Introducción al entorno Arduino

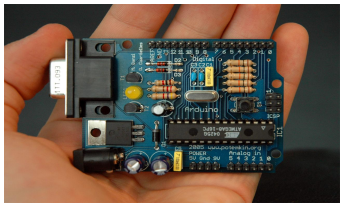
Departamento de Automática

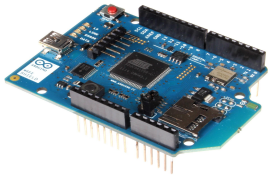
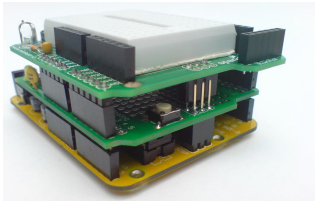
Universidad de Alcalá

/gso>

¿Qué es Arduino?

- Compañía de desarrollo de circuitos integrados e IDEs
- Proyecto hardware y software de código abierto
- Diseños abiertos basados en placas con microcontroladores
- Expandible mediante shields a través de interfaces de entrada/salida
- Programación empleando un subconjunto de C/C++
- Carga de programas usando un PC a través del puerto USB
- Existe una familia entera de productos:

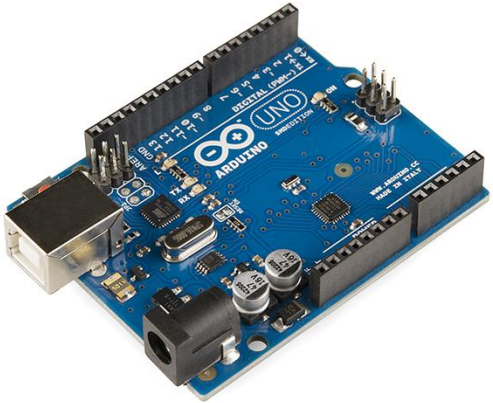


Arduino Uno

Características técnicas

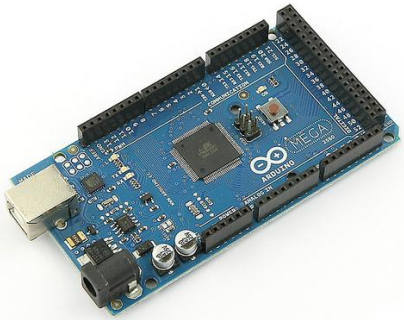
- Microcontrolador ATmega328P
- Voltaje operativo: 5V
- Pines de Entrada/Salida digitales: 14 (6 PWM)
- Pines de entrada analógica: 6
- Memoria Flash de 32 KB (0,5 KB *bootloader*)
- SRAM de 2 KB
- EEPROM de 1 KB
- Frecuencia de trabajo: 16 MHz
- Dimensiones: 68,6 x 53,4 mm
- Peso: 25 g



Arduino Mega 2560

Características técnicas

- Microcontrolador ATmega2560
- Voltaje operativo: 5V
- Pines de Entrada/Salida digitales: 54 (15 PWM)
- Pines de entrada analógica: 16
- Memoria Flash de 256 KB (8 KB *bootloader*)
- SRAM de 8 KB
- EEPROM de 4 KB
- Frecuencia de trabajo: 16 MHz
- Dimensiones: 101,52 x 53,3 mm
- Peso: 37 g

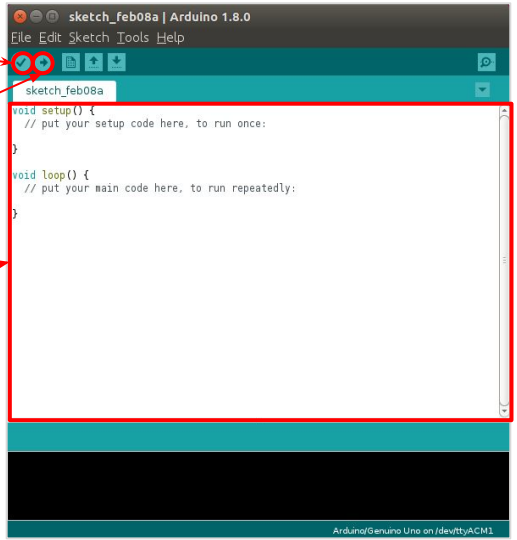


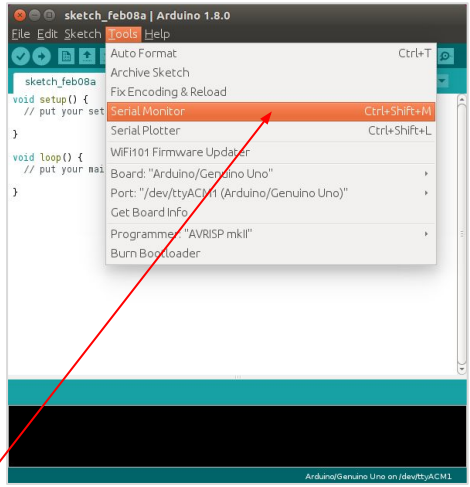
Entorno de desarrollo

Verify

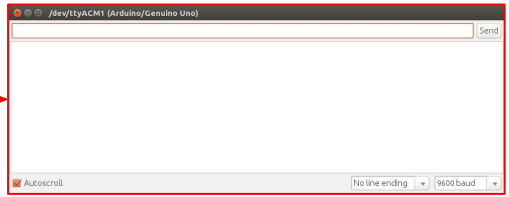
Upload

Editor de código





Monitor de la puerta serie



Primeros pasos

- Estructura de un programa (*sketch*) Arduino:
  - `void setup()`
    - Función de inicialización
    - Es ejecutada una única vez tras el arranque del sistema
    - Se utiliza para configurar las entradas/salidas e inicializar variables y bibliotecas
  - `void loop()`
    - Función principal de ejecución
    - Se ejecuta de forma permanente

Primeros pasos

- Programa propuesto:
  - Envío de mensajes a través de la puerta serie (`Serial.print()`)
  - Lectura del tiempo del sistema (`millis()`)
  - Empleo de la función de retardo (`delay()`)
- El programa debe realizar los siguientes pasos:
  - Configurar la puerta serie a 9600 baudios (`Serial.begin()`)
  - Enviar un mensaje cada segundo con el siguiente contenido:
    - Serial output. Time: XXX s
    - Donde XXX son los segundos transcurridos desde el arranque
- Esqueleto del programa: `first_steps.ino`

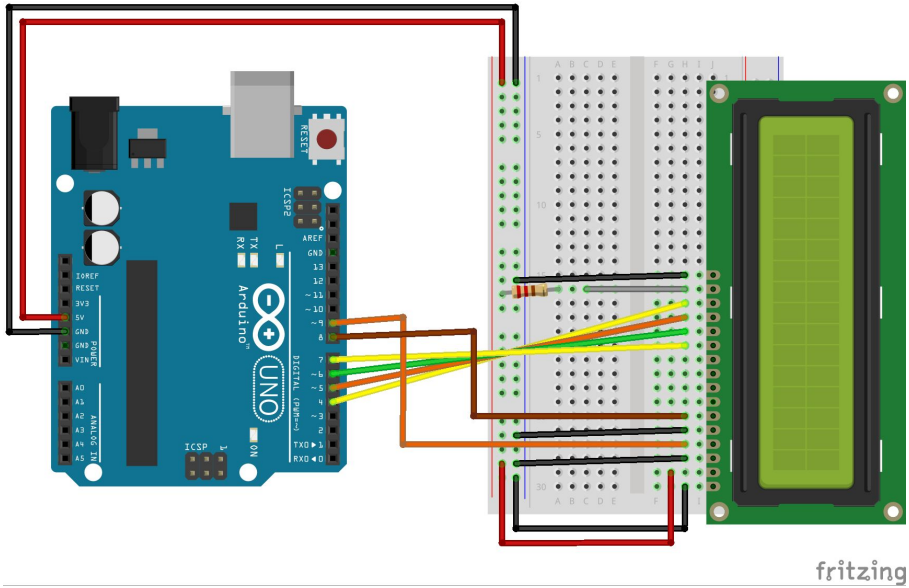
El display de cristal líquido

- Display de cristal líquido JHD659
- Muestra dos líneas de 16 caracteres/línea
- Existe una biblioteca de funciones externa para manejar el display:
  - Configurar el display (`LiquidCrystal()`, `lcd.begin()`)
  - Mover cursor de escritura a una posición (`lcd.setCursor()`)
  - Escribir en el display a partir del cursor (`lcd.print()`)

El display de cristal líquido

Conexiones:

- GND (Pin 1) → GND
- VCC (Pin 2) → VCC
- V0 (Pin 3) → GND
- RS (Pin 4) → Digital OUT 9
- R/W (Pin 5) → GND
- E (Pin 6) → Digital OUT 8
- DB4 (Pin 11) → Digital OUT 7
- DB5 (Pin 12) → Digital OUT 6
- DB6 (Pin 13) → Digital OUT 5
- DB7 (Pin 14) → Digital OUT 4
- LED+ (Pin 15) → Resistencia de 220 Ω → VCC
- LED- (Pin 16) → GND



El display de cristal líquido

- Programa propuesto:
  - Mostrar mensajes en el display de cristal líquido
- El programa debe realizar los siguientes pasos:
  - Configurar el display de cristal líquido
  - Mostrar el siguiente mensaje en la línea 0:
    - LCD Message
  - Mostrar cada segundo el siguiente mensaje en la línea 1:
    - Time: XXX s
    - Donde XXX son los segundos transcurridos desde el arranque
- Esqueleto del programa: `lcd.ino`

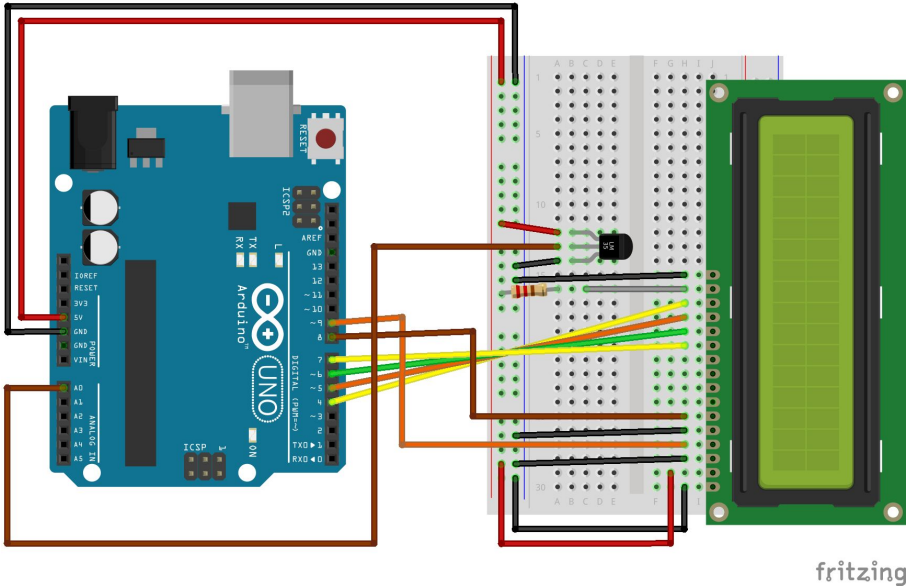
Entrada analógica: sensor de temperatura

- Sensor de temperatura TMP36
  - Rango de temperaturas: -40 ºC, +125 ºC
  - 10 mV por grado centígrado
  - Offset de medio voltio (0,5 V a 0 ºC)
- Entradas analógicas en Arduino:
  - ADC de 10 bits: 1024 niveles lineales entre 0 y 5 V
  - Función nativa para leer el valor del ADC (`analogRead()`)

Entrada analógica: sensor de temperatura

Conexiones:

- Pin 1 → VCC
- Pin 2 → Analog Input A0
- Pin 3 → GND



Entrada analógica: sensor de temperatura

- Programa propuesto:
  - Leer y mostrar la temperatura
- El programa debe realizar los siguientes pasos:
  - Configurar el display de cristal líquido y la entrada analógica
  - Mostrar **cada 250 milisegundos** el siguiente mensaje en la línea 0:
    - Degrees C: DD.DD
    - Donde DD.DD es la temperatura medida en grados centígrados
- Esqueleto del programa: `temp_sensor.ino`
- La medida de temperatura es demasiado variable. ¿Soluciones?

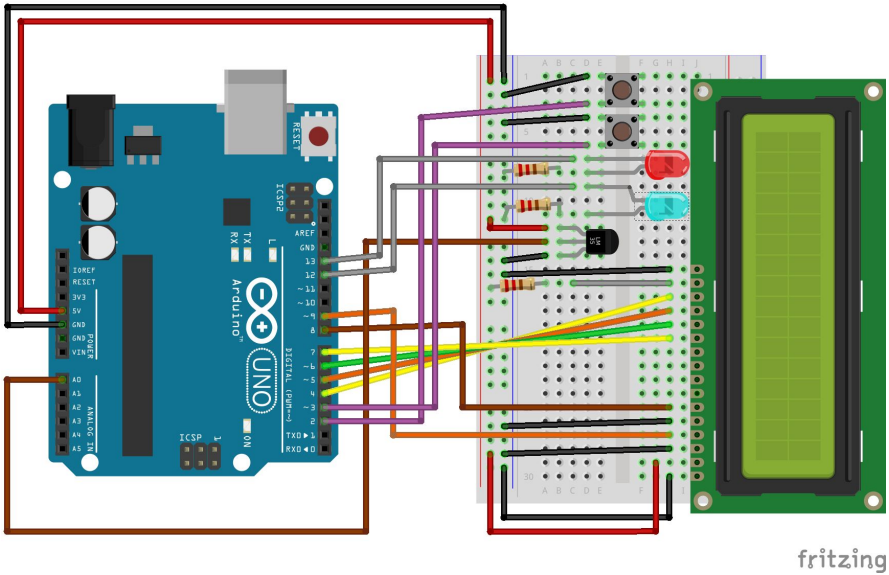
Entrada/salida digital: pulsadores y LEDs

- Cada pin digital se puede configurar como entrada o salida (`pinMode()`)
- Función nativa para leer de los pines de entrada (`digitalRead()`)
- Función nativa para escribir en los pines de salida (`digitalWrite()`)
- Voltaje de trabajo: 5 V
- Corriente recomendada: 20 mA. Corriente máxima soportada: 40 mA
- Los pines tienen integrada una resistencia de *Pull-Up* de 20-50 kΩ configurable

Entrada/salida digital: pulsadores y LEDs

Conexiones:

- Pulsador 1, Pin 4 → GND
- Pulsador 1, Pin 2 → Digital IN 2
- Pulsador 2, Pin 4 → GND
- Pulsador 2, Pin 2 → Digital IN 3
- LED rojo, ánodo → Digital OUT 13
- LED rojo, cátodo → Resistencia de 220 Ω → GND
- LED azul, ánodo → Digital OUT 12
- LED azul, cátodo → Resistencia de 220 Ω → GND



Entrada/salida digital: pulsadores y LEDs

- Programa propuesto:
  - Definir un rango de temperaturas de ± 5 grados sobre una medida de temperatura inicial
  - El rango se puede desplazar con los pulsadores
  - La temperatura actual es la media de las temperaturas de los últimos 5 segundos
  - Si la temperatura es menor que el límite inferior del rango, encender el LED azul
  - Si la temperatura es mayor que el límite superior del rango, encender el LED rojo

Entrada/salida digital: pulsadores y LEDs

- El programa debe realizar los siguientes pasos:
  - Configurar el display de cristal líquido y las entradas y salidas digitales y analógicas
  - Tomar una primera medida de temperatura y definir el rango inicial
  - Mostrar cada 250 milisegundos el siguiente mensaje en la línea 0:
    - Degrees: DD.DD
    - Donde DD.DD es la temperatura media en grados centígrados de los últimos 30 segundos
  - Mostrar el siguiente mensaje en la línea 1:
    - Range: [XX, YY]
    - Donde XX e YY son los límites inferior y superior del rango de temperaturas
  - Cada vez que se pulse el pulsador 1, decrementar los límites en una unidad y actualizar el display
  - Cada vez que se pulse el pulsador 2, incrementar los límites en una unidad y actualizar el display
  - Si la temperatura es menor que el límite inferior del rango, encender el LED azul
  - Si la temperatura es mayor que el límite superior del rango, encender el LED rojo
- Esqueleto del programa: `temp_range.ino`

Referencias

- Página oficial de Arduino. https://www.arduino.cc/
- Imágenes utilizadas:
  - Arduino Serial Board, de Nicholas Zambetti, publicada bajo licencia CC BY-SA 3.0.
  - Arduino Nano, de David Mellis, publicada bajo licencia CC BY 2.0.
  - WiFi Shield for Arduino, de oomlout, publicada bajo licencia CC BY-SA 2.0.
  - Arduino Protoboard Shield, de Marlon J. Manrique, publicada bajo licencia CC BY-SA 2.0.
  - Arduino Uno R-3, de SparkFun Electronics from Boulder, USA, publicada bajo licencia CC BY 2.0.
  - Arduino Mega 2560, de Snootlab, publicada bajo licencia CC BY 2.0.
- Los gráficos de los circuitos se han creado con Fritzing. http://fritzing.org/home/