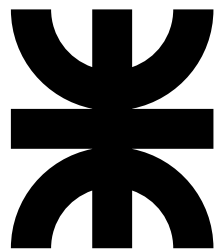
Informática II

Introducción a Arduino

Gonzalo F. Pérez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2017 –

Introducción a Arduino

Un poco de historia

Es una compañía de hardware libre y una comunidad tecnológica que diseña y fabrica Hw y Sw de desarrollo (placa, software, y lenguaje de programación).

Introducción a Arduino

Un poco de historia

Es una compañía de hardware libre y una comunidad tecnológica que diseña y fabrica Hw y Sw de desarrollo (placa, software, y lenguaje de programación).

- Inicia en el año 2005 como proyecto estudiantil en el Instituto de diseño interactivo Ivrea (Italia). [cierra en 2005]
- Fundador: Massimo Banzi (profesor en Ivrea).
- Colaborador: Hernando Barragán (Colombia), creador de la placa Wiring (2003), el lenguaje de programación y la plataforma de desarrollo.
- Objetivo: desarrollo económico y de código abierto (Hw y Sw). Utilizaban μC BASIC Stamp (U$S 100).
- A partir de 2012 se agregan modelos con μC Cortex M3 (ARM 32bits).
- Placas producidas comercialmente: 300K en 2011, y 700K en 2013.

Introducción a Arduino
Algunos detalles

- Plataforma de desarrollo de productos electrónicos (público no experto: artistas, entusiastas, etc.).
- Es hardware libre (diseño abierto que puede ser re-utilizado, modificado, etc.).
- Dispone de un IDE (Integrated Development Environment) de programación.
- Licencia GPL para el entorno de programación. LGPL para el código fuente de gestión y control del μC. Creative Commons Attribution Share-Alike para el hardware.
- Placas de expansión o shields.

Introducción a Arduino
Algunos detalles

- Plataforma de desarrollo de productos electrónicos (público no experto: artistas, entusiastas, etc.).
- Es hardware libre (diseño abierto que puede ser re-utilizado, modificado, etc.).
- Dispone de un IDE (Integrated Development Environment) de programación.
- Licencia GPL para el entorno de programación. LGPL para el código fuente de gestión y control del μC. Creative Commons Attribution Share-Alike para el hardware.
- Placas de expansión o shields.

Breve descripción

- Placa con microcontrolador marca Atmel
- Con componentes de soporte: regulador de tensión, conversión USB-serie, conectores, etc.
- Dispone de 14 pines configurables como entrada o salida.
- Dispone también de entrada y salida analógica (PWM).
- Dispone de 6 pines de entrada analógica (ADC de 10 bits).

Introducción a Arduino
¿Por qué elegir Arduino?

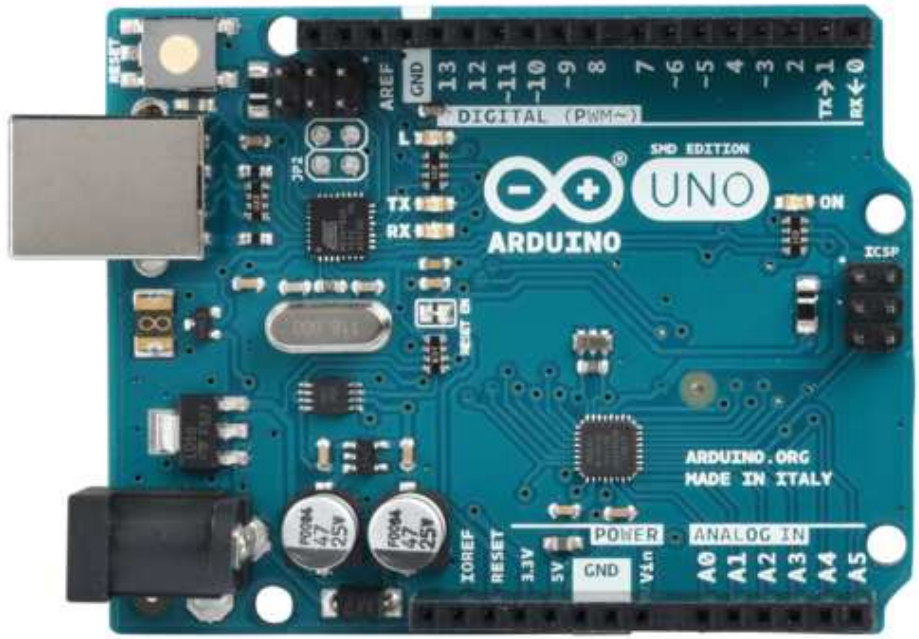
- Es libre y extensible.
- Tiene una gran comunidad.
- Entorno de programación multi-plataforma.
- Entorno y lenguaje de programación simple, claro y bien documentado.
- Las placas son económicas.
- Las placas son reutilizables y versátiles.

Introducción a Arduino

Arduino UNO R3

Introducción a Arduino

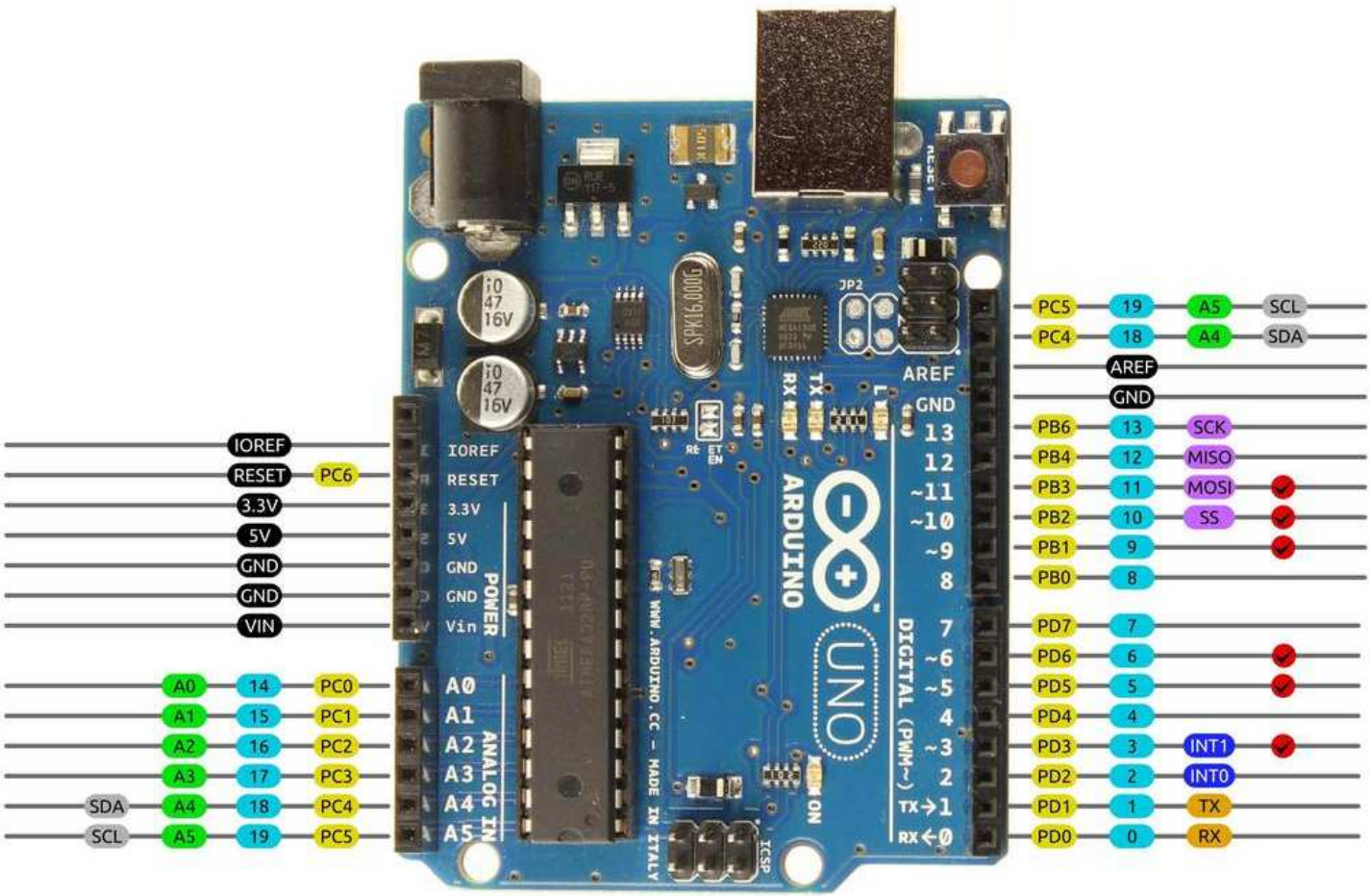
Arduino UNO R3



Microcontrolador (μC)	Atmega328 (Atmel AVR)
Memoria flash	32 KB (0.5 KB bootloader)
SRAM	2 KB
EEPROM	1 KB
Arquitectura	Hardvard
Frecuencia de reloj	16 MHz
Voltaje de operación	5V
Pines de E/S (I/O)	14 (6 PWM)
Pines de entrada analógica	6
Corriente por pin	40mA
Corriente por pin 3.3V	50mA

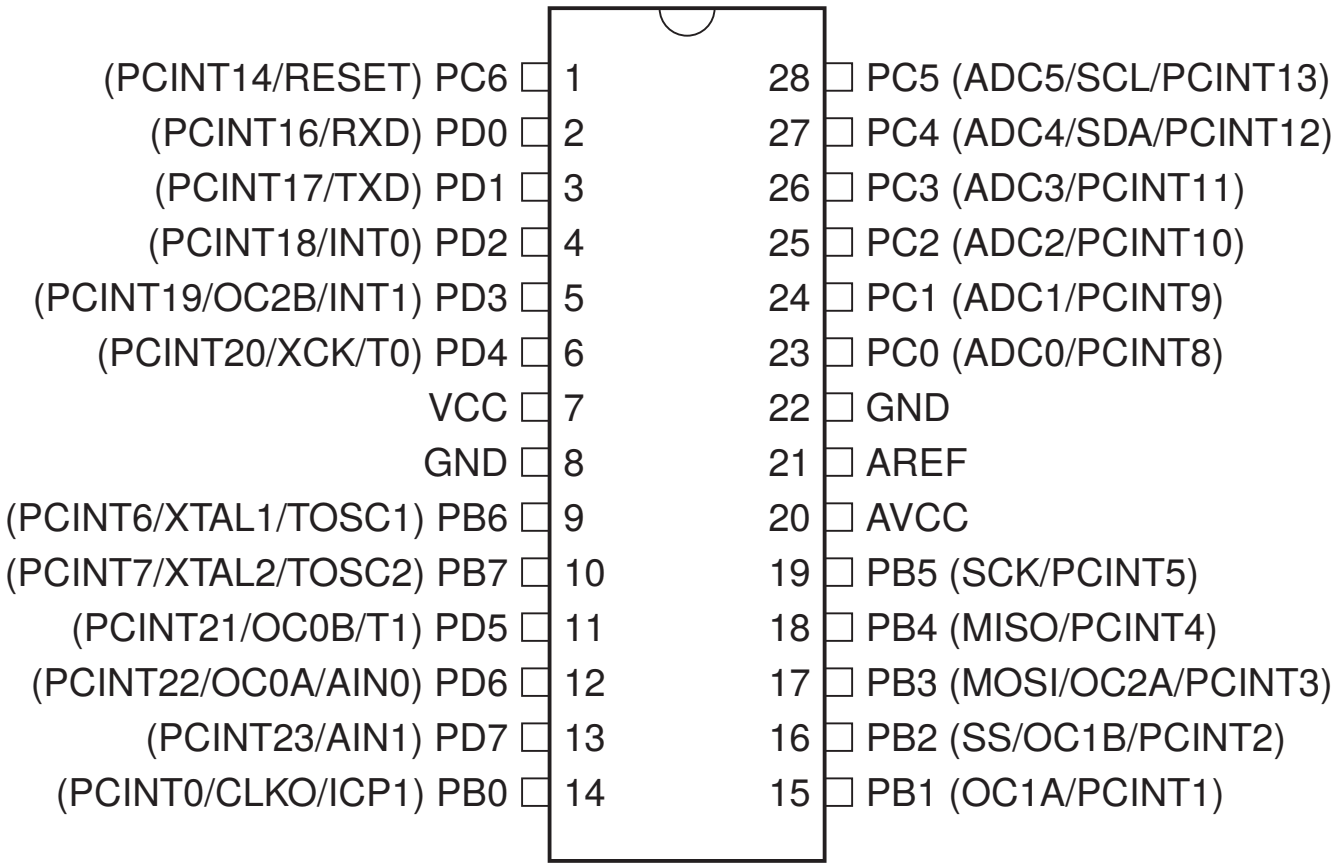
Introducción a Arduino

Arduino UNO R3 – Entrada/salida especiales y periféricos



Introducción a Arduino

Arduino UNO R3 – el ATmega328P

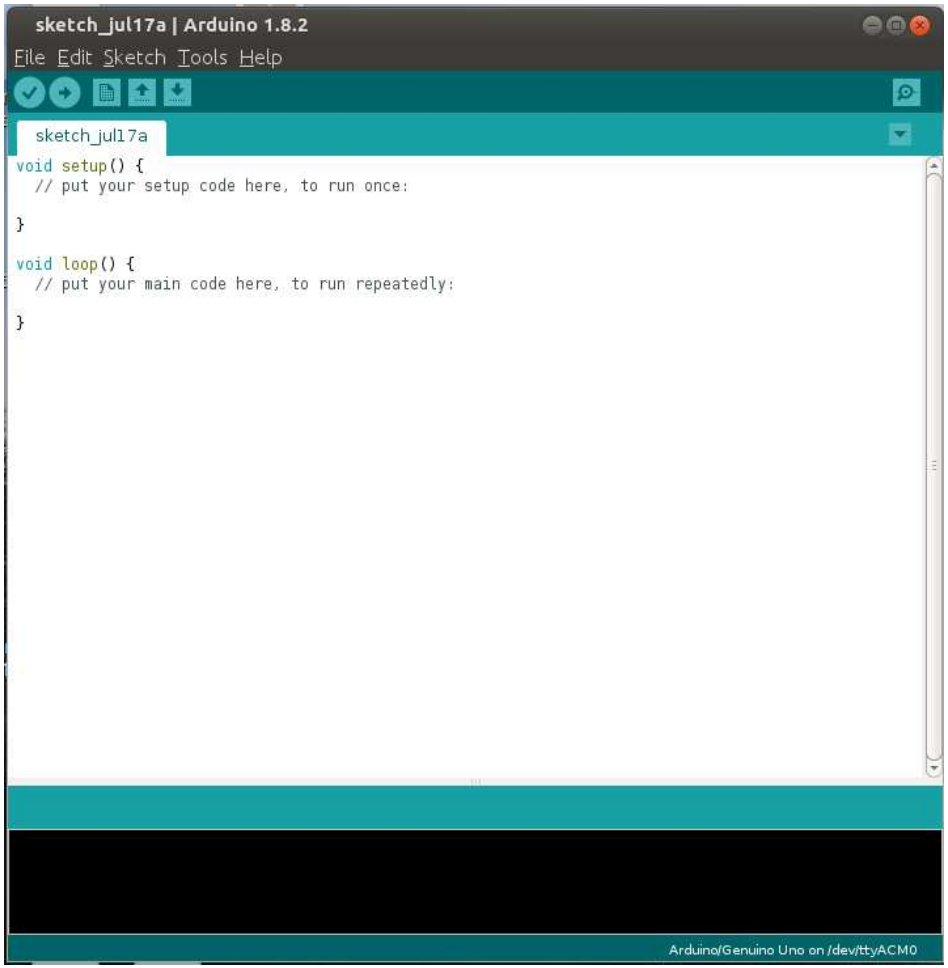


Introducción a Arduino
Programando Arduino – IDE

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.

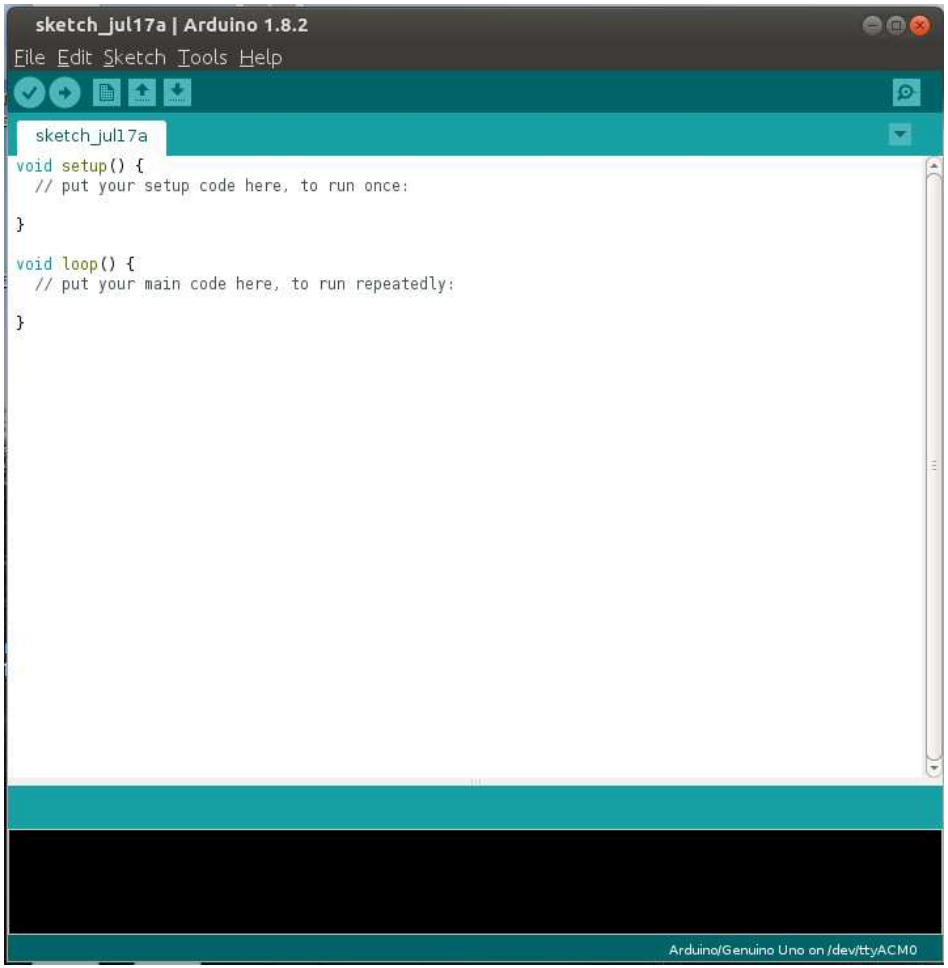
Introducción a Arduino
Programando Arduino – IDE

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.



Introducción a Arduino
Programando Arduino – IDE

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.

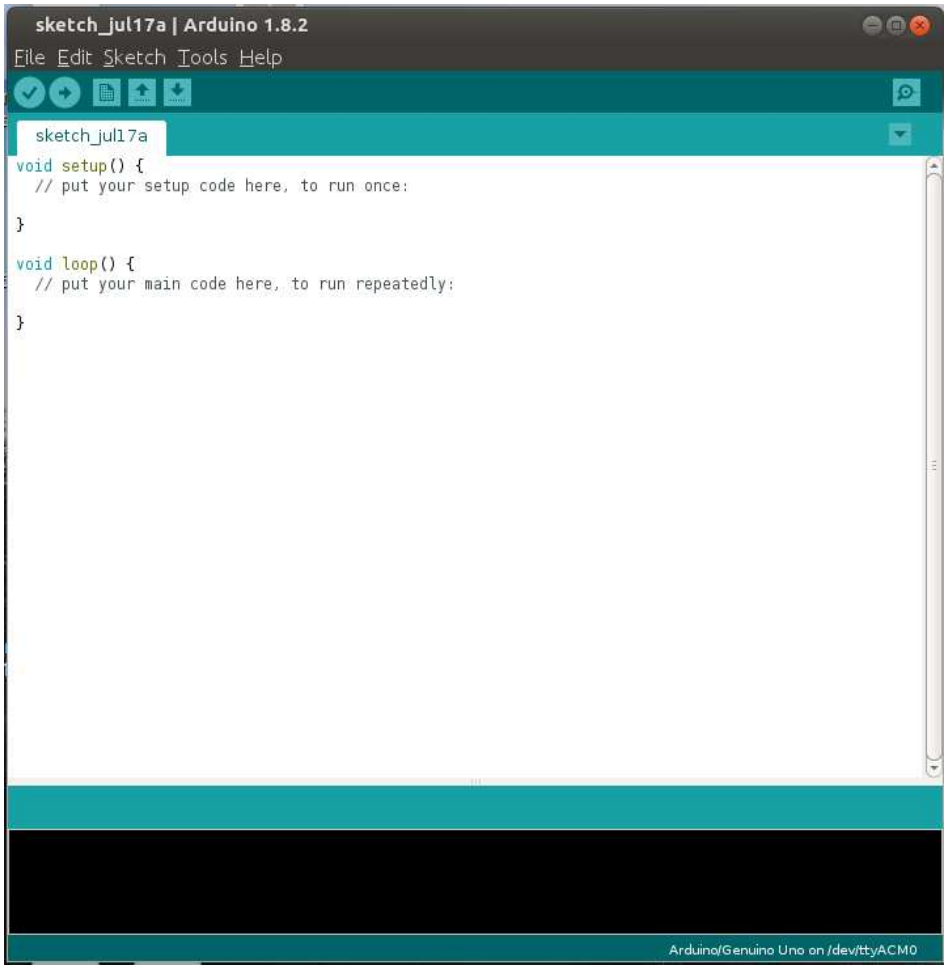


Se divide en 4 áreas:

1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

Introducción a Arduino
Programando Arduino – IDE

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.



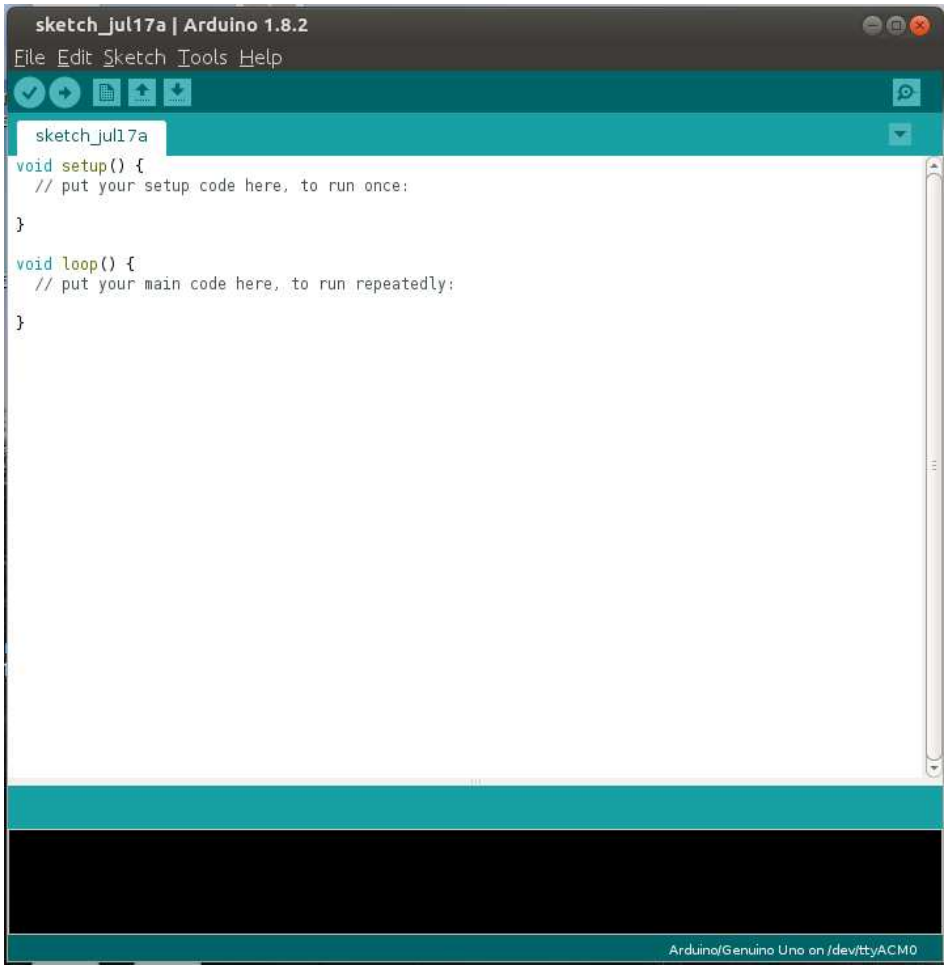
Se divide en 4 áreas:

1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

`sketch` Arduino – 2 bloques:

Introducción a Arduino
Programando Arduino – IDE

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.



Se divide en 4 áreas:

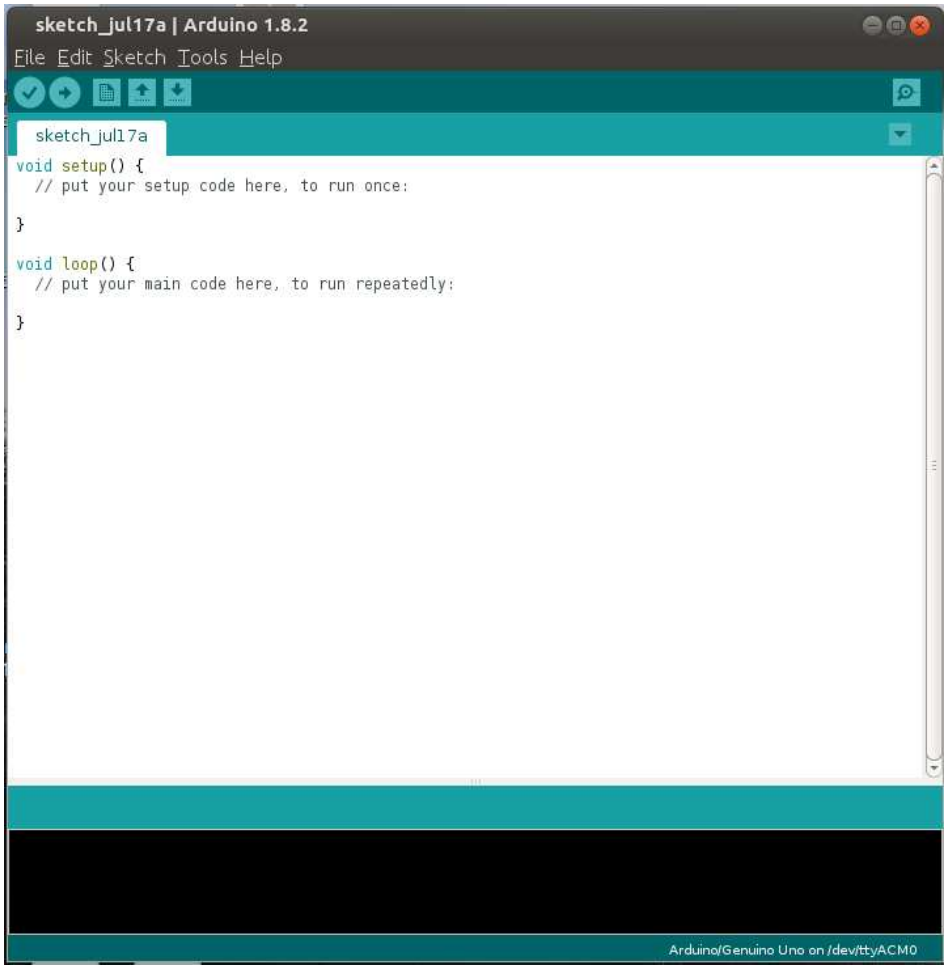
1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

`sketch` Arduino – 2 bloques:

- `setup()`: se ejecuta una única vez cuando se enciende o resetea la placa

Introducción a Arduino
Programando Arduino – IDE

IDE: Conjunto de herramientas de Sw que permite a los programadores desarrollar (básicamente escribir y probar) sus propios programas.



Se divide en 4 áreas:

1. Barra de menús
2. Barra de botones
3. Editor
4. Ventana de mensajes

`sketch` Arduino – 2 bloques:

- `setup()`: se ejecuta una única vez cuando se enciende o resetea la placa
- `loop()`: se ejecuta de forma constante (bucle)

Introducción a Arduino

Programando Arduino – ejemplo: blink

```
1  // the setup function runs once when you press reset or
2  // power the board
3  void setup() {
4    // initialize digital pin LED_BUILTIN as an output.
5    pinMode(LED_BUILTIN, OUTPUT);
6  }
7
8  // the loop function runs over and over again forever
9  void loop() {
10   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on
11   delay(1000);                       // wait for a second
12   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off
13   delay(1000);                       // wait for a second
14 }
```

Introducción a Arduino

Programando Arduino – ejemplo: blink

```
1  #define GPIO_LED 8  // LED_BUILTIN = 13
2
3  // the setup function runs once when you press reset or
4  // power the board
5  void setup() {
6    // initialize digital pin GPIO_LED as an output.
7    pinMode(GPIO_LED, OUTPUT);
8  }
9
10 // the loop function runs over and over again forever
11 void loop() {
12   digitalWrite(GPIO_LED, HIGH);       // turn the LED on
13   delay(1000);                        // wait for a second
14   digitalWrite(GPIO_LED, LOW);        // turn the LED off
15   delay(1000);                        // wait for a second
16 }
```

Introducción a Arduino

Programando Arduino – ejemplo: puerto serie

```
1  #define MESSAGE "Hello"
2
3  // the setup routine runs once when you press reset:
4  void setup() {
5    // initialize serial communication at 9600 bits per second:
6    Serial.begin(9600);
7  }
8
9  // the loop routine runs over and over again forever:
10 void loop() {
11   // print out the state of the button:
12   Serial.println(MESSAGE);
13   delay(500);      // delay in between reads for stability
14 }
```

Ver más ejemplos incluidos en el IDE Arduino

Introducción a Arduino
Funciones y bibliotecas

Entrada y salida digital

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

Introducción a Arduino
Funciones y bibliotecas

Entrada y salida digital

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

Entrada y salida analógica

- `analogRead()`
- `analogWrite()`
- `analogReference()`

Introducción a Arduino
Funciones y bibliotecas

Entrada y salida digital

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

Entrada y salida analógica

- `analogRead()`
- `analogWrite()`
- `analogReference()`

Temporizador

- `delay()`
- `delayMicroseconds()`
- `millis()`
- `micros()`

Introducción a Arduino
Funciones y bibliotecas

Entrada y salida digital

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

Entrada y salida analógica

- `analogRead()`
- `analogWrite()`
- `analogReference()`

Temporizador

- `delay()`
- `delayMicroseconds()`
- `millis()`
- `micros()`

Algunas bibliotecas

- LiquidCrystal
- EEPROM
- SD
- Ethernet
- Firmata
- SPI
- Wire
- SoftwareSerial
- Servo y Stepper

Introducción a Arduino
Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++

Introducción a Arduino

Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++
2. **Compilación:** (`avr-gcc`) convierte el código fuente a código objeto

Introducción a Arduino

Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++
2. **Compilación:** (`avr-gcc`) convierte el código fuente a código objeto
3. **Linkeo:** se combina el código objeto con la biblioteca estándar Arduino, y se genera un archivo `.hex`

Introducción a Arduino

Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++
2. **Compilación:** (`avr-gcc`) convierte el código fuente a código objeto
3. **Linkeo:** se combina el código objeto con la biblioteca estándar Arduino, y se genera un archivo `.hex`
4. **Grabación:** se graba el archivo al microcontrolador (bootloader)

Introducción a Arduino

Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++
2. **Compilación:** (`avr-gcc`) convierte el código fuente a código objeto
3. **Linkeo:** se combina el código objeto con la biblioteca estándar Arduino, y se genera un archivo `.hex`
4. **Grabación:** se graba el archivo al microcontrolador (bootloader)

Pasos:

- Se combinan todos los archivos `.ino` de la carpeta del sketch en un único archivo `.cpp`
- Se le agrega el archivo de cabecera `Arduino.h` que tiene las funciones core de Arduino
- Se generan los prototipos de funciones de los archivos `.ino`
- Se agregan las directivas `#line` para los mensajes de warning o error

(el pre-procesamiento se realiza solo a los archivos de extensión `.ino`)

(Ver `Arduino.h`, y `main.cpp`)

Introducción a Arduino

Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++
2. **Compilación:** (`avr-gcc`) convierte el código fuente a código objeto
3. **Linkeo:** se combina el código objeto con la biblioteca estándar Arduino, y se genera un archivo `.hex`
4. **Grabación:** se graba el archivo al microcontrolador (bootloader)

Se compila usando `avr-gcc` y `avr-g++` de acuerdo a las variables del archivo `boards.txt` y de la placa usada. Se incluyen los siguientes paths:

- Variante de la placa (variable `build.variant`)
- Directorio core (variable `build.core`)
- Directorio include de avr (`hardware/tools/avr/include/`)
- sub-directorios de `libraries/` del IDE, placa/hardware, y sketchbook

(el sketch se construye en un directorio temporal, `/tmp` en Linux)

Introducción a Arduino

Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++
2. **Compilación:** (`avr-gcc`) convierte el código fuente a código objeto
3. **Linkeo:** se combina el código objeto con la biblioteca estándar Arduino, y se genera un archivo `.hex`
4. **Grabación:** se graba el archivo al microcontrolador (bootloader)

Configurar modo “verbose” del IDE Arduino y analizar salida de terminal :)

Introducción a Arduino

Proceso de construcción

1. **Pre-procesamiento:** convertir el sketch en un programa C++
2. **Compilación:** (`avr-gcc`) convierte el código fuente a código objeto
3. **Linkeo:** se combina el código objeto con la biblioteca estándar Arduino, y se genera un archivo `.hex`
4. **Grabación:** se graba el archivo al microcontrolador (bootloader)

Configurar modo “verbose” del IDE Arduino y analizar salida de terminal :)

Introducción a Arduino
Programación en C

- Compilador: `avr-gcc`
- Biblioteca: `avr-libc`
- Herramienta de grabación: `avrdude`

Introducción a Arduino
Programación en C

- Compilador: `avr-gcc`
- Biblioteca: `avr-libc`
- Herramienta de grabación: `avrdude`

(Ver ejemplo de Francesco Balducci)

Introducción a Arduino
Programación en C

- Compilador: `avr-gcc`
- Biblioteca: `avr-libc`
- Herramienta de grabación: `avrdude`

(Ver ejemplo de Francesco Balducci)

Construcción

- `$avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o blink.o blink.c`
- `$avr-gcc -mmcu=atmega328p blink.o -o blink.elf`
- `$avr-objcopy -O ihex -R .eeprom blink.elf blink.hex`
- `$avrdude -F -V -c arduino -p ATMEGA328P -P /dev/ttyACM0 -b 115200 -U flash:w:blink.hex`

Introducción a Arduino
Programación en C

- Compilador: `avr-gcc`
- Biblioteca: `avr-libc`
- Herramienta de grabación: `avrdude`

(Ver ejemplo de Francesco Balducci)

Construcción

- `$avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o blink.o blink.c`
- `$avr-gcc -mmcu=atmega328p blink.o -o blink.elf`
- `$avr-objcopy -O ihex -R .eeprom blink.elf blink.hex`
- `$avrdude -F -V -c arduino -p ATMEGA328P -P /dev/ttyACM0 -b 115200 -U flash:w:blink.hex`

(Ver archivos `iom328p.h` y `sfr_defs.h`)

Introducción a Arduino
Referencias

- Arduino. Curso práctico de formación. Óscar Torrente Artero. Alfaomega 2013 (libro)
- Curso de supervivencia con Arduino. Juan Gregorio Regalado Pacheco (creative commons)
- Build Process: `https://github.com/arduino/Arduino/wiki/Build-Process`
- Programming Arduino Uno in pure C. Balau's technical blog on open hardware, free software and security