Nombre del alumno: Yahir Aguilar Sicalhua.

Nombre del tema: Unidad IV. Modelo de Implementación.

Parcial: 1.

Nombre de la materia: Ingeniería de Software.

Nombre del profesor: Juan José Ojeda Trujillo.

Nombre de la licenciatura: Ingeniería en Sistemas Computacionales.

Cuatrimestre: 8.

Unidad IV. Modelo de Implementación.

- 4.1 Modelos de Implementació
  - El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema.
- 4.2 Diagramas de Componentes.
  - Un componente es una parte física de un sistema (modulo, base de datos, programa ejecutable, etc.). Se puede decir que un componente es la materialización de una o más clases.
- 4.3 Elementos del Diagrama de Componentes.
  - Normalmente los diagramas de Componentes contienen:
  - • Componentes • Interfaces • Relaciones de dependencia, generalización, asociación y realización • Paquetes o subsistemas
- 4.4 Diagramas de Despliegue.
  - El Diagrama de Despliegue es un diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.
- 4.5 Modelos de Prueba.
  - Objetivos de las pruebas:
  - Encontrar defectos en el software. • Una prueba tiene éxito si descubre un defecto.
  - Una prueba fracasa si hay defectos, pero no los descubre. • Pruebas de Verificación
  - Pruebas de Validación
- 4.6 Implementación en Java de los Diagramas de Clase.
  - Cuando se construye un objeto es necesario dar un valor inicial a sus atributos, es por ello que existe un método especial en cada clase, llamado constructor, que es ejecutado de forma automática cada vez que es instanciada una variable.
- 4.7 Constructores y Destructores Declaración, Uso y Aplicaciones.
  - Para crear un objeto se necesita reservar suficiente espacio en memoria e inicializar los valores de los campos que representan el estado del objeto. Este trabajo es realizado por un tipo especial de método denominado constructor.
- 4.8 Realización de los Casos de Uso y Diagramas de Interacción.
  - Los casos de uso son una técnica para la especificación de requisitos funcionales propuesta inicialmente por Ivar Jacobson [Jacobson, 1987], [Jacobson et al. 1992] e incorporada a UML Modela la funcionalidad del sistema tal como la perciben los agentes externos, denominados actores, que interactúan con el sistema desde un punto de vista particular.
- 4.9 Arquitectura Lógica con Patrones: Separación Modelo-Vista.
  - Modelo-vista-controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.
- 4.10 MVC y Bases de Datos.
  - Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos el cual gestiona los datos que debe utilizar la aplicación; en líneas generales del MVC dicha gestión corresponde al modelo.
- 4.11 Uso en Aplicaciones Web.
  - Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor.

Fuentes Bibliográficas:
https://plataformaeducativauds.com.mx/libro.php?idLibro=171259172

Prototipo de Brazo Metálico (avances).

SERVO_JISTICK_2.ino

```ino
#include<Servo.h>
//DEFINE SERVOS
Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;

//DEFINIR PINES DE JOYSTICK (ANALOGOS)
int joyX = 0;
int joyY = 1;
int joyX2 = 2;
int joyY2 = 3;
//VARIABLE PARA LEER LOS VALORES DE LOS PINES ANALÓGICOS
int joyVa1;
int joyVa12;

void setup() {
//CONECTA NUESTROS SERVOS EN PINS PWM 3 , 5
  servo1.attach(3);
  servo2.attach(5);
  servo1.attach(6);
  servo2.attach(9);
  }

void loop() {

//LEER EL VALOR DEL JOYSTICK (ENTRE 0 - 1023)
  joyVal = analogRead (joyX);
joyVal = map (joyVa1, 0, 1023, 0, 180);
servol. write (joyVal);
```

Fase final de los códigos para su funcionamiento.

SERVO_JISTICK_2.ino

```ino
  servo2.attach(5);
  servo1.attach(6);
  servo2.attach(9);
  }

void loop() {

//LEER EL VALOR DEL JOYSTICK (ENTRE 0 - 1023)
  joyVal = analogRead (joyX);
joyVal = map (joyVa1, 0, 1023, 0, 180);
servol. write (joyVal);

joyVal = analogRead (joyX);
joyVal = map (joyVa1, 0, 1023, 0, 180);
servo2. write (joyVal);

joyVa2= analogRead (joyX2);
joyVa2 = map (joyVa2, 0, 1023, 0, 180);
servo3. write (joyVal2);

joyVa2 = analogRead (joyX2);
joyVa2 = map (joyVa12, 0, 1023, 0, 180);
servo4. write (joyVal2);
delay(15);
//AJUSTE LA POSICIÓN DEL SERVO SEGÚN EL VALOR DEL JOYSTICK
  }
```