Light Sensor Study Companion PCB

Yilan Liu

Last updated: January 11, 2025

Overview

As part of a project I did at UChicago, I designed and soldered this printed circuit board (PCB) using a few key components, including a ATSAMD21G18 mirocontroller, a light dependent resistor (LDR), and a pushbutton to toggle modes. The board is powered with 3.3V - 5V via USB-C and the coding was done in the Arduino IDE.

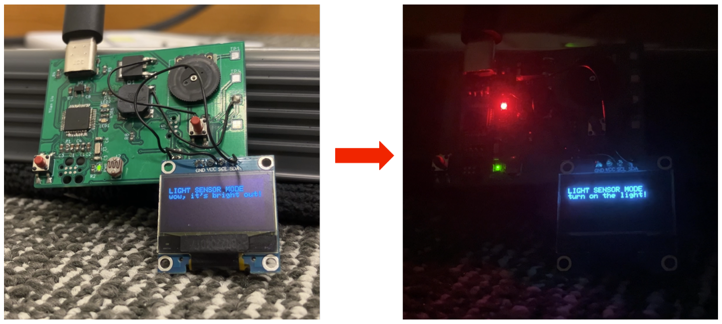
There are two modes that the PCB can toggle between: light sensor mode and productivity mode.

The light sensor mode reminds you to get up and turn on the light when the brightness in the room decreases to a certain level. A buzzer sounds to alert you, a red LED lights up, and the OLED displays a message to let you know.

The productivity mode is really just a "positive affirmations" mode in disguise where you can display motivating text through the OLED display.

Light Sensor Mode

Objective: Shine red LED, sound buzzer, and display a reminder on the OLED display once the darkness in room reaches a certain level.



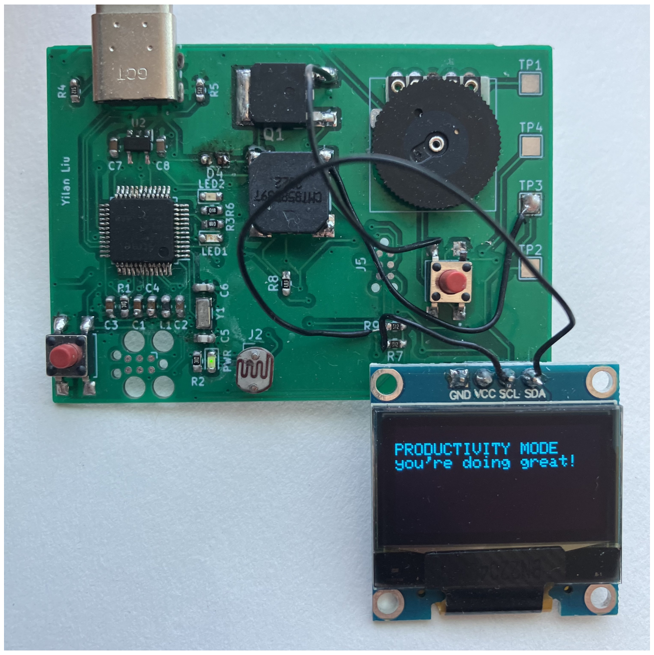
I wired the light dependent resistor (LDR) to an analog input pin of the ATSAMD21G18 microcontroller, a red LED to a digital output pin, and a buzzer to a digital output pin.

Logic in code: If the value coming in from the input pin connected to the LDR is less than or equal to a certain value, I write a HIGH value to the digital pin connected to the red LED and also to the digital pin connected to the buzzer. Finally, I change the text of the OLED display to “turn on the light!”.

Productivity Mode

Objective: Display message of choice on OLED display.

Logic in code: Once the button is pressed to change to this mode, I erase the existing text on the OLED display and replace it with “PRODUCTIVITY MODE” followed by an encouraging message.



Pushbutton For Changing Modes

I wired the mode button to a digital input pin of the microcontroller and then enabled the internal pull-up in the code.

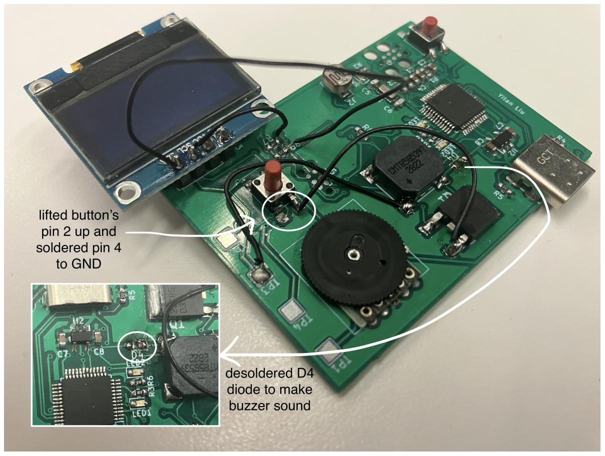
Logic in code: I keep a counter of how many times the button has been pressed by tracking the number of times input value changes from HIGH → LOW and LOW → HIGH and dividing by two (the division by two is because every time the button is pressed, there is a change from HIGH → LOW and another one from LOW → HIGH). Then, I mod the value stored by the counter by two to tell me which mode should be displayed on the OLED display.

Fixes

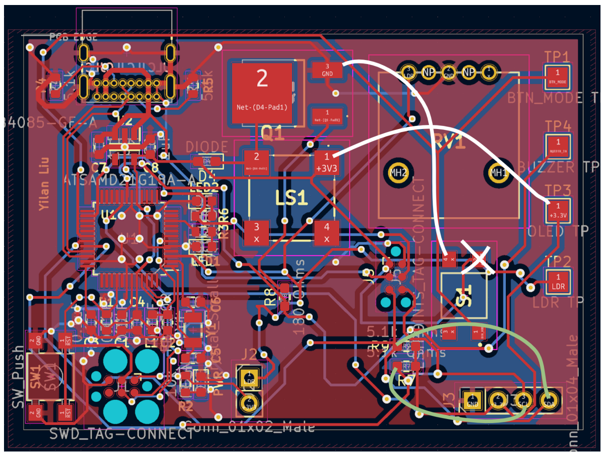
There were so many fixes I had to do! Ironically though, the open heart surgeries I had to do on my board turned out to be my favorite part of the project.

To start, I made the rookie mistake of accidentally making two variable names for the 3.3V power source in the KiCad schematic: “+3V3” and “+3.3V”, the latter of which was not connected to any power source. Lucky for me, though – I had a test pad that was connected to “+3.3V”, so I just connected it to the “+3V3” pad of the buzzer using an external wire.

Another mistake I made was connecting the mode button completely incorrectly in the KiCad schematic. From the datasheet, it’s clear that when the button is pressed, pins 1, 3, and 4 short.


lifted button’s pin 2 up and soldered pin 4 to GND
desoldered D4 diode to make buzzer sound

However, I wired up pins 1 and 2 in the schematic (facepalm), and also neglected to add a pull-down nor pull-up resistor. I didn’t realize these mistakes until I had already soldered on the button, so here’s what I did instead: I got as much solder as I could off of pin 2 (which was connected to +3V3), pried the leg up, then connected pin 4 (previously floating) to the GND pin on the Q1 NPN transistor using an external wire. Finally, I enabled the internal pull-up resistor in the Arduino code.



The third mistake I made was accidentally burning off the traces for the SCL and SDA lines of the OLED display. It was a simple fix, though – all I had to do was connect the SCL and SDA pins to the SCL and SDA pads of the two resistors connected in those same tracks using two external wires.

The final fix I did was to remove the D4 diode, which was recommended in the datasheet of the buzzer to be placed in parallel with it. When I had the diode in, the buzzer would make a quiet clicking sound instead of buzzing, and after debugging for a bit, my TA and I concluded that it had to be the diode causing problems. We desoldered it, and the buzzer started working! Strange.

Lastly, here is the schematic for the project: