INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

DIVISIÓN:

INGENIERÍA MECATRÓNICA

MATERIA:

SISTEMAS EMBEBIDOS BASADOS EN PROCESAMIENTO DIGITAL DE SEÑALES

DOCENTE:

JOSE ANGEL NIEVES VAZQUEZ

UNIDAD 4:

SISTEMAS BASADOS EN PROCESAMIENTO DIGITAL DE SEÑALES

INVESTIGACION DE LA UNIDAD 4

ALUMNOS:

ISRAEL ANTONIO LÓPEZ ESCRIBANO
JESUS ALEJANDRO ROSAS ROSAS
ARGELIO SANTIAGO REYES

GRUPO:

711-B

FECHA DE ENTREGA:

San Andrés Tuxtla Ver. 01 de Diciembre del 2025

Introducción:

Por medio de esta investigación documental abordaremos temas relacionados con “sistemas basados en procesamientos digitales de señales”, debido a que estos temas son de vital importancia al hablar señales digitales, en esta recopilación documental veremos cómo se encuentran estructuradas la arquitectura de un sistema DSP ya que esta tecnología permite controlar y purificar las señales de un entorno digital, observaremos como se usa para que sirve, como puede ser aplicado, entre otras cosas.

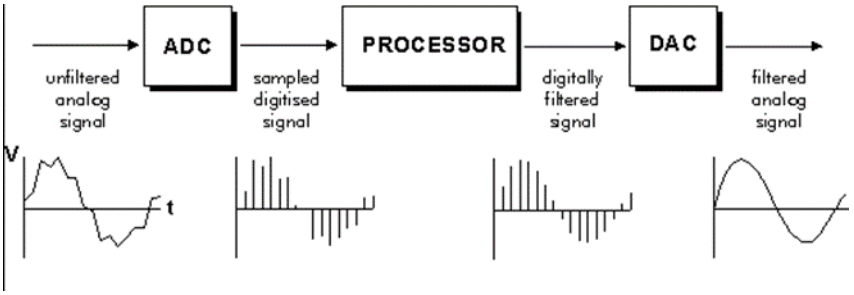
Además, hay que considerar que para que esto funcione debe de requerir de ciertas instrucciones en el DSP, también un factor importante es considerar las herramientas para su programación, los DSP pueden programarse en dos niveles: a bajo nivel, mediante lenguaje ensamblador, o a un nivel más alto, como C o C++.

Y finalmente veremos el desarrollo de las aplicaciones, El desarrollo de aplicaciones basadas en DSP es un proceso que combina teoría de procesamiento de señales, diseño de algoritmos y conocimiento del hardware. En la se utilizan herramientas como MATLAB, Python o Simulink para diseñar los algoritmos necesarios, para conocer más de este tema tan importante procedemos a mostrar la información contenida en esta investigación documental.

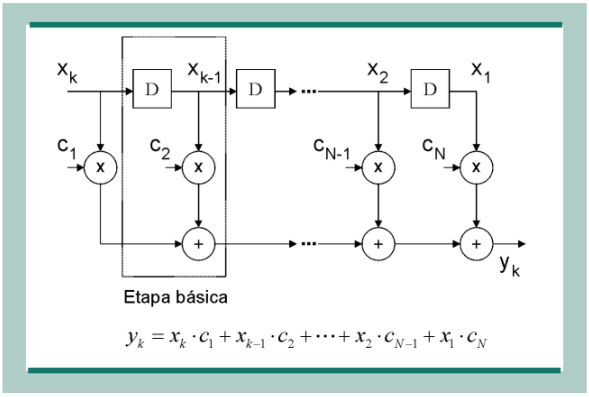
4.1 Introducción a arquitectura de un sistema DSP.

Los procesadores de señales digitales (DSP) toman diversas señales de voz, video, audio, presión, posición o temperatura que se digitalizan y luego las manipulan matemáticamente. En otras palabras, el DSP es el manejo y procesamiento de señales digitales para lograr un resultado específico. Estas señales suelen representar una amplia gama de datos, incluyendo imágenes, videos, audio y más. El DSP es una excelente tecnología que permite controlar y purificar estas señales en el entorno digital, lo que permite utilizarlas en diversas aplicaciones.

Sin embargo, para comprender la importancia del DSP, es necesario distinguir entre señales digitales y analógicas. Las señales analógicas son constantes y pueden tener cualquier valor dentro del rango (0 mA a 20 mA), lo que las hace ampliamente utilizables. Por el contrario, las señales digitales son silenciosas y constan de diferentes valores. El DSP permite transformar señales analógicas en digitales y controlarlas, ofreciendo mayor flexibilidad y precisión en el procesamiento de señales.



Un sistema de procesado digital de señal puede definirse como cualquier sistema electrónico que realice procesado digital de señal, entendiéndose por él la aplicación de operaciones matemáticas a señales representadas de forma digital. Las señales son representadas de forma digital mediante secuencias de muestras. A menudo, estas muestras se obtienen de señales físicas (por ejemplo, señales de audio) utilizando transductores (un micrófono en este caso) y convertidores analógico-digitales. Después del procesado matemático, las señales digitales pueden volver a convertirse en señales físicas mediante convertidores digital-analógicos.

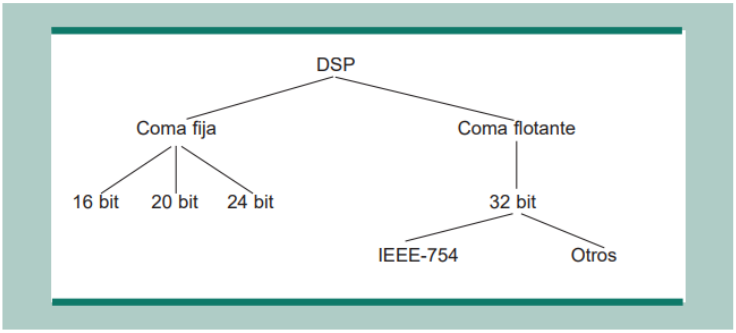


Estructura de un filtro de respuesta impulsional finita (FIR)

Si bien, en principio, el corazón de un sistema de procesado digital puede ser un microcontrolador, un procesador de propósito general o un procesador digital de señal (DSP), en sistemas en los cuales la carga computacional es extremadamente intensa la solución óptima pasa por escoger a un DSP. En la actualidad, los cuatro grandes fabricantes de DSP son Texas Instruments, con la serie TMS320; Motorola, con las series DSP56000, DSP56100, DSP56300, DSP56600 y DSP96000; Lucent Technologies (anteriormente AT&T), con las series DSP1600 y DSP3200; y Analog Devices, con las series ADSP2100 y ADSP21000.

¿Qué es un DSP?

Estrictamente hablando, el término DSP se aplica a cualquier chip que trabaje con señales representadas de forma digital. En la práctica, el término se refiere a microprocesadores específicamente diseñados para realizar procesado digital de señal. Los DSP utilizan arquitecturas especiales para acelerar los cálculos matemáticos intensos implicados en la mayoría de sistemas de procesado de señal en tiempo real. Por ejemplo, las arquitecturas de los DSP incluyen circuitería para ejecutar de forma rápida operaciones de multiplicar y acumular, conocidas como MAC. A menudo poseen arquitecturas de memoria que permiten un acceso múltiple para permitir de forma simultánea cargar varios operandos, por ejemplo, una muestra de la señal de entrada y el coeficiente de un filtro simultáneamente en paralelo con la carga de la instrucción. También incluyen una variedad de modos especiales de direccionamiento y características de control de flujo de programa diseñadas para acelerar la ejecución de operaciones repetitivas. Además, la mayoría de los DSP incluyen en el propio chip periféricos especiales e interfaces de entrada salida que permiten que el procesador se comunique eficientemente con el resto de componentes del sistema, tales como convertidores analógico-digitales o memoria.



Representaciones numéricas comunes en los DSP comerciales

La diferencia esencial entre un DSP y un microprocesador es que el DSP tiene características diseñadas para soportar tareas de altas prestaciones, repetitivas y numéricamente intensas. Por contra, los microprocesadores de propósito general o microcontroladores no están especializados para ninguna aplicación en especial; en el caso de los microprocesadores de propósito general, ni están orientados a aplicaciones de control, en el caso de los microcontroladores. Aunque el ejemplo del filtro de respuesta impulsional finita (FIR) ha sido ampliamente utilizado en el entorno DSP, es quizás el más simple que permite ilustrar

la necesidad de estas prestaciones en los DSP, las cuales permiten concebir muchas de las funciones de procesado en tiempo real.

Un DSP para cada Aplicación:

Una forma de clasificar los DSP's y aplicaciones es a través de su rango dinámico. El rango dinámico es un conjunto de números, desde pequeños a grandes, que deben ser procesados en el curso de una aplicación. Por ejemplo, para representar una forma de onda entera de una señal particular es necesario un cierto rango de números para manejar sus valores mayores y menores.

El DSP debe ser capaz de manejar los números generados tanto en la transformación análoga – digital como durante los cálculos (multiplicaciones, sumas, divisiones) con dicha señal. Si no es capaz de manejar todo el rango de números ocurrirá "overflow" o "underflow", lo cual producirá errores en los cálculos. La capacidad del procesador es una función de su ancho de datos (el número de bits manipulados) y el tipo de aritmética que posee (punto fijo o flotante). Un procesador de 32 bits tiene un ancho de datos mayor que uno de 24 bits, el cual a su vez tiene un rango mayor que uno de 16 bits. DSP's de punto flotante tienen rangos mayores que uno de punto fijo. Cada tipo de procesador es ideal para un rango particular de aplicaciones. DSP's de 16 bits son ideales para sistemas de voz tales como teléfonos ya que ellos trabajan con un estrecho rango de frecuencias de audio. Stereos de alta fidelidad requieren ADCs de 16 bits y un procesador de 24 bits de punto fijo.

Los 16 bits del conversor permiten capturar todo el rango de la señal de audio y los 24 bits del procesador permiten operar cómodamente los grandes valores resultantes de la operación con los datos. Procesamiento de imágenes, gráficos 3–D y simulaciones científicas necesitan un rango dinámico mucho mayor y por lo tanto requieren procesadores de punto flotante de 32 bits y ADC's de 24 bits.

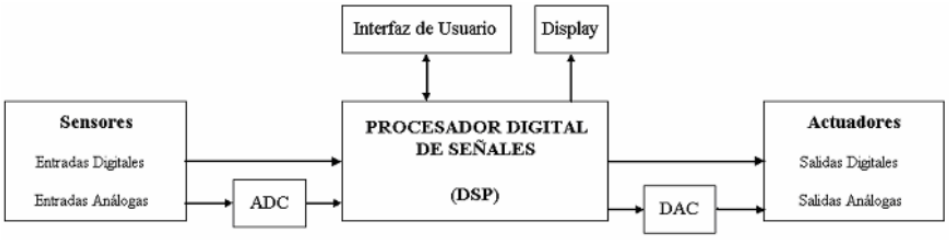


Figura 1: Diagrama de bloques conceptual de un sistema típico
ADC : Conversor Análogo Digital. – **DSP : D**igital **S**ignal **P**rocessor.
DAC : Conversor Digital Análogo.

Algunos ejemplos de los usos de DSP's en la actualidad son:

1. Wireless LAN
2. Reconocimiento de Voz
3. Manejo de imágenes digitales
4. Reproductores digitales de audio
5. Teléfonos celulares
6. Modems inalámbricos

7. Cámaras digitales
8. Control de motores
9. Manejo de bombas, ventiladores, HVAC
10. Inversores industriales
11.
12. Automatización de fábricas
13. Transporte

VENTAJAS Y DESVENTAJAS DE LOS DSP'S

• La Tecnología VLSI (Very Large Scale Integration) da la posibilidad de diseñar sistemas con la capacidad para ejecutar procesamiento en tiempo real de muchas de las señales de interés para aplicaciones en comunicaciones, control, procesamiento de imagen, multimedia, etc.

• Los sistemas digitales son más confiables que los correspondientes sistemas análogos. • Los sistemas digitales ofrecen una mayor flexibilidad que los correspondientes sistemas análogos.

• Mayor precisión y mayor exactitud pueden ser obtenidas con sistemas digitales, comparado con los correspondientes sistemas análogos. • Un sistema programable permite flexibilidad en la reconfiguración de aplicaciones DSP.

• La tolerancia de los componentes en un sistema análogo hacen que esto sea una dificultad para el diseñador al controlar la exactitud de la señal de salida análoga. Por otro lado, la exactitud de la señal de salida para un sistema digital es predecible y controlable por el tipo de aritmética usada y el número de bits usado en los cálculos.

• Las señales digitales pueden ser almacenadas en un disco flexible, Disco Duro o CD–ROM, sin la pérdida de fidelidad más allá que el introducido por el conversor Análogo Digital (ADC). Éste no es el caso para las señales análogas.

A pesar de ellas existen algunos inconvenientes que deberán ser tomados en cuenta al momento de escoger una plataforma para el procesamiento de señales analógicas por medio digitales:

• La conversión de una señal analógica en digital, obtenida muestreando la señal y cuantificando las muestras, produce una distorsión que nos impide la reconstrucción de la señal analógica original a partir de muestras cuantificadas.

• Existen efectos debidos a la precisión finita que deben ser considerados en el procesado digital de las muestras cuantificadas.

• Para muchas señales de gran ancho de banda, se requiere procesado en tiempo real. Para tales señales, el procesado analógico, o incluso óptico, son las únicas soluciones válidas. Sin embargo, cuando los circuitos digitales existen y son de suficiente velocidad se hacen preferibles.

4.2 Instrucciones de un DSP.

Las instrucciones de un DSP (Procesador de Señales Digitales) se refieren tanto a las operaciones de bajo nivel que realiza el hardware como a la configuración de los parámetros de audio para una función específica. A nivel de hardware, las instrucciones son para operaciones como multiplicar y acumular (MAC), que son fundamentales para procesar señales. A nivel de usuario, las instrucciones implican la configuración del software del DSP para controlar aspectos como la ecualización, la ganancia, los retardos, la asignación de canales y la supresión de ruido, mediante una interfaz como una aplicación o un puerto USB.

Instrucciones MAC (Multiply and Accumulate)

Son las más representativas de la arquitectura DSP. Permiten realizar una multiplicación y posteriormente acumular ese resultado en un registro. Esta operación es fundamental en algoritmos como filtros FIR o IIR, convoluciones o transformadas.

Instrucciones de desplazamiento (Shift)

Se utilizan para escalar señales, normalizar datos o realizar operaciones rápidas relacionadas con potencias de dos. Estas instrucciones son esenciales para evitar saturación o pérdida de precisión.

Instrucciones en paralelo

Muchos DSP permiten ejecutar varias operaciones en un solo ciclo de reloj. Esto incrementa el rendimiento sin aumentar la frecuencia del procesador, permitiendo cálculos altamente eficientes.

Instrucciones de saturación

Los DSP incorporan instrucciones que evitan que los resultados sobrepasen los rangos permitidos. Esto es muy importante en señales de audio o control, donde un desbordamiento podría generar distorsiones.

Instrucciones de acceso optimizado a memoria

Permiten manejar buffers circulares, estructuras de datos usadas en filtrado digital y algoritmos FFT. Estas instrucciones reducen el número de ciclos necesarios para cargar o almacenar datos.

4.3 Herramientas de programación.

El desarrollo de software para un DSP requiere herramientas adecuadas que permitan diseñar, simular, programar, depurar y optimizar algoritmos de procesamiento de señales. A diferencia de un microcontrolador convencional, donde la programación suele ser más directa, en un DSP es importante considerar el rendimiento, el número de ciclos que consume cada instrucción y la forma en que se manejan los datos. Por esta razón, las herramientas de programación juegan un papel esencial para lograr que el sistema funcione en tiempo real.

Generalmente, los DSP pueden programarse en dos niveles: a bajo nivel, mediante lenguaje ensamblador, o a un nivel más alto, como C o C++. El ensamblador ofrece un control muy preciso sobre el hardware, permitiendo aprovechar al máximo instrucciones especializadas como las operaciones MAC o el paralelismo interno. Sin embargo, programar de esta forma puede resultar complejo y lento, por lo que normalmente solo se utiliza para optimizar partes críticas del código. Por otro lado, lenguajes como C y C++ permiten escribir programas de manera más clara y estructurada, manteniendo buenas velocidades gracias a los compiladores optimizados diseñados específicamente para DSP.

Para programar un DSP, se utilizan entornos de desarrollo integrados (IDE), que incluyen compiladores, simuladores, depuradores y herramientas gráficas. Un ejemplo muy conocido es Code Composer Studio, utilizado para los DSP de Texas Instruments. Este entorno permite escribir el código, compilarlo, ejecutarlo en un simulador o cargarlo al hardware real, además de analizar el comportamiento del programa en tiempo real. De manera similar, Analog Devices ofrece herramientas como VisualDSP++ o CrossCore Embedded Studio, que integran módulos de análisis, manejo de memoria y visualización de señales.

Otra herramienta fundamental en el diseño de sistemas DSP es MATLAB o Simulink. Estas plataformas permiten crear y simular algoritmos complejos sin necesidad de programarlos directamente en el DSP desde el inicio. Con ellas es posible diseñar filtros, hacer pruebas, generar gráficas, comprobar el comportamiento de la señal y ajustar parámetros en cuestión de segundos. Posteriormente, estos algoritmos se pueden convertir a código C o incluso generar automáticamente código optimizado para DSP.

Además de los IDE y los simuladores, existen bibliotecas optimizadas que contienen funciones matemáticas ya preparadas para operar de manera eficiente en hardware DSP. Estas bibliotecas simplifican enormemente la programación, ya que en lugar de escribir desde cero un algoritmo de filtrado, FFT o convolución, el programador puede usar una función preexistente diseñada para ejecutarse en pocos ciclos. Un ejemplo es la librería CMSIS-DSP para procesadores ARM o las bibliotecas de Texas Instruments para sus familias C2000 y C6000.

4.4 Desarrollo de aplicaciones.

El desarrollo de aplicaciones basadas en DSP es un proceso que combina teoría de procesamiento de señales, diseño de algoritmos y conocimiento del hardware. A diferencia de programar un sistema convencional, trabajar con un DSP implica considerar la rapidez con la que se deben procesar los datos, la latencia permitida y el comportamiento de la señal en tiempo real. Por ello, el flujo de trabajo para crear una aplicación DSP es más estructurado y requiere diversas etapas que garantizan que el resultado cumpla con las exigencias funcionales.

El primer paso consiste en definir claramente el problema. Es necesario identificar qué tipo de señal se va a procesar, qué características tiene y cuál es el objetivo final. Por ejemplo, si se trata de audio, puede ser necesario eliminar ruido, ecualizar o comprimir la señal; si es una imagen, tal vez se necesite mejorar el contraste o detectar bordes. En esta fase también se

determinan los requisitos técnicos como la frecuencia de muestreo, la resolución, la velocidad del DSP y los límites de tiempo real del sistema.

Una vez que el problema está bien planteado, se pasa a la etapa de modelado y simulación. Aquí se utilizan herramientas como MATLAB, Python o Simulink para diseñar los algoritmos necesarios. Esta etapa es clave porque permite analizar cómo se comporta la señal sin necesidad de utilizar el DSP físico. El modelo matemático se puede probar con diferentes parámetros, ajustar filtros, realizar transformadas o simular condiciones extremas. Esto acelera el desarrollo, reduce errores y ayuda a comprender mejor el comportamiento del sistema.

Cuando el algoritmo está validado teóricamente, se procede a la implementación en el DSP. La programación suele hacerse en C, aunque en partes críticas puede utilizarse ensamblador para obtener un rendimiento máximo. Durante esta fase se deben considerar aspectos como el uso eficiente de memoria, la estructura de datos, el número de ciclos que consume cada operación y la forma en que el DSP accede a los datos. El objetivo es transformar el modelo matemático inicial en un código práctico, funcional y listo para ejecutarse en tiempo real.

La optimización es una etapa fundamental. No basta con que el algoritmo funcione: debe hacerlo con la velocidad suficiente. Aquí se aplican técnicas como el uso de instrucciones MAC, aprovechamiento del paralelismo interno, reducción de operaciones pesadas y reorganización de datos para minimizar accesos a memoria lenta. También se implementan buffers circulares para manejar flujos continuos de datos y mantener estabilidad en la ejecución.

Después de la optimización, el código se prueba directamente en el hardware. El DSP se conecta a los sensores, actuadores o fuentes de señal reales y se analiza su desempeño. Se verifica si el sistema responde con la velocidad requerida, si la latencia es adecuada y si los resultados coinciden con lo obtenido en la simulación. En esta fase se utilizan depuradores, analizadores lógicos y herramientas del IDE para observar señales internas, variables y tiempos de ejecución.

Finalmente, se realiza la validación en tiempo real. El sistema se prueba bajo condiciones reales de funcionamiento para asegurarse de que mantiene estabilidad, precisión y rendimiento. Aquí se evalúa cómo responde ante variaciones inesperadas de la señal, ruidos, cambios de carga y condiciones dinámicas. Solo después de superar esta etapa se considera que la aplicación DSP está lista para implementarse.

Conclusión:

Gracias a esta valiosa información recopilada logramos comprender como funciona la arquitectura de un sistema DSP, en el estudio de señales, y tomando este punto como inicio de partida observamos que estos análisis de sistemas cuentan con instrucciones que, a nivel de hardware, las instrucciones son para operaciones como multiplicar y acumular (MAC), que son fundamentales para procesar señales. A nivel de usuario, las instrucciones implican la configuración del software del DSP para controlar aspectos como la ecualización, la ganancia, los retardos, la asignación de canales y la supresión de ruido, mediante una interfaz como una aplicación o un puerto USB.

Por otro lado, también entendimos que este tipo de sistemas requieren de herramientas de programación, los DSP pueden programarse en dos niveles: a bajo nivel, mediante lenguaje ensamblador, o a un nivel más alto, como C o C++. Y finalmente gracias a estas herramientas de programación podemos realizar el desarrollo de aplicaciones. Una vez que el problema está bien planteado, se pasa a la etapa de modelado y simulación. Aquí se utilizan herramientas como MATLAB, Python o Simulink para diseñar los algoritmos necesarios.

Agradecemos la atención prestada en este archivo que contiene información importante y muy relevante sobre un “SISTEMAS BASADOS EN PROCESAMIENTO DIGITAL DE SEÑALES” el cual es un tema súper importante el cual debemos seguir analizando y estudiando para entender sus comportamientos.

Referencias:

[1]RODRIGO HUERTA CORTÉS y ALEJANDRO HERRERA, “INTRODUCCIÓN A LOS DSP'S”, LABORATORIO DE PROCESAMIENTO DIGIT. DE SEÑALES, 2004, art. n.º ELO–385. Accedido el 1 de diciembre de 2025. [En línea]. Disponible: https://www.um.es/documents/4874468/19345367/ssee-da-t04-01.pdf/20ed49bb-90d7-4d1b-a5d5-bc5cadc34fef

[2]JORDI SALAZAR. “Procesadores digitales de señal (DSP) Arquitecturas y criterios de selección”. DPTO. INGENIERÍA ELECTRÓNICA. CENTRO DE SISTEMAS Y SENSORES ELECTRÓNICOS, UNIVERSIDAD POLITÉCNICA DE CATALUÑA. [En línea]. Disponible: https://www.um.es/documents/4874468/19345367/ssee-da-t04-01.pdf/20ed49bb-90d7-4d1b-a5d5-bc5cadc34fef

[3]“A Beginner's Guide to Digital Signal Processing (DSP)”. https://www.analog.com/. Accedido el 1 de diciembre de 2025. [En línea]. Disponible: https://www.analog.com/en/lp/001/beginners-guide-to-dsp.html

[4]contributor1. “What Does DSP Stand For? 6 DSP Uses Explained with Pros and Cons - Hollyland”. Hollyland. Accedido el 1 de diciembre de 2025. [En línea]. Disponible: https://www.hollyland.com/blog/tips/what-does-dsp-stand-for

[5]“DSP y su funcionamiento”. Genius Audio. Accedido el 1 de diciembre de 2025. [En línea]. Disponible: https://geniusaudio.com/blogs/genius-blog/dsp-audio

INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

DIVISIÓN:

INGENIERÍA MECATRÓNICA

MATERIA:

SISTEMAS EMBEBIDOS BASADOS EN PROCESAMIENTO DIGITAL DE SEÑALES

DOCENTE:

JOSE ANGEL NIEVES VAZQUEZ

UNIDAD 5:

PROYECTO DE LA UNIDAD 5

ALUMNOS:

ISRAEL ANTONIO LÓPEZ ESCRIBANO
JESUS ALEJANDRO ROSAS ROSAS
ARGELIO SANTIAGO REYES

GRUPO:

711-B

FECHA DE ENTREGA:

San Andrés Tuxtla Ver. 01 de diciembre del 2025

Introducción:

En este proyecto se desarrolló una maqueta en la cual se colocó un sistema de control de acceso utilizando un teclado matricial 4x4 y un servomotor, programado con una tarjeta Arduino Uno. El objetivo principal fue crear un mecanismo de apertura que solo se active al ingresar una contraseña correcta, aplicando conceptos vistos en la materia de Sistemas Embebidos.

Materiales

- Arduino Uno

- Teclado matricial 4x4

- Servomotor

- Cables jumper

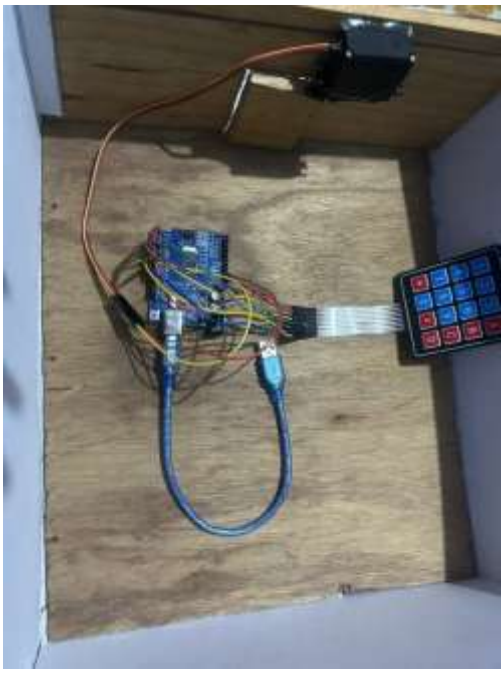
- Maqueta de puerta

Desarrollo

El sistema permite que el usuario ingrese una contraseña de cuatro dígitos mediante un teclado matricial. Si la clave coincide con la contraseña programada, el servomotor ejecuta una secuencia de apertura y cierre de puerta. En caso contrario, el sistema notifica error por el monitor serial.

Se configuró el teclado matricial con la librería Keypad.h, se programó el manejo del servomotor mediante señales PWM y se desarrolló la lógica de ingreso y validación de contraseña. Si la clave es correcta, se activa una secuencia de movimiento del servo

Esto lo colocamos en la maqueta de donde la puerta tiene movimiento:



Código

Para esto realizamos el siguiente código el cual al momento de ingresar la combinación correcta (1,2,3,4) la puerta se abre y luego se cierra:

```
#include <Keypad.h>
#include <Servo.h>

Servo myservo;

// ---- CONFIGURAR TECLADO 4x4 ----
const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'D','C','B','A'},
  {'#','9','6','3'},
  {'0','8','5','2'},
  {'*','7','4','1'}
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
// ---- VARIABLES ----
String inputCode = "";
String correctCode = "1234";

void setup() {
  Serial.begin(9600);
  myservo.attach(10);

  // Posición neutral (servo detenido)
  myservo.writeMicroseconds(1500);
}

void loop() {
  char key = keypad.getKey();

  if (key) {
    Serial.println(key);

    if (key == '#') {
      inputCode = "";
      Serial.println("Codigo borrado");
    } else {
      inputCode += key;
    }

    if (inputCode.length() == 4) {
      if (inputCode == correctCode) {
        Serial.println("Codigo correcto");
        moverServoSecuencia();
      } else {
        Serial.println("Codigo incorrecto");
      }
      inputCode = "";
    }
  }
}

// ---- FUNCION PARA SERVO CONTINUO POR TIEMPO ----
void moverServoSecuencia() {

  // 1️⃣ Girar primero al sentido inverso por 0.5 s
  myservo.writeMicroseconds(1000);  // Giro hacia el otro lado
  delay(500);

  // 2️⃣ Regresar al sentido contrario por 0.5 s
```

```
  myservo.writeMicroseconds(2000);  // Regresa
  delay(1000);

  // 3️⃣ Detener el servo
  myservo.writeMicroseconds(1500);
}
```

Resultados

Al colocar la contraseña al teclado (1234) la puerta abre y después de un momento cierra como lo muestra las siguientes imágenes.

En el siguiente link se encuentra el video donde se muestra que el proyecto funciona de manera muy eficaz y correcta.

https://drive.google.com/file/d/1s78Ijom6kR12SoUqMn2L25k5pQmY3_gP/view?usp=drivesdk

Conclusión:

El proyecto permitió aplicar conceptos fundamentales de sistemas embebidos, logrando un sistema de control de acceso de una puerta completamente funcional y demostrando la interacción efectiva entre hardware y software.