Bachelor's thesis

Information Technology

2019

Timo Kemppainen

INDEPENDENT WIRELESS WEB SERVER AS AN INTERMEDIARY FOR SMART DEVICES

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2019 | 32 pages, 2 pages of appendices

Timo Kemppainen

INDEPENDENT WIRELESS WEB SERVER AS AN INTERMEDIARY FOR SMART DEVICES

Smart devices are prevalent in modern society, but their long-term usability can be hindered by software. Popular brands of smart phones are strongly linked to their manufacturers’ ecosystem. For example, software installation and updates are often handled through the device manufacturer’s or operating system developer’s application store. While a smart device’s strong integration into an ecosystem can provide a more convenient user experience it also can possibly result in loss of functionality if the relied services are unavailable.

The issue of smart devices relying on external services, and possible loss of functionality due to this, was sought to be solved by implementing a self-hosted, automatic, and autonomous intermediary device. For this purpose, a wireless web server operating on a Raspberry Pi was built to allow typical customer grade smart devices to retain their basic functionalities of file transfer, communication, and light productivity; using only Wi-Fi connectivity and the pre-installed web browser. To enable the execution of the beforementioned operations, web applications and services released under permissive licenses were installed on the web server. After implementing the intermediary device its viability was assessed through testing with smart devices. The testing proved the concept viable, although some minor issues were discovered. Some web applications could not be properly operated due to a smart device’s low resolution, and some integral application features did not work on the smart devices’ web browsers.

Overall the idea of using a self-hosted web server to serve web applications to smart devices was proven feasible. The minor issues encountered can be resolved through additional configurations made to the software or by using alternative software. Unfortunately, the intermediary device as a solution to the reliance of external services is not conclusive. One of the intended testing devices could not complete its initial set up, for it required a connection to the manufacturer’s server, which could not be established. This meant the device could not be used at all, and the intermediary device would not fix this issue. Nevertheless, so long as the smart device has a functional web browser and working Wi-Fi connectivity it can utilize web applications independently of external services.

KEYWORDS:

Consumer, Linux, self-hosted, smart device, software obsolescence, web server

OPINNÄYTETYÖ AMK | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2019 | 32 sivua, 2 liitesivua

Timo Kemppainen

ITSENÄINEN LANGATON WWW-PALVELIN VÄLILAITTEENA ÄLYLAITTEILLE

Älylaitteiden yleistyminen kuluttajien keskuudessa on ollut nopea prosessi. Muutaman vuosikymmenen jälkeen useimmat kuluttajat omistavat vähintään yhden älylaitteen. Yksi älylaitteiden vakio-ominaisuuksista on verkkoyhteys. Laitteet hyödyntävät sitä ohjelmien asennukseen ja päivittämiseen sekä palveluissa kuten pilvitallennuksessa. Tämä voi muodostua ongelmaksi, kun älylaitteet eivät enää pelkästään hyödynnä verkkopalveluita vaan ovat riippuvaisia niistä. Jos älylaite käyttää ainoastaan pilvipalvelua tiedostojen varmuuskopiointiin, varmuuskopiointi ei jatkossa onnistu, jos pilvipalvelu ei ole käytettävissä tai se lakkautetaan.

Opinnäytetyön tarkoitus oli pyrkiä ratkaisemaan älylaitteiden riippuvuus käyttäjän hallitsemattomista palveluista toteuttamalla itsenäinen, autonominen välilaite, joka mahdollistaa älylaitteiden peruskäytön itse hallinnoidulla web-palvelimella. Palvelimella toimii kokoelma web-sovelluksia ja palveluja, joita älylaitteet voivat hyödyntää selaimella. Välilaite rakennettiin asentamalla Raspberry Pi -tietokoneelle web-palvelin, ja palvelimelle halutut web-applikaatiot. Asennetut ohjelmat pyrittiin valitsemaan julkaisulisenssien sallivuuden perusteella. Lisäksi web-sovellusten tuli mahdollistaa älylaitteiden käytön tiedostojen siirtämiseen, kommunikointiin ja dokumenttien käsittelyyn. Kun valitut ohjelmistot olivat asennettu ja laitteen toiminta oli automatisoitu, sen toimivuus ratkaisuna ulkopuolisten palveluiden riippuvuuteen testattiin älylaitteilla. Tulokset osoittivat välilaitteen konseptin toimivaksi, mutta pieniä ongelmia ilmeni web-sovellusten käytössä. Älylaitteiden Internet-selaimet eivät toimineet minimalistisen wiki web-applikaation valikoiden käytössä, mikä teki sovelluksen hyödyntämisestä mahdotonta. Lisäksi yhden testatun älypuhelimen resoluutio ei riittänyt web-sovelluksen näyttämiseen näytöllä, vaikeuttaen huomattavasti sovelluksen käyttämistä.

Web-sovelluksissa esiintyneistä pienistä ongelmista huolimatta välilaite toimi asetettujen vaatimusten mukaisesti. Sovellusten viat on mahdollista korjata muokkaamalla asetustiedostoja tai valitsemalla muita sovelluksia. Välilaite on modulaarinen, joten jokainen hyödynnettävä web-applikaatio tai palvelu on mahdollista korvata. Lisäksi on mahdollista hyödyntää samoja tarkoituksia palvelevia ohjelmistoja samaan aikaan, joten vaikka yksi sovellus ei toimi kaikilla laitteilla, voidaan asentaa vaihtoehtoisia sovelluksia takaamaan laajemman älylaitetuen.

ASIASANAT:

Kuluttaja, Linux, WWW-palvelin, älylaitteet

CONTENT

APPENDICES

LIST OF ABBREVIATIONS

ARM	Computer processor architecture
COTS	Commercial off-the-shelf
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GPL	GNU General Project License
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
LAMP	Linux, Apache, MySQL, PHP. Popular package of software used for web servers
LAN	Local Area Network
MIT	Massachusetts Institute of Technology
PHP	Hypertext Preprocessor
SSID	Service Set Identifier, commonly referred to as the Wi-Fi network name
URL	Uniform Resource Locator
VPN	Virtual Private Network

1 INTRODUCTION

Nowadays, owning at least a single smart device is rather the rule than the exception. Statista estimates the number of smart phone users in 2018 equaled to over two and a half billion phones. [1] The number of smart devices and internet connected devices is already massive, and it is only expected to grow. In his 2016 published article, Forbes contributor M. Kanellos points out estimations demonstrating there could be up to 30 billion Internet-connected devices come 2020. [2] Every year there are new models of smart devices released, but it is not that the old devices will not immediately disappear. The support the older devices will naturally decrease as the smart device manufacturers focus their resources on the newer devices. Eventually the old devices are expected to become officially unsupported, or they will end receiving any updates and remain in a state of permanent limbo.

Many electronical devices' lifespan is no longer dictated by the hardware alone. These days modern electronical devices rely increasingly more on sophisticated software for usability. This complex software always has some room for better optimization and bug fixes, therefore updates for these devices are both frequent and desirable. Since IoT- and smart devices are designed to be connected to the Internet, if not continuously, then at least on a regular basis. The distribution of software updates is handled primarily through the Internet, relying on the device manufacturer's, the software developer's, or third-party service provider's services. Additionally, the usage of the software itself may be contingent, either partially or entirely, upon services outside of the user's control, good examples being the reliance on cloud services for computing and storage, and software installation being only possible from the product developer's service. In conclusion, software functionality plays an ever-increasing role in smart device's general operability.

It becomes an issue when software relying on not-user-controlled services is no longer supported by the service provider. This could be due to the software becoming outdated and then it is no longer allowed access to the service; or if the service is either temporarily or permanently unavailable. A device could lose parts or maybe even its entire functionality due to software becoming obsolete, even if there were no issues with the device's hardware. Software obsolescence is not a new concern. In 2006 L. Merola researched the possible risks of COTS software obsolescence in the United States' Department of Defense, finding there is no proper contingency plan to prevent it, but that departments will deal with the possible issues reactively as they appear. [3] Research and studies regarding software obsolescence are frequent: in the oil and gas industry [4], in military and defense [3] [5], and in general [6]. While the matter is being actively investigated and researched, it is often done from the perspective of specific industry, and not from the point of a view of an average consumer, and thus, less relevant to a consumer. Though recognized and research issue, software obsolescence is often not directly looked upon from the perspective of private individuals.

This thesis approaches the issues of software obsolescence and a smart device losing its functionality due to unavailability of not-user-controlled services from the viewpoint of a consumer. The main focus being in turning a Raspberry Pi, a single-board computer, into an independent wireless web server that allows access devices, typical household smart devices i.e., smart phones and tablets, and computers, to connect into the intermediary server and retain the devices usability for typical usage: transferring data, communication, and light productivity; using only Wi-Fi connection and the device's built-in web browser. Furthermore, certain restrictions are placed upon the project. First, the intermediary device, after initial set-up, must not be dependent upon Internet connection or other external services. It must be able to operate independently. Secondly, the access devices should not require any modifications to be made or any new software being installed, as would be the case if they no longer had Internet connection or could use the smart device's respective application store. The only usage requirements placed on the access devices are working Wi-Fi connection and a web browser.

Additionally, some less strict guidelines are placed on setting up the intermediary device. A preference is being shown to software released under permissive licenses such as GPL and MIT-license. Therefore, the intermediary device will be running Linux as its operating system with Apache and PHP as additions according to the popular model of LAMP-web servers. Due to the purposes of the web server some important considerations are omitted from this project. Concerns regarding security and hardware capability are ignored unless they significantly affect usability. The final product is meant to be a proof of concept instead of a device used in production. Finally, in order to reduce unnecessary registrations and logins, software and services that require such things are less likely to be included. The end goal is an independently operating intermediary device which allows smart devices to be utilized even after their official product lifespan has ended, or the devices' required services can no longer be used.

There have been projects and studies which have resulted in systems comparative to this thesis' planned intermediary device. These self-hosted solutions often have different focus or purpose compared to this thesis' product. The projects' purposes vary from cloud servers: sovereign and Syncloud, to media servers like OpenMediaVault. Many are meant to have an Internet connection but due to their free open source nature it is possible to use them as independent Wi-Fi servers. Problems may arise if additional software is installed or configurations are done due to possible conflicts with pre-existing configurations or built-in scripts overwriting manually added configurations. Overall, these existing projects are a viable alternative to completely self-implementing a server should the server's purpose match with the self-hosting solution's goals and software selections. In the end, for this thesis a self-implemented server was chosen primarily in order to learn more about installing and configuring an independent Wi-Fi server with desired software.

2 RISK OF OBSOLESCENCE THROUGH RELIANCE ON EXTERNAL SERVICES

2.1 Smart device ecosystems

The most popular smart devices are strongly linked to their respective ecosystems. These ecosystems act as all-in-one solutions for the smart devices offering diverse and easy to use collection of services. For example, Apple has its own enclosed ecosystem accessed by single Apple ID allowing access to featured services and software offered and maintained by Apple. The Apple ecosystem allows for data synchronization and easy interoperability of all Apple smart devices such as iPhone, iPad, iMac, and Apple Watches through using a single account for login with all previously mentioned devices. This results in a streamlined user experience where the user has a single service provider that handles virtually all the support, and distribution of software and services. But on the other hand, this comes at the cost of strong integration into a service the user has limited control over.

Google’s Android is a more fragmented ecosystem compared to Apple’s. Even though Google products and services are commonly an integral part of Android devices, they are not an absolute necessity. Android is known to be a more open natured: software using permissive licenses, being open source, and not forcing complete reliance on services. This openness makes it possible for phone manufacturers and open source communities to implement their own customized variations of the operating system. For instance, by default Android uses Google Play Store for installing and updating applications, but most Android variations allow for sideloading an application from a package file; and it is possible to install additional application stores as is done by the phone manufacturer Samsung with Galaxy Store. It should be noted that the reliance on Google’s services is still noteworthy, most community created Android variants are dependent on Google services to a degree, for it would take significant development time to completely replace Google services.

Smart device ecosystems do provide value for their users, but not without uncertainties and potential issues. Having multitude of services accessible through one account across multiple devices with automatic data sharing is an immense quality of life improvement, but this is assuming the services work flawlessly uninterrupted. In a worst-case scenario, where all services in the ecosystems were suddenly shut down and abandoned, the devices with more open systems would likely fare better. For example, if a user wanted to install applications to a device: most Android devices require the user to only modify one specific setting: For comparison, on iPhone you need to download and install a developer tool to your device, officially only available from Apple, after. [7] Another potential issue is third-party application's relying on ecosystem services. By the virtue of being integral services on the device, third-party developers are prone to support only those services. This could result in a domino effect where linked applications become obsolete when one service is terminated. To summarize, services and software functioning together can provide better user experience it can also result in critical dependency, which could result in loss of function.

2.2 End of Windows Phone lifecycle

A product or service being manufactured by a well-established international industry giant is no guarantee that it will continue to be supported for a long period of time. An apt example of this is the case of Windows Phone. The lifespan of Windows Phones lasted nearly a decade. Starting from 2010 with the release of the first phones and ending in late 2019 when the support for the last Windows 10 Mobile phones will end according to the official support page [8]. To clarify, the end of support specifically refers to the end of system updates and security patches. Application store, third-party application support, and system services such as system restore may continue after this date. Third-party application support is dependent on the application's developer. The fate of Microsoft's application store and their other services is unknown. The end of support page makes no conclusive statements and declares the services may continue for a time after their official end dates. [8] In the end, Windows Phones never managed to reach a significant market portion of smart phones therefore Microsoft decided to end their phone line as an unsuccessful endeavor.

One Windows 8.1 phone, Lumia 820, is available for the use in this project. Overall, the device is in a perfectly serviceable condition. The hardware appears to be in adequate shape: the screen is not damaged, battery life does not seem to have degraded significantly, and the device does not suffer from overheating or crash. On the other hand, the software side has some problems. In normal daily use, there are no issues. Communicating with contacts works normally, web browser is operational, sending and receiving emails causes no issues, and the application marketplace is operable. The problems appear when the device tries to use built-in Microsoft hosted services such as creating device backups. Unfortunately, the error messages are not comprehensive, consequently it is unclear whether the failure of the operations is due to Microsoft axing their backend or if something is wrong on the device side. Similar problems happen with applications downloaded from the application store. Trying to use a first-party application to create a backup of saved contacts results in a failure with the application giving a generic error message. The Lumia 820 is a perfect example of a smart phone that has still functional hardware but where on the software side problem are starting to accrue.

While the cause of the application's occurring errors is not obvious due to ambiguous error messages, it is to be expected future issues will occur as Microsoft and third-party developers end their support for Lumia 820's software. Significant portion of the fault lies with how locked in the Windows Phone ecosystem is. The devices are designed from the inception to be used extensively with Microsoft's own ecosystem; support of third-party services is secondary. Lumia 820 uses OneDrive based cloud for device backup with the back end of the service being on Microsoft's servers. There is no support for other popular cloud services such as iCloud or Dropbox. Therefore, if Microsoft's back end is not available to be used, device backup is not feasible. Assuming the backup process is as simple as collecting a specific group of files into a package, and restoration is handled through unpacking the package; by allowing users to manually export and import the package file, the problem could have been avoided in entirety.

Microsoft has made some questionable design choices with Windows Phones. If a user could create a local backup file on their device, they could not manipulate the file directly and upload it without relying on the device's integrated feature which is locked into Microsoft's online storage. Furthermore, by default Lumia 820 does not come with a first-party file browser installed. It is only available as an optional addition from the official application store. Same is true for another common feature: contact backup. Generic install of Lumia 820 does not have a dedicated application or feature to mass backup contacts into a file. The designed method for contact back up is done through synchronizing contacts to a cloud or email account. Both backup methods are reliant on external services, although, installing the file browser application only requires one-time connection to the application store. But then again as previously mentioned, the future of Windows Phone application store is uncertain, and it is currently the exclusive source for application installation. Windows Phones do not support sideloading applications without unlocking developer options. If the application store for Windows Phones is taken down there is no obvious way for users to acquire and install additional applications. In conclusion, it is possible many features of the Lumia 820 will become entirely obsolete should Microsoft eventually shut down these exclusive services.

2.3 Video games and consoles

Video games are a fascinating case of software. They can retain their relevance throughout years for they are products with artistic and entertainment value, therefore a video game's life span is not necessarily fixed. Decades old video games can still have small but active communities. A relevant aspect of this is the that some old games still have active multiplayer communities even as the official multiplayer servers and master lists have been permanently disabled. Because Internet connectivity was not always taken for granted, and bandwidth costs were more of an issue, game developers often implemented support for multiplayer through LAN and directly connecting by IP. This means multiplayer functionality had to be available even without Internet connection, consequently the developers could not implement a system where multiplayer would always require a constant access to a service maintained by the developer. So, when a game has multiplayer functionality through LAN this allows the players to use VPN technology to make their devices appear to be in a single local area network, permitting them to use multiplayer functionality regardless of normally mandatory Internet services.

LAN support, a simple additional feature in a game, has ended up being an integral part of allowing old game's multiplayer communities to thrive.

In more modern video game titles LAN support has become less prevalent. With the release of *StarCraft II* Blizzard Entertainment decided to forego LAN support for the title, even though, the game's previous version *StarCraft* had full LAN support. The reason for not implementing support for the technology was a desire to integrate customers to Blizzard's Battle.Net technology and in order to combat piracy. [9] A Blizzard spokesperson also stated the popularity of their LAN supporting titles amongst pirates utilizing LAN was a deciding factor for the choice. [10] It is possible other companies share similar concerns and not implement LAN support in their products. This would make current games entirely reliant on external services for multiplayer functionality.

Video game consoles are focused computing devices meant for primarily one purpose: playing video games approved by device manufacturer. They are strongly tied to the manufacturer's services. A console developed by Microsoft mainly uses software developed by Microsoft, publishing a video game for the console requires it goes through a process overseen by Microsoft, and the online functionalities rely, at least partially, upon Microsoft's services. Some of these functionalities now serve same needs which were once handled manually. For example, save game data is one of the more typical user-produced data which the user may wish to back up. Formerly this was done using proprietary memory units usually referred to as memory cards, but with technological advancements and changing of standard, USB-based storage solutions became standard. Finally, with the strong integration to online services an automatic cloud save synchronization can be used in addition to manual backing up. An oddity in this fashion amongst the latest console generation is the Nintendo Switch. The only way to transfer or back up user save game data is from one device to another, which removes the data from the source device, or in cloud storage included in Nintendo's paid online service. Currently Nintendo Switch does not have any support for local manual save data backups. [11] For comparison. Sony's PlayStation 4 supports both: cloud synchronized save data, requiring a paid subscription to the online service; and manual local backups using USB-stick or hard drive. [12] Nintendo has decided to entirely remove the support of a tried and true backup method offering its users instead a cloud-based storage solution requiring a paid online subscription.

2.4 Concerns and possible solutions

There seems to be a trend in technology industry where devices due to their connectivity rely extensively upon external services. This does not equate into a problem by itself, the quality of life improvements and interoperability of software on multiple devices allows for a better user experience, but the problems of the reliance may prove to be severe. Loss of connection or unavailability of the service may prevent user from using a basic function of the device, as is demonstrated by Lumia 820 being unable to back up phone information to Microsoft's OneDrive. It is likely the trend will continue in the future and that strong reliance on external services will be a common feature.

Devices and software do not need to be entirely dependent on services beyond user's control. Android devices, while often strongly reliant on Google's services, can use alternative services and software. Applications don't need to be installed from Google's own application store; users can install applications manually or install alternative application offering platforms on device. Similarly, PlayStation 4 allowing to manually back up data on USB device, and offering cloud storage as paid premium service; a device can offer both choices: permitting independent user-controlled solution, and a convenient easy-to-use service. Even if the cloud storage is unavailable, temporarily or permanently, the user can still save to a local storage device, guaranteeing functionality in long term. It is especially perturbing when developers seem to deliberately forego common user-controlled technologies in favor of further integrating the software to not-user-controlled services as was the case with Blizzard Entertainment. By not providing LAN support in video games and by relying entirely on Blizzard's services the future of the game's multiplayer functionality is unclear. Now, it is possible the functionality could be added retroactively as the product becomes less popular, but a company may not desire to spend time and resources on a phasing product. In the end, forcing software or device to rely entirely on external services seems rather shortsighted, by allowing user-controlled solutions the avoidance of early product obsolescence can be greatly improved.

The most obvious solution to try alleviating hardware obsolescence by software obsolescence is to use software which does not rely entirely on external services or the services it uses have alternatives or are user controlled. The issue is devices are open to widely different degrees: the most open devices fully support installing alternative operating systems, some have operating systems that allow full control over software installations, others only allow software installation from developer or manufacturer approved sources, and finally you have devices that entirely locked down and user has almost no control over software except for usage. It is often possible to bypass limitations on many devices and custom software can be installed by non-supported methods such as through exploits. This likely falls in the legally grey area as it may break the device’s terms of service, and there is not recommended. Nevertheless, even the locked down devices can possibly be utilized through inventive software usage. As previously mentioned, older video games can still be played over Internet despite official servers being shut down. It is unclear whether it was intended to allow multiplayer over Internet by combining LAN support with VPN technology, but it does work. A question arises: is it possible to utilize locked down systems with just the device’s featured stock software?

Web browsers are one of the staple applications present in most smart devices. A web browser’s function is mainly to display a web page served from a web server. Most of the workload is carried out by the web server: it hosts the files and usually handles the scripts; this data is then downloaded by the client and displayed on the browser. Since most web pages are universal, accessible from virtually all operating systems and web browsers, even a locked down device should be able to access a website so long as it has a web browser. Therefore, by implementing a web server that serves diverse assortment of web applications, a locked down device could, regardless of its other limitations so long as it can connect to the server, utilize applications that would not otherwise be available to be used. The result would be a locked down device able to use wide variety of applications so long as Wi-Fi connectivity and the browser are functional.

3 INTERMEDIARY DEVICE

3.1 Design

The aim of this thesis project is to attempt alleviate old devices' lifespan limitations due to software obsolescence, by implementing an intermediary device which allows devices to be utilized for general tasks using Wi-Fi connection and a generic web browser. Due to World Wide Web's popularity, most Internet-connected devices come with web browsers, thus, as even the more closed-up systems and devices normally feature a web browser it should be possible to utilize the wide variety of available web applications to use a locked-in device for normal tasks independently of the manufacturer's software and service restrictions. The intermediary device will be an independent, consumer-controlled Wi-Fi server running a variety of web applications and services to allow access devices interconnectivity and to be able to perform tasks using a browser.

The requirements for accessing devices are deliberately few. The goal is to support a widest possible assortment of devices, though with minimal adjustments. The only prerequisites on the devices are working Wi-Fi connection and a generic web browser with support for ordinarily used technologies such as JavaScript. Simulating a scenario where most of the services used are rendered nonfunctional apart from the browser. No significant adjustments should be necessary for the accessing device the intentioned use scenario requires the user only to connect to the Wi-Fi, open browser and access the server, and start using the services. While it is true significant improvements can be made to device lifespan through installation of a custom operating system or software, that goes beyond the focus of this thesis.

The intermediary device's role can be considered two-part. First, the device oversees all networking: it handles all network traffic, operates as the wireless access point, serves as both the DNS and DHCP server. When an access device accesses the wireless network through the SSID broadcasted by the intermediary device, the access device will receive an automatic configuration for all networking needs without requiring user input. Second, the intermediary device acts as the web server. It hosts and serves all the implemented web pages and web services retrieved and operated by the access devices. The intermediary device should allow the access devices to upload and download files, communicate amongst the access devices, and light productivity.

The choices of hardware and software used in this thesis are not due to firm deliberations. All hardware used are chosen solely due to their availability. On the other hand, the software choices are made in part that the software is released under permissive licenses. Additionally, ease of access is a critical consideration for the software. Applications and services requiring the access device users to register an account or to login are not preferred. Furthermore, it should be emphasized the intermediate device is designed to be a proof-of-concept and not a product meant for production use. Therefore, substantial omissions are made in the design of the device. Minimal emphasis is placed on security and it will be intentionally foregone in order to improve ease of use.

After the implementation of the intermediary device, its viability will be verified. Assortment of devices will be used to access the intermediary device and the hosted software. The purpose is to ensure all devices can perform the predetermined variety of use scenarios: file transfer, chatting, and productivity.

3.2 Device and software choices

The intermediary device is based on Raspberry Pi 3 Model B. It is a popular single-board computer used for wide variety of different projects from acting as a network device to as a learning tool about programming and computing. As the board contains an in-built wireless networking chip no wireless adapter is required. For storage Raspberry Pi requires a MicroSD-card, and a 64-gigabyte MicroSDHC-card is used.

Permissive software licenses such as Apache, MIT, and GNU General Project License allow users to freely use, modify, and distribute the software; the only requirement usually being the license is attributing the original licensed code to its developers. [13] Therefore, as preference is shown to software released under permissive license, the device foundation is built around LAMP, which stands for Linux, Apache, MySQL, and PHP. LAMP is a popular software package used for web servers where Linux servers as the operating system. Apache receives web requests such as HTTP from clients and forwards the request according to configurations. MySQL is used for database management. Databases generally used in web servers to store and manage user account data. Closing with PHP, which is a programming language used to generate dynamic websites.

Raspberry Pi’s manufacturer Raspberry Pi Foundation also develops an operating system Raspbian. It is a derivative of the Linux distribution Debian; and is like another Debian derivative Ubuntu. It therefore benefits from the wide shared documentation of Debian and Ubuntu. Because Raspberry Pi’s developers have made a device specific variation of Linux, Raspbian is chosen as the Linux distribution used for this project. There are a few variations of Raspbian differentiated by the default software included in the image. This project uses the lite version which has bare minimum software pre-installed. In order to implement necessary networking functions hostapd is chosen for implementing wireless access point; and dnsmasq is installed to manage DHCP and serve as a DNS server. Furthermore, latest versions of Apache and PHP available from Raspbian’s repositories are used.

A great number of possible web applications exist which fulfill the previously stated sought services of the web server. Further defining of desired features for services was required, an exception being chat service, which is implemented with a simple PHP-based chat application Chatr. For uploading and downloading, two ways of file sharing were looked-for. First, simple way to upload a data, and to view and download already uploaded data. Second way being a more private and directed implementation where uploaded files would be given a unique URL address. A simple way of uploading files is implemented through a PHP based web page that allows users to pick a file from their device and upload it. Viewing and downloading already uploaded data is handled by h5ai, a web server file indexer that allows to preview various media files, and supports users choosing multiple files to download simultaneously. Finally, for the private file sharing, Jirafeau is chosen. Jirafeau provides a unique URL for each uploaded file and it has additional features such as download expiration time and password protection for uploads and downloads.

Productivity, as used in the context of this thesis, means for purposes of work and study. The intention is to use web browser applications for simple work tasks, mostly writing, preferably with support for collaboration. This is realized with CryptPad, which is a cloud-based office tools suite akin to Office365 and LibreOffice. It has variety of different applications: text editor, presentation tool, poll tool, and whiteboard for drawing. Each application supports collaboration and features integrated chat. In addition, a more minimalist and simpler wiki-type web application, cowyo, is installed for writing and making to-do lists. It also features password protection and deletion after reading for pages.

4 SETTING UP THE INTERMEDIARY DEVICE

4.1 Initial set up

The chosen Linux distribution Raspbian Stretch Lite requires little user choices during installation. It can be installed using a MicroSD-card preloaded with the system image or the NOOBS -operating system installer. Raspbian will be installed upon the MicroSD-card storing the system image or installer. After the installation basic system settings: locales, time, and language options; can be configured within the system using Raspberry Pi Software Configuration Tool. No changes should be made to network options, especially wireless options, as they will be configured manually later.

Preliminary set up of the intermediary device and the software installation require Internet connectivity. It is recommended a wired connection is maintained throughout the set-up process allowing to freely configure the wireless network interface without disturbing the necessary Internet connection.

Before installing individual packages, it is good practice to ensure the system and the repository information are up to date. It should be noted, in this thesis, lines beginning with the symbol *$* designate a command entered into terminal. Meanwhile, the symbol # is followed by the path of a configuration file, lines subsequent to the symbol indicate lines included in the configuration file.

```
$ sudo apt update && apt upgrade –y
```

After the possible updates have completed, all the necessary packages and their dependencies, which are available from the official repositories; are installed.

```
$ sudo apt Install dnsmasq hostapd –y
$ sudo apt install apache2 php git git-core npm nodejs –y
```

4.2 Network configuration

4.2.1 Wireless interface

The device’s wireless interface is configured through dhcpcd, a DHCP client program included in the base system. The configuration file can be accessed with command-line text editors. For the purposes of this project, the preinstalled text editor Nano is used. A static IP address is assigned to the interface, additionally the daemon is instructed to not use the default wireless supplicant.

```
$ sudo nano /etc/dhcpcd.conf
```

```
# /etc/dhcpcd.conf
interface wlan0
static ip_address=172.16.1.1/24
nohook wpa_supplicant
```

4.2.2 DNS and DHCP server

The previously installed dnsmasq is used to configure DNS and DHCP server settings. The original template configuration file is deleted and a new one is configured from the ground up.

```
$ sudo rm /etc/dnsmasq.conf
```

The DNS service is configured to listen the loopback and the wireless interface for requests. Meanwhile, the wireless interface is configured to have a pool of nineteen IP addresses available to lease for the duration of two hours.

```
# /etc/dnsmasq.conf
listen-address=127.0.0.1,172.16.1.1
interface=wlan0
dhcp-range=172.16.1.2,172.16.1.20,255.255.255.0,2h
```

The services are then set to start automatically upon system boot.

```
$ sudo systemctl unmask hostapd
$ sudo systemctl enable hostapd
$ sudo systemctl enable dnsmasq
```

4.2.3 DNS translation

Desired domain name translations are to be included in the system’s hosts file. For the intermediary device to have a human readable name, the URL *http://webbox.local* ought to be translated into the intermediary device’s interface’s IP address 172.16.1.1. The IP address and domain name are added to the hosts-file, which will populate the DNS server’s translation list. This configuration will affect all access devices as the DNS settings will be automatically provided by the intermediary device.

```
# /etc/hosts
172.16.1.1        webbox.local
```

4.2.4 Wireless access point

Finally, the hostapd will be configured to enable the wireless interface to act as a wireless access point. As the focus is on the ease of use, the access point will be configured to broadcast the Wi-Fi SSID and require no password.

```
# /etc/hostapd/hostapd.conf
interface=wlan0
ssid=webbox
hw_mode=g
channel=5
```

The program’s general configuration file is instructed to look for the newly created Wi-Fi configuration by adding the Wi-Fi configuration file path to the general configuration file.

```
# /etc/default/hostapd
DAEMON_CONF=”/etc/hostapd/hostapd.conf”
```

4.3 Installation of web pages, applications, and services

4.3.1 Simple file upload and h5ai file indexer

First, a folder is designated on the web server where h5ai will control the display of files; and where the files uploaded through simple file upload will be located. Path */var/www/html/filebox* is used and it will be available through *http://webbox.local/filebox*.

```
$ sudo mkdir /var/www/html/filebox
```

h5ai is downloaded from the developer's site.

```
$ wget https://release.larsjung.de/h5ai/h5ai-0.29.2.zip
$ sudo unzip h5ai-0.29.2.zip –d /var/www/html/filebox
```

To allow h5ai manage the folder *filebox* and its contents, h5ai's public index must be included in the end of Apache's directory index list. Otherwise, all previous configurations should remain the unchanged.

```
# /etc/apache2/mods-available/dir.conf
DirectoryIndex index.html … /filebox/_h5ai/public/index.php
```

A simple upload page is created. Full contents included in Appendix 1.

```
$ sudo nano /var/www/html/upload.php
# /var/www.html/upload.php
<!DOCTYPE html>
…
```

4.3.2 Jirafeau

Jirafeau is installed by cloning the latest collection of files into path */var/www/html/Jirafeau*, hence it is usable from the URL *http://webbox.local/Jirafeau*. In addition, a folder *Jirafeau_data* is created outside the web server root folder to contain uploaded files.

```
$ sudo mkdir /var/www/Jirafeau_data
$ cd /var/www/html
$ sudo git clone https://gitlab.com/mojo42/Jirafeau.git
```

4.3.3 Chatr

Chatr is installed by copying its files into a subdirectory inside web root.

```
$ cd /var/www/html
$ sudo git clone https://github.com/weex/Chatr
```

4.3.4 cowyo

Since cowyo does not need to reside inside web root and it is started from a binary file, a new system user is *cowyo* created for dedicated use of cowyo with its own home folder. Additionally, no password is set to the user to allow easy usage. The application is available for use through *http://localhost:port* and in this usage scenario through *http://webbox.local:8050*.

```
$ sudo useradd –m –r –d /home/cowyo
$ sudo passwd –d cowyo
```

cowyo is available for download as binary files for multiple operating systems and processor architectures. A respective binary is downloaded according to the system and processor architecture: cowyo Linux ARM.

```
$ cd /home/cowyo
$ sudo wget “https://github.com/schollz/cowyo/releases
                  /download/v2.12.0/cowyo_linux_arm”
```

The file is made executable.

```
$ sudo chmod +x /home/cowyo/cowyo_linux_arm
```

cowyo can be started as the system user *cowyo* by executing the binary file.

```
su cowyo –c "/home/cowyo/cowyo_linux_arm"
```

4.3.5 CryptPad

Comparable to cowyo, CryptPad is located outside the web folders and therefore a passwordless system user is created to operate the application.

```
$ sudo useradd –m –r –d /home/cryptpad cryptpad
$ sudo passwd –d cryptpad
$ cd /home/cryptpad
```

Installation files are cloned from CryptPad's official GitHub to system user's home folder.

```
$ sudo git clone
"https://github.com/xwiki-labs/cryptpad.git cryptpad"
$ sudo chmod –R cryptpad:cryptpad /home/cryptpad/cryptpad
```

CryptPad and its dependencies are installed using the package managers npm and bower.

```
$ cd cryptpad
$ sudo npm install && sudo npm install –g bower
$ su cryptpad –c "bower install"
```

CryptPad's default configurations are satisfactory for the purposes of this scenario, thus they are copied into the used configuration file.

```
$ cd config
$ sudo cp config.example.js config.js
$ exit
```

4.3.6 Frontpage

After all other web pages, applications and services are installed, a central access page, index.html, is made. It consists of a simple HTML table containing links to all available services categorized according to their roles: Download/upload, Communication, and Productivity. The frontpage will be located in the web root, so all users accessing the intermediary device through *http://domain.local* are automatically redirected to the front page. Full contents are in Appendix 2

```
# /var/www/html/index.html
<html>
…
```

4.4 Post set up

After downloading and installing the applications and files is done, the intermediary device can be disconnected from Internet. It is recommended all services are configured to start automatically during the device boot.

In the case of CryptPad and cowyo, their startup will be automatized through system scripts. The services are started as the previously created service specific system users, using the files located in their respective home folders.

```
# /etc/systemd/system/cryptpad.service
[Unit]
Description=cryptpad service
[Service]
User=cryptpad
ExecStart=/usr/bin/node /home/cryptpad/cryptpad/server.js
WorkingDirectory=/home/cryptpad/cryptpad
Restart=always
[Install]
WantedBy=multi-user.target
```

A similar service is created for cowyo.

```
# /etc/systemd/system/cowyo.service
[Unit]
Description=cowyo service
[Service]
User=cowyo
ExecStart=/home/cowyo/cowyo_linux_arm
Restart=always
[Install]
WantedBy=multi-user.target
```

Also, both services are set up to be automatically started during system boot process.

```
$ sudo systemctl enable cryptpad.service
$ sudo systemctl enable cowyo.service
```

The file size limitation of uploaded files in PHP’s default configuration is set to two megabytes, which is too restrictive for the intended use. The limitation is increased to five hundred megabytes in the PHP configuration file.

```
# /etc/php/7.0/apache2/php.ini
upload_max_filesize = 500M
post_max_size = 500M
```

In order to ensure the web server user *www-data* has ownership of all files and folders used by the web server, the following commands are entered. These controls give the user *www-data* control over web root and the data upload folder used by Jirafeau.

```
$ sudo chown -R www-data:www-data /var/www/html
$ sudo chown -R www-data:www-data /var/www/Jirafeau_data
```

The configuration for Jirafeau and Chatr is done on web pages using a browser, therefore general operability of the web server is first required. Therefore, when all the services are automated, the intermediary device is rebooted to confirm their automatic start. Services status is verified through the operating system's system and service manager systemd.

Status of all default services is checked. Services: apache2, dhcpcd, dnsmasq, and hostapd; should have a plus sign in front of them to indicate the service is up and running.

```
$ service --status-all
```

Additionally, the status of the custom-made services is done individually.

```
$ systemctl status cowyo.service
$ systemctl status cryptpad.service
```

If all services are active, it means the intermediary device and all its implemented software and services are available. In order to finish the set-up of Chatr and Jirafeau, which is done through web pages, a device is connected to the wireless network broadcast by the intermediary device and the setup pages are accessed through a web browser.

Jirafeau setup page is accessed at *http://webbox.local/Jirafeau/*. The user is then redirected to an install script where the administrator password, base address, and data directory are configured. Base address of *http://webbox.local/Jirafeau/* is used and for the data directory previously created folder in the path */var/www/Jirafeau_data/* is used.

Chatr installation page is located from *http://webbox.local/Chatr/install.php*. An administrator account and used domain are configured.

5 TESTING

The goal of the testing is to evaluate does the intermediary device allow the access devices to fulfill the previously defined use cases: file transfer and sharing, communications, and productivity. Moreover, the used standards are rudimentary: can the access device connect to the intermediary device, is the web application or service functional on the access device and is the use case streamlined for the user. The user experience is mostly disregarded outside of ease of use, unless it significantly impacts use functionality. The method of testing is to use the access device to connect to the Wi-Fi network broadcast by the intermediary device. Then a browser is opened and the URL *http://webbox.local* is entered. Lastly, all offered services and web applications are tested for their functionality on the device's web browser.

Access devices used for testing are based on what are available for the use of this thesis. The tests are run with the access devices using the stock operating system and the default web browser. The following list of devices were available for use:

- Nokia Lumia 820
- Nook HD+
- Xiaomi Redmi Note 4 MTK
- Xiaomi Pocophone F1

Additionally, a Lenovo ThinkPad T410 laptop running a Debian operating system with multiple browsers is used to test Wi-Fi connection and establish a baseline experience of web application and service functionality.

Although Nook HD+ was available, it could not be used for testing purposes. It had been previously set up using a community implemented customized Android variant and was therefore unable to meet the set criteria of featuring stock software. In order to mend this issue, a stock Android version, created by the manufacturer, was installed on the device. It was then discovered the device's initial set up requires a connection to the manufacturer's server to check the latest security updates. For reasons unknown the server could not be reached and as this step could not be bypassed the device could not be made operational using the stock Android and, consequently, could not be used for testing.

Initial baseline testing with the T410 laptop found no issues with the intermediary device. Wi-Fi connection was established successfully. The intermediary device provided correct configuration through DHCP: the intermediary device’s IP was assigned as the default gateway and DNS server, and the translation of the custom domain resulted in the intermediary device’s IP address. The intermediary device’s web server was functional. It forwarded the user to the implemented frontpage containing links to all services and applications. Simple file upload worked correctly, and the files were available and displayed in the filebox. The file indexer allowed for media files to be previewed or downloaded. Similarly, Jirafeau operated as intended files were uploaded to its own folder and functional unique addresses were created for the uploaded files. The communication application Chatr also proved to operate normally, user was able to access the chat room after choosing a nickname and messages could be sent and received. Finally, the productivity software cowyo and CryptPad presented no issues. The simplistic wiki cowyo could be used to write notes, save the notes, turn them into lists, and to remove the notes. CryptPad web suite functioned as projected. Variety of different applications provided by CryptPad were utilized successfully.

Tests done with the access devices mirrored similar results with the baseline experience. Beginning to use the applications and services was simple. Everything was accessible through few user inputs. The file transfer and communication applications and services had zero complications, and file preview worked on the older Lumia 820. On the other hand, issues rose with the productivity tools. All smart devices experienced glitches with the cowyo. Lumia 820 could not operate the menus of the service. Likewise, the Xiaomi devices could not operate the selection drop menu. It would open and close soon after, and no selection of other notes could be made. A lesser issue was with CryptPad suite on Lumia 820. Although the web applications would open correctly and be fully functional, Lumia’s small resolution was not available to display all elements. On the CryptPad frontpage web elements and links would be cover the applications preventing the user of accessing some of the applications, though, they could be accessed through URLs. Some of the applications such as the rich text editor would be entirely functional even with the optional chat window open, whereas the drawing application whiteboard could not be fit on the device’s screen rendering the application unusable. On both Xiaomi devices there were no issues with CryptPad as the larger resolution allowed all elements to be displayed properly.

Overall, the intermediary device was proved mostly viable. Fulfilling the important requirements set for the device. The device, after initial set up, is entirely independent. When the device is powered up it will start the operating system and automatically enable all services, broadcasting the SSID for simple easy access, requiring no password. The network side of the device had no complications. Connection through Wi-Fi was a non-issue and DHCP server functioned as intended, providing automatic settings for the access devices. The reason for cowyo's problems is unclear as it operated with no issues on the laptop's browser, especially when similar problems occurred on all tested smart phones, further testing would be required for more extensive investigation. Lumia 820's issues with CryptPad were due to the access device's resolution, though it can be argued CryptPad's design limits its availability arbitrarily for requiring large resolution. Similar issues that rose with the few applications could be prevented through a more careful selection of software. Because of the intermediary device's flexible nature observed complications can be effortlessly fixed, therefore they do not result in a significant impediment in the device's viability.

6 CONCLUSION

The purpose of the thesis was to implement an independent wireless web server to allow smart devices to retain capability basic functionality using web Wi-Fi and web browser without relying on external services. These goals were achieved in a satisfactory manner. The intermediary device is, after the initial set up, autonomous and automatic. All necessary services and web applications are automatically started and available after successful boot. The access devices require no additional configuration and utilizing the services and applications on the intermediary is simple. The user only needs to connect to the broadcast Wi-Fi, enter the URL of the server to the web browser, and they can start using the software.

Some issues rose with the hosted web applications during testing. Lumia 820’s resolution was not high enough to properly display CryptPad’s front page nor to properly operate CryptPad’s Whiteboard drawing applet. Additionally, a more troublesome problems were had with the minimalist wiki, cowyo. The access devices’ browsers could not be used to operate the dropdown menus in cowyo, making it impossible to publish any writings, thereby rendering the web application mostly unusable.

Device and browser variety used in testing was to some extent deficient. The main limiting factor of the number of test devices was due to only using devices readily available, by actively seeking more devices a more conclusive testing could have been done. The two Xiaomi smart phones used ran different versions of Xiaomi’s own Android implementation, and it is likely the stock browsers were very similar – if not identical. A more comprehensive testing ought to be done using wider variety of devices from multiple operating systems using more dissimilar browsers in order to gauge the universal usability of the intermediary device.

On the other hand, it is likely most issues between different browsers and offered services by the intermediary devices can be avoided entirely. By actively seeking more simplistic software, meaning less complex code and technologies used in it, to be served, a more uniform user experience could be achieved. Moreover, due to the intermediary device’s modular nature multiple software serving the same purposes can be present. Therefore, even if some applications are inoperable on certain access devices while working perfectly on others, and vice versa, the presence of the multitude of software would have no significant negative impact on the intermediary device.

REFERENCES

[1] “Number of smartphone users worldwide from 2014 to 2020 (in billions)”, Statista, 2019, www.statista.com/statistics/330695/number-of-smartphone-users-worldwide (accessed 4.5.2019)

[2] M. Kanellos, “152,000 Smart Devices Every Minute In 2025: IDC Outlines The Future of Smart Things”, Forbes, 2016, www.forbes.com/sites/michaelkanellos/2016/03/03/152000-smart-devices-every-minute-in-2025-idc-outlines-the-future-of-smart-things (accessed 4.5.2019)

[3] L. Merola, "The COTS software obsolescence threat," Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'05), Orlando, FL, USA, 2006, pp. 7

[4] J. A. Erkoyuncu, S. Ononiwu, R. Roy, “Mitigating the Risk of Software Obsolescence in the Oil and Gas Sector,” Procedia CIRP, Volume 22, 2014, pp. 81-86

[5] S. Rajagopal, J.A. Erkoyuncu, R. Roy, “Software Obsolescence in Defence,” Procedia CIRP, Volume 22, 2014, pp 76-80

[6] K. Jenab, K. Noori, P. D. Weinsier, S. Khoury, "A dynamic model for hardware/software obsolescence", International Journal of Quality & Reliability Management, Vol. 31 Issue: 5, 2014, pp.588-600

[7] “Non-market App Distribution: For iOS Apps”, Monaca, 2019, https://docs.monaca.io/en/products_guide/monaca_ide/deploy/non_market_deploy/ (accessed 8.5.2019)

[8] “Windows 10 Mobile End of Support: FAQ”, Microsoft, 2019, support.microsoft.com/en-us/help/4485197/windows-10-mobile-end-of-support-faq (accessed 6.5.2019)

[9] C. Faylor "StarCraft 2's Non-existent LAN Support Explained, Piracy Cited as Key Reason", Shacknews, 30.6.2009, www.shacknews.com/article/59340/starcraft-2s-non-existent-lan (accessed 8.5.2019)

[10] J. Papadopoulos "Blizzard finally admits that the lack of LAN in StarCraft II was due to piracy", DSOGaming, 22.6.2012, www.dsogaming.com/news/blizzard-finally-admits-that-the-lack-of-lan-in-starcraft-ii-was-due-to-piracy (accessed 8.5.2019)

[11] "Nintendo Switch Online Overview & FAQ", Nintendo, 2019, www.nintendo.com.au/help/nintendo-switch-online-overview-faq/does-nintendo-switch-support-local-save-data-backups (accessed 5.5.2019)

[12] "Saved Data in System Storage", Sony, 2019, manuals.playstation.net/document/en/ps4/settings/data_system.html (accessed 5.5.2019)

[13] New Media Rights “Open Source Licensing Guide”, New Media Right, 15.10.2015, www.newmediarights.org/open_source/new_media_rights_open_source_licensing_guide (accessed 10.5.2019)

upload.php

Modified from gist.github.com/taterbase/2688850 (accessed 10.5.2019)

```
<!DOCTYPE html>

<html>

<head>

<title>Simple File Uploader</title>

</head>

<body>

<form enctype="multipart/form-data" action="upload.php"
method="POST">

<p>Upload your file</p>

<input type="file" name="uploaded_file"></input><br />

<input type="submit" value="Submit"></input>

</form>

</body>

</html>

<?PHP

if(!empty($_FILES['uploaded_file']))

{

$path = "filebox/";

$path = $path . basename( $_FILES['uploaded_file']['name']);

if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'],
$path)) {

echo "The file ".  basename( $_FILES['uploaded_file']['name']).

" has been uploaded";

} else{

echo "There was an error uploading the file, please try again";

}

}

?>
```

index.html

```
<html>
         <body>
                  <center>
<br></br>
<table style=font-size:250% border="1"x>
<tr>
         <th> Upload/download </th>
         <th> Communication </th>
         <th> Productivity </th>
</tr>
<tr>
<td><a href="/upload.php">Upload to filebox</a></td>
<td><a href="/Chatr">Chat</a></td>
<td><a href="http://webbox.local:3000">CryptPad</a></td>
</tr>
<tr>
<td><a href="/filebox">filebox</a></td>
<td> </td>
<td><a href="http://webbox.local:8050">cowyo</a></td>
</tr>
<tr>
<td><a href="/Jirafeau">Jirafeau</a></td>
<td> </td>
<td> </td>
</tr>
</table>
                  </center>
         </body>
</html>
```