UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

The Management System of Network Switch Based on an Embedded Nano Pi Platform

Author:
Danylo SLUZHYNSKYI

Supervisor:
Anton PUTRYA

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Danylo SLUZHYNSKYI, declare that this thesis titled, "The Management System of Network Switch Based on an Embedded Nano Pi Platform" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"God helps those who help themselves"

Algernon Sidney

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

The Management System of Network Switch Based on an Embedded Nano Pi Platform

by Danylo SLUZHYNSKYI

Abstract

The purpose of this bachelor's thesis is to design and implement devices that can address the problem of managing ethernet and power paths, with scheduling and a modern security authentication policy.

Code can be found here:
Github repository

Acknowledgements

I am thankful to my family for their emotional, personal, and financial support that they provided me with throughout all four years of my study. I want to thank my supervisor Anton PUTRYA and Andrew DOBUSH for mentoring me, for their guidance and advice during my research, and for all the consultations they provided.

I am deeply grateful to all my teachers from university, especially Oleg FARENYUK for all the knowledge that came in handy in this bachelor's thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

GUI	**G**raphical **U**ser **I**nterface
UI	**U**ser **I**nterface
MAC	**M**edia **A**ccess **C**ontrol
LLC	**L**ogical **L**ink **C**ontrol
VoIP	**P**ower **V**oice **O**ver **I**nternet **P**rotocol
QOS	**Q**uality **O**f **S**ervice
STP	**S**panning **T**ree **P**rotocol
IOT	**I**nternet **O**f **T**hings
OSI	**O**pen **S**ystems **I**nterconnection
PoE	**P**ower **O**ver **E**thernet
UART	**U**niversal **A**synchronous **R**eceiver **T**ransmitter
SBC	**S**ingle **B**oard **C**omputer
GSM	**G**lobal **S**ystem for **M**obile
MII	**M**edia **I**ndependent **I**nterface
SGMII	**S**erial **G**igabit **M**edia **I**ndependent **I**nterface
RGMII	**R**educed **G**igabit **M**edia **I**ndependent **I**nterface
HSGMII	**H**igh **S**erial **G**igabit **M**edia **I**ndependent **I**nterface
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor

To my Father, Markiyan Sluzhynskyi

Chapter 1

Introduction

1.1 Motivation

Decades ago, saying a command to turn music on, or asking "will it rain today?" and getting an actual answer from the speaker was nothing short of a pipe dream.

Today, however, it is common to say: "Siri, set a timer for 45 minutes" or "Alexa, turn on bedroom lights." Nevertheless, nowadays smart devices have developed and expended far beyond smart speakers. From energy-saving thermostats to remote control devices, like smart bulbs, or intelligent yoga mats, all of them need an internet connection, or they at least need to be connected to a local network. Some of them support a Wi-Fi connection, but some of them support only wired ethernet connectivity, like cameras, printers, or even intelligent ovens. Here, the smart switch comes in front, and the main idea is to control some devices or nodes of devices being connected to ethernet and power, because if they are connected all the time, it will cause security issues or could damage to devices or even the whole home, office, factory. Also it is reasonable from an energy-saving perspective.

1.2 Goals

Provide software control based on the Nano Pi platform for a multiport network switch. Achieving this goal in this bachelor thesis is associated with solving the following tasks:

1. Configure the Armbian operating system to run on the Nano Pi platform;
2. Implementation of the authorization subsystem in the port status management system (email, phone);
3. Implementation of the Nano Pi network settings subsystem;
4. Implementation of the module of direct control of a condition of ports of the switch;
5. Implementation of the module of planning of tasks on the management of a condition of ports;
6. Implementation of the command interpreter that controls switch;
7. Implementation of the basic functionality for managing Nano Pi;
8. Implementation of the module of scanning Nano Pi in local area network.

1.3 Thesis Structure

The remainder of the thesis is structured as follows. Chapter 2 reviews the required knowledge for project understanding and essential information related to its topics. In Chapter 3, we review the relevance of the project, existing related works, solutions, and competitors. In Chapter 4, we present used hardware, including details of setup. Used frameworks and software modules we present in Chapter 5. In Chapter 6, we present our approach, including the details of the implementation of flows and main functionalities. Finally, we make conclusive remarks in Chapter 7 with a discussion of future work that would be applied.

Chapter 2

Background Information

2.1 OSI Model

One of the best ways to understand the purpose of different network devices is to understand the layers of the OSI model (Petryschuk, 2021).

The OSI Model is a conceptual framework used to describe the functions of a networking system. This model characterizes computing functions into a universal set of rules and requirements in order to support interoperability between different products and software (Froehlich, 2021). In the OSI reference model, the communications between a computing system are split into seven different abstraction layers: Physical, Data Link, Network, Transport, Session, Presentation, and Application.

Table 2.1 shows the venerable OSI model in all its seven-layer glory, along with major functions for each layer (Harry Reynolds, 2009; Emmett Dulaney, 2011)

TABLE 2.1: OSI Model

OSI Layer	**Major functions**
Physical (Layer 1)	Defines the physical structure of the network and the topology.
Data link (Layer 2)	Provides error detection and correction. Uses two distinct sublayers: the MAC and LLC layers. Identifies the method by which media are accessed. Defines hardware addressing through the MAC sublayer.
Network (Layer 3)	Handles the discovery of destination systems and addressing. Provides the mechanism by which data can be passed and routed from one network system to another.
Transport (Layer 4)	Provides connection services between the sending and receiving devices and ensures reliable data delivery. Manages flow control through buffering or windowing. Provides segmentation, error checking, and service identification.
Session (Layer 5)	Synchronizes the data exchange between application on separate devices.
Presentation (Layer 6)	Translates data from the format used by applications into one that can be transmitted across the network. Handles encryption and decryption of data. Provides compression and decompression functionality. Formats data from the application layer into a format that can be sent over the network.
Application (Layer 7)	Provides access to the network for applications.

2.2 Network Devices

A network device is an individual component of the network that participates at one or more of the protocol layers (McCabe, 2007).They are required for communication and interaction between hardware on a computer network (Netwrix, 2019), which includes end devices, routers, switches, firewalls, hubs, modems, etc. These devices may be in a local network or internetwork. To put it another way, a network device is a node in the wireless mesh network. It can transmit and receive wireless HART data and perform the basic functions necessary to support network formation and maintenance (McCabe, 2007).

2.3 Types of Network Devices

There are different types of network devices used in a computer network which include the following:

- **Firewall** - is a network security device that monitors and either blocks or allows traffic based on a set of rules. Firewalls can be software, hardware, or a combination of both. Additionally, the rules that firewalls use can be based on something straightforward like ports and IP addresses or use heuristics to identify malicious behavior (iPass.Inc, 2021; Petryschuk, 2021);
- **Routers** - are the network devices that route packets between networks. These Layer 3 devices enable everything from communication between multiple subnets within the same WAN to the internet connection that allows you to read this article. A good way to think of routers is this: They are the network device that deals with IP addresses;
- **Switch** - The textbook definition of a network switch is a Layer 2 device that sends and receives frames. These switches are the basic building block of Ethernet networks. By sending the data to a specific device, the switch is breaking up collision domains and greatly reduces network congestion when compared to network hubs. That breaking up of collision domains is the basic benefit of a Layer 2 switch. However, this basic example of a Layer 2 switch is just one of the many types of network switches. Here is a list of common types of network switches:
  - **Unmanaged switches** - simply provide Layer 2 switching of Ethernet frames. They do not offer any additional management or configuration features;
  - **PoE switches** - switches that provide PoE functionality can provide both network connectivity and power to connected devices. For example, it is common to VoIP phones using PoE switches. And also it is common to have low power consumption cameras. PoE switches can be Layer 2 or Layer 3 switches and can be managed or unmanaged;
  - **Managed switches** - switches that vary greatly in their features and functionality. For example, some managed switches are targeted to gamers for use at home while others are targeted to large enterprises for use on corporate networks. One of the most important aspects of a managed switch is the ability to create VLANs. Other popular managed switch features include QoS to prioritize certain types of traffic and STP to prevent network loops. Managed switches can be either Layer 2 or Layer 3 switches;

- **Layer 3 switches** - these switches offer the same Layer 2 functionality as other switches, but add Layer 3 routing to the mix. Layer 3 switches are aware of IP addresses and can route packets between networks;
- **Stackable switches** - dome network switches can be "stacked." These stackable switches can be connected to one another to operate as a single logical switch. Stacking switches can be a useful way to increase the capacity of a network. For example, stacking two 24-port switches would create a single 48-port switch from a management and functionality perspective (Petryschuk, 2021).

Chapter 3

Market Overview

3.1 Industry Research

The global home networking devices market is foreseen to portray stable continual growth in all the regions around the world, driven by surging customer broadband penetration and increased network device adoption. By regulating various systems at home through network devices, end users can create a contented and gratifying environment, while decreasing energy consumption and aligned expenditures. This knack is expected to upkeep the revenue growth of the home networking devices. Consequently, market accomplices would pursue innovations that allow homeowners to integrate all their systems and reduce energy consumption.

The North American market is expected to portray a stable growth rate due to the presence of prominent manufacturers and maximum adoption of smart home systems utilizing full networking functionality. The European market is expected to grow faster than the global average in the coming years due to its debauched economic repossession. In the Asia Pacific region, the market is predicted to grow substantially due to increasing consumer income, higher technological adaptation, and increasing consumer awareness (GrandViewResearch, 2019).

The enterprise network equipment market was valued at USD 9.83 billion in 2020 and is expected to reach USD 15.48 billion by 2026, at a CAGR of 7.85% forecast period 2021 to 2026 (IndustryResearch, 2020)

3.2 Cisco MS-Series

Cisco provides port schedules feature in their MS series switches, the cheapest device example of that series would be the Cisco Meraki MS220-8 gigabit switch with layer 2 access switching, that has 8 PoE+ ports. The price starts from $434.90 on Amazon (Cisco, 2020a; Cisco, 2020b).

The "Port schedules" screen (Figure 3.1) shows how Cisco gives users control on scheduling should be mentioned that they offer users to use templates and show it on a separate diagram enabled ranges of each port.

FIGURE 3.1: GUI for scheduling on Cisco devices

3.3 Zyxel GS1900 Series

Zyxel provides port schedules (time-controlled) features in their GS1900 HP Series, the most congruent competitor would be the 8-port GbE Smart Managed PoE Switch. The 1900 Series consists of nine (9) models—the GS1900-8, GS1900-8HP, GS1900-10HP, GS1900- 16, GS1900-24E, GS1900-24, GS1900-24HP, GS1900-48 and GS1900-48HP. Providing GbE switches with power-saving functions. In addition, the PoE models GS1900-8HP/10HP/24HP/48HP Gigabit switch complies with the IEEE 802.3at PoE+. The price starts from $159.99 (Zyxel, 2019b; Zyxel, 2019a).

The "Time Range Group" setting (Figure 3.2) allows also the use of scheduling, but in that type of UI, users do not have access to create several schedulings on each port. It is only applicable for all ports or nothing, and scheduling jobs is only one.

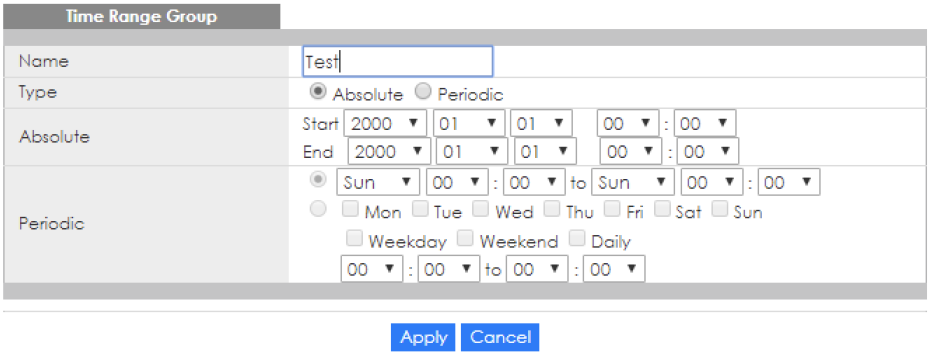

FIGURE 3.2: GUI for scheduling on Zyxel devices

Chapter 4

Hardware Overview

4.1 Device Set Up

The hardware part of this project consists of 3 modules: board with switch and stm32 (Section 4.2), Nano Pi (Section 4.3), and gsm module. The diagram (Figure 4.1) shows how modules are connected (with black lines) and how network cable is connected (blue lines) to devices. Also it represents how buttons work on board, and if a button is lighted up, that port is turned on, but if the light is off, the port is turned off. PC1, PC2, and PC3 represent connected devices, but they could be any other nodes or switches, etc.

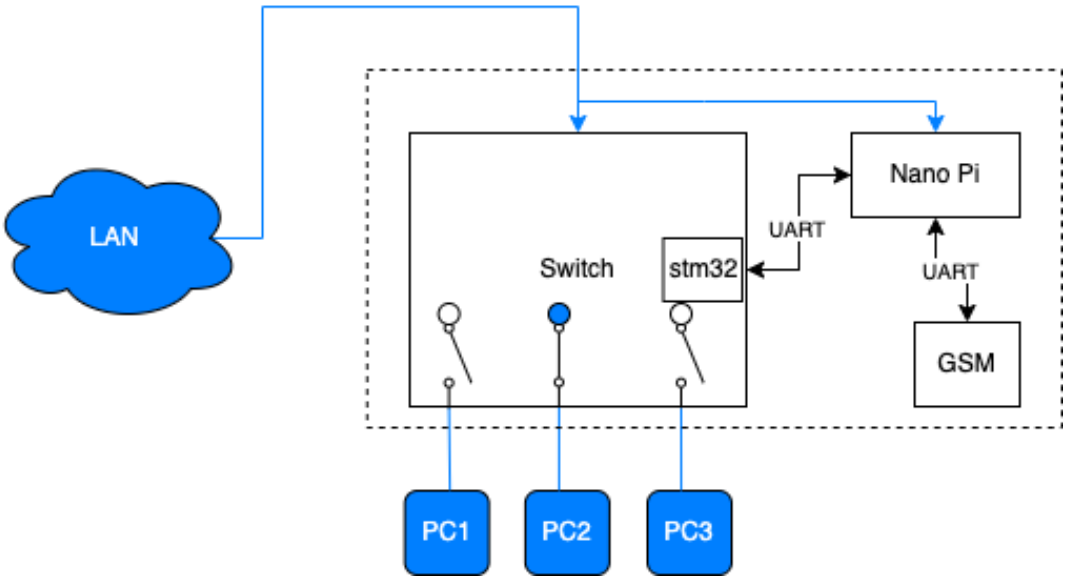

FIGURE 4.1: Device setup diagram

4.2 Switch

The board is manually developed based on layer 2 Realtek switch (Figure 4.2) RTL8367S-CG that has 3 ethernet ports and stm32.

The RTL8367S-CG is an LQFP-128, high-performance 5+2-port 10/100/1000M Ethernet switch featuring a low-power integrated 5-Port Giga-PHY that supports 1000BaseT, 100Base-TX, and 10Base-T.

For specific applications, the RTL8367S supports one extra interface that could be configured as RGMII/MII interface. The RTL8367S also supports one Ser-Des interface that could be configured as SGMII/HSGMII interfaces. The RTL8367S integrates all the functions of a high-speed switch system; including SRAM for packet

buffering, non-blocking switch fabric, and internal register management into a single CMOS device.

Short Features list:

- Single-chip 5+2-port 10/100/1000M nonblocking switch architecture;
- Embedded 5-Port 10/100/1000Base-T PHY;
- Each port supports full duplex 10/100/1000M connectivity (half duplex only supported in 10/100M mode);
- Extra Interface (Extension GMAC1) supports:
  - SGMII (1.25GHz) Interface;
  - High SGMII (3.125GHz) Interface;
- Extra Interface (Extension GMAC2) supports:
  - Media Independent Interface;
  - Reduced 10/100/1000M Media Independent Interface;
- Full-duplex and half-duplex operation with IEEE 802.3x flow control and back pressure (Realtek, 2019).

For debugging used Debug header port (Figure 4.3), and for updating MCU used the stm32 prog header (Figure 4.4) with link v2 to the USB adapter. Nano Pi and stm32 are connected by bus (Figure 4.5), that is for UART connection, and power supply for Nano Pi.

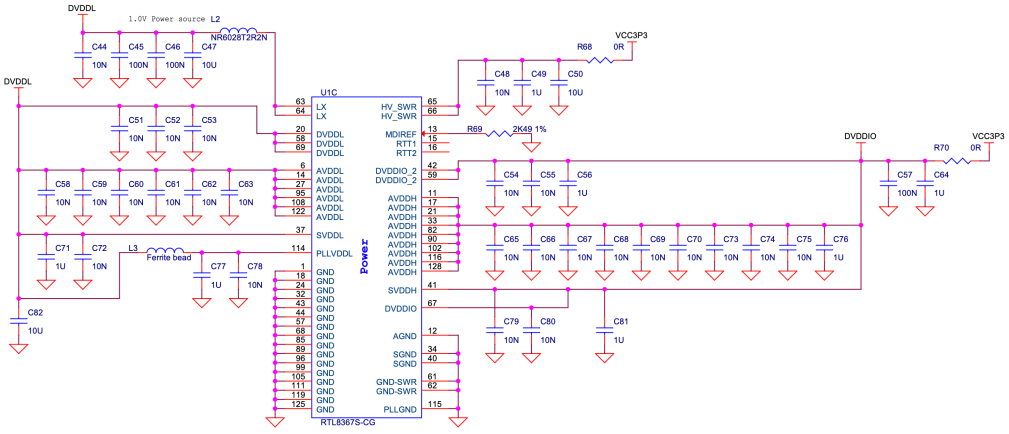

FIGURE 4.2: Switch pinout diagram

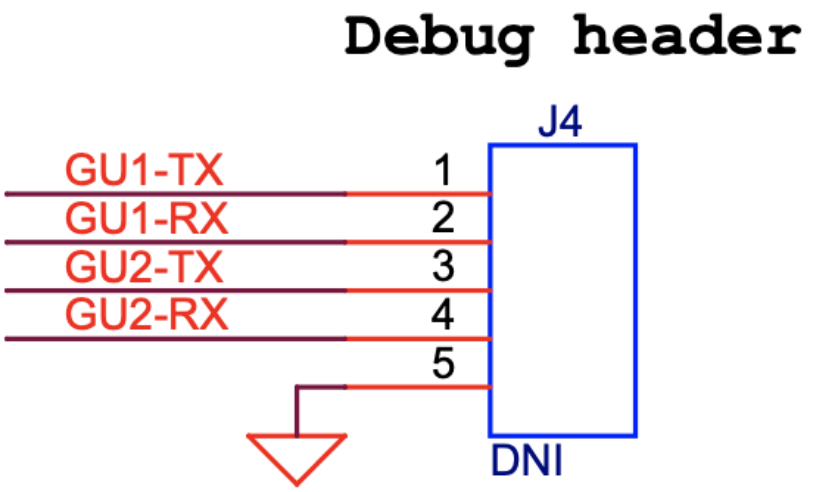

FIGURE 4.3: Debug headers diagram

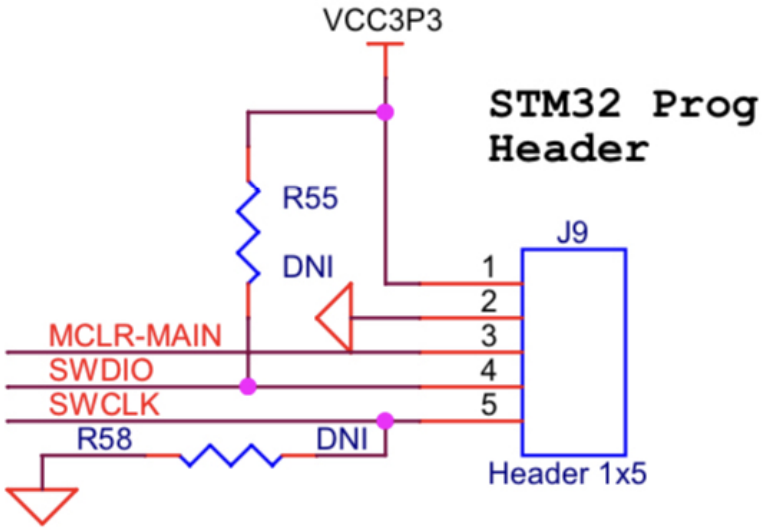

FIGURE 4.4: STM32 programmation header

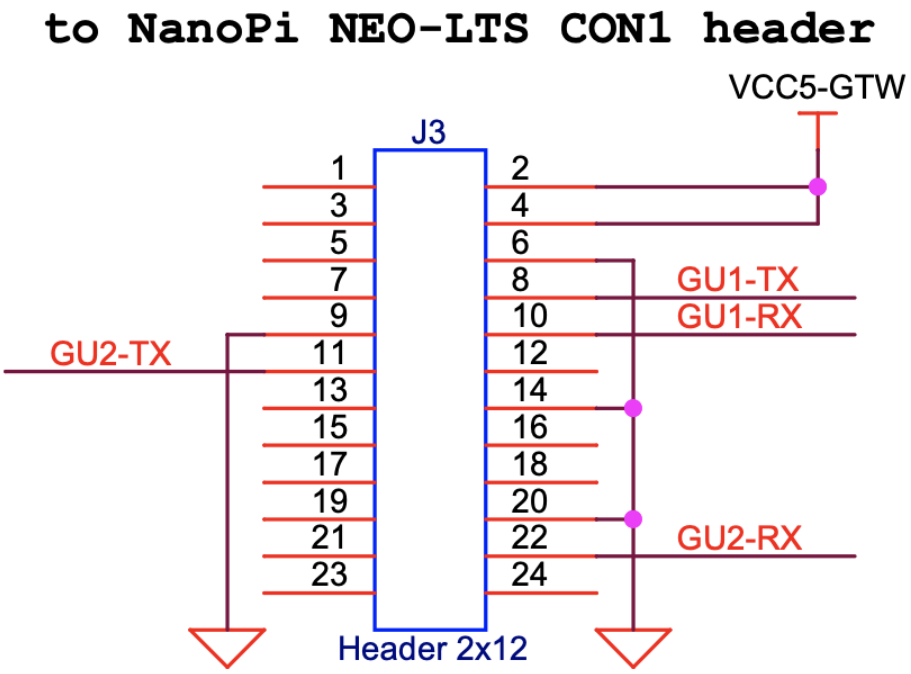

FIGURE 4.5: Nano Pi headers diagram

4.3 Nano Pi

NanoPI NEO (Figure 4.6) is a single-board system from FriendlyARM with an RAM memory of 256 MB. It is equipped with the Allwinner H3 system offering 4 Coretex-A7 cores. Each core can operate at a frequency of 1.2 GHz. In addition, the board has a Mali400MP2 graphics processor clocked at 600 MHz. The board is equipped with, among others, an RJ45 port, 2 USB ports (1 USB 2.0 port and 1 micro-USB 2.0 connector), a micro SD memory card slot, and interface ports such as USB (2 additional USB 2.0 ports), UART (2 ports RS232), I2C, SPI, PWM, GPIO and audio (microphone input and line audio output). The system is powered by 5V, and power consumption according to the manufacturer does not exceed 2A. The system's operation is responsible for the Ubuntu Core distribution, specially prepared by the manufacturer, based on the Linux kernel version 3.4. The system is not factory installed (kamami, 2016).

Nano Pi has as reference the Raspberry Pi Zero but it is faster and 12% smaller, and it is sold at about $7, so it is comparable with its benchmark board (Ruggeri, 2016).

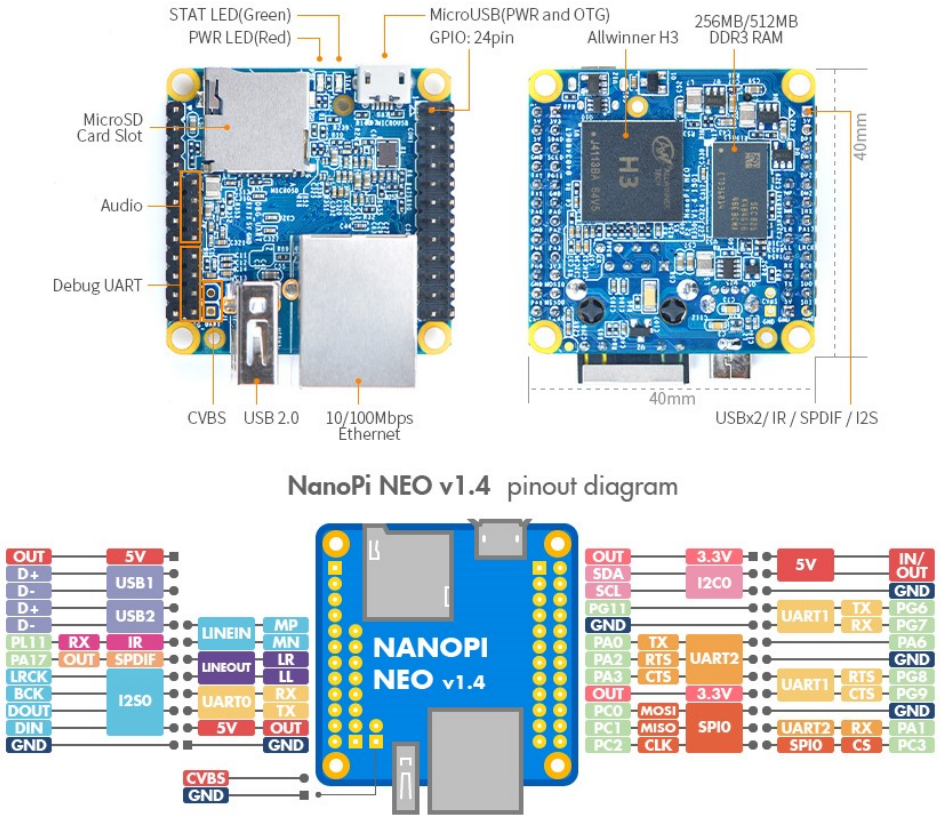

FIGURE 4.6: NanoPi pinout diagram

Chapter 5

Technology overview

5.1 Web Framework

There are two most popular types of web frameworks for Python: Flask and Django. While Django is a high-level python web framework, Flask is much easier to understand and learn. It is used for small applications compared to Django; therefore, it is more suitable for this project.

Flask is a Python microframework (Backend) that has the principle of having as little tech as possible to get a website up and running. It uses Jinja2 templating, is RESTful, and has a built-in debugger (Livingston, 2017). "Python", "Lightweight" and "Minimal" are the key factors why we considered Flask in this project. Along with Bootstrap and JQuery, it allows the development of simple lightweight web applications that are perfect for this type of project.

5.2 Database

In Flask applications, to manipulate databases, can be used SQL and Object Relational Mapping, the most popular and developed is SQLAlchemy.

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

SQLAlchemy is most famous for its object-relational mapper (ORM), an optional component that provides the data mapper pattern, where classes can be mapped to the database in open ended, multiple ways - allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.(SQLAlchemy, 2019).

The main reason for using the database in this project is to save authorization and scheduling data even after the device reboot.

5.3 Computing build framework

For the creation of modified custom builds with pre-installed scripts and modules for single-board computers, there is no other alternative than Armbian.

Armbian is a base operating system platform for single-board computers that other projects can be trusted to build upon (Armbian, 2015).

- Lightweight Debian or Ubuntu-based Linux distribution specialized for ARM development boards;
- Each system is compiled, assembled, and optimized by Armbian Build Tools;

- It has powerful build and software development tools to make custom builds;
- It is a vibrant community.

Armbian gives access to create custom builds, with custom scripts pre-installed for this. Using this, we pre-installed our project, closed all Nano Pi ports, and started several services on autostart. The first service starts flask on Nano Pi, and the other starts a python module that through UDP shares nano-pi's IP. In the output, we get an iso file that we can flash on any Nano Pi.

Chapter 6

Proposed Approach

6.1 Architecture

This project is not such a large one because when it just started, it has architecture from documentation of Flask (Flask, 2020), but as the project grew , it became overwhelming to keep all the code in such a hierarchy. We then moved it to another structure (Figure 6.1) with jinja2 templates and blueprints for reusing and easier developing (Soheili, 2020; Birchard, 2021).

```
├── wsgi.py                 (App starting point)
├── config.py               (Flask configurations)
├── managed_switch
    ├── __init__.py         (Flask application factory)
    ├── db_utils.py
    ├── models              (All database models)
        ├── __init__.py
        ├── network_configuration.py
        ├── ...
        └── users.py
    ├── routes              (Backend logic)
        ├── auth
            └── auth.py
        ├── ...
        └── terminal
            └── terminal.py
    ├── services            (3-rd party modules)
        ├── __init__.py
        ├── controller.py
        ├── ...
        └── scheduler.py
    ├── static
        ├── css
        ├── fonts
        ├── js
    ├── templates           (Templates jinja2)
    └── utils.py
├── requirements.txt
└── app.sqlite
```

FIGURE 6.1: Architecture of a project

6.2 Database Tables

The main tables in the project are: users table schema (Figure 6.2) and corresponding example of data (Figure 6.3) - saves all data for authorizations and also separates not privileged users from admin, ports table (Figures 6.4, 6.5) - that saves the status of each port, and jobs table (Figures 6.6, 6.7) which consist necessary data for scheduling.

```
create table users
(
    id            INTEGER not null primary key,
    is_admin      BOOLEAN,
    username      VARCHAR(100),
    email         VARCHAR(345),
    phone_number  VARCHAR(10),
    password_hash VARCHAR
);
```

FIGURE 6.2: Users table types and properties

```
+--+--------+--------+--------------------+------------+------------+
|id|is_admin|username|email                |phone_number |password_hash|
+--+--------+--------+--------------------+------------+------------|
|1 |1       |admin   |test.admin@example.com|+380999999999|260000$54... |
|2 |0       |user1   |test.user1@example.com|+380777777777|260000$Xg... |
|3 |0       |user2   |test.user2@example.com|+380555555555|260000$Hr... |
|4 |0       |user3   |test.user3@example.com|+380333333333|260000$Tr... |
+--+--------+--------+--------------------+------------+------------+
```

FIGURE 6.3: Users table

```
create table ports
(
    id           INTEGER not null primary key,
    port_number  INTEGER,
    active_state BOOLEAN
);
```

FIGURE 6.4: Ports table types and properties

```
+--+-----------+------------+
|id|port_number|active_state|
+--+-----------+------------+
|1 |1          |0           |
|2 |2          |1           |
|3 |3          |0           |
+--+-----------+------------+
```

FIGURE 6.5: Ports table

```
-- auto-generated definition
create table apscheduler_jobs
(
    id            VARCHAR(191) not null primary key,
    next_run_time FLOAT,
    job_state     BLOB not null
);

create index ix_apscheduler_jobs_next_run_time
    on apscheduler_jobs (next_run_time);
```

FIGURE 6.6: Jobs table types and properties

```
+-----------------------------------------------+-------------+---------+
|id                                             |next_run_time|job_state|
+-----------------------------------------------+-------------+---------+
|start-3101350801334785005250729673682641114 27|1654070400   |'8059...'|
|end-310135080133478500525072967368264111427   |1654079400   |'8059...'|
|start-3150052645976504596347530236066796557 47|1654338600   |'8059...'|
|end-315005264597650459634753023606679655747   |1654347600   |'8059...'|
|start-3221711666756806176184737371563496461 47|1654500600   |'8059...'|
|end-322171166675680617618473737156349646147   |1654016400   |'8059...'|
|start-3279688277015095607697588562949785408 67|1654174800   |'8059...'|
|end-327968827701509560769758856294978540867   |1654209000   |'8059...'|
+-----------------------------------------------+-------------+---------+
```

FIGURE 6.7: Jobs table

6.3 Application

This application needs only to be accessible from the local network, on any platform. But to reach that website, the user needs to know the IP address of the Nano Pi, which controls the switch. The popular solution is to nmap the local network and find a device that has a name associated with "Nano Pi." Another popular solution is to go to a specific IP address that is reserved for that device, but it works only

for routers. Cisco, in their products, recommends running their application on a Windows PC and connects by serial to switch.

But our solution is based on sockets, UDP broadcasting, first on autostart on Nano Pi starts UDP broadcast server program that sends it an IP by specific port, and the client program listens to that specific port and gets the IP address of Nano Pi.

6.3.1 Authorization

This project has two types of users: admin and regular user. The admin can add new users with email and phone; it is almost like family control. Access control Screen (Figure 6.8) shows forms that are optional and required to help create or edit user data. On the login screen (Figure 6.9), you can only be authorized using username/password flow, or use email or phone flow, but only if that email or phone number has already been added by the admin.

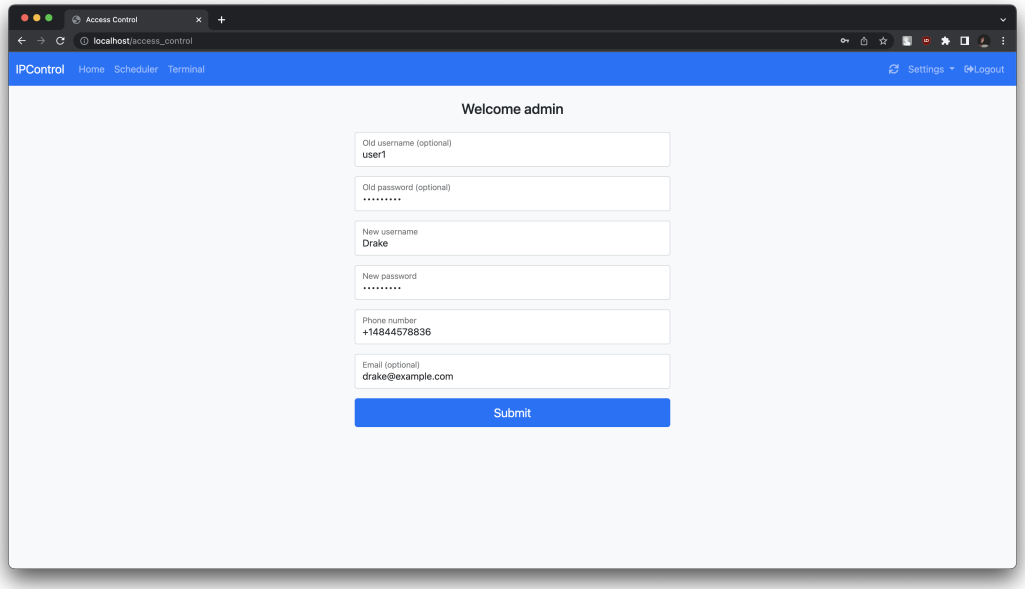

FIGURE 6.8: Access control Screen

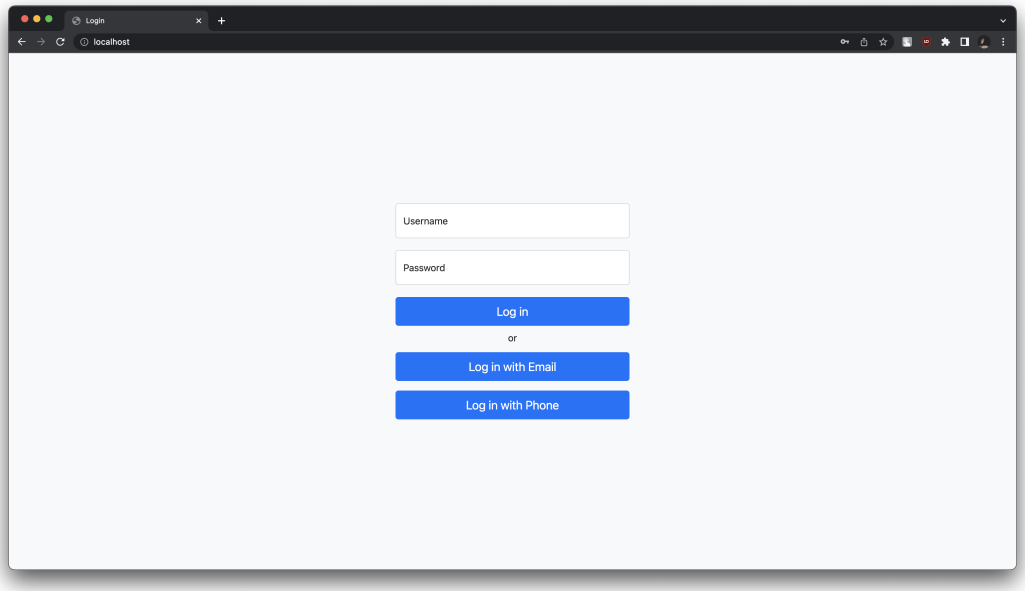

FIGURE 6.9: Login Screen

6.3.2 Home

On the home screen (Figure 6.10) users can check the status of ports, and turn them on or off. But if there was a disconnect where a physically lighted button differs from a web page, they have to press on the "Resync button" that is in the up-right corner of the page, in the navigation bar.

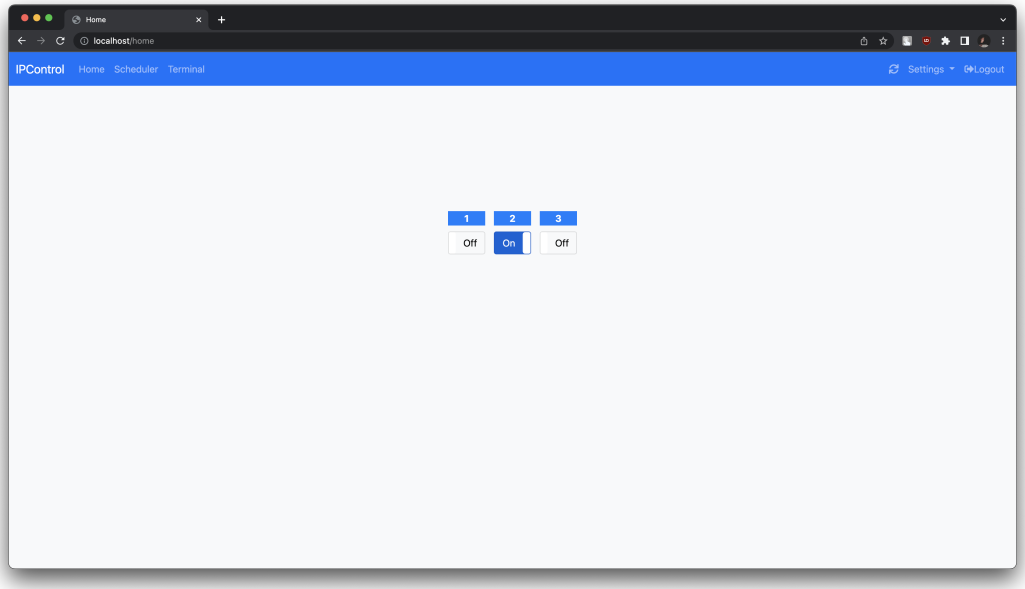

FIGURE 6.10: Home Screen

6.3.3 Scheduling

For scheduling, users can add a job or several jobs on the Scheduling page (Figure 6.11), those recurring jobs control the status of ports, and users can add any job on a

separate port. For example (Figure 6.11) on port 2 added two jobs:

- On working days the port is enabled only from 9:00 to 20:30;
- On weekends the port is enabled only from 21:00 to 23:00.

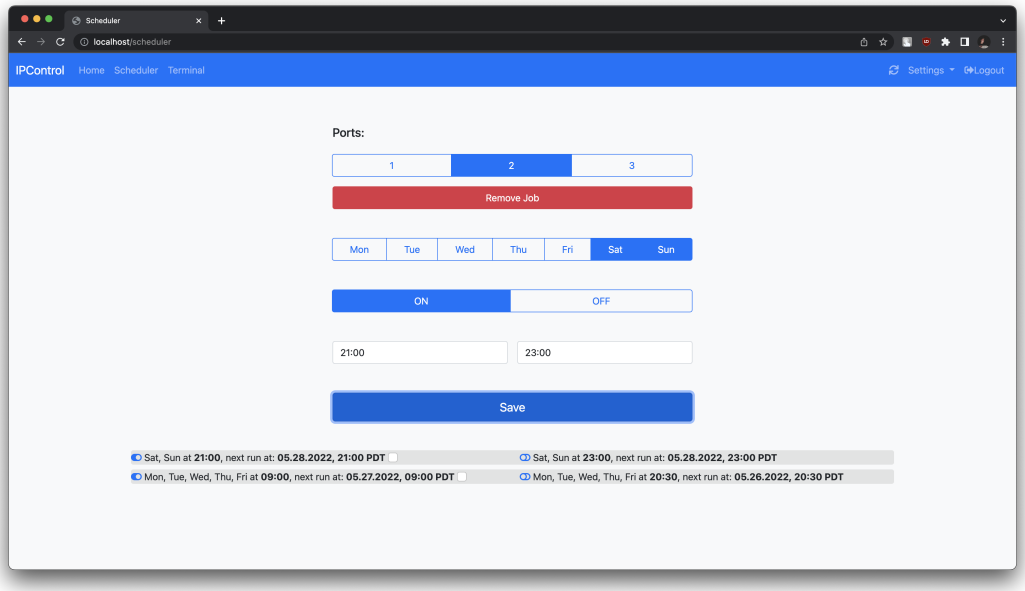

FIGURE 6.11: Scheduling Screen

6.3.4 Terminal

On the terminal page (Figure 6.12), a user can write commands that send to STM32 by UART, which is connected to the switch. For now, there are two types of commands, first, they tell stm32 to change port status on switch (//P**X**R**Y**) where **X** - is a port number, and **Y** is 1 or 0, to turn port on or off. The second type tells which ports are on (//Q1) or which ports are off (//Q0).

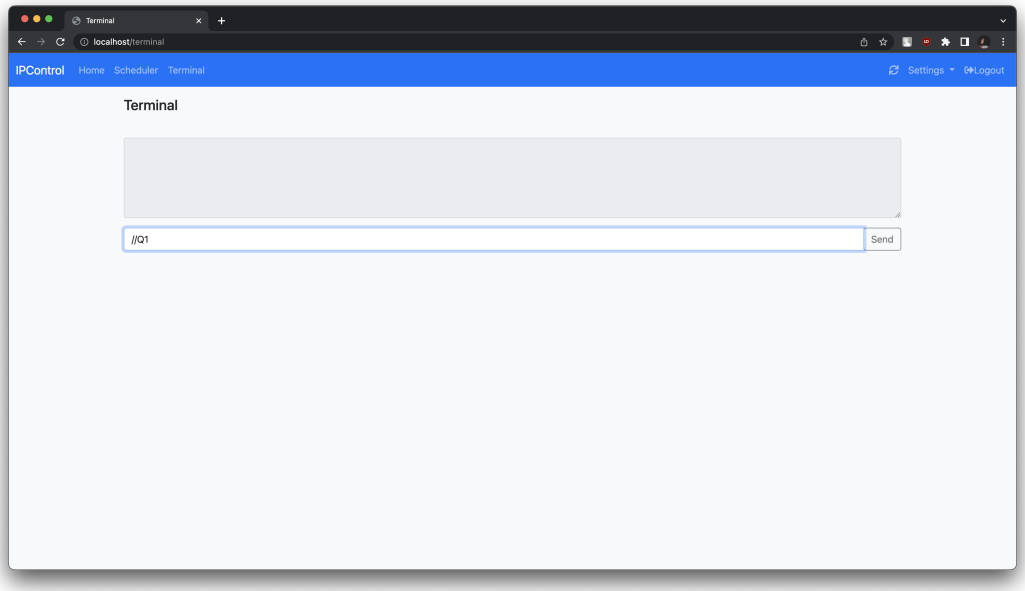

FIGURE 6.12: Terminal Screen

6.3.5 Settings

On the management screen (Figure 6.13), users can reboot Nano Pi if it is necessary. Also on the settings screen (Figure 6.14), users can change the number of ports that are on the switch, it is now for testing and in the future, and it would be replaced by the automated discovery of ports by Nano Pi.

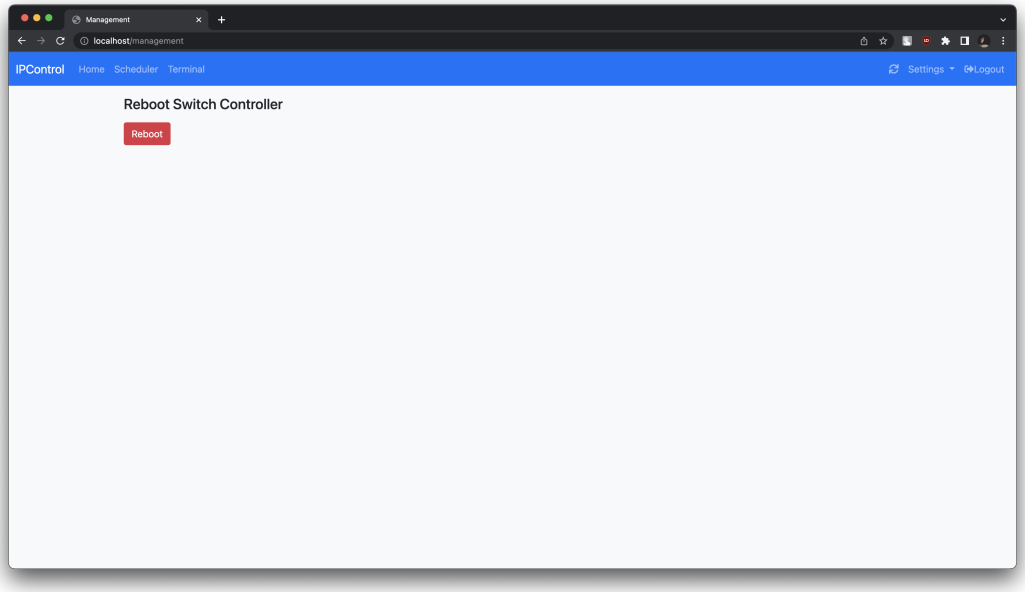

FIGURE 6.13: Nano Pi's management Screen

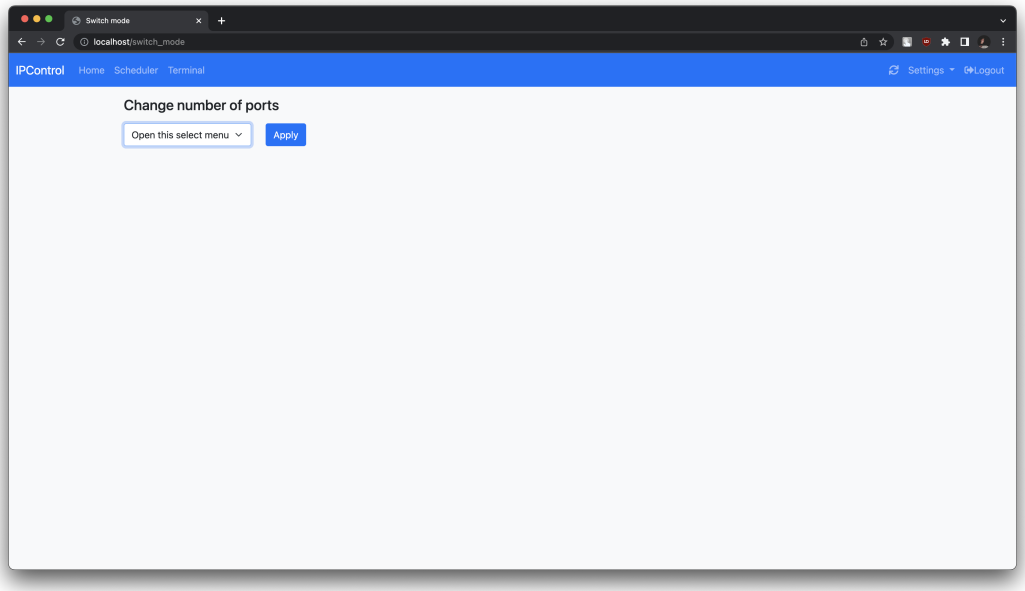

FIGURE 6.14: Switch Settings Screen

Chapter 7

Summary

In this work, considered how to provide software control of a multiport network switch. Also, this project covered the listed goals in the Introduction of this bachelor thesis, specifically:

- Configuration of the Operation system for Nano Pi, with taking into account security gaps that could be existed;
- Implementation of the web application for controlling and observing the switch ports.

This project required knowledge of Web development, IoT, Networking, Operation systems, and for good quality project maintenance - OOP. The current application is user-friendly and optimized for a good mobile experience. As a result, there is a complete device set up with a custom modified operation system based on Ubuntu Linux, with a web application that gives access for users to control that device by Web application.

7.1 Future work

- Improve the Hardware part of the project, replace the switch device with that one that has PoE+ ports;
- Add async services to always listen to manual button changes and update automatically webpage.

Bibliography

Armbian (2015). *What is Armbian?* URL: https://docs.armbian.com/.

Birchard, Todd (2021). *Organize Flask Apps with Blueprints*. URL: https://hackersandslackers.com/flask-blueprints/.

Cisco (2020a). *MS - Switches*. URL: https://documentation.meraki.com/MS.

— (2020b). *Port Schedules*. URL: https://documentation.meraki.com/MS/Access_Control/Port_Schedules.

Emmett Dulaney, Michael Harwood (2011). *CompTIA Network+ N10-005 Exam Cram*.

Flask (2020). *Project Layout*. URL: https://flask.palletsprojects.com/en/2.1.x/tutorial/layout/.

Froehlich, Andrew (2021). *OSI model (Open Systems Interconnection)*. URL: https://www.techtarget.com/searchnetworking/definition/OSI.

GrandViewResearch (2019). *Networking Devices Market Size, Share & Trends Reports, 2019 To 2025*.

Harry Reynolds, Doug Marschke (2009). *JUNOS Enterprise Switching: A Practical Guide to JUNOS Switches and Certification*.

IndustryResearch (2020). *Networking Equipment Market 2020 | Global Industry Trends*. URL: https://www.mordorintelligence.com/industry-reports/enterprise-network-equipment-market.

iPass.Inc (2021). *iPass Glossary*. URL: http://help.ipass.com/doku.php?id=glossary.

kamami (2016). *NanoPi Neo LTS 256MB single board computer*. URL: https://kamami.pl/en/nano-pi-board/578222-nanopi-neo-lts-256mb-single-board-computer.html.

Livingston, Matt (2017). "Full Stack Web Development". In: *Becoming Human: Artificial Intelligence Magazine*. DOI: https://becominghuman.ai/full-stack-web-development-python-flask-javascript-jquery-bootstrap-802dd7d43053.

McCabe, James D. (2007). *Network Device*. URL: https://www.sciencedirect.com/topics/engineering/network-device.

Netwrix, Ryan (2019). *Network Devices Explained*. URL: https://community.spiceworks.com/topic/2191428-network-devices-explained.

Petryschuk, Steve (2021). *What Are the Types of Network Devices?* URL: https://www.auvik.com/franklyit/blog/network-devices/.

Realtek (2019). *LAYER 2 MANAGED 5+2-PORT*. URL: https://www.realtek.com/en/products/communications-network-ics/item/rtl8367s-cg.

Ruggeri, Luca (2016). *NanoPi NEO – Smaller and faster than Raspberry Pi Zero*. URL: https://www.open-electronics.org/nanopi-neo-smaller-and-faster-than-raspberry-pi-zero/.

Soheili, Arash (2020). *Structuring a Large Production Flask Application*. URL: https://levelup.gitconnected.com/structuring-a-large-production-flask-application-7a0066a65447.

SQLAlchemy (2019). *The Python SQL Toolkit and Object Relational Mapper*. URL: https://www.sqlalchemy.org.

Zyxel (2019a). *PoE switch on / off by schedule for GS1900 HP Series*. URL: https://support.zyxel.eu/hc/en-us/articles/360011460199--PoE-switch-on-off-by-schedule-time-controlled-for-GS1900-HP-Series.

— (2019b). *Zyxel GS1900-10HP*. URL: https://www.zyxelguard.com/GS1900-10HP.asp.