

Introduction to Arduino programming

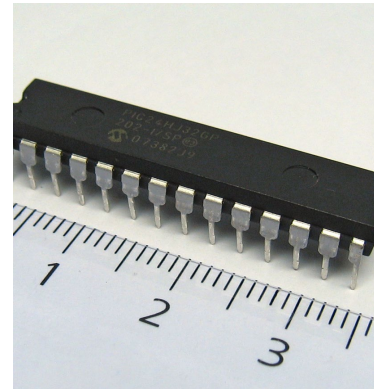
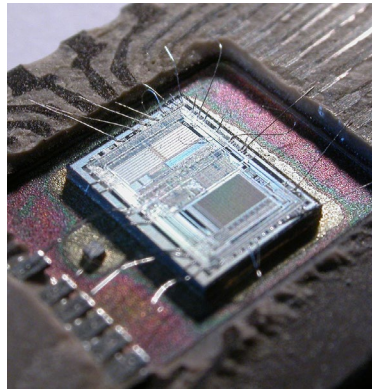
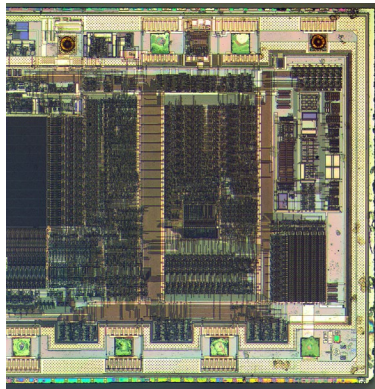
MARCO BARUZZO

ICTP SCIFABLAB

Summary

- ▶ Introduction
- ▶ Interacting with the World
- ▶ Arduino Programming
- ▶ Arduino Family and other microcontroller
- ▶ Some projects

Introduction: What is a microcontroller:



Small computing unit equipped with all the required hardware (RAM, ROM, bus, IO) to perform real-time tasks designed for embedded applications

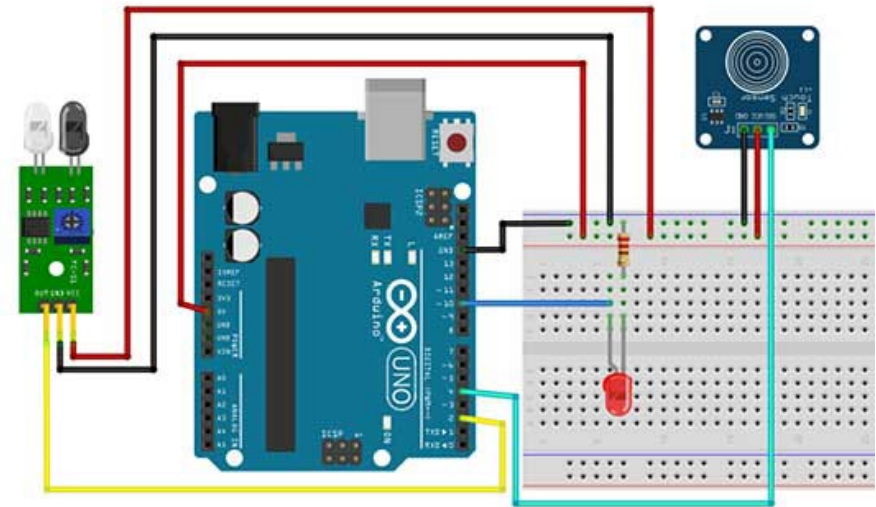
(It's different to microprocessor (CPU) which has to be supplied with all the additional hardware)

Introduction: What is a microcontroller:

Small computing unit equipped with all the required hardware (RAM, ROM, bus, IO) to perform real-time tasks designed for embedded applications

Standard example on the use of microcontrollers:

- Connecting to sensors
- Controlling devices
- Automatizing tasks
- Doing repetitive tasks!

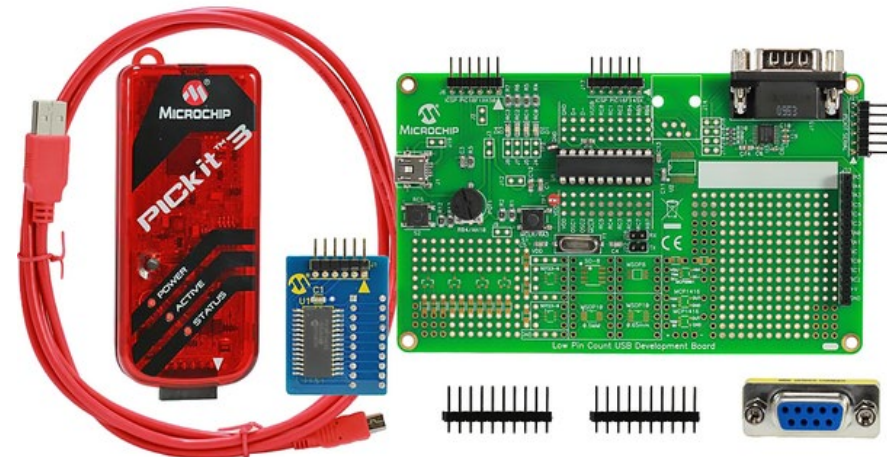


Introduction: Programming microcontroller before Arduino

- Preparing your own board and programming interface with the microcontroller
- Usually assembly coding (not every case)
- Lack of libraries
- Lack of support

Because of these reasons:

- Not really fast for R&D
- Not encouraging for the general public

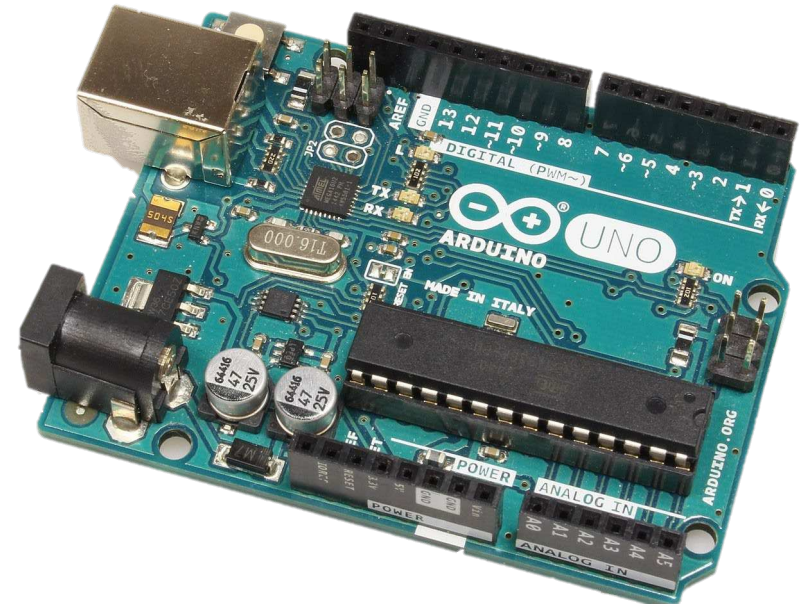


Introduction: The Arduino breakthrough

- ~~Preparing your own board and programming interface with the microcontroller~~
- Arduino provides you a standard board with programming interface and compatible with lots of devices
- ~~Usually assembly coding (not every case)~~
- Simplified C++ coding
- ~~Lack of libraries~~
- A HUGE selection of libraries
- ~~Lack of support~~
- HUGE and supportive community!

Because of these reasons:

- Super fast R&D
- A new gold era of homemade projects

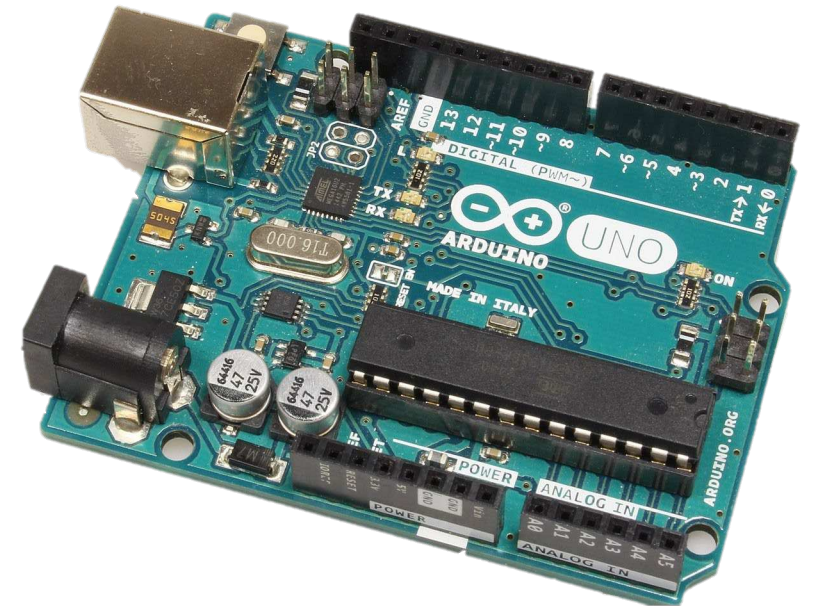


Introduction: Arduino

It is an electronic board hosting a microcontroller.

This board permits you to:

- Connect in a simple way the microprocessor with the PC, using serial-USB converter, in order to be programmed with the firmware
- Power electronics
- Connect to the digital and analog pin of the microcontroller via jumper cable

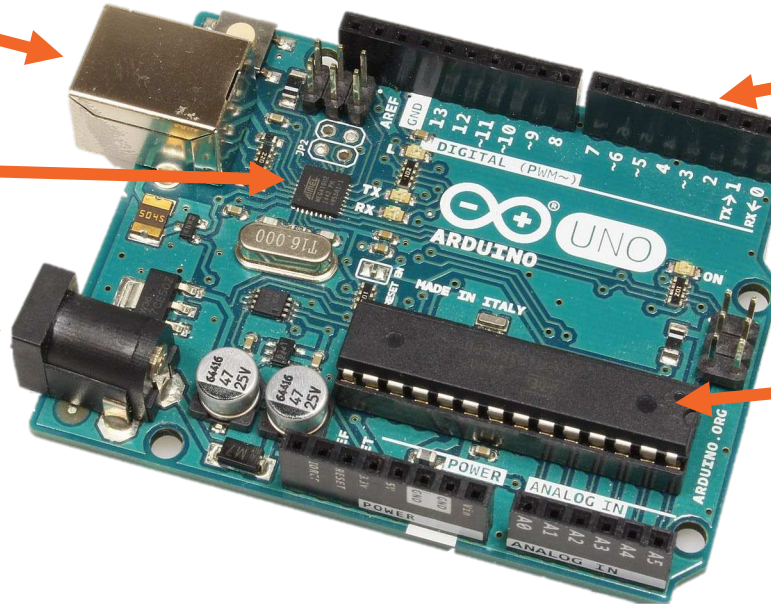


Introduction: Anatomy of an Arduino board

Programming
USB

USB interface

Power
connector



IO Pins

ATMEGA
microcontroller

Major features on board

Introduction:

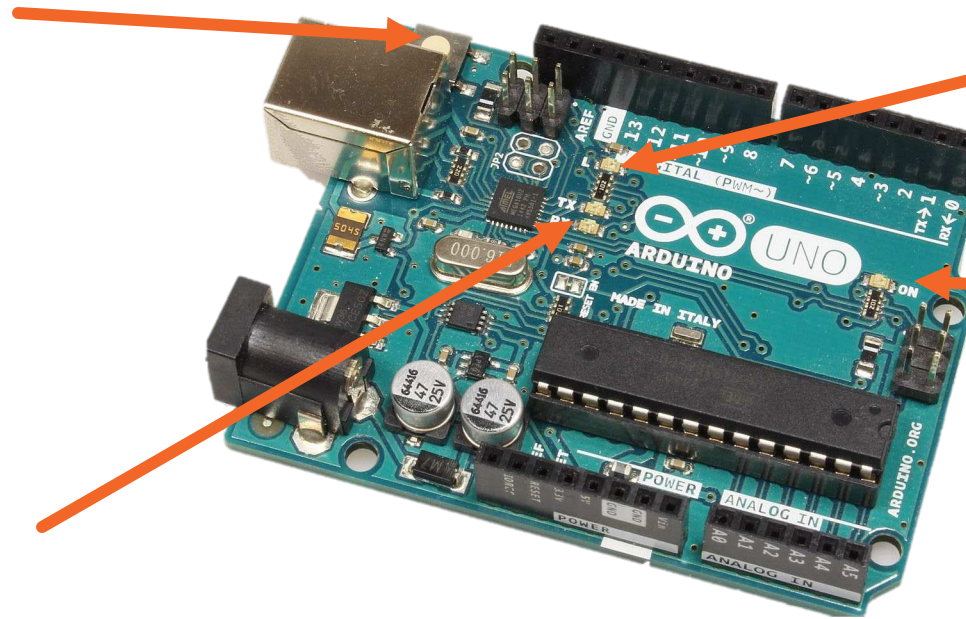
Anatomy of an Arduino board

Reset button
(restart code)

LED connected
to pin 13

ON indicator

Indicates Rx/Tx
connections on
Serial Port



LED indicator and button

Introduction: Powering Arduino

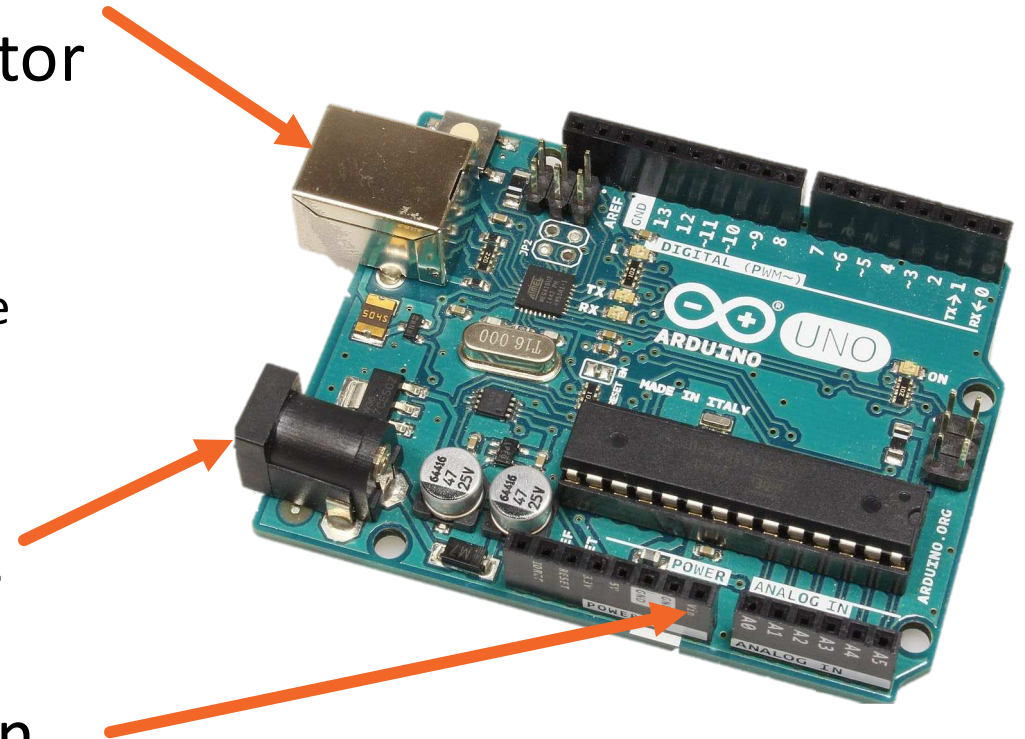
Arduino can be powered via:

- Via USB connector (with PC or USB charger)
- Via power connector (connected to internal voltage regulator input 7-12 V)
- Via Vin pin (only regulated 5 V)

USB
Connector

Power
Connector

Vin pin



Interaction with the World: Arduino senses and arms: IO pins

USB

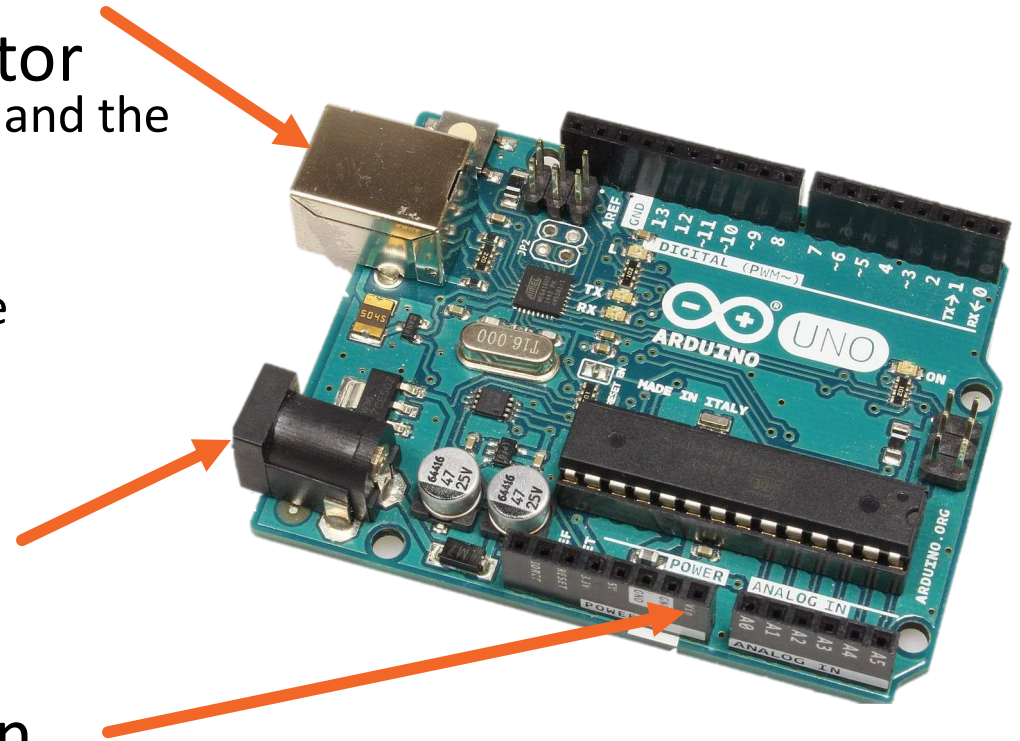
Connector

An I/O pin is an interface between the microcontroller and the external world, these can be:

- Via USB connector (with PC or USB charger)
- Via power connector (connected to internal voltage regulator input 7-12 V)
- Via Vin pin (only regulated 5 V)

Power
Connector

Vin pin

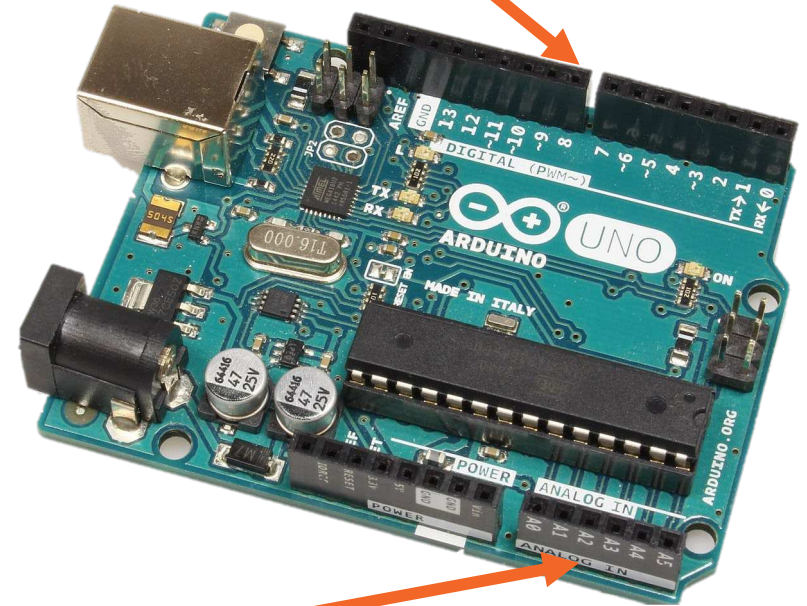


Interaction with the World: Arduino senses and arms: IO pins

I/O pins are the interfaces between the microcontroller and the external world, these can be:

- Input
- Output
- Analogic
- Digital
- PWM
- General Purpose

Digital pins



Analog pins

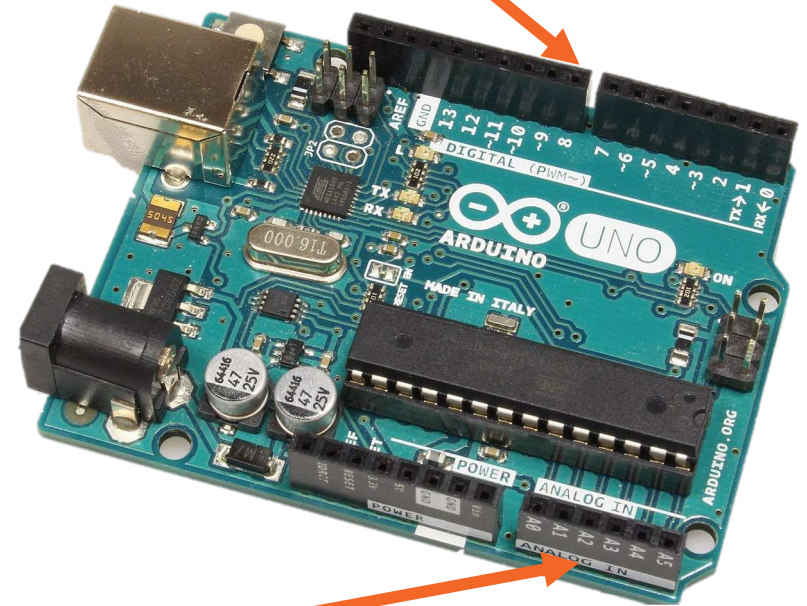
Interaction with the World: Arduino senses and arms: IO pins

The inputs are the way in which the microcontroller acquire information and output are used by the microcontroller to “act” on the external environment

As human we use light and light sensors (eyes) to acquire most of the information.

The microcontroller uses voltage and voltage meter

Digital pins

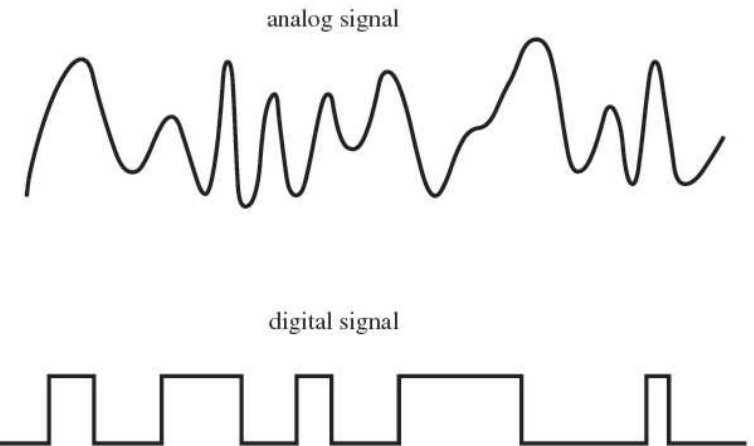


Analog pins

Interaction with the World: Arduino senses and arms: IO pins

Voltage can vary continuously, from 0 to 5 Volts, to give an idea of a continuous value (like the luminosity of a light source). This is called Analogic value

Or voltage can be set in only two “levels”, High and Low, 5 Volts and 0 Volts, to say On or Off (there is light, there’s not light). This is called Digital value

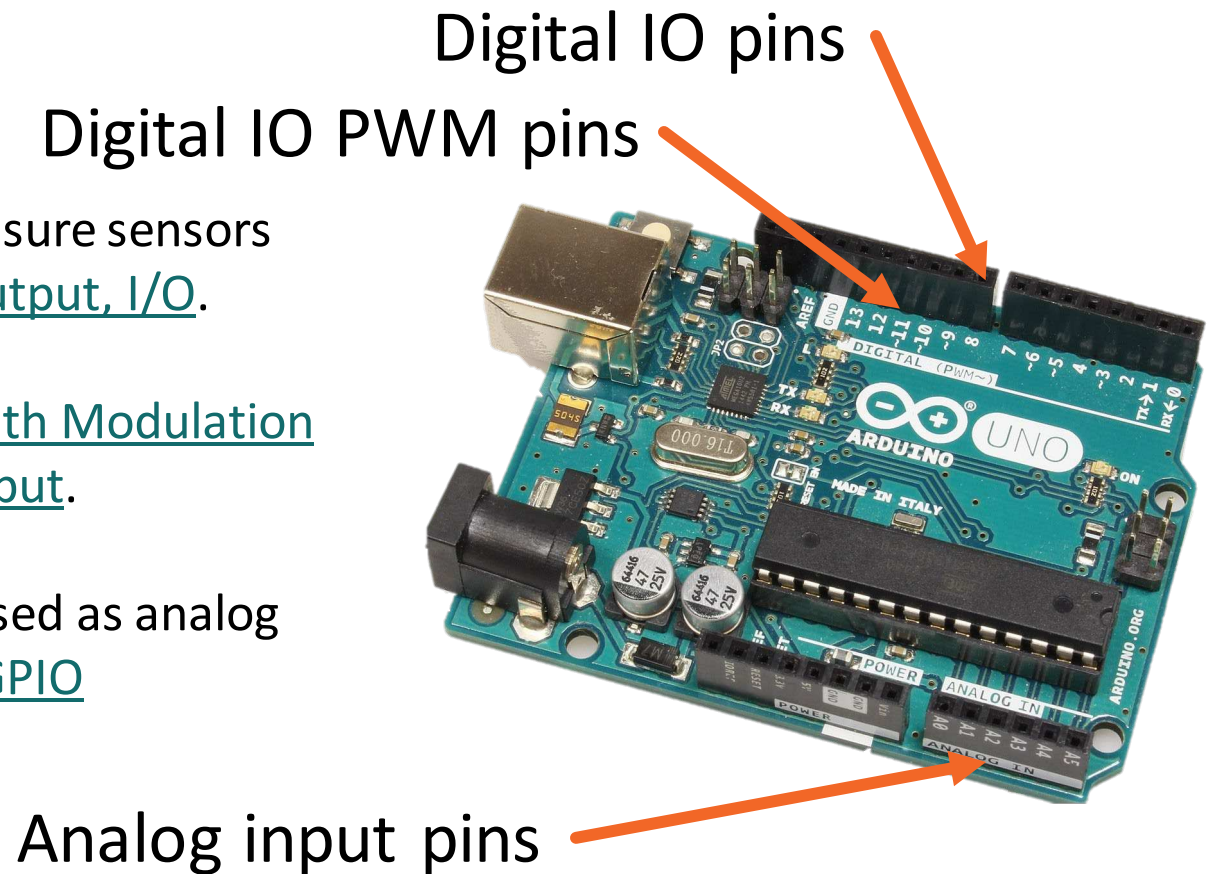


Interaction with the World: Arduino senses and arms: IO pins

Most Arduinos have a set of [analog inputs](#) to measure sensors and a set of [digital inputs](#) that can work also as [output, I/O](#).

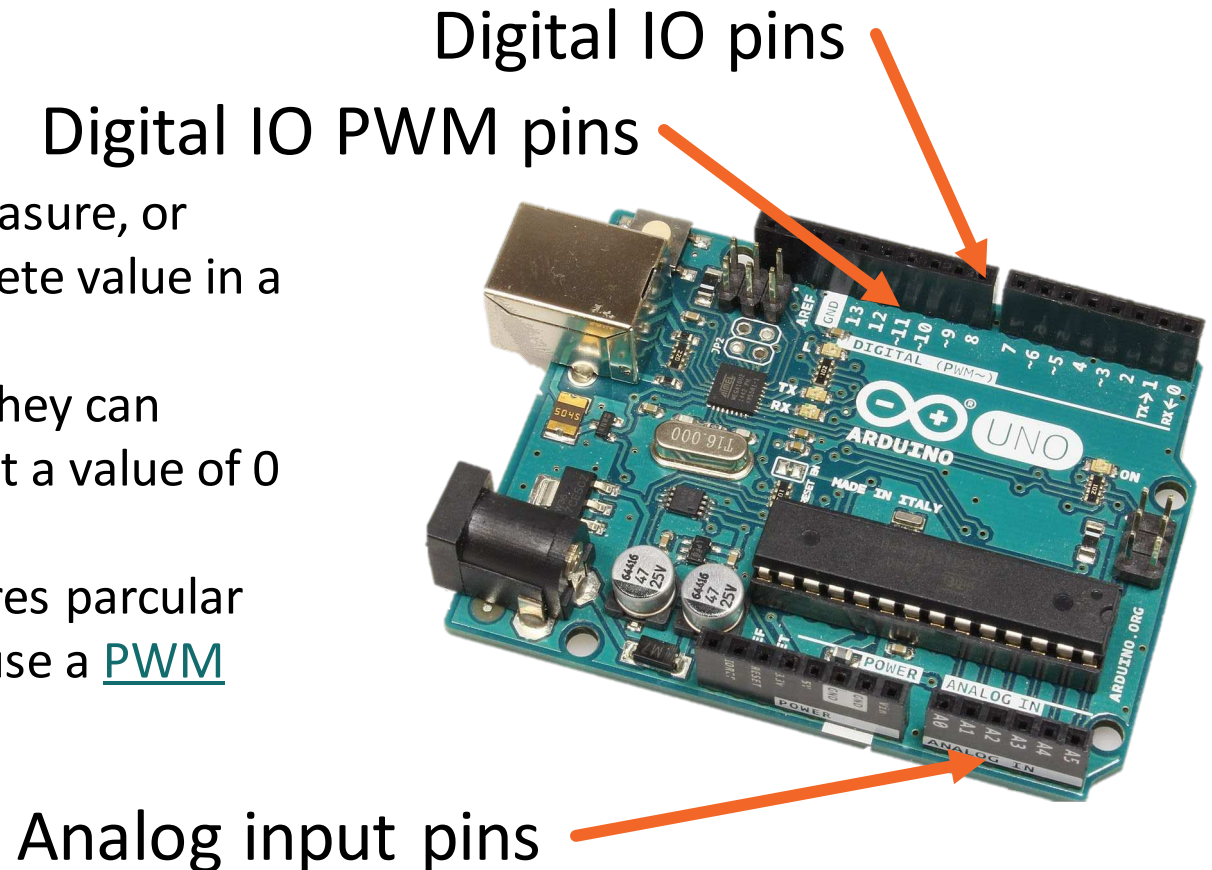
Some digital output can also perform a [Pulse-Width Modulation or PWM](#). This technique simulates an [Analog Output](#).

Pins on most advanced microcontrollers can be used as analog or digital and are called [General Purpose I/O, or GPIO](#)

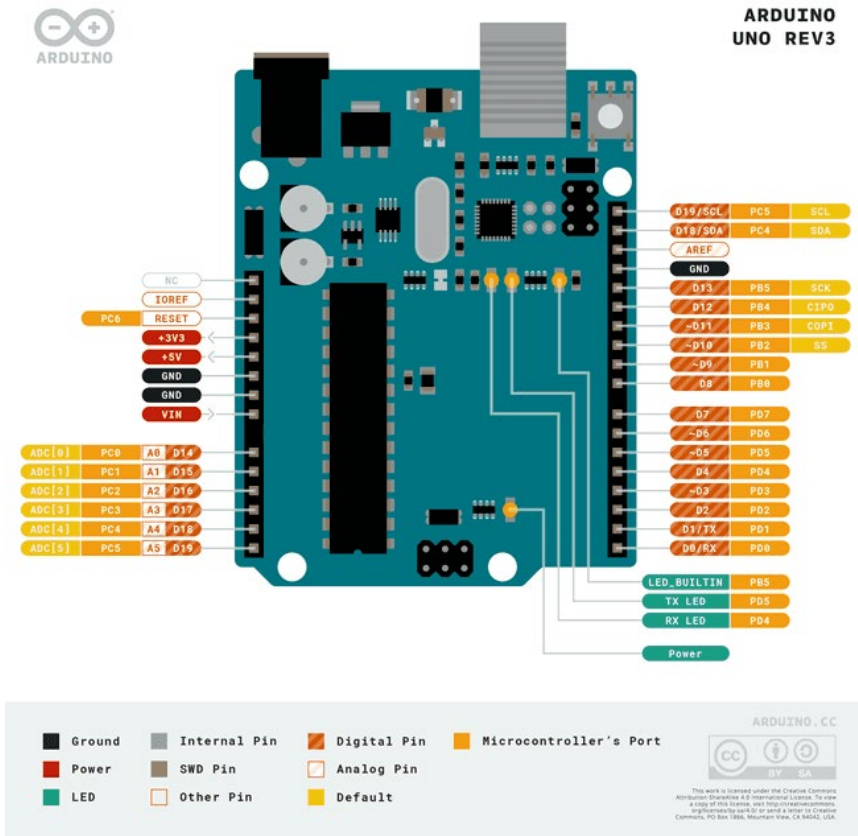


Interaction with the World: IO pins recap

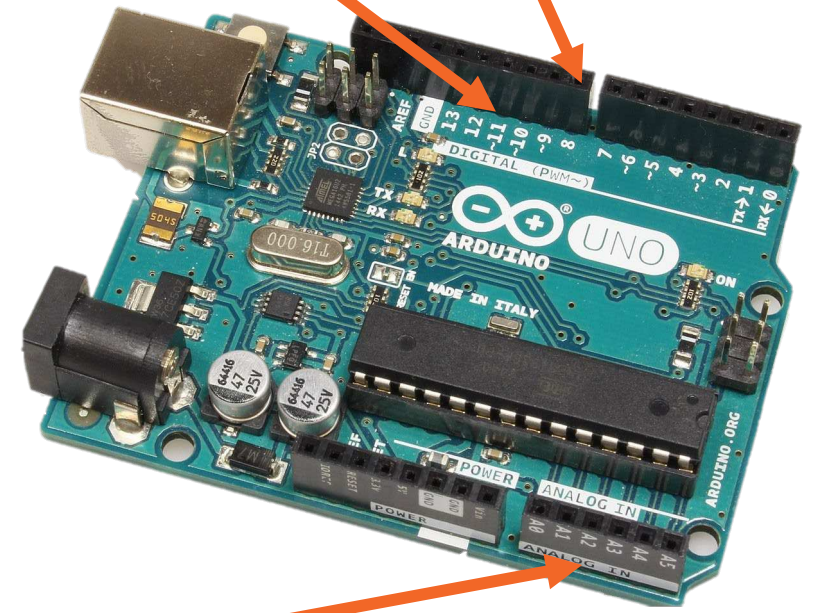
- [Analog pins](#) are usually only inputs and can measure, or convert, a continuous analog signal into a discrete value in a specific range
- [Digital pins](#) usually can be inputs and output. They can measure if a value is 0 or 5V or they can put out a value of 0 or 5V.
- [Analog output](#) are really uncommon and requires particular devices. To simulate an analog output we can use a [PWM](#) output.



Interaction with the World: IO pins recap



Digital IO pins
Digital IO PWM pins

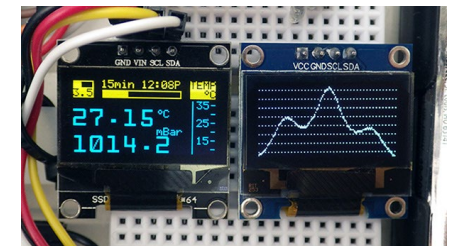
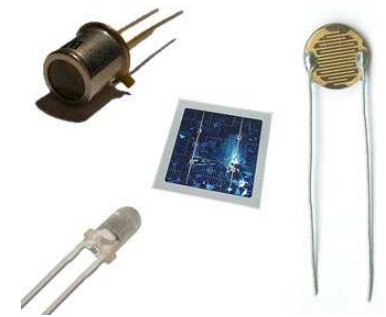
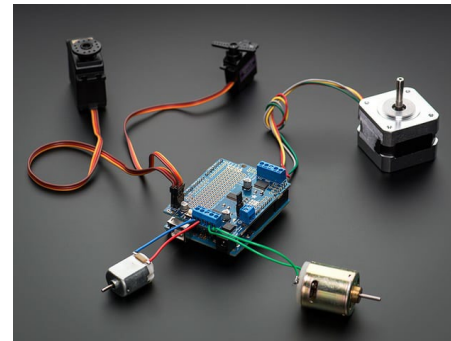


Analog input pins

Interaction with the World: Using pins: connecting devices

- LED =)
- A lot of different types of sensors (Temperature, Light)
- Controllers (Buttons, Analog stick, Touch)
- Motors (DC motors, Steppers, Servo)
- Relays (Microcontroller controlled switches)
- Screens (LCD, OLED)
- Speakers (Not so simple)
- Memory (SD card)

And we can use them automatically



Interaction with the World:

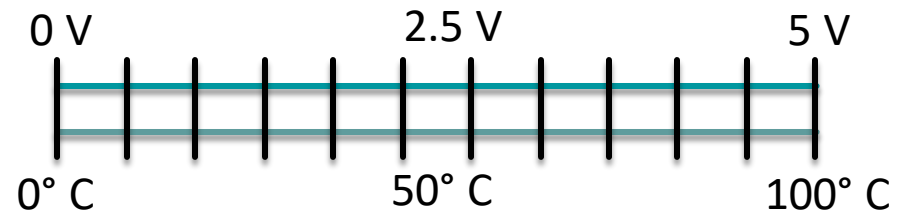
Analog sensors

- Temperature
- Humidity
- Pressure
- Light
- Strain
- Weight
- Accelerometer
- Movement
- Distance
- Presence
- Sound (microphone)
-

Sensors can be analogic or digital.

Analogic means that the sensor will produce a voltage “proportional” to the value that you are measuring with the sensor.

This means that you have to measure the voltage with an analog pin and calculate the proportion to have the value of the temperature.
Choosing the relation between voltage and temperature is called calibration.



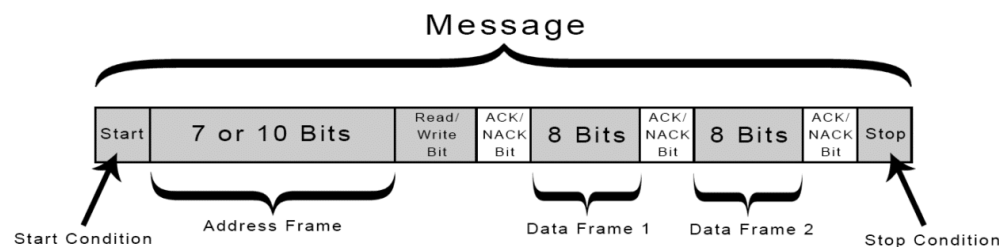
Interaction with the World:

Digital sensors

- Temperature
- Humidity
- Pressure
- Light
- Strain
- Weight
- Accelerometer
- Movement
- Distance
- Presence
- Sound (microphone)
-

Digital sensors have internally another small controller that measure and do all the math required to have the result, but you need to connect/collect the controller with to the Arduino through a digital.

The values are encoded in a series of 0 and 1 in a particular format that you need to decode. This is a [communication protocol](#) (More on this later)



Interaction with the World: LEDs

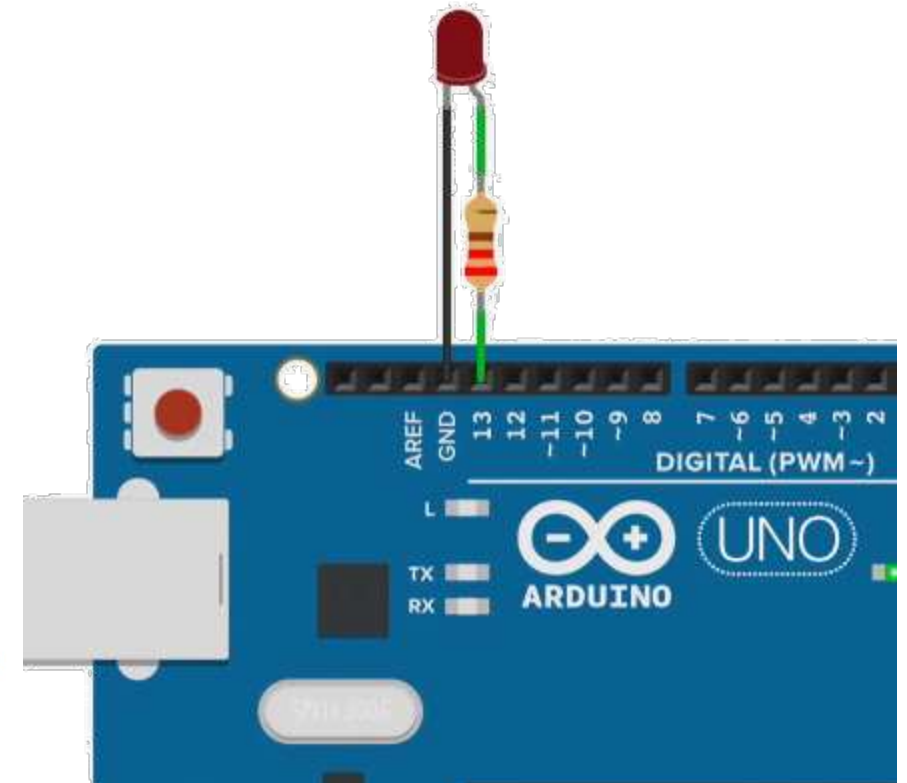
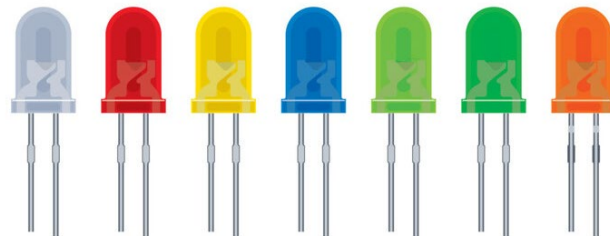
LED are really simple to be used with Arduino.

To light up a LED you need to:

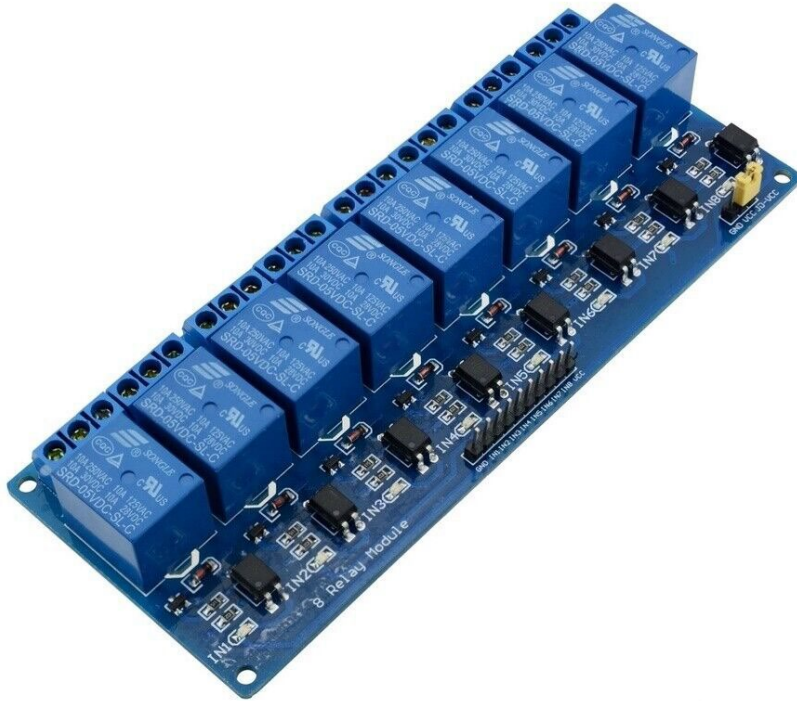
- Connect the negative wire of the LED to GND on the Arduino
- Connect the positive wire of the LED a digital pin with a resistor in series (to limit the current flowing into the LED, usually around 300Ω)

Write the code to use that pin as a digital output and set it to 5V

Tips: if it not working probably you need to invert the LED wires=)



Interaction with the World: Relays

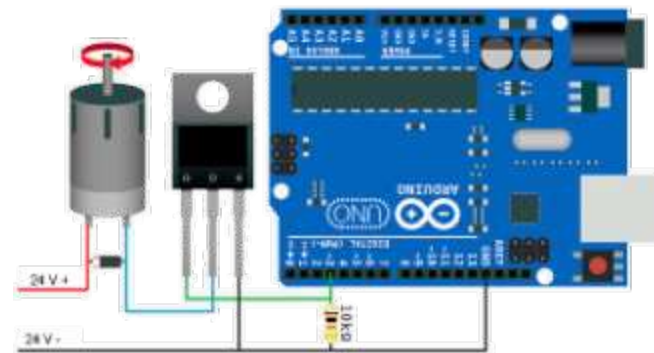
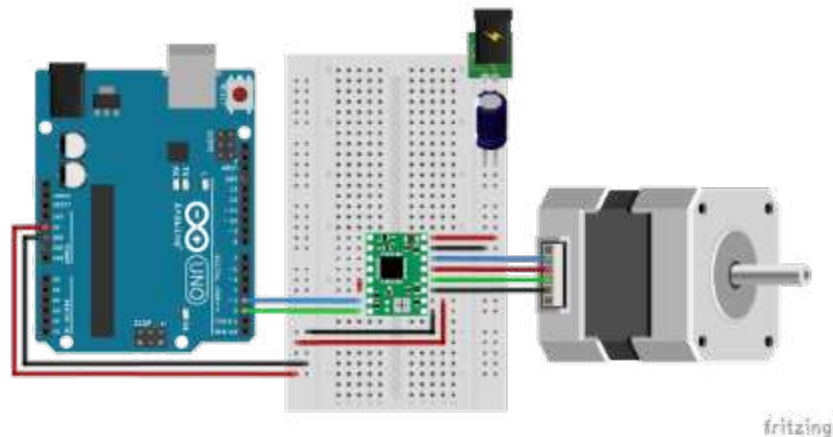
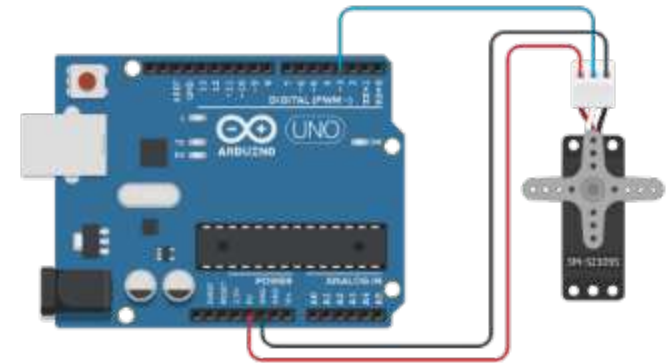


Relays are very useful to control other devices and more powerful lamps, some motors, or any other device that can be used with a switch. Also in this case you can control it directly with a digital pin

Interaction with the World: Motors

The most common motors to be connected to Arduino are:

- Servomotor: it can do precise repeatable movements and, in case of small servos, can be powered and used directly with an Arduino
- DC motors: it's a very fast and powerful motor, but no precise small movement. It required additional controller circuit (MOSFET, relay,...)
- Stepper motors: powerful, precise and repeatable but it requires another driver circuit that can be controlled by an Arduino



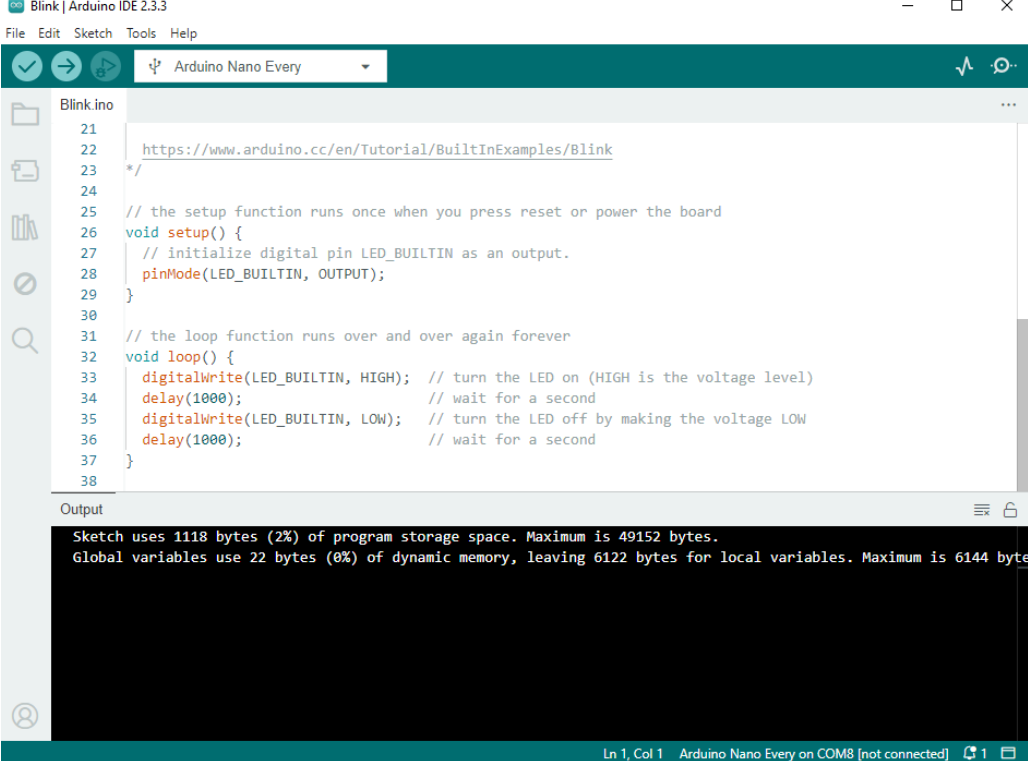
Arduino Programming: Connecting Arduino to PC: Arduino IDE

Important part of the Arduino ecosystem:
Arduino Integrated Development Environment (IDE).

This is a PC program can:

- Connect to Arduino (simple driver managing)
- Editor where to write your code
- Compile the code for the specific model of Arduino
- Upload the code in the microcontroller
- Monitor in real-time the Arduino Serial Port
- Full list of code examples

You can download it for free from the Arduino webpage



The screenshot displays the Arduino IDE 2.3.3 interface. The main editor window shows the code for the 'Blink.ino' sketch. The code includes a comment linking to the Arduino website, a setup function that initializes the built-in LED pin as an output, and a loop function that toggles the LED on and off with 1000ms delays. The Output window at the bottom shows the compilation status: 'Sketch uses 1118 bytes (2%) of program storage space. Maximum is 49152 bytes. Global variables use 22 bytes (0%) of dynamic memory, leaving 6122 bytes for local variables. Maximum is 6144 bytes.' The status bar at the bottom indicates 'Ln 1, Col 1' and 'Arduino Nano Every on COM8 [not connected]'.

```
Blink | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Arduino Nano Every

Blink.ino
21
22 https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
38

Output
Sketch uses 1118 bytes (2%) of program storage space. Maximum is 49152 bytes.
Global variables use 22 bytes (0%) of dynamic memory, leaving 6122 bytes for local variables. Maximum is 6144 bytes

Ln 1, Col 1 Arduino Nano Every on COM8 [not connected]
```


Arduino Programming: Connecting Arduino to PC: Arduino IDE

- Managing driver and compilers for all Arduino (and not only models)
- Library manager
- Serial Port Monitor
- Editor
- Compiler Output

```
Blink | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Arduino Nano Every

Blink.ino
21
22 https://www.arduino.cc/en/Tutorial/Blink#examples/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
38

Output
Sketch uses 1118 bytes (2%) of program storage space. Maximum is 49152 bytes.
Global variables use 22 bytes (0%) of dynamic memory, leaving 6122 bytes for local variables. Maximum is 6144 bytes.
```

Ln 1, Col 1 Arduino Nano Every on COM8 [not connected]

Arduino Programming:

Arduino IDE

Arduino IDE from my PC:

- Adding a new board
- Adding a new library
- Opening a code example
- Compiling

Arduino Programming: Arduino language

Arduino IDE supports a C++ with simplified syntax as programming language



```
--
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
```

But it has a lot of other functions and variables that are not standard C++, why?

Arduino Programming: Arduino language

Because you have to control pins and other stuff in real-time....

First things to be noted:

- `void setup()`
This is the first part of the code that the microprocessor will run only one time when it is switched on. Usually it runs the initialization of hardware and objects.
- `void loop(pin, value)`
This loop contains all the instructions that will run continuously indefinitely until the microprocessor will be powered off

```
--  
25 // the setup function runs once when you press reset or power the board  
26 void setup() {  
27     // initialize digital pin LED_BUILTIN as an output.  
28     pinMode(LED_BUILTIN, OUTPUT);  
29 }  
30  
31 // the loop function runs over and over again forever  
32 void loop() {  
33     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
34     delay(1000); // wait for a second  
35     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
36     delay(1000); // wait for a second  
37 }
```

Arduino Programming: Arduino language

Because you have to control pins and other stuff in real-time....

There are of course some strange functions:

- **pinMode**(pin, value)
Used to set one of the Arduino pins in input or output
- **digitalWrite**(pin, value)
Used to change the status of the Arduino pin from HIGH to LOW and vice versa
- **delay**(value)
Used to wait some time before next instruction

```
--  
25 // the setup function runs once when you press reset or power the board  
26 void setup() {  
27     // initialize digital pin LED_BUILTIN as an output.  
28     pinMode(LED_BUILTIN, OUTPUT);  
29 }  
30  
31 // the loop function runs over and over again forever  
32 void loop() {  
33     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
34     delay(1000); // wait for a second  
35     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
36     delay(1000); // wait for a second  
37 }
```

Arduino Programming: Arduino language

Other common Arduino functions are:

- **analogRead**(pin)
Returns a value proportional to the voltage applied to the pin
- **digitalRead**(pin)
Returns 1 or 0 if the voltage applied to the pin is 5V or 0V
- **Serial.Begin**(baudrate)
Open a Serial COM Port
- **Serial.print**(string)
Write on Serial COM Port
- **Serial.read**()
Read on Serial COM Port

```
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34     delay(1000); // wait for a second
35     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36     delay(1000); // wait for a second
37 }
```

Arduino Programming: Arduino language

For example this code:

- In the setup initialize a digital pin to be an output
- In the main loop:
 1. Set the pin to HIGH (5V)
 2. Wait 1 second
 3. Set the pin to LOW (0V)
 4. Wait 1 second
 5. Restart forever

This code blink a LED=)

```
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34     delay(1000); // wait for a second
35     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36     delay(1000); // wait for a second
37 }
```

Arduino Programming: Arduino language

For example this code:

- In the setup initialize a Serial COM Port
- In the main loop:
 1. Read the value from the Analog Pin
 2. Map the value (Calibration)
 3. Write the results in the Serial
 4. Wait 2 millisecond
 5. Restart forever

This is perfect to read an analog temperature sensor

```
23 // These constants won't change. They're used to give names to the pins used:
24 const int analogInPin = A0; // Analog input pin that the potentiometer is attached to
25
26 int sensorValue = 0; // value read from the pot
27 int outputValue = 0; // value output to the PWM (analog out)
28
29 void setup() {
30   // initialize serial communications at 9600 bps:
31   Serial.begin(9600);
32 }
33
34 void loop() {
35   // read the analog in value:
36   sensorValue = analogRead(analogInPin);
37   // map it to the range of the analog out:
38   outputValue = map(sensorValue, 0, 1023, 0, 255);
39
40   // print the results to the Serial Monitor:
41   Serial.print("sensor = ");
42   Serial.print(sensorValue);
43   Serial.print("\t output = ");
44   Serial.println(outputValue);
45
46   // wait 2 milliseconds before the next loop for the analog-to-digital
47   // converter to settle after the last reading:
48   delay(2);
49 }
```

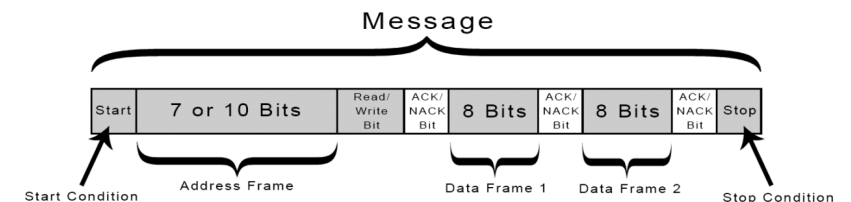

Arduino Programming: Libraries

I mentioned drivers for motors and connection protocols for devices or sensors. Of course these requires a specific code to be run.

A communication protocol is a standard set of rules that describes how two circuits speaks each other.

Common Arduino examples are I2C, SPI, UART, OneWire

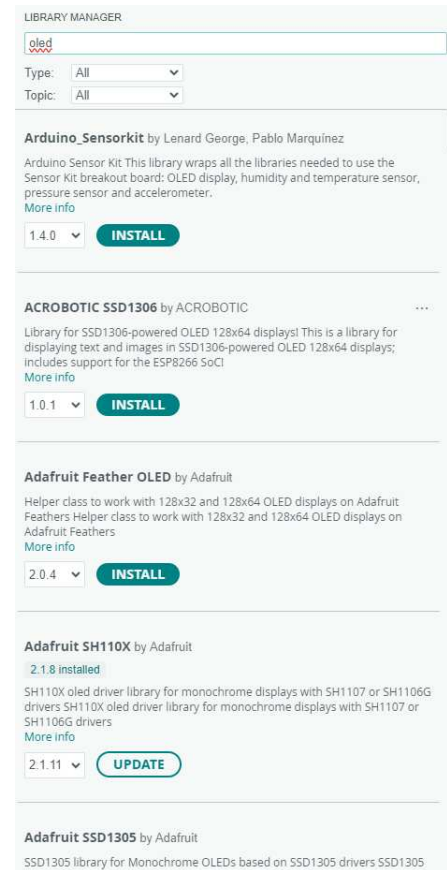
Luckily there are library already prepared for our devices.



Arduino Programming: Libraries

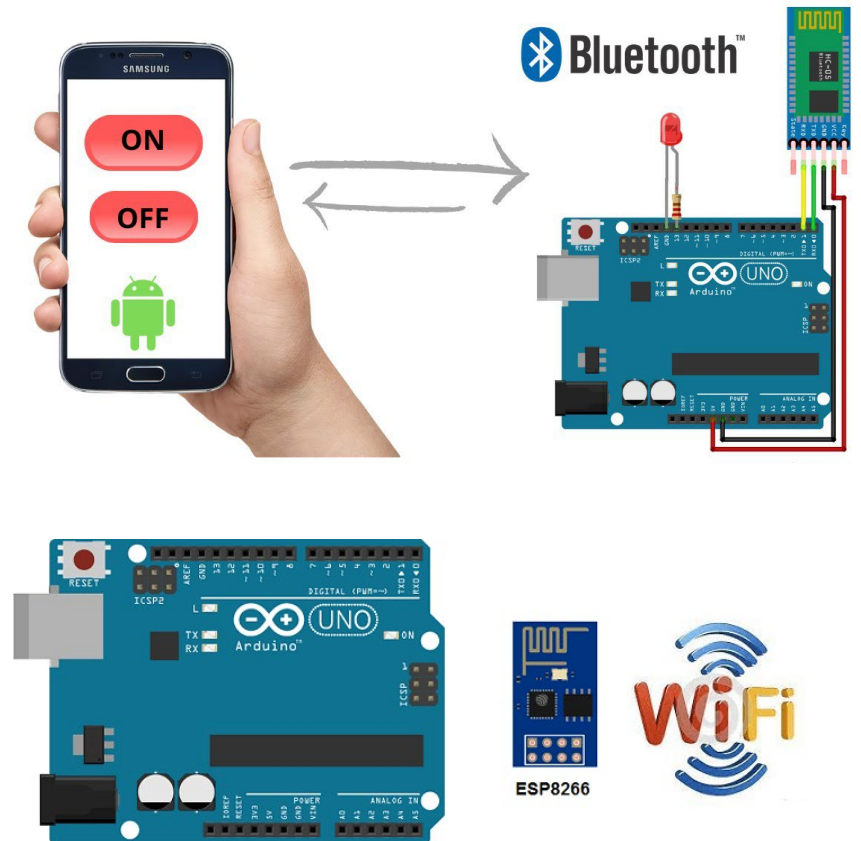
Most of the library can be downloaded directly from the Arduino IDE using the specific tool and searching the sensor or driver that is required.

For most of the library there are also well commented example on how to use the various functions.



Arduino Programming: WiFi and Bluetooth

Some Arduinos can be connected via WiFi or Bluetooth. This can add a lot of features as remote control, remote data logging, etc



Arduino Programming: New features from Arduino

Arduino App

- Arduino App provides a user-friendly interface for controlling and monitoring Arduino projects
- On the App can be added features like data visualization, remote control, and real-time feedback
- Develop apps for iOS, Android, or web platforms to reach a wider audience

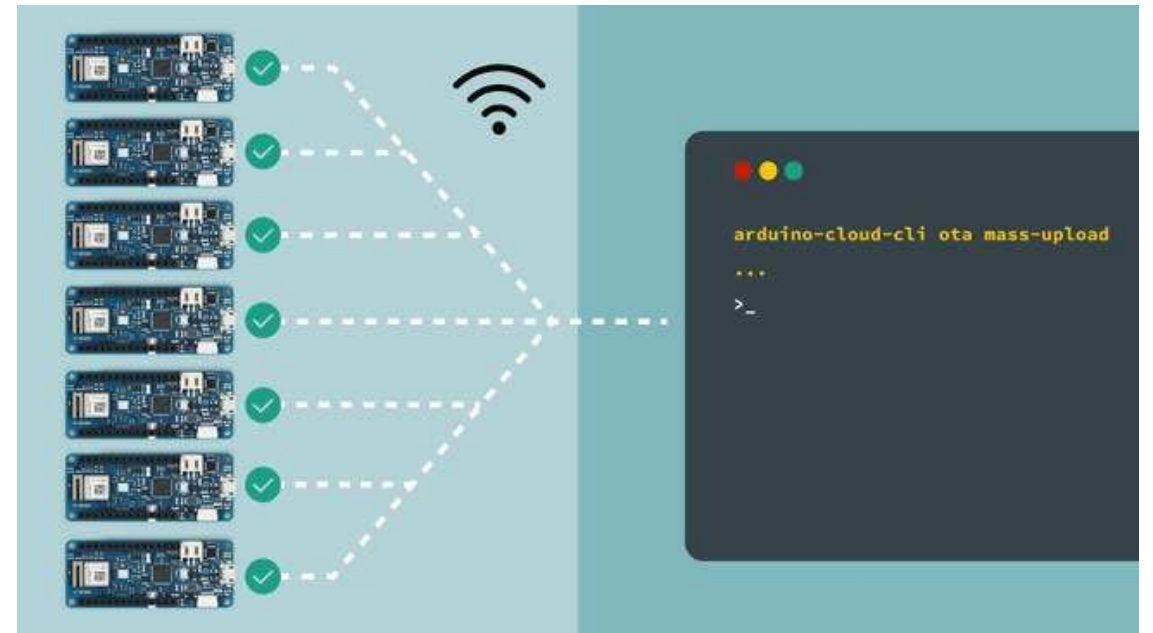


Arduino Programming: New features from Arduino

37

Arduino Cloud

- Simplified IoT Development: user-friendly platform for creating and managing IoT projects.
- Remote Monitoring and Control: Access and control your Arduino devices from remote.
- Data Collection and Analysis
- Integration with Other Services (Google Assistant, Amazon Alexa)
- Over-the-Air Updates of firmware



Arduino Family and other microcontroller

Arduino UNO specifications

- Microcontroller: ATmega328P
- Digital Pin IO: 14
- Analog Pin I: 6
- PWM Pin: 6
- Working voltage: 5V
- Max current on Digital Pin: 20 mA
- Flash Memory: 32KB
- SRAM Memory: 2KB
- EEPROM Memory: 1KB
- Clock: 16 MHz
- Led built-in: D13
- Native USB: no
- ADC: 1x 10bit
- DAC: no
- Time/Counter: 2x 8bit, 1x 16bit



Arduino Family and other microcontroller

Arduino UNO specifications

- Microcontroller: ATmega328P
- Digital Pin IO: 14
- Analog Pin I: 6
- PWM Pin: 6
- Working voltage: 5V
- Max current on Digital Pin: 20 mA
- Flash Memory: 32KB
- SRAM Memory: 2KB
- EEPROM Memory: 1KB
- Clock: 16 MHz
- Led built-in: D13
- Native USB: no
- ADC: 1x 10bit
- DAC: no
- Time/Counter: 2x 8bit, 1x 16bit



A lot of data!

Do we need all of them?

It depends on our project!

Arduino Family and other microcontroller

Arduino UNO specifications

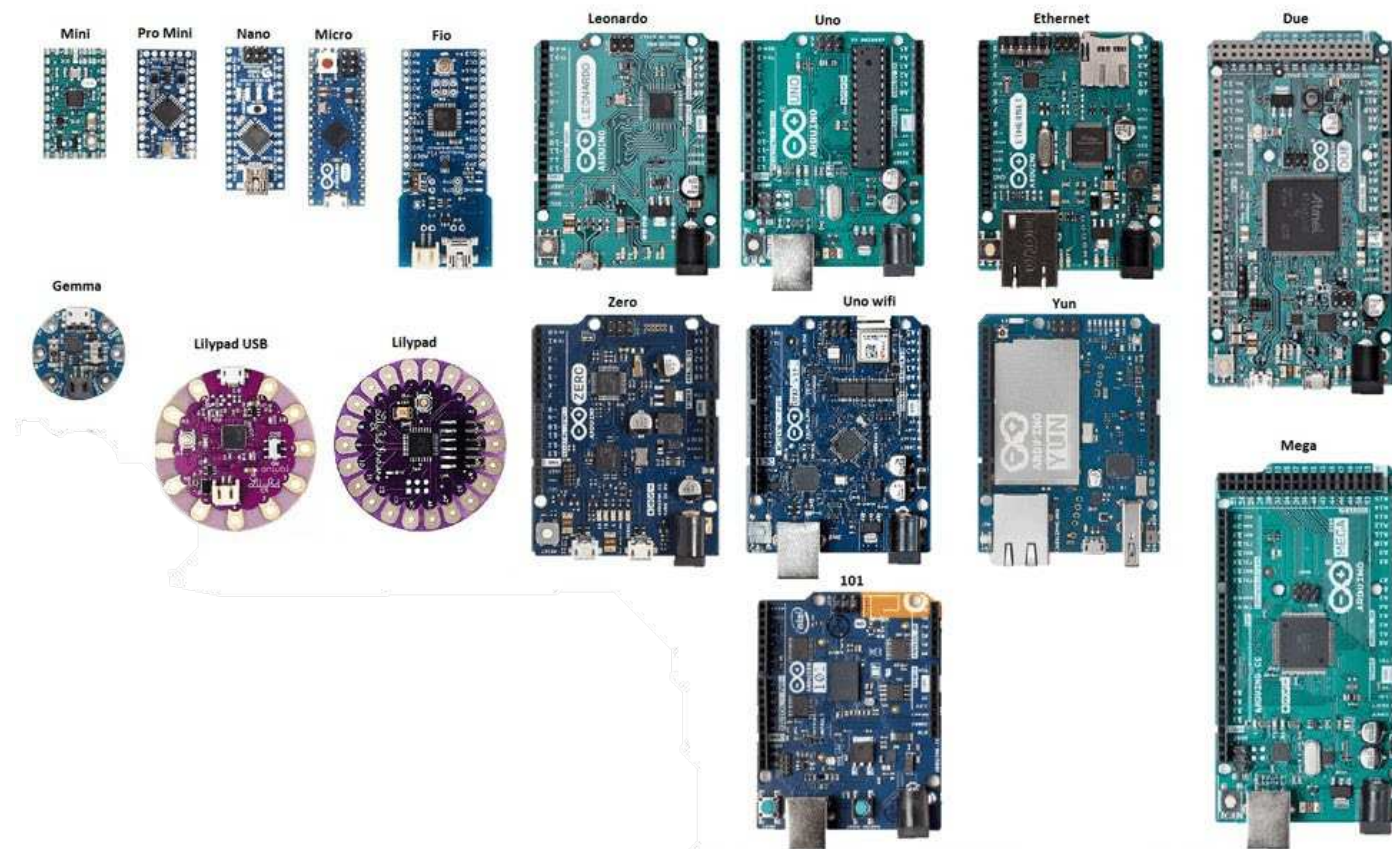
- Microcontroller: ATmega328P
 - Digital Pin IO: 14
 - Analog Pin I: 6
 - PWM Pin: 6
 - Working voltage: 5V
 - Max current on Digital Pin: 20 mA
 - Flash Memory: 32KB
 - SRAM Memory: 2KB
 - EEPROM Memory: 1KB
 - Clock: 16 MHz
 - Led built-in: D13
 - Native USB: no
 - ADC: 1x 10bit
 - DAC: no
 - Time/Counter: 2x 8bit, 1x 16bit
- ← Most common criteria



What if these IO pins are not enough?

Arduino Family and other microcontroller

Arduino Big Family



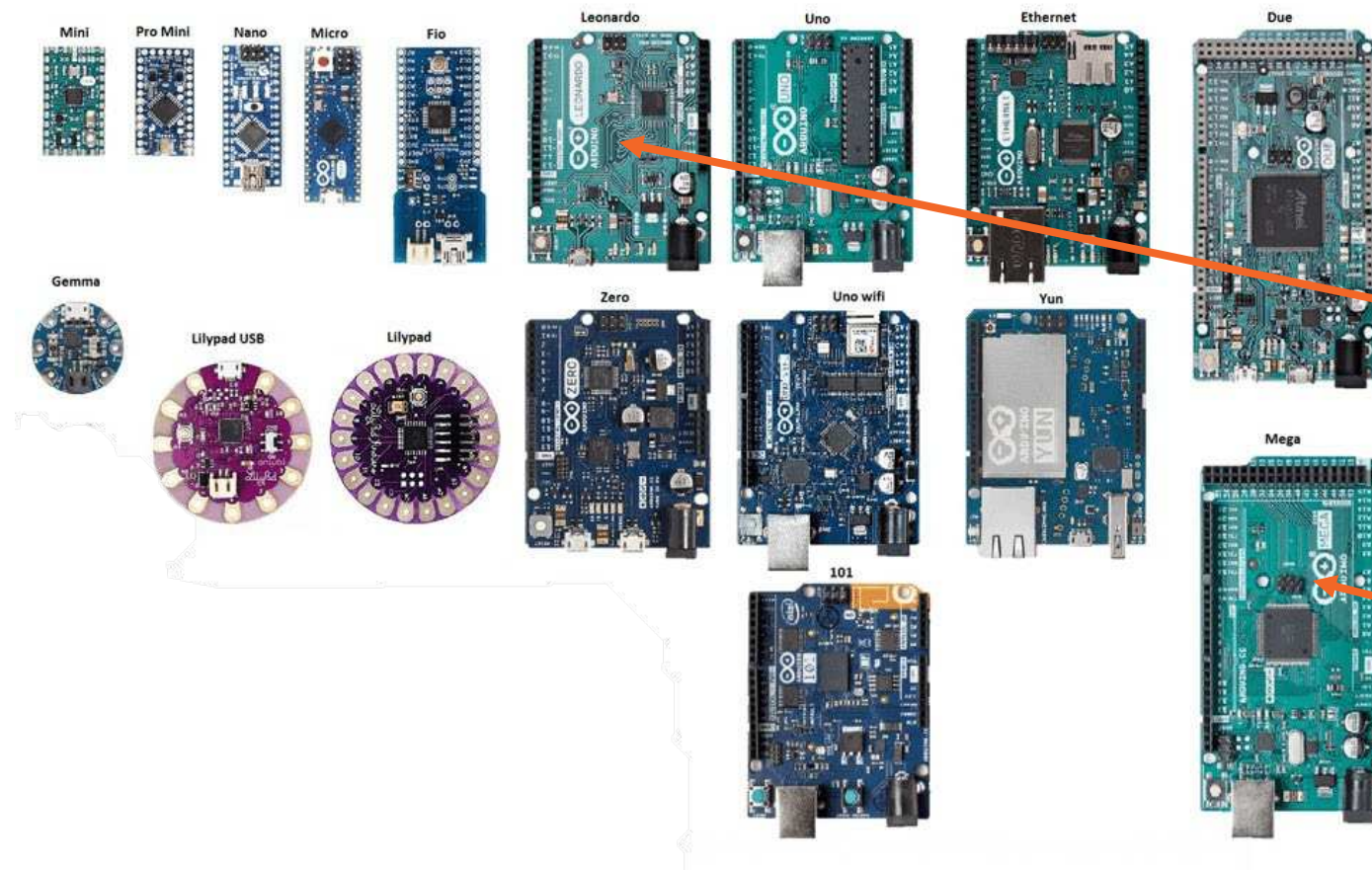
Uno
Leonardo
Mega
Fio
Nano
LilyPad
Yun
101
Nano 33 BLE
Nano Every
Micro
MKR1000
...and many others!

The list is very long

What are your project requirements?

Arduino Family and other microcontroller

Arduino Big Family



An example:
UNO has 13 digital pins

More pins?

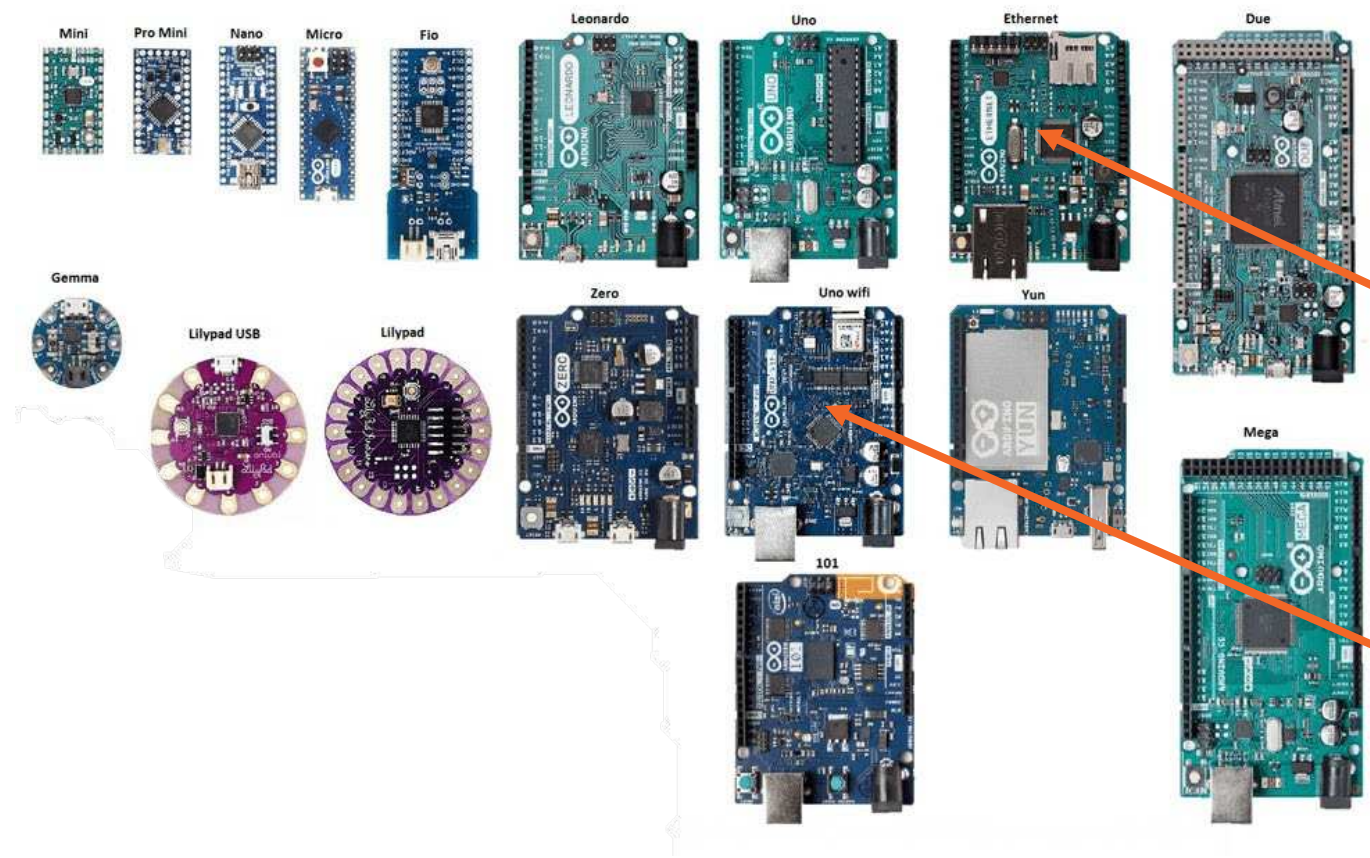
Leonardo has
20 digital IO pins

More Pins?

Mega has 54 Digital IO
pins!

Arduino Family and other microcontroller

Arduino Big Family



Another example:
UNO can be connected to
the PC only via USB (serial
port)

Other connection?

Ethernet!

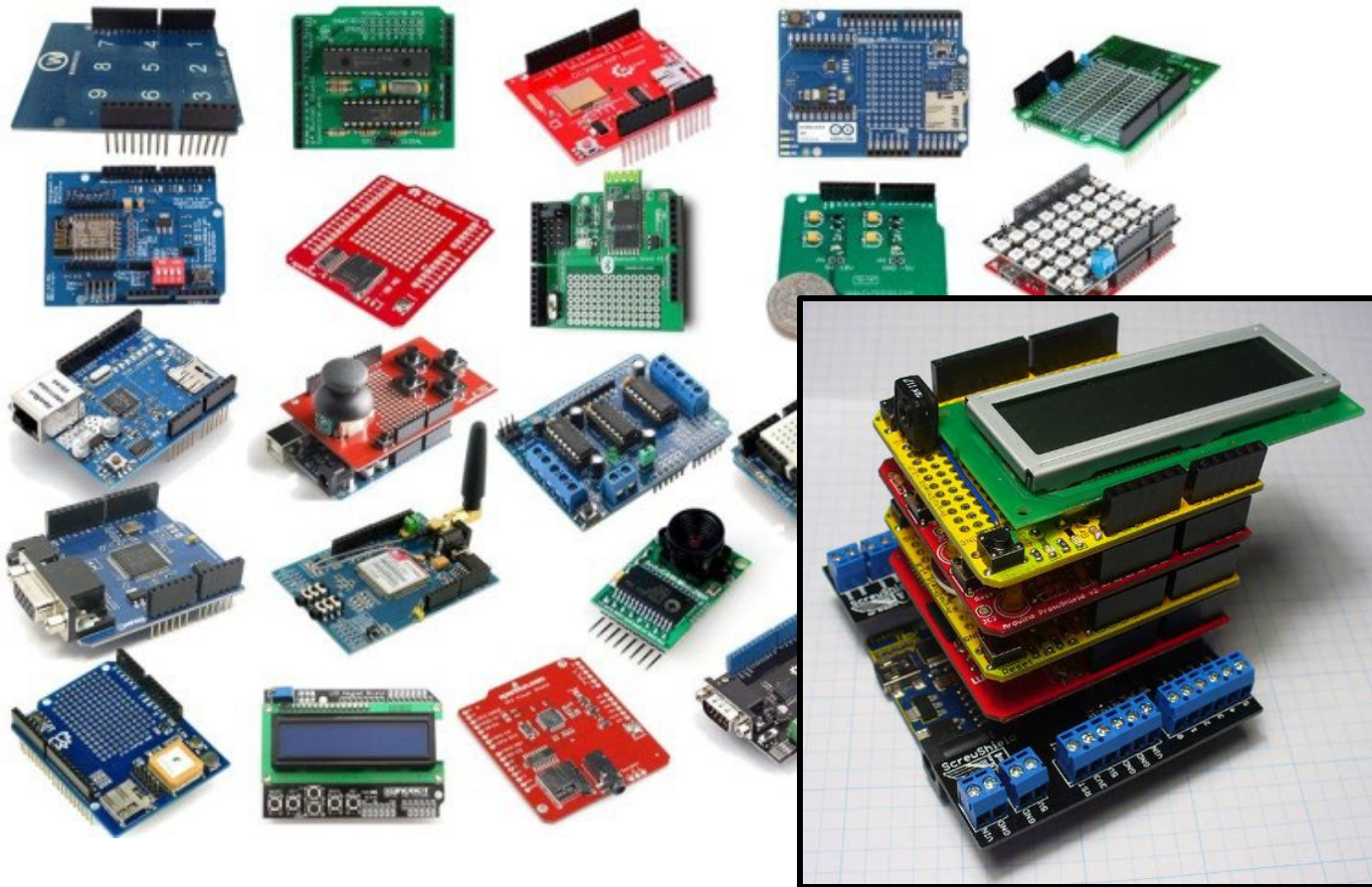
Cannot run the cable?

Uno WIFI!

Arduino Family and other microcontroller

More features? Arduino shields!

44



Try a shield!

Stackable electronic board that can be connected on the Arduino board!

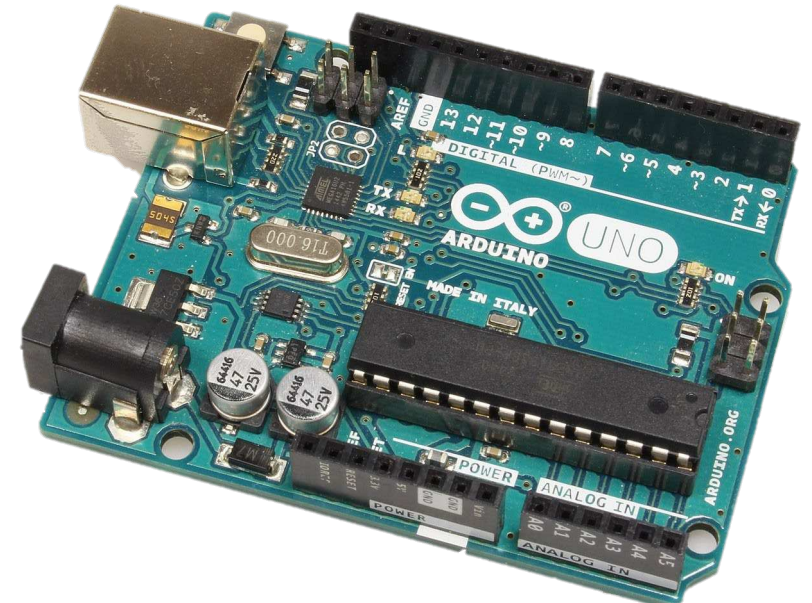
Different versions for different requirements:

- Screens
- Controllers
- Motor drivers
- Sensors
- Connections

Arduino Family and other microcontroller

This was the basic Arduino

The Arduino that I showed you are the basics...
They are our allies in our small project.
What if I need something more powerful?



Arduino Family and other microcontroller

Arduino MKR

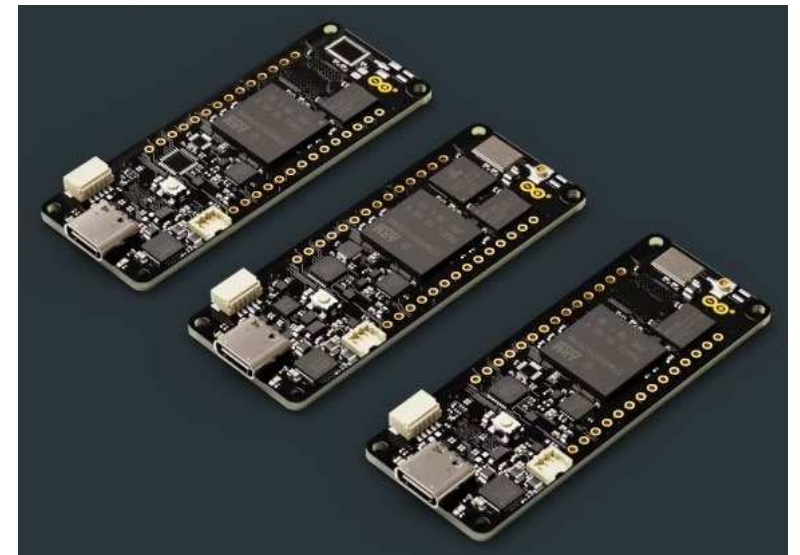
- Small Arduino board with more advanced micro than the other
- Designed for IOT
- Different sensors or receiver (GPS)
- They have some type of wireless connection

		
Arduino MKR 1000 WiFi	Arduino MKR WiFi 1010	Arduino MKR FOX 1200
		
Arduino MKR WAN 1300	Arduino MKR WAN 1310	Arduino MKR GSM 1400
		
Arduino MKR NB 1500	Arduino MKR Vidor 4000	Arduino MKR Zero

Arduino Family and other microcontroller

Arduino Portenta

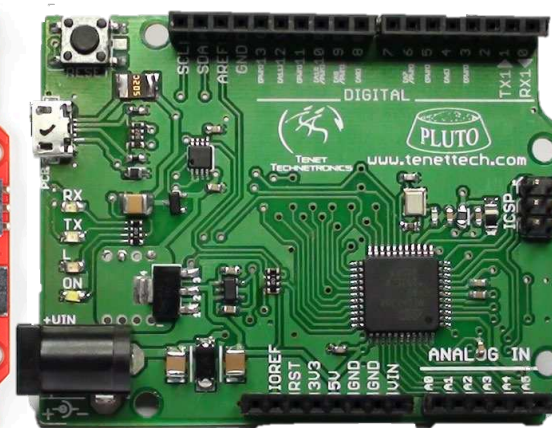
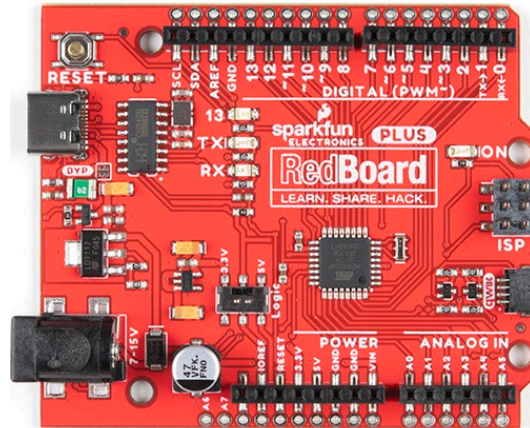
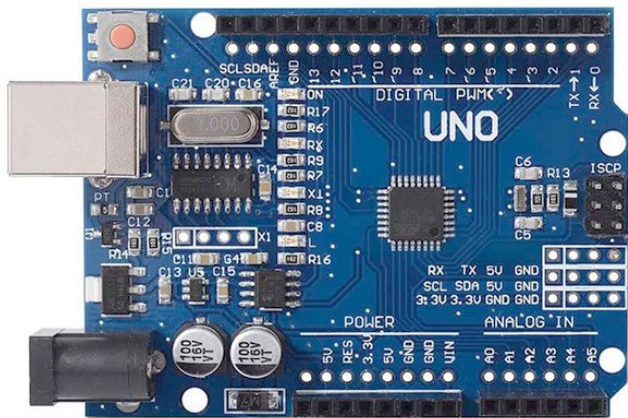
- Powerful SOC (system on chip)
- It is like a tiny computer (it also runs Linux)
- You can run AI stuff and other programs that require more resources
- Developed for fast industrial R&D



Arduino Family and other microcontroller

Arduino Clones

Arduino hardware is mostly opensource/openhardware,
It can be made by everyone!

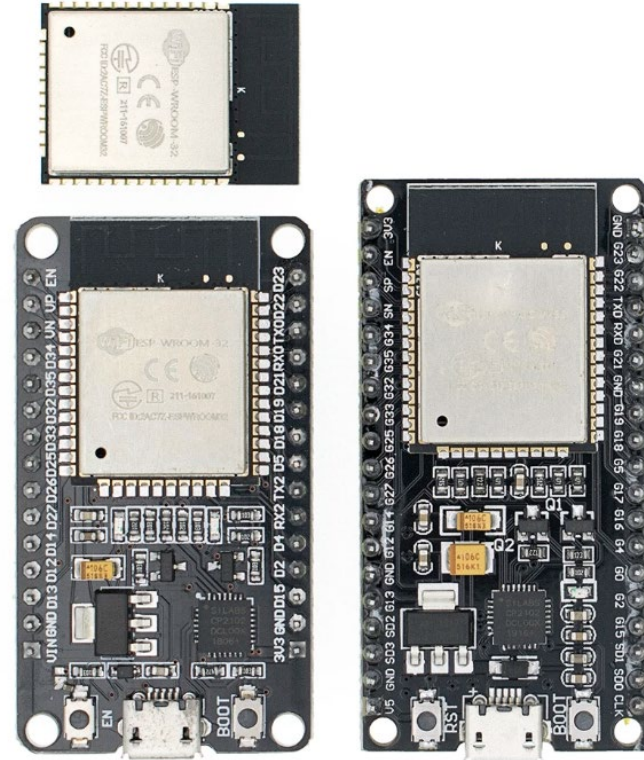


Arduino Family and other microcontroller

Arduino Competitors

Nowadays there are a lot of competitors with different spec.

The most common is the ESP32

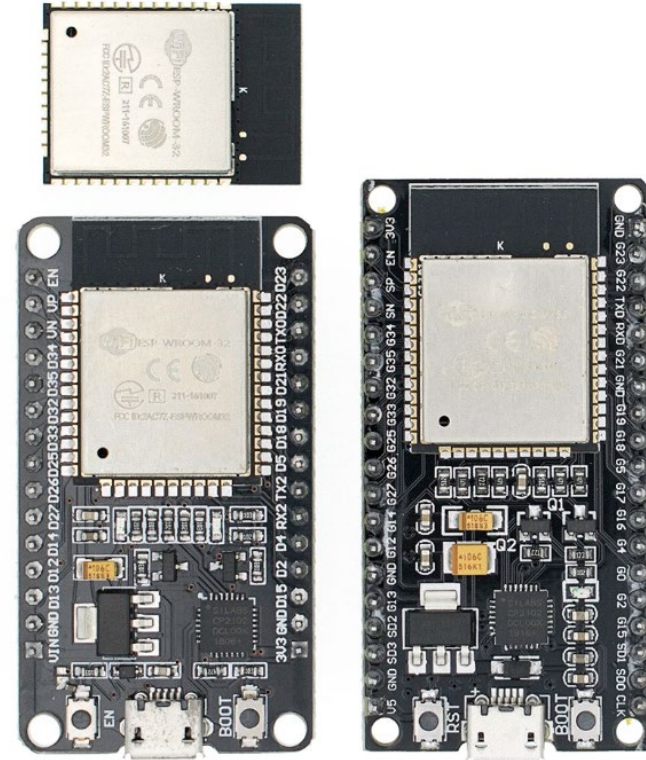


Arduino Family and other microcontroller ESP32

It is a SOC microcontroller very common, with a very large community and very cheap (10\$).

It has:

- X32 core (some version dual)
- WiFi
- Bluetooth
- Up to 34 GPIO

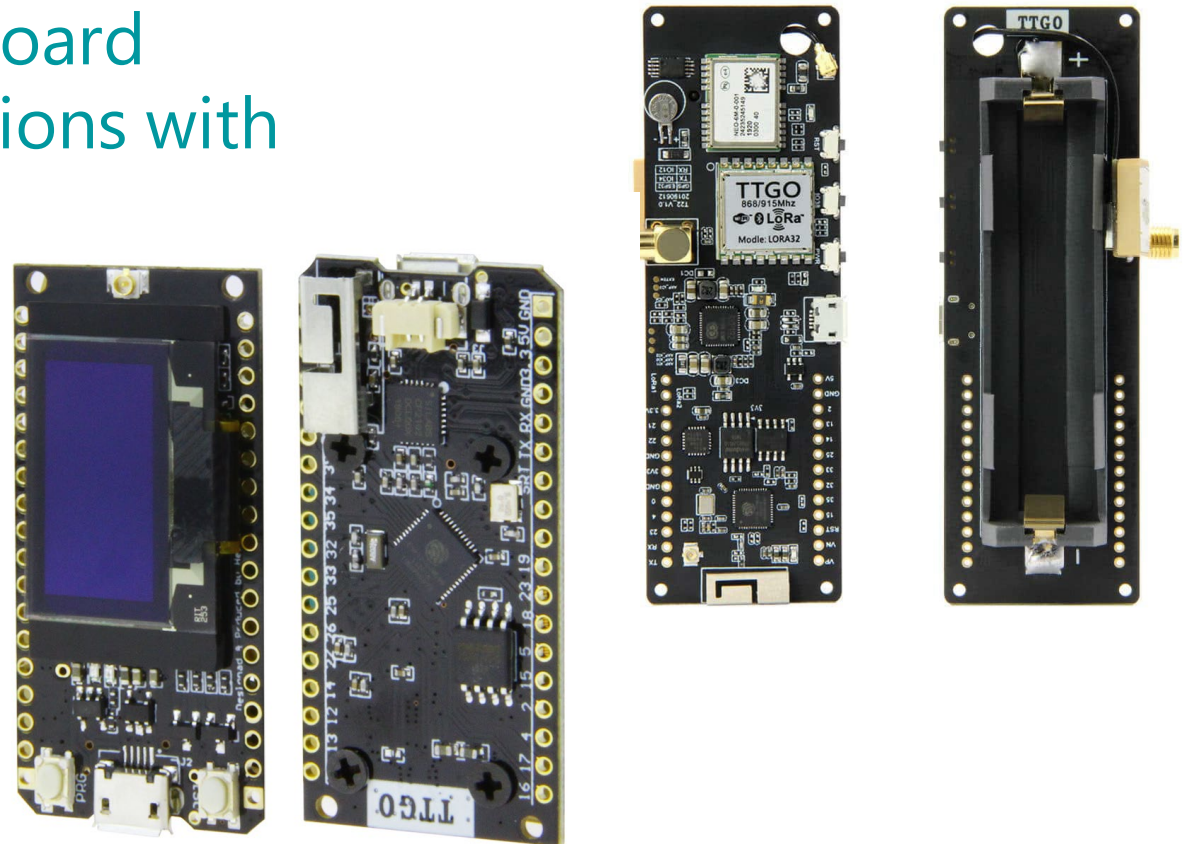


Arduino Family and other microcontroller

To be mentioned: TTGO

Based on ESP32, it is a series of board which comprehend different versions with different setup:

- Built in OLEO screen
- Power circuit
- LoRa chips
- GPS receiver
- SD card reader

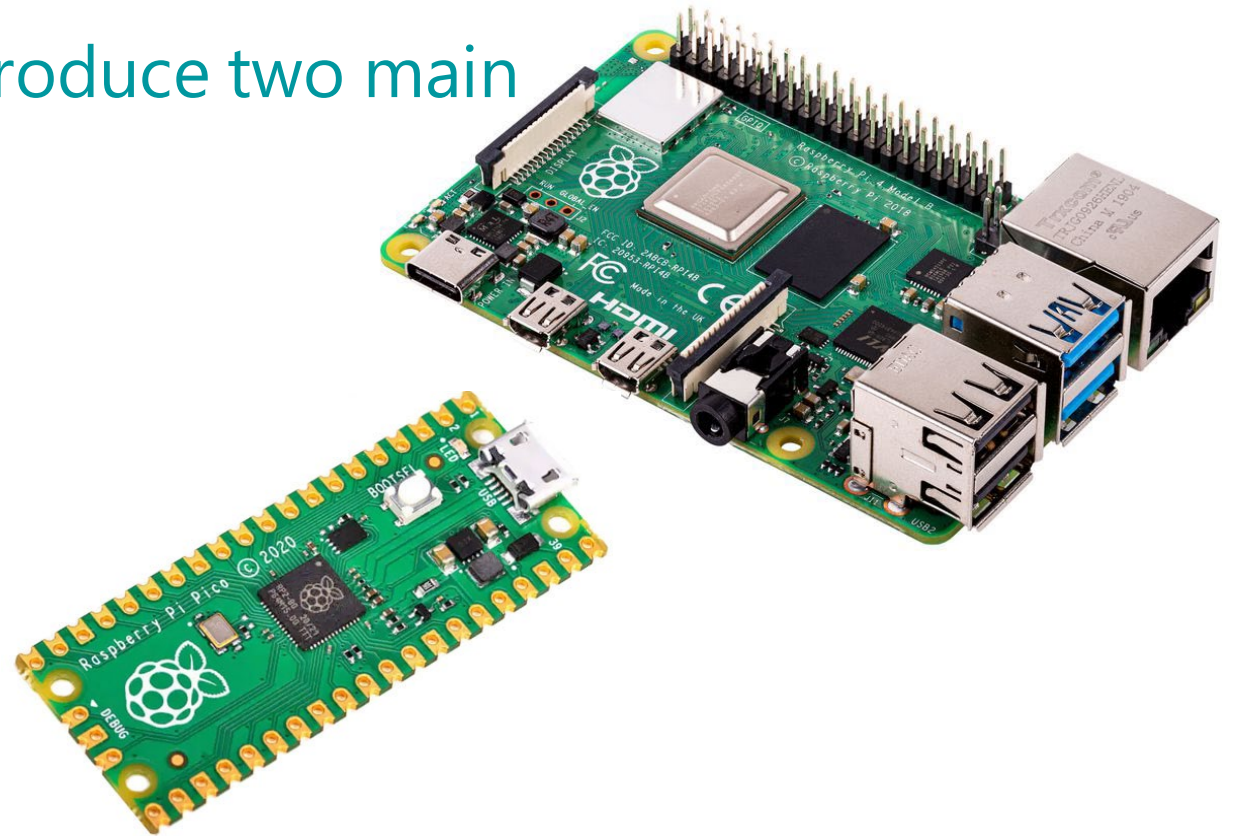


Arduino Family and other microcontroller I heard about Raspberry Pi

52

It is another known brand, they produce two main categories of of board:

- Raspberry Pi Pico – RP2040
- Raspberry Pi 3 – 4 – 5 - Zero



Arduino Family and other microcontroller

Raspberry Pi 3B-4-5-Zero

These are like real computer, they run Linux and are useful for an incredible number of applications, like:

- Low cost desktop computer
- Video/multimedia player
- Small server
-

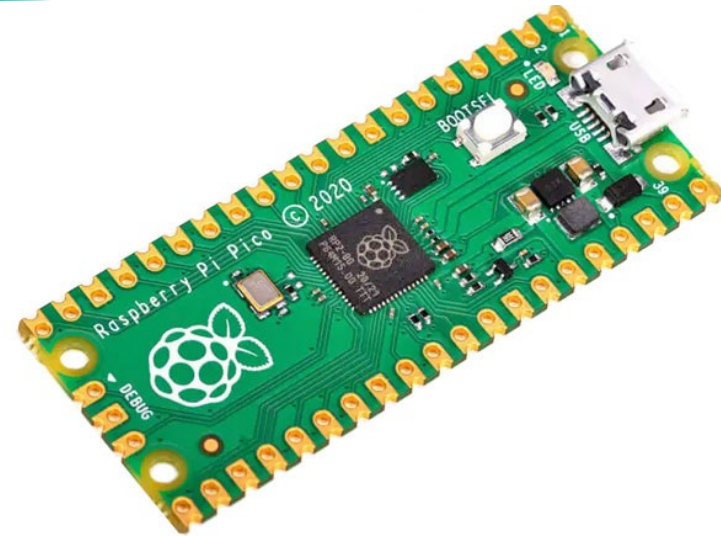
I use them to collect data for tracking planes=)



Arduino Family and other microcontroller

Raspberry Pi Pico

It is like an Arduino,
in some way similar to a MKR

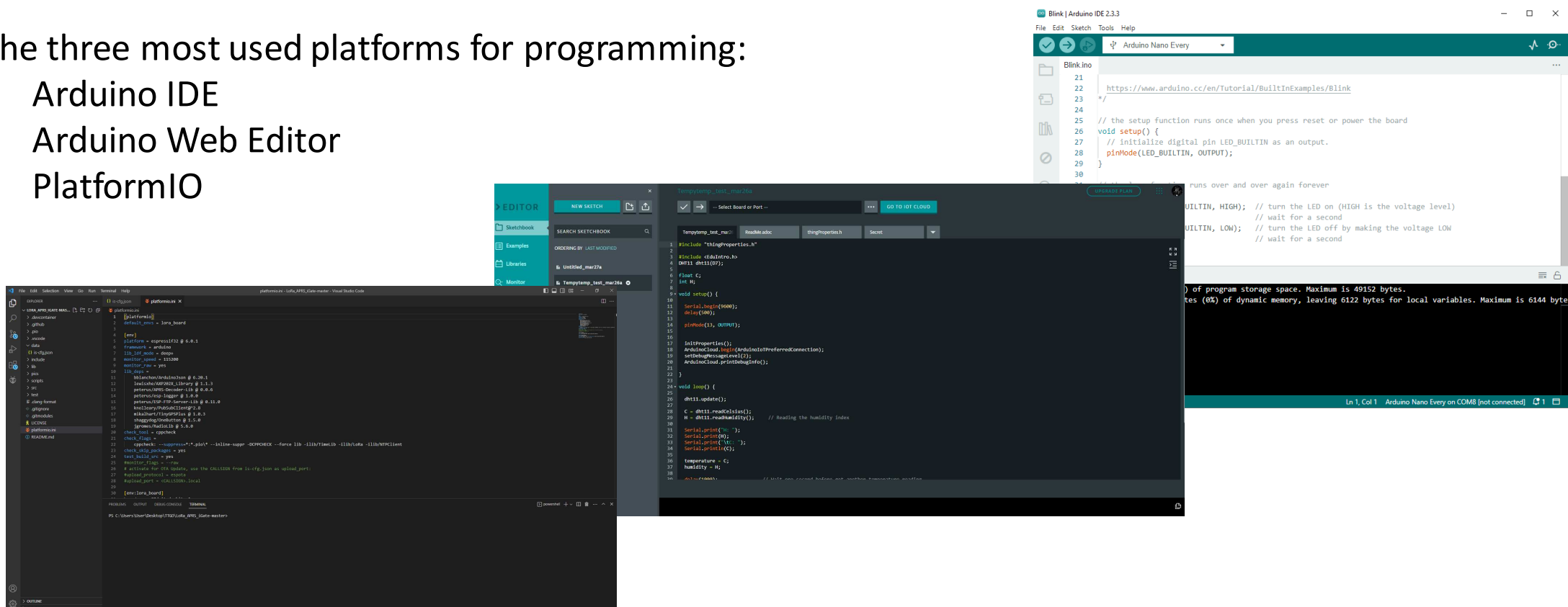


Arduino Family and other microcontroller

Other Arduino programming platforms

The three most used platforms for programming:

- Arduino IDE
- Arduino Web Editor
- PlatformIO



Arduino Family and other microcontroller

Micropython

There are different boards which support also micropython and the code is uploaded to the board via a simple file copy/paste. Then, it is interpreted by the firmware running on the microprocessor



NANO 33 BLE
SENSE



NANO 33 BLE



NANO RP2040
CONNECT



GIGA R1

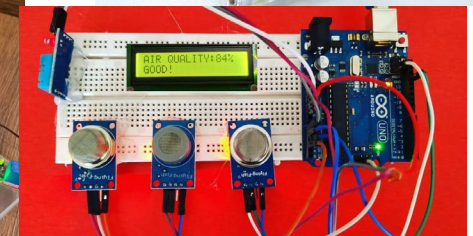
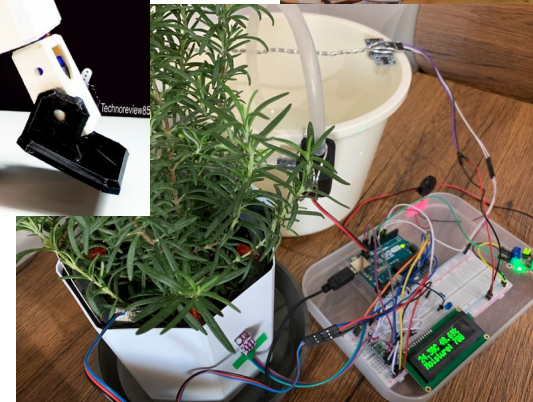
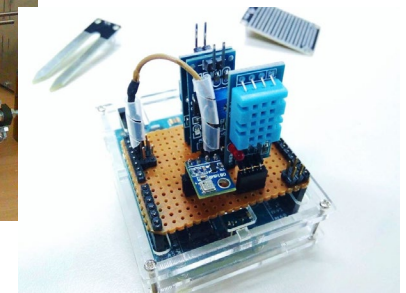
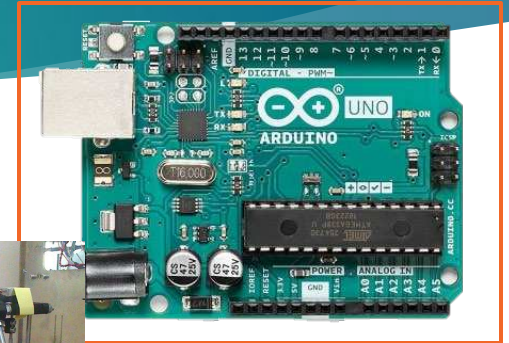
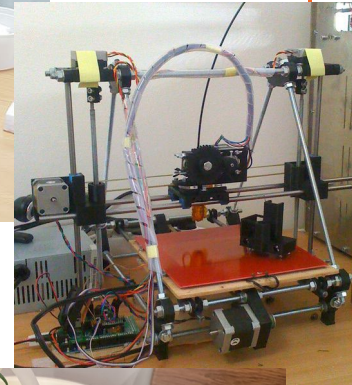
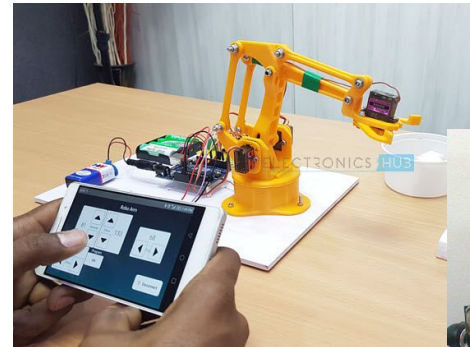


PORTENTA H7

ARDUINO PRO™

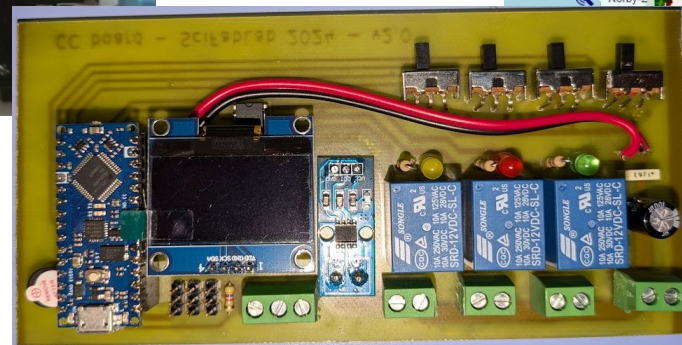
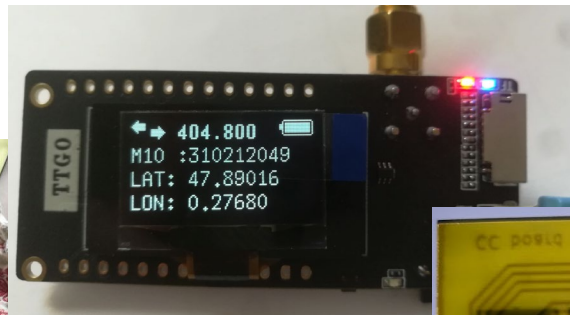
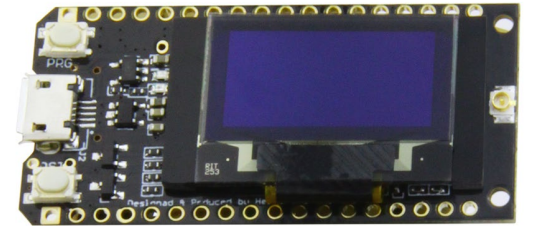
Some Arduino projects

- Robotic arm
- 3D printer
- Watering system
- Weather station
- Air pollution monitor
- Robots
-



Some other projects

- LoRa satellite receiver (tinygs.com)
- Weather balloons receiver (mysondy.altervista.org)
- Cloud Chamber =)



Some other side

59

Wokwi – Arduino code/project tester

The screenshot displays the Wokwi online IDE interface. On the left, the sketch editor shows the following code:

```
30 // Variables
31
32
33
34
35
36
37
38 void setup() {
39   Serial.begin(9600);
40
41   // OLED setup
42   if(!display.begin(i2c_address, true)) {
43     Serial.println("OLED initialization failed");
44     for(;;); // Don't proceed, loop forever
45   }
46
47   display.setTextSize(2); // Normal 11x14 pixel scale
48   display.setTextColor(SH110X_WHITE); // Draw white text
49   display.setCursor(0, 0); // Start at top-left corner
50   display.cp437(true); // Use full 256 char 'Code Page 437' font.
51
52   display.clearDisplay();
53   display.setCursor(0, 10);
54   display.write("Digital");
55   display.setCursor(25, 40);
56   display.write("Sensor");
57   display.display();
58
59 // temperature sensor initialization
60 sensors.begin();
61 Serial.println("Temperature sensor initialized");
62
63 delay(5000);
64
65 }
66
67 void loop() {
68
69   sensors.requestTemperatures();
70   temperature = sensors.getTempByIndex(0);
71   Serial.print("Temperature measured: ");
72   Serial.println(temperature);
73
74   display.clearDisplay();
75   display.setCursor(1, 20);
76   display.println("Te = " + String(temperature));
```

On the right, the simulation window shows a digital temperature sensor (DS18B20) connected to an Arduino Uno. The sensor is connected to the Arduino's VCC, GND, and data pins. The simulation includes a play button, a plus sign, and a minus sign.

Questions?

60

Thank you for your attention