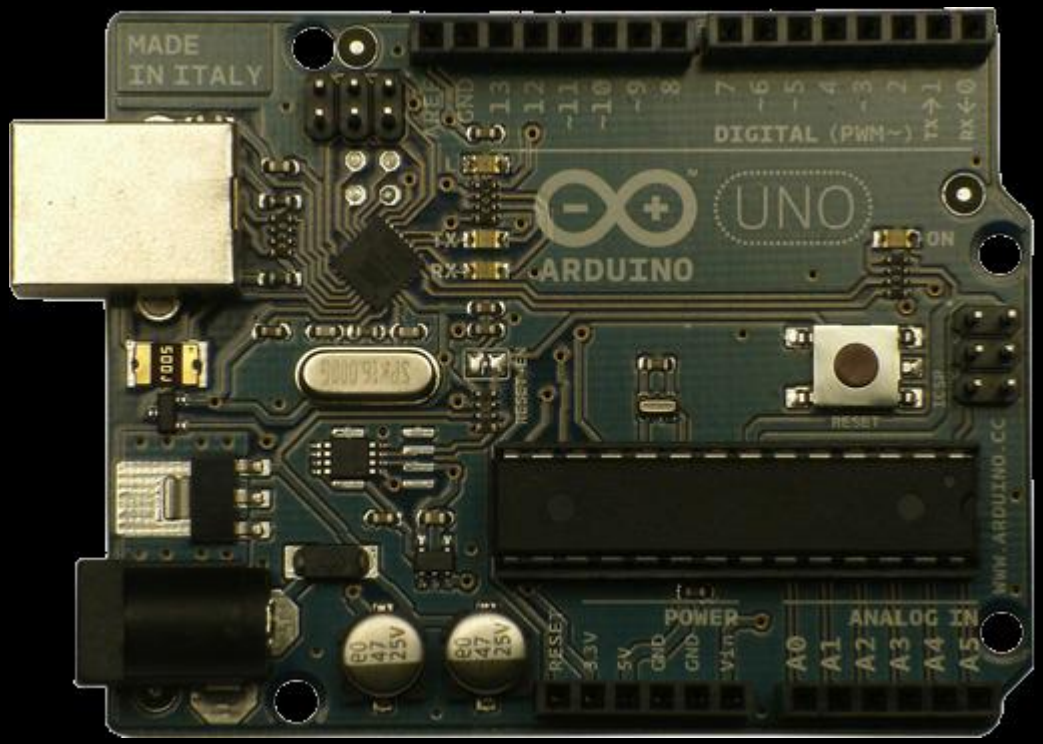
HAB 621 Instrumentation and Measurement in Biomechanics

Çizim: De Motu Animalium G.Borelli (1680)

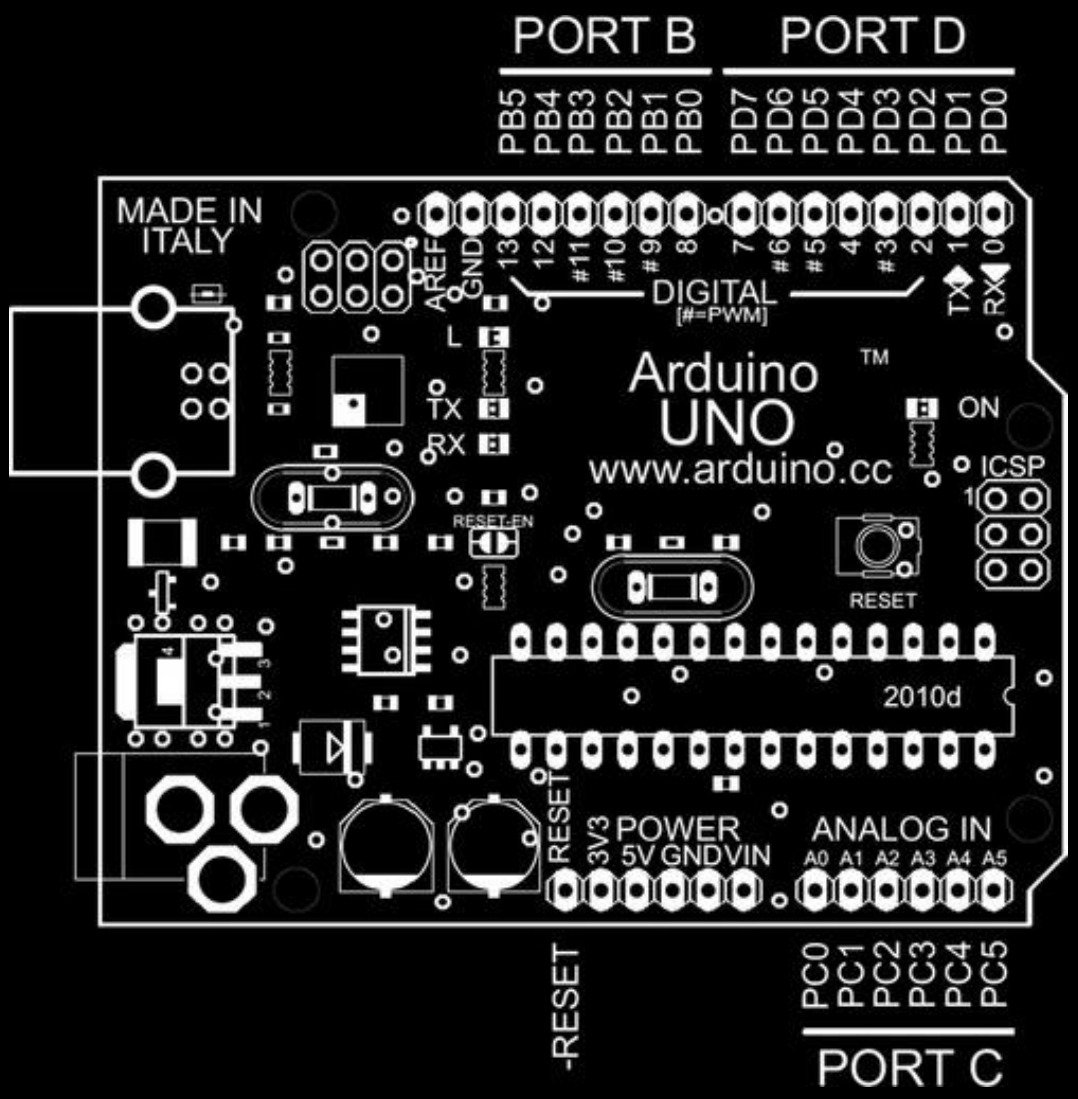
Serdar Arıtan
serdar.aritan@hacettepe.edu.tr
Hacettepe Üniversitesi
www.hacettepe.edu.tr
Spor Bilimleri Fakültesi
www.sbt.hacettepe.edu.tr
Biyomekanik Araştırma Grubu
www.biomech.hacettepe.edu.tr

What Is an Arduino?

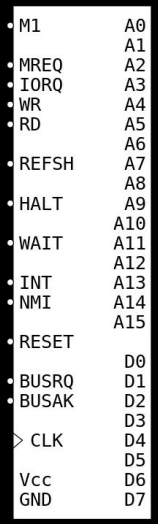


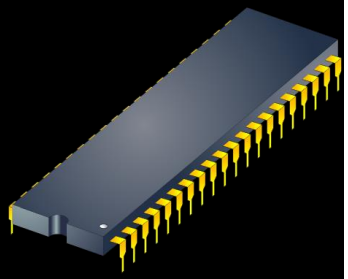
The name ***Arduino*** is a masculine Italian name meaning "**strong friend**." Being a proper name, ***Arduino*** is always capitalized.

What Is an Arduino?



The Amstrad CPC (short for Colour Personal Computer) is a series of 8-bit home computers produced by Amstrad between 1984 and 1990.





Atmel

ATmega48A/PA/88A/PA/168A/PA/328/P

ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH

DATASHEET

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

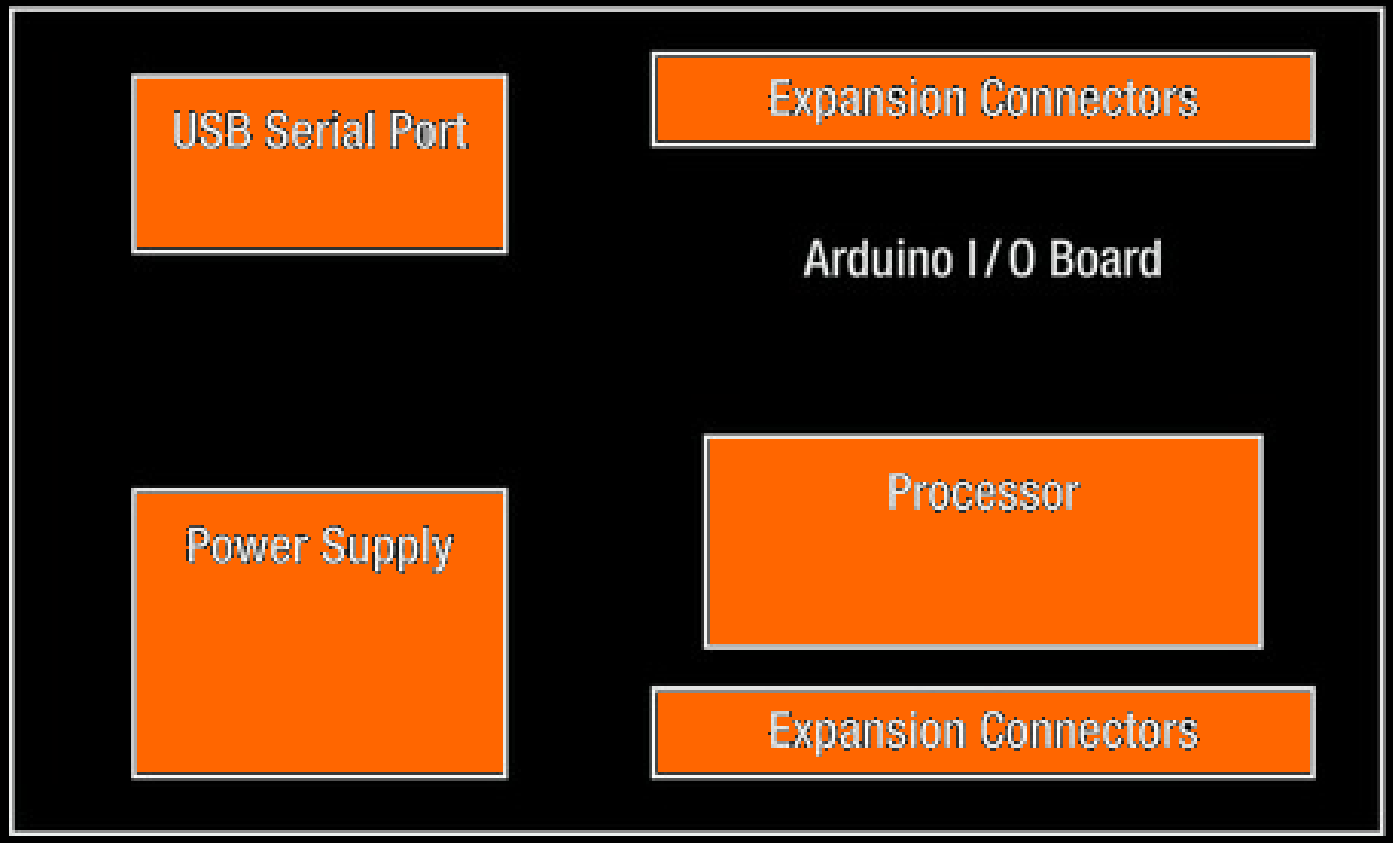
What Is an Arduino?

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

- High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

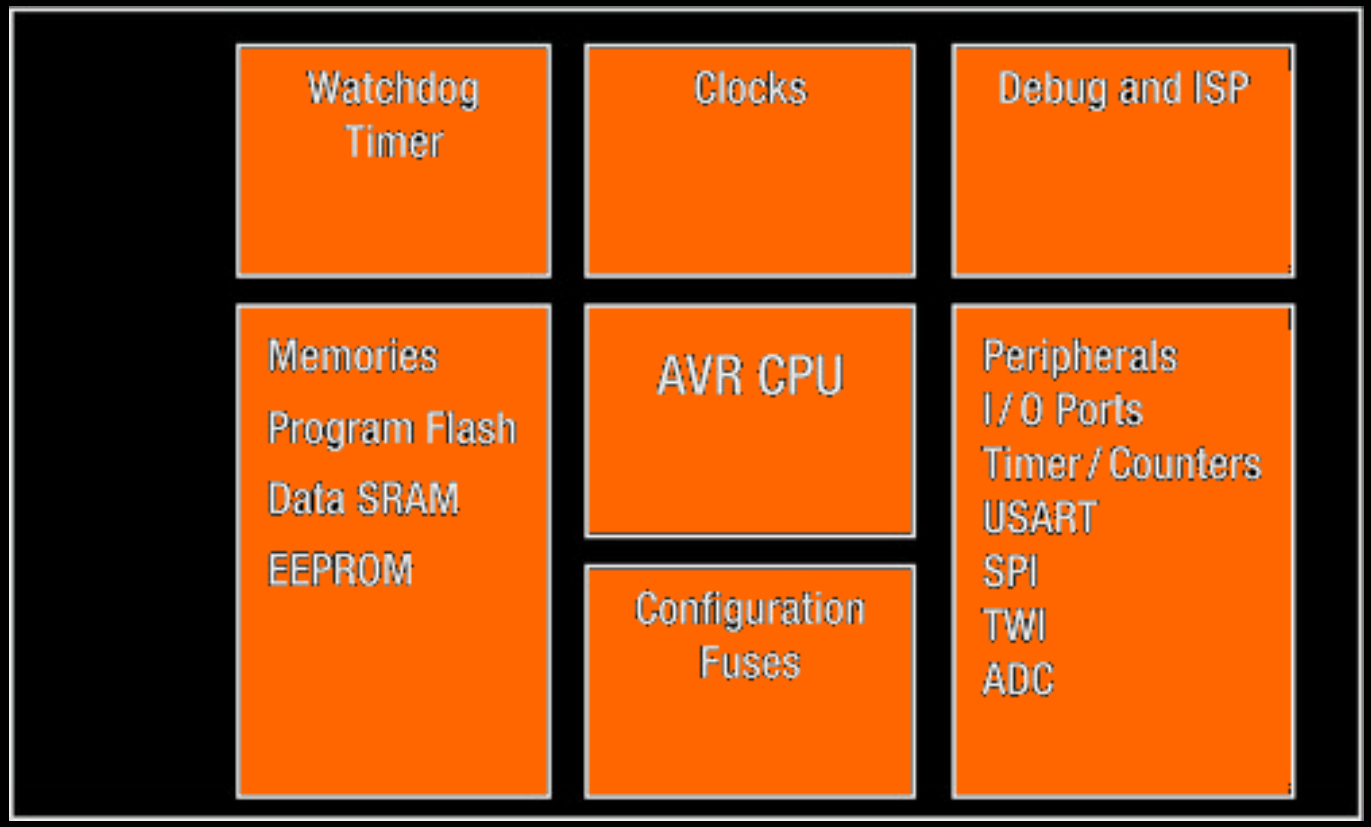

(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

What Is an Arduino?



The main brain of the **Arduino Uno** is the Atmel AVR **ATmega328**

What Is an Arduino?

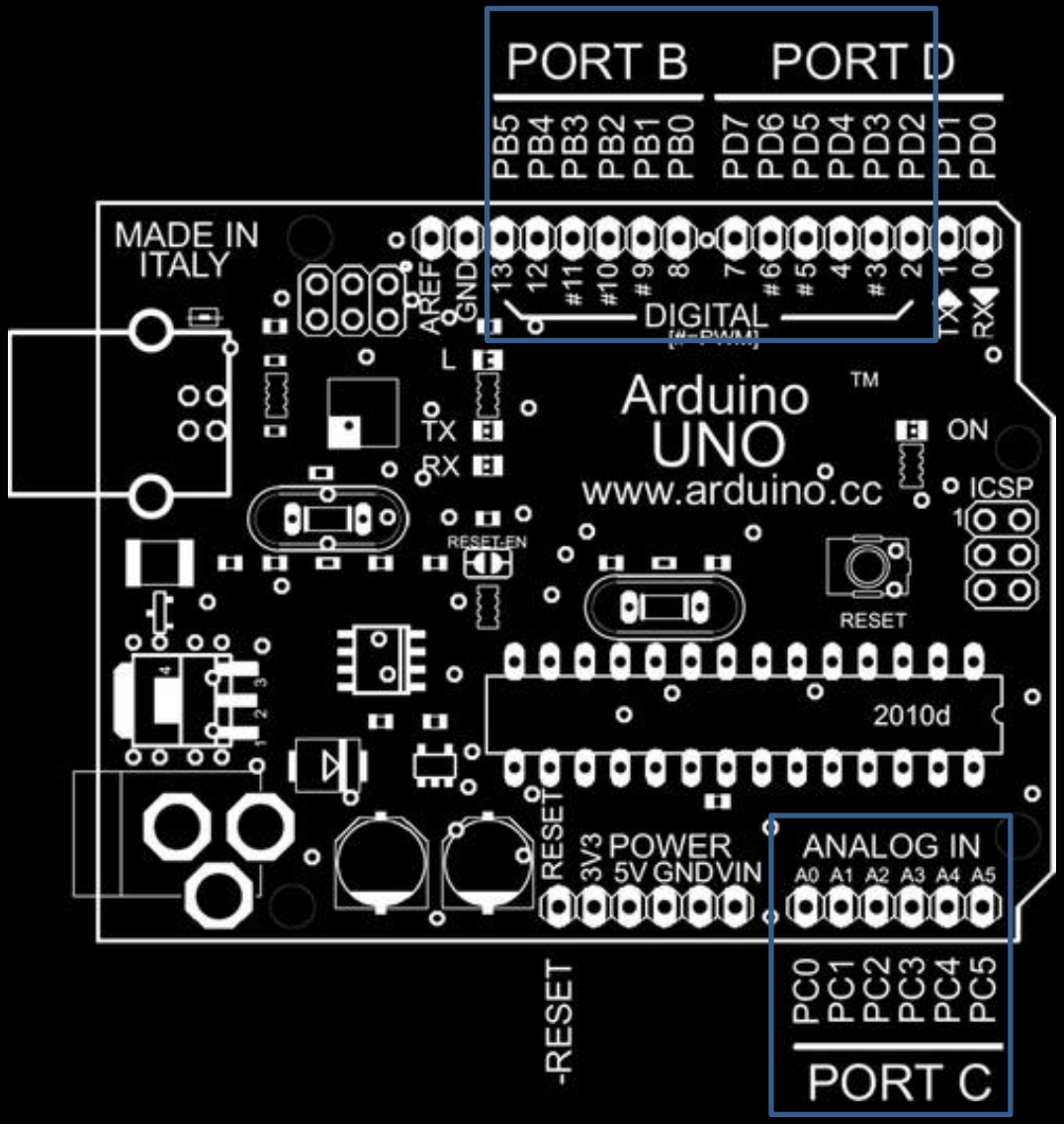


This device is essentially a computer on a chip, containing a central processing unit (CPU), memory arrays, clocks, and peripherals in a single package

What Is an Arduino?

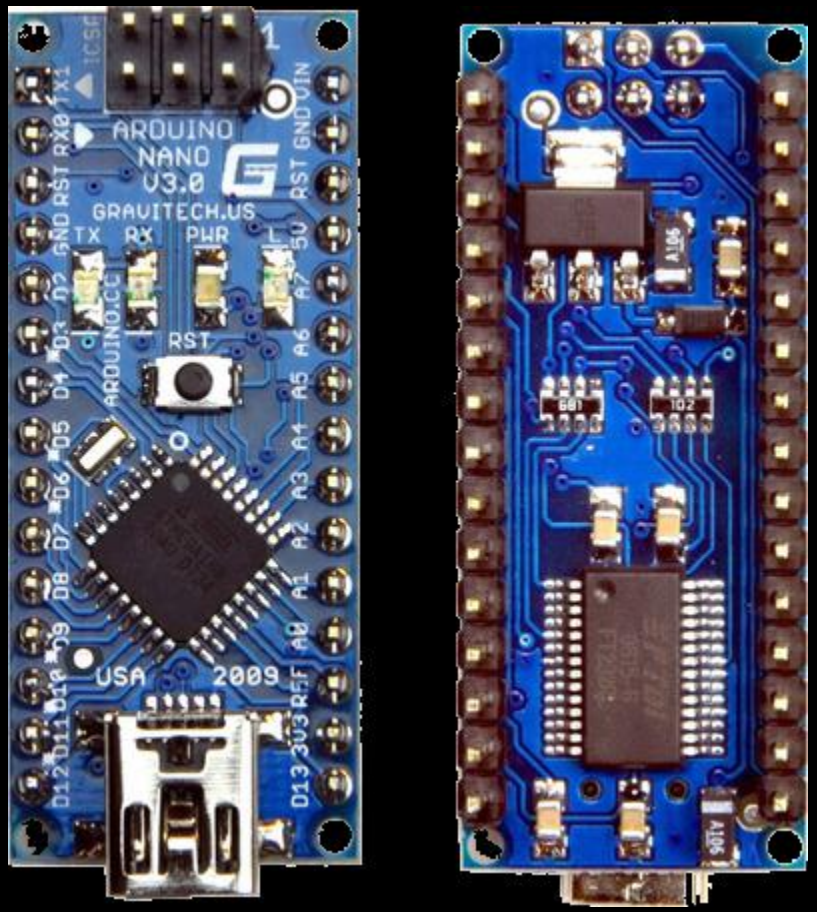
The **ATmega328** chip is derived from the original Arduino processor, the ATmega8. It contains more memory and more peripheral capability than its predecessor while using less power. The **ATmega328** processor can operate from a wide range of power-supply voltages, from **1.8V to 5.5V**. This makes it well suited for battery-powered applications. At the lowest voltages, the processor has a maximum clock rate of **4MHz** (millions of cycles per second). Increase the supply voltage to at least **2.7V**, and you can increase the clock rate to 10MHz. To run at the rated maximum clock rate of 20MHz, the chip needs at least 4.5V. The Arduino I/O Board provides 5.0V for the **ATmega328** chip, so it can run at any speed, up to the maximum of **20MHz**. The current crop of ATmega328 chips from Atmel feature the company's picoPower technology,

What Is an Arduino?

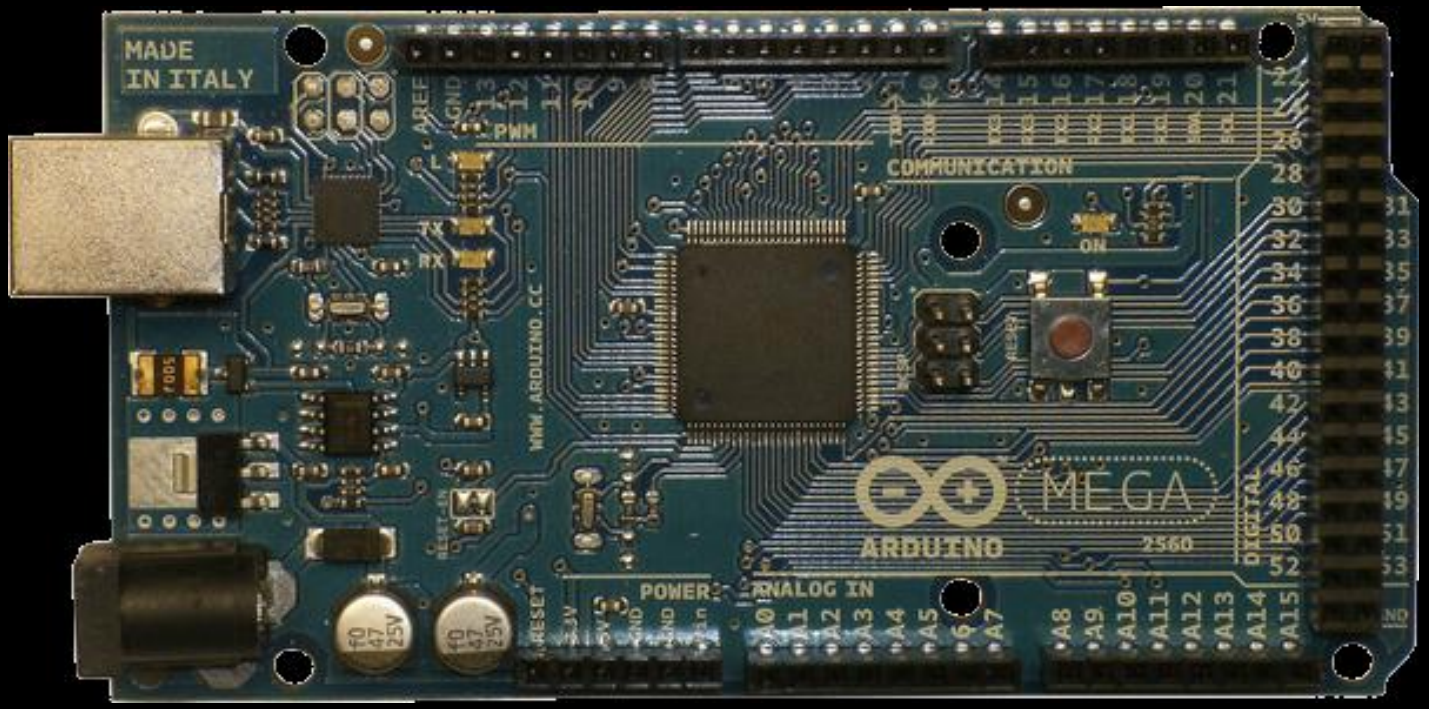


Arduino Shield

Arduino Nano



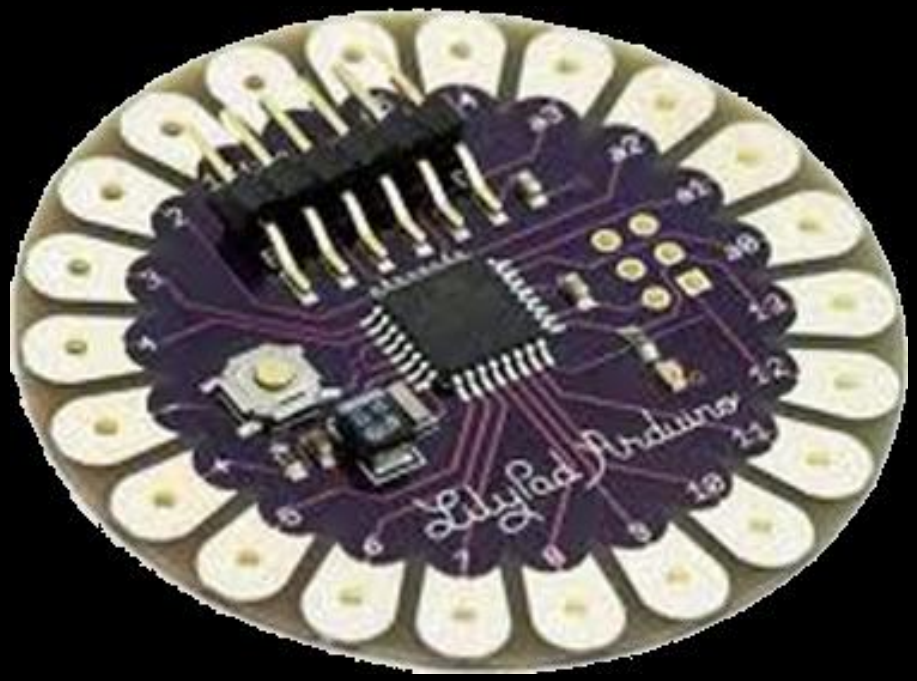
Arduino Mega



What is the difference ?

Specification	Arduino Uno	Arduino Mega 1280	Arduino Mega 2560
Processor	ATmega328	ATmega1280	ATmega2560
Program memory	32KB	128KB	256KB
Data memory	2KB	8KB	8KB
EEPROM	1KB	4KB	4KB
Device pins	28/32*	100	100
Digital I/O pins	14	54	54
Analog inputs	6	16	16
PWM outputs	6	14	14
Serial ports	1	4	4

The LilyPad Arduino



The LilyPad Arduino was developed by Leah Buechley of the MIT Media Lab. It's designed to be sewn into fabric using conductive thread to produce e-textiles, or wearable computers.

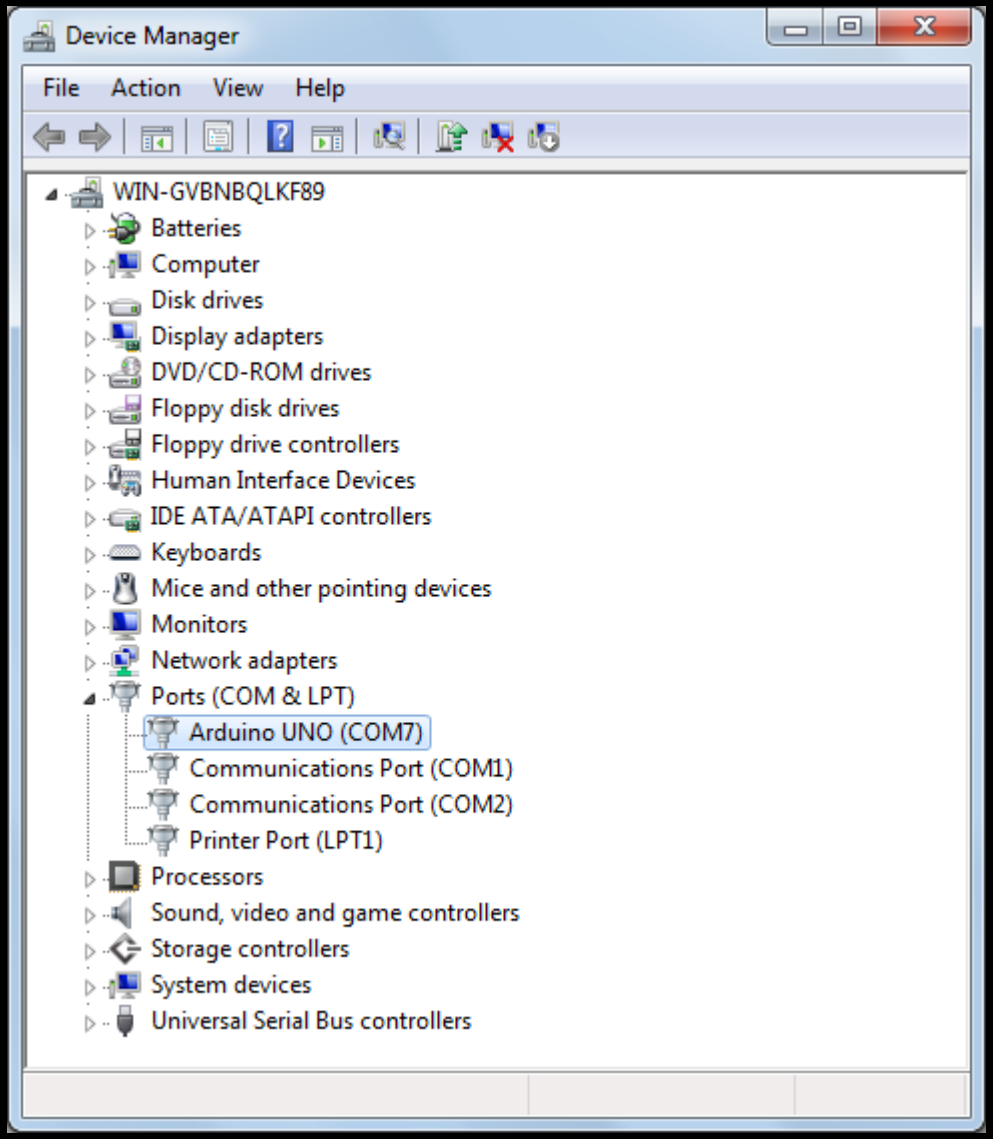
How to install Arduino drivers

Installing Drivers: Windows

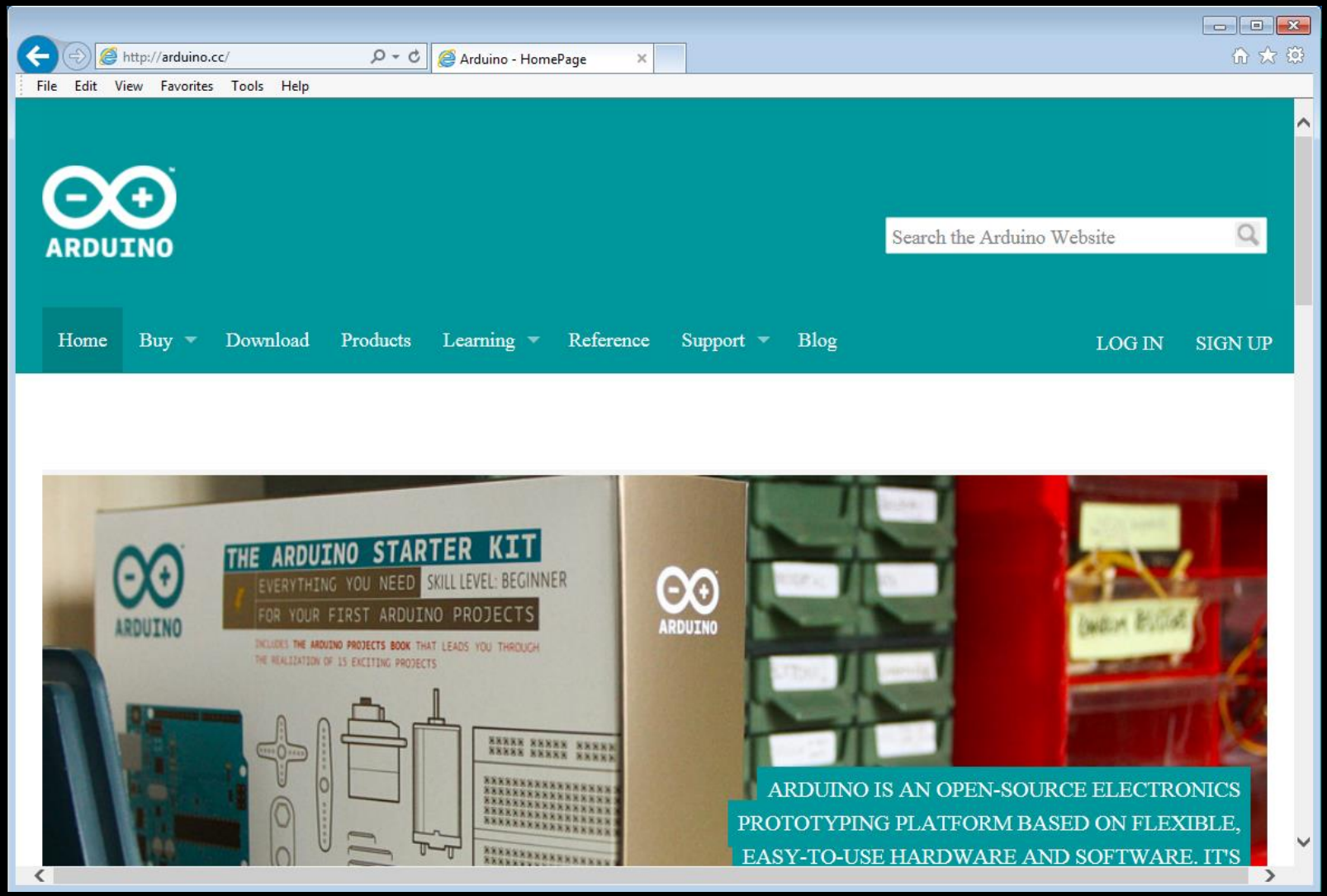
Plug the Arduino board into the computer; when the Found New Hardware Wizard window comes up, Windows will first try to find the driver on the Windows Update site.
Windows XP will ask you whether to check Windows Update; if you don't want to use Windows Update, select the "No, not at this time" option and click Next.

On the next screen, choose "Install from a list or specific location" and click Next. Navigate to and select the Uno's driver file, named *ArduinoUNO.inf*, located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). Windows will finish up the driver installation from there.

How to install Arduino drivers



https://www.arduino.cc/



http://arduino.cc/en/Main/Software

Download the Arduino IDE

ARDUINO 1.8.8

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

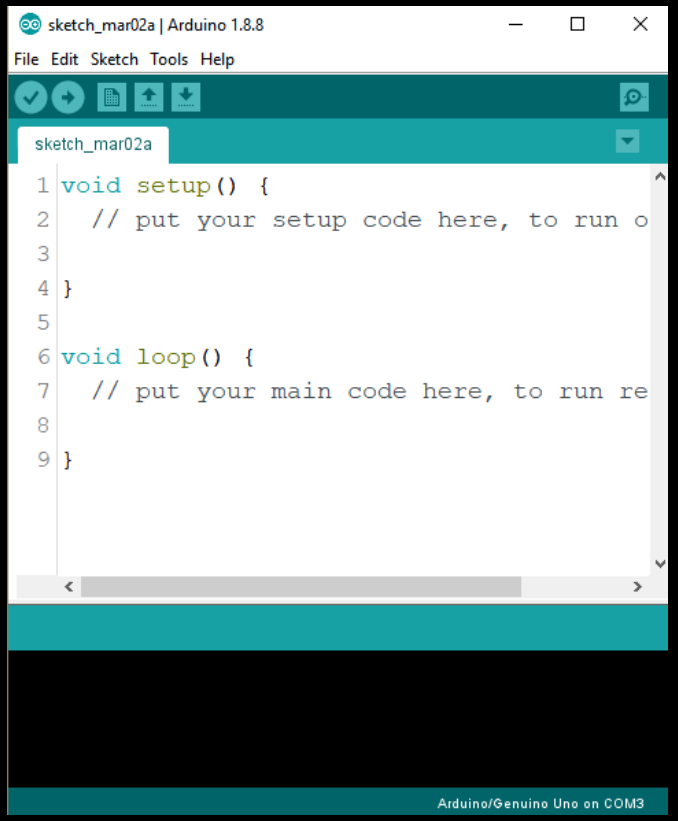
Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

The Bare Minimum Arduino Sketch

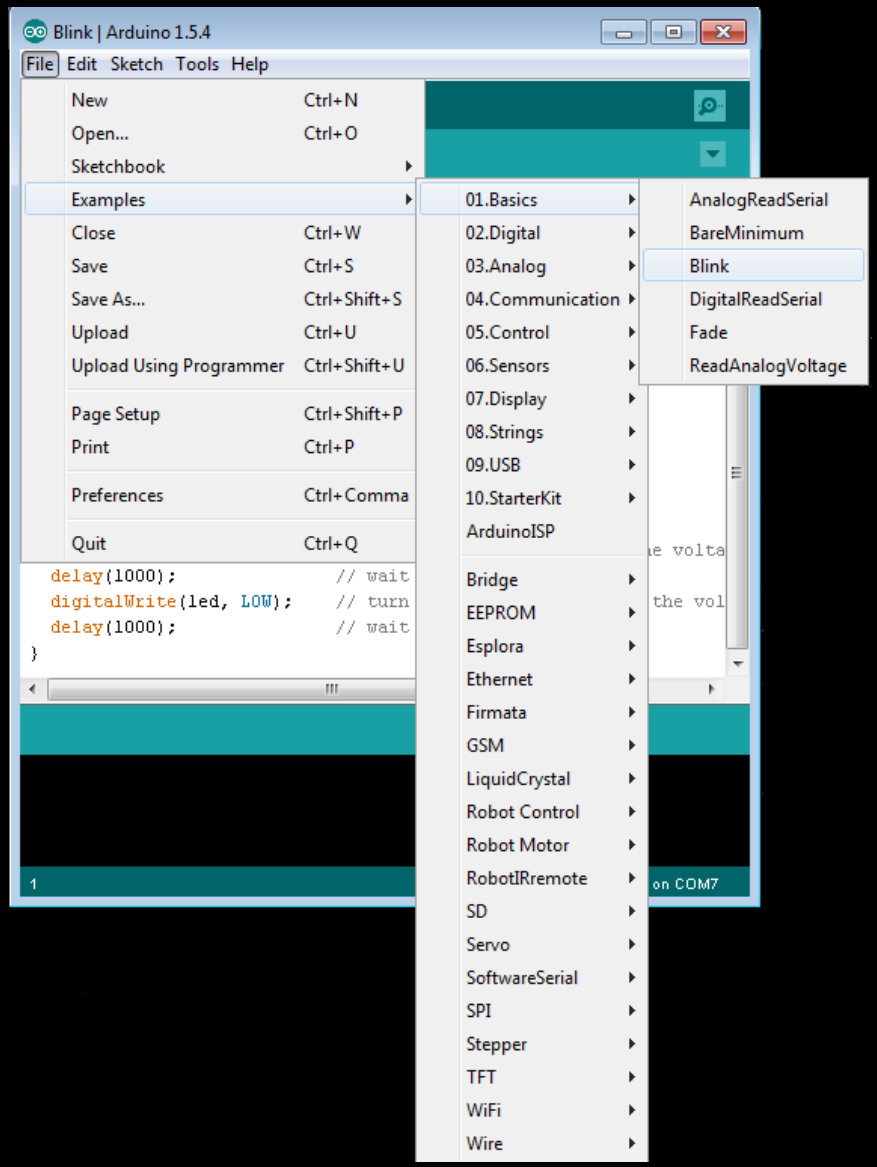
```
void setup() {
}

void loop() {
}
```



The `setup()` and `loop()` function definitions are required, even if they do nothing. If they're missing, you get a nasty "undefined reference to …" error message.

"Hello, World" in the Arduino Programming Language

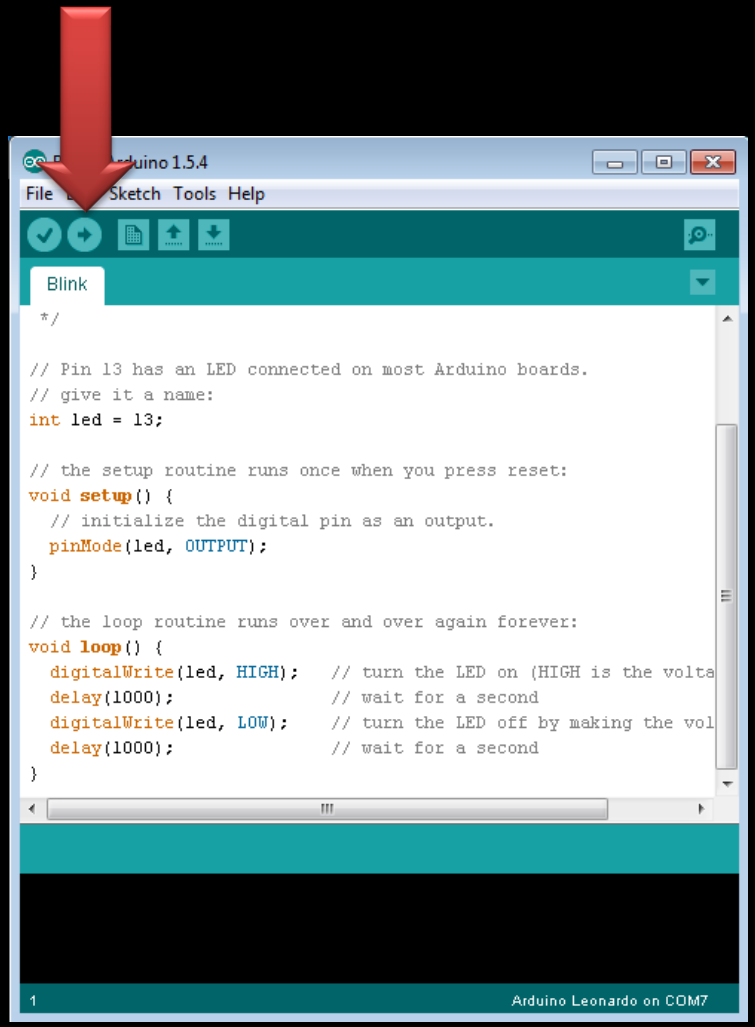
```
void setup() {
      Serial.begin(9600);
      Serial.println("Hello, world!");
}

void loop() {
}
```


Blink | Arduino 1.5.4
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit
ArduinoISP
Bridge
EEPROM
Esplora
Ethernet
Firmata
GSM
LiquidCrystal
Robot Control
Robot Motor
RobotIRremote
SD
Servo
SoftwareSerial
SPI
Stepper
TFT
WiFi
Wire
AnalogReadSerial
BareMinimum
Blink
DigitalReadSerial
Fade
ReadAnalogVoltage
delay(1000); // wait
digitalWrite(led, LOW); // turn
delay(1000); // wait
}
on COM7


Blink | Arduino 1.5.4
File Edit Sketch Tools Help
Blink
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH); // turn the LED on (HIGH is the volta
delay(1000); // wait for a second
digitalWrite(led, LOW); // turn the LED off by making the vol
delay(1000); // wait for a second
}
1
Arduino Leonardo on COM7

One Button Does It All

```
#define LED 13
void setup() {
        pinMode(LED, OUTPUT);
}
void loop() {
        digitalWrite(LED, HIGH);
        delay(1000); // one-second delay
        digitalWrite(LED, LOW);
        delay(1000); // one-second delay
}
```



```
#define LED 13
void setup() {
      pinMode(LED, OUTPUT); // D13 is an output
}
void loop() {
      digitalWrite(LED, HIGH); // turn on the LED
      delay(1000); // one-second delay
      digitalWrite(LED, LOW); // turn off the LED
      delay(1000); // another one-second delay
}
```

The first line, **`#define LED 13`**, is a *macro definition*. This allows you to assign a value to a meaningful name.

Remember, it helps the humans when source code can be read by humans.

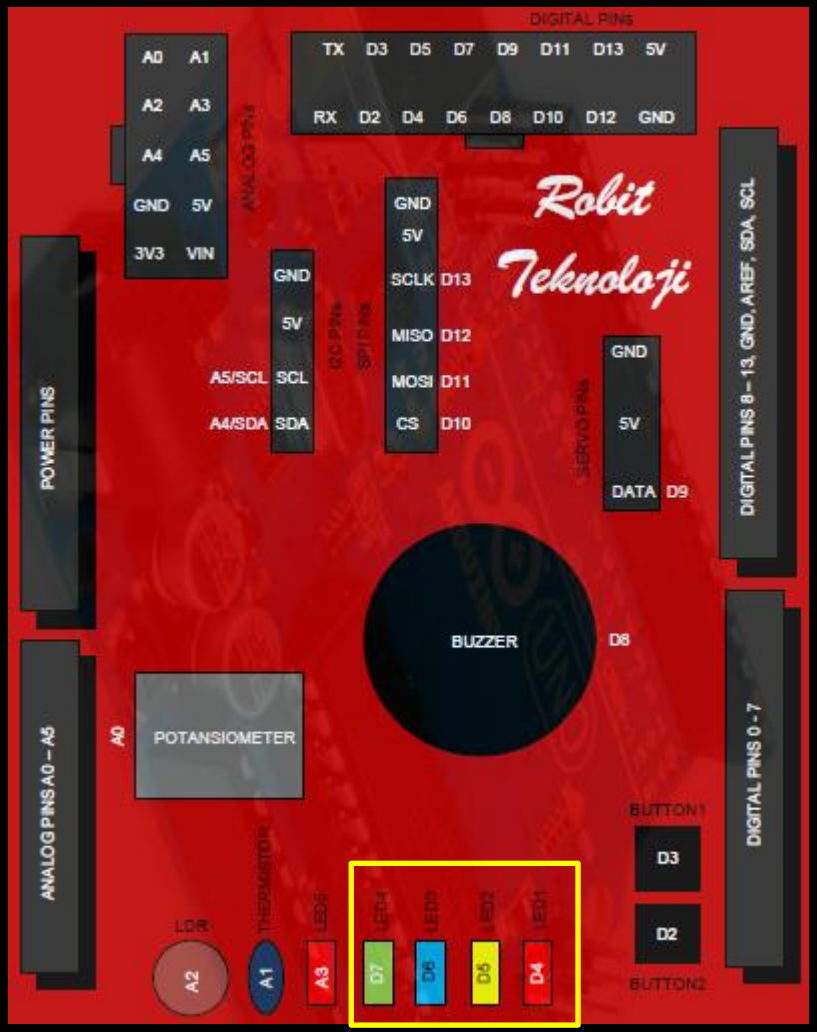
```
#define LED 13
void setup() {
      pinMode(LED, OUTPUT); // D13 is an output
}
void loop() {
      digitalWrite(LED, HIGH); // turn on the LED
      delay(1000); // one-second delay
      digitalWrite(LED, LOW); // turn off the LED
      delay(1000); // another one-second delay
}
```

The first function is **digitalWrite()**, which writes a digital (one or zero) value to a device pin. The predefined values **HIGH** and **LOW** correspond to the values one and zero, respectively. Writing a one (**HIGH**) to digital pin 13 causes the LED to turn on. A zero, or **LOW** value, causes the LED to turn off.

```
#define LED 13
void setup() {
      pinMode(LED, OUTPUT); // D13 is an output
}
void loop() {
      digitalWrite(LED, HIGH); // turn on the LED
      delay(1000); // one-second delay
      digitalWrite(LED, LOW); // turn off the LED
      delay(1000); // another one-second delay
}
```

The **`delay()`** function wastes a bit of time. The exact amount of time is specified in *milliseconds* (thousandths of a second) and passed as the argument. This results in a one-second delay between the LED changing state.

Try to turn On and Off LEDs



GNU Binutils

The name GNU Binutils stands for "Binary utilities". It contains the GNU assembler (gas), and the GNU linker (ld), but also contains any other utilities that work with binary files that are created as part of the software development toolchain. Again, when these tools are built for the AVR target, the actual program names are prefixed with "avr-".

GNU Binutils

For example in, `X:\arduino-1.8.8\hardware\tools\avr\bin`

avr-as	The Assembler.
avr-ld	The Linker.
avr-ar	Create, modify, and extract from libraries (archives).
avr-ranlib	Generate index to library (archive) contents.
avr-objcopy	Copy and translate object files to different formats.
avr-objdump	Display information from object files including disassembly.
avr-size	List section sizes and total size.
avr-nm	List symbols from object files.
avr-strings	List printable strings from files.
avr-strip	Discard symbols from files.
avr-readelf	Display the contents of ELF format files.
avr-addr2line	Convert addresses to file and line.
avr-c++filt	Filter to demangle encoded C++ symbols.

This isn't what happens at all

The Arduino software takes your sketch and combines it and then passes it to the **avr-gcc** compiler

```
void setup(void); // setup() function prototype
void loop(void);  // loop() function prototype

int main(void) {
        setup(); // perform one-time initializations are required

        while(1) {
                loop(); // repeat this over and over again
        }
        return 0; // this never happens
}
```

This isn't what happens at all

```
#include <avr/io.h>
int main(void) {
        long i;
        DDRB = 1<<5; // PB5/D13 is an output
        while(1) {
                PORTB = 1<<5; // LED is on
                for(i = 0; i < 100000; i++); // delay
                PORTB = 0<<5; // LED is off
                for(i = 0; i < 100000; i++); // delay
        }
}
```

The **#include <avr/io.h>** compiler directive reads in a long list of predefined values that pertain to your specific processor. These include the names and addresses of the I/O ports (**DDRB**, **PORTB**) that are referenced in the program. The processor is identified in a command-line option passed to the compiler.

The **long i;** is a long integer declaration, which reserves a spot for a 32-bit value. This value is used as a counter to kill some time.

This isn't what happens at all

```
#include <avr/io.h>
int main(void) {
        long i;
        DDRB = 1<<5; // PB5/D13 is an output
        while(1) {
                PORTB = 1<<5; // LED is on
                for(i = 0; i < 100000; i++); // delay
                PORTB = 0<<5; // LED is off
                for(i = 0; i < 100000; i++); // delay
        }
}
```

The 1<<5 notation represents a binary 1 shifted left 5 times. This is only one of many ways to represent this value using C. You can also use binary notation **0b00100000**, hexadecimal **0x20**, or decimal **32**. Name your new C source file **blink.c**, and save it in a convenient location. Now open a command line window and navigate to this location. Enter the following command to compile this simple program:

```
avr-gcc -mmcu=atmega328p blink.c -o blink.o
```

This isn't what happens at all

```
avr-gcc -mmcu=atmega328p blink.c -o blink.o
```

Assuming everything goes well, the **avr-gcc** compiler reads your source file, converts it into machine language instructions specifically for the ATmega328P (the **-mmcu=atmega328p** command-line option) and writes it out to a file called **blink.o**, an object file.

You now need to convert this object file into a binary image file that you can program using the **avrdude** utility. This conversion is accomplished with the **avr-objcopy** utility, which speaks a variety of object-file dialects. Type this command to perform the conversion:

```
avr-objcopy -O ihex blink.o blink.hex
```

Now you have a file suitable for uploading to the Arduino Uno. You use the **avrdude** utility to send the bits over the wire to the Arduino, like this:

```
avrdude -p atmega328p -c stk500v1 -P \\.\COM11 -U
flash:w:blink.hex:i
```

This isn't what happens at all

The **avrdude** command syntax is a bit more complex than the previous commands you've used. Let's look at each of the options and see what they do.

• **-p atmega328p**: Tells **avrdude** what kind of chip you want to program. This is similar to the **-mmcu** option for the **avr-gcc** compiler. Different AVR chips use slightly different programming algorithms, and **avrdude** needs to know which one to use.

• **-c stk500v1**: Indicates that the STK500 version 1 protocol is to be used to negotiate the programming of the chip. This is the protocol spoken by the Arduino bootloader.

• **-P \\.\COM11**: Tells **avrdude** which serial port to use. Your serial port will most likely be different. The **\\.** gibberish is needed under Windows, for some reason, after you've exceeded the single-digit COM ports (1–9).

• **-U flash:w:blink.hex:i**: The complete memory programming instruction, pointing out the memory area of interest (the flash program memory), a *w* for write, the file name (**blink.hex**), and the file format (**i** = Intel HEX format).