Taller de Iniciación a Arduino

Presentación

Este documento sirve como apoyo visual y repositorio de código para las tres sesiones del taller.

RESUMEN: https://gemini.google.com/share/efb3c0ea058e

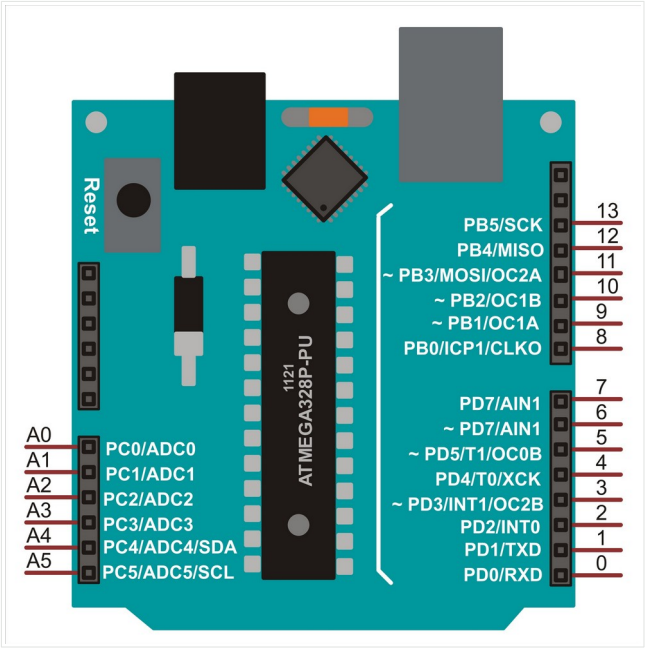
SESIÓN 1: Introducción, IDE y Control de LEDs

1. El Ecosistema Arduino

A. La Placa Arduino Uno

La Placa Arduino Uno es el corazón de nuestro sistema. Es un microcontrolador que puede ser programado para interactuar con el mundo físico.

- **Pines Digitales (0-13):** Se usan como entradas (ON/OFF) o salidas (HIGH/LOW). Como salidas, el microcontrolador puede entregar una señal de 5 voltios (HIGH) o 0 voltios (LOW). Estos pines son fundamentales para controlar actuadores sencillos como LEDs o motores, o para leer el estado de interruptores y pulsadores. Los pines marcados con ~ (3, 5, 6, 9, 10, 11) soportan **PWM** (Pulse Width Modulation - Modulación por Ancho de Pulso), una técnica que permite simular una salida analógica variando el tiempo que la señal está en HIGH, lo cual es ideal para controlar el brillo de un LED o la velocidad de un motor.
- **Pines Analógicos (A0-A5):** Estos pines están diseñados para leer valores variables y continuos, como el voltaje que proviene de sensores de temperatura, luz o potenciómetros. La placa utiliza un Conversor Analógico-Digital (ADC) interno de 10 bits, lo que significa que traduce el voltaje de entrada (de 0 a 5 voltios) en un valor entero que va de 0 a 1023.



Shutterstock Explorar

Este rango de 1024 valores nos da la precisión necesaria para percibir pequeños cambios en las condiciones físicas.

- **Pines de Alimentación (Power):** Además de recibir energía por el conector USB o de barril, la placa proporciona voltajes regulados. **5V** y **3.3V** son cruciales para alimentar otros componentes y sensores externos. **GND** (Tierra) es la referencia de 0 voltios que debe ser común a todos los elementos del circuito.

B. El Software (IDE)

El Entorno de Desarrollo Integrado (IDE) es donde escribimos y cargamos el código que define el comportamiento del microcontrolador. Un Sketch de Arduino siempre debe contener las siguientes dos funciones estructurales:

1. **void setup():** Esta función se ejecuta **una sola vez** inmediatamente después de que la placa se enciende o se reinicia. Su propósito es la configuración inicial del hardware y el software. Aquí es donde utilizamos la función pinMode() para declarar si un pin digital trabajará como entrada (INPUT) o salida (OUTPUT), y donde inicializamos las comunicaciones seriales (Serial.begin()) o las librerías que gestionan sensores complejos. Si se omite, el microcontrolador no sabrá cómo interactuar con los pines externos.
2. **void loop():** Esta función se ejecuta **continuamente** en bucle de forma indefinida, justo después de que la función setup() ha finalizado su trabajo. Es el programa principal, el corazón de la lógica de Arduino. En el loop() colocamos el código que queremos que se repita constantemente: leer el estado de un sensor, procesar un valor, tomar una decisión (lógica condicional) y, finalmente, activar un actuador. Un ciclo de ejecución en el loop() suele ser extremadamente rápido.

2. Control de Salidas Digitales (LED)

Tarea: Conectar y hacer parpadear un LED.
Componentes: 1 LED, 1 Resistencia (220 Ohm), Cables Jumper.
Esquema de Conexión:

- El ánodo (pata larga del LED) se conecta al Pin Digital (ej. Pin 13).
- El cátodo (pata corta del LED) se conecta a la Resistencia (220 Ohm).
- La Resistencia se conecta al pin **GND** (Tierra).

Código 1: Parpadeo Básico (Blink)
Este código demuestra cómo configurar el pin 13 como una salida y alternar su estado.

```
// Definimos el pin donde está conectado el LED
const int LED_PIN = 13;

void setup() {
  // Inicializa el pin 13 como una SALIDA (OUTPUT)
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // Pone el Pin a ALTO (HIGH), encendiendo el LED
  digitalWrite(LED_PIN, HIGH);
  // Espera 1000 milisegundos (1 segundo)
  delay(1000);

  // Pone el Pin a BAJO (LOW), apagando el LED
  digitalWrite(LED_PIN, LOW);
  // Espera 1000 milisegundos (1 segundo)
  delay(1000);
}
// Usamos el Pin 9, que es compatible con PWM (tiene el símbolo ~)
const int LED_PIN = 9;

void setup() {
```

```
  // No necesitamos pinMode() para analogWrite en pines PWM
}
```

Código 2: Salida Analógica (Fading con PWM)
Usamos un pin PWM (~) para controlar el brillo del LED. analogWrite() acepta valores de 0 (apagado) a 255 (máximo brillo).
Código Fading PWM

```
void loop() {
  // Incrementa el brillo (fade in)
  for (int brillo = 0; brillo <= 255; brillo += 5) {
    analogWrite(LED_PIN, brillo);
    delay(30); // Pequeña pausa para ver el cambio
  }

  // Decrementa el brillo (fade out)
  for (int brillo = 255; brillo >= 0; brillo -= 5) {
    analogWrite(LED_PIN, brillo);
    delay(30);
  }
}
```

SESIÓN 2: Adquisición de Datos (Sensor BMP280)

3. Lectura de Datos de un Sensor

Tarea: Conectar un sensor de Presión y Temperatura BMP280 y leer sus datos a través de la comunicación I2C, visualizándolos en el Monitor Serie.
A. Comunicación I2C

El BMP280 usa el protocolo I2C, que solo necesita dos cables de datos más alimentación:

- **SDA (Serial Data):** Línea de datos. En Arduino Uno, se conecta al pin **A4**.
- **SCL (Serial Clock):** Línea de reloj. En Arduino Uno, se conecta al pin **A5**.
  Esquema de Conexión (BMP280 I2C):

| Pin Arduino Uno | Pin BMP280 | Función |

| :--- | :--- | :--- |

| 5V | VCC | Alimentación |

| GND | GND | Tierra |

| A4 | SDA | Datos I2C |

| A5 | SCL | Reloj I2C |

B. Instalación de Librería

Antes de compilar, debemos instalar la librería "Adafruit BMP280" desde el *Gestor de Librerías* del Arduino IDE.

Código 3: Lectura del BMP280 y Monitor Serie

Este código inicializa la comunicación Serial y I2C, lee los datos del sensor y los imprime en el Monitor Serie.

Código Lectura BMP280

```
// Incluimos las librerías necesarias para I2C y el sensor
#include <Wire.h> // Para comunicación I2C
#include <Adafruit_BMP280.h> // Para el sensor BMP280

// Creamos un objeto para manejar el sensor
Adafruit_BMP280 bmp; // Usaremos la dirección I2C por defecto

void setup() {
  // Inicializamos la comunicación Serial para enviar datos al PC
  Serial.begin(9600);
  while (!Serial); // Espera a que el Monitor Serie esté abierto (solo para placas con USB nativo)

  Serial.println(F("Prueba de Sensor BMP280"));

  // Intentamos inicializar el sensor
  if (!bmp.begin()) {
    Serial.println(F("¡No se ha encontrado un sensor BMP280 válido!"));
    while (1) delay(10); // Bucle infinito si hay error
  }

  // Configuramos el modo de muestreo (opcional, por defecto es ok)
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Modo Normal (operación continua) */
```

```
                Adafruit_BMP280::SAMPLING_X2,     /* 2x muestreo de Temperatura */
                Adafruit_BMP280::SAMPLING_X16,    /* 16x muestreo de Presión */
                Adafruit_BMP280::FILTER_X16,      /* Filtrado de 16 */
                Adafruit_BMP280::STANDBY_MS_500); /* Tiempo de espera */
}

void loop() {
  // Leemos y mostramos los valores
  Serial.print(F("Temperatura = "));
  Serial.print(bmp.readTemperature()); // Lee en grados Celsius
  Serial.println(F(" *C"));

  Serial.print(F("Presion = "));
  Serial.print(bmp.readPressure() / 100.0F); // Convierte de Pascal a hPa (milibares)
  Serial.println(F(" hPa"));

  Serial.println();
  delay(2000); // Lee cada 2 segundos
}
```

SESIÓN 3: Comunicación Inalámbrica (APC220)

4. Construcción del Sistema de Comunicación Radio

Tarea: Configurar dos placas Arduino con módulos APC220 para que una envíe los datos del BMP280 y la otra los reciba y muestre.

A. El Módulo APC220

El APC220 permite la comunicación Serial inalámbrica. Necesita ser conectado a un adaptador USB/TTL para el PC y al Arduino para la comunicación.

Configuración Previa: Antes de conectar al Arduino, se recomienda usar el software del fabricante para verificar que ambos módulos tengan la misma configuración (ej. Tasa de Baudios: 9600, Frecuencia: la misma banda).

B. Placa Transmisora (Placa 1)

- **Función:** Leer BMP280, empaquetar los datos y enviarlos por el APC220.
- **Conexiones:** BMP280 a pines I2C (A4/A5) y APC220 a pines Serial por Software. Usaremos **SoftwareSerial** para liberar el Pin 0 y 1 del Arduino.
- **APC220 TX** $\rightarrow$ Arduino **Pin 10 (RX Software)**

- **APC220 RX** $\rightarrow$ Arduino **Pin 11 (TX Software)**
  Código 4: Transmisor (BMP280 $\rightarrow$ APC220)
  Código Transmisor

```
#include <SoftwareSerial.h> // Para comunicación con el APC220
#include <Wire.h>
#include <Adafruit_BMP280.h>

// Pines para SoftwareSerial (APC220)
#define RX_APC 10
#define TX_APC 11
SoftwareSerial apc220(RX_APC, TX_APC); // RX, TX

// Sensor BMP280
Adafruit_BMP280 bmp;

void setup() {
  Serial.begin(9600); // Monitor Serie para depuración
  apc220.begin(9600); // Inicializa comunicación con APC220 a 9600 baudios

  // Inicialización del BMP280 (igual que en S2)
  if (!bmp.begin()) {
    Serial.println("Error BMP280. Deteniendo el Transmisor.");
    while(1);
  }
}

void loop() {
  float temperatura = bmp.readTemperature();
  float presion = bmp.readPressure() / 100.0F; // en hPa

  // Creamos la cadena de datos a enviar
  // Formato: T:XX.XX,P:YYYY.YY
  String dataString = "T:" + String(temperatura, 2) + ",P:" + String(presion, 2);

  // Enviamos la cadena por radio
  apc220.println(dataString);
  Serial.print("Enviando: ");
  Serial.println(dataString);
```

```
  delay(5000); // Envía datos cada 5 segundos
}
```

C. Placa Receptora (Placa 2)

- **Función:** Recibir la cadena de datos del APC220 y parsearla (separar los valores) para mostrarla en el Monitor Serie del PC.
- **Conexiones:** Usaremos el puerto Serial hardware del Arduino (pines 0/1) para el APC220 y el Monitor Serie del PC para visualizar la salida **(Esto requiere que el APC220 se desconecte temporalmente al cargar el código)**.
- **APC220 TX** $\rightarrow$ Arduino **Pin 0 (RX Hardware)**
- **APC220 RX** $\rightarrow$ Arduino **Pin 1 (TX Hardware)**
  NOTA IMPORTANTE: Para evitar conflictos, usaremos **SoftwareSerial** también en el Receptor, conectando el APC220 a los pines 2 y 3. Esto permite usar el Serial hardware (Monitor Serie) al mismo tiempo para la visualización.
- **APC220 TX** $\rightarrow$ Arduino **Pin 2 (RX Software)**
- **APC220 RX** $\rightarrow$ Arduino **Pin 3 (TX Software)**
  Código 5: Receptor (APC220 $\rightarrow$ Monitor Serie)
  Código Receptor

```
#include <SoftwareSerial.h> // Para comunicación con el APC220

// Pines para SoftwareSerial (APC220)
#define RX_APC 2
#define TX_APC 3
SoftwareSerial apc220(RX_APC, TX_APC); // RX, TX

void setup() {
  Serial.begin(9600);  // Monitor Serie (PC) a 9600 baudios
  apc220.begin(9600);  // Comunicación APC220 a 9600 baudios

  Serial.println("Receptor APC220 Iniciado. Esperando datos...");
}

void loop() {
  // Verificamos si hay datos disponibles del módulo APC220
  if (apc220.available()) {
    // Leemos la línea completa que fue enviada por el transmisor
    String data = apc220.readStringUntil('\n');
    data.trim(); // Limpiamos espacios y saltos de línea
```

```
    if (data.length() > 0) {
      // 1. Mostrar la cadena recibida
      Serial.print("Recibido: ");
      Serial.println(data);

      // 2. Parsear la cadena (Extraer T y P)
      // Buscar el inicio y fin de los valores
      int indexT = data.indexOf("T:") + 2;
      int indexP = data.indexOf(",P:");
      int indexEnd = data.lastIndexOf(' '); // Fin de la cadena

      if (indexT > 1 && indexP > indexT) {
        String tempStr = data.substring(indexT, indexP);
        String presStr = data.substring(indexP + 3);

        float temperatura = tempStr.toFloat();
        float presion = presStr.toFloat();

        // 3. Mostrar los valores parseados
        Serial.print(" -> Temp Procesada: ");
        Serial.print(temperatura);
        Serial.println(" *C");
        Serial.print(" -> Pres. Procesada: ");
        Serial.print(presion);
        Serial.println(" hPa");
        Serial.println("------------------------");
      }
    }
  }
  delay(100);
}
```