IoT Enabled Reed Switch Door Sensor Architecture

Using Arduino, Raspberry Pi, RF Transmission, NodeJS/MongoDB & Android

Chad Davidson
University of Tennessee –Undergraduate – Computer Science
Knoxville, TN
cdavid11@vols.utk.edu

***Abstract*—This paper lays out architecture for a cost effective "smart" door sensor that will inform a user, via Android application, of door open events at their house or office door. Many different languages are used in the implementation and further applications of the door sensor are discussed as well as some of its shortcomings such as possible interference from other RF devices. This document is meant as a way for interested parties to rebuild the project utilizing the instructions provided within, as well as code provided at:**

https://github.com/cdavid11/opendoor

***Index Terms*—Arduino, Raspberry Pi, C++, NodeJS, MongDB, RF.**

I. Introduction

There are many ways in which smart homes are starting to become part of everyday life. From lamps that are set on timers to turn off at a specific time of the day, to smart thermostats that will regulate the temperatures in your house and give detailed reports about your energy usage, the smart home revolution has arrived.

The purpose of this paper, and project, was to create an RF based communication system in a household to create an IoT enabled magnetic door sensor. Many smart home devices, especially those utilizing low power (smart bulbs, door sensors, etc.) use RF transceivers to communicate with each other. In this paper, I propose a cheap architecture for a "smart" door sensor that will utilize an Elegoo microcontroller, Raspberry Pi 2, a web server, and Android application.

II. Materials and Schematics

A. Materials

- Elegoo Mega 2560 Board (or Arduino)
- 433 Hz Receiver/Transmitter Pair
- Magnetic Reed Switch
- Raspberry Pi 2
- Female to Female/Female to Male Jumper Cables
- Android Enabled phone or emulator

B. Schematics

There are two schematics that I will provide here that will allow you to wire the Elegoo board and the Raspberry Pi in the proper format. First, the diagram for the Elegoo board:

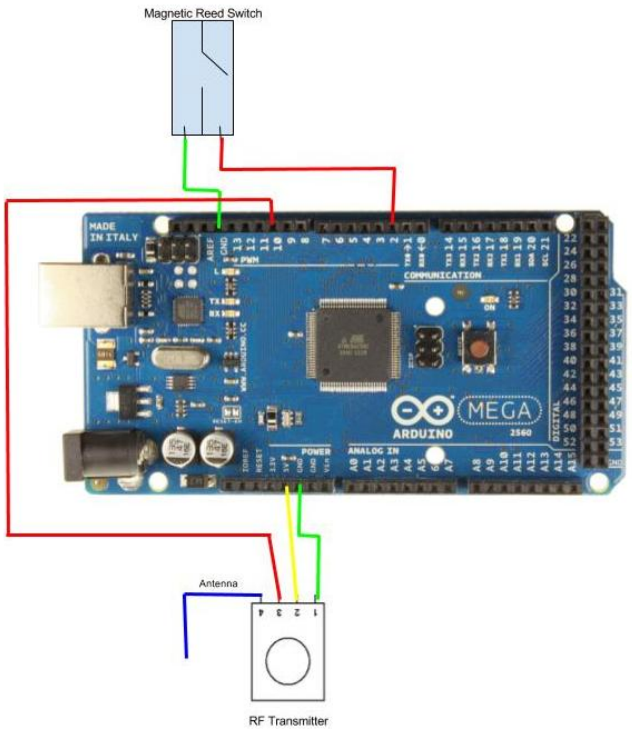

Figure 1: Mega 2560 Board w/ Magnetic Reed Switch and RF Transmitter Diagram

In Figure 1, there is a magnetic reed switch and RF 433Hz transmitter attached to our Mega 2560 board. The ordering of the wires doesn't matter for the reed switch. One wire leads to ground and for the code located in the repository, pin 2 on the Mega 2560 board is where the other wire will lead. The RF transmitter has leads to ground, a 5 Volt power supply, and a lead to pin 10 on our board. The antenna used for this project is a simple rolled piece of aluminum foil.

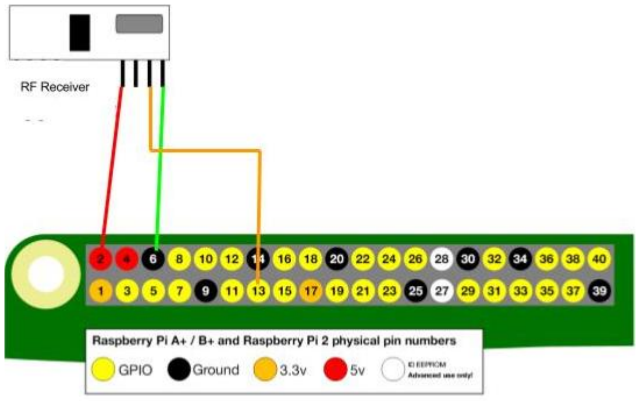
RF Receiver
Raspberry Pi A+ / B+ and Raspberry Pi 2 physical pin numbers
GPIO
Ground
3.3v
5v
ID EEPROM Advanced use only!

Figure 2: Raspberry Pi 2 pinout diagram with RF Receiver Wiring

In Figure 2 we simply have a diagram of the pinout of a Raspberry Pi 2 attached to an RF Receiver. The RF Receiver has leads to ground, a 5 Volt power supply, and to pin 13 on the Raspberry Pi (used in OpenDoor.cpp in repository). Note that there is a second Data pin that is not be utilized in this project and an antenna (not pictured) was formed out of rolled aluminum foil.

III. Communication Flow

A. Overview

In each proceeding subsection, I will discuss in detail the way in which each of the devices will communicate with each other. A simple overview will help to understand why I have broken the sections up as I have. Initially, the reed switch will be opened which will cause the Elegoo board to send an RF signal from its transmitter to the RF receiver located on the Raspberry Pi 2. The Raspberry Pi will then send and HTTP POST request to a RESTful web server that I set up in the "cloud." This web server will then either push information, or receive GET requests from an Android application and the end user will see when their door has been opened. All libraries mentioned in the following sections are open source.

B. Elegoo to Raspberry Pi 2

The connection between the Elegoo and the Raspberry Pi is the most important as it is where the sensor in our IoT architecture communicates with another device. Once the door is open and the reed switch activated, an RF transmission of 433 Hz is sent from the transmitter using the rc-switch library for the Arduino. A binary code is sent to the receiver connected to the Raspberry Pi and it is incremented for every subsequent door open event. This way, the Raspberry Pi can keep track of individual door open events.

The RF transmission range was tested in a 1200 sq. ft apartment. When the reed switch is activated, a constant stream of binary numbers is transmitted to the receiver attached to the Raspberry Pi. Using a simple piece of aluminum foil as antennae on the receiver and the transmitter, the entire 1200 sq. footage of the house is covered. The Raspberry Pi utilized the wiringPi library and the 433Utils library to receive the binary codes. These codes are then output to a text file that is read by a Python script that will perform the next step of the communication process.

C. Raspberry Pi 2 to NodeJS Web Server

Once the binary codes have been received and output to the text file, a Python script utilizing the Requests library will check the text file every second for new updates. When a new code is placed in the text file, a POST request is then sent to an HTTP web server that is set up with NodeJS and a MongoDB database. This server implements a RESTful API that stores door open events by date and time. The data located in this database are accessible via GET requests that can be performed by our Android application.

D. Web Server and Android Communication

The final step in the communication process is between the web server and the Android application written in Java. While it's possible to send push notifications whenever a new door open event is detected by the web server, for the purposes of this project, the Android application will be making GET requests to the server. The Android application uses a variety of libraries that are available through Android studio but the library used to make the requests is the Volley library.

IV. Further Applications

This design will also allow for further applications to be developed with the current sensor architecture, and it provides a framework, through the Raspberry Pi by which other sensors can be added to the "smart home" network.

A. Reed Switch Applications

The door sensor provides a way of seeing whether or not a door has been opened. The most obvious way to extend this application is to record the amount of time a door is open. While this may be useful to individual end users, there lies a possible business interest in knowing how often the doors to your store are open/closed. This information could be used by businesses to lower their energy costs.

A second future application for this architecture could include connecting a Bluetooth module to the Mega 2560 board to identify who is coming through the door by pairing with an end users phone. While there are many useful reasons for this sort of connectivity, I always envisioned theme music being played every time I walk through the door! It could also be used by parents to find out which child is trying to sneak out that night.

B. Further Raspberry Pi Applications

Since our Raspberry Pi uses an RF receiver it can not only receive the transmission from the magnetic reed switch but it can also be outfitted with the ability to receive transmissions from other RF enabled devices in the household. Truly, the Raspberry Pi could be used as a hub for RF enabled smart home devices throughout the house.

The Raspberry Pi can serve as a fog computing device which can store information locally before sending it to the cloud server. This can be useful when applications might need real time support or low latency that cannot be provided by the cloud service.

C. Android Application Improvements

The Android application for this project, written in Java, is very basic and not especially pleasing to the eye. In the future, I would like to add time zone support from the device so we're not just using Unix timestamps. There can also be many ways to view and display that data located in the MongoDB database. For instance, there could be a feature in the app where a calendar could be integrated so people could have a more robust view of the door open events in their household.

V. Potential Issues and Solutions

The main area where potential issues may arise is through interference on the 433Hz RF frequency. As stated in the introduction, many home devices use RF signals to communicate. There could be more than on RF receiver trying to send signals to the Raspberry Pi or it could be picking up signals that it wasn't intended to receive. Unfortunately, the scope and swiftness with which this assignment was to be completed forced me to cut out interference testing with the RF units. In the case of multiple transmitters attempting to communicate with the Raspberry Pi, there would need to be a registration system in place on the Raspberry Pi that kept track of incoming signals and their sources. As it is now, this architecture does not provide that support.

VI. Conclusion

This project was a great learning experience with receivers, wiring microcontrollers, and connecting heterogenous devices with each other through many different communication methods. I could get a working solution for a single, particular use; however, I was also happy to discover the other potential applications that could be based on such an architecture.

VII. References

Figure 1: Arduino Mega 2560 Board Image – http://www.electroschematics.com/wp-content/uploads/2013/01/Arduino-Mega-2560-Pinout.jpg

Figure 2: Raspberry Pi 2 Pinout Image -- https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/