UNIVERSIDAD AUTONOMA DEL ESTADO DE MEXICO

CENTRO UNIVERSITARIO UAEM TEXCOCO

"PROTOTIPO AUTOMATIZADO PARA REHABILITACIÓN FÍSICA DE LA MANO"

TESIS

PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION

PRESENTADO POR:
GONZALEZ RODRIGUEZ JESUS EDUARDO

DIRECTOR
DR. LUGO ESPINOSA OZIEL

REVISORES:
DR. ZARCO HIDALGO ALFONSO
DR. AYALA DE LA VEGA JOEL

2014

HOJA DE REGISTRO DE TESIS

Quien nunca ha cometido un error
nunca ha probado algo nuevo.
Albert Einstein

Una persona usualmente se convierte en aquello que el cree que es.
Si yo sigo diciéndome a mi mismo que no puedo hacer algo,
es posible que yo termine siendo incapaz de hacerlo.
Por el contrario si yo tengo la creencia que sí puedo hacerlo,
con seguridad yo adquiriré la capacidad de realizarlo aunque
no la haya tenido al principio.
Gandhi

Una persona no puede directamente escoger sus circunstancias,
pero si puede escoger sus pensamientos e indirectamente y
con seguridad darle forma a sus circunstancias,
James Allen

AGRADECIMIENTOS

A Dios

Le agradezco a Dios por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y brindarme una vida llena de aprendizajes, experiencias y sobre todo felicidad.

A mi Madre

Teresa por el gran amor y la devoción que tienes a tus hijos, por el apoyo ilimitado e incondicional que siempre me has dado, por tener siempre la fortaleza de salir adelante sin importar los obstáculos, por haberme formado como un hombre de bien, y por ser la mujer que me dio la vida y me enseñó a vivirla… no hay más palabras en este mundo para agradecerte, mama.

A mi Padre

Fernando por el valor y el coraje que has tenido para levantarme ante cualquier adversidad, por las enseñanzas que me has dado, y por darme animo siempre diciéndome lo orgulloso que te sientes de tus hijos, muchas gracias, papa.

A mis Hermanos

Karen y Fernando por apoyarme en aquellos momentos de necesidad y representar la unidad familiar llenado mi vida de alegrías y amor cuando más lo necesito. Los amo.

A mi Novia

Consuelo por ser una parte muy importante en mi vida, gracias por los sentimientos tan lindos que tienes hacia mí, gracias por haberme dado el impulso de seguir con mis estudios, por el apoyo, y sobre todo por su paciencia y amor incondicional.

A mis amigos

Jonathan, Sanabria, Chelo, Gabriel, Fátima, Gaby, Rosa, Silvia, Benita y Alejandra por pasar a mi lado los momentos de mi vida universitaria y estar siempre en las buenas y en las malas, jamás los olvidare.

Al Dr. Oziel

Le agradezco la confianza, apoyo, dedicación y por haberme brindado la oportunidad de desarrollar mi tesis profesional, por darme la oportunidad de crecer profesionalmente y aprender cosas nuevas.

Profesores:

Gracias Dr. Zarco, al Dr. Joel y al Mtro. Jair Palma les agradezco por todo el apoyo brindado a lo largo de la carrera, por su tiempo, amistad y por los conocimientos que me transmitieron.

A la Universidad

Por abrirme sus puertas y darme la gran oportunidad de culminar mi formación académica albergándome todos estos años.

Muchas gracias, lo logramos

J. Eduardo Gonzalez Rodríguez

Contenido

I INTRODUCCION 1
1.1 Introducción *1*
1.2 Justificación *2*
1.3 Planteamiento del problema *2*
1.4 Objetivos 3
1.4.1 Objetivo general *3*
1.4.2 Objetivos específicos *3*
1.5 Hipótesis 3
II ANTECEDENTES 4
2.1 Antecedentes Generales *4*
III SOFTWARE 5
3.1 Fundamentos Ingeniería de software *5*
3.2 Modelo de Diseño 7
3.2.1 El proceso de diseño *8*
3.2.2 Fundamentos de diseño *8*
3.2.3 Diseño de datos *9*
3.2.4 Principios para el diseño de datos. *9*
3.2.5 Diseño arquitectónico *10*
3.3 Modelo de Implementación 11
3.3.1 Diagrama de componentes *11*
3.3.2 Diagrama de despliegue *12*
3.4 Modelo de prueba 13
3.5 Seguridad en Ingeniería de Software 15
3.5.1 Problemática actual de la seguridad en el software *15*
3.5.2 Objetivos para un software seguro *15*
3.6 Desarrollo del Software 17
3.6.1 Java *17*
3.6.1.1 Botón “Registrar” 20
3.6.1.2 Botón “Ver Registro” 26
3.6.1.3 Botón “Valoración” 30

3.6.1.4 Botón “Salir” 39
3.6.2 Base de Datos (MySQL Workbench) *40*
3.6.2.1Modelo Entidad-Relación 41
3.6.2.2 Modelo Relacional 41
IV ARDUINO 44
4.1 Introducción al Arduino 44
4.2 Elementos necesarios 44
4.2.1 El Hardware de Arduino *44*
4.2.1.1 Placa Arduino 44
4.2.1.2 Cable de comunicación (Serie/USB) 45
4.2.1.3 Fuente de alimentation 46
4.3 ¿Con qué elementos podemos interactuar? 48
4.4 Software 49
4.4.1 Entorno de desarrollo *49*
4.4.2 Drivers UBS *49*
4.5 Instalación de Arduino en Windows 50
4.5.1 Configuración de las comunicaciones *52*
4.6 Desarrollo del Software 55
4.7 Subiendo el programa a la placa Arduino 56
V PROTOTIPO 59
5.1 Material 59
5.1.1 Servomotores *59*
5.1.2 Polea *62*
5.1.3 Base metálica *63*
5.1.4 Guante *63*
5.2 Costos 64
5.3 Armado/Ensamble 64
5.4 Conexión 73
5.4.1 Algoritmo de comunicación *75*
VI CONCLUSIONES Y RECOMEDACIONES 76
6.1 Conclusiones 76
6.2 Trabajos futuros 77

ANEXO..... 78
VII REFERENCIAS BIBLIOGRÁFICAS..... 79

Índice de imágenes y tablas

FIGURA. 1: PANTALLA PRINCIPAL 17
FIGURA. 2: MENÚ 19
FIGURA. 3: BOTÓN REGISTRAR 20
FIGURA. 4: REGISTRO PACIENTE/CITA 21
FIGURA. 5: REGISTRO PACIENTE 22
FIGURA. 6: REGISTRO PACIENTE GUARDADO 23
FIGURA. 7: BOTÓN ATRÁS 24
FIGURA. 8: REGISTRO CITA 25
FIGURA. 9: AGENDAR CITA 26
FIGURA. 10: VER REGISTRÓ 27
FIGURA. 11: VER REGISTRÓ PACIENTE 28
FIGURA. 12: ELIMINAR REGISTRÓ PACIENTE/CITA 29
FIGURA. 13: VER REGISTRO CITA 29
FIGURA. 14: VALORACIÓN 30
FIGURA. 15: SELECCIONAR RUTINA 31
FIGURA. 16: CREAR RUTINA 31
FIGURA. 17: RUTINA NUEVA 32
FIGURA. 18: NUMERO DE RUTINA 33
FIGURA. 19: NOMBRE RUTINA 34
FIGURA. 20: RUTINA GUARDADA 35
FIGURA. 21: SELECCIÓN RUTINA 35
FIGURA. 22: RUTINA SELECCIONADA 36
FIGURA. 23: INICIO RUTINA 37
FIGURA. 24: PAUSA/REANUDACIÓN RUTINA 38
FIGURA. 25: DETENER RUTINA 38
FIGURA. 26: SALIR DEL SISTEMA 39
FIGURA. 27: SALIR DEL SISTEMA 40
FIGURA. 28: MODELO ENTIDAD-RELACION 41
FIGURA. 29: TABLAS GENERADAS 41
FIGURA. 30: PLACA USB 45
FIGURA. 31: CABLE USB 46
FIGURA. 32: POLARIDAD 47
FIGURA. 33: FUENTE DE ALIMENTACIÓN 47
FIGURA. 34: ALIMENTACIÓN A TRAVÉS DEL CABLE USB 47
FIGURA. 35: DESCRIPCIÓN DE COMPONENTES ARDUINO 48
FIGURA. 36: ENTORNO DE DESARROLLO 49
FIGURA. 37: ASISTENTE DE INSTALACIÓN 51
FIGURA. 38: ASISTENTE DE INSTALACIÓN 51
FIGURA. 39: ASISTENTE DE INSTALACIÓN 52
FIGURA. 40: CONFIGURACIÓN PUERTO SERIE 53
FIGURA. 41: ADMINISTRADOR DE DISPOSITIVOS 53
FIGURA. 42: CONFIGURACIÓN DE VELOCIDAD 54
FIGURA. 43: CONFIGURACIÓN DE PUERTO USB 54
FIGURA. 44: VERIFICANDO EL CÓDIGO FUENTE 56

FIGURA. 45: COMPROBACIÓN CORRECTA 57
FIGURA. 46: SUBIENDO EL PROGRAMA A LA PLACA 57
FIGURA. 47: SUBIENDO EL PROGRAMA 57
FIGURA. 48: PROGRAMA CARGADO CORRECTAMENTE 58
FIGURA. 49: SERVOMOTOR 59
FIGURA. 50 SEÑAL DEL SERVOMOTOR 60
FIGURA. 51 5% DEL CICLO DE TRABJO 61
FIGURA. 52 7.5% DE CICLO DE TRABAJO 61
FIGURA. 53 12.5% CICLO DE TRABAJO 62
FIGURA. 54 POLEA 62
FIGURA. 55 BASE METALICA 63
FIGURA. 56 GUANTE 63
FIGURA. 57 DISEÑO PROTOTIPO 64
FIGURA. 58 BOCETO DIMENCIONES 65
FIGURA. 59 BOCETO PARTE SUPERIOR 65
FIGURA. 60 BOCETO PARTE INFERIOR 66
FIGURA. 61 PRIMER PROTOTIPO FASE 1 66
FIGURA. 62 PRUEBA PRIMER PROTOTIPO FASE 1 67
FIGURA. 63 PRIMER PROTOTIPO FASE 2 68
FIGURA. 64 PRIMER PROTOTIPO FASE 3 68
FIGURA. 65 PROTOTIPO BASE 69
FIGURA. 66 PROTOTIPO POLEAS 70
FIGURA. 67 PROTOTIPO SERVOMOTOR LATERAL 70
FIGURA. 68 PROTOTIPO COLOCACIÓN SERVOMOTORES 71
FIGURA. 69 PRIMER GUANTE 72
FIGURA. 70 GUANTE FINAL 72
FIGURA. 71 DIAGRAMA CONEXIÓN 73
FIGURA. 72 DISTRIBUCIÓN PINES 73
FIGURA. 73 PINES 74
FIGURA. 74 MODELO CASO DE USO 78

TABLA. 1 PACIENTE 43
TABLA. 2 CÓDIGO POSTAL 43
TABLA. 3 ESTADO 43
TABLA. 4 NUEVA CITA 43
TABLA. 5 RUTINA 43
TABLA. 6 COSTOS 64

RESUMEN

En el presente trabajo se muestra la relación de la computación con la ciencia de la salud, ya que se propone un prototipo para la automatización de la rehabilitación física de los dedos de la mano. El elemento base del prototipo es un conjunto de servomotores controlados por la plataforma Arduino que se conecta a un computadora. Se presenta a su vez el software que controla los servomotores y almacena los datos generados en una base de datos para un análisis posterior.

La utilización del software libre Java para generar rutinas de movimiento y control de los servomotores, la construcción del prototipo con materiales económicos y el uso de la plataforma Arduino, reduce costos y evita el traslado de las personas afectadas hacia centro especializados de rehabilitación física, que dan como resultado un impacto social directo a la población afectada.

I INTRODUCCIÓN

1.1 Introducción

La mano constituye un órgano extremadamente complejo morfológicamente pues está formado por muy disímiles estructuras, pero funcionalmente no es menos complejo, pues este órgano se encuentra en la extremidad distal del antebrazo siendo de suma importancia para poder articular al resto del cuerpo y por el cual nos permitimos la comunicación del medio con diferentes zonas cerebrales donde se integra la información y regresa una respuesta a ejecutar, esta es la verdadera esencia de toda la actividad que la mano puede realizar, no es posible ver la mano como un simple órgano más, el desarrollo alcanzado por este, así como las distintas actividades que realiza obedece a las necesidades funcionales que el encéfalo ha ido exigiendo.

La habilidad en el movimiento de los dedos, obedece al mandato encefálico que a su vez responde al programa de movimiento previamente creado, es decir la mano desconectada al sistema nervioso central y periférico es un ente inservible.

Es por eso que la fisioterapia y el terapeuta ocupacional son los profesionales imprescindibles que tratan la alteración de la función motora y favorecen la conservación o adquisición de la máxima autonomía o independencia del paciente en el entorno. Las rutinas o terapias que se aplican al paciente consisten en diferentes movimientos que estimulan los tendones y músculos que componen a la parte del cuerpo humano que perdió movilidad. Estas rutinas tienen que ser constantes para obtener un beneficio palpable, lo que se traduce en invertir recursos económicos y tiempo para acudir a la terapia.

1.2 Justificación

La presente Tesis se realizó para dar un resultado por escrito en el desarrollo del proceso para automatizar alguna disciplina sanitaria orientada a la prevención, tratamiento y la rehabilitación física de las alteraciones funcionales de la mano como son dedos y muñeca, se busca el diseño de software libre para generar rutinas de movimiento en servomotores, los cuales disminuyen costos y proporciona una plataforma robusta para controlar el dispositivo generando un impacto social directo a la población afectada, beneficiando tanto a individuos cualquier complexión fisca, clase social, condición motora, familia y la comunidad en el ambiente donde se desenvuelvan, dotándolos de acceso a la rehabilitación física sin tener que acudir a instalaciones especializadas o cubrir altos costos por la obtención de la rehabilitación en donde de forma inmediata será posible la utilización del dispositivo en cualquier lugar y hora sin la necesidad de un especialista frente a la persona.

1.3 Planteamiento del problema

En nuestro país, el porcentaje de discapacitados y limitados tanto funcionales como estructurales alcanzan cifras que superan el 5.1% del total de la población [14]. A este porcentaje agreguémosle el rubro de quienes al momento están enfermos y con alto riesgo de padecer discapacidad o quedar con limitaciones funcionales o estructurales. Si a esto le sumamos que en nuestro país no existen programas de prevención de patologías incapacitantes y limitantes, por lo que se deben considerar los potenciales nuevos casos de discapacitados y limitados que se irán sumando en los próximos años. Por otro lado tenemos también que el 75% de las personas con algún daño cerebral causado por accidentes o golpes [15], pierden el movimiento de los dedos y/o brazo. Con todo lo anterior podemos generar la siguiente pregunta de investigación:

¿Será posible la construcción de un dispositivo para automatizar la rehabilitación física de los dedos y muñeca de la mano, reduciendo costos con la ayuda de hardware y software libre?

1.4 Objetivos

1.4.1 Objetivo general

Implementar un dispositivo automatizado así como un guante para mejorar la calidad de vida de las personas que son afectadas por la pérdida de movilidad de dedos y muñeca, alcanzando el mayor grado de autonomía física y optimización para una mejor realización de sus actividades de la vida diaria del paciente.

1.4.2 Objetivos específicos

- Diseñar una base de datos para llevar el control de los pacientes así como el avance de cada uno de ellos.
- Crear un guante el cual sea fácil de adaptarse a las distintas manos de cada paciente.
- Construir un prototipo para dar soporte a mano y dedos de pacientes con pérdida de movilidad en mano(s) y dedo(s).

1.5 Hipótesis

Al implementar el prototipo ayudará a las personas a recuperar movilidad ya que pueden tener acceso a las rutinas de rehabilitación con menor costo y sin tener que trasladarse o tener alguna persona especializada en rehabilitaciones físicas.

II ANTECEDENTES

2.1 Antecedentes Generales

Se han desarrollado dispositivos robóticos que ayudan a mejorar los movimientos de los músculos tendones y ligamentos para ayudar a la rehabilitación de pacientes con problemas en sus extremidades. Los aparatos utilizan plásticos suaves y materiales compuestos en lugar de un exoesqueleto rígido. Estos materiales se combinan con músculos neumáticos artificiales, sensores de peso y un software de control para conseguir movimiento natural.

El sistema reproduce la arquitectura muscular-tendón-ligamento-piel de la anatomía. La ortesis suave también alberga una serie de sensores situados en la las articulaciones que miden el movimiento, la tensión y la presión. Está alimentado por una serie de baterías de iones de litio y está atado a una fuente de aire que mueve los músculos artificiales.

Los dispositivos robóticos también podrán ayudar a personas con desórdenes neuromusculares asociados con parálisis cerebral, esclerosis lateral amiotrofia o esclerosis múltiple. Sin embargo, los expertos han advertido que un uso a largo plazo puede conllevar atrofia muscular [4].

III SOFTWARE

3.1 Fundamentos Ingeniería de software

¿Qué es software?

- Programas de cómputo y su documentación asociada: requerimientos, modelos de diseño y manuales de usuario
- El software puede ser desarrollado para un cliente en particular o para un mercado general
- El software puede ser:
  - Genérico: desarrollado para venderse a múltiples clientes (Excel, Word, etc.)
  - A la medida: desarrollado bajo demanda del cliente a un desarrollador específico

¿Qué es la Ingeniería de Software?

- Una disciplina de la Ingeniería que concierne a todos los aspectos de la producción de software
- Los Ingenieros de Software deben:
  - Adoptar un enfoque sistemático para llevar a cabo su trabajo Utilizar las herramientas y técnicas apropiadas para resolver el problema planteado, de acuerdo a las restricciones de desarrollo y a los recursos disponibles

¿Cuál es la diferencia entre Ingeniería de Software e Ingeniería de Sistemas?

- La Ingeniería de Sistemas concierne a todos los aspectos del desarrollo de sistemas basados en cómputo incluyendo hardware, software y la ingeniería de procesos.
- La Ingeniería de Software es una parte de este proceso que comprende el desarrollo de software, control, aplicaciones y bases de datos del sistema

¿Qué es un proceso de software?

- Un conjunto estructurado de actividades cuya meta es el desarrollo o evolución de un software.
- Algunas actividades genéricas en todos los procesos de software son:
  - Especiación, qué debe hacer el software y cuáles son sus especiaciones de desarrollo.
  - Desarrollo, producción del sistema de software.
  - Validación, verificar que el software cumple con lo solicitado por el cliente.
  - Evolución, cambiar/adaptar el software a las nuevas demandas.

¿Qué es un modelo de proceso de software?

- Representación formal y simplificada de un proceso de software, presentada desde una perspectiva especiada.
- Ejemplos de perspectivas del proceso de software:
  - Flujo de trabajo, secuencia de actividades
  - Flujo de datos, flujo de la información
  - Rol/acción, quién realiza qué

- Modelos Genéricos:
  - Cascada, separar en distintas fases de especiación y desarrollo
  - Desarrollo Iterativo, la especiación, desarrollo y validación están interrelacionados
  - Prototipado, un modelo sirve de prototipo para la construcción del sistema final
  - Basado en componentes, asume que partes del sistema ya existen y se enfoca a su integración

¿Cuáles son los costos de la Ingeniería de Software?

El costo total de un software está dividido aproximadamente de la siguiente forma:

  - 60 % costos de desarrollo
  - 40 % costos de pruebas
- En el software a la medida los costos de evolución a menudo exceden los costos de desarrollo
- Los costos dependen del tipo de sistema que se desarrolla y de los requerimientos del mismo tales como desempeño y confiabilidad
- La distribución de los costos depende del modelo de desarrollo empleado

¿Qué es CASE?

- CASE (Computer-Aided Software Engineering)
- Programas que son usados para dar soporte automatizado a las actividades del proceso de software:
  - Análisis de requerimientos, modelado del sistema, pruebas y depuración (debugging)
- Las herramientas CASE son comúnmente usadas para dar soporte a los métodos de software
  - Editores para la notación del método
  - Módulos de análisis que verifican que las reglas del método se cumplan
  - Generadores de reportes que facilitan la creación de la documentación del sistema
  - Generadores de código a partir del modelo del sistema

¿Cuáles son los atributos del software de calidad?

- El software debe proveer la funcionalidad y desempeño requeridos por el usuario y debe ser mantenible, confiable y aceptable
  - Mantenible, el software debe poder evolucionar para continuar cumpliendo con las especificaciones
  - Confiable, el software no debe causar daños físicos o económicos en el caso de que falle
  - Eficiente, el software no debe desperdiciar los recursos del sistema
  - Aceptable, el software debe ser aceptado por los usuarios para los que fue diseñado. Debe ser entendible, utilizable y compatible con otros sistemas

3.2 Modelo de Diseño

El software provee la funcionalidad y desempeño requeridos por el usuario y es mantenible, confiable y aceptable.

Mantenible puede evolucionar para continuar cumpliendo con las especificaciones. Confiable, el software no causa daños físicos o económicos en el caso de que falle Eficiente, el software no desperdicia los recursos del sistema.

Aceptable, el software es aceptado por los usuarios para los que fue diseñado. Es entendible, utilizable y compatible con otros sistemas.

El diseño de software es la primera de tres actividades técnicas:

- Diseño
- Codificación
- Prueba

Mediante alguna de las metodologías existentes para el diseño se realizan tres tipos de diseño:

a) Diseño de Datos: Transforma el modelo del campo de la información en las estructuras de datos que se van a requerir para implementar el software.

b) Diseño Arquitectónico. Define las relaciones entre los principales elementos estructurales del programa.

c) Diseño Procedimental. Transforma los elementos estructurales en una descripción procedimental del software.

d) Diseño de la Interfaz. Establece la disposición y los mecanismos para la interacción Hombre-Máquina.

3.2.1 El proceso de diseño

El diseño de software es un proceso mediante el que se traducen los requisitos en una representación del software.

El diseño se realiza en dos pasos:

1. El diseño preliminar. Se centra en la transformación de requisitos en los datos y la arquitectura del software.

2. El diseño detallado. Se ocupa del refinamiento de la representación arquitectónica que lleva a una estructura de datos detallada y a las representaciones algorítmicas del software.

3.2.2 Fundamentos de diseño

a) Modularidad

El software se divide en componentes con nombres determinados que se denominan módulos. Un programa compuesto de un solo módulo no puede ser fácilmente manejado intelectualmente. Es más fácil resolver problemas complejos cuando se descomponen en trozos más manejables.

b) Arquitectura del Software

La arquitectura del software se refiere a:

1. La estructura jerárquica de los componentes procedimentales, y
2. La estructura de los datos.

La arquitectura del software se obtiene mediante un proceso de partición, que relaciona los elementos de una solución de software con partes de un problema del mundo real definido en el análisis de requisitos.

c) Jerarquía de Control

La jerarquía de control, también denominada “estructura del programa”, representa la organización de los componentes del programa (módulos). Esto no representa aspectos procedimentales del software, tales como la secuencia de procesos, la ocurrencia u orden de las decisiones o la repetición de operaciones.

d) Procedimientos del Software.

El procedimiento del software se centra sobre los detalles de procesamiento de cada módulo individual. El procedimiento debe proporcionar una especificación precisa del procesamiento, incluyendo la secuencia de procesos, las decisiones y la repetición de operaciones. La representación procedimental del software se realiza por capas.

3.2.3 Diseño de datos

El diseño de datos es la primera de las tres actividades de diseño, los datos bien diseñados pueden conducir a una mejor estructura de programa, a una modularidad efectiva y a una complejidad procedimental reducida.

3.2.4 Principios para el diseño de datos.

1.- Deben identificarse todas las estructuras de datos y las operaciones que se han de realizar sobre cada una de ellas.

2.- Debe establecerse y usarse un diccionario de datos para definir el diseño de los datos del programa.

3.- El diseño de datos de bajo nivel debe realizarse hasta el diseño detallado.

4.- El lenguaje de programación debe soportar la especificación y la realización de tipos abstractos de datos

3.2.5 Diseño arquitectónico

El objetivo principal del diseño arquitectónico es desarrollar una estructura de programa modular y representar las relaciones de control entre los módulos.

Los métodos de diseño disponibles para realizar el diseño arquitectónico son:

a) Diseño orientado al flujo de datos (estructurado)

b) Diseño orientado a los objetos

c) Diseño orientado a los datos

- La especificación del diseño

El equipo de diseño debe generar un documento llamado “Especificación del Diseño del Software” en dos etapas:

1.- Primero el diseño estructural en una versión preliminar.

2.- Un documento de diseño detallado.

- Formato de especificación de diseño arquitectónico

1. Descripción conceptual de estructuras y bases de datos

2. Nombres y atributos de los elementos de datos

3. Nombre y descripción funcional de cada modulo

4. Especificación de interfaces para cada modulo

5. Estructura de interconexión entre módulos

6. Interconexiones entre módulos y estructuras de datos

- Formato de diseño detallado

1. Descripción física de estructuras y bases de datos

- Algoritmos detallados para cada módulo
- Técnicas específicas de programación
- Procedimientos de inicio
- Especificación de diccionario de datos para todos los elementos

3.3 Modelo 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.

Un diagrama de implementación muestra:

- Las dependencias entre las partes de código del sistema (diagramas de componentes).
- La estructura del sistema en ejecución (diagrama de despliegue).

3.3.1 Diagrama 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.

Respecto a los componentes

- Es implementado por una o más clases/objetos del sistema.
- Es una unidad autónoma que provee una o más interfaces.

- Las interfaces representan un contrato de servicios que el componente ofrece.

Los componentes pueden ser:

- Archivos
- Código fuente + Cabeceras
- Librerías compartidas
- Ejecutables
- Paquetes

Muestra como el sistema está dividido en componentes y las dependencias entre ellos.

- Proveen una vista arquitectónica de alto nivel del sistema.
- Ayuda a los desarrolladores a visualizar el camino de la implementación.
- Permite tomar decisiones respecto a las tareas de implementación y los Skills requeridos.

3.3.2 Diagrama 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.

- Permiten modelar la disposición física o topología de un sistema.
- Muestra el hardware usado y los componentes instalados en el hardware.
- Muestra las conexiones físicas entre el hardware y las relaciones entre componentes.

Usos que se les da a los diagramas de despliegue son para modelar:

- Sistemas cliente-servidor
- Sistemas completamente distribuidos
-

El elemento principal del diagrama son los NODOS.

Los nodos representan un recurso físico:

- Computadoras
- Sensores
- Impresoras
- Servidores
- Dispositivos externos

Los nodos pueden ser interconectados mediante líneas para describir una estructura de red.

Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento.

Estereotipo de nodo

- Estereotipo, son cosas u objetos q se repiten sin variación.
- El estereotipo de un nodo es la manera de poder verificar que tipo de nodo es el que se está observando.

3.4 Modelo 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: Ver si cumple las especificaciones de diseño

Pruebas de Validación: Ver si cumple los requisitos del análisis.

El proceso de pruebas del software tiene dos objetivos:

1. Demostrar al desarrollador y al cliente que el software satisface sus requerimientos.

2. Descubrir defectos en el software: que su comportamiento es incorrecto, no deseable o no cumple su especificación.

Pruebas de “caja blanca”

- Pruebas en que se conoce el código a probar
- Caja blanca (clear box: caja clara o transparente)
- Se procura ejercitar cada elemento del código

Algunas clases de pruebas

- Pruebas de cubrimiento
- Pruebas de condiciones
- Pruebas de bucles

Pruebas de “caja negra”

- Pruebas en que se conoce sólo la interfaz
- Caja negra (black box: caja opaca)
- Se procura ejercitar cada elemento de la interfaz

Algunas clases de pruebas

- Cubrimiento invocar todas las funciones (100%)
- Clases de equivalencia de datos
- Pruebas de valores límite

Estrategias de prueba del software

- Pruebas de unidades: Se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software: el componente o módulo del software.

- Pruebas de integración: La prueba de integración es una técnica sistemática para construir la arquitectura del software, mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados con la interfaz

- Pruebas de regresión: La prueba de integración es una técnica sistemática para construir la arquitectura del software, mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados con la interfaz.

- Pruebas de validación: Las pruebas de validación empiezan tras la culminación de la prueba de integración, cuando se han ejercitado los componentes individuales. Se ha terminado de ensamblar el software como paquete y se han descubierto y corregido los errores de interfaz

3.5 Seguridad en Ingeniería de Software

El concepto de la seguridad en los sistemas de software es un área de investigación que ha pasado a ser vital dentro de la Ingeniería de Software. Con el crecimiento de Internet, y otras aplicaciones sobre redes, como el comercio electrónico, correo electrónico, etc., la posibilidad de ataques se ha incrementado notablemente, como también lo han hecho las consecuencias negativas de estos ataques.

3.5.1 Problemática actual de la seguridad en el software

Los puntos débiles más importantes de la Ingeniería de Software con respecto a la seguridad pueden ser clasificados en dos grandes categorías:

- Fallas para implementar software seguro.
- Fallas para implementar seguridad en el software.

Fallas para implementar software seguro

Lamentablemente, la mayoría de las herramientas que tiene disponible un desarrollador de software sufren de fallas propias de seguridad. Una de las debilidades más trascendentes al momento de implementar software seguro surge del estado de los lenguajes de programación desde el punto de vista de la seguridad. Son escasos los lenguajes que proveen primitivas “seguras” que ayuden al programador a escribir un mejor código.

3.5.2 Objetivos para un software seguro

En pos de conseguir un software seguro, se debe dejar claro qué se entiende por seguridad, para así luego poder establecer requisitos mínimos que debe satisfacer un sistema que pretenda ser considerado seguro.

Las múltiples dimensiones de la seguridad son:

- Autenticación: el proceso de verificar la identidad de una entidad.
- Control de acceso: el proceso de regular las clases de acceso que una entidad tiene sobre los recursos.
- Auditoria: un registro cronológico de los eventos relevantes a la seguridad de un sistema. Este registro puede luego examinarse para reconstruir un escenario en particular.
- Confidencialidad: la propiedad de que cierta información no esté disponible a ciertas entidades.
- Integridad: la propiedad de que la información no sea modificada en el trayecto fuente-destino.
- Disponibilidad: la propiedad de que el sistema sea accesible a las entidades autorizadas.
- No repudio: la propiedad que ubica la confianza respecto al desenvolvimiento de una entidad en una comunicación.

La seguridad puede tener diferentes significados en distintos escenarios. En general, cuando se habla de seguridad implica referirse a más de una de las dimensiones mencionadas anteriormente. Por ejemplo:

- Seguridad en correo electrónico: involucra confidencialidad, no repudio e integridad.
- Seguridad en compras online: implica autentificación, confidencialidad, integridad y no repudio.

Bajo este punto de vista, se define un ataque a la seguridad como un intento de afectar en forma negativa una o más de las dimensiones del concepto de seguridad.

Una vez definido el concepto de seguridad, se pueden establecer objetivos básicos para un software seguro:

- Independencia de la seguridad: la seguridad debe construirse y utilizarse de manera independiente de la aplicación.
- Independencia de la aplicación: la aplicación no debe depender del sistema de seguridad usado, debe ser desarrollada y mantenida en forma separada.

- Uniformidad: la seguridad debe aplicarse de manera correcta y consistente a través de toda la aplicación y del proceso que desarrolla la misma.
- Modularidad: mantener la seguridad separada. Entre otras ventajas, esto nos brindará mayor flexibilidad y menor costo de mantenimiento.
- Ambiente seguro: se debe partir de un entorno confiable. Es decir, las herramientas de desarrollo y lenguajes de programación no deben contener agujeros de seguridad.
- Seguridad desde el comienzo: la seguridad debe ser considerada como un requerimiento desde el inicio del diseño.

3.6 Desarrollo del Software

Desarrollar un software significa construirlo simplemente mediante su descripción. Está es una muy buena razón para considerar la actividad de desarrollo de software como una ingeniería. En un nivel más general, la relación existente entre un software y su entorno es clara ya que el software es introducido en el mundo de modo de provocar ciertos efectos en el mismo, a continuación se desarrollara una breve explicación del mismo.

3.6.1 Java

La Figura 1, muestra la pantalla principal del proyecto, dentro de esta interfaz contamos con el botón “Entrar” el cual lleva acabo la instrucción de cargar las librerías de Driver MySQL JDBC para la base de datos y la librería RXTX para la plataforma Arduino.

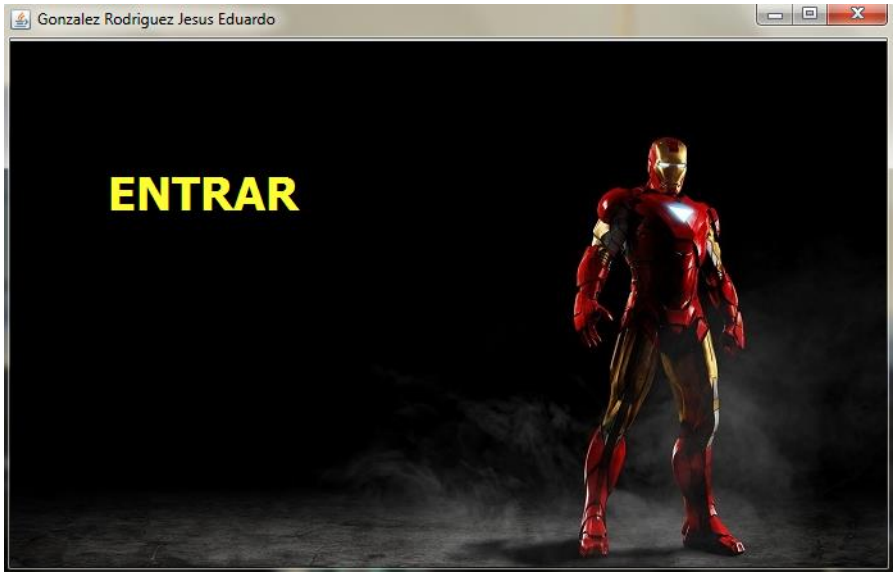


Figura. 1: Pantalla principal

Código Realizado

Java

Driver MySQL JDBC (Java Database Connectivity)

- Notas de ayuda:
  Conectividad de Java con la Base de datos

Class.forName("com.mysql.jdbc.Driver").newInstance();

- Notas de ayuda:
  En primer lugar, se debe cargar en memoria el Driver, esto se hace mediante el método cargarDriver que se ejecuta desde el constructor de la clase:

Class.forName("com.mysql.jdbc.Driver");

- Notas de ayuda:
  Se tiene una clase llamada "Class" que tiene un método estático llamado *forName*, al mismo hay que pasar el nombre de la clase a importar.

  com.mysql.jdbc es el nombre del paquete donde se encuentra la clase Driver, de esta forma se importa el driver en Java.

Dentro de la clase DriverManager se tiene un método llamado *getConnection* que retorna un objeto a la clase Connection:

miConexion = DriverManager.getConnection(url, usuario, password);

- Notas de ayuda:
  A este metodo se le pasan tres String el primero indica el nombre de la base de datos que queremos acceder (en este caso " IronHand"), el segundo metodo es el nombre de usuario (recordemos que cuando instalamos el MySQL se crea un usuario por defecto llamado "root") y el último es la clave del usuario "root", por defecto esta clave es un String vacío.

Luego se crea un objeto de la clase *Statement* a partir del objeto de la clase *Connection*:

Statement comando=conexion.createStatement();

- Notas de ayuda:
  La clase *Statement* tiene un método llamado *executeUpdate* que pasa el comando SQL insert para agregar una fila a la tabla articulos:

Seguidamente se define una variable de la clase ResultSet llamada consulta_mysql y se llama al método executeQuery de la clase Statement del objeto que se acaba de crear previamente:

tablaresultado = acuerdo.executeQuery(consultaSQL);

Una vez establecida la conexión con ambos software esta lista para almacenar en la base de datos así como también comenzar con la rutina para lo que es el movimiento de los dedos. Dentro de la Figura 2, se cuenta con tres botones el primero de ellos “Registrar” el segundo será “Valoración” y el tercero “Ver registro” cada uno de ellos será descrito a continuación:

Figura. 2: Menú

3.6.1.1 Botón “Registrar”

En la Figura 3, se muestra un botón el cual tiene la acción de mandar una pantalla nueva dentro del cual se puede hacer el registro de un paciente nuevo así como asignarle una nueva cita a paciente previamente existente dentro de la base de datos.

Figura. 3: Botón Registrar

Código Realizado

```
JDialogPacienteCita derecha=new JDialogPacienteCita(null, true);
derecha.setLocationRelativeTo(null);
derecha.setVisible(true);
```

- Notas de ayuda:
  Establece la posición de la ventana relativa a un componente pasado como parámetro, en este caso se le pasa null como parámetro, posicionandolo en el centro de la pantalla.

En la Figura 4, se puede apreciar que se cuenta con un obotn para registrar paciente y otro para agendar una cita al paciente existente o previamente dado de alta en la base de datos a continuación se describe la selccion de cada uno de ellos:

Figura. 4: Registro paciente/cita

Seleccionando PACIENTE

Se lleva un registro sencillo pero completo para cada paciente en el cual tienen campos como lo que es un numero único e irrepetible para evitar que se tenga confusión, a esto se le suma su RFC generado automáticamente, nombre completo, fecha de nacimiento, teléfono, dirección, el tipo de lesión que presenta y la fecha en la cual se está dando de alta.

Código Realizado:

```
JDialog10BDDatos derecha=new JDialog10BDDatos(null, true);
derecha.setLocationRelativeTo(null);
derecha.setVisible(true);
```

- Notas de ayuda:
  Establece la posición de la ventana relativa a un componente pasado como parámetro, en este caso se le pasa null como parámetro, mostrandolo en el centro de la pantalla.

La Figura 5, muestra al accionar del botón Paciente dentro de la cual lanza un número único automáticamente con el cual se identifica al paciente.

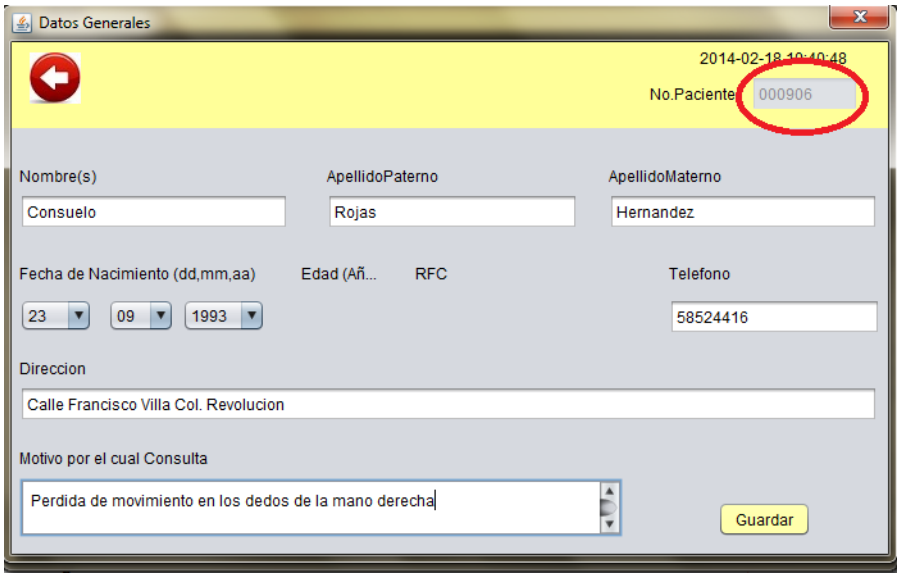


Figura. 5: Registro paciente

Código Realizado:

Random r = new Random();

- Notas de ayuda:
  Proporciona un generador de números aleatorios.

Figura 6, muestra un boton “Guardar” que inserta los valores de los campos previamente llenados a la base de datos.

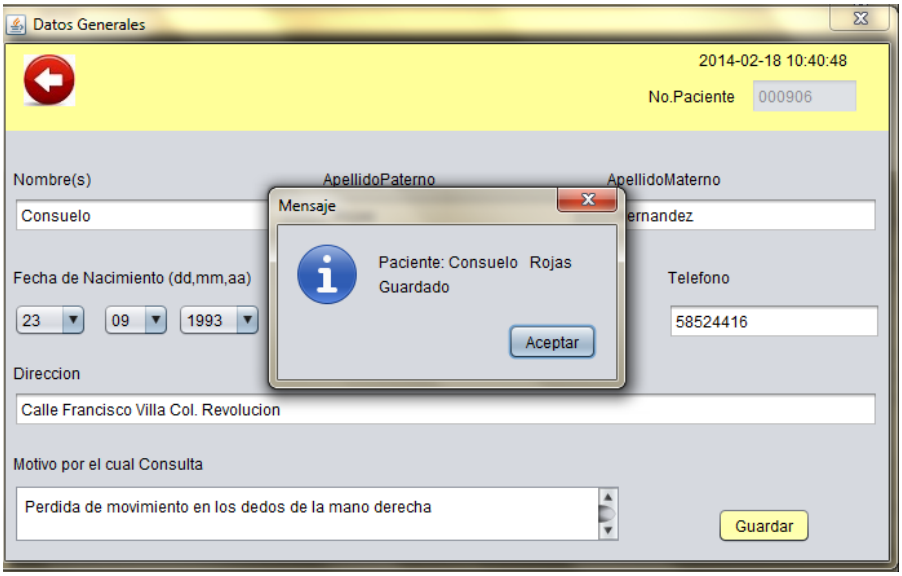


Figura. 6: Registro paciente guardado

Código Realizado

String nombre = jTextFieldnombre.getText();
String sentenciaInsert = "INSERT INTO PACIENTE VALUES (?,?....)";

- Notas de ayuda de su funcion:
  Captura cada uno de los valores que fueron colocados en los campos y los inserta en filas múltiples dentro de una tabla creada en la base de datos llamada PACIENTE.

Figura 7, muestra un boton “atrás” que tiene como medida de seguridad aseguar no sea perdida la informacion por algun error.

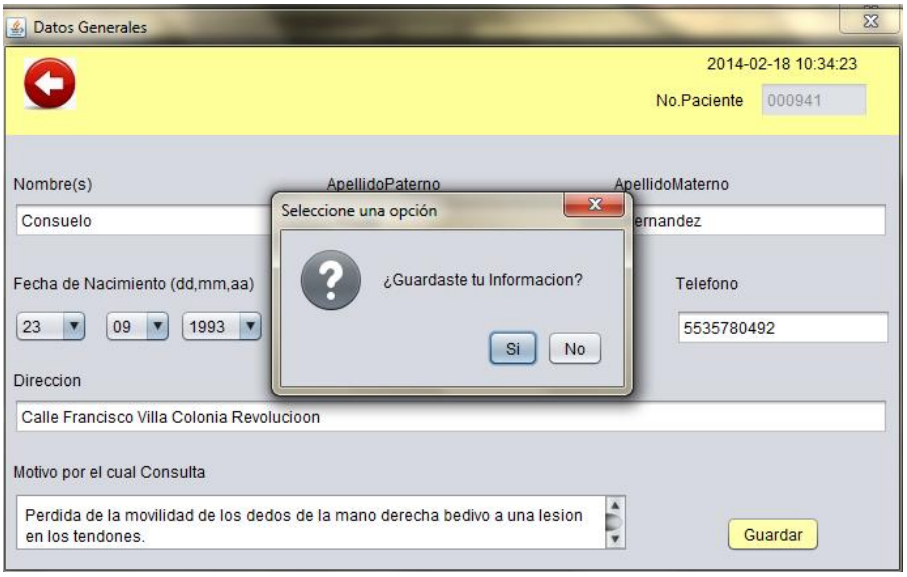


Figura. 7: Botón atrás

Código Realizado

int seleccion = JOptionPane.showOptionDialog

- Notas de ayuda:
  Manda un cuadro de diálogo simple con una advertencia, si se presiona “SI” mandara a la pantalla con el menú mientras la opción “NO” deja la misma venta pudiéndose hacer cambios o verificar que se allá guardado correctamente.

Seleccionando CITA

Esta selección se agenda una cita para el paciente ya registrado, para ello es necesario que seleccionar el número de paciente que se generó al ser dado de alta en la base de datos, como se muestra en la Figura 8.

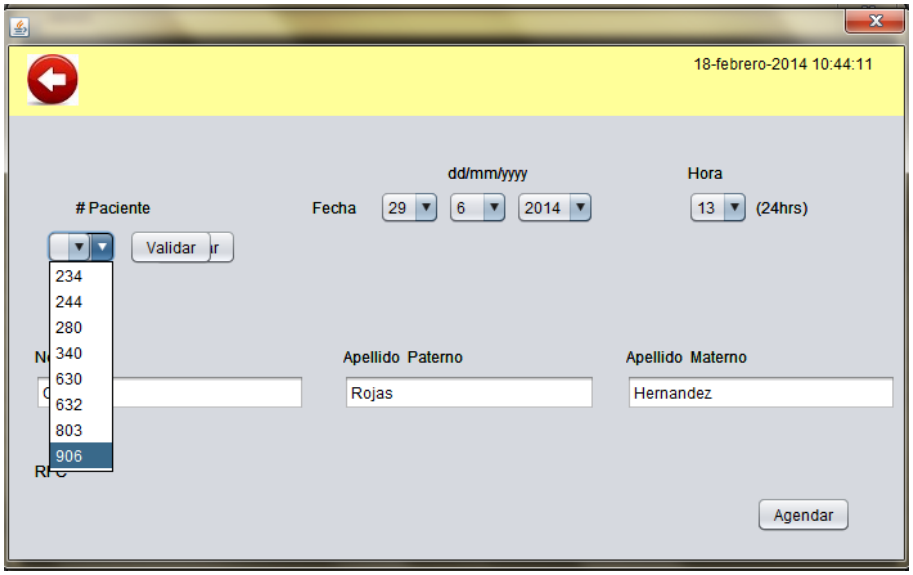


Figura. 8: Registro cita

Código Realizado

```
JDialogAgendar derecha=new JDialogAgendar(null, true);
derecha.setLocationRelativeTo(null);
derecha.setVisible(true);
```

- Notas de ayuda:
  Establece la posición de la ventana relativa a un componente pasado como parámetro, en este caso se le pasa null como parámetro, mostrandolo en el centro de la pantalla.

Figura 9, muestra un boton“Agendar” ubicado en la parte inferor derecha el cual inserta los valores de los campos previamente llenados a la base de datos.

Figura. 9: Agendar cita

Código Realizado

String sentenciaInsert = "INSERT INTO N_CITA VALUES(?,?,?,?,?)";

- Notas de ayuda:

  Captura cada uno de los valores que fueron colocados en los campos y se insertan en filas múltiples dentro de una tabla creada en la base de datos llamada N_CITA.

Nota: se puede validar que el paciente sea el correcto con los campos de RFC, y Nombre que se llena de manera automática.

3.6.1.2 Botón "Ver Registro"

En la Figura 10, se muestra un botón el cual tiene la acción de mandar a una pantalla nueva dentro de la cual se observa el registro de cada uno de los pacientes así como la fecha de las citas que fueron agendadas dentro de la base de datos.

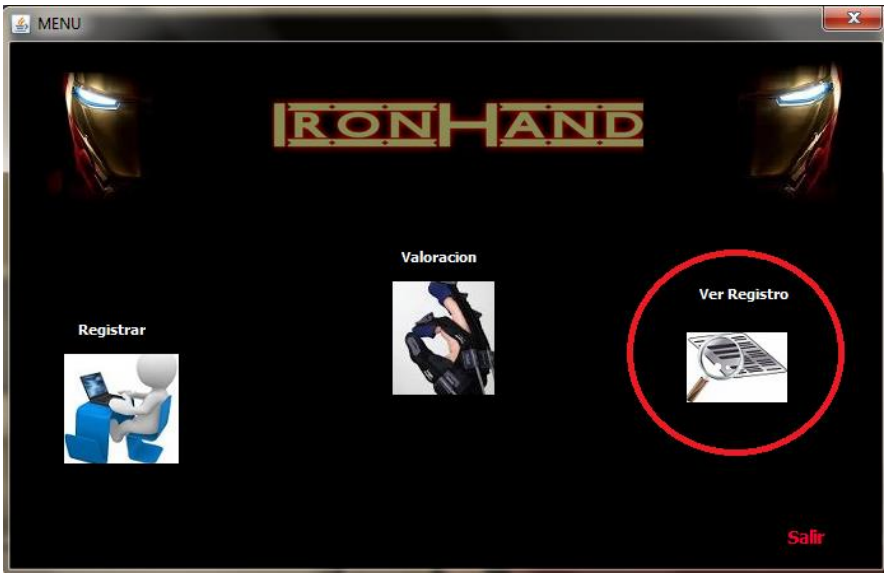


Figura. 10: Ver registró

Código Realizado

JDialogMostrarPaciente derecha=new JDialogMostrarPaciente(null, true);
derecha.setLocationRelativeTo(null);
derecha.setVisible(true);

- Notas de ayuda :
  Se carga en el centro de la pantalla nuevamente una ventana con cierta posición mediante un componente pasado como parámetro,

La Figura 11 muestra la lista de los pacientes que se guardaron dentro de la base de datos así como de las citas que le(s) pertenecen a cada uno de ellos.

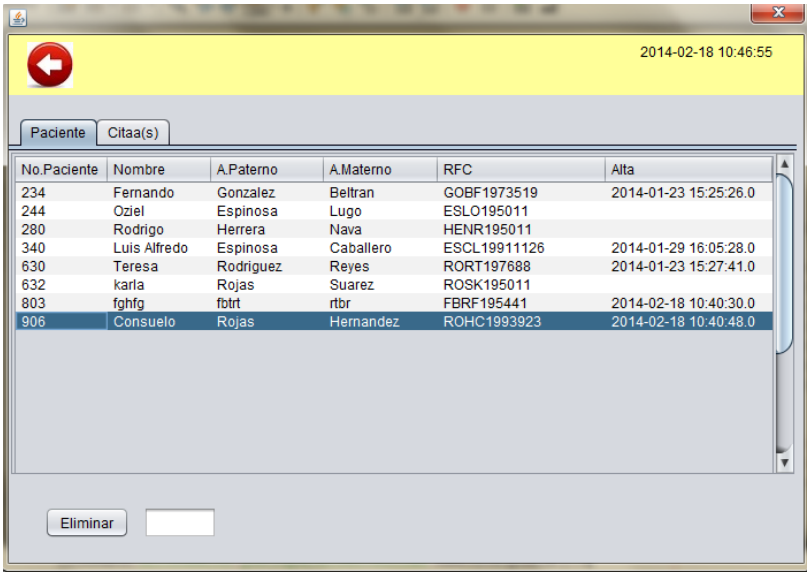


Figura. 11: Ver Registró paciente

La pestaña Paciente: contiene una tabla con todos los datos generales de cada paciente, así como la fecha en la cual fueron dados de alta para tener un mayor control del tiempo notando la evolución que ha transcurrido para adquirir nuevamente la movilidad de la mano.

Código Realizado

String elSQL = "SELECT * FROM PACIENTE";

- Notas de ayuda:
  Muestra los datos guardados en la base de datos de la tabla “PACIENTE”

El botón “Eliminar” borra de la base de datos el registro del paciente desde los datos generales hasta las citas agendadas

Código Realizado

String sentenciaDelete = "DELETE FROM PACIENTE WHERE id_paciente =?";

- Notas de ayuda de su funcion:
  En la Figura 12, se especifica el registro id_paciente dentro de las tablas “PACIENTE” y “CITA” marcandolo para su eliminación

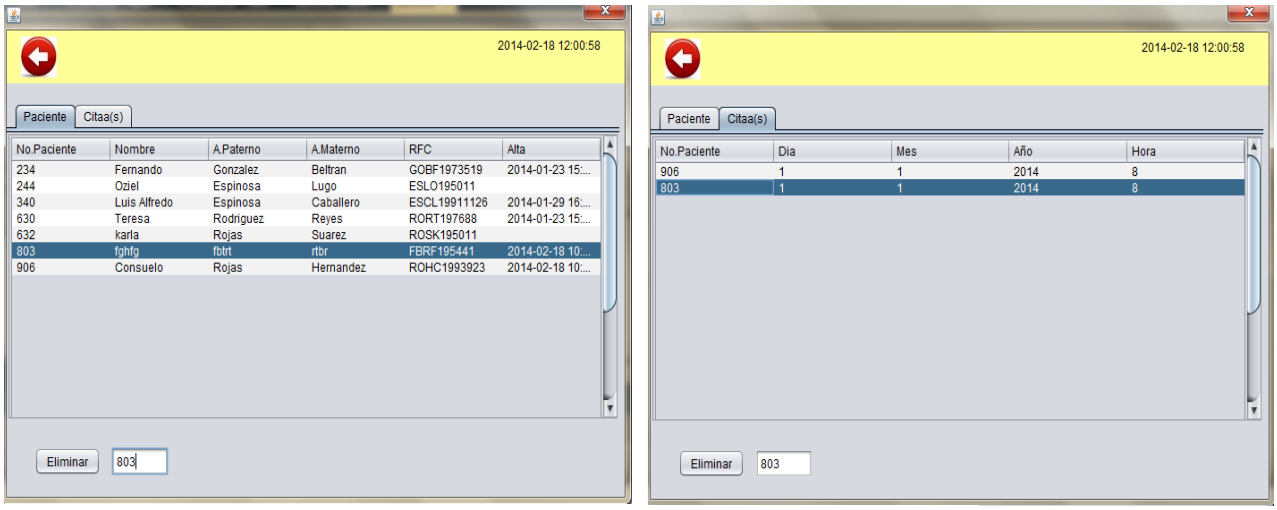


Figura. 12: Eliminar registró paciente/cita

La figura 13, Cita: contiene a cada paciente que ha agendado cada paciente puede tener diversas citas bajo mismo número de identificación.

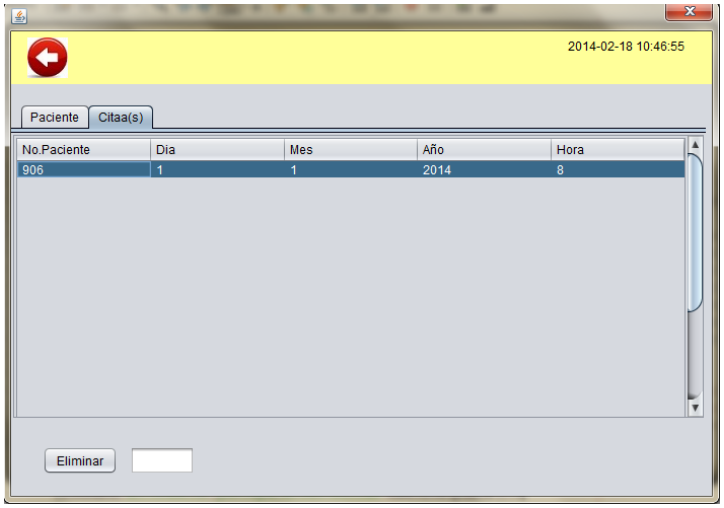


Figura. 13: Ver registro cita

Código Realizado

String elSQL = "SELECT * FROM N_CITA";

- Notas de ayuda:
  Se indica la tabla “N_CITA” de la cual obtendremos los datos.

3.6.1.3 Botón “Valoración”

Figura 14, muestra el botón que tiene mayor importancia dentro del desarrollo de este software ya que aquí se encuentra la creación de nuevas ruinas que son guardadas en la base de datos para después ser ejecutadas así como también la ejecución de las que ya están creadas:

Figura. 14: Valoración

Una vez seleccionado el botón de “Valoración” se despliega la siguiente pantalla (Figura 15), dentro de la cual se puede seleccionar mano derecha o mano izquierda a la cual se le aplicar la ratina ya establecida o crear una nueva.

Código Realizado

JDialog2Manos derecha=new JDialog2Manos(null, true);
derecha.setLocationRelativeTo(null);
derecha.setVisible(true);

- Notas de ayuda:
  Nuevamente se establece la carga una nueva ventana esta vez llamada "manos" que tiene cierta posición en este caso nuevamente es al centro de la pantalla.

Figura. 15: Seleccionar Rutina

En la Figura 16, se muestra como crear una nueva rutina la cual se llamara "Posterior Reanudación" como ejemplo.

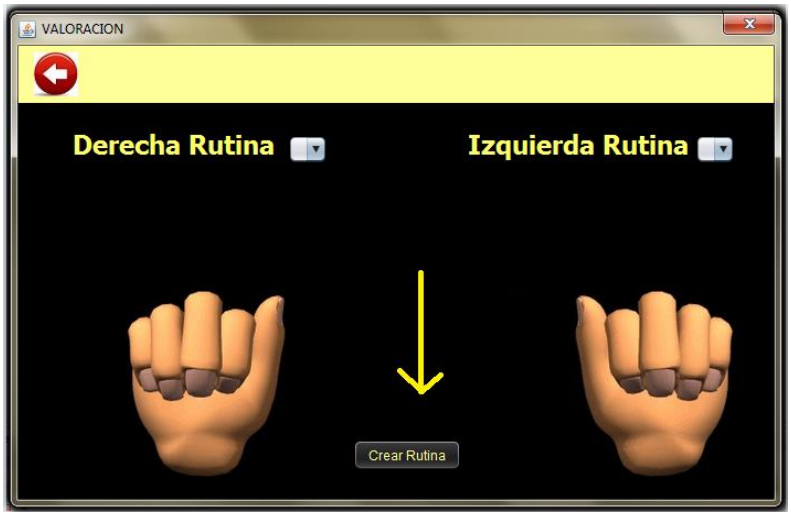


Figura. 16: Crear rutina

Una vez presionado el boton"Crear Rutina" se establece la carga de la ventana llamada "Rutina Nueva" (Figura 17), mostranda en el centro de la pantalla dentro de la cual se seleccionan los dedos de la mano, tiempo que durara la rutina, la velocidad que tendrán el movimiento de los dedos seleccionados así como la mano a la que se le asignaran estos parámetros.

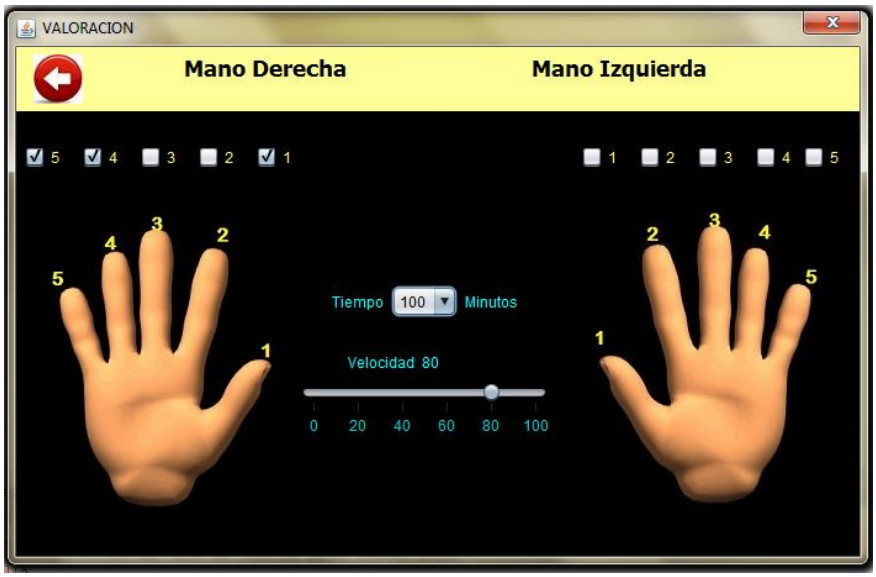


Figura. 17: Rutina Nueva

Código Realizado
Para la selección de los dedos se acupo el siguiente código:

if (jCheckBox.isSelected()==true) { }else { }

- Notas de ayuda:
  Verifica si un *JCheckBox* está seleccionado o no.

Para el tiempo de la rutina

jComboBox1.getSelectedItem();

- Notas de ayuda:
  Devuelve la opción seleccionada dentro del *JComboBox*.

Para la velocidad de la rutina

jSlider1.getValue();

- Notas de ayuda:
  Devuelve un valor entero con la posicion del *scrollla* seleccionada.

En la siguiente Figura 18, se asigna un numero de rutina para que sea identificada como un numero único dentro de la base de datos con esto evitamos generar ambigüedad.

Figura. 18: Numero de rutina

Código Realizado

JOptionPane.showInputDialog("Numero de Rutina: ");

- Notas de ayuda:
  Crea un cuadro de diálogo en el cual se pede insertar un nuemero de rutina.

Ahora se tiene la Figura 19, en la cual se asigna un nombre de rutina para que sea más fácil de identificar al momento de que se quiera llevar acabo.

Figura. 19: Nombre Rutina

Código Realizado

JOptionPane.showInputDialog("Nombre de Rutina: ");

- Notas de ayuda de su funcion:
  Nuevamente se crea un cuadro de diálogo en el que se pide el nombre de la rutina.

Con este último paso se concluye la creación de una nueva rutina (Figura 20) para una futura terapia quedando guardada dentro de la base y pudiendo ser utilizada para cualquier paciente.

Figura. 20: Rutina Guardada

En la Figura 21, se muestra en pantalla todas las rutinas existentes y creadas que se tienen disponibles.

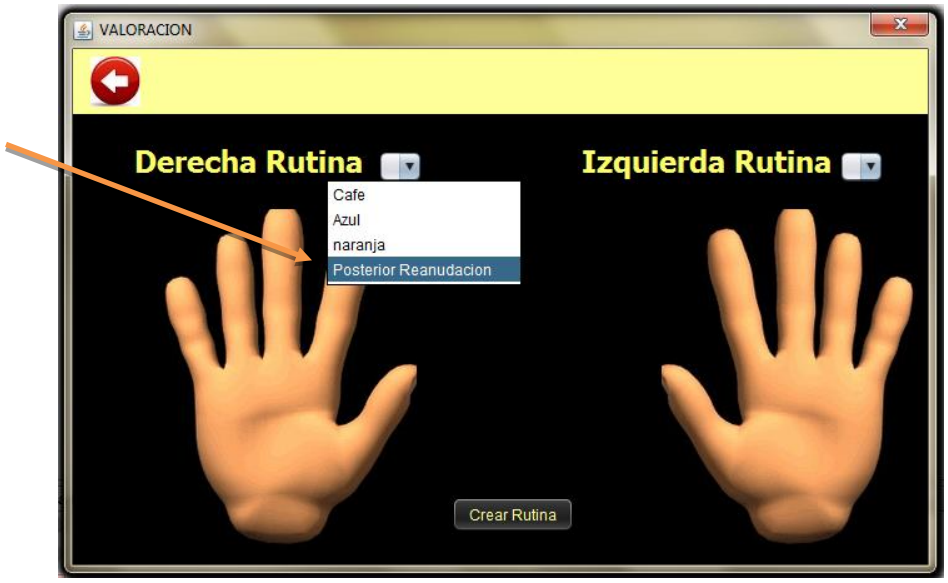


Figura. 21: Selección Rutina

Código Realizado

modeloCombo.addElement(rs.getObject());

- Notas de ayuda:
  Se devuelve el valor de la columna determinada (en este caso es “Nombre”) como un objeto de Java. El tipo del objeto de Java será predeterminado y corresponde al tipo SQL de la columna siendo ambos de tipo String.

En la Figura 22 ya se seleccionó la rutina “Posterior Reanudación” la cual se creo antes como ejemplo, en esta aparecen el campo dedos a mover, velocidad y tiempo configurados con los valores que pertenecen a ese nombre y esta guardados en la base de datos.

Figura. 22: Rutina Seleccionada

En la Figura 23 se aprecia el botón de “Iniciar” el cual se inicia el movimiento de los dedos seleccionados así como el conteo del cronometro de manera ascendente hasta llegar al límite que se configuro, el botón de “Pausa” y “Detener” están disponibles para cualquier ajuste o fallo dentro de la rutina.

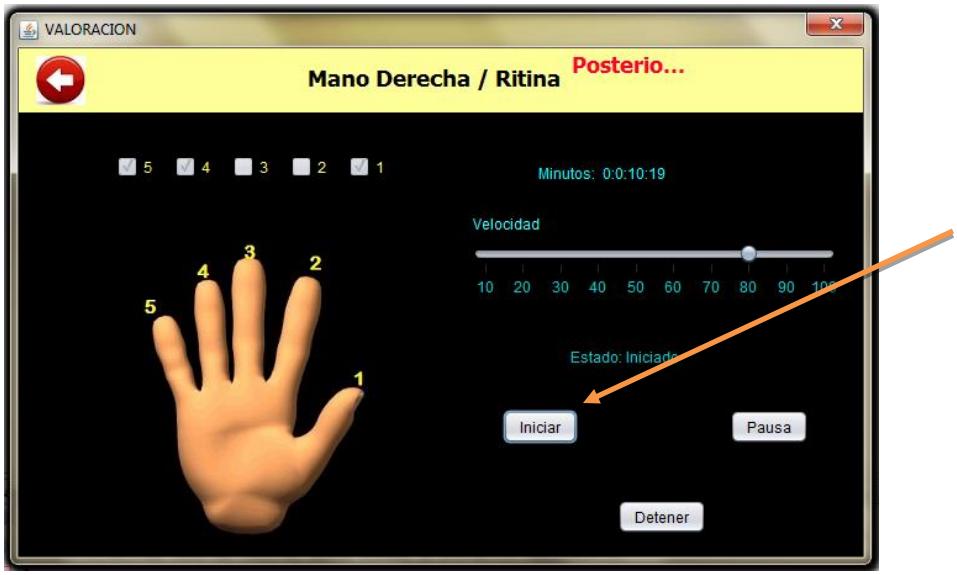


Figura. 23: Inicio Rutina

Código Realizado

hilo.start();

- Notas de ayuda:
  Inicia el conteo acendente del hilo definido por el método *run().*

El botón “Pausa” dentro de la Figura 24 suspende temporalmente el movimiento de los servomotores así como el conteo en el cronometro, dejando la opción de reanudar la rutina con el botón “Iniciar” o para por completo con el botón “Detener” según se presente el caso.

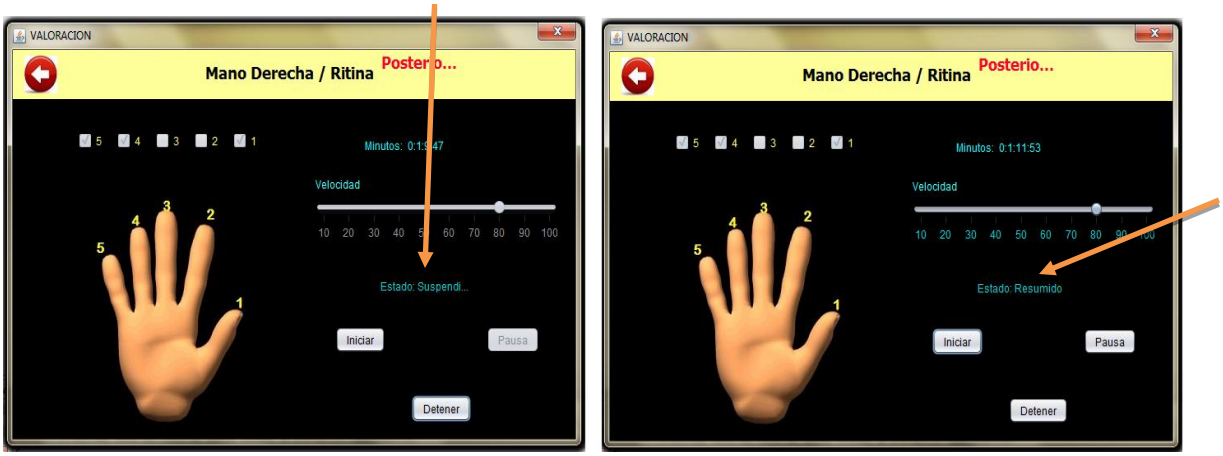


Figura. 24 Pausa/Reanudación rutina

Código Realizado

hilo.suspend();

- Notas de ayuda:
  Para temporalmente la ejecución del hilo para continuar nuevamente damos la instrucción *hilo.start().*

hilo.resume();

- Notas de ayuda:
  Reactiva el hilo suspendido

Por último el botón “Detener” que se marca en la Figura 25, apaga de manera automática los servomotores y ya no se cuenta con la opción de seguir adelante. Para continuar con esa rutina se tiene que volver a seleccionar desde el menú de manos para comenzar desde cero.

Figura. 25: Detener Rutina

Figura. 24: Pausa/Reanudación rutina

Código Realizado

hilo.stop();

- Notas de ayuda:
  Detiene la ejecución del hilo no importando consideración alguna.

3.6.1.4 Botón “Salir”

En la Figura 26 y Figura 27 se muestra el funcionamiento del botón “Salir” el cual solo pregunta al usuario si ha terminado cualquier actividad dentro del sistema ya sea “Registrar”, “Ver Registros” o “Valorar” al paciente para poder cerrarse.

Figura. 26: Salir del sistema

Figura. 27: Salir del sistema

Código Realizado

int seleccion = JOptionPane.showOptionDialog

- Notas de ayuda:
  Advierte bajo un cuadro de diálogo simple, que el sistema se cerrara por completo si presionamos la opción “si” de lo contrario bajo la opción “no” nos deja dentro del sistema para futuras operaciones.

3.6.2 Base de Datos

Dentro de esta base de datos se recopilar y organizar información en 3 tablas creadas con el nombre de “Paciente”, “Cita”, “Rutina” con la característica de agregar más datos, modificarlos, eliminarlos, organizarlo de distintas formas y compartir los datos de los usuarios mediante informes.

3.6.2.1 Figura 28 Modelo Entidad-Relación

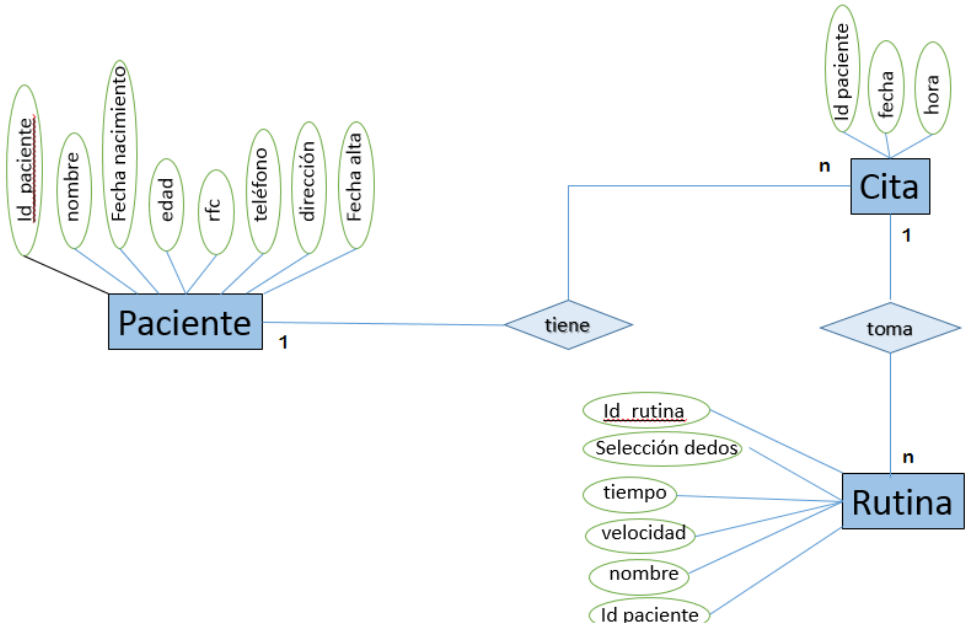


Figura. 28: Modelo Entidad-Relacion

3.6.2.2 Modelo Relacional

La figura 29 muestra el Modelo Relacional:

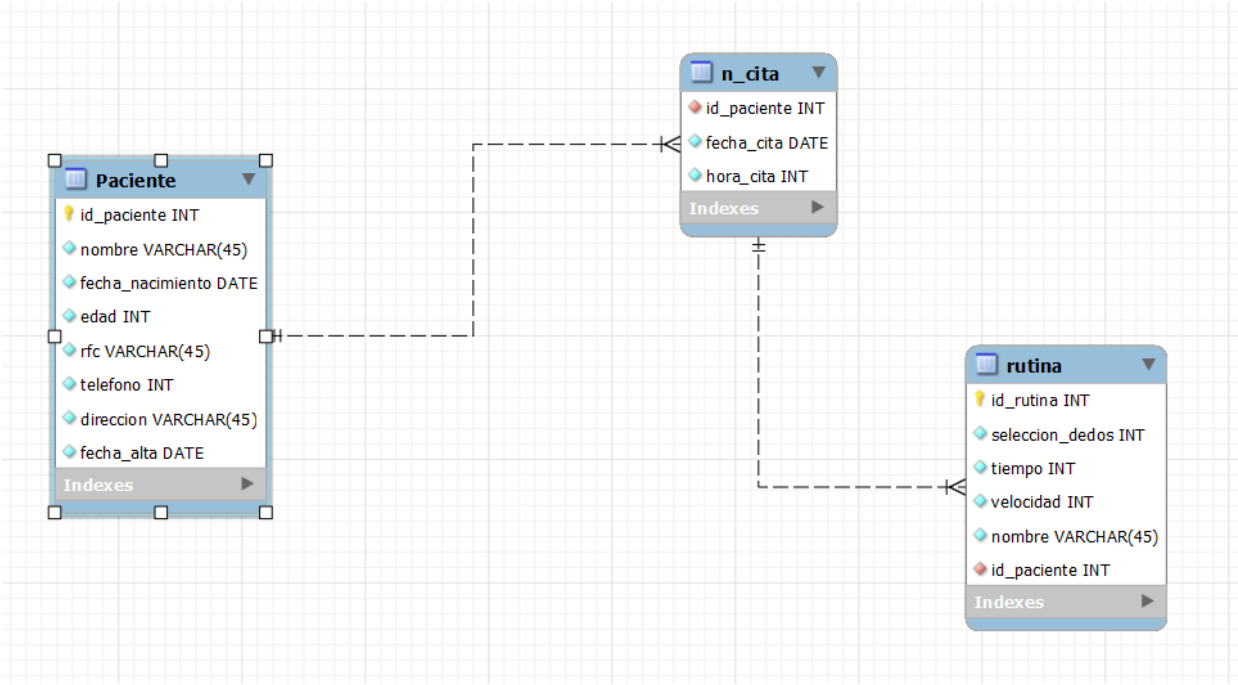


Figura. 29: Tablas Generadas

3.6.2.2.1 Normalización

1FN

Paciente (id paciente, nombre, fecha nacimiento, edad, rfc, teléfono, dirección, fecha alta)

n_cita (id paciente, fecha cita, hora cita)

Rutina (id rutina, selección dedos, tiempo, velocidad, nombre, id paciente)

2FN

Paciente (id paciente, a.paterno, a.materno, nombre, fecha nacimiento, edad, rfc, teléfono, Calle, Colonia, Municipio, Estado, fecha alta)

n_cita (id paciente, fecha cita, hora cita)

Rutina (id rutina, selección dedos, tiempo, velocidad, nombre, id paciente)

3FN

Paciente (id paciente, a.paterno, a.materno, nombre, fecha nacimiento, edad, rfc, teléfono, Numero, Calle, C.P, Estado, fecha alta)

C.P (C.P, Colonia)

Estado (Nombre Estado, Municipio)

n_cita (id paciente, fecha cita, hora cita)

Rutina (id rutina, selección dedos, tiempo, velocidad, nombre, id paciente)

3.6.2.2.2 Tablas Generadas

Paciente

Id paciente	a.patern	a.matern	nombre	fecha_na	edad	rfc	teléfono
	numero	calle	C.P	Estdo	Fecha alta		

Tabla. 1 Paciente

C.P

C.P	Colonia

Tabla. 2 Código Postal

Estado

Nombre Estado	Municipio

Tabla. 3 Estado

n_cita

Id paciente	fecha cita	hora cita

Tabla. 4 Nueva Cita

Rutina

id rutina	selección dedos	tiempo	velocidad	nombre	id paciente

Tabla. 5 Rutina

IV ARDUINO

4.1 Introducción al Arduino

¿Qué es Arduino?

- Arduino es una plataforma open-hardware basada en una sencilla placa con entradas y salidas (E/S), analógicas y digitales, y en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Su corazón es el chip Atmega8, un chip sencillo y de bajo coste que permite el desarrollo de múltiples diseños.

  Al ser open-hardware tanto su diseño como su distribución es libre. Es decir, puede utilizarse libremente para desarrollar cualquier tipo de proyecto sin tener que adquirir ningún tipo de licencia.

¿Para qué puedo utilizar Arduino?

- Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede conectarse a un PC a través del puerto serie utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación.

  Asimismo, su sencillez y su bajo coste, recomiendan su uso como elemento de aprendizaje e iniciación en el mundo de la electrónica digital.

4.2 Elementos necesarios

4.2.1 El Hardware de Arduino

4.2.1.1 Placa Arduino

Lo primero que se necesita es una placa Arduino. Existen varios modelos, e incluso nos podemos construir nuestra propia placa. La placa Arduino es “open hardware”, lo que quiere decir que su diseño es de libre distribución y utilización. En la página web se proporcionan todos los esquemas necesarios para integrar nuestra propia placa.

No obstante y para iniciarse, se recomienda adquirir uno de los modelos que se distribuyen a través de la web de Arduino, en concreto las placas serie (RS232) y Figura 30 USB. Si bien el modelo serie tenemos que soldar todos los elementos (resistencias, condensadores, etc.), el modelo USB ya se encuentra lista para usar. Queda a nuestra lección decidirnos por uno u otro modelo.

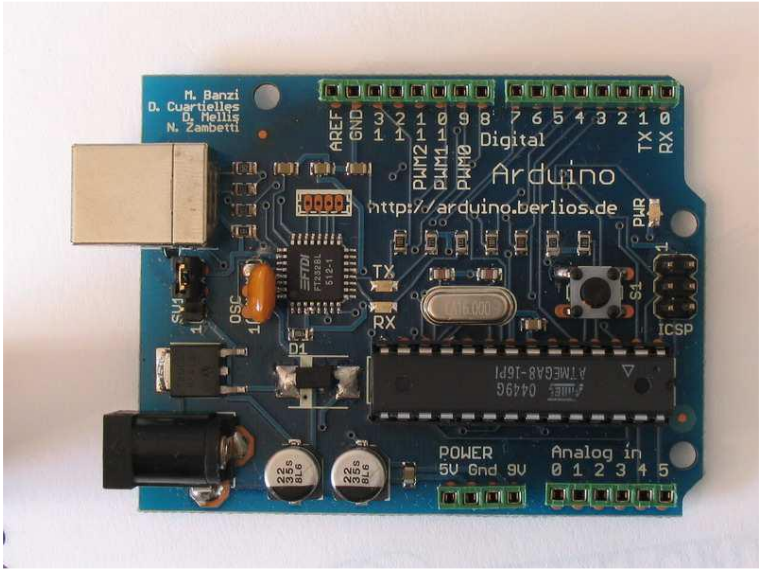


Figura. 30: Placa USB

4.2.1.2 Cable de comunicación (Serie/USB)

En función del modelo de placa que hayamos adquirido tendremos que elegir un cable serie o USB.

El cable serie debe tener en sus extremos dos conectores de tipo DB-9. Uno macho (para conectar la placa) y otro hembra (para conectar al PC). Es muy importante comprobar que el cable serie NO sea del tipo “NULL MODEM” ya que no nos sirve.

El cable USB debe ser tal y como se muestra en la Figura 31. Con un conector tipo A (para conectar al PC) y otro tipo B (para conectar a la placa) en sus extremos No hay que equivocarlo con el cable mini-USB que habitualmente se

utiliza con dispositivos más pequeños como cámaras de fotos y lectores de tarjetas.

Figura. 31: Cable USB

4.2.1.3 Fuente de alimentación

Si bien en el caso de la placa USB no es preciso utilizar una fuente de alimentación externa como se muestra en la Figura 33, ya que el propio cable USB la proporciona, en el caso de la placa serie es necesario disponer de una fuente externa.

Se puede utilizar una fuente de alimentación de corriente continua o una pila/batería con el conector apropiado. Se recomienda no obstante el uso de la primera ya que no tenemos que estar pendientes de sustituir las pilas en caso de que se queden sin carga.

En ambos casos el voltaje de la fuente puede ser de entre 6 y 25 voltios, y la polaridad del conector debe ser como se indica en la Figura 32.

Figura. 32: Polaridad

Figura. 33: Fuente de Alimentación

Un tema muy importante a tener en cuenta es que en la placa conectada por USB se nos ofrece la posibilidad de alimentar la placa a través de una fuente de alimentación externa. En la Figura 34 se muestra la posición en la que debe estar el “jumper” para que la alimentación de la placa se realice desde el cable USB. Si se coloca de en la otra posición posible la placa tomará la alimentación de la fuente externa.

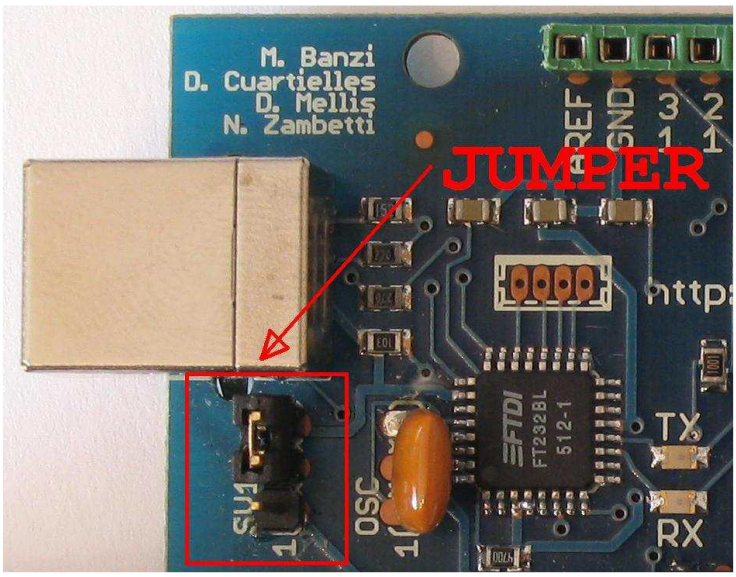


Figura. 34: Alimentación a través del cable USB

4.3 ¿Con qué elementos podemos interactuar?

La placa Arduino está basada en el chip Atmega8 o Atmega168. Alrededor de uno de estos se monta toda la circuitería necesaria para poder sacarle el máximo partido. La figura 35 muestra la descripción de los componentes principales de la placa Arduino.

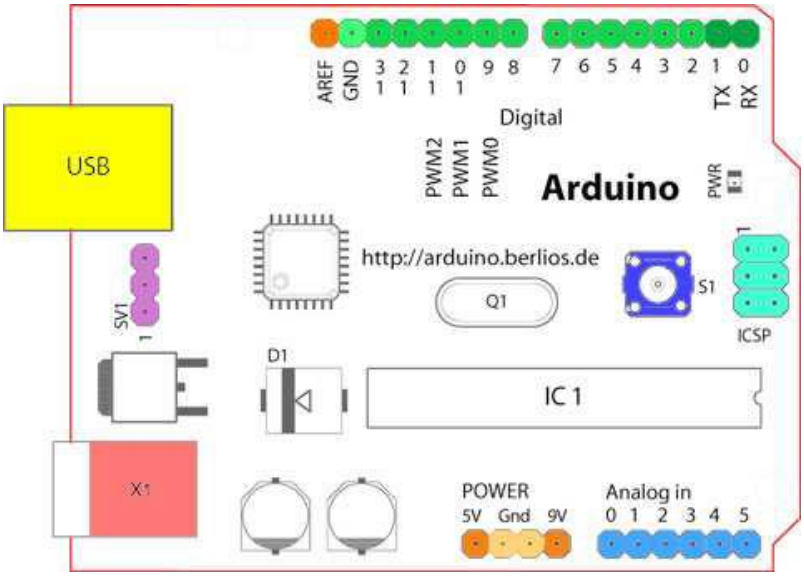


Figura. 35: Descripción de componentes Arduino

Comenzando en el sentido de las agujas del reloj desde el centro de la parte superior:

- Pin de referencia analógica (naranja)
- Señal de tierra digital (verde claro)
- Pines digitales 3-13 (verde)
- Pines digitales 1-2 / entrada y salida del puerto serie: TX/RX (verde oscuro)
- Botón de reset (azul oscuro)
- Entrada del circuito del programador serie (azul turquesa)
- Pines de entrada analógica 0-5 (azul claro)
- Pines de alimentación y tierra (fuerza: naranja, tierra: naranja claro)
- Entrada de la fuente de alimentación externa (9-12V DC) – X1 (rosa)
- Conmuta entre fuente de alimentación externa o alimentación a través del puerto USB – SV1 (violeta)
- Puerto USB (amarillo)

4.4 Software

4.4.1 Entorno de desarrollo

Para programar la placa es necesario descargarse de la página web de Arduino (http://www.arduino.cc/en/Main/Software) el entorno de desarrollo (IDE). Se dispone de versiones para Windows y para MAC, así como las fuentes para compilarlas en LINUX. La Figura 36 Muestra la interfaz gráfica de la plataforma Arduino ejecutada en un Sistema Operativo Windows Seven.

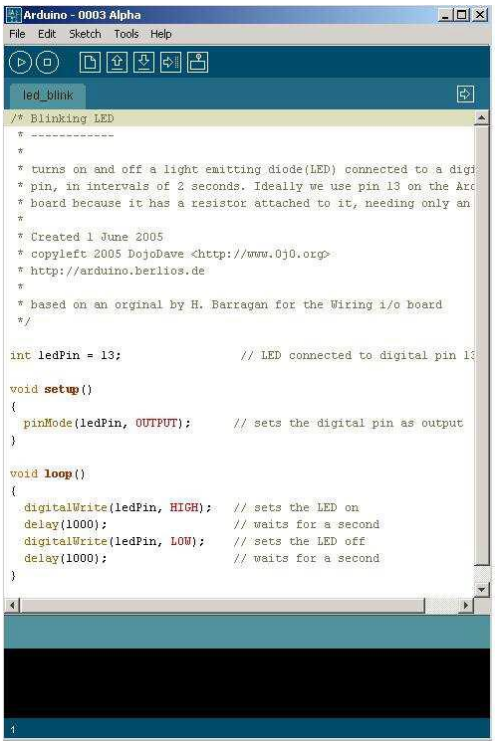


Figura. 36: Entorno de desarrollo

4.4.2 Driver

En caso disponer de una placa con USB es necesario instalar los drivers FTDI. Estos drivers vienen incluidos en el paquete de Arduino mencionado en el apartado anterior. Existen en la web versiones para distintos sistemas operativos.

4.5 Instalación de Arduino en Windows

A continuación se muestran los pasos básicos para instalar Arduino en Windows. Esta guía se ha realizado utilizando Windows 7 y la versión 1.0.3 de la IDE de Arduino.

NOTA - No conectar la placa todavía.

NOTA - Para una explicación más detallada sobre como instalar la IDE de Arduino se recomienda visitar los siguientes enlaces:

- Instalación en Windows http://www.arduino.cc/es/Software/Windows
- Instalación en Linux (Ubuntu) http://www.arduino.cc/es/Software/Linux

Pasos a seguir

1.- Descargue la versión de la IDE de Arduino.

2.- Descomprimir el fichero en el directorio/carpeta raíz (c:\) manteniendo la estructura original.

3.- De entre todas las carpetas que se han creado en C:\arduino, cabe destacar las siguientes:

· C:\arduino\bootloader --> Contiene el software necesario para cargar el firmware en el chip Atmega8, necesario para trabajar con Arduino.

· C:\arduino\drivers --> Contiene los drivers necesarios para hacer funcionar la placa Arduino con nuestro PC: giveio.zip y FTDI USB Drivers.zip.

Al conectar la placa USB y se abrirá automáticamente el "Asistente para nuevo hardware encontrado" ver Figura 37:

Figura. 37: Asistente de Instalación

Selecciona “No por el momento” y pulsa “Siguiente” ver Figura 38.

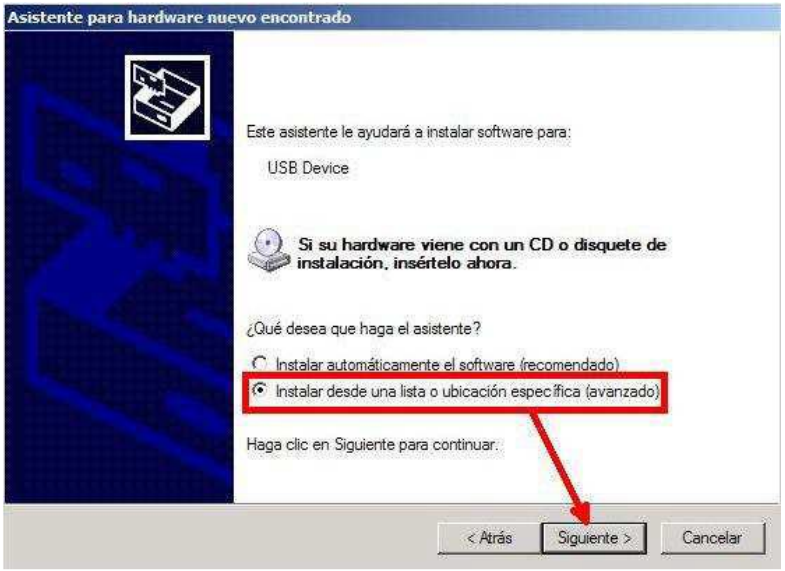


Figura. 38: Asistente de instalación

En la Figura 39 se selecciona “Instalar desde una lista o ubicación especifica (avanzado)” y se pulsa “siguiente”.

Figura. 39: Asistente de instalación

Una vez selecciono “Se buscar el controlador más adecuado” y se pulsa “Examinar”. Ya seleccionada la carpeta temporal donde se descomprimieron los drivers se pulsa “Siguiente” y con eso se concluye la instalación del Software.

4.5.1 Configuración de las comunicaciones

Lo primero que se tiene que hacer es configurar las comunicaciones entre la placa Arduino y el PC. Para ello se debe abrir en el menú “Tools” las opciones “Serial Port” y “Serial Monitor Baud Rate”.

En la primera de las dos opciones se debe seleccionar el puerto serie al que está conectada nuestra placa como se muestra en la Figura 40. En Windows el puerto será COM1 o COM2 para la placa serie, COM3, COM4 para la placa USB (o para la placa serie conectada mediante un adaptador serie-USB).

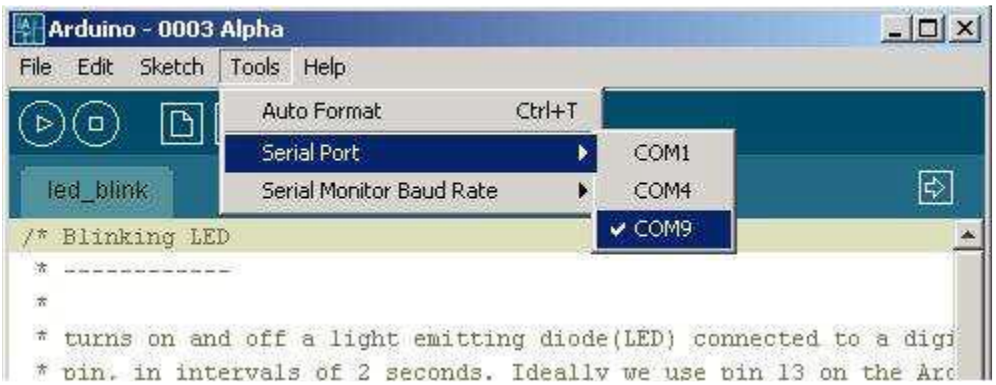


Figura. 40: Configuración puerto serie

En Windows, si se desconoce el puerto al que está conectado la placa se puede descubrirlo a través del "Administrador de dispositivos" como se ilustra en la Figura 41.

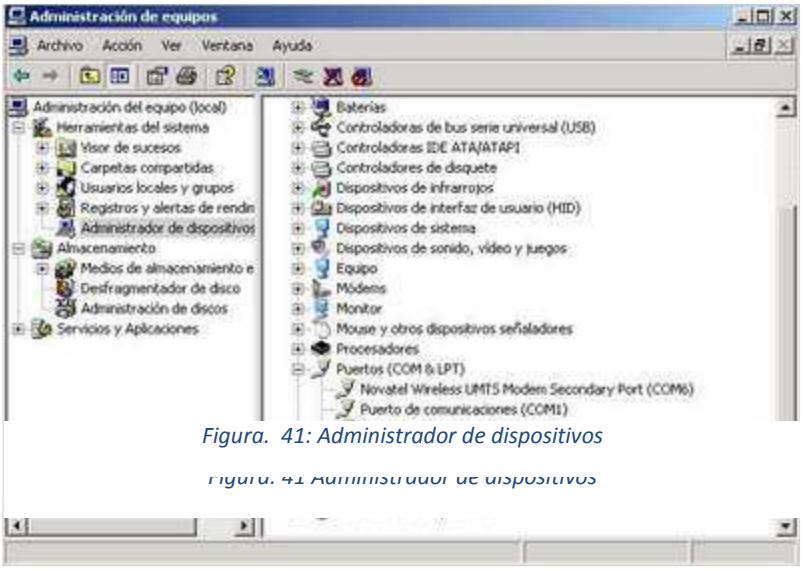


Figura. 41: Administrador de dispositivos

Figura. 41 Administrador de dispositivos

En la Figura 42 se configurar la velocidad a la que la placa y el PC se comunican. Esto lo se hace desde el menú "Serial Monitor Baud Rate".

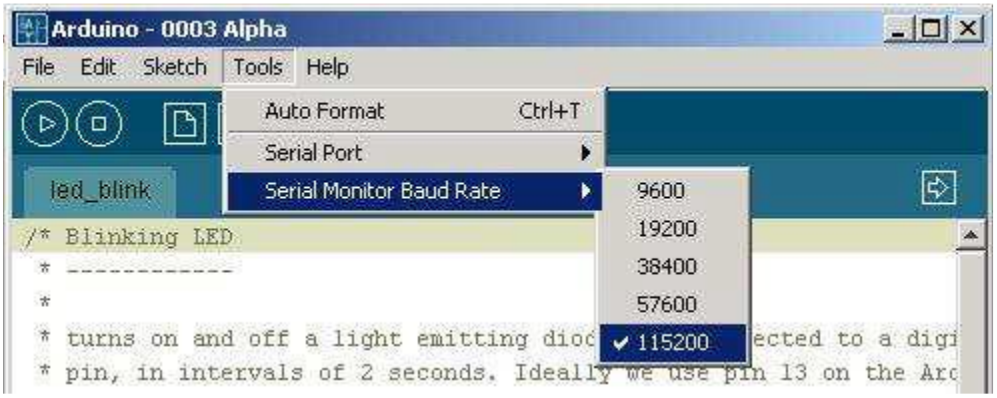


Figura. 42: Configuración de velocidad

Ya sólo queda ejecutar el fichero Arduino.exe para abrir la interfaz. Una vez abierta sólo se tiene que configurar el puerto USB (Figura 43) al que se conectara la placa para empezar a trabajar.

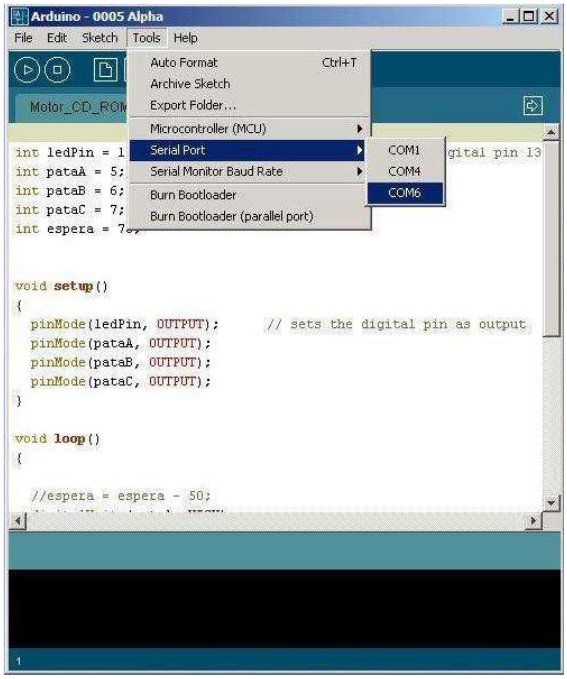


Figura. 43: Configuración de puerto USB

Una vez descargado, para comenzar a trabajar con el entorno de desarrollo en Windows, tan sólo es necesario descomprimir el contenido del fichero comprimido en una carpeta de nuestro PC. Una vez descomprimido tan sólo es necesario ejecutar el fichero “Arduino.EXE”.

4.6 Desarrollo del Software

Dentro de Java se realiza el siguiente código:

SerialPort serialPort;

- Notas de ayuda:

  Es una interfaz de comunicaciones entre ordenadores y periféricos el cual envía y recibe información BIT por BIT

private final String PORT_NAME = "COM4";

- Notas de ayuda:

  Obtiene o establece el puerto de comunicaciones, incluidos por lo menos todos los puertos COM disponibles.

private static final int TIME_OUT = 2000;

- Notas de ayuda:

  Tiempo que tarda en responder

private static final int DATA_RATE = 9600;

- Notas de ayuda:

  Obtiene o establece la longitud estándar de los datos por byte

Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

- Notas de ayuda:

Lista que posee todos los puertos del sistema

CommPortIdentifier actualPortId = (CommPortIdentifier)portEnum.nextElement();

- Notas de ayuda:
  Obtiene un elemento de la lista

serialPort = (SerialPort) PortId.open(this.getClass().getName(), TIME_OUT);

- Notas de ayuda:
  Se abre la comunicación con el puerto serial

output.write(Datos.getBytes());

- Notas de ayuda:
  Escribe el comando de salida

4.7 Copiado del programa a la placa Arduino

En la Figura 44, se comprueba que el código fuente es el correcto. Para ello se pulsa el botón de verificación de código que tiene forma de triángulo inclinado 90 grados.

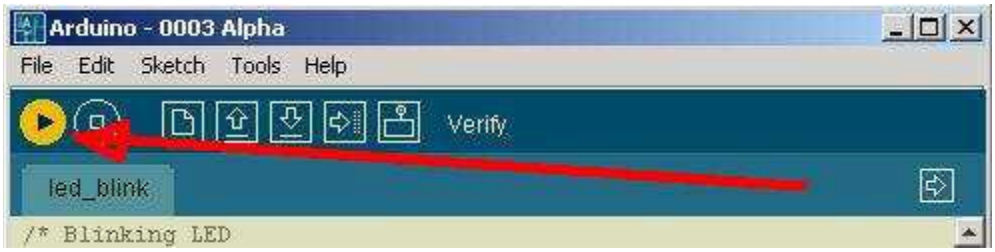


Figura. 44: Verificando el código fuente

Si todo va bien deberá aparecer un mensaje en la parte inferior de la interfaz indicando “Done compiling” como se puede observar en la Figura 45.

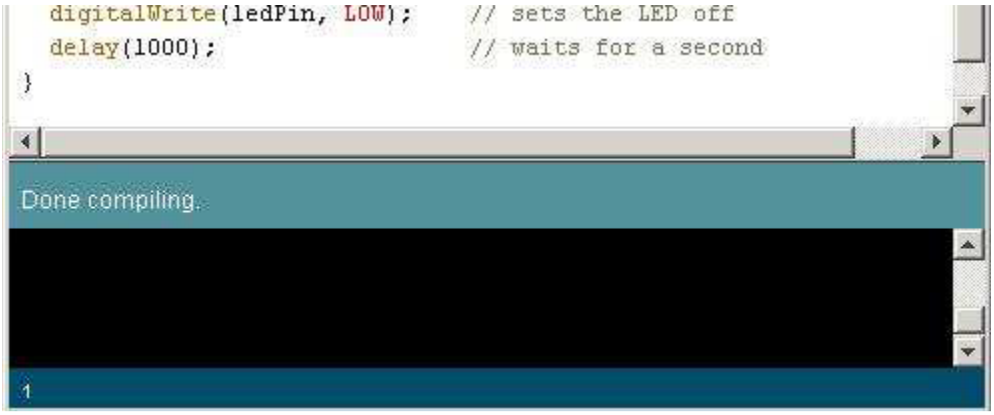


Figura. 45: Comprobación correcta

En la figura 46 se pulsa el botón cargar identificado con el símbolo de una flecha.

Figura. 46: Subiendo el programa a la placa

Durante la carga del programa, en la placa USB, se encenderán los LED que indican que se están enviando y recibiendo información por el puerto serie: TX/RX como se puede apreciar en la Figura 47.

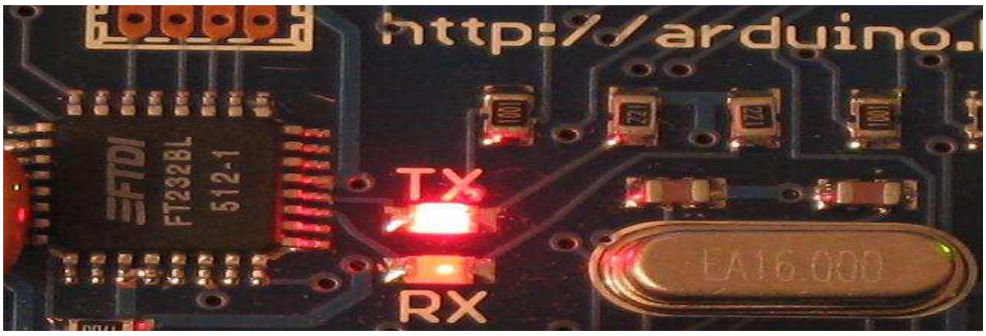


Figura. 47: Subiendo el programa

Realizado esto correctamente aparece un mensaje como el que se muestra en la Figura 48:

Figura. 48: Programa cargado correctamente

V PROTOTIPO

5.1 Material

5.1.1 Servomotores

En esencia un Servo Motor es un motor común de Corriente Directa (CD) al cual se le ha acoplado una transmisión o moto reductor (Juego de Engranes que reducen la velocidad y aumentan el torque) y un poco de electrónica para el control de su posición. Por lo anterior podemos hacer referencia a la siguiente expresión:

Motor CD + Moto Reductor + Circuito de Control = Servo Motor Figura 49.

Figura. 49: Servomotor

Los Servo Motores o Servos por lo general pueden realizar movimientos rotatorios de su eje en el orden de 0 ° a 180° grados, pudiendo quedarse fijos en cualquiera de estos ángulos. Existen también motores de este mismo tipo que tienen la cualidad de poder desplazar su eje en los 360° ya sea en sentido horario o anti horario y a una velocidad definida por el usuario.

Para realizar la tarea de desplazamiento rotatorio los Servo Motores se valen de una señal de control PWM (Modulación por Ancho de Pulso), este tipo de señales consisten en modificar el ciclo de trabajo de una señal periódica haciendo que una

parte del tiempo del periodo (T) de la señal se encuentre a un nivel bajo y otra a un nivel alto.

La Figura 50 muestra las señales admitidas para controlar servo motores por lo regular oscilan entre periodos de los 10 a 20 mili-segundos o en términos de frecuencia, entre los 100 a 50 Hz. La siguiente imagen es una representación de una señal PWM de aproximadamente un 25% de ciclo de trabajo, ya que podemos observar que del tiempo "0" al tiempo "T" la señal solo estuvo a un nivel alto de voltaje aproximadamente un 25% del tiempo, el otro 75% estuvo a un nivel bajo, y después de eso se vuelve a repetir, por esto último es que se llama una señal periódica:

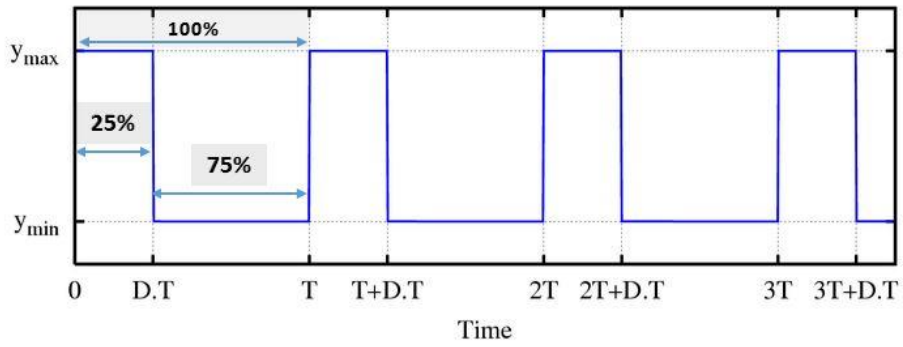


Figura. 50 Señal del servomotor

Hemos mencionado anteriormente que los servomotores hacen uso de una señal PWM para modificar su posición, y ahora te explicare que características debe tener dicha señal.

En primer lugar debe repetirse entre cada 10 y 20 mili-segundos (100-50 Hz).

Dependiendo la posición que se desee mover el servo, el pulso a nivel alto debe durar cierto tiempo, como se muestra en la Figura 51 y se describe a continuación:

Si lo que queremos es que nuestro Servo, se mueva a una posición de 0° grados, debemos mandar una señal con una duración de pulso alto entre los (0.5 y 1 mili-segundos), que Considerando una duración de pulso alto de 1ms y un periodo de señal de 20ms, tendríamos una señal de un 5% de Ciclo de Trabajo

Figura. 51 5% del ciclo de trabjo

Para una posición de 90° grados, debemos mandar una señal con un ancho de pulso de 1.5ms, o en otras palabras de un 7.5% de Ciclo de Trabajo. Ver Figura 52.

Figura. 52 7.5% de ciclo de trabajo

Para una posición de 180° grados, debemos mandar una señal con un ancho de pulso de 2.5 ms, o en otras palabras de un 12.5% de Ciclo de Trabajo. Ver Figura 53.

Figura. 53 12.5% ciclo de trabajo

5.1.2 Polea

La Figura 54 muestra el funcionamiento básico de una polea, una máquina simple, o un dispositivo mecánico de tracción, que sirve para transmitir una fuerza. Además, formando conjuntos, aparejos o polipastos, sirve para reducir la magnitud de la fuerza necesaria para mover un peso.

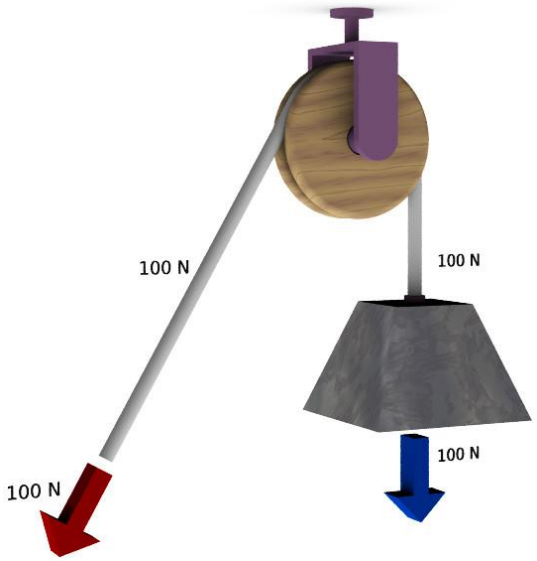


Figura. 54 Polea

5.1.3 Base metálica

Figura 55.

Figura. 55 Base metalica

5.1.4 Guante

Prenda, cuya finalidad es la de sujetar los dedos del paciente que se vayan a manipular. Figura 56.

Figura. 56 Guante

5.2 Costos

Material	Costo Unitario	Piezas	Sub-total
Placa Arduino	$570.00	1	$570.00
Servomotores	$250.00	6	$1500.00
Polea metálica	$5.00	45	$225.00
Base metálica	$400.00	1	$400.00
Guante	$10.00	2	$20.00
Total			**$2715.00**

Tabla. 6 Costos

5.3 Armado/Ensamble

En la Figura 57 se observa claramente un análisis para determinar de qué manera podría funcionar mejor el diseño tomando en cuenta todas aquellas variaciones que podemos encontrar en los pacientes ya sean tamaño, grosor y peso de la mano así como dedos.

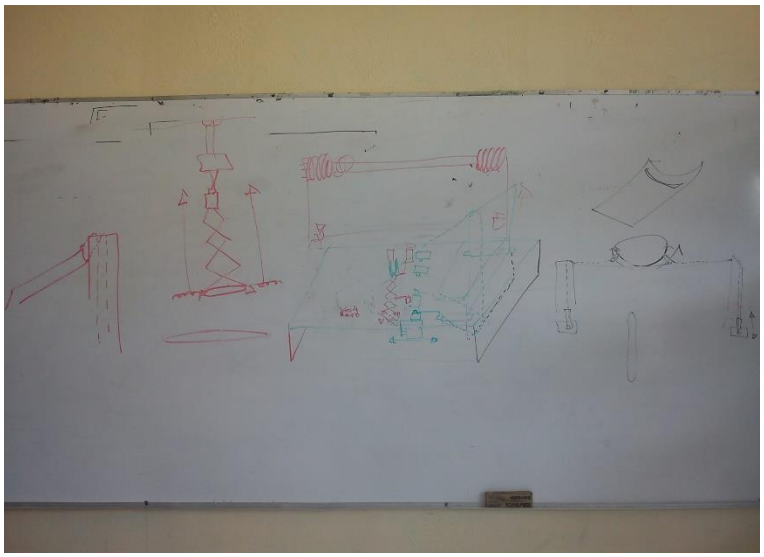


Figura. 57 Diseño prototipo

Una vez realizado el análisis se dibujó el primer boceto para determinar las dimensiones de la base así como cuál sería el mejor lugar donde se pudieran colocar las poleas para obtener un mayor grado de movimiento en los dedos y una mejor posición del brazo del paciente como se muestra en la Figura 58.

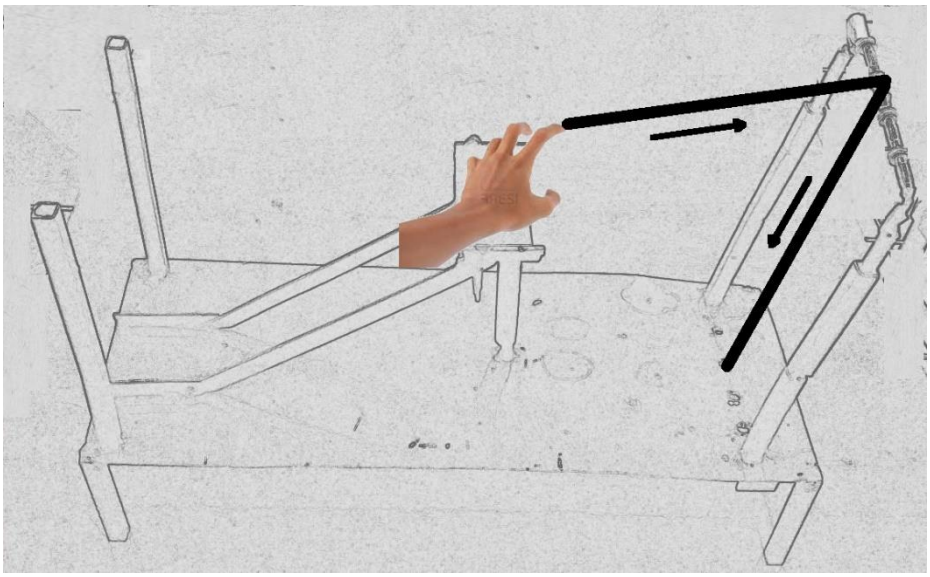


Figura. 58 Boceto dimenciones

La Figura 59 muestra la parte superior del prototipo donde se pondrá el brazo las poleas y el tamaño de la base.

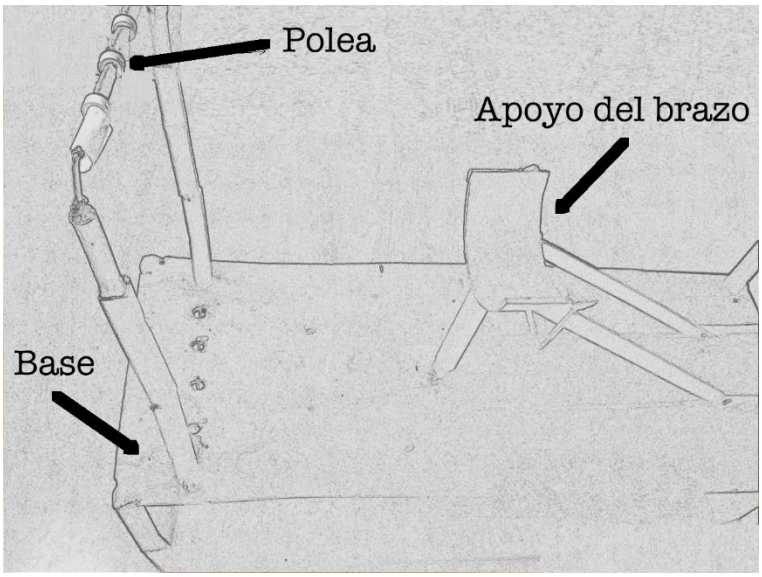


Figura. 59 Boceto parte superior

Dentro de la parte inferior (Figura 60) es donde se realizaran todas las conexiones pertinentes para que se puedan llevar acabo las rutinas esto quiere decir que es la parte donde se colocaran los servomotores y el cableado.

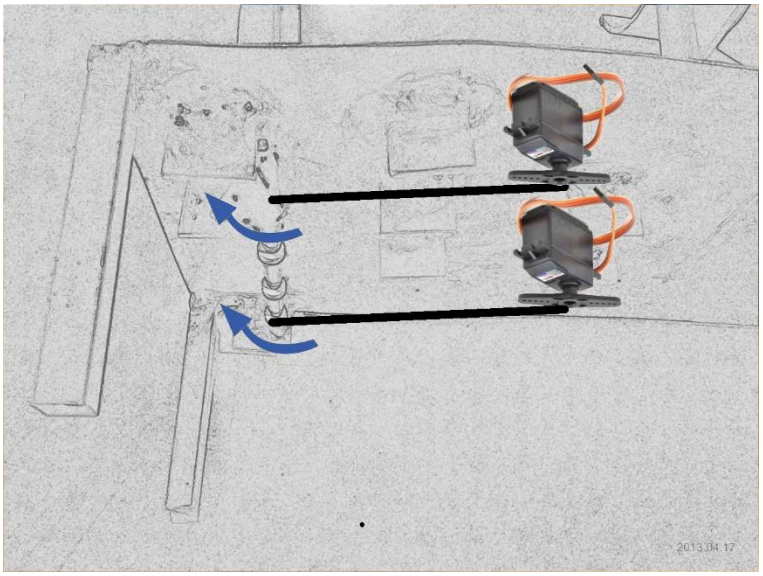


Figura. 60 Boceto parte inferior

Una vez terminado el boceto así como una gran lluvia de ideas que se tuvo para la construcción del prototipo, la Figura 61 muestra el primer resultado real y palpable del mismo.

Figura. 61 Primer prototipo fase 1

La Figura 62 muestra la primera prueba de la colocación de la mano donde se puede observar que la base y las dimensiones son aptas para llevar acabo la continuación con las modificaciones pertinentes en las cuales se procede a poner poleas y servomotores.

Figura. 62 Prueba primer prototipo fase 1

Para la colocación de las poleas esta fue una de las ideas en la cual se pretende pasar el hilo que va del dedo del paciente al servomotor logrando así el movimiento deseado. Figura 63.

Figura. 63 Primer prototipo fase 2

Para la colocación de los servomotores se realizó la modificación de alzar la base 10 cm (Figura 64), con esta distancia se consigue el espacio necesario para la conexión y cableado.

Figura. 64 Primer prototipo fase 3

Una vez probado el correcto funcionamiento del primer prototipo se construyó un segundo prototipo bajo el mismo patrón pero este siendo más robusto y sofisticado como por ejemplo se le adaptó una base para el brazo con una forma más adecuada y cómoda para colocar el brazo esta a su vez puede ser colocada con diferente inclinación. Figura 65.

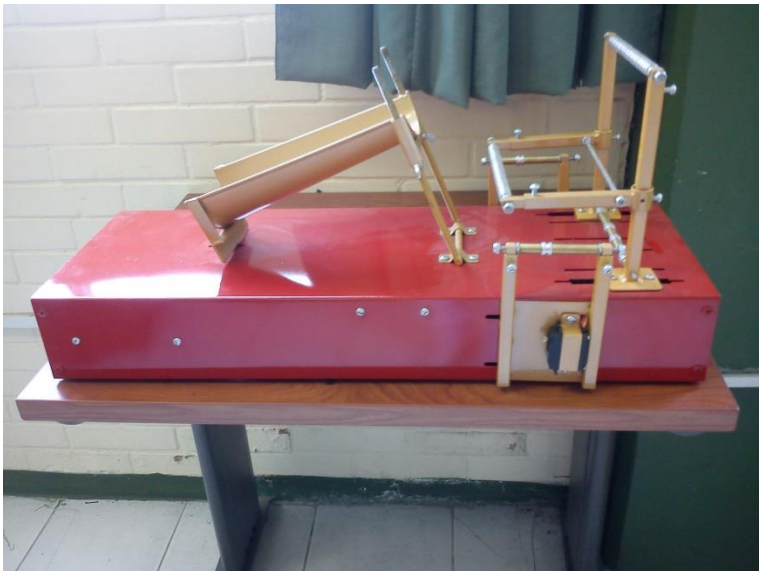


Figura. 65 Prototipo base

Para adaptar la distancia que se tiene entre los rieles que sostienen las poleas y los dedos del paciente se hicieron corredizas la estructura con esto se gana que cualquier persona con cualquier tipo de mano y brazo sea atendida.

La Figura 66 muestra cómo se colocó un mayor número de poleas en toda la parte superior de la estructura por donde pasan los hilos que jalan el servomotor para que la guía no tenga ningún problema.

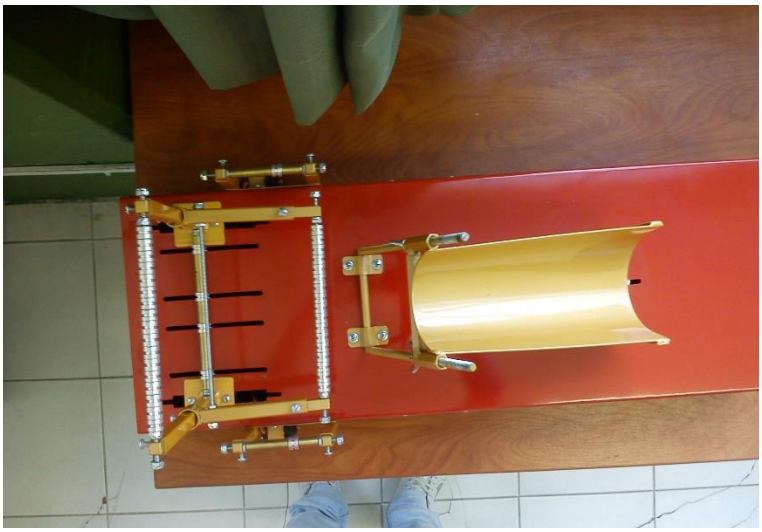


Figura. 66 Prototipo poleas

Se adaptó de cada costado de la base un servomotor (Figura 67) para lo que es el movimiento del dedo índice y de la muñeca, esta base al igual que la superior es corrediza con lo cual se adapta fácilmente.

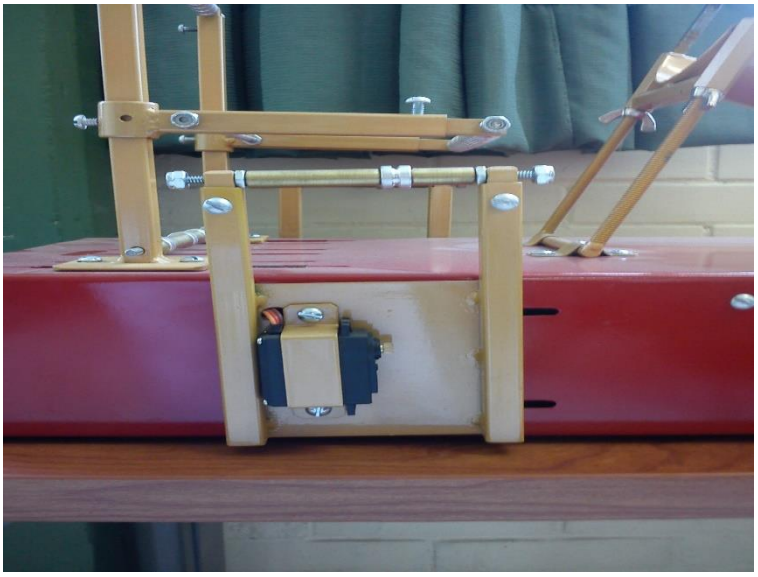


Figura. 67 Prototipo servomotor lateral

La figura 68 muestra la parte inferior donde se encuentran las conexiones, se pude apreciar que fueron colocados de manera cautelosa para evitar cualquier rose entre ellos pudiendo ocasionar algún problema en el funcionamiento, así como de tal manera que el movimiento sea más libre y con un mayor número de grados.

Figura. 68 Prototipo colocación servomotores

Una parte muy importante es el guante donde el paciente colocara la mano y dedos los cuales serán movidos, en este primer intento el problema la falta de adherencia a los dedos lo cual lleva a un mayor contacto entre ellos y de esta manera se pierde movimiento y eficacia en las rutinas. Figura 69.

Figura. 69 Primer guante

Para lograr una mayor estabilidad al movimiento de los dedos se creó este segundo guante (Figura 70) en el cual ya se consiguió que se adhiera al dedo de cualquier paciente, en l parte extrema donde se coloca el dedo se pondrá el hilo el cual jalara el servomotor.

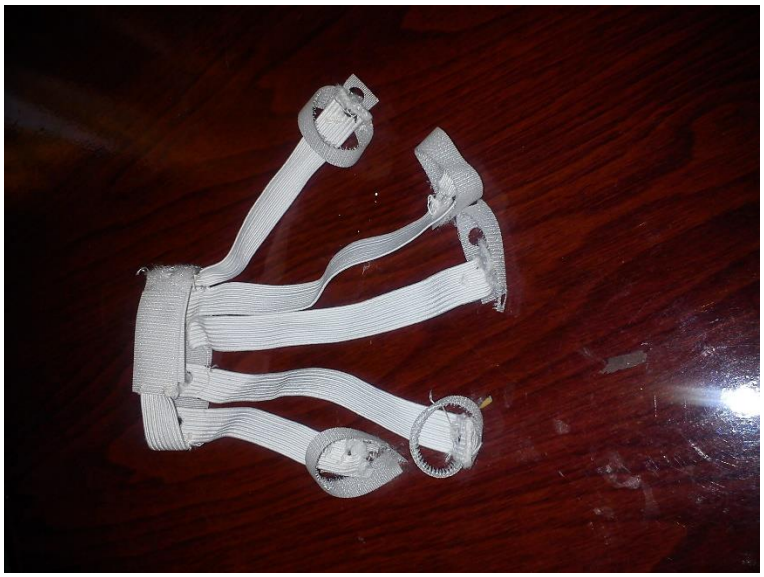


Figura. 70 Guante final

5.4 Conexión

El siguiente diagrama muestra la conexión básica y principal del software que con el que se controla el mecanismo del prototipo siendo la pc el componente principal por el cual interactúan Arduino, Java y MySql Workbenck

Figura. 71 Diagrama conexión

La Figura 72 explica la distribución de pines de algunas marcas muy importantes de servomotores.

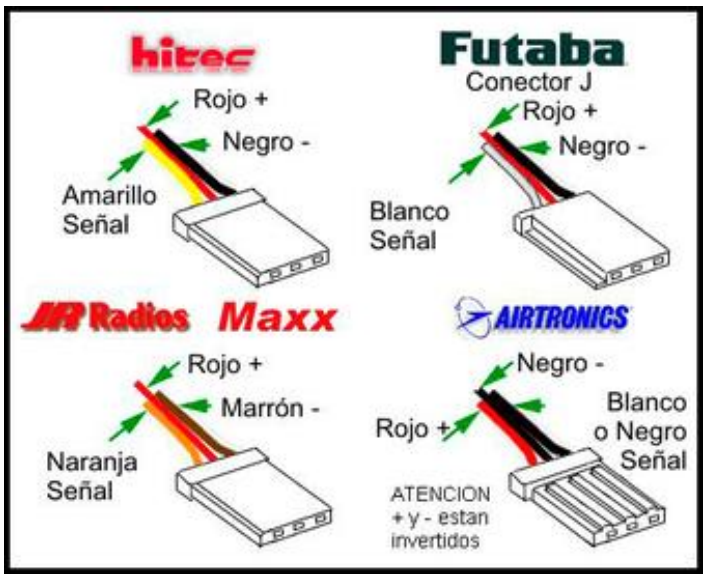


Figura. 72 Distribución pines

Los servomotores que se manejaron son los de la marca JR RADIOS y MAXX (Esquina inferior izquierda de la Figura 72), es importante hacer mención acerca de que no se tuvo ningún problema, su funcionamiento es muy aceptable.

La Figura 73, muestra una imagen la cual representa la forma en cómo se conectó el cableado que va de cada servomotor a la placa Arduino, el color marrón representa el cable por donde se enviaran los datos que previamente están cargado en la placa Arduino, el rojo el voltaje y el café la tierra.

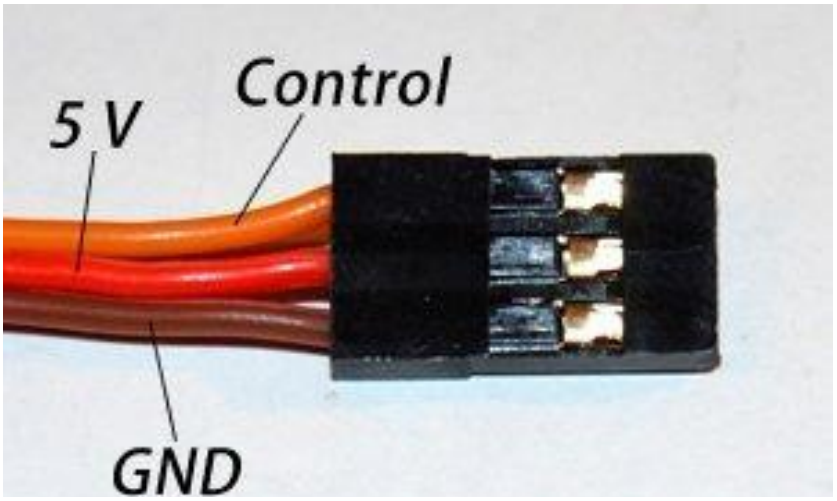
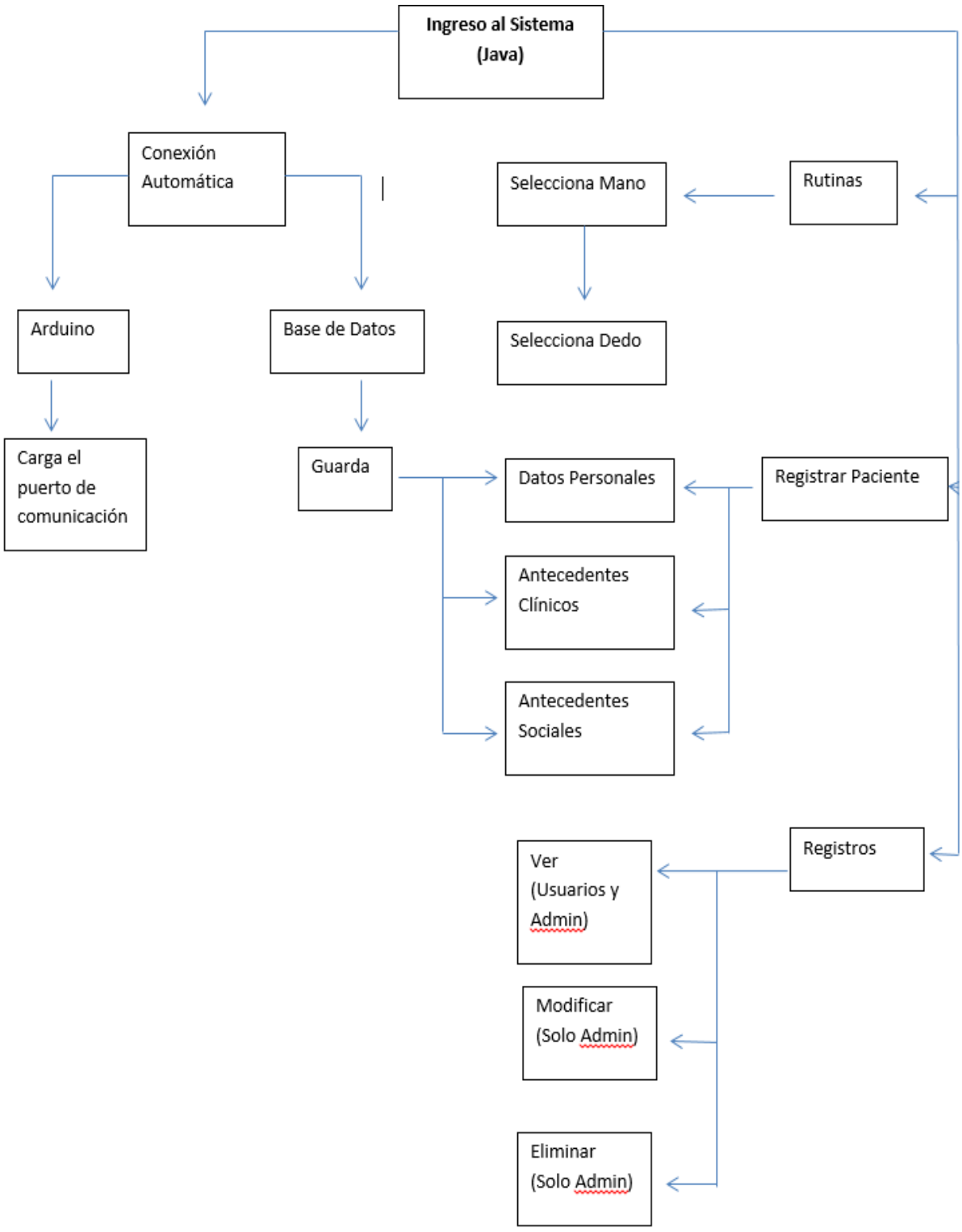


Figura. 73 Pines

5.4.1 Algoritmo de comunicación



VI CONCLUSIONES Y RECOMEDACIONES

6.1 Conclusiones

La implementación del dispositivo automatización así como el guante para la rehabilitación física de la mano, permitirá a las personas recuperar la movilidad con lo cual podrán realizar sus actividades con mayor autonomía de tal forma que podrán seguirse desempeñando de manera habitual, y seguir mejorando la movilidad de la mano hasta poder recuperarla a casi un cien por ciento esto tomando en cuenta que será de gran comodidad poder contar con el aparato en su hogar lo cual implica un menor costos, además de brindarles mayor seguridad.

Por último agregar que debemos darle un mayor auge a la tecnología en nuestra sociedad, para así permitir un desarrollo tanto de dispositivos como de aplicaciones de calidad y con un grado alto en cuanto a utilidad y funcionalidad de esta manera se tendrán nuevo proyectos elaborados con un uso más fácil, rápido y eficaz.

.

6.2 Trabajos futuros

Dentro de este proyecto que es muy amplio por lo que como primera etapa se busca que el usuario pueda ocupar la aplicación más que en su computador, en su teléfono móvil ya que como se sabe hoy en día la mayoría de personas tiene un teléfono celular en este particular caso sería bajo el Sistema Operativo Android por lo que se modificara el software creado para adaptarlo a esa plataforma.

Como una segunda etapa se busca que la consulta de cualquier paciente se pueda hacer desde la comodidad de su casa por lo cual se creara una base de datos que este dentro de un sitio Web.

Y por último como una tercera etapa se podría definir que el prototipo conste con mayores funciones de tal manera que sea más dinámico por lo cual se pretende buscar alcanzar el movimiento del codo, para la rehabilitación completa del brazo de la persona que sufra esta característica de perdida de movimiento.

ANEXO: Casos de uso

Figura. 74 Modelo caso de uso

VII REFERENCIAS BIBLIOGRÁFICAS

- [1]¿Qué es la rehabilitación?

[Internet], http://www.sabersinfin.com/index.php?option=com_content&task=view&id=558&Itemid=46 [Acceso 12 de septiembre 2013].

- [2]Historia de la rehabilitación y la terapia física universal.

[Internet], http://www.slideshare.net/DanLopez1/historia---de---la---rehabilitación---la---terapia---física [Acceso 12 de septiembre 2013].

- [3]Historia de la fisioterapia

[Internet], http://www.juntadeandalucia.es/averroes/~29701428/salud/fisio.htm [Acceso 12 de Septiembre 2013].

- [4]Terapia Física

[Internet], http://www.unocero.com/2014/01/23/dispositivo-robotico-bio-inspirado-para-ayudar-a-la-rehabilitacion-fisica/ [Acceso 5 de septiembre 2013].

- [5]Unidad de patología de la mano y extremidad superior

[Internet],http://www.icatme.com/home.php?secc=4&idUnit=8&part=1 [Acceso 14 Septiembre 2013].

- [6]Cirugía Ortopédica Anatomía de la mano

[Internet],http://nyp.org/espanol/library/orthopaedics/handpain.html [Acceso 14 Septiembre 2013].

- [7]Arduino.

[Internet], http://arduino.cc/es/Main/ArduinoBoardMega [Acceso 12 de septiembre 2012].

- [8]Servomotores

[Internet] http://www.todorobot.com.ar/documentos/servomotor.pdf [Acceso 12 Septiembre 2012]
[Internet]

- [9]Java

[Internet] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/java/java.htm [Acceso 12 Septiembre 2012]
[Internet] http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf [Acceso 12 Septiembre 2012]

- [10]MySQL

[Internet] http://es.wikibooks.org/wiki/MySQL/Introducci%C3%B3n_a_MySQL [Acceso 12 Septiembre 2012]

- [11]Gripmaster Pro, para fortalecer manos y dedos.
[Internet], http://www.vitonica.com/equipamiento/gripmaster---pro---para---fortalecer---manos---y---dedos [Acceso 5 de septiembre 2013].

- [12]Noticias de Navarra.
[Internet], http://www.noticiasdenavarra.com/2012/06/20/sociedad/estado/presentan---en---espana---un---robot---para---rehabilitar---la---movilidad---de---la---mano---en---personas---con---danos---cerebrales [Acceso 5 de Septiembre 2013].

- [13]NEOTEO ABC.
[Internet], http://www.neoteo.com/servomotores---el---primer---paso---hacia---tu---robot [Acceso 12 de septiembre 2013].

- [14] INEGI.
[Internet], http://cuentame.inegi.org.mx/poblacion/discapacidad.aspx?tema=P [Acceso 28 Octubre 2013].

- [15] INEGI
[Internet] http://www.inegi.org.mx/prod_serv/contenidos/espanol/bvinegi/productos/censos/poblacion/2000/discapacidad/discapacidad2004.pdf [Acceso 28 de Octubre].