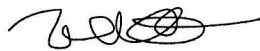


**Safe Shop: A Hardware and Software Solution for Workshop Management in
School and Community Workshops**

by
Joanna Sumner

An Undergraduate Thesis Submitted to
The University Honors Program
Auburn University at Montgomery

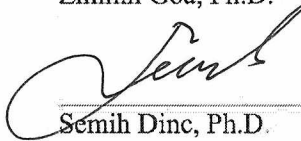
In partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science



4/26/21

Zhimin Goa, Ph.D.

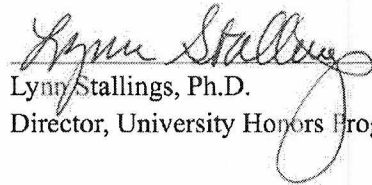
Date



4/27/21

Semih Dinc, Ph.D.

Date



4/26/21

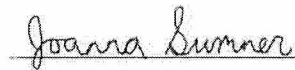
Lynn Stallings, Ph.D.

Date

Director, University Honors Program

© Copyright by Joanna Sumner, April 20, 2021

I understand that my project will become part of the permanent collection of the Auburn University at Montgomery Library, and will become part of the University Honors Program collection. My signature below authorizes release of my project and thesis to any reader upon request.



**Safe Shop: A Hardware and Software Solution for Workshop Management in
School and Community Workshops**

by
Joanna Sumner

An Undergraduate Thesis Submitted to
The University Honors Program
Auburn University at Montgomery

In partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science

TABLE OF CONTENTS

LIST OF FIGURES	3
I. INTRODUCTION	4
II. RELATED WORK	6
A. AccessPack by Sole Digital	6
B. Build Brighton Member and Access Control System by Arthur Guy	8
C. Card Reading Power Switch by Casey Horton	9
D. The Gap	10
III. SYSTEM FEATURES	12
A. Tool Access Control	12
B. User Management	13
C. Tool Management	13
D. Training Management	14
E. Device Management	14
F. Expandable, Repeatable, and Economical	15
IV. HARDWARE DESIGN	16
A. Raspberry Pi	17
B. Relay	18
C. RFID Reader	18
D. OLED Screen	19
E. Membrane 3x4 Matrix Keypad	20
F. LED	20
G. Switch	21
H. Case	21
V. SOFTWARE DESIGN	24
A. STAC Device	25
B. Web Portal	26
C. Amazon S3 (Simple Storage Service)	26
D. Amazon API Gateway, Amazon Lambda, and Amazon DynamoDB	27
E. Amazon Cognito	29
F. Amazon Identity and Access Management (IAM)	29
G. Amazon CloudWatch	29
H. Amazon CloudFormation	30
I. Summary	30
VI. IMPLEMENTATION	32

A. Source Code	32
B. Documentation	32
C. GEARS, Inc. Workshop	33
VII. CONCLUSION	35
REFERENCES	37

LIST OF FIGURES

Figure 1: STAC Device Hardware Wiring Diagram	16
Figure 2: STAC Device Exterior	22
Figure 3: STAC Device Interior	22
Figure 4: Safe Shop Workshop Management System Software Diagram	25
Figure 5: STAC Device in the GEARS, Inc. Workshop	33

I. INTRODUCTION

For middle and high school students, gaining technical skills through a robotics, woodworking, or metalworking class is a rewarding experience. For members of the community, a makerspace can provide access to tools which fuel creativity and DIY projects. However, these opportunities must be safe experiences. The product of this thesis is an access control system for workshops that can be utilized to prevent untrained individuals from using potentially dangerous workshop tools. Safety includes multiple layers. This system provides an additional layer of protection, in addition to adult supervision and other measures, for school and community workshops.

The widespread availability of inexpensive microcomputers and open source software provides the resources for an inexpensive hardware and software solution for shop tool access control. The Safe Shop system provides a unique solution composed of a set of devices, one connected to each tool, and a web application. The Safe Tool Access Control (STAC) devices, each equipped with a card reader and keypad, require individuals to be authenticated before granting users access to a tool. Access is granted or denied by turning on or off power to a tool. The STAC devices all reference a common database, the contents of which is controlled by a web application. The web application allows administrators to track users and their safety training in order to grant tool access to users appropriately. This solution includes unique features not part of existing related work.

The system has been fully developed and implemented in a local robotics team's workshop to showcase the functionality. The design and development documentation and source code files are provided as an open-source project on the web so the system can be

utilized by other school and community workshops interested in adding another layer to their safety protocols.

This document first provides an overview of work related to the Safe Shop system, highlighting the gap between existing work and the goal of this project. The features of the system are then described. Following this, the system hardware and software design are detailed. The implementation section provides links to all the resources required to implement the system. The document concludes with plans for future work and expansion of the system.

II. RELATED WORK

The challenge of access control is universal. When starting a new job, an employee may be issued a card used to open the door of the office building each day. That card would almost certainly be making use of Radio Frequency Identification (RFID) technology. Tutorials for connecting an RFID card reader to a microcomputer are easy to find as are tutorials for creating various kinds of web applications and web services. While the idea of using RFID cards to grant access to something and administering that system through a web application is not new, the concept of using such technology for a school or community workshop is not widely documented. The focus of this project is on providing a tool access control system for school and community workshops. Since these entities may not have many resources, the system must be as inexpensive as possible. Also, particularly given that this is a system used by creative makers, it must be open to customization by the makers so they can make it work best for them.

In researching existing solutions that meet this project's goal or similar goals, one commercial equipment access control solution, an open-source makerspace access control system, and a hobbyist's power access control project were found. All solutions have similar goals in mind to that of this project, however, they all lack certain key features. Analysis of these solutions is the focus of this section.

A. AccessPack by Sole Digital

AccessPack is a commercial equipment access control system designed by an Australian company, Sole Digital [1]. The device utilizes a smart card reader to allow

users to swipe their card for access. The device can be fitted onto a wide variety of tools, from cranes to hand-held power tools, so long as they run on electricity and have an on/off switch. One of the AccessPack features highlighted by Sole Digital is the ability to use the device without an internet connection. Each device is configured for the equipment on which it is installed, and then users' cards are programmed with the access information including which tools they can utilize and expiration dates for tool access. Companies that use the device utilize AccessPack software to program cards for access.

The AccessPack system is an innovative solution for companies in need of an equipment access control system. However, it lacks desirable features for school and community workshops. The software is not extensible so that organizations can integrate it into their existing systems as they see fit. In particular, there is no way to fully integrate the access control system with safety training records. While beneficial for companies with remote work spaces, the device's lack of reliance on web services for access data is a potential security problem. Expiration dates can be applied to cards, but since the cards store access data on them rather than reference a web service, revoking access for a particular user requires updating the user's card. If a card is lost or an individual's access rights terminated, the only way to ensure access is actually revoked is to recover the card. Otherwise, a user with the card can still access the tool until access has expired. Another potential issue for this commercial solution is the cost, particularly since the device is manufactured in Australia and importing it from there can add difficulty.

The AccessPack is a relevant commercial solution, and looking at the provided features provides insight into how such a system can be applied to many types of equipment. However, for the purposes of this project, it is not a viable solution.

B. Build Brighton Member and Access Control System by Arthur Guy

Build Brighton is a Makerspace in Brighton, England [2]. Arthur Guy, a trustee for the makerspace, set up an access control system for the space which includes devices for each tool that reference a database in the cloud to grant access appropriately. These devices are coupled with a web application used by administrators to grant access appropriately. Mr. Guy documented the project through a series of YouTube videos [3], his own website [4], and two GitHub repositories [5], [6].

The BBMS, Build Brighton Member System, web application provides users the ability to sign up for membership, pay membership fees, manage their profile, and see other members [5]. The application also includes a role system, equipment usage log, a voting system for proposals, and more. This system interfaces with the RFID Access Control system, which regulates access both to the makerspace doors and to the individual tools in the space [6]. The device is made from a custom microcontroller board, an RFID card reader, a buzzer, and an LED, all of which are enclosed in a 3D printed case [3].

The system is very similar to the goal of this project and the documentation suggests it could be utilized for other community makerspaces. However, there are several features missing. The system does have the ability to track what individual trained another, but there is not a comprehensive safety training record system particularly desirable in an educational setting. The system lacks a second factor for authentication, which is particularly desirable to provide additional security so a user can't just swipe someone else's card without also knowing their PIN number. The system also lacks sufficient documentation for easily getting it up and running. The infrastructure resources

are not clearly documented, and it would take some effort for an interested party to read through all the code and determine how it all worked. Additionally, the hardware requirements, though discussed in the YouTube video series [3], are not fully documented. In particular, Mr. Guy used a custom board which is not documented, so it would be difficult to replicate the system.

The Build Brighton system has many useful features and the available documentation and source code provide a helpful reference point for developing a similar system. However, there are many desired features missing. Additionally, Mr. Guy used a traditional architecture for the backend application. As discussed in the Software Design section later on, for the scope of this project, a serverless architecture is more cost effective. Since the Build Brighton system is based on a different architecture and lacks a number of desired features, it served only as an initial reference when designing the Safe Shop system, but was not referenced further.

C. Card Reading Power Switch by Casey Horton

The Card Reading Power Switch, created by Casey Horton [7] and documented via an Instructable [8], is a simple example of an access control system. This switch reads magnetic stripe cards and then checks onboard storage to determine if the card is on the list of authorized users or not. The switch controls power for a shop's main power system, rather than an individual tool. The system does not communicate with any web services. Rather, one administrator identified at setup with their own card can swipe additional cards to register them with the system as authorized. The system can also be activated with a key. The system tracks who turns on the shop power and then will

automatically shut off the power at the end of the workday if not manually shut off already with the button. The entire device is run by a microcontroller and housed in a steel box with a screen, keypad, LED, and button accessible from outside the box.

While the device is not designed for tool-specific access control and does not incorporate a comprehensive user or training management system, it is another interesting example of an access control device. In particular, it provides a neat setup for housing the device in a metal box. Though the STAC device is currently housed in a plastic VHS tape case, future iterations could involve experimenting with other types of cases. Horton's project provides some ideas for how the device case could be improved in the future.

D. The Gap

As described above, there are a variety of solutions with similar goals to that of this project. However, they all lack certain features and pose challenges for replication. Nonetheless, they provided inspiration for how the project was pursued. In particular, the YouTube videos from Arthur Guy for Build Brighton [3] provided a number of ideas that were helpful during the initial hardware design for the STAC device. These are noted below in the design sections. Additionally, identifying the gaps of other solutions ensures that this project can fill in those gaps appropriately. In terms of gaps there are two significant ones. The most significant is a lack of documentation. Arthur Guy's project is the closest to the desired solution, but it is not documented well enough for less experienced individuals to pick up and implement. Though some familiarity with software and hardware is required, this project's extensive documentation ensures that it is accessible for others wishing to implement it. The second main gap is the lack of a

safety training record system, and this feature is included in the Safe Shop system.

Details of this and the other features of the system now follow.

III. SYSTEM FEATURES

The Safe Shop system version 1.0.0 provides the fundamental features needed for effective tool access control. The list of features described below was driven by consultations with the initial target user organization for the project, the GEARS (General Engineering And Robotics Specialists), Inc. robotics team. GEARS, Inc. is a middle and high school robotics team which spends a great deal of time in the workshop using tools. The team's mentor identified what the team was looking for in a tool access control system. Using this insight, and the author's own experience on the robotics team, the features required to meet the users' goals were identified.

A. Tool Access Control

The core feature of the Safe Shop system is the ability to allow or block access to a tool based on whether a user is authorized to use the tool or not. The key enabler of this feature is the Safe Tool Access Control (STAC) device. The workshop tool is plugged into the device, and the device then turns power on and off to the tool. By default, power to the tool is off. To use a tool, a user scans their card and enters their PIN number. If the user is allowed access to the tool, power to the tool is turned on. If the user is not authorized for the tool to which the STAC device is associated, then power to the tool remains off. Users are required to leave their card in the STAC device as long as they are using the tool. The STAC device checks at a predefined time interval whether the authorized user's card is still in the device. If either the authorized user's card is no longer detected or a different card is detected, power to the tool is shut off and the next user prompted to scan their card.

B. User Management

The Safe Shop web portal provides administrators with the ability to manage workshop users. Administrators can add new users and view, edit, and delete existing users. User profile information, including id, name, and PIN number, is required. Based on the tools available, administrators can mark users as having completed safety quizzes and hands-on training for specific tools. If a user has completed both a safety quiz and hands-on training for a specific tool, the user can also be marked as having authorization to use the specified tool. In addition, based on the general training available, administrators can mark users as having completed general training courses. All of the aforementioned information is set using the web portal. However, to use the STAC device, a user must have a card registered with their user profile. The STAC device provides administrators the ability to specify a user ID, scan a card, and then update the database to associate the user with that specific card.

C. Tool Management

The Safe Shop web portal also allows administrators to manage workshop tools. Administrators can add new tools and view and edit existing tools. Administrators are not given the ability to delete tools since user training data is tied to tools and such training data should be available for record keeping purposes. Administrators can specify a tool ID, name, type (stationary or hand-held), a restriction level (unrestricted, 15+, adult only), and whether the tool is enabled or not. The list of tools is viewable when editing users so that users' tool training and access permissions can be set.

D. Training Management

The final main feature of the Safe Shop web portal is to allow administrators to manage general training courses. In addition to tool specific training, workshop administrators likely require additional general training such as general workshop safety and first aid training. The training management feature allows administrators to add new training and view and edit existing training. Administrators are not given the ability to delete training since user training data is tied to the training items and such training data should be available for record keeping purposes. Administrators can specify a training ID and name for each training. The list of training is viewable when editing users so that users' training statuses can be set.

E. Device Management

While all user, tool, and training data is stored in a database, each STAC device has two configurable properties. The first is which tool the STAC device is assigned to. Access is granted to users based on the tool the STAC device is associated with. Certain tools can be configured for different purposes utilizing different attachments. Depending on the attachment, the item would be considered a different tool. The STAC device allows for the tool ID with which the device is associated to be set by an administrator with just a few presses on the device keypad. Access to this administrator feature and to the ability to scan and assign cards to users is blocked by an administrator PIN number. This PIN is set locally on the device during setup, but the PIN can be changed with a few presses on the device keypad after the existing PIN is entered.

F. Expandable, Repeatable, and Economical

In addition to the features listed above, key considerations during the development of the system were making the system expandable so more features could be added, repeatable so groups other than GEARS could reference and apply it if desired, and economical so that GEARS and others with limited budgets could both implement and maintain it. As will be detailed in the conclusion, the author has many ideas in mind for ways in which the system can be expanded. As is noted in the implementation section, the system has been documented in detail, to include cost data, so that others can use it. All features listed here and above served as the driving force behind the design detailed in the following sections.

IV. HARDWARE DESIGN

As noted earlier, the core feature of the Safe Shop system, access control, is enabled by the Safe Tool Access Control (STAC) device. The design for the Safe Shop system began with the design of the STAC device hardware. Careful consideration was given to what hardware components were needed and the specifications of those components. The following wiring diagram was created as part of the component design process to ensure all components could be connected appropriately and to provide a reference for implementation.

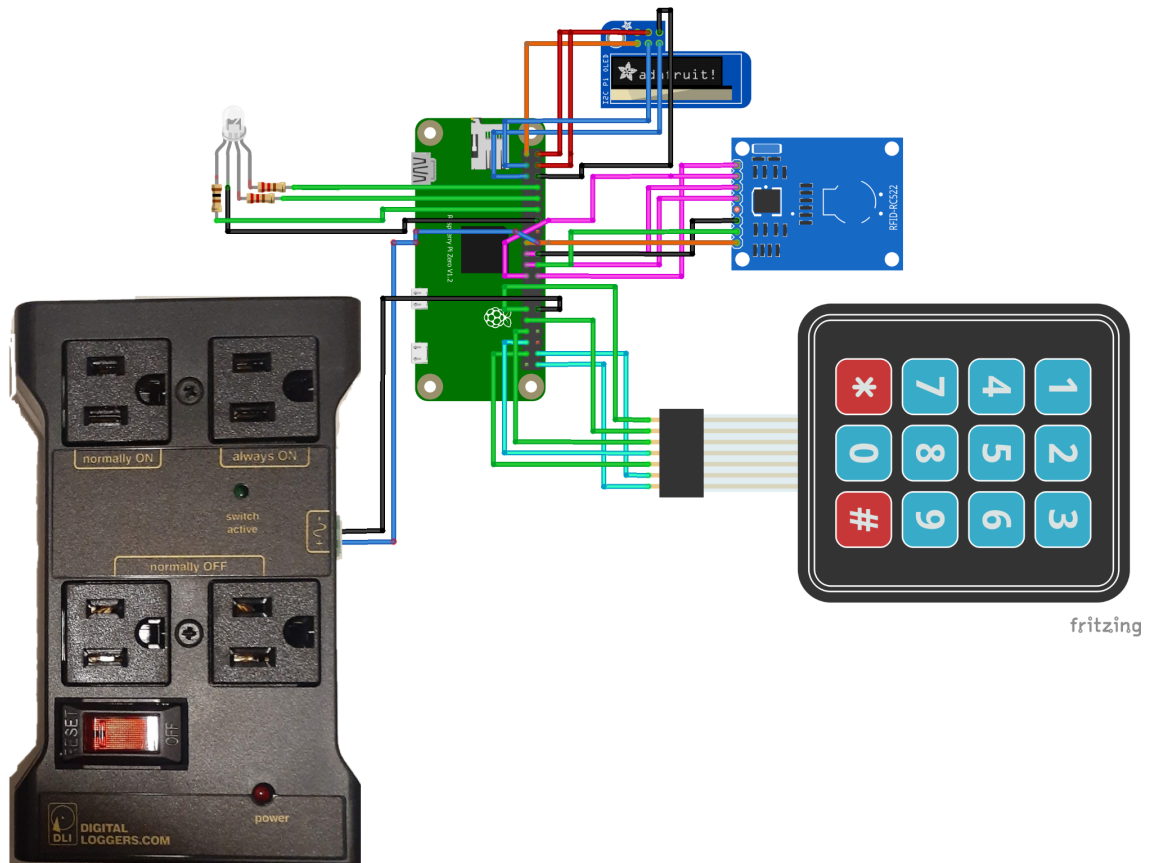


Fig. 1 A hardware wiring diagram for the Safe Shop Access Control (STAC) device. The figure was created using the fritzing software [9].

Depicted in the above figure are all the major components of the STAC device. Not depicted are the power cords for the relay and the Raspberry Pi microcomputer, the micro SD card for the Raspberry Pi, and the on/off switch which is connected to the Raspberry Pi power cord. Below are descriptions of the components, their uses, and their specifications.

A. Raspberry Pi

The Raspberry Pi is a microcomputer that serves as the mastermind behind the device. It provides a 40 pin header to allow multiple input/output devices to connect. The Pi can be run as a stand-alone computer by connecting a mouse, keyboard, and monitor or it can be run in headless mode without a graphical user interface (GUI) and connected to remotely over a wireless network. The Pi can run a Linux operating system which makes it very flexible and capable of running many different programming languages and utilizing many libraries quite easily [10].

Though the Raspberry Pi, specifically the Raspberry Pi Zero W, was chosen for this project, the Arduino, a microcontroller which is also popular for small electronic projects, was also considered as a possible controller for the STAC device [11]. The Pi was chosen because it provides much more out-of-the-box capability than the Arduino does. The Pi is a standalone computer, while the Arduino is not. While the full capabilities of the Pi may not be necessary, those capabilities allow the STAC device to be easily expanded later. Additionally, the Pi has many more input/output pins than the Arduino board. Most likely additional components would have to be added to the Arduino to enable it to support all the input/output devices required for the project. The

low-end Raspberry Pi and Arduino boards are comparable in price. However, Wifi is a necessity which means at least either the Raspberry Pi Zero W (~\$10) or Arduino Nano 33 IOT (~\$18) is required [12], [13]. In the end, with all its extra functionality, the Raspberry Pi comes out cheaper than the Arduino, so it was chosen as the base for the STAC device.

B. Relay

While the Raspberry Pi is the brains of the operation, also required is a relay to interface with the tools for which access is being controlled. A relay is simply a switch controlled by an electrical signal. When a signal comes on, the outlets are switched on or off depending on the configuration of the relay. The main criteria for the project was that the relay be rated to a high enough current to handle large power tools in the shop. The relay selected for this project was the IoT Relay from Digital Loggers, Inc., which is rated for up to 12 Amps and was previously used for a robotics team project with great success [14].

C. RFID Reader

There are many tutorials for creating an RFID card reader, many of which have references to the MFRC522 reader. This reader is compatible with the MIFARE communication protocols [15]. MIFARE is a popular option for smart cities and provides smartcard technology [16]. It is important to acknowledge that the MFRC522 is not the most up-to-date reader/writer model. However, the MFRC522 is widely available as a complete module ready to plug and play for a cheap price. There are also a number of

tutorials for it. Due to its low cost and the availability of resources for it, the MFRC522 reader/writer was selected for this project.

D. OLED Screen

Providing instructions and feedback to the user is important, and for this reason a digital display is included in the project. Initial research for displays led to consideration of a 20x4 character LCD display [17]. However, after reviewing Arthur Guy's discussions of the Build Brighton project [3], a small OLED screen similar to what he was using was considered [18]. Comparing prices and functionality made it clear that the OLED screen is the better option. The screen is small, however, it has the ability to display far more pixels than the character display enabling drawing text of varying sizes and even drawing various shapes. Following the recommended setup by the screen manufacturer allows display of 4 lines of 20 characters each. In contrast, the character LCDs take up a much greater amount of space and the larger one which can display 4 lines of 20 characters is still limited to only displaying characters in the library. Due to a sale from the manufacturer, the cost of the OLED screen was \$8 as opposed to \$18 for the 20x4 screen. A smaller character LCD screen would have run \$10, but that would have been too small to fit adequate messaging on the device [19]. Finally, arguably the most important decision factor is the ease of installation. The OLED screen is designed specifically as an add-on for the Raspberry Pi. It can be directly plugged into the input/output header on the Raspberry Pi with no wires needed. Additionally, the manufacturer provides a quick-state guide for getting the display running [20]. With these factors at play, the OLED screen was an easy choice.

E. Membrane 3x4 Matrix Keypad

One of the unique features of this project is including a second factor of authentication. It is not enough for a user to have a badge; they have to know the PIN that corresponds with their badge. For the sake of security, it is important to use a keypad with all 10 numbers. This greatly increases the possible PIN combinations. Additionally, it is important to protect the product. In a workshop environment there may be dust particles, so a membrane keypad was chosen [21]. The alternative was a more traditional, raised button design, however this was almost twice as expensive as the membrane keypad and does not protect the keypad from a dusty shop environment [22]. The keypad manufacturer provides a quick start guide to get the keypad working with the Raspberry Pi. The availability of this resource was also a factor in purchasing this particular model [23].

F. LED

As noted above, the OLED screen is rather small. Utilizing a tip gleaned from Mr. Guy's video discussions of his project [3], an LED is also included on the device to augment the feedback given to the user on the screen. For the sake of saving space, a three-color LED was chosen so that any color can be specified via RGB (Red, Green, Blue) values and utilized to convey different statuses. The programming is pretty simple thanks to a working example that is available for an input/output library for the Raspberry Pi [24].

G. Switch

The Raspberry Pi does not come with a built-in power switch. Power comes on when it is plugged in, and the Pi shuts down when power is cut off or the command given by a user. Because the Raspberry Pi is completely enclosed along with the rest of the components, it was important to include a sizable exterior switch. The switch is connected to the Raspberry Pi power cord to break or connect the flow of electricity, thereby turning on and off the device.

H. Case

The entire device, consisting of all the parts previously mentioned along with jumper wires between components and the necessary power cords, must be contained within some sort of enclosure to avoid tampering and to protect the device. Arthur Guy, the maker of the Build Brighton system, utilized a 3D printed case [3], [6]. Casey Horton, who made the Card Reading Power Switch, utilized a steel box [7]. Both options were considered, however, a VHS tape case was ultimately chosen. The case is small and lightweight but easily configurable to support the different inputs and outputs for the STAC device. The case option is also cheap. The case provides access to the on/off button, the OLED screen, the keypad, LED, and the RFID card reader. The only component exterior to the case is the relay.

All components were integrated according to design. The pictures below show the final product.



Fig. 2 The exterior view of the completed STAC device

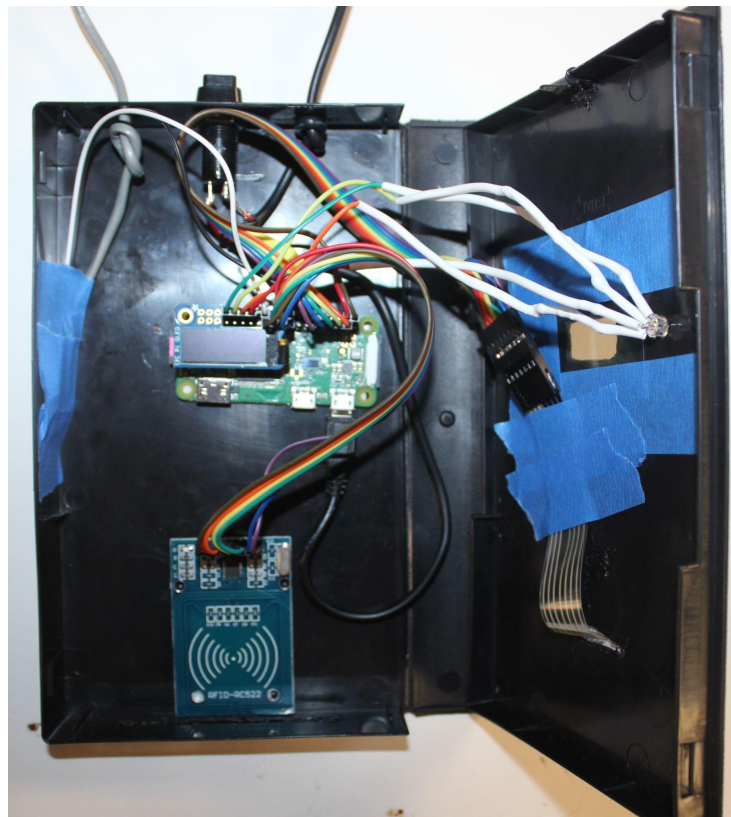


Fig. 3 The interior view of the completed STAC device.

Note the two cables coming out of the case are the power cord and the control cord which connects to the relay. Both cords are long to allow the case to be mounted on a tool and the relay to be placed on the floor next to the tool. The inclusion of the nightlight on the relay illustrates how a tool would be plugged in to connect to the STAC device.

V. SOFTWARE DESIGN

The software to support the Safe Shop system can be divided into three parts. The first is the program running on the Raspberry Pi which interacts with the input/output components and can send and receive data from web services in the cloud. The second is the web portal that is accessible from the cloud by users via the internet and is served to their web browsers. The web portal also interacts with the web services in the cloud. The third piece is the web services themselves in the Amazon Web Services (AWS) cloud. AWS was chosen as the provider for web services because of the author's previous experience with AWS and because AWS provides a large amount of services at no cost. The Safe Shop system makes use of a number of different web services for content storage, authentication, data storage, and data manipulation. The diagram below depicts the different parts of the system, the cloud services that are utilized, and how the components and services interact with each other.

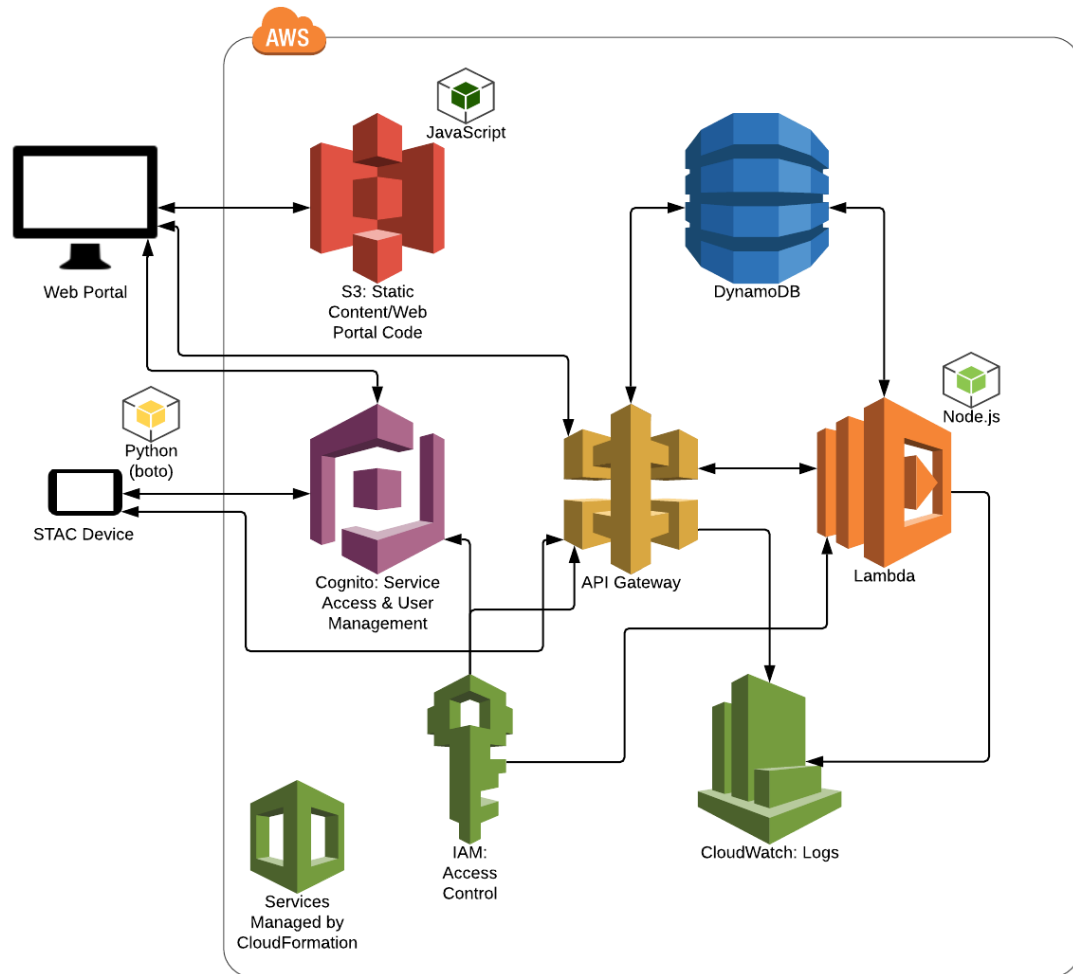


Fig. 4 A software diagram for the Safe Shop Workshop Management System depicting major components of the system software. The figure was created using Lucidchart [25].

A. STAC Device

The program that runs on the STAC device is a Python program. This language is minimalistic and easy to use. One motivation for choosing Python was the existence of several clear tutorials for using the keypad, OLED screen, and RFID card reader with Python libraries in a Python script. Additionally, there is a well supported Python library for interaction with AWS.

B. Web Portal

The web application is written in JavaScript and utilizes the React library for building the user interface [26]. There are a number of popular web application user interface frameworks, including React, Angular, and Vue [27]. Reviewing articles on the different frameworks revealed the pros and cons for each, but no clear reason for choosing one over the other. Ultimately, React was chosen. There were no drawbacks discovered that would prevent developing an excellent application with it. The author also had particular interest in learning it due to its use in the author's workplace. Given the popularity of React, there was no concern for a lack of resources to reference when developing the portal application.

In addition to using the React library, the Material-UI library is utilized. This library includes React components based around the Material Design principles [28]. Due to the author's experience using Material Design principles as a mobile app developer, the Material-UI library was a natural starting place for building out the web portal.

C. Amazon S3 (Simple Storage Service)

While S3 was the final choice to store the web application, the option of using a web server to host the web application was also considered. Utilizing a web server requires managing the run of the service itself. AWS can help automate the management of this, but the cost of this is the cost of running a server instance. Based on the cost of a reserved EC2 (Elastic Compute Cloud) micro server instance, the estimated cost of running a web server for a year would be a minimum of around \$50 [29]. While Amazon does offer many free services for the first year of use, after that first year, the organization

would need to pay at least \$50 to run the web application server. A cheaper alternative to this is to utilize an S3 bucket to store the web application files and then configure the bucket to serve the files as a website [30]. This is quick and easy to do. Additionally, the cost is minimal for a small application with a small user base. For example, according to the S3 pricing documentation, using 1 GB of storage would cost \$0.023/month [31]. Add to that 1 million GET requests to retrieve data from the bucket and you add \$0.40/month. A generous estimate of S3 usage cost is \$0.43/month. The web portal application currently utilizes less than 1 MB of storage in S3 and the number of GET requests used is expected to remain below the Free Tier limit of 20,000 requests. This means the first year of usage will be free and subsequent years will likely cost less than \$5 annually for S3. The pay as you go model used by S3 saves money, particularly for an application that does not have heavy or consistent usage, since you only pay for the storage and requests that are used.

D. Amazon API Gateway, Amazon Lambda, and Amazon DynamoDB

In addition to utilizing S3 rather than a web server for serving up the web application, a serverless architecture was also used for the web services that manage the logic and data storage for the application.

Data is stored in DynamoDB, a NoSQL/non-relational database [32]. While the data for the application could easily be stored in a relational database, the cost to run a relational database is similar to the cost of running a web server since both rely on running a server constantly. Instead, DynamoDB can be utilized using more of a pay as you use model. AWS provides a limited amount of free usage of DynamoDB for anyone

forever [33]. The database setup for the Safe Shop system is such that use will not exceed the free tier limits for read and write requests. Also, given that 25 GB of data storage is provided free, there is no expectation data use will exceed free tier limits. Therefore, use of DynamoDB is free for the system. Though most cost-effective, utilizing a NoSQL database does require modeling data differently than if using a relational database. However, this has been considered throughout the project development and the application is well equipped to manage data stored in a NoSQL database.

The logic and services utilized to send to and retrieve data from the database are provided with the API Gateway and Lambda [34], [35]. The API Gateway is the application programming interface to which the web and Raspberry Pi clients can make requests with and/or for data. These requests are then either forwarded directly to DynamoDB, if they are simple reads or writes to the database, or, for more complex requests, the requests are forward to a Lambda function which will execute to complete the required task. Similar to the other services, the API Gateway and Lambda are both pay as you use so that you only have to pay for the number of requests or function executions that are made. For the API gateway, if a million requests were made per month, then the cost will be \$3.50/month [36], though for the first year, the first million requests every month are free. For the Lambda service, Amazon always provides 1 million requests free to each user per month in addition to 400,000 GB-seconds of compute time [37]. It is not expected that use of the application will hit the free tier limit on lambda function executions, nor is it expected that more than 1 million API requests will be used per month. Therefore, the expected cost for Lambda is \$0 and the expected cost for the API Gateway is \$42 annually after the first year.

E. Amazon Cognito

Amazon Cognito manages users and access control for applications [38]. It is integrated into the Safe Shop system to manage web portal users and provides users with the appropriate credentials to access the other AWS resources in use by the Safe Shop system. Cognito is also used to grant STAC devices appropriate credentials to access AWS resources. Using Cognito is a simple way to ensure authorized users can access the APIs and the system data and ensure unauthorized users cannot. AWS provides free management for up to 50,000 active users per month, therefore there is no expected cost from using Cognito [39].

F. Amazon Identity and Access Management (IAM)

Amazon Identity and Access Management (IAM) is a free but critical service which provides various ways to manage access to AWS services and resources [40]. IAM users, roles, and policies are defined for various parts of the application and provide the ability to enforce the principle of least privilege. Enforcing least privilege means ensuring that AWS services can only access the other services they need and that users granted AWS access through Cognito can only access the resources they need.

G. Amazon CloudWatch

Amazon CloudWatch provides various tools to monitor services in AWS [41]. For the Safe Shop system, CloudWatch is used to store logs from the API Gateway and Lambda services. Whenever these services are used, logs of the events are sent to CloudWatch and stored there. If an error occurs, these logs can be referenced to help

understand the problem and solve it. AWS allows 5 GB of log data per month for free [42]. It is not expected that the Safe Shop system would exceed this limit, therefore, the CloudWatch services will be used cost-free.

H. Amazon CloudFormation

Amazon CloudFormation enables management of infrastructure as code through the use of templates [43]. Use of templates ensures that all infrastructure configurations are documented. The Safe Shop system utilizes CloudFormation templates for all of the aforementioned AWS resources which ensures that anyone interested in using the system can very easily replicate it. Use of CloudFormation is free for AWS resources, so this service costs nothing for the Safe Shop system [44].

I. Summary

The software design for the Safe Shop system was completed with expandability, repeatability, and cost in mind. Through the use of various AWS services the system provides the features desired at a low cost while allowing for expansion and for others to emulate the system. Below is a summary of the services used and their cost:

- Amazon S3 (Simple Storage Service) - Free first year / \$5 per year after
- Amazon API Gateway - Free first year / \$42 per year after
- Amazon Lambda - Free
- Amazon DynamoDB - Free
- Amazon Cognito - Free
- Amazon Identity and Access Management (IAM) - Free

- Amazon CloudWatch - Free
- Amazon CloudFormation - Free

VI. IMPLEMENTATION

The implementation of the Safe Shop system has been fully documented and made public online. Any school or community workshop or hobbyist interested in setting the system up themselves can utilize the source code and follow the detailed instructions available at the locations specified below.

A. Source Code

- Infrastructure [45]
- STAC Device [46]
- Web Portal Application [47]

B. Documentation

- The system documentation is contained in one main wiki and includes the pages listed below [48].
- System Overview: In addition to an overview of the system, similar to what is contained in this document, this page also provides links to video demonstrations of the system in action.
- Setup Guide: This guide provides materials and tools lists along with instructions for setting up the entire system from start to finish.
 - Initial Project Setup
 - Infrastructure Setup
 - STAC Device Setup
- User Guide: This guide provides instructions for utilizing the system features.

- Tool Access Control
- User Management
- Tool Management
- Training Management
- Device Management

C. GEARS, Inc. Workshop

One of the applications of the Safe Shop system is to middle and high school robotics teams like the GEARS, Inc. robotics team. To demonstrate the capabilities of the Safe Shop system, the system was implemented for the team in their workshop.



Fig. 5 The STAC device installed in the GEARS, Inc. robotics team's workshop.

In the image above, you'll notice a ShopSmith Power Station configured with a sander and hooked up to a vacuum to collect dust. Atop the power station on the right side, you can see the STAC device with a little green light showing it is on. On the floor to the right of the power station, you can see the relay into which the power station and vacuum are plugged so that when an authorized user scans their card and enters their PIN on the STAC device, the sander and vacuum are powered on together for use.

The successful implementation of the Safe Shop system for GEARS, Inc., which included STAC devices along with the web portal and required infrastructure, demonstrates the capabilities of the system.

VII. CONCLUSION

Individuals young and old can learn a great deal from getting their hands dirty in a workshop. The Safe Shop system provides a mechanism for controlling access to tools so that the workshop experience is safe. During initial system development, related work was referenced so that lessons learned could be applied to the Safe Shop system while also adding unique features to cater to the needs of a school or community workshop. The system provides tool access control, user management, tool management, training management, and device management. Utilizing industry practices including Infrastructure as Code (IaC) and Continuous Integration/Continuous Deployment (CI/CD) pipelines, the system was also carefully designed and implemented to ensure it is expandable, repeatable, and economical.

While much work has been done to this point to create a fully-functional system, much more work can be done to expand and refine it. Throughout the development process, new features and new approaches to existing features were identified and documented on the Safe Shop Product Backlog issue board [49]. In addition to ideas documented on the issue board, many more are up for consideration.

The focus of the project from the beginning has been to develop a system that provides value for the user, one of the key themes of Agile, a software development methodology that focuses on providing value to users and regularly re-evaluating the product created to modify it as necessary to best satisfy the user. The Agile Manifesto was written by a group of leaders in the software industry and outlines these values [50]. The Agile Principles further state that: “Continuous attention to technical excellence

and good design enhances agility” [51]. While providing features for the user is important, there should also be a focus on excellence to ensure the process of delivering value can be continued. In keeping with this theme of providing value while maintaining technical excellence, future work includes both new features and refactoring existing features.

The purpose of refactoring is to make the software more maintainable and more amenable to integrating additional capabilities. Future refactoring efforts will include expanding the scope of the CI/CD pipelines, better organizing the IaC, and restructuring the web application API calls to more efficiently manage data.

In addition to refining existing features in accordance with user need, future work to expand the features of the Safe Shop system may include adding the ability to log maintenance actions for tools. Along with maintenance logs, the STAC device may be configured to log use metrics which allow administrators to better understand the value of existing tools and plan maintenance activities. Additionally, the web portal may be integrated with the existing safety quiz system and opened up to all workshop users to allow them to view their training records, follow links to take safety quizzes, and see those quiz results populate their training records. Finally, the system may also be expanded to include time and task management features for a team in addition to tool and safety management for a workshop.

Supporting individuals in honing their technical skills is important. The Safe Shop system aims to play a role in that by enabling tool access control for school and community workshops.

REFERENCES

- [1] "AccessPack." Sole Digital. <https://www.soledigital.com.au/accesspack.html>.
- [2] Build Brighton. <https://buildbrighton.com>.
- [3] "Build Brighton - Makerspace." YouTube.
<https://www.youtube.com/playlist?list=PLFHlBmYTPHMTURKB-8yz1JYi6gNIR0S3G>.
- [4] A. Guy. "Build Brighton Member System." ArthurGuy.
<https://arthurguy.co.uk/portfolio/bbms>.
- [5] "BBMembershipSystem." GitHub.
<https://github.com/ArthurGuy/BBMembershipSystem>.
- [6] "RFIDAccessControl." GitHub.
<https://github.com/ArthurGuy/RFIDAccessControl>.
- [7] C. Horton. "Card Reading Power Switch." Casey Horton.
<https://www.hortoncasey.com/cardpowerswitch>.
- [8] OnyxEpoch (C. Horton). "Card Reading Shop Power Switch." Instructables.
<https://www.instructables.com/Card-Reading-Shop-Power-Switch/>.
- [9] fritzing. <https://fritzing.org/>.
- [10] "Getting Started with the Raspberry Pi Zero Wireless." SparkFun.
<https://learn.sparkfun.com/tutorials/getting-started-with-the-raspberry-pi-zero-wireless/all>
- [11] "Arduino vs. Raspberry Pi: A Comparison of the Microcontroller and Single-board Computer." Ionos.
<https://www.ionos.com/digitalguide/server/know-how/arduino-vs-raspberry-pi-affordable-diy-hardware/>.
- [12] "Raspberry Pi Zero." Raspberry Pi.
<https://www.raspberrypi.org/products/raspberry-pi-zero>.
- [13] "Arduino Nano 33 IOT." Arduino. <https://store.arduino.cc/usa/nano-33-iot>.
- [14] "IoT Relay." Digital Loggers, Inc. <https://dlidirect.com/products/iot-power-relay>.
- [15] "MFRC522: Standard Performance MIFARE and NTAG Frontend." NXP.
<https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>.
- [16] "About MIFARE." MIFARE. <https://www.mifare.net/en/about-mifare/>.
- [17] "Standard LCD 20x4 + extras - white on blue." Adafruit.
<https://www.adafruit.com/product/198>.
- [18] "Adafruit PiOLED - 128x32 Monochrome OLED Add-on for Raspberry Pi." Adafruit. <https://www.adafruit.com/product/3527>.
- [19] "Standard LCD 16x2 + extras - white on blue." Adafruit.
<https://www.adafruit.com/product/181>.
- [20] "Adafruit PiOLED - 128x32 Mini OLED for Raspberry Pi." Adafruit.
<https://learn.adafruit.com/adafruit-pioled-128x32-mini-oled-for-raspberry-pi/>.
- [21] "Membrane 3x4 Matric Keypad + extras - 3x4." Adafruit.
<https://www.adafruit.com/product/419>.

- [22] “3x4 Matric Keypad.” Adafruit. <https://www.adafruit.com/product/3845>.
- [23] “Matrix Keypad.” Adafruit.
<https://learn.adafruit.com/matrix-keypad/python-circuitpython>.
- [24] “2. Basic Recipes: 2.16. Full Color LED.” Gpiozero.
<https://gpiozero.readthedocs.io/en/stable/recipes.html#full-color-led>.
- [25] Lucidchart. <https://www.lucidchart.com/>.
- [26] React. <https://reactjs.org/>.
- [27] S. Martin. “The Top JavaScript Frameworks For Front-End Development in 2020.” freeCodeCamp.
<https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/>.
- [28] Material-UI. <https://material-ui.com/>.
- [29] “Amazon EC2 Reserved Instances Pricing.” AWS.
<https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/>.
- [30] “Amazon S3.” AWS. <https://aws.amazon.com/s3/>.
- [31] “Amazon S3 Pricing.” AWS. <https://aws.amazon.com/s3/pricing/>.
- [32] “Amazon DynamoDB.” AWS. <https://aws.amazon.com/dynamodb/>.
- [33] “Amazon DynamoDB Pricing for Provisioned Capacity.” AWS.
<https://aws.amazon.com/dynamodb/pricing/provisioned/>.
- [34] “Amazon API Gateway.” AWS. <https://aws.amazon.com/api-gateway/>.
- [35] “Amazon Lambda.” AWS. <https://aws.amazon.com/lambda/>.
- [36] “Amazon API Gateway Pricing.” AWS.
<https://aws.amazon.com/api-gateway/pricing/>.
- [37] “Amazon Lambda Pricing.” AWS. <https://aws.amazon.com/lambda/pricing/>.
- [38] “Amazon Cognito.” AWS. <https://aws.amazon.com/cognito/>.
- [39] “Amazon Cognito Pricing.” AWS. <https://aws.amazon.com/cognito/pricing/>.
- [40] “Amazon Identity and Access Management (IAM).” AWS.
<https://aws.amazon.com/iam/>.
- [41] “Amazon CloudWatch.” AWS. <https://aws.amazon.com/cloudwatch/>
- [42] “Amazon CloudWatch Pricing.” AWS.
<https://aws.amazon.com/cloudwatch/pricing/>
- [43] “Amazon CloudFormation.” AWS. <https://aws.amazon.com/cloudformation/>
- [44] “Amazon CloudFormation Pricing.” AWS.
<https://aws.amazon.com/cloudformation/pricing/>
- [45] “Safe Shop/Infrastructure.” <https://gitlab.com/safe-shop-wms/infrastructure>.
- [46] “Safe Shop/STAC Device.” <https://gitlab.com/safe-shop-wms/stac-device>.
- [47] “Safe Shop/Web Portal.” <https://gitlab.com/safe-shop-wms/web-portal>.
- [48] “Product Backlog Wiki Home.”
<https://gitlab.com/safe-shop-wms/product-backlog/-/wikis/home>.

- [49] “Safe Shop Product Backlog Issue Boards.” GitLab.
<https://gitlab.com/safe-shop-wms/product-backlog/-/boards>.
- [50] “Manifesto for Agile Software Development.” <https://agilemanifesto.org/>.
- [51] “Principles behind the Agile Manifesto.”
<https://agilemanifesto.org/principles.html>.