NOMBRE DEL PROFESOR: Juan José Ojeda Trujillo

NOMBRE DEL ALMUNO: Eddi David Aguilar Martinez

MATERIA: Ingeniería en software

CARRERA: Ingeniería en sistemas computacionales

CUATRIMESTRE: 8

UNIDAD IV MODELO DE IMPLEMENTACIÓN

- 4.1 Modelos de implementación
  - El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.
- 4.2DIAGRAMAS 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, porque una abstracción con atributos y métodos pueden ser implementados en los componentes.
- 4.3 ELEMENTOS DEL DIAGRAMA DE COMPONENTES
  - Componentes
  - Interfaces
  - Asociación
  - Agregación
  - Realización
  - Asociación
- 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
  - Encontrar defectos en el software
  - Pruebas de Verificación
  - Pruebas de Validación
    - Pruebas en que se conoce el código a probar
    - Algunas clases de pruebas
    - Pruebas de condiciones
- 4.6 IMPLEMENTACIÓN EN JAVA DE LOS DIAGRAMAS DE CLASE
  - Cuando se va a construir un sistema software es necesario conocer un lenguaje de programación, pero con eso no basta. Si se quiere que el sistema sea robusto y sustentable, es necesario que el problema sea analizado y la solución sea cuidadosamente diseñada. Se debe seguir un proceso robusto, que incluya las actividades principales.
- 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

REPORTE DEL BRAZO ROBOTICO

SERVO_JISTICK_2.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);
servo1. write (joyVal);

```

SERVO_JISTICK_2.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);
servo1. 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
  }
```