UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

Project Documentation

Dual Unit Access Control System with RFID and PIN
IoThings Project

David Gherghita

Project advisor:

Gabriel-Cosmin Chenaru

BUCHAREST

2022

CONTENTS

1 Introduction **2**

2 Circuit Schematics **4**

3 Securing the Cards **6**

4 Communication Protocol **8**

5 Program Flow **10**

6 Components **12**

7 References **13**

1 INTRODUCTION

The aim of the project is to showcase a possible implementation of an access control system that can be used to secure access through a door.

The system is composed of two units, one on the unsafe side of the door, on the outside, and one on the safe side of the door, on the inside. The inside unit is also the one that has control over the doors locking mechanism. A use-case of such a system is presented in figure 1.

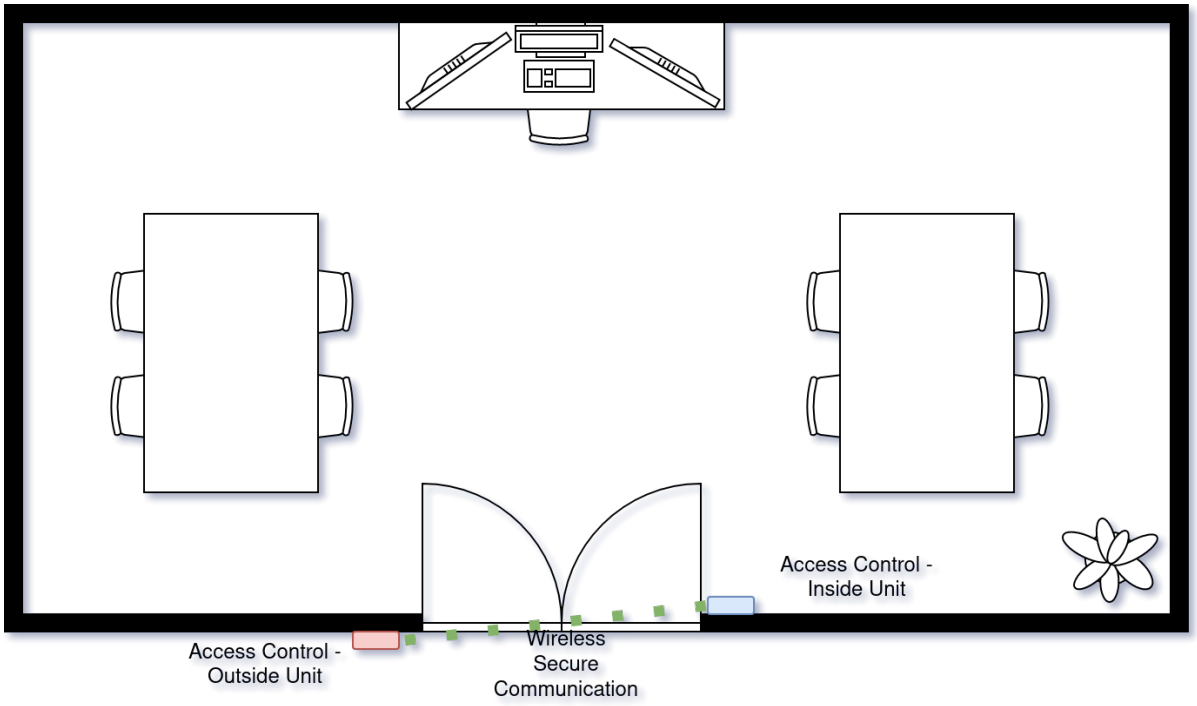

Figure 1: Building schematics of the access control system

A dual-unit design was chosen instead of a single device on the outside that has control over the doors locking mechanism because in the second scenario, an attacker could bypass the authentication mechanism by opening the device and bridging the correct electric contacts in order to trigger the opening of the door.

The communication between the 2 devices is wireless, in order to greatly reduce the installation complexity. As the devices used are ESP32 boards, direct wireless transfers are easy to implement with the help of the ESP-NOW protocol. However, wireless data transfers pose many risks.

First of all, because wireless snooping is trivial, the 2 devices must transfer data over an encrypted channel. This is achieved by using the encryption functionality of the ESP-NOW

protocol.

Secondly, wireless communication can also be spoofed in a replay attack. Even if the transmission is encrypted, the data can be captured and then sent again in order to open the door. This is made impossible by using a challenge-response protocol which guarantees that only valid messages are accepted.

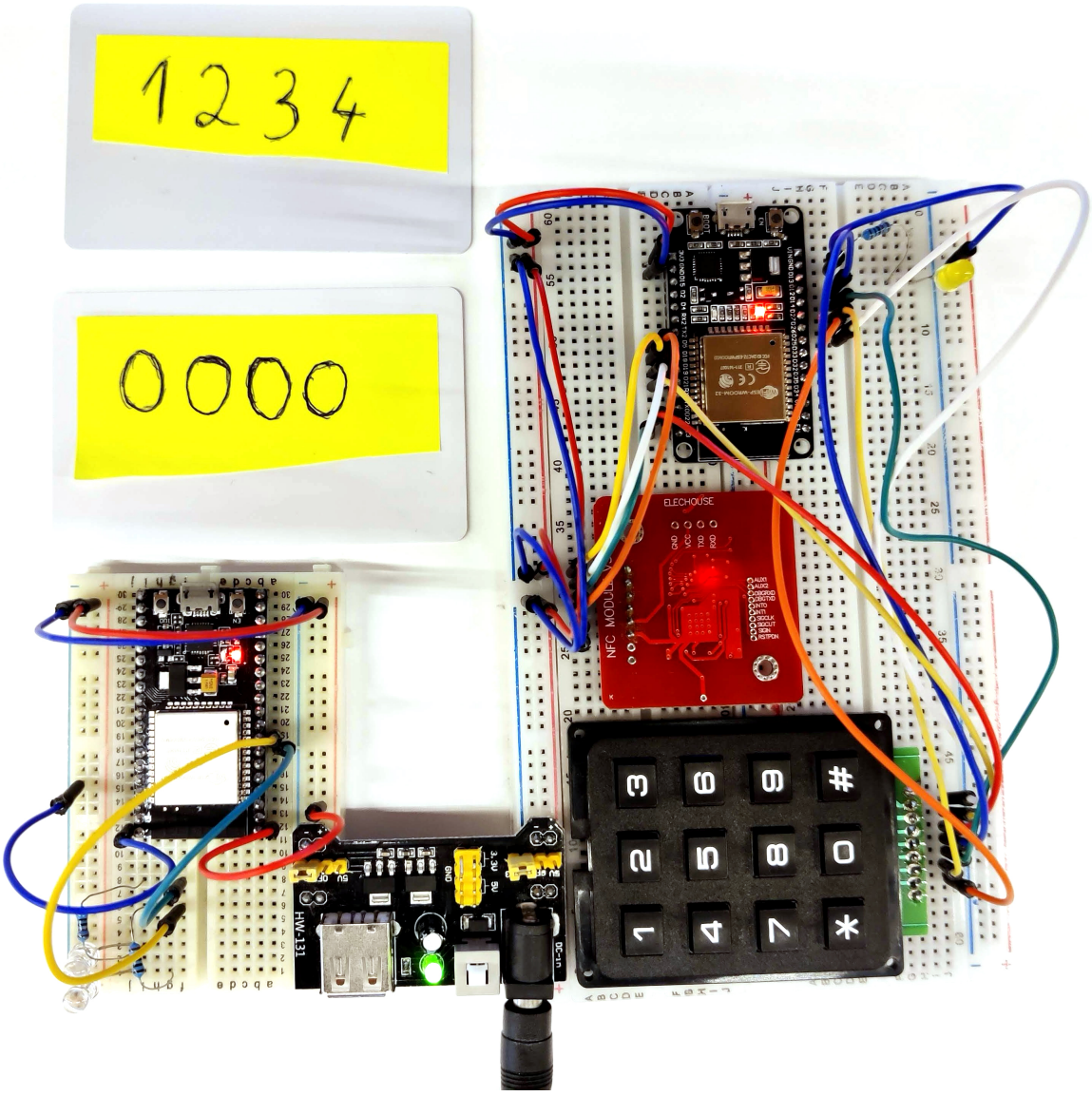

Figure 2: Circuit build on my desk

Figure 2 shows the actual circuit tested. On the right side it's the external unit and on the left side it's the internal unit.

2 CIRCUIT SCHEMATICS

The figures show a simplified mode of connecting the components.

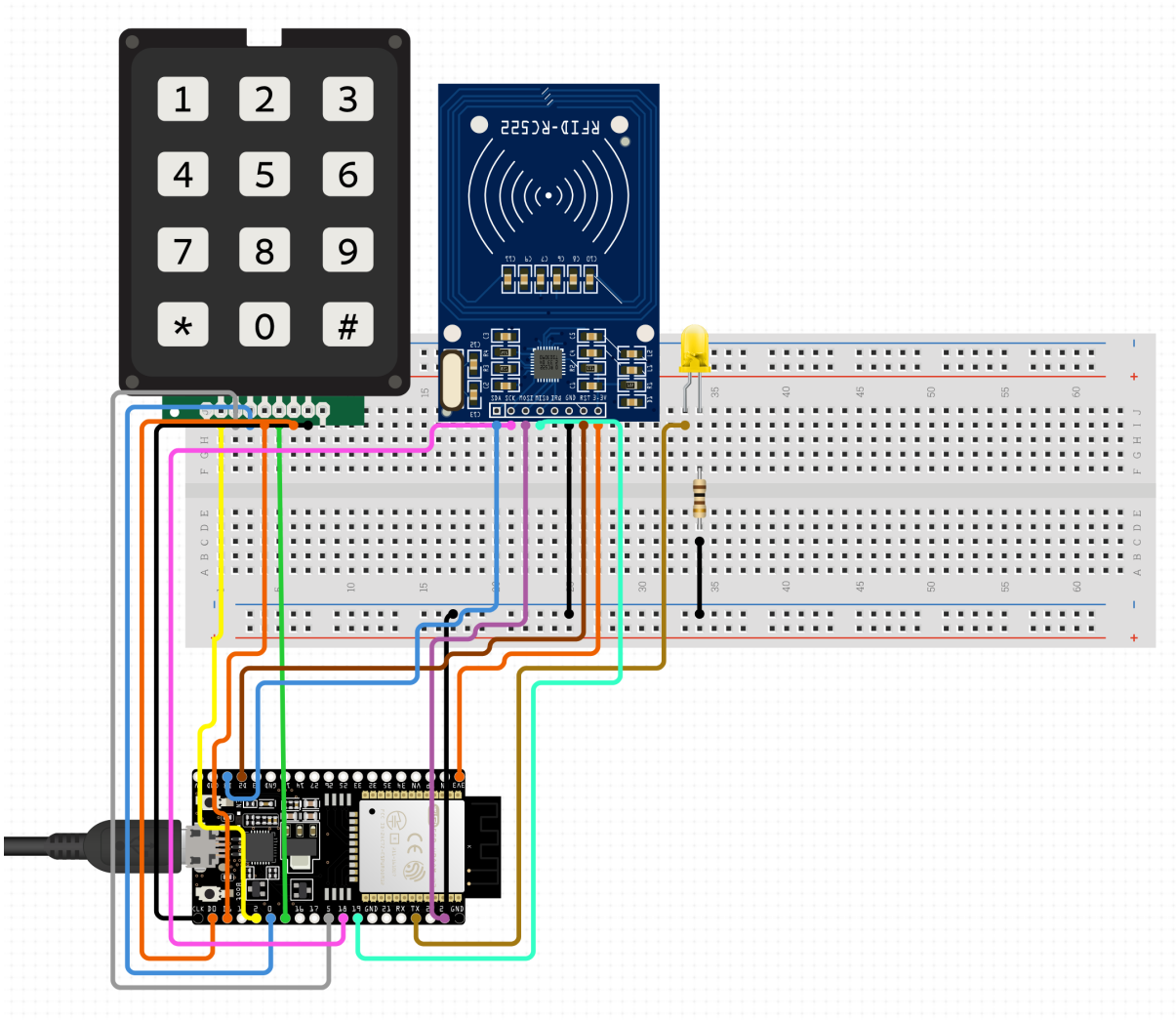

Figure 3: Circuit schematics of the client (the outside) unit

Figure 3 shows the circuit for the client, i.e. the outside, unit of the system. The RFID reader is connected to the development board via I2C. The code of the unit is written as read data from the keypad only a certain time (a few seconds) after a valid card was presented. That time period is signaled using the yellow LED. When the LED turns on, it means that the card was successfully read and that the PIN is required to continue.

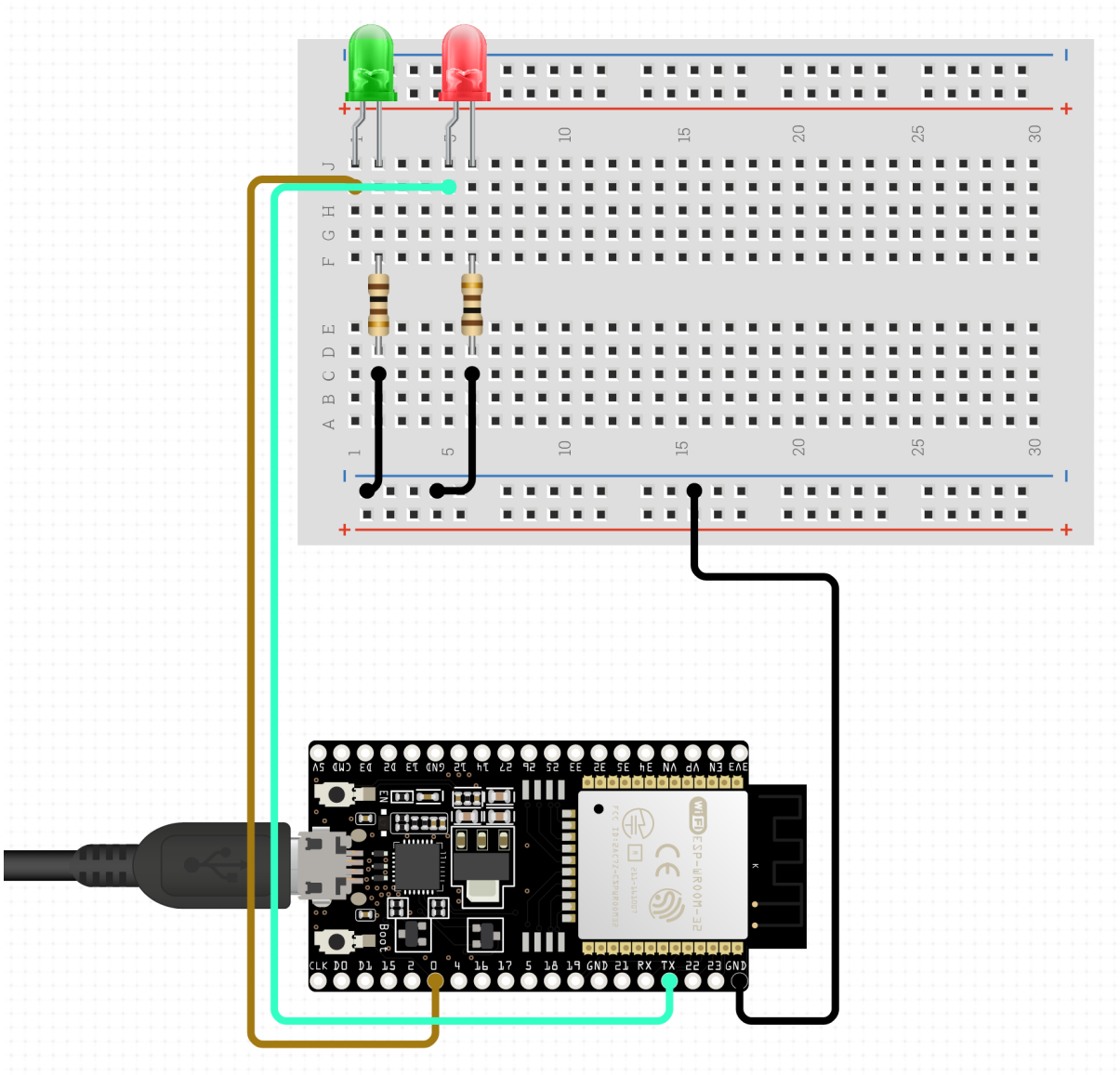

Figure 4: Circuit schematics of the server (the inside) unit

In figure 4 it's shown that the inside unit consists of only the ESP32 connected to two LEDs: green and red. The green LED means that the authentication was successful and that access is granted, while the red one means that the authentication failed and the door stays closed.

3 SECURING THE CARDS

As the project is security-oriented, special care was taken to secure the cards used. The type of the cards is MIFARE Classic 1K. The same ones are currently used for the municipal public transport in Bucharest, STB.

The easiest way to use this cards is with NDEF (NFC Data Exchange Format), but this is an unauthenticated method of interacting with the card, as even a mobile phone with a simple app can read or write data to the card. In order to obtain a secure card, it was necessary to understand the built-in mechanisms of the MIFARE card. This was accomplished using the data sheet provided by NXP, especially "Section 8.7: Memory Access".

The MIFARE Classic cards have the memory laid out into 16 sectors of 4 blocks each. A block contains 16 bytes. Except from sector 0, the first 3 blocks of a sector are data blocks to be used for user data, and the last one is the trailer, which, as seen in figure 5, contains Key A, Access Bits and Key B.

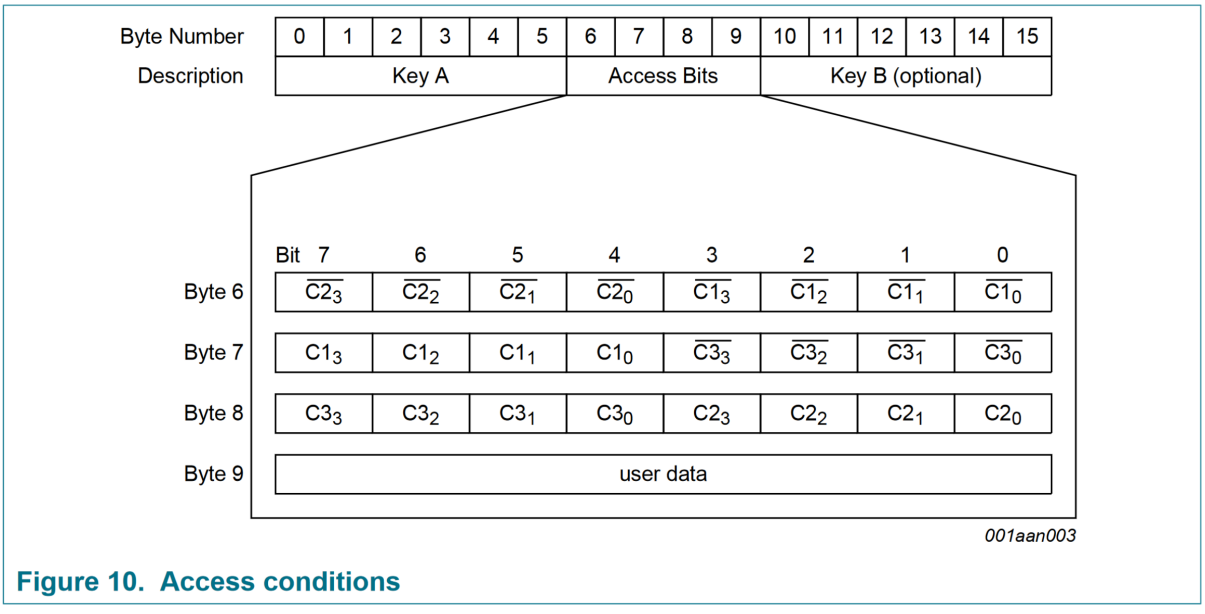

Figure 5: Format of the trailer block

In order to secure the cards, a secret key was written into Key B and the access bits were set as follows.

As the data blocks contained the card number used in the access control system implemented, read-only access was needed after successfully authenticating the card with KEY B. As a result of this, the combination used was `1 0 1`, which can be seen in figure 6.

Table 8. Access conditions for data blocks

Access bits			Access condition for				Application
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A|B	key A|B	key A|B	key A|B	transport configuration[1]
Access bits			**Access condition for**				**Application**
0	1	0	key A|B	never	never	never	read/write block[1]
1	0	0	key A|B	key B	never	never	read/write block[1]
1	1	0	key A|B	key B	key B	key A|B	value block[1]
0	0	1	key A|B	never	never	key A|B	value block[1]
0	1	1	key B	key B	never	never	read/write block[1]
1	0	1	key B	never	never	never	read/write block[1]
1	1	1	never	never	never	never	read/write block

Figure 6: Access bits for the data blocks

For the trailer access bits, the most secure choice would be `1 1 1`, as shown in figure 7, but since the project aims only to showcase the access control system, forever locking the cards would bring no benefit. So, the combination used was `0 1 1`, which allows changing the access bits with Key B. With this permission set, the cards can be brought back to a default NDEF-compatible configuration.

Table 7. Access conditions for the sector trailer

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read[1]
0	1	0	never	never	key A	never	key A	never	Key B may be read[1]
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration[1]
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

[1] For this access condition key B is readable and may be used for data

Figure 7: Access bits for the trailer blocks

4 COMMUNICATION PROTOCOL

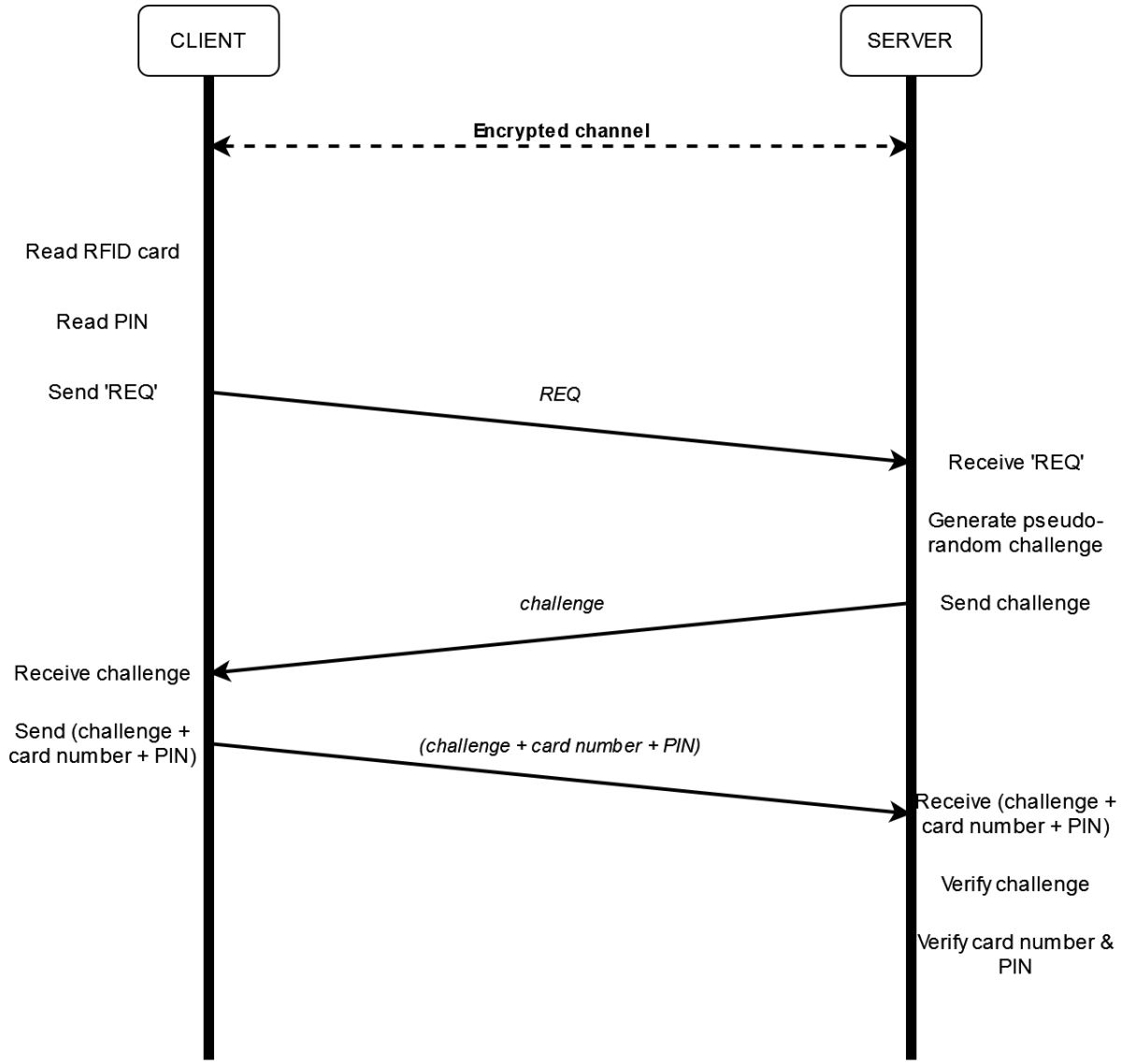

Figure 8: Communication protocol between the client and the server units

Figure 8 shows the data transitioned between the 2 ESP32 units on the encrypted channel established with ESP-NOW.

The communication begins only after the client unit has successfully read an RFID card and a PIN. When these conditions are met, the client sends a "request" message, i.e. a message that contains only the message `REQ`.

When the server receives the `REQ` message, it generates a pseudo-random integer, called a challenge, and sends it to the client. This integer has the purpose of deterring the use of old

messages, as it's done in the case of a replay attack.

After receiving the challenge, the client sends in a single message all the information needed:

1. The challenge;
2. The data from the card;
3. The PIN.

As the message is encrypted, a snooping attacker is not able to get the card data or the PIN. Moreover, by including the pseudo-random challenge in the encrypted message, it's guaranteed that there every sent message will be different, therefore, an attacker can't replay an already sent message.

This mechanism can be better implemented by using hashing algorithms. However, due to limited support on the ESP32 boards, I've decided against it. The added benefit would be that the user credentials are never sent between the 2 devices, so even if an attacker could actually manage to decrypt a message, he could neither find out the card data or PIN, or replay the message.

The last steps of the protocol involve the server unit receiving the message, checking the challenge, card number and PIN and taking the correct action: if the data is right power on the green LED (access granted) and if the data is wrong power on the red LED (access denied).

5 PROGRAM FLOW

As the 2 units run different programs, the program flows are displayed for the client in figure 9 and for the server in figure 10. Although similar, they differ in key aspects so that they can function as a whole and provide the needed service.

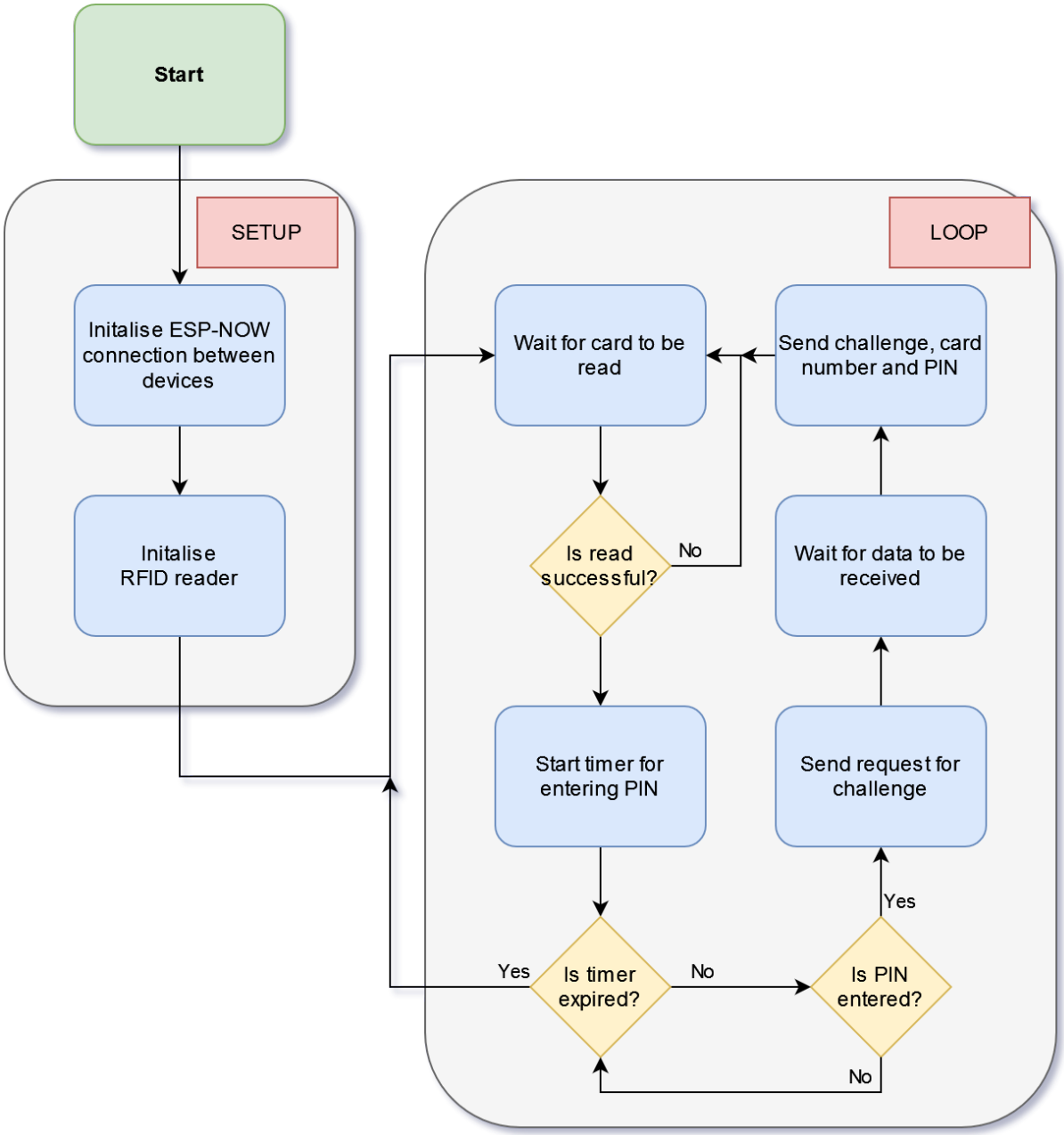

Figure 9: Program flow of the client device

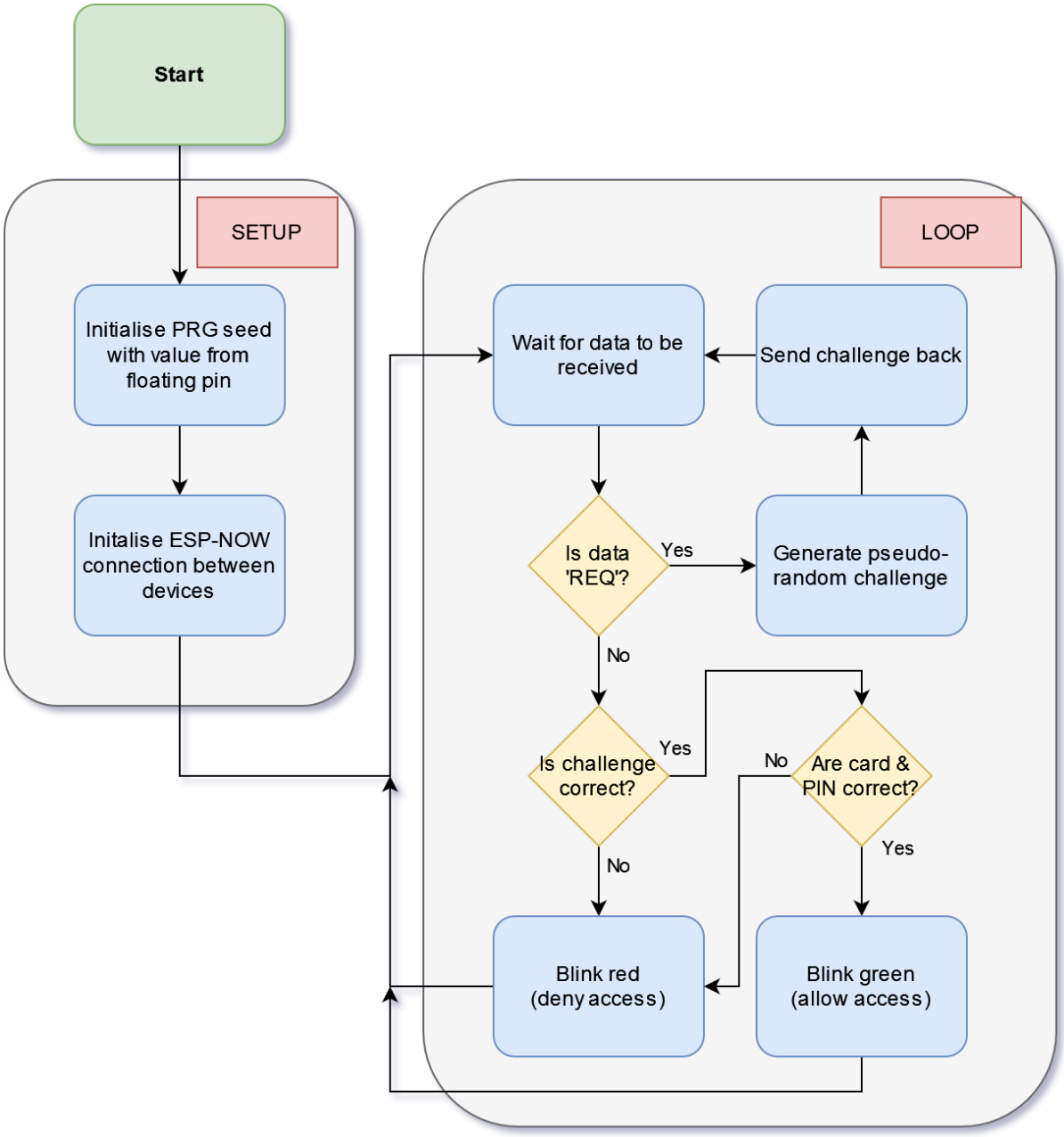

Figure 10: Program flow of the server device

6 COMPONENTS

- ESP-WROOM-32 (client-side)
  https://www.sigmanortec.ro/
  placa-dezvoltare-esp32-cu-wifi-si-bluetooth
- NodeMCU-32S (server-side)
  https://ardushop.ro/ro/home/1449-nodemcu-32s-38.html
- RFID Reader PN532
  https://www.optimusdigital.ro/ro/wireless-rfid/
  3905-modul-rfid-pn532-placa-electronica.html
- Numpad
  https://www.sigmanortec.ro/tastatura-numerica-4x3-12-butoane
- Breadboard + Breadboard Power Supply + Wires
  https://www.sigmanortec.ro/
  Kit-Breadboard-MB102-Sursa-Fire-dupont-p136264993
- 9V Power Supply
  https://www.sigmanortec.ro/
  Sursa-alimentare-9v-Pentru-Arduino-5-5x2-1-p126029436

7 REFERENCES

- LockPickingLawyer - RFID Lock Defeated With a Paperclip
  https://www.youtube.com/watch?v=z4lVyl07y5U
- LockPickingLawyer - Bugging an RFID Card Reader
  https://www.youtube.com/watch?v=0SEHUqkbIjU
- Getting Started With ESP32
  https://randomnerdtutorials.com/getting-started-with-esp32
- How To Set Up a Keypad On an Arduino
  https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino
- PN532 NFC Library for Arduino
  https://github.com/elechouse/PN532
- MIFARE Classic Data Sheet
  https://www.nxp.com/docs/en/data-sheet/MF1S50YYX_V1.pdf