets the led colour to the received lock mode and `readGridEye()` detects motion.

Furthermore, the microcontroller also interacts with the MQTT server. The microcontroller is subscribed to the `door/request` topic. It listens to it and if the user changes the lock mode, the information should be read from the MQTT-Server. To do so the microcontroller also has to connect to MQTT and has to start an Ethernet connection.

```
randomSeed(analogRead(0));
String user = "ARDUINO_" + String(random(300));

Ethernet.begin(mac, ip);
if (client.connect(user.c_str())) {
  client.subscribe("door/request");
} else {
  Serial.println("fail");
}
}
```

Additionally, the microcontroller sends the hex value of the NFC card to the client's webpage by publishing the data on `door/Rfid`. The server receives the MQTT message and processes it.

```
client.publish("door/Rfid", hex);
```

6 Conclusion

To conclude the project was a little bit challenging, but turned out successful in the end. The microcontroller was changed from an LPC to an Arduino, because there were some challenges with the programming. Furthermore, the Grideye sensor, couldn't be implemented as expected, because it is very complex to actually detect a human with it. It was only working on a basic level (detecting heat changes) and therefore, we decided to use a passive infrared motion sensor in the end. Nonetheless the door control system is working and we provided a technical documentation, as well as a User Manual. Moreover, we were able to gain experience in working as team as well as acquire new knowledge in the area of IOT.