Arduino 101
Hands-on: LED Level with Buttons

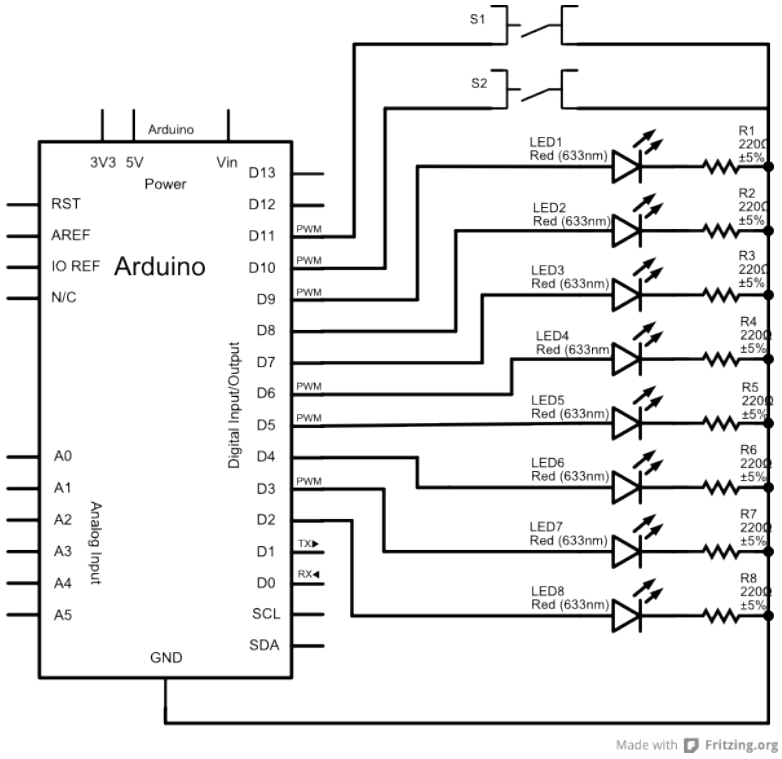
Project Description

This project will demonstrate how to use digital inputs to read button presses to control the setting of the LED Level

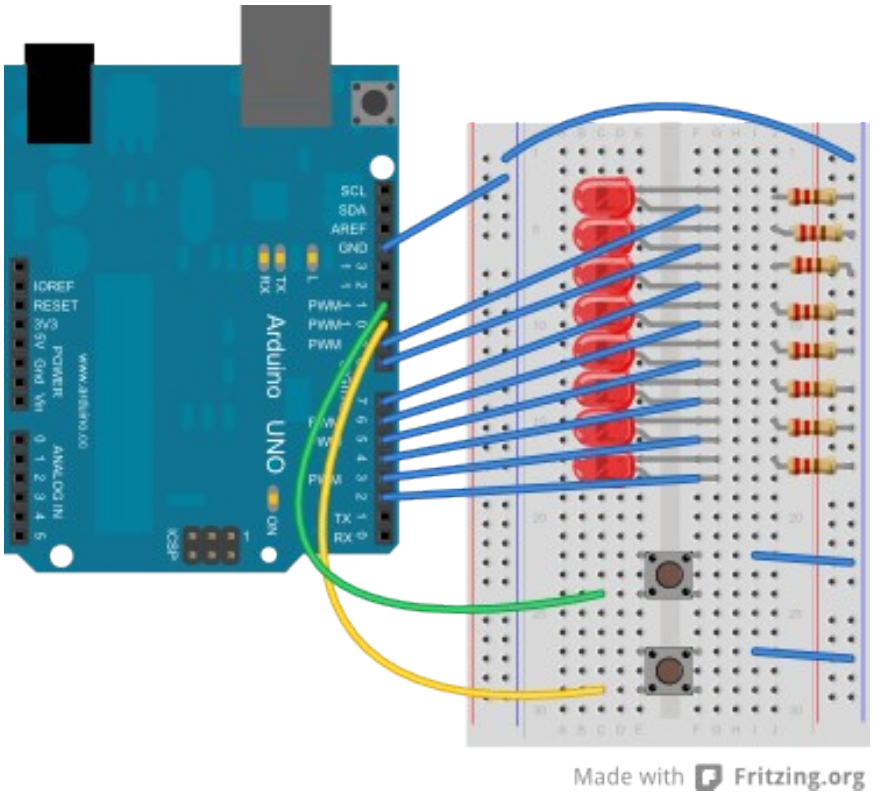
Required Parts

8 red LEDs
8 220Ω Resistors (red, red, black, black)
2 12mm push-button switches

Schematic



Circuit



NOTES: The LED circuit is the same as the previous project, but we're also connecting some push-button switches to digital pins 10 and 11. On these switches, the top 2 pins and bottom 2 pins in the diagram are connected; make sure you connect the input and ground wires on opposite corner terminals of the switch, otherwise the connection will always be closed!

Code

```
// LED Level with Buttons Sketch for Arduino 101
// by Nick Borko

// function to set the LEDs to a specific level
void setLed(int level) {
  // iterate through the pins
  for (int pin = 2; pin <= 9; pin += 1) {
    // compare the pin to the level...
    if (pin < level + 2) {
      // LED is on
      digitalWrite(pin, HIGH);
    } else {
      // LED is off
      digitalWrite(pin, LOW);
    }
  }
}

void setup() {
  int pin;
  // initialize pins 2-9 to be output pins
  for (pin = 2; pin <= 9; pin += 1) {
    pinMode(pin, OUTPUT);
  }
  // initialize pin 10 for input
  pinMode(10, INPUT);
  // enable internal pull-up resistor
  digitalWrite(10, HIGH);
  // initialize pin 11 for input
  pinMode(11, INPUT);
  // enable internal pull-up resistor
  digitalWrite(11, HIGH);
}

// use a global variable to track the current level
int level = 0;

void loop() {
  // if pin 10 reads low...
  if (digitalRead(10) == LOW) {
    // decrease the level, but no lower than 0
    level = max(0, level - 1);
    // set the LED level
    setLed(level);
    // debounce the switch by waiting
    delay(250);
  }
  // if pin 11 reads low...
  if (digitalRead(11) == LOW) {
    // increase the level, but no higher than 8
    level = min(8, level + 1);
    // set the LED level
    setLed(level);
    // debounce the switch by waiting
    delay(250);
  }
}
```

Discussion

For those familiar with electronics, you will recall that you normally need a **pull-up** or **pull-down** resistor on a switch to keep an IC pin from “floating” and giving unpredictable readings. Why don't we use any in our circuit? Because the AVR has internal 20kΩ pull-up resistors that can be enabled by setting a digital pin to **HIGH**, even though it is configured as an input. Handy!

(If you don't understand what that means, don't worry about it! You've got plenty of time to learn all there is about designing electronic circuits, and you're just starting!)

If you recall from a previous project, we talked about **local** vs. **global** variables. Since the variable **level** is declared outside any function definition, it can be referenced by any function, and is said to be **global**. We are using a global variable to track the level, because variables inside the **loop()** function only exist for **one** execution of the function, and we need to remember this value for *every* execution of **loop()**. Since **level** is declared outside of the **loop()** function, it will always retain its value.

In **loop()**, we check to see if a button is being pressed by reading the connected pin and checking if it is in a **LOW** state, due to the pin being connected to ground by the switch. Depending on which button was pressed, we either increase or decrease the level. We also wait little to keep the level from shooting up or down too fast for a human to control.