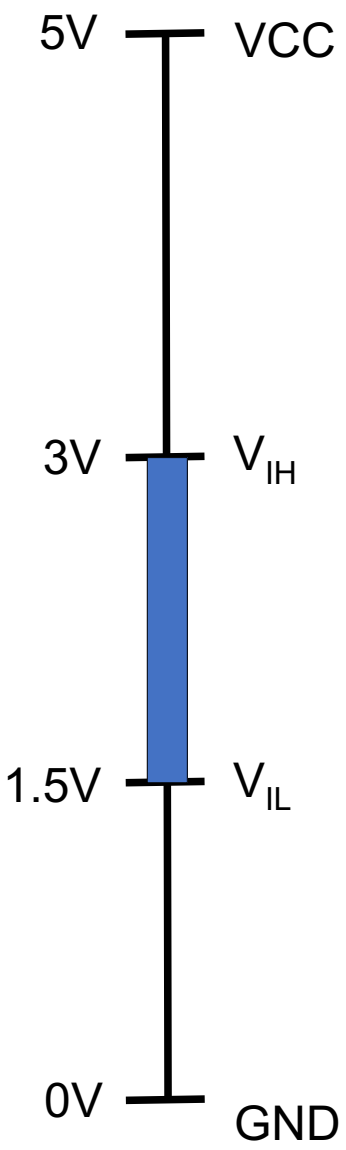
# Special Aspects of HCI: Prototyping with Arduino

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

# Today:
# analog vs digital signals

# Digital signals

- Can be 0 or 1, LOW or HIGH
- For inputs:
  - The voltage have to be greater than 3V to be recognized as HIGH
  - The voltage have to be lower than 1.5V to be recognized as LOW
  - A voltage of 2.5V can be LOW or HIGH depending on the previous state
    - If its rising from low to high (1V->2.5V), the state is still LOW
    - If its falling from high to low (4.5V->2.5V), the state is still HIGH
- For outputs:
  - HIGH = 5V
  - LOW = 0V

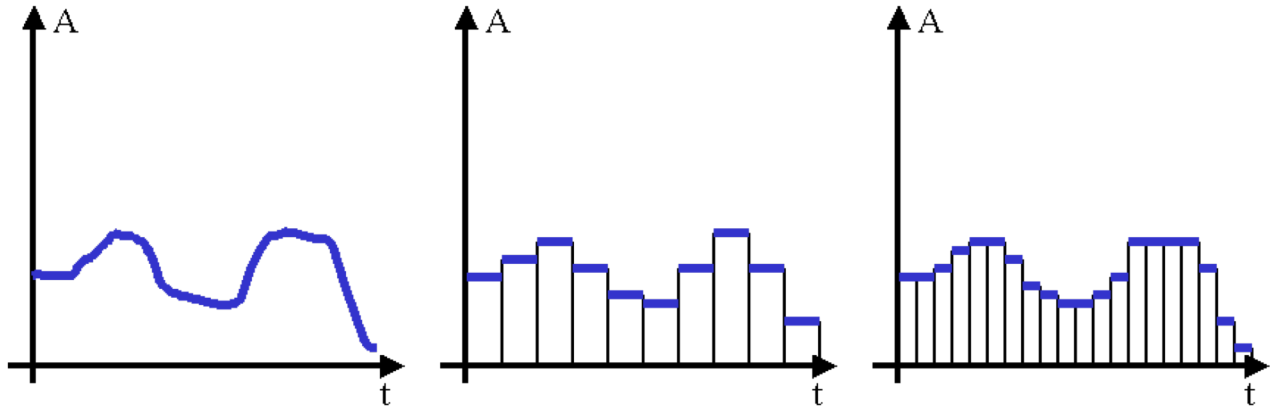


# Analog signals

- Can represent a infinite amount of values between to points (0V and 5V)
- Its continuous in time, for each point in time there is a value
- Physical phenomenon can be descript with analog signals
  - E.g. Light, sound, temperature, voltage
- To process an analog signal with an Arduino it need to convert to a digital signal

# Analog digital converter

- in a specific time interval the analog signal is measured
- the measured value is converted into a digital value according to the resolution of the converter




Analog signal – continuously varying

Digital signal – large time divisions

Digital signal – small time divisions

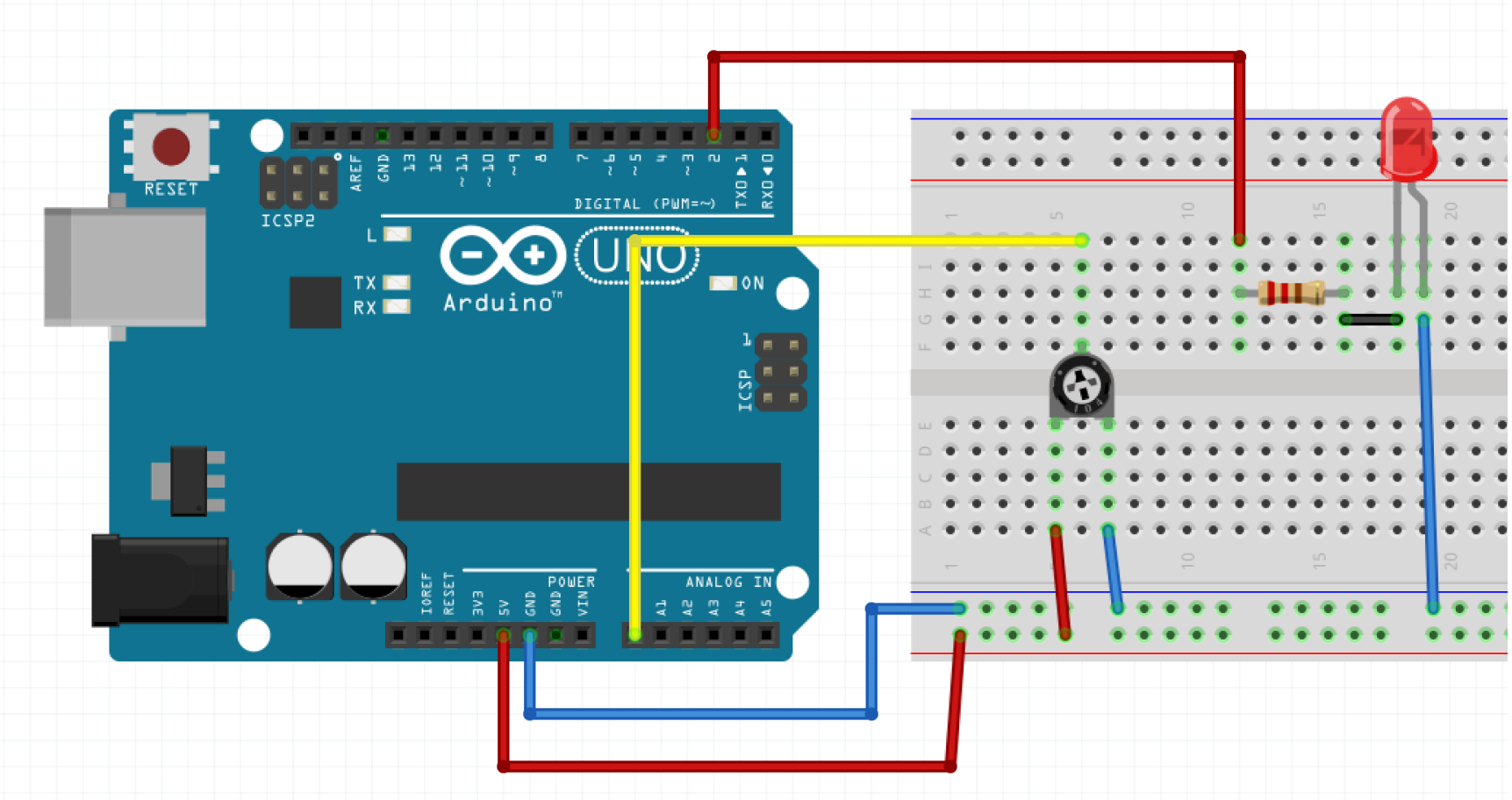
# Analog inputs

- Arduino uno has 6 analog inputs (A0-A5)
- Analog inputs only can read voltages between 0 and 5V
- Arduino ADC has a resolution of 10 bits -> 1024 steps, 0 - 1023
- Values can be read in 5V/1024 = 0,00488V steps
- Analog inputs don’t have to be initialized with pinMode()
- Get the value from analog input with analogRead(pin_number);

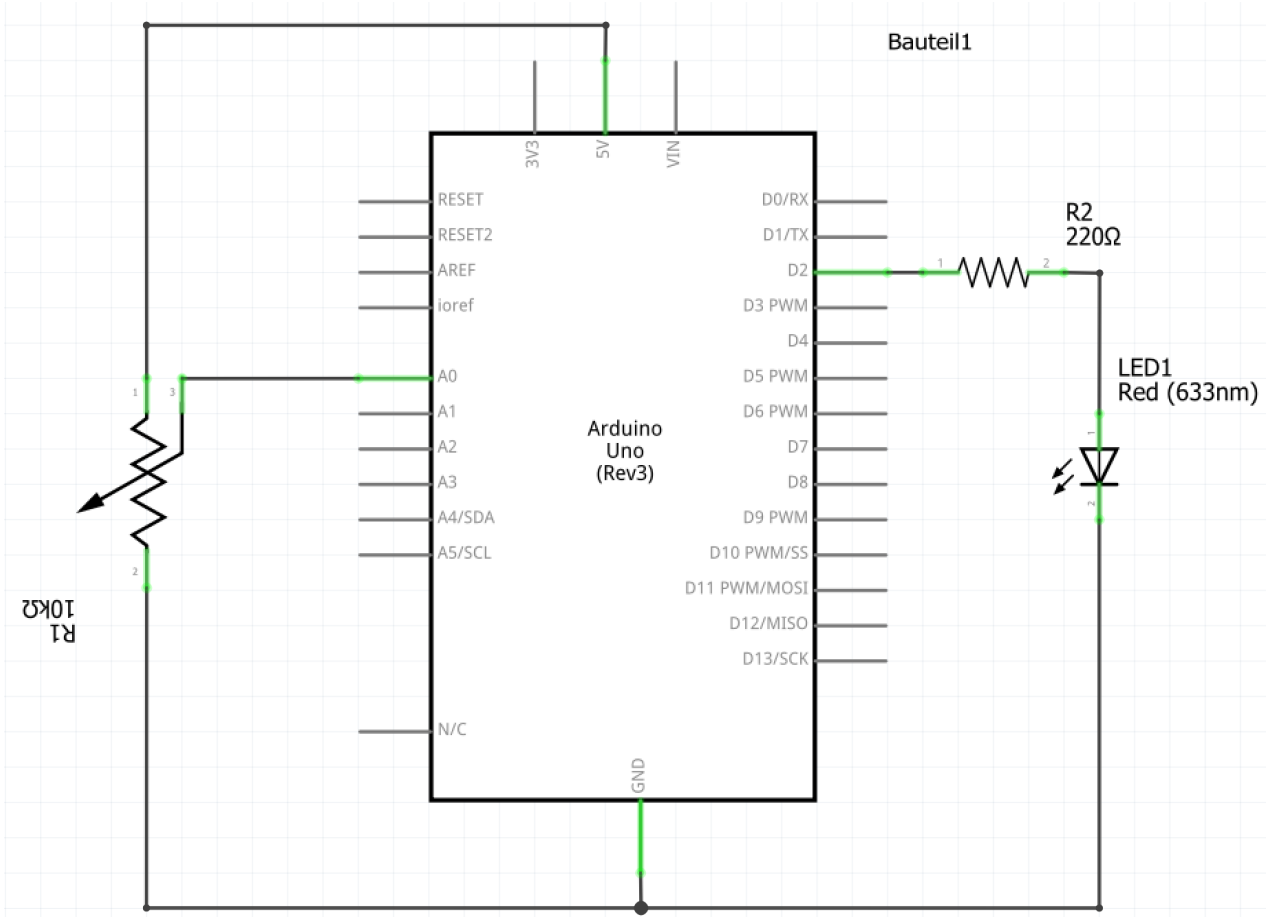
# Hands on

- Goal: control a LED with a potentiometer
  - For analog value from 0-255: LED off
  - 256-511: LED blink 1 time per second
  - 512-767: LED blink 2 times per second
  - 768-1023: LED blink 3 times per second
  - On:off ration = 1:1

# Wiring the circuit



# Schematic


Bauteil1
3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Arduino
Uno
(Rev3)
R2
220Ω
LED1
Red (633nm)
R1
10kΩ

# 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 analogRead(pin);
  - pin: the pin number of analog input
  - Returns: an integer between 0 and 1023
- void delay(time);
  - time: time to wait in milliseconds
- unsigned long millis();
  - Return: Number of milliseconds since the program started (unsigned long)

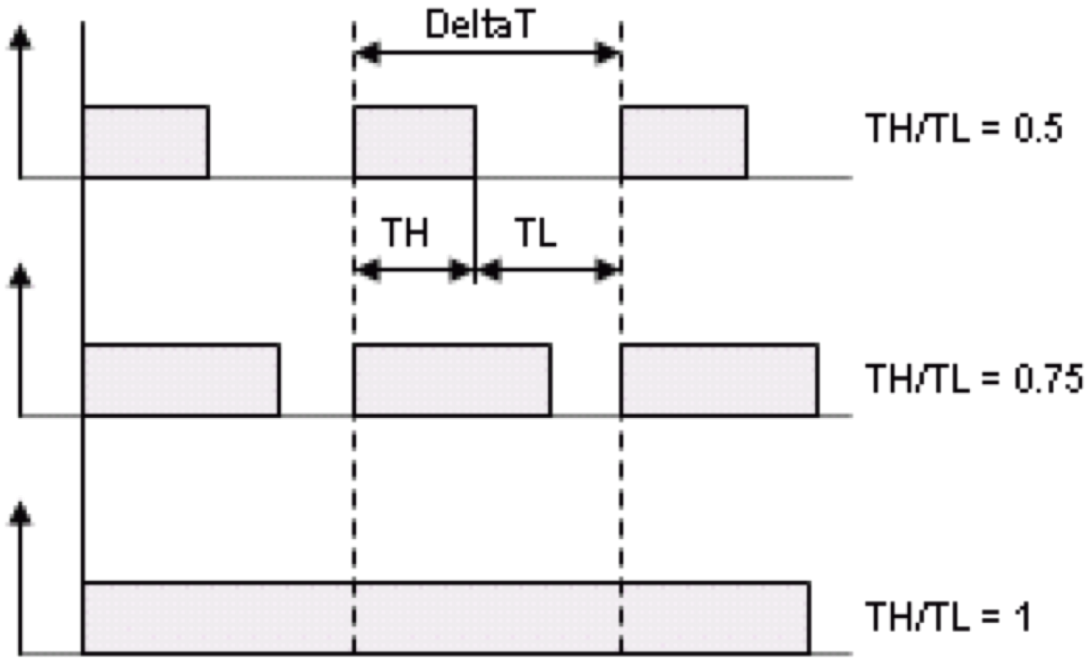
```
int ledPin = 2;                 // choose the pin for the LED
int analogPin = 0;              // choose the input pin
int potiValue = 0;              // variable to store the value read
int waitingTime = 0;            // variable to store the time to wait before toggle LED
int lastToggle = 0;              //variable to store the last time the led was toggled
int ledState = 0;

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

void loop()
{
  potiValue = analogRead(analogPin);        // read the input pin
  if(potiValue <=255)
  {
    waitingTime = -1;
    digitalWrite(ledPin, LOW);
  }
  else if(potiValue <= 511)
  {
    waitingTime = 500;
  }
  else if(potiValue <= 767)
  {
    waitingTime = 250
  }
  else
  {
    waitingTime = 167;
  }

  if((millis() - lastToggle) >= waitingTime && waitingTime > 0)
  {
    ledState = !ledState;                   // toggle ledState
    digitalWrite(ledPin, ledState);
    lastToggle = millis();
  }
}
```

# Analog outputs

- Are used to dim light or control speed of a motor
- There are no real analog outputs on an Arduino Uno
  - There are Arduinos with real analog outputs, but they are more expensive
- You can simulate an analog signal with Pulse-Width-Modulation (PWM)

# Pulse-Width-Modulation

- A PWM signal is a square wave with values of low and high (0V or 5V)
- It has a fixed time period (Delta T)
  - Default: 2ms (500Hz)
- You can control the ratio between high and low (duty-cycle)
  - In 8 bit resolution
  - 0 = always off
  - 255 always on



# Pulse-Width-Modulation

- Which pins can be used for PWM?

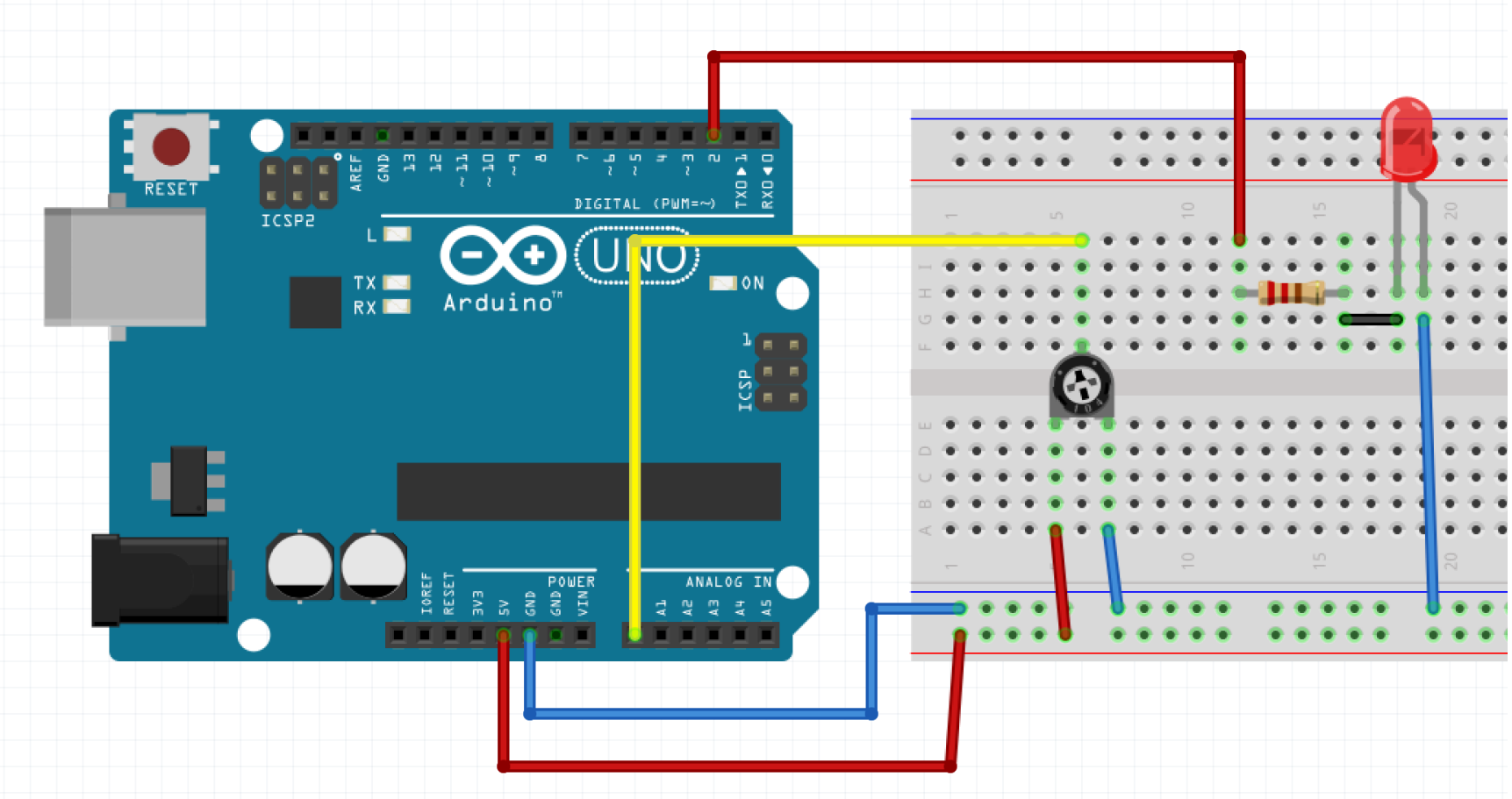

- How to use?
  - Initialize the pin as output: pinMode(pwmPin, OUTPUT);
  - Write analog value to pin: analogWrite(pwmPin, value);
- Use for what?
  - E.g. to dim LED by turning it rapidly on and off again

# Hands on!

- Goal: dim a LED with a potentiometer

- Steps:
  - Use the previous circuit
  - Adjust your previous code
  - Use the analog value from potentiometer to dim the LED
    - Attention: potentiometer value range from 0-1023 and dim value range from 0-255

# Wiring the circuit



```
int ledPin = 2;             // LED connected to digital pin 2
int analogPin = 0;          // potentiometer connected to analog pin 0
int potiValue = 0;          // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT); // sets the pin as output
}

void loop()
{
  potiValue = analogRead(analogPin); // read the input pin
  analogWrite(ledPin, potiValue / 4);
}
```

# Hands on!

- Goal: combine your knowledge
  - Use button(s)
  - Use LED(s)
  - Use some kind of analog input (potentiometer, fotoresistor…)

- Play around and have fun!