**Project 1:  
Hello World Electronic Lock**

**Cal Poly CPE 329-01**

**Spring 2017**

short line

Brendan Baronia

Melinda Ong

Professor Gerfen, Jeffrey B.

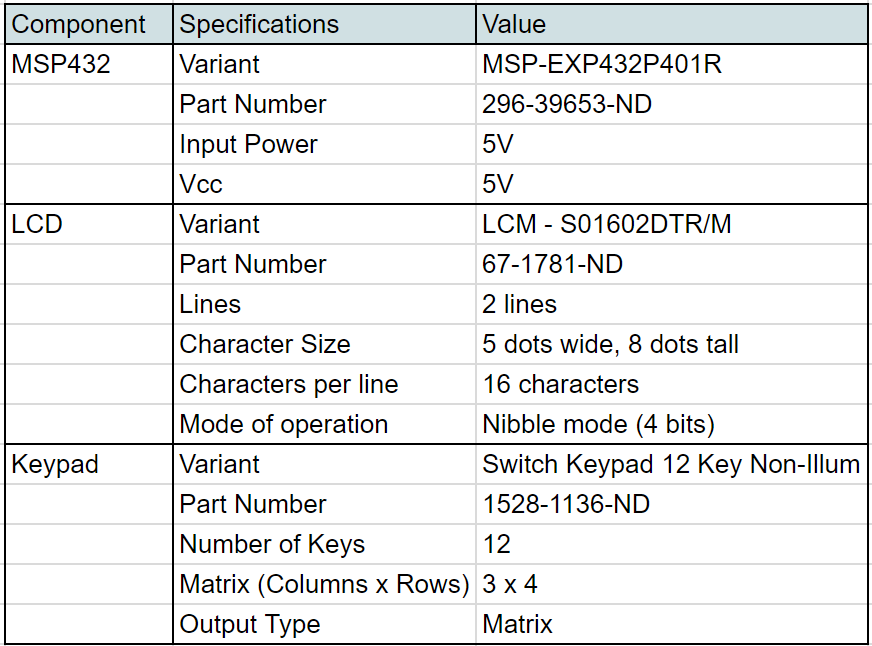
**April 17, 2017**

# Purpose: The purpose of this project was to integrate the LCD and the keypad to create an electronic lock. The system is interfaced to an LCD that will display whether the user entered a correct or an incorrect key code on the keypad.

# System Requirements:

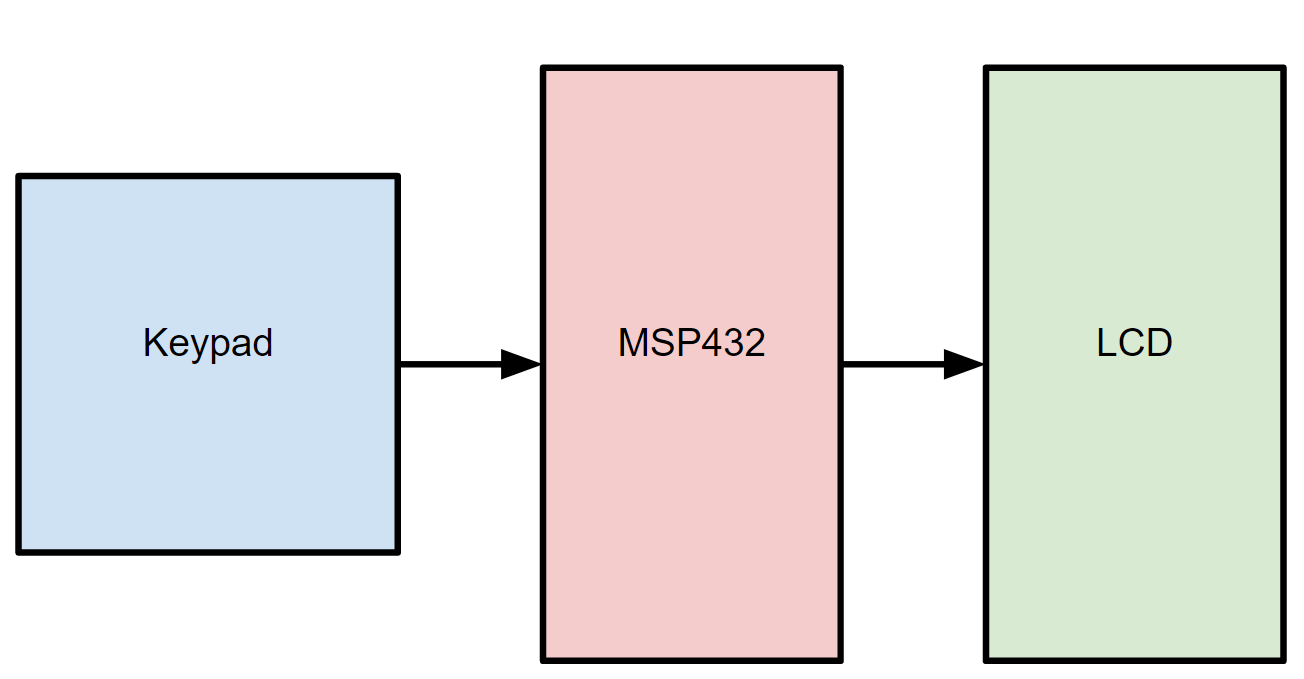
1. Shall display LOCKED ENTER KEY on an LCD screen until the user enters the correct 4 digit key
2. Shall display the pressed keys of the keypad on the bottom row after KEY
3. Shall display HELLO WORLD if the key was corrected
4. Shall display the same LOCKED ENTER KEY screen and wait for a new key sequence if incorrect
5. Shall clear if the \* key is pressed and display the LOCKED ENTER KEY screen until a new key sequence is entered

# System Specification:



**Table 1: System Specifications**

# System Architecture:

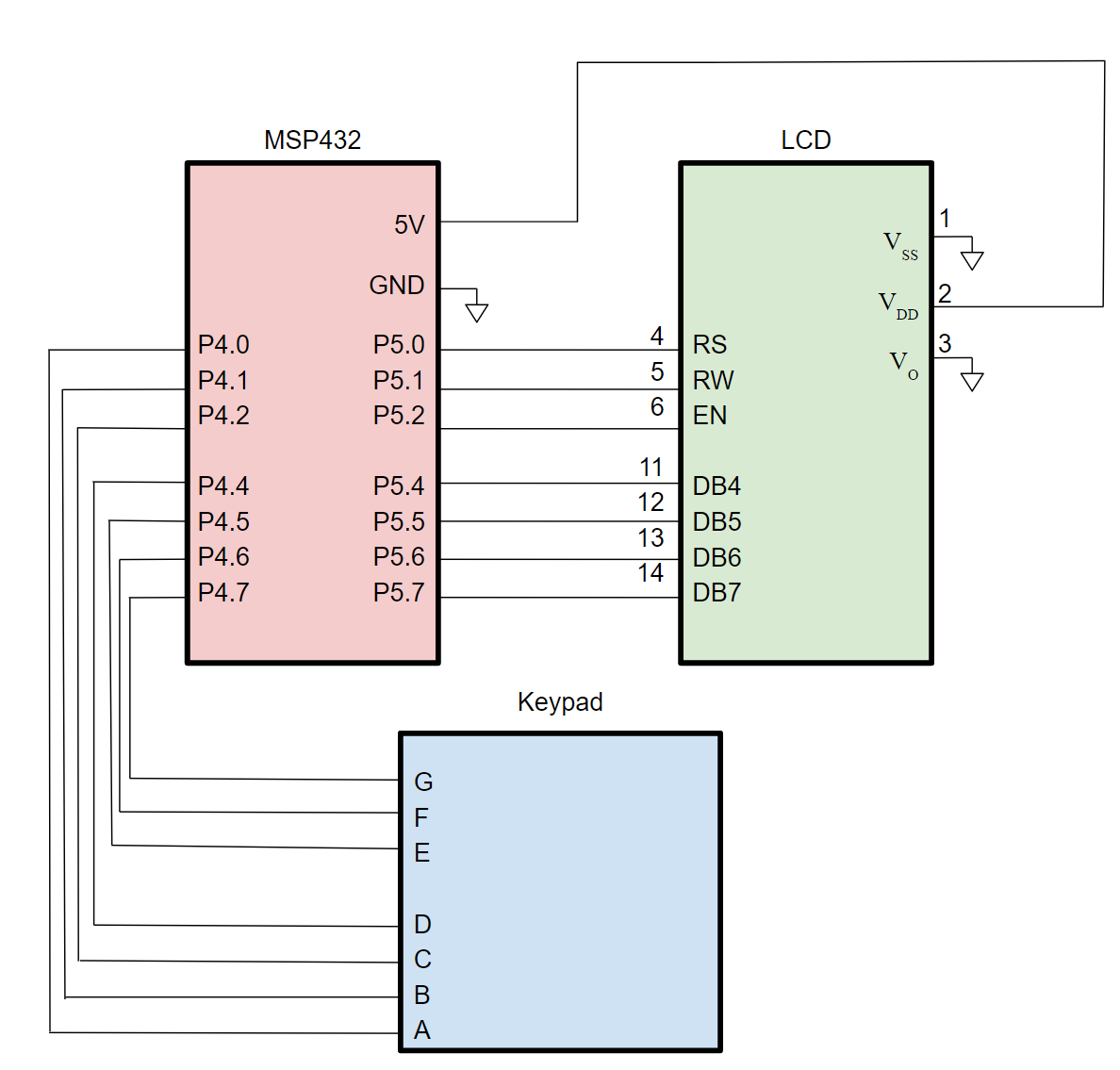


**Figure 1: Overall system block diagram**

****

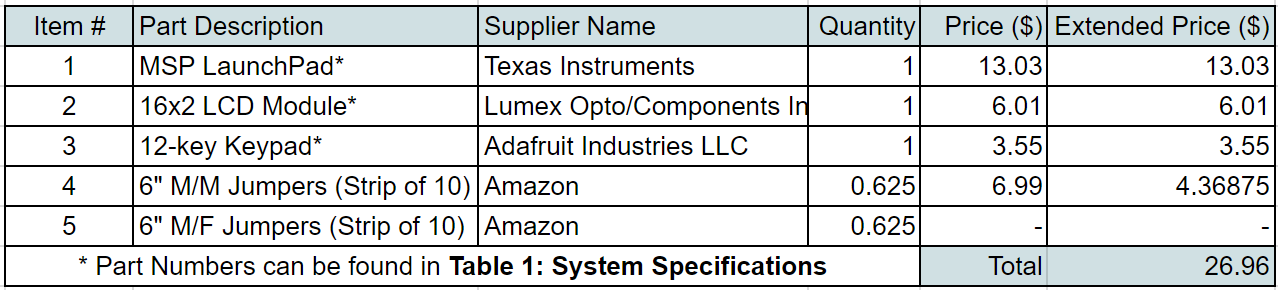
**Figure 2: Top-level Software flowchart**

# Component Design;



**Figure 3: Schematic Diagram of the MSP432, the LCD, and the keypad**

# Bill of Materials:



**Table 2: Bill of Materials**

# System Integration:

***a.) Development Process***

In order to build the system, the LCD utilized Port 5 and the keypad utilized Port 4 (so that the Source Code 3-5 calculations of the matrices did not need to be modified).

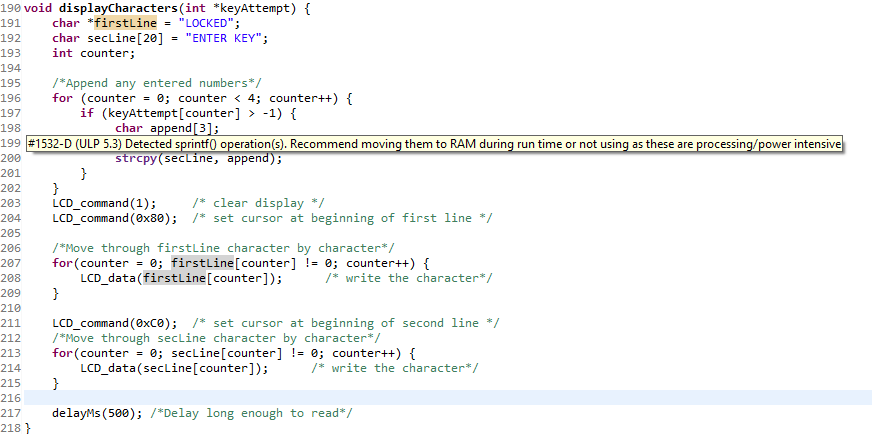
The LCD and keypad were first both tested separately with simple test code. The LCD was verified to print “Hello”, and the keypad was verified to light up an LED on the MSP432.

Once both systems were working successfully on their own, they were integrated using a simple program that used the LCD to display numbers pressed on the keypad.

The code for the combination lock program was developed independently of the hardware for unit testing. It was written in C and used the flowchart shown in Figure 2 as its design. Each step in the flowchart represents a separately tested function, with simple input used to transition between states.

Once the hardware and software were both verified to be working, the process moved to integrating them. This involved constructing code to interpret keypad input in a format the software could understand, and moving software output onto the screen.

***b.) Screenshots***

**

**Figure 4: CCS sprintf warning**

***c.) Bugs and Resolutions***

Significant bugs arose when moving the code onto the MSP432. Initial releases of the software used C provided methods such as strcpy and sprintf to format output in the displayCharacters function. These methods worked well on a powerful system, but when moved onto the board, there were issues with the strings output. These issues were hinted at by the CCS software, shown in Figure 4, allowing for quick debugging of the system. As a result, the code was refactored to avoid use of these methods, and instead used the system shown in the final C code release.

# Project Demonstration:

**https://www.youtube.com/watch?v=Fk0dHrHnIEA**

# Conclusion:

Project 1’s goal was to design an electronic lock using the LCD screen and the 12-key keypad. The system interfaced the MSP432 to take in the input from the keypad and display a message on the LCD indicating whether or not the user entered the correct code. To accomplish this, software needed to handle both keypad inputs and LCD outputs, as well as the underlying logic to create a locking system. This software had to interface timing constraints of the external hardware with the speed of the processor to create a cohesive product.

The project could be improved by adding an extra prompt for the user to know if their code was incorrect. It could also allow for reprogramming the lock code using only the hardware rather than changing software aspects.

The system could also be better designed for the real world by creating a custom PCB to house both the LCD and keypad without excess wiring. The system could be redesigned to run off a standard supply voltage to allow for battery power rather than USB power.

In addition, several improvements could be made to the software. The unlock key is currently stored in a standard character array, which is insecure and easy to access. A future implementation should contain security features to protect the password and prevent unauthorized access. The system also currently blocks waiting for input, and repeatedly probes the keypad for key presses. A future system could save power by using interrupts to read keypad actions and entering a low power mode between key presses.

# Appendices:

***a.) C Code***

/\*

\* Project 1 - Combination Lock

\*/

#include "msp.h"

#include "LCD.h"

#include "keypad.h"

/\*function declarations\*/

void checkKey(int \*keyAttempt, int \*key);

void manageInput(int \*keyAttempt, int \*key);

void resetKey(int \*keyAttempt);

void displayCharacters(int \*keyAttempt);

void displayHello();

int main() {

/\* Initialize LCD and keypad

\* Initialize keyAttempt

\* Set key to 1234

\*/

LCD\_init();

keypad\_init();

int keyAttempt[4] = {-1, -1, -1, -1};

int key[4] = {1, 2, 3, 4};

/\*Print first display\*/

displayCharacters(keyAttempt);

while (1) {

/\*Manage inputs infinitely\*/

manageInput(keyAttempt, key);

}

}

/\*

\* Checks the key attempt against the valid key

\* calls displayHello if good

\* otherwise calls resetKey and displayCharacters

\*/

void checkKey(int \*keyAttempt, int \*key) {

char good = 1;

int counter;

/\*check for correct key\*/

for (counter = 0; counter < 4; counter++) {

if (keyAttempt[counter] != key[counter])

good = 0;

}

/\*if correct, show hello, otherwise reset\*/

if (good) {

displayHello();

}

else {

/\*reset key and display empty attempt\*/

resetKey(keyAttempt);

displayCharacters(keyAttempt);

}

}

/\*

\* Manages input from the keypad

\* if \* pressed, calls resetKey

\* Otherwise, reads in number and adds to attempt

\* Calls displayCharacters

\* If attempt is full, calls checkKey

\*/

void manageInput(int \*keyAttempt, int \*key) {

/\*Get input from keypad\*/

char input = 0;

while (input == 0)

input = keypad\_getkey();

/\*Check for 10 (\* on keypad)\*/

if (input == 10) {

resetKey(keyAttempt);

displayCharacters(keyAttempt);

}

/\*Check for 1-9 or 11 (1-9 and 0 on keypad)\*/

else if (input >=1 && input <= 9 || input == 11) {

int position = 0;

if (input == 11)

input = 0;

/\*find first empty spot\*/

while (keyAttempt[position] > -1 && position < 3)

position++;

/\*add number to attempt\*/

if (position < 4)

keyAttempt[position] = input;

/\*Update display to show new character\*/

displayCharacters(keyAttempt);

/\*check for full key attempt to pass to checker\*/

if (position == 3)

checkKey(keyAttempt, key);

}

}

/\*

\* Resets keyAttempt to all -1 (invalid)

\*/

void resetKey(int \*keyAttempt) {

int counter;

/\*reset attempt to all -1\*/

for (counter = 0; counter < 4; counter++) {

keyAttempt[counter] = -1;

}

}

/\*

\* displays the following

\* LOCKED

\* ENTER KEY xxxx

\* Where xxxx is the current attempt, up to 4 numbers

\*/

void displayCharacters(int \*keyAttempt) {

char \*firstLine = "LOCKED";

char \*secLine = "ENTER KEY";

int counter;

LCD\_command(1); /\* clear display \*/

LCD\_command(0x80); /\* set cursor at beginning of first line \*/

/\*Move through firstLine character by character\*/

for(counter = 0; firstLine[counter] != 0; counter++) {

LCD\_data(firstLine[counter]); /\* write the character\*/

}

LCD\_command(0xC0); /\* set cursor at beginning of second line \*/

/\*Move through secLine character by character\*/

for(counter = 0; secLine[counter] != 0; counter++) {

LCD\_data(secLine[counter]); /\* write the character\*/

}

/\*Write any entered numbers\*/

LCD\_data(' ');

for (counter = 0; counter < 4; counter++) {

if (keyAttempt[counter] > -1) {

LCD\_data(keyAttempt[counter] + '0');

}

}

delayMs(500); /\*Delay long enough to read\*/

}

/\*

\* Displays the following:

\* HELLO WORLD

\*/

void displayHello() {

char \*line = "HELLO WORLD";

int counter;

LCD\_command(1); /\* clear display \*/

LCD\_command(0x80); /\* set cursor at beginning of first line \*/

/\*Move through line character by character\*/

for(counter = 0; line[counter] != 0; counter++) {

LCD\_data(line[counter]); /\* write the character\*/

}

}

***b.) References***

LCD Datasheet: *PmodCLPTM Parallel LCD Display Module Reference Manual*; Digilent Inc.; April, 28, 2008.

LCD Pin Descriptions: *TI MSP432 ARM Programming for Embedded Systems*; Mazidi and Naimi ARM series, 2016, (pgs. 57-58).

KeypadPinout: Provided by Professor Gerfen via PolyLearn

Bill of Materials: Provided by Professor Gerfen via PolyLearn shared through Cal Poly IEEE