NFC Smart Door

Near Field Communication Smart Door Lock and Web Interface

An NFC door lock with a web interface for remote control and monitoring.

Daniel Fiske | dfiske@ualberta.ca
Michael Lam | michael.lam@ualberta.ca
Daniel Tiam | tiam@ualberta.ca

Abstract

The Near Field Communication (NFC) Smart Door involved designing, integrating, and creating components to enhance a simple door lock using a Terasic DE2 Board. While still supporting standard keys the door also reads NFC devices including Android devices with NFC to electronically control access using a powered door strike. A web server provides a browser independent interface used for administration, status monitoring and remote lock control. All door events are also logged with timestamps from a Real Time Clock (RTC) module and are viewable via the web interface. The registered keys and history are backed up periodically to an SD card for data persistence.

Table of Contents

Functional Requirements

The NFC Smart Door can be summarized in the following use cases:

1. The user unlocks the door with a registered NFC key.
2. The user controls the current status of the lock using the web interface.
3. The user monitors the current and past status of the door and lock using the web interface.
4. The user manages the registered keys using the web interface.

Design and Description of Operation

The following list outlines the operational feature set:

- **Electronic Door Access:**

The door lock is electronically controlled by the system. It fails secure which means the door locks when power is lost. For safety, the door can be manually unlocked to exit.

- **NFC Device Support:**

The system supports ISO 14443 compatible NFC devices including credit cards, ID cards, stickers, and Android phones as keys to unlock the door. Up to 20 keys are allowed.

- **Remote Web Interface:**

The system web server provides an interface to:

  - View the current door status
  - Unlock and lock the door
  - View and clear history
  - Add and remove keys
  - Set the RTC

- **Track Status and History:**

The system stores the last 200 events. The following are valid events:

  - Unlocked by NFC key
  - Unlocked by web
  - Locked by web
  - Doorbell ring
  - Invalid NFC key used
  - Door opened
  - Door closed

- **Persistent Data:**

The historical data and registered keys are stored on the SD card as persistent data.

- **Accurate Timestamps:**

In order for the system to have accurate timestamps after power or internet loss, an RTC module is used to keep time.

- **Android App:**

An app was created that uses Host Card Emulation to enable Android devices as keys.

System Architecture

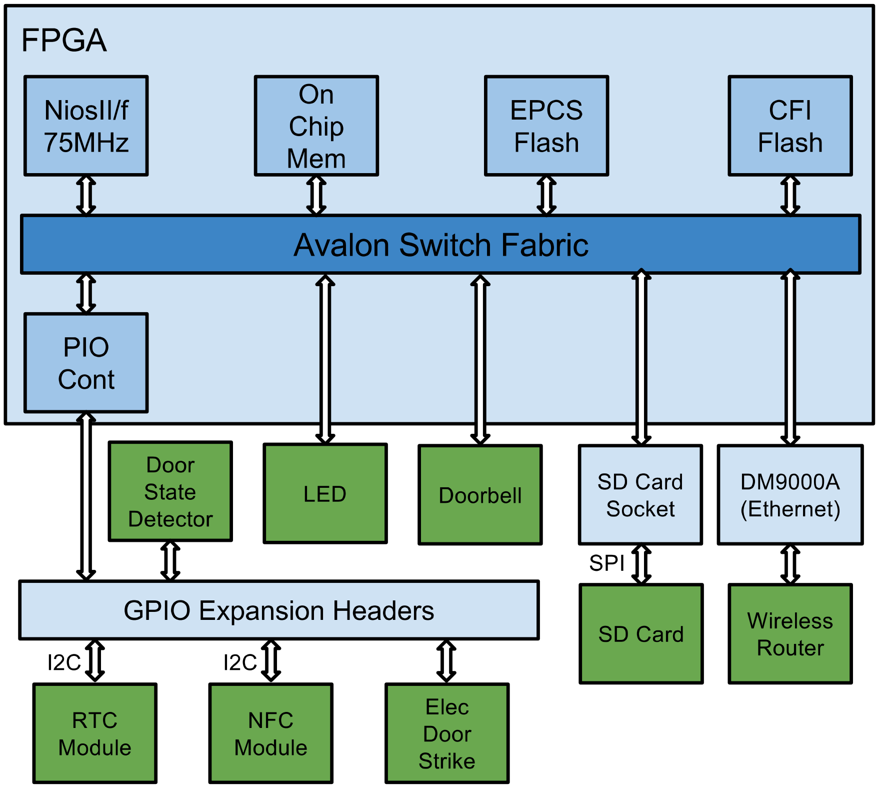


Figure 1: Hardware Block Diagram

Data Flow

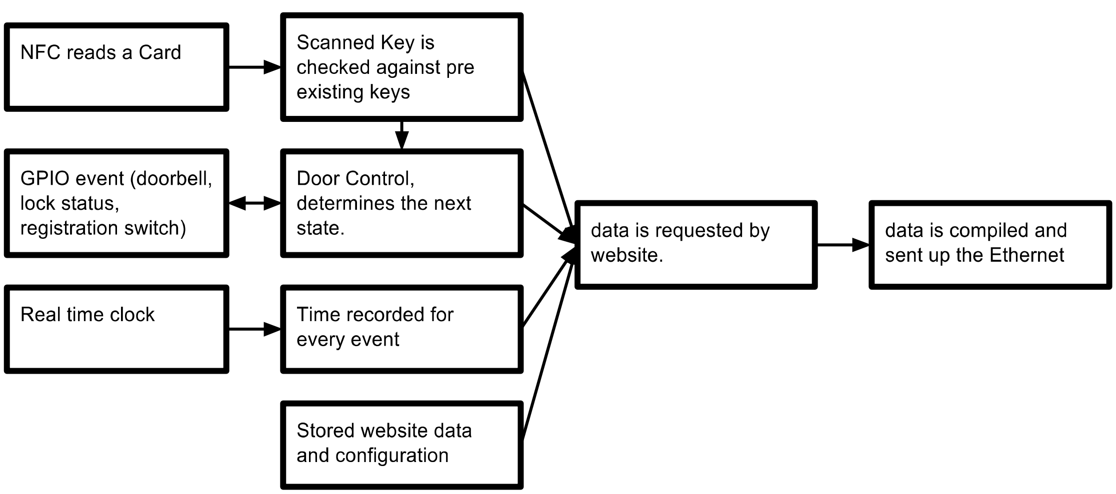


Figure 2: Use Case Data Flows

Web Server Request Sequence

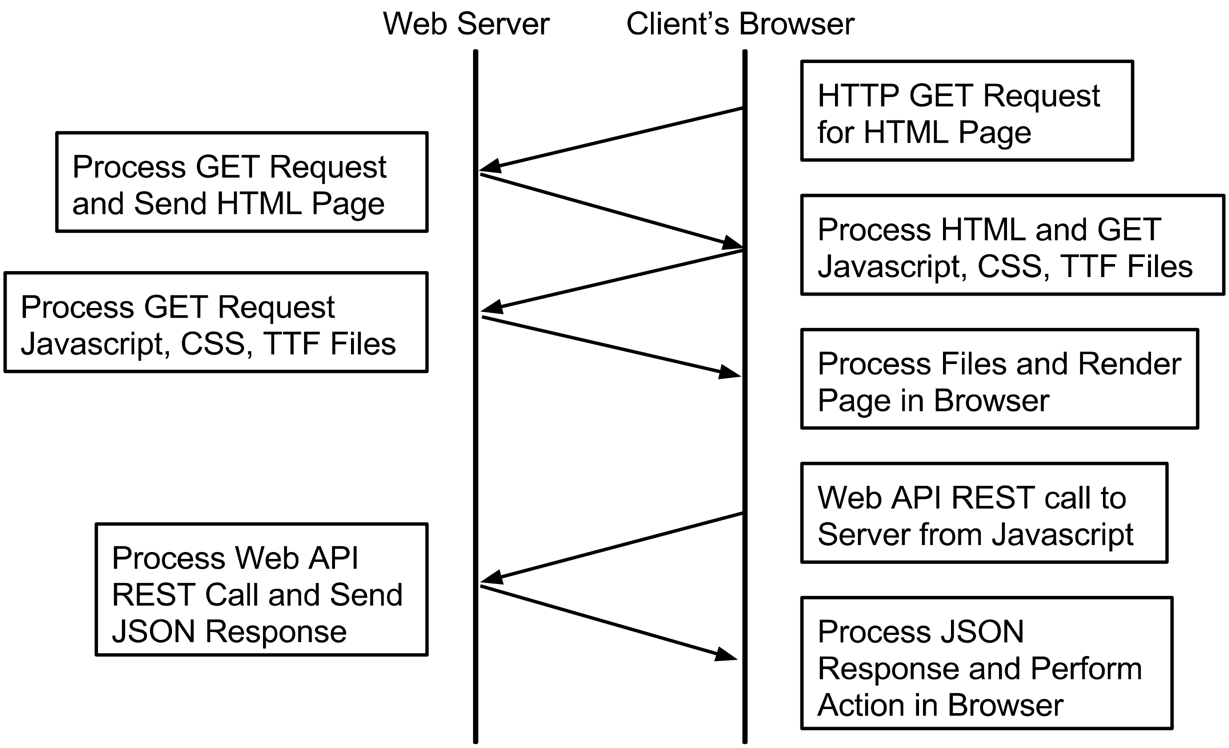


Figure 3: Representative Web Server Request Diagram with API Call

Bill of Materials

Product Description	Unit Cost ($CAD)	Qty	Cost ($CAD)
Terasic Altera DE2 Development Board Vendor: U of A Datasheets: ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE2/DE2_User_Manual.pdf	$495	1	$495
Seco-Larm SK-990AQ Enforcer Electric Door Strike Vendor: http://www.amazon.ca/Seco-Larm-SK-990AQ-Enforcer-Electric-Fail-Secure/dp/B0032UYTQ6/ref=sr_1_10?ie=UTF8&qid=1390854068&sr=8-10&keywords=door+strike Datasheets: http://www.seco-larm.com/pdfs/PI-SD-990A.pdf	$29.95	1	$29.95

PN532 RFID (NFC) R/W Module with Mifare 1k Classic Tag Vendor: http://www.adafruit.com/products/364 Datasheets: http://www.adafruit.com/datasheets/pn532ds.pdf http://www.adafruit.com/datasheets/pn532um.pdf http://www.adafruit.com/datasheets/PN532C106_Application%20Note_v1.2.pdf http://www.adafruit.com/datasheets/S50.pdf	$39.95	1	$39.95
RTC Module BOB-00099 Vendor: https://www.sparkfun.com/products/99 Datasheets: http://www.sparkfun.com/datasheets/Components/DS1307.pdf	$16.68	1	$16.68
2GB Sandisk SD Card	$5.99	1	$5.99
Linksys E1200 Wireless N Router User Manual: http://downloads.linksys.com/downloads/userguide/E_Series_UG_E900Rev_3425-01486_Web.pdf	$39.99	1	$39.99
Ribbon Cable	$0.49	2	$0.98
NPN BJT TIP41C	$0.39	1	$0.39
Diode 1N4002	$0.21	1	$0.21
Zener Diode Fairchild 1N5248B	$0.14	1	$0.14
Resistor 4k7 Ohm	$0.10	4	$0.40
Resistor 2k Ohm	$0.10	1	$0.10

Total Cost	$629.39

Reusable Design Units

SD Controller

Either the University Program SD Card IP Core or the EFSL plus Nios II Endpoint could be used for the Fat file system and SD card access. Based on suggestions in the SD Card Interfacing application notes [2], the EFSL plus Nios II Endpoint was used for SD Card access.

Near Field Communication Library

The initial thought was to integrate an open source NFC library to control the NFC reader/writer module. Both openNFC[22] and libnfc[14] were researched as options. However due to the small NFC requirements of this project and the difficult process involved in porting either library, both libraries were deemed unnecessary. Instead, code was written directly for the popular PN532 NFC chip used in this project to implement only the features required.

Database Library
As registered keys and door event history is logged and stored on the SD Card a database would be useful for managing the data. SQLite [13] is a compact open source database engine library with a C API meant for embedded systems use. Similar to SQLite, Berkeley DB[23] could also be used. In order to keep transactions with the SD card small and limit the size of code, data is instead being stored and managed in delimited flat text files on the SD card. An attempt to integrate SQLite with the NiosII environment ultimately failed, but could potentially work given more time.

I2C Controller
This project used 2 GPIO pins to serve as the SCL and SDA lines in an I2C bus. The GPIO lines were then bit banged in software. The software for reading/writing with I2C was based off code found in the SD Card Audio demonstration project from Altera’s DE2 CD-ROM. Instead, an open source I2C core could have been used to handle I2C. Considering the implementation used was both simple and fast, there was no reason to explore an alternative.

Altera IP Cores
The following default cores provided by Altera’s Qsys program were used:

- Nios II/f core
- SPI (3 wire)
- PIO (GPIO)
- DM9000A Interconnect (Ethernet)
- Ethernet Drivers

Datasheet
The project was built on the Terasic Altera DE2 Board.

Component Voltage Requirements

Component	**Voltage**	**Source**	**Amperage (Sleep, Active)**	**Power Usage (Sleep, Active)**
DE2 Board	9V	Power Supply	700mA	6.3W
PN532 NFC Module	5V	Onboard 5V pin	140 mA	0.7W
Electronic Door Strike	10 - 14Vdc	Bench Power Supply	400mA	0, 4.8W
RTC Module	5V or 3V Lithium Cell	Onboard 5V pin & Lithium onboard	200uA, 1.5mA	0.001W, 0.0075W

Power Consumption

The power consumption of the system has two components the DE2 Board with all attached hardware and the door strike. The values for the board were not measured. The values for the door strike are included in the following table:

Device	Voltage (DC)	Current	Power
Door Strike (Active)	12.0V (Fixed from Power supply)	305mA	3.66W

SD Card Performance

The SD card is used to store the registered keys and history files. While limiting file sizes in an embedded system is considered good practice, file sizes were also limited to increase system performance. The system allows for 20 registered keys and holds the last 200 door events in history. The performance of the SD card was tested by writing 200 history entries to file. This produced an actual file size of 7.63 KB in 3 seconds. The write speed is therefore only 2.54 kBps. To further increase system responsiveness, history is backed up only every 10 minutes.

Web Server Performance

The web server is stateless and sessionless so there are no large memory concerns. However there is approximately 500kB of HTML, CSS, and JavaScript that needs to be loaded initially to populate the website. The browser will be caching the web response of these files after initial load so this isn't a large concern. The webserver runs from flash memory using the rozipfs supplied by Altera and uses the fast CPU core running at 75MHz to enhance performance as outlined in the Experiments and Characterization section.

GPIO protection

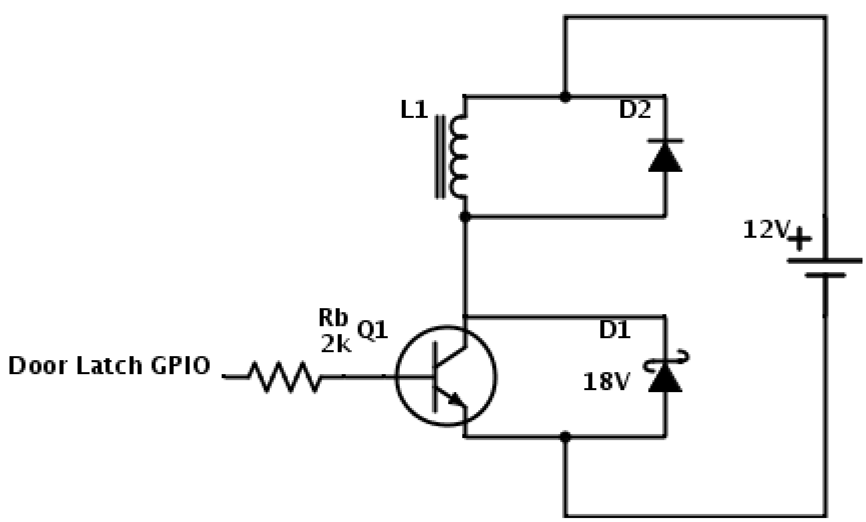


Figure 4: Door Latch Protection Circuit

The BJT is rated for 100 V
Diode D2 is rated for 1 Amp
Zener Diode D1 breaks down at 18V

I2C Pull up Resistors

The 2 I2C busses are open-drain and thus require external pull-up resistors. A single resistor is attached to the SDA and SCL lines. The resistance is = 4.7K

FPGA Pinout Map

Pinout Map:

device	Pinout A (closest to CPU)	Pinout B (furthest from CPU)	interface	interface subname
Electrical	GPIO 0 Pin 29	Common 3.3V	Electrical	Vcc3.3V
Electrical	GPIO 0 Pin 11	Common 5V	Electrical	Vcc 5V
Electrical	GPIO 0 Pin 12	Common Gnd	Electrical	Gnd
Electrical	GPIO 0 Pin 30	Common Gnd	Electrical	Gnd
Ethernet	onboard		Ethernet	
Door bell	GPIO 0 Pin 14		GPIO	
Door bell	KEY3		GPIO	IRQ
Door latch	GPIO 0 Pin 16	Latch pin	GPIO	
Door status	GPIO 0 Pin 26		GPIO	
extra out 1	GPIO 0 Pin 20		GPIO	extra out 1
extra out 2	GPIO 0 Pin 22		GPIO	extra out 2
gpio interrupt 2	GPIO 0 Pin 28		GPIO	
NFC	GPIO 0 Pin 18	NFC Cable A Pin 07	GPIO	RSTOUT_N
NFC	GPIO 0 Pin 24	NFC Cable A Pin 05	GPIO	IRQ
extra	GPIO 0 Pin 09		I2C	sda
extra	GPIO 0 Pin 13		I2C	scl
NFC	GPIO 0 Pin 01	NFC Cable A Pin 11	I2C	MOSI/SDA/TX
NFC	GPIO 0 Pin 03	NFC Cable A Pin 09	I2C	SSEL/SCL/RX
RTC	GPIO 0 Pin 05		I2C	sda
RTC	GPIO 0 Pin 07		I2C	scl
extra	GPIO 0 Pin		SPI	SCLK
extra	GPIO 0 Pin		SPI	MOSI
extra	GPIO 0 Pin		SPI	SS
extra	GPIO 0 Pin		SPI	MISO
SD card	onboard		SPI	SCLK
SD card	onboard		SPI	MOSI
SD card	onboard		SPI	MISO
SD card	onboard		SPI	SS
Wifi	GPIO 0 Pin 15		SPI	SCLK
Wifi	GPIO 0 Pin 17		SPI	MOSI
Wifi	GPIO 0 Pin 19		SPI	MISO
Wifi	GPIO 0 Pin 21		SPI	SS

Wifi	GPIO 0 Pin		UART	Dout
Wifi	GPIO 0 Pin		UART	nCTS
Wifi	GPIO 0 Pin		UART	Din
Wifi	GPIO 0 Pin		UART	nRTS
Camera	onboard		USB	
NFC	Common 5V	NFC Cable A Pin 01		5.0V
NFC	Common Gnd	NFC Cable A Pin 03		GND
NFC	n/c	NFC Cable A Pin 13		MISO
NFC	n/c	NFC Cable A Pin 15		SCK
NFC	Common 3.3V	NFC Cable A Pin 17		3.3V
reset	KEY0			
None	KEY1			
None	KEY2			
LED	onboard		GPIO	Red LED
LED	onboard		GPIO	Green LED

IRQ Hardware Priorities.

Software Device	**IRQ Device**	**IRQ**
ucos Timer	Interval Timer	0
second Timer	Interval Timer	1
JTAG Uart	JTAG Uart	3
USB hc	USB	4
USB dc	USB	5
Ethernet	Ethernet	6
Wifi	SPI	7
SD	SPI	8
Extra	SPI	9
Wifi	UART	10
doorbell	GPIO doorbell	11
NFC interrupt	GPIO int0	12
Doorstatus	GPIO int1	13
GPIO int2	GPIO int2	14
extra 0	GPIO extra 0	15
extra 1	GPIO extra 1	16
extra 2	GPIO extra 2	17

Background Reading

Representational State Transfer (REST)

REST is a design paradigm that allows data to be transferred between systems and was created by Roy Fielding [29]. It is commonly applied to websites and services and relies on some functions of HTTP. REST was chosen to unify the models between the web client in JavaScript and the web server in C.

NFC Technology and Mobile Phone Services

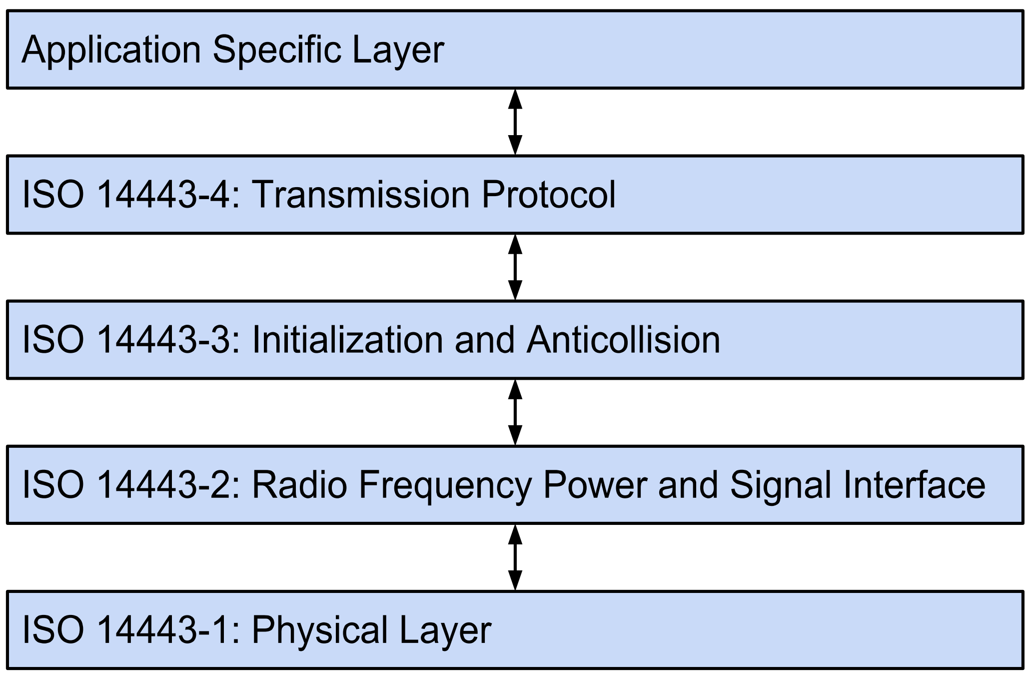
The paper “NFC Technology in Mobile Phone Next-Generation Services” by Aziza, H. [26] presents a mobile phone application that uses NFC tags to communicate with a remote social networking service. The system uses the tags to identify an entity and its location which is then sent to the social network service with additional action commands. Such a system allows the benefits of NFC tags to be realized without relying on a large infrastructure to be built on the NFC devices. A similar system could be used to authenticate the smartphones used as keys in our NFC smart door if they are on the same wireless network, or even if they are both internet enabled. Such a system could allow the NFC authentication to be an extension of future HTTPS web authentication unifying the systems.

Long-range NFC Reading

Long-range NFC was researched and it was determined that it’s not feasible due to the size and required orientation of the antenna [11]. It states that the read range is proportional to the antenna size. Using sample value, for example, reading a perfectly positioned tag from 1m away, would require a coil with a radius of 1.41m (~3m diameter) which is unfeasibly large. Increasing the power of the remote NFC tag would not be feasible without potentially damaging the device, so increasing the power in an attempt to increase range is also not an option. The research paper does not.

NFC Interfacing and ISO 14443

The NFC standard that we plan to use is ISO 14443. It consists of 4 different layers that handle the physical[17], RF[18], initialization[19], and transmission[20] of data using the protocol. The application specific communication is not provided by ISO 14443 however. Since different devices will use different application specific communications, it is not feasible to have support for all of them. Also, adding application specific communications would not be possible in some cases if there are proprietary systems that must be licensed or registered with. For example, in order to interface with Visa payWave[21], the following procedure must be followed in order to gain access to the SDK and communicate with the cards. The goal is to have universal communication with NFC tags so this would not be feasible



.

Figure 5: NFC 14443 Communication Stack

Android NFC Interfacing

Unfortunately, Android devices return a randomized Unique ID (UID) unless application code is written using Host Card Emulation[12]. HCE allows an Android application to emulate an ISO 14443 card and communicate with an NFC reader using ISO 7816 Application Protocol Data Units (APDUs). An HCE application runs as a service on the Android phone. When an NFC reader initiates communication with the phone, the CPU directs traffic to the selected application. Applications are selected using a Select Application ID (AID) APDU where the AID is defined for the given application. All communication is then routed to this application for the duration of the transaction.

Software Design

Overview

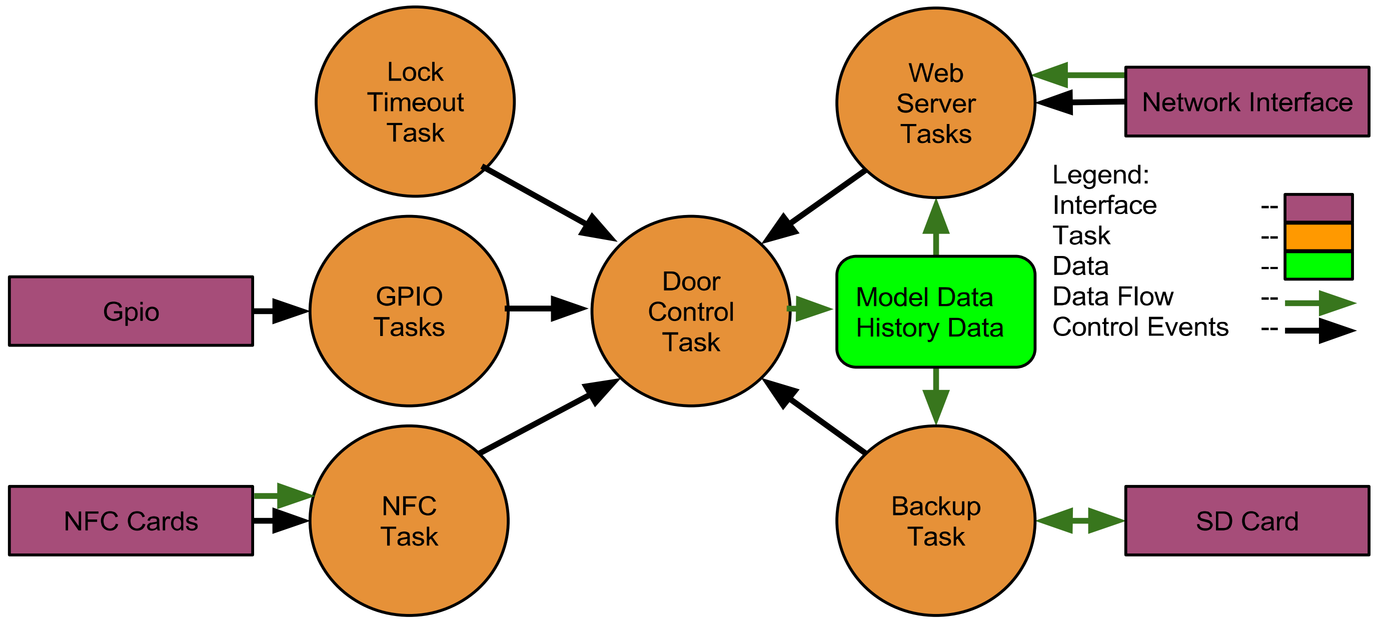


Figure 6: Task and Data Interaction

The main software is event driven. The door control task will handle all events triggered by the surrounding events. All other tasks will feed the door control task events to register and process. Model and history data is mainly compiled by the door control task and is used by the web server tasks to serve content to the door administrative website. In the background, the backup task is responsible for periodically backing up data to the SD card.

Model Data

The model is a globally accessed struct containing the current status of the door, valid keys, invalid keys available for registration, and the last 200 history events. Access to the model is controlled using a mutex since multiple tasks will be reading and updating the data model. The DoorModel and NFCKey structs used are defined as:

```
typedef struct DoorModel{
    /* Control task info */
    NFCKey lastNFCKey;
    NFCKey lastWebKey;

    /* Door status */
    bool doorOpened;
    bool doorUnlocked;

    /* Memory array of keys */
    int keyCount;
    NFCKey keys[MAX_KEYS];
```

```
        /* Last Access History */
        int historyCount;
        StatusHistory* lastAccess;
        StatusHistory history[MAX_HISTORY];
    } DoorModel;

    typedef struct NFCKey{
        char id[MIFARE_CHAR_LEN];
        char note[MAX_CHAR];
        bool isRegistered;
    } NFCKey;
```

History Data

The following struct is used to define a history entry:

```
    typedef struct StatusHistory{
        struct StatusHistory* nextNode;
        NFCKey key;
        time_t timeStamp;
        char eventSource[MAX_EVENT_SOURCE_CHAR];
        char eventOutcome[MAX_EVENT_OUTCOME_CHAR];
    } StatusHistory;
```

Door Controller Task

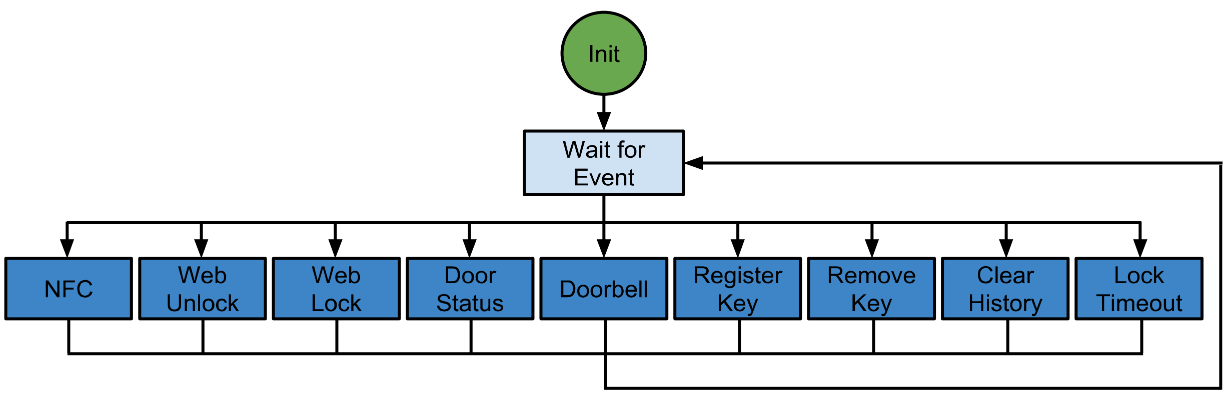


Figure 7: Door Control Flow Diagram

The door control task is controlled by a message queue. The queue resolves into an enum of possible events in which the door control task will execute in order. An easy to use event registrar is provided for other tasks to queue new events. The use of the door controller task is used to centralize all event handling and keep code easy to maintain. Events can be easily modified and added when necessary.

A switch statement is then used to determine how to process the current event as shown in the figure above.

Event	Trigger	Operation
NFC	NFC task, card scanned and identified	Will complete an NFC interaction. Will unlock or register key depending on key validity. A history of the event is then saved
Web Unlock	Website Unlock	Will unlock door and log an entry of the event.
Web Lock	Website Lock	Will lock door and long an entry of the event.
Door Status	Door Status Interrupt	Determines the new door status and records the history event and sets status LEDs
Doorbell	Doorbell Interrupt	Saves doorbell event into history
Register Key	Website button	Will register the desired key specified by the website to be a valid key
Remove Key	Website button	Will remove the desired key from the list of valid keys
Clear History	Website button	Will clear all history from data.
Lock Timeout	Lock Timeout has occured	Will lock doors, set status LEDs and update model

NFC Interfacing Task

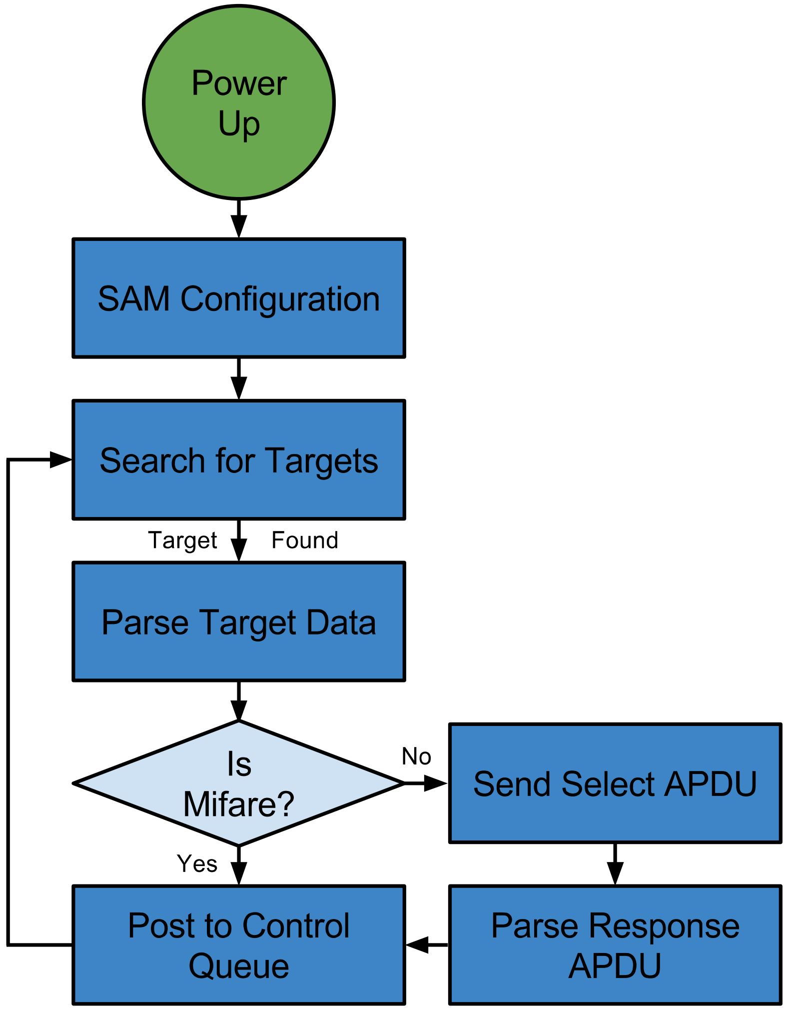


Figure 8: NFC Flow Diagram

The PN532 NFC module includes an IRQ pin. The IRQ is pulled low to alert the host that the PN532 has data to transmit in response to the last command received. The ISR associated with this IRQ will post to a semaphore used by the NFC task to wait for responses. This task uses code written for the PN532 to tell the module to look for NFC targets. When a target is found, the IRQ fires and the task reads back information about the target. This information includes data responses outlined in the NFC Forum specification such as the Select Acknowledgement and the device UID. If the sixth bit of the Select Acknowledgement is a 1, then the device is fully ISO 14443 compliant and is assumed to be an Android device. Otherwise, the device does not fully comply to the ISO 14443 protocol and is assumed to be a MIFARE card. These are the two types of keys supported by the system. A MIFARE UID is used as a MIFARE key identifier. However, since Android devices return a random UID, an additional command is sent to our app running on the target Android device. The Android app will then return the device's 8-byte unique Android ID. This will be used as the identifier for Android keys. The NFC task then posts to the door control queue to process the key.

GPIO Tasks

This project contains 2 GPIO tasks to control debouncing and prevent flooding to the history data. The GPIOs are in its separate task to prevent blocking on the door control task. When the GPIO interrupts are fired, a wait time will be triggered before the GPIO can be read. This is to ensure a proper read of the GPIO. A door control event is then sent to the door control task to handle the event.

The 2 GPIO are:
- Door Status, with a debounce time of 500 us
- Doorbell, with a debounce time of 100us

Backup Task

To provide data persistence, the keys and history are backed up to the SD Card and can be restored upon initialization. The backup task is responsible for saving the model and history data to the SD card periodically. The backup period is set to 5 minutes.

Web Server Tasks

The web server tasks consist of 2 main web tasks and other Interniche specific tasks. The Interniche Stack was not modified and is outlined in their documentation [28]. The two web server tasks are the WSInitialTask() and the WSTask(). The WSInitialTask() initializes the web server and starts the WSTask() which is the main web server task that accepts incoming HTTP socket connections on port 80. The web server and web api component are outlined below in their given sections.

Hardware Interfacing

The following table outlines the software drivers and subsystems created to communicate with particular hardware elements:

Hardware Element	Software Description
Ethernet	Provides an interface for the dm9000a for the webserver to use.
I2C	Provides I2C interface for the RTC and NFC modules.
GPIO	Provides GPIO interfacing. Includes interfaces for door status, doorbell, door latch, registration switch, NFC interrupt pin and the red and green status LEDs.
RTC	Provides access to and control of the RTC module.
NFC	Provides basic PN532 communication and sam configuration.
SD	Handles the SD card interfacing and file system functionality.

REST Interface

A Representational State Transfer (REST) architecture is a web based system for transferring data between internet connected servers and clients. For the scope of this project a standard client-server model will be used. REST consists of a standardized interface that allows clients to synchronize and change the state of data on the server using HTTP packets. The methods implemented are not fully symmetric because the web interface does not need full model control. All responses return JSON objects that match the model data used internally.

The following table outlines which REST interfaces were implemented:

Model Type	Method	Example URL (ip/api/…)	Function
Status	POST	/api/status	Lock/Unlock the door
Status	GET	/api/status	Get the lock and door status
Status History	GET	/api/statushistory	Get all history
Status History	DELETE	/api/statushistory	Delete all history
Keys	GET	/api/key	Get all keys
Keys	DELETE	/api/key?id=[#id]	Delete 1 key with specified id
Keys	POST	/api/key	Update 1 key with specified id
Image	GET	/api/image	Gets a new image from the camera

Webserver

The Web Server is based on the embedded MP3 player project from Winter 2013 [3] which is based on the NIOS II web server template from Altera. This streamlines the integration and design of the web component which will speed up development through code reuse. The server was modified to support REST interfaces with PUT, DELETE, and OPTIONS methods as needed. The existing music player API was removed and replaced with the REST Door API outlined above.

Web Interface JavaScript Architecture

The following web libraries were used:

Library	Function	License	Version
Bootstrap	Web UI Framework	MIT	3.0.3
Knockout.js	Javascript MVVM	MIT	3.0.0
jQuery	Javascript Library	MIT	2.1.0
Dark Bootstrap Theme	Theme	Apache 2.0	3.0.3

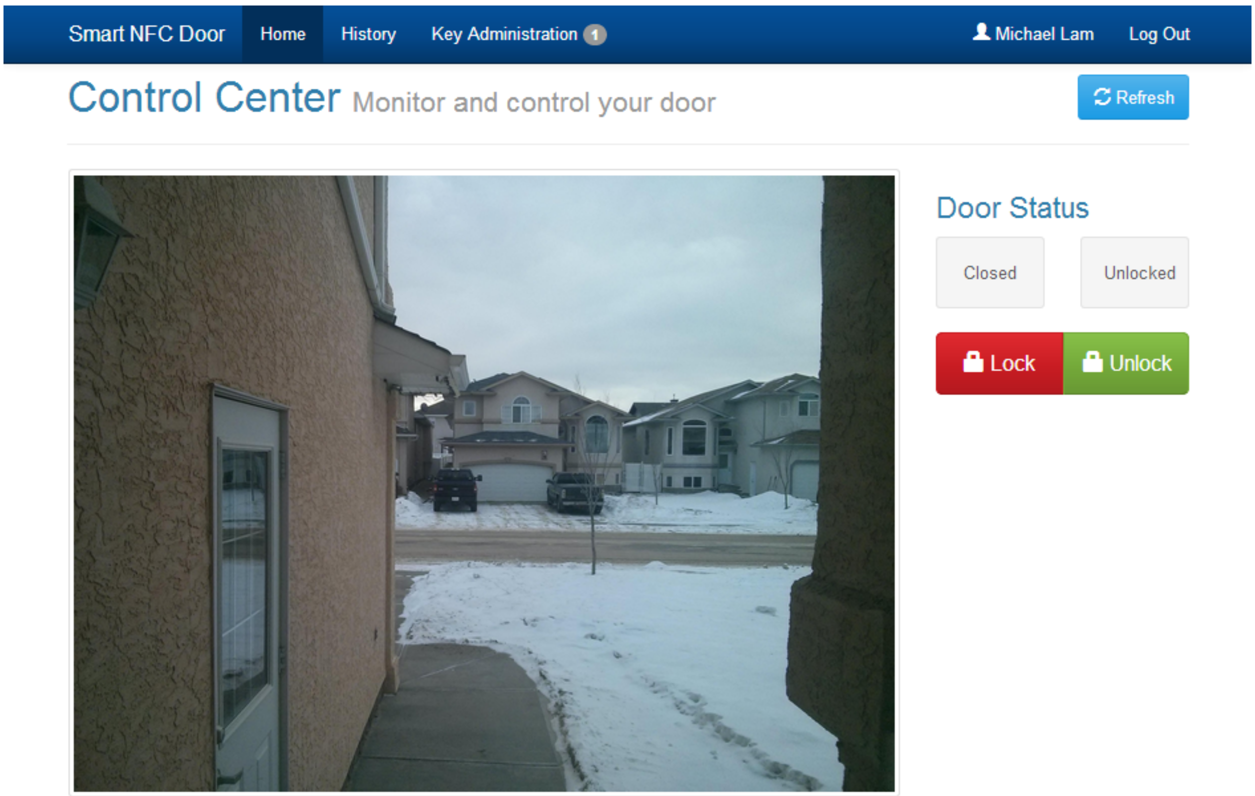
The web interface uses jQuery as a base with the Bootstrap and Knockout.js libraries utilizing it. jQuery is JavaScript library with many helpful functions that helps streamline and simplify JavaScript web development. Bootstrap is a UI framework that includes reuseable CSS styles and JS utilities that aid in web UI design. The UI uses default Bootstrap elements and integrates them into a simple but functional design. Knockout.js is a Model-View-ViewModel JavaScript library that helps with viewmodel based development which is the core of REST web interfaces. The Knockout viewmodel is a direct mapping of the model implemented on the DE2 board. It is synchronized and controlled through the REST interface and is used by the UI to display and update information from the model.

Web User Interface Design

The web interface is designed for ease of use and is platform independant utilizing HTML5 and CSS3 web standards. It is also responsive to the width of the browser which means full mobile support with any HTML5 browser. There are two sets of screens, the old and final screens. The old represents the first iteration of the UI and features a camera image placeholder. The final screens are of the final UI which has the same general layout, some new features, and no camera image placeholder to account for the removal of the camera from the project scope. The following screens are outlined below:

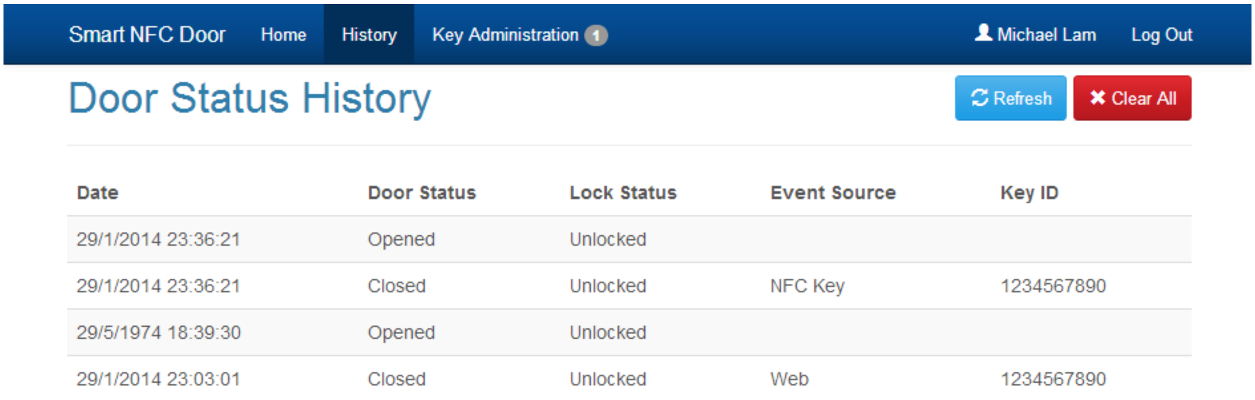
UI Screen 1: Old Main Page

This page contains the latest web camera image which is refreshed on page load, and the status of the door. It also has buttons to lock/unlock the door.



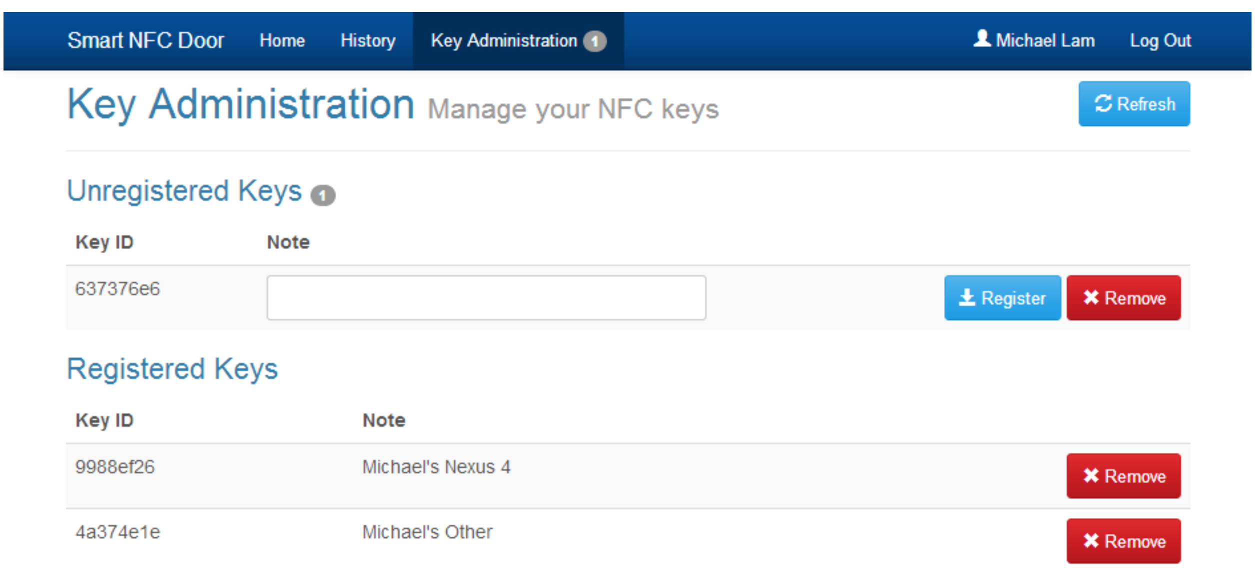
UI Screen 2: Old History Tab

This page contains the history list of the door's last statuses. It also contains a button to remove all history entries.



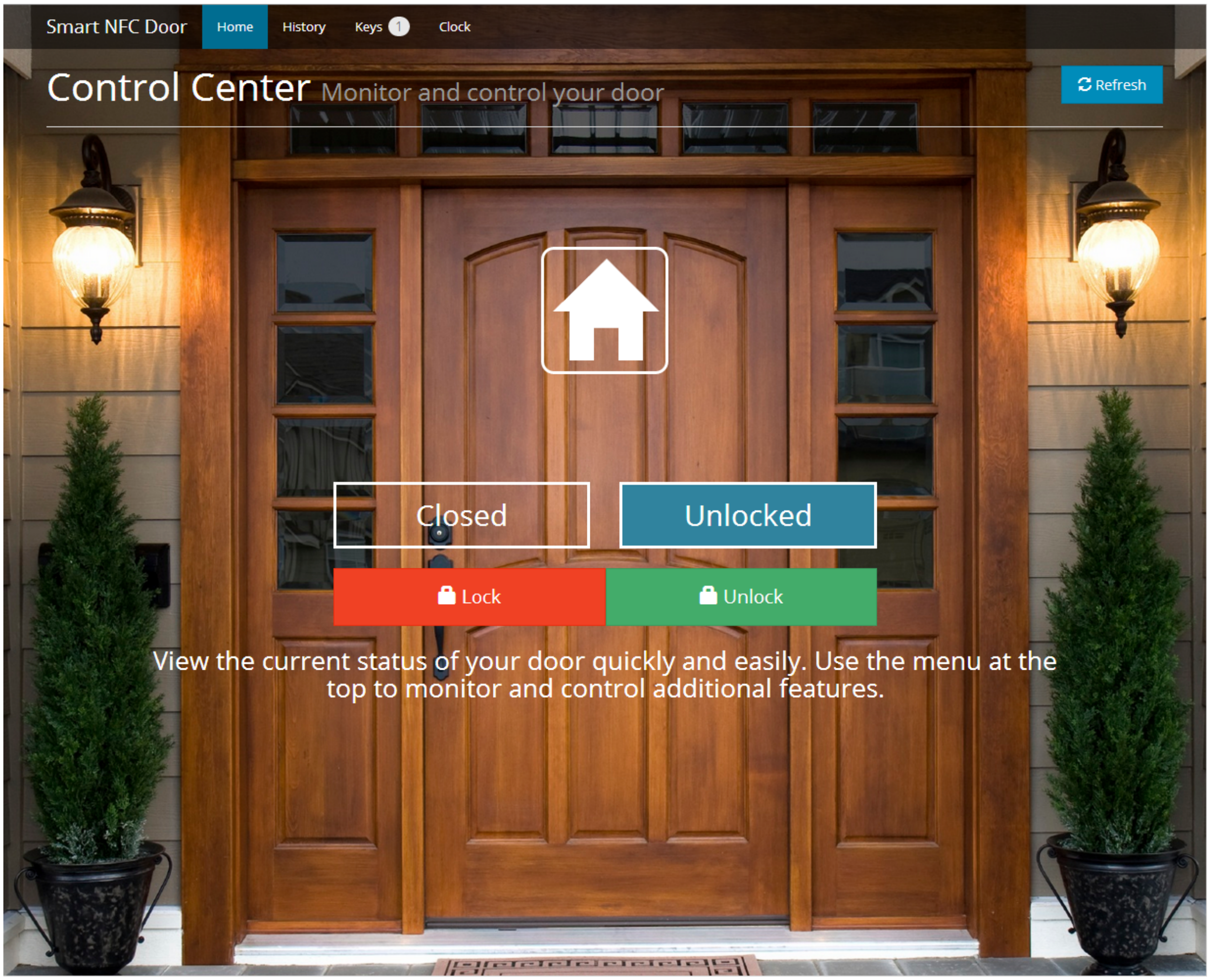
UI Screen 3: Old Key Administration

This page contains the list unregistered and registered keys. It also contains buttons to register, or remove those keys.



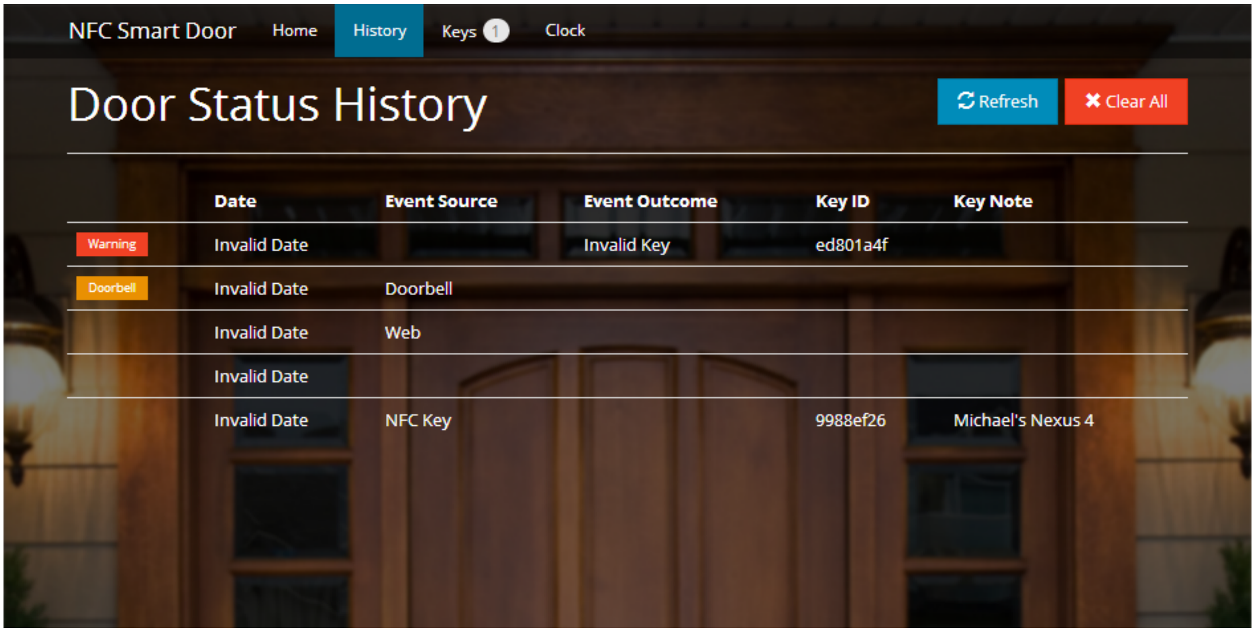
UI Screen 4: Main Page

This page contains the status of the door and buttons to lock/unlock the door.



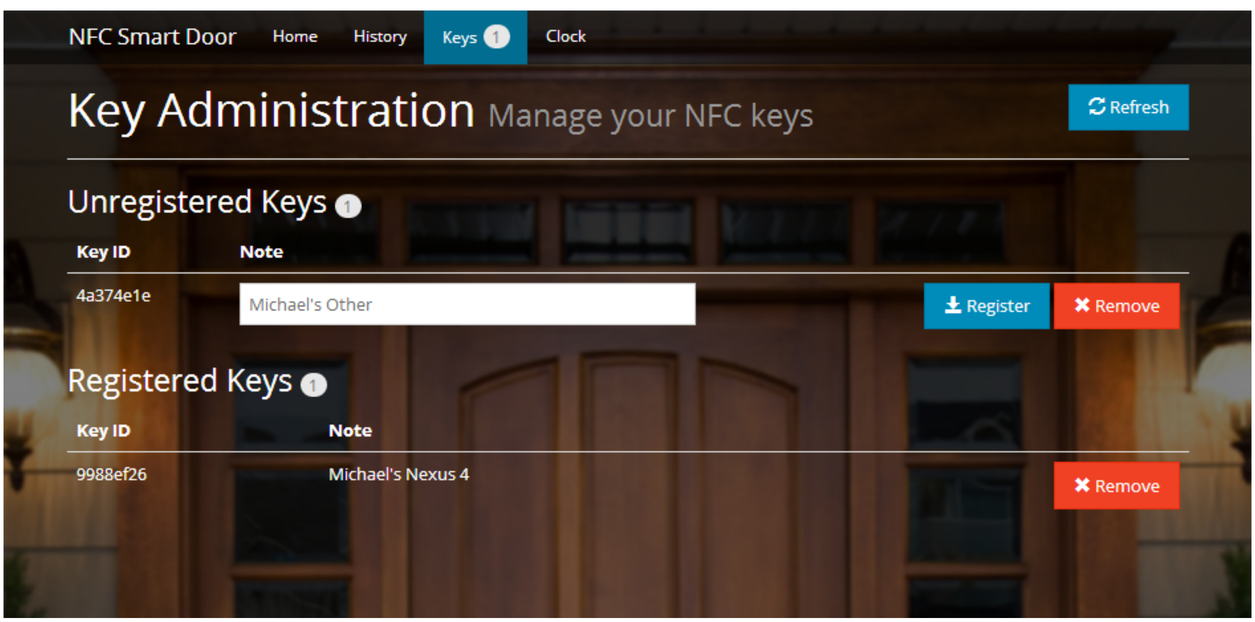
UI Screen 5: History Tab

This page contains the history list of the door's last statuses. It also contains a button to remove all history entries.



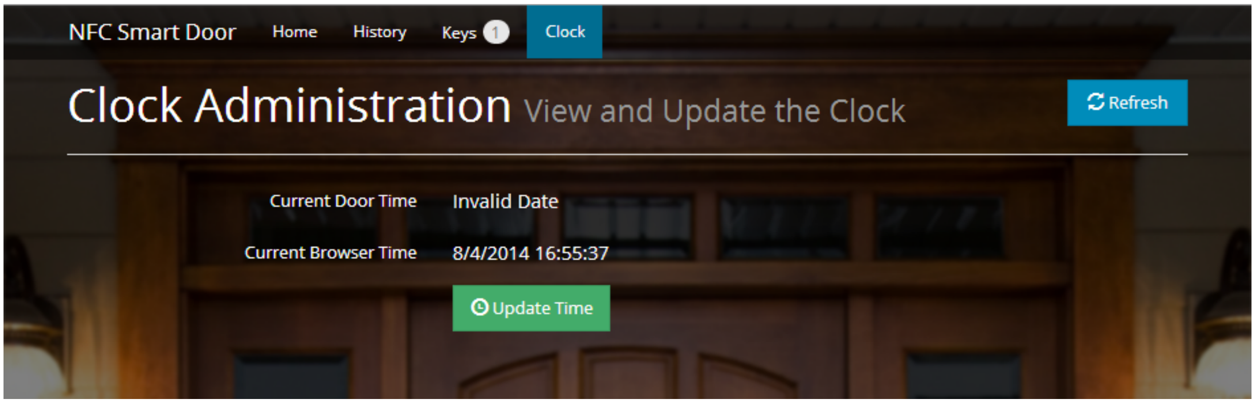
UI Screen 6: Key Administration

This page contains the list unregistered and registered keys. It also contains buttons to register, or remove those keys.



UI Screen 7: Clock Administration

This page contains the current door time, the current browser time and a button to update the board time which will resynchronize the board clock.



Android Application

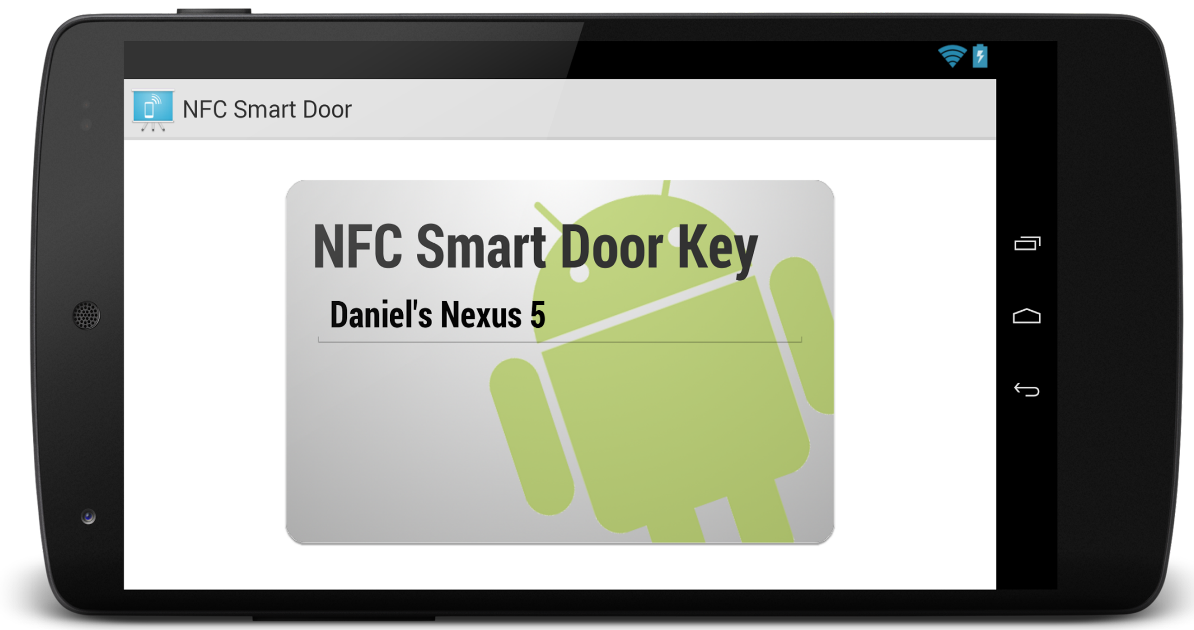


Figure 9: NFC Smart Door Android App Preview with Phone View

The android application is based on the Card Emulation Sample application by Google [30]. It has modifications for work with the system and sends android IDs which are used as keys. There are also modifications to remove the debug log, change the app name and add a name field for easy identification.

Test Plan

Hardware and Device Interface Testing

A test framework was written to unit test IO hardware in a single project. As components were built, the multimeter was used to ensure proper connection between hardware components.

Using test.c and disabling main.c via a macro define, hardware components can be unit tested independently. The following test cases were created and run successfully:

1. Door Open Detector (GPIO)
    a. Double-edge interrupt and ISR function correctly
2. Door Bell (Onboard key)
    a. Interrupt and ISR function correctly
3. Electric Door Strike
    a. Lock state of the electric door strike controlled correctly
4. Ethernet
    a. Successfully served custom 404 page defined in the web server code
5. RTC Module
    a. Time was set and read back by the second correctly
6. NFC Module
    a. Properly initialized with ISO 14443 card UIDs read back correctly
7. SD Card
    a. Files removed and created on the SD card correctly
8. Registration Switch (Onboard switch)
    a. On/Off position read correctly
9. USB Camera
    a. USB controller registers can be accessed
10. DE2 Red and Green LED
    a. LEDs were turned on and off in the correct manner.
11. A Heartbeat task with a low priority was created to ensure the system had not crashed

Software Testing

Events such as door open and NFC tag reading triggered LEDs in the door control task. The LEDs give confirmation that the main door control task is running and accepting tasks.

Web Server Testing

Web Server Testing was completed through the integration of ethernet as a unit test. Full web GET request tests were performed in addition to full REST API testing.

JavaScript Testing

Javascript unit testing was accomplished using the Knockout.js viewmodel and redirecting the AJAX api calls to a test function. The test function would use 2 sets of test data to simulate data updates, additions and deletions from the viewmodel independent of a web server. All of the computations are performed on a client machine and are highly susceptible to changes in hardware, software and browser versions on the device so performance measurements were not taken. Additionally the JavaScript is not computationally heavy and did not show slow performance throughout any of the testing so there shouldn't be any issues with performance.

Experiments and Characterization

Android NFC Experiments

Basic technological feasibility experiments were performed with an Android Smartphone as an NFC reader. The smartphone used was a Nexus 4 with 4.4 KitKat that is compatible with ISO 14443-3 tags. The tag was successfully read and interfaced. The tag contains an application specific interface which could not be interfaced without reverse engineering, but the tag itself contained a static UID (4 bytes) that could be used for identification. This experiment shows that it is possible to interface and identify relatively unique ISO 14443-3 tags but security could not be guaranteed using this method.

SD Card and Database Characterization

Tradeoffs between storing data as delimited flat text files versus using a database such as SQLite or Berkeley DB were investigated. While using a database to store and compare keys would likely be faster for a large set of keys, the number of keys was limited to 20. Furthermore, the size of the database would add overhead when backing up to the SD card. Since SD card performance is already very slow, delimited flat text files were used. Integrating SQLite or Berkeley DB would also take significant effort and therefore this design decision makes sense in terms of effort, size and speed.

CPU Speed and Web Server Performance

Initially, certain processes such as web page delivery took far too long. It took approximately 90 seconds to deliver the web page from the SD card to a client. This long wait time was unacceptable, thus investigation to improving the upload rate was performed.

The following configurations were done to improve performance.

Load Times for the Smart Door web application (917KB)

Core type, Core speed, Content location	Page Load time and Results
NIOS II Economic, 50Mhz, SD card	90 Seconds = 10.2 KB/s
NIOS II Economic, 50Mhz, Flash memory	32 Seconds = 28.6 KB/s
NIOS II Fast core*, 50Mhz, Flash memory	18 Seconds = 50.9 KB/s
NIOS II Fast core*, 100Mhz, Flash memory	Board does not run software.
NIOS II Fast core*, 75Mhz, Flash memory	12 Seconds = 76.3 KB/s

*The Fast core also has 8kB data and instruction caches, and burst mode enabled.

Running the NIOS II fast core improves page load times, however the quicker core speeds may have introduced system instability.

GPIO Debouncing and Timeout

Physical GPIO switches such as the doorbell and door status were prone to bouncing signals. Thus to prevent multiple event triggers and to prevent blocking on the main event handler, each Physical GPIO were placed into a separate task.

Bouncing Times for Doorbell and Door Status

GPIO	Maximum measured bounce time.
Door status	189 us
Doorbell	20 us

To ensure proper GPIO values were measured, additional time was added to the measured debouncing. In the software, Door status has a timeout of 500 us and 100us for the doorbell.

GPIO Door Status Characterization

In order to control the door latch, the GPIO must be able to control 12V at 400mA. This is done by controlling the latch using a BJT. Configured as a Common emitter amplifier, the configuration will amplify input voltage. In order to reduce GPIO current as well as provide the proper level shifts, the base resistor is set at 2k Ohm. Experiments show that the latch turns on when base voltage rises beyond 2.4V and turns off when base voltage falls below 1.8V. These voltage provide a reasonable threshold between the GPIO high of 3.3V and low of 0V. Additionally, since the BJT is driving an inductive load, protection diodes have been installed. A diode parallel to the door latch is rated 1 Amp and will dissipate any voltages higher than 12V. A Zener diode is also parallel with the BJT for added protection. The BJT is rated for 100V, while the Zener diode will breakdown at voltages greater than 18V. See the data sheet for Door status wiring diagram.

Safety

RTC Module - Contains a coin battery - (3V). If the battery is puncture or broken acid could be spilled on the user.

Door - Potential pinching hazard. Users will exercise caution to not close doors on others or their own appendages.

Environmental Impact

The project does not contain any specific hazardous materials for operation. The following list outlines which devices are RoHS compliant, and which are not.

RoHS Compliant	**Non-Compliant**
PN532 NFC Module	Altera DE2
RTC Module	Resistors
Door Strike	Solder

Sustainability

The project is currently separate IO components that do not function together. Therefore power consumption will be assumed to be the sum of each individual component's power usage in sleep or active mode.

Sleep mode represents the lowest power state that the system can be in while still responding to input requests. For simplicity, the active state will be the normal event of scanning a key tag, opening the door, and taking a picture. The duty cycle will be approximately 6 events/day.

Power Sleep = 6.3 + 0.7 + 0.001 = 7.001W
Power Active = 6.3 + 0.7 + 0.0075 = 7.0075W
Power Door Strike = 3.66W

An event will run for approximately 1 second, and afterwards, the door strike will be powered for 15 seconds to unlock the door. Therefore the average power usage in a day would be:

3600s/hour sleep - 6s/hour active = 3554s/hour sleep
Average Power Usage =(3554s * 7.001W + 6events * (1s*7.0075W+15s*3.66W))/3600s = 7.0147Wh

In Edmonton, it costs 8.7cents per kWh. Assuming the device is running for 1 year:

24hours/day * 365days/year * 7.0147Wh=61.449kWh/year
8.7cents * 61.449kWh/year= $5.35/year

In Edmonton the majority of electricity is coal generated which produces 0.989kg CO_2/kWh.

61.449kWh/year * 0.989CO_2/kWh = 60.77kg CO_2/year

References

[1]	B. Jongerius, M. Jun and S.Hewson. (2013). I2C Device Integration [Online] Available: http://www.ece.ualberta.ca/~elliott/ece492/projects/2012w/g4_motion_detection/
[2]	J.Brown, and B. Thornton (2013, February 21) SD Card Interfacing [Online] Available: https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/SD_card_interfacing/
[3]	J.Brown, and B. Thornton. (2013, February 21) Network-Controllable Embedded MP3 Player https://www.ualberta.ca/~delliott/local/ece492/projects/2013w/g12_EmbeddedMP3/
[4]	T. Kaddoura and J. Nahar. (2013, February 27) DM9000A Ethernet Controller Application Notes https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/Ethernet_DM9000A/
[5]	T. Kaddoura and J. Nahar. (2013, February 27) ISP1362 USB Controller Application Notes https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/USB_ISP1362/
[6]	R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, Univ. California, Irvine., 2000.
[7]	M Otto, and J. Thornton, et al. (2014, January 20) Bootstrap [Online] http://getbootstrap.com/
[8]	S. Sanderson (2014, January 20) Knockout.js [Online] http://knockoutjs.com/
[9]	The jQuery Foundation (2014, January 20) jQuery [Online] http://jquery.com/
[10]	T. Park. (2014, January 20) Bootwatch [Online] http://bootswatch.com/
[11]	Y. Lee, "Antenna Circuit Design for RFID Applications", Microchip Technology Inc. 2003. http://ww1.microchip.com/downloads/en/AppNotes/00710c.pdf
[12]	"Host-based Card Emulation" Google. [Online] http://developer.android.com/guide/topics/connectivity/nfc/hce.html
[13]	R. Hipp, D. Kennedy, and J. Mistachkin. (2013, December 6) SQLite [Online] http://www.sqlite.org/
[14]	Libnfc Developer Community. (2013, September 3) Libnfc [Online] http://nfc-tools.org/index.php?title=Libnfc
[15]	L. Ysboodt, and M. De Nil. (2013, April 24) Embedded File Systems Library [Online] http://sourceforge.net/projects/efsl/
[16]	M. Troccoli. (2007, November 7) Nios2 Endpoint [Online] http://www.ohloh.net/p/efsl/commits/68269113
[17]	Intl. Standards Org. (2013, Sept. 19) ISO/IEC 14443-4:2008 Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 1: Physical characteristics http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?

	csnumber=39693
[18]	Intl. Standards Org. (2010, Aug. 19) ISO/IEC 14443-2:2010 Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 2: Radio frequency power and signal interface http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50941
[19]	Intl. Standards Org. (2011, April 12) ISO/IEC 14443-3:2011 Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 3: Initialization and anticollision http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50942
[20]	Intl. Standards Org. (2013, Dec. 18) ISO/IEC 14443-4:2008 Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 4: Transmission protocol http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50648
[21]	Visa. (2014, Feb 9) Visa payWave for Mobile https://developer.visa.com/paywavemobile/docs
[22]	The Open NFC Project. (2013, April 25) openNFC [Online] http://open-nfc.org/wp/
[23]	Oracle. (2013, May 31) Berkeley DB [Online] http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index-085366.html
[24]	L. Fried (Ladyada), and K. Townsend. (2013, October) Adafruit NFCShield I2C [Online] https://github.com/adafruit/Adafruit_NFCShield_I2C
[25]	Nathan Seidle (Sparkfun.com). (2004, Nov. 23) Example 16F88 Code [Online] http://www.sparkfun.com/datasheets/Components/rtc-demo.zip
[26]	Aziza, H., "NFC Technology in Mobile Phone Next-Generation Services," *Near Field Communication (NFC), 2010 Second International Workshop on* , vol., no., pp.21,26, 20-20 April 2010 doi: 10.1109/NFC.2010.18
[27]	Web UI Door Background Image [Online] http://www.interiordev.com/imagedir/awesome-wooden-front-door-with-fancy-two-lamp-on-the-wall.jpg
[28]	Altera. “Ethernet and the NicheStack TCP/IP Stack - Nios II Edition” http://www.altera.com/literature/hb/nios2/n2sw_nii52013.pdf
[29]	Fielding, R.T.; Taylor, R.N., "Principled design of the modern Web architecture," *Software Engineering, 2000. Proceedings of the 2000 International Conference on* , vol., no., pp.407,416, 2000 doi: 10.1109/ICSE.2000.870431
[30]	Google. “CardEmulation” https://developer.android.com/samples/CardEmulation/index.html
[31]	Altera. “DE2_NIOS_HOST_MOUSE_VGA”

Appendices

A. Quick Start Manual

1. Program the board with the .pof/.sof file as outlined in the ECE 492 tutorials.
2. Program the website found in “~/software/system/WebUI.zip” into flash memory with memory offset 0x200000.
3. Open the NIOS II Eclipse IDE and import an existing project. Choose the project folder “~/software/NFCProject/”.
4. Follow the wizard entering in values as outlined in the ECE 492 tutorials.
5. Add a new NIOS II Board Support Package using the “~/niosII_system.sopcinfo” file.
6. Name the BSP project to NFCProject_bsp.
7. Right click on the bsp project and click NIOS II -> BSP Editor.
8. Go to the software packages tab and enable the “altera_ro_zipfs” and “altera_iniche” components.
9. The project is setup to use a static IP of 192.168.1.111 on the 192.168.1.0/24 subnet. If needed, change the static IP or remove it completely to fallback to DHCP by editing “~/software/NFCProject/webserver/web_server.h” line 64 to 77.
10. Build both projects.
11. Right click on NFCProject and choose Run as -> NIOS II Hardware.
12. Visit the site using a network connected device by entering the IP into the browser.

B. Future Work

The following is a list of future additions to the project:

Feature	Description
More statistics and analytics	In addition to history, have a statistics page that outlines information and graphs with relevant data that users would want to see.
Camera with live streaming	Initial plans were to use a camera for images. This is still a potential feature and could be extended to live streaming.
HTTPS and NFC security	Standard HTTPS and NFC security could be implemented.
User accounts	User accounts could be implemented to allow keys to be linked to uses and users to be allowed different levels of access.
Time restricted access	Using the current time of day and a predetermined schedule, restrict or allow access to a key.
Google Glass Integration	Since the JSON REST API is accessible from any web enabled device on the same network, google glass integration could be as simple as making an app that uses the API directly.
Pet door support	Develop a version of the system to work with pet doors and pets.

C. Hardware documentation

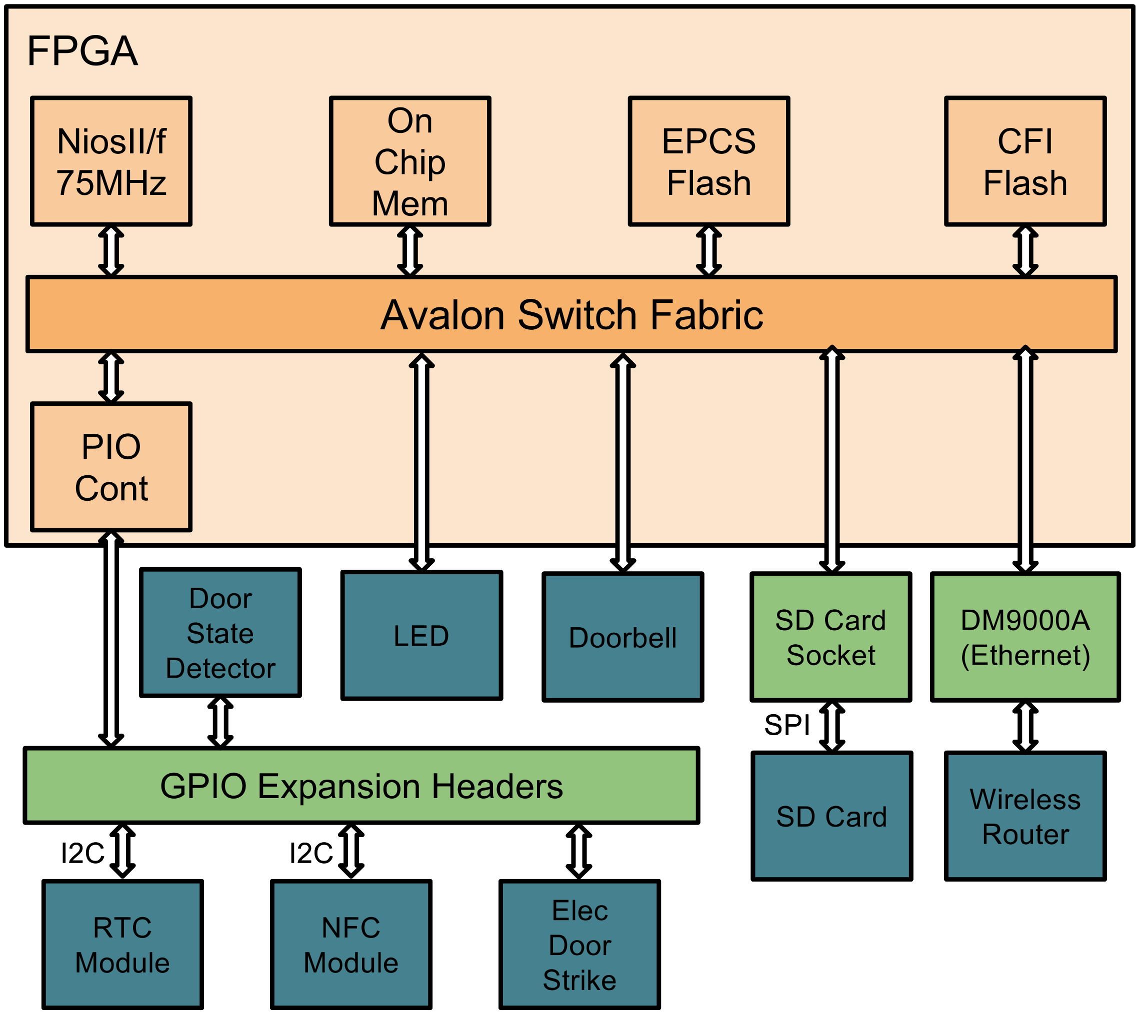
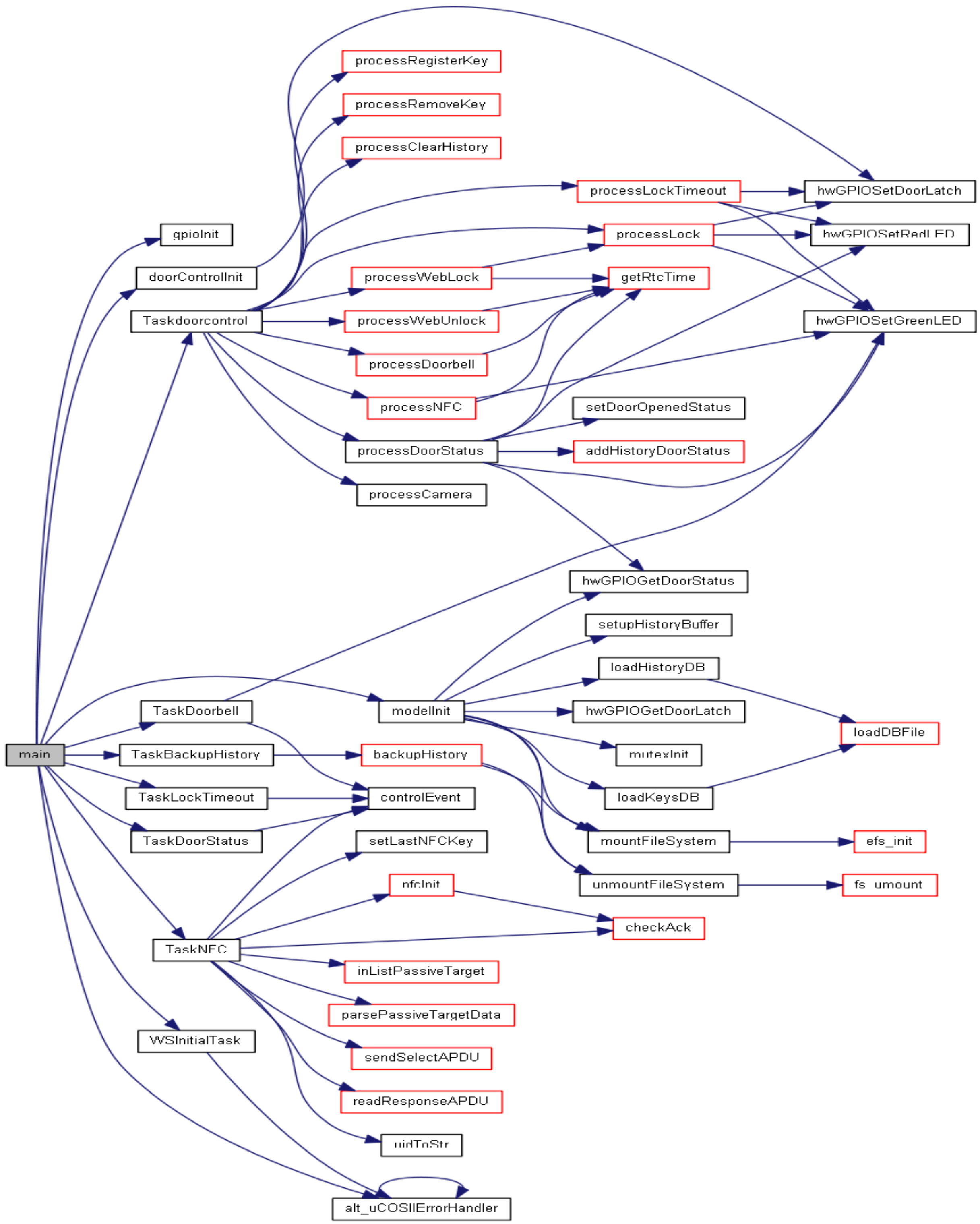


Figure 10: Hardware Block Diagram

D. Source Code

The project software has been Doxygened. See software/NFCProject/Doxygen/html/files.html for full call stack. This is the main() call stack.



The files listed below are authored solely by us or else the original author will be mentioned. Any files not mentioned were not authored by the group.

The Status legend is {**N**ot compiled successfully, **C**ompiled without errors, **E**xecuted or otherwise demonstrated, **T**ested and passed}.

Android Application

The android application is based on the Card Emulation Sample project from Google [30]. There are 34 files and listing each of them is unfeasible. The following list contains all of the modified files:

File	Description	Status
src\main\res\values\strings.xml	String resource file, added some fixed strings, removed others that weren't needed.	T
src\main\res\values\base-strings.xml	Other string resource file, changed the app name	T
src\main\java\com\example\android\cardemulation\AccountStorage.java	This file contains the data storage class and was modified to handle the name string and to get and store the Android ID.	T
src\main\java\com\example\android\cardemulation\CardEmulationFragment.java	This file contains the view fragment for the card and was modified to have a name field.	T
src\main\java\com\example\android\cardemulation\CardService.java	This file contains an Android HCE Service and was modified to work with the PN532, and send the Android ID as a hex string.	T
src\main\java\com\example\android\cardemulation\MainActivity.java	This file contains the main activity and was modified to remove the log.	T

VHDL

File	Description	Status
DM9000A_IF.v	Ethernet interconnect. From Altera	T
ISP1362_IF.v	USB interconnect. From Altera	T
niosII_NFCProject.vhd	Top level, FPGA interconnect. Modified from ECE 492 Lab1	T

H

File	Description	Status
web_api.h	This is the api handler header file. Modified from the MP3 player project [11]	T

http.h	This is the http implementation header file. Modified from the MP3 player project [11]	T
web_server.h	Modified version of the web_server.h from Altera	T
jsonConverter.c	This is the header for the helper methods to convert model objects to json	T
database.h	Declarations and definitions for database.c	T
model.h	Defines the data model struct and all substructures.	T
doorcontrol.h	Declarations for door control	T
camera.h	Declarations for camara Task	C
gpio.h	Declarations for gpio service routines	T
nfc.h	Holds the NFC task and hardware interface	T
i2c.h	Header file for i2c.c	T
test.h	Contains test framework for unit testing	T
rtc.h	Declarations and definitions for rtc.c	T
pn532.h	Declarations and definitions for pn532.c	T

C

File	Description	Status
web_api.c	This is the api handler that implements the REST interface. Modified from the MP3 player project [11]	T
http.c	This is the http implementation file. Modified from the MP3 player project [11]	T
web_server.c	Modified version of the web_server.c from Altera	T
jsonConverter.c	This file holds helper methods to convert model objects to json	T
network_utilities .c	This file is from Altera and was modified to use a fixed serial number.	T
camera.c	Holds the camera task and hardware interface, potentially uses USB interface from Altera Mouse demo.	C
gpio.c	Holds the gpio interrupt routines and interface	T
nfc.c	Holds the NFC task and hardware interface	T

database.c	Handles reading/writing files on the SD card.	T
model.c	Initializes and maintains the data model.	T
doorcontrol.c	Controls and responds to door hardware events.	T
i2c.c	A general i2c driver rewritten to support clock stretching and communication with devices not utilizing internal registers	T
test.c	Contains test framework for unit testing	T
rtc.c	Methods for reading/writing the time from/to the RTC module	T
pn532.c	Methods for sending commands and reading responses from the PN532 NFC module	T
hce.c	Methods for communicating with an Android device using Host Card Emulation	T

HTML

File	Description	Status
index.html	This is the html page with styles that contains the full markup for the web UI. The file does not contain any comments since the full file is sent to the user.	T

JS

File	Description	Status
index.js	This is the JavaScript that is used by the index.html page and is loaded with it in the browser.	T

Code Folders

Folder	Description	Status
/hw/usb	Code here is used to potentially control the USB, Code is originally sourced from Altera Mouse Demo.	C
/sd	This folder contain code to control and use the SD card. Code sourced from 2013w G12 SD card interfacing.	T
/hw/ethernet	Used to interface with the dm9000a chip, Code from Altera demo.	T
~/software/WebUI/	This folder contains bootstrap, the bootstrap theme, jquery, and knockout. It also contains some authored code files which were outlined above.	T