Key Master (New)

Team: Zerui An (zeruian2), Tingfeng Yan (ty7), Celine Chung (mwchung2)
ECE445 Design Document - Spring 2020
TA: Ruomu Hao

1 Introduction

1.1 Problem and Solution Overview

How many keys do you carry around? One probably wouldn't feel any inconvenience if there are only a few keys to manage. However, it would be a pain to memorize all the key-lock pairs if one owns many keys. According to several polls hosted on online forums[1][2], more than 30% of people own 6 or more keys. Managing a lot of keys is not easy, so these people would surely benefit from a good key organizer. To help these key owners match a key to a lock easily, we plan to design a key holder that is able to recognize a lock and find the key to it.

In fact, some companies, like Orbitkey, have already noticed this problem and developed their own solution[3]. However, their products are just high-quality key holders which are unable to match a key to a specific lock.

Our design, on the other hand, "memorizes" the key-lock pairs for users, in addition to being a key holder. Once a key-lock pair is registered, one can look up the correct key by scanning a QR code attached to the lock.

This problem has also been addressed by Sp17 group 14. Their solution used RFIDs instead of QR codes to identify locks, and LEDs instead of numbers to locate the right key. More details will be discussed in the "Project Differences" section.

1.2 Visual Aid

To find a key, the user would need to press a button on our product to enable the scanner, then scan the QR code attached near the lock. Once the right key is found, its index would be shown on the 7-segment display. This process is illustrated in Figure 1.

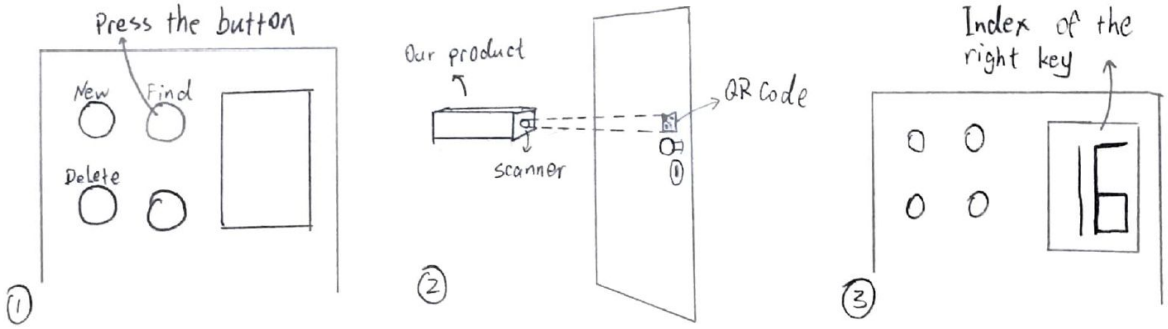

Figure 1. Product Usage

1.3 High-level Requirements

- The product must respond within 7 seconds after a QR code is scanned.
- The product should be able to process 150 find requests without replacing the battery.
- The accuracy should be at least 90%

2 Design

2.1 Block Diagram

As shown in Figure 2, our block diagram, we plan to use alkaline batteries (with regulator) to supply 5V. Signals in between the blocks will be implemented as 5V TTL.
Key storage block is not shown in the figure because it requires no electric signals.

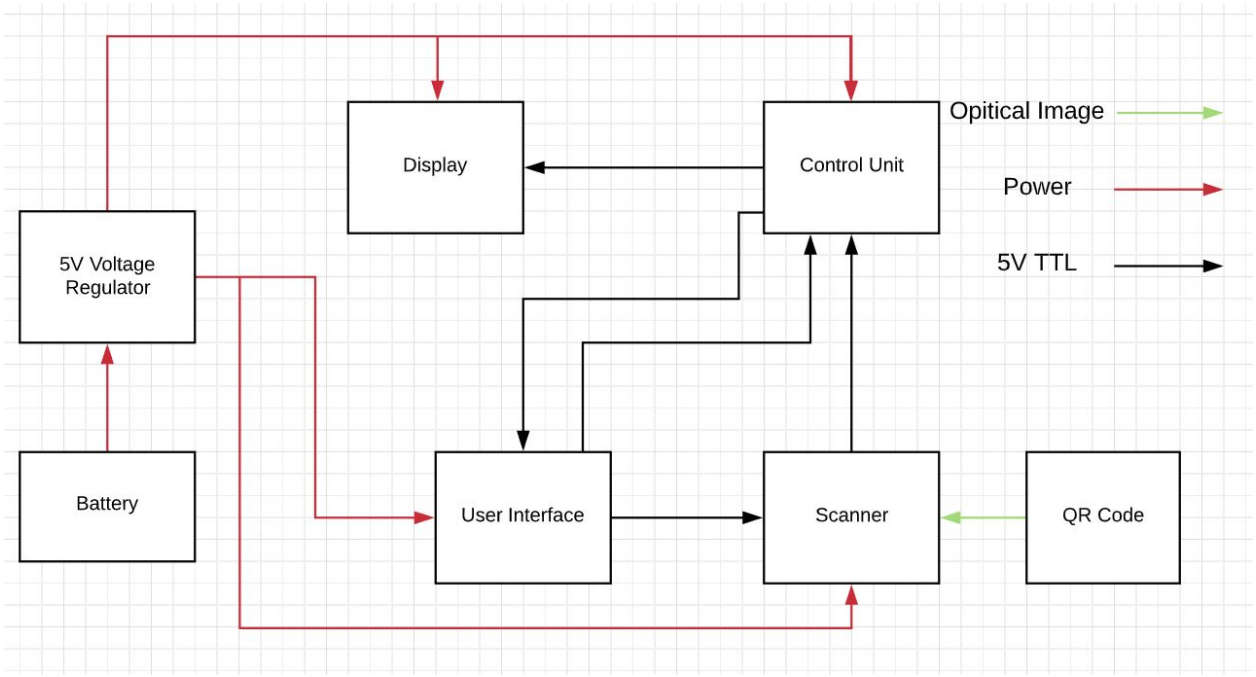

Figure 2. Block Diagram

2.2 Physical Design

A rough sketch of the physical design is shown in Figure 3. Our product is box-shaped, with a scanner on the top and user interface (including buttons and display) on the front. Keys are kept inside the box, and a number is labeled on each slot. Batteries are installed in the back (not shown in the figure).
Since we are designing a hand-held device, the size should be roughly the same as a wallet.

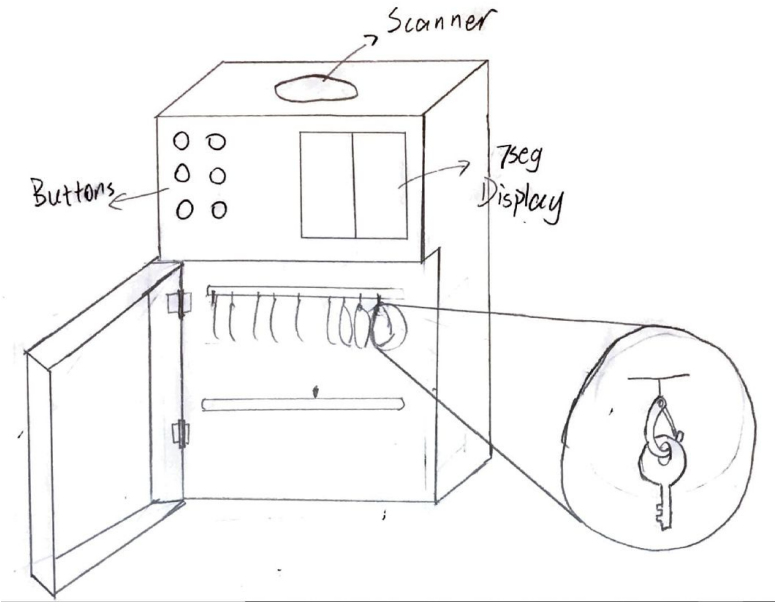

Figure 3. Physical Design

2.3 Block Requirements and Verification

2.3.1 Control Unit

The control unit takes processed image data from the camera and then compares it with existing data in the key storage which the user saved in advance. The control unit will send proper signals to the display according to whether it finds the right key or not.

The control unit is also able to receive orders like “New”, “Delete” from the user interface and communicate with the key storage to do the right work. Once the control unit switches its working state, it will also give feedback to the user interface, changing the LED color. For example, blue while it’s finding keys, green while it’s adding a new key, and red while it’s deleting an existing key.

Meantime, the control unit should send the key position it’s working on to the display so that the user can receive correct information about changes.

Requirement	Verification
1. When receiving Delete(000) from the user interface, the control unit should be able to mark the corresponding slot as “unused”. 2. When receiving Select1(001) or	Run the control unit code on Arduino, and make it continuously print the state of a few selected slots, as well as other system states such as index counter, so that we can easily observe the effect of user commands.

Select2(010), the control unit should be able to switch the key position it's working on and send the correct index to the display within 1s. 3. When receiving Find(011), the control unit should be able to check the bit string which it just received from the camera with all existing index/string pairs. Once the control unit finds the same bit string, it will send the corresponding index to the display. In case it goes through all 24 key positions but finds no matched bit string, it will send 88 to the display. 4. When receiving New(100), the control unit should be able to pair a string (translated from a QR code) with an unused key position whose index will be delivered to the display at the same time. If the selected slot is already occupied, the display should become 88 and no new key would be registered.	1. A. Manually set the selected slot to "occupied", then send the Delete signal; B. Manually set the selected slot to "unused", then send the Delete signal; In both cases, the final state of the slot should be "unused" 2. Randomly send Select1 or Select2 to the control unit, and observe the counter value. Be sure to cover edge cases such as 24+1 and 22+10. 3. A. Assign a simple string to a slot, for example, "11111111". Then send the exact same string to the control unit during Find mode. Check if the control unit correctly sends the slot position after finding the match. B. Send an unused string to the control unit and check if the control unit correctly sends 88 when it finds no matched pairs. 4. A. Set a selected slot to "unused" and use New to pair a string to it. Check the printed slot states to see if paired correctly. Check if the index of the selected slot is sent by the control unit. B. Set a selected slot to "occupied". Press New when selecting this slot. Check if 88 is sent by the control unit. Also, check the printed slot states to see if the previous string is overwritten.

2.3.2 QR code scanner

The scanner is responsible for reading the QR codes.

Requirement	Verification
1. The scanner must send a character string to the MCU within 6 seconds after scanning a QR code	First, connect the scanner to the serial pins on an Arduino, then open a serial terminal to display the scanned string. Scan 20 randomly

2. The accuracy must be above 90%	generated QR codes, and verify that: 1. All the scans take no more than 6 seconds 2. At least 18 of the scans produce the correct result. (If the correct results are not provided by the generator, they can be obtained using any QR code reader app)

2.3.3 Key Storage

The key storage is where the keys are stored. Every slot has its unique index.
This is a pure mechanical module.

2.3.4 User Interface

The user interface is supposed to have the following buttons:

- Select1: by pressing Select1 once, the selected key position index will increment by 1. The index starts from 01 and ends with 24. 01 would be selected upon pressing Select1 at 24.
- Select2: by pressing Select2 once, the selected key position index will increment by 10. 10 would be selected upon pressing Select2 at 24.
- Find: checking image with all existing data to find the right key.
- New: Register a new key/lock pair at the selected position.
- Delete: Overwrite the selected key/lock pair to 0's.

The user interface also has LEDs indicating the current mode.

Requirement	Verification
1. Pressing Delete sends "000" to the control unit; pressing Select1 sends "001" to the control unit; pressing Select2 sends "010" to the control unit; pressing Find sends "011" to the control unit; pressing New sends "100" to the control unit. 2. The LED on the user interface turns green for New (add a new pair), blue for Find (find the right key), and red for Delete (delete a pair).	1. Using 3 LEDs to check for the right outputs. 2. Switch mode and observe the LED color.

2.3.5 Display

This is a 2-bit 7-seg display and it shows the index of the key position that is currently worked on. If the box has a maximum load of 24 keys, the display will be able to present number 01 to 24.
It should also be capable of displaying special characters in special circumstances (e.g. error).

Requirement	Verification
1. The key position should be displayed as a 2-digit decimal integer between 1-24. 2. The number 88 should be displayed in special conditions: pressing New when the selected slot or the new QR code is already occupied, or no matched key is found during Find mode.	1. Use Select1 and Select2 to set various key positions between 1-24. 2. Test New and Find mode to see 88 displayed.

2.3.6 Power Supply

We plan to use AAA batteries as the power supply. All our components require 5V Vin, so we need at least 4x1.5V batteries and possibly a 5V voltage regulator.

Requirement	Verification
1. Voltage should stay within 4-5V when current is about 300mA.	1. Connect the power supply to a 130ohm load, verify that supply voltage is within 4-5V.

2.4 Tolerance Analysis

As a hand-held device, our product needs a long battery life to be actually useful. If the batteries need to be replaced every few days, our product would actually introduce more problems than it solved.

We assume that our product would be used 5 times a day, each time for 20 seconds. Furthermore, we believe that changing the batteries every month is an acceptable frequency, which means our power supply needs to support 30 * 5 = 150 uses, equivalent to 150 * 20sec = 3000sec, or about 1 hour.

The scanner consumes 200mA current during operation[4], and we estimate that the current of other components sums to 100mA. Because the scanning process is supposed to complete within 6 seconds, less than half of the operation time. The average current draw of our product would be less than 200mA.

Thus, the required battery capacity is about 200mAh.

According to the battery capacity graph (Figure 4) and discharge curve (Figure 5) we found on the datasheet [5], the battery capacity is around 600mAh when discharge rate is around 250mA, and the voltage of a single battery won't drop below 1.0V until used for 3 hours. Therefore, we conclude that this type of battery would meet our requirements.

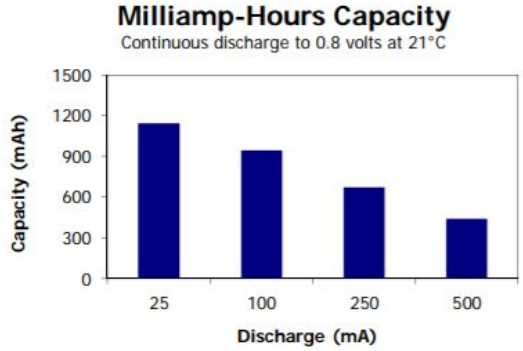

Figure 3. Battery Capacity

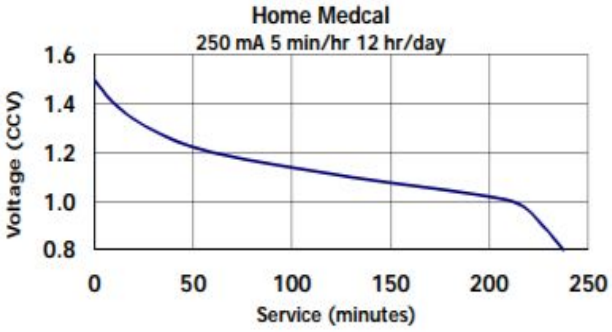

Figure 4. Discharge Curve

3 Project Difference

3.1 Overview

Our project is inspired by Sp17 group 14's Key Master project. In the original project, an RFID tag is attached to each lock. To identify a key-lock pair, one would use the RFID reader on the keyring to scan the tag on the lock, and the LED on the corresponding key would lit up.
The main difference in our projects lies in the method to identify each key-lock pair. We plan to use QR codes instead of RFIDs. Printed QR codes are much easier to acquire compared to RFID tags, so we believe this change would make the process of replacing/adding key-lock pairs much simpler.

We also made a few changes to the user interface. Instead of having one LED per slot (which may not be a good idea when there are lots of keys to manage), we plan to physically label each slot, and have a 7-segment LED display the selected slot number. We will also add a few buttons to support functions like add and delete key-lock pairs. By adding these buttons and displays, our product would have increased functionality but also increased size.

3.2 Analysis

We noticed an issue with the original RFID approach: To add a new key-lock pair, one would either have to purchase a new RFID tag or reuse one that is already attached to some other lock. To solve this problem, we plan to use QR codes instead, since they can be easily generated and printed.

However, we are aware that reading QR codes requires much higher computation power than reading RFID tags, which would result in longer response time in our new design.

The MCU is responsible for comparing a character string to around 20 strings and determine if there's a match. We claim that this is a fairly simple task and shouldn't take very long. We plan to use version 1 QR code, which generally contains less than 20 bytes of data [6]. In the worst-case scenario, the 20 bytes would be compared with all 24 previously-stored strings, byte by byte, which would sum up to 20x24=480 comparisons. We estimate that each loop contains 10 assembly instructions, including comparison, memory access, and other loop overheads. The total instruction per use would be 480x10 which is roughly 5K. The microcontroller we plan to use, ATMEGA328P, runs at 16MHz, and the average CPI is about 2 [7]. Therefore, we can expect the software program on the MCU to complete within 10K cycles, or 10K x 1/16MHz < 1ms, which is negligible compared to the delay in the scanner module.

The scanner module, on the other hand, would contribute to most of the delay. It is responsible for converting an image to a character string, which is a pretty heavy workload.
According to the datasheet, the response time of the scanner we chose may vary depending on the environment but is usually within 6 seconds. We consider this delay acceptable.

4 Cost and Schedule

4.1 Cost of labor

We assume that all people working on this project are compensated $50/hour. We will each be working on this project for about 10 hours a week for 11 weeks. The estimated machine shop work time is about 10 man-hours.

$$3 * \$50/hr * 10hr/week * 11\ weeks * 2.5 + 10 * \$25/hr = \$41,500$$

4.2 Cost of parts

Part	**Cost**
QR Code Reader Module	$41.99
2x 7-segment display	$3.99

Buttons	$5.89
LEDs	$5.65
Hooks	$12.99
4x AAA batteries	$4.49
Project Box	$24.99
ATMega328P Microcontroller	$2.08
Total	$102.07

Grand Total = $41,500 + $102.07 = $41,602.07

4.3 Schedule

Week	Zerui An	Tingfeng Yan	Celine Chung
1	Research UART Protocol	Design user-interface circuit (on breadboard)	Write controller code (on Arduino)
2	Test user-interface circuit	Create power-supply schematic	Create user-interface schematic
3	Create scanner schematic	Write controller test code (on Arduino)	Create controller schematic
4	Combine all schematics	Complete PCB version 1 design	Design & test power supply
5	Build controller circuit (on breadboard)	Test controller circuit	Research permanent storage on ATMega328P chip
6	PCB testing and Troubleshooting (focus on scanner)	PCB testing and Troubleshooting (focus on controller)	PCB testing and Troubleshooting (focus on user interface)
7	Finish the final PCB design	Finish the final PCB design	Finish the final PCB design
8	Reserved for unexpected delay		
9	Mock demo	Mock demo	Mock demo

10	Prepare for demo	Prepare for demo	Prepare for demo
11	Final demo, Final report	Final demo, Final report	Final demo, Final report

5 Ethics and Safety

The general safety of our design can be guaranteed because no hazardous or volatile material is used in our design. The mechanical design of our product is also user-friendly. The only safety concern might come from the overheating of some electronic components.

But, in the case when users lose their key-master, it would be worse than just losing a simple keychain. So, from the #7 of the IEEE Code of Ethics, “seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors” [8], we will try our best to solve this problem with our teammates. If we can’t solve the problem by ourselves, we will ask for help from the professor and TA.

Our product is box-shaped, with a scanner on the top and user interface (including buttons and display) on the front. We want to make sure its size is small and portable like a wallet to make it a hand-held device. Also, the weight of our product should be less than 1kg when we combine all the components.

We will use a battery for the 5V power supply to power our product when users are using it. We plan to purchase a AAA battery so users can use our product easily just buying a new battery. Therefore, we want to make sure that the battery and the supplied voltage to the circuit are within the safe range of 4-5V. To adhere to the IEEE Code of Ethics, #9 “to avoid injuring others, their property”. [8] regarding the safety concern about the power supply, we will be responsible for ensuring the voltage and current fed into all different modules won’t exceed their standard thresholds.

We believe that our design is in compliance with the IEEE Code of Ethics and the ACM Code of Ethics and Professional Conduct. By following the code #1 from IEEE Code of Ethics, we will make our design process and the final product as safe as we can and try to prevent any possible harm done to the user and society. Additionally, as stated in the ACM Code of Ethics code 2.9, “Design and implement systems that are robustly and usably secure”, our camera will be intuitive and easy to use for users. This will make sure that our key detecting system is made to make the user's life better.

References

[1] Kenny, “How many keys do you carry?”, *Pricescope*, 08-Oct-2012. [Online]. Available: https://www.pricescope.com/community/threads/how-many-keys-do-you-carry.180672/ [Accessed: 01-Apr-2020]

[2] Mediocrebot, “How many keys are you usually carrying?”, *Meh*, 07-Dec-2018. [Online]. Available: https://meh.com/forum/topics/how-many-keys-are-you-usually-carrying?sort=most-likes [Accessed: 01-Apr-2020]

[3] “Orbitkey - Our Story”, *Orbitkey*. [Online]. Available: https://www.orbitkey.com/pages/about [Accessed: 01-Apr-2020]

[4] Maikrt, “MC2500 Series Image Bar Code Reading Module Specifications”, 23-June-2018.

[5] Energizer, “E92 Product Datasheet”. [Online]. Available: https://data.energizer.com/PDFs/E92.pdf [Accessed: 16-Apr-2020]

[6] “Information Capacity and Versions of the QR Code”, *QRcode*. [Online]. Available: https://www.qrcode.com/en/about/version.html [Accessed: 16-Apr-2020]

[7] Cybergibbons, “Arduino misconceptions 2: Arduino is “slow””, *Cybergibbons*, 01-Feb-2013. [Online]. Available: https://cybergibbons.com/uncategorized/arduino-misconceptions-2-arduino-is-slow/ [Accessed: 16-Apr-2020]

[8]”IEEE Code of Ethics”, *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 02-Apr-2020]