University of Illinois at
Urbana-Champaign

Door Access Tracker

Patrick Connelly (prc2), Ben Wasicki (wasicki2),

and John Scholl (johnts2)

ECE 445 Design Document
Team #41
TA: Chi Zhang
February 27, 2020

Table of Contents

1. Introduction

1.1 Objective**:**

Many areas of day-to-day life involve the opening and closing of a door. We believe that better information on the state of a door can improve one's quality of life. For example, one could monitor a door as a security measure, such as a front door, a liquor closet, or a medicine cabinet. Alternatively, some doors may also have a tendency of getting stuck open. In this case, knowing that the door was not closed properly may be good information to have. In addition, knowing when the mailbox has been accessed could be time saving, especially for someone who has mobility problems because they would not need to check the mailbox unnecessarily.

Our proposed solution is the Door Access Tracker. This tracker would consist of a sensor to detect the state of a door, a microcontroller, a wifi card, a cloud server, and an android app. This would be a portable device that would be adhered to a door. The primary functionality involves the user getting an update on their phone via an application when the state of the door is changed. In order to make this product more versatile, we would allow for different configurations on when to send notifications. For example, a consumer may want to know the instant a medicine cabinet or liquor cabinet is opened; however, they may only care about a door's state if it were to be left open for a specific amount of time before being closed.

1.2 Background:

There are many situations in which the monitoring of a door or cabinet may be useful. From a security perspective, knowing when an area is accessed could be extremely useful information, especially for knowing when something has been tampered with. From a convenience perspective, putting this device on something such as a mailbox would let someone know when they should go to check for mail. Finally, from an energy-savings perspective, this product could let a person know when a door or window is left ajar, leading to heat loss in the winter and air conditioning loss in the summer. A key issue that needs to be addressed as well is that a user may want a different notification or set of notifications for different situations. For example, they may want to know immediately when a door state is changed, they may only want to know when a door is opened, or they may only want to know if a door is left open for a certain amount of time.

There are some products on the market that attempt to achieve the same functionality as our project. Our design would not only be cheaper than available products, but it would also have additional functionality [6]. The application we propose would be more configurable than currently available alternatives; giving the user the option to select when, why, and how they are notified. Our solution is also stand-alone and does not require any subscriptions or any hub device [5].

1.3 Visual Aid:

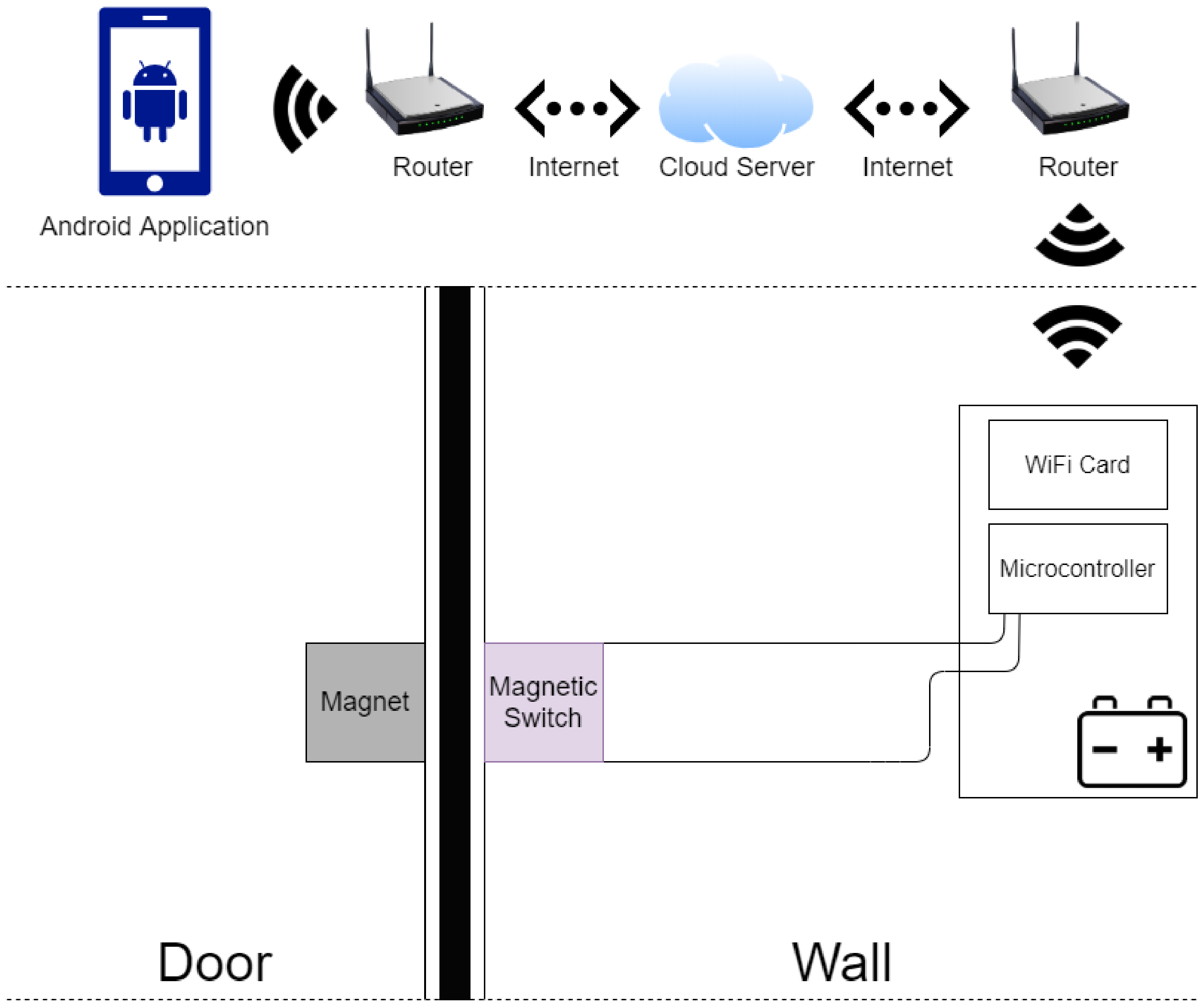

Fig. 1. Door Access Tracker High Level Visual Aid

1.4 High Level Requirements:

The following are the three most important qualities our project must exhibit in order to be successful:

- The door sensor sends a high (3.3V) signal when the door is open and a low (0V) signal when the door is closed. This signal will be sensed by the Communication and Processing System. When the Communication and Processing System senses a change in the signal from the sensor, it will send a packet to the Cloud Server. The body of this packet would be 0x00 if the signal goes from 0V to 3.3V and 0x01 if the signal goes from 3.33V to 0V.
- The hardware will be able to last at least 6 months before the batteries must be replaced.
- The back-end server can send the Android application an update based on the information it receives from the system controller and the current configuration set.

2. Design

2.1 Block Diagram

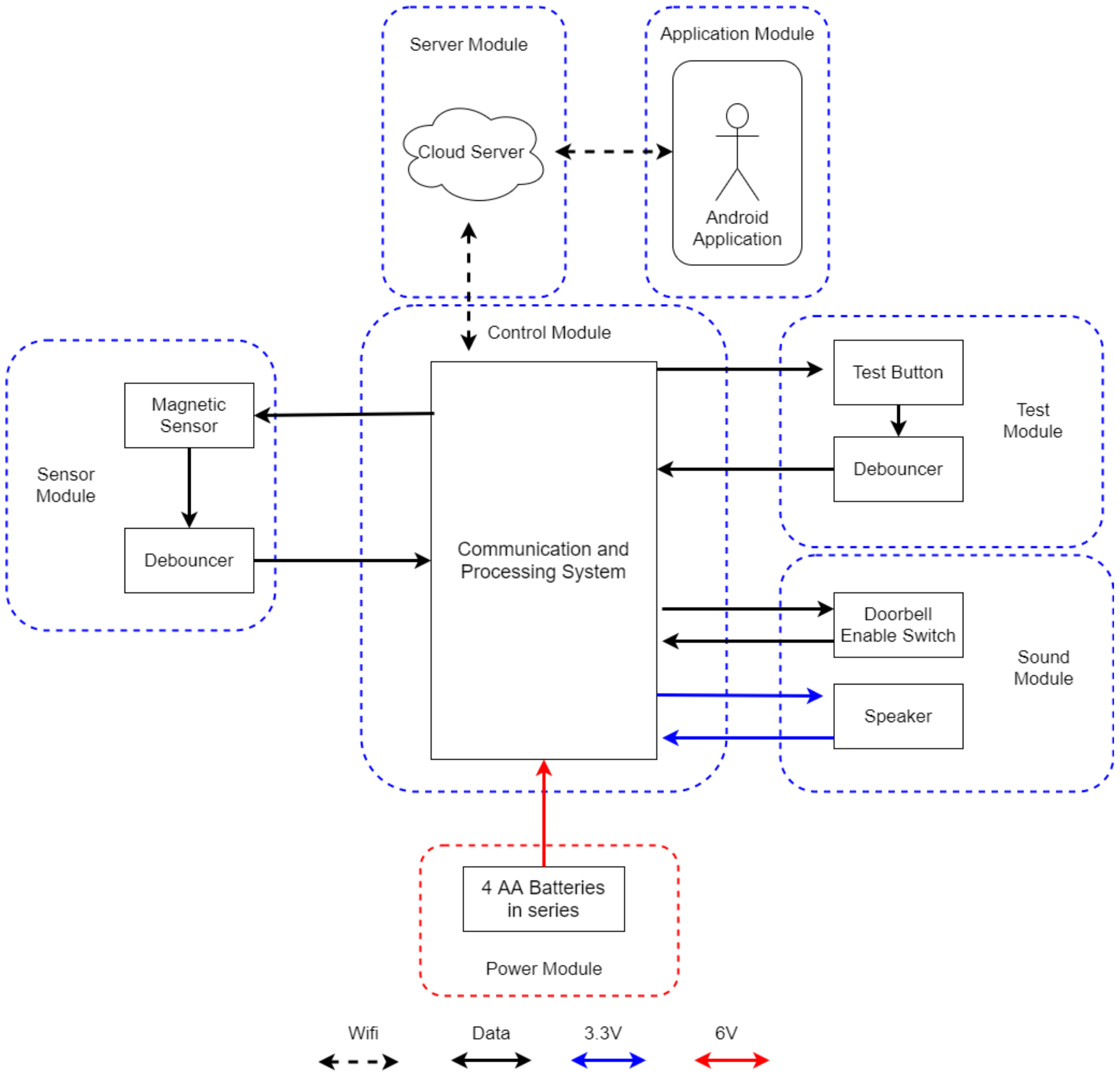

Fig. 2. Door Access Tracker Block Diagram

2.2 Physical Design

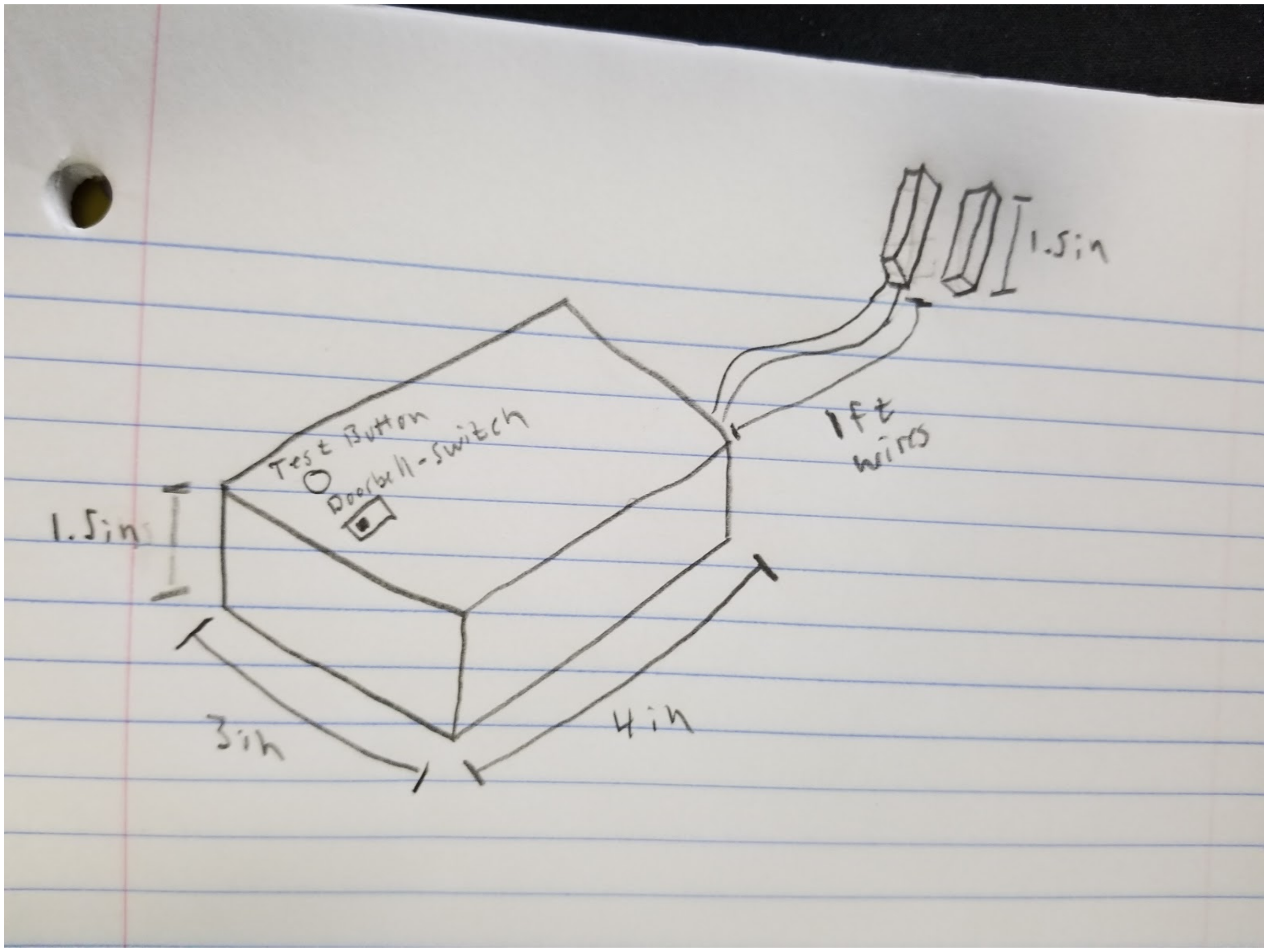

Fig. 3. Door Access Tracker Physical Design

2.3 Subsystems

2.3.1 Magnetic Sensor

We will use the PMC-1001THY reed switch for our sensor. The Magnetic Sensor will be connected to a data pin from the Communication and Processing System on one side and a debouncer on the other side. It will be used to determine the state of the door.

Requirements	**Verification**
1. The voltage across the reed switch will be 0 - 0.1 when the magnet is at least 100mm away. 2. The voltage across the reed switch will be 3 - 3.6V when the magnet is less than or equal to 10mm	1. A. Hook switch up to 3.3V source and ground with magnet at a distance greater than or equal to 100mm. B. Measure the voltage across the switch to see if it is between 0V and 0.1V. 2. A. Hook switch up to 3.3V source and ground with magnet at a distance less than or equal to 10mm. B. Measure the voltage across the switch to see if it is between 3V and 3.6V.

2.3.2 Debouncers

Two debounce circuits will stabilize physical inputs- one for the magnetic sensor, and one for the test button. These will ensure that erroneous values are not read when the program starts and looks for active flags.

Requirements	**Verification**
For each circuit, the output must not change more than once within a 20ms time span that begins when the input has changed	1. Hook up the output of the circuit to an oscilloscope. 2. Set the oscilloscope to pause on a rising/falling edge, and change the input (door open/close or button press) 3. Set the scale to 20ms, and place the first edge at the left edge of the display. If there is only one edge within that time span, the debounce circuit works as intended.

2.3.3 Doorbell Enable Switch

This SPST switch is used for enabling the speaker to produce sound upon opening the door.

Requirements	**Verification**
Voltage drop across the the switch is 0 - 0.1V when the switch is closed and 3 - 3.6V when the switch is open.	1. Hook switch up to 3.3V source and ground. 2. Measure voltage across switch when open and closed.

2.3.4 Speaker

This PKM22EPPH2001-B0 piezo buzzer is used for creating a doorbell chime and debugging. When the the doorbell switch is enabled, the speaker will produce sound upon opening the door. Additionally, pressing the test button will produce one of two unique tone sequences, depending on whether or not the controller can successfully connect to the server (regardless of the doorbell switch state).

Requirements	**Verification**
1. Speaker does not produce sound when voltage supplied is between 0V - 0.1V 2. Speaker produces an audible tone when supplied with a 3 - 3.6V voltage	Connect speaker to 3.3V source and ground, then listen to see if tone is produced

2.3.5 Test Button

This is a push-button used for testing the controller's ability to communicate with the server. It will cause sound to be produced on the speaker.

Requirements	**Verification**
Voltage drop across the the button is 0 - 0.1V when the button is pressed and 3 - 3.6V when the button is not pressed.	1. Hook button up to 3.3V source and ground. 2. Measure voltage across button when pressed and not pressed

2.3.6 Communication and Processing System

We will use the ESP8266 ESP-12E NodeMCU development module to handle all input processing and communication with the server. This allows for a simple design, as the onboard microcontroller has enough resources to process both WiFi communications and our sensor input, which removes the need for a separate microcontroller. Some of the primary features of the module include:

- 4MB of flash memory. This will be used for storing the program to read from the sensor and sending data to the server.
- USB input. This will allow us to easily flash programs onto the board for rapid software testing. The software will be written in the Arduino language.
- Voltage regulator. Tolerating up to 10V, this will allow us to safely use our 6V battery array to provide a 3.3V, 600mA output to the board components. All peripherals will connect to the 3.3V output ports on the module for power.

While transmitting, the ESP8266 WiFi chip uses between 120mA and 170mA, depending on transmission power. Receiving will use between 50mA and 56mA. When not in use, the ESP8266 will go into “deep-sleep mode”, where only the RTC is powered, drawing 20µA and allowing our device to operate for many months at a time.

Requirements	**Verification**
Must be able to communicate over IEEE 802.11b/g/n at a 1 Mbps data rate	1. Flash the network credentials and a test program to the board through the USB port with the “Flash” button 2. Press the test button. The Wifi chip will send 1 Mb of data to the server, which will time the transfer. 3. Check the server test logs and ensure the transfer time was under 1 second.

2.3.7 Battery Array

The battery array contributes to the first and second high level requirements. It is responsible for providing power to the communication and processing system, which in-turn powers all peripherals. It consists of four 1.5V AA Alkaline batteries connected in series, producing a total of 6V. This is connected directly to the Vin and GND inputs on the communication and processing system.

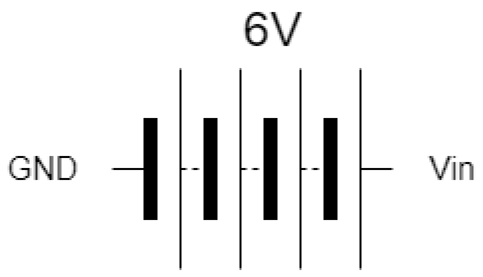

Fig. 4. Battery Array connected to inputs on communication and processing system

Requirements	**Verification**
Provides at least 3V and no more than 10V	Use a voltmeter to ensure that the battery system remains within this range

2.3.8 Cloud Server

The cloud server contributes to the third high level requirement. It contains most of the computations and memory that will be needed. As such, it will take requests from the user via the android application and from the control module, be able to set configurations, store history, and send updates to the android application based on the set configurations. This will be run as a pod in a kubernetes cluster with a service and ingress to connect it to the outside. The benefits of running this on a cluster are persistent volumes, easy scalability, and self-healing.

*The Following are steps that must be repeated for many of the below tests. They are presented here to avoid repetition:

- ***Step 1***: Ensure that the Kubernetes cluster is running with both the server Pod's Ingress and Service, and the test program Pod's Ingress and Service configured correctly.
- ***Step 2***: Start the Pod using the verbose=true argument specified in the pod yaml file.

Requirements	**Verification**
1. Server is able to receive state change from at least 1 control module	1. A. Do ***Step 1***. B. Do ***Step 2***. C. Start the test pod with the -*-test1=true* argument specified in the pod yaml. D. Check the logs of the server Pod to see if they have the following output: *"<Timestamp> : test1-serial-number : open"* *"<Timestamp> : test1-serial-number : closed"*
2. Server will be able to complete registration of a specific application with a specific serial number. 3. Server will be able to send updates to at least 1 application after a state change is received from a control-module. 4. Server will be able to use configurations specified by an application to deliver updates to said application as set by configurations. 5. Server will keep a history of door state changes with corresponding timestamps based on the number	2. A. Do ***Step 1***. B. Do ***Step 2***. C. Start the test pod with the -*-test2=true* argument specified in the pod yaml. D. Check the logs of the server Pod to see if they have the following output: *"<Timestamp> : test2-serial-number : <IP>"* 3. A. Do ***Step 1***. B. Do ***Step 2***. C. Start the test pod with the -*-test3=true* argument specified in the pod yaml. D. Check the logs of the test Pod to see if they have the following output: *"<Timestamp> : test3-serial-number : open"*

specified by the application and will send state changes to an application by request.	*“<Timestamp> : test3-serial-number : closed”* 4. A. Do ***Step 1***. B. Do ***Step 2***. C. Start the test pod with the --*test4=true* argument specified in the pod yaml. D. Wait 10 seconds then check the logs of the test Pod to see if they have the following output: *“Timestamp1 : test4-serial-number : open”* *“Timestamp2 : test4-serial-number : closed”* E. Ensure *Timestamp2* is approximately 5 seconds after *Timestamp1*. 5. A. Do ***Step 1***. B. Do ***Step 2***. C. Start the test pod with the --*test5=true* argument specified in the pod yaml. D. Check the logs of the server Pod to see if they have the following output: *“---HISTORY BEGIN---”* *“<Timestamp> : test5-serial-number : open”* *“<Timestamp> : test5-serial-number : closed”* *“<Timestamp> : test5-serial-number : open”* *“<Timestamp> : test5-serial-number : closed”* *“---HISTORY END---”*

2.3.9 Android Application

The android application contributes to the third high level requirement. It will act as an interface for the user to register itself with a serial number of a control-module, update configurations for different control-modules, request state change history from the cloud server, and receive updates from the cloud server.

*The Following are steps that must be repeated for many of the below tests. They are presented here to avoid repetition:

- ***Step 1***: Download and open the application on an Android device.
- ***Step 2***: Do the Cloud Server tests, ensuring that it is functional.
- ***Step 3***: Ensure that the Kubernetes cluster is running with both the server Pod's Ingress and Service, and the test program Pod's Ingress and Service configured correctly.
- ***Step 4***: Start the Pod using the *verbose=true* argument specified in the pod yaml file.
- ***Step 5***: Go to the registration interface in the application and register the device to serial number *"test1-serial-number"*
- ***Step 6***: Start the test pod with the --test1=true argument specified in the pod yaml file.

Requirements	**Verification**
1. The application will have an interface from which registration, configurations, and history requests can be input for communication with the cloud server. 2. Application will be able to register itself with the cloud server based on a specific serial number corresponding to a control-module.	1. A. Do ***Step 1***. B. Ensure that once the application opens, there is an interface that contains options for *registration, configuration input,* and *history requests*. C. For each of these options, ensure that data can be input or a button can be pushed. 2. A. Do ***Step 1***. B. Do ***Step 2***. C. Do ***Step 3***. D. Do ***Step 4***. E. Do ***Step 5***. F. Check the logs of the server Pod to see if they have the following output: *"<Timestamp> : test1-serial-number : <IP>"*

3. Application will be able to notify the user when it receives an update from the cloud server. 4. Application will be able to retrieve and display door state history upon user request.	3. A. Do ***Step 1***. B. Do ***Step 2***. C. Do ***Step 3***. D. Do ***Step 4***. E. Do ***Step 5***. F. Do ***Step 6***. G. Check to see if an *open* push-notification was sent to the phone followed by a *closed* push notification. 4. A. Do ***Step 1***. B. Do ***Step 2***. C. Do ***Step 3***. D. Do ***Step 4***. E. Do ***Step 5***. F. Do ***Step 6***. G. In the application, click on the *“Door State History”* button. H. Ensure that the following histories appear: *“<Timestamp> : open”* *“<Timestamp> : closed”*

2.4 Tolerance Analysis

2.4.1 Circuit Schematic

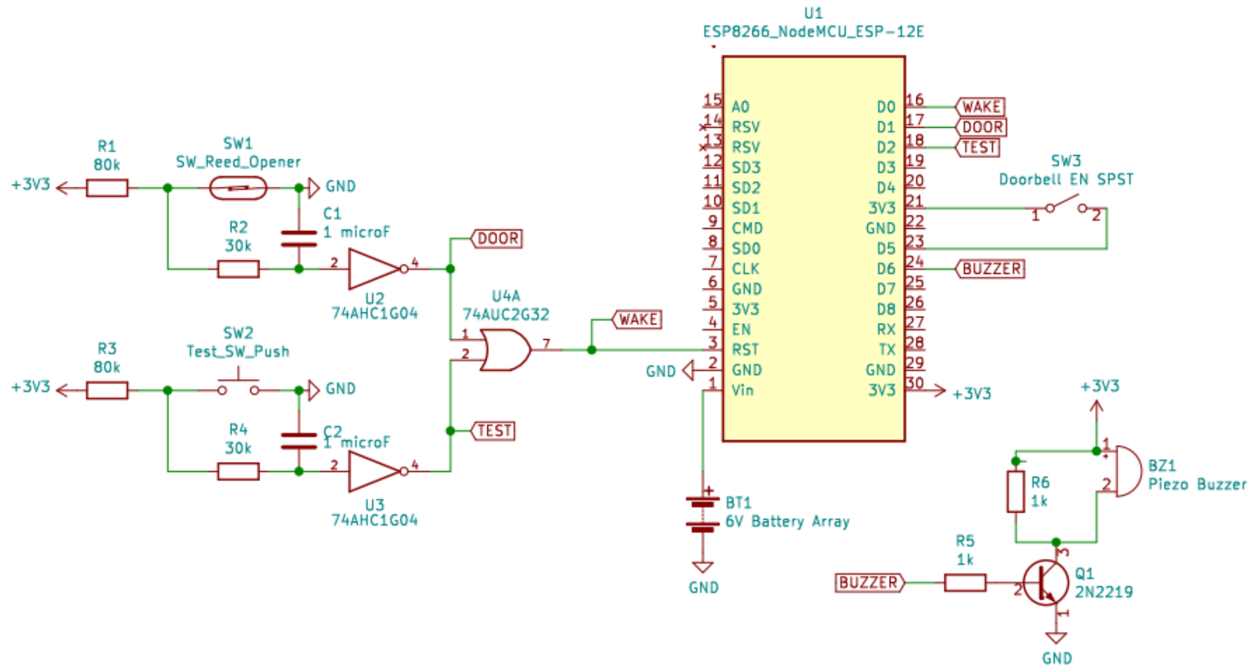

Fig. 5. Door Access Tracker Circuit Schematic

2.4.2 Calculations

- When the door is closed, the WiFi chip is the primary power draw, running on about 20μA in its "deep-sleep" mode. Four 1.5V AA Alkaline batteries in series have a typical capacity of 500mAh. Thus, with no closing or opening of the door, our device should last approximately:

$$\frac{500\ mAh}{0.02\ mA} = 25000\ h\ \approx 1041\ days\ \approx 2.85\ yrs$$

- If transmitting and receiving constantly using the maximum current draw of 170mA for transmitting and maximum 56mA for recieving, this will be a draw of 226mA. Four 1.5V AA Alkaline batteries in series have a typical capacity of 500mAh. Thus the worst case battery life would be:

$$\frac{500\ mAh}{226\ mA} \approx\ 2.21\ h$$

- For an average use case, we will assume the door opens and then closes once every hour, or 48 times a day for a total of 96 state changes. The average transmission current draw is 145mA and the average receiving current draw is 53mA. If the door state is updated, 1 transmission will need to be sent and 1 ack received. In testing by pinging www.google.com 100 times, the average RTT was 13ms with a 23ms maximum and there was a 0% packet loss.

  For these calculations, I will assume a 0% packet loss, an RTT of 40ms, and a transmission time of 1ms. While the packet is actually being transmitted, the circuit will draw 145mA, while waiting for the ACK, it will draw 53mA, and during the rest of the time, it will be in sleep mode drawing .02mA. Four 1.5V AA Alkaline batteries in series have a typical capacity of 500mAh.

  While running, the circuit will draw an avg:

  $$\frac{145\ mA}{40} + \frac{39 \times 53\ mA}{40} = 55.3\ mA$$

  It will do this four times per hour on avg giving us:

  $$55.3\ mA \text{ for } 160\ ms \text{ every hour}$$

  The amount of time in hours is given by:

  $$0.16\ s \times \frac{1\ h}{3600 s} = \frac{1}{22500}\ h$$

  The average current draw per hour is:

  $$\frac{55.3\ mA}{22500} + \frac{22499 \times 0.02\ mA}{22500} = 0.0225\ mA$$

  This gives us:

  $$\frac{500\ mAh}{0.0225\ mA} \approx 22222\ h \ \approx 925\ d \ \approx 2.53\ yrs$$

3. Cost and Schedule

3.1 Cost Analysis

We estimated the cost of our development assuming a 10 hour per week of a 16 week schedule. Considering the average starting salary of a BE ECE, we calculated our hourly wage to be $45 per hour [4].

$$\frac{\$45}{hr} \cdot \frac{10hr}{wk} \cdot 16wk \cdot 3 \ \cdot 2.5 \ = \$54,000$$

Part Number	Manufacturer	Description	Module	Price
MN1500B4Z	Duracell	AA alkaline battery x4	Control	$5.21
ESP8266 ESP-12E NodeMCU	MakerFocus	Wifi card and microcontroller	Control	$9.39
PMC-1001THY	PIC GmbH	Reed Switch	Sensor	$2.00
GF-123-0054	CW Industries	Slide Switch	Sound	$1.20
1301.9314	Schurter Inc.	Button	Test	$0.24
Assorted resistors, capacitors, and ICs	Various		Various	$5.00
PKM22EPPH2001-B0	Murata Electronics	Speaker	Sound	$0.73
8005	Radial Magnet Inc.	Magnet	Sensor	$0.23

Total Part Cost: $24.00

Total Development Cost: $54,024.00

3.2 Schedule

Week	Ben	Patrick	John
2/17/20	Figure out high level server overview	Design circuit and decide on components	Begin circuit design and calculation.
2/24/20	Research platforms to build server and application on	Order parts and begin assembling hardware modules	Purchase prototyping parts and begin building test circuit.
3/2/20	Get basic kubernetes cluster running on google cloud and make sure it can be accessed outside cluster.	Hardware module assembly and breadboard testing between sensor and controller	Finish building test circuit for sound, sensor, and test module. Put in order if more parts are needed.
3/9/20	Write code for the server and test that requirements 1-3 are working.	Submit machine shop order, finish basic control build, test sending and receiving packets	Integrate sound, sensor, and test module with control module for test.
3/16	Write code for the server and test that requirements 4 and 5 are working.	Ensure test button and speaker work, controller communicates properly with server	Establish communication between control module and cloud server.
3/23	Server debug buffer week.	Finish hardware testing, finalize design	Finalize PCB design and send order and make sure final machine shop revisions are done.
3/30	Begin app development and get the requirement 1 completed.	Hardware verification, research solutions for in-app WiFi configuration	Finish revisions and verifications prepare for prototype assembly.
4/6	Continue app development and get requirements 2-4 completed.	PCB testing and verification	Test assembled prototype and debug if necessary.

4/13	App debug buffer week.	PCB final fixes, implement configurable WiFi if time allows	Circuit debug buffer week.
4/20	TA mock demo/ last minute bugfixes	TA mock demo/ last minute fixes	TA mock demo/ Final fixes.
4/27	Prepare for final presentation and write report.	Prepare for final presentation and write report.	Prepare for final presentation and write report.
5/4	Final presentation and turn in report.	Final presentation and turn in report.	Final presentation and turn in report.

4. Discussion of Ethics and Safety

As the developers of this project, we believe it is important that we produce a safe, reliable, and efficient product to our user. We commit ourselves to holding a high degree of professional conduct in accordance with both the IEEE and ACM Code of Ethics. We will avoid ethical breaches by following all device specifications, working in our respective areas of competence, and clearly stating proper operating procedure (ACM 2.6) [2]. At the same time, we acknowledge that our device could be misused; therefore, we will take all necessary precautions to prevent any harmful modes of operation.

In accordance with the ACM Code of Ethics, this project will pose no risk to the user or community under standard operations. Given that our project monitors when a door is opened and closed, it could pose a safety risk to the user if the data is compromised. We will ensure that all wireless protocols are followed, and communications will be secure. The data gathered by our sensor will be the sole property of the intended user of the device (ACM 2.9) [2]. All software will follow accepted community standards.

Following the IEEE Code of Ethics [1], we have decided it is important to make our design as energy efficient as possible to minimize waste. As designers, it is our responsibility to limit the environmental impact of our device. We have implemented a circuit break when the door is closed to ensure power is only consumed when necessary. This will limit the amount of waste associated with battery replacements.

In addition, we will ensure there is no exposed wiring or electrical components in our design to minimize the risk of electrical shock. Similarly we will ensure all components are operating within their respective operating regions to reduce the risk of a short or fire hazard.

Door clearance regulations state that “Required maneuvering clearances provide space for opening and proceeding through doors, doorways, and gates using wheelchairs and other mobility aids” [3]. Our product will not impede a person opening a door by blocking the door so we are compliant here. In addition, The regulations state that a door cannot have an opening force of more than 5 pounds of force [3]. We will be adding a magnet of negligible weight onto the doorframe, so this will not be a source of concern. In addition, the magnet will only be strong enough to interact with the reed switch so the magnetic force helping to keep the door closed will also be negligible.

5. Citations

[1] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 12-Feb-2020].

[2] “ACM Code of Ethics and Professional Conduct,” *Association for Computing Machinery*. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 12-Feb-2020].

[3] “Chapter 4: Entrances, Doors, and Gates, ” *United States Access Board*. https://www.access-board.gov/guidelines-and-standards/buildings-and-sites/about-the-ada-standards/guide-to-the-ada-standards/chapter-4-entrances,-doors,-and-gates [Accessed: 27-Feb-2020].

[4] “Salary Averages,” *ECE ILLINOIS*. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages.asp. [Accessed: 28-Feb-2020].

[5] “Smart Door and Windows Sensor,” *Amazon*. [Online]. Available: https://www.amazon.com/Personal-Security-Automation-Doorbell-Compatible/dp/B07HMP9LQV. [Accessed: 26-Feb-2020].

[6] “Automatic Door Safety Beam Sensor,” *Amazon*. [Online]. Available: https://www.amazon.com/Automatic-Door-Safety-Beam-Sensor/dp/B01A5EAUKK. [Accessed: 24-Feb-2020].