Team 10: IdentiKey

Final Report

5/10/17

Team 10

Andrew Davis, Brian DeVries, Jonathan Engle, Jared Haan

Team Advisor

Professor Mark Michmerhuizen

Senior Design — Engineering 340 — Spring 2017

Calvin College Engineering

Executive Summary

Team 10 is a group of four senior Electrical & Computer Engineering students: Andrew Davis, Brian DeVries, Jonathan Engle, and Jared Haan. Each group member attends Calvin College—a liberal arts college located in Grand Rapids, MI—and will graduate with a Bachelor of Science in Engineering (B.S.E.) with an electrical and computer concentration in May of 2017. This group was formed with the goal of completing a senior design project over the course of the academic year. This project is for Calvin College's engineering capstone course, ENGR-339 and ENGR-340. The senior design course spans two semester—ENGR-339, the first-semester course, emphasizes team management and engineering design while the second-semester course, ENGR-340, emphasizing the prototyping, testing, business analysis, and completion of the major design project.

This report discusses the design, research, business plan, testing, and system integration of an electronic system called IdentiKey. IdentiKey is a novel system that addresses a long-standing commercial problem: the organization and storage of a large number of keys. Large organizations must maintain massive sets of keys in order to control building and individual room access. The immense size of these key collections makes it difficult to efficiently organize and distribute the keys.

IdentiKey aims to mitigate the challenge of mass key management by automating the organization process. Users will be able to return and dispense keys in a simple, quick, and secure manner. Managers will be able to control access to different keys and reference records of key access easily. IdentiKey aims to solve the problem of mass key management by reducing the effort and time required by users while improving overall security.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1 Project Background

Many companies/institutions such as Calvin College have large collections of keys that are used for access to dozens or even hundreds of unique locks. These keys are a necessary burden because the school/organization cannot distribute master keys freely. Rather, a unique key is required for many doors and buildings. As a result, institutions end up with a quantity of keys too large to easily manage in an organized and efficient manner using traditional methods.

Three major aspects of traditional approaches to mass key management—key retrieval, key return, and security—are inept and cumbersome. An example of a traditional key management system, used by Calvin College's Physical Plant, is shown in Figure 1 and Figure 2. With this system, keys are stored on hooks in locked cabinets. Each key is assigned a hook, which is labeled on a separate sheet.

Figure 1: First Key Storage Example

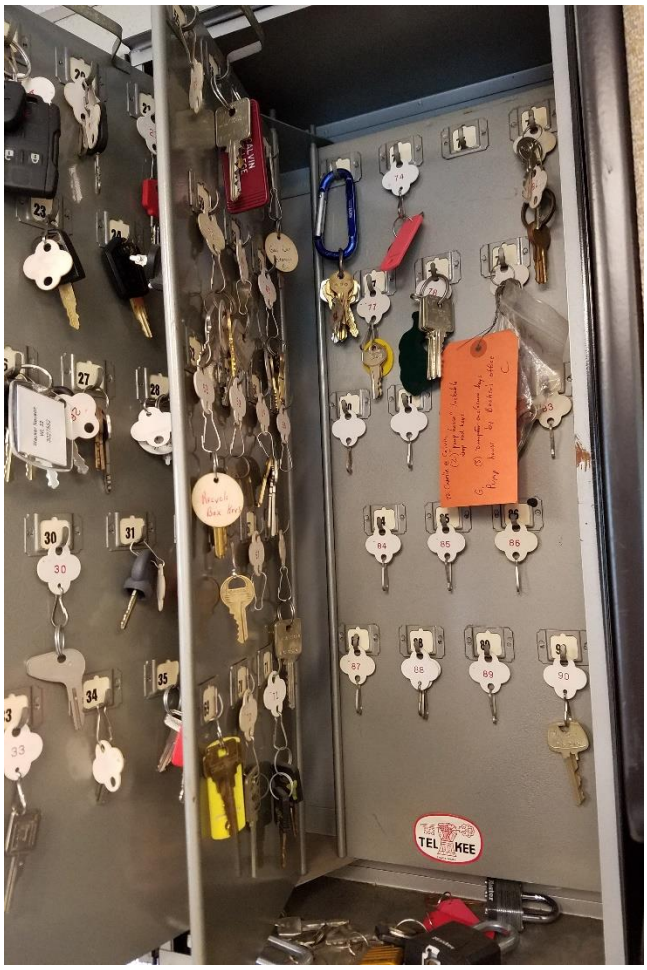

Figure 2: Second Key Storage Example

In the traditional key management system used by Calvin's Physical Plant, when a key needs to be used the user must read the key reference sheet to determine which hook the key should be on. Then the user must look through the cabinet, find the proper hook, and hope the key is there. If the last person to use the key placed it on the wrong hook, or on no hook, the user may not be able to find the key at all without manually searching through all keys in the cabinet. Alternatively, if the key is not on the assigned hook, it may be in use by another employee. In the case that it is currently being used, the searcher's only way to determine who has the key, and when it was taken out, is to reference a separate data sheet that is supposed to be filled out upon every key use. However, this system of manual record keeping is open to the same problems of human error and negligence.

When a user is finished with a key, he/she must manually return it to the proper hook. The user must use the hook reference sheet to determine which hook a key goes on, then manually find the hook and place

the key there. The returning system introduces high potential for human error. If a user is in a rush, he/she may throw the key in the cabinet, rather than properly placing the key on the correct hook, which is visible in Figure 2. Additionally, it is possible that the user returns the key to the wrong hook. This can be detrimental to the efficiency of the system since the next person to need the key will have to manually search through all keys in the cabinet to find the proper key. If the user takes the time to properly return the keys, this process can be time-consuming, especially if he/she is returning several keys at once.

Security of the keys is managed by placing the key hooks inside of a cabinet that can be locked. However, this broad approach to security fails to manage the security of individual keys. If a user needs access to one key in a cabinet, he/she will need access to that entire cabinet, and therefore, he/she will have access to all of the keys in that cabinet. Additionally, any key can be taken with no way to ensure that an accurate record of the use of the key is kept. Any person with access to a cabinet could take any key inside and decide not to record his/her withdrawal. This would make it impossible to look up the last person to use a key after a security breach situation.

Because of these insufficiencies, traditional systems of mass key management are insecure and prone to human error. Current implementations have room for substantial improvements in order to make a more effective system.

1.2 Team Description

Team 10 was comprised of four senior electrical and computer engineering majors. The knowledge and experiences of each team member were vital to the success of this project, as it involved both hardware integration and software development. Each team member brought different experiences to the table, ensuring that Team 10 had four bright and determined minds to get the job done well and on time. The members of Team 10 can be seen below in Figure 3.

Figure 3: Team 10

1.2.1 Andrew Davis

Drew, pictured second from the left, is a senior Electrical & Computer Engineering major with a Computer Science minor. He grew up in Hamilton, MI. Drew's interest in engineering stemmed from his interest in the technology and computers that he grew up with. This interest progressed into a lifestyle as he began his studies at Calvin. At Calvin, Drew is currently the President of the IEEE Student Chapter.

In the summers of 2014 and 2015, Drew interned at JR Automation in Holland, MI. He worked as a controls engineering intern. In this position, he assisted with the implementation of automation solutions both in Holland and at customer plants, including Corning and Tesla Motors. Last summer he worked as a research intern in the Electrical and Computer Engineering Department at Carnegie Mellon University under Dr. Shawn Blanton. While at Carnegie Mellon, alongside Jonathan, he worked to develop a system that utilized machine learning to reduce the size of post-silicon test sets on integrated circuits.

Outside of engineering, Drew enjoys rock climbing, tennis, browsing Reddit, taking weekend trips, and finding new life experiences. Fun fact: time is an unstoppable force that is slowly guiding us towards our inevitable deaths.

After graduation, Drew will desperately cling to his happy life as a student by pursuing a Ph.D. in the Computer Science & Engineering program at the University of Michigan. Before beginning the next round of education in the fall, Drew will spend the summer working one last undergrad internship at Tesla Motors in Fremont, CA.

1.2.2 Brian DeVries

Brian, pictured second from the right, is a senior Electrical & Computer Engineering major at Calvin College. He hails from a small family farm in Loveland, CO where his interest in engineering began. Brian grew up learning the trade of carpentry from his father, as well as the mechanics of tractors and the basics of farming from his grandfather. Through the encouragement of his family to learn to fix things when they were broken, his innate curiosity and knack for math, he decided to see what Calvin College engineering had in store for him.

Throughout the past 4 years, Brian has worked in a neighborhood helping lead a summer program, in a woodshop in Colorado, and as an athletic field maintenance assistant at Calvin College. Brian has taken the opportunity to learn on his own despite the lack of an internship, learning to use Arduino and applying academic knowledge to the real world.

Not only does Brian have a curiosity for systems and how things work, but he also has a deep appreciation for the outdoors. Some of his other interests include electronics, woodworking, and variety in physical activity. However, Brian and his wife have a hedgehog, puppy, and fish as pets that keep them very busy during the school year.

Brian has accepted a position with Gentex Corporation in Zeeland, MI where he will work as a Production Support Engineer.

1.2.3 Jonathan Engle

Jonathan, pictured far right, is a senior Electrical & Computer Engineering major and Mathematics minor from Troy, MI. Jonathan's love for electronics began at an early age when he became fascinated with anything and everything in his home that was powered by electricity. This passion continued to grow in high school both in the classroom and at his first job working as an assembler in a small electronics manufacturing facility. During his time at Calvin College, Jonathan has been involved in Resident Life leadership roles including Dorm Treasurer and Resident Assistant; Jonathan also currently serves as the Vice President of Calvin College's IEEE Student Chapter.

In the summer of 2015, Jonathan interned at Continental AG in their Body & Security division in Troy, MI where he worked on project testing/integration, circuit prototyping and debugging, and system

requirements. This past summer, the summer of 2016, Jonathan interned at Carnegie Mellon University's Center for Silicon System Implementation (CSSI), specifically the Advanced Chip Test Laboratory (ACTL), in Pittsburgh, PA in which he and Drew worked alongside a CMU professor and Ph.D. student to develop a system that minimizes JTAG test sets utilizing machine learning.

Outside of school, Jonathan is an avid music and art lover. He enjoys tinkering with audio electronics/programs, staying active, and playing all sorts of card/board/video games. Fun fact: Jonathan is considered the funniest person on Team 10, everyone on the team appreciates his jokes (especially his puns).

Jonathan will participate in Northrop Grumman Corporation's Professional Development Program after graduation at their Rolling Meadows, IL campus.

1.2.4 Jared Haan

Jared, pictured far left, is a senior Electrical & Computer Engineering major from Munster, IN. Jared first became interested in engineering in high school when he took his first electronics and CAD classes. His love for science, problem solving, and design pushed him to continue learning about electronics both in his studies at Calvin and on his own. While at a Calvin, Jared has served on dorm leadership as a Social Events Team member and currently serves as the Treasurer of the IEEE Student Chapter.

In the summer of 2016, Jared interned at GE Aviation in Grand Rapids, MI. During the internship, he worked on process alignment and simplification as a systems engineer. He was able to consolidate many of the project/product management processes and documents at GE. In previous summers, Jared worked as a crew leader for C&T Lawn and Landscape in his hometown.

Outside the academic realm, Jared enjoys many outdoor activities such as hiking, camping, and biking. He also has a passion for music and plays the guitar and trombone. Jared's love for the outdoors has also spilled over into a love for small animals, mainly hamsters.

Jared has accepted a job as an Associate Electrical Engineer at Colorado Engineering Inc. in Colorado Spring, CO.

1.3 Abbreviations/Acronyms

A:	amps
CPU:	central processing unit
DDR3:	double data rate three
DSI:	display serial interface
ENGR:	engineering
GB:	gigabyte
GHz:	gigahertz
GPIO:	general-purpose input/output
GUI:	graphical user interface
HDMI:	high-definition multimedia interface
ID:	identification
I/O:	input/output
in-oz:	inch-ounce
KB(/s):	kilobyte (per second)
kHz:	kilohertz
lbs:	pounds
LCD:	liquid crystal display
MB:	megabyte
MHz:	megahertz
mm:	millimeter
oz:	ounce
PWM:	pulse-width modulation
RAM:	random-access memory
RFID:	radio frequency identification
SD:	secure digital
SRAM:	static random-access memory
UID:	Universal Identifier
USB:	universal serial bus
V:	volts

2. Project Management

2.1 Team Organization

In order to complete the project in a timely and efficient manner, tasks and roles were divided amongst the members of Team 10. This allowed multiple parts of the project to be completed at the same time, ensuring a streamlined approach to the project. The task categories were divided up primarily based on the different hardware components (or "blocks") of the final system.

Drew was responsible for the research and implementation of the automation component of IdentiKey. This is the step between when a key is deposited and when the key is in storage. He was also in charge of the implementation of the selected CPU and storage mechanisms. The CPU served as the "central hub" to which all other hardware components of our system were connected. Drew also took the lead on the main system integration code to ensure that all of the hardware components were communication with one another properly.

Brian was also responsible for the implementation of the CPU alongside Drew. In addition, he took the lead on key storage, which needed to store keys as efficiently as possible. Brian also served as the primary contact between Team 10 and Calvin College's physical plant—the primary client and beneficiary of this project. Brian also took the lead on the physical aspects of the project, meaning that he was in charge of assembling the exterior casing and interior pulley system.

Jonathan served as the webmaster of Team 10; he has created and maintained Team 10's website to reflect project progress and project milestones accurately as well as post important documents for public view. Jonathan was tasked with researching the user input and interface systems as well as the key identification system(s). Jonathan also served as the head scheduler, making sure that all team members were up-to-date on project deadlines and assignments pertaining to the senior design course. Jonathan also created and finalized the team's presentations and report documents that they had to give throughout the spring semester.

Jared served as Team 10's treasurer; he was in charge of approving and reviewing every purchase that Team 10 will make to ensure that the team stays within the allotted budget given to them in the ENGR-339/340 course. The components/systems that Jared researched were the automation and storage solutions alongside Drew. He also assisted Jonathan in implementing the user input and interface systems. In the second half of the semester, Jared also ensured that our motor system was working properly.

All the members of Team 10 provided input to schedule, budget, and document the project. Team meetings were held at least once a week, or as needed to meet the deadlines of the project. Most team meetings were used to discuss design alternatives after each member has done his respective research and system integration execution plans. In addition, meetings were used to review documentation, revise the schedule, and approve expenses. All project documents and other documentation were kept on Team 10's shared Google Drive—this ensured easy access and collaboration for all of the team members.

Team 10's organization structure can be seen in Figure 4, which includes Team 10's advisor, Professor Mark Michmerhuzien, Team 10's industrial consultant, Mr. Eric Walstra, and other course instructors.

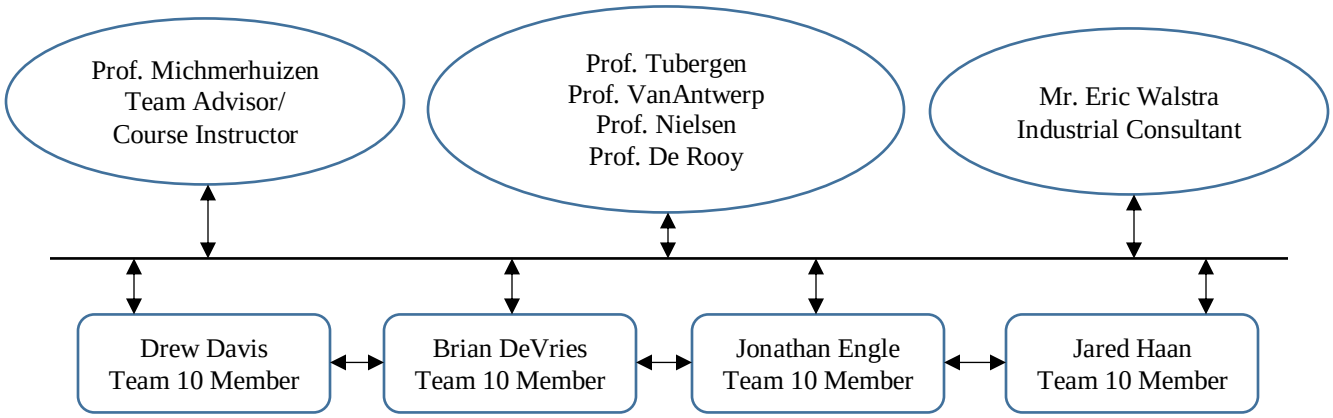

Figure 4: Team 10 Organizational Diagram

2.2 PPFS

The Project Proposal Feasibility Study (PPFS) was completed by the assigned due date for the ENGR-339 Senior Design course. This document was broken down by section and each section was assigned to a team member for a rough draft to be completed. Once all of the sections were completed, the group met to work through any questions the team members had on individual sections. This draft was then given to our team advisor, Professor Mark Michmerhuizen, who provided insights and comments for moving forward with creating the final version of this document.

The purpose of the PPFS was to consider the risks and benefits of the IdentiKey project. These risks and benefits were weighted to determine whether this project was feasible and in the scope of the Senior Design course.

2.3 Final Report

The Final Report was to be completed by the assigned due date for the ENGR-340 Senior Design course. This document, like the PPFS, was broken down by section and each section was assigned to team members for a rough draft to be completed. Once all of the sections were completed, the group met to work through any questions the team members had on individual sections. This draft was then given to our team advisor, Professor Mark Michmerhuizen, who provided insights and comments for moving forward with creating the final version of this document.

The Final Report goes into more detail about how the prototype was assembled, which components were selected and why, how the components were integrated into the casing, etc.

2.4 Website

Another requirement for this course was to create and maintain a website for our team. This website contained information about our project, the team's status, and archives of presentations and documents. Team 10's website can be accessed via <*http://engr.calvinblogs.org/16-17/srdesign10/*> and was actively maintained by Jonathan. Jonathan was responsible for uploading important documents and presentations to the website as they were completed.

2.5 Oral Presentations

In addition, four oral presentations were required for ENGR-339 and ENGR-340. Two of these four oral presentations were given in the fall semester for ENGR-339 and two were given in the spring semester for ENGR-340. Because each member of Team 10 was required to talk at least once during these four presentations, it was decided that each team member would give at least one full oral presentation. For ENGR-339, Drew gave the first oral presentation and Brian gave the second presentation. For ENGR-340, Jared and Jonathan teamed up and gave the third and fourth oral presentation. Two more miscellaneous presentations were given during the spring semester—one presentation in front of CEAC (Calvin Engineering Advisory Board) members and one presentation at the senior design night banquet. For these two presentations, every team member participated.

The goals of these presentations were to update the class on the project status, present challenges and obstacles that the team was currently facing, and answer any questions from classmates or professors regarding our project.

2.6 Schedule

In order to facilitate a smooth flow of work throughout the semester, a detailed schedule was created using Microsoft Project and then exported to a second planning software, Asana. The schedule was based on senior design class deadlines and expectations. All tasks were broken down into subtasks of eight hours or less and assigned a start date, end date, and estimated hours required. The full schedule for the project can be seen in the Appendix; a list of the major tasks for the fall semester is shown below in Table 1:

Table 1: IdentiKey Major Tasks for Fall 2016

Task Name	Estimated Work	Start	Finish
Brainstorming	8 Hours	Fri 9/9/16	Fri 9/16/16
Requirements	3 Hours	Mon 9/19/16	Fri 9/23/16
Team Collaboration	48 Hours	Fri 9/9/16	Fri 9/16/16
Research	45 Hours	Mon 9/19/16	Mon 11/14/16
Documentation	63 Hours	Mon 10/10/16	Mon 12/12/16

The spring semester focused on implementation of our project, and thus, the major tasks differed from the major tasks done in the fall. A list of the major tasks for the spring semester is shown below in Table 2:

Table 2: IdentiKey Major Tasks for Spring 2017

Task Name	Estimated Work	Start	Finish
Part Selection	8 Hours	Mon 1/30/17	Mon 2/6/17
Part Testing	30 Hours	Mon 2/6/17	Mon 3/6/17
Team Collaboration	40 Hours	Mon 1/30/17	Mon 4/3/17
Prototyping/Testing	50 Hours	Mon 3/6/17	Mon 4/10/17
Documentation	60 Hours	Mon 4/3/17	Mon 5/1/17

2.7 Budget

Team 10's assigned budget was $500. It was the team's goal and desire to stay within this budget, if possible. Jared was in charge of maintaining the budget and updating it as needed. This budget was used as a management tool to guide some of Team 10's decisions concerning design alternatives. Desired components that would have compromised the budget might have been substituted for a cheaper or lower-quality option. If no suitable substitutes existed and the budget needed to be compromised, Team 10 met to reassess the scope of the project and the possibility of increasing the budget. All project expenses and budget updates were approved by the entire team and were subsequently carried out by Jared as the team's treasurer. A complete, detailed chart of how our expenses were used can be found in Appendix 5.

3. System Design

3.1 Design Norms

While working on IdentiKey, Team 10 made it a priority to develop a solution that affirms and upholds our values as future Christian engineers and bearers of God's image. The difference in the way a Christian Engineer approaches a problem from the way a simply moral individual approaches the problem is not necessarily in the outlined design norms found below, but in the attitude of the approach and the reason for the motivation to solve the problem. Christian engineers have been called to do God's work for a long time, dating back to the work and construction of the tabernacle by God's people in the Bible. Work needs to be done out of gratitude for God and to serve him through the stewardship of his creation.

3.1.1 Delightful Harmony

Delightful harmony encompasses the user experience and the aesthetic look of the finished product. IdentiKey is a system that is pleasing to the eye as well as one that is simple and intuitive to use.

3.1.2 Trust

Trust involves the marketing of the final product and transparency throughout the production process. Because IdentiKey is a system that holds items that should stay in the right hands, it needs to be one that is safe and secure. A user will feel comfortable with relying on IdentiKey to do what it is intended to do.

3.1.3 Caring

Throughout the project planning and implementation process, Team 10 must exhibit care. Fundamentally, caring means to not only think of others but also to act on kind thought and put words into practice. IdentiKey saves time and eliminates frustration. One of the major reasons for creating our system was to alleviate stress on an individual and make a usually mundane task easier and less time-consuming.

3.2 System Requirements

The IdentiKey project is an unsolicited project. Therefore, a client did not lay out the system requirements; rather, the requirements were generated based on the underlying problems of mass key management. The requirements initially created for IdentiKey can be seen in the list below:

- The key storage and retrieval system shall accept keys from the user and sort them according to what the keys access.
- The accuracy of detection and sorting shall be at least 95%.

- The time to complete one instance of accepting a key from a user and sorting it shall be within 15 seconds.
- Upon the user request for a key, the system shall dispense the appropriate key and no other keys to the user.
- The time to complete one key retrieval after a user request shall be within five seconds of the user's request.
- The system should be powered by a 120-volt wall outlet for easy positioning within buildings.
- The system shall retain its data despite losing power.
- The data shall be tracked and easily accessible through a user interface.
- Along with tracked key data, the system shall be physically secured from theft to prevent keys from being stolen. System access shall be granted by a secure and user-unique user identification protocol.

These requirements were put in place to ensure the goals of the solution were met and to set a standard while the system was being developed. Requirements were important to Team 10 and the team strived to meet these requirements to the best of their ability with the design norms in mind.

3.3 System Architecture

3.3.1 Hardware System Architecture

Below in Figure 5 is the basic hardware architecture in which all hardware components "talked" to one another. The Raspberry Pi 3 acted as the main connector of all the hardware components. The way in which the hardware components interfaced with one another was dependent on whether the user was depositing or retrieving a key, which was determined via the user interface.

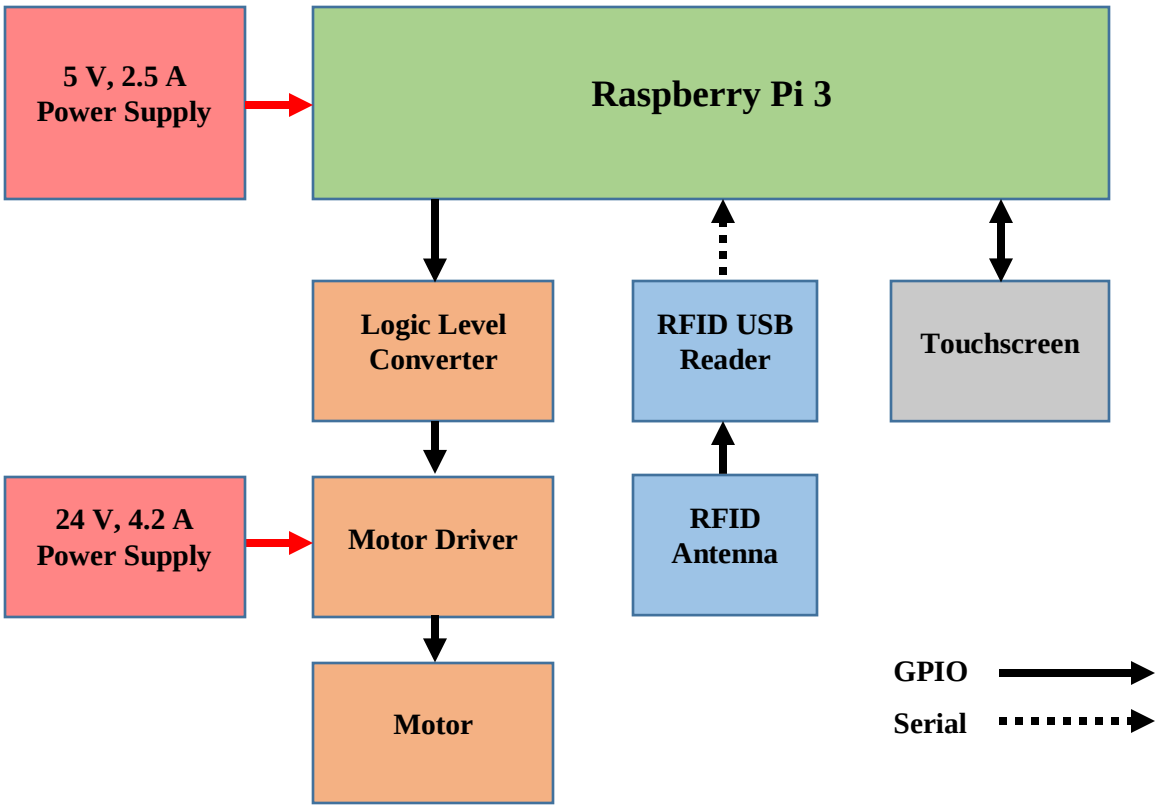

Figure 5: Hardware System Architecture

3.3.2 Software System Architecture

Below in Figure 6 was the basic software system architecture for our system. The majority of programs were written in Python due to the ease of integration with our Raspberry Pi and the experience with the language that the team members had.

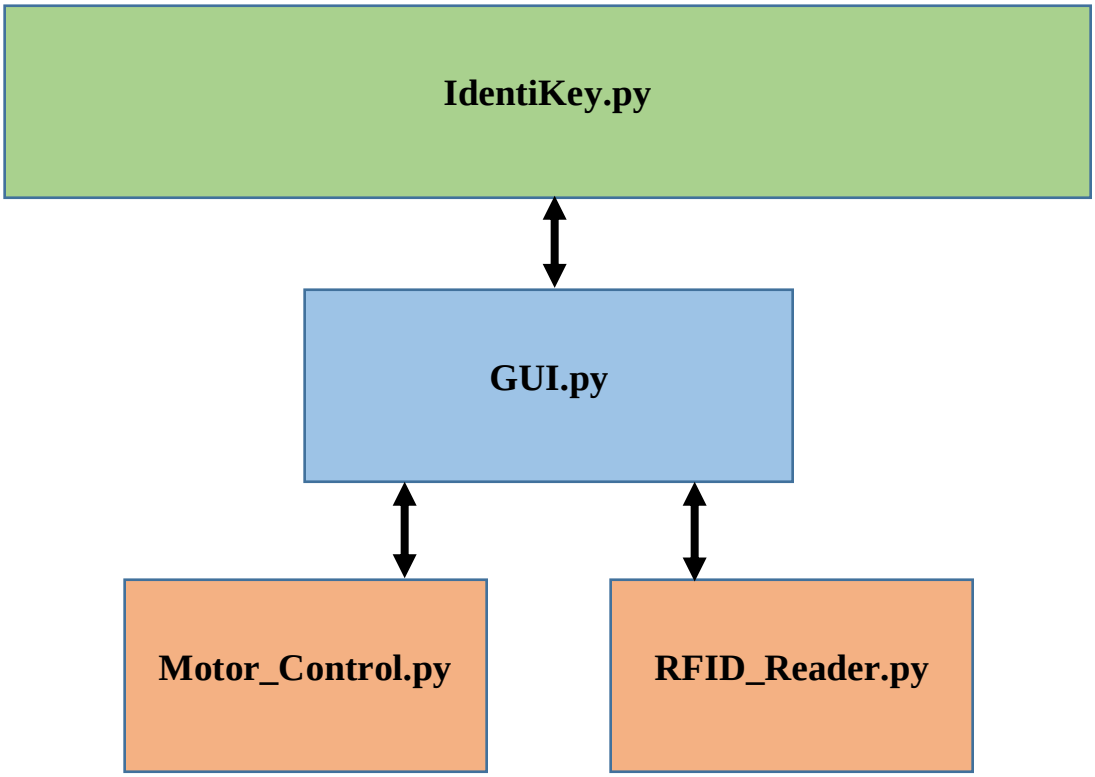

Figure 6: Software System Architecture

3.4 CPU

The first component of the system, which was a driving factor behind the decisions for additional components, is the CPU. This component served as the brains of the entire system. The options for the CPU were the familiar Raspberry Pi 3 and the Arduino Uno, as well as the less familiar BeagleBone Black. The three alternatives were analyzed based on their ability to satisfy three essential criteria: low cost, efficiency, and ability to render a user-friendly design.

3.4.1 CPU Selection

Team 10 decided to implement a Raspberry Pi 3 Model B as the CPU of IdentiKey. A detailed analysis and comparison of the alternatives can be found in the following section. With a 1.2 GHz processor and 1 GB of RAM, the Raspberry Pi was powerful enough to handle the processing necessary to run IdentiKey. The Raspberry Pi featured 4 USB ports, 40 GPIO pins, an HDMI port, and a Micro SD card slot, all of which were essential for the development of IdentiKey and controlling its various subsystems. Our Raspberry Pi can be seen below in Figure 7.

Figure 7: Raspberry Pi 3 B

3.4.2 CPU Alternatives

First, the Raspberry Pi 3 offered several attractive features. It had four USB ports available, allowing interfacing with other USB components. There were also 40 GPIO pins that could serve as connections between additional Raspberry Pi CPUs, Arduino boards, or other serial components such as an RFID tag reader or a display screen. The Raspberry Pi offered one GB of RAM on board with a 1.2 GHz processor, as well as a port to attach an SD card to hold more information and data to expand its memory storage capabilities. An onboard HDMI and graphics connection offered versatility when displaying information.

The next CPU option, the Arduino Uno, had data storage of approximately 32 KB flash memory and 2 KB SRAM. It also had a clock speed of only 16 MHz, and 26 total I/O pins. The Arduino was able to quickly process incoming signals from sensors which made it useful for instances of smaller data requirements on the chip, but more rapid response times necessary with sensors involved.

Finally, the BeagleBone Black, which essentially served as a combination of the Raspberry Pi and Arduino platforms, was considered. It offered up to 92 I/O pins, a single USB port, a clock speed of 1 GHz, 4 GB of flash memory, and 512 MB DDR3 RAM. It also offered broad software capabilities.

In summary, the three CPU alternatives that were considered are the Raspberry Pi 3, the Arduino Uno, and the BeagleBone Black. The Raspberry Pi had many superior characteristics. It had the highest clock speed at 1.2 GHz and maintained its own operating system on which to run programs. The graphics abilities of the Raspberry Pi 3 were also superior to those of the BeagleBone, and due to the lack of a processor on the Arduino, it had no need for a graphics card. However, the Arduino performed better

when used with sensors and was very responsive to changes in voltage and current, quickly changing outputs based on incoming signals.

Memory capabilities were also an important specification to consider, as the system required an adequate amount of data to be stored in order to keep track of all the keys and user accesses. The BeagleBone had the most memory available, followed by the Pi, and then the Arduino.

Another distinction was the amount of I/O each board offers. The BeagleBone contained the most, followed by the Raspberry Pi, and then the Arduino. The decision matrix for the CPU can be seen below in Table 3.

Table 3: CPU Decision Matrix

Criterion	Weight	Arduino	BeagleBone Black	Raspberry Pi 3 B
Processing Power	8	3	8	7
I/O Connections	8	6	8	7
Availability	8	8	3	8
Documentation	7	8	4	8
Familiarity	2	8	1	2
Cost	4	3	7	7
Ease of Implementation	4	3	4	7
	Total:	232	226	**252**

Ultimately, a Raspberry Pi was chosen as the CPU. The Raspberry Pi offered enough processing power and memory to store all the data necessary—it was also easy to use. The GPIO pins allowed interfacing with other boards to communicate any sensor data needed to be processed concerning the key storage. The cost of the Raspberry Pi 3 and an Arduino Uno together was also comparable to the cost of a single BeagleBone Black, giving no advantage to using a single board to fulfill the needs of the project.

3.5 Identification System

Arguably, the most important component of our system was the one responsible for properly identifying keys so that they can be properly sorted and stored. For this component, two solutions were considered: machine vision and RFID tagging.

3.5.1 Identification System Selection

Implementing RFID tags was chosen to identify keys. A detailed analysis and comparison of the identification alternatives can be found in the following section. The biggest reason why a RFID system was chosen was that it was decided that our system needs to be able to handle key rings as well as individual keys. In addition, there was simply not a machine vision system on the market that was robust enough to scan and remember a complicated physical item such as a key while being affordable enough for the team's allotted budget.

Two electronic components, made and distributed by SparkFun, comprised the identification system—a RFID USB-based reader, and a RFID antenna.

The SparkFun RFID USB Reader (P/N: SEN-09963) was a simple to use, USB to serial base unit for various RFID readers/antennas. This USB reader served as the "base" for the antenna which then connected to the Raspberry Pi via a mini-USB cable. A picture of this component can be seen below in Figure 8.

Figure 8: RFID USB Reader

The SparkFun RFID ID-20LA (P/N: SEN-11828) was a very simple to use RFID reader module from ID Innovations. With a built-in antenna, the only holdup is the 2-mm pin spacing. This antenna was capable of a 125 kHz read frequency and operated at a baud rate of 9,600 bps. A picture of this component can be seen below in Figure 9:

Figure 9: RFID Antenna

The RFID tags used were small, about the size of a quarter, and operated at a frequency of 125 kHz. When a tag came within approximately 1.75 inches of the RFID reader, the module beeped (the beeping feature was later terminated as it was an annoyance) and the tag's unique 32-bit identification data was shown on a terminal screen. A Python program was created to be able to output these pieces of data to the Raspberry Pi, and subsequently, other Python programs.

3.5.2 Identification System Alternatives

The biggest advantage of implementing machine vision was that the keys themselves need no extra electronics attached in order for our system to identify them. The goal of implementing machine vision was to scan a key that a user entered, identify its unique physical properties, and then associate those properties with specific key information (based on previously entered user data). There were numerous camera modules on the market that could attach directly to the Raspberry Pi 3; one of the camera modules considered was the Camera Module v2, which had a Sony IMX219 8-megapixel sensor that attached to the Raspberry Pi via a 15-cm ribbon cable.

Another technology that was considered was RFID tags. Each key would have its own unique RFID tag that a RFID scanner could quickly recognize. The system would then know which key is associated with that particular tag (based on previously entered user data). One of the major advantages of RFID readers was that orientation of the RFID tags does not matter. Thus, it allowed for more variability in the movement of the keys in IdentiKey. Additionally, the use of a RFID reader would have allowed the system to identify an entire ring of keys at one time, rather than being limited to identifying one key at a time. The primary disadvantage of a RFID system was its limited range, so the RFID system needed to be placed carefully in order for it to function properly.

The decision matrix for the identification system can be seen below in Table 4. Implementing RFID tags was chosen as the way to properly and quickly identify keys. The biggest reason why a RFID system was chosen was that it was decided that our system needed to be able to handle key rings as well as individual

keys. In addition, there was simply not a machine vision system on the market that is robust enough to scan and remember a complicated physical item such as a key while being affordable enough for the team's allotted budget.

Table 4: Identification System Decision Matrix

Criterion	Weight	Machine Vision	Barcode	RFID
Accuracy	9	3	9	9
Speed	4	4	9	9
Orientation Versatility	6	1	5	8
Cost	5	1	6	6
Ease of Implementation	4	1	5	8
	Total:	58	197	**227**

3.6 User Interface

Another important part of IdentiKey was the user interface—the way in which a user interacted with our system. Options that were considered were a touchscreen interface and a keyboard input with an LCD display.

3.6.1 User Interface Selection

The team purchased a touch panel that was compatible with our chosen CPU—the proprietary Raspberry Pi 7-inch Touchscreen Display. The display had a resolution of 800 by 480 pixels and connected to an adapter board that handled power and signal conversion to and from the Raspberry Pi. This touchscreen ran the GUI and handled user login and key selection and retrieval. The touchscreen can be seen below in Figure 10. A detailed analysis and comparison of all interfacing alternatives can be found in the following section.

Figure 10: Raspberry Pi 7" Touchscreen

3.6.2 User Interface Alternatives

The first option considered, a touchscreen, was the first natural thought that the team brainstormed. The team believed that a touchscreen would feel natural for user interaction. This explained a touchscreen's biggest advantage—simplicity of use and it felt organic. This fit well with our design norms. The biggest disadvantage of a touchscreen was that it would take more work and coding to fully operate and integrate within our system. A GUI (graphical user interface) would have to be created that took into account that a user would be using his/her finger rather than a mouse.

A keyboard and/or a pin pad was also considered. A user would use the keyboard and/or pin pad to enter his/her login credentials (for user verification) and for selection via the arrow keys. A computer mouse in addition to the keyboard/pin pad could be used as well for movement of a cursor to select options on an LCD screen. The decision matrix for the user interface can be seen below in Table 5:

Table 5: User Interface Decision Matrix

Criterion	Weight	LCD Screen	Touchscreen
User Friendly	9	5	9
Cost	5	5	5
Ease of Implementation	4	5	4
	Total:	90	**122**

It was determined that a touchscreen was the best option for user input and identification. This way, one screen can handle both the user entry and user identification via username and password. In addition, a touchscreen streamlined the entire system and exhibited characteristics that added to the goal of delightful harmony. Ultimately, the official Raspberry Pi touchscreen was chosen due to its ease of use and cost. To get the touchscreen working correctly, numerous updates had to be performed. Kivy, the open-source Python package for multi-touch displays, also had to be installed.

3.7 Automation & Storage

The part of our design that set IdentiKey apart from other similar systems was the automation. This refers to the component of the system that moved the keys from where they enter the machine to their location in storage. The three options considered were a robotic arm, linear actuators, and a conveyor system.

3.7.1 Automation & Storage Selection

Team 10 decided to construct a conveyor system to automate the key movements and storage. A detailed analysis of the various automation systems considered can be found in the following section. The conveyor system was designed to accept rings of keys from the user and automatically move the rings into storage. In order to accomplish this, a belt with attached hooks was strung around two pulleys, one of which was driven by a motor. The user would set rings of keys on a hook, which dangle from the hooks while being stored out of the user's reach. IdentiKey's timing belt can be seen below in Figure 11.

Figure 11: Timing Belt

3.7.2 Automation & Storage Alternatives

The first option considered to transport keys through the system was electric linear actuators. Actuators had several advantages over other automation options, especially considering the other components that were implemented in IdentiKey. The first advantage was simplicity. Linear actuators were relatively simple to use and implement—they had a minimal number of moving parts and were driven by a simple

input voltage. Actuators were also modular; hooking one up to a motor controller circuit allowed the user to control the speed, direction, and distance traveled by the actuator. In addition, the motor controller could be connected to an Arduino or another microcontroller to be controlled by a PWM signal from one of the I/O ports. Therefore, a linear actuator would have been relatively simple to implement because the system was already incorporating a microcontroller.

The biggest disadvantage of linear actuators was because they were linear. They only allowed one dimension of movement that severely limited the possible range of key movement. Actuators could be connected to one another to create more degrees of motion, but more actuators implied a bulkier and more complex automation system.

In addition to electric actuators, a multi-axis robotic arm was considered. Robotic arms had the distinct advantage of being able to move in all three dimensions, which provided maximum maneuverability of the keys. Robotic arms, at the scale of the IdentiKey project, were constructed with onboard microcontrollers, which resulted in a contained and simple system. The robots considered for IdentiKey were electrically powered with minimal voltage; therefore, robots did not require any additional connections or systems within IdentiKey to operate.

There were drawbacks of robotic arms as well. By their nature, robotic arms were expensive. A capable robot required the bulk of the team budget. With the confined budget on the senior design project, the options of robotic arms were limited. Within the affordable options, all robots suffered from a limited reach: 200 mm to 425 mm, typically. The size of the entire IdentiKey system would have needed to be tailored to the size of the robot.

Finally, the last option considered for automation was a conveyor system. This system had multiple advantages over the other two systems mentioned. A conveyor system was simple with a minimal number of moving parts. The only moving parts needed were a motor drive and a conveyor belt. In addition, a conveyor system eliminated the need to have separate storage and automation systems. The storage elements would already be attached to the conveyor belt and would rotate within the machine. This allowed individual storage elements to be moved to the point where a user would enter a key into the machine. Thus, the user could place the key directly into a specific storage element without relying on an automated system to move the key into storage. This greatly reduced both the cost and complexity of the automation system.

The final advantage of using a conveyor system was that it integrated well with the key identification system. Making the user place the key into storage with a specific orientation ensured that the identification system would not have any problems identifying keys because of the way they were stored.

In addition, the fact that the storage elements themselves could rotate within the machine means that the identification system did not have to move at all. The keys could simply be rotated directly to the identification system.

The biggest drawback of using a conveyor system was that it introduced a new problem that was not present in other automation options. Using a conveyor system with attached storage elements meant that the system had to be controlled user accessibility. The user must have only had access to a single storage element at a time, while the other storage elements were secured and inaccessible. A device had to have been developed that would restrict the user to accessing only the single, specific storage element necessary to store a specific key.

The decision matrix used to select an automation system is shown in Table 6. Based on this decision matrix, a conveyor-based system was chosen. The conveyor system was simpler and much more cost-effective than any other automation option. It also offered a greater storage capacity than the other options and was much easier to implement. The speed of the system was slightly less than other possible options, but that fact was easily outweighed by the potential benefits of a conveyor system.

Table 6: Automation Decision Matrix

Criterion	Weight	Robot	Actuator	Hanging Conveyor
Speed	4	7	6	5
Ease of Use	3	5	5	5
Key Storage Size	7	6	5	8
Cost	8	1	3	6
Ease of Implementation	5	7	5	4
	Total:	128	123	**159**

3.8 Motor System

The motor was one of the most important aspects of the automation and storage system. This component took care of moving the conveyor system inside IdentiKey so that the user could access different keys and key hooks. The speed, direction, and distance traveled by the motor shaft all needed to be precisely

controlled so that the CPU would be able to monitor the exact location of each key hook. For this reason, a stepper motor was chosen as the only type of motor suitable for IdentiKey. Stepper motors types and specifications can vary which fraction of a step is moved on each clock tick, as well as the direction and speed of the shaft. This allowed the CPU controlling the motor to keep track of the exact distance traveled by the motor, and thus the exact distance traveled by each key hook.

3.8.1 Motor System Selection

Since a stepper motor was the only type of motor considered for the automation system, the only alternatives that were considered were different sizes of stepper motors. To determine the motor size needed, basic torque calculations were utilized. The appendix shows a table that calculates the amount of weight that the motor would need to move in a worst case scenario. The heaviest possible pulleys along with the largest possible number of keys were used in this table. The total weight to be moved by the motor (generous safety factor included) was calculated from the table as follows:

*Total Weight = Total Weight of Keys * Weight of Pulleys = 56.4 oz * (14.5 oz * 2) =* ***85.4 oz***

From this, the torque needed to move the conveyor system was calculated as follows:

*Torque = Weight of Keys * Radius of the Pulley = 85.4 oz * 1.25 in =* ***106.75 oz-in***

The forces of friction were ignored in this equation since a timing belt attached to the pulley minimized slippage between the belt and pulley. The total number of keys per ring was assumed to be five, and the total number of rings that the system prototype would hold was assumed to be 20. Although these numbers changed, the calculations were done with those values in mind.

Table 7: Estimated Torque Calculation Parameters

Criteria	Units	Actual	Safety Factor
Average Key Weight	grams	8	16
Single Key Ring Weight	grams	40	80
Total Weight of Keys	grams	800	1,600
Total Weight of Keys	pounds	1.8	3.5
Total Weight of Keys	ounces	28.2	56.4
Weight of Single Pulley	ounces	14.5	14.5

Based on the torque calculations, the 17060 Nema 17 stepper motor was selected as the ideal motor for our conveyor system. The motor was capable of 115 oz-in of holding torque, which was more than enough to drive the pulley system. This was the only motor with the reasonable Nema 17 frame size that provided enough torque to drive the pulley system.

Equally important to the motor was the motor controller board/driver. The driver took pulse and direction signals from the CPU and converted them into signals that the motor could utilize. It also took power from an attached voltage supply and provided it to the motor as necessary. The characteristics of the driver were all determined based on the motor decision.

Based on the specifications of the 17060 Nema 17 stepper motor, the driver board needed to be compatible with a two-phase, bipolar, two-amp stepper motor. Since the stepper motor was capable of pulling up to 2 A of current, the driver needed to have an even higher current rating to ensure that it would not fail when handling the maximum possible current. In addition, the driver needed to have a very large heat sink to ensure that it would not overheat when exposed to long periods of use. Finally, the driver needed to accept 3.3 V logic signals based on the logic level of the chosen Raspberry Pi CPU input.

The Toshiba TB656-based CNC Router Single Axis Driver board was selected as the ideal driver for the 17060 Nema 17 stepper motor. This driver could handle up to three amps of current and had a large heat sink, ensuring that it would not fail under maximum power conditions. It was compatible with a two-phase bipolar stepper motor and was extremely cost effective compared to most comparable drivers. The only drawback was that it required five-volt logic level signals. However, this problem was resolved by using a SparkFun Bi-directional Logic Level Converter to convert the 3.3-volt signals from the Raspberry Pi to five-volt signals for the driver. The stepper motor used can be seen below in Figure 12.

Figure 12: Nema STP-MTR-17060 Bipolar Stepper Motor

3.8.2 Motor System Alternatives

As discussed earlier, it was decided that the automation component of IdentiKey would be implemented with a conveyor. The conveyor system would be composed of pulleys and a belt, which would be driven by a motor. Additionally, a motor driver was required to translate signals from the Raspberry Pi into motor commands.

The conveyor system imagined by the team would have held rings of keys on hooks and rotated to control the position of the rings of keys. The team considered two types of conveyor belts. The first was a standard smooth belt and the second was a timing belt. The timing belt provided the distinct advantage of being toothed. Because timing belts have teeth, they could be matched to specific pulleys that also have teeth. These teeth prevented the belt from any amount of slippage as it goes around the pulley. This anti-slip feature ensured that the belt moved exactly as much as the pulleys did. This pairing enabled the team to accurately control the position of the conveyor belt and the rings of keys. The standard smooth belt did not offer the assurance of the timing belts and offered no advantages beyond simplicity.

The positioning and number of pulleys also needed to be planned by the team. The various ideas of pulley setups revolved around the dimensions the systems encompassed. The pulleys, and by association the belt, could be set up to span one, two, or all three spatial dimensions within the case of IdentiKey. A three-dimensional setup would have enabled the belt to move both horizontally and vertically. This system would optimize the use of space within IdentiKey and maximize its total capacity. However, the challenges of creating a conveyor system that can move keys in all dimensions would have significantly increased the complexity. A two-dimensional setup would have enabled the keys to move horizontally and would wrap back-and-forth to fill one layer of horizontal space. This setup would use the horizontal space within IdentiKey's structure optimally and provide a significant capacity. Again, however, designing a conveyor system to wrap around in two dimensions would have increased the overall complexity of the system. A one-dimensional setup would use two pulleys and simply move keys deeper into the structure or closer to the exit. This setup would have been inefficient at using space within the structure and would minimize the number of keys the system can hold. However, it would have been substantially simpler to implement.

The team discussed all options available for the implementation of the conveyor system. In the end, the team felt the choices were obvious. It was decided that the belt would be implemented with a timing belt, as there were not clear disadvantages to using them. As for the pulley setup, the team opted for a one-dimensional system. Ultimately, IdentiKey systems should use a three-dimensional system to optimize space and maximum key capacity; however, implementing a massively complex conveyor system was

outside of the scope of Team 10's project. The one-dimensional system provided adequate storage for the prototype and enabled the team to focus their efforts on the other critical system components.

3.9 Structure

Once the various mechanical and electrical subsystems had been decided, the structure in which to house the entire system could be designed. The structure plan was less essential, as the decision was not critical enough to alter the design of the system as a whole significantly. The structure needed to secure the various subsystems in their proper place and create an environment in which different components could easily connect and interact. A tabletop model and a floor-standing model were both considered. The two materials considered for constructing the structure were wood or aluminum.

3.9.1 Structure Selection

Once the subsystem designs were validated, the structure design was chosen. The team decided to build a tabletop model made out of wood. The final design of the conveyor system proved that the tabletop model would provide enough internal space for the prototype and it would be unnecessary to increase the spatial burden of the system. The team also chose to construct the structure with wood in order to take advantage of the team's strengths and minimize the burden of construction due to having a team composed only of electrical engineering students.

The material used for the frame was building material 2 inches by 4 inches. These boards were readily available and cost effective for their purpose. The frame needed to be sturdy enough to withstand the tension on the belt for the pulley system without bowing or bending. Metal would have been ideal for security purposes, but wood suited the needs of the project. Screws were used to fasten everything together.

The base was constructed of ¾-inch particleboard, providing a sturdy base for the system to rest on and a strong foundation for the frame to attach to as well. While the material of the base was not critical, having a firm, square board was necessary to maintain a sturdy and robust system.

The covering and paneling on the outer structure was done using ¼-inch plywood and a combination of plexiglass and acrylic to allow the user to see the inside of the system for presentation. An actual system would need to be enclosed in steel for security purposes with an access door, but would not allow the user to see inside to ensure that no one could infiltrate the system without adequate permission. A picture of the initial structure from can be seen below in Figure 13.

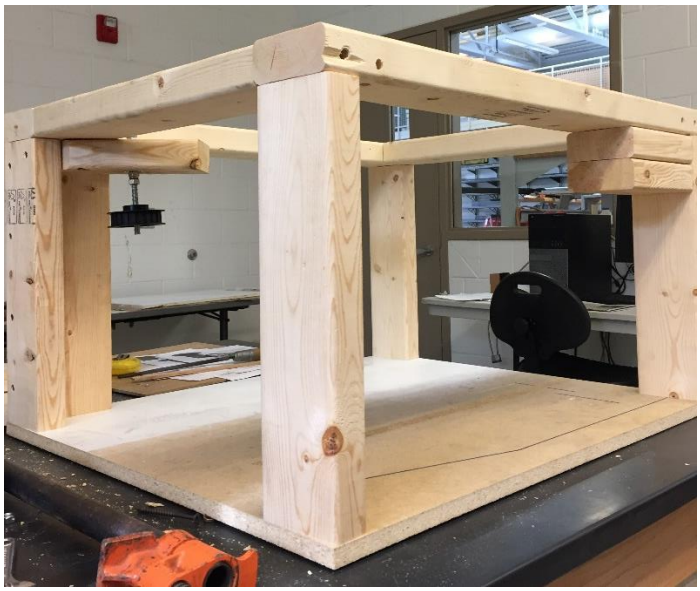

Figure 13: Frame Structure

3.9.2 Structure Alternatives

The first option considered for the structure was a tabletop model. This structure setup would stand less than four feet tall and would be optimized to fit on a standard table or counter. The primary advantage of this structure type was that IdentiKey would be minimally invasive to any preexisting office space that it was installed into. The corresponding disadvantage, however, was that the internal space available for the system would be limited. In addition to the height limit, the depth and width of the system would have needed to be kept small enough so it could comfortably fit on a table. This size limit could have hindered the total quantity of key rings that IdentiKey could manage at once.

Alternatively, the team considered building the structure to stand on the ground. A ground-standing model would have provided significantly more freedom to extend the dimensions in all directions. This would have substantially increased the internal space available in the system and potentially increased the key capacity. However, a system that stood on the ground would have required a much larger space and potentially deter organizations from implementing the system. The increased size of the system would have also increased the construction costs.

In addition to the shape of the structure, Team 10 had to decide how to build the structure. Building the system out of aluminum would have improved the overall structure strength and enabled a sleeker internal structure, which would have better utilized the system space. However, building the structure from aluminum would have introduced a larger construction cost and increased the amount of time necessary to complete the construction. A wooden structure would have used space less efficiently, but would have

minimized construction costs and time. Additionally, members of the team had preexisting experience working with wood.

4. Integration and Testing

Figure 14: Completed IdentiKey Prototype—Front View

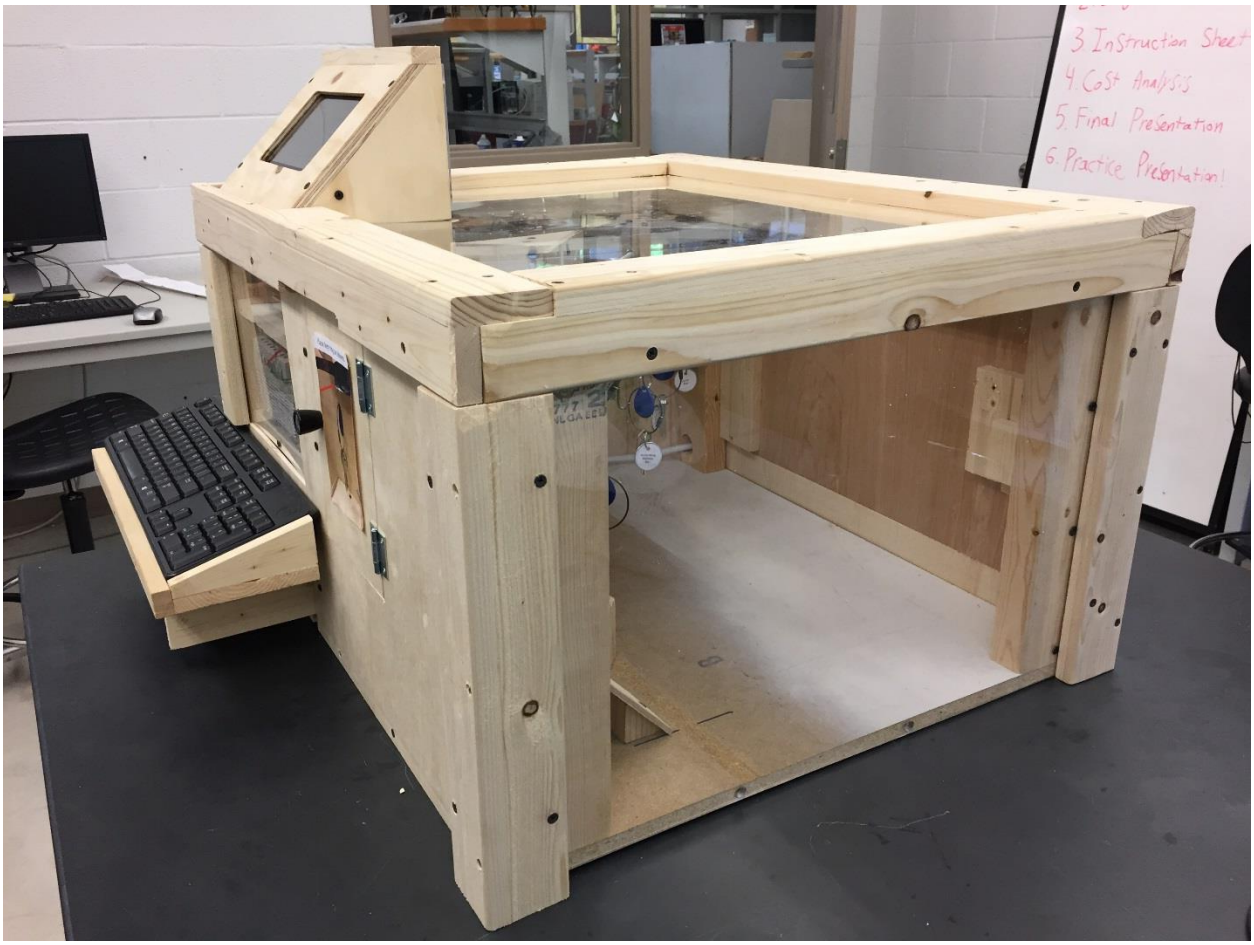

Figure 15: Completed IdentiKey Prototype—Side View

4.1 CPU

Before any progress could be made directly on individual subcomponents of the system, the Raspberry Pi had to be set up. A simple setup was achieved by utilizing NOOBS, a pre-built operating system installer. NOOBS was loaded onto an SD card and inserted into the Raspberry Pi. Using a mouse, keyboard, and monitor, a simple initialization process was then followed to install the Raspbian operating system. This system enabled access of the Raspberry Pi through a simple user interface. Python programs could also be run using the built-in terminal.

Two major hardware components were integrated with the Raspberry Pi for the final laboratory project for a different course, ENGR-325. The touchscreen and RFID—the two hardware components used in the final laboratory project—were connected to the Raspberry Pi. This first RFID reader used seven GPIO pins, while the touchscreen used four GPIO pins in conjunction with a ribbon cable. A more robust RFID

reader later replaced the original RFID reader. This second reader connected to the Raspberry Pi via USB instead of GPIO pins, which served to clean up the GPIO pins and eliminate pin sharing or wiring issues. Both the RFID reader and touchscreen could be controlled by the Raspberry Pi, which verified that that Raspberry Pi outputs were working properly. In the spring, additional hardware components were successfully operated by the Raspberry Pi, further verifying its functionality.

In the final integrated system, the Raspberry Pi is connected to several components.

- Touchscreen, via a ribbon cable and GPIO pins
- RFID Reader, via USB
- Motor Driver, via GPIO pins and the pulse-width modulation GPIO pin

The Raspberry Pi acted as the brains of the system by receiving inputs from each of these components and sending information and commands out as necessary.

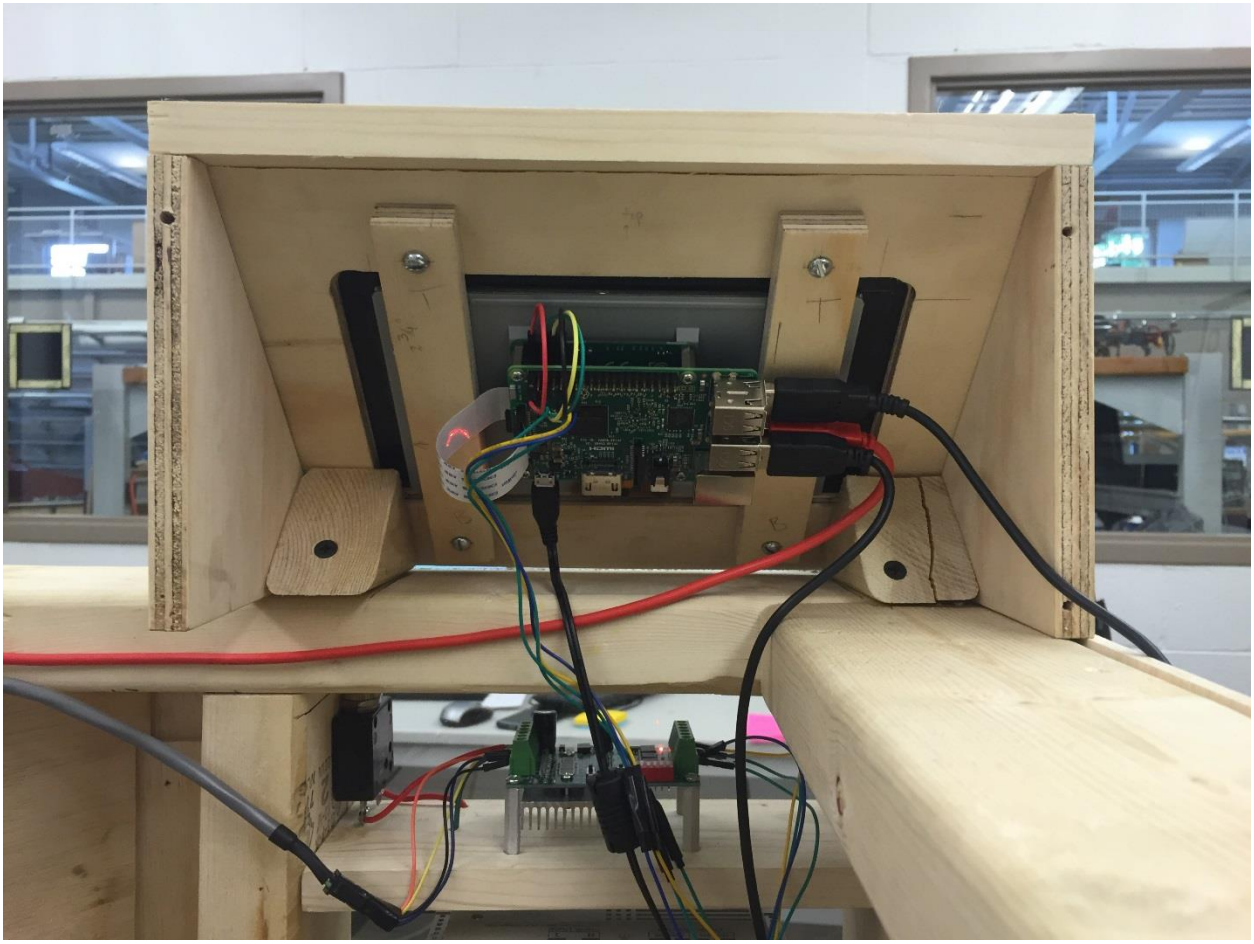

Figure 16: Final CPU Placement in IdentiKey Prototype

4.2 Identification System

For the final iteration of IdentiKey, Team 10 decided to purchase a SparkFun RFID reader and 20 basic RFID tag fobs. The RFID reader connected to the Raspberry Pi via USB and communicated serially. Both the reader and the tags operated at 125 kHz and the reader had roughly 1.5 inches of range with the purchased tags. The reader was capable of quickly reading tags and sending the associated tag ID back to the Raspberry Pi.

As mentioned previously, the key identification system chosen was a serial RFID system operating at 125 kHz. Once the individual component was chosen, it needed to be integrated into the system and tested. By writing the software before the integration, the testing could actually take place before its installation. The methods of testing and its installment into the collective system are outlined below.

Testing the RFID reader was relatively straightforward. Due to the simple USB connection and serial communication, the software written to operate the module was equally simple. A program was written to serially read the information from an RFID tag that was detected by the module and then store the information in a readable way so the system could use the information. Once the program was written, testing its ability to differentiate between multiple tags was executed. Each RFID tag ordered has a different ID number associated with it. Therefore, the system should be able to determine which tag it is and print the correct value on the screen for confirmation. Since the identification number is visible on the tag itself, it was very simple to cross reference the information and check for program accuracy. The data recorded by the program had extra characters and numbers that needed to be trimmed off. The code to perform the data trimming is shown below in Figure 17.

```
with Timeout(1):
    Key_Data_Str = str(aSerial.readline())

### trim extreneous bits off of the serial information
if len(Key_Data_Str) == 15:
    trimmed_data = Key_Data_Str[5:11]
elif len(Key_Data_Str) == 16:
    trimmed_data = Key_Data_Str[6:12]
elif len(Key_Data_Str) == 23:
    trimmed_data = Key_Data_Str[10:16]
elif len(Key_Data_Str) == 27:
    trimmed_data = Key_Data_Str[14:20]
else:
    print("Incorrect # of bits read from RFID!")
    trimmed_data = "0"
Key_Data = int(int(trimmed_data, 16))
```

Figure 17: Code to Trim RFID Serial Data

By passing multiple different tags over the reader and cross-referencing the read out in the terminal to the printed number on the tag, the program was tested.

The RFID reader was easily integrated into the system as well. The door where users could reach in to retrieve or return a key was used as a reference point for the placement of the RFID reader. It was set up part way between the entry and the nearest hook in the counterclockwise direction. This allowed the system to check the RFID tag of the ring at the entry using simple motor movements. Therefore, it was simple for the system to know if the correct key ring was returned or if the user had not yet placed the key on the hook. The reader was positioned about one and a half inches from the outside edge of the conveyor. This ensured that when a ring was moved in front of the reader it could be identified.

Figure 18: Final RFID Placement in IdentiKey Prototype

4.3 User Interface

The user interface was the component of IdentiKey that the user was directly able to interact with. The user interface was coded in Kivy—a Python-based language meant for touchscreen applications. Before the touchscreen was ready to be integrated into the final design, updates had to be performed on the Raspberry Pi to install Kivy and update Kivy to the latest version release. After this was done, a basic touchscreen application was run on the touchscreen to ensure touchscreen functionality. In this case, the Kivy installer library included basic programs for the installer to test. Various widgets were tested on the touchscreen, ranging from buttons, to sliders, to multi-touch zooming. After ensuring that the touchscreen worked properly with all of the test programs, a basic username and password login application was created by Jared and Jonathan for a final laboratory exercise in ENGR-325. This program was created in part to ensure that Kivy would be appropriate to use for IdentiKey's final GUI design. After the test login program was completed, the response time of the screen was tested by watching the response time of the

button presses to the execution of the Python function mapped to that particular button. In this case, the login function was mapped to the "Login" button. The login function checked two text inputs, username and password, to see if they were in the username and password database. If so, the GUI should have transitioned to another screen and if not, the GUI should have flashed a label saying that the credentials entered were not valid/not correct.

The final placement of the touchscreen was on the top-front-left corner of IdentiKey. This was deemed to be the most natural placement as the keyboard was positioned to the left of the key door. The Raspberry Pi was mounted on the back of the touchscreen, which is where all of the other electronic components were connected. The final GUI had three major screens: a login screen, an admin screen, and a user key selection screen. A picture of the login screen (the screen that a user sees when he/she first walks up to the system) is shown below in Figure 12. In addition, a diagram of the final GUI design flow can be seen below in Figure 13.

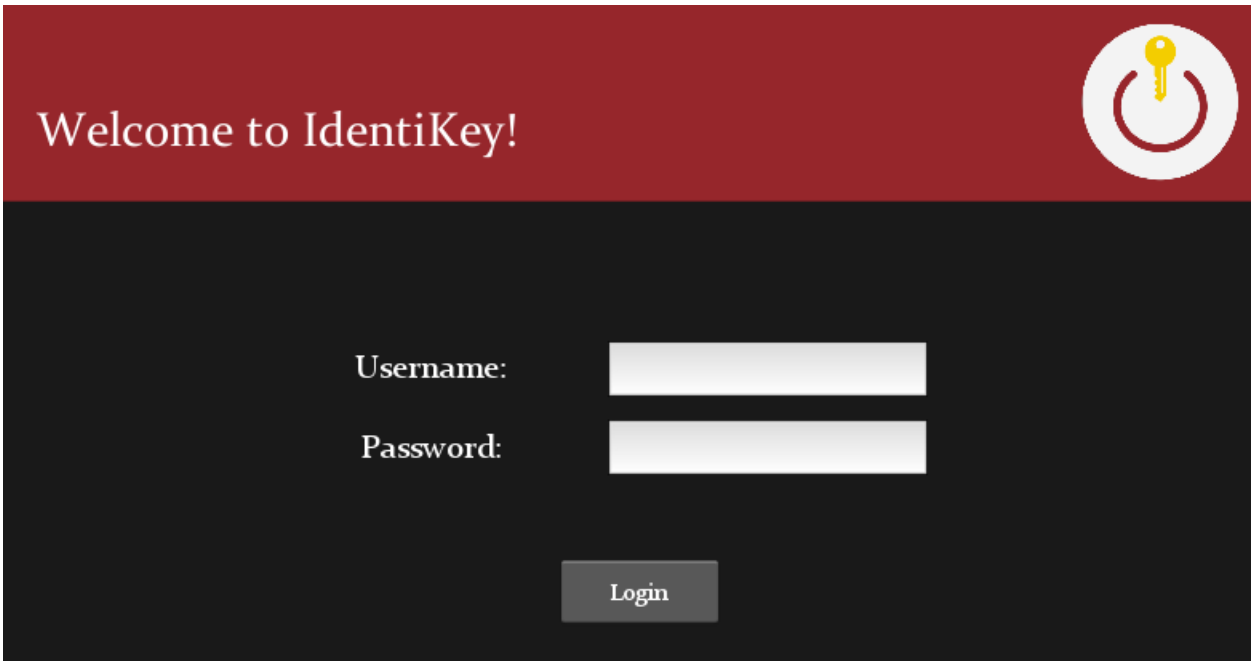

Figure 19: IdentiKey Login Screen

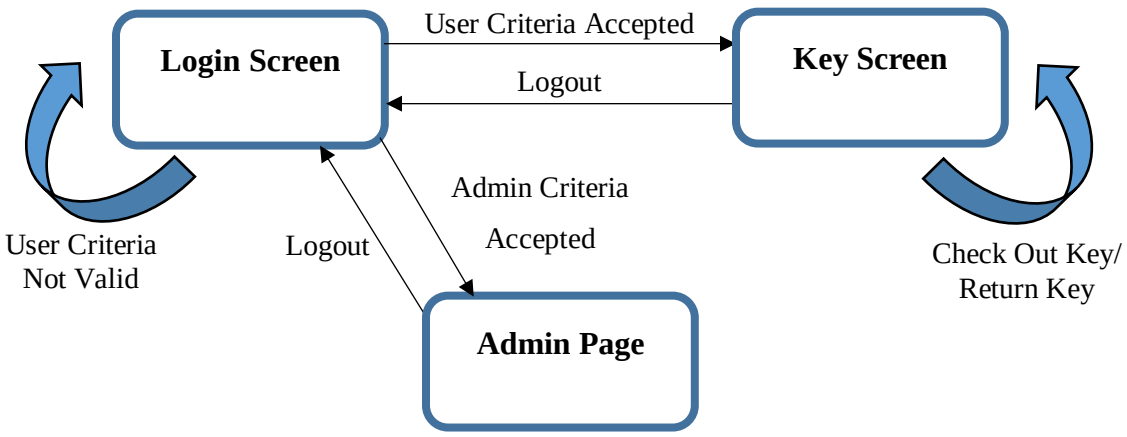

Figure 20: IdentiKey GUI Screen Flow

As mentioned previously, the visuals of the GUI were written in Kivy. An example of the code used for the login page can be seen below in Figure 21:

```
<LoginScreen@CustomScreen>:
    FloatLayout:
        # Welcome text
        Label:
            text: 'Welcome to IdentiKey!'
            color: (1, 1, 1, 1)
            font_name: 'Constantia.ttf'
            font_size: 30
            pos: -self.width / 3 + 40, self.height / 2 - 80

        # Username label
        Label:
            text: 'Username:'
            font_name: 'Constantia.ttf'
            font_size: 20
            pos: -130, 18
```

Figure 21: Kivy Code for GUI

With the code snippet above, a basic screen was created with welcome text in the upper-left-hand corner, and a label that says “Username:” towards the left-center of the screen. The rest of the GUI code followed this basic Python-like, tab-based syntax. The visuals were programmed in Kivy code as well as the function mapping of all the buttons.

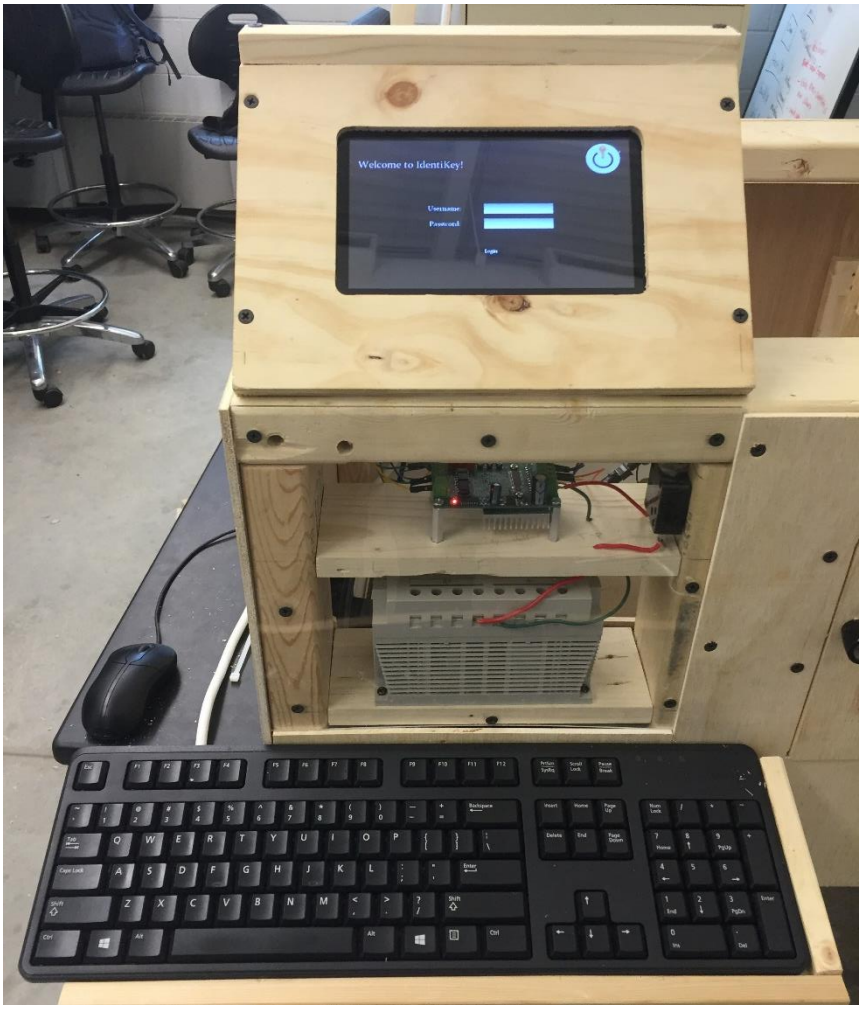

Figure 22: Final Touchscreen Placement in IdentiKey Prototype

4.4 Automation & Storage

The automation and storage system of IdentiKey were combined into a single conveyor system. The system was composed of two pulleys, a timing belt, and a stepper motor. These four components worked in conjunction to create a machine that was capable of moving and holding up to 20 rings of keys.

The assembly of the physical conveyor belt was relatively simple. The team ordered two pulleys and a matching timing belt, as discussed previously. The first pulley was connected directly to the shaft of the motor.

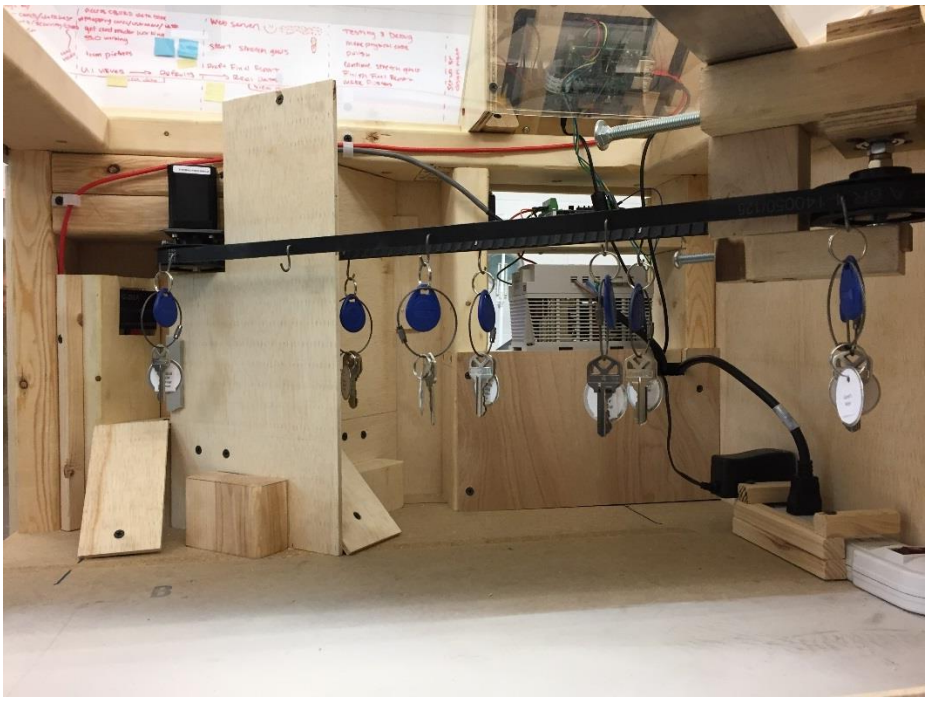

Figure 23: IdentiKey Prototype Pulley System

The second pulley was mounted to a free spinning shaft on the frame. The free spinning shaft was created using a bolt, two bearings, and several nuts. The frame was built so that the free spinning axis could be adjusted laterally, which enabled the team to put the belt on and move the second pulley as necessary to provide the proper amount of tension in the belt.

4.5 Motor System

Controlling the conveyor via a motor turned out to be one of the most challenging aspects of IdentiKey's design. The conveyor system required a motor that was capable of precise movements that would enable the team to track the belt rotation. It also needed to be strong enough to turn a belt loaded with full key rings. To create the system, two basic components were required: a stepper motor and a stepper motor driver. The motor and driver had to meet the requirements of the system as well as be compatible with each other.

Early in the design process, the team decided to use stepper motors, as discussed previously. However, within the realm of stepper motors, there is a massive range of options. Navigating the plethora of choices was not straightforward. In order to be acquainted with stepper motors, the team started with an old used motor in the engineering building. The motor was found in the supply room, likely used by a design team in previous years. The motor was a small two-phase stepper motor; however, due to lack of attached documentation, that was the extent of information the team had on the motor. A small motor driver, capable of driving a two-phase motor up to 700 mA, was ordered to test the motor.

A simple Python program was written on the Raspberry Pi for motor control. Through the driver, the program was able to set the rotation direction and step size. The program also was able to control exactly how many steps the motor would take and how quickly it would take those steps. The team was able to successfully connect and control the stepper motor through this program, which verified that the motor was working correctly. Unfortunately, though, the stepper proved too weak to drive the conveyor system the team had designed.

Therefore, the team decided to order a new, more powerful, motor. Approximate torque calculations were made so that the motor ordered would meet all of the necessary requirements. A 2 A two-phase stepper motor was selected and ordered. Along with that motor, a new and larger driver was necessary. After the team had acquired the new motor, its functionality was tested by hooking it up to the previously tested 700 mA motor driver. Although the low current rating of the driver prevented the motor from operating at maximum torque, the stepper did operate as expected. From this testing, it was proven that the new stepper motor was functional and capable of the simple movements necessary to control the conveyor.

With every motor driver implemented by the team, a motor driver (or several) had to be set up. The conveyor needed to be controlled through Python programs on the Raspberry Pi. Therefore, the motor driver's responsibility was to translate signals from the CPU to commands for stepper motors. The signals from the Raspberry Pi were programmed to be sent through the standard GPIO pins, which use 3.3 V signals. In order to control the original two-phase stepper that the team acquired from the Engineering Building supply room, a 700 mA driver designed for a two-phase motor was ordered. The driver was wired up to the Raspberry Pi such that the GPIO pins could directly control the direction, step size, and steps. This setup enabled the team to manually trigger and count each individual step of the motor, which allowed for precise position tracking of the conveyor. The driver successfully relayed the signals from the GPIO pins and controlled the motor. However, as explained previously, the motor and motor driver were not powerful enough to provide the torque necessary to handle the conveyor system design by the team.

This lack of power led to the purchasing of a new stepper motor and several new motor drivers. As stated previously, the second stepper was substantially larger than the one found in the Engineering Building supply room and was rated for two amps In order to provide the power necessary to utilize the motor's full potential, a second motor driver that was rated for 2.2 A was ordered. The new driver worked similarly to the original driver and should have allowed the motor to be accurately controlled by a Python program on the Raspberry Pi. However, the team failed to get the new driver operational. The motor responded to signals from the GPIO pins of the Raspberry Pi, but not in the intended fashion. The team was only able to get the motor to spin sporadically, slowly, and with minimal torque. When adjusting the built in potentiometer of the driver, the driver circuit would often disrupt the motor's motion or cut it off entirely. These behaviors made it became apparent that the driver was cheap and poorly built. With this realization, the team assumed the driver was faulty.

After the second driver was disregarded, yet another driver needed to be ordered. Rather than try the team's luck with the same driver again, a slightly larger and more robust driver was ordered. Additionally, due to the motor component of the project falling significantly behind schedule, a duplicate of the third circuit was ordered to insure against more shipping delays. The third driver was rated for three amps. Again, the driver had a similar design to the original driver and allowed the motor to be controlled by a program on the Raspberry Pi. Once again, the motor responded to the signals from the Raspberry Pi, but not in the correct way. Both of the new drivers were exhibiting unexpected behavior.

At this point, the team realized the issue was likely outside of the physical driver circuit. As the motor had already been proven to work, the team deduced (with the help of Professor Kim) that the issue was originating from the Raspberry Pi. Upon further research, it was discovered that the standard GPIO pins on the Raspberry Pi are not capable of outputting the steady pulsing signal that the drivers required. Instead, the Raspberry Pi has a special pulse-width modulation (PWM) pin that can be used to provide the steady and reliable signal necessary. The PWM pin was set up inside of the motor control programs and connected to the step signal of the third motor driver. After the pins were properly set up and connected, the motor driver was able to control the motor effectively and accurately.

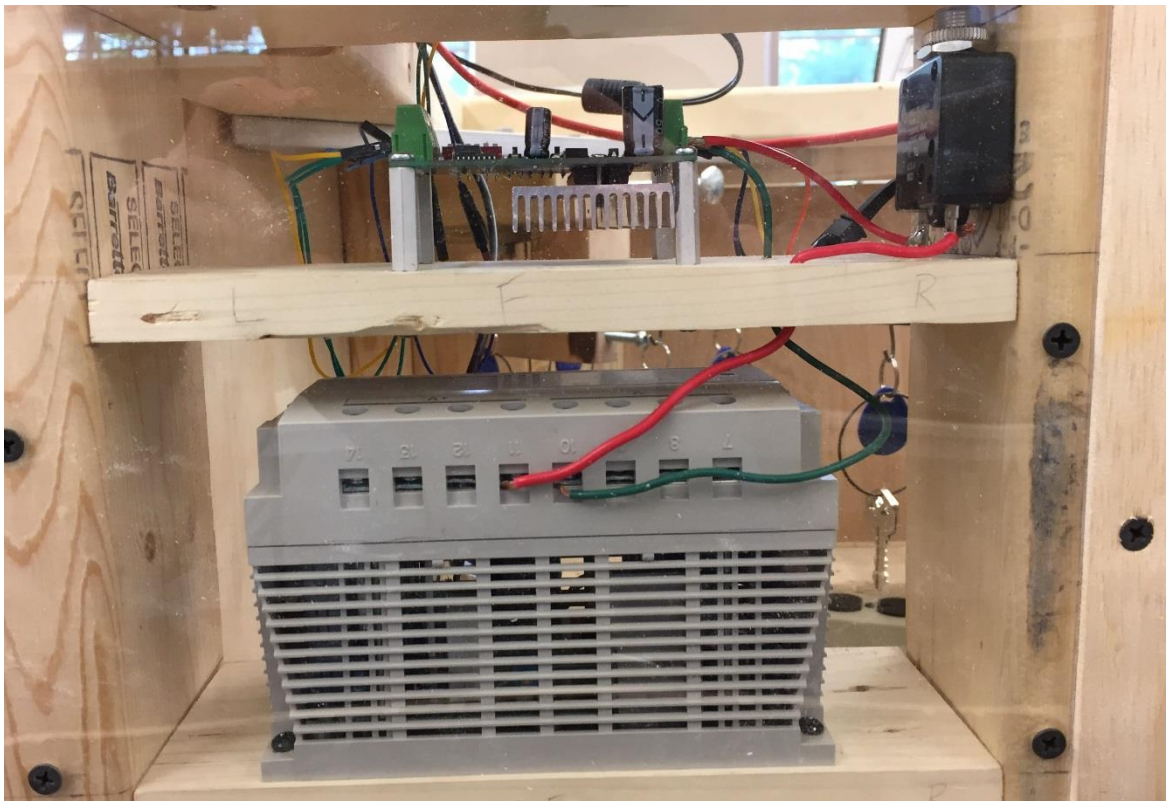

Figure 24: Final Motor Driver and Power Supply Placement in IdentiKey Prototype

Figure 25: Final Motor Placement in IdentiKey Prototype

4.6 Structure

As stated earlier, the structure of IdentiKey was decided to be constructed using wood primarily with various metal and plastic components to complete the enclosure. This structure adequately fulfilled its purpose as a housing for the prototype and allowed the concept to be easily communicated with people who interacted with the system. Wood was chosen as the building material due to cost limitations, available time, and experience on the team. It was not the focus of the project, but there needed to be a decent amount of consideration due to the security needs of the system as a whole and, due to tension forces, the frame would need to be able to withstand to prove the concept.

The frame was tested as soon as it was built to ensure no bowing or bending occurred under considerable pressure from opposite corners. This was important because the motor and adjustable opposite pulley would need to rely on the rigid frame to hold the components in place and keep them from moving. Once the frame was crudely tested, the construction of the motor and adjustable pulley mount could be continued.

The belt and pulley system was incorporated into the frame using fasters to hold the pulleys in position. Testing the rigidity of the adjustable pulley tensioner was challenging. Through testing, several revisions were made to improve how rigid this tensioner was by adding a metal bar to reduce flexing and using a bolt to optimally tension the belt.

After these issues were tested and solved, all that remained for structure completion was the application of access doors on the front for retrieving and returning keys, the rear access door for maintenance purposes. In addition, the walls that would hinder users from having access to more than one hook on the belt at once and the aesthetic trim to bring the look of the structure together needed to be constructed. These steps were completed simultaneously and were incorporated together seamlessly, providing a simple but effective system that was a pleasure to use and to look at.

Figure 26: Pulley Tensioning Mechanism

5. Business Plan

5.1 Potential Customers

The potential customers for this product are large organizations that have hundreds to thousands of keys. These large institutions often want to prevent employees from having total access to all areas of a building or campus, making a master key impractical. Some possible customers that have these problems are colleges and large businesses. Colleges often employ student workers that are allowed only limited access to the campus and therefore require a specific set of keys. Large businesses may employ outside companies to do work on their campus (e.g. janitorial services, external consultants, repair services) and want those companies to have limited access to their campus. Other large business may have many vehicles that all have specific keys (e.g. trucking companies, taxi services). These companies need a way to easily identify and store those keys, which will provide quick and easy access to a specific vehicle.

5.2 Target Market

Of these many potential customers, we will primarily be targeting college campuses. We will specifically target Calvin College, since we are working closely with their physical plant to provide a unique solution to the key storage problem. Once implemented at Calvin College, IdentiKey's design can be easily and minimally modified to fit other college campuses or large businesses.

5.3 Distribution Implementation

Concerning the distribution of our product, we will only be creating one prototype of IdentiKey. This prototype will be implemented on Calvin College's campus as a model for future products. Because IdentiKey will already be able to handle any type of key or ring of keys, it will require minimal modifications in order to be implemented at another organization. This means that the production and distribution processes for IdentiKey for different customers will be virtually the same. The only requirement that will differ based on the customer is the quantity of keys. Since each organization will have a different number of keys, the physical construction of IdentiKey should match the volume needed. This may require several systems working conjunction or a more advanced conveyor system. Thus, it will be difficult to mass-produce IdentiKey and sell it wholesale. Additionally, due to the naturally small customer base that IdentiKey targets, it is unlikely that it would need to be produced in a quantity that would drastically alter the production costs. Future products will therefore be sold retail directly from our website. This will allow us to implement a just-in-time manufacturing model where products are built and distributed based on the current demand. In addition, we will then be in direct contact with our customers

and will be able to aid them in the implementation process by adding their specific set of keys into the memory of an IdentiKey unit.

5.4 SWOT Analysis

5.4.1 Strengths

The strengths of IdentiKey lie in its comprehensiveness. There are similar products on the market, such as Traka's Intelligent Key Management System, but none that offer the same range of capabilities as IdentiKey. IdentiKey provides the same basic functionality as other key storage products: security, key identification, and simple user interface. However, IdentiKey also fully automates the key storage process, unlike any other key storage products. This eliminates human error when putting the keys in their proper storage location. IdentiKey also provides the user with information about the key being stored or retrieved, takes care of the case where there are multiple copies of a single key, and retains information about who currently has a certain key (based on the user ID before a key is accessed). All this ensures that a user will always be able to find and obtain the correct key. IdentiKey is also capable of handling a large number of keys at once, unlike other similar products on the market that are capable of handling only 20 to 30 keys.

5.4.2 Weaknesses

The weaknesses of IdentiKey could lie in its size and price. IdentiKey will most likely be slightly larger than other products on the market and will take up more floor space. Of course, this coincides with the fact that IdentiKey will also be able to handle more keys than traditional key sorting products. IdentiKey will also most likely be more expensive than other products given that it is a more sophisticated and complicated machine. Both of these drawbacks could prompt a customer to choose a cheaper, smaller option.

5.4.3 Opportunities

The opportunities for IdentiKey lie in the fact that it is taking advantage of a relatively small market. There are only a small number of key sorting and storing products on the market, and those that exist are not nearly as sophisticated or capable as IdentiKey. This presents the opportunity for IdentiKey to become the premium key sorting product on the market. In addition, the opportunity exists for IdentiKey to make RFID tags unnecessary. With a larger budget and further advances in technology, IdentiKey could implement a machine vision system to identify keys. This would eliminate the need to add RFID tags to the keys and thus reduce the initial cost of implementing a key sorting machine such as IdentiKey. If a

machine vision system were implemented in IdentiKey, it would have yet another advantage over other similar products.

5.4.4 Threats

The threats to IdentiKey lie in changing trends in technology. Card swipe readers or RFID scanners are replacing many door locks. In addition, many cars are switching over to push-button ignition instead of key ignition. These trends reduce the need for physical keys and, therefore, the need for a large key sorting machine. As the number of physical keys diminishes, other smaller key sorting products may become more appealing to customers. However, physical keys will not become obsolete anytime soon due to the expense of new door locking technologies. It is not practical for businesses to replace every single door lock with RFID scanners, especially when it is a low-use, low-security door, such as a janitor closet. Key cards may also have issues when the power fails, prompting businesses to keep physical keys on hand, even after installing RFID scanners.

5.5 Cost Analysis

All the components Team 10 purchased throughout the year are listed in the expenses chart in the Appendix. The first Raspberry Pi purchased by the team was supplied by the Engineering Department of Calvin College for a lab project the team completed, and therefore did not come out of the team budget. The team incurred several extra costs throughout the semester as components needed to be replaced; however, even with these extra costs, the production of the IdentiKey prototype fell within the budget of $500.

Together, these parts form a rough estimate of how much it would cost to reproduce the physical IdentiKey system. Some of the components purchased were duplicates, upgrades, or replacements. In these cases, the prices of the original components are not included in the cost estimate. The parts not included in the cost estimate are the first Raspberry Pi, the first RFID reader, the first and second stepper motor drivers, the RFID test cards and tags, one of the third stepper motor drivers, and one of the logic level converters. After removing these unnecessary costs, the structure of the IdentiKey prototype was valued at roughly $360. However, for the final design of IdentiKey, Team 10 would make a few physical changes to the system, as discussed in the Future Work section. These changes would likely increase the overall cost of the physical system. Considering final changes, it is feasible that the physical components of IdentiKey would cost roughly $500. It is probable that the cost of components would go down as more IdentiKey systems were ordered and built; however, due to the reasonable quantity of IdentiKey units the team would anticipate selling, it is unlikely that the cost advantages would significantly alter the total cost of production.

The team put in a substantial amount of time to design, build, and program IdentiKey. However, it is assumed that in the future Team 10 would be able to recreate the system in a significantly shorter amount of time. For a standard sized unit that did not require additional customization, it is predicted that it could be constructed in eighty hours. These hours would exclusively be dedicated to assembly. Assuming a $30/hour labor rate, the assembly of IdentiKey would cost $2,400.

Therefore, excluding the various business costs that would come with production, it is believed that standard IdentiKey systems could be reproduced for roughly $3,000 each. Because IdentiKey would be sold to institutions and not individuals, this cost does not seem unreasonable.

6. Future Work

Team 10 dedicated their work over the course of the academic year to creating a prototype of IdentiKey. The prototype proved to be fully functional and served as a successful proof of concept. However, the prototype also demonstrated several areas, which were outside of the original scope of the team's senior design project that could be expanded upon to improve the final design of IdentiKey.

6.1 Physical Security

IdentiKey was a successful prototype and had no major security concerns as long as only upstanding citizens with good intentions used it. However, IdentiKey cannot blindly depend on the morals of its users. For this reason, the system would need enhanced physical security. The structure would need to be built in a way that an average person could not dissemble the system and take the keys. This would mean using metal as the primary construction material in lieu of wood, and not leaving exposed fasteners that could be easily removed. Additionally, the access doors would both need locking capabilities. The front access door, where standard users retrieve and return keys, would need a programmable lock that would automatically prevent the door from opening whenever the user is not supposed to be accessing the key in front of the door.

6.2 Software Security

While the software controlling IdentiKey was fully functional for standard users, it lacked the security and full admin functionality that Team 10 would strive for before putting IdentiKey on the market. The database of information for user logins and key rings was stored inside of the programs that operate IdentiKey. While no user should have access to those files, it would be naïve to assume that nobody would find a way to look at them. Therefore, it would be necessary to implement a secure form of encryption that prevents any person from accessing the information or system controls located within IdentiKey. Additionally, several admin-level features could be added to IdentiKey to improve its overall usability. These features would enable admins to manage the key inventory, adjust users and their access level, temporarily freeze the system from use, check who currently has a specific key ring, see the history of a user's access, and other managerial functions. Ideally, these functions would be accessible from offsite through a desktop or mobile app.

6.3 Expanded Automation and Storage

As mentioned previously, it is highly likely that institutions attempting to implement IdentiKey will have more than ten rings of keys. The conveyor system implemented in IdentiKey is optimal for this because it offers a lot of scalability. However, the expansion may not be entirely trivial. Therefore, if Team 10 were to bring IdentiKey to the market, the team would first work to implement an expanded conveyor system that is capable of holding more keys. The expanded system would wrap around within the IdentiKey structure to utilize the available space fully. Additionally, a larger motor and conveyor belt could be implemented to sustain more keys.

6.4 Increased Usability

Another aspect of IdentiKey that could be improved upon is the usability. If Team 10 were to bring IdentiKey to the market, the team would find a way to make the processes of key return and retrieval more user-friendly. One improvement would be to increase the robustness of the RFID reader so that the user would not have to place keys on a hook with a specific orientation in order for the keys to be successfully read by the system. This would make the system easier to use, especially for new users. Another improvement would be to add an RFID card swipe reader to the system to allow a user to login instead of using a username and password system. This would allow a user to login to the system more quickly without having to remember a password.

7. Conclusion

IdentiKey enables the secure and efficient management of physical keys. The myriad of security and accessibility features offered by IdentiKey separate it from any other key management solution currently on the market. Users can request or return keys without needing to worry where to place the key or how to keep track of it. In addition, keys are kept secure by preventing users from accessing keys they are not authorized to use.

Team 10 learned a lot in regards to working on a project of this scale and working with a team for a long period. The experience of initial brainstorming, to making critical design decisions, to testing and working on system integration proved extremely valuable.

8. Acknowledgements

8.1 Professor Mark Michmerhuizen

Professor Michmerhuizen served as the advisor for Team 10. He is a wise sage who helped guide the design process throughout the duration of this yearlong course and gave helpful feedback on documentation, presentations, and scheduling.

8.2 Phil Jasperse

Phil Jasperse was essential in the construction of the exterior frame of IdentiKey. He also provided scrap wood and plexiglass for the side panel. His oversight and guidance was greatly appreciated throughout our metalworking and wood working process.

8.3 Professor Yoon Kim

Professor Kim's knowledge on electro-mechanical systems and analog systems helped us out when figuring out our optimal motor setup. Professor Kim also provided input on to certain hardware components that would work best for our project.

8.4 Eric Walstra

Mr. Walstra, from Gentex Corporation, was Team 10's industrial advisor who provided helpful feedback to the team regarding brainstorming, planning, system architecture, and the prototype concept.

8.5 Calvin College Engineering Department

Special thanks is extended to the Calvin College's Engineering Department for making this project possible by providing funding, a faith-based, rigorous education, and a generously-sized project space in the Engineering Building. Thanks to Michelle Krul for her logistical work and Bob DeKraker for acquiring the materials necessary for our project.

9. References

[1] https://www.raspberrypi.org/products/camera-module/

[2] https://www.youtube.com/watch?v=L5tx64G1ilQ

[3] https://www.youtube.com/watch?v=2EDNwUOcdrM

[4] http://7bot.cc/

[5] http://www.lynxmotion.com/c-27-robotic-arms.aspx

[6] http://store.ufactory.cc/uarm-metal/

[7] https://www.youtube.com/watch?v=Hz8hzTWVtNQ

[8] www.trakausa.com

[9] www.morsewatchmans.com

[10] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[11] https://www.arduino.cc/en/Main/ArduinoBoardUno

[12] https://beagleboard.org/black

[13] http://makezine.com/2013/04/15/arduino-uno-vs-beaglebone-vs-raspberry-pi/

[14] https://www.element14.com/community/docs/DOC-78156/l/raspberry-pi-7-touchscreen- display

[15] https://kivy.org/#home

[16] http://www.mpja.com/RFID_NFC-Reader_Writer-Raspberry-Pi-Arduino-Compatible-Sensor/productinfo/31111+MP/?gclid=CNHC76e3l9ACFRAxaQod5zwJiQ

[17] http://www.traka.com/en-US/site/traka-usa/usa-applications/intelligent-key-management/

[18] https://images-na.ssl-images-amazon.com/images/I/913XYU1VtjL._SL1500_.jpg

[19] https://cdn.sparkfun.com//assets/parts/4/0/9/5/09963-01.jpg

[20] https://cdn.sparkfun.com//assets/parts/8/1/8/8/11828-01.jpg

[21] https://images-na.ssl-images-amazon.com/images/I/41VIed27UPL._SY355_.jpg

10. Appendices

10.1 Appendix 1: Gantt Chart – Fall Semester

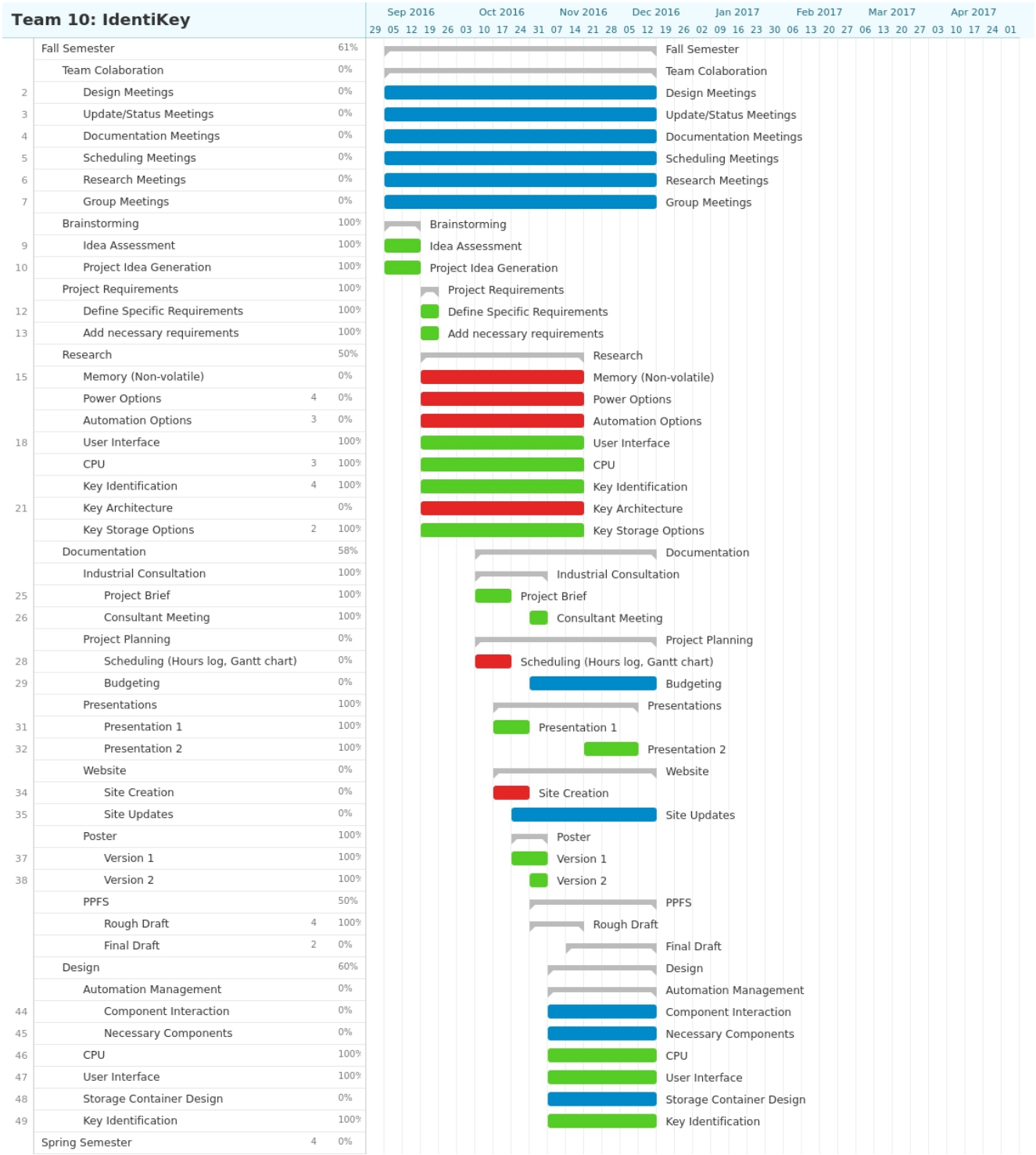

Figure 27: Team 10 Gantt Chart—Fall Semester

10.2 Appendix 2: Hourly Work Breakdown – Fall Semester

Table 8: Hours Breakdown Fall Semester

Hours Breakdown by Team Member						
Member	Brainstorming	Requirements	Team Collaboration	Research	Documentation	Totals
Drew	2	1	15	4	18.5	40.5
Brian	2	1	15	4	18.5	40.5
Jonathan	2	1	25	5	22.5	55.5
Jared	2	1	25	5	18.5	51.5
Totals	**8**	**4**	**80**	**18**	**78**	**188**

10.3 Appendix 3: Gantt Chart – Spring Semester

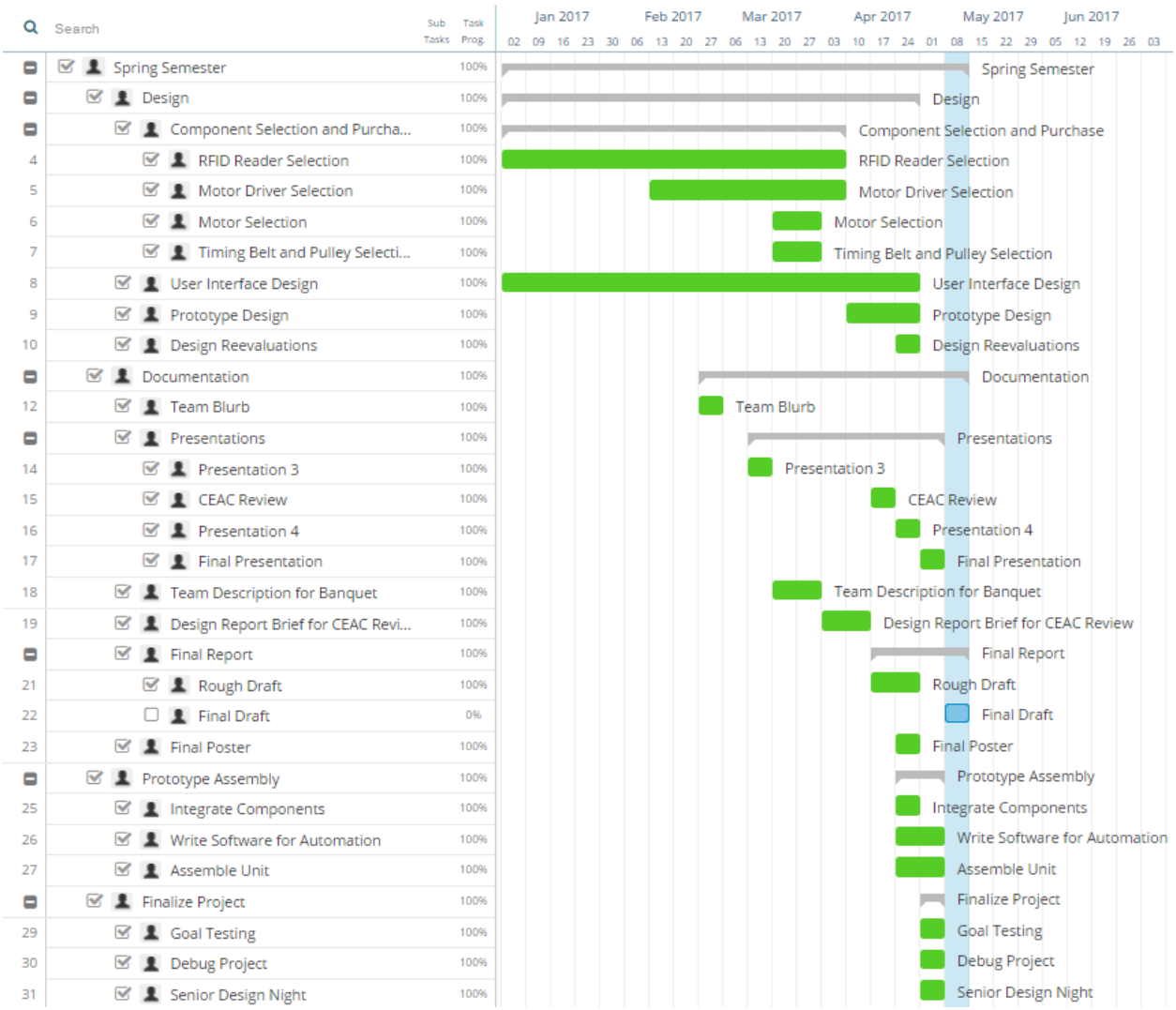

Figure 28: Team 10 Gantt Chart—Spring Semester

10.4 Appendix 4: Hourly Work Breakdown – Spring Semester

Table 9: Hours Breakdown Spring Semester

Hours Breakdown by Team Member					
Member	Design	Documentation	Prototype Assembly	Finalization	Totals
Drew	15	10	30	4	59
Brian	5	3	30	2	40
Jonathan	20	15	10	5	50
Jared	20	3	30	3	56
Totals	**60**	**31**	**100**	**14**	**205**

10.5 Appendix 5: Detailed Budget

Table 10: Team 10 Budget Breakdown

Team 10 Budget				
Date	Part	Cost	Budget Before	Budget After
11/7/16	Raspberry Pi	$33.53	$500.00	$500.00
11/7/16	Touchscreen	$70.00	$500.00	$430.00
11/7/16	RFID Reader	$8.00	$430.00	$422.00
2/15/17	Motor Driver	$25.06	$422.00	$396.94
3/27/17	Motor	$35.50	$396.94	$361.44
3/27/17	Motor Driver #2	$11.99	$361.44	$349.45
3/28/17	Timing Belt	$20.38	$349.45	$329.07
3/28/17	Timing Pulleys	$25.94	$329.07	$303.13
4/3/17	Raspberry Pi	$38.27	$303.13	$264.86
4/5/17	RFID Reader	$24.95	$264.86	$239.91
4/5/17	RFID Antenna	$34.95	$239.91	$204.96
4/5/17	Mini USB Cable	$3.95	$204.96	$201.01
4/5/17	RFID Test Card	$1.95	$201.01	$199.06
4/5/17	RFID Test Tags	$4.85	$199.06	$194.21
4/8/17	Motor Driver #3	$11.00	$194.21	$183.21
4/8/17	Motor Driver #3	$11.00	$183.21	$172.21
4/8/17	Logic Converter	$2.95	$172.21	$169.26
4/8/17	Logic Converter	$2.95	$169.26	$166.31
4/8/17	Header Pins	$1.50	$166.31	$164.81
4/10/17	RFID Tags	$8.60	$164.81	$156.21
4/10/17	Key Rings	$9.03	$156.21	$147.18
4/11/17	Lumber	$9.73	$147.18	$137.45
4/27/17	Various Hardware	$53.55	$137.45	$83.90
5/4/17	Key Labels	$9.53	$83.90	$74.37