UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

DISEÑO DE UNA WSN PARA LA DETECCIÓN Y ALERTA DE CAÍDA DE PERSONAS GERIÁTRICAS UTILIZANDO SISTEMA LIDAR Y REDES NEURONALES

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Electrónica y Automatización

AUTOR: Andrés Isaac Chávez Cajas
Luis Enrique Posso Luzuriaga

TUTOR: Andrés Sebastián Calero Calero

QUITO - ECUADOR

2023

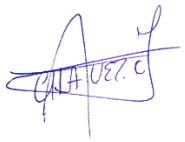
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

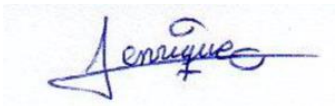
Nosotros, Andrés Isaac Chávez Cajas con documento de identificación No. 1721979399 y Luis Enrique Posso Luzuriaga con documento de identificación No. 1719809723 manifestamos que:

Somos autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 08 de agosto del año 2023

Atentamente,





Andrés Isaac Chávez Cajas
1721979399

Luis Enrique Posso Luzuriaga
1719809723

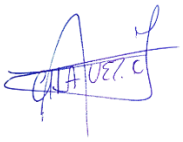
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

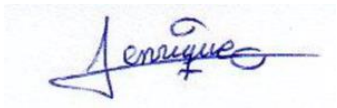
Nosotros, Andrés Isaac Chávez Cajas con documento de identificación No 1721979399 y Luis Enrique Posso Luzuriaga con documento de identificación No 1719809723 expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del proyecto técnico: DISEÑO DE UNA WSN PARA LA DETECCIÓN Y ALERTA DE CAÍDA DE PERSONAS GERIÁTRICAS UTILIZANDO SISTEMA LIDAR Y REDES NEURONALES, el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica y Automatización en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana

Quito, 08 de agosto del año 2023

Atentamente,





Andrés Isaac Chávez Cajas
1721979399

Luis Enrique Posso Luzuriaga
1719809723

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Andrés Sebastián Calero Calero con documento de identificación No 1719252346 docente de la Universidad Politécnica Salesiana declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO DE UNA WSN PARA LA DETECCIÓN Y ALERTA DE CAÍDA DE PERSONAS GERIÁTRICAS UTILIZANDO SISTEMA LIDAR Y REDES NEURONALES, realizado por Andrés Isaac Chávez Cajas con documento de identificación No. 1721979399 y Luis Enrique Posso Luzuriaga con documento de identificación No. 1719809723, obteniendo como resultado final el trabajo de titulación bajo la opción proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 08 de agosto del año 2023

Atentamente,

Ing. Andrés Sebastián Calero Calero Ms.C.
1719252346

DEDICATORIA

Dedico este proyecto de pregrado a mi esposa Mari por apoyarme en todos los momentos buenos y en especial esos momentos que me sentía derrotado, ella me ha enseñado lo que es la perseverancia hasta el final de cualquier meta.

A mi padre Roque y mi madre Mayra les doy como ofrenda este proyecto por sus grandes sacrificios para yo poder seguir una carrera universitaria luego de tantos intentos por mi parte.

A mis hermanas Valery y Catherine que siempre me han apoyado aun sin ser su responsabilidad, les dedico con mucho amor este proyecto.

A mis abuelos Manuel y Natividad que con su amor incondicional les doy como ofrenda por apoyarme para poder culminar mi carrera profesional.

A mis seres amados Esteban y Pablo que cuando estuvieron a mi lado me demostraron la valentía y esfuerzo por conseguir lo que uno se propone y que ahora desde el cielo me están cuidando en cada decisión que estoy tomando.

Y finalmente a mi gran amigo Luis le dedico porque, aunque haya tenido trabas en la elaboración de este proyecto me acompañó hasta la culminación del mismo.

Chávez Andrés

DEDICATORIA

Esta Trabajo de Titulación para el Nivel de Grado se la dedico a mis padres, María y Luis, por acompañarme en cada paso que doy en la búsqueda de ser mejor persona, responsable y profesional.

También se la dedico a mis familiares, algunos desde el cielo, que son y eran esa luz que me daba fuerzas para continuar.

A mis hermanas, por todo su apoyo incondicional, espero les sirva de ejemplo de que todo se puede lograr.

Y a mi gran compañero de tesis, quien me hacía reaccionar cuando pensaba que no podía continuar.

Muchas gracias de corazón.

Posso Luis

AGRADECIMIENTO

Primero quiero agradecer a Dios por ponerme a todas las personas en el tiempo que me tomó realizar este proyecto de pregrado.

Agradezco a mi tutor Andrés Calero por toda esa dedicación y paciencia, los momentos que el me compartió sus conocimientos técnicos y valores los tendré siempre en consideración para el resto de mi vida.

Mi gran agradecimiento a mi esposa y compañera de vida Mari por enseñarme todos sus grandes valores ya que sin ella no sería la persona que soy hoy en día y sin nada de ello no hubiera podido culminar este gran paso académico de mi vida, gracias por ese amor infinito e incondicionalidad.

Un agradecimiento muy pero muy grande a mis padres que me han apoyado y acompañado en todos los momentos y en especial en el desarrollo de este trabajo de pregrado que ha sido un gran reto para mí, ellos fueron un pilar fundamental para lograr este gran reto. ¡Muchísimas gracias!

Agradezco a mis abuelos Manuel y Natividad y mis hermanas porque me han ayudado inmensamente para poder culminar mis estudios aun sin ser su responsabilidad, son las personas que me han enseñado lo que es el cariño.

Finalmente, a la Universidad Politécnica Salesiana le agradezco por abrirme las puertas de su institución y enseñarme que ser un profesional no solo conlleva un ambiente académico, sino que también es importante ser alguien que apoya a la comunidad y en especial a quien más lo necesita, su espíritu salesiano impactó en mi vida y lo recordaré siempre.

Chávez Andrés

AGRADECIMIENTO

Le agradezco muy profundamente a mi tutor por su gran dedicación y gran paciencia, sin sus palabras, perseverancia y correcciones precisas no hubiese podido lograr llegar a esta instancia tan anhelada. Gracias por su guía y todos sus consejos, los llevaré grabados para siempre en la memoria en mi futuro profesional.

Además, las gracias a la gran Universidad Politécnica Salesiana, por permitirme tener tan buena experiencia dentro de la universidad, por permitirme convertirme en ser un profesional en lo que tanto me apasiona, gracias a cada maestro de mis materias que hizo parte de este proceso integral de formación.

Finalmente agradezco a quien lea este apartado, por permitir a mis experiencias, investigaciones y conocimiento, incurrir dentro de su repertorio de información.

Posso Luis

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Este proyecto de grado desarrollará un sistema de detección de caídas de adultos mayores, el cual estará constituido por una Red de Sensores Inalámbricos (WSN: Wireless Sensor Network), específicamente con los sensores LiDAR VL53LXX-V2 y GYUL53L0X, conjuntamente con el sensor ultrasónico HC-SR04, los cuales tienen el mismo funcionamiento que es medición de distancia por el tiempo de vuelo.

Para la transmisión de datos se usará la tecnología de radiofrecuencia con el uso del módulo nrf24l01. Para la implementación y desarrollo de la red neuronal artificial se usará el programa Matlab el cual con sus herramientas se entrenará la red y se obtienen los pesos para luego implementar la red neuronal en lenguaje C en la tarjeta de Texas Instruments Tiva C, la misma que de manera simultánea creará el servidor web para él envió de notificaciones hacia una dirección IP específica.

Se usará una tarjeta FPGA Basys 3 para la creación de una interfaz gráfica en 2D pues dispone de un puerto VGA el cual permitirá la conexión con un monitor que tenga el mismo puerto de entrada. Finalmente se realizarán pruebas que simulen las diferentes acciones que conlleva el diario vivir.

ABSTRACT

This project will develop a fall detection system for older adults, which will be integrated by a Wireless Sensor Network (WSN: Wireless Sensor Network), specifically with the LiDAR sensors VL53LXX-V2 and GYUL53L0X, and the ultrasonic sensor HC-SR04, which have the same operation as measure of distance for flight time.

For data transmission, radio frequency technology will be used with the nrf24l01 module. The implementation and development of the neural network will be trained by, Matlab software, which with its tools will train the network and obtain the weights to then implement the neural network in C language on the Texas Instruments Tiva C card, which will simultaneously create the web server to send notifications to a specific IP address.

A Basys 3 FPGA card will be used to create a 2D graphic interface since it has a VGA port which will allow connection to a monitor that has the same input port. Finally, tests will be carried out that simulate the different actions that daily living entails.

INTRODUCCIÓN

Este capituló presenta de manera general lo que se trata en cada capítulo.

El capítulo número 1 describe la problemática e importancia de este proyecto hacia las personas geriátricas al momento de caerse; se exponen datos sobre la probabilidad de mortalidad que está aumentando y las caídas al ser un problema común en las personas geriátricas se determina la importancia que tiene el desarrollo de sistemas que detecten sus caídas sin incomodarlos.

El capítulo 2 recopila información cualitativa sobre todos los componentes que conforman este proyecto, tales como: sensores LiDAR, módulos de radiofrecuencia, tarjetas FPGA y tarjetas TIVA, redes neuronales, entre otros más. Es decir, se enfoca en la explicación del proceso, la funcionalidad, las variables que intervienen, las condiciones de operación y sus formas de operación.

El capítulo 3 explica el proceso por el cual se siguió para lograr cumplir con los objetivos, se explica cómo se experimentó con los sensores para luego crear la red inalámbrica de sensores, posterior a esto se explica cómo se creó la red neuronal para implementarla en la tarjeta TIVA C asi como también el servidor web. Se explica los pasos que se tomaron para la creación de la interfaz gráfica en 2D con la tarjeta FPGA Basys 3, todos estos procesos son mostrados también con los resultados obtenidos durante el proceso.

El capítulo 4 contiene las conclusiones de cada objetivo y las recomendaciones que se tiene para futuras mejoras en el trabajo.

CAPITULO 1
ANTECEDENTES

1.1.Descripción del problema.

El estado fisiológico, psicológico y morfológico de las personas geriátricas ha ido decayendo pues tienen dificultad para realizar cualquier actividad de la vida cotidiana. De acuerdo a estudios realizados a 61 adultos mayores se determinó que del total de ellos: el 41% padecen de hipertensión, el 25% de ellos sufren de diabetes y osteoporosis, lo que conlleva a un alto riesgo de que sufran ya que gracias a estas enfermedades y condiciones existe perdida de equilibrio para caminar (Flórez & Mahecha, 2020).

De acuerdo a la Organización Mundial de la Salud (OMS), El grupo de personas considerados como adultos mayores se está incrementando a nivel mundial (Badgujar & Pillai, 2020); De acuerdo a Porras durante los años 2000 y 2050 se estima que del 11[%] pase al 22[%] de la población(Porras,2015). Para 2050 se estima que una o más de cada cinco personas tendrá 65 años o más (Singh, Rothe & Rathkanthiwar, 2017).

La Organización de las Naciones Unidas (ONU) menciona que nuestro país también forma parte del proceso de envejecimiento poblacional. Los habitantes ecuatorianos considerados adultos mayores serán el grupo mayoritario aproximadamente para el año 2065 y esto se debe a que la mortalidad ha ido cambiando desde los años 1950 porque de acuerdo a estudios se tenía un 49.3[%] para sobrevivir; hoy en día, la tasa de mortalidad que llegaba a los 60 años aumentó al 85.5[%]; siendo este porcentaje una cifra no muy lejana a los países de primer mundo, pues ellos tienen una tasa de mortalidad del 91.2[%] (Miller & Mejía, 2020).

Los ancianos enfrentan desafíos como: falta de asistencia adecuada, elevados costos de manutención, entre otros. Por lo cual, la calidad de asistencia y cuidado en el sistema de salud geriátrico occidental está decayendo (Ismail, 2019). Alarcón y Flores (2017) plantean que existe un porcentaje elevado en la dificultad para la movilidad en adultos mayores. Debido a esta crisis, ha incrementado el interés por monitorizar las condiciones de salud de los adultos mayores a través de sistemas inteligentes (Inteligencia Artificial) y dispositivos electrónicos de última tecnología para la supervisión y notificación a los cuidadores.

Se han realizado varias investigaciones de sistemas de detección de caída basados en sensores. Hassan et al. (2019) usó un acelerómetro para registrar los datos y enviarlos a un smartphone, Gonzáles et al. (2016) desarrollaron un sistema pensado para ser llevado en la cintura; Andrade y Redrován (2016) diseñaron un sistema que debe portar el adulto mayor en el tórax para su monitoreo. Todos estos sistemas causan incomodidad en los adultos mayores ya que deben portarlos permanentemente, es así que surge la idea de establecer un sistema de monitoreo para la detección de caídas en los espacios que habitan los adultos mayores sin interferir en sus movimientos para crear una mayor comodidad para ellos.

1.2.Objetivos.

1.2.1. Objetivo general

- Desarrollar un sistema de detección de caídas de personas geriátricas para la notificación al personal de cuidado mediante sistema LIDAR y redes neuronales.

1.2.2. Objetivos específicos

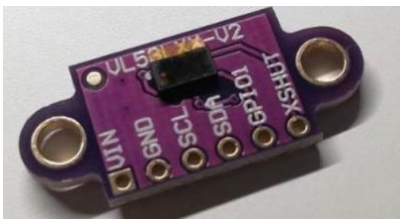
- Diseñar un sistema LIDAR de escaneo para la obtención de variables de personas geriátricas para el envío de datos a través de una red inalámbrica.
- Desarrollar un algoritmo de redes neuronales que permita la identificación de la caída de una persona a través de tarjeta de desarrollo TIVA.
- Desarrollar una interfaz gráfica de 2D para la ubicación y monitoreo de la persona geriátrica en el entorno a través de una tarjeta FPGA.
- Diseñar una arquitectura unidireccional para el envío de notificaciones de alerta de caída desde dispositivos IoT (TIVA) hacia entorno de nube.
- Verificar el funcionamiento del sistema de detección de caídas de personas geriátricas para la validación a través de pruebas experimentales en entorno controlado.

CAPITULO 2
MARCO TEÓRICO

2.1. LiDAR

Para el sistema LiDAR se usaron sensores con numeración VL53L0X los cuales tienen un sistema óptico con filtros anti - infrarrojo. Estos sensores usan el método de medición de distancia por tiempo de vuelo (ToF: Time of Flight). Están fabricados en diferentes versiones, las dos versiones usadas en este proyecto poseen las mismas características a nivel electrónico, pero varía el modelo del empaque; el modelo VL53LXX-V2 se lo puedo divisar en la figura 2.1 mientras que el modelo GYUL53L0X se puede observarlo en la figura 2.2 Estos sensores son muy pequeños ya que miden 2x1[cm] y ofrecen mediciones casi precisas en cualquier superficie reflectante, puede alcanzar medidas de hasta 2m. Tiene un emisor de un haz de luz cada cierto tiempo, al rebotar el haz de luz sobre el objeto, el sensor LiDAR mide el tiempo entre la emisión y recepción del haz de luz luego de rebotar en el objeto; al saber que la velocidad de propagación del haz de luz es una constante se podría calcular la distancia al objeto. (Gorrostieta, Ramos y Vargas, 2018).

Figura 2.1: Sensor LiDAR



Sensor LiDAR modelo VL53LXX-V2, Autores: Chávez Andrés y Posso Luis.

Figura 2.2: Sensor LiDAR

Sensor LiDAR modelo GYUL53L0X, Autor: Chávez A. y Posso L.

Tabla 2.1. Datos sensor LiDAR VL53L0X

Voltaje de entrada	3 a 5 VDC
Corriente de consumo	10mA a 40mA
Rango de distancia	0.05 m a 1.20 m
Precisión	±0.003m
Comunicación	I2C
Dirección I2C programable	Por defecto: 0x52
Funciona correctamente con ambientes con luz infrarroja.	
Xshutdown (reset) y pin de interrupción	

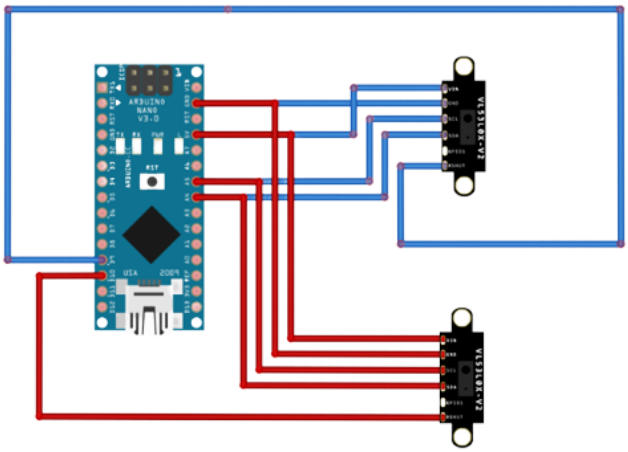
Tabla de especificaciones de los sensores VL53LXX-V2 y GYUL53L0X, Fuente: Gorrostieta, Ramos y Vargas, 2018

2.2. Comunicación I2C

Es un modelo de comunicación entre un elemento maestro y uno o muchos elementos esclavos. El elemento maestro es el que está administrando y gestionando qué elemento esclavo debe transmitir en cierto momento. Cada dispositivo posee una dirección específica para que el elemento maestro pueda identificar a cada esclavo.

Este bus de comunicación posee dos líneas o dos alambres llamados SDA y SCL. La línea llamada SCL es la que se encarga de transmitir la señal del reloj que trabaja el nodo maestro mientras que la señal SDA es la que se encarga de llevar la información. En caso de que no se esté transmitiendo ningún dato las dos líneas se mantienen en nivel alto (Álvarez A., 2022). Los sensores LiDAR necesitan de la comunicación I2C.

Figura 2.3: Sensores LiDAR

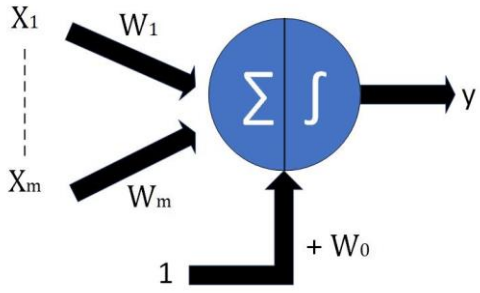


Múltiples sensores LiDAR conectados al Arduino Nano con comunicación I2C. Autor: Álvarez A., 2022

2.3. Red Neuronal Artificial (ANN)

Es la simulación con algoritmos computacionales del sistema nervioso que posee el ser humano. Su arquitectura se basa en la conexión de varias neuronas artificiales que realizan un cálculo matemático individualmente a través de algoritmos. Cada neurona posee un peso y una función de activación que a través de un entrenamiento se pueden ajustar los pesos y así satisfacer los requerimientos de desempeño de cada neurona para encontrar un modelo que se ajuste a los datos de entrada que ingresen a la ANN. Existe diferentes tipos de entrenamiento de la red, pero los más conocidos son los entrenamientos supervisado y no supervisado. (Salas, R., 2004)

Figura 2.4: Diagrama de una red neuronal artificial



La red Neuronal posee un vector **W** de pesos que serán multiplicadas por los valores de entrada del vector **X** y su sumatoria pasará por una función de activación y un sesgo W_0 que dará una única salida **y**. Autor: (Salas, R., 2004)

El modelo de una neurona artificial se la puede considerar como se observa en la figura 2.4, esta neurona tiene un vector de entrada **W** que contiene los pesos equivalentes a la conexión sináptica de una neuronal real; el bias o sesgo representado por w0 es un umbral de activación que ayuda a la activación de la neurona. La simulación consiste en la sumatoria de la multiplicación de los valores que tienen las entradas y los pesos correspondientes y que esta sumatoria atraviese por una función de activación que en este caso se usa la función tangente hiperbólica (Salas, R., 2004)., cuya expresión matemática es la siguiente:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad \text{Ec. 2.1}$$

2.3.1.Redes de propagación hacia atrás (Back Propagation).

Este tipo de entrenamiento se basa en propagar el error, empezando desde la capa de salida pasando por las capas ocultas hasta la primera capa y así poder ir modificando el peso de las interconexiones durante el entrenamiento. Ciertas funciones de activación no son lineales (sigmoidea o tangente hiperbólica), Lo primero que se calcula es la derivada de la función con respecto al peso. Esta derivada cambia los pesos para multiplicar por la derivada de la función de activación con cada una de las neuronas pertenecientes a la capa de salida. (Matich, 2001)

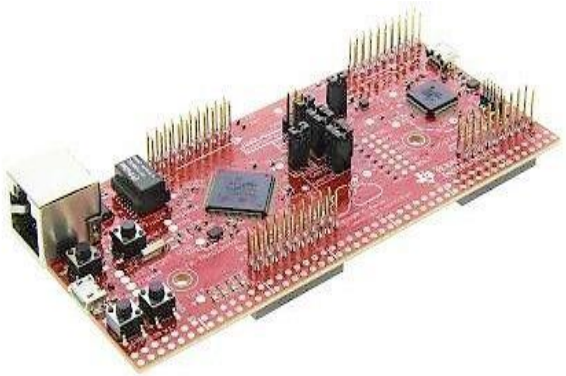
2.4. Tarjeta TIVA

La tarjeta TIVA C modelo TM4C1294 es una placa de desarrollo con bajo costo de adquisición. Está basados en Arm Cortex-M4F. Se destaca por su puerto de Ethernet, su módulo de entrada USB 2.0, tiene la capacidad de quedar en estado de hibernación, un bloque para la generación de señales. Sus características son las siguientes:

- Microcontrolador modelo ARM Cortex-M4 con frecuencia de 120 MHz y 32 bits.
- Memoria flash de 1 MB, memoria EEPROM de 6 kB y SRAM de 256 kB.
- Ethernet 10/100 integrado.
- Bloque ADC de 12 bits.
- Host USB 2.0 de alta velocidad.

- Sitios de conexión BoosterPack XL dobles y apilables.
- Compatibilidad con varias cadenas de herramientas de desarrollo: CCS, Keil e IAR.
- 4 LED integradas.
- 2 botones integrados.
- 1 botón de hibernación.
- 1 botón para el reinicio del microcontrolador.
- Jumpers para seleccionar la fuente de alimentación: USB ICDI, Dispositivo USB, Paquete de impulso.

Figura 2.5: Tarjeta TIVA



Distribución de puertos y pines de la tarjeta TIVA TM4C1294, Fuente: Texas Instruments.

2.5 Comunicación por Radiofrecuencia

El uso de la radiofrecuencia como un canal de comunicación ha abierto las puertas a diferentes aplicaciones en las diferentes industrias, tales como la automotriz, las telecomunicaciones y muchas más. En este caso se está usando para un sistema básico de monitoreo que está diseñado para alertar y evaluar posibles problemas de caídas en el espacio en donde se coloquen los sensores de monitoreo (Bell, 2013).

El módulo nRF24L01 es un módulo de recepción y transmisión de datos con una banda de 2.4 – 2.5 GHz (Bandas de frecuencia no comerciales). Este módulo contiene los siguientes módulos: sintetizador de frecuencia, un bloque para amplificar la potencia, oscilador de cristal, un modulador y un demodulador de señales. Su programación es mediante la interfaz SPI (Gagliardi T., 2018).

Figura 2.6. Módulo nRF24L01

Vista del hardware del módulo nRF24L01, Autor: Rivera M., 2020

La antena del módulo genera una onda a ciertas frecuencias para que el receptor pueda identificar la señal enviada, cabe recalcar que el receptor y el transmisor se deben encontrar a la misma frecuencia ya que en la banda a la que trabaja existen un grupo de bandas de radiofrecuencias no comerciales. (Bell, 2013).

Los módulos NRF24L01 son los más usados al momento de establecer una red de sensores debido a su bajo precio y buenas características para la comunicación de paquetes de baja tasa de bits En la tabla 2.3 se puede observar una comparativa entre los diferentes tipos de comunicación inalámbrica (Rivera M., 2020).

Tabla 2.2. Tipos de comunicación Inalámbrica

Consideraciones / Tecnología	Wi-Fi	Bluetooth	ZigBee	NRF24l01
Transmisión	50 Mbps	1 Mbps	250 kbps	2 Mbps
Uso de energía	Alto	Medio	Bajo	Muy Bajo
Consumo de corriente [mA]	400	40	30	11.3
Valor de adquisición	Alto	Medio	Bajo	Bajo
Implementación	Media	Media	Fácil	Muy Difícil
Rango de alcance	50 km	10 m	75 m	1 km

Comparación entre los diferentes tipos de comunicaciones, Fuente: Chávez Andrés y Posso Luis.

2.6. Comunicación Interfaz Periférica Serial (SPI)

Es una interfaz que se basa en la señal de reloj del maestro, las tramas de datos son enviados y recibidos de manera paralela a través de una línea serial. Su relación se basa en maestro-esclavo y debido a que este tipo de comunicación no posee direccionamiento es un poco más compleja para el microcontrolador ya que la comunicación I2C es muy útil cuando se tienen varios esclavos pues trabaja punto a punto (Villegas D. y Dager J., 2013).

2.6.1. Señales de la comunicación SPI.

Las 4 señales más importantes en la comunicación SPI son las siguientes:

- Master Out Slave Input (MOSI): Por esta línea el maestro envía la trama de datos a los nodos esclavos.
- Master Input Slave Output (MISO): Por esta línea el nodo esclavo envía la trama de datos al nodo maestro.
- Serial Clock (SCK): Es la señal sincrónica de reloj con la que trabaja el maestro para dirigir a los esclavos.
- Slave Select (SS): Esta señal utiliza el maestro para seleccionar con que esclavo quiere realizar la comunicación (Villegas D. y Dager J., 2013).

2.7. Red de Sensores Inalámbricos (WSN)

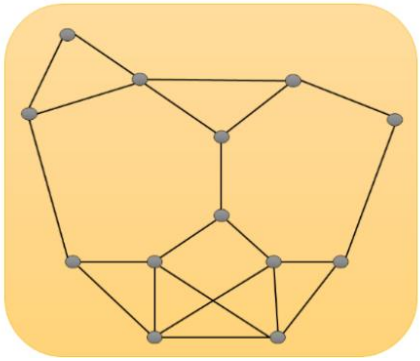
Es la agrupación de sistemas digitales independientes (nodos) colocados en diversos lugares para obtener información del ambiente o entorno que les rodea para luego procesarla y se pueda transmitirla de manera inalámbrica a través de las diferentes tecnologías de comunicación inalámbrica. Las limitaciones que poseen estas redes de sensores es la alimentación de los nodos. Por otro lado, estas redes son flexibles pues se pueden reconfigurarlas de acuerdo a las necesidades y con esto se puede decir que son escalables y de bajo coste (Rivera M.,2020).

2.7.1. Topologías de las WSN

Las topologías son las distribuciones y localizaciones de los diferentes nodos pertenecientes a la red, pues según su disposición los datos son tratados y transmitidos para lograr cumplir los requisitos de la aplicación que se esté realizando con la WSN (Serrano J., 2016).

- Topología Plana Todos los nodos tienen el mismo rango dentro de la red.

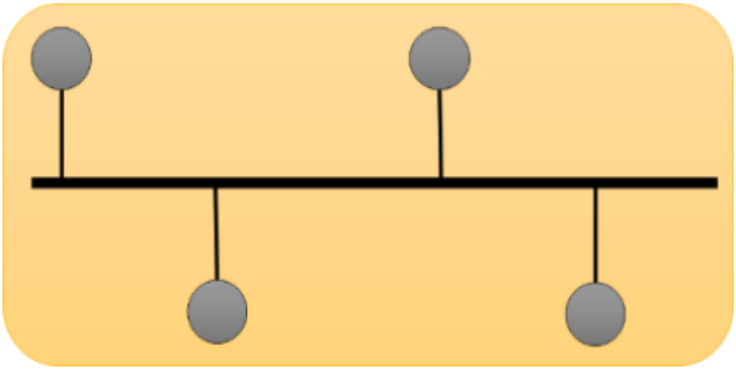
Figura 2.7: Topología plana



Distribución de nodos en topología plana, Autor: Rivera, M.,2020

- Topología Bus: Tiene una única línea de transmisión en la cual todos los nodos escuchan cuando un nodo va a transmitir, esta topología es de la más fácil de colocar, pero al tener una sola línea puede sufrir problemas de transmisión si existe mucho tráfico de tramas por lo cual no puede trabajar con un gran número de nodos.

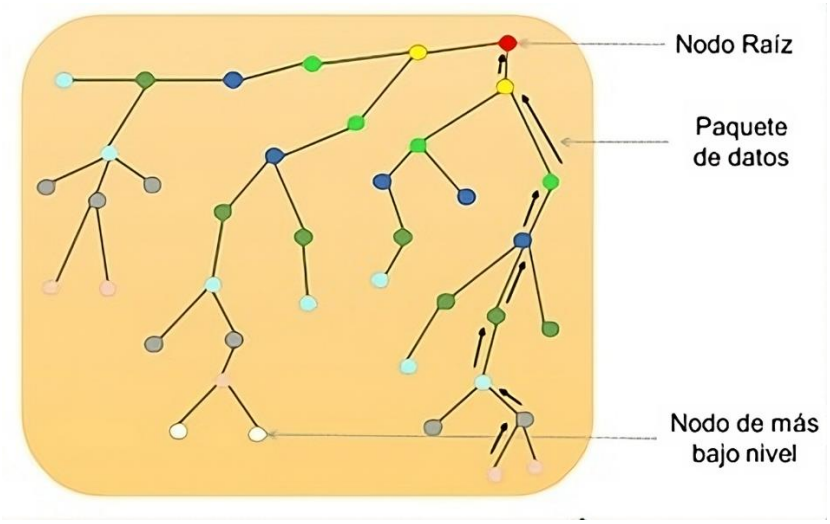
Figura 2.8: Topología bus



Distribución de nodos en topología bus, Autor: Rivera, M.,2020

- Topología Árbol: Se basa en la transmisión de datos en forma jerárquica,

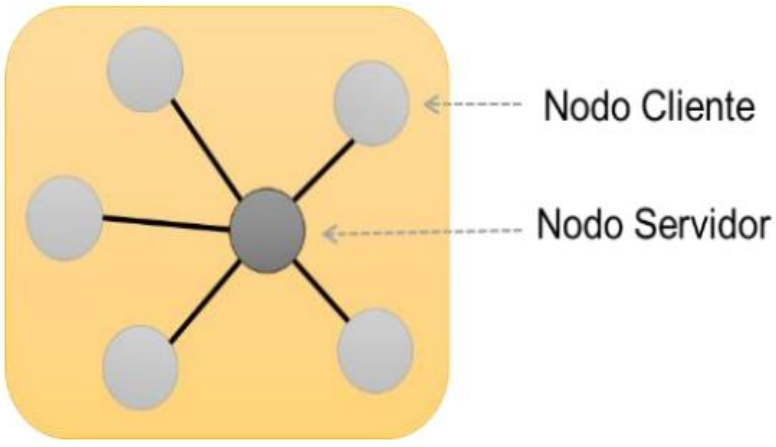
Figura 2.9: Topología árbol



Distribución de nodos en topología árbol, Autor: Rivera, M.,2020

- Topología estrella: Los nodos individuales se encuentran conectados y únicamente se conectan con un nodo central y estos nodos individuales no pueden comunicarse entre sí, es decir el nodo central toma el rol de servidor. El mayor problema de esta topología es que en caso de fallar el nodo maestro la red no podrá seguir trabajando.

Figura 2.10: Topología estrella



Distribución de nodos en topología estrella, Autor: Rivera, M.,2020

2.8. FPGA (Field Programmable Gate Array)

Son llamadas también LCA (Logic Cell Array). Se componen matrices que conforman bloques configurables mediante programación. Cuando se está programando una FPGA lo que hace es entrelazar los conmutadores para crear un circuito digital a base de matrices o compuertas lógicas (Arevalo S.,2014).

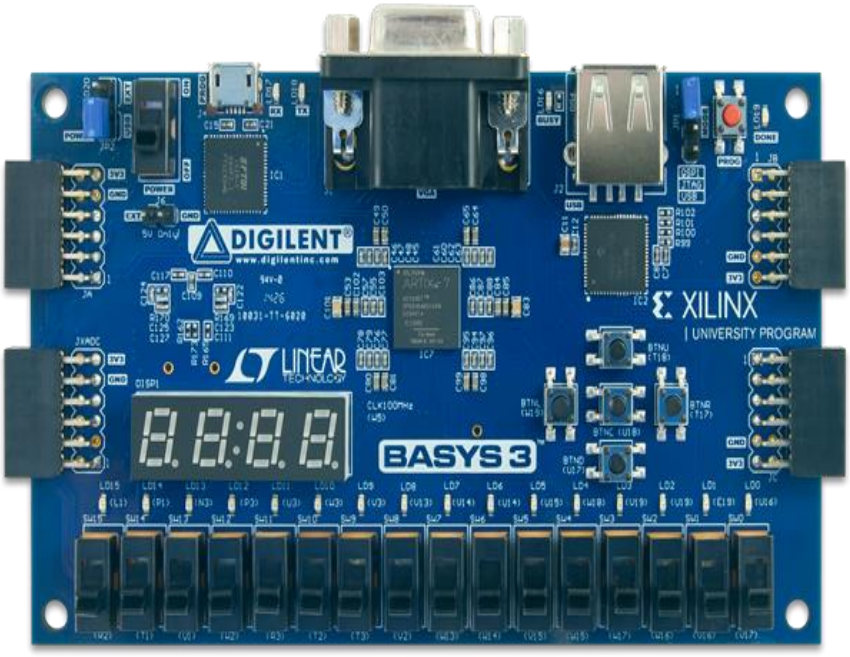
Los elementos de una FPGA son: Bloques lógicos que en conjunto se le denomina arquitectura y pueden ser desde una simple compuerta hasta módulos que componen un circuito secuencial, poseen además bloques de entradas, salidas, switches y dependiendo el modelo poseen puertos adicionales (López M. & Ayala J.,2004). En este caso se usará la FPGA Basys 3 que posee un puerto VGA el cual será útil para la creación de interfaces graficas en 2D.

2.8.1. FPGA Basys 3

La placa Basys 3 es una tarjeta de desarrollo que permite crear circuitos digitales a través de matrices con puertas programables, Pertenece a la familia Artix-7 la empresa Xilinx. El precio en el mercado es bajo, posee puertos USB, VGA y otros. la tarjeta Basys 3 tiene la capacidad de generar circuitos combinacionales simples hasta circuitos secuenciales completos. Tiene bloques de: botones, switches, luces LED y otros dispositivos de E/S necesarios para diseños de circuitos digitales(DILIGENT, 2016).

Su reloj interno la hace tan versátil pues puede llegar a los 450 MHz, su alimentación es de 5V y trabaja también con niveles de 3.3V. Posee un puerto VGA que junto con sus 12 BITS resolución permite una gama bien grande de colores, la desventaja que esta FPGA posee es que no posee memoria ROM por lo que se debe configurar los jumpers para que pueda cargar algún diseño de circuito desde una unidad USB.

Figura 2.11: Tarjeta FPGA



Distribución de pines de la tarjeta Basys 3, Fuente: DILIGENT.

2.8.2. Interfaz gráfica con la FPGA

La Interfaz cumple un rol importante en las diferentes aplicaciones, ya que este es el medio por cual se puede crear la interacción hombre – computadora. Una interfaz gráfica debe estar desarrollada de tal manera que el usuario pueda entender de manera rápida, precisa y satisfactoria la tarea que debe realizar el sistema (Albornoz C., Berón, M. y Montejano G.,2017).

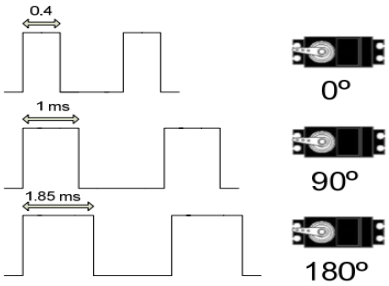
El formato VGA contiene 640 columnas por 480 filas de elementos (píxeles). Para la interfaz gráfica el monitor es escaneado de forma horizontal y vertical para variar los pixeles de acuerdo al periodo de tiempo que el controlador sea programado en la FPGA.

El monitor VGA necesita de 5 señales: las 3 señales para los colores RGB (rojo, verde y azul), una señal para la sincronización horizontal y otra señal para la sincronización vertical. Las tres señales de color son conocidas comúnmente como RGB y de acuerdo a sus combinaciones pueden crearse diferentes colores en la interfaz gráfica. Las señales de sincronización en cambio sirven para frecuencia de actualización de la pantalla y poder generar una imagen sin perder ningún pixel (Arévalo S.,2014).

2.9. Servomotor

Es un actuador que funciona con una señal con pulsos de anchos (PWM) es decir que se puede controlar la posición con mucha precisión y se puede mantener hasta no variar la señal PWM. Básicamente es un motor a corriente directa, con una caja reductora que a través de un circuito de control que dependiendo el modelo se puede mover desde los cero grados hasta los 180 grados o también desde los 0 grados y dar una vuelta completa de 360 grados. Su voltaje de funcionamiento es de 5V (Thompson C., 2009).

Figura 2.12: Señal PWM para el control de un servomotor.



Variación del ancho de pulso para girar el engranaje de 0 a 180 grados, Autor: Thompson C. 2009

2.10 Arduino NANO

Arduino Nano es una placa compacta y muy completa, es de la familia de microcontroladores ATmega328. Tiene una gran similitud con la tarjeta Duemilanove que pertenece igualmente a Arduino, la diferencia es el diseño que tienen ambas. No posee pines integrados por lo que toca comprar por separado, su programación es través del cable USB con terminal Mini-B.

Figura 2.13: Arduino NANO

Placa de desarrollo Arduino NANO, Fuente: ARDUINO, 2023

Tabla 2.3. Especificaciones de la tarjeta Arduino NANO

Microcontrolador	ATmega328
Voltaje de funcionamiento	5 V
Memoria flash	32 kB
Memoria SRAM	2 kB
Velocidad de reloj	16 MHz
Pines analógicos	8
Memoria EEPROM	1 kB
Corriente DC que entregan los pines	40 mA (I/O Pins)
Voltaje de entrada	7-12 V
Digital I/O Pins	22
Pines PWM	6
Consumo de Corriente	19 mA
Dimensiones	18 x 45 mm

Tabla de especificaciones de la tarjeta ARDUINO NANO, Fuente: (ARDUINO,2023)

2.11. Servidor Web

Un servidor web siempre se encuentra esperando peticiones o que se conecte un cliente el cual puede ser un celular, un pc o algún dispositivo que pueda ejecutar un servidor web. La mayoría de aplicaciones que requieran de un servidor web pueden demandar un diseño y ajuste bien meticuloso para ofrecer un rendimiento adecuado ya que puede aparecer latencia en aplicaciones en las cuales el tiempo de respuesta es crítico. Para aprovechar al máximo los recursos disponibles del servidor web se debe saber cómo está organizado la red y como se debe enviar los datos desde cada nodo (Navarro L. y Vilajosana G.,2019)

2.12. Arquitectura Unidireccional

Esta arquitectura solo puede llevar los datos en una sola dirección desde la tarjeta de desarrollo TIVA hacia el servidor web, este tipo de arquitectura es más confiable y con menos latencia que la arquitectura bidireccional por lo cual presenta ciertas ventajas:

- No requiere de un proceso de verificación.
- Las tramas de datos pueden ser extensas.
- Los datos se pueden actualizar en tiempo real.

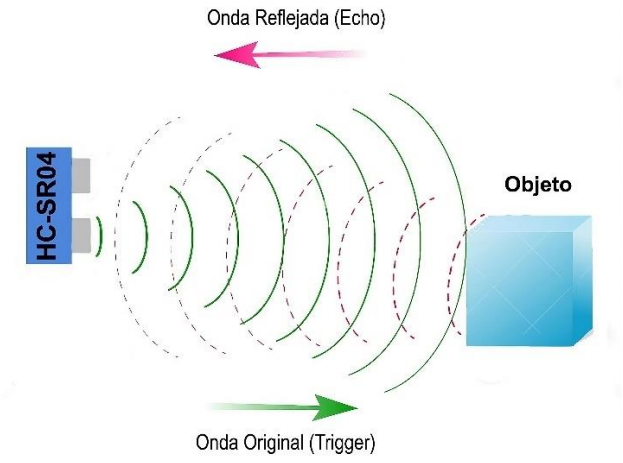
Figura 2.14: Arquitectura Unidireccional

Diagrama de flujo de la arquitectura Unidireccional para el servidor WEB. Autores: Chávez Andrés Y Posso Luis.

2.13. Sensor Ultrasónico HC SR04

Los sensores ultrasónicos a través de su diseño calculan el tiempo que ocurre desde el instante que se emite desde el módulo TRIG una señal de sonido a una alta frecuencia hasta regresar por el módulo ECO y posteriormente con procesos matemáticos en un microcontrolador se puede obtener la distancia a partir de la constante universal de la velocidad de viaje del sonido (Martínez V.,2014).

Figura 2.14: Sensor ultrasónico



Funcionamiento del sensor ultrasónico, Autor: Martínez V., 2014.

CAPITULO 3

DISEÑO E IMPLEMENTACIÓN DE UNA WSN PARA LA DETECCIÓN Y ALERTA DE CAÍDA DE PERSONAS GERIÁTRICAS UTILIZANDO SISTEMA LIDAR Y REDES NEURONALES.

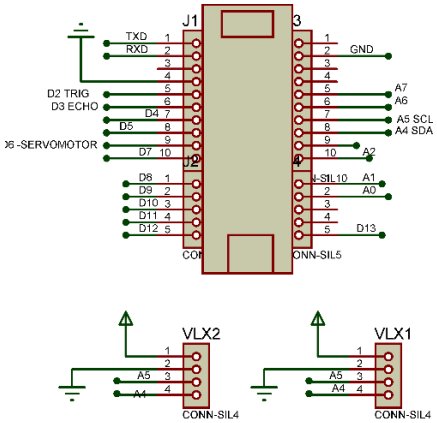
En este capítulo se presenta el proceso y la metodología que se empleó para la programación, el diseño e implementación de las diferentes partes del sistema para lograr cumplir los objetivos. La tabla 3.1 describe los componentes que se utilizaron en el proyecto.

3.1. Diseño del sistema LiDAR

3.1.1. Conexión de los sensores LiDAR

Como primer paso se realizó las conexiones en un protoboard los dos sensores LiDAR: el modelo VL53LXX-V2 y el modelo GYUL53L0X a un Arduino NANO para establecer la comunicación I2C y posteriormente a través del IDE de Arduino corroborar si los sensores estaban midiendo de manera correcta. Se usó el método mixto secuencial exploratorio ya que primeramente se recolectó datos cualitativos es decir determinar las especificaciones necesarias para realizar las conexiones y posteriormente se recolectarían datos cuantitativos. (Zita A., 2023)

Figura 3.1: Esquemático de la conexión de los sensores LiDAR



Esquemático de la conexión de sensores LiDAR con Arduino NANO, Autores: Chávez Andres y Posso Luis

Tabla 3.1 Componentes electrónicos utilizados con sus precios en USD

Cantidad	Nombre	Precio Unitario (USD)	Precio (USD)
6	Capacitores electrolíticos de 220uF	0.10	0.30
3	Capacitores electrolíticos de 10uF	0.5	0.15
3	Capacitores electrolíticos de 100nF	0.10	0.30
3	Regulador de voltaje 1117 3.3V	0.50	1.50
3	Regulador de voltaje 7805 5V	0.50	1.50
6	Servomotor SG 90	2.5	15
6	Módulo Vl53l0x	11.50	69
3	Fuente de adaptador de corriente de 9V y 1A	6.60	19.5
3	Arduino Nano	10	30
8	Conector Macho Header	1	8
3	Sensores ultrasónicos HC Sr04	2.50	7.50
4	Modulo Transceptor De 2.4 GHz Nrf24l01	2.50	10
3	Switches	0.50	1.50
30	Cable calibre AW18 en metros	0.25	7.50
6	Canaleta de 2x2 cm y 2 metros de largo	2	12
1	1 tarjeta TIVA C serie tm4c1294	70	70
1	FPGA Basys 3	70	70
6	Diodos Led	0.15	0.90
6	Resistencias de 330 Ohm	0.5	0.30
3	Cajas en MDF	4.50	13.50
3	Placas PCB	5	15
1	Monitor VGA	30	30
1	Modem	10	10
TOTAL			339

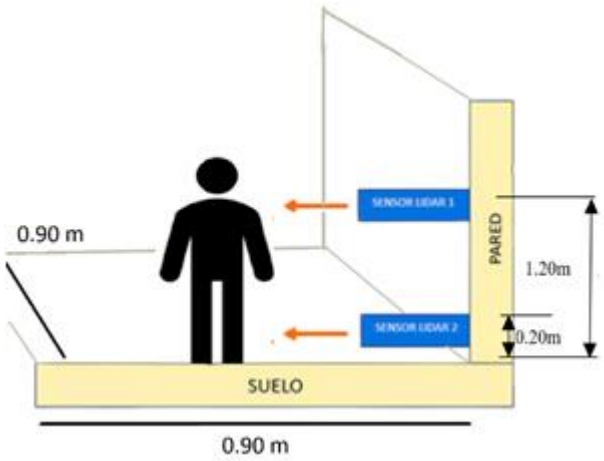
Componentes electrónicos pasivos y activos utilizados con sus precios en USD.

Autores: Chávez Andrés y Posso Luis.

3.1.2. Colocación de los sensores LiDAR

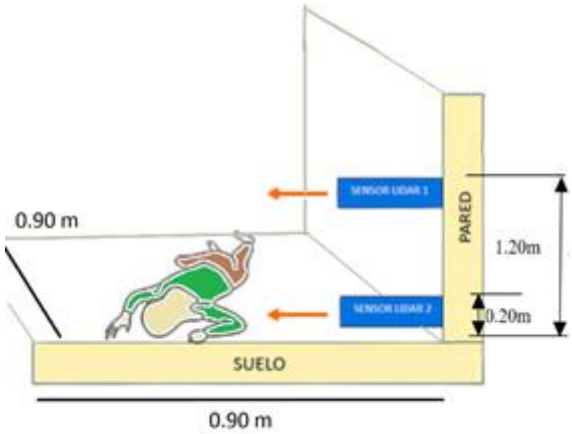
Se colocaron 2 sensores LiDAR para la medición de la persona geriátrica de manera horizontal. Con la ubicación de los sensores de esta manera, se podrá envió los datos a la tarjeta TIVA para determinar si la persona se encuentra de pie o en el piso.

Figura 3.2: Posición Sensores LiDAR



Colocación de los sensores LiDAR para medir de manera horizontal y obtener mejores resultados cuando una persona está parada, Autores: Chávez Andrés y Posso Luis.

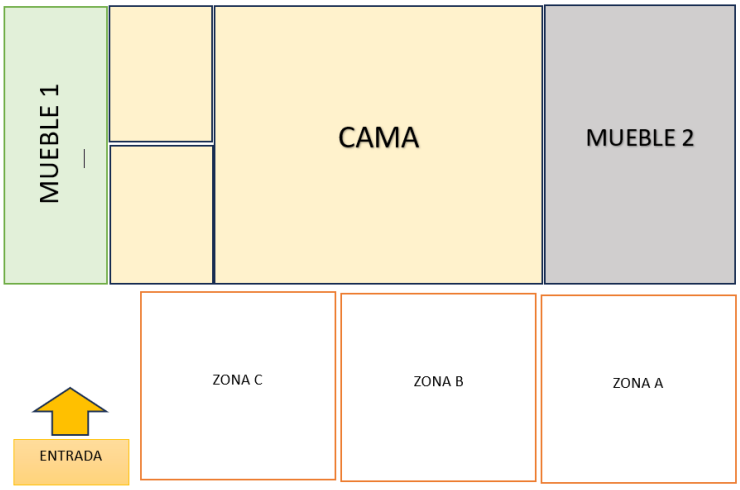
Figura 3.3: Posición Sensores LiDAR



Colocación de los sensores LiDAR para medir de manera horizontal y obtener mejores resultados cuando una persona está caída. Autores: Chávez Andrés y Posso Luis.

Posterior a la colocación de los sensores, se procedió a dividir el cuarto en 3 zonas como se observa en la figura 3.4. Esto nos sirvió para delimitar los espacios en los cuales los sensores deben medir.

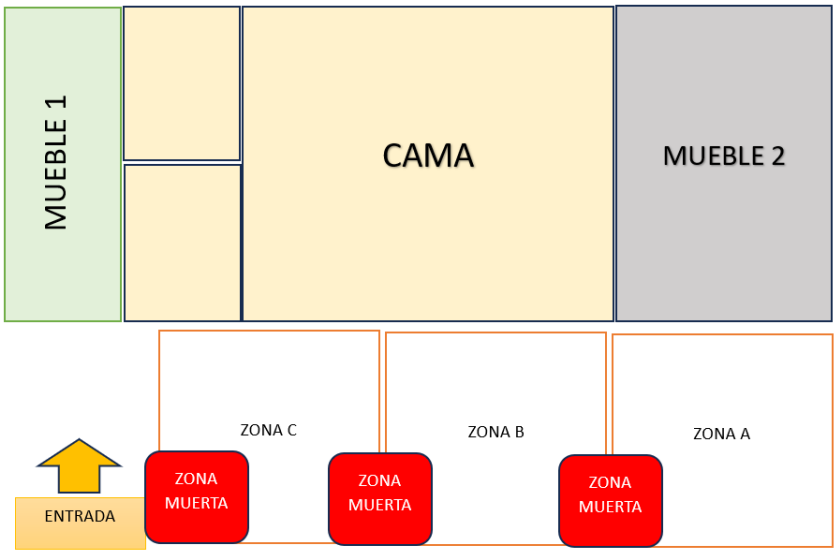
Figura 3.4: Zonas del entorno controlado



Creación de 3 zonas en el entorno controlado, Autores: Chávez Andrés y Posso Luis

Sin embargo, al colocar los sensores LiDAR para cada zona se crearon puntos muertos en los cuales los sensores no podían registrar la presencia de la persona, en la figura 3.5 se demuestran estas zonas.

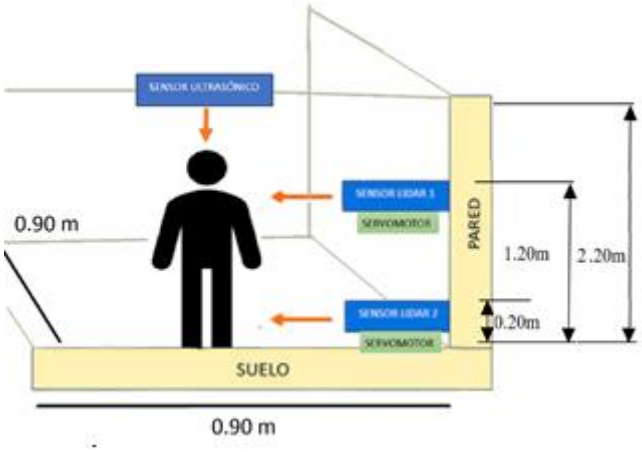
Figura 3.5: Zonas muertas del entorno controlado



3 zonas muertas en el entorno controlado, Autores: Chávez Andrés y Posso Luis

Como solución se procedió a colocar sensores ultrasónicos que permitan la detección de forma vertical en las zonas muertas y con la ayuda de servomotores puedan girar cierta cantidad de ángulos y se pueda medir de manera adecuada.

Figura 3.6: Posición sensores y servomotor

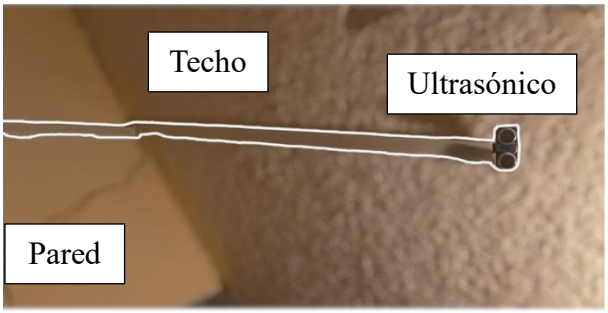


Colocación de los sensores LiDAR en conjunto con los servomotores para medir de manera horizontal mientras que el sensor ultrasónico lo hace de manera vertical y así obtener mejores resultados en las zonas muertas. Autores: Chávez Andrés y Posso Luis.

Figura 3.7: Sensor LiDAR colocado sobre el servomotor

Colocación del sensor LiDAR sobre el servomotor para que pueda girar a 0°, 90° y 180° para que pueda medir en las zonas muertas. Autores: Chávez Andrés y Posso Luis.

Figura 3.8: Sensor Ultrasónico colocado en el techo.



Sensor Ultrasónico colocado en el techo para la detección de la persona de forma vertical. Autores: Chávez Andrés y Posso Luis.

3.2. Diseño de la red sensores inalámbricos WSN

Al momento de elegir los elementos que formarán la WSN se consideró: el precio de adquisición de los componentes, Cantidad de datos a transmitir, la distancia entre nodos que es muy pequeña, por ende, no existe interferencia y como último punto que la red sea escalable. Con todos estos criterios se consideró el módulo nRF24L01 para los 3 nodos (Arduino Nano) y un módulo nRF24L01 conectado al nodo central (Tarjeta TIVA C).

Con la placa de desarrollo Arduino Nano existió problemas y no llegaban los datos de manera adecuada, llegaban tramas de datos que contenían los caracteres mostrados en la figura 3.9

Figura 3.9: Error de recepción de datos.

Se estaban enviando los datos, pero al momento de recibir no eran los correctos, Autores: Chávez Andrés y Posso Luis.

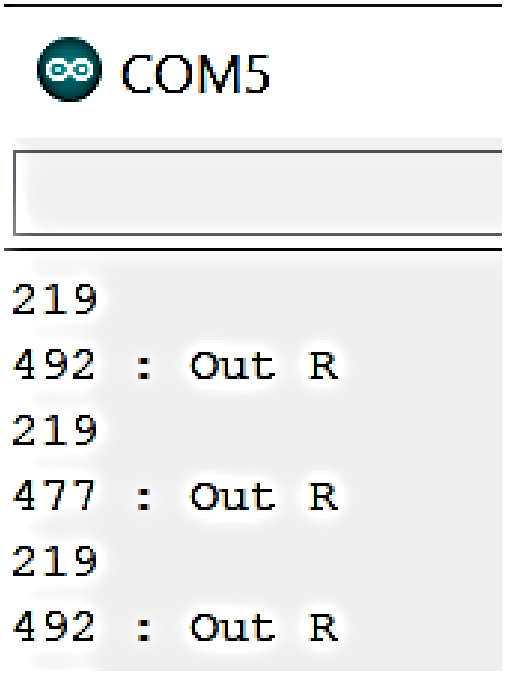
Para solventar este problema se colocó un capacitor para filtrar mejor el voltaje de entrada.

Figura 3.10: Colocación del capacitor en la entrada de voltaje del módulo nRF24L01

Capacitor para filtrar el voltaje de entrada, Autores: Chávez Andrés y Posso Luis

Al realizar experimentaciones se observó que si estaban midiendo adecuadamente los sensores y se procedió a trabajar con datos booleanos de la siguiente manera: como cada zona mide aproximadamente 90x90 [cm^2] en el caso de que el sensor mida menos de 85 cm de distancia significa que la persona está parada en la zona, se enviará un '1' en formato carácter y en el monitor serie de Arduino imprime la distancia en milímetros, mientras que si es superior a 85 cm se enviará un '0' e imprime el mensaje "Out R" y se entenderá que no existe la presencia de esa persona en la respectiva zona. De igual manera el sensor ultrasónico si recibe una medida mayor a 90 cm este enviará un carácter equivalente a '0' y si es menor un carácter equivalente a '1' con la diferencia que en el monitor serie se imprimirá la distancia obtenida gracias al sensor ultrasónico.

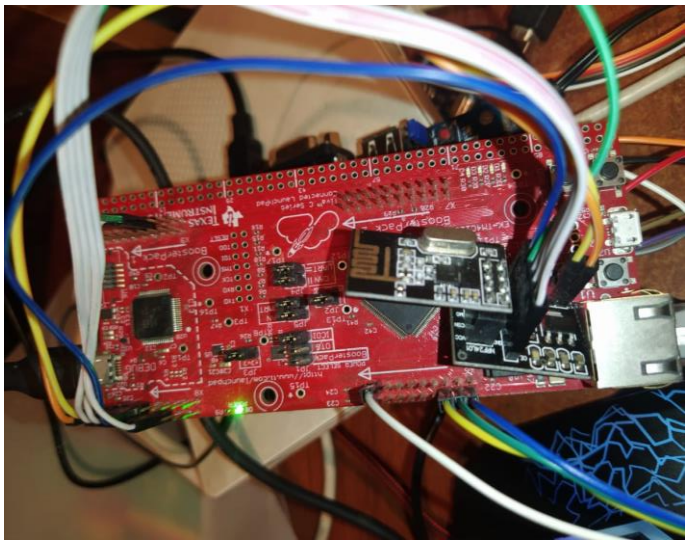
Figura 3.11: Monitor Serial de Arduino



El primer valor es la distancia obtenida gracias al sensor ultrasónico, los valores separados por los dos puntos son los valores medidos por los sensores LiDAR 1 y el sensor LiDAR 2 respectivamente, Autores: Chávez Andrés y Posso Luis

Una vez lograda la interconexión entre los 3 nodos esclavos con el nodo master para la comunicación, se eligió la topología tipo estrella ya que, al ser una arquitectura unidireccional, es decir que solo se envía datos desde el esclavo al maestro; cumplía con los requisitos necesarios para esta aplicación.

Figura 3.12: Modulo nRF24L01 conectado a la tarjeta TIVA



Conexión del módulo nRF24L01 con la TIVA C, Autores: Chávez Andrés y Posso Luis

Posteriormente cada nodo de la red inalámbrica consta de los siguientes elementos:

Figura 3.13 Esquema general de los componentes de cada nodo.

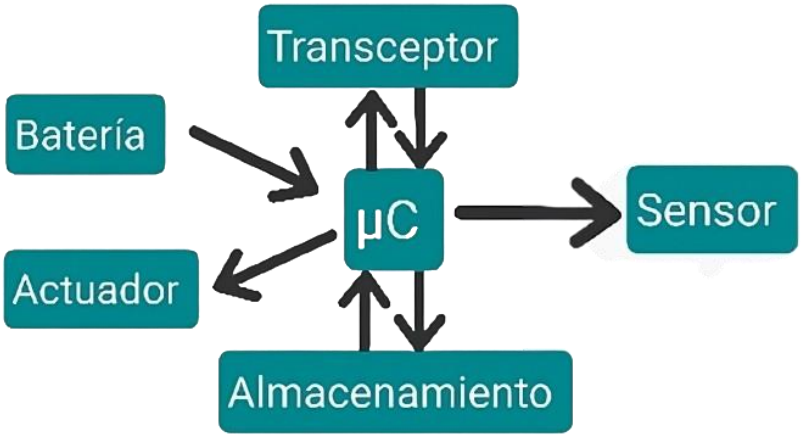
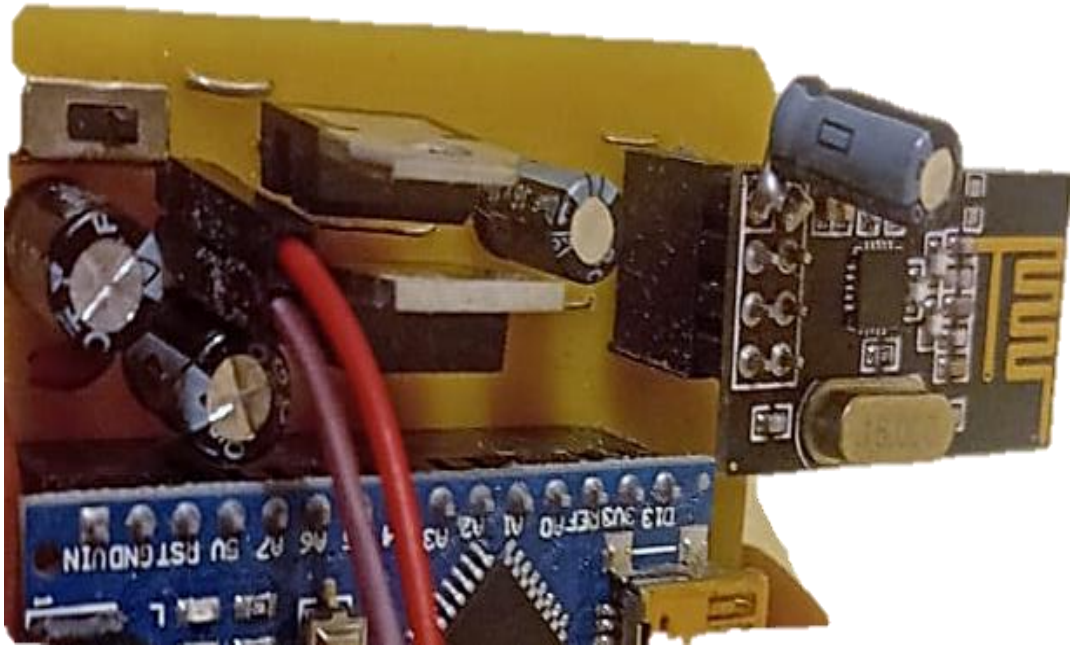


Diagrama de bloques de cada nodo, Autores: Chávez Andrés y Posso Luis.

- Bloque de Alimentación: Está compuesto por un regulador de voltaje de 9[V] y 1[A], adicionalmente se colocaron dos reguladores de voltaje el primero es el regulador 7805 a 5[V] y el segundo es el regulador LM1117T a 3.3[V]
- Adicionalmente se colocaron capacitores electrolíticos para el filtrado de los picos de voltaje. El diseño tiene 4 capacitores de los cuales 2 son de 220 uF, uno de 100 nF, y uno de 10 nF.

Figura 3.14: Condensadores y reguladores de voltaje de 3.3 V y 5 V

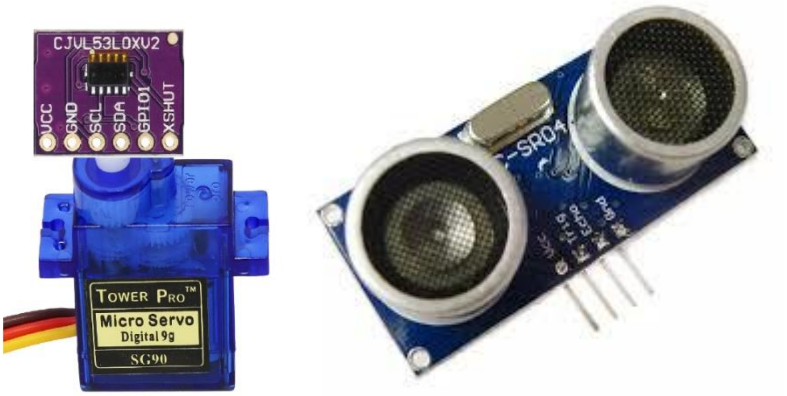


4 capacitores de los cuales 2 son de 220 uF, uno de100 nF, y uno de10 nF.

Autores: Chávez Andrés y Posso Luis

- Bloque de sensores: Tienen 2 sensores LiDAR para el muestreo de forma horizontal y 1 sensor ultrasónico para el muestro de forma vertical en la zona muerta.

Figura 3.15: Bloque de sensores



Fuente: Andrés Chávez y Posso Luis

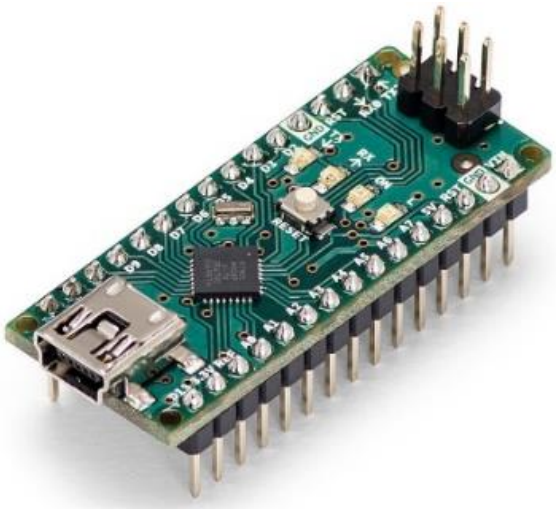
- Bloque de actuadores: Este bloque puede ser opcional, pero en este proyecto de titulación se usaron servomotores para poder girar cierta cantidad de grados al sensor LiDAR y se pueda obtener medidas en las zonas ciegas.

Figura 3.16: Bloque de actuadores

Servomotor unido al sensor LiDAR, Autores: Chávez Andrés y Posso Luis

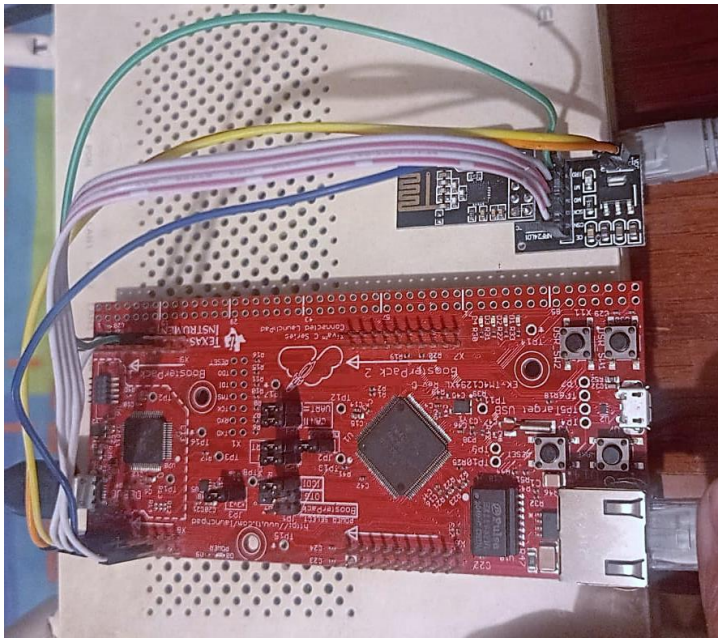
- Bloque de procesamiento: Este bloque se encarga de analizar y dar tratamiento a los datos obtenidos del bloque de sensores. Normalmente será un microprocesador el encargado de estas funciones. En este caso la placa de desarrollo Arduino Nano serán los que cumplan el rol de esclavo y la tarjeta de desarrollo TIVA C es toma el rol de maestro y también el encargado de enviar los datos al servidor web.

Figura 3.17: Bloque de procesadores



Nodo esclavo con la placa Arduino Nano, Autores: Chávez Andrés y Posso Luis

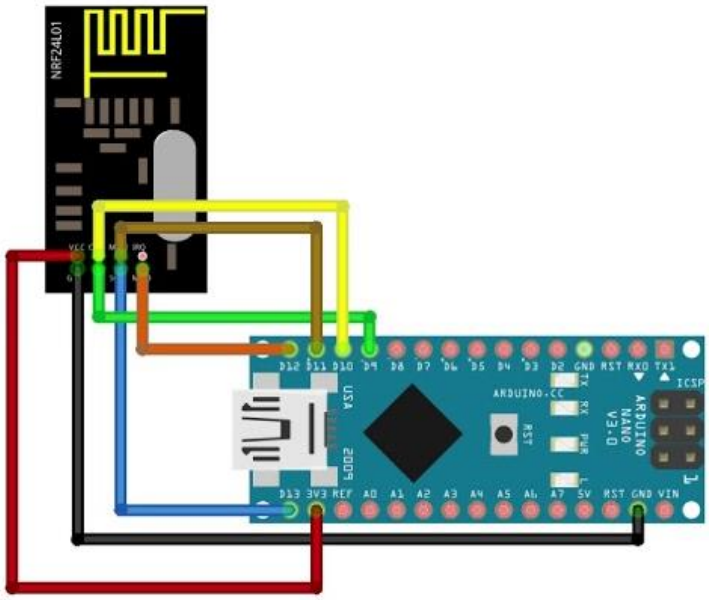
Figura 3.18: Bloque de procesadores



Nodo master con la placa TIVA C, Autores: Chávez Andrés y Posso Luis.

- Bloque Transceptor: Este bloque se encarga de realizar la transmisión de datos entre los 3 nodos esclavos y el nodo master, en este bloque se encuentra el módulo nRF24L01 cuyo medio de comunicación es la radiofrecuencia.

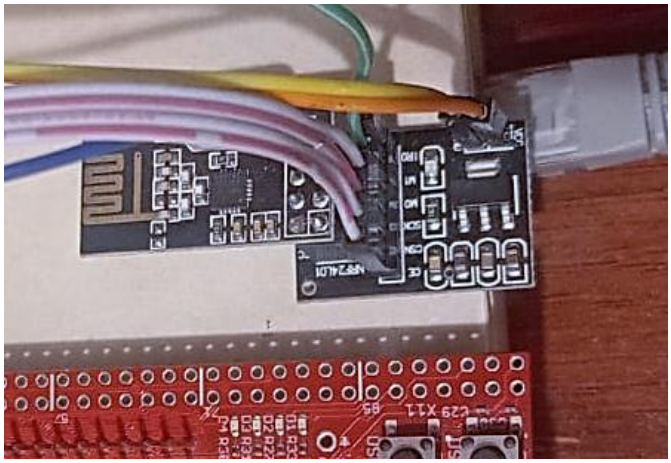
Figura 3.19: Bloque de transmisión



Módulo nRF24L01 conectado a Arduino Nano

Autores: Chávez Andrés y Posso Luis

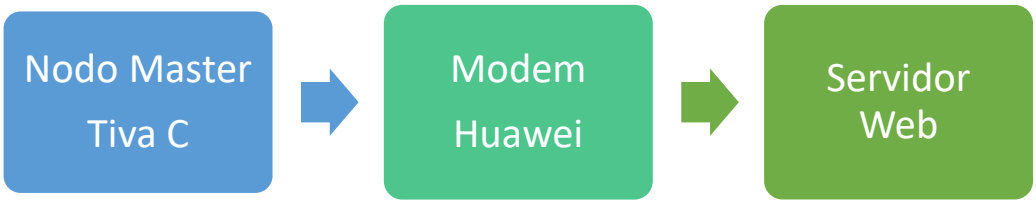
Figura 3.20: Bloque de transmisión



Módulo nRF24L01 conectado a TIVA C, Autores: Chávez Andrés y Posso Luis.

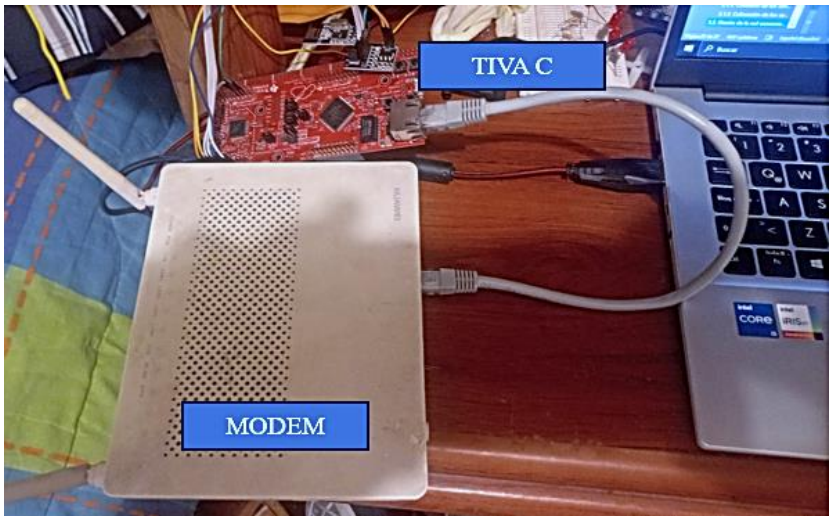
- Bloque de servidor Web: La tarjeta de desarrollo TIVA C se encuentra conectada a través de un cable UTP al modem que genera la red local. Se configura la dirección IP para lograr crear una página HTML.

Figura 3.21: Diagrama de flujo del bloque de Servidor Web



Fuente: Chávez Andrés y Posso Luis

Figura 3.22: Bloque de Servidor Web



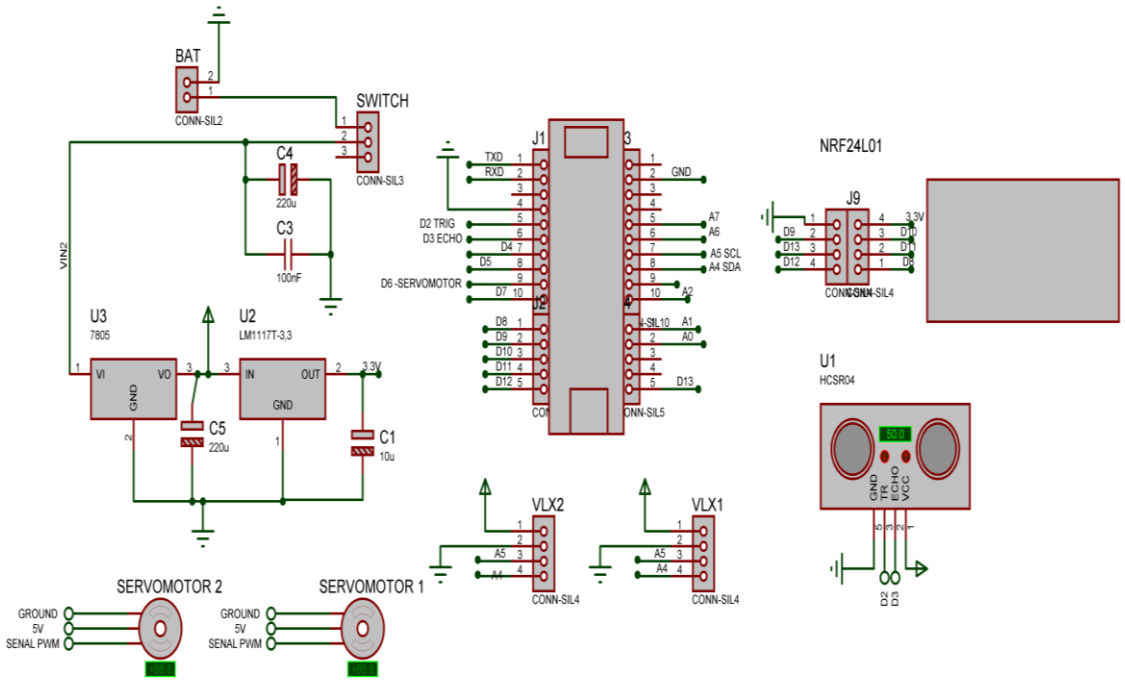
TIVA C conectado al Modem de la red local, Autores: Chávez Andrés y Posso Luis

3.3. Diseño de la placa PCB

3.3.1. Diseño esquemático del nodo en el programa “PROTEUS”

Para la creación y diseño de la tarjeta PCB se usó el programa Proteus, en el cual se realizó el diagrama esquemático con todos los componentes. Se realizó las conexiones entre los componentes para que al momento de usar la herramienta “PCB Layout” el enrutamiento de las pistas se lo haga de manera automatizada.

Figura 3.23: Diagrama esquemático del nodo esclavo

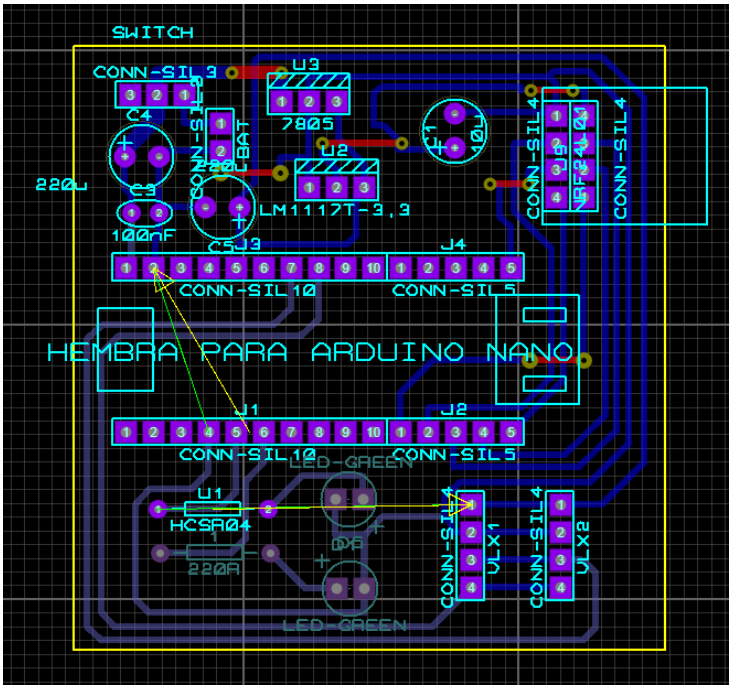


Autores: Chávez Andrés y Posso Luis

3.3.2. Diseño de la PCB

Con la opción “PCB Layout” se creó la PCB, de acuerdo a la figura 3.27 se puede referenciar las siguientes situaciones: se delimita la placa con los bordes de color amarillo, posterior a esto se debe colocar los componentes dentro de los límites de una manera conveniente en donde no se crucen las pistas. Luego con la herramienta “Auto Ruter” dejamos que se tracen las pistas de manera automática.

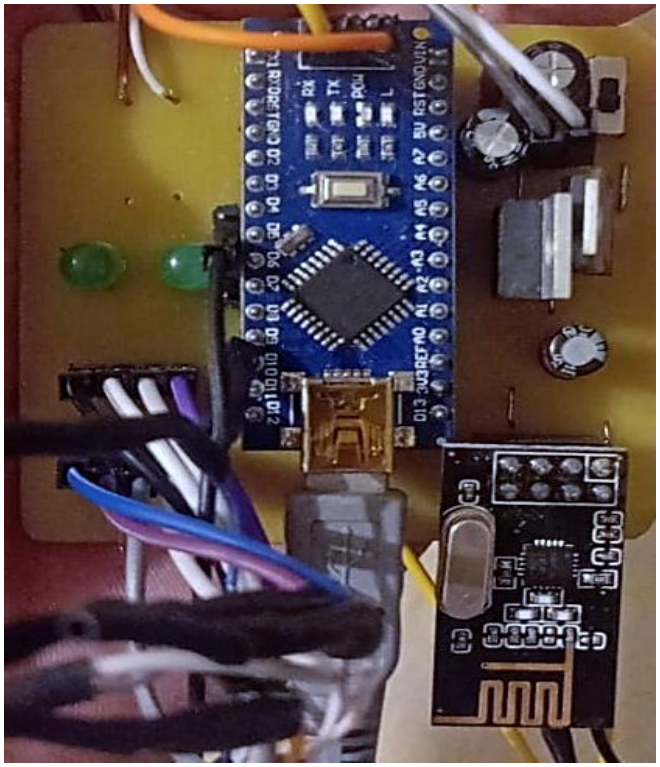
Figura 3.24: Diagrama del PCB del nodo esclavo



Diseño de pistas del PCB, Fuente: Chávez Andrés y Posso Luis

Este diseño se envió a la tienda “Megatrónica” para la elaboración de la placa.

Figura 3.25: PCB del nodo esclavo

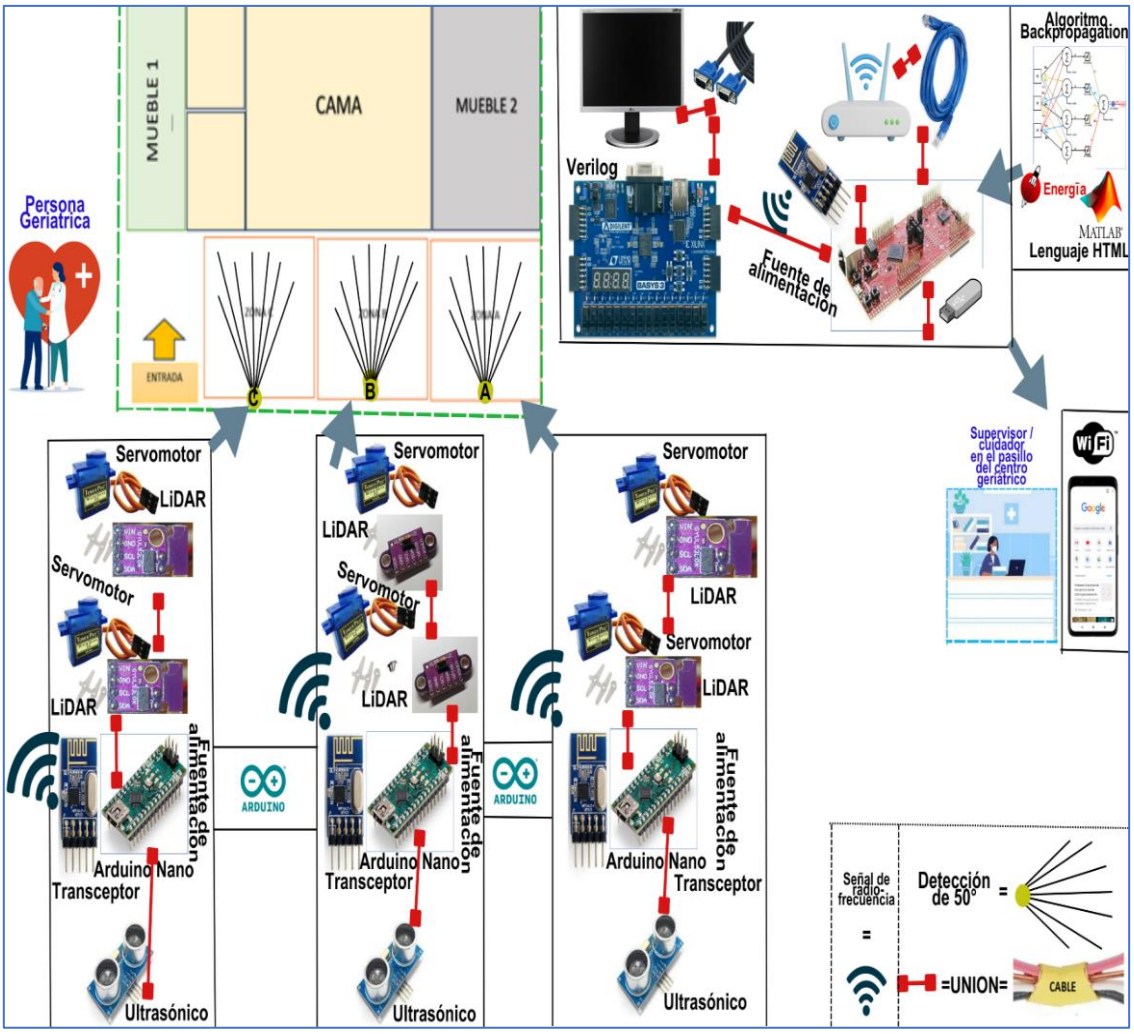


Diseño final de la PCB para el nodo esclavo, Fuente: Chávez Andrés y Posso Luis

3.4. Esquema general del desarrollo, funcionamiento y ubicación de cada uno de los dispositivos instalados.

La figura 3.26 describe de forma gráfica los bloques que componen el sistema de detección de caída de personas geriátricas.

Figura 3.26: Esquema general del sistema de detección de caídas para personas geriátricas.



Fuente: Chávez Andrés y Posso Luis

3.5. Diseño de la Red Neuronal Artificial

Para la implementación de la red neuronal artificial se partió de una tabla predeterminada que permita organizar las entradas que existen, así como también las salidas deseadas.

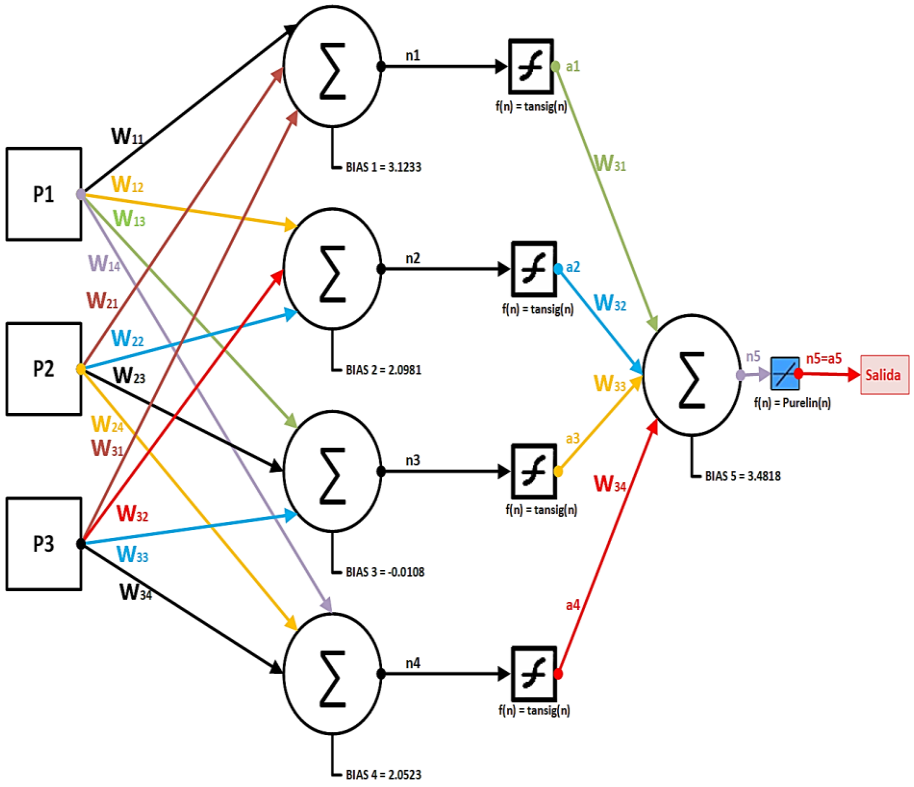
Tabla 3.2 Datos para la creación de la NNA

Entrada			Salida	
LIDAR 1 P1	LIDAR 2 P2	Ultrasónico P3	Mensaje de Salida	Valor
0	0	0	Persona ausente	1
0	0	1	Revisar sensores, error en lectura	2
0	1	0	Alerta, persona caída	3
0	1	1	Alerta, persona caída	4
1	0	0	Revisar sensor LIDAR 1 , Error en la medición.	5
1	0	1	Revisar sensor LIDAR 1 , Error en la medición.	6
1	1	0	Persona parada	7
1	1	1	Persona parada	8

Datos para la creación de la NNA, Autores: Chávez Andrés y Posso Luis.

Una vez realizada la tabla de datos de la red neuronal se puede determinar la arquitectura de la red, y se obtuvo la siguiente red neuronal:

Figura 3.27: Arquitectura de la red neuronal.

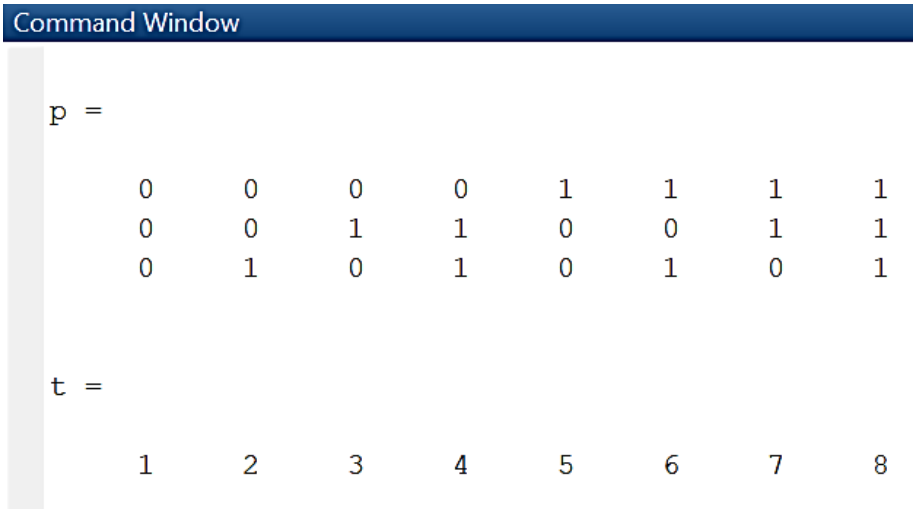


Arquitectura de la red neuronal, Fuente: Chávez Andrés y Posso Luis.

La red posee 3 entradas con 1 capa compuesta de 4 neuronas con función de activación tangente hiperbólica y cuyas salidas salen a una neurona de salida cuya función de activación es la función lineal.

El entrenamiento se realizó mediante el software de Matlab. Este software posee herramientas muy fuertes para el campo de las redes neuronales. Como primer paso se establecen las matrices de entradas y el vector de salida que deseamos. En este caso la matriz de entradas se la denominó con la letra 'p' y el vector salida se la denominó con la letra 't'.

Figura 3.28. Matrices de entrada y salida en el software de Matlab.



Fuente: Chávez Andrés y Posso Luis

Una vez establecidas las matrices a través de los comandos de la figura 3.29 se establece que con el comando 'tansig' que la función de activación será la tangente hiperbólica para las neuronas de la primera capa y el comando 'purelin' será la función de activación para la neurona de salida. El comando 'traincgf' establece que se use el algoritmo de retroalimentación por propagación.

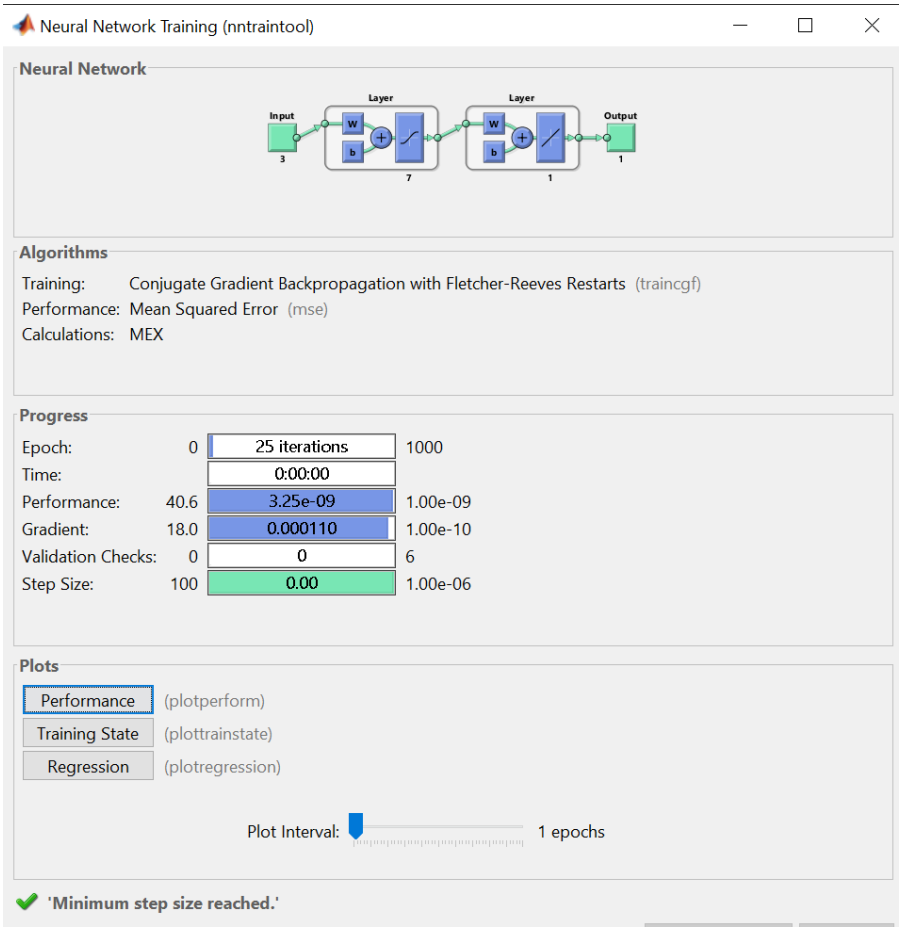
Figura 3.29. Comandos de Matlab para la creación de la red neuronal.

```
net2=newff(minmax(p),[2 1],{'tansig','purelin'},'traincgf');
```

Comandos de Matlab para la creación de la red neuronal, Fuentes: Chávez Andrés y Luis Posso

A través de la herramienta de Matlab llamada Neural Network Training (nntraintool) se puede determinar automáticamente que tipo de entrenamiento se puede usar para obtener las salidas deseadas, en este caso el tipo de entramiento es con backpropagation y gradiente conjugado. Para la estimación del error se usó el método del error medio cuadrático. Gracias a esta herramienta también se pueden determinar los pesos para cada neurona respectivamente. Se crearon 7 redes para poder comparar las salidas y observar cual red se acerca más a la salida real.

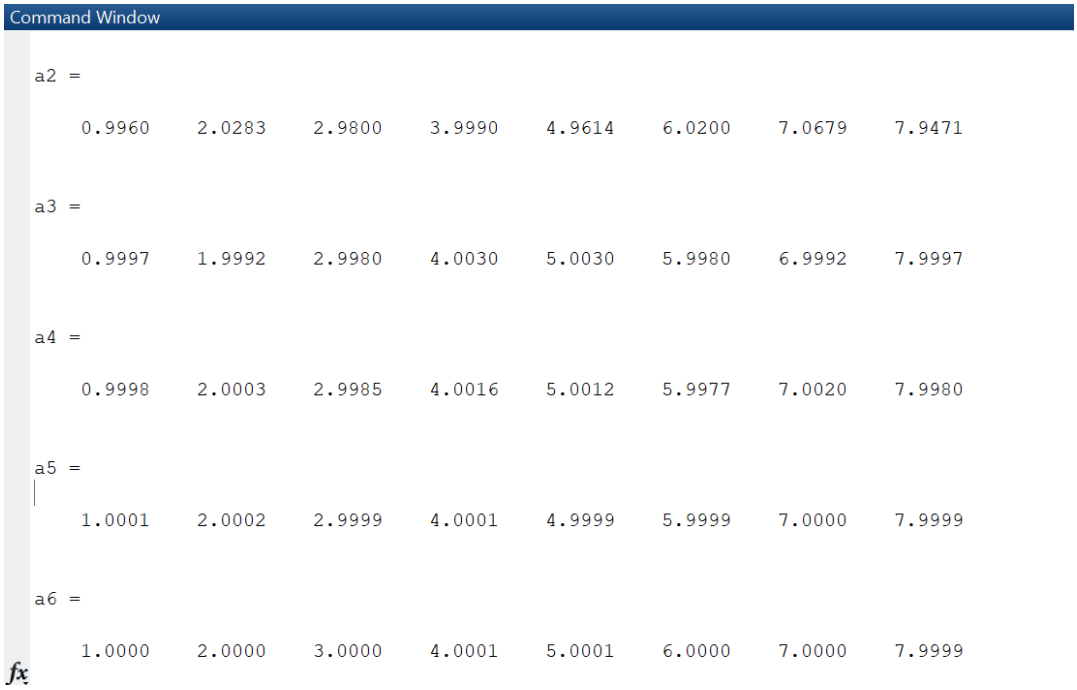
Figura 3.30. Herramienta "Neural Network Training (nntraintool)"



Entrenamiento de la red con la selección automática de algoritmo de entrenamiento y también el cálculo del error, Autores: Chávez Andrés y Posso Luis

Como se puede observar, con 25 iteraciones la red pudo obtener las salidas deseadas y con un error aproximado de 3.25e-09 lo cual es un error super bajo y aceptable. Se realizaron 6 tipos de redes de las cuales cada red tuvo sus respectivas salidas.

Figura 3.31. Vectores de salida de las diferentes redes neuronales



Vectores de salida de las diferentes redes neuronales, Autores: Chávez Andrés y Luis Posso

Se selecciona el vector salida que más se acerca a la salida deseada y con el comando de la figura 3.32 se obtienen los pesos y bias de cada neurona.

Figura 3.32: Algoritmo para obtener los pesos y bias de las neuronas que conforman la red neuronal.

```
pesos4_1=net4.iw{1,1}
bias4_1=net4.b{1,1}
pesos4_2=net4.lw{2,1}
bias4_2=net4.b{2,1}
```

Autores: Chávez Andrés y Posso Luis

Una vez obtenido los valores se puede escribir en código en lenguaje C, las respectivas operaciones matemáticas para implementar la red neuronal en la tarjeta TIVA. El inconveniente que se presentó fue que el IDE de Energía no posee la función tangente hiperbólica por lo que se tuvo que acudir a su función equivalente es decir la ecuación 2.1.

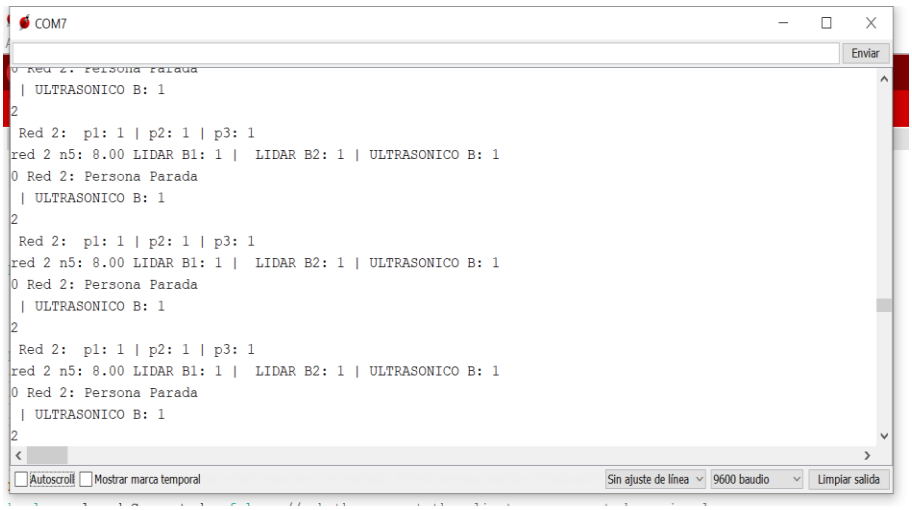
Figura 3.33: Código fuente de la red neuronal en lenguaje C.


```
//----CAPA OCULTA CON 4 NEURONAS----------------------
//n1=P1*W11+P2*W21+P3*W31+BIAS1
  n1= p1*(-3.2511)+p2*(-1.4598)+p3*(2.8150)+(3.1233);
//n2=P1*W12+P2*W22+P3*W32+BIAS2
  n2= p1*(-4.9281)+p2*(-0.2071)+p3*(0.0162)+(2.0981);
//n3=P1*W13+P2*W23+P3*W33+BIAS3
  n3= p1*(2.4710)+p2*(-1.1276)+p3*(3.5718)+(-0.0108);
//n4=P1*W14+P2*W24+P3*W34+BIAS4
  n4= p1*(-0.1218)+p2*(-4.3438)+p3*(1.9248)+(2.0523);

//-------FUNCION DE ACTIVACION TANSIG --------------
  a1= tansig(n1);
  a2= tansig(n2);
  a3= tansig(n3);
  a4= tansig(n4);

//------CAPA DE SALIDA - FUNCION DE ACTIVACION LINEAL-------

//n5= a1*W31+a2*W32+a3*W33+a4*W34+BIAS5
  n5=a1*(0.9346)+a2*(-2.0399)+a3*(1.0354)+a4*(-1.4699)+(3.4818);
```

Autores: Chávez Andrés y Posso Luis

En la figura 3.34 se demuestra los resultados que salen de la red neuronal, de acuerdo a los valores de entrada que se estén tomando.

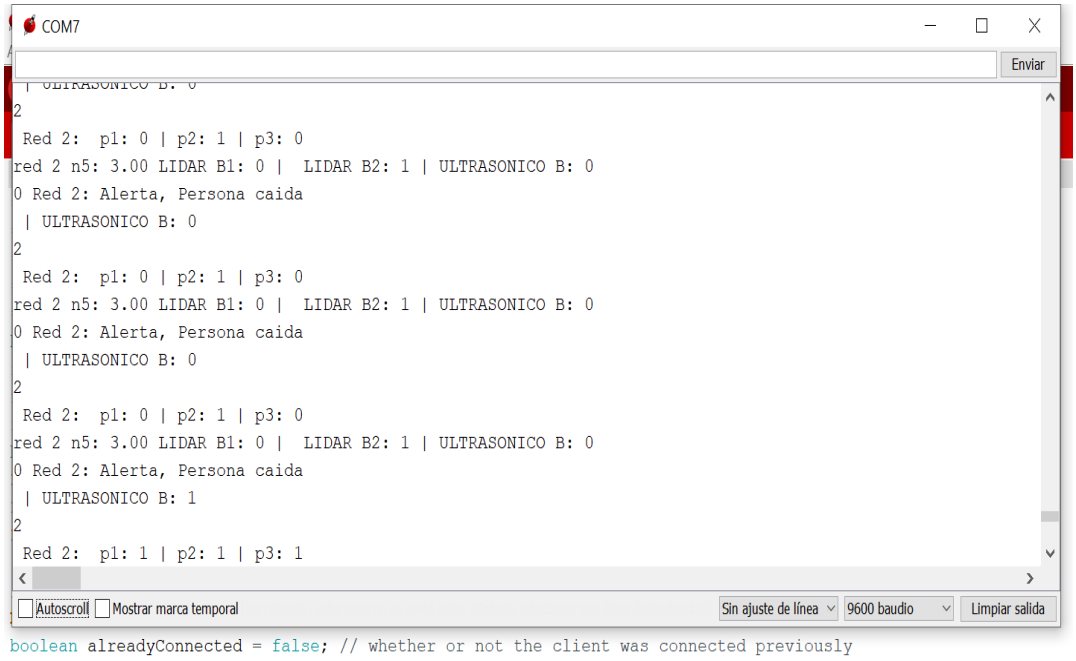
Figura 3.34: Salida en el monitor Serial de la red Neuronal.



Autores: Chávez Andrés y Posso Luis

Las entradas de los valores p1, p2 y p3 son iguales a 1 y de acuerdo a la tabla 3.2 si las entradas de los 3 sensores se iba a obtener un valor de salida n5=8 lo que es equivalente a “Persona parada”. Por lo tanto, la red neuronal está funcionando.

Figura 3.35: Salida en el monitor Serial de la red Neuronal.



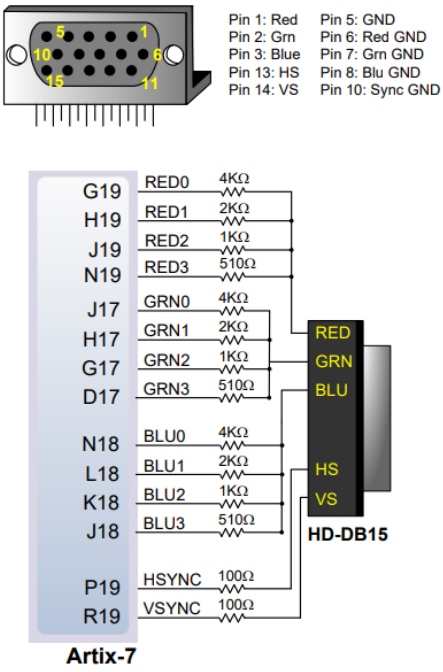
Autores: Chávez Andrés y Posso Luis

En la figura 3.35 la salida de la red neuronal n5 es igual a 3 y al momento de comparar con la tabla 3.2 la salida “Alerta persona caída” coincide con los valores de entrada de la tabla.

3.6. Diseño de la interfaz gráfica 2D con la tarjeta FPGA Basys 3

El desarrollo de interfaz gráfica se hizo con el programa Vivado 2020.1 para la programación de la tarjeta Basys 3 en lenguaje Verilog; la placa Basys 3 dispone de una entrada VGA entonces se puede usar un monitor con la misma entrada. El monitor VGA funciona a través de 5 señales: 3 son los colores RGB, las otras dos señales son las de sincronización horizontal y sincronización vertical. La tarjeta Basys tiene una gran variedad de colores, pero por motivos de facilidad se van a usar únicamente 3 bits para la creación de colores por lo cual solo se obtendrá una gama de 8 colores. Las señales de sincronización horizontal y vertical (hsync y vsync) son usadas para controlar la tasa de escaneo. hsync analiza el tiempo que toma escanear una fila, mientras que vsync establece el tiempo que demora en escanear la pantalla completa.

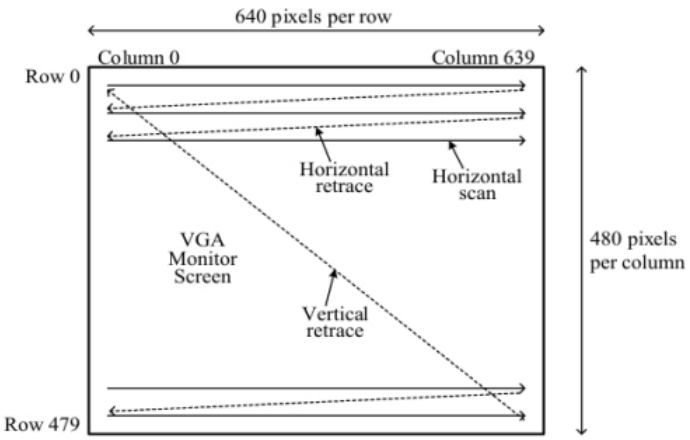
Figura 3.36: Puerto VGA de la tarjeta Basys 3



Fuente: DILIGENT

El sistema se compone de dos bloques, el primer: genera el tiempo y las señales sincronizadas, h_count_reg y v_count_reg y al estar a una taza de refresco diferente toca crear un contador que permita transformar los 100 MHz de la tarjeta a 25MHz generando una señal llamada p_tick. Esta tasa de refresco se la utiliza porque es compatible con la resolución de 640x480 pixeles. Las señales X y, indican la posición actual que se está escaneando y además especifican en que píxel se encuentra.

Figura 3.37 Pantalla VGA



Autor: Arévalo S., 2019

El segundo bloque: Es el encargado de producir los colores de pixeles en la pantalla, este es el que va permitir al usuario la interfaz. Primeramente, se crearon registros para cada elemento de la interfaz gráfica mostrada en la figura 3.38. para una mayor facilidad en la programación.

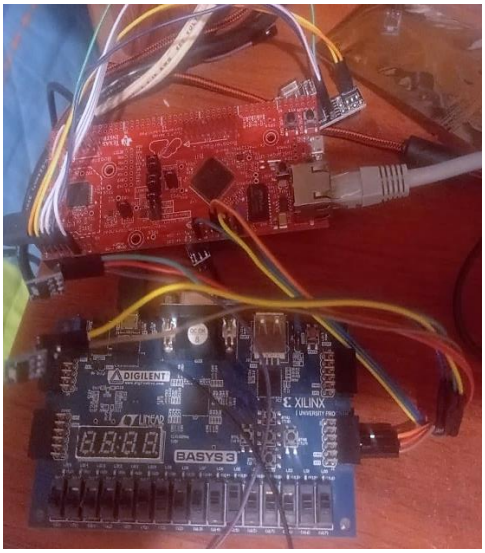
Figura 3.38 Creación de registros en lenguaje Verilog.

```
// Edredon
assign rect_x_19 = 77;                    // left boundary
assign rect_y_t9 = 25;                    // top boundary
assign rect_x_r9 = 397;    // right boundary
assign rect_y_b9 =250 ;    // bottom boundary

// Letra "I"
assign rect_x_111 = 46;                    // left boundary
assign rect_y_t11 = 445;                    // top boundary
assign rect_x_r11 = 60;    // right boundary
assign rect_y_b11 =476 ;    // bottom boundary

// lETRA "N" SUPERIOR
assign rect_x_122 = 65;                    // left boundary
assign rect_y_t22 = 445;                    // top boundary
assign rect_x_r22 = 100;    // right boundary
assign rect_y_b22 =455 ;    // bottom boundary

// lETRA "N" VERTICAL IZQUIERDA
assign rect_x_133 = 65;                    // left boundary
assign rect_y_t33 = 456;                    // top boundary
assign rect_x_r33 = 80;    // right boundary
assign rect_y_b33 =476 ;    // bottom boundary
```

Creación de registros en lenguaje Verilog Autores: Chávez Andrés y Posso Luis.

Una vez desarrollado el código de la interfaz gráfica, se usaron condicionales que activaran o desactivaran las zonas en la interfaz gráfica a través de señales enviadas desde los puertos digitales de salida de la TIVA C hacia las entradas digitales de la tarjeta FPGA Basys 3.

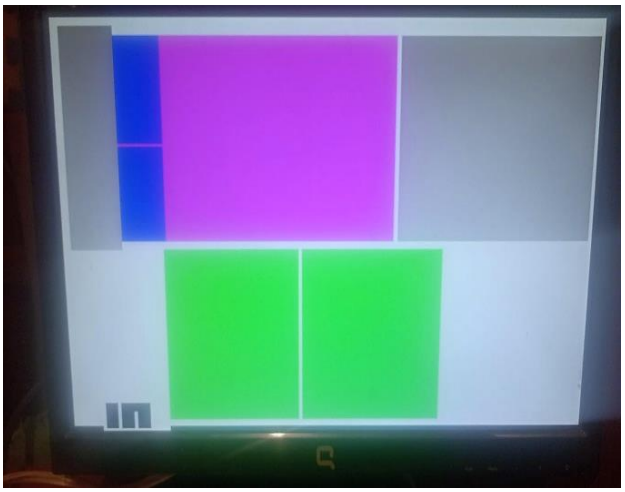
Figura 3.39 Conexión TIVA C con FPGA Basys 3



Conexión a través de los pines digitales de salida de la tarjeta TIVA C con los pines digitales de entrada de la tarjeta Basys 3, Autores: Chávez Andrés y Posso Luis.

Los recuadros que representan las zonas se van a pintar de color rojo en caso de que la red neuronal detecte una persona geriátrica caída mientras que si dicha persona se encuentra de pie este se va a pintar de color verde indicando que no existe ninguna alarma

Figura 3.40: Interfaz gráfica en 2D

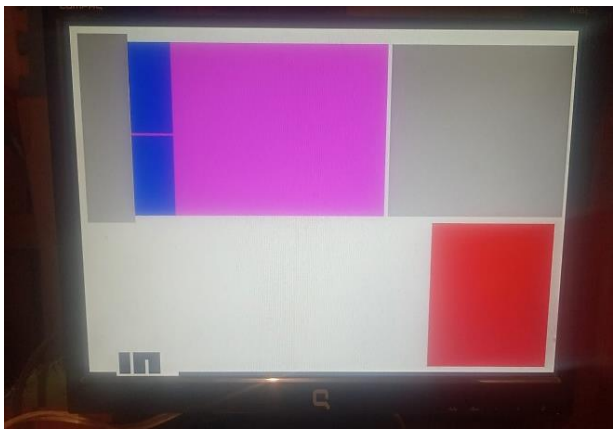


Interfaz gráfica en 2D. Autores: Chávez Andrés y Posso Luis

En la figura 3.40 se puede observar que los recuadros de las zonas están de color verde lo que significa que la persona está de pie y está en la zona C y B lo que permite deducir que se encuentra en una zona donde tuvo que activar el servomotor para detectar a la persona en la zona muerta.

En la figura 3.41 el recuadro de la zona A está de color rojo lo que significa que es una notificación de alerta de que la persona se encuentra caída.

Figura 3.41: Interfaz gráfica en 2D



Interfaz gráfica en 2D. Autores: Chávez Andrés y Posso Luis

3.7 Diseño del servidor Web

El servidor Web se encuentra en la dirección 192.168.100.70, cualquier usuario que solicite el ingreso deberá hacerlo desde cualquier dispositivo que pueda ejecutar un navegador web. La programación del entorno de la página se lo desarrolló en el IDE de Energía y es subido a la placa de desarrollo TIVA C.

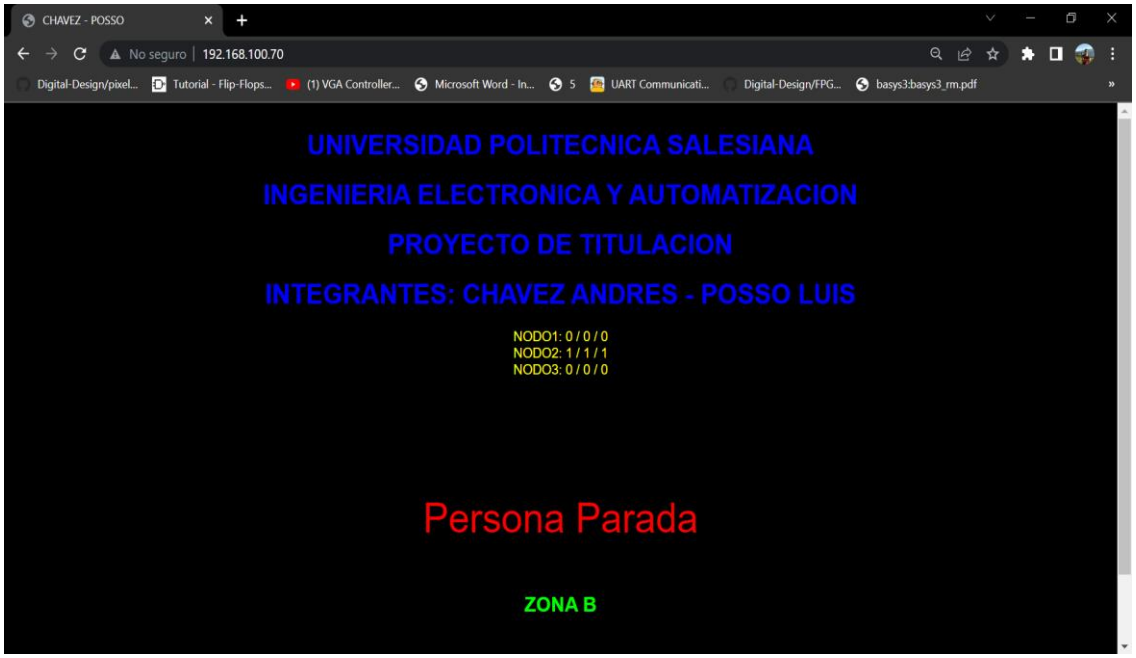
La facilidad del IDE de Energía es que simula el IDE de Arduino por lo tanto la programación del entorno del servidor web se hizo a base del lenguaje HTML y con la metodología experimental se fue modificando los elementos que la componen para crear un ambiente más amigable y entendible. Al ser una arquitectura unidireccional este solo recibe datos de los nodos, en el código se ha configurado un tiempo de actualización de 1 segundo la página web.

El diseño que se observa en la figura 3.46 se puede observar que se compone de un encabezado en donde se encuentran los datos del proyecto, en la segunda sección se encuentran enlistados los nodos con los valores de cada sensor, estos valores son binarios ya que solo se trabajan con ‘1’ si se detecta la presencia de la persona o ‘0’ si no detecta la presencia. En la siguiente sección se encuentra el mensaje que emitirá de acuerdo a la salida n5 proporcionada por la red neuronal, la siguiente sección de letras verdes indica la zona en la cual este detectando la presencia de la persona.

Figura 3.42 Código en lenguaje HTML para la creación de la interfaz del servidor web.

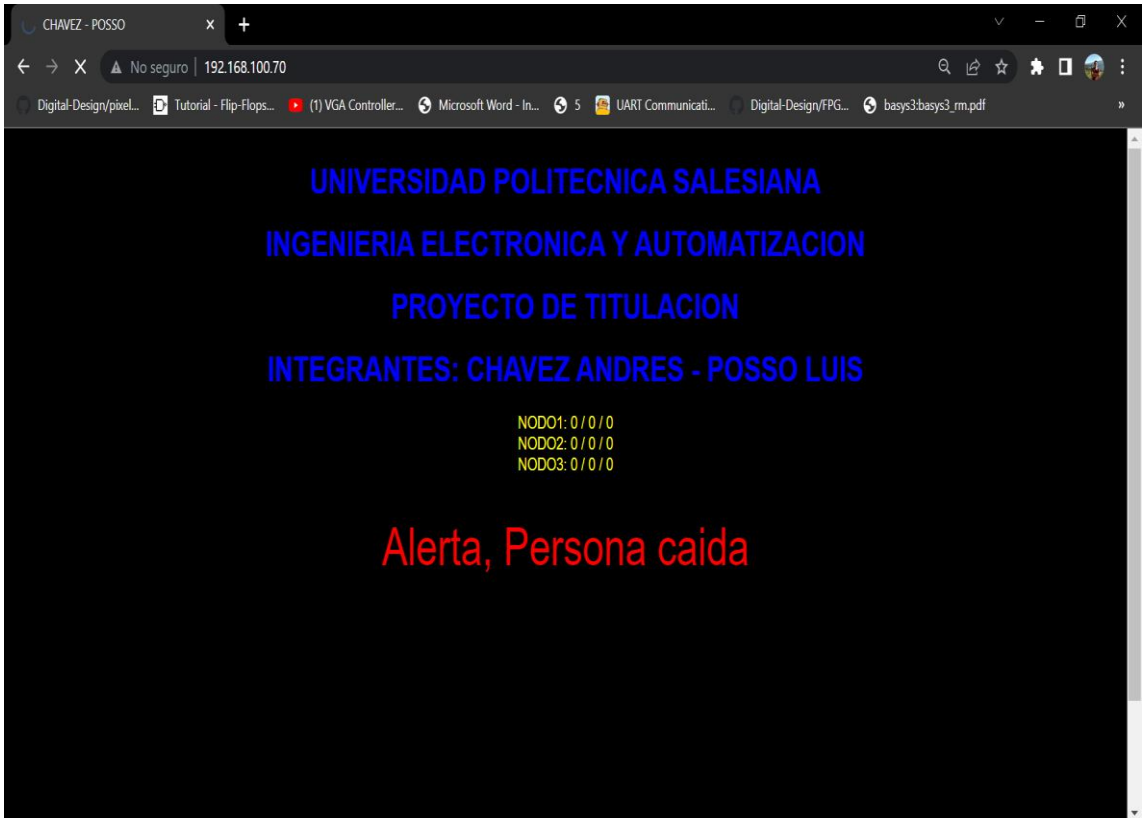
Código en lenguaje HTML para la creación de la interfaz del servidor web. Autores: Chávez Andrés y Posso Luis

Figura 3.43 Interfaz del servidor Web



Mensaje de Persona Parada en la zona B, Autores: Chávez Andrés y Posso Luis

Figura 3.44 Interfaz del servidor Web



Mensaje de Alerta de persona caída en la zona B, Autores: Chávez Andrés y Posso Luis

3.8. Tabla de resultados

Para verificar el funcionamiento del sistema de detección de caídas junto a una persona con condiciones de salud estable se realizaron movimientos similares al de una caída y registraron los resultados en las siguientes tablas. Además, se usará la ecuación 3.1 para el calcular porcentualmente el número de aciertos.

$$\%\ Aciertos = \frac{intentos\ totales - fallos}{intentos\ totales} * 100 \quad \text{Ec. 3.1}$$

Tabla 3.3 Registro de pruebas experimentales con la persona parada en las 3 zonas no muertas y a una distancia de 0.50 [m] de los sensores.

Zonas	Cantidad de pruebas experimentales	Detección de persona parada		Porcentaje de aciertos de persona parada
		Aciertos	Fallidos	
Zona A	90	80	10	88.88 [%]
Zona B	95	88	7	92.63[%]
Zona C	110	90	20	81.81[%]

Fuente: Chavez Andrés y Posso Luis.

Con la persona parada respectivamente en las 3 zonas no muertas y a una distancia de 0.50 [m] de los sensores se registró: en la zona A de un total de 90 pruebas experimentales se obtuvo un 88.88[%] de aciertos en el envío de notificación de persona parada.

En la zona B de un total de 95 pruebas experimentales se obtuvo un 92.63[%] de aciertos en el envío de notificación de persona parada. En la zona C de un total de 110 pruebas experimentales se obtuvo un 81.81[%] de aciertos en el envío de notificación de persona parada; todos estos valores se encuentran registrados en la tabla 3.3.

Tabla 3.4 Registro de pruebas experimentales con la persona parada en las 3 zonas no muertas y a una distancia de 1 [m] de los sensores.

Zonas	Cantidad de pruebas experimentales	Detección de persona parada		Porcentaje de aciertos de persona parada
		Aciertos	Fallidos	
Zona A	90	74	16	82.22[%]
Zona B	95	80	15	84.21[%]
Zona C	110	88	22	80[%]

Fuentes: Chávez Andrés y Posso Luis.

Realizando pruebas experimentales con la persona parada en las 3 zonas no muertas y a una distancia de 1 [m] de los sensores se registró: en la zona A de un total de 90 pruebas experimentales se obtuvo un 82.22[%] de aciertos en el envío de notificación de persona parada. En la zona B de un total de 95 pruebas experimentales se obtuvo un 84.21[%] de aciertos en el envío de notificación de persona parada. En la zona C de un total de 110 pruebas experimentales se obtuvo un 80[%] de aciertos en el envío de notificación de persona parada; todos estos valores se encuentran registrados en la tabla 3.4.

Tabla 3.5 Registro de pruebas experimentales con la persona parada en las 3 zonas muertas y a una distancia de 0.50 [m] de los sensores girados por los servomotores

Zonas	Cantidad de pruebas experimentales	Detección de persona parada		Porcentaje de aciertos de persona parada
		Aciertos	Fallos	
Zona A	90	69	21	76.66[%]
Zona B	95	84	11	88.42[%]
Zona C	110	87	23	79.09[%]

Fuente: Chavez Andrés y Posso Luis.

Realizando pruebas experimentales con la persona parada en las 3 zonas muertas y a una distancia de 0.50 [m] de los sensores girados por los servomotores se registró: en la zona muerta A de un total de 90 pruebas experimentales se obtuvo un 76.66[%] de aciertos en el envío de notificación de persona parada. En la zona muerta B de un total de 95 pruebas experimentales se obtuvo un 88.42[%] de aciertos en el envío de notificación de persona parada. En la zona muerta C de un total de 110 pruebas experimentales se obtuvo un 79.09[%] de aciertos en el envío de notificación de persona parada; todos estos valores se encuentran registrados en la tabla 3.5.

Tabla 3.6 Registro de pruebas experimentales con la persona parada en las 3 zonas muertas y a una distancia de 1 [m] de los sensores girados por los servomotores.

Zonas	Cantidad de pruebas experimentales	Detección de persona parada		Porcentaje de aciertos de persona parada
		Aciertos	Fallos	
Zona A	90	67	23	74.44[%]
Zona B	95	79	16	83.15[%]
Zona C	110	81	29	73.63[%]

Fuentes: Chávez Andrés y Posso Luis.

Con pruebas experimentales con la persona parada en las 3 zonas muertas a una distancia de 1 [m] de los sensores girados por los servomotores, se registró: en la zona muerta A de un total de 90 pruebas experimentales se obtuvo un 74.44[%] de aciertos en el envío de notificación de persona parada. En la zona muerta B de un total de 95 pruebas experimentales se obtuvo un 83.15[%] de aciertos en el envío de notificación de persona parada. En la zona muerta C con 110 pruebas experimentales se obtuvo un 73.63[%] de aciertos en el envío de notificación de persona parada; de acuerdo a la tabla 3.6.

Tabla 3.7 Registro de pruebas experimentales con la persona caída en las 3 zonas no muertas y a una distancia de 0.50 [m] de los sensores.

Zonas	Cantidad de pruebas experimentales	Detección de persona caída		Porcentaje de aciertos de persona caída
		Aciertos	Fallos	
Zona A	90	81	9	90[%]
Zona B	95	83	12	87.36[%]
Zona C	110	98	12	89.09[%]

Fuente: Chavez Andrés y Posso Luis.

Con pruebas experimentales con la persona caída en las 3 zonas no muertas y a una distancia de 0.50 [m] de los sensores, se registró: en la zona A de un total de 90 pruebas experimentales se obtuvo un 90[%] de aciertos en el envío de notificación de persona caída. En la zona B de un total de 95 pruebas experimentales se obtuvo un 87.36[%] de aciertos en el envío de notificación de persona caída. En la zona C de un total de 110 pruebas experimentales se obtuvo un 89.09[%] de aciertos en el envío de notificación de persona caída; todos estos valores se encuentran registrados en la tabla 3.7.

Tabla 3.8 Registro de pruebas experimentales con la persona caída en las 3 zonas no muertas y a una distancia de 1 [m] de los sensores.

Zonas	Cantidad de pruebas experimentales	Detección de persona caída		Porcentaje de aciertos de persona caída
		Aciertos	Fallos	
Zona A	90	76	14	84.44[%]
Zona B	95	71	24	74.73[%]
Zona C	110	89	21	80.90[%]

Fuentes: Chávez Andrés y Posso Luis.

Realizando pruebas experimentales con la persona caída en las 3 zonas no muertas y a una distancia de 1 [m] de los sensores, se registró: en la zona A de un total de 90 pruebas experimentales se obtuvo un 84.44[%] de aciertos en el envío de notificación de persona caída. En la zona B de un total de 95 pruebas experimentales se obtuvo un 74.73[%] de aciertos en el envío de notificación de persona caída. En la zona C de un total de 110 pruebas experimentales se obtuvo un 80.90[%] de aciertos en el envío de notificación de persona caída; todos estos valores se encuentran registrados en la tabla 3.8.

Tabla 3.9 Registro de pruebas experimentales con la persona caída en las 3 zonas muertas y a una distancia de 0.50 [m] de los sensores girados por los servomotores.

Zonas	Cantidad de pruebas experimentales	Detección de persona caída		Porcentaje de aciertos de persona caída
		Aciertos	Fallos	
Zona A	90	78	12	86.66[%]
Zona B	95	79	16	83.15[%]
Zona C	110	88	22	80[%]

Fuente: Chavez Andrés y Posso Luis.

Pruebas experimentales con la persona caída en las 3 zonas muertas y a 0.50 [m] de distancia de los sensores, se registró: en la zona muerta A de un total de 90 pruebas experimentales se obtuvo un 86.66[%] de aciertos en el envío de notificación de persona caída. En la zona B de un total de 95 pruebas experimentales se obtuvo un 83.15[%] de aciertos en el envío de notificación de persona caída. En la zona C de un total de 110 pruebas experimentales se obtuvo un 80[%] de aciertos en el envío de notificación de persona caída; todos estos valores se encuentran registrados en la tabla 3.9.

Realizando pruebas experimentales con la persona caída en las 3 zonas muertas y a una distancia de 1 [m] de los sensores girados por los servomotores, se registró: en la zona muerta A de un total de 90 pruebas experimentales se obtuvo un 64.44[%] de aciertos en el envío de notificación de persona caída. En la zona B de un total de 95 pruebas experimentales se obtuvo un 66.31[%] de aciertos en el envío de notificación de persona caída. En la zona C de un total de 110 pruebas experimentales se obtuvo un 72.72[%] de aciertos en el envío de notificación de persona caída; todos estos valores se encuentran registrados en la tabla 3.10.

Tabla 3.10 Registro de pruebas experimentales con la persona caída en las 3 zonas muertas y a una distancia de 1 [m] de los sensores girados por los servomotores.

Zonas	Cantidad de pruebas experimentales	Detección de persona caída		Porcentaje de aciertos de persona caída
		Aciertos	Fallos	
Zona A	90	58	32	64.44[%]
Zona B	95	63	32	66.31[%]
Zona C	110	80	30	72.72[%]

Fuentes: Chávez Andrés y Posso Luis.

CAPÍTULO 4
CONCLUSIONES

El sistema de detección de caídas de personas geriátricas está diseñado para un cuarto con un corredor dividido en 3 zonas de 0.90x0.90 [m^2] respectivamente; la WSN se conforma por un nodo maestro (Placa TIVA C) y 3 nodos esclavos (Arduino Nano); cada nodo contiene: 2 sensores LiDAR ubicados a 0.20[m] y 1.20[m] correspondientemente, un servomotor SG90, un módulo de radiofrecuencia nRF24L01 y un sensor ultrasónico HC-SR04. Para la detección de una persona geriátrica caída se usa la red neuronal compuesta por 4 neuronas en la capa oculta y 1 neurona en la capa de salida, esta red se encuentra implementada en la tarjeta TIVA para que por medio de sus puertos ethernet y puertos digitales envíe la notificación en tiempo real al servidor web y también la localización y escaneo de la persona geriátrica por medio de la interfaz gráfica en 2D programada en lenguaje de alto nivel Verilog en la placa FPGA Basys 3.

El sistema LiDAR está diseñado con los sensores VL53LXX-V2 y GYUL53L0X, con un alcance máximo de 0.90[m] de distancia y una apertura de medición de 12 grados para el escaneo de una persona geriátrica en caso de una caída. Se colocaron servomotores en los sensores LiDAR ya que al caer la persona geriátrica en la zona muerta este puede girar, escanear y localizar apropiadamente para remitir las notificaciones adecuadas al personal encargado de la supervisión. Por ende, el sistema es operativo y funcional para la detección de caídas de las personas geriátricas. El Arduino Nano de acuerdo a la distancia medida, envía por el módulo de radiofrecuencia (nRF24L01) valores binarios (cero o uno) hacia la TIVA C (nodo maestro).

La red neuronal está diseñada con 4 neuronas activadas por la función tangente hiperbólica en la capa oculta y 1 neurona activada por la función lineal en la capa de salida; que se encuentra entrenada por el algoritmo Backpropagation; La red se encuentra implementada a través de operaciones aritméticas en la tarjeta TIVA C, y se encarga de actualizar los valores de entrada de la red neuronal a la hora de la detección de una persona levantada pero sobre todo cuando la persona geriátrica sufre una caída en las zonas establecidas.

Por medio de la FPGA Basys 3 y su programación en lenguaje de alto nivel Verilog, se establece 3 módulos: vga_controller, pixel_gen y top; las cuales remiten las señales de conteo, sincronización y generación de píxeles al puerto VGA para que se proyecte en un monitor en tiempo real un cuadro rojo si la persona está caída o un cuadro verde si la persona está parada en cada una de las zonas del entorno controlado.

La arquitectura unidireccional está compuesta por la tarjeta TIVA C programada con la función "server" en lenguaje HTML para enviar la notificación por medio del puerto Ethernet hacia el módem; y éste a su vez hacia el servidor web; Además la conexión unidireccional de la placa TIVA C con la FPGA Basys 3 es por medio de sus puertos digitales; todo esto con la finalidad de generar notificaciones con un tiempo de aproximadamente 3 segundos desde que se detecta la caída de la persona geriátrica.

Para la verificación se realizaron un promedio de 98 pruebas experimentales en el entorno controlado, de los cuales se observa un 84.58[%] de aciertos de una persona parada y un 86.043[%] de aciertos de una persona caída a 0.50 [m] de distancia de los sensores LiDAR con la persona geriátrica; mientras que un 79.60[%] de aciertos de una persona parada y 73.92[%] de aciertos de una persona caída a una distancia de 1[m] de los sensores LiDAR hasta la persona detectada. Por lo tanto, se puede concluir que los sensores LiDAR tienen un mejor funcionamiento de escaneo con distancias menores a 0.90[m].

RECOMENDACIONES

- Los sensores LiDAR deben estar colocados para una medición máxima de 0.90[m] de distancia.
- Los sensores ultrasónicos deben estar colocados a 2.20[m] de distancia del suelo.
- La placa PCB debe ser alimentada con un adaptador de 9[V] y 1[A].
- La velocidad de internet debe ser superior a los 27 Mbps para el correcto funcionamiento del servidor web.
- Trabajar con un modem Wifi con doble antena, mínimo de hace 2 años salido al mercado, para el correcto funcionamiento y mayor cobertura de la red local.
- Se debe cargar el archivo con extensión .bit en una USB con formato fat32 y colocar los jumpers JP1 en posición USB para la programación de la FPGA Basys 3.
- Evitar la colocación de objetos y/o dispositivos emisores de señales inalámbricas que interfieran con la comunicación de radiofrecuencia de la WSN.

REFERENCIAS BIBLIOGRÁFICAS

Albornoz, M. C., Berón, M., & Montejano, G. A. (2017, September). *Interfaz gráfica de usuario: el usuario como protagonista del diseño*. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

Álvarez Carulla, A. (2022). *MASB (Arduino): Comunicación serie IV*. Barcelona. España. URL: https://diposit.ub.edu/dspace/handle/2445/197241

Andrade, J. Redrován, H. (2016). *Diseño e implementación de un sistema de detección y notificación de caídas en personas de la tercera edad.* [Tesis de grado, Universidad Politécnica Salesiana]. Archivo digital. https://dspace.ups.edu.ec/bitstream/123456789/11805/1/UPS-CT005580.pdf

Arduino. (2023). *Arduino Nano.* URL: https://store-usa.arduino.cc/products/arduino-nano?selectedStore=us

Badgujar, S. y Pillai, A. (2020). Fall Detection for Elderly People using Machine Learning. *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT),* pp. 1-4. URL: https://bibliotecas.ups.edu.ec:2095/document/9225494

Alarcón Contreras, R., y Flores Marín, M. (2017). Sistemas domóticos para adultos mayores con movilidad reducida. *INVESTIGATIO, (6)*, 83–98. https://doi.org/10.31095/irr.v0i6.25

Díaz, P. Yabar, J. (2017). *Diseño e implementación de un sistema domótico para aplicaciones en pacientes parapléjicos mediante control remoto a través de internet y reconocimiento de voz.* [Tesis de pregrado, Universidad Ricardo Palma]. Repositorio institucional. URL: http://repositorio.urp.edu.pe/bitstream/handle/URP/3355/1%26TSistemaDomoticoSoporte.pdf?sequence=1&isAllowed=y

DILIGENT. (2016). *Basys 3™ FPGA Board Reference Manual*. Versión C. URL: https://digilent.com/

Flores, J. Mahecha, D. (2020). *Sistema domótico para adultos mayores con dependencia.* [Tesis de pregrado, Universidad Católica De Colombia]. Repositorio institucional. URL: https://repository.ucatolica.edu.co/bitstream/10983/25545/1/T.%20Grado.pdf

Forttes, P. (2020). Envejecimiento y atención a la dependencia en ECUADOR. *Banco Interamericano de Desarrollo, División de Protección Social y Salud.* URL: https://www.gerontologia.org/portal/archivosUpload/uploadManual/Envejecimiento-y-atencion-a-la-dependencia-en-Ecuador.pdf

Gagliardi, T. (2019). *Análisis de la comunicación de radio frecuencia con módulos nRF24L01 (Doctoral dissertation*, Universidad Nacional de La Plata). URL: http://sedici.unlp.edu.ar/handle/10915/89815

Gonzáles, R. Hernández, R. Jiménez, B. (2016). *Desarrollo de un sistema de detección de caídas.* [Tesis de pregrado, Universidad Complutense de Madrid]. Archivo digital. https://eprints.ucm.es/id/eprint/38704/1/MemoriaTFG.pdf

González, S. A. *Prototipo de interfaz gráfica descrito en FPGA*. URL: https://www.researchgate.net/profile/Santiago-Arevalo-Gonzalez/publication/280529258_Prototipo_de_interfaz_grafica_descrito_en_FPGA/links/55b7896b08aed621de046669/Prototipo-de-interfaz-grafica-descrito-en-FPGA.pdf

Hassan, A. Gumaei, G. Aloj, G. Fortino, G. y Zhou, M. (2019). A Smartphone-Enabled Fall Detection Framework for Elderly People in Connected Home Healthcare. *IEEE Network, vol. 33, no. 6.* pp. 58-63. URL: https://ieeexplore.ieee.org/abstract/document/8933560

Ismail, Y. (2019). *Internet of Things (IoT) for Automated and Smart Applications.* Intechopen Limited. URL: https://books.google.es/books?id=O0P8DwAAQBAJ&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Miller, T. Mejía, I. *El envejecimiento de la población en Ecuador: la revolución silenciosa.* Consejo Nacional para la Igualdad Intergeneracional. https://www.igualdad.gob.ec/wp-content/uploads/downloads/2020/05/el_envejecimiento_poblacion_ecuador.pdf

Navarro L. y Vilajosana G., (2019). *Arquitectura de aplicaciones web.* Universidad Abierta de Catalunya. PID_00184783. España.

Porras, J. (2015). *Sistema de control domótico inalámbrico, para personas adultas mayores en el Cantón Salcedo. s* [Tesis de grado, Universidad Técnica de Ambato] LA Referencia. https://repositorio.uta.edu.ec/jspui/handle/123456789/13065

Singh, N. Rothe, P. y Rathkanthiwar, A. (2017). Implementation of safety alert system for elderly people using multi-sensors. *International conference of Electronics,*

Communication and Aerospace Technology (ICECA), pp. 282-286, URL: https://ieeexplore.ieee.org/document/8203688

Ramos, J., Vargas, J., Gorrostieta, H. (2018) Robotica y Mecatronica. *Asociación Mexicana de Mecatrónica A.C.* El Cerrito, C.P. 76160, Querétaro, México. 1ra edición. ISBN 978-607-9394-14-1

López, M. M. R. (2020). *Diseño e implementación de una red de sensores auto-configurable utilizando transceptores NRF24L01*. ISPJAE, La Habana, CUBA. URL: https://www.researchgate.net/profile/Mario-Rivera-Lopez/publication/345686677_Tesis_Mario_Manuel_Rivera_Lopez/links/5faac1c9a6fdcc331b92cf1e/Tesis-Mario-Manuel-Rivera-Lopez.pdf

Martínez, V.,(2014). *Introducción a la plataforma Arduino y al Sensor ultrasónico HC-SR04.* Trabajo de fin de grado. Universidad Carlos III de Madrid. Madrid. España. URL: https://e-archivo.uc3m.es/bitstream/handle/10016/22916/TFG_Virginia_Martinez_Fuentes.pdf

Matich, J. (2001). *Redes Neuronales:Conceptos Básicos y Aplicaciones*. Obtenido de https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/mono graias/matich-redesneuronales.pdf

Salas, R. (2004). *Redes neuronales artificiales*. Universidad de Valparaıso. Departamento de Computación, 1(1), 1-7. BELL, URL: https://www.academia.edu/download/50358783/Redes_Neuronales_Artificiales.pdf

Serrano, J. (2016). *Red Inalámbrica de Sensores Para la Optimización del Riego en Plantaciones Agrónomas,* Universidad Politécnica de Madrid*,* URL: https://oa.upm.es/49636

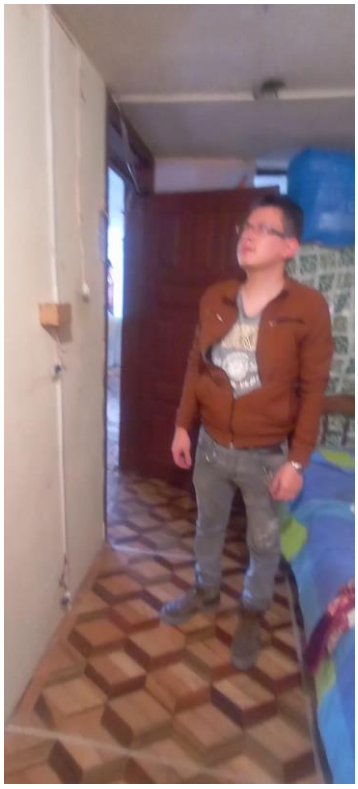
Texas Instruments. (s.f). *Ek-tm4c1294xl*. Recuperado el 27 de julio de 2023, de Www.ti.com URL: https://www.ti.com/tool/EK-TM4C1294XL

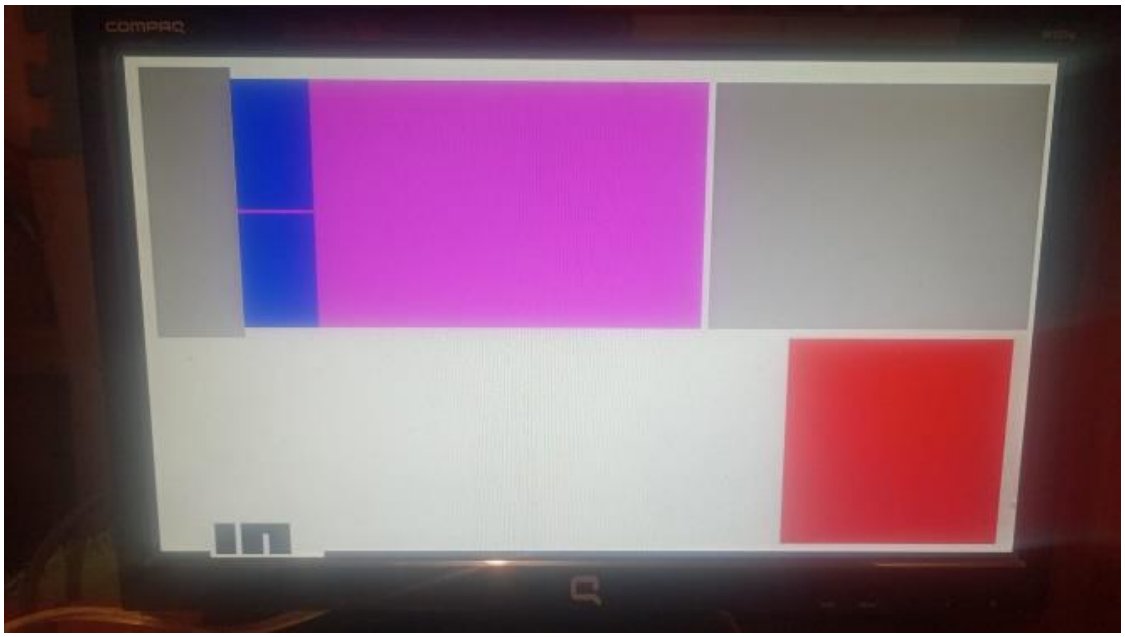
Thompson, C.(2009). *Como utilizar un servo motor con Arduino.* MCI Electronics. p.p. 6. URL: https://www.academia.edu/download/31743512/Guia_MCI_-_Servo_motor_con_Arduino.pdf

Villegas Redroban, D. X., & Dager Pacheco, J. E. (2013). Analizador de protocolo SPI (Bachelor's thesis, Espol). URL: https://www.dspace.espol.edu.ec/handle/123456789/42249

ANEXOS

Anexo 1: Fotos Experimentales





Anexo 2: Código nodo Arduino

```
//UNIVERSIDAD POLITECNICA SALESIANA
//ELECTRONICA Y AUTOMATIZACION
//PROYECTO DE TITULACION
//CHÁVEZ ANDRÉS - POSSO LUIS
#include <Servo.h>
 Servo servoUno;                  // crea el objeto para controlar un servo
 int posicion = 0;

#include <SPI.h>
#include <Enrf24.h>
#include <string.h>
#include "Adafruit_VL53L0X.h"
// Direcciones de los sensores laser LOX
   #define LOX1_ADDRESS 0x30
   #define LOX2_ADDRESS 0x31
// Pines digitales de los sensores laser LOX
   #define SHT_LOX1 5
   #define SHT_LOX2 4
// Creacion de objetos para el uso de la biblioteca.
   Adafruit_VL53L0X lox1 = Adafruit_VL53L0X();
   Adafruit_VL53L0X lox2 = Adafruit_VL53L0X();
// Variablesque guardarán los datos medidos por los sensores laser
   VL53L0X_RangingMeasurementData_t measure1;
   VL53L0X_RangingMeasurementData_t measure2;
//se crea el objeto "radio" para usar la librería del módulo Enrf24
   Enrf24 radio(9, 10, 8);  // PIN 9=CE , PIN 10=CSN , PIN 8=IRQ
//Direccion del canal de radiofrecuencia
   const uint8_t txaddr[] = { 0xF0, 0xF0, 0xF0, 0xF0, 0xE1 };
//Constantes con formato caracter
   const char *str_on = "ON";
   const char *str_off = "OFF";
//Iniciar la comunicación por radiofrecuencia
   void dump_radio_status_to_serialport(uint8_t);
//Declaración de variables
   //Variables para los sensores LIDAR
   int d1 = 0;
   int d2 = 0;
   //Variables para el sensor ultrasónico
   int trig = 2;
   int echo = 3;
   int duracion;
   int pinservo = 6;
   int distancia;
//Inicializacion de los sensores LIDAR
   void setID() {
     digitalWrite(SHT_LOX1, LOW);
     digitalWrite(SHT_LOX2, LOW);
     delay(10);
     digitalWrite(SHT_LOX1, HIGH);
     digitalWrite(SHT_LOX2, HIGH);
     delay(10);
     digitalWrite(SHT_LOX1, HIGH);
```

```
    digitalWrite(SHT_LOX2, LOW);
    //Verificación del estado de inicializacion del LIDAR 1
    if (!lox1.begin(LOX1_ADDRESS)) {
      Serial.println(F("FailedVLX"));
      while (1);}
    delay(10);
    digitalWrite(SHT_LOX2, HIGH);
    delay(10);
    //Verificación del estado de inicializacion del LIDAR 1
    if (!lox2.begin(LOX2_ADDRESS)) {
      Serial.println(F("FailedVLX2"));
      while (1);}
    Serial.println("LOXOK");
  }
// Funcion para la lectura de la medición con los sensores LIDAR
  void read_dual_sensors() {
  lox1.rangingTest(&measure1, false);
  lox2.rangingTest(&measure2, false);
  if (measure1.RangeStatus != 4) {
    if (measure1.RangeMilliMeter < 850){
      d1 = 1;
    }else {
      d1 = 0;}
  Serial.print(measure1.RangeMilliMeter);
  } else {
      d1 = 0;
      Serial.print(F("Out R"));}
  Serial.print(F(" : "));
  if (measure2.RangeStatus != 4) {
    if (measure2.RangeMilliMeter < 850){  // si es menor de 50cm
      d2 = 1;
    } else {
      d2 = 0;}
  Serial.print(measure2.RangeMilliMeter);
  } else {
    d2 = 0;
    Serial.print(F("Out R"));}
    Serial.println();
}

void setup() {
  servoUno.attach(pinservo);
  servoUno.write(90);// conecte la variable servoUno al pin 6
  Serial.begin(9600);          //Inicializar comunicacion serial a 9600 baudios
  pinMode(trig, OUTPUT);       //Pin Trigger del sensor ultrasónico como salida
  pinMode(echo, INPUT);        //Pin Echo del sensor ultrasónico como entrada
  pinMode(SHT_LOX1, OUTPUT);   //Pin digital del sensor lidar 1 como salida
  pinMode(SHT_LOX2, OUTPUT);   //Pin digital del sensor lidar 2 como salida
  digitalWrite(SHT_LOX1, LOW);  //Inicializar el sensor lidar 1 en nivel bajo
  digitalWrite(SHT_LOX2, LOW);  //Inicializar el sensor lidar 2 en nivel bajo
  setID();                     //Funcion para inicializar la comunicacion con los sensores LIDAR
  SPI.begin();                 //Inicializar comunicación SPI
  SPI.setDataMode(SPI_MODE0);
  SPI.setBitOrder(MSBFIRST);
```

```
  radio.begin();                  //Configuracion de la comunicacion por radiofrecuencia con valores
default 1Mbps,, max TX power
  dump_radio_status_to_serialport(radio.radioState());
  radio.setTXaddress((void *)txaddr);
}

void loop() {
  char str[8]; //Variable que se enviara por radiogrecuencia
  //Medicion con el sensor ultrasonico
  digitalWrite(trig, HIGH);
  delay(1);
  digitalWrite(trig, LOW);
  duracion = pulseIn(echo, HIGH);
  distancia = duracion/58.2;

    read_dual_sensors();
  //concatenar el caracter "B" con el valor "d1" en la variable "str"
  sprintf(str, "B%d", d1);
  //se envía el string "str" al MAESTRO
  radio.print(str);
  //se envía el caracter ","
  radio.print(",");
  //Se concatena el valor "d2" en la variable str
  sprintf(str, "%d", d2);
  //se envía el string "str" al MAESTRO
  radio.print(str);
  //se envía el caracter ","
  radio.print(",");
  // Condicion para la deteccion vertical con el sensor ultrasonico
  if (distancia >= 90) { //Persona ausente
    sprintf(str, "0");   //Concatenar el valor del sensor ultrasonico a la variable "str"
    servoUno.write(90);
  } else {              //Concatenar el valor del sensor ultrasonico a la variable "str"
    sprintf(str, "1");   //Persona presente
    servoUno.write(5);
  }
  //se envía el string "str" al MAESTRO
  radio.print(str);

  radio.flush();  // Force transmit (don't wait for any more data)
 Serial.println(distancia);
 // Serial.println(str);

}

void dump_radio_status_to_serialport(uint8_t status) {
   Serial.print(status);
}
```

Anexo 3: Código nodo Master TIVA

```
//UNIVERSIDAD POLITECNICA SALESIANA
//ELECTRONICA Y AUTOMATIZACION
//PROYECTO DE TITULACION
//CHÁVEZ ANDRÉS - POSSO LUIS
#include <SPI.h>
#include <Ethernet.h>
#include <Enrf24.h>
#include <nRF24L01.h>
#include <string.h>
#include <math.h>
byte p1, p2,p3;
float n1, n2, n3, n4, n5;
float a1, a2 ,a3 ,a4;
float e=2.7182818;
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,100,70);
IPAddress gateway(192,168,100,1);
IPAddress subnet(255,255,255,0);
// telnet defaults to port 23
EthernetServer server(80);
boolean alreadyConnected = false; // whether or not the client was connected previously
 //PE_0 = CE, PE_1 = CSN, PE_2 = IRQ
// SPI pins : SCK = PB_5 , MOSI = Pe4, MISO = Pd3
Enrf24 radio(PE_0, PE_1, PE_2);
const uint8_t rxaddr[] = { 0xF0,0xF0,0xF0,0xF0,0xE1 };
  int d0;  int d1;  int d2;  int d3; int d4;
  int d5;  int d6;  int d7;  int d8; int st1=0;
  int n51; int distancia; int st2=0; int st3=0;
const char *str_on = "ON";
const char *str_off = "OFF";
unsigned long prev_time;
void setup() {
  Serial.begin(9600);
  Serial.flush();
  SPI.setModule(2);
  SPI.begin();
  SPI.setDataMode(SPI_MODE0);
  SPI.setBitOrder(1);
  pinMode(48,OUTPUT); //48
  pinMode(49,OUTPUT); //49
  pinMode(50,OUTPUT);  //50
  radio.begin();  // Defaults 1Mbps, channel 0, max TX power
  radio.setRXaddress((void*)rxaddr);
  radio.enableRX();  // Start listening
  delay(100);
Ethernet.begin(mac, ip, gateway, subnet);
```

```
  // start listening for clients
  server.begin();
    //Serial.print("Chat server address:");
  //Serial.println(Ethernet.localIP());
  serverip();
}
void loop() {
    leerrf();
    if((d0==1 or d1==1 or d2==1)and st1==1){redneuronal1();}
    if((n5>=3.00 and n5<=4.75) and (d0==1 or d1==1 or
d2==1)){digitalWrite(47,HIGH);}else{st1=0;digitalWrite(47,LOW);}
    if((n5>=6.76 and n5<=8.75) and (d0==1 or d1==1 or
d2==1)){digitalWrite(48,HIGH);}else{st1=0;digitalWrite(48,LOW);}
if((d3==1 or d4==1 or d5==1)and st2==2){redneuronal2();}//Serial.println("red 2");}
    if((n5>=3.00 and n5<=4.75) and (d3==1 or d4==1 or
d5==1)){digitalWrite(49,HIGH);}else{st1=0;digitalWrite(49,LOW);}
    if((n5>=6.76 and n5<=8.75) and (d3==1 or d4==1 or
d5==1)){digitalWrite(50,HIGH);}else{st1=0;digitalWrite(50,LOW);}
 if((d6==1 or d7==1 or d8==1) and st3==3){redneuronal3();}//Serial.println("red 3");}
    if((n5>=3.00 and n5<=4.75) and (d6==1 or d7==1 or
d8==1)){digitalWrite(67,HIGH);}else{st1=0;digitalWrite(67,LOW);}
    if((n5>=6.76 and n5<=8.75) and (d6==1 or d7==1 or
d8==1)){digitalWrite(68,HIGH);}else{st1=0;digitalWrite(68,LOW);}
if(d0==0 and d1==0 and d2==0 and d3==0 and d4==0 and d5==0 and d6==0 and
d7==0  and d8==0){
      st1=0;st2=0;st3=0;n5=0;Serial.println("No detecta ningun sensor, NO ENTRO A
NINGUNA RED");Serial.println(n5);
digitalWrite(47,LOW);digitalWrite(48,LOW);digitalWrite(49,LOW);digitalWrite(50,L
OW);digitalWrite(63,LOW);digitalWrite(64,LOW);
digitalWrite(65,LOW);digitalWrite(66,LOW);digitalWrite(43,LOW);digitalWrite(44,L
OW);digitalWrite(67,LOW);digitalWrite(68,LOW);
      }
    serverip();
}
```

```
void redneuronal1(){
 p1=d0;
 p2=d1;
 p3=d2;
 char buffer[40];
  sprintf(buffer, "Red 1:  p1: %d | p2: %d | p3: %d ", p1,p2,p3);
  Serial.println(buffer);
//----CAPA OCULTA CON 4 NEURONAS----------------------
//n1=P1*W11+P2*W21+P3*W31+BIAS1
  n1= p1*(-3.2511)+p2*(-1.4598)+p3*(2.8150)+(3.1233);
//n2=P1*W12+P2*W22+P3*W32+BIAS2
  n2= p1*(-4.9281)+p2*(-0.2071)+p3*(0.0162)+(2.0981);
//n3=P1*W13+P2*W23+P3*W33+BIAS3
  n3= p1*(2.4710)+p2*(-1.1276)+p3*(3.5718)+(-0.0108);
//n4=P1*W14+P2*W24+P3*W34+BIAS4
  n4= p1*(-0.1218)+p2*(-4.3438)+p3*(1.9248)+(2.0523);

//-------FUNCION DE ACTIVACION TANSIG ----------
  a1= tansig(n1);
  a2= tansig(n2);
  a3= tansig(n3);
  a4= tansig(n4);

//------CAPA DE SALIDA - FUNCION DE ACTIVACION LINEAL-------

//n5= a1*W31+a2*W32+a3*W33+a4*W34+BIAS5
  n5=a1*(0.9346)+a2*(-2.0399)+a3*(1.0354)+a4*(-1.4699)+(3.4818);
   Serial.print("red 1 n5: ");
   Serial.print(n5);
        Serial.print(" |  LIDAR A1: ");
      Serial.print(d0);
      Serial.print(" |  LIDAR A2: ");
      Serial.print(d1);
      Serial.print(" | ULTRASONICO A: ");
```

```
      Serial.println(d2);


  if(n5<=2.99){Serial.print(n51);Serial.println(" Red 1: Revisar sensores, error en
lectura");}
  if(n5>=3.00 and n5<=3.75){Serial.print(n51);Serial.println(" Red 1: Alerta, persona
caida en zona no muerta");}
  if(n5>=3.76 and n5<=4.75){Serial.print(n51);Serial.println(" Red 1: Alerta, persona
caida en zona muerta ");}
  if(n5>=4.76 and n5<=5.75){Serial.print(n51);Serial.println(" Red 1: Revisar sensor
LiDAR 1 , Error en la medicion de zona no muerta");}
  if(n5>=5.76 and n5<=6.75){Serial.print(n51);Serial.println(" Red 1: Revisar sensor
LIDAR 1 , Error en la medicion de zona muerta");}
  if(n5>=6.76 and n5<=7.75){Serial.print(n51);Serial.println(" Red 1: Persona parada en
zona no muerta");}
  if(n5>=7.76 and n5<=8.75){Serial.print(n51);Serial.println(" Red 1: Persona parada en
zona muerta");}
}
void redneuronal2(){
 p1=d3;
 p2=d4;
 p3=d5;
 char buffer[40];
  sprintf(buffer, " Red 2:  p1: %d | p2: %d | p3: %d ", p1,p2,p3);
  Serial.println(buffer);
//----CAPA OCULTA CON 4 NEURONAS----------------------
//n1=P1*W11+P2*W21+P3*W31+BIAS1
  n1= p1*(-3.2511)+p2*(-1.4598)+p3*(2.8150)+(3.1233);
//n2=P1*W12+P2*W22+P3*W32+BIAS2
  n2= p1*(-4.9281)+p2*(-0.2071)+p3*(0.0162)+(2.0981);
//n3=P1*W13+P2*W23+P3*W33+BIAS3
  n3= p1*(2.4710)+p2*(-1.1276)+p3*(3.5718)+(-0.0108);
//n4=P1*W14+P2*W24+P3*W34+BIAS4
  n4= p1*(-0.1218)+p2*(-4.3438)+p3*(1.9248)+(2.0523);

//-------FUNCION DE ACTIVACION TANSIG ----------
  a1= tansig(n1);
```

```
  a2= tansig(n2);
  a3= tansig(n3);
  a4= tansig(n4);

//------CAPA DE SALIDA - FUNCION DE ACTIVACION LINEAL-------

//n5= a1*W31+a2*W32+a3*W33+a4*W34+BIAS5
  n5=a1*(0.9346)+a2*(-2.0399)+a3*(1.0354)+a4*(-1.4699)+(3.4818);
   Serial.print("red 2 n5: ");
   Serial.print(n5);
         Serial.print(" LIDAR B1: ");
       Serial.print(d3);
     Serial.print(" |  LIDAR B2: ");
      Serial.print(d4);
      Serial.print(" | ULTRASONICO B: ");
       Serial.println(d5);

  if(n5<=2.99){Serial.print(n51);Serial.println(" Red 2: Revisar sensores, error en lectura");}
  if(n5>=3.00 and n5<=3.75){Serial.print(n51);Serial.println(" Red 2: Alerta, persona caida en zona no muerta");}
  if(n5>=3.76 and n5<=4.75){Serial.print(n51);Serial.println(" Red 2: Alerta, persona caida en zona muerta ");}
  if(n5>=4.76 and n5<=5.75){Serial.print(n51);Serial.println(" Red 2: Revisar sensor LiDAR 1 , Error en la medicion de zona no muerta");}
  if(n5>=5.76 and n5<=6.75){Serial.print(n51);Serial.println(" Red 2: Revisar sensor LIDAR 1 , Error en la medicion de zona muerta");}
  if(n5>=6.76 and n5<=7.75){Serial.print(n51);Serial.println(" Red 2: Persona parada en zona no muerta");}
  if(n5>=7.76 and n5<=8.75){Serial.print(n51);Serial.println(" Red 2: Persona parada en zona muerta");}
}

void redneuronal3(){
 p1=d6;
 p2=d7;
 p3=d8;
          char buffer[40];
  sprintf(buffer, " Red 3:  p1: %d | p2: %d | p3: %d ", p1,p2,p3);
  Serial.println(buffer);
//----CAPA OCULTA CON 4 NEURONAS----------------------
//n1=P1*W11+P2*W21+P3*W31+BIAS1
  n1= p1*(-3.2511)+p2*(-1.4598)+p3*(2.8150)+(3.1233);
//n2=P1*W12+P2*W22+P3*W32+BIAS2
  n2= p1*(-4.9281)+p2*(-0.2071)+p3*(0.0162)+(2.0981);
//n3=P1*W13+P2*W23+P3*W33+BIAS3
  n3= p1*(2.4710)+p2*(-1.1276)+p3*(3.5718)+(-0.0108);
//n4=P1*W14+P2*W24+P3*W34+BIAS4
  n4= p1*(-0.1218)+p2*(-4.3438)+p3*(1.9248)+(2.0523);

//-------FUNCION DE ACTIVACION TANSIG ----------
  a1= tansig(n1);
  a2= tansig(n2);
  a3= tansig(n3);
  a4= tansig(n4);
```

```
//------CAPA DE SALIDA - FUNCION DE ACTIVACION LINEAL-------

//n5= a1*W31+a2*W32+a3*W33+a4*W34+BIAS5
  n5=a1*(0.9346)+a2*(-2.0399)+a3*(1.0354)+a4*(-1.4699)+(3.4818);
   Serial.print("red 3 n5: ");
   Serial.print(n5);
       Serial.print(" LIDAR C1: ");
       Serial.print(d6);
       Serial.print(" | LIDAR C2: ");
       Serial.print(d7);
       Serial.print(" | ULTRASONICO C: ");
       Serial.println(d8);

  if(n5<=2.99){Serial.print(n51);Serial.println(" Red 3: Revisar sensores, error en lectura");}
  if(n5>=3.00 and n5<=3.75){Serial.print(n51);Serial.println(" Red 3: Alerta, persona caida en zona no muerta");}
  if(n5>=3.76 and n5<=4.75){Serial.print(n51);Serial.println(" Red 3: Alerta, persona caida en zona muerta ");}
  if(n5>=4.76 and n5<=5.75){Serial.print(n51);Serial.println(" Red 3: Revisar sensor LiDAR 1 , Error en la medicion de zona no muerta");}
  if(n5>=5.76 and n5<=6.75){Serial.print(n51);Serial.println(" Red 3: Revisar sensor LIDAR 1 , Error en la medicion de zona muerta");}
  if(n5>=6.76 and n5<=7.75){Serial.print(n51);Serial.println(" Red 3: Persona parada en zona no muerta");}
  if(n5>=7.76 and n5<=8.75){Serial.print(n51);Serial.println(" Red 3: Persona parada en zona muerta");}

}

float tansig(float x){
   float a;
  a=(pow(e,x)-pow(e,-x))/(pow(e,x)+pow(e,-x));
 return a;
}
void leerrf(){
  //Serial.println("---- EMPIEZA 'LEER' -------- ");
  char inbuf[33];

    delta_set();
    while (!radio.available(true)&& delta_get()<1000);
     if (radio.read(inbuf)) {
       //Serial.println(inbuf);
     if(inbuf[0]=='A'){
      d0=inbuf[1]-48;//'1'=49
      d1=inbuf[3]-48;
      d2=inbuf[5]-48;
      st1=1;
//       Serial.print (st);
//       Serial.print ("  ");
//       Serial.print(inbuf[0]);
//       Serial.print(" | LIDAR A1: ");
//       Serial.print(d0);
//       Serial.print(" | LIDAR A2: ");
//       Serial.print(d1);
       Serial.print(" | ULTRASONICO A: ");
```

```
      Serial.println(d2);
//      Serial.println(st);//llego dato de nodo 1
          }
     if(inbuf[0]=='B'){
     d3=inbuf[1]-48;
     d4=inbuf[3]-48;
     d5=inbuf[5]-48;
     st2=2;
//     Serial.print (st);
//      Serial.print ("  ");
//      Serial.print(inbuf[0]);
//      Serial.print(" |  LIDAR B1: ");
//      Serial.print(d3);
//    Serial.print(" |  LIDAR B2: ");
//     Serial.print(d4);
     Serial.print(" | ULTRASONICO B: ");
      Serial.println(d5);
      //////entonces ya llego dato nodo 2
//     Serial.println(st);
       //}
     }
     if(inbuf[0]=='C'){
     d6=inbuf[1]-48;
     d7=inbuf[3]-48;
     d8=inbuf[5]-48;
     st3=3;
      Serial.print(" | ULTRASONICO C: ");
      Serial.println(d8);
      //////entonces ya llego dato nodo 3
//      Serial.println(st);
     }
  }
}

void delta_set() {
  prev_time = millis();
}

// TimeOut
unsigned long delta_get() {
  unsigned long time;
  unsigned long delta;

  time = millis();
  if (time < prev_time) { // TimeOut
    delta = 0xffffffff - prev_time + time + 1;
  }
  else {
    delta = time - prev_time;
  }
  return delta;
}
void dump_radio_status_to_serialport(uint8_t status)
{
  Serial.print("Enrf24 radio transceiver status: ");
```

```
  switch (status) {
    case ENRF24_STATE_NOTPRESENT:
      Serial.println("NO TRANSCEIVER PRESENT");
      break;

    case ENRF24_STATE_DEEPSLEEP:
      Serial.println("DEEP SLEEP <1uA power consumption");
      break;

    case ENRF24_STATE_IDLE:
      Serial.println("IDLE module powered up w/ oscillators running");
      break;

    case ENRF24_STATE_PTX:
      Serial.println("Actively Transmitting");
      break;

    case ENRF24_STATE_PRX:
      Serial.println("Receive Mode");
      break;

    default:
      Serial.println("UNKNOWN STATUS CODE");
  }
}

void serverip(){
  //Serial.println("SERVER IP");
  //Serial.print("N5:");
  //Serial.println(n5);



 // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    //Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be closed after completion of the
response
          client.println("Refresh: 5");  // refresh the page automatically every 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
```

```
        client.println("<html>");

         client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\">");
         client.println("<link rel=\"icon\" href=\"data:,\">");
         // CSS to style the on/off buttons
         // Feel free to change the background-color and font-size attributes to fit your preferences
         client.println("<style>html { font-family: Arial; display: inline-block; margin: 10px auto;
text-align: center}");
         client.println("</style></head>");

        client.print("<title>CHÁVEZ - POSSO </title>");
        client.print("<body bgcolor=\"black\">");
        client.print("<h1><font color=\"blue\">UNIVERSIDAD POLITECNICA
SALESIANA</font></h1>");
        client.print("<h1><font color=\"blue\">INGENIERIA ELECTRONICA Y
AUTOMATIZACION</font></h1>");
        client.print("<h1><font color=\"blue\">PROYECTO DE TITULACION</font></h1>");
        client.print("<h1><font color=\"blue\">INTEGRANTES: CHÁVEZ ANDRÉS - POSSO
LUIS</font></h1>");
                //client.print("</body>");
                 client.print("<font color=\"yellow\">NODO1:  ");
         client.print(d0);
         client.print(" / ");
         client.print(d1);
         client.print(" / ");
         client.print(d2);
         client.println("<br />");

         client.print("  NODO2:  ");
         client.print(d3);
         client.print(" / ");
         client.print(d4);
         client.print(" / ");
         client.print(d5);
         client.println("<br />");

         client.print("  NODO3:  ");
         client.print(d6);
         client.print(" / ");
         client.print(d7);
         client.print(" / ");
         client.print(d8);
         client.println("</font><br />");

         client.println("<br />");
         client.println("<br />");

         if(n5==0){
         client.println("<font  SIZE='8'> </font>");//-----
         client.println("<br />");
         }
         if(n5>=3.00 and n5<=3.75){
         client.println("<font  SIZE='8' color=\"red\">Alerta, persona caida</font>");//-----
         client.println("<br />");
```

```
}else{
  client.println("<font color=\"lime\">  </font>");
  client.println("<br />");
  }
if(n5>=3.76 and n5<=4.75){
client.println("<font  SIZE='8' color=\"red\">Alerta, persona caida.</font>");
client.println("<br />");
}else{
  client.println("<font color=\"lime\">  </font>");
  client.println("<br />");
  }
if(n5>=4.76 and n5<=5.75){
client.println("<font  SIZE='8' color=\"red\">Revisar sensor LIDAR 1 , Error en la medicion</font>");
client.println("<br />");
}else{
  client.println("<font color=\"lime\">  </font>");
  client.println("<br />");
  }
if(n5>=5.76 and n5<=6.75){
client.println("<font  SIZE='8' color=\"red\">Revisar sensor LIDAR 1 , Error en la medicion.</font>");
client.println("<br />");
}else{
  client.println("<font color=\"lime\">  </font>");
  client.println("<br />");
  }
if(n5>=6.76 and n5<=7.75){
client.println("<font  SIZE='8' color=\"red\">Persona parada</font>");
client.println("<br />");
}else{
  client.println("<font color=\"lime\">  </font>");
  client.println("<br />");
  }
if(n5>=7.76 and n5<=8.75){
client.println("<font  SIZE='8' color=\"red\">Persona parada </font>");
client.println("<br />");
}else{
  client.println("<font color=\"lime\">  </font>");
  client.println("<br />");
  }


if(d6==1 or d7==1 or d8==1){
  client.println("<h2><font color=\"red\">ZONA C </font></h2>");
  client.println("<br />");
}else{
  client.println("<h2><font color=\"red\">  </font></h2>");
  client.println("<br />");
  }
if(d3==1 or d4==1 or d5==1){
  client.println("<h2><font color=\"red\">ZONA B </font></h2>");
  client.println("<br />");
}else{
  client.println("<h2><font color=\"red\">  </font></h2>");
```

```
            client.println("<br />");
            }
          if(d0==1 or d1==1 or d2==1){
            client.println("<h2><font SIZE='8' color=\"red\">ZONA A </font></h2>");
            client.println("<br />");
          }else{
            client.println("<h2><font color=\"lime\">  </font></h2>");
            client.println("<br />");}
          if(d0==0 and d1==0 and d2==0 and d3==0 and d4==0 and d5==0 and d6==0 and d7==0
and d8==0){
            client.println("<h2><font    SIZE='8'  color=\"red\">PERSONA  FUERA  DE  LAS  3
AREAS </font></h2>");
            client.println("<br />");
          }else{
            client.println("<h2><font color=\"lime\">  </font></h2>");
            client.println("<br />");
            }
        // output the value of each analog input pin

        client.println("</html>");
        break;
      }
      if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
      } else if (c != '\r') {
        // you've gotten a character on the current line
        currentLineIsBlank = false;
      }
    }
  }
  // give the web browser time to receive the data
  delay(50);
  // close the connection:
  client.stop();
  Serial.println("client disconnected");
 }
}
```

Anexo 4: Código Matlab

```
clear all; clc;
p= [0 0 0 0 1 1 1 1;0 0 1 1 0 0 1 1;0 1 0 1 0 1 0 1]
t=[1 2 3 4 5 6 7 8]
net2=newff(minmax(p),[2 1],{'tansig','purelin'},'traincgf');
net3=newff(minmax(p),[3 1],{'tansig','purelin'},'traincgf');
net4=newff(minmax(p),[4 1],{'tansig','purelin'},'traincgf');
net5=newff(minmax(p),[5 1],{'tansig','purelin'},'traincgf');
net6=newff(minmax(p),[6 1],{'tansig','purelin'},'traincgf');
net7=newff(minmax(p),[7 1],{'tansig','purelin'},'traincgf');
net2=init(net2);
net3=init(net3);
net4=init(net4);
net5=init(net5);
net6=init(net6);
net7=init(net7);
net2.trainparam.goal=1e-9;
net3.trainparam.goal=1e-9;
net4.trainparam.goal=1e-9;
net5.trainparam.goal=1e-9;
net6.trainparam.goal=1e-9;
net7.trainparam.goal=1e-9;
net2.trainParam.epochs=1000;
net3.trainParam.epochs=1000;
net4.trainParam.epochs=1000;
net5.trainParam.epochs=1000;
net6.trainParam.epochs=1000;
net7.trainParam.epochs=1000;
net2=train(net2,p,t);
net3=train(net3,p,t);
net4=train(net4,p,t);
net5=train(net5,p,t);
net6=train(net6,p,t);
net7=train(net7,p,t);
a2=sim(net2,p)
a3=sim(net3,p)
a4=sim(net4,p)
a5=sim(net5,p)
a6=sim(net6,p)
a7=sim(net7,p)
pesos4_1=net4.iw{1,1}
bias4_1=net4.b{1,1}
pesos4_2=net4.lw{2,1}
bias4_2=net4.b{2,1}
```

Anexo 5: Código Verilog

Módulo Generador de pixel

```
//UNIVERSIDAD POLITECNICA SALESIANA
//ELECTRONICA Y AUTOMATIZACION
//PROYECTO DE TITULACION
//CHÁVEZ ANDRÉS - POSSO LUIS

`timescale 1ns / 1ps

module pixel_gen(
    input clk,
    input reset,
    input video_on,
    input [9:0] x, y,
    //input [4:0] enc,
    input [2:0]sw,
    //input sw,
    output reg [11:0] rgb
    );

    // RGB Color Values
    parameter WHITE   = 12'hFFF;    // sw ON
    parameter BLACK   = 12'h000;    // sw OFF
    parameter RED1    = 12'hF00;    // 0
    parameter GREEN1  = 12'h0F0;    // 1
    parameter BLUE1   = 12'h00F;    // 2
    parameter YELLOW1 = 12'h0FF;    // 3
    parameter AQUA1   = 12'hFF0;    // 4
    parameter VIOLET1 = 12'hF0F;    // 5
    parameter GRAY1   = 12'hAAA;    // 6
    parameter RED2    = 12'hA00;    // 7
    parameter GREEN2  = 12'h0A0;    // 8
    parameter BLUE2   = 12'h00A;    // 9
    parameter YELLOW2 = 12'h0AA;    // 10
    parameter AQUA2   = 12'hAA0;    // 11
    parameter VIOLET2 = 12'hA0A;    // 12
    parameter RED3    = 12'h500;    // 13
    parameter GREEN3  = 12'h050;    // 14
    parameter BLUE3   = 12'h005;    // 15
    parameter YELLOW3 = 12'h055;    // 16
    parameter AQUA3   = 12'h550;    // 17
    parameter VIOLET3 = 12'h505;    // 18
    parameter GRAY2   = 12'h555;    // 19

    /*----------------------
    / rectangular area in center of screen
    parameter RECT_WIDTH  = 320;
    parameter RECT_HEIGHT = 240;
    parameter RECT_WIDTH  = 90;
    parameter RECT_HEIGHT = 411;
    wire [9:0] rect_x_l, rect_x_r;              // left and right boundary
    wire [9:0] rect_y_t, rect_y_b;              // top and bottom boundary
```

```
*/

//rectangle boundaries and position
//Rectangulo 1
wire [9:0] rect_x_l1, rect_x_r1;            // left and right boundary
wire [9:0] rect_y_t1, rect_y_b1;            // top and bottom boundary
//Rectangulo 2
wire [9:0] rect_x_l2, rect_x_r2;            // left and right boundary
wire [9:0] rect_y_t2, rect_y_b2;            // top and bottom boundary
//Rectangulo 3
wire [9:0] rect_x_l3, rect_x_r3;            // left and right boundary
wire [9:0] rect_y_t3, rect_y_b3;            // top and bottom boundary
//Rectangulo 4
wire [9:0] rect_x_l4, rect_x_r4;            // left and right boundary
wire [9:0] rect_y_t4, rect_y_b4;            // top and bottom boundary
//Rectangulo 5
wire [9:0] rect_x_l5, rect_x_r5;            // left and right boundary
wire [9:0] rect_y_t5, rect_y_b5;            // top and bottom boundary
//Rectangulo 6
wire [9:0] rect_x_l6, rect_x_r6;            // left and right boundary
wire [9:0] rect_y_t6, rect_y_b6;            // top and bottom boundary
//Rectangulo 7
wire [9:0] rect_x_l7, rect_x_r7;            // left and right boundary
wire [9:0] rect_y_t7, rect_y_b7;            // top and bottom boundary
//Rectangulo 8
wire [9:0] rect_x_l8, rect_x_r8;            // left and right boundary
wire [9:0] rect_y_t8, rect_y_b8;            // top and bottom boundary
//Rectangulo 9
wire [9:0] rect_x_l9, rect_x_r9;            // left and right boundary
wire [9:0] rect_y_t9, rect_y_b9;            // top and bottom boundary
//Rectangulo 11
wire [9:0] rect_x_l11, rect_x_r11;          // left and right boundary
wire [9:0] rect_y_t11, rect_y_b11;          // top and bottom boundary
wire [9:0] rect_x_l22, rect_x_r22;          // left and right boundary
wire [9:0] rect_y_t22, rect_y_b22;          // top and bottom boundary
wire [9:0] rect_x_l33, rect_x_r33;          // left and right boundary
wire [9:0] rect_y_t33, rect_y_b33;          // top and bottom boundary
wire [9:0] rect_x_l44, rect_x_r44;          // left and right boundary
wire [9:0] rect_y_t44, rect_y_b44;          // top and bottom boundary
wire [9:0] rect_x_l55, rect_x_r55;          // left and right boundary
wire [9:0] rect_y_t55, rect_y_b55;          // top and bottom boundary
wire [9:0] rect_x_l66, rect_x_r66;          // left and right boundary
wire [9:0] rect_y_t66, rect_y_b66;          // top and bottom boundary
wire [9:0] rect_x_l77, rect_x_r77;          // left and right boundary
wire [9:0] rect_y_t77, rect_y_b77;          // top and bottom boundary
wire [9:0] rect_x_l88, rect_x_r88;          // left and right boundary
wire [9:0] rect_y_t88, rect_y_b88;          // top and bottom boundary
wire [9:0] rect_x_l99, rect_x_r99;          // left and right boundary
wire [9:0] rect_y_t99, rect_y_b99;          // top and bottom boundary
// Posicion 1
assign rect_x_l1 = 125;                 // left boundary
assign rect_y_t1 = 260;                 // top boundary
assign rect_x_r1 = 284;   // right boundary
assign rect_y_b1 = 465 ;   // bottom boundary
```

```
// Posicion 2
assign rect_x_l2 = 289;                 // left boundary
assign rect_y_t2 = 260;                 // top boundary
assign rect_x_r2 = 456;   // right boundary
assign rect_y_b2 = 465 ;   // bottom boundary

// Posicion 3
assign rect_x_l3 = 461;                 // left boundary
assign rect_y_t3 = 260;                 // top boundary
assign rect_x_r3 = 620;   // right boundary
assign rect_y_b3 = 465 ;   // bottom boundary

// Cabecera cama
assign rect_x_l4 = 15;                  // left boundary
assign rect_y_t4 = 15;                  // top boundary
assign rect_x_r4 = 75;   // right boundary
assign rect_y_b4 =260 ;   // bottom boundary

// Mueble
assign rect_x_l5 = 403;                 // left boundary
assign rect_y_t5 = 25;                  // top boundary
assign rect_x_r5 = 629;   // right boundary
assign rect_y_b5 = 250 ;   // bottom boundary

// Almohada 1
assign rect_x_l6 = 76;                  // left boundary
assign rect_y_t6 = 25;                  // top boundary
assign rect_x_r6 = 127;   // right boundary
assign rect_y_b6 =140 ;   // bottom boundary

// Almohada 2
assign rect_x_l7 = 76;                  // left boundary
assign rect_y_t7 = 144;                 // top boundary
assign rect_x_r7 = 127;   // right boundary
assign rect_y_b7 = 250 ;   // bottom boundary

// Almohada 3
assign rect_x_l8 = 76;                  // left boundary
assign rect_y_t8 = 141;                 // top boundary
assign rect_x_r8 = 127;   // right boundary
assign rect_y_b8 =143 ;   // bottom boundary

// Edredon
assign rect_x_l9 = 77;                  // left boundary
assign rect_y_t9 = 25;                  // top boundary
assign rect_x_r9 = 397;   // right boundary
assign rect_y_b9 =250 ;   // bottom boundary

// Letra "I"
assign rect_x_l11 = 46;                 // left boundary
assign rect_y_t11 = 445;                // top boundary
assign rect_x_r11 = 60;   // right boundary
assign rect_y_b11 =476 ;   // bottom boundary

// lETRA "N" SUPERIOR
```

```
assign rect_x_l22 = 65;                 // left boundary
assign rect_y_t22 = 445;                 // top boundary
assign rect_x_r22 = 100;   // right boundary
assign rect_y_b22 =455 ;   // bottom boundary

// lETRA "N" VERTICAL IZQUIERDA
assign rect_x_l33 = 65;                 // left boundary
assign rect_y_t33 = 456;                 // top boundary
assign rect_x_r33 = 80;   // right boundary
assign rect_y_b33 =476 ;   // bottom boundary

// lETRA "N" VERTICAL DERECHA
assign rect_x_l44 = 85;                 // left boundary
assign rect_y_t44 = 456;                 // top boundary
assign rect_x_r44 = 100;   // right boundary
assign rect_y_b44 =476 ;   // bottom boundary

  // PARED IZQUIERDA
assign rect_x_l55 = 0;                 // left boundary
assign rect_y_t55 = 0;                 // top boundary
assign rect_x_r55 = 5;   // right boundary
assign rect_y_b55 =479 ;   // bottom boundary

  // PARED SUPERIOR
assign rect_x_l66 = 6;                 // left boundary
assign rect_y_t66 = 0;                 // top boundary
assign rect_x_r66 = 639;   // right boundary
assign rect_y_b66 =5 ;   // bottom boundary

   // PARED DERECHA
assign rect_x_l77 = 634;                 // left boundary
assign rect_y_t77 = 6;                 // top boundary
assign rect_x_r77 = 639;   // right boundary
assign rect_y_b77 =479 ;   // bottom boundary

   // PARED INFERIOR DERECHA
assign rect_x_l88 = 125;                 // left boundary
assign rect_y_t88 = 474;                 // top boundary
assign rect_x_r88 = 638;   // right boundary
assign rect_y_b88 =479 ;   // bottom boundary

// PARED INFERIOR IZQUIERDA
assign rect_x_l99 = 6;                 // left boundary
assign rect_y_t99 = 474;                 // top boundary
assign rect_x_r99 = 41;   // right boundary
assign rect_y_b99 =479 ;   // bottom boundary
// square status signal
wire rect_on1;
assign rect_on1 = (rect_x_l1 <= x) && (x <= rect_x_r1) &&
            (rect_y_t1 <= y) && (y <= rect_y_b1);
wire rect_on2;
assign rect_on2 = (rect_x_l2 <= x) && (x <= rect_x_r2) &&
            (rect_y_t2 <= y) && (y <= rect_y_b2);
wire rect_on3;
assign rect_on3 = (rect_x_l3 <= x) && (x <= rect_x_r3) &&
```

```
            (rect_y_t3 <= y) && (y <= rect_y_b3);
wire rect_on4;
assign rect_on4 = (rect_x_l4 <= x) && (x <= rect_x_r4) &&
            (rect_y_t4 <= y) && (y <= rect_y_b4);
wire rect_on5;
assign rect_on5 = (rect_x_l5 <= x) && (x <= rect_x_r5) &&
            (rect_y_t5 <= y) && (y <= rect_y_b5);
wire rect_on6;
assign rect_on6 = (rect_x_l6 <= x) && (x <= rect_x_r6) &&
            (rect_y_t6 <= y) && (y <= rect_y_b6);
wire rect_on7;
assign rect_on7 = (rect_x_l7 <= x) && (x <= rect_x_r7) &&
            (rect_y_t7 <= y) && (y <= rect_y_b7);
wire rect_on8;
assign rect_on8 = (rect_x_l8 <= x) && (x <= rect_x_r8) &&
            (rect_y_t8 <= y) && (y <= rect_y_b8);
wire rect_on9;
assign rect_on9 = (rect_x_l9 <= x) && (x <= rect_x_r9) &&
            (rect_y_t9 <= y) && (y <= rect_y_b9);
wire rect_on11;
assign rect_on11 = (rect_x_l11 <= x) && (x <= rect_x_r11) &&
            (rect_y_t11 <= y) && (y <= rect_y_b11);
wire rect_on22;
assign rect_on22 = (rect_x_l22 <= x) && (x <= rect_x_r22) &&
            (rect_y_t22 <= y) && (y <= rect_y_b22);
wire rect_on33;
assign rect_on33 = (rect_x_l33 <= x) && (x <= rect_x_r33) &&
            (rect_y_t33 <= y) && (y <= rect_y_b33);
wire rect_on44;
assign rect_on44 = (rect_x_l44 <= x) && (x <= rect_x_r44) &&
            (rect_y_t44 <= y) && (y <= rect_y_b44);
 wire rect_on55;
assign rect_on55 = (rect_x_l55 <= x) && (x <= rect_x_r55) &&
            (rect_y_t55 <= y) && (y <= rect_y_b55);
 wire rect_on66;
assign rect_on66 = (rect_x_l66 <= x) && (x <= rect_x_r66) &&
            (rect_y_t66 <= y) && (y <= rect_y_b66);
 wire rect_on77;
assign rect_on77 = (rect_x_l77 <= x) && (x <= rect_x_r77) &&
            (rect_y_t77 <= y) && (y <= rect_y_b77);
 wire rect_on88;
assign rect_on88 = (rect_x_l88 <= x) && (x <= rect_x_r88) &&
            (rect_y_t88 <= y) && (y <= rect_y_b88);
 wire rect_on99;
assign rect_on99 = (rect_x_l99 <= x) && (x <= rect_x_r99) &&
            (rect_y_t99 <= y) && (y <= rect_y_b99);
 always @*
 begin

   if(~video_on)
      rgb = BLACK;
   else
   if(sw[0] && rect_on1)
         rgb = RED1;
   else if(sw[1] && rect_on2)
```

```
            rgb = RED1;
        else if(sw[2] && rect_on3)
            rgb = RED1;
        else if(rect_on4)
            rgb = GRAY1;
        else if(rect_on5)
            rgb = GRAY1;
        else if(rect_on6)
            rgb = BLUE1;
        else if(rect_on7)
            rgb = BLUE1;
        else if(rect_on8)
            rgb = VIOLET1;
        else if(rect_on9)
            rgb = VIOLET1;
        else if(rect_on11)
            rgb = BLACK;
        else if(rect_on22)
            rgb = BLACK;
        else if(rect_on33)
            rgb = BLACK;
        else if(rect_on44)
            rgb = BLACK;
        else if(rect_on55)
            rgb = BLACK;
        else if(rect_on66)
            rgb = BLACK;
        else if(rect_on77)
            rgb = BLACK;
        else if(rect_on88)
            rgb = BLACK;
        else if(rect_on99)
            rgb = BLACK;
        else
            rgb = WHITE;
    end
endmodule
```

Módulo VGA controller

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Reference Book:
// Chu, Pong P.
// Wiley, 2008
// "FPGA Prototyping by Verilog Examples: Xilinx Spartan-3 Version"
//
//UNIVERSIDAD POLITECNICA SALESIANA
//ELECTRONICA Y AUTOMATIZACION
//PROYECTO DE TITULACION
//CHÁVEZ ANDRÉS - POSSO LUIS
// FOR USE WITH AN FPGA THAT HAS A 100MHz CLOCK SIGNAL ONLY.
// VGA Mode
// 640x480 pixels VGA screen with 25MHz pixel rate based on 60 Hz refresh rate
// 800 pixels/line * 525 lines/screen * 60 screens/second = ~25.2M pixels/second
```

```
//
// A 25MHz signal will suffice. The Basys 3 has a 100MHz signal available, so a
// 25MHz tick is created for syncing the pixel counts, pixel tick, horiz sync,
// vert sync, and video on signals.
//////////////////////////////////////////////////////////////////////////////////

module vga_controller(
   input clk_100MHz,   // from Basys 3
   input reset,        // system reset
   output video_on,    // ON while pixel counts for x and y and within display area
   output hsync,       // horizontal sync
   output vsync,       // vertical sync
   output p_tick,      // the 25MHz pixel/second rate signal, pixel tick
   output [9:0] x,     // pixel count/position of pixel x, max 0-799
   output [9:0] y      // pixel count/position of pixel y, max 0-524
   );

   // Based on VGA standards found at vesa.org for 640x480 resolution
   // Total horizontal width of screen = 800 pixels, partitioned  into sections
   parameter HD = 640;            // horizontal display area width in pixels
   parameter HF = 48;             // horizontal front porch width in pixels
   parameter HB = 16;             // horizontal back porch width in pixels
   parameter HR = 96;             // horizontal retrace width in pixels
   parameter HMAX = HD+HF+HB+HR-1; // max value of horizontal counter = 799
   // Total vertical length of screen = 525 pixels, partitioned into sections
   parameter VD = 480;            // vertical display area length in pixels
   parameter VF = 10;             // vertical front porch length in pixels
   parameter VB = 33;             // vertical back porch length in pixels
   parameter VR = 2;              // vertical retrace length in pixels
   parameter VMAX = VD+VF+VB+VR-1; // max value of vertical counter = 524

   //               ***               Generate               25MHz               from               100MHz
**********************************************************
         reg  [1:0] r_25MHz;
         wire w_25MHz;

         always @(posedge clk_100MHz or posedge reset)
                  if(reset)
                    r_25MHz <= 0;
                  else
                    r_25MHz <= r_25MHz + 1;

         assign w_25MHz = (r_25MHz == 0) ? 1 : 0; // assert tick 1/4 of the time
   //
**********************************************************************************
**********

   // Counter Registers, two each for buffering to avoid glitches
   reg [9:0] h_count_reg, h_count_next;
   reg [9:0] v_count_reg, v_count_next;

   // Output Buffers
   reg v_sync_reg, h_sync_reg;
   wire v_sync_next, h_sync_next;
```

```
    // Register Control
    always @(posedge clk_100MHz or posedge reset)
      if(reset) begin
        v_count_reg <= 0;
        h_count_reg <= 0;
        v_sync_reg  <= 1'b0;
        h_sync_reg  <= 1'b0;
      end
      else begin
        v_count_reg <= v_count_next;
        h_count_reg <= h_count_next;
        v_sync_reg  <= v_sync_next;
        h_sync_reg  <= h_sync_next;
      end

    //Logic for horizontal counter
    always @(posedge w_25MHz or posedge reset)      // pixel tick
      if(reset)
        h_count_next = 0;
      else
        if(h_count_reg == HMAX)                 // end of horizontal scan
          h_count_next = 0;
        else
          h_count_next = h_count_reg + 1;

    // Logic for vertical counter
    always @(posedge w_25MHz or posedge reset)
      if(reset)
        v_count_next = 0;
      else
        if(h_count_reg == HMAX)                 // end of horizontal scan
          if((v_count_reg == VMAX))             // end of vertical scan
            v_count_next = 0;
          else
            v_count_next = v_count_reg + 1;

    // h_sync_next asserted within the horizontal retrace area
    assign h_sync_next = (h_count_reg >= (HD+HB) && h_count_reg <= (HD+HB+HR-1));

    // v_sync_next asserted within the vertical retrace area
    assign v_sync_next = (v_count_reg >= (VD+VB) && v_count_reg <= (VD+VB+VR-1));

    // Video ON/OFF - only ON while pixel counts are within the display area
    assign video_on = (h_count_reg < HD) && (v_count_reg < VD); // 0-639 and 0-479
respectively

    // Outputs
    assign hsync  = h_sync_reg;
    assign vsync  = v_sync_reg;
    assign x      = h_count_reg;
    assign y      = v_count_reg;
    assign p_tick = w_25MHz;

endmodule
```

Archivo CONSTRAINT

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk_100MHz]

        set_property IOSTANDARD LVCMOS33 [get_ports clk_100MHz]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk_100MHz]

##Buttons
set_property PACKAGE_PIN U18        [get_ports reset]

set_property IOSTANDARD LVCMOS33 [get_ports reset]

##Pmod Header JB
##Sch name = JB1
set_property PACKAGE_PIN V17 [get_ports {JB[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JB[0]}]
##Sch name = JB2
set_property PACKAGE_PIN V16 [get_ports {JB[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JB[1]}]
##Sch name = JB3
set_property PACKAGE_PIN W16  [get_ports {JB[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]

##VGA Connector
set_property PACKAGE_PIN N19    [get_ports {rgb[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[11]}]
set_property PACKAGE_PIN J19    [get_ports {rgb[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[10]}]
set_property PACKAGE_PIN H19    [get_ports {rgb[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[9]}]
set_property PACKAGE_PIN G19    [get_ports {rgb[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[8]}]
set_property PACKAGE_PIN D17    [get_ports {rgb[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[7]}]
set_property PACKAGE_PIN G17    [get_ports {rgb[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[6]}]
set_property PACKAGE_PIN H17    [get_ports {rgb[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[5]}]
set_property PACKAGE_PIN J17    [get_ports {rgb[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[4]}]
set_property PACKAGE_PIN J18    [get_ports {rgb[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[3]}]
set_property PACKAGE_PIN K18    [get_ports {rgb[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN L18    [get_ports {rgb[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN N18    [get_ports {rgb[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN P19    [get_ports hsync]
set_property IOSTANDARD LVCMOS33 [get_ports hsync]
set_property PACKAGE_PIN R19    [get_ports vsync]
set_property IOSTANDARD LVCMOS33 [get_ports vsync]
```

Módulo TOP

```
//UNIVERSIDAD POLITECNICA SALESIANA
//ELECTRONICA Y AUTOMATIZACION
//PROYECTO DE TITULACION
//CHÁVEZ ANDRÉS - POSSO LUIS
//
//////////////////////////////////////////////////////////////////////////////////
module top(
   input clk_100MHz,       // 100MHz from Basys 3
   input reset,
   input [2:0] JB,         // PMOD JB
   /*output [1:0] LED,       // LED[1], LED[0]
   output [3:0] an,        // 7 Segment anodes
   output [6:0] seg,        // 7 Segment cathodes
   */
   output hsync,           // VGA port on Basys 3
   output vsync,           // VGA port on Basys 3
   output [11:0] rgb       // to DAC, 3 bits to VGA port on Basys 3
   );

         // Internal wires
         //wire [4:0] w_enc;
         wire w_l, w_r;
         wire w_vid_on, w_tick;
         wire [9:0] w_x, w_y;
         reg [11:0] rgb_reg;
         wire [11:0] rgb_next;

         /* Instantiate Modules
   debounce db(.clk(clk_100MHz), .Ain(JB[4]), .Bin(JB[5]), .Aout(w_A), .Bout(w_B));
         encoder enc (.clk(clk_100MHz), .A(w_A), .B(w_B), .BTN(JB[6]), .EncOut(w_enc),
.LED(LED));
         seg7_control seg7 (.clk(clk_100MHz), .sw(JB[7]), .enc(w_enc), .anode(an),
.seg_out(seg));
   */
   pixel_gen pg(.clk(clk_100MHz), .reset(reset), .video_on(w_vid_on),
            .x(w_x), .y(w_y), /*.enc(w_enc),*/ .sw(JB), .rgb(rgb_next));
   vga_controller vga(.clk_100MHz(clk_100MHz), .reset(reset), .video_on(w_vid_on),
                  .hsync(hsync), .vsync(vsync), .p_tick(w_tick),
                  .x(w_x), .y(w_y));

   always @(posedge clk_100MHz)
      if(w_tick)
         rgb_reg <= rgb_next;

   assign rgb = rgb_reg;

endmodule
```

aa