Design and Implementation of a Cost-Effective Motion Detection and Home Climate Module for Homeowners

Nick Johnson, Logan Relph, and Nicholas Pagan

CS370 Colorado State University

Fort Collins, CO, USA

Ⅰ. Introduction

Recent technological advances have allowed homeowners to fully control their homes using their smartphones or their voices. Home automation devices such as smart light bulbs, thermostats, and similar products are popular for their technological convenience. Homeowners can seamlessly integrate these devices into their homes and begin using them immediately to assume control of many aspects of their home. On the other hand, the recent rise in popularity and availability of microcontrollers and small computers such as the Raspberry Pi allow homeowners to create home automation systems of their own. These systems often allow for more flexibility in functionality, since they lack the specific functions that typical home automation devices come with. The systems homeowners can create on their own can incorporate as much or as little functionality as they wish, depending on the hardware they choose to incorporate into the system.

Ⅱ. Problem Characterization

Smart home devices are expanding and increasingly capable of controlling aspects of modern homes including lighting, temperature, outlets, automatic vacuums, and security. Many homeowners would consider security one of their top priorities. Unfortunately, home security systems often prove to be either expensive or primitive. A full home security system can easily be upwards of $200 [1]. While there are inexpensive security devices such as automatic floodlights, they do very little to notify a homeowner of trespassing unless they are at home. The problem lies at the fine line between affordable and informative smart devices. Self-made home security systems have been developed with varying degrees of complexity. One such system incorporates only a heat and motion sensor, along with a few other components to give the system Wi-Fi capabilities [2]. It is then controlled through a either a smartphone application or a web interface, depending on the user’s preference. Another, more complex system, incorporates the use of a soil humidity sensor, photoresistor, temperature and humidity sensors, a poisonous gases sensor, flame sensor, and a motion sensor [3]. This proposed system would automate temperature control, electricity usage, and the irrigation of plants within the house. It would also provide a high level of security and awareness of the state of the household even when the owner was away. The gas and fire sensors would alert the homeowner of any emergencies within the house, and the motion sensor would ensure any trespassing was recorded. All of this capability is controlled through a smartphone application, providing a high level of remote control over the household. The extensive cost of this second system is apparent, based off the sheer number of sensors required to implement it. Our development group prioritized simplicity in order to create a compact product with increased affordability and versatility.

Ⅲ. Proposed Solution and Implementation Strategy

1. Methodology

Our group decided that basic security and climate data would be the main focus of our smart module. These data would help homeowners keep their house secure while, also preparing the owner for the day by informing them the type of weather they could expect. We also added an email functionality to the system that sends the homeowner information of how many times the motion sensor was triggered, the current temperature and humidity, as well as the average temperature and humidity of the past hour. We decided to code the module with python so that the code could be easily understood by users. Our development group opted for a simple system in order to minimize cost and size. This would allow the system to be affordable and versatile in regards to where it could be placed.

1. Raspberry Pi B+

For this project, we thought it best to use a Raspberry Pi B+ as the means to implement our smart module. The Raspberry Pi’s versatility allows homeowners to have control over their smart module, adding more or less functionality as they see fit. Since the Raspberry Pi is a computer, users that have little background with working with microcontrollers can instead interact with an operating system installed on the

Pi, something more familiar to them.

1. P.I.R. Sensor

The first sensor we use is a HC-SR501 Passive Infrared Sensor (PIR). These sensors are commonly used in security devices for intruder detection [4]. Anything with a temperature above absolute zero emits infrared energy, a wave length invisible to the human eye. These primitive sensors detect influxes in infrared energy radiating from matter in its apparatus [5]. These sensors, however, cannot provide data about what triggered these influxes. The sensor can be manually modified to adjust the sensitivity and time delay of the sensor. The range of the sensor is anywhere from 3 meters to 7 meters, and the delay can be set to anywhere from 3 seconds to 5 minutes, approximately [6]. The sensor also has two delay modes: Single trigger and Repeatable trigger. Single trigger mode activates the delay immediately upon detection of motion, during this time continue detection is blocked. In repeatable trigger mode, the time delay is restarted every time motion is detected, with no continued detection blocking.

1. DHT22 Sensor

The second sensor we use is a DHT22 temperature and humidity sensor. This sensor can measure humidity from 0 - 100% with an accuracy of 2%, compared to its cheaper counterpart, the DHT11 which can only measure humidity from 20-90% with an accuracy worse than 5%. The DHT22 is also good for temperatures -40-80 degrees Celsius with an accuracy of +-0.5 degrees Celsius [7].

1. Libraries Used and Personal Development

In order to incorporate our sensors, we had to make use of the Raspberry Pi’s general-purpose input/output (GPIO) pins. The RPi.GPIO library allows the Pi’s pins to be assigned as input or output. Data can then be sent or received to the various pins depending on the desired functionality of the python script. For our use, a pin on the Pi was used to monitor output from the P.I.R. sensor. If the value from the pin returned true, it indicated that the P.I.R. sensor had sensed motion. Similarly, the DHT22 required a similar library called Adatfruit\_Python\_DHT in order to read the sensor’s temperature and humidity values. Using this library, the script can obtain values from the sensor. Sometimes the sensor fails, in which case, the script makes another attempt to get temperature and humidity values.

To send the informational email every hour, a few libraries were needed. The first library required was the smtplib library. This library defines a Simple Mail Transfer Protocol (SMTP) object that can be used to send email. The email.mime library was used to import some objects and functions that help construct the actual email being sent. More specifically, we used a Multipurpose Internet Mail Extension (MIME) multipart object to define a subject for an email and to help format the various data in the body of the email itself. Finally, the time library allows time to be tracked within the module’s script. This was needed in order to send the informational emails to the user on an hourly cycle.

From these libraries and a few classes of our own, we created the main script for our module. When ran, the script asks the user for the email address they wish the hourly updates be sent to. Then the script enters an infinite loop, that can be interrupted manually by a keyboard interrupt (ctrl-c). This loop sets up a fifteen-minute timer, collecting data from both the P.I.R. and DHT22 sensors every tenth of a second and saving them. At the end of the fifteen-minute timer, the script creates an email based on the data and sends it to the user-specified destination email. The script also prints “Motion Detected” to the terminal whenever the P.I.R. sensor detects motion, in case the user would prefer have a live update of when motion is detected for security purposes.

Ⅳ. Trial and Error

With our project, we as a team thought it would be a good idea to implement a graphical user interface(GUI) to go with the program. The reason being that many people are not familiar with the command line interface, and that a GUI adds a professional look to a program. But, when we tried to actually implement the GUI, we ran into the issue of needing multiple threads to run the scripts for reading the sensors. If we didn’t implement our program with multithreading, the GUI would crash due to the scripts running endless loops. The scripts must be run in endless loops, as if they weren’t, the continuous toggling off and on of the sensors would burn them out very quickly. So we went on to implement the threads, but quickly found out that we would be held back by the hardware of the Pi. With the threads running, the GUI would lag severely, as would the updating of the data from the sensor. So in the end, we decided to scrap the idea of the GUI, as we valued accuracy and response time of our data over the appearance of our program.

Ⅴ. Observations

1. Power limitations

Originally, planned to have magnetic switch sensors that would have been placed on a door or window, letting the user know when these entrances were opened, however, the Raspberry Pi B+ did not meet the voltage requirements of the switch sensors.

1. Accuracy and Response Time:

The DHT22 sensor is a very responsive and accurate device, measuring the temperature and humidity every 2 seconds with a temperature error of +-0.5 degrees Celsius and a humidity error of 2%, as mentioned previously. However, the PIR sensor proved to be difficult to tune to this project’s standards. The sensor itself has two knobs that control it’s range and its output time [8]. The factory settings had the range much too high to test, and the output buffer was too long (about three minutes between activations). Turning these features down to their lowest settings allowed for us to test motion much closer (less than a meter), and the time buffer less than 3 seconds. While this solved a good portion of our testing problems, there is one minor issue that cannot be resolved. There is some disparity between the timing of  the loops within our code and the hardwired time buffer setting, our code will continually test for an input from the sensor even when it is “cooling down” and not accepting any input, ultimately the time buffer of the sensor itself dictates when motion is ready to be detected. As for the email sent every hour, we found the email notification was received within 7 - 12 seconds.

1. Cost and Marketability:

A fully functioning Raspberry Pi B+, along with an SD card, and sensors cost around $52 dollars. Selling this module for say $70 dollars, provides an attractive mid-range smart home device. As said before, smart home devices that notify the consumer remotely cost hundreds of dollars. Devices cheaper than ours, such as automatic flood lights, do little to notify the consumer of motion except when they’re at home. The fact that our module provides useful data about the consumer’s environment remotely, should be very attractive to consumers, and thus very marketable.

Ⅵ. Conclusions

The rise in popularity of smart devices allows homeowners to take full control of their households from the convenience of their phones. However, these systems can be costly, and often lack customization that some homeowners may desire. Our development group was able to design a cost and size efficient smart module for homeowners using the Raspberry Pi B+. It incorporated both a P.I.R. motion sensor and a DHT22 temperature and humidity sensor. The module is more cost effective than most smart devices on the market, as it cost approximately forty-seven dollars in total. More functionality can be added to the module with either by adding sensors or modifying to the main script that runs on the module. We encountered issues when attempting to implement a GUI with the module, but in the future, it could support both a GUI and a smartphone app to allow it to be monitored conveniently.

References

[1] Ring Alarm – Home Security System with optional 24/7 Professional Monitoring – No contracts – 11-piece kit – Works with Alexa, “https://www.amazon.com/Ring-Alarm-Professional-Monitoring-contracts/dp/B07D7Q6CHB?th=1”

[2] Semanur Karaca, Alper Si SMAN, SAVRUK Ibrahim, "A Low-Cost Smart Security and Home Automation System Employing an embedded Server and a Wireless Sensor Network", 2016 International Conference on Consumer Electronics.

[3]J. Waleed, A. M. Abduldaim, T. M. Hasan, Q. S. Mohaisin, "Smart home as a new trend a simplicity led to revolution", 2018 1st International Scientific Conference of Engineering Sciences-3rd Scientific Conference of Engineering Science (ISCES). IEEE, pp. 30-33, 2018.

[4]Moghavvemi, M., and Lu Chin Seng. "Pyroelectric infrared sensor for intruder detection. " In TENCON 2004. 2004 IEEE Region 10 Conference, vol. 500, pp. 656-659. IEEE, 2004.

[5]“Passive Infrared Sensor.” Wikipedia, Wikimedia Foundation, 7 Apr. 2019, en.wikipedia.org/wiki/Passive\_infrared\_sensor.

[6]“Arduino HC-SR501 Motion Sensor Tutorial.” Henry's Bench, 10 Feb. 2018, henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-hc-sr501-motion-sensor-tutorial/.

[7]“50 Of the Most Important Raspberry Pi Sensors and Components.” Raspberry Pi Tutorials, tutorials-raspberrypi.com/raspberry-pi-sensors-overview-50-important-components/.

[8] Adafruit Industries. “DHT22 Temperature-Humidity Sensor + Extras.” Adafruit Industries Blog RSS, [www.adafruit.com/product/385](http://www.adafruit.com/product/385).

[9] Rathod, Hardik. “PIR Motion Sensor with Raspberry Pi.” Hackster.io, 17 June 2017, www.hackster.io/hardikrathod/pir-motion-sensor-with-raspberry-pi-415c04.