Special Aspects of HCI: Prototyping with Arduino

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

Today: Introduction

About this course

- Lecture
  - Theoretical background and hand on sessions

- Project Work
  - Create a interactive thing including a Arduino (or some other kind of microcontroller)
  - Presenting your project idea in the first week of June
  - In groups with up to 3 persons
  - Document your process of creating
  - Fix deadline: 30.9.2018 (early submission is possible)

Timetable

Session	Date	Topic
1		Introduction
2		Crash course electrical engineering
3		Analog vs digital signals
4		Communication
5		
6		Presentation of project ideas
7		
8		
9		

After that: project work.

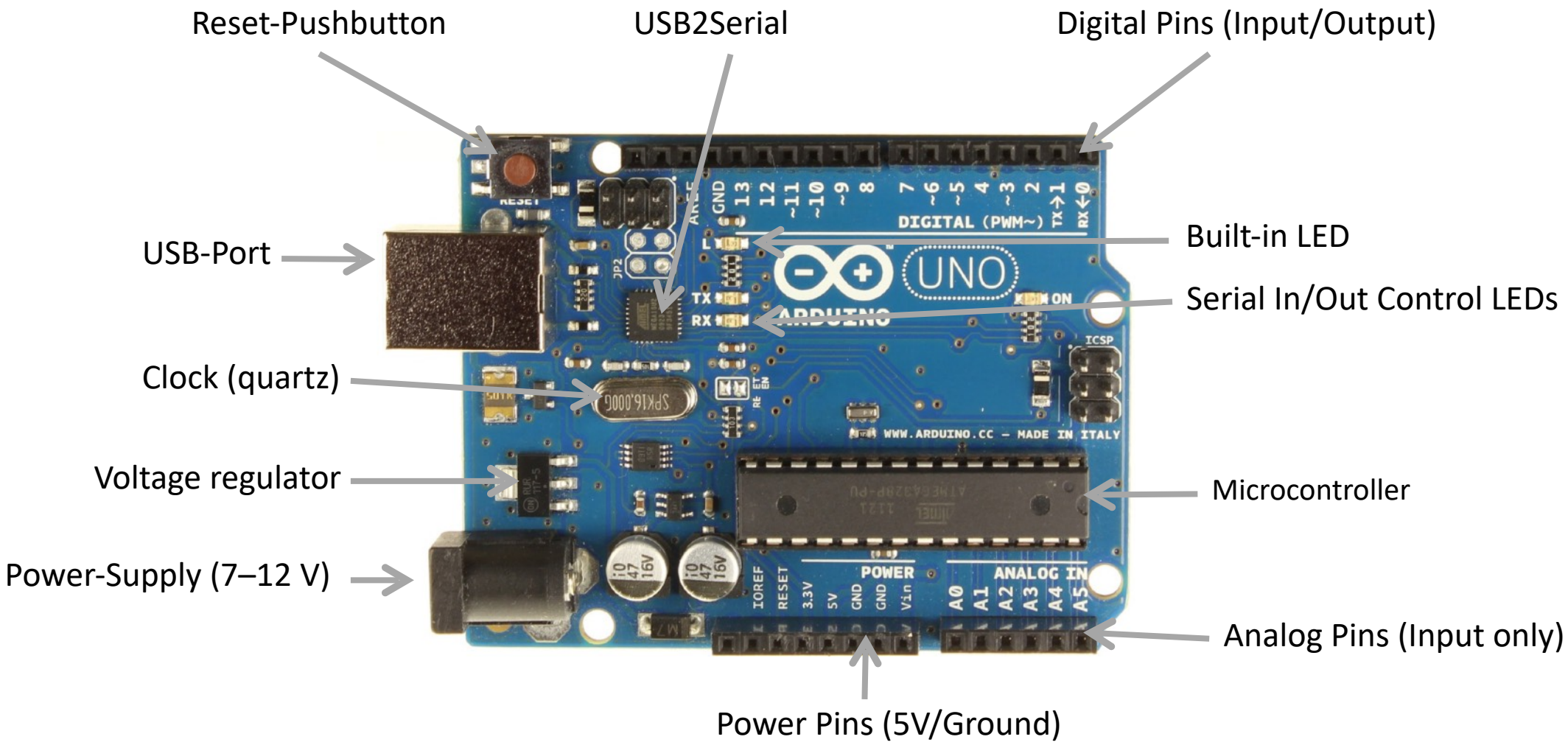
Old projects TBD

Where to get information about Arduinos and inspiration for your project?

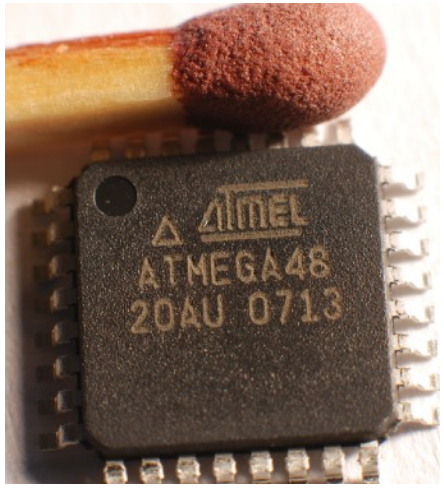
- Books and magazines
  - Arduino Cookbook (Michael Margolis, O'REILLY)
  - Programming Interactivity (Joshua Noble, O'REILLY)
  - MAKE: MAGAZINE

- Internet
  - arduino.cc
  - instructables.com

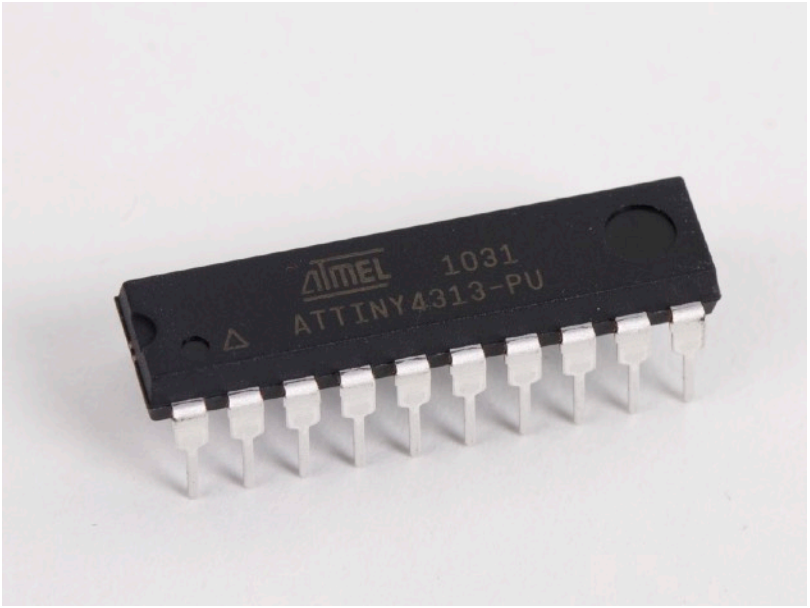
Let’s have look at an Arduino Uno



What is a microcontroller?



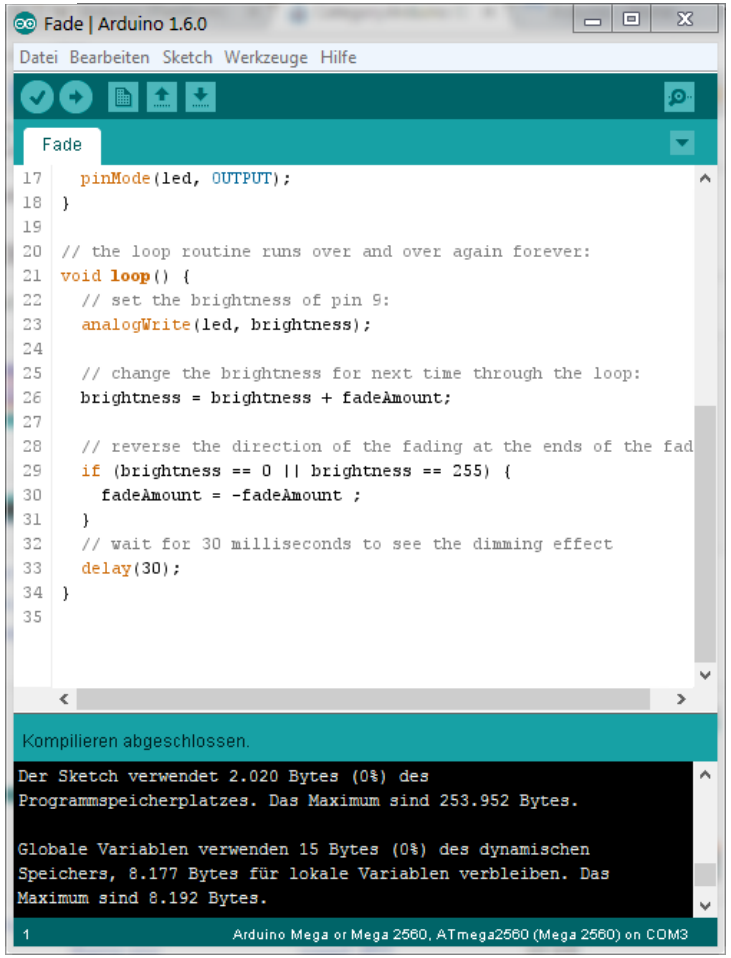
- Small computer on a single integrated circuit (IC)
- Contains a processor core, memory, and programmable input/output peripherals
- Program memory is often included on chip
- Typically small amount of RAM
  (4-8kb in Arduino ATmega case)
- Microcontrollers are designed for embedded applications, usually programmed for one specific task
- Usually just one process at a time



1. „Chip“ by Henner Zelleris licensed under CC BY-SA 2.0.
2. „ATtiny4313-PU “ by Windell Oskay licensed under CC BY 2.0.

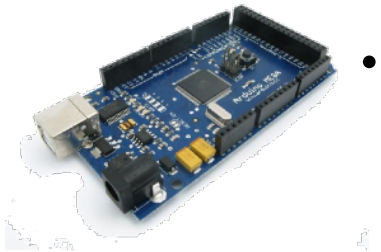
Arduino Platform

- Open source hardware and software platform
- Designed to make the process of using electronics in multidisciplinary projects more accessible
- Based on different Atmel AVR microcontrollers
- Make the functions of the microcontroller easily accessible through:
  - Pin bar for input and output
  - USB interface for programming
  - Power supply
  - Reset-Button



Arduino Boards & Shields

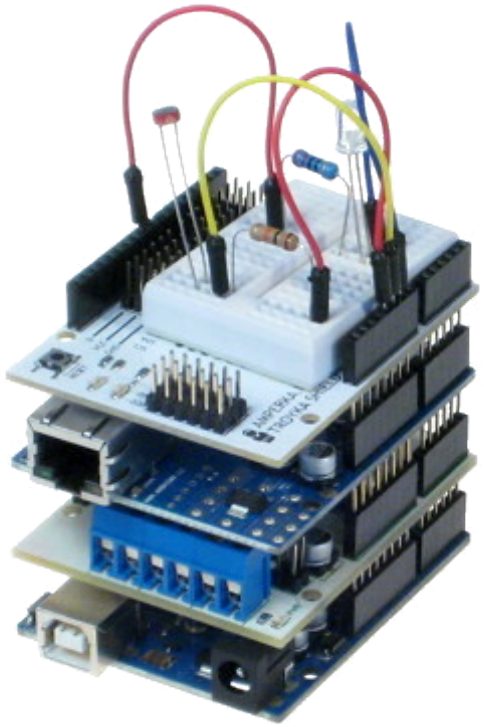
- Arduino Duemilanove
  - ATmega168/328P
  - 14/6 Pins (digital/analog)



- Arduino Mega(2560)
  - ATmega1280/2560
  - 54/16 Pins (digital/analog)

- Arduino Nano
  - ATmega168 or ATmega328
  - 14/8 Pins (digital/analog)

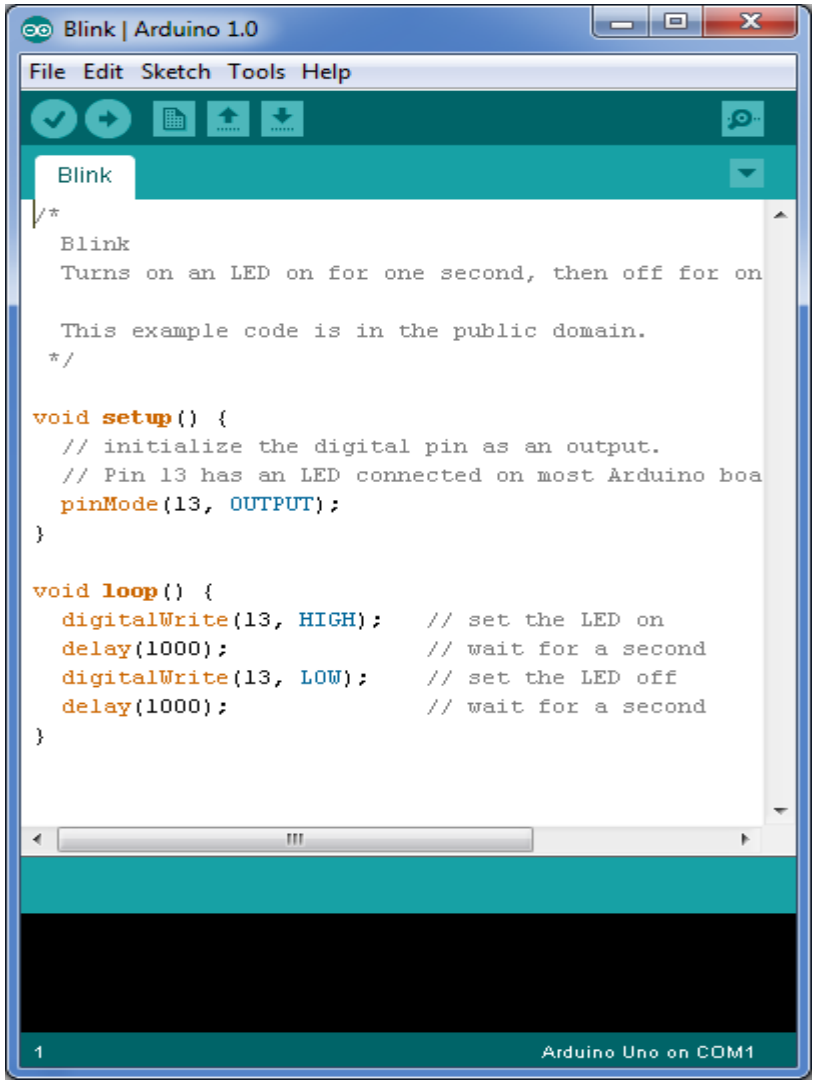
- Arduino Mini Pro
  - ATmega168
  - 14/6 Pins (digital/analog)



- Shields are stackable
- Shields adding functionality to Arduino boards like:
  - Networking
  - Controlling electrical motor
  - Sound
  - …

Arduino programming

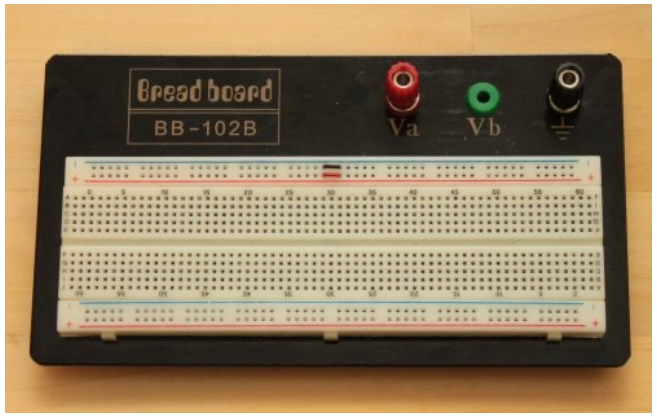
- Arduino programming language is a combination of C and C++
- Arduino IDE
- Plugin for Eclipse and Visual Studio
- Each Arduino program have to consists at least out of a setup and a loop function
  - void setup() – initializing the microcontroller
  - void loop() – measuring and processing input generate output

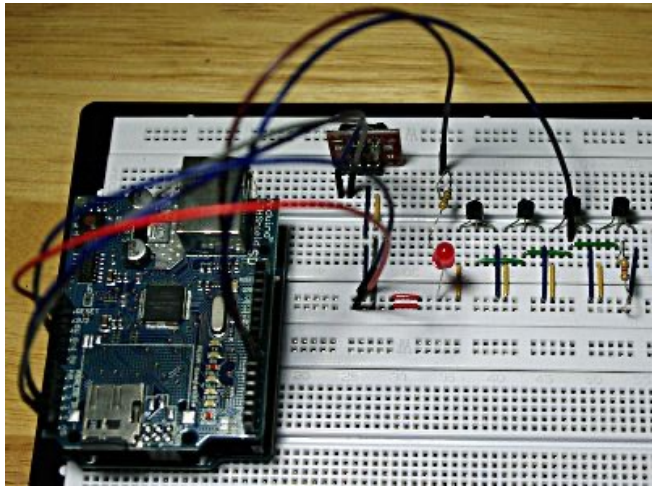


IPO Model

- Measure Input
  - Analog and digital: Buttons, temperature, light, sound, serial devices, …
- Process
  - Process input through the program code
- Generate Output
  - Digital: High/Low, PWM, serial signals

Prototyping Tools

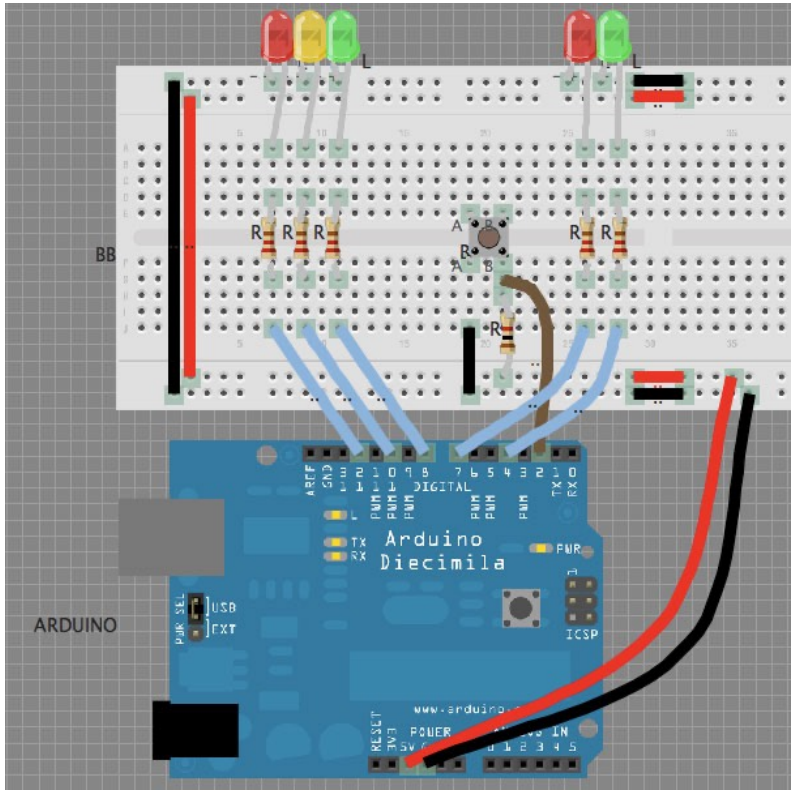


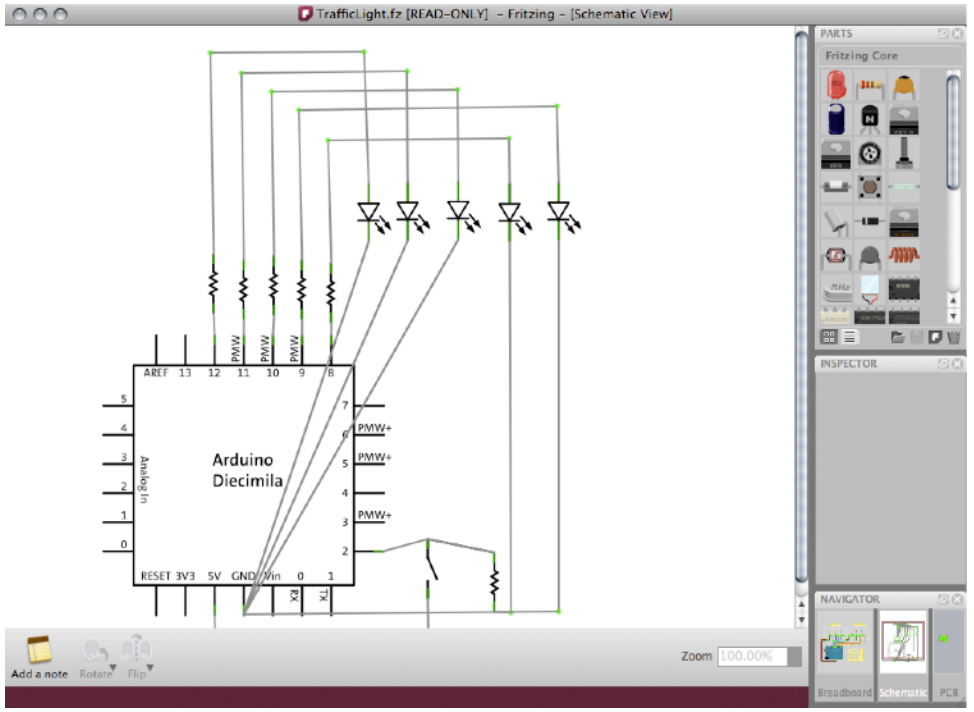


- Protoboard / Breadboard
  - Vertical and Horizontal connectors
  - Plug wires and connect components
  - Avoid soldering
  - Speed up sketching
  - Avoid complex planning of electrical circuits

Planning and documentation

Fritzing (www.fritzing.org)





Where to get parts for your project?

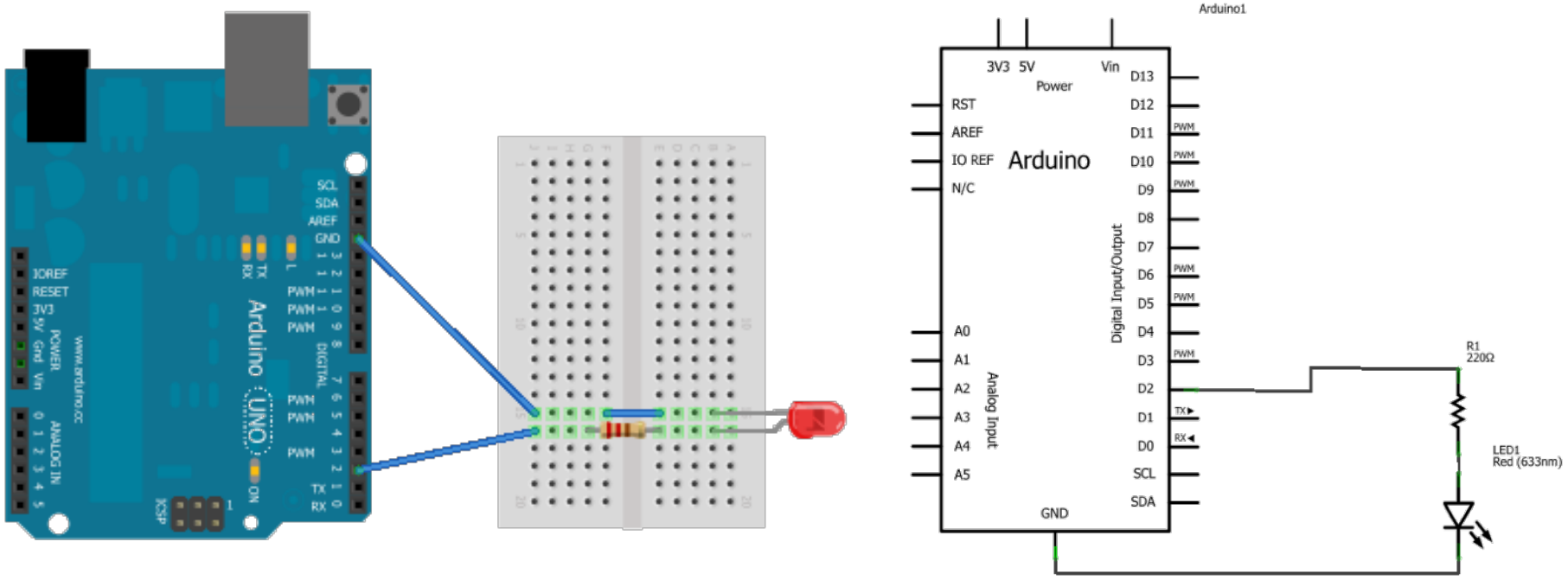
Hands on!

- Goal: Let LED blink

- Steps to go:
  - See through the kits
  - Create an electronic circuit
  - Connect electronic circuit with Arduino board
  - Write code to let LED blink 5 times/second
  - Upload code to the Arduino board

- Play arround:
  - Change parameters, add more LEDs
  - Be inspired for more complex projects
  - Have fun!

Wiring the circuit



Long leg of the Led is the positive pole.

Use a 220 Ohm resistor to limit the current (Why? We'll learn it in a later session)

- Use this basic structure

```
// the setup function runs once when you press reset or power the board
void setup() {
  // insert initialization here
}

// the loop function runs over and over again forever
void loop() {
  // insert program logic here
}
```

- Methods to get the job done
  - pinMode(pin, mode);
    - pin: the pin number
    - mode: INPUT, OUTPUT, or INPUT_PULLUP
  - digitalWrite(pin, value);
    - pin: the pin number
    - value: HIGH or LOW
  - delay(time);
    - Time: time in milliseconds

- One possible solution

```
const int pinNumber = 2;
const int waitingTime = 100; // in ms


// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin 2 as an output.
   pinMode(pinNumber, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
   digitalWrite(pinNumber, HIGH);   // turn the LED on by making the voltage HIGH
   delay(waitingTime);                        // wait for 100 ms
   digitalWrite(pinNumber, LOW);   // turn the LED off by making the voltage LOW
   delay(waitingTime);                 // wait for 100 ms
}
```

Special Aspects of HCI: Prototyping with Arduino

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

Today: communication

Types of communication

- Serial
  - One wire for data
  - Bits are transmitted one after another

- Parallel
  - Multiple wire for data
  - All bits are transmitted at the same time

Example transfers of 01100011

Universal Asynchronous Receiver Transmitter (UART)

- All Arduino boards have at least one UART / serial port
- UART is for serial communication
- Does only allow two endpoints
- UART can be used to show debug messages on a PC
- UART can also be used for communication between two Arduinos

UART Arduino Code Snippets

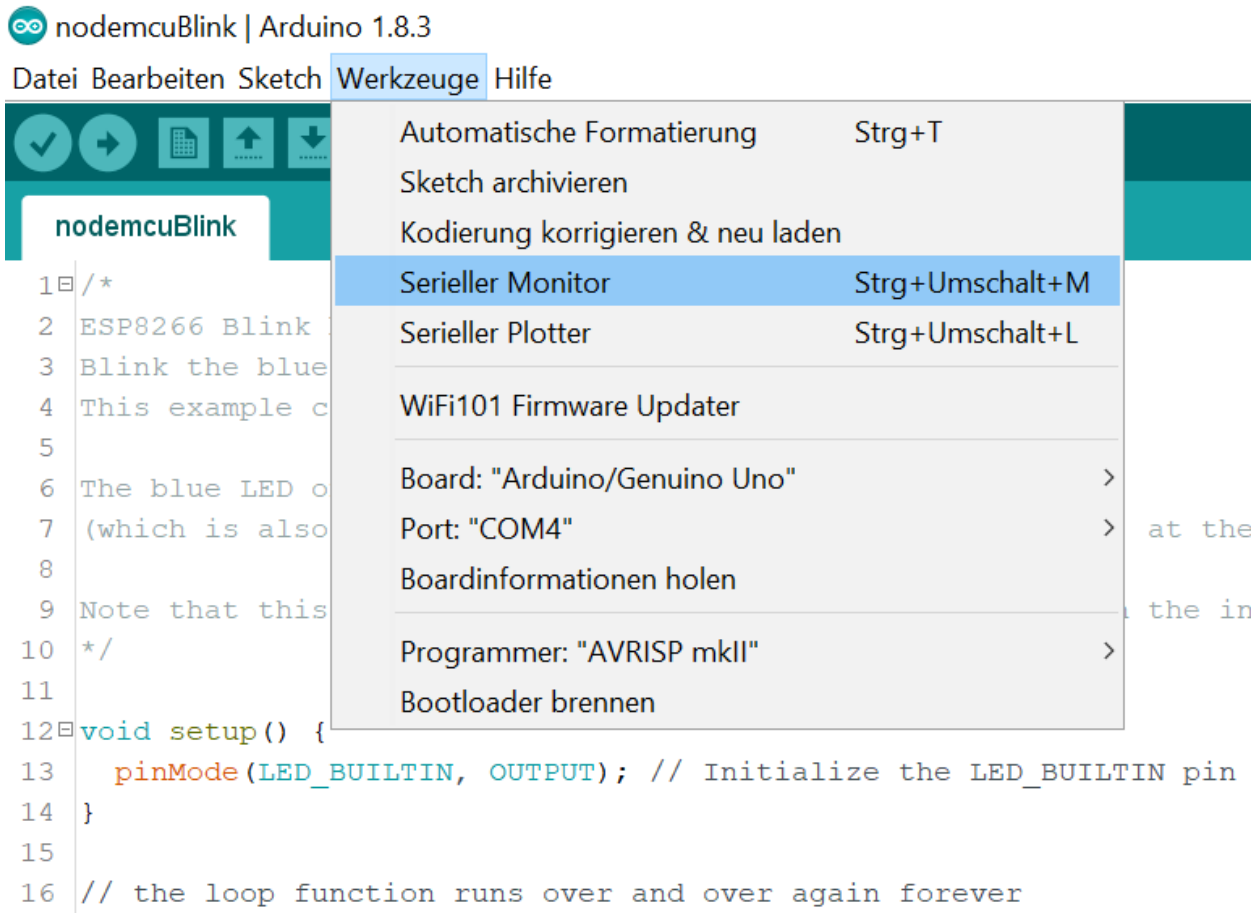
- Initialization:
  - Serial.begin(int baudrate);
- Read and write:
  - Serial.println(char[]);
  - Serial.print(char[]);
  - Serial.write(byte[]);
  - byte Serial.read();
  - boolean Serial.available();
- Close the connection:
  - Serial.end()

Send data from arduino to PC

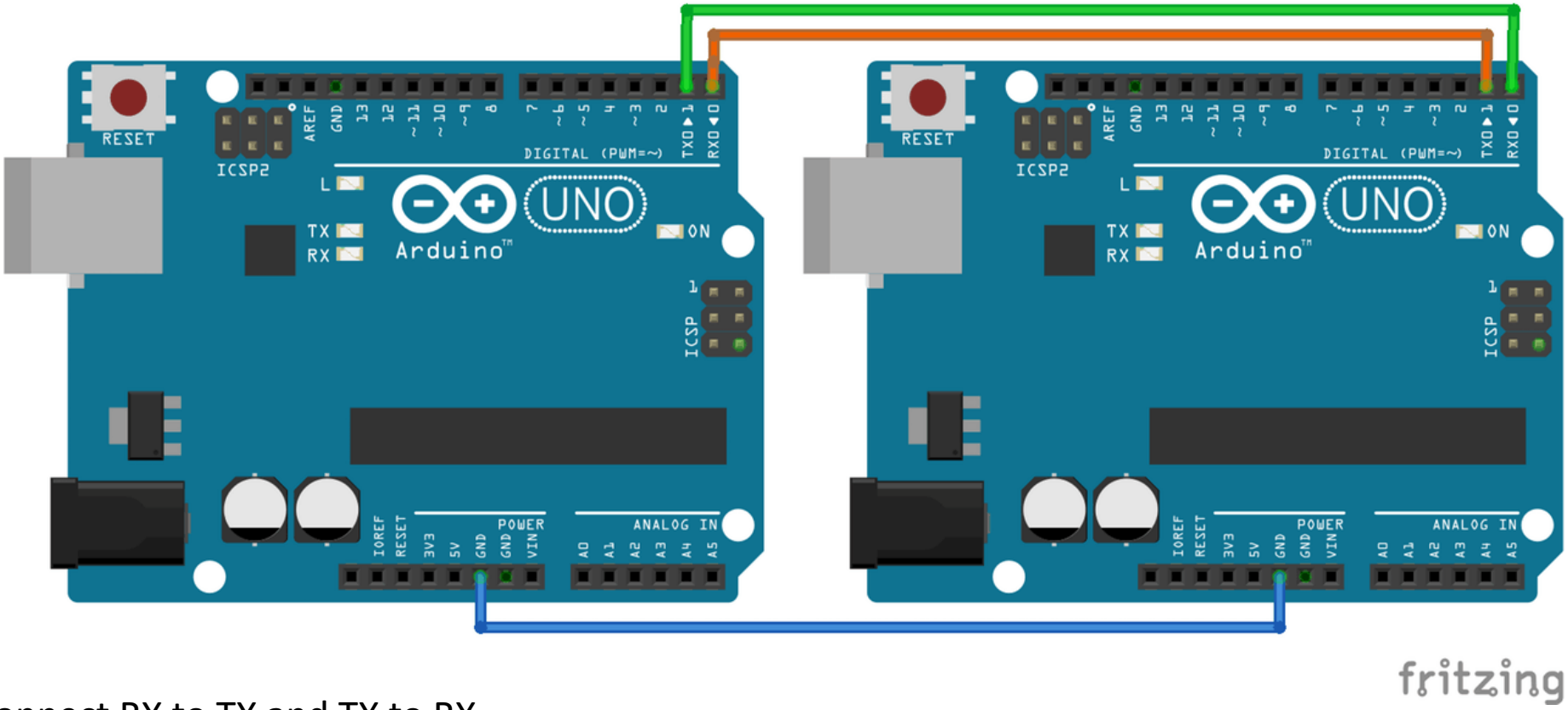
```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Hello world");
}
```

How to see data on PC?



Use UART for communication between two Arduinos



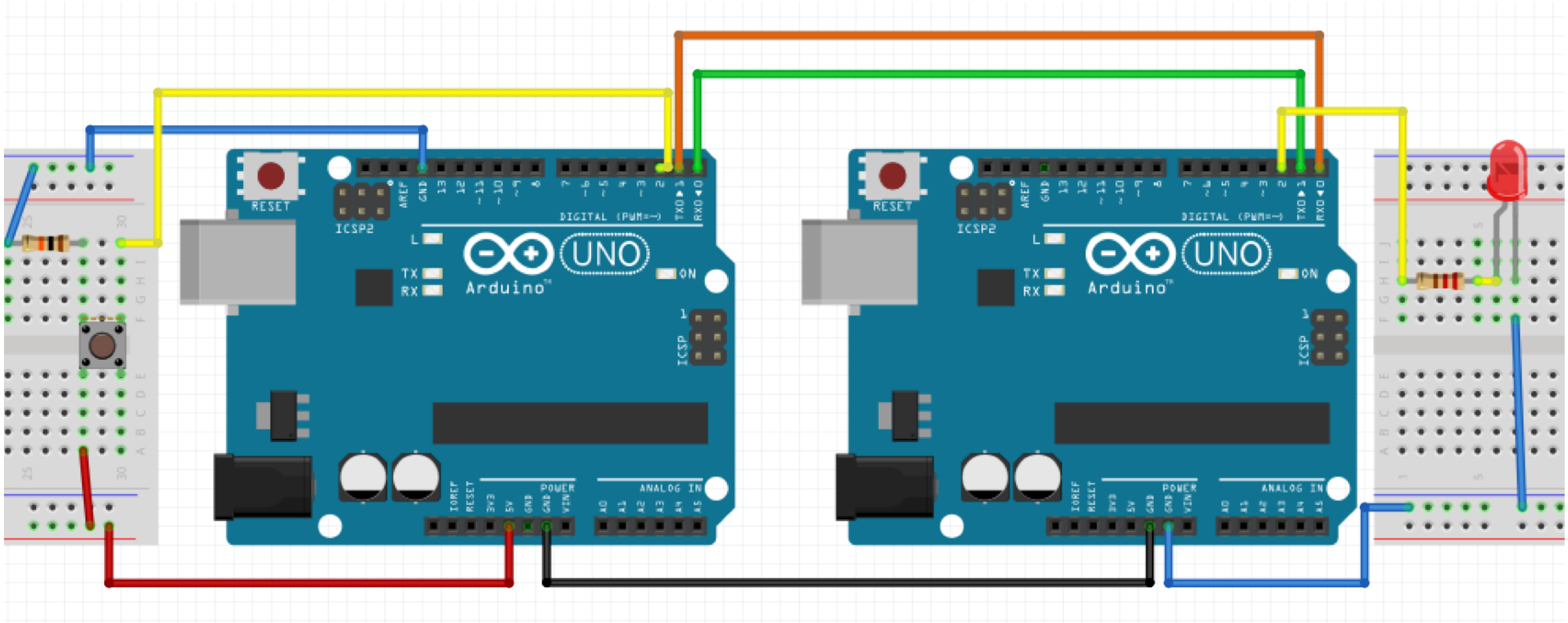
Connect RX to TX and TX to RX

Use a wire and connect GND-pins

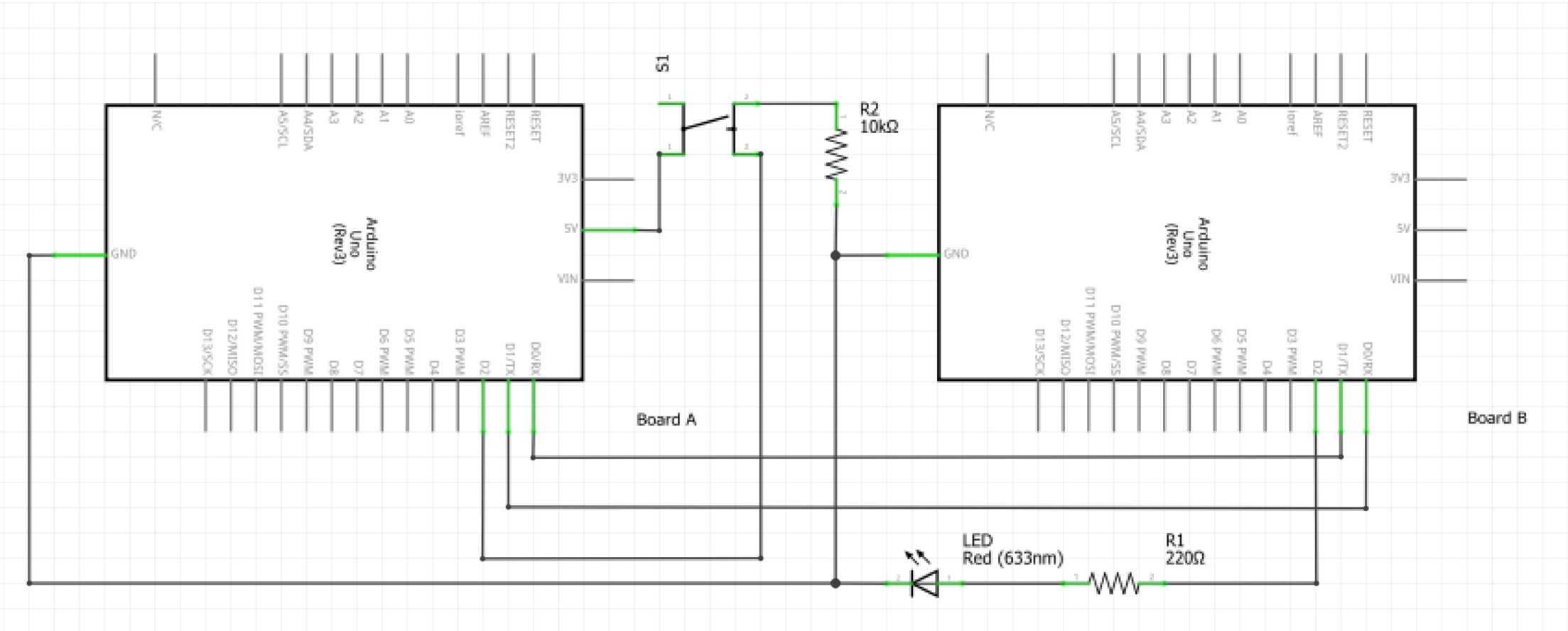
Hands on

- Goal: turn on/off a LED connected to board A by pressing a button connected to board B
  - Two groups work together
  - Use UART

Wiring the circuit



Schematic


S1
R2
10kΩ
Arduino
Uno
(Rev3)
Board A
Board B
LED
Red (633nm)
R1
220Ω

Methods to get the job done

- Methods form previous sessions about input and output

- void Serial.begin(baudrate);
  - baudrate: number of byte transmitted per second (use 9600 here)
- byte Serial.read();
  - Return: first byte recieved by RX (if data is available) as int
- int Serial.available()
  - Return: Get the number of bytes available for reading from the serial port
- byte Serial.write(value);
  - value: a value to send as a single byte

Possible solution for sender

```
int inputPin = 2;              // choose the input pin (for a pushbutton)
int buttonValue = 0; // variable for reading the pin status, HIGH=pressed, LOW=released

void setup()
{
  Serial.begin(9600);
  pinMode(inputPin, INPUT);     // declare pushbutton as input
}

void loop()
{
  buttonValue = digitalRead(inputPin);      // read input value
  Serial.write(buttonValue);
}
```

Possible solution for receiver

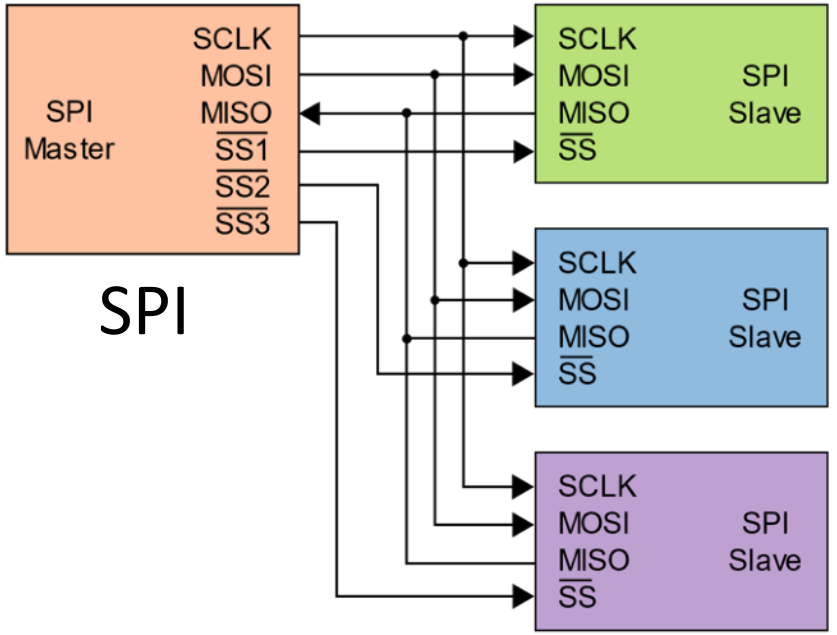
```
int ledPin = 2;                 //choose the pin for the LED
int incomingByte = 0;           // variable for reading the pin status, HIGH=pressed, LOW=released

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);   // declare pushbutton as input
}

void loop()
{
        if (Serial.available() > 0)
        {
             incomingByte = Serial.read(); // read the incoming byte
              digitalWrite(ledPin, incomingByte);
        }
}
```

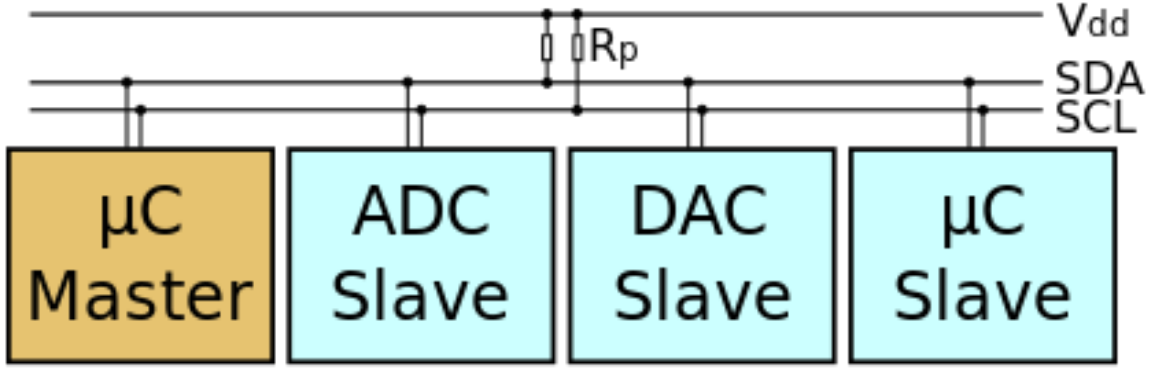
Want to connect more than two devices?

- Use a communication bus
  - I²C or SPI
- Sensors and shields are often use a bus



SPI

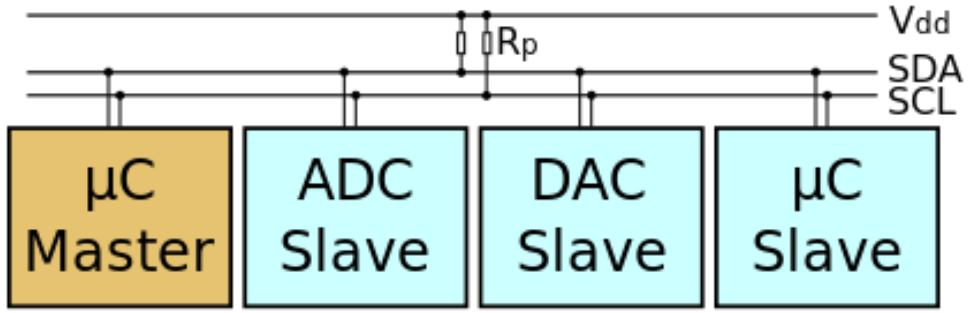
I²C



Lets have a deeper look at I²C

Inter-Integrated Circuit - I²C

- Master and slaves
  - Master generates clock
  - Slave only responses when addressed by master
  - Communication is only between master and slave, not slave to slave
- Only needs two wires
- Up to 112 nodes
- Each node has a unique address
- Use *Wire* library
- I²C uses special pins on arduino boards
  - For Arduino Uno A4 for data, A5 for clock



Master-slave communication -
Requesting data from slave

Master

(1) Initailize Master:

- Wire.begin();

(2) Request data:

- Wire.requestFrom(8, 9);

(4) Read received data:

- while (Wire.available())
  {
  byte b = Wire.read();
  }

Slave

(1) Initailize Slave:

- Wire.begin(8);
- Wire.onRequest(requestEvent);

(3) Receive request and write data:

- void requestEvent()
  {
  Wire.write("UniSiegen");
  }

Master-slave communication -
Sending data to slave

Master

(1) Initailize Master:

- Wire.begin();

(2) Sending data:

- Wire.beginTransmission(8);
  Wire.write("x");
  Wire.endTransmission();

Slave

(1) Initailize Slave:

- Wire.begin(8);
- Wire.onReceive(receiveEvent);

(3) Receive data:

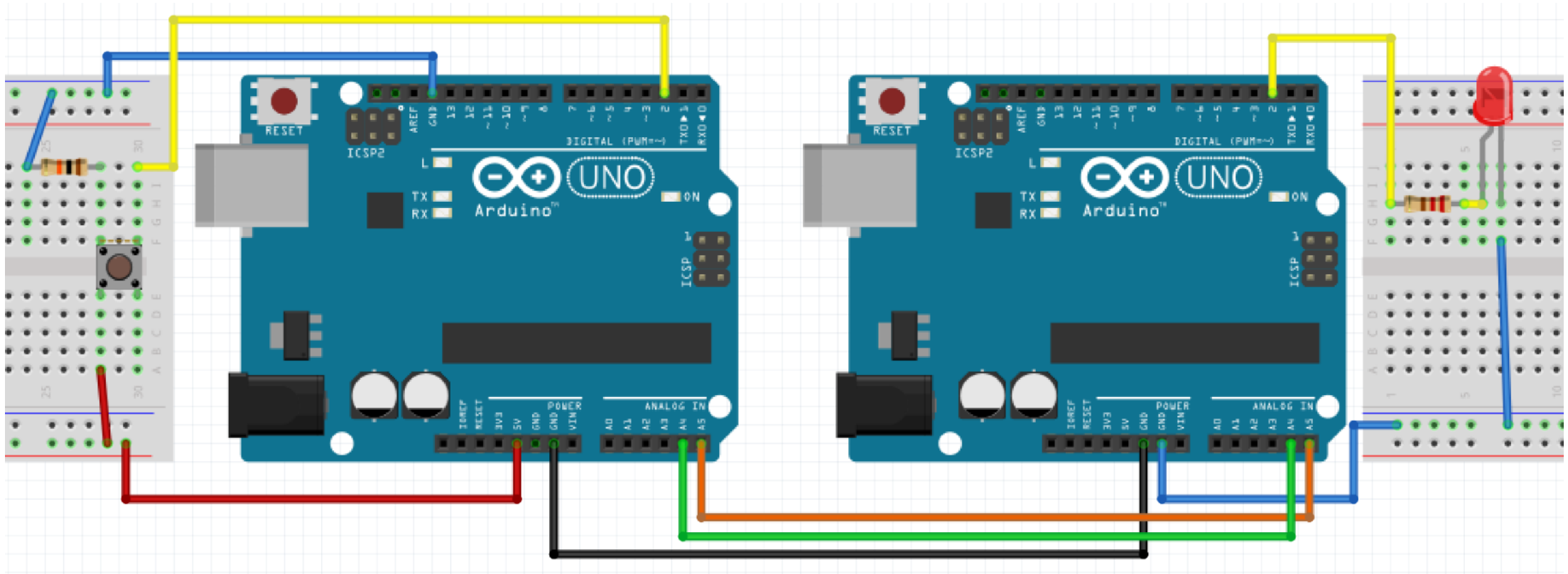
- ```
  void receiveEvent(int howMany)
  {
    while (Wire.available())
    {
      byte b = Wire.read();
      //Process data
    }
  }
  ```
  ```

Hands on

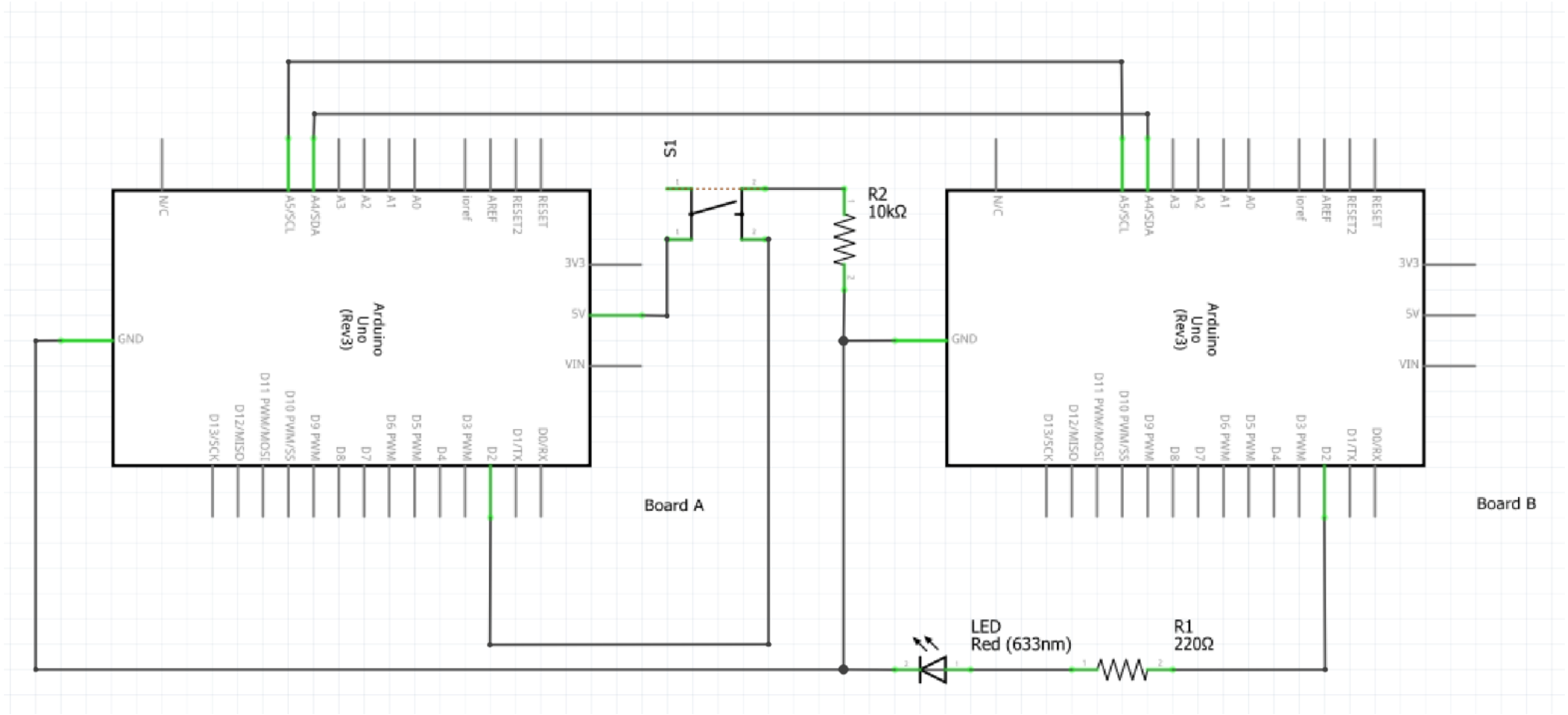
- Goal: turn on/off a LED connected to board A by pressing a button connected to board B
  - Two groups work together
  - Use I²C

- Optional: use 3 boards:
  - Board A: master (control)
  - Board B: button (input)
  - Board C: led (output)

Wiring the circuit



Schematic


S1
R2
10kΩ
Arduino
Uno
(Rev3)
Board A
Board B
LED
Red (633nm)
R1
220Ω

Methods to get the job done

- void Wire.begin(address);
  - address: keep blank for master, number < 112 for slave
- byte Wire.requestFrom();
  - Used by the master to request bytes from a slave device. The bytes may then be retrieved with the available() and read() functions.
- void Wire.onRequest(handler)
  - Register a function to be called when a master requests data from this slave device.
  - handler: the function to be called, takes no parameters and returns nothing
- byte Wire.read();
  - Return: The next byte received
- byte Wire.write();
  - Writes data from a slave device in response to a request from a master, or queues bytes for transmission from a master to slave device (in-between calls to beginTransmission() and endTransmission())
- void Wire.beginTransmission(address);
  - Begin a transmission to the I2C slave device with the given address.
  - Address: address of slave
- byte Wire.endTransmission();
  - Ends a transmission to a slave device that was begun by beginTransmission() and transmits the bytes that were queued by write().
  - Return: byte, which indicates the status of the transmission

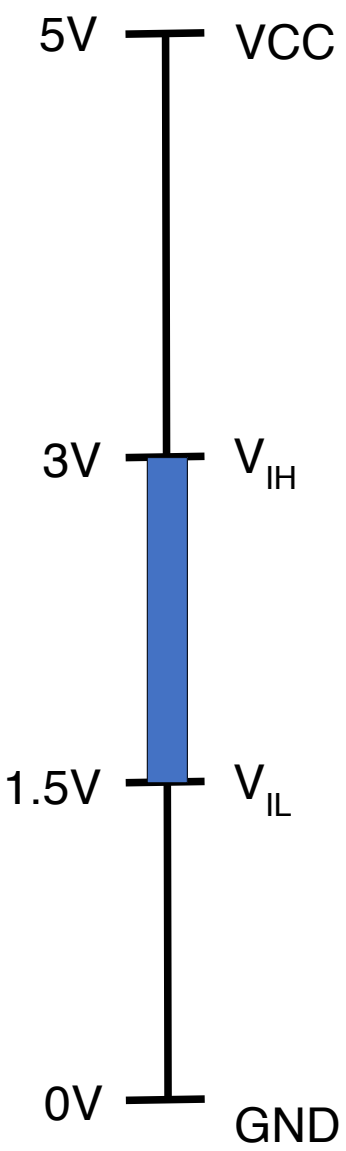
Special Aspects of HCI: Prototyping with Arduino

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

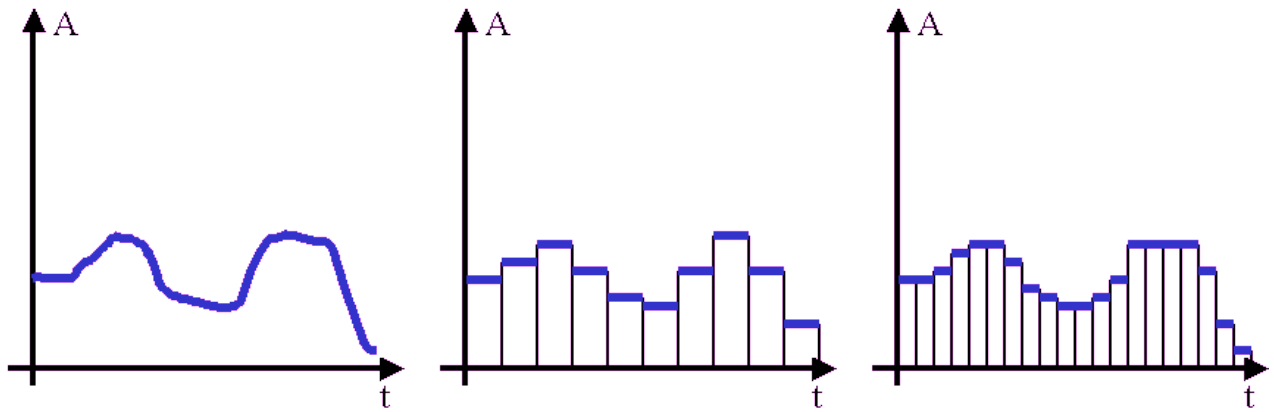


Analog signals

- 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

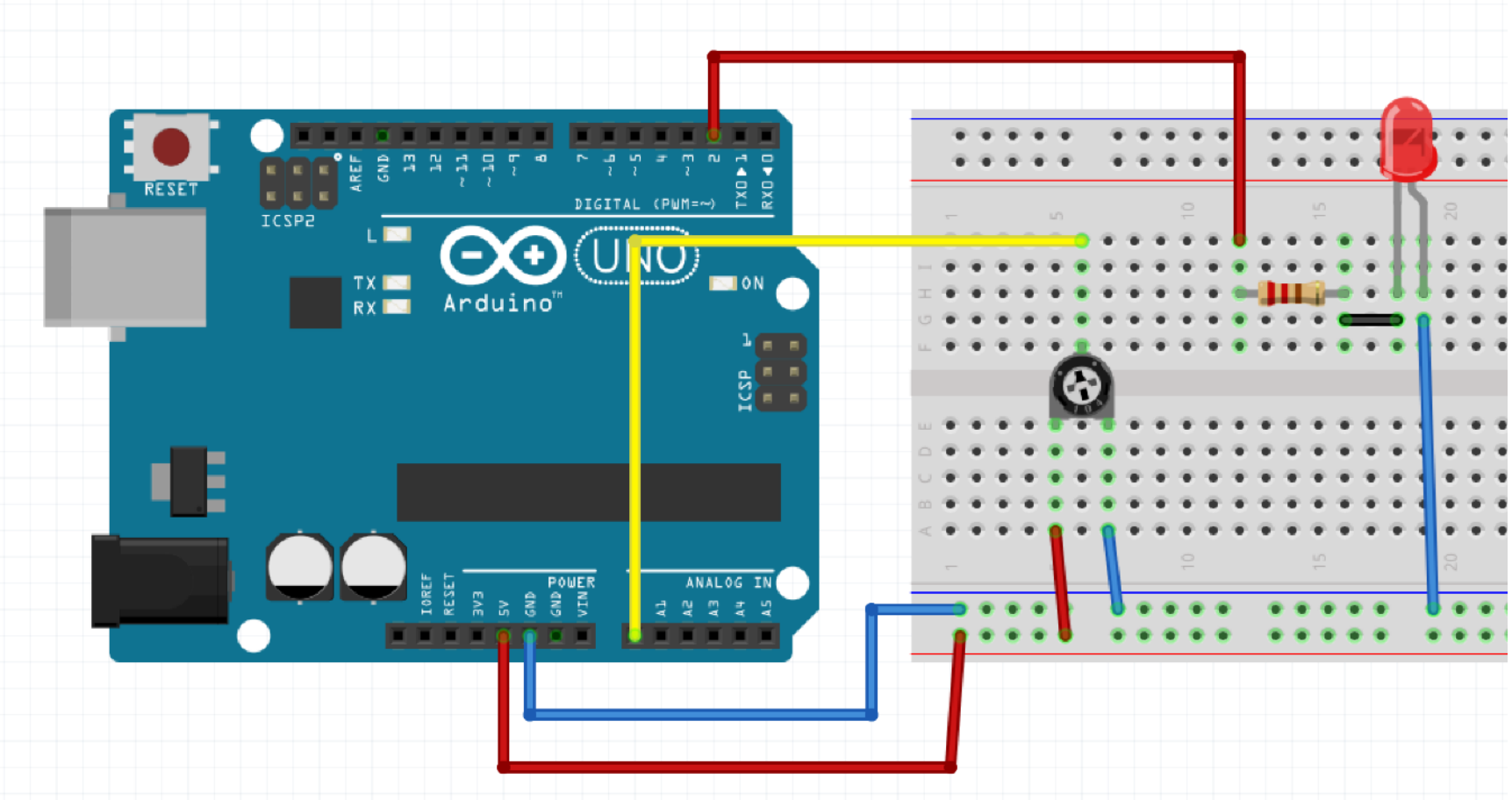
Analog inputs

- 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);

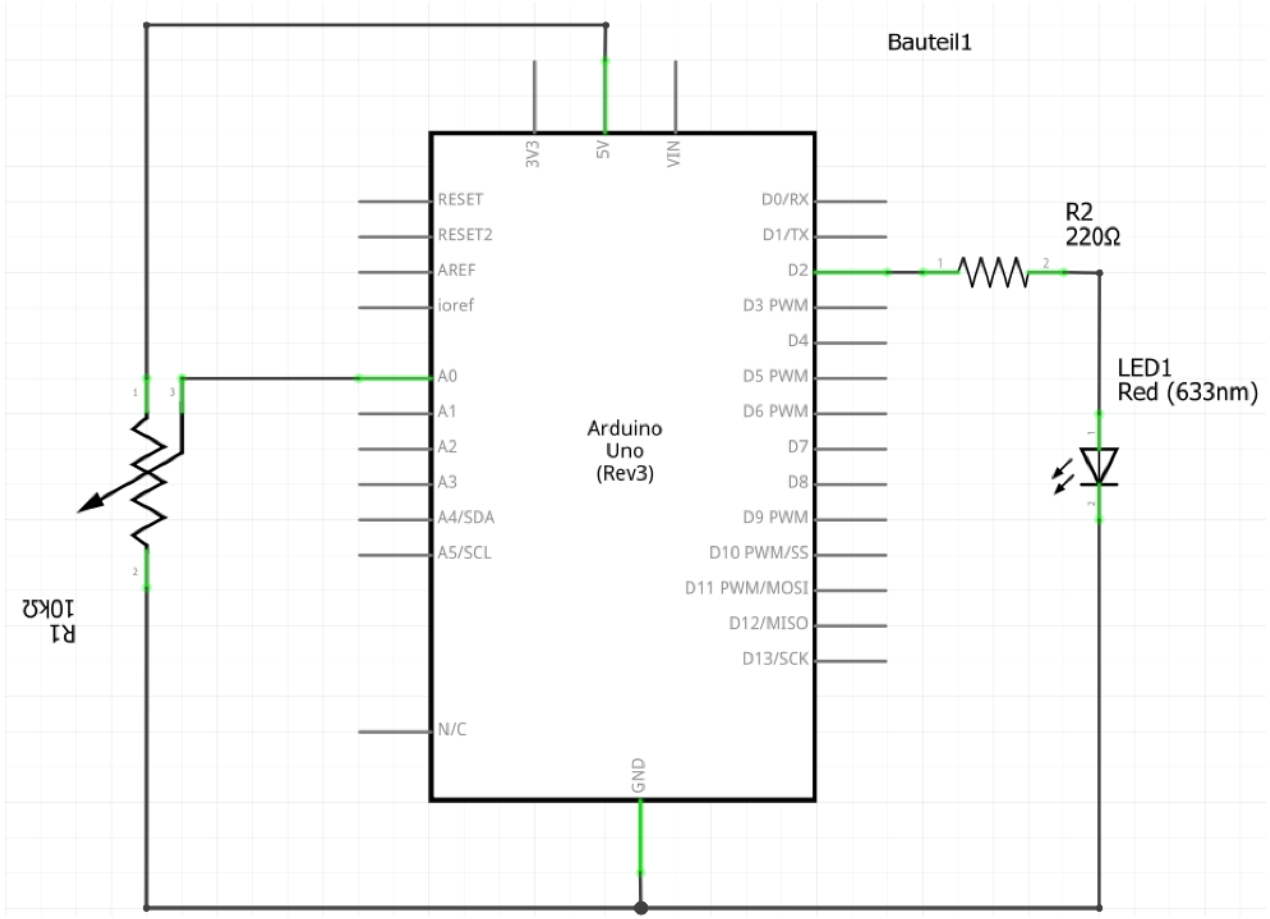
Hands on

- 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;
   }

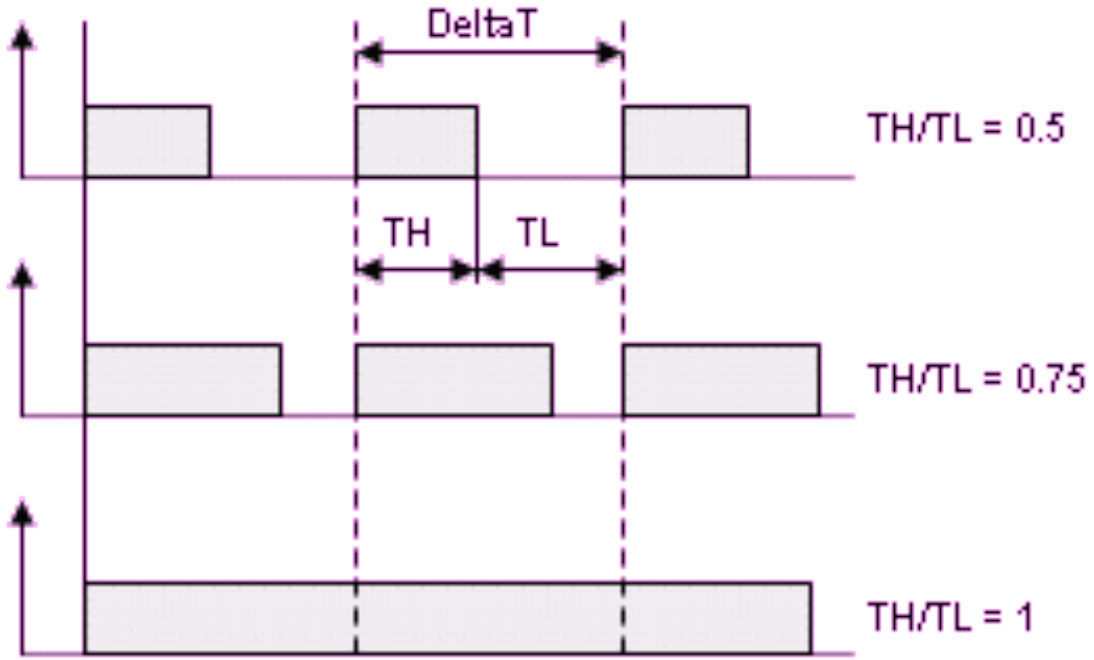
   if((millis() - lastToggle) >= waitingTime && waitingTime > 0)
   {
      ledState = !ledState;                    // toggle ledState
      digitalWrite(ledPin, ledState);
     lastToggle = millis();
   }
}
```

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)

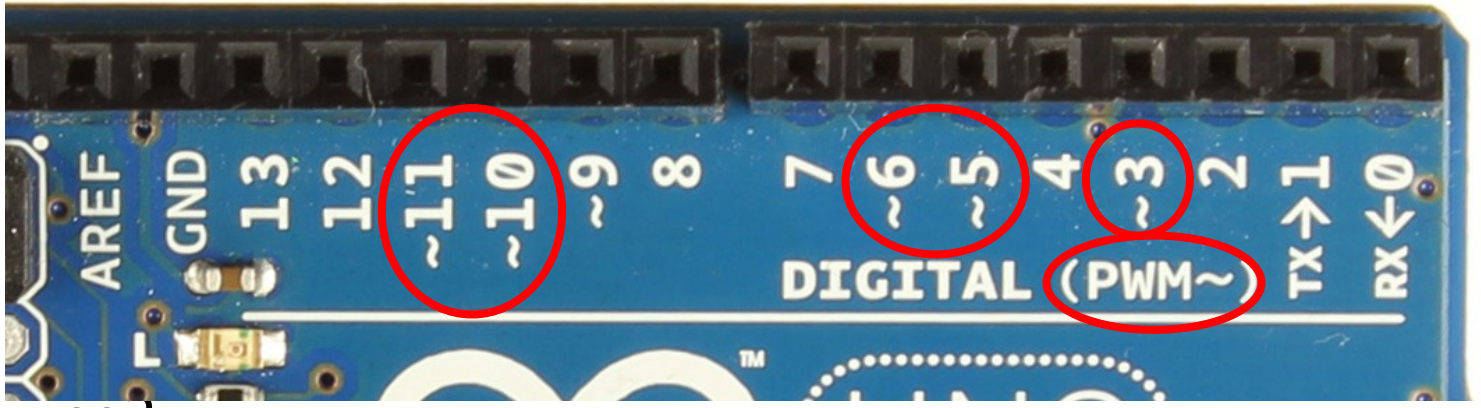
Pulse-Width-Modulation

- 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?



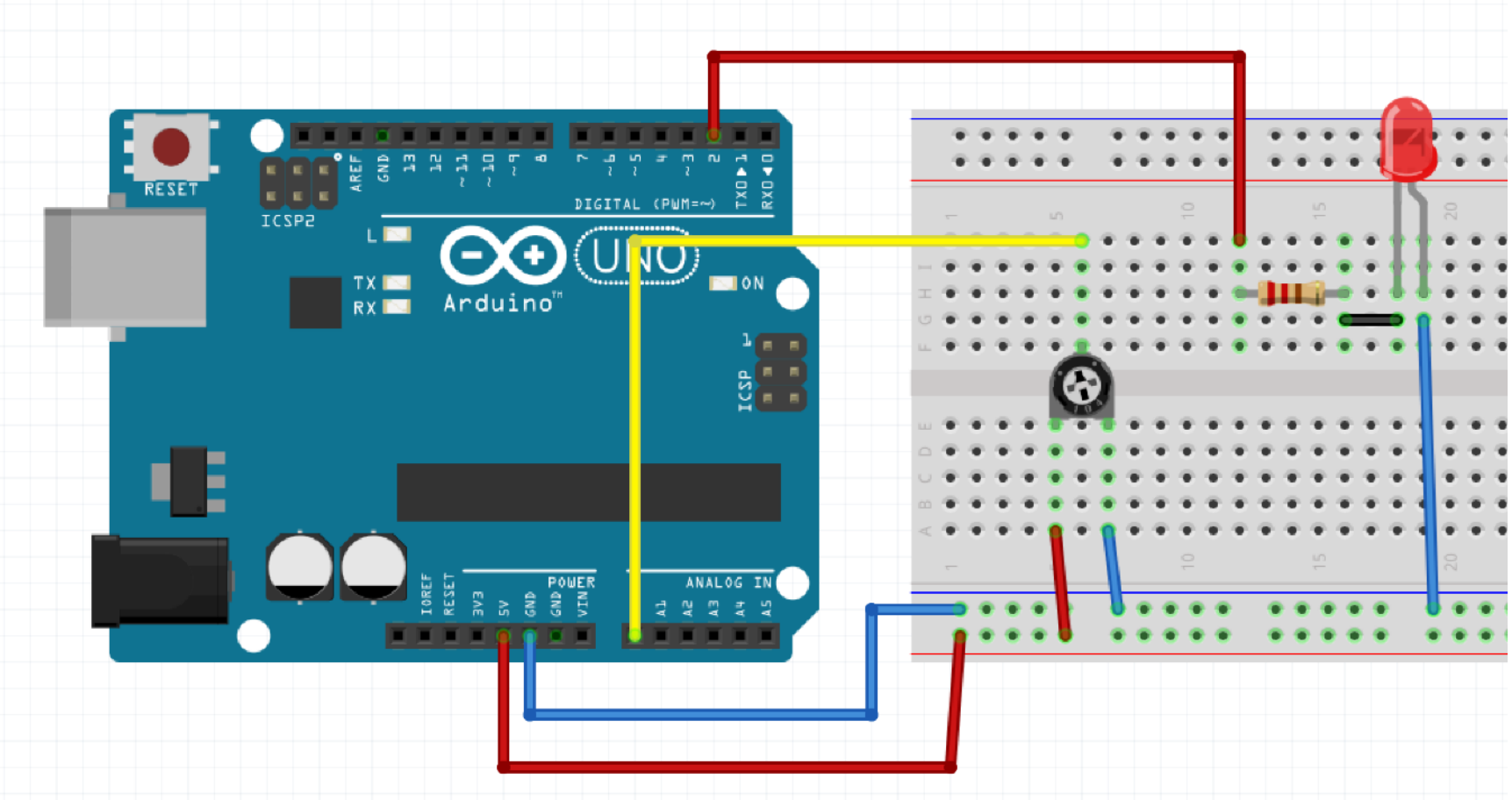
- How to use?
  - Initialize the pin as output: `pinMode(pwmPin, OUTPUT);`
  - Write analog value to pin: `analogWrite(pwmPin, value);`
- Use for what?
  - E.g. to dim LED by turning it rapidly on and off again

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!

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

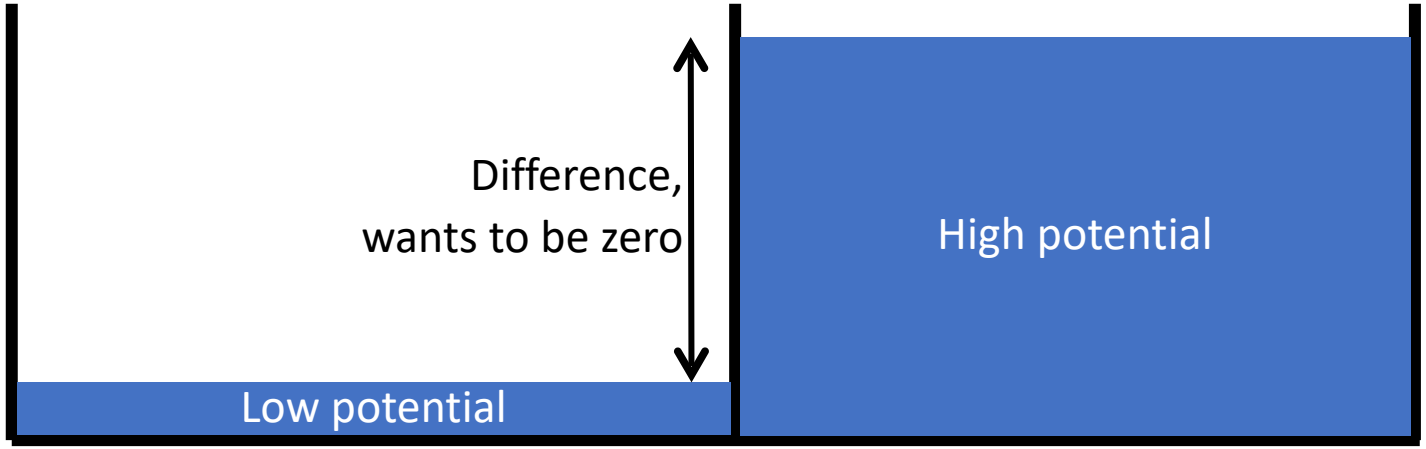
Refreshing the basics

- We keep it simple
- No scientific claim

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

Voltage

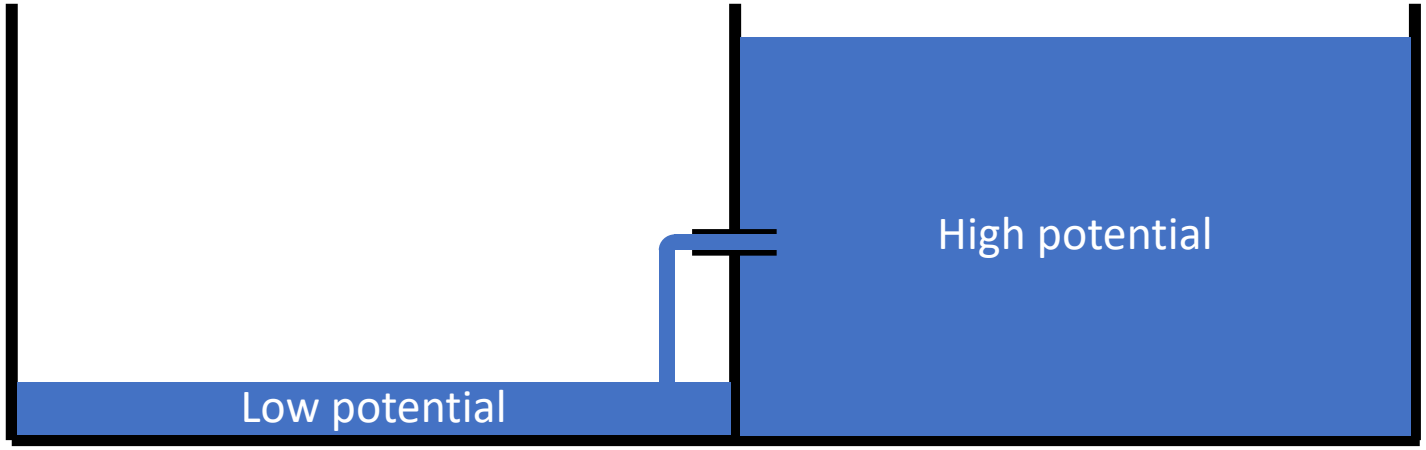
- Symbol: U
- Unit: V (Volt)
- is the difference in electric potential between two points
- High difference = high voltage



Water analogy

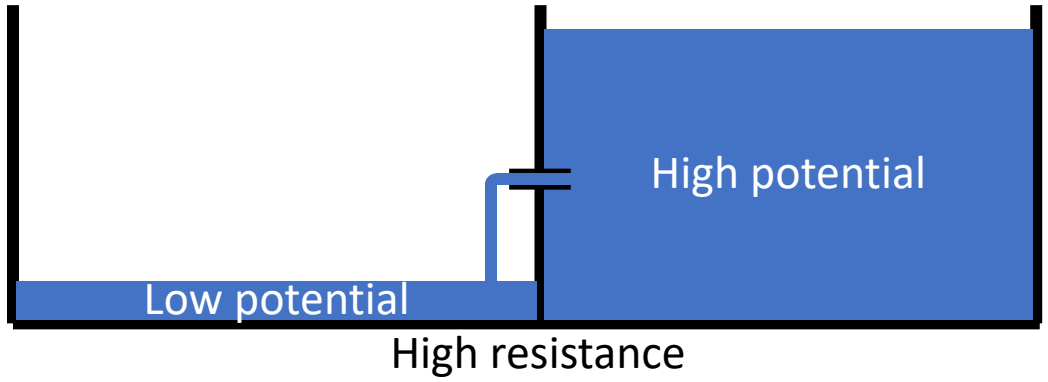
Electrical current

- 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

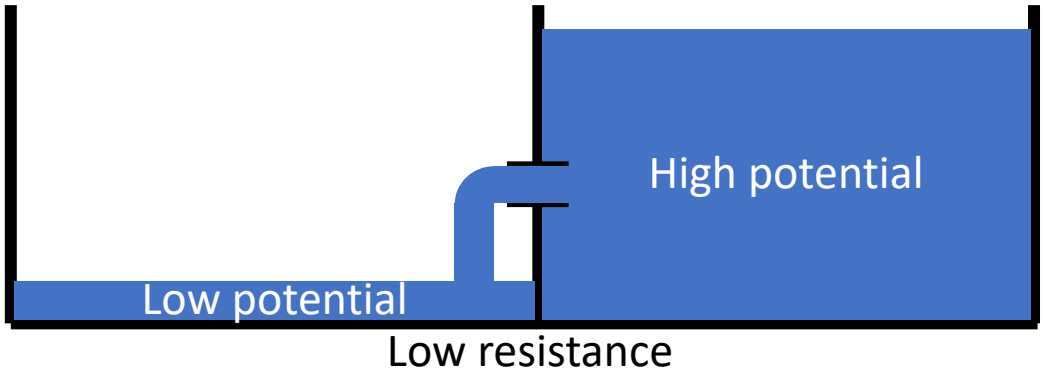


Electrical resistance

- 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

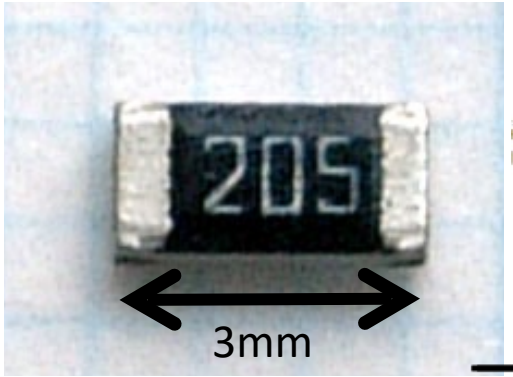


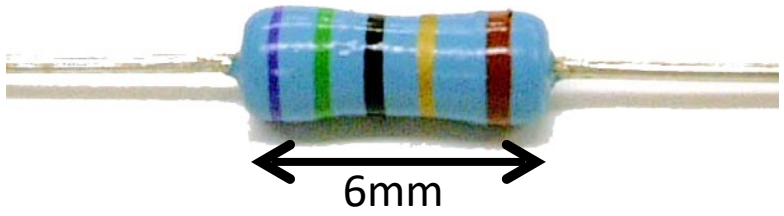
High resistance



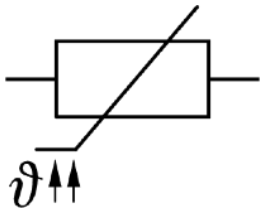
Low resistance

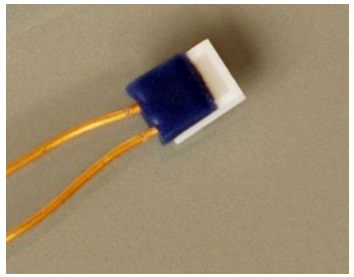
Resistor

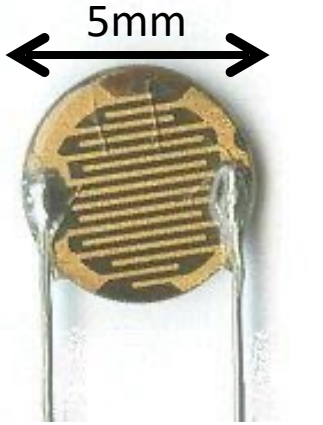


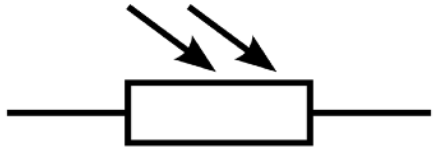


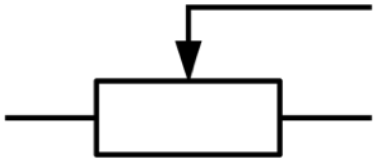
- Fixed resistance
- Manually changeable
- Resistance depends on other physical parameters (like light or temperature)











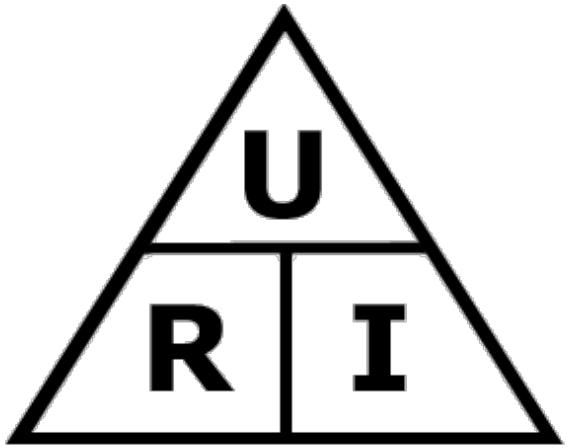
Ohm's law

- How voltage, current and resistance interact?

$$U = R \bullet I$$

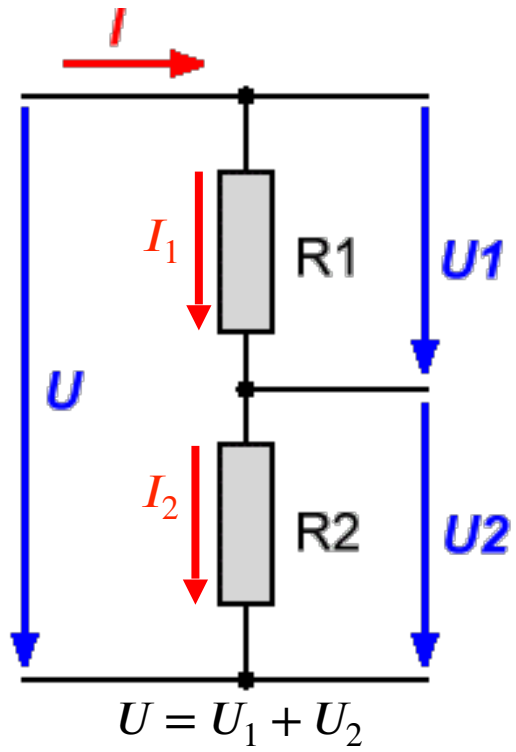
$$I = \frac{U}{R}$$

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



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
  - $I = I_1 = I_2$



$$\frac{U_1}{U_0} = \frac{R_1}{R_1 + R_2} => U_1 = U_0 \bullet \frac{R_1}{R_1 + R_2}$$

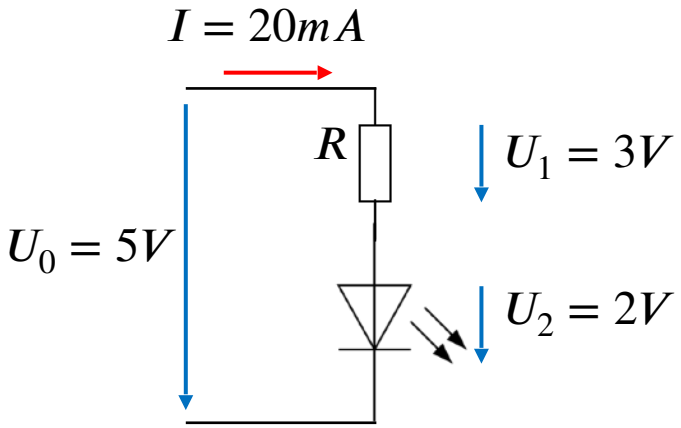
$$\frac{U_0}{U_2} = \frac{R_1 + R_2}{R_2} => U_2 = U_0 \bullet \frac{R_2}{R_1 + R_2}$$

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

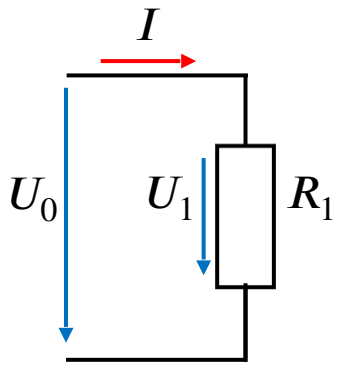
- Example:
  - LED can handle 2 – 2.5V (depending on type, see datasheet)
  - LED need around 20mA to light up (depending on type, see datasheet)
  - Arduino supplies 5V
  - 2.5 – 3V too much, needs to be compensated by resistor



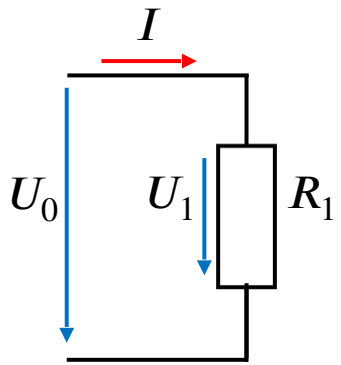
$$R = \frac{U_1}{I} = \frac{3V}{20 \bullet 10^{-3}A} = 150\ \Omega$$

Voltage divider for input

- What is the difference between this circuits?



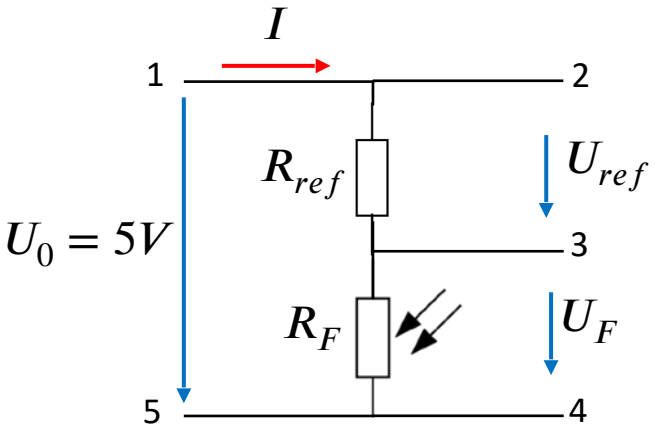
$U_0 = 5V$
$R_1 = 100\Omega$
$U_1 = \ ?$
$I = \ ?$



$U_0 = 5V$
$R_1 = 200\Omega$
$U_1 = \ ?$
$I = \ ?$

- An Arduino cant measure current directly, only voltage

Voltage divider for photoresistor (analog input)



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

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

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

$2k\Omega$ at brightness

$11k\Omega$ at darkness

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

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

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

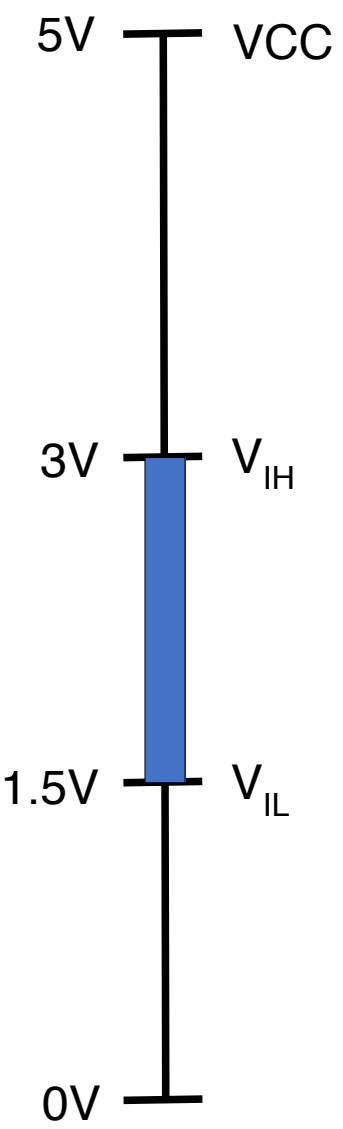
$$U_{ref\,R_{max}} = U_0 - U_{F\,R_{max}} = 1{,}5V$$

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?

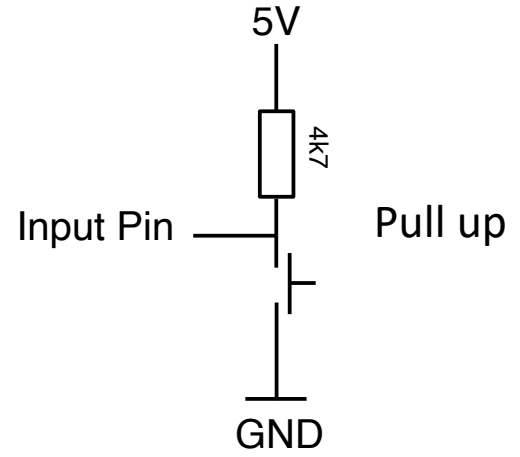
Digital inputs

- 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

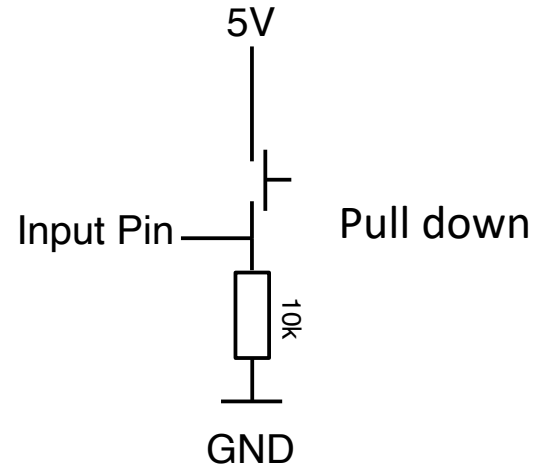


Pull up / pull down resistor

- 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



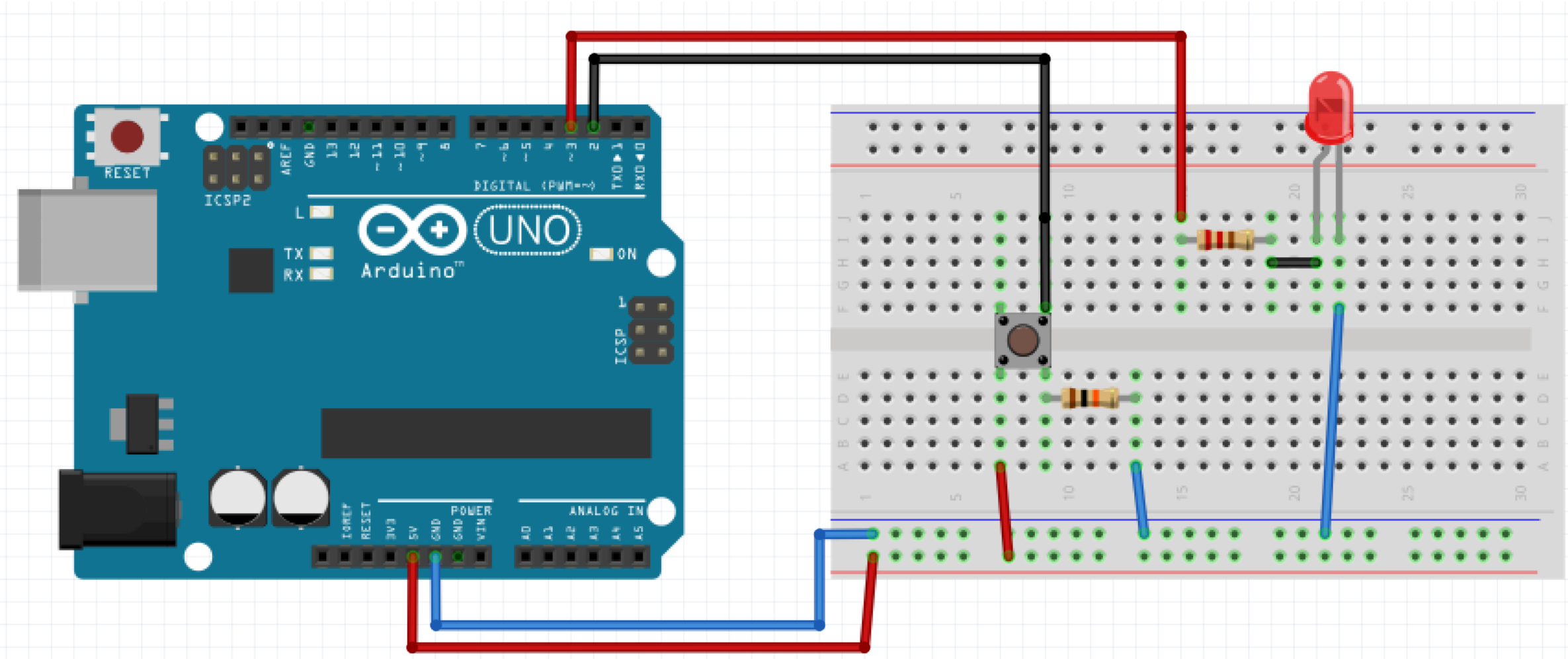
- Arduinos have a built in pull up
  - The built in pull up can be used by configuring a digital pin with pinMode(pin_number, INPUT_PULLUP)

Hands on!

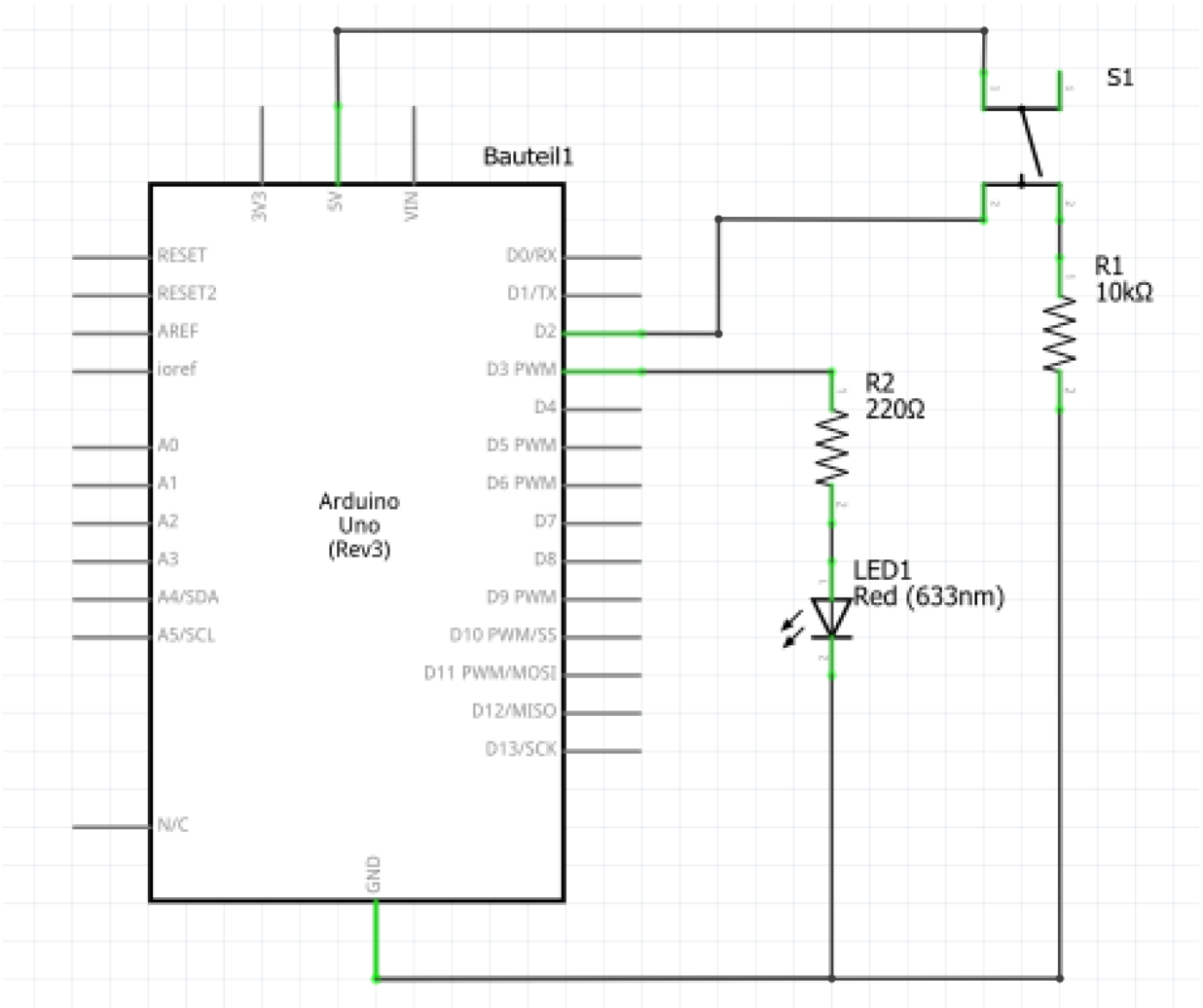
- Goal: control a LED with a button
  1. LED is on when the button is pressed
  2. LED is 5 seconds on after the button is pressed, doesn't matter how long it is pressed
  3. LED toggles each time you press the button, not on release

- 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


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)
S1
R1
10kΩ
R2
220Ω
LED1
Red (633nm)

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=pressed, LOW=released

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=pressed, 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)

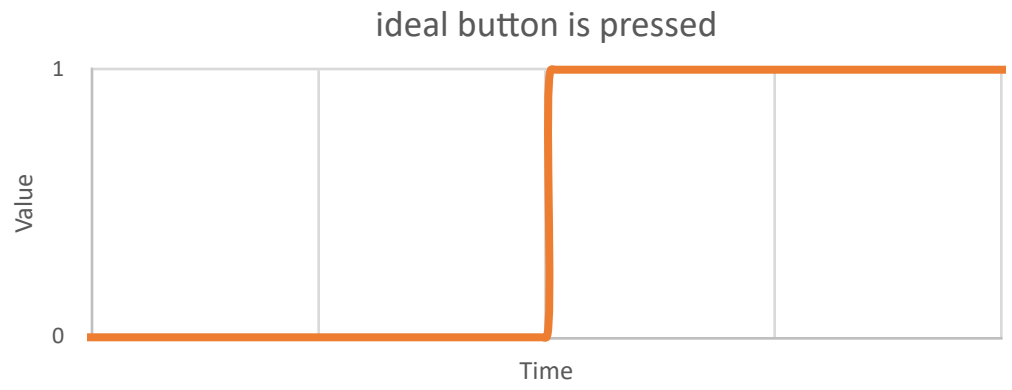
```
int ledPin = 3;              // choose the pin for the LED
int inputPin = 2;            // choose the input pin (for a pushbutton)
int buttonValue = 0;    // variable for reading the pin status, HIGH=pressed, LOW=released
int previousButtonValue = 0;
int ledState = 0;         // variable for storing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

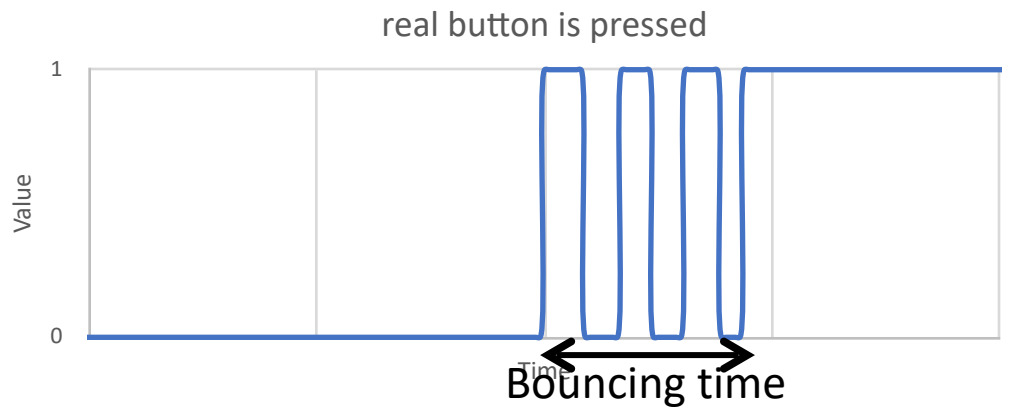
void setup()
{
  pinMode(ledPin, OUTPUT);  // declare LED as output
  pinMode(inputPin, INPUT);   // declare pushbutton as input
}

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 "de-bounced".
- 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=pressed, 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;
  }
}
```