1. Problem Definition

For this project, an engineered system with embedded intelligence needed to be designed. Our group decided to examine door locking mechanisms, as almost everyone is sure to encounter one in their daily lives.

For this project, we decided to implement a NFC door control system in order to help people save time and make their lives more convenient. This system would allow users to be able to lock and unlock a door using either a key card or their smartphone. By tapping or placing the card or phone on the sensor installed on the door, the door will automatically either lock or unlock, depending on which state it was previously in. This device includes a microcontroller, an NFC card reader, a solenoid to serve as the lock, and some LEDs to indicate if the card matched or not.

This could offer great convenience benefits to users as their door would no longer require them to carry around an easy to lose metal key. Users can either keep the key card in their wallet with them, or choose to use their phone, which would be most convenient as almost everyone carries their phone with them and it wouldn't require users to carry anything additional to unlock the door.

Additionally, according to our anecdotal research, it was found that a door with a traditional lock takes about 8-10 seconds to unlock whereas a door equipped with a keycard scanner takes 1-2 seconds. On average a family of 4 will lock/unlock their door a total of 8 times a day, or 2912 times a year [1]. From these numbers it can be said that 7-8 seconds will be saved each time the door is unlocked, or 5.7-6.5 hours per year. Split per person this would be 1.4-1.6 hours per year, which is a lot of time that could be spent on other things.

2. Functional Description

A traditional door lock requires users to carry around a set of keys, whereas with an NFC door lock we can grant access without requiring anything additional than just your phone or even a key card. The user interacts with the system by tapping their NFC card or device on the reader, which then reads the data from the card and sends it to the microcontroller. As NFC cards usually have a capacity between 96 and 512 bytes [2], it can be assumed that this sent data would fall into this range and likely closer to 96 bytes. This is sent through UART, transmitting the 96 byte packet at 57600 baud [3] in a 8-0-1 communications frame. The microcontroller then compares this value with the expected one stored in memory to determine if the door should change state (match) or remain in the current state (no match). If there is a match, the microcontroller will signal a green LED for 2 seconds using a timer to indicate to the user that there was a match, and it will send a signal to the solenoid to switch its state. If there was not a match, the microcontroller will signal a red LED for 2 seconds using a timer to indicate to the user that there was no match.

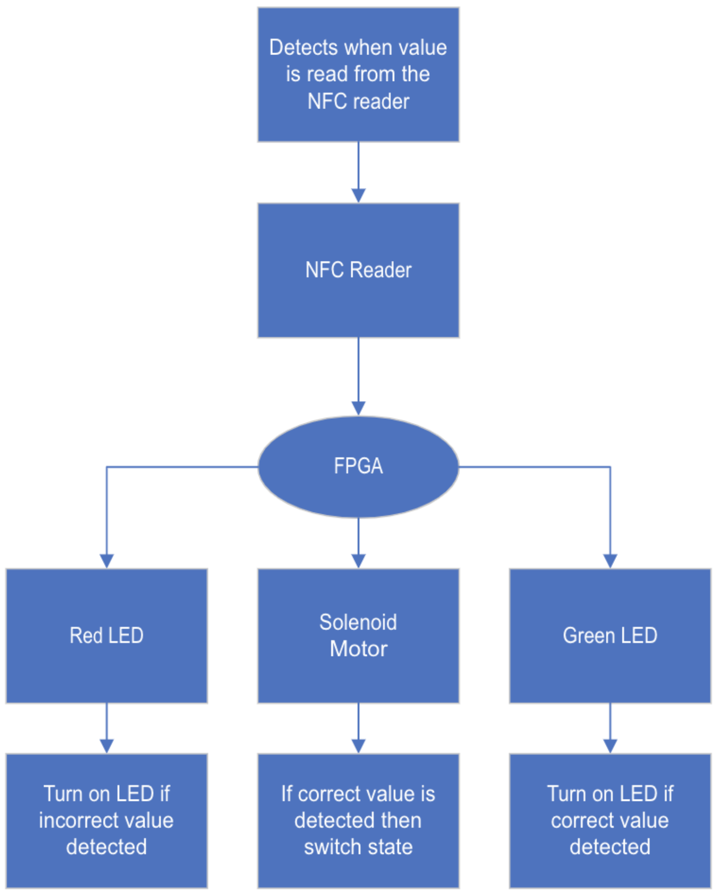
Detects when value is read from the NFC reader
NFC Reader
FPGA
Red LED
Solenoid Motor
Green LED
Turn on LED if incorrect value detected
If correct value is detected then switch state
Turn on LED if correct value detected

Figure 1: Block Diagram

3. Identify I/O Requirements

This design will include one input method, and two outputs. The input is the NFC sensor detecting the card or phone, and the outputs include the two LEDs displaying if the input was accepted and the solenoid changing the lock state. For the NFC sensor, the design would use the CR95HF-VMD5T, which is a NFC sensor capable of UART communication, which can connect directly to the microcontroller. This device is responsible for reading the input from the NFC card and then transferring this data to the microcontroller. The device will remain in an idle low power state until it detects an NFC tag, prompting it to switch to its ready state. Then, it will transfer the received data to the microcontroller through pin UART_TX, which will be connected to the microcontroller's appropriate UART receiving pin.

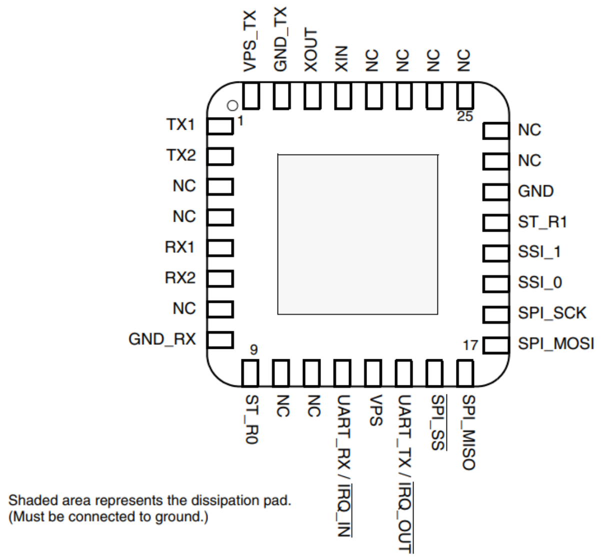

Figure 2: CR95HF-VMD5T pinout description

For the LEDs, the WP710A10ID5V (red) can be used and turned on when the key is incorrect, and the WP710A10GD5V (green) can be used to inform the user that their key matched. The power pin on these LEDs can be connected to the microcontroller's GPIO port in order to provide them with 5V whenever needed to turn on.

Figure 3: Green and Red LED

Regarding the locking mechanism, the DSOS-0416-05D solenoid can be used to control the lock's state. The solenoid requires power to be in its pushed out state, which corresponds to the door being locked. When there is no power, a spring retracts the plunger to the non activated position, which corresponds to the door being unlocked. This device can be connected to the microcontroller through the GPIO port, connecting the power pin to the port to provide 5V when needed. The microcontroller's GPIO port will therefore be required to have 3 pins set as output, one for each output device.

Figure 4: Solenoid Device

4. Initial Software Design

- What initialization does your software need to perform?

Upon startup, the software needs to do any initialization regarding the NFC scanner's reader/writer segments. Additionally, the appropriate GPIO pins must be set to output in order for writing to the output devices to work. Also, at this stage the software should set both LEDs to off by default and write the appropriate value to the solenoid to set the device into the locked state, plus it will store the value of the authorized key in memory. The software will also initialize the timer to count down at intervals of 2 seconds. This timer will be used later to only keep the LEDs on for 2 seconds after switching to on.

- How are your various inputs sampled? How is the data communicated to the rest of the software?

The only input will be received from the NFC scanner and will be gathered through the UART connection of the DE10-Standard Board. The contents of the received input will later be accessed through the on board memory.

- How are your various outputs computed?

It would compare the tag's ID with the authorized ID's in memory. If a match is detected it would activate the lock to open through the solenoid and enable the green LED. If the ID does not match the mechanism will be kept in its current state (either locked or unlocked) and the red LED will be enabled.

- What does your software spend most of its time doing? Can it run forever?

Most of the time the system is standing at idle with the NFC receiver running in its low power consumption mode which it can run indefinitely. The rest of the system will simply be waiting for input from that device. Once an input is received, it will go through its cycle then loop back to an idle state where it will wait once again for another input. This can theoretically run forever.

- How do the various components of your software (input processing, output processing, computation) interact?

The DE10 would need to sample the NFC reader and based on the readings provided would adjust the locking mechanism state, the input would be from the NFC reader and the outputs would be the LED and solenoid. The input will be used to modify a boolean value, this value will decide which LED will be turned on and whether to lock/unlock it or do nothing.

5. Prototyping Plan

The current testing plan includes testing the NFC scanner, LEDs and solenoid. Since none of these parts will be available when making the prototype, some of the DE10-Standard Board's built in features will be used for testing purposes.

As the NFC scanner will not be available, the DE10-Standard Board's JTAG UART will be used to mimic the input received from the real scanner. With the physical scanner, about 96 bytes of data would be sent through UART to the microcontroller, so this can be mimicked by sending about 12 bytes of data through the JTAG UART. The code would function identically with the actual data packet length of 96 bytes, so a shorter packet will be used for simplicity.

Since the LEDs would be connected to the GPIO port regardless, two pins of the DE10-Standard Board's GPIO port can be set to output to simulate the two LEDs. For additional visual feedback, two of the board's built in LEDs will also be used to represent them. The rightmost LED represents success, and the left one means failure. To ensure that the LEDs only stay on for the desired two second duration, the A9 Private Timer found on the board will be used.

As there is no equivalent alternative for the solenoid on the DE10-Standard Board, one pin of the GPIO will be used to indicate the solenoid's state. For additional visual feedback, the leftmost LED on the board will be used to visualize the door's status, with on meaning locked, and off meaning unlocked.

6. Select a Microcontroller

EK-TM4C123GXL	MSP-EXP430G2131 on MSP-EXP430G2ET	ARM Cortex-A9
Supports UART, has enough memory	Does not support UART, does not have enough memory	Supports UART, has lots of memory and I/O support
Inexpensive	Inexpensive	Expensive

Figure 5: Microcontroller Comparisions

As multiple components need to be connected externally and two 96 byte values need to be stored in memory, the Texas Instruments TM4C123GH6PM would be the best choice for this project. This device supports UART which will work great with the UART NFC scanner, it is relatively inexpensive at a cost of 29.89 CAD, and it contains enough memory to store both the authorized key and the inputted key.

Compared to the ARM Cortex-A9, this microcontroller uses significantly less power, and is also cheaper. Compared to the MSP-EXP430F2131, both microcontrollers have a similar cost, but the MSP-EXP430F2131's lacks UART support and does not have a lot of memory. This would create problems connecting it with the chosen NFC sensor, and more importantly not be able to store both the keys to do the comparisons, ultimately making the TM4C123GH6PM the best choice.

7. Revised Software Design

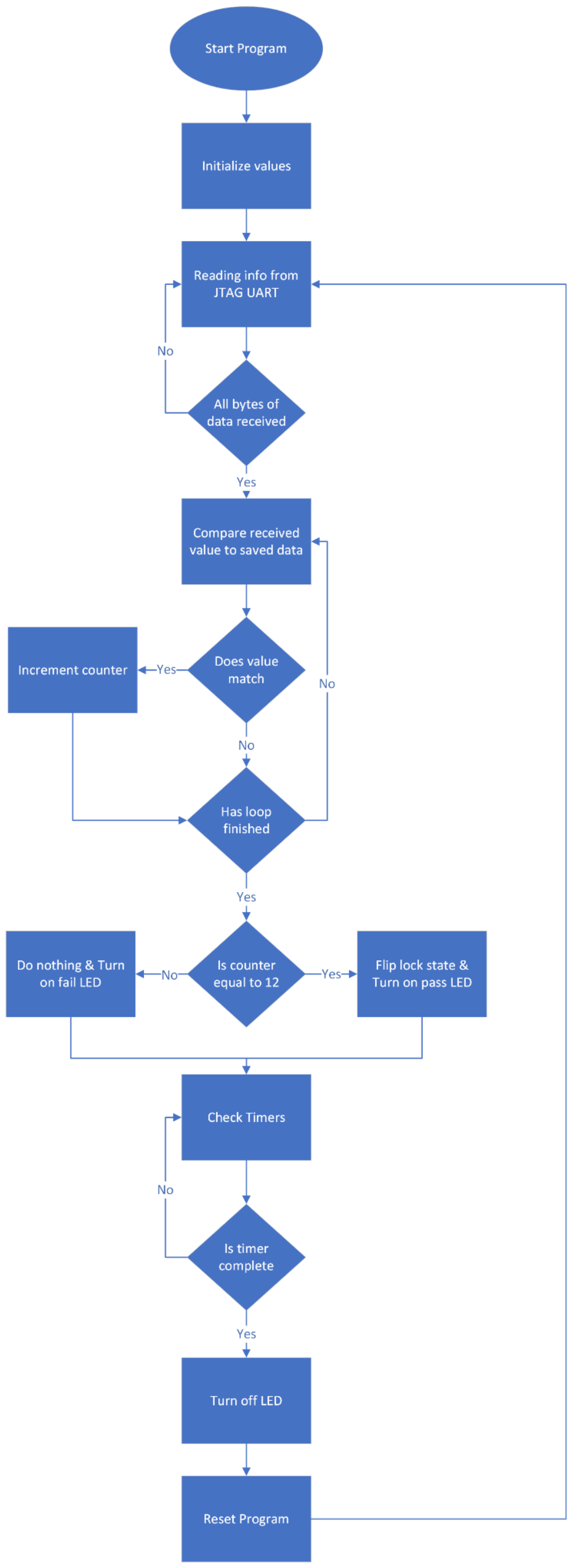

Figure 6: Program Algorithm Flow Chart

The revised software design remains largely the same as the initial design but with some small changes. The main change that was made between the initial design and the revised design is with how it is checked that the keys match. Originally, the design was to have a boolean value indicating if the input was a match, however this was changed to use a counter instead. In the revised software there is a loop that checks each of the inputted bytes and compares them to the authorized key's bytes, incrementing whenever there is a match. Then, if the counter's value is the same size as the total byte length of the key, there was a match. Other than this change, there were not many other software revisions that were identified. The only changes that were identified relate to program execution, such as including interrupts or implementing another timer.

When working with the ARM, not many problems arose, however some small issues were encountered. This includes the LEDs sometimes staying on for too short of a duration (less than 2 seconds) due to the implementation. Originally, the timer was always running, and the status flag would be checked after the LED was turned on. Since it was always running this would result in the LED staying on for less than 2 seconds sometimes. This was resolved by setting up the timer at program start up, but only starting it once the LED was turned on. Then, once the flag was activated, the timer was turned off to ensure that the next time it's started it will also count for a full 2 seconds. The other issue encountered was with reading from the JTAG UART. When first writing the code, it was overlooked that the last input remains in the bottom 8 bits of the JTAG UART's memory address, which would cause too many of the same input being written to memory. This was resolved by properly checking the ready bit to ensure that there was new data received and only writing that to memory.

While interrupts were not used in this implementation, they could be used to replace the main loop of the program. Currently, there is an infinite loop always waiting to receive new data through the UART port, checking on every loop if the ready bit of the UART is set. As the JTAG UART supports interrupts, these could be enabled instead of using the main loop, with the main loop's code running only when an interrupt is received. Additionally, the timer used also has support for interrupts. With the current implementation, there is a loop checking that the timeout bit is set to know if the timer interval has passed. This could also be replaced with an interrupt instead of using a loop. Essentially, interrupts could be used to replace all code dedicated to checking status bits.

8. Results from Prototyping

When trying the code on the board in the lab, everything worked as expected. All of the LEDs and GPIO outputs appeared as expected, and the timer counted down consistently for 2 seconds. Additionally, the LEDs connected through the GPIO port on the board were helpful to show that the GPIO port outputs worked, as those would be used in the real implementation. The only issue that was encountered was with sending data through the JTAG UART to the board, as the board was not receiving any data at first. However, this was remedied by checking through the project settings and properly enabling the ability for the terminal to send data through the JTAG UART.

Fortunately, there were not many unexpected challenges that slowed progress. When implementing the prototype, the only major issues that arose were bugs that needed to be resolved, such as writing the same input from the JTAG UART to memory several times and the LEDs not remaining on for long enough, both of these issues being described in the Revised Software Design section. While these issues needed some time and effort to resolve, it was not unexpected as bugs will almost always arise in software development.

However, the prototype implemented does have some drawbacks related to the lack of accurate testing for the UART NFC scanner. Although the JTAG UART provides a great way to input data to the microcontroller and test that the comparisons of the data to the authorized key work, it does not guarantee that in the real implementation that the data would be received properly. Once actual hardware is received, further testing would need to be done with the NFC scanner to ensure that it can properly send data to the microcontroller. Also, it would need to be tested that the NFC scanner properly receives the NFC card's data, and that the expected data is sent. Furthermore, testing should be done with a larger data size to ensure that the software would still work. While realistically it should function the same as before, only 12 bytes of data are used in the prototype compared to the 96-512 bytes of actual NFC cards [2].

The locking mechanism should also be tested once the hardware is ready. While it is expected that the implementation of the locking mechanism through the GPIO port should work properly with the actual solenoid, it cannot be confirmed without physically connecting all of the hardware. Additionally, testing with the hardware would provide insight into whether the solenoid would be strong enough on its own as a locking mechanism, or if any additional hardware would be required.

9. Source Code

Refer to the Code section in the appendix.

10. Conclusions

With the testing and prototyping done so far, the product is expected to work in actual implementation. Also considering that the device would simply need to fit within a door and that most of the parts are not large in size, it is likely that everything would fit well in the completed design. However, this would need to be tested. Regarding cost, the device is expected to cost about $57.43, as seen in Figure 7. The cost of traditional locking mechanisms vary, however it seems that on average the mechanisms cost about the same amount as the proposed price for this design, and sometimes a bit less.

Something to consider is that these traditional locks include a sturdier locking switch in the price, which is something that the current cost breakdown does not accommodate for. Including a better locking switch would likely need to be added to the design, therefore increasing the price by a bit. Also, the design would need some sort of casing for the NFC scanner, which would also increase the price slightly. This would likely bring the price to around $69.90 by adding a proper lock to the price [4], which would still be inline with traditional locks, although a little more on the expensive side. However, this is still significantly cheaper than existing electronic locks, making the price quite attractive and appealing to the general market. The price also becomes more attractive when the potential time savings, as discussed in the Problem Definition section, are considered.

PART	Cost (CAD)
EK-TM4C123GXL (Microcontroller)	29.89
CR95HF-VMD5T (RFID Reader/transponder)	6.35
WP710A10GD5V (Green LED)	0.78
WP710A10ID5V (Red LED)	0.80
DSOS-0416-05D (Solenoid)	6.78
B73180A0101M199 (Rechargeable Battery)	12.83
Total	57.43

Figure 7: Cost Breakdown

Once the final product is complete and commercially available, it would be packaged as one unit containing all of the parts, which can be installed within any door. Most doors come with holes to insert a traditional locking mechanism, so it would be beneficial for both consumers and the environment for the product to be able to work with existing cutouts. Requiring a unique cutout or installation process is not viable as it would cut out a large number of possible customers, and waste resources unnecessarily to create a new cutout.

In an additional effort to reduce e-waste and make customers' lives easier, the two pieces that will degrade the quickest, those being the battery and solenoid, should be connected to the main board in a way that they can be removed without replacing the entire unit. A rechargeable battery is included within the design, however batteries naturally degrade and lose their maximum capacity over time. Similarly, the solenoid is also only rated for a certain number of actuations, and could become weaker or not work at all eventually. Both of these devices should not be directly soldered onto the board, but rather attached with clips or straps. These clips or straps would be included with the product when shipped, however users can easily use third party ones if they choose. This will allow for users to service their own device and replace only these parts once they reach end of life instead of the entire unit. Also, a low power microcontroller is used, resulting in low power consumption, and therefore increasing the amount of time between charges and entire battery replacements.

The process of choosing a problem and coming up with a solution using a microcontroller has been a great learning experience. Unlike using alternative tools such as an Arduino or a Raspberry Pi, which simplify the process of integrating and controlling peripherals, working with the bare metal helped further our understanding of how microcontrollers function. Additionally, this project gave us an opportunity to apply many of the things learned in the course together, solidifying our knowledge of the course topics. This has also helped us see how microcontrollers are integrated within projects like this, and gives us an idea on how we could use them in future projects.

Appendix

Citations:

[1]
Richardson, I. (2019, November 14). *How many times A day do you open and lock your door?* Timber Composite Doors Blog. Retrieved April 10, 2023, from https://www.timbercompositedoors.com/blog/how-many-times-a-day-do-you-open-and-lock-your-door/

[2]
Dipole. (n.d.). *What is the NFC technology: Dipole RFID*. DipoleRFID. Retrieved April 7, 2023, from https://www.dipolerfid.com/rfid-blog/whats-is-nfc#:~:text=The%20capacity%20of%20NFC%20tags,between%2096%20and%20512%20bytes.

[3]
13.56-mhz multi-protocol contactless transceiver ... - stmicroelectronics. (n.d.). Retrieved April 9, 2023, from https://www.st.com/content/ccc/resource/technical/document/datasheet/d2/cc/cb/f6/7c/63/48/75/DM00025644.pdf/files/DM00025644.pdf/jcr:content/translations/en.DM00025644.pdf

[4]
Defiant single cylinder stainless steel Deadbolt DL61. The Home Depot. (n.d.). Retrieved April 12, 2023, from https://www.homedepot.com/p/Defiant-Single-Cylinder-Stainless-Steel-Deadbolt-DL61/100352212#ratings-and-reviews

Parts:

NFC Sensor:
https://www.digikey.ca/en/products/detail/stmicroelectronics/CR95HF-VMD5T/3182396
Green LED:
https://www.digikey.ca/en/products/detail/kingbright/WP710A10GD5V/3084184
Red LED:
https://www.digikey.ca/en/products/detail/kingbright/WP710A10ID5V/3084187
Solenoid:
https://www.digikey.ca/en/products/detail/delta-electronics/DSOS-0416-05D/6599945
Microcontroller:
https://www.digikey.ca/en/products/detail/texas-instruments/EK-TM4C123GXL/3996736
Battery:
https://www.digikey.ca/en/products/detail/epcos-tdk-electronics/B73180A0101M199/10413399

Code:

```
.global _start
```

```
_start:

        ldr r4, UART
        ldr r5, P_PORT
        ldr r6, LED_BASE
        ldr r7, TIMER
        ldr r8, KEY_START
        ldr r9, RES_START

@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@ START OF MAIN PROGRAM @@

        @ INITIALIZE TIMER
        ldr r0, =400000000 @ every two seconds
        str r0, [r7]

        @ INITIALIZE GPIO PORT
        ldr r0, =0x15 @ 0b10101
        str r0, [r5, #4]

        @ Set it to locked on startup
        mov r0, #1
        lsl r0, #4
        str r0, [r5] @ pin 4 in GPIO

        lsl r0, #5
        str r0, [r6] @ top LED

        @ INITIALIZE KEY
        ldr r1, =chars @ get array
        mov r2, #12 @ loop counter
        _loop:
        /* read next character from array, then
        increment array address */
        ldr r0, [r1], #4
        str r0, [r8], #4 @ write to address
        subs r2, #1 @ increment counter
        bne _loop

        @ INITIALIZE COUNTER
```

```
	mov r10, #0

	mov r2, #0

_main_loop:
	ldr r2, [r4] @ read from JTAG UART

	ands r1, r2, #0x8000 @ check if bit 15 is 1
	andne r2, #0xFF @ 0b11111111
	strne r2, [r9], #4
	addne r10, #1
	movne r2, #0

	cmp r10, #12
	blt _main_loop @ only continue once we receive all 12 bytes of data

	@ compare values
	bl _compare

	@ enable correct led and solenoid
	bl _set_state

	@ reset counter
	bl _reset

	b _main_loop
@@@ END OF MAIN PROGRAM @@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@ START OF SUBROUTINES @@@
_compare:
	push {r0, r1, r2, r3, lr}

	ldr r8, KEY_START
	ldr r9, RES_START
	mov r10, #0
	mov r2, #12 @ loop counter
	_cmploop:
	ldr r0, [r8], #4
```

```
        ldr r1, [r9], #4

        cmp r0, r1
        addeq r10, #1 @ if match, increment counter

        subs r2 , #1 @ increment loop counter
        bne _cmploop

        pop {r0 - r3, lr}

        bx lr

_set_state:
        push {r0, r1, r2, r3, lr}

        cmp r10, #12
        bleq _pass @ the keys match
        blne _fail @ no match

        @ start timer
        mov r0, #0b11
        str r0, [r7, #8]
        wait:
        ldr r3, [r7, #12]
        cmp r3, #0
        beq wait
        @ clear timeout flag
        str r3, [r7, #12]
        @ stop timer
        mov r0, #0
        str r0 , [r7, #8]

        @ turn off lights
        @ GPIO port
        ldr r0, [r5]
        mov r1, #1
        lsl r1, #4
        and r0, r1 @ set everything to 0 except for lock
        str r0, [r5]
        @ LEDs
        ldr r0, [r6]
        mov r1, #1
        lsl r1, #9
        and r0, r1
        str r0, [r6]

        pop {r0 - r3, lr}

        bx lr

_pass:
        push {r0, r1, r2, r3, lr}

        @ GPIO port
        ldr r0, [r5]
        mov r1, #1
        orr r0, r1 @ turn on light

        lsl r1, #4
        eor r0, r1 @ flip lock state

        str r0, [r5]

        @ LEDs
        ldr r0, [r6]
        mov r1, #1
        orr r0, r1

        lsl r1, #9
        eor r0, r1

        str r0, [r6]

        pop {r0 - r3, lr}

        bx lr

_fail:
        push {r0, r1, r2, r3, lr}
```

```
	@ GPIO port
	ldr r0, [r5]
	mov r1, #1
	lsl r1, #2
	orr r0, r1 @ turn on light
	str r0, [r5]

	@ LEDs
	ldr r0, [r6]
	mov r1, #1
	lsl r1, #4
	orr r0, r1
	str r0, [r6]

	pop {r0 - r3, lr}

	bx lr

_reset:
	@ reset everything
	mov r10, #0 @ reset counter
	ldr r8, KEY_START
	ldr r9, RES_START

	@ clear memory
	mov r0, #0
	mov r2, #12 @ loop counter
	_rst_loop:
	str r0, [r9], #4 @ write to address
	subs r2, #1 @ increment counter
	bne _rst_loop

	ldr r9, RES_START

	bx lr
@@@ END OF SUBROUTINES @@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ labels for constants and addresses
```

```
LED_BASE:       .word   0xFF200000
TIMER:                  .word   0xFFFEC600
P_PORT:                 .word   0xFF200060
UART:                   .word   0xFF201000
KEY_START:              .word   0x00001000
RES_START:      .word   0x00001030

.data
/* data structure for storing array
of characters */
chars:
.word 113 @ q in ASCII
.word 119 @ w
.word 101 @ e
.word 114 @ r
.word 97  @ a
.word 115 @ s
.word 100 @ d
.word 102 @ f
.word 122 @ z
.word 120 @ x
.word 99  @ c
.word 118 @ v
.text
```