Arduino @ NetStudent

Federico Vanzati
f.vanzati@arduino.cc

Officine Arduino Torino

23/04/2013

Arduino Uno

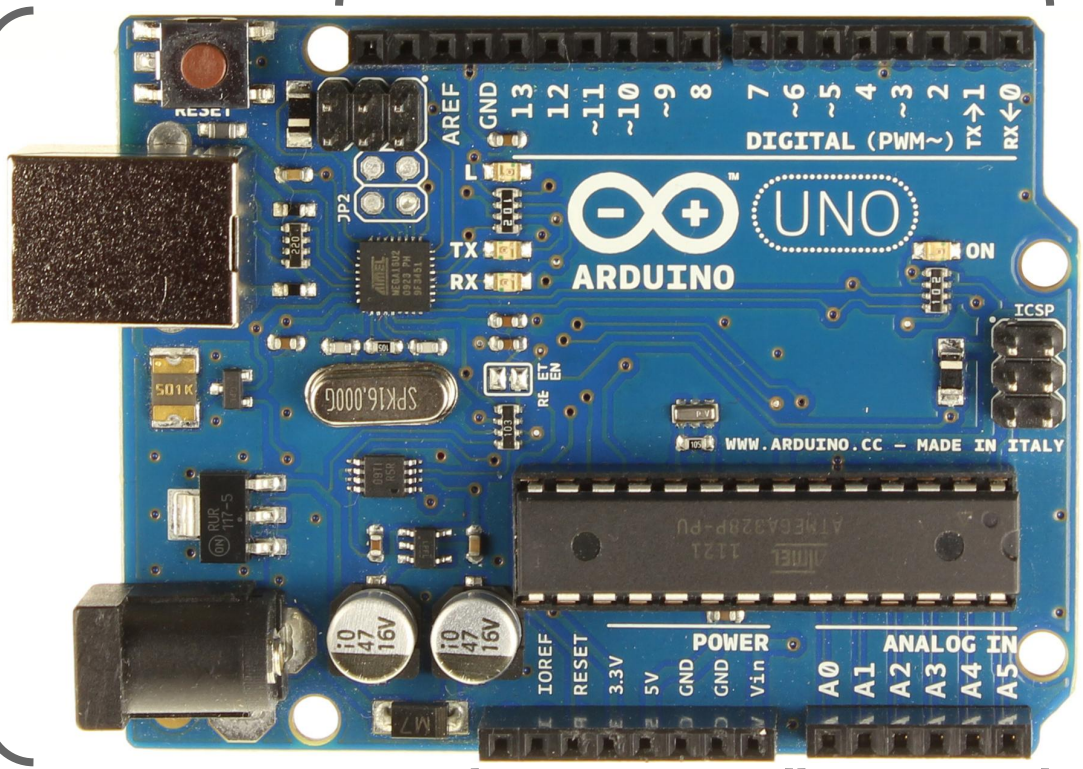
Layout

Digital I/O - PWM

USB

and

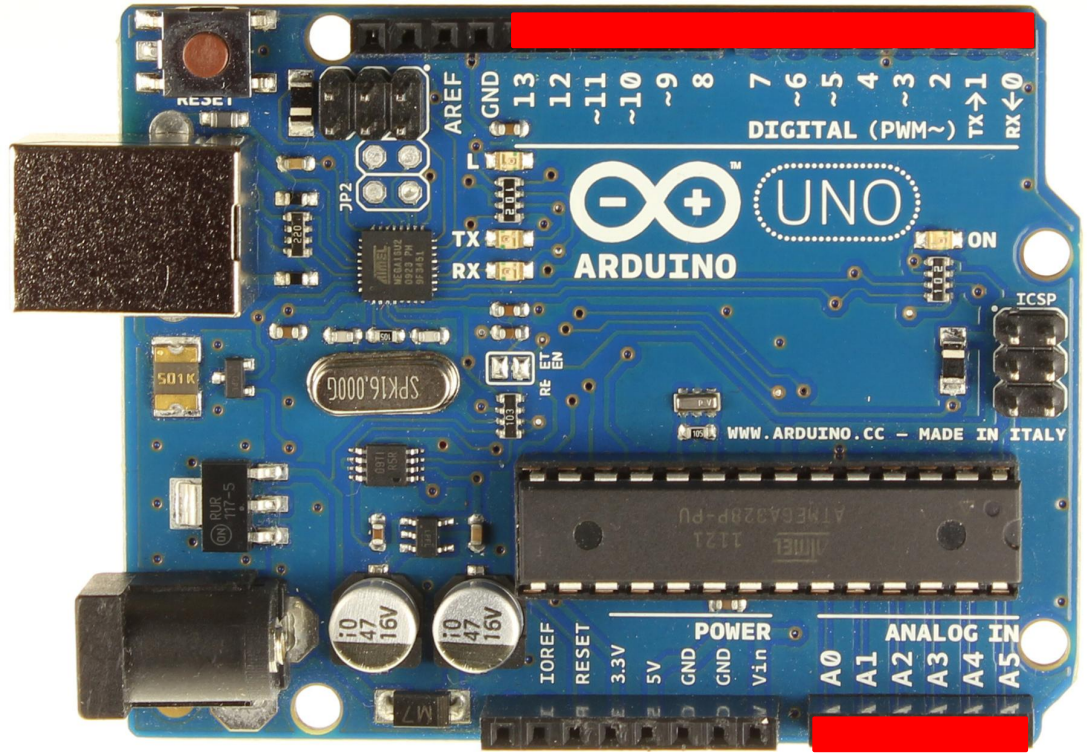
external
power
supply
plug



Power AnalogIn

Arduino Uno

Digital Input



Inizializzazione:

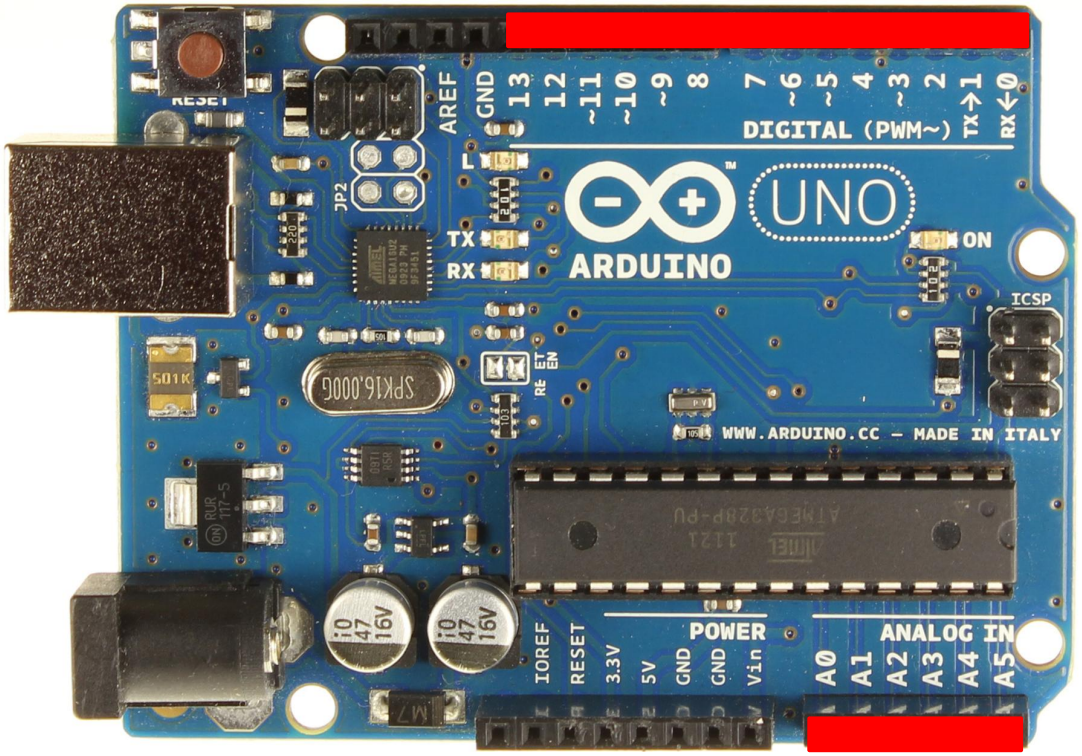
pinMode(pin, INPUT);

Utilizzo:

pinState = digitalRead(pin);

Arduino Uno

Digital Output



Inizializzazione:

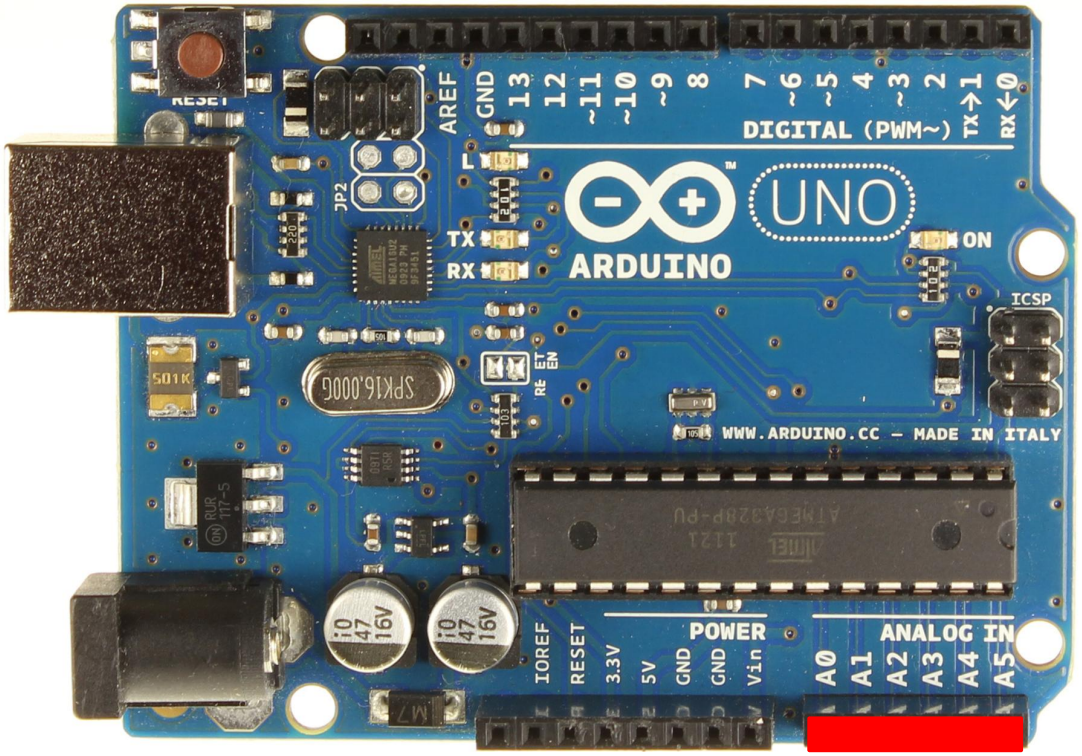
pinMode(pin, OUTPUT);

Utilizzo:

digitaWrite(pin, HIGH);
digitaWrite(pin, LOW);

Arduino Uno

Analog Input



Inizializzazione:

non necessaria

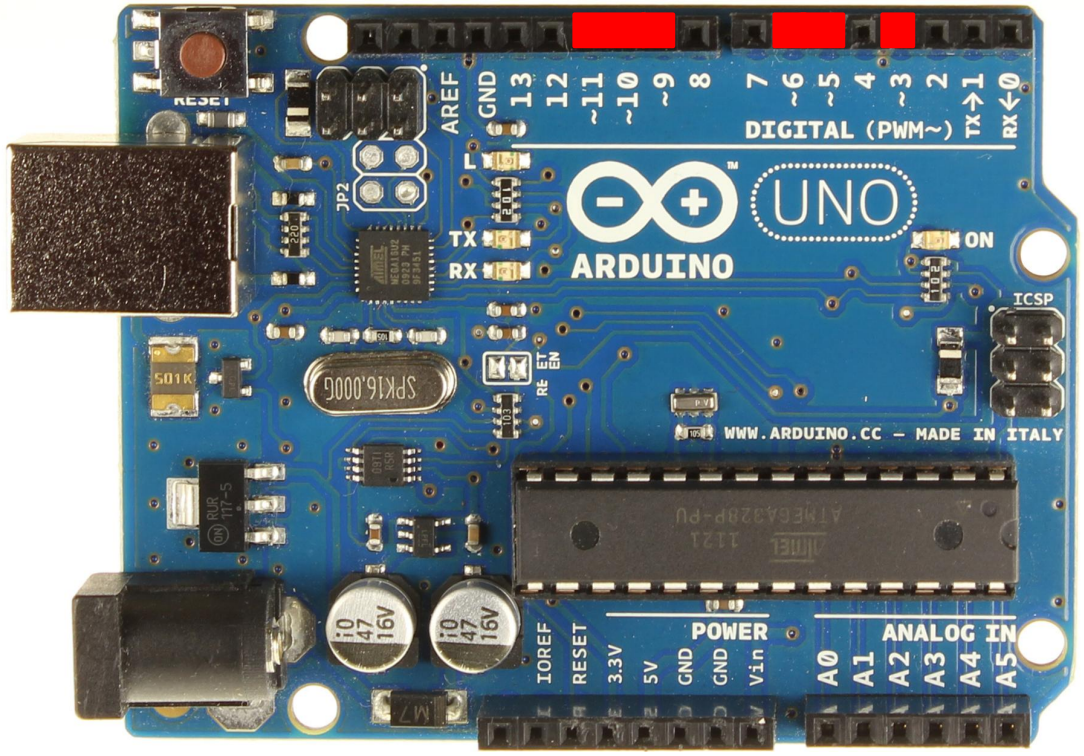
Utilizzo:

val = analogRead(A0...A5);

risoluzione 10-bit (0-1023)

Arduino Uno

Analog Output



Inizializzazione:

non necessaria

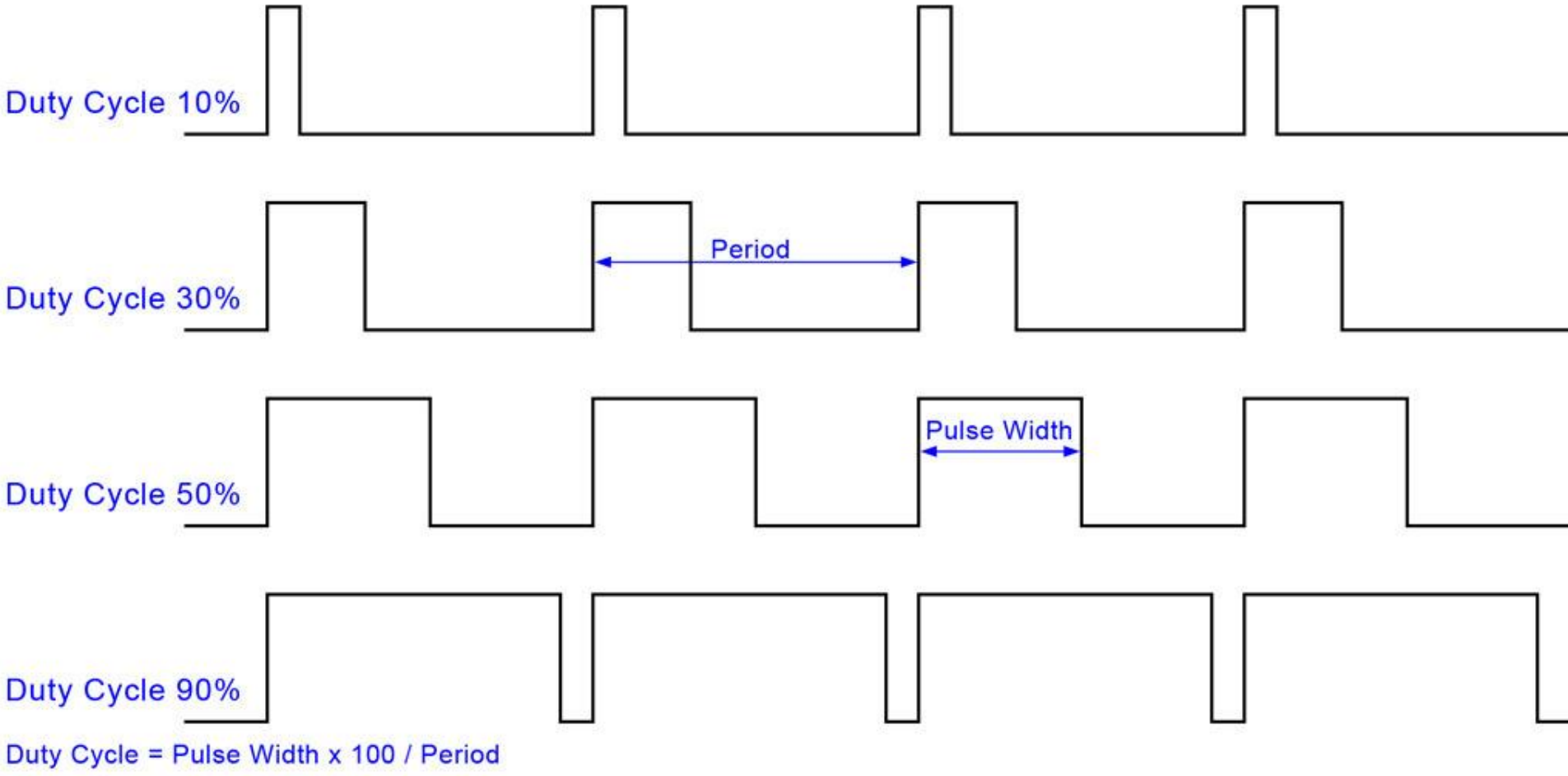
Utilizzo:

analogWrite(pin, val);

risoluzione 8-bit (0-255)

Arduino Uno

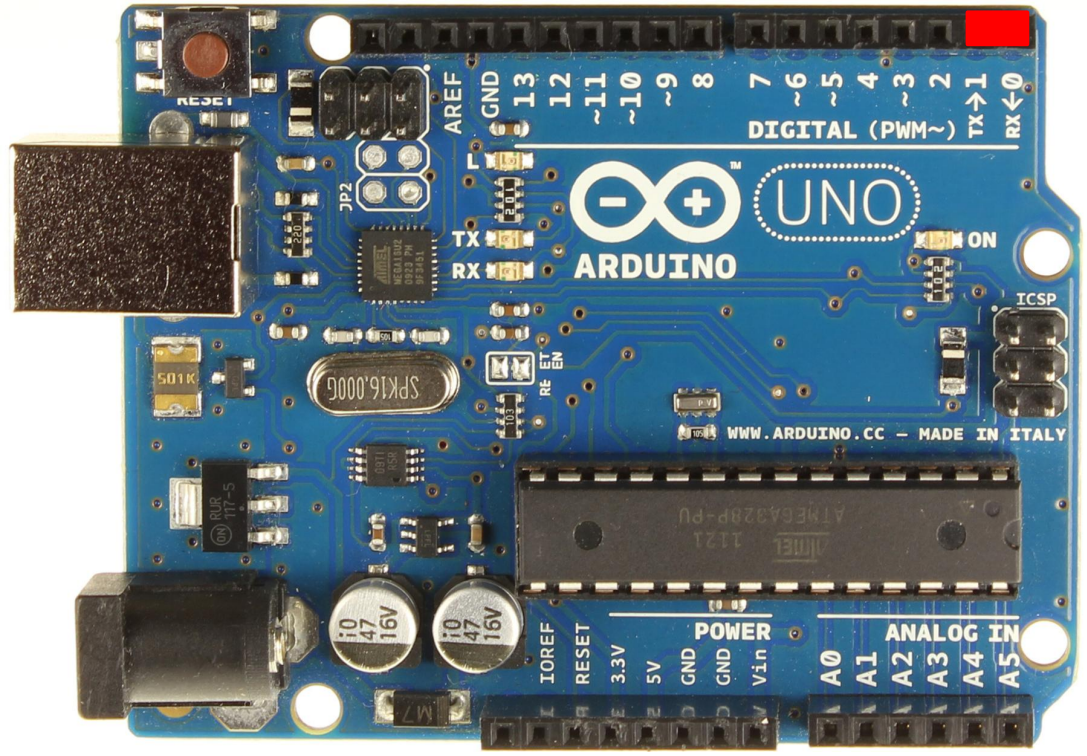
PWM



Duty Cycle = Pulse Width x 100 / Period

Arduino Uno

Serial Port



Inizializzazione:

Serial.begin(baudrate);

Utilizzo:

Serial.read(); // per leggere

Serial.print(); // per scrivere

Arduino Uno

Serial Methods

Class:

Serial

su schede con più seriali:

Serial1
Serial2
Serial3

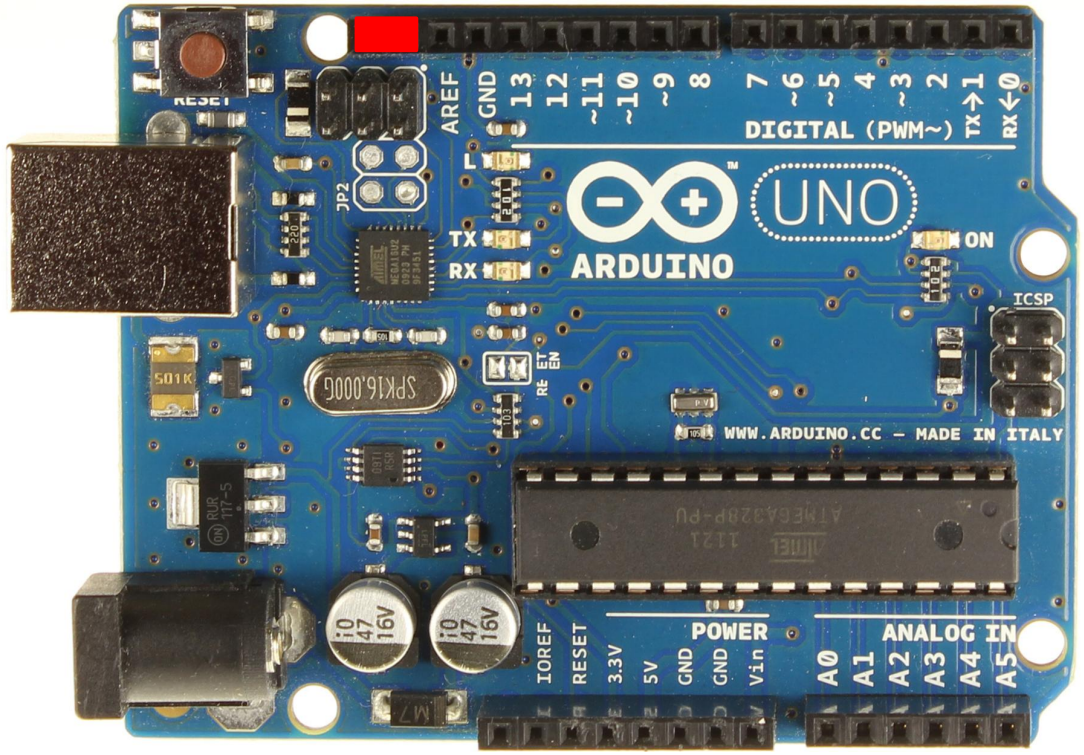
available()
begin()
end()
find()
findUntil()
flush()
parseFloat()
parseInt()
peek()
print()
println()
read()

readBytes()
readBytesUntil()
setTimeout()
write()
serialEvent()

arduino.cc/en/Reference/serial

Arduino Uno

TWI (Two Wires Interface)



Inizializzazione:

```
Wire.begin(slaveAddress);
```

Utilizzo:

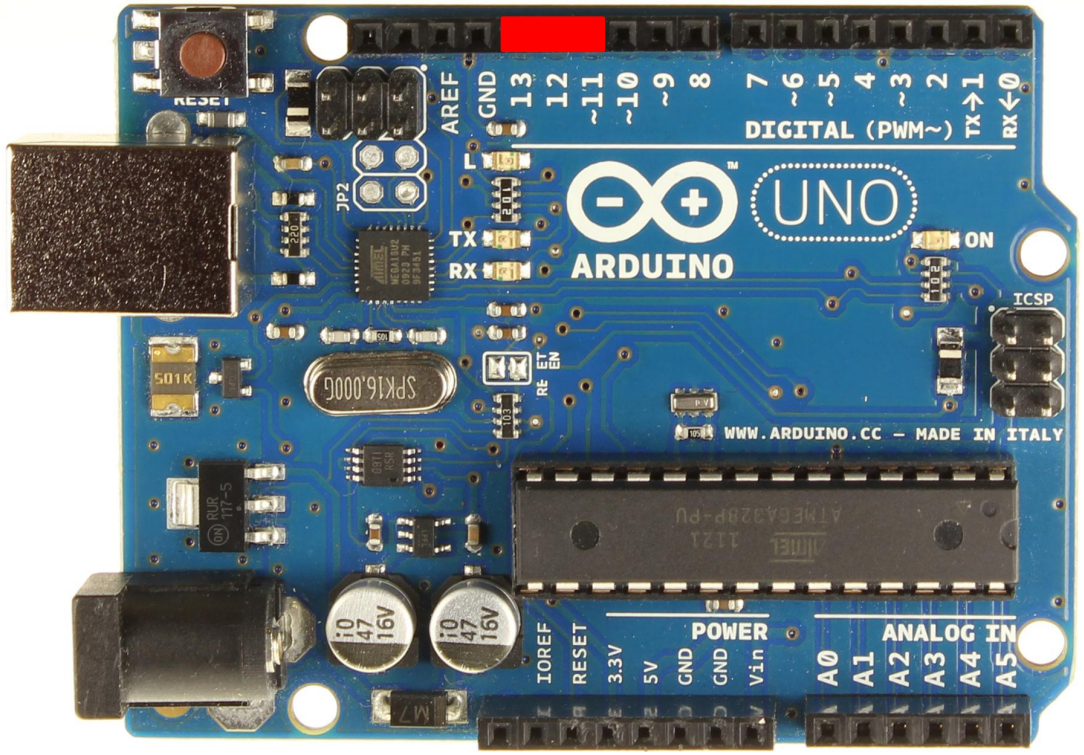
```
Wire.read();  // per leggere

Wire.write();  // per scrivere
```

arduino.cc/en/Reference/Wire

Arduino Uno

SPI (Serial Peripheral Interface)



Inizializzazione:

```
SPI.begin();
SPI.end();
```

Utilizzo:

```
SPI.trasfer(val);  // trasferisce
                   // 1 byte
```

arduino.cc/en/Reference/SPI

Arduino IDE

Layout

Compile
Upload

Serial Monitor

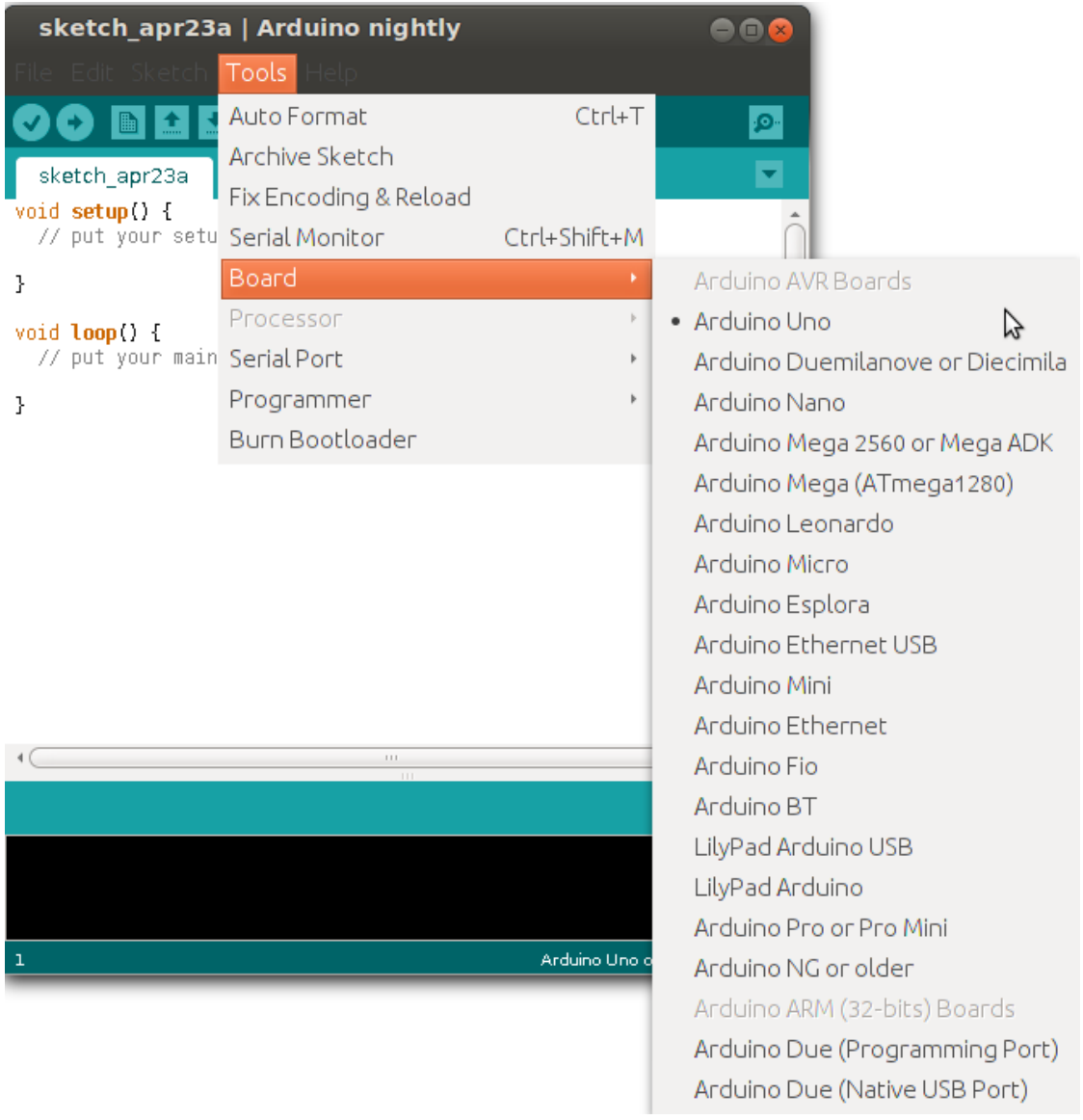
Edit Area

Message
Area

Board and Po

Arduino IDE

Board Selection



Arduino IDE

Blink - Arduino "Hello Word!"

```
const int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Arduino IDE

Fade

```
int led = 9;                // the pin that the LED is attached to
int brightness = 0;         // how bright the LED is
int fadeAmount = 5;         // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  { }

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Arduino IDE

AnalogInOutSerial #1

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9;  // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);

  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);

  // change the analog out value:
  analogWrite(analogOutPin, outputValue);
```

Arduino IDE

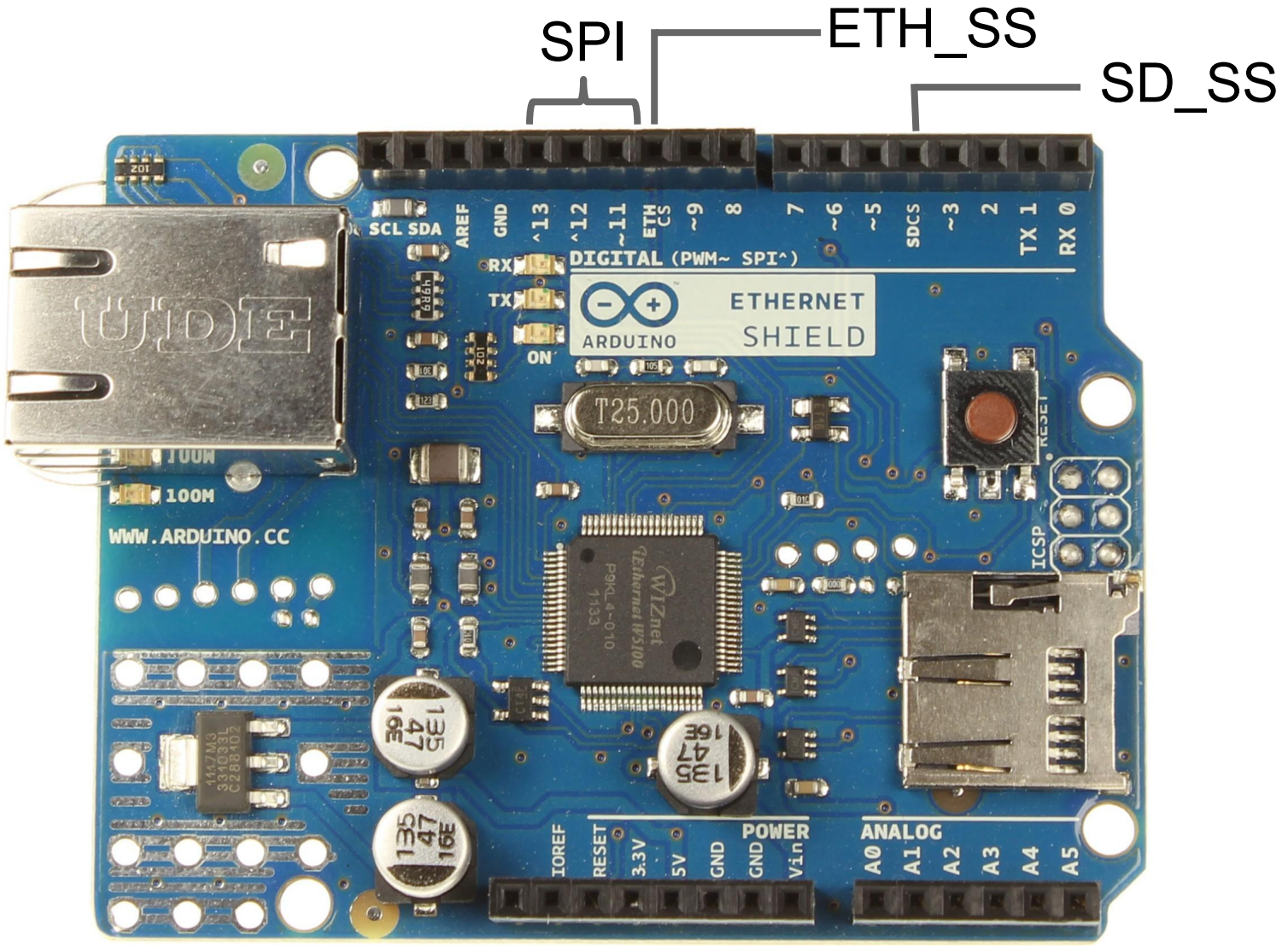
AnalogInOutSerial #2

```
  // print the results to the serial monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```

Arduino Shields

Ethernet Shield



Arduino Shields

Ethernet Library

Ethernet Class

begin()

localIP()

maintain()

IPAddress Class

IPAddress()

Server Class

Server

EthernetServer()

begin()

available()

write()

print()

println()

Client Class

Client

EthernetClient()

if (EthernetClient)

connected()

connect()

write()

print()

println()

available()

read()

flush()

stop()

Arduino IDE

Ethernet WebServer

```
#include <SPI.h>
#include <Ethernet.h>
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

IPAddress ip(192,168,1, 177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
   // Open serial communications and wait for port to open:
  Serial.begin(9600);

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
```

Arduino IDE

Ethernet WebServer

```
void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
```

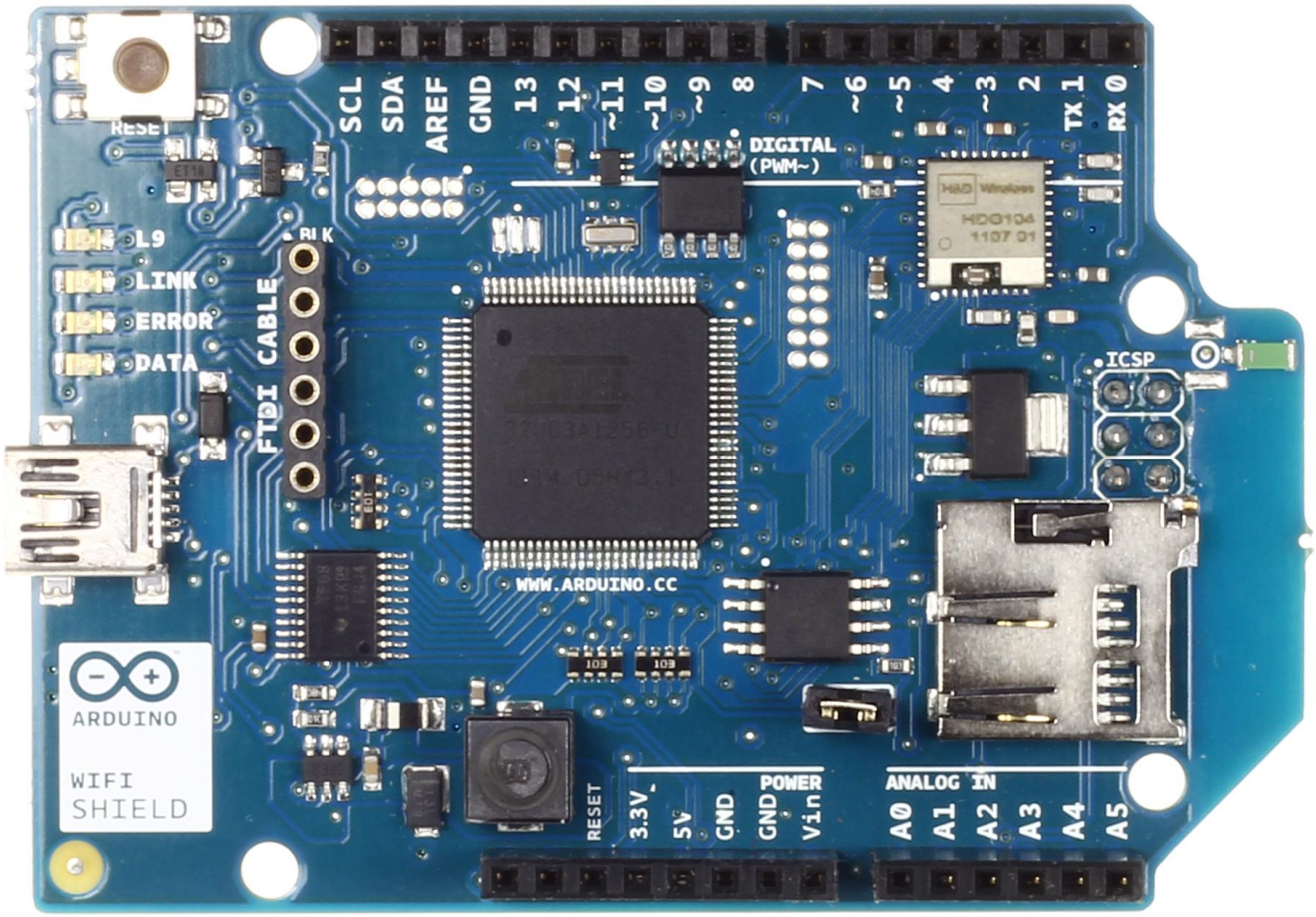
Arduino IDE

Ethernet WebServer

```
          client.println("<meta http-equiv=\"refresh\" content=\"5\">");
          // output the value of each analog input pin
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n') {  currentLineIsBlank = true;   }
        else if (c != '\r') {  currentLineIsBlank = false;   }
      }
    }
    delay(1);    // give the web browser time to receive the data
    client.stop();    // close the connection:
    Serial.println("client disonnected");
  }
}
```

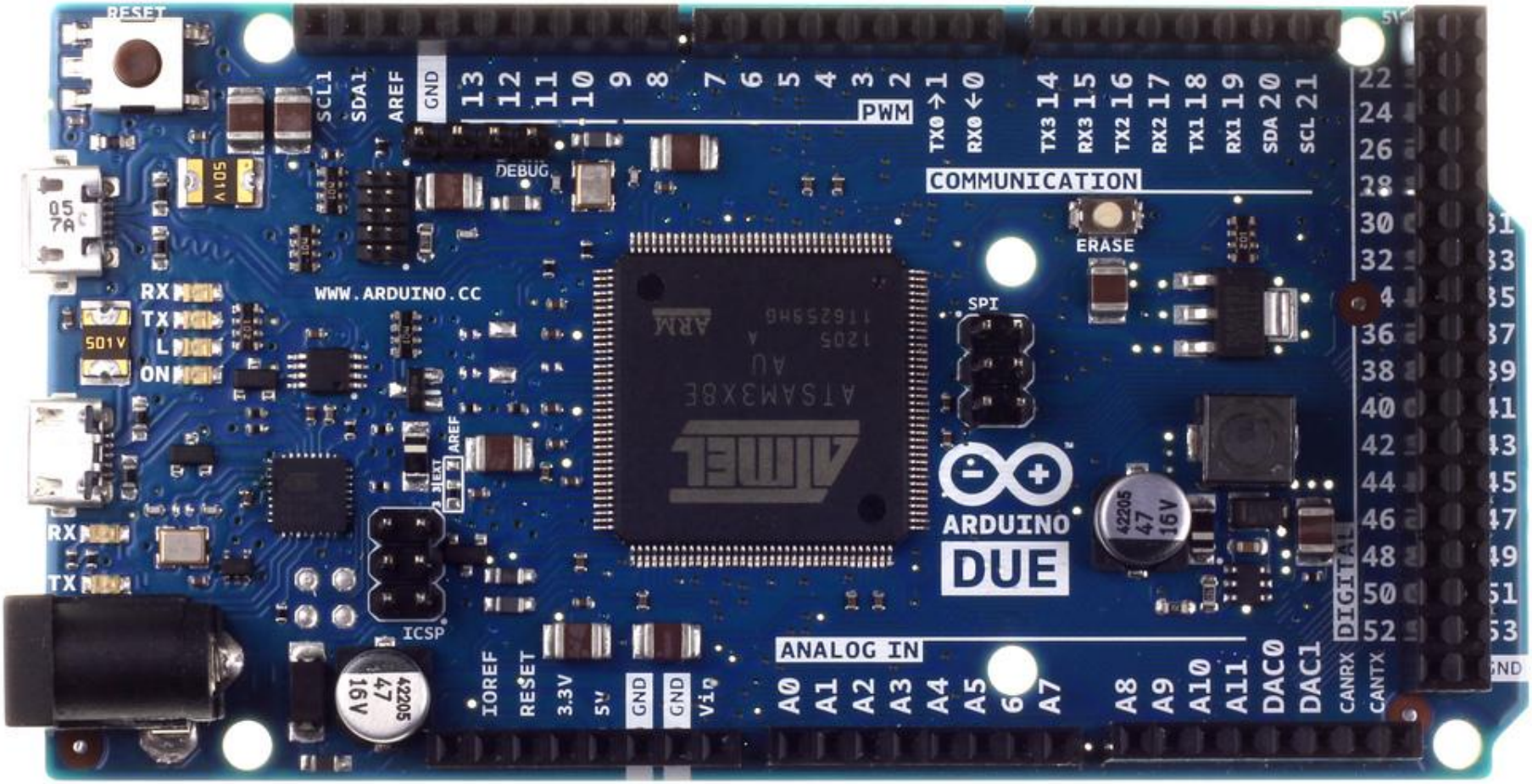
Arduino Shields

WiFi Shield



Arduino Due

Due



Arduino Due

Due

ARM Core benefits

The Due has a 32-bit ARM core that can outperform typical 8-bit microcontroller boards. The most significant differences are:

- A 32-bit core, that allows operations on 4 bytes wide data within a single CPU clock. (for more information look int type page).
- CPU Clock at 84Mhz.
- 96 KBytes of SRAM.
- 512 KBytes of Flash memory for code.
- a DMA controller, that can relieve the CPU from doing memory intensive tasks.

Arduino Due

Due

Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Outputs Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz