Smart Air Conditioning Project Outline

Start of project: 11/12/24
End of project: 11/20/24
Problem:
My room is not the best at insulating the environments' temperature and it is subject to change based on the outside weather. To put it simply, if it is hot, my room will be hot, and if it is cold my room will be cold. In contrast, the other rooms in the house are much better insulated and maintain a stable temperature for longer periods. During the day, my mother usually sets the house temperature to 78°F. While that's not an ideal setting for comfort, with the help of ceiling fans, we find it manageable.

The real issue arises at night. When it's time to sleep, my room often becomes uncomfortably hot, making it difficult to rest. The other rooms remain cool enough that no one else feels the need to adjust the AC. So, I'm often forced to get out of bed to lower the temperature to 75°F, just to power on the AC and cool down my room.

To solve this problem, I decided to build an automatic temperature regulator that takes over at 6 pm, lowering the house temperature to a more comfortable 75°F to ensure better sleep. Then, at 9 am, it will reset the temperature to 78°F to conserve energy during the day when everyone is awake.
Materials needed:

- Arduino Uno
    - My intro to electrical and computer engineering course required us to buy a kit that contains an arduino Uno so I will be reusing this component for this project.
- Wiring
    - Fortunately, the same kit mentioned previously, provided a lot of jumper cables that I will reuse for this project.
- Medium sized breadboard
    - Compact, but sufficient pins for connectivity.
- Single-channel Songle SRD-05VDC-SL-C Relay Module (can handle 5V direct current and 24v AC which is suitable for this project)
- RTC (real-time-clock) DS3231
    - Needed to monitor the time of day and trigger the relay based on the code.
- DHT22 temperature sensor module
    - Monitors the temperature input needed to decide whether to turn the AC on or off.
    - The DS3231 Module has an integrated temperature sensor, but it has an accuracy of ±3°C while the DHT22 has an accuracy of ±0.5°C. Therefore, we must use a DHT22 for more accurate readings.
- Braeburn thermostat
    - This is the thermostat my house uses and what I will be working with to regulate the house temperature.

Days 1-2: Project Setup & Planning

Objective:

Finalize project scope, gather materials, and plan the circuit setup. After ordering the materials needed online (the medium sized breadboard, rtc, DHT22, and relay module), it will take 3 days (until 11/15) to be delivered.
For the next 2 days, I will work on planning and researching to have a solid understanding of each component and how everything will communicate with each other to make the project work.

Tasks:

1. **Define Project Requirements**:
    - Confirm project goal: Controlling the Braeburn thermostat's cooling system based on time of day.
    - Identify key features: Time-based control, basic relay operation, and safety considerations.
2. **Component Acquisition**:
    - **Microcontroller**: Arduino.
    - **Relay Module**: To interface with the thermostat.
    - **Power Supply**: To power the microcontroller and relay.
3. **Thermostat Wiring**:
    - Understand the thermostat wiring (especially **R**, **Y**, and **C** terminals).
    - Confirm how to safely interface with the thermostat (low-voltage side).
4. **Research and Plan System Design**:
    - Research basic relay control and identify the appropriate pin configuration.
    - Decide on the best time-based logic (e.g., turn AC on until 75 degrees at night for sleep and off in the morning until 78 degrees for daytime routine).

Background Research:

I am using a braeburn non-programmable thermostat, specifically the 1020 NC.
https://www.amazon.com/BRAEBURN-1020NC-Thermostat-Non-Programmable-1H/dp/B00ECC

4W3I/ref=asc_df_B00ECC4W3I?mcid=d3ac68d6cd53346da1074383a5bd3314&tag=hyprod-20&linkCode=df0&hvadid=693570188397&hvpos=&hvnetw=g&hvrand=22824905459819059 72&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1026629&hvtargid=pla-762094679432&psc=1
This is the specifications pdf:
https://www.braeburnonline.com/sites/braeburn/files/products/specifications/1020NC%20Spec.pdf

The important information for me to note when wiring the circuit will be the terminal arrangement and setup for the thermostat. The braeburn 1020 NC thermostat has the following terminals:
C (Common): Provides a continuous 24V power supply.
Rh (Transformer): Provides 24V when the heating system is powered on.
O (Reversing Valve): Energizes the reversing valve to switch between heating and cooling modes.
Rc (Transformer): Provides 24V when the cooling system is powered on.
W1 (Heat): Controls heating (if using a heating system).
Y1 (Cool): Controls cooling (if using an air conditioner).
G (Fan): Controls the fan.

For my specific problem, I will only need to interfere with regulating C (common) and Y1 (cooling). I plan on using the relay module to control when the system will need to power on. When the system is cooling, the thermostat will be sending a signal from the Y terminal. I can use the relay to connect/disconnect this terminal to trigger the AC.

On the relay module, I will connect the Y terminal to the NO (normally open). And I can connect the common on the relay, to the c terminal to complete the circuit.
The relay will close when activated by the microcontroller, effectively sending the signal to the thermostat to turn on the cooling system.

I will use a yellow LED to portray the Y1 being activated and the AC turning on.

How the relay works:
The relay has a coil in between a wire which can switch from normally closed to normally open. When the coil becomes energized as a result of either end of the coil providing a voltage, the wire switches to the normally open.

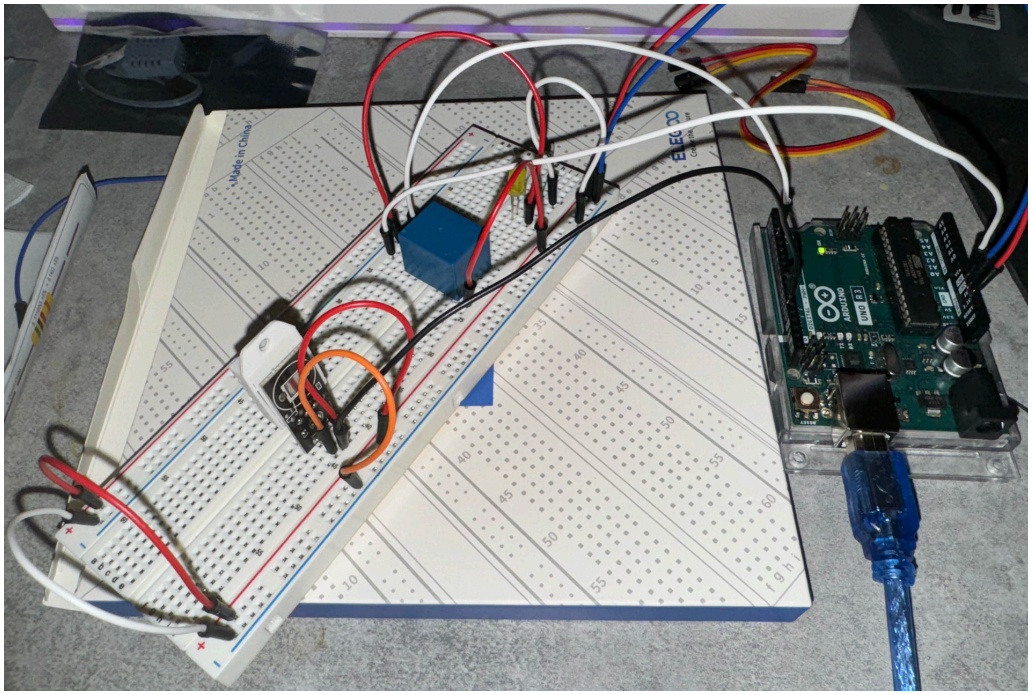
Day 3-4: Circuit Design & Initial Coding

Objective:

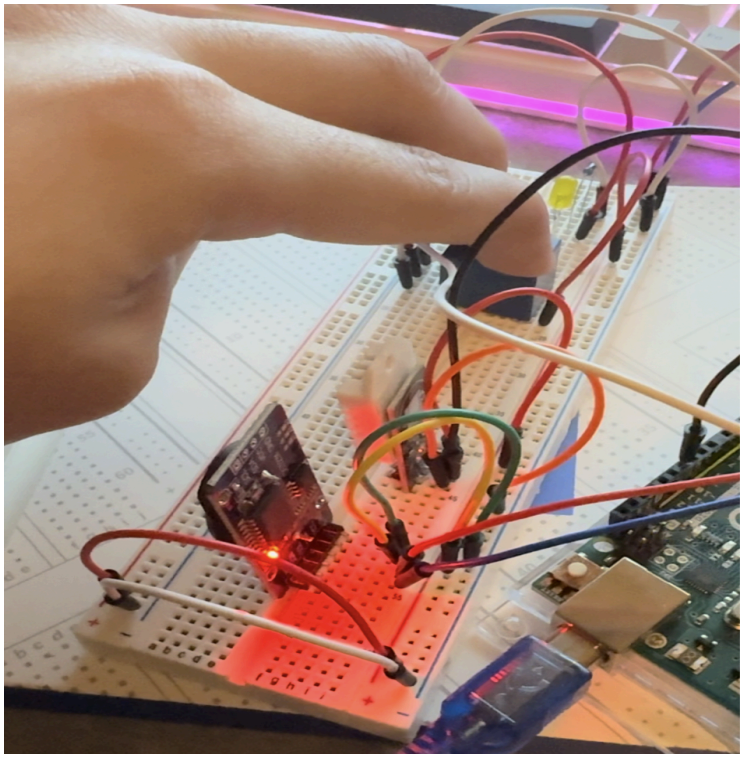
Build the initial circuit and write basic code for time-based control.

Tasks:

1. **Circuit Assembly**:
    - Connect the relay module to the Arduino.



    - Wire the relay module to the LED, where normally open is connected to VCC, and the common is connected to the positive of the LED. One end of the coil is connected to ground, and the other end is connected to a digital pin on the Arduino. Then, the LED is closed by a resistor. Essentially, this will allow control of the relay using the Arduino, and the relay is setup so giving power to an end of the coil will cause the flow to go to the normally open, which will result in an closed circuit and turn on the LED.
    - Connecting the DHT22 sensor was simple. Provide it a current, with the digital output intercepting the flow and connect the output to an digital input on the Arduino.



- To connect the RTC module, I connected the Vcc and ground with the breadboard. The SDA to A4 , which is the data line on the arduino, and the SCL to the A5, which is the clock line. This resulted in accurate time readings.

2. **Programming**:
   - For the programming, I had to consult the internet to figure out how the DS3231 and DHT22 could be implemented.
   - For the DHT, the library was simple enough. I had to initialize the DHT sensor, and then I was able to read the values when I needed to by using the function: dht.ReadTemperature();. The values were in celsius, so I added a conversion.
   - For the DS3231, the library had a bit more complicated functions to work with. For starters, I had to create the object DS3231 to be able to manipulate the values, which I did not have experience with but it seemed intuitive enough. Afterwards, I had to check for a response from the module so I used if(!rtc.begin()). To set the time, I used a function from the RTClib.

Code:

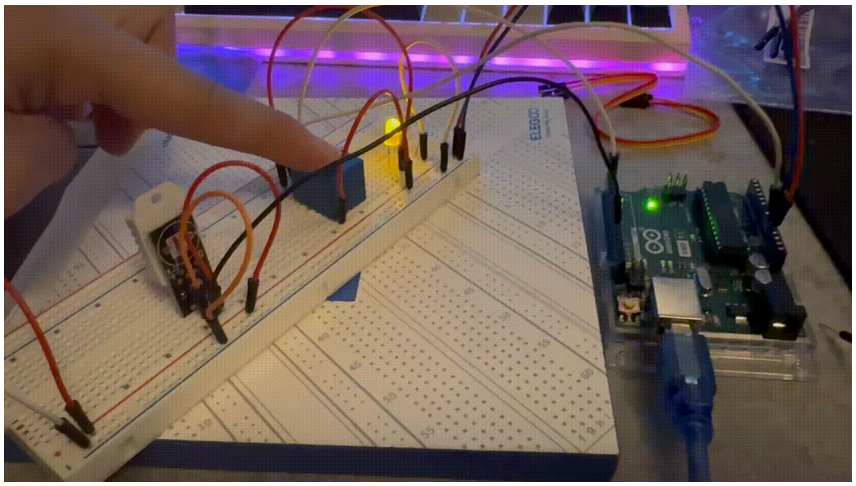
temperature_regulator.ino

```ino
#include <Wire.h>
#include <RTClib.h>
#include <DHT.h>

RTC_DS3231 rtc;  // Create RTC object

const int relayPin = 2;  // Relay is connected to pin 2
const int dhtPin = 11;   // DHT22 sensor connected to pin 11

DHT dht(dhtPin, DHT22);  // Initialize DHT22 sensor

void setup() {
  // Start serial communication for debugging
  Serial.begin(9600);

  // Initialize RTC
  if (!rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);  // Halt execution if RTC is not found
  }

  // Set the RTC time to the computer's current time
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));  // Set RTC to compile time

  Serial.println("Time set to computer's current time");

  // Initialize the DHT22 sensor
  dht.begin();

  // Set relay pin as output
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);  // Ensure relay is off initially
}

void loop() {
  // Get the current time from the RTC
  DateTime now = rtc.now();
```

temperature_regulator.ino

```
38
39    // Read temperature and humidity from DHT22
40    float temperatureC = dht.readTemperature();  // Read temperature in Celsius
41    if (isnan(temperatureC)) {
42      Serial.println("Failed to read temperature from DHT sensor.");
43      return;  // Exit the loop if temperature reading fails
44    }
45  float temperatureF = (temperatureC * 9.0 / 5.0) + 32;
46    // Print current time and temperature to Serial Monitor
47    Serial.print("Current time: ");
48    Serial.print(now.hour(), DEC);
49    Serial.print(":");
50    if (now.minute() < 10) {
51      Serial.print("0");
52    }
53    Serial.print(now.minute(), DEC);
54    Serial.print(":");
55    Serial.println(now.second(), DEC);
56
57    // Print temperature in Celsius
58    Serial.print("Temperature: ");
59    Serial.print(temperatureF);
60    Serial.println(" °F");
61
62    // Check if the time is past 7:00 pm and before 9 am, and if it is, check if it is above 75 degrees and if it is, turn the relay on
63     if ((now.hour() >= 18 || now.hour() < 9) && temperatureF > 77.5) {
64      // Turn on the relay if conditions are met
65      digitalWrite(relayPin, HIGH);
66      Serial.println("Relay ON (Time is between 6 PM and 9 AM and Temperature > 75°F)");
67    } else {
68      // Otherwise, turn off the relay
69      digitalWrite(relayPin, LOW);
70      Serial.println("Relay OFF (Conditions not met)");
71    }
72
73    // Wait for a second before checking again
74    delay(30000);
75  }
```

3. **Test the Basic Relay Control**:
    - Test the relay with a simple LED circuit first to ensure it turns on and off correctly based on time.



Deliverables:

- Working circuit with relay control logic.
- Basic code for time-based relay control.

Days 5-6: Integration with Thermostat & Testing

Objective: Test with thermostat to see if it turns on when intended

Tasks:.

1. **Testing with Thermostat**:
    - Test to see if the temperature readings on the DHT22 are a good enough match with the reading on the thermostat (e.g., turning the cooling system on between 9 AM and 6 PM).
        i. The readings on the DHT22 were off by about 2 degrees. When the thermostat read 75 degrees, the DHT22 read on average 77.3 degrees.
        ii. Because of this, I updated the code to account for the 2 degree difference, but I can assume inaccuracies.
    - Check to see if the relay turns on when intended (when the DHT22 reads over 77.5 degrees and it is past 6 pm)
        i. Worked as intended. The LED turned on based on conditions.

Results and Conclusions

Results:

Results:

After integrating all the components, I did a number of tests to make sure everything was running as it should. The main goals for this project were controlling the AC depending on the time of day and temperature reading to maintain comfort in my room while minimizing the expense of energy consumption.

Time-based Relay Control:
The relay was triggered accordingly through reading the time of day from the DS3231 Real-Time Clock module. In this system, it turned the relay on at 6 pm to eventually lower the temperature to 75°, and then off at 9 am to go back up to 78°.
Time was correctly synchronized. It kept the proper time in RTC and changed the activation of the AC accordingly.

Temperature Readings with DHT22:
The DHT22 sensor provided current temperature readings, even though it had an average offset of about 2°F from the thermostat.
This discrepancy was accounted for in code, where the system continued to run as expected-turning the cooling system on above 77.5°F room temperature and turning it off at thresholds set.

Indication by LEDs:
An LED on the relay was used to visually indicate if the system was on and actively controlling the AC. It would light up when the relay turned on, giving immediate feedback on the operation of the system.

Conclusion:
The project successfully achieved the goal of an automatic temperature regulation system that can, with the help of a time condition and a temperature condition, control the AC. Accurate time tracking in this system was enabled by the DS3231 RTC module integrated into it, while the DHT22 provided reliable temperature data. The relay controlled the thermostat's cooling function in such a way that this adjustment of room temperature occurred at just the right times without human intervention.

The system was working right in the following conditions:
Temperature goes above 77.5°F and it is after 6 pm, with the system turning the AC on and cooling to 75°.
AC is turned off in the morning while the temperature reaches desired threshold of 78°F, when the system resets at 9 am.
One of the challenges was the temperature reading from the DHT22 and the discrepancy that the thermostat had; the code, however, was changed to include this 2°F offset. Although this calibration worked well, there are slight inaccuracies due to limitations inherent to the DHT22 sensor.

Future development:
More accurate calibration of the temperature sensor.
More accurate and precise temperature sensor for better system performance.
I could perhaps connect a EPS32 or a wifi microcontroller so that I am able to connect it to the house wifi. Then, connected to the house wifi, I can maybe open up its IP and regulate the temperature from there.

In general, this was a successful project with the functional solution it provided for my room's temperature control while conserving energy whenever the AC was not needed.