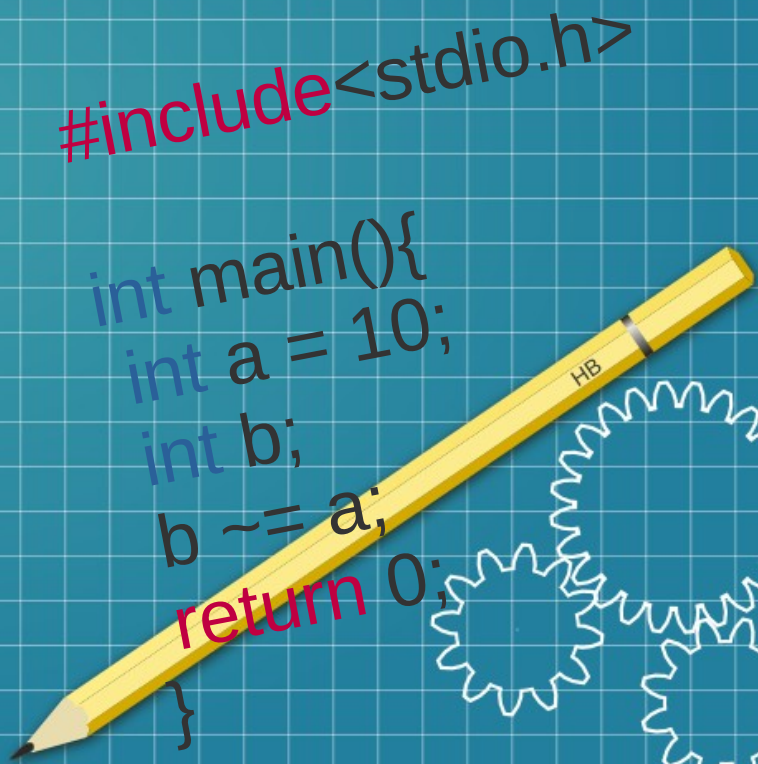
¿Qué es Arduino?

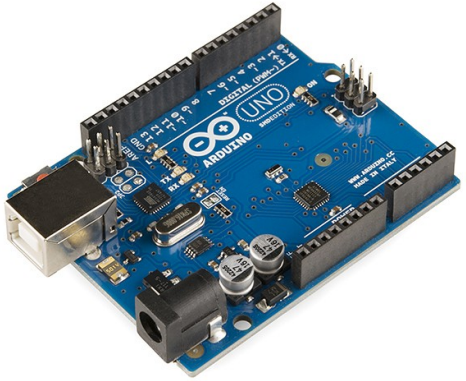
- Es una plataforma de prototipado electrónico de código abierto.
- Está diseñada para facilitar la creación de proyectos.
- Está constituido por hardware (placas) y software (entorno de desarrollo).



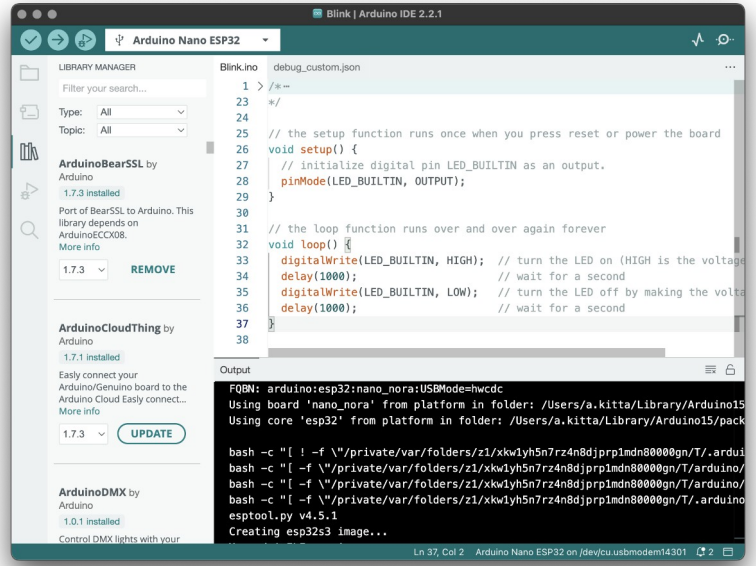
Palomeque Nestor Levi

Componentes Principales

Placas



IDE



Lenguaje de programación

Está Basado en C/C++

También:

- Python
- JavaScript
- Blockly
- Scratch

Puertos

Puertos Digitales:

Los pines digitales permiten conectar botones, sensores digitales o cualquier dispositivo que funcione con señales binarias (0 o 1).

Estos pines se numeran desde D0 hasta D13 en la placa Arduino Uno.

Puedes usarlos para encender o apagar luces, leer entradas de sensores o controlar relés.

Puertos PWM (Modulación por Ancho de Pulso):

Algunos pines digitales también son PWM. Estos permiten controlar la intensidad de una señal (como el brillo de un LED) variando el ancho de pulso.

Los pines D3, D5, D6, D9, D10 y D11 son PWM en el Arduino Uno.

Puertos Analógicos:

Los pines A0 hasta A5 son entradas analógicas.

Puedes conectar sensores que proporcionan valores continuos (como temperatura o luz) y leer esos valores en el Arduino.

Puertos de Comunicación:

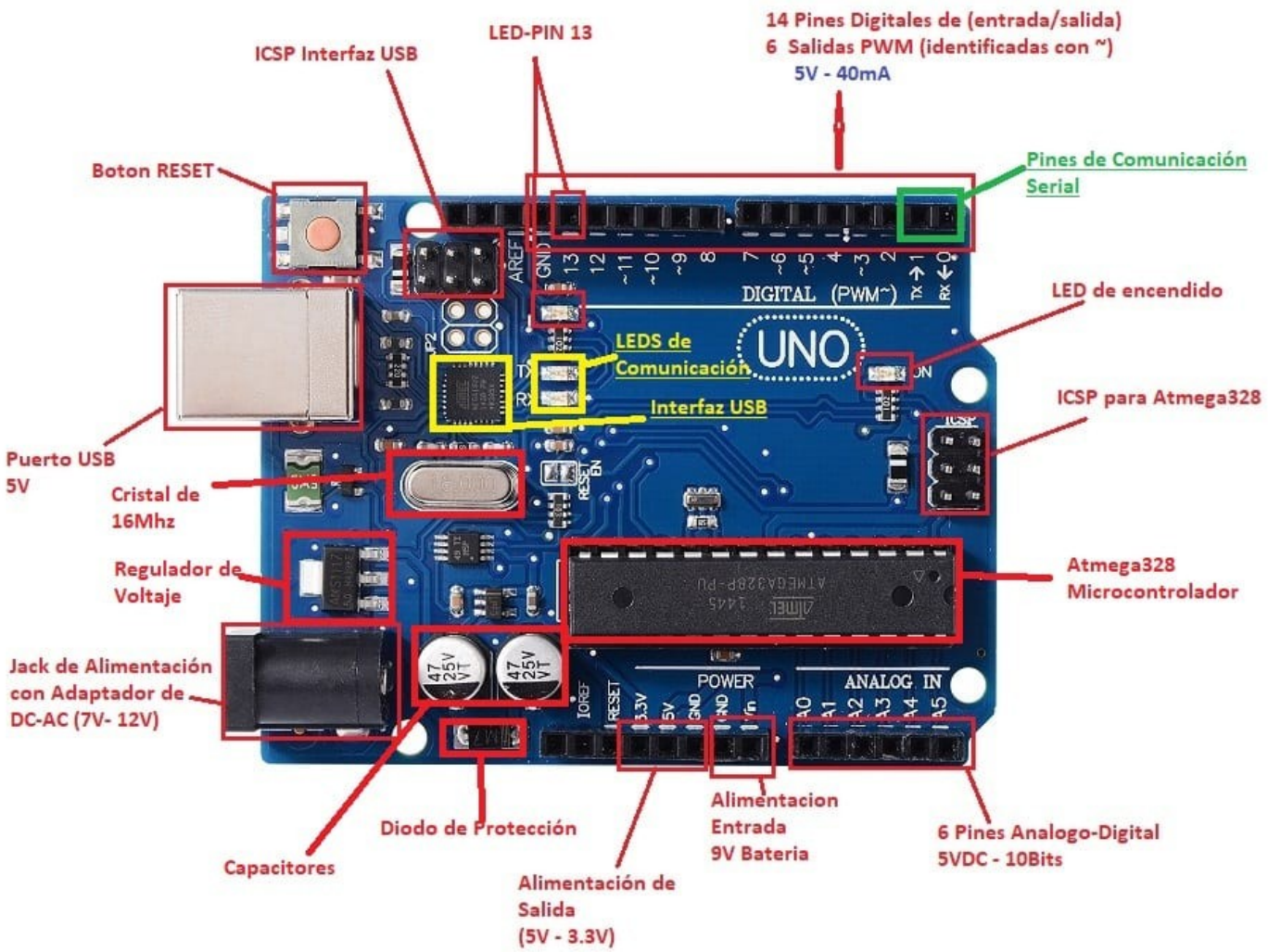
El Arduino Uno tiene un puerto USB para programarlo y comunicarse con la computadora.

Además, hay pines TX y RX (D0 y D1) que se utilizan para la comunicación serie con otros dispositivos.

Puertos de Alimentación:

5V y 3.3V proporcionan energía a otros componentes conectados al Arduino.

GND es el pin de tierra para completar los circuitos.


ICSP Interfaz USB
LED-PIN 13
14 Pines Digitales de (entrada/salida)
6 Salidas PWM (identificadas con ~)
5V - 40mA
Boton RESET
Pines de Comunicación Serial
LED de encendido
LEDS de Comunicación
Interfaz USB
ICSP para Atmega328
Puerto USB 5V
Cristal de 16Mhz
Regulador de Voltaje
Atmega328 Microcontrolador
Jack de Alimentación con Adaptador de DC-AC (7V- 12V)
Capacitores
Diodo de Protección
Alimentación de Salida (5V - 3.3V)
Alimentacion Entrada 9V Bateria
6 Pines Analogo-Digital 5VDC - 10Bits
DIGITAL (PWM~)
UNO
POWER
ANALOG IN

Herramientas de simulación

Algunas de los simuladores que permiten emplear Arduino son:

1) **Tinkercad Circuits: Tinkercad es gratuito para uso personal, educativo y sin fines de lucro.**
2) Fritzing: Fritzing es de código abierto y completamente gratuito para su uso.
3) SimulIDE: SimulIDE es otro software de código abierto y gratuito que incluye soporte para Arduino.
4) Proteus: es de pago.

Código inicial

Cuando se programa Arduino, el código se divide típicamente en dos secciones principales: setup() y loop().

Función setup():

La función setup() se ejecuta una vez al inicio del programa y se utiliza para realizar configuraciones iniciales y preparar el entorno para la ejecución del programa principal.

Función loop():

La función loop() se ejecuta continuamente después de que la función setup() haya terminado de ejecutarse. Es el corazón del programa y contiene el código principal que se ejecutará repetidamente en un bucle infinito.

```
// Función de configuración inicial
void setup() {
    // Código de inicialización aquí
}

// Función principal del programa
void loop() {
    // Código principal del programa aquí
}
```

```
int main() {
    // Configuración inicial
    setup();
    // Bucle principal
    while (1) {
        loop();
    }
    return 0;
}
```

Algunas funciones de arduino

- **pinMode(pin, mode):** se utiliza para configurar el modo de un pin digital en Arduino, es decir, si se debe usar como entrada o como salida. **Pin** es el número de pin que se desea configurar y **mode** puede ser INPUT o OUTPUT, indicando si se desea configurar el pin como entrada o salida, respectivamente.

  Ejemplo: pinMode(13, OUTPUT); configurará el pin 13 como salida.

- **digitalWrite(pin, value)**: se utiliza para establecer el estado de un pin digital en Arduino. **Pin** es el número de pin que se desea controlar y **value** puede ser HIGH o LOW, lo que indica si se desea establecer el pin en un estado alto (5V) o bajo (0V), respectivamente.

  Ejemplo: digitalWrite(13, HIGH); establecerá el pin 13 en un estado alto.

- **digitalRead(pin):** se utiliza para leer el estado de un pin digital en Arduino. **Pin** es el número de pin que se desea leer. Devuelve HIGH si el pin está en un estado alto (conectado a 5V) o LOW si está en un estado bajo (conectado a tierra).

  Ejemplo: int estado = digitalRead(2); leerá el estado del pin 2 y lo almacenará en la variable estado.

- **delay(milliseconds):** se utiliza para hacer pausas en la ejecución del programa durante un cierto período de tiempo. **Milliseconds** es la cantidad de tiempo que se desea pausar, en milisegundos. Durante el tiempo especificado, Arduino no ejecutará ninguna otra instrucción.

  Ejemplo: delay(1000); pausará la ejecución del programa durante 1 segundo (1000 milisegundos).

Algunas funciones del puerto serieal

- **Serial.begin(baudrate):** se utiliza para inicializar la comunicación serie en Arduino (inicializar el puerto). **Baudrate** especifica la velocidad de transmisión en baudios, es decir, la velocidad de transferencia de datos en bits por segundo.

  Ejemplo: Serial.begin(9600); inicializa la comunicación serie con una velocidad de 9600 baudios.

  Serial.read():

- **Serial.available():** se utiliza para comprobar si hay datos disponibles para ser leídos del puerto serie. Devuelve el número de bytes disponibles para lectura en el búfer de entrada del puerto serie.

  Ejemplo: int bytesDisponibles = Serial.available(); almacenará en la variable bytesDisponibles la cantidad de bytes disponibles para ser leídos.

- **Serial.read():** lee un byte de datos del puerto serie. Devuelve el byte leído como un valor entero en el rango de 0 a 255 o -1 si no hay datos disponibles para leer.

  Ejemplo: int byteLeido = Serial.read();
  Serial.readBytes(buffer, length):

- **Serial.readBytes((buffer, length):** lee una serie de bytes de datos del puerto serie y los almacena en un búfer. Buffer es un puntero al búfer donde se almacenarán los datos leídos, length es el número de bytes que se desean leer. Devuelve el número de bytes leídos o -1 si no se leyó ningún byte antes de que se cumpla el tiempo de espera.

  Ejemplo: char buffer[20];
  int bytesLeidos = Serial.readBytes(buffer, 20);

- **Serial.parseInt():** es una función de la clase String en Arduino que convierte una cadena de caracteres en un valor entero. Devuelve el valor entero correspondiente a los dígitos en la cadena hasta que encuentra un carácter que no es un dígito.

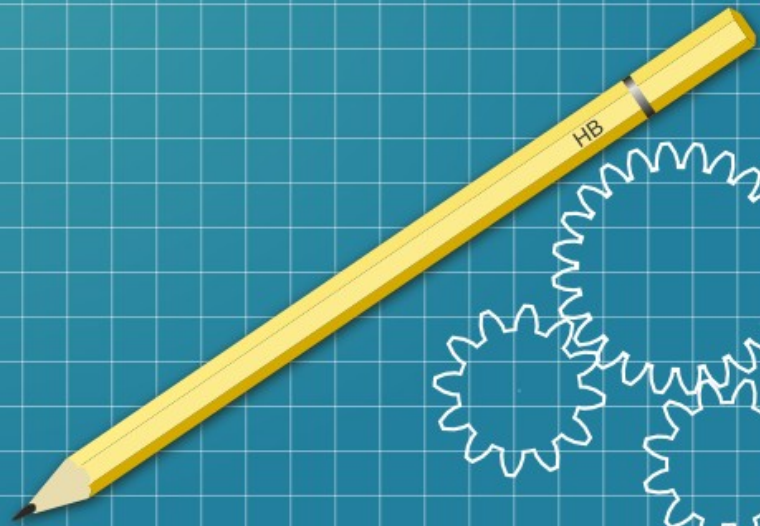
  Ejemplo: int numero = datosRecibidos.parseInt();

- **Serial.print(data), Serial.println(data)**: se utilizan para enviar datos formateados a través del puerto serie. **Data** es el valor que se desea enviar, que puede ser de diferentes tipos de datos (entero, flotante, cadena, etc.)

Ejercicio

- Realizar un programa que haga parpadear un led en el pin 13: 1 segundo prendido y 2 segundos apagados.
- Modificar el programa anterior para agregar un segundo led en el pin 12, el cual enciende por 1.55 segundos y está apagado por 2,75 segundo.
- Modificar el programa anterior para enviar por el puerto serie el estado del pin 13.
- Modificar el programa anterior para recibir por el puerto serie la orden que habilita el encendido del pin 13 y el pin 12, por separado.





HB