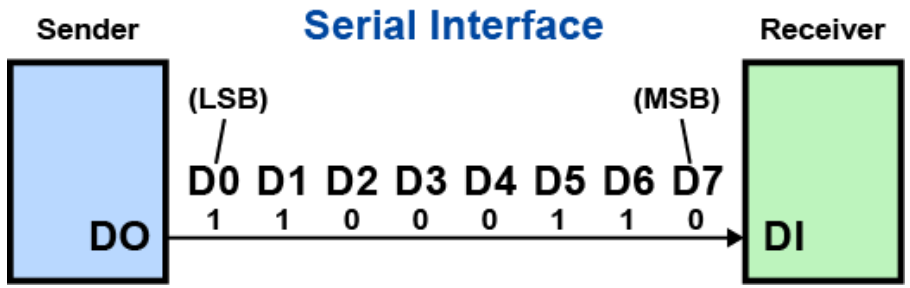
Special Aspects of HCI: Prototyping with Arduino

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

Today: communication

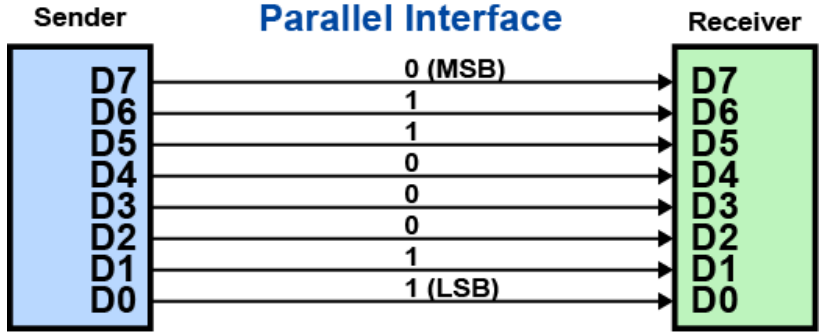
Types of communication

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



Example transfers of 01100011

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



Universal **A**synchronous **R**eceiver **T**ransmitter (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

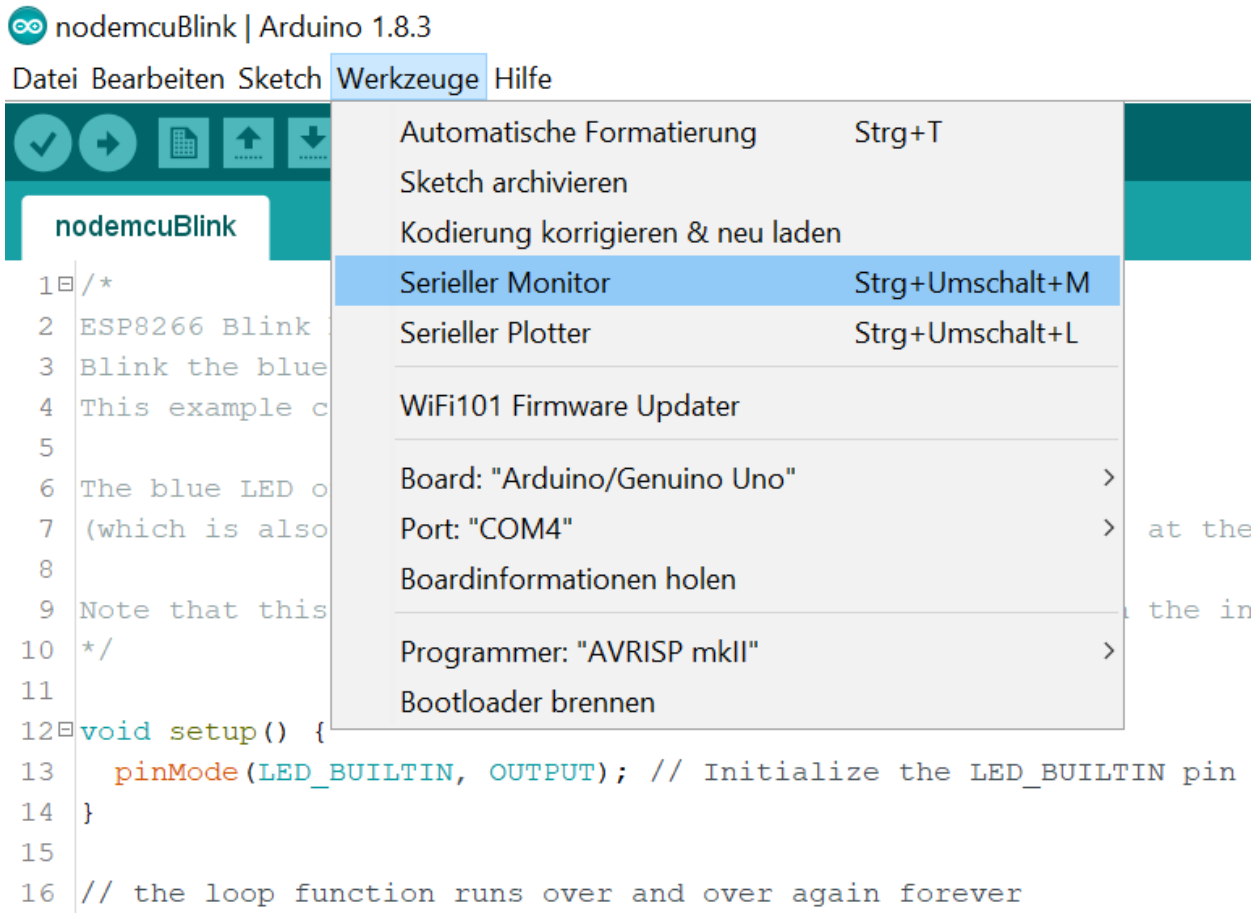
- Initialization:
  - Serial.begin(int baudrate);
- Read and write:
  - Serial.println(char[]);
  - Serial.print(char[]);
  - Serial.write(byte[]);
  - byte Serial.read();
  - boolean Serial.available();
- Close the connection:
  - Serial.end()

Send data from arduino to PC

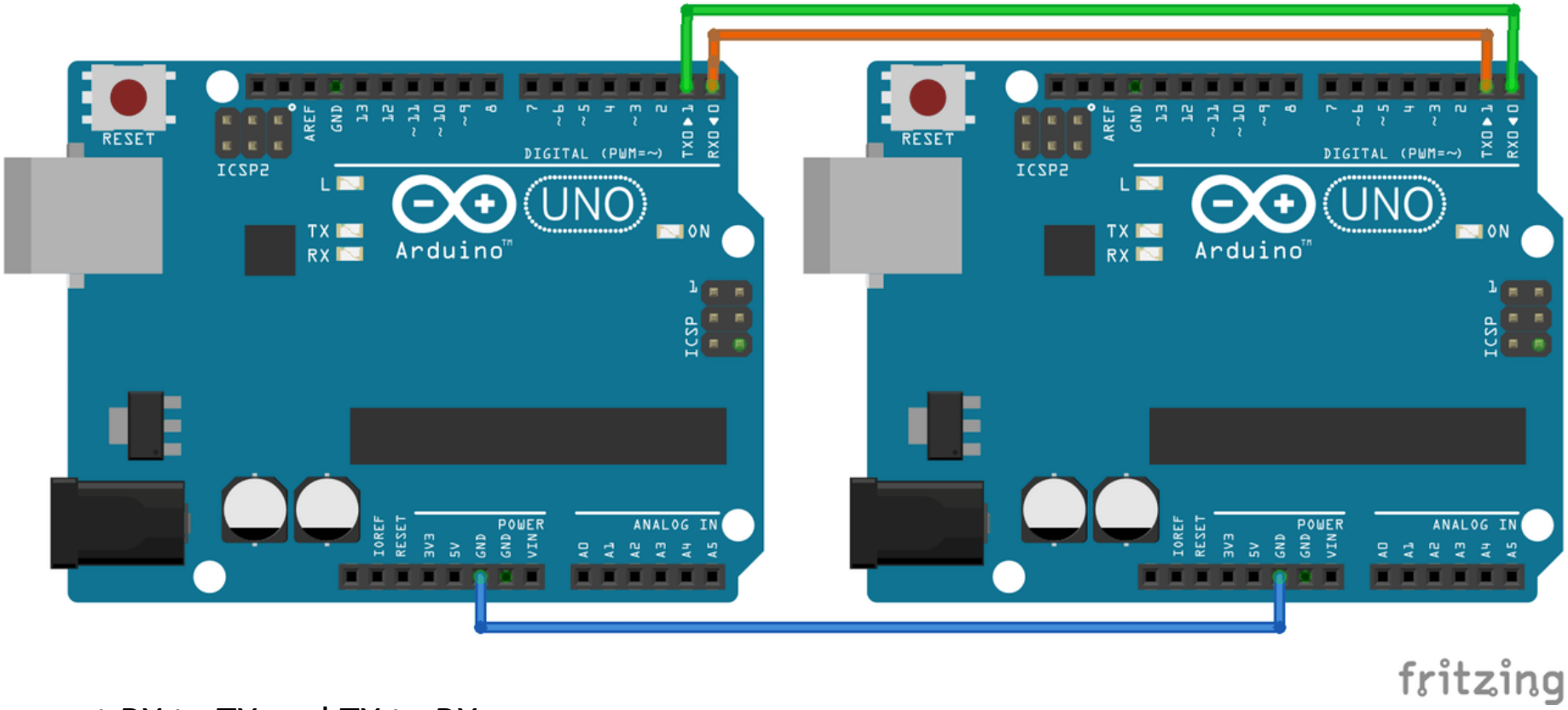
```
void setup()
{
  Serial.begin(9600);
}

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

How to see data on PC?



Use UART for communication between two Arduinos

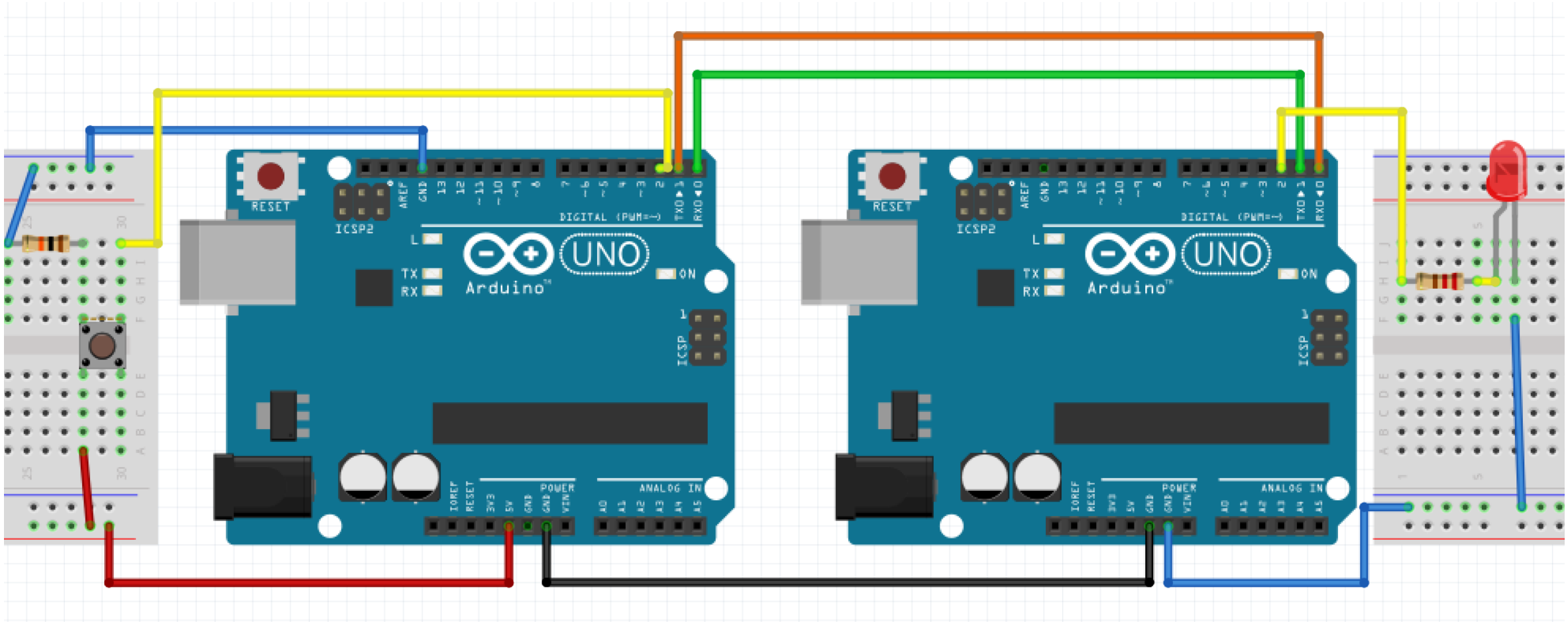


Connect RX to TX and TX to RX
Use a wire and connect GND-pins

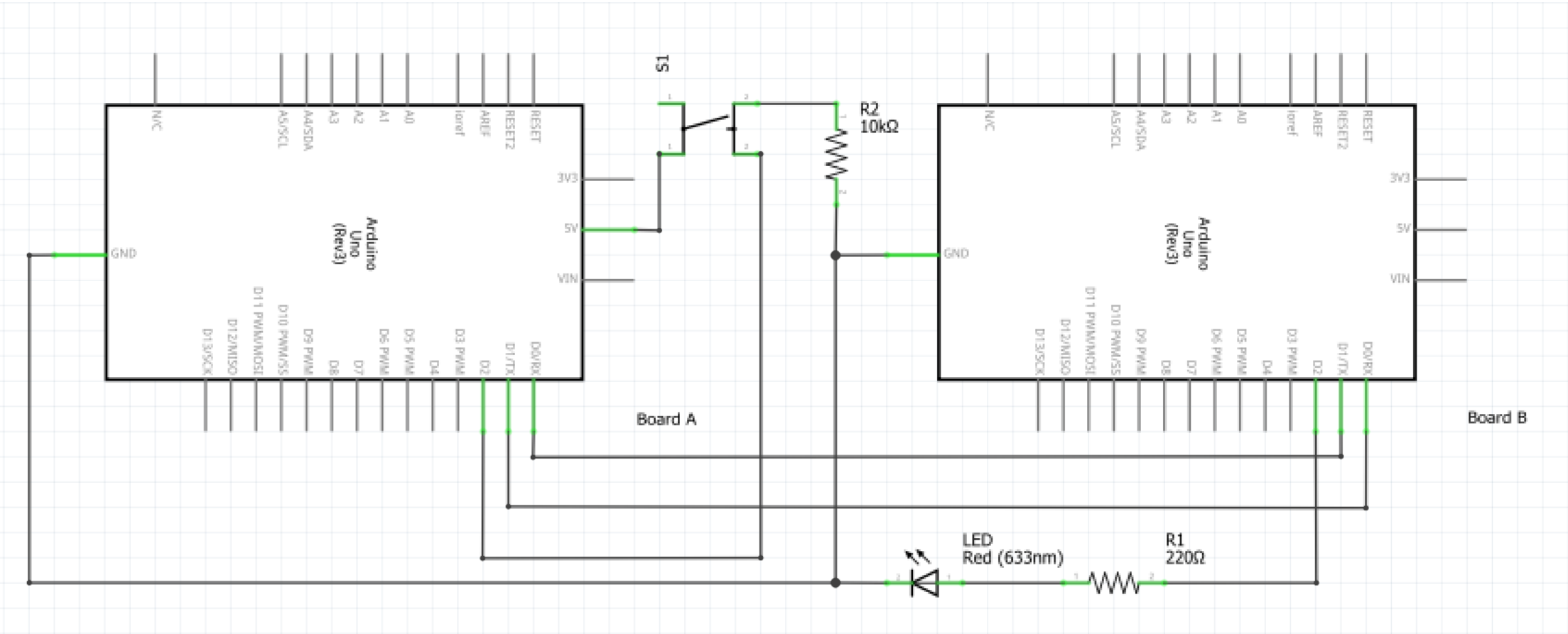
Hands on

- 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

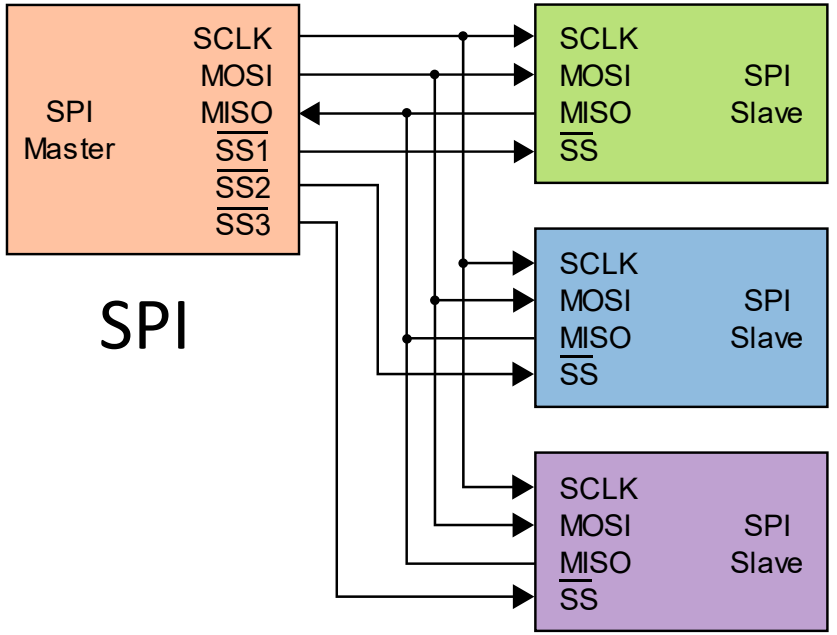
```
int ledPin = 2;                 //choose the pin for the LED
int incomingByte = 0;           // variable for reading the pin status, HIGH=pressed, LOW=released

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);
        }
}
```

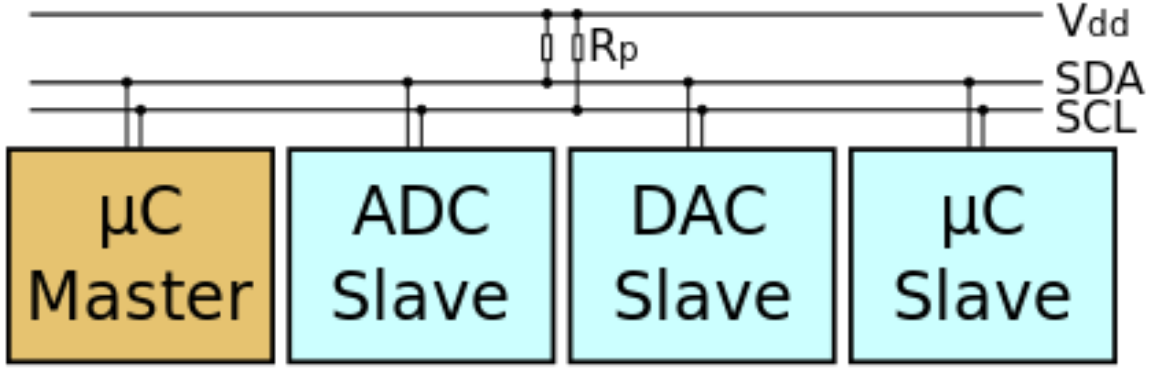
Want to connect more than two devices?

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



SPI

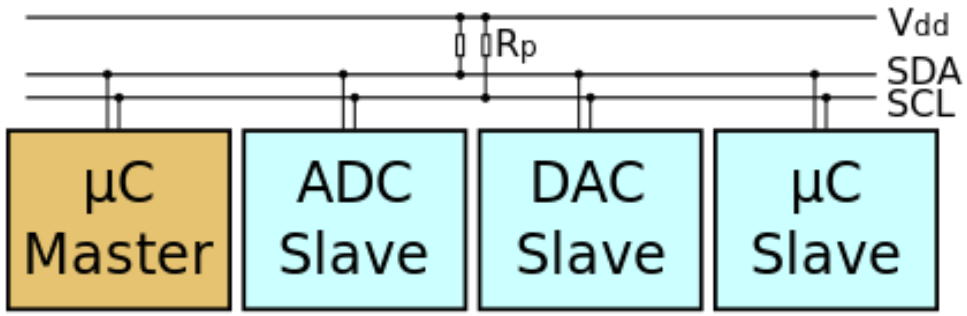
I²C



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);

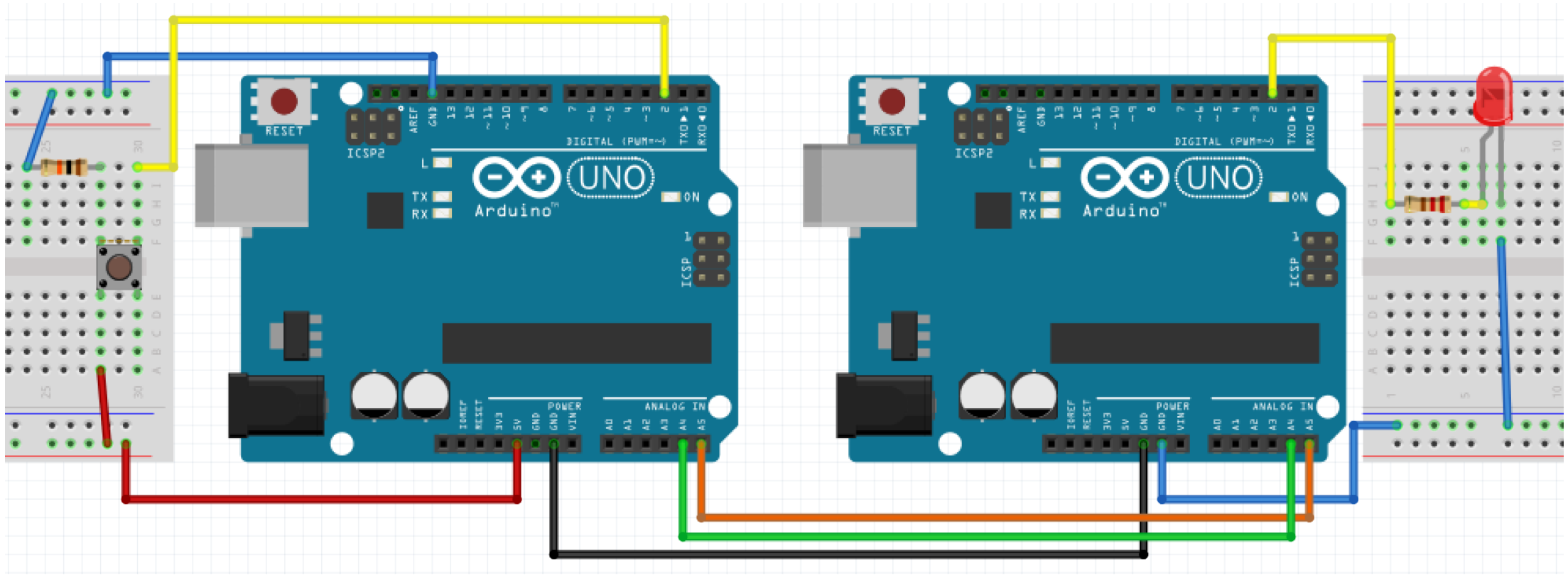
(3) Receive data:

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

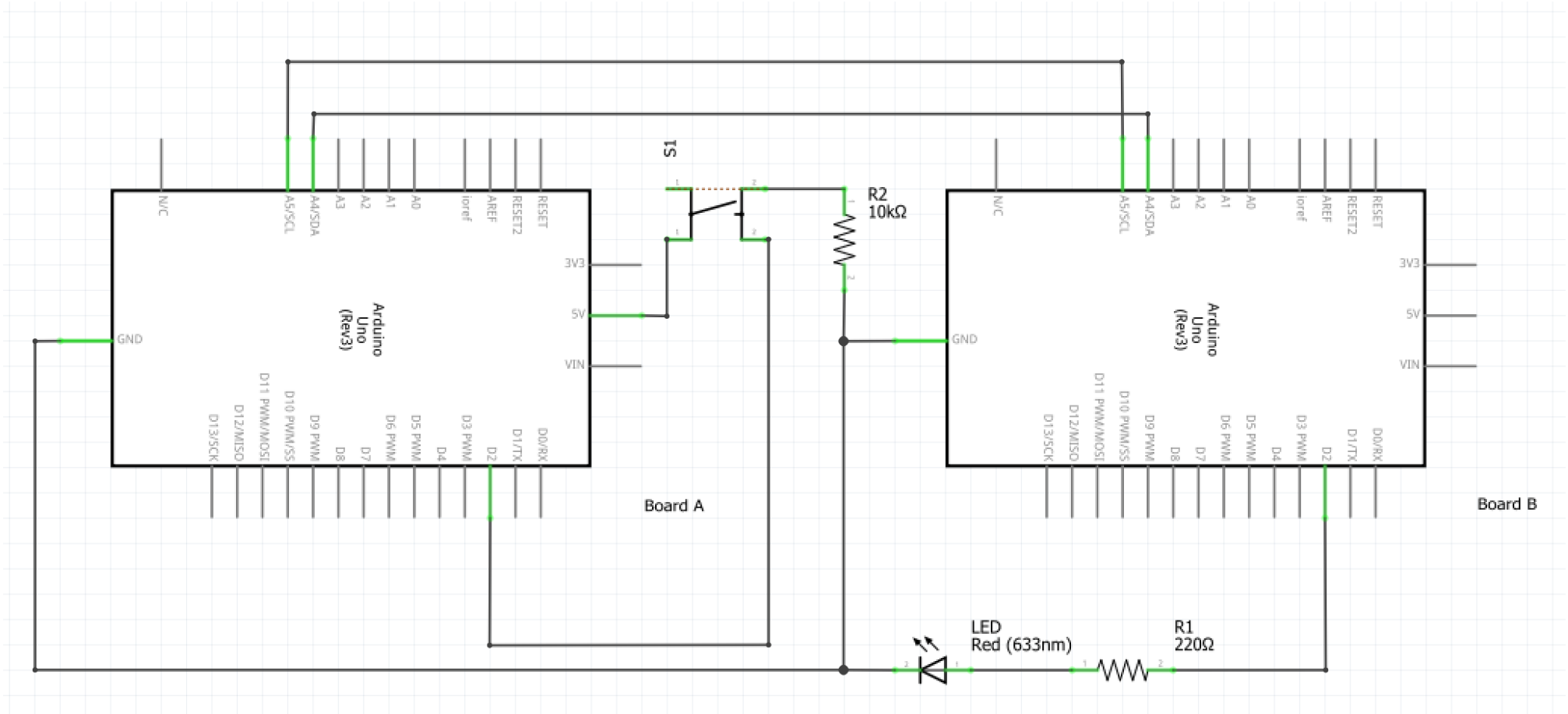
Hands on

- Goal: turn on/off a LED connected to board A by pressing a button connected to board B
  - Two groups work together
  - Use I^2C
- 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