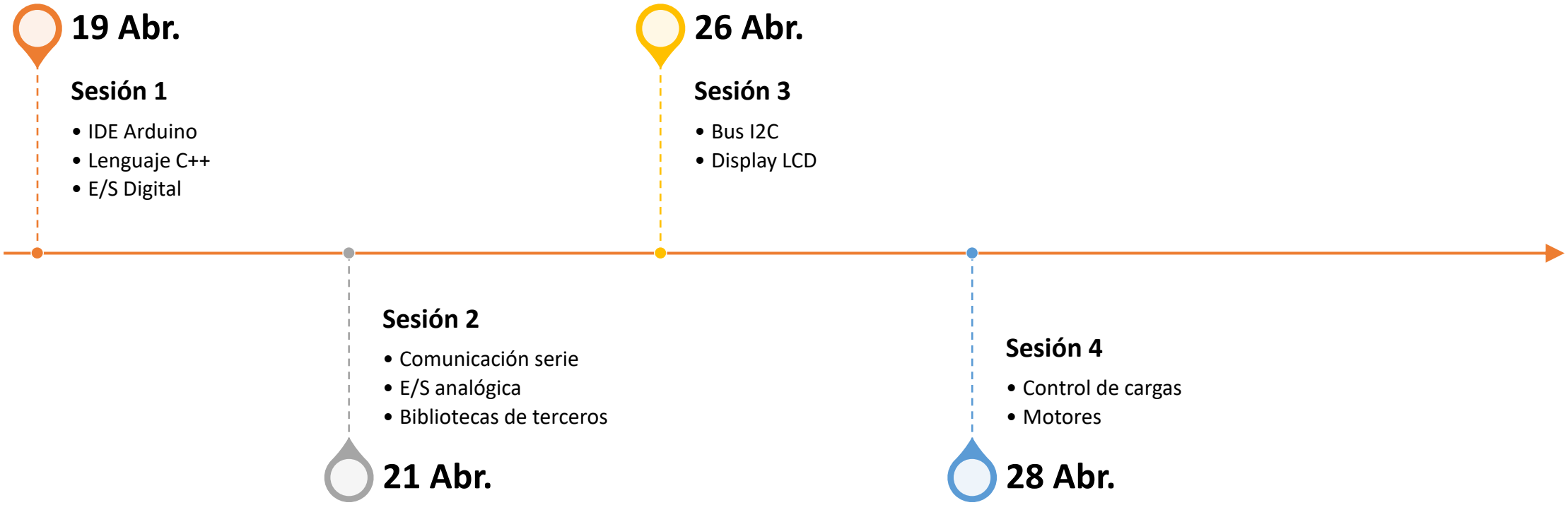
ARDUINO

Programación con C++, nivel inicial

CFIE Palencia - Abril 2022

JESÚS PIZARRO PELÁEZ

Contenidos


19 Abr.
Sesión 1
• IDE Arduino
• Lenguaje C++
• E/S Digital
21 Abr.
Sesión 2
• Comunicación serie
• E/S analógica
• Bibliotecas de terceros
26 Abr.
Sesión 3
• Bus I2C
• Display LCD
28 Abr.
Sesión 4
• Control de cargas
• Motores

Sesión I

Placas Arduino

IDE Arduino

Lenguajes de programación

Lenguaje C++

E/S Digital

Salida digital

Retardos

Entrada digital. Pulsadores

Plataforma Arduino

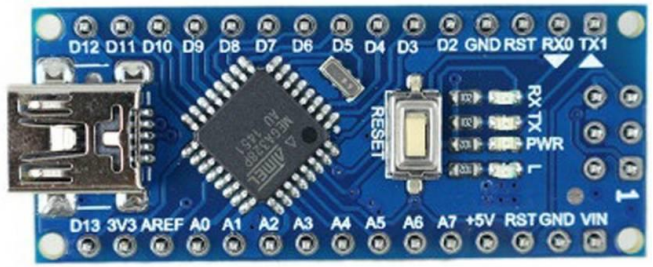
Plataforma de código abierto hardware y software

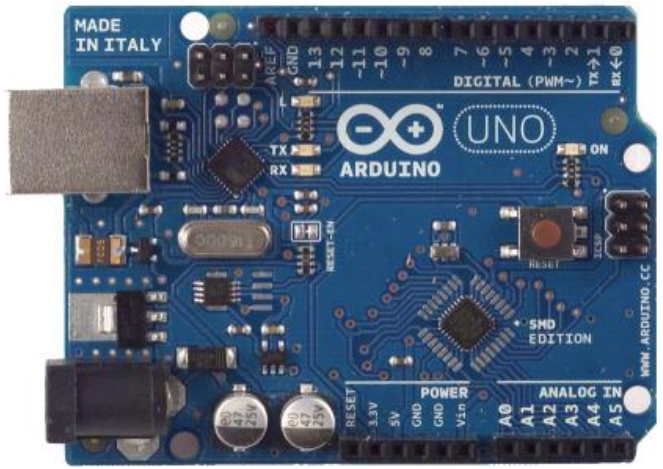
IDE basado en C++

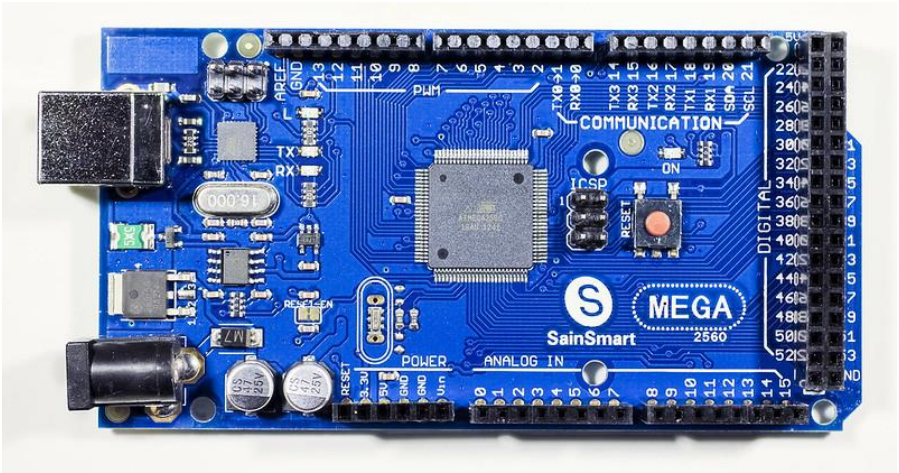
Entornos de programación en bloques

Placas con pines accesibles y etiquetados para montaje rápido y sencillo

Multitud de librerías y desarrollos de terceros

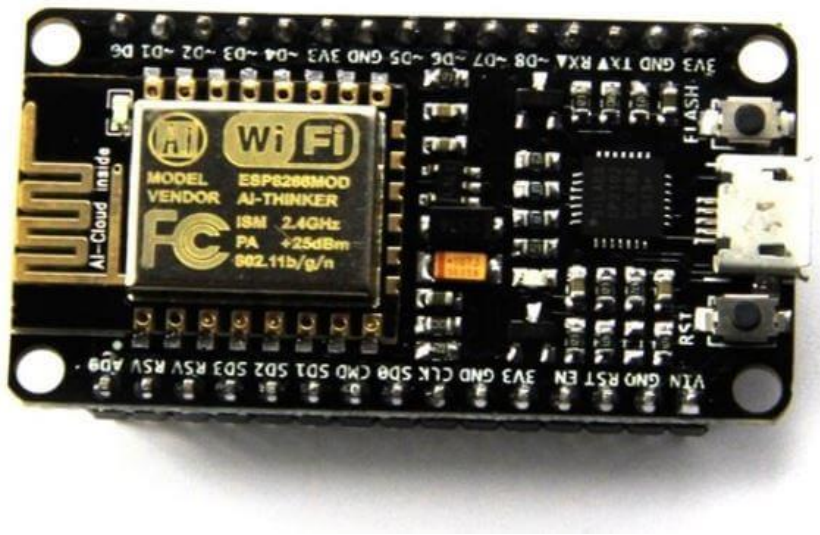


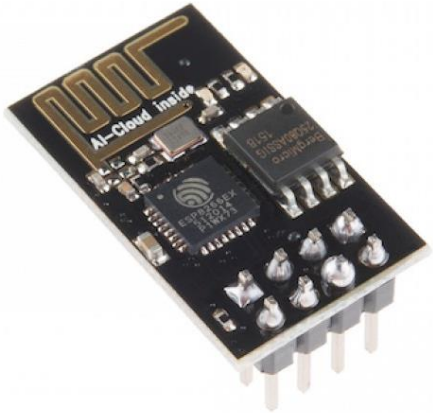




Placas Arduino

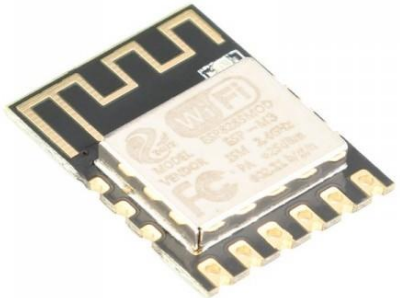
- Arduino UNO
- Arduino MEGA
- Arduino Nano
- Etc.



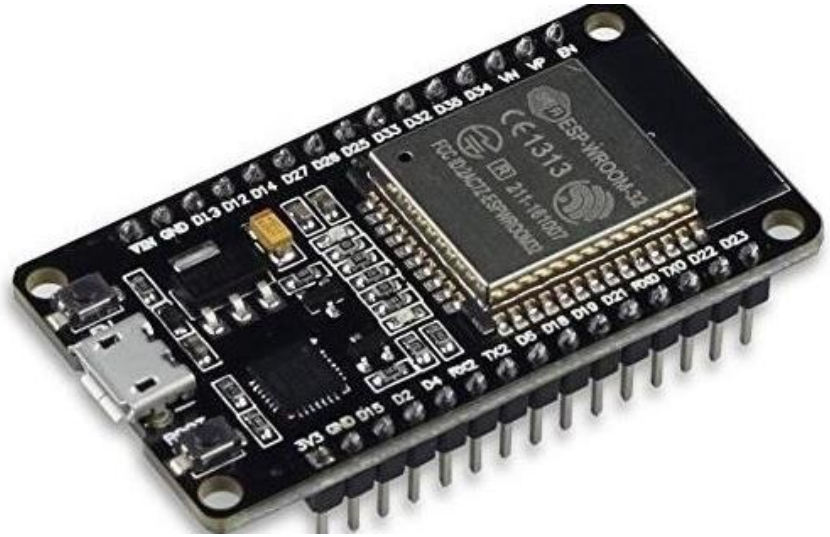


Otras placas ESP8266

- SoC de la compañía Expressif Systems
- Microcontrolador, TCP/IP y Wifi en un chip
- Considerablemente más potente que Arduino
- Compilador cruzado
- Tensión de trabajo 3.3V



Otras placas
ESP32



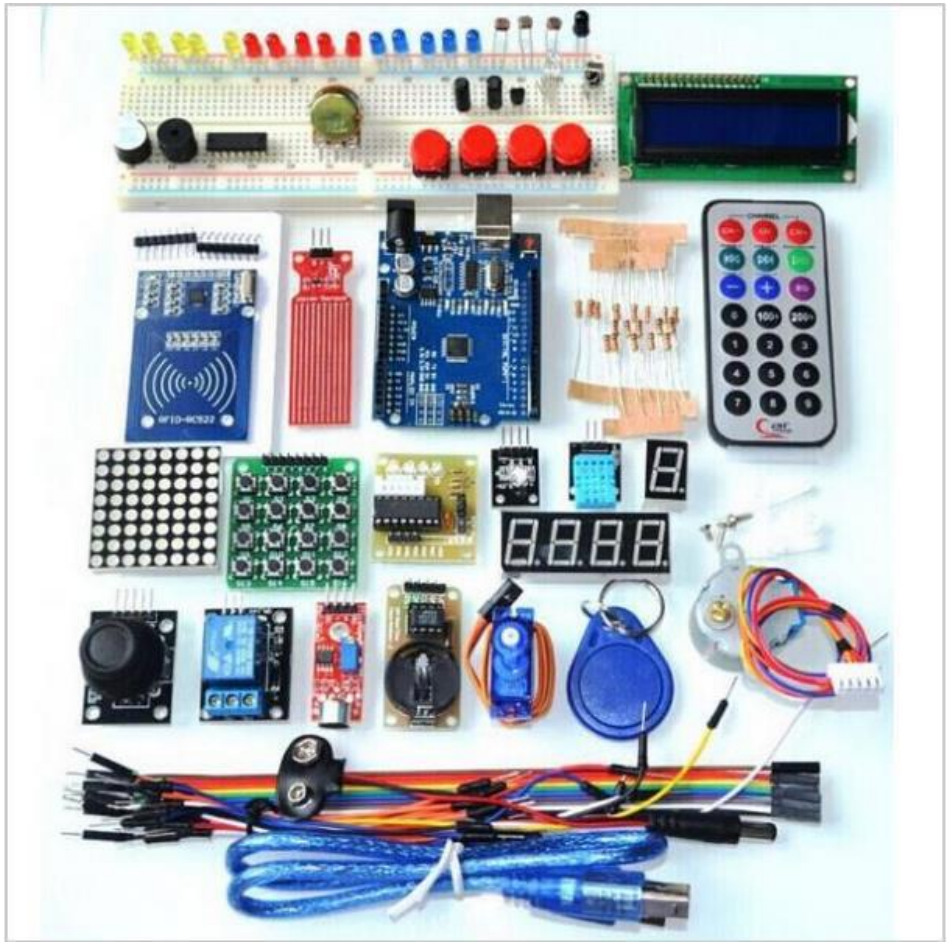
- Gran potencia
- Orientado a conectividad total
- Herramientas de programación en desarrollo

Comparativa

Especificaciones	ESP32	ESP8266	Arduino UNO
Número de núcleos	2	1	1
Arquitectura	32 bits	32 bits	8 bits
Frecuencia de la CPU	240MHz	80MHz	16MHz
WIFI	SI	SI	NO
Bluetooth	SI	NO	NO
RAM	512kB	160kB	2kB
Flash	16MB	16MB	32kB
Pines GPIO	36	17	14
Entradas analógicas	18	1	6
Salidas analógicas	2	0	0

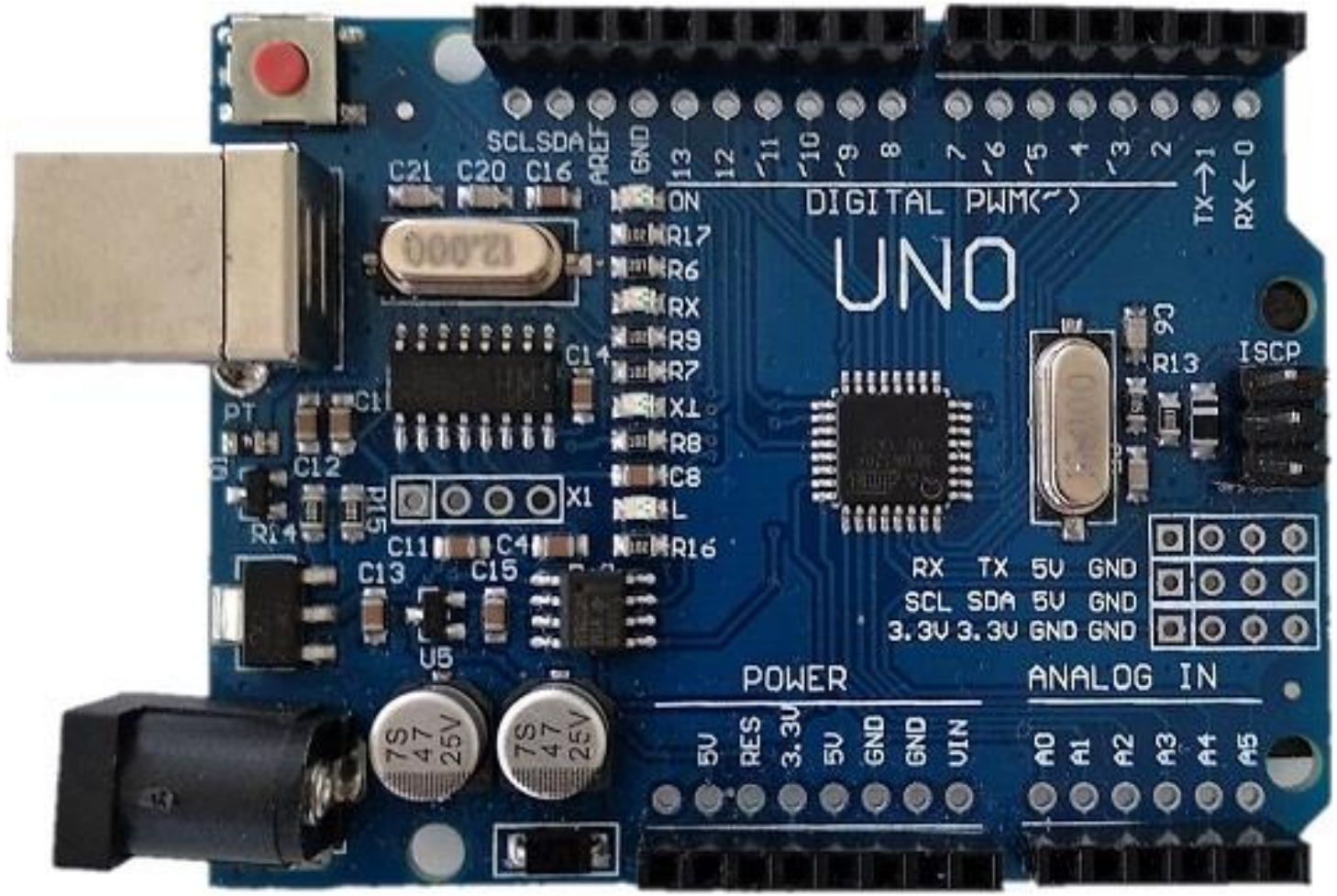
Arduino starter kit

- Placa Arduino UNO
- Multitud de componentes
- Ideal para empezar



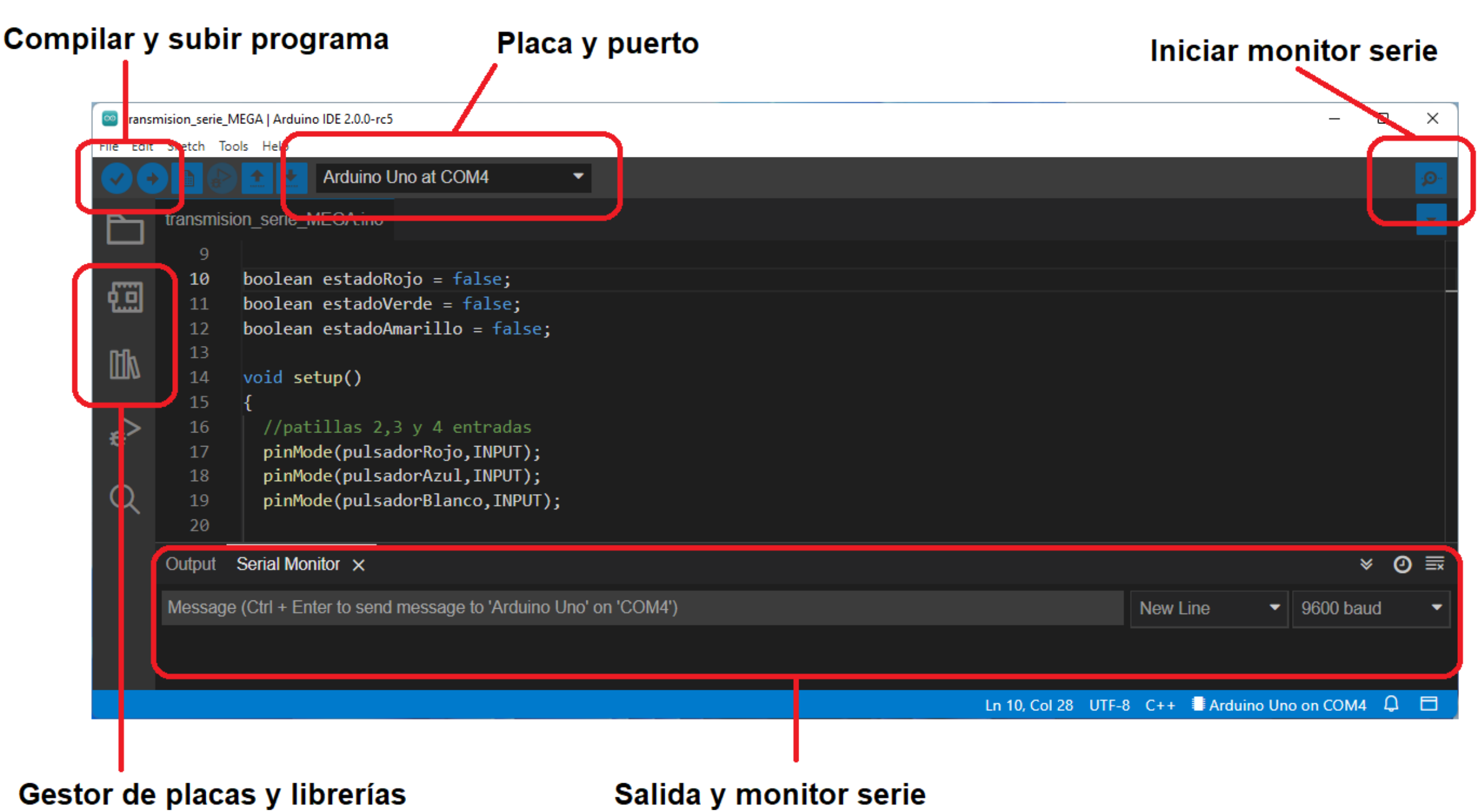
Arduino UNO

- Procesador ATmega328P
- 20 pines E/S digital
- 6 Entradas analógicas
- 6 salidas PWM
- Pines de tensión y tierra
- Conexión USB



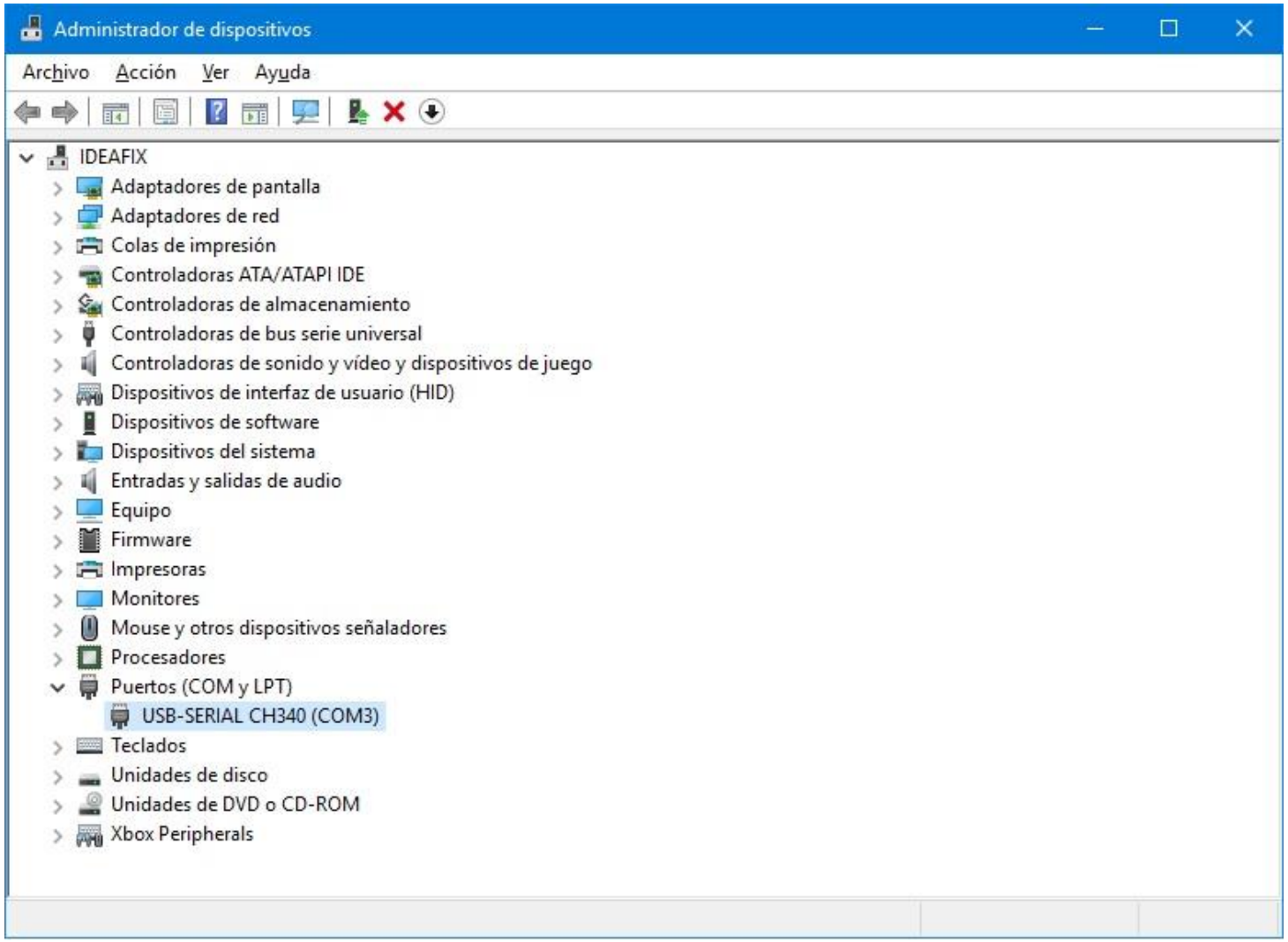
IDE Arduino

- Open source
- Multiplataforma
- Versión portable
- https://www.arduino.cc/en/Main/Software


Compilar y subir programa
Placa y puerto
Iniciar monitor serie
ransmision_serie_MEGA | Arduino IDE 2.0.0-rc5
File Edit Sketch Tools Help
Arduino Uno at COM4
transmision_serie_MEGA.ino
9
10 boolean estadoRojo = false;
11 boolean estadoVerde = false;
12 boolean estadoAmarillo = false;
13
14 void setup()
15 {
16 //patillas 2,3 y 4 entradas
17 pinMode(pulsadorRojo,INPUT);
18 pinMode(pulsadorAzul,INPUT);
19 pinMode(pulsadorBlanco,INPUT);
20
Output Serial Monitor
Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM4')
New Line
9600 baud
Ln 10, Col 28 UTF-8 C++ Arduino Uno on COM4
Gestor de placas y librerías
Salida y monitor serie

Conexión al PC

- Interfaz USB
- Puerto COM virtual
- Placas originales
- Placas compatibles
- Descargar drivers



Lenguajes de programación

Lenguaje C++

- Evolución de C
- Lenguaje de propósito general
- Muy utilizado en ingeniería
- Programación orientada a objetos
- Base de otros lenguajes (Java, php, etc.)

Sintaxis básica

Bloques	Instrucciones	Directiva `#define`	Comentarios
• Secuencia de instrucciones que se ejecutan secuencialmente • Entre llaves **{ }** • setup() y loop()	• Tareas que debe realizar el programa • Finalizan con **;** • Llamadas a funciones	• Al comienzo del programa • Para definir constantes • #define LedRojo 3	• Facilitan la lectura del código • Multilínea: /* comentario */ • Una sola línea (siempre al final): // comentario

Bloques setup() y loop()

`setup()`

- Se ejecuta una sola vez al inicio
- Inicialización de variable y sistemas

`loop()`

- Se ejecuta indefinidamente
- Al finalizar vuelve al comienzo

Llamada a función

```
nombreFuncion(param1,param2,...,paramN);
```

- Instrucción más empleada
- Puede llevar parámetros o no
- Siempre con paréntesis
- Pueden devolver un valor

Ejemplos

```
tiempo = millis();
pinMode(2, OUTPUT);
estadoEntrada=digitalRead(3);
```

Variables

- Almacenan información en la memoria del sistema
- Se puede modificar su valor en tiempo de ejecución
- Deben declararse antes de usarlas
- Un único tipo de dato
- Globales y locales

Tipos de datos

y tamaño en bits

Enteros

- char – 8
- int – 16
- long – 32
- [unsigned]

Reales

- float – 32
- double – 64

bool

- Solo 0 o 1
- Variables lógicas

String

- Cadenas de texto
- Objeto complejo

Declaración de variables

```
int temperatura;
bool estadoTermostato = true;
float x1, x2;
int a = 2*b + 20;
char c = 'a';
String mensaje = "Hola mundo";
```

Operadores

Aritméticos	(+ , - , * , / , %)
Comparación	(!= , == , <= , >= , < , >)
Lógicos	(! , && , ||)
Orientados a bit	(~ , & , | , ^ , << , >>)
Abreviados	(a++ , a-- , a+=2 , a*=10)

map(var, mín inicial, máx inicial, mín final, máx final);

Arrays y objetos

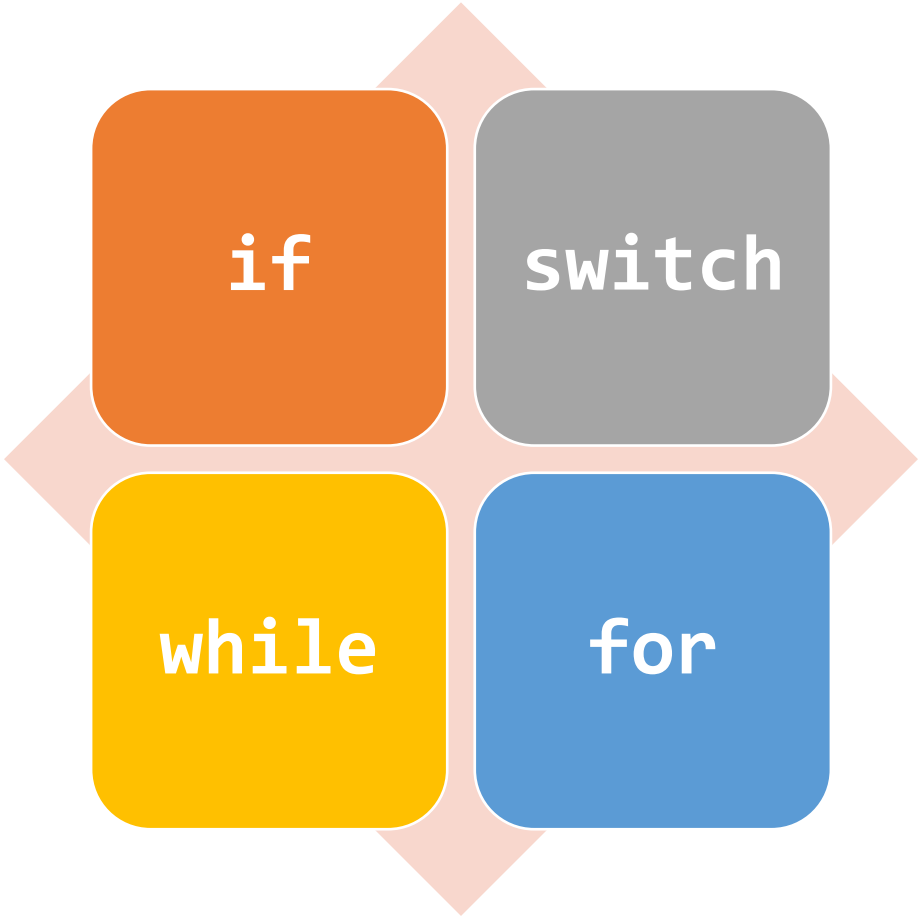
Array

- Varios datos del mismo tipo
- Multidimensionales
- Acceso indexado con []
- `char caracter[4] = {'h', 'o', 'l', 'a'};`

Objeto

- Tipo de datos complejo
- Con propiedades y métodos
- Facilita la programación
- `stepper.step(1024);`

Estructuras de control



Estructura condicional `if - else`

- Evalúa si se cumple una condición
- Si se cumple se ejecuta el bloque
- Si no se cumple se sale o…
- Se pueden realizar más comprobaciones

Condiciones lógicas

```
if(digitalRead(3)==LOW)     //verificar estado de una entrada
if(caracter == 'a')         //evaluar valor de una variable
if(temp<10 || temp>30)      //variable dentro de un intervalo
if(!alarmaActiva)           //evaluación de una variable lógica
if(temp<20 && calefaON)     //comprobar dos condiciones
```

Estructura `if else`

```
if(condicion1){
      instrucciones a ejecutar si se cumple condicion1
}
else if(condicion2){
      ejecutar si no se cumple condicion1 y si condicion2
}
else{
      ejecutar si no se cumple ninguna condicion
}
…
Continuación del flujo del programa
```

Estructura condicional `switch – case`

- Evalúa estados de una variable
- Código más claro que con if
- Solo para evaluar una misma variable

Estructura switch – case

```
switch (variable){
      case valor1:
            acciones a realizar si variable=valor1
            break;
      case valor2:
            acciones a realizar si variable=valor2
            break;
      …
      default:
            acciones a realizar si no toma ningún valor previo
}
```

Bucles `while` y `do while`

- El contenido se repite mientras se cumpla la condición
- While realiza la comprobación al principio
- Do while realiza la comprobación al final
- Estructuras intercambiables

Bucles `while` y `do while`

```
while(condicion){
     instrucciones a ejecutar
}


do{
     instrucciones a ejecutar
}while(condicion);
```

Bucle `for`

- El contenido se repite un determinado número de veces
- Se emplea una variable de control de bucle (vcb)
- Se define su valor inicial, final e incremento
- El valor de la variable puede usarse dentro

Bucle for

```
for(vcb=valorInicial;condicionPermanecia;incremento)
{
    instrucciones a ejecutar
}
```

```
for (int i=0;  i<10;  i++)  //10 rep, i = 0, 1 … 9
for (int i=10; i>0;   i--)  //10 rep, i = 10, 9 … 1
for (int i=0;  i<10;  i+=2) //5  rep, i = 0, 2, … 8
for (int i=0;  i<=10; i+=2) //6  rep, i = 0, 2, … 10
```

Funciones

- Código más claro y limpio
- Reutilización de código
- Código más fácil de mantener
- Bibliotecas de terceros
- Pueden recibir parámetros
- Pueden devolver un valor

Declaración y uso de funciones

```
tipoDevuelto nombreFuncion(tipo parametro1, … , tipo parametroN){
        líneas de Código de la función
        …
        return;
}
```

Declaración:

```
void controlSalida(int pin, bool sentido){
   if(sentido){
      digitalWrite(pin,HIGH);
   }
}
```

Llamada:

- `controlSalida(3,true);`

Entrada - salida digital

`pinMode(pin, función)`

- Define la función del pin
- Normalmente en setup()
- Función: **INPUT, OUTPUT, INPUT_PULLUP**

`digitalRead(pin)`

- Devuelve el nivel presente en la entrada
- 0 / LOW / false si se detecta un 0 en el pin
- 1 / HIGH / true si se detecta un 1 en el pin

`digitalWrite(pin, estado)`

- Establece el estado de un pin de salida
- Nivel alto: 1, HIGH, true
- Nivel bajo: 0, LOW, false

TEMPORIZACIÓN

delay(milisegundos)

- Espera el tiempo pasado como argumento
- Bloquea la ejecución del programa (se pueden perder eventos)
- **`delayMicroseconds(microsegundos)`** para tiempos más pequeños

millis()

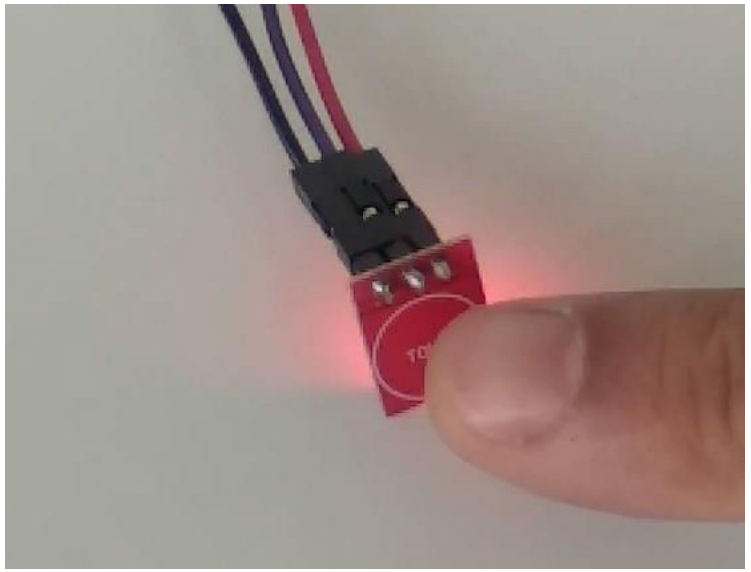
- Devuelve los milisegundos transcurridos desde el inicio del sketch en UINT
- Misma funcionalidad que con `delay()` sin dejar de atender el sistema
- Desbordamiento al cabo de un tiempo en función del tipo de variable usado

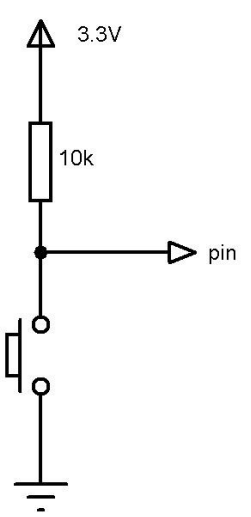
Temporización con `millis()`

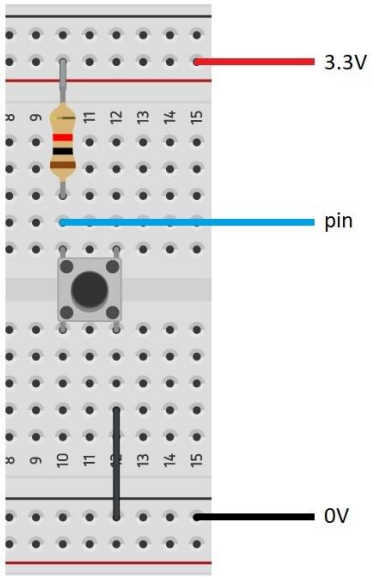
```
msAct = millis();  //leer tiempo actual
if(msAct - msAnt >= 1000){
      msAnt = msAct;      //actualizar msAnt
      tareas a realizar
}
```

Pulsadores

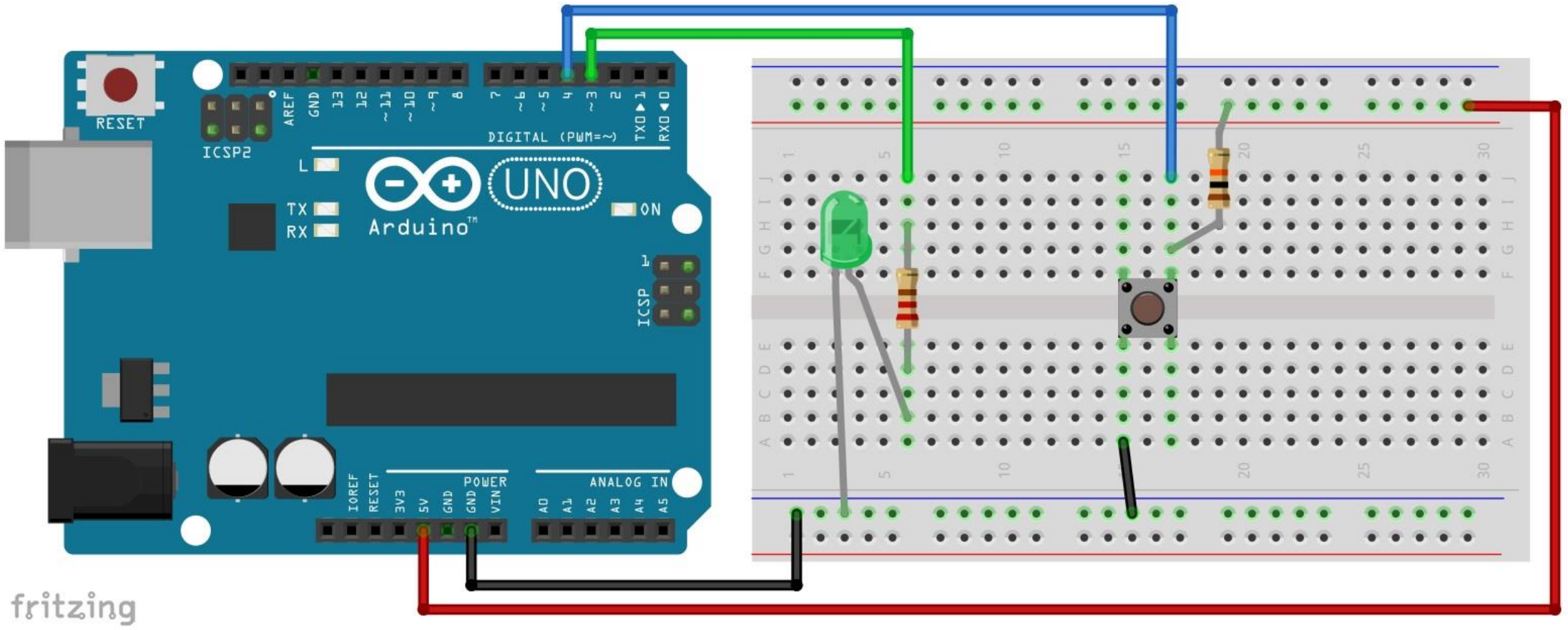
- Mecánicos
  - Necesitan resistencia de pull-up
  - 1 lógico en reposo
  - Rebotes
- Táctiles
  - Necesitan alimentación
  - 0 lógico en reposo
  - Sin rebotes







Telerruptor con pulsador



Ejercicios

- Hacer parpadear LED con `delay()`
- Leer pulsador y escribir estado en salida
- Hacer parpadear LED con `millis()`
- Telerruptor con dos pulsadores
- Telerruptor con un pulsador

Sesión II

Puerto serie

E/S analógica

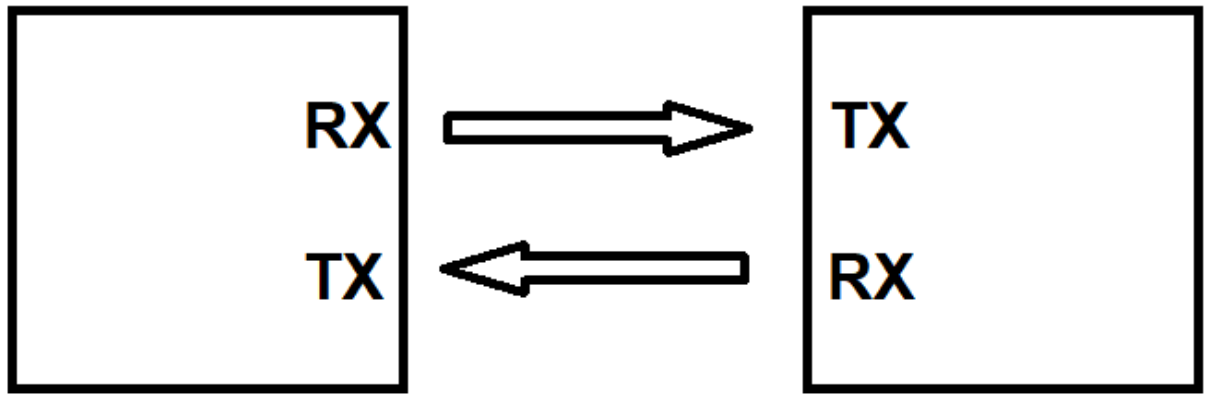
Bibliotecas de terceros

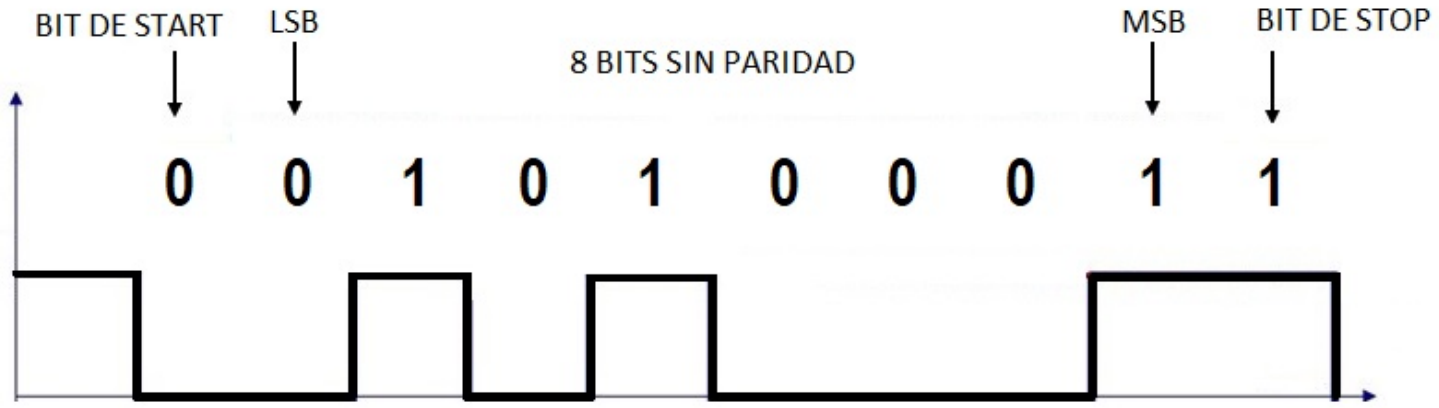
Interfaces de comunicación

Canales	Reloj	Conectividad
• Serie • Paralelo	• Síncrona • Asíncrona	• Punto a punto • En bus

Puerto serie RS232

- Comunicación serie
- Asíncrono
- Punto a punto
- Bidireccional
- Conexión cruzada
- Muy utilizado





Puerto serie en Arduino

- Para comunicación con el PC y otros periféricos
- Arduino UNO → un único puerto serie
- Puerto serie sofware
- IDE de Arduino → Herramientas → Monitor Serie
- Misma velocidad que en Arduino

Puerto Serie I

`Serial.begin(baudios)`

- Inicia el puerto serie
- Normalmente en `setup()`
- Mismo bitrate que en PC

`Serial.available()`

- Estado del buffer de entrada
- **`true`** si se han llegado datos por el puerto serie
- **`false`** cuando se leen los datos y se vacía el buffer

`Serial.read()`

- Lee un carácter recibido por el puerto serie
- El dato leído se elimina del buffer de entrada
- Devuelve un valor de tipo `char`

Puerto Serie II

`Serial.write(carácter)`

- Envía un carácter por el puerto serie
- Solo caracteres ASCII

`Serial.print(dato)`

- Envía un dato complejo por el puerto serie
- cadena, número, variable, etc.

`Serial.println(dato)`

- Mismo uso que `print`
- Inserta un salto de carro al final

Control de salidas desde PC

```
if(Serial.available()){
  char c = Serial.read();
  switch (c){
      case '0': digitalWrite(LEDRojo,LOW);  break;
      case '1': digitalWrite(LEDRojo,HIGH); break;
      case '2': digitalWrite(LEDVerde,LOW); break;
      case '3': digitalWrite(LEDVerde,HIGH);  break;
      default : Serial.print("opción incorrecta");  break;
  }
}
```

Entradas analógicas

Módulo ADC (Analoge to Digital Converter)

Integrado en varios microcontroladores

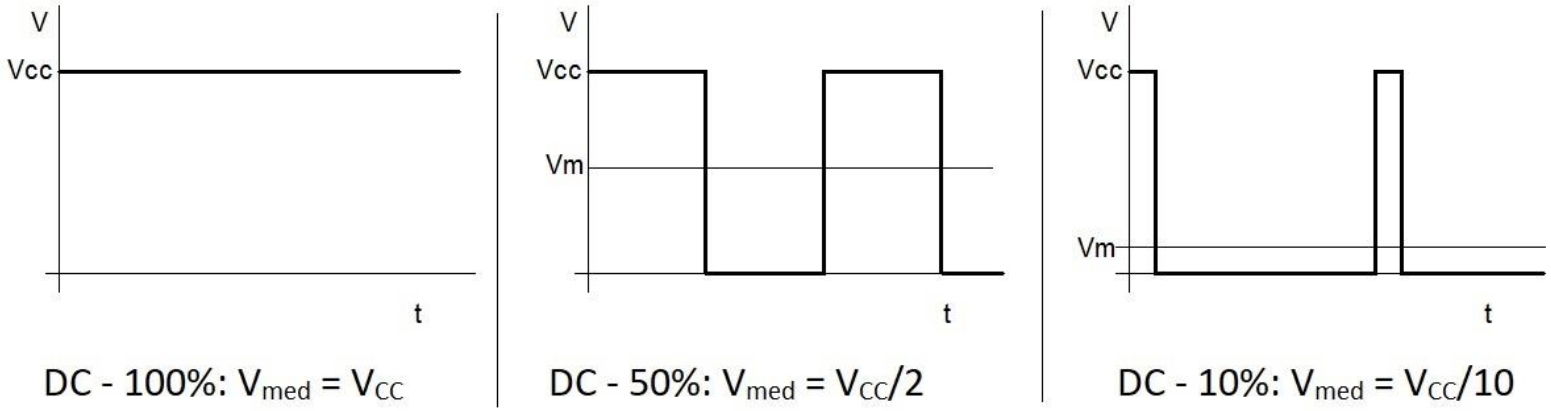
Arduino UNO: Un ADC con 6 entradas

Fondo de escala: 0 – 5V (y Vref externa)

Resolución: 10 bits (0 -1023)

Salida analógica PWM

- No es analógica
- PWM (Pulse Widh Modulation)
- Ciclo de trabajo (0 – 100%)
- Resolución 8 bits (0 – 255)



Entrada - salida analógica

`analogRead(pin)`

- Devuelve el valor en un pin analógico (A0 – A5)
- Entero entre 0 y 1023
- Fondo de escala 0 – 5V

`analogReference (tipo)`

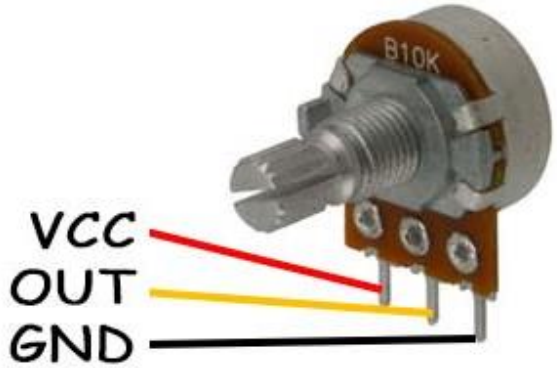
- Permite cambiar el fondo de escala
- `INTERNAL` para referencia interna de 1.1V
- `EXTERNAL` para referencia en pin AREF

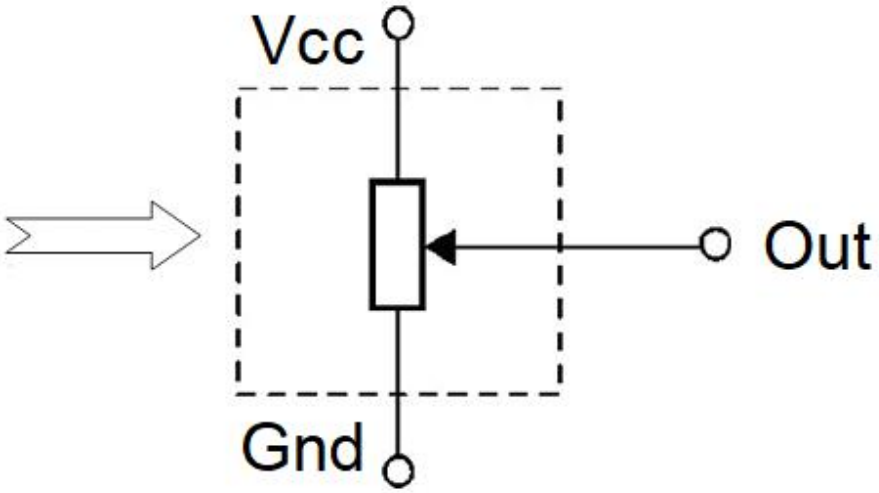
`analogWrite(pin, CT)`

- Entrega en el pin una onda PWM
- Ciclo de trabajo (CT) entre 0 y 255
- Solo los pines marcados con ~ (3, 5, 6, 9, 10, 11)

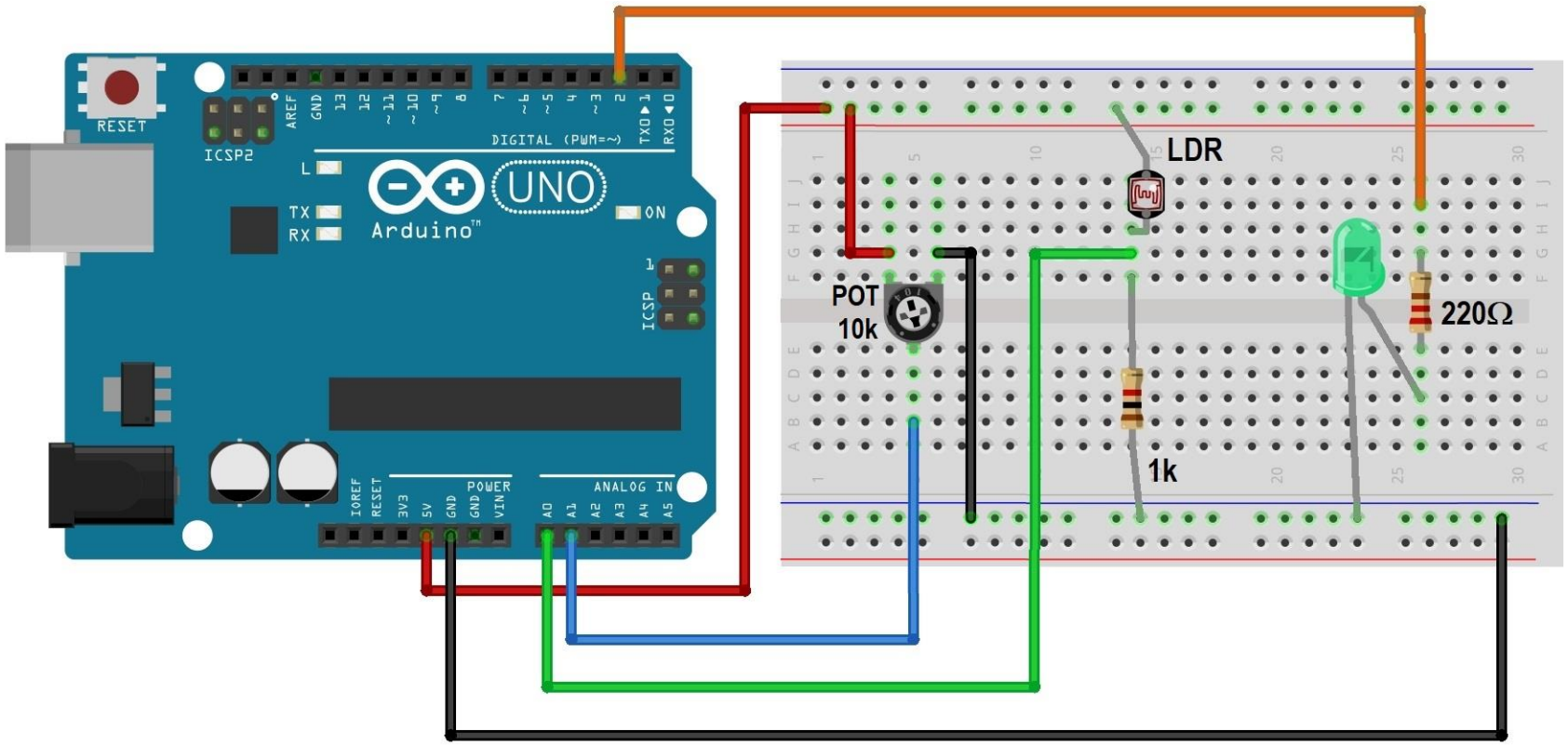
Elementos analógicos

- Potenciómetro
- Joystick
- LDR
- LM35



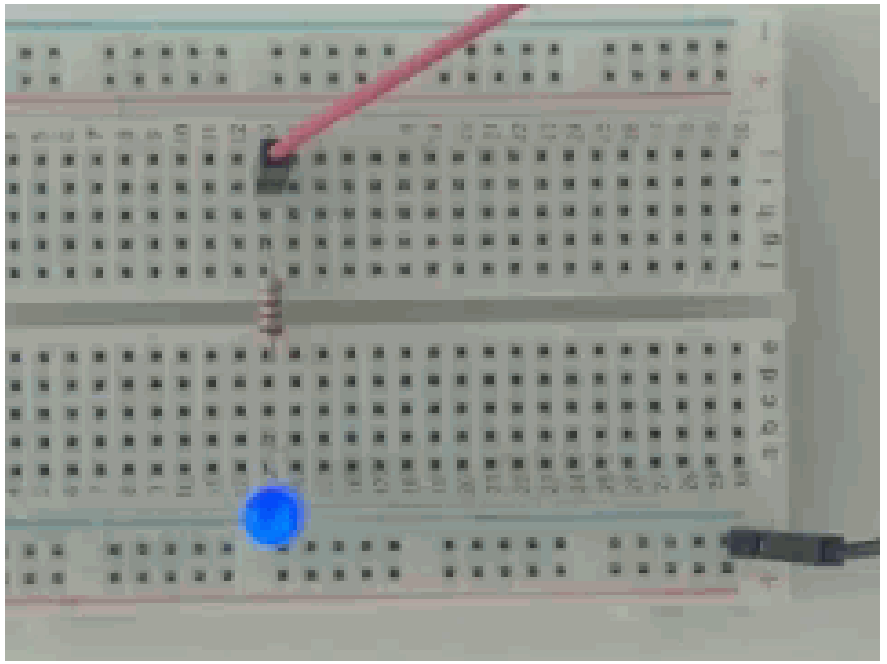


Sistema de iluminación automática



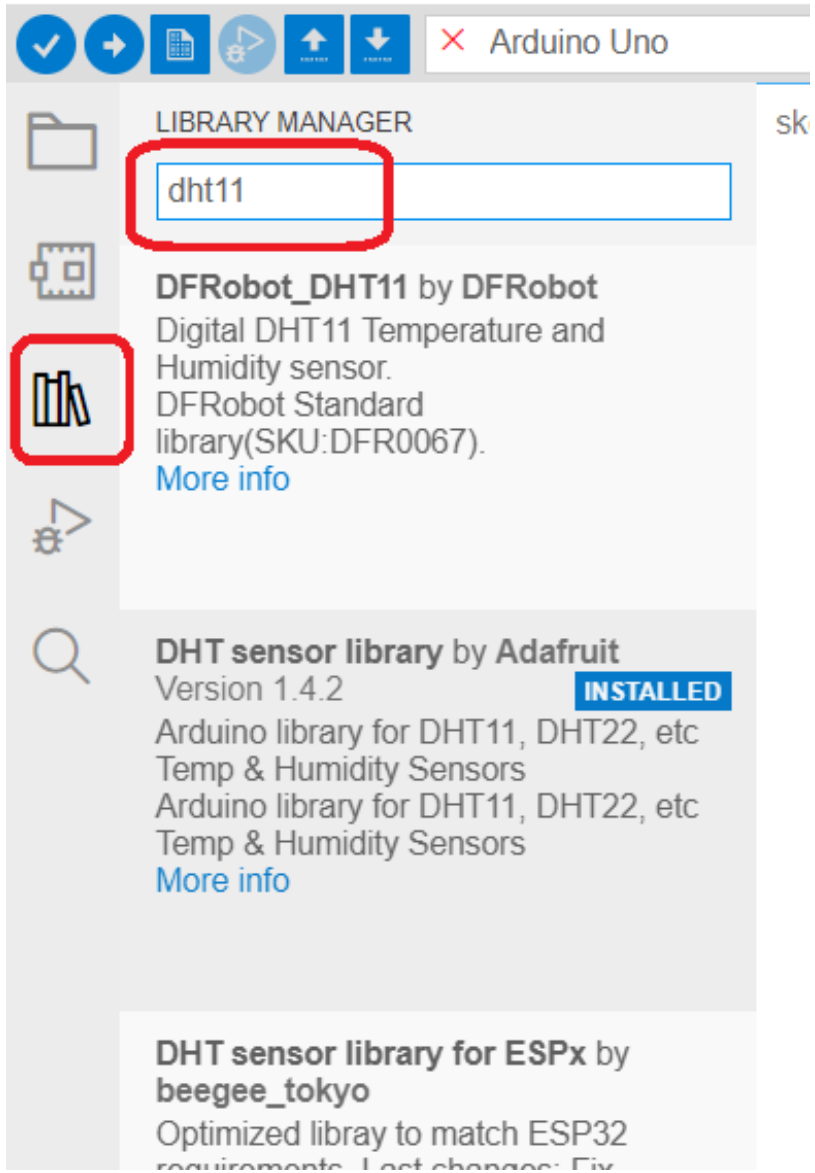
Control de iluminación PWM

```
for(int i=0;i<256;i++){
  analogWrite(salida,i);
  delay(1);
}
for(int i=255;i>=0;i--){
  analogWrite(salida,i);
  delay(1);
}
```



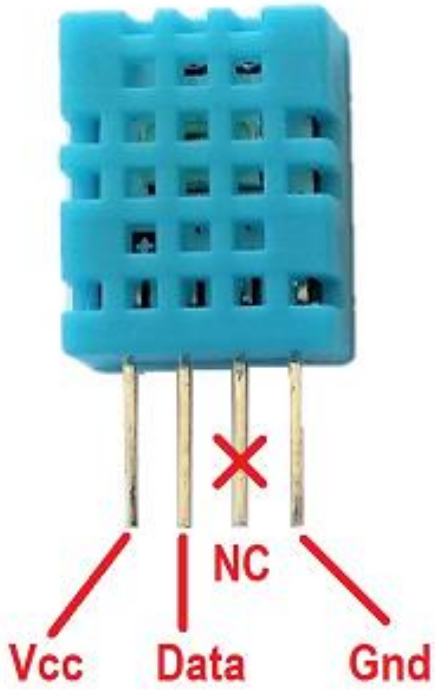
Bibliotecas

- Ficheros desarrollados por terceros
- Incluyen ejemplos de uso
- Gestor de librerías
- Directiva #include



Sensor de Tª DHT11

- Sensor de bajo coste
- Humedad y temperatura
- Poco preciso (±2 ºC)
- Conexión one-wire
- Salida en colector abierto



Uso de sensor DHT11

```
#include <DHT.h>
DHT dht(DHTPin, DHTTYPE);

//iniciar en setup()
dht.begin();

//realizar lecturas
float h = dht.readHumidity();
float t = dht.readTemperature();
```

Ejercicios

- Enviar cuenta por puerto serie
- Activar LEDs desde PC
- Leer potenciómetro y enviar por puerto serie
- Detector de niveles con potenciómetro
- Lectura de LDR. Iluminación automática
- Control de salida analógica con potenciómetro
- Parpadeo de Leds suave
- Lectura de temperatura con DHT11

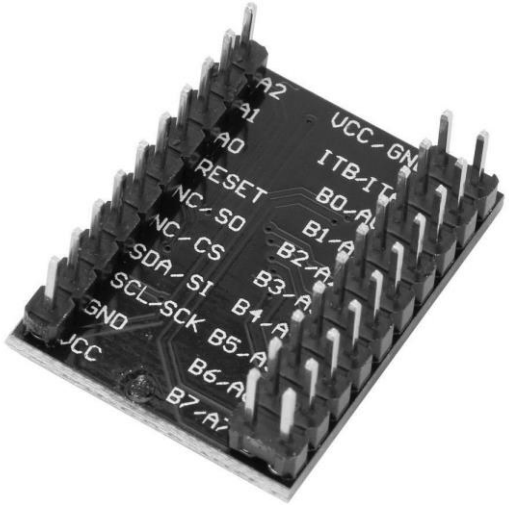
Sesión III

Buses de comunicación

Display LCD 1602

Control de climatización

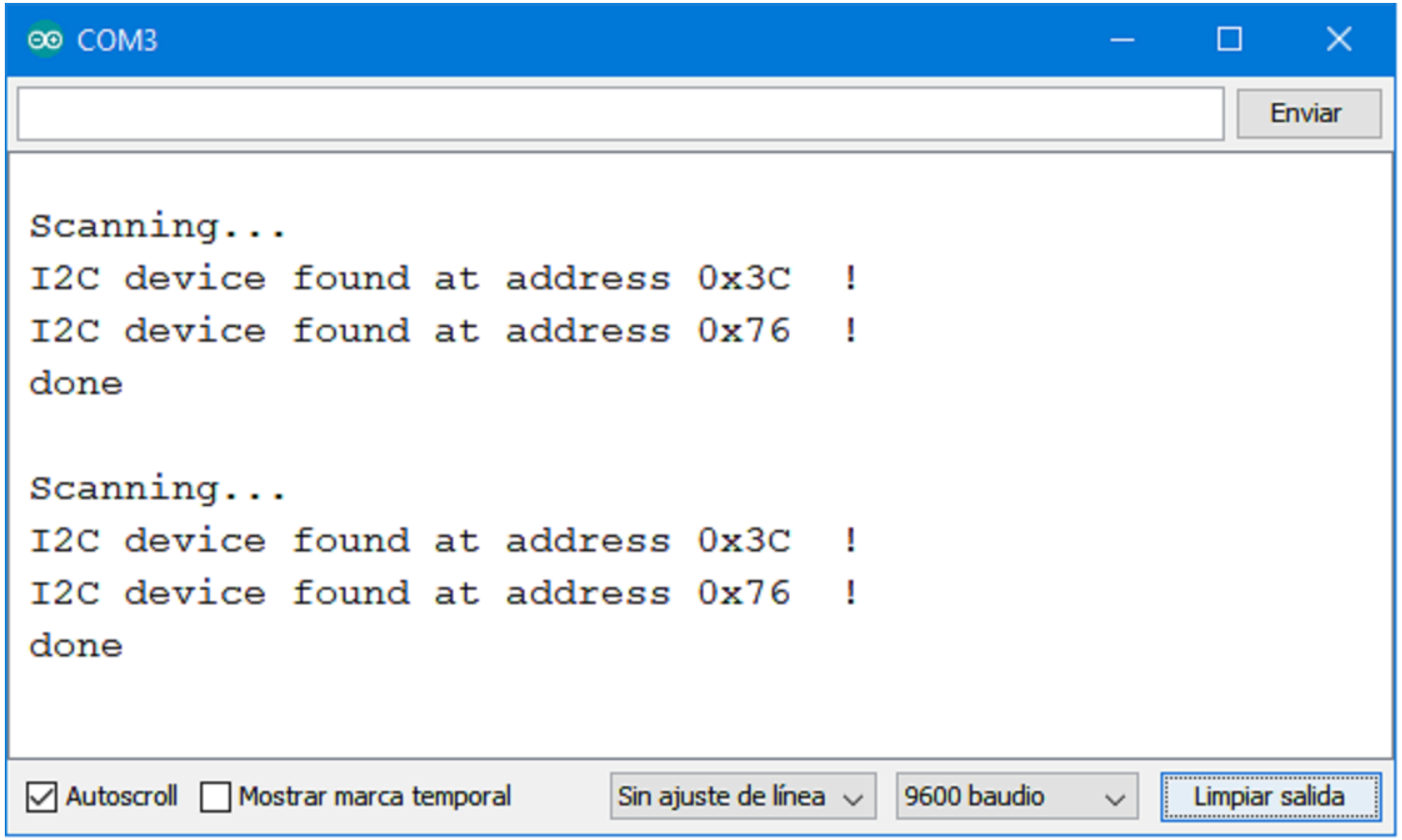
BUS I2C



- Bus de comunicación serie síncrono M/S
- Dos señales con resistencias de pull-up
  - SCL para reloj (pin A4)
  - SDA para transmisión de datos bidireccional (pin A5)
- Hasta 112 dispositivos
- Pantallas, sensores, expansores, ADC…

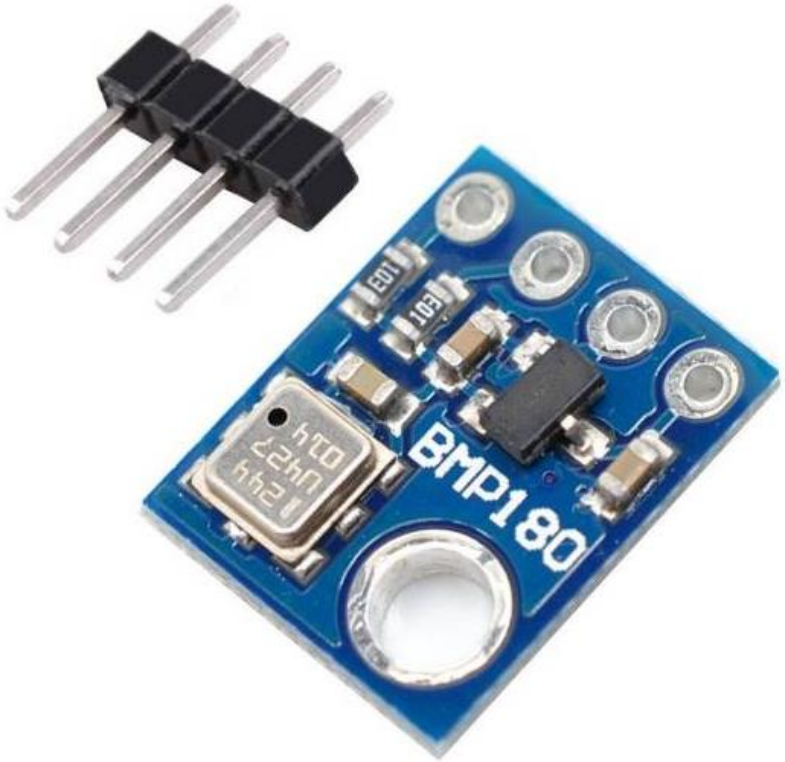
I2C scanner

- Busca dispositivos conectados
- Devuelve la dirección
- Útil cuando se desconoce la dirección



BMP180

- Sensor de presión, temperatura y altura
- Evolución de BMP085
- Conexión I2C en la dirección 0x77
- Presión: 300~1100 hPa. Precisión: 0.02 hPa
- Temperatura: -40 ~ 85°C. Precisión: 2 ºC
- Alternativas: DHT11, DHT22, BMP280, BME280



Uso de sensor BMP180

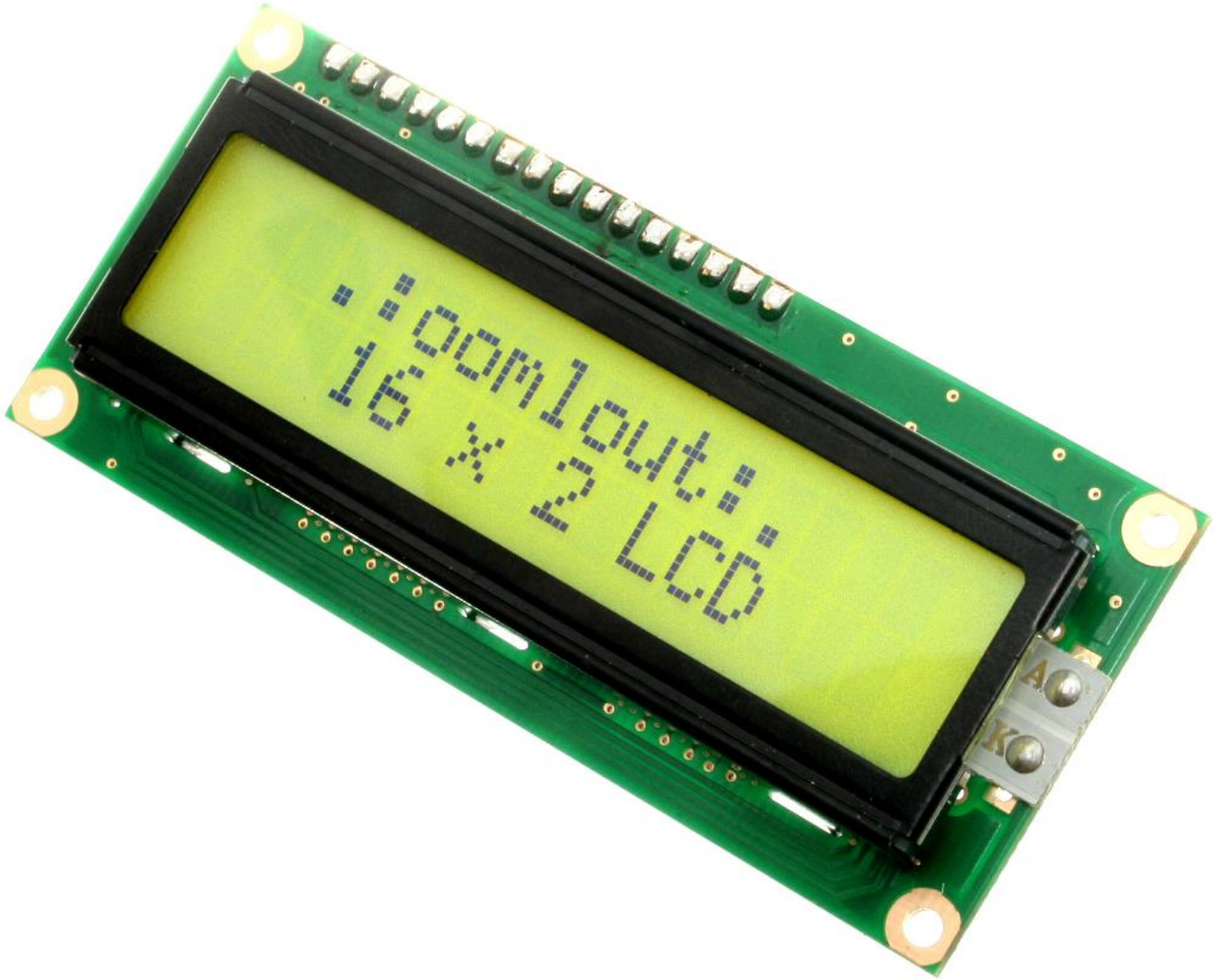
```
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp;

//iniciar en setup()
bmp.begin();

//realizar lecturas
float t = dht.readTemperature();
float h = dht.readPressure();
float p = dht.readAltitude();
```

Display LCD 1602

- Display alfanumérico
- 2 filas x 16 columnas
- Retroiluminado
- Necesita de 7 a 11 pines



Adaptador I2C

- Permite conexión con solo 2 pines
- Incorporado a adquirido por separado
- Control de contraste
- Ajuste de dirección
- Control de retroiluminación
- Liqu

Biblioteca **LiquidCrystalI2C**
Frank de Brabander

Función	Descripción
`LiquidCrystal_I2C(Add,c,r)`	Crea una instancia de la clase
`init()`	Inicializa el display LCD
`clear()`	Borra la pantalla y sitúa el cursor en la posición 0,0
`setCursor(col, row)`	Coloca el cursor en la posición col, row
`print(text)`	Escribe un texto en la posición actual
`printf(text)`	Imprime texto formateado
`scrollDisplayLeft()`	Desplaza el texto en pantalla a la izquierda
`scrollDisplayRight()`	Desplaza el texto en pantalla a la derecha
`backlight() / noBacklight()`	Activa / desactiva la retroiluminación del display
`createChar (dir, datos)`	Crea un carácter personalizado en la RAM del display

Ejemplo de uso LCD 1602

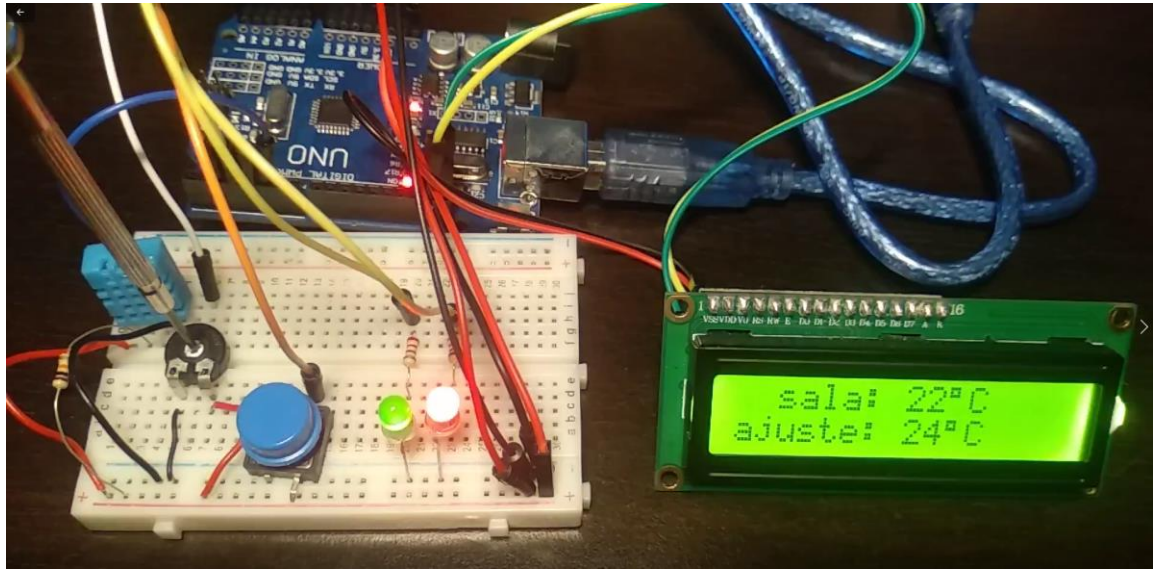
```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C  lcd(0x27,16,2);

lcd.init();          //en setup
lcd.backlight();

lcd.setCursor(0, 0);  //en loop
lcd.print("Cuenta");
lcd.setCursor(0, 1);
lcd.print(horas);
```

Control de calefacción

- Medida de temperatura con DHT11
- Ajuste de temperatura con potenciómetro
- Display mostrando ambas temperaturas
- Pulsador para activar/desactivar el sistema
- Led indicador de estado del sistema
- Led indicador de estado de la caldera



Ejercicios

Leer temperatura con BMP180

Reloj (HH:mm:ss) en display LCD 1602

Control de calefacción con display

Sesión IV

Control de cargas

Motor paso a paso

Servomotor

Control de cargas de potencia

Relés

- Libres de tensión
- Grandes cargas
- Solo control ON / OFF

Transistores

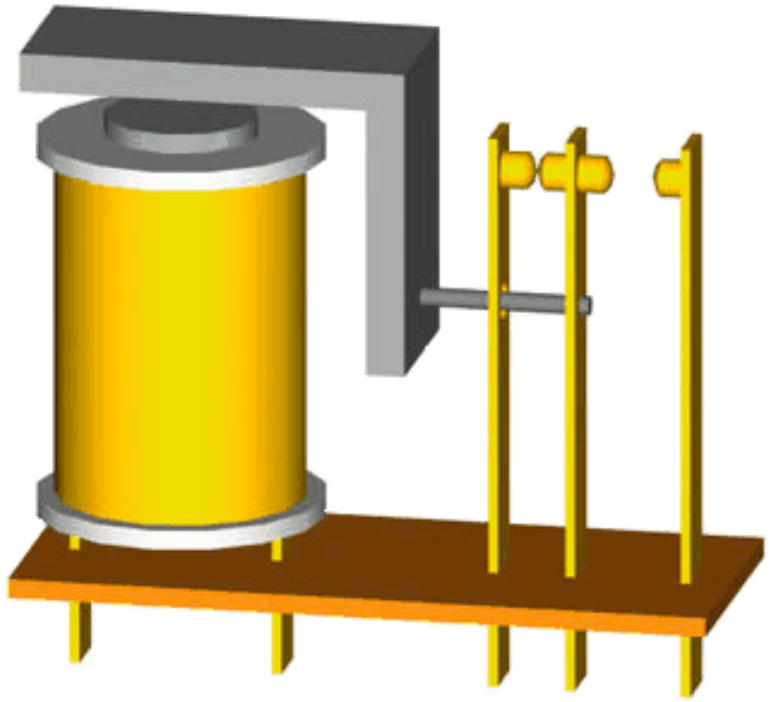
- Solo CC
- Limitación de potencia
- Control PWM

Drivers

- Transistores integrados
- Montaje más simple
- Limitaciones de potencia

Relé

- Circuito electromagnético
- Mecánico
- Libre de tensión
- No se puede activar directamente
- Módulo Relé



Transistor

Interruptor electrónico

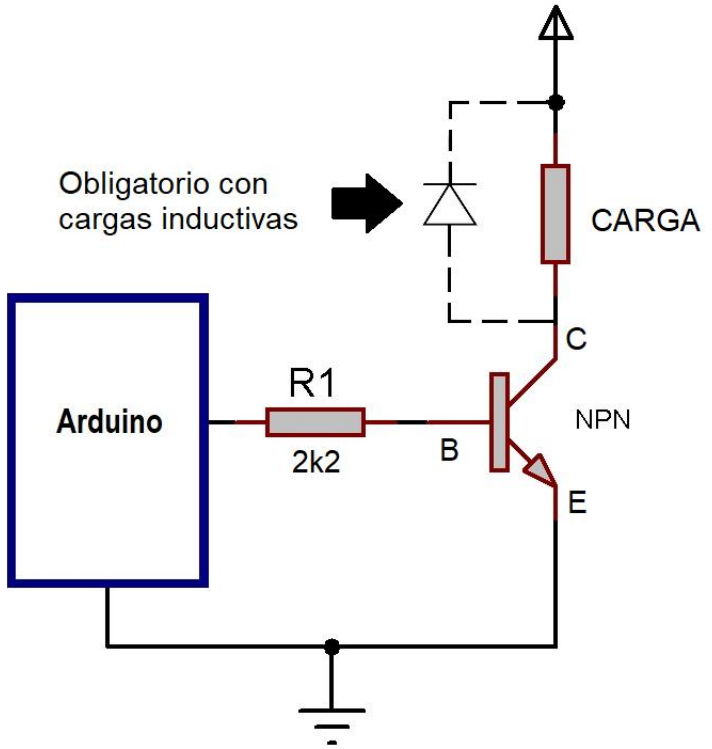
- Conexión directa al micro
- Tensiones de CC
- Unir tierras de las fuentes

Tres terminales

- Emisor
- Base
- Colector

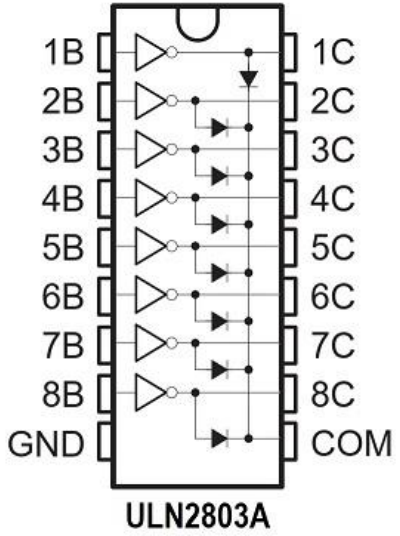
Cargas Inductivas

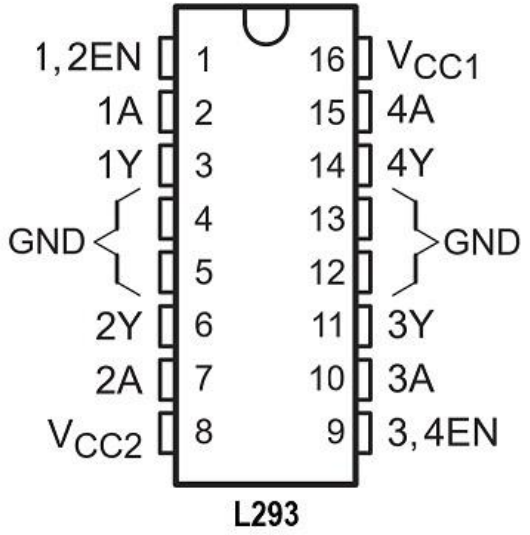
- Se puede destruir el transistor
- Diodo en antiparalelo
- Relés, motores, etc.



Drivers

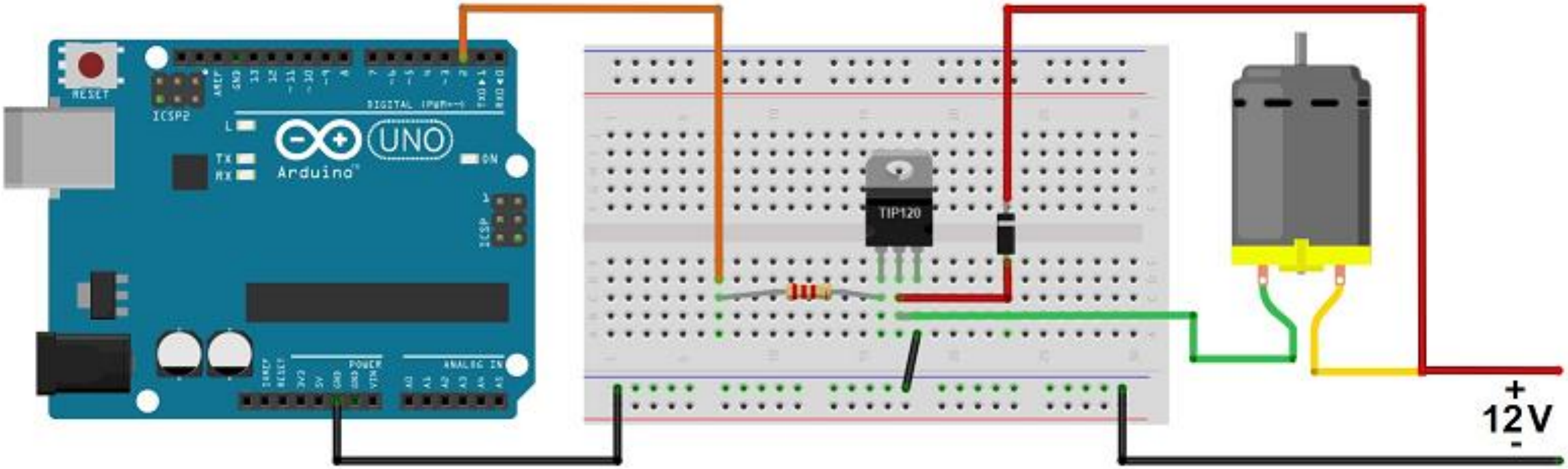
- Transistores integrados
- ULN2003 /ULN2803
- L293 / L298
  - Control del sentido de giro





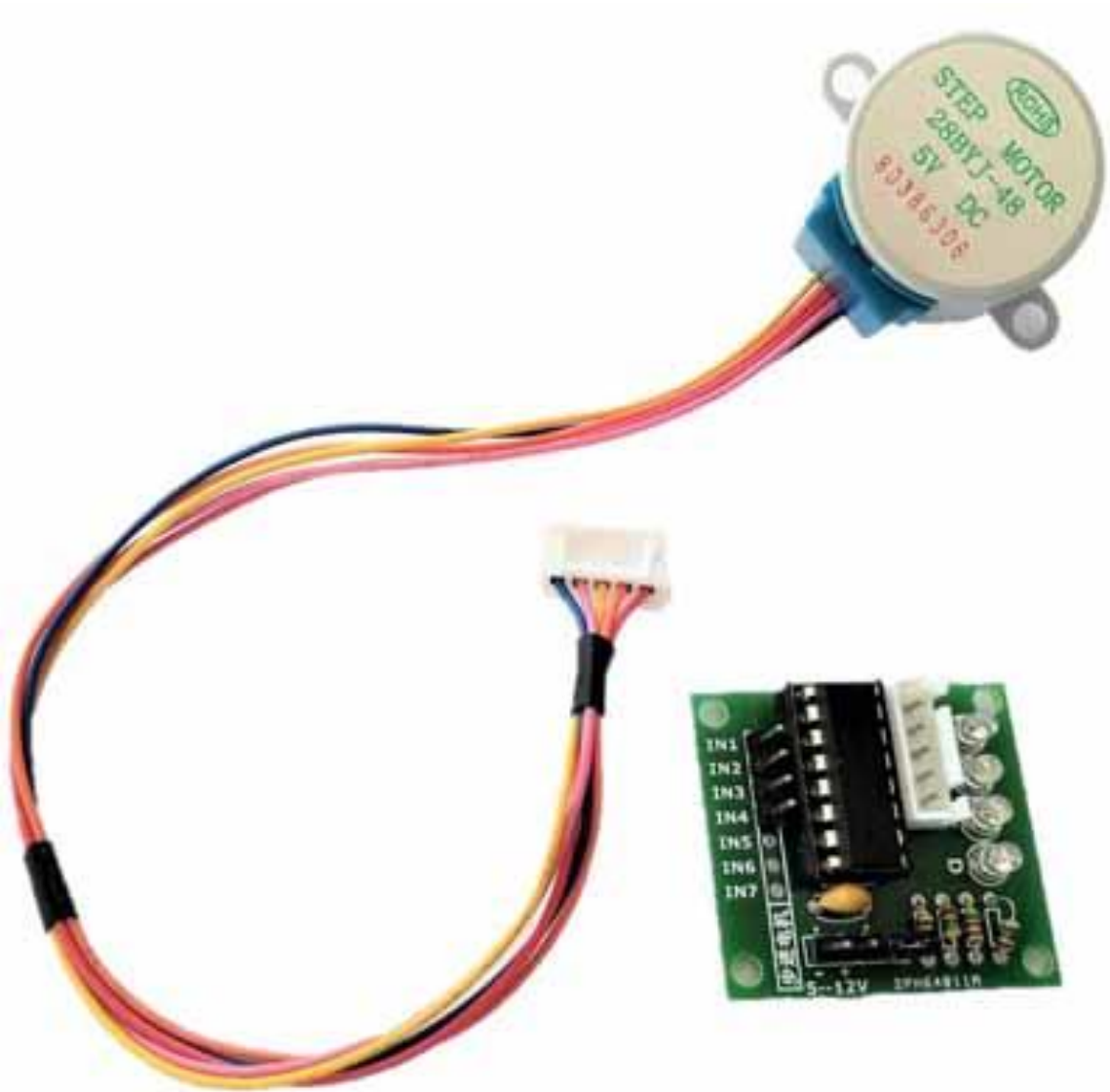
Motor de corriente continua

- Control con un pin
  - Control de velocidad con PWM
  - Transistor o driver ULN2003
- Sentido del giro cambiando polaridad
  - Dos pines
  - Driver L293



Motor paso a paso

- Control con 4 pines
- Se indica el número de pasos
- Ajuste preciso
- Stepper 28BYJ-48
- Placa con driver
- 2048 pasos / vuelta



Servo

- Permite ajustar el ángulo de posicionamiento
- Control mediante un único pin
- No necesita driver
- Microservo SG90
- Pequeños robots

Control de servomotor

```
#include <Servo.h>
Servo servo1;           // crear objeto Servo

// En setup()
servo1.attach(9);       // servo en pin 9

servo1.write(0);        //posicionar en 0º
servo1.write(180);      //posicionar en 180º
```

Ejercicios

- Control bombilla relé
- Control motor CC
- Control Stepper
- Control servo con potenciómetro

Referencias

- INTERNET DE LAS COSAS **(IOT)** CON **ARDUINO**

  Jesús Pizarro Peláez

  Editorial Paraninfo

  ISBN 9788428341868

- INTERNET DE LAS COSAS **(IOT)** CON **ESP**

  Jesús Pizarro Peláez

  Editorial Paraninfo

  ISBN 9788428344968