Μικροελεγκτές Arduino

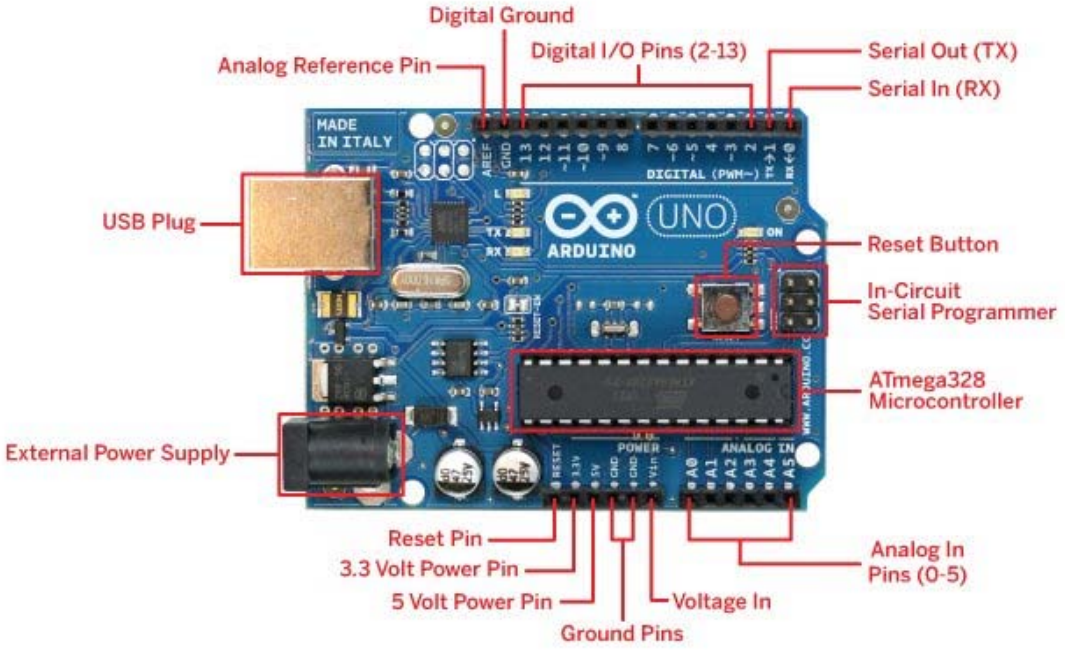
Ι. Έλληνας

Σεπτέμβριος 2016

Αναφορές

1. Παναγιώτης Παπάζογλου, Σπύρος Λιωνής, “Ανάπτυξη εφαρμογών με το Arduino”, Εκδόσεις Τζιόλα, 2016
2. Michael Margolis, “Arduino Cookbook”, O’Reilly, 2012
3. Arduino cheatsheet

Με μια ματιά

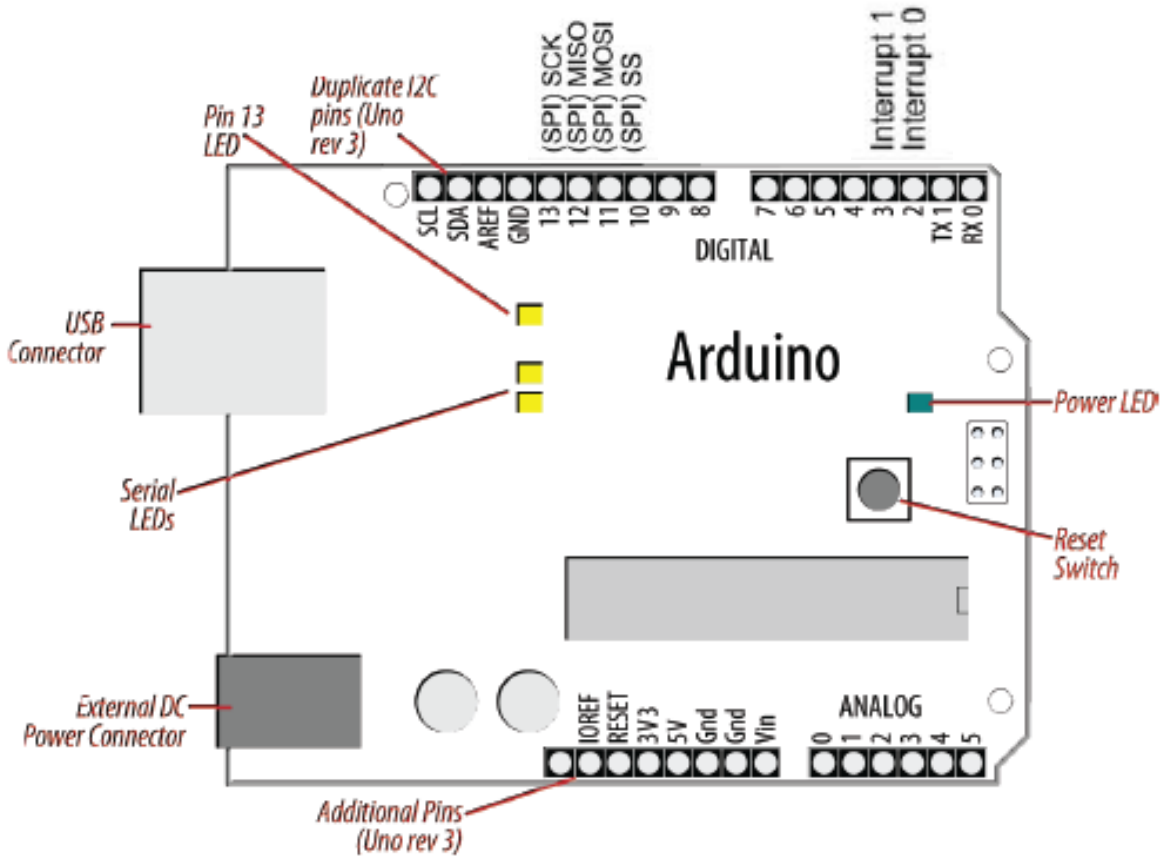


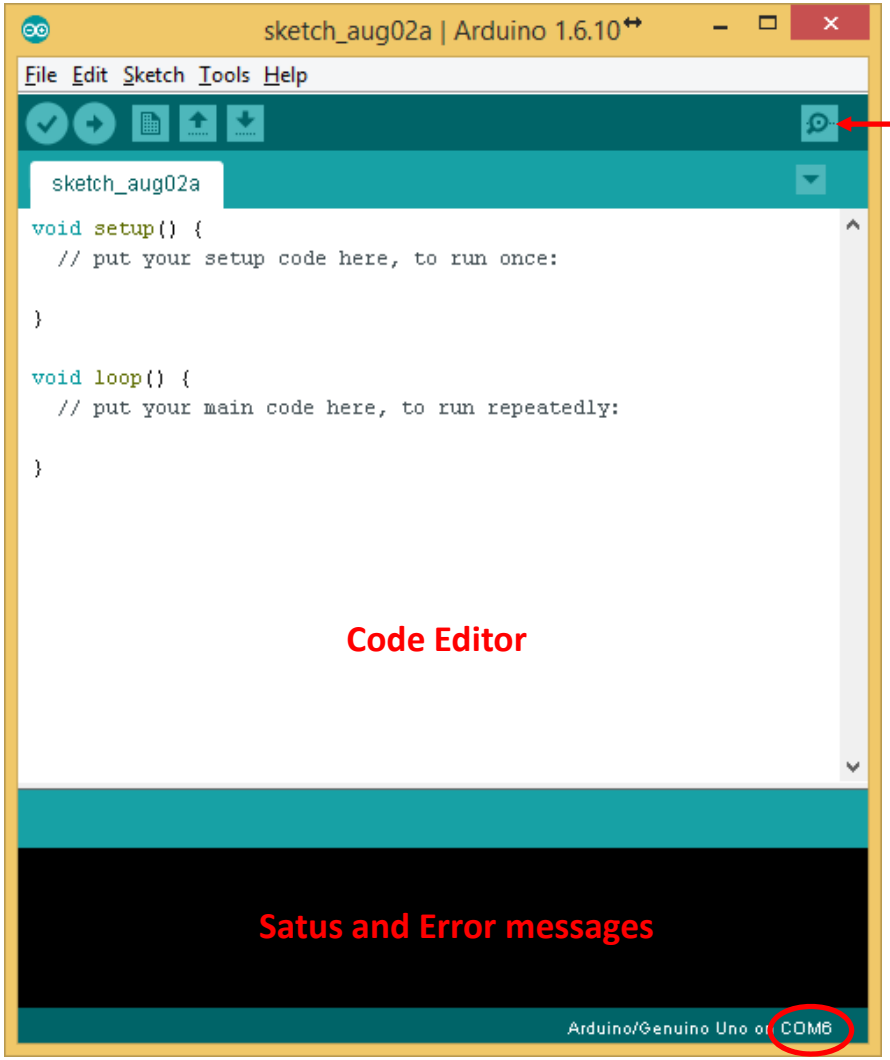
Microcontroller: ATMega328P

(http://www.atmel.com/devices/atmega328p.aspx?tab=documents)

Τάση λειτουργίας	+ 5 V
Εξωτερική τροφοδοσία	7-12 V
Ψηφιακές είσοδοι/έξοδοι	14
Υποστήριξη PWM	6 από τις 14
Αναλογικές είσοδοι	6
Μέγιστο ρεύμα για κάθε I/O	40 mA
Μέγιστο ρεύμα για τα 3.3 V	50 mA
Μνήμη προγράμματος (Flash)	32 KB (0.5 KB bootloader)
Μνήμη δεδομένων (SRAM)	2 KB
Μνήμη EEPROM	1 KB
Ρολόϊ	16 MHz
Υποστηριζόμενα πρωτόκολλα	UART, SPI, I2C

- Διαθέτει το ATMega16U2 σαν μετατροπέα USB-to-serial, για σύνδεση με υπολογιστή. Μπορεί να λειτουργήσει με την τροφοδοσία του USB για ρεύμα έως 500 mA.
- Ενσωματώνει ένα LED το οποίο είναι συνδεδεμένο με το pin 13.
- Διαθέτει την είσοδο AREF μέσω της οποίας μπορεί να δοθεί εξωτερική τάση αναφοράς στον ενσωματωμένο A/D.
- Διαθέτει ένα πλήκτρο RESET.
- Διαθέτει δύο LED (RX και TX) τα οποία αναβοσβήνουν όταν υπάρχει επικοινωνία με τον υπολογιστή μέσω USB.


Pin 13
LED
Duplicate I2C
pins (Uno
rev 3)
(SPI) SCK
(SPI) MISO
(SPI) MOSI
(SPI) SS
Interrupt 1
Interrupt 0
SCL
SDA
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TX 1
RX 0
DIGITAL
USB
Connector
Arduino
Power LED
Serial
LEDs
Reset
Switch
External DC
Power Connector
IOREF
RESET
3V3
5V
Gnd
Gnd
Vin
ANALOG
0
1
2
3
4
5
Additional Pins
(Uno rev 3)



Serial monitor

Compile

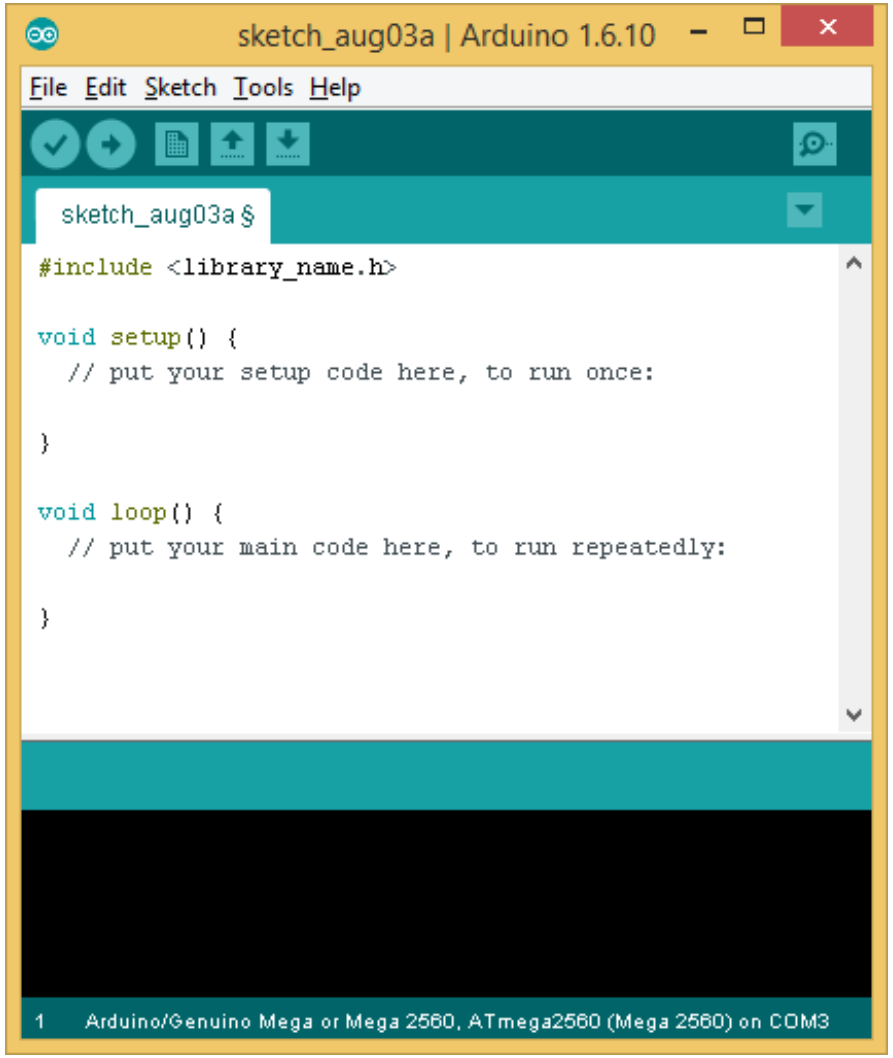
Upload to board

New sketch

Open existing sketch

Save sketch

https://www.arduino.cc/en/Main/Software (version 1.6.10)



- Η συνάρτηση **setup()** καλείται κατά την έναρξη του προγράμματος ή μετά από Reset και εκτελείται μόνο μια φορά. Χρησιμοποιείται για την αρχικοποίηση των μεταβλητών, της λειτουργικότητας των χρησιμοποιούμενων pins, κλπ.

- Η συνάρτηση **loop()** εκτελείται μετά την setup() και είναι το κυρίως πρόγραμμα που εκτελείται συνεχώς.


```
int led=13;

void setup() {
  pinMode(led,OUTPUT)
}
```

```
const int led=13;    //read-only variable

void setup() {
  pinMode(led,OUTPUT)
}
```

```
define led 13

void setup() {
  pinMode(led,OUTPUT)
}
```

Άσκηση

Να γραφτεί πρόγραμμα το οποίο να κάνει blink το led 13 10 φορές και μετά να το αφήνει σβηστό.

Arduino cheatsheet

ARDUINO CHEAT SHEET V.02c

Mostly taken from the extended reference:
http://arduino.cc/en/Reference/Extended
Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace

Structure
void **setup**() void **loop**()

Control Structures
if (x<5){ } **else** { }
switch (myvar) {
case 1:
break;
case 2:
break;
default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } **while** (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto // considered harmful :-)

Further Syntax
// (single line comment)
/* (multi-line comment) */
#**define** DOZEN 12 //Not baker's!
#**include** <avr/pgmspace.h>

General Operators
= (assignment operator)
+ (addition) - (subtraction)
* (multiplication) / (division)
% (modulo)
== (equal to) != (not equal to)
< (less than) > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and) || (or) ! (not)

Pointer Access
& reference operator
* dereference operator

Bitwise Operators
& (bitwise and) | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left) >> (bitshift right)

Compound Operators
++ (increment) -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

Constants
HIGH | LOW
INPUT | OUTPUT
true | false
143 // **Decimal** number
0173 // **Octal** number
0b11011111 //**Binary**
0**x**7B // **Hex** number
7**U** // Force unsigned
10**L** // Force long
15**UL** // Force long unsigned
10.0 // Forces floating point
2.4**e**5 // 240000

Data Types
void
boolean (0, 1, false, true)
char (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte (0 to 255)
int (-32,768 to 32,767)
unsigned int (0 to 65535)
word (0 to 65535)
long (-2,147,483,648 to 2,147,483,647)
unsigned long (0 to 4,294,967,295)
float (-3.4028235E+38 to 3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes

Strings
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
//Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";

Arrays
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};

Conversion
char() **byte**()
int() **word**()
long() **float**()

Qualifiers
static // persists between calls
volatile // use RAM (nice for ISR)
const // make read-only
PROGMEM // use flash

Digital I/O
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int **digitalRead**(pin)
//Write High to inputs to use pull-up res

Analog I/O
analogReference([DEFAULT,INTERNAL,EXTERNAL])
int **analogRead**(pin) //Call twice if switching pins from high Z source.
analogWrite(pin, value) // PWM

Advanced I/O
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
unsigned long **pulseIn**(pin, [HIGH,LOW])

Time
unsigned long **millis**() // 50 days overflow.
unsigned long **micros**() // 70 min overflow
delay(ms)
delayMicroseconds(us)

Math
min(x, y) **max**(x, y) **abs**(x)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
pow(base, exponent) **sqrt**(x)
sin(rad) **cos**(rad) **tan**(rad)

Random Numbers
randomSeed(seed) // Long or int
long **random**(max)
long **random**(min, max)

Bits and Bytes
lowByte() **highByte**()
bitRead(x,bitn) **bitWrite**(x,bitn,bit)
bitSet(x,bitn) **bitClear**(x,bitn)
bit(bitn) //bitn: 0-LSB 7-MSB

External Interrupts
attachInterrupt(interrupt, function, [LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()

Libraries:

Serial
begin([300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200])
end()
int **available**()
int **read**()
flush()
print()
println()
write()

EEPROM (#include <EEPROM.h>)
byte **read**(intAddr)
write(intAddr,myByte)

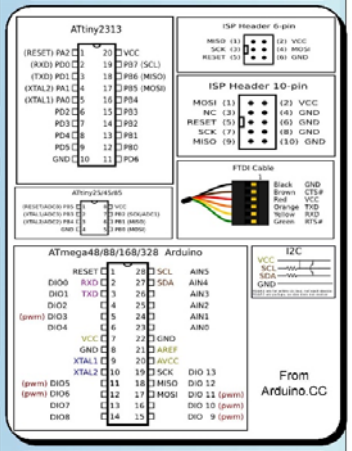
Servo (#include <Servo.h>)
attach(pin , [min_uS, max_uS])
write(angle) // 0-180
writeMicroseconds(uS) //1000-2000, 1500 is midpoint
read() // 0-180
attached() //Returns boolean
detach()

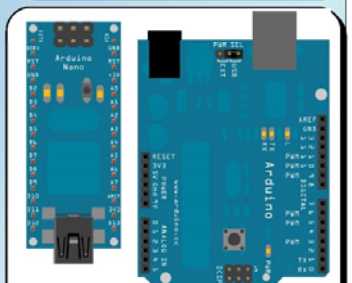
SoftwareSerial(RxPin,TxPin)
// #include<**SoftwareSerial**.h>
begin(longSpeed) // up to 9600
char **read**() // blocks till data
print(myData) or **println**(myData)

Wire (#include <Wire.h>) // For **I2C**
begin() // Join as master
begin(addr) // Join as slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(mybyte) // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission() // Step 3
byte **available**() // Num of bytes
byte **receive**() //Return next byte
onReceive(handler)
onRequest(handler)

	ATMega168	ATMega328	ATmega1280
Flash (2k for bootbloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (Nano has 14+8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,19,18 (IRQ0- IRQ5)
PWM pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0-13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL





Pics from Fritzing.Org under C.C. license

Data types

short ⟷

uint16_t ⟷

word

uint8_t ⟷

Numeric types	Bytes	Range	Use
`int`	2	–32768 to 32767	Represents positive and negative integer values.
`unsigned int`	2	0 to 65535	Represents only positive values; otherwise, similar to `int`.
`long`	4	–2147483648 to 2147483647	Represents a very large range of positive and negative values.
`unsigned long`	4	4294967295	Represents a very large range of positive values.
`float`	4	3.4028235E+38 to –3.4028235E+38	Represents numbers with fractions; use to approximate real-world measurements.
`double`	4	Same as `float`	In Arduino, `double` is just another name for `float`.
`boolean`	1	`false` (0) or `true` (1)	Represents true and false values.
`char`	1	–128 to 127	Represents a single character. Can also represent a signed value between –128 and 127.
`byte`	1	0 to 255	Similar to `char`, but for unsigned values.

Other types	Use
`String`	Represents arrays of `chars` (characters) typically used to contain text.
`void`	Used only in function declarations where no value is returned.

https://www.arduino.cc/en/Reference/HomePage

Παραδείγματα

```
int x;
int y;
float z;

x = 1;
y = x / 2;            // y now contains 0, ints can't hold fractions
z = (float)x / 2.0;   // z now contains .5 (you have to use 2.0, not 2)
```

```
int inputPins[] = {2,3,4,5};  // create an array of pins for switch inputs

int ledPins[] = {10,11,12,13};  // create array of output pins for LEDs

void setup()
{
  for(int index = 0; index < 4; index++)
  {
    pinMode(ledPins[index], OUTPUT);        // declare LED as output
    pinMode(inputPins[index], INPUT);       // declare pushbutton as input
```

```
char myChar = 'A';
char myChar = 65;       // both are equivalent
```

```
char Str1[15];
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[ ] = "arduino";
char Str5[8] = "arduino";
char Str6[15] = "arduino";
```

Συναρτήσεις για character strings

- strlen(string)
- strcpy(destination, source)
- strncpy(destination, source, 6)
- strcat(destination, source)
- strcmp(str, "Arduino")

Συναρτήσεις της κλάσης String

```
char oldString[] = "I want this character array in a String object";
String newString = oldString;
```

`charAt(n)`	Returns the *n*th character of the `String`
`compareTo(S2)`	Compares the `String` to the given `String` S2
`concat(S2)`	Returns a new `String` that is the combination of the `String` and S2
`endsWith(S2)`	Returns true if the `String` ends with the characters of S2
`equals(S2)`	Returns true if the `String` is an exact match for S2 (case-sensitive)
`equalsIgnoreCase(S2)`	Same as `equals` but is not case-sensitive
`getBytes(buffer,len)`	Copies `len`(gth) characters into the supplied byte buffer
`indexOf(S)`	Returns the index of the supplied `String` (or character) or –1 if not found
`lastIndexOf(S)`	Same as `indexOf` but starts from the end of the `String`
`length()`	Returns the number of characters in the `String`
`replace(A,B)`	Replaces all instances of `String` (or character) A with B
`setCharAt(index,c)`	Stores the character c in the `String` at the given index
`startsWith(S2)`	Returns true if the `String` starts with the characters of S2
`substring(index)`	Returns a `String` with the characters starting from index to the end of the `String`
`substring(index,to)`	Same as above, but the substring ends at the character location before the 'to' position
`toCharArray(buffer,len)`	Copies up o `len` characters of the `String` to the supplied buffer
`toInt()`	Returns the integer value of the numeric digits in the `String`
`toLowerCase()`	Returns a `String` with all characters converted to lowercase
`toUpperCase()`	Returns a `String` with all characters converted to uppercase
`trim()`	Returns a `String` with all leading and trailing whitespace removed

Παραδείγματα

number → string

```
String str = String(13)        → str="13"
String str = String(13,HEX) → str="D"
String str = String(13,BIN)  → str="1101"
```

number → string

```
int value1 = 13;
long value2 = 1234;
char buffer1[7];
char buffer2[12];
itoa(value1,buffer1,10);
itoa(value2,buffer2,10);
```

string → number

```
char str[]="13";
int var=atoi(str);
```

string → number

```
String str1="1234";
int var=str1.toInt();
```

string → number

```
String str1="1234";
char str2[5];
str1.toCharArray(str2,5);
int var=atoi(str2);
```

Compare strings

```
char string1[] = '1234';
char string2[] = '4567';
if(strcmp(string1,string2) == 0) {

}
```

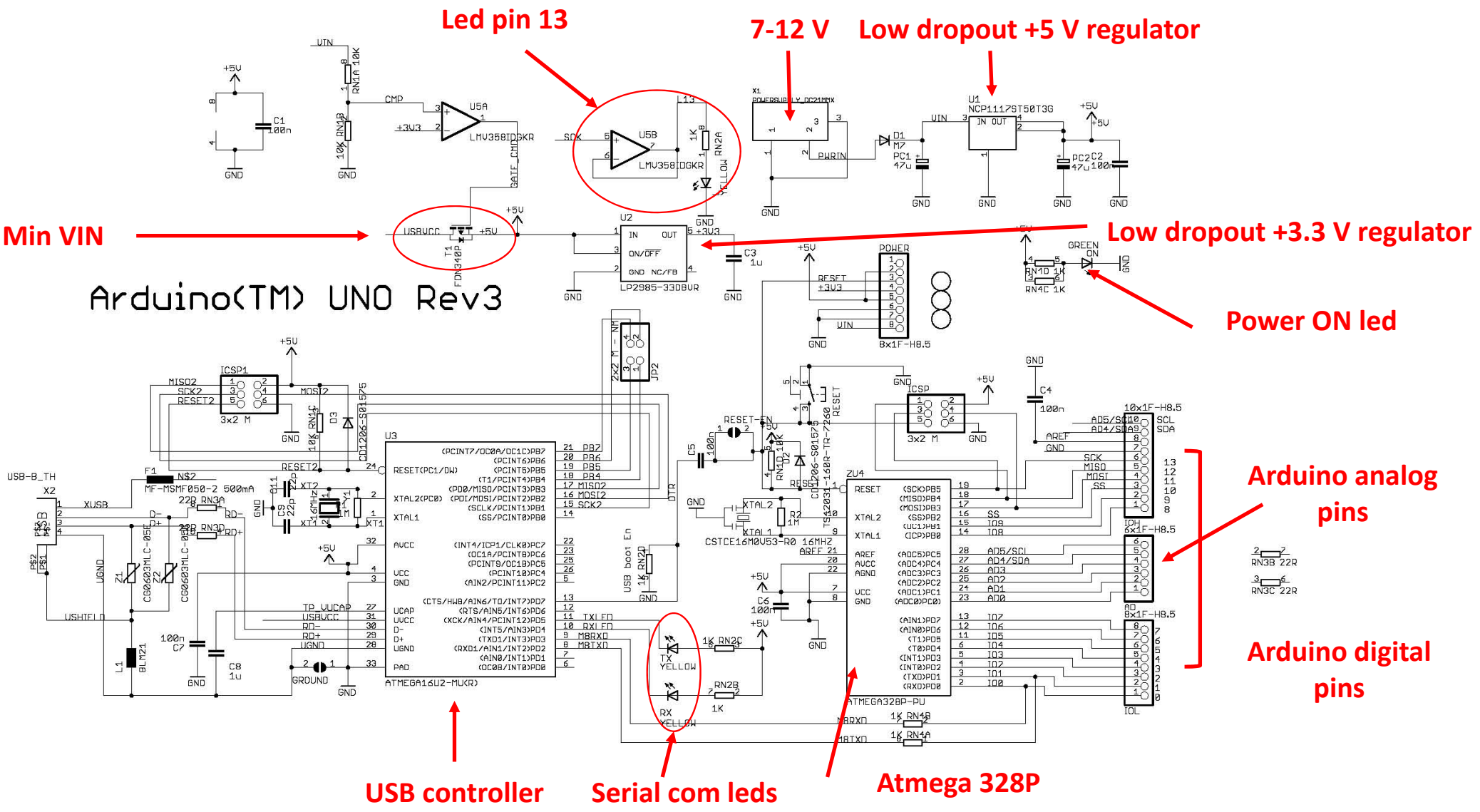
Operator	Example	Equivalent expression
+=	value += 5;	value = value + 5; // add 5 to value
-=	value -= 4;	value = value - 4; // subtract 4 from value
*=	value *= 3;	value = value * 3; // multiply value by 3
/=	value /= 2;	value = value / 2; // divide value by 2
>>=	value >>= 2;	value = value >> 2; // shift value right two places
<<=	value <<= 2;	value = value << 2; // shift value left two places
&=	mask &= 2;	mask = mask & 2; // binary-and mask with 2
|=	mask |= 2;	mask = mask | 2; // binary-or mask with 2

```
int x = 6;
int result = x << 1; // 6 shifted left 1 is 12
int result = x >> 2; // 12 shifted right 2 is 3;
```

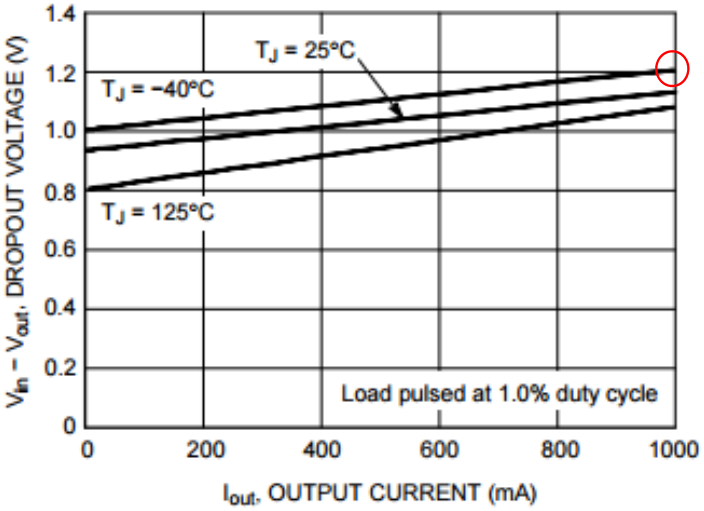
```
int x = 258;                  // 258 is 0x102 in HEX
hiByte = highByte(x);         // hiByte=01
loByte = lowByte(x);          // loByte=02
x = word(hiByte,loByte);  // x=258
```

- constrain(x, min, max) → Επιστρέφει τιμή μεταξύ min και max
- min(x,y) → Επιστρέφει την ελάχιστη τιμή μεταξύ x και y
- max(x,y) → Επιστρέφει τη μέγιστη τιμή μεταξύ x και y
- pow(x, y) → Επιστρέφει x^y
- sqrt(x) → Επιστρέφει την τετραγωνική ρίζα του x
- floor(x) → Επιστρέφει την ακέραια τιμή που είναι μικρότερη του x
- ceil(x) → Επιστρέφει την ακέραια τιμή που είναι μεγαλύτερη του x
- sin(x)-cos(x)-tan(x) → Ημίτονο-Συνημίτονο-Εφαπτομένη της γωνίας x σε rad. (x=degrees*PI/180)
- random(max) → Επιστρέφει τυχαία τιμή από 0 έως max-1
- random(min,max) → Επιστρέφει τυχαία τιμή από min έως max-1
- randomSeed(1234) → Επανεκκινεί την ψευδοτυχαία γεννήτρια
- bitSet(x, bitPosition)
- bitClear(x, bitPosition)
- bitRead(x, bitPosition)
- bitWrite(x, bitPosition, value)

Arduino schematic


Led pin 13
7-12 V
Low dropout +5 V regulator
Min VIN
Low dropout +3.3 V regulator
Power ON led
Arduino(TM) UNO Rev3
Arduino analog pins
Arduino digital pins
USB controller
Serial com leds
Atmega 328P

NCP1117-5 V (1 A)



Max dropout voltage=1.2 V

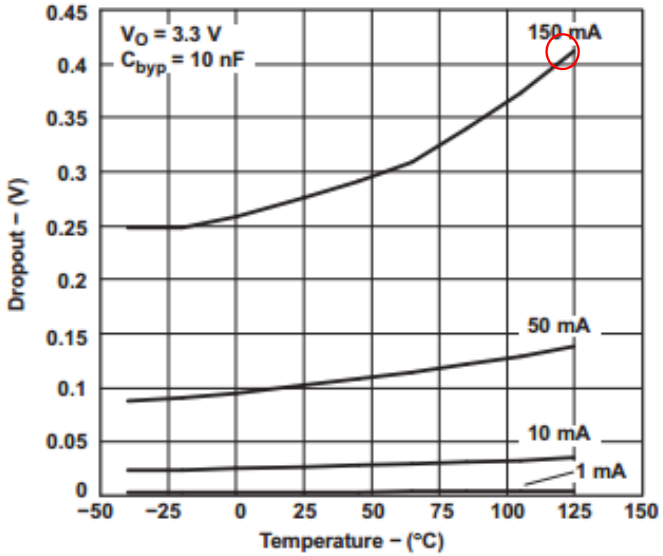


Figure 3. Dropout Voltage vs Temperature

LP2985 150-mA Low-noise Low-dropout Regulator – 3.3 V

Digital pin 0 → Rx (pin 2/RxD of μC)
Digital pin 1 → Tx (pin 3/TxD of μC)
Digital pin 2 → General I/O (pin 4/PD2/INT0 of μC)
Digital pin 3 → General I/O ή PWM (pin 5/PD3/INT1 of μC)
Digital pin 4 → General I/O (pin 6/PD4/T0 of μC)
Digital pin 5 → General I/O ή PWM (pin 11/PD5/T1 of μC)
Digital pin 6 → General I/O ή PWM (pin 12/PD6/AIN0 of μC)
Digital pin 7 → General I/O (pin 13/PD7/AIN1 of μC)

Σημείωση: Οι εναλλακτικοί τρόποι λειτουργίας των digital pins φαίνονται στο data sheet, σελ. 88, Πίνακας 13-9.

Digital pin 8 → General I/O (pin 14/PB0/ICP of μC)
Digital pin 9 → General I/O ή PWM (pin 15/PB1/OC1 of μC)
Digital pin 10 → General I/O ή PWM (pin 16/PB2/SS of μC)
Digital pin 11 → General I/O ή PWM (pin 17/PB3/MOSI of μC)
Digital pin 12 → General I/O (pin 18/PB4/MISO of μC)
Digital pin 13 → General I/O (pin 19/PB5/SCK of μC)

ICSP header

Σημείωση: Οι εναλλακτικοί τρόποι λειτουργίας των digital pins φαίνονται στο data sheet, σελ. 82, Πίνακας 13-3.

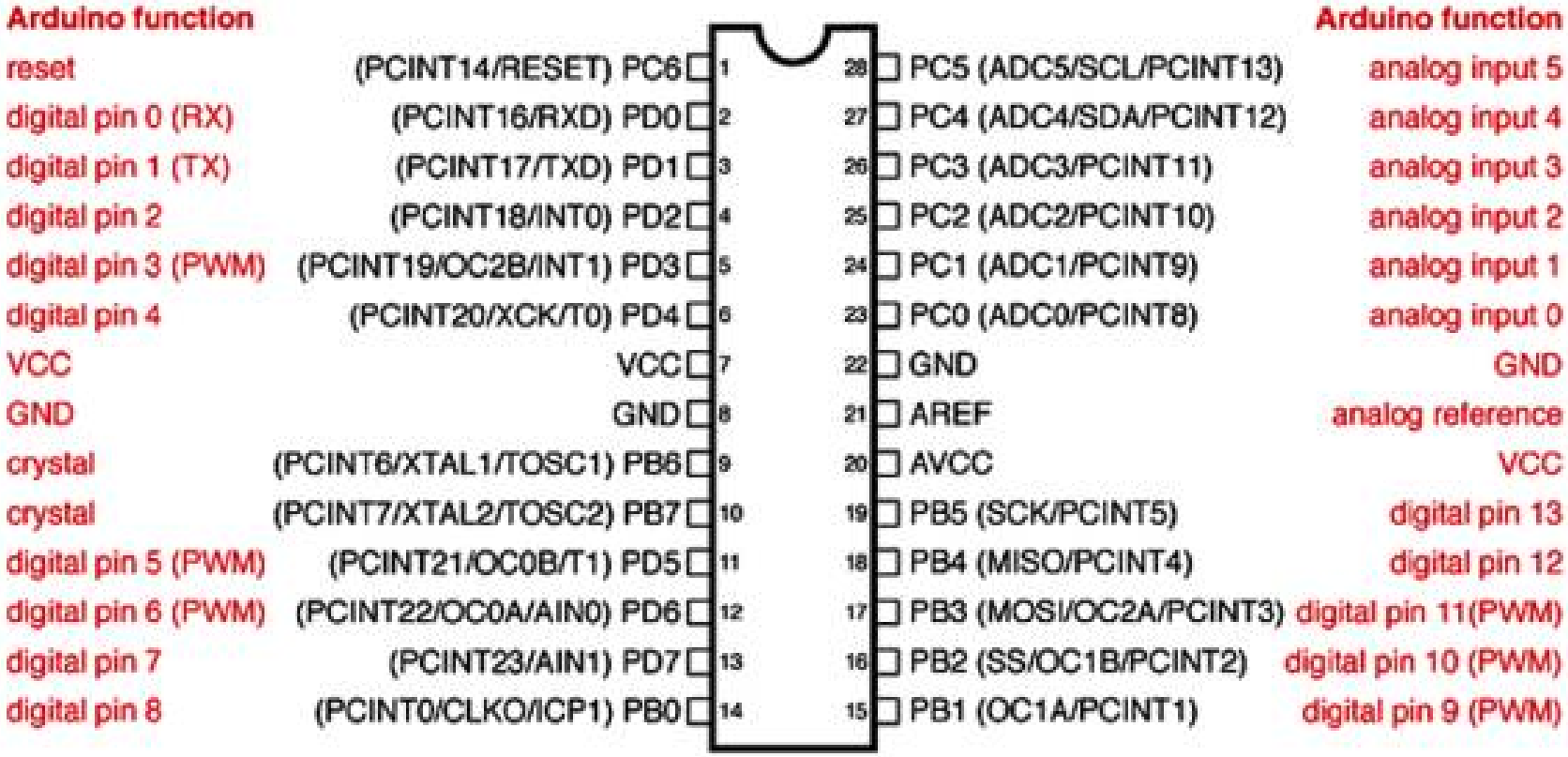
Analog pin 0 → General I/O ή Analog in (pin 23/PC0/ADC0 of μC)
Analog pin 1 → General I/O ή Analog in (pin 24/PC1/ADC1 of μC)
Analog pin 2 → General I/O ή Analog in (pin 25/PC2/ADC2 of μC)
Analog pin 3 → General I/O ή Analog in (pin 26/PC3/ADC3 of μC)
Analog pin 4 → General I/O ή Analog in (pin 27/PC4/ADC4 of μC)
Analog pin 5 → General I/O ή Analog in (pin 28/PC5/ADC5 of μC)

Σημείωση: Οι εναλλακτικοί τρόποι λειτουργίας των digital pins φαίνονται στο data sheet, σελ. 85, Πίνακας 13-6.

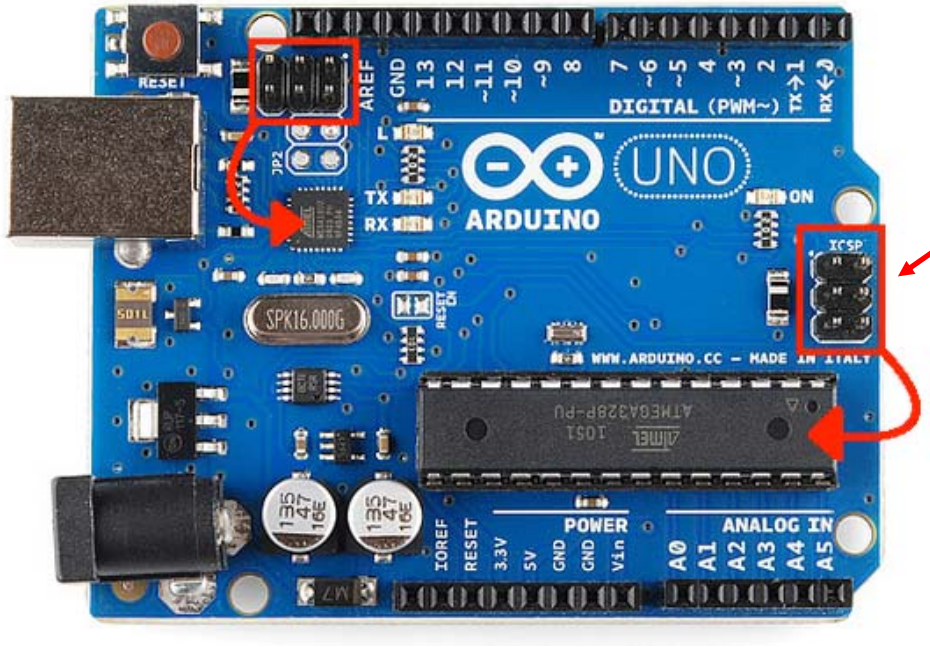
AREF → pin 21/AREF of μC (Εξωτερική είσοδος αναφοράς για τις αναλογικές εισόδους. Ο A/D είναι 10 bits με εσωτερική τάση αναφοράς 5 V. Δηλαδή μετατρέπει τάσεις από 0 έως 5V σε 0 έως 1024 ή 000h έως 3FFh).

GND → pin 22/AGND of μC (Αναλογική γείωση).

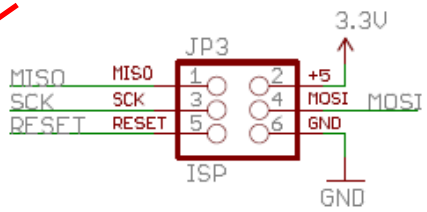
ATmega168/328 Pin Mapping



Digital Pins 11,12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.



ICSP header (In-Circuit Serial Programming)



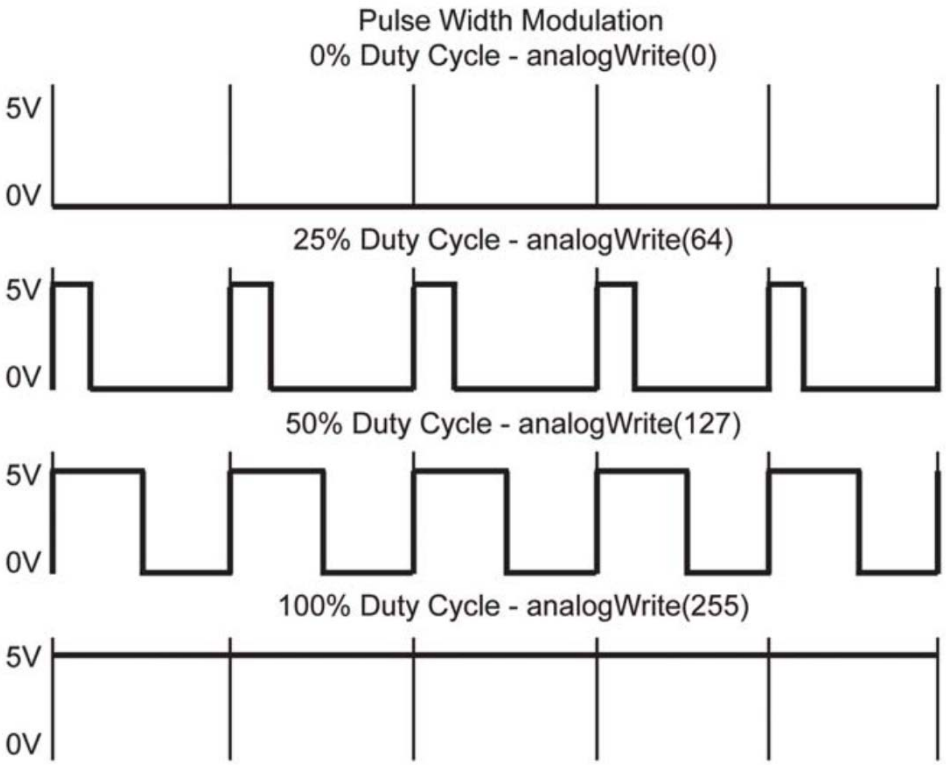
e.g. Plug a programmer to flash the bootloader (AVRISP mkII)

Λειτουργικότητα Εισόδων-Εξόδων

Digital pins → μπορεί να είναι HIGH ή LOW

- Ο ακροδέκτης που προγραμματίζεται σαν **INPUT** με την εντολή **pinMode()** τίθεται σε κατάσταση υψηλής αντίστασης. Είναι σαν να υπάρχει μια αντίσταση 100 Megohms μπροστά από τον ακροδέκτη. Ο τρόπος είναι χρήσιμος για την ανάγνωση αισθητήρων αλλά δεν χρησιμοποιείται για την οδήγηση LED.

- Ο ακροδέκτης που προγραμματίζεται σαν **OUTPUT** με την εντολή **pinMode()** παρουσιάζει χαμηλή αντίσταση. Αυτό σημαίνει ότι μπορεί να διαθέσει αρκετό ρεύμα για να οδηγήσει άλλα κυκλώματα. Οι ακροδέκτες του Atmega μπορούν να δώσουν (source) ή να πάρουν (sink) ρεύμα μέχρι 40 mA. Επομένως, μπορούν να οδηγήσουν LED αλλά δεν χρησιμοποιούνται για την ανάγνωση αισθητήρων. Οι ακροδέκτες που προγραμματίζονται σαν έξοδοι μπορούν να καταστραφούν εάν συνδεθούν στη γείωση ή την τάση. Το παραπάνω ρεύμα δεν είναι αρκετό για την οδήγηση ρελαί ή μοτέρ και γι’ αυτόν το λόγο χρειάζονται ενδιάμεσα κυκλώματα σε αυτές της περιπτώσεις.



Λειτουργία των digital pins σαν εξόδων PWM (Pulse Width Modulation)

Τα Digital pins 3, 5, 6, 9, 10, 11 μπορούν να χρησιμοποιηθούν για τη λειτουργία PWM, δηλαδή να κάνουν έξοδο τετραγωνικούς παλμούς με μεταβλητό Duty Cycle. Η εντολή με την οποία δημιουργούμε PWM είναι η **analogWrite(pin, value)**, όπου value είναι μια τιμή από 0 έως 255. Εάν η τιμή είναι 0, τότε η τάση εξόδου είναι μηδενική. Εάν η τιμή είναι 255 τότε η τάση εξόδου είναι +5 V, ενώ με ενδιάμεσες τιμές έχουμε τετραγωνικούς παλμούς όπως αυτοί απεικονίζονται στην εικόνα. Η συχνότητα των τετραγωνικών παλμών είναι περίπου 500 Hz και επομένως η περίοδος είναι περίπου 2ms.

Λειτουργία των ψηφιακών ακροδεκτών

Input

pinMode(pin,INPUT)

Το *digitalRead(pin)* επιστρέφει HIGH εάν η τάση εισόδου είναι >=3V και LOW εάν είναι <=2 V.

Εάν ο ακροδέκτης εισόδου γίνει HIGH με το *digitalWrite*(pin,HIGH), η εσωτερική pull-up αντίσταση των 20 ΚΩ ενεργοποιείται. Τότε, διαβάζοντας αυτόν τον ακροδέκτη με *digitalRead*(pin) θα επιστρέψει HIGH εκτός εάν γίνεται LOW από το εξωτερικό κύκλωμα. Αυτό χρησιμοποιείται για να έχουμε κάποιον ακροδέκτη σε γνωστή κατάσταση και όχι στον αέρα.

Output

pinMode(pin,OUTPUT)

Το *digitalWrite*(pin,HIGH) θα δώσει έξοδο 5V και θα δώσει ρεύμα 40 mA (source).

Το *digitalWrite*(pin,LOW) θα δώσει έξοδο 0V και θα τραβήξει ρεύμα 40 mA (sink).

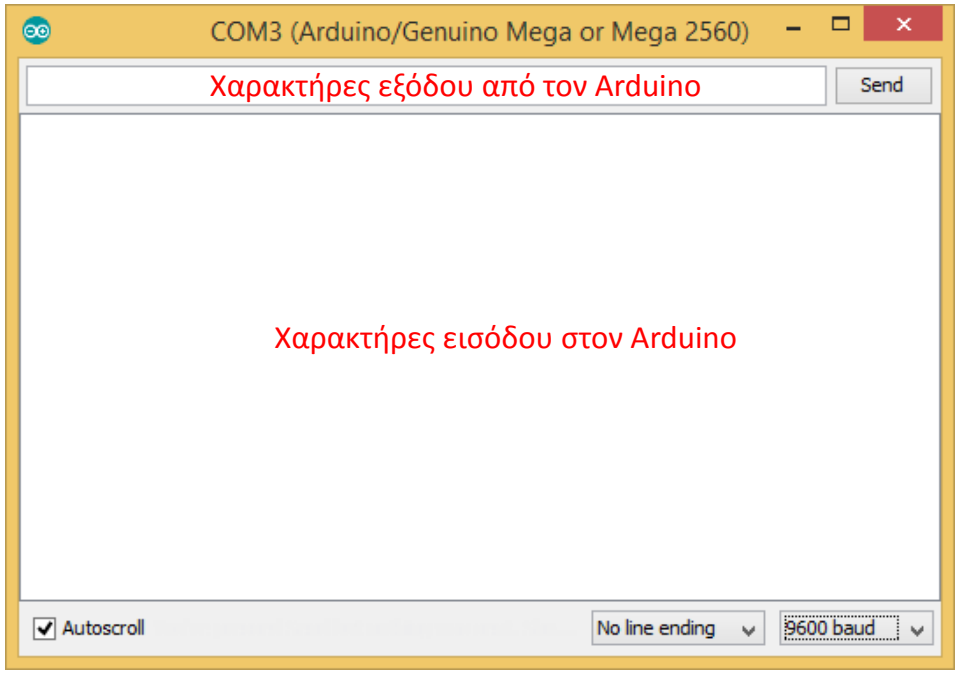
Τα **Analog pins** του Arduino είναι 6 και προσδιορίζονται σαν Analog 0 έως 5. Οι αναλογικές είσοδοι χρησιμοποιούνται κυρίως για την ανάγνωση αισθητήρων μέσω των 6 καναλιών του Analog to Digital Converter (ADC). Ο ADC μετατρέπει την τάση εισόδου (0V to 5V) σε ακέραιους από 0 έως 1023 (ανάλυση 10 bits). Η ανάγνωση της αναλογικής εισόδου γίνεται με τη συνάρτηση **analogRead()**. Η τάση αναφοράς ρυθμίζεται με τη συνάρτηση **analogReference()** ως εξής:

- DEFAULT (+5V)
- INTERNAL (built-in 1.1V)
- EXTERNAL (Voltage to pin AREF)

Σειριακή οθόνη

Το περιβάλλον ανάπτυξης διαθέτει την οθόνη σειριακής επικοινωνίας (**serial monitor**) η οποία απεικονίζει τη σύνδεση του υπολογιστή με τον Arduino Uno μέσω USB. Οι ακροδέκτες που πραγματοποιούν τη σειριακή επικοινωνία είναι τα pins 0 (RX) και 1 (TX) της κάρτας. Έτσι, μπορούμε να κάνουμε έξοδο στην οθόνη του υπολογιστή (χαρακτήρες ASCII) και είσοδο από τον υπολογιστή (χαρακτήρες ASCII). Η προ-τοποθετημένη ταχύτητα σειριακής επικοινωνίας είναι 9600 bps.

Η ενεργοποίηση της σειριακής επικοινωνίας στον Arduino γίνεται με την εντολή:
Serial.begin(9600);



Εάν δεν υπάρχει σειριακή επικοινωνία εμφανίζεται το παρακάτω σφάλμα. Για τη διόρθωση του σφάλματος πρέπει να έχουν εγκατασταθεί οι σωστοί drivers, επιλεγεί η σωστή κάρτα και η σωστή σειριακή θύρα επικοινωνίας.

avrdude; stk500_getsync(): resp=0x00

Tools → Serial Monitor

Σύνοψη εντολών

ΣΥΝΟΨΗ ΒΑΣΙΚΩΝ ΕΝΤΟΛΩΝ	
pinMode(pin, INPUT/OUTPUT)	Ο ψηφιακός ακροδέκτης pin γίνεται είσοδος/έξοδος
pinMode(pin, INPUT_PULLUP)	Ο ψηφιακός ακροδέκτης pin γίνεται είσοδος ενεργοποιώντας την εσωτερική αντίσταση pull-up
int x = digitalRead(pin)	Ανάγνωση της λογικής κατάστασης του ψηφιακού ακροδέκτη pin
digitalWrite(pin, LOW/HIGH)	Εγγραφή 0/1 (0 V/+5 V) στον ψηφιακό ακροδέκτη pin
analogWrite (pin,value)	Εγγραφή της τιμής value (0 έως 255) στον ψηφιακό ακροδέκτη pin
int x = analogRead(pin)	Ανάγνωση της αναλογικής εισόδου pin (0 έως 5)
analogReference("DEFAULT/INTERNAL/EXTERNAL")	Τοποθέτηση της τάσης αναφοράς του A/D στα +5 V/+1.1 V/AREF
Serial.begin(9600)	Ενεργοποίηση της σειριακής επικοινωνίας με baud rate=9600 bps
Serial.println(x)	Εκτύπωση στη σειριακή οθόνη της μεταβλητής x

delay(ms)	Καθυστέρηση ms. Ο επεξεργαστής δεν μπορεί να κάνει κάτι άλλο (π.χ. να χειριστεί εισόδους ή εξόδους). Δεν επηρεάζονται: η σειριακή επικοινωνία, οι έξοδοι PWM και οι διακοπές.
millis()	Επιστρέφει τον αριθμό ms από την έναρξη του προγράμματος (σαν unsigned long). Μηδενίζεται μετά από 50 ημέρες.

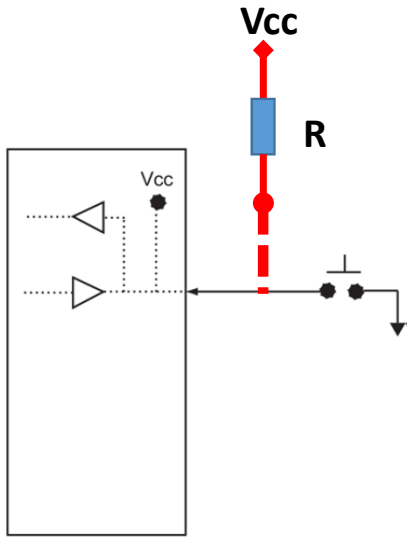
Παραδείγματα προγραμματισμού

Είσοδος δεδομένων

Digital Input Pull-up

(p1)

```
void setup(){
  // start serial connection
  Serial.begin(9600);
  // pin2 as an input enables the internal pull-up
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop(){
  // read the pushbutton value into a variable
  int sensorVal = digitalRead(2);
  // print out the value of the pushbutton
  Serial.println(sensorVal);

  // Keep in mind the pull-up means the push button's
  // logic is inverted. It goes HIGH when it's open,
  // and LOW when it's pressed. Turn on pin 13 when
  // button pressed, and off when not pressed
  if (sensorVal == HIGH) {
    digitalWrite(13, LOW);
  }
  else {
    digitalWrite(13, HIGH);
  }
}
```



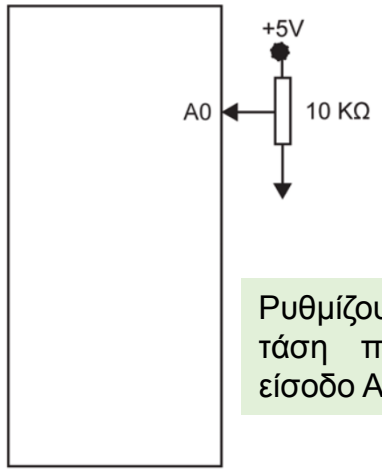
Read Analog Voltage

(p2)

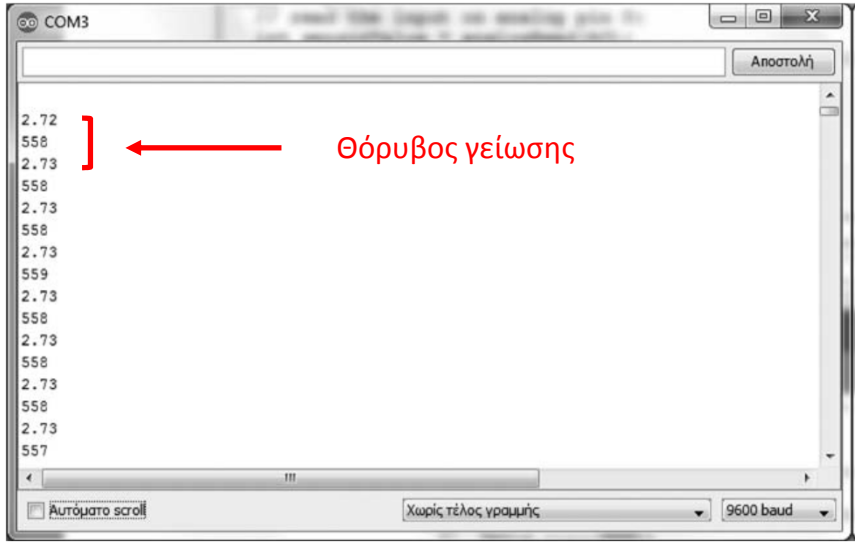
```
void setup() {
  // start serial connection
  Serial.begin(9600);
}

void loop() {
  // read the input on analog pin 0
  int sensorValue = analogRead(A0);
  // Convert analog reading (0-1023) to a voltage (0-5 V)
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read
  Serial.println(sensorValue);
  // print out the value of applied voltage
  Serial.println(voltage);
}
```

Στα +5 V η έξοδος του A/D είναι 1023

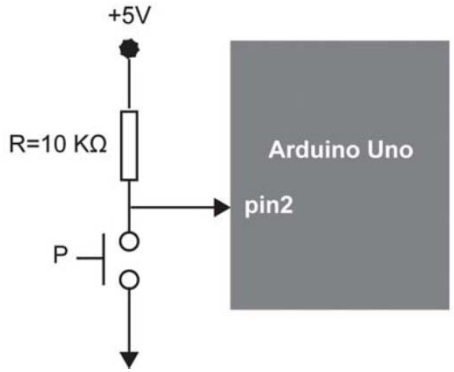
Στα voltage (+2.52 V) η έξοδος του A/D είναι sensorValue

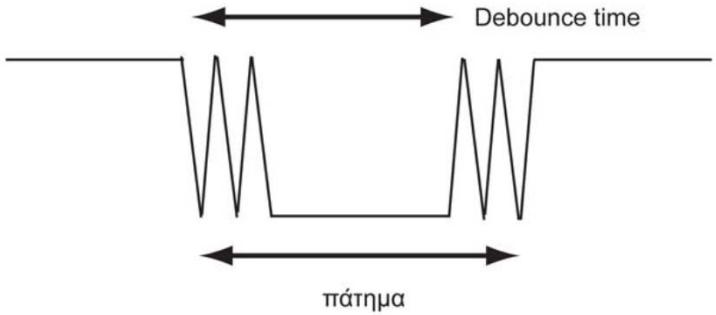


Ρυθμίζουμε το ποτενσιόμετρο έτσι ώστε η τάση που εφαρμόζεται στην αναλογική είσοδο A0 να είναι περίπου **2.52 V**



Σύνδεση πλήκτρου (Debounce)





- Όταν το πλήκτρο δεν είναι πατημένο, η τάση στην είσοδο είναι 5 V. Όταν είναι πατημένο, η τάση είναι 0 V. Ένα πάτημα (ON-OFF) δημιουργεί έναν παλμό.
- Ο παρακάτω κώδικας ανάβει το LED που είναι μόνιμα συνδεδεμένο στον Arduino Uno (pin 13) όσο το πλήκτρο είναι πατημένο και το σβήνει όταν το αφήσουμε.

```
int ledPin = 13;          // choose pin for the Arduino LED
int inputPin = 2;         // choose the input pin
int val = 0;              // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT);          // ledPin is output
  pinMode(inputPin, INPUT);         // inputPin is input
}

void loop() {
  val = digitalRead(inputPin);      // read push button
  if (val = = HIGH) {               // check if button released
    digitalWrite(ledPin, LOW);      // turn LED OFF
  } else {
      digitalWrite(ledPin, HIGH);  // turn LED ON
    }
}
```

(p3)

```
int ledPin = 13;          // pin for the Arduino LED
int inputPin = 2;         // input pin
int val = 0;              // variable for the pin status
int state = LOW;          // current state of the LED (initially LOW)
int previous = HIGH;      // previous reading of input pin
long time = 0;            // last time output pin was toggled
long debounce = 200;      // debounce time

void setup() {
  pinMode(ledPin, OUTPUT);     // ledPin as output
  pinMode(inputPin, INPUT);    // inputPin as input
}

void loop() {
  val = digitalRead(inputPin);   // read push button
  if (val = = HIGH && previous = = LOW && millis() - time > debounce) {
    // invert the output
    if (state = = HIGH)
      state = LOW;
    else
      state = HIGH;
    // remember when the last button press was
    time = millis();
  }
  digitalWrite(ledPin, state);
  previous = val;
}
```

(p4)

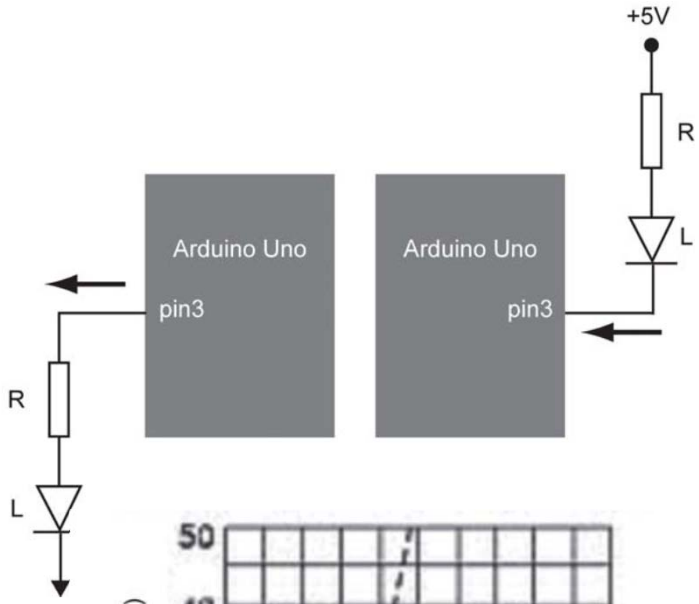
- Δημιουργούμε την τρέχουσα τιμή του πλήκτρου *val* και την προηγούμενη τιμή του *previous*, που αρχικά είναι και τα δύο “HIGH” (κατάσταση ηρεμίας).
- Όταν πατηθεί το πλήκτρο, τα *val* και *previous* σταθεροποιούνται σε κατάσταση “LOW” (πλήκτρο πατημένο).
- Όταν αφεθεί το πλήκτρο, για να θεωρηθεί ότι έγινε πληκτρολόγηση, πρέπει το *val* να είναι “HIGH” ενώ το previous να είναι “LOW” και να έχει παρέλθει χρόνος μεγαλύτερος από 200 ms (μεταβλητή “debounce”) από το τελευταίο πάτημα.

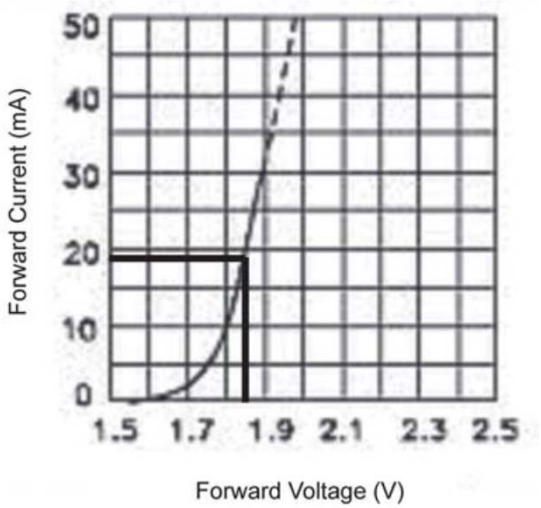
Η συνάρτηση *millis()* δίνει το χρόνο από τη στιγμή εκκίνησης του προγράμματος.

Παραδείγματα προγραμματισμού

Έξοδος δεδομένων

Σύνδεση LED





Συνδεσμολογία (α)

- Όταν η έξοδος Digital Pin 3 γίνει HIGH (VOH=5V) θα υπάρξει ένα ρεύμα προς το LED (source IOH) το οποίο από τα χαρακτηριτικά του μικροελεγκτή είναι περίπου 18 mA για έξοδο 4.5 V.
- Το ρεύμα αυτό είναι αρκετό για να ανάψει το LED. Εάν τοποθετήσουμε αντίσταση περιορισμού R πρέπει να υπολογίσουμε την τιμή της. Για ρεύμα IF=18mA, δημιουργείται πτώση τάσης VF=1.85 V, όπως δείχνει η χαρακτηριστική καμπύλη του LED στο Σχήμα 2.9. Επομένως:

R = (4.5-1.85) V/18 mA = 147 Ω ⇨ R = 120 Ω

Συνδεσμολογία (β)

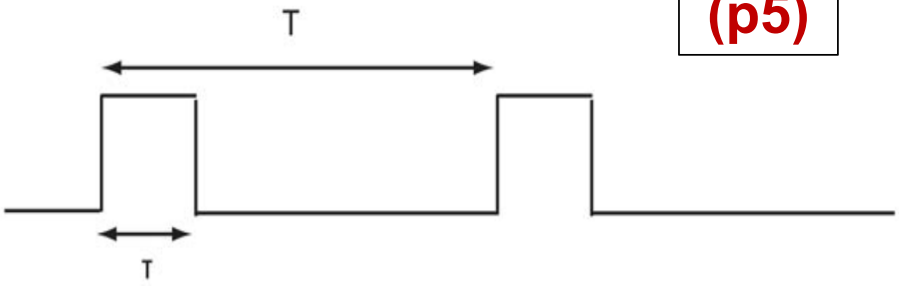
- Όταν η έξοδος Digital Pin 3 γίνει LOW (VOL=0V) θα υπάρξει ένα ρεύμα από την πηγή προς το LED (sink IOL) το οποίο από τα χαρακτηριστικά του μικροελεγκτή είναι περίπου 18 mA για έξοδο 0.5 V.
- Ο υπολογισμός της αντίστασης R γίνεται τώρα ως εξής:

R = (5-1.85-0.5) V/18 mA = 147 Ω ⇨ R = 120Ω

Παλμική οδήγηση (PWM)

```
int value = 0;           // variable to keep the actual value
int ledPin = 3;          // LED connected to digital pin 3

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // fade in (from min to max)
  for(value = 0 ; value <= 255; value+=5) {
    analogWrite(ledPin, value);          // range from 0-255
    delay(80);                           // delay of 80 ms
  }
  // fade out (from max to min)
  for(value = 255; value >=0; value-=5) {
    analogWrite(ledPin, value);
    delay(80);
  }
}
```

(p5)



Η παλμική οδήγηση χαρακτηρίζεται από τον κύκλο οδήγησης ή Duty Cycle, που ορίζεται ως εξής:

$$\textit{Duty cycle} = \tau * 100 / T$$

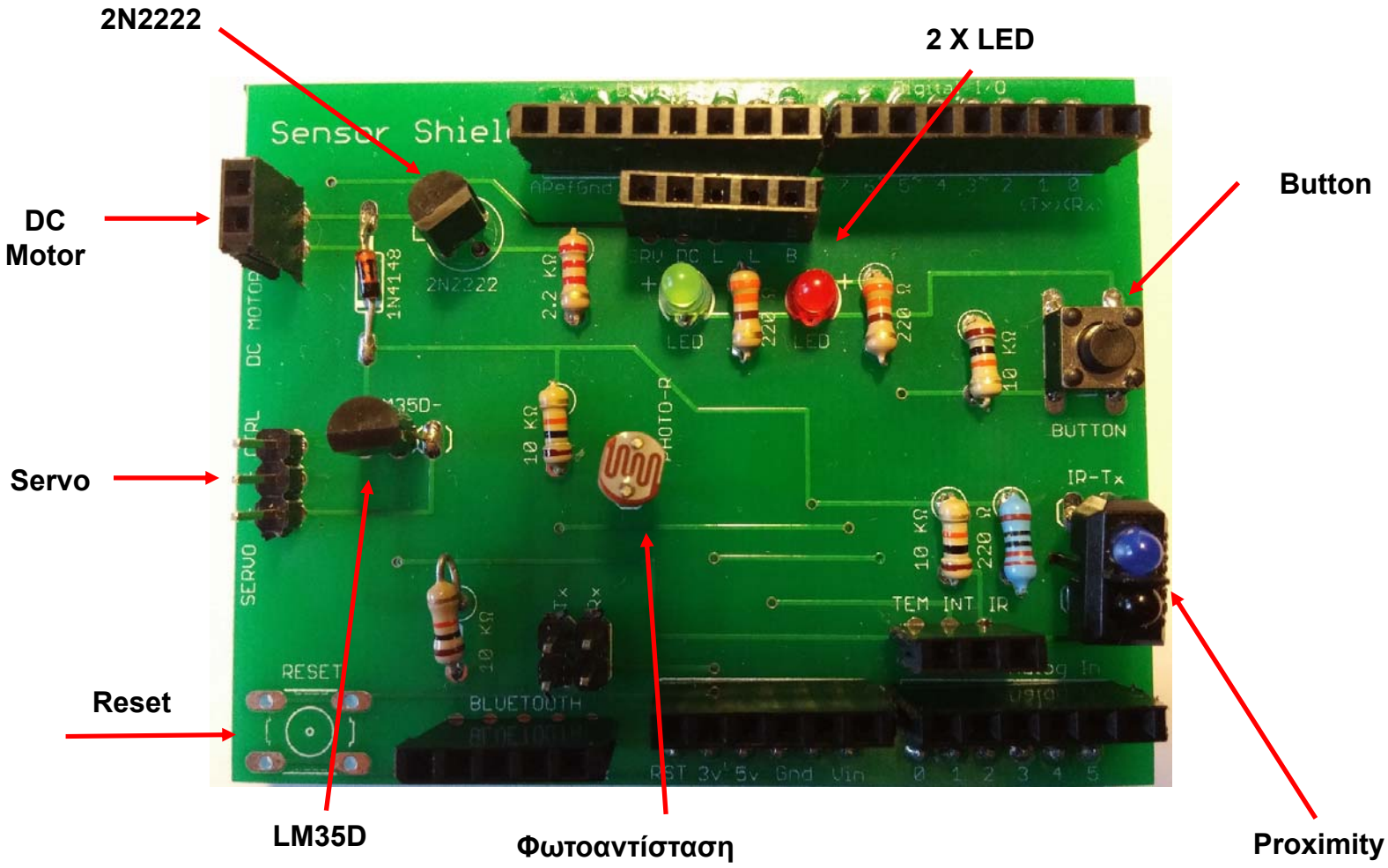
Οι λόγοι που οδηγούμε τα LED με παλμικό ρεύμα είναι οι εξής:

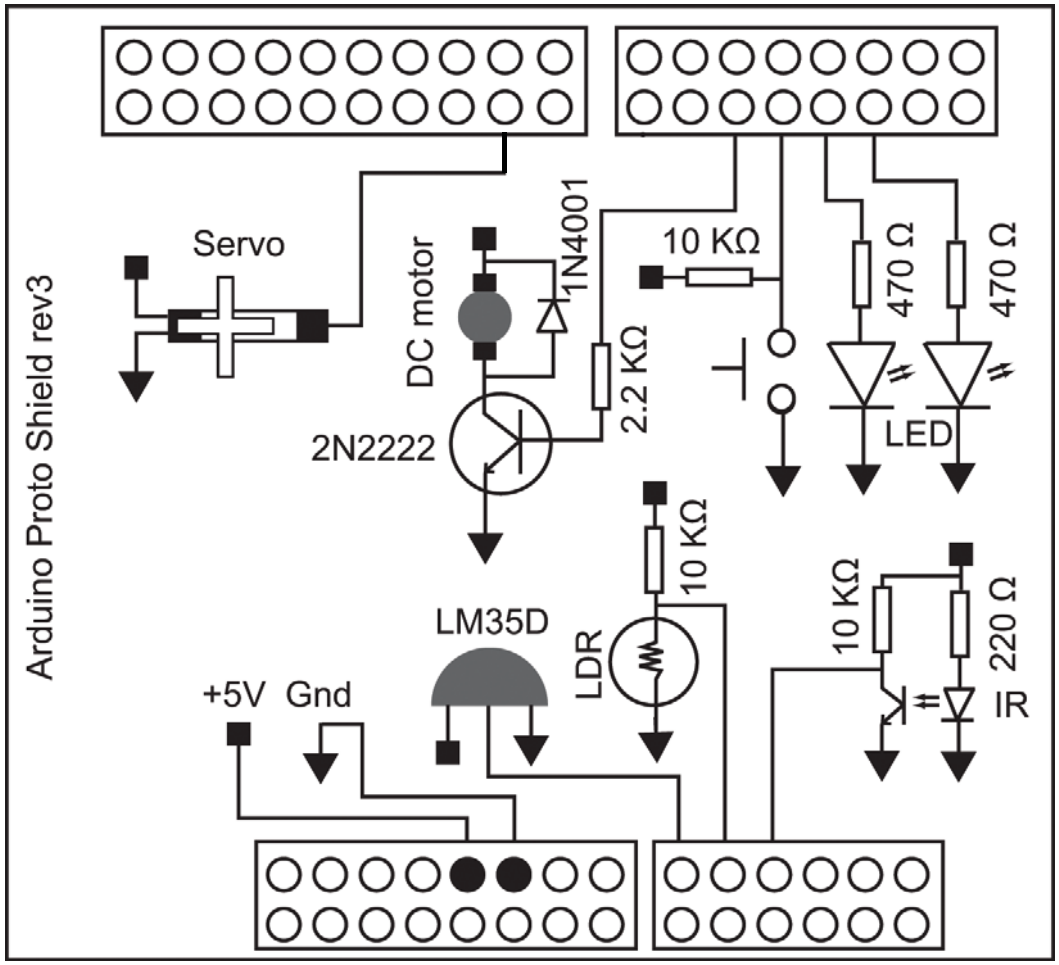
- Παρέχουν μεγαλύτερη φωτεινότητα λόγω της παραβολικής μορφής της καμπύλης φωτεινότητας των LED's σε σχέση με το ρεύμα ορθής πόλωσης. Ένα LED που οδηγείται με παλμικό ρεύμα 100 mA και Duty Cycle=10% (μέση τιμή ρεύματος 10 mA) παρέχει μεγαλύτερη φωτεινότητα από τη συνεχή του οδήγηση με ρεύμα 10 mA.
- Μεγαλύτερη διάρκεια ζωής.
- Μικρότερη κατανάλωση.

Η συχνότητα της οδήγησης πρέπει να είναι μεγαλύτερη των 50 Hz για να μην υπάρχει τρεμόπαιγμα (συνήθως είναι >=200 Hz). Το πρόγραμμα για PWM οδήγηση ανάβει και σβήνει το LED προοδευτικά (dimming).

Μια τιμή *value*=64 (256/64=4) σημαίνει ότι το LED θα μένει αναμμένο το ¼ της περιόδου T και συνεπώς θα βλέπουμε το ¼ της μέγιστης φωτεινότητας. Η συχνότητα είναι περίπου 500 Hz ή T=2 ms.

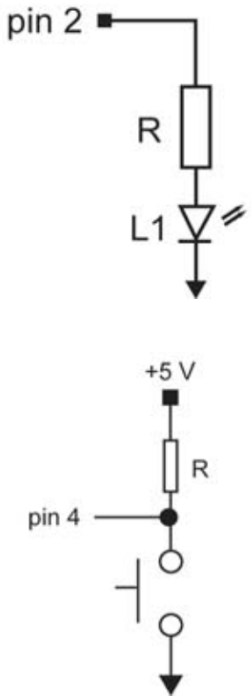
Shield
με αισθητήρια


2N2222
2 X LED
DC Motor
Button
Servo
Reset
LM35D
Φωτοαντίσταση
Proximity
Sensor Shiel
DC MOTOR
SERVO
1N4148
2N2222
2.2 KΩ
LED
LED
220 Ω
10 KΩ
BUTTON
PHOTO-R
IR-Tx
TEM INT IR
RESET
BLUETOOTH


Arduino Proto Shield rev3
Servo
DC motor
1N4001
2.2 KΩ
2N2222
10 KΩ
470 Ω
470 Ω
LED
LM35D
LDR
10 KΩ
10 KΩ
220 Ω
IR
+5V
Gnd

Στατική οδήγηση LED

Σε πέντε πλήρη πατήματα του πλήκτρου το LED θα γίνεται ON/OFF




```
#define led 2
#define btn 4
int ledStatus = HIGH;
int count = 5;

void setup() {
 Serial.begin(9600);          // for serial monitor
 pinMode(led, OUTPUT);
 pinMode(btn, INPUT_PULLUP);
}

void loop() {
 int x=digitalRead(btn);
 while (x==LOW) {
   delay(50);                 // delay 50 ms
   x=digitalRead(btn);
   if (x==HIGH) {
     count=count-1;
     if (count==0) {
       count=5;
       toggleLed();
     }
     break;
   }
 }
}
```

```
void toggleLed() {
 digitalWrite(led, ledStatus);
 ledStatus = (ledStatus == LOW) ? HIGH : LOW;
}
```

(p6)

Αφαιρέστε το delay(50) και εξηγείστε τη λειτουργία του προγράμματος

Παλμική οδήγηση LED

Να γραφτεί πρόγραμμα το οποίο θα ανάβει ένα LED στο pin 3 του Arduino με PWM. Την τιμή PWM θα δίνουμε μέσω της σειριακής οθόνης.

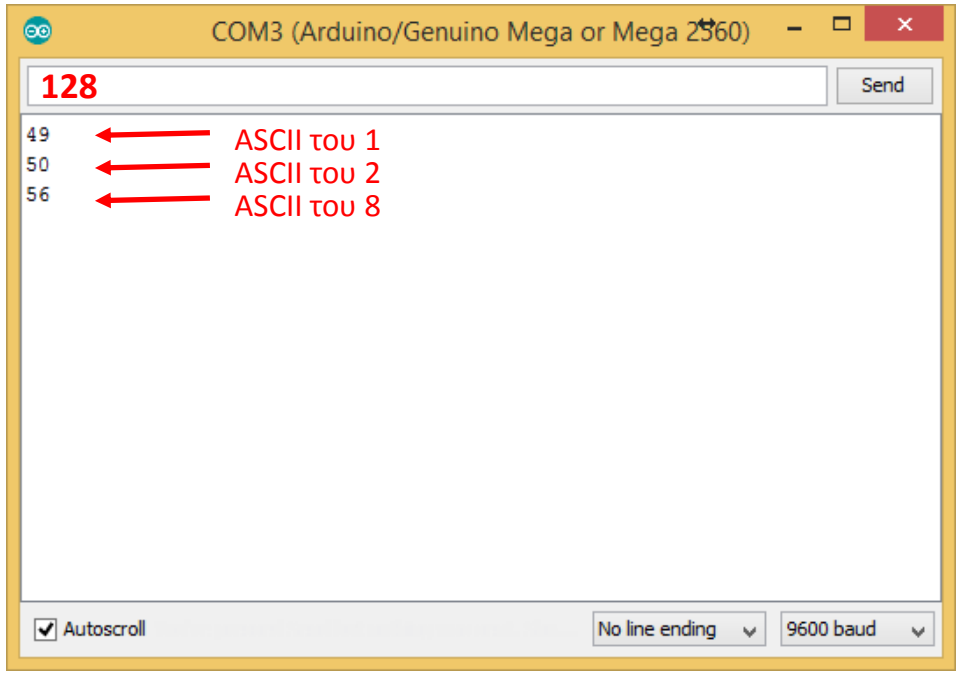
```
int led = 3;
void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  while (Serial.available()>0) {
    int x = Serial.read();
   Serial.println(x);
    analogWrite(led, x);
  }
}
```

Serial.available()

Δεδομένα τα οποία έχουν ήδη αποθηκευτεί στο σειριακό buffer και είναι διαθέσιμα. Μέγιστο 64 bytes.

Serial.read()

Διαβάζει το πρώτο διαθέσιμο σειριακό δεδομένο από τον buffer.



(p7)

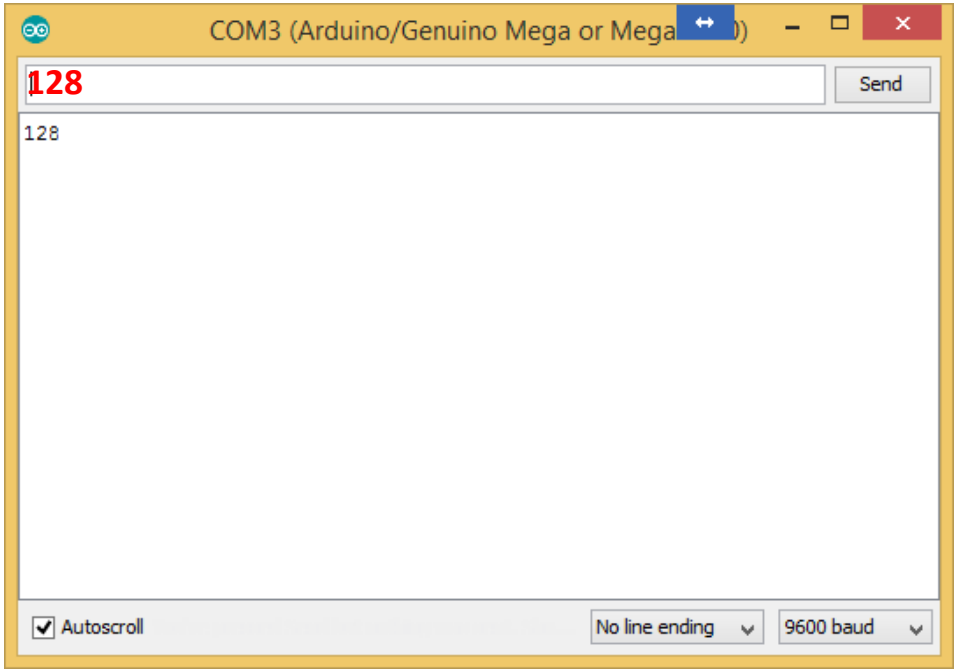
Παλμική οδήγηση LED (Ορθό)

```
int led = 3;
void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  while (Serial.available()>0) {
    int x = Serial.parseInt();
   Serial.println(x);
    analogWrite(led, x);
  }
}
```

(p8)

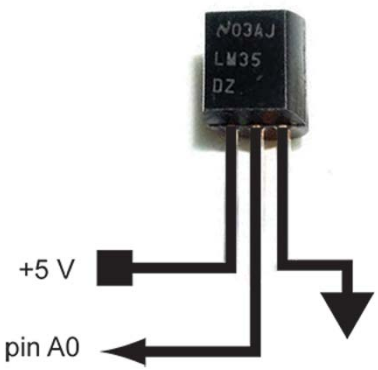
Serial.parseInt()

Διαβάζει το πρώτο ακέραιο σειριακό δεδομένο από τον buffer (long).



Μέτρηση θερμοκρασίας

LM35 D



- LM35 D – 10 mV/^{0}C
- Ανάλυση μετατροπέα (Resolution) = 10 bits
- Τάση αναφοράς = +5 V
- Είσοδος = 0 V – Έξοδος = 000h
- Είσοδος = +5 V – Έξοδος = 3FFh (2^{10}-1=1023)
- Κάθε υποδιαίρεση αντιστοιχεί σε 5000 mV/1023 = **4.9 mV** δηλαδή σε περίπου **0.5 ^{0}C**.

- Για 0 ^{0}C - Είσοδος = 0 V → Έξοδος = 0
- Για 100 ^{0}C – Είσοδος = 1 V → Έξοδος = 1023/5 = 205 ή μεταβολή 2.05 **(~=2**) υποδιαιρέσεων για κάθε βαθμό Κελσίου → **μη εκμετάλλευση όλης της κλίμακας**.

- Εσωτερική τάση αναφοράς = 1.1 V (συνάρτηση analogReference(INTERNAL))
- Κάθε υποδιαίρεση αντιστοιχεί σε 1100 mV/1023 = **1.08 mV** δηλαδή σε περίπου **0.1 ^{0}C**.
- Για 100 0C – Είσοδος = 1 V → Έξοδος = 1023/1.1 = 930 ή μεταβολή **9.3** υποδιαιρέσεων για κάθε βαθμό Κελσίου → **ευαισθησία στο θόρυβο**.

Ανάλυση A/D ή Resolution – 10 bits ή 1024 υποδιαιρέσεις (από 0 έως 1023).

Ανάλυση μέτρησης ή ανάλυση απεικόνισης (Display Resolution) – 0 έως 100 ^{0}C σε μορφή nn.n (ενός δεκαδικού ή 0.1). Π.χ. 32.6 ^{0}C Γι' αυτήν την κλίμακα χρειάζονται 1000 υποδιαιρέσεις, άρα ο A/D είναι κατάλληλος.

Ακρίβεια μέτρησης – Η ακρίβεια της ψηφιακής ένδειξης με την ένδειξη ενός προτύπου οργάνου.

Επαναληψιμότητα ή Repeatability – Η απεικόνιση παραμένει σταθερή σε επαναλήψεις μετρήσεων.

- **Αύξηση της τάσης αναφοράς** → Μεγαλύτερο βήμα, μικρότερη ανάλυση απεικόνισης, πιθανή ανεκμετάλλευτη περιοχή
- **Μείωση της τάσης αναφοράς** → Μικρότερο βήμα, μεγαλύτερη ανάλυση απεικόνισης, ευαισθησία στο θόρυβο.

Αλγόριθμοι μέτρησης θερμοκρασίας

Τάση αναφοράς +5 V

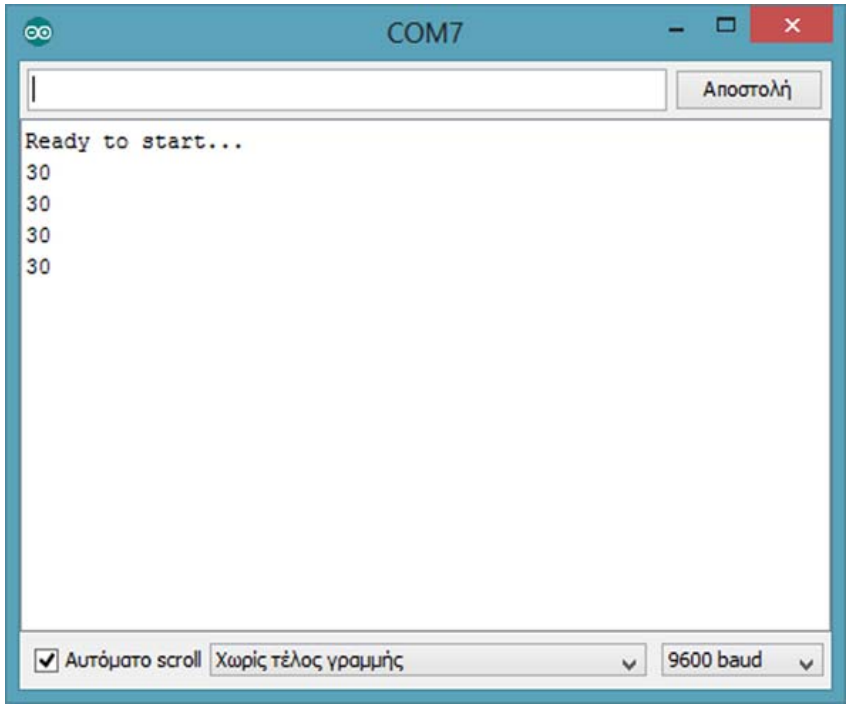
Temperature = Analog Reading from A0/2

Τάση αναφοράς +1.1 V

Temperature = Analog Reading from A0/9.3

Για την αποφυγή του θορύβου που δημιουργεί η γείωση του κυκλώματος δημιουργούμε ένα φίλτρο κινουμένης μέσης τιμής (moving average filter), με το οποίο λαμβάνουμε 20 διαδοχικές μετρήσεις και βρίσκουμε τη μέση τιμή τους.

```
#define pin A0     // (or     int pin=A0;)

int temperature;

void setup() {
  Serial.begin(9600);                    // for serial monitor
  analogReference(INTERNAL);             // reference to 1.1 V
  Serial.println("Ready to start...");
}

void loop() {
  int span = 20;
  int analog = 0;
  for (int i = 0; i < span; i++) {
    analog = analog + analogRead(pin);
  }
  analog = analog / 20;
  temperature = analog/9.3;
  Serial.println(temperature);
}
```

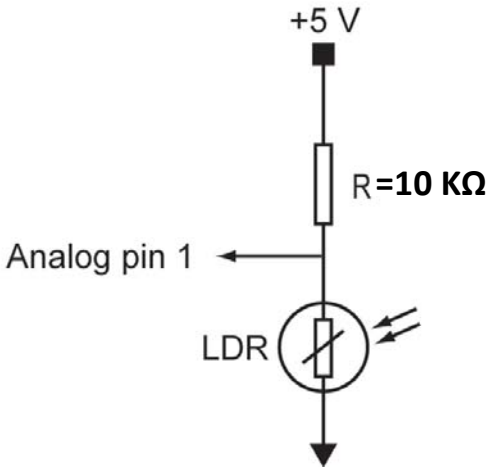


(p9)

Άσκηση

Να γραφτεί πρόγραμμα το οποίο να απεικονίζει τη θερμοκρασία με ακρίβεια 0.1 ^{0}C.

Μέτρηση φωτεινότητας

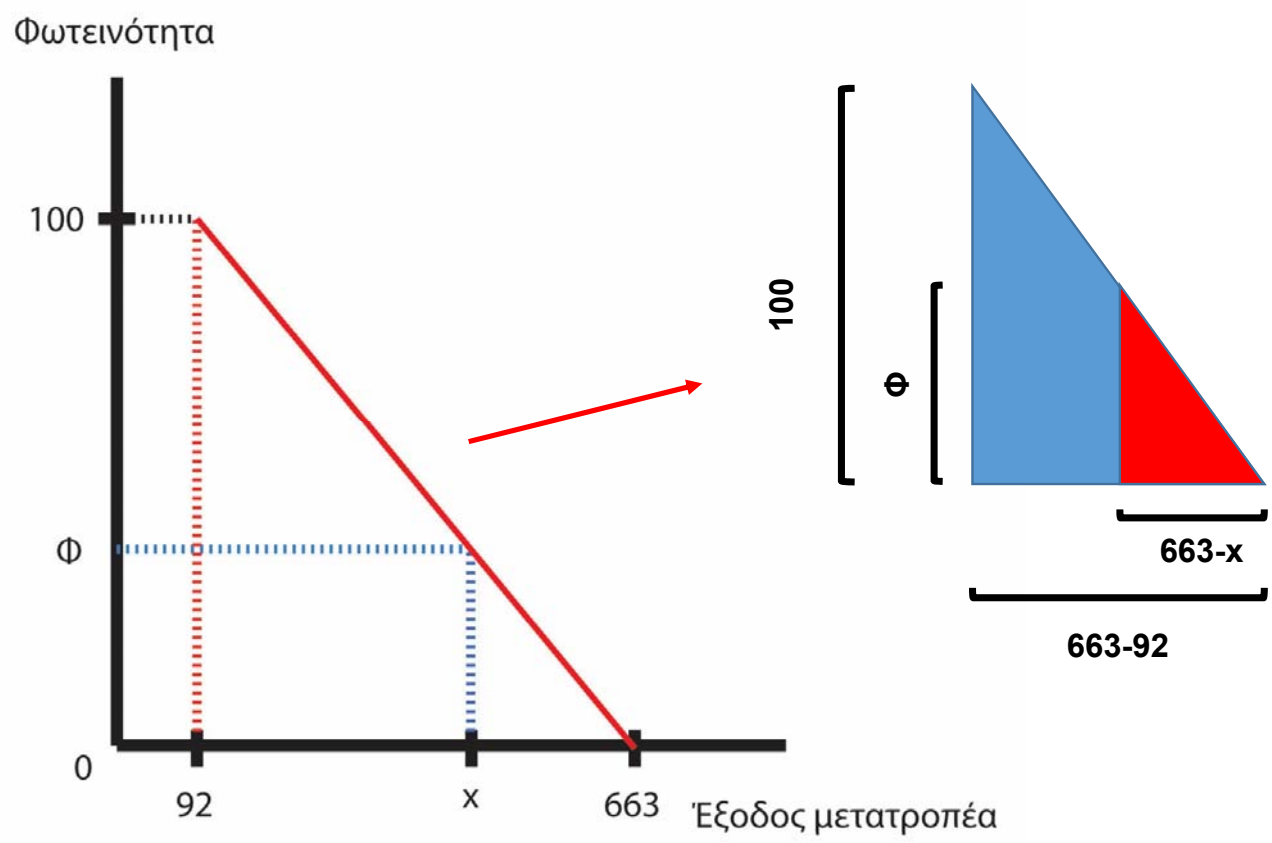


Χαρακτηριστικά φωτο-αντίστασης LDR 5 mm

Αντίσταση φωτός = 8~20 ΚΩ
Αντίσταση σκότους = 1 ΜΩ (min)

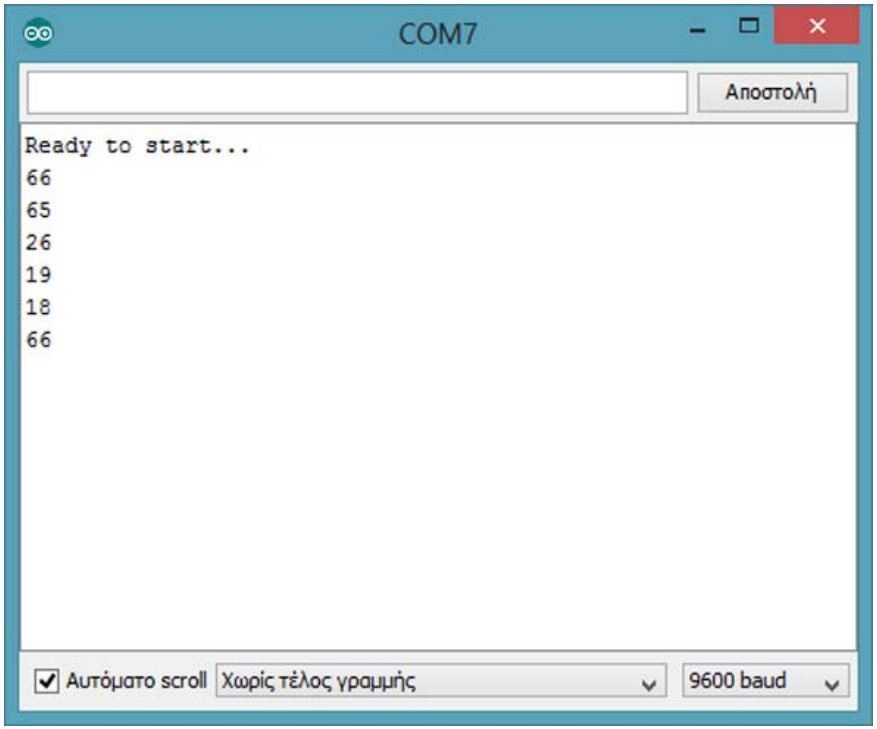
- Σε πλήρη φωτεινότητα, η τάση που εφαρμόζεται στην αναλογική είσοδο του μετατροπέα είναι +0.45 V, ενώ σε πλήρες σκοτάδι η τάση αυτή είναι τα 2/3 των +5 V ή 3.24 V.
- Εάν η τάση αναφοράς του μετατροπέα είναι +5 V, τότε σε πλήρη φωτεινότητα ο μετατροπέας δίνει 92 ενώ σε πλήρες σκοτάδι δίνει 663.
- Για κλίμακα φωτεινότητας από 0 έως 100% και με δεδομένο ότι η ένδειξη του μετατροπέα αυξάνεται όσο το φως γίνεται λιγότερο:

Intensity = 100-(analog reading-92)*100/(663-92)


Φωτεινότητα
100
Φ
0
92
x
663
Έξοδος μετατροπέα
100
Φ
663-x
663-92

(p10)

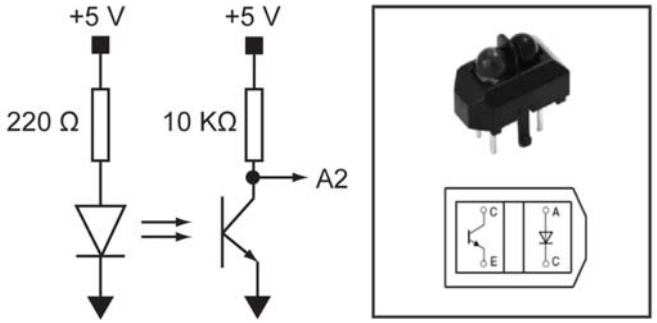
```
#define pin A1     // (or     int pin=A1;)

int intensity;

void setup() {
  Serial.begin(9600);         // for serial monitor
  Serial.println("Ready to start...");
}

void loop() {
  int span = 20;
  double analog = 0;
  for (int i = 0; i < span; i++) {
    analog = analog + analogRead(pin);
  }
  analog = analog / 20;
  intensity = 100-(analog-92)*100/(663-92);
  // intensity = map(analog, 92, 663, 100, 0);
 Serial.println(intensity);
  delay(2000);
}
```

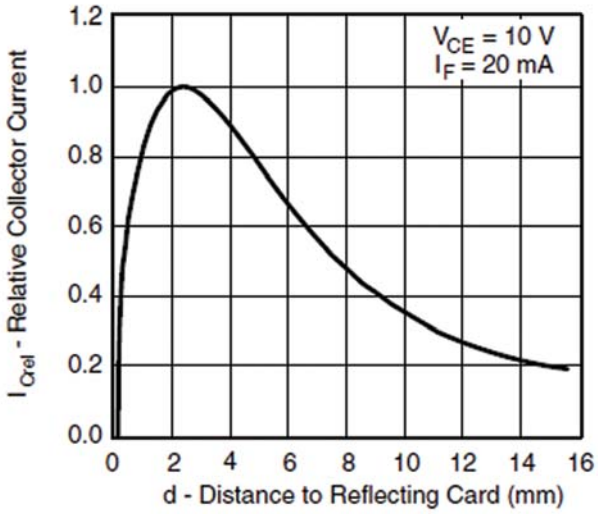


map(value, fromLow, fromHigh, toLow, toHigh);

Η συνάρτηση *map()* μετατρέπει την κλίμακα [92 663] στην κλίμακα [100 0].

Μέτρηση απόστασης





Το αισθητήριο εγγύτητας που χρησιμοποιείται είναι ένας πομπο-δέκτης υπερύθρου, το TCRT5000 στα 950 nm.

- Φωτοδίοδος και φωτο-τρανσίστορ υπερύθρου σε συσκευασία που μπλοκάρει το ορατό φως.
- Μήκος κύματος υπερύθρου 950 nm.
- Απόσταση για μέγιστη μεταφορά ρεύματος Iout/Iin (CTR-Current Transfer Ratio) 2.5 mm.
- Περιοχή λειτουργίας 0.2-15 mm.

- Όταν δεν υπάρχει αντικείμενο πάνω από το αισθητήριο, ο δέκτης δίνει τάση 4.77 V. Όταν πλησιάσουμε μια λευκή επιφάνεια σε απόσταση 3 cm από το αισθητήριο η τάση εξόδου είναι περίπου 4 V, ενώ σε απόσταση 0.5 cm είναι περίπου 0.2 V.
- Επομένως η δυναμική περιοχή λειτουργίας του αισθητηρίου εγγύτητας είναι 3.8 V για μια απόσταση από 0.5 έως 3 cm.
- Με τάση αναφοράς στο μετατροπέα +5 V, τα 0.2 V αντιστοιχούν σε 0.2*1023/5 ή σε 41 υποδιαιρέσεις ενώ τα 4 V σε 818 υποδιαιρέσεις. Αυτό σημαίνει ότι για 2.5 cm ή για 25 mm (πάντοτε μέσα στην περιοχή από 0.5 cm έως 3 cm) ο μετατροπέας θα μεταβάλλεται 777 υποδιαιρέσεις και επομένως 1 mm μετατόπισης του αντικειμένου θα αντιστοιχεί σε μεταβολή 31 υποδιαιρέσεων.

Θεωρούμε ότι αποστάσεις μικρότερες των 5 mm ή μεγαλύτερες των 30 mm είναι εκτός κλίμακας μέτρησης.

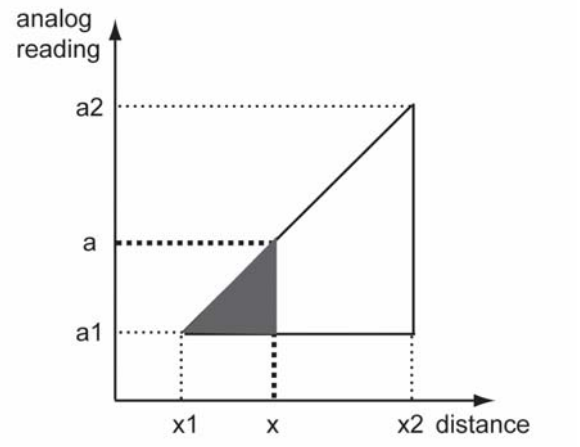
- Σύμφωνα με τα παραπάνω x1=5 mm, x2=30 mm, a1=41, a2=818. Από τα δύο σχηματιζόμενα όμοια τρίγωνα, προκύπτει η σχέση:

$$\frac{x - x1}{x2 - x1} = \frac{a - a1}{a2 - a1}$$

- Επομένως η απόσταση σε mm δίνεται από τη σχέση:

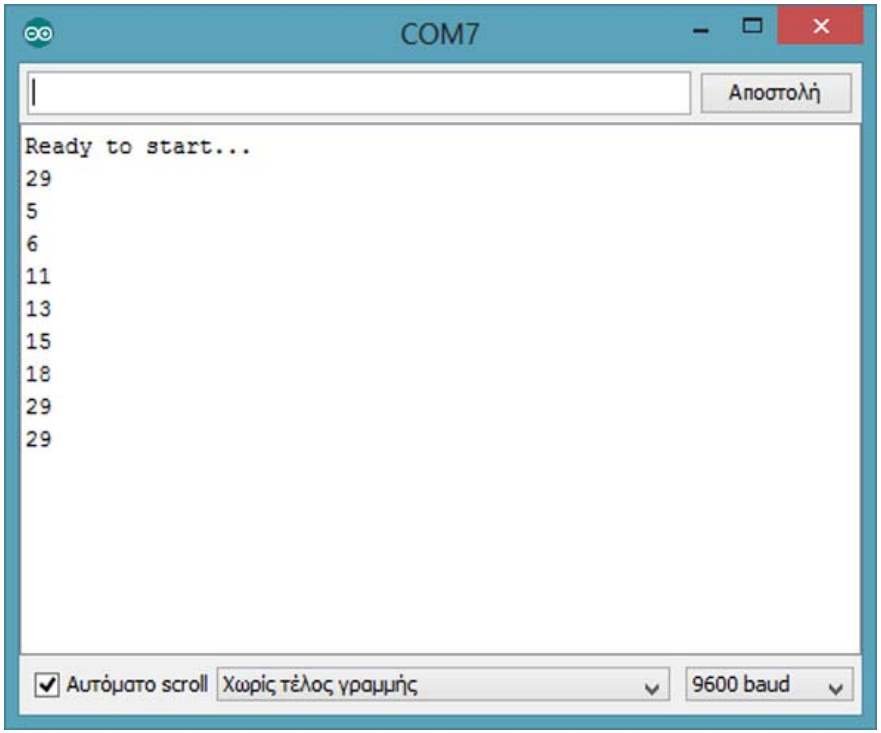
$$x = 5 + \frac{a - 41}{777} \times 25$$

- Για να μην υπάρχουν ψευδείς ενδείξεις, εάν η έξοδος του μετατροπέα είναι μικρότερη του 41 ή μεγαλύτερη του 818, η εφαρμογή θα δείχνει “Out of range”.

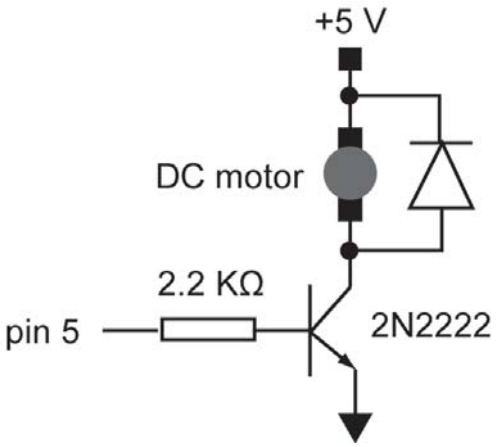


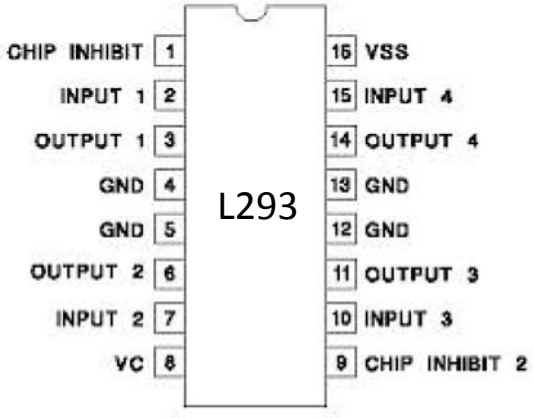
(p11)

```
#define pin A2     // (or     int pin=A2;)

int distance;

void setup() {
  Serial.begin(9600);         // for serial monitor
  Serial.println("Ready to start...");
}

void loop() {
  int span = 20;
  double analog = 0;
  for (int i = 0; i < span; i++) {
    analog = analog + analogRead(pin);
  }
  analog = analog / 20;
  distance = 5 + (analog-41)*25/777;
  Serial.println(distance);
  delay(2000);
}
```



Οδήγηση DC κινητήρα



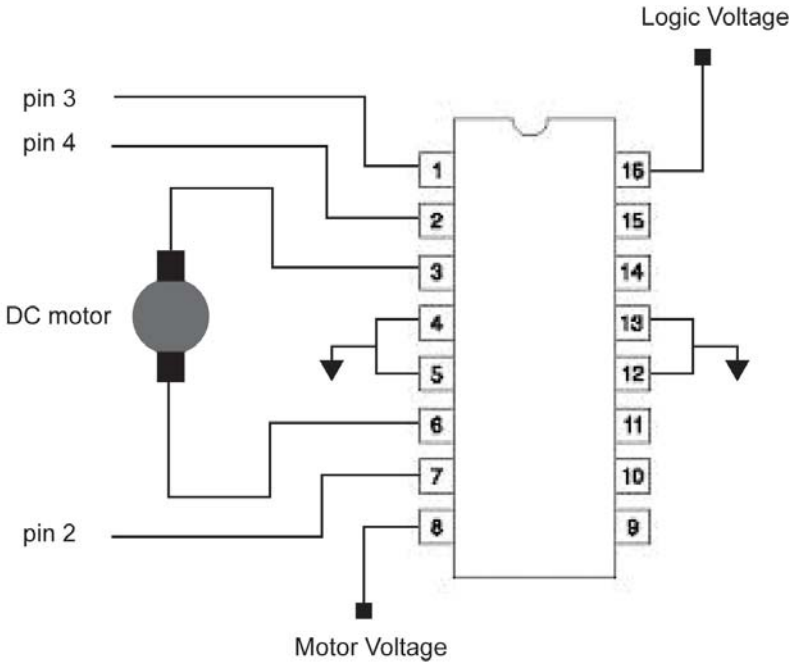


FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

- Τάση λειτουργίας για τους κινητήρες στην ακίδα V_C από 4.5 έως 36 V.
- Ρεύμα λειτουργίας για τους κινητήρες 600 mA.
- Ξεχωριστή τάση λειτουργίας του λογικού κυκλώματος, V_{SS}.
- Έλεγχος των δύο οδηγών από τα CHIP INHIBIT 1 και CHIP INHIBIT 2.



- Δεξιόστροφη κίνηση όταν pin 4 = H και pin 2 = L.
- Αριστερόστροφη κίνηση όταν pin 4 = L και pin 2 = H.
- Η ταχύτητα περιστροφής ελέγχεται με PWM στο pin 3.
- Όταν pin 3 = L, ο κινητήρας σταματά.

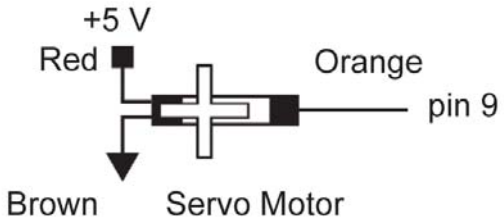
- **Για εξοικονόμηση μιας εξόδου, μπορεί το CHIP INHIBIT 1 να συνδεθεί στα +5 V αλλά τα σήματα ελέγχου της φοράς του κινητήρα πρέπει να συνδεθούν σε εξόδους PWM έτσι ώστε να ελέγχεται η ταχύτητα.**

```
int motor = 5;                  // motor connected to digital pin 5
void setup() {
  Serial.begin(9600);
  pinMode(motor, OUTPUT);
}
 void loop() {
  while (Serial.available()>0) {
    int x = Serial.parseInt();
   Serial.println(x);
    analogWrite(motor,x);
  }
}
```

(p12)

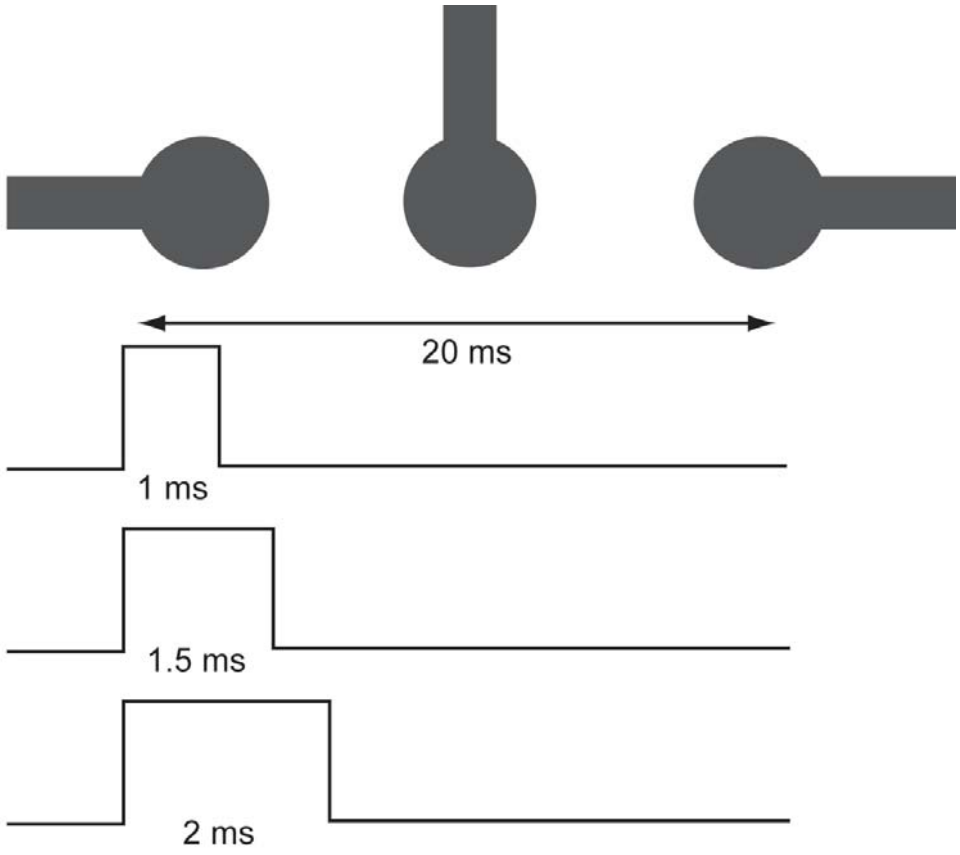
Ρύθμιση ταχύτητας κινητήρα DC με τιμή 0 έως 255 από τη σειριακή οθόνη.

Οδήγηση σερβο-κινητήρα



- Τύπος Tower Pro MG90.
- Τάση λειτουργίας 4.8 έως 6 V.
- Ταχύτητα 0.1 sec/60 degrees (στα 4.8V), 0.08 sec/60 degrees (στα 6.0V).
- Ροπή: Στα 4.8V: 1.8 kg-cm. Στα 6.0V: 2.2 kg-cm.
- Περιστροφή 180^{o}.

Η θέση ενός σερβο-κινητήρα καθορίζεται από το μήκος ενός παλμού HIGH σε ένα χρονικό διάστημα 20 ms. Εάν αυτός ο παλμός είναι HIGH για 1 ms, τότε η γωνία του σέρβο θα είναι μηδέν μοίρες, εάν είναι 1.5 ms θα είναι 90 μοίρες και εάν είναι 2ms θα πάρει θέση στις 180 μοίρες.

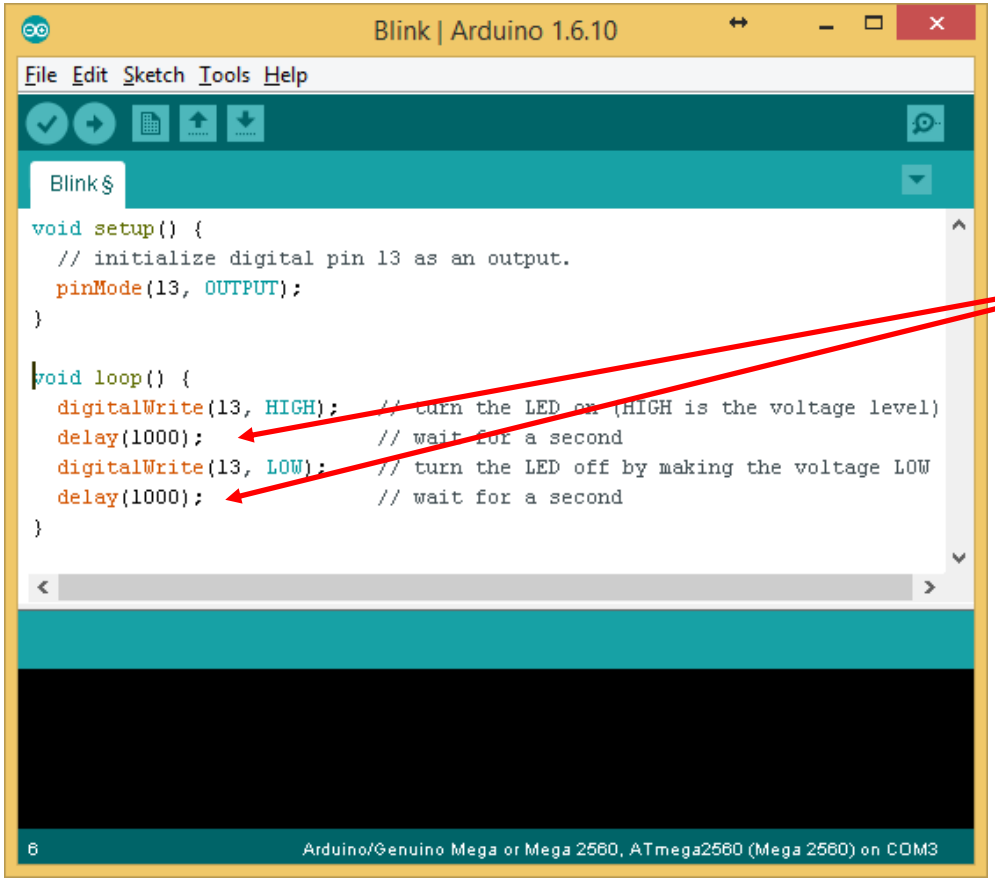

20 ms
1 ms
1.5 ms
2 ms

```
#include <Servo.h>
Servo myservo;      // create servo object to control a servo
int pin = 9;        // servo connected to digital pin 9
int pos = 0;        // angle = 0 degrees
void setup() {
  Serial.begin(9600);
  pinMode(pin, OUTPUT);
  myservo.attach(pin);    // attach servo on pin 9
  myservo.write(pos);     // initialize servo
  delay(15);              // 15 ms to reach position
}
 void loop() {
  while (Serial.available()>0) {
    int x = Serial.parseInt();
   Serial.println(x);
   pos = map(x, 0, 255, 0, 180);
   Serial.println(pos);
    myservo.write(pos);
  }
}
```

- **Ρύθμιση γωνίας servo από 0 έως 180 δίνοντας μια τιμή 0 έως 255 από τη σειριακή οθόνη.**
- **Η εντολή map() μετατρέπει το διάστημα 0-255 στο διάστημα 0-180.**

(p13)

Multi-tasking



Πρόβλημα!

Ο επεξεργαστής δεν μπορεί να κάνει καμιά άλλη δουλειά.

BlinkWithoutDelay | Arduino 1.6.10

File Edit Sketch Tools Help

BlinkWithoutDelay §

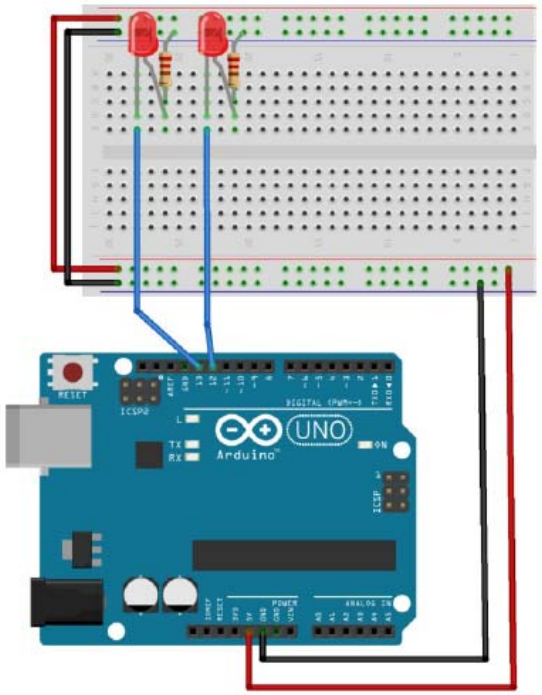
```
// constants won't change. Used here to set a pin number :
const int ledPin =  13;      // the number of the LED pin

// Variables will change :
int ledState = LOW;             // ledState used to set the LED

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;        // will store last time LED was updated

// constants won't change :
const long interval = 1000;           // interval at which to blink (milliseconds)

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

```
void loop() {
  // here is where you'd put code that needs to be running all the time.

  // check to see if it's time to blink the LED; that is, if the
  // difference between the current time and last time you blinked
  // the LED is bigger than the interval at which you want to
  // blink the LED.
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

1 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

Χρήση της millis() για προσδιορισμό μιας χρονικής στιγμής αντί για την χρήση της delay()

Multi-tasking με 2 LED




```
// constants won't change. Used here to set a pin number :
const int ledPin1 =  13;     // the number of the LED1 pin
const int ledPin2 =  12;     // the number of the LED2 pin

// Variables will change :
int ledState1 = LOW;          // ledState used to set the LED1
int ledState2 = LOW;          // ledState used to set the LED2

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis1 = 0;   // will store last time LED1 was updated
unsigned long previousMillis2 = 0;   // will store last time LED1 was updated

// constants won't change :
const long interval1 = 1000;     // interval at which to blink LED1
const long interval2 = 500;      // interval at which to blink LED2

void setup() {
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
}
```

```
void loop() {
  // here is where you'd put code that needs to be running all the time.

  // check to see if it's time to blink the LED1; that is, if the
  // difference between the current time and last time you blinked
  // the LED1 is bigger than the interval at which you want to blink the LED1.
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis1 >= interval1) {
    // save the last time you blinked the LED1
    previousMillis1 = currentMillis;

    // if the LED1 is off turn it on and vice-versa:
    if (ledState1 == LOW) {
      ledState1 = HIGH;
    } else {
      ledState1 = LOW;
    }

    // set the LED1 with the ledState of the variable:
    digitalWrite(ledPin1, ledState1);
  }
```

```
  // check to see if it's time to blink the LED2
    if (currentMillis - previousMillis2 >= interval2) {
      // save the last time you blinked the LED2
      previousMillis2 = currentMillis;

      // if the LED2 is off turn it on and vice-versa:
      if (ledState2 == LOW) {
        ledState2 = HIGH;
      } else {
        ledState2 = LOW;
      }

      // set the LED2 with the ledState of the variable:
      digitalWrite(ledPin2, ledState2);
    }
}
```

(p14)

Multi-tasking with Object Oriented Programming!

1 Δημιουργία της κλάσης “Blink “

```
class Blink {
  // Class Member Variables initialized at startup
  int ledPin;                 // the number of the LED pin
  long interval;              // delay time
  // These maintain the current state
  int ledState;               // ledState used to set the LED
  unsigned long previousMillis; // will store last time LED was updated
```

2 Δημιουργία κατασκευαστή αντικειμένου (Constructor)

```
// Constructor - creates a Blink and initializes the member variables and state
public:
Blink(int pin, long delay) {
  ledPin = pin;
  pinMode(ledPin, OUTPUT);
  interval = delay;
  ledState = LOW;
  previousMillis = 0;
}
```

3 Δημιουργία συνάρτησης στην κλάση (class function)

```
void Update() {
  //record the current time
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState= LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
}
};
```

4 Κυρίως πρόγραμμα

```
// creation of the two led objects with constructor parameters (pin, delay)
Blink led1(12, 500);
Blink led2(13, 1000);

void setup() {
  //initialization is done in the constructor
}

void loop() {
  led1.Update();
  led2.Update();
}
```

Κλάση ή αντικείμενο "Blink"

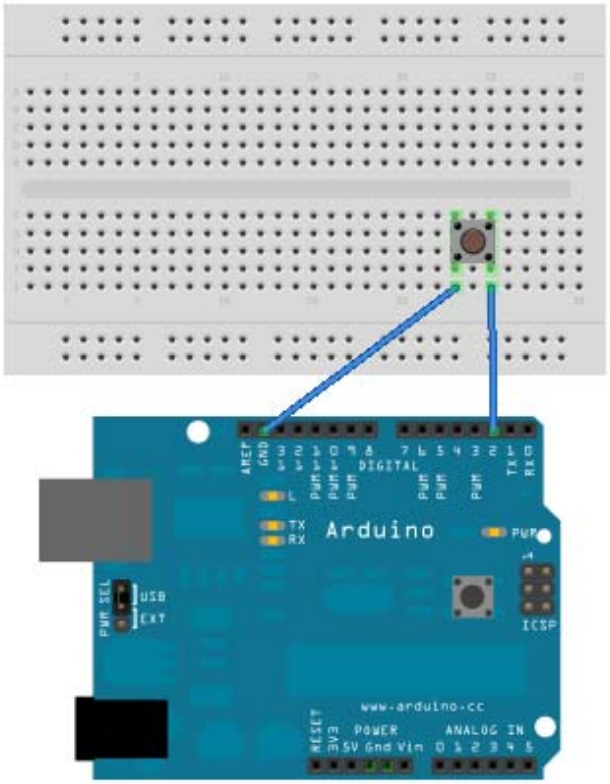
```
class Blink {
  // Class Member Variables initialized at startup
  int ledPin;                 // the number of the LED pin
  long interval;              // delay time
  // These maintain the current state
  int ledState;               // ledState used to set the LED
  unsigned long previousMillis; // will store last time LED was updated

// Constructor - creates a Blink and initializes the member variables and state
public:
Blink(int pin, long delay) {
  ledPin = pin;
  pinMode(ledPin, OUTPUT);
  interval = delay;
  ledState = LOW;
  previousMillis = 0;
}

void Update() {
  //record the current time
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
       ledState = LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
};
```

```
// creation of the two led objects with constructor parameters (pin, delay)
Blink led1(12, 500);
Blink led2(13, 1000);

void setup() {
  //initialization is done in the constructor
}

void loop() {
  led1.Update();
  led2.Update();
}
```

(p15)

Κυρίως πρόγραμμα

External Interrupts



Πηγές Διακοπών στον Arduino

External interrupts	Timer interrupts
INT0 – pin2	Timer 0, 1, 2
INT1 – pin3	
PCINT0 έως PCINT23	

Διαδικασία διακοπής

- Εκτέλεση τρέχουσας εντολής
- Αποκλεισμός άλλων διακοπών
- Αποθήκευση του καταχωρητή προγράμματος στο σωρό
- Το πρόγραμμα θα μεταπηδήσει στην ΥΕΔ (ISR) – Interrupt handler
- Επιστροφή στο κυρίως πρόγραμμα με εντολή return
- Ανάκτηση του καταχωρητή προγράμματος από το σωρό
- Ενεργοποίηση των διακοπών

- Οι διακοπές είναι ενεργοποιημένες by default.
- Οι διακοπές απενεργοποιούνται με την εντολή **noInterrupts()**.
  - Σε αυτήν την περίπτωση μπορούν να επηρεαστούν κάποιες συναρτήσεις και η σειριακή επικοινωνία.
- Οι διακοπές επανενεργοποιούνται με την εντολή **interrupts()**.
  - Σε αυτήν την περίπτωση επηρεάζεται ο χρονισμός.
- Η εξυπηρέτηση μιας διακοπής γίνεται με την εντολή:

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)

ή

attachInterrupt(0 ή 1, ISR, mode)

pin = το pin διακοπής (π.χ. 2 ή 3)

ISR = η υπορουτίνα εξυπηρέτησης της διακοπής

mode = LOW – όποτε το pin είναι 0 V

= CHANGE – όποτε το pin αλλάζει κατάσταση από 0 V σε +5 V ή αντίστροφα

= RISING – όποτε το pin αλλάζει κατάσταση από 0 V σε +5 V

= FALLING - όποτε το pin αλλάζει κατάσταση από +5 V σε 0 V

- Η εξυπηρέτηση διακοπών από συγκεκριμένο pin απενεργοποιείται με την εντολή:

  detachInterrupt(digitalPinToInterrupt(pin))

- Οι μεταβλητές που χρησιμοποιούνται στο κυρίως πρόγραμμα και στην ΥΕΔ δηλώνονται σαν **volatile.** Έτσι ο compiler συγχρονίζει τις τιμές τους.
- Οι συναρτήσεις delay() και millis() δεν λειτουργούν μέσα στην ΥΕΔ.

Παράδειγμα εξωτερικών διακοπών

Χρησιμοποιώντας το παραπάνω κύκλωμα, να γραφτεί πρόγραμμα το οποίο να ανάβει/σβήνει το Led 13 σε κάθε 10 πλήρη πατήματα του πλήκτρου.

```
const int ledPin = 13;
const int btn = 2;
volatile int ledStatus = HIGH;
volatile int count = 10;

void Update() {
  count=count-1;
  if (count==0) {
    count=10;
    digitalWrite(ledPin, ledStatus);
    ledStatus = (ledStatus == LOW) ? HIGH : LOW;
  }
}

void setup() {
  pinMode(btn,INPUT_PULLUP);
  pinMode(ledPin,OUTPUT);
  attachInterrupt(digitalPinToInterrupt(btn), Update, RISING);
}

void loop() {
  }
```

Ο θόρυβος του πλήκτρου δημιουργεί πρόβλημα στο μέτρημα του αριθμού πατημάτων επειδή δεν μπορεί να χρησιμοποιηθεί debounce με καθυστέρηση στην ΥΕΔ.

(p16)

Λύση του προβλήματος

```
const int ledPin = 13;
const int btn = 2;
int ledStatus = HIGH;
int count = 10;
volatile boolean flag = false;

void Update() {
  flag=true;
}

void toggleLed() {
  digitalWrite(ledPin, ledStatus);
  ledStatus = (ledStatus == LOW) ? HIGH : LOW;
}

void setup() {
  pinMode(btn,INPUT_PULLUP);
  pinMode(ledPin,OUTPUT);
  attachInterrupt(digitalPinToInterrupt(btn), Update, FALLING);
}
```

```
void loop() {
  if (flag==true){
    flag=false;
    int x=digitalRead(btn);
    while (x==LOW) {
      delay(50);
      x=digitalRead(btn);
      if (x==HIGH) {
        count=count-1;
        if (count==0) {
          count=10;
          toggleLed();
        }
        break;
      }
    }
  }
}
```

(p17)

- Ο ATMega328 και κατ’ επέκταση ο Arduino διαθέτει δυνατότητα διακοπών στις μεταβολές ενός εκ των παρακάτω pins. Οι αντιστοιχίες με τα pins του Arduino ευρίσκονται στο ATMega pin mapping.

  PCINT[7:0] – PORT B [7:0]
  PCINT[14:8] – PORT C [6:0]
  PCINT[23:16] – PORT D [7:0]

- Οι διακοπές ενεργοποιούνται μέσω του καταχωρητή PCICR για κάθε ομάδα pins.

PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PORTD PORTC PORTB

- Οι είσοδοι κάποιας ομάδας μπορούν να αποκλειστούν (masking) με τον αντίστοιχο καταχωρητή PCMSK0, PCMSK1, PCMSK2. Ο αποκλεισμός γίνεται με εγγραφή του “0” στο αντίστοιχο bit. Η μορφή ενός τέτοιου καταχωρητή είναι:

PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Οι ΥΕΔ που καλούνται όταν υπάρξει διακοπή από κάποια ομάδα είναι:

```
ISR(PCINT0_vect){}   // Port B, PCINT0 - PCINT7
ISR(PCINT1_vect){}   // Port C, PCINT8 - PCINT14
ISR(PCINT2_vect){}   // Port D, PCINT16 - PCINT23
```

- Όταν προκληθεί διακοπή από κάποια ομάδα, το αντίστοιχο flag του καταχωρητή PCIFR γίνεται “1”. Το flag αυτό γίνεται ξανά “0” μόλις εξυπηρετηθεί η διακοπή.

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	-	-	-	-	-	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Άσκηση

Συνδέστε το digital pin 13 με το PCINT0. Κάνετε blink το Led 13 με χρονοκαθυστέρηση 1 s για 10 φορές. Προγραμματίστε το PCINT0 έτσι ώστε στο τέλος να εκτυπώνεται στη σειριακή οθόνη ο αριθμός των εναλλαγών σε αυτό το pin.

Arduino Timers & Timer Interrupts

Αρχή λειτουργίας χρονιστών/μετρητών (Timers/Counters)

- Η βασική λειτουργία ενός timer/counter είναι να μετράει χρόνο σύμφωνα με ένα ρολόϊ το οποίο προκύπτει από την συχνότητα του μικροελεγκτή διαιρεμένη με ένα συντελεστή (prescaler). Για παράδειγμα, σε ένα σύστημα με συχνότητα 16 MHz και prescaler=64, ένας timer των 8 bits μπορεί να μετρήσει μέγιστο χρόνο από 0 έως 255 οπότε και υπερχειλίζει:

t=1/(16.000.000/64) X 256 = 1,024 ms

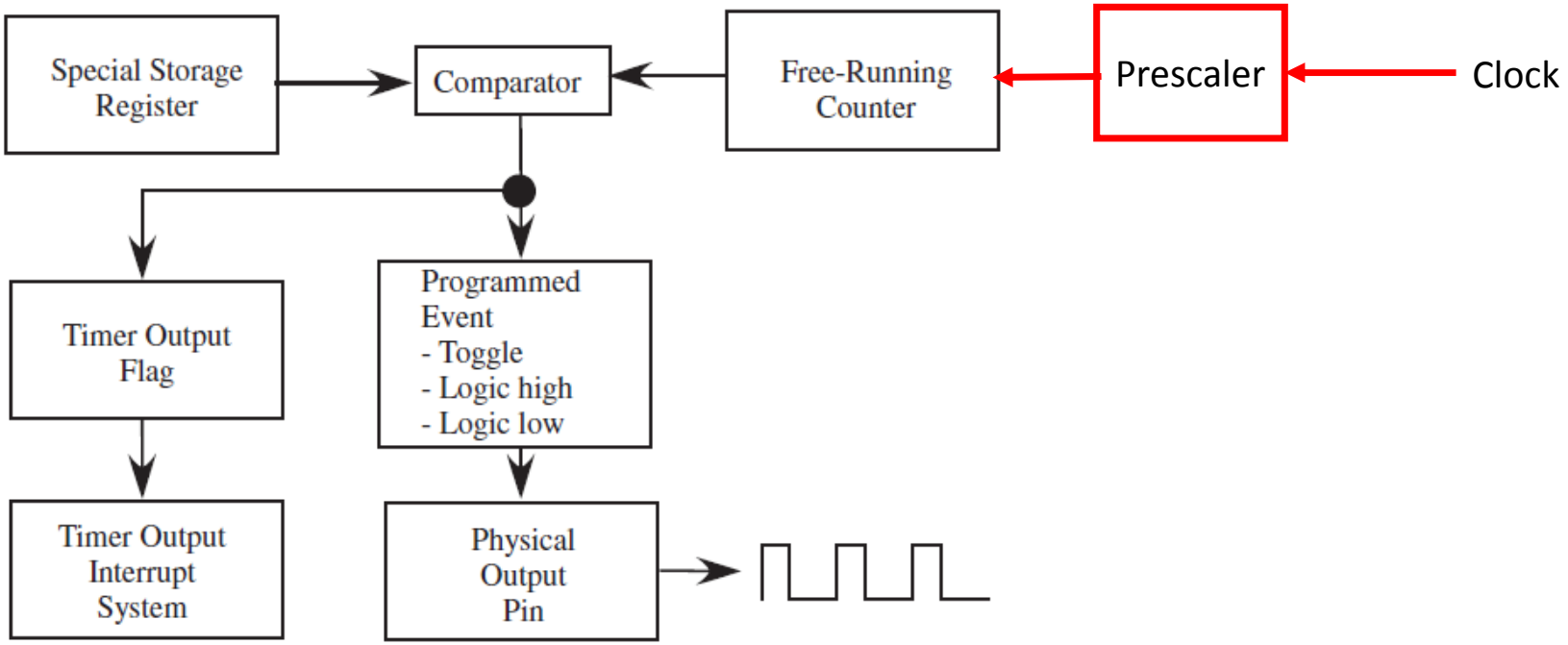
- Εάν ο καταχωρητής του μετρητή (TCNT-Timer Counter) φορτωθεί με μια τιμή από το 0 – 255 τότε ο χρόνος γίνεται μικρότερος. Για παράδειγμα αν TCNT = 100, τότε ο χρόνος μέτρησης μέχρι την υπερχείλιση είναι:

t=1/(16.000.000/64) X (256-100) = 0,624 ms

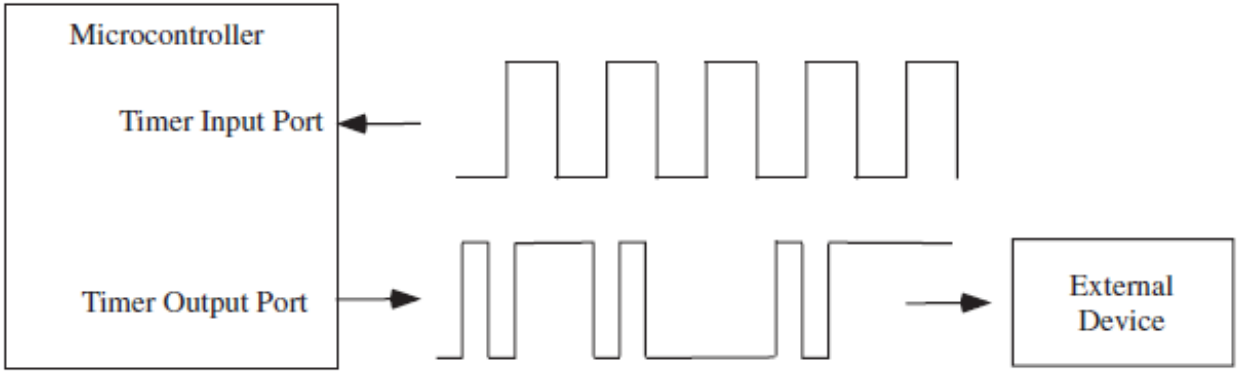
- Σε κάθε υπερχείλιση η σημαία υπερχείλισης (TOV-Timer Overflow) γίνεται 1 και μηδενίζεται για την επόμενη μέτρηση. Στην υπερχείλιση μπορεί να προγραμματιστεί η δημιουργία διακοπής. Επομένως, αν ο μετρητής μας ξεκινήσει από το “100” και υπερχειλίσει 5 φορές, ο μετρούμενος χρόνος θα είναι:

t=1/(16.000.000/64) X (256-100) + 4 X 1/(16.000.000/64) X 256= 4,72 ms

- Εκτός από τη λειτουργία χρονισμού, ένας timer/counter μπορεί να συγκρίνει την τιμή του με μια προ-τοποθετημένη τιμή (OCR-Output Compare Register) και να παράγει μια έξοδο όταν είναι ίση με αυτήν. Φυσικά και σε αυτήν την περίπτωση κάποιο flag τοποθετείται και μπορεί να γίνει κλήση μιας διακοπής.



- Ένας timer/counter μπορεί να έχει είσοδο για να ανιχνεύει παλμούς (capturing) ή να μετράει παλμούς (counting) και στη συνέχεια να παράγει παλμούς σε κάποια έξοδο ή σήματα PWM.



- Ο Arduino διαθέτει 3 χρονιστές (Timers): Timer0, Timer1 και Timer2. Στο multi-tasking χρησιμοποιήθηκε η συνάρτηση millis() για να πραγματοποιηθεί μια καθυστέρηση. Ο Timer0 χρησιμοποιείται για να δημιουργεί μια διακοπή κάθε 1 ms και έτσι ενημερώνεται ο μετρητής της συνάρτησης millis().
- Οι χρονιστές είναι και μετρητές που μετρούν με βάση το ρολόϊ του επεξεργαστή (16 MHz ή 62.5 ns) διαιρεμένο με ένα συντελεστή (clock divisor ή prescaler) και με διάφορους τρόπους (modes).

Timer0:

Timer0 is an 8 bit timer.

Ο timer0 χρησιμοποιείται για τις συναρτήσεις χρόνου delay(), millis() και micros(). Επίσης χρησιμοποιείται για την analogWrite στα pins 5 και 6.

Timer1:

Timer1 is a 16 bit timer.

Ο timer1 χρησιμοποιείται για τη βιβλιοθήκη <Servo.h>. Επίσης χρησιμοποιείται για την analogWrite στα pins 9 και 10.

Timer2:

Timer2 is an 8 bit timer.

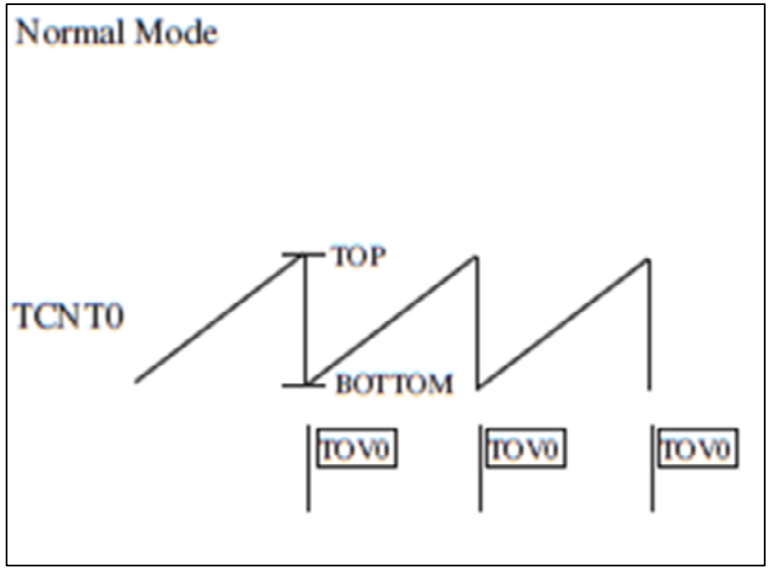
Ο timer2 χρησιμοποιείται για τη συνάρτηση tone(). Επίσης χρησιμοποιείται για την analogWrite στα pins 3 και 11.

Τρόποι λειτουργίας (modes)

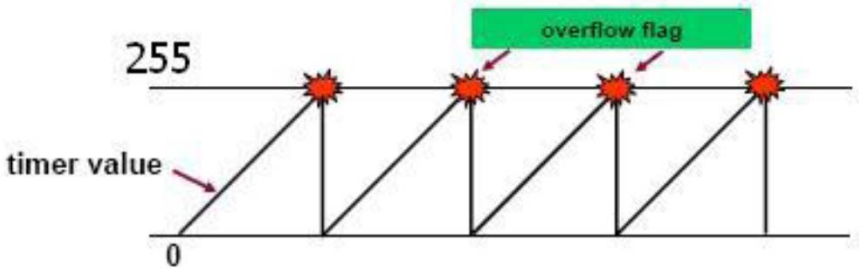
1. **Normal Timer**
2. **Clear Timer on Compare Match (CTC)**
3. **Fast PWM**
4. **Phase Correct PWM**

ΠΙΝΑΚΑΣ ΚΑΤΑΧΩΡΗΤΩΝ TIMERS	
Timer Counter Control Register A (TCCRnA)	Καθορίζει τον τρόπο λειτουργίας (mode)
Timer Counter Control Register B (TCCRnB)	Καθορίζει τον διαιρέτη (prescaler)
Timer Counter Register (TCNTn)	Περιέχει την επιθυμητή μέτρηση (count)
Input Capture Register (ICR1)	Παίρνει την τιμή του TCNT1 σε κάθε γεγονός στο pin ICP1
Output Compare Register A (OCRnA)	Δημιουργία διακοπής στην επιθυμητή μέτρηση
Output Compare Register B (OCRnB)	Δημιουργία διακοπής στην επιθυμητή μέτρηση
Timer/Counter Interrupt Mask Register (TIMSKn)	Συνθήκες δημιουργίας διακοπής
Timer/Counter 0 Interrupt Flag Register (TIFRn)	Ένδειξη δημιουργίας διακοπής

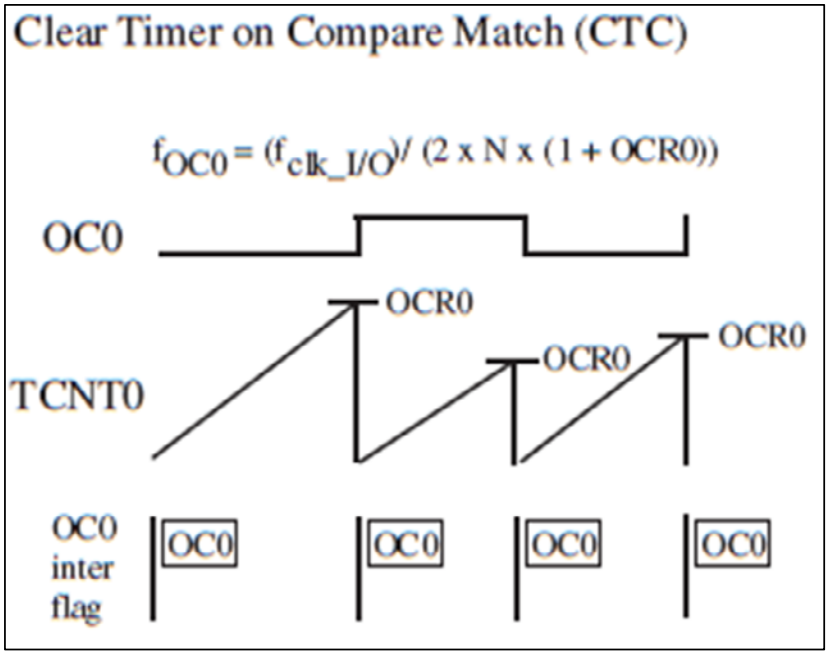
Timer 0	Timer 1	Timer 2
- 8-bit timer/counter	- 16-bit timer/counter	- 8-bit timer/counter
- 10-bit clock prescaler	- 10-bit clock prescaler	- 10-bit clock prescaler
- Functions:	- Functions:	- Functions:
-- Pulse width modulation	-- Pulse width modulation	-- Pulse width modulation
-- Frequency generation	-- Frequency generation	-- Frequency generation
-- Event counter	-- Event counter	-- Event counter
-- Output compare -- 2 ch	-- Output compare -- 2 ch	-- Output compare -- 2 ch
- Modes of operation:	-- Input capture	- Modes of operation:
-- Normal	- Modes of operation:	-- Normal
-- Clear timer on compare match (CTC)	-- Normal	-- Clear timer on compare match (CTC)
-- Fast PWM	-- Clear timer on compare match (CTC)	-- Fast PWM
-- Phase correct PWM	-- Fast PWM	-- Phase correct PWM
	-- Phase correct PWM	

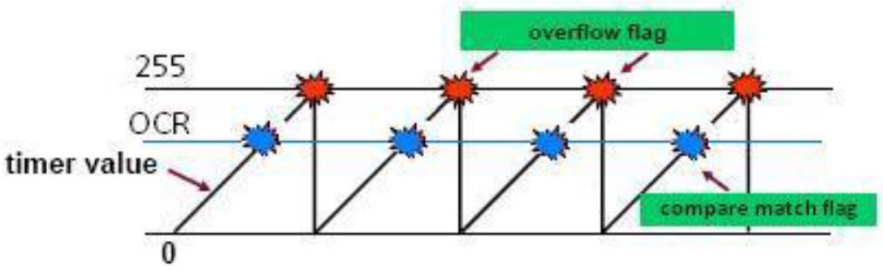


- Ο χρονιστής (π.χ. Timer0) μετράει από την τιμή TCNT0 (min=0x00 ή BOTTOM) στην τιμή 0xFF (TOP).
- Με την επιστροφή του μετρητή στην τιμή BOTTOM, τοποθετείται το flag TOV0.
- Χρήσιμο mode για μέτρηση χρόνου, δημιουργία χρονοκαθυστερήσεων.

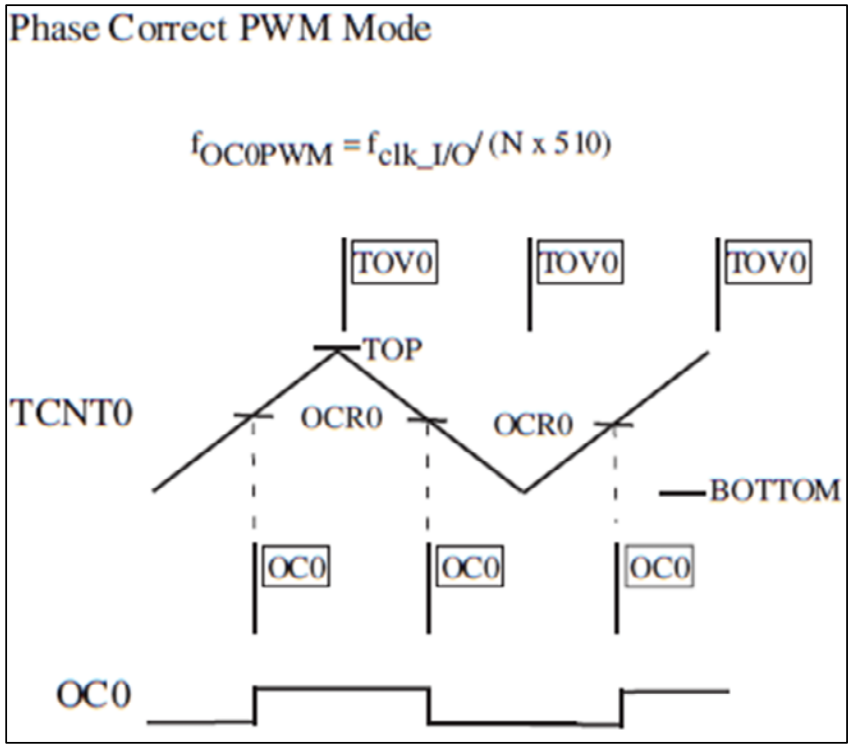


$$f_{timer} = f_{clock} / 256$$

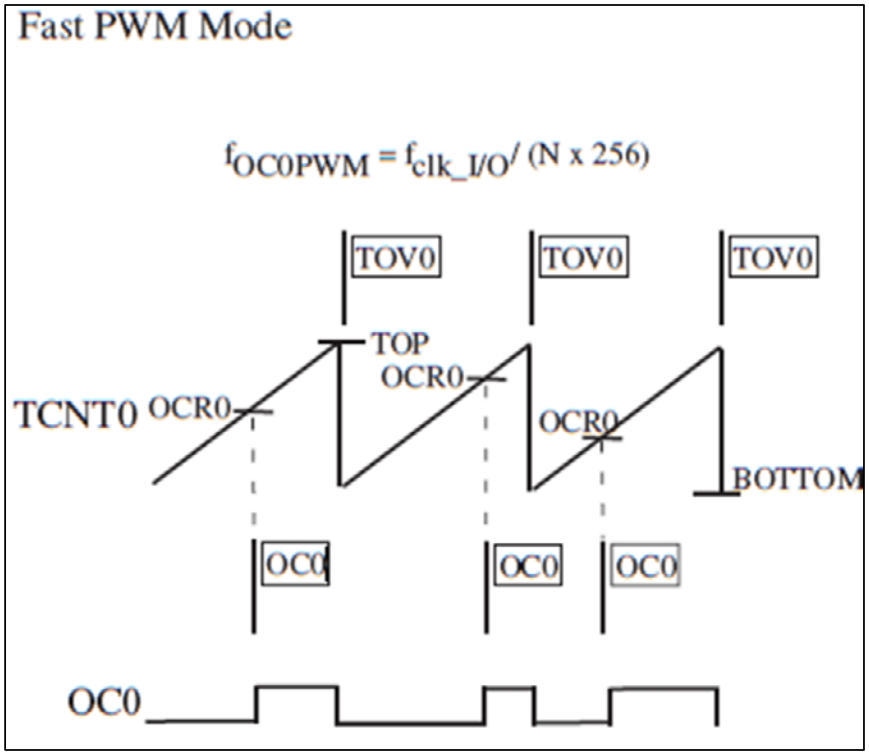




- Ο χρονιστής μηδενίζεται κάθε φορά που ο μετρητής TCNT0 γίνεται ίσος με τον OCR0A ή OCR0B.
- Το flag OCF0A ή OCF0B τοποθετείται.
- Το pin εξόδου OC0 μπορεί να πάρει κάποια τιμή.
- Δημιουργείται διακοπή αν το flag OCIE0A ή OCIE0B στον καταχωρητή TIMSK0 είναι ενεργοποιημένο.
- Χρησιμοποιείται για τη δημιουργία παλμών ή κυματομορφών ακριβείας.



- Ο μετρητής TCNT0 μετράει από BOTTOM σε TOP και στη συνέχεια από TOP σε BOTTOM συνεχώς.
- Κάθε φορά που η τιμή του γίνεται ίση με τον OCR0A ή OCR0B, το flag OCF0A ή OCF0B τοποθετείται και έτσι δημιουργείται στην έξοδο παλμός PWM.



- Χρησιμοποιείται για τη δημιουργία σήματος ακριβείας PWM στην επιθυμητή συχνότητα και με το επιθυμητό Duty Cycle.
- Καλείται Fast PWM επειδή η μέγιστη συχνότητα είναι διπλάσια εκείνης του Phase Correct PWM.
- Όταν ο μετρητής TCNT0 γίνει ίσος με τον OCR0A ή OCR0B, θα προκαλέσει μεταβολή στο σήμα PWM σύμφωνα με τον προγραμματισμό που έχει προηγηθεί. Ο μετρητής TCNT0 συνεχίζει να μετράει μέχρι την τιμή TOP οπότε τοποθετεί το flag υπερχείλισης TOV0.

Καταχωρητές του Timer0

Timer/Counter Control Register A (TCCR0A)

COM0A1	COM0A0	COM0B1	COM0B0	---	---	WGM01	WGM00
7							0

Timer/Counter Control Register B (TCCR0B)

FOC0A	FOC0B	---	---	WGM02	CS02	CS01	CS00
7							0

Timer/Counter Register (TCNT0)

7							0

Output Compare Register A (OCR0A)

7							0

Output Compare Register B (OCR0B)

7							0

Μετρητές

Timer/Counter Interrupt Mask Register 0 (TIMSK0)

---	---	---	---	---	OCIE0B	OCIE0A	TOIE0
7							0

Ενεργοποίηση διακοπών όταν TCNT0=OCR0A ή OCR0B

Ενεργοποίηση διακοπών σαν Timer

Timer/Counter Interrupt Flag REgister 0 (TIFR0)

---	---	---	---	---	OCF0B	OCF0A	TOV0
7							0

Flag όταν TCNT0=OCR0A ή OCR0B

Flag υπερχείλισης σαν Timer

Prescaler

Καταχωρητές TCCR0A και TCCR0B του Timer0

CS0[2:0]	Clock Source
000	None
001	$clk_{I/O}$
010	$clk_{I/O}/8$
011	$clk_{I/O}/64$
100	$clk_{I/O}/8clk_{I/O}/256$
101	$clk_{I/O}/8clk_{I/O}/1024$
110	External clock on T0 (falling edge trigger)
111	External clock on T1 (rising edge trigger)

Clock Select

Timer/Counter Control Register B (TCCR0B)

FOC0A	FOC0B	---	---	WGM02	CS02	CS01	CS00
7							0

Timer/Counter Control Register A (TCCR0A)

COM0A1	COM0A0	COM0B1	COM0B0	---	---	WGM01	WGM00

Waveform Generation Mode

Mode	WGM[02:00]	Mode
0	000	Normal
1	001	PWM, Phase Correct
2	010	CTC
3	011	Fast PWM
4	100	Reserved
5	101	PWM, Phase Correct
6	110	Reserved
7	111	Fast PWM

Compare Output Mode, non-PWM Mode

COM0A[1:0]	Description
00	Normal, OC0A disconnected
01	Toggle OC0A on compare match
10	Clear OC0A on compare match
11	Set OC0A on compare match

Compare Output Mode, Fast PWM Mode

COM0A[1:0]	Description
00	Normal, OC0A disconnected
01	WGM02 = 0: normal operation, OC0A disconnected WGM02 = 1: Toggle OC0A on compare match
10	Clear OC0A on compare match, set OC0A at Bottom (non-inverting mode)
11	Set OC0A on compare match, clear OC0A at Bottom (inverting mode)

Compare Output Mode, Phase Correct PWM

COM0A[1:0]	Description
00	Normal, OC0A disconnected
01	WGM02 = 0: normal operation, OC0A disconnected WGM02 = 1: Toggle OC0A on compare match
10	Clear OC0A on compare match, when upcounting. Set OC0A on compare match when down counting
11	Set OC0A on compare match, when upcounting. Set OC0A on compare match when down counting

Compare Output Mode, non-PWM Mode

COM0B[1:0]	Description
00	Normal, OC0B disconnected
01	Toggle OC0B on compare match
10	Clear OC0B on compare match
11	Set OC0B on compare match

Compare Output Mode, Fast PWM Mode

COM0B[1:0]	Description
00	Normal, OC0B disconnected
01	Reserved
10	Clear OC0B on compare match, set OC0B at Bottom (non-inverting mode)
11	Set OC0B on compare match, clear OC0B at Bottom (inverting mode)

Compare Output Mode, Phase Correct PWM

COM0B[1:0]	Description
00	Normal, OC0B disconnected
01	Reserved
10	Clear OC0B on compare match, when upcounting. Set OC0B on compare match when down counting
11	Set OC0B on compare match, when upcounting. Set OC0B on compare match when down counting

Παράδειγμα με τον Timer1 σε λειτουργία χρονιστή (Normal mode)

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Overflow Interrupt Enable: Όταν TOIE1=1 δημιουργείται διακοπή μόλις ο χρονιστής ή μετρητής υπερχειλίσει.

Να γραφτεί πρόγραμμα το οποίο να αναβοσβήνει το Led 13 κάθε φορά που υπερχειλίζει ο χρονιστής Timer1. Να επιλεγεί prescaler 256.

- Ο prescaler 256 επιλέγεται αν στον καταχωρητή TCCR1B τοποθετηθεί τιμή 00000100 ή 0x4. Αυτό σημαίνει ότι η συχνότητα λειτουργίας του χρονιστή θα είναι 16 MHz/256 = 62.5 KHz ή θα μειώνεται κάθε 1/62500 = 16 μs. Επομένως το Led 13 θα αλλάζει κατάσταση κάθε 65536 * 16 μs ~= 1 s (Ο Timer1 είναι 16 bits και επομένως μπορεί να μετρήσει από 0 έως 65535). Για την ακρίβεια είναι 1.048576 s.
- Ο καταχωρητής TCCR1A τοποθετείται σε μηδενική τιμή έτσι ώστε το mode να είναι 0, δηλ. Normal.
- Το bit TOIE1 του καταχωρητή TIMSK1 γίνεται 1 για να μπορεί να ενεργοποιήσει διακοπή στην υπερχείλιση.
- Μετά την υπερχείλιση καλείται η ΥΕΔ “ISR(TIMER1_OVF_vect)”.
- Ο προγραμματισμός των χρονιστών γίνεται με τις διακοπές απενεργοποιημένες.

```
#include <avr/io.h>   // AVR microcontrollers library

const int ledPin = 13;

void setup() {
  pinMode(ledPin,OUTPUT);
  // Timer1 initialization
  noInterrupts();   //disable interrupts. Also with cli().
  // mode=NORMAL
  TCCR1A = 0;
  TCCR1B = 0;
  // set CS12 bit of TCCR1B (prescaler=256)
  TCCR1B |= (1 << CS12);  // shift 00000001 to CS12 position
                          // equivalent: TCCR1B = _BV(CS12) --> 1<<CS12
                          // equivalent: bitSet(TCCR1B,CS12)
  // enable TOIE1 of TIMSK1
  bitSet(TIMSK1, TOIE1);
  interrupts();     //enable interrupts. Also with sei().
}

void loop() {

}

ISR(TIMER1_OVF_vect) {
  digitalWrite(ledPin, !digitalRead(ledPin));
  // or PORTB ^= _BV(PB5); --> PB5=pin 19 or digital pin 13
}
```

- Το πρόγραμμα είναι ισοδύναμο με το Blink το οποίο χρησιμοποιεί την χρονοκαθυστέρηση delay(1000), δηλαδή 1 s.
- **Τι πρέπει να γίνει ώστε αυτή η χρονοκαθυστέρηση να γίνει περίπου 4 s?**

(p18)

Στο προηγούμενο παράδειγμα, με CS12=1 ή με prescaler=256, ο χρόνος υπερχείλισης ήταν 1,048576 s. Τι μπορούμε να κάνουμε ώστε ο χρόνος αυτός να γίνει ακριβώς 1 s?

- Είναι φανερό ότι αν ο Timer1 ξεκίναγε να μετράει από μια μεγαλύτερη τιμή και όχι από το 0, τότε η χρονοκαθυστέρηση θα ήταν μικρότερη επειδή θα μετρούσε λιγότερα βήματα από 65536. Επομένως για χρονοκαθυστέρηση 1 s ο Timer1 πρέπει να μετρήσει 65536/1.048576 = 62500 βήματα.
- Άρα η τιμή εκκίνησής του δεν θα είναι το 0 αλλά το 65535-62500 = 3035 ή το 0x0BDC.
- Η τιμή εκκίνησης του Timer1 αποθηκεύεται στον καταχωρητή TCNT1 με την εντολή: TCNT1 = 0x0BDC;
- Στο παραπάνω πρόγραμμα θα τοποθετηθεί στο setup() και στην ΥΕΔ.

Παράδειγμα με τον Timer1 σε λειτουργία CTC

- Η λειτουργία αυτή λέγεται CTC (Clear Timer on Compare match). Ο χρονιστής αντί να υπερχειλίζει, συγκρίνει κάθε φορά την τιμή του με την τιμή ενός άλλου προ-φορτωμένου καταχωρητή (OCR1A). Όταν οι δύο τιμές είναι ίδιες, τότε τοποθετείται κάποιο flag ή δημιουργείται διακοπή.

- Χρονοκαθυστέρηση = (Τιμή preset+1)/(16 MHz / prescaler). Το +1 τοποθετείται στη σχέση υπολογισμού επειδή όταν ο χρονιστής φτάσει στην επιθυμητή τιμή του άλλου καταχωρητή, μηδενίζεται δηλαδή κάνει ένα κύκλο παραπάνω.

- Επομένως για να δημιουργηθεί χρονοκαθυστέρηση ακριβείας 1 s πρέπει να διαλέξουμε prescaler=256 και να φορτώσουμε τον καταχωρητή OCR1A με την τιμή: Τιμή preset = Χρονοκαθυστέρηση * (16 MHz/256) -1, δηλαδή με την τιμή 62499.

- Σε αυτήν την περίπτωση ο χρονιστής Timer1 τοποθετείται σε CTC mode (WGM12=1 του TCCR1B) και ενεργοποιείται το αντίστοιχο interrupt flag (OCIE1A=1 του TIMSK1).

- Η ΥΕΔ σε αυτό το mode λειτουργίας είναι η ISR(TIMER1_COMPA_vect).

Να γραφτεί πρόγραμμα το οποίο να αναβοσβήνει το Led 13 κάθε 10 s χρησιμοποιώντας το CTC mode του χρονιστή Timer1.

```
#include <avr/io.h>   // AVR microcontrollers library

const int ledPin = 13;
volatile int seconds=0;

void setup() {
  pinMode(ledPin,OUTPUT);
  // Timer1 initialization
  noInterrupts();   //disable interrupts. Also with cli().
  // iniialize TCCR1A and TCCR1B
  TCCR1A = 0;
  TCCR1B = 0;
  // set compare match register to desired value for 1 s delay
  OCR1A = 62499;
  // CTC mode
  bitSet(TCCR1B,WGM12);     // or TCCR1B |= (1<<WGM12);
  // set CS12 bit of TCCR1B (prescaler=256)
  bitSet(TCCR1B,CS12);
  //TCCR1B |= (1 << CS12);       // shift 00000001 to CS12 position
  // or TCCR1B = _BV(CS12) --> 1<<CS12
 // enable OCIE1A of TIMSK1
  bitSet(TIMSK1,OCIE1A);
  interrupts();     //enable interrupts. Also with sei().
}
```

```
void loop() {

}

ISR(TIMER1_COMPA_vect) {
  seconds++;
  if(seconds == 10){
    seconds = 0;
    digitalWrite(ledPin, !digitalRead(ledPin));
  }
}
```

Με αυτόν τον τρόπο μπορούμε να δημιουργήσουμε μεγάλες χρονοκαθυστερήσεις.

(p19)

Να γραφτεί πρόγραμμα το οποίο να μετρά εξωτερικούς παλμούς στην είσοδο T1

- Το pin PD5 του ATMega328 μπορεί να λειτουργήσει σαν είσοδος T1 και αντιστοιχεί στο digital pin 5 του Arduino Uno.
- Εάν TCCR1A=0 και TCCR1B=7 (CS12, CS11, CS10=1), η είσοδος T1 μπορεί να διεγείρεται σε ανόδους παλμών. Σε κάθε άνοδο παλμού το περιεχόμενο του TCNT1 αυξάνεται κατά 1.
- Συνδέουμε το digital pin 13 με το digital pin 5 και κάνουμε το Led 13 ON/OFF 10 φορές με χρονοκαθυστέρηση 1 s.
- Απεικονίζουμε το περιεχόμενο του TCNT1 στη σειριακή οθόνη.

```
#include <avr/io.h>

const int ledPin = 13;
int count=0;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin,OUTPUT);
  // Reset Timer1
  TCCR1A=0;
  TCCR1B=0;
  // Program CS10, CS11, CS12=1 for rising edge
  bitSet(TCCR1B,CS10);
  bitSet(TCCR1B,CS11);
  bitSet(TCCR1B,CS12);
}
```

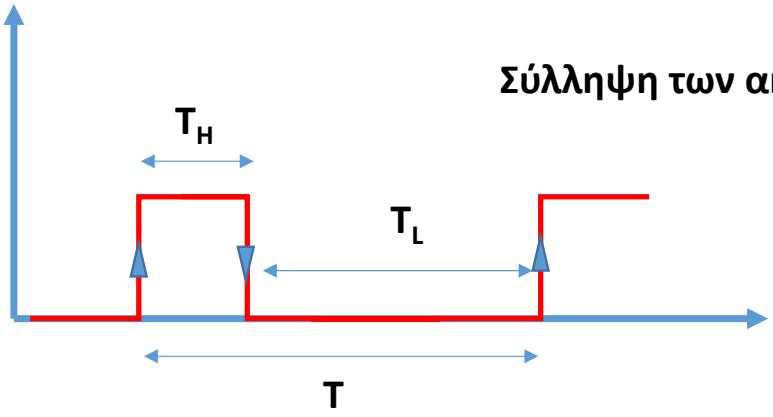
```
void loop() {
  for (int i=0;i<10;i++) {
    digitalWrite(ledPin,HIGH);
    delay(1000);
    digitalWrite(ledPin,LOW);
    delay(1000);
  }
  count=TCNT1;
  TCNT1=0;
  Serial.println(count);
}
```

(p20)

Input Capture Unit

- Το digital pin 8 μπορεί να λειτουργήσει και σαν ICP1 (Input Capture του Timer1).
- Σε αυτήν τη λειτουργία, ο ICR1 (Input Capture Register 1) φορτώνεται με την τιμή του TCNT1 όταν στον ακροδέκτη ICP1 εμφανιστεί η επιθυμητή ακμή.
- Η λειτουργία αυτή χρησιμοποιείται για τη μέτρηση της διάρκειας παλμών, συχνοτήτων, PWM, κλπ.

Να υπολογιστεί η διάρκεια των ON και OFF ενός PWM παλμού στην είσοδο ICP1



- Συνδέουμε το digital pin 13 με το digital pin 8 στον Arduino Uno και δημιουργούμε μια περίοδο με T_H=100 ms και T_L=200 ms.
- Ενεργοποιούμε τον Timer1, τη μονάδα ICP και τις διακοπές που τα αφορούν.
- Ανιχνεύουμε την ακμή ανόδου και αποθηκεύουμε την τιμή του ICR1 (capt1).
- Ανιχνεύουμε την ακμή καθόδου, αποθηκεύουμε τη νέα τιμή του ICR1 (capt2) και τις φορές που ο Timer1 έχει υπερχειλίσει (tov_1).
- Ανιχνεύουμε την ακμή ανόδου, αποθηκεύουμε τη νέα τιμή του ICR1 (capt3) και τις φορές που ο Timer1 έχει υπερχειλίσει (tov_2).
- T_H = (capt2-capt1)+tov_1* 65536 και T_L = (capt3-capt2)+tov_2* 65536
- Επειδή δεν χρησιμοποιείται prescaler, ο καταχωρητής TCNT1 υπερχειλίζει αφού μετρήσει 65536 περιόδους ρολογιού T (T=1/f=1/16 MHz).

- **Το flag ICES1 του TCCR1B καθορίζει την ακμή με την οποία διεγείρεται η είσοδος ICP1. Εάν ICES1=0 η είσοδος διεγείρεται με πτώση παλμού ενώ εάν ICES1=1 διεγείρεται με άνοδο παλμού.**
- **Η διέγερση τοποθετεί το αντίστοιχο flag ICF1 και δημιουργεί διακοπή αν το flag ICIE1 είναι ενεργοποιημένο στον TIMSK1.**
- **Η υπερχείλιση δημιουργεί διακοπή αν το flag TOIE1 είναι ενεργοποιημένο στον TIMSK1.**

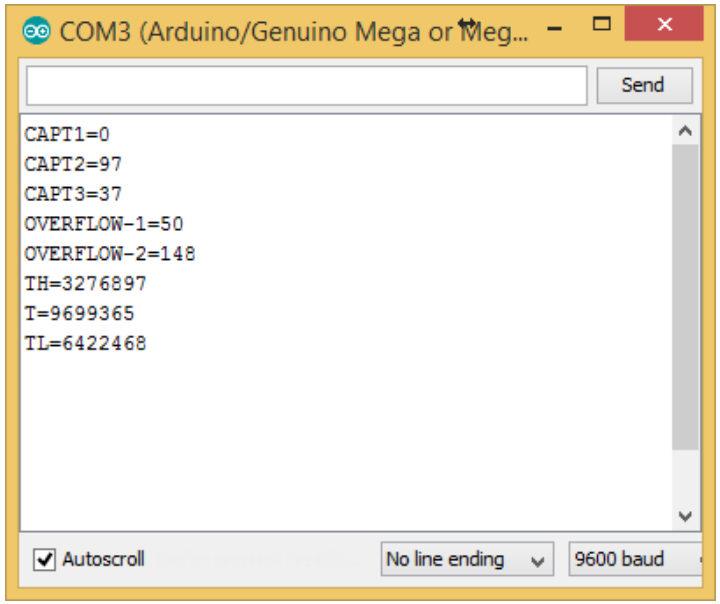
```
#include <avr/io.h>
const int ledPin = 13;
const long interval1 = 100;
const long interval2 = 200;
volatile unsigned long tov_1, tov_2; //overflows counters
volatile unsigned long capt1, capt2, capt3;  //TCNT1 values for each capture
volatile int flag=0;     //defines the edge

void myDelay(long interval) {
  unsigned long previousMillis = millis();
  while (millis() - previousMillis < interval) {
  }
}

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin,LOW);
  noInterrupts();
  //Set Initial Timer value
  TCNT1=0;
  //Enable first capture on rising edge
  bitSet(TCCR1B,ICES1);
  //enable capture and overflow interrupts
  bitSet(TIMSK1,ICIE1);
  bitSet(TIMSK1,TOIE1);
  //Start timer without prescaller
  bitSet(TCCR1B,CS10);
  interrupts();
```

```
 digitalWrite(ledPin,HIGH);
  myDelay(interval1);
  digitalWrite(ledPin,LOW);
  myDelay(interval2);
  digitalWrite(ledPin,HIGH);
  while(1) {
    //calculate TH and TL
    if (flag==3){
      long th = capt2-capt1+tov_1*65536;
      long t = capt3-capt1+tov_2*65536;
      long tl=t-th;
      Serial.print("CAPT1=");      Serial.println(capt1);
      Serial.print("CAPT2=");      Serial.println(capt2);
      Serial.print("CAPT3=");      Serial.println(capt3);
      Serial.print("OVERFLOW-1="); Serial.println(tov_1);
      Serial.print("OVERFLOW-2="); Serial.println(tov_2);
      Serial.print("TH=");       Serial.println(th);
      Serial.print("T=");      Serial.println(t);
      Serial.print("TL=");       Serial.println(tl);
      //clear flag
      flag=0;
      //clear interrupt flags to avoid any pending interrupts
      bitClear(TIFR1,ICF1);
      bitClear(TIFR1,TOV1);
    }
  }
}
```

```
void loop() { }
//capture the edges
ISR(TIMER1_CAPT_vect) {
  if (flag==0){
    //save captured timestamp
    capt1=ICR1;
    //reset overflows
    tov_2=0;
    //enable capture on falling edge
    bitClear(TCCR1B,ICES1);
  }
  if (flag==1){
    capt2=ICR1;
    //save first overflow counter
    tov_1=tov_2;
    //change capture on rising edge
    bitSet(TCCR1B,ICES1);
}
  if (flag==2){
    capt3=ICR1;
    //stop input capture and overflow interrupts
    bitClear(TIMSK1,ICIE1);
    bitClear(TIMSK1,TOIE1);
  }
  //increment Flag
  flag++;
}
```

```
//Overflow ISR
ISR(TIMER1_OVF_vect){
  //increment overflow counter
  tov_2++;
}
```

(p21)



Να υπολογιστούν οι χρόνοι HIGH και LOW.

Βιβλιοθήκες για τους Timers/Counters

Timer0 → SimpleTimer (http://playground.arduino.cc/Code/SimpleTimer)

Timer1 → Timer1 (http://playground.arduino.cc/Code/Timer1)

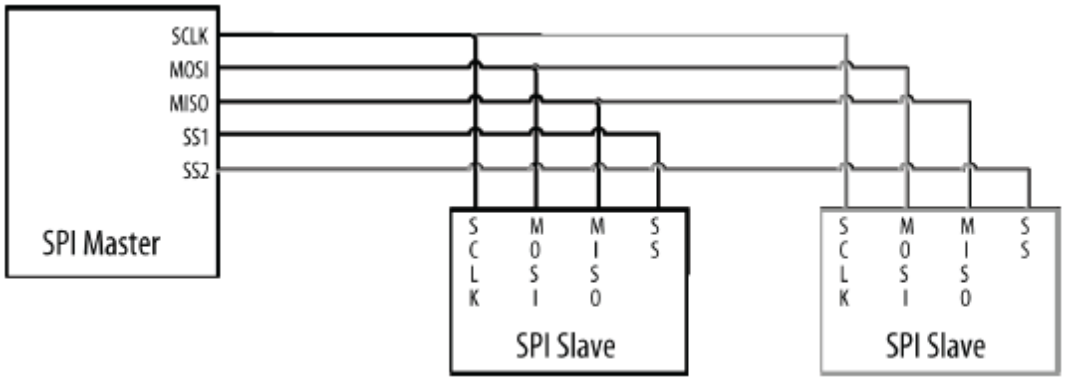
Timer2 → MsTimer2 (http://playground.arduino.cc/Main/MsTimer2) ή

→ FlexiTimer2 (http://playground.arduino.cc/Main/FlexiTimer2)

Διασύνδεση SPI

- SPI – Serial Peripheral Interface
- Σύγχρονη Σειριακή Μετάδοση δεδομένων
- Χρήση των εντολών shiftOut() και shiftIn()
- Χρήση της βιβλιοθήκης <SPI.h>

SPI signal	Standard Arduino board	Arduino Mega
SCLK (clock)	13	52
MISO (data out)	12	50
MOSI (data in)	11	51
SS (slave select)	10	53

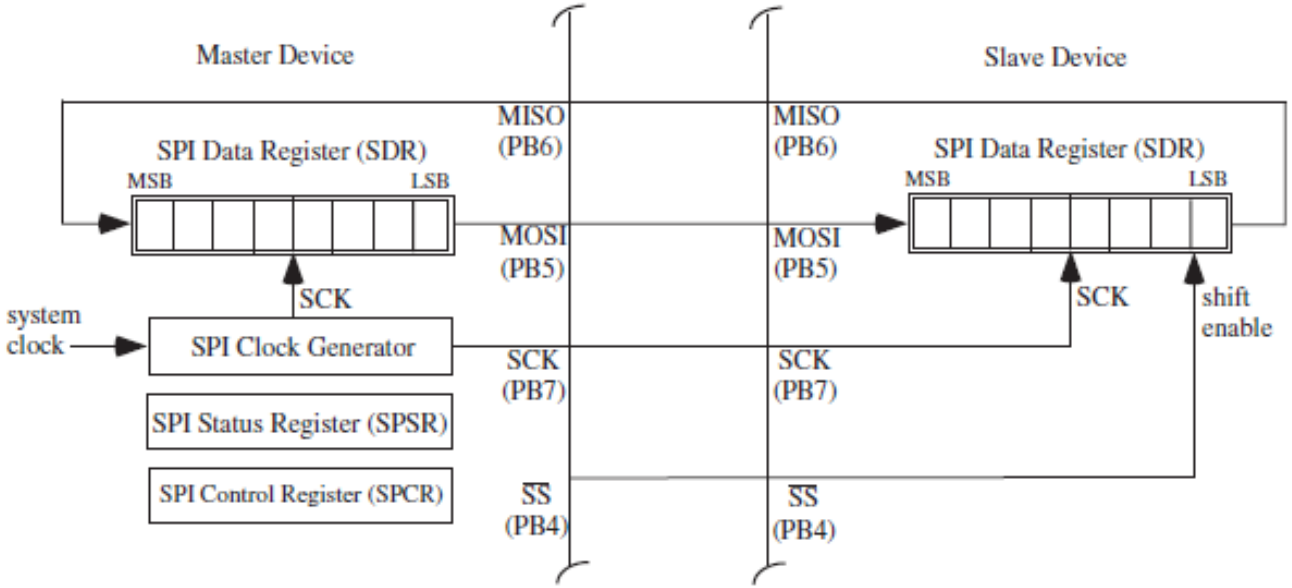


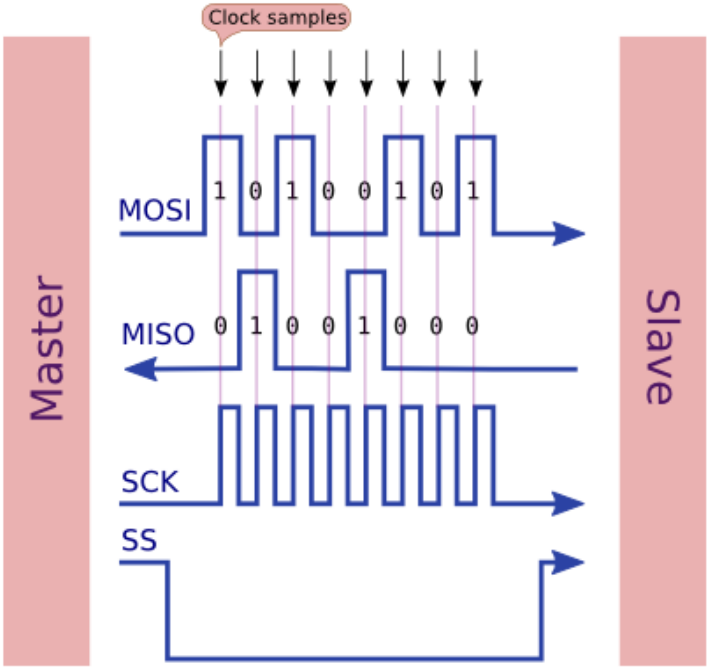
MISO – Master In Slave Out
MOSI – Master Out Slave In
SCLK – Κοινό ρολόϊ συγχρονισμού
SS – Slave Select, επιλογή περιφερειακού σαν Slave

shiftOut(dataPin, clockPin, bitOrder, value)

byte incoming = shiftIn(dataPin, clockPin, bitOrder)

MSBFIRST ή LSBFIRST





- Το SPI ενεργοποιείται όταν φορτωθεί ένα byte στον Master SDR.
- Κοινό ρολόϊ → Σύγχρονη μετάδοση. Σε 8 ανόδους ή καθόδους ρολογιού έχει μετακινηθεί ένα byte από τον Master στον Slave και ένα byte από τον Slave στον Master.
- Μόλις ολοκληρωθεί η μετακίνηση του ενός byte τοποθετείται το flag SPIF στον Master και στον Slave (SPI Interrupt Flag) που βρίσκεται στον καταχωρητή SPSR.
- Μεγάλη ταχύτητα μετάδοσης δεδομένων σε σχέση με το UART ή το I2C.

Η βιβλιοθήκη <SPI.h>

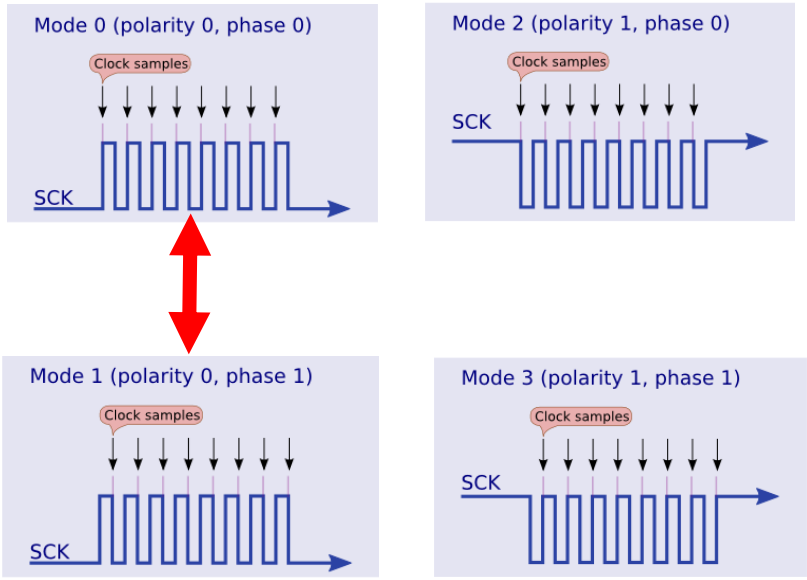
- SPISettings(frequency, bitOrder, spiMode)
  frequency – 16000000 για τη συχνότητα του Arduino.
  bitOrder – MSBFIRST ή LSBFIRST
  spiMode – Όπως στον Πίνακα

- SPI.beginTransaction()

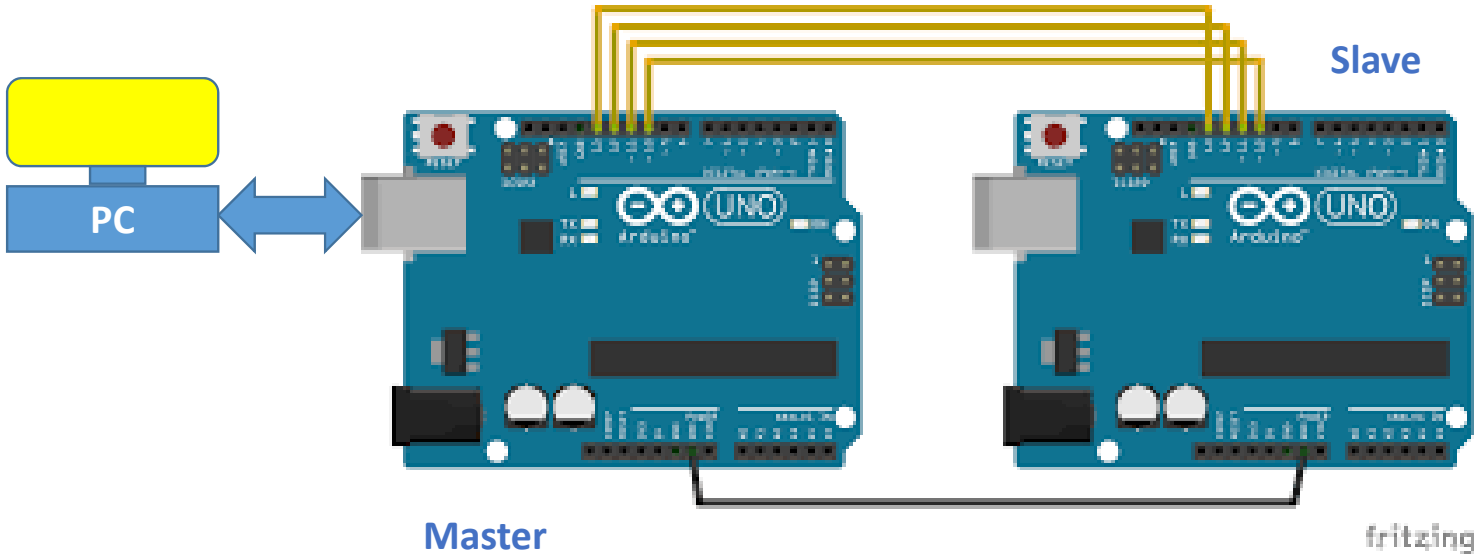
- SPI.transfer() - Πριν την κλήση το pin SS του Slave πρέπει να γίνει LOW και αμέσως μετά την κλήση πρέπει να γίνει HIGH.

- SPI.endTransaction()

Mode	Clock Polarity (CPOL)	Clock Phase (CPHA)	Output
SPI_MODE0	0	0	Falling
SPI_MODE1	0	1	Rising
SPI_MODE2	1	0	Falling
SPI_MODE3	1	1	Rising



Παράδειγμα SPI: Σύνδεση δύο Arduino Uno



```
SPCR
| 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
| SPIE | SPE  | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 |

SPIE - Enables the SPI interrupt when 1
SPE - Enables the SPI when 1
```

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Frequency Divider

Σενάριο-1

- Ο Master στέλνει στον Slave μια σειρά από χαρακτήρες.
- Ο Slave λαμβάνει τους χαρακτήρες και τους τοποθετεί σε έναν buffer.

Master code

```
#include <SPI.h>

char msg[]="SAM-Arduino\n";

void setup (){
  digitalWrite(SS, HIGH);   // disable slave
  SPI.begin();
}

void loop (){
  // send test string
  for (int i=0;i<12;i++){
    // enable Slave Select
    digitalWrite(SS, LOW);
    SPI.transfer(msg[i]);
    // disable Slave Select
    digitalWrite(SS, HIGH);
  }
    delay (1000);  // 1 seconds delay
}
```

(p22m)

Slave code

```
#include <SPI.h>

char buf [100];
volatile byte pos;
volatile boolean process_it;

void setup (void){
  Serial.begin (115200);
  // turn on SPI in slave mode
  bitSet(SPCR,SPE);
  // have to send on master in, *slave out*
  pinMode (MISO, OUTPUT);
  // get ready for an interrupt
  pos = 0;   // buffer empty
  process_it = false;
  // now turn on interrupts
  bitSet(SPCR,SPIE);
}
```

```
// SPI interrupt routine
ISR (SPI_STC_vect){
  byte c = SPDR;  // take byte from SPI Data Register
  // add to buffer if there is space
  if (pos < sizeof buf){
    buf [pos++] = c;
    // newline means the end of the buffer
    if (c == '\n')
      process_it = true;
  }
}

void loop (void){
  if (process_it){
    buf [pos] = 0;
    Serial.println (buf);
    pos = 0;
    process_it = false;
  }
}
```

(p22s)

Σενάριο-2

- Ο Master στέλνει στον Slave την εντολή “a” (που σημαίνει add) ή την εντολή “s” (που σημαίνει subtract) και 4 bytes.
- Ο Slave προσθέτει ή αφαιρεί στα 4 bytes ένα συγκεκριμένο αριθμό (π.χ. προσθέτει το 15 ή αφαιρεί το 8) και στέλνει στον Master το αποτέλεσμα της πράξης το οποίο εμφανίζεται στη Σειριακή Οθόνη του.

Master code

```
#include <SPI.h>

void setup (void){
  Serial.begin(115200);
  Serial.println("Start SPI...");
  digitalWrite(SS, HIGH);  //ensure slave not selected
  SPI.beginTransaction (SPISettings (16000000, MSBFIRST, SPI_MODE0));
}

byte transferAndWait (const byte what){
  byte a = SPI.transfer (what);
  delayMicroseconds (20);
  return a;
}

void loop (void){
  byte a, b, c, d;
  digitalWrite(SS, LOW);  // enable Slave Select
  transferAndWait('a');   // add command
  transferAndWait(10);
  a = transferAndWait(17);
  b = transferAndWait(33);
  c = transferAndWait(42);
  d = transferAndWait(0);
  digitalWrite(SS, HIGH); // disable Slave Select
```

```
 Serial.println("Adding results:");
  Serial.println(a, DEC);
  Serial.println(b, DEC);
  Serial.println(c, DEC);
  Serial.println(d, DEC);
  digitalWrite(SS, LOW);  // enable Slave Select
  transferAndWait('s');   // subtract command
  transferAndWait(10);
  a = transferAndWait(17);
  b = transferAndWait(33);
  c = transferAndWait(42);
  d = transferAndWait(0);
  digitalWrite(SS, HIGH); // disable Slave Select
  Serial.println("Subtracting results:");
  Serial.println(a, DEC);
  Serial.println(b, DEC);
  Serial.println(c, DEC);
  Serial.println(d, DEC);
  delay (1000);
}
```

(p23m)

Slave code

```
// command for incoming data
volatile byte command = 0;

void setup (){
  // slave sends data with MISO
  pinMode(MISO, OUTPUT);
  // turn on SPI in slave mode
  bitSet(SPCR,SPE);
  // turn on interrupts
  bitSet(SPCR,SPIE);
}

void loop (){
  // if SPI not active, clear current command
  if (digitalRead (SS) == HIGH)
    command = 0;
}
```

(p23s)

```
// SPI interrupt routine
ISR (SPI_STC_vect){
  byte c = SPDR;
  switch (command){
    // first dummy byte
    case 0:
      command = c;
      SPDR = 0;
      break;
    // add to incoming byte, return result
    case 'a':
      SPDR = c + 15;  // add 15
      break;
    // subtract from incoming byte, return result
    case 's':
      SPDR = c - 8;  // subtract 8
      break;
  }
}
```

Σειριακή οθόνη

```
Adding results:
25
32
48
57
Subtracting results:
2
9
25
34
```

Παρατηρήσεις

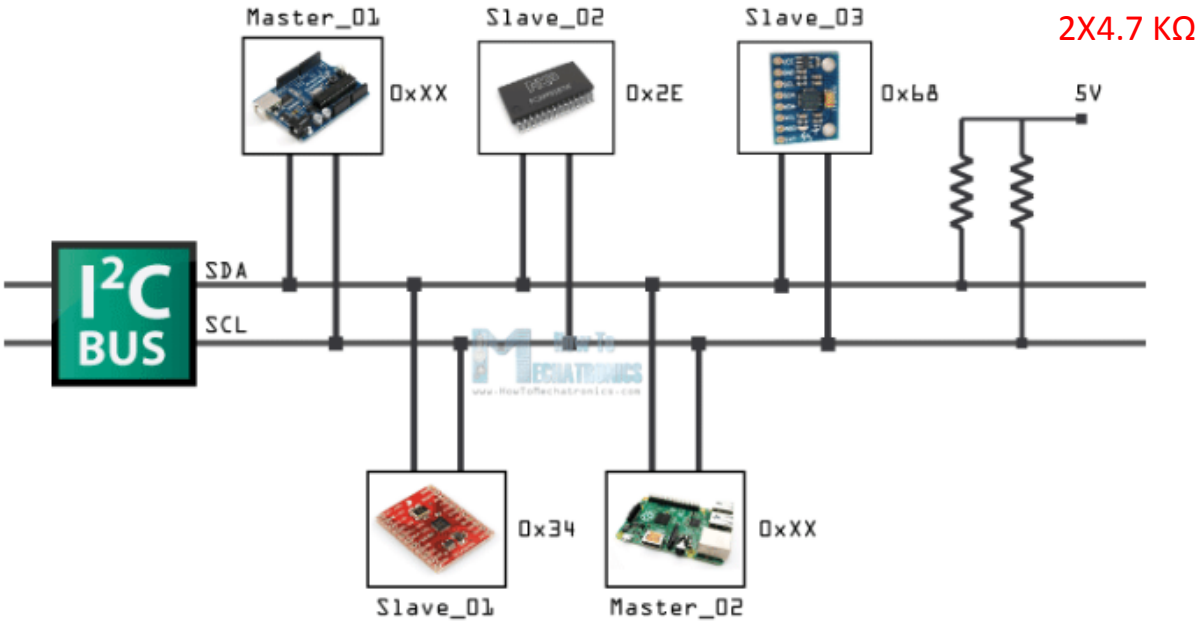
- Ο Master στέλνει: “a” → 10 → 17 → 33 → 42 → 0
- Όταν ο Master στείλει το “a” θα πάρει πίσω την τιμή SPDR του Slave που είναι 0. Ο Slave παίρνει το “a” και το αποθηκεύει command=“a”.
- Όταν ο Master στείλει το 10 θα πάρει πίσω το 0.
- Όταν ο Master στείλει το 17 θα πάρει πίσω το άθροισμα 10+15.
- Όταν ο Master στείλει το 33 θα πάρει πίσω το άθροισμα 17+15.
- Όταν ο Master στείλει το 42 θα πάρει πίσω το άθροισμα 33+15.
- Όταν ο Master στείλει το 0 θα πάρει πίσω το άθροισμα 42+15.

- **Επομένως πρέπει να στείλουμε ένα επί πλέον δεδομένο (dummy) για να πάρουμε πίσω το τελευταίο επιθυμητό.**

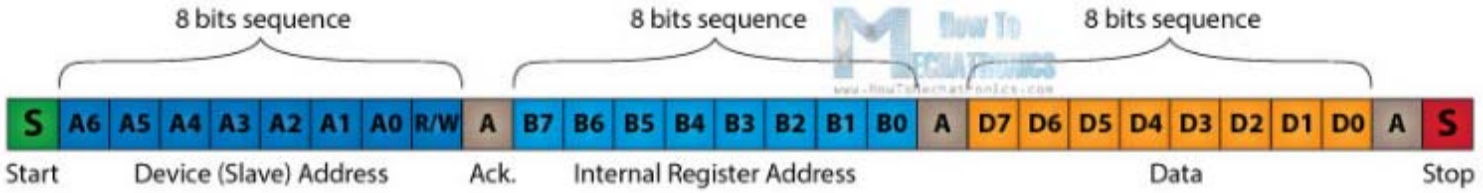
- **Επί πλέον πρέπει να χρησιμοποιήσουμε στον Master μια χρονοκαθυστέρηση (π.χ. 20 μs) για να μπορεί να ανταποκριθεί ο Slave.**

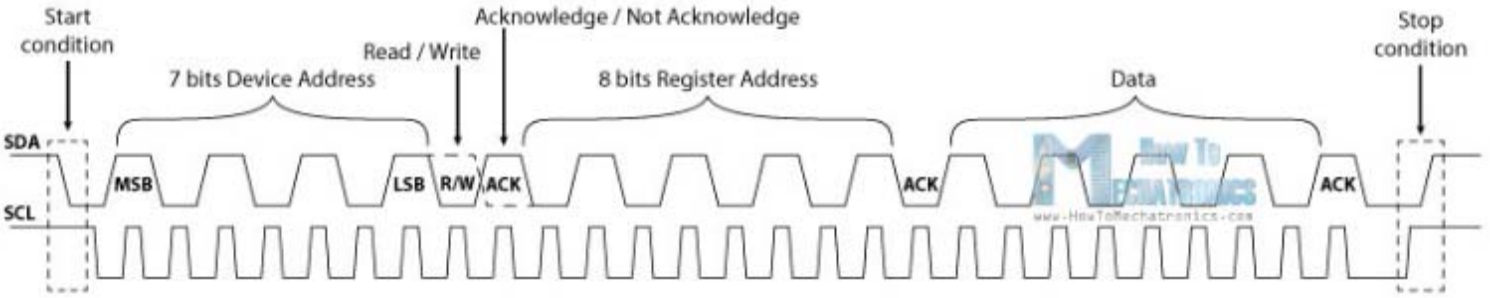
Διασύνδεση I2C

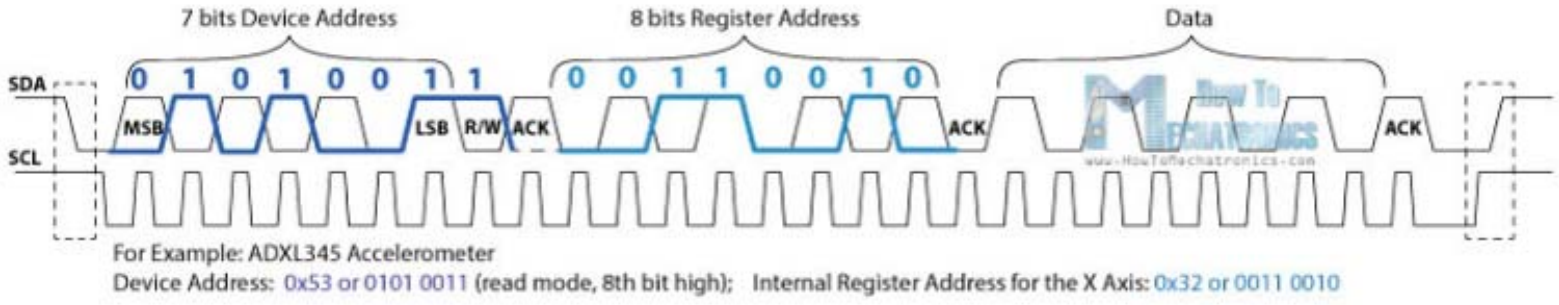
- I2C (Inter-Integrated Circuit) – Πρωτόκολλο της Philips για την απλοποίηση σύνδεσης συσκευών.
- Δύο γραμμές σύνδεσης – SDA για τα δεδομένα και SCL για το ρολόϊ συγχρονισμού που παράγεται από τον Master.
- Γραμμές “Open Drain” → 2 αντιστάσεις από 2 ΚΩ (ταχύτητα 400 kbps) έως 10 ΚΩ (ταχύτητα 100 kbps).
- Τα pins που χρησιμοποιεί ο Arduino είναι: A4 = SDA και A5= SCL



- Παλμός έναρξης με ακμή πτώσης (Start).
- Διεύθυνση συσκευής Slave (7 bits) με πρώτο το MSB → έως 128 συσκευές.
- Bit για ένδειξη ανάγνωσης ή εγγραφής (R-/W).
- Συνθήκη γνωστοποίησης λήψης (ACKnowledge) με HIGH ή μη λήψης (NACK-Not ACKnowledge) με LOW.
- Διεύθυνση εσωτερικού καταχωρητή (8 bits).
- Δεδομένα 8 bits κάθε φορά τα οποία ακολουθούνται πάντα από ένα ACK.
- Παλμός τέλους με ακμή ανόδου (STOP).







- Το επιταχυνσιόμετρο ADXL345 έχει μια μοναδική διεύθυνση 0x53 και τρεις εσωτερικούς καταχωρητές στους οποίους αποθηκεύονται τα δεδομένα για την επιτάχυνση στους 3 άξονες x, y και z. Η εσωτερική διεύθυνση του καταχωρητή x είναι 0x32.
- Η αποστολή ή λήψη δεδομένων εξαρτάται από τον παλμό R-/W.

Η βιβλιοθήκη <Wire.h>

- Ο ATMega328 διαθέτει τη μονάδα TWI (Two Wire Interface) η οποία διαχειρίζεται την επικοινωνία I2C. Η διαδικασία επικοινωνίας βασίζεται στη μετάδοση δεδομένων σε μορφή bytes με τη δημιουργία διακοπής όταν αυτή ολοκληρωθεί. Με αυτόν τον τρόπο η MCU μπορεί να ασχοληθεί με κάποια άλλη διαδικασία.
- Η βιβλιοθήκη <Wire.h>, αντίθετα, δεσμεύει την MCU στη διαδικασία I2C μέχρι αυτή να ολοκληρωθεί. Εάν αυτό δεν είναι επιθυμητό, τότε η επικοινωνία πρέπει να γίνει με κατάλληλες εντολές και χρήση των εσωτερικών καταχωρητών της μονάδας TWI.

Συναρτήσεις για συσκευή Master – Αποστολή δεδομένων	
Wire.begin()	Η εντολή αυτή καλεί τη συνάρτηση twi_init() η οποία αρχικοποιεί την επικοινωνία I2C. Πρέπει να βρίσκεται στο setup().
Wire.begin(address)	Αρχικοποίηση της επικοινωνίας I2C θέτοντας τον Arduino σαν Slave με μια μοναδική αυθαίρετη διεύθυνση 7-bits.
Wire.beginTransmission(address)	Σηματοδοτεί την έναρξη αποστολής δεδομένων σε μια συσκευή Slave η οποία έχει συγκεκριμένη διεύθυνση.
Wire.send(value) **Wire.send(string)** **Wire.send(data, quantity)**	Εγγραφή δεδομένων στον buffer για αποστολή στη συσκευή Slave, σε μορφή byte ή string ή bytes με δεδομένο μήκος μέχρι το BUFFER_LENGTH το οποίο είναι μέγιστο 32.
Wire.endTransmission()	Αποστολή στη συσκευή Slave και λήξη της επικοινωνίας με αποτέλεσμα: 0 = Επιτυχής αποστολή δεδομένων. 1 = Αποστολή περισσοτέρων των 32 bytes. 2 = Λήψη NACK μετά την αποστολή της διεύθυνσης του Slave. Λάθος διεύθυνση ή ο Slave δεν υπάρχει. Ο Master πρέπει να στείλει STOP. 3 = Λήψη NACK μετά την αποστολή δεδομένων. Ο Master πρέπει να στείλει STOP ή να ξεκινήσει πάλι με START. 4 = Κάποιο άλλο σφάλμα επικοινωνίας έχει συμβεί.
Wire.requestFrom(address, count)	Ο Master ζητά από τον Slave με διεύθυνση address να του στείλει count αριθμό bytes που δεν πρέπει να υπερβαίνουν τα 32.

Συναρτήσεις για συσκευή Master – Λήψη δεδομένων	
Wire.requestFrom(address, count)	Ο Master ζητά από τον Slave με διεύθυνση address να του στείλει count αριθμό bytes που δεν πρέπει να υπερβαίνουν τα 32.
Wire.available()	Επιστρέφει τον αριθμό των bytes που υπάρχουν στον buffer της Wire. Ο buffer αυτός γεμίζει με τη μέθοδο requestFrom() ή τη μέθοδο onReceiveService(). Η τελευταία καλείται μετά από αποστολή STOP ή επαναλαμβανόμενων START από τον Master.
Wire.receive()	Επιστρέφει το επόμενο byte από τον buffer της Wire.

Συναρτήσεις για συσκευή Slave	
Wire.onReceive(handler)	Καλεί τη συνάρτηση handler όταν η συσκευή Slave λάβει δεδομένα. Η μορφή της είναι: void handler (int byteCount), όπου byteCount είναι ο αριθμός των bytes που έχουν ληφθεί. Στον handler χρησιμοποιείται η μέθοδος receive() για διάβασμα των δεδομένων.
Wire.onRequest(handler)	Καλεί τη συνάρτηση handler όταν η συσκευή Slave θέλει να στείλει δεδομένα. Η μορφή της είναι: void handler (). Ανταποκρίνεται όταν ο Master έχει στείλει διεύθυνση Slave και Read και χρησιμοποιεί τη μέθοδο send() για να γεμίσει τον buffer.

Σύνδεση δύο Arduino μέσω I2C

- Η σύνδεση πραγματοποιείται ενώνοντας μεταξύ τους τα A4, A5 και Ground. Δεν χρειάζονται αντιστάσεις pull-up γιατί η βιβλιοθήκη ενεργοποιεί τις εσωτερικές των δύο ακροδεκτών με τη συνάρτηση begin().
- Ο Master στέλνει ή δέχεται δεδομένα από τον Slave με σύγχρονο τρόπο με τη χρήση των μεθόδων send()/write() ή receive()/read().

Ο Master στέλνει δεδομένα στον Slave

```
void setup(){
  // Start I²C bus as master
  Wire.begin();
}

void loop(){
  int input=analogRead(AnalogPin);
  // Send two bytes to slave
  Wire.beginTransmission(SlaveDeviceId);
  Wire.send(input >> 8);                  //MSB digit
  Wire.send(input & 0xFF);                //LSB digit
  Wire.endTransmission();
  delay(1000);
}
```

Ο Master δέχεται δεδομένα από τον Slave

```
void loop(){
 // Request data from slave
 Wire.beginTransmission(SlaveDeviceId);
  int available = Wire.requestFrom(SlaveDeviceId, 2);
  if (available==2){
    int receivedValue = Wire.receive()<<8 | Wire.receive();
    Serial.println(receivedValue);
  }
  else{
    Serial.print("Unexpected number of bytes received=");
    Serial.println(available);
  }
  Wire.endTransmission();
  delay(1000);
}
```

- Ο Slave λειτουργεί ασύγχρονα μέσω των μεθόδων επανάκλησης onReceive() και onRequest() για να λάβει ή να στείλει δεδομένα στον Master και μετά από εντολή του Master.

Ο Slave δέχεται δεδομένα από τον Master

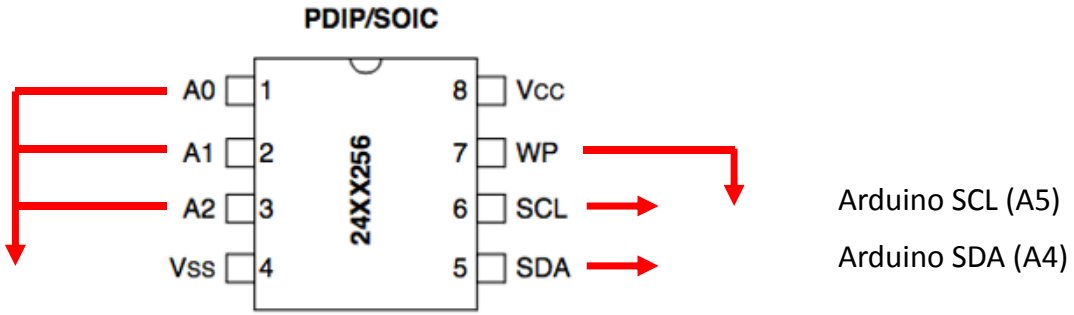
```
const byte SlaveDeviceId = 1;

void setup(){
  // Start I²C bus as a slave
  Wire.begin(SlaveDeviceId);
  // Set the callback method when data received
  Wire.onReceive(receiveCallback);
  // Watch received data
  Serial.begin(9600);
}

void loop(){ }

void receiveCallback(int byteCount){
  if (byteCount==2){
    int receivedValue = Wire.receive() << 8 | Wire.receive();
    Serial.println(receivedValue);
  }
  else{
    Serial.print("Unexpected number of bytes received=");
    Serial.println(byteCount);
  }
}
```

Ο Slave στέλνει δεδομένα στον Master

```
const byte AnalogPin = 0;
const byte SlaveDeviceId = 1;

void setup(){
  // Start I²C bus as a slave
  Wire.begin(SlaveDeviceId);
  // Set the callback method when data requested by Master
  Wire.onRequest(requestCallback);
}

void loop(){ }

void requestCallback(){
  //read & transmit to Master the analog reading from A0
  int input = analogRead(AnalogPin);
  // To send multiple bytes fill your own buffer and send it all at once
  uint8_t buffer[2];
  buffer[0] = input >> 8;
  buffer[1] = input & 0xff;
  Wire.send(buffer, 2);
}
```

Παράδειγμα I2C: Σύνδεση μνήμης EEPROM στον Arduino

- Θα χρησιμοποιηθεί η μνήμη 24C256 η οποία έχει χωρητικότητα 256 kbits ή 32 KB.
- Σύνδεση των SDA και SCL στα αντίστοιχα pins του Arduino (A4 και A5).
- Σύνδεση του WP στο Ground → Επιτρέπεται η εγγραφή της EEPROM.
- Η διεύθυνση του ολοκληρωμένου είναι “0101A2A1A0” και επομένως μπορούν να συνδεθούν μέχρι 8. Εάν A2A1A0=000, τότε η διεύθυνση είναι 0x50 (μόνο τα 7 λιγότερο σημαντικά αποστέλλονται στον Slave).

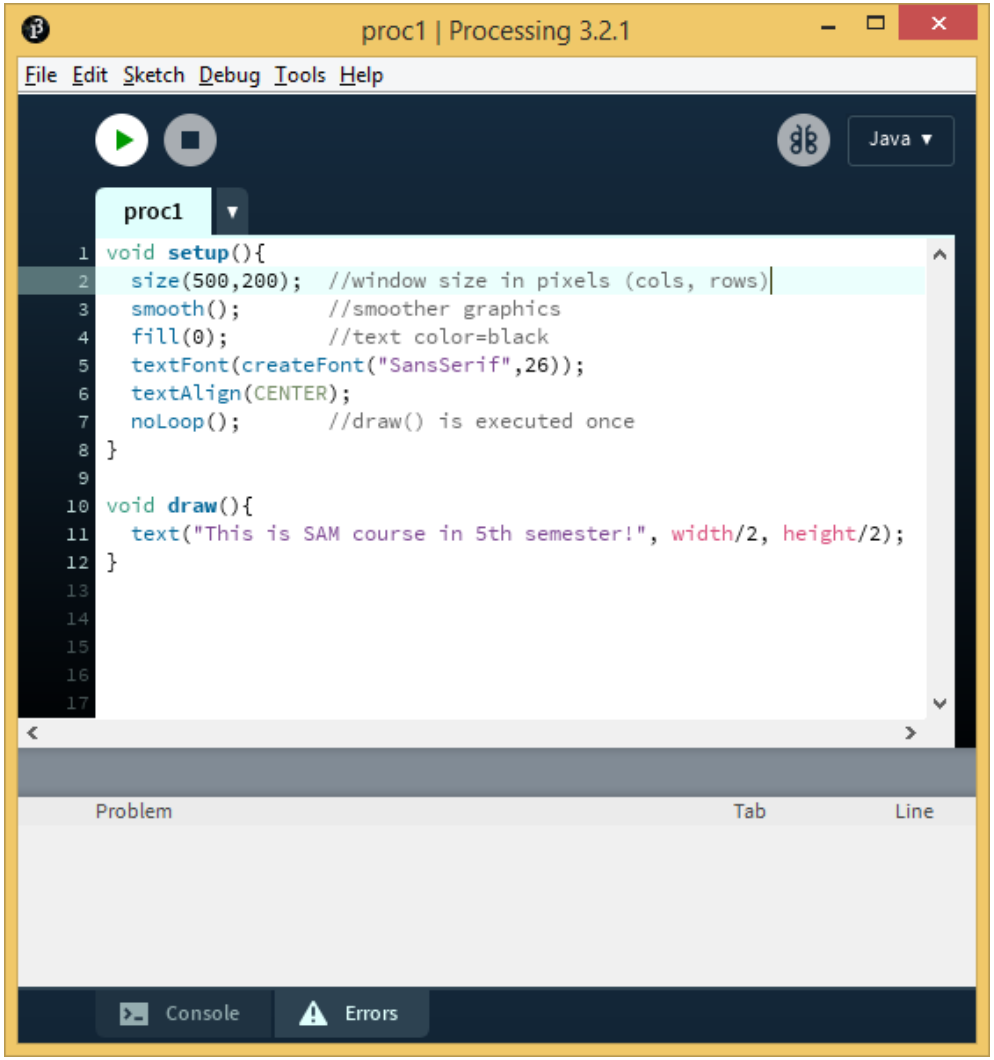


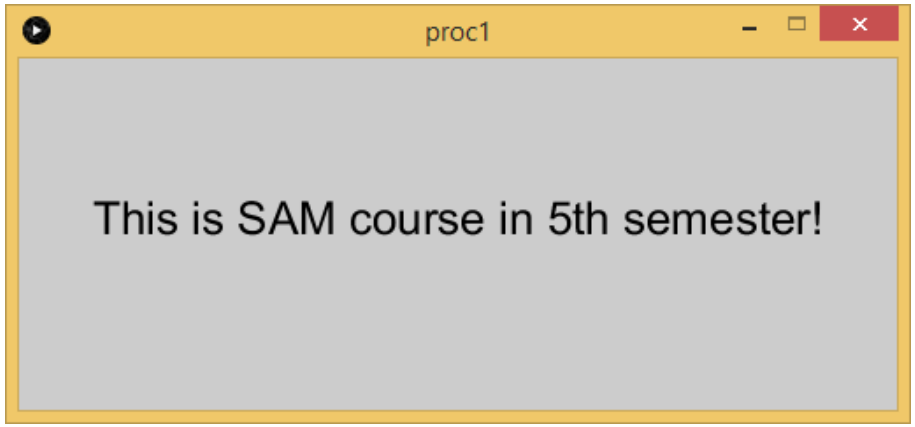
```
#include <Wire.h>

 #define eeprom1 0x50    //Address of 24LC256 eeprom chip

void setup() {
  Serial.begin(9600);
  Wire.begin();
  unsigned int address = 0;
  writeEEPROM(eeprom1, address, 123);
  Serial.print(readEEPROM(eeprom1, address), DEC);
 }

void loop(){ }

void writeEEPROM(int deviceaddress, unsigned int eeaddress, byte data) {
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8));                    // MSB
  Wire.send((int)(eeaddress & 0xFF));                  // LSB
  Wire.send(data);
  Wire.endTransmission();
  delay(5);
}
```

```
byte readEEPROM(int deviceaddress, unsigned int eeaddress ) {
  byte rdata = 0xFF;
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8));                    // MSB
  Wire.send((int)(eeaddress & 0xFF));                  // LSB
  Wire.endTransmission();
  Wire.requestFrom(deviceaddress,1);
  if (Wire.available())
    rdata = Wire.receive();
    return rdata;
}
```

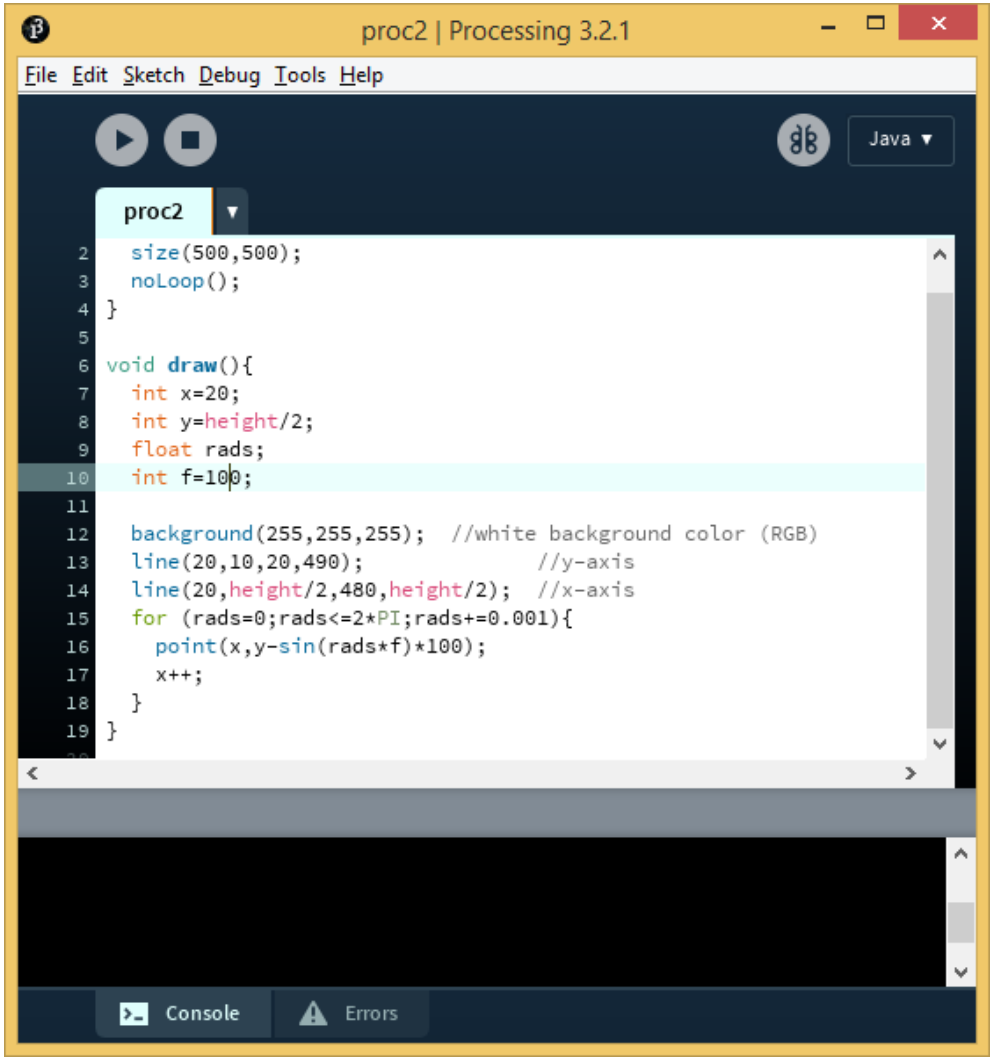
(p24)

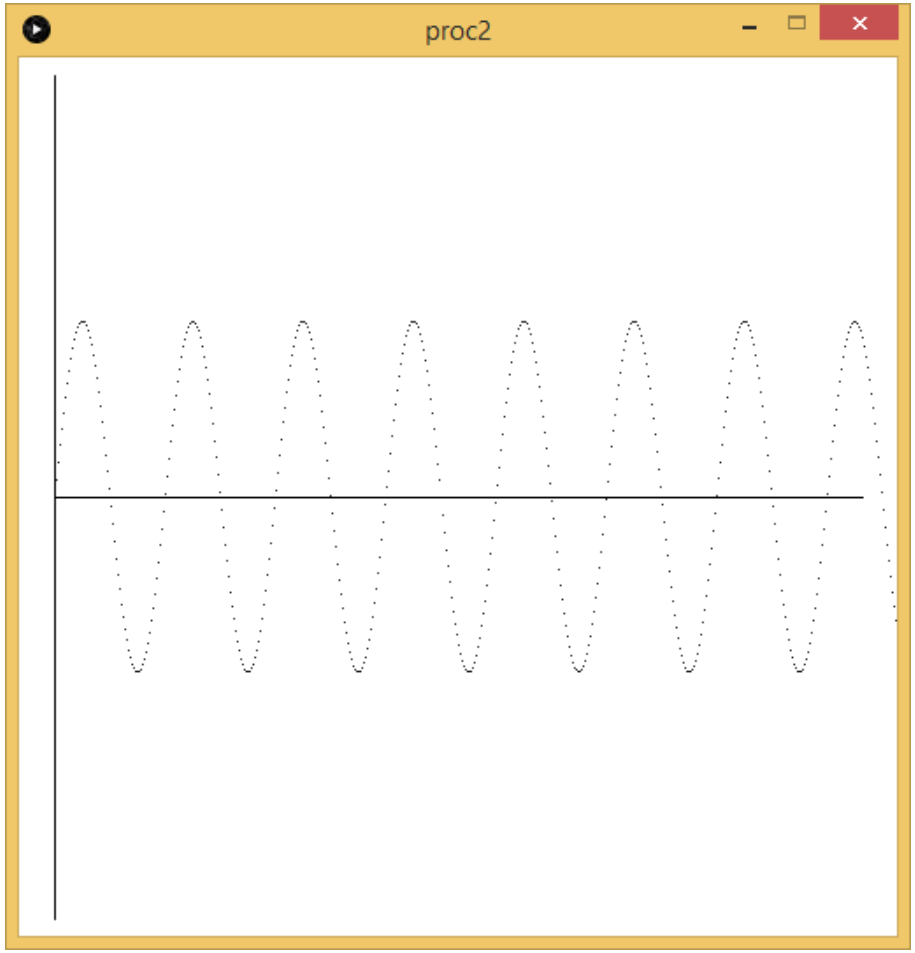
Μεταφορά δεδομένων (Processing)


proc1 | Processing 3.2.1
File Edit Sketch Debug Tools Help
Java
proc1
1 void setup(){
2 size(500,200); //window size in pixels (cols, rows)
3 smooth(); //smoother graphics
4 fill(0); //text color=black
5 textFont(createFont("SansSerif",26));
6 textAlign(CENTER);
7 noLoop(); //draw() is executed once
8 }
9
10 void draw(){
11 text("This is SAM course in 5th semester!", width/2, height/2);
12 }
Problem
Tab
Line
Console
Errors


proc1
This is SAM course in 5th semester!

(proc1)


proc2 | Processing 3.2.1
File Edit Sketch Debug Tools Help
Java
proc2
size(500,500);
noLoop();
}
void draw(){
int x=20;
int y=height/2;
float rads;
int f=100;
background(255,255,255); //white background color (RGB)
line(20,10,20,490); //y-axis
line(20,height/2,480,height/2); //x-axis
for (rads=0;rads<=2*PI;rads+=0.001){
point(x,y-sin(rads*f)*100);
x++;
}
}
Console
Errors


proc2

(proc2)

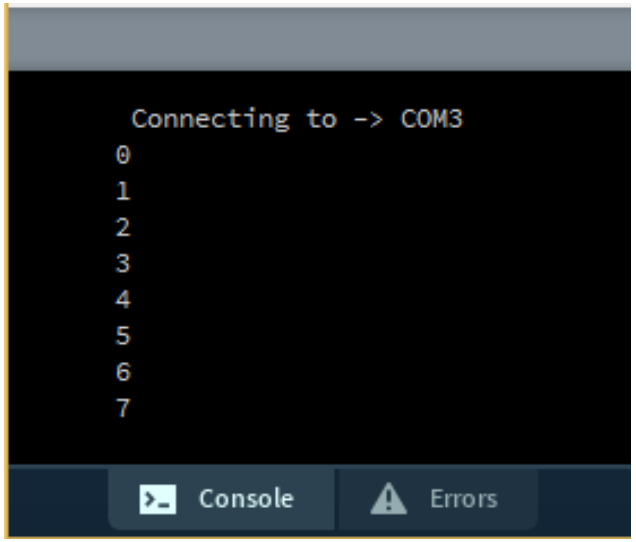
Αποστολή δεδομένων από τον Arduino στο Processing

```
int x=0;

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

void loop() {
  Serial.println(x);
  x++;
  delay(2000);
}
```

(p25)

```
import processing.serial.*;

Serial myPort;       // Create object from Serial class
int portIndex = 0; // set this to the port connected to Arduino

void setup(){
  println(" Connecting to -> " + Serial.list()[portIndex]);
  myPort = new Serial(this, Serial.list()[portIndex], 9600);
  myPort.bufferUntil('\n');
}

void draw(){}

void serialEvent(Serial myPort){
  String str=myPort.readStringUntil('\n');
  print(str);
}
```

(proc3)



Παράδειγμα: Αποστολή δεδομένων από τον Arduino στο Processing

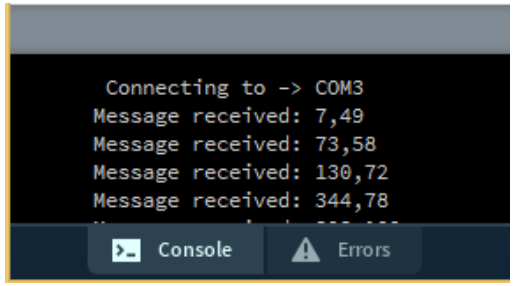
- Ο Arduino στέλνει σειριακά έναν Header και δύο τυχαίους ακέραιους αριθμούς από 0 έως 400.
- Το Processing τους παίρνει σειριακά και σχηματίζει ένα ορθογώνιο παραλληλόγραμμο.

```
int val;

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

void loop(){
 Serial.print('H'); // send a header character
 // send a random integer
 val = random(400); // random number between 0 and 400
 // send the two bytes that comprise an integer
 Serial.write(lowByte(val)); // send the low byte
 Serial.write(highByte(val)); // send the high byte
 // send another random integer
 val = random(400); // random number between 0 and 400
 // send the two bytes that comprise an integer
 Serial.write(lowByte(val)); // send the low byte
 Serial.write(highByte(val)); // send the high byte
 delay(5000);
}
```

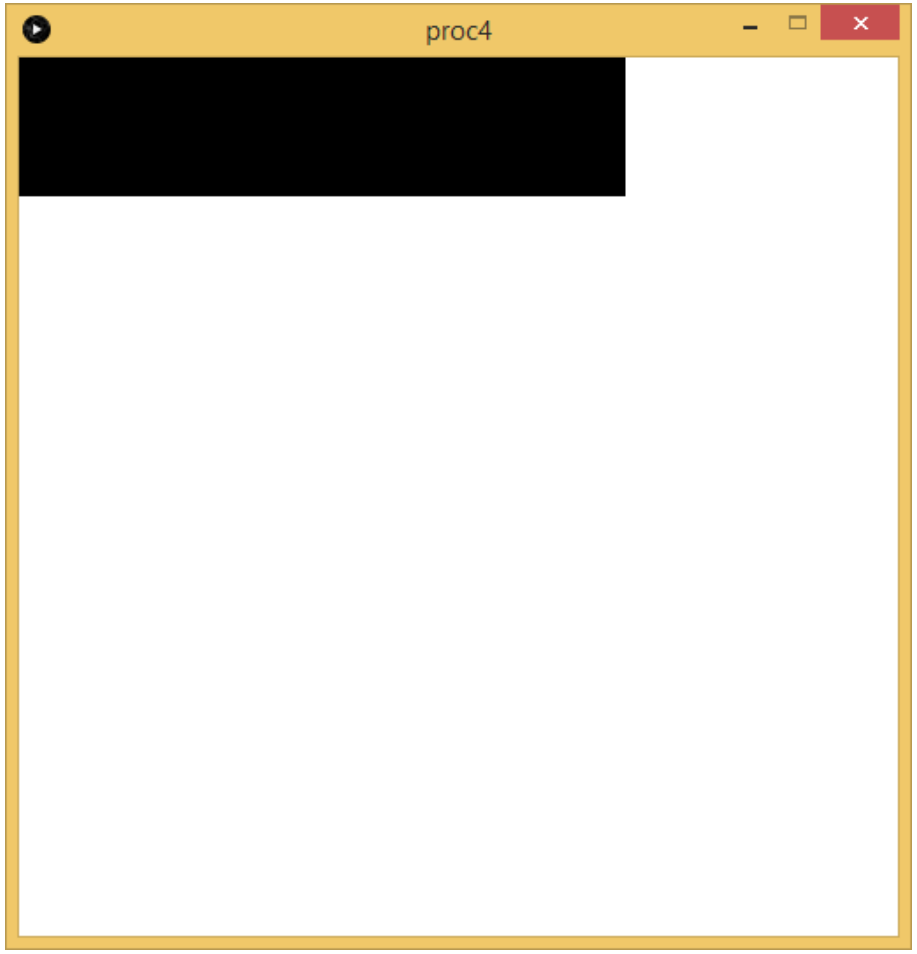
```
import processing.serial.*;

char HEADER = 'H';
int val1, val2; // Data received from the serial port

Serial myPort;    // Create object from Serial class
int portIndex = 0; // set this to the port connected to Arduino

void setup(){
 size(500,500);
 println(" Connecting to -> " + Serial.list()[portIndex]);
 myPort = new Serial(this, Serial.list()[portIndex], 9600);
 myPort.bufferUntil('\n');
}
```

(p26)

```
void draw(){
 // read the header and two integers (5 bytes)
 if ( myPort.available() >= 5){
  if( myPort.read() == HEADER){        // is this the header?
    val1 = myPort.read();              // read the LSB of first integer
    val1 = myPort.read() * 256 + val1; // add the MSB
    val2 = myPort.read();              // read the LSB of second integer
    val2 = myPort.read() * 256 + val2; // add the MSB
    println("Message received: " + val1 + "," + val2);
  }
 }
 background(255);          // set background to white
 fill(0);                  // set fill to black
 // draw rectangle with coordinates based on the integers received from Arduino
 rect(0, 0, val1,val2);
}
```



(proc4)



Αποστολή δεδομένων από τον Arduino και λήψη με αποθήκευση σε αρχείο

- Χρησιμοποιείται το πρόγραμμα (p26) για τον Arduino.
- Το Processing καταχωρεί τις τιμές σε αρχείο με όνομα την τρέχουσα ημερομηνία και ώρα. Το αποθηκεύει στον ίδιο φάκελο με το sketch μόλις πατηθεί ένα οποιοδήποτε πλήκτρο.

```
import processing.serial.*;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

PrintWriter output;
DateFormat df = new SimpleDateFormat("ddMMyyyy_hhmm");
DateFormat tf = new SimpleDateFormat("hh:mm:ss");
String fileName;
Serial myPort;      // Create object from Serial class
short portIndex = 0;
char HEADER = 'H';

void setup(){
  size(200, 200);
  // Open the serial port is connected to Arduino.
  String portName = Serial.list()[portIndex];
  println(" Connecting to -> " + portName);
  myPort = new Serial(this, portName, 9600);
  Date now = new Date();
  fileName = df.format(now);
  output = createWriter(fileName + ".csv"); // save file in sketch folder
}
```

```
void draw(){
  int val1, val2;
  if ( myPort.available() >= 5){    // wait the entire message
    if( myPort.read() == HEADER){   // is this the header
      String timeString = tf.format(new Date());
      output.print(timeString);
      val1 = myPort.read();           // read the LSB of first integer
      val1 = myPort.read() * 256 + val1; // add the MSB
      val2 = myPort.read();           // read the LSB of second integer
      val2 = myPort.read() * 256 + val2; // add the MSB
      println("Message received: " + val1 + "," + val2);
      output.print("," + val1 + "," + val2);
      output.println();
    }
  }
}

void keyPressed() {
  output.flush(); // Writes the remaining data to the file
  output.close(); // Finishes the file
  exit(); // Stops the program
}
```

(proc5)

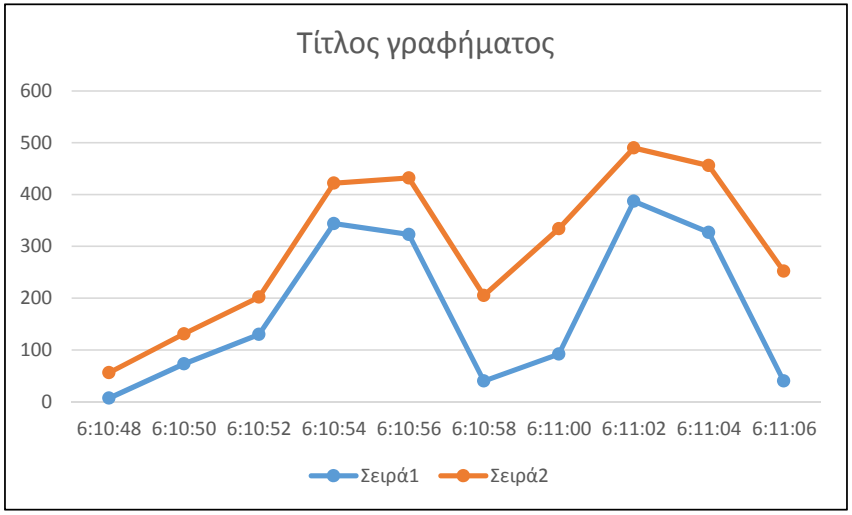
```
Connecting to -> COM3
Message received: 7,49
Message received: 73,58
Message received: 130,72
Message received: 344,78
Message received: 323,109
Message received: 40,165
Message received: 92,242
Message received: 387,103
Message received: 327,129
Message received: 40,212
```

Console | Errors

20082016_0610.csv	20/8/2016 6:11 μμ	Αρχείο τιμών διαχωρισμένων με κόμμα το...	1 KB
proc5.pde	20/8/2016 6:10 μμ	PDE File	2 KB

Άνοιγμα σαν csv

	A	B	C	D
1	06:10:48,7,49			
2	06:10:50,73,58			
3	06:10:52,130,72			
4	06:10:54,344,78			
5	06:10:56,323,109			
6	06:10:58,40,165			
7	06:11:00,92,242			
8	06:11:02,387,103			
9	06:11:04,327,129			
10	06:11:06,40,212			



	A	B	C	D
1	6:10:48	7	49	
2	6:10:50	73	58	
3	6:10:52	130	72	
4	6:10:54	344	78	
5	6:10:56	323	109	
6	6:10:58	40	165	
7	6:11:00	92	242	
8	6:11:02	387	103	
9	6:11:04	327	129	
10	6:11:06	40	212	
11				

Άνοιγμα Excel → Δεδομένα → Λήψη Εξωτερικών Δεδομένων → Από Κείμενο → Αρχείο → Επόμενο → Τικ στο Κόμμα → Τέλος

Σύνδεση Arduino με το Matlab

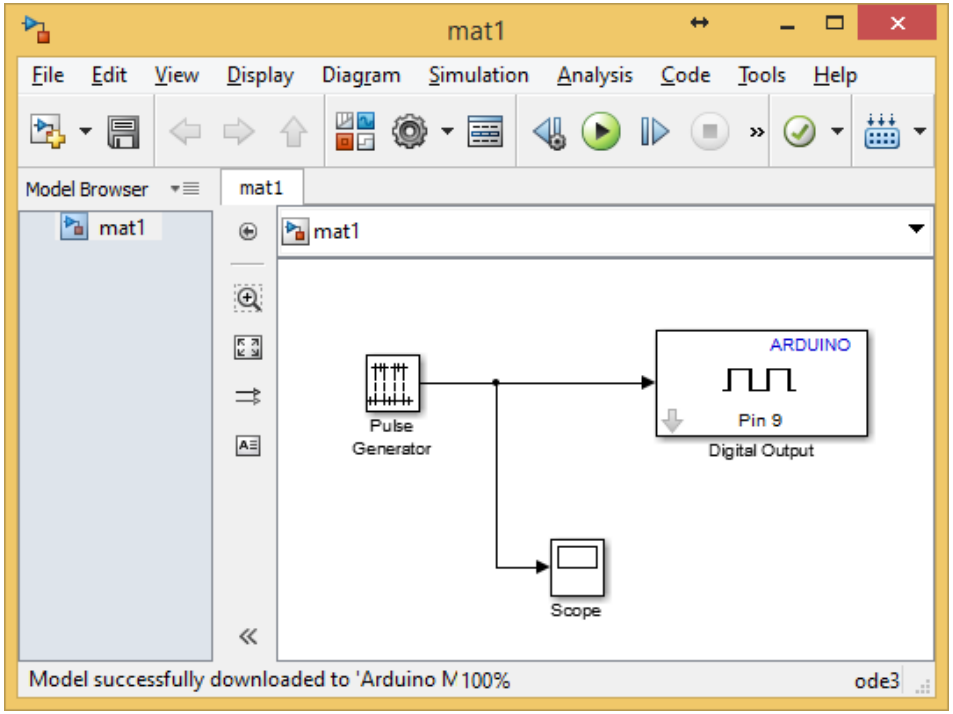
ΧΡΗΣΗ ΤΟΥ SIMULINK ΓΙΑ ΣΥΝΔΕΣΗ ΜΕ ΤΟΝ ARDUINO UNO

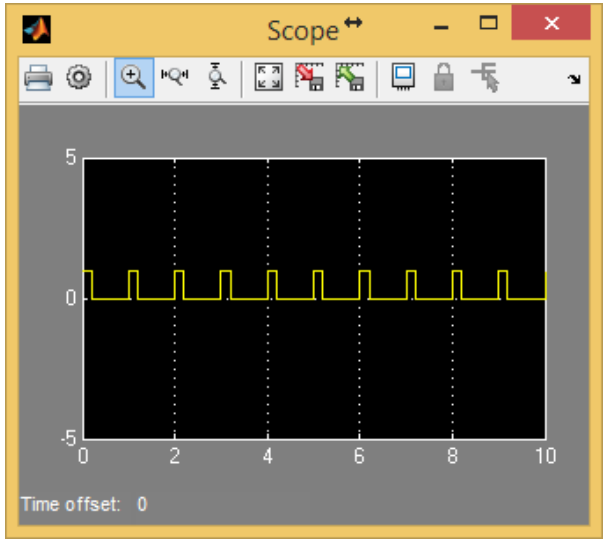
- Matlab R2013a
- Δημιουργούμε ένα account στην Mathworks
- Add-Ons → Get Hardware Support Packages → Arduino

Παράδειγμα 1 – Οδήγηση ενός LED με γεννήτρια παλμών (mat1.slx)

- Matlab → Simulink Library → Simulink Support Package for Arduino Hardware
- Matlab → New → Simulink Model → mat1.slx
- Σύρουμε από το “Simulink Support Package for Arduino Hardware” το μπλοκ “Digital Output” στο mat1
- Σύρουμε από το “Sources” ένα “Pulse Generator”
- Σύρουμε από το “Sinks” ένα “Scope”
- Το path του Matlab πρέπει να δείχνει τον φάκελο του παραδείγματος
- mat1 → Tools → Run On Target Hardware → Prepare to Run ή Options → Target Hardware: Arduino Uno → Set host COM port: Manually → COM port number: 3 (Η COM που έχει συνδεθεί ο Arduino)
- mat1 → Tools → Run On Target Hardware → Run
- Το αντίστοιχο πρόγραμμα φορτώνεται στον Arduino και εκτελείται

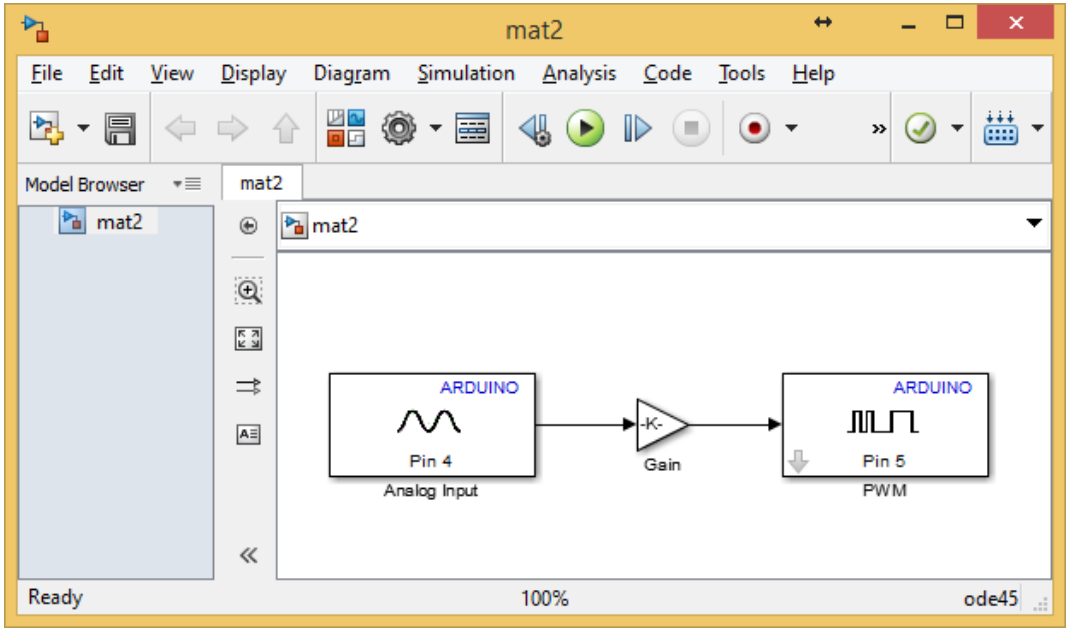
Συμβουλευτείτε το αρχείο “ΣΥΝΔΕΣΗ ΤΟΥ MATLAB ΜΕ ΤΟΝ ARDUINO UNO.docx” για περισσότερες λεπτομέρειες


mat1
File Edit View Display Diagram Simulation Analysis Code Tools Help
Model Browser
mat1
mat1
mat1
ARDUINO
Pin 9
Digital Output
Pulse
Generator
Scope
Model successfully downloaded to 'Arduino M
100%
ode3


Scope
5
0
-5
0
2
4
6
8
10
Time offset: 0

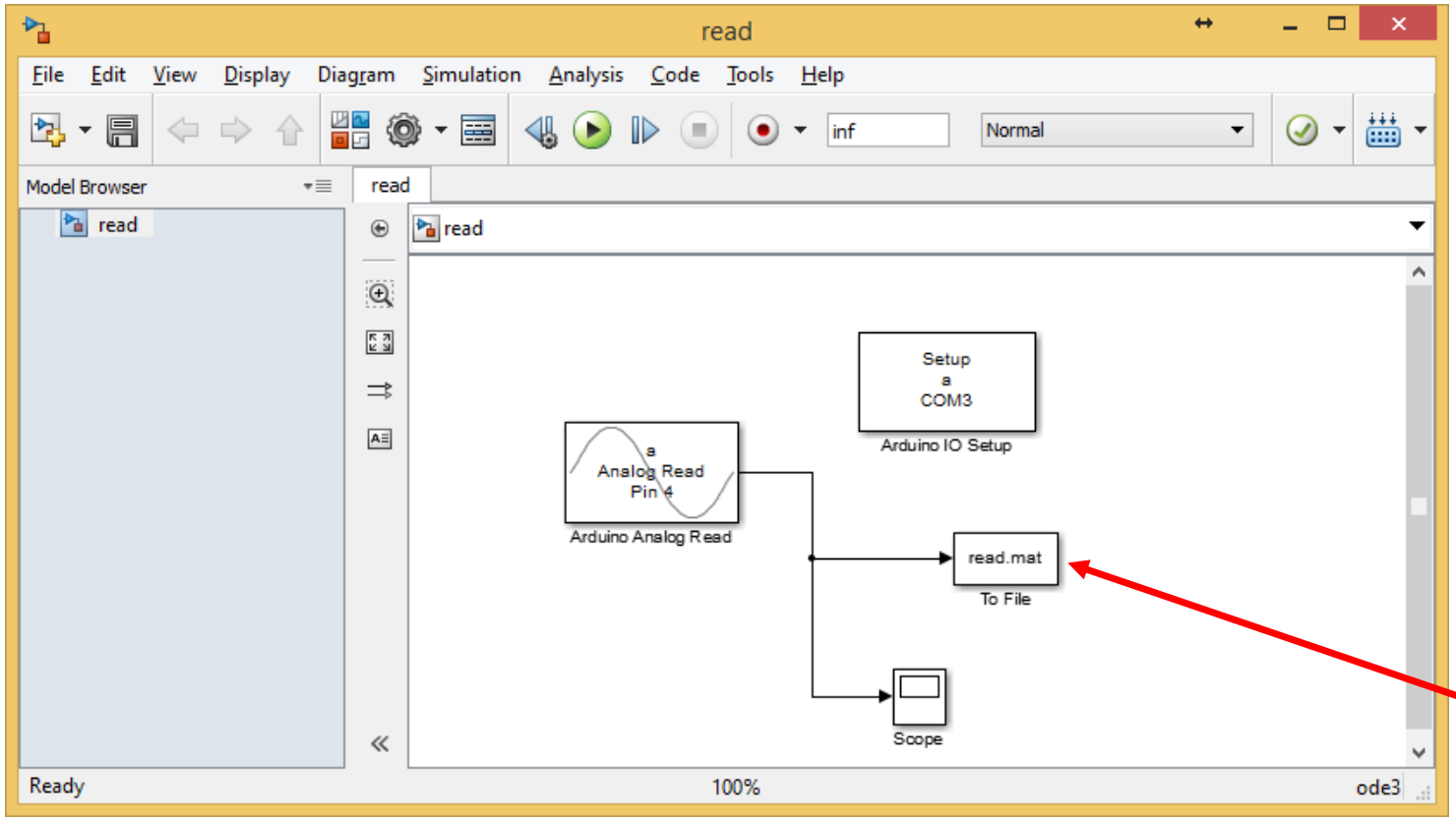
Παράδειγμα 2 – Ανάγνωση ποτενσιομέτρου (mat2.slx)

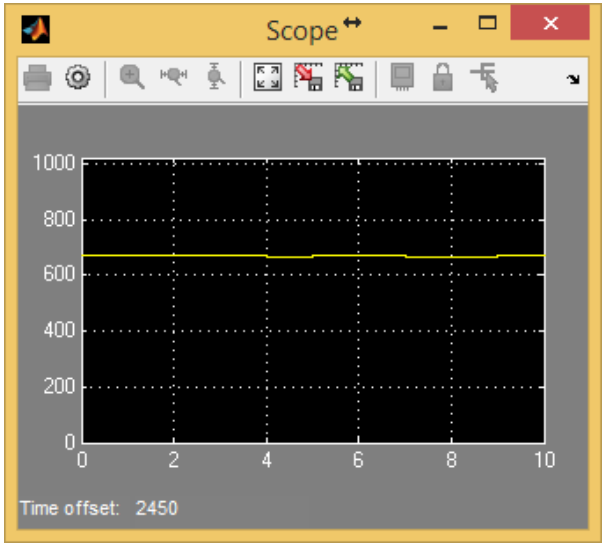
- New → Simulink Model και το αποθηκεύουμε με το όνομα “mat2.slx”
- Simulink Support Package for Arduino Hardware → Analog Input
- Simulink Support Package for Arduino Hardware → PWM
- Simulink → Commonly Used Blocks → Gain → Τοποθετούμε K=255/1023 για περιοχή λειτουργίας 0-255
- Συνδέουμε στο pin A4 του Arduino ένα ποτενσιόμετρο μεταξύ τάσης +5 V και Gnd
- Συνδέουμε στο pin 5 του Arduino ένα LED
- Run και παρατηρούμε την ένδειξη του LED ανάλογα με την τιμή του ποτενσιομέτρου



ΧΡΗΣΗ ΤΟΥ SIMULINK ΓΙΑ ΔΙΑΒΑΣΜΑ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟΝ ARDUINO UNO

- Φορτώνουμε στον Arduino το “adioes.pde”.
- Εκτελούμε a=arduino(‘COM3’); → Arduino successfully connected !
- Στο Simulink δημιουργούμε το read.slx από το “ArduinoIO Library”

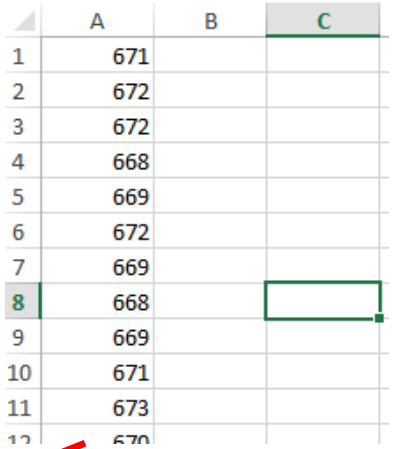




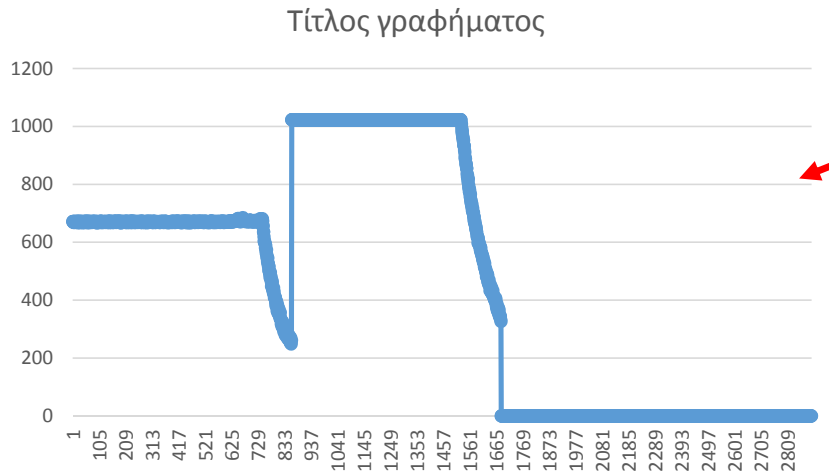
Αρχείο read.mat

- Μετατροπή του αρχείου “read.mat” στο αρχείο Excel “read.xls”

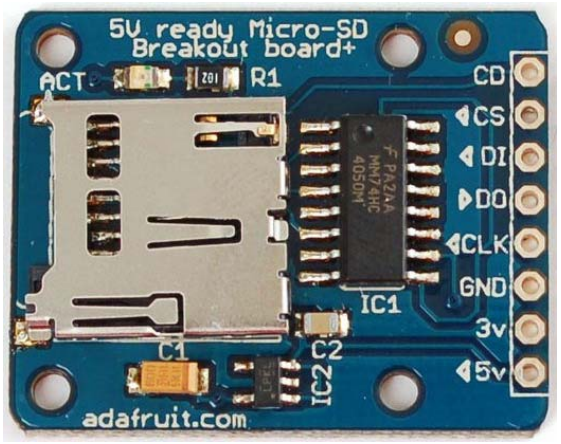
```
>> load read
>> x=ans.data;
>> xlswrite('read.xls',x);
```



	A	B	C
1	671		
2	672		
3	672		
4	668		
5	669		
6	672		
7	669		
8	668		
9	669		
10	671		
11	673		



Αποθήκευση δεδομένων σε κάρτα SD



- Σύνδεση **5V** και **GND** με τα αντίστοιχα του Arduino
- Σύνδεση **CLK** στο pin **13** (SCK)
- Σύνδεση **DO** στο pin **12** (MISO)
- Σύνδεση **DI** στο pin **11** (MOSI)
- Σύνδεση **CS** στο pin **10** (SS)
- CD=0 όταν η κάρτα είναι μέσα

Βιβλιοθήκη <SD.h>

- Τα αρχεία πρέπει να έχουν τύπο 8.3 (π.χ. LOGGER00.CSV)
- Υπάρχουν έτοιμα παραδείγματα στο Arduino IDE
- https://www.arduino.cc/en/Reference/SD για τις συναρτήσεις

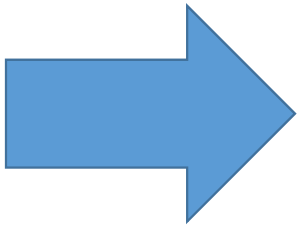
```
#include <SD.h>
#include <SPI.h>

#define LOG_INTERVAL  1000  // mills between entries
#define SYNC_INTERVAL 1000 // mills between write data to the card
uint32_t syncTime = 0;           // time of last sync()

#define sdSelect  10      // SD cs line
#define flexPin 0          // data logging analog input 0

// the logging file
File logfile;

void error(char *str){
  Serial.print("error: ");
  Serial.println(str);
  while(1);
}

void setup(){
  Serial.begin(9600);
  // initialize the SD card
  Serial.print("Initializing SD card...");
  pinMode(sdSelect, OUTPUT);
  // card initialization
  if (!SD.begin(sdSelect)) {
    error("Card failed, or not present");
  }
  Serial.println("card initialized");
```

```
  // create a new file
  char filename[] = "LOGGER00.CSV";
  for (uint8_t i = 0; i < 100; i++) {
    filename[6] = i/10 + '0';
    filename[7] = i%10 + '0';
    if (! SD.exists(filename)) {
      // only open a new file if it doesn't exist
      logfile = SD.open(filename, FILE_WRITE);
      break;  // leave the loop!
    }
  }
  if (! logfile) {
    error("couldnt create file");
  }
  Serial.print("Logging to: ");
  Serial.println(filename);
  logfile.println("TIME, ANGLE");
  Serial.println("TIME, ANGLE");
}

void loop(){
  // delay between readings
  delay((LOG_INTERVAL -1) - (millis() % LOG_INTERVAL));
  // log milliseconds since starting
  uint32_t m = millis();
  logfile.print(m);          // milliseconds since start
  logfile.print(", ");
  int flexReading = analogRead(flexPin);
  logfile.print(flexReading);
  logfile.println();
```

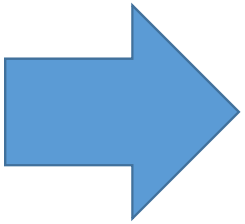
```
  Serial.print(m);           // milliseconds since start
  Serial.print(", ");
  Serial.print(flexReading);
  Serial.println();

  // write data to disk
  // Don't sync too often - requires 2048 bytes of I/O to SD card
  if ((millis() - syncTime) < SYNC_INTERVAL) return;
  syncTime = millis();
  logfile.flush();
}
```

Δημιουργία βιβλιοθήκης

```
int led 13;

void pwm_1(){
  analogWrite(led,0);
}

void pwm_2(){
  analogWrite(led,255);
}

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}

void loop() {
  pwm_1();
  delay(5000);
  pwm_2();
  delay(5000);
}
```



```
#include <PWM.h>

PWM pwm(13);

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

void loop() {
  pwm.pwm_1 ();
  delay(5000);
  pwm.pwm_2();
  delay(5000);
}
```



PWM.h
PWM.cpp
keywords.txt

Header file: PWM.h

```
#ifndef PWM_h
#define PWM_h

#include <Arduino.h>

class PWM{
  public:
    PWM(int pin);       // CLASS CONSTRUCTOR
    void pwm_1();       // CLASS FUNCTION
    void pwm_2();       // CLASS FUNCTION
    int pinout;         // CLASS VARIABLE
};

#endif
```

Αποφεύγουμε την διπλή δήλωση μιας βιβλιοθήκης

Access to the standard types and constants of the Arduino language

Source file: PWM.cpp

```
#include <PWM.h>

PWM::PWM(int pin){
  pinMode(pin, OUTPUT);
  pinout=pin;
}

void PWM::pwm_1(){
  analogWrite(pinout, 0);
}

void PWM::pwm_2(){
  analogWrite(pinout, 255);
}
```

Δημιουργία της βιβλιοθήκης

- Δημιουργούμε έναν φάκελο “PWM” στον φάκελο libraries της διανομής Arduino.
- Τοποθετούμε μέσα το PWM.h και το PWM.cpp.
- Επειδή ο Arduino δεν αναγνωρίζει τις κλάσεις και τις συναρτήσεις της βιβλιοθήκης μας, δημιουργούμε ένα αρχείο “keywords.txt” το οποίο περιέχει δηλώσεις για αυτά. Έτσι μπορεί να χρωματίσει τα keywords. **Δεν ισχύει πλέον.**
- Μπορούμε να δημιουργήσουμε και έναν φάκελο με όνομα “examples”, όπου θα τοποθετήσουμε ένα παράδειγμα χρήσης της βιβλιοθήκης και θα φαίνεται στα Examples του Arduino.