# Textures

Professor provided an example showing how to transpose two images using OpenGL:

* Showed how to use arrow keys to interactively transition from one image to another.
* The geometry used is the two triangles from the previous lecture with the textures superimposed.
* Using external opensource library SOIL (Simple OpenGL Image Library)
  + #include <SOIL.h>
  + Can be found using: apt-get SOIL (integrated with ubuntu repository package manager)
* Declare global variable GL\_float mix\_value = 0.2f
  + Very important
* Rest of the setup for shaders is boilerplate from previous examples
* New stuff related to textures
  + Generate two texture maps to bind to geometry
  + texture1 and texture2
  + set texture parameters, wrapping texture around edges of geometry
  + set texture filtering
  + load, create, and generate mipmaps
    - convert image into texture map
    - mipmap hierarchical representation of an image at various resolutions (for zooming in and out) for faster rendering at runtime
  + SOIL\_load\_image
  + glTexImage2d
  + glGenerateMipmap
  + SOIL\_free\_image\_data
  + glBindTexture
* use vertices[] to map the image onto the geometry
  + affects how the images are laid out
  + two floating point values associated with each vertex and then a texture coordinate
* when adding attribute to vertex, need to be mindful of the spacing/offset of the data (in this example 8 floating-point values between vertex values)
* inside display group we have three new things that we should take note of:
  + first call shader program
  + then bind the texture
    - active texture -> bind texture mapping to first texture and second texture
  + pass mix\_value to the shader, argument passed to shader to change its behavior on how it’s going to display the data
* the draw the triangles and nothing new is happening after
* Looking at the shader.vs
  + Data is laid out with input, position, and color coordinate
  + In the texture coordinate we take the x coordinate and 1-y coordinate
* Looking at the shader.frag
  + Input of the shader.frag is the output of the shader.vs
* key\_callback() in main.cpp
  + Enables use of keyboard to interact with the window at runtime

# Arduino

Onto some hardware examples:

LED:

Current

Resistor: 

* 1 kOhm resistor in our kit

Transistor: 

Current

* b – base
* c – connector
* e – emitter
* allows Arduino to handle higher voltages, controls the flow of current

Capacitor: 

* Reduces/ slows down the fluctuations in current
* Holds a charge effectively
* Will use to implement a switch
* We should use the smaller capacitor in kit

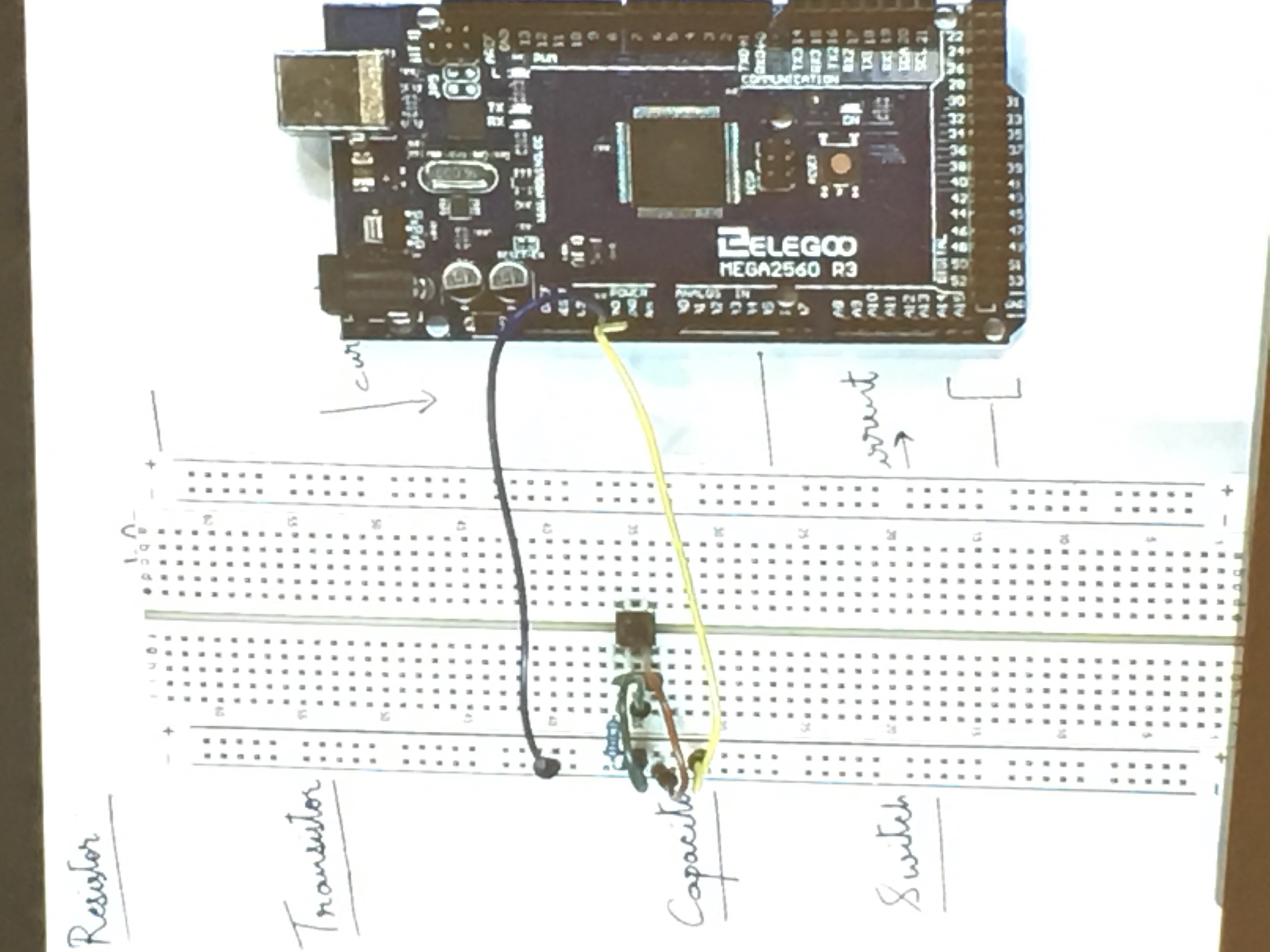
Switch: 

* Use to control current on/off
* Put the switch on the breadboard straddling the divider
  + 



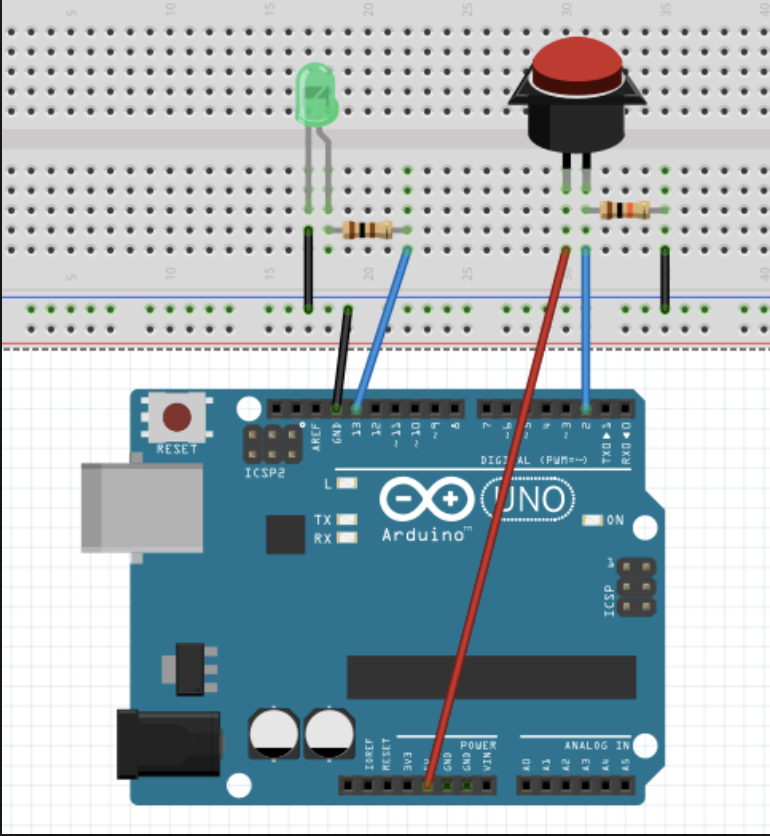
* + May have to stretch the four pins

In class example LED switch circuit:





* The blue wire is 5v source
* Yellow wire is ground





Sketch code:

#define LED 12

#define BUTTON 7

void setup() {

pinMode(LED, OUTPUT);

pinMode(BUTTON, INPUT);

}

void loop() {

if (digitalRead(BUTTON) == HIGH) {

digitalWrite(LED, HIGH);

delay(500);

digitalWrite(LED, LOW);

}

}