



# Data Acquisition and Sensor Networks

Jacobs University Bremen  
Instructor: Fangning Hu

# Smart System





# Sensors

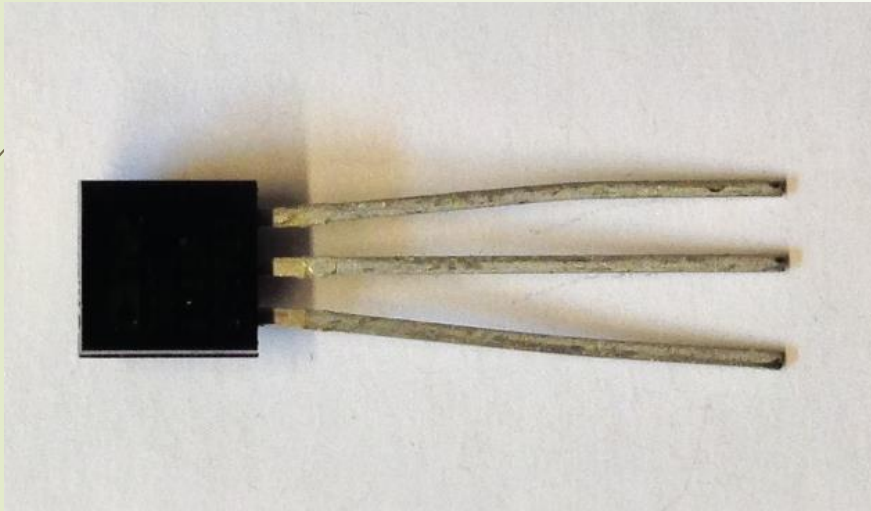
- Examples of sensors from biology: the human body
  - eyes: capture optical information (light)
  - ears: capture acoustic information (sound)
  - nose: captures olfactory information (smell)
  - skin: captures tactile information (temperature, texture)

Then convert gathered information into electro-chemical impulses in neurons.

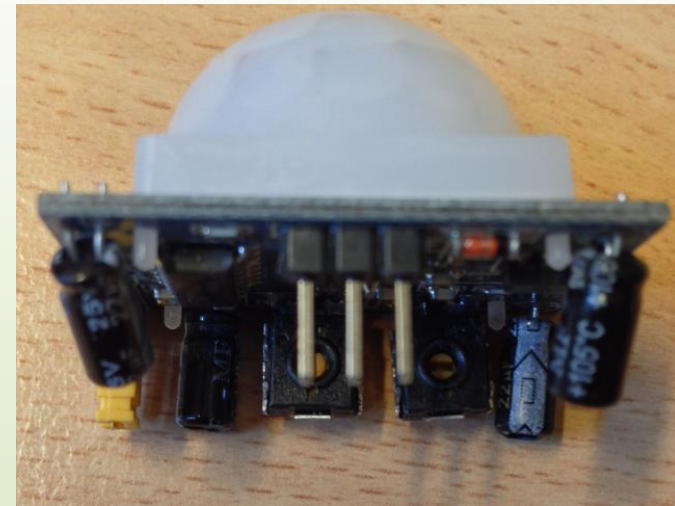


# Sensors in electronic world

- object converting one form of energy in the physical world into electrical energy



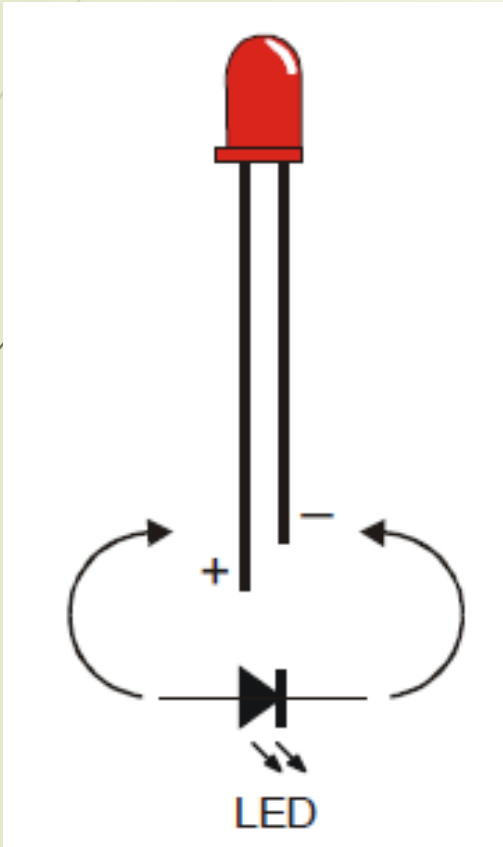
Temperature Sensor



Infra-Red Sensor

Convert temperature and Infra-Red light to voltages: 1.5V, 5V, ...

# Actuators (arms, legs, etc.)



Servo Motor



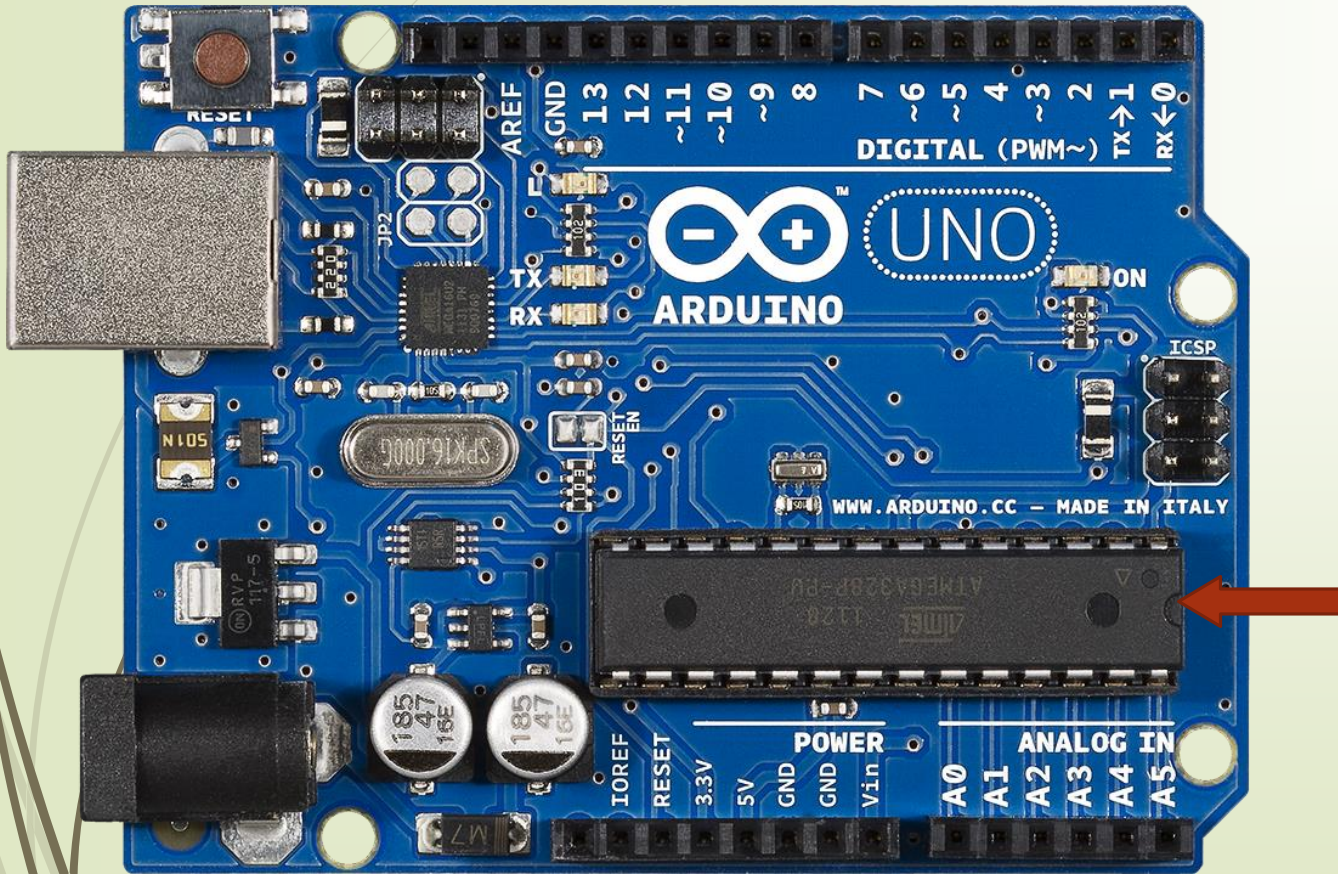
# Microcontroller

Functions:

- Read in sensor data
- Logic operation and computation
- Control actuators
- Communicate with computer

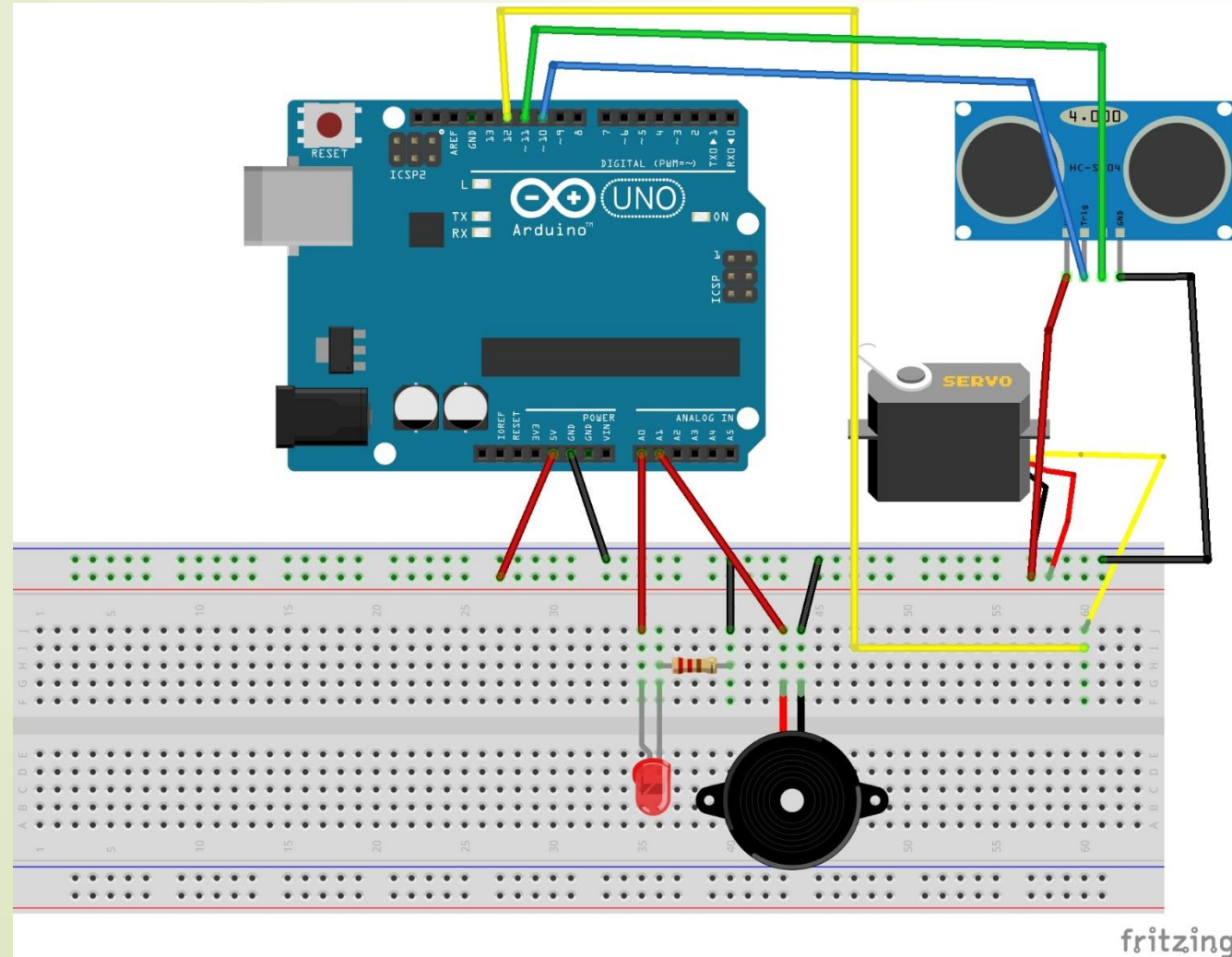
A small computer on a single integrated circuit:

- Input Output pins
- Memory
- Processor



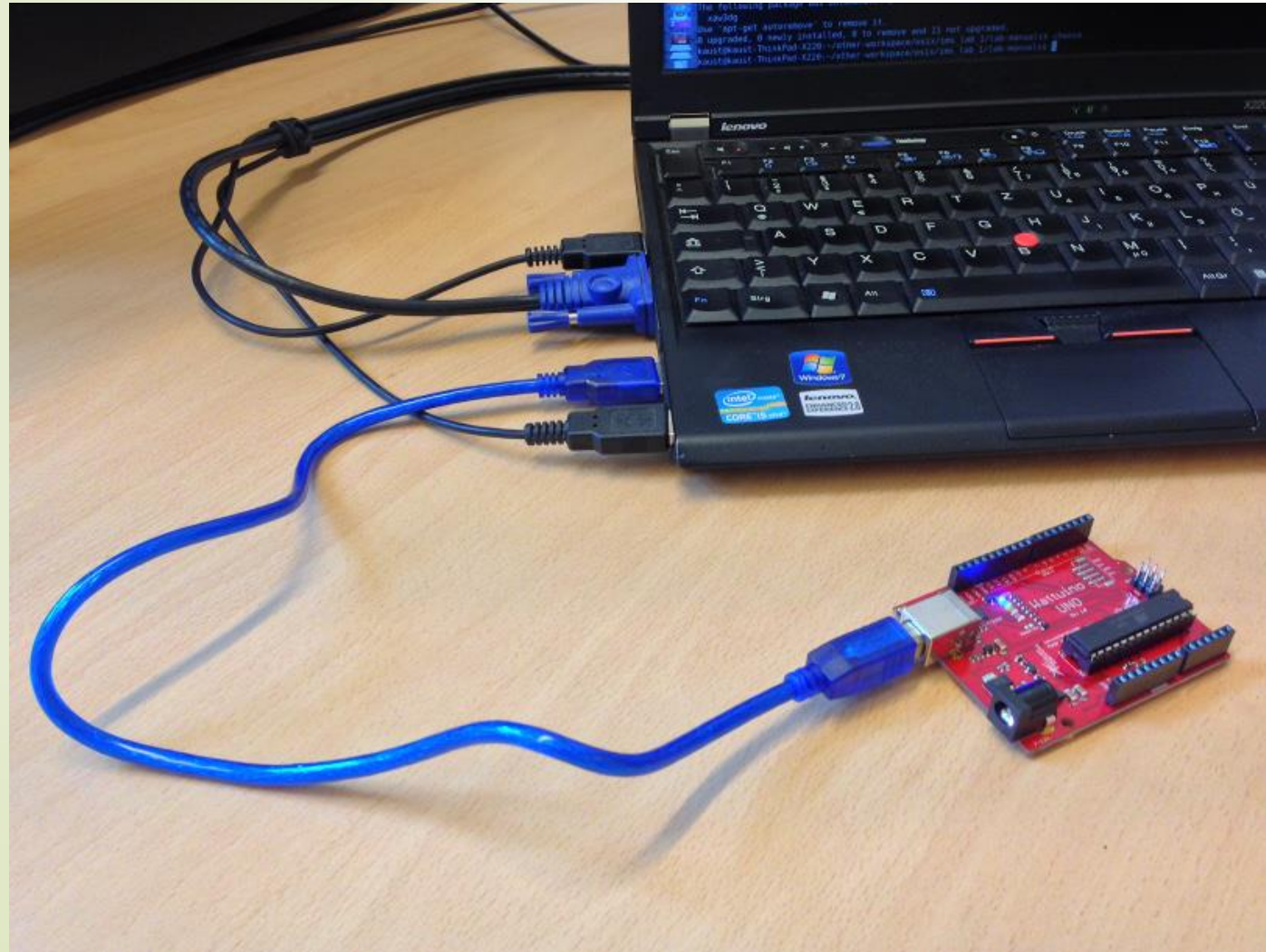
The Arduino UNO is based on the ATmega328P microcontroller

# Program Arduino to achieve tasks

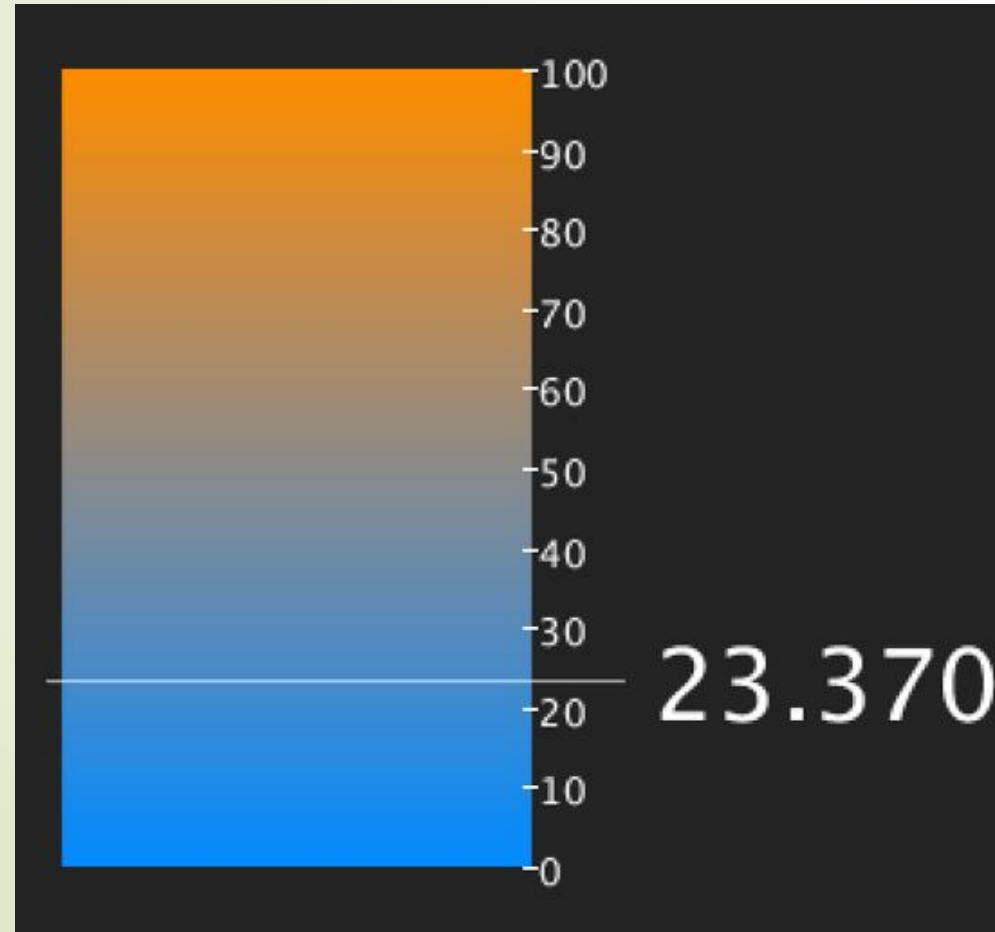




# Communicate to computer



# Display Data on the Computer



# Remote Control





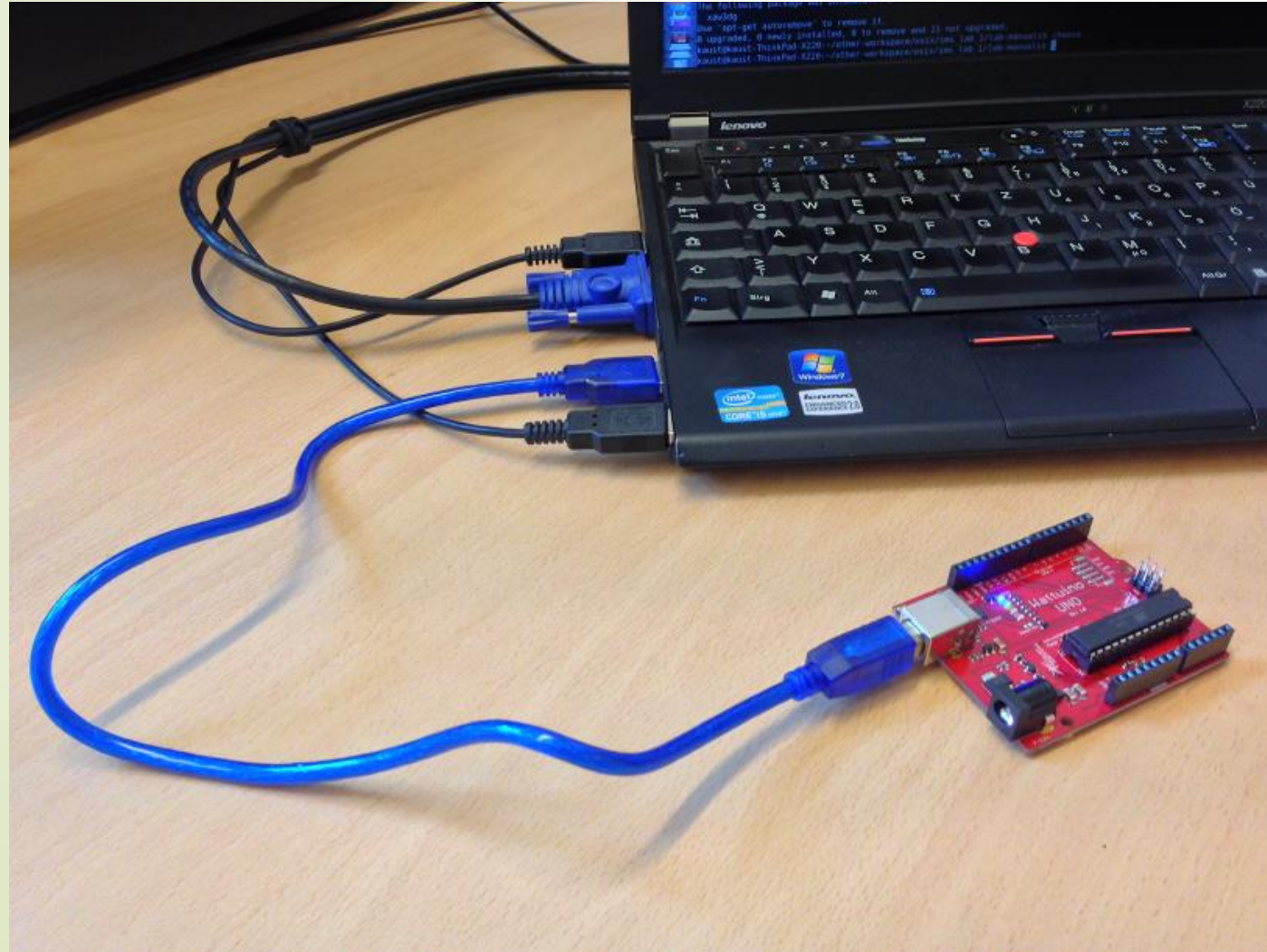
# Schedule



- Lab1: Learn basic concepts of resistors, multimeter, LED, Potentiometer, basic program of Arduino
- Lab2: Learn how to collect data by different sensors (LDR, temperature sensor, PIR, Ultrasonic distance measurement) and control different actuators (Piezo buzzer, Servo)
- Lab3: Learn how to visualize the sensor data on computer by program – “Processing”
- Lab4: Learn how to set up local wireless network
- Lab5: Learn website technologies and database
- Lab6: Learn how to connect the local network to internet
- Lab7: Final project. Design your own system based on what you learn









# Start: Connect Arduino to computer





# The Arduino IDE



Button	Description
	Compile and check errors
	Upload codes to Arduino
	Create a new sketch
	Open an existing sketch
	Save the current sketch
	Open the serial monitor



# Download Material

Campusnet

**Manual**

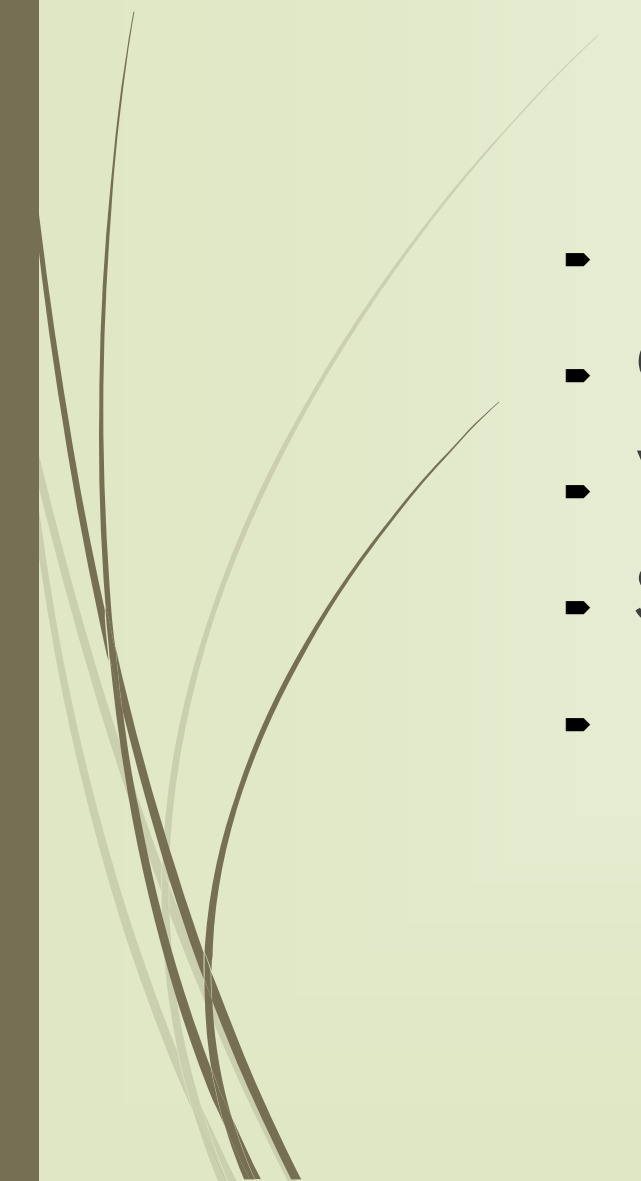
**Slide**

**Reference**

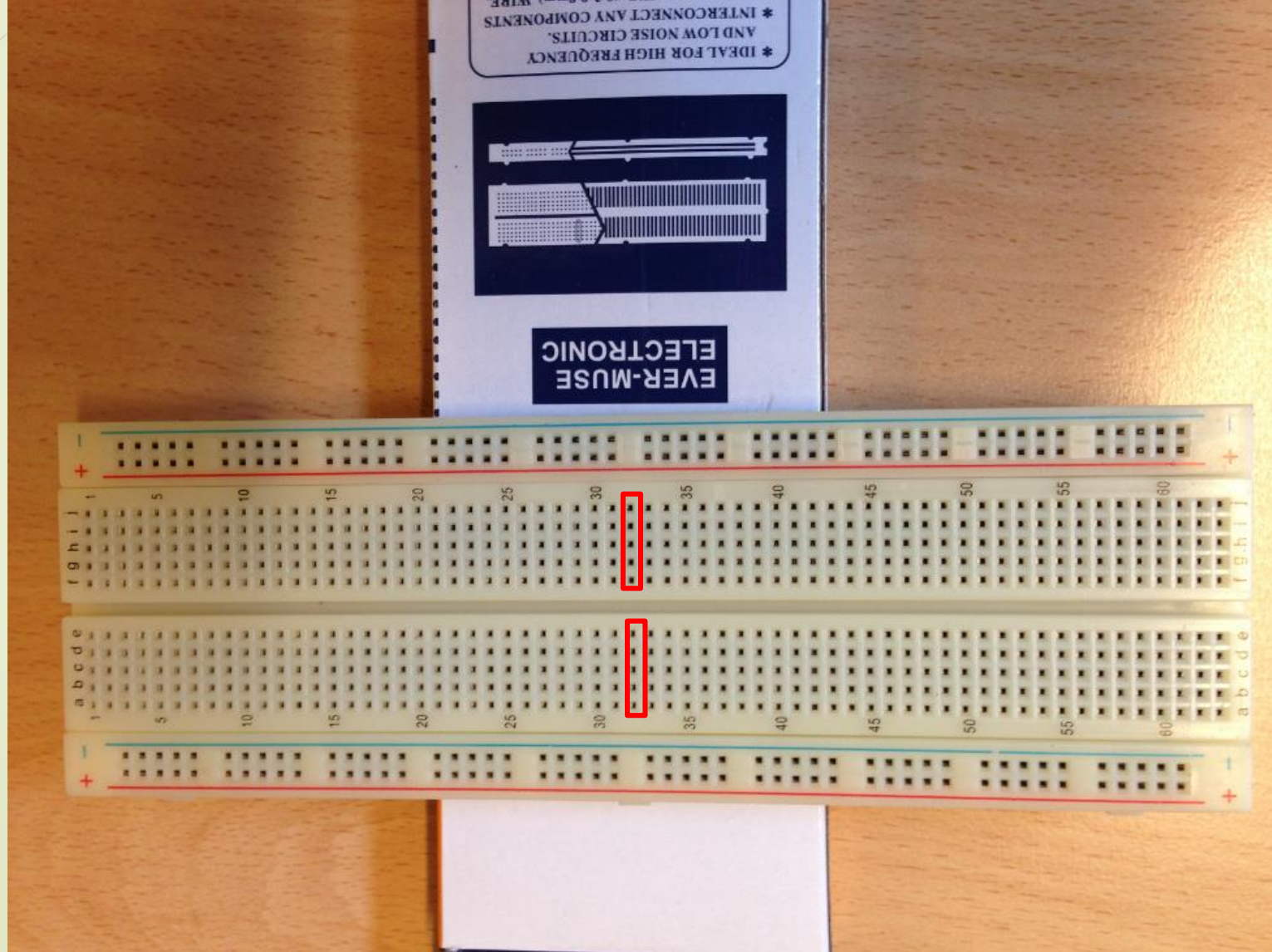




# Basic electronic you need in the lab

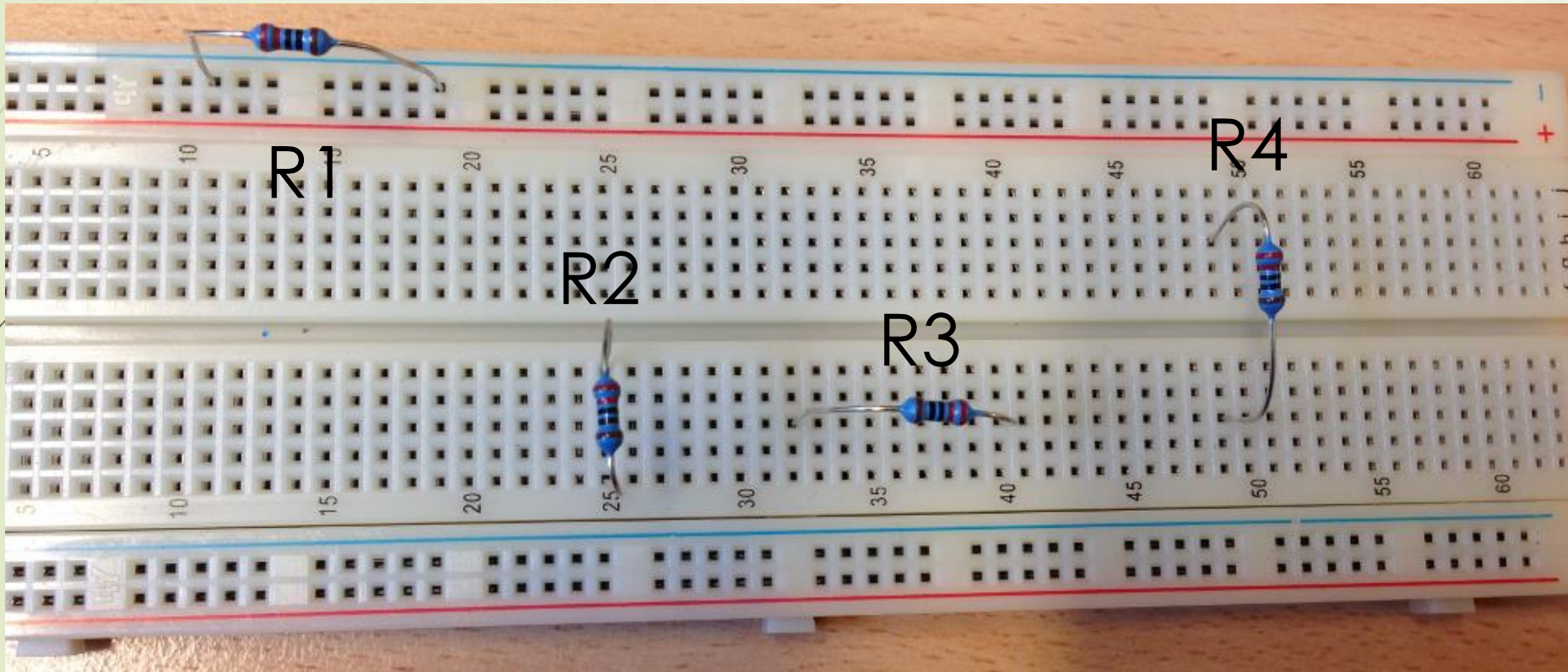
- Using a breadboard
  - Ohm's law:  $V=IR$
  - Voltage-Divider
  - Safety issues
  - Using a multimeter
- 

# Breadboard



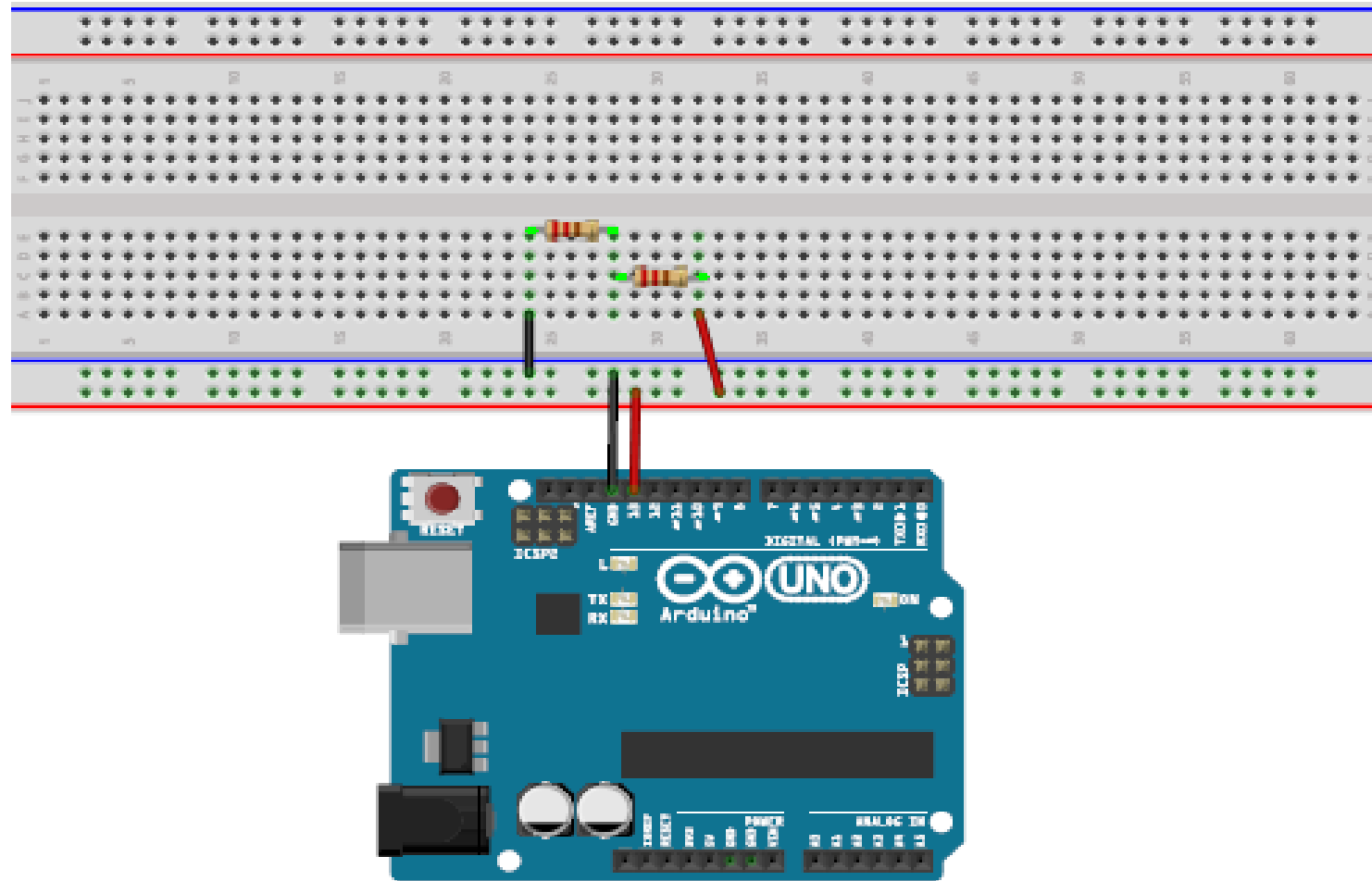


# Which resistors are correctly connected

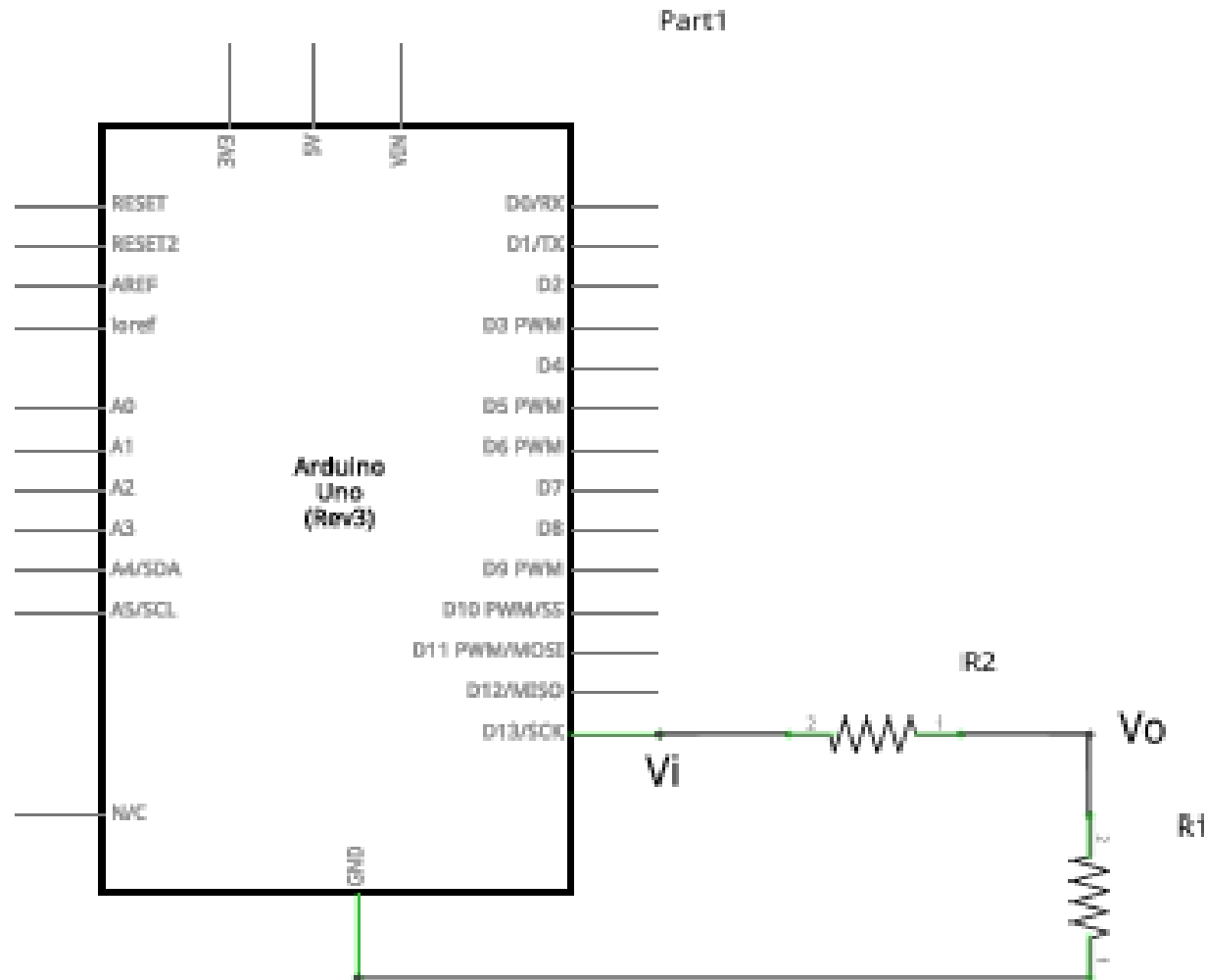


R3 and R4 are correctly placed.  
The ends of R1 and R2 are shorted, which could be dangerous.





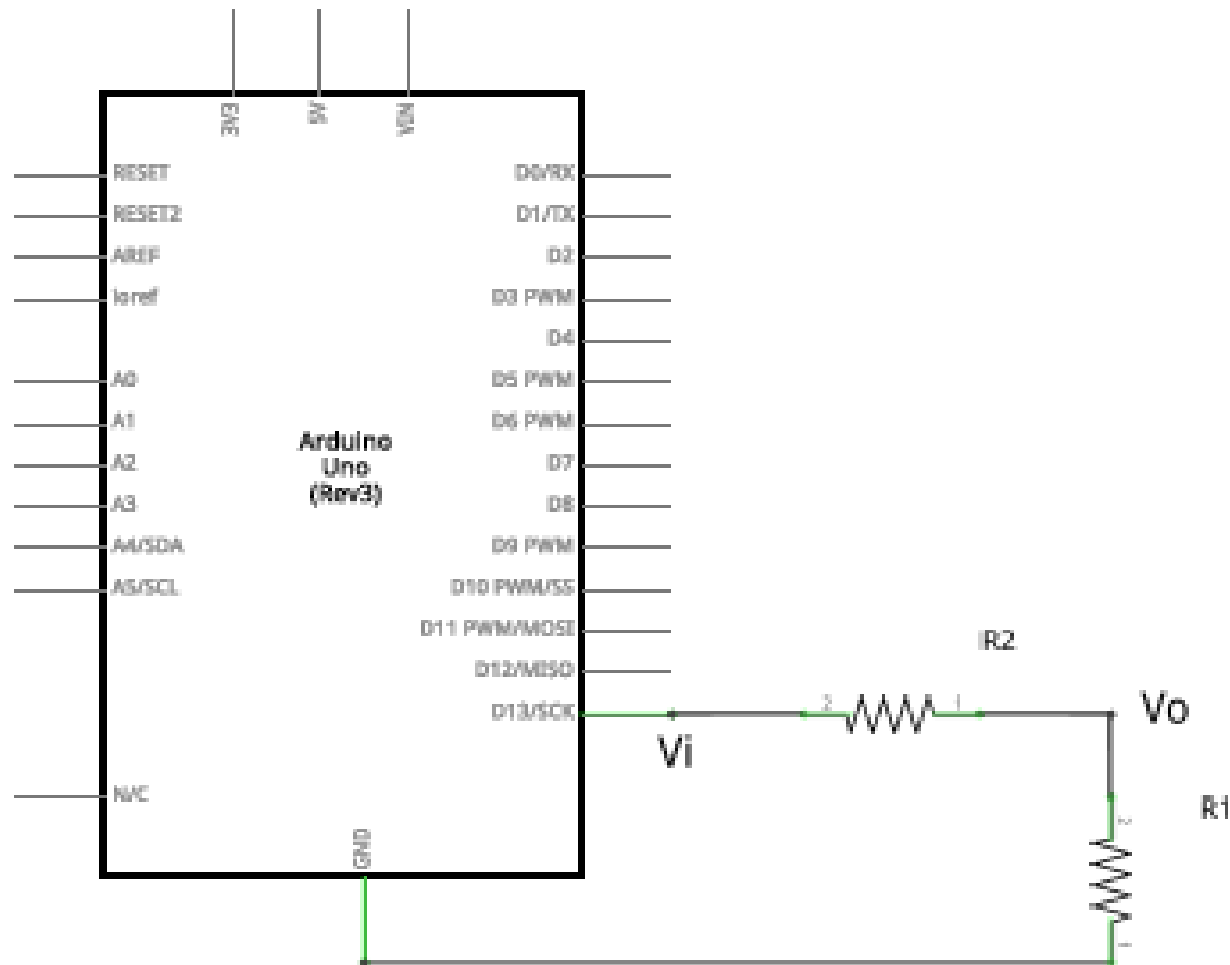
The Arduino provide 5V voltage supply. The safety current is 20mA, what is the requirement for the resistors in between? How large should they be?



$$R1 + R2 \geq \frac{V}{I} = \frac{5}{20 \times 10^{-3}} = 250\Omega$$

The voltage supplied by an Arduino digital OUTPUT pin is 5V. The safety current is 20mA, there should be a **250Ω** between an OUTPUT pin and the ground.

Part1



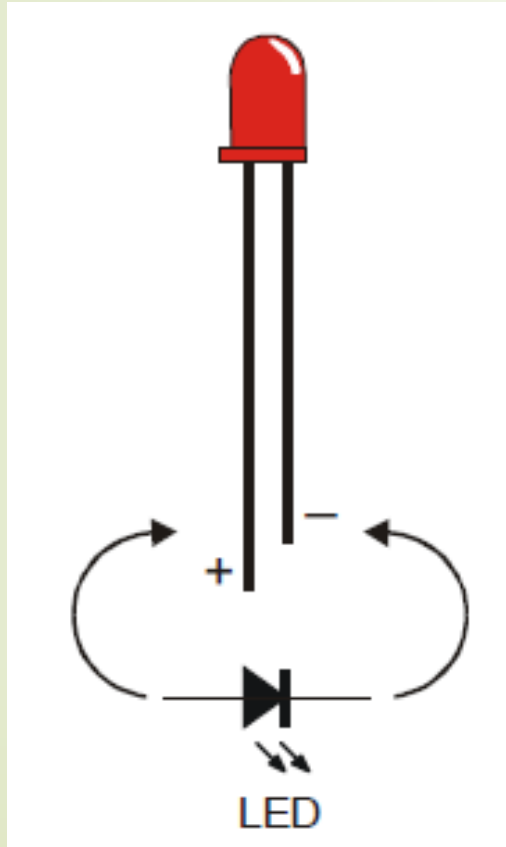
## The Voltage-Divider Circuit

$$I = \frac{V}{R_1 + R_2}$$

$$V_0 = I \cdot R_1 = V \cdot \frac{R_1}{R_1 + R_2}$$

We can divide  $V_i$  in the proportion of resistance.

# A Light Emitting Diode (LED)



The longer lead is the anode (+) and the shorter is the cathode (-). The anode should be connected to the more positive voltage.

# Recommended Resistor Values

To avoid burning out the LED due to excessive current, a resistor is always needed.

LED						White	Red	Yellow	Green	Blue
LED	White	Red	Yellow	Green	Blue	3.3V	2.1V	2.2V	3.7V	3.1V
$V_{Forward}$	3.3V	2.1V	2.2V	3.7V	3.1V					
Resistance										
Resistance						100 $\Omega$	200 $\Omega$	200 $\Omega$	100 $\Omega$	100 $\Omega$







# The Blink.ino

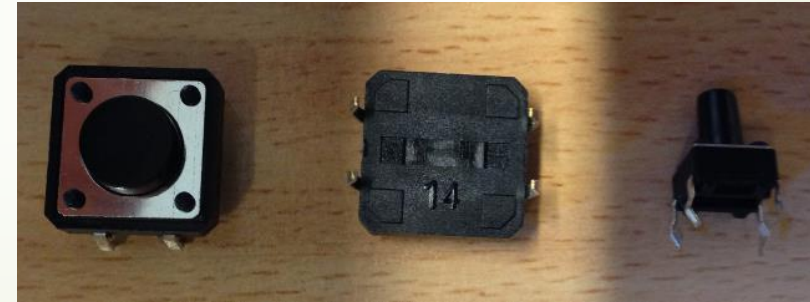
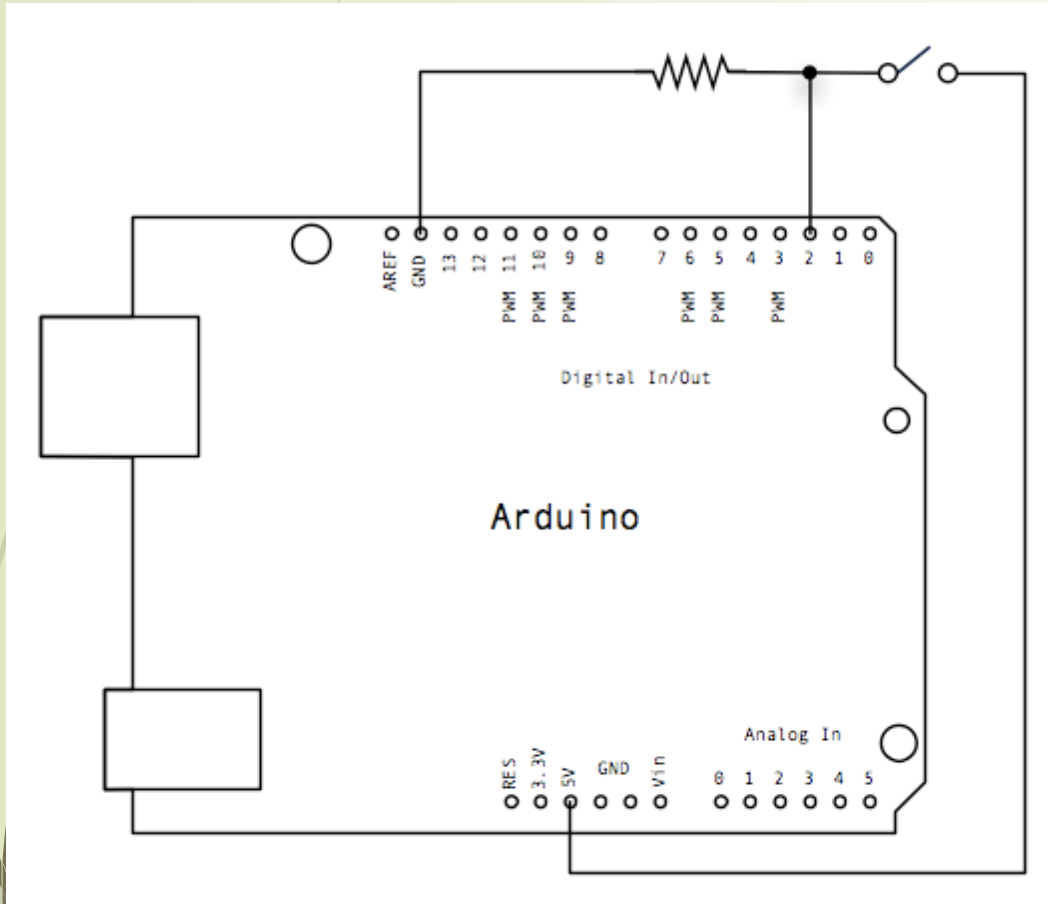
```
void setup() { // Called once to initialize
  pinMode(13, OUTPUT); // Initialize pin 13 for
                        // digital-write
}

void loop() { // Called repeatedly
  digitalWrite(13, HIGH); // Set pin 13 to 5V
  delay(1000);             // Wait 1 sec = 1000 millisec.
  digitalWrite(13, LOW);  // Set pin 13 to 0V
  delay(1000);
}
```



This is the built-in LED connected internally to pin 13. It has its own built-in resistor. We are going to make the same circuit on a breadboard.

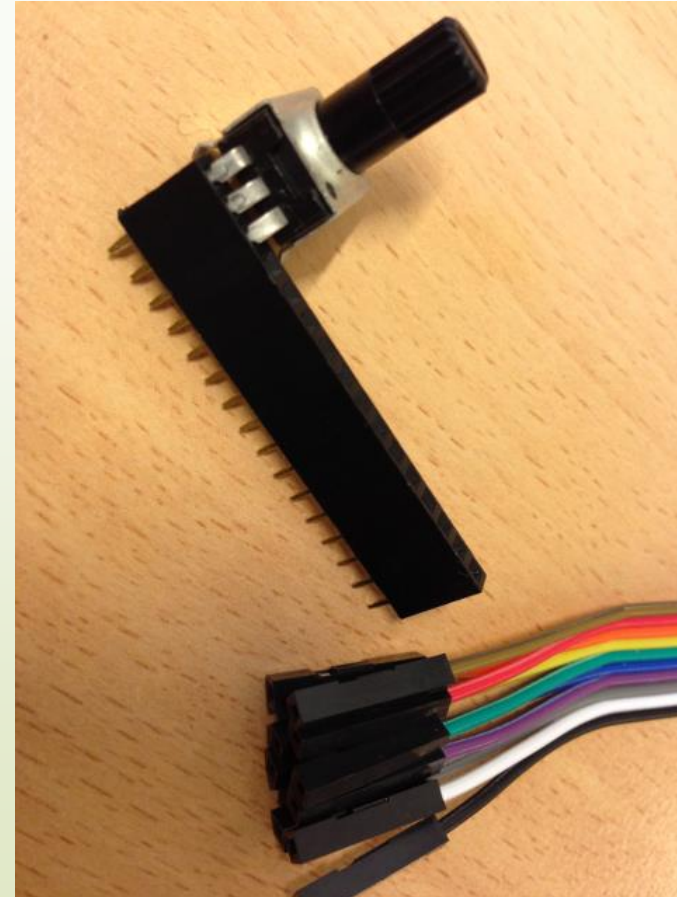
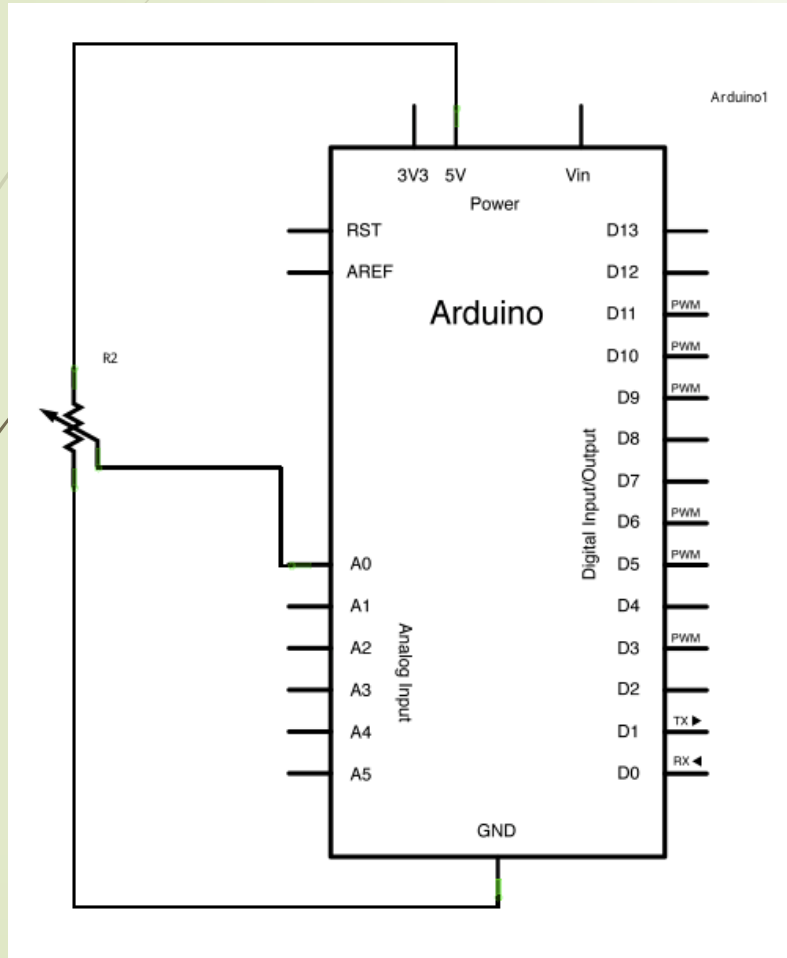
# Button and Digital Read



```
pinMode ( 2 , INPUT );
```

```
buttonState = digitalRead (2 );  
if ( buttonState == HIGH ) { // turn LED on:  
  digitalWrite (13 , HIGH );  
}  
else { // turn LED off:  
  digitalWrite (13 , LOW );  
}
```

# Potentialmeter and AnalogRead

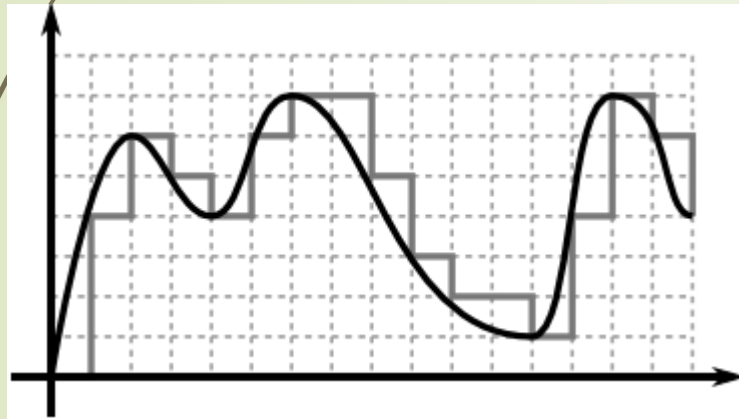




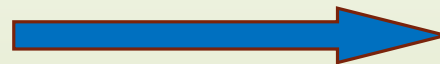
# Analog-to-Digital Conversion

- The gathered data such as voltages are in continuous form --- analog signal
- Computer can only process digital data, i.e., 0, 1 sequence

Analog Signal: A0=0V~5V



Analog-to-Digital  
Conversion



```
int sensorValue= analogRead(A0);
```

Digital Data: SensorValue

Binary	: Decimal
0000000000	: 0
0000000001	: 1
0000000010	: 2
0000000011	: 3
....	: ...
1111111111	: 1023



# Serial Monitor

```
void setup () { // initialize serial communication at 9600 bits per second :  
Serial.begin (9600) ;  
}
```

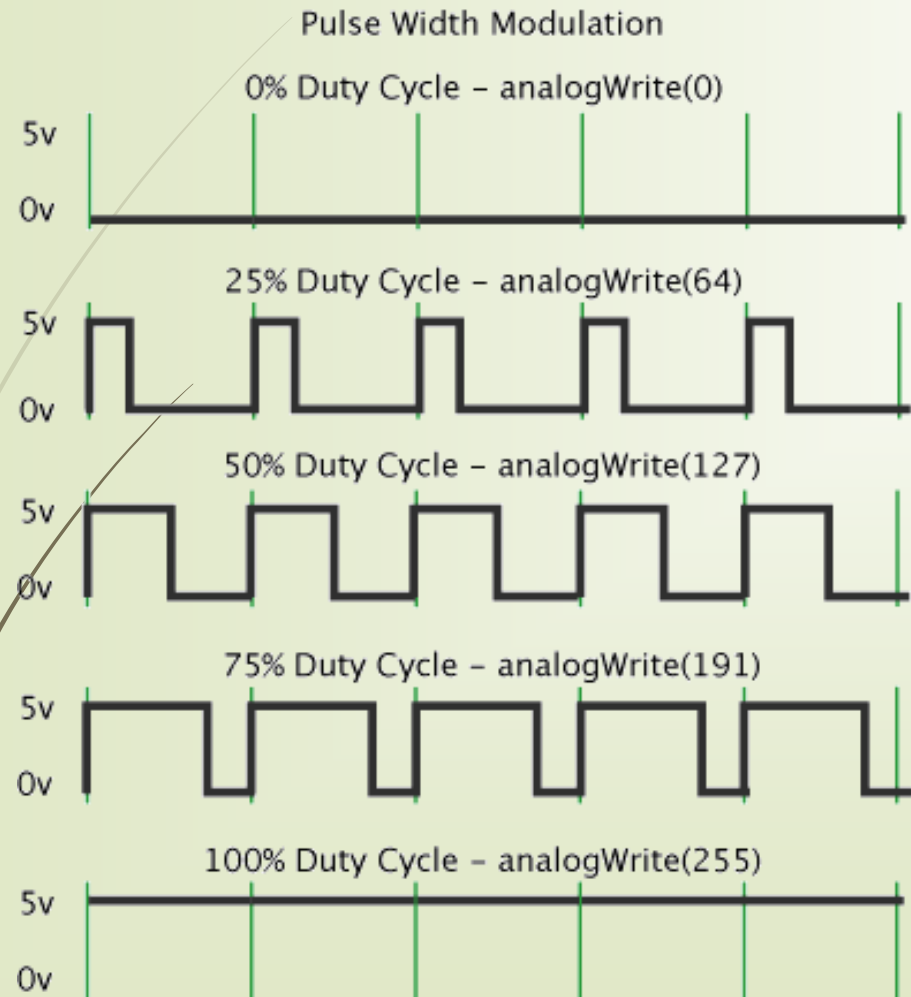
```
// the loop routine runs over and over again forever :  
void loop () {
```

```
// read the input on analog pin 0:  
int sensorValue = analogRead (A0);
```

```
// Convert the analog reading ( from 0 - 1023) to a voltage (0 - 5V):  
float voltage = sensorValue * (5.0 / 1023.0) ;
```

```
// print out the value you read :  
Serial.println ( voltage );  
}
```

# Digital-to-Analog Conversion

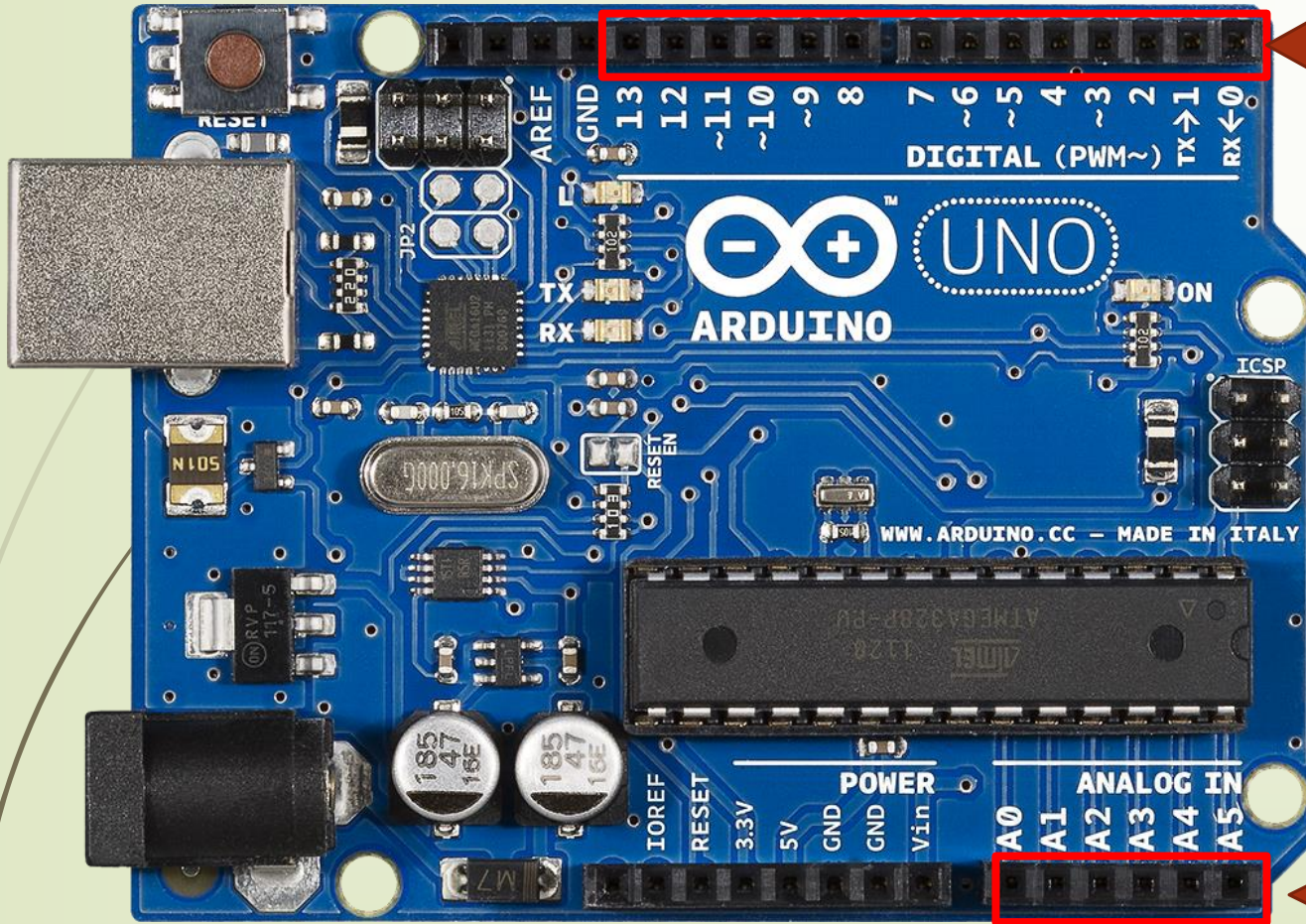


- To output “analog” signal by digital means
- It is called Pulse Width Modulation (PWM)

`analogWrite(pin, value);`

Value : 0 ~ 255

Output: 0V ~ 5V

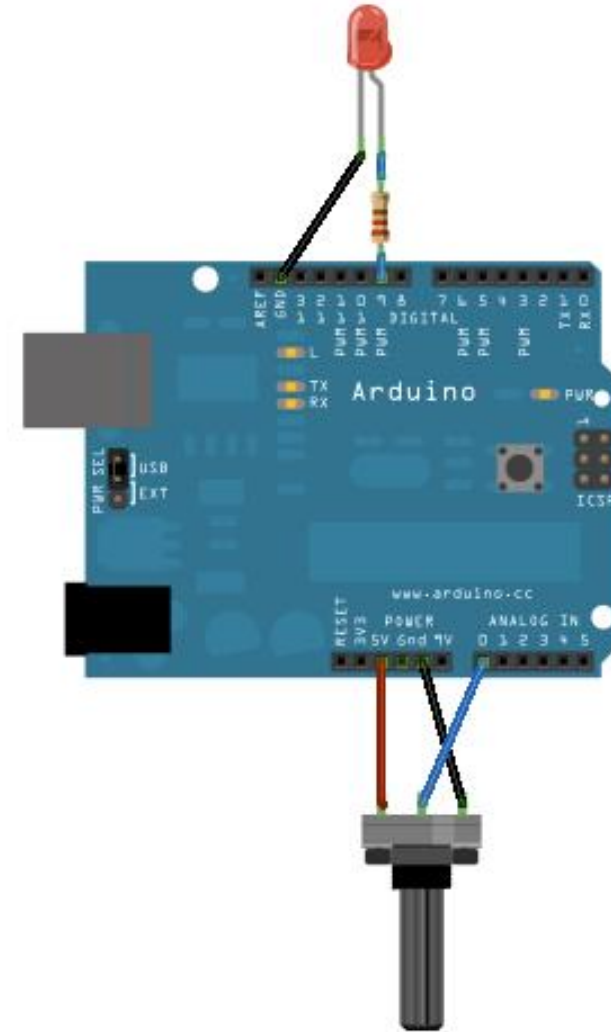


14 Digital I/O pins

6 PWM output pins,  
marked with ~

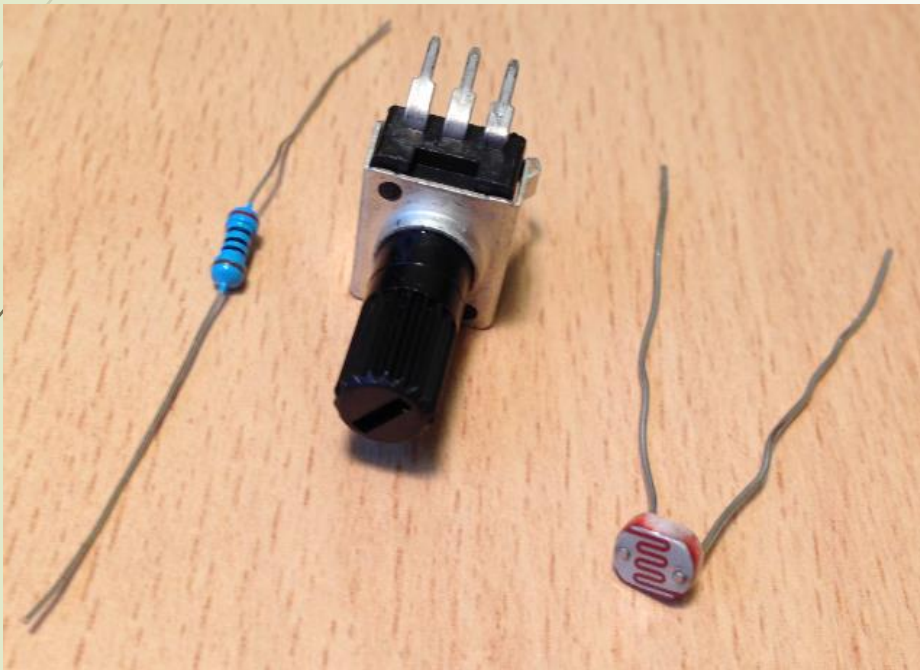
6 Analog Input pins

## Use Potentialmeter to control The Brightness of the LED





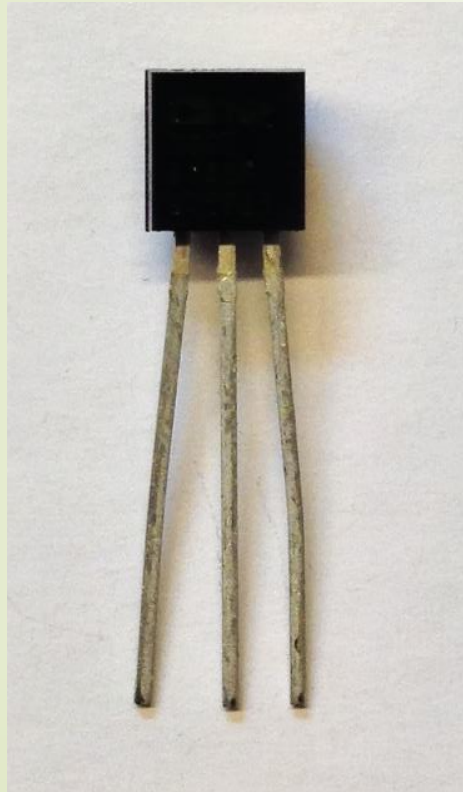
# Light Dependent Resistor



## Applications:

- Streetlight (Turn on when it is dark, save energy)
- Camera Light meters
- Night Clocks
- Alarm Devices (Detector for a light beam)

# Temperature Sensor

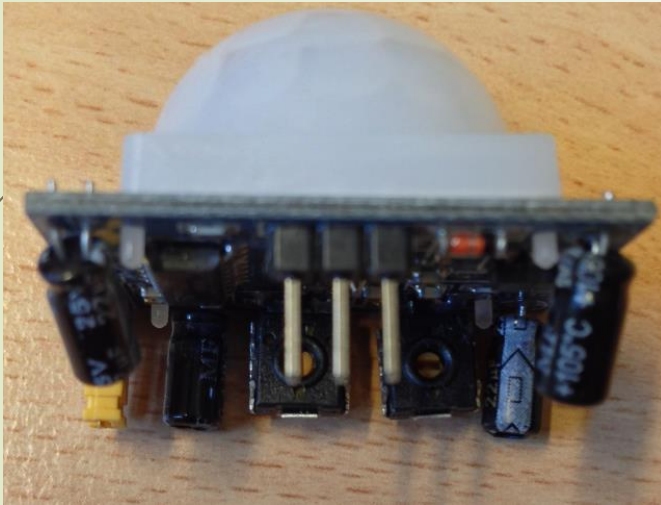


## Applications:

- Medical Application (Measure body temperature)
- Biology research (Temperature collection and control)
- Electrical Products (Avoid overheating)
- Materials research (Temperature collection and control)



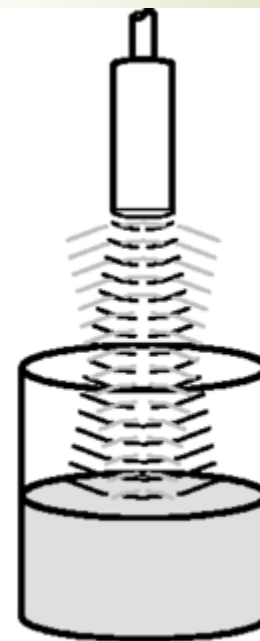
# Passive Infra-Red Sensor for Motion Detection



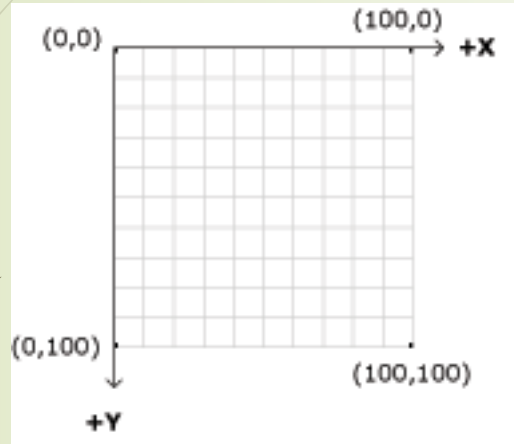
## Applications:

- Security Alarms
- Automatic Lighting systems
- Animal observation systems

# Ultrasonic Distance Sensor



# Processing



`line(x1, y1, x2, y2)`  
`triangle(x1, y1, x2, y2, x3, y3)`  
`quad(x1, y1, x2, y2, x3, y3, x4, y4)`  
`rect(x, y, width, height)`  
`ellipse(x, y, width, height)`  
`arc(x, y, width, height, start, stop)`

```
size(400,200);  
noStroke(); // no stroke  
smooth(); //smooth  
background(0,26,51); //set background to be dark blue  
fill(255,0,0); // set color to be red  
ellipse(132,50,150,150); // draw a circle
```

Task 2.1, Task 2.2: Draw a car



# Processing

```
float x = 60;
float y = 60;

int diameter = 60;

void setup() {
  size(480, 360);
  frameRate(30);
}

void draw()
{
  background(102);
  x = x+2.8;
  y = y+2.2;
  ellipse(x, y, diameter, diameter);
}
```

Task 2.3 solution: <https://processing.org/examples/bounce.html>

## Button Control the Color (Arduino Code)

```
// Code for sensing a switch status and writing the value to the serial port.

int switchPin = 4;           // Switch connected to pin 4
char temp = 0;
void setup() {
    pinMode(switchPin, INPUT);    // Set pin 0 as an input
    Serial.begin(9600);           // Start serial communication at 9600
    bps
}

void loop() {
    if (digitalRead(switchPin) == HIGH) { // If switch is ON,
        temp = 1;                        // send 1 to Processing
    } else {                             // If the switch is not ON,
        temp = 0;                        // send 0 to Processing
    }
    Serial.print(temp);    //send the value of temp to serial port
    delay(100);            // Wait 100 milliseconds
}
```

## Button Control the Color (Processing Code)

```
import processing.serial.*;

Serial port;           // Create object from Serial class
int val;               // Data received from the serial port

void setup() {
  size(200, 200);
  // Open the port that the board is connected to and use the same speed (9600
  bps)
  port = new Serial(this, "COM3", 9600);
  // The port name may be different on your computer from "COM3",
}

void draw() {
  if (0 < port.available()) {           // If data is available,
    val = port.read();                  // read it and store it in val
    println(val);                       // if the type of val is char, use port.
    readChar()
  }

  background(255);                      // Set background to white
  if (val == 0) {                       // If the serial value val is 0
    fill(0);                            // set fill to black
  }
  else {                                // If the serial value is not 0,
    fill(204);                          // set fill to light gray
  }
  rect(50, 50, 100, 100);
}
```



## Mouse Control the LED (Arduino Code)

```
const int ledPin = 9;      // the pin that the LED is attached to

void setup()
{
    // initialize the serial communication:
    Serial.begin(9600);
    // initialize the ledPin as an output:
    pinMode(ledPin, OUTPUT);
}

void loop() {
    int brightness;

    // check if data has been sent from the computer:
    if (Serial.available()) {
        // read the most recent byte (which will be from 0 to 255):
        brightness = Serial.read();
        // set the brightness of the LED:
        analogWrite(ledPin, brightness);
    }
}
```

## Mouse Control the LED (Processing Code)

```
import processing.serial.*;

Serial port;                                // Create object from Serial class
int val;                                    // Data received from the serial port

void setup() {
    size(200, 200);
    // Open the port that the board is connected to and use the same speed (9600 bps
    )
    port = new Serial(this, "COM3", 9600);
    // The port name may be different on your computer from "COM3",
    // you can use println(Arduino.list()); to check the port name
}

void draw() {
    // draw a gradient from black to white
    for (int i = 0; i < 256; i++) {
        stroke(i);
        line(i, 0, i, 150);
    }

    // write the current X-position of the mouse to the serial port as
    // a single byte
    port.write(mouseX);
}
```

## About Serial.println()

- `Serial.println(„23“)` will treat 23 as a stream of two characters ,2' ,3' and add a return character ,\n' at the end which corresponds to integer numbers 50, 51, 10 in the ASCII code <http://www.asciitable.com/>
- In order to correctly receive the integer number 23, we need to use command `port.readStringUntil(, \n')` to first receive it as a string „23“ and then convert it back to an integer by `int(float(port.readStringUntil(, \n') ))`
- `Serial.println(„23:34:49“)` can send multiple data at once. The receiver will use `port.readStringUntil(, \n')` to receive it as a whole stream and then separate each substrings by searching for ,:', and at the end convert each substring back to integer

# Interface Arduino to Python

1. Library: `Import serial`  
`Import time`
2. Define the serial port and baud rate: `ser = serial . Serial ('COM3 ', 9600)`
3. Send data from python to the serial port: `ser . write (b'on\n')`
4. Receive data from the serial port:  
`receivestring = ser. readline (). decode ("utf -8"). strip ()`  
`receivelists = receivestring . split (',')`
5. Open a command window, go to the directory where you save your python code. Type the command „python xxxx.py“ (xxxx is your file name)



## Graphic Display by Python

1. Basic Python Programming:

<https://www.w3schools.com/python/default.asp>

2. Graphic display:

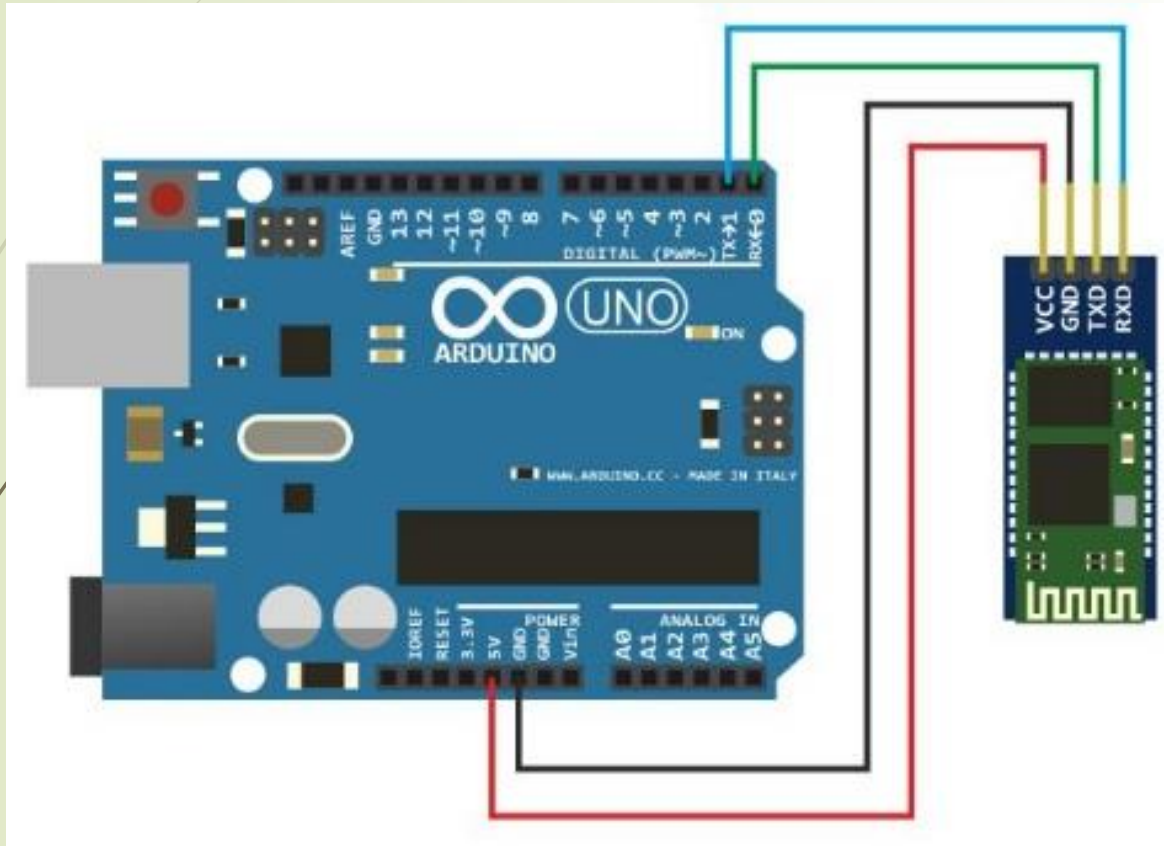
`pip install plotly==4.1.0`

`pip install jupyter`

<https://plot.ly/python/>

3. Type command „jupyter notebook“ and start a new file by clicking „New“ ->python3

# Bluetooth

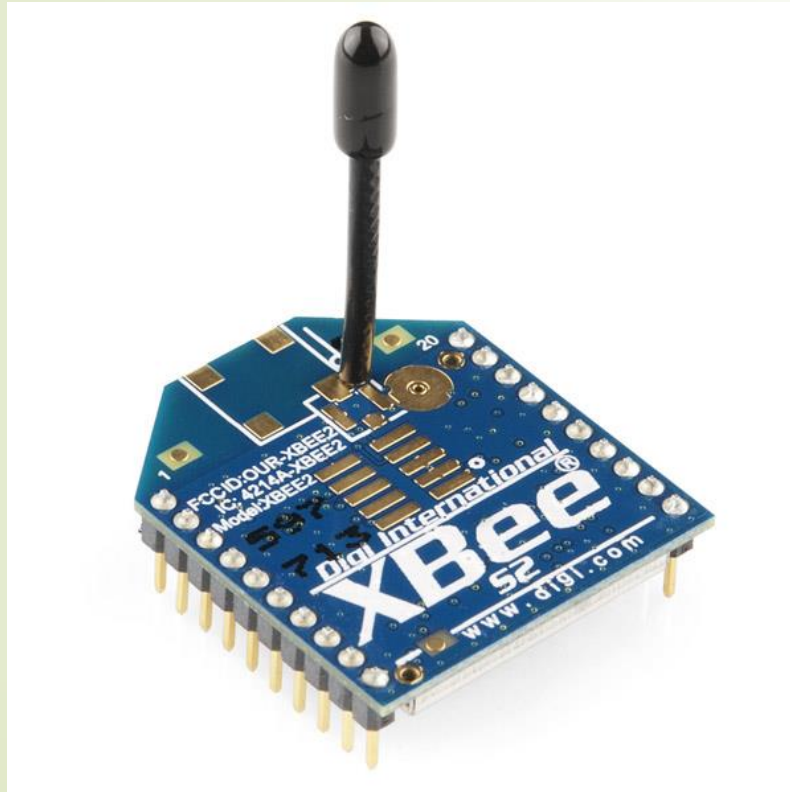


## Features:

- Wireless communicate with Laptop, Smartphone, Pad
- Using 2.4GHz-2.485GHz
- Communication Distance: Less than 10m
- Data rate (1Mb/s)
- Headset, Speaker, Mouse, Printer, Keyboard, Game Controller



# ZigBee (IEEE802.15.4)



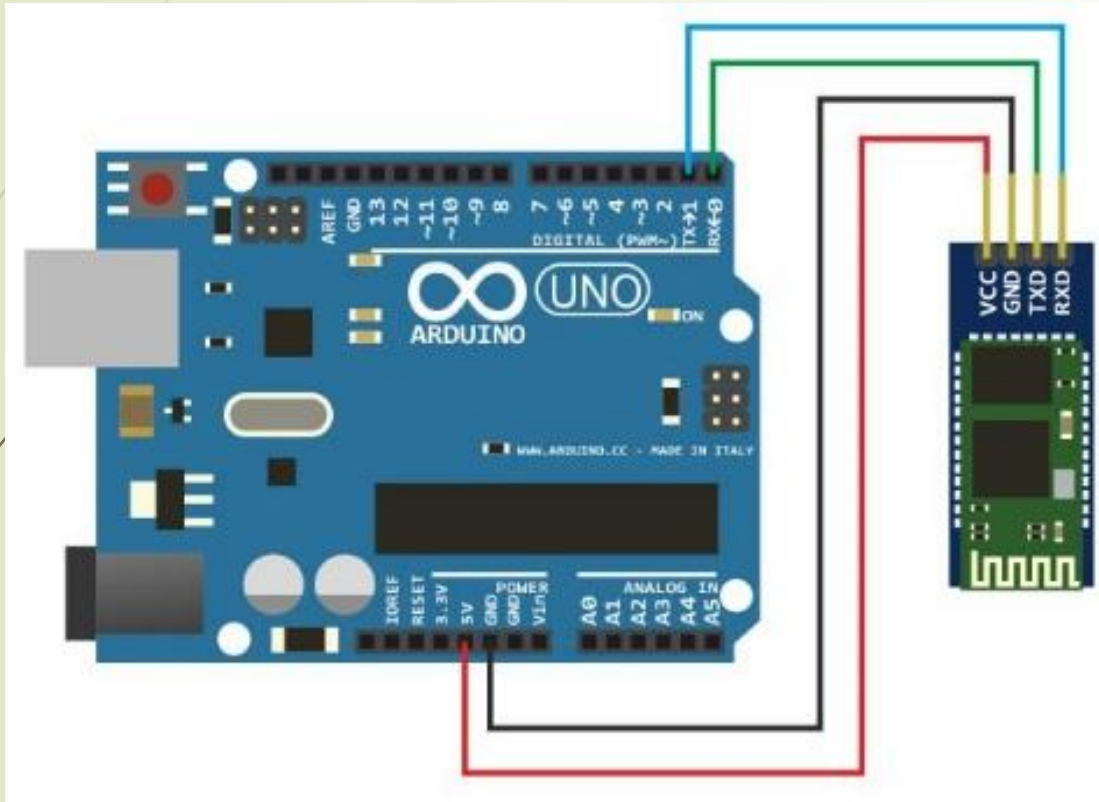
## Features:

- Intended to be less expensive than Bluetooth, WiFi
- Using 2.4GHz
- Communication Distance:  
Less than 100m indoor  
Less than 1.6km outdoor
- Low power,  
Low data rate (20kb/s-250kb/s)
- Home automation, Wireless sensor network, Industrial control systems, Medical data collection, Building automation

## Compare between Bluetooth, ZigBee and WiFi

Bluetooth	ZigBee	WiFi
Personal Area Network	Personal Area Network	Local Area Network
<= 10 meter	<= 100 meter (indoor) 1.6km (outdoor)	Hundreds Kilometers
Peer to peer	Home, building	City, Campus
1Mb/s	20kb/s- 250kb/s	Up to 1Gb/s
Low power	Low power	High power
2.4GHz frequency band	2.4GHz frequency band	2.4GHz frequency band
IEEE802.15.1	IEEE802.15.4	IEEE802.11

# Bluetooth – Experiment 1



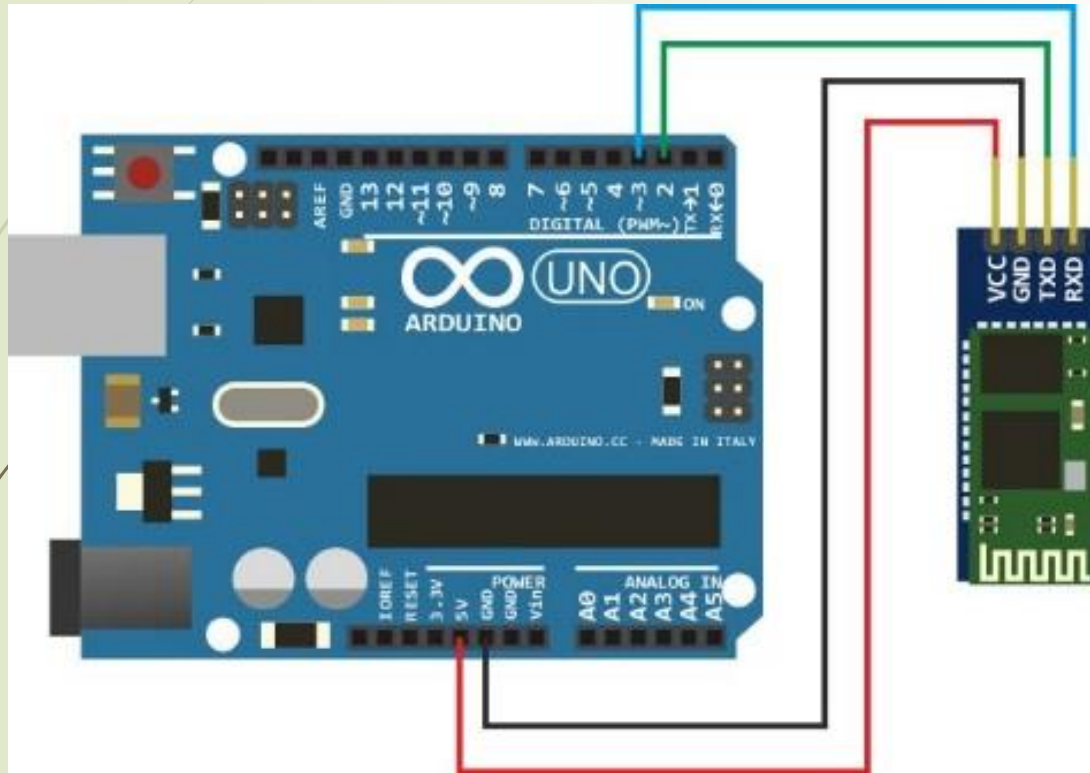
## Steps:

- Connect the Bluetooth Module only after the example code runs successfully
- Connect Bluetooth (TX-RX)
- Find and change the port (COM??)
- Using power supply, disconnect USB
- Try to control the LED via the new port (the Bluetooth channel)

**Important: After the first step, never upload the code to Arduino again.**

**If you want to upload the code, please disconnect the wire from pin 0 and pin 1**

# Bluetooth – Experiment 2



<http://remotexy.com/en/help/start/arduino-hc05/>  
(connect to smartphone)

## Steps:

- Define Arduino RX TX (6,7) and bluetoothSerial by the library <SoftwareSerial.h> (given in the Manual)
- Bluetooth TX – Arduino RX
- Replace „Serial“ to „bluetoothSerial“ in your program
- Change the port to USB cable's port, to upload the code
- Change the port back to the Bluetooth port, and test

# ZigBee (IEEE802.15.4)



xBee

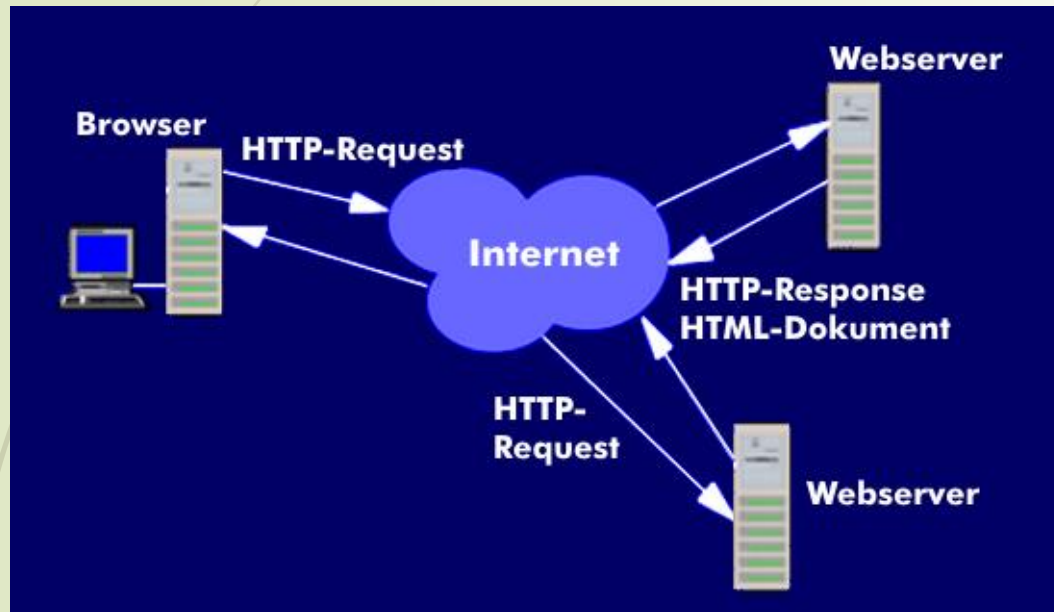


xBee  
Shield

## Steps:

- xBee modules need to be configured to the same frequency channel in order to communicate with each other (they are already configured in this lab)
- Connect xBeeshield to Arduino as the manual describes
- Using the given program to test
- Serial.print() will send data to the computer serial monitor via the USB cable
- xBeeSerial.print() will send data between two xBee antennas

# Website



- Get an Username and a Password from the instructor. They are used to access the server. The IP address of the server can be acquired from the instructor.
- You can create your own .html file and transfer the file to your personal directory of the server using a software called WinSCP.
- Others can browser your .html file by entering the IP address of the file:

<http://xx.xx.xx.xx/user/userxx/xxxx.html>

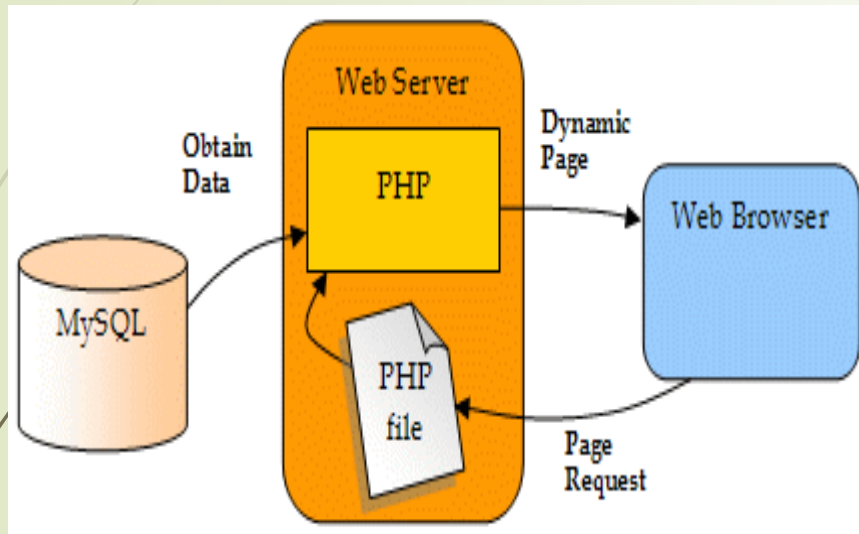
IP address  
of the  
server

Your  
personal  
directory

The file  
you  
created

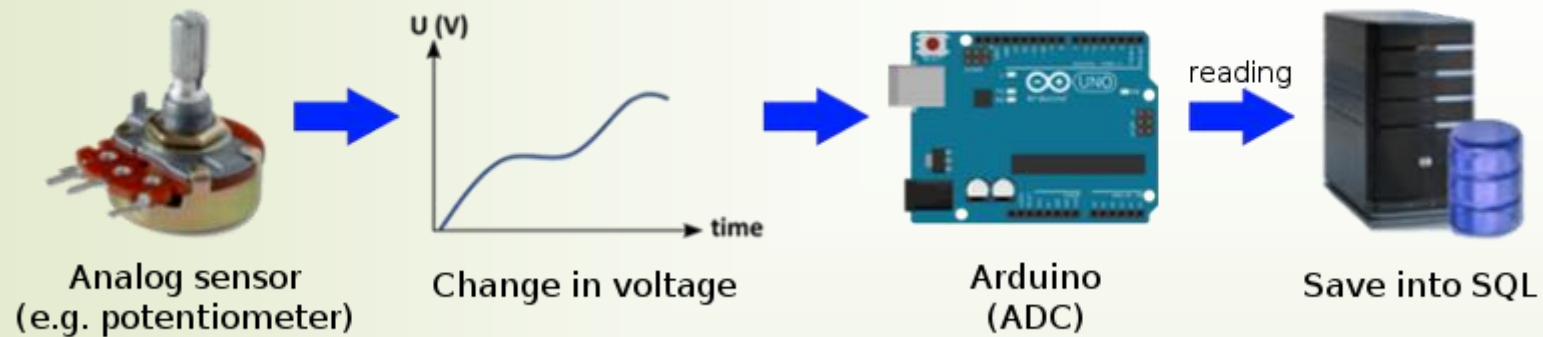


# PHP and Database



- The browser requires to run a .PHP file on the server via a method called „POST“
- The .PHP file contains SQL commands to interact with the database such as creating a table, fetching a record, or inserting a record in the database.
- The .PHP file will then generate a webpage containing the necessary data obtained from the database and return to the browser. Since the returned webpage contains data based on what the browser requires, the generated page is called dynamic page.

# Connect Sensors to Database



1. Processing (import de. bezier . data . sql .\*; ) given in the manual
2. Python (pip -m install mysql-connector-python)  
[https://www.w3schools.com/python/python\\_mysql\\_getstarted.asp](https://www.w3schools.com/python/python_mysql_getstarted.asp)
3. Obtain data from the database and display the data graphically



# Final Project

- The technical complexity (30%);
  - The usability (30%);
  - The documentation (40%)
- 