Electronic and electromechanical prototyping

Introduction

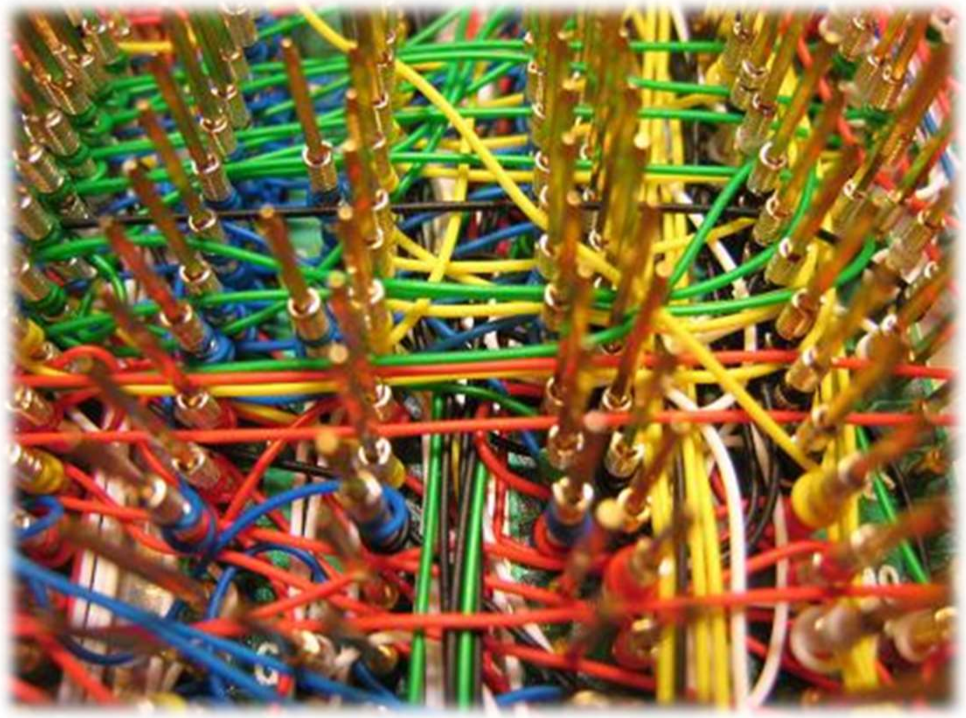
Corso LM ‘Materiali Intelligenti e Biomimetici’ – Prof. A. Ahluwalia
20/04/2017

ludovica.cacopardo@ing.unipi.it

Electronic and electromechanical prototyping

If you wanted to build a circuit prior to the 1960s, chances are you would have used a technique called **wire-wrap**.

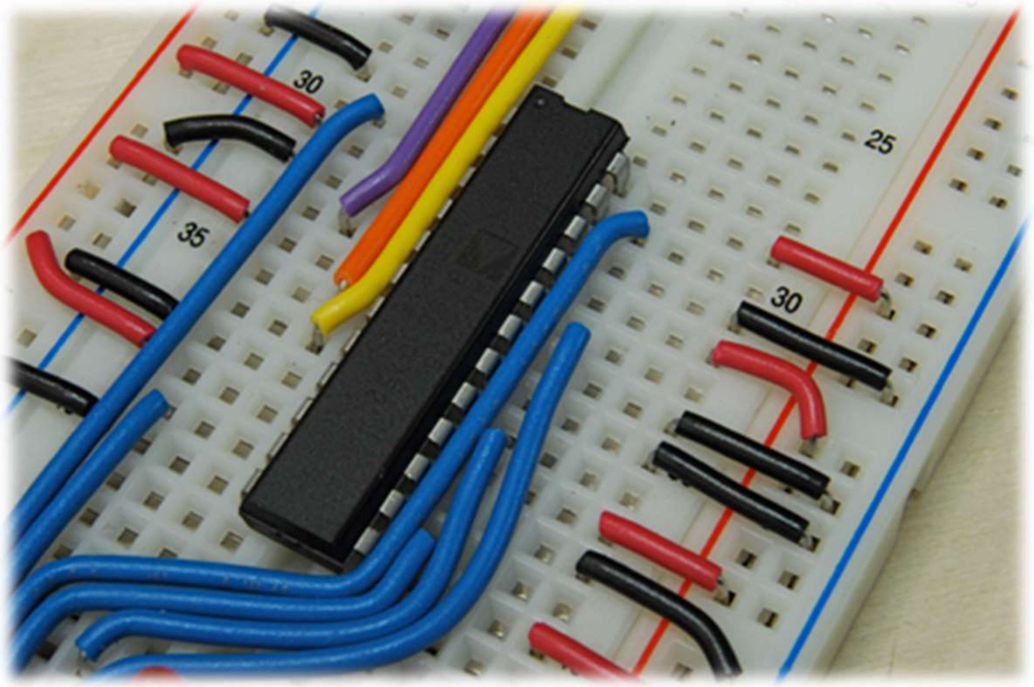
Wire wrap is a process that involves wrapping wires around conductive posts attached to a **perfboard**. As you can see, the process can get rather complex very quickly. Although this method is still used today, there is something that makes prototyping much easier, **breadboards**!



Breadboards

An electronics breadboard is actually referring to a **solderless breadboard**. These are great units for making temporary circuits and prototyping, and they require absolutely no soldering.

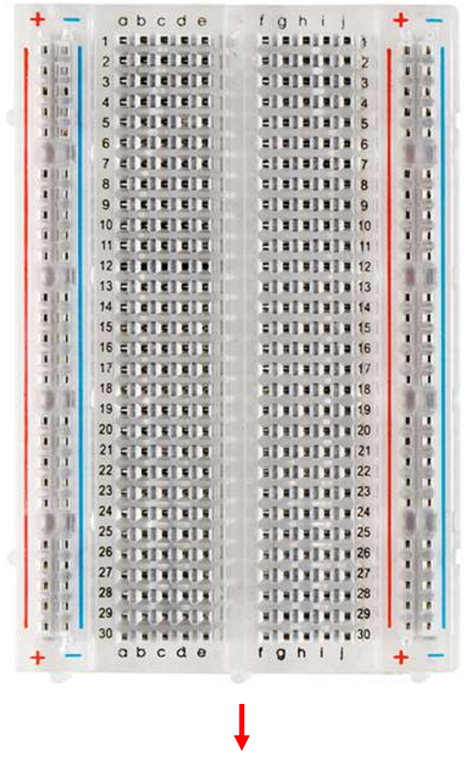
Another common use of breadboards is testing out new parts, such as Integrated circuits (ICs). When you are trying to figure out how a part works and constantly rewiring things, you don't want to have to solder your connections each time.

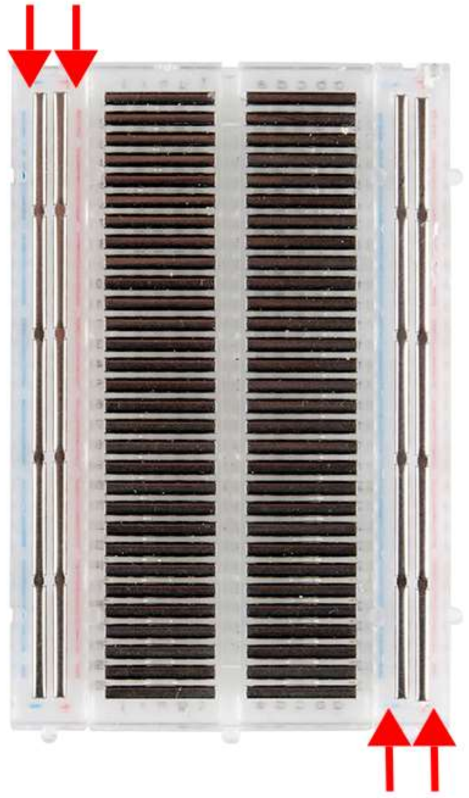


Breadboards 2

Terminal Strips are horizontal rows of metal strips on the bottom of the breadboard. *Once inserted that component will be **electrically connected** to anything else placed in that row*. This is because the metal rows are conductive and allow current to flow from any point in that strip.

Power Rails are metal strips that run vertically along the sides. When building a circuit, you tend to need power in lots of different places. The power rails give you lots of ***easy access to power*** wherever you need it in your circuit. Usually they will be labeled with a ‘+’ and a ‘-’, to indicate the positive and negative side. It is important to be aware that the *power rails on either side are not connected*, so if you want the same power source on both sides, you will need to connect the two sides with some jumper wires.





Breadboards 3

DIP Support: Many integrated circuits (ICs) or, simply, chips, are manufactured specifically to fit onto breadboards. In order to minimize the amount of space they take up on the breadboard, they come in what is known as a Dual in-line Package (DIP). These DIP chips have legs that come out of both sides and fit perfectly over the ravine that isolates the two sides of a breadboard. Since each leg on the IC is unique, we don't want both sides to be connected to each other.

Providing Power to a Breadboard:

- Borrowing from Other Power Sources: If you are working with a development board such as an Arduino, then you can simply pull power from the Arduino's female headers. The Arduino has multiple power and ground pins that you can connect to the power rails or other rows on a breadboard.

- Benchtop power supplies that allow you to provide a wide range of voltage and current to your circuit. Using a banana connector you can provide power from the supply to the binding posts.

Components: Resistances

The principal job of a resistor within an electrical or electronic circuit is to “resist” or **regulate the flow of electrons (current)** through them by using the type of conductive material from which they are composed.

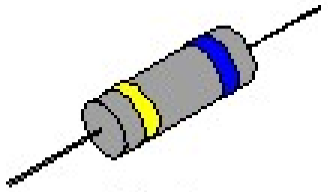
Resistors can also be connected together in various **series and parallel combinations** to form resistor networks which can act as voltage droppers, voltage dividers or current limiters within a circuit.

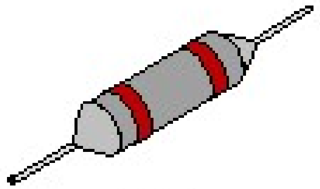
Resistors are what are called “**Passive Devices**”, that is they contain no source of power or amplification but only attenuate or reduce the voltage or current signal passing through them. This *attenuation results in electrical energy being lost in the form of hea*t as the resistor resists the flow of electrons through it.

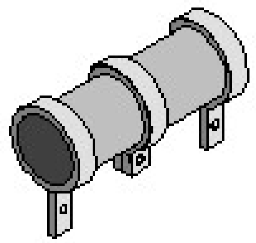
Components (2): Types of Resistance

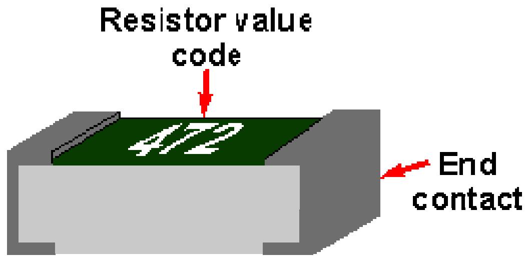
All modern fixed value resistors can be classified into four broad groups:

- Carbon Composition Resistor – Made of *carbon dust or graphite paste*, low wattage values
- Film or Cermet Resistor – Made from *conductive metal oxide paste*, very low wattage values
- Wire-wound Resistor – *Metallic bodies* for heatsink mounting, very high wattage ratings
- Semiconductor Resistor – High frequency/precision *surface mount thin film technology*


Carbon


Film


Adjustable wirewound



Components (3): Capacitors

Basically a capacitor is formed *from two conducting plates separated by a thin insulating layer*. They are manufactured in many forms, styles, and from many materials. Capacitors are widely used in electrical and electronic circuits.

In electronic circuits, small value capacitors are used to couple signals between stages of amplifiers, as components of electric filters and tuned circuits, as parts of power supply systems to smooth rectified current.

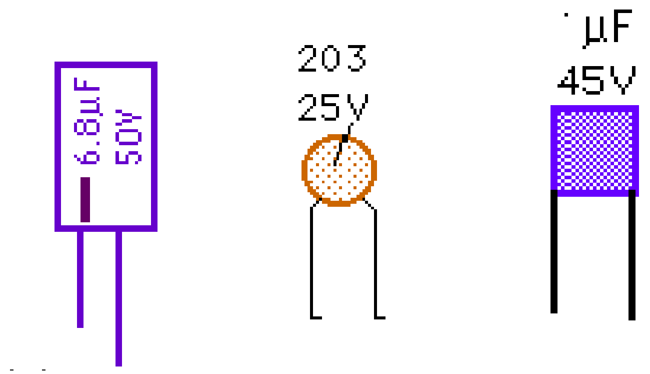
In electrical circuits, larger value capacitors are used for energy storage in such applications as strobe lights, as parts of some types of electric motors, for power factor correction in AC power distribution systems

Components (4): Types of Capacitors

a) Electrolytic:
Electrolytic capacitors are a type of capacitor that is *polarised*. They are able to offer *high capacitance values* - typically above 1µF, and are most widely used for low frequency applications (frequency limit if around 100 kHz) - power supplies, decoupling and audio coupling applications.

b) Ceramic capacitors:
Values range from a few picofarads to around 0.1 microfarads. Ceramic capacitor types are by far the most commonly used type of capacitor being cheap and reliable and their loss factor is particularly low although this is dependent on the exact dielectric in use.

c) Polymer capacitors: Polystyrene, Polyester, Metallised Polyester, Polycarbonate, Polypropylene.

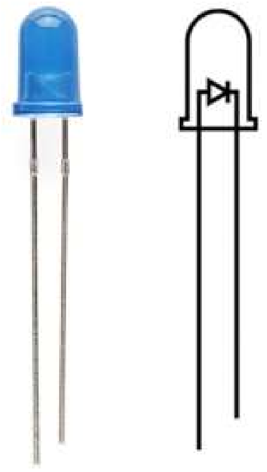


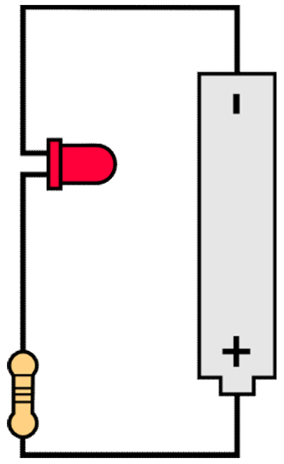
Components (5): LEDs

- LEDs (Light Emitting Diodes), being diodes, will only allow current to flow in one direction. The positive side of the LED is called the "anode" and is marked by having a longer leg. The other, negative side of the LED is called the "cathode." Current flows from the anode to the cathode and never the opposite direction.

- More Current, More Light: The brightness of an LED is directly dependent on how much current it draws. This means that you can control the brightness of a LED by controlling the amount of current through it.

- If you connect an LED directly to a current source it will try to dissipate as much power as it's allowed to draw and it will destroy itself. That's why it's important to limit the amount of current flowing across the LED with resistors. Resistors limit the flow of electrons in the circuit and protect the LED from trying to draw too much current.





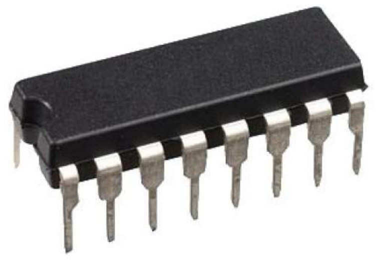
Critical Components Selection

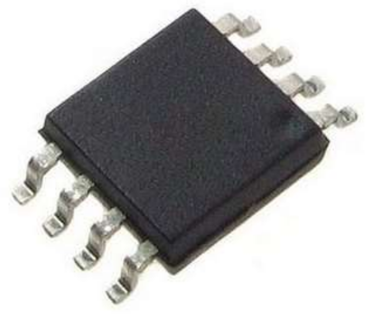
The first step of designing the electronics is to select the various microchips* (i.e. integrated circuits), sensors, displays, connectors, and other electronic devices needed based upon the desired functions and target retail price of your product.

Digikey, RS, and Mouser are the most popular suppliers of electronic components. You can purchase most electronic components in ones (for prototyping and initial testing) or up to thousands (for low-volume manufacturing)

*Chip Packaging

- DIP (dual in-line package): The package may be through-hole mounted to a PCB or inserted in a socket
- SMD (surface mount device): the components are mounted or placed directly onto the surface of PCBs





Arduino

Arduino is an *open-source electronics platform* based on *easy-to-use hardware and software*. Arduino boards are able to read inputs - light on a sensor, a finger on a button- and turn it into an output - activating a motor, turning on an LED. You can tell your board what to do by sending a set of instructions to the **microcontroller on the board**.

To do so you use the Arduino programming language (based on C/C++), and the Arduino Software (IDE).

Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.

https://www.arduino.cc/en/Main/Software

μC and μP

Microprocessor is an IC which has **only the CPU** (central processing unit) inside them i.e. only the processing powers. These microprocessors **don’t have RAM, ROM and other peripheral on the chip**. A system designer has to add them externally to make them functional. Application of microprocessor includes Desktop PC’s, Laptops, notepads etc.

Microprocessor find applications where **tasks are unspecific** like developing software, games, websites, photo editing, creating documents etc. In such cases the *relationship between input and output is not defined*. They need *high amount of resources* like RAM, ROM, I/O ports etc. The clock speed of the Microprocessor is quite high as compared to the microcontroller. Whereas the microcontrollers operate from a few MHz to 30 to 50 MHz, today’s microprocessor operate above 1GHz as they perform complex tasks.

Microcontroller **has a CPU, in addition with a fixed amount of RAM, ROM and other peripherals** all embedded on a single chip. At times it is also termed as a mini computer or a computer on a single chip. Today different manufacturers produce microcontrollers with a wide range of features available.

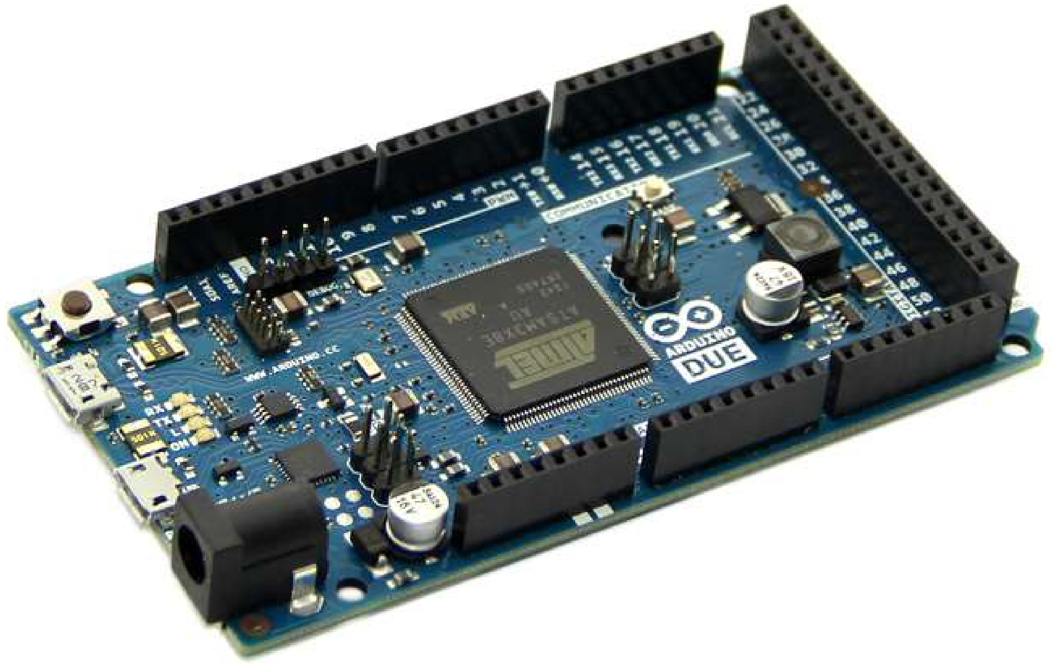
Microcontrollers are designed to perform **specific tasks**. Specific means applications where the *relationship of input and output is defined*. Depending on the input, some processing needs to be done and output is delivered. For example, keyboards, mouse, washing machine, digicam, pendrive, remote, microwave, cars, bikes, telephone, mobiles, watches, etc. Since the applications are very specific, they need *small resources* like RAM, ROM, I/O ports etc and hence can be embedded on a single chip. This in turn *reduces the size and the cost*.

Arduino DUE board

The Arduino Due is a microcontroller board based on the *Atmel SAM3X8E ARM Cortex-M3 CPU*.

It is the first Arduino board based on a **32-bit ARM core microcontroller**.

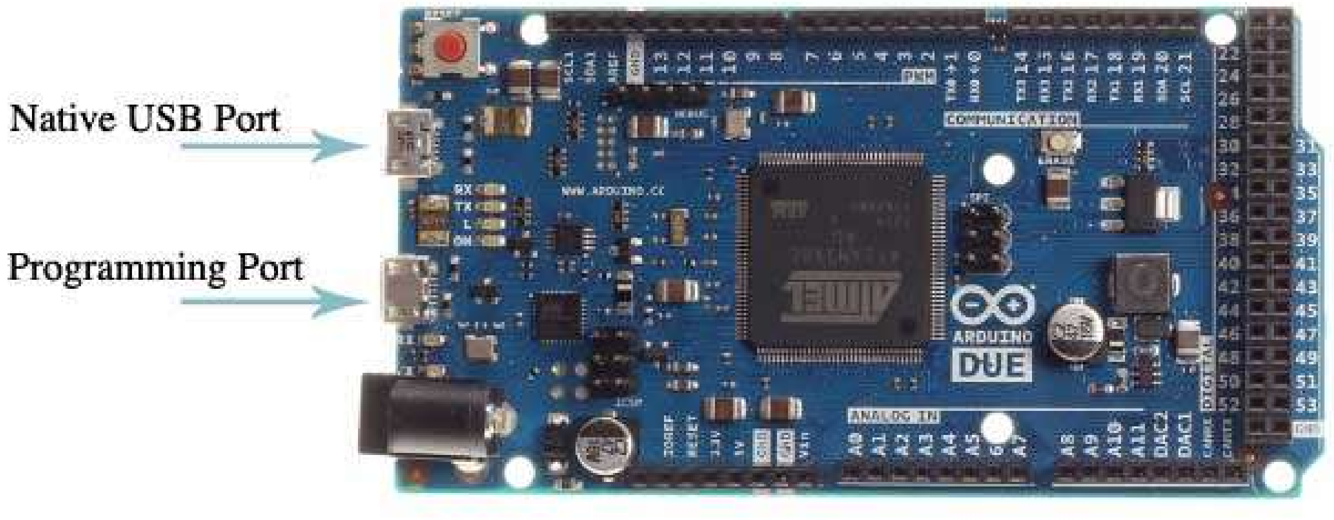
It has *54 digital input/output pins* (of which 12 can be used as PWM outputs), *12 analog inputs*, *4 UARTs* (hardware serial ports), a *84 MHz clock*, an *USB OTG capable connection*, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.



https://www.arduino.cc/en/Main/arduinoBoardDue

Serial Communication – USB

USB (Universal Serial Bus) is an industry standard initially that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices.



Serial Communication – USB (2)

- **Programming port**: The programming port is connected to an ATmega16U2, which provides a virtual COM port to software on a connected computer. It uses the 16U2 as a USB-to-serial chip connected to the first UART of the SAM3X (RX0 and TX0).
  The 16U2 has two pins connected to the Reset and Erase pins of the SAM3X. Opening and closing the Programming port connected at 1200bps triggers a *“hard erase” procedure of the SAM3X chip*, activating the Erase and Reset pins on the SAM3X before communicating with the UART. This is the recommended port for programming the Due.

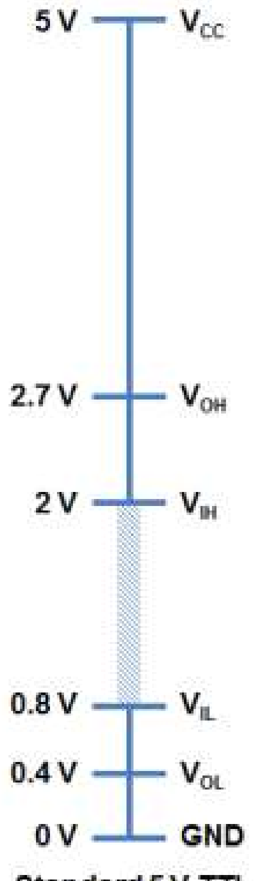
- **Native port**: The Native USB port is connected directly to the SAM3X. Opening and closing the Native port at 1200bps triggers a '*soft erase*' procedure: the flash memory is erased and the board is restarted with the bootloader. Opening and closing the native port at a different baud rate will not reset the SAM3X.

Serial Communication - TTL

Serial is used for communication between the Arduino board and a computer or other devices. Serial communication on **pins TX/RX** (just two wires - one for sending data and another for receiving) uses ***TTL* (Transistor-Transistor Logic)** *logic levels* (5V or 3.3V depending on the board).

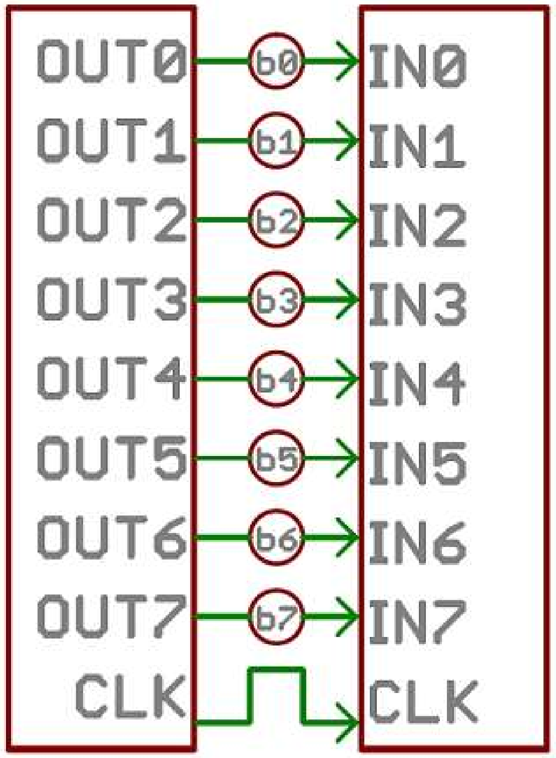
For any logic family, there are a number **of threshold voltage levels** to know:
V_{OH} – Minimum OUTPUT Voltage level a TTL device will provide for a HIGH signal.
V_{IH} – Minimum INPUT Voltage level to be considered a HIGH.
V_{OL} – Maximum OUTPUT Voltage level a device will provide for a LOW signal.
V_{IL} – Maximum INPUT Voltage level to still be considered a LOW.

All Arduino boards have at least one **serial port** (a UART).
The **universal asynchronous receiver/transmitter (UART)** is a *computer hardware device* that takes bytes of data and *transmits the individual bits in a sequential fashion*. At the destination, a second UART re-assembles the bits into complete bytes.
Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms.



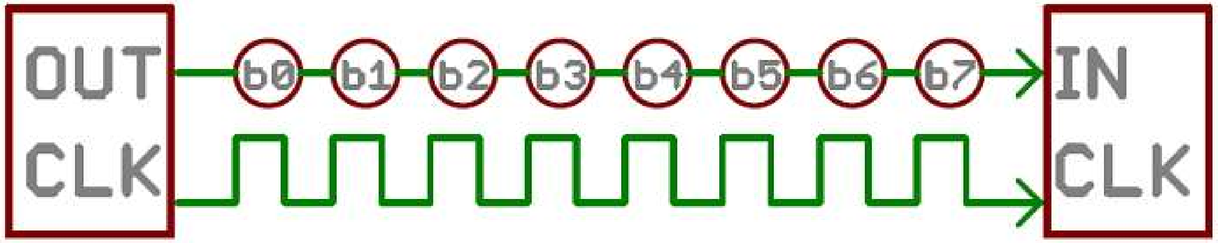
Standard 5 V TTL

Serial Communication – TTL (2)



1. ***Serial transmission*** *of digital information (bits) through a single wire or other medium is less costly than* ***parallel transmission*** *through multiple wires.*

2. A **synchronous** serial interface always pairs its data line(s) with a clock signal, so all devices on a synchronous serial bus share a common clock. This makes for a more straightforward, often faster serial transfer, but it also requires at least one extra wire between communicating devices.
   Asynchronous means that data is *transferred without support from an external clock* signal. This transmission method is perfect for minimizing the required wires and I/O pins, but it does mean we need to put some extra effort into reliably transferring and receiving data



Serial Communication - TTL (3)

Arduino communicates on **digital pins 0 (RX) and 1 (TX)**. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

The Arduino Due has three additional 3.3V TTL serial ports:

Serial on pins 0 (RX) and 1 (TX);
Serial1 on pins 19 (RX) and 18 (TX);
Serial2 on pins 17 (RX) and 16 (TX);
Serial3 on pins 15 (RX) and 14 (TX).

Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

Serial Communication – SPI and I2C

- Pins that support SPI (Serial Peripheral Interface) communication using the SPI library

  It has only three lines (i.e. MISO, MOSI and SCK) for data transmission. In SPI communication there is only one MASTER controller and one SLAVE controller, and hence the slave addressing is not required. It is a full duplex serial data communication process.

- Pins for TWI (Two wire interface)/ I2C (I-two-C) protocol -> TWI 1: 20 (SDA) and 21 (SCL); TWI 2: SDA1 and SCL1. These pins support TWI communication using the Wire library.

  In TWI the serial data transmission is done in asynchronous mode. This protocol uses only two wires for communicating between two or more ICs. The two bidirectional open drain lines named SDA (Serial Data) and SCL (Serial Clock)

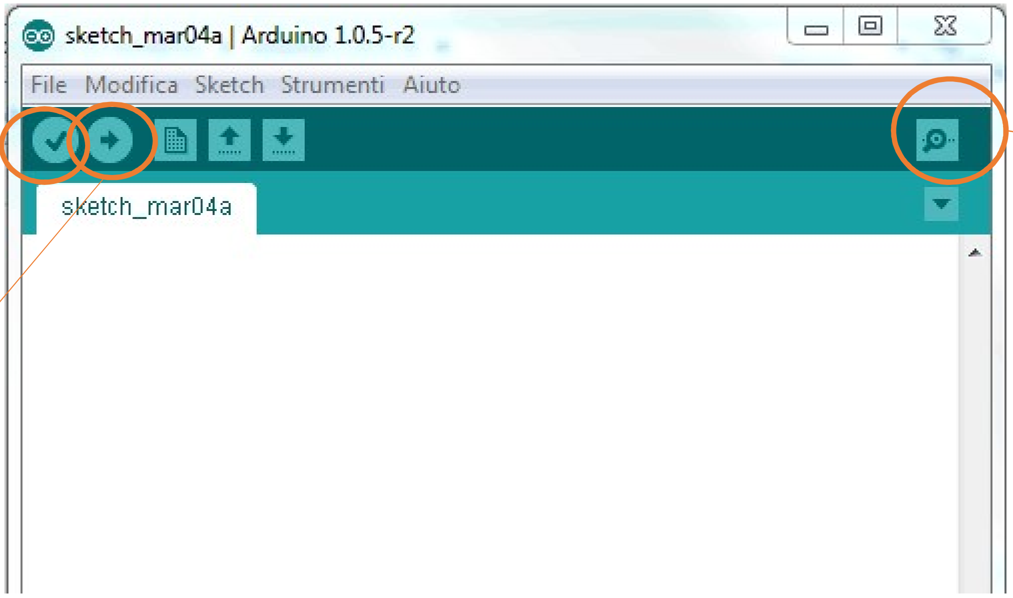
Arduino IDE

The IDE (**Integrated development environment**) allows us to write, compile and transfer our programs on the arduino board.

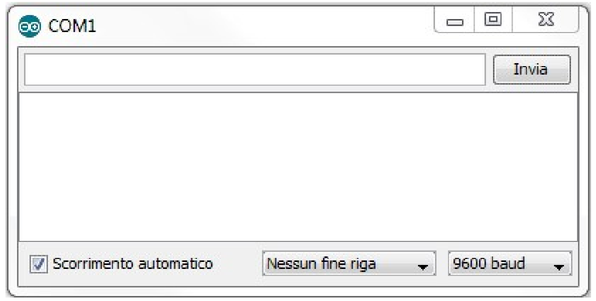
In primo luogo, bisogna sottolineare che il linguaggio di programmazione utilizzato per Arduino è il C/C++; si può parlare di entrambi i linguaggi, in quanto è possibile utilizzare o meno il supporto per le classi che il C++ mette a disposizione

Verify: performs verification of the written code and its compilation;

Load : it loads the compiled firmware on board;



Serial Opening



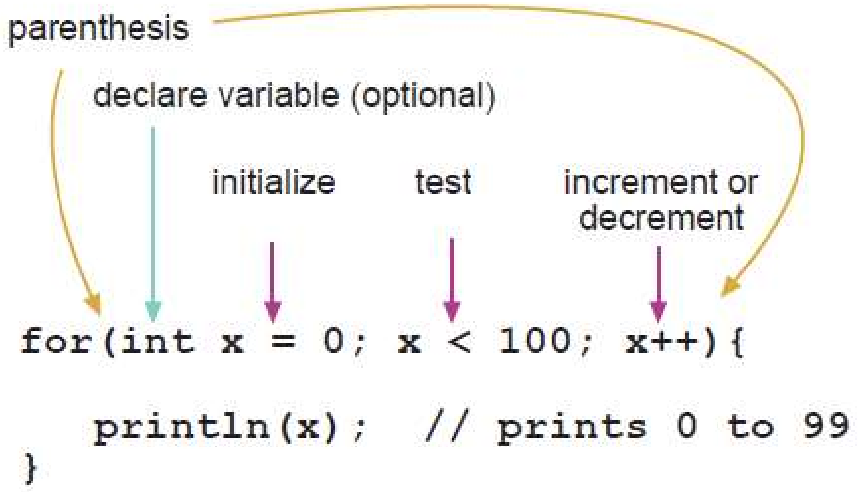
For Loop

The for statement is used to **repeat a block of statements** enclosed in curly braces.
An increment counter is usually used to increment and terminate the loop.

There are *three parts to the for loop header*:

for (**initialization**; **condition**; **increment**) {
//statement(s);
}

- The initialization happens first and exactly once.
- Each time through the loop, the condition is tested; if it's true, the statement block and the increment is executed.
- Then the condition is tested again. When the condition becomes false, the loop ends.



```
for(int x = 0; x < 100; x++){
   println(x);  // prints 0 to 99
}
```

While Loop

while loops will loop continuously, until the expression inside the parenthesis becomes false.
This could be in your code, such as an incremented variable or an external condition, such as testing a sensor.

```
while(expression){
  // statement(s)
}
```

Example:

```
var = 0;
while(var < 200){
  // do something repetitive 200 times
  var++;
}
```

If/else Instruction

if/else allows greater control over the flow of code than the basic if statement, by allowing **multiple tests** to be grouped together. Each test will proceed to the next one until a true test is encountered. *When a true test is found, its associated block of code is run*, and the program then skips to the line following the entire if/else construction. If no test proves to be true, the default else block is executed, if one is present, and sets the default behavior.

```
If (pinFiveInput < 500)
{  // action A
}
else if (pinFiveInput >= 1000)
{  // action B
}
else
{  // do Thing C
}
```

Switch

switch...case controls the flow of programs by allowing programmers **to specify different code that should be executed in various conditions**. In particular, a switch *statement compares the value of a variable to the values specified in case statements*. When a case statement is found whose value matches that of the variable, the code in that case statement is run.

The break keyword exits the switch statement, and is typically used at the end of each case. Without a break statement, the switch statement will continue executing the following expressions ("falling-through") until a break, or the end of the switch statement is reached.

```
switch (var) {
  case 1:
    //do something when var equals 1
    break;
  case 2:
    //do something when var equals 2
    break;
  default:
    // if nothing else matches, do the default
    // default is optional
  break;
 }
```

References

- http://predictabledesigns.com/how-to-develop-and-prototype-a-new-product/
- https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard
- http://www.electronics-tutorials.ws/resistor/res_1.html
- http://www.studyelectrical.com/2016/12/different-types-classification-of-capacitors.html
- https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds
- http://www.madehow.com/Volume-2/Printed-Circuit-Board.html
- https://www.arduino.cc/en/Guide/Introduction
- https://www.engineersgarage.com/tutorials/twi-i2c-interface