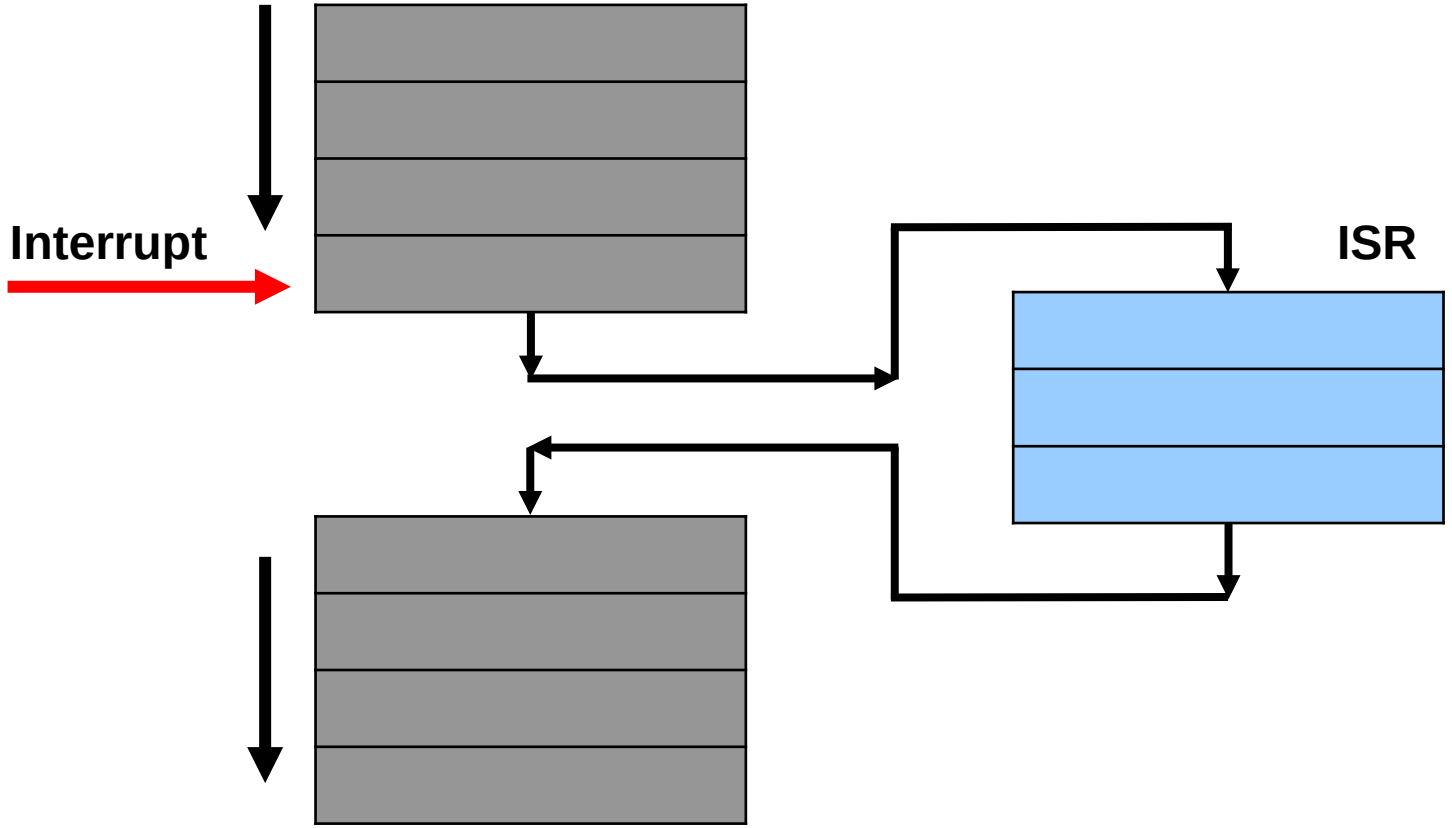
# Engineering Technology (ENGR 101)

# Arduino and Interrupt

# Interrupt

- Interrupts are a way for a microcontroller to temporarily stop what it is doing to handle another task.

- The currently executing program is paused, an ISR (interrupt service routine) is executed, and then your program continues.



# Types of Interrupts

- **Hardware Interrupt:** It happens when an external event occurs like an external interrupt pin changes its state from LOW to HIGH or HIGH to LOW.

- **Software Interrupt:** It happens according to the instruction from the software. For example, *Timer interrupts* are software interrupt.

# Interrupts in Arduino

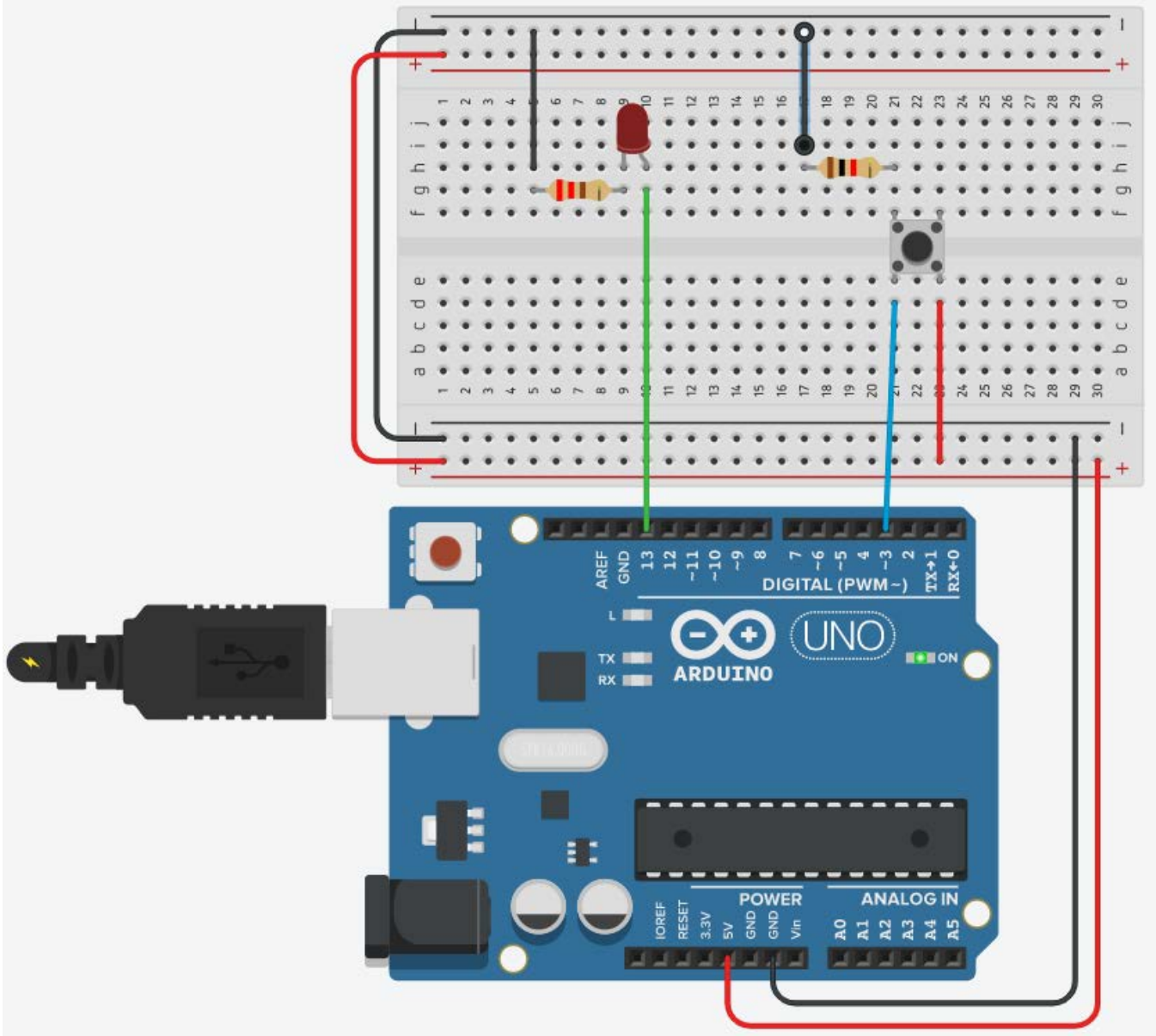
- **External Interrupt:** These interrupt are interpreted by hardware and are very fast. These interrupts can be set to trigger on the event of RISING or FALLING or LOW levels.
  - External interrupts pins: 2 and 3

- **Pin Change Interrupt:** Arduinos can have more interrupt pins enabled by using pin change interrupts.

# Example 1 (no interrupts)

```
const int LEDpin =  13;
const int switchPin = 3;
void setup() {
    pinMode(LEDpin,OUTPUT);
    pinMode(switchPin,INPUT);
}
void loop() {

    handleSwitch();
}
void handleSwitch (){
    int reading = digitalRead(switchPin);
    digitalWrite(LEDpin, reading);
}
```

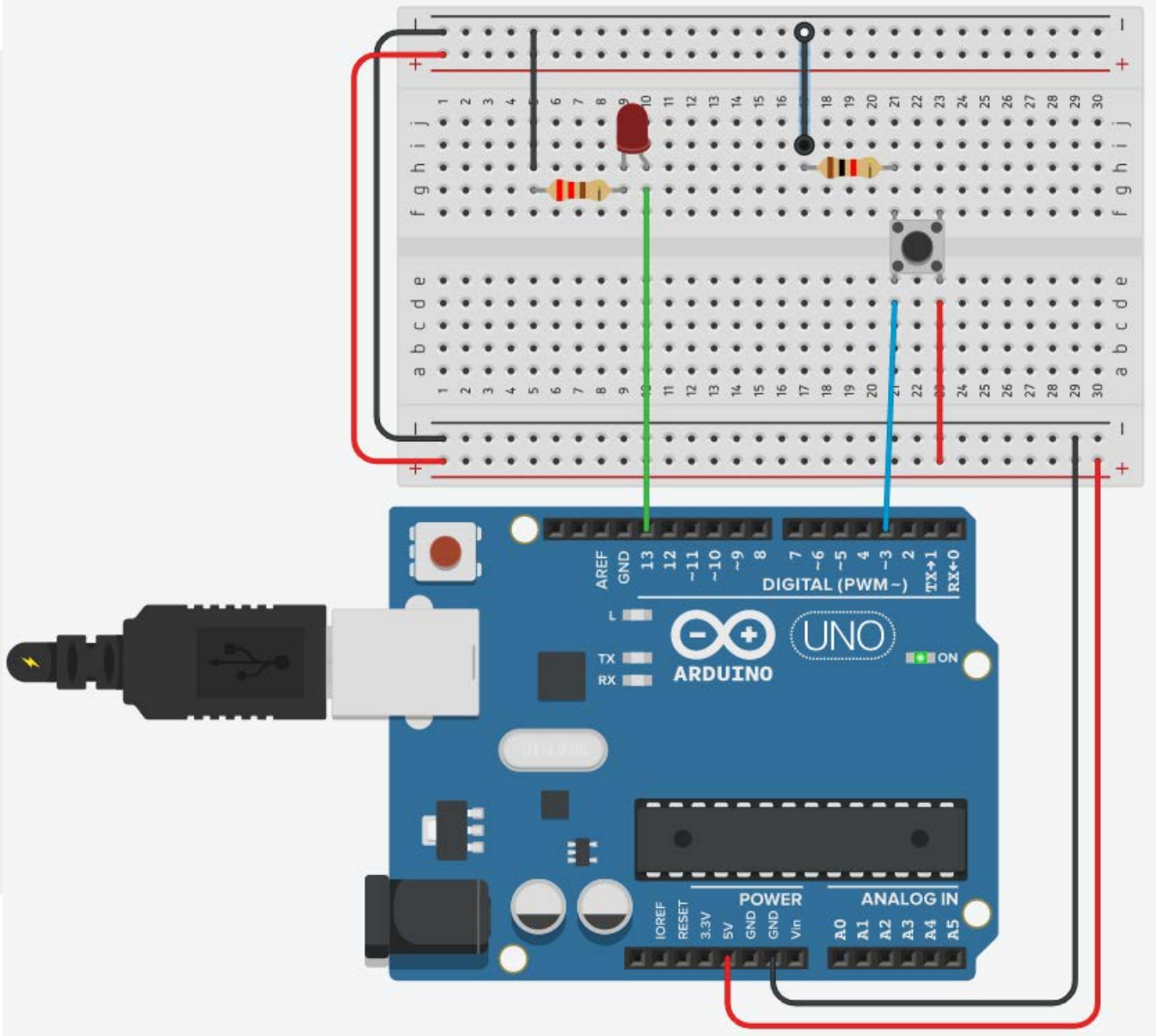


# Example 2 (no interrupts)

```
const int LEDpin =  13;
const int switchPin = 3;
void setup() {
    pinMode(LEDpin,OUTPUT);
    pinMode(switchPin,INPUT_PULLUP);
}
void loop() {

    handleSwitch();
    handleOtherStuff();
}
void handleSwitch (){
    int reading = digitalRead(switchPin);
    digitalWrite(LEDpin, reading);
}

void handleOtherStuff (){
    delay(1000);
}
```



# Using Interrupts

- **Interrupt Service Routine (ISR)**

- Interrupt Service Routine or an Interrupt handler is an event that has small set of instructions in it.

- When an external interrupt occurs, the processor first executes these code that is present in ISR and returns to state where it left the normal execution.

# ISR syntax in Arduino

- **attachInterrupt(** *digitalPinToInterrupt(*pin*)*, ISR, mode**)**

- *pin*: the Arduino pin number. In Arduino Uno, the pins used for interrupt are **2,3**.
- *ISR*: the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.
- *mode*: defines when the interrupt should be triggered. Four constants are predefined as valid values:
  - LOW to trigger the interrupt whenever the pin is low,
  - CHANGE to trigger the interrupt whenever the pin changes value
  - RISING to trigger when the pin goes from low to high,
  - FALLING for when the pin goes from high to low.

# ISR syntax in Arduino

- **digitalPinToInterrupt(pin)**, translate the actual digital pin to the specific interrupt number.
- For example, if a component is connected to pin 3, use **digitalPinToInterrupt(3)** as the first parameter to **attachInterrupt()**.
- Interrupt Service Routine function (ISR) must be as short as possible.
- **Delay**() function doesn’t work inside ISR and should be avoided.
- **millis**() relies on interrupts to count, so it will never increment inside an ISR.

# Example 3 (interrupt)

```
const int LEDpin =  13;
const int switchPin = 3;
void setup() {
    pinMode(LEDpin,OUTPUT);
    pinMode(switchPin,INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(switch_pin), handleSwitch, CHANGE);
}
void loop() {

    // handleSwitch(); ← commented out
    handleOtherStuff();
}
void handleSwitch (){  //ISR
    int reading = digitalRead(switchPin);
    digitalWrite(LEDpin, reading);
}

void handleOtherStuff (){
    delay(1000);
}
```