Kevin Palani and Matthew Pham
ECE 120 Honors Lab
12/12/2018

Final Lab Report

Introduction

Problem Description

After seeing all the various ideas for the ECE 120 Honors Lab projects, we saw that there was a common theme: dorm room automation. However, all the proposed ideas were specific to an appliance (such as a light or curtain). Moreover, there was no standardized way to operate and allow for the appliances to communicate. We decided to approach dorm room automation with a more modular project that would be able to control all appliances that were connected to a "hub" and decided to call it IoT SmartHub.

Design Concept

IoT SmartHub has a simple setup: a "hub" that a device that users want to control can connect to, as well as a website with a portal to configure that device. For development purposes, the website would be run on Cloud9, a server development platform, and consist of a MySQL database that stores the user's information and device ID's, an InfluxDB database that stores all statistics regarding the device's performance, and 3 HTML pages. The website directly interacts with the ESP32, the central processor that powers the "hub". The first HTML page is the home page, where the user can see their list of devices and easily change the devices' states. There would also be links to proceed to the other 2 web pages, the first being the "Settings" page for that device, which allows the user to more specifically customize the device's behavior. The other link would proceed to the "Data" page for the device, which is a graph that displays all the statistics of the device on an easy to read graph. For the hardware side, we would have a switch that choose between 9V, 5V, and 3.3V. These signals would pass through a and gate to give us an enable signal. The signals then switch the corresponding transistor, which connects the power source the the correct voltage divider. This then connects to an H bridge, which will allow us to have a forward and reverse control. Finally, the output gets adjusted using another transistor and a pwm signal. In total, there would be 3 control signals from the microcontroller to the rest of the hardware: enable, direction, and pwm. A sensor would input into the microcontroller, and a switch and power source would connect to the other parts of the hardware.

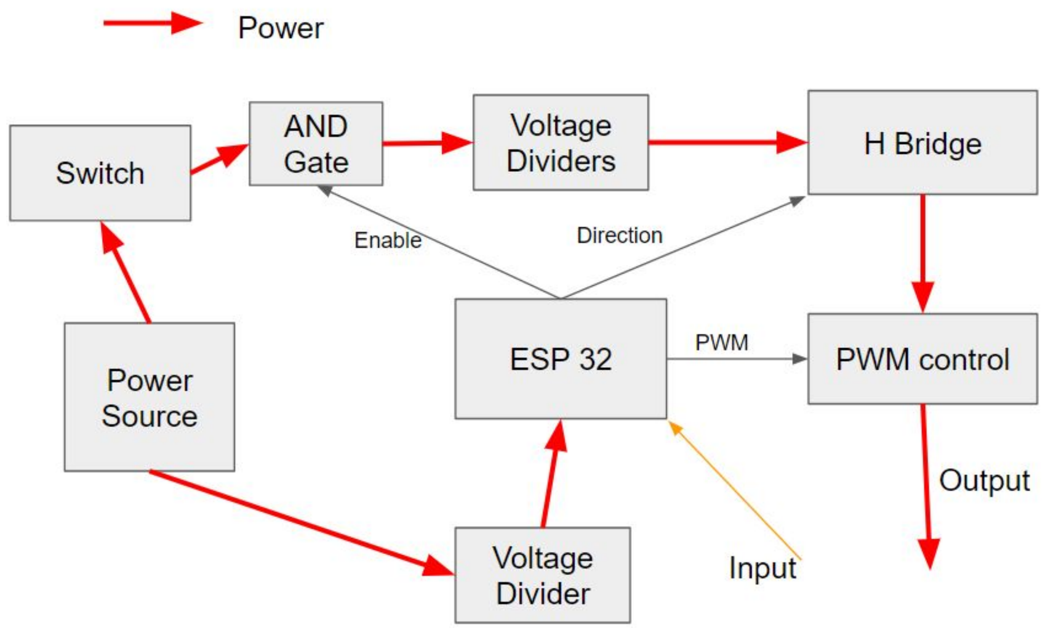
Analysis of Components

A potentiometer was used as a sample input, as we could easily set its value without worrying about real world values. A DC motor would be our output, as it would be able to show the different power levels, forward/reverse, and varying speeds

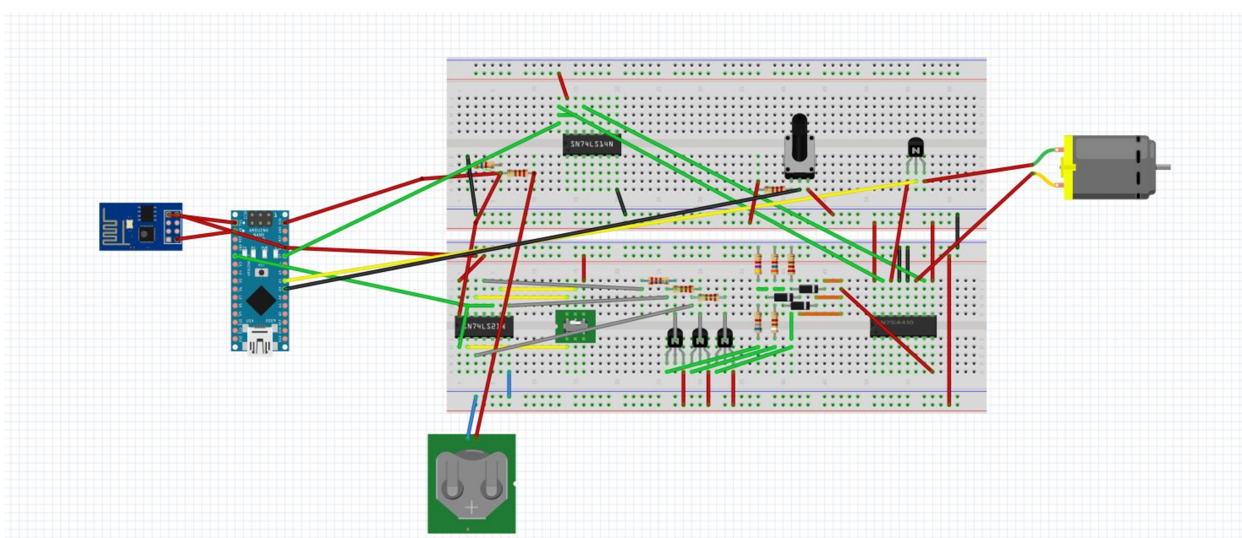
We believe the design was mostly good, other than the 2 issues of using bjts instead of mosfets, and voltage dividers instead of voltage regulators.

Design Description

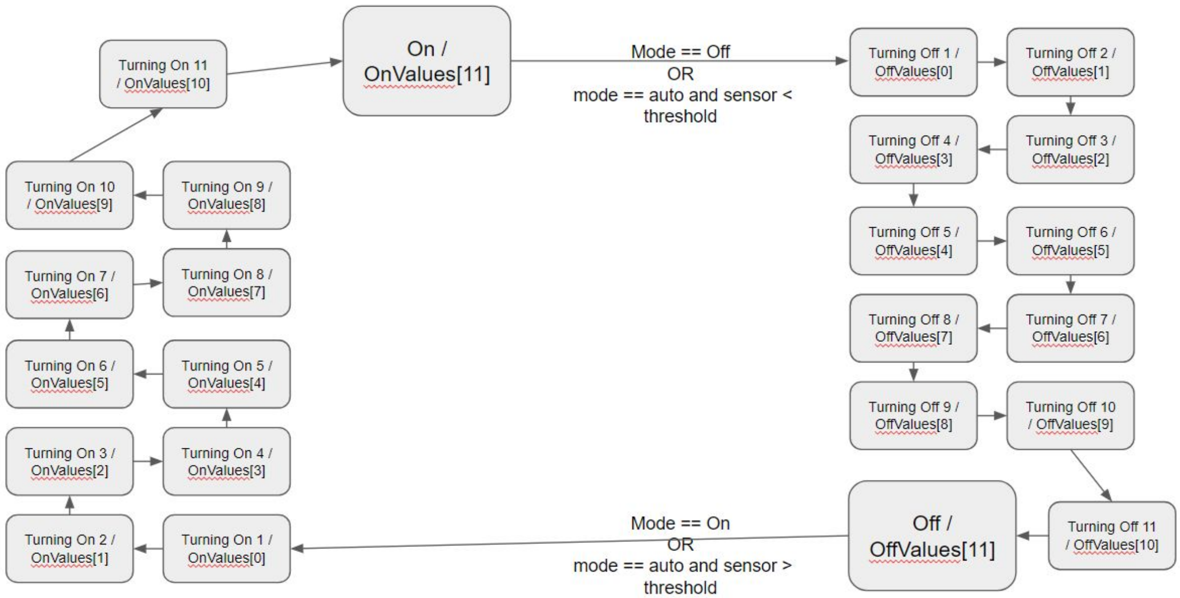
Block Diagram



Wiring



FSM



Conclusion

Lessons Learned

After working on the project, we discovered that our method for supplying power was not the most optimal. Also, the wifi was shaky, which was probably related to not having enough power.

Self-assessment

Unfortunately, we were not able to reach all the goals that we had set at the beginning of the semester. On the software side of the project, we completed all requirements and were happy with the outcome (a simplistic web interface that interacted with the ESP32 over WiFi). The ESP32 was able to parse the settings and interpolate between different output values, and properly set all the control signals. However, we encountered issues such as bad WiFi connection during our demo, and not being able to supply enough power. We learned that we should be using mosfets instead of bjts when we are driving outputs, since the bjts could not provide enough current. We also learned that voltage dividers are a bad way of converting voltages, and that we should have used a voltage regulator instead. Next time, instead of treating power signals like a data pathway, I should calculate the voltage and current drops at all points in the circuit to make sure everything is properly being powered.