An Updated Raspberry Pi Based Interactive Home Automation System through E-mail

Dhavalkumar Patel, Shin-Jung Shieh, Venkata Bharat Varma Namburi

dpp56@njit.edu,

***Abstract*— Home automation is a collection of internet of things devices working together to provide users with increasingly greater control of appliances and electronics in their homes. Using ready available affordable hardware, this paper outlines a simple home automation system that is easy to set up and maintain. Using simple email protocols and python we can use simple logic to control appliances with the Raspberry Pi. The product of this is a cost efficient method building extendable home automation systems.**

***Keywords*—Home automation, Python, Raspberry Pi, E-mail, Web interfaces.**

I. INTRODUCTION

Home automation systems are used to connect many different systems that normally exist in the home separate from one another. This allows for user control to happen from one gateway rather than going to each device for their own interfaces. With todays technology, home automation is a highly connected system with many difference smart controls. For example, many home automation providers offer home control that can be access through a mobile phone using an app that may be accessed from outside of the homes network. With so many devices connected to the internet, home appliances enter into what is known as the "Internet of Things" or IoT. IoT devices are internet enabled devices that can interact with each other and be controlled remotely. Home automation is a rapidly growing industry with a variety of tech companies entering each year.

This paper extends the capabilities of a simple home automation system which used a raspberry pi system connected to three LED lights which can be controlled through the email protocol. The base system does not provide any extra functionality such as automatically turning certain appliances on or off based on whether a set condition has been met. Also, no impulse based systems are implemented. For example, a door opening and closing system is a system where an impulse is sent to open or close the door locking mechanic, but no signal is continuously sent like in an LED light system to keep the light on.

II. GOALS

A. Milestone 1– Implement a basic home automation system

In this phase of the project, the main goal is to build a basic home automation system that is simple yet extendable. The main components of this system include a Raspberry Pi running Raspbian Linux, server software written in Python and LED lights along with other related hardware. For this stage, the only hardware other than LED lights were cables, resistors and a breadboard to hold our components. On the software side, a simple server application was developed using standard Python libraries that checks for an email at a specified email address, parses the subject field of the email and takes actions based on the content of the email. For the email address, it used a simple Gmail account has enough functionality for our needs. The only actions the server could handle were setting the state of three LED'd on and off. The loop ran with a set delay at the end in order to prevent sending too many requests to check email to the specified email. The delay that is used in our version is 1 second. When the delay is set higher, the response time of incoming requests can take a long time if received at the beginning of a delay.

B. Milestone 2– Extend functionalities by implement new hardware and software

To make the home automation system more useful, we need to add different kind of hardware or sensors to detect the change of environment. In this project, we create a mini home model which include appliances like heaters, lights, stoves, and doors. This model uses LED lights to represent some features but are controlled by real sensors. The heater, represented by a red led, can be turned on or off through e-mail. Also, a simple monitor system is included to keep the room at the user set temperature. This functionality is implemented by expanding upon the main server loop in the Python script. This can also be forced on or off by the user. The light control is still controlled in the same way as before and performance has not changed in this area. Another functionality implemented is door control. Unlike light control which sends a constant stream of power through the GPIO pins, a door lock is controlled by sending short bursts through the GPIO pins. For our model, we used a motor to represent a lock on a door. Locking and unlocking is represented by the motor spinning in opposite directions momentarily. The door lock can only be controlled through the e-mail in our model. Another appliance modeled is a stove represented by a led. For this system, we wanted to have physical controls along with remote control with email in order to provide more options for control. The stove can turn on manually by the switch or by email. Here, we also include a safety system by detecting if people are in the room. Using a distance sensor, we can tell if a room is empty or not based on simple logic. The sensor uses the changes in distance to tell if a person has entered or left a room.

For example, if the sensor detects a change of distance going towards it, it will increment a variable representing the amount of people in the room. If the distance is increasing, it will assume the person has left the room and decrement the variable. If the stove is left on and the program detects that there are no people in the room, it will notify that the stove is left on.

C. Milestone 3– Increase user friendliness by implementing easy to use user interfaces

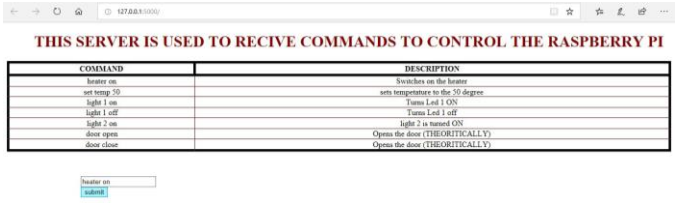

Figure 1. An image showing the web service running in the browser.

To make the home automation system more useful, we need to add different ways the user can interact with the various devices attached to the raspberry pi. The functionality to send commands through an e-mail application is not intuitive or user friendly. We need to add a Web server which provides a list of commands and their description and the ability for the user to send their commands via email to the raspberry server with the click of a button. We also need to add SMS functionality wherein a user can send in his/her commands in the form of SMS and received a confirmation message if command was received. This functionality particularly proved useful when the user does not have access to the internet.

In order to implement these functionalities, we made use of Flask server on python and Twilio for SMS. For the web server, we made a simple webpage that takes a command and sends that command to the home automation server for processing. For our server to be visible on the internet outside of the local network, ngrok was implemented. SMS functionality required a phone number and an account on Twilio's service to be registered. Twilio is set up to listen for an SMS on a specified phone number and do a post on our web server to deliver the commands to our home automation service.

III. Implementation and Installation

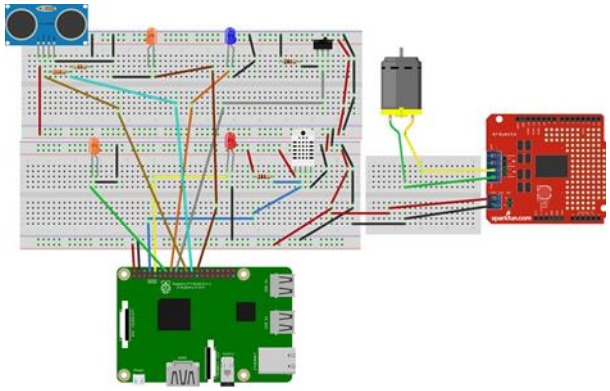

Figure 2. A diagram of the system that includes all hardware components used. The orange led's is our lights system, the blue led represents our stove system along with its switch, and the red led is our heating system which is controlled in software with the help of the temperature sensor.

The installation of our system can be done relatively quickly. We used Raspbian Linux on our Raspberry Pi, but any Linux distribution made for the Raspberry Pi should work as well. The only requirements are that the distribution supported the GPIO pins on the Raspberry Pi. Next, make sure Python is installed and working correctly. The version used in this project is Python 3.7, but Python 2.7 will work with our program as well. To install the required libraries, we use pip. The needed libraries are available in the how-to documentation. For our server software, we have three different scripts for the three different branches of our system. Only the home automation script (homeautomation.py) is required for usage. On the hardware side, components need to be assembled as described in figure 2. The components need to connect to the same GPIO pins in the diagram or the system will not work. The script is set up to use a temporary email that was specifically created for the project, but a new email should be created so only desired emails will appear in the inbox. Lastly, start the server software using the commands in the documentation.

IV. Experiment Designs

The home automation system is a combination of lots of functionalities, we first make sure the raspberry pi can receive and read the e-mail subject then we test every function separately. after making sure all the hardware can work properly, we then all them together onto the pi. In order to test our components, we created test scripts to understand how the components work through the GPIO pins and to confirm their functionality. After confirming all our components are set up correctly, we test our home automation software. We first test for functionality, then we test for speed. Using a script that sends emails randomly to the server, we can test how well the server handles multiple requests and what its response time is.

V. Experiment Results and Analysis

Testing this system at this point is relatively simple. First, the response time was tested. At any delay setting, the worst-case response time was the sum of the delay and the response time of the email servers. The processing delay is negligible, so it does not add to our delay. The best-case response time is just the email server response time if the request arrived just as the program starts to check for new emails. For our system, we used a delay time of one second, so our response times were always in one second or less. Another behavior we noticed after using test scripts is that when a lot of emails were sent within the programs set loop delay, only the first email received would be processed. This is a huge bottleneck in how many actions our system can perform.

VI. Conclusion

Using Raspberry Pi to rapidly prototype a home automation system openned a lot of doors in knowing what could be done on a small an affordable system. Our system is relatively simple to set up but there is still a lot of potential for future expansion. There is a lot of functionality still missing from our system. Security is an area the program can be greatly improved. Also, the components are mostly all hardcoded and would make more sense if they were loaded dynamically through a configuration file. All in all, our system is a good starting point for a more full featured system.