An Arduino Crash Course

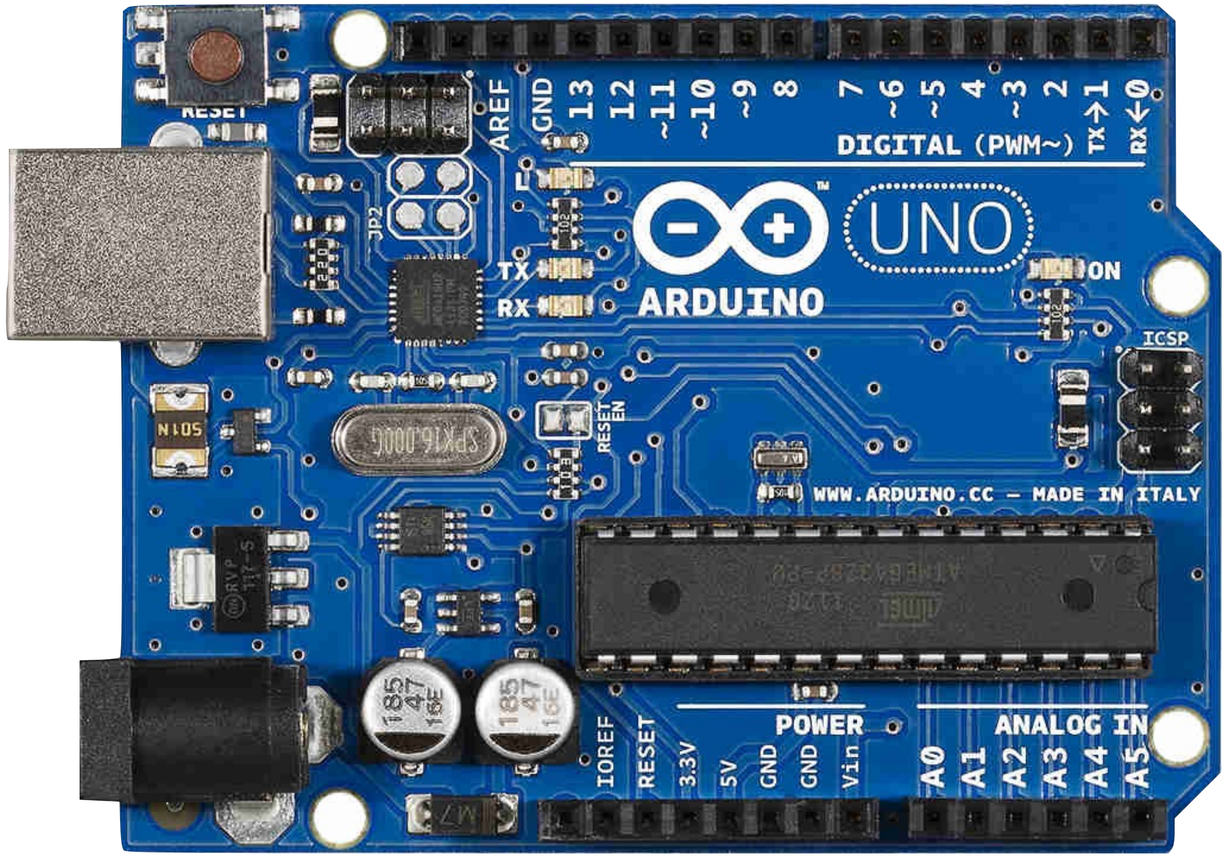
Giovanni Organtini

Dipartimento di Fisica

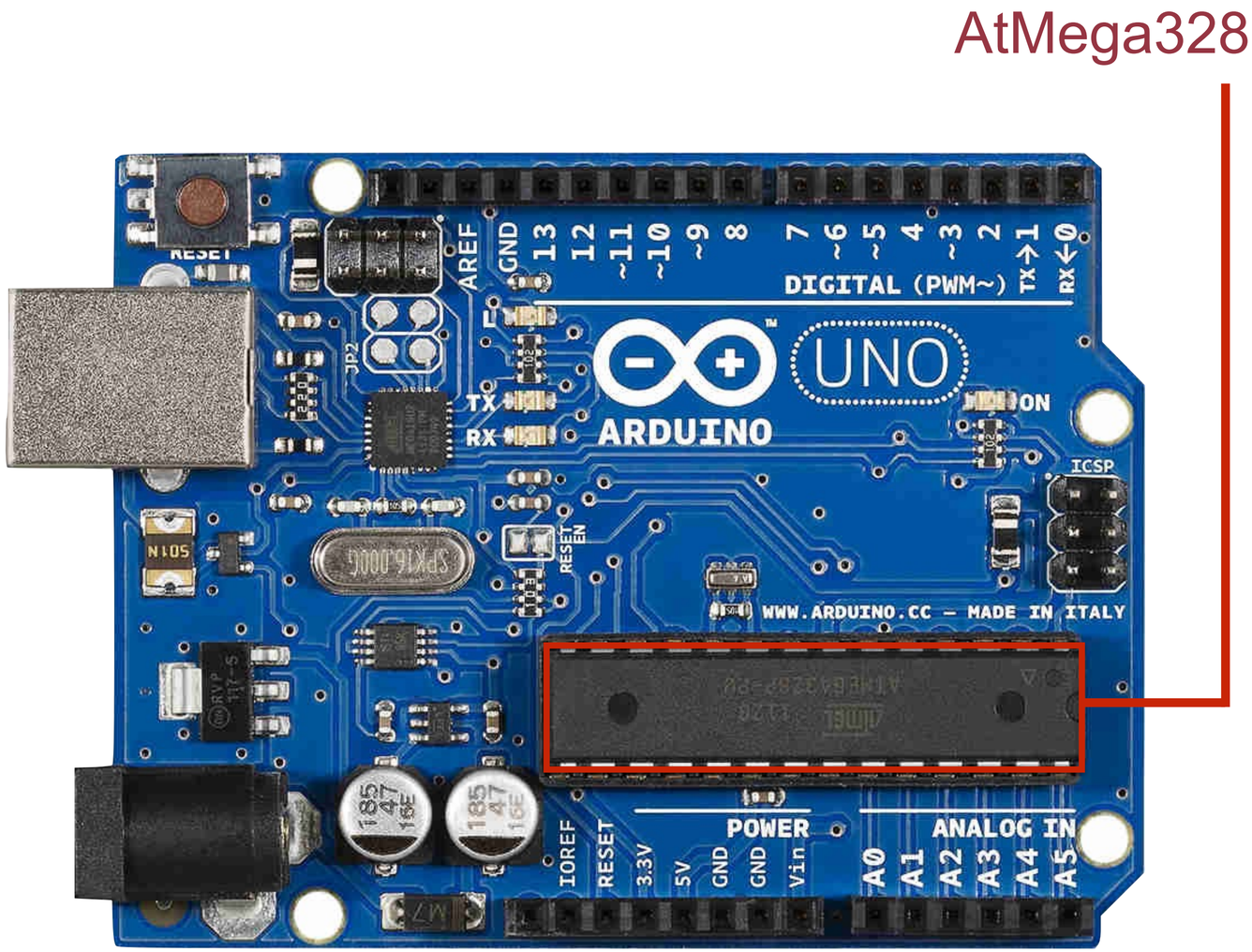
Sapienza
Università di Roma

Desy - Zeuthen

What is Arduino?

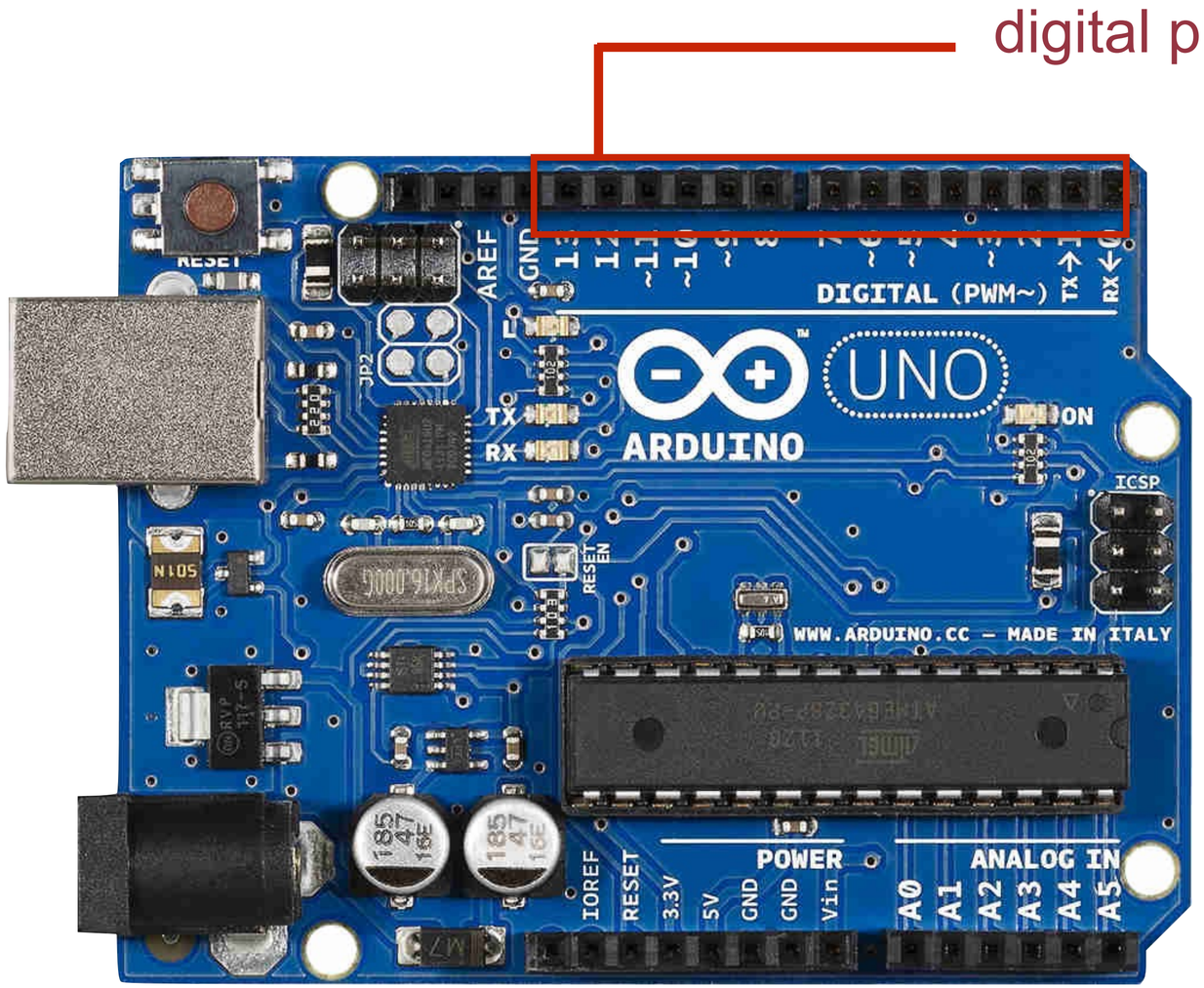


What is Arduino?

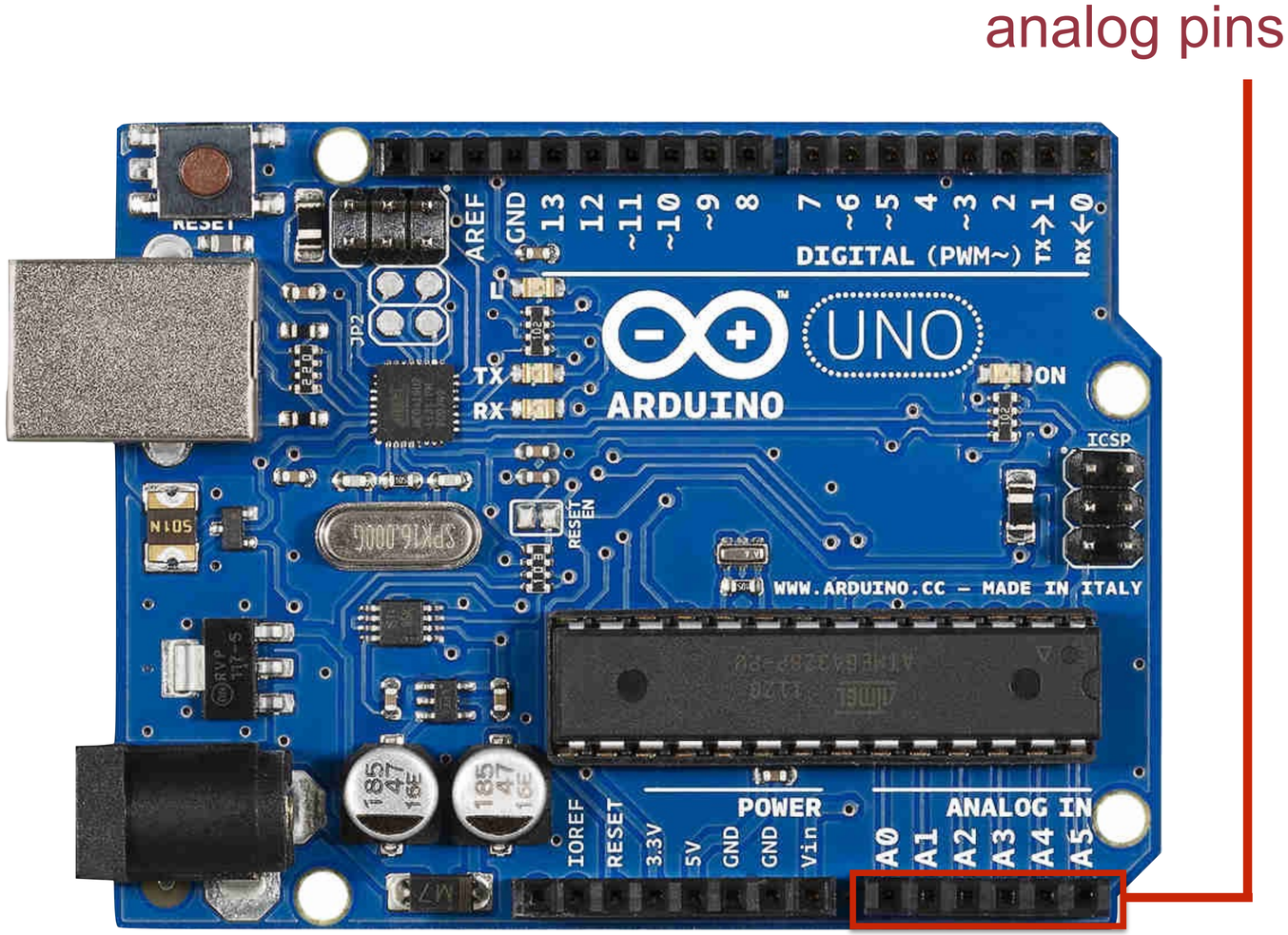


What is Arduino?

What is Arduino?

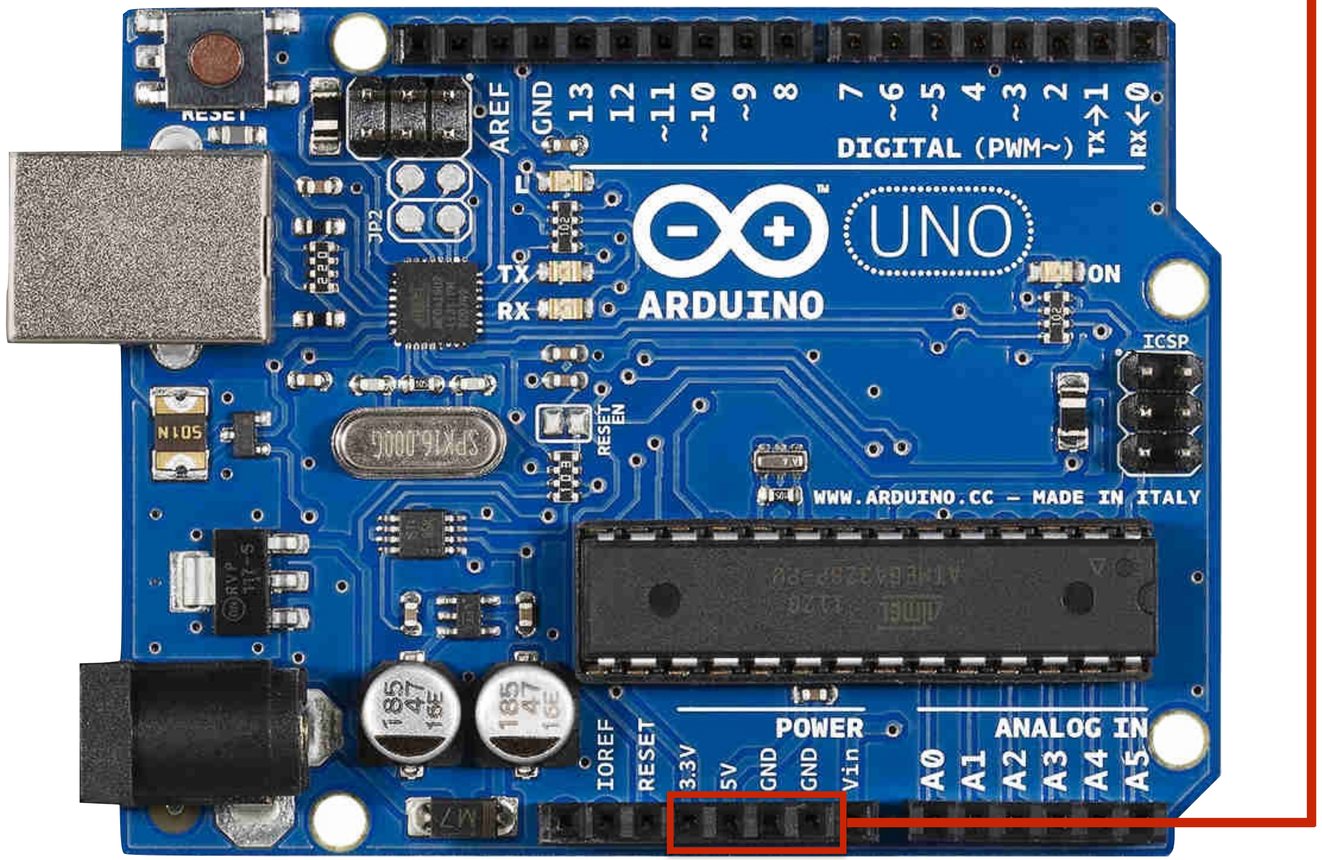


What is Arduino?

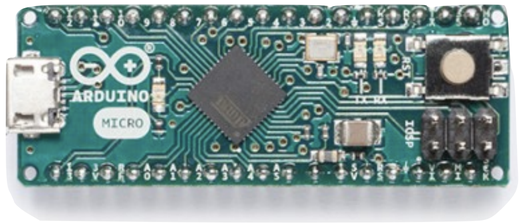


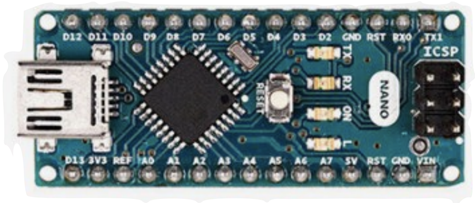
What is Arduino?

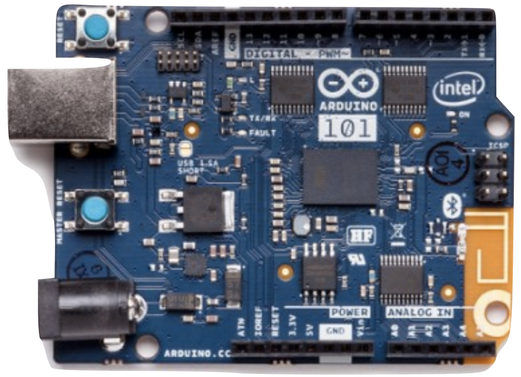
power pins

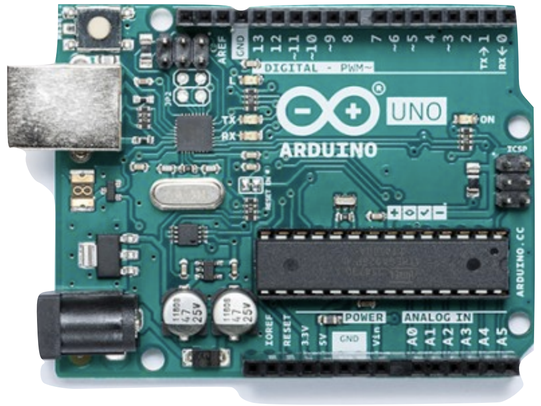


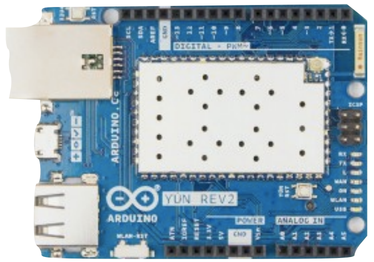
A whole Arduino family

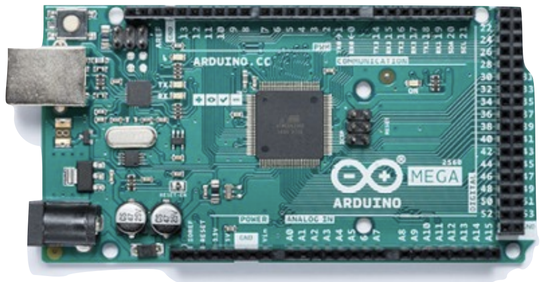


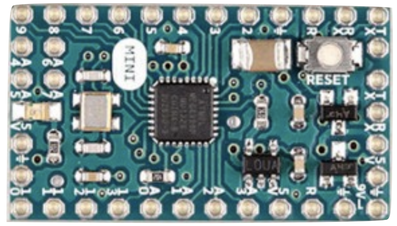


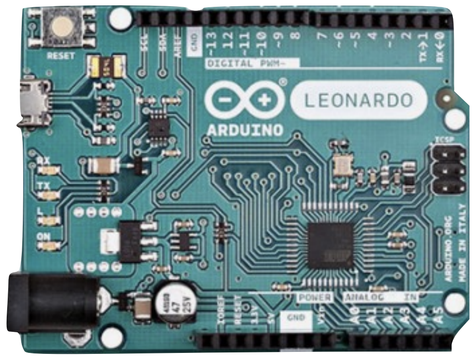




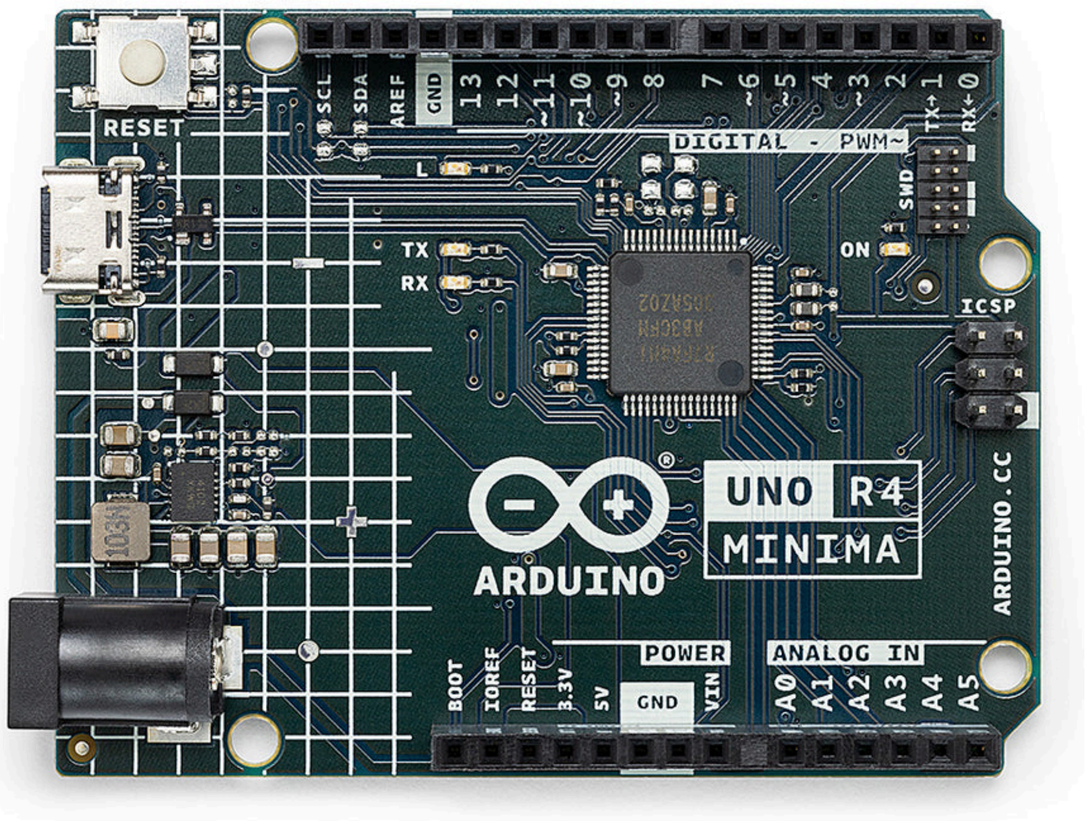


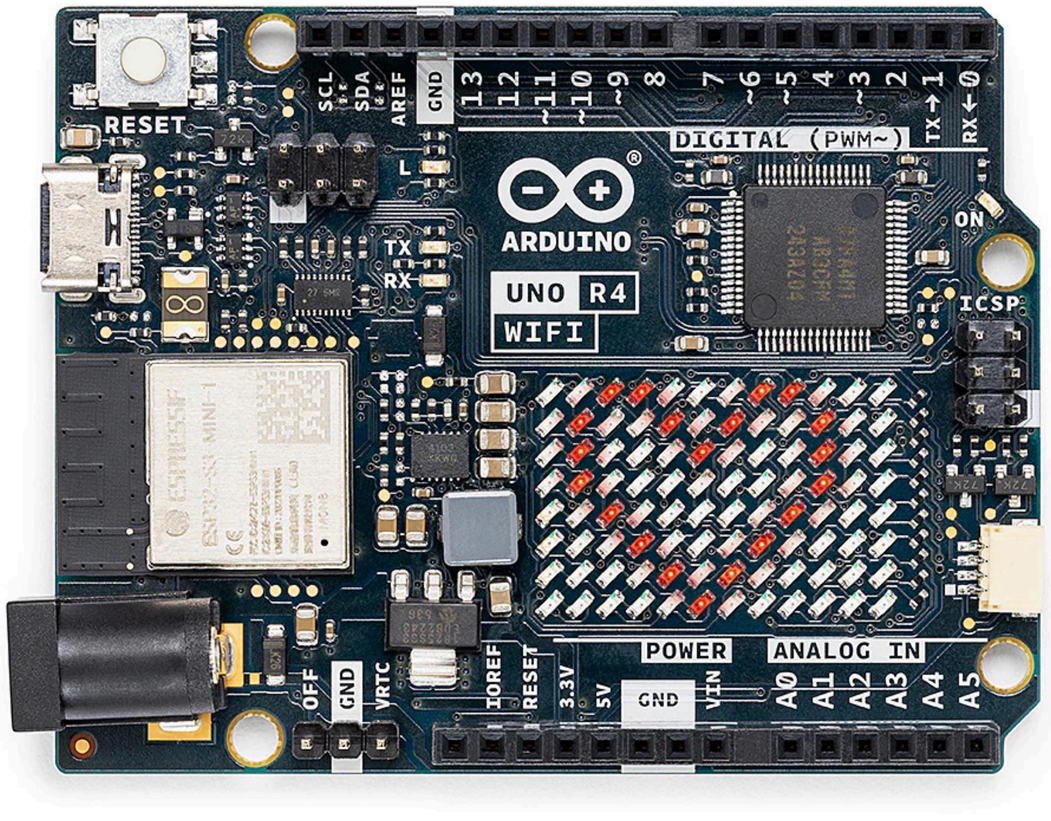




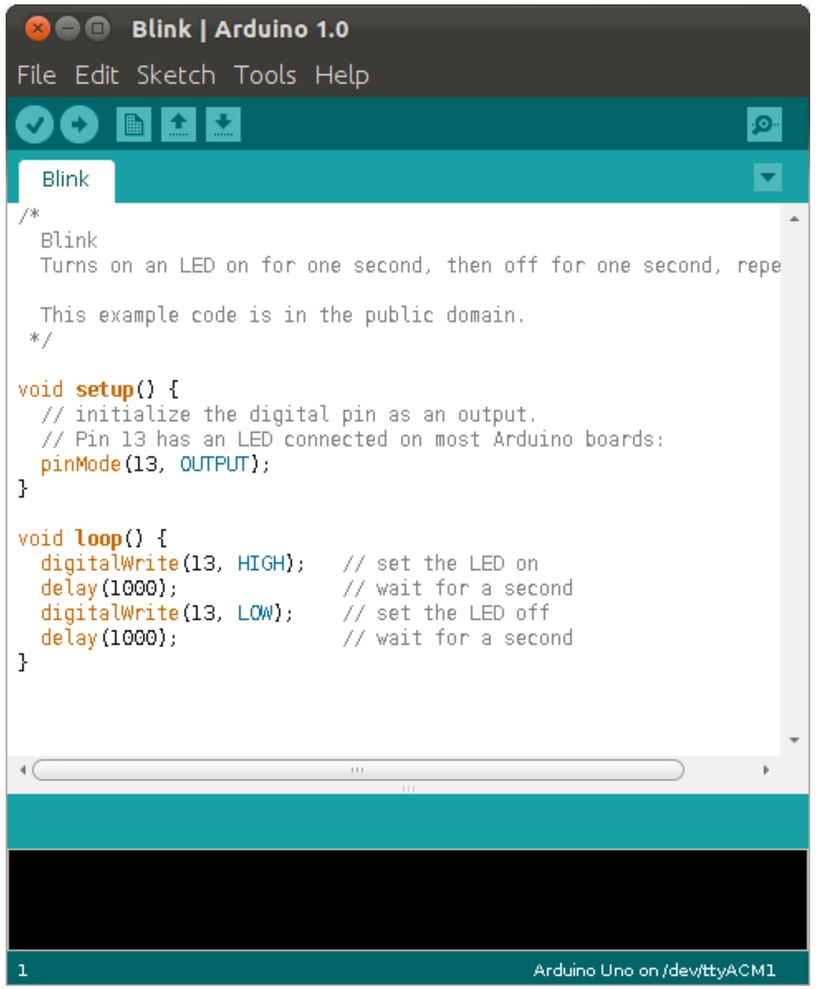


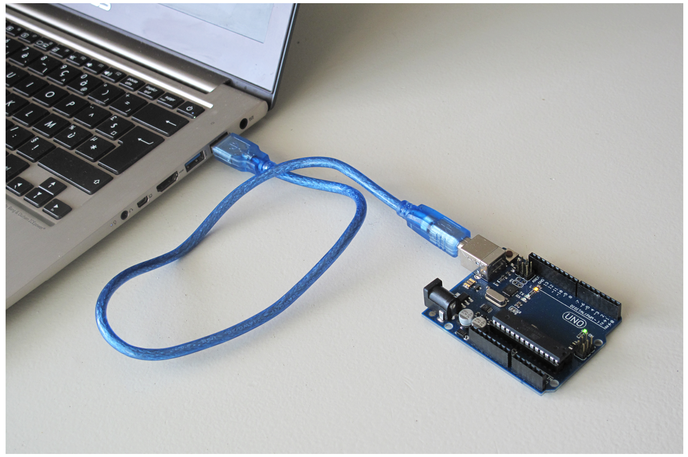
A whole Arduino family





Arduino IDE

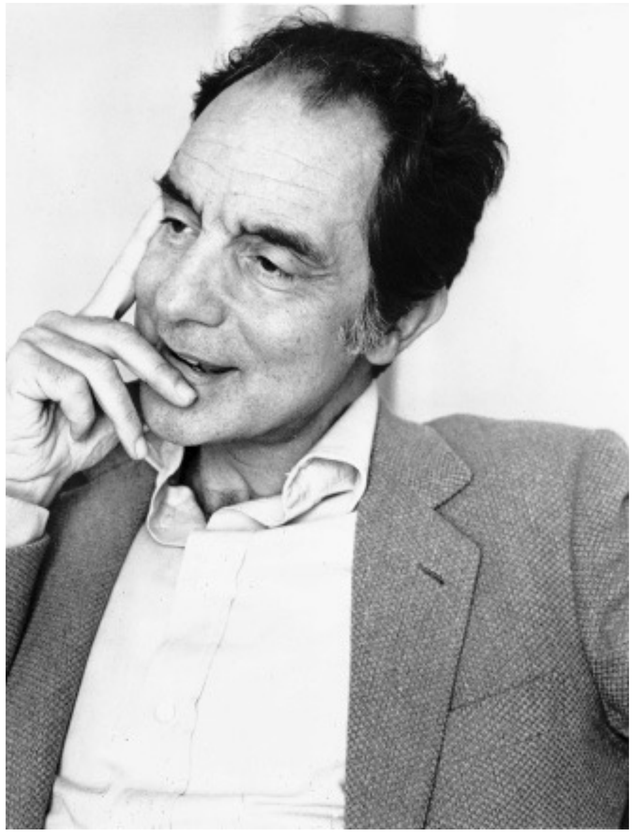

Blink | Arduino 1.0
File Edit Sketch Tools Help
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repe
This example code is in the public domain.
*/
void setup() {
// initialize the digital pin as an output.
// Pin 13 has an LED connected on most Arduino boards:
pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(13, LOW); // set the LED off
delay(1000); // wait for a second
}
1
Arduino Uno on /dev/ttyACM1



Why Arduino is interesting?

- physics @ school = part of the global training process
- laboratory activities is compulsory (IBSE)
  - from “look” to “make”
  - consolidate conceptual understanding
  - acquire “soft skills”
- commercial devices = “black boxes”

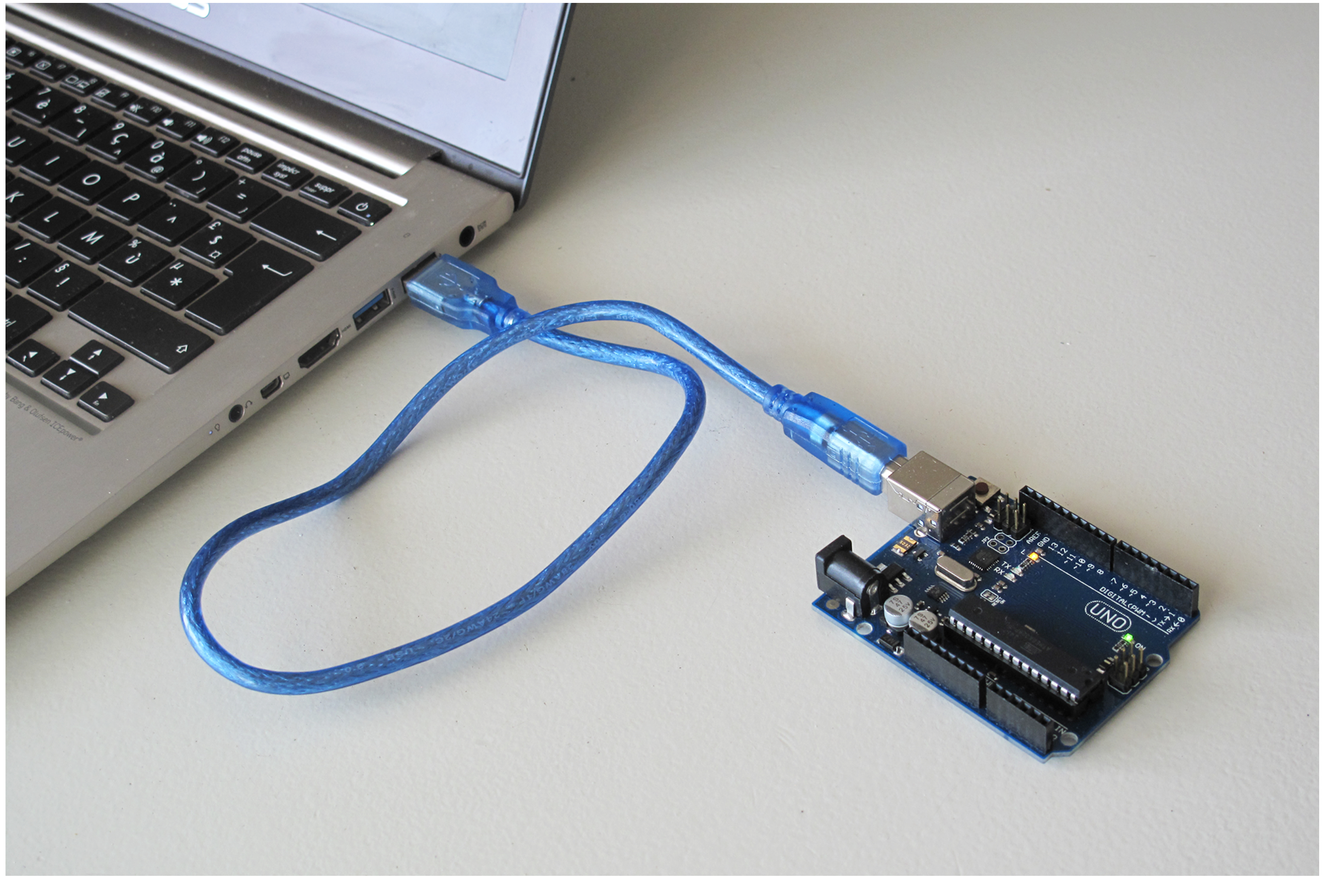
Perché Arduino?



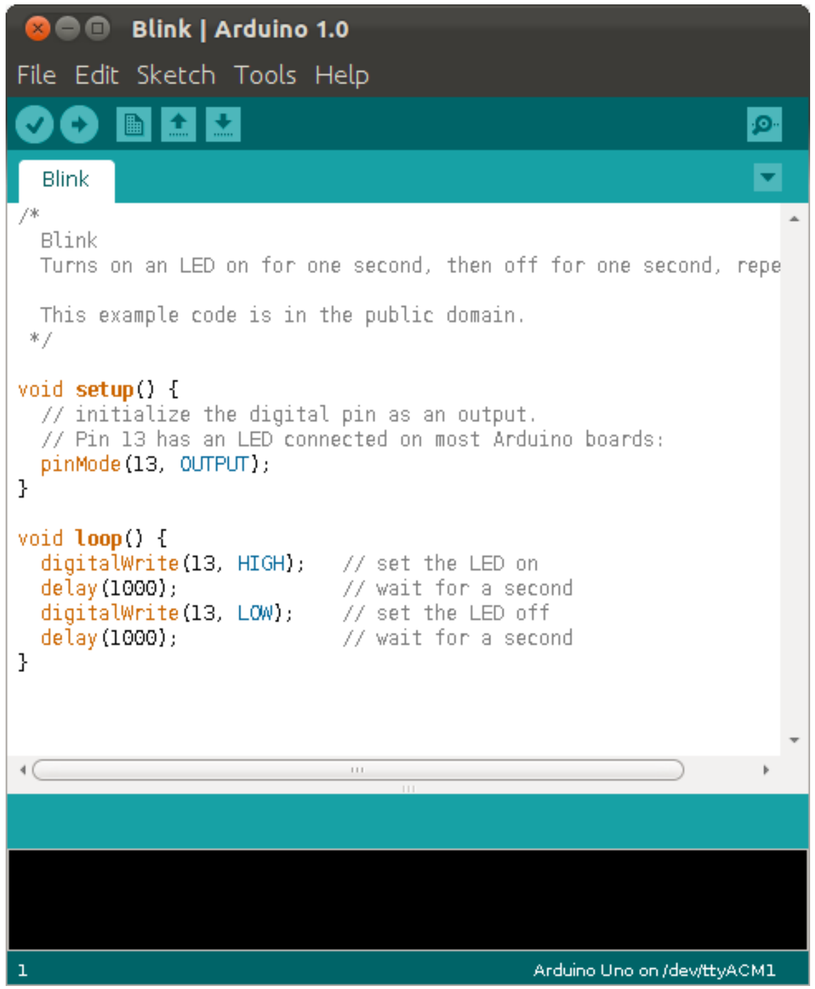
Italo Calvino - Six memos for the next millennium (1988)

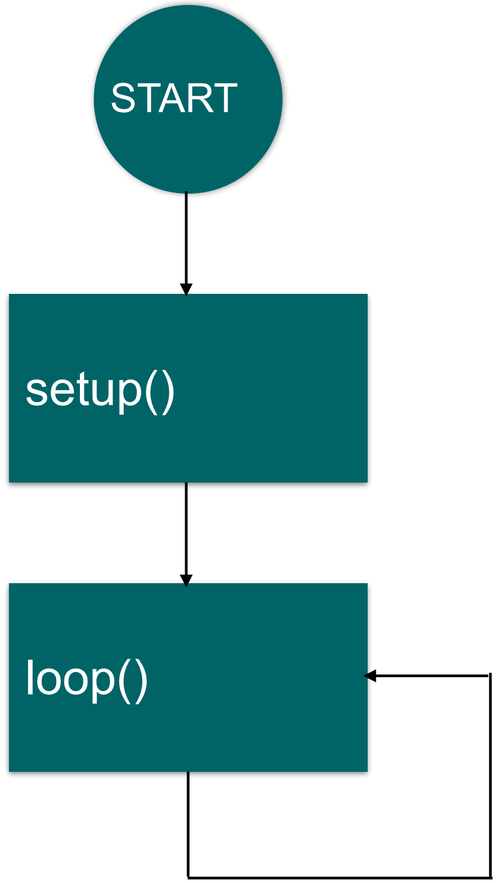
> “It is true that software cannot exercise its powers of lightness except through the weight of hardware. But it is software that gives the orders…”

How to…give the orders…



How to…give the orders…





Anatomy of a *sketch*

setup()

```
void setup() {
   pinMode(10, INPUT);
}
```

loop(): analog pins

```
float V;

void setup() {
   …
}

void loop() {
    V=analogRead(pin)*5./1023;
}
```

loop(): analog pins

```
float V;

void setup() {
   …
}

void loop() {
    V=analogRead(pin)*5./1023;
}
```

analogRead(pin)
reads the voltage
on an analog pin

loop(): analog pins

```
float V;

void setup() {
   …
}

void loop() {
    V=analogRead(A0)*5./1023;
}
```

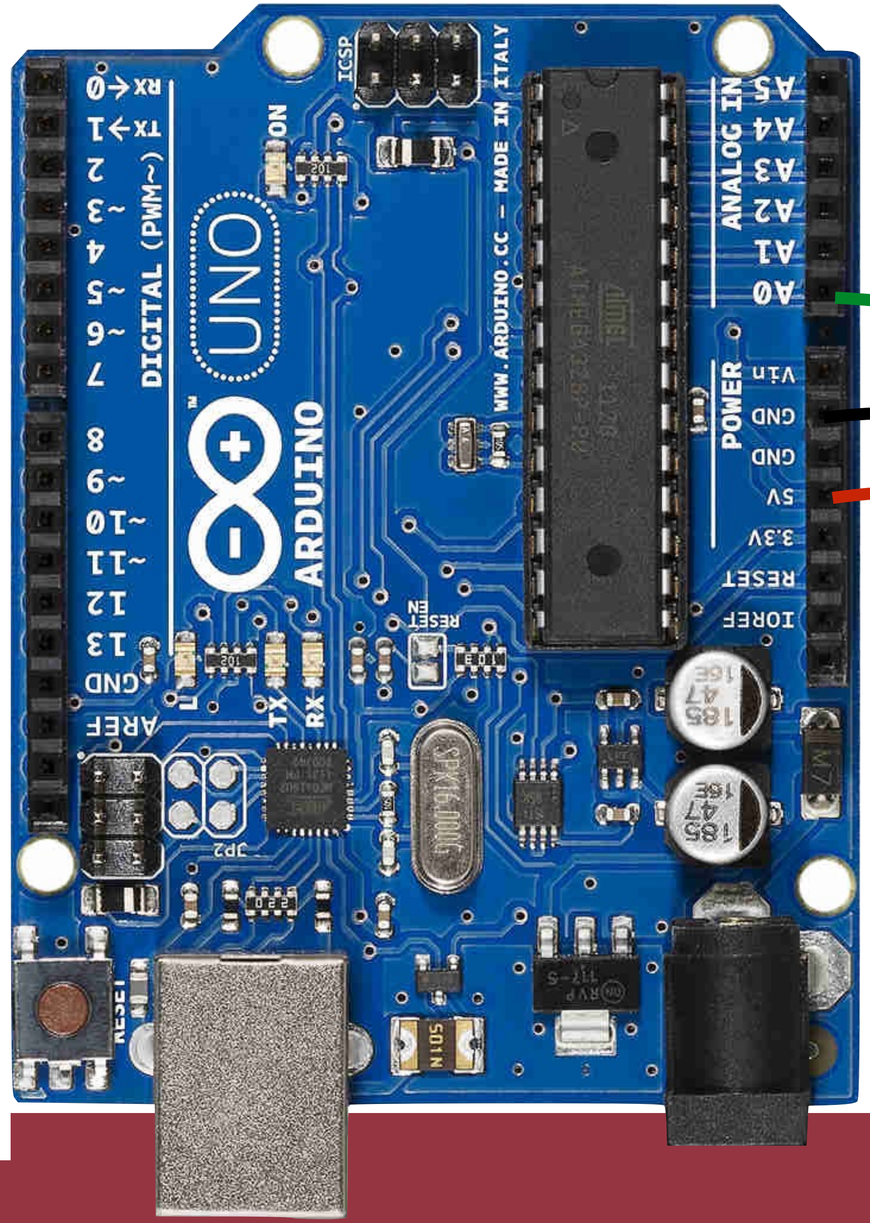
pins are named A0, A1, …

loop(): analog pins

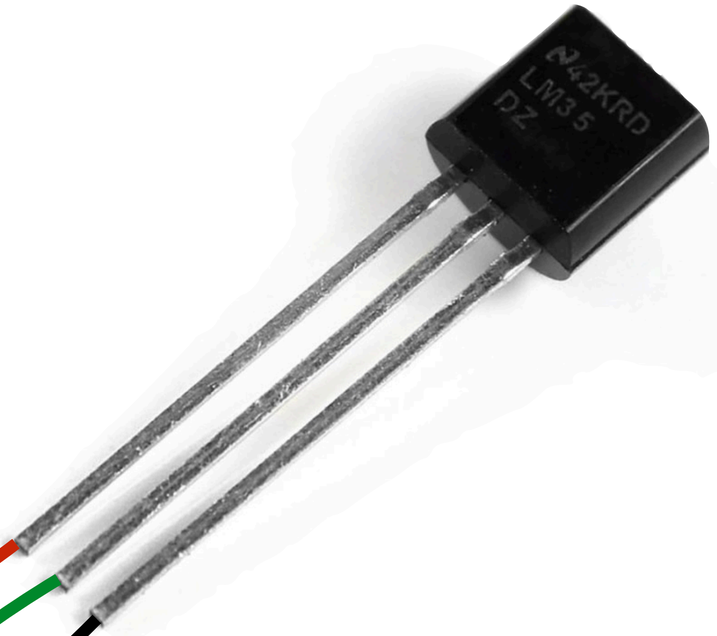
```
float V;

void setup() {
   …
}

void loop() {
    V=analogRead(pin)*5./1023;
}
```

conversion factor

An analog pin example



Temperature Sensor LM35



$$V_{out} = T \times 10\,\mathrm{mV}$$

loop(): digital pins

```
float V;

void setup() {
  …
}

void loop() {
  digitalWrite(pin, HIGH);
  delayMicroseconds(150);
  digitalWrite(pin, LOW);
  delay(1);
}
```

set the pin status

loop(): digital pins

```
float V;

void setup() {
  …
}

void loop() {
  digitalWrite(pin, HIGH);
  delayMicroseconds(150);
  digitalWrite(pin, LOW);
  delay(1);
}
```

HIGH = TRUE = 1

loop(): digital pins

```
float V;

void setup() {
  …
}

void loop() {
  digitalWrite(pin, HIGH);
  delayMicroseconds(150);
  digitalWrite(pin, LOW);
  delay(1);
}
```

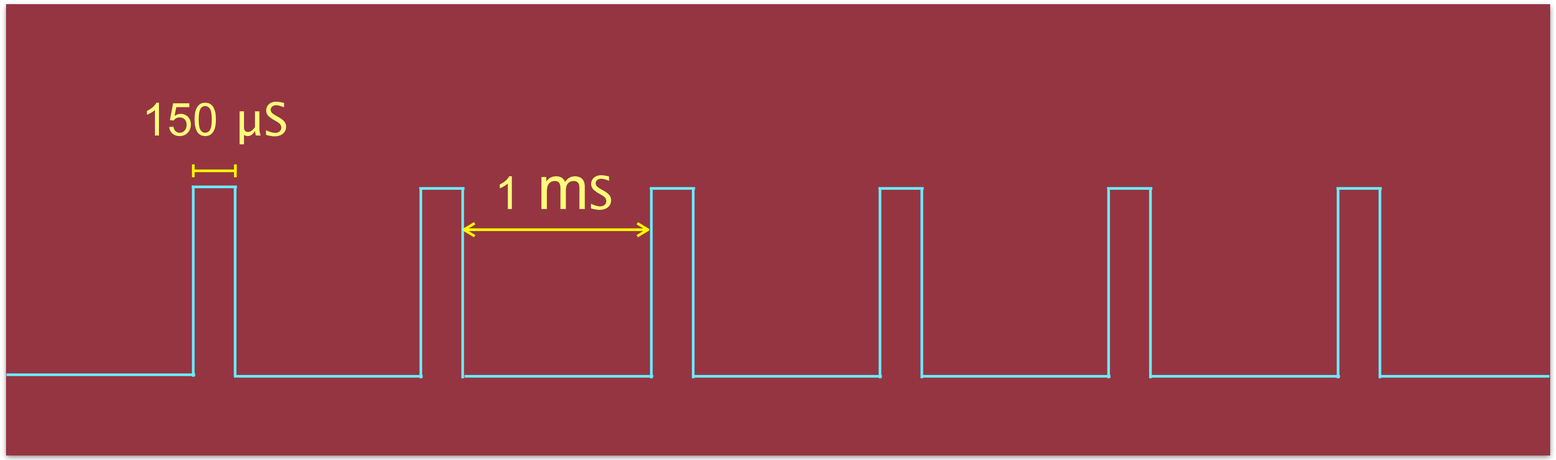
LOW = FALSE = 0

loop(): digital pins

timing functions

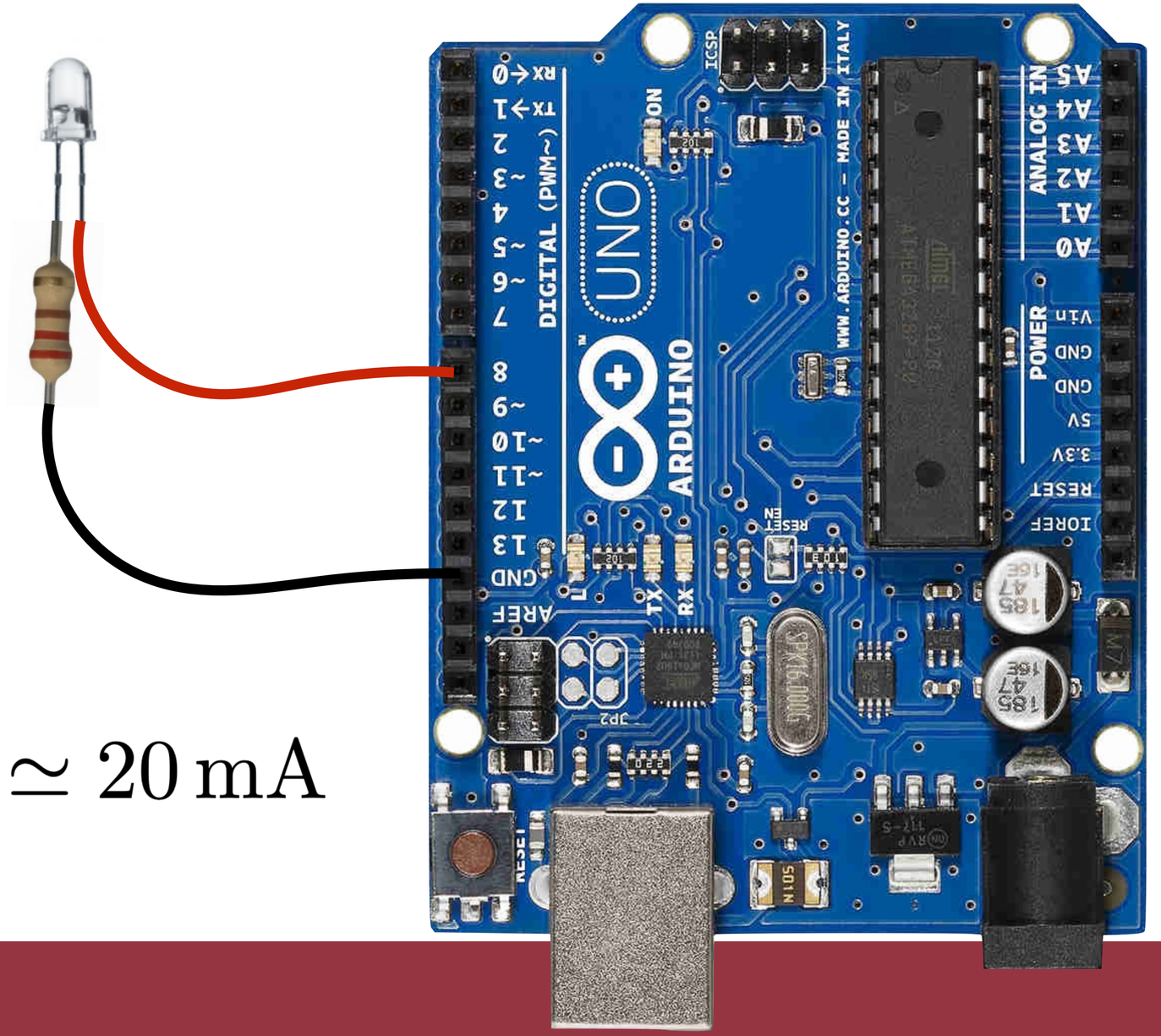
```
float V;

void setup() {
  …
}

void loop() {
  digitalWrite(pin, HIGH);
  delayMicroseconds(150);
  digitalWrite(pin, LOW);
  delay(1);
}
```

loop(): digital pins



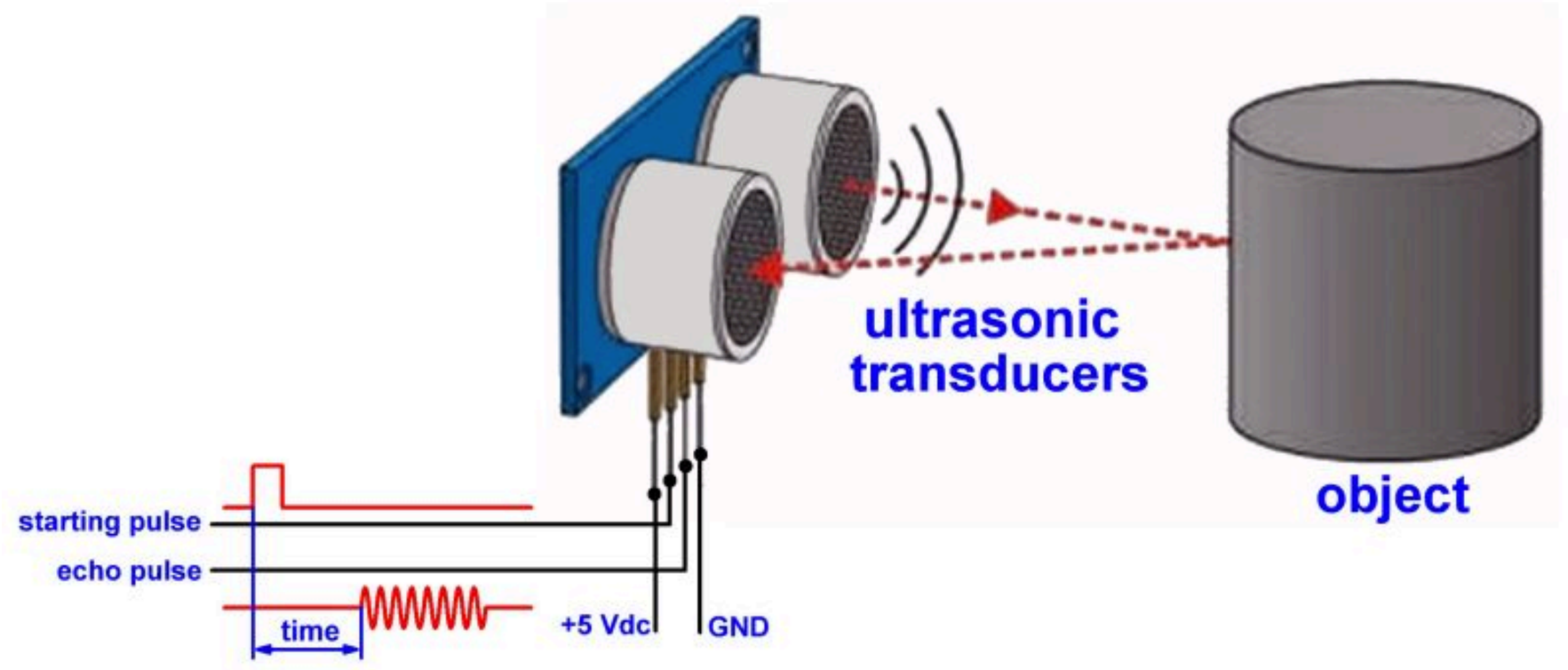

```
void loop() {
  digitalWrite(pin, HIGH);
  delayMicroseconds(150);
  digitalWrite(pin, LOW);
  delay(1);
}
```

A digital pin example

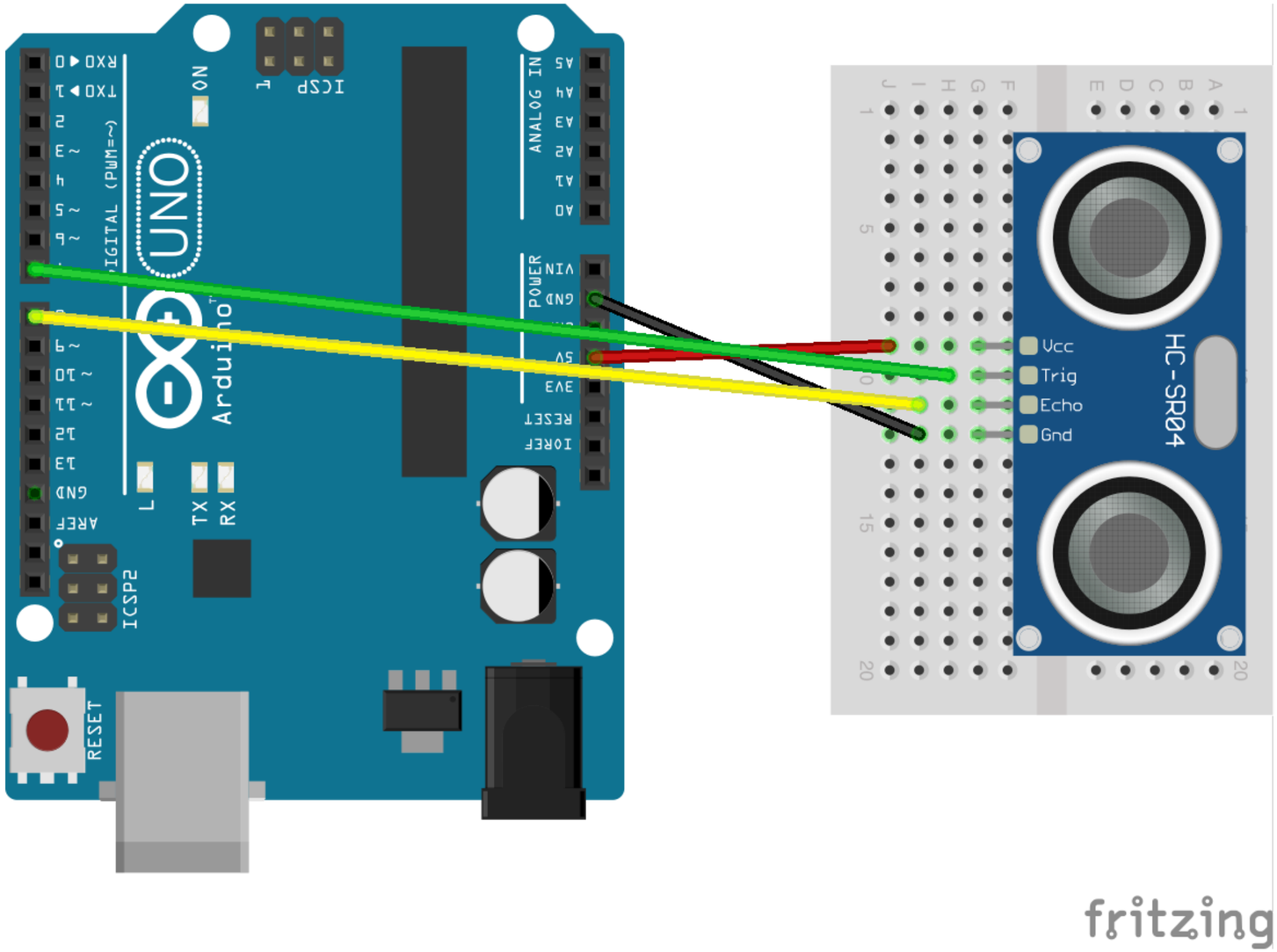


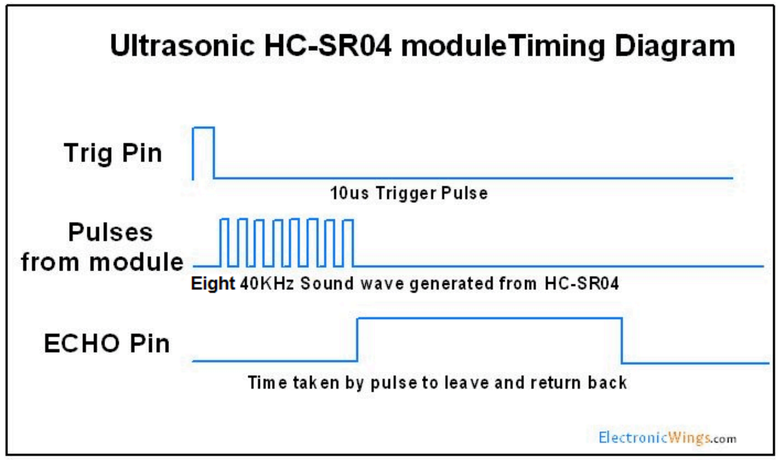
$$I = \frac{\Delta V}{R} = \frac{5}{220} \simeq 20\,\mathrm{mA}$$

Ultrasonic sensors



Ultrasonic sensors





Ultrasonic sensors

```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d);
  delay(50);
}
```

Ultrasonic sensors

```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d);
  delay(50);
}
```

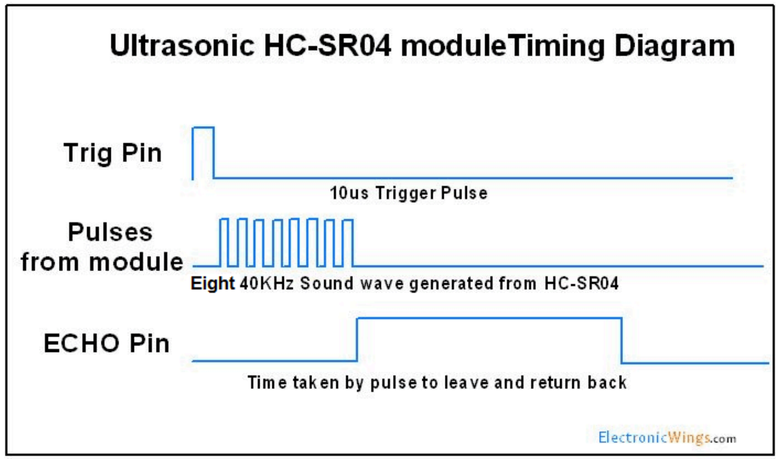
Set pin mode
pin 7: TRIG
pin 8: ECHO

Ultrasonic sensors

```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d);
  delay(50);
}
```

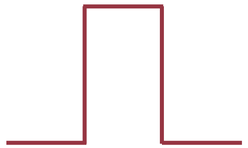
Assign pin status
pin 7 (TRIG): 0 V

Ultrasonic sensors



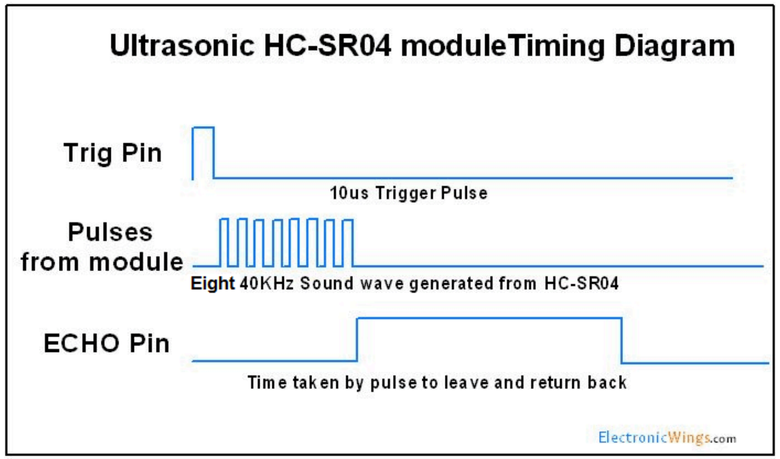

```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d);
  delay(50);
}
```

Generate the trigger pulse



10 µs

Ultrasonic sensors




```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d);
  delay(50);
}
```

Return the time spent in function

Ultrasonic sensors

```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d);
  delay(50);
}
```

Compute distance

Ultrasonic sensors

```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d);
  delay(50);
}
```

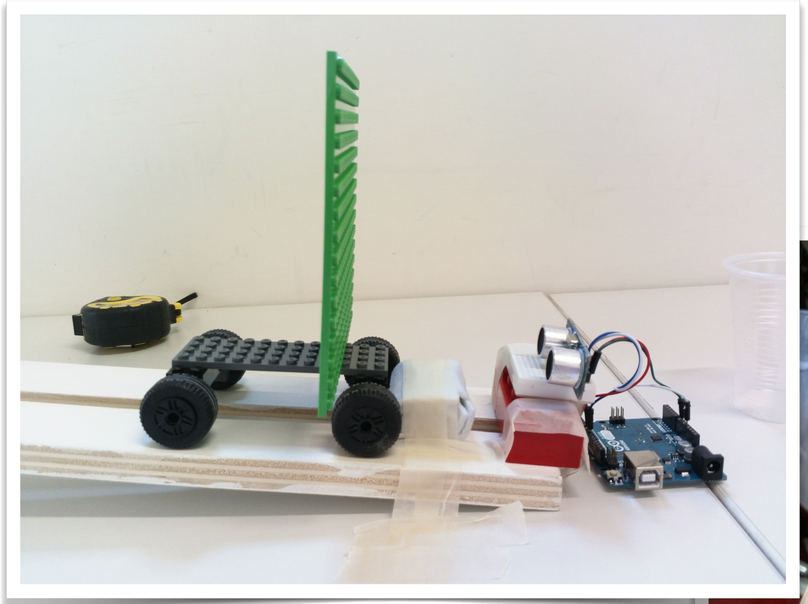
Send distance to USB

Ultrasonic sensors

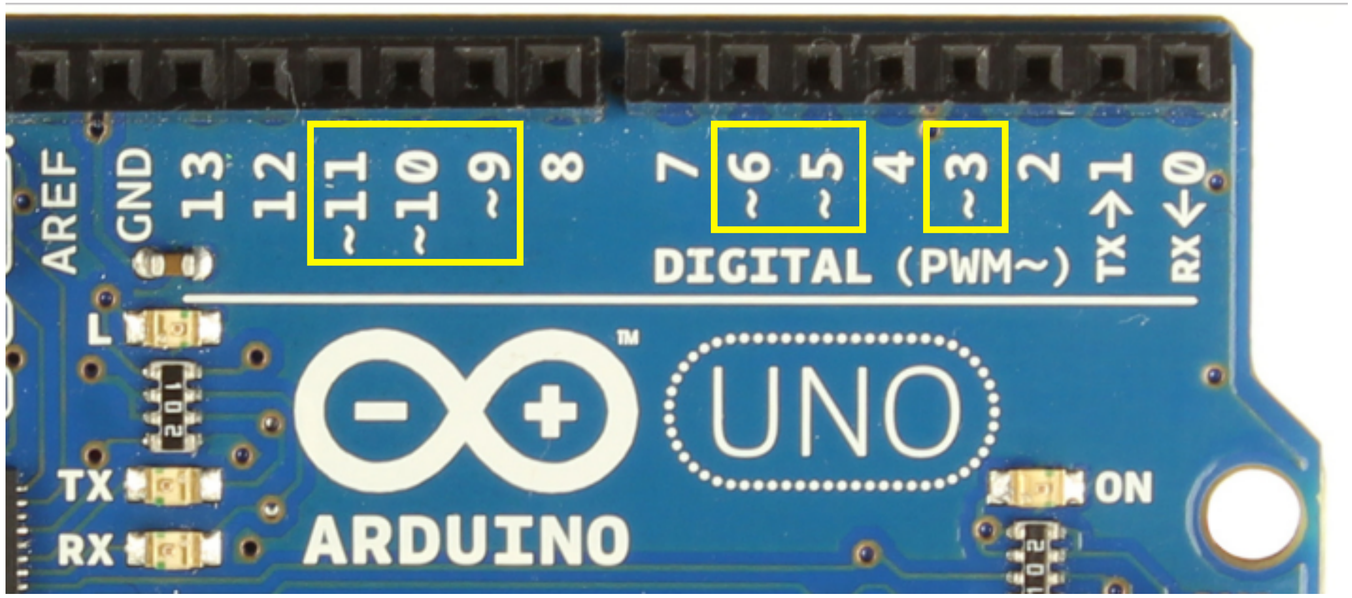
```
void setup() {
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(8, INPUT);
  digitalWrite(7, LOW);
}

void loop() {
  digitalWrite(7, HIGH);
  delayMicroseconds(10);
  digitalWrite(7, LOW);
  unsigned long t = pulseIn(8, HIGH);
  float d = 343.*t*1e-6/2;
  Serial.println(d/2);
  delay(50);
}
```

wait 50 ms


SoundSpeed
Fisica con Arduino 2019



PWM: Pulse Width Modulation



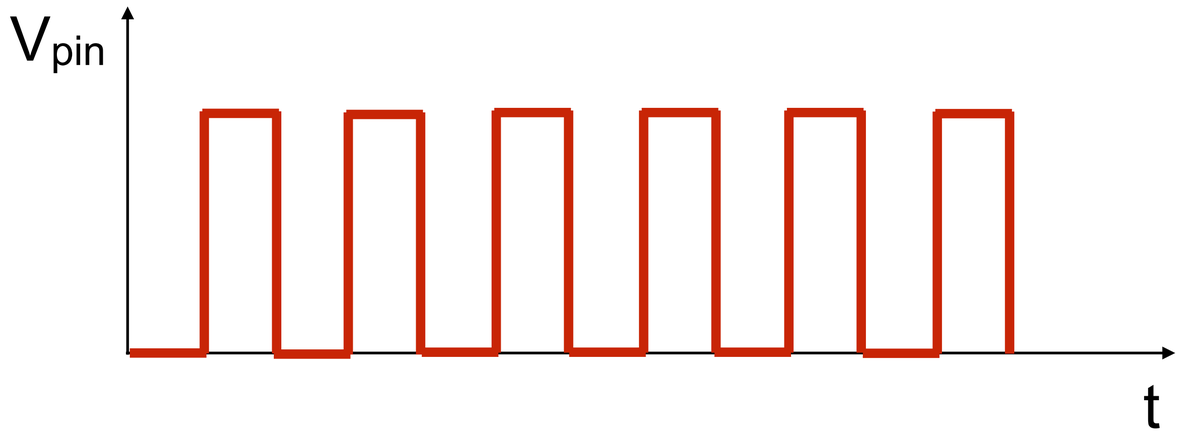

```
analogWrite(pin, value);
```

PWM: Pulse Width Modulation

```
analogWrite(9, 0);
```

PWM: Pulse Width Modulation

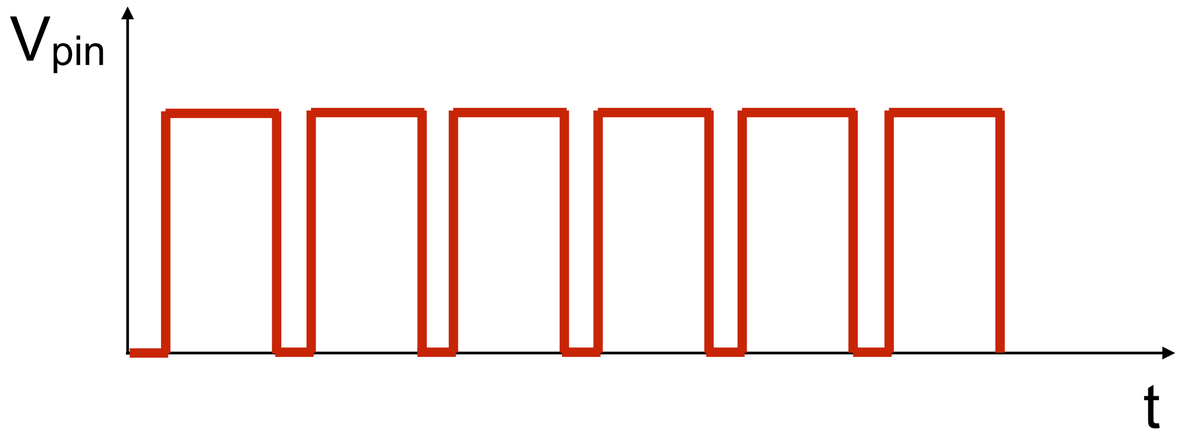
```
analogWrite(9, 127);
```



50% duty cycle

PWM: Pulse Width Modulation

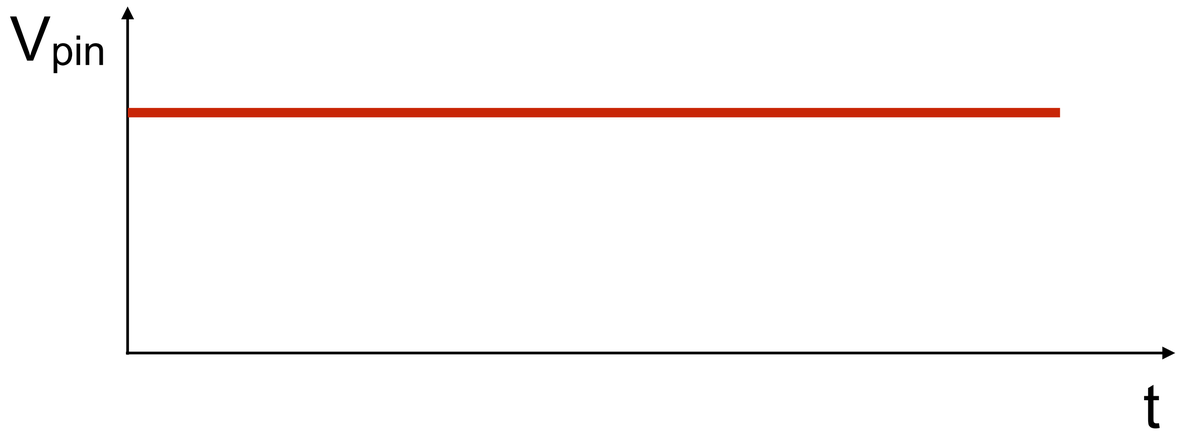
```
analogWrite(9, 191);
```



75% duty cycle

PWM: Pulse Width Modulation

```
analogWrite(9, 255);
```



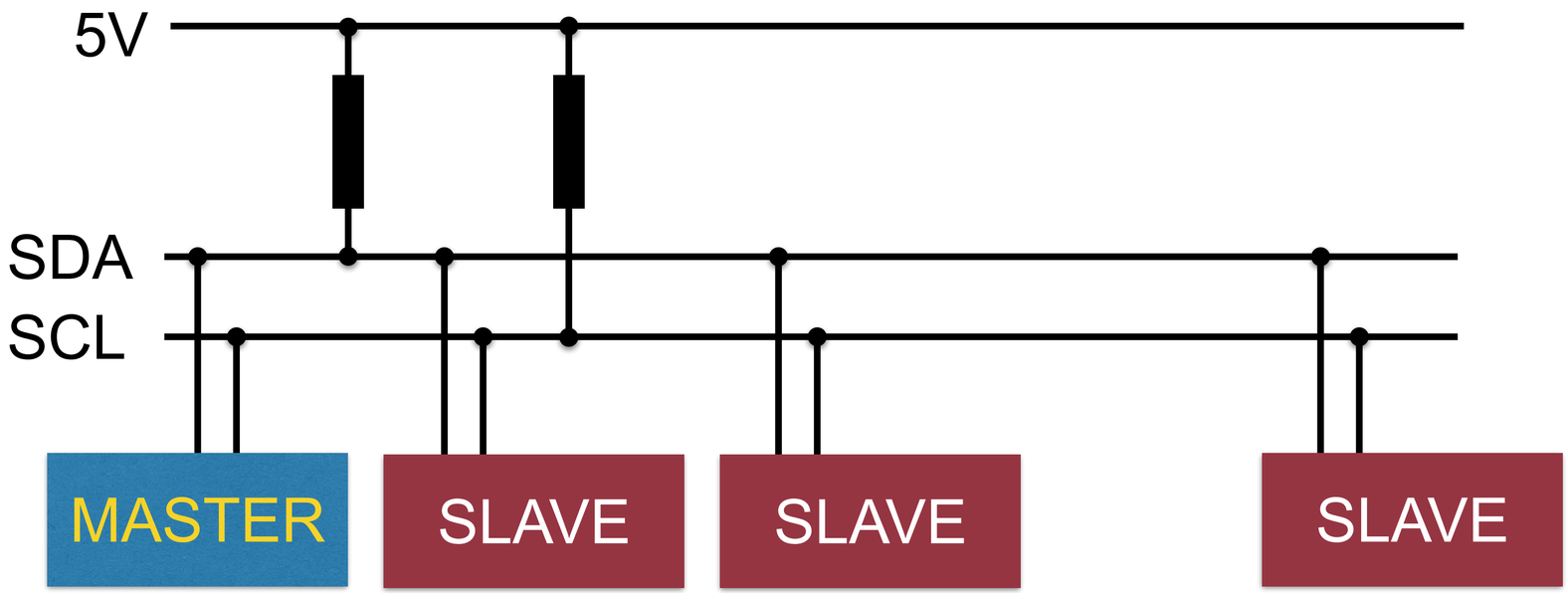
100% duty cycle

Applications of PWM

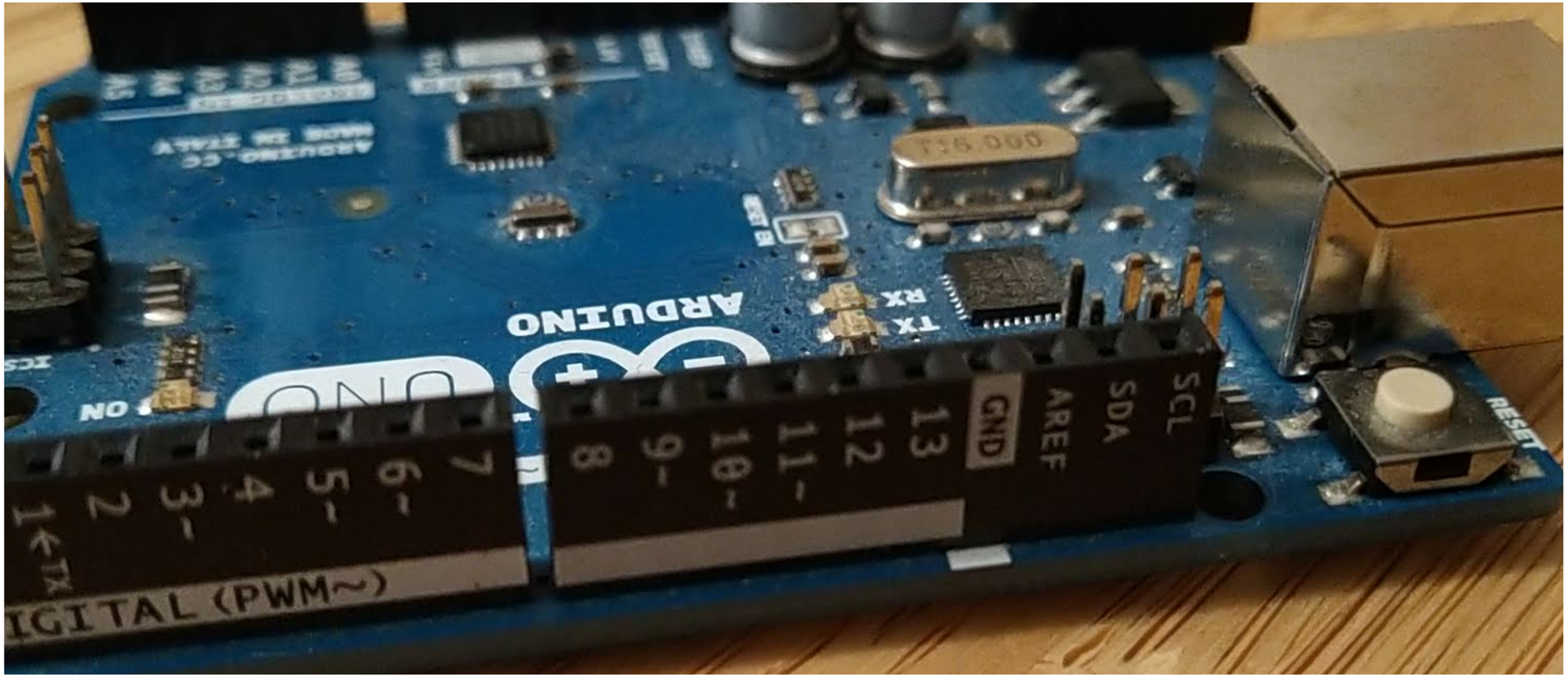
- LED Intensity modulation
- Sound: tone(pin, frequency, duration)
- Motors (servo)

Serial communications

The I²C protocol



Serial communications



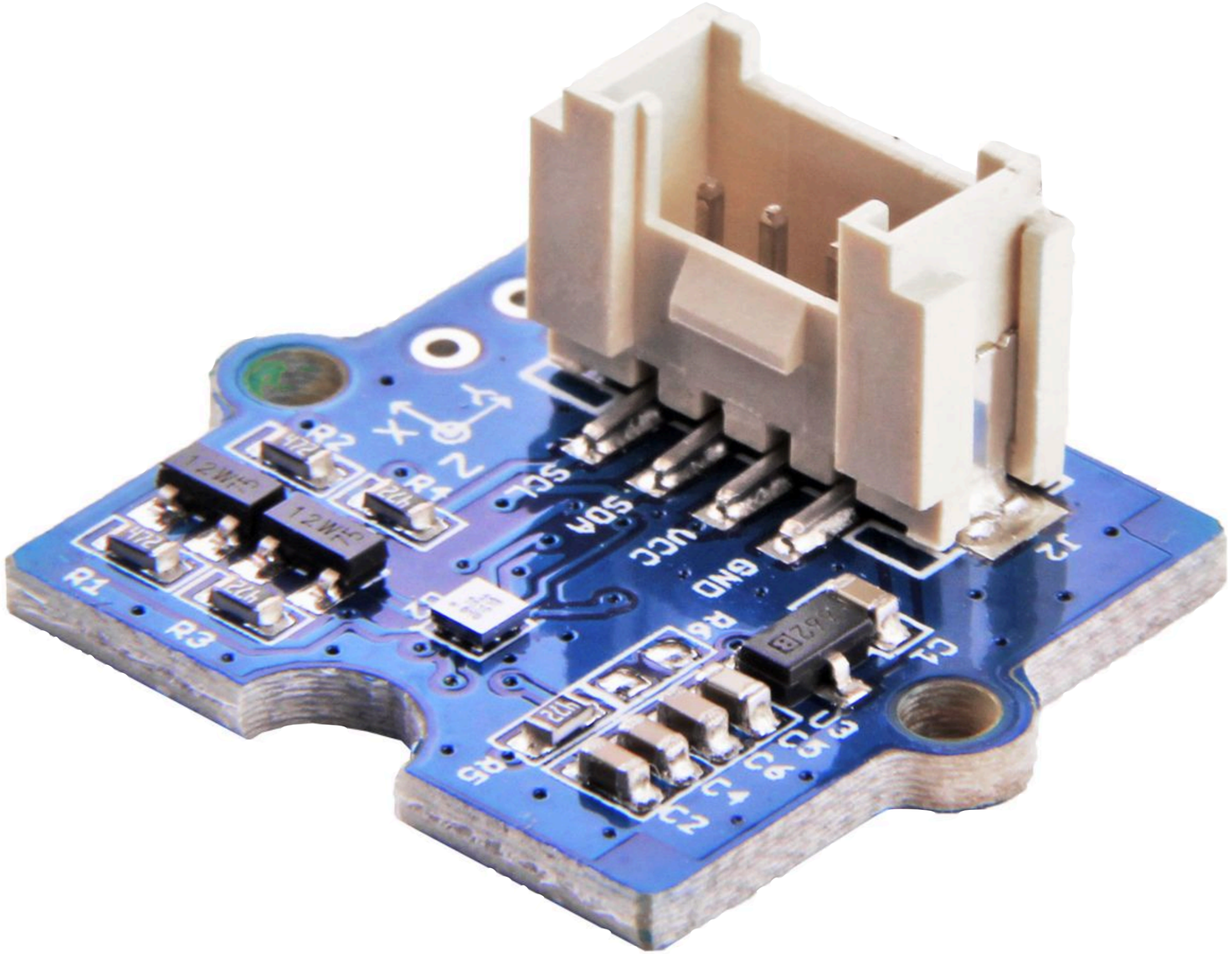
The I²C protocol

- Usually, manufacturers distribute dedicated libraries

- Each slave device has a unique ADDRESS (usually in hexadecimal)
- Start data transmission
- Write/read data as a sequence of bytes
- End data transmission

- Slave addresses can be found on device data sheet
- The Wire library can be used (`#include <Wire.h>`)

The I²C protocol



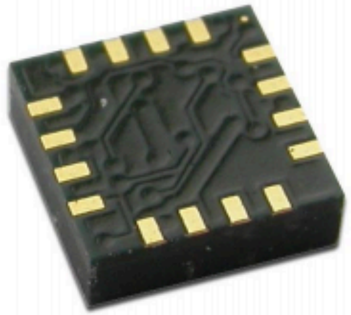
The Wire library

3-Axis Digital Compass IC HMC5883L

Honeywell

Advanced Information

The Honeywell HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing with a digital interface for applications such as low-cost compassing and magnetometry. The HMC5883L includes our state-of-the-art, high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy. The I^2C serial bus allows for easy interface. The HMC5883L is a 3.0x3.0x0.9mm surface mount 16-pin leadless chip carrier (LCC). Applications for the HMC5883L include Mobile Phones, Netbooks, Consumer Electronics, Auto Navigation Systems, and Personal Navigation Devices.



The HMC5883L utilizes Honeywell's Anisotropic Magnetoresistive (AMR) technology that provides advantages over other magnetic sensor technologies. These anisotropic, directional sensors feature precision in-axis sensitivity and linearity. These sensors' solid-state construction with very low cross-axis sensitivity is designed to measure both the direction and the magnitude of Earth's magnetic fields, from milli-gauss to 8 gauss. Honeywell's Magnetic Sensors are among the most sensitive and reliable low-field sensors in the industry.

The Wire library

```
void setup() {
   Wire.begin();
}
```

The Wire library

```
void loop() {
    …
    Wire.beginTransmission(ADDRESS);
    Wire.write(register);
    Wire.write(data);
    Wire.endTransmission();
    …
}
```

The Wire library

Register List

The table below lists the registers and their access. All address locations are 8 bits.

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

Table2: Register List

Register Access

This section describes the process of reading from and writing to this device. The devices uses an address pointer to indicate which register location is to be read from or written to. These pointer locations are sent from the master to this slave device and succeed the 7-bit address (0x1E) plus 1 bit read/write identifier, i.e. 0x3D for read and 0x3C for write.

The Wire library

Register List

The table below lists the registers and their access. All address locations are 8 bits.

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

Table2: Register List

Register Access

This section describes the process of reading from and writing to this device. The devices uses an address pointer to indicate which register location is to be read from or written to. These pointer locations are sent from the master to this slave device and succeed the 7-bit address (0x1E) plus 1 bit read/write identifier, i.e. 0x3D for read and 0x3C for write.

The Wire library

Configuration Register B

The configuration register B for setting the device gain. CRB0 through CRB7 indicate bit locations, with *CRB* denoting the bits that are in the configuration register. CRB7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. CRB default is 0x20.

CRB7	CRB6	CRB5	CRB4	CRB3	CRB2	CRB1	CRB0
GN2 (0)	GN1 (0)	GN0 (1)	(0)	(0)	(0)	(0)	(0)

Table 7: Configuration B Register

The Wire library

Configuration Register B

The configuration register B for setting the device gain. CRB0 through CRB7 indicate bit locations, with *CRB* denoting the bits that are in the configuration register. CRB7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. CRB default is 0x20.

CRB7	CRB6	CRB5	CRB4	CRB3	CRB2	CRB1	CRB0
GN2 (0)	GN1 (0)	GN0 (1)	(				

Table 7

GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/ Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)
0	1	0	± 1.9 Ga	820	1.22	0xF800–0x07FF (-2048–2047)
0	1	1	± 2.5 Ga	660	1.52	0xF800–0x07FF (-2048–2047)
1	0	0	± 4.0 Ga	440	2.27	0xF800–0x07FF (-2048–2047)
1	0	1	± 4.7 Ga	390	2.56	0xF800–0x07FF (-2048–2047)
1	1	0	± 5.6 Ga	330	3.03	0xF800–0x07FF (-2048–2047)
1	1	1	± 8.1 Ga	230	4.35	0xF800–0x07FF (-2048–2047)

Table 9: Gain Settings

The Wire library

Configuration Register B

The configuration register B for setting the device gain. CRB0 through CRB7 indicate bit locations, with *CRB* denoting the bits that are in the configuration register. CRB7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. CRB default is 0x20.

CRB7	CRB6	CRB5	CRB4	CRB3	CRB2	CRB1	CRB0
GN2 (0)	GN1 (0)	GN0 (1)	(				

Table 7

GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/ Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)
0	1	0	± 1.9 Ga	820	1.22	0xF800–0x07FF (-2048–2047)
0	1	1	± 2.5 Ga	660	1.52	0xF800–0x07FF (-2048–2047)
1	0	0	± 4.0 Ga	440	2.27	0xF800–0x07FF (-2048–2047)
1	0	1	± 4.7 Ga	390	2.56	0xF800–0x07FF (-2048–2047)
1	1	0	± 5.6 Ga	330	3.03	0xF800–0x07FF (-2048–2047)
1	1	1	± 8.1 Ga	230	4.35	0xF800–0x07FF (-2048–2047)

Table 9: Gain Settings

The Wire library

```
#define ADDRESS 0x1E

void loop() {
  …
  Wire.beginTransmission(ADDRESS);
  Wire.write(0x01); // select configuration register B
  Wire.write(0x20); // set range to +- 1.3Ga
  C = 1.e2/1090.;
  Wire.endTransmission();
  …
}
```

The Wire library

```
void loop() {
    …
    Wire.requestFrom(ADDRESS, nBytes);
    for (i = 0; i < nBytes; i++) {
        data[i] = Wire.read();
    }
    …
}
```

The Wire library

Register List

The table below lists the registers and their access. All address locations are 8 bits.

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

Table2: Register List

Register Access

This section describes the process of reading from and writing to this device. The devices uses an address pointer to indicate which register location is to be read from or written to. These pointer locations are sent from the master to this slave device and succeed the 7-bit address (0x1E) plus 1 bit read/write identifier, i.e. 0x3D for read and 0x3C for write.

The Wire library

Register List

The table below lists the registers and their access. All address locations are 8 bits.

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

Table2: Register List

Register Access

This section describes the process of reading from and writing to this device. The devices uses an address pointer to indicate which register location is to be read from or written to. These pointer locations are sent from the master to this slave device and succeed the 7-bit address (0x1E) plus 1 bit read/write identifier, i.e. 0x3D for read and 0x3C for write.

The Wire library

```
void loop() {
    …
    Wire.requestFrom(ADDRESS, nBytes);
    for (i = 0; i < nBytes; i++) {
        data[i] = Wire.read();
    }
    …
}
```

The Wire library

```
void loop() {
  …
  Wire.beginTransmission(ADDRESS);
  Wire.write(0x03);
  Wire.endTransmission();
  Wire.beginTransmission(ADDRESS);
  Wire.requestFrom(ADDRESS,6);
  while (!(Wire.available())) {};
  int x = (Wire.read() << 8) | Wire.read();
  int z = (Wire.read() << 8) | Wire.read();
  int y = (Wire.read() << 8) | Wire.read();
  Wire.endTransmission();
  …
}
```

The Wire library

```
void loop() {
  …
  int x = (Wire.read() << 8) | Wire.read();
  int z = (Wire.read() << 8) | Wire.read();
  int y = (Wire.read() << 8) | Wire.read();
  Wire.endTransmission();
  float Bx = C*x;
  float Bz = C*z;
  float By = C*y;
  float B = sqrt(Bx * Bx + By * By + Bz * Bz);
  …
}
```

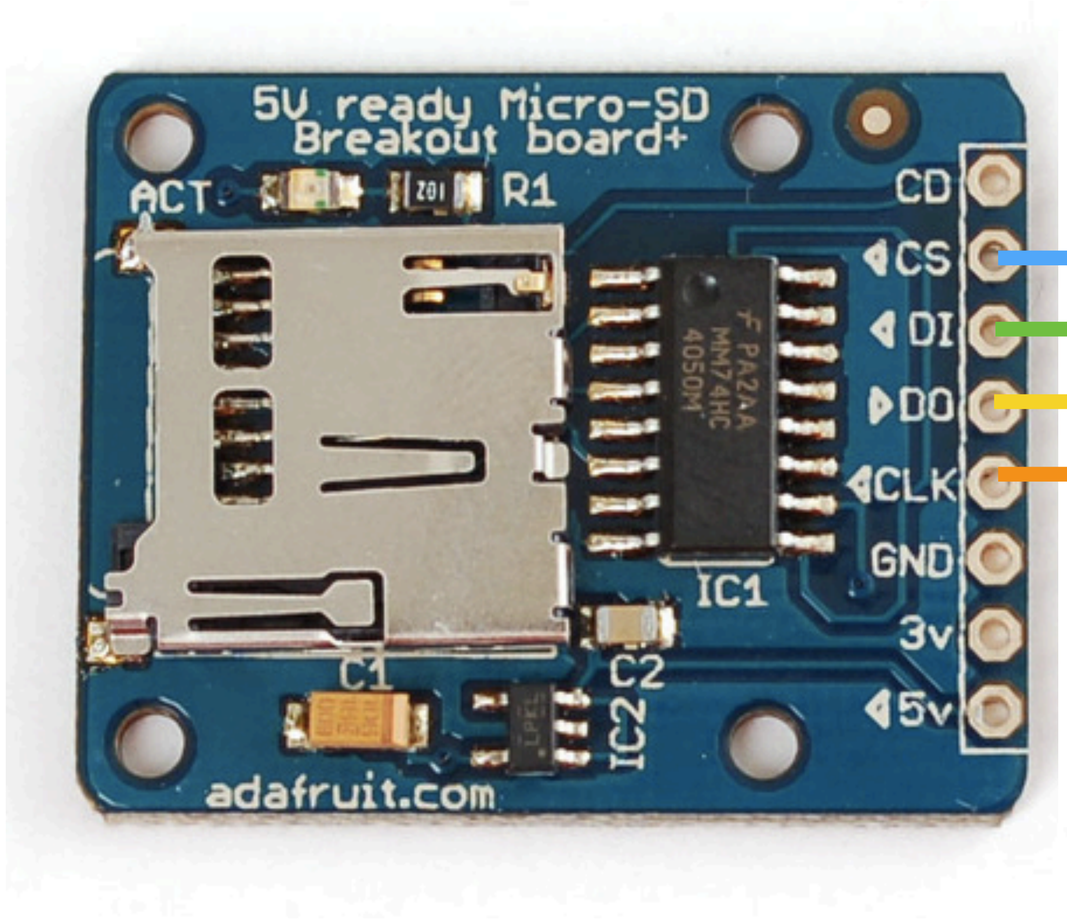
The Wire library

```
#include <HMC5883L.h>
HMC5883L compass;

void setup() {
  Wire.begin(); // Start the I2C interface.
  compass.setScale(1.3);
}

void loop() {
  …
  MagnetometerScaled B = compass.readScaledAxis();
  float Bx = B.XAxis;
  …
}
```

Not only I²C

- Other serial protocols exist/can be implemented
  - SPI: Serial Peripheral Interface
  - Uses 4 lines: SCK (Serial Clock), MISO (Master In Slave Out), MOSI (Master Out Slave In) and optionally SS (Slave Select)
- Other libraries exist/can be implemented
  - #include <SPI.h>
  - SPI.begin();
  - SPI.transfer()

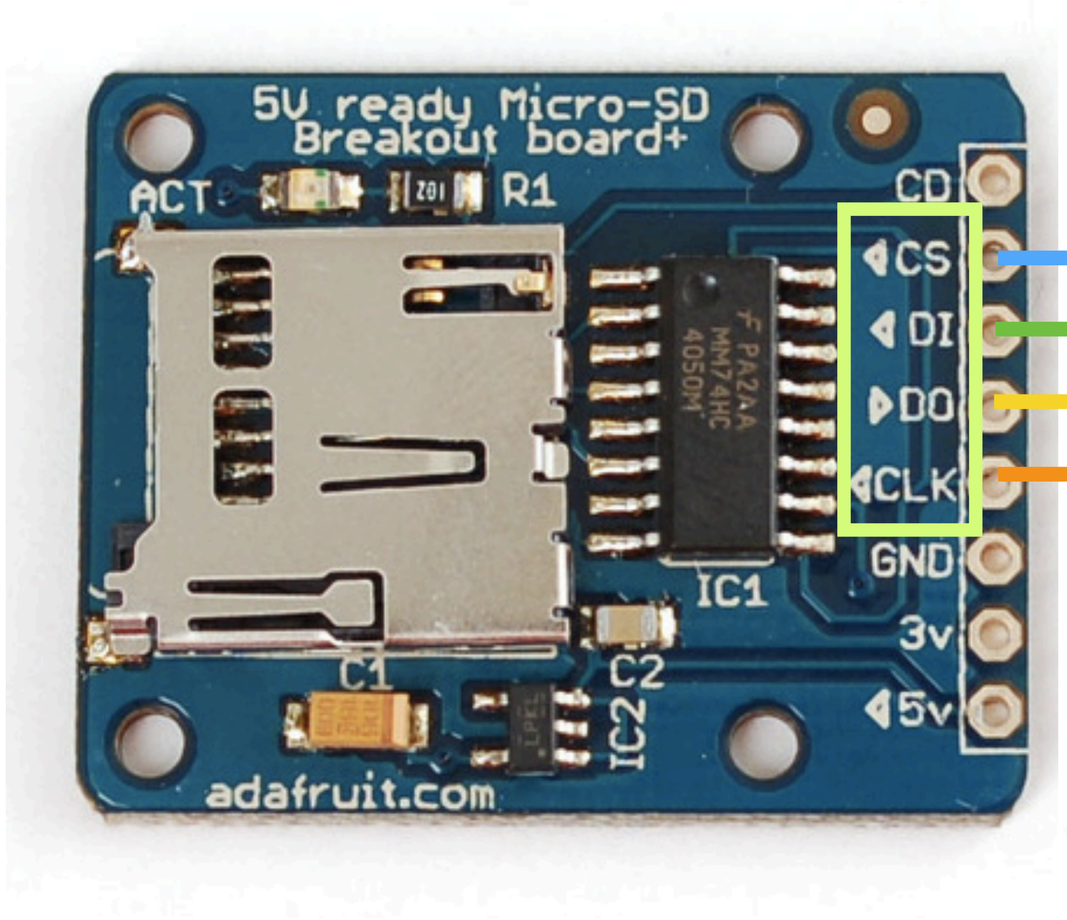
An SPI device



Arduino UNO ports

SS	10
MOSI	11
MISO	12
SCK	13

An SPI device



Arduino UNO ports

SS	10
MOSI	11
MISO	12
SCK	13

Simplified usage

- The SD library

```
File f;

void setup() {
  …
  SD.begin(10);
  f.open(“filename.txt”, FILE_WRITE);
  …
}
```

Simplified usage

- The SD library

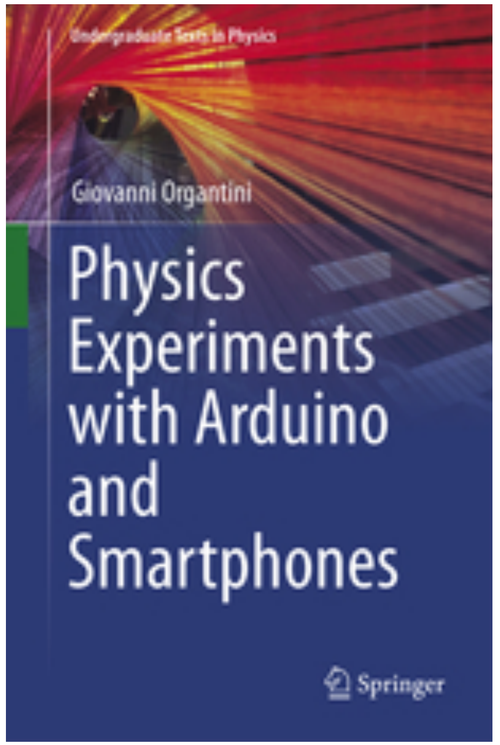
```
void loop() {
  …
  f.println(“a string”);
  f.println(variable);
  …
  f.close();
}
```

Resources


Giovanni Organtini
Fisica
con Arduino
Dalla fisica
ad Arduino,
e ritorno
Un laboratorio
alla portata di tutti
In laboratorio come
un professionista
File di lavoro
SCIENZE ZANICHELLI

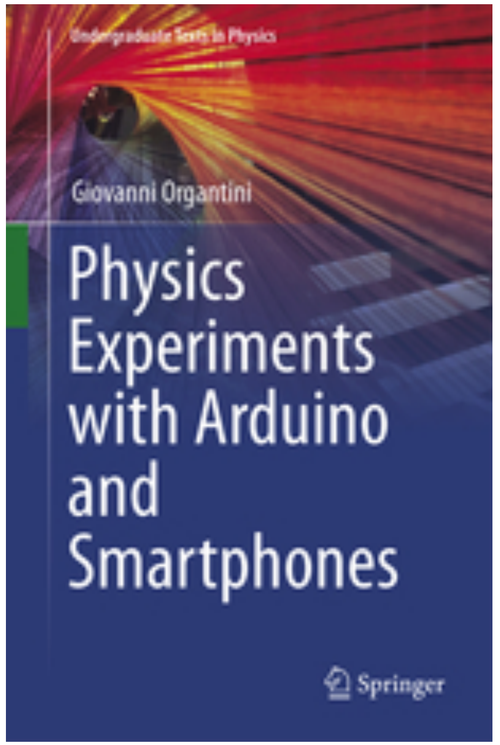
Resources


Giovanni Organtini
Fisica
con Arduino
Dalla fisica
ad Arduino,
e ritorno
Un laboratorio
alla portata di tutti
In laboratorio come
un professionista
File di lavoro
SCIENZE ZANICHELLI


Giovanni Organtini
Physics
Experiments
with Arduino
and
Smartphones
Springer

Resources


Giovanni Organtini
Fisica
con Arduino
Dalla fisica
ad Arduino,
e ritorno
Un laboratorio
alla portata di tutti
In laboratorio come
un professionista
File di lavoro
SCIENZE ZANICHELLI


Giovanni Organtini
Physics
Experiments
with Arduino
and
Smartphones
Springer


2000
problemi
con soluzioni
Fondamenti
di Fisica
Sesta edizione
James S. Walker
A cura di Giovanni Organtini
MyLab
Codice per accedere
alla piattaforma
Pearson

Have Fun!