



**"El saber de mis hijos  
hará mi grandeza"**

**Universidad de Sonora**

**Ingenieria en Mecatronica**

**Departamento de Ingeniería Industrial**

## **PROGRAMACIÓN PARA INGENIEROS 2**

**Proyecto final:**

### **Integrantes:**

Campa Moran Ximena  
Vega Ramos Ivana  
Urias Sánchez Arnoldo Edmundo  
Amaya Munguía Angel Randu

### **Maestro:**

Jose Luis Aguilera

## **INTRODUCCIÓN**

Para este proyecto estuvimos aplicando los conocimientos adquiridos en las clases como primeramente fueron los algoritmos, luego empezar a codificar y al final estuvimos viendo sobre la programación orientada a objetos así que para aplicarlo estuvimos eligiendo hacer un programa que conectara una entrada conectada a arduino y las señales que esta reciba representarlas en Visual Studio utilizando las librerías de raylib y raygui que fueron las que el maestro nos estuvo asignando para hacer las tareas y trabajos en clase, el sensor que se utilizó en este caso fue un sensor de temperatura en particular fue el DTH11 que mandara señales al arduino el cual se conecta al Visual Studio, también un botón el cual activa el sistema y prende un LED verde en la protoboard y otro rojo si no está activo, esto igualmente sucede en el Visual Studio cumpliendo la misma función.

## MARCO TEÓRICO

### ANTECEDENTES:

La programación orientada a objetos se remonta a la década de los 60's con el lenguaje Simula 67 en Noruega con conceptos como clases, objetos y herencia los cuales siguen siendo utilizados hasta la fecha. Después en los años 70's llegó el lenguaje Smalltalk el cual refinó y popularizó más los conceptos de clases y objetos. Para la década de los 80's Bjarne Stroustrup creó el lenguaje de programación C++ como una extensión del lenguaje C en el cual incorporó ideas de la POO (Programación Orientada a Objetos) de una manera más accesible para un público más amplio, lo que le ayudó a que muchas más personas sin un conocimiento tan profesional pudieran adquirirlo. Posteriormente para la década de los 90's Java, principalmente, se convirtió en un lenguaje muy popular y un gran impulsor de la POO, especialmente ya en internet. Otros que se consolidaron en este momento fueron Python y Ruby. Por último para la década de los 2000 los lenguajes como C# Microsoft y la versión 5 de PHP incorporaron plenamente los principios de la POO.

### DEFINICIONES CLAVE:

**La programación orientada a objetos (POO)** es un modelo de desarrollo de software que organiza el código en torno a objetos, que combinan datos (atributos) y funciones (métodos) en una sola entidad. Su objetivo es crear software más reutilizable, adaptable y fácil de mantener.

**Clases:** Son las plantillas o moldes a partir de los cuales se crean los objetos.

**Objetos:** Son instancias específicas de una clase.

**Herencia:** Permite que una clase hereda atributos y métodos de otra clase, promoviendo la reutilización del código.

**Encapsulamiento:** Consiste en agrupar los datos y los métodos en una sola unidad y controlar el acceso a los atributos, de modo que solo se puedan modificar a través de sus propios métodos públicos.

**Polimorfismo:** Es que los objetos de diferentes clases pueden responder al mismo mensaje de manera diferente, adaptándose a la lógica específica de cada clase.

### BENEFICIOS Y APLICACIONES:

La programación orientada a objetos (POO) ofrece a los estudiantes beneficios como la modularidad, la reutilización del código y la resolución de problemas de manera más organizada, al modelar el mundo real a través de objetos.

**Modularidad y organización:** Descompone los programas en objetos independientes lo que facilita la comprensión, el desarrollo y la depuración individual de cada parte del código.

**Resolución de problemas:** Ayuda a pensar de manera más estructurada, relacionando los conceptos del mundo real con entidades abstractas, lo que mejora las habilidades de diseño y resolución de problemas.

**Mantenibilidad:** Facilita la actualización y el mantenimiento del código, ya que los cambios en un objeto tienen menos probabilidades de afectar al resto del programa.

**Para las aplicaciones están:**

**Desarrollo de videojuegos:** Es fundamental para crear motores de juegos, IA y la lógica de los personajes utilizando objetos que representan entidades del juego.

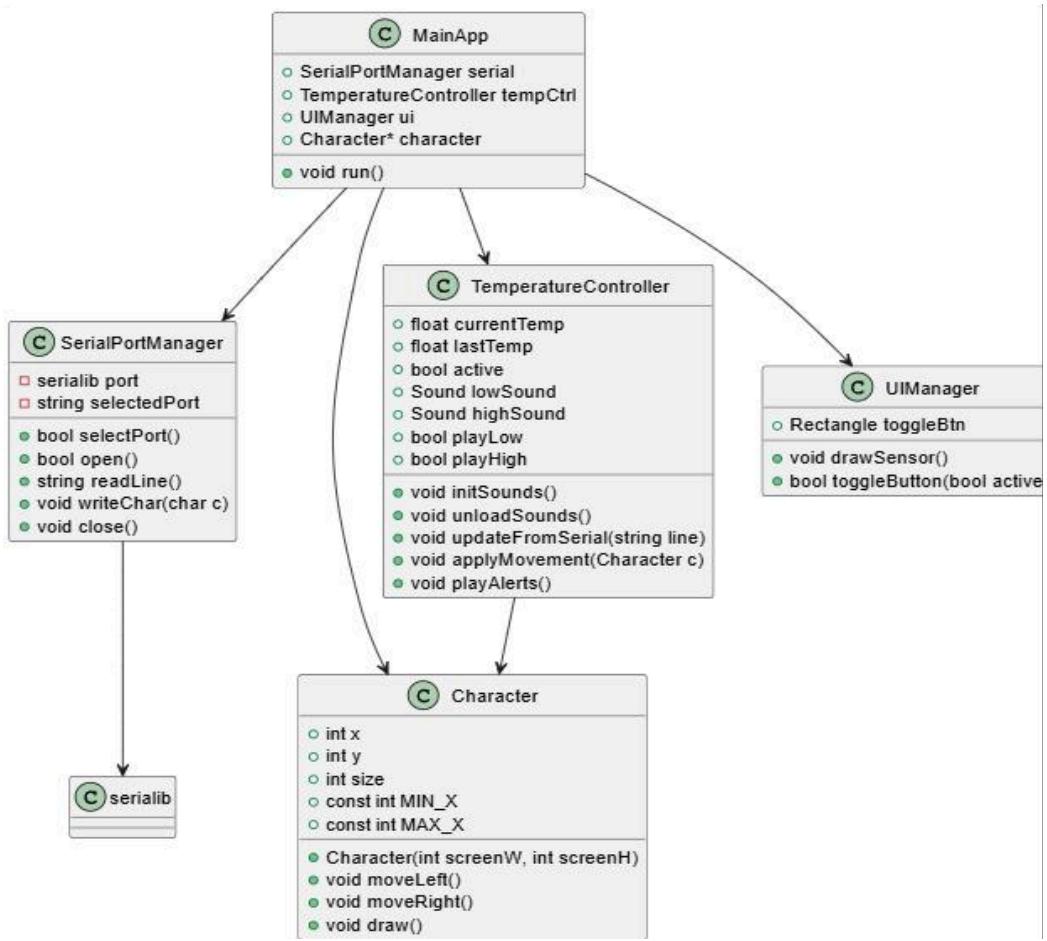
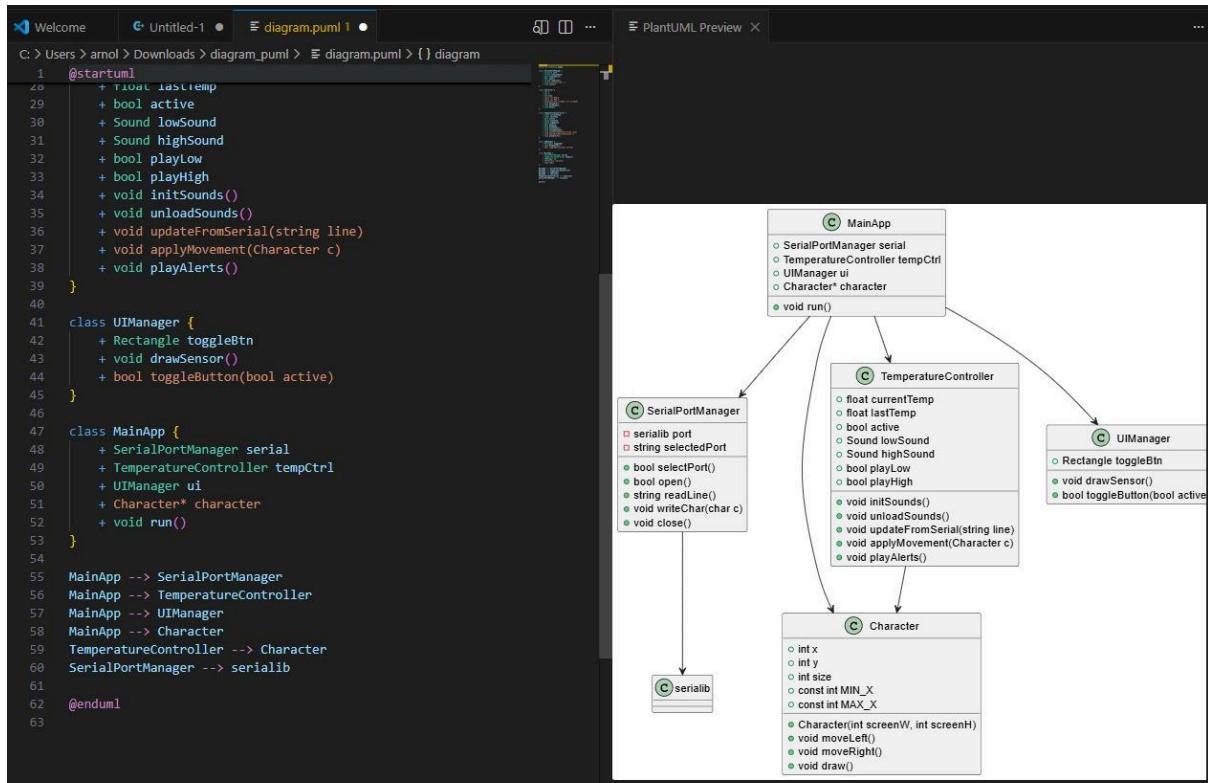
**Aplicaciones móviles:** Permite construir aplicaciones robustas y escalables para dispositivos móviles, manejando la lógica de la aplicación y la interfaz de usuario a través de objetos.

**Simulaciones:** Ideal para modelar sistemas del mundo real, como simuladores de tráfico, simulaciones científicas o sistemas de gestión de inventario, que se benefician de la representación de entidades a través de objetos.

**Desarrollo web:** Es la base de muchos frameworks web modernos, permitiendo a los estudiantes construir aplicaciones web dinámicas y con estructura más limpia y mantenible.

**Conclusión:** Por lo último esto nos deja que la programación orientada a objetos es muy importante desde décadas pasadas pero la última década esto tomó mucha más fuerza y hoy en dia es algo fundamental y gracias a los avances de esta y la facilidad que se le ha dado a su uso, es mucho más accesible para cualquier persona y ya no es necesario estar en algún área que se especialice en esto, lo cual es muy bueno ya que hace que se tengan más oportunidades.

## DIAGRAMA DE CLASE



## DESARROLLO

El proyecto tiene como objetivo comunicar dos formas de lenguaje: Arduino y C++. Para ello, contamos primero con un circuito físico que incluye un sensor de temperatura, un botón y dos LEDs (uno rojo y uno verde), todos conectados a una protoboard y, desde ahí, a los pines del Arduino.

En Visual Studio (C++) se han instalado las librerías necesarias para leer el puerto serial que envía el Arduino, así como las librerías utilizadas para dibujar los elementos visuales del programa: raygui, raylib y serialib.

El funcionamiento del proyecto inicia mostrando en pantalla un menú que permite al usuario seleccionar entre varias entradas COM. Una vez elegida la entrada correspondiente al sistema, se despliega una segunda pantalla que muestra los dibujos implementados: Bob Esponja y nuestro sensor de temperatura.

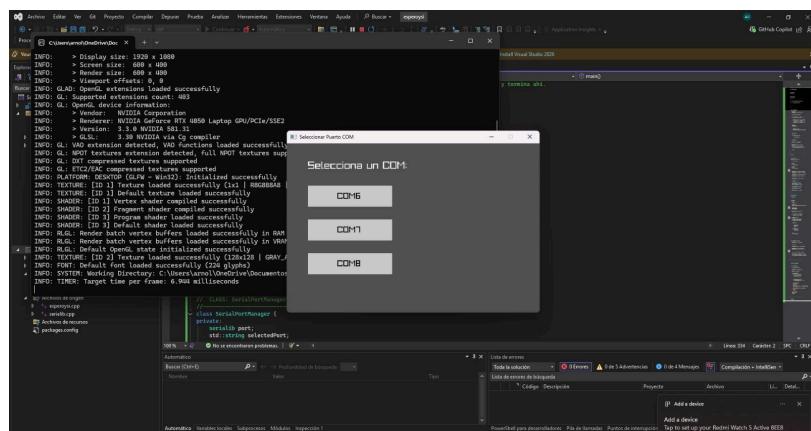
Al presionar el botón físico, el programa se activa y, simultáneamente, se actualiza el estado de un botón digital dentro de la interfaz, el cual cumple la misma función. Esto significa que al presionar el botón se enciende el LED verde (indicando que el sistema está encendido) y se apaga el rojo. Si se vuelve a presionar, el LED verde se apaga, el rojo se enciende y el programa se detiene.

Una vez iniciado el programa, la pantalla muestra la temperatura actual, que en promedio es de aproximadamente 32 °C. Al aumentar la temperatura cerca del sensor (con ayuda de un encendedor), el dibujo comienza a desplazarse hacia la derecha, indicando el incremento. Cuando esta temperatura supera un límite preestablecido dentro del programa, se activa una alerta auditiva que indica que se ha sobrepasado el límite superior.

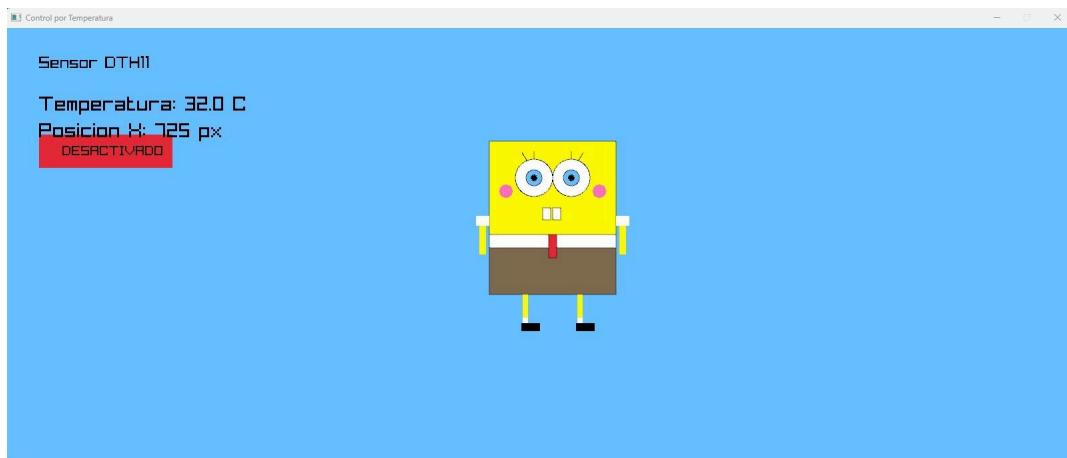
Por el contrario, al disminuir la temperatura (en nuestro caso utilizando hielo), el valor mostrado comienza a bajar y el dibujo se desplaza hacia la izquierda. Si se rebasa el límite inferior establecido, se activa otra alarma distinta a la del límite superior.

Finalmente, al presionar nuevamente el botón físico se desactiva el sistema, haciendo que el Arduino deje de recopilar datos y dando por concluido el funcionamiento del proyecto.

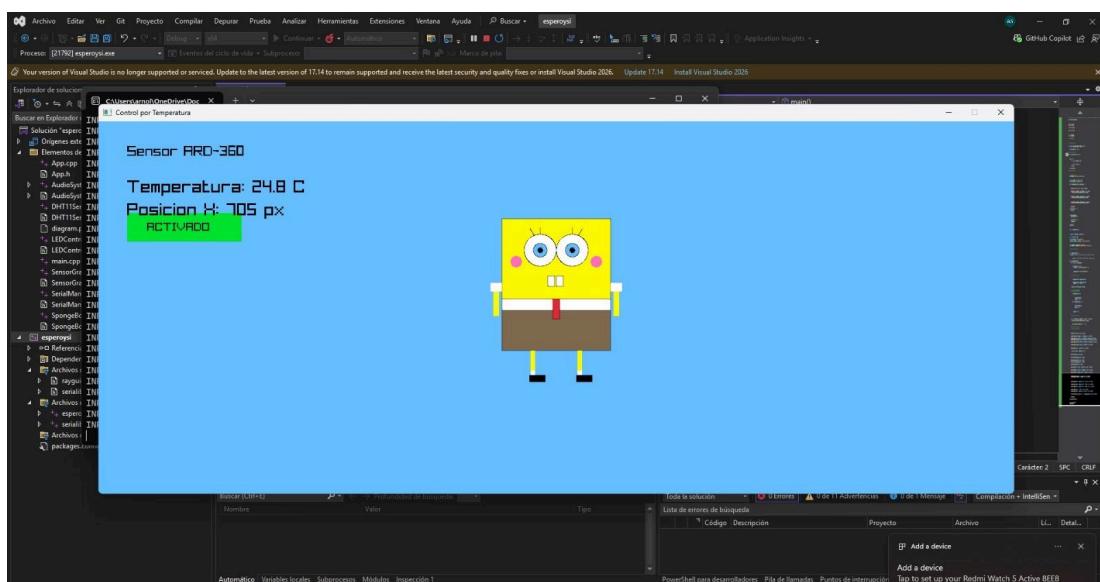
## Seleccionar entrada COM



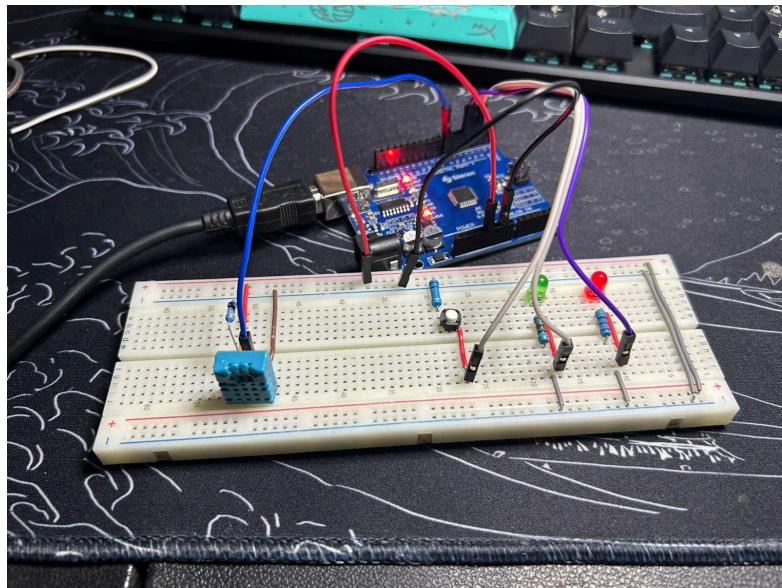
## Programa desactivado



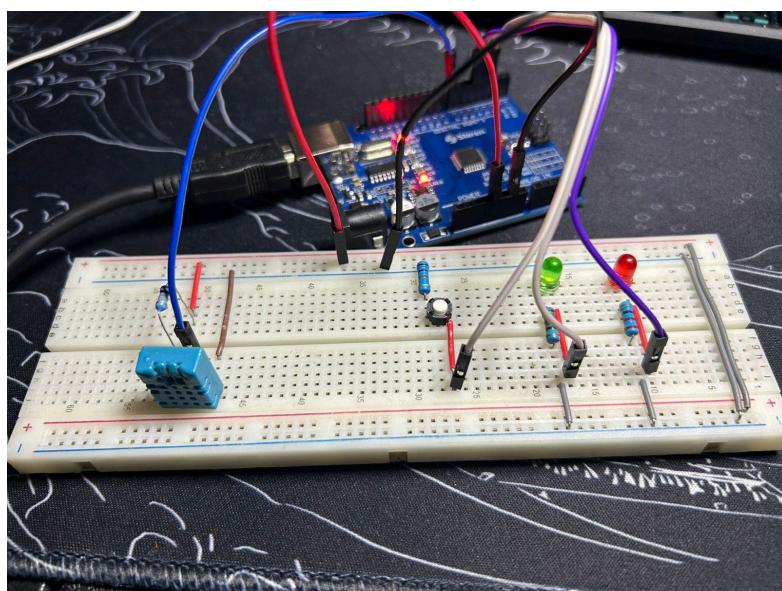
## Programa activado



**Círcuito físico desactivado**



**Círcuito físico activado**



## **CONCLUSIONES**

El desarrollo de este proyecto nos permitió aplicar de manera práctica los conocimientos vistos durante el semestre, especialmente los relacionados con algoritmos, programación en C++ y programación orientada a objetos. Pudimos integrar hardware y software al conectar un sistema físico con Arduino a una interfaz gráfica en Visual Studio usando raylib, raygui y serialib. Esto nos ayudó a entender mejor cómo se comunican diferentes lenguajes y plataformas.

También, al trabajar con el sensor, el botón y las señales seriales, comprendimos la importancia de leer datos en tiempo real y mostrar esa información de una forma clara y visual. El funcionamiento del sensor de temperatura y las alertas nos mostró cómo los conceptos de lógica, objetos y eventos pueden usarse para crear un sistema funcional.

En general, este proyecto fortaleció nuestras habilidades técnicas y también nos enseñó a resolver problemas, trabajar en equipo y combinar electrónica con programación y diseño visual. Fue una experiencia completa que refuerza lo aprendido y nos prepara mejor para proyectos más avanzados.

## BIBLIOGRAFÍA

Luis Llamas. (s. f.). *Breve historia de la Programación Orientada a Objetos*.

[https://www.luisllamas.es/historia-programacion-orientada-objetos/ Luis Llamas](https://www.luisllamas.es/historia-programacion-orientada-objetos/)

Gómez, F. J. (s. f.). *Antecedentes de la Programación Orientada a Objetos*. Scribd.

[https://es.scribd.com/document/237708929/Antecedentes-de-La-Programacion-Orientada-a-Objetos Scribd](https://es.scribd.com/document/237708929/Antecedentes-de-La-Programacion-Orientada-a-Objetos_Scribd)

Profile.es. (s. f.). *¿Qué es la programación orientada a objetos?*

[https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/ Profile Software Services](https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/)

IBM. (s. f.). *Object-Oriented Programming*. IBM Documentation.

[https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming\\_IBM](https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming_IBM)

98thpercentile.com. (s. f.). *Basics of Object-Oriented Programming: Core Concepts and Benefits*. <https://www.98thpercentile.com/blog/basics-of-object-oriented-programming/>