Objective of the Day:

- Show 2 images and combine them essentially
    - 1 image is a face
    - 1 image is a texture
    - Can control the amount of overlap between the two
- Texture superimposed on the two triangles
- Because images are being used, he uses an open source library called SOIL
    - It is only for opening and manipulating images
    - Examples in class are only using open source libraries
    - SOIL is integrated in the Linux distribution itself

Discussion of Code:

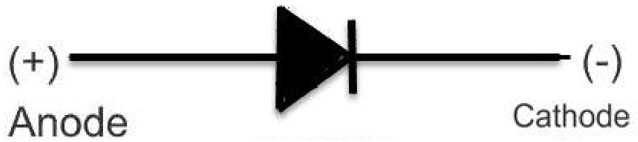
- Mix_value is 0.2f ← Very important to note
- Initial set up is same as before in order to open the GL Window
    - All the code from Initial to the shader is essentially the same
    - Shader is changed a little bit
- Generate 2 texture tabs that we will later bind to the images on the triangle
- Generate 1st texture:
    - We do GenTextures: assigns texture to 1st image
    - Textures are 2D because the images are 2D
    - How to wrap the texture around the edges:
    - How to have the texture when there is no direct mapping
    - We are opening the image and doing the width and height of the image and having it load as an RGB
        - Take the image and convert into a texture map that we will later use
        - GL_TEXTURE_2D for the image
        - glGenerateMipMap → A mipmap is a hierarchical image that you can use at run time that determine how to run and load
    - After you will free image
    - After image is free then we will unbind the image
    - Bind the texture, specify the parameters, then unbind the texture
- Same code applies as is for the 2nd texture (the face)
- Specify how to map the images on the geometry
    - We have the positions and colors as previously done for each vertex
    - Now we do the textures for that as well
    - Indices array remains the same as we have 2 triangles
    - Buffer object remains the same except for the specification for the attributes
        - All of type float, and 3rd parameter is false, 4th parameter is of 8 because we need 8 floating point values before going onto the next vertex
        - Position starts at 0, 3 floating point values
            - Color starts at 1, 3 floating point values
            - Texture starts at 2, 2 floating point values
    - The Display Loop

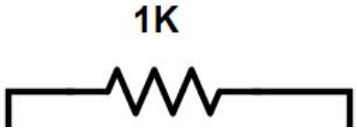
- Use our shader program
- Bind the texture
    - activeTexture(GL_TEXTURE0)
        - First texture
- Pass the mix_value to the shader
    - Used to change its behavior on how to display the data

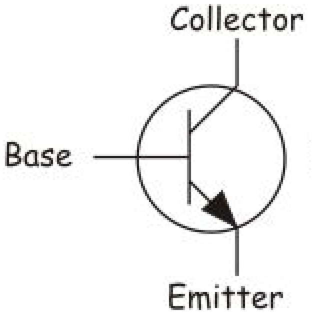
- Shader.vs
    - Layout
        - Position vector3
        - Color vector3
        - Texture vector2
    - Output
        - Color
        - Texture coordinate
    - For texture coordinate, we do x, and then 1 - y coordinate
- Shader.frag
    - Input:
        - Our_color //output of shader.vs
        - Texture coordinate //output of shader.vs
    - Output:
        - Color vector4
    - Uniform
        - Data type to allow the shader to communicate with the data
        - Lets the GPU talk directly to the main program without sending an array object
    - For our texture 1, we do the TexCoord
    - For our texture 2, we do 1-TexCoord.x, TexCoord.y
    - Mix_value changes based off the input from the keyboard
- Window Behavior
    - Escape key remains the same
    - Up key:
        - We will add 1
        - If goes at 1 we will stay at one
    - Down key
        - We will subtract 1
        - If goes below 0, we will stay at 0

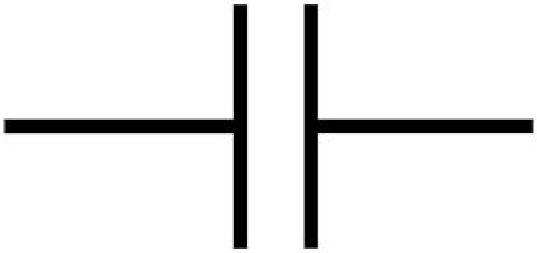
Elements of a Circuit:

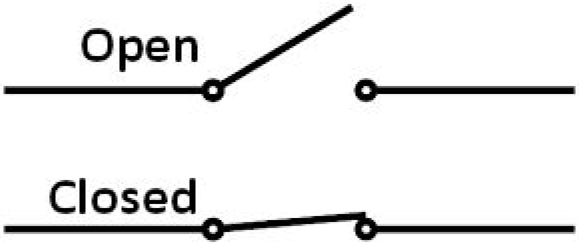
- LED:
    - had the anode → connect to positive
    - Cathode → connect to negative

- 

- Resistor:
  - 

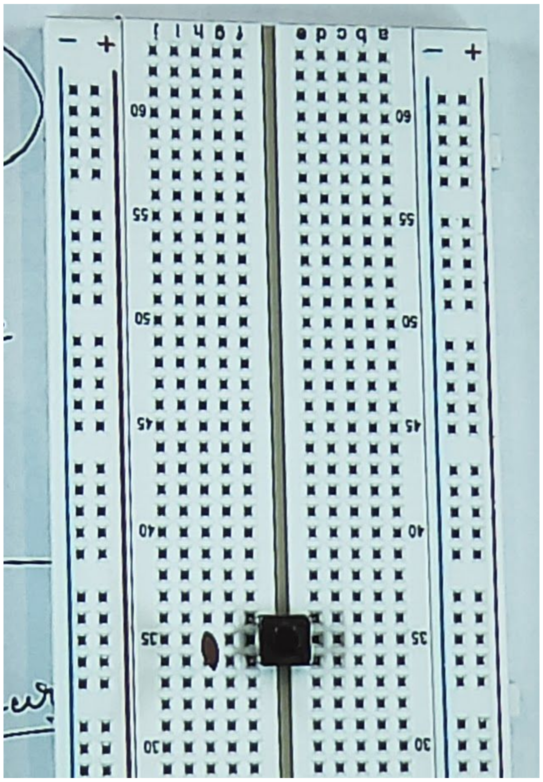
- Transistor:
  - 

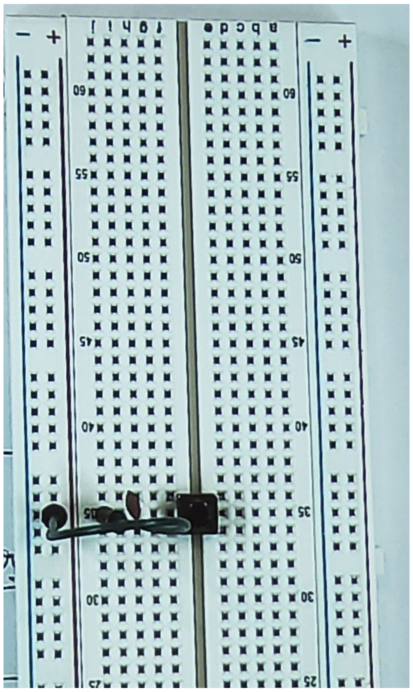
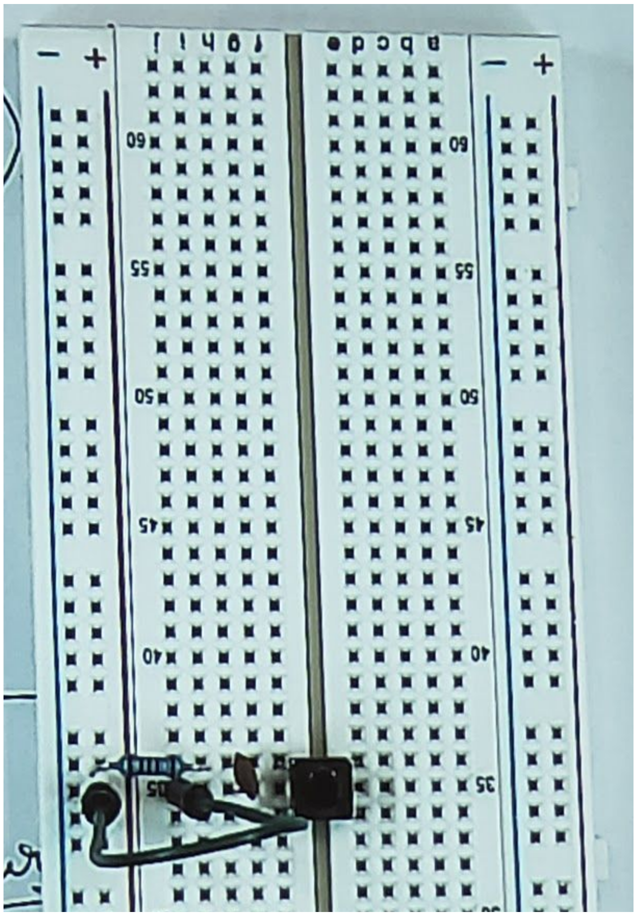
  - If the base is off, no current can flow
  - The current flows from top to bottom
  - Max voltage supply in Arduino is 5V
  - If you want a larger voltage supply, then use a transistor
    - It will supply a low current through the transistor
    - Collector can be anything you want
      - Outer source, not from the Arduino
    - Main goal: Have large currents without frying the arduino
- Capacitor:
  - 
  -

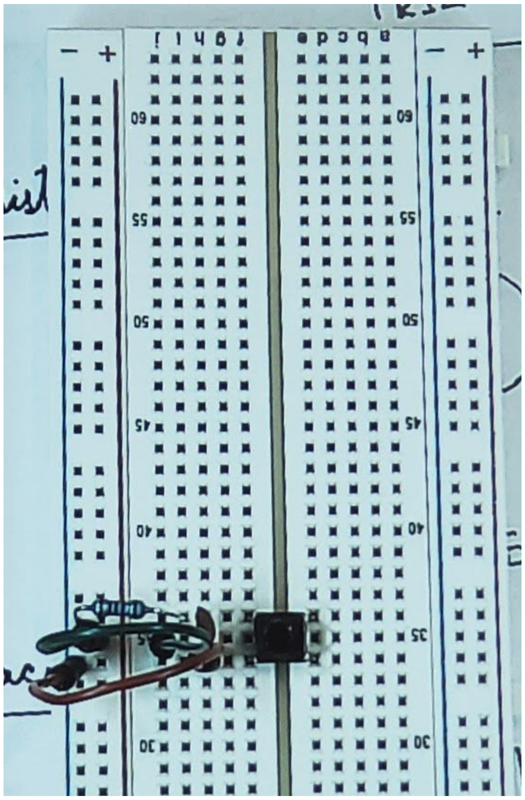
    - 2 Kinds of Capacitors that come with the kit, we will use the smaller one
        - 100nF is the one we will use
    - Slow down fluctuations in your circuit
    - Acts as a temporary power source
    - Purpose is to hold charge
        - Smaller capacitors hold smaller amount of charge
        - Larger capacitors hold larger amount of charge
- Switch
    - 

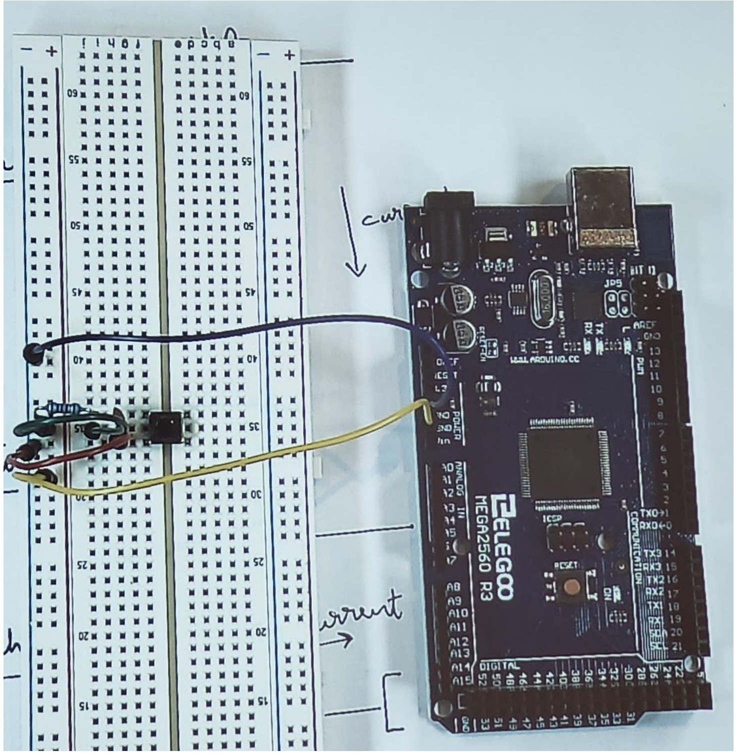
    - Four Legs with a Button on top that will close the circuit when pressed
    - Best way to put the switch on the breadboard is put it on the divider
        - May need to stretch the legs out a bit to fit

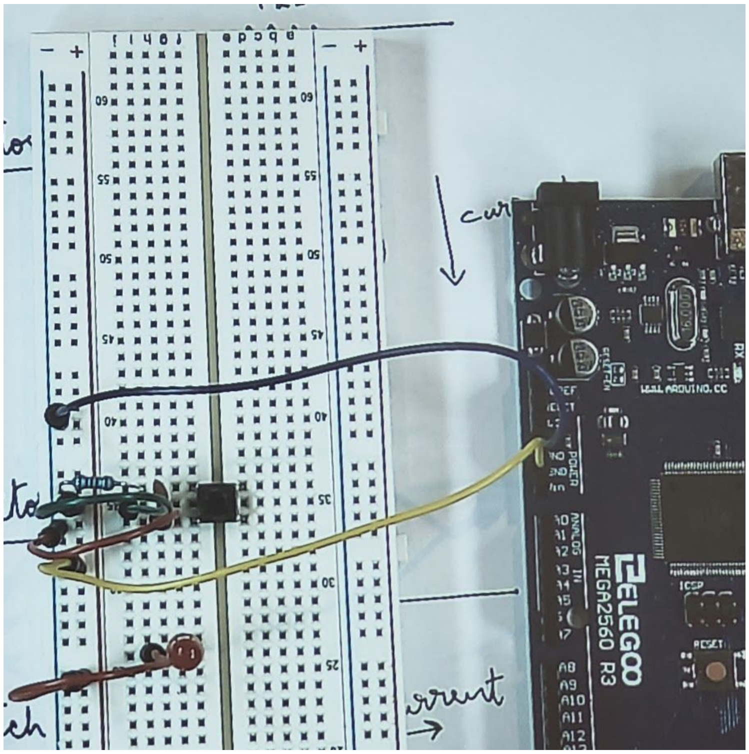
Assembly of the Circuit on the BreadBoard:

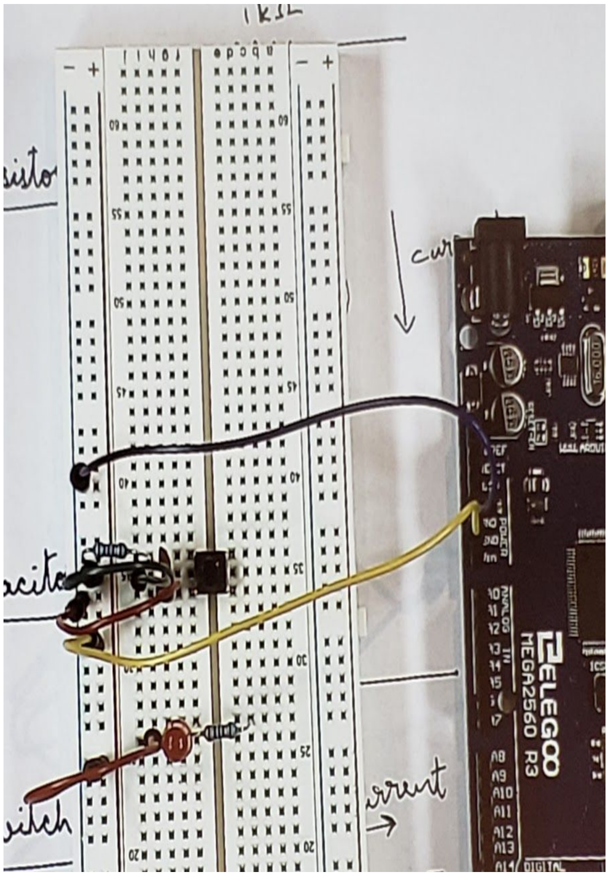
- Most of the time when operating with small elements it is best to use forceps
- Put the capacitor to the left of the switch
    - 

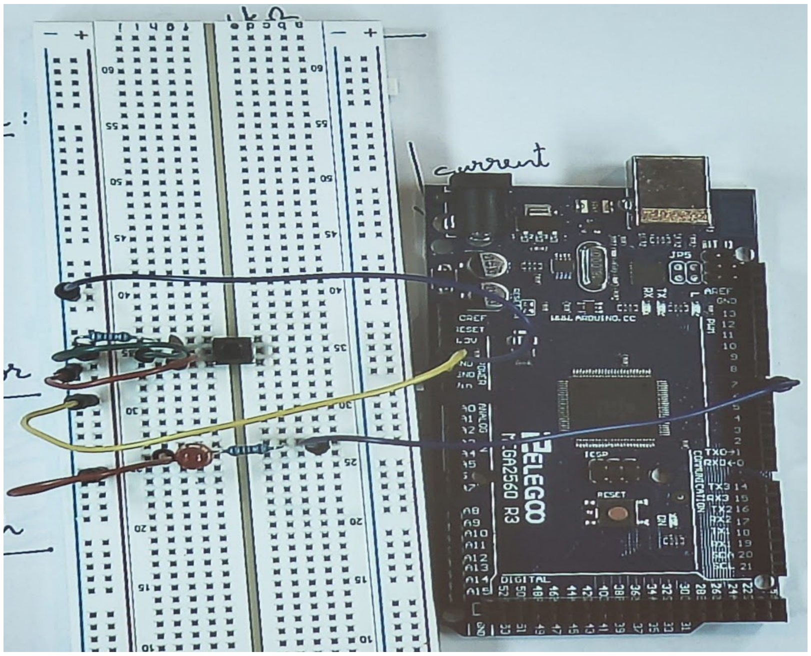
- Connect the ground supply to the same row as the switch(opposite of the board)
    - 
- Put the Resistor in parallel to the capacitor
    - 

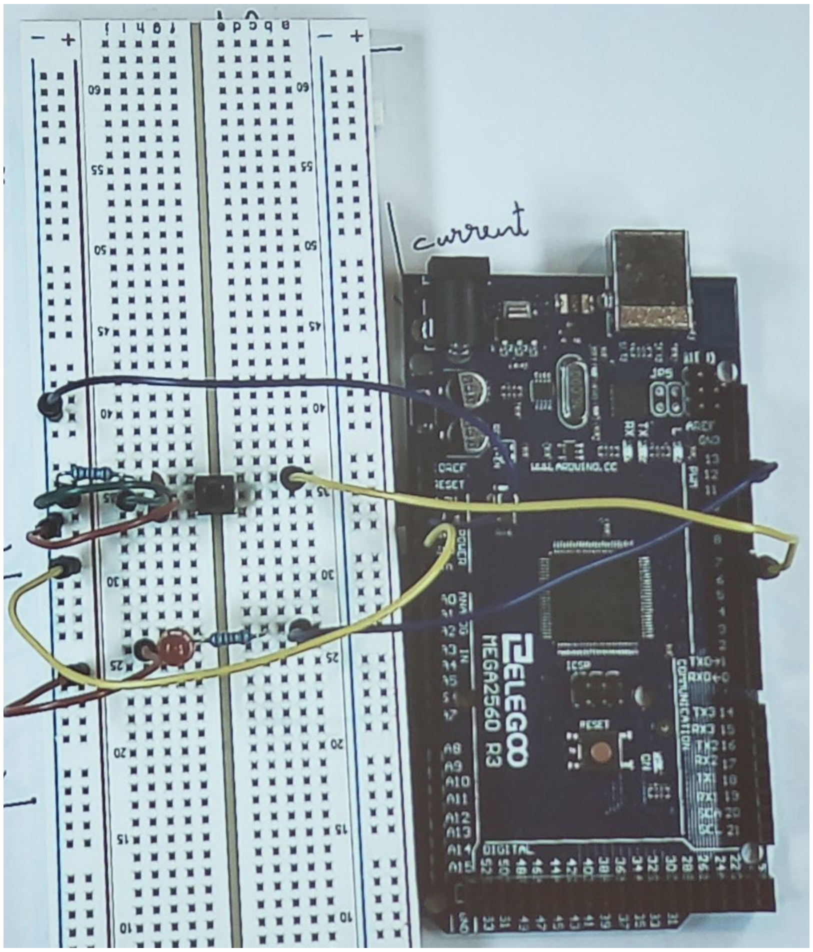
- Connect the other leg of the switch to power supply(opposite of the board)
    - ○ 

- 5V (blue wire) and Ground (yellow wire) from Arduino is connected to BreadBoard (Opposite of what the board says on the top)
    - ○ 

- Connect the LED to Ground
    - 

- Connect the 1k Ohm Resistance in series with the LED
    - 

- Connect LED to pin 12
    - 

- Connect the Switch to Pin 7
    - 

- Code for the Circuit
    - Set LED to OUTPUT
    - Set Button to INPUT
    - In the loop
        - If the button is high (pressed)
            - Set the LED for High, delay for 500 (.5s) then set it back to low
- Behavior of the Code/Circuit
    - Press the button
    - The LED lights up for 0.5sec
    - The LED turns off