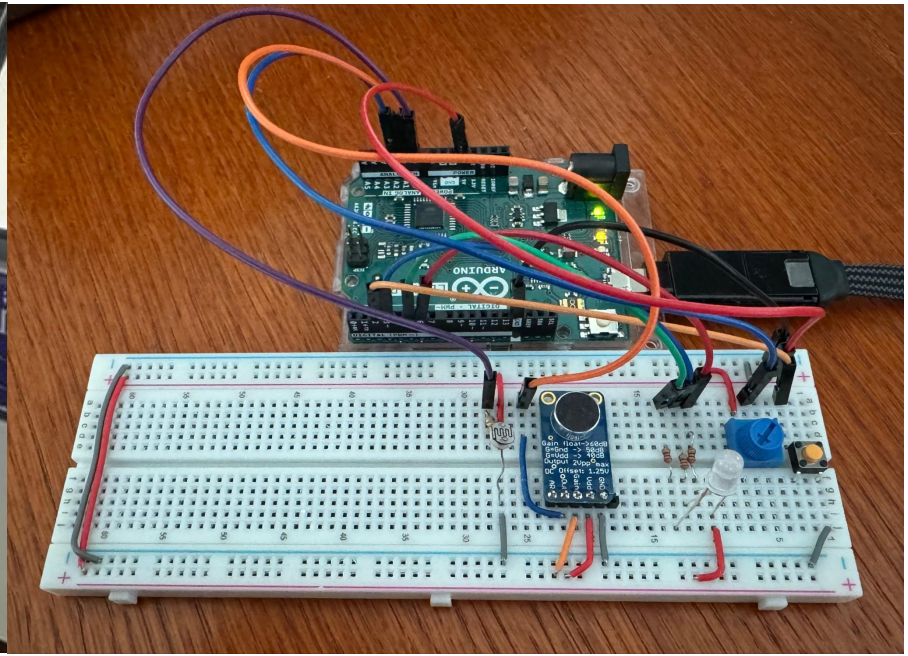
A1: Responsive Night Light

Rahul Surti

Night Light With Cover

Circuit View



Links

Demo: rsurti_A1.MOV

Source Code: nite_light.ino

Overview of Night Light

- Debounced button that toggles between modes:
  - **Mode 1: Crossfading RGB LED** with brightness determined by photoresistor. The darker the room, the brighter the LED.
  - **Mode 2: Sound Based Lighting** where the microphone value determines the brightness of the LED. Uses the highest peak to peak reading in a given 40ms window to determine the output. Even though peak to peak reading could higher than 1.0V, I truncated the reading to just be 0.0-1.0V to make the mic more sensitive.
  - **Mode 3: Manual Lighting Using Potentiometer** where rotating the potentiometer smoothly transitions between red, green, and blue. I converted the single 0-1023 potentiometer value into a 3 component RGB value, where red peaks at 256, green peaks at 512, and blue peaks at 768, and both 0 and 1023 turn the LED off.
  - **Mode 4: Hex Code Input From Computer** where I parse a string buffer from serial input, validate that it is a hex code, and set the RGB value of the LED accordingly.

Design Process

1. Started by setting up the crossfading LED from A1 Midpoint
2. Added debouncing button functionality and ability to transition between button states
3. Followed the makeabilitylab lessons to build circuits for the photoresistor, microphone, and potentiometer. Adapted the code for each to output to existing RGB LED.
4. Researched how to read serial input from computer, wrote some validation code to verify the input was 6 digit hex code, then set the RGB LED value accordingly.

Key Struggles and Challenges

- Reading the datasheets by themselves was difficult. I found the makeabilitylab lessons to be the easiest to learn from to design and build the circuit.
- I struggled with debugging via hardware itself- I knew something was wrong with the RGB LED since it was not behaving as expected, even when using only 5V and ground connections. It wasn't clear until we discovered in class that the RGB pins follow a non-standard format.
- It was a challenge to organize the breadboard to be understandable at a glance via color coding wires and intentional component layout
- It was a challenge to write the code in such a way to avoid using actual delays. Instead, I leveraged storing state and reading the current `millis()` time to determine the timing window.

Learnings

- How to interface both simple and complex sensors with Arduino to manipulate LEDs, easily can extrapolate to other output components.
- How to use to both `Serial` input and output
- How to manage state in an Arduino application
- How to properly use different timing windows for multiple components simultaneously
- How to neatly and intentionally organize breadboard