Shan Liang
David Tran
Biometric Lock Box Group
EC450
Final Project

The goal of our project was to create a lockbox with authentication and display features. The project was intended to be similar to an electric safe with similar authentication features of an iPhone 5s. These features include a biometric fingerprint scanner and a number pad. The design of the project was comprised of a wooden box that will have the fingerprint scanner, keypad, and screen all be placed on the front cover of the box. A servomotor was used as the lock of the box and locks and unlocks the box when the user enters the correct authentication. The user is able to enroll multiple fingerprints as well as add and change a 4-digit password on the keypad. The screen displays feedback for successful and unsuccessful authentication on the keypad and fingerprint scanner. To control the lockboxes features, a simple user interface was designed. The lock box has designated inputs on the keypad for certain commands such as enrolling a fingerprint, changing a password, and fingerprint and password authentication. The user interface includes visual feedback that is displayed on the screen. All of these components are driven by two MSP430s and a rechargeable battery pack is used as a power source.

Implementation: System Level

The system of our lock box consists of two MSP430s. We use two MSP430s due the need for more GPIO pins for all of our components. One of the MSP430s is used to control the display as well as the keypad. Because the keypad is used to control our user interface, this will be referred to as the ‘master’ MSP430. The master controls

the commands for entering successful 4-digit passwords, choosing when to enroll or authenticate using the fingerprint scanner and also displaying data on the screen for the user to see. The 'slave' MSP430 is used to communicate with the fingerprint scanner as well as the servomotor lock.

To have the master talk to the slave, we used 4 GPIO pins from each MSP430 for the communication channel between the two MSP430s. The master and slave both have 2 output pins and 2 input pins. The outputs of each MSP430 is connected to the inputs of the other. We then can assign commands to each possible 2 bit word on each MSP430, giving us a total of 4 (2^2) possible commands for each. The MSP430s receive messages via GPIO interrupts. Each transmission triggers the interrupt, and the messages are read based on which pins are high and low.

In our implementation, the master has the possibility to send 3 commands to the slave: Lock/unlock the box via password, enroll a fingerprint, and authenticate a fingerprint. The slave has the possibility of sending 2 messages to the master: Success and Failure. When a user enters the correct 4-digit pin on the keypad, the master will send the slave a lock/unlock request and the slave will control the servomotor to unlock or lock the box and send back a success message. When the user wants to add an additional a fingerprint, the master will send an enroll request. The slave will communicate with the fingerprint scanner to authenticate and will send a success or failure back to the master depending on if the enrollment was successful or not. When the user wants to authenticate using their fingerprint, the master will send an authentication request. If the user authenticates their finger correctly, the slave will send a success command and control the servomotor to

unlock or lock the box depending on if it is locked or not. If the fingerprint is not accepted, nothing will change and a failure is sent back to the master. Whenever the master receives a success or failure signal from the slave, the master displays that data on the screen. To know when the fingerprint scanner is on, we programmed commands to turn the built in LED in the fingerprint scanner on and off depending on the situation.

Component Level: Fingerprint Scanner

The fingerprint scanner that we used was the GT-511C1R. This scanner has its own onboard processor as well as flash storage. It handles all the processing for the fingerprints. To communicate with the fingerprint scanner, you need to send command packets using UART protocol. The fingerprint scanner has 4 pins: Vcc, GND, TX (UART) and RX (UART). For our project, we created case statements on the MSP430 for the fingerprint scanner. Each state consisted of a unique command on the scanner: enrolling, identifying, verifying or removing a fingerprint. There were additional states for the LED on the scanner being on or off, IDLE mode when the scanner wasn't being used and checking if a finger is on the scanner. The states are changed according to the responses of the fingerprint scanner and the master MSP430. The states are all controlled in the watchdog timer.

A build command packet function was designed to create the 12-byte UART data packet that had the correct header, ID, and checksum to be prepared to send. The arguments passed to the function were the 2-byte command and the 2-byte parameter (if the command needs more data). This packet is stored in a 12-byte TX array. Once the packet is built, we then enable the UART TX and RX interrupt and

send the 12 bytes to the Fingerprint scanner. The scanner will then reply with a 15-byte packet response. From there, the MSP430 will store that response in a RX array and will read the response and handle it from there. This function is in every state. The arguments are different depending on the states command.

Component Level: Servomotor

The servo that was used for our project was the Hitec 21055S HS-55 Economy Sub Micro Universal Servo. This servo has a 180-degree angle of rotation and has a pulse width of 600 microseconds to 2400 microseconds. The servo has 3 pins, Vcc, GND, and PWM. The servo needed to be powered using 5v instead of 3v so we connected Vcc to TP1, which outputs 5V. To control this servo on the MSP430, we set Timer A1 to OUTMOD_7, for RESET and SET on TA1CCR1 and TA1CCR0 respectively. We set TA1CCR0 to about 20,000 microseconds (SET to high) and connected the output of Timer A1 to a GPIO (P2.2) associated with that timer. From there to move the servo, we would set the TA1CCR1 on Timer A1 to 2000 microseconds to unlock and 750 microseconds to lock (RESET to low). These PWM values are within the pulse width described earlier.

Some of the next steps we might take to make this project more successful would be buying strong materials for the lock and adding more features to the lock box such as controlling it using an application on a phone through Bluetooth. We could also add more implementations to personalize the user interface.

Component Level: LCD Display

The LCD Display used was a Sparkfun Serial Enabled 16x2 Black on Green LCD 3.3V. The LCD Display was written to via the UART TX pin of an MSP430. The

MSP430 would output to the display based on its state machine, which would change based on the keypad inputs and received information from the Servo/Scanner MSP430.

We wrote to the display one line at a time, or 16 characters at a time since it is a 2 line 16 character display. A function was written to simplify this process by just giving the function the string literal and the number of characters in the string literal. We would store the string in a character array and have another variable keep track of the number of characters given to the function. Then we made use of the USCI TXBUF interrupt to iterate through the array and write to the LCD character by character.

Component Level: Matrix Keypad

The Matrix Keypad was an Adafruit 3x4 Phone-style Matrix Keypad. The keypad completely consumes the Port 2 GPIO pins and P1.7 on one MSP430. This is because 7 pins are required for the MSP430, 3 for the columns, and 4 for the rows. Because the Matrix Keypad connects a column with a row on any keypress, we would need to check both the columns and rows at any time in order to decide on which button was pressed.

In our implementation of the keypad, we had the 3 column pins act as high inputs and the 4 rows as low outputs. The Port 2 interrupt would be enabled for the columns, and once an interrupt was detected we would raise all of the rows to high except for one at any given time. In doing that, we would be able to see which row was attached to the column because the column would be low only if its connecting

row is low. By doing a linear search through the rows, we could find out the key pressed before the user lifts their finger off of the keypad and breaks the connection.

Assessment of Success

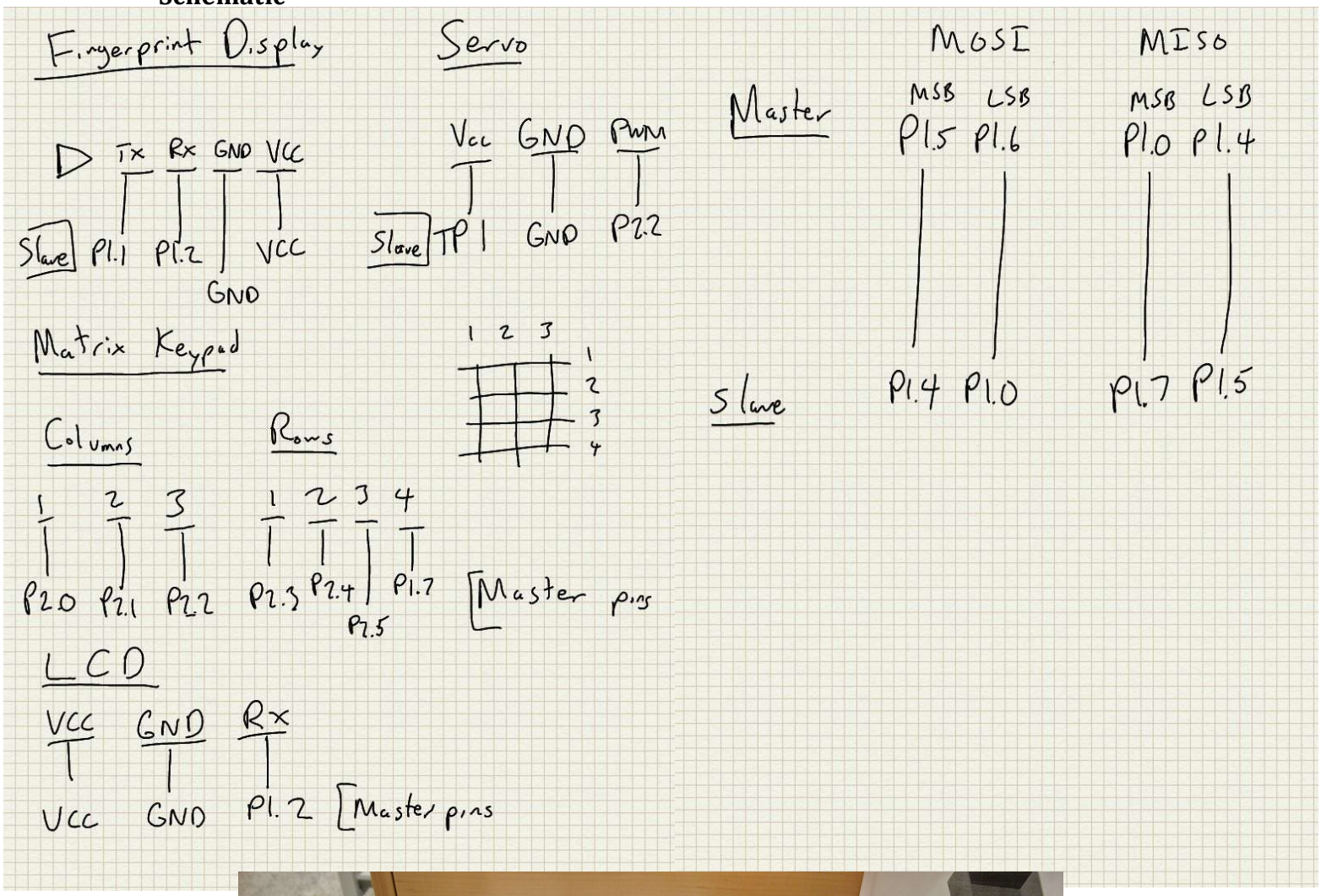
While our project diverged from our original intentions, we believe that our final product was a complete success. The change in original goals came from time constraints and prioritization of more valuable core goals. Our core goals consisted of maintaining visual feedback for the user on the LCD Display, adding a user's fingerprint to memory, enabling users to lock/unlock the box via the fingerprint scanner, enabling users to lock/unlock the box via the keypad, and allowing the user to change passwords. We were able to fulfill all of these goals without fail, making the box reliable and easy to use, while still being relatively secure and fulfilling its intended purpose.

The time constraints came from creating the box, programming two MSP430's to interact with the 4 peripherals we had (the Matrix Keypad, LCD Display, Fingerprint Scanner, and Servo lock), and also programming the two MSP430's to work together. Two MSP430's were required due to the fact that we didn't have enough USCI or GPIO pins on one MSP430.

Summary of Team Contributions

Shan worked on the slave MSP430, which contained the fingerprint scanner logic and the servo logic. David worked on the master MSP430, which contained the keypad and display logic. Shan and David both worked together on building and designing the parts for the box in and putting all the components together to fit the box.

Schematic


Fingerprint Display
Tx Rx GND VCC
Slave P1.1 P1.2 GND VCC
Servo
Vcc GND PWM
Slave TP1 GND P2.2
Matrix Keypad
1 2 3
1 2 3 4
Columns
1 2 3
P2.0 P2.1 P2.2
Rows
1 2 3 4
P2.3 P2.4 P2.5 P1.7
[Master pins
LCD
VCC GND Rx
VCC GND P1.2 [Master pins
Master
MOSI
MSB LSB
P1.5 P1.6
MISO
MSB LSB
P1.0 P1.4
Slave
P1.4 P1.0
P1.7 P1.5