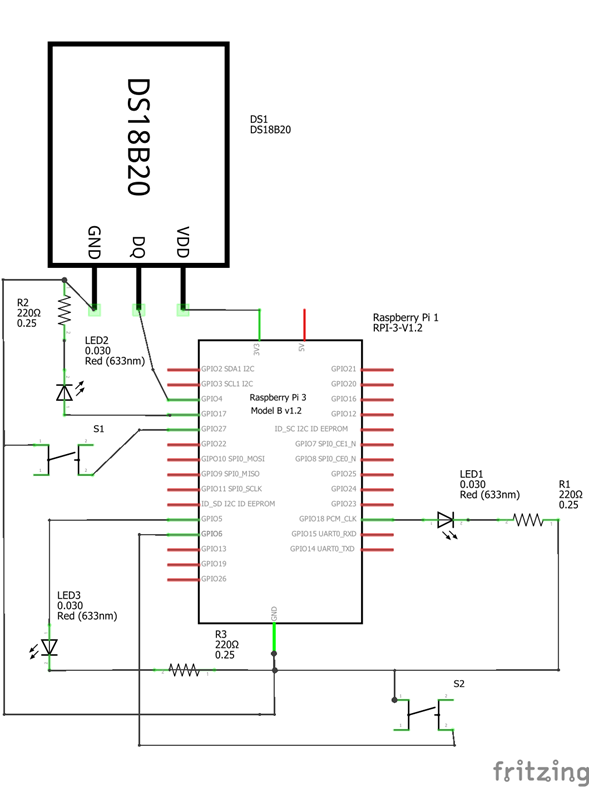
Project 1

**Node Design**

For this project, as I am not in a group, I did not want to add the complexities of making the buttons work in tandem. So, for this project I opted out of the third requirement for Step 2 in the project prompt. The platform that I am working with is the Raspberry Pi model 3B (Referred to as Pi from now on). My node setup was very straightforward. For the following breakdowns all the LEDs that were used were red, all the resistors were 220 ohms (unless otherwise specified), all grounds were connected to the ground bus on my breadboard that was connected to a ground pin on my Pi, and finally 3.3V power from my Pi to the power bus on my breadboard.

For the Pulse Width Modulation (PWM) LED, I have a cable connecting a GPIO pin to the anode of an LED. On the cathode side there is a resistor and finally a cable connecting the circuit to ground. I also have a push button with one side connected to a GPIO pin with a cable and the other side of the button connected to ground with a cable.

For the blinking LED, I have the same setup as above since the difference in functionality will largely come down to code.

For the third part of the project, as I am using a Pi, I had to use the DS18B20 thermal sensor. This sensor requires a 4700 ohm resistor, but it came with a module that includes the resistor built in. This sensor has three prongs that I plugged into my breadboard. These prongs are GND (ground), VCC (power), and DAT (data output). Each of these corresponds to a wire that the sensor has, and I connected each to their respective place on the module. From my Pi, I connected a GPIO pin to the DAT line, 3.3V power to the VCC line, and GND line to ground. After the sensor was connected, I also added another LED to the board in the exact same manner as the above mentioned.

**Code**

For my project I decided to code in Python as I honestly haven’t got much experience in it and I thought this would be a fun way to learn and it didn’t require downloading compilers or

messing with POM files. I decided to use the GPIO Zero library for my project as it made the most sense to me and is pretty intuitive. I used the documentation for GPIO Zero on their website while learning and creating my own code so there may be some similarities in calls to specific components.

For my LED application, I imported LED, PWMLED, and Button from the gpiozero library, as well as time from the python base library. I sectioned off my code into three parts for the blinking functionality, the PWM functionality, and the main program section.

For the blinking LED, I first declare the button\_blink and led\_blink variables that hold the GPIO pins of those components. I then initialize a variable clock\_start by getting the current time. I initialize another variable blink\_start to the value of clock\_start and a variable time\_delta to 0. I initialize a flag variable clock\_paused to False and cycle\_time\_passed to 0. Lastly, I initialize a variable blink\_frequency to 1. There are two functions that I created for the blinking function of this application called change\_speed and stop\_start. Change\_speed will decrement the blink\_frequency variable by 0.21 seconds and will reset back to 1 once it goes past zero. Stop\_start checks if the clock is paused by checking the clock\_paused flag. If the clock is paused, it will set the clock\_start as the current time minus the time that has already passed in the cycle when the clock was paused and then resume the clock. If the clock isn’t paused, the time that has already passed in the cycle time is saved into cycle\_time\_passed and the clock is paused.

For the PWM LED section, I declare led\_pwm and button\_pwm with the GPIO pins for those components as well as a dc\_coefficient that is initialized at 4. There is a single function for this section called change\_duty\_cycle that decrements the dc\_coefficient by 1 and back to 4 when it’s past 0.

For the main function portion of my code, I have everything inside of a while True loop. Inside that loop I set time\_delta to the current time minus the clock\_start which gives me how much time has passed since the clock was started. Then I set blink\_delta to the current time minus the blink\_start variable which gives me the time that has passed since the last blink cycle. The blink\_delta is then checked against blink\_frequency and if it’s greater than or equal, it toggles led\_blink and sets blink\_start to the current time. This effectively makes the blink\_frequency as the time in seconds the light will be both on and off during the cycle. I have button handlers for each function. When button\_blink is pressed it calls the function stop\_start which pauses or resumes the clock. When button\_pwm is pressed it calls the change\_duty\_cycle function which decrements the dc\_coefficient by 1 between 4 and 0. The duty cycle of the led\_pwm is then set to the dc\_coefficient multiplied by 0.25. This results in the LED having five settings for brightness 1, 0.75, 0.5, 0.25 and 0. The main program then checks if the clock is paused and continues to the next iteration of the main loop if it is. If not, the program continues and checks if the time\_delta is less than 5. This is the number of seconds that the blinking light will stay at the same speed for. For instance, if you change this value to 10 the blinking light will stay at its current blink\_frequency for 10 seconds. If the time\_delta is less than this number, the program continues to the next iteration of the main loop. If both the clock is not paused and sufficient time has passed in the cycle, the program will call the change\_speed function which will decrement the blink\_frequency and make the led blink faster for the next five seconds. Lastly, clock\_start is set to the current time so that we can keep track of how many seconds have passed in the next cycle.

The code for my thermal sensor is fairly straightforward. I save the GPIO pin the led is connected to. I declare a temp threshold that will be the deciding temp for when the LED lights up. I then access the sensor per the documentation given for gpiozero to and set it to write to a file. I have a function to get the raw temperature from the sensor as well as a device to convert the given raw data into useable data that accepts a parameter that will allow me to use this function for a future project and return either Celsius or Fahrenheit. For this project it is defaulted at Fahrenheit. The main function then loops through a while true loop to get the temp, check it against the threshold and turn on the led if it is above threshold or off it is below. The program then sleeps for 1 second and the loop repeats.

**Issues**

I did not run into many issues with this project. I had a minor issue with one component that I received in the kit I bought for my Pi but it was just a matter of swapping out that component for a functional one. The biggest issue I had was when learning how to get these buttons and LEDs to function at first a lot of the documentation and tutorials implemented the sleep function. When I tried to use this function with both of my buttons and LEDs setup I realized that there was going to be a problem because of the way sleep works. From my understanding, sleep in gpiozero makes the whole program stop for the given amount of seconds which means that while the sleep is being used, I can’t register button presses and if the timing would be hard to predict with everything. Because of this, I decided to use the time library to make my own simple clock to manage and keep track of time. Another issue I had is when I tried to just add my thermal sensor to the same application as my LEDs. It was throwing off my blinking LED and making it blink at a consistent time no matter the setting. I did not debug it much and I suspect it had to do with the calculations I was doing with the data from the sensor but I read in the project prompt that part 3 was to build an application for the thermal sensor itself so I just made a new application.

**Takeaways**

Overall, I learned a lot doing this project. For one, I learned how fun this can be. I already have ideas for a bunch of things I could do in my own time. On top of that, I definitely learned a bit about python and its capabilities. I got more practice with connecting to my Pi over SSH as well as setting up an SSH key for my desktop so I can connect to my Pi easier. I got more comfortable with electrical circuits and more confident in not only building the circuits but understanding what is needed for each component I want to connect. Lastly, I think the biggest takeaway that I got is that I need to really get into the documentation to decide if there are functions that exist that are suited for what I want to accomplish or at least give me an idea of how to do what I want with my own code.

Diagram, schematic

Description automatically generated