Creating a fun game with Arduino!

Basics of Arduino Circuits *Eric, Andy, Allen*

Introduction

—

- We are going over everything from the basics
- Learning *why* is just as important as *what*

For Students

—

- Follow along with the workshop
- Ask questions whenever you want
- Try to be creative (ignore rule one sometimes)

The area of a circle with radius r is:

$$A = 2\int_{-r}^{r} \sqrt{r^2 - x^2}\,dx$$

$$\text{let } x = r\sin\theta \implies dx = r\cos\theta d\theta$$

$$= 2\int_{x=-r}^{x=r} \sqrt{r^2 - (r\sin\theta)^2}\cdot r\cos\theta d\theta$$

$$= 2r^2\int_{-r}^{r} \sqrt{1-\sin\theta^2}\cdot\cos\theta d\theta$$

$$= 2r^2\int_{-r}^{r} \cos^2\theta d\theta$$

$$= 2r^2\int_{-r}^{r} \frac{\cos 2\theta + 1}{2} d\theta$$

$$= r^2\int_{-r}^{r} (\cos 2\theta + 1) d\theta$$

$$= r^2\left[\frac{1}{2}\sin 2\theta + \theta\right]_{-r}^{r}$$

$$= r^2[\sin\theta\cos\theta + \theta]_{-r}^{r}$$

$$= r^2\left[\frac{x}{r}\cdot\frac{\sqrt{r^2-x^2}}{r} + \arcsin\left(\frac{x}{r}\right)\right]_{-r}^{r}$$

$$= r^2\left(\frac{r}{r}\cdot\frac{\sqrt{r^2-r^2}}{r} + \arcsin\left(\frac{r}{r}\right)\right) - \left(\frac{-r}{r}\cdot\frac{\sqrt{r^2-(-r)^2}}{r} + \arcsin\left(\frac{-r}{r}\right)\right)$$

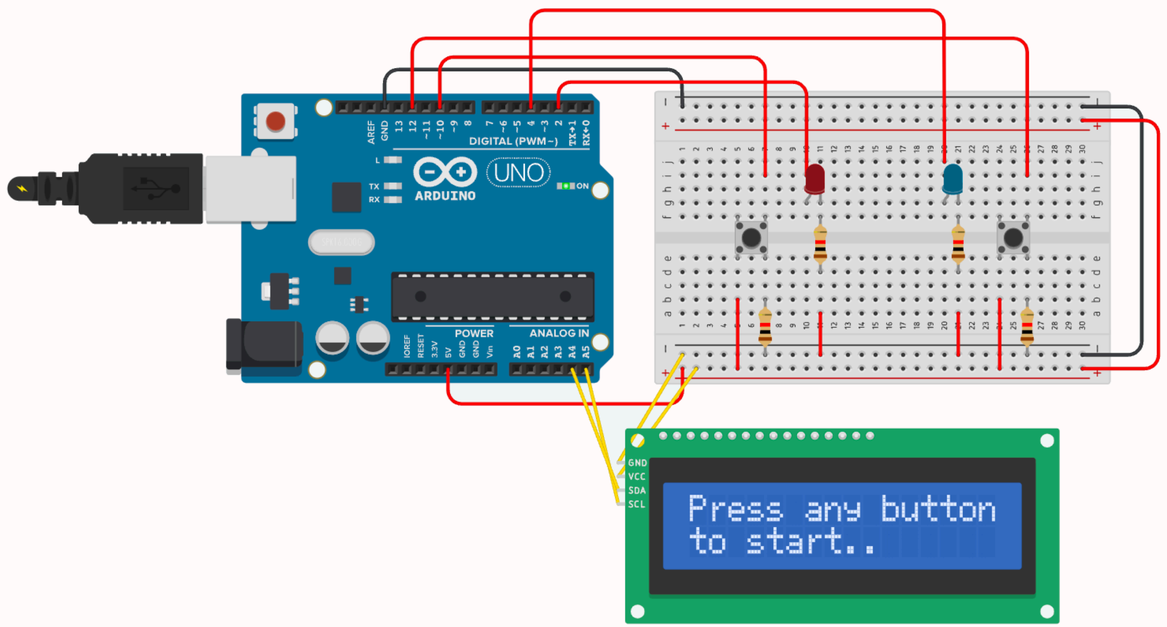
$$= r^2\left(\left(\frac{\pi}{2}\right) - \left(-\frac{\pi}{2}\right)\right)$$

$$= r^2\pi$$

$$\boxed{A = \pi r^2}$$

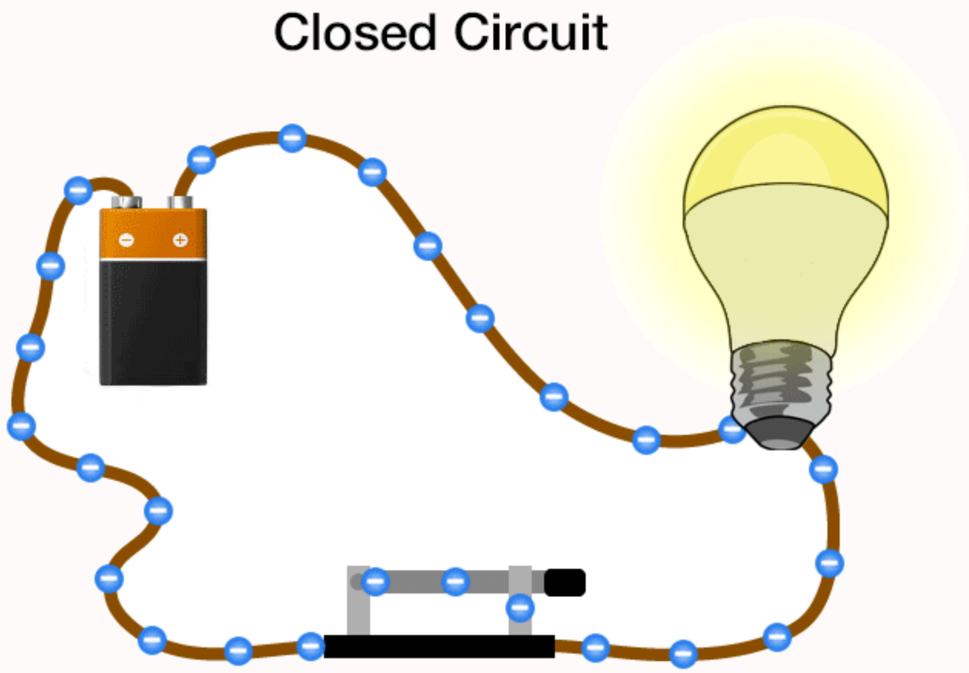
Outline of Workshop

- Electricity
- Circuits
- Integration of Arduino
- Arduino Programming
- LCD I2C Protocol
- Programming the Project

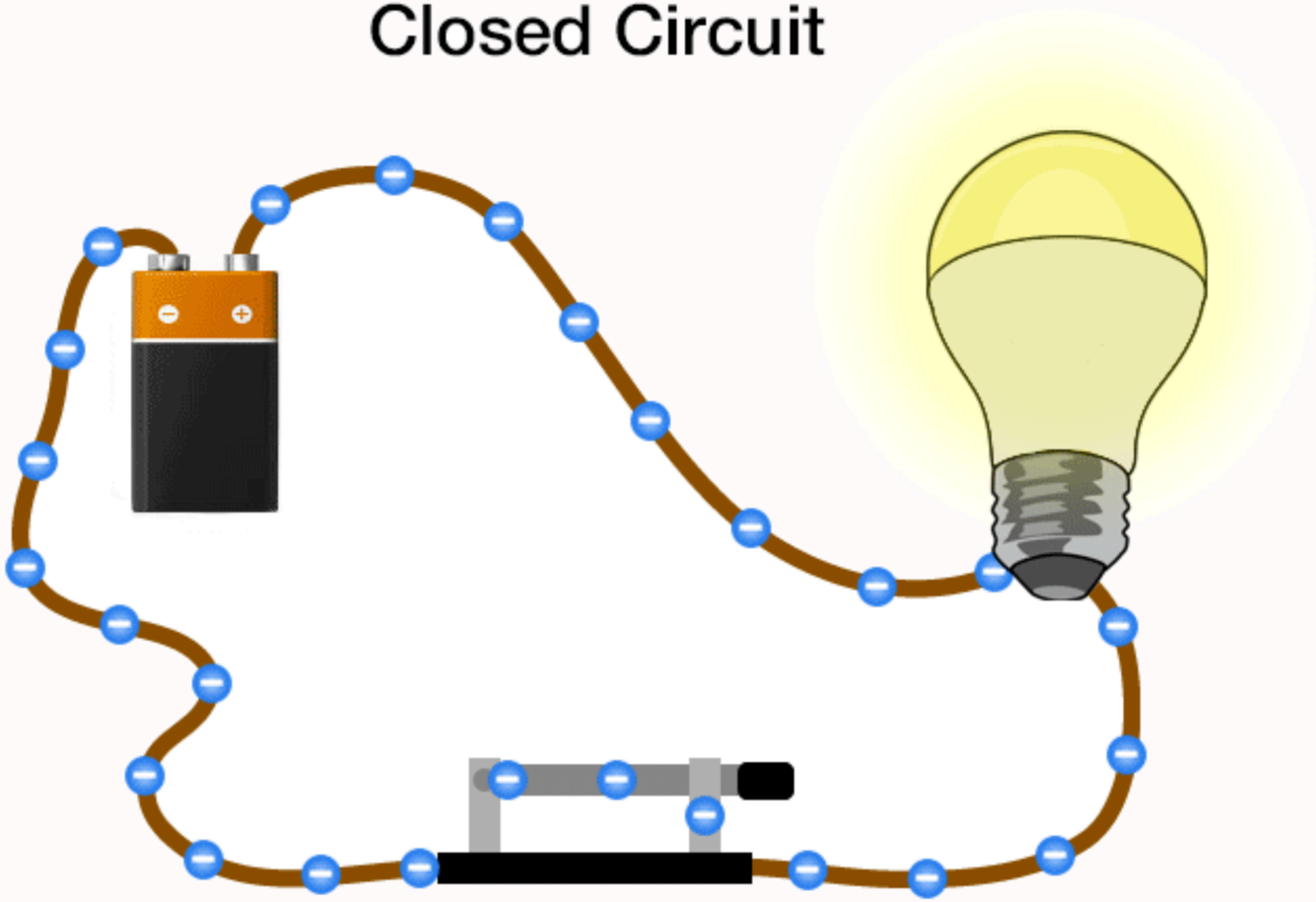


Electricity

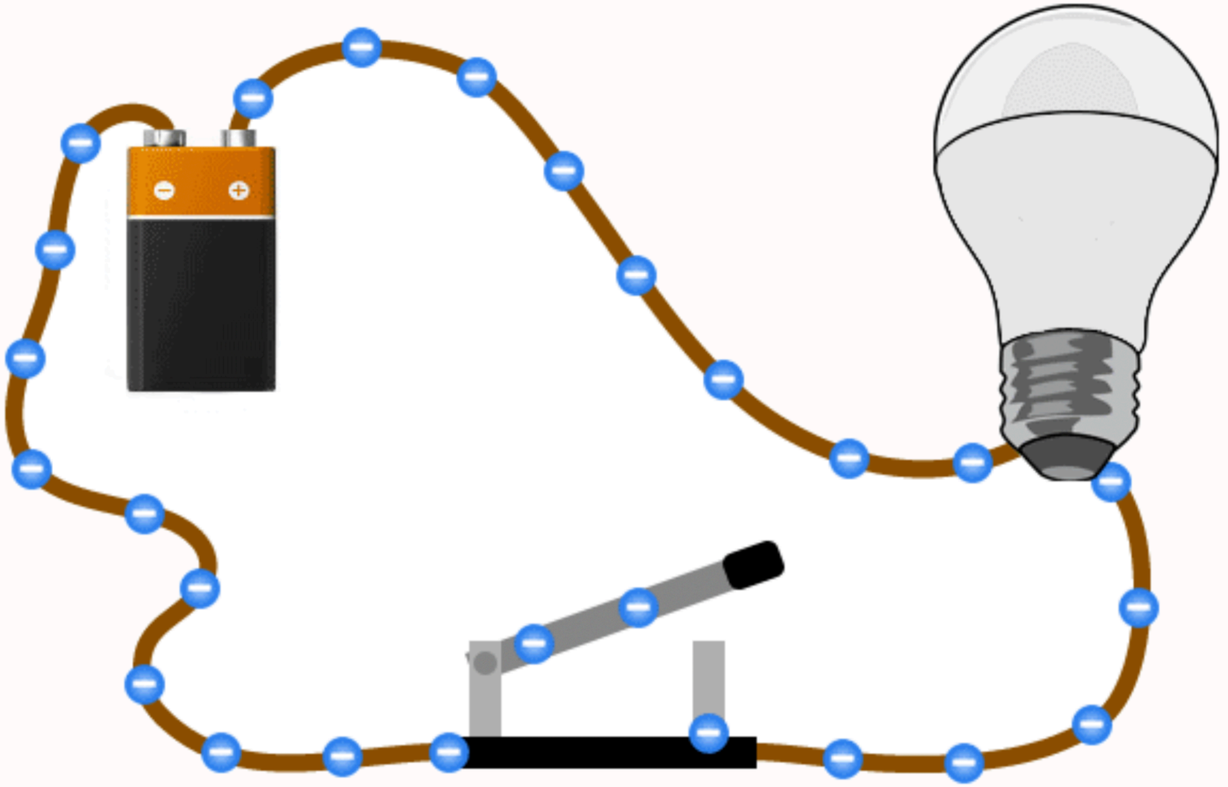
- Electrons flowing from negative to positive
- Powers components in its path (LEDs, Arduino, Light Bulb)
- Electricity can only flow if there is a path

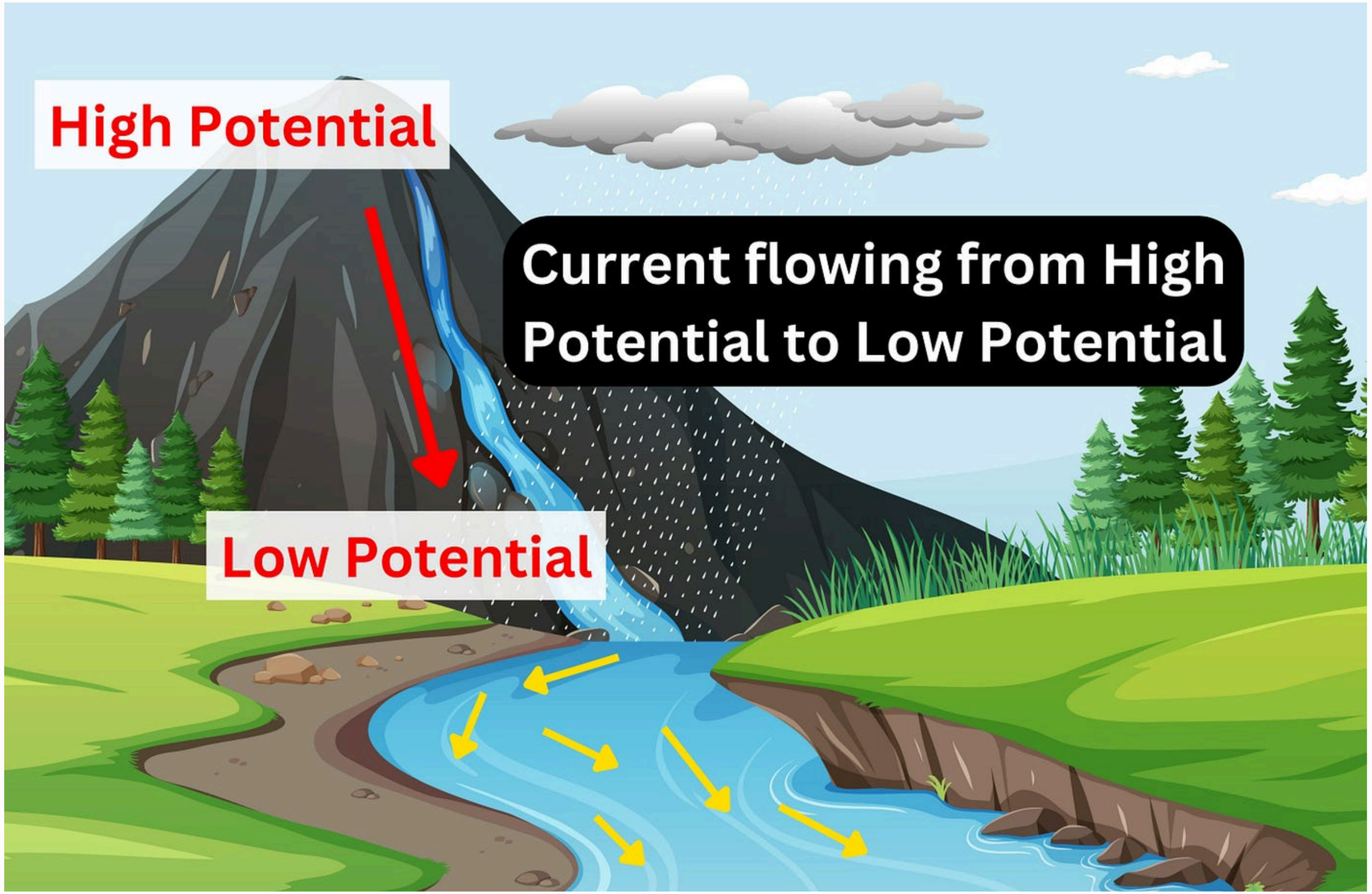


Closed Circuit



Open Circuit

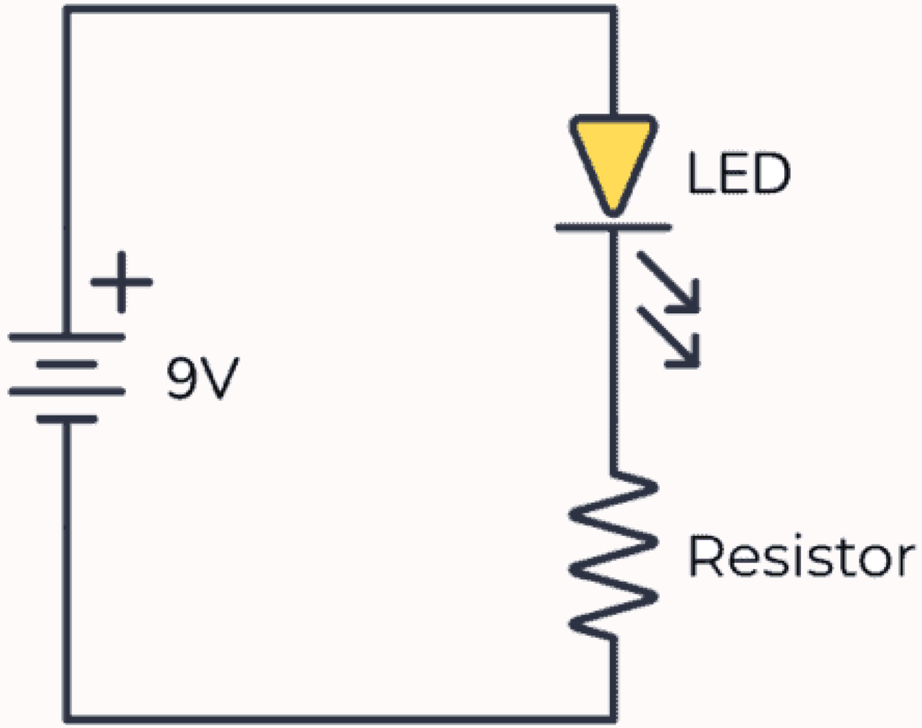


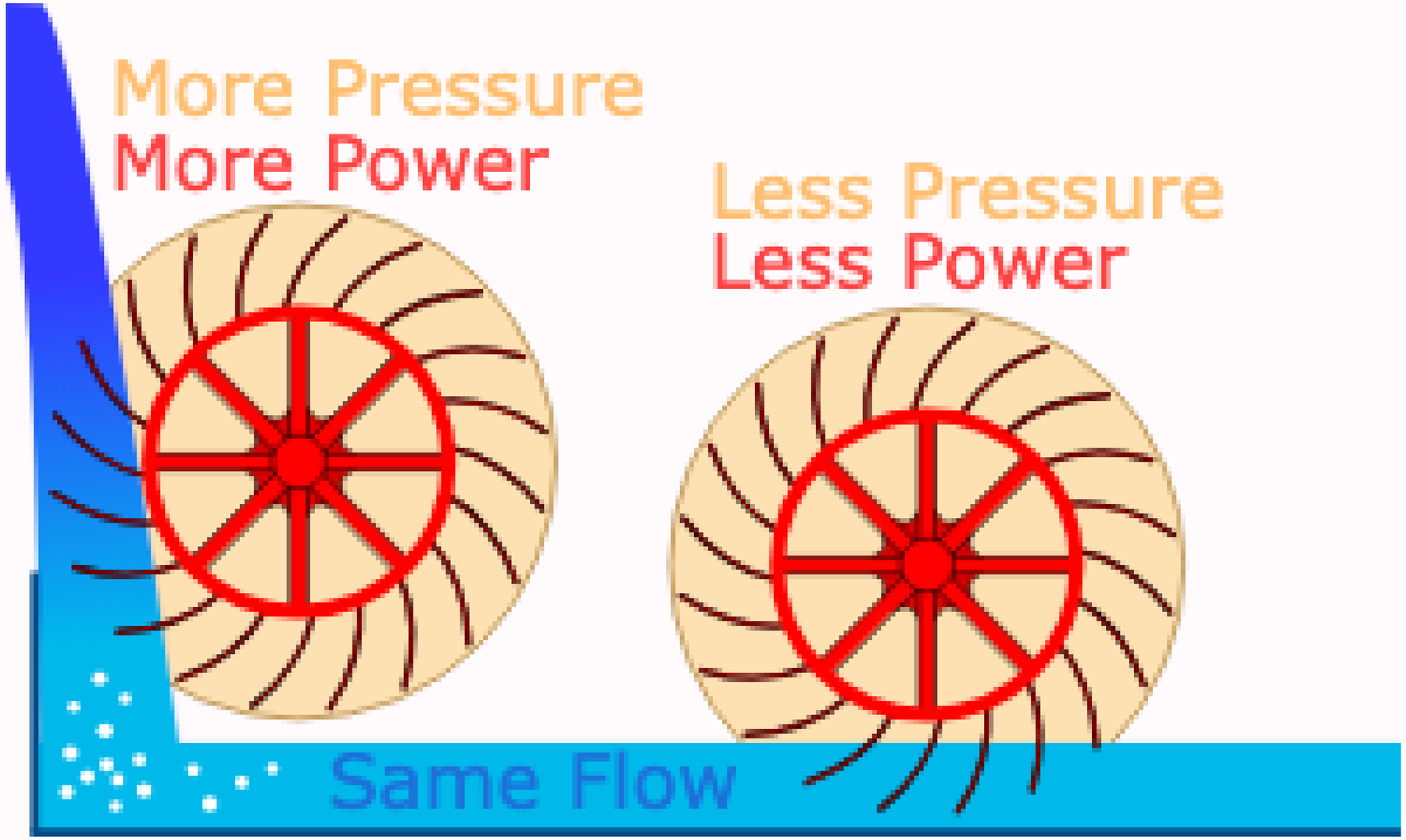

High Potential
Current flowing from High Potential to Low Potential
Low Potential

Electricity Terms

- **Current** = How much electricity flows per second
- **Voltage** = Potential electricity
- **Resistance** = How much electricity a material stops

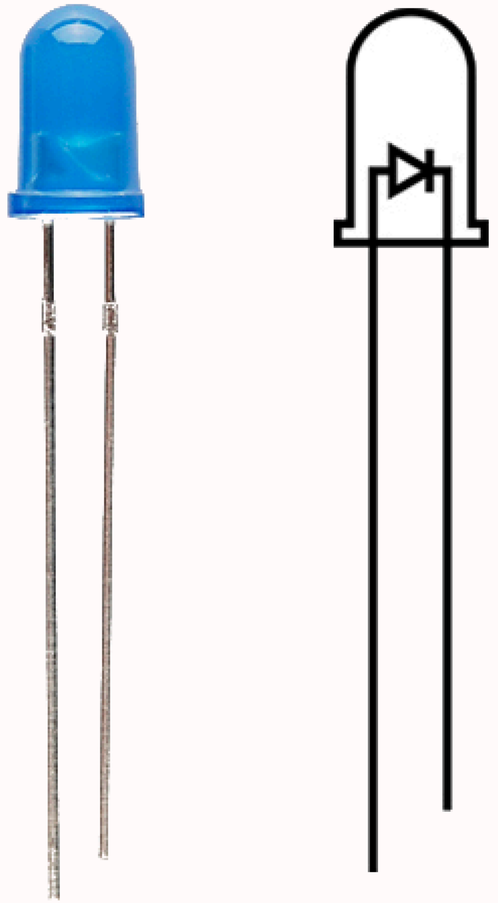
- Ohm's Law: $V = IR \implies I = \frac{V}{R}$
- Too much current = some components overheat and break
  - LEDs, Arduino, etc.




More Pressure
More Power
Less Pressure
Less Power
Same Flow

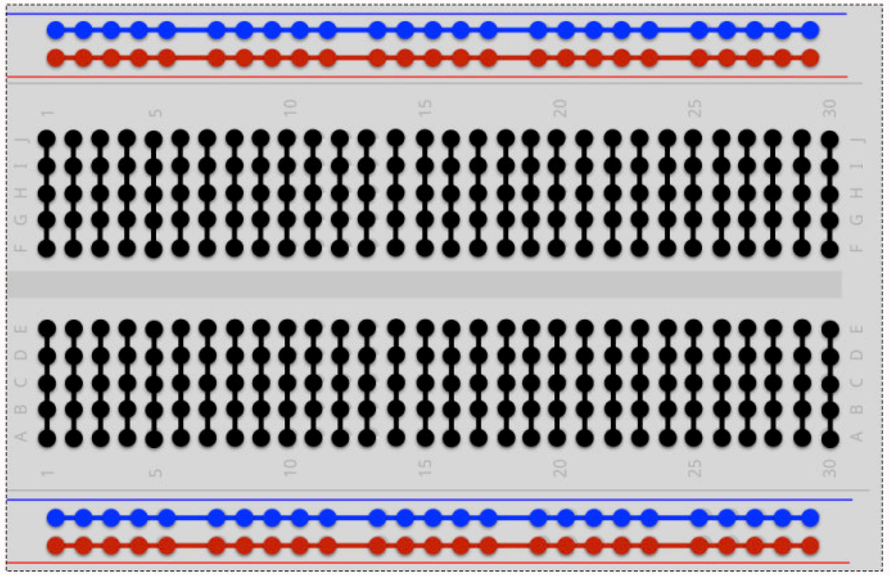
LEDs

- **Light-emitting diode**
- Diode lets current go through *one direction*
- LED = *“thing that lights up when current goes through in a certain direction”*
- Longer end to high voltage (**5V**), shorter end to low voltage (**Ground**)



Circuits and Wiring

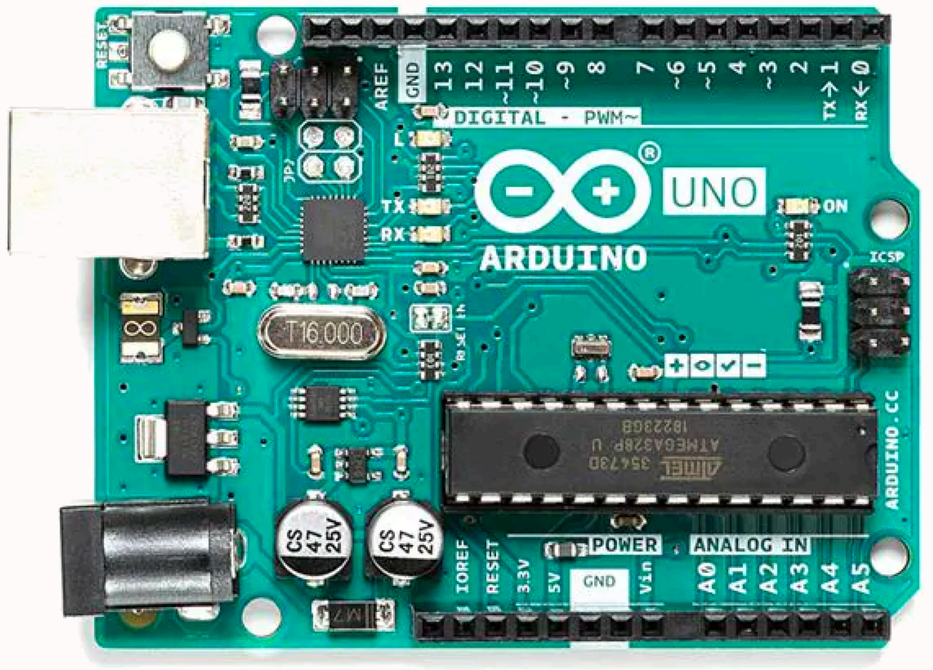
- Circuit on **breadboard**, with wires and other components
  - LEDs, Buttons, LCDs, Arduino, resistor, etc.
- Breadboards make wiring *easy to change*
  - Connects wires just by plugging in
  - Blue is connected, red is connected, black is connected
  - Great for learning & projects

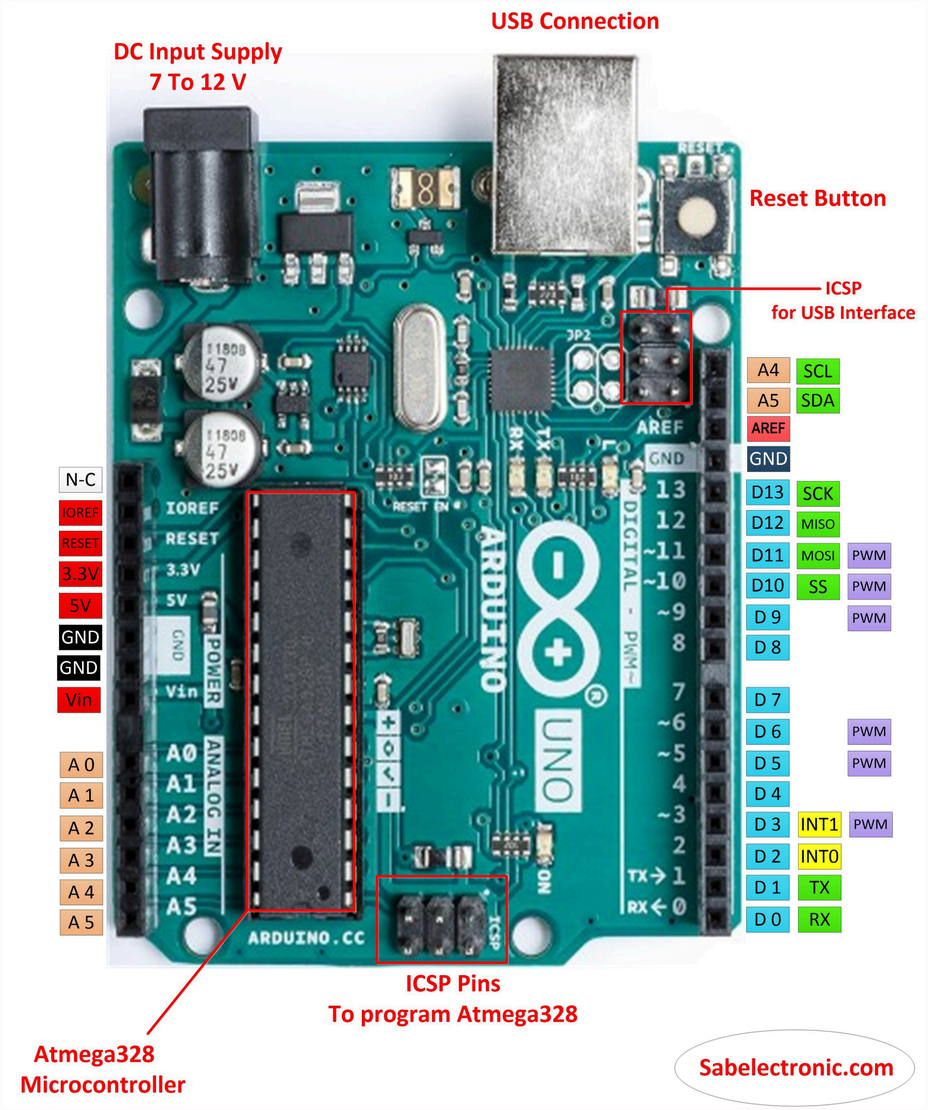


Arduino

—

- Has pins for *output* and *input*
- Has pins for **ground** and **V5** / **V3.3** (constant)
- Connects to power source (could be computer)
- Computer connects to upload code
- Arduino runs code that is uploaded, with any power source




USB Connection
DC Input Supply
7 To 12 V
Reset Button
ICSP
for USB Interface
N-C
IOREF
RESET
3.3V
5V
GND
GND
Vin
A 0
A 1
A 2
A 3
A 4
A 5
A4 SCL
A5 SDA
AREF
GND
D13 SCK
D12 MISO
D11 MOSI PWM
D10 SS PWM
D 9 PWM
D 8
D 7
D 6 PWM
D 5 PWM
D 4
D 3 INT1 PWM
D 2 INT0
D 1 TX
D 0 RX
ICSP Pins
To program Atmega328
Atmega328
Microcontroller
Sabelectronic.com

Arduino Code

—

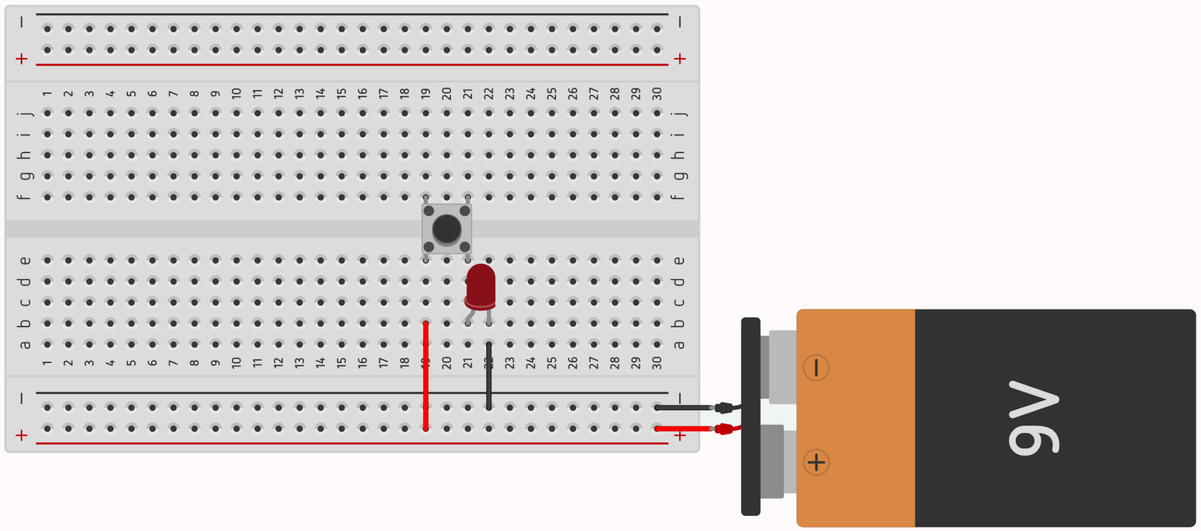
- Uses Arduino language (C++ with special built-in functions)
  - *digitalWrite*(…), *delay*(…), *analogRead*(…), etc.
- Runs **setup**, then runs **loop** function until it is off

Code that turns LED on and off

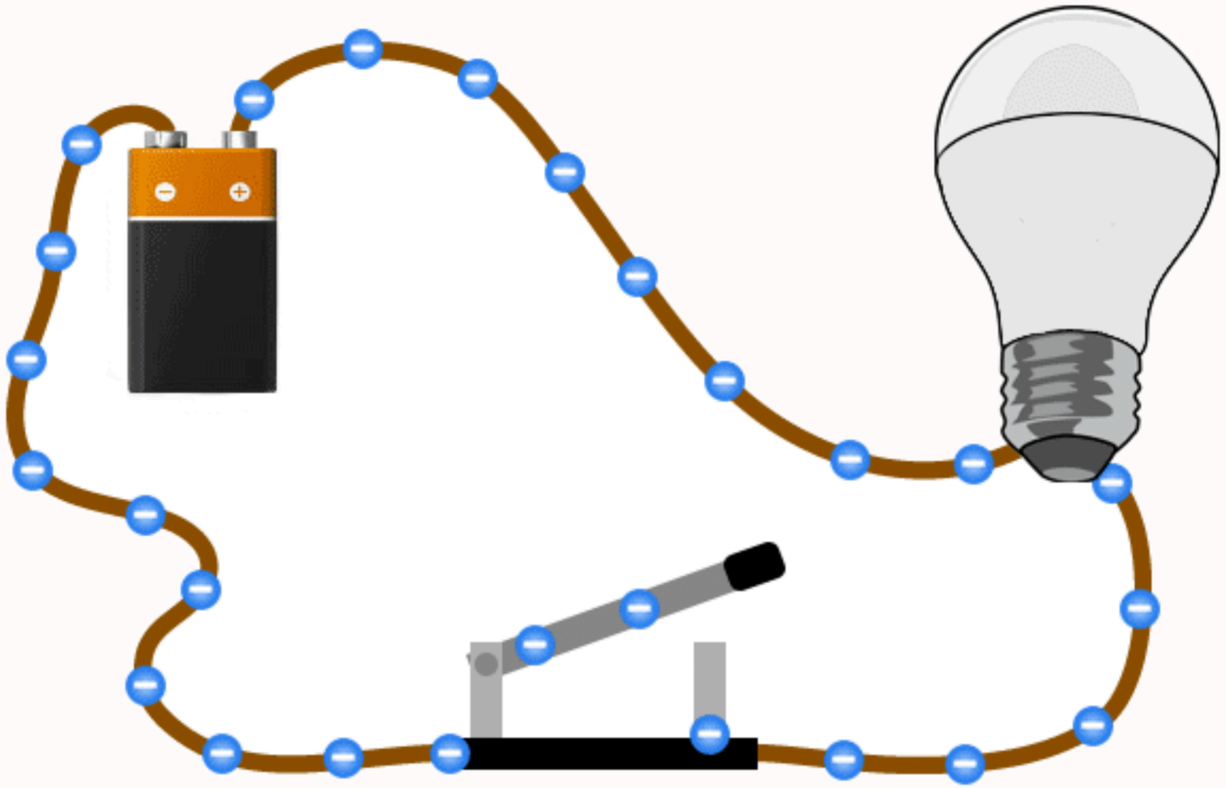
```
// The setup() function runs once arduino gets power
void setup() {
  // initialize digital pin 13 for output
  pinMode(13, OUTPUT);
}

// the loop() function runs over and over again
forever
void loop() {
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(13, HIGH);

  // wait for a second (1000 milliseconds)
  delay(1000);

  // turn the LED off by making the voltage LOW
  digitalWrite(13, LOW);

  // wait for a second (1000 milliseconds)
  delay(1000);
}
```

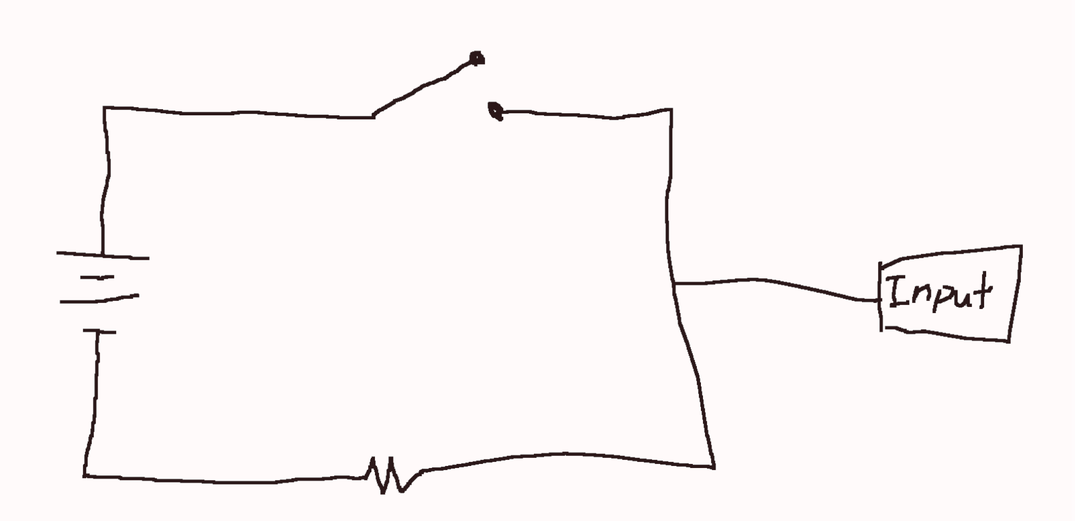
Buttons

- Buttons control when circuit is open or closed
- Pushing button connects circuit
- *“Pull-Down Resistors”* are necessary when using buttons for input
  - Get rid of excess charge (sends it to **ground**)

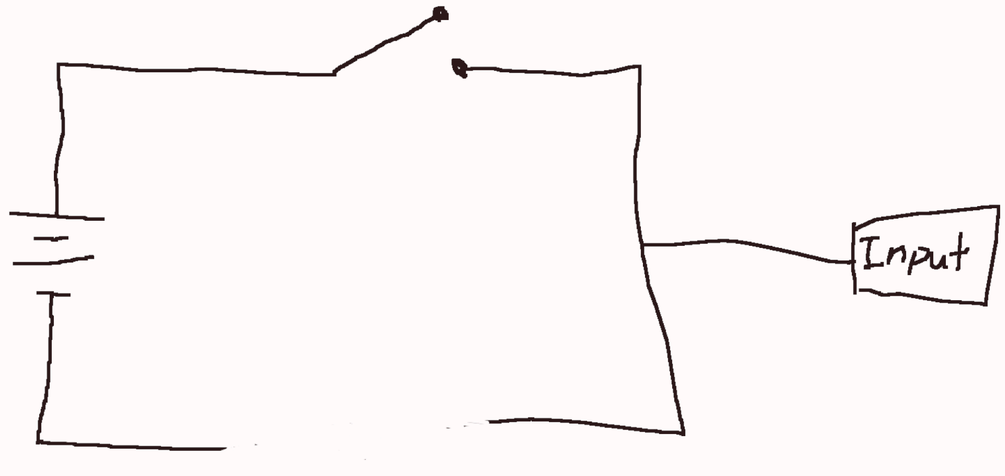


Open Circuit





(with pulldown)



(without pulldown)

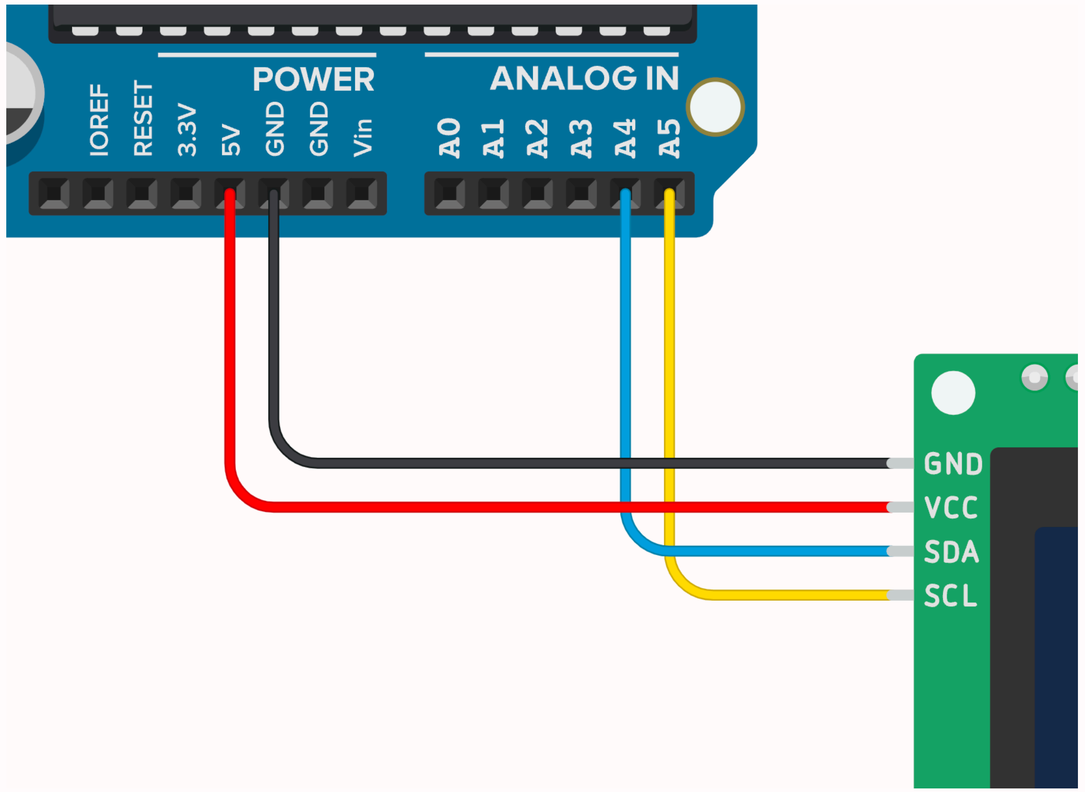
(the slideshow budget ran out)

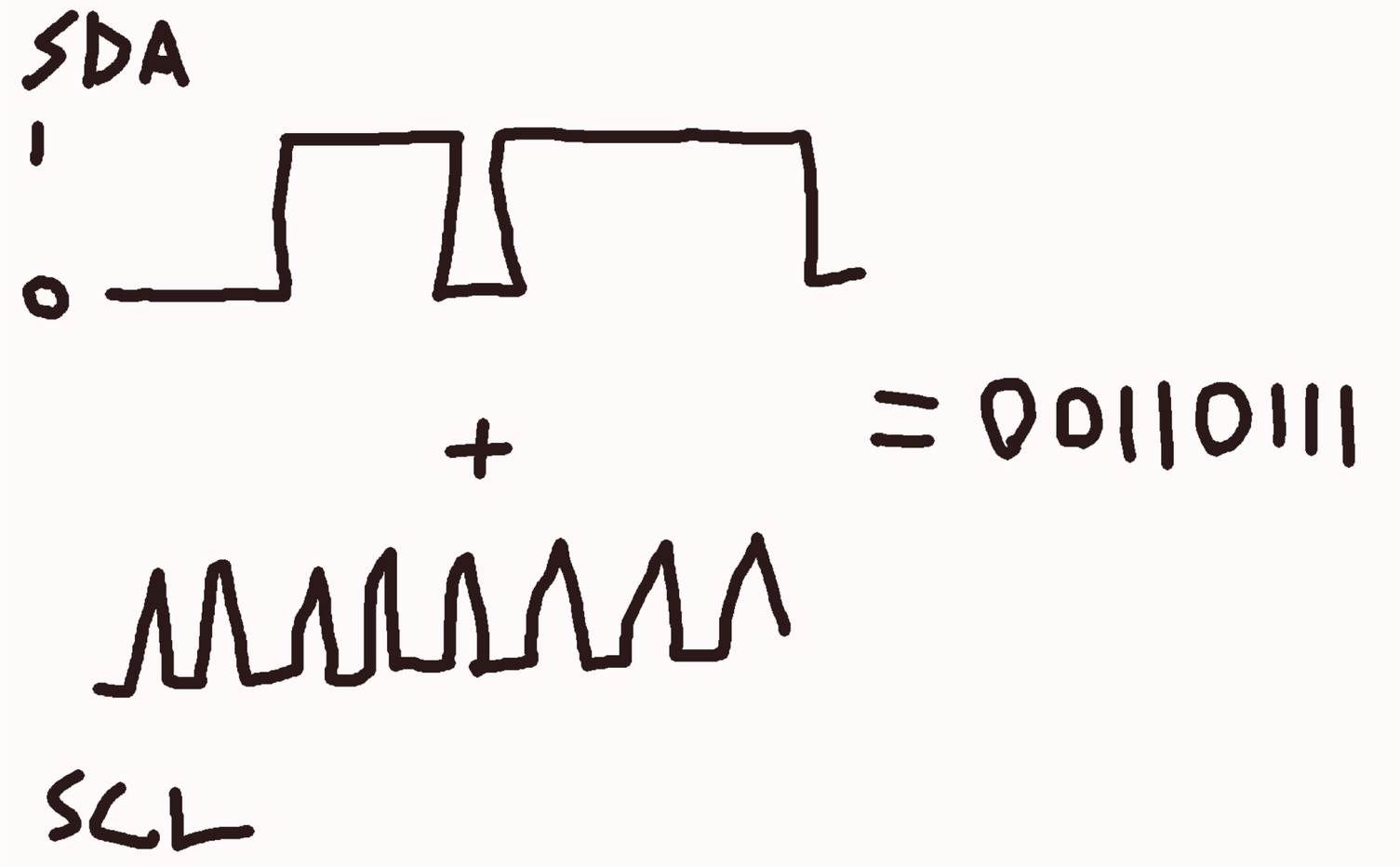
Code with Button Logic

```
const int BUTTON_PIN = 10;
const int LED_PIN = 6;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT); // set pin to input
}

void loop() {
   // runs if there is high input to the pin
  if (digitalRead(BUTTON_PIN) == HIGH) {
     // flash light once
    digitalWrite(LED_PIN, HIGH);
    delay(100);
    digitalWrite(LED_PIN, LOW);
    delay(100);
  }
}
```

LCD I2C Protocol

- Uses two signals
  - SDA used to transmit data
  - SCL used for data timing
- Two other pins are…
  - VCC, for positive voltage
  - GND, for zero voltage
- With LCD, the data transmitted is the text to display





(slideshow budget ran out again)

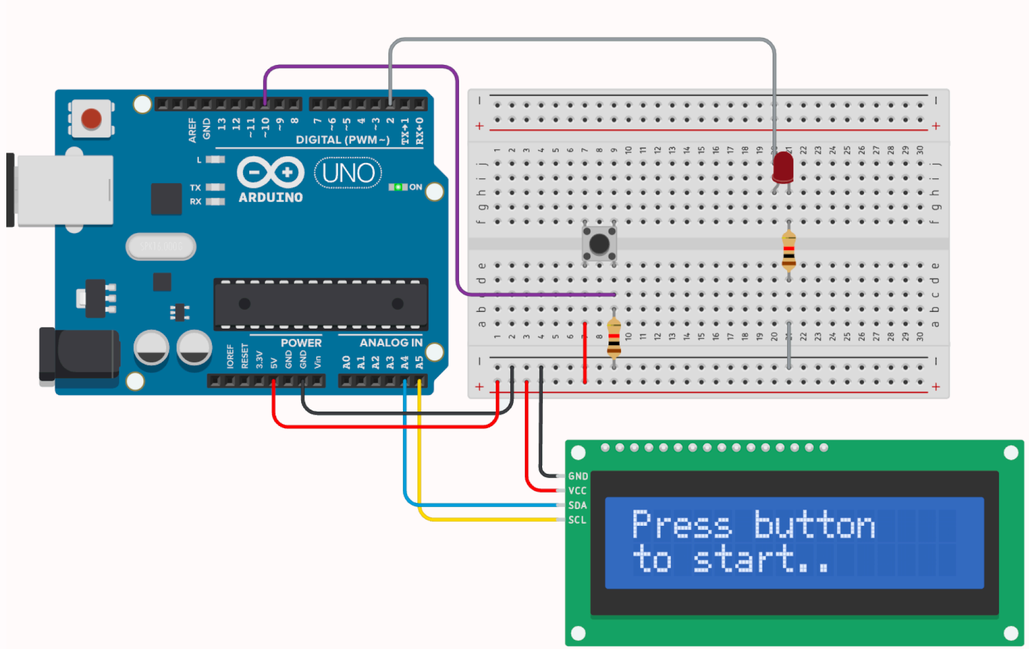
LCD I2C Code

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // port, rows, columns

void setup() {
  lcd.init(); // initializes the lcd
  lcd.backlight(); // turns the backlight on
  lcd.print("Starting the LCD"); // prints to the lcd
}

void loop() {
  lcd.setCursor(0, 1); // set the cursor to the second row
  lcd.print("                "); // clear line
  lcd.setCursor(0, 1); // set the cursor to the second row again
  lcd.print(millis()); // print time in milliseconds
  lcd.print("ms");
  delay(200);
}
```

Reaction Time Game

_

- Wait for an LED to turn on
- Player presses button ASAP
- Displays their reaction time
- That’s it!



```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const int BUTTON_PIN = 10;
const int LED_PIN = 2;

long ledStartTime; // stores start time of LED turning on

void setup() {
  // set up pins to proper input/output
  pinMode(BUTTON_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);

  // set up lcd, and print welcome message
  lcd.init();
  lcd.backlight();
  lcd.print("CLICK TO START!");

  // initialize / reset is the same
  reset();
}
```

```
void loop() {
  if (digitalRead(BUTTON_PIN) == HIGH) { // check if clicked early
    lcd.clear();
    lcd.print("TOO EARLY");
    reset();
  }
  if (millis() >= ledStartTime) { // check if time to turn on led
    // turn on led and display message
    digitalWrite(LED_PIN, HIGH);
    lcd.clear();
    lcd.print("CLICK!");

    // wait until user clicks and calculate time it took
    waitForClick();
    int resultTime = millis() - ledStartTime;

    // clear lcd and print resulting time
    lcd.clear();
    lcd.print(resultTime);
    lcd.print("ms");
    reset();
  }
}
```

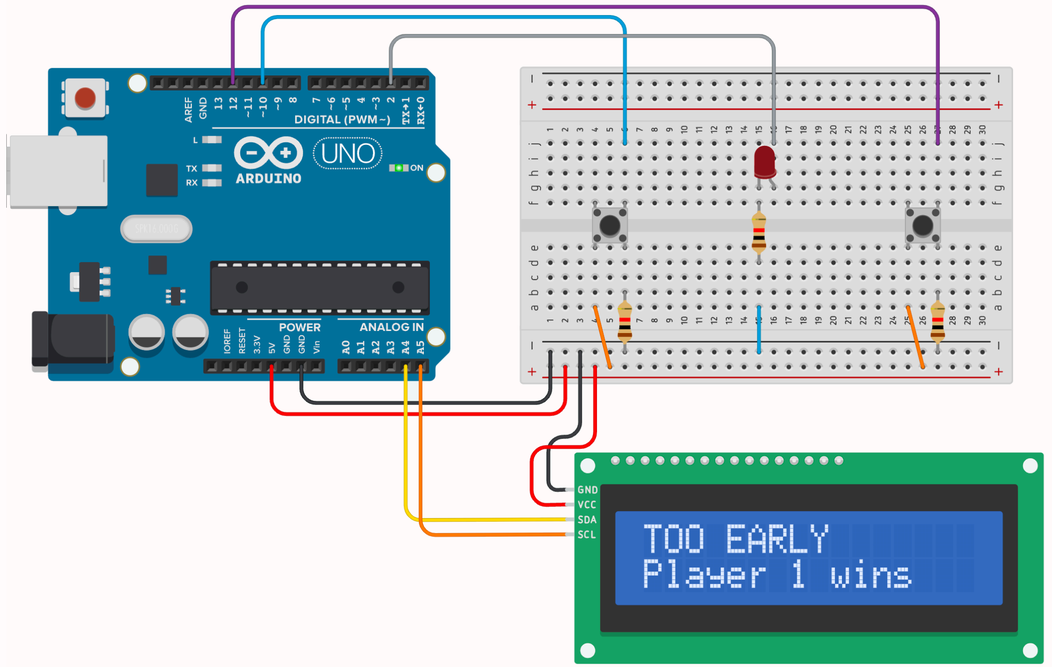
```
void reset() {
  waitForClick(); // click to reset

  // turn off led
  digitalWrite(LED_PIN, LOW);

  // clear lcd and print wait message
  lcd.clear();
  lcd.print("WAIT FOR LIGHT!");

  // set led to turn on in 1-3 seconds
  ledStartTime = millis() + random(1000, 3000);
}
```

```
void waitForClick() {
  // wait for button release
  while (true) {
    if (digitalRead(BUTTON_PIN) == LOW) {
      break;
    }
  }
  // wait for button down
  while (true) {
    if (digitalRead(BUTTON_PIN) == HIGH) {
      break;
    }
  }
  // wait for button release
  while (true) {
    if (digitalRead(BUTTON_PIN) == LOW) {
      break;
    }
  }
}
```

Reaction Time Game 1v1

_

- Wait for an LED to turn on
- Two players press button as fast as possible
- Displays their reaction time, and who won
- A bit more complicated…?



```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const int BUTTON_PIN_1 = 10;
const int BUTTON_PIN_2 = 12;
const int LED_PIN = 2;

long ledStartTime; // stores start time of LED turning on

void setup() {
  // set up pins to proper input/output
  pinMode(BUTTON_PIN_1, INPUT);
  pinMode(BUTTON_PIN_2, INPUT);
  pinMode(LED_PIN, OUTPUT);

  // set up lcd, and print welcome message
  lcd.init();
  lcd.backlight();
  lcd.print("CLICK TO START!");

  // initialize / reset is the same
  reset();
}
```

```
void loop() {
  checkEarlyClick();
  if (millis() >= ledStartTime) { // check if time to start
    lcd.clear();
    lcd.print("CLICK!");
    digitalWrite(LED_PIN, HIGH);

    int winningPlayer = waitForClick();
    int resultTime = millis() - ledStartTime;

    lcd.clear();
    lcd.print(resultTime);
    lcd.print("ms");
    lcd.setCursor(0, 1);
    lcd.print("Player ");
    lcd.print(winningPlayer);
    lcd.print(" wins");

    reset();
  }
}
```

```
void reset() {
  waitForClick(); // click to reset

  digitalWrite(LED_PIN, LOW);
  lcd.clear();
  lcd.print("WAIT FOR LIGHT!");

  ledStartTime = millis() + random(1000, 3000);
}
```

```
void checkEarlyClick() {
  if (digitalRead(BUTTON_PIN_1) == HIGH) { // check if clicked
early
    lcd.clear();
    lcd.print("TOO EARLY");
    lcd.setCursor(0, 1);
    lcd.print("Player 2 wins");
    reset();
  }
  if (digitalRead(BUTTON_PIN_2) == HIGH) { // check if clicked
early
    lcd.clear();
    lcd.print("TOO EARLY");
    lcd.setCursor(0, 1);
    lcd.print("Player 1 wins");
    reset();
  }
}
```

```
int waitForClick() {
  int p1 = 0;
  int p2 = 0;
  while (true) {
    if (p1 == 0 && digitalRead(BUTTON_PIN_1) == LOW) {
      p1++;
    }
    if (p1 == 1 && digitalRead(BUTTON_PIN_1) == HIGH) {
      p1++;
    }
    if (p1 == 2 && digitalRead(BUTTON_PIN_1) == LOW) {
      return 1;
    }

    if (p2 == 0 && digitalRead(BUTTON_PIN_2) == LOW) {
      p2++;
    }
    if (p2 == 1 && digitalRead(BUTTON_PIN_2) == HIGH) {
      p2++;
    }
    if (p2 == 2 && digitalRead(BUTTON_PIN_2) == LOW) {
      return 2;
    }
  }
  // shouldn't be possible
  return -1;
}
```

THANK YOU FOR COMING TO THE WORKSHOP

consider

learning more about programming

learning more about electronic circuits

learning more about your interests

learning more