UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR

FACULTAD DE INGENIERÍA Y GESTIÓN

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

“SISTEMA INTELIGENTE DE SEGURIDAD BIOMÉTRICA USANDO IoT PARA LA ALERTA DE ROBOS EN LAS RESIDENCIAS DE VILLA MARÍA DEL TRIUNFO”

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de

INGENIERO ELECTRÓNICO Y TELECOMUNICACIONES

PRESENTADO POR EL BACHILLER

SUÁREZ DÍAZ, LEONARDO HILARIO

ASESOR:

Carlos Andres, Mugruza Vassallo

Villa el Salvador

2021

DEDICATORIA

El presente trabajo se lo dedico a mi madre porque gracias a sus cuidados, sus consejos y sacrificio pude salir adelante, también se lo dedico a mi padre poque gracias a su arduo trabajo me brindó una educación y me enseñó los valores de la vida y fue un gran ejemplo a seguir y a Dios por brindarme la fortaleza que se necesita para afrontar los obstáculos de la vida.

AGRADECIMIENTO

Agradezco a mi hermano por apoyarme en mis estudios, a mis familiares por ser una fuente más de inspiración y a mis amigos que demostraron estar en los momentos más difíciles de la vida.

ÍNDICE

LISTADO DE FIGURAS

LISTADO DE TABLAS

RESUMEN

El objetivo de este trabajo de Investigación fue el envío de una señal de Alerta si en caso se produjese una intrusión a la Residencia para que de esta forma el Usuario pueda reportar este incidente a las autoridades respectivas, para ello se realizó un estudio previo de otros trabajos de investigación sobre las formas de implementación de estos Sistemas de Reconocimientos Faciales que sirven para poder dar alerta de una intrusión imprevista en la Residencia, así como también los Algoritmos y materiales que se necesitan para implementar el Sistema. Los Algoritmos que usamos fueron el BackPropagation en el caso de la Red Neuronal y el Algoritmo LBPH para poder llevar a cabo el Reconocimiento Facial. Dentro del Desarrollo del Sistema de Alerta se realizó una planificación para saber cuáles son las etapas a seguir en el proceso de cada Implementación ya sea simulada o en físico, complementariamente se agregó un diagrama de flujo para saber cómo se da el escalamiento de las etapas. Se realizó las programaciones de los Nodos Electrónicos en Arduino para luego poder realizar las simulaciones de cada una de las etapas en la que consiste el Sistema de Alerta y de esta forma poder evaluar los Resultados antes de la implementación física. En la primera fase de la implementación se evaluó el correcto funcionamiento del Sensor HC-sr04 y del Sensor PIR de forma independientemente por lo que se ve necesario realizar una mejora en la programación realizada para que no haya pérdida de información al momento de ser enviados los datos de forma inalámbrica a través del Bluetooth y de esta forma obtener una lectura correcta al momento de que estos sean llevados al campo. La integración de los Sensores con la Raspberry Pi se realizó por partes siendo la Red Neuronal la última etapa en integrarse para poder llevar a cabo el Sistema de Alerta. Obtenemos resultados favorables al momento de dar inicio al Sistema de Alerta por lo que se recibe un mensaje en la aplicación de Telegram en donde se puede observar una imagen de lo sucedido además del número de la comisaría más cerca de la Residencia. Se realizó también un presupuesto del Sistema con la lista de materiales que se usó para que puedan realizar la compra de estos si se desea replicar, así mismo se tiene un Diagrama de Gantt en donde se ve los tiempos que se necesitaron cada una de las etapas para poder llevar a cabo la Implementación del Sistema de Alerta.

INTRODUCCIÓN

El Sistema Inteligente de Seguridad Biométrica se realizó teniendo en cuenta el aumento de la criminalidad en nuestro país afectando a los distintos distritos de Lima siendo uno de ellos Villa María del Triunfo por lo que se ha hecho un estudio previo de la zona en la que nos enfocamos y llegamos a la conclusión de que no existe alguna medida de seguridad para las residencias que suelen ser robadas por delincuentes de la zona o de otras urbanizaciones.

Este trabajo se implementó en el ambiente de la entrada principal en donde los sensores están ubicados en el marco de la ventana ya que es una posible entrada a la residencia, en la puerta ya que es una entrada principal que se usa a diario para que de una de estas dos formas al detectar el sensor de distancia se active el programa de Reconocimiento Facial previamente configurado en el Raspberry Pi que a su vez la cámara cumplirá la función como la de un sensor ya que esta será capaz de detectar si la persona que está ingresando a la Residencia es alguien conocido o alguien desconocido por lo que si fuese una persona desconocida se estaría tomando una captura de imagen de la intrusión que se está produciendo para poder avisar al usuario mediante un mensaje enviado a la aplicación de Telegram, así mismo la Raspberry Pi al detectar este evento emitirá una señal de "1" Lógico a través de su pin 12 configurado como salida y que esta irá al Arduino Uno al pin 2 que está configurado como entrada y también recibirá el valor de "1" Lógico a través de su pin 3 que será enviado por parte del sensor de Movimiento que se comunicará mediante Bluetooth con el Nodo Receptor del Sensor de Movimiento para que de esta forma al obtener los dos estímulos que se necesitan para la Red Neuronal programada en Arduino se pueda activar la alarma que sonará para que los usuarios si se encuentran en la Residencia estén advertidos, para el Reconocimiento Facial se usó el Algoritmo LBPH y para la Red Neuronal se usó el Algoritmo BackPropagation. Nuestro Sistema cuenta con Limitaciones con respecto a la durabilidad de los Nodos Electrónicos ya que estos funcionan con baterías de 9V las cuales en algún momento se agotarán.

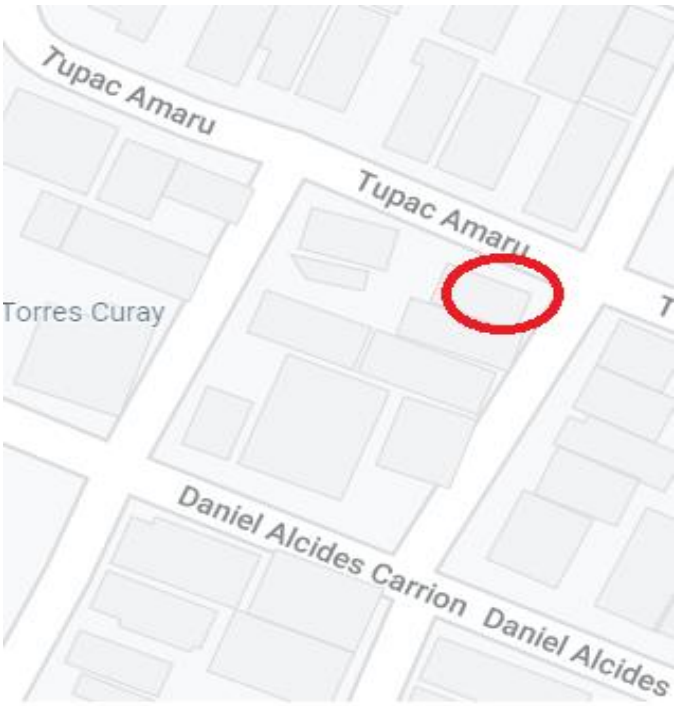
CAPÍTULO I. ASPECTOS GENERALES

1.1 CONTEXTO

Debido al aumento del índice delictivo de la delincuencia en nuestro país, actualmente en la urbanización 1° Mayo Mz D Lt 8 de Villa María del Triunfo señalada con un círculo rojo como se ve en la Figura 1 no existe ningún tipo de protección de rejas o tranqueras en las avenidas Tupac Amaru y Daniel Alcides Carrión que son las vías principales para poder acceder a esta vivienda.

Debido a la falta de seguridad en esta zona surge la idea de crear un sistema de alerta para avisar a los dueños de las Residencias sobre el estado en el cual se encuentra sus hogares, todo esto es posible con el apoyo de los avances de las tecnologías de comunicación que nos serán de gran utilidad al momento de que estos interactúen con la cámara que realizará el reconocimiento facial de las personas que entren a la Residencia así mismo activando los estímulos de la Red Neuronal para poder dar alerta a los usuarios.

Figura 1. Plano de la Ubicación de la Residencia



Fuente: Elaboración Propia

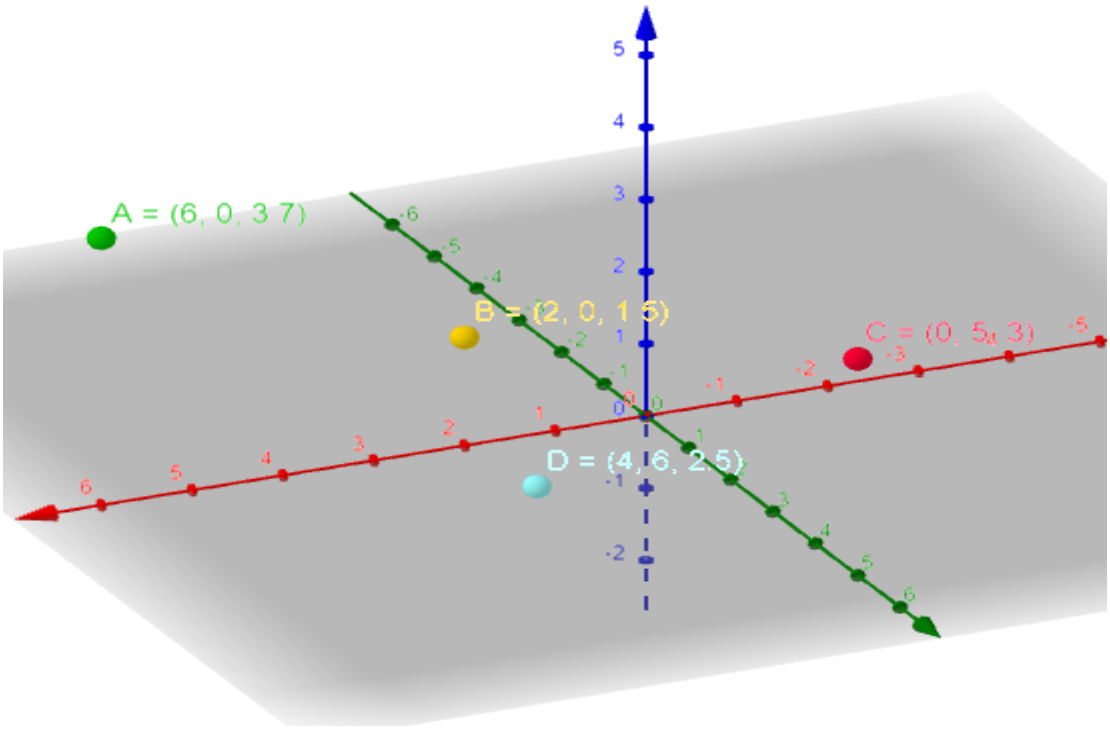
La Figura 2 muestra el espacio físico en donde se plasma el Sistema de Alerta, se detalla las distancias en las cuales son posicionados los sensores de distancia y de movimiento para que de esta forma cumplan con su función. El sensor de la ventana, el de la puerta y el de la pared están a 140 cm, 60 cm y 150 cm respectivamente con respecto al nivel del suelo.

Figura 2. Niveles de Referencia de las Posiciones de los Sensores

Fuente: Elaboración Propia

En la Figura 3 mostramos la ubicación de los sensores en un plano XYZ del ambiente en donde se plasmó el Sistema de Alerta para que de esta forma se vea la distribución de los sensores. El sensor de Distancia que se ubicó en la ventana es el punto A cuyas coordenadas son (6, 0, 3.7), el sensor de distancia que se ubicó en la puerta es el punto B cuyas coordenadas son (2,0,1.5), el sensor de Movimiento que se ubicó en la pared es el punto C cuyas coordenadas son (0, 5, 3) y la cámara se posicionó en el punto D cuyas coordenadas son (4, 6, 2.5).

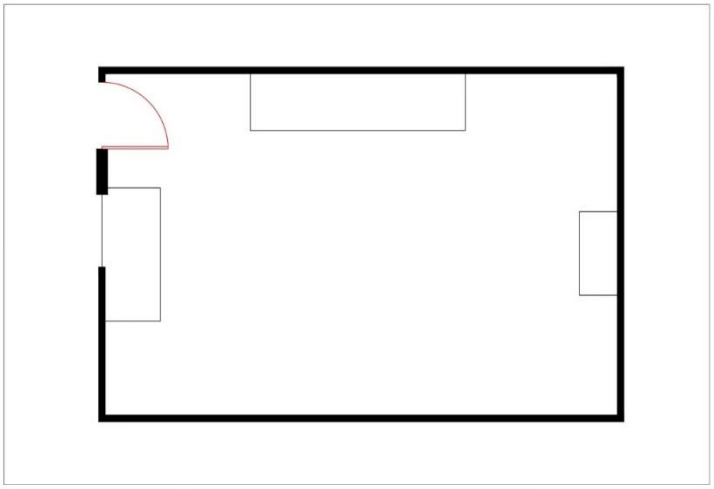
Figura 3. Coordenadas de los Sensores en un Plano XYZ



Fuente: Elaboración Propia

En la Figura 4 podemos observar una vista superior del ambiente en el cual se ha implementado el Sistema de Alerta.

Figura 4. Plano Interior de la Residencia



Fuente: Elaboración Propia

1.2 DELIMITACIÓN TEMPORAL Y ESPACIAL DEL TRABAJO

1.2.1 Delimitación Teórica

En el presente trabajo se desarrolló un prototipo en el cual se usó nodos que se comunicarán de manera inalámbrica, en la cámara se aplicó el reconocimiento facial usando el algoritmo LBPH y por último una red neuronal usando el algoritmo BackPropagation.

1.2.2 Delimitación Espacial

El trabajo se desarrolló en la urbanización 1° Mayo Mz D Lt 8.

1.2.3 Delimitación Temporal

Comprendió desde el mes de septiembre a diciembre del 2021.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar un sistema para alertar posibles robos en las residencias de Villa María del Triunfo usando las tecnologías de IoT para la interacción del usuario.

1.3.2 Objetivos específicos

- ✓ Seleccionar los componentes necesarios para llevar a cabo un sistema adecuado de reconocimiento facial que interactúe con el Sistema IoT.
- ✓ Definir un método digital para poder obtener una lectura correcta de los sensores para el uso del Sistema de Alerta.
- ✓ Desarrollar los algoritmos(programas) de reconocimiento facial basándonos en Python para detectar si las personas son conocidas o desconocidas.
- ✓ Desarrollar una red neuronal en el cual el entrenamiento se de en Matlab para que sea capaz de interpretar lo que sucede en la residencia dependiendo de los estímulos de los sensores.

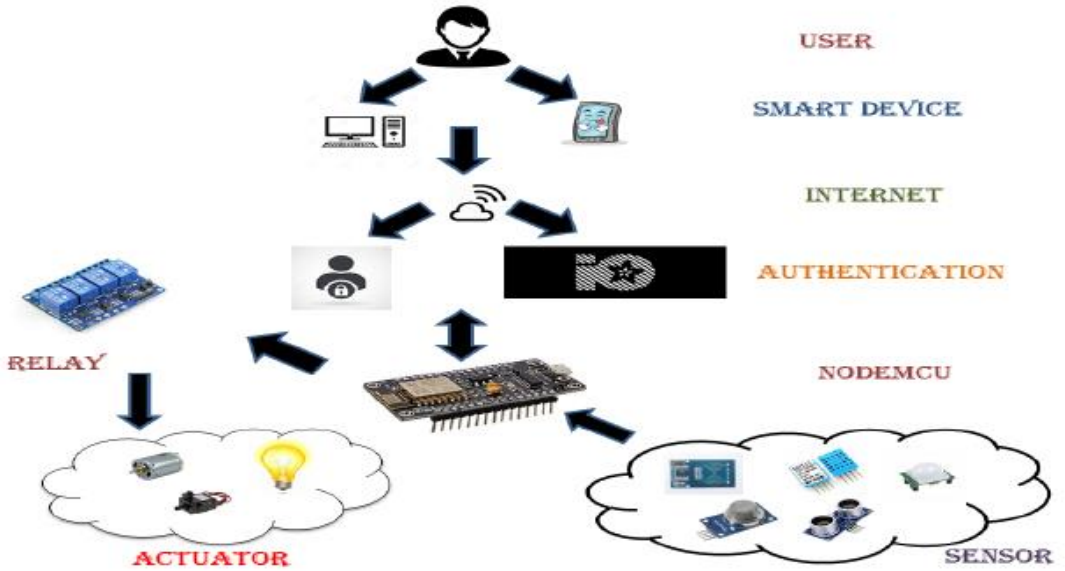
CAPÍTULO II. MARCO TEÓRICO

2.1 Antecedentes de la Investigación

2.1.1 Antecedentes Internacionales

Las tecnologías de las Smart Home (casas inteligentes) se encuentran en el mercado a precios elevados haciendo que no sea accesible para todo el público es por ello que se realiza un sistema de automatización rentable para que cualquiera persona independientemente de su estatus económico pueda tener acceso, se busca que este sistema sea híbrido es decir que el usuario pueda tener acceso del control de su casa cuando se encuentre dentro de ella, así como también cuando se encuentre en el exterior para ello nos basamos en IoT con un interfaz fácil de usar para teléfonos inteligentes y computadoras portátiles como se ve en la Figura 5. El protocolo que se usa es el MQTT que está basado en TCP para la comunicación máquina-máquina. Se usa el IFTT que opera en la web Adafruit IO que utiliza una plataforma para enviar notificaciones al teléfono inteligente del usuario cuando Adafruit detecta una situación anormal (Waheb et al., 2019). El Sistema que se realizó se basó en la estructura anteriormente mencionada en donde el usuario va tener acceso a la cámara cuando se encuentre dentro o fuera de su Residencia conectándose mediante la aplicación VNC Viewer el cual nos permite tomar el control de la Raspberry Pi y poder observar el enfoque de la cámara, ya sea que esté instalada en el dispositivo móvil o en una computadora, los sensores se comunicarán a la Raspberry Pi el cual si es necesario comenzará a ejecutar el programa de reconocimiento facial y si encuentra algo extraño se mandará un mensaje de alerta al usuario usando la aplicación Telegram donde se obtendrá el mensaje con el número de la comisaría más cercana y una foto de la incidencia para que el usuario tome la decisión final si lo reporta o no a las autoridades correspondientes, el Arduino Uno recibe una señal por parte de la Raspberry y el nodo Receptor del Sensor de Movimiento para que active la alarma si cumple las condiciones de la Red Neuronal que se ejecuta en el Arduino Uno.

Figura 5. Forma de conexión del usuario a su dispositivo inteligente para el monitoreo de los sensores

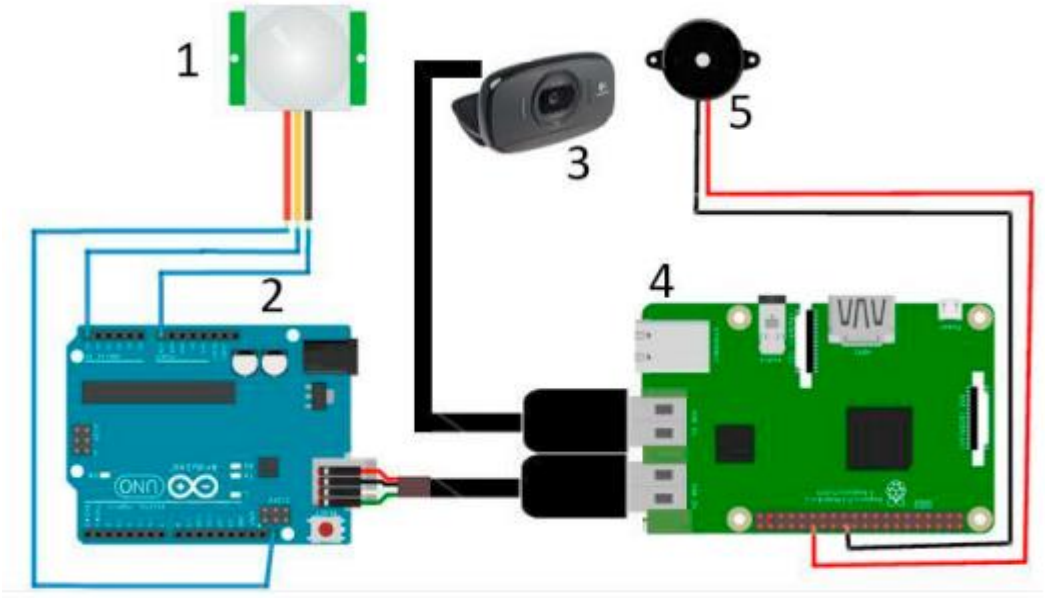


Fuente: Waheb et al (2019, p.12)

Para poder tener una casa segura y así poder evitar los robos que generarían pérdidas de bienes al propietario se creó un Sistema de IoT con la capacidad para detectar objetos extraños usando los métodos HOG y SVM.

El método HOG convertirá la imagen RGB (rojo, verde, azul) a escala de grises, luego el valor del gradiente de cada píxel se calculará dividiéndolo en celdas de 8x8 y la función HOG resultante se procede utilizando el método SVM para determinar si la función es humana o no. En la Figura 6 se puede observar el módulo de procesamiento se usa el Raspberry Pi 3 modelo B, esta placa está equipada con un módulo LAN inalámbrico para la comunicación, Arduino se utiliza para recoger la señal del sensor PIR que están conectados a través de cables, Arduino se conecta a Raspberry Pi a través de un cable USB, para capturar la imagen la cámara web USB se monta en la Raspberry Pi 3 mediante un cable USB y para dar la alarma de advertencia el módulo del zumbador está conectado a Raspberry Pi 3 a través del puerto GPIO (Nico y Wingky, 2018)

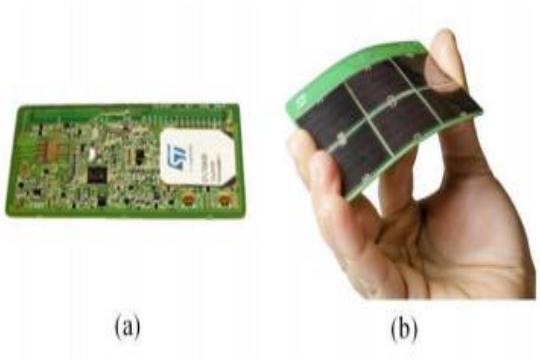
Figura 6. Implementación de los Componentes



Fuente: Nico y Wingky (2018, p.3)

En comparación con otros prototipos el diseño de la Figura 7 tiene ventajas en términos de configuración del sistema, tiene dimensiones reducidas para no poder tener complejidad al momento de su instalación ya que se adecua en cualquier superficie, la potencia que suministra es la más óptima con respecto a los otros diseños, la transmisión de datos se realiza a larga distancia y a una velocidad de 500 Kbps. Se emplea el protocolo de baja energía bluetooth que reduce el consumo de energía y proporciona conexión a teléfonos inteligentes, para mejorar el tiempo de vida de los nodos se utiliza paneles fotovoltaicos para que de esta forma aumentar la durabilidad de los nodos a meses e incluso años (Crispino et al., 2019). Los Nodos que se implementaron para que interactúen en el Sistema de Alerta fueron diseñados con sus medidas exactas para obtener un dimensionamiento reducido, funcionan con una batería de 9 Voltios por lo que se usó componentes electrónicos que requieran un bajo consumo de energía como es el caso del Módulo Bluetooth HC-05 que tiene un alcance de hasta 10 m con una velocidad de transmisión de 1200bps hasta 1.3 Mbps y así alargar la durabilidad de cada Nodo de forma independiente.

Figura 7. Diseño de un nodo electrónico autosustentable



Fuente: Crispino et al (2019, p.5)

Existen plataformas de IoT en el cual se usan sensores para obtener datos y en el caso de que estos datos cumplan ciertas condiciones la cámara toma fotos.
En la primera fase se utilizó el modo manual de la cámara IP, se tomaron 160 imágenes manualmente: 64 con personas y 96 sin personas. En la Figura 8 mostramos un ejemplo con tres imágenes: dos con una persona y otra sin personas. En la segunda fase programamos la cámara para enviar una imagen cuando la cámara detecta algún movimiento en el área. En la Figura 9 la cámara envió una secuencia de imágenes desde el primer momento en que la cámara detecta el movimiento hasta la última detección del mismo movimiento (Gonzales et al., 2017). En nuestro Sistema de Alerta se planteó usar la cámara de 5 MP como un sensor ya que esta al momento de realizar el reconocimiento facial con la ayuda de la Raspberry Pi enviará una señal como lo realizase un sensor y las capturas de las imágenes que se realicen se guardarán dentro de una carpeta de Incidencias previamente creada en la Raspberry Pi en donde se detalla la fecha en la que se dio a cabo el suceso.

Figura 8. Ejemplo con imágenes de interiores tomadas manualmente

Fuente: Gonzales (2017, p.13)

Figura 9. Ejemplo de una de las secuencias de imágenes que envía la cámara IP de Canon cuando detecta movimiento

Fuente: Gonzales (2017, p.14)

Otra forma de mejorar la seguridad del hogar inteligente es calificar los puntos de acceso como primarios o secundarios. Los puntos de acceso primario van a ser aquellos que el usuario utiliza con normalidad para poder entrar o salir de su casa, los accesos secundarios son aquellos puntos de acceso que no son concurrentes por el usuario. Se propone el uso de sensores de movimiento y de proximidad para detectar el comportamiento del usuario en los accesos principales. La ubicación de los sensores debe garantizar que cualquier persona que use la puerta para entrar y salir de la casa no pueda hacerlo sin activar los sensores de movimiento y proximidad. El Algoritmo de Detección Lógica funciona analizando múltiples valores de los sensores antes y después de que la puerta se abra o se cierre como se muestra en la Tabla 1 para saber si el usuario entró o salió del hogar (Arun y Reza, 2017). Se tomó la idea de poder calificar las lecturas de los sensores en nuestro Sistema corresponde a los sensores de distancia y de movimiento los cuales interactúan con el Sistema de Alerta para poder enviar mensaje al usuario, los sensores de distancia se colocaron en la venta y puerta de la entrada principal y el sensor de movimiento en el ambiente que da cuando uno ingresa por la entrada principal, esta calificación de los estados de los sensores nos sirve para poder tener los estímulos que se requieren para el entrenamiento de la Red Neuronal.

Tabla 1. Estados posibles que puede tomar una puerta principal

State No	Initial State → Intermediate State	Final State	Motion Sensor Trigger		Proximity Sensor Trigger		Home Empty
			Before	After	Before	After	
1	C → O	O	✓	✓	✓	✓	F
2	C → O	O	×	✓	×	✓	F
3	C → O	O	×	×	×	×	F
4	C → O	C	✓	×	✓	×	F
5	C → O	C	×	×	×	×	F
6	C → O	C	✓	✓	✓	✓	F
7	O → C	C	✓	✓	✓	✓	F
8	O → C	C	✓	×	✓	×	F
9	O → C	C	×	×	×	×	F
10	O → C	O	×	×	×	×	F
11	O → C	O	✓	×	✓	×	F
12	O → C	O	✓	✓	✓	✓	F
13	C → O	O	✓	×	✓	×	F
14	O → C	C	×	✓	×	✓	F
15	C → O	C	×	✓	×	✓	F
16	O → C	O	×	✓	×	✓	F
17	C → O	C	×	✓	×	✓	T
18	C → O	O	✓	✓	✓	✓	T
19	C → O	O	×	✓	×	✓	T
20	C → O	O	×	×	×	×	T
21	C → O	C	✓	×	✓	×	T
22	C → O	C	×	×	×	×	T
23	C → O	C	✓	✓	✓	✓	T
24	O → C	C	✓	✓	✓	✓	T
25	O → C	C	✓	×	✓	×	T
26	O → C	C	×	×	×	×	T
27	O → C	O	×	×	×	×	T
28	O → C	O	✓	×	✓	×	T
29	O → C	O	✓	✓	✓	✓	T
30	C → O	O	✓	×	✓	×	T
31	O → C	C	×	✓	×	✓	T
32	O → C	O	×	✓	×	✓	T

Fuente: Arun y Reza (2017, p.4)

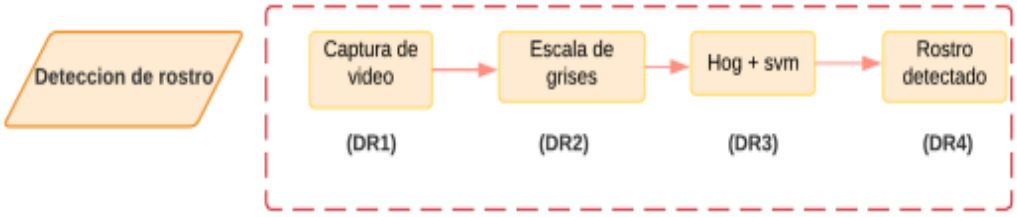
2.1.2 Antecedentes Nacionales

Para poder realizar el algoritmo de detección de Rostro se tiene que seguir una secuencia de pasos tal como se muestra en la Figura 10 en donde la Captura de video se realizó con una cámara que trabaja a 30 fps con una resolución de 1280 x 720 pixeles, se importa la librería de Procesamiento y se crea el objeto VideoCapture que recibe un parámetro con valor 0, el valor 0 es porque se está trabajando con la cámara integrada del equipo. En caso que fuese una cámara externa, se pasa un valor de 1.

Luego la imagen se pasa a escala de grises con la función cvtColor() de Opencv, este proceso se realiza con la finalidad de reducir la intensidad de luz en cada

cuadro de imagen, luego se hace el cálculo de la gradiente, el cálculo del histograma de la gradiente, el cálculo de HGO para luego proceder a caargar el programa de Reconocimiento Facial (Becerra Suárez, 2019)

Figura 10. Diagrama de Secuencia para la detección de Rostro



Fuente: Becerra Suárez (2019, p.63)

Debido a la pandemia surgida a causa del Sars Cov 2 los exámenes de admisión se han vuelto virtuales los cuales para evitar las suplantaciones de los postulantes de la Universidad de Huánuco se usó el Algoritmo LBPH para el reconocimiento facial usando las librerías de Haar-Cascade Classifier obteniendo como resultado un correcto reconocimiento a pesar de la variabilidad de las condiciones ambientales de los participantes (Beraún Barrantes, 2020)

Figura 11. Casos de Reconocimiento

Fuente: Beraún Barrantes (2021, p.46)

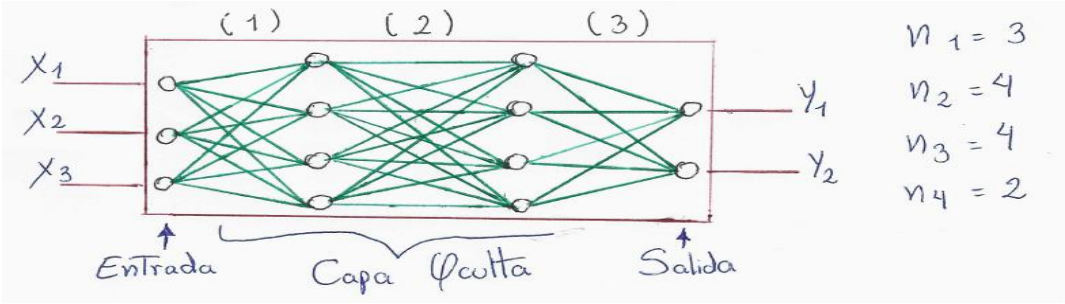
2.2 Bases Teóricas

2.2.1 Algoritmo Backpropagation

El algoritmo de backpropagation se basa en generalizar la regla de aprendizaje de Widrow-Hoff. Utiliza el aprendizaje supervisado lo que significa que el algoritmo se proporciona con ejemplos de las entradas y salidas que la red debe calcular (Arnavat y Bruno, 2015).

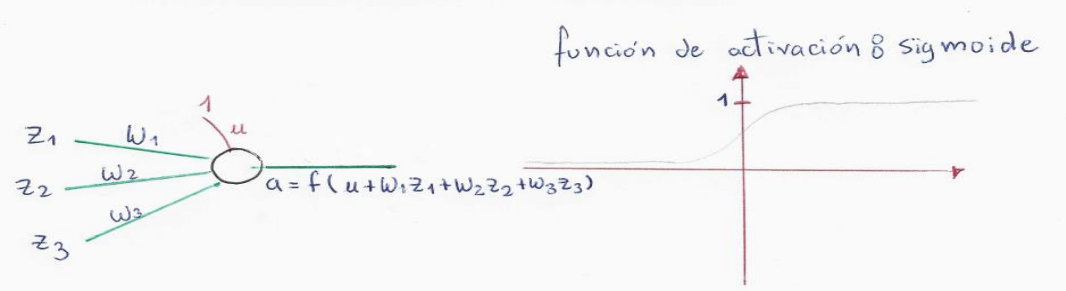
Una red neuronal se conforma como se muestra en la Figura 12, el funcionamiento de una red neuronal se define en la Figura 13.

Figura 12. Red Neuronal



Fuente: Elaboración Propia

Figura 13. Funcionamiento de una red Neuronal teniendo como estímulo una función sigmoide



Fuente: Elaboración Propia

Para poder entender cómo funciona este algoritmo vamos a partir de la función de activación.

$$f_{(x)} = (1+e^{-x})^{-1} \quad \dots(\alpha)$$

$$f_{(x)} = \frac{1}{1+e^{-x}} \quad \dots(\beta)$$

$$f_{(x)}{}^2 = \frac{1}{(1+e^{-x})^2}$$

De β:

$$1 + e^{-x} = \frac{1}{f_{(x)}}$$

$$e^{-x} = \frac{1}{f_{(x)}} - 1$$

Hallamos la primera derivada de la Función Sigmoide de β

$$f'_{(x)} = \left(\frac{1}{1+e^{-x}}\right)'$$

$$f'_{(x)} = \frac{(1+e^{-x})X(1)' - (1)X(1+e^{-x})'}{(1+e^{-x})^2}$$

$$f'_{(x)} = \frac{-(1+e^{-x})'}{(1+e^{-x})^2}$$

$$f'_{(x)} = \frac{-(0+-e^{-x})}{(1+e^{-x})^2}$$

$$f'_{(x)} = (1+e^{-x})^{-2} \times e^{-x}$$

Reemplazamos las expresiones:

$$f'_{(x)} = f_{(x)}{}^2 \times e^{-x}$$

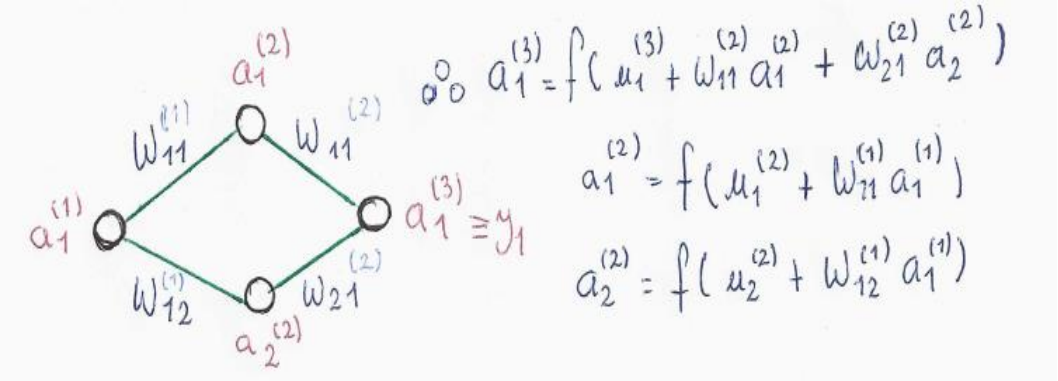
$$f'_{(x)} = f_{(x)}{}^2 \times \left(\frac{1}{f_{(x)}} - 1\right)$$

$$f'_{(x)} = f_{(x)} - f_{(x)}{}^2$$

$$f'_{(x)} = f_{(x)} \times (1 - f_{(x)})$$

En la Figura 14 se va a representar como se obtiene los valores a la salida de la red neuronal

Figura 14. Algoritmo Backpropagation



Fuente: Elaboración Propia

Aplicamos las derivadas parciales para poder ver las variaciones

$$a_1{}^{(3)} = f(u_1{}^{(3)} + w_{11}{}^{(2)} \times f\left(u_1{}^{(2)} + w_{11}{}^{(1)} \times a_1{}^{(1)}\right) + w_{21}{}^{(2)} \times f(u_2{}^{(2)} + w_{12}{}^{(1)} \times a_1{}^{(1)})$$

$$\frac{da_1{}^{(3)}}{dw_{11}{}^{(1)}} = a_1{}^{(3)} \times (1 - a_1{}^{(3)}) \times w_{11}{}^{(2)} \times a_1{}^{(2)} \times (1 - a_1{}^{(2)}) \times a_1{}^{(1)}$$

$$\frac{da_1{}^{(3)}}{dw_{12}{}^{(1)}} = a_1{}^{(3)} \times (1 - a_1{}^{(3)}) \times w_{21}{}^{(2)} \times a_2{}^{(2)} \times (1 - a_2{}^{(2)}) \times a_1{}^{(1)}$$

$$\frac{da_1{}^{(3)}}{dw_{11}{}^{(2)}} = a_1{}^{(3)} \times (1 - a_1{}^{(3)}) \times a_1{}^{(2)}$$

$$\frac{da_1{}^{(3)}}{dw_{21}{}^{(2)}} = a_1{}^{(3)} \times (1 - a_1{}^{(3)}) \times a_2{}^{(2)}$$

$$\frac{da_1{}^{(3)}}{du_1{}^{(2)}} = a_1{}^{(3)} \times (1 - a_1{}^{(3)}) \times w_{11}{}^{(2)} \times a_1{}^{(2)} \times (1 - a_1{}^{(2)})$$

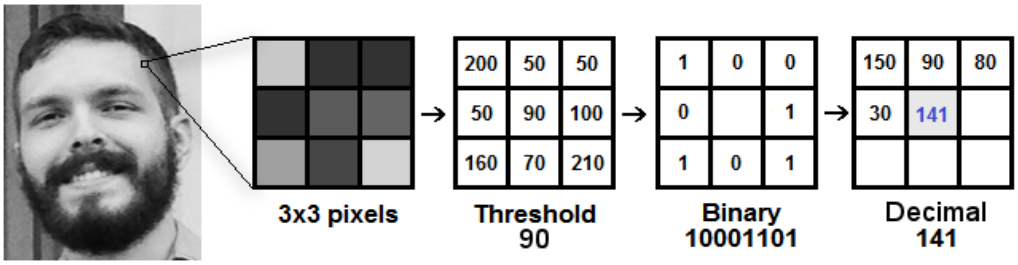
2.2.2 Algoritmo LBPH

El entrenamiento se hace usando el algoritmo LBPH el cual a partir de una imagen a escala de grises se va a extraer una ventana de 3x3 píxeles, la intensidad de cada píxel va de 0-255.

Se toma el valor central de la matriz para usarlo como umbral, este valor se usará para definir los nuevos valores de los 8 vecinos. Se establece el valor de 1 para valores iguales o mayores al umbral y 0 para valores inferiores al umbral.

Obtendremos el número binario de cada posición de la matriz línea por línea, luego se convierte el valor binario en un valor decimal y lo pondremos en el valor central de la matriz tal como lo muestra la Figura 15 (Ojala, Pietikäinen y Harwood, 1996).

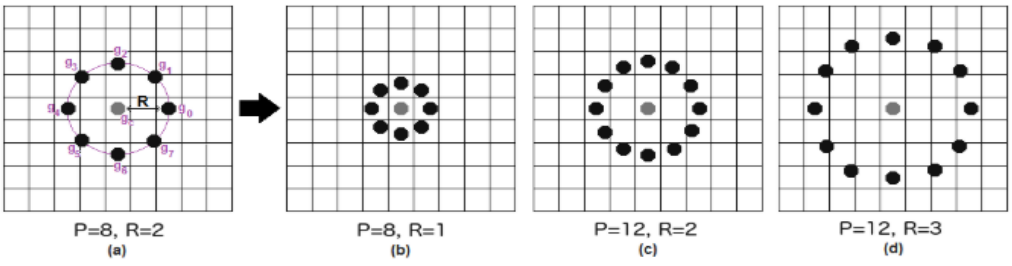
Figura 15. Aplicación del algoritmo LBP



Fuente: ICHI PRO (2021)

El procedimiento LBP se expandió para usar un número diferente de radio y vecinos como se observa en la Figura 16.

Figura 16. Circular LBP

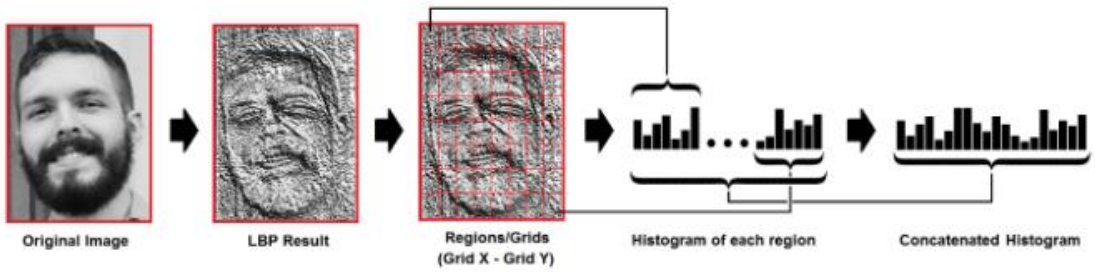


Fuente: ICHI PRO (2021)

Como se tiene una imagen a escala de grises, cada histograma contendrá solo 256 posiciones 0-255 que representan la intensidad de pixel.

Si se obtiene cuadriculas 8x8 se tendrá 8x8x256 = 16384 posiciones en el histograma final. El histograma final representa las características de la imagen original de la imagen como se muestra en la Figura 17.

Figura 17. Extracción de los histogramas



Fuente: ICHI PRO (2021)

2.3 Definición de términos básicos

Algoritmo: "Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema" (Real Academia Española, s.f., definición 1).

Reconocimiento Facial: "El reconocimiento facial es una manera de identificar o confirmar la identidad de una persona mediante su rostro. Los sistemas de reconocimiento facial se pueden utilizar para identificar a las personas en fotos, videos o en tiempo real" (Latam, 2021).

Nodo: "En informática y en telecomunicación, de forma muy general, un nodo es un punto de intersección, conexión o unión de varios elementos que confluyen en el mismo lugar" (Wikipedia, 2021).

Sensor: "Un sensor es un dispositivo que detecta el cambio en el entorno y responde a alguna salida en el otro sistema. Un sensor convierte un fenómeno físico en un voltaje analógico medible" (Dewesoft, 2021).

Trigger: Es la señal que se emite desde el Sensor de Distancia.

Echo: Es el punto final donde llega la señal emitida por el Trigger.

Comunicación Inalámbrica: La comunicación inalámbrica es aquella que se realiza entre 2 o más dispositivos sin que haya un medio físico que los una.

Red Neuronal: "Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas" (International Business Machines, 2021).

CAPÍTULO III. DESARROLLO DEL TRABAJO PROFESIONAL

3.1 Determinación y Análisis del Problema

En el distrito de Villa María del Triunfo la seguridad ciudadana es un tema que se ha dejado de lado dando lugar al aumento de la delincuencia ocasionando los robos en la la urbanización 1° Mayo perjudicando a los vecinos que sufren la pérdida de sus bienes materiales como es el caso que a ocurrido en la Mz D Lt 8 que en 2 ocasiones entraron a robar a esta residencia en horas de la madrugada a través de la ventana llevándose con ellos objetos de valor que se encontraban en la sala.

Este hecho de robo no solo ha sucedido en esta residencia, sino que una cuadra más abajo también tuvo lugar un incidente delictivo que sucedió a plena luz del día en donde 3 delincuentes pudieron palanquear la puerta principal para poder entrar a robar a la residencia llevándose consigo objetos de valor y posteriormente echándose a la fuga en un automóvil.

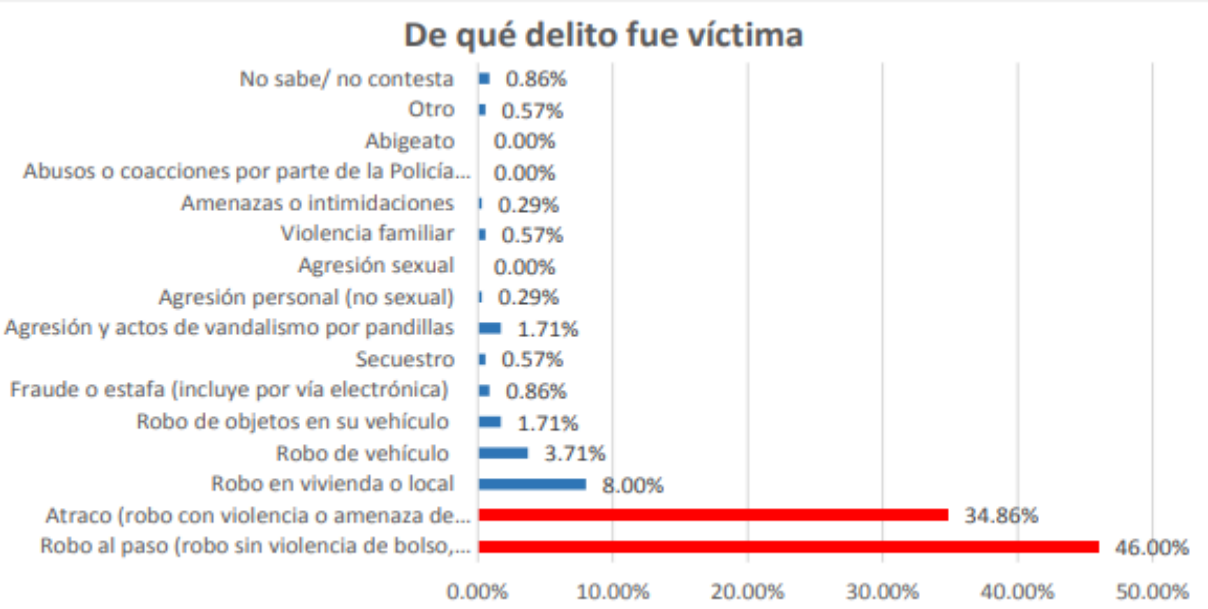
Bajo los indicadores de la tasa de homicidios, tasa de victimización y tasa de internos por lugar de residencia el Ministerio del Interior concluye que son 120 los distritos considerados como los más peligrosos.

En el caso de la capital, se detalla que entre los distritos con mayor índice de criminalidad se encuentran Lima, La Victoria, El Agustino, Rímac, Breña y Villa El Salvador.

Así también, Independencia, Comas, San Juan de Lurigancho, Ancón, Villa María del Triunfo, San Juan de Miraflores, Ate, Lince, Los Olivos, entre otros. En el caso del Callao, se identifican, por ejemplo, Bellavista, La Perla, Carmen de la Legua Reynoso y Ventanilla (Gestión, 2019)

Como se puede observar en el diagrama de barras horizontal de la Figura 18 tenemos como resultado de una encuesta realizada por la Municipalidad de Villa María del Triunfo a los pobladores de este Distrito que el índice delictivo más común es el Robo al paso que refleja un 46.00 %, Atraco con 34.86 % y el Robo en vivienda o Local siendo un 8.00 % relacionado con el objetivo general, en nuestro tema abarcaremos sobre los robos a las viviendas.

Figura 18. Agravios más frecuentes en el Distrito de Villa María del Triunfo



Fuente: Encuesta sobre la percepción de seguridad ciudadana realizada por Sub Gerencia de Planeamiento 2019

El Plan de Acción Distrital de Seguridad Ciudadana 2021 de Villa María del Triunfo realizó una recopilación de información de los años de 2017, 2018, 2019 y 2020 en 5 comisarías de Villa María del Triunfo las cuales se ubican en Cercado, José Carlos Mariátegui, Nueva Esperanza, Tabla de Lurín y José Gálvez, indican que entre las incidencias delictivas se centran principalmente cinco delitos recurrentes los cuales son los siguientes:

- ✓ Contra la seguridad pública.
- ✓ Contra la libertad.
- ✓ Contra la vida, el cuerpo y la salud.
- ✓ Contra el patrimonio.
- ✓ Contra la Administración.

Se observa en la Tabla 2 como los delitos contra el Patrimonio en el 2017 fue de 4257 incidencias delictivas la cual para el 2018 tuvo un incremento a 5118 incidencias delictivas y este para el año 2019 siguió aumentando a 5278 incidencias delictivas sin embargo en el 2020 vemos un decremento de estas cifras a 1501 incidencias delictivas, este decremento se ha debido a la crisis sanitaria que se está viviendo actualmente.

Tabla 2. Delitos más comunes en el distrito de Villa María del Triunfo durante el 2017, 2018, 2019 y 2020

Tipo de Delitos	2017	2018	20219	2020	Total
Contra la seguridad Pública	527	474	561	147	1709
Contra la Libertad	410	590	731	284	2015
Contra la vida, el cuerpo y la salud	811	919	913	308	2951
Contra el Patrimonio	4257	5118	5278	1501	16154
Contra la Administración Pública	21	23	62	65	171
Total	6026	7124	7545	2305	23.000

Fuente: 5 comisarías-REGPOL-LIMA/DIVPOL-SUR3-2020.

Los siguientes sucesos ocurrieron en Villa María del Triunfo en el presente año del 2021

Villa María del Triunfo: Delincuentes roban tableta y celular que escolar utilizaba para clases virtuales.

Este suceso ocurrió en el asentamiento humano San Camilo cuando no había nadie en el domicilio, los ladrones se llevaron Tablet, celular, equipo de sonido con un valor de 2,500 soles y dinero en efectivo.

La denuncia se realizó en la Comisaría de José Gálvez (América Noticias, 2021)

Villa María del Triunfo: Sujeto armado asalta peluquería en menos de 30 segundos.

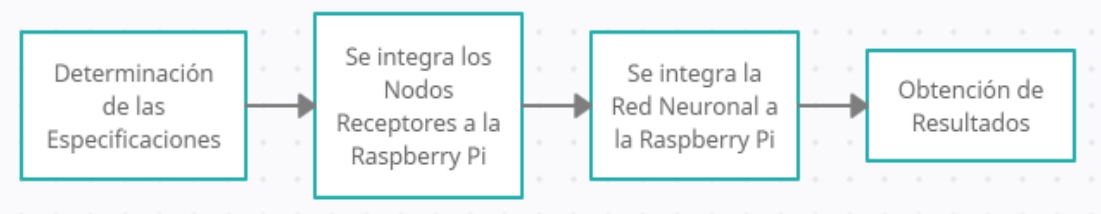
El hecho sucedió en un local de peluquería ubicado en la avenida 28 de Julio, zona que pertenece a Nueva Esperanza en donde todo lo ocurrido fue captado por una cámara de seguridad en donde el delincuente amenazó con disparar a toda persona que no le daba sus bienes llevándose tres celulares y una cartera valorizada en 400 soles (América Noticias, 2021).

3.2 Modelo de Solución Propuesto

3.2.1 Planificación del Sistema

La Figura 19 muestra el diagrama de Bloques de forma General de cómo se fue dando cada uno de las etapas para poder llegar a cumplir los objetivos mencionados.

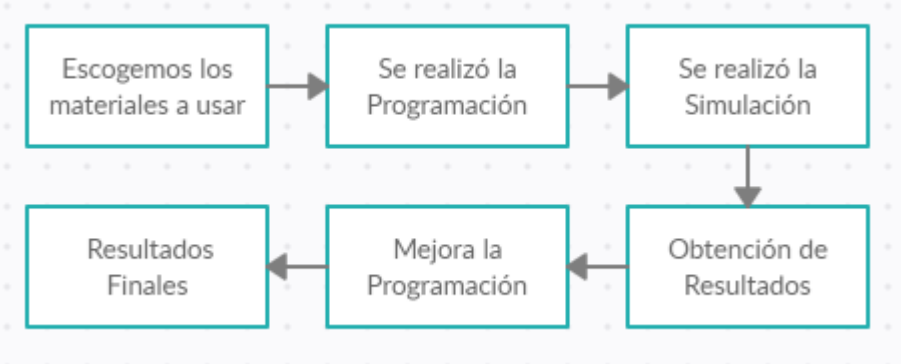
Figura 19. Diagrama de Bloques de la Implementación del Sistema de Alerta



Fuente: Elaboración Propia

En la Figura 20 se detalla cómo fue la implementación de los Nodos Electrónicos partiendo desde la elección de los materiales hasta la etapa de Resultados Finales en donde se obtiene un correcto funcionamiento de los Nodos al momento de enviar la información de un extremo a otro de forma inalámbrica.

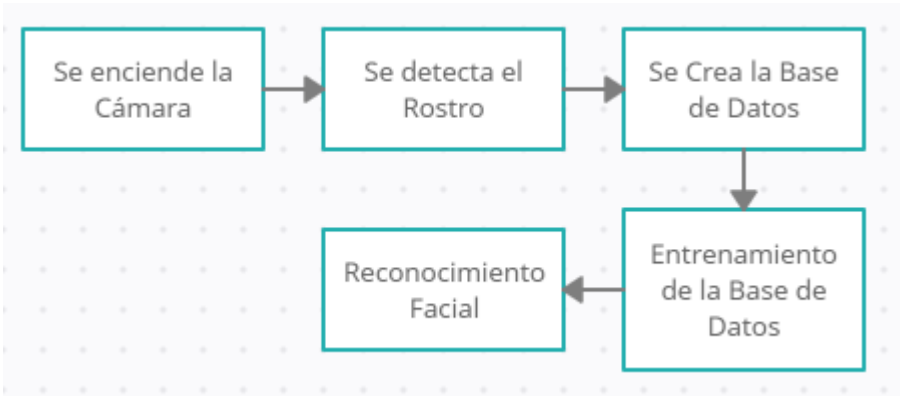
Figura 20. Diagrama de Bloques de los Nodos Electrónicos



Fuente: Elaboración Propia

Las etapas que comprende el proceso de Reconocimiento Facial se detallan en la Figura 21 en donde se observa el avance de las lógicas de programación desde lo mas básico como lo es encender la cámara de la Raspberry Pi hasta llegar al proceso de Reconocimiento Facial.

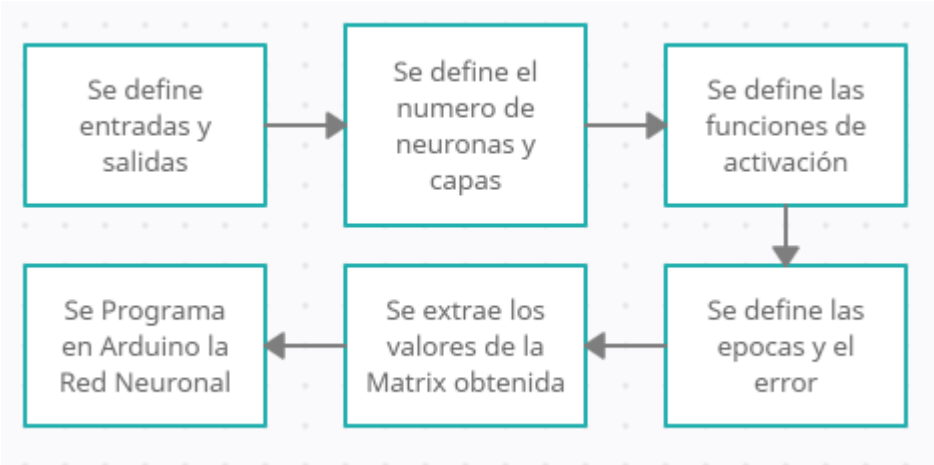
Figura 21. Diagrama de Bloques del Reconocimiento Facial



Fuente: Elaboración Propia

La Red Neuronal implementada y Simulada sigue los pasos que se mencionan en la Figura 22.

Figura 22. Diagrama de Bloques de la Red Neuronal



Fuente: Elaboración Propia

3.2.2 Determinación de las Especificaciones Requeridas

Para poder llevar a cabo nuestro trabajo hemos analizado con mucho cuidado los requerimientos que se necesitan en cada una de las etapas para que estas puedan cumplir sus funciones de la mejor manera posible es por ello que en la etapa de la implementación de los Nodos Electrónicos se tuvo que tener en cuenta lo siguiente:

- Sensor de distancia con rango de medición mayor a 1 metro
- Bajo consumo energético
- Componentes con Dimensiones reducidas

Así mismo en la etapa de Reconocimiento Facial las cámaras Domo se utilizan en su mayoría para interiores del hogar sin embargo estas cuentan con entrada BNC las cuales no se podrían conectar de forma directa a la Raspberry Pi por ese motivo aprovechando la entrada CSI que la Raspberry Pi poseé se usó el Módulo de Cámara de 5 MP porque a comparación del Módulo de Cámara de 8 MP esta cuenta con visión nocturna ya que se puede integrar fácilmente los leds infrarrojo.

Existen distintos Arduinos en el mercado, pero basándonos en sus características más relevantes como:

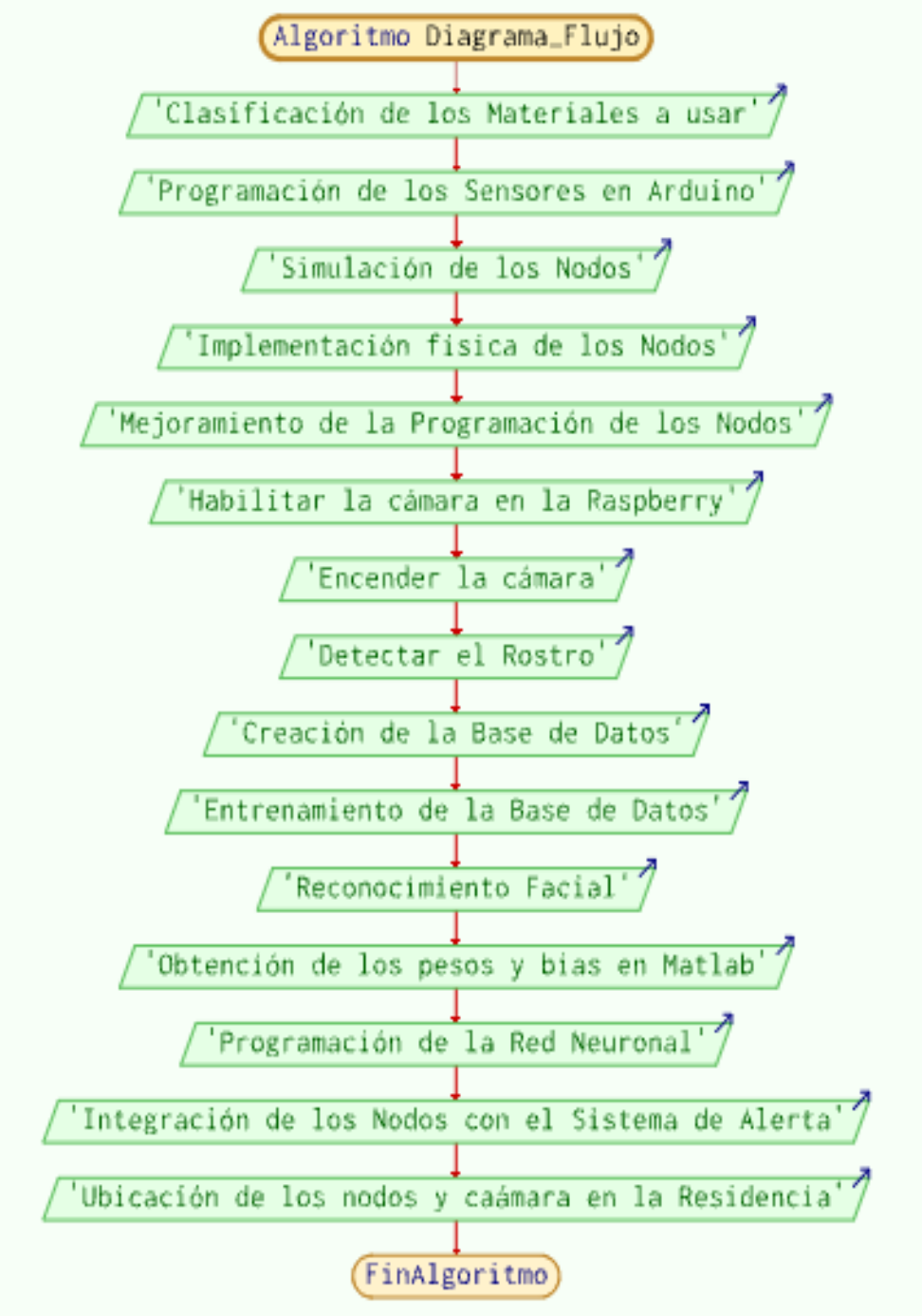
- Procesador
- Voltaje
- E/S Digital

Optamos por el Arduino Nano para su uso en los Nodos de Transmisión y Recepción de datos ya que solamente se usará 5 pines digitales en la Transmisión de información y 3 pines digitales en la Recepción de información. En el caso de la Red Neuronal se implementó en el Arduino Uno

3.2.3 Flujo de Trabajo

En la Figura 23 se observa el Flujo de Trabajo desarrollado en donde se implementó el Sistema de Alerta para las Residencias de Villa María del Triunfo.

Figura 23. Diagrama de Flujo del Desarrollo del Sistema de Alerta

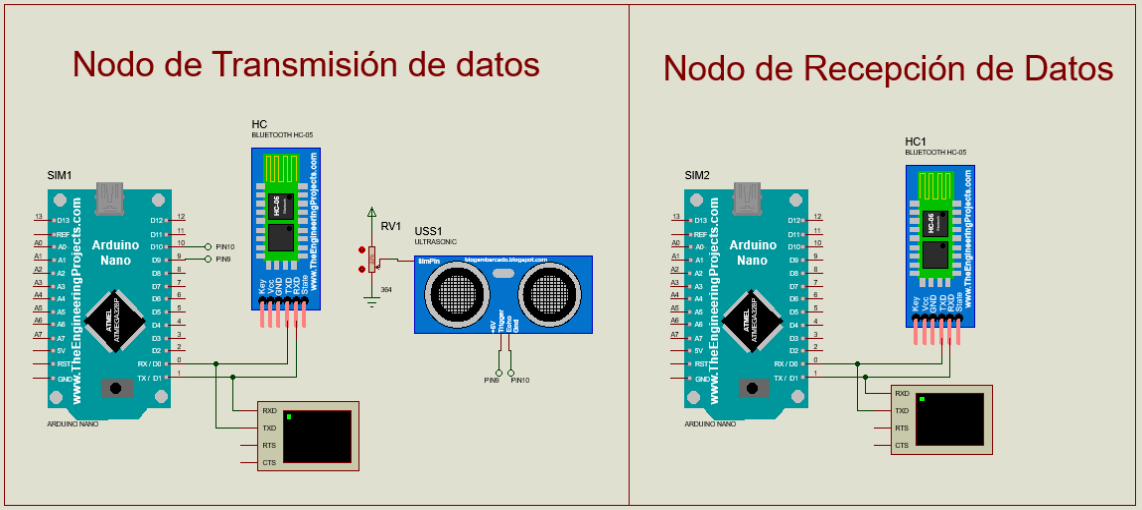


Fuente: Elaboración Propia

3.2.4 Simulación del Sistema

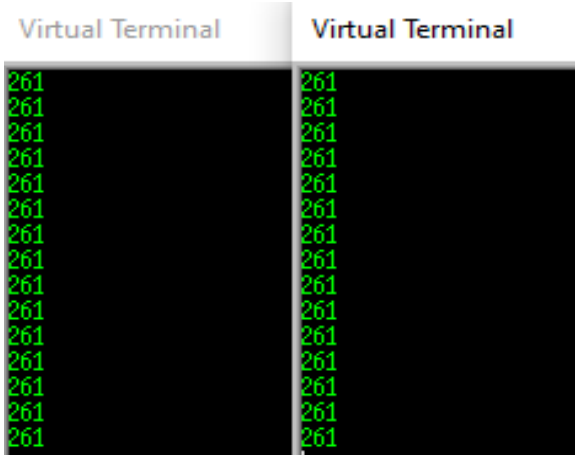
En la Figura 24 vemos como los nodos han sido simulados en el software *Proteus 8 Professional* para ello se usó la librería de Arduino Nano, Módulo Bluetooth HC-05, Ultrasonic, POT-HG que es necesario para simular el choque con un objeto y usando el Virtual Terminal podemos observar que la transmisión y recepción de datos en ambas partes están sincronizados por lo que la programación realizada entre los Arduinos Nanos son eficientes. Los resultados de la simulación muestran que los 15 paquetes enviados son recibidos con los mismos valores en los terminales (ventanas con fondo negro de la Figura 25) tanto de transmisión como de recepción.

Figura 24. Simulación del nodo Transmisor del Sensor de Distancia usando el software Proteus



Fuente: Elaboración Propia

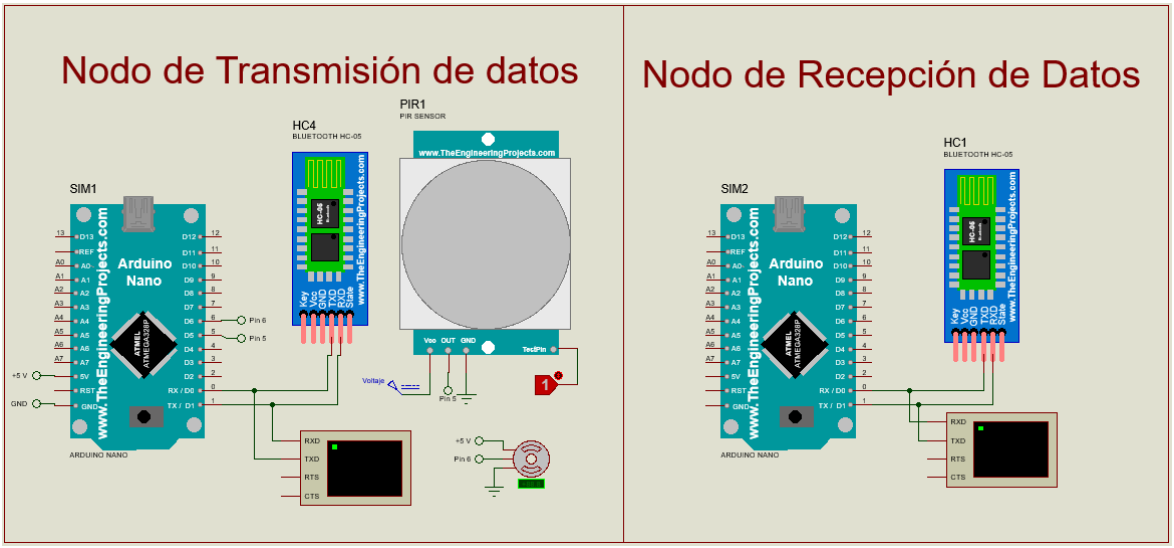
Figura 25. Resultados de la Simulación



Fuente: Elaboración Propia

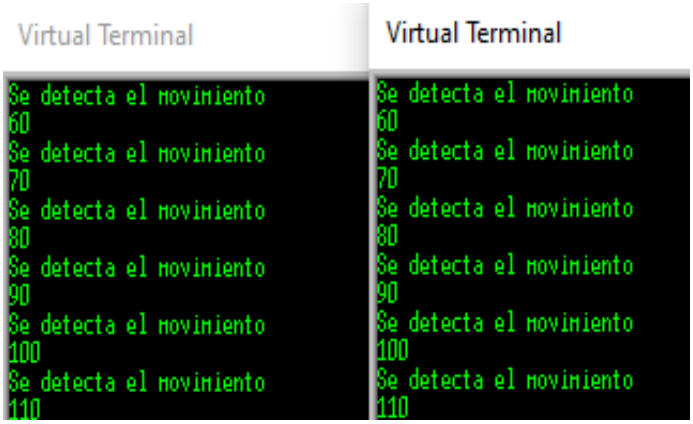
En la Figura 26 se muestra el Nodo del Sensor de Movimiento y el Nodo Receptor implementados en el software *Proteus 8 Professional,* en este caso se está usando un servomotor en el Nodo de Transmisión de datos porque de esta forma se puede sensar una mayor área al momento que el sensor gire con el servomotor y usando el virtual terminal podemos observar que la transmisión y recepción de datos en ambas partes están sincronizados por lo que la programación realizada entre los Arduinos Nanos son eficientes. Los resultados de la simulación muestran que los 12 paquetes enviados son recibidos con los mismos valores en los terminales (ventanas con fondo negro de la Figura 27) tanto de transmisión como de recepción.

Figura 26. Simulación del nodo Transmisor del Sensor de Movimiento usando el software Proteus



Fuente: Elaboración Propia

Figura 27. Resultados de la Simulación



Fuente: Elaboración Propia

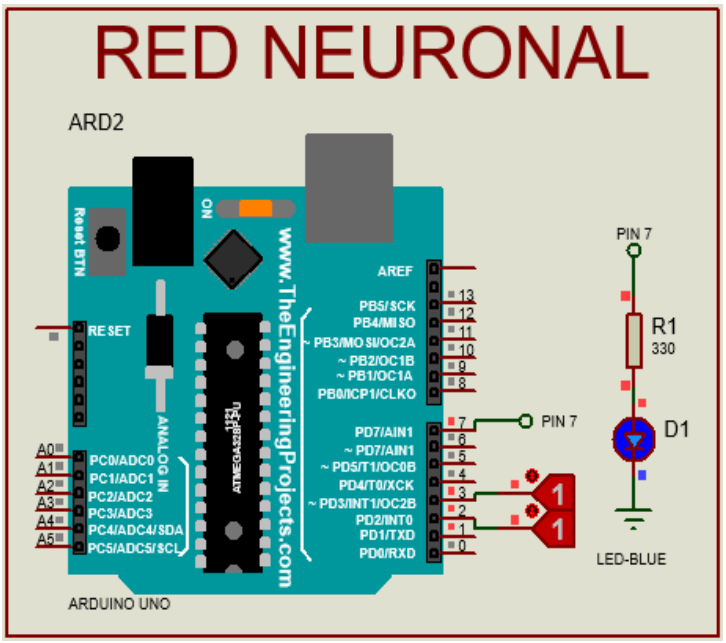
En la Figura 28 se muestra la simulación de la Red Neuronal implementada en el software *Proteus 8 Professional* en el cual se tiene los pines 2 y 3 del Arduino Uno programados como entrada, esta simulación se realizó de forma independiente por lo que se usó el Logictoggle para simular las entradas del Sistema, bajo las condiciones de que ambas entradas reciban un voltaje de entrada se emitirá la señal de Alarma representada con un Led Azul.

Después de haber Simulado los Nodos Emisores, Receptores y la Red Neuronal en Arduino se pasa a la etapa de la integración de los Nodos Receptores a la Raspberry Pi la cual está comprendida por 2 Arduinos Nano, 2 Módulo Bluetooth HC-05 tal como se muestra en la Figura 29 en donde los pines de entrada son el 11 y 13 siendo el pin de salida el 12, la Raspberry Pi recibe los estímulos de los Sensores para luego poder emitir una función específica, en este caso de Simulación se encenderá un Led cuando reciba la señal de los Nodos Receptores.

En la Figura 28 se presentó la Simulación de forma independiente de la Red Neuronal en Arduino por lo que ahora en la Figura 30 se muestra la Integración de la Red Neuronal al Nodo Receptor del Sensor de Movimiento para poder evaluar el funcionamiento en conjunto, teniendo como resultado una buena integración en donde la Alarma simbolizada por el Led Azul se enciende después de detectar el Movimiento.

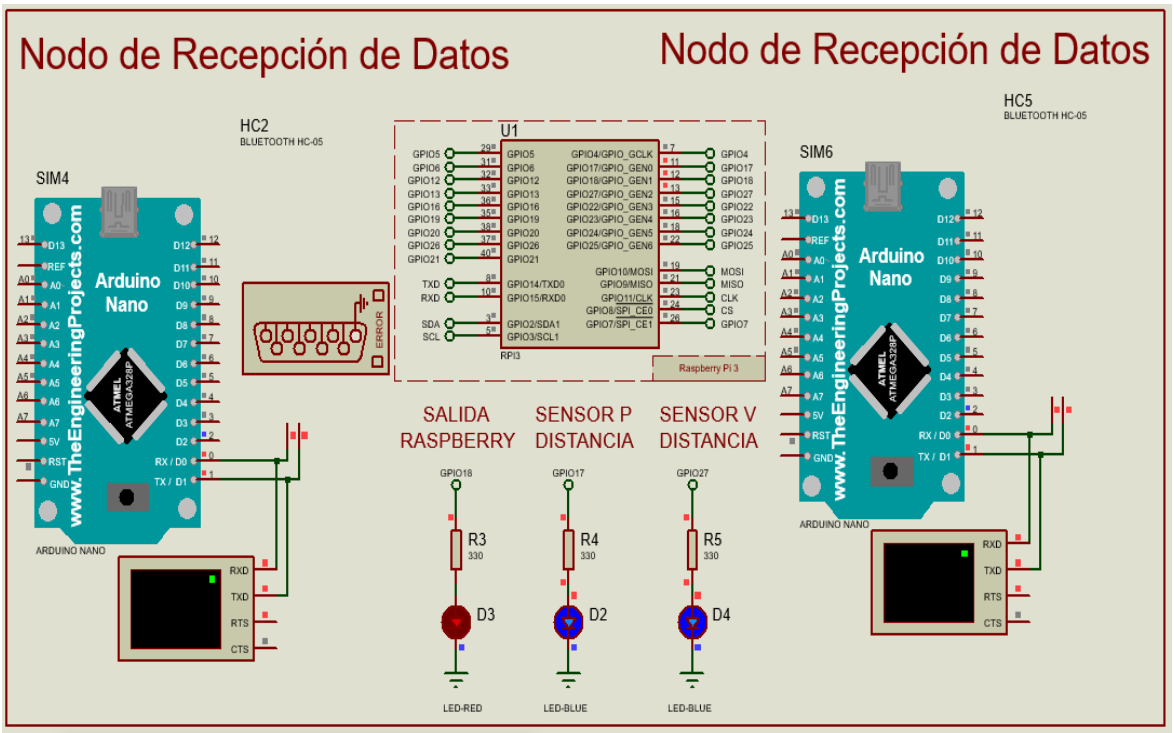
En la Figura 31 se puede observar los 3 Nodos Receptores pertenecientes a los Nodos Transmisores del Sensor de Distancia y del Nodo Transmisor del Sensor de Movimiento quienes se comunicarán a la Raspberry Pi incluyendo la Red Neuronal realizada en Arduino, las entradas de los estímulos de los Sensores son el pin 11 y 13 que según lo que reciban se mandará una señal de HIGH o LOW por el pin 12 de la Raspberry Pi que servirá de estímulo para el Arduino Uno que en conjunto con la señal que reciba del Sensor de Movimiento se activará la alarma representada por el Led de color Azul.

Figura 28. Simulación de la Red Neuronal usando Arduino Uno el software Proteus



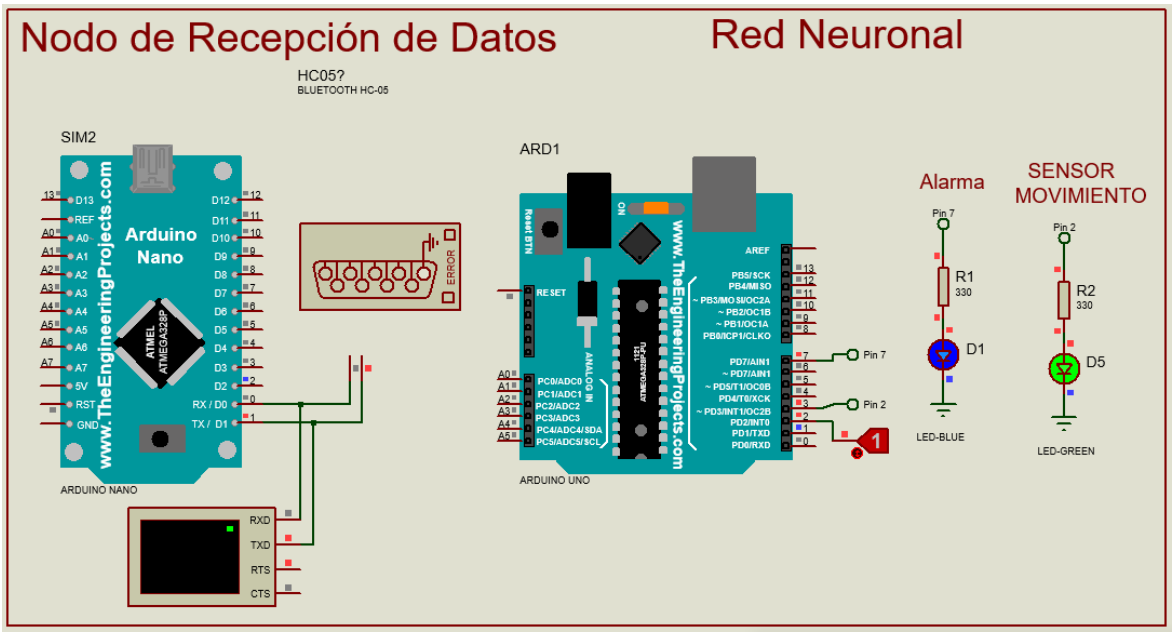
Fuente: Elaboración Propia

Figura 29. Simulación de la Integración de los Nodos Emisores de los Sensores de Distancia con la Raspberry Pi usando el software Proteus



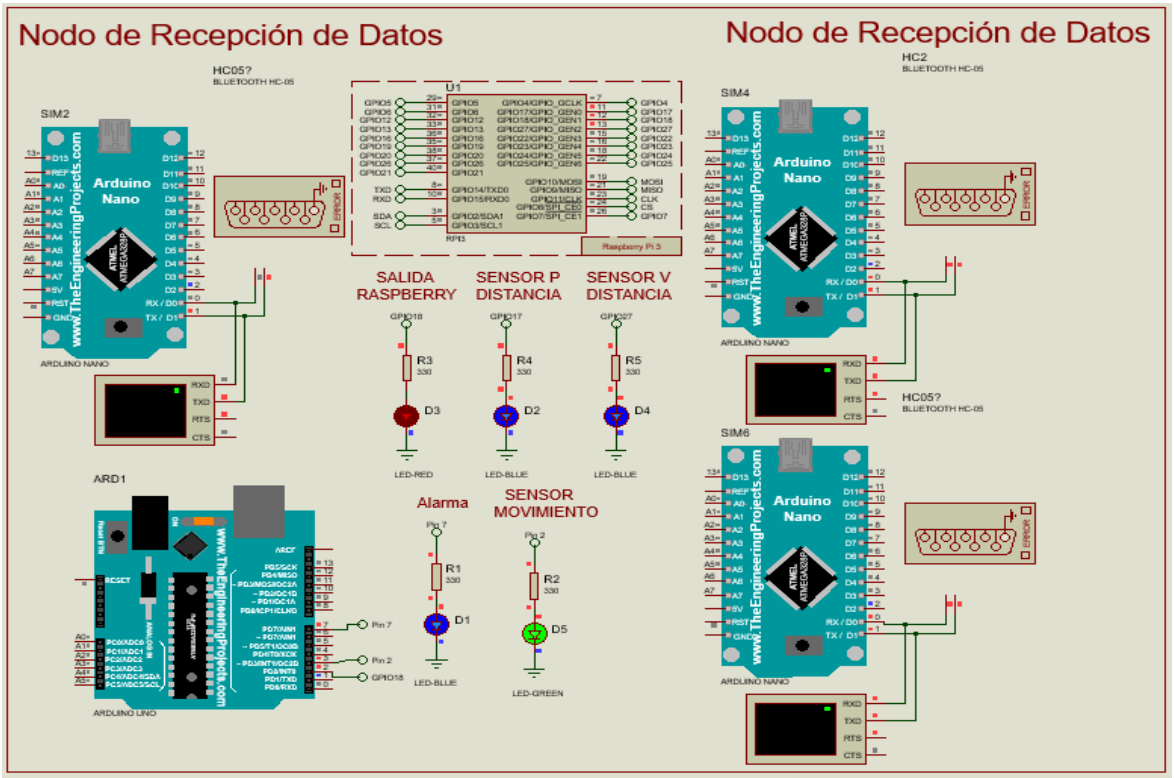
Fuente: Elaboración Propia

Figura 30. Simulación de la Integración del Nodo Emisor de Movimiento con la Red Neuronal en Arduino Uno usando el software Proteus



Fuente: Elaboración Propia

Figura 31. Simulación Final de la Integración de los Nodos Receptores y la Red Neuronal en Arduino Uno usando el software Proteus



Fuente: Elaboración Propia

3.2.5 Librerías usadas en Eagle

Se usó el software *Eagle 7.6.0 Professional* para poder desarrollar las interconexiones de los componentes electrónicos que se usaron para el desarrollo de los nodos.

Eagle a pesar de contar con varias librerías ya preinstaladas para su uso hay ocasiones en las cuales no podemos encontrar los componentes que vamos a requerir para la implementación de nuestra placa y es por ello que desarrollamos nuestras propias librerías el cual detallaremos a continuación.

3.2.5.1 Creación del Package

Para el diseño del encapsulado del Arduino Nano, el Módulo Bluetooth HC-05 y el Sensor Ultrasónico HC-sr04 se tiene que tener en consideración la separación de 2.54 mm por cada uno de sus 15, 6 y 4 pines respectivamente para que de esta forma al momento de realizar la impresión de la placa sea exacta al componente en físico, estos encapsulados se pueden observar en las Figuras 32, 34 y 36.

3.2.5.2 Creación del Symbol

Al momento de realizar el símbolo del Arduino Nano, el Módulo Bluetooth HC-05 y el Sensor Ultrasónico HC-sr04 vamos a tener la posibilidad de acomodar los pines de forma libre de acuerdo a nuestra comodidad para desarrollar la interconexión con los demás componentes electrónicos, es aquí en donde se tiene que nombrar cada pin del componente electrónico para poder realizar el correcto diagrama del circuito del Nodo Emisor y Receptor tal como se muestra en la Figura 33, 35 y 37.

3.2.6 Diagrama Esquemático de los Nodos Electrónicos

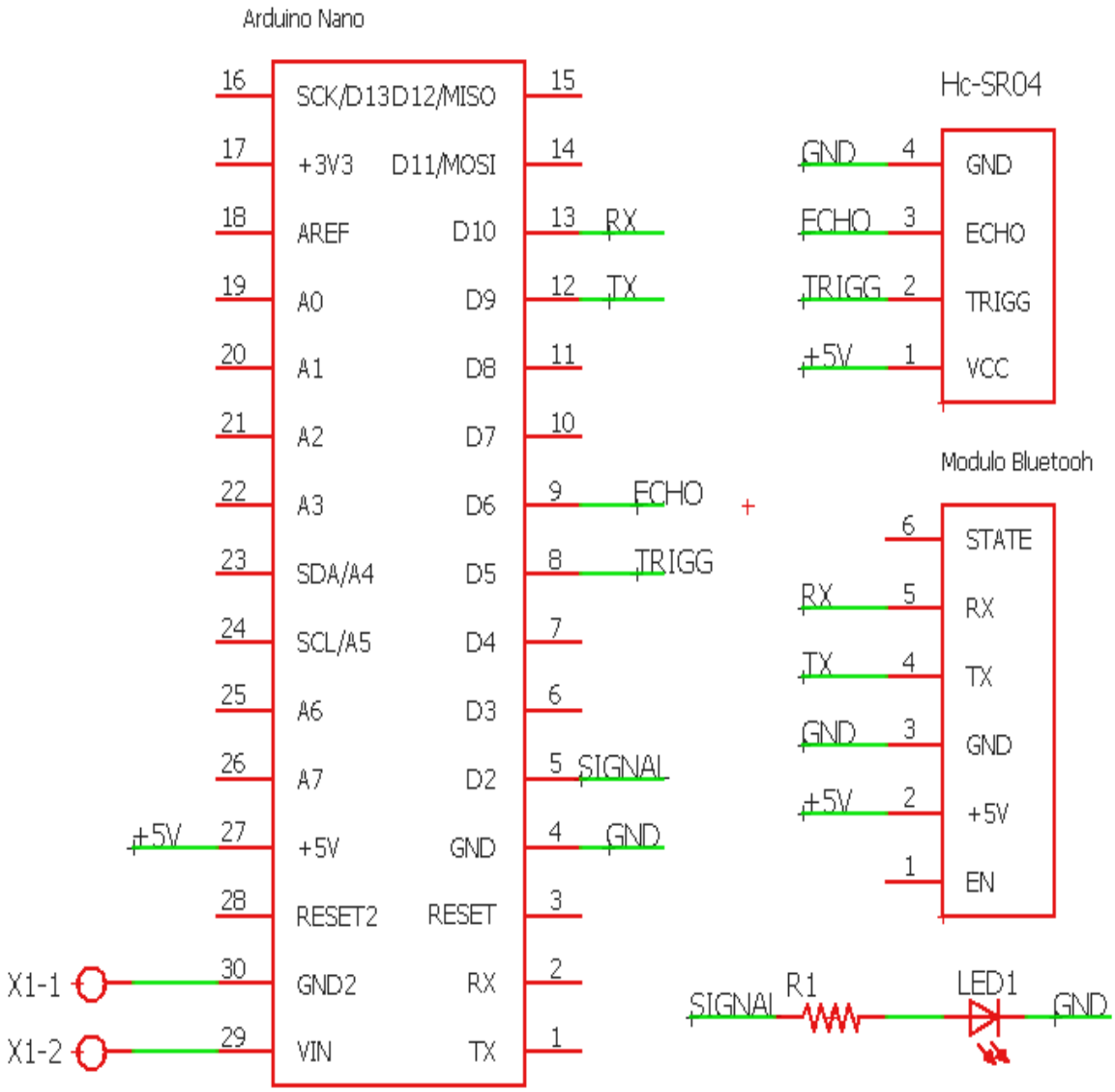
3.2.6.1 Nodo Emisor del Sensor de Distancia

Schematic del Nodo Emisor del Sensor de Distancia

En el *Schematic* del Nodo Emisor de la Figura 32 vamos a ver como a cada pin se le va asignar un nombre el cual tiene que ser igual al nombre del pin del componente electrónico al cual deseamos conectar.

El nodo Emisor del Sensor de Distancia está compuesto del Arduino Nano, el sensor ultrasónico HC-SR 04, el módulo Bluetooth HC-05, un resistor de 330 ohmios y un led.

Figura 32. Schematic del Nodo Emisor del Sensor de Distancia

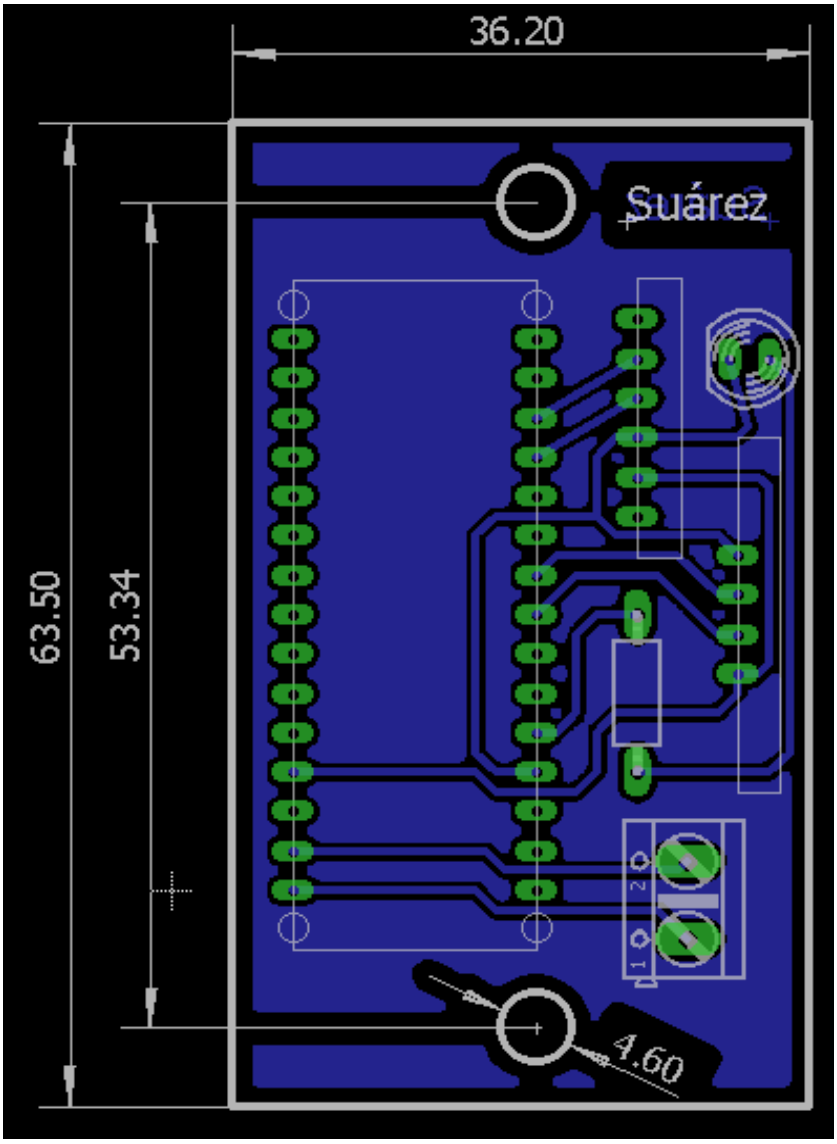


Fuente: Elaboración Propia

Board del Nodo Emisor del Sensor de Distancia

El board que observamos en la Figura 33 es el diseño que se realizó para la placa del Nodo Emisor que está posicionado en la ventana y en la puerta, podemos apreciar la distribución de los componentes electrónicos para que puedan ocupar un espacio reducido.

Figura 33. Board del Nodo Emisor del Sensor de Distancia



Fuente: Elaboración Propia

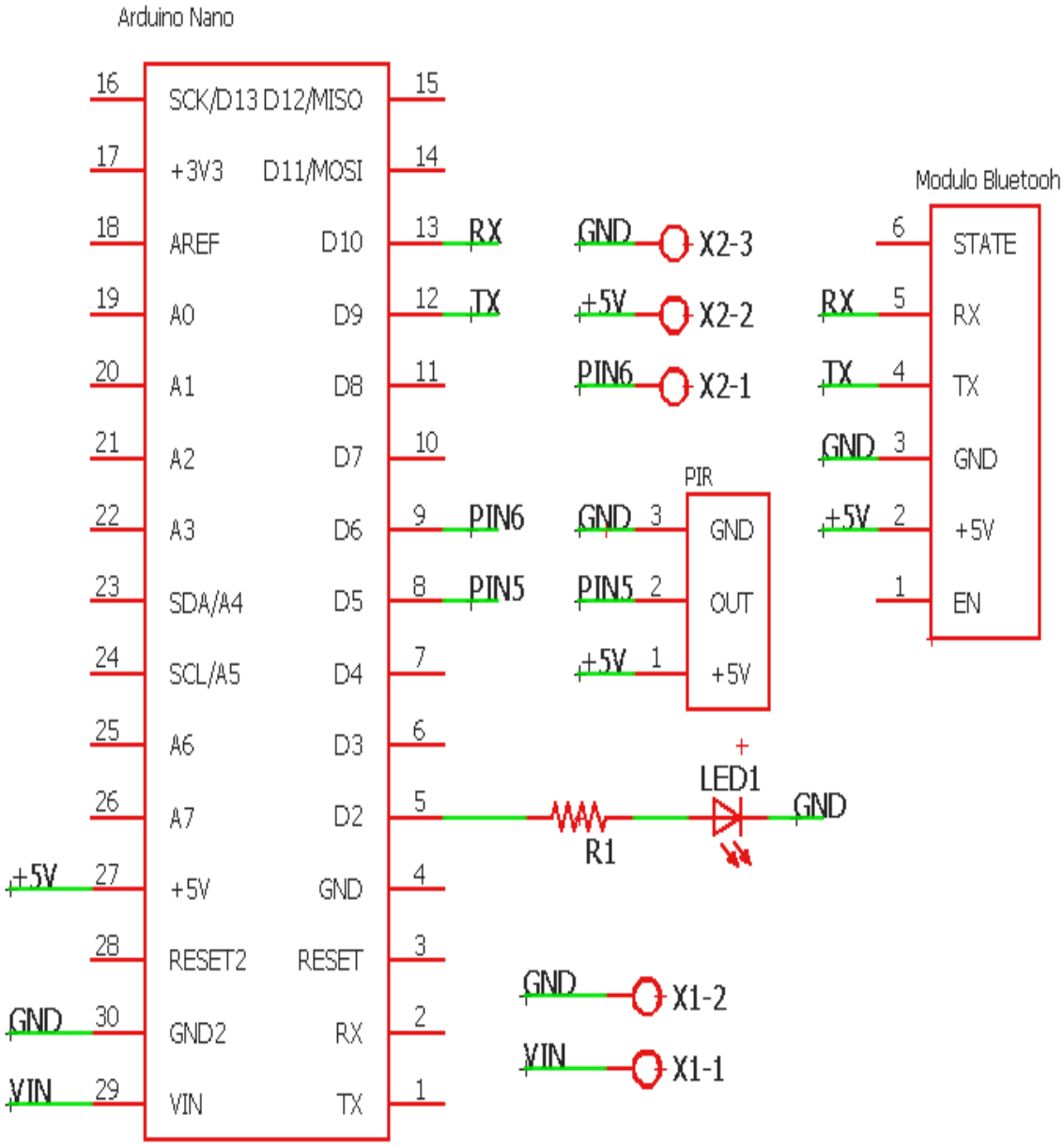
3.2.6.2 Nodo Emisor del Sensor de Movimiento

Schematic del Nodo Emisor del Sensor de Movimiento

En el *Schematic* del Nodo Emisor de la Figura 34 vamos a ver como a cada pin se le va asignar un nombre el cual tiene que ser igual al nombre del pin del componente electrónico al cual deseamos conectar.

El nodo Emisor está compuesto del Arduino Nano, el sensor PIR, el módulo Bluetooth HC-05, un servomotor, un resistor de 330 ohmios y un led.

Figura 34. Schematic del Nodo Emisor del Sensor de Movimiento

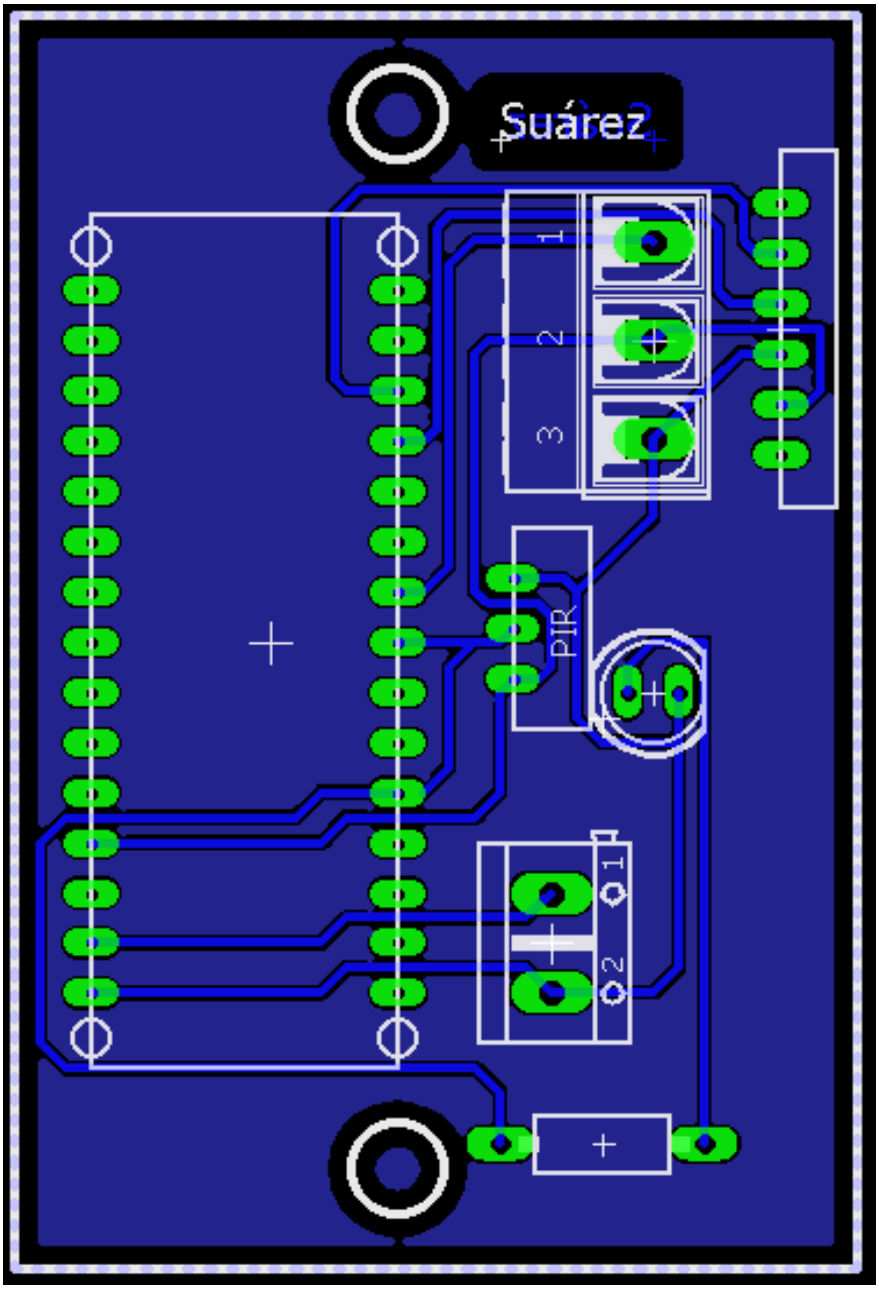


Fuente: Elaboración Propia

Board del Nodo Emisor del Sensor de Movimiento

El board que observamos en la Figura 35 es el diseño que se realizó para la placa del Nodo Emisor que irá posicionado en la pared a una altura de 150 cm, podemos apreciar la distribución de los componentes electrónicos para que puedan ocupar un espacio reducido.

Figura 35. Board del Nodo Emisor del Sensor de Movimiento



Fuente: Elaboración Propia

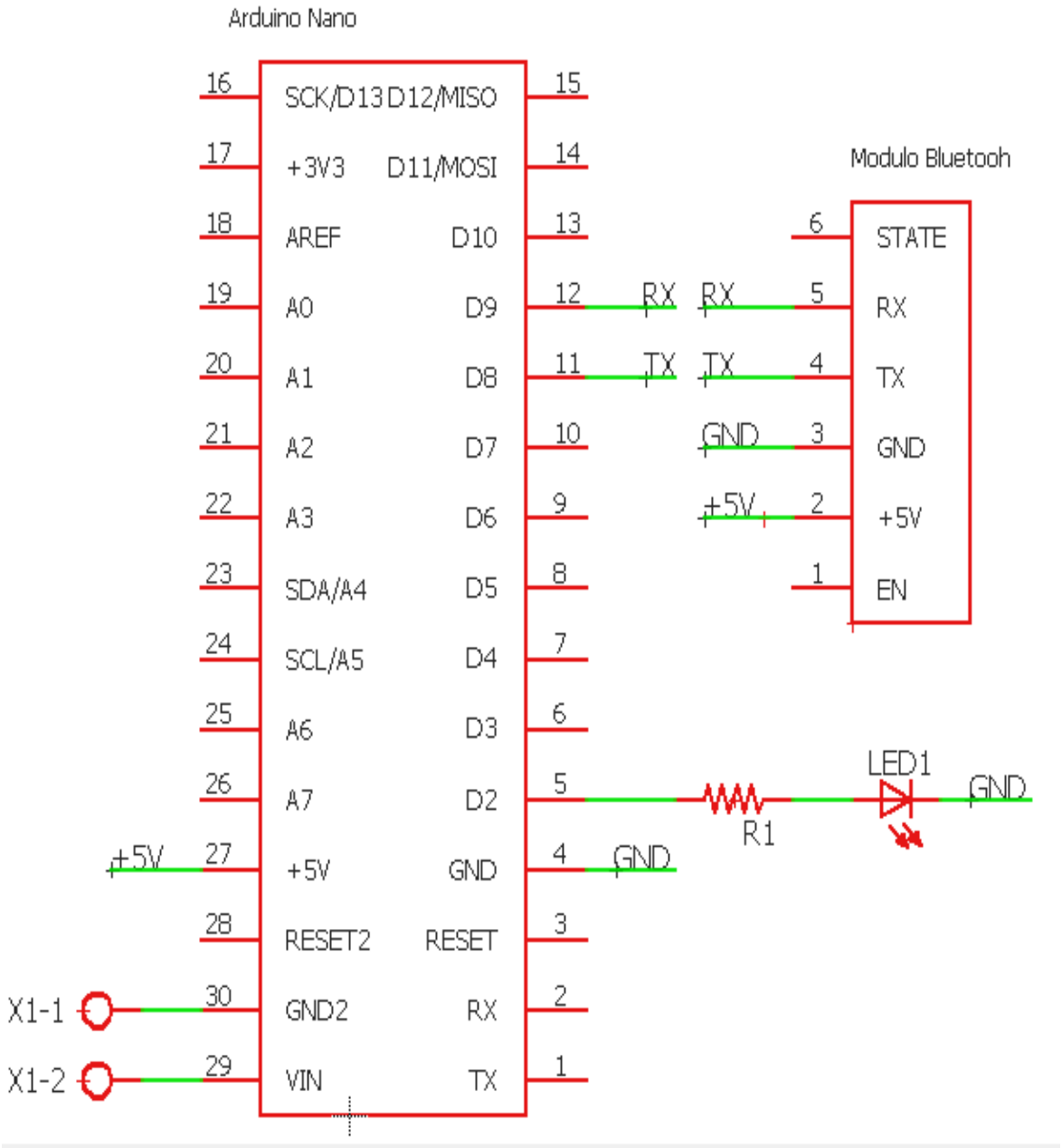
3.2.6.3 Nodo Receptor

Schematic del Nodo Receptor

En el *Schematic* del Nodo Receptor de la Figura 36 vamos a ver como a cada pin se le va asignar un nombre el cual tiene que ser igual al nombre del pin del componente electrónico al cual deseamos conectar.

El nodo Receptor está compuesto del Arduino Nano, el módulo Bluetooth HC-05, un resistor de 330 ohmios y un led.

Figura 36. Schematic del Nodo Receptor

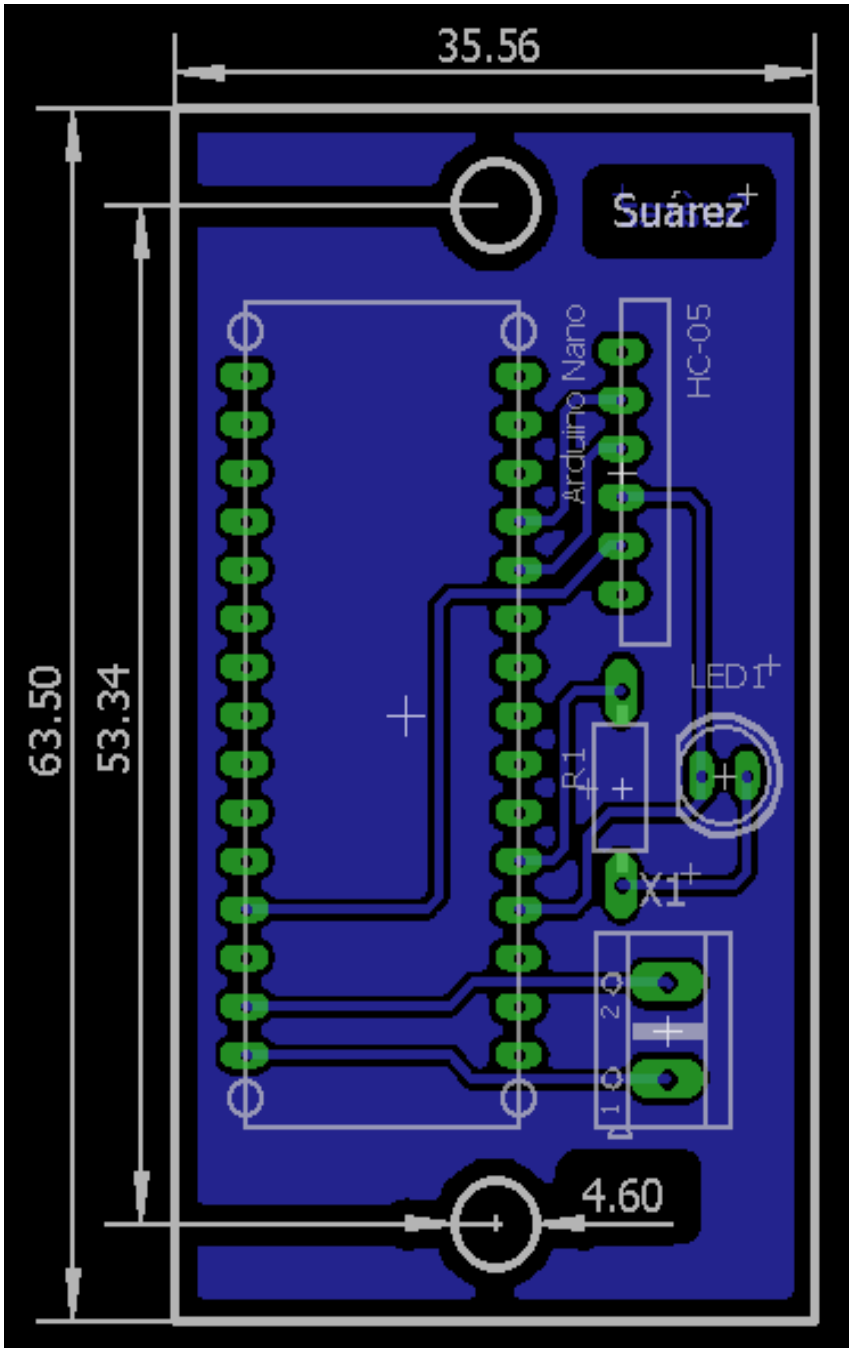


Fuente: Elaboración Propia

Board del Nodo Receptor

El board que observamos en la Figura 37 es el diseño que se realizó para la placa del Nodo Emisor que se encargará de recibir las lecturas de los Nodos Trasmisores para que puedan enviar esta información a la Raspberry Pi, podemos apreciar la distribución de los componentes electrónicos para que puedan ocupar un espacio reducido.

Figura 37. Board del Nodo Receptor



Fuente: Elaboración Propia

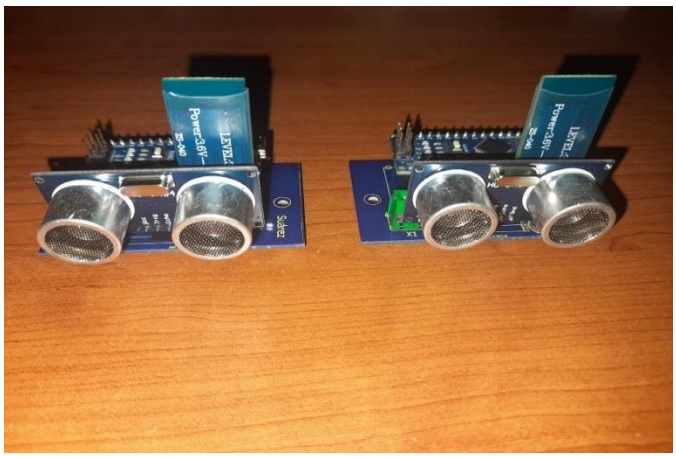
3.2.7 Implementación Física de los Nodos Electrónicos

El primer nodo es el Emisor el cual está compuesto del sensor Ultrasónico hc-sr04 que tiene como objetivo medir la distancia entre el nodo y el marco de la ventana o entre el nodo y la pared más cercana a la puerta ya que de esta forma vamos a tener unas distancias que se van a mantener de forma constante y si alguien pasa a través de la ventana o cruza a través de la puerta hace que la medición del sensor se vea afectado mandando una lectura distinta a la previamente definida pero si solo se envía la distancia al Nodo Receptor esta puede que se lea de forma incorrecta debido a que en la trasmisión inalámbrica se pierde información por tal motivo se mejoró la programación tal como se muestra en el Anexo 1 en donde si la lectura del Arduino Nano con respecto a la distancia esta fuera del rango establecido como normal se envía una señal con valor de "1" a través del Módulo Bluetooth HC-05 configurado como esclavo de lo contrario se envía el valor de "0".

La configuración de los módulos bluetooth se hace mediante comandos AT usando el monitor Serial del software Arduino, para un vínculo correcto entre el Nodo Emisor y el Nodo Receptor es necesario que ambos Módulos Bluetooth tengan la misma velocidad de transmisión para ello se usa el comando AT+UART = 9600,0,0 lo que significa que estamos configurando ambos Módulos Bluetooth a una velocidad de transmisión de 9600 baudios para que se puedan comunicar entre ellos. Para obtener la mac del Nodo Emisor debemos de usar el comando AT+UART? Con ello sabremos que la mac del Nodo Emisor colocado en la ventana es de 21:13:5e255 y en el caso del Nodo Emisor colocado en la puerta es de 98D3:31:F5F738, se debe de tener en cuenta que todos los Módulos Bluetooth que quieran comunicarse entre sí deben de tener la misma contraseña, por defecto es "1234" pero esto se puede cambiar usando el comando AT+PSWD = xxxx donde se define la contraseña de estos módulos para que se puedan conectar, si no se usa el comando AT+CMODE = 0 en nuestro módulo Bluetooth configurado como maestro puede ocasionar problemas ya que no se sabría con exactitud a que módulo conectarse puesto que dentro del ambiente pueda darse que exista otro Módulo Bluetooth que cuente con la misma contraseña y este puede conectarse a cualquiera de estos

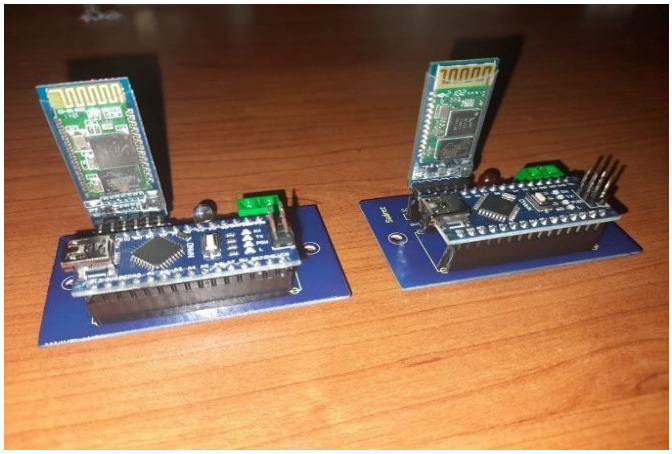
Módulos y no siempre puede ser el correcto, cabe mencionar que la velocidad de transmisión de datos debe de ser la misma para que haya una sincronización. Los componentes de los Nodos se deben de soldar teniendo mucho cuidado que no caiga estaño entre pines que no deberían de conectarse porque eso ocasionaría un mal funcionamiento de los Nodos, después de soldar cada componente como se muestra en la Figura 38 y 39 se podrá interactuar entre ellas cumpliendo sus funciones programadas en el Arduino Nano.

Figura 38. Nodos Emisores



Fuente: Elaboración Propia

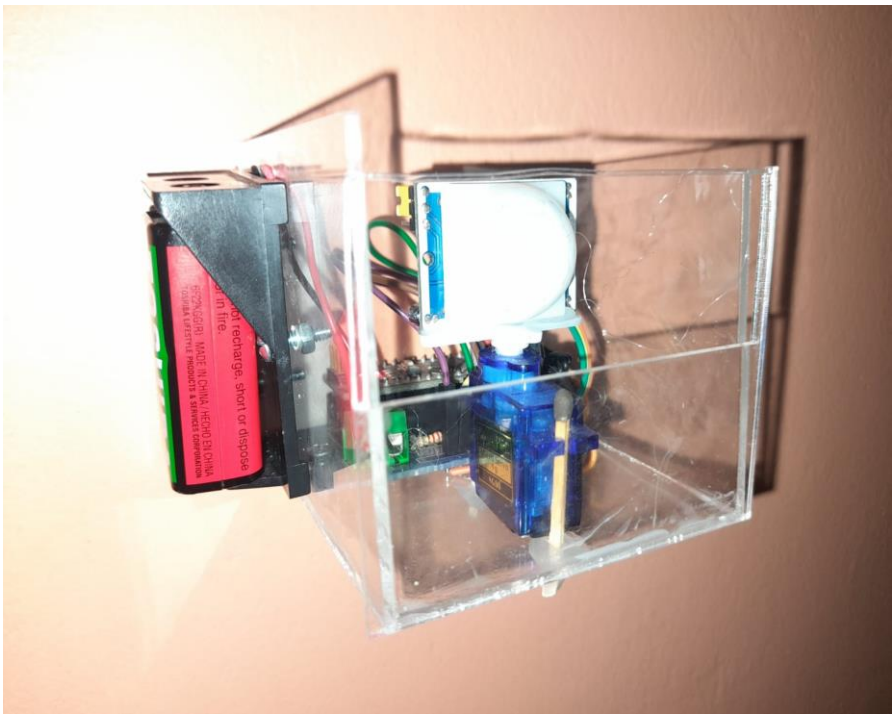
Figura 39. Nodos Receptores



Fuente: Elaboración Propia

Para un mayor entendimiento de cómo están situados los sensores dentro de la residencia se tiene la Figura 40 en donde se observa el sensor de movimiento que está montado sobre un servomotor para que de esta forma pueda sensar una mayor área al estar en movimiento cada 10° grados partiendo desde un mínimo de 0° hasta un máximo de 180°, cabe indicar que el sensor de movimiento capta dentro de un área de 120°.

Figura 40. Nodo Emisor el sensor de Movimiento



Fuente. Elaboración Propia

El nodo emisor que se ve en la Figura 41 se coloca en la ventana a pesar de que esta sea una entrada secundaria a la Residencia puede ser usado por terceros para cometer robo y en la Figura 42 el Nodo Emisor está colocado en la pared cercana a la puerta ya que esta es una entrada principal se usa frecuentemente por los dueños de la Residencia pero también puede ser usado por terceros que tengan intención de cometer robo es por ello que estas formas de ser colocados los sensores permite detectar si alguien cruza a través de estos para que se pueda registrar los cambios de distancias que implican y enviarse una señal al nodo Receptor.

Figura 41. Nodo Emisor del sensor de Distancia de la Ventana

Fuente. Elaboración Propia

Figura 42. Nodo Emisor del sensor de Distancia de la Puerta

Fuente. Elaboración Propia

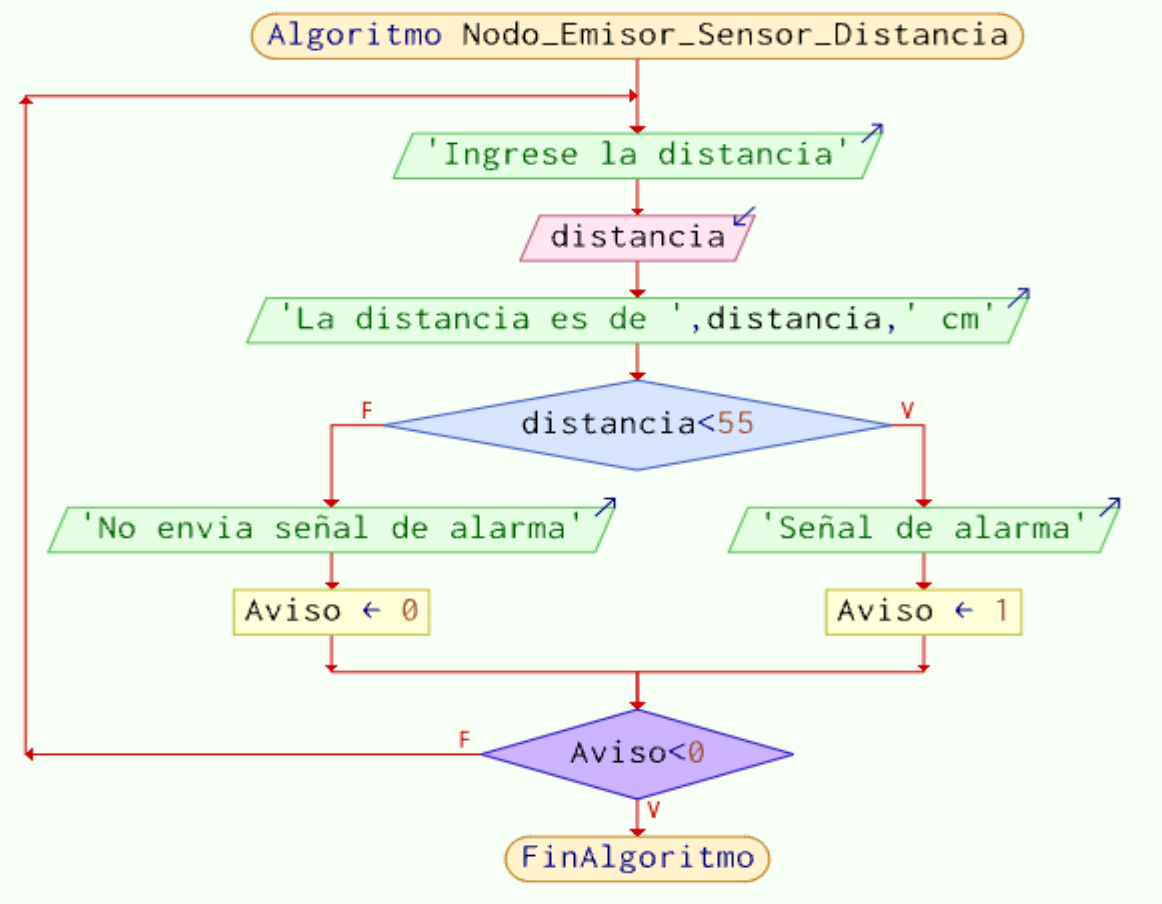
Para un mejor entendimiento de la Lógica de Programación realizada en el Anexo 1 se tiene el Pseudocódigo mostrado en la Figura 43 seguido de un diagrama de Flujo mostrado en la Figura 44.

Figura 43. Pseudocódigo del Nodo Emisor del Sensor de Distancia

```
Algoritmo Nodo_Emisor_Sensor_Distancia
    Repetir
        Escribir 'Ingrese la distancia'
        Leer distancia
        Escribir 'La distancia es de ', distancia, ' cm'
        Si distancia < 55 Entonces
            Escribir 'Señal de alarma'
            Aviso ← 1
        SiNo
            Escribir 'No envia señal de alarma'
            Aviso ← 0
        FinSi
    Hasta Que Aviso < 0
FinAlgoritmo
```

Fuente: Elaboración Propia

Figura 44. Diagrama de Flujo del Nodo Emisor del Sensor de Distancia



Fuente: Elaboración Propia

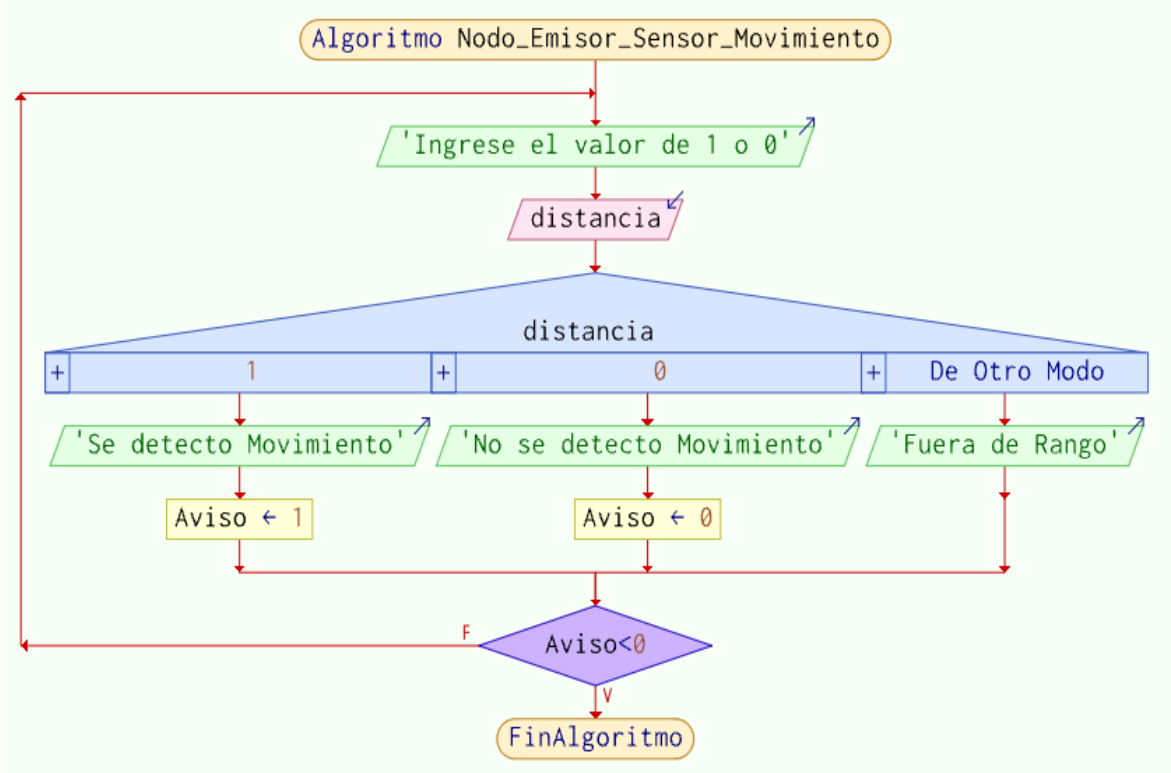
Para un mejor entendimiento de la Lógica de Programación realizada en el Anexo 2 se tiene el Pseudocódigo mostrado en la Figura 45 seguido de un diagrama de Flujo mostrado en la Figura 46.

Figura 45. Pseudocódigo del Nodo Emisor del Sensor de Movimiento

```
Algoritmo Nodo_Emisor_Sensor_Movimiento
    Repetir
        Escribir 'Ingrese el valor de 1 o 0'
        Leer distancia
        Segun distancia Hacer
            1:
                Escribir 'Se detecto Movimiento'
                Aviso = 1
            0:
                Escribir 'No se detecto Movimiento'
                Aviso = 0
            De Otro Modo:
                Escribir  'Fuera de Rango'
        Fin Segun
    Hasta Que Aviso < 0
FinAlgoritmo
```

Fuente: Elaboración Propia

Figura 46. Diagrama de Flujo del Nodo Emisor del Sensor de Movimiento



Fuente: Elaboración Propia

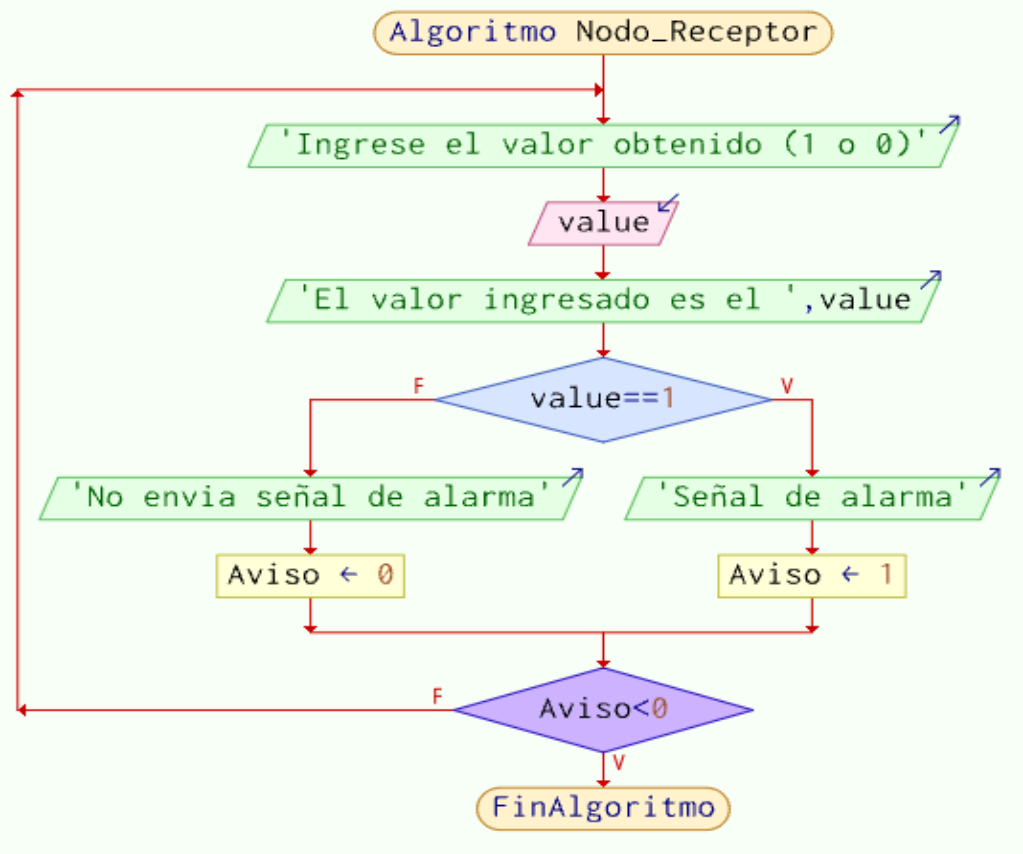
Para un mejor entendimiento de la Lógica de Programación realizada en el Anexo 3 se tiene el Pseudocódigo mostrado en la Figura 47 seguido de un diagrama de Flujo mostrado en la Figura 48.

Figura 47. Pseudocódigo del Nodo Receptor

```
Algoritmo Nodo_Receptor
    Repetir
        Escribir 'Ingrese el valor obtenido (1 o 0)'
        Leer value
        Escribir 'El valor ingresado es el ', value
        Si value == 1 Entonces
            Escribir 'Señal de alarma'
            Aviso ← 1
        SiNo
            Escribir 'No envia señal de alarma'
            Aviso ← 0
        FinSi
    Hasta Que Aviso < 0
FinAlgoritmo
```

Fuente: Elaboración Propia

Figura 48. Diagrama de Flujo del Nodo Receptor



Fuente: Elaboración Propia

3.2.8 Creación del Bot en la Aplicación Telegram

3.2.8.1 Descargar la Aplicación Telegram

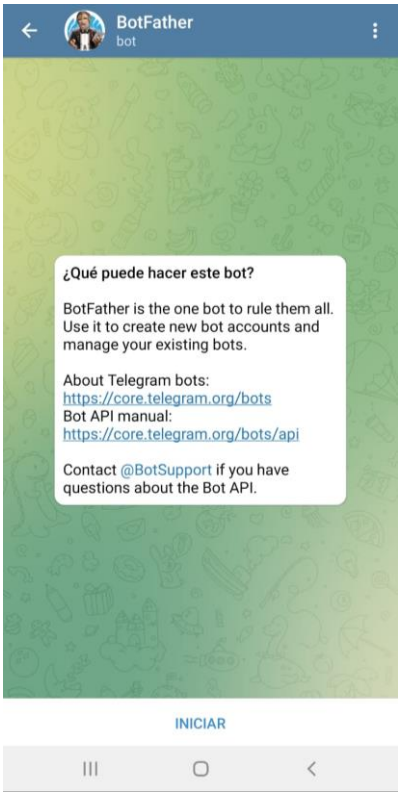
Para poder crear un Bot en la Aplicación de Telegram lo primero que debemos de realizar es descargar la Aplicación desde *Play Store*.

Cuando ya se tiene la Aplicación de Telegram en nuestro dispositivo móvil vamos a poder registrarnos ingresando un código de verificación que nos mandarán al momento de registrar nuestro número telefónico.

3.2.8.2 Creación del Bot en Telegram

Cuando ya nos hemos registrado en Telegram y obtenido una cuenta vamos a dirigirnos al buscador que tiene la aplicación y colocaremos la palabra @BotFather en donde debemos de seleccionar el icono de @BotFather para que se nos apertura la Figura 49.

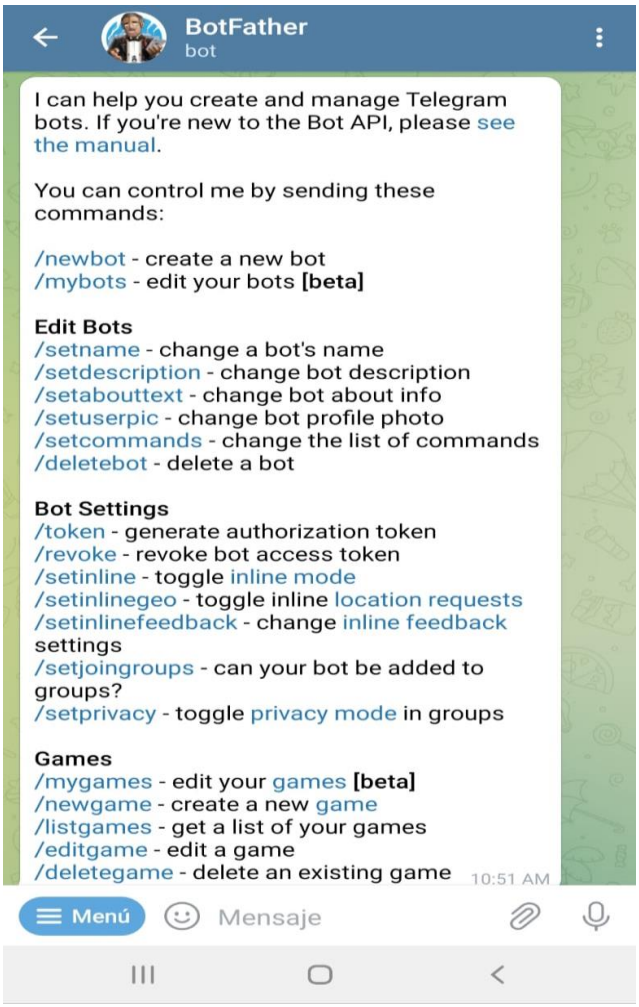
Figura 49. Chat con BotFather



Fuente: Elaboración Propia

Al momento de darle click a la opción “INICIAR” se abrirá una nueva pantalla como en la Figura 50 en donde se muestra los comandos que son reconocidos por el @BotFather.

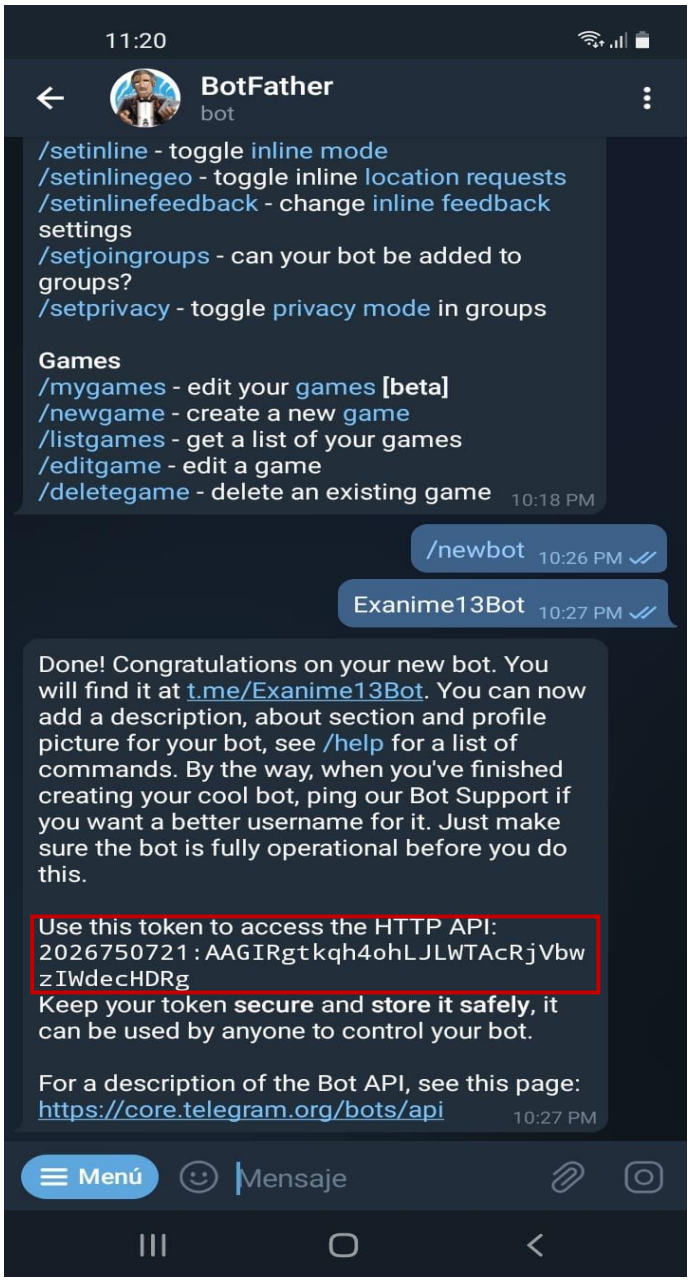
Figura 50. Comandos reconocidos por BotFather



Fuente: Elaboración Propia

Luego de obtener la comunicación con BotFather vamos poder enviarle comandos de acuerdo a lo que requerimos, en este caso le enviaremos el comando de /newbot para poder crear un nuevo bot dentro de nuestra aplicación de Telegram para ello debemos de asignarle un nombre el cual no esté en uso, cuando creemos a nuestro bot podremos obtener el token que se muestra en la Figura 51 que es único para que podamos establecer una comunicación enviándole mensajes desde Python.

Figura 51. Creando Bot de Telegram



Fuente: Elaboración Propia

3.2.9 Reconocimiento Facial

Para poder realizar el Reconocimiento Facial se establecen pasos a seguir los cuales mencionaremos a continuación.

3.2.9.1 Encender la Cámara

Para que la Raspberry Pi 3 pueda reconocer la cámara que se está insertando en su puerto CSI debemos de activar la cámara ingresando al CMD de la Raspberry Pi 3 y colocar el comando sudo raspi-config, activar la cámara y reiniciar la Raspberry Pi 3 para que se aplique la configuración.

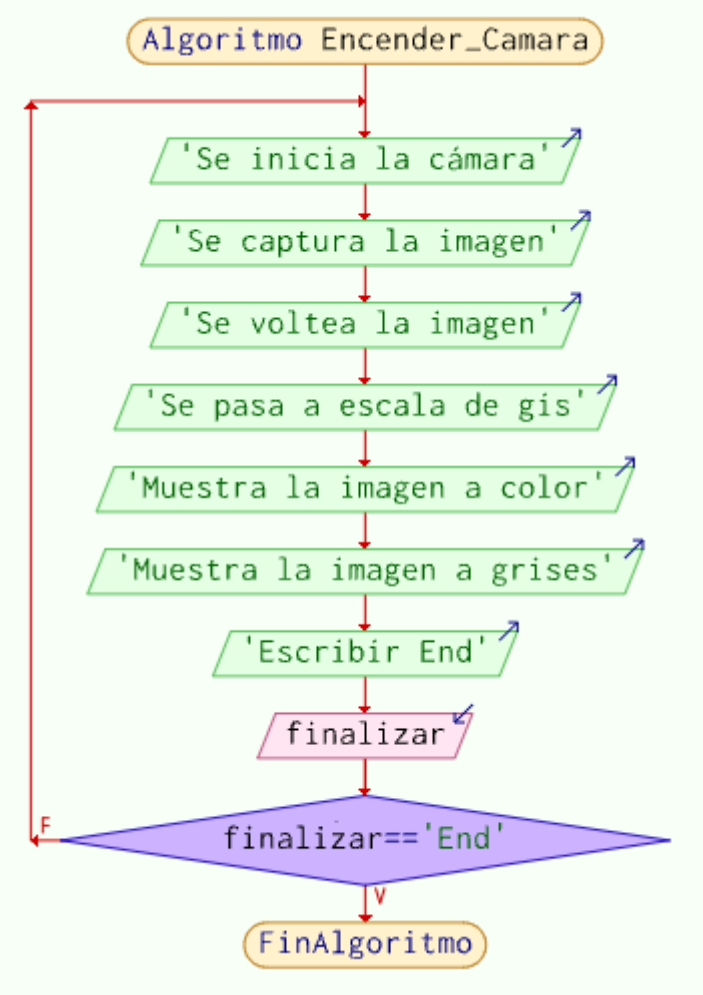
Se cuenta con una cámara de 5 MP que trabaja a 30 fps la cual para poder encenderla vamos ejecutar el Script “Encender_Camara.py” del Anexo 7 donde se describe los pasos a seguir para su funcionamiento, asimismo se tiene el pseudocódigo de la Figura 52 y el diagrama de Flujo de la Figura 53 para un mayor entendimiento de cómo se lleva a cabo el procedimiento.

Figura 52. Pseudocódigo de Encender Cámara

```
Algoritmo Encender_Camara
    Repetir
        Escribir 'Se inicia la cámara'
        Escribir 'Se captura la imagen'
        Escribir 'Se voltea la imagen'
        Escribir 'Se pasa a escala de gis'
        Escribir 'Muestra la imagen a color'
        Escribir 'Muestra la imagen a grises'
        Escribir 'Escribir End'
        Leer finalizar
    Hasta Que finalizar == 'End'
FinAlgoritmo
```

Fuente: Elaboración Propia

Figura 53. Diagrama de Flujo de Encender Cámara



Fuente: Elaboración Propia

3.2.9.2 Detectar Rostro

El siguiente paso para poder avanzar con el sistema de reconocimiento facial es preparar a la Raspberry Pi 3 para que sea capaz de reconocer los rasgos faciales de una persona para ello se usó las librerías de Cascades en la cual se encuentra el siguiente clasificador de rostros frontales (haarcascade_frontalface_default.xml) esto hará que se encierre dentro de un marco verde el rostro de una persona si en caso la cámara enfocase a una persona, el desarrollo de esta programación se realizó en Python, para un mejor entendimiento de la Lógica de Programación realizada en el Anexo 8 se tiene el Pseudocódigo mostrado en la Figura 54 seguido de un diagrama de Flujo mostrado en la Figura 55.

Figura 54. Pseudocódigo de Detectar Rostro

```
Algoritmo Detectar_Rostro
    Repetir
        Escribir 'Se inicia el clasificador de objetos'
        Escribir 'Se inicia la cámara'
        Escribir 'Se captura la imagen'
        Escribir 'Se voltea la imagen'
        Escribir 'Se pasa a escala de gis'
        Escribir 'Se detecta el rostro'
        Escribir 'Muestra la imagen a color'
        Escribir 'Escribir End'
        Leer finalizar
    Hasta Que finalizar == 'End'
FinAlgoritmo
```

Fuente: Elaboración Propia

Figura 55. Diagrama de Flujo de Detectar Rostro

Fuente: Elaboración Propia

3.2.9.3 Base de Datos.

Es de vital importancia tener una Base de Datos en nuestro Sistema de Reconocimiento Facial porque de esa forma se va poder comparar las semejanzas del rostro que la cámara esté enfocando con los rasgos faciales que se tiene almacenado dentro de la Raspberry Pi 3.

Nuestra Base de Datos cuenta con un registro de 500 imágenes por usuario. Al momento de realizar la Base de Datos los usuarios que estén frente a la cámara deberían de realizar movimientos suaves girando la cabeza de izquierda hacia derecha, de derecha hacia izquierda, de arriba hacia abajo, de abajo hacia arriba, mirando de frente, cerrando los ojos, abriendo los ojos, hablando, sonriendo para que de esta forma se pueda tener una gran variedad de imágenes con las que se puedan comparar y realizar el proceso de Reconocimiento Facial ya que cada persona es distinta.

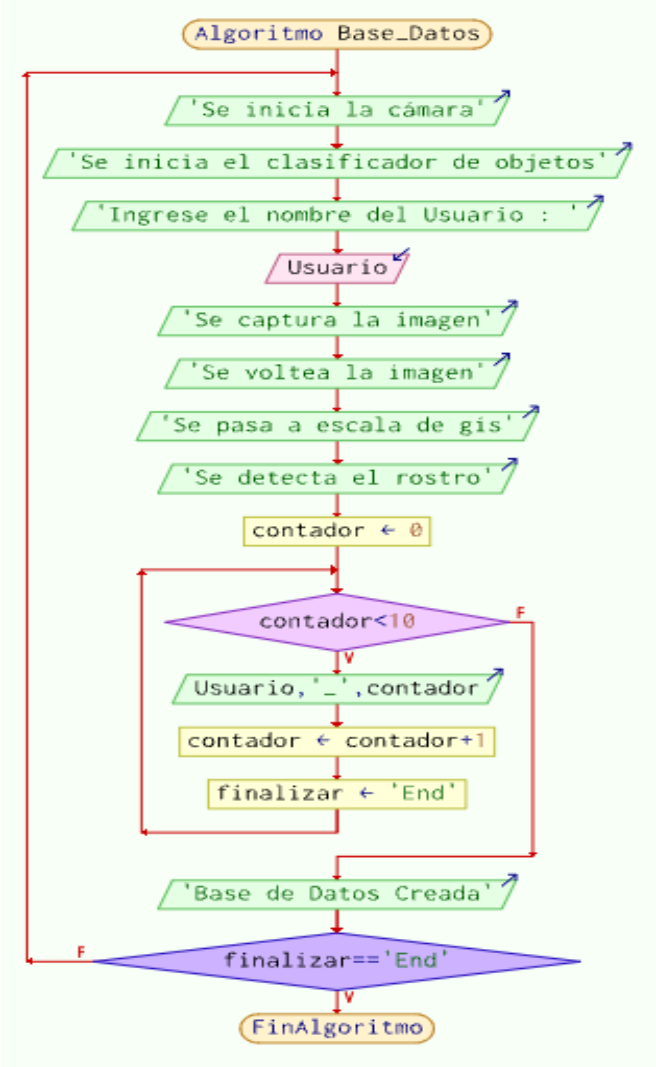
Para un mejor entendimiento de la Lógica de Programación realizada en el Anexo 9 se tiene el Pseudocódigo mostrado en la Figura 56 seguido de un diagrama de Flujo mostrado en la Figura 57.

Figura 56. Pseudocódigo de la Base de Datos

```
Algoritmo Base_Datos
    Repetir
        Escribir 'Se inicia la cámara'
        Escribir 'Se inicia el clasificador de objetos'
        Escribir 'Ingrese el nombre del Usuario : '
        Leer Usuario
        Escribir 'Se captura la imagen'
        Escribir 'Se voltea la imagen'
        Escribir 'Se pasa a escala de gis'
        Escribir 'Se detecta el rostro'
        contador ← 0
        Mientras contador < 10 Hacer
            Escribir Usuario, '_', contador
            contador ← contador + 1
            finalizar ← 'End'
        Fin Mientras
        Escribir 'Base de Datos Creada'
    Hasta Que finalizar == 'End'
FinAlgoritmo
```

Fuente: Elaboración Propia

Figura 57. Diagrama de Flujo de la Base de Datos



Fuente: Elaboración Propia

3.2.9.4 Entrenamiento de la Base de Datos

El entrenamiento de la Base de Datos se hace recorriendo cada una de las imágenes almacenadas dentro de las subcarpetas con nombre de los usuarios de la Residencia que se encuentran dentro de la carpeta de nombre BaseDatos, se recorren las 1000 imágenes de formato JPG.

Para el entrenamiento de la Base de Datos se tiene que pasar todas las imágenes almacenadas a escala de grises, luego para evitar el solapamiento de las imágenes se usa el parámetro ANTIALIAS para evitar la distorsión de las imágenes, luego se van a crear matrices de 8x8 que tienen valores que van de 0 a 255, se hace uso del Algoritmo LBPH mencionado en las Bases

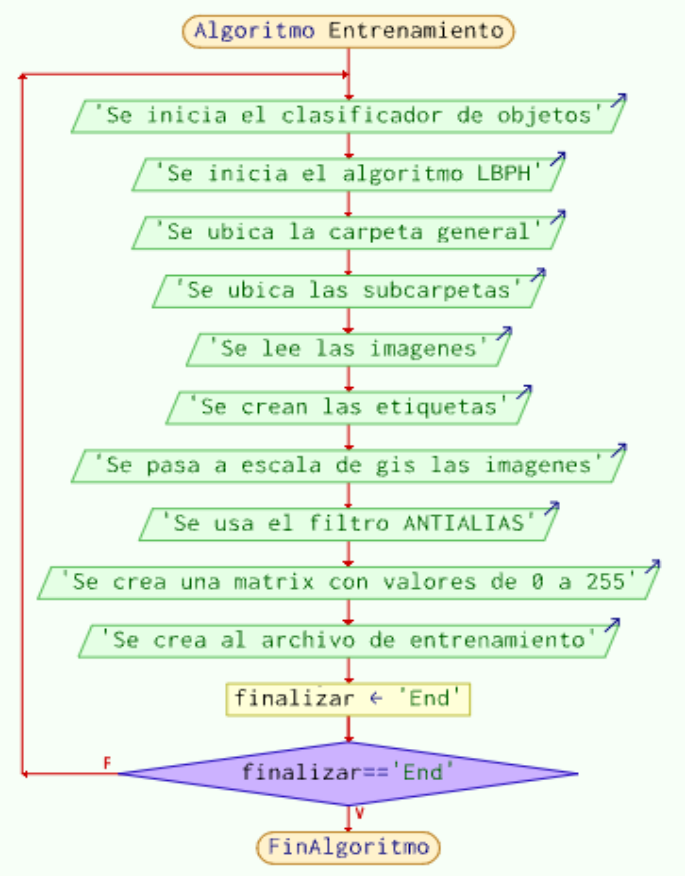
Teóricas. Para un mejor entendimiento de la Lógica de Programación realizada en el Anexo 10 se tiene el Pseudocódigo mostrado en la Figura 58 seguido de un diagrama de Flujo mostrado en la Figura 59.

Figura 58. Pseudocódigo de la Base de Datos

```
Algoritmo Entrenamiento
    Repetir
        Escribir 'Se inicia el clasificador de objetos'
        Escribir 'Se inicia el algoritmo LBPH'
        Escribir 'Se ubica la carpeta general'
        Escribir 'Se ubica las subcarpetas'
        Escribir 'Se lee las imagenes'
        Escribir 'Se crean las etiquetas'
        Escribir 'Se pasa a escala de gis las imagenes'
        Escribir 'Se usa el filtro ANTIALIAS'
        Escribir 'Se crea una matrix con valores de 0 a 255'
        Escribir 'Se crea al archivo de entrenamiento'
        finalizar ← 'End'
    Hasta Que finalizar == 'End'
FinAlgoritmo
```

Fuente: Elaboración Propia

Figura 59. Diagrama de Flujo del Entrenamiento de la Base de Datos

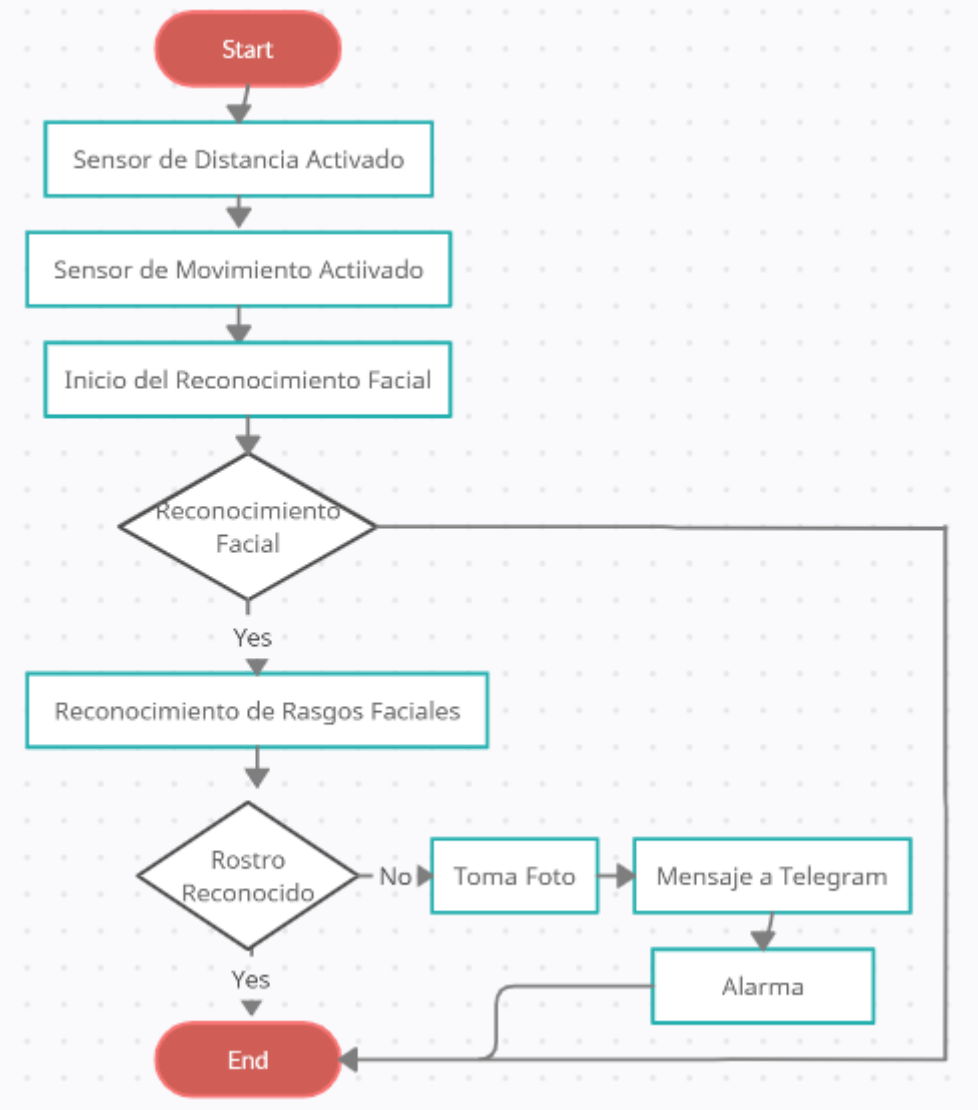


Fuente: Elaboración Propia

3.2.9.5 Reconocimiento Facial

Después de haber realizado los pasos ya mencionados nos queda realizar el proceso de Reconocimiento Facial que consiste en comparar las características del rostro enfocado por la cámara con las características que se obtiene de los parámetros del entrenamiento de la base de datos

Figura 60. Diagrama de Flujo del Proceso de Reconocimiento Facial



Fuente: Elaboración Propia

3.2.10 Metodología de la Red BackPropagation

Para poder obtener los pesos y bias que se requieren para implementar la Red Neuronal en el Arduino Uno se desarrolló una programación tal como se muestra en el Anexo 13, esta programación está basada en el Algoritmo BackPropagation explicado dentro de las Bases Teóricas, es por ello que se necesita proporcionar las entradas que se van a obtener de los Nodos Emisores como también debemos de proporcionar las salidas esperadas que hacen que se encienda la alarma en caso de una intrusión, dentro de la programación se define la cantidad de neuronas de la primera y segunda capa como también la función de activación para cada una de estas. Luego se pasa a iniciar nuestra red neuronal para luego establecer las épocas en la cuales se tiene que dar nuestro entrenamiento para obtener resultados más precisos, también se define el error del entrenamiento, luego se pasa a realizar una simulación de la red neuronal pasándole como parámetro nuestras entradas de esta manera obtendremos los pesos y bias tanto de la primera capa como de la segunda capa, se usó la función celldisp para poder visualizar los valores de las matrices que nos botan al momento de realizar la simulación de la Red Neuronal.

3.2.11 Implementación Física de la Cámara

Los Nodos receptores van a cumplir la función de enviar la información obtenida por parte de los Nodos Emisores a la Raspberry Pi, esta información obtenida sirve de estímulo para que se ejecute o no el Reconocimiento Facial en la cámara, además también sirven de entrada para la Red Neuronal como es el caso del Nodo Receptor del Sensor de Movimiento. En la Figura 58 se observa la implementación física de la cámara con los Nodos Receptores tal como se presentó en la simulación de la Figura 31.

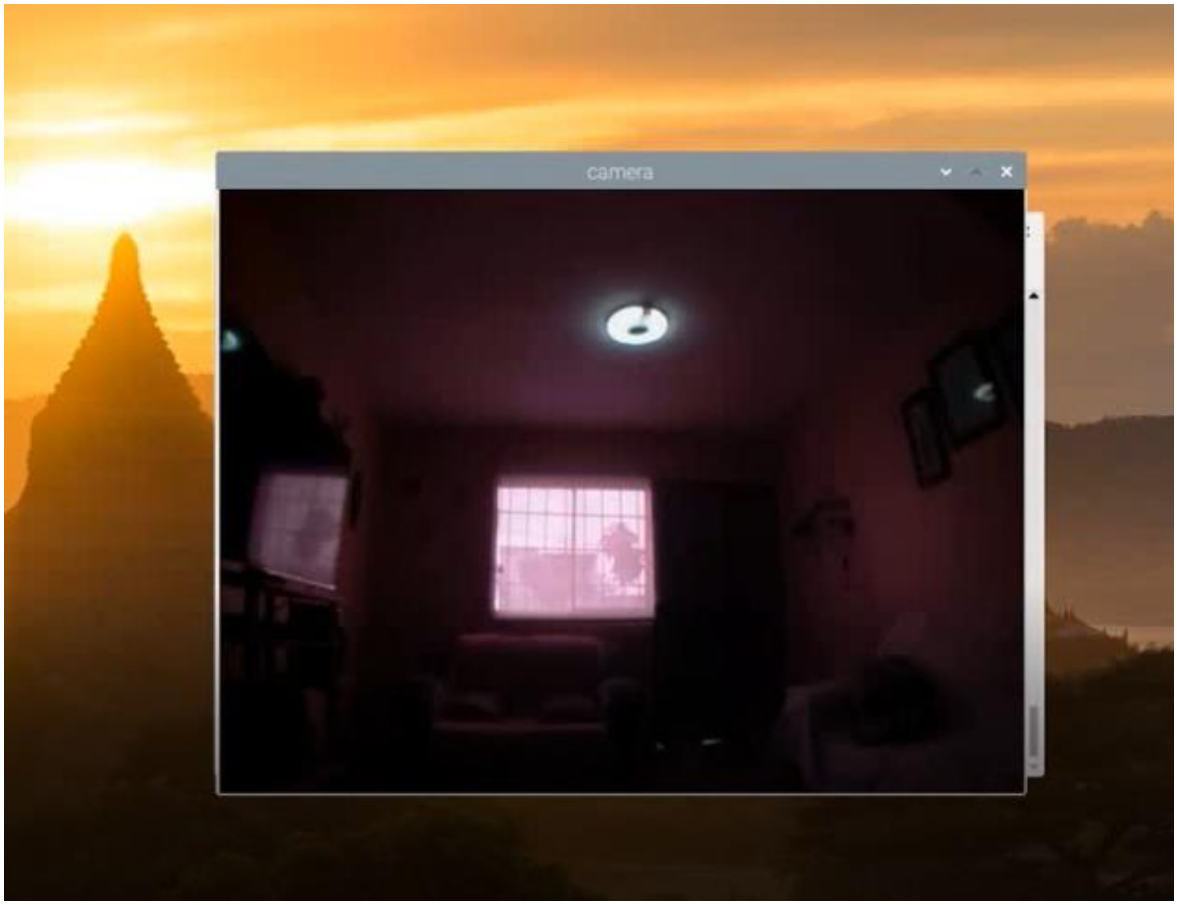
Figura 61. Nodos Receptores con la cámara de la Raspberry Pi

Fuente: Elaboración Propia

3.2.12 Conexión VNC

La conexión VNC que se presenta consiste en poder conectarnos de forma remota a nuestro Raspberry Pi para que de esta forma podamos observar lo que nuestra cámara nos muestre para ello necesitamos la ip de nuestra Raspberry Pi la cual obtendremos al momento de ejecutar el comando ifconfig dentro de la terminal de Raspbian y una vez obtenido esa ip vamos a poder configurar nuestro programa de VNC ya sea instalado en nuestra Laptop o en nuestro dispositivo móvil en donde al momento de conectarnos nos van a pedir nuestra ip y contraseña la cual es raspbian si en caso no se le ha cambiado al momento de instalar el S.O de Raspbian y de esta forma obtendremos la conexión tal como se muestra en la Figura 62.

Figura 62. Conexión a VNC



Fuente: Elaboración Propia

3.2.13 Presupuesto del Sistema de Alerta

La Tabla 3 nos da a conocer el presupuesto que se utilizó para poder llevar a cabo el Sistema de Alerta.

Tabla 3. Precio de los materiales a usar

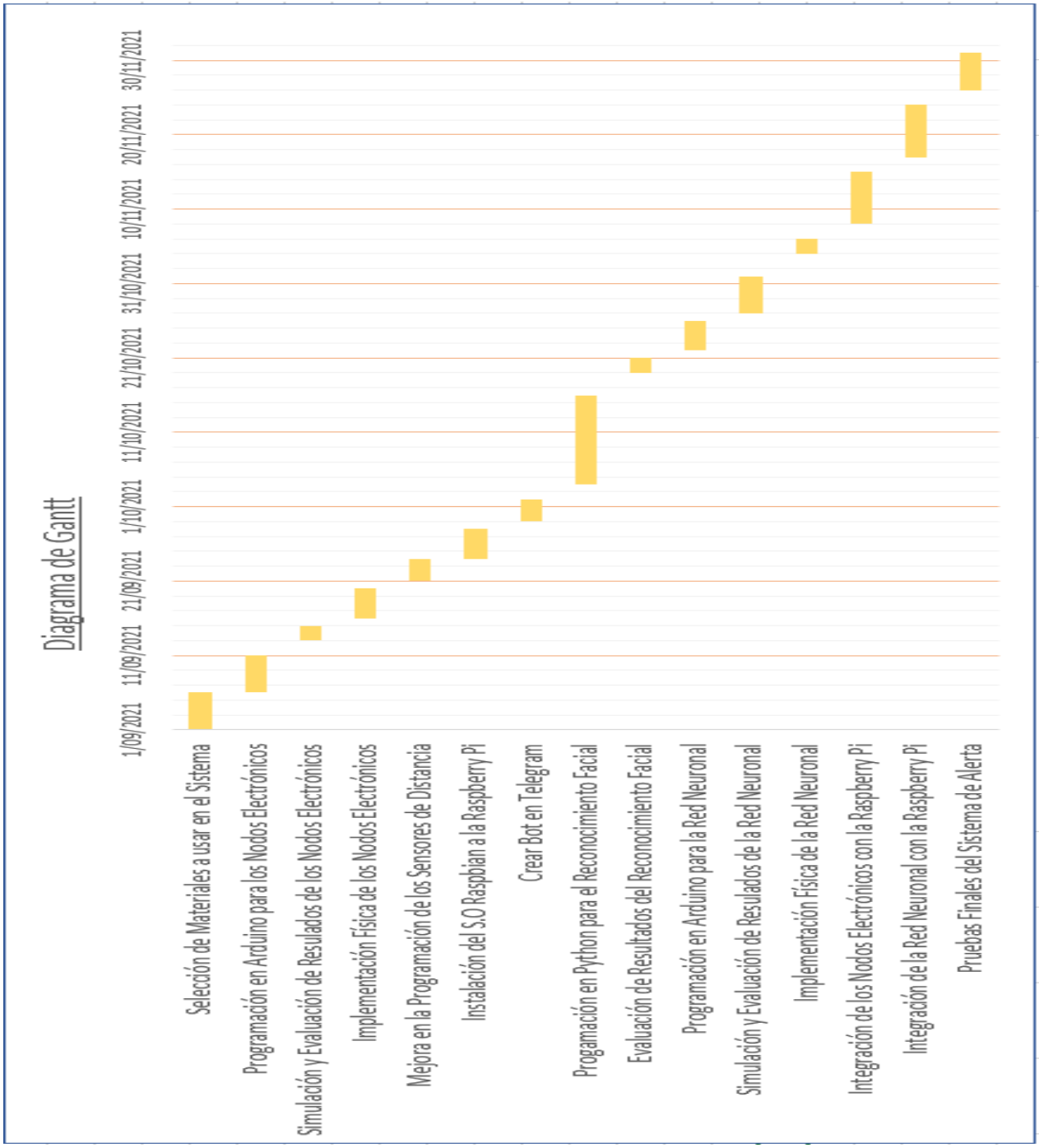
Materiales	Precio unitario	Cantidad
Raspberry Pi	S/. 277.00	1
Cámara 5Mp	S/. 130.00	1
Memoria MicroSD	S/. 99.95	1
Lector MicroSD	S/. 10.00	1
Case de Raspberry Pi	S/. 22.00	1
Arduino Nano	S/. 28.00	6
Sensor Ultrasónico	S/. 7.50	2
Modulo Bluetooth HC-05	S/. 24.50	6
Sensor PIR	S/. 10.00	1
Servomotor SG90S	S/. 12.00	1
Resistores de 330 ohms ½ Watss	S/. 1.00	100
Baterías 9V DC	S/. 3.00	7
Porta baterías	S/. 1.80	7
Led Rojo	S/. 0.25	6
Bornera 2 pines	S/. 0.50	6
Buzzer Zubador	S/. 1.30	1
Cable Macho-Macho de 40 hilos	S/. 5.00	1
Cable Macho-Hembra de 40 hilos	S/. 5.00	1
Protoboard 170 puntos	S/. 3.50	2
Protoboard 830 puntos	S/. 14.00	2
Multímetro UT33C+	S/. 69.00	1
Cautín	S/. 25.00	1
Supresor de pico	S/. 15.00	1
Estaño	S/. 15.00	1
Acrílico	S/. 25.00	1
1 litro de Cloroformo	S/. 15.00	1
Navaja para Acrílico	S/. 10.00	1
Discos de Matlab	S/. 6.00	3
Precio Total	S/. 1156.35	

Fuente: Elaboración Propia

3.2.14 Diagrama de Gantt

En la Figura 63 se puede observar las fechas de inicio de cada una de las actividades que se realizaron para poder llevar a cabo el Sistema de Alerta, así como también se puede observar el tiempo de duración de cada una de estas actividades.

Figura 63. Diagrama de Gantt del Desarrollo del Sistema de Alerta



Fuente: Elaboración Propia

3.3. Resultados

3.3.1 Resultados de Selección de los componentes para el reconocimiento facial que interactúe con el Sistema de Alerta.

Existen varios componentes electrónicos que cumplen la misma función, pero debemos de saber distinguir cuales de ellos se adecuan mejor a nuestro sistema es por ello que se realizó una clasificación de los materiales que requerimos para poder seleccionarlos en base a sus características más resaltantes.

Elección de la Cámara a usar:

En la Tabla 4 se comparan las resoluciones de las cámaras CCTV existentes en el mercado y como se definió se usó el módulo de la Cámara de 5 MP tal como se muestra en la Figura 64.

Tabla 4. Comparación de las resoluciones de las cámaras de CCTV

Cámara	Resolución
Cámara Web	640 x 480 pixeles
STV-Cámara Domo ST-D120HIB	720 pixeles
Dahua-Cámara Domo HAC-T1A11	720 pixeles
Hikvision-Cámara Domo Turbo	720 pixeles
Módulo de Cámara para Raspberry	5 M pixeles
Módulo de Cámara para Raspberry	8 M pixeles

Fuente: Elaboración Propia

Figura 64. Cámara de 5MP con visión nocturna

Fuente: Elaboración Propia

Elección del Sensor de Distancia

Veremos 3 diferentes sensores que se usan comúnmente para la medición de la distancia en cm los cuales presentaremos en la Tabla 5.

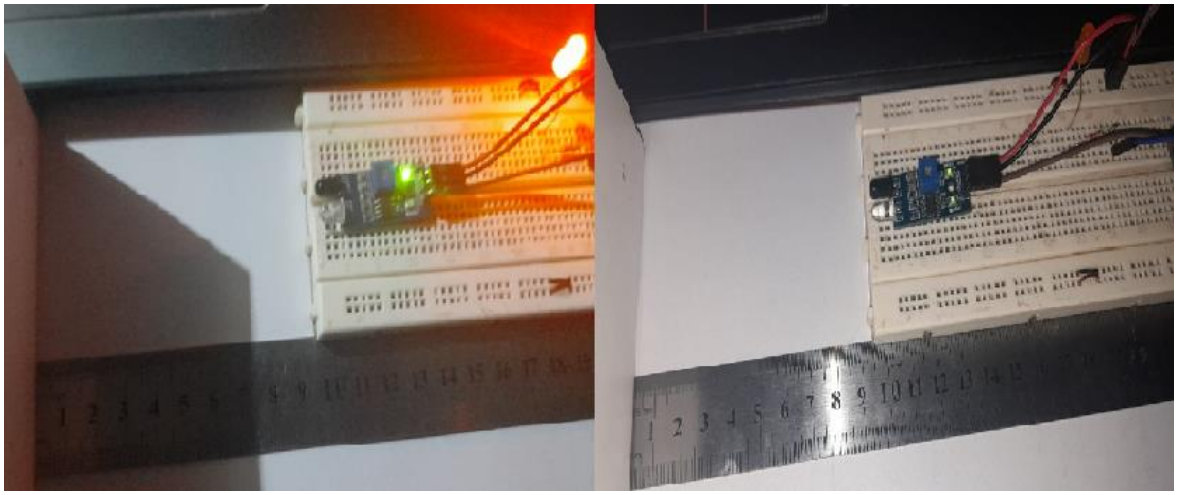
Tabla 5. Clasificación de los Sensores de Distancia

Sensor	Modelo	Rango
Infrarrojo	FC-51	50cm
Ultrasonido	HC-sr04	2 – 450 cm

Fuente: Elaboración Propia

a) **Sensor Infrarrojo:** La distancia que puede percibir el led Infrarrojo es muy reducida tal como se muestra en la Figura 66 en donde se percibe solo una distancia de 9.5 cm de detección la cual no dispone de los requerimientos básicos que se necesita para el Sistema de Alerta.

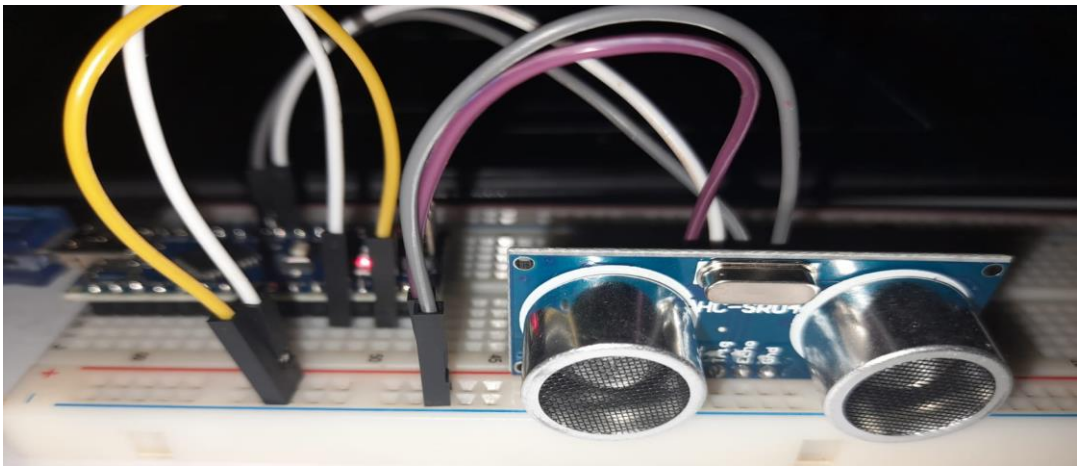
Figura 66. Implementación del Sensor Infrarrojo



Fuente: Elaboración Propia

b) **Sensor Ultrasonido:** El sensor Ultrasónico tiene un alcance hasta de 450 cm lo cual realizando una prueba logramos obtener que nos será muy útil al momento de realizar la implementación de estos en las ventanas o puertas, este sensor no se ve afectado por la luz solar ni por los materiales de color negro.

Figura 67. Implementación del Sensor Ultrasonido



Fuente: Elaboración Propia

Elección del Arduino para los Nodos Emisores y Receptores

La tabla 6 nos da a conocer los distintos Arduinos existentes en el mercado

Tabla 6. Comparaciones entre los Modelos de Arduino

Nombre	Procesador	Voltaje	Velocidad	E/S Analógica	E/S Digital
Uno	ATmega328P	5 - 12 V DC	16 MHZ	6 / 0	14 / 6
Leonardo	ATmega32U4	5 - 12 V DC	16 MHZ	12 / 0	20 / 7
101	Intel Core	3.3 - 12 V DC	32 MHZ	6 / 0	14 / 4
Esplora	ATmega32U4	5 - 12 V DC	16 MHZ		
Arduino Zero	ATSAMD21G18	3.3 - 12 V DC	48 MHZ	6 / 1	14 / 10
Mega 2560	ATmega2560	5 - 12 V DC	16 MHZ	16 / 0	54 / 15
Arduino Nano	ATmega328	5 V DC	16 MHZ	8 / 6	14

Fuente: Elaboración Propia

En la Tabla 7 vemos las características de los diferentes modelos de Raspberry que existen, vemos como en cada versión va mejorando cada componente con la cual está compuesto un Raspberry y de entre todas las características vemos que el Raspberry 3 modelo 3 tiene una RAM de 1Gb lo cual nos será adecuado para ejecutar todas las instrucciones que le serán enviadas y será más veloz al momento de ejecutar procesos con respecto a las anteriores versiones.

Tabla 7. Comparaciones entre los Modelos de Raspberry Pi

Raspberry Pi	Modelo A	Modelo A+	Modelo B
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835
CPU	700 MHz ARM1176JFZ-S	700 MHz ARM1176JFZ-S	700 MHz ARM1176JFZ-S
GPU	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256Mb	512Mb	512Mb
USB	1	1	2
Video	RCA, HDMI	Jack, HDMI	RCA, HDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI
Boot	Memoria SD	Memoria micro SD	Memoria SD
Wireless	No tiene	No tene	No tiene
Red Ethernet	No tiene	No tiene	No tiene
Alimentación	5V / 2Amp	5V / 2Amp	5V / 2Amp
GPIO	26 pines GPIO	40 pines GPIO	26 pines GPIO
Tamaño	85.6 x 53.98 mm	65 x 56 mm	85.6 x 53.98 mm

Raspberry Pi	Modelo B +	RPi V2 modelo B	Rpi 3 modelo B
SoC	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2836
CPU	700 MHz ARM1176JFZ-S	900 MHz Quad-core ARM Cortex-A7	1.2 GHz Quad Cortex A53
GPU	VideoCore IV	250 MHz VideoCore IV	400 MHz VideoCore IV
RAM	512Mb	1Gb	1Gb
USB	4	4	4
Video	Jack, HDMI	Jack, HDMI	Jack, HDMI
Audio	Jack, HDMI	Jack,	Jack, HDMI
Boot	Memoria microSD	Memoria microSD	Memoria microSD
Wireless	No tiene	No tiene	802.11n / Bluetooth 4.1
Red Ethernet	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100
Alimentación	5V / 2Amp	5V / 2Amp	5V / 2.5 Amp
GPIO	40 pines GPIO	40 pines GPIO	40 pines GPIO
Tamaño	85 x 53 x 17 mm	85 x 56 x 17 mm	85 x 56 x 17 mm

Fuente: Elaboración Propia

3.3.2 Resultados del método para lectura de los sensores

Se probó dos formas distintas para poder enviar la información desde el Nodo Emisor hacia al Nodo Receptor en donde la primera de ellas consistía en realizar el cálculo de la distancia en el Nodo Receptor por lo que se enviaba el resultado de la duración de la función PulseIn programada en el Arduino Nano del Nodo Emisor y mediante la siguiente ecuación se realizaba el cálculo de la distancia.

$$Distancia = (duración)/58.4$$

En estos resultados vemos que la distancia obtenida no es correcta por lo que no guarda relación con las medidas físicas y es debido a que en el proceso de transmisión se perdió información.

Ya que el problema radicó en la transmisión de la información por la cantidad de bits vimos la necesidad de cambiar la programación habitual y realizar los cálculos en el Nodo Emisor en donde también se aplicó la fórmula antes mencionada pero no se transmitiría la distancia sino que el valor de “0” o “1” dependiendo las condiciones que se programó en el Arduino Nano de esta manera al solo enviar un bit no habría perdida de información en el Nodo Receptor el cual al momento de recibir estos valores sabrá en qué momento mandar la señal a la Raspberry Pi 3.

Sabemos que el Sensor Ultrasónico puede medir hasta una distancia de 450 cm por lo que diferentes tipos de fabricantes recomiendan usar ciertas fórmulas para poder obtener las medidas exactas al momento de usar este sensor, las fórmulas son las siguientes.

$$Fórmula\ 1 = (duración * 0.034)/2$$

$$Fórmula\ 2 = (duración)/58.4$$

Al momento de realizar nuestras mediciones para poder saber cuál de estas fórmulas usaremos obtuvimos las siguientes tablas.

Tabla 8. Distancias obtenidas con el Sensor Ultrasónico HC-sr04 usando la fórmula 1

Distancias	Valor Medido	Valor Exacto	Error absoluto	Error relativo
Distancia de 200 cm	196-197 cm	200 cm	4 cm	2%
Distancia de 190 cm	186-187 cm	190 cm	4 cm	2.11%
Distancia de 180 cm	177 cm	180 cm	3 cm	1.67%
Distancia de 170 cm	167 cm	170 cm	3 cm	1.76%
Distancia de 160 cm	158-159 cm	160 cm	2 cm	1.25%
Distancia de 150 cm	147-148 cm	150 cm	3 cm	2%
Distancia de 140 cm	137-138 cm	140 cm	3 cm	2.14%
Distancia de 130 cm	127-128 cm	130 cm	3 cm	2.31%
Distancia de 120 cm	118 cm	120 cm	2 cm	1.67%
Distancia de 110 cm	108-109 cm	110 cm	2 cm	1.82%
Distancia de 100 cm	98-99 cm	100 cm	2 cm	2%
Distancia de 90 cm	88 cm	90 cm	2 cm	2.22%
Distancia de 80 cm	78-79 cm	80 cm	2 cm	2.5%
Distancia de 70 cm	69 cm	70 cm	2 cm	2.86%
Distancia de 60 cm	59 cm	60 cm	1 cm	1.67%
Distancia de 50 cm	49-50 cm	50 cm	1 cm	2%
Distancia de 40 cm	39-40 cm	40 cm	1 cm	2.5%
Distancia de 30 cm	30 cm	30 cm	0 cm	0%
Distancia de 20 cm	20 cm	20 cm	0 cm	0%
Distancia de 10 cm	10 cm	10 cm	0 cm	0%

Fuente: Elaboración Propia

Tabla 9. Distancias obtenidas con el Sensor Ultrasónico HC-sr04 usando la fórmula 2

Distancias	Valor Medido	Valor Exacto	Error Absoluto	Error Relativo
Distancia de 200 cm	196-197 cm	200 cm	4 cm	2%
Distancia de 190 cm	186-187 cm	190 cm	4 cm	2.11%
Distancia de 180 cm	177 cm	180 cm	3 cm	1.67%
Distancia de 170 cm	167 cm	170 cm	3 cm	1.76%
Distancia de 160 cm	156 cm	160 cm	4 cm	2.5%
Distancia de 150 cm	147 cm	150 cm	3 cm	2%
Distancia de 140 cm	137 cm	140 cm	3 cm	2.14%
Distancia de 130 cm	129-130 cm	130 cm	1 cm	0.77%
Distancia de 120 cm	118-119 cm	120 cm	2 cm	1.67%
Distancia de 110 cm	108 cm	110 cm	2 cm	1.82%
Distancia de 100 cm	98 cm	100 cm	2 cm	2%
Distancia de 90 cm	89 cm	90 cm	1 cm	1.11%
Distancia de 80 cm	79 cm	80 cm	1 cm	1.25%
Distancia de 70 cm	69 cm	70 cm	1 cm	1.43%
Distancia de 60 cm	59 cm	60 cm	1 cm	1.67%
Distancia de 50 cm	50 cm	50 cm	0 cm	0%
Distancia de 40 cm	40 cm	40 cm	0 cm	0%
Distancia de 30 cm	30 cm	30 cm	0 cm	0%
Distancia de 20 cm	20 cm	20 cm	0 cm	0%
Distancia de 10 cm	10 cm	10 cm	0 cm	0%

Fuente: Elaboración Propia

Después de haber obtenido los resultados de las mediciones podemos concluir que la fórmula N°2 nos da medidas más exactas conforme se amplía el rango de medición lo cual es favorable si nosotros deseamos obtener resultados confiables.

3.3.3 Resultados del Desarrollo de los algoritmos de reconocimiento con rostros conocidos o extraños.

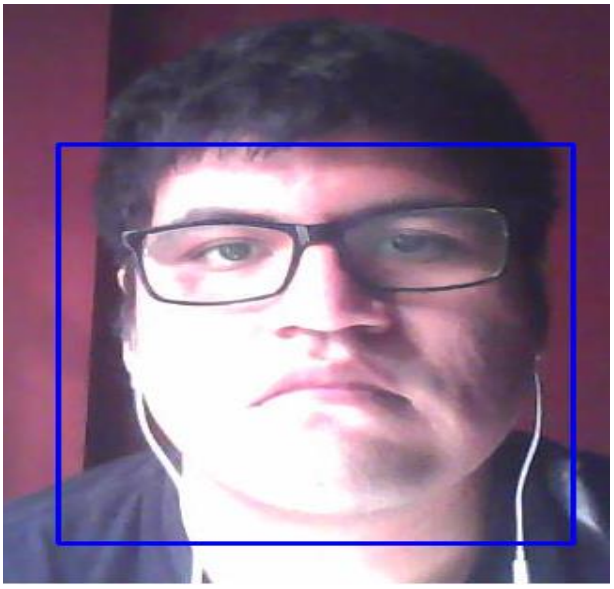
Al momento de ejecutar el programa del Anexo 7 desarrollado en Python se va activar la cámara de 5 MP de la Raspberry Pi donde probamos su funcionalidad con la Figura 68 en donde se puede observar la imagen a color y escala de grises.

Figura 68. Captura de imagen de la Cámara

Fuente: Elaboración Propia

Cuando ya se tiene la cámara de 5 MP activada en la Raspberry Pi se procede a realizar la programación del Anexo 8 en el cual se hará la detección del rostro de la persona encerrándolo en un marco de color azul tal como se muestra en la Figura 69.

Figura 69. Detección de Rostro



Fuente: Elaboración Propia

El siguiente paso después de confirmar la detección del rostro es crear una base de datos como se muestra en la Figura 70 para ello se realiza la programación del Anexo 9 en donde al momento de ejecutarlo el usuario deberá de mostrar la mayor cantidad de posiciones de su rostro para poder luego extraer los parámetros requeridos para el reconocimiento Facial.

Figura 70. Imágenes de la Base de Datos

Fuente: Elaboración Propia

Después de haber creado la base de datos con las 500 imágenes por usuario se pasa a realizar el entrenamiento de la base de datos en donde vamos a recorrer

todas las carpetas e imágenes que se encuentro en el directorio BaseDatos y se crearán las etiquetas para que luego se pueda reconocer a las personas con su respectivo nombre al momento de ejecutar el programa de Reconocimiento Facial, para el entrenamiento estas imágenes se pasan a escala de grises y para poder evitar el sobrelapamiento de las imágenes se usa la función ANTIALIAS y se crea la matrix de 8x8 con valores de 0 a 255.

El último paso se basa en poder comparar los parámetros que se extraen del entrenamiento de la base de datos con lo que la cámara comienza a detectar al momento de recibir los estímulos de los sensores como se muestra en la Figura 71, ya que el entrenamiento de la base de datos se realizó con las imágenes a escala de grises la comparación de parámetros en tiempo real también se debe de realizar a escala de grises sino se puede ocasionar una falsa lectura debido al ruido de la iluminación que puede estar presente en el ambiente del suceso.

Si en caso al momento de realizar la comparación de parámetros este no coincide con lo que se tiene en el archivo de entrenamiento entonces se enviará un mensaje informativo de la intrusión que se está realizando a la Residencia, dentro del mensaje se consideró pertinente enviar el número de teléfono de la comisaría más cercana en este caso el de la Comisaria de José Gálvez, además se envió una foto del suceso como se muestra en la Figura 73 para que el usuario decida si lo reporta a las autoridades correspondientes.

Figura 71. Reconocimiento Facial

Fuente: Elaboración Propia

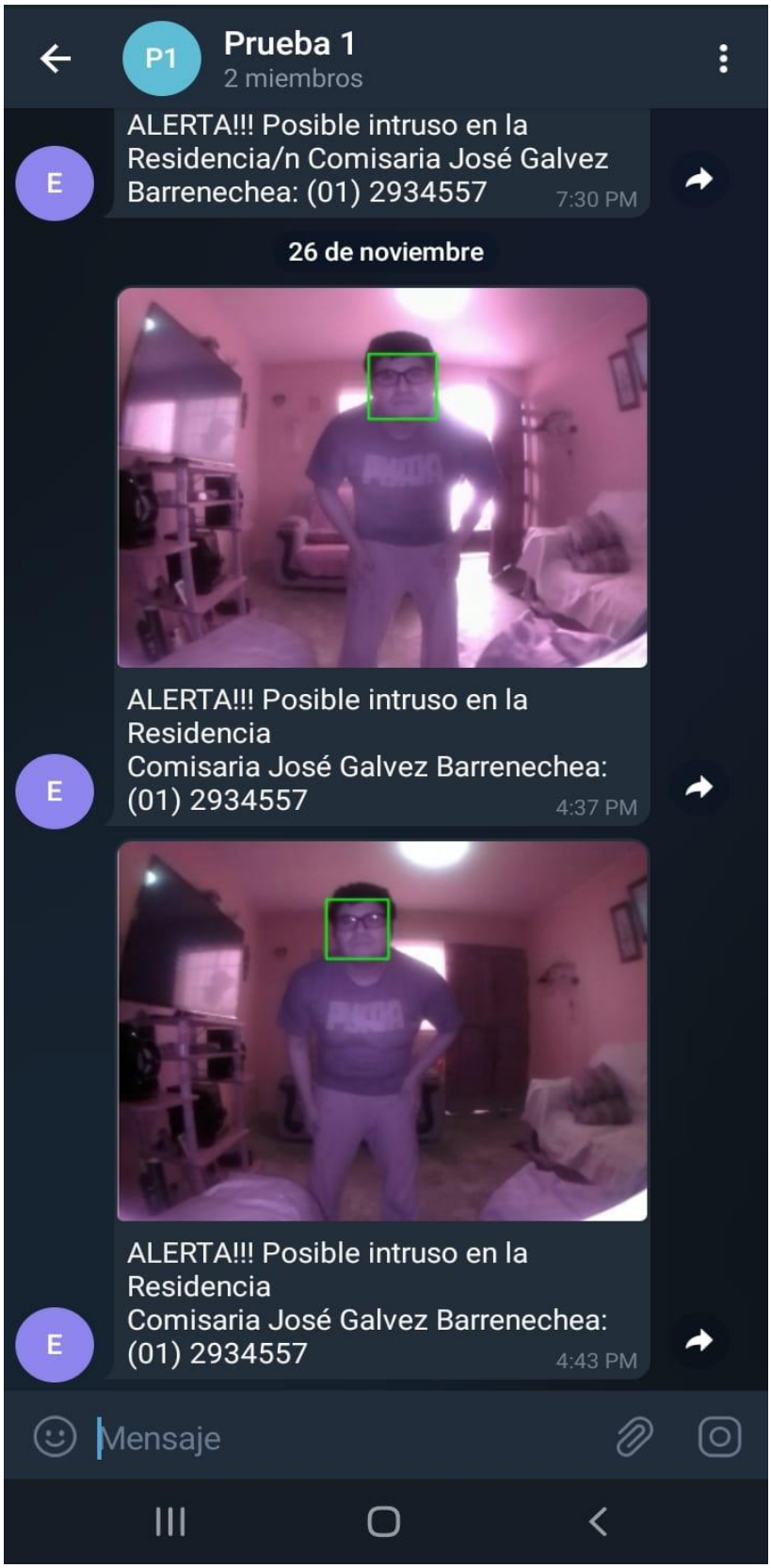
Figura 72. Alcance del Reconocimiento Facial

Fuente: Elaboración Propia

En el Anexo 11 se ve como funciona las entradas y salidas de la Raspberry Pi la cual vamos a añadir al Anexo 12 para que esta sea capaz de recibir los estímulos de los sensores de distancia y de movimiento para poder mandar una señal al Arduino Uno.

Para poder recibir mensajes en la Aplicación de Telegram enviada desde la Raspberry Pi vamos a añadir al programa de reconocimiento facial el Anexo 6 ya que esta es más completa que la lógica de programación del Anexo 5 en el cual solo en un mensaje se puede enviar texto pero con el Anexo 6 vamos poder tomar una captura de lo sucedió, guardarlo en una carpeta de Incidencias y poder enviar esa imagen con un texto descriptivo de posible intrusión además de enviar el número de teléfono de la comisarías más cercana.

Figura 73. Mensaje de Alerta a Telegram



Fuente: Elaboración Propia

3.3.4 Resultados del Desarrollo de la red neuronal.

De acuerdo al Anexo 13 hemos definido nuestras salidas como t = [0 0 0 1] lo cual al momento de ejecutar el Script obtenemos como resultado el valor de a en donde debemos de evaluar si lo obtenido se acerca próximamente al valor de la salida esperado sino se recomienda volver a ejecutar el Script así mismo se obtendrá una matriz de celdas para los pesos y bias de la primera capa y de la segunda capa por lo que usamos la función “celldisp” del Anexo 13 el cual nos permitió poder ver los valores de las matrices

```
a =

   -0.0033    0.0053    0.0025    0.9996

pesos1 =

  1×1 cell array

    {2×2 double}

bias1 =

  1×1 cell array

    {2×1 double}

pesos2 =

  1×1 cell array

    {1×2 double}

bias2 =

  1×1 cell array

    {[-0.6487]}

pesos1{1} =

    1.5538    7.6322

    6.3543   -5.0882

bias1{1} =

   -8.6144
```

4.4522

pesos2{1} =

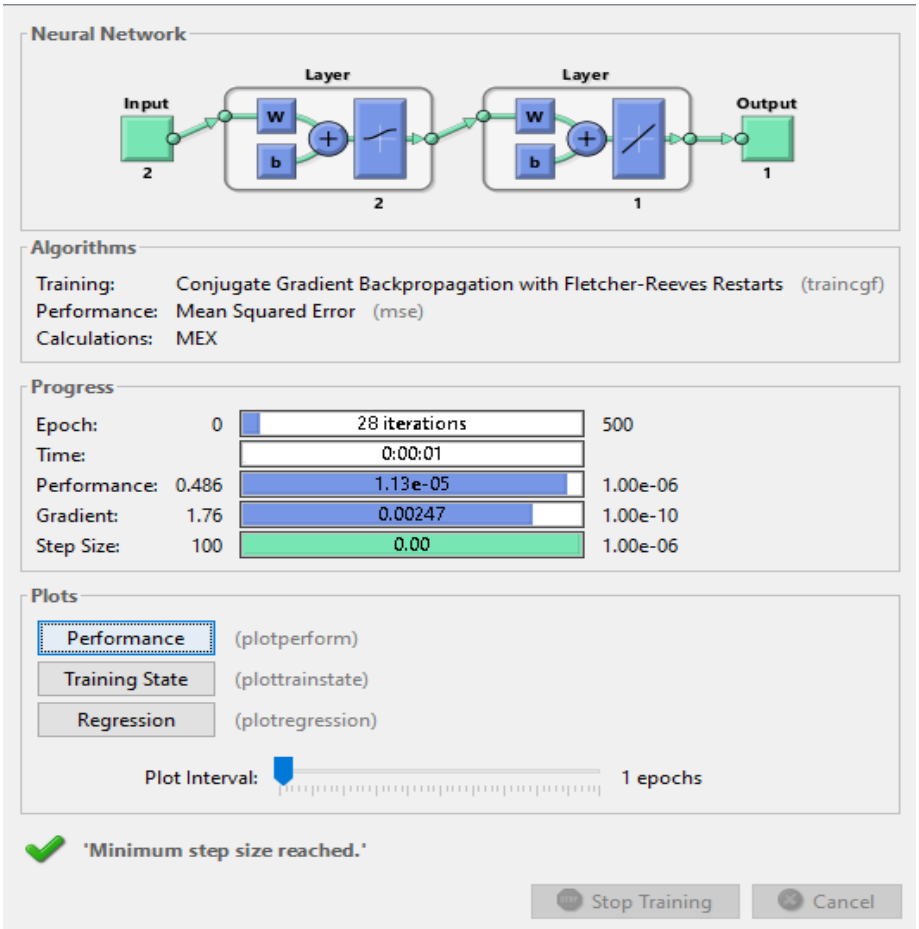
1.5612 0.6527

bias2{1} =

-0.6487

Tal como se muestra en la Figura 74 podemos observar que la Red Neuronal consta de 2 neuronas en la capa de entrada, tiene 2 capas ocultas con 2 y 1 neurona además de 1 neurona en la capa de salida, también se observa las gráficas de las funciones de activaciones de cada una de las capas, para la primera y segunda capa se especificó en el Anexo 12 que se usen las funciones sigmoide y lineal respectivamente, así mismo se indicó el valor de 500 como el máximo de épocas lo cual al ejecutar el Script solo lo realizó en 28 interacciones,

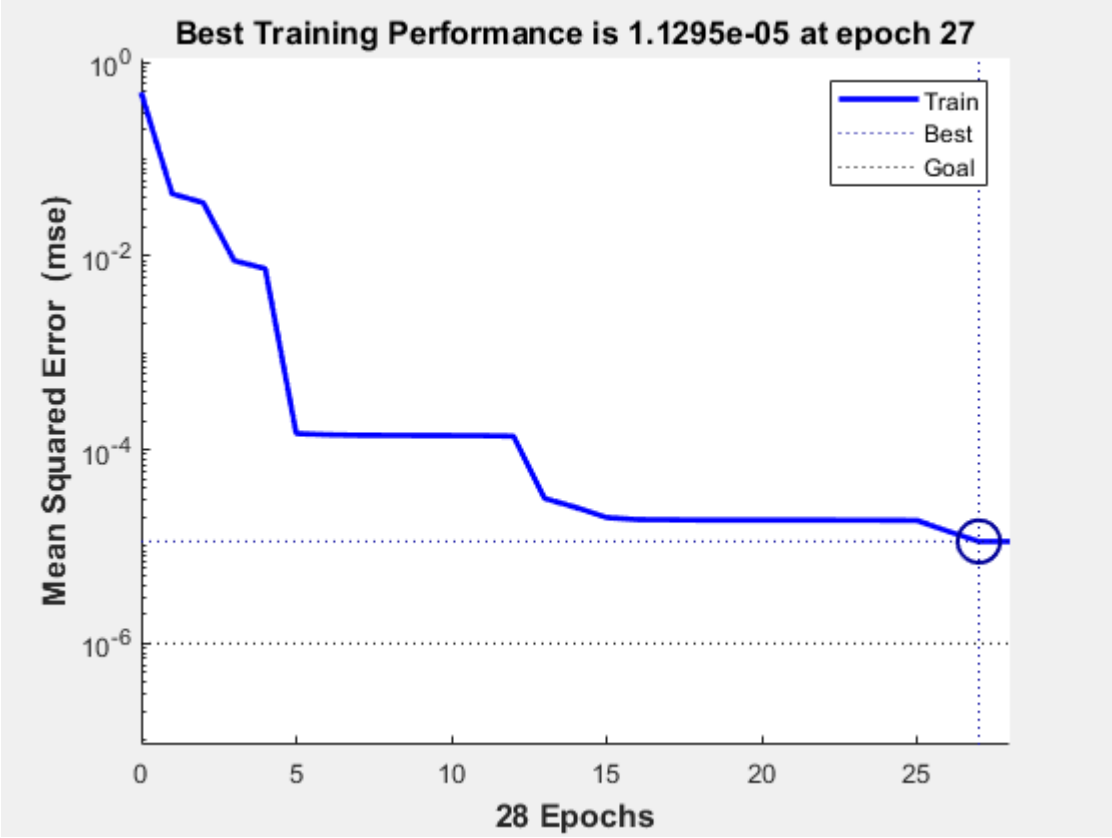
Figura 74. Simulación de la Red Neuronal



Fuente: Elaboración Propia

La Figura 75 muestra el Rendimiento del algoritmo TRAINCGF lo cual nos detalla que el mejor desempeño de validación es 1.1295e-05 realizado en 28 epocas en un tiempo de 1 segundo.

Figura 75. Rendimiento del Algoritmo TRAINCGF



Fuente: Elaboración Propia

Cuando ya se ha obtenido los pesos y bias al momento de ejecutar la programación del Anexo 13 pasamos a realzar la programación en Arduino tal como se muestra en el Anexo 4, esta será la última etapa para poder lograr que la alarma se encienda dependiendo de los estímulos que reciba del sensor de movimiento y de la identificación que realice la cámara ya que esta actúa también como si fuese un sensor más.

COMPARACIONES

Cuadro comparativo de la prueba de concepto con respecto a otros trabajos previos

Tabla 10. Comparación de datos obtenidos acerca de las imágenes

Factores	Datos utilizados por el autor Nico Surantha	Datos utilizados por nuestras simulaciones
Algoritmo utilizado	HOG y SVM	LBPH
Convierte las imágenes a escalas de grises	Si	Si
División de celdas	8x8	8x8
Normalización de gamma	Si	Si
Determina el número de bin	Si	Si
Detección de presencia humana	Si	No
Reconocimiento facial	No	Si
Resolución de cámara	No especifica	5 Mp
Personas registradas	0	2
Imágenes por persona	0	500
Tamaño de la base de datos	0	1000 imágenes

Fuente: Elaboración Propia

En el trabajo realizado por Hailay Berihu Abebe, Chih-Lyang Hwang se procesan previamente las imágenes para luego poder recortar el rostro utilizando el detector de rostros automático Viola-Jones luego se da el proceso de extracción de características LBP para la concatenación de los vectores de características RGB-D LBP y clasificación de SVM multiclase, se tiene 4 bases datos las cuales son IIIT-D RGB-D, VAP RGB-DT, EURECOM, NTUST_IRL RGB-D que tienen 4605 15300, 936, 2953 imágenes respectivamente. Para la creación de la base de datos la cual usan la cámara ASUS Xtion Pro Live y OpenNI SDK, en comparación con lo desarrollado no se indica bajo que condiciones es la que se activa la cámara ni si se pueda realizar el proceso de Reconocimiento Facial en la oscuridad, ni el tipo de hardware que se desarrolla para que se logre el objetivo final, nuestro sistema usa el algoritmo LBPH tiene una base de datos de 500 imágenes por usuario, se usa una cámara de 5 MP de resolución y al ser infrarrojo no existe problema alguno para poder realizar el Reconocimiento facial a oscuras.

CONCLUSIONES

- El sistema desarrollado es capaz de alertar al usuario si se produce alguna intrusión enviando un mensaje a la aplicación de Telegram tal como se muestra en los Resultados obtenidos.
- Tener una planificación del Sistema que se desea desarrollar ayudó mucho para poder evaluar los requerimientos de cada etapa de nuestro trabajo y con ello se logró escoger adecuadamente cada uno de los componentes electrónicos o materiales los cuales fuimos basándonos en sus características mas relevantes que cumplan con lo mencionado en la pag 29-30.
- Al momento de realizar una simulación en el software Proteus a pesar de que se obtenga los resultados que se esperan pueden llegar a fallar cuando se implementa de manera física debido a que un entorno virtual no se tiene en cuenta los factores externos que puedan causar pérdida de información en una comunicación inalámbrica por lo que se tiene que realizar una modificación en la Lógica de la Programación tal como se muestra en el Anexo 1 la cual consiste en solo enviar el valor de “1” o “0” para subsanar los errores al momento de realizar la comunicación inalámbrica y de esta forma obtener valores más verídicos.
- El proceso de Reconocimiento Facial explicado en el Anexo 12 se llega a obtener después de una serie de Programas que parten desde como encender la cámara (Anexo 7), detectar el rostro de una persona encerrándolo en un marco azul (Anexo 8), tomar captura de los perfiles de las personas (Anexo 9) como lo son sus gestos sus estados de ánimo para obtener la mayor cantidad de características posibles y luego hacer un entramiento de estas imágenes (Anexo 10) para poder compararlas con la imagen de la persona que esté enfocando al momento que se activen los sensores lo cual se explica en el Anexo 11 y enviar mensaje a Telegram como se explica en el (Anexo 6) y se aplique el Reconocimiento Facial bajo ciertas condiciones que dependen de los sensores.
- Al momento de ejecutar la Red Neuronal explicada en el Anexo 13 se ha obtenido los valores de los pesos y bias que se muestran en el punto 3.3.4

lo cual nos fueron de ayuda para poder plasmar las ecuaciones del Anexo 4 y de esta forma al momento de recibir los estímulos del sensor de movimiento y de la cámara se puede lograr encender la alarma la cual está comprendida por el buzzer de sonido.

RECOMENDACIONES

- Se pueden cambiar los sensores de distancias HC-sr04 colocados en el marco de la ventana y en la puerta por los sensores Sharp para obtener resultados más exactos ya que este último usa el método de la triangulación el cual se basa en medir uno de los ángulos que forma el triángulo emisor-objeto-receptor que detecta el punto de incidencia el cual depende del ángulo y a su vez de la distancia del objeto.
- Al momento de ejecutar el programa de Matlab para la Red Neuronal dependiendo de las entradas se obtendrán las salidas esperadas y en ocasiones se obtiene resultados no muy exactos a los valores de 0 y 1 que corresponden a las salidas es por ello que se recomienda realizar las simulaciones sucesivas hasta obtener valores próximos a los deseados de esta manera se obtendrá los valores de los pesos y bias que fueron necesarios para obtener los resultados que serán necesarias para la programación de las ecuaciones en el Arduino Uno.
- El Flex que une la cámara con la Raspberry Pi tiene que ser de unos 30 cm para tener una mayor facilidad de acomodamiento.
- La Base de Datos se tiene que ir actualizando cada cierto tiempo debido a que cabe la posibilidad que el usuario se deje crecer la barba o el cabello haciendo que de esta forma su apariencia cambie y posiblemente no sea reconocido por nuestro algoritmo ya que sus características van a ser otras.
- Para un buen funcionamiento de las entradas que recibe la Raspberry Pi por parte de los Nodos Receptores se recomienda que todos los Microcontroladores implicados tengan un solo GND en común de lo contrario la Raspberry Pi no podrá identificar de forma correcta las entradas.
- Al momento de usar los sensores de distancia se recomienda comprobar por uno mismo las distancias máximas que se logran medir porque en algunas

ocasiones no son exactamente como nos indican en su información, para poder ver las distancias obtenidas se recomienda usar el Monitor Serial.

- Para una mayor durabilidad de los Nodos electrónicos se deberían de usar baterías Duracell Alcalinas que dura hasta 10 veces más que las baterías de Zinc-Carbón.
- Al momento de pegar el Acrílico se puede realizar con diferentes pegamentos, pero para una mayor estética se recomienda usar Cloroformo ya que este hace que los acrílicos se junten como una sola pieza y se desvanece sin dejar mancha o marca.
- Siempre al momento de cambiar algún componente electrónico de la Raspberry o de los Nodos debe de realizarse el apagando la fuente de alimentación de lo contrario pueda que se queme algún componente.

LIMITACIONES

- El funcionamiento de los Nodos Transmisores y Receptores depende de su fuente de alimentación, en este caso solo se está usando una batería de 9V por lo que si uno de estos deja de funcionar no existe un plan de contingencia o una señal que avise al usuario que la fuente se está agotando y que necesita ser cambiada.
- El enlace VNC que se proporcionó para poder enlazarnos de forma remota solo se da si se está conectado a la misma red de Área local sino no se podrá conectar a la Raspberry Pi.
- Si en caso uno de los extremos ya sea la Raspberry Pi o el usuario Final no cuenta con internet a pesar de que haya una intrusión no se podrá enviar el mensaje o no se podrá recibir el mensaje.
- Los colores que se muestran en la cámara sufren un poco de variación por lo que no son exactos.

REFERENCIAS BIBLIOGRÁFICAS

Arun Cyril, J y Reza, M. (2017). Improving Smart home security; intetrating logical sensing into Smart home. *IEEE Sensors Journal,* 17(13), pp. 4269-4286. 10.1109/JSEN.2017.2705045

Becerra Suárez, F.L., (2019). Patrones de Conducta Facial para Identificar Accesos Informáticos no Autorizados. [Tesis para Título]. Universidad Señor de Sipán.

Crispino, A, Salvo, B, Antonino, C, Alfio, G, Salvatore, A, Antonio, I, Gianluca, G, Giorgio, M, Giuseppe, N, Gino, S, Gaetano, P, Antonio, S y Salvatore, P. (2019). Autonomous Energy-Efficient Wireless sensor network platform for home/office automation. *IEEE Sensors Journal,* 19(9), pp. 3501-35127. 10.1109/JSEN.2019.2892604

Gonzáles García, C, Meana-Llorián, D, Pelayo G-Bustelo, B.C., Cueva Lovelle, J.M. y García Fernandez, N. (2017). Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns and Smart Homes. *Future Generation Computer Systems*. 76, pp. 301-313. 10.1016/j.future.2016.12.033

Hailay Berihu A. y Chih-Lyang H. (2019). RGB-D face recognition using LBP with suitable feature dimension of depth image. *Cyber-Physical Systems: Theory & Applications IET*, 4, pp. 189-197. 10.1049 / iet-cps.2018.5045

ICHI.PRO, (22 noviembre, 2021). Sistema de reconocimiento facial con algoritmo LBPH.https://webcache.googleusercontent.com/search?q=cache:pL3etSMrpNwJ:https://ichi.pro/es/sistema-de-reconocimiento-facial-con-algoritmo-lbph-54175450924614+&cd=10&hl=es&ct=clnk&gl=pe

Nico, S y Wingky R, W. (Agosto, 2018). *Design of Smart home security system using object recognition and PIR sensor.* Comunicación presentada en 3rd

International Conference on Computer Science and Computational Intelligence, Indonesia. 135, pp 465-472. 10.1016/j.procs.2018.08.198

Nodo. (s.f.). En es.wikipedia.org. Recuperado el 30 de noviembre de 2021, de https://es.wikipedia.org/wiki/Nodo_(inform%C3%A1tica)

Ojala, T, Pietikäinen, M y Harwood, D. (1996), "A comparative study of texture measurements with classification based on feature distributions, 29, pp 51-59.

Prashanth Balraj, B y K. T, J. (2018). IoT-Based Facial Recognition Security System. IEEE. 10.1109 / ICSCET.2018.8537344

Real Academia Española. (s.f.). Algoritmo. En Diccionario de la lengua española. Recuperado en 30 de noviembre de 2021, de https://dle.rae.es/algoritmo

Reconocimiento Facial. (s.f.). En latam.kaspersky. Recuperado el 30 de noviembre de 2021, de https://latam.kaspersky.com/resource-center/definitions/what-is-facial-recognition

Redacción Gestión (06 junio, 2019) Estos son los 120 distritos del Perú con mayor delincuencia y violencia del país, según la PNP. *Gestión*. Recuperado de https://gestion.pe/peru/policia-detecta-120-distritos-crimenes-violencia-269349-noticia/

Red Neuronal. (s.f.). En ibm.com. Recuperado el 30 de noviembre de 2021, de https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

Sensor. (s.f.). En dewesoft.com. Recuperado el 30 de noviembre de 2021, de https://dewesoft.com/es/daq/que-es-un-sensor

Villa María del Triunfo: Delincuentes roban tableta y celular que escolar utilizaba para clases virtuales. (31 marzo, 2021). *América noticias*. Recuperado de

https://www.americatv.com.pe/noticias/actualidad/villa-maria-triunfo-delincuentes-roban-tableta-y-celular-que-escolar-utilizaba-clases-virtuales-n437108

Villa María del Triunfo: Sujeto armado asalta peluquería en menos de 30 segundos (12 setiembre, 2021). *América Noticias.* Recuperado de https://www.americatv.com.pe/noticias/actualidad/villa-maria-triunfo-sujeto-armado-asalta-peluqueria-menos-30-segundos-n445355?ref=irela

Waheb, J, Tee, K, Roshahliza, R, Siti, Z, Nurthaqifah, Z, Mohammed, B, Vladimir, S y Soltan, A. (2019). Design and fabrication of Smart home with internet of things enabled automation system. *IEEE Access.* 7, pp. 144059-144074. 10.1109/ACCESS.2019.2942846

ANEXOS

ANEXO 1. PROGRAMACIÓN REALIZADA EN ARDUINO DEL NODO EMISOR DEL SENSOR DE DISTANCIA

```
#include <SoftwareSerial.h> // Esta libreria me permite utilizar los
pines
                            // digitales como RX y TX
SoftwareSerial mySerial(9, 10); // Definimos los pines digitales 9 y 10
como RX
                              // y TX respectivmente
int trig = 5; // Definimos la variable trig al pin 5
int echo = 6; // Definimos la variable echo al pin 6
long duracion; // Variable para medir el tiempo de duración del pulso
int distancia; // Variable para medir la distancia

int alarma = 2; // Se asigna a la variable "alarma" el pin digital 2
int aviso; // Variable de estado para enviar al nodo receptor

void setup() {
  pinMode(trig, OUTPUT); // Se define el pin 9 como salidaa
  pinMode(echo, INPUT); // Se define el pin 10 como entrada
  pinMode(alarma, OUTPUT); // Se define el pin 2 como salida
  Serial.begin(9600); // Comunicación con el monitor Serial
  mySerial.begin(9600); // Comunicación serie entre el Arduino y el
modulo bluetooth
  }

void loop() {
  digitalWrite(trig, LOW); // No se envia la señal del pulso
  delay(2); // Retardo de 0.002 segundos para la activación de la señal
  digitalWrite(trig, HIGH); // Se envia la señal del pulso
  delay(10); // Activación de 0.01 segundos de la señal trigger
  digitalWrite(trig, LOW); // No se envia la señal del pulso
  duracion = pulseIn(echo,HIGH); // Detecta el cambio de estaddo
  distancia = (duracion*0.034)/2; // Fórmula para calcular la distancia
  //distancia = (duracion)/58.4;
  if(distancia < 55) // Si la distancia medida es menor a 55
  {
    aviso = 1; // La variable aviso toma el valor de "1"
    digitalWrite(alarma, HIGH); // Led de prueba encendido
  }
  else // Distancia diferente a la condicion del if
  {
    aviso = 0; // La variable aviso toma el valor de "0"
    digitalWrite(alarma, LOW); // Led de prueba apagado
  }

  mySerial.write (aviso); // Se envia a traves del bluetooth la variable
valor
  Serial.println(distancia); // Se imprime el valor de la distancia en el
monitor Serial
  delay(1000); // Retardo de 1 segundo para la siguiente acción
 }
```

ANEXO 2. PROGRAMACIÓN REALIZADA EN ARDUINO DEL NODO EMISOR DEL SENSOR DE MOVIMIENTO

```
#include <Servo.h> // Librería para usar el Servomotor
#include <SoftwareSerial.h> // Esta libreria me permite utilizar los
pines
                            // digitales como RX y TX
Servo ServoSensor; // Asignamos un nombre a nuestro Servomotor
SoftwareSerial mySerial(9, 10); // Definimos los pines digitales 9 y 10
como RX
                                // y TX respectivmente
int i; // Creamos la variable "i" para la suma de los grados del
Servomotor
int movimiento; //
int salida = 2; // Se asigna a la variable "salida" el pin digital 2
byte sensor_movimiento = 5; // Variable para el Sensor de Movimiento
void setup() {
  ServoSensor.attach(6); // Variable para el Servomotor
  pinMode(sensor_movimiento,INPUT); // Se define el pin 5 como entrada
  pinMode(salida,OUTPUT); // Se define el pin 2 como salidaa
  Serial.begin(9600); // Comunicación con el monitor Serial
  mySerial.begin(9600); // Comunicación serie entre el Arduino y el
modulo bluetooth

}

void loop() {
  for (i = 0; i <= 180; i=i+10) // Se suma 10 a la variable "i" hasta que
sea menor o igual a 180
  {
    ServoSensor.write(i); // El Servomotor girará los grados que tome la
variable "i"
    delay(250); // Retardo de 0.25 segundos
    Serial.println(i); // Se imprime el valor de la variable "i"
    if (digitalRead(sensor_movimiento) == HIGH) // Si se detecta el
movimiento
    {
      Serial.println("Se detecta el movimiento"); // Se emprime el
mensaje en el Monitor Serial
      digitalWrite(salida,HIGH); // El pin 2 se pone en alto
      movimiento = 1; // La variable moviiento toma el valor de "1"
      mySerial.write (movimiento); // Se envia la varibale movimiento a
trasves del bluetooth
    }
    else // Si no se detecta el movimiento
    {
      digitalWrite(salida,LOW); // El pin 2 se pone en bajo
      movimiento = 0; // La variable moviiento toma el valor de "0"
      mySerial.write (movimiento); // Se envia la varibale movimiento a
trasves del bluetooth
    }
  }

  for (i = 170; i >= 0; i=i-10) // Se resta 10 a la variable "i" hasta
que sea menor o igual a 180
  {
    ServoSensor.write(i); // El Servomotor girará los grados que tome la
variable "i"
    delay(250); // Retardo de 0.25 segundos
```

```
    Serial.println(i); // Se imprime el valor de la variable "i"
    if (digitalRead(sensor_movimiento) == HIGH) // Si se detecta el
movimiento
    {
      Serial.println("Se detecta el movimiento"); // Se emprime el
mensaje en el Monitor Serial
      digitalWrite(salida,HIGH); // El pin 2 se pone en alto
      movimiento = 1; // La variable moviiento toma el valor de "1"
      mySerial.write (movimiento); // Se envia la varibale movimiento a
trasves del bluetooth
    }
    else // Si no se detecta el movimiento
    {
      digitalWrite(salida,LOW); // El pin 2 se pone en bajo
      movimiento = 0; // La variable moviiento toma el valor de "0"
      mySerial.write (movimiento); // Se envia la varibale movimiento a
trasves del bluetooth
    }
  }
}
```

ANEXO 3. PROGRAMACIÓN REALIZADA EN ARDUINO DEL NODO RECEPTOR

```
#include <SoftwareSerial.h> // Esta libreria me permite utilizar los
pines
                            // digitales como RX y TX
SoftwareSerial mySerial(8, 9); // Definimos los pines digitales 8 y 9
como RX
                               // y TX respectivmente

// Definimos las variables y pines a usar
int valor; // Creamos la variable valor para poder recepcionar lo que nos
envia el
          // bluetooth configurado como esclavo
int alarma = 2; // Se asigna a la variable "alarma" el pin digital 2
void setup()
{
 pinMode(alarma, OUTPUT); // Definimos la variable alarma como "salida"
 mySerial.begin(9600); // Comunicación serie entre el Arduino y el modulo
bluetooth
 Serial.begin(9600); // Comunicación con el monitor Serial
}

void loop()
{
 if(mySerial.available()) // Si llego algún dato del bluetooth esclavo se
realiza
                          // la acción
 {
  valor = mySerial.read(); // Leo el dato que llegó
  Serial.println(valor); // Imprimo en el monitor Serie el valor obtenido
  delay(1000); // Retardo de 1 segundo para poder ver el siguiente valor
 }

 if(valor == 1) // Si el valor obtenido por el módulo bluetooth es igual
a "1"
 {
  digitalWrite(alarma, HIGH); // El pin 8 tiene como salida 5 voltios
 }
 else // Si mi valor obtenido es dintinto a "1"
 {
  digitalWrite(alarma, LOW); // El pin 8 tiene como salida 0 voltios
 }
}
```

ANEXO 4. PROGRAMACIÓN REALIZADA EN ARDUINO DE LA RED NEURONAL

```
#include<math.h> // Librería para realizar cálculos matemáticos
int alarma = 7; // Se asigna a la variable "alarma" el pin digital 7
byte p1,p2; // Se crea las variables de las entradas
float n1,n2,n3; // Se crea las variables de la sumatoria de la salida de
la neurona
float a1,a2; // Se crea las variables de la salida de la función de
activación
float e = 2.7182818; // Se usa para la función tangente hiperbólica
void setup() {
  pinMode(2,INPUT); // Se define el pin 2 como entrada de la señal de la
Cámara
  pinMode(3,INPUT); // Se define el pin 3 como entrada de la señal del
Sensor de Movimiento
  pinMode(7,OUTPUT); // Se define el pin 7 como salida para el Buzzer de
Sonido
  Serial.begin(9600); // Comunicación con el monitor Serial
}

void loop() {
  p1 = digitalRead(2); // Se lee el valor del pin digital 2
  p2 = digitalRead(3); // Se lee el valor del pin digital 3
  Serial.print(p1); // Se emprime el valor de p1 en el Monitor Serial
  Serial.print(" ");
  Serial.print(p2); // Se emprime el valor de p2 en el Monitor Serial
  Serial.print(" ");
  n1 = p1*(1.5538)+p2*(7.6322)+(-8.6144); // Sumatoria de la neurona 1
  n2 = p1*(6.3543)+p2*(-5.0882)+(4.4522); // Sumatoria de la neurona 2
  a1 = sigmoide(n1); // Obtención de la salida de la primera neurona
  a2 = sigmoide(n2); // Obtención de la salida de la segunda neurona

  n3 = a1*(1.5612) + a2*(0.6527) + (-0.6487); // Salida de la red
Neuronal
  n3 = round(n3); // Se redondea el valor de n3
  if (n3 == 1) // Si la salida es igual a "1"
  {
   digitalWrite(7,n3); // El pin 7 se pone en alto
   Serial.println(n3); // Se emprime el valor de n3 en el Monitor Serial
  }
  else
  {
   digitalWrite(7,LOW); // El pin 7 se pone en alto
   Serial.println(n3); // Se emprime el valor de n3 en el Monitor Serial
  }
  }
 // Funcion sigmoide
float sigmoide(float x){
  float a; // Se crea la variable para la Función
  a = (1/(1+pow(e,-x)));
  return a; // Devuelvo el valor de "a"
  }
```

ANEXO 5. PROGRAMACIÓN REALIZADA EN PYTHON PARA ENVIAR MENSAJE A LA APLICACIÓN DE TELEGRAM

```
import os
import requests
import json
# id = "1571007118" # ID del Chat Individual con el Bot
id = "-1001377608376" # ID del Chat con el grupo
token = '2026750721:AAGIRgtkqh4ohLJLWTAcRjVbwzIWdecHDRg' # Token del Bot
mensaje = 'ALERTA!!! Posible intruso en la Residencia' # Mensaje a mandar
url = "https://api.telegram.org/bot" + token + "/sendMessage" # URL del
servidor
parametros = {'chat_id': id, 'text': mensaje} # parametros para enviar al
servidor
for i in range(3): # la varibale i tomara el valor de 0,1,2
    if i < 3: # Mientras i sea menor a 3 se cumplira lo siguiente
        enviar = requests.post(url, data=parametros) # Se envia datos al
sevidor de la URL
        respuesta = enviar.json() # convierte una cadena a diccionario
        print(respuesta) # Observamos los parametros que hemos enviado y
la respuesta que nos da
```

ANEXO 6. PROGRAMACIÓN REALIZADA EN PYTHON PARA ENVIAR MENSAJE Y FOTO A LA APLICACIÓN DE TELEGRAM

```
import telegram
from datetime import date
chat_id = "-1001377608376" # id del grupo de Telegram
bot_token = '2026750721:AAGIRgtkqh4ohLJLWTAcRjVbwzIWdecHDRg' # Token del
Bot
bot = telegram.Bot(token=bot_token)
with open("Incidencias/21-10-2021.jpg","rb") as photo_file:
        bot.sendPhoto(chat_id=chat_id, photo=photo_file, caption =
'ALERTA!!! Posible intruso en la Residencia')
```

ANEXO 7. PROGRAMACIÓN REALIZADA EN PYTHON PARA ENCENDER LA CÁMARA

```
import cv2
cap = cv2.VideoCapture(0) # Se inicia la cámara
cap.set(3,640) # Se define el ancho de la cámara
cap.set(4,480) # Se define el alto de la cámara
while(True): # Se apertura el bucle
    ret, frame = cap.read() # Captura de la cámara
    frame = cv2.flip(frame, -1) # Se usa para poder voltear la imagen de
la cámara
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # Se usa para poder
pasar a escala de grisis la imagen
    cv2.imshow('frame', frame) # Muestra la imagen de la cámara a color
    cv2.imshow('gray', gray) # Muestra la imagen de la cámara a gris
    k = cv2.waitKey(30) & 0xff # Se da un delay de 30 ms
    if k == 27: # El código ASCII de Esc es 27
        break # Cierra el bucle While
cap.release() # Deja de ejecutarse la caámara
cv2.destroyAllWindows() # Cierra todas las ventanas aperturada
```

ANEXO 8. PROGRAMACIÓN REALIZADA EN PYTHON PARA DETECTAR LOS ROSTROS

```
import numpy as np
import cv2
faceCascade =
cv2.CascadeClassifier('Cascades/haarcascade_frontalface_default.xml') #
Se usa para la deteccion de objetos
cap = cv2.VideoCapture(0) # Se inicia la cámara
cap.set(3,640) # Se define el ancho de la cámara
cap.set(4,480) # Se define el alto de la cámara
while True:
    ret, img = cap.read() # Captura de la cámara
    img = cv2.flip(img, -1) # Se usa para poder voltear la imagen de la
cámara
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Se usa para poder
pasar a escala de grisis la imagen
    faces = faceCascade.detectMultiScale(
    gray, # Imagen para detectar rostros
    scaleFactor = 1.2, # Escala de la imagen
    minNeighbors = 5, # El numero de vecinos que debe de tener cada
rectangulo
    minSize = (20, 20) # Tamaño minimo de la imagen
    )
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2) # rectangle sirve
para dibujar un rectangulo en la imagen
# Parametros: Imagen, punto inicial, punto final, color, grosor
        roi_gray = gray[y:y+h, x:x+w] # Region de interes
        roi_color = img[y:y+h, x:x+w] # Region de interes
        cv2.imshow('video',img) # Sirve para mostrar la imagen en una
ventanas                                                    #
Paramaetros: Nombre de la ventana y la imagen que se va mostrar
        k = cv2.waitKey(30) & 0xff # Se da un delay de 30 ms
        if k == 27: # El código ASCII de Esc es 27
            break # Cierra el bucle While
cap.release() # Deja de ejecutarse la caámara
cv2.destroyAllWindows() # Cierra todas las ventanas aperturadas
```

ANEXO 9. PROGRAMACIÓN REALIZADA EN PYTON PARA CREAR LA BASE DE DATOS

```
import cv2
import os
cam = cv2.VideoCapture(0) # Se inicia la cámara
cam.set(3, 640) # Se define el ancho de la cámara
cam.set(4, 480) # Se define el alto de la cámara
face_detector=
cv2.CascadeClassifier('Cascades/haarcascade_frontalface_default.xml') #
Se usa para la deteccion de objetos
usuario = input('Ingrese el nombre del Usuario : ') # Se ingresa el
nombre del Usuario
count = 0 # Iniciamos el contador para poder tomar las fotos
while(True):
    ret, img= cam.read() # Captura de la cámara
    img= cv2.flip(img, -1) # Se usa para poder voltear la imagen de la
cámara
    gray= cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Se usa para poder pasar
a escala de grisis la imagen
    faces= face_detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2) # rectangle
sirve para dibujar un rectangulo en la imagen
# Parametros: Imagen, punto inicial, punto final, color, grosor
        count += 1 # Sumo en 1 el contador de imagenes
# Se guarda las imagenes tomadas en la carpeta de BasedeDatos
        cv2.imwrite("BaseDatos/"+ usuario + "/"+ usuario + "_"
+str(count)+".jpg", grises[y:y+h, x:x+w])
        cv2.imshow('Creando Base de Datos', img)
        print("Se está tomando la foto número " + str(count))
        k= cv2.waitKey(100) & 0xff # Presionmos la tecla Esc para salir
del video
        if k == 27: # El código ASCII de Esc es 27
            break # Cierra el bucle While
        elif count >= 500: # Cuando llegue a tomar 500 fotos
            break # Se cierra el programa
print("")
print("Operación terminada. Base de Datos Creada")
cap.release() # Deja de ejecutarse la caámara
cv2.destroyAllWindows() # Cierra todas las ventanas aperturadas
```

ANEXO 10. PROGRAMACIÓN REALIZADA EN PYTHON PARA REALIZAR EL ENTRENAMIENTO DE LA BASE DE DATOS

```
import cv2
import os
import numpy as np
from PIL import Image

cascPath =
"C:/Users/Laptop/Desktop/Reconocimiento_Facial/Cascades/haarcascade_front
alface_alt2.xml"
faceCascade = cv2.CascadeClassifier(cascPath) # Ubicacion de la librerías

reconocimiento = cv2.face.LBPHFaceRecognizer_create() # Algoritmo LBPH

Ubicacion = os.path.dirname(os.path.abspath(__file__)) # Ubicacion de la
carpeta General
Ubicacion_imagenes = os.path.join(Ubicacion,"BaseDatos") #
Direccionamiento a la carpeta BaseDatos

current_id = 0
etiquetas_id = {} # Diccionario de las etiquetas
y_etiquetas = [] # Lista de las etiquetas
x_entrenamiento = []

for root, dirs, archivos in os.walk(Ubicacion_imagenes): # Recorrer todos
los archivos que se encuentran en la carpeta
    for imagen in archivos:
        if imagen.endswith("png") or imagen.endswith("jpg"):
            pathImagen = os.path.join(root,imagen) # Ubicacion de las
imagenes
            etiqueta = os.path.basename(root).replace(" ", "-") # Nombre
de la equiqueta

            # Se crea las etiquetas para cada Usuario en la Base de Datos
            if etiqueta not in etiquetas_id: # Si el nombre  de la
etiqueta no se encuentra en el diccionario etiquetas_id
                                             # hace lo siguiente
                etiquetas_id[etiqueta] = current_id # Se agrega la llave
etiqueta con el valor del current id
                current_id += 1 # sumo 1 unidad al valor inicial de la
variable current_id
            id_ = etiquetas_id[etiqueta]
            # print(etiquetas_id)
            # print(id_)

            pil_image = Image.open(pathImagen).convert("L") # Abre la
imagen y lo convierte a escala de grises
            tamanio = (550,550)
            imagenFinal = pil_image.resize(tamanio, Image.ANTIALIAS) #
ANTIALIAS se usa para evitar el
            image_array = np.array(pil_image,"uint8") # Se crea una
matriz con valores enteros que van de 0 a 255
            rostros = faceCascade.detectMultiScale(image_array, 1.5, 5)
```

```
            for (x,y,w,h) in rostros:
                roi = image_array[y:y+h, x:x+w]
                x_entrenamiento.append(roi)
                y_etiquetas.append(id_)

reconocimiento.train(x_entrenamiento, np.array(y_etiquetas))
reconocimiento.write('Entrenamiento/trainer.yml') # Se crea el archivo
de extención yml correspondiente al entrenamiento
```

ANEXO 11. PROGRAMACIÓN REALIZADA EN PYTHON PARA LOS SENSORES

```
import RPi.GPIO as GPIO
import time
Receptor_01 = [] # Receptor de la Ventana
Receptor_02 = [] # Receptor de la Puerta

entrada_01 = 11 # Defino mi primera variable al pin 11
entrada_02 = 13 # Defino mi segunda variable al pin 13
GPIO.setmode(GPIO.BOARD) # Numeración exterior de los pines
GPIO.setup(entrada_01, GPIO.IN) # Defino mis variables como entradas
GPIO.setup(entrada_02, GPIO.IN) # Defino mis variables como entradas

while True:
    Sensor_Ventana = GPIO.input(11) # Se lee el estado del pin 11
    Sensor_Puerta = GPIO.input(13) # Se lee el estado del pin 13
    print("El sensor de la puerta es", Sensor_Puerta)
    print("El sensor de la ventana es", Sensor_Ventana)

    if Sensor_Ventana == 1 and Sensor_Puerta == 0: # Si el pin 11 esta en
alto y el pin 13 en bajo
        Receptor_01.append('1') # Se agrega el valor de "1" a la lista
Receptor_01
        print("Solo Ventana")
    if Sensor_Puerta == 1 and Sensor_Ventana == 0: # Si el pin 13 esta en
alto y el pin 11 en bajo
        Receptor_02.append('1') # Se agrega el valor de "1" a la lista
Receptor_02
        print("Solo Puerta")
    if Sensor_Puerta == 1 and Sensor_Ventana == 1: # Si el pin 11 esta en
alto y el pin 13 en alto
        Receptor_01.append('1') # Se agrega el valor de "1" a la lista
Receptor_01
        Receptor_02.append('1') # Se agrega el valor de "1" a la lista
Receptor_02
        print("Ambos")
    if Sensor_Puerta == 0 and Sensor_Ventana == 0: # Si el pin 11 esta en
bajo y el pin 13 en bajo
        print("No se Activará la Cámara")
        print("")
        time.sleep(1.5) # Tiempo de retraso

GPIO.cleanup() # Limpio los puertos de las Raspberry que se han usado
```

ANEXO 12. PROGRAMACIÓN REALIZADA EN PYTHON PARA REALIZAR EL RECONOCIMIENTO FACIAL

```
import os
import cv2
import numpy as np
import requests
import json
import RPi.GPIO as GPIO
import time
import telegram
import pickle
from datetime import date

# Variables para el uso de la Raspberry
Receptor_01 = [] # Receptor de la Ventana
Receptor_02 = [] # Receptor de la Puerta
Signal = [] # Lista de Señal de salida para el arduino
entrada_01 = 11 # Se define el pin de entradaa
entrada_02 = 13 # Se define el pin de entrada
salida_01 = 12 # Se define el pin de salida
GPIO.setmode(GPIO.BOARD) # Manejo de los pines
GPIO.setup(entrada_01, GPIO.IN) # Se declara al pin 11 como entrada
GPIO.setup(entrada_02, GPIO.IN) # Se declara al pin 13 como entrada
GPIO.setup(salida_01, GPIO.OUT) # Se declara al pin 12 como salida

print("Si lee el programa de Reconocimiento")
# Variables para poder enviar mensaje a Telegram
Idd = "-1001377608376" # ID del Chat con el grupo
Token = '2026750721:AAGIRgtkqh4ohLJLWTAcRjVbwzIWdecHDRg' # Clave del chat
del BOT
mensaje = 'ALERTA!!! Posible intruso en la Residencia\nComisaria José
Galvez Barrenechea: (01) 2934557' # Mensaje a enviar
fecha = date.today() # Fecha de la Incidencia
bot = telegram.Bot(token=Token)
Lista_Mensaje = [] # Lista Mensaje

# Variables para realizar el reconocimiento facial
recognizer= cv2.face.LBPHFaceRecognizer_create() # Algoritmo LBPH
recognizer.read('Entrenamiento/trainer.yml') # Se lee el archivo de
Reconocimiento
cascadePath= 'Cascades/haarcascade_frontalface_default.xml' # Ubicacion
del clasificador de rostros
faceCascade= cv2.CascadeClassifier(cascadePath);
font= cv2.FONT_HERSHEY_SIMPLEX

etiquetas = {"nombre_persona" : 1 }
with open("labels.pickle",'rb') as f:
    pre_etiquetas = pickle.load(f)
    etiquetas = { v:k for k,v in pre_etiquetas.items()}
print("Las etiquetas son", etiquetas)
cam= cv2.VideoCapture(0) # Se inicia la cámara
cam.set(3, 640) # Se define el ancho de la cámara
cam.set(4, 480) # Se define el alto de la cámara
```

```
minW= 0.1*cam.get(3) # Tamaño mínimo del ancho de la cámara
minH= 0.1*cam.get(4) # Tamaño mínimo del alto de la cámara

while True:
    print("y lo está ejecutando")
    ret, img=cam.read() # Captura de la cámara
    img= cv2.flip(img, -1) # Se usa para poder voltear la imagen de la
cámara
    gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) # Se usa para poder pasar
a escala de grisis la imagen

    faces= faceCascade.detectMultiScale(
        gray, # Imagen para detectar rostros
        scaleFactor= 1.2, # Escala de la imagen
        minNeighbors= 5, # El numero de vecinos que debe de tener cada
rectangulo
        minSize= (int(minW), int(minH)), # Tamaño minimo de la imagen
       )
    Sensor_Ventana = GPIO.input(11) # lectura del pin 11
    Sensor_Puerta = GPIO.input(13) # Lectura del pin 13
    print("El sensor de la puerta es", Sensor_Puerta)
    print("El sensor de la ventana es", Sensor_Ventana)

    if Sensor_Ventana == 1 and Sensor_Puerta == 0: # Si el pin 11 esta en
HIGH y el pin 13 en LOW
        Receptor_01.append('1') # Se agrega el valor de "1" a la Lista
del Receptor de la Ventana
        print("Solo Ventana")
    if Sensor_Puerta == 1 and Sensor_Ventana == 0: # Si el pin 11 esta en
LOW y el pin 13 en HIGH
        Receptor_02.append('1') # Se agrega el valor de "1" a la Lista
del Receptor de la Puerta
        print("Solo Puerta")
    if Sensor_Puerta == 1 and Sensor_Ventana == 1: # Si el pin 13 esta en
HIGH y el pin 13 en HIGH
        Receptor_01.append('1') # Se agrega el valor de "1" a la Lista
del Receptor de la Puerta
        Receptor_02.append('1') # Se agrega el valor de "1" a la Lista
del Receptor de la Puerta
        print("Ambos")
    if Sensor_Puerta == 0 and Sensor_Ventana == 0: # Si el pin 13 esta en
LOW y el pin 11 en LOW
        print("No se Activará la Cámara")
        print("")
    time.sleep(1.5) # tiempo de espera para la lectura de los pines
    if Receptor_01 != [] or Receptor_02 != []: # Si la lista del Receptor
de la Ventana o de la Puerta es diferente a vacio
        for(x,y,w,h) in faces:
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2) #
rectangle sirve para dibujar un rectangulo en la imagen
            id, confidence= recognizer.predict(gray[y:y+h,x:x+w]) #
Precide las etiquetas
            # Check if confidence is less them 100==> "0" is perfect
match
            if (confidence < 100): # si la confianza es menor a 100
```

```
                nombre= etiquetas[id]
                confidence= "  {0}%".format(round(100 - confidence)) #
valor de la confianza
            else:
                nombre= "Desconocido" # Si la confianza es mayor a 100
                confidence= "  {0}%".format(round(100 - confidence)) #
valor de la confianza
                Lista_Mensaje.append('1')
                if len(Lista_Mensaje) == 1: # Si el numero de elementos
de la lista Mensaje es 1
                    print("La fecha es", fecha)
                    cv2.imwrite("Incidencias/" + str(fecha) + ".jpg",
img) # Se toma captura de la visualizacion de la cámara
                    with open("Incidencias/"+ str(fecha) + ".jpg","rb")
as photo_file: # Se abre la ubicacion de la foto tomada
                        bot.sendPhoto(chat_id=Idd, photo=photo_file,
caption = mensaje) # Se envia mensaje al grupo de Telegram
                    Signal.append("1") # Se asigna el valor de "1" a la
lista de Señal

                elif len(Lista_Mensaje) == 100:# Si el numero de
elementos de la lista Mensaje es 100
                    Lista_Mensaje.clear() # Se borra los elementos de la
lista
                    cv2.imwrite("Incidencias/" + str(fecha) + ".jpg",
img)  # Se toma captura de la visualizacion de la cámara
                    with open("Incidencias/"+ str(fecha) + ".jpg","rb")
as photo_file: # Se abre la ubicacion de la foto tomada
                        bot.sendPhoto(chat_id=Idd, photo=photo_file,
caption = mensaje) # Se envia mensaje al grupo de Telegram
                else:
                    ""
            cv2.putText(img, str(nombre), (x+5,y-5), font, 1,
(255,255,255), 2)
            cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1,
(255,255,0), 1)
    if Signal == []: # Si la lista Signal es igual a vacio
        GPIO.output(12, GPIO.LOW) # La salida del pin 12 esta en LOW
    else:
        print("Señal al arduino")
        GPIO.output(12, GPIO.HIGH) # La salida del pin 12 esta en HIGH

    cv2.imshow('camera',img) # Sirve para mostrar la imagen en una
ventanas
    k= cv2.waitKey(10) & 0xff # # Se da un delay de 10 ms
    if k== 27: # El código ASCII de Esc es 27
        break # Cierra el bucle While

print("\nSaliendo del programa")
GPIO.cleanup() # Limpia los pines que se han usado
cam.release() # Deja de ejecutarse la caámara
cv2.destroyAllWindows() # Cierra todas las ventanas aperturadas
```

ANEXO 13. PROGRAMACIÓN REALIZADA EN MATLAB PARA LA OBTENCION DE LOS PESOS Y BIAS DE NUESTRA RED NEURONAL

```
function Red_Neuronal
% Se borra la ventana de comandos y las variables
clc, clear all,
p = [0 1 0 1; % valores de entrada de la cámara
     0 0 1 1]; % valores de entrada del sensor de movimiento
t = [0 0 0 1]; % salida para el buzzer de sonido
% Se define el número de capas, funcion de activación y el algoritmo de
% aprendizajes
net=newff(minmax(p),[2 1],{'logsig','purelin'},'traincgf');
% Se inicializa la Red Neuronal
net=init(net);
% Se define las epocas máximas para el entrenamiento
net.trainParam.epochs=500;
% Se define el error máximo del entrenamiento
net.trainParam.goal=1e-6;
% Se realiza el entrenamiento con las entradas y salidas
net=train(net,p,t);
% Se inicia la Simulación
a=sim(net,p)
% Pesos y bias de la primera capa
pesos1 = net.iw(1,1)
bias1 = net.b(1,1)
% Pesos y bias de la segunda capa
pesos2 = net.lw(2,1)
bias2 = net.b(2,1)
% Muestra los valores de las matrices de celdas
celldisp(pesos1)
celldisp(bias1)
celldisp(pesos2)
celldisp(bias2)
```