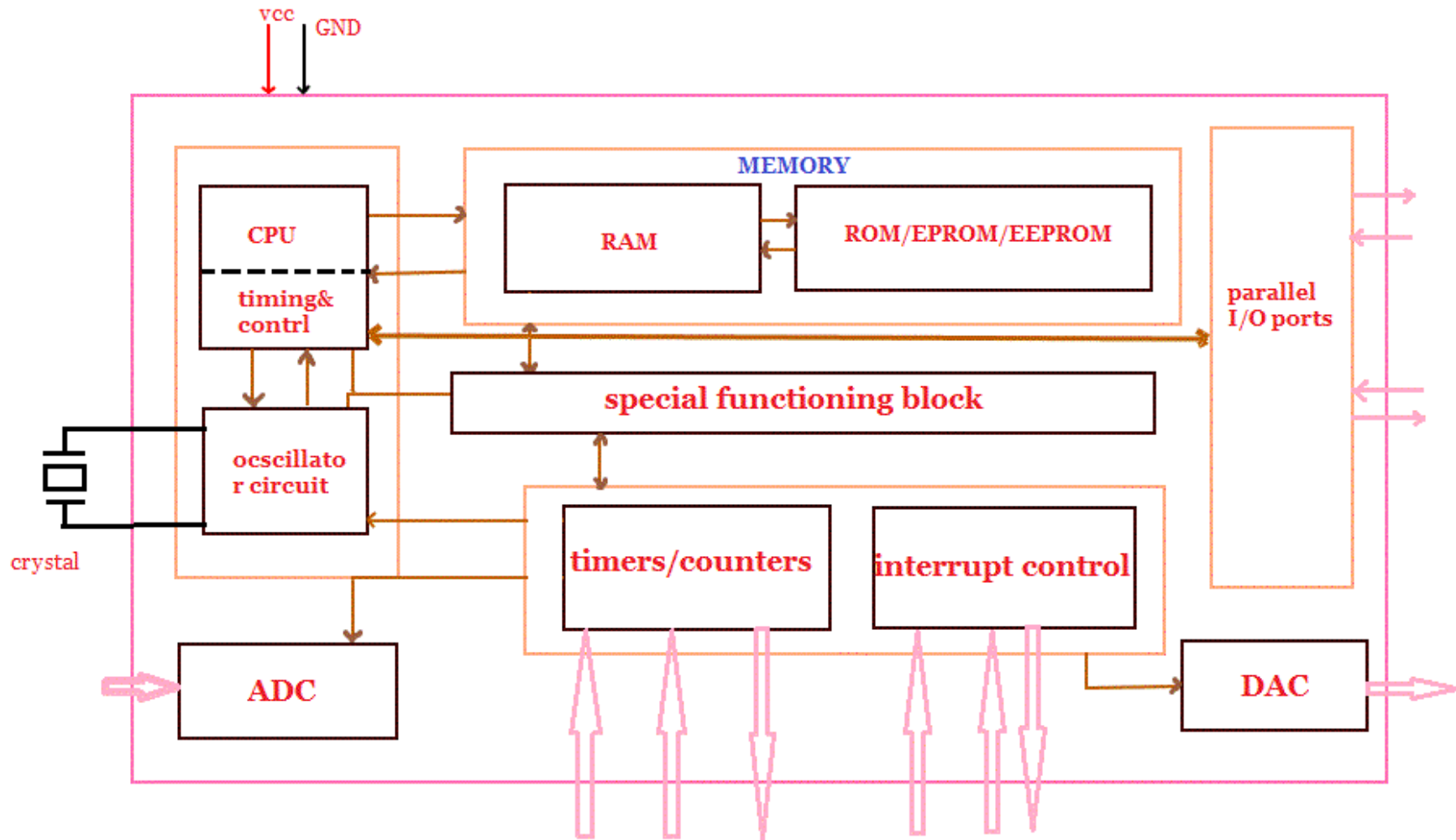


Introduction to Biomedical Engineering

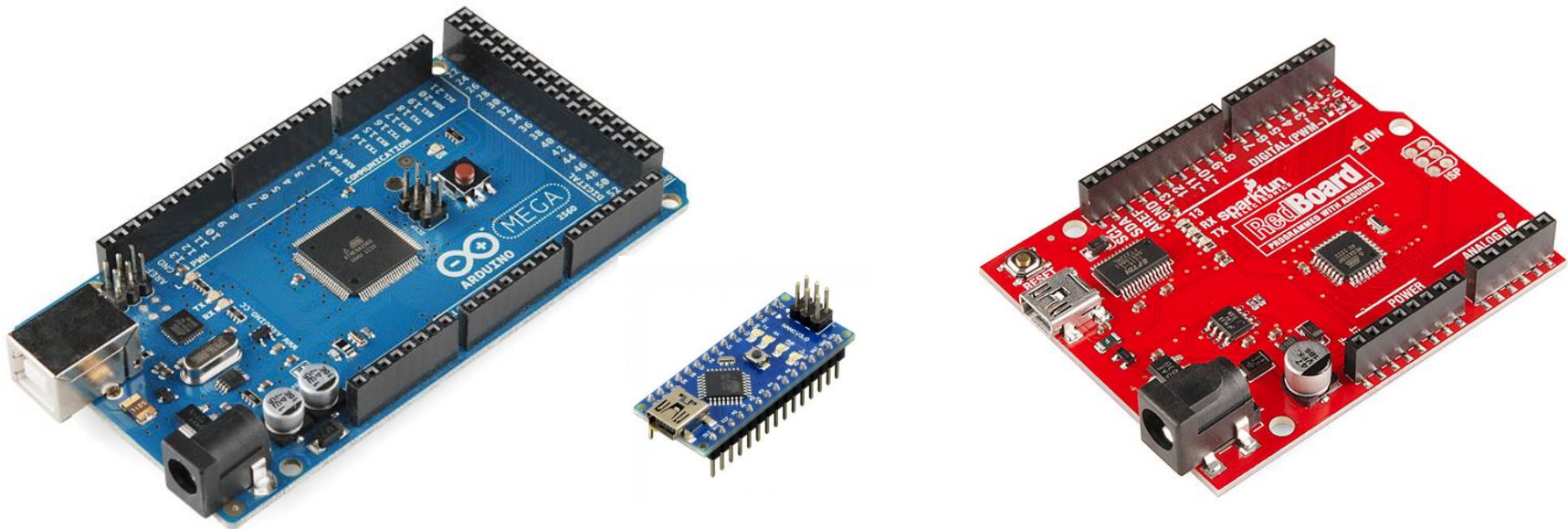
Section 3: Microcontrollers/Arduino

Lecture 3.2: System components and hardware

Microcontroller structure

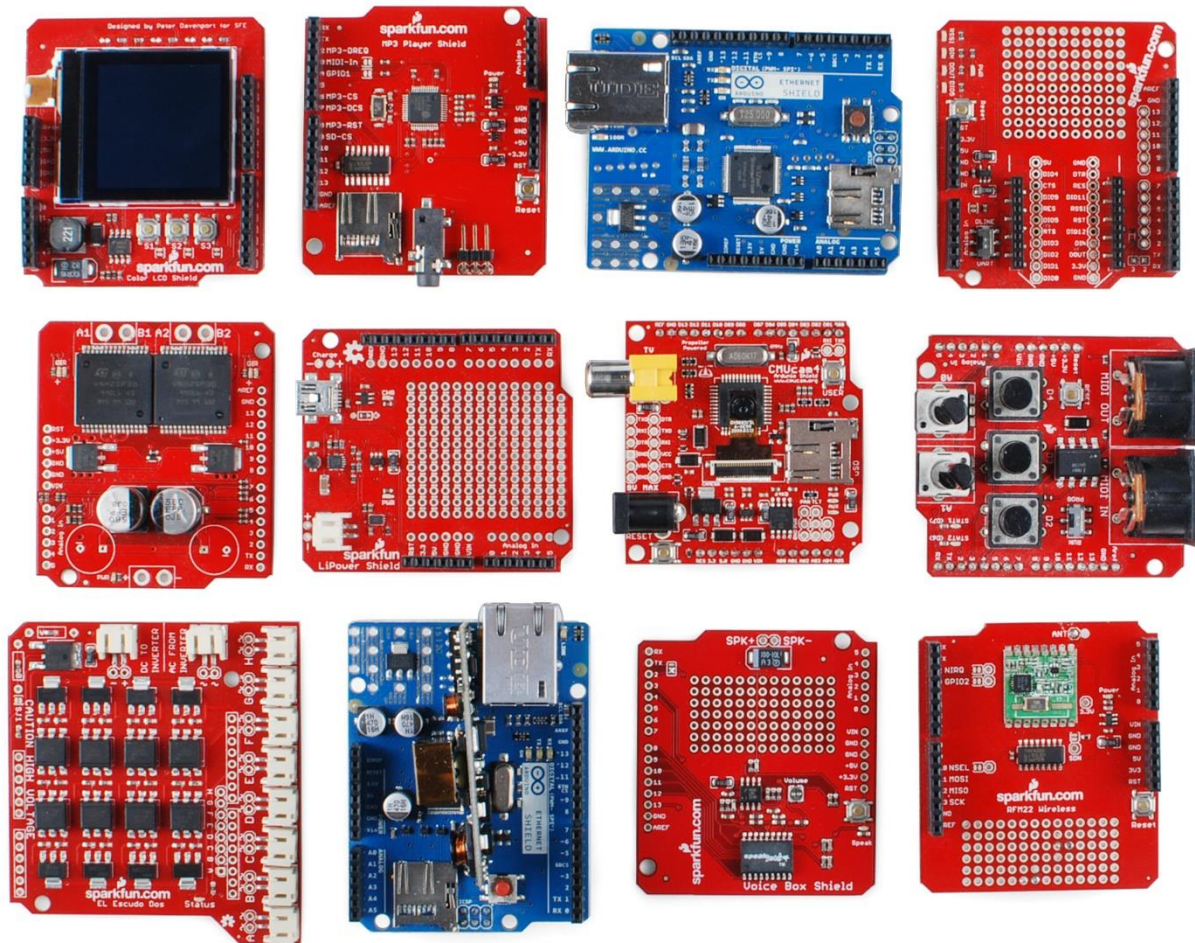


ARDUINO: what is available and what is it good for?



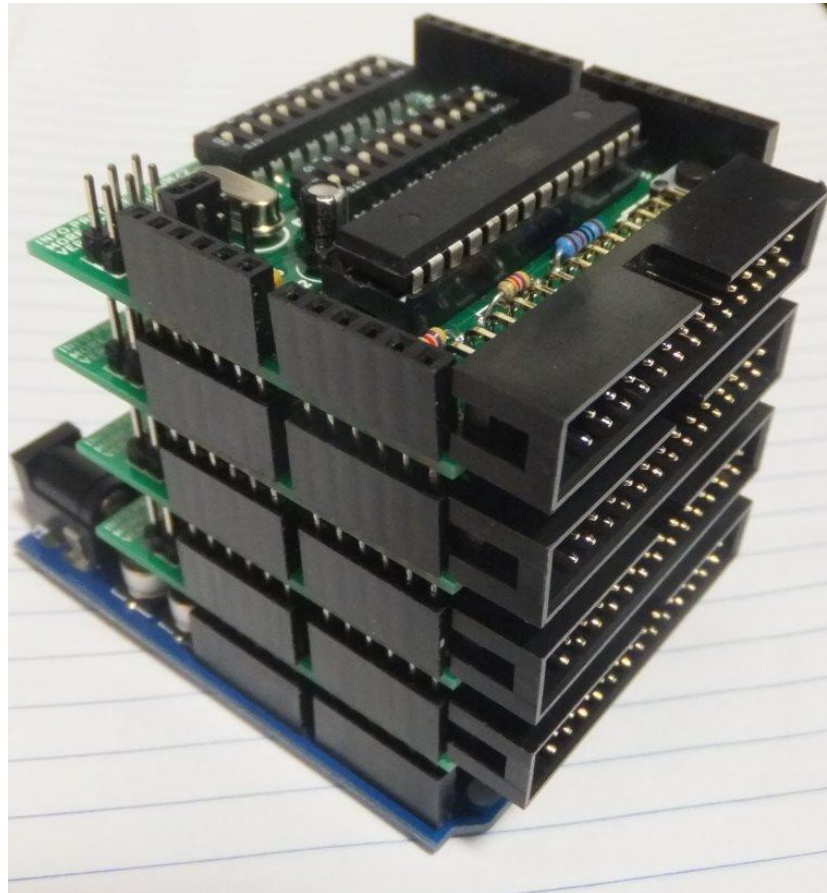
ARDUINO: what is available and what is it good for?

Shields

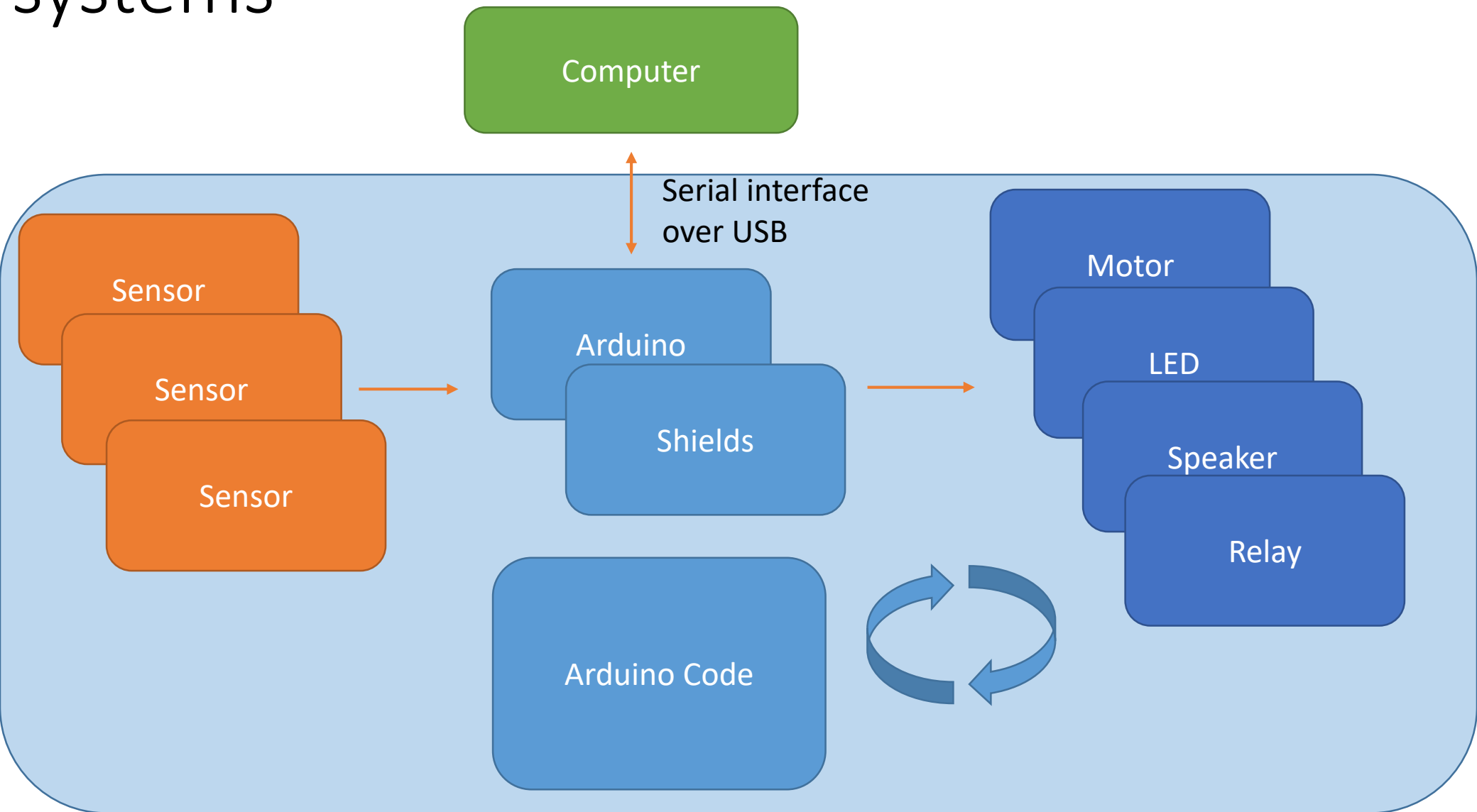


ARDUINO: what is available and what is it good for?

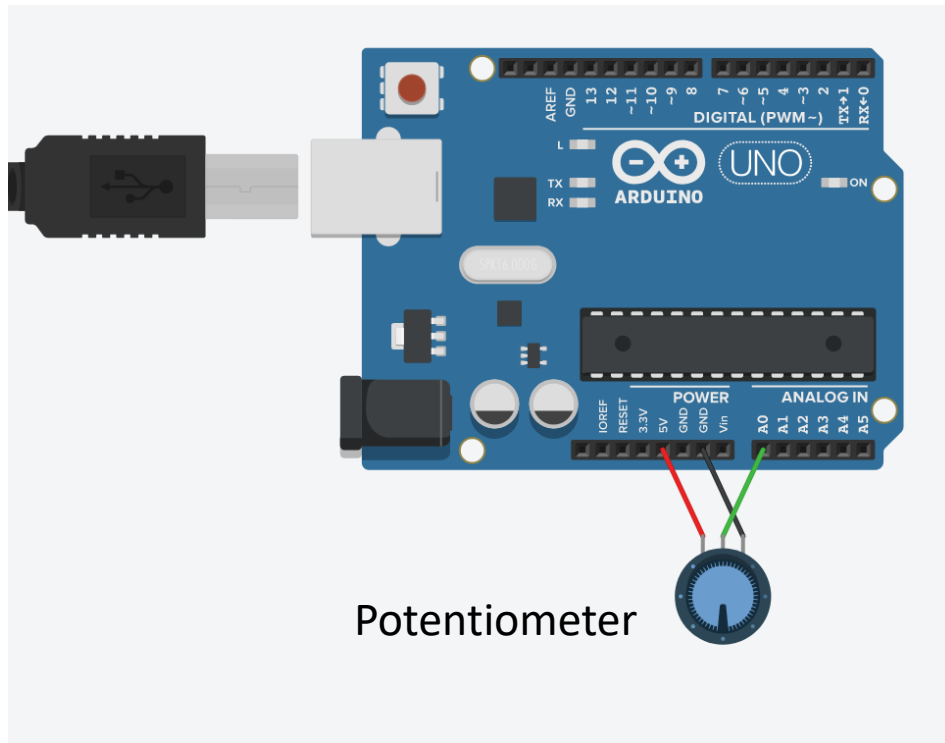
Shields



General idea: real-time embedded/robotic systems



Sensors example

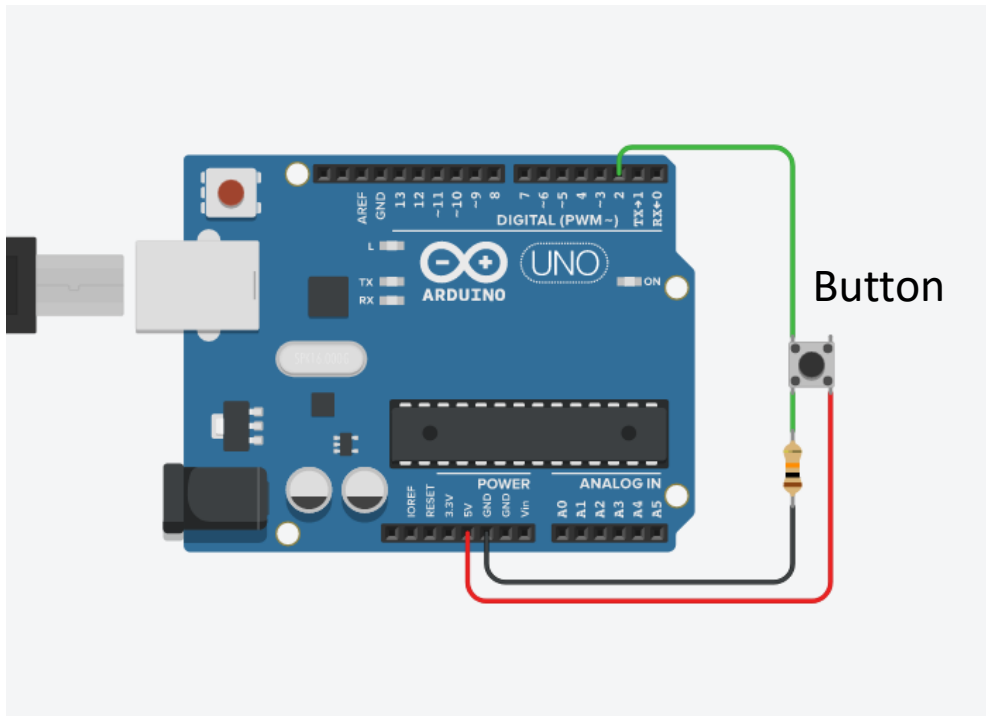


```
int sensorValue = 0;

void setup()
{
  pinMode(A0, INPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  // read the value from the sensor
  sensorValue = analogRead(A0);
  // turn the LED on
  digitalWrite(13, HIGH);
  // stop the program for the <sensorValue>
  // milliseconds
  delay(sensorValue); // Wait for sensorValue millisecond(s)
  // turn the LED off
  digitalWrite(13, LOW);
  // stop the program for the <sensorValue>
  // milliseconds
  delay(sensorValue); // Wait for sensorValue millisecond(s)
}
```


Serial interface: sending stuff



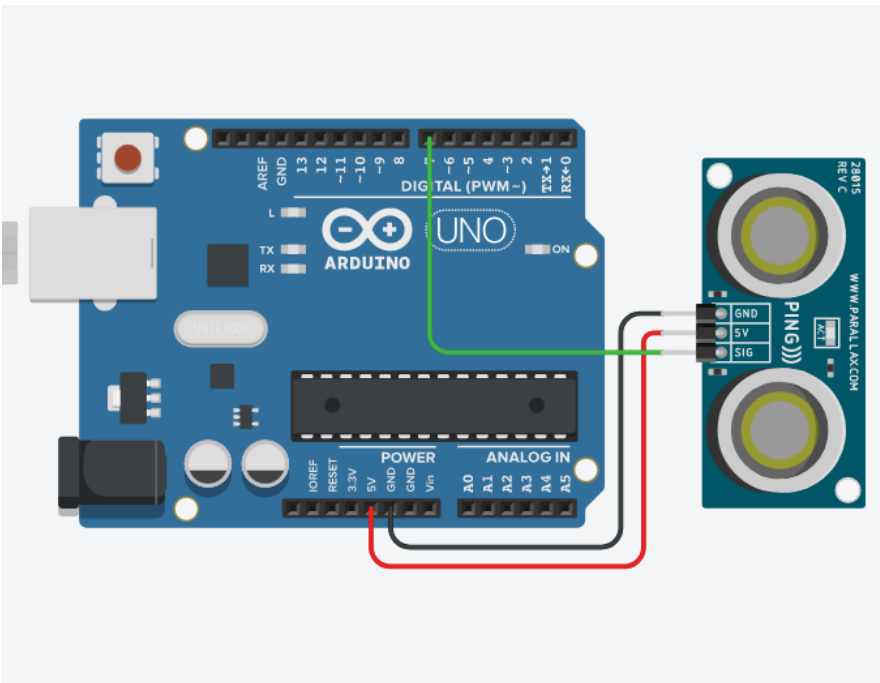
```
int buttonState = 0;

void setup()
{
  pinMode(2, INPUT);
  Serial.begin(9600);
}

void loop()
{
  // read the input pin
  buttonState = digitalRead(2);
  // print out the state of the button
  Serial.println(buttonState);
  delay(10); // Delay a little bit to improve simulation performance
}
```

Sensors example: more sophisticated

Ultrasonic range detector



```
int cm = 0;

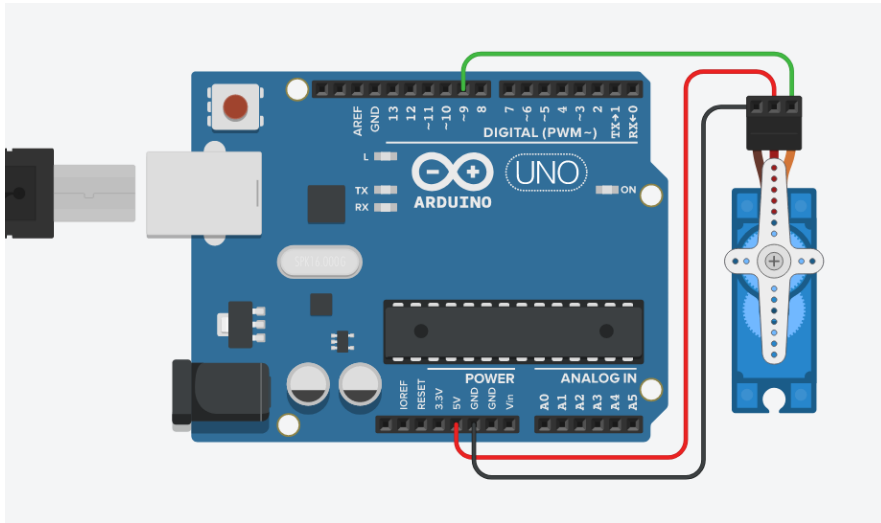
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    // measure the ping time in cm
    cm = 0.01723 * readUltrasonicDistance(7, 7);

    Serial.print(cm);
    Serial.println("cm");
    delay(100); // Wait for 100 millisecond(s)
}

long readUltrasonicDistance(int triggerPin, int echoPin)
{
    pinMode(triggerPin, OUTPUT); // Clear the trigger
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    // Sets the trigger pin to HIGH state for 10 microseconds
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
    pinMode(echoPin, INPUT);
    // Reads the echo pin, and returns the sound wave travel time in microseconds
    return pulseIn(echoPin, HIGH);
}
```

Motors



```
#include <Servo.h>

int pos = 0;

Servo servo_9;

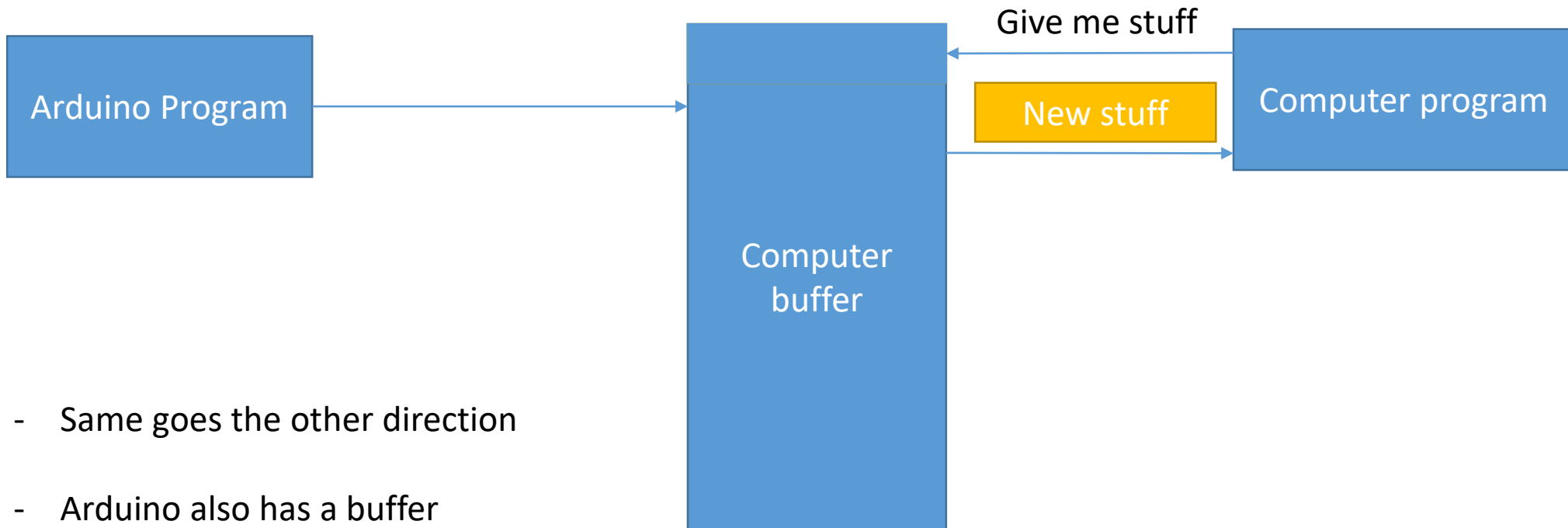
void setup()
{
    servo_9.attach(9);
}

void loop()
{
    // sweep the servo from 0 to 180 degrees in steps
    // of 1 degrees
    for (pos = 0; pos <= 180; pos += 1) {
        // tell servo to go to position in variable 'pos'
        servo_9.write(pos);
        // wait 15 ms for servo to reach the position
        delay(15); // Wait for 15 millisecond(s)
    }
    for (pos = 180; pos >= 0; pos -= 1) {
        // tell servo to go to position in variable 'pos'
        servo_9.write(pos);
        // wait 15 ms for servo to reach the position
        delay(15); // Wait for 15 millisecond(s)
    }
}
```

Serial in depth

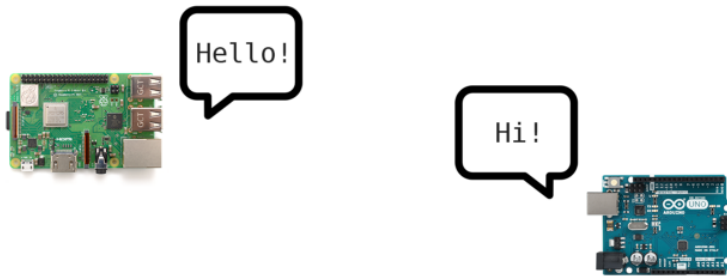
- Serial is used for communication between the Arduino board and a computer or other devices.
- All Arduino boards have at least one serial port (also known as a UART or USART): Serial.
- It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB.

How it works

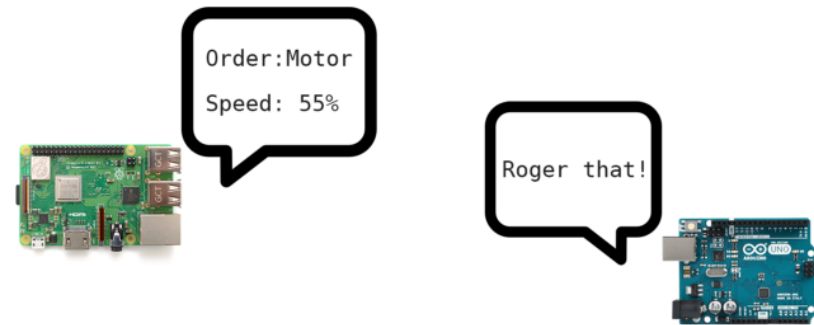


- Same goes the other direction
- Arduino also has a buffer
- What has been sent: bytes (in serial order, hence serial interface)

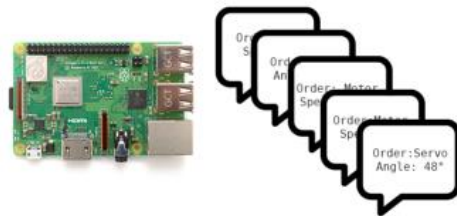
Some preliminary considerations



First step: connection,handshake



Sending messages and aknowlegment of the receipt



Buffer Overflow Problem



Follow the structure and rules

- 1) Connection, you make sure the Arduino is connected
 - Then, for each message you want to exchange, you send:
- 2) The type of order (e.g. “MOTOR”)
- 3) The parameters (e.g. speed)
- 4) And then you wait for an acknowledgment of the Arduino to send new orders.

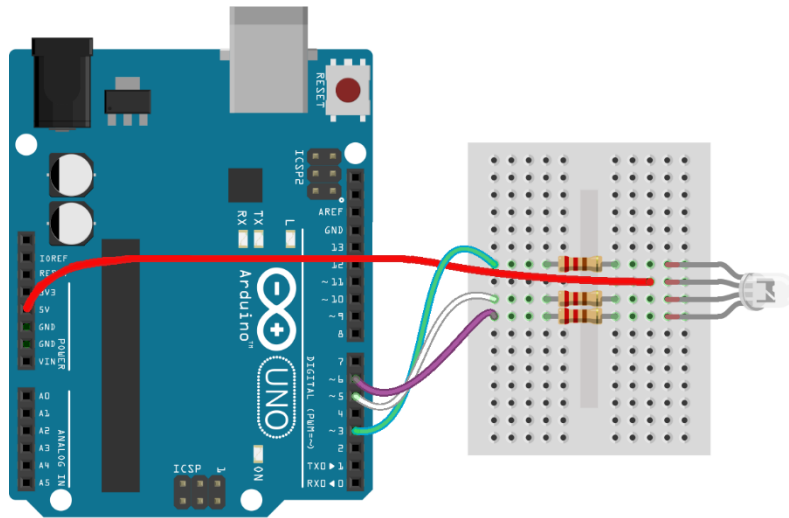
Practical implementation

- It is up to you to design
- You can use built in functions, but READ exactly what they are doing first to avoid long debugging nights

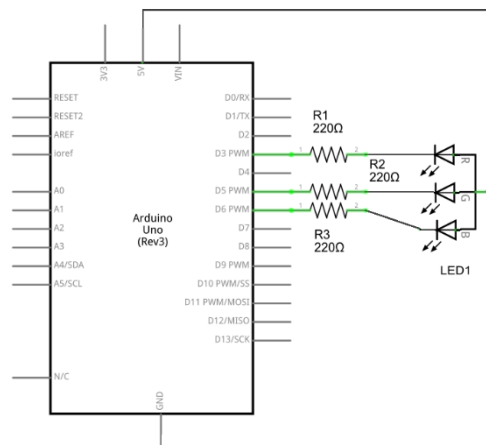
Functions

- `if (Serial)`
- `available()`
- `availableForWrite()`
- `begin()`
- `end()`
- `find()`
- `findUntil()`
- `flush()`
- `parseFloat()`
- `parseInt()`
- `peek()`
- `print()`
- `println()`
- `read()`
- `readBytes()`
- `readBytesUntil()`
- `readString()`
- `readStringUntil()`
- `setTimeout()`
- `write()`
- `serialEvent()`

Example: read ASCII string



fritzing



fritzing

```
// pins for the LEDs:
const int redPin = 3;
const int greenPin = 5;
const int bluePin = 6;

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // make the pins outputs:
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  // if there's any serial available, read it:
  while (Serial.available() > 0) {

    // look for the next valid integer in the incoming serial stream:
    int red = Serial.parseInt();
    // do it again:
    int green = Serial.parseInt();
    // do it again:
    int blue = Serial.parseInt();

    // look for the newline. That's the end of your sentence:
    if (Serial.read() == '\n') {
      // constrain the values to 0 - 255 and invert
      // if you're using a common-cathode LED, just use "constrain(color, 0, 255);"
      red = 255 - constrain(red, 0, 255);
      green = 255 - constrain(green, 0, 255);
      blue = 255 - constrain(blue, 0, 255);

      // fade the red, green, and blue legs of the LED:
      analogWrite(redPin, red);
      analogWrite(greenPin, green);
      analogWrite(bluePin, blue);

      // print the three numbers in one string as hexadecimal:
      Serial.print(red, HEX);
      Serial.print(green, HEX);
      Serial.println(blue, HEX);
    }
  }
}
```

Debugging serial

- Arduino IDE has build-in terminal, lets you see what is in the buffer and send stuff to Arduino buffer from a computer (tools-> serial monitor)
- Use structures you can read or designed yourself: this way you know exactly what you expect
 - Particularly: are you sending raw bytes or strings? Did you convert strings to bytes? Integers? Floating point? Data converters will help you, also knowledge of the data types and their byte composition helps!

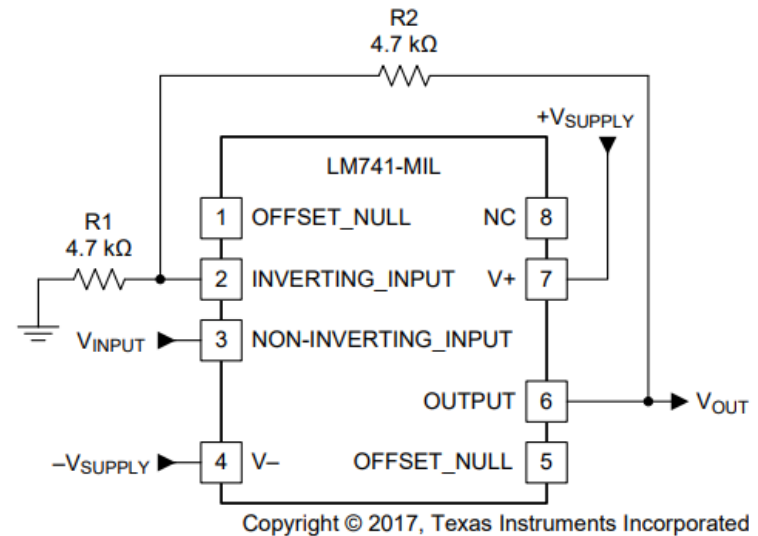
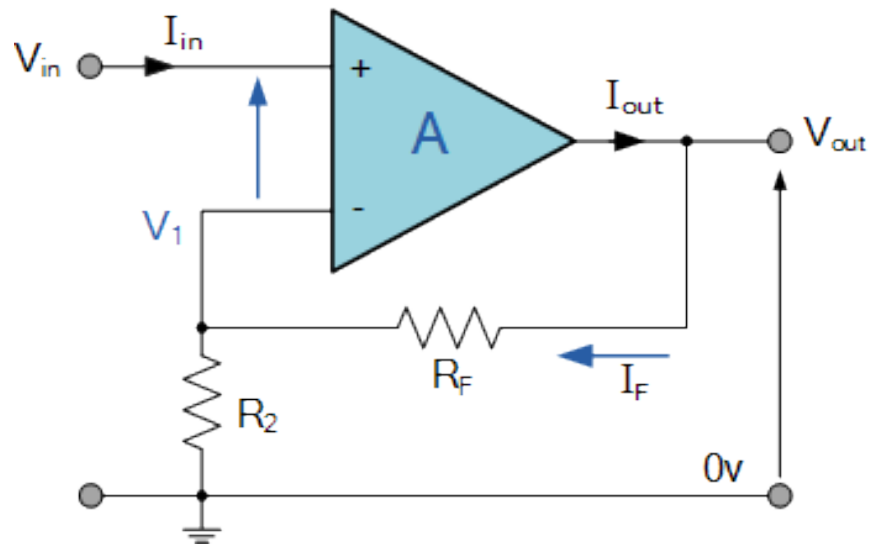
Lets have some fun

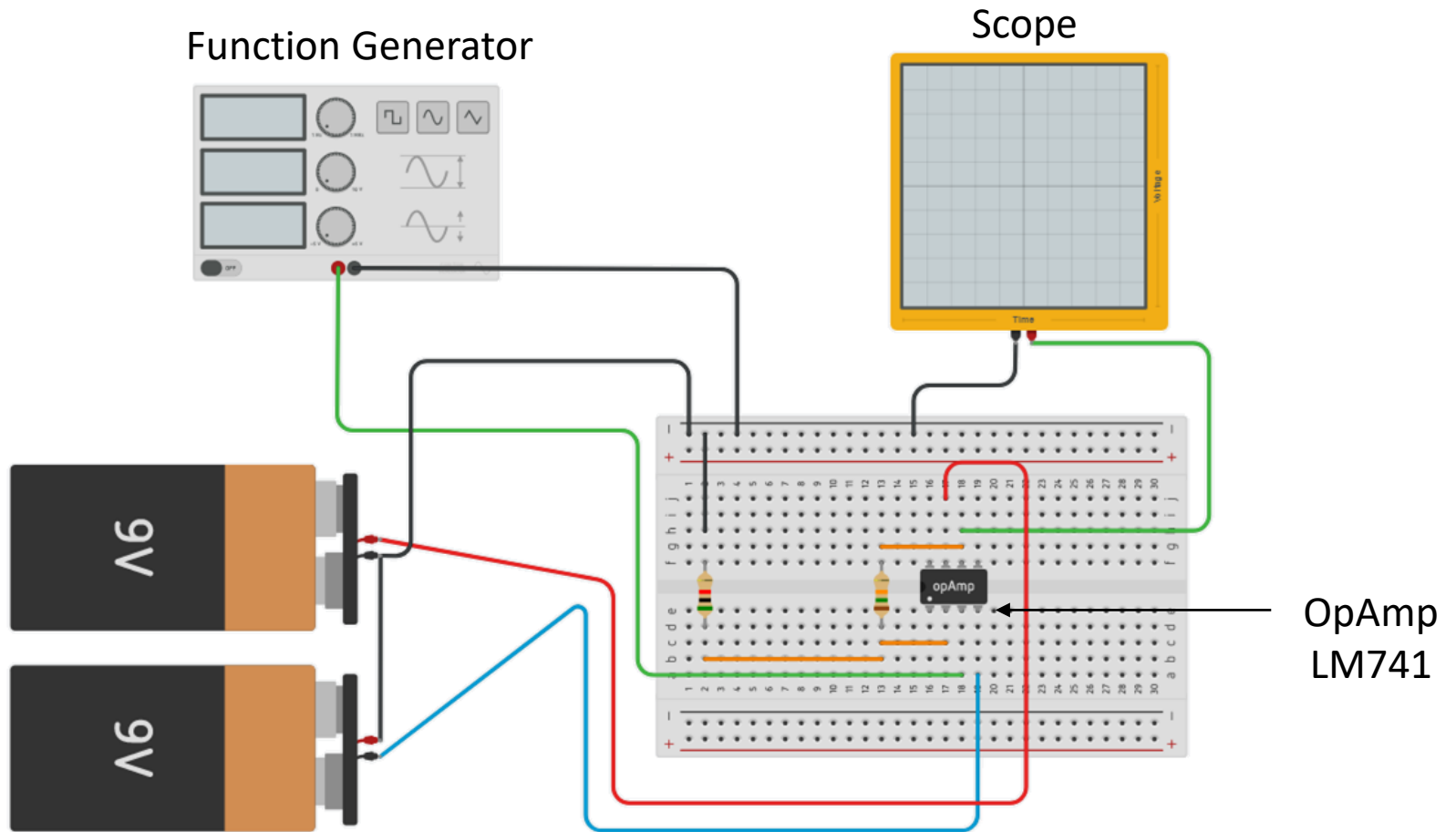
<https://www.tinkercad.com>

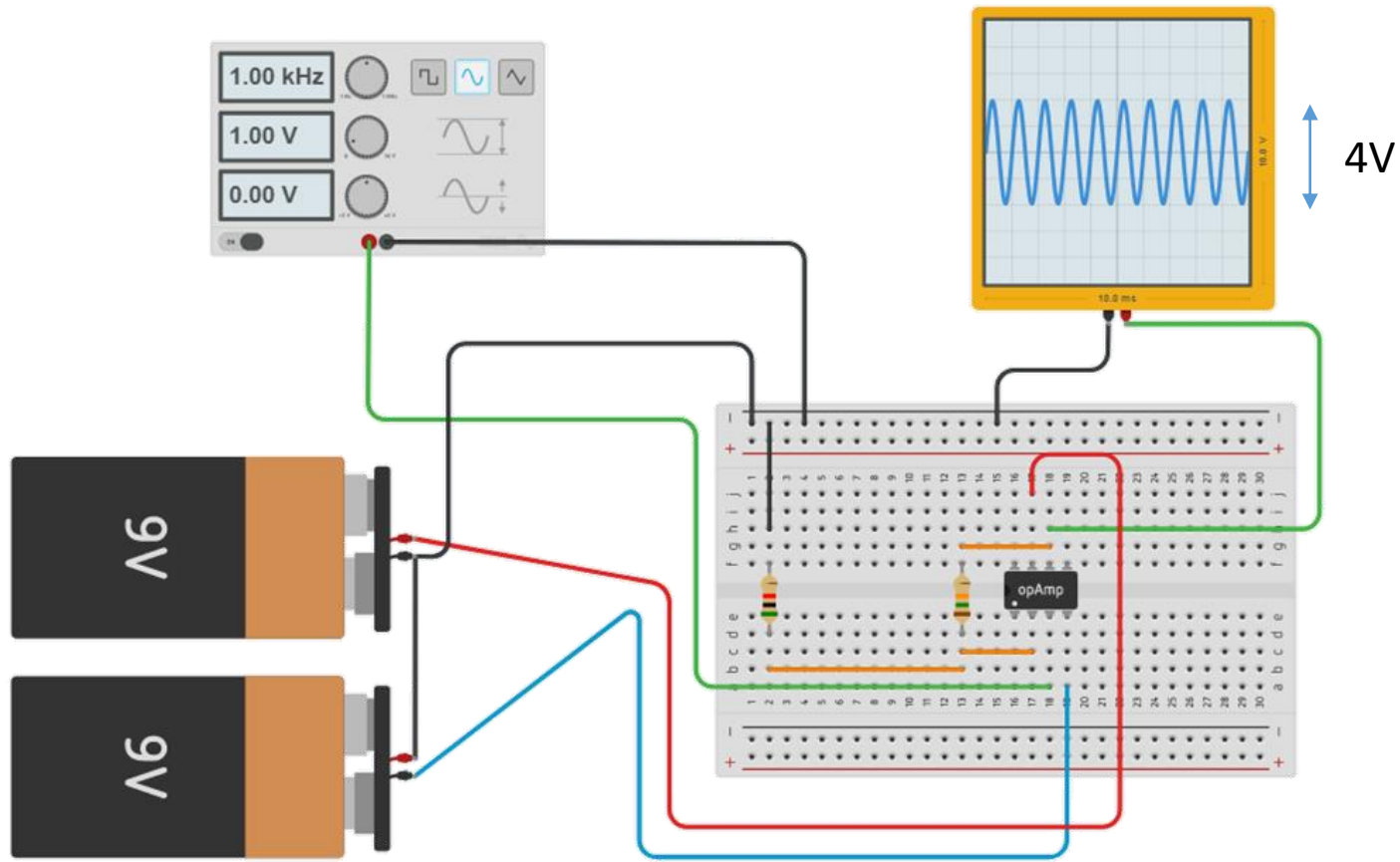
Circuits

Create new Circuit

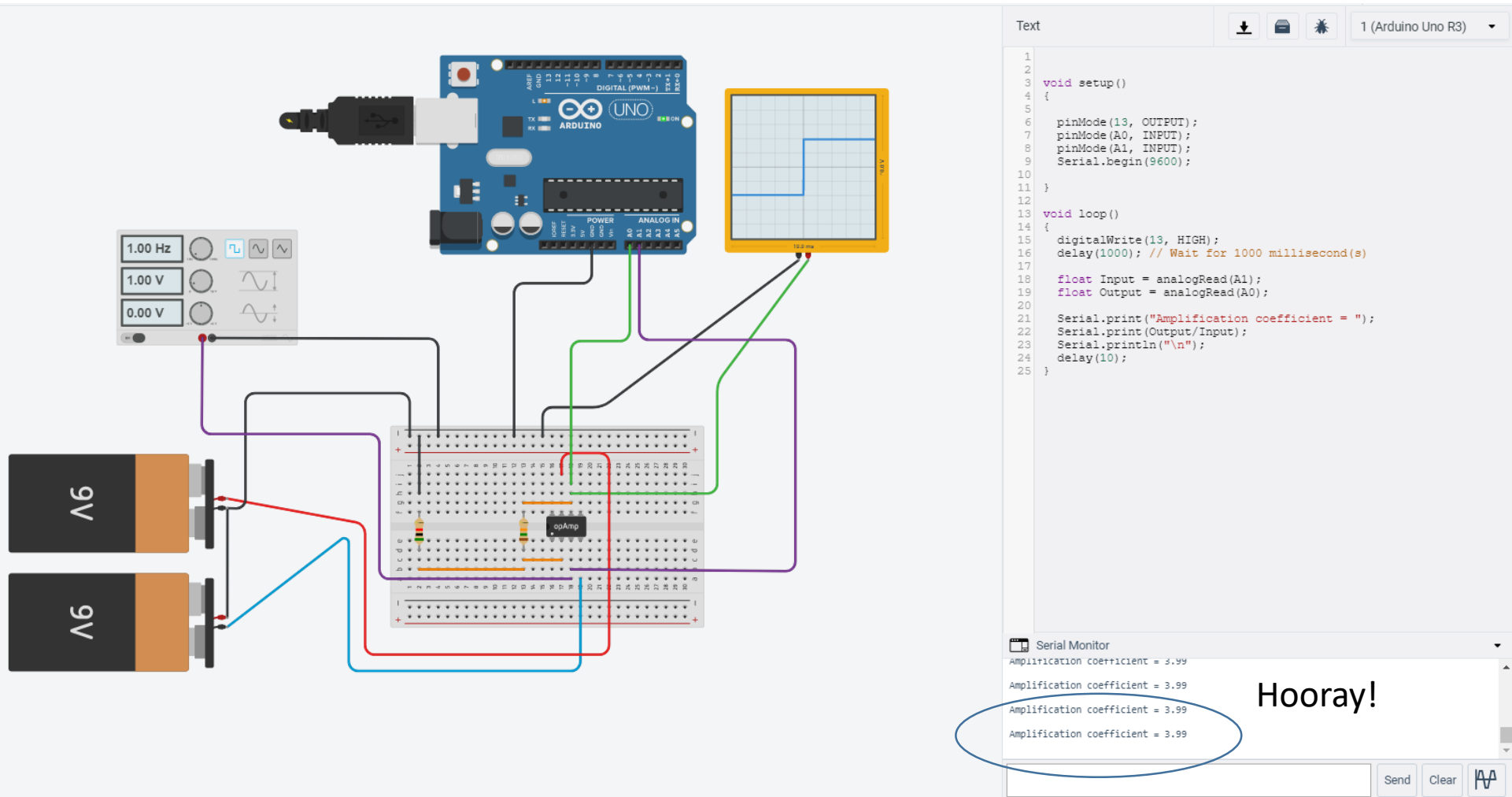
Non-inverting amplifier based on OpAmp LM741







Now, lets do the same with arduino



The image shows a breadboard circuit with an opAmp (opAmp) connected to an Arduino Uno R3. The circuit is powered by two 9V batteries. The Arduino is connected to the opAmp and an oscilloscope. The function generator is connected to the opAmp input. The oscilloscope shows a square wave output. The Arduino code in the Serial Monitor shows the amplification coefficient is 3.99.

```
1
2
3 void setup()
4 {
5
6   pinMode(13, OUTPUT);
7   pinMode(A0, INPUT);
8   pinMode(A1, INPUT);
9   Serial.begin(9600);
10
11 }
12
13 void loop()
14 {
15   digitalWrite(13, HIGH);
16   delay(1000); // Wait for 1000 millisecond(s)
17
18   float Input = analogRead(A1);
19   float Output = analogRead(A0);
20
21   Serial.print("Amplification coefficient = ");
22   Serial.print(Output/Input);
23   Serial.println("\n");
24   delay(10);
25 }
```

Serial Monitor

Amplification coefficient = 3.99
Amplification coefficient = 3.99
Amplification coefficient = 3.99
Amplification coefficient = 3.99

Hooray!

Thank you for your attention!

Some graphic material used in the course was taken from publicly available online resources that do not contain references to the authors and any restrictions on material reproduction.