Totte Sjöman

Alternative AV Control Platform

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Alternative AV Control Platform

15 March 2018

Metropolia

Author Title Number of Pages Date	Totte Sjöman Alternative AV Control Platform 34 pages 15 March 2018
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Jonna Eriksson, Senior Lecturer Antti Piironen, Principal Lecturer

The technology of today is steering away from dedicated hardware and more toward software-based solutions. May it be in entertainment systems or household appliances, software is making its presence known. A greater need for integration between different systems may also be felt across several industries, escalating the need for flexible platforms.

Faced with many of the same challenges as the IT (Information Technology) industry, the Audio-Visual (AV) industry is increasingly merging with IT by adopting many of the same principles. The progression of IP based technology acts as a catalyst for this phenomenon. By connecting everything from projectors to TVs, to networks, previously almost solely occupied by servers, switches, and modems, it has added to the cacophony of networked devices, all communicating on some level. This increased need for networked communication has subsequently ramped up the need for APIs (Application Programmable Interfaces) in devices. These APIs are also starting to play a larger part in how AV control systems are implemented today.

This study delves into where the current state of affairs has left the AV industry. As a field with a rich history of proprietary protocols and dedicated hardware, it may have a lot to gain from embracing modern innovative patterns. In AV, there is often a strong demand for integration, and varying needs among customers drive the urgency for flexibility and scalability.

Through using software development as a core part of the process, solutions can potentially be made more streamlined and more in tune with these needs. In this study a proof-of-concept project carried out a Company X is examined to shed some light upon the subject. Through evaluating the work flow and findings of this project, a better understanding of the role of software in the AV industry may be gained.

Keywords	AV, software development, API, control

Metropolia

Författare Arbetets namn Sidantal Datum	Totte Sjöman Alternative AV Control Platform 34 sidor 15.3.2018
Utbildningsnivå	Ingenjör (YH)
Utbildningsprogram	Informationsteknik
Inriktning	Mjukvaruutveckling
Handledare	Jonna Eriksson, universitetslektor Antti Piironen, huvudlärare

Dagens teknologi tar avstånd från dedikerad hårdvara och fokuserar i högre grad på lösningar baserade på mjukvara. I allt från underhållningsystem till hushållsapparater, spelar mjukvara en notabel roll. Behovet av integration mellan olika typer av system trappar också upp behovet av flexibla plattformar.

AV-industrin (audio-visuell) bemöter många av de utmaningar som IT-industrin (informationsteknologi) ställs inför. AV-industrin håller därmed på att smälta samman med IT i allt snabbare takt genom att bejaka många av dess huvudsakliga principer. Framskridandet av IP-baserade teknologier har en katalyserande effekt på detta fenomen. Genom att koppla allt från projektorer till TV-skärmar, till nätverk som tidigare bestod av enbart servrar, nätverkswitchar och modem, har gett upphov till en kakofoni av apparater som alla kommunicerar på något plan. Det ökade behovet av nätverkskommunikation har även ökat behovet av API (Application Programmable Interface) gränssnitt i apparater. Dessa gränssnitt spelar också en allt större roll i hur kontrollsystem för AV byggs upp idag.

Denna studie undersöker hur AV industrin ser ut I dagens läge. Med en rik historia av att utnyttja specialiserade hårdvaruprodukter och protokoll, kan ett segment som AV dra stor nytta av att bejaka moderna mönster för innovation. Inom AV finns det ofta ett stort behov av integration. Detta i kombination med varierande kundkrav, gör att flexibilitet och skalbarhet i allt högre grad blir nödvändigheter för AV-industrin.

Det kan vara möjligt att skapa mer modifierbara lösningar genom att tillämpa mjukvaruutveckling, som en grundpelare i processen. I denna studie begrundas ett bevis på konsept project, som utfördes åt företag X. Genom att betrakta arbetssättet och resultaten av projektet är syftet att få en bättre insyn i mjukvarans roll inom AV-industrin.

Nyckelord	AV, mjukvaruutveckling, API, kontroll

Metropolia

Table of Contents

List of Abbreviations and Key Concepts

AV	Audio-Visual. Referring to the industry providing permanent installations of entertainment systems.
RS-232	Recommended Standard 232, a serial transmission standard originally developed in 1962. It has been widely used in the AV control context.
API	Application Programmable Interface.
Company X	The Case Company at which the proof-of-concept project was carried out. The company name is undisclosed for reasons of confidentiality.
IR	Infrared technology. Commonly found control method in consumer products such as TV remotes.
IP	The Internet Protocol stack. A set of communication protocols widely used today including Transfer Layer protocols TCP and UDP.
Crestron	Crestron Electronics, Inc. AV manufacturer, platform provider. Registered trademark. All rights reserved.
AMX	AMX by Harman. AV manufacturer, platform provider. Registered trademark. All rights reserved.
Extron	Extron Electronics. AV manufacturer, platform provider. Registered trademark. All rights reserved.
QSC	QSC Audio Products LLC. AV manufacturer, platform provider. Registered trademark. All rights reserved.
HTTP	Hypertext Transfer Protocol. Application Layer protocol for communication between a client and a server.
Request	An HTTP request message.

Response	An HTTP response message.
REST	Representational State Transfer is an architectural design style for creating web service APIs.
Drag & Drop	The ability to move an object on a user interface using a mouse, as defined in the Oxford Dictionary.
LUA	A standard lightweight scripting language.
Skype	Registered Trademark of Microsoft Corporation. All rights reserved.
CodeAcademy	Online learning platform.
Agile method	A group of conventions for software development aligned with the ideals expressed in the Agile Manifesto.
Adafruit Industries	Registered trademark. All rights reserved.
Raspberry Pi	Registered trademark of Adafruit. All rights reserved.
Beagleboard	Registered trademark. All rights reserved.
NewTek Inc.	Registered trademark. All rights reserved.
TriCaster	Registered trademark of NewTek. All rights reserved.
.NET	Microsoft Corporation development framework.
Windows	Microsoft Corporation operating system.
Java	Registered trademark of Oracle. Programming language.
JavaScript	A programming language for web development.

Android	Registered trademark of Google. All rights reserved.
IOS	Registered trademark of Apple Inc. All rights reserved.
vMix	Registered trademark. All rights reserved.
vMix Software	Registered trademark of vMix. All rights reserved.
SSH	SSH Communications Security Inc. All rights reserved.
Filezilla	Open source file transfer software distributed free of charge under the terms of the GNU General Public License.
VS Code	Microsoft Corporation. Development tool.
Netgear	Registered trademark. All rights reserved.
SOLID	A set of principles related to software design
HP	HP Development Company. Registered Trademark. All rights reserved.

1 Introduction

The impact of software and software-based solutions on the manner in which people work, may be noticed across a wide range of industries and fields today. This is also true for the Audio-Visual (AV) industry where manufacturers have launched mobile, web and desktop applications for controlling many products.

The concept of AV may refer to a multitude of interlinked subjects, ranging from products and companies to technologies and entire industries. For the sake of this study, the term, AV, will be used to describe the industry based on building, programming, and deploying permanent entertainment systems. The industry in question has long employed solutions for integrating controls for entertainment technology, such as audio, video, and lights. The foundation of these systems has traditionally been and currently is, a combination of different standards, protocols, manufacturer patented dedicated hardware, and proprietary applications. Accumulated through several decades, the premises for constructing systems such as these may in some cases become quite complex in nature. The lack of conformity and well-established industry standards and design principles have spawned a unique field open to alternative solutions. The purpose of this study is to deliberate upon the notion of utilizing software development as an alternative to creating AV controls. The project was conducted as a proof-of-concept at case company X. The primary goals were to propose alternative control solutions and to assist in evaluating the usability of such solutions in future projects as well as determining if further research on the subject is recommendable.

The thesis is based upon industry journals, application notes and research papers. Several web pages relating to software development and forums for AV system design, have been referenced as well. This study includes five core sections. The Introduction is followed by a description of best practices in the AV (Audio-Visual) industry in section 2. This is to provide the reader with a background for referencing the rest of the work. Section 3 determines the scope of the project and section 4 explores the methods used. Finally, section 5 deliberates upon the findings and summarizes the conclusions.

2 AV Industry Best Practices

The Audio-Visual (AV) industry has employed a large variety of paradigms for designing and controlling entertainment systems. An AV installation may constitute a wide collection of different devices, ranging from audio, video, broadcast and lighting equipment to collaboration codecs and motorized screens etc. When implementing larger systems, dispersed over many spaces in a building or even multiple buildings, conventional approaches for constructing the control systems for such setups result in the control logic becoming increasingly complex in nature.

Due to the progression of IP technology and software applications, the AV industry is in tune moving progressively towards software focused solutions. As discussed in a podcast dedicated to AV control methods and practices, “A State of Control”, published by AVNation, a prominent AV news forum, the role of the AV programmer is shifting and becoming more closely linked to that of a software developer. Simultaneously older control protocols previously commonplace and popular such as Infrared technology (IR), are getting pushed out in favor of Internet Protocol (IP) based technologies. [1] [2]

2.1 The End of RS-232

A host of different protocols and standards have been used as control methods throughout the history of AV. One such protocol, which has earned a noticeable status across the industry, is RS-232. [3]

EIA/TIA-232-E, more commonly known as RS-232 is a standard for serial communication developed by the Electronic Industry Association and the Telecommunications industry Association (EIA/TIA) and was originally introduced in 1962. The letters stand for “Recommended Standard” and RS-232 is one of the most widely used in a family of related standards. In short, the standard provides a simple way of communication between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE). The DTE acts as the client, or host machine, initiating the connection, whereas the DCE is the remote party. [4]

A 25-pin connector is specified as the mechanical interface. However, since most of the pins provide functionality such as hand-shaking and feedback among other features, many of these are not required by most of the related applications. This in turn has led to the common use of a 9-pin version, in the form of a DB9 connector with a reduced set of signals. [4]

In the AV scenario, a control processor might constitute the DTE and a controllable device the DCE. In order for a serial connection to be established between the devices a dedicated cable is needed [3]. The cable may not exceed the length restricted by the RS-232 specified maximum load capacitance of 2500pF [4]. In addition, the cable needs to be appropriately terminated by an industry professional. This procedure has been a common practice for controlling everything from projectors and TVs to residential lighting etc. as discussed in “3 Reasons an Experienced Integrator Embraced AV over IP” [3]. However, the industry is moving toward IP based solutions whereby existing IP networks may be utilized for controlling devices. This is a shift involving breaking away from not only RS-232 but also IR, which has long been a popular method for controlling consumer electronics, along with other standards and non-standard approaches. [5]

As stated by Hagan D. [3], utilizing IP and Ethernet networks for controlling devices offers a multitude of advantages. A single cable may be used to carry the control signals needed in addition to providing a route for media streams such as audio and video. The same infrastructure may then also provide power in the form of Power-over-Ethernet. This results in reduced deployment costs. In addition to this IP offers a foundation suitable for encryption and security features as well as radically increases bandwidth and enables centralized management of systems.

As previously noted, newer technologies such as Ethernet, especially IP communications, are offering remarkably faster transmission rates among other benefits in comparison to RS-232. It is in any case worth noting that these gains are accompanied by a higher level of complication. As discussed in “Older Communication Standards Still Compete with USB” [6] the case may be made for RS-232 as a proven, reliable, and simple way of establishing a point-to-point connection between two devices. [6]

Even though newer, faster and more flexible ways of transmission most probably will ultimately render RS-232 obsolete, this is not yet the case. As noted in a popular Electronics Community blog [7], there are still vast amounts of devices equipped with and reliant upon the legacy standard. In addition, by the time of the publication of the blog in question, in 2013, a manufacturing company by the name Sealevel Systems, was announcing new hardware in the form of a computer interface card equipped with an RS-232 port.

Summarizing the fate of RS-232, one may conclude that the standard will eventually be phased out in favor of IP based technology. However, the change will not take place over night. [7]

2.2 Restful APIs & Web Sockets

As the AV industry is shifting closer towards the IP stack and software development, this naturally entails adopting methods to fit the technologies. This section will discuss the basic principles of Application Programmable Interfaces (API), Representational State Transfer design specifications (REST) and the Hypertext Transfer Protocol (HTTP) along with briefly deliberating on how these methods tie into the AV context.

The heart of an AV control system has traditionally been some form of control processor as mentioned in “Leveraging IP and the Cloud for AV Control” [8]. This control processor may offer methods of control using protocols such as IR and RS-232 as discussed in a previous section. However, these types of processors are increasingly leveraging IP based technology, in the form of web sockets, to control devices. A predicate for this is that the controllable devices be equipped with some form of API built for interfacing with IP stack protocols. Examples of such protocols are the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The specifications for these may be found in detail in [9] and [10] respectively. This is nevertheless outside of the scope of this study. Throughout this study, the term protocol is used to describe a collection of different protocols relating to communication.

The API or Application Programmable Interface, may be defined as a specified mapping of controls to desired actions. An API is the programmer’s tool for interfacing with the

service providing the API [11]. The documentation of the API declares what protocols may be used for establishing a connection to the server and how the client requests are to be formatted, along with any other relevant details for successfully interacting with the API. [12]

When contemplating the topic of APIs, another term that frequently emerges is REST. REST is an architectural pattern for constructing APIs and defining the manner in which client applications should interact with them [13]. Although the term is used quite deliberately, all APIs described as RESTful, are not necessarily conforming to the rules of REST [15, 14]. One common deviation from the guidelines is the client making requests based on previous knowledge of the format required by the server and the resources residing on the server. In doing so the developer is not adhering to the principle of REST stating that apart from the root URL (Uniform Resource Locater), the part of the address pointing to the target server, the client should only be allowed to make requests based on the data sent back in hypertext from the server. Straying from this rule leads the implemented design pattern away from the Restful architecture and more towards using a Remote Procedure Call (RPC) technique [11]. Like REST, the RPC method is designed for use in the creation of client-server communications. The basic concept of RPC entails a client making procedure calls outside the scope of the local address space [14].

Implementing IP based controls for AV devices, the approach may often be based on a strictly defined set of formatting rules and conditions that need to be met for the application to work properly. As stated previously this is not in line with all restrictions of the REST design pattern presented in [13]. Hence it may in many cases be appropriate to swap out the term “REST”, for “non-REST” [11] when referencing APIs in implementations relating to AV. The use of APIs in an AV context will be deliberated on further in the later sections.

Discussing client-server relations and APIs, be they REST compliant or not, a very relevant application layer protocol is HTTP, which enables the request/response communication as defined in a request for comments by the Internet Engineering Taskforce [15].

An important property of the HTTP/1.1 protocol is that the data representation may be agreed upon by the communicating parties. This creates an opportunity for the client application and the server application to be developed autonomously from each other,

as noted in both “Do you know what a REST API is?” [12] and “Hypertext Transfer Protocol HTTP/1.1” [15].

The request/response structure of HTTP makes it a suitable protocol for creating AV controls. Simply put, a request may be sent to a device, triggering some function. In addition, this structure may also be taken advantage of in creating polling functions. As defined by Rouse M. [16], polling is the method of repetitively requesting information from another entity. In the context of AV, this might for example be requesting lamp hours from a projector.

Taking advantage of IP technology, the AV industry has much to gain as deliberated on by Shamir D. in “Leveraging IP and the Cloud for AV Control” [8]. The article published on AVNetwork, a distinguished website targeted toward AV and IT (Information Technology) integrators, further discusses that adopting this paradigm may enable a substantial increase in the scalability and redundancy of AV systems. Shamir argues that this in turn may lead to a considerable rise in efficiency compared to the more conventional solution, in effect revolving around dedicated hardware and proprietary applications. [8]

2.3 Examining the Concept of Dedicated Hardware

A typical AV installation may include a multitude of devices used in conference rooms, event halls, lobbies, and other venues. These may include audio systems, video collaboration codecs, lights, projectors, and TVs among other applications. Adding to this multiple rooms over multiple floors and integration to third-party systems, the complexity of the control logic quickly increases when using conventional approaches. The traditional way in which these AV installations has been done is usually based around some form of dedicated control processor [2]. This generally results in a labor intensive and intricate programming process, not to mention lofty hardware expenses. These challenges are discussed in detail by AV professionals in an online seminar [17] sponsored by QSC, a prominent AV platform manufacturer and developer. [17]

Across the AV industry today, a shift in how integrated solutions are implemented is taking place, as noted in an article on AVNetwork [2]. The basis for this change lies in the approach to constructing platforms for integrated and centralized control. Previously

integrators have been forced to rely upon proprietary hardware, built and developed by a single manufacturer for a specific task [17]. This has made for rigid systems that may be difficult and expensive to not only build but also maintain, especially when discussing larger installations with multiple spaces [17].

Today the industry is moving more towards utilizing IT standards and software based solutions [2]. By doing so, the process for building integrated systems may become considerably more cost-effective by increasing simplicity, scalability, and the possibilities for centralized management [17].

QSC takes part in this transformation of the AV field by providing equipment and software built on general IT standards, rather than proprietary protocols. By grounding their control processors on Linux distributions and Intel technology, QSC is creating a solid hardware base with a vast global network of industry professionals with the expertise and experience to both support it and help develop it further. Thus, the result of essentially remolding the AV control processor into the form of a standard PC appears to be a significantly more limber and diverse platform. [17]

In “Development of computer control systems with hardware-software codesign” [18] a hardware-software codesign model was discussed. The model suggested that delegating responsibilities between software and hardware components at a later stage of a given project would make the system design process more flexible. Especially when faced with systems with a rising level of complication, the authors argued that streamlining the hardware side of things would lead to a reduction in cost and improved sustainability overall.

Further building on paradigm of IT standards, QSC offers a software layer on top of the standardized hardware, to handle all signal, mixing and control processing operations [17]. The move towards software defined systems is not only felt in the AV field but in other industries as well. An example of this is IT infrastructure itself [19].

QSC has set out to create an intuitive way for the AV programmer to use and take advantage of their software layer. Subsequently systems may be constructed in a shorter period and arguably to a higher general standard. Joining drag and drop functionality with a standard scripting language, LUA, means that programmers have powerful tools

at their disposal, as well as a rich support community that generally form around standard programming languages. This is in contrast with other large AV platform manufacturers. Crestron, for example, offers a proprietary scripting language called SIMPL+ [20] and AMX in turn uses the proprietary Netlinx language [21].

Examining the AV industry, there is an ongoing migration from dedicated hardware solutions to using standardized hardware platforms as discussed by Mitchell K. [2]. This is in large part due to the growing demands on the capabilities of AV systems. As deliberated on in a white paper by Harman [22], the AV industry today still resembles what the computer field was like in its earlier years; there are a huge number of manufacturers, dealers and integrators resulting in both highly customized and costly systems. Like the IT industry before it, AV needs to embrace a set of general principles to grow its capacity. Harman quite appropriately refers to these principles as the "Six Pillars of IT", namely, "open standards, network capability, scalability, security, and centralized monitoring and maintenance". [22]

2.4 The Impact of a Growing Open Source Community

As discussed in previous sections of this study the AV industry has a history and practice of being heavily dependent on proprietary tools. Everything from the hardware installed to the protocols it supports and from the software platforms to the programming languages used, proprietary and industry specific technology seems to be the norm. However, integration with IT and the strengthening role of software is impacting the industry in many ways. The rapidly expanding open source community is yet another factor which may spur the rate of change in the AV world.

The term open source software, not to be confused with free software [23], refers to a set of principles of a piece of software and its accompanying license. These include but are not restricted to that software being freely used or altered by anyone without the need for paying royalties to the developer. A set of ten requirements for such a software may be found as defined by The Open Source Initiative, a prominent entity behind open source. [24]

Although the original term relates to software exclusively, today, open, has seemingly become an adjective used to describe a wide array of different applications cradling certain aspects of the open source principles [25]. An article on rAVe Publications [26], a well-known AV oriented news site, builds upon values, like those of open source software, by discussing the benefits of using open and agreed upon standards in the AV industry. The importance of such standards in the context of broadcast is also noted by McGarvey [27].

Further extending the topic of openness while contemplating how the industry could address the apparent shortage of AV oriented control programmers. By replacing proprietary software platforms for standard languages and open source platforms, the industry would not only gain from an exploding increase in support from developer communities but might also help make AV more appealing to software engineers early in their careers [28].

Referring to the support community and the value added by open source, one group of researchers, working on a study relating to wind energy harvesting by airplanes [29], developed their own modular software platform as an open source project. As deliberated on in the study, the group required a highly flexible platform suited for rapid prototyping of real-time control applications. Prior to deciding upon creating an open source platform from scratch, the research team investigated proprietary alternatives on hand on the market. However, the open source concept prevailed after the team deemed the alternatives inherently too restrictive for the experimental nature of their research. Providing the modular platform in question under the open source license, the research group invited all who work on similar control projects, to gain from and to add to the work they themselves have already done. This expansion of the development process to a larger community, beyond the confines of a single company, was also considered in a discussion paper on Open Source by the Ministry of Justice in New Zeeland [30].

In summation, the AV field seems to be showing signs of shifting away from proprietary solutions towards more standardized approaches [28]. This combined with the onslaught of software-based platforms entails that AV may be increasingly affected by and developed using open source software and its principles.

2.5 Considering the Role of Software in AV

The AV industry is progressively merging with IT. As discussed by Harman [22], a leading AV platform and hardware manufacturer, this means adopting much of the practices traditionally coupled with IT.

One of the core principles related to the doctrines driving the IT community, is scalability [22]. This may often prove a major challenge for AV programmers building systems on hardware based proprietary platforms [17], while comparing a software-based platform with conventional approaches relying heavily on specialized hardware.

The speakers in “A State of Control” [1], a popular podcast provided by AVNation, pointed out that apart from enabling a nimbler and more easily modifiable structure, using standardized programming languages also brings with it a broader set of learning paths. Instead of having to depend upon manufacturer provided specialized and costly certification courses, programmers may find education from online open learning platforms like CodeAcademy. Assuming this will lead to an accelerated sharing of information, it might also attract new talent to the AV industry [1].

Influenced by IT and the standardized paradigms it brings to the table, the AV programmers are to varying extent undergoing metamorphosis and becoming more like software developers as noted by Backus, one of the speakers in the podcast. Backus further referenced this by discussing the emergence of Agile practices in AV. [1]

Agile software development is a paradigm built upon the ideals of the Agile Manifesto, as explained by Agile Alliance [31] and defined in detail in the Agile Manifesto [32]. Summarizing the key purpose of Agile methodology, one may find that as the term itself describes, the intent is to create an Agile approach to coming up with solutions. This includes building software in a manner that is easily altered at any stage of development [32].

As stated in the AVNation podcast [1], AV will need to embrace the move to standardized languages and software development to maintain relevance and keep up with the demand of future customers. The end users’ growing expectations of AV systems is also

deliberated on by Harman [22]. These expectations include but are not limited to a higher level of system stability and enhanced ease of use.

One factor to consider is the use of visual programming tools. As explained in “What Is Visual Programming? [33], the concept of visual programming entails creating processes and functions using visual elements instead of writing code. The documentation by Crestron [34], describing the main Crestron development environment, SIMPL, short for Symbol Intensive Master Programming Language, explains how graphical symbols, sub menus and drag & drop functions are used to build program logic in a Windows-like environment.

The role of software is not only changing as an instrument for AV professionals and programmers but also as a tool for the end user. One example of this is given in “Hardware/Software Codec Comparison” [35], a blog published by AVNation, studying how collaboration codecs based on software and hardware respectively, compare with each other. The article noted that hardware codecs often provide an easy to use touch screen or other interface, offering a more refined end user experience. This comes attached to a hefty price tag and restrictions in the implementation of the service providing software, for example Skype, usually manifesting in the form of limitations in content-sharing. Purely software-based codecs on the other provide increased flexibility and may be natively integrated with the underlying software, resulting in a richer set of functionalities as well as improved security and scalability. [35]

Considering the best practices deliberated upon in this section, it is worth noting that the AV industry is undergoing a transformation whereby IT related principles are gaining ground. This can be noticed as an increase in software-based solutions rather than rigid dedicated hardware. Referencing communications, IP networks are dominating the market and have become the de facto infrastructure for building control systems, leaving older protocols and paradigms behind. The open source community and the benefits of shared experiences are also having a substantial impact on the industry.

3 Project Specifications

Theory discussed in Section 2 have described the trend of IT principles increasingly taking hold in AV. In tune with this trend of convergence, the author of this study carried out a proof-of-concept project aimed at studying the use of a standardized programming languages and hardware in the AV context. This topic was chosen not only for its relevance for the case company but also as a major current phenomenon in the AV industry as a whole. The following sections describe the specifications and requirements of the project documented in this thesis.

3.1 Determining a Template for Proof-of-Concept

Software development is a vast subject and may refer to a multitude of applications, standards, methodologies, and paradigms. The intent of this paper is not to delve into all of these, instead, when referring to software development, the emphasis is given to the use of standardized programming languages for creating control applications for devices, traditionally controlled through the use of proprietary manufacturer specific, hardware based, systems.

The first step in learning more about such software implementations was to identify a suitable template for proof-of-concept. In effect, determining a simple application for a simple purpose, which would highlight the concept of standardized programming languages and standardized hardware in the AV context.

Early in the process of defining the outlines of this template, the afore-mentioned criteria of simplicity, was considered by the developer as the core ingredient for a successful execution. The project needed to be carried out from start to finish in the allotted time and needed to be easily demonstrated. Additionally, it was deemed necessary for the design to be uncomplicated, enabling the description of the concept to individuals lacking experience in software development.

The template was required to be centered around a clear application. In addition, this target application was required to be familiar to the AV environment, and thereby

strengthen the relevance of the project. Recreating something previously encountered in a new manner was also intended to invoke more interest among industry professionals.

Determining a recognizable control-oriented setup in an AV system, a typical composition would constitute some form of control processor, a target device to be controlled and an intuitive user interface. As discussed previously in section 2.3, the traditional way of implementing such a setup would include dedicated and specialized hardware. For the sake of presenting an alternative concept of control, the components responsible for issuing controls would be swapped out for standardized hardware, available to consumers at a comparatively low cost.

The target device itself was required to be equipped with an API, based on standardized protocols, for integrating with third party systems. The nature of the chosen device was not of special importance, most audio, video, or lighting equipment supporting third-party control would have been suitable for the project.

Highlighting a clear purpose was vital. A common scenario when building AV control systems, would be a user centrally issuing triggers of some kind, to other parts of the system. These triggers would then understandably trigger further functions, configured in the target devices themselves.

Thus, creating a software application, based on standardized programming languages, with the capability of issuing such triggers, would fulfill the requirements of the discussed template. The upcoming section will deliberate in more detail on the specific components chosen for the project. Consideration will also be given, as to how each element was selected.

3.2 Recognizing Hardware & Software Requirements

Contemplating a standardized alternative for housing the traditional dedicated hardware of the control processor, the first platform that came to mind was the Raspberry Pi, a single-board computer by Adafruit Industries [36] [37]. This open source hardware basically constitutes a small, pocket-sized computer. The cost of acquisition is low compared to other products. Other platforms were considered, like the microcontrollers by Arduino

and the single board computers manufactured by Beagleboard [38] [39]. However due to the large support community behind Adafruit products, the Raspberry Pi 3 model B, was chosen as the controller and platform for the project.

Another reason for selecting the Raspberry Pi, was that the Linux based operating system distributions supported on the platform. These distributions in turn include support for a wide range of developer tools and programming languages as well as frameworks. Among the required features for working with the platform were a connection application to access the platform remotely, this could be provided by using for example using SSH (Secure Shell) tunneling, support for which is provided with many Linux distributions [40]. The ability to develop the web application using a work laptop and later transfer the files to the platform was also recognized as necessary. This would be made possible by using some form of ftp application. Filezilla is an example of such a software [41].

In addition to the platform itself, a device for providing the user interface was required. From hotel ballrooms to meeting rooms, in different venues housing AV installations, a common sight is a touch panel offering centralized control over the space. Having chosen Adafruit and the Raspberry Pi for this project, a logical choice for the user interface (UI) was the Pi Foundation Display [42]. The 7-inch screen in question integrates neatly with the Raspberry Pi board.

Through choosing a platform not only qualifying as standardized hardware, available to all consumers, but also based upon, and in support of the principals of open source, the project was expected to gain from the vast experiences of the open source community. This would likely aid in advancing previously discussed topics of software driven control concepts, beyond the scope of the project in this study as well. Providing a versatile platform for future expansion and further development was of crucial importance for this project.

The target device acting as the subject to be controlled, was chosen primarily through evaluating a few currently relevant products available on the AV market. As previously noted, a prerequisite was the presence of an API for third-party integration. The intended use of the product itself was not considered a priority, however the device in question needed to benefit in some way, from a simplified custom user interface.

The broadcast field is closely related to AV and employs powerful applications for managing the setups required for broadcasting, however these usually require professional expertise. In case a smaller entity, such as a local radio station, or a congregation, wanted to acquire the capability to stream events, the resources for a hired technician might not be available. Thus, a simplified user interface would be required as part of the installation in this scenario.

One device considered was the TriCaster Mini, Advanced Edition, by NewTek Inc. [43] [44]. This device is intended for producing, editing, and distributing video content, among other things. Ultimately, however, another manufacturer of video production tools, vMix, was chosen. Unlike NewTek, which puts substantial effort in producing versatile hardware with accompanying applications, vMix mainly provides software solutions. This was the main reason for choosing to implement controls for the vMix software. It was noted that the software would have to be installed and configured on a physical computer in order to demonstrate the setup. The hardware chosen for this purpose was a Lenovo Thinkpad Yoga1 laptop running the Windows 10 operating system. The vMix software enables the creation of live video productions. As such, the software is a professional tool, requiring technical competence. Specifications for the vMix software are found on the company website [45].

Implementations of tools, such as the vMix software, which include a vast array of options and configurability, provide a great deal of flexibility for the end user. This is also something an AV professional would come to expect from such equipment. In some cases, however, a customer, acquiring some form of AV setup, might not need to adjust all of the available parameters of any given system. For individuals not working as professionals in technology-oriented fields, such as AV, the control requirements may often boil down to be able to change between previously defined presets. In other words, adhering to his notion, all the parameters may be configured by a professional in the product software, while only a stripped down, simplified interface is presented to the end user. The intent of the project described in this study was to create such a reduced user interface.

The previously mentioned software provides an API, based on the HTTP protocol. The software documentation also specifies APIs based on TCP commands, VB.NET scripting and web scripting, as documented in the user guide on the manufacturer website [46]. HTTP was chosen for this project. Through using specific commands, the device may be

triggered to perform certain activities. Complying to the HTTP protocol, upon receiving a request, the device would then respond accordingly, while indicating success or failure of the request.

Contemplating the software requirements of the project, a versatile structure providing the possibilities to implement different forms of communications, logic and UI interactions was needed. While examining standard programming languages offering the necessary tools, C# and Java among others were considered.

C# offers a strong foundation upon which to build applications for a wide array of purposes and could certainly have been used to implement the sought-after functionality of this project. However, due to relying upon the .NET framework and since many of said framework's libraries are built for the Windows environment, entailing that using the Windows 10 IoT Core would have been the only option [47], C# was not chosen for this project.

The Java platform is powerful, versatile and provides cross-platform compatibility. This was a strong contender for the project. However, after deciding to leave the possibility of future mobile integration open, it was decided to go for a web development language. In doing so, the control application can be made available for mobile devices through a browser. This may be achieved by tailoring an HTML page to mobile screen proportions, without having to develop the app from start with a mobile platform, like Android or IOS.

Ultimately it was concluded to build the software as a web application, through a combination of using HTML and CSS for the UI, as well as JavaScript to provide the business logic. In addition, popular JavaScript frameworks, Node and Angular were chosen to provide the necessary functionalities of the app. The testing environment would be provided by the command line tool Karma, and the Jasmine framework [48].

The Angular framework would also enable the web application to be built as a single-page application (SPA), effectively meaning that the entire page would not have to be rendered every time the user interacts with the application. As noted in a chapter by Code School, an online learning platform specialized on software development [49], this permits the web application to behave more like a native desktop application. Table 1 illustrates the main requirements of the project.

Table 1 Requirements of proof-of-concept project.

Requirements	
Hardware	
	> Single board computer (small scale) > Touchscreen > Network switch
Software	
	> Linux OS distribution > General purpose programming language (suited for web development, including business logic) > Markup language for the UI > Frontend development framework > Backend development framework > Flexible development environment > Development framework for testing > Testing environment > Text editor/programming tool > File transferring tool

Categorized as relating to software or hardware, Table 1 summarizes the major requirements for the proof-of-concept template. With the hardware and software components decided upon, the next stage of the project was to define the functionalities of the setup. Staying true to the realities of the AV environment, it was noted that even a proof-of-concept application needed to work with minimal effort by the end user. Therefore, special consideration had to be given, as to what kind of initial setup the application would require before functioning as intended. This will be discussed further in the following subsection.

3.3 Focusing on Usability

Planning a project used to analyze a concept required the process to consider both the nature of the implementation phase and the functionalities of the end result. In other words, in addition to creating an application which would be easily demonstrated, the work process itself needed to be dissected and evaluated.

For a control concept based upon standardized protocols to be presented as a plausible alternative to the traditional manufacturer specific systems, the production process needed to be viable from start to finish. Thus, focusing on usability, required taking into account both the end application and the methods of working.

Evaluating what kind of methods would be required for implementing the project in question, principles relating to the Agile development methodology were emphasized. As discussed in previous sections, one of the core doctrines is to maintain the possibility to alter the application all the way through the development process. Figure 1 depicts the Agile Manifesto as defined by its authors [32].

Figure 1 Core values of the Agile Manifesto. [50].

Observing figure 1, the Agile Manifesto explicitly states the importance of reacting to change, even at the cost of adhering to a previously defined course of action. The last principle could perhaps be interpreted as stating that planning is not a high priority. However, based on the work carried out in the project of this study, which was partially conducted in accordance with Agile doctrines, strongly suggests such an interpretation to be

less than accurate. Planning was crucial for ensuring that the project maintain its coherency. The case may be made that what the previously mentioned Agile principle states is that following a plan should never be allowed to affect the end result in a negative manner. Effectively, a plan is necessary for a successful development process, but said plan should be agile enough to allow for adjustment in later stages of the project. This thought process was of central importance in the project described in this study.

Design patterns deriving from the SOLID principles were also referenced and utilized in the application. The letters in SOLID, stand for single responsibility, this principle basically states that a given function should have only one task [51].

In addition to following Agile doctrines, the project was required to respect the fundamental notion of simplicity and ease-of-use. The development of the web application would be aimed at producing code supporting a modular structure. In other words, the plan was to create an application which would be easily interpreted, modified, and added to by any software developer. Software built to these standards would also strengthen the principle of scalability, mentioned previously as belonging to the “Six Pillars of IT” as referred to by the authors of “When Industries Collide” [22].

Pursuing usability when referring to the application itself, it was recognized at an early stage that effort would have to be invested in making the setup easy to use. The user interface would have to be clear and intuitive. This entailed evaluating how and where to place elements on screen, as well as giving thought to the size of potential buttons, labels and other visual components.

The hardware, consisting of the Raspberry Pi and accompanying touch screen, would be required to function without extensive effort by the user. Getting the system up and running should not demand engineering competence, nor should the user of the application be inclined to wonder how the setup is functioning, unless sharing a personal interest for development. A setup designed for ease-of-use should not be allowed to visibly complicate any part of the user experience.

Considering the demonstration of a proof-of-concept, careful consideration was given to the weight of first impressions. A concept poorly presented would most likely not spur further investigation into the subject. Likewise, a concept theoretically sound and well

explained could suffer massively in credibility, in case the practical part of the demonstration was unsuccessful.

Thus, demonstration of the setup of the project in this study needed to be swift, easy, and require minimal human intervention. To achieve this level of usability three core requirements were identified. It was assumed the target device would be in place and powered on, waiting to be triggered by the control application. Firstly, the user interface and control platform would have to be powered on by plugging a single cable to a socket, probably requiring the two components to be attached to each other in some manner, forming a single piece of hardware. Second, the web application would have to launch by itself. The user should not be required to start any part of the application.

Lastly, the application should independently establish a connection to the targeted device, automatically enabling the user to issue triggers using the interface on the touch screen. This naturally requires the Raspberry Pi to be connected to the same subnet as the targeted device, and that the targeted device's IP address be known to the control platform.

Equipped with the requirements and outlines for the project, the next phase in implementing the alternative control concept was to begin the development. Visual Studio Code was selected as the main tool for developing the application. Being an open source software, the tool fitted in with the theme of using open source hardware. Subsequent sections will describe the process of realizing the proposed solution. [52]

4 Proposed Solution

The project in this study was aimed at presenting and evaluating an alternative concept to implementing controls in the AV context. The concept was based on utilizing a combination of standardized hardware and standardized programming languages to create the control platform. IP based communication protocols supported this foundation.

4.1 Examining Chosen Technologies

The proposed solution for highlighting the concept of AV controls built upon standardized components was carried out as a proof-of-concept. It was decided that the project implementation itself should be easily demonstrated once finished. The starting point for the design of this concept template, was to recreate a familiar setup using alternative technology.

The traditional AV control setup would typically consist of a control processor, a user interface, and the target device, as discussed in earlier sections. This pattern was recreated using a Raspberry Pi as the processor, a Pi Foundation Display, as the interface, and the vMix software running on a Lenovo laptop, as the target of the control.

Instead of building the control logic using the platforms of AV manufacturers like Crestron or AMX, the logic of the proposed solution was developed as a web application, entirely using standardized programming languages including HTML and CSS. JavaScript and accompanying frameworks, Node and Angular were used to build the business logic. By choosing a software development process based on a popular and widely used standardized language, the support community available to the developer is substantially larger than the support available when using manufacturer specific programming tools. This contrast is due to the fact that specialized implementations are used by a smaller group of individuals. Thus, improvements and updates are not available to the same degree and frequency as they are for standardized languages, as discussed previously.

4.2 Project Execution

The topic of AV control and the theory behind some of the key concepts have been presented from different angles in this study. This was important in deciding on how to implement the alternative control concept demonstrated in the project. Having determined the components to be used in the project, along with a design pattern to strive for, the practical work was ready to commence.

The first phase of the project involved setting up the hardware, in this case the Raspberry Pi. This entailed installing the operating system along with configuring the platform to be

accessible from an external laptop during the implementation phase. SSH and FileZilla were utilized to enable connections and file transfers to the Raspberry Pi.

This was followed by initiating the development of the web application, which would ultimately be administering the controls to the target software, the vMix. The Visual Studio Code software [52] was used for writing and debugging the application.

The following step involved assembling the rest of the hardware. This included mounting the touch screen onto the Raspberry Pi board and confirming the setup functioned properly. The target to be controlled was ultimately chosen to be the vMix software. As the vMix software was not dependent on any specialized hardware, the target device was setup by simply installing said software on a ThinkPad X1 Yoga laptop equipped with the Windows 10 Pro operating system.

After the hardware setup, the web application was finalized. The design of the HTML page was refined, and the application code files were sifted through to remove any remnants of unnecessary code.

The next phase of the project comprised connecting the Raspberry Pi, accompanied by the touch screen, to same network as the laptop running the vMix software. This was accomplished using a standard network switch, in this case a Netgear ProSafe GS108PE [53]. In addition, the network cards of both computers were assigned IP addresses in the same subnet, to allow for IP communication between the two parties. The network switch was not necessary, as most modern ethernet adapters support the Auto-MDIX (Media Dependent Interface with Crossover) technology developed by HP. Thus, the Raspberry Pi and vMix laptop could have been connected directly. However, a physical switch clearly symbolizes any standard wired IP network and was thus used in the setup for demonstrational purposes.

Finally, the system wide testing was performed. With all hardware setup and configured, commands could be issued from the Raspberry Pi to the vMix software. After adjusting the command format of the HTTP calls, the controls were verified as functioning. This was done by pressing buttons on the user interface and observing the changes in configuration of the vMix software running on the laptop.

4.3 Developing AV Control

The growing role of software in the AV industry was deliberated on in the previous sections. The project tied into this theme by using software development and accompanying principles to create an alternative control paradigm. This phenomenon also presents a challenge for the AV industry. As mentioned in a previous section, by the speakers in “A State of Control” [1], the AV programmer is being confronted with the need to evolve into a software developer.

The same challenge was recognized early on in this project. Being a proof-of-concept, the project also aimed at identifying how the shift toward software development impact the way an AV programmer works. Determining what kind of processes were required to be put in place to allow for AV controls to be implemented in this manner was a key factor in evaluating the viability of such development.

Two clearly distinguishable software structures needed for this project were identified. The first one was issuing a command to the target application, the vMix software, when a button was pressed on the HTML page displayed on the touch screen. The second was a polling function able to continuously request status information from the target.

These structures were developed for the web application utilizing the HTTP protocol and tools available to the JavaScript language. Issuing commands triggered by button presses were made possible by implementing services in JavaScript. These services would then open a TCP socket and send a properly formatted request to the target, using the HTTP protocol. The request content needed to be formatted in accordance with the vMix API documentation, to invoke the desired effects upon the software.

Polling was made possible through using a combination of HTTP services and autonomous functions. The response gotten from the target was then processed internally by the logic of the application, and the HTML page was updated accordingly.

In the project setup, button presses were used to trigger functions preconfigured in the vMix software. These included selecting inputs and starting or stopping a video stream. Polling was utilized to indicate the status of the video stream. This was done to provide

the user with visual feedback for whether the stream was active or not. The user interface is shown in figure 2.

Figure 2 User interface of the web application developed in the project.

As can be seen in figure 2, the status of the stream is displayed in red when active. The same status element is colored a neutral gray once the stream is stopped.

The button elements were designed without clear, explicit borders, a trend gaining popularity through, among others, the Windows 10 platform. Labels were written with bold fonts and capital letters to make them easy to read.

The user interface was made up of a single HTML page. Taking advantage of the Angular framework, the web application was developed as a single-page application (SPA). By using a well-established markup language such as HTML, the user interface could flexibly be tailored according to the specifications of the project.

4.4 Approach to Implementation

Throughout the project, consideration was given to how different parts of the project were approached. In practice, this meant taking notes of how challenges were tackled while carrying out the work. This also included analyzing which the major differences were, if any, in building AV controls using standardized programming languages versus using more traditional manufacturer specific platforms.

From the start of the work process, the value of accurate planning was recognized. Following the notion that thorough planning would save time in later stages of the project, this was strived for by researching and taking notes, before initiating the actual implementation phase.

Putting emphasis on project planning did not, however, spawn a rigid work methodology. On the contrary, Agile principles were utilized from the beginning to the end of the project. The Agile principle most acknowledge in this project was, responding to change over following a plan.

Since the project was carried out as a proof-of-concept, no immediate customer was part of the equation, and thus changes due to fluctuations in customer needs were not confronted. Responding to changes was, however, a large part of the development process itself. For example, a function would be developed and verified as working properly, only to later find a superior way of implementing the same function, which would then require other areas of the application to be altered for the application to retain coherency throughout the code base.

Constructing the application in a modular fashion and utilizing tools, such as the VS Code software, it was noted that changing certain aspects of the application was made easier and faster by using standardized languages as opposed to proprietary programming tools. One example of this, was adjusting the user interface. Using HTML and CSS as markup, the changes are made in code, where all properties are immediately available for editing. In contrast, using specialized visual programming tools, the process may often require multiple mouse clicks to navigate to different submenus, to then be able to modify necessary parameters by filling in form like structures.

Though working directly with standardized programming languages and their accompanying frameworks, the developer is presented with a very large array of tools and options when implementing software applications. This is in contrast with the manufacturer specific, proprietary languages discussed in previous sections, which in the end are higher-level abstractions of standardized languages, and thus more restrictive in nature.

The benefit of visual programming tools like the one's provided by Crestron, AMX or QSC, is the ability to send technologically knowledgeable individuals on manufacturer certification courses, to learn a specific manner of programming logic, without the need for the attendants to master software construction or development principles. As the requirements for AV systems increase in complexity, however, this benefit may need to be weighed against the advantages offered by standardized languages.

Given these facts relating to manufacturer specific platforms, combined with the smaller support networks accompanying them, compared to standardized languages, speaks in favor of using the powerful assets available to languages such as JavaScript, C#, and Java. As previously noted, JavaScript was chosen as the language for this project, in part for these reasons.

It must however be taken into consideration that employing software development into AV projects, will require resources, especially the first time a certain function or structure is created in a standardized language. This was recognized implementing the required functions of the project web application. Using a standardized language such as JavaScript could be described as all but restrictive. Thus, a given desired outcome could have been attained through many different paths. Finding the one best suited for the application at hand, was an essential part of the development process.

4.5 Meeting Requirements

Throughout the project, creating a simple template for presenting the alternative concept for AV control, was a key objective. Today, smartphones are commonplace, and by providing a touchscreen interface, most users of the system would likely feel comfortable operating the setup.

Deviating from a norm, in this case deviating from how control applications are created, the reasons for doing so need to be clearly defined and implemented accordingly. Referring to the proof-of-concept, the intention was to evaluate an alternative method of integrating controls. The deviation in this case, was the tools, protocols and development processes utilized to produce the control application. The goal on the other hand, was to essentially give the users the same experience they were accustomed to when using other platforms.

Thus, the immediate intent of demonstrating the proof-of-concept template, was to present the user with something familiar, using a different platform underneath. During the implementation, a great deal of thought was given to recreating the controls using alternative methods, rather than trying to invent something new.

In addition to providing a touch screen for the user to interact with, the layout of the web application was given special consideration to make the platform more familiar to the user. This was accomplished by using clear and sizeable push-buttons equipped with descriptive labels. The labels were meant to explain the function of the buttons; for example, a button labeled, “input1”, would refer to routing the first input to the output.

Sizing the buttons to the same dimensions, strengthened the cohesion of the page visually. Making the buttons large, made them more suitable for the touch screen setup. Many mobile devices in wide use today have screens smaller than 7 inches and yet employ applications with significantly smaller elements than the application in this project. However, considering that the user interface in the setup will be sitting on a table attached to a cable, rather than being a pocket sized, handheld device, it was concluded that the larger elements make for a better user experience and improved ease-of-use of the application. Several sizes were evaluated before deciding upon the final version.

A core requirement for the development of the web application was to make the code as reusable as possible. In other words, the implementation was carried out to allow for as much of the code as possible to be used in future projects. The basis for this notion was formed through using common design principles. Syntax, for which the purpose was not considered self-evident, was commented, enabling other developers to comprehend what a given portion of the application was initially intended for.

Agile principles were respected during the development of the application. Aiming at working software, some functionalities were rewritten several times to find the best suited solution for the application. Different responsibilities were delegated to different parts of the application, adhering to the principle of single-responsibility. It was noted that this made for a cleaner and better structured code, as well as resulted in an application which could be modified with less effort, in response to potential future changes.

Capitalizing on the notion of ease-of-use, the setup was designed to be installed with minimal effort. The touchscreen was mounted on the Raspberry Pi, and by encasing the two components together, a compact set was achieved.

The installation required connecting the Raspberry Pi and touch screen to the same sub-net as the laptop running the vMix software. When powering on the Raspberry Pi, the server would be automatically started along with a web browser directed the web page constituting the web application developed in this project. This automation was achieved through writing a set of shell scripts for the Raspberry Pi.

In addition, the web application presented a simplified user interface, offering a restricted set of controls. As discussed previously, many AV systems might not necessarily be operated by technically oriented personnel. Thus, for a customer without technical experience to be able to operate such a system, the user interface needs to be made as simple, intuitive, and unambiguous as possible. These requirements were central when developing the web application for the project.

4.6 Testing the Setup

Constituting an essential part of the development process, a substantial amount of time was resourced for testing. This involved testing of both the code constituting the web application as well as the setup as a whole.

JavaScript frameworks Jasmine and Karma were utilized to implement the tests necessary to ensure the inner workings of the application. Karma was used to provide the testing environment, consisting of a chrome emulator, and to run the tests themselves. Unit tests were implemented to verify the functionality intended in separate parts of the

code. These were written using the Jasmine framework. As more tests were written and run successfully, more parts of the code could be verified as syntactically correct.

By developing in this manner, the application became easier to troubleshoot. Making certain that specific functions work in the way they were intended, proved to save a lot of time when making changes to the application, causing the code to break and introducing errors. If a given function was proven to function properly, prior to a modification to said function, but not afterwards, the modification was the most probable cause for the issue.

After confirming smaller parts of the code were functioning properly, the code was also tested using tests conforming to end-to-end testing methodologies. The end-to-end testing was decided to be performed manually, since the web application in question was quite small and straightforward for a user to test thoroughly.

This included making sure that HTTP responses were handled correctly inside the application, and that events, triggered by for example button presses, were handled properly. Automating this process was contemplated, however, considering the small scale of the application, it was recognized that a user could more cost-effectively, manually do the testing of the application flow.

Subsequently the setup was assembled in accordance with figure 3.

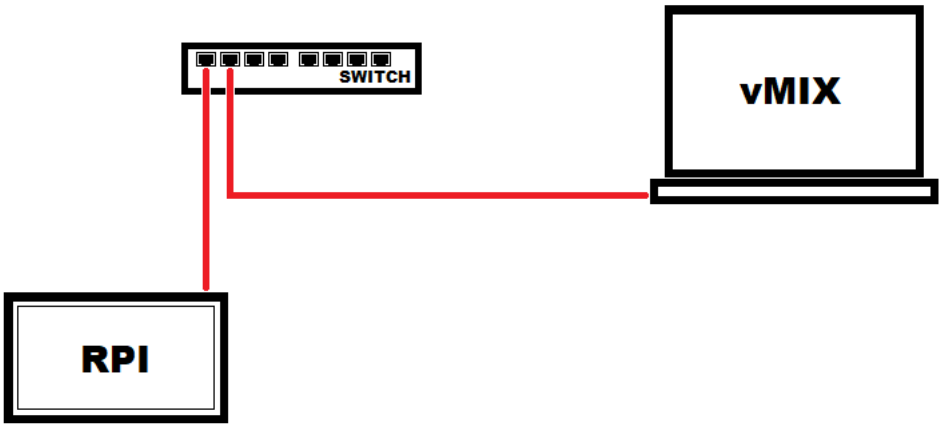

Figure 3 displays the project demonstration setup. Red lines represent standard Cat6 cabling.

In figure 3, the setup for demonstrating the proof-of-concept project, can be seen. This includes the Raspberry Pi and accompanying touch screen, a network switch, and the laptop equipped with the vMix software.

This was followed by conducting system testing. A key factor for successfully testing the setup was identifying common fault scenarios. Example of these might be a circuit breaker tripping or a user accidentally disconnecting the network cable.

System testing included testing the application UI and verifying the effects in the vMix software as well as disconnecting and reconnecting the cable connecting the Raspberry Pi to the network switch. This was done to establish that the web sockets used with the HTTP requests were re-established successfully in case of a scenario including a network issue.

Following this, the Raspberry Pi was also disconnected and reconnected to the power supply. This was to mimic a scenario, for example a power outage, causing the platform to perform a power cycle. The test was considered successful once all core functionality was automatically launched on startup. In addition to launching the HTTP server and the

Chromium browser, this entailed directing the browser to the web application page provided by the server and triggering the browser to fill the entire screen with the page in question.

5 Discussion & Conclusions

The purpose of this study was to examine how software development may be used to create AV controls. Essential objectives included assessing the viability of developing said controls, using a proof-of-concept project as reference. Viability in this context would be evaluated by how successfully standardized hardware and standardized programming languages could be utilized for constructing a platform from which to issue controls to an AV system. A successful implementation would include the alternative platform being able to control a target device or system. In addition, the study aimed at determining if further research in using software development as a method for integrating AV controls would be recommendable based on the template project.

5.1 Recommendations

The project covered in this study was conducted as a proof-of-concept. The concept centered around the notion of building a platform capable of issuing commands to AV target systems through a standard IP network, using standardized hardware and programming languages. The conventional manner of implementing such control systems has historically been to utilize manufacturer specific hardware and proprietary tools.

From the observations made while carrying out the project in this study, a preliminary evaluation of the suitability of the chosen collection of standards-based technologies, for building AV control systems, could be given. This was accomplished through considering the development process itself, the flexibility of the platform used as well as the maintenance aspect of supporting a system built using these methods.

Considering the development process of a software application to perform the same tasks as a given conventional AV system, the methods of implementation are fundamentally different. Using a manufacturer platform, the programmer would typically use visual programming tools, providing graphical elements relating to logical functions, which the

programmer would then assemble and arrange to create desired functionalities. When developing a software application, the developer must first write the functions themselves as well as create a working whole. This will generally require more time to implement than using manufacturer platforms. The advantage of coding the functions from scratch is that the developer retains more control over the process. A graphical element says nothing about the implementation underneath, therefore a programmer will be forced to rely upon the help file for that specific element to determine how it may be used.

Standardized programming languages have fewer restrictions than proprietary counterparts. In the case of JavaScript, which is a very popular language globally, every developer has access to vast amounts of information and support from the global community of other developers. Proprietary implementations usually have more restricted documentation and as the community of users is significantly smaller, support is often restricted to the manufacturer's support department and a few, smaller online forums.

In addition to providing a vast array of built-in functions, given a certain task, a standard programming language is likely to have at least one library or framework in existence to make the implementation of such functionality easier and more structured. Given there is a multitude of alternatives on how to implement a specific desired action, several methods should be tried to determine the one best suited for the task. This translates to a higher cost, in labor, the first time a function is coded. The aim should however be to write reusable components, which would then give return on investment when implemented in future applications.

Testing is paramount when developing software applications. Writing unit tests for the different parts of the application creates an environment which can detect faults by running automated tests. This form of testing along with system testing results in a more stable code base. In addition, proper testing, and designing code for reusability makes the application more responsive to change, and thus more in line with the Agile principles.

The template system constructed for the project, functioned according to specifications. The setup was successful in showing a simple UI for remotely controlling the vMix software running on the laptop, from the Raspberry Pi touchscreen. When buttons were

pressed on the touchscreen, the corresponding functions took place in the vMix software displayed in real time on the laptop.

By implementing the whole setup using standardized hardware, the Raspberry Pi, and a standardized programming language, JavaScript and complementing frameworks, the setup has proven an alternative way of implementing AV controls plausible.

Using standardized hardware has the added benefit of potentially resulting in a substantial cut in cost, compared to manufacturer specific control processors. The price ranges vary notably depending on manufacturer, model, and potential dealership reductions; as such, an exact price comparison was not practical for this study. However, mass produced components on the IT sector are produced in much higher volumes than in the AV industry, as such, prices set by AV manufacturers for their devices are often comparatively higher.

The security aspect is beyond the scope of this project, as the focus has been on providing alternative ways of implementing controls. The security features implemented in the setup were limited to password protecting the Raspberry Pi. However, if the methods described in this study were to be implemented in a real-world scenario, security should be an immediate concern. Security in this context might involve providing the web app with a login page or perhaps locking the touchscreen with a password.

Based upon the template project setup in this study, standardized programming languages and standardized hardware may successfully be used to implement AV controls, and thus offers an alternative to conventional approaches. However, it is worth noting that the proof-of-concept project presented was a small-scale system, and depending on the case in question, a software development approach may or may not be suitable. Using standardized programming languages requires the AV programmer to possess the expertise of a developer, something not necessarily required when using visual programming tools. In addition, the development process requires careful planning and sufficient time to test and execute properly, to produce a reusable codebase, responsive to change. The findings in this study suggests that further research and development of AV controls using these methods is recommendable in order to define to which extent the alternative approach may be utilized in practice.

5.2 Closing Words

The AV industry is currently undergoing a substantial shift, adopting principles and standards from the IT sector. In practice, this may be seen as an increase in the implementation of IP based technologies and software-based solutions. By constructing a platform for controlling AV equipment, using standardized hardware and software, the project carried out in this study has shown how software development may be used for implementing AV controls. In the spirit of the industry-wide change, this project has presented an alternative approach to the conventional method of building AV control systems.

6 References

[1] Backus C, Hatz D, DeVito J, Greenblatt S. Sate of Control 38: Code Switching. State of Control [internet]. AVNation. 2017 [cited 6.12.2017]. Available from: https://av-nation.tv/podcast/state-of-control-38-code-switching/

[2] Mitchell K. Can a Hardware-Free AV Industry Exist? [internet]. AVNetwork: Home. 2014 [cited 15.12.2017]. Available online 17.7.2014 from: http://www.avnetwork.com/article.aspx?articleid=119943

[3] Hagan D. 3 Reasons an Experienced Integrator Embraced AV over IP [internet]. IVCI Collaborate Anywhere. 2017 [cited 9.12.2017]. Available online 10.5.2017 from: https://ivci.com/3-reasons-for-av-over-ip/

[4] Dallas Semiconductor. Application Note 83: Fundamentals of RS-232 Serial Communications. 1998. Available from: https://www.lammertbies.nl/download/dallas-appl-83.pdf

[5] Clauser G. Is IP Control Replacing IR? [internet]. Electronic House: Smart Home. 2012 [cited 9.12.2017]. Available online 7.12.2012 from: https://www.electronichouse.com/smart-home/is-ip-control-replacing-ir/

[6] Dorsch J. Older Communication Standards Still Compete with USB [internet]. Mouser Electronics, Applications & Technologies. 2017 [cited 9.12.2017]. Available from: https://www.mouser.fi/applications/article-rs232-still-competes/

[7] Designlines Test & Measurement. The Serial Port: Never Say Die [internet]. EETimes Connecting the Global Electronics Community. 2013 [cited 9.12.2017]. Available online 28.8.2013 from: https://www.eetimes.com/author.asp?section_id=36&doc_id=1319282

[8] Shamir D. Leveraging IP and the Cloud for AV Control [internet]. AVNetwork: AVTechnology. 2017 [cited 10.12.2017]. Available online 8.5.2017 from: http://www.avnetwork.com/article.aspx?articleid=126363

[9] ISI University of Southern California. Transmission Control Protocol: Darpa Internet Program Protocol Specification [internet]. Internet Engineering Taskforce: Request for Comments: 793. 1981 [cited 10.12.2017]. Available online from: https://tools.ietf.org/html/rfc793

[10] Postel J, ISI. User Datagram Protocol [internet]. Internet Engineering Taskforce: Request for Comments: 768. 1980 [cited 10.12.2017]. Available online from: https://tools.ietf.org/html/rfc768#ref-2

[11] Highley J. There Is No Right Way [internet]. Blogger.com. 2012 [cited 10.12.2017]. Available online 1.5.2012 from: http://thereisnorightway.blogspot.fi/2012/05/api-example-using-rest.html

[12] Deering S. Do you know what a REST API is? [internet]. Sitepoint: JavaScript. 2012 [cited 10.12.2017]. Available online 7.12.2012 from: https://www.sitepoint.com/developers-rest-api/

[13] Fielding R. Architectural Styles and the Design of Network-based Software Architectures [internet]. University of California, Irvine. 2000 [cited 10.12.2017]. Available online from: https://www.ics.uci.edu/~fielding/pubs/dissertation/fag.htm

[14] Marshall D. Remote Procedure Calls [internet]. Cardiff School of Computer Science & Informatics. 1999 [cited 14.12.2017]. Available from: http://users.cs.cf.ac.uk/Dave.Marshall/C/node33.html

[15] Fielding R, UC Irvine, Gettys J, Compaq/W3C, Mogul J, Compaq, Frystyk H, W3C/MIT, Masinter L, Xerox, Leach P, Microsoft, Berners-Lee T, W3C/MIT. Hypertext Transfer Protocol HTTP/1.1 [internet]. Internet Engineering Taskforce: Request for Comments: 2616. 1999 [cited 10.12.2017]. Available online June 1999 from: https://www.ietf.org/rfc/rfc2616.txt

[16] Rouse M. Definition Polling [internet]. Whatis.com. 2005 [cited 6.1.2017]. Available online: http://whatis.techtarget.com/definition/polling

[17] Wisehart C, Barbour M, Mattson G, Christ B, Christoff J, Freshman S. Liberating AV Control from Dedicated Hardware [internet]. Webinar Sponsored by QSC, hosted by NewBay Media. 2017 [cited 15.12.1017]. Available online 14.12.2017 from: https://www.youtube.com/watch?v=rHmET04ek_4

[18] Alonso A, Sánchez L, Groba A, Pickin S, Martinez N, de la Puente J. Development of computer control systems with hardware-software codesign. IFAC Proceedings Volumes. [serial online] 1999;32(2):8722-8727. DOI:10.1016/S1474-6670(17)57488-1 Available online 7.7.2017 from: https://doi.org/10.1016/S1474-6670(17)57488-1

[19] Paul German. Time to bury dedicated hardware-based security solutions. Network Security. [serial online] 2017;2017(8):13-15. DOI:10.1016/S1353-4858(17)30083-1 Available from: https://doi.org/10.1016/S1353-4858(17)30083-1

[20] Crestron Technical Documentation Department. Crestron SIMPL+ Software Language Reference Guide [internet]. 2003 [cited 16.12.0217]. Available online from: https://www.crestron.com/downloads/pdf/product_misc/simpl_plus_language_ref.pdf

[21] AMX by Harman. Netlinx Programming Language [internet]. AMX by Harman. 2016 [cited 16.12.2017]. Available online from: https://trade.amx.com//assets/manuals/NetLinx.LanguageReferenceGuide.pdf

[22] AMX by Harman. When Industries Collide (white paper) [internet]. AMX by Harman: AV&IT, Understanding the Implications of AV/IT Convergence. 2015 [cited 16.12.2017]. Available online from: https://www.amx.com/en-US/premium-content_when-industries-collide

[23] Klang M. Free Software and Open Source: The Freedom Debate and It's Consequences [internet]. First Monday, Peer Reviewed Journal On the Internet. 2005 [cited 20.12.2017]. Available online from: https://web.archive.org/web/20050908104318/http://www.firstmonday.org:80/issues/issue10_3/klang/index.html

[24] Open Source Initiative. The Open Source Definition [internet]. Open Source Initiative. 2007 [cited 19.12.2017]. Available online 22.3.2007 from: https://opensource.org/definition

[25] Stallman R. Free Software, Free Society: The Selected Essays of Richard M. Stallman [internet]. gnu.org. 2016 [cited 19.12.2017]. Available online from: https://www.gnu.org/philosophy/free-software-for-freedom.html

[26] Coxon M. Open Source Culture [internet]. Rave Publications: Blogsquad. 2015 [cited 19.12.2017]. Available online 20.11.2015 from: http://www.ravepubs.com/open-source-culture/

[27] McGarvey J. Why Open and Authentic Standards are Critical to the future of the Broadcast and Media Industry [internet]. Imaginecommunications.com 2016 [cited 20.1.2018]. Available online 21.3.2016 from: https://www.imaginecommunications.com/resources/blog/why-open-and-authentic-standards-are-critical-future-broadcast-and-media-industry

[28] Vezina A. The AV Industry Needs to Open Up [internet]. Rave Publications: Blogsquad. 2015 [cited 19.12.2017]. Available online 13.11.2015 from: http://www.ravepubs.com/the-av-industry-needs-to-open-up/

[29] Jörger T, Schlagenhauf J, Rosch E, Ernestus M, Dold M, Greulich J, Schleusener B, Reimer J, Diehl M. TOPCORE open modular platform for real-time control of experimental setups. IFAC-PapersOnLine. [serial online] 2017;50(1):11287-11294. DOI:10.1016/j.ifacol.2017.08.1641 Available from: https://doi.org/10.1016/j.ifacol.2017.08.1641

[30] Polley B. Ministry of Justice Open Source Discussion Paper [internet]. New Zeeland Ministry of Justice: Discussion Paper. 2007 [cited 20.12.2017]. Available online from: https://nzoss.org.nz/system/files/moj_oss_strategy_1.0.pdf

[31] Agile Alliance. What is Agile Software Development? [internet]. Agilealliance.com. 2017 [cited 23.12.2017]. Available online from: https://www.agilealliance.org/agile101/

[32] Beck K, Beedle, Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin R, Mellor S, Schwaber K, Sutherland J, Thomas D. Twelve Principles of Agile Software [internet]. Agilemanifesto.org. 2001 [cited 23.12.2017]. Available online from: http://agilemanifesto.org/principles.html

[33] Revell M. What Is Visual Programming? [internet]. Outsystems.com. 2018 [cited 7.1.2018] Available online from: https://www.outsystems.com/blog/what-is-visual-programming.html

[34] Crestron SIMPL Windows Software Installation & Operations Guide [internet]. Crestron.com. 2018 [cited 7.1.2018]. Available online from: https://www.crestron.com/getmedia/4ee6b512-f741-46c7-af71-b809f7562351/mg_io_sw-simpl_1

[35] Kult B, HGA Architects, and Engineers, AVNation Organization. Hardware/Software Codec Comparison [internet]. AVNation. 2017 [cited 6.12.2017]. Available online 30.10.2017 from: https://avnation.tv/2017/10/hardwaresoftware-codec-comparison/

[36] Adafruit, About Us [internet]. Adafruit.com. 2017 [cited 29.12.2017]. Available online from: https://www.adafruit.com/about

[37] Raspberry Pi 3 – Model B [internet]. Adafruit.com. 2017 [cited 29.12.2017]. Available online from: https://www.adafruit.com/product/3055

[38] Arduino, About Us [internet]. Arduino.cc. 2017 [cited 29.12.2017]. Available online from: https://www.arduino.cc/en/Main/AboutUs

[39] About Beagleboard [internet]. Beagleboard.org. 2017 [cited 29.12.2017]. Available online from: https://beagleboard.org/about

[40] SSH (Secure Shell) [internet]. Ssh.com. 2017 [cited 5.1.2017]. Available online from: https://www.ssh.com/ssh/

[41] FileZilla The Free Ftp Solution [internet]. Filezilla-project.org. 2017 [cited 5.1.2017]. Available online from: https://filezilla-project.org/

[42] Pi Foundation Display – 7” Touchscreen Display for Raspberry Pi [internet]. Adafruit.com. 2017 [cited 29.12.2017]. Available online from: https://www.adafruit.com/product/2718

[43] TriCaster Mini [internet]. Newtek.com. 2017 [cited 30.12.2017]. Available online from: https://www.newtek.com/products/tricaster-mini/#tricaster-mini-top-features

[44] About NewTek [internet]. Newtek.com. 2017 [cited 30.12.2017]. Available online from: https://www.newtek.com/company/about-us/

[45] vMix Live Production & Streaming Software [internet]. Vmix.com. 2017 [cited 3.1.2017]. Available online from: https://www.vmix.com/

[46] vMix User Guide [internet]. Vmix.com. 2017 [cited 3.1.2017]. Available online from: https://www.vmix.com/help20/

[47] IoT Applications [internet]. Microsoft.com. 2017 [cited 30.12.2017]. Available online from: https://www.microsoft.com/net/learn/apps/iot

[48] Unit Testing [internet]. Angularjs.org. 2017 [cited 30.12.2017]. Available online from: https://docs.angularjs.org/guide/unit-testing

[49] Single-page Applications [internet]. Beginner's Guide to Web Development. 2018 [cited 6.1.2018]. Available online from: https://www.codeschool.com/beginners-guide-to-web-development/single-page-applications

[50] Van Baelen R. Agile Manifesto Wallpapers [internet]. Wordpress.com. 2012 [cited 1.1.2018]. Available online 18.10.2012 from: https://rafvb.files.wordpress.com/2012/10/manifestowallpaper1024.jpg

[51] Metz S. SOLID Object-Oriented Design [internet]. GORUCO (Gotham Ruby Conference). 2009 [cited 11.1.2018]. Available online from: http://confreaks.tv/videos/goruco2009-solid-object-oriented-design

[52] Visual Studio Code [internet]. Code.visualstudio.com. 2018 [cited 5.1.2017]. Available online form: https://code.visualstudio.com/

[53] Gigabit Smart Managed Plus Switch Series [internet]. Netgear.com. 2018 [cited 6.1.2017]. Available online from: https://www.netgear.com/business/products/switches/webmanaged/GS108PE.aspx#tab-techspecs