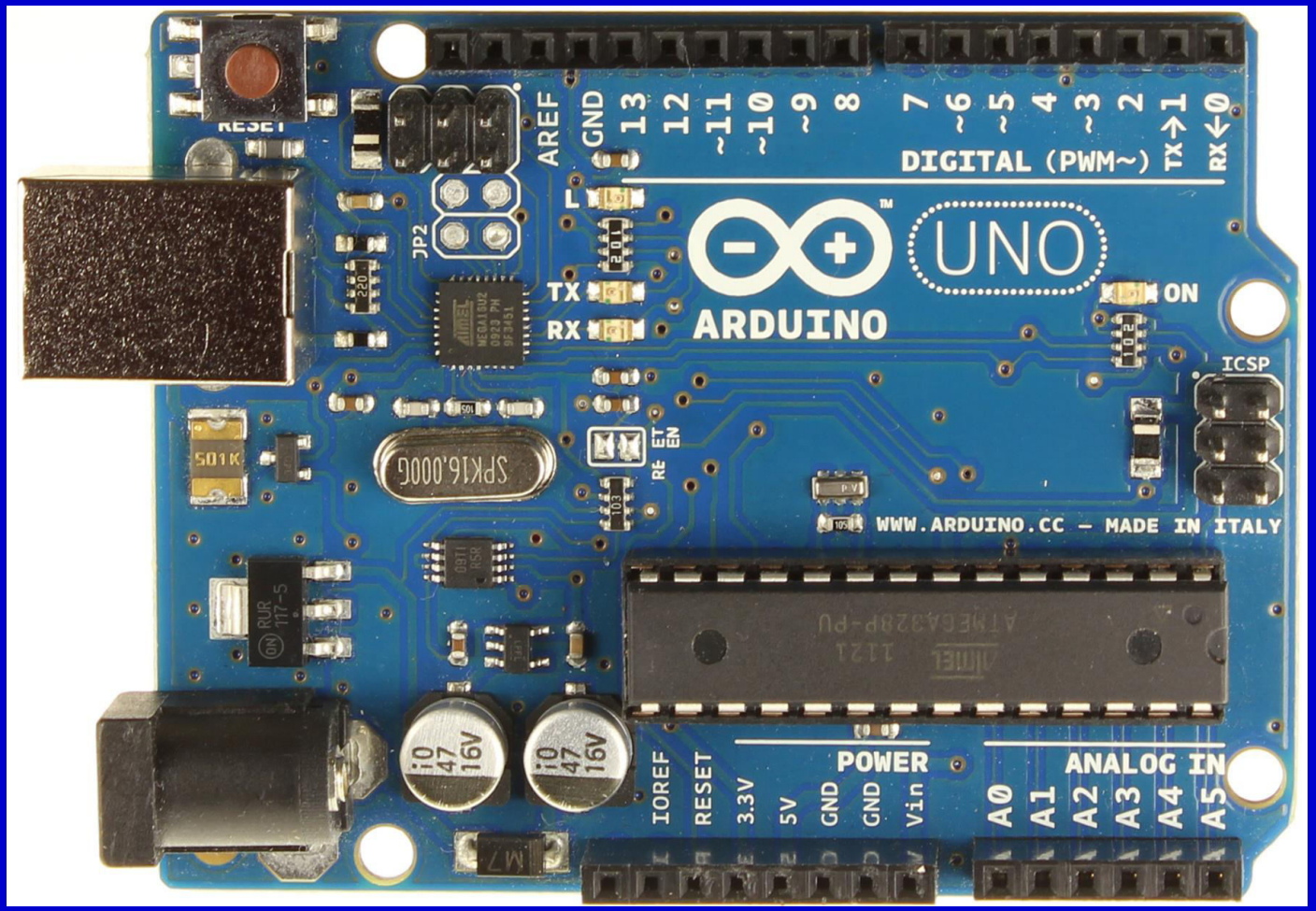
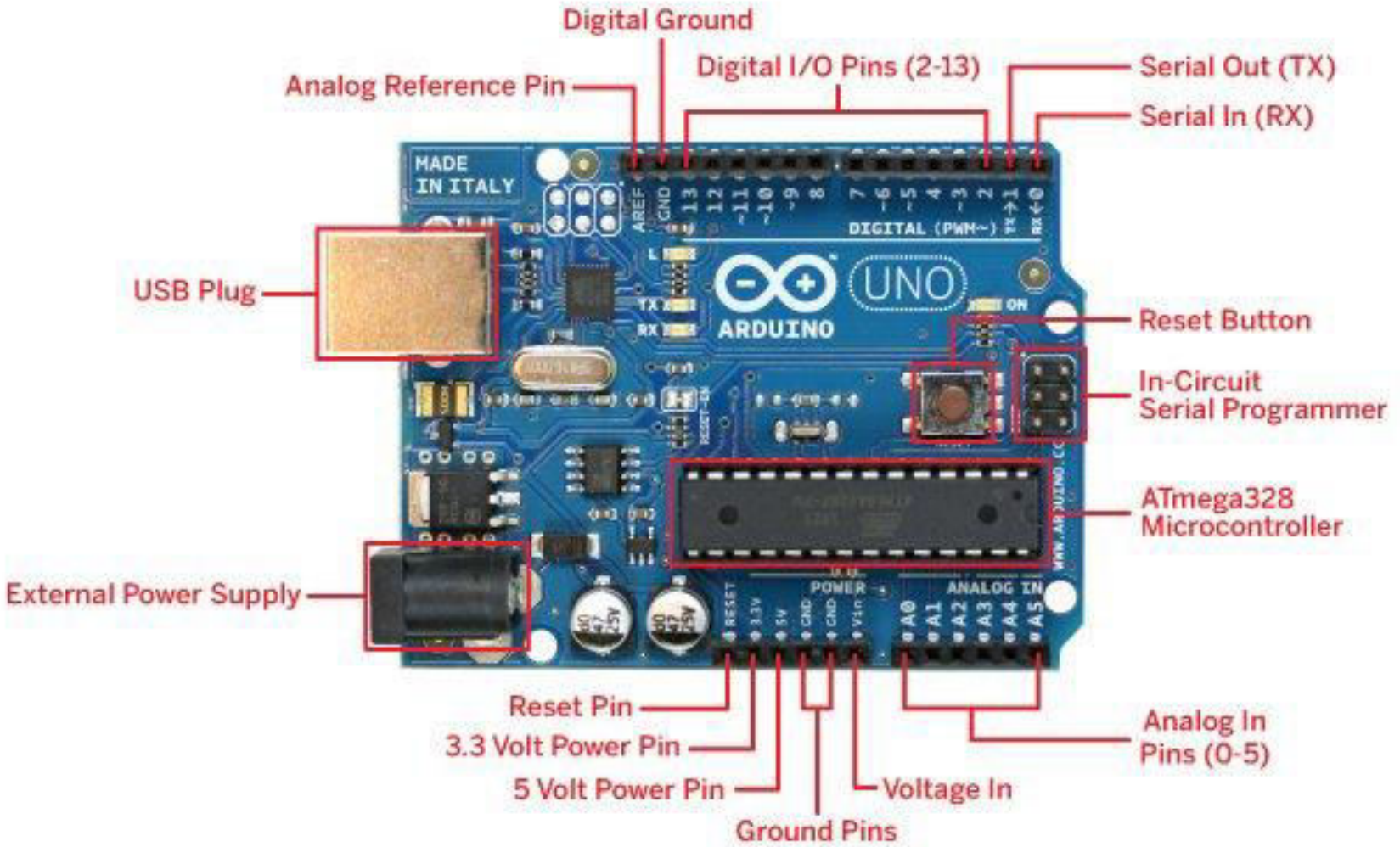
Θύρες εισόδου/εξόδου του Arduino

Ακίδες I/O του Arduino



Ακίδες I/O του Arduino



Εντολές εισόδου/εξόδου του Arduino

Digital I/O Pin modes

INPUT,

INPUT_PULLUP,

OUTPUT

HIGH

When a pin is configured as an INPUT with pinMode(), and read with digitalRead(), the Arduino (ATmega) will report HIGH if

- A voltage greater than 3 volts is present at the pin (5V boards)
- A voltage greater than 2 volts is present at the pin (3.3V boards)

When a pin is configured to OUTPUT with pinMode(), and set to HIGH with digitalWrite(), the pin is at:

- 5 volts (5V boards)
- 3.3 volts (3.3V boards)

LOW

When a pin is configured as an INPUT with pinMode(), and read with digitalRead(), the Arduino (ATmega) will report LOW if:

- a voltage less than 3 volts is present at the pin (5V boards)
- a voltage less than 2 volts is present at the pin (3.3V boards)

When a pin is configured to OUTPUT with pinMode(), and set to LOW with digitalWrite(), the pin is at 0 volts (both 5V and 3.3V boards).

pinMode(pin, mode)

Η εντολή pinMode(pin, mode) καθορίζει ότι η συγκεκριμένη ψηφιακή ακίδα (pin) του Arduino θα λειτουργεί σαν **είσοδος η σαν έξοδος.**

pin: the number of the pin whose mode you wish to set

mode: INPUT, OUTPUT, or INPUT_PULLUP.

digitalWrite()

Writes a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

Atmega pins can source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), for example, **but not enough current to run most relays, solenoids, or motors.**

digitalRead(pin)

digitalRead(*pin*) reads the value from a specified digital pin, either HIGH or LOW.
pin: the number of the digital pin you want to read (*int*)

delay(ms)

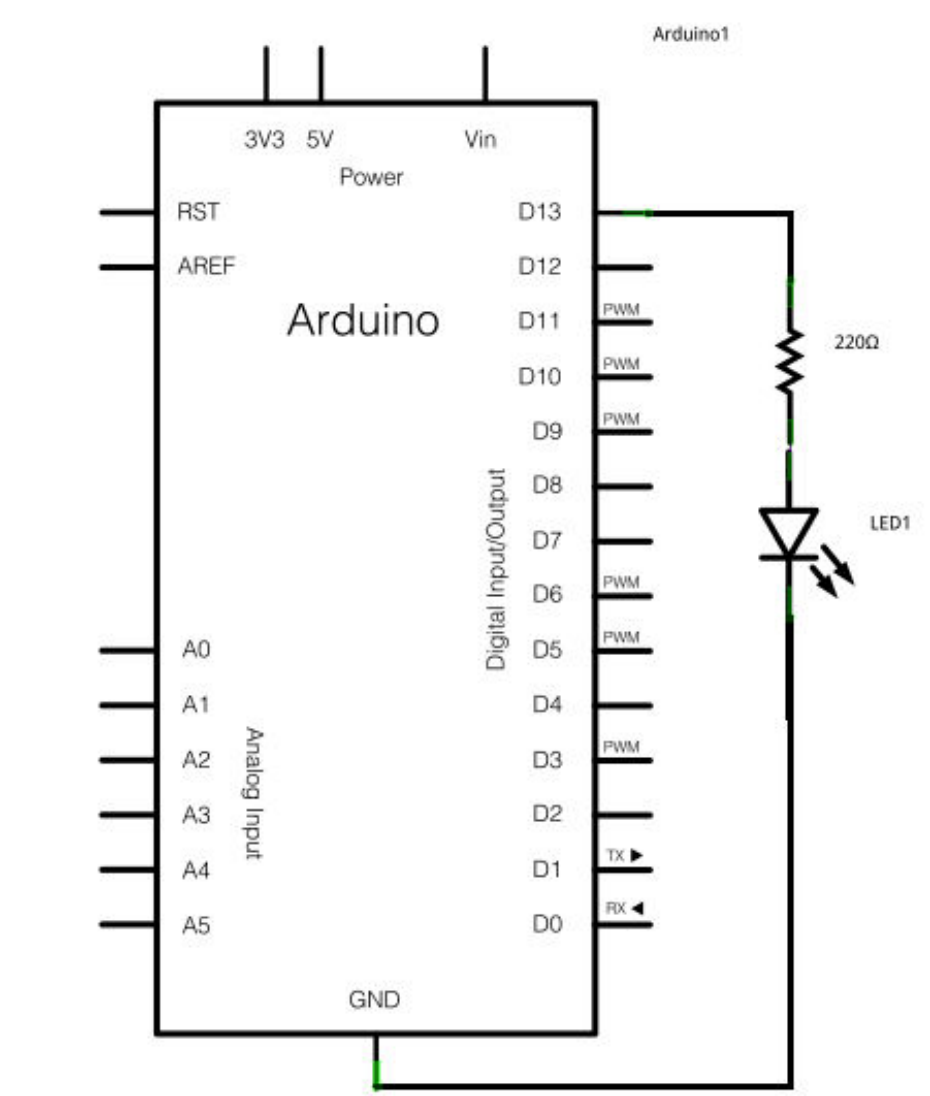
delay(*ms*) pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second).

millis()

Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Ενσωματωμένο LED του Arduino

Κύκλωμα ενσωματωμένου LED του Arduino



Πρόγραμμα που αναβοσβήνει το LED που είναι τοποθετημένο στην πόρτα 13 κάθε ένα δευτερόλεπτο.

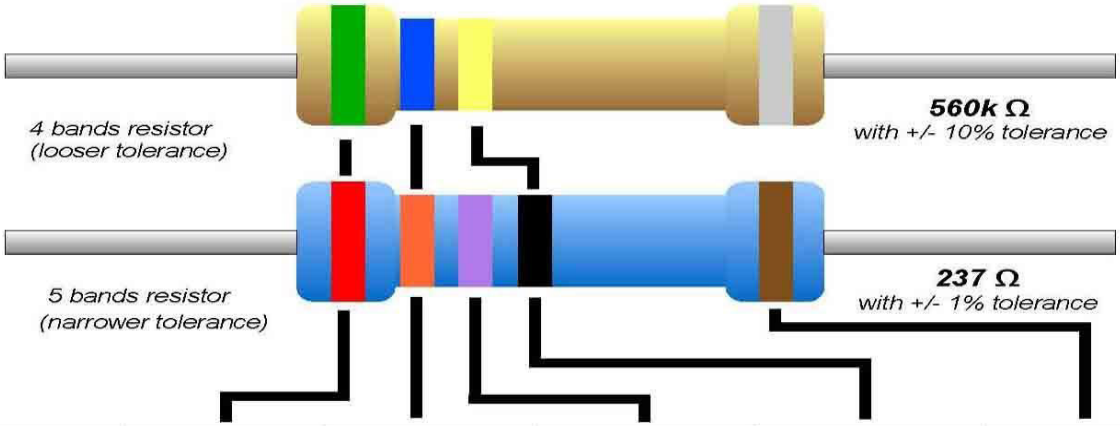
```
int ledPin = 13;    // LED connected to digital pin 13

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Αυξομοιώστε τον χρόνο που το LED 13 παραμένει αναμμένο τροποποιώντας το delay, μεταφράζοντας φορτώνοντας και εκτελώντας το πρόγραμμα.

Resistors

A *resistor* is a passive two-terminal electrical component that implements electrical resistance as a circuit element. Resistors may be used to reduce current flow, and, at the same time, may act to lower voltage levels within circuits.

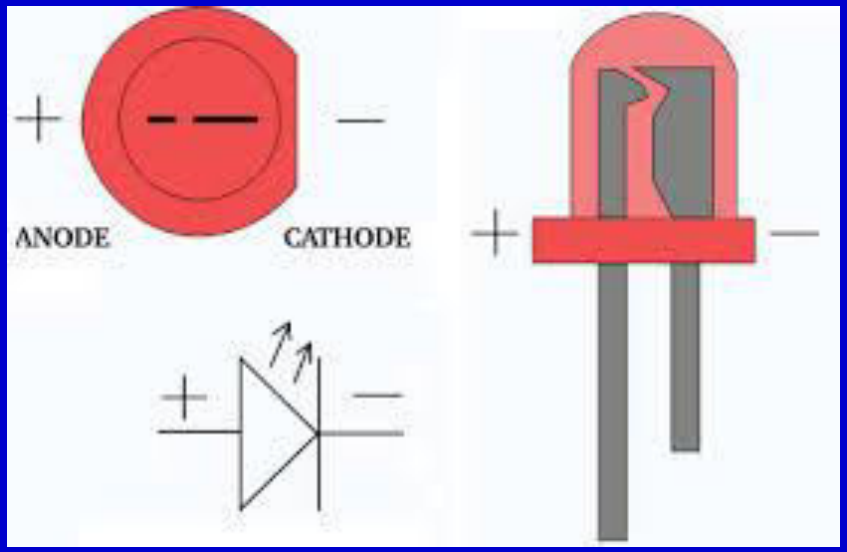
Resistor Color Code



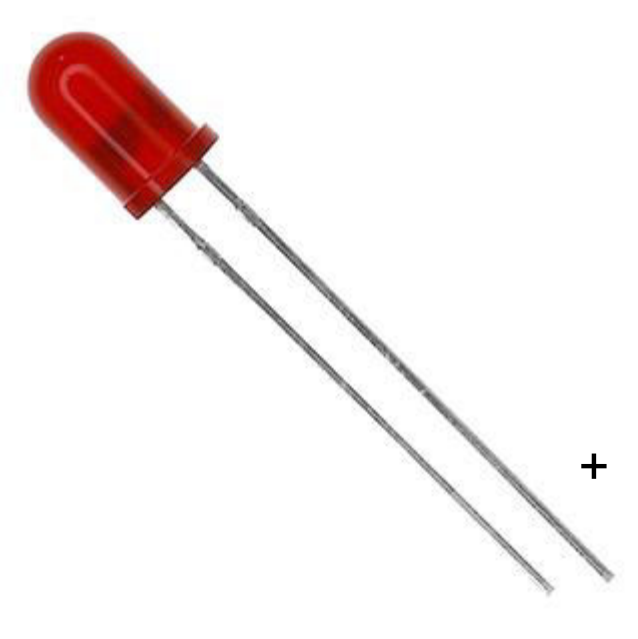
Color	1st Band	2nd Band	3rd Band	Multiplier	Tolerance
Black	0	0	0	x 1 Ω	
Brown	1	1	1	x 10 Ω	+/- 1%
Red	2	2	2	x 100 Ω	+/- 2%
Orange	3	3	3	x 1K Ω	
Yellow	4	4	4	x 10K Ω	
Green	5	5	5	x 100K Ω	+/- .5%
Blue	6	6	6	x 1M Ω	+/- .25%
Violet	7	7	7	x 10M Ω	+/- .1%
Grey	8	8	8		+/- .05%
White	9	9	9		
Gold				x .1 Ω	+/- 5%
Silver				x .01 Ω	+/- 10%

LED

Τα LED είναι ειδικές δίοδοι ημιαγωγού που φωτοβολούν όταν μέσα από αυτές διέρχεται η κατάλληλη τιμή ηλεκτρικού ρεύματος. Διακρίνονται σε miniature και high power.

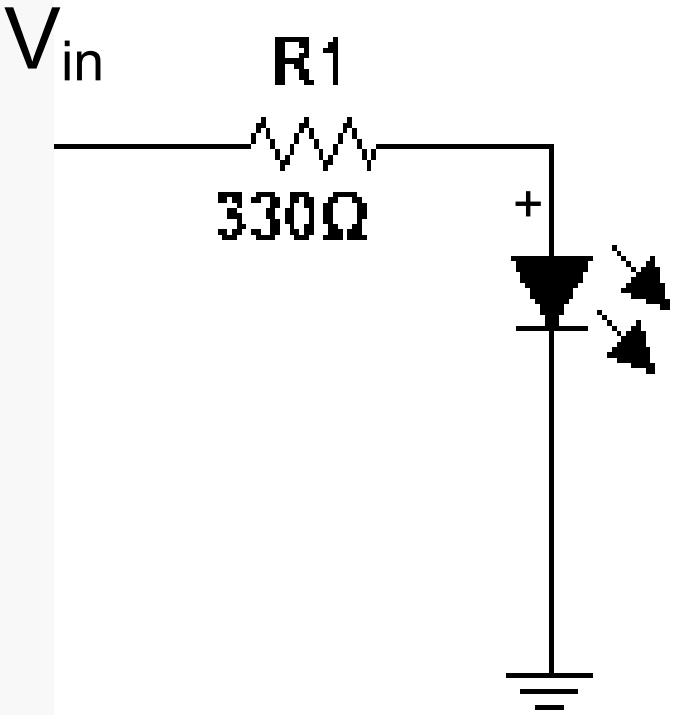


LED 5mm κόκκινο

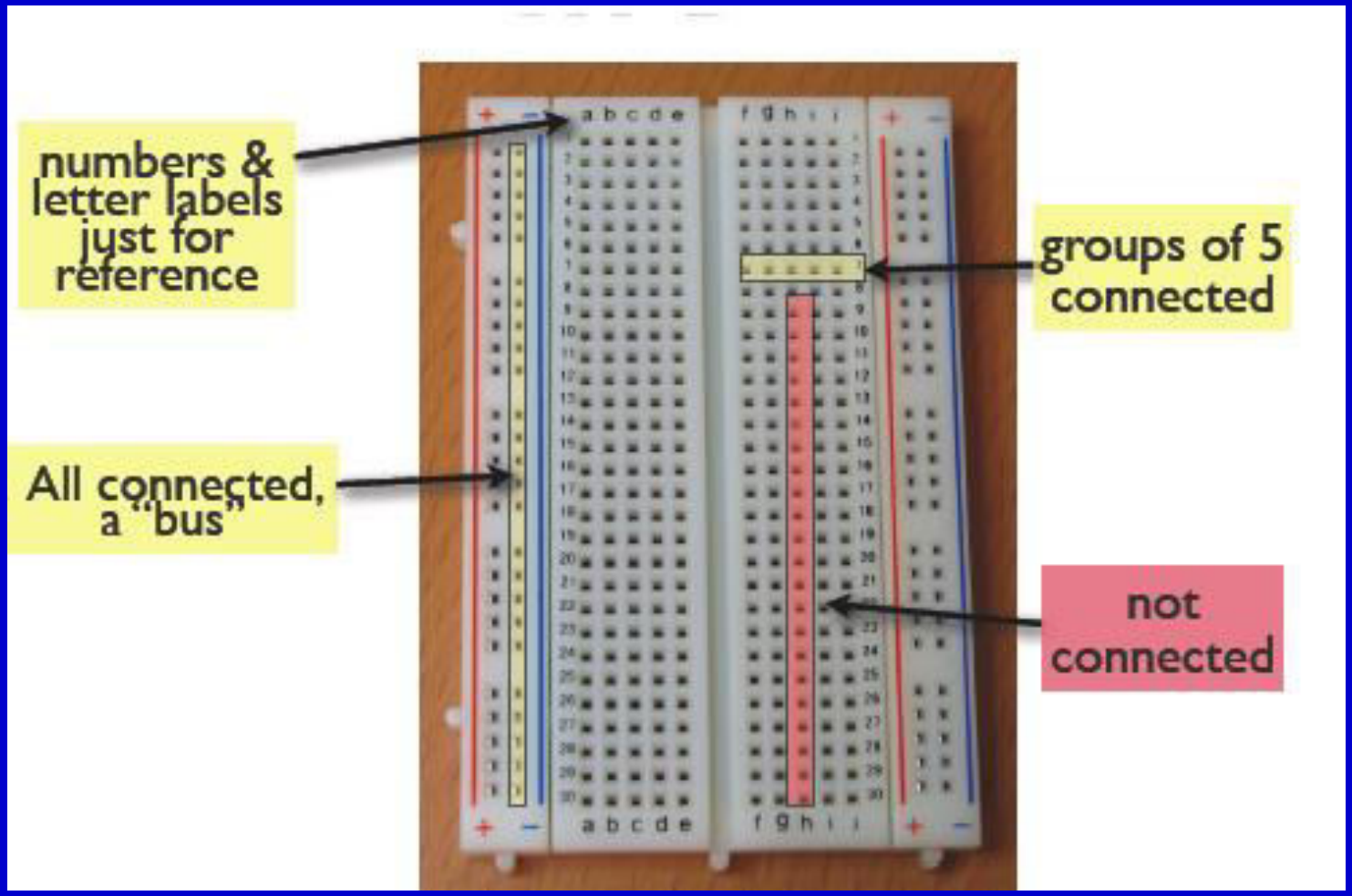


1.8-2.2V DC forward drop
Max current: 20mA
Suggested using current: 16-18mA
Luminous Intensity: 150-200mcd

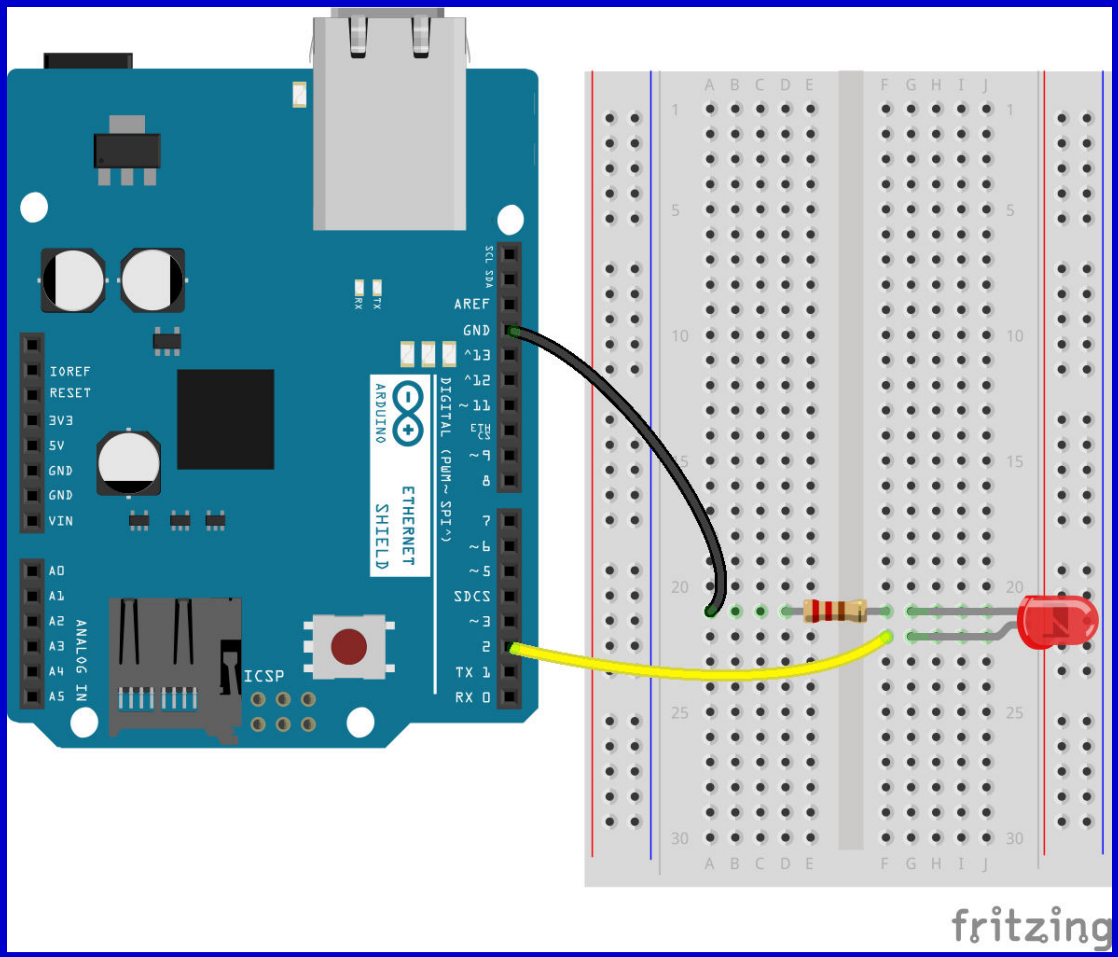
Κύκλωμα ένδειξης του 0 και του 1 με LED. Το LED ανάβει για V_{in}=5V.



Breadboard


numbers & letter labels just for reference
All connected, a "bus"
groups of 5 connected
not connected

Σύνδεση LED στην πόρτα 2 του Arduino



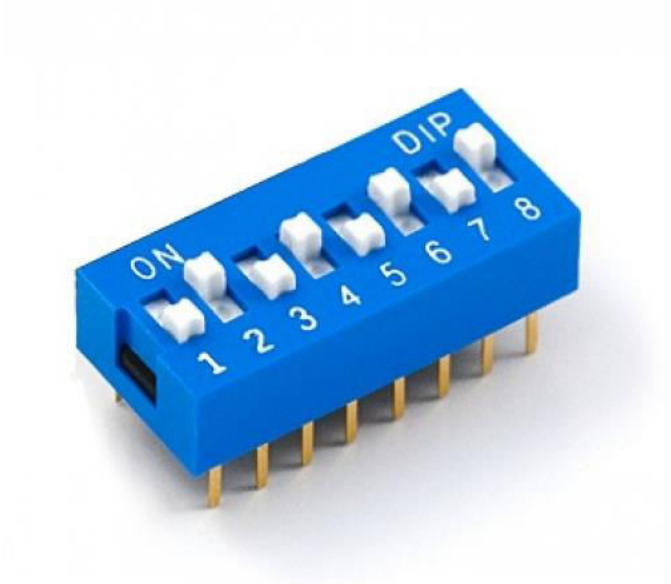
Πρόγραμμα που αναβοσβήνει το LED που είναι τοποθετημένο στην πόρτα 2 κάθε μισό δευτερόλεπτο.

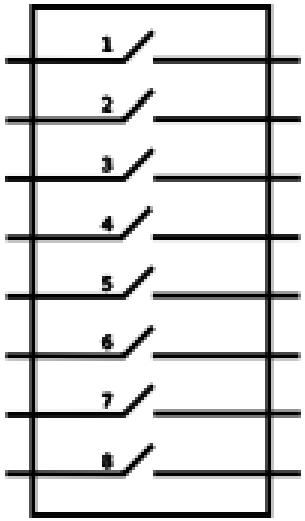
```
int ledPin = 2;    // LED connected to digital pin 2

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(500);                   // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(500);                   // waits for a second
}
```

DIP switch

A *DIP switch* is a manual electric switch that is packaged with others in a group in a standard dual in-line package (DIP). The term may refer to each individual switch, or to the unit as a whole. This type of switch is designed to be used on a printed circuit board along with other electronic components and is commonly used to customize the behavior of an electronic device for specific situations

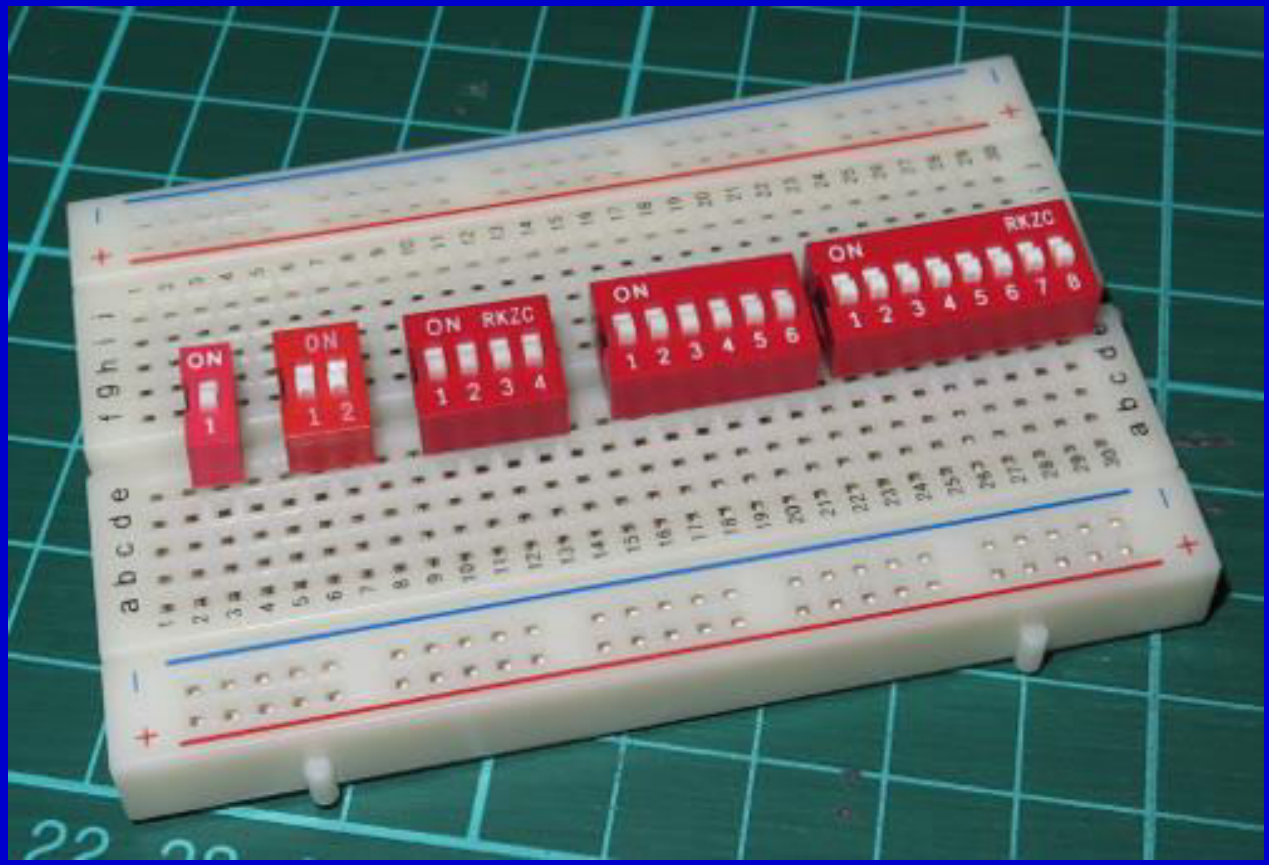
Dip Switches





Στην θέση ON οι αντίστοιχες ακίδες είναι βραχυκυκλωμένες

Τοποθέτηση DIP switch σε breadboard



Κυκλώματα παραγωγής των 0, 1 με switch.

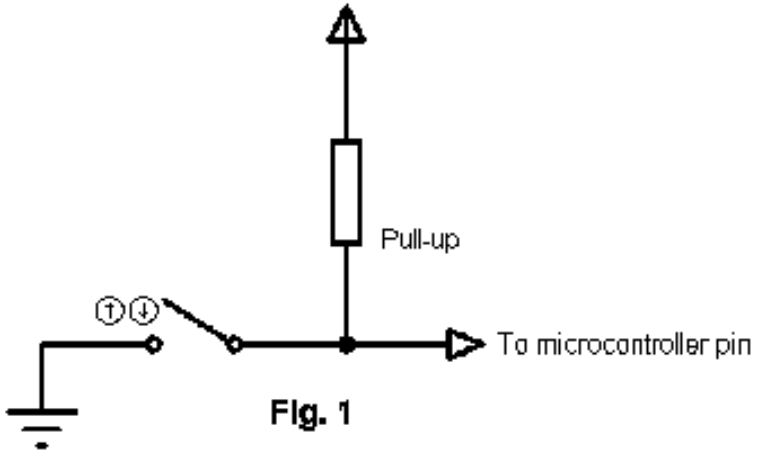


Fig. 1

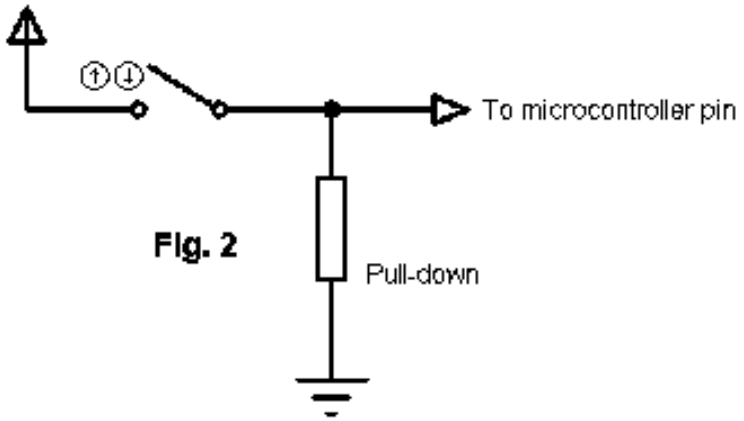
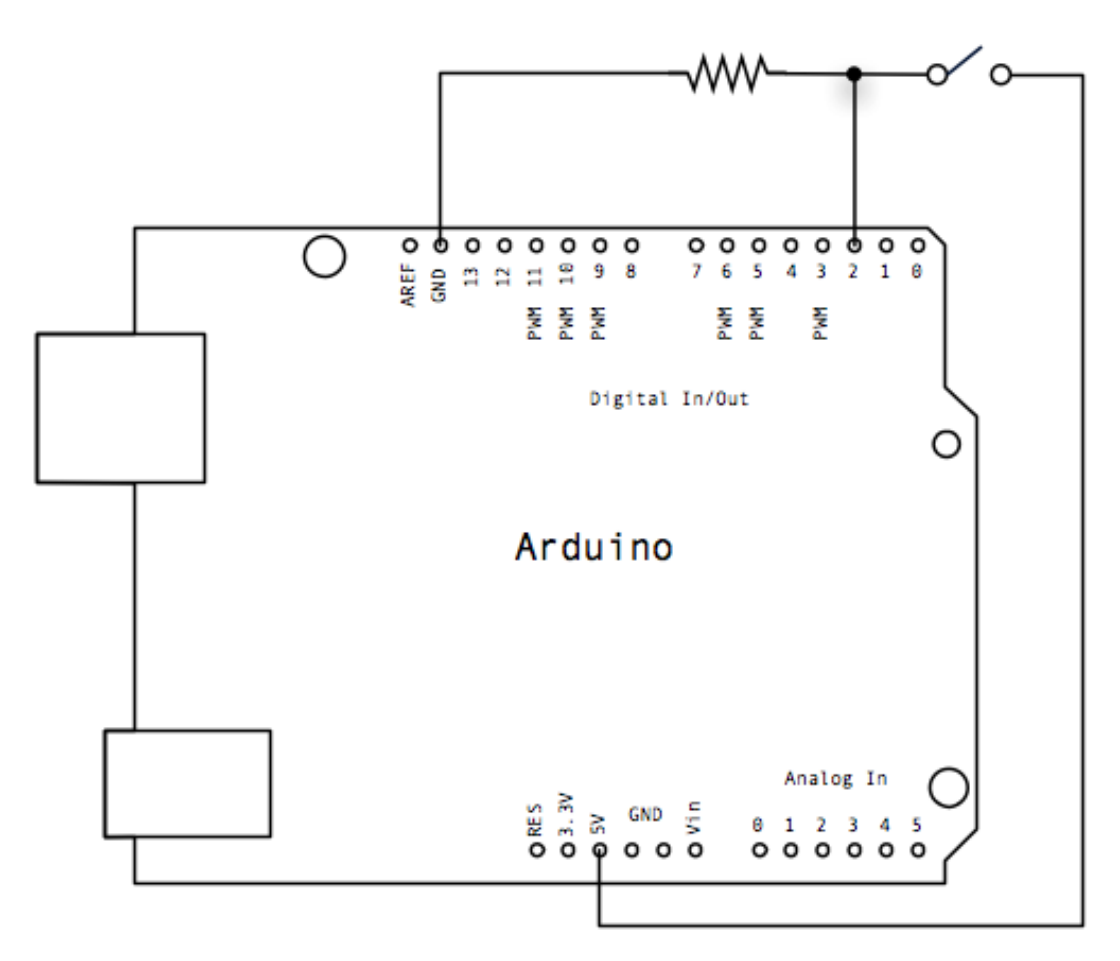


Fig. 2

Σύνδεση Arduino με κύκλωμα παραγωγής των 0, 1 με αντίσταση σε συνδεσμολογία pull-down



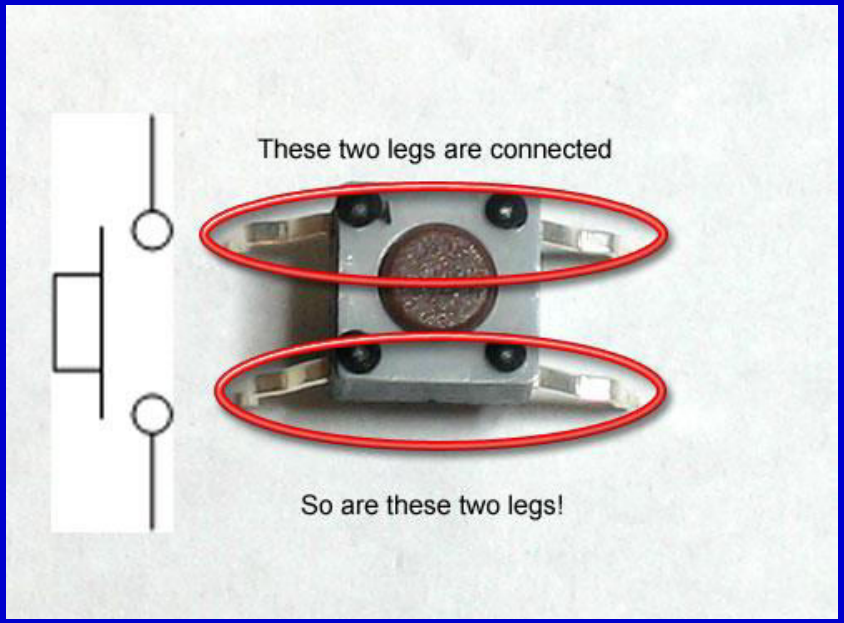
Πρόγραμμα αυτό η τιμή του pin 13 με το LED γίνεται η ίδια με την τιμή εισόδου στο pin 2. Η τιμή της εισόδου στο pin 2 καθορίζεται με χρήση DIP switch και αντίστασης.

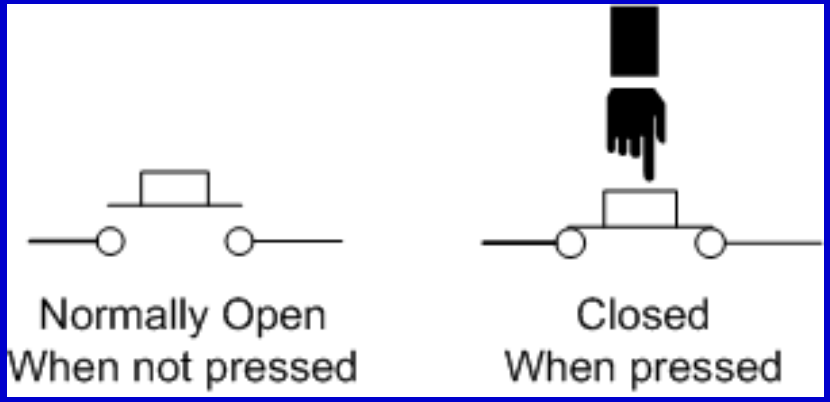
```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 2;     // switch connected to digital pin 2
int val = 0;       // variable to store the read value
void setup()  {
     pinMode(ledPin, OUTPUT);      // sets the digital pin 13 as output
     pinMode(inPin, INPUT);          // sets the digital pin 7 as input
   }
void loop()  {
     val = digitalRead(inPin);     // read the input pin
     digitalWrite(ledPin, val);    // sets the LED to the button's value
   }
```

Internal input pull-up resistors

There are 20 KOhm pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed by setting the pinMode() as INPUT_PULLUP.

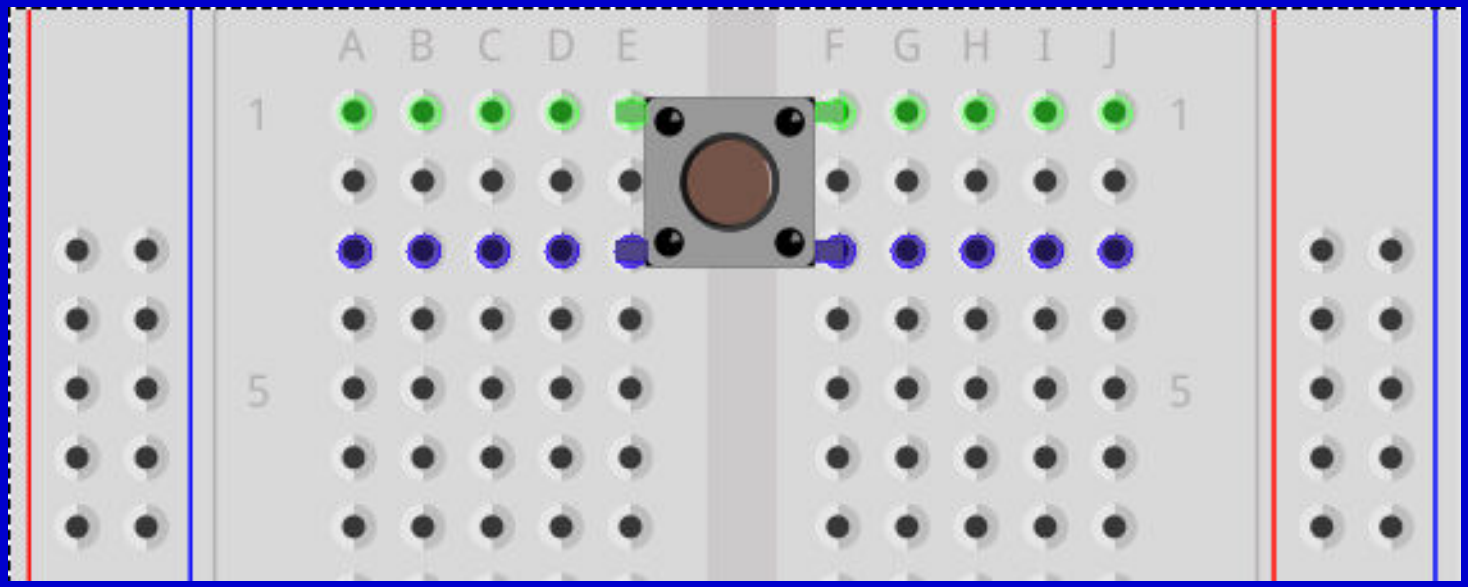
Push button (Tack switch)



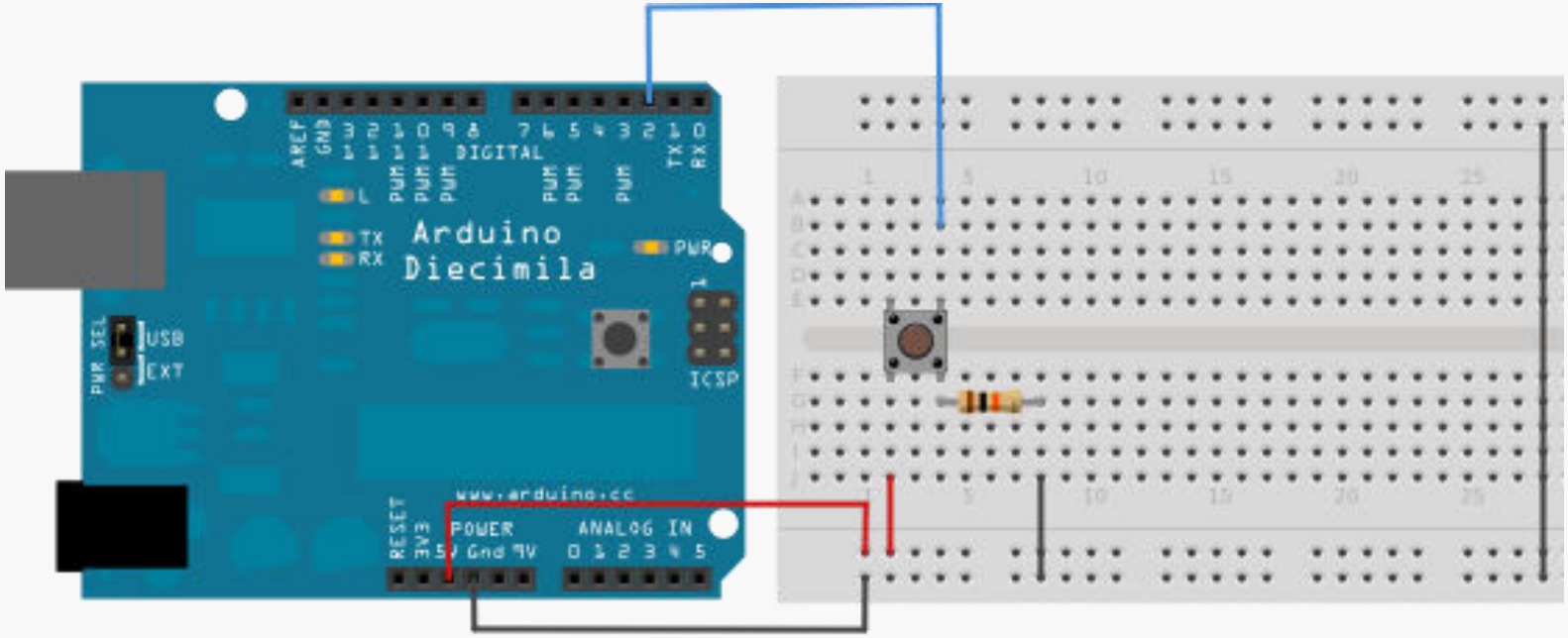


T

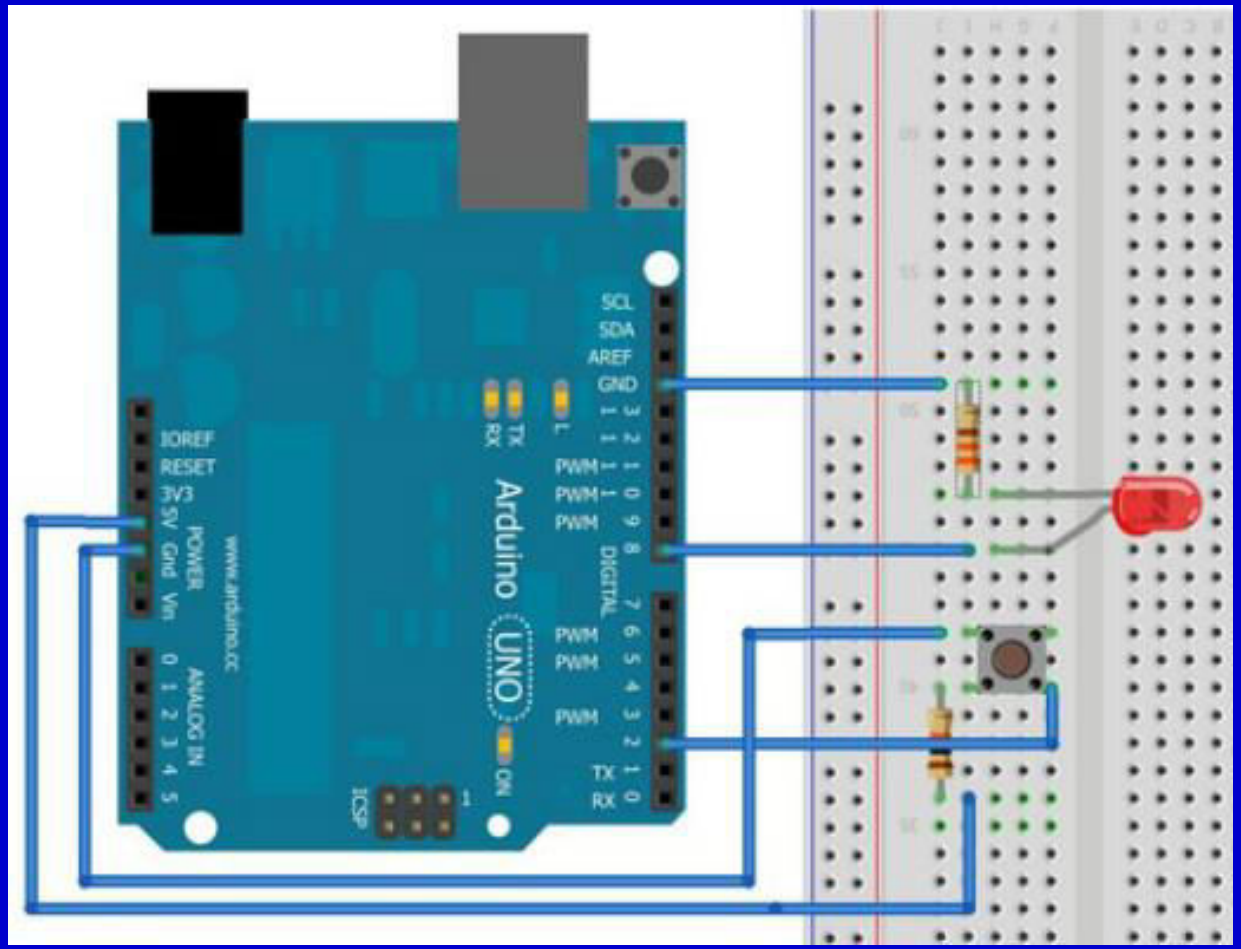
Τοποθέτηση tack switch σε breadboard



Σύνδεση Arduino με tact switch και pull down resistor



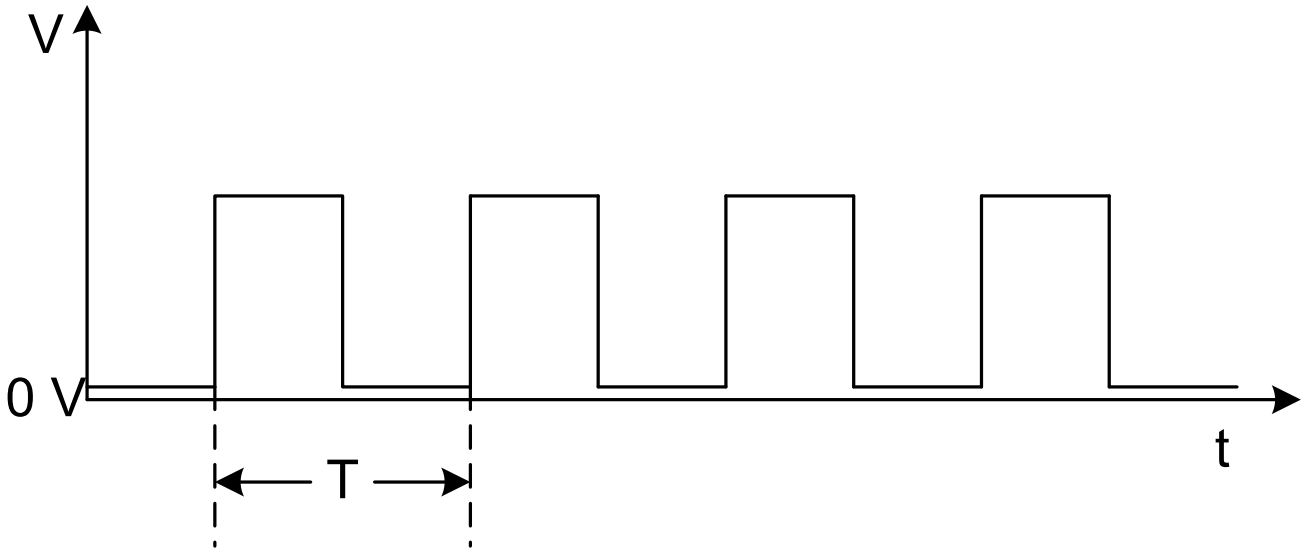
Σύνδεση Arduino με push button (tact switch) και LED



Πρόγραμμα που ανάβει και σβήνει το LED που είναι τοποθετημένο στην πόρτα 8 κάθε φορά που πιέζεται το push button που είναι τοποθετημένο στην πόρτα 2.

```
into LEDpin = 8;       // LED on pin 8
int switchPin = 2;   // momentary switch on 2, other side
                     // connected to ground (pull up)
boolean running = false;
void setup() {
   pinMode(LEDpin, OUTPUT);
   pinMode(switchPin, INPUT);
  }
void loop(){
if (digitalRead(switchPin)==LOW) // switch is pressed – pullup
                                 // keeps pin high normally
   {
     delay(100);                 // delay to debounce switch
     running = !running;         // toggle running variable
     digitalWrite(LEDpin, running);   // indicate via LED
   }
 }
```

Ωρολογιακοί παλμοί



f=1/T T: period f: frequency

Duty cycle

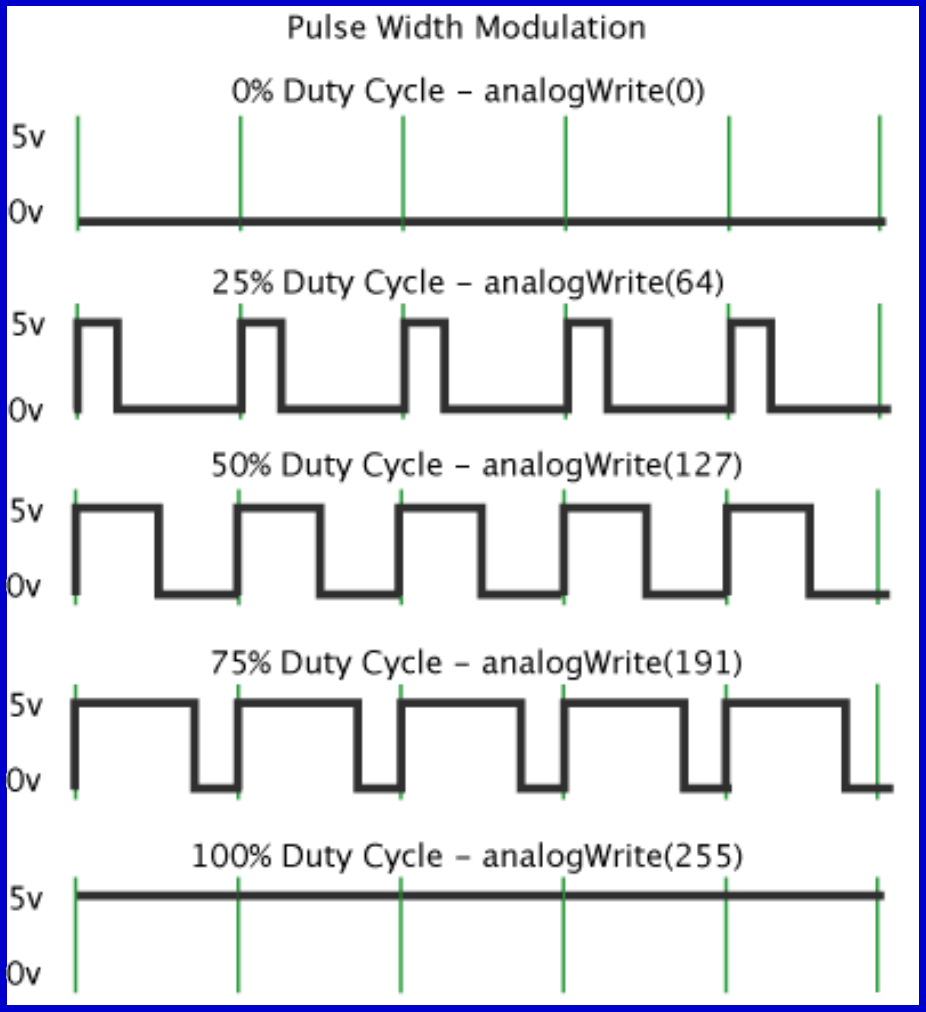
A *duty cycle* is the percentage of one period in which a signal is active. A period is the time it takes for a signal to complete an on-and-off cycle. As a formula, a duty cycle may be expressed as:

$$D = \frac{A}{T} \times 100\%$$

where D is the duty cycle, A is the time the signal is active, and T is the total period of the signal.

Κυματομορφές PWM με διαφορετικό duty cycle

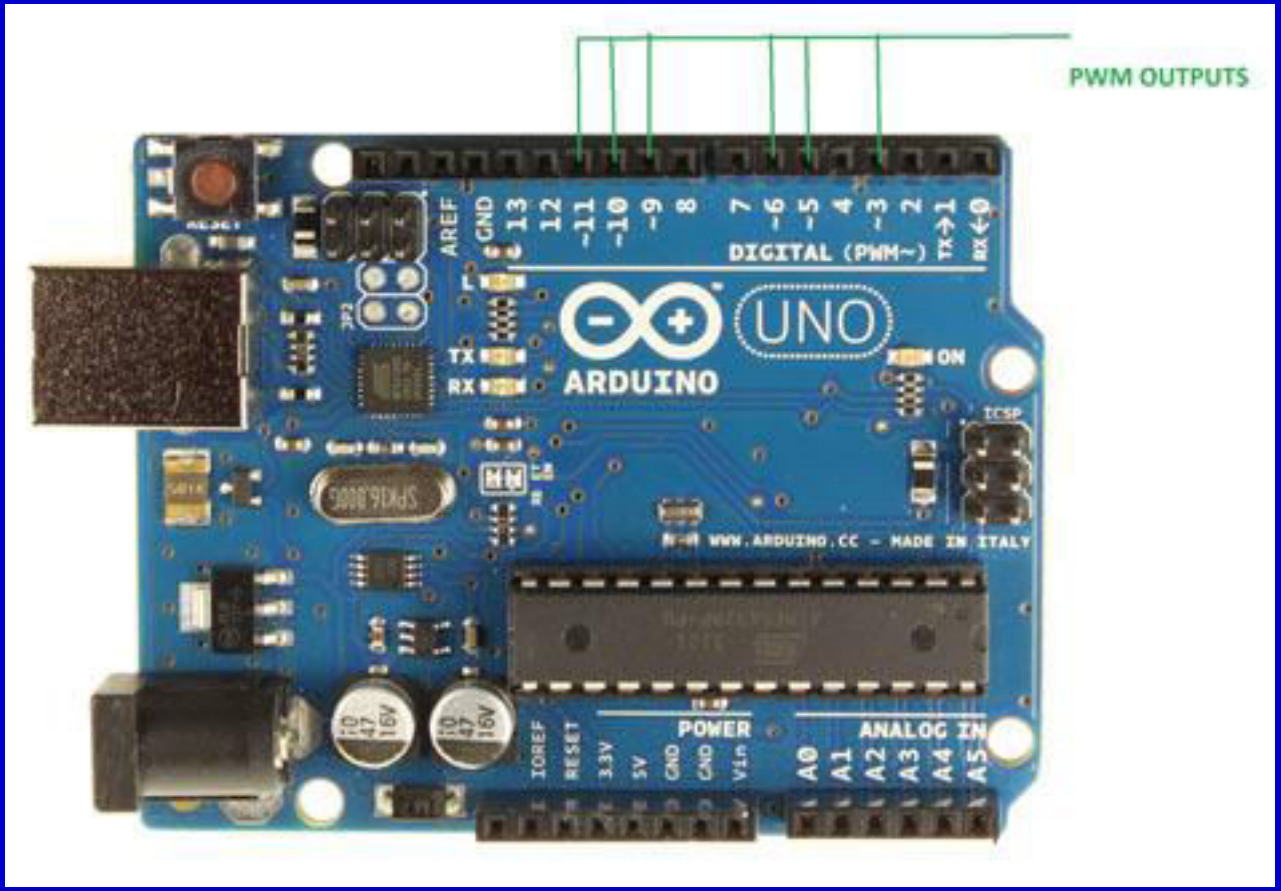
PWM : Pulse Width Modulation



AnalogWrite(pin, value)

AnalogWrite(*pin, value*) writes an analog value (PWM wave) to a pin. value: the duty cycle: between **0** (always off) and **255** (always on).
Can be used to light a LED at varying brightness or drive a motor at various speeds.
After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() **or** digitalWrite() on the same pin).
The frequency of the PWM signal on most pins is approximately **490 Hz**. On the Arduino Uno and similar boards, pins 5 and 6 have a frequency of approximately **980 Hz**.

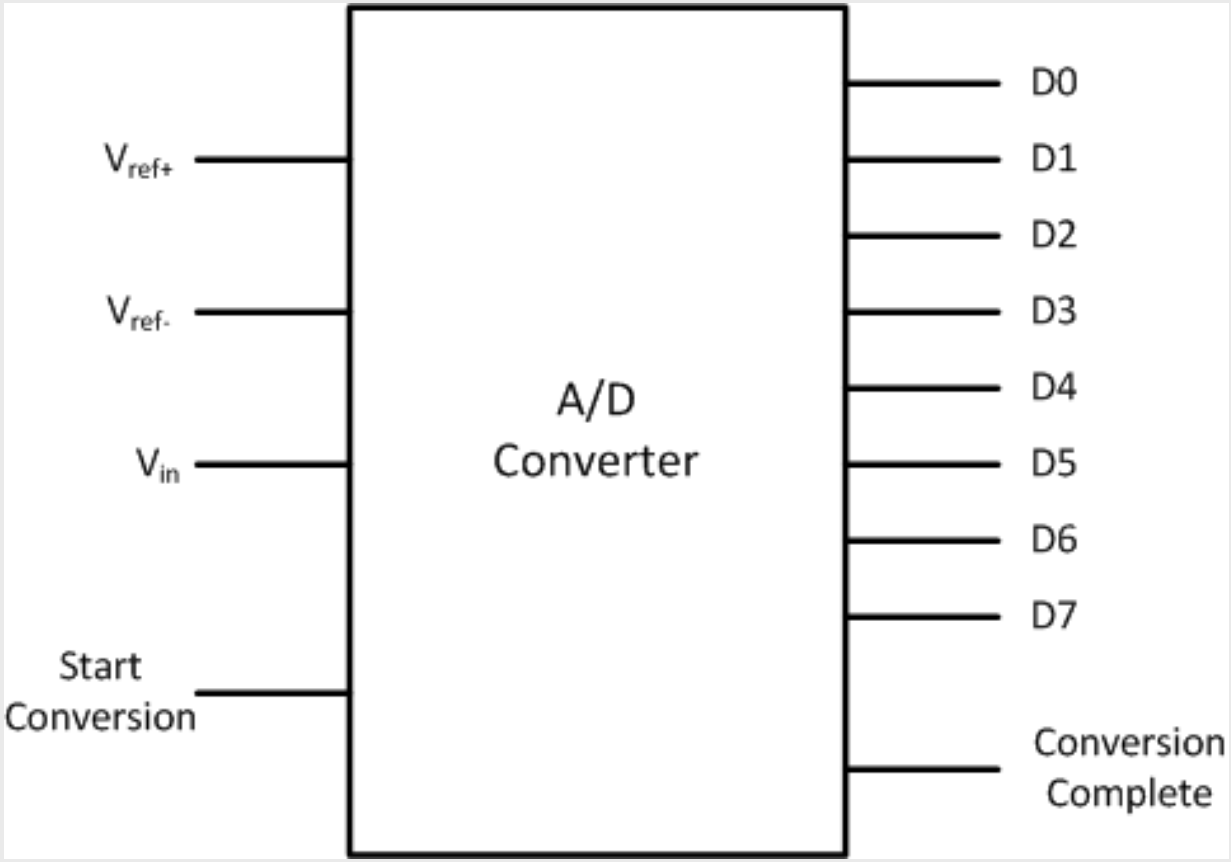
On Arduino Uno function *AnalogWrite*(*pin, value*) works on pins 3, 5, 6, 9, 10, and 11.



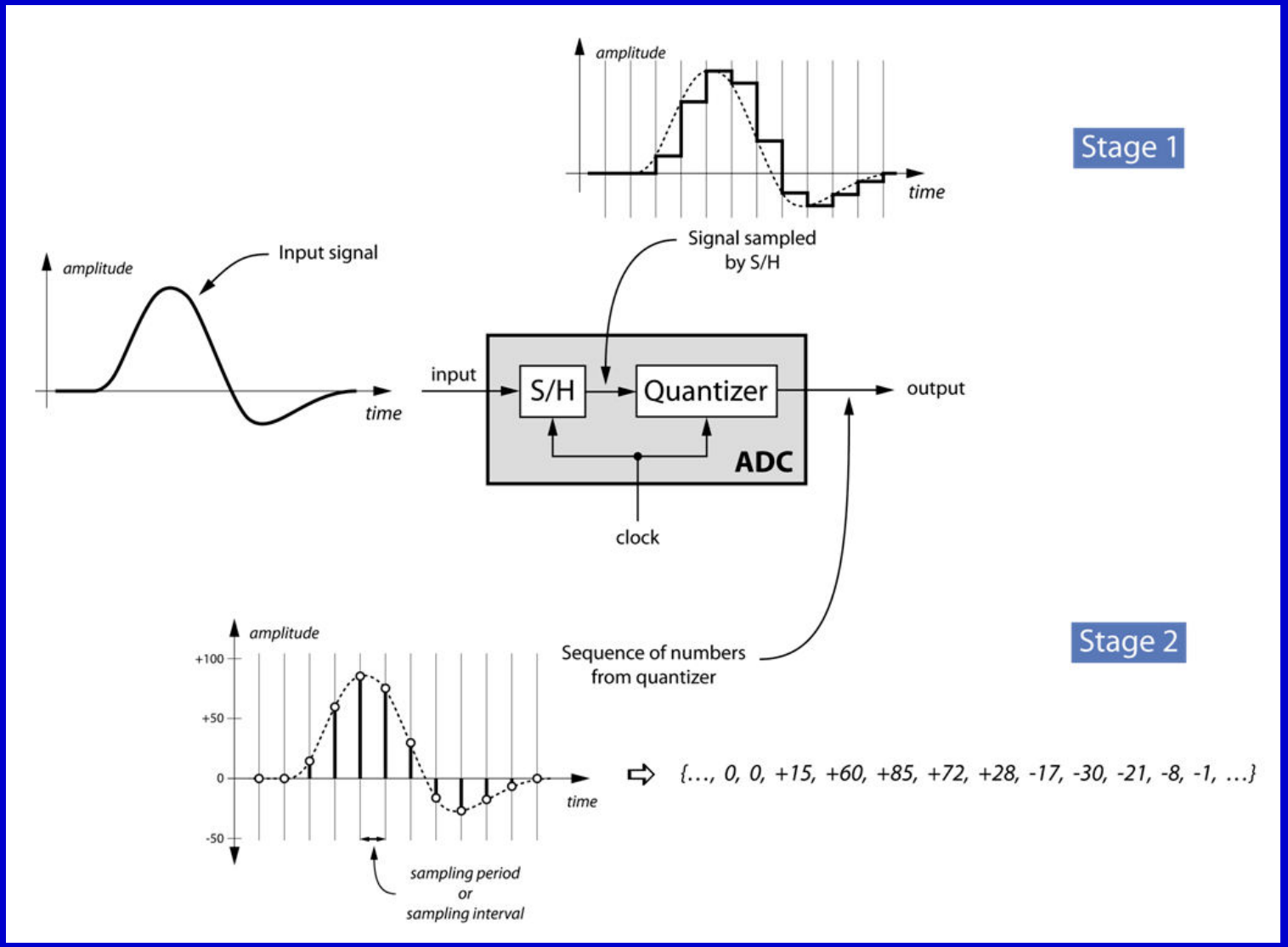
ADC

Οι *analog-to-digital converter* (ADC, A/D) είναι μονάδες που μετατρέπουν ένα συνεχές φυσικό μέγεθος (συνήθως τάση) σε ένα ψηφιακό αριθμό που παριστάνει το πλάτος του.
Η μετατροπή περιλαμβάνει κβαντισμό του σήματος εισόδου. Με αυτό τον τρόπο εισάγεται ένα μικρό σφάλμα.
Ο ADC αντί να κάνει μία απλή μετατροπή συνήθως κάνει την μετατροπή ("samples" the input) περιοδικά. Το αποτέλεσμα είναι να μετατρέπεται ένα αναλογικό σήμα το οποίο είναι συνεχές στον χρόνο σε ένα σήμα το οποίο είναι διακριτό στον χρόνο.

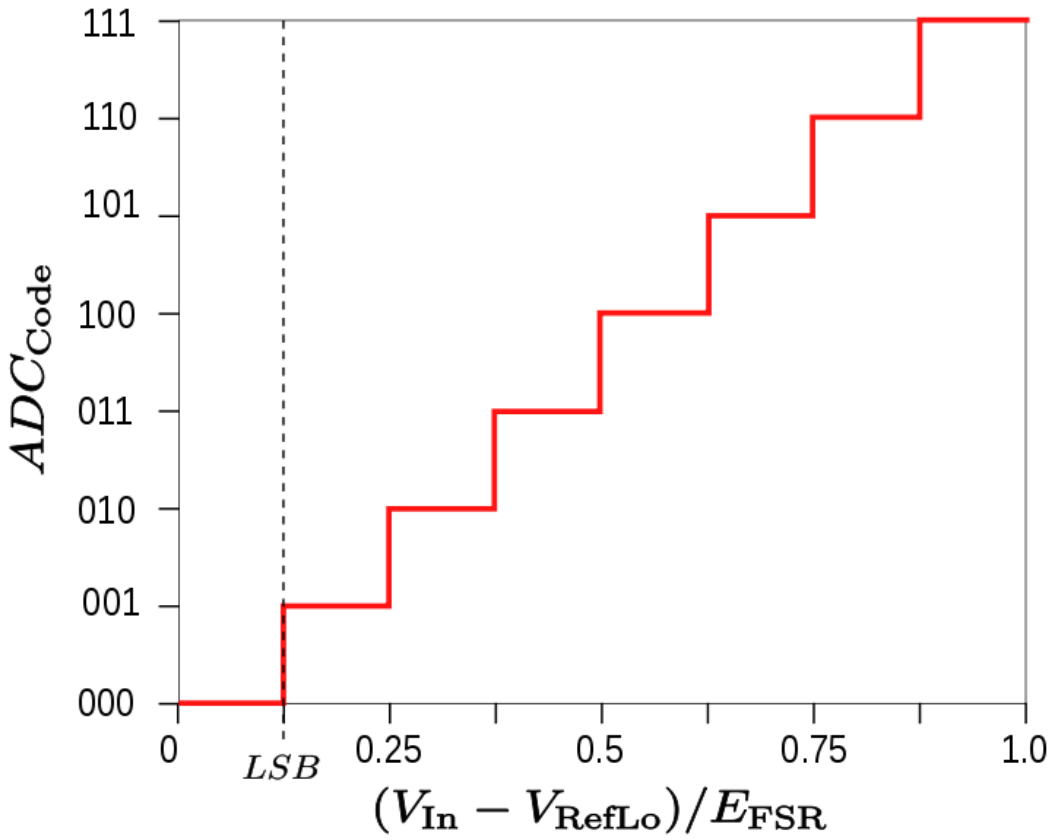
ADC


V_{ref+}
V_{ref-}
V_{in}
Start Conversion
A/D Converter
D0
D1
D2
D3
D4
D5
D6
D7
Conversion Complete

Analog to digital conversion



3-bit analog to digital conversion



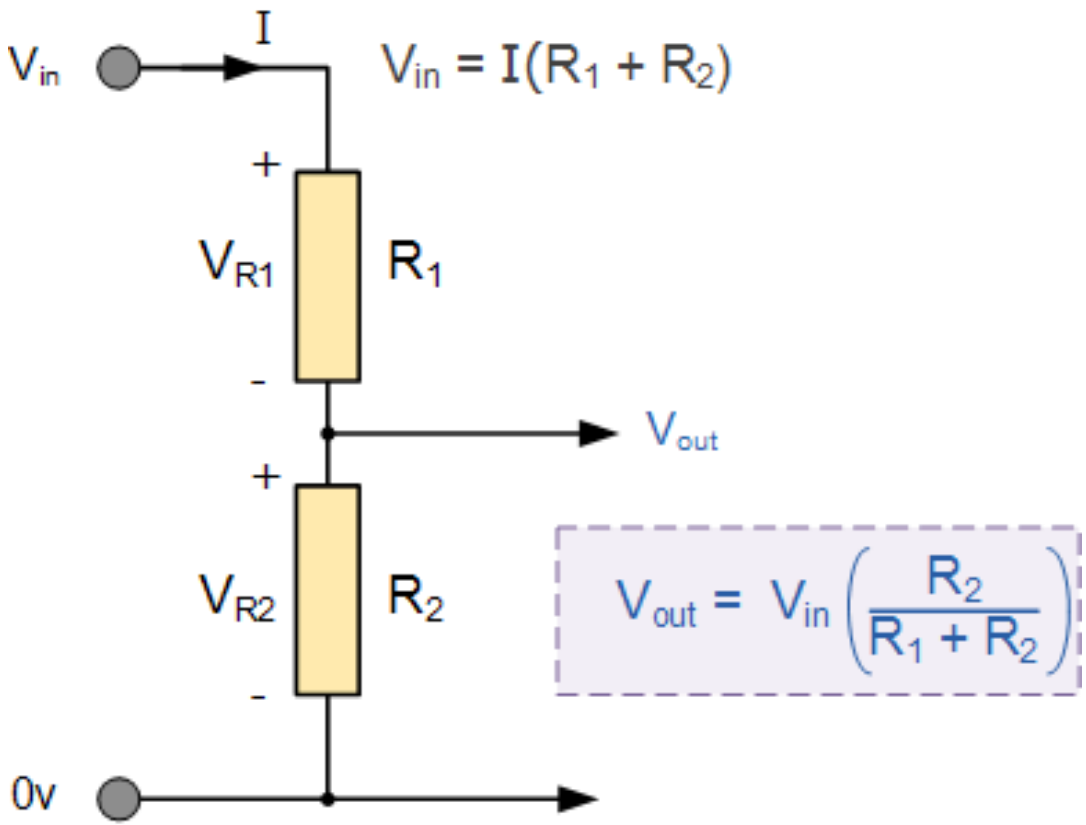
$$E_{\mathrm{FSR}} = V_{\mathrm{RefHi}} - V_{\mathrm{RefLow}},$$

Αναλογικές είσοδοι του Arduino

analogRead()

AnalogRead() reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter.
This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023 (2^{10}-1). This yields a resolution between readings of: 5 volts/1024 units or, 0.0049 volts (4.9 mV) per unit.
It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

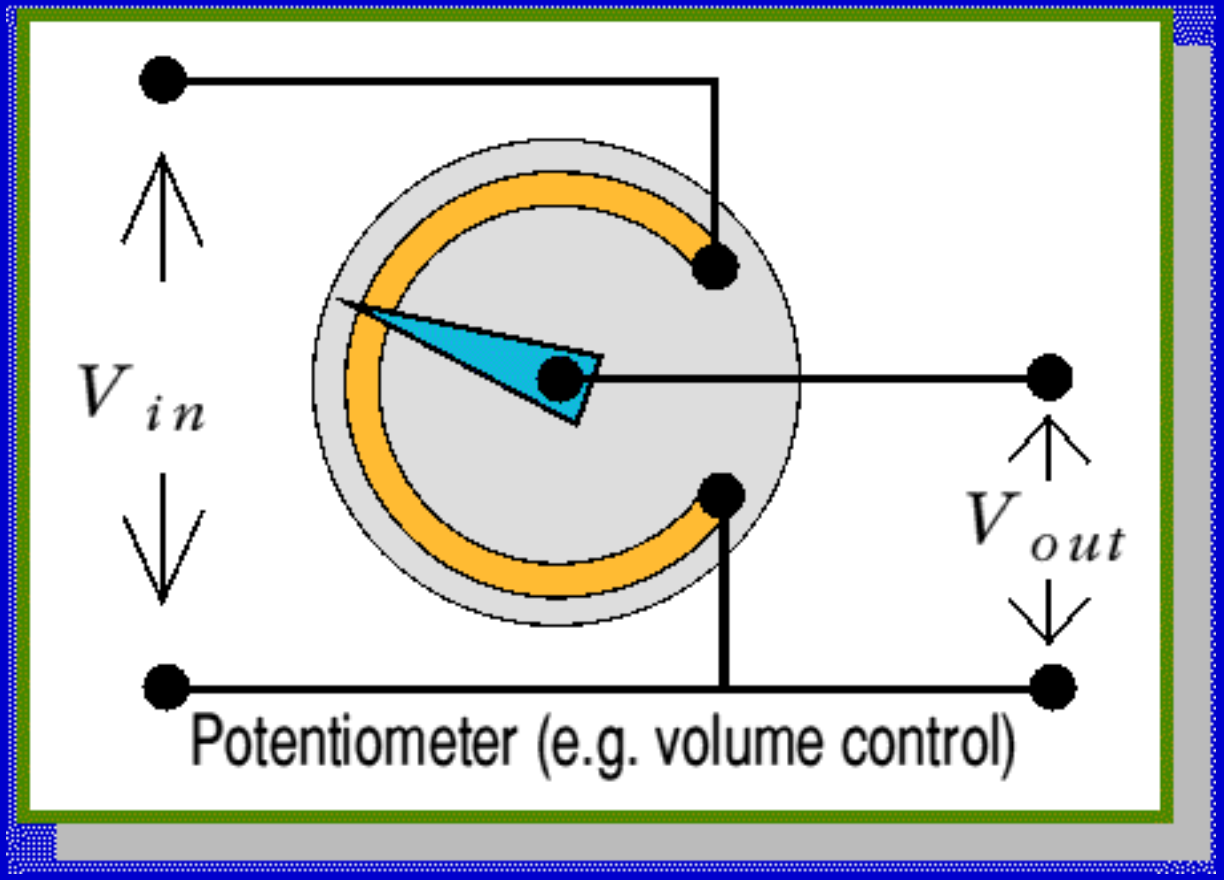
Διαιρέτης τάσης



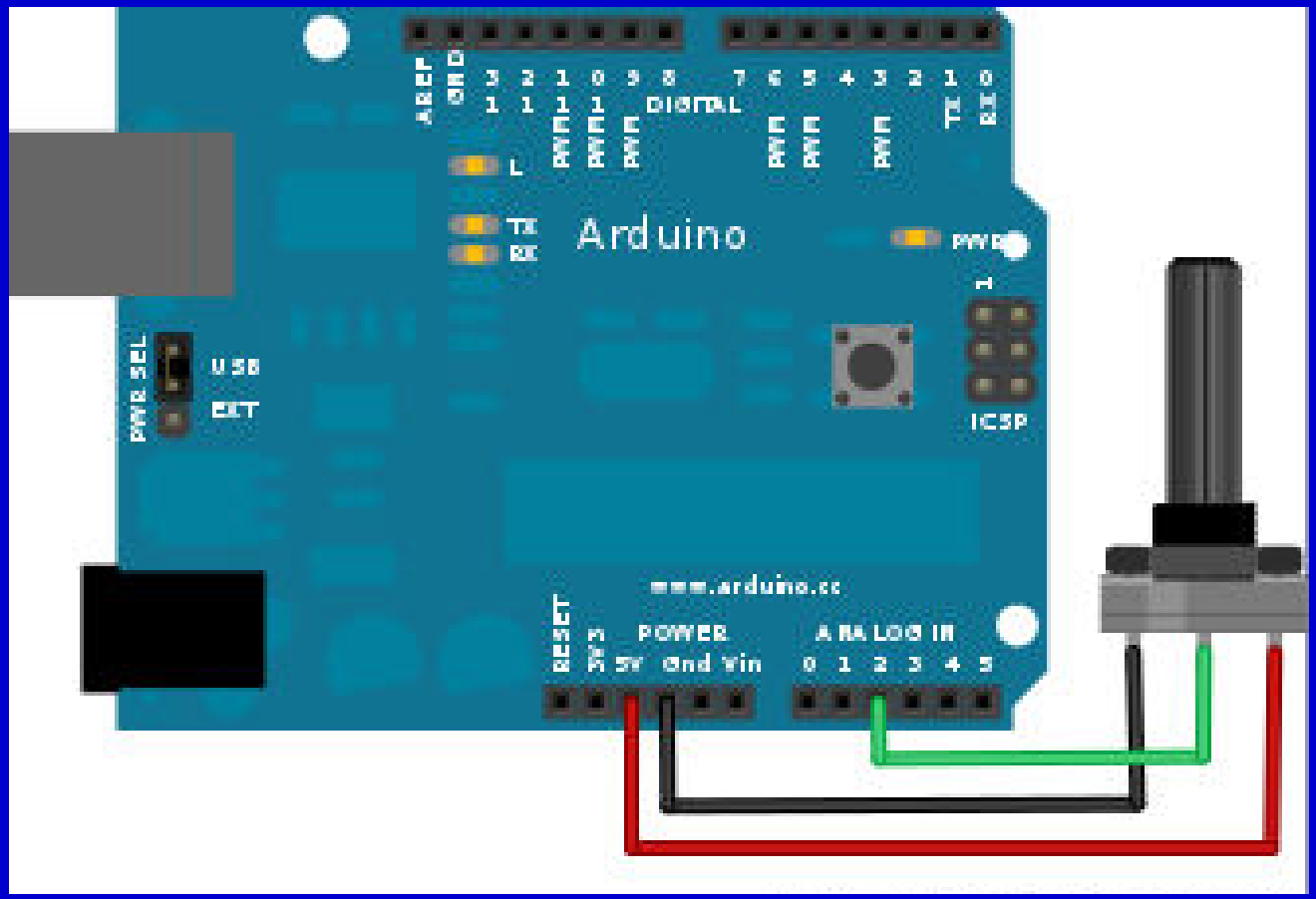
Ποτενσιόμετρα

Το **ποτενσιόμετρο** είναι αναλογικό ηλεκτρονικό εξάρτημα, που χρησιμοποιείται στα κυκλώματα ως μεταβλητή αντίσταση. Αποτελείται από αγώγιμη πλάκα σχήματος Ω, πάνω στην οποία γυρίζει, με τη βοήθεια ενός στροφέα, μια επαφή. Ανάλογα με την απόσταση της επαφής από την είσοδο του ρεύματος στο ποτενσιόμετρο μεταβάλλεται και η αντίσταση.

Παραγωγή μεταβλητής τάσης με ποτενσιόμετρο



Σύνδεση Arduino με ποτενσιόμετρο

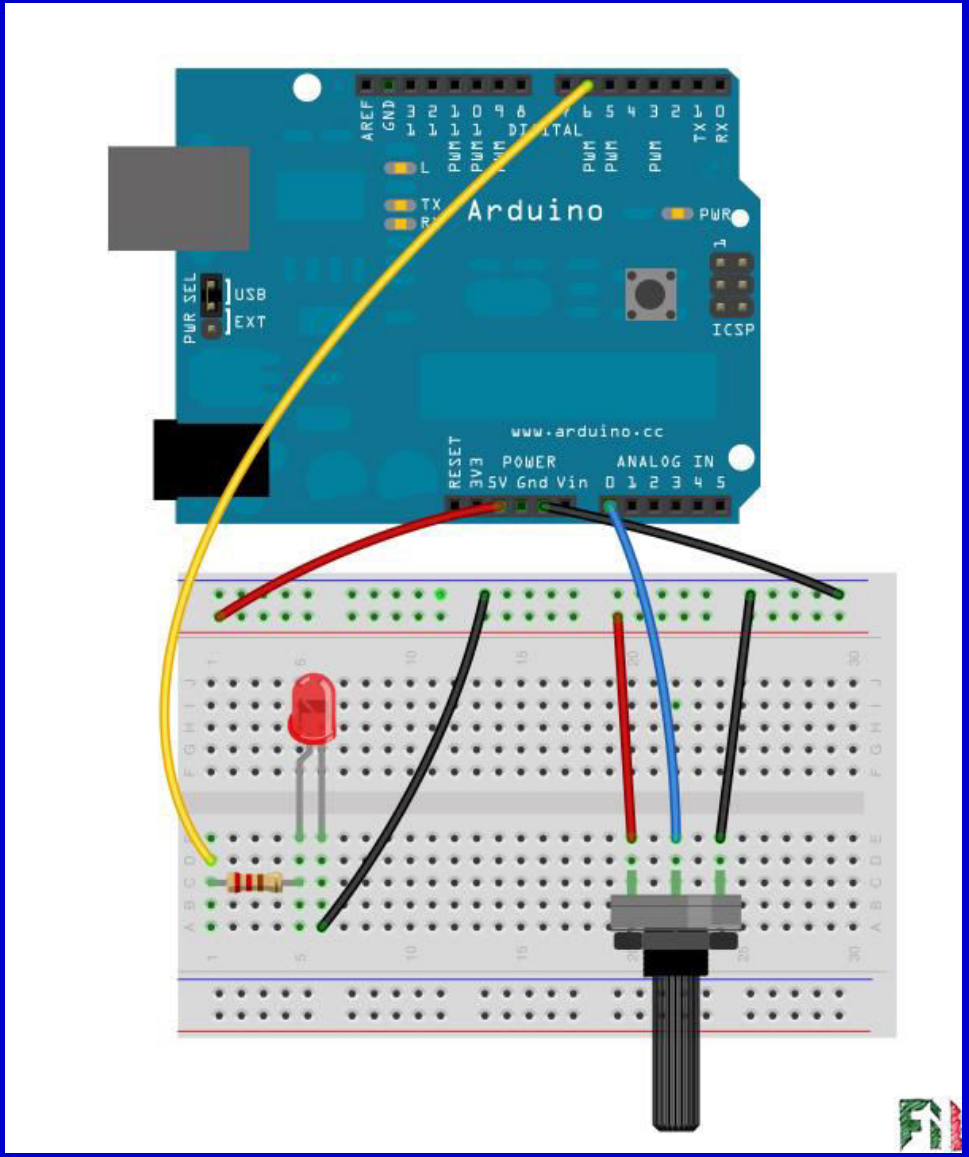


Πρόγραμμα που διαβάζει την τιμή τάσης που εφαρμόζεται στην αναλογική είσοδο και στέλνει την αντίστοιχη ψηφιακή τιμή στην σειριακή έξοδο.

```
int analogPin = 2;    // potentiometer wiper (middle terminal)
                      // connected to analog pin 2
                      // outside leads to ground and +5V
int val = 0;          // variable to store the value read

void setup() {
  Serial.begin(9600);         //  setup serial
 }

void loop(){
  val = analogRead(analogPin);    // read the input pin
  Serial.println(val);            // debug value
 }
```

```
//Reads an analog input on analog in 2, prints the value out.

int analogValue;    // variable to hold the analog value

void setup() {
   Serial.begin(9600); // open the serial port at 9600 bps
}
void loop() {
    analogValue = analogRead(2); // read the analog input on pin 0
  // print it out in many formats:
  Serial.println(analogValue);           // print as an ASCII-encoded decimal
  Serial.println(analogValue, DEC);  // print as an ASCII-encoded decimal
  Serial.println(analogValue, HEX);  // print as an ASCII-encoded hexadecimal
  Serial.println(analogValue, OCT);  // print as an ASCII-encoded octal
  Serial.println(analogValue, BIN);   // print as an ASCII-encoded binary
  delay(10);
}
```

Κύκλωμα παραγωγής μεταβλητής τάσης με ποτενσιόμετρο η οποία εισάγεται σε αναλογική είσοδο και του Arduino για να ρυθμίζεται ανάλογα η φωτοβολία του LED.



Πρόγραμμα που διαβάζει την τιμή τάσης που εφαρμόζεται σε αναλογική είσοδο και ρυθμίζει ανάλογα την φωτεινότητα LED που είναι τοποθετημένο σε ψηφιακή πόρτα.

```
int ledPin = 9;        // LED connected to digital pin 9
int analogPin = 3;  // potentiometer connected to analog
                       //pin 3
int val = 0;           // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);   // sets the pin as output
}

void loop() {
  val = analogRead(analogPin); // read the input pin
  analogWrite(ledPin, val/4);
  //analogRead values from 0 to 1023,
  //analogWrite values from 0 to 255
}
```

map(value, fromLow, fromHigh, toLow, toHigh)

map remaps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh wold get maped to toHigh, values in-between to values in-between, etc.

```
/* Map an analog value to 8 bits (0 to 255) */
void setup() {
    pinMode(9, OUTPUT);
 }
void loop() {
  int val = analogRead(3);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
}
```

pulseIn(pin, value)
pulseIn(pin, value, timeout)

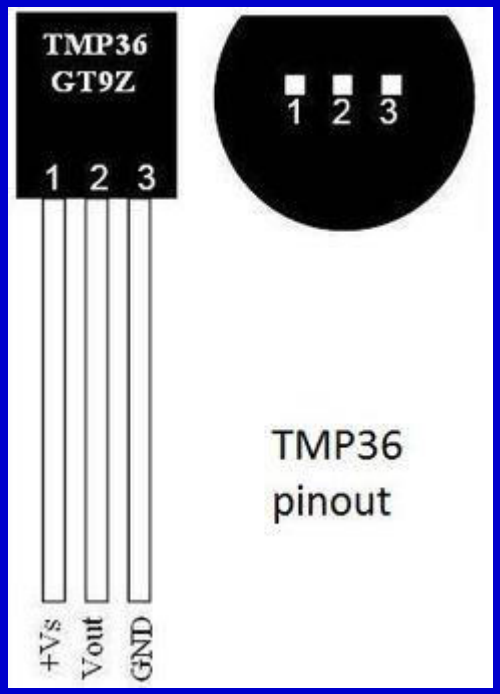
pulseIn reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or 0 if no complete pulse was received within the timeout.

pin: the number of the pin on which you want to read the pulse.
value: type of pulse to read: either HIGH or LOW. (*int*)
timeout (optional): the number of microseconds to wait for the pulse to be completed: the function returns 0 if no complete pulse was received within the timeout.
Default is one second (*unsigned long*).

Σ

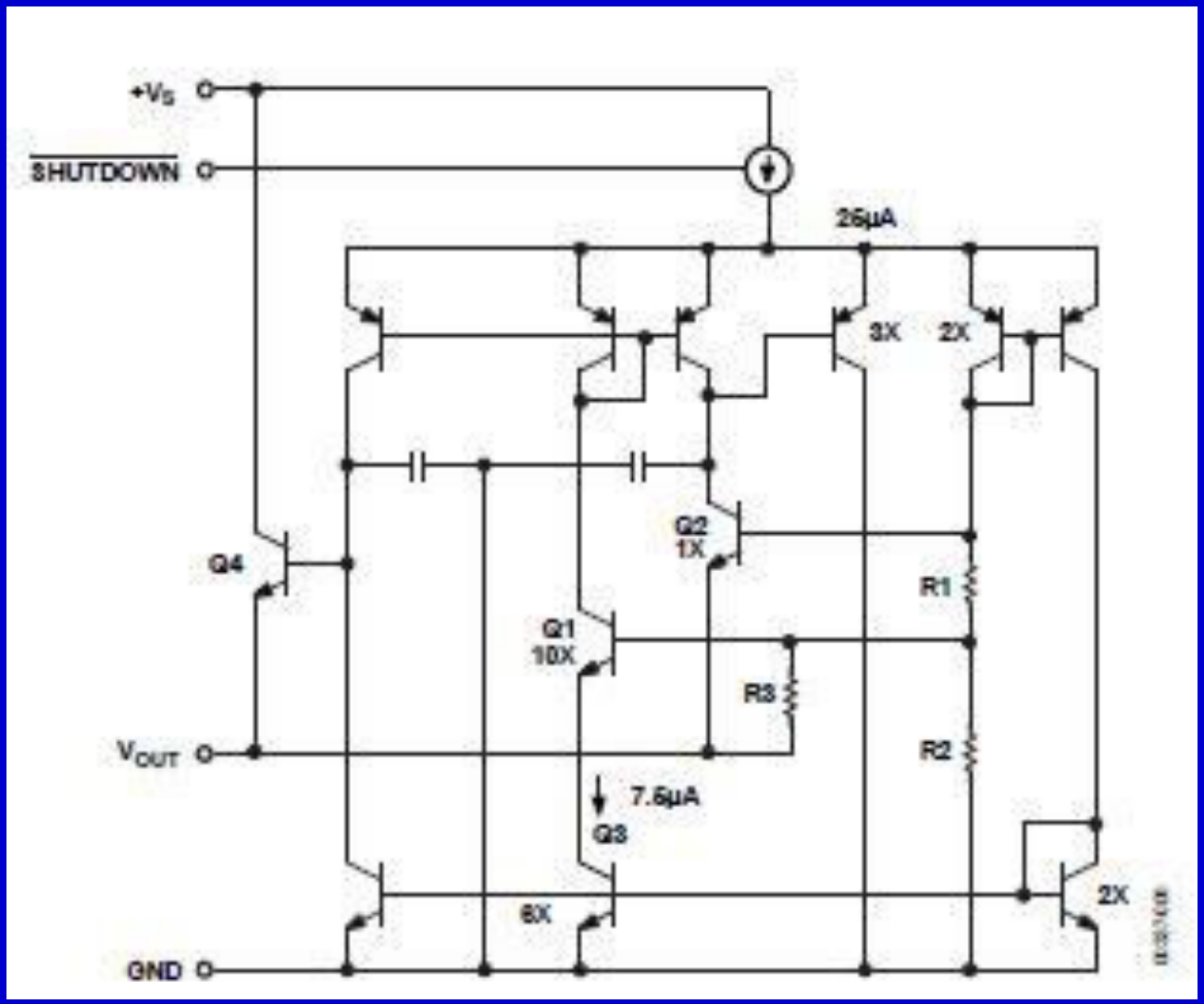
```
int pin = 7;

unsigned long duration;

void setup()
{
  pinMode(pin, INPUT);
}

void loop()
{
  duration = pulseIn(pin, HIGH);
}
```

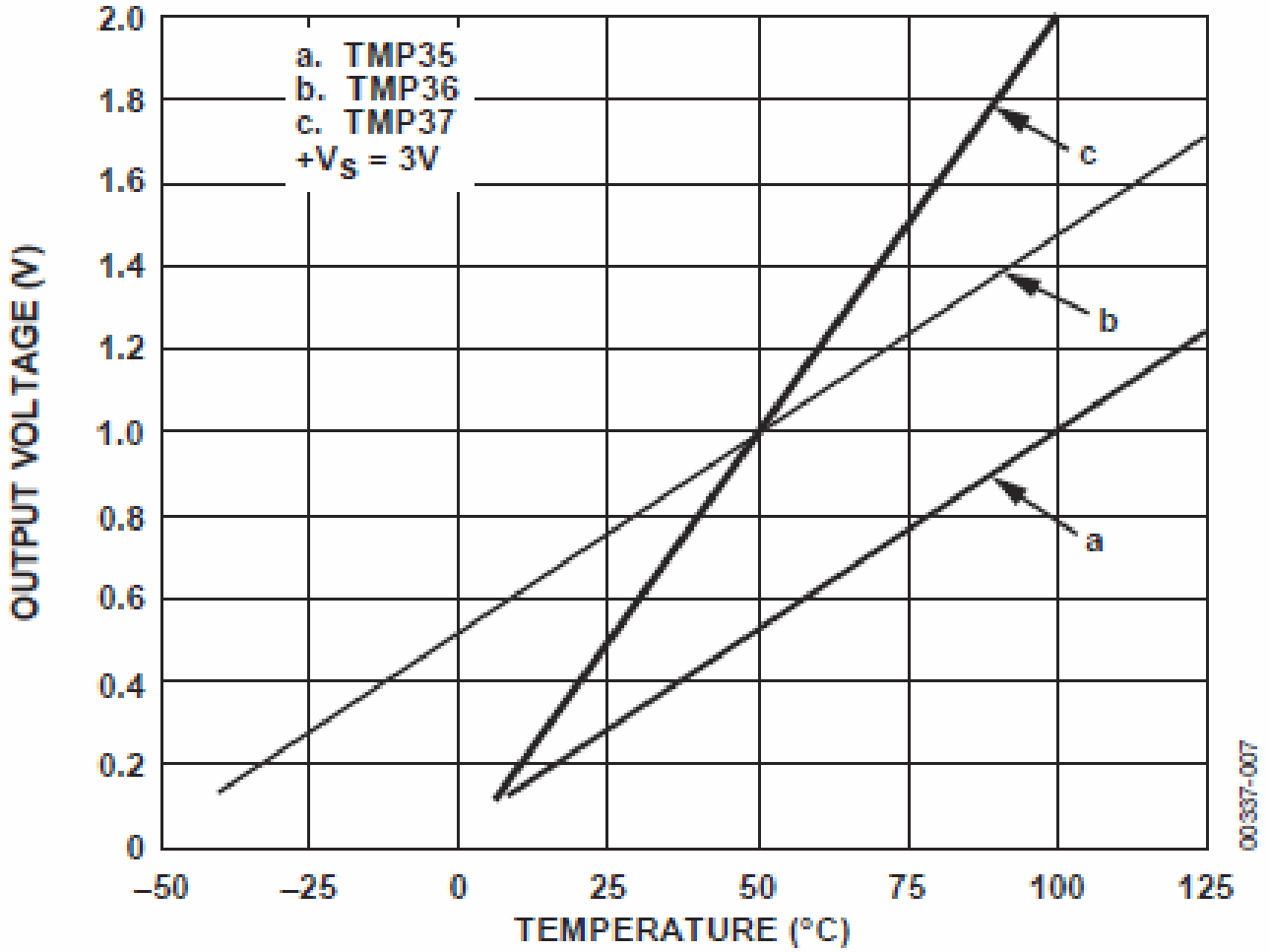
Αισθητήρας θερμοκρασίας TMP36



Low Voltage Operation (+2.7 V to +5.5 V)
Calibrated Directly in °C
10 mV/8°C Scale Factor (20 mV/8°C on TMP37)
±2°C Accuracy OverTemperature (typ)
±0.5°C Linearity (typ)
Stable with Large Capacitive Loads
Specified -40 °C to +125 °C, Operation to +150 °C
Less than 50 μA Quiescent Current

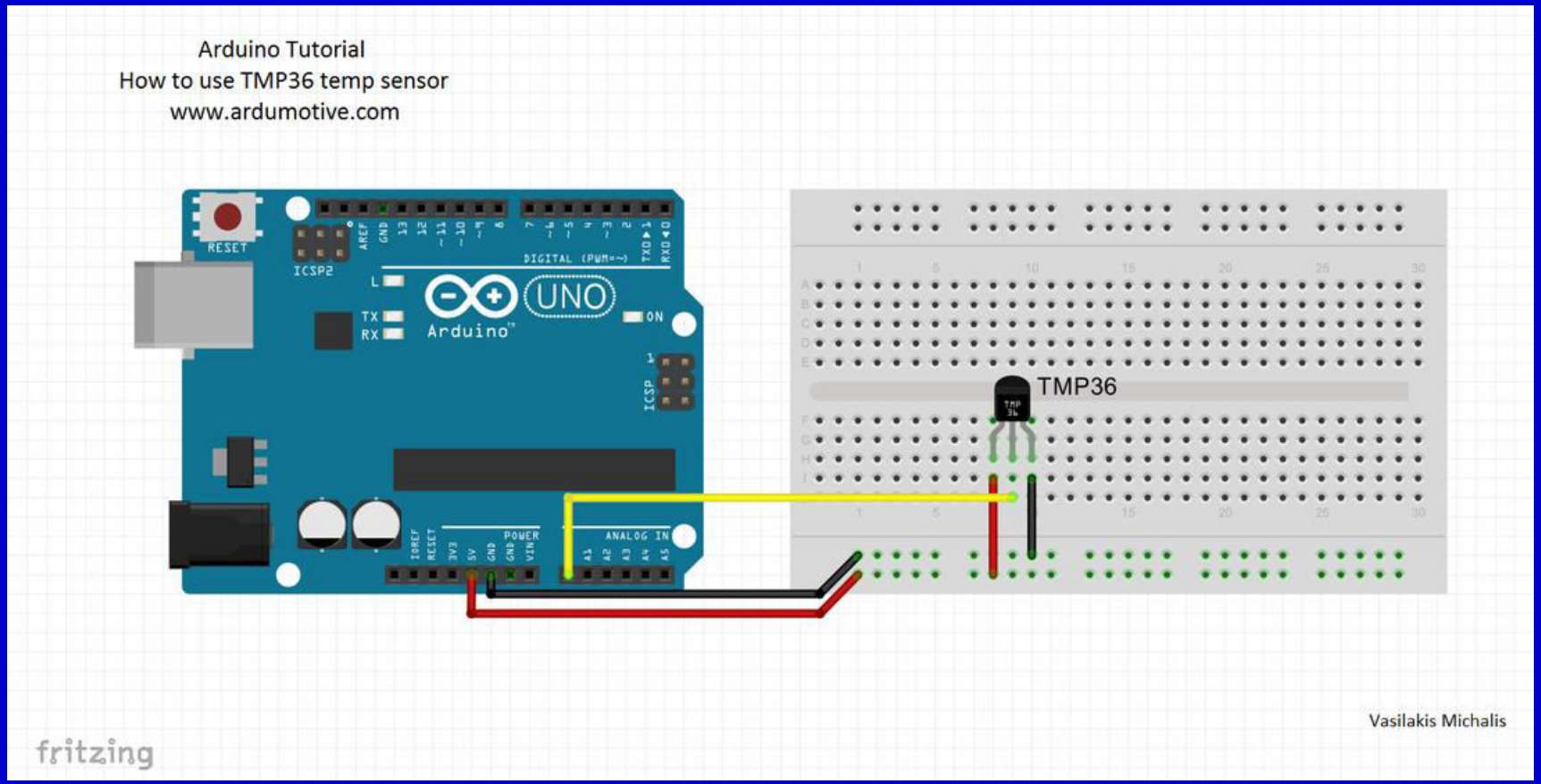
Δομή του αισθητήρα TMP36





Output Voltage vs. Temperature

Σύνδεση του Arduino με τον αισθητήρα θερμοκρασίας TMP36



Πρόγραμμα το οποίο στέλνει στην σειριακή θύρα του Arduino την θερμοκρασία που μετράει ο αισθητήρας TMP36

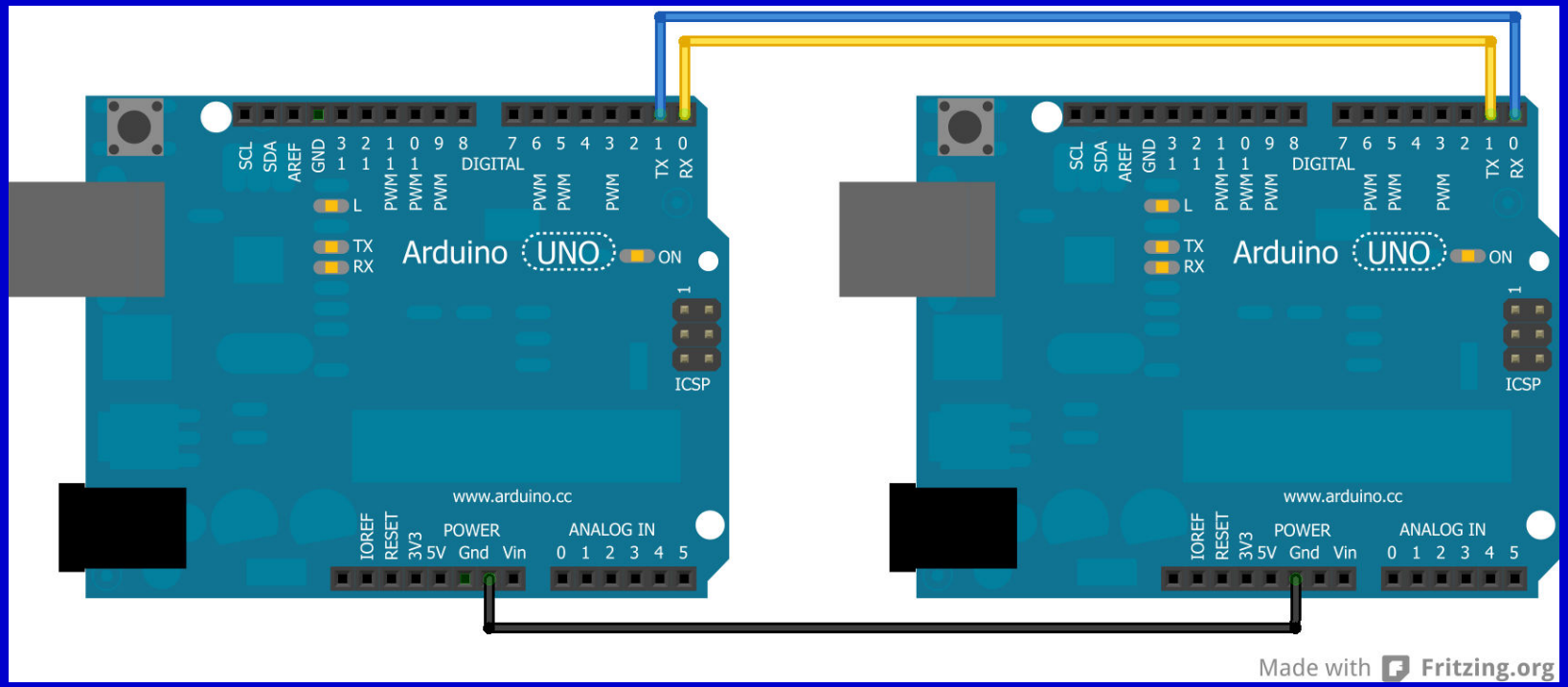
```
// Analog input 0 is used to measure the temperature  sensor's
//signal pin.

const int temperaturePin = 0;
float voltage, degreesC, degreesF;

void setup()
{
   Serial.begin(9600);
}
```

```
void loop() {
//function call
   voltage = getVoltage(temperaturePin);
//
  degreesC = (voltage - 0.5)*100.0;
  degreesF = degreesC*(9.0/5.0) + 32.0;

  Serial.print("voltage: ");
  Serial.print(voltage);
  Serial.print(“ deg C: ");
  Serial.print(degreesC);
  Serial.print(" deg F: ");
  Serial.println(degreesF);
  delay(1000);
}
```

```
//Function

float getVoltage(int pin)
{
  return (analogRead(pin) * 0.004882814);

  // This equation converts the 0 to 1023 value that analogRead()
  // returns, into a 0.0 to 5.0 value that is the true voltage
  // being read at that pin.
}
```

Σύνδεση δύο Arduino μέσω σειριακών πορτών και προγραμματισμός τους ώστε το ένα να αναβοσβήνει λαμπάκι στο άλλο

Σύνδεση Arduino με Arduino μέσω των σειριακών πορτών



Πρόγραμμα που ανάβει ή σβήνει το LED που είναι συνδεδεμένο στην πόρτα 13 ανάλογα αν στην σειριακή πόρτα έλθει Η ή L.

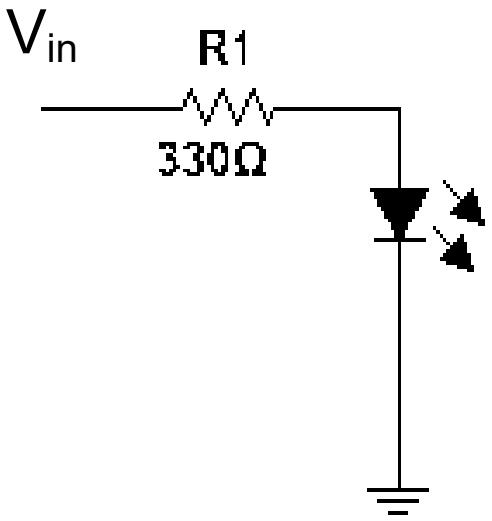
```
const int ledPin = 13; // the pin that the LED is attached to
int incomingByte;      // a variable to read incoming serial data into

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

```
void loop() {
  // see if there's incoming serial data:
  if (Serial.available() > 0)
  {
     // read the oldest byte in the serial buffer:
     incomingByte = Serial.read();
     // if it's a capital H (ASCII 72), turn on the LED:
      if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
      }
      // if it's an L (ASCII 76) turn off the LED:
      if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
      }
   }
 }
```

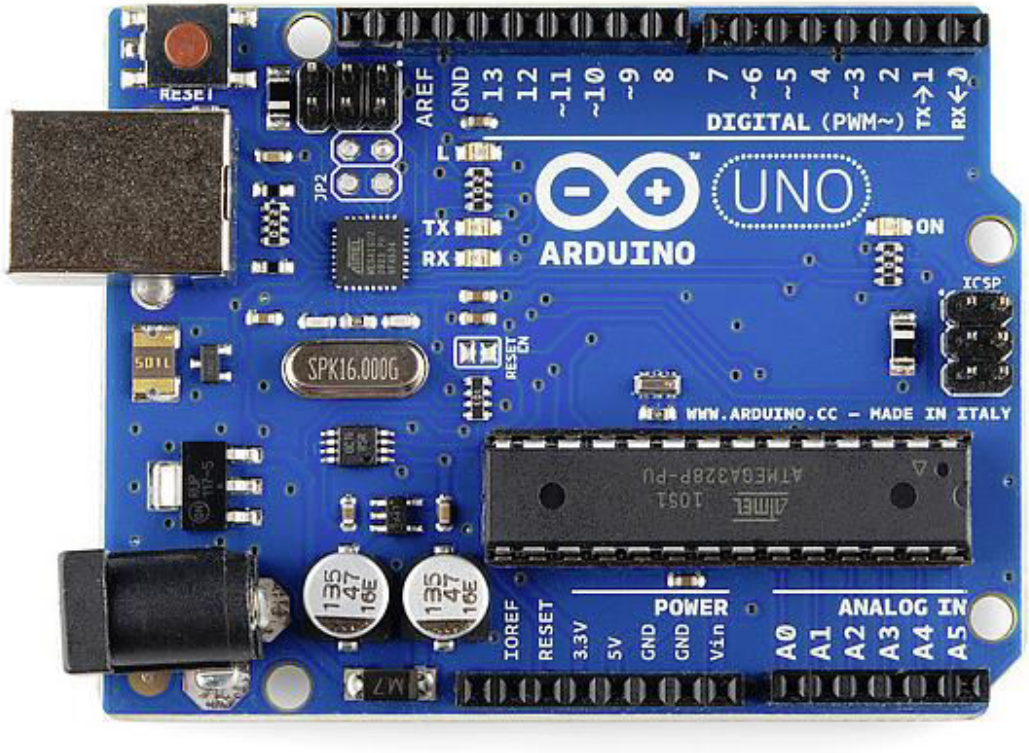
Για να αναβοσβήνει το ένα Arduino το λαμπάκι 13 στο άλλο σε αυτό με το λαμπάκι τοποθετείται το προηγούμενο πρόγραμμα και στο άλλο το επόμενο.

```
void setup() {
Serial.begin(9600);
}

void loop() {
  Serial.print('H');
  delay(1000);
  Serial.print('L');
  delay(1000);
}
```

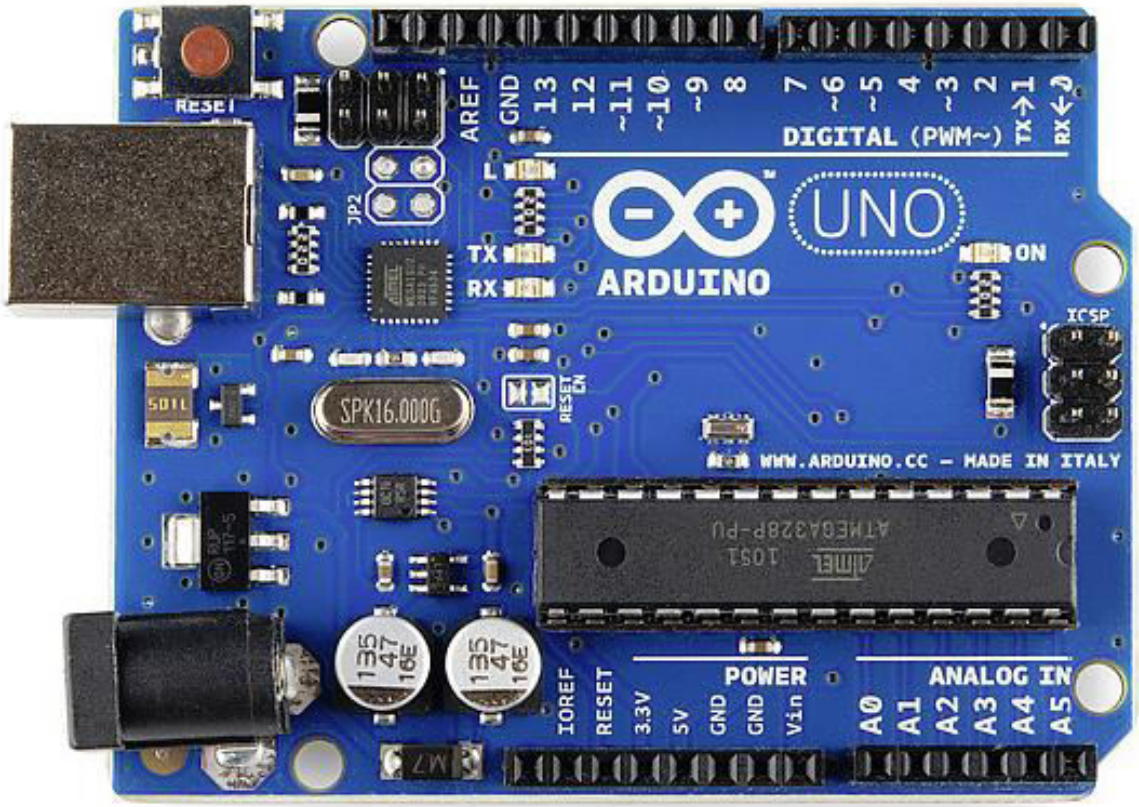
5.1. Υπολογίστε το ρεύμα που διαρρέει το LED στο κύκλωμα που δίδεται στην συνέχεια για Vin=5V και για Vin=3.3 Volt. Θεωρήστε ότι το voltage drop του LED είναι Vd=2 Volt



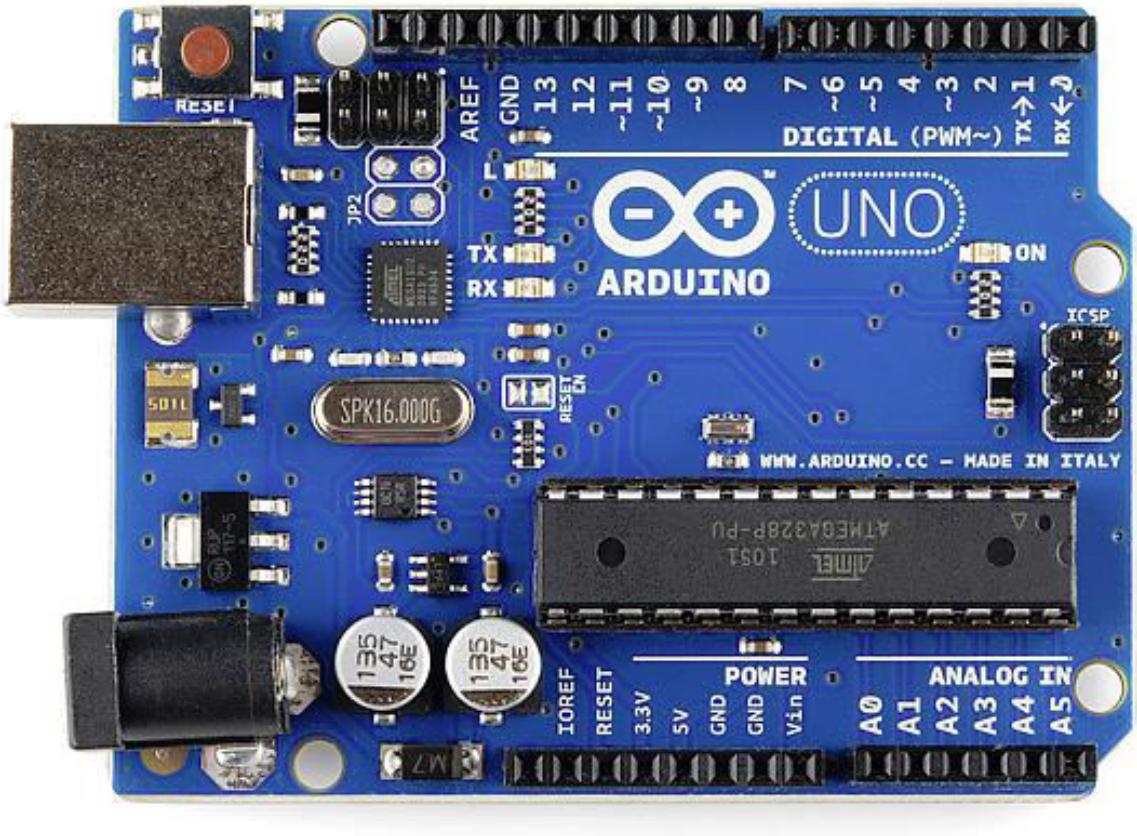
5.2. Συνδέστε το LED που δίδεται στην συνέχεια σε μία πόρτα του Arduino ώστε να δίδει ένδειξη των 0 και 1.



5.3. Συνδέστε το DIP switch και την αντίσταση που δίδονται στην συνέχεια ώστε να υλοποιείται μια είσοδος των 0, 1.



5.4. Συνδέστε το takt switch που δίδεται στην συνέχεια ώστε να χρησιμοποιείται σαν είσοδος των 0, 1.



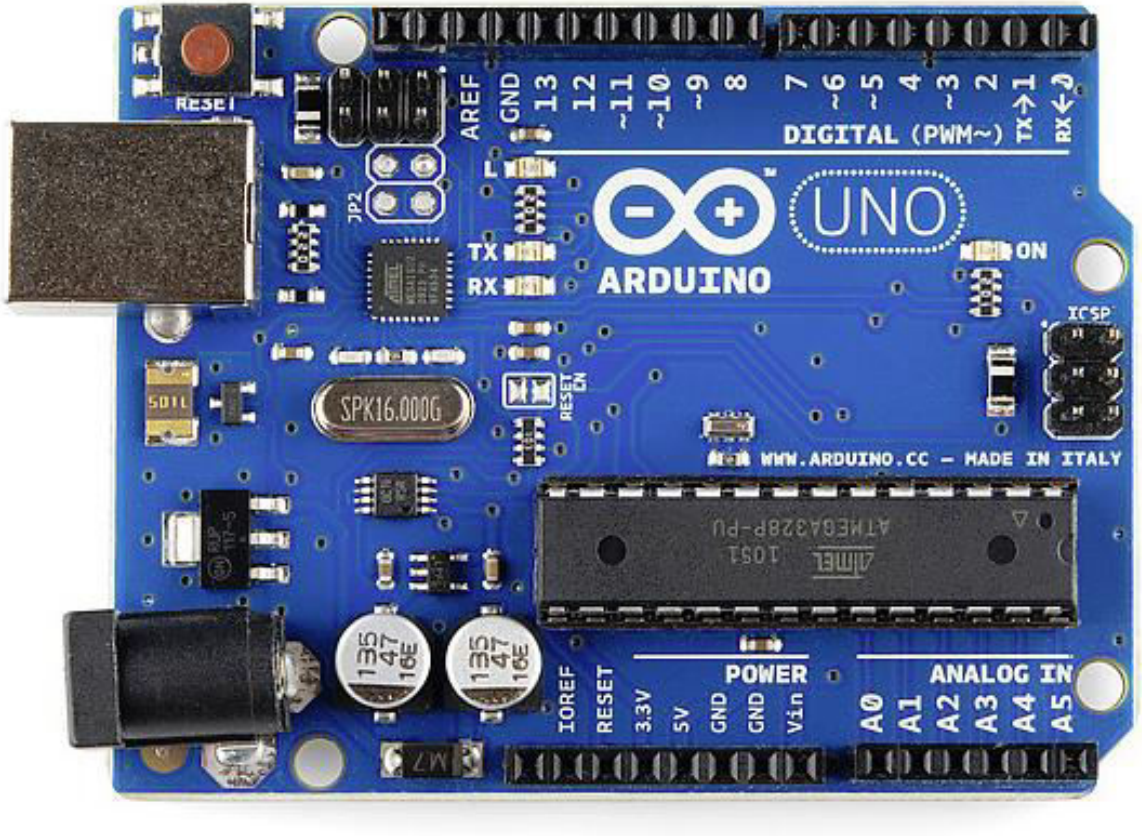
5.5. Χρησιμοποιώντας αντιστάσεις του 1 KOhm σχεδιάστε κύκλωμα που να παράγει τάση 2.5 Volt από πηγή τάσης των 5 Volt.

5.6. Χρησιμοποιώντας αντιστάσεις των 2 KOhm υλοποιείστε μία αντίσταση 1 KOhm.

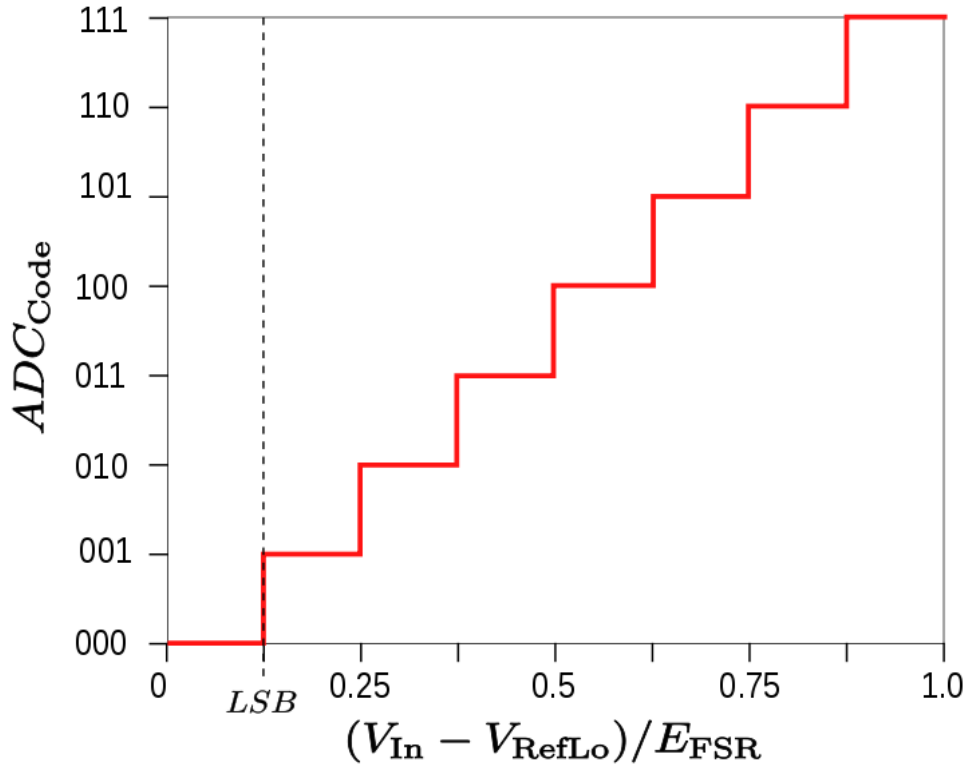
5.7. Στο Arduino που δίδεται στην συνέχεια σημειώστε τις αναλογικές πόρτες.

5.8. Στο Arduino που δίδεται στην συνέχεια σημειώστε τις πόρτες με δυνατότητα PWM.

5.9. Συνδέστε το ποτενσιόμετρο που δίδεται στην συνέχεια στε να χρησιμοποιείται σαν είσοδος αναλογικής τάσης.

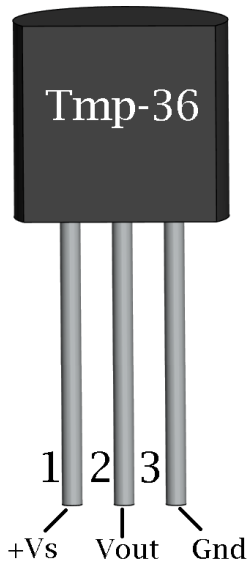


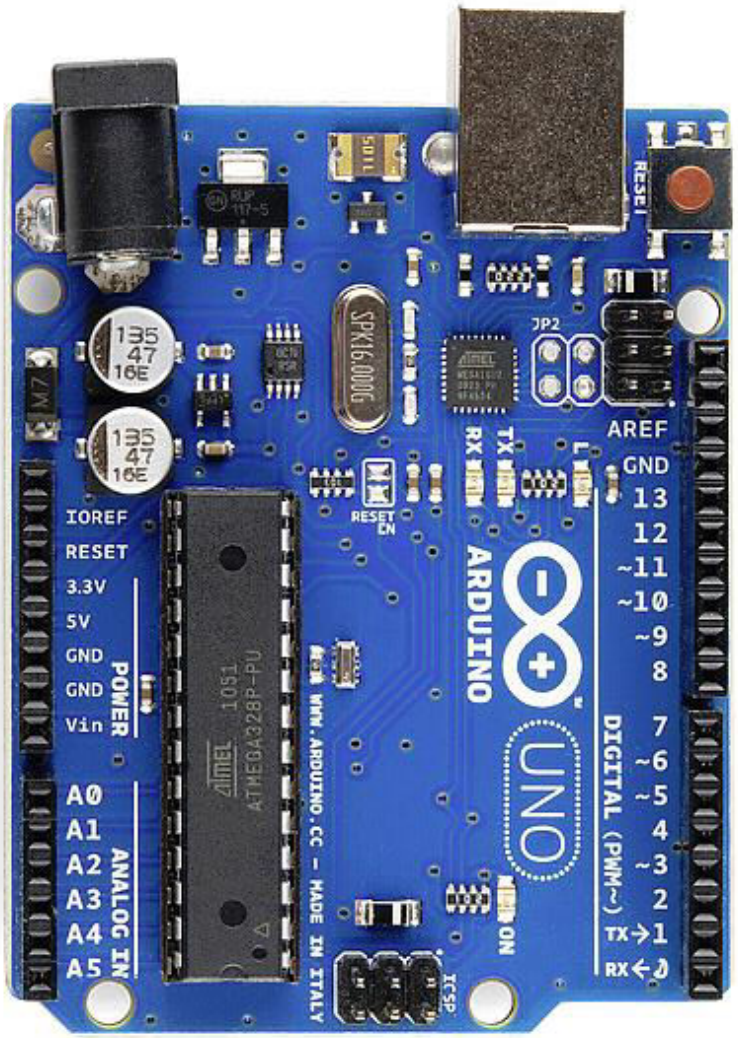
5.10. Σε έναν ADC με έξοδο των 3 bit όπως αυτή που περιγράφεται στην συνέχεια, V_{RefHi}=5V, V_{RefLow}=0V. Έστω V_{in}=3V. Να υπολογισθεί η έξοδός του.



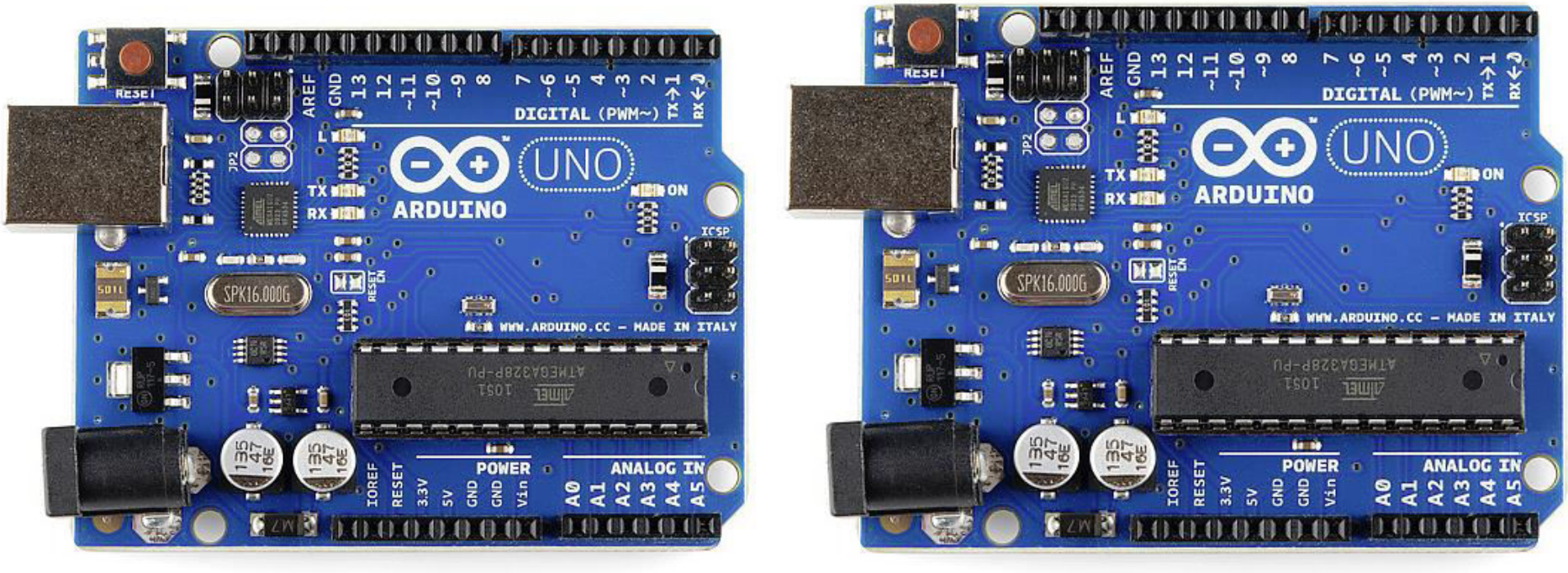
$$E_{\mathrm{FSR}} = V_{\mathrm{RefHi}} - V_{\mathrm{RefLow}},$$

5.11. Συνδέστε το αισθητήρα θερμοκρασίας TMP36 στο Arduino.





5.12. Συνδέστε τα δύο Arduino ώστε το ένα να μπορεί να στέλνει σειριακά δεδομένα στο άλλο.



5.13 Περιγράψτε το αποτέλεσμα της εκτέλεσης της εντολής digitalWrite(4, HIGH)

5.14 Περιγράψτε το αποτέλεσμα της εκτέλεσης της εντολής analogWrite(3, 127)