

Special Aspects of HCI: Prototyping with Arduino

Using the Arduino Open Hardware Platform to sketch and develop physical interactions and tangible user interfaces

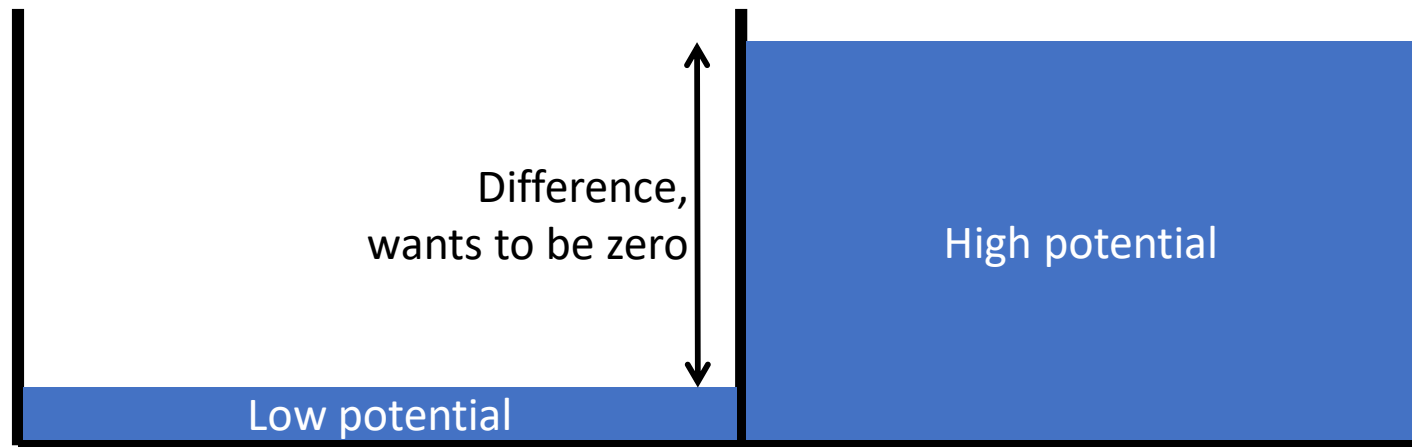
Today:
crash course electrical engineering

Refreshing the basics

- We keep it simple
- Some rules for us
 - Only use direct voltage and direct current
 - Keep Voltage below 30 Volt

Voltage

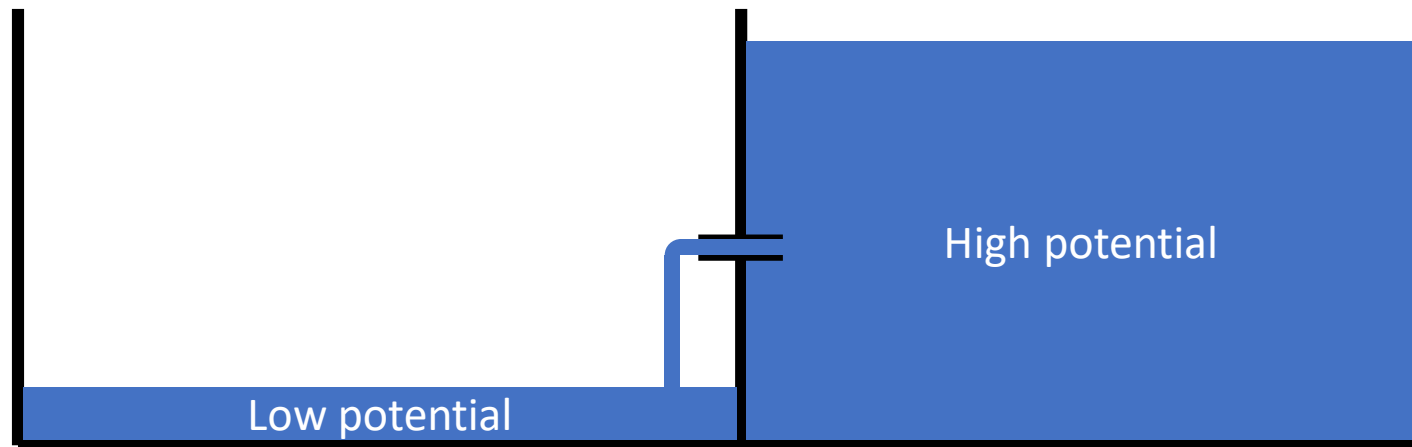
- Symbol: V or U
- Unit: V (Volt)
- is the difference in electric potential between two points
- High difference = high voltage



Water analogy

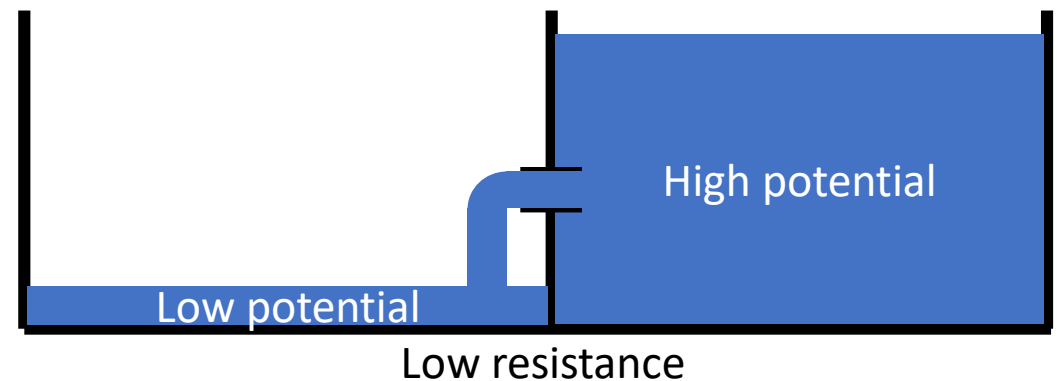
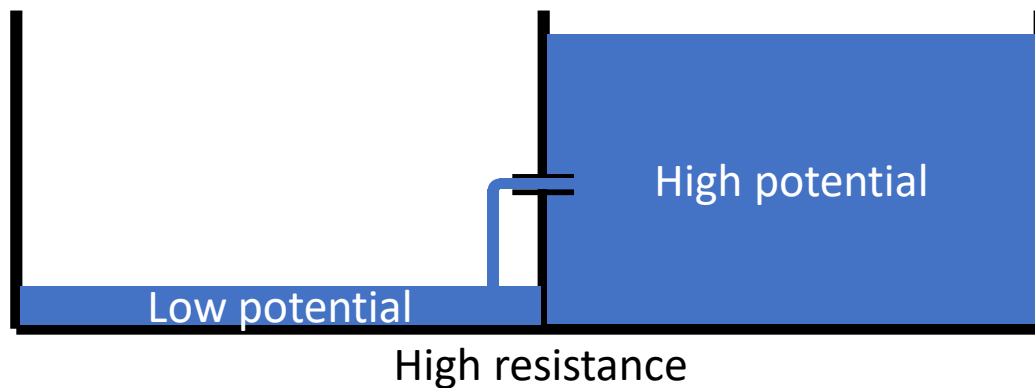
Electrical current

- Symbol: I
- Unit: A (Ampere)
- Is the process of leveling out different potentials
- Is basically the number of electron flowing through a conductor per time



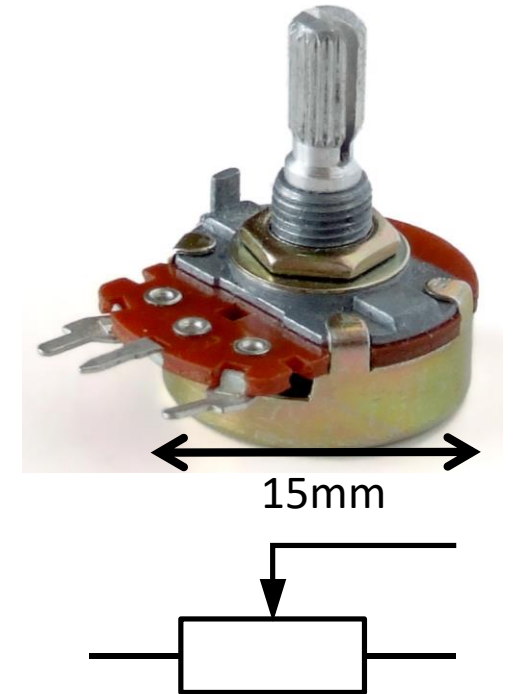
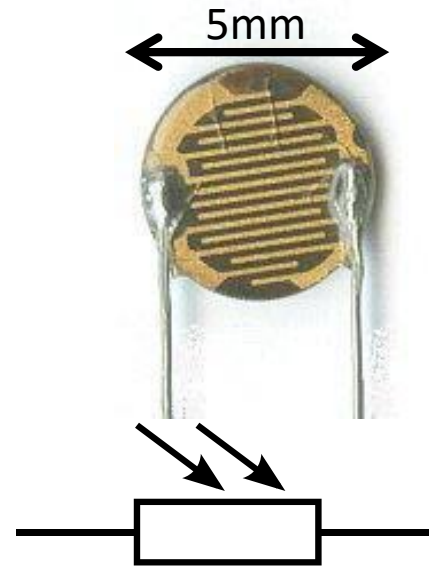
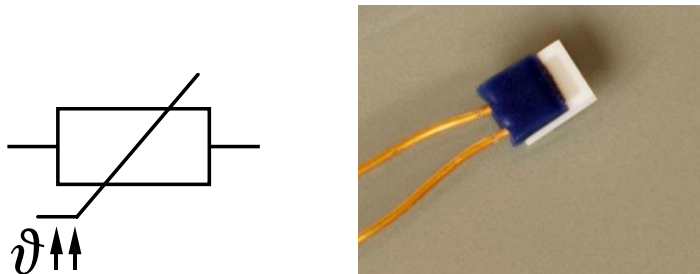
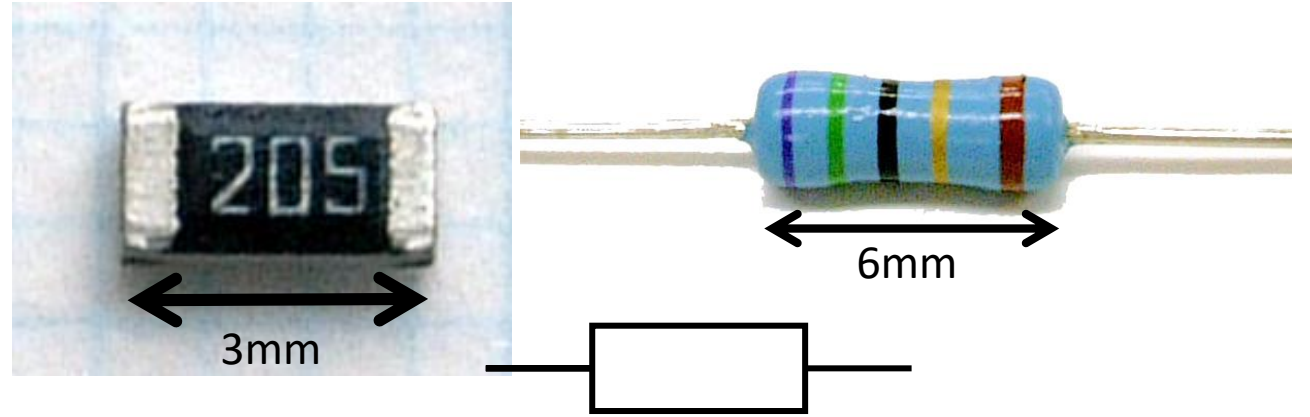
Electrical resistance

- Symbol: R
- Unit: Ω (Ohm)
- is the difficulty for the current to flow through a conductor
- Every conductor has a specific resistance
 - Conductors like copper or gold: low resistance
 - Isolators like plastic or glass: high resistance



Resistor

- Fixed resistance
- Manually changeable
- Resistance depends on other physical parameters (like light or temperature)



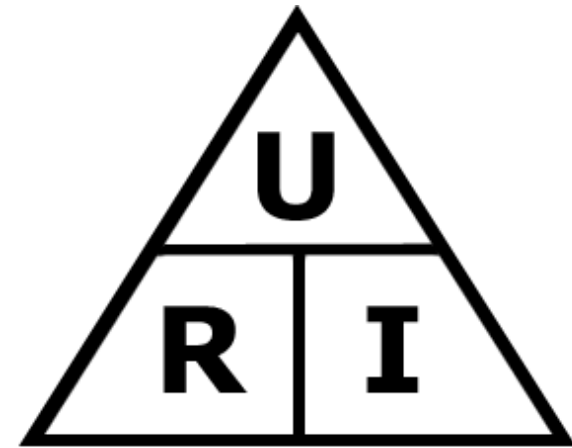
Ohm's law

- How voltage, current and resistance interact?

$$U = R \cdot I$$

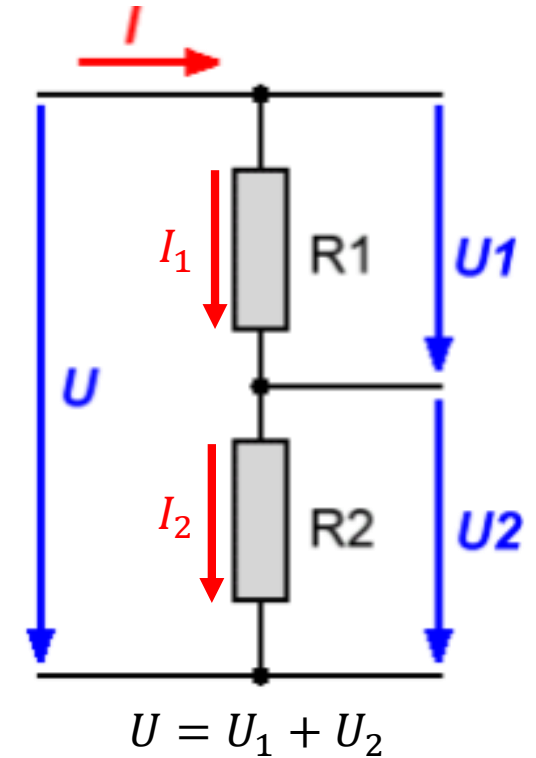
$$I = \frac{U}{R}$$

$$R = \frac{U}{I}$$



Series circuit and voltage divider

- The resistance adds up with a series circuit
 - $R_{total} = R_1 + R_2$
- The total voltage is divided in the ratio of resistances
 - $\frac{U_1}{U_2} = \frac{R_1}{R_2}$
- The current flow is the same in each part
 - $I = I_1 = I_2$



$$\frac{U_1}{U_0} = \frac{R_1}{R_1 + R_2} \Rightarrow U_1 = U_0 \cdot \frac{R_1}{R_1 + R_2}$$

$$\frac{U_0}{U_2} = \frac{R_1 + R_2}{R_2} \Rightarrow U_2 = U_0 \cdot \frac{R_2}{R_1 + R_2}$$

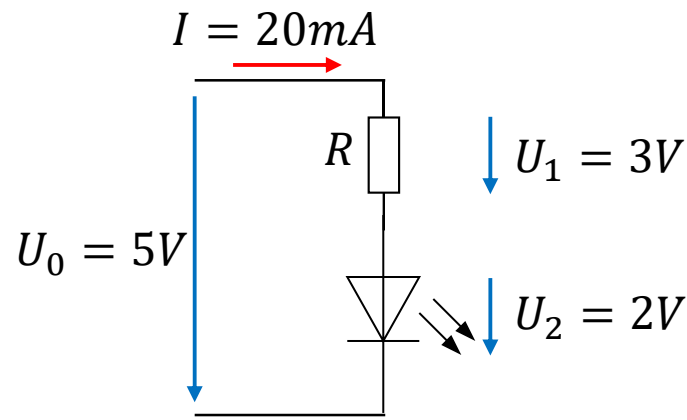
Voltage divider for output

- Some components just can handle a specific amount of voltage
 - Popular example: light emitting diode (LED)
- Use a resistor to lower the voltage

How to calculate the resistor

- Example:

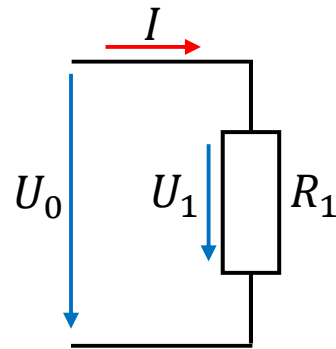
- LED can handle 2 – 2.5V (depending on type, see datasheet)
- LED need around 20mA to light up (depending on type, see datasheet)
- Arduino supplies 5V
- 2.5 – 3V too much, needs to be compensated by resistor



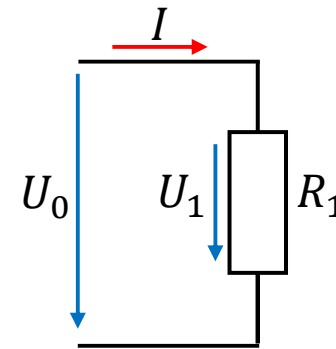
$$R = \frac{U_1}{I} = \frac{3V}{20 \cdot 10^{-3}A} = 150 \Omega$$

Voltage divider for input

- What is the difference between these circuits?



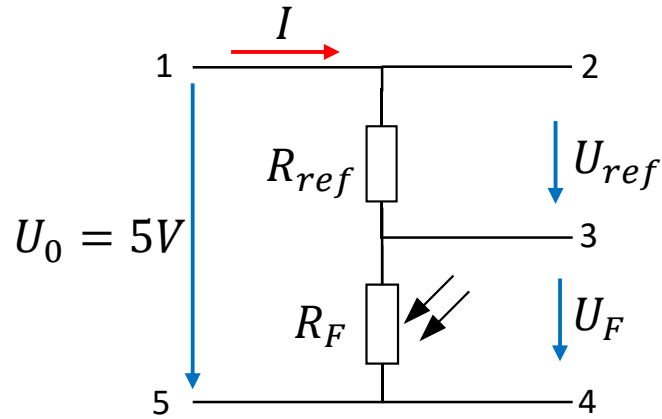
$$\begin{aligned}U_0 &= 5V \\R_1 &= 100\Omega \\U_1 &=? \\I &=?\end{aligned}$$



$$\begin{aligned}U_0 &= 5V \\R_1 &= 200\Omega \\U_1 &=? \\I &=?\end{aligned}$$

- An Arduino can't measure current directly, only voltage

Voltage divider for photoresistor (analog input)



$$R_{ref} = \sqrt{R_{min} \cdot R_{max}}$$

$$R_{ref} = 4,7k\Omega$$

$$R_F = 2 \dots 11k\Omega$$

2kΩ at brightness

11kΩ at darkness

$$U_{F R_{min}} = U_0 \cdot \frac{R_F}{R_{ref} + R_F} = 5V \cdot \frac{2k\Omega}{4,7k\Omega + 2k\Omega} = 1,49V$$

$$U_{F R_{max}} = U_0 \cdot \frac{R_F}{R_{ref} + R_F} = 5V \cdot \frac{11k\Omega}{4,7k\Omega + 11k\Omega} = 3,5V$$

$$U_{ref R_{min}} = U_0 - U_{F R_{min}} = 3,51V$$

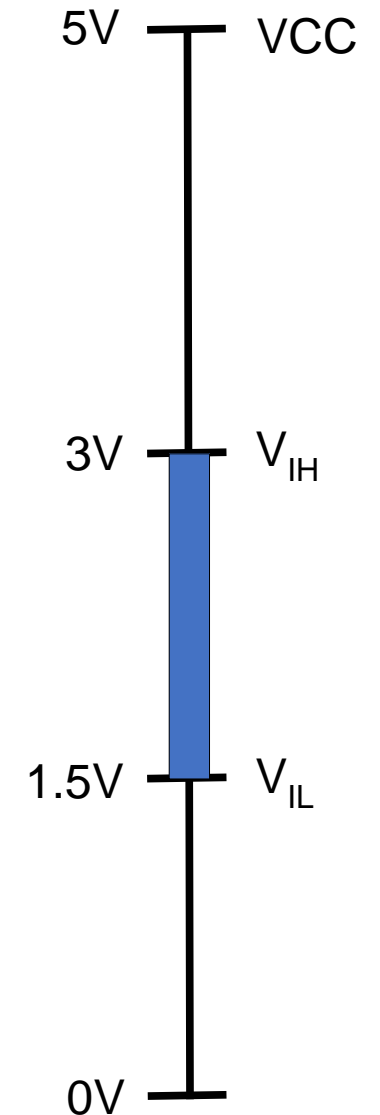
$$U_{ref R_{max}} = U_0 - U_{F R_{max}} = 1,5V$$

To which pin of the Arduino you need to connect point 1 and 5?

Which point (2, 3 or 4) should you connect to the Arduino to measuring the level of brightness? And which Arduino pin do you use?

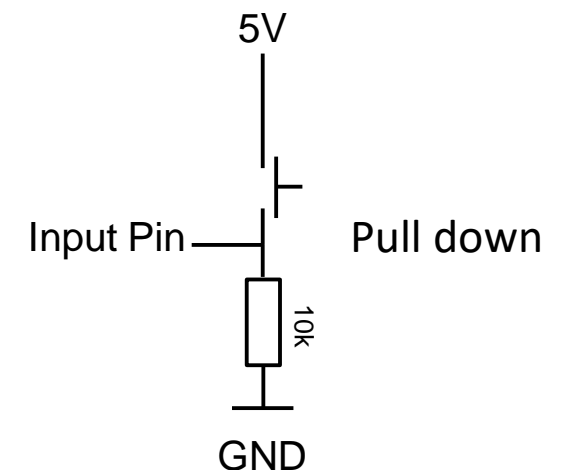
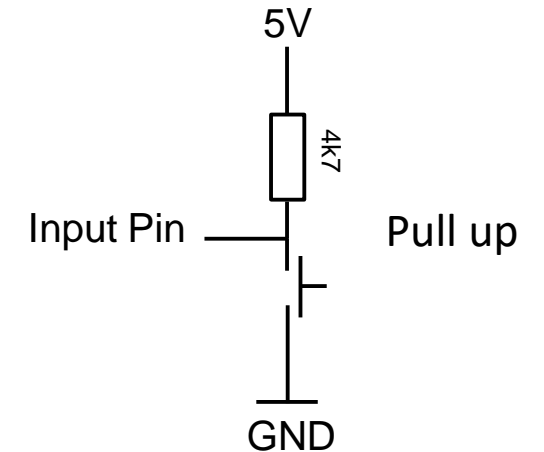
Digital inputs

- A digital pin can have two states: LOW or HIGH
- The voltage have to be greater than 3V to set the pin HIGH
- The voltage have to be lower than 1.5V to set the pin LOW
- The range 1.5V and 3V is undefined
- If the pin isn't connected to anything is somewhere between LOW and High
 - EMF and induction can cause weird errors
 - While using buttons/switches use pull up or pull down resistor to set the input on a defined level when the circuit is open



Pull up / pull down resistor

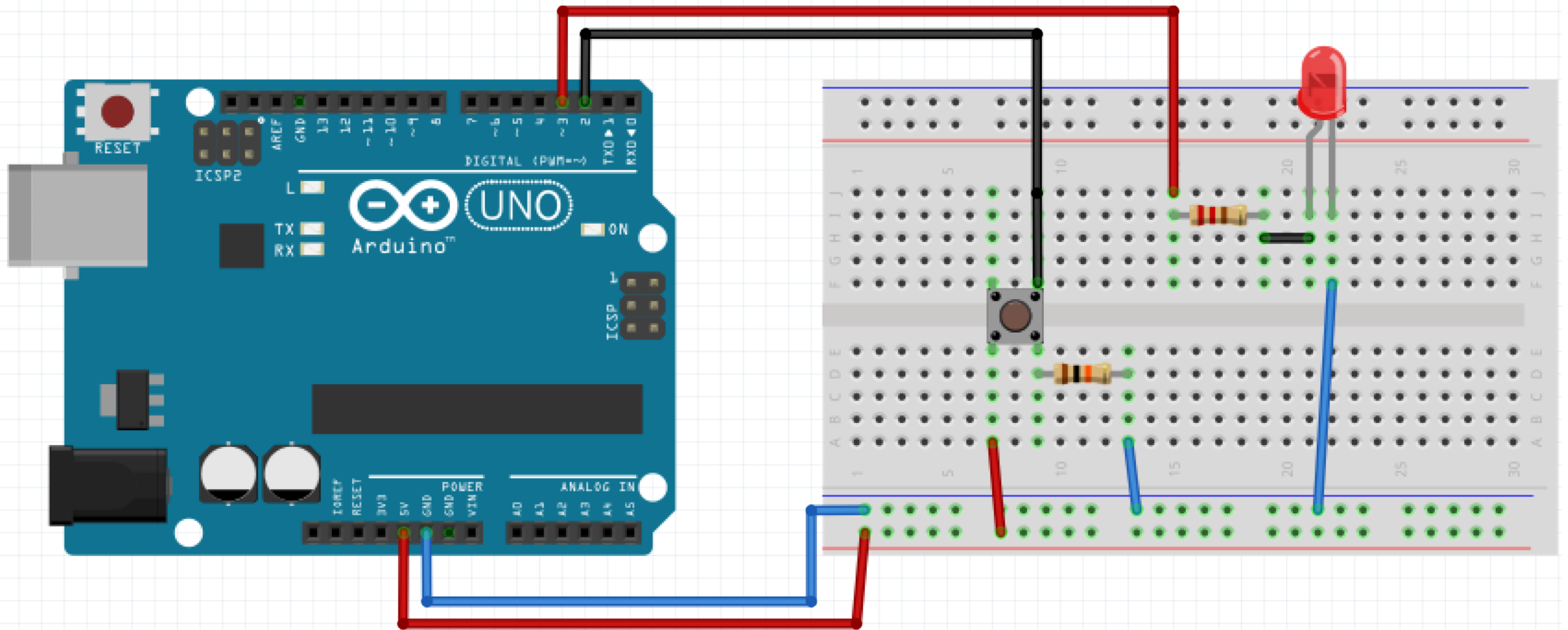
- Pull up
 - Between VCC and Input
 - In open state => the resistor pulls up the input to 5V
 - In closed state => the button pulls the input down to ground
- Pull down
 - Between Input and ground
 - In open state => the resistor pulls down the input to ground
 - In closed state => the button pulls the input up to 5V
- Arduinos have a built in pull up
 - The built in pull up can be used by configuring a digital pin with `pinMode(pin_number, INPUT_PULLUP)`



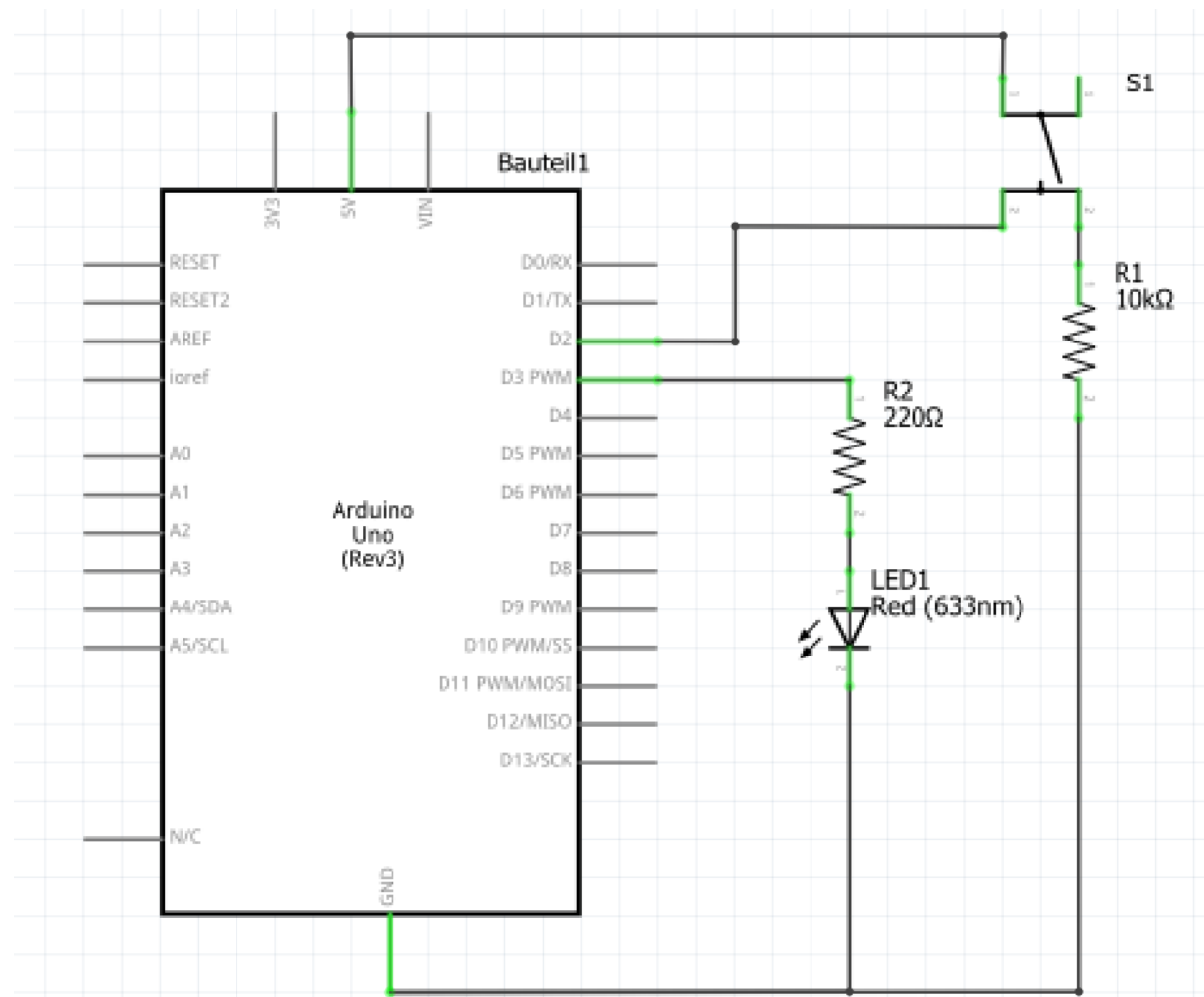
Hands on!

- Goal: control a LED with a button
 1. LED is on when the button is pressed
 2. LED is 5 seconds on after the button is pressed, doesn't matter how long it is pressed
 3. LED toggles each time you press the button, not on release
- Steps:
 - Create an electronic circuit
 - Connect electronic circuit with Arduino board
 - Write code to control the LED with the button
 - Upload code to the Arduino board

Wiring the circuit



Schematic



Methods to get the job done

- `void setup()` and `void loop()`
- `void pinMode(pin, mode);`
 - pin: the pin number
 - mode: INPUT, OUTPUT, or INPUT_PULLUP
- `void digitalWrite(pin, value);`
 - pin: the pin number
 - value: HIGH or LOW
- `int digitalRead(pin);`
 - pin: the pin number
 - Returns: LOW or HIGH
- `void delay(time);`
 - time: time in milliseconds

- One possible solution (1)

```
int ledPin = 3;      // choose the pin for the LED
int inputPin = 2;   // choose the input pin (for a pushbutton)
int buttonValue = 0; // variable for reading the pin status, HIGH=present, LOW=absent

void setup()
{
  pinMode(ledPin, OUTPUT);      // declare LED as output
  pinMode(inputPin, INPUT);    // declare pushbutton as input
}

void loop()
{
  buttonValue = digitalRead(inputPin); // read input value
  digitalWrite(ledPin, buttonValue);
}
```

- One possible solution (2)

```
int ledPin = 3;           // choose the pin for the LED
int inputPin = 2;        // choose the input pin (for a pushbutton)
int buttonValue = 0;     // variable for reading the pin status, HIGH=presed, LOW=released
int previousButtonValue = 0;
int timeLEDon = 5000;    // in ms

void setup()
{
  pinMode(ledPin, OUTPUT); // declare LED as output
  pinMode(inputPin, INPUT); // declare pushbutton as input
}

void loop()
{
  buttonValue = digitalRead(inputPin); // read input value
  if(previousButtonValue == LOW && buttonValue == HIGH)
  {
    digitalWrite(ledPin, HIGH);
    delay(timeLEDon);
    digitalWrite(ledPin, LOW);
  }
  previousButtonValue = buttonValue;
}
```

- Why is this solution bad?
- What is happening if the button is pressed a second time in this 5 seconds?
- What would happen if there would be two LEDs with one button each and the same behavior?

- One possible solution (3)

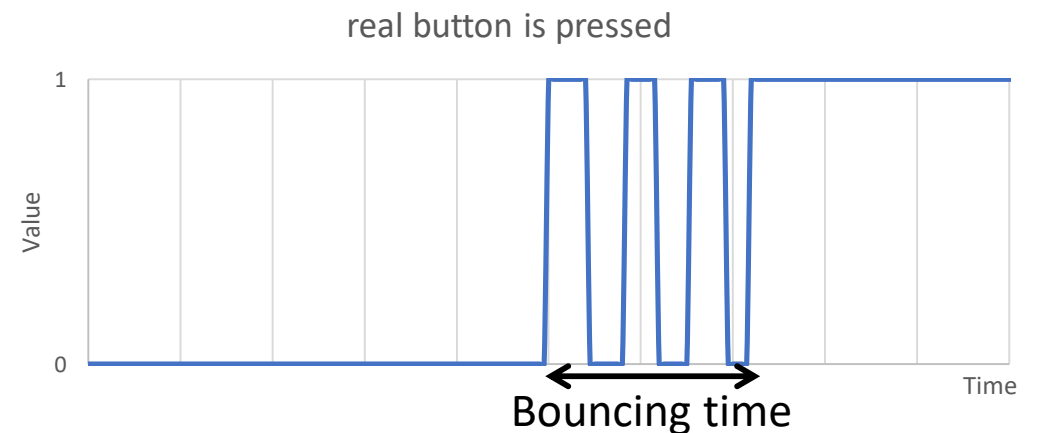
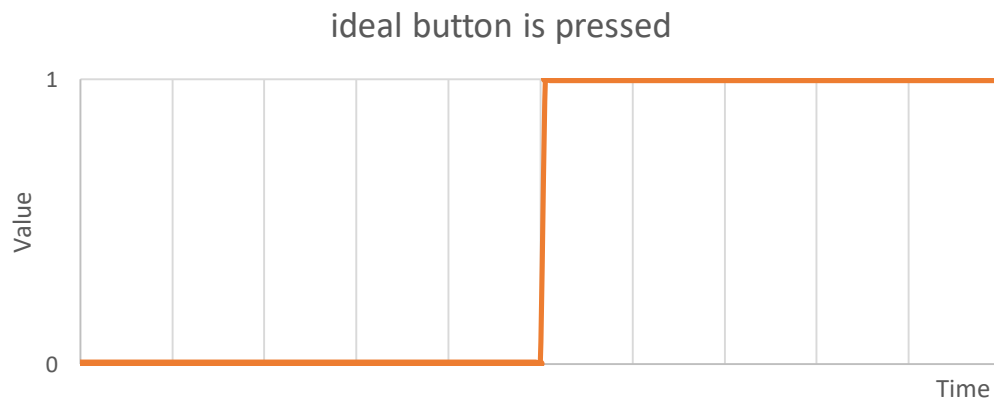
```
int ledPin = 3;      // choose the pin for the LED
int inputPin = 2;   // choose the input pin (for a pushbutton)
int buttonValue = 0; // variable for reading the pin status, HIGH=present, LOW=absent
int previousButtonValue = 0;
int ledState = 0;   // variable for storing the LED state

void setup()
{
  pinMode(ledPin, OUTPUT); // declare LED as output
  pinMode(inputPin, INPUT); // declare pushbutton as input
}

void loop()
{
  buttonValue = digitalRead(inputPin); // read input value
  if(previousButtonValue == LOW && buttonValue == HIGH)
  {
    ledState = !ledState;           // toggle ledState
    digitalWrite(ledPin, ledState);
  }
  previousButtonState = buttonState;
}
```

Did everything work?

- Maybe not
- One reason could be the bouncing of buttons
- Mechanical buttons physically vibrate - bounce - when they are first pressed or released.
- This creates spurious state changes that need to be filtered or "debounced".
- Bouncing time depends on the button, mostly under 20 ms, can be higher



Hands on!

- Goal: include some kind of debouncing
- Steps:
 - Use previous circuit
 - Do it manually
 - Detect a signal edge and wait for a couple of milliseconds
 - After that, process the input as usually
 - Or use Bounce library or Button library
 - Bounce library: <https://playground.arduino.cc/Code/Bounce>
 - Button library: <https://playground.arduino.cc/Code/Button>

Simple manually debounce

```
int debouncingTime = 20; // in ms
int buttonValue = 0; // variable for reading the pin status, HIGH=presed, LOW=released
int previousButtonValue = 0;

void setup() {
  pinMode(inputPin, INPUT); // declare pushbutton as input
}

void loop(){
  if(millis() - startDebounceTime > debouncingTime){
    buttonValue = digitalRead(inputPin); // read input value
    if(buttonValue != previousButtonValue){
      startDebounceTime = millis();
    }
    previousButtonValue = buttonValue;
  }
}
```