Μικροελεγκτές Arduino

Μάρτιος 2019

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

Τί είναι ο Μικροελεγκτής

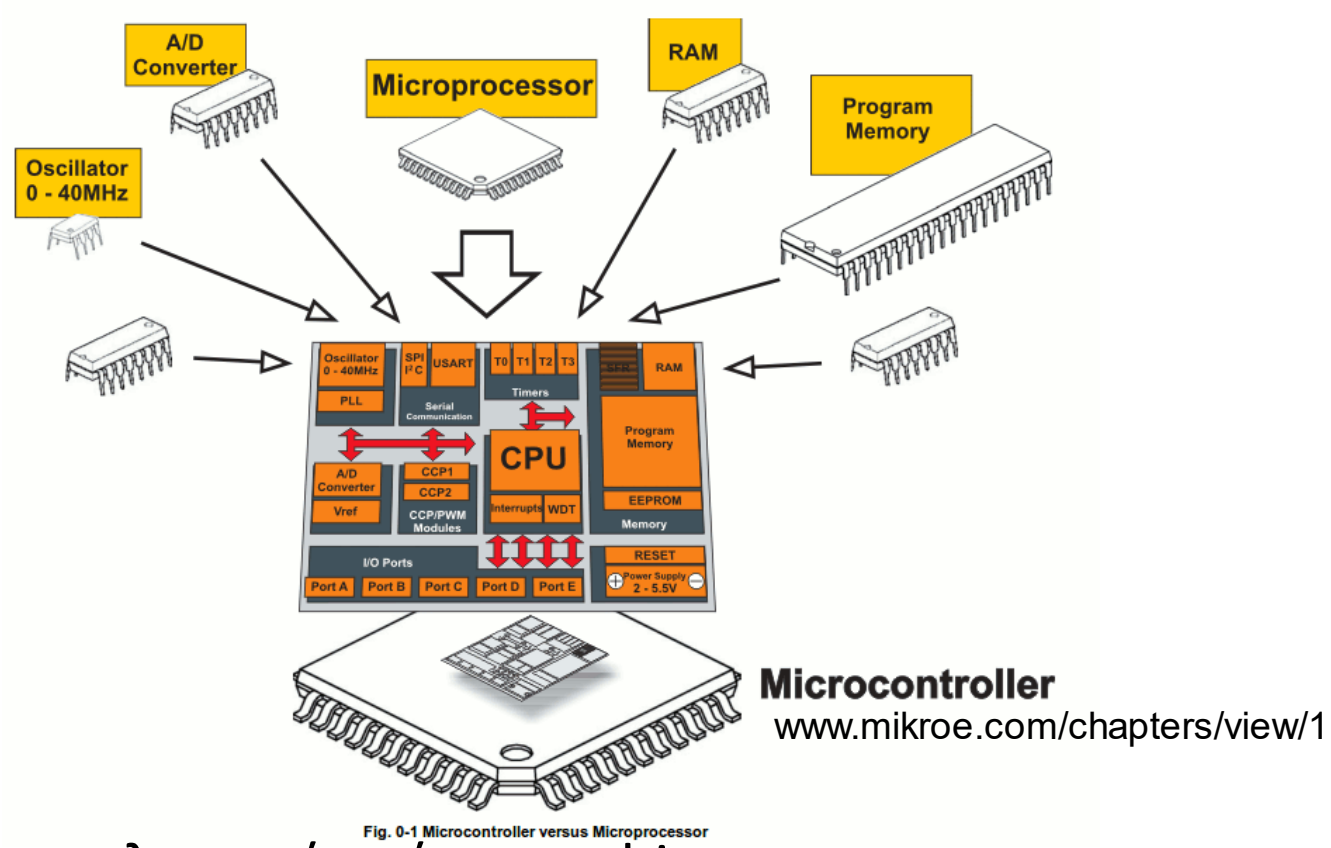
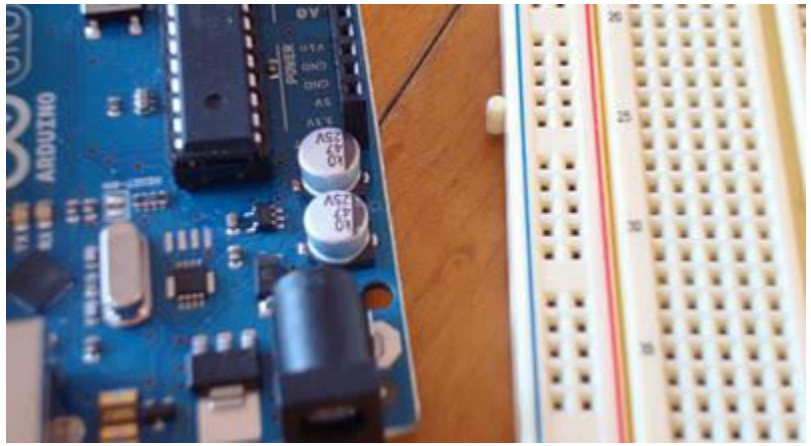


Fig. 0-1 Microcontroller versus Microprocessor

- Μικρός υπολογιστής μέσα σε chip
  - Περιέχει επεξεργαστή, μνήμη και εισόδους/εξόδους
- Συνήθως ενσωματώνεται σε μια συσκευή που ελέγχει
- Ένας μικροελεγκτής είναι συχνά μικρός σε μέγεθος και έχει χαμηλό κόστος

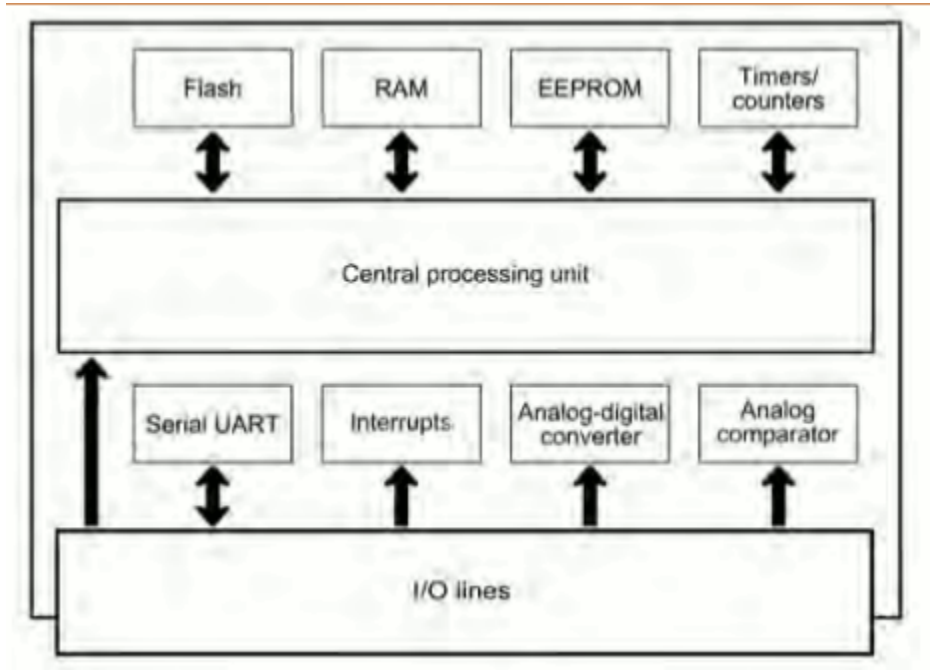
Τί είναι το Board Ανάπτυξης



- Ένα PCB σχεδιασμένο να υποστηρίξει διεργασίες ενός μικροελεγκτή.

- Τυπικά τμήματά του είναι:
  - Κύκλωμα τροφοδοσίας
  - Προγραμματιζόμενο κύκλωμα
  - Βασικές είσοδοι, συνήθως κουμπιά και LED
  - Ακροδέκτες I/O

Ο Μικροελεγκτής Arduino: Atmel ARV Atmega 328



Γιατί δημιουργήθηκε το Arduino

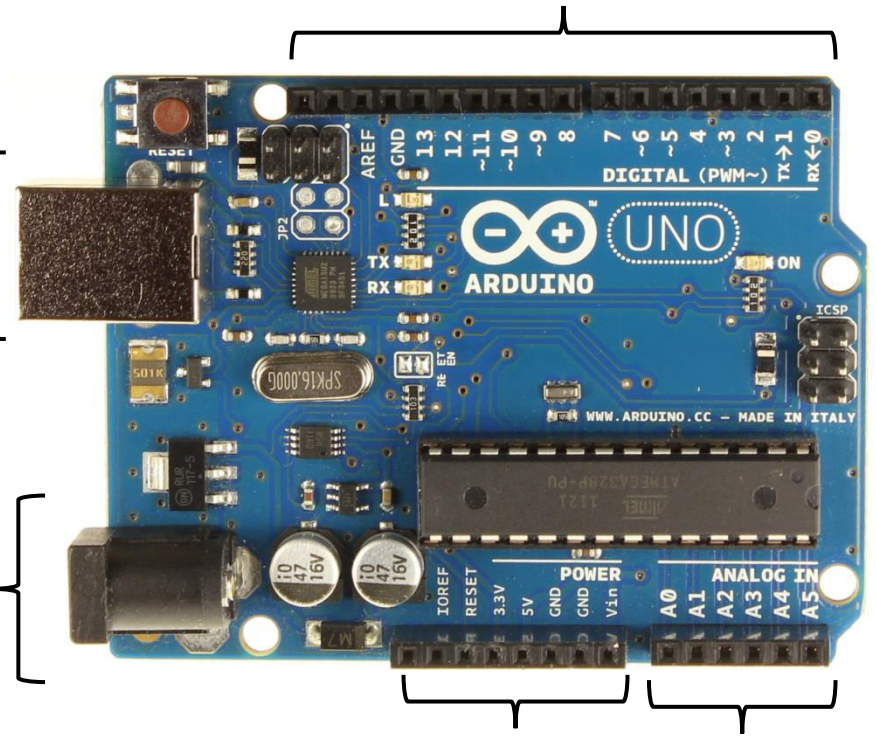
- Physical Computing – χρήση στοιχείων που μπορούν να αλληλεπιδρούν με το περιβάλλον γύρω μας
- Το Arduino αρχικά δημιουργήθηκε από σχεδιαστές και καλλιτέχνες για την πρωτότυπη αλληλεπίδραση ηλεκτρονικών ενδείξεων
  - Δημιουργήθηκε για μη σχετικούς με την επιστήμη των ηλεκτονικών/πληροφορικής
  - Απαιτεί τυπική γνώση αρχών προγρμματισμού
  - Ανοχή σφαλμάτων καλωδίωσης

Μοντέλα Arduino

Digital Input / Digital output
(PWM on pins 3, 5, 6, 9, 10, 11)

USB connection

7-12 volt input power (9v is common)



Arduino Uno Revision 3

Power pins

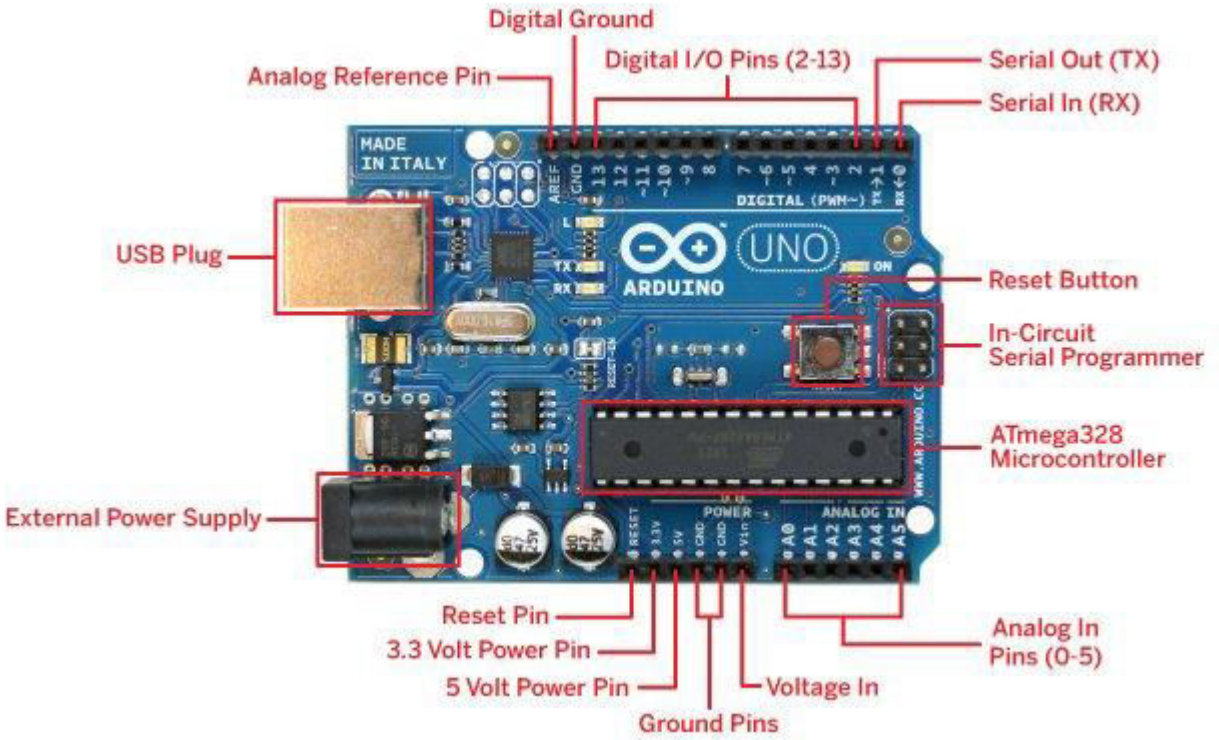
Analog input / Digital input or output

Δυνατότητες Εκμάθησης με τη χρήση του Arduino Uno

- Εισαγωγή στην ηλεκτρονική (τάση, ρεύμα, αντίσταση)
- Πώς λειτουργούν οι αισθητήρες
- Εισαγωγή στον προγραμματισμό
- Σχεδιασμός βασικού επιστημονικού εξοπλισμού
- Εύρεση λαθών/Επαλήθευση
- Συγκέντρωση στατιστικών και δεδομένων προς αξιολόγηση

Arduino

- Πλατφόρμα υλικού ανοιχτού κώδικα βασισμένο στον μικροελεγκτή Atmel AVR 8-bit και σε IDE βασισμένο σε C++
- Πάνω από 300000 boards έχουν σχεδιαστεί
- Το Arduino Due βασίζεται στο 32-bit ARM Cortex

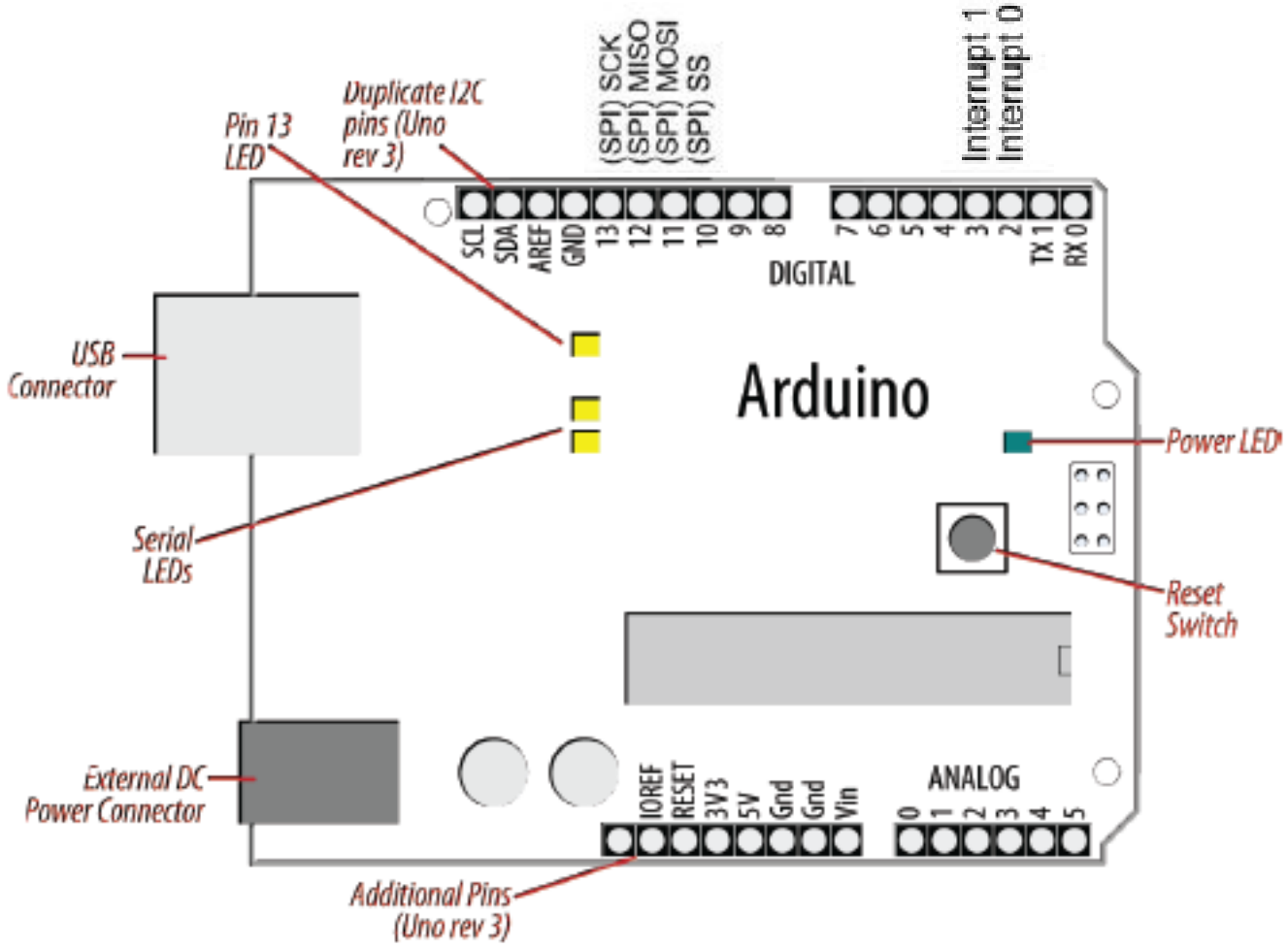


Microcontroller: ATMega328P

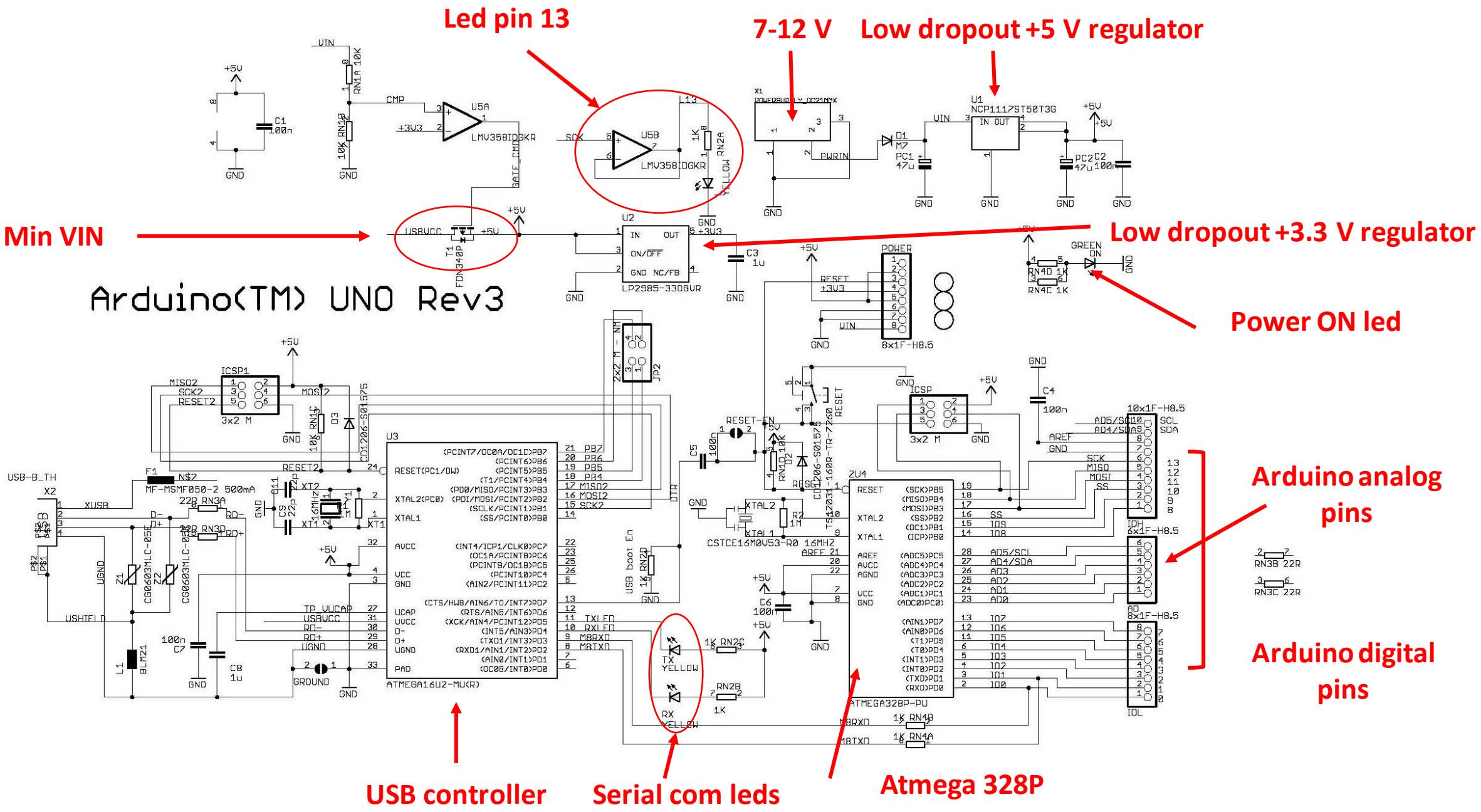
(http://www.atmel.com/devices/atmega328p.aspx?tab=documents)

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

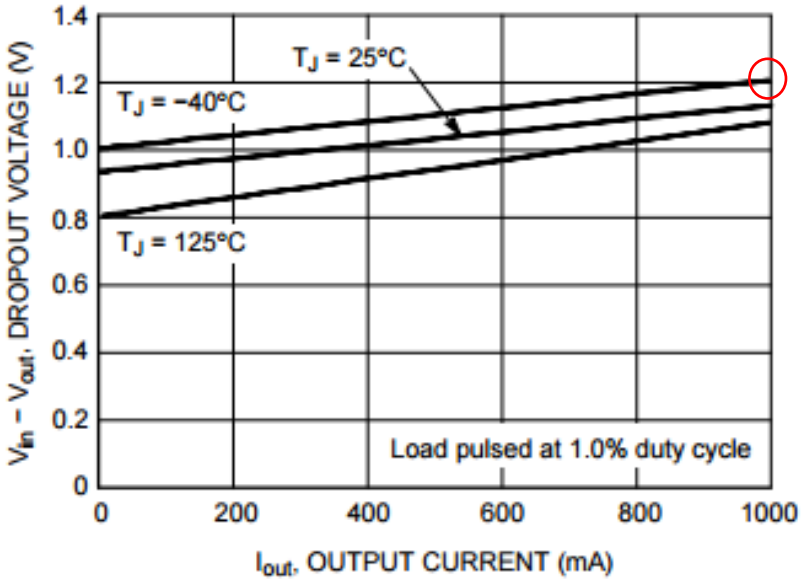
- Διαθέτει το ATMega16U2 σαν μετατροπέα USB-to-serial, για σύνδεση με υπολογιστή. Μπορεί να λειτουργήσει με την τροφοδοσία του USB για ρεύμα έως 500 mA.
- Ενσωματώνει ένα LED το οποίο είναι συνδεδεμένο με το pin 13.
- Διαθέτει την είσοδο AREF μέσω της οποίας μπορεί να δοθεί εξωτερική τάση αναφοράς στον 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)

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
ATMEGA16U2-MUR
ATMEGA328P-PU
LMV358IDGKR
LP2985-33DBVR
NCP1117ST50T3G
CSTCE16M0V53-R0 16MHZ

NCP1117-5 V (1 A)



Max dropout voltage=1.2 V

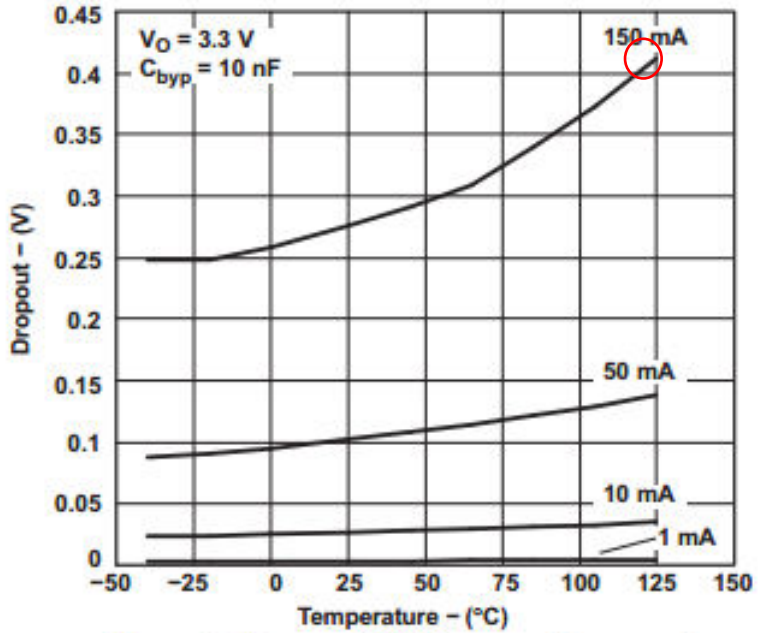


Figure 3. Dropout Voltage vs Temperature

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.

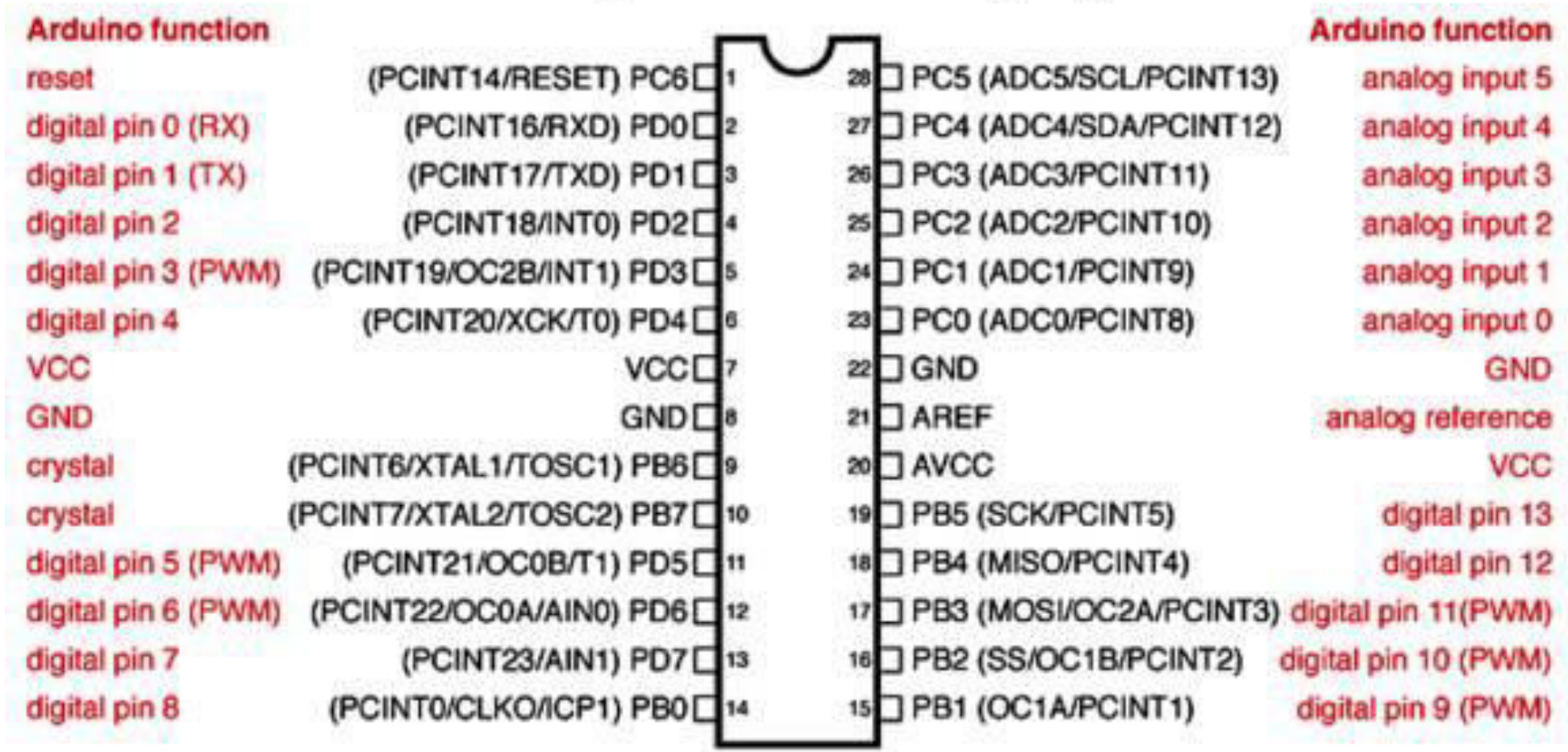
Analog pin 0 → General I/O ή Analog in (pin 23/PC0/ADC0 of μC)
Analog pin 1 → General I/O ή Analog in (pin 24/PC1/ADC1 of μC)
Analog pin 2 → General I/O ή Analog in (pin 25/PC2/ADC2 of μC)
Analog pin 3 → General I/O ή Analog in (pin 26/PC3/ADC3 of μC)
Analog pin 4 → General I/O ή Analog in (pin 27/PC4/ADC4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

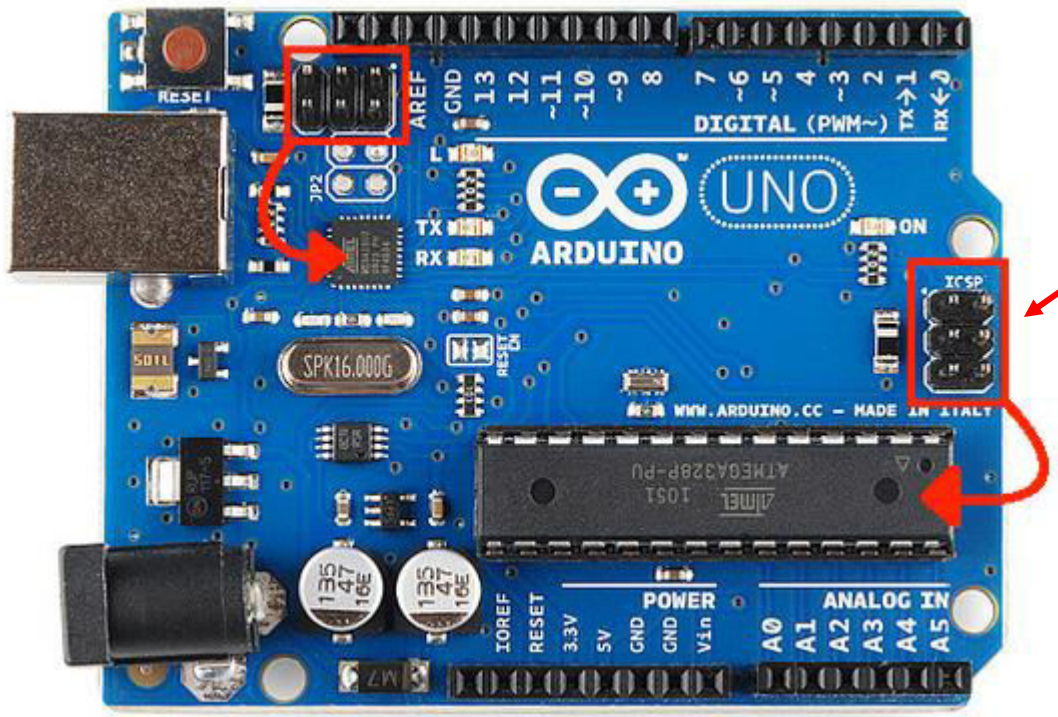
AREF → pin 21/AREF of μC (Εξωτερική είσοδος αναφοράς για τις αναλογικές εισόδους. Ο A/D είναι 10 bits με εσωτερική τάση αναφοράς 5 V. Δηλαδή μετατρέπει τάσεις από 0 έως 5V σε 0 έως 1024 ή 000h έως 3FFh).

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

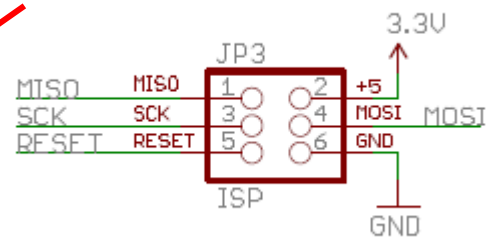
ATmega168/328 Pin Mapping



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)

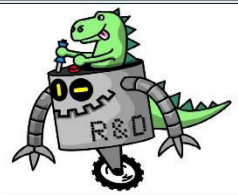


e.g. Plug a programmer to flash the bootloader (AVRISP mkII)

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
7U // Force unsigned
10L // Force long
15UL // 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)

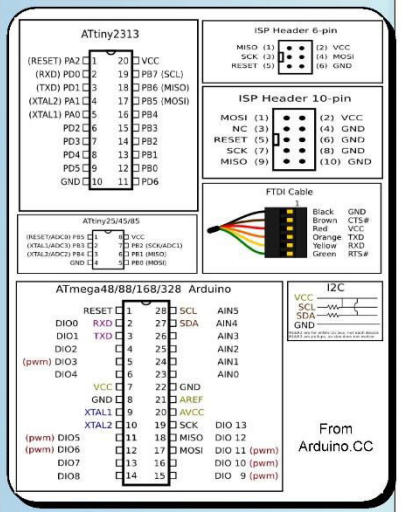
Servo (#include <Servo.h>)
attach(pin , [min_uS, max_uS])
write(angle) // 0-180
writeMicroseconds(uS) //1000-2000, 1500 is midpoint
read() // 0-180
attached() //Returns boolean
detach()

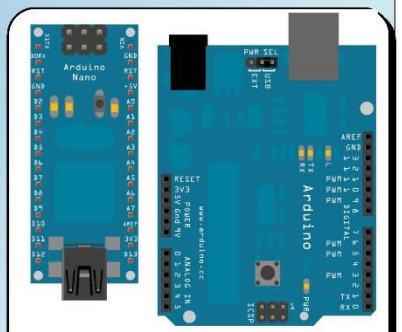
SoftwareSerial(RxPin,TxPin)
// #include<**SoftwareSerial**.h>
begin(longSpeed) // up to 9600
char **read**() // blocks till data
print(myData) or **println**(myData)

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 boobtloader)	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

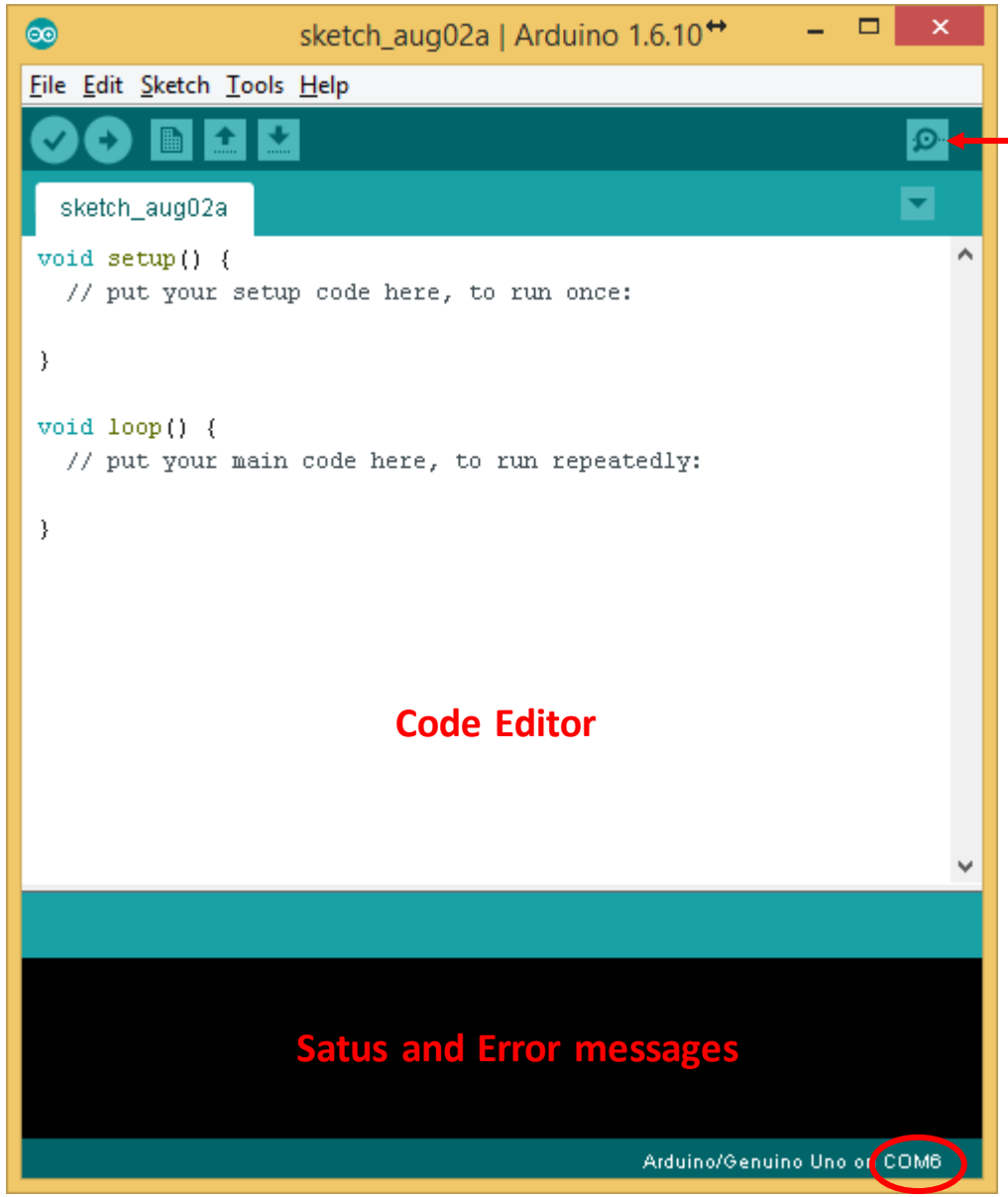
```
int x = 6;
int result = x << 1; // 6 shifted left 1 is 12
int result = x >> 2; // 12 shifted right 2 is 3;
```

```
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)

Βήματα Εκκίνησης

- Στην ιστοσελίδα: http://arduino.cc/en/Guide/HomePage
  1. **Εγκατάσταση του περιβάλλοντος Arduino (IDE)**
  2. **Σύνδεση του board στον υπολογιστή με USB κάλώδιο**
  3. **Αν χρειάζεται, εγκατάσταση οδηγών**
  4. **Εκτέλεση του Arduino IDE**
  5. **Επιλογή board**
  6. **Επιλογή σειριακής θύρας**
  7. **Άνοιγμα του παραδείγματος blink**
  8. **Εγκατάσταση του προγράμματος στο Arduino board**



Serial monitor

Compile

Upload to board

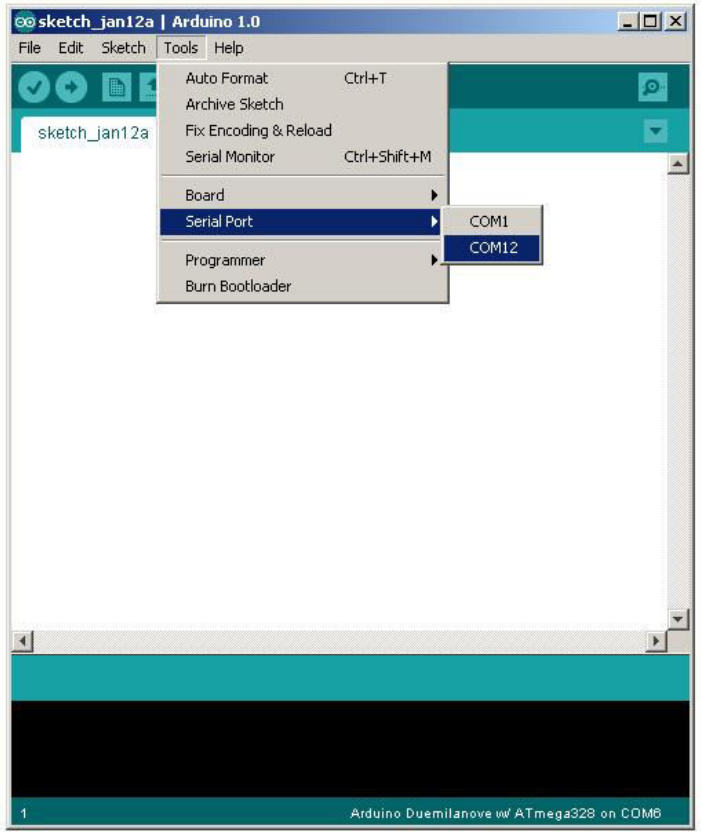
New sketch

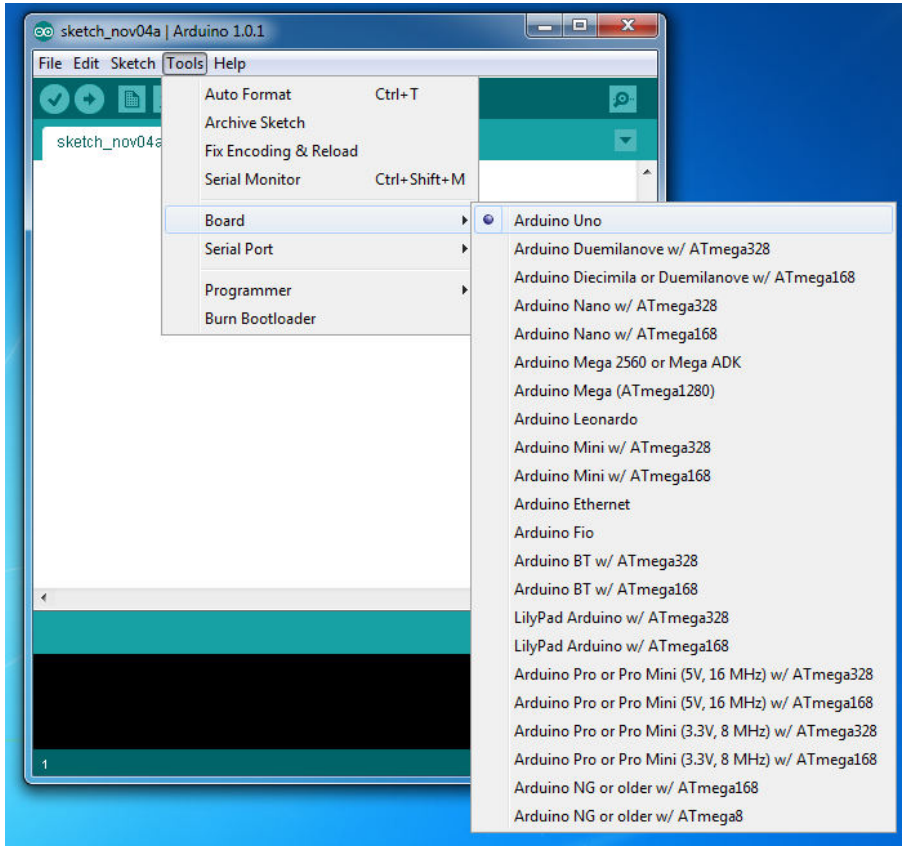
Open existing sketch

Save sketch

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

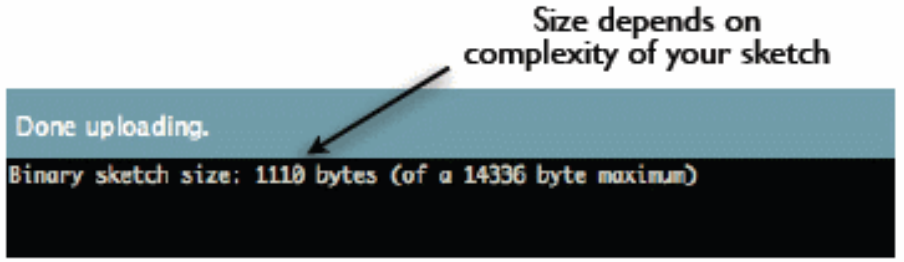
Select Serial Port and Board



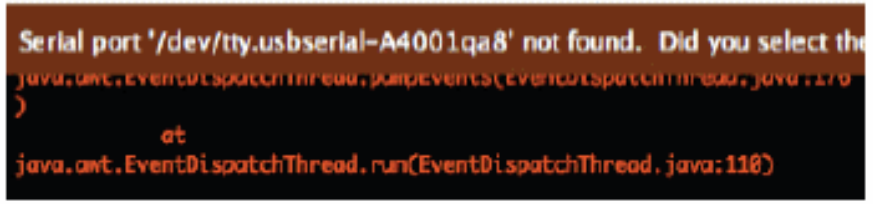


Μηνύματα Κατάστασης

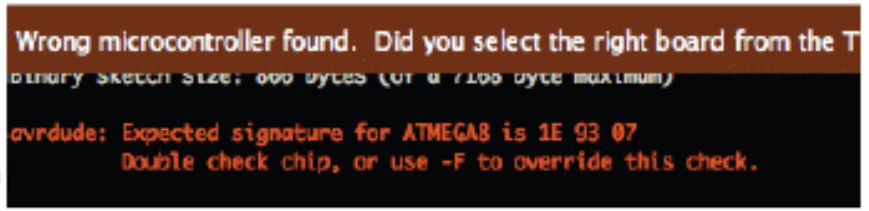
Uploading worked



Wrong serial port selected



Wrong board selected

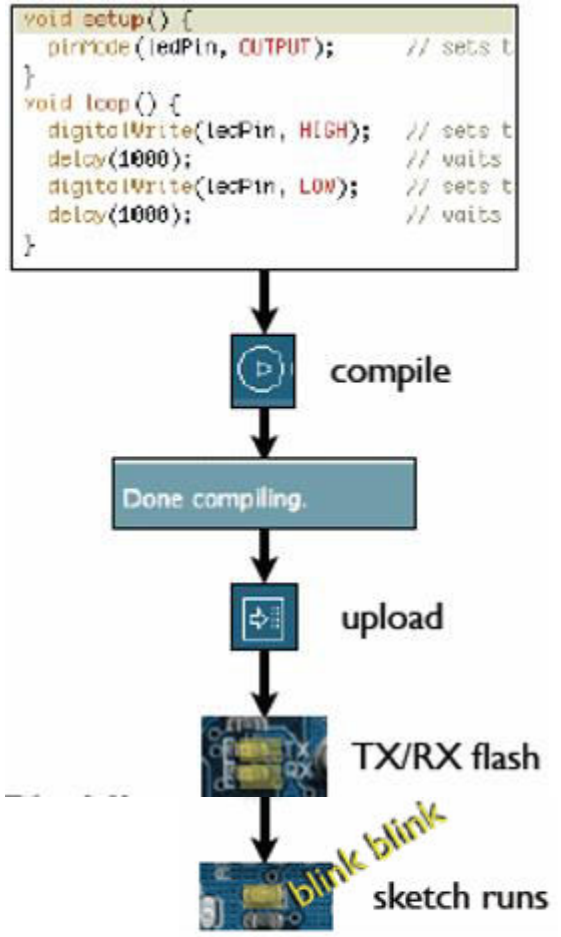


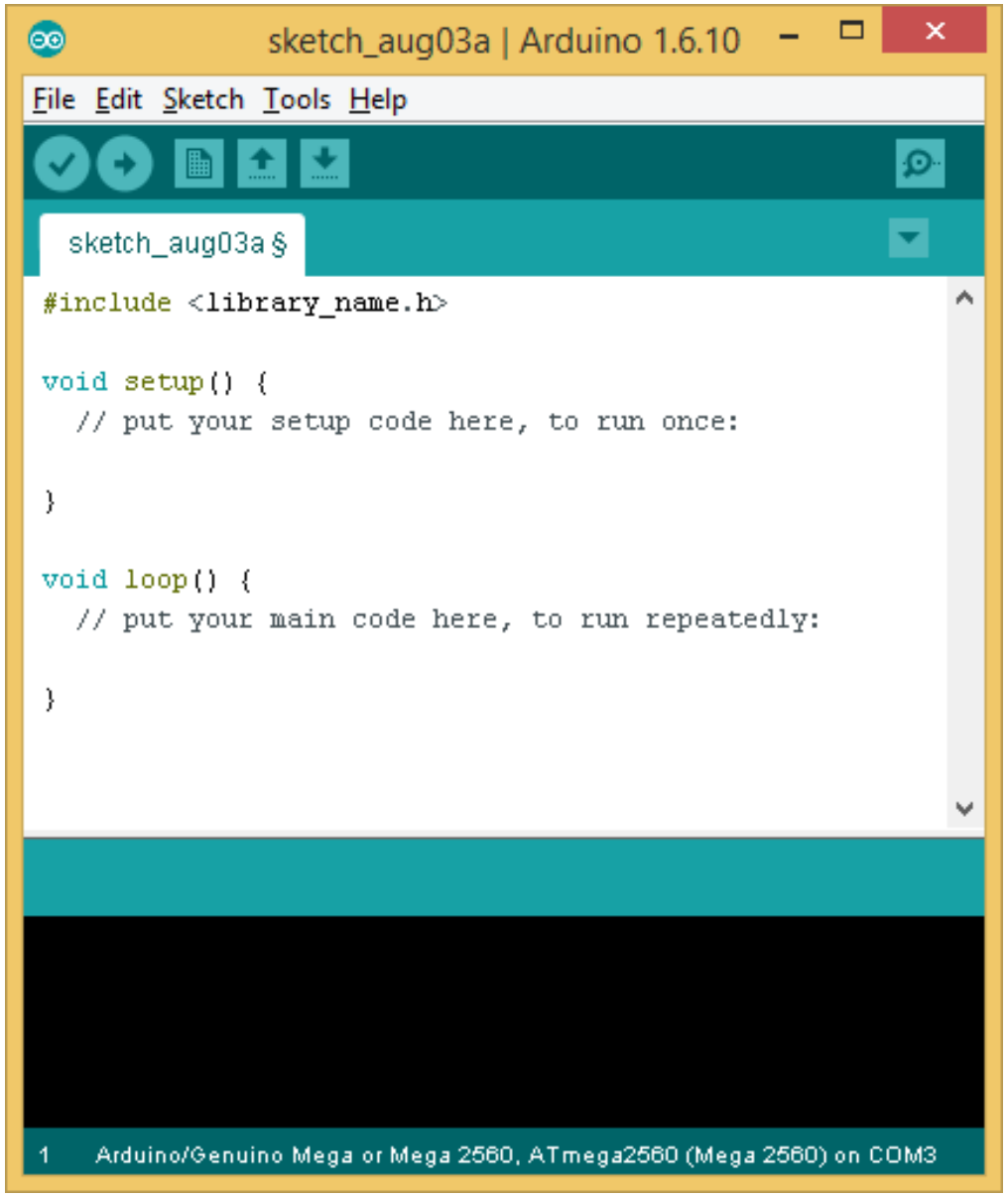
nerdy cryptic error messages

todbot.com/blog/bionicarduino

Παράδειγμα Χρήσης Arduino

- Εγγραφή κώδικα
- Compile
- Upload





- 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. Χρησιμοποιώντας το Arduino Shield αναβοσβήστε τα LED εναλλάξ και διαδοχικά.