

Tassu Takala
7 Feb 2019

Emergent User Interfaces

CS-E4200

Lecture 5 - Physical Computing

—

- Sensors, Actuators, Interfaces
- Practical Notes on Audio & Video

—

Matchmaking

Agenda

- Interfacing with physical world
- Components
 - sensors, actuators
 - controllers: embedded vs. generic computer
- Programming environments
 - Arduino, Processing
- Audio and video
 - signal processing for sound and images



Aalto University
School of Science

Acknowledgement:

This part based on last year's lectures by Mikko Kytö

Emergent User Interfaces

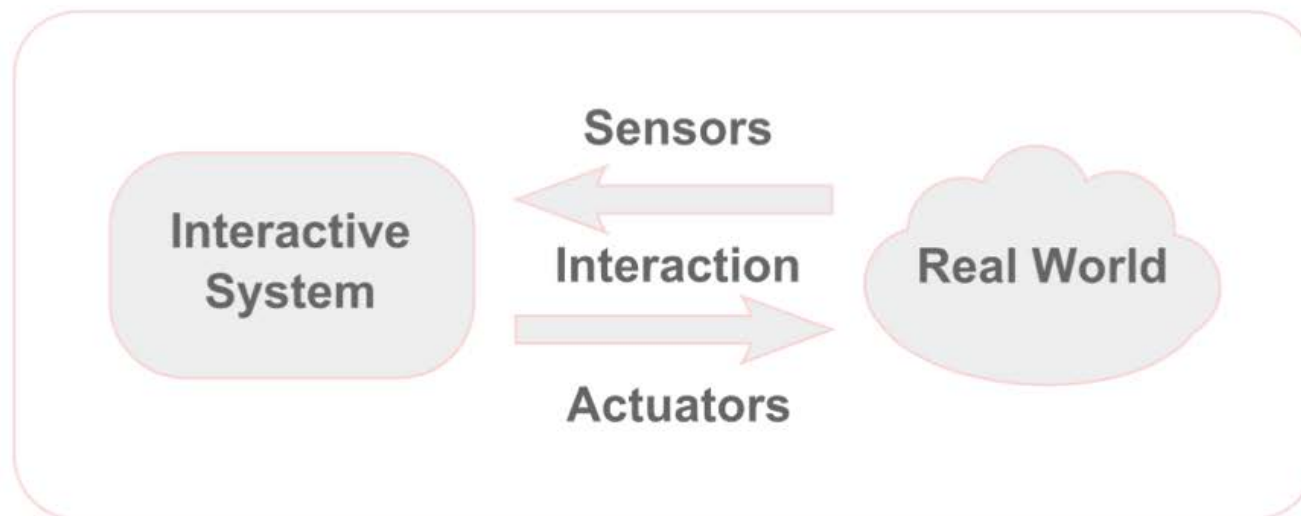
Introduction to Physical Computing

Mikko Kytö

8.2.2018

Physical computing

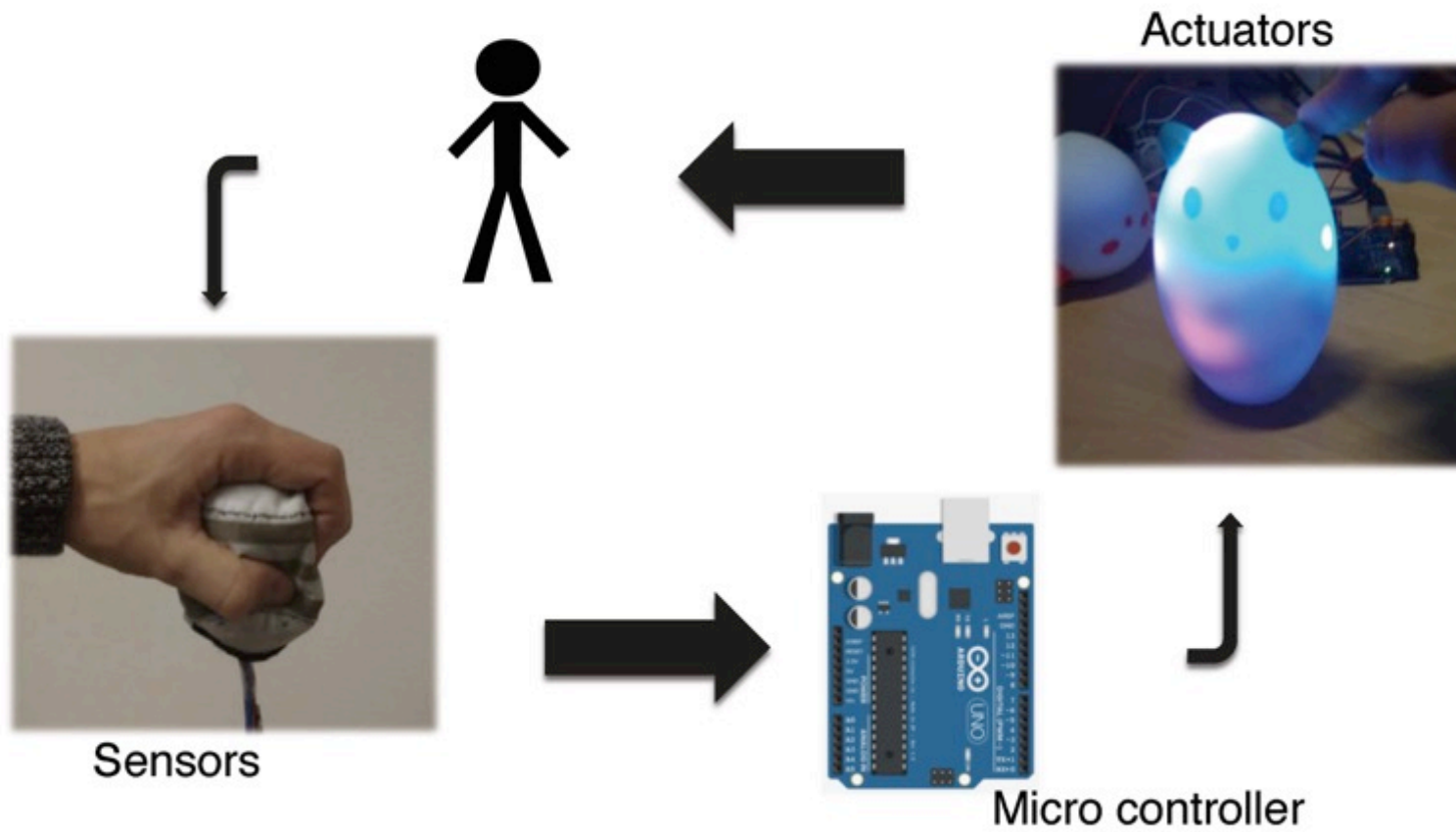
Computing that senses and responds to real world

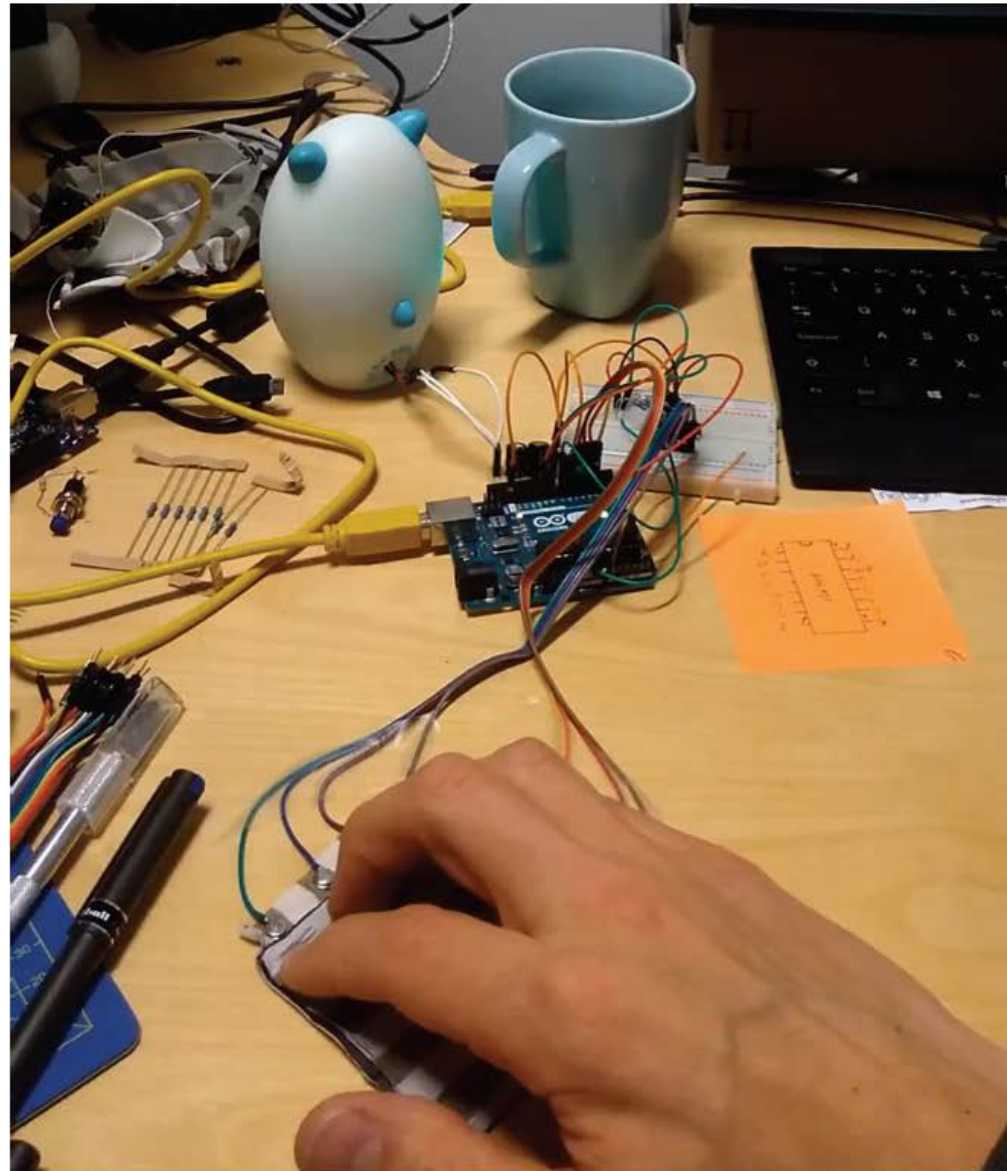


https://en.wikipedia.org/wiki/Physical_computing

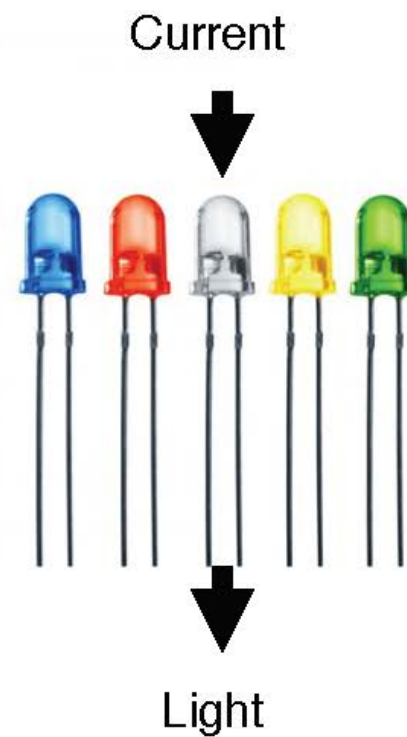
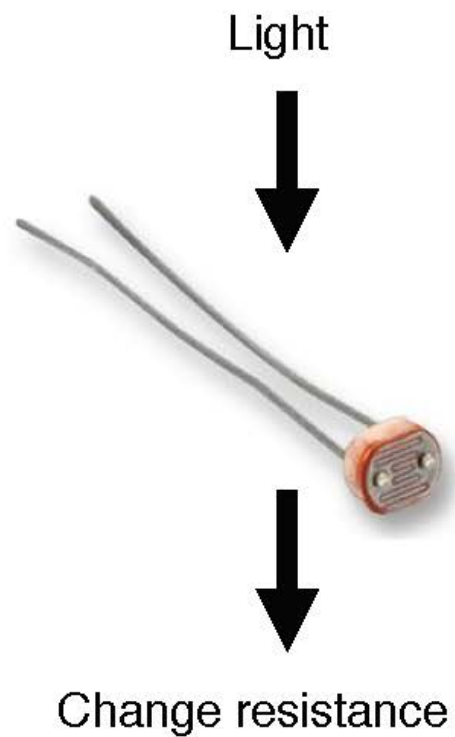
<https://en.wikipedia.org/wiki/Sensor>

Interaction





Photocell & LEDs



Binary Sensors

Push



Change signal routing

Tilt



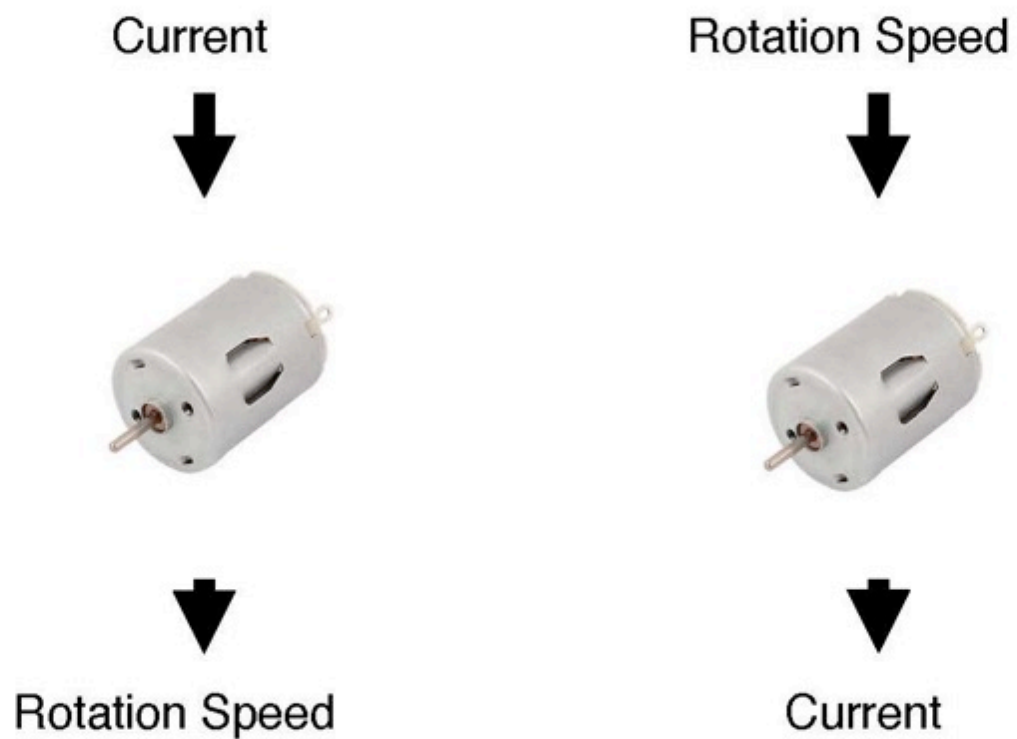
Change signal routing



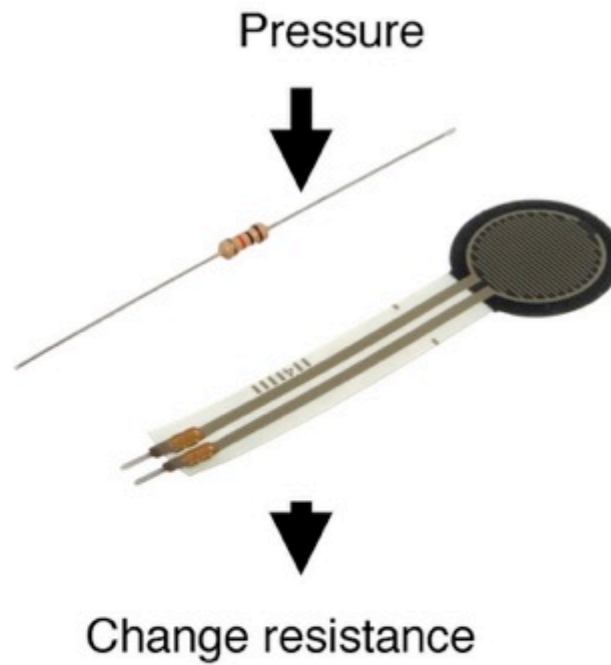
Pots & Servos



DC Motors



Pressure



Flex sensor



Some advanced components

- Stripe of individually programmable LEDs
<https://www.adafruit.com/products/1138>



- Heart rate sensor
<https://pulsesensor.com>
 - measures light permeability of blood in the finger



- Accelerometer

<https://playground.arduino.cc/Main/MPU-6050>





Aalto University
School of Science

Arduino

Available from arduino.cc

Arduino

IDE Available from [arduino.cc](https://www.arduino.cc)

Lots of different models:

<https://www.arduino.cc/en/Main/Products>

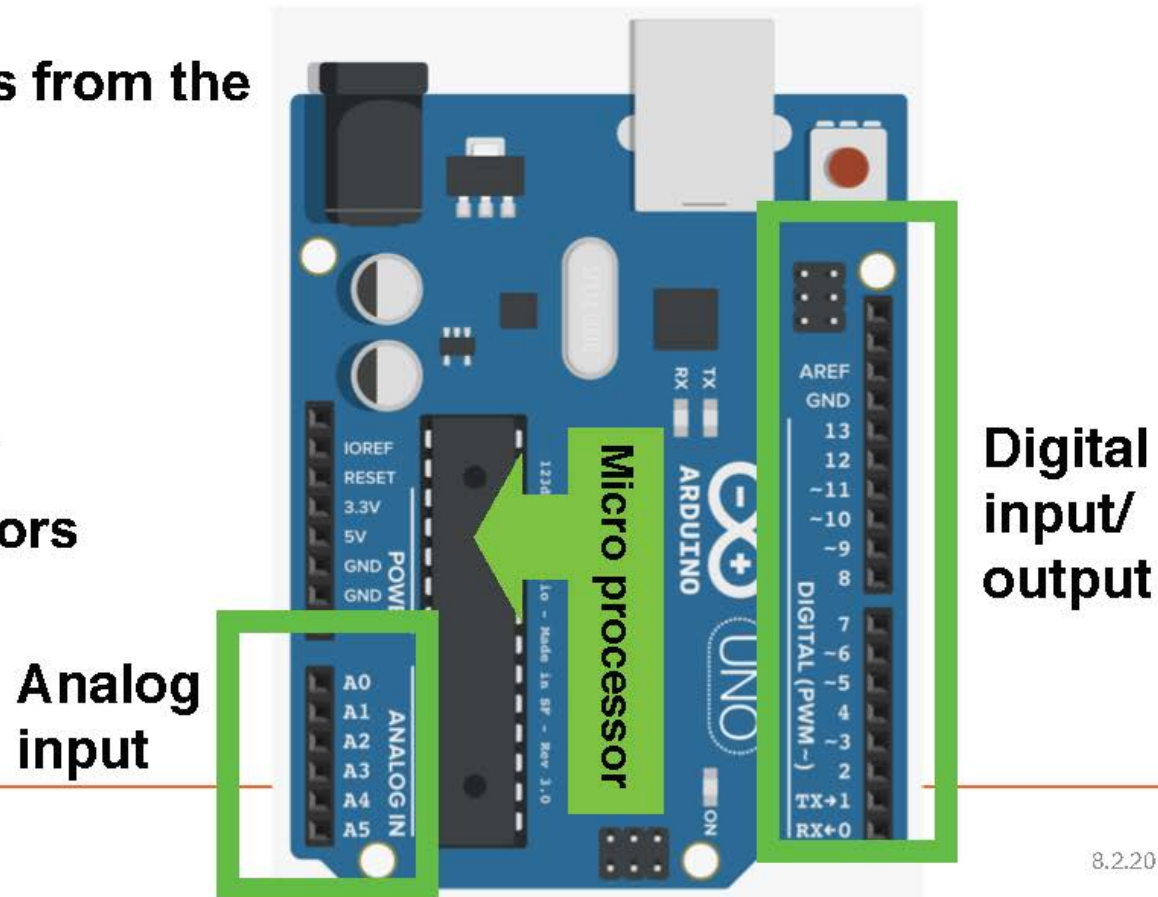
Arduino

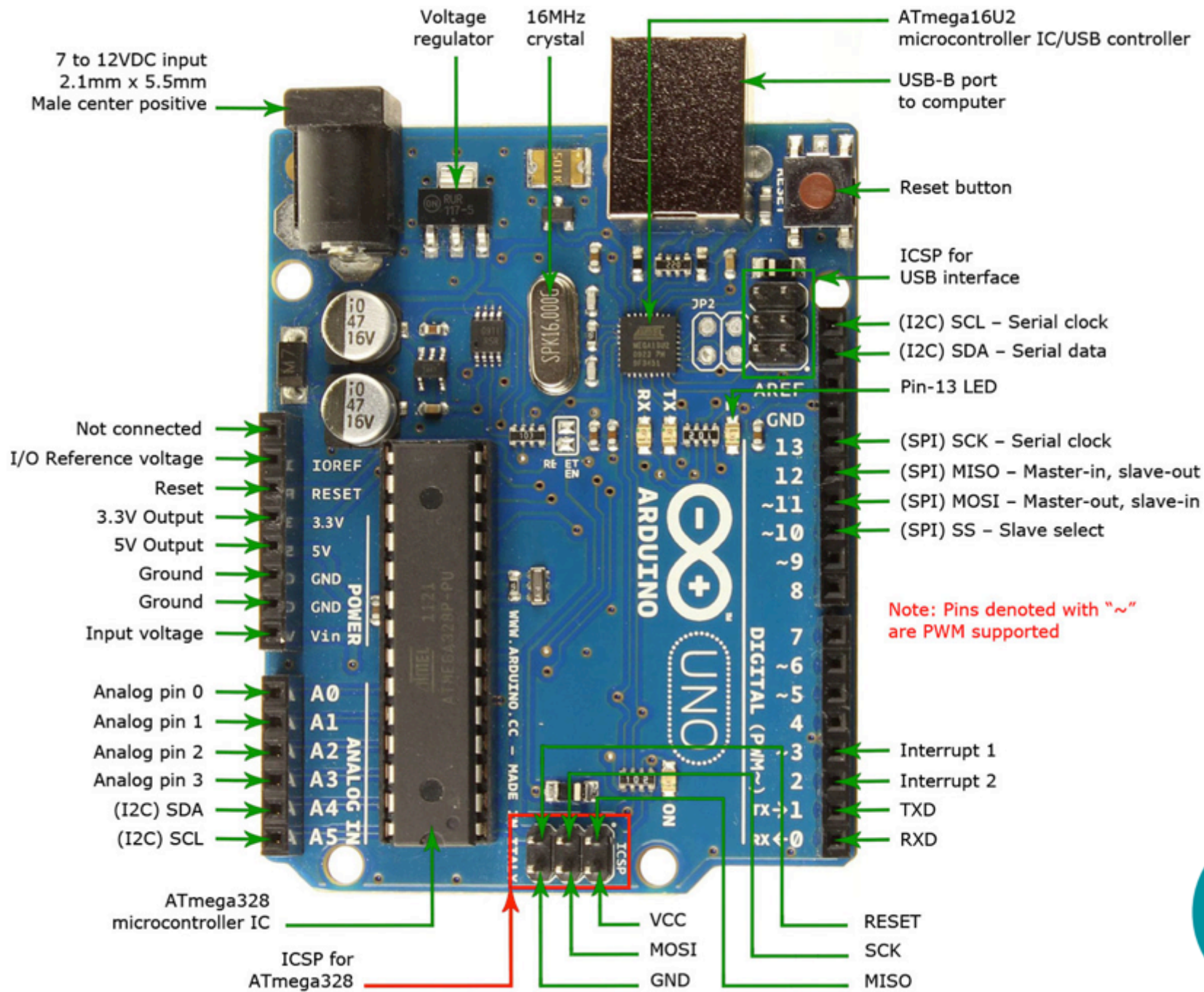
- <https://www.arduino.cc>
- Programmable microprocessor, with multiple I/O-ports
- Can run independently (with an attached power source) as an embedded system, or as peripheral device connected to a computer through USB
- Used for prototyping by hackers, as it allows easy connection to various sensors and actuators



Arduino's core functionalities

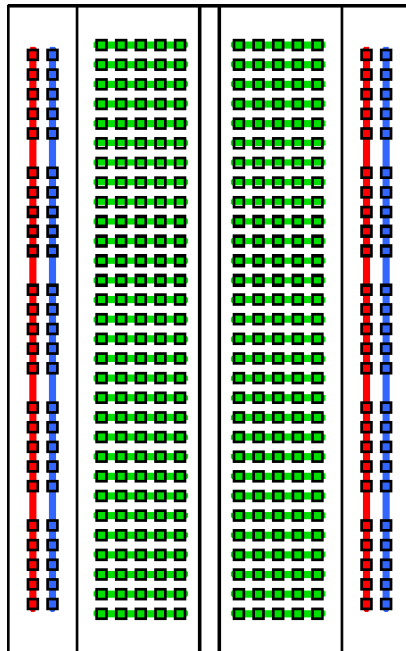
- **Reads the values from the sensors**
 - Analog input
 - Digital input
- **Computation**
 - Micro processor
- **Output to actuators**
 - Digital output
 - PWM



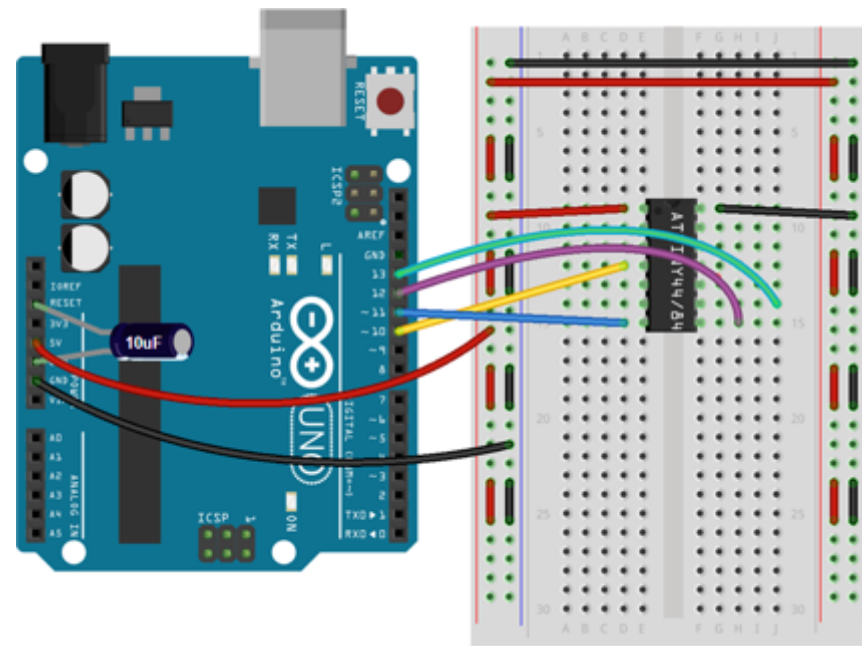


The 'breadboard'


- Connecting base for building your prototype electronics
- Main connections to Arduino:
power voltage (GND and +5V) and I/O-channels



built-in connections

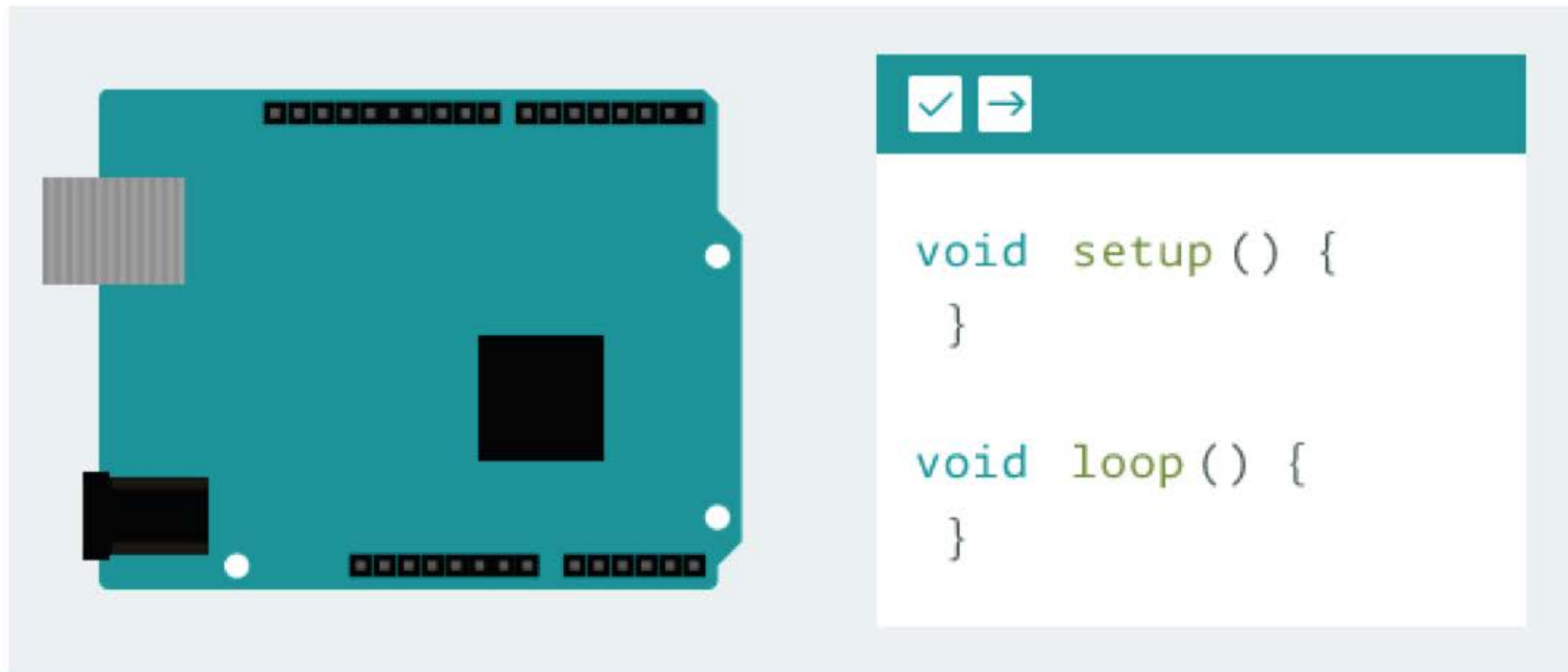


Getting started

- <https://www.arduino.cc/en/Guide/HomePage>
- Download the Arduino IDE (Integrated development environment)
- Connect Arduino with USB to your laptop
- Start the Arduino application 
- Open the example Basics / Blink
- Select from Tools-menu the right card (Arduino Uno) and USB-port (names depend on your configuration)
- Run and test the program

Then let's have a look inside the code...

Arduino software structure



Arduino – digital output

in Arduino Examples/Basics/Blink
the number is 13 (**LED_BUILTIN**)

```
//DECLARING VARIABLES  
int ledPin = 11; // LED IS CONTROLLED BY PIN #11  
  
void setup {  
  //SETTING PIN #11 TO OUTPUT MODE  
  pinMode(ledPin, OUTPUT);  
  
}
```



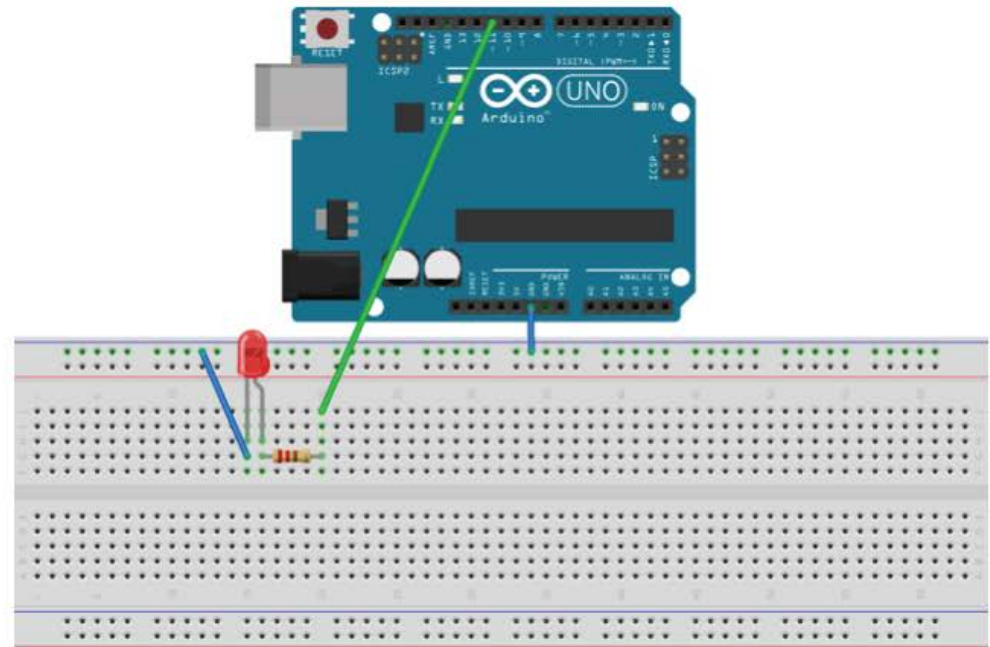
Arduino – digital output

```
void loop() {  
  //THIS CODE RUNS CONTINUOSLY  
  digitalWrite(ledPin,HIGH);  
  delay(1000); // in milliseconds  
  digitalWrite(ledPin,LOW);  
  delay(1000);  
}
```

Example 1.1 – blinking LED

```
int ledPin = 11; // LED IS CONTROLLED BY PIN #11
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  //THIS CODE RUNS CONTINUOSLY
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```


Example 1.1



fritzing

see also Arduino Examples/Fade

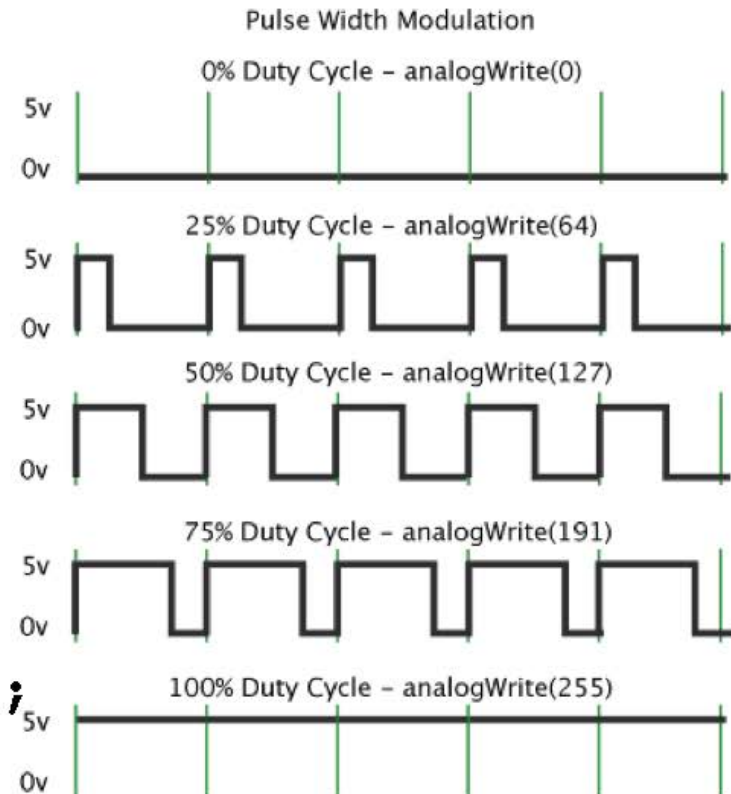
Example 1.2 – fading LED, "Analog output"

File -> Examples -> Analog -> Fading

Change PIN 9 to PIN 11

And run!

```
analogWrite(ledPin, fadeValue);
```



Arduino – digital input

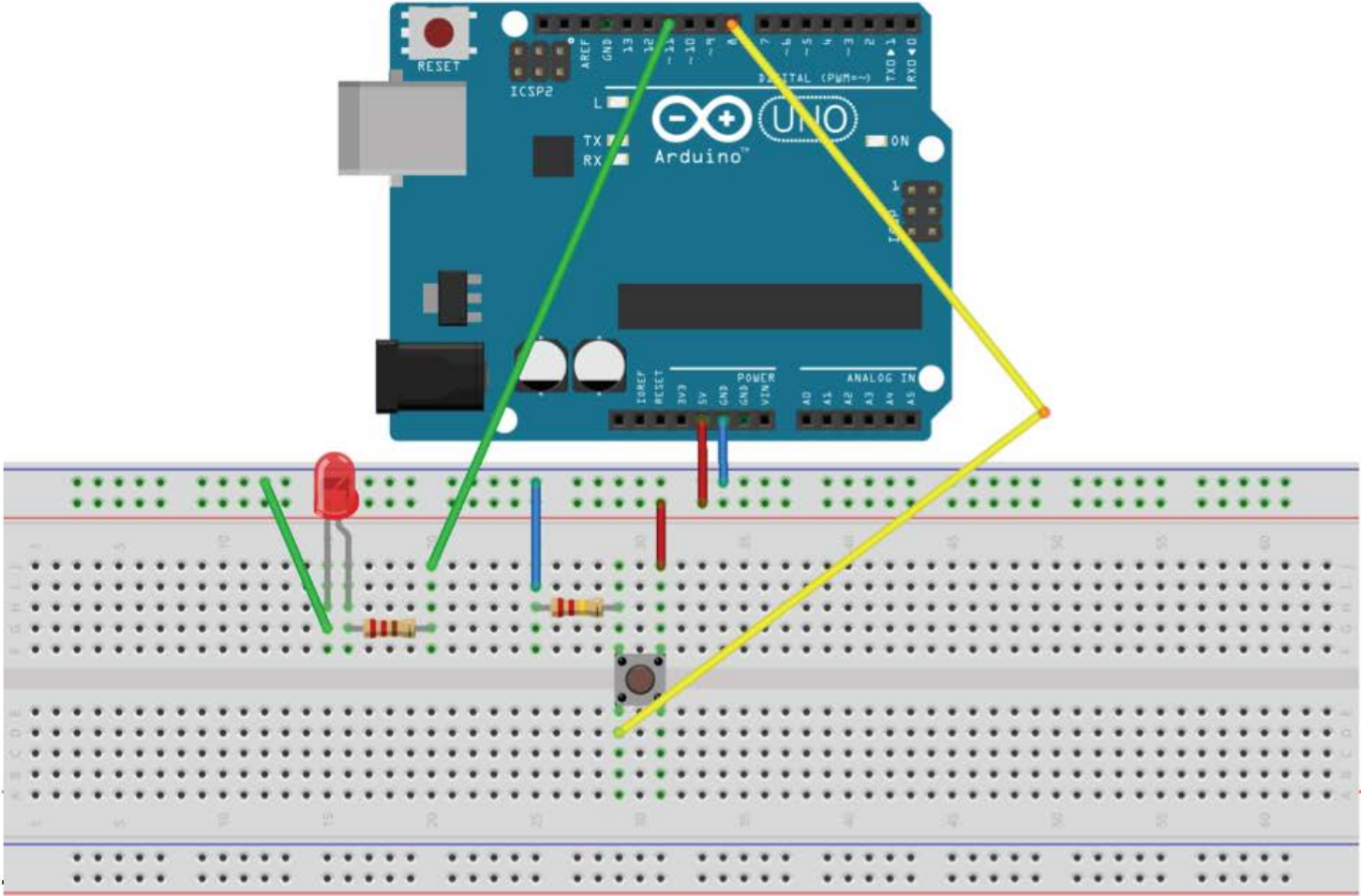
```
//DECLARING VARIABLES
int buttonPin = 8; // BUTTON IS CONNECTED TO PIN #8
int buttonState = 0;
void setup() {
  //SETTING PIN #8 TO INPUT MODE
  pinMode(buttonPin, INPUT);
}
void loop() {
  //THIS CODE RUNS CONTINUOUSLY
  buttonState = digitalRead(buttonPin);
}
```

Example 2 – switching on/off LED

```
int ledPin = 11; // LED IS CONTROLLED BY PIN #11
int buttonPin = 8; // BUTTON IS CONNECTED TO PIN #8
int buttonState = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if(buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);}}}
```

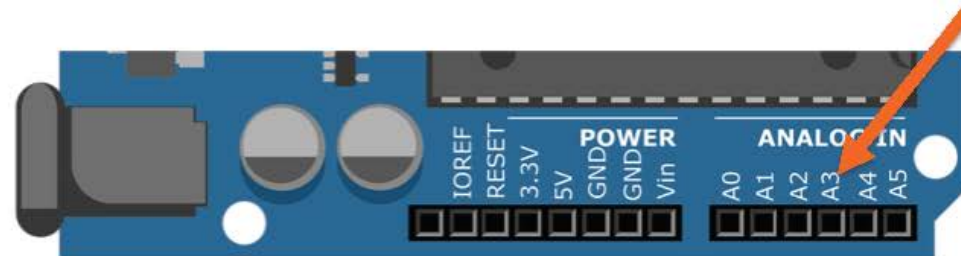
see also Arduino Examples/Digital/Button
(uses `LED_BUILTIN`)
→ try connecting `buttonPin` to GND

Example 2



Arduino – analog input

```
int sensorPin = 3; //SENSOR IS CONNECTED TO ANALOG IN A3  
int sensorValue = 0;  
void setup() {  
}
```



```
void loop() {  
  sensorValue = analogRead(sensorPin);  
  // sensorValue is now between 0 and 1023:  
  // 0V -> 0  
  // 5V -> 1023  
}
```



Aalto University
School of Science

Processing

Available from processing.org

What is Processing?

A flexible **software sketchbook** and a **language** for learning how to code within the context of the visual arts.

a subclass of the PApplet Java class

```
//Hello mouse.  
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
}  
  
void draw() {  
    line(150, 25, mouseX, mouseY);  
}
```

(a) language based on Java



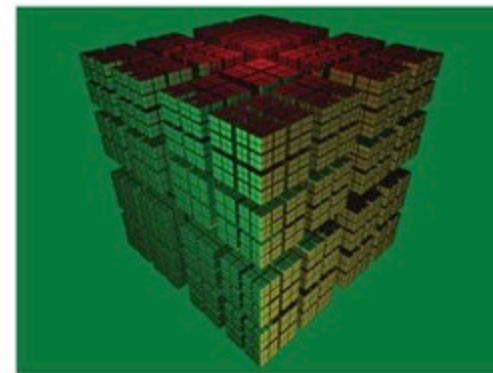
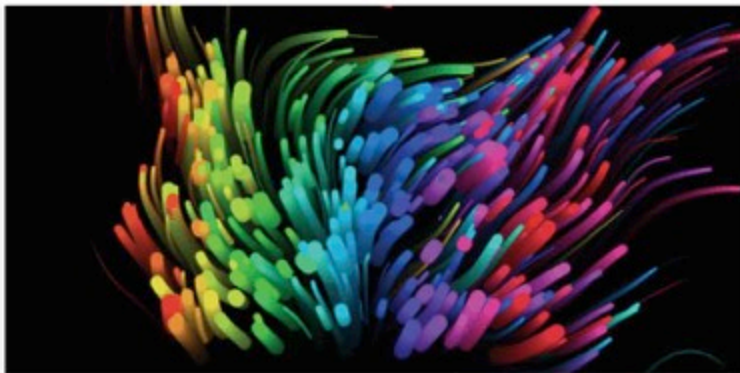
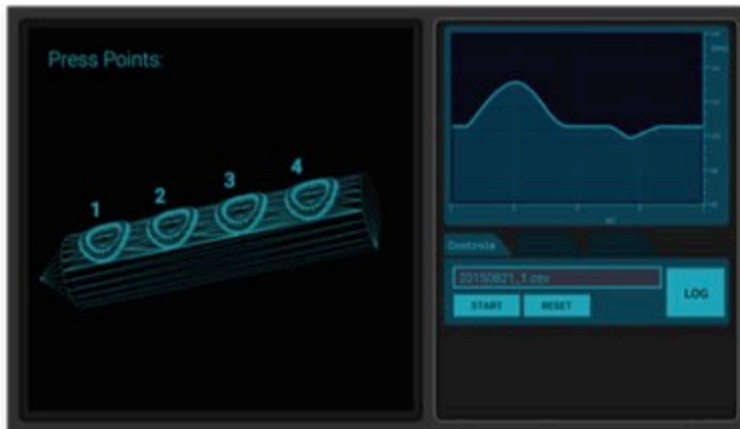
Software (IDE)
download from [processing](https://processing.org)

IDE

Similar to Arduino



2D & 3D

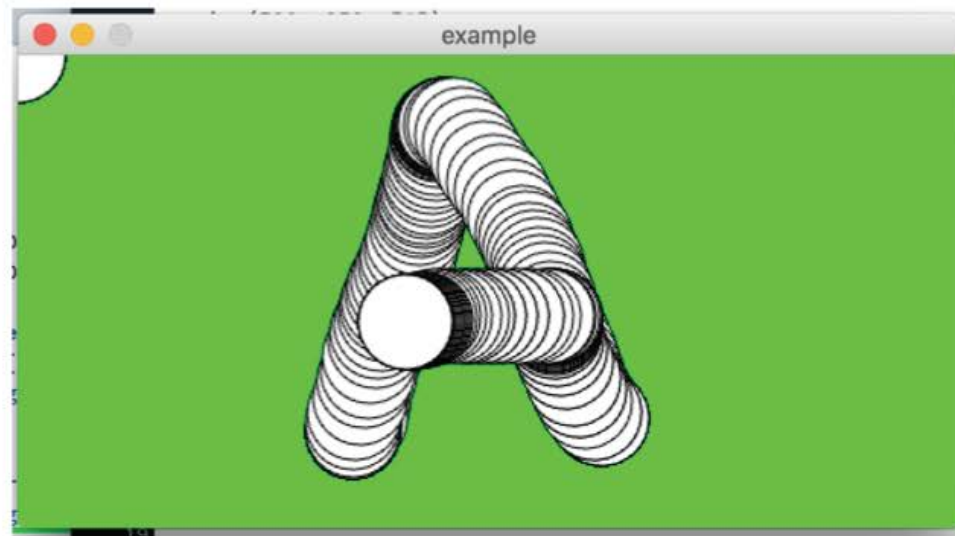


Inputs and Outputs

1. Mouse input events: `mousePressed`, `mouseClicked`, `mouseDragged`, `mouseMoved`, `mouseReleased`, `mouseWheel`.
2. Keyboard input events: `keyPressed`, `keyReleased`, `keyTyped`
3. Outputs in various form: Image, text file, XML ... etc.

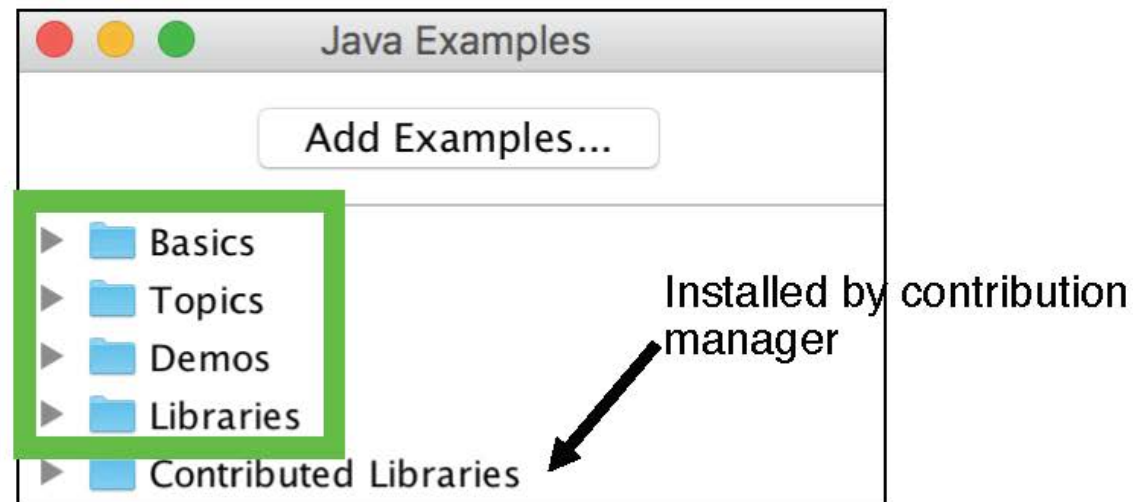
Inputs and Outputs

```
void draw() {  
  ellipse(mouseX, mouseY, 50, 50);  
}
```




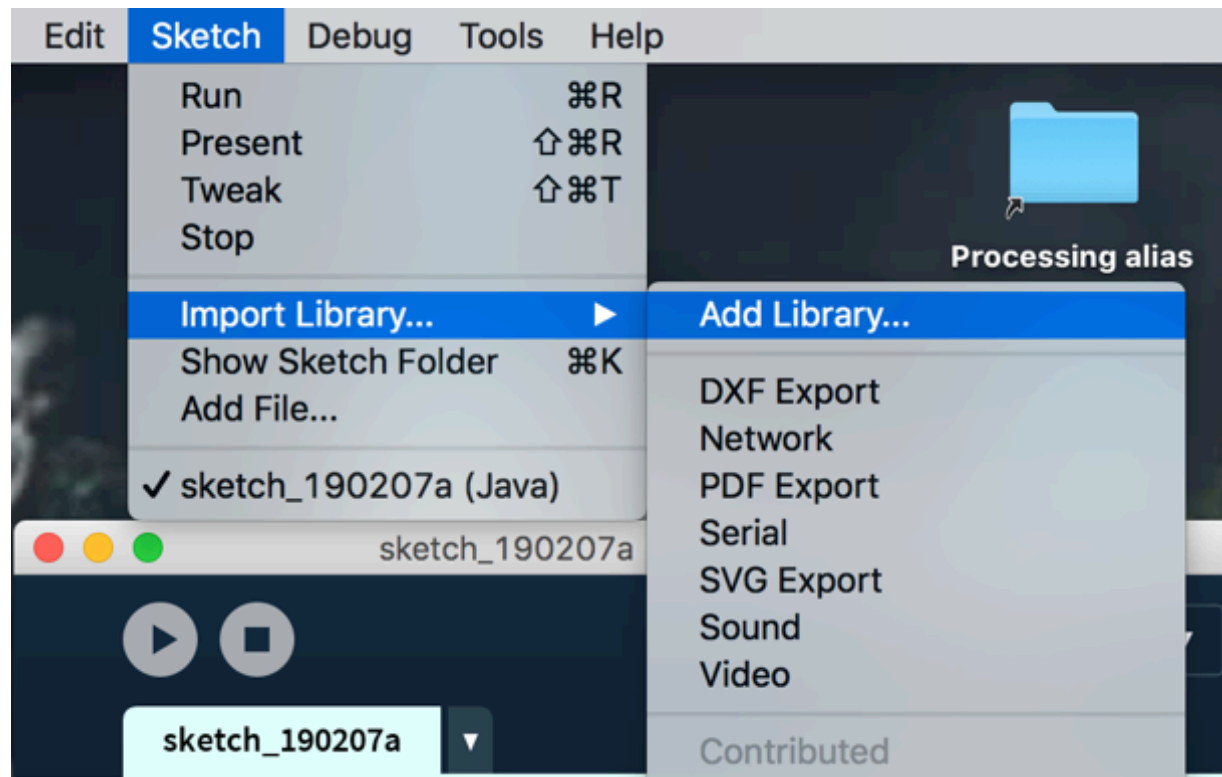
Example sketches

1. Default example sketches
 - demonstrating the most basic features,
 - File -> Examples...



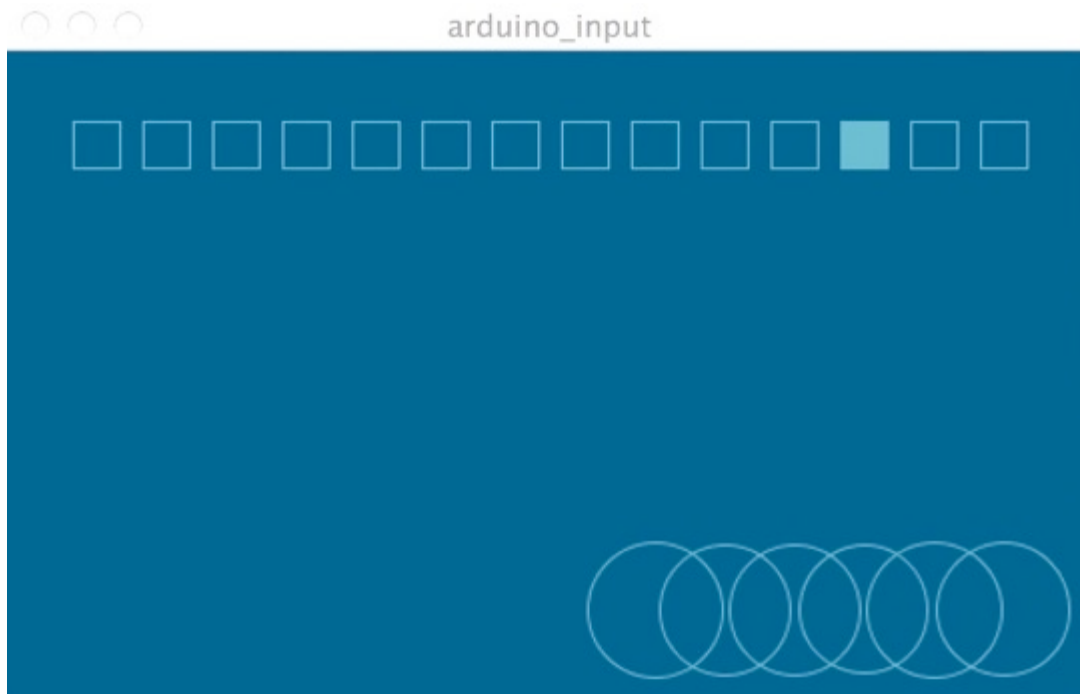
Arduino and Processing

- Need a communication firmware in Arduino
- Open example *StandardFirmata* in the IDE 
 - make sure the card&port selections are ok, run the program
- Download Arduino library in Processing



Testing Arduino in Processing

- Open example *arduino_input*
- Check the USB port name or number, correct the code if necessary (see comments in the program header)
- Run and test
 - input values should vary randomly (touch the card to see changes)



Easy development and debugging if the Arduino app is first simulated with Processing !

Hands-On Experiment: Measuring light

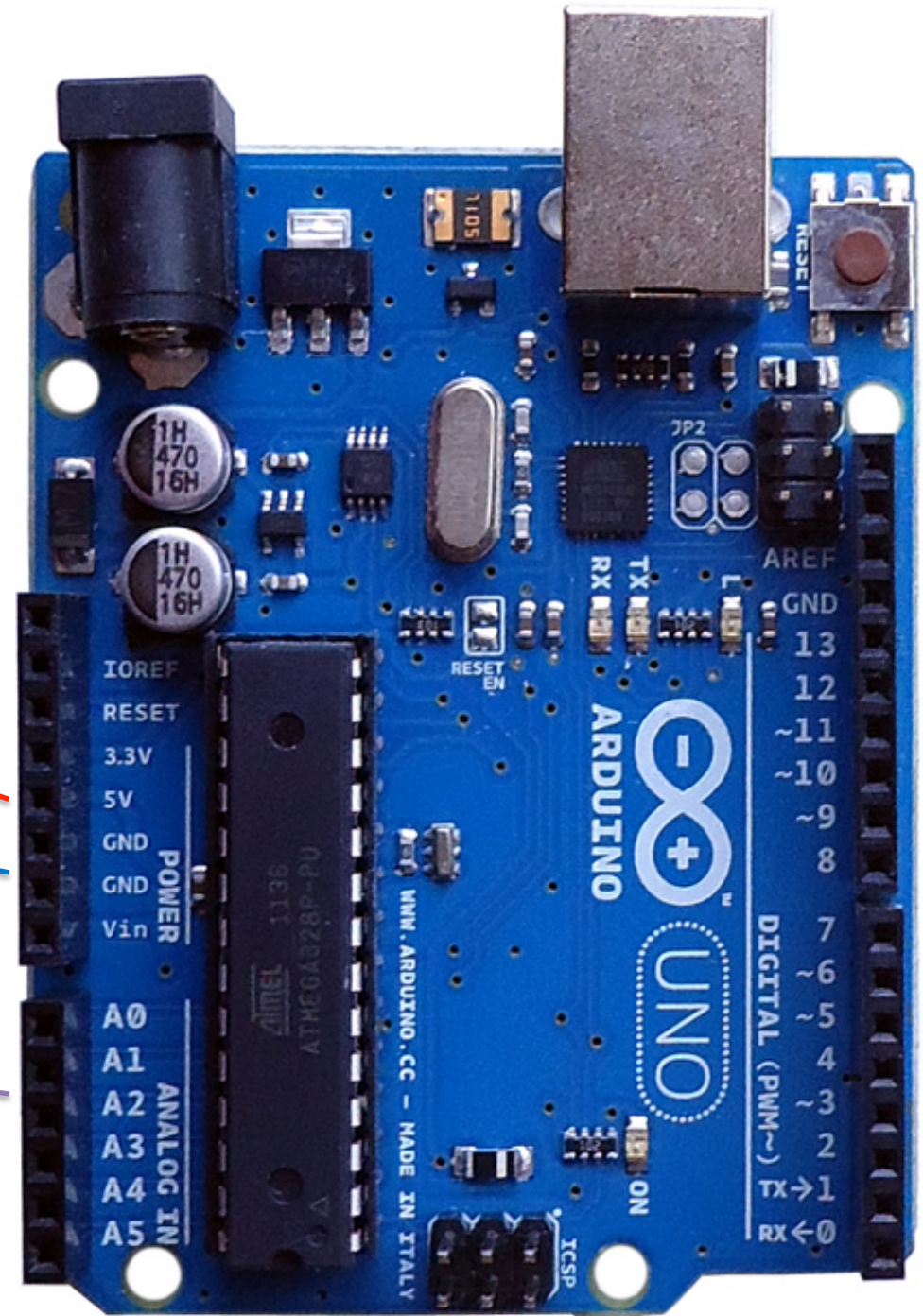
The connectors
needed:

(voltage)+5V

GND (- 0V)

A0...A5

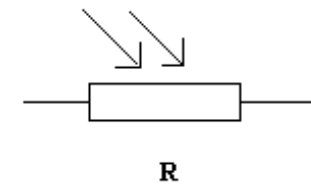
(analog input)



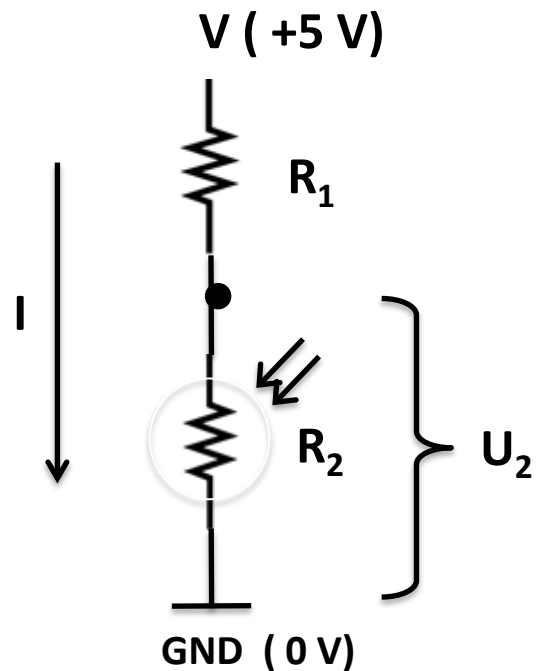
Using the sensor

- Many sensors are based on the change of electrical resistance of the material, eg. photo resistor (LDR)

– <https://en.wikipedia.org/wiki/Photoresistor>



- Resistance can be measured relative to a known resistance



Ohm's law:

$$I = V / (R_1 + R_2)$$

$$U_2 = I \cdot R_2 = V \cdot R_2 / (R_1 + R_2)$$

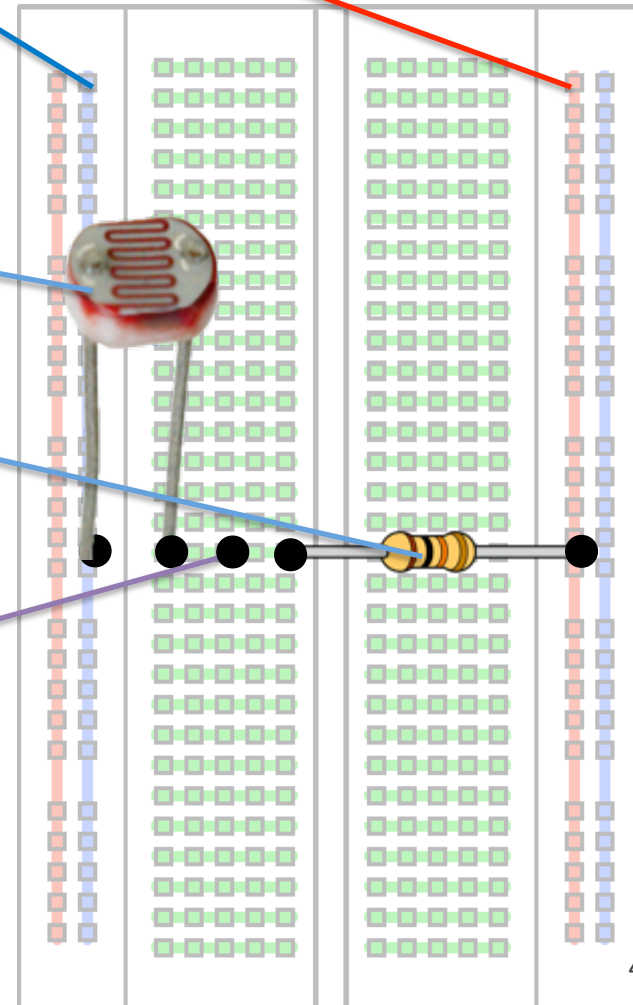
Configuration on the breadboard

- Voltage from Arduino (GND and +5V)

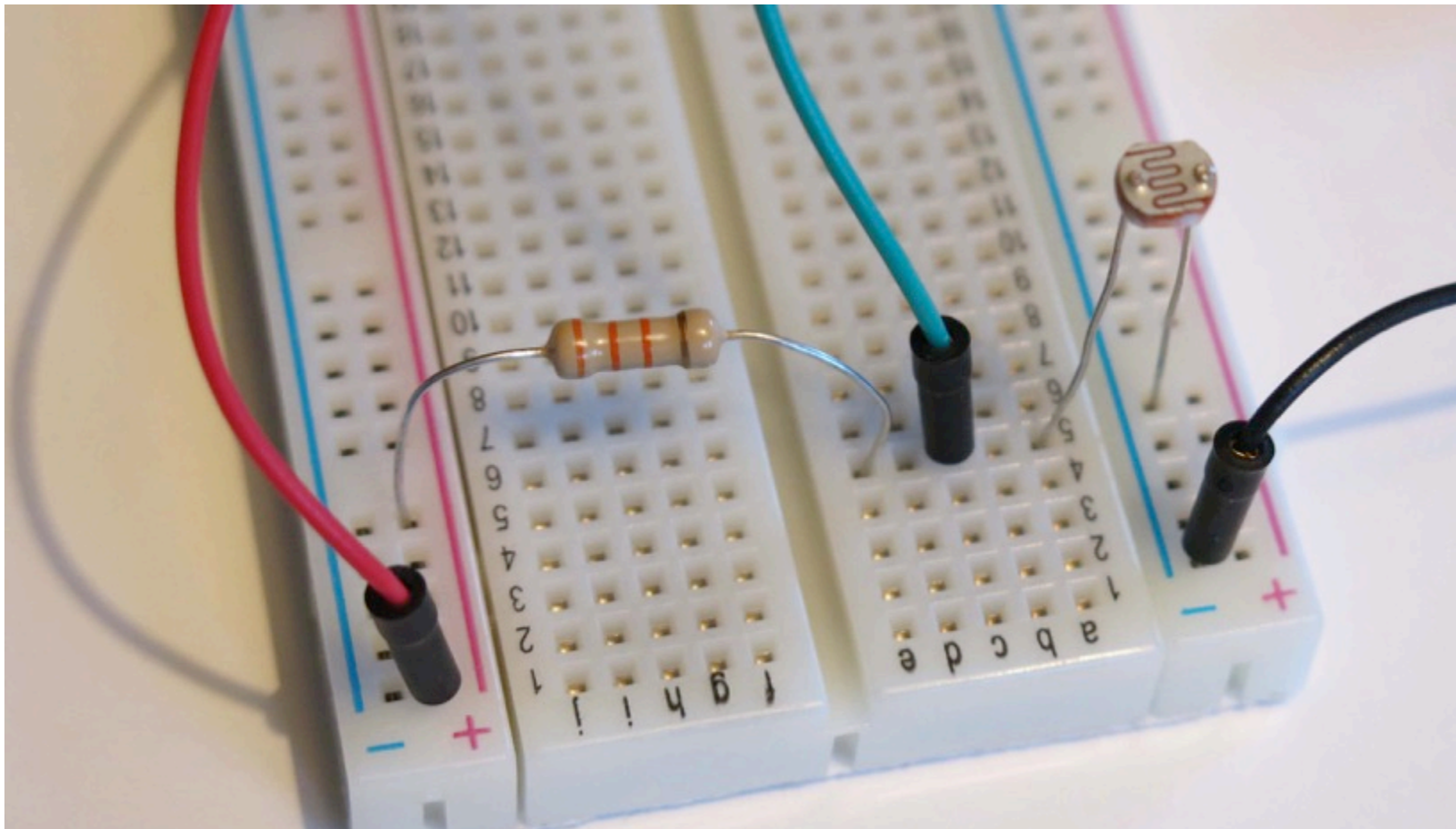
- Photoresistor from GND to a point...

...from which constant resistor to +5V

- Arduino's analog input (eg. A0) measures the voltage U_2 (0...5V scaled into range 0...1023)



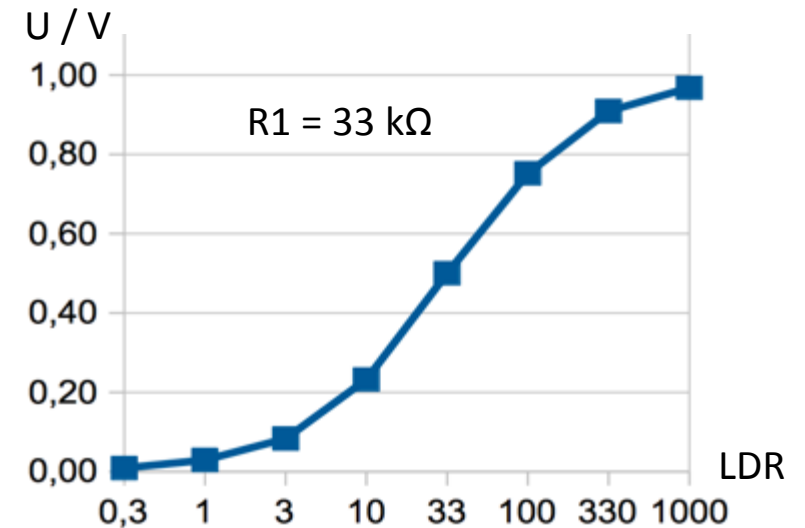
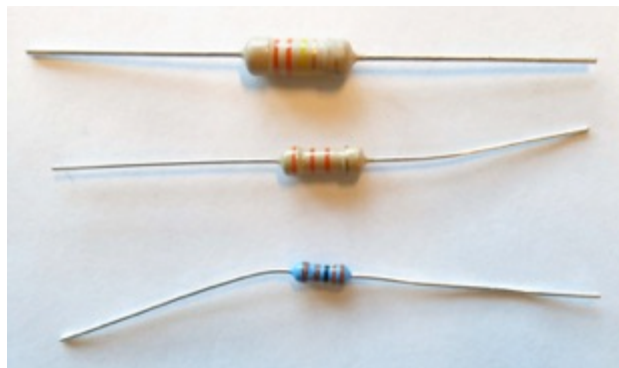
In Practice



Amount of light...

- ...varies a lot in practice, thus the LDR:n resistance may vary between 200 Ω ... 500 k Ω
- the value of constant resistance should be selected accordingly
 - best if the reference resistance equals that of the LDR in average lighting
- try out different values to find suitable

- 330 k Ω
- 33 k Ω
- 3.3 k Ω



Audio Signal Processing

- High sample rate → requires threading
 - "patch" ready-made control functions, or
 - manipulate the audio buffer directly with your own code
- Handled as streams connected to input/output/files
- Processing: *Sound* library (based on the older *Minim*)
 - poorly documented ☹ <https://processing.org/tutorials/sound/>
- Input
 - record *Sound/IO/AudioInput*
 - analyze *Amplitude, FFT*
- Synthesis *Noise, Oscillators*
- Files *SoundFile*
- Lots of effects for manipulation and playback

Using Video

- Check Processing examples in Libraries/Video/Capture
 - *GettingStartedCapture*
 - check that the camera works
 - *BackGroundSubtraction, Frame Differencing*
 - ways to separate static background from moving parts
- Finding an object of given color: *blobfinder*
 - code will be available in MyCourses

Internet is full of help

Programming

- . Google
- . Arduino
- . Instructables
- . Stack Exchange
- . W3 schools
- . Youtube

Hardware support

- . Google
- . Youtube
- . Instructables
- . Google
- . Adafruit
- . Sparkfun

Further reading



Next Steps

- Prepare your project plan
 - focus on the interaction (not only a fun application)
- Write a short description of
 - how the system will work (in principle) ?
 - what will be the user experience with it ?
 - what technology you plan to use (think about alternative solutions) ?
 - what work is needed and what are the roles of each in your group ?
- Present this next week in the pitching session

For Interaction Design: Think out of the box!

- Device abstraction: detach the physical device from its function (i.e. computational meaning)
- Same function can be realized in many ways
 - example: the program needs numerical input; digitally with keyboard or analogically with a slider?
- Realize input as abstract classes in your code
 - mapping to physical devices implemented as subclasses

Exercise: fill in the matrix

How about the mouse?

physical device:

information for the computer:

	keyboard	slider	function keys	drawing tablet	camera
text string (char)	native	?	?	?	?
real number (float)	?	native	?	?	?
choice (one out of few alternatives)	?	?	native	?	?
2D position (x,y)	?	?	?	native	?
object picked from screen (identifier/name)	?	?	?	?	?