**Task A [20 points]**:
*Build the proper circuit to operate the DC motor as described. Share an image, video, or schematic of your circuit via GitHub.*

I did not include a separate video for Task A, as it is inherently demonstrated in the video for the final iteration of the assignment.

**Task B [50 points]**:
*Write correctly operating C code to perform all three options listed in the discussion section. Be sure to include CTRL-C as a valid means for safely exiting the program. Share your C code and a video of your running project. The video may contain a demonstration of all parts of the assignment in operation, thereby requiring only one video.*

I did not include a separate code/video for the first two parts of Task B, as they are inherently demonstrated in the video of the final part of the assignment. In order to function in a binary way (on/off), the inputs for 'off' and 'high' could be used for OFF and ON commands. The code for the final version could be slightly altered to indicate to the user inputs for ON and OFF, but this is basically the same thing as having inputs for 'off' 'low' 'medium' and 'high', as I included in the final version which is displayed in the video(s).

I will also note that I included two separate videos, one for the user interface and inputs that control the hardware, and another of the actual motor and LEDs operating based on those inputs.

**Task C [30 points]**:

Answer the follow questions:

1. **The PWM interface of the GPIO pins allows for analog frequencies to be sent from the Pi to the device. Why would the Pi not allow reading analog frequencies on a GPIO pin interface?**

The Raspberry Pi only has digital pins, meaning it cannot actually read OR send analog signals, but only HIGH or LOW signals. In this assignment, we learned how to imitate sending an analog signal using PWM (Pulse width modulation), by adjusting the width of the pulse being sent by the pin to the hardware, which effectively changes the voltage being sent by the pin to simulate an analog signal. In reality, however, we are just adjusting the length of time or frequency with which the pin is alternating between a LOW and HIGH signal to change the apparent voltage in a seemingly analog way.

The Pi cannot read analog frequencies because the pins are digital and are still only able to read HIGH and LOW, despite being able to approximate sending analog signals with the PWM interface. In order to actually read analog signals, we would need to use a digital-to-analog converter.

2. **Motors have diodes in the circuit to prevent the flow of electricity in a direction that is not desired to protect the circuit. What might we need to be able to do to allow the basic DC motor to operate forward and backwards? How would you propose to address that need in software, ignoring the hardware requirements?**

In order to allow the motor to operate both forwards and backwards, we would need a way to change the direction of the flow of the current through the motor. If this were possible on the hardware side, we could change the software to be able to send signals to the motor from one of two different pins— by sending a HIGH signal from pin A and a LOW signal from pin B, the current would move through the motor in one direction, turning the motor that way. Then, if we stop sending a HIGH signal and instead send a LOW signal from pin A, and change pin B to send a HIGH signal, hooked up to the other end of the circuit, we could send the current in the other direction, which would turn the motor in the opposite direction.

Additionally, we could use the PWM interface with both of these pins to send signals of different intensities, allowing us to simulate an analog signal and adjust the intensity of the current in either direction.

I imagine we would also need to ensure that at any given moment, at least one of the pins is in its LOW state, because I am not entirely sure what might happen if we tried to send high signals from both ends of the circuit, but that seems like it has the potential to be problematic.