EECS 113: Final Project - Building Management System

Project Report

Johnny Tran

19196029

June 9, 2021

Schematic

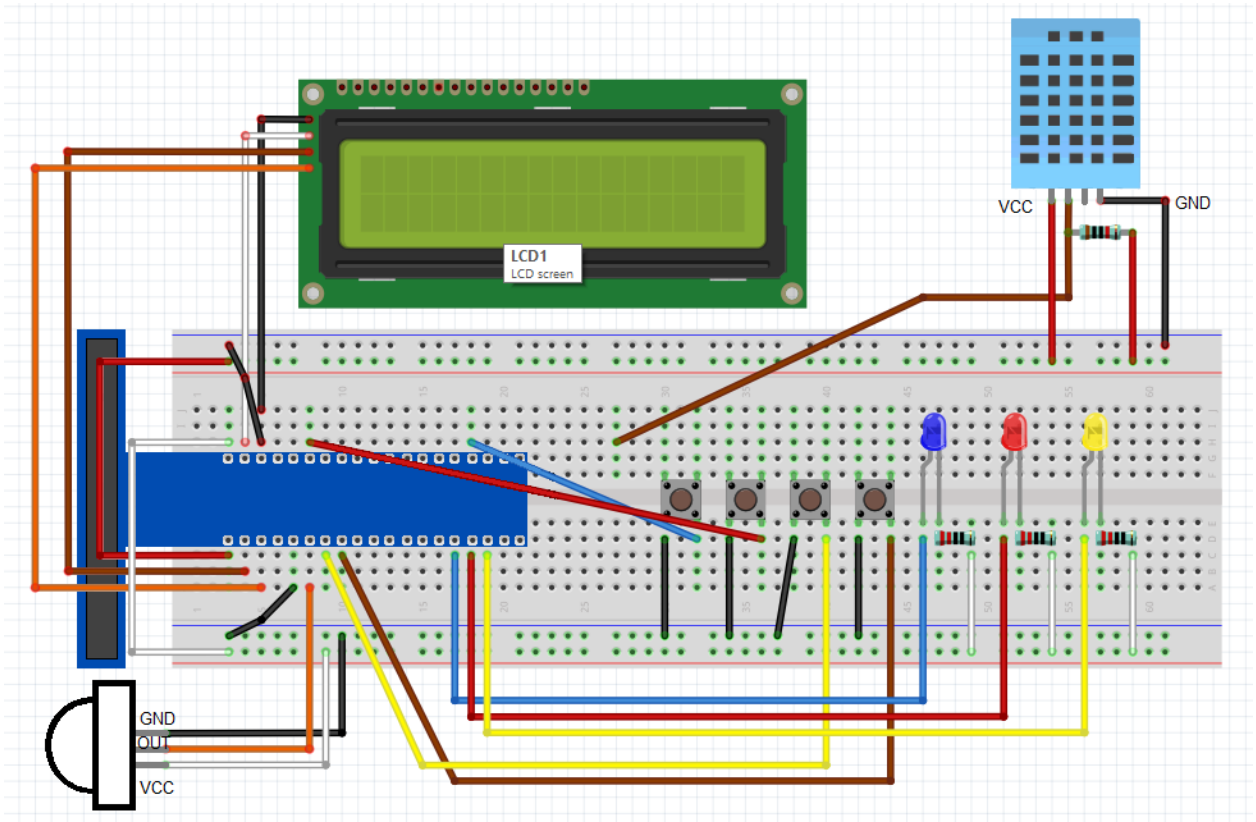

Figure 1: Components used - LCD, DHT-11, PIR Sensor, 3-4 push buttons, 3 LEDs, USB Microphone (extra credit)

Ambient Light Control

PIR Sensor is connected to GPIO 17 pin 11. The LED is attached to GPIO 13 pin 33. Both connect to 5V input with the LED connecting into a 220kΩ resistor. In code, it is detected if the GPIO input is true and sets a flag variable "occupied=True". As the **Green LED was not working**, possibly due to a malfunction, it was **replaced in the final build with a Yellow LED**. The flag is used to detect when the input of the PIR sensor is =0 (meaning no more input from the sensor) to start counting down 10 seconds. If no additional movement is detected, then it will

turn off the Yellow LED and afterwards set the “occupied flag” as false to indicate that the sequence of occupancy and leave is complete.

Room Temperature (HVAC)

The DHT-11 sensor recorded both the temperature and humidity. As **CIMIS crashed a few times and data for humidity was not received**, the TA recommended to **take the latest data** I had and use it (~75). The weather index (WI) formula was used to be displayed onto the LCD which in the main system status screen formatted “[WI] / [user_temperature]” on the top line, left side corner (e.g. WI = 78, user_temperature = 80 -> 78/80). The **program was slow** so it was only able to **take a temperature measurement every 5-10 seconds**, thus updates to the system statuses were slow. Sometimes the measure meant upon start up were half the future values (e.g. $temp_1$ = 40°F, $temp_{2, 3, 4, ...}$ = 80°F), so a flag “ignore_temp1=False" was set for the first while loop run and ended with it set as True to indicate it was ignored. The condition would be checked at the start of the second loop onwards where if “ignore_temp1=True”, it would be able to start reading and averaging out the temperatures.

Average temperature was calculated using an array “temp_arr[0]*3” of length 3 and all indices initialized at 0. An incremental counter “temp_cntr=0” was set to assign temp_arr[0], temp_arr[1], and temp_arr[2], and resetted the counter to 0 when incremented to “temp_cntr=3”. The weather index was recorded in the array before it starts to calculate the average so to not count the first 2 averages with 0 in the list (e.g. temp_arr = [#, 0, 0], [#, #, 0]), the numpy library was imported to count only the nonzero values. Round() was then used on the average to find the nearest whole number but was displayed as #.0 as an error (e.g. 78.0/80).

User input temperature (UIT) that would be adjusted using the increment and decrement buttons was retrieved using the first recorded weather index temperature to serve as the base

value the user can adjust by 3°F degrees to turn on/off the ac/heater. When “button1” (at GPIO 25 is pin 22) detected with the conditional statement that the UIT is at least 65, it will decrement the temperature by 1°F, whereas “button2” (at GPIO 18 pin 12) when UIT is at most 85 will increment the UIT by 1°F. Once the UIT is detected to be 3°F or more lower than the WI for some “[WI] / [65 ~ (WI-3)]” (meaning the user wants a lower temperature than the WI), then the AC status is set as “acStatus=’ON’ " for the LCD and the Blue LED is turned on. Alternatively when the UIT is detected to be 3°F or more higher than the WI for some “[WI] / [(WI+3) ~ 85]” (meaning the user wants a higher temperature than the WI), then the Heater status is set as “heaterStatus=’ON’ " for the LCD and the Red LED is turned on. Both the AC and Heater statuses are turned off once these conditions are no longer fulfilled.

Displaying “HVAC {AC/HEATER} {ON/OFF}” was not implemented as it was originally added the check condition for “acStatus” and “heaterStatus” portion of the code described above. It would be constantly checked in the while loops as it wasn’t checking for activation only but continuous status so it would have displayed the “HVAC {AC/HEATER} {ON/OFF}” every loop, which is a simple change but blocked the overall system status display and thus removed for consistency.

Security System

The third button “button3” at GPIO 27 pin 13 controlled the door/window status. Once pressed, it cleared the LCD of its main system statuses to display the “DOOR/WINDOW {OPEN/CLOSE}” status for 3 seconds. After this if either AC, Heater, or both statuses are on, they are turned off for 3 seconds and turned back on. This decision was made as if they were left off, as described in the section above that the continuous status check for AC and Heater would

have just turned them back on on the next while() loop since the conditions were still fulfilled for them to turn on the LEDs.

Extra Credit

A minor extra credit system was implemented that imported the datetime library to get the current system time. Upon startup while the system status is empty on the LCD display, this time is used to display the current real world time when starting the system. Alternatively the time can be found when turning off the program using Ctrl+C where it will display on the LCD that is turning off and show the real world time it turns off.

A fourth button “button4” at GPIO 22 pin 15 was implemented that used the Google API speech recognition system. This means that it has to connect to Wifi for it to connect to Google or else it will return an error. A USB microphone was purchased to connect to the USB ports on the side of the Raspberry Pi. The voice recognition works when “button4” is pressed, later prompting the user to “Say something”. At this point when it recognizes the user’s command, it will transcribe it into a string of text and find substrings for specific on/off orders for AC, heater, and door status (e.g. ‘ac’ and ‘on’ is found from transcribed ‘turn the ac on’, which will then switch its status to “acStatus=ON “ for the LCD and activate the Blue LED.

Figure 2: USB Microphone what was used to interact with Google API's speech recognition

Additional/Submission Notes

Submission:

- Extra credit is final_EC.py
- If extra credit does not run, main program submission is final.py

System

- While() loop was slow so temperature and WI was calculated every 5-10 seconds

Build:

- **Green LED** did not light up during execution, so it was replaced with **Yellow LED** in build
- Extra credit portion was not featured in the demonstration video due to time constraints (Full video was 4:50, 10 seconds below limit)
- Link for USB microphone used: https://www.amazon.com/Lavalier-Microphone-Cardioid-Condenser-Computer/dp/B077VNGVL2/