Mini Project Report

Password Based Door Lock

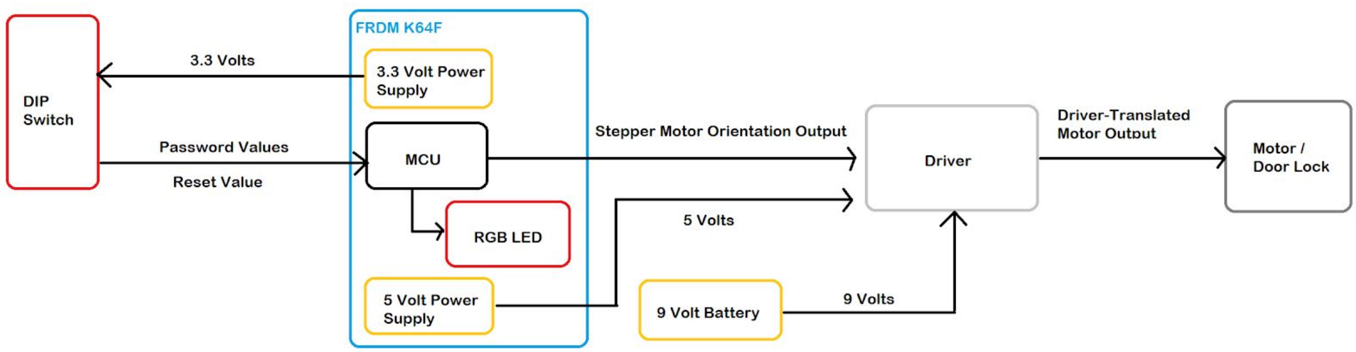
Michael Tang (861209107)
Jiahao Wang (862188757)
Group 10
June 7th, 2020
TA: Han Zhou
Section 021

Project Description

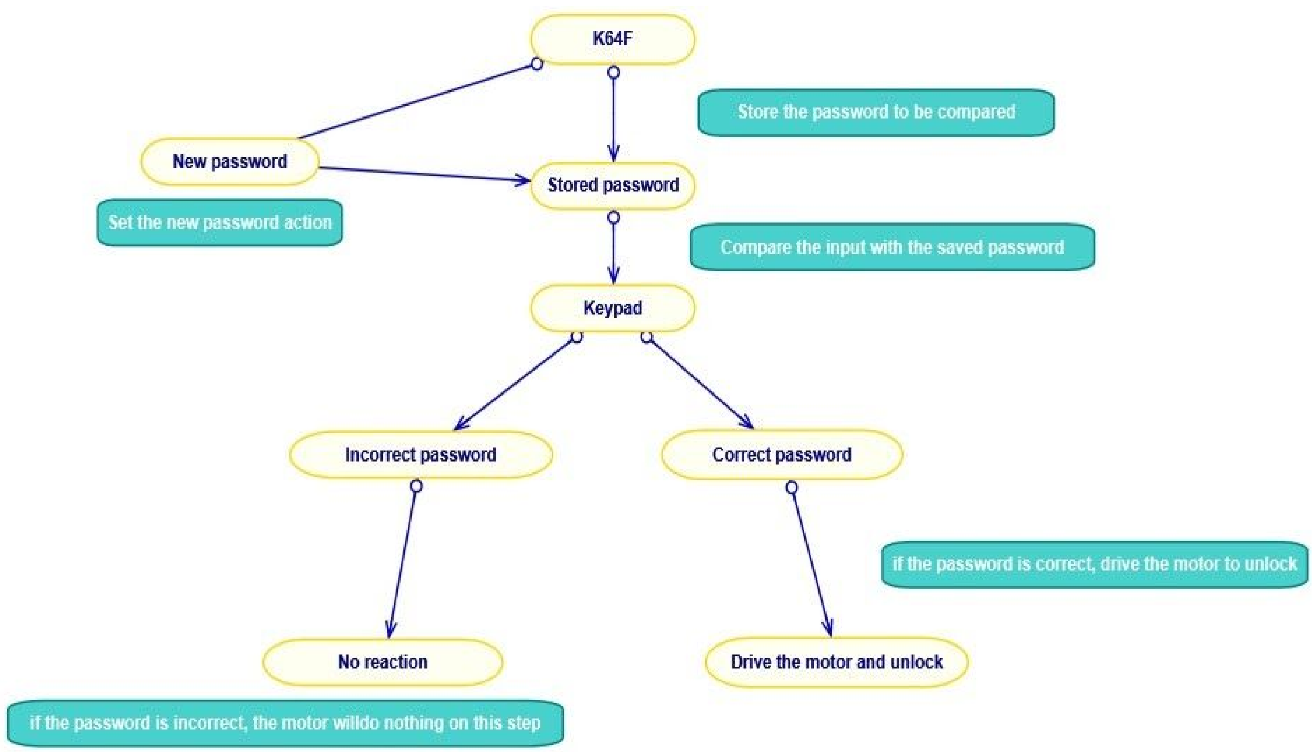
The Password Based Door Lock is a supplemental door-related security measure. This lock should be able to allow users to enter a password and only 'unlocks' when the correct password is entered. The lock must also automatically 'relocks' itself after a few seconds of a successful unlock. Additional reset features include: a reset function for the door lock and a 'led indicator' using the FRDM LEDs (Red: Wrong, Green: Correct, Blue: Reset).

System Design

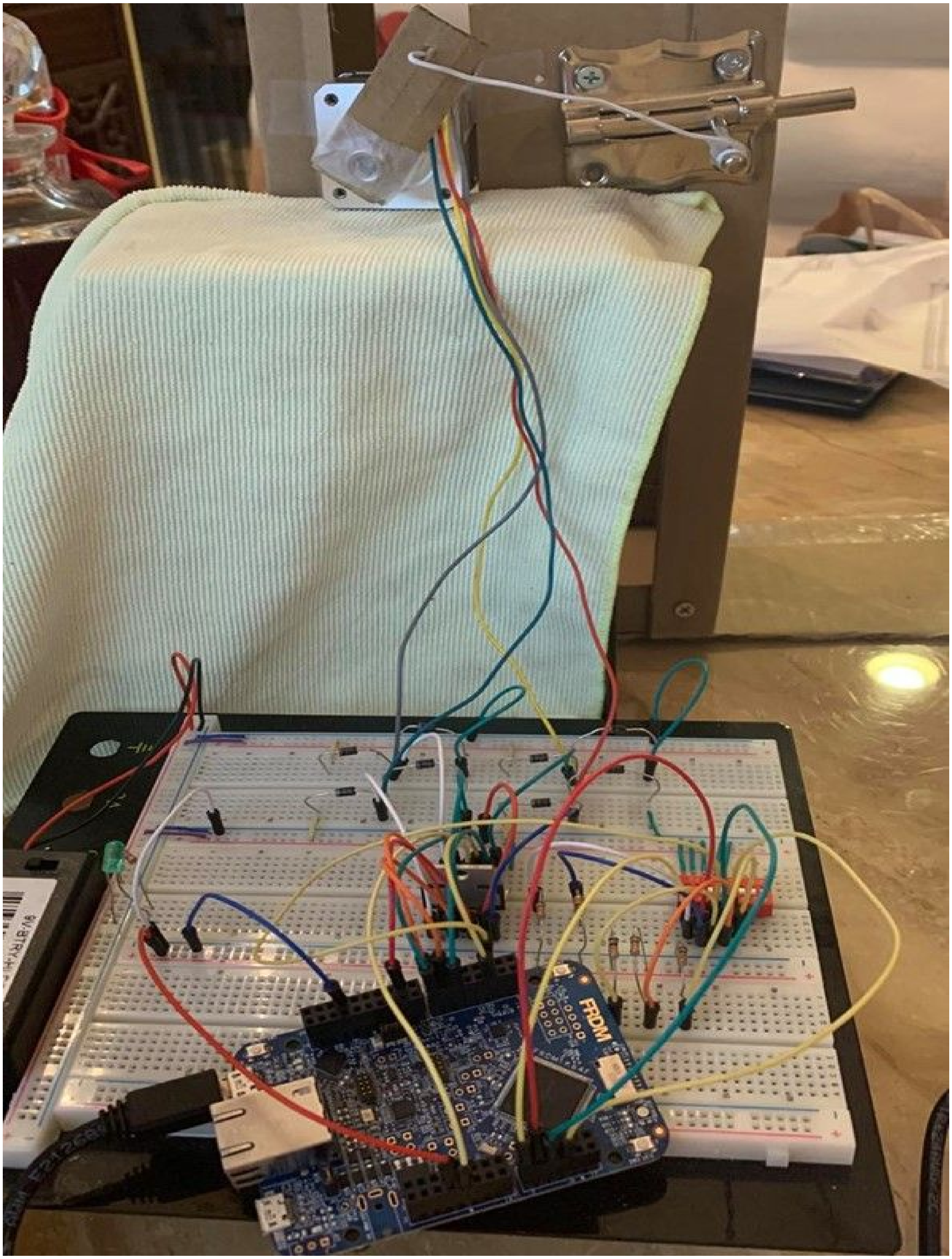
- Block Diagram:



- Flowchart:

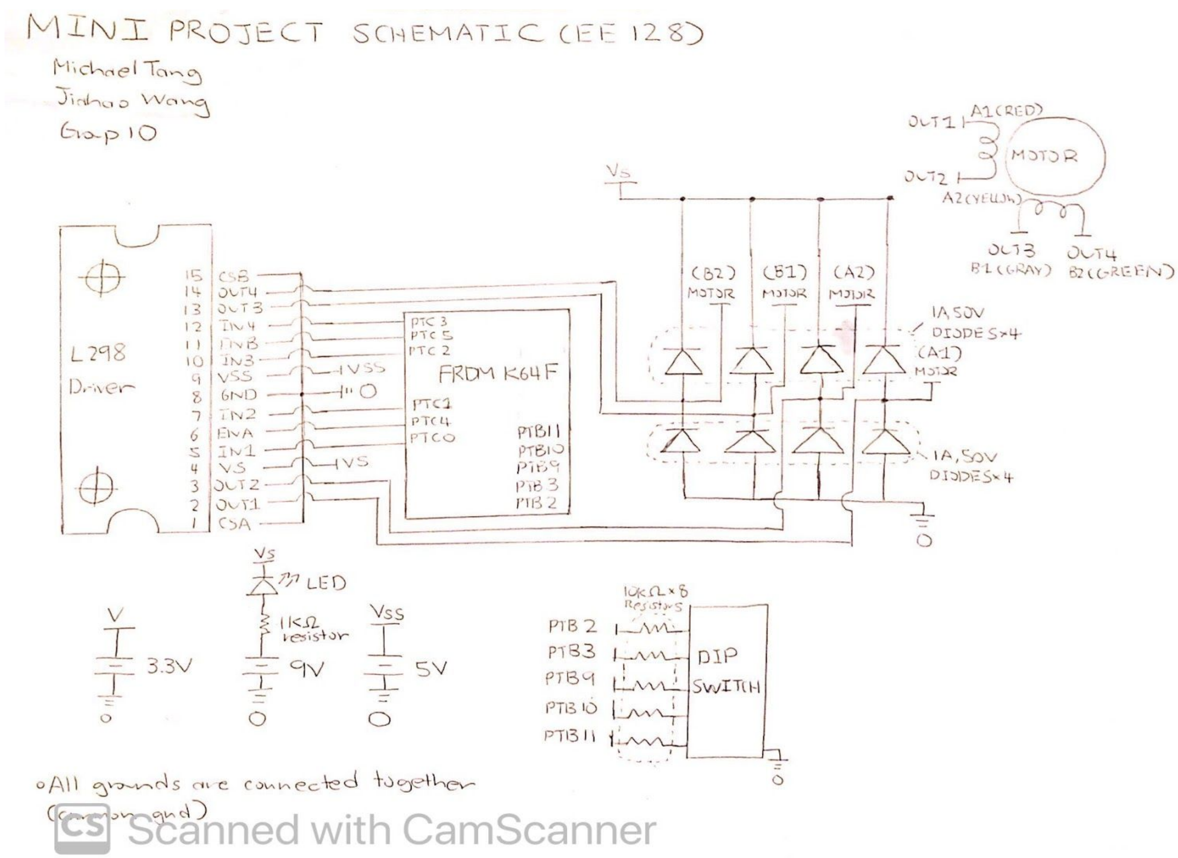


- Breadboard Circuit:



Implementation Details

- Schematic:



- Key Portion of Code:

```
#include "fsl_device_registers.h"                    //.h file for FRDM commands
#include <stdint.h>                                  //.h file for ADC commands

unsigned int a, k;
unsigned long delay;                                 //Variables for motor delays
unsigned long i;                                     //Variable for Switch and Motor delay
unsigned long SW_1=0x00;                             //Variable for Switch 1
unsigned long SW_2=0x00;                             //Variable for Switch 2
unsigned long SW_3=0x00;                             //Variable for Switch 3
unsigned long SW_4=0x00;                             //Variable for Switch 4
unsigned long SW_5=0x00;                             //Variable for Switch 5

void software_delay(unsigned long delay)             /*Function for motor delay*/
{
    while (delay > 0) delay--;                       //Decrement delay until delay is no longer over 0
}

void main(void)
{
    SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;               /*Enable Port A Clock Gate Control*/
    SIM_SCGC5 |= SIM_SCGC5_PORTB_MASK;               /*Enable Port B Clock Gate Control*/
    SIM_SCGC5 |= SIM_SCGC5_PORTC_MASK;               /*Enable Port C Clock Gate Control*/
    SIM_SCGC5 |= SIM_SCGC5_PORTE_MASK;               /*Enable Port E Clock Gate Control*/

    /*GPIO configurations*/
    PORTA_PCR4  = 0x100;                     //NMI to GPIO

    PORTB_PCR2  = 0x100;                     //Configures PB2 for GPIO
    PORTB_PCR3  = 0x100;                     //Configures PB3 for GPIO
    PORTB_PCR9  = 0x100;                     //Configures PB9 for GPIO
    PORTB_PCR10 = 0x100;                     //Configures PB10 for GPIO
    PORTB_PCR11 = 0x100;                     //Configures PB11 for GPIO
    PORTB_PCR21 = 0x100;                     //Blue LED configured as GPIO
    PORTB_PCR22 = 0x100;                     //Red LED configured as GPIO
```

GPIO Configs

```
for (;;)
{
    /*Switches to Pins Configuration*/
    uint32_t  input = 0;                            //Unique Integer for DIP switch
    input = GPIOB_PDIR & 0xFFFFC;                   //Reads and stores pins 2 to 20 on PORT B
    //for (i = 0; i < 100000; i++);                 //S/W Delay

    SW_1 = input & 0x4;                             //Store the pin 2 of the Port B
    SW_2 = input & 0x8;                             //Store the pin 3 of the Port B
    SW_3 = input & 0x200;                           //Store the pin 9 of Port B
    SW_4 = input & 0x400;                           //store the pin 10 of port b
    SW_5 = input & 0x800;                           //store the pin 11 of port B

    /*Rotation Speed*/
    delay = 7000;                                   //Approx. 7.88 degrees/second speed

    /*Default Starting Lock*/
    /*None of the switches are pressed and door lock should not move!*/
    /*To ease confusion: 0 is pressed, not 0 is not pressed (i.e. 1)*/
    if((SW_1 != 0) && (SW_2 != 0) && (SW_3 != 0) && (SW_4 != 0))
    {
        if(SW_5 == 0) /*Reset Mode*/
        {
            GPIOB_PDOR = (1 << 22);                 //Turn on blue LED
            software_delay(delay);                  //Wait Delay
            a = 0;                                  //Set a to 0
              for(a = 0; a <= 10; a++)              //Secure the lock back to 'lock' position(clockwise)
              {
                GPIOC_PDOR = 0x3A;                  //0011 1010 ENA/ENB and PTC1 and PTC3
                software_delay(delay);              //Delay(speed)
                GPIOC_PDOR = 0x39;                  //0011 1001 ENA/ENB and PTC0 and PTC3
                software_delay(delay);              //Delay(speed)
                GPIOC_PDOR = 0x35;                  //0011 0101 ENA/ENB and PTC0 and PTC2
                software_delay(delay);              //Delay(speed)
                GPIOC_PDOR = 0x36;                  //0011 0110 ENA/ENB and PTC1 and PTC2
                software_delay(delay);              //Delay(speed)
              }
            a = 0;                                  //Set a back to 0 for repeated performance
            k = 0;                                  //Set k back to 0 for repeated performance
            GPIOB_PDOR = (1 << 21) | (1 << 22);     //Turn off blue LED
            software_delay(delay);                  //Wait Delay
```

Reset Lock Function

Testing/Evaluation

- **9 Volt Battery Test**
    - Testing Equipment:
        - A single LED component
        - A single resistor
    - Required Equipment:
        - 9 Volt Battery
        - Breadboard
    - Testing Environment:
        - An open table
        - No open liquids nearby
        - Room temperature with normal humidity
    - Test Procedure/Scenario:
        1. Place single LED and resistor on breadboard
            a. Positive LED end connects to first resistor end
            b. Negative LED end connects to ground rail
            c. Second resistor end connects to power rail
            d. 9 Volt Battery wires onto respective power and ground rails
        2. Switch on 9 Volt Battery
        3. Test PASSES if LED lights up when 9 Volt Battery is switched on.

4. Test FAILS if LED does not light up when 9 Volt Battery is switched on.

- **4x4 Matrix Keypad Test**
  - Testing Equipment:
    - A single arduino genuino board
    - A personal computer with the arduino program
  - Required Equipment:
    - 4x4 Matrix Keypad
  - Testing Environment:
    - Arduino computer program
  - Test Procedure/Scenario:
    1. Connect 4x4 Matrix Keypad pins into arduino digital pins 3 to 10.
    2. Use the arduino code and install the needed keypad.ino library inside the arduino program.
    3. Connect arduino to the necessary personal computer port and have the port be available on the arduino program.
    4. Test PASSES if the arduino serial monitor displays the correct keys pressed.
    5. Test FAILS if the arduino serial monitor does not display the correct keys pressed.

- **Driver and Motor Test**
  - Testing Equipment:
    - A personal computer with the Kinetis Design Studio program
  - Required Equipment:
    - FRDM K64F board
    - Motor Driver
    - Motor
    - DIP Switch
    - 8 Diodes
    - 1 Resistor
  - Testing Environment:
    - Arduino computer program
  - Test Procedure/Scenario:
    1. Follow all the steps of the EE 128 Lab 5 Manual
    2. Test PASSES if the motor runs clockwise and counterclockwise when switch is used.
    3. Test FAILS if the motor does not run correctly or at all when switch is used.

Discussions

- Challenges
  - Implementation of the 4x4 matrix keypad and the arduino with the FRDM K64F.
  -
- Limitations
  - Sole use of the FRDM K64F
  - Less than two weeks of project development (not including preparations of other finals)
- Possible Improvements
  - Enable the use of the 4x4 matrix keypad for more complex passwords, ignore the need to reset the switch positions, and simpler mode changes.
  - Add an LCD screen or equivalent for password display and menu display.

Roles and Responsibilities of Group Members

- Michael:
  - Create the breadboard schematic.
  - Create the block diagram.
  - Create the physical demo circuit.
  - Perform the demonstration.
  - Perform and create the in-class presentation.
- Jiahao:
  - Create the main.c code.
  - Create the flowchart.
  - Plan the demonstration.
  - Perform the in-class presentation.

References

- µTasker – Keypad User's Guide,including Port Simulation
- 4x4 matrix keypad datasheet pdf
- 4x4 matrix keypad implementation on Arduino board
- EE128 Labs 2, 3, and 5 manuals
- EE128 Lecture Notes

Conclusion

In this mini-project we designed the circuit and implemented the stepper motor and the DIP switch, the lock is driven by the motor and it can be automatically locked after a few seconds(while unlocked). The project idea is carefully discussed by Michael and Jiahao as well as the details, most of the expected functions are implemented successfully. Michael created the schematic and finished all the hardware design parts including the physical lock model. Jiahao created the flowchart and created the code with Michael. Michael created the demo circuit and made a successful demo.