Nicholas Maresco

Anderson Vanegas

EC450

Final Project*: Report*

**Misson:**

The goal of this project was to design a door unlocking system. The system receives user input from a number pad, which provides the user with an simplistic interface to enter a password. If the password is correct, the door will unlock allowing the user to enter. We aim to create an intuitive device that would be easy to use while adding a simpler way to unlock your dorm room.

**Components and Materials:**

*Sender Circuit*

* MSP430G225 Integrated Circuit
* 415 kHz RF Transmitter
* 8 Pin Number Pad 4x4 keypad matrix
* 16 Pin LCD Screen (4 Pin Mode)
* 10K Ohm Potentiometer
* Push Buttons: Reset, Force Signal Transmission
* 3V battery
* 5V battery - LCD
* LED: Signal Transmission
* 3D Case (and sub components)
* Active Low Resistor 10M Ohm
* 330 Ohm BL Resistor

*Receiver Circuit*

* 415 kHz RF Receiver
* IRF520 Y47K AJ - Power MOSFET
* DC Motor
* 1008N4007 - Diode
* 6V battery
* 3D - Dock Unlocking Arm
* 3D - Arm Slider
* 3D - Motor Gear Head

**Design & Description:**

The number pad and LCD screen will be placed outside of the door for the user to interact with. The LCD screen prompts the user to enter the correct password, which is hardwired into the chip (cryptographic integrity was not our focus in this prototype). Both are connected to a breadboard, which the MSP430 is also connected to. If an incorrect password is entered, the MSP430 will not send out a data signal to activate the unlocking-motor.

The number pad uses all of pins 1.0 to 1.7 except for pin 1.3 which is used to send data. P1.3 was not used as we would not have enough GPIO for a 4x4 matrix number pad and still be able to send data. The schematic will make the reasoning for P1.3 not being used more apparent. As a result, we went with a 3x4 matrix in order to leave us with one free pin. P1.0 is connected to column 1, P1.1 is connected to column 2 and P1.2 is connected to column 3. P1.4 to P1.7 are connected to Row 1 to Row 4, respectively.

As the user enters number combinations, the LCD screen will update with their attempts, and clear their failed attempts.

However, when the correct password is entered, the MSP430 will output a signal on P1.3 to the receive circuit. The data is sent through an RF sender to an RF receiver on another board both of which are operating at 415 KHz. The receiver module then sends a high to the gate of the power MOSFET operating as a switch to the DC motor. Ideally this would then be used to unlock the door.

**Schematics:** (See Attached)

* Number Pad Circuit
* Receiver Circuit
* Sender Circuit
* LCD Circuit
* Master Circuit

**Assessment:**

Looking at what we have done so far, our team agrees that our project is a success. We were able to configure the number pad and LCD to work, all with only using one MSP430. We were also able to successfully send and receive signals in order to activate the motors.

The only place where we were having issues would be on the mechanical side of the project. The motor takes 9 volts to run at its most efficient rate. At the time we were only able to provide 6 volts. Even if we were able to supply the motor with the voltage required, we do not think it would be strong enough to ultimately open a door. We would need to find another workaround by either acquiring a stronger motor with a voltage we can supply, or modify the gear configuration the motor uses in order to create an optimized gear ratio in order to amplify the torque.

**Future Expansion:**

In the future if we were to do this project again or find some ways to improve on it we would focus on the following points:

*Reconfigure Motor Circuit*

Currently, our circuit for controlling the motor does not work all the of the time. The signal is not transmitted reliably enough to be of practical use. It is possible that there is too much interference or noise on the receiving side. So our first focus would be to improve the reliability of the entire system, and the consistency of transmitting the signal. The receiver circuit also behaves suboptimally as once the signal transmission ends (indicating the motor should stop), the motor continues to work despite not intended.

*Motor and Mechanical Components*

Another issue that we have at the moment is the motor is not strong enough to move the lock on the door. We would need a bigger motor which requires more voltage to power. Another idea we thought could also work would be to acquire a fast motor and figure out a gear ratio that can work off of this motor in order to generate enough torque.

*One Pin Serial Communication Number Pad & Shift Register*

This is one area we know we could improve upon for the future. At the moment, we use a rather large amount of pins. A total of seven pins are dedicated just to pinpoint which key is pressed on the number pad. We believe that by using a shift register we could loop through our row and column arrays to pinpoint which one was pressed, all while only using one pin. This would free up six other pins to add more features, or maybe even just to make the wiring seem simple and cleaner.

**Code:** (See Attached)

* Main.c
* LcdLib.c
* LcdLib.h

**Code Description:**

For our code , we implemented a MSP430 LCD library originally written by Elliot Gurrola and later modified by Luiz Carlos Bauelos-Chacon and Elias N Jaquez.

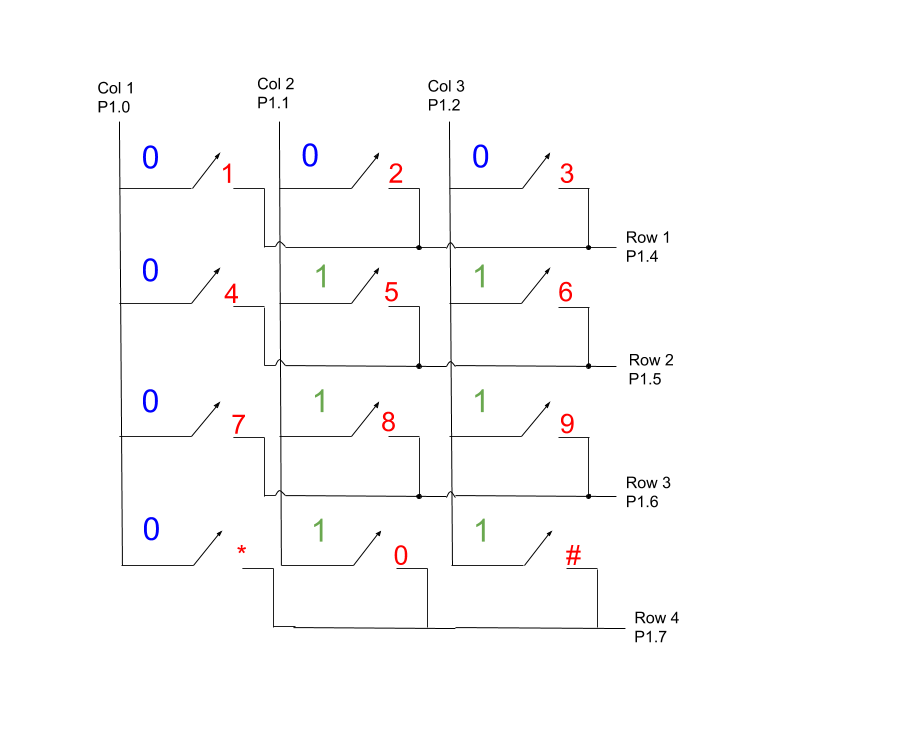
Their code sends data and commands to the LCD using 4 pins, which means they send information a nibble at a time. One of the functions that is used is lcdSetText(). What this function does it takes in the text that is going to be written to the LCD. It also takes in the x and y coordinates of where it is going to be written. It sets this location using lcdWriteCmd(). Then it uses the lcdWriteData() function to actually send the data we wish to display on screen nibble by nibble.

In our code, Main.c we start with variables that we will need, such as the values for the number pad and the correct password that will allow a signal to be sent. Then follows the initial main function where we initialize the LCD, setup the text in the LCD to prompt the user for the password, configure and enable the WDT, enable interrupts and low power mode, and setup the pins to read the signals from the number pad.

Several functions that were written for this project are: get\_key, pressed\_return, and clear\_password.

*get\_key()*

This function is what make the number pad function. Initially all of the key values are set to be high. The pins are set to be pulled high. What the function does is iterate through the columns. It then sets the entire column to be it is on at the moment to be low. It does this constantly until a button is pressed. When it is pressed, the entire row of the button is also set to low. Since we already know which column we are on, since it is low, we check against which row we are on. Only one row will now also be set to low. By knowing the row and column, we determine where in the matrix the button press is coming from and return the value to print to the LCD screen.



(Number Pad Button Press Example)

For example say button one is pressed. The columns are constantly being iterated. Immediately after one is pressed. The whole row it is on becomes low as well. What we are doing is !(pin\_mask & row). So for each column the masks for P1.4 to 1.7 are 0001 0000, 0010 0000, 0100 0000, 1000 0000. We & each with 1110 0000. For only Row 1 (P1.4) we will get a 0 which when paired with ! becomes a 1 alerting us that Column 1, Row 1, has been pressed.

*pressed\_return()*

this waits for a button to pressed. Once a button is pressed it resets itself to keep waiting for another input and returns the value of the number from our array of values.

*clear\_password()*

what this function does is start at the index on the LCD where the password shows up, it then loops and sets 4 spaces as blanks, allowing another guess to be made.

Finally we have the watchdog timer which wakes up every once in a while to check on the conditions of the LCD and buttons that have been pressed. We store the pressed value, and then print that value to the screen and reset out variables to await another press. We also have a condition in the watchdog timer in case the same button is pressed twice in a row. Finally, once the user enters 4 values, we check to see if the password is correct and we output a message of success. Otherwise we reset the password and they are allowed to try again.

**Contributions:**

*Anderson*

* Documentation
* Schematic Formalization
* Planning
* Design Framework

*Nicholas*

* Circuitry
* Number Pad Implementation
* 3D Designs
* User Interface