



Proyecto Arduino 2023

Desarrollo del trabajo practico integrador de la catedra “Teoria de Control 2023”. Implementacion arduino.

Docentes :

- ❖ José Luis Catalano
- ❖ Elvio Nicolas Oliva Saccenti

Integrantes Grupo :

- Botta Francisco
- Frighetto Franco
- Sánchez Facundo
- Giménez Tomas
- Gribaudo Facundo
- Botta Joaquin
- Carnino Martin

Resumen.....	2
Descripción del Proyecto.....	2
Objetivos Principales.....	2
Resultados destacados.....	2
Introducción.....	2
Contexto y justificación del Proyecto.....	2
Objetivo.....	2
Objetivos generales del proyecto.....	2
Objetivos específicos.....	3
Marco Teórico.....	3
Conceptos claves.....	3
Diseño y Metodología.....	3
Esquema de Conexionado.....	5
Desarrollo del Código.....	6
1. Declaración de Librerías y Variables.....	6
2. Configuración del Entorno.....	6
3. Bucle Principal.....	7
4. Control del LED según la Temperatura.....	7
5. Paquetes.....	8
Dificultades.....	9
Conclusiones.....	10

Resumen

Descripción del Proyecto

El Proyecto consiste del diseño e implementación de un sistema de control con tecnología de microcontroladores, y poder accionar acorde al control.

Objetivos Principales

Los objetivos principales planteados para este proyecto fueron:

- Realizar la implementación de un circuito utilizando microcontrolador, dispositivos de entrada y salida y de comunicación.
- Investigar y realizar las conexiones necesarias para lograr el funcionamiento deseado.

Resultados destacados

Logramos llevar a cabo la implementación de un ESP8266 realizando las conexiones pertinentes para lograr el funcionamiento de un sensor de temperatura (HW-503), capturar los datos provistos por el mismo y poder enviarlos mediante un paquete y recibir el mismo en una computadora personal, mediante WI-FI utilizando el protocolo HTTP (TCP).

Introducción

Contexto y justificación del Proyecto

Este informe aborda el diseño y la implementación de un sistema de control de temperatura utilizando un módulo ESP8266, que integra capacidades WiFi. El proyecto se centra en la medición de la temperatura ambiente y en la activación de un dispositivo de salida, en este caso, un LED, con una intensidad que varía según la temperatura medida.

En nuestro caso optamos por un sensor de temperatura HW-503 debido a que consideramos que el control de temperatura está presente en una amplia variedad de entornos, como por ejemplo sistemas de climatización y queríamos comprender cómo se implementan y cómo funcionan.

Objetivo

Objetivos generales del proyecto

Como objetivo general nos planteamos lograr que el circuito funcione en su totalidad, con todos los componentes integrados, de tal manera que nos permita conocer y controlar la temperatura de un sensor, simulando que se comporta como un aire acondicionado.

Objetivos específicos

Como objetivos específicos del proyecto planteamos:

- Lograr la conexión WI-FI del ESP
- Lograr la transmisión de paquetes mediante el protocolo HTTP
- Lograr el encendido de un LED bajo una condición determinada
- Lograr la implementación de PWM
- Implementar y capturar los datos de un sensor de temperatura
- Implementar un RELÉ

Marco Teórico

Conceptos claves

Destacamos como concepto clave el de **PWM**, este es el que nos permite llevar a cabo una analogía a aplicar de forma proporcional impulsos electrónicos para controlar la intensidad de nuestro led. Esto se realizará en base a una condición en particular, que en nuestro caso, es el rango de temperatura que captura el sensor.

PWM lleva a cabo el control de intensidad en base a valores que van desde 0 a 255 en forma progresiva

Las condiciones planteadas en nuestro caso son las siguientes:

```
if (tempCelcius < 29.0) {  
    analogWrite(Led_Pin, 0);  
} else if (tempCelcius >= 29.0 && tempCelcius < 30.0) {  
    analogWrite(Led_Pin, 50);  
} else if (tempCelcius >= 30.0 && tempCelcius < 31.0) {  
    analogWrite(Led_Pin, 150);  
} else if (tempCelcius >= 31.0) {  
    analogWrite(Led_Pin, 255);  
}
```

Diseño y Metodología

El sistema está diseñado para medir la temperatura ambiente utilizando un sensor analógico conectado a un Arduino y, posteriormente, controlar la intensidad de un LED en función de la temperatura medida. Además, se integra el módulo ESP8266 para habilitar la conectividad WiFi.

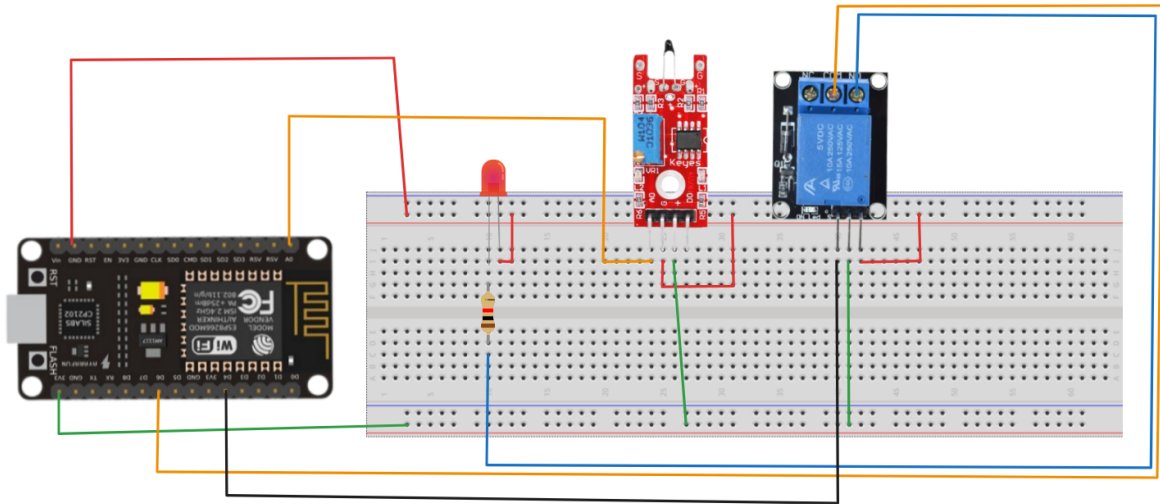
Descripción detallada del diseño del sistema.

- Medición de Temperatura:
 - El sensor de temperatura analógico está conectado a uno de los pines analógicos del Arduino.
 - El Arduino lee el valor analógico del sensor y lo convierte a una temperatura en grados Celsius mediante una ecuación específica.
- Control del LED:
 - La temperatura medida se utiliza para controlar la intensidad luminosa de un LED.
 - Se establecen diferentes niveles de intensidad luminosa del LED en base a rangos de temperatura predefinidos.
 - A temperaturas más altas, el LED emite una luz más intensa.
- Integración de WiFi (sin activar en el código actual):
 - El módulo ESP8266 se utiliza para habilitar la conectividad WiFi del sistema.
 - Esta funcionalidad podría permitir la transmisión de datos de temperatura a través de la red WiFi o la implementación de controles remotos.

Metodología utilizada en el desarrollo.

- Configuración de Hardware:
 - Conexión del sensor de temperatura al Arduino.
 - Configuración de pines y componentes en el entorno de desarrollo Arduino.
- Desarrollo del Código:
 - Programación en Arduino IDE para la lectura del sensor, conversión de valores analógicos a temperatura y control del LED en función de la temperatura medida.
 - Uso de la biblioteca "ESP8266WiFi.h" para la integración del módulo ESP8266 y la gestión de la conectividad WiFi (aunque no está activo en el código actual).
- Pruebas y Depuración:
 - Verificación del correcto funcionamiento del sistema.
 - Identificación y corrección de posibles errores en el código.
 - Ajustes en la ecuación de conversión de temperatura si es necesario para obtener mediciones más precisas.

Esquema de Conexionado



Componentes:

- microcontrolador: ESP8266
- Sensor: HW-503
- Relé: hw-482
- LED
- Protoboard
- Resistencia

Explicación:

Componentes como el relé y el sensor son alimentados por la línea de voltaje (Cable Verde), el Relé le envía la señal al led para que se prenda (Cable Azul) y este voltaje pasa por la resistencia y llega al LED.

Con respecto a las señales de entradas y salida, el sensor solo tiene conexión de entrada ya que es analógica (Cable Naranja), en cambio el relé posee dos conexiones digitales una controla el Relé (Cable Negro), y la otra controla el LED (Cable Naranja).

Desarrollo del Código

1. Declaración de Librerías y Variables

```
// librerías
#include "ESP8266WiFi.h"
#include <Wire.h>
#include <RTClib.h>

// Declaraciones de pines
int sensorAnalogico = A0;
int Led_Pin = 12;
int sensorRele = 2;

// Declaracion wifi
const char* ssid = "Sancheta";
const char* password = "20sanche02";
WiFiServer server(80);
RTC_DS3231 rtc;
```

- Se incluye la librería ESP8266WiFi.h para habilitar las funciones de Wi-Fi en el módulo ESP8266.
- Se declaran las variables para los pines analógicos, el LED, el relé y las credenciales del Wi-Fi.

2. Configuración del Entorno

```
void setup ()
{
    // Modos de los pines
    pinMode(sensorAnalogico, INPUT);
    pinMode(Led_Pin, OUTPUT);
    pinMode(sensorRele, OUTPUT);

    Serial.begin(9600);

    // Rele y pin activados
    digitalWrite(Led_Pin, HIGH);
    digitalWrite(sensorRele, HIGH);

    // Conexion WiFi
    WiFi.begin(ssid, password);

    // imprime hasta que conecta
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi conectado | ip: ");
    Serial.println(WiFi.localIP());
    Serial.println("Server ON");
}
```

- Se configuran los pines como entrada o salida según sea necesario.
- Se inician las comunicaciones seriales y se enciende el LED y el relé.
- Se conecta a WiFi

3. Bucle Principal

```
void loop ()
{
    float tempAnalogica;

    tempAnalogica = analogRead(sensorAnalogico);
    float tempF = 125.315-0.175529*tempAnalogica-2.5;
    float tempCelcius = ((tempF-32)/1.8)-5;
    Serial.print ("Valor temperatura: "); Serial.print (tempCelcius, 4); Serial.print (" C");
```

Se lee la señal analógica del sensor de temperatura y se convierte a grados Celsius, que luego usaremos para controlar el led. El loop va a contar con un retardo de 2.5 segundos para evitar una rápida secuencia de lecturas.

4. Control del LED según la Temperatura

```
if (tempCelcius<28.0 ){
    analogWrite(Led_Pin, 0);
    Serial.println (" Nivel PWM 1 ");
    sendTcp("Nivel PWM 1 - apagado - " + String(tempCelcius)+" C");
} else if (tempCelcius>=28.0 && tempCelcius<29.0 ){
    analogWrite(Led_Pin, 50);
    Serial.println (" Nivel PWM 2 ");

    sendTcp("Nivel PWM 2 - " + String(tempCelcius)+" C");
} else if (tempCelcius>=29.0 && tempCelcius<30.0 ){
    analogWrite(Led_Pin, 150);
    Serial.println (" Nivel PWM 3 ");

    sendTcp("Nivel PWM 3 - " + String(tempCelcius)+" C");
} else if (tempCelcius>=30.0){
    analogWrite(Led_Pin, 255);
    Serial.println (" Nivel PWM 4 ");

    sendTcp("Nivel PWM 4 - maximo - " + String(tempCelcius)+" C");
}
```

Ajustamos y variamos la intensidad del LED según el rango de temperaturas predefinido utilizando la técnica de PWM, para el cual definimos 4 niveles.

- Menor a 28° - apagado
- Entre 28° y 29°
- Entre 29° y 30°
- Mayor a 30° - máximo

A temperaturas más altas, el LED se ilumina más intensamente.

Por cada medición que hace, manda un paquete por la red.

5. Paquetes

















```
void sendTcp(String message) {  
    if (WiFi.status() == WL_CONNECTED) {  
        WiFiClient client;  
        // Trata de mandar el mensaje  
        // ip y puerto  
        if (client.connect("192.168.19.3", 8080)) {  
            Serial.println("Conexión establecida");  
            client.println(message);  
            client.println();  
            client.write("HolaMundo");  
        } else {  
            Serial.println("Fallo en la conexión");  
        }  
  
        // Espera y lee la respuesta del servidor  
        while (client.available()) {  
            Serial.println("Client Available");  
            char c = client.read();  
            Serial.print(c);  
        }  
  
        // Cierra la conexión  
        client.stop();  
    }  
}
```

Para mandar paquetes indicamos la ip de nuestra computadora y el puerto 8080 que lo definimos por defecto.

En el paquete vamos a mandar el nivel de PWM, la temperatura medida, y si es apagado o encendido al máximo en los niveles extremos.

Realizamos el envío con `client.println()`. Posteriormente cerramos la conexión.

Para la lectura de los paquetes, utilizamos el software Packet Sender, y se ve de la siguiente manera

Borrar registro(89)										Registrar Tráfico	Guardar Registro	Guardar Paquete de Tráfico	Copiar al Portapapeles
Tempo	IP Inicial	Puerto Inicial	Para Final	Puerto Final	Método	Erro	ASCII	Hex					
 19:54:41.452	192.168.19.145	50373	You	8080	TCP		Nivel PWM 1 - apagado - 27.41 C	4E 69 76 65 6C 20 50 57 4D 20 31 20 2D 20 61 70 61 67 61 64 6F 20 2D 20 32 37 2E 34 31 20 43					
 19:54:38.912	192.168.19.145	65208	You	8080	TCP		Nivel PWM 1 - apagado - 27.90 C	4E 69 76 65 6C 20 50 57 4D 20 31 20 2D 20 61 70 61 67 61 64 6F 20 2D 20 32 37 2E 39 30 20 43					
 19:54:36.042	192.168.19.145	64349	You	8080	TCP		Nivel PWM 2 - 28.97 C	4E 69 76 65 6C 20 50 57 4D 20 32 2D 20 32 38 2E 39 37 20 43					
 19:54:33.039	192.168.19.145	61668	You	8080	TCP		Nivel PWM 2 - 28.68 C	4E 69 76 65 6C 20 50 57 4D 20 32 2D 20 32 38 2E 36 38 20 43					
 19:54:30.511	192.168.19.145	49413	You	8080	TCP		Nivel PWM 2 - 28.88 C	4E 69 76 65 6C 20 50 57 4D 20 32 2D 20 32 38 2E 38 38 20 43					
 19:54:27.234	192.168.19.145	61778	You	8080	TCP		Nivel PWM 2 - 28.29 C	4E 69 76 65 6C 20 50 57 4D 20 32 2D 20 32 38 2E 32 39 20 43					
 19:54:23.753	192.168.19.145	63769	You	8080	TCP		Nivel PWM 3 - 29.75 C	4E 69 76 65 6C 20 50 57 4D 20 33 2D 20 32 39 2E 37 35 20 43					
 19:54:20.789	192.168.19.145	64972	You	8080	TCP		Nivel PWM 3 - 29.85 C	4E 69 76 65 6C 20 50 57 4D 20 33 2D 20 32 39 2E 38 35 20 43					
 19:54:18.224	192.168.19.145	56488	You	8080	TCP		Nivel PWM 2 - 28.68 C	4E 69 76 65 6C 20 50 57 4D 20 32 2D 20 32 38 2E 36 38 20 43					
 19:54:15.222	192.168.19.145	56475	You	8080	TCP		Nivel PWM 2 - 28.68 C	4E 69 76 65 6C 20 50 57 4D 20 32 2D 20 32 38 2E 36 38 20 43					
 19:54:12.694	192.168.19.145	58681	You	8080	TCP		Nivel PWM 2 - 28.39 C	4E 69 76 65 6C 20 50 57 4D 20 32 2D 20 32 38 2E 33 39 20 43					
 19:54:09.769	192.168.19.145	64356	You	8080	TCP		Nivel PWM 4 - maximo - 30.73 C	4E 69 76 65 6C 20 50 57 4D 20 34 2D 20 6D 61 78 69 6D 6F 20 2D 20 33 30 2E 37 33 20 43					
 19:54:07.242	192.168.19.145	51084	You	8080	TCP		Nivel PWM 1 - apagado - 27.80 C	4E 69 76 65 6C 20 50 57 4D 20 31 20 2D 20 61 70 61 67 61 64 6F 20 2D 20 32 37 2E 38 30 20 43					
 19:54:04.708	192.168.19.145	56740	You	8080	TCP		Nivel PWM 1 - apagado - 27.31 C	4E 69 76 65 6C 20 50 57 4D 20 31 20 2D 20 61 70 61 67 61 64 6F 20 2D 20 32 37 2E 33 31 20 43					
 19:54:01.636	192.168.19.145	52155	You	8080	TCP		Nivel PWM 1 - apagado - 27.31 C	4E 69 76 65 6C 20 50 57 4D 20 31 20 2D 20 61 70 61 67 61 64 6F 20 2D 20 32 37 2E 33 31 20 43					
 19:53:58.772	192.168.19.145	65138	You	8080	TCP		Nivel PWM 1 - apagado - 27.90 C	4E 69 76 65 6C 20 50 57 4D 20 31 20 2D 20 61 70 61 67 61 64 6F 20 2D 20 32 37 2E 39 30 20 43					
								UDP:54024	TCP:8080	SSL:61113	IPv4 Mod		

Recortando a únicamente los mensajes, se ve así:

```

Nivel PWM 3 - 29.85 C
Nivel PWM 2 - 28.68 C
Nivel PWM 2 - 28.68 C
Nivel PWM 2 - 28.39 C
Nivel PWM 4 - maximo - 30.73 C
Nivel PWM 1 - apagado - 27.80 C

```

Dificultades

Durante el desarrollo de este proyecto se nos presentaron una serie de inconvenientes, dentro de las cuales podemos destacar:

1. Bloque de firewall para la recepción de paquetes en nuestro equipo local. Este inconveniente fue solucionado desactivando momentáneamente el firewall del equipo.
2. Armado y conexión de todos los componentes del ESP en cuestión. Para solucionar este inconveniente primero simulamos la conexión en la plataforma "Tinkercad" asegurándonos que todo funcionaba correctamente y una vez hecho esto lo pasamos al modelo físico.
3. Inconveniente en el desarrollo del código con la lectura de los puertos al momento de intentar capturar los datos del sensor de temperatura ya que el ESP en cuestión los interpreta en un orden distinto. Para solucionar este inconveniente buscamos en internet el dataSet del sensor de temperatura y buscar las coincidencias de puertos que necesitábamos.
4. Inconveniente para determinar si el sensor de temperatura funcionaba a través de señales digitales o analógicas. Para solucionar este problema en un principio lo conectamos de forma digital ya que teníamos entendido que funcionaba de

cualquiera de las dos formas. Sin embargo de forma digital nos seguía presentando inconvenientes, con lo cual lo conectamos de forma analógica y pudimos solucionar los inconvenientes existentes.

Conclusiones

Consideramos que fue un proyecto interesante, con un nivel de complejidad menor al que esperábamos, sabiendo que partimos con conocimientos nulos en el armado de arduinos y todo lo relacionado al mismo. Si bien se nos presentaron ciertos inconvenientes no fue nada grave y pudimos solucionarlos rápido con ayuda de internet.

Sin embargo nos hubiera gustado lograr realizar la conexión con el backend y frontend, que por motivos de tiempo no llegamos, pero hubiera estado interesante entender cómo se realiza para dar un cierre definitivo y entender cómo construir un sistema completo teniendo como componente un controlador y hacer uso del mismo.