ECE492 FINAL REPORT

BIOMETRIC HOME ENTRY SYSTEM

BIOLOCK

GROUP #9

Mavis Chan **maviskay@ualberta.ca**

Brent Erickson **be@ualberta.ca**

Sydney Bitner **bitner@ualberta.ca**

System that utilizes a fingerprint sensor to authenticate

a user before entry into private property is permitted

ABSTRACT

The goal of this project is to create a fingerprint enabled lock designed for home entry control. This document outlines the concepts, implementation, and required hardware/software in order to successfully create this project. The design is implemented on an Altera DE2 Cyclone II FPGA board along with several external peripherals. A user will scan their fingerprint on an optical sensor. If the fingerprint is enrolled and the user has scheduled access at that time, the locking mechanism will be opened. If the user does not have access, an alarm will be sounded. In both cases, the action will be logged to a database stored on an SD card. An Android application was also created that communicates with the DE2 via a web server. This allows the owner to view and manage users, roles and scheduling in the database. The owner is also able to remotely unlock the door from the Android application. In addition, the DE2 has a camera module attached which allows the current person at the door to be viewed via the Android application. The design of a fingerprint enabled home entry system was successfully implemented, meeting all functional requirements with the exception of saving images on the SD card. Future possible implementations have been listed in the appendix.

TABLE OF CONTENTS

FUNCTIONAL REQUIREMENTS

The main functionality of this system is controlling user access to a property through the use of biometrics. Authorization of a user or administrator is detected through a fingerprint sensor. Through the use of an Android application, the owner or admin will have the capability of modifying user privileges, unlocking the door remotely or viewing a snapshot of the person at the door, and users will have the capability of modifying personal preferences.

OPERATION - ACCESS

Initially, the fingerprint sensor will continually poll for prints and the verification process will begin when a print is detected. Access to the property is granted once the user is verified.

1. Fingerprint sensor detects user's fingerprint.
2. Fingerprint data is transferred to board.
3. Board will interface with database and compare fingerprint with a list of authorized prints.
4. After receiving result from database, board will either:
    a. Grant access: Display authorization & unlock the lock.
    b. Deny access: Display unauthorization & lock remains locked.
        i. Sound alarm.
5. Result is logged.

OPERATION - MODIFICATIONS

With the use of an Android application, admins can add additional users to the system and users can modify their preferences. In addition, admins can modify user privileges such as their roles, which dictate the timeframe of allowable access. Admins are also able to view a log of entry attempts, view a snapshot of person at the door, and unlock the door remotely.

1. User accesses Android application.
2. Application will notify the board of user ID and board will either:
    a. Detect admin and allow actions on application for:
        i. Addition of users.
        ii. Modification of preferences.
        iii. Scheduling timeframe of allowable access for different roles.
        iv. Viewing of entry log.
        v. Viewing snapshot of person at door.
        vi. Unlocking door.
    b. Detect normal users:
        i. Inform application to send request for scan on fingerprint sensor.
        ii. Fingerprint sensor detects user's fingerprint and transfers data to board.
        iii. Board will interface with database and compare fingerprint with list of authorized prints.
        iv. After receiving result from database, board will either:
            1. Display authorization & notify application to allow modification of preferences.
            2. Display unauthorization & notify application to do nothing.

For the design of this system, all functional requirements were met with the exception of a few implementations. Initially, the plan was to include logging of the guests at the door at the occurrence of an attempted entry failure. Due to the speed of the SD card, this feature was not implemented. In addition, the goal was to allow for multiple

attempts followed by a successful attempt within a timespan before an alarm was sound. Due to time constraints, this feature could not be implemented. The design of this system would be considered successfully with the exception of performance. As writing to the SD card was slow, it affected the performance of the system drastically. Time was needed for modifications to occur in the database before the system could respond to any other requests.

DESIGN & OPERATION

HARDWARE - DE2 COMPONENTS

This system utilizes the NIOS II/f processor on the Altera DE2 FPGA board. Since reading and writing to the SD card is quite slow, the fast processor core is needed to provide our system with acceptable performance. A 27MHz and 50 MHz clock is used for the processor, as well as 8 MB SDRAM, 512 KB SRAM, and 4 MB Flash. A SD card is also implemented to hold the database. For testing purposes, the LEDs, LCD display, switches, and buttons are added to the system. Figure 1 shows the hardware components and external peripherals used for the system.

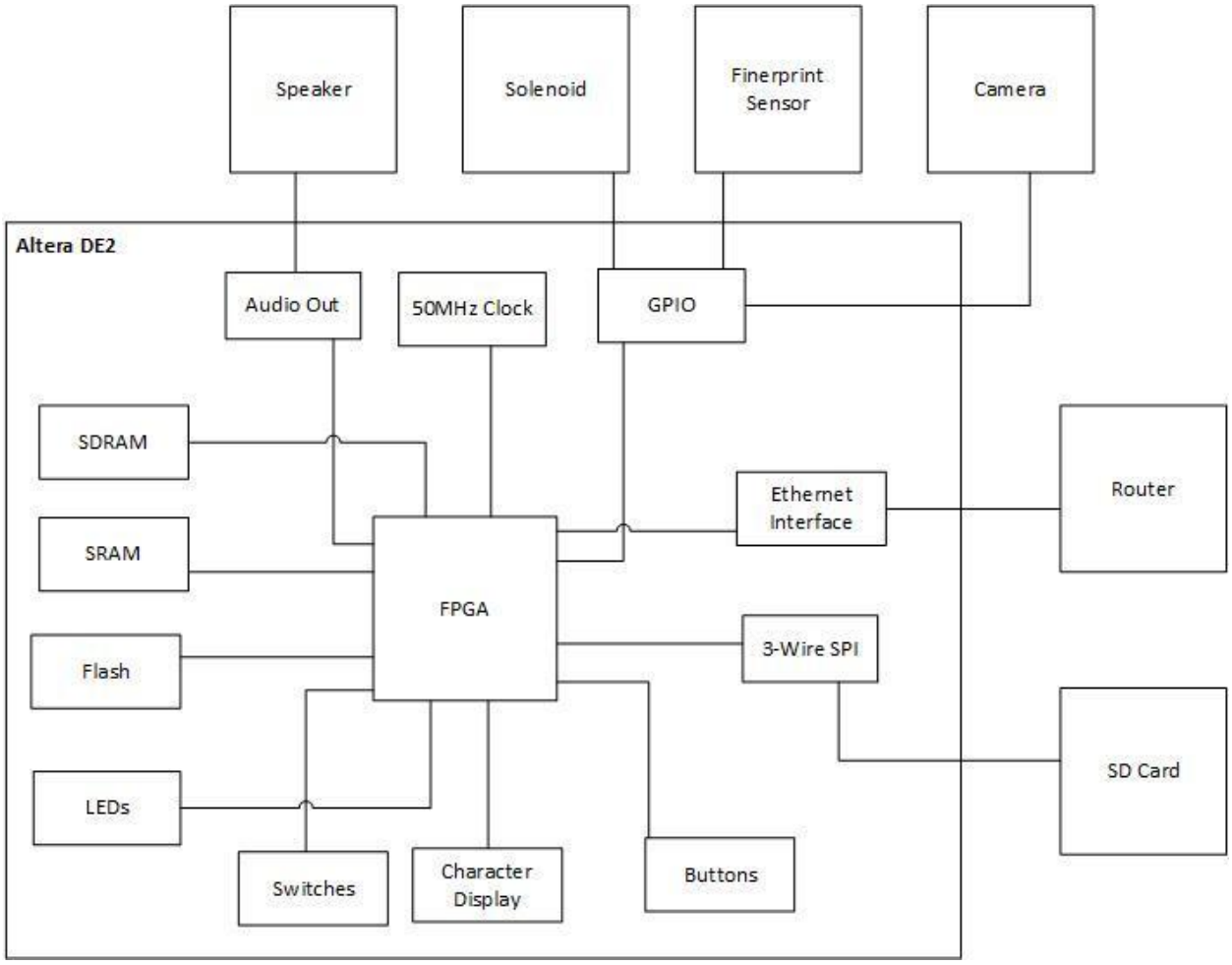


Figure 1: Hardware Block Diagram

HARDWARE - FINGERPRINT SENSOR

The main peripheral used is the ZhianTec ZFM fingerprint sensor. This component communicates with the DE2 board through UART using GPIO pins 8 and 9 for transmit and receive respectively. The board will transmit a request for a fingerprint from the sensor through pin 8 at 3.3 V. Each response made by the sensor through pin 9 is approximately 4.0 V. Although the DE2 has a 3.3 V I/O, the expansion header pins all have a 5 V tolerance [7]. Each GPIO pin has a 47 ohm resistor and a BAT54 schottky clamp to provide a safe voltage limitation. Since the sensor is the only peripheral sending data to the board through GPIO, it was concluded that it was safe to keep the transmission at 4.0 V. If additional interfacing is implemented through GPIO, a resistor can be placed in series with

each pin to lower the voltage sent to the DE2 board. In addition, the board will supply 5 V to power the sensor through a separate pin as the sensor's operating voltage is between 3.6 V to 6.0 V. In addition, the board and the sensor communicate with a packet content of a maximum 512 bytes with an additional 11 bytes for additional packet information. The sensor has a baud rate of 9600*N (N = [1, 12]), the default rate is 57600 bps. With the defined packet size and baud rate, serial communication between the DE2 and the fingerprint sensor should be possible. All of the registered fingerprints are stored in the sensor itself and the library of fingerprints stored in sensor can be accessed by the user. The fingerprint sensor utilizes three buffers of size 512 bytes to store the print: one image buffer to store the image, and two character buffers to store the translated image into data which will then pass the data into the library.

HARDWARE - SOLENOID

To control the locking mechanism of the door, the Pontiac F0432A open frame solenoid is used. When no power is supplied to the solenoid, the armature can move freely, and when power is supplied, the armature is pulled into the casing. As the board is not capable of supplying enough power to keep the door locked, an external power source is needed to drive the solenoid with enough current. The solenoid requires 12V and 19W to operate in an intermittent duty cycle pulling 1.6A of current [9]. The driver circuit shown in Figure 2. will be used to control the solenoid from the board through a GPIO pin. This circuit uses an NPN transistor capable of handling the required voltage and current of the solenoid. A diode is connected in parallel with the solenoid to ensure that the electromagnetic field from the solenoid does not damage the transistor.

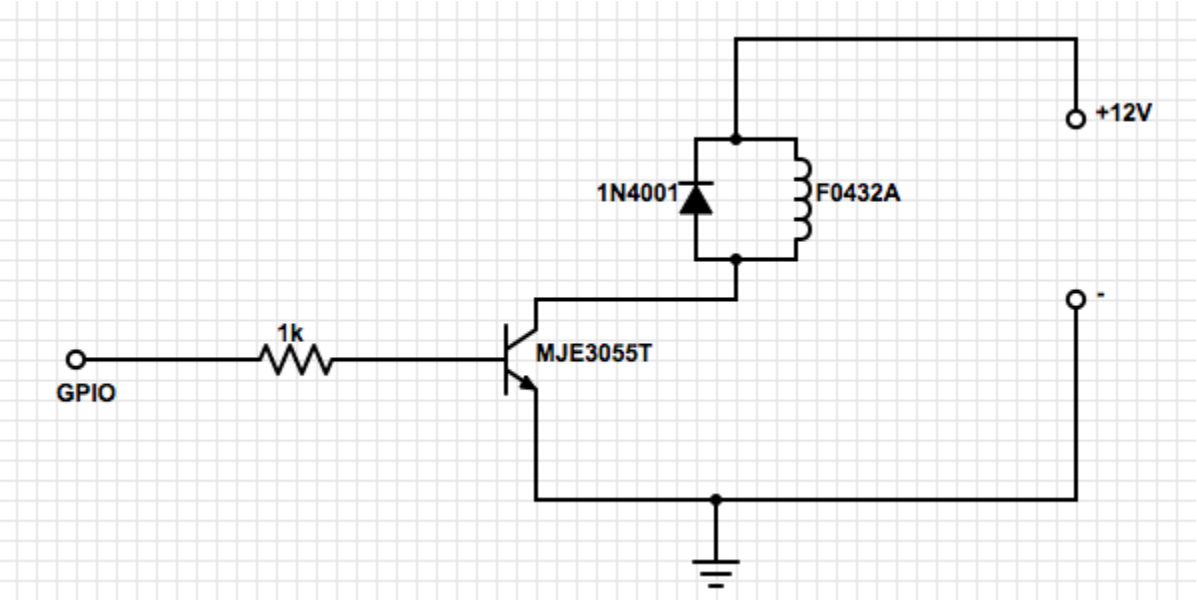


Figure 2: Solenoid Driver Circuit

HARDWARE - SD CARD

The SD card in this design is used to store files such as the alarm, log history and the database. Communication between the board and the SD card is implemented using SPI. This interface was chosen as opposed to the SD card interface due to the use of EFSL library, which creates file storage requirements that would need to be implemented on the card. In addition, the SD card is formatted as FAT16 with the cluster size as 32 kB, allowing for a 2GB limit. Since the SPI interface is being utilized and the contents of the SD card are accessed as clusters, speed is an issue when reading and writing to SD card, therefore the NIOS II/f is used.

HARDWARE - CAMERA

The Terasic D5M camera is used to capture images of the users. This camera is interfaced via GPIO: 12 pins for data, a pin to check for valid line, a pin to check for valid frame, and one pin for the dma transfer. The camera requires 3.3 V, which is supplied through one of the GPIO pins. The camera module is initialized by components supplied by Altera in their University IP. First the data stream is passed into a Bayer Resampler, which converts the image from a Bayer Pattern image into an RGB patterned image. This image is scaled and stored in SRAM via a DMA controller. The image resolution is limited to 320x240 due to memory constraints of the SRAM. Each pixel in the RGB image requires 4 bytes(Red, Green, Blue, Padding Byte), thus our image requires 307200 bytes of memory. The DE2 has 524288 bytes of SRAM, thus this was the largest image we could store. The image stored in SRAM is continually updated and can be read at any time.

SOFTWARE - FINGERPRINT SENSOR API

As the sensor did not come with an API library, it had to be created. The format of the data transmission packet is as follows:

2 bytes	4 bytes	1 byte	2 bytes	-	2 bytes
Header	Address	PID	Length	Content	Checksum

The content size is variable and is equivalent to the value set in the length field. The maximum content length of a packet is 512 bytes, however on average it is around 4 bytes. With the serial communication data packet format known, the communication functions described in the datasheet were implemented.

The fingerprint sensor has the ability to be password protected. If the password is changed from the default of "0000", the DE2 must confirm the password as part of the fingerprint sensor initialization sequence. This is done to protect the fingerprint data saved in the sensor. Once connection is established between the DE2 and sensor, the DE2 can utilize the API to communicate with the sensor.

To enroll a fingerprint, the DE2 requests for the sensor to store the current fingerprint in the image buffer to one of the two character buffers on the sensor. The finger must be scanned again and stored to the other character buffer on the sensor. A request is then sent for the sensor utilize both stored fingerprints to create a template and store it in flash memory. The generated template is used to match a scanned fingerprint.

To search for a fingerprint, the DE2 requests for the fingerprint sensor to scan a fingerprint and the resulting image is stored in the image buffer. The DE2 then requests for the fingerprint sensor to search the stored fingerprints for a match. If a match is found, the index in which it is stored is returned, otherwise -1 is returned. We can use the returned ID to match the fingerprint to a registered user.

SOFTWARE - DATABASE (EFSL)

The initial plan was to incorporate SQLite to store user and access history data. Due to missing dependencies, SQLite could not be compiled for the NIOS-II processor without making time consuming changes to the core of SQLite. Instead, a modified version of the EFSL (Embedded File System Library) created by group 12 from Winter 2013 was used [6].

In order to store our data in an efficient and logical manner, we chose to utilize the file system file structure for data storage. Each folder on the file system represents a table and each file represents a row. The name of the file is a combination of primary keys used in the file. The data in the file is stored in JavaScript Object Notation (JSON). This provides an easily parsable and readable dataset. An added benefit of storing data as JSON is that the data becomes easy to integrate into web applications or smartphone applications. The tables in the database include: roles, users, role schedule, user roles, user prints, and history. The database layout can be seen in the appendix.

Since a full-fledged SQL database was not required, only the necessary functions were implemented. This includes listing all the entries and their respective attributes for the requested table, inserting new entries into the table, entry lookup by ID, modification of entries and deletion of entries.

SOFTWARE - WEB SERVER / REST API

The web server is based off the example Web Server provided in the NIOS II development environment. Functions that handle incoming GET and POST requests have been altered in order to process incoming requests from our management application. Each incoming request URI is parsed and compared to a list of predefined accepted URIs. The requested action is performed (generally database calls) and relevant data is returned back to the requestor.

SOFTWARE - ANDROID APPLICATION

The Android application is used to provide remote control and modification capabilities for the owner. In order to use the application, the user must confirm their identity by scanning their fingerprint on the sensor to be verified before any modification to the system is allowed. If the device running the Android application is recognized by the system, the fingerprint verification step can be skipped and users can directly access the Android application; this is applicable for remote management.

The application includes a few management options including: users, roles, picture, log, time, unlock, and logout. The users option allows the owner to view a list of users currently stored in the database, this includes users who are not enabled but are still enrolled in the system. The owner also has the ability to modify any user's information such as: their name, if they are currently enabled, their roles, and their enrolled fingerprint. The owner can also add any additional users and assign them new roles or enroll their fingerprints.

The role option allows the owner to view a list of roles currently stored in the database, this includes roles that are not enabled but still exist within the system. The owner also has the ability to modify any role information such as: role name, if it is currently enabled, if it is an administrative role, users assigned to this role, and the role access schedule. The owner can also add any additional roles and assign new users or schedule to the role. In addition, the owner can add a new schedule to a role or modify the schedule of a role. This includes the role access privileges for: day of week, start time, end time, start date, and end date. Roles can also be deleted from the system.

The picture option allows the owner to view a current snapshot of the view at the door, checking the status of their property. The log option allows the owner to view a list of access history, displaying the date and time of access. A more detailed view of each access history can be viewed, displaying information such as: who attempted to access the property, if access was granted or denied, and the date and time of access.

The time option allows the owner to set the current time of the system. Each time the DE2 board is powered on, the system time is reset. Hence, this ensures that the current date and time of the system is correct. The unlock option allows the owner to remotely unlock the door. The logout option allows the system to forget the device running the Android application. This means that when the owner next uses the application, he will be required to scan his fingerprint to confirm his identity before management from the application can be completed. The logout option provides extra security so in the situation the device is lost or stolen, thieves cannot access the private property or make any modifications to the system.

The application wireframe can be seen in the appendix. By communicating with the REST API, the Android application can communicate with the board and send requests or updates to the database.

Software - Camera

To save an image from our camera, we allocate a buffer for our image, create and insert a BMP header and then copy the data from SRAM into our buffer. The contents of this buffer can be written to the SD card or transmitted via the web server to a remote endpoint.

Operation

With all the hardware integrated with the software, the two main functionalities of the system can be provided: accessing the private property and modification to the access privileges. The following describes the general flow of the procedure where Figure 3 shows the access flow and Figure 4 shows the modification flow.

Operation - Access

1. Fingerprint sensor pends for fingerprint.
2. Once fingerprint is detected, fingerprint ID is sent to the DE2 board via GPIO.
3. Board will interface with the database located on the SD card.
4. Database will compare fingerprint with list of authorized prints and return the result to the board.
5. If user has authorized and scheduled access, the board will unlock solenoid via GPIO and details are logged to the database.
6. If the user is not authorized, an alarm will sound (via speakers/audio out). The failed entry attempt is also logged to the database.

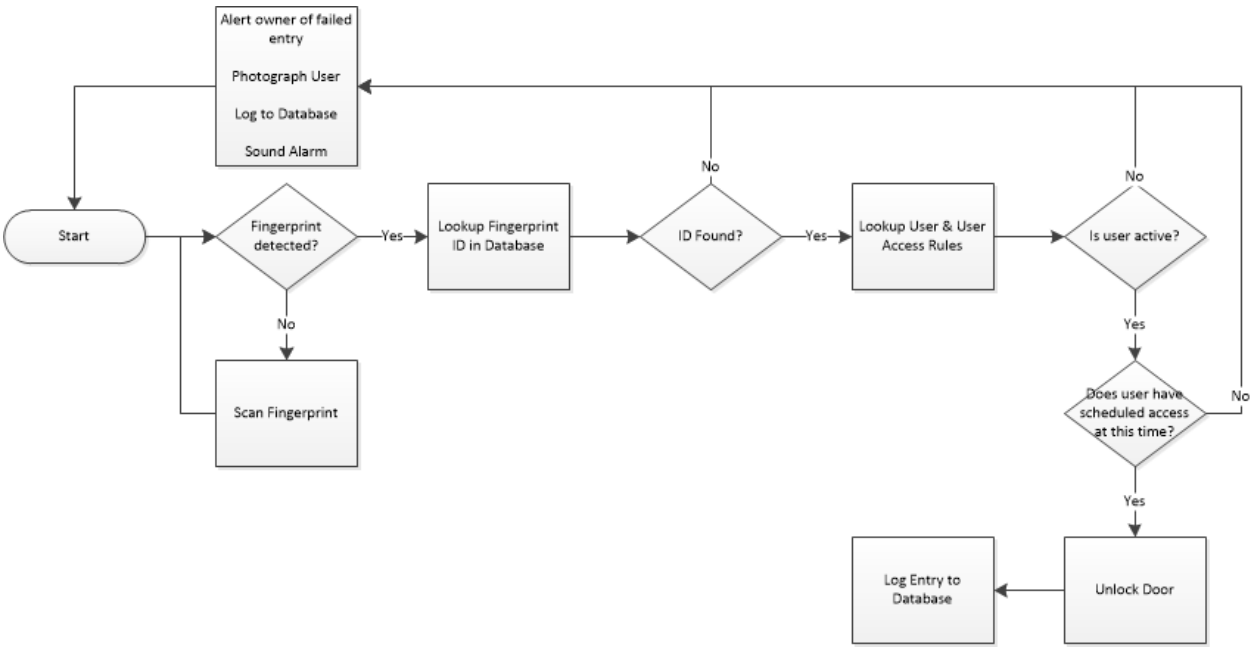


Figure 3: Operation for Accessing Property

OPERATION - MODIFICATION

1. User will access Android application and request for modification.
2. Application will communicate with board through REST API to send request for scan from fingerprint sensor.
3. Board will send request from fingerprint sensor via GPIO.
4. User scans fingerprint & sensor detects the print.
5. Fingerprint is transferred from sensor to board via GPIO.
6. Board will interface with the EFSL database.
7. Database will compare fingerprint with list of authorized prints and return result to the board.
8. If fingerprint is accepted, board will notify the application using the REST API, allowing the application to display the possible actions.
    a. Admin can add user or modify preferences through the application.
    b. Users can modify preferences through the application.
9. If fingerprint is not accepted, board will notify the application using REST API and the application will do nothing.

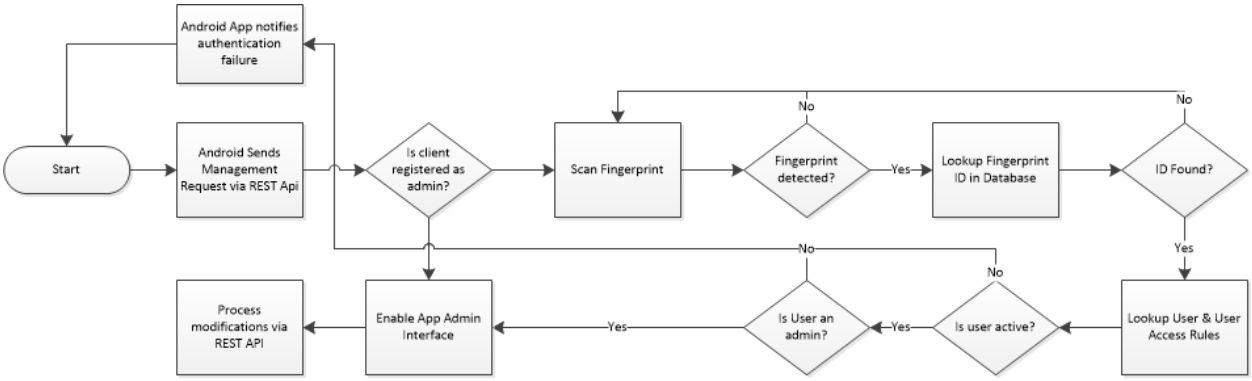


Figure 4: Operation for Modification of Preferences

BILL OF MATERIALS

Part	Description	Specifications	Supplier	Documentation	Cost
Altera DE2	FPGA Microprocessor board	Cyclone II 8 MB SDRAM 512 KB SRAM 4 MB Flash 50 MHz Clock	Altera	Link [2]	$495*
ZFM-20	Fingerprint sensor	TTL Serial 3.3 V 57600bps	Adafruit	Link [3]	$91
F0461A Solenoid	Locking mechanism for door	Open Frame Continuous Cycle 12 V	DigiKey	Part 1 Part 2	$36.79*
D5M Camera	Video intercom	Video In	Terasic	Link	$85*
Speakers	Alarm for failed entry	Audio Out	-	-	$20*
SD Card	Storage of database and log history	2 GB	SanDisk	-	$20*
Perfboard	Mounting of electronic components	-	-	-	$7.50*
Ribbon Cable	40-pin Ribbon Cable	40-pin Ribbon Cable	-	-	$10*
Sensor Enclosure	3D printed enclosure for fingerprint sensor	-	Edmonton Public Library	-	$3.50
				Total	$768.79

*Part supplied already/owned

**5V 2A Power supply required, not included in total

RESOURCES

WEB SERVER

The web server used is based off the example in the NIOS II development environment along with references to the application notes. This allows for the communication between the Android application and the DE2 board. The web server performed as expected, responding to all requests efficiently.

- Source Size: 141 KB
- Compiled Size: 41.9 KB
- https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/G9_WebServer_Appendum/

ETHERNET IP CORE - DM9000A

This IP core is the controller used to provide access to the Ethernet, allowing for the establishment of the web server, hence the communication of the DE2 board and the Android application. This core performed as expected, allowing the web server to run efficiently.

- https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/Ethernet_DM9000A/AppNotesEthernet.pdf

UART IP CORE - FIFOED AVALON UART

This IP core is the upgraded version of the Avalon UART which includes FIFO for the transfer and receive signal of the UART. By using this IP core, we are able to transmit the data between the DE2 board and the fingerprint sensor efficiently.

- http://www.alterawiki.com/wiki/FIFOed_Avalon_Uart

FILE STORAGE SYSTEM

This library is used for file storage on the SD card. This library allows access to the SD card as a file storage system. This creates easy access and modification of the database stored in the SD card.

- Source Size: 253 KB
- Compiled Size: 67.8 KB
- http://sourceforge.net/projects/efsl/

JSON LIBRARY

This library is used to encode data from a C++ struct to JSON. Data is stored in a JSON format because it is both readable by humans and it is a standard data format used by many REST APIs. By utilizing JSON objects, data can be transmitted and stored directly from the REST API. JSONCpp is used to parse and create JSON objects in C++ in our database access component.

- Source Size: 1783 KB
- Compiled Size: 233 KB
- http://jsoncpp.sourceforge.net/files.html

DATASHEET

BIOLOCK SYSTEM

Maximum Ratings
Current Draw = 0.771 A Voltage Draw = 9.07 V

The maximum power ratings were determined using a power measurement wiring harness. These values are obtained when the system was running actively, this included signaling the solenoid and also requesting data from the SD card.

FINGERPRINT SENSOR

FPGA to Board:

Board -- UART (transmit and receive) -- GPIO (2 pins)

Off board:

Power: GPIO_0(10) = 5 V

Transmit: GPIO_0(9) = 3.3 V

Receive: GPIO_0(8) = 3.3 V

Power Supply	Transmit	Receive
VIN = 3.6 V to 6.0 V IIN = 110 mA max	VOL = 0.4 V max VOH = 2.4 V min & 3.3 V max IOH = 4 mA	VIL = 0.6 V max VIH = 2.4 V min IIH = 30 uA

SOLENOID

FPGA to Board:

Board - UART (transmit) -- GPIO (1 pin)

Off board:

Power: Driver circuit = 12 V

Receive: GPIO_0(7) = 3.3 V

Power Supply
P = 19 W VIN = 12 V IIN = 1.6 A

CAMERA

FPGA to Board:

RAM -- Scaler -- Bayer Pattern Sampler -- Video In -- Audio & Video Config

Off board:

Power Supply	Transmit/Receive
VIN = 3.3 V	V = 1.7 V to 3.1 V

Pin	**Name**	**Direction**	**Description**
1	PIXCLK	Output	Pixel Clock
2	D[11]	Output	Pixel data bit
4	D[10]	Output	Pixel data bit
5	D[9]	Output	Pixel data bit
6	D[8]	Output	Pixel data bit
7	D[7]	Output	Pixel data bit
8	D[6]	Output	Pixel data bit
9	D[5]	Output	Pixel data bit
10	D[4]	Output	Pixel data bit
12	GND	N/A	Ground
13	D[3]	Output	Pixel data bit
14	D[2]	Output	Pixel data bit

15	D[1]	Output	Pixel data bit
16	D[0]	Output	Pixel data bit
19	XCLKIN	Input	External input clock
20	RESETn	Input	D5M reset
22	TRIGGER	Input	Snapshot trigger
23	STROBE	Output	Snapshot strobe
24	LVAL	Output	Line valid
25	FVAL	Output	Frame valid
26	SDATA	I/O	Serial data
27	SCLK	Input	Serial clock
29	VCC33	N/A	Power 3.3V
30	GND	N/A	Ground

NETWORK

FPGA to Board:

Board -- Ethernet Controller

Off board:

Router is connected to ethernet port. Router also has own power supply.

SD CARD

FPGA to Board:

Board - 3 Wire SPI

Off board:

SD card is placed in the SD card slot.

SPEAKER

FPGA to Board:

Board -- Audio Codec -- Audio Out

Off board:

Speaker is connected to the audio out.

BACKGROUND READING

USB CAMERA

The initial goal was to implement a camera using USB, research was done on interfacing a USB component and whether it is a feasible option. Research was first completed on this topic by reading on articles related to USB communication with FPGAs.

- Article on USB Camera communicating with FPGA http://ieeexplore.ieee.org.login.ezproxy.library.ualberta.ca/xpls/icp.jsp?arnumber=6140863

Research on the hardware was also completed by looking at the VHDL examples.

- Top-Level for USB https://github.com/mzakharo/usb-de2-fpga/blob/master/src/usb.vhd
- VHDL Component https://github.com/mzakharo/usb-de2-fpga/blob/master/support/DE2_NIOS_DEVICE_LED/HW/ip/ISP1362/ISP1362_IF.v

As time was limited, the use of a USB camera was replaced with the D5M camera.

FINGERPRINT SENSOR

As the fingerprint sensor did not come with an API library, implementation of the API library was required for the system. The newest version of the datasheet was in Chinese, therefore an Arduino API library was examined.

- Arduino API library used for reference https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library

ETHERNET

Upon looking at the application notes created from previous years, an ethernet example was found. This application note was used as guidance for setting up ethernet and the web server. However, modifications were made to the web server to handle all requests made from the Android application.

- https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/Ethernet_DM9000A/AppNotesEthernet.pdf

AUDIO

An audio application note was also found. Although the example utilized the the codec fully and converted audio signals from Mic-In to Audio-Out, this was helpful in completing the hardware components of the system.

- https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/audio_altera_university_ip_cores/

SOLENOID

This example was useful in designing the solenoid locking/unlocking mechanism.

- https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/G2_GPIO_Interfacing/

D5M CAMERA

This research article was used as a guidance when implementing the D5M camera module. As the documentation provided with Altera's University IP is poor, this was used as a reference for settings and features required/provided by the D5M.

- http://www.diva-portal.org/smash/get/diva2:403409/FULLTEXT01.pdf

SOFTWARE DESIGN

Many different software processes need to interact with each other in order for the system to function as a whole. As there are several components to the design, several threads were used with separate processes that interact with each other.

LIBRARIES

In order for the system to be integrated more efficiently, two libraries have been selected for use: JSONCpp and EFSL.

LIBRARIES - EFSL

The embedded file system library is used to help provide more features in accessing the SD card. If the SD card HAL device driver functions are used, options available are the ability to search for files, read and write to files, and set the file attributes. With the use of EFSL, available options include the ability to use the subroutines provided by the HAL device driver, as well as make directories, remove files, etc. With these additional features, it is easier to implement a custom database system.

LIBRARIES - JSONCPP

By using the JSONCpp library, JSON objects can be parsed and created in C++. This allows for easy transmission and storing of data as it is the data format used by our REST API. Once a new request is received from the REST API, data can be sent directly to the database as JSON objects without requiring more manipulation to the data. Database results can also be sent directly to the REST API without excessive manipulation of the data.

WEB SERVER/REST API

The web server thread is used for communications between the Android device and different peripherals in the system such as fingerprint sensor and the database. The following is the flow of the thread:

```
Wait for incoming connections.
if (connection.isAdminAttempt){
        if (!device is recognized admin device &&
        (scan fingerprint && !fingerprint.user.isAdmin()){
                return -1;
        }
        if(user enrollment request){
                Scan fingerprint, save user ID with user details provided in the request.
                Return new user ID.
        else if((user / role / role schedule / print / history) listing request){
                Return relevant data.
        } else if (role creation request){
                Create role with associated data.
                Return success indication.
        } else if (user-role enrollment request){
                Add user to specified role.
                Return success indication.
        } else if.(role scheduling request){
```

```
                Create / edit / update role as requested.
                Return success indication.
        }
}
Return error message
```

ANDROID APPLICATION

The Android application will have options of allowing the owner to manage users and roles, check the log history, view a snapshot of the guest at the door, set current time, and unlock the door. Users are able to manage their preferences. Depending on the role of the user, they will have the capability of selecting these different options. Once the owner clicks on a button, the Android application will send requests for data from the database or access requests from the fingerprint sensor through the web server.

FINGERPRINT SENSOR

As the fingerprint sensor did not come with an API, it was necessary to implement it. Using the created API, it is possible to communicate with the board to send requests and replies. The following is the flow of the fingerprint sensor thread:

```
while(1){
        while(fingerprint not detected){
                Poll for fingerprint
        }
        if (alternate task is waiting for fingerprint){
                Send data to mailbox
                continue;
        }
        lookup fingerprint in database
        if (fingerprint.id == -1 || !fingerprint.accessAllowed()){
                Entry failure notification
        }
}

fingerprint.accessAllowed(){
        if ( !fingerprint.user.exists() ||
        !fingerprint.user.enabled ||
        !fingerprint.user.active ||
        !fingerprint.user.roles.allowedAccess()){
                return false;
        }
        return true;
}
```

DATABASE

The database will store the information of the system. This includes the users permitted to unlock the lock, their roles and their respective access periods, access history, etc. Access to the database can be made through the custom library created. This includes accessing all entries within a table, accessing a certain entry within a table, etc. Access to the database is integrated into the system, where references to the database will be made when necessary.

TEST PLAN

In order to test the different components on the board, the board will be put into testing mode. Switches will be used to enable different modes in order to debug different components separately. The LCD display can be used to display debugging information and system state during testing.

FINGERPRINT SENSOR

When testing the fingerprint sensor, a diagnostic sequence was used. This included enrolling a fingerprint and ensuring it was correctly enrolled by verify that the same print could be recognized. The fingerprint was then erased, where the same finger was scanned again to ensure the print was not recognized. This would verify proper communication, connectivity and functionality of the sensor.

SPEAKER

The speaker was tested by continually playing an audio sample through a test thread when the DIP switch was toggled.

CAMERA

The camera was tested by displaying the image via the web server. The image was also saved to the SD card. This ensures that the image could be transferred successfully and was a clear image.

REST API

The REST API was tested via a web browser to ensure expected data is returned. This was achieved by accessing the specified URI and checking the response was a valid JSON format.

SD CARD

The SD card was tested by writing and reading from the SD card. A DIP switch was used to clear and write test data to the SD card. Once the database was cleared, a laptop was used to verify that the SD card was empty. When the database was populated with test data, a laptop was used once again to verify that the SD card contained the correct data.

LOCKING MECHANISM/SOLENOID

The solenoid driver circuit was tested by connecting to a DC power supply before connecting to the board. It was then tested with the board by sending a request to unlock when a DIP switch was toggled.

SOFTWARE

The software was tested by allowing it to run for 30 minutes. Checks for memory allocation were made periodically. There should be no substantial increase in memory usage.

ANDROID APPLICATION

For each user selectable option in the Android application, it was tested by sending a request to a server. The server contained simulated responses for each request. These responses included sample data, invalid responses, and empty datasets. Cases where no response were sent was also tested.

INTEGRATED TESTING

After testing each component separately, a test was completed with the whole system integrated together. This was done by registering a fingerprint to the sensor, then attempting to unlock the door through the Android application as well as through the fingerprint sensor. Many test cases were generated to test for failure at each component, and for functionality of the system. The performance of the system was also be tested to ensure the system is fast enough for normal usage. After this test was completed, it was concluded that all functional requirements have been met. The only issue was the speed of the SD card. Initially the NIOS/economy was used but was too slow for normal usage. Therefore, the processor was upgraded to the NIOS/fast where acceptable speed was achieved.

RESULTS

FINGERPRINT SENSOR

Before connecting the fingerprint sensor to the board, it was necessary to ensure that the RS-232 from the board to the sensor transmitted a signal of 3.3V. This was done by measuring the transmit signal transmitted out of the board using the oscilloscope. From there, we measured an output signal that was too high to be supplied to the fingerprint sensor without damaging it. Instead, serial communication through GPIO was used as it would only require changes to the top level. Once again, measurements were made at the transmit signal, where the oscilloscope measured a voltage of approximately 3.3V out of the board. When measuring the transmit signal from the sensor, the oscilloscope measured an output voltage of approximately 4.0 V. Although the GPIO pins on the board have a VIO of 3.3 V, the board has a 5 V tolerance, 47 ohm resistor and a BAT54 schottky clamp to provide a safe voltage limitation. From these considerations, the sensor was directly connected to the GPIO pins of the board.

WEB SERVER

While preparing to set up nginx as the HTTP server for preliminary testing, compilation was attempted for the DE2 board. It was noted that nginx could not be compiled as it creates multiple worker threads. Unfortunately, MicroC/OS-II does not support the creation of multiple threads at the same priority. In addition, MicroC/OS-II is not POSIX compliant, so significant changes would need to be made to the nginx code base. Therefore, the choice of server was changed to a socket server instead.

The web server now being used is based on the supplied SimpleWebServer code from Altera. The web server serves requests as expected.

ETHERNET

The Ethernet component has been set up and is functional in an environment with an active DHCP server.

SD CARD/DATABASE

When implementing the SD card, the SD card interface IP core was first chosen. When attempting to read and write to the SD card, it was noted that function calls available were too simple to implement the database design required. Therefore, the SPI IP core was used along with the use of the EFSL. This allowed for creation of directories, removal of files, and reading and writing to file. Once the SD card was set up, the peripheral was tested with the database design by creating test data to input to the database. All data was checked for proper storage in the SD card by plugging in the SD card to a computer, and checking that files existed in the correct place, and were not empty. After testing that all data was successfully stored into the SD card, the reading capability of files was tested. All file contents were able on the console, concluding that reading from the SD card was successful.

ANDROID APPLICATION

The Android application functions properly and responds to all user inputs. When a user selects an option, the transition between screens occurs smoothly with no delay. If a request is sent from the application to the DE2, a loading screen will display prompting the user to wait. When a response is received by the application, the loading screen disappears and the application will respond as directed. In addition, the user is disabled from clicking any buttons on the screen while waiting for a response. This was implemented to ensure that only one request is sent at a time and only one copy of the following screen will be displayed. This concluded that the Android application works successfully. The only issue with the application is the wait time due to the speed of the SD card. As the DE2 needs

to wait for a response from the database, it forces the Android application to wait for the database as well. Hence, there is a large wait time which affects the usability of the application.

D5M CAMERA

Very good results were achieved from the D5M camera. The image stored in SRAM was very clear with accurate colour representation. Initially, large amounts of image skew were present. This skew was likely caused by wire length differences due to wire wrapping combined with interference. Skew issues were resolved by slowing down the data transmission rate and using a shorter ribbon cable. It is important to note that when using the provided University IP for the D5M camera, the registers to flip the image horizontally or vertically should not be set. During our trials, these registers were set, flipping the image output from the camera before processing. The image outputted from software had image colouring issues. This was due to the Bayer Pattern Resampler naively interpreting the camera output data as a valid bayer pattern. Since a bayer pattern is not symmetric, all blue pixels were swapped with green pixels and half of the green pixels were swapped with red pixels. We were able to resolve this issue by flipping the image vertically in software instead after being processed in hardware.

SPEAKERS

The speaker utilizes the audio video configuration core along with the audio out core. As the config core automatically sets up the hardware component, work was only needed in transmitting the audio signal to the codec. When the audio signal was sent to the codec, the speaker produced an audible signal, but also included some noise. The speed of the audio signal being played was also slightly off, hence it caused a change in pitch.

SAFETY

External power sources were not used except for the power supply to the board (9V adapter) and to the router. In addition, all signals are less than 5 V or 120 mA (calculated power of 0.6W), meaning there are very few major safety concerns. A small concern is the driver circuit for the solenoid which utilizes 12 V and 1.6 A, producing 19 W. When dealing with the driver circuit, it is important to ensure that all grounds and power supplies are connected properly to avoid hazards such as loose grounds. As there are no moving parts in the design, there are no concerns about tripping hazards. While the solenoid coils may heat up slightly, the peripherals will not heat up significantly. There are not high enough temperatures for concern about burning oneself while dealing with the equipment. In order to protect the equipment, grounding straps must be used to prevent static discharge from damaging components.

ENVIRONMENTAL IMPACT

The system does not contain any hazardous substance, where all components are lead free. This allows the project to be RoHS compliant. The need to supply constant power to the DE2 board as well as the solenoid may increase the usage of power consumption. One way to reduce environmental impact would be to use solar panels to power the board and solenoid driver circuit. With enough sunlight during the day, solar panels may be able to store enough energy to also supply power to the system at night.

SUSTAINABILITY

The DE2 board needs to be running continuously for the system to be working. This also includes the power supply to the driver circuit in preparation for when the door needs to be unlocked. The DE2 board will need a constant supply of 4.6 V, and the solenoid driver will need 12 V and 19 W. These values are determined for when the system is in active mode. When the system is in idle mode, the solenoid driver will not require 19 W. Therefore, the average power consumption would depend on the DE2. When measuring the power consumption of the system using the power measurement wiring harness, it measured a maximum current draw of 0.771A and a maximum voltage of 9.07V. This measurement was taken when all components were running, which includes instructing the solenoid to unlock. This produces a power consumption of approximately 7W. Assuming the cost is 8.7 cents per kW, the energy cost per year is quite small. If the user decides to reduce their environmental footprint, solar power can also be used to provide power to the system. If the fingerprint sensor needs to be replaced, all the fingerprint data on the sensor will be lost as well. The sensor enclosure will prevent some damage to the sensor; however, users must remain careful in ensuring the fingerprint sensor is used carefully. Since the SD card is flash storage, users must also ensure writing to the SD card is set to minimum. This applies especially to users who might change the users or roles in the system quite frequently, causing the SD card to need to be replaced very quickly.

REFERENCES

[1] Altera Corporation, *Literature: Nios II Processor*, V11.0, May 2011

[2] Altera Corporation, *DE2 Development and Education Board User Manual*, V1.6, 2012, ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE2/DE2_User_Manual.pdf

[3] Tarek Kaddoura and Jigar Nahar, *Altera DE2: DM9000A Ethernet Controller Application Notes*, 2013, https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/Ethernet_DM9000A/AppNotesEthernet.pdf

[4] Zhiantec, *ZFM-20 Series Fingerprint Identificatino Module User Manual*, V1.4, Sept 2008, https://github.com/berickson1/ECE492/blob/master/Resources/ZFM%20Datasheet.pdf

[5] Altera, *FIFOed Avalon Uart*, June 2012, http://www.alterawiki.com/wiki/FIFOed_Avalon_Uart

[6] ECE 492 Winter 2013, *SD Card Interfacing - Group 12*, Feb 2013, https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/SD_card_interfacing/

[7] Amiga FPGA Accelerators, *Voltage Level Translation*, March 2013, http://majsta.com/modules.php?name=News&file=article&sid=13

[8] Altera Corporation, Literature: *DC Characteristics and Timing Specifications*, February 2008, http://www.altera.com/literature/hb/cyc2/cyc2_cii51005.pdf

[9] DigiKey, *F0461A Pontiac Coil*, http://www.digikey.ca/product-detail/en/F0461A/527-1020-ND/668305

[10] SourceForge, *JSONCppI*, http://jsoncpp.sourceforge.net/files.html

[11] Edgewise-Media, *SanDisk SD Secure Digital Media*, 2010, http://www.edgewise-media.com/sasdca20gb.html

[12] Terasic, *TRDB D5M User Guide*, Aug 2010, http://www.terasic.com.tw/attachment/archive/282/TRDB_D5M_UserGuide.pdf

[13] Codeguru, *URI Encoding and Decoding*, Nov 2006, http://www.codeguru.com/cpp/cpp/algorithms/strings/article.php/c12759/URI-Encoding-and-Decoding.htm

APPENDIX

QUICK START MANUAL

Setup:

1. Connect the DE2 to an active DHCP enabled network via ethernet.
2. Connect a 40-pin ribbon cable to the forward header on the DE2.
3. Connect the 40-pin ribbon cable to the middle 40-pin header on the perf board.
4. Attach the D5M camera sensor to the forward header on the perfboard.
5. Attach the ZFM Fingerprint module to the 4-pin header on the perfboard.
6. Attach the solenoid to the 2 pin header on the perfboard.
7. Connect 12 V power supply to wires on perfboard. (Power to red, ground to black)
8. Power on the DE2 and load BioLock.pof into flash
9. Press button 1 on the DE2 to reset the board.
10. Load the Android application onto an Android device connected to the same network as the DE2.

Controls:

1. Set time as current time on the Android application
2. Add new user to the system on the Android application
3. Add new print to user on the Android application
4. Add new role to the system on the Android application
5. Assign schedule to the role on the Android application
6. Assign role to user on the Android application

FUTURE WORK

Currently, the system will sound the alarm at every failed entry attempt. This can become quite irritating if the user places an authorized print at a slight angle. Therefore, a simple option is to implement a timer before an alarm is sounded. This means that if an authorized print was undetected or considered invalid, there is a set period of time for the user to scan a valid print before the alarm is sounded.

One possible extension is to automatically capture an image of the guest for each failed entry attempt. This would allow the owner to track all attempted entries and place a visual image to each entry. In addition, if valuables are stolen from the private property, the owner can notify the authorities and provide a visual representation of the perpetrator.

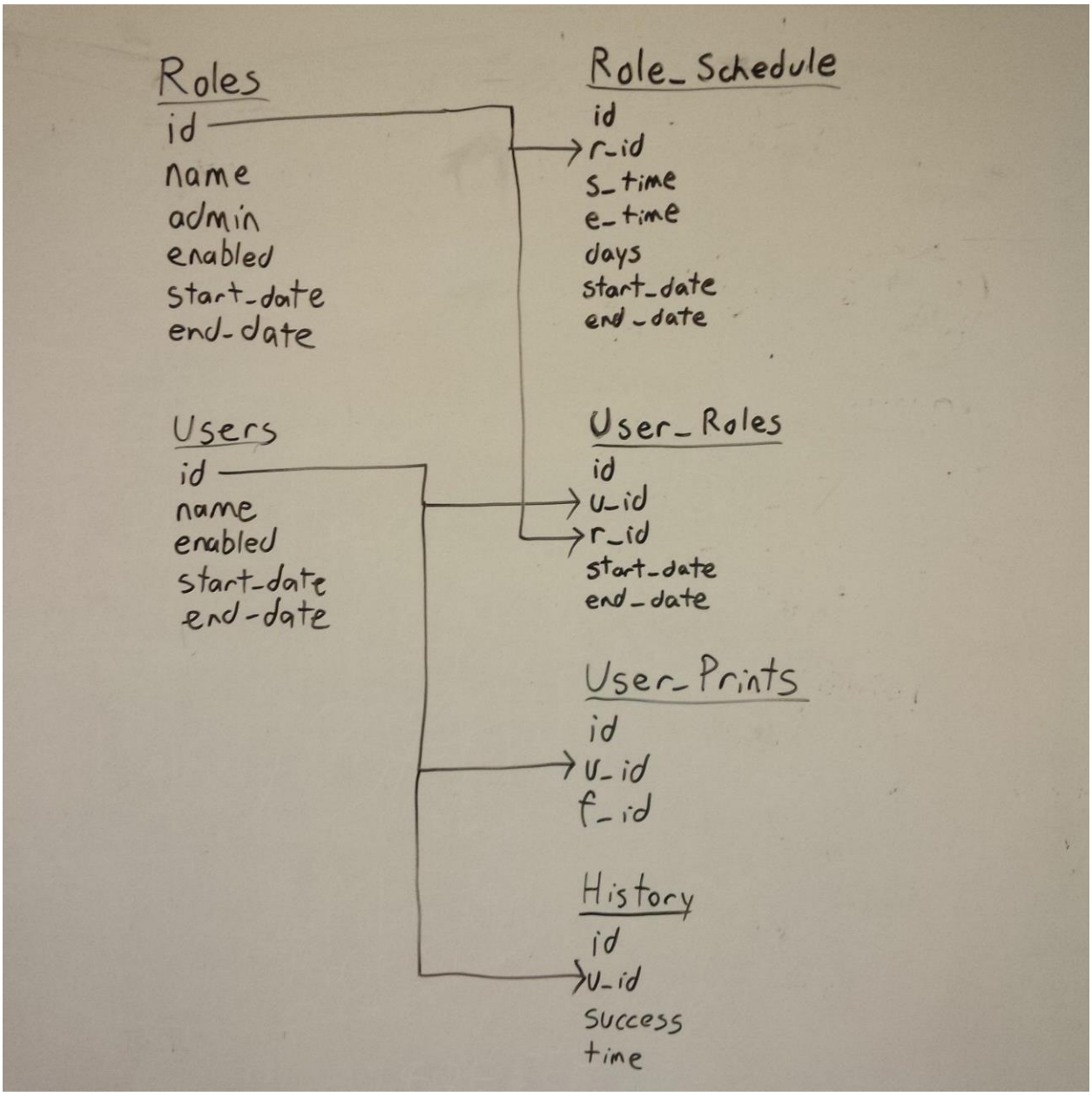
Another possible extension would be the addition a VOIP intercom to use with the D5M camera. This would allow the owner to communicate with the guest vocally. If video streaming is also incorporated, the owner can also communicate with the guest visually. This would give the homeowner the ability to decide if it is safe to bypass the lock system and allow the guest(s) to enter by remotely unlocking the door.

Another possible extension would be to introduce two-factor authentication. This could be done by either including a keypad on which the user would enter a PIN code, or through the use of the Android application.

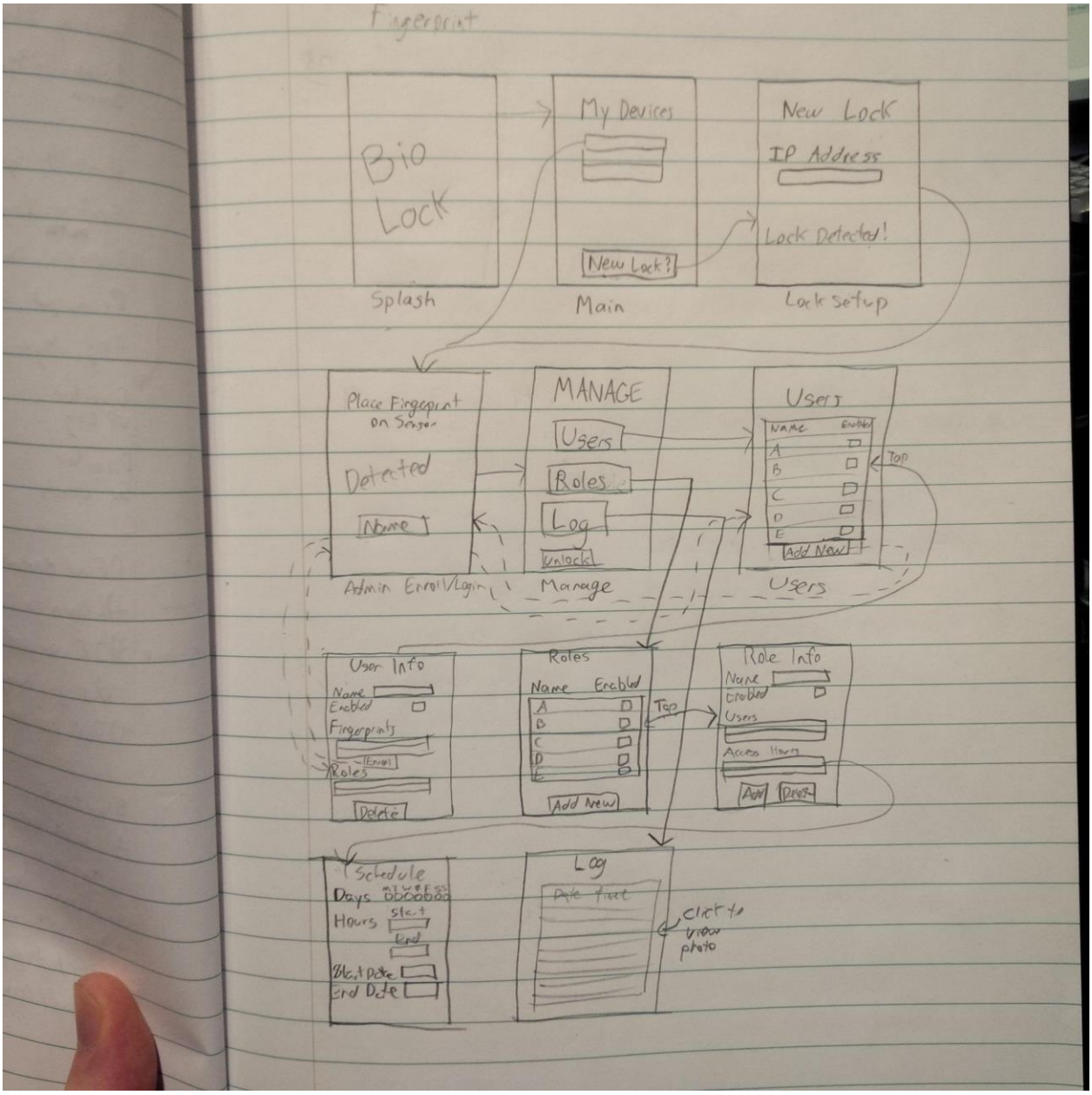
A possible option for fingerprint verification includes the use of verification directly on the device. This would eliminate the use of embedded flash storage on the sensor, which would allow for replacement of the sensor without losing data.

Communication security was largely ignored when implementing this solution. In the future, all communication with the web server should occur over HTTPS along with a time limited token to ensure that unauthenticated users are unable to make changes to the system.

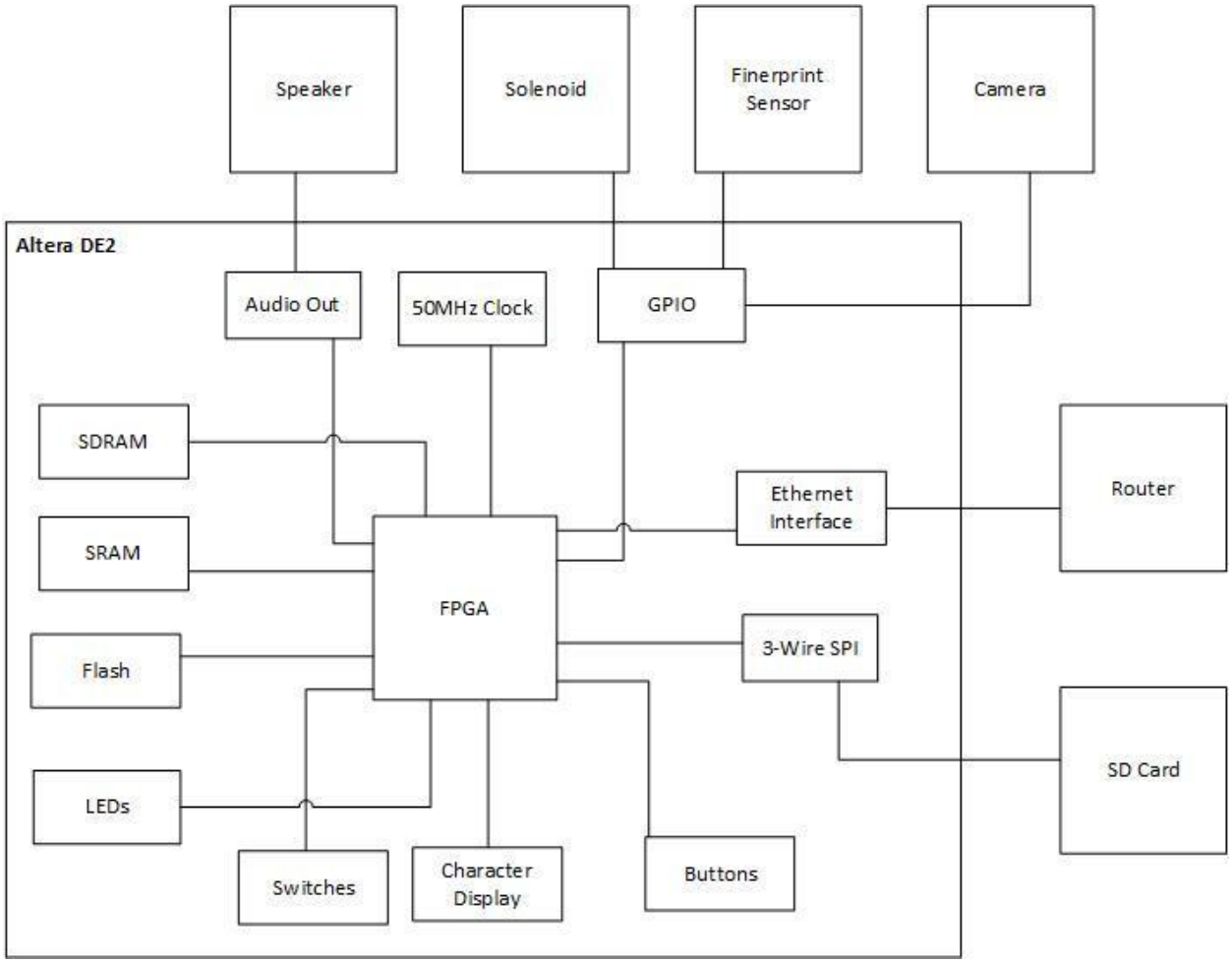
DATABASE SCHEMA



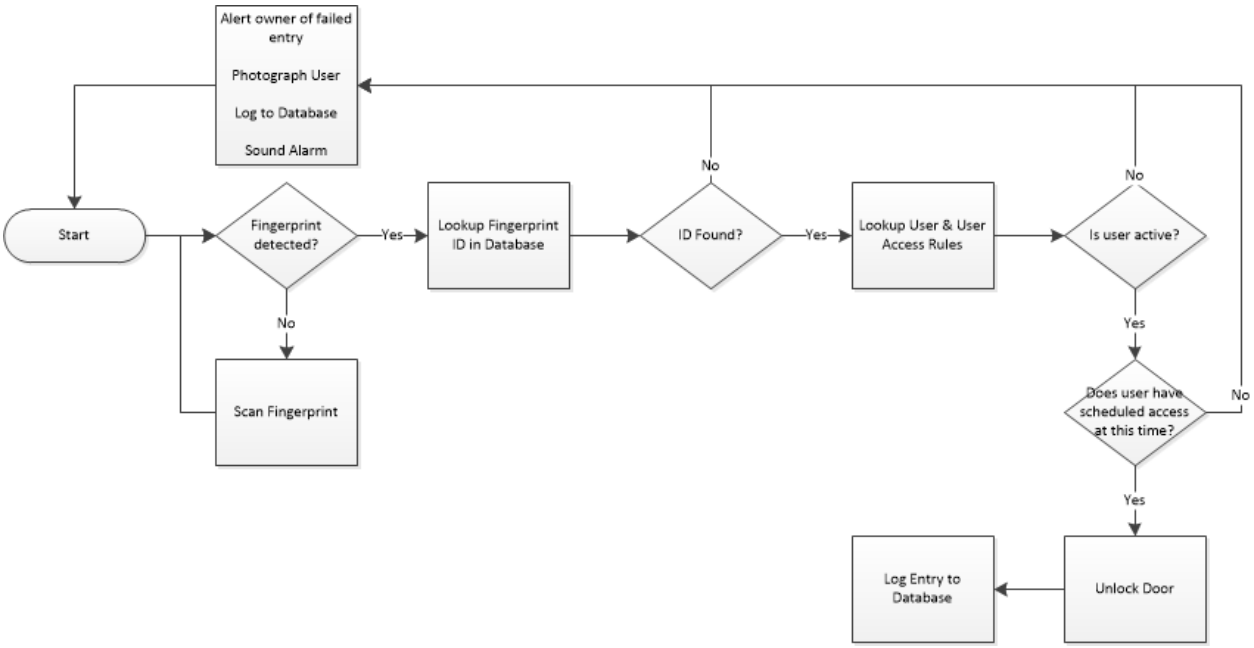
ANDROID APPLICATION WIREFRAME



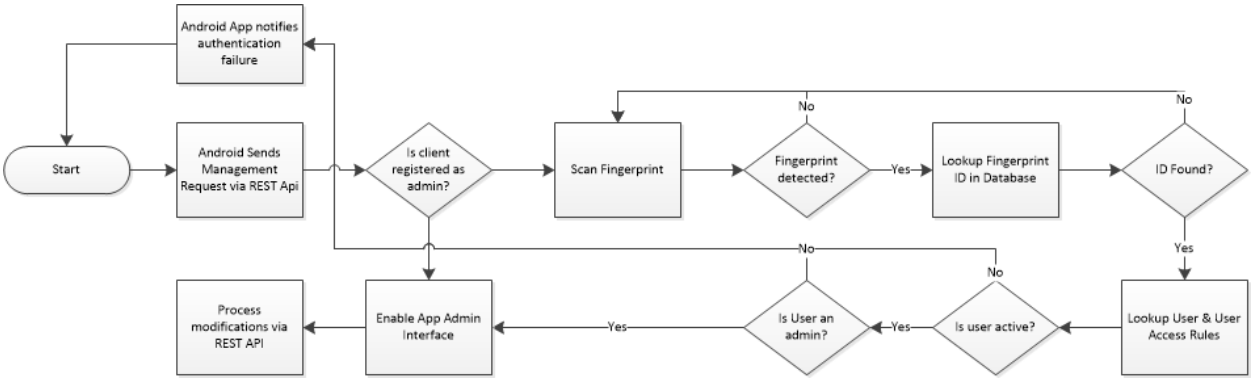
HARDWARE DOCUMENTATION



SOURCE CODE



Operation for Access



Operation for Modifications

ALL SOURCE CODE IS AVAILABLE AT [HTTP://GITHUB.COM/BERICKSON1/ECE492](http://github.com/berickson1/ECE492)

MAIN FUNCTIONS:

BioLock.vhd - Top level for BioLock project, T

main.cpp - Main software file used for all processes, T

CAMERA:

Camera.cpp - File used to capture image and process to BMP format, T

Camera.h - Header file for camera, T

TIME:

Time.cpp - File used to set and retrieve system time, T

Time.h - Header file for time, T

REST - API:

RestAPI.cpp - Rest API library for application & database communication, T

RestAPI.h - Header file for Rest API, T

WEB SERVER:

web_server.c - Web server, T

web_server.h - Header file for web server, T

http.c - HTTP response for communication with REST API, T

http.h - Header file for HTTP response, T

DATABASE:

Database.cpp - API for database, T

Database.h - Header file for database API, T

Database_CONST.h - Constants for database API, T

DatabaseTableTypes.h - Definition of contents for database entries, T

ANDROID APPLICATION:

/Software/Android/ - Android application files, T

AUDIO:

Audio.cpp - File used to sound the alarm, E

Audio.h - Header file for audio, E

SOLENOID:

Solenoid.cpp - File used to unlock the solenoid, E

Solenoid.h - Header file for solenoid, E

FINGERPRINT SENSOR API IS AVAILABLE AT HTTPS://GITHUB.COM/BERICKSON1/ADAFRUIT-ZFM-LIB

ZFMComm.cpp - API for fingerprint sensor, T

ZFMComm.h - Header file for sensor API, T

ZFM_CONST.h - Constants for sensor API, T