ENEL 351 Project Final Report

Gregory Sveinbjornson

200427591

ENEL 351

Tessa Herzberger

March 3, 2022

Home Alarm and Motion Detection System

Table of Contents

1: Overview

1.1 Introduction

The purpose of this project is to work as a functioning motion detection alarm system using the STM32F103RB microcontroller. A HC-SR501 PIR Motion Detector infrared motion sensor watches its field of view for any motion inside an empty home. Should the motion sensor be alerted to movement, this will set off the PIEZO 3V buzzers and scary red LED lights, which scares away the home intruder. This project is designed for use in homes that have no occupants, as any movement will trip the alarm, causing loud noises and scary red lights, designed to make any would-be thieves run away.

1.2 Block Diagram

The block diagram is shown in Figure #1. This block diagram will be modified as project construction moves further. I have 4 main groups in the original diagram.

The first is the STM32F103RB Microcontroller, as this is the most important piece and where all the calculations occur. The second is the motion sensors, these are in their own group from the other inputs as this is a digital input, not an analog one. The third group is the outputs, this being the buzzers and LED lights. This group is the alarm portion of the project, and is what alerts the user to movement. The final group is the analog inputs, these are the ON/OFF switch and the Alarm OFF button, both of these are intended for use by the user to control the alarm. The system is powered by the STM32F103RB, which is plugged into a power source using the USB cable.

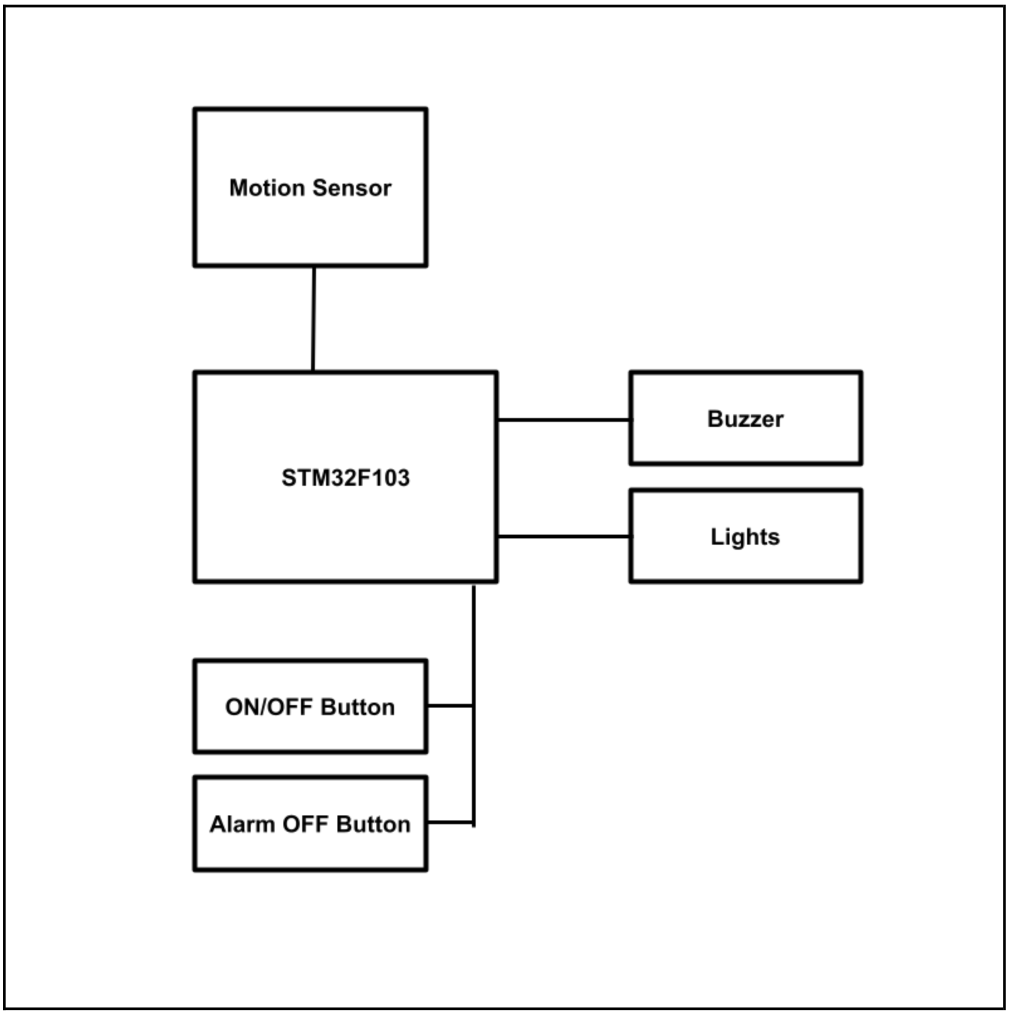

Figure #1

1.3 Schematic

The schematic for my system is shown in Figure #2. The system is powered by the USB connection to the STM board. I initially planned to use two separate pins to control the buzzer and LED separately, however as the project progressed it became clear that it would be simpler to control them both with a single output. My initial plan with the On/Off switch was to input it into the board and control it from the code, however it was much simpler to use the switch to cut the alarm, which still allowed the sensor to have a stable flow of current which was important for the stability of the readings. I also eliminated the Alarm Off Button, as there was no need for a secondary off switch.

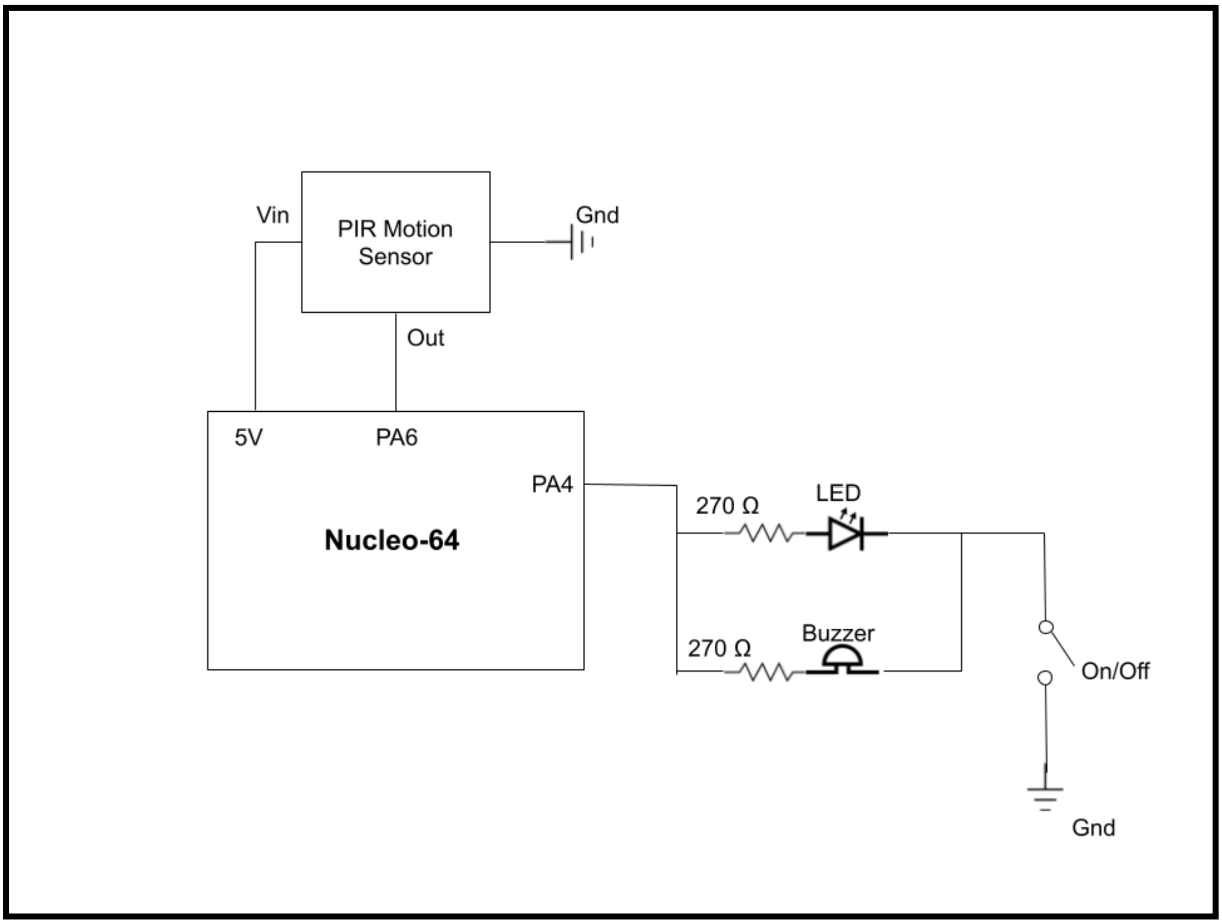

Figure #2

1.4 Equipment

Required equipment:

- STM32F103RB microcontroller
- Computer
- Power source
- HC-SR501 PIR Motion Detector
- PIEZO Buzzers
- Red LED lights
- Analog buttons
- Breadboards
- Jumper wires
- Resistors

2: Inputs

2.1 User Inputs

- ON/OFF Button
    - This is the first thing that a user will use to start the system. It will be bright and red at the front of the system.

- Alarm Off Button
    - This button is meant for use when the alarm is active> Once pressed, the buzzers and lights will turn off.

2.2 Other Inputs

- Infrared Motion Sensor
    - This is the main sensor of the system, this sensor will alert when motion is detected. This will set off the alarm.

3: Outputs

3.1 Alarm Outputs

- Red LED's
    - I will be using the red LED's from the ENEL 384 parts kit. They will turn on when motion is detected.

- Buzzers
    - I will be using PIEZO 3V Buzzers. They will turn on when motion is detected.

4: Analysis and Discussion

4.1 Code Construction

The code for this project was in two main files, main.c and main.h. I had constructed everything in these main files to keep all my code together as I went through different generations of my project. I had 13 project versions until I got my final version working how I wanted. The first 5 versions were tests for the Nucleo-64

Board, testing registers, pins, and ports to make sure everything was working. The next 3 versions were tests for the buzzer. I played around with PWM to get different frequencies to get different sounds for my buzzer. I ultimately abandoned this in my final version as the only sound I wanted for my buzzer was for it to be as loud as possible. The next 3 versions were tests for the motion sensor. This was by far the most difficult as it seemed to be one problem after the other which I will detail in the Debugging section. My final two versions were putting everything together. Version 12 was initially meant to be the final version until I accidentally deleted all the initialization functions, so then a Version 13 was needed.

4.2 Physical Construction

I used a breadboard to construct my project. I chose this over soldering as it was easier to change, and all my wiring would be static and hidden away in a box. I used an old golf ball box as a container, as it was easy to cut holes for my components to stick out of. I used the datasheets to determine how to connect the motion sensor and buzzer. I secured most of the components with tape, as it was what I had on hand.

4.3 Debugging

Most of my problems on this project came from the PIR Motion Sensor. Specifically because a lot of what was on my original datasheet was incorrect/ missing information (I found a correct datasheet to include with the final report). For a few days, my alarm kept going off even though there was no motion. In testing, I checked my code, the pin, and the buzzer itself. It turned out that the problem was from the sensor itself, as even though my original datasheet said the sensor could be run at 3.3 volts, it turned out that the sensor didn't work properly unless connected to 5 volts. Once connected to 5V on the STM32, the sensor actually began to work as intended.

The next major problem I had was the inconsistency of the sensor. It gave strange readings for the first 30-60 seconds of operation, and then would magically fix itself and work as intended. This problem persisted up until the day before my presentation; I found out that the sensor I bought needed to be connected to power for a minute before it could give reliable readings. Once I realized this, I quickly changed my On/Off switch to allow the sensor to continue running and to only turn the alarm off. Otherwise, my system would need 60 seconds to calibrate before working every time you started it. With the final design, it only needs this calibration time for 60 seconds after being plugged in, while the system can stay off the whole time. I felt this was necessary to change even though it was so close to the deadline, as a 60-second wait to use the system every time was unacceptable to me.

5: Conclusion

5.1 If I Could Start Over

As I progressed through this project, I found that there were enough things that I would have done differently to warrant its own section. In a future project, here is what I would have done differently.

- Confirm that all my datasheets had the correct info

- Find a partner, this would have been more fun to do with another person's input, would have saved time on issues that a second pair of eyes would have spotted
- Buy better parts, spending a little bit more money would have gone a long way
- Research my parts better, find any known issues or weird quirks
- Build a final version with soldered wires, I thought the projects with more solid construction looked better and more complete
- Enjoy the process more, I was so worried about my project not working, that I never stopped to realize the progress I had made

5.2 Final Thoughts

When I read the requirements for this project at the beginning of the semester, I was nervous that I didn't know enough to build a working project. But through the material I learned in the labs, I slowly began to feel more confident about my project. It was the small victories that I enjoyed the most, the first time I got my buzzer to go off, and the first time I waved my hand around and my testing LED came on. This class was the final ENEL class that I will be taking at the U of R, and while it is not my passion, I have definitely come away with a great appreciation for all the work that goes into making these electrical systems work.