

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

ALCOHOLÍMETRO Y DETECTOR DE IDENTIDAD

Trabajo de Titulación para optar al
Título de TÉCNICO UNIVERSITARIO EN
ELECTRÓNICA

Alumno:

Juan Pedro Toro Martin

Profesor Guía:

Sergio Riquelme Bravo

2023

Profesor Correferente:

José Llantén Álvarez

RESUMEN

KEYWORDS: ALCOHOLÍMETRO, SISTEMA DE ENCENDIDO VECHICULAR, SEGURIDAD VIAL, DETECCIÓN FACIAL

En el presente informe se plantea la problemática del actuar humano en la conducción, esencialmente cuando el individuo se encuentra bajo la influencia de alcohol. El desarrollo de este proyecto de titulación está compuesto de tres capítulos: fundamentación teórica, desarrollo de la solución y resultados de la implementación.

Primero se expondrán estadísticas asociadas a problemáticas de seguridad vial, abordando las causas más probables y evidenciando las consecuencias de estos siniestros. En este trabajo se realizará una investigación sobre el uso de redes neuronales en la detección de patrones y formas, con el fin de incrementar la seguridad de la solución al problema. Los objetivos del proyecto se establecen en el último punto del primer capítulo.

Posteriormente, se propondrá una solución para mejorar los índices de siniestros vehiculares, explicándose cada componente que conforma y permite el funcionamiento general del programa.

En última instancia, se mostrarán los resultados de la implementación de la solución propuesta, exponiendo conclusiones y complementando con recomendaciones sobre el prototipo y se evaluarán, además, los costos asociados al desarrollo del proyecto.

ÍNDICE

RESUMEN	2
ÍNDICE	3
ÍNDICE DE FIGURAS	5
ÍNDICE DE TABLAS	8
SIGLAS Y SIMBOLOGÍA	1
INTRODUCCIÓN	1
CAPÍTULO 1: PROBLEMÁTICA EN LA SEGURIDAD VIAL	2
1. PROBLEMÁTICA EN LA SEGURIDA VIAL	3
1.1 Accidentes vehiculares	3
1.1.1 De las causas de los accidentes.....	3
1.1.2 De los costos del siniestro	3
1.1.3 Control de alcohol en la sangre.....	4
1.2 Sensor de gases MQ3	7
1.2.1 De las características técnicas	7
1.2.2 De la estructura interna del sensor.	9
1.2.3 Del funcionamiento del módulo MQ3	9
1.3 Inteligencia Artificial.....	10
1.3.1 De los tipos de inteligencia	10
1.3.2 De la detección facial	13
1.4 Redes neuronales artificiales.....	14
1.4.1 Clasificación según topología o estructura de la red	17
1.4.2 Clasificación según su algoritmo de aprendizaje	19
1.4.3 De la red neuronal convolucional.....	22
1.4.4 Del aprendizaje en profundidad.....	30
1.4.5 De los modelos y técnicas en la detección de caras humanas.....	31
1.5 Computadoras de tarjeta simple	33
1.5.1 Raspberry Pi	34
1.6 Objetivos del proyecto	36
1.6.1 Objetivo general.....	36
1.6.2 Objetivos específicos	36
CAPÍTULO 2: ESTRUCTURA DE LA SOLUCIÓN	37
2. ESTRUCTURA DE LA SOLUCIÓN	38
2.1 Sistema de bloqueo.....	38

2.1.1	<i>Sistema de encendido de un motor</i>	39
2.2	Diseño alcoholímetro	42
2.2.1	<i>De la conversión análoga a digital</i>	43
2.2.2	<i>Del cálculo de la función del sensor</i>	45
2.2.3	<i>Cálculo de alcohol en la sangre</i>	49
2.3	Detector de identidad	49
2.3.1	<i>De la adición de usuarios a la base de datos</i>	51
2.3.2	<i>De la detección facial</i>	52
CAPÍTULO 3: IMPLMETACIÓN Y RESULTADOS DEL PROYECTO		54
3.	IMPLEMETACIÓN Y RESULTADOS DEL PROYECTO	55
3.1	Implementación del sistema	55
3.1.1	<i>De la conversión DC-DC</i>	57
3.1.2	<i>Del ajuste de sensibilidad del sensor MQ3</i>	58
3.1.3	<i>De la configuración del sensor de posición</i>	59
3.1.4	<i>De la configuración para Arduino Nano</i>	60
3.1.5	<i>De la configuración para Raspberry Pi</i>	60
3.2	Resultados de la solución	68
3.2.1	<i>Resultados en la creación de usuarios</i>	68
3.2.2	<i>Resultados de la medición de alcohol</i>	72
3.2.3	<i>Del prototipo del proyecto</i>	75
3.3	Costos del proyecto.....	76
3.3.1	<i>Estudio de mercado de los componentes eléctricos</i>	76
3.3.2	<i>Evaluación de costos de recursos humanos</i>	78
3.3.3	<i>Costo total del proyecto</i>	78
CONCLUSIONES		80
BIBLIOGRAFÍA.....		82

ÍNDICE DE FIGURAS

Figura 1-1. Tramos de concentración de alcohol en la sangre y sus consecuencias legales en un accidente.....	6
Figura 1-2. Relación entre alcohol en la sangre y el alcohol medido en el aire en el transcurso de seis horas.	7
Figura 1-3 . Imagen del sensor MQ3.	9
Figura 1-4 . Esquema de pines del módulo MQ3.	9
Figura 1-5 . Circuito polarizador de sensor MQ3.	10
Figura 1-6. Imagen simplificada de una neurona artificial.	16
Figura 1-7. Imagen referencial de una capa de neuronas con conexión: A. Recurrente. B. Hacia adelante.	18
Figura 1-8. Imagen de un filtro con dimensiones 3x3.	22
Figura 1-9. Imagen del uso de un filtro sobre un mapa de características.	23
Figura 1-10. Construcción de patrones mediante capas.....	24
Figura 1-11. Ilustración de una convolución a una imagen con dimensión 32 x 32 x 3 pixeles. ...	25
Figura 1-12. Convolución de una imagen de un rostro.	26
Figura 1-13. Gráfica de la función ReLU.	27
Figura 1-14. Aplicación de la función ReLU a una mapa de características.	27
Figura 1-15. Agrupación por máximos de una imagen simple.	28
Figura 1-16. Aplicación de la función de agrupación a una mapa de características.	29
Figura 1-17. Aplastamiento de una matriz agrupada simple con dimensiones 3x3.	29
Figura 1-19. Características del Clasificador Haar.	32
Figura 1-20. Clasificador Haar sobre una imagen de un rostro.	33
Figura 1-21. Módulo V1 Rev 1.3 de cámara para Raspberry Pi	34
Figura 2-1. Diagrama de flujo del sistema de bloqueo de corriente a la unidad de encendido. ..	38
Figura 2-2. Esquema de un sistema de encendido convencional de la marca BOSH.	39
Figura 2-3 . Diagrama de un Interruptor de encendido de cuatro posiciones.	40
Figura 2-4. Componentes de un sistema de encendido de motor D-Jetroninc de Bosh.	42
Figura 2-5. Diagrama de flujo del programa de medición de alcoholemia.	43

Figura 2-6. Esquema de típica aplicación de conversión analógica a digital con el componente ADC0408.	44
Figura 2-7. Gráfica logarítmica de la relación R_s/R_o vs. la concentración de gas detectada.	46
Figura 2-8. Diagrama esquemático del módulo MQ3.	47
Figura 2-9. Diagrama de flujo del programa de detección de identidad.	50
Figura 2-10. Diagrama de flujo de la actualización de datos de identidad en la base de datos interna.....	51
Figura 2-11. Diagrama de flujo de la detección de caras y sus características personalizadas.....	53
Figura 3-1. Diagrama de la conexión eléctrica entre los componentes del proyecto.	55
Figura 3-2. Circuito eléctrico simulado de la conexión entre placas y módulos.	57
Figura 3-3. Esquema del circuito regulador de voltaje.....	58
Figura 3-4. Esquema del circuito sensor de posición para realizar la medición.	59
Figura 3-5. Funciones implementadas en el código de Arduino Nano.	60
Figura 3-6. Configuración del ecosistema del proyecto.	61
Figura 3-7. Creación del directorio para usuario nuevo.....	61
Figura 3-8. Función para agregar nuevos usuarios a la base de datos.	62
Figura 3-9. Función encargada de actualizar la base de datos de los usuarios.	63
Figura 3-10. Función encargada de la identificación de personas detectadas.....	65
Figura 3-11. Clasificación de casos para la validación de identidad del usuario.	66
Figura 3-12. Bucle para la ejecución del test de alcohol.	66
Figura 3-13. Función encargada de realizar el test de alcoholemia.....	67
Figura 3-14. Comprobación de la base de datos del sistema.	68
Figura 3-15. Captura de imágenes del usuario detectado.	69
Figura 3-16. Creación del perfil para el primer usuario.	69
Figura 3-17. Detección de usuario registrado y usuario desconocido.	70
Figura 3-18. Captura de imágenes del usuario nuevo detectado.	71
Figura 3-19. Almacenamiento de imágenes del usuario adicional.	71
Figura 3-20. Detección de usuarios dentro del sistema.	72
Figura 3-21. Inicio del test de alcoholemia a usuario identificado.	73
Figura 3-22. Resultados del test de alcoholemia con presencia de alcohol.....	73

Figura 3-23. Inicio del test de alcoholemia a distinto usuario registrado.	74
Figura 3-24. Resultados del test de alcoholemia sin presencia de alcohol.	75
Figura 3-25. Captura fotográfica de la maqueta del proyecto.	76

ÍNDICE DE TABLAS

Tabla 1-1. Costos sociales de accidentes vehiculares (en UF).	4
Tabla 1-2. Descripción de los efectos del alcohol en distintas cantidades.	5
Tabla 1-3. Condiciones de trabajo estandar del sensor MQ3.....	8
Tabla 1-4. Características de la sensibilidad del sensor MQ3 en alcohol.....	8
Tabla 1-5. Comparación técnica de módulos de cámaras para Raspberry Pi	35
Tabla 2-1. Condiciones de encendido/apagado de un interruptor de encendio.	41
Tabla 2-2. Detalles técnicos del microprocesador Arduino NANO	45
Tabla 3-1. Selección de pines para las variables de entrada en Arduino Nano.....	56
Tabla 3-2. Selección de pines de comunicaión serial de Raspberry Pi y Arduino Nano.....	56
Tabla 3-3. Selección de pines para las variables de entrada y salida de la Raspberry PI.	57
Tabla 3-4. Resultados en el cálculo de valores resistivos del módulo del sensor.	58
Tabla 3-3. Listado de cantidad y precio de componentes por proveedores.	77
Tabla 3-4. Costo mano de obra.	78
Tabla 3-5. Costos finales asociados a la compra de los componentes.	79
Tabla 3-6. Costo final del proyecto.....	79

SIGLAS Y SIMBOLOGÍA

A. SIGLAS

ADC	:	Convertidor Análogo a Digital
AI	:	Artificial Intelligence (Inteligencia Artificial)
ANN	:	Artificial Neural Network (Red Neuronal Artificial)
BAC	:	Blood Alcohol Content (Contenido de Alcohol en la Sangre)
BrAC	:	Breath Alcohol Content (Contenido de Alcohol en el Aliento)
CNN	:	Convolutional Neural Network (Red Neuronal Convolucional)
CONASET	:	Comisión Nacional de Seguridad de Tránsito
CSI	:	Camera Serial Interface
DC	:	Corriente Directa
GPIO	:	General Purpose Input/Output (Entrada/Salida de Propósito General)
GPU	:	Graphics Processing Unit (Unidad Procesadora de Gráficas)
IBM	:	International Business Machines Corporation
ML	:	Machine Learning (Aprendizaje de Máquinas)
MLFF	:	Multi-Layer Feed-Foward
PIB	:	Producto Interno Bruto
PWM	:	Pulse Width Modulation (Modulación por Ancho de Pulsos)
SBC	:	Single-Board Computer (Computadora de Placa Simple)
UF	:	Unidad de Fomento
PCA	:	Principal Component Analysis (Análisis de Componentes Principales)

B. SIMBOLOGÍA

A	:	Amper
cm	:	Centímetro
GB	:	Gigabyte
Ghz	:	Gigahercio
g	:	Gramo
g/l	:	Gramo en litro
hrs	:	Horas
KB	:	Kilobyte
KHz	:	Kilohertz
KΩ	:	Kilo-ohm
L	:	Litro
mm	:	Milímetro
MΩ	:	Mega-ohm
mg	:	Mili-gramo
mg/L	:	Mili-gramo en litro
ms	:	Mili-segundo
mW	:	Mili-Watts
s	:	Segundo
Ω	:	Ohm
us	:	Micro-segundo
V	:	Volt
\$	:	Peso Chileno

INTRODUCCIÓN

El alcohol es una sustancia psicoactiva cuyas propiedades son causantes de dependencia. La ingesta de alcohol ha sido una práctica durante siglos por distintas civilizaciones y culturas. Esta actividad trae consigo factores de riesgo inmediato en las personas, desde el consumo hasta la creación de dependencia en el largo plazo. Actualmente el consumo de esta sustancia se ha normalizado, y en nuestro país se trata de una costumbre socialmente aceptada.

Los riesgos no solo agreden a quienes se encuentran bajo los efectos del alcohol, ya que si se añade un vehículo a la ecuación, la situación se convierte en un entorno perfecto para que el conductor cometa errores y se provoque un accidente, resultando en una amenaza para cualquiera que se cruce en el camino de un conductor alcoholizado.

Para resolver esta problemática se exploran distintas áreas que permitan mejorar la seguridad vial. El uso de herramientas como la inteligencia artificial, redes neuronales, y clasificadores abren la puerta para el desarrollo de soluciones más robustas y seguras.

CAPÍTULO 1: PROBLEMÁTICA EN LA SEGURIDAD VIAL

1. PROBLEMÁTICA EN LA SEGURIDA VIAL

1.1 ACCIDENTES VEHICULARES

Según la Comisión Nacional de Seguridad de Tránsito (CONASET), en Chile 1 de cada 10 siniestros de tránsito con consecuencias fatales se debe al consumo de alcohol en el conductor.

1.1.1 De las causas de los accidentes

Según un estudio elaborado por CONASET, las causas de mayor recurrencia en accidentes viales corresponde a:

- a) Consumo de alcohol en el conductor
- b) Infracciones de tránsito
- c) Imprudencia del conductor
- d) Pérdida del control del vehículo
- e) Exceso de velocidad

1.1.2 De los costos del siniestro

Según los datos recopilados por el observatorio de la CONASET y representados en la tabla 1-1, se estima que en el año 2020, las repercusiones económicas para el Estado chileno producto de siniestros vehiculares equivale al 2,01% del PIB.

Tabla 1-1. Costos sociales de accidentes vehiculares (en UF).

Costo Social Total	(UF)	% del Costo	% PIB
Daños a vehículo	14.880.166,82	11%	0,21%
Asociados a lesionados	123.415.452,50	89%	1,79%
Total	138.295.619,32		
% PIB	2,01%		

Fuente: <https://www.conaset.cl/wp-content/uploads/2021/05/Costos-accidentes-2020.pdf>.

Las consecuencias que se producen en un accidente vehicular pueden ser catalogadas de la siguiente forma:

- a) Daños a la propiedad pública o privada, financiamiento médico y el costo administrativo sea privado o público.
- b) Daño indirecto: se refiere a los costos por pérdida de productividad asociada a las víctimas, como también al valor de bienes y servicios que habrían sido producidos de no suceder el accidente.
- c) Costo humano: pérdida de calidad de vida de las víctimas, agravios y perjuicios que afectan tanto a las víctimas como a sus familias.

1.1.3 Control de alcohol en la sangre

El test de alcohol en la sangre mide la cantidad de etanol en el cuerpo y usualmente se utiliza la métrica gramos sobre litros (g/l). El alcohol se absorbe rápidamente en la sangre y es posible de medir luego de diez minutos desde su ingesta y alcanza su nivel más alto aproximadamente una hora después del consumo. Como se aprecia en la tabla 1-2, la mínima cantidad de alcohol en la sangre puede causar juicio deficiente y una disminución general de la capacidad de reacción.

Tabla 1-2. Descripción de los efectos del alcohol en distintas cantidades.

Alcohol en la sangre (g/l)	Efectos colaterales en el cuerpo
0.02 - 0.3	Mayor tolerancia al riesgo, aumento del tiempo de reacción y disminución de la sensibilidad a la luz roja
0.31 - 0.79	Alteraciones en la toma de decisiones, impulsividad y agresividad, capacidad reducida para seguir objetos en movimiento y problemas para adaptar la visión a los cambios de luz
0.8 – o más	Alteraciones motoras y psicomotoras, incremento de la somnolencia, graves problemas perceptivos (visión doble, visión en túnel, etc.) y pérdida significativa de equilibrio

Fuente: Tabla creada por el autor en base a <https://www.sciencedirect.com/topics/immunology-and-microbiology/alcohol-blood-level>

La medición de concentración de alcohol en la sangre es calculada bajo la métrica de gramos sobre litro (g/l), a diferencia de la concentración que es detectada a través del aire espirado en dispositivos electrónicos, que utiliza miligramos sobre litros (mg/l) como métrica.

De acuerdo a la ley 20.580, comúnmente “ley de tolerancia cero”, se establece que la concentración límite de alcohol en la sangre vigente es de 0.3 g/l, como se expone a continuación en la figura 1-1.

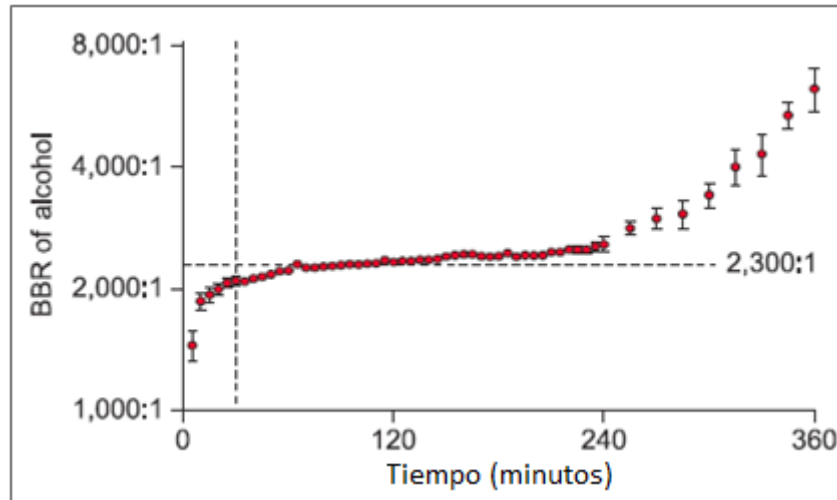
GRAMOS DE ALCOHOL POR LITRO DE SANGRE	ESTADO ETÍLICO	LESIÓN / DAÑO CAUSADO	REINCIDENCIA	TIEMPO SUSPENSIÓN LICENCIA
0,31 - 0,79	Bajo la influencia del alcohol	Sin daño ni lesiones	1ra vez	3 meses
		Daños materiales o lesiones leves		6 meses
		Lesiones menos graves		9 meses
		Lesiones graves		18 - 36 meses
		Lesiones gravísimas o muerte		36 - 60 meses
		—	2da vez	48 - 72 meses
0,8+	Estado de ebriedad o bajo la influencia de estupefacientes o sicotrópicos	Sin daño, daños materiales o lesiones leves	1ra vez	2 años
			2da vez	5 años
			3ra vez	cancelación
		Lesiones menos graves	1ra vez	36 meses
		Lesiones graves	1ra vez	5 años
		—	2da vez	cancelación
		Lesiones gravísimas	1ra vez	Inhabilidad de por vida y comiso del vehículo
		Muerte	1ra vez	

Fuente <https://www.conaset.cl/ley-tolerancia-cero/>

Figura 1-1. Tramos de concentración de alcohol en la sangre y sus consecuencias legales en un accidente.

La figura 1-2 muestra la relación entre concentraciones de alcohol medido en la sangre y en el aliento de doce sujetos de prueba. En esta investigación (Jones, 2010), se le suministran dosis de alcohol a los sujetos y se monitorea la concentración del alcohol intravenosa como la proveniente de la boca, en un periodo de 360 minutos. La concentración de alcohol en el aliento es aproximadamente 2000 veces menor que la concentración de sangre en un mismo volumen.

Esta relación de alcohol se pensó originalmente en 2100:1 cuando se aprobó el uso del alcoholímetro en los EE. UU. para estimar el nivel de alcohol en la sangre (Borkenstein y Smith, 1961). En la gran mayoría de los países, las proporciones pueden variar entre 2000:1 y 3000:1, dependiendo del equipamiento utilizado.



Fuente: https://www.researchgate.net/figure/Concentration-time-profiles-of-alcohol-in-venous-blood-and-breath-in-a-controlled-study_fig2_266580842

Figura 1-2. Relación entre alcohol en la sangre y el alcohol medido en el aire en el transcurso de seis horas.

1.2 SENSOR DE GASES MQ3

El sensor MQ3 es uno de los más utilizados en la serie de sensores MQ. Este sensor es del tipo MOS (Semiconductor de óxido metal). El funcionamiento de este sensor se basa en la variación de su resistencia interna al estar en presencia de un gas.

1.2.1 De las características técnicas

Debido a las propiedades del sensor, es necesario esperar un tiempo de calentamiento para que la salida sea estable y tenga las características que el fabricante muestra en la hoja de datos. Dicho tiempo, dependiendo del modelo, puede ser variable entre 12 y 48 horas. Este calefactor puede necesitar un voltaje entre dos y cinco voltios dependiendo del modelo.

En la tabla 1-3 se aprecia que el módulo MQ3 requiere de una tensión de 5 volts entre los pines VCC y GND.

Tabla 1-3. Condiciones de trabajo estandar del sensor MQ3.

Símbolo	Características	Valor
V_C	Tensión de funcionamiento	$5V \pm 0.1$
V_H	Tensión del calefactor	$5V \pm 0.1$
R_L	Resistencia de carga	200 K Ω
R_H	Resistencia del calefactor	$33\Omega \pm 5\%$
P_H	Consumo de calefacción	< 750 mW

Fuente: Tabla construída por el autor a partir del datasheet del sensor MQ3.

En la tabla 1-4 se muestra que el rango de valores que toma la resistencia interna del sensor MQ3 varía entre 1 M Ω y 8 M Ω . Además, R_S cuenta con una tasa de concentración con una pendiente equivalente a 0.4 mg de alcohol por cada litro de volumen.

Tabla 1-4. Características de la sensibilidad del sensor MQ3 en alcohol.

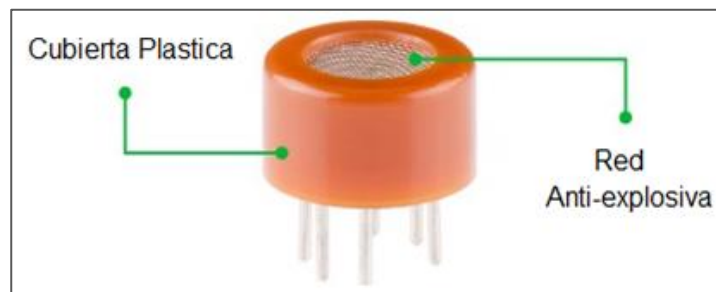
Símbolo	Nombre del parámetro	Parámetro técnico	Observaciones
R_S	Resistencia de detección	1M Ω - 8 M Ω (0.4mg/L alcohol)	Rango de la concentración de detección para alcohol: de 0.05 mg/L hasta 10 mg/L
α (0.4 mg/L)	Pendiente de tasa de concentración	≤ 0.6	
Condiciones referenciales de detección	Temp: 20°C $\pm$ 2°C V_C : 5V $\pm$ 0.1 Humedad: 65% $\pm$ 5% V_H : 5V $\pm$ 0.1		
Tiempo de precalentamiento	> 24 hrs		

Fuente: Tabla construída por el autor a partir del datasheet del sensor MQ3.

Este sensor tiene un rango de detección que va desde 0.05 mg/L hasta 10 mg/L de alcohol, por lo que cantidades que estén fuera de este rango no serán medidas.

1.2.2 De la estructura interna del sensor.

El funcionamiento del sensor MQ3 necesita una temperatura de regulación realizada por un calefactor interno, por lo que el sensor está cubierto con dos capas de malla fina de acero inoxidable como se aprecia en la figura 1-3. Esta red metálica asegura que el elemento calentador dentro del sensor no provoque una explosión cuando esté detectando un gas de tipo inflamable.

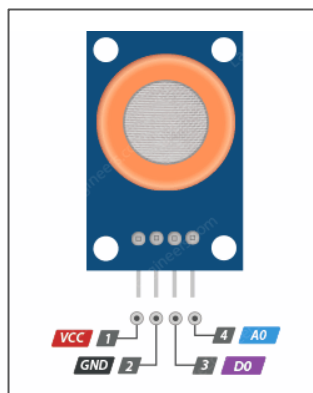


Fuente: <https://lastminuteengineers.com/mq3-alcohol-sensor-arduino-tutorial/>

Figura 1-3 . Imagen del sensor MQ3.

1.2.3 Del funcionamiento del módulo MQ3

El módulo MQ3 posee dos salidas: “A0” (analógica) y “D0” (digital), y dos pines de polarización: “VCC” (tensión de encendido) y “GND” (tierra), como se aprecia en la figura 1-4.

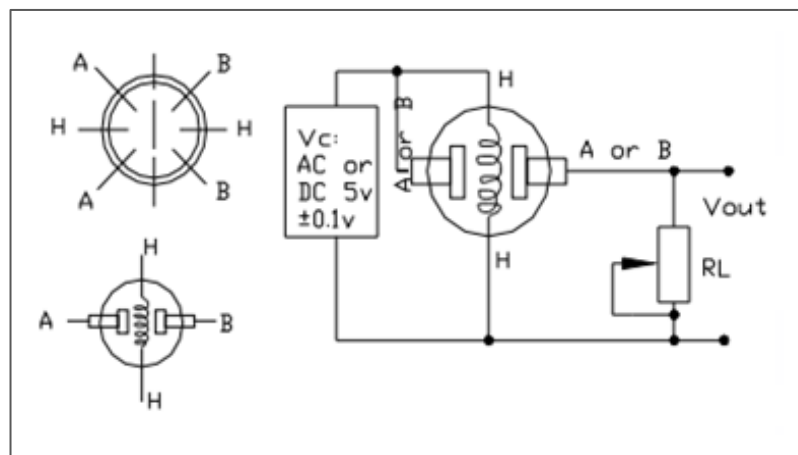


Fuente: <https://lastminuteengineers.com/mq3-alcohol-sensor-arduino-tutorial/>

Figura 1-4 . Esquema de pines del módulo MQ3.

El voltaje de la salida analógica del sensor varía en proporción a la concentración de alcohol. Cuanto mayor sea la concentración de alcohol en el aire, mayor será el voltaje de salida, por ende, cuando la concentración sea menor, menor será también el voltaje de salida. Esta salida será la encargada de enviar la información de tipo analógica hacia un microprocesador que pueda recibir este tipo de datos, y luego convertirla en un dato digital.

El sensor se comporta como una resistencia y utiliza una resistencia de carga (R_L) para cerrar el circuito, como se puede ver en la figura 1-5.



Fuente: <https://www.sparkfun.com/datasheets/Sensors/MQ-3.pdf>

Figura 1-5 . Circuito polarizador de sensor MQ3.

1.3 INTELIGENCIA ARTIFICIAL

1.3.1 De los tipos de inteligencia

Dado que la investigación de la Inteligencia Artificial (IA) pretende hacer que las máquinas emulen un funcionamiento similar al humano, las IA se pueden clasificar dependiendo del grado en que un sistema artificial puede replicar las capacidades humanas.

1.3.1.1 Máquinas reactivas

Una IA que pueda realizar funciones con niveles de competencia equivalentes, se considera más evolucionada, mientras que una IA que tenga una funcionalidad y un rendimiento limitados se considera un tipo más simple y menos evolucionado.

Según este sistema de clasificación, se pueden describir cuatro tipos o sistemas basados en IA: máquinas reactivas, máquinas de memoria limitada, teoría de la mente e IA autoconsciente.

1.3.1.2 Máquinas reactivas

Se trata del primer sistema de IA, por lo que, resulta ser su forma más básica. Posee una capacidad limitada y solo responde determinados tipos de estímulos. Estas máquinas son de gran utilidad cuando deben responder automáticamente a un conjunto limitado o una combinación de entradas. No tienen la capacidad de utilizar una memoria, por lo que, no pueden mejorar su operatividad basándose en esta propiedad.

Aunque tiene un alcance limitado y no se modifica fácilmente, la inteligencia artificial de la máquina reactiva puede alcanzar un nivel de complejidad y ofrece confiabilidad cuando se crea para realizar tareas repetibles. Un ejemplo popular de una máquina de IA reactiva es Deep Blue de IBM, una máquina que venció al gran maestro de ajedrez Garry Kasparov en 1997.

1.3.1.3 Memoria Limitada

La inteligencia artificial de memoria limitada tiene la capacidad de almacenar datos y realizar predicciones con la información recopilada anteriormente, sopesando decisiones potenciales. Entre los procesos que utiliza la inteligencia artificial de memoria limitada, el aprendizaje automático para desarrollarse debe realizar los siguientes pasos: creación de los datos de entrenamiento, elección del modelo de aprendizaje automático, el modelo elegido debe tener la capacidad de realizar predicciones, así como poder recibir comentarios humanos o ambientales. Esta retroalimentación debe almacenarse en forma de datos, para que así se genere un ciclo que posteriormente elaborará predicciones.

En otras palabras, este tipo de IA busca en el pasado pistas sobre lo que vendrá después, por lo que su complejidad es evidentemente mayor respecto de las máquinas reactivas.

Existen tres modelos principales de aprendizaje automático que utilizan la inteligencia artificial de memoria limitada, los que son:

- Aprendizaje por refuerzo, aquél que aprende a hacer mejores predicciones mediante la repetición de pruebas y errores.
- Memoria de corto plazo o largo plazo, aquella que utiliza datos anteriores para ayudar a predecir el siguiente elemento de una secuencia. Estos sistemas elaboran predicciones a partir de la información más reciente, perdiendo relevancia los datos de mayor antigüedad, aunque aún podrían ser utilizados para formar conclusiones.
- Redes Adversarias Generativas Evolutivas (E-GAN), se trata de aquellas que evolucionan con el tiempo, crecen para explorar caminos ligeramente modificados basados en experiencias previas con cada nueva decisión.

Una IA de reconocimiento de imágenes es ejemplo de lo anterior y consiste en aquella que se entrena utilizando miles de imágenes y sus etiquetas para enseñarle a nombrar los objetos que escanea. Cuando una imagen es escaneada por una IA de este tipo, utiliza las imágenes de entrenamiento como referencias para comprender el contenido de la imagen que se le presenta y, basándose en su “experiencia de aprendizaje”, etiqueta las nuevas imágenes con una precisión cada vez mayor. Casi todas las aplicaciones de IA actuales, desde chatbots y asistentes virtuales hasta vehículos autónomos, están todas impulsadas por IA de memoria limitada.

1.3.1.4 Teoría de la mente

Esto significaría que la inteligencia artificial deberá atribuir emociones a los humanos, animales y otras máquinas, con el propósito de tomar decisiones a través de la autorreflexión y la determinación.

El concepto se basa en la teoría psicológica que explica la capacidad humana de comprender que otras personas también tienen pensamientos y emociones que afectan su comportamiento.

1.3.1.5 Autoconsciente

Esta inteligencia es hipotéticamente la última etapa de desarrollo artificial. Este tipo de inteligencia no solo podrá comprender y transmitir emociones con quienes interactúa, también tendrá emociones, necesidades, creencias y potencialmente deseos propios.

El sistema alternativo de clasificación que se usa más generalmente en el lenguaje tecnológico es la clasificación de la tecnología en Inteligencia Artificial Estrecha (ANI), Inteligencia Artificial General (AGI) y Superinteligencia Artificial (ASI).

1.3.2 De la detección facial

La detección facial es la base para realizar cualquier tipo de análisis o algoritmo facial, incluyendo alineación del rostro, modelaje de caras, reiluminación del rostro opacado, verificación o autenticación, rastreo y seguimiento de la posición del rostro, seguimiento y reconocimiento de expresiones faciales, reconocimiento de género y edad, entre otras aplicaciones. A partir del reconocimiento facial inicial, el objetivo de la detección facial es determinar si existe o no, un rostro reconocible en la imagen presente, y si lo hay, se devolverá una imagen con todos los puntos localizados en un rostro.

La dificultad de esta tarea está asociada a que la detección puede ser atribuida a múltiples variables. Por ende, la apariencia del rostro varía directamente de acuerdo con los siguientes factores:

- De escala o de locación
- De orientación dentro del plano

También influyen factores que afecten la calidad la imagen, tales como:

- Iluminación (espectros, distribución e intensidad de la fuente),
- Características de la cámara (respuesta del sensor, control de ganancia, lente utilizado, distancia focal)
- Resolución de imagen

Y finalmente, en la detección de la cara los factores que influirán son:

- La presencia o ausencia de características particulares o accesorios presentes en un rostro (bigotes, barba, lentes, accesorios, gorros o sombreros)
- Oclusiones, es decir, situaciones en que el rostro está parcialmente oculto por objetos externos
- Expresiones faciales

Todas estas problemáticas deben considerarse ya que son factores relacionados directamente con la apariencia del rostro a detectar.

1.4 REDES NEURONALES ARTIFICIALES

Artificial Neural Network (ANN) o Redes Neuronales Artificiales en su traducción, son modelos computacionales evolucionados a partir de diversas aportaciones científicas registradas en la historia. Estas redes comprenden un conjunto de unidades, llamadas neuronas artificiales, conectadas y con la capacidad de transmitir señales entre sí.

Para una comprensión del funcionamiento de las Redes Neuronales Artificiales, se requiere puntualizar en los siguientes conceptos claves referidos a las neuronas artificiales. Las neuronas artificiales son modelos que tratan de simular el comportamiento de las neuronas biológicas, en donde cada neurona se representa como una unidad de proceso que forma parte de una entidad mayor, la red neuronal.

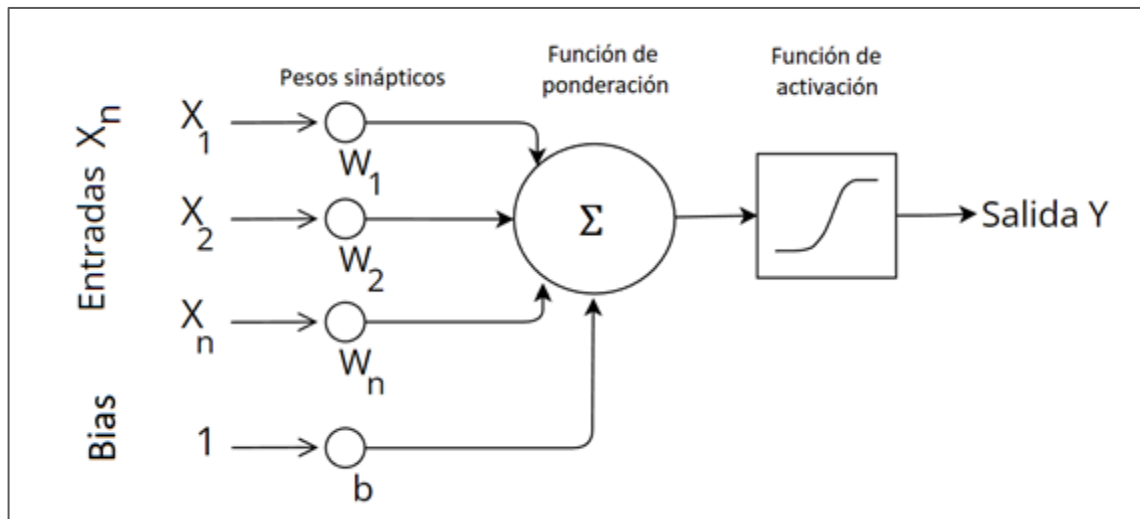
- A. Weights (Pesos Sinápticos): Cada neurona esta conectada a otras, a través de enlaces que contienen información llamada peso sináptico y datos de las señales de entrada. El valor de la salida de la neurona depende unicamente de los pesos y la información de las señales de entrada.
- B. Bias (Sesgo): Representa un valor constante que es añadido al total de la ponderación de las entradas y los pesos sinápticos que llegan a la neurona. Se utiliza para cambiar el resultado a un lado positivo o negativo. El peso neto de la entrada se incrementa cuando se añade un sesgo positivo, o al contrario, disminuye cuando es un sesgo de valor negativo.
- C. Learning Rate (Tasa de Aprendizaje): Esta variable toma valores de cero a uno, y es usado para balancear los pesos sinápticos durante el proceso de aprendizaje.
- D. Weighting Function (Función de Ponderación o Propagación): Esta función se encarga de transformar las diferentes entradas que provienen de la sinapsis en el potencial de la neurona. Normalmente se usa como función de propagación la suma ponderada de las entradas multiplicadas por los pesos. En esta función se interpreta como un regulador de las señales que se emiten entre neuronas al ponderar las salidas que entran a la neurona.
- E. Activation Function (Función de Activación): Esta función combina el potencial postsináptico, que proporciona la función de propagación, con el estado actual de la neurona para conseguir el estado futuro de activación de la neurona. Dentro de las funciones más utilizadas están: las funciones lineales, de fácil uso y que pueden alcanzar un alto nivel de eficiencia; las funciones de escalón, usada para redes neuronales binarias como el Perceptrón Simple y Hopfield; y por último, las funciones tangenciales o hiperbólicas, que se utilizan en redes como el Perceptrón Multicapa con retropropagación, debido a que su algoritmo de aprendizaje requiere una función derivable.
- F. Target (Valor Objetivo): Es el valor correcto de la variable de salida.

G. Error : Es la imprecisión de la salida en comparación a los valores objetivos.

H. Treshold (Umbral): Se representa por un valor constante y se compara con la entrada neta para obtener la salida, en función de esta comparación se define la función de activación.

Las redes de neuronas se organizan en distintas capas, que pueden realizar diferentes tipos de transformaciones a sus entradas. Por sí solo, un solo elemento de procesamiento no es muy poderoso ni eficaz, por lo que estas se agrupan en estructuras más complejas.

La figura 1-6 muestra una unidad de proceso o neurona artificial, que se representa por una serie de entradas " X_n " que son ponderadas con pesos " W_n ". Estos valores son llevados junto con el sesgo a la entrada de la función de ponderación, reduciendo todos los valores a uno solo. Finalmente este valor ingresa a la función de activación para poder determinar si la neurona se activa o no.



Fuente: Diagrama creado por autor.

Figura 1-6. Imagen simplificada de una neurona artificial.

1.4.1 Clasificación según topología o estructura de la red

Las redes neuronales pueden ser clasificadas según la estructura que poseen, como la cantidad de capas, con una sola o varias; o también según la direccionalidad de las conexiones de las neuronas, hacia adelante o hacia atrás.

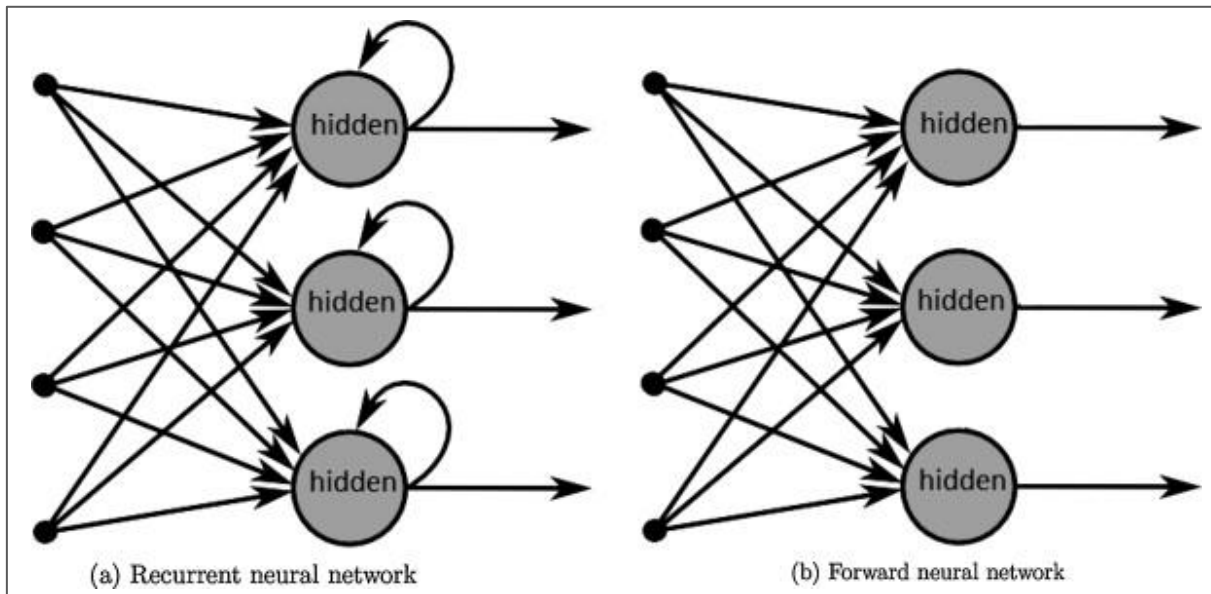
1.4.1.1 De la estructura monocapa

Las redes monocapa son redes con una sola capa. Para unirse las neuronas crean conexiones laterales con otras neuronas de su capa. Las redes monocapa han sido ampliamente utilizada en circuitos eléctricos ya que debido a su topología, son adecuadas para ser implementadas mediante hardware, usando matrices de diodos que representan las conexiones de las neuronas. Dentro de las redes más representativas existentes estan: la red de Hopfield, la red BRAIN-STATE-IN-A-BOX y las maquinas estocásticas de Botzmann y Cauchy.

Entre las redes neuronales monocapa, existen algunas que permiten que las neuronas tengan conexiones a si mismas y se denominan autorecurrentes. Una red recurrente significa que el estado de excitación de una neurona no depende tan sólo de los impulsos sinápticos que recibe de otras células, sino también de sus propios estados anteriores, lo que constituye una forma de memoria. Este tipo de redes neuronales se utilizan en el procesamiento natural del lenguaje.

1.4.1.2 De la estructura multicapa

Las redes multicapa están formadas por dos o mas capas de neuronas, usualmente las capas realizan conexiones dependiendo del orden en que se reciben las señales, desde la entrada hacia la salida, conservando el orden de llegada. Desde este punto de vista, las redes multicapa se pueden clasificar en redes alimentadas hacia adelante (Feed-Foward) y redes con retroalimentación (Feed-Back) como se aprecia en la figura 1-7. -



<https://datascience.stackexchange.com/questions/63079/what-is-difference-between-feed-forward-neural-network-and-lstm>

Figura 1-7. Imagen referencial de una capa de neuronas con conexión:

A. Recurrente. B. Hacia adelante.

A. Redes con conexiones hacia adelante:

Este tipo de redes solo contienen conexiones entre capas hacia adelante, o sea, la información fluye unidireccionalmente desde las neuronas de entrada hasta las neuronas de salida, sin que sea posible la creación de bucles, implicando que una capa no puede realizar conexiones con otra, mientras no reciba la señal antes.

Un ejemplo de esta red es el Perceptrón Multicapas, en que las neuronas se estructuran en capas sucesivas, conformada por una capa inicial de entrada, capas intermedias encargadas del procesamiento de la información, y una capa de salida.

B. Redes con conexiones hacia atrás:

Este tipo de redes se caracteriza por poseer mas de una capa y se diferencia de las redes anteriores, en la existencia de conexiones hacia atrás, por lo tanto pueden intercambiar información recíprocamente con diferentes capas de la red a través de bucles de retroalimentación, tal como sucede en los cerebros biológicos.

1.4.2 Clasificación según su algoritmo de aprendizaje

El aprendizaje de las redes neuronales consiste en el proceso de entrenamiento de la red con los patrones de muestra, con el objetivo de modificar los pesos sinápticos de las conexiones utilizando una regla de aprendizaje. Esta regla consiste en algoritmos matemáticos, fórmulas de optimización o minimizadoras del error, que modifican el peso de las neuronas en función de las entradas ingresadas a la red. Este algoritmo usualmente se ejecuta iterativamente, hasta que los pesos convergen a un valor óptimo o deseado.

Se pueden distinguir tres tipos de aprendizaje de las redes neuronales. Primero, el aprendizaje supervisado, que consiste en que la red dispone de los patrones de entrada y los patrones de salida deseados, y en función de ellos, se modifican los pesos de las sinapsis para ajustar la entrada a esa salida.

Por otro lado, el aprendizaje no supervisado no necesita de la entrega de patrones objetivos, solo de patrones de entrada, y la red realiza la clasificación de dichos patrones en función de las características comunes que posee la información.

Y por último, en una fórmula híbrida, el aprendizaje por reforzamiento posee un supervisor que no enseña patrones objetivos a la red, si no que solo califica los resultados de la red ante un patrón de entrada.

1.4.2.1 Algoritmos de aprendizaje supervisado

El aprendizaje supervisado usualmente requiere de una interacción humana que realice la etiquetación de grandes conjuntos de datos. Este conjunto de datos incluye datos de entrada y resultados correctos, que permiten que el modelo aprenda con el tiempo. El algoritmo mide su precisión a través de la función de pérdida, ajustándose hasta que el error se haya minimizado lo suficiente.

Estos métodos cuentan con dos herramientas de entrenamiento. Primero, la clasificación, que utiliza algoritmos para asignar con precisión datos de prueba en categorías específicas. Dentro de los algoritmos de clasificación más comunes se encuentran clasificadores lineales, máquinas de vectores de soporte (SVM), árboles de decisión, vecinos K más cercanos (KNN) y bosques aleatorios.

La segunda herramienta es Backpropagation o retropropagación, que se utiliza para comprender la relación entre variables dependiente e independientes. El algoritmo de retropropagación es responsable de actualizar los pesos sinápticos con el objetivo de reducir el error de la red. Dentro de los algoritmos de regresión más populares están la regresión lineal, regresión logística y regresión polinomial.

La retropropagación implica el cálculo del gradiente que avanza hacia atrás, a través de la red de alimentación, desde la última capa hasta la primera. Para calcular el gradiente en una capa en particular, los gradientes de todas las capas siguientes se combinan mediante la regla de la cadena del cálculo. Esto permite que cada peso se actualice individualmente para reducir gradualmente la función de pérdida durante muchas iteraciones de entrenamiento.

Las funciones de error indican qué tan cerca están las salidas predichas de las salidas deseadas, y su valor óptimo es cero. Una de las funciones de error es la función de error al cuadrado. La red finaliza su entrenamiento cuando los resultados deseados son idénticos a los previstos.

1.4.2.2 Algoritmos de aprendizaje sin supervisión

El aprendizaje no supervisado es un tipo de aprendizaje automático en el que un modelo debe buscar patrones en un conjunto de datos sin etiquetas y con supervisión humana mínima. Los modelos de aprendizaje no supervisados se utilizan para cuatro tareas principales: agrupación, asociación y reducción de dimensionalidad.

- A. Agrupación: La agrupación es una técnica de minería de datos que se utiliza para encontrar los patrones de relación y/o similitud entre las muestras de datos de entrada y que luego se agrupan en función de estos patrones. Los algoritmos de agrupación en clústeres se utilizan para procesar objetos de datos brutos y sin clasificar en grupos representados por estructuras o patrones en la información.
- B. Asociación: Una regla de asociación es un método basado en reglas para encontrar relaciones entre variables en un conjunto de datos determinado.
- C. Reducción de dimensionalidad: Es una técnica que se utiliza cuando el número de características o dimensiones de un conjunto de datos determinado es demasiado alto. Reduce la cantidad de entradas de datos a un tamaño manejable y al mismo tiempo preserva la integridad del conjunto de datos tanto como sea posible. Se usa comúnmente en la etapa de preprocesamiento de datos. El Análisis de Componentes Principales (PCA) es uno de los métodos de reducción de dimensionalidad que utiliza una transformación lineal para crear una nueva representación de datos, produciendo componentes principales, o sea, un conjunto de datos con las características principales.

1.4.2.3 Por refuerzo

Este tipo de aprendizaje se basa en la noción de condicionamiento por refuerzo, es decir, se aprenden las conductas reforzadas positivamente y se descartan las conductas reforzadas negativamente. Esto se traduce que los pesos sinápticos se premian cuando se acierta la salida y se penalizan cuando no. Se pueden nombrar seis variables características de este aprendizaje:

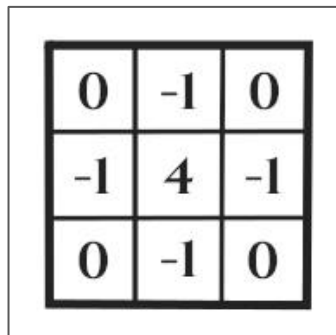
- A. Ambiente: Es el entorno donde el agente interactúa. Contiene todos los elementos que constituye el estado, y en donde se establecen las reglas y limitaciones.
- B. Estado: Es la situación actual del ambiente, teniendo en cuenta todos sus elementos y variables. En pocas palabras, es un indicador actual del ambiente.
- C. Agente: Es la máquina o el modelo al cual se le aplica inteligencia por refuerzo. Es el autor del escenario y quien toma las decisiones con autonomía.

- D. Acciones: Son las posibles decisiones que puede tomar el agente en diversas circunstancias.
- E. Recompensas: Son los premios que se le otorgan al agente por acertar o tomar el camino correcto.
- F. Penalizaciones: Son los castigos que se le otorgan al agente por fallar o tomar el camino incorrecto.

1.4.3 De la red neuronal convolucional

Una red neuronal convolucional se mecaniza de tal forma que busca características en un objeto. Esta red extrae o identifica una característica en una imagen en particular. Esta habilidad es una de las operaciones más fundamentales en el aprendizaje automático y se usa ampliamente como modelo base en la mayoría de las redes neuronales como GoogleNet, VGG19 y otras, para diversas tareas como la detección de objetos y la clasificación de imágenes.

La red se estructura en capas de convolución y capas de agrupación de forma alternada. A cada una de estas capas le corresponde un tipo de neurona diferente, que comparten un mismo mapa de características, número de conexiones y conjunto de pesos sinápticos, al que se le llama kernel. Este detector de características corresponde matemáticamente a una pequeña matriz de dimensiones $n * m$ que se utiliza como filtro. Por ejemplo, el filtro de nitidez viene constituido por la matriz que se ve en la figura 1-8. Este filtro resalta el pixel central, y atenúa los que lo rodean.

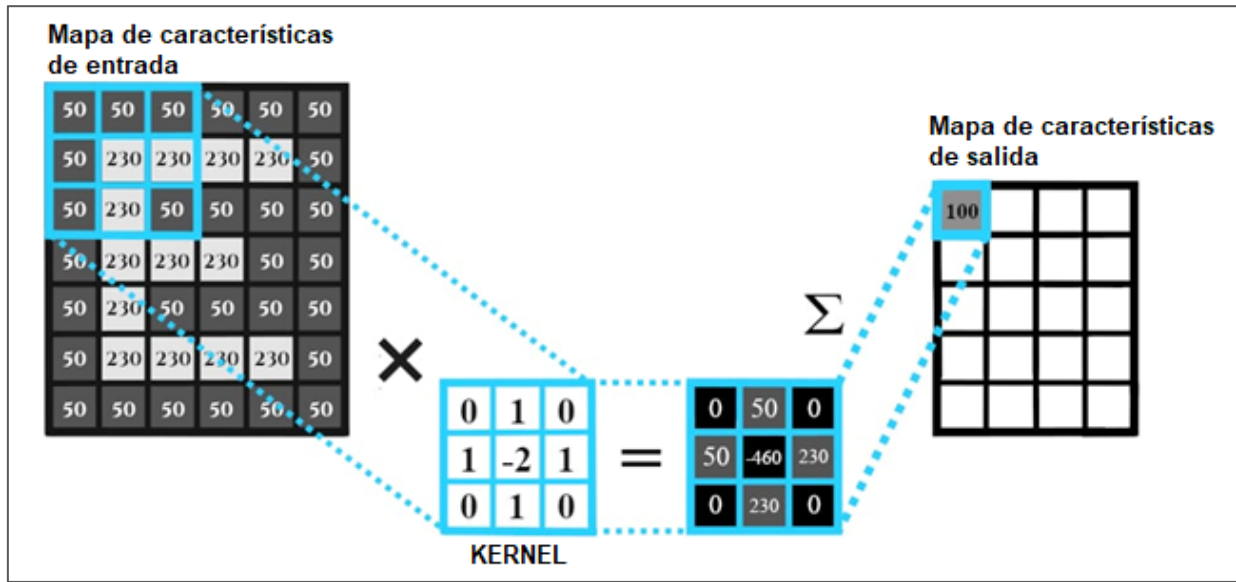


0	-1	0
-1	4	-1
0	-1	0

Fuente: <https://lamaquinaoraculo.com/computacion/redes-neuronales-convolucionales/>

Figura 1-8. Imagen de un filtro con dimensiones 3x3.

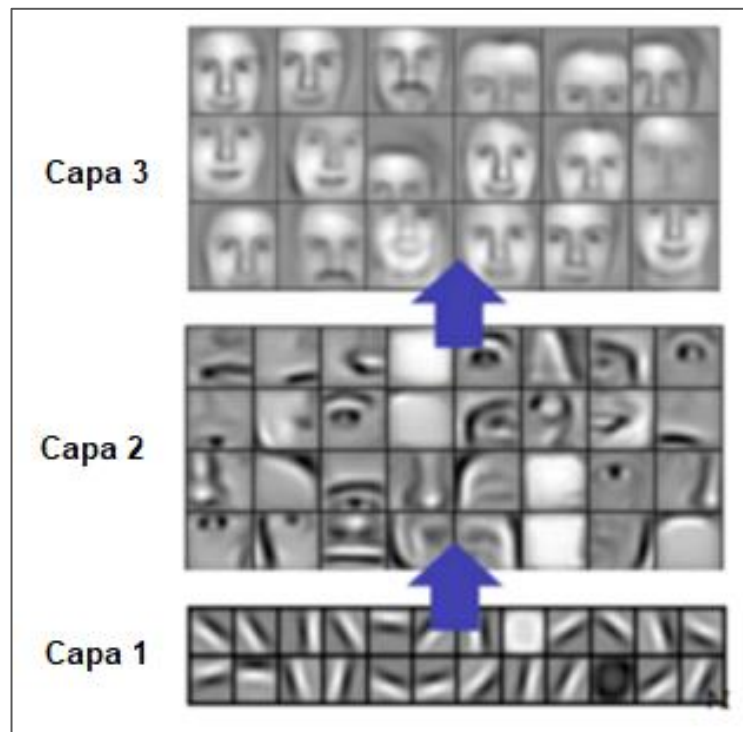
La figura 1-9 muestra como se aplica la convolución, donde el filtro se sobrepone en una porción de los datos de entrada también con dimensiones $n * m$, y se va desplazando a través de la totalidad de imagen de entrada, multiplicando la información del pixel con el coeficiente del filtro, repitiéndose así hasta cubrir todo el espacio.



Fuente: <https://lamaquinaoraculo.com/computacion/redes-neuronales-convolucionales/>

Figura 1-9. Imagen del uso de un filtro sobre un mapa de características.

Cada una de las capas se compone a su vez de un cierto número de mapas de características. Las primeras se encargan de extraer patrones dentro de los datos de entrada, luego utilizando los bordes correspondientes, se reconocen formas simples como se muestra en la figura 1-10.



Fuente: <http://incredible.ai/artificial-intelligence/2016/06/12/Convolutional-Neural-Networks-Part1/#non-linearity-relu>

Figura 1-10. Construcción de patrones mediante capas

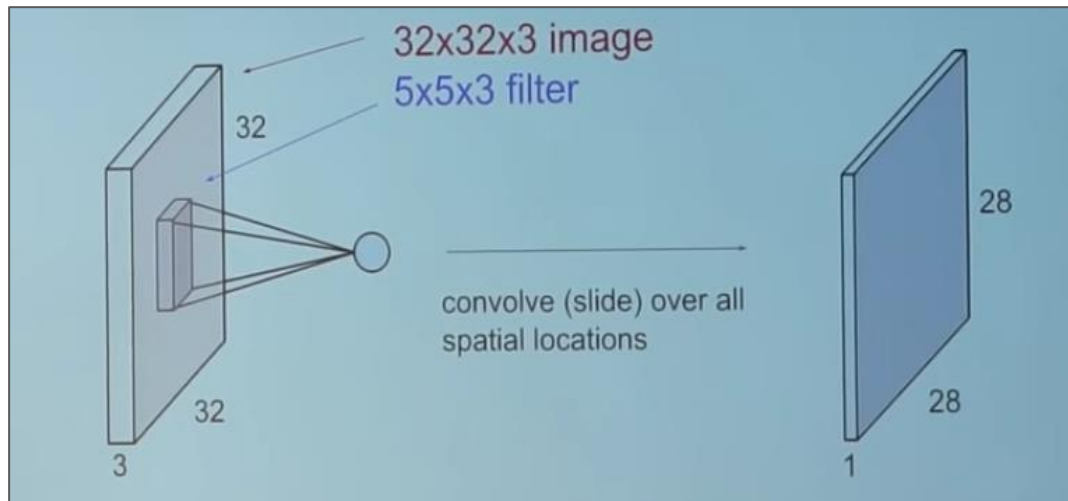
Algunas líneas o contornos se reconocen en las capas iniciales, pero las formas complejas, como las caras, se reconocen a medida que avanzan las capas. En última instancia, las redes convolucionales logran aprender automáticamente los valores de los filtros.

Una red convolucional está compuesta básicamente por una combinación de los siguientes cinco componentes:

- I. Convolución: para detectar características de una imagen
- II. ReLU: para suavizar la imagen y eliminar bordes
- III. Agrupación: para reducir resolución y mantener características
- IV. Aplastamiento: para convertir la imagen en una representación adecuada
- V. Conexión completa: para procesar los datos en una red neuronal

1.4.3.1 Convolución

Su función es conservar la relación entre píxeles, al aprender las características de la imagen utilizando pequeños cuadrados de datos de entrada. Esta operación matemática toma dos entradas, una matriz de origen y un filtro.

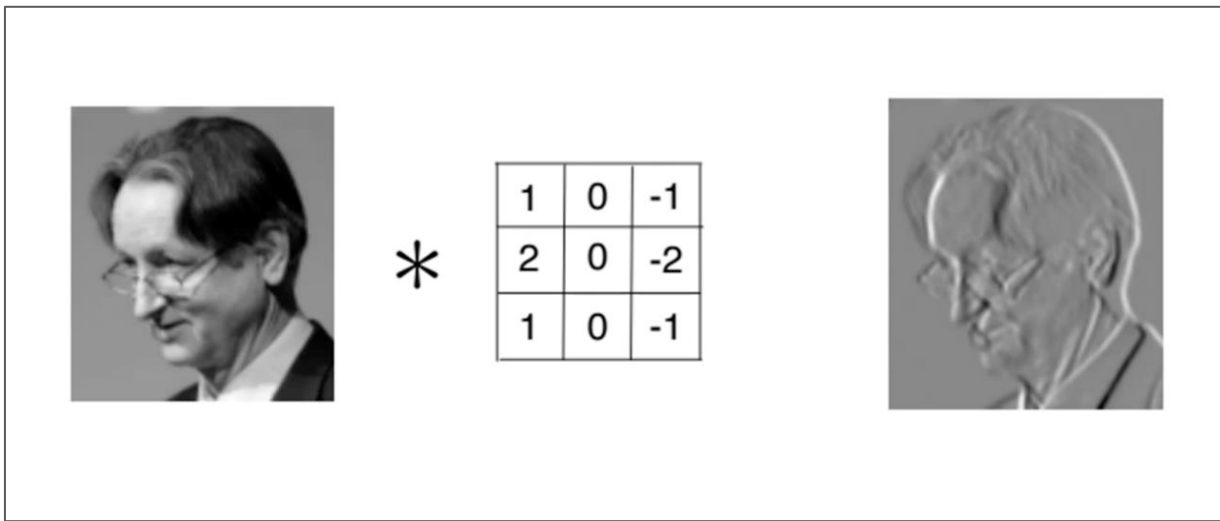


Fuente: https://iq-opengenius-org.translate.goog/content/images/2018/10/CNN.png?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419

Figura 1-11. Ilustración de una convolución a una imagen con dimensión 32 x 32 x 3 píxeles.

En la figura 1-11, se expone una imagen de dimensiones 32 x 32 x 3 píxeles, a la que se aplica un filtro convolucional de menor escala del área plana, pero comparte la dimensión en profundidad resultando en una imagen de profundidad singular.

Al utilizar un filtro, se extrae la información más relevante de la imagen y, por lo tanto, el modelo resultante solo se centrará en las características deseadas y no en la información innecesaria. Ciertos filtros específicos o detectores de características hacen que la imagen tenga diferentes efectos, como se aprecian en la figura 1-12.



Fuente: <https://iq.opengenus.org/content/images/2018/10/Screen-Shot-2018-10-11-at-10.38.45-PM.png>

Figura 1-12. Convolución de una imagen de un rostro.

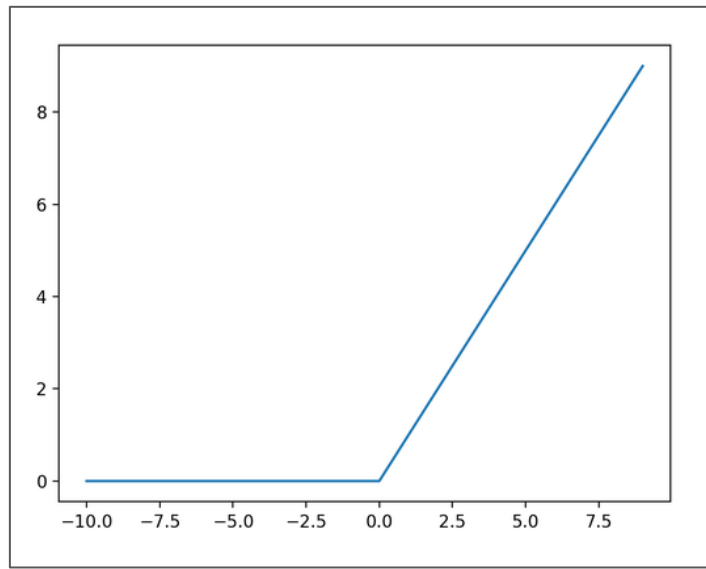
El resultado muestra una imagen con menos información, menos detallada, pero que conserva la información del relieve de la silueta. Existen filtros que se pueden utilizar para difuminar una imagen, detectar los bordes e incluso para la detección de figuras.

1.4.3.2 ReLU

Unidad Lineal Rectificada (ReLU) es una operación inteligente de elementos, por lo que se aplica píxel por píxel y convierte todos los píxeles negativos en valores cero. Esta función se define como:

$$f(x) = \max(0, x)$$

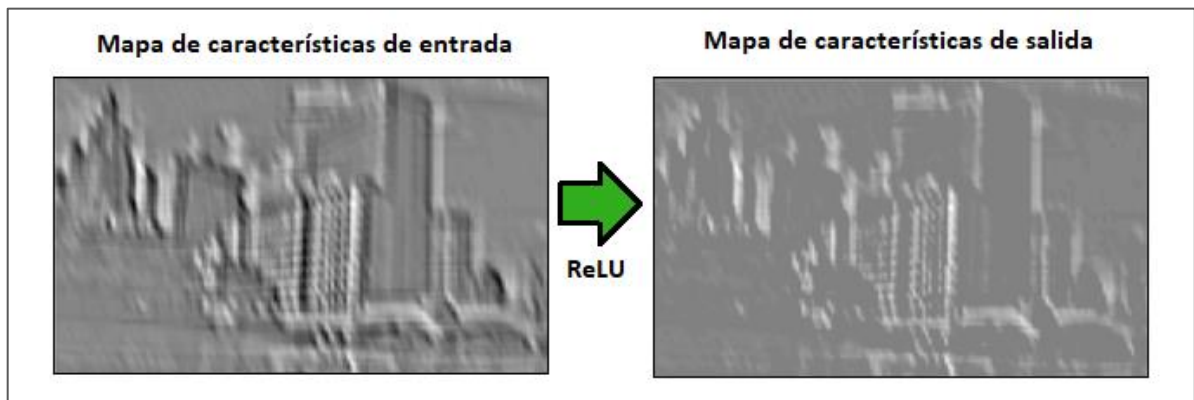
Generalmente una imagen tiene distintos valores por cada píxel, esto se traduce en que la información no tiene una linealidad. Para lograr disminuir estas irregularidades se agrega una función rectificadora, ayudando a romper el flujo del degradado de la imagen y resaltando cambios drásticos cuando hay características distintas. Lo que muestra la figura 1-13, es que la función ReLU transforma los valores introducidos anulando los valores negativos y manteniendo solo los positivos.



Fuente: <https://qph.fs.quoracdn.net/main-qimg-4598dba7124625aba22bc291cb2d326c>

Figura 1-13. Gráfica de la función ReLU.

Una red convolucional utiliza datos que son mayoritariamente no lineales, por lo que mediante la introducción de una función no lineal como ReLU, se obtiene un resultado no lineal.



Fuente: <http://incredible.ai/artificial-intelligence/2016/06/12/Convolutional-Neural-Networks-Part1>

Figura 1-14. Aplicación de la función ReLU a una mapa de características.

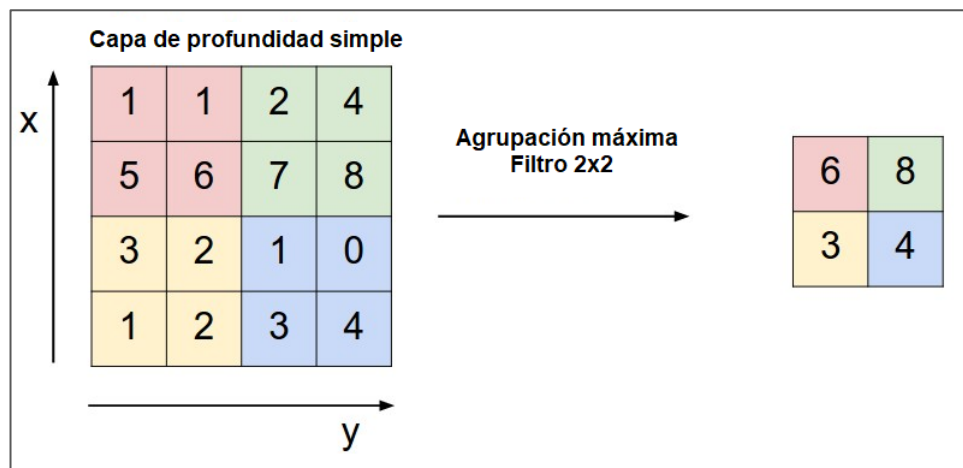
En la figura 1-14 se observa como una imagen de entrada con valores no lineales, con el color negro representando los valores negativos, y el blanco los positivos. Al aplicar la función rectificadora sobre el mapa de características se obtiene como resultado una imagen que conserva solo los valores positivos y elimina los valores negativos.

Además de ReLU, se pueden usar otras funciones no lineales como tanh o sigmoid, pero ReLU tiene el mejor rendimiento en redes convolucionales.

1.4.3.3 Agrupación

La agrupación asegura que, incluso si las características están un poco distorsionadas o no son muy similares, se cuenta con la flexibilidad para identificar la característica.

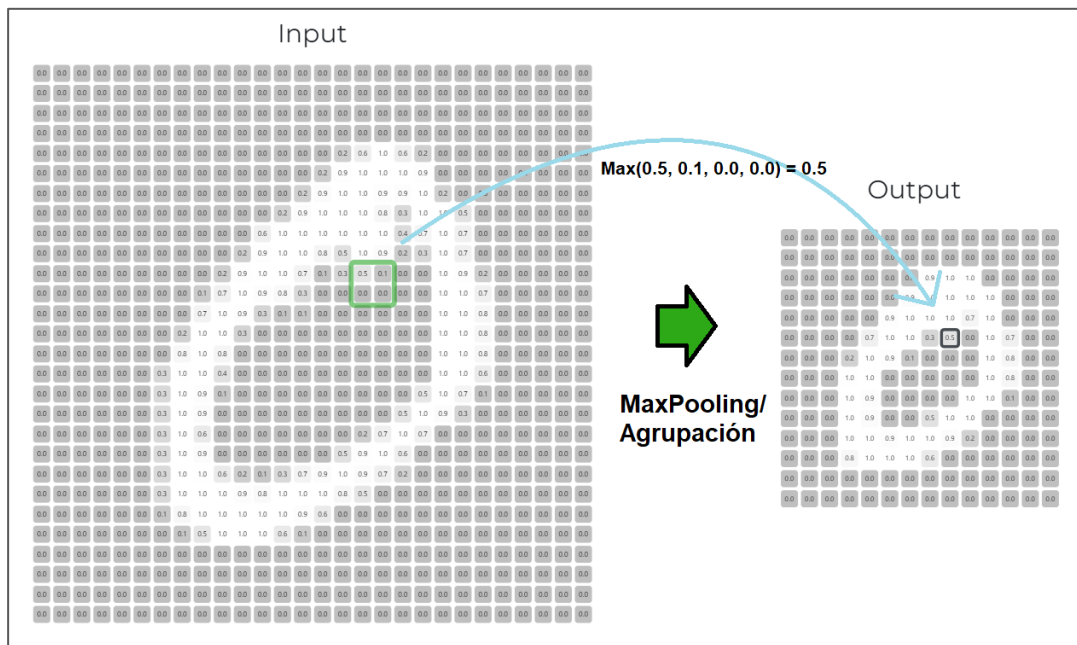
Su función es reducir la dimensionalidad de cada mapa de características y al mismo tiempo conservar la información más importante. La agrupación utiliza distintas funciones como promedio o suma, sin embargo, generalmente en la detección de figuras o patrones en imágenes se utiliza la operación de máximo. En la figura 1-15, se expone un ejemplo resumido de la función agrupación; en este caso, la matriz de píxeles es de 4 x 4 y al utilizarse el filtro 2 x 2 con agrupación por máximos, resulta una matriz de 2 x 2 que conserva el valor más alto de cada área por donde sobrepasa el filtro.



Fuente: <https://cs231n.github.io/assets/cnn/maxpool.jpeg>

Figura 1-15. Agrupación por máximos de una imagen simple.

En la figura 1-16 se muestra la aplicación de esta función sobre un mapa de características utilizando un filtro de maximos. El mapa resultante es una copia de la imagen entrante pero con la resolución reducida a la mitad.

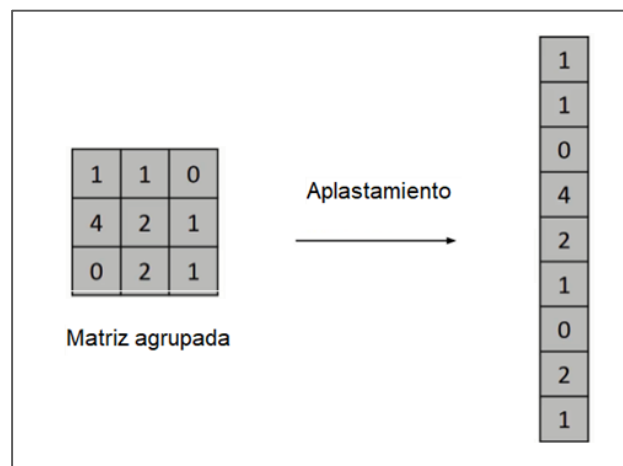


Fuente: Captura de pantalla en la aplicación interactiva del sitio <https://deeplizard.com/resource/pavq7noze3>

Figura 1-16. Aplicación de la función de agrupación a una mapa de características.

1.4.3.4 Aplastamiento

El aplastamiento transformará cada mapa de características agrupado en una forma vectorial. En la figura 1-17 se muestra como cada valor de la matriz se acopla en una columna de datos.



Fuente: https://sds-platform-private.s3-us-east-2.amazonaws.com/uploads/73_blog_image_1.png

Figura 1-17. Aplastamiento de una matriz agrupada simple con dimensiones 3x3.

Este simple pero muy importante paso de transformación permite que la información pueda tomar una arquitectura compatible con una ANN.

1.4.3.5 De la conexión completa

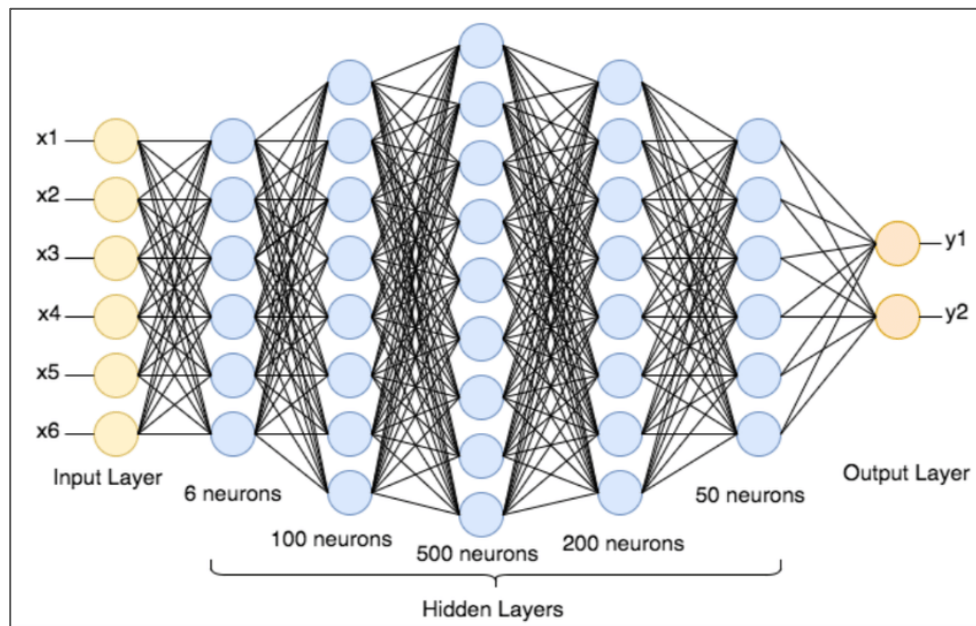
Finalmente, se asignan pesos aleatorios a cada sinapsis, la capa de entrada se ajusta por peso y se pone en una función de activación. El resultado de esto se compara luego con los valores reales y el error generado se propaga hacia atrás, es decir, los pesos se reajustan. Finalmente el proceso se repite hasta lograr un error mínimo y obtener un valor deseado.

Las CNN han mejorado significativamente a lo largo de los años en rendimiento y se utilizan en muchas aplicaciones de vanguardia (Parkhi, Vedaldi, Zisserman y otros en 2015; Amos, Ludwiczuk y Satyanarayanan en 2016). En el área de la computación visual, el factor más importante en el desempeño de una CNN es la cantidad de imágenes sobre las que puede entrenar. DeepFace implementó en su red neuronal 4,4 millones de imágenes y ha demostrado resultados provechosos. FaceNet es otra implementación de CNN que hace uso de uno de dos modelos diferentes que se entrenó en 200 millones y 3.3 millones de imágenes respectivamente. OpenFace es también una implementación de CNN que se ha capacitado en 500 mil imágenes.

1.4.4 Del aprendizaje en profundidad

El aprendizaje de profundidad son todos aquellos métodos de creación de modelos de predicción de información, usualmente utilizando redes neuronales de retroalimentación, que a su vez, están compuestas por capas ocultas, como se observa en la figura 1-18, mientras más capas ocultas tenga la red, mayor complejidad y precisión tendrá la respuesta. Por ejemplo, una red neuronal en sus primeras etapas puede aprender a detectar formas, para luego aprender a detectar ruedas, retrovisores, y puertas, para terminar aprendiendo a distinguir cada característica del automóvil.

Este sub campo del aprendizaje de máquinas está basado en la comprensión de distintas formas de representación, estas se ordenan en una jerarquía de elementos, factores o conceptos, teniendo como origen una variable general de nivel bajo.



Fuente: [https://www.researchgate.net/profile/Meriem-](https://www.researchgate.net/profile/Meriem-Bahi/publication/330120030/figure/fig1/AS:735637925797888@1552401157053/Deep-Neural-Network-architecture.ppm)

[Bahi/publication/330120030/figure/fig1/AS:735637925797888@1552401157053/Deep-Neural-Network-architecture.ppm](https://www.researchgate.net/profile/Meriem-Bahi/publication/330120030/figure/fig1/AS:735637925797888@1552401157053/Deep-Neural-Network-architecture.ppm)

Figura 1-18. Arquitectura de una red neuronal.

El aprendizaje en profundidad se encuentra en el estudio de modelos gráficos, optimización, reconocimiento de patrones y procesamiento de señales. El dramático incremento de la habilidad procesamiento de los chips GPU, el considerable aumento del tamaño de la información que es usada para el entrenamiento y los recientes avances en la tecnología de máquinas inteligentes, beneficia enormemente a que estos modelos puedan realizar funciones no lineales compuestas de alta complejidad, aprender a resolver problemáticas de distribución y ordenamiento jerárquico, y además, hacer posible el uso de tanto información etiquetada como no etiquetada.

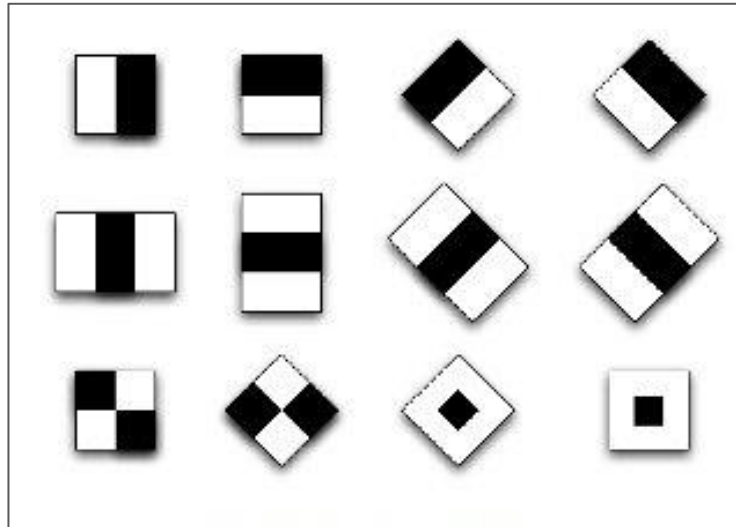
1.4.5 De los modelos y técnicas en la detección de caras humanas

Dentro de las múltiples técnicas de detección de características, la mayoría necesita de un hardware de alta potencia que logre realizar el análisis. Sin embargo, existen distintos modelos para la detección de rostros.

A continuación, se muestran métodos de mayor eficiencia, que logran tener alto índice de precisión en sus resultados.

1.4.5.1 Detección de objetos en cascada de Haar

La detección de objetos mediante clasificadores en cascada basados en funciones de Haar es un método eficaz de detección de objetos propuesto por Paul Viola y Michael Jones en su artículo “Detección rápida de objetos mediante una cascada potenciada de funciones simples” (2001).



Fuente: <https://unipython.com/wp-content/uploads/2018/01/Haarcascada1.jpg>

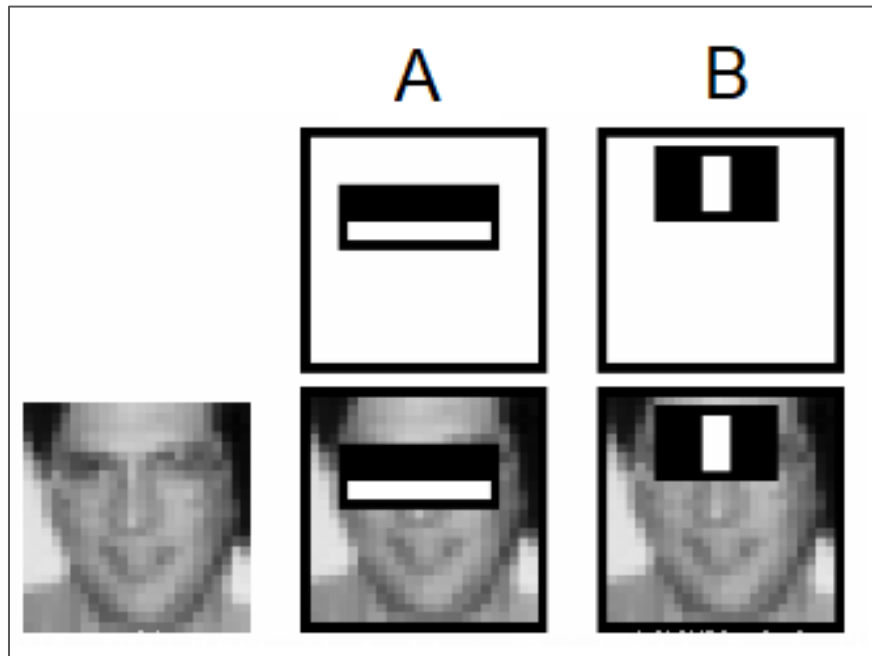
Figura 1-19. Características del Clasificador Haar.

Inicialmente el algoritmo necesita de la utilización de múltiples imágenes con caras (imágenes positivas) e imágenes sin caras (imágenes negativas). Estas imágenes conforman la base de datos para que el algoritmo realice el aprendizaje de características, tal como se aprecia en la figura 1-19.

Luego de que un clasificador está entrenado, puede ser aplicado a una región de interés en una imagen de entrada. El clasificador emite un símbolo, en este caso, un “1”, si en la región es probable que se encuentre la cara y un “0”, en caso contrario.

En la figura 1-20, se evidencia que la fila superior muestra dos características. La característica A se centra en la propiedad que la región de los ojos suele ser más oscura que la región de la nariz y las mejillas. La característica B se enfoca en la zona entre los ojos, que tiende a ser más sombreada, en relación al puente de la nariz, que posee una tonalidad más clara.

Estas simples generalizaciones son llamadas clasificaciones débiles, ya que por sí solas no pueden detectar un rostro, sin embargo, la suma de este tipo de funciones débiles construyen un clasificador de mayor complejidad, más fuerte y de mayor precisión.



Fuente: https://docs-opencv-org.translate.goog/3.4/db/d28/tutorial_cascade_classifier.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=wapp

Figura 1-20. Clasificador Haar sobre una imagen de un rostro.

El clasificador está diseñado de modo que pueda ser fácilmente “cambiar de tamaño” con el fin de ser capaz de encontrar los objetos de interés en diferentes tamaños, lo que es más eficiente que cambiar el tamaño de la imagen en sí.

El documento señala que incluso un mínimo de 200 funciones brindan una detección con precisión del 95%. Su configuración final tiene alrededor de 6000 características.

1.5 COMPUTADORAS DE TARJETA SIMPLE

Single Board Computer (SBC) o computadoras de tarjeta simple en su traducción, son pequeñas placas compuesta de circuitos integrados, con la capacidad de funcionar como una computadora más grande pero de menor potencia debido a su tamaño compacto.

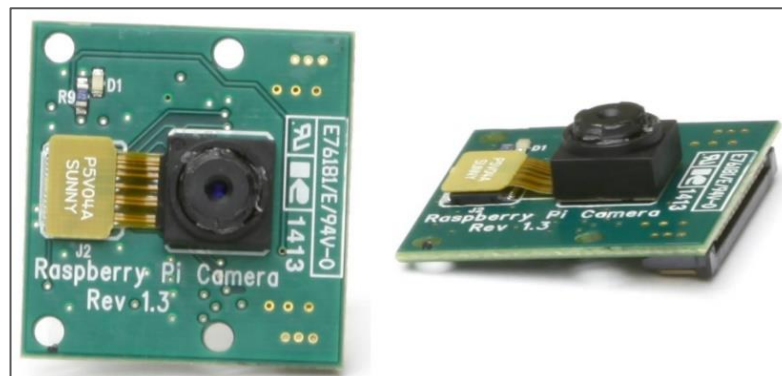
Debido a la configuración que poseen estas placas se reduce el costo total de un sistema, ya que al reducir la cantidad de placas requeridas, se eliminan los conectores y circuitos de controlador de bus, integrandose todas estas funciones en la misma placa. A continuación, se expone sobre la placa Raspberry Pi, un ordenador de bajo costo con la capacidad de funcionar como una computadora personal.

1.5.1 Raspberry Pi

La tarjeta Raspberry Pi 3B+ tiene como componente principal al System on Chip (SoC) Broadcom BCM2837 compuesto por un procesador ARMv8 de cuatro núcleos de 64 bits a una velocidad de 1.4GHz, GPU Broadcom VideoCore IV y memoria ram de 1GB. Incluye 4 puertos USB 2.0, conexión HDMI, Wifi doble banda, Bluetooth 4.2, y puerto ethernet.

Esta minimalista computadora cuenta con distintas funciones. Permite programar el uso de sus pines GPIO, que incluye también comunicación serial, SPI, I2C y UART. Estas funciones hacen que pueda ser empleado en proyectos de electrónica y robótica interactuando con sensores y actuadores, especialmente útil en aplicaciones con procesamiento de imágenes/vídeo, cámaras y cálculos matemáticos complejos.

La Fundación Raspberry Pi ofrece un modelo de cámara de fácil instalación. La figura 1-21 muestra la primera versión del módulo de cámara para esta placa y cuenta con una resolución de imagen de 5 megapíxeles.



Fuente: http://www.truetex.com/raspberry_pi_4.jpg

Figura 1-21. Módulo V1 Rev 1.3 de cámara para Raspberry Pi.

La tabla 1-5 muestra los detalles técnicos de dos versiones de modelos de cámaras para Raspberry Pi. La primera versión de este módulo posee características convenientes en la captura de imagen, tales como, enfoque fijo y grabación de video 1080p a 30fps, 720p a 60fps y 480p hasta 90 fps. Este módulo se conecta a la Raspberry Pi utilizando la interfaz MIPI-CSI, a través de un cable plano de 15 pines. El bus CSI es capaz de alcanzar velocidades de transmisión de información extremadamente altas y transporta exclusivamente datos de los píxeles al procesador BCM2835.

Tabla 1-5. Comparación técnica de módulos de cámaras para Raspberry Pi

Característica	Módulo de cámara V1	Módulo de cámara V2
Peso	3 g	3g
Resolución fija / sensor	5 Mpx / 259 x 1944 pixeles	8 Mpx / 3280 x 2464 pixeles
Modos de video	1080p30, 720p60 y 480p60/90	1080p30, 720p60 y 480p90
Profundidad de campo	> 1 m	Ajustable
Distancia focal	3.6 mm +/- 0.01	3.04 mm
Campo de visión horizontal	53.50 +/- 0.13 grados	62.2 grados
Campo de visión vertical	41.41 +/- 0.11 grados	48.8 grados
Radio focal	2.9	2.0

Fuente: Tabla creada a partir de imagen en <https://www.raspberrypi.com/documentation/accessories/camera.html>

1.6 OBJETIVOS DEL PROYECTO

A continuación se describen los objetivos del proyecto de titulación, desglosando desde el general a los específicos.

1.6.1 Objetivo general

- I. Diseñar e implementar un sistema de bloqueo vehicular mediante un alcoholímetro electrónico y analizador facial.

1.6.2 Objetivos específicos

- I. Obtener datos estadísticos de seguridad vial e información legislativa del tránsito y exponer la problemática identificada.
- II. Proponer una solución que disminuya los índices de accidentes automovilísticos.
- III. Diseñar un sistema de medición de alcoholemia a partir de la configuración de un sensor de alcohol.
- IV. Diseñar un sistema de detección de personas con funciones para reconocer y agregar nuevos usuarios.
- V. Controlar el sistema de encendido del vehículo en base a los datos obtenidos del usuario detectado.

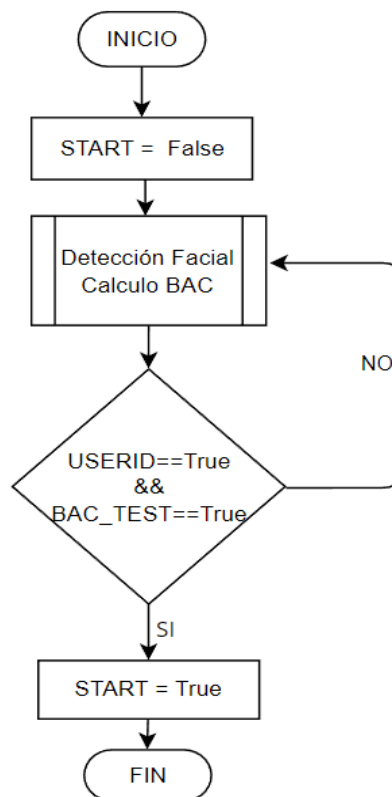
CÁPITULO 2: ESTRUCTURA DE LA SOLUCIÓN

2. ESTRUCTURA DE LA SOLUCIÓN

A continuación se exponen, la propuesta para el sistema de bloqueo, el diseño del alcoholímetro y el programa de identificación de usuario.

2.1 SISTEMA DE BLOQUEO

Como se muestra en la figura 2-1, para que el desbloqueo del sistema ocurra, el usuario debe comprobarse la identidad del usuario y el desarrollo correcto del test de alcohol, al comprobar la capacidad del usuario para una conducción segura. A través de estas funciones se evalúa el resultado de las mediciones en dos variables, “USERID” y “BAC_TEST” respectivamente. El proceso termina exitoso cuando ambas variables son verdaderas, y por ende, se asigna a la variable “START” un valor verdadero.



Fuente: Diagrama de flujo creado por el autor de la investigación

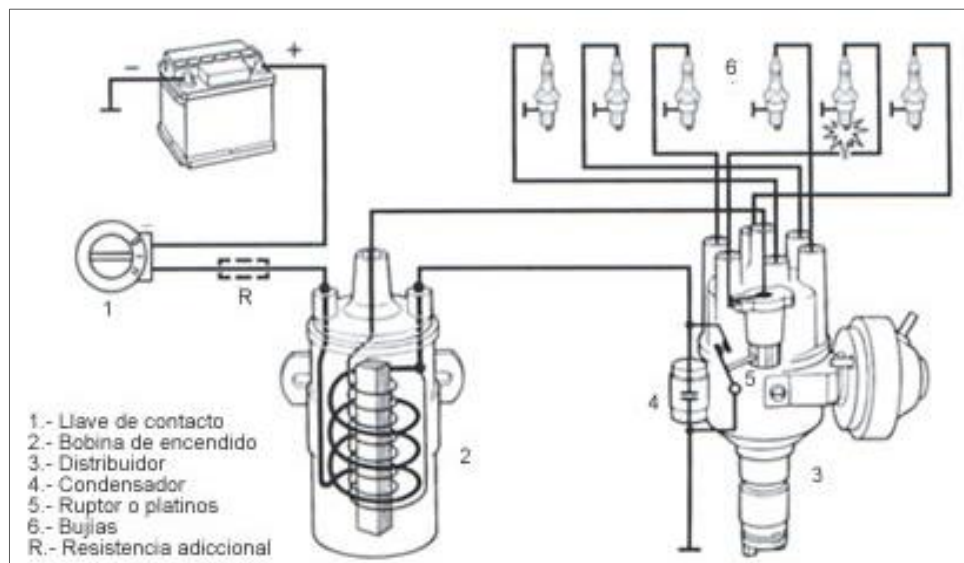
Figura 2-1. Diagrama de flujo del sistema de bloqueo de corriente a la unidad de encendido.

En el caso que una o ambas funciones retornen un valor falso, “START” tomará también un valor falso. El valor positivo de esta variable determinará si el relé entra en funcionamiento, o de lo contrario, permanece en su estado original.

2.1.1 Sistema de encendido de un motor

El sistema de encendido consta de varios componentes que trabajan juntos, controlados por la computadora interna del vehículo. Un sistema convencional de la marca reconocida BOSH, como el de la figura 2-2, posee componentes tales como:

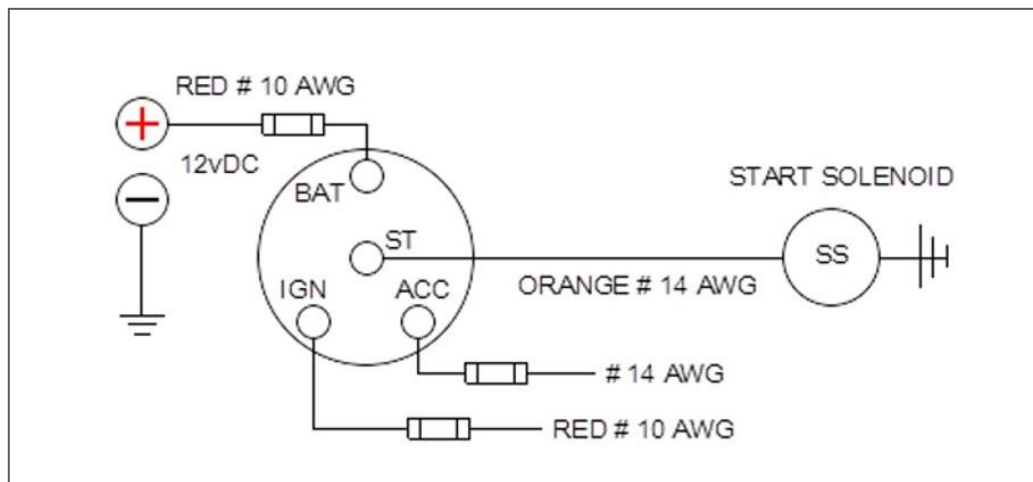
- a) Batería y interruptor de encendido (líneas de baja tensión)
- b) Bobina de encendido, distribuidor, condensador y bujías con distribución de cables (líneas de alta tensión)



Fuente: https://autoytecnica.com/wp-content/uploads/2017/09/sistema_encendido_conv_bobina.jpg

Figura 2-2. Esquema de un sistema de encendido convencional de la marca BOSH.

Para dar energía al sistema de bloqueo y posteriormente encendido del motor, se intervienen las terminales eléctricas del interruptor de encendido del vehículo. La figura 2-3 muestra las terminales que posee este conmutador, donde cada una de estas líneas activa distintas funciones del automóvil.



Fuente: <https://mgispeedware.com/wp-content/uploads/2018/08/4-position-IKS-diagram.jpg>

Figura 2-3 . Diagrama de un Interruptor de encendido de cuatro posiciones.

Por lo general, hay cuatro terminales en estos interruptores con las siguientes etiquetas:

- BAT: Es el terminal positivo de la batería y tiene una tensión constante de 12v.
- ACC: El terminal de accesorios envía energía a los componentes secundarios del automóvil (radio, luces y los limpiaparabrisas).
- IGN: El terminal de ignición entrega energía a los componentes principales del vehículo (bobina de encendido, bomba de gasolina, computadora)
- ST: El terminal de inicio es responsable de activar el solenoide de arranque del motor.

Como muestra la tabla 2-1, al seleccionar una de estas posiciones se activan o desactivan los voltajes de las demás posiciones del interruptor, en donde “ON” representa las terminales encendidas y “OFF” las apagadas.

Tabla 2-1. Condiciones de encendido/apagado de un interruptor de encendido.

Posición del interruptor	ACC	IGN/ON	ST
ACC	ON	OFF	OFF
IGN	ON	ON	OFF
ST	OFF	ON	ON

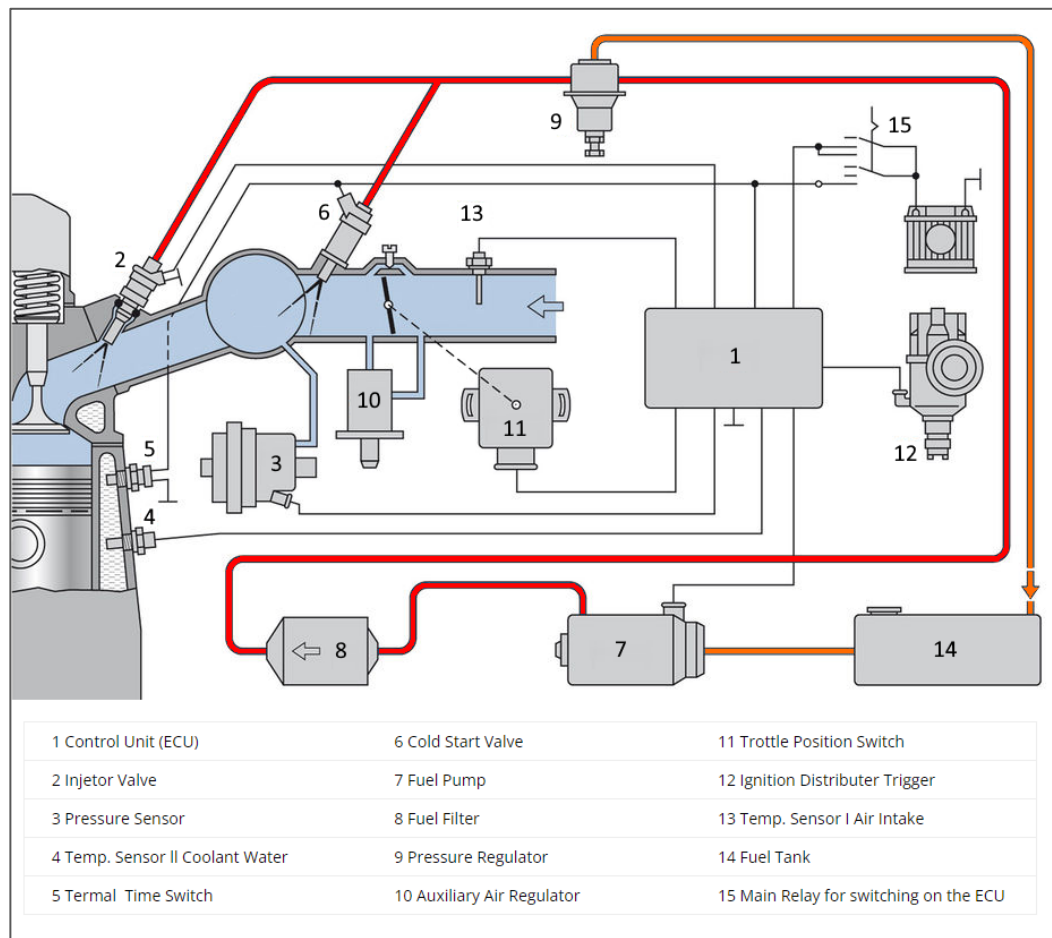
Fuente: Tabla creada por el autor en base a <https://www.infinitybox.com/wiring/wiring-an-ignition-switch/>

Cuando la posición de inicio esta encendida, el terminal de accesorios se encuentra desactivado, esto debido a que el automóvil utiliza toda la intensidad de corriente necesaria para activar la bobina de encendido, la cual está formada por dos devanados, primario y secundario. El devanado primario utiliza la energía de la batería para crear una chispa, el secundario aumenta la corriente de la chispa a un voltaje cercano de 30.000 volts y los envía al distribuidor.

El distribuidor es un girador preciso que distribuye las chispas a través de cables individuales a las bujías con una sincronización precisa mediante el uso de un rotor. Cuando las válvulas de admisión han distribuido la cantidad adecuada de combustible y vapor en el cilindro, la bujía produce una chispa que calienta la mezcla hasta lograr una combustión.

El sistema de inyección electrónico de la figura 2-4 cuenta con distintos sensores de presión y temperatura, reguladores, filtros, tanque de almacenamiento y, una unidad que procesa los valores de entrada de los distintos sensores y controla las válvulas de inyección.

Este modelo posee dos relés en el funcionamiento del motor. El relé principal y el de bomba; el principal solo puede ser reemplazado por uno original con diodo de bloqueo y marcado con un punto azul. La utilización de un relé de corriente normal, podría provocar daños en la unidad.

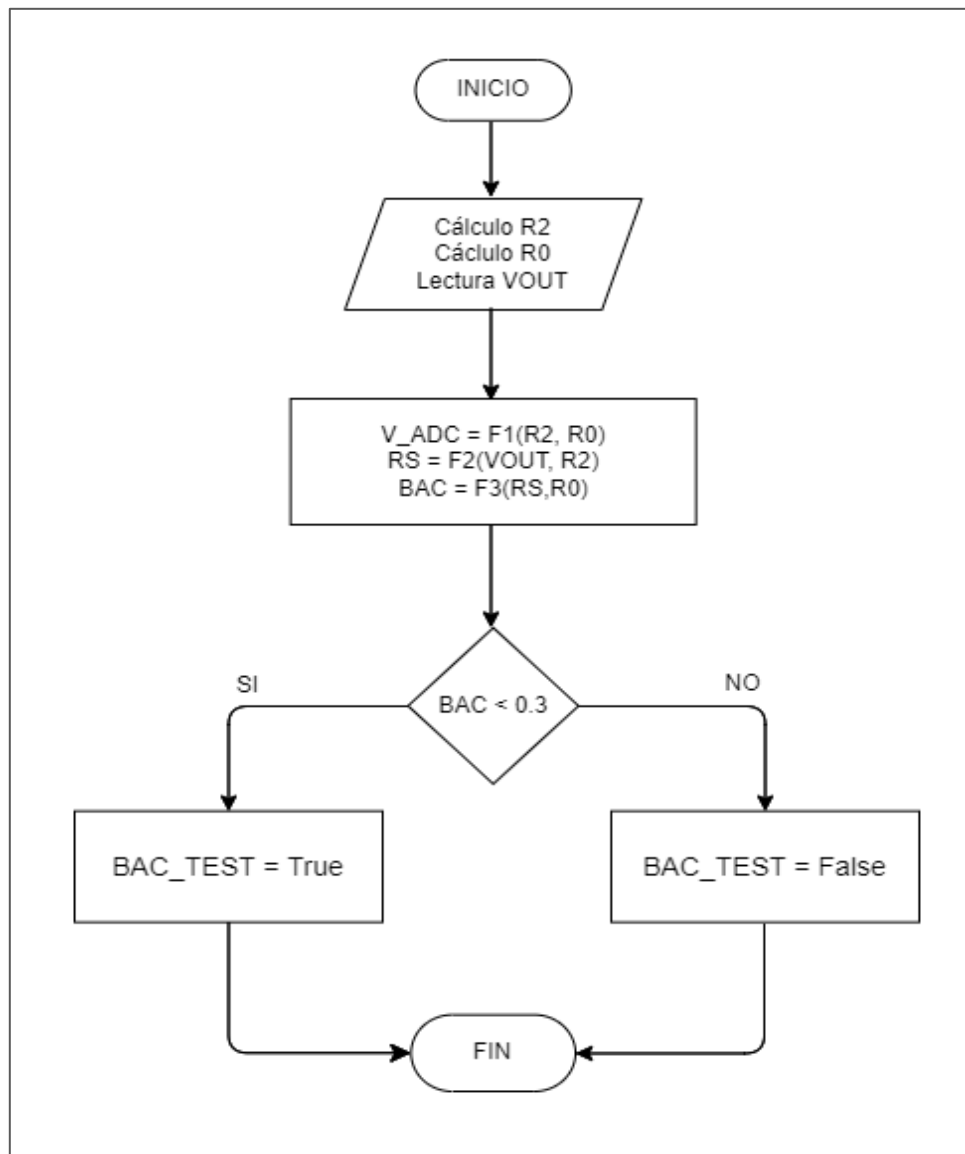


Fuente: <https://volvo-gruk.ch/en/d-jetronic/function-of-components/>

Figura 2-4. Componentes de un sistema de encendido de motor D-Jetronic de Bosh.

2.2 DISEÑO ALCOHOLÍMETRO

El cálculo de los niveles de alcohol en la sangre cuenta de dos partes, una analógica y otra digital. Como se presenta en la figura 2-5, en el primer nivel se calculan los valores de las resistencias del módulo y se lee el voltaje analógico del sensor, mientras que en el segundo, la Raspberry Pi resuelve las conversiones necesarias para calcular el valor del contenido de alcohol en el aire.



Fuente: Diagrama de flujo creado por autor.

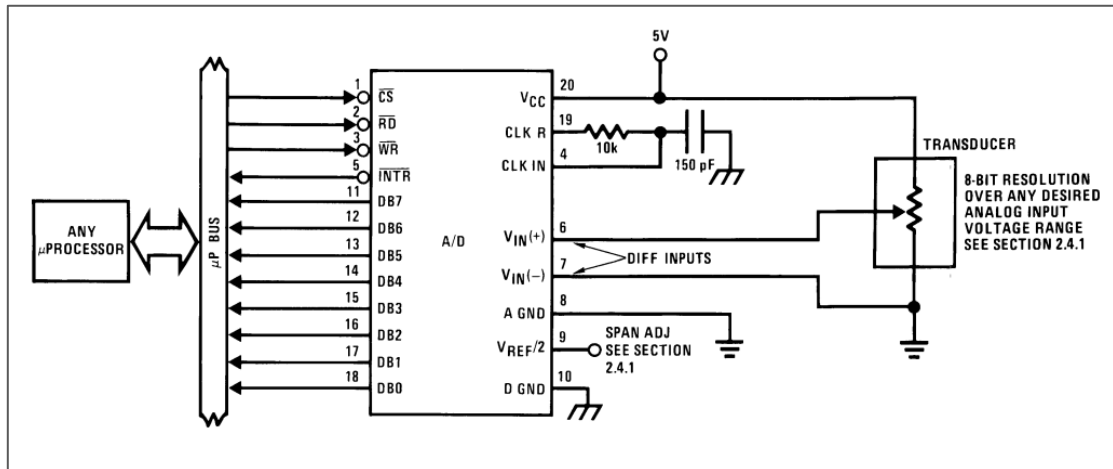
Figura 2-5. Diagrama de flujo del programa de medición de alcoholemia.

2.2.1 De la conversión analógica a digital

En primer lugar, como se tomó conocimiento en el primer capítulo, el nivel alcohol detectado por el sensor MQ3 es mostrado en el pin analógico “AOUT”, y al ser un voltaje variable es necesaria su transformación a un formato digital. De esta forma, la información puede ser recibida en la Raspberry Pi y luego ser interpretada por código para su conversión inversa.

2.2.1.1 Del conversor ADC0804

El ADC0804 es un convertidor de señal analógica a digital de 8 bits y se alimenta con 5V. La hoja de datos del fabricante ofrece distintas configuraciones; a continuación, es posible evidenciar en la figura 2-6 las conexiones de una aplicación típica de conversión analógica a digital con un tamaño de 8 bits, es decir, divide el voltaje de entrada desde 0 a 255.



Fuente: Captura fotográfica del datasheet del componente ADC0408.

Figura 2-6. Esquema de típica aplicación de conversión analógica a digital con el componente ADC0408.

El ADC0804 requiere una fuente de reloj que oscile entre 100 KHz y 1460 KHz para su funcionamiento. Se puede utilizar la señal de su reloj interno cuya frecuencia es de 460 KHz o se puede generarla frecuencia con un circuito RC entre el pin 6 y el pin 9, sabiendo que el tiempo mínimo de conversión es 110us. La frecuencia de esta red queda determinada por la siguiente ecuación:

$$F_{clock} = \frac{1}{1.1RC}$$

2.2.1.2 Del Arduino NANO

El Arduino Nano es una placa microcontroladora de uso fácil, con memoria flash que varia entre 16 y 32KB. Esta placa cuenta con la capacidad de recibir datos de tipo tanto analógico como digital.

Al ser una versión más compacta y fácil de adaptar que la Arduino Uno, debido a su tamaño mas pequeño, y es útil para desarrollar proyectos en los que se necesite utilizar un espacio reducido.

Como se muestra en la tabla 2-2, el Arduino NANO trae consigo el microcontrolador ATmega328 el cual posee comunicación serial del tipo USART y SPI, siendo ambas compatibles con la Raspberry Pi. Además, posee un canal de puertos análogos lo que permiten recibir un voltaje de este tipo y almacenarlos en la memoria interna, con el fin de transformar esa información a una escala deseada.

Tabla 2-2. Detalles técnicos del microprocesador Arduino NANO

Características	Arduino NANO
Microcontrolador ATmega328	Contador de tiempo real con oscilador independiente Puerto serial programable USART Interfaz serial SPI (Master/Slave)
Fuente de energía	Conexión de 5V via USB tipo Mini-B Fuente de alimentación externa regulada de 5V (pin 27) Fuente de alimentación externa no regulada de 6-20V (pin 30)
Entradas/Salidas	22 puertos digitales 8 puertos análogos 6 puertos PWM (salida)

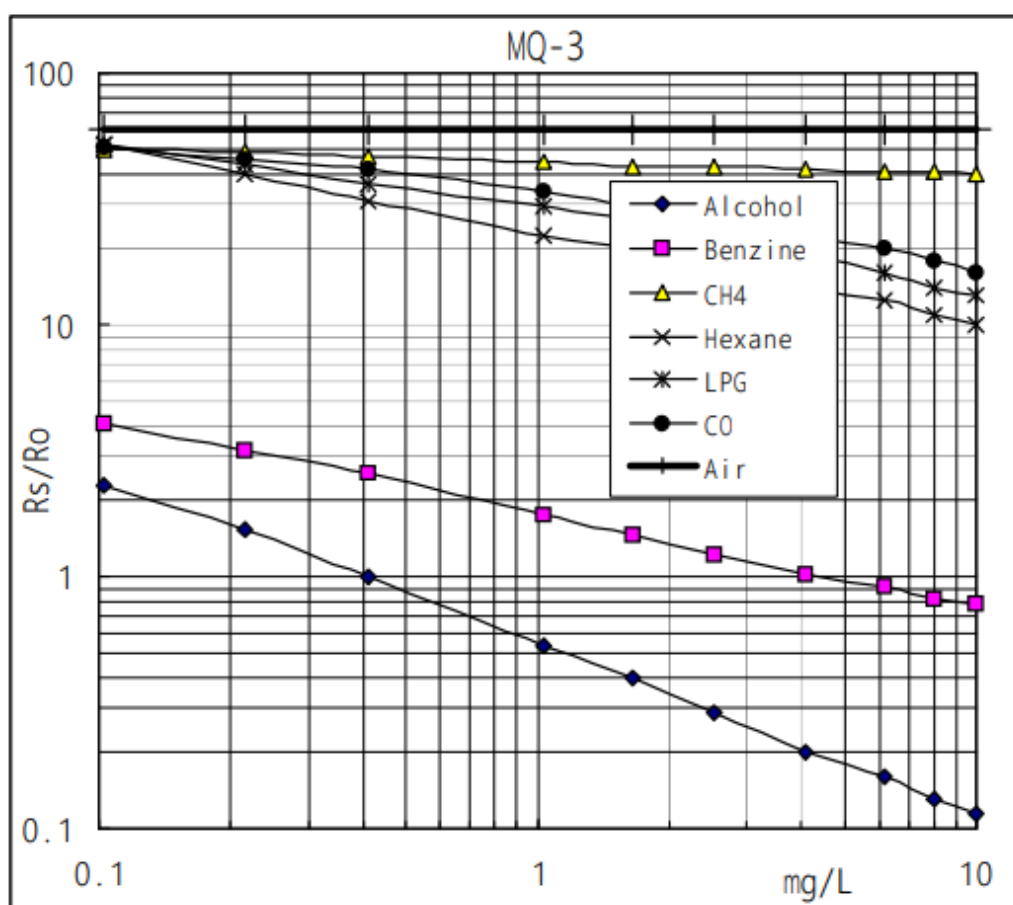
Fuente: Tabla creada a partir de datasheet del fabricante en <https://docs.arduino.cc/hardware/nano>

2.2.2 Del cálculo de la función del sensor

El sensor MQ3 posee una capa hecha de dióxido de estaño, que reacciona a distintos gases. Este material aumenta la conductividad eléctrica del sensor cuando entra en contacto con el gas a detectar, resultando en una variación de la resistencia interna del sensor R_s , inversamente proporcional a la concentración del gas.

Por otro lado, cada sensor posee una constante resistiva R_0 , cuyo valor nominal equivale a R_s cuando se expone a una concentración de 0.4mg/L de alcohol en aire limpio o sin la presencia de otros gases. Este valor se calcula para poder realizar ajustes sensitivos, ya que este valor varía dependiendo de cada sensor.

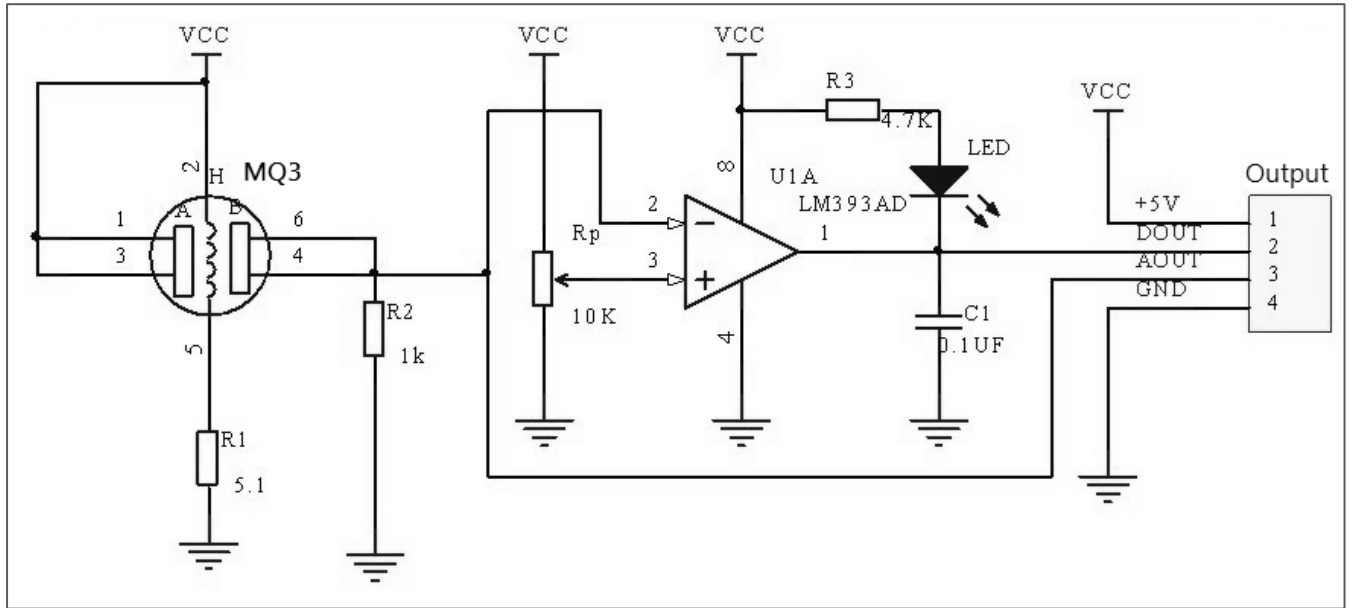
Utilizando la información que entrega la gráfica logarítmica de la figura 2-7, se pueden calcular los valores correspondientes de R_s y R_0 , ya que se relacionan con la cantidad de miligramos de gas detectado en un volumen de 1 litro. En base a la hoja de datos, cuando un gas se expone sobre el sensor, R_0 se mantiene fijo y solo varía la magnitud de R_s . Otro dato a evaluar es que cuando el sensor es expuesto a aire puro, la relación de R_s sobre R_0 toma un valor igual a 60.



Fuente: Captura fotográfica de datasheet del sensor MQ3.

Figura 2-7. Gráfica logarítmica de la relación R_s/R_0 vs. la concentración de gas detectada.

Según el circuito de la figura 2-8 la resistencia R_2 forma un divisor de voltaje en la salida del sensor MQ3. El valor de esta resistencia se calcula midiendo el valor óhmico en ambos extremos de la resistencia. Otra opción para obtener el valor resistivo es medir entre la salida “AOUT” y “GND”, con el módulo sin ser energizado.



Fuente: <https://forum.mysensors.org/uploads/files/1482692241224-mq2-sch.jpg>.

Figura 2-8. Diagrama esquemático del módulo MQ3.

Finalmente se procede a alimentar el sensor, el convertidor ADC y la Raspberry Pi. Manteniendo el sensor expuesto a aire limpio y utilizando el valor medido de R_2 , se lee la salida análoga del módulo y se utiliza la ecuación (1) y (2), para calcular el valor óhmico de R_s y de R_o , respectivamente.

$$R_s = \frac{V_{cc} * R_2}{AOUT} - R_2 \quad (1)$$

Ya que la relación de R_s sobre R_o en aire limpio es igual a 60, se despeja R_o para encontrar el valor de constante.

$$R_o = \frac{R_s}{60} \quad (2)$$

Es posible calcular la ecuación de una recta en una gráfica logarítmica, sabiendo que “ F ” es proporcional a “ x ” elevado a la pendiente de la recta “ m ”. Donde “ $F(x)$ ” representa los valores en el eje de las ordenadas, y “ x ” los valores en el eje de las abscisas.

$$F(x) = F_0 \left(\frac{x}{x_0} \right)^{\frac{\log(F_1/F_0)}{\log(x_1/x_0)}} \quad (3)$$

Al reemplazar los valores de dos puntos elegidos en la gráfica, se obtiene la ecuación que representa la recta de la concentración de alcohol detectado

$$\begin{aligned} F(x) &= (1) \left(\frac{x}{0.4} \right)^{\frac{\log(0.2/1)}{\log(4/0.4)}} \\ F(x) &= \frac{1}{0.4} x^{-0.699} \end{aligned} \quad (4)$$

Reemplazando las unidades de los ejes del gráfico de la figura 2-7, en las variables de la ecuación (4), se obtiene que:

$$\frac{R_s}{R_o} = \frac{1}{0.4} * mg/L_{aire}^{-0.699} \quad (5)$$

Se despeja miligramos sobre litro, se obtiene la ecuación para alcohol detectado en el aire:

$$mg/L_{aire} = 0.4 * \frac{R_s}{R_o}^{-1.43} \quad (6)$$

2.2.3 Cálculo de alcohol en la sangre

Finalmente se calcula la ecuación para alcohol en la sangre, utilizando la proporción investigada y ajustándola a la métrica estándar de la legislación vigente. Donde los gramos de alcohol por litro de sangre son equivalentes a 2000 veces los mg de alcohol en un volumen de un litro de aire.

$$g/L_{sangre} = 2000 * mg/L_{aire} \quad (7)$$

Sustituyendo la ecuación (6) en la ecuación (7) se obtiene:

$$g/L_{sangre} = \left(\frac{2000}{1}\right) * 10^{-3} g/L_{aire} = 2 * 0.4 \frac{R_s^{-1.43}}{R_o} \quad (8)$$

Al resolver, se obtiene la ecuación que calcula el nivel de alcohol en la sangre, utilizadno como incógnita la resistencia del sensor.

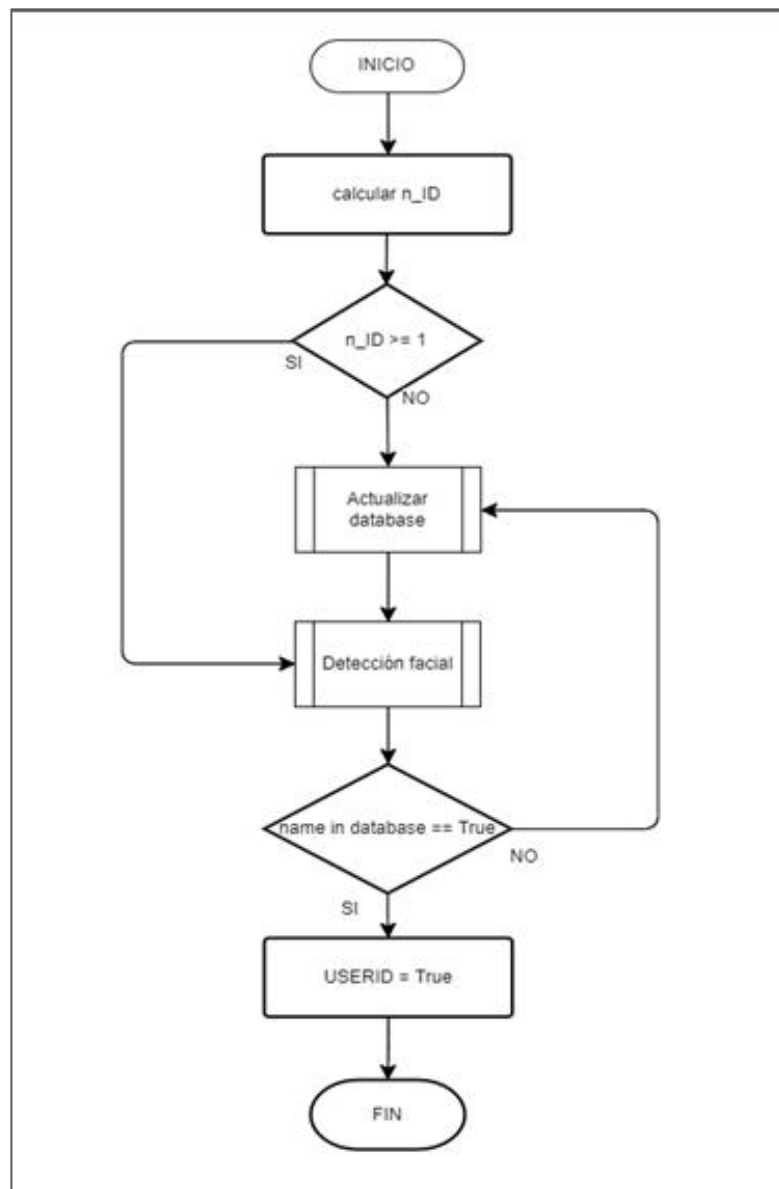
$$g/L_{sangre} = 0.8 \frac{R_s^{-1.43}}{R_o} \quad (9)$$

2.3 DETECTOR DE IDENTIDAD

El diagrama de la figura 2-9 muestra el proceso para la identificación de usuarios detectados por la cámara. En primero paso se calculará la cantidad de usuarios registrados en la base de datos del programa y almacenará en la variable “n_ID”.

Para realizar el proceso de detección facial tanto para la actualización de la base de datos con imágenes y el reconocimiento de identidad, se utilizará el método de detección de características Haar junto a la clasificación por histograma de gradientes orientadas, a través del módulo de reconocimiento facial *Face Recognition*, que se encuentra en la librería de libre acceso *GitHub*.

Luego, si la base de datos se encuentra sin usuarios ingresados, se procede a ejecutar el programa de actualización de data base, creando un nuevo perfil con las características del rostro del usuario detectado, en caso contrario, se realiza la detección facial directamente sin actualizar la base de datos. Dependiendo de las identidades que se detecten, se agregarán las personas nuevas y se actualizará la información, o por el contrario, si se reconoce al usuario registrado, la variable “USERID” toma un valor verdadero.

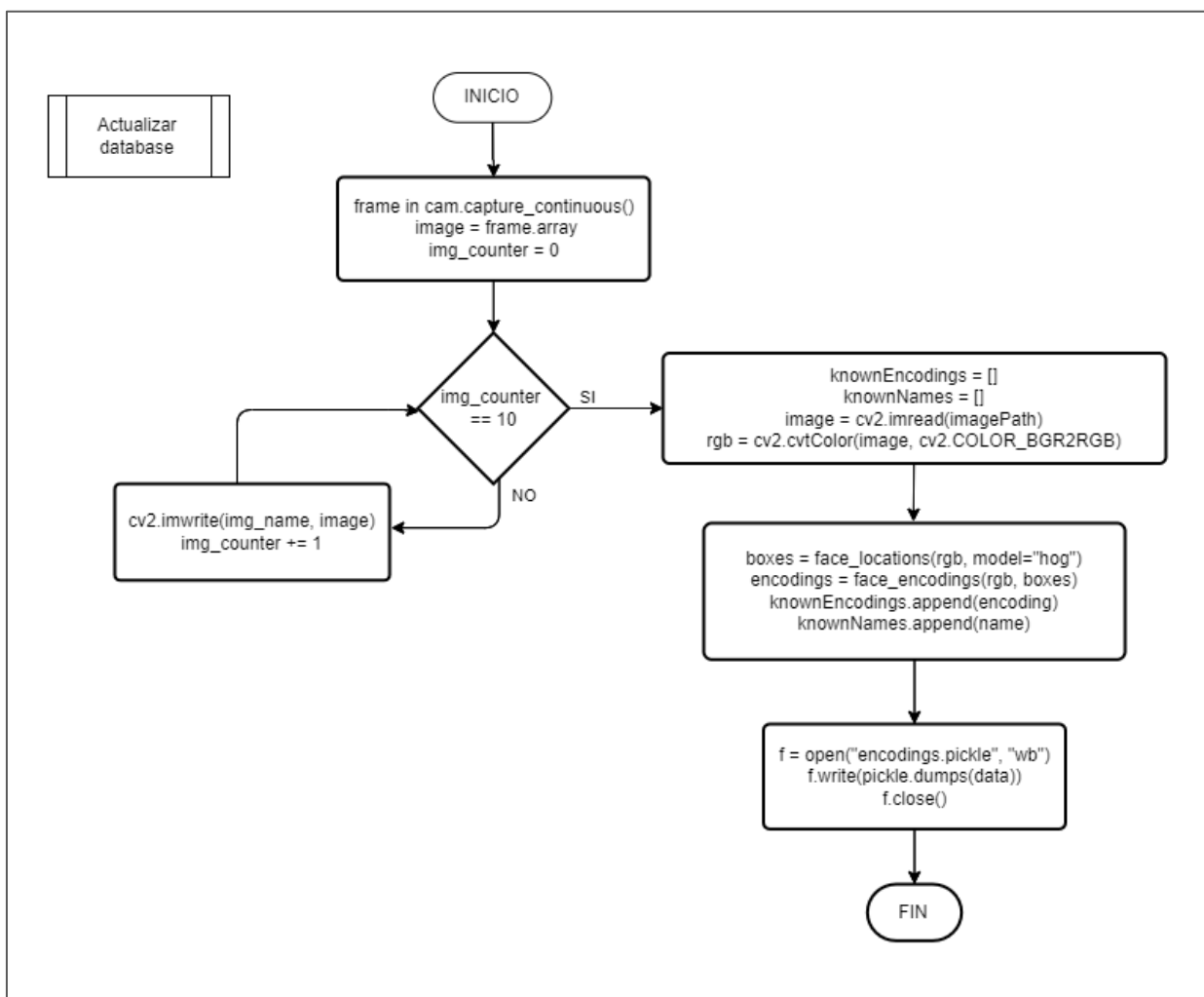


Fuente: Diseño creado por autor.

Figura 2-9. Diagrama de flujo del programa de detección de identidad.

2.3.1 De la adición de usuarios a la base de datos

La figura 2-10 plantea los pasos para crear perfiles de usuario en la base de datos. El programa comienza realizando las configuraciones para la comunicación con la cámara Raspberry Pi. El módulo PiCamera, es una biblioteca de funciones creadas específicamente para el uso de esta cámara. Entre las múltiples funciones para ajustar o modificar la imagen, se utiliza específicamente la función PiRGBArray.



Fuente: Diseño creado por autor.

Figura 2-10. Diagrama de flujo de la actualización de datos de identidad en la base de datos interna.

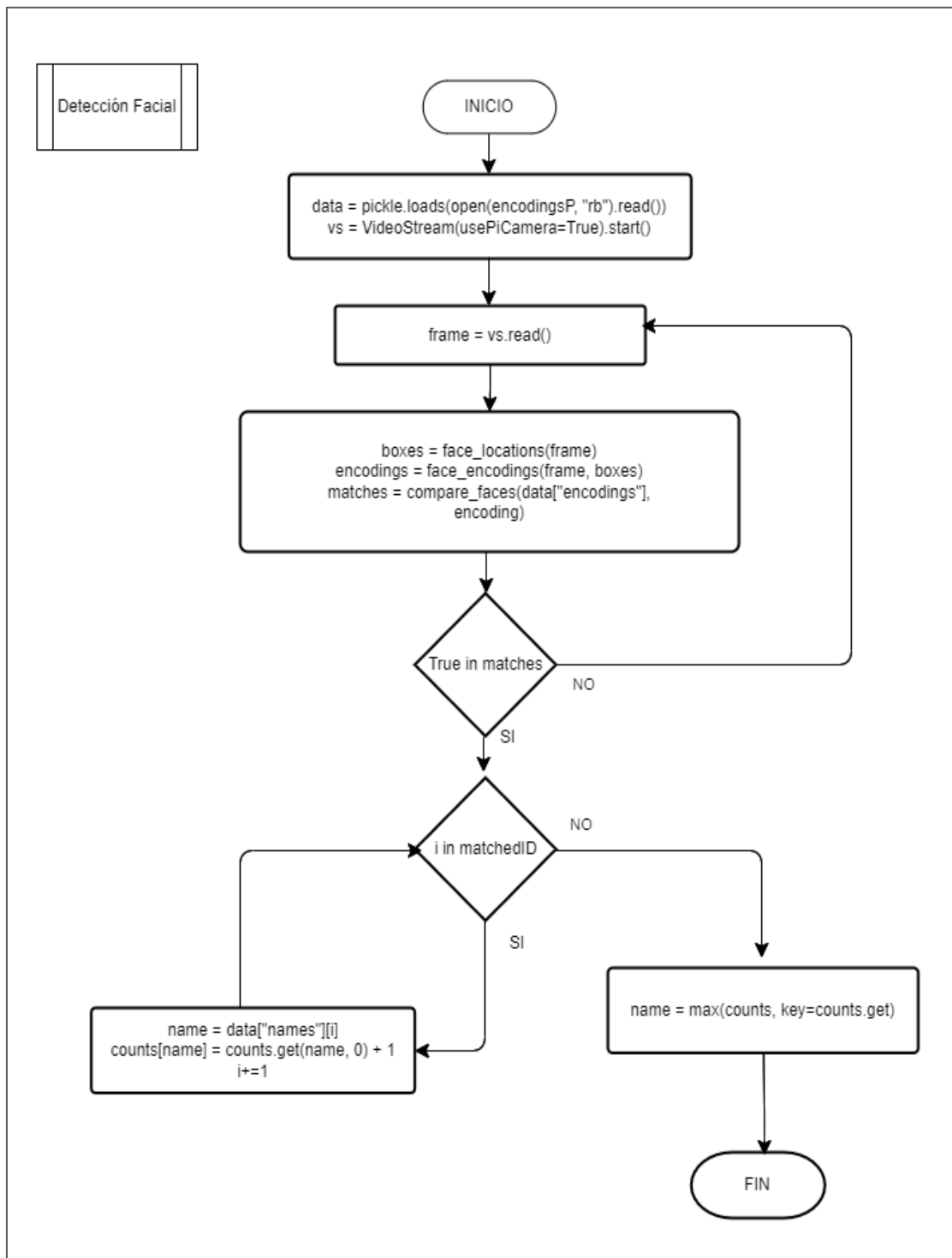
También se crea un directorio de imágenes del usuario a registrar con el nombre deseado. Con el uso de un contador, se realiza la captura de imágenes con la cara de la persona y se almacenan dentro de una carpeta con el nombre deseado en un directorio interno.

Terminado el bucle de creación, se recorrerán las imágenes de cada usuario y se crearán dos listas, una con las características Haar detectadas y otra con los nombres de cada usuario. Finalmente los datos se almacenan en un archivo de extensión “.pickle”, en donde se enlistan los usuarios registrados junto a las características faciales detectadas codificadas.

2.3.2 De la detección facial

La detección de caras ocurre como se aprecia en la figura 2-11. Primero se agrega la base de datos de usuarios del directorio interno y se almacena en una variable de nombre *data*. Luego, se inician las configuraciones de la cámara para realizar una grabación continua, para esto la variable *frame* contendrá la captura de imagen realizada.

Una vez abierta la imagen se utilizan dos variables, *encodings* y *matches*; en la primera las marcas faciales del usuario se comparan con las características de las almacenadas en la base de datos; en la segunda los aciertos indican que la identidad de la persona detectada tiene una alta probabilidad de estar dentro del directorio interno, ya que es seleccionada con el perfil de mayor cantidad de votos. Finalmente, se almacenan los nombres de identidad del usuario en la variable *name*.



Fuente: Diseño creado por autor.

Figura 2-11. Diagrama de flujo de la detección de caras y sus características personalizadas.

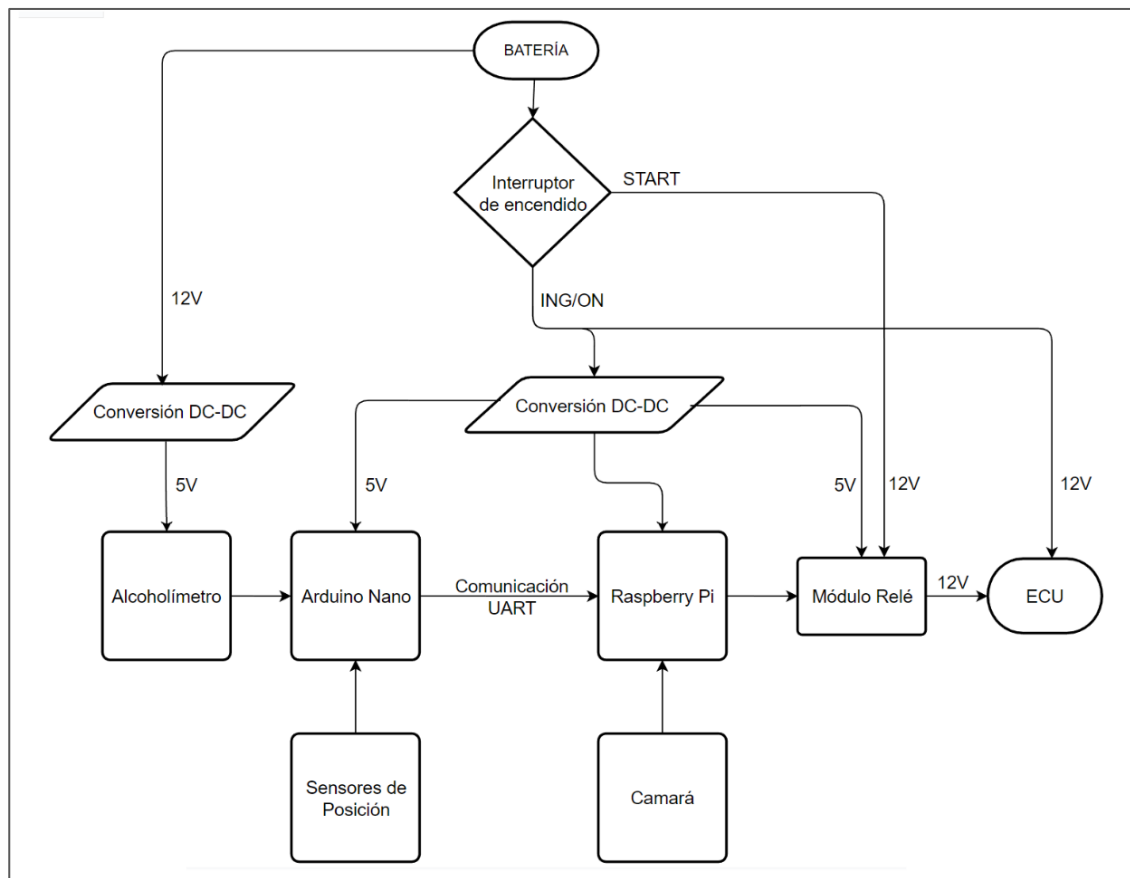
CAPÍTULO 3: IMPLEMENTACIÓN Y RESULTADOS DEL PROYECTO

3. IMPLEMETACIÓN Y RESULTADOS DEL PROYECTO

En este tercer y último capítulo se realiza la implementación de la solución con base en los puntos presentados en la segunda parte del trabajo. Los cálculos y las mediciones realizadas serán expuestas según el orden de importancia en el desarrollo del proyecto. Finalmente, se creará un detalle de los costos asociados a la realización del trabajo de titulación.

3.1 IMPLEMENTACIÓN DEL SISTEMA

A continuación, se muestran en detalle los ecosistemas de cada componente seleccionado, las conexiones eléctricas y secciones de la programación con los códigos necesarios para el proyecto. La figura 3-1 describe esquemáticamente la estructura eléctrica empleada de la solución elegida.



Fuente: Diseño creado por autor.

Figura 3-1. Diagrama de la conexión eléctrica entre los componentes del proyecto.

En una primera línea, el sensor de alcohol se energiza a través de un convertidor de voltaje DC-DC conectado al terminal de 12 V de la batería. En cambio, el Arduino Nano, la Raspberry Pi y el módulo relé se conectan a un segundo convertidor de voltaje DC-DC, que a su vez esta conectado al interruptor de encendido en la posición de *ING*. Esto debido a que el encendido de motor requiere un voltaje constante en los componentes eléctricos esenciales del vehículo, y además debido a que el sensor de gas necesita una temperatura de ajuste previo.

El sensor de alcohol MQ3 entrega desde su salida análoga un voltaje hacia el Arduino Nano, a través de un cable conectado al puerto análogo del microcontrolador. Este, también recibe un voltaje de carácter digital proveniente del sensor de posición, y que es conectado al pin digital número siete, según muestra la tabla 3-1.

Tabla 3-1. Selección de pines para las variables de entrada en Arduino Nano.

Comunicación variables	S_MED (input)	VOUT (input)
Arduino Nano	PIN D7	PIN A0

Fuente: Diseñada por el autor.

La conexión entre la placa Raspberry Pi y el microcontrolador Arduino Nano se realizó utilizando el puerto serial de ambas placas, como se ve en la tabla 3-2. Ya que los datos viajan desde el Arduino hacia la Raspberry, se considera como emisor al primero y receptor al segundo. De esta forma se conecta el *pin D1* con el *pin 10*, y el *pin D0* con el *pin 8* respectivamente.

Tabla 3-2. Selección de pines de comunicación serial de Raspberry Pi y Arduino Nano.

Comunicación UART	TX	RX
Raspberry Pi	PIN 8	PIN 10
Arduino Nano	PIN D1	PIN D0

Fuente: Diseñada por el autor.

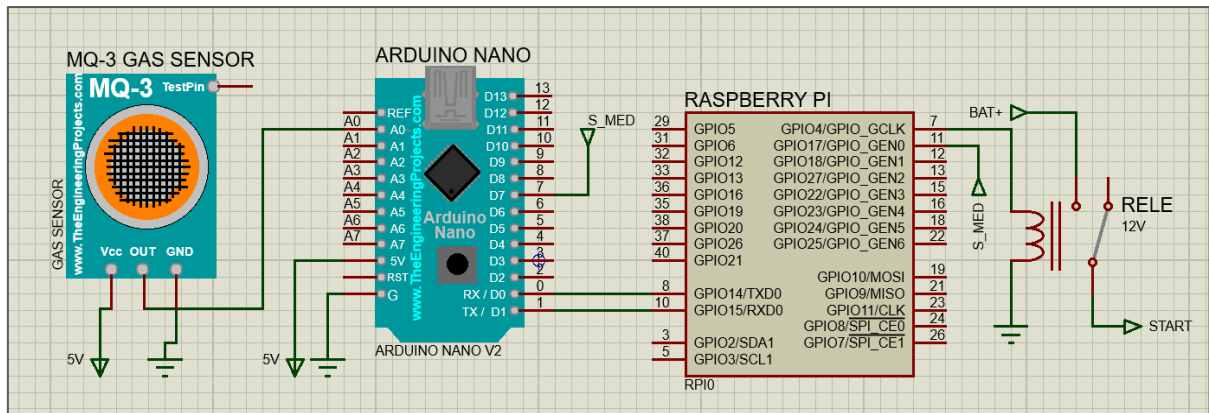
Tabla 3-3. Selección de pines para las variables de entrada y salida de la Raspberry Pi.

Comunicación variables	S_MED (input)	START (output)
Raspberry Pi	PIN 11	PIN 7

Fuente: Diseñada por el autor.

Adicionalmente, como se ve en la tabla 3-3, se definieron los puertos GPIO17 (pin 11) como entrada y GPIO4 (pin 7) como salida. El primero lee la señal que enviará el circuito de activación de medición y el segundo deshabilita el sistema de bloqueo.

Finalmente, la figura 3-2 muestra como se conectan todos los componentes del proyecto. Todos los dispositivos se encuentran energizados utilizando conversión DC-DC, que se mostrará a continuación.

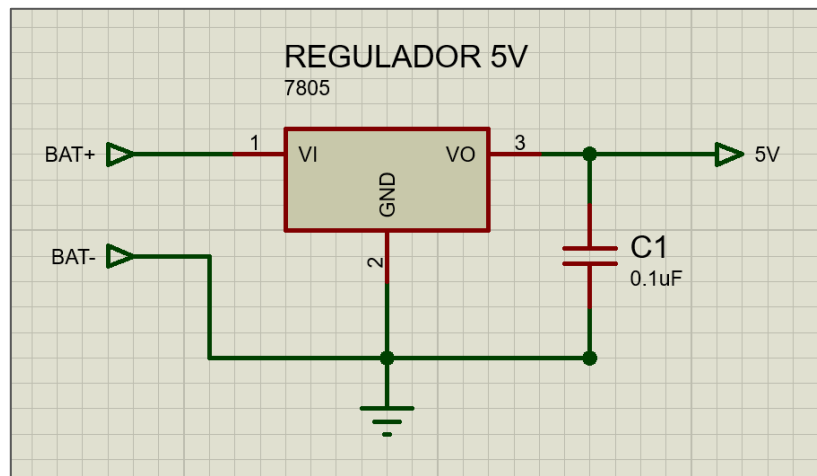


Fuente: Circuito diseñado por el autor en Proteus.

Figura 3-2. Circuito eléctrico simulado de la conexión entre placas y módulos.

3.1.1 De la conversión DC-DC

La regulación de voltaje es la primera parte del proyecto, ya que el encendido de los módulos corresponde a 5 V, es necesario reducir el voltaje proveniente de la batería de 12 V. Para esto se implementó un circuito con el integrado 7805 como se muestra en la figura 3-3.



Fuente: Circuito diseñado por el autor en Proteus.

Figura 3-3. Esquema del circuito regulador de voltaje.

3.1.2 Del ajuste de sensibilidad del sensor MQ3

En primer lugar, se realiza la medición resistiva de la salida analógica del módulo MQ3, donde se obtiene un valor de 1000 ohms. Este valor corresponde a la variable R_2 .

El siguiente paso consiste se exponer el sensor a una corriente de aire limpio hasta lograr la estabilización del voltaje de salida del módulo. Bajo esta condición, el valor de R_5 se obtiene utilizando la ecuación (1) calculada en el segundo capítulo.

Luego, utilizando la relación de R_5 con R_0 de la ecuación (2) se obtiene el valor resistivo R_0 . Se puede ver en la tabla 3-1, los valores calculados por el programa en la configuración inicial del sensor MQ3.

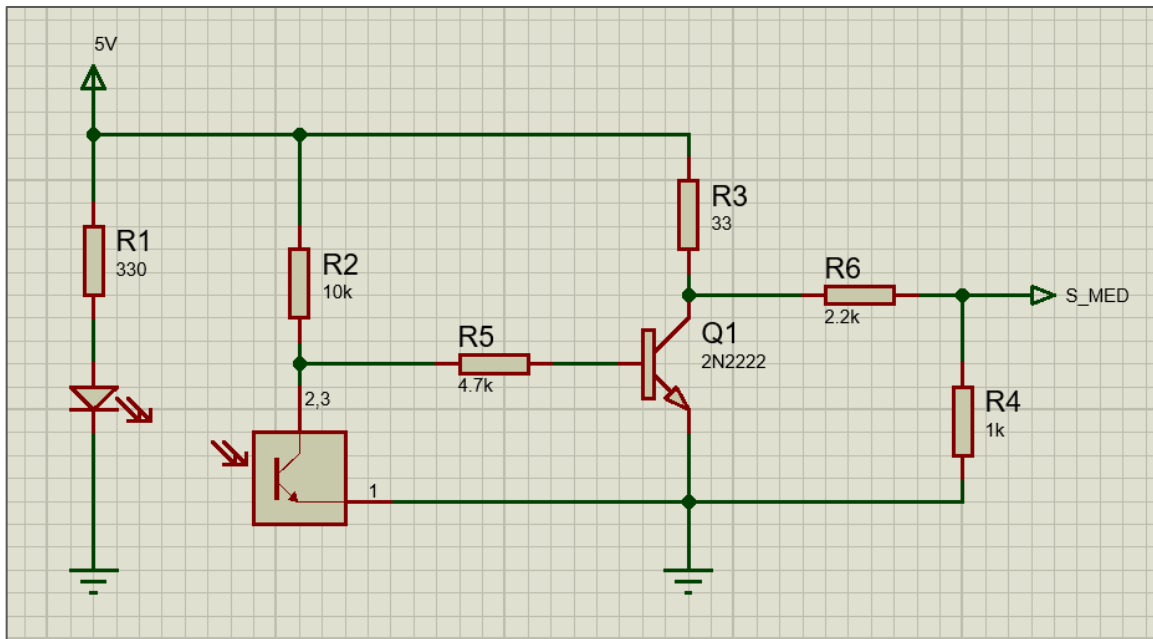
Tabla 3-4. Resultados en el cálculo de valores resistivos del módulo del sensor.

Resistencia	Valor medido
R_2	1000 Ω
R_5	5250 Ω
R_0	87,5 Ω

Fuente: Tabla diseñada por el autor en base a la configuración inicial del sensor.

3.1.3 De la configuración del sensor de posición

Para que el puerto digital del Arduino reciba la variable adjudicada al sensor de posición, se diseñó un circuito que se activa en el momento que el fototransistor reciba luz infrarroja desde el led emisor, como se ve en la figura 3-4.



Fuente: Circuito diseñado por el autor en Proteus.

Figura 3-4. Esquema del circuito sensor de posición para realizar la medición.

Cuando el usuario tome el equipo, el led emisor activará el led receptor. En consecuencia, la corriente comenzará a circular por el receptor, provocando que el voltaje de la base del transistor *Q1* disminuya a cero, reflejando un valor de 5 V en el colector del transistor. Se añaden a la salida dos resistencias en serie de 1000 ohm con el fin de disminuir el voltaje de activación a 3.3 V, protegiendo tanto el puerto digital del Arduino Nano como el de la Raspberry Pi.

Ambos fotodiodos son posicionados entre sí a una distancia de 5 cm, protegidos por dos cubiertas, que impiden que la luz externa llegue fácilmente al fototransistor y que pueda recibir la luz de fotodiodo adecuadamente. Ambos componentes se colocan en la base del alcoholímetro, de manera que al levantar el sensor de alcohol de su posición, la luz infrarroja emitida por el fotodiodo logre avanzar hasta el fototransistor.

3.1.4 De la configuración para Arduino Nano

Para realizar el cálculo del índice de contenido de alcohol en la sangre se implementó un código de programación en Arduino Nano utilizando las funciones de la figura 3-5.

```
void setup() {
  Serial.begin(9600);
  pinMode(3, INPUT);
}

void loop() {
  BAC = readBAC();
  sensor = digitalRead(3);
  Serial.println(BAC, sensor);
  delay(1000);
}

float readBAC() {
  sensor = analogRead(A0);
  voltaje = (VCC * sensor)/1024;
  Rs = (VCC * Ro)/voltaje - Ro;
  BAC = pow(abs((0.8*Ro/Rs)), 1.43);
  return BAC;
}
```

Fuente: Código creado por autor.

Figura 3-5. Funciones implementadas en el código de Arduino Nano.

En la configuración inicial del programa se define la frecuencia de la comunicación serial del microcontrolador en 9.600 baudios. También se declara el pin 3 como entrada digital.

Se define la función *readBAC()* que se encarga de realizar la conversión del voltaje enviado por el sensor MQ3. Utilizando la ecuación (8) se calcula la cantidad de alcohol presente en la sangre del usuario, y se retorna como un valor decimal de dos dígitos.

El programa principal se ejecuta en la función *loop()*, en donde 2 variables, *BAC* y *sensor* son enviadas a través del puerto de comunicación serial.

3.1.5 De la configuración para Raspberry Pi

Para este proyecto se implementó un código en la Raspberry Pi que se encargará de realizar la detección facial, el reconocimiento de identidad, realizar el test de alcoholemia y activar el sistema de encendido del vehículo.

Como muestra la figura 3-6, en primer lugar se importan las librerías encargadas de la gestión de puertos de la placa, acceso a los directorios y el reloj interno.

```
import time
import os
import serial
import RPi.GPIO as gpio

import cv2
import imutils
from imutils.video import VideoStream
from imutils import paths

import face_recognition
import pickle

gpio.setwarnings(False)
gpio.setmode(gpio.BCM)
gpio.setup(17, gpio.IN)
gpio.setup(4, gpio.OUT)

START = False
```

Fuente: Fragmento del código principal del proyecto.

Figura 3-6. Configuración del ecosistema del proyecto.

También se importan los módulos para manipular imágenes y archivos de video, seguido del módulo encargado en la detección y reconocimiento facial. Finalmente se configuran los puertos *GPIO17* como entrada y *GPIO4* como salida.

```
data = pickle.loads(open("encodings.pickle", "rb").read())
vs = VideoStream(usePiCamera=True).start()
users = dict({})
imagePaths = list(paths.list_images("dataset"))
for (i, imagePath) in enumerate(imagePaths):
    name = imagePath.split(os.path.sep)[-2]
    users[name] = 0
if len(users) == 0:
    print('Base de datos vacia\nCreando perfil para nuevo usuario')
    add_id(None)
    update_db()
```

Fuente: Fragmento del código principal del proyecto.

Figura 3-7. Creación del directorio para usuario nuevo.

Luego, como se muestra en la figura 3-7 se crean dos variables; *data*, que almacena el listado de usuarios con acceso al sistema; y *vs*, que inicia la configuración del módulo de cámara para Raspberry Pi utilizando el modo de transmisión de imagen continua. Luego, en caso de no existir una base de datos con la información codificada de los usuarios, se realiza la creación de esta.

Para realizar esto se utiliza la función *add_id(encoding_0)*, que se ve en la figura 3-8. Primero se crea el directorio con el nombre deseado por el usuario y luego se realiza la detección de identidades utilizando las características recibidas como punto de partida. Si no se reciben rasgos codificados, el programa realiza una captura del usuario no registrado en la base de datos.

```
def add_id(encoding_0):
    data = pickle.loads(open("encodings.pickle", "rb").read())
    name = input('Nombre usuario: ')
    os.mkdir('dataset/'+ name)
    . . . . .
    while value < 100:
        frame = vs.read()
        f = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        boxes = face_recognition.face_locations(f)
        encodings = face_recognition.face_encodings(f, boxes)
        if len(encodings) == 1:
            matches = face_recognition.compare_faces(data["encodings"], encodings[0])
            if True not in matches:
                if encoding_0 is None:
                    encoding_0 = encodings[0]
            matches = face_recognition.compare_faces(encodings, encoding_0)
            if True in matches:
                cv2.imwrite(img_name, frame)
                value += 10
                for (top, right, bottom, left) in boxes:
                    . . . . .
                if value == 100:
                    cv2.imshow("Almacenando en DB", frame)
                    cv2.waitKey(1)
                    cv2.destroyAllWindows()
            else:
                . . . . .
                cv2.imshow("Almacenando en DB", frame)
                cv2.waitKey(1)
```

Fuente: Fragmento del código principal del proyecto.

Figura 3-8. Función para agregar nuevos usuarios a la base de datos.

Luego, dentro de un bucle se realizará la comparación de las características recibidas con las características del usuario detectado por la cámara. En el caso de que esta comparación tenga un resultados positivos, se almacenarán en el directorio creado la captura de diez imágenes del usuario desconocido. Finalmente se realiza la actualización de la base de datos con la información del usuario nuevo utilizando la función *update_db()*.

Su funcionamiento se puede ver en la figura 3-9. Inicialmente se crean dos listas vacías que guardarán el nombre y las características faciales codificadas respectivamente. Luego, se recorre la directorio en donde se calculan las características y localidades faciales por cada imagen de los usuarios.

```
def update_db():
    imagePaths = list(paths.list_images("dataset"))
    knownEncodings = []
    knownNames = []
    for (i, imagePath) in enumerate(imagePaths):
        name = imagePath.split(os.path.sep)[-2]
        image = cv2.imread(imagePath)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        boxes = face_recognition.face_locations(f, model="hog")
        encodings = face_recognition.face_encodings(rgb, boxes)
        for encoding in encodings:
            knownEncodings.append(encoding)
            knownNames.append(name)
    data = {"encodings": knownEncodings, "names": knownNames}
    f = open("encodings.pickle", "wb")
    f.write(pickle.dumps(data))
    f.close()
```

Fuente: Fragmento del código principal del proyecto.

Figura 3-9. Función encargada de actualizar la base de datos de los usuarios.

Estas características son almacenadas junto al nombre de cada usuario, dentro de los listados de nombres y de características previamente creados. Finalmente en un diccionario se agrupan ambas listas, y se utiliza el módulo de manejo de archivos para realizar la escritura de la información de la identidad de los usuarios codificada. El uso de esta función se repite cada vez que un usuario nuevo se agrega al sistema, por lo que es necesario cerrar el archivo luego de realizar cambios.

La figura 3-10 muestra la función utilizada para detectar la presencia de usuarios, y la comprobación de su existencia en la base de datos del proyecto. Primero se ajusta la dimensión de la imagen recibida para que sea compatible con el algoritmo de detección facial y luego se detectan las ubicaciones dentro de la imagen que contienen posibles candidatos con características faciales.

La función *ID_detector()* compara las características faciales de la imagen con las del usuario inicial detectado, y dependiendo de la exactitud en sus similitudes se le asigna un porcentaje de validación, que aumenta a medida que se reconoce el usuario. Para detectar si estas características se encuentran dentro del sistema, se comparan las características faciales detectadas en la imagen con las almacenadas en la base de datos. Dentro de un diccionario se realiza una sumatoria de todas las veces en que las características del usuario sean asociadas a un nombre. El nombre de usuario que contenga la mayor cantidad de votos es seleccionado como etiqueta para el rostro detectado.

Cada vez que ocurra la asignación del nombre al usuario detectado, se aumentará el porcentaje de carga del usuario, y la cifra se almacenará en un diccionario asociado a los usuarios registrados. En caso de reconocer un individuo desconocido, el porcentaje de carga de este aumentará solo cuando se detecte también un usuario registrado.

Finalmente, se define un bucle donde se recorren las ubicaciones espaciales de los rasgos faciales en la imagen vinculados a los usuarios respectivos. Para cada detección se agrega un recuadro sobre el rostro junto al valor de carga del usuario. Una vez realizado los cálculos anteriores, se comprueba si es que la variable de carga *value* es igual a cien. Si se cumple y el nombre del usuario esta registrado, se termina el bucle y la función *ID_detector()* retorna un valor verdadero. En el caso de que si la persona detectada no se encuentra registrada, se llama la función *add_id(encoding_0)*, entregando como argumento las características faciales de la persona desconocida.

El bucle inicial del programa comienza como muestra la figura 3-11, donde se comprueba el valor booleano de la variable *ID_correct* en cada iteración. El valor de esta se invierte únicamente a través del retorno de la función *ID_detector()*.

```

def ID_detector():
    frame = vs.read()
    f = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
    f = cv2.cvtColor(f, cv2.COLOR_BGR2RGB)
    boxes = face_recognition.face_locations(f)
    encodings = face_recognition.face_encodings(f, boxes)
    names = []
    data = pickle.loads(open("encodings.pickle", "rb").read())
    for encoding in encodings:
        currentname = 'Desconocido'
        matches = face_recognition.compare_faces(data["encodings"], encoding)
        if True in matches:
            matchedIds = [i for (i, b) in enumerate(matches) if b]
            counts = {}
            for i in matchedIds:
                name = data["names"][i]
                counts[name] = counts.get(name, 0) + 1
            name = max(counts, key=counts.get)
            users[name] += 5
            . . . . .
        else:
            currentname = 'Desconocido'
            names.append(currentname)
            if currentname not in users:
                users[currentname] = 0
                . . . . .
            else:
                users[currentname] += 20
            . . . . .
    for ((top, right, bottom, left), name) in zip(boxes, names):
        . . . . .
        value = users[name]
        if name != 'Desconocido' and value == 100:
            . . . . .
            print('Bienvenid@ ', name)
            return True
        else:
            . . . . .
            cv2.imshow("Running Facial Recognition", frame)
            cv2.waitKey(1)
            if name == 'Desconocido' and value == 100:
                cv2.imshow("Running Facial Recognition", frame)
                print('Se ha detectado un usuario nuevo')
                add_id(encoding_0)
                update_db()
            . . . . .

```

Fuente: Fragmento del código principal del proyecto.

Figura 3-10. Función encargada de la identificación de personas detectadas.

```

ID_correct = False
while not ID_correct:
    if ID_detector():
        ID_correct = True
        print('Preparando test de alcoholemia, exhale sobre el sensor')
        . . . . .
        . . . . .

```

Fuente: Fragmento del código principal del proyecto.

Figura 3-11. Clasificación de casos para la validación de identidad del usuario.

Terminada la validación del usuario detectado, se entra a un bucle encargado de realizar el test de alcohol correctamente. Como se ve en la figura 3-12, la ejecución del test se realiza solo cuando el sensor de posición es activado. Una vez llamada la función responsable de la medición del BAC, se compara la cifra con el rango permitido actualmente por la ley.

```

while True:
    frame = vs.read()
    cv2.imshow("Esperando Usuario", frame)
    cv2.waitKey(1)
    s_med = gpio.input(17)
    if s_med == 1:
        cv2.destroyAllWindows()
        time.sleep(1)
        bac = medicion_UART(s_med)
        if bac != None:
            if 0.01 <= bac < 0.3:
                print('Usuario completa la prueba de alcohol exitosamente
                        con {} g/L en la sangre'.format(bac))
                print('ID y alcoholemia valida')
                cv2.destroyAllWindows()
                vs.stop()
                START = True
                break
            else: #USUARIO FALLA PRUEBA DE ALCOHOL
                print('Usuario deshabilitado para conduccion\n
                        Posee {} g/L de alcohol en la sangre'.format(bac))
                cv2.destroyAllWindows()
                ID_correct = False
                . . . . .
                print('Reiniciando sistema')
                break

```

Fuente: Fragmento del código principal del proyecto.

Figura 3-12. Bucle para la ejecución del test de alcohol.

En caso de que se encuentre dentro del límite permitido, se modifica el valor de la variable booleana *START* por un valor verdadero, de lo contrario la variable permanece en falso. En el primer caso, se quiebran todos los bucles del programa y se activa el relé de encendido del motor.

En el caso de que el valor de *START* permanezca en falso, se abandona el bucle encargado del test de alcohol y se reinicia el sistema, comenzando nuevamente a realizar la detección de usuarios.

La función *medicion_UART(s_med)* que se ve en la figura 3-13, es la responsable de la medicion de BAC del usuario. En esta se realiza la comunicación serial con Arduino Nano, en donde la información recibida es transformada a un valor decimal. En caso de que el sensor sea devuelto a su posición de anclaje, el test se interrumpe. Si el test se desarrolla satisfactoriamente, se retorna el valor de la variable *BAC*.

```
def medicion_UART(s_med):
    ser = serial.Serial('/dev/ttyUSB0', 9600)
    i = 0
    while s_med == 1:
        time.sleep(1)
        data = ser.read_until(b'\n')
        if data[0] != 79 :
            if data[1] == 46:
                line = data[-3:-7:-1][::-1].decode()
                i += 1
                BAC_m = float(line)
                BAC = max(BAC, BAC_m)
                print('test: ', i, ' - BAC : ', BAC_m)
                if BAC >= 0.3:
                    return BAC
    if i == 10:
        if BAC <= 0.01:
            return None
        return BAC
```

Fuente: Fragmento del código principal del proyecto.

Figura 3-13. Función encargada de realizar el test de alcoholemia.

3.2 RESULTADOS DE LA SOLUCIÓN

3.2.1 Resultados en la creación de usuarios

A continuación, se muestran los casos en que se agregan nuevos usuarios a la base de datos.

3.2.1.1 Base de datos vacía

Iniciado el sistema, se comprueba si la cantidad de usuarios dentro de la base de datos esta vacío. Como muestra la figura 3-14, al no encontrarse usuarios registrados, se procede a crear el directorio en donde se guardará el perfil de la persona.

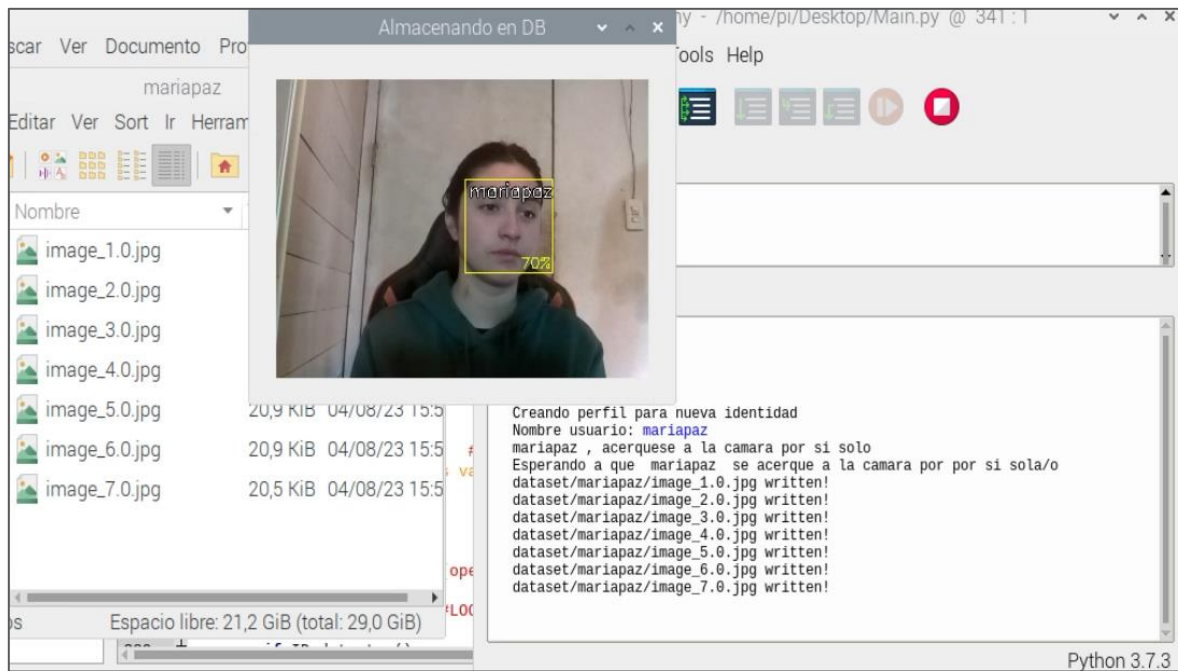


```
Shell x
>>> %Run Main.py
Iniciando sistema
Base de datos vacia
Creando perfil para nueva identidad
Nombre usuario: |
Python 3.7.3
```

Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-14. Comprobación de la base de datos del sistema.

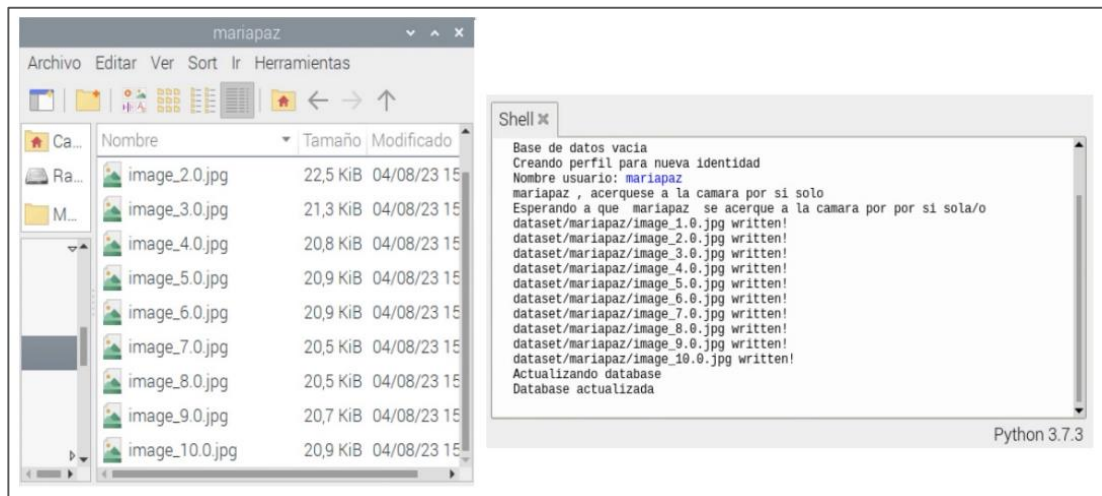
Para esto se requiere el ingreso de un nombre de usuario, que será utilizado para nombrar la carpeta dentro del directorio principal y etiquetar los rasgos faciales del usuario. Una vez recibido un nombre, se le pide al usuario que se acerque a la cámara sin la compañía de otra persona. Si se cumple esta condición, se realiza la captura de diez fotografías de la persona detectada, como se ve en la figura 3-15.



Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-15. Captura de imágenes del usuario detectado.

A medida de que el reconocimiento del usuario es realizado, las imágenes van siendo almacenadas dentro de la carpeta previamente creada. El porcentaje de carga es mostrado dentro del cuadro de detección del usuario nuevo, y es completado al llegar al 100%.



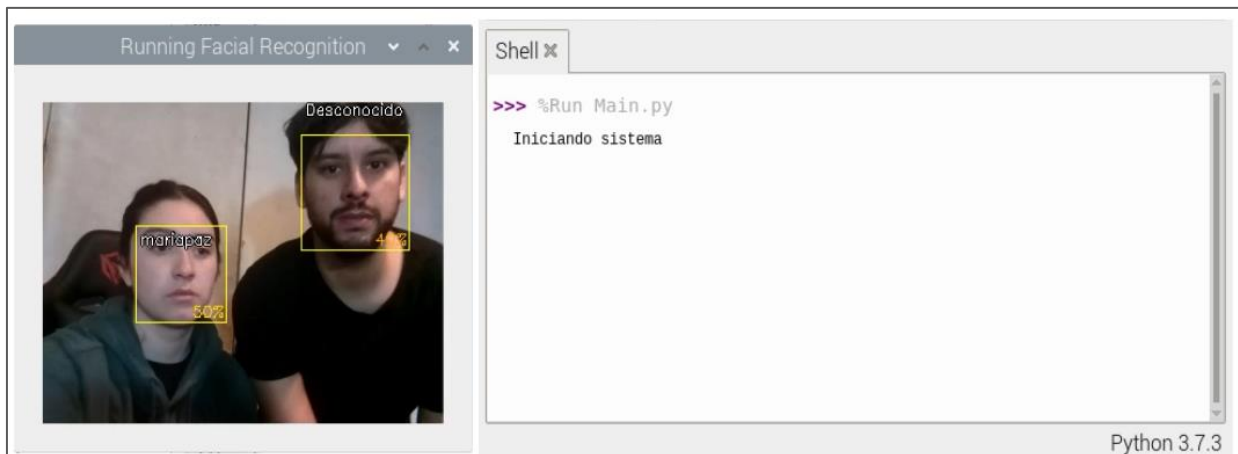
Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-16. Creación del perfil para el primer usuario.

Finalmente, cuando todas las capturas del usuario son almacenadas, la base datos se actualiza. Como se ve en la figura 3-16, las imágenes son enumeradas y guardadas dentro de la carpeta creada con el nombre del usuario registrado.

3.2.1.2 Al agregar un usuario adicional

Para agregar usuarios adicionales al sistema, es necesario la presencia de un usuario registrado. Como se ve en la figura 3-17, si se detecta un usuario desconocido junto a uno registrado, un porcentaje de carga para la creación del perfil del usuario desconocido irá aumentando.



Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-17. Detección de usuario registrado y usuario desconocido.

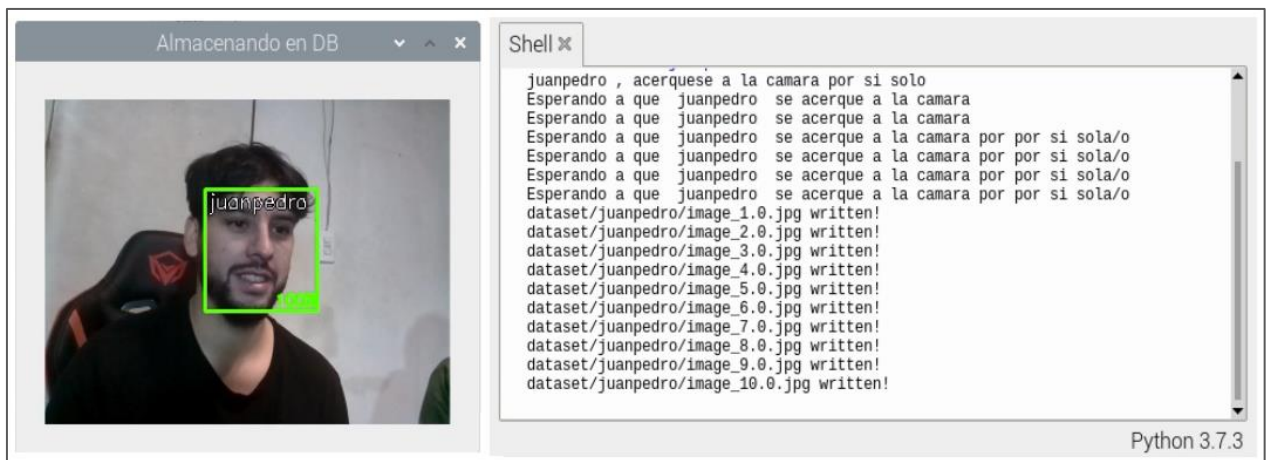
Una vez alcanzado el porcentaje requerido, se ve en la figura 3-18, como se realiza el proceso de agregación de usuario nuevo con normalidad.



Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-18. Captura de imágenes del usuario nuevo detectado.

Como se mostró previamente, el usuario debe encontrarse por sí solo frente a la cámara, esto con el fin de almacenar imágenes solo del usuario, sin la posible detección de rasgos faciales externos. Finalmente se muestra en la imagen 3-19, que se alcanza el 100% en el proceso de almacenamiento de imágenes del usuario, y por lo tanto continuará realizando la actualización de la base de datos.



Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-19. Almacenamiento de imágenes del usuario adicional.

Una vez reiniciado el sistema, el programa comenzará nuevamente a detectar los usuarios frente a la cámara, esta vez con los nombres asignados a las respectivas características faciales, como se ve en la figura 3-20.



Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-20. Detección de usuarios dentro del sistema.

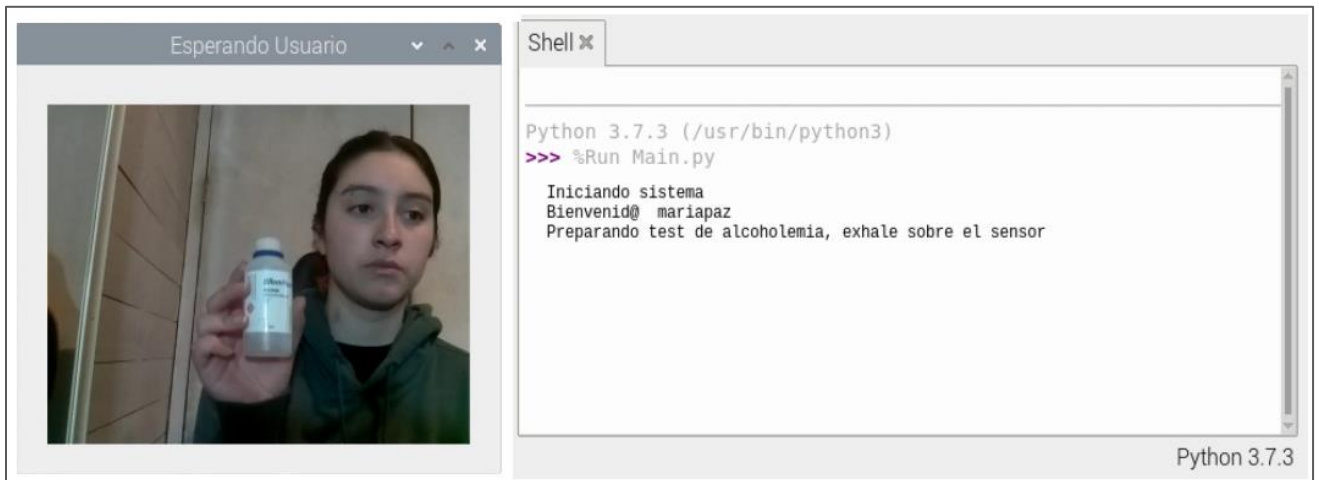
3.2.2 Resultados de la medición de alcohol

A continuación se muestran los dos resultados posibles en el funcionamiento del test de alcoholemia.

3.2.2.1 Test de alcoholemia con presencia de alcohol

Una vez realizada la identificación del usuario se procede a la toma de la prueba de alcoholemia. En este caso, como se muestra en la figura 3-21, se expone al sensor a una botella de alcohol desnaturalizado al 70%.

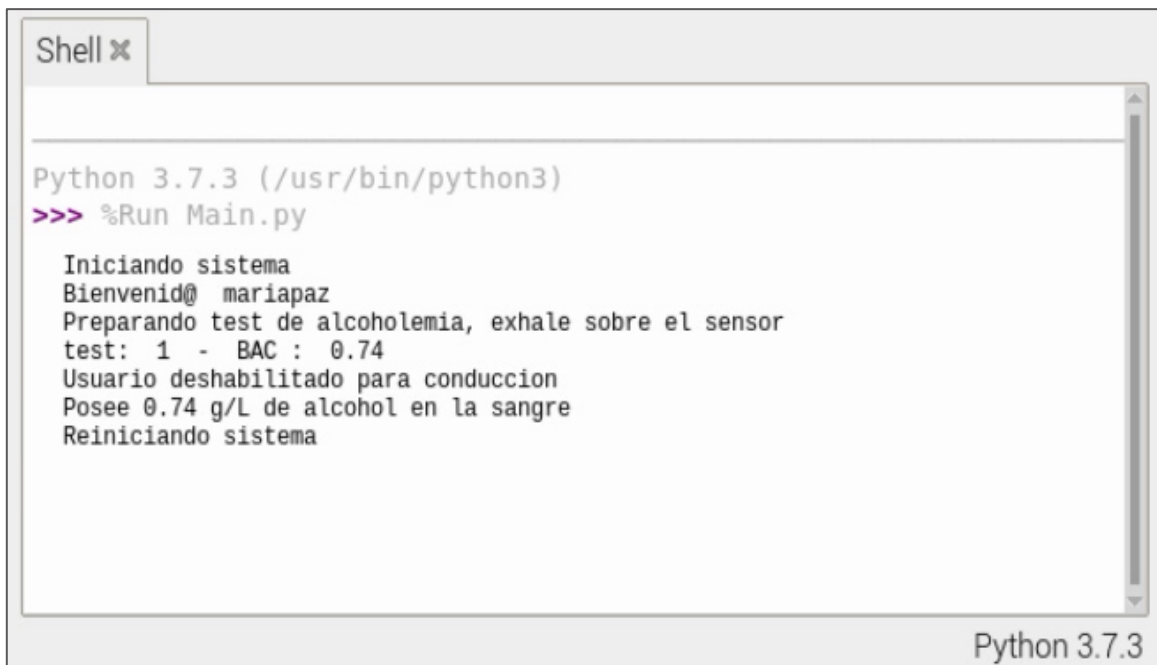
Como se muestra en la figura 3-23, al realizar la primera medición, el programa detecta una cantidad que representa 0,74 g/l de alcohol en la sangre. Inmediatamente se reconoce que esta cantidad es superior al valor límite establecido en 0,3 g/l, y por lo tanto no continúa con el examen de alcoholemia.



Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-21. Inicio del test de alcoholemia a usuario identificado.

Finalmente el programa señala que el conductor se encuentra inhabilitado para realizar la conducción y el sistema de identificación de usuarios vuelve a comenzar.

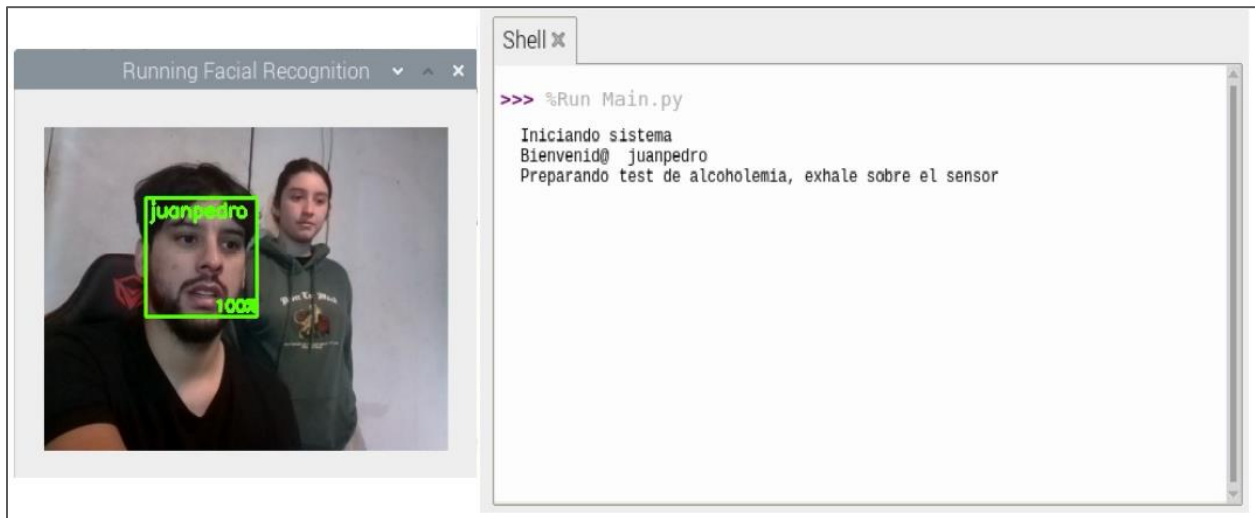


Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-22. Resultados del test de alcoholemia con presencia de alcohol.

3.2.2.2 Test de alcoholemia sin presencia de alcohol

De igual manera que en el caso anterior, luego de realizar la identificación del usuario frente a la cámara, se prepara el test de alcoholemia. Como se ve en la figura 3-23, el usuario que completa primero el 100% de carga, es quien el programa autentificará para la realización del test.



Fuente: Captura de pantalla de la ejecución del programa.

Figura 3-23. Inicio del test de alcoholemia a distinto usuario registrado.

Durante la medición del sensor de alcohol, el usuario marca distintos valores a lo largo de la realización del test, sin embargo, el valor que determina si el test se aprueba, es la máxima cifra recibida en la Raspberry. En esta ocasión el usuario realiza el test hasta completar diez mediciones y así finalmente obtiene el índice de mayor valor.

Como muestra en la figura 3-24, se registró una concentración de alcohol en la sangre equivalente a 0,06 g/l. Al encontrarse en el rango permitido se procede a activar el relé de encendido de motor y se finaliza el programa.



```
>>> %Run Main.py

Iniciando sistema
Bienvenid@ juanpedro
Preparando test de alcoholemia, exhale sobre el sensor
test: 1 - BAC : 0.0
test: 2 - BAC : 0
test: 3 - BAC : 0
test: 4 - BAC : 0.02
test: 5 - BAC : 0.04
test: 6 - BAC : 0.05
test: 7 - BAC : 0.06
test: 8 - BAC : 0.05
test: 9 - BAC : 0.06
test: 10 - BAC : 0.06
Usuario completa la prueba de alcohol exitosamente con 0.06 g/L en la sangre
ID y alcoholemia valida
Sistema de bloqueo desactivado
```

Fuente: Captura de pantalla de la ejecución del programa.

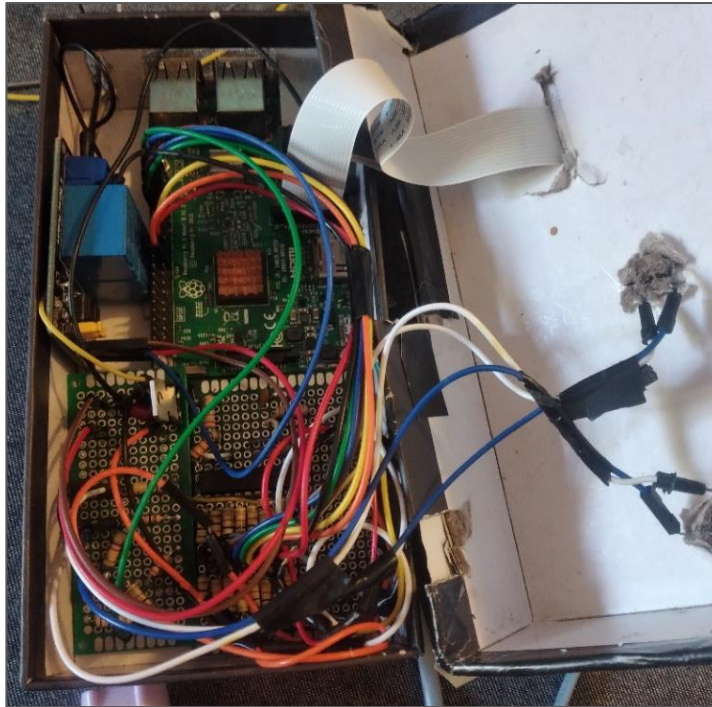
Figura 3-24. Resultados del test de alcoholemia sin presencia de alcohol.

3.2.3 Del prototipo del proyecto

El desarrollo de la creación de la maqueta se diseñó como un artefacto externo al vehículo que se energiza utilizando solo la batería del auto.

Se ve en la figura 3-25, que para integrar todos los componentes del proyecto en un solo sistema, se utilizó una caja de dimensiones 17 x 9 x 5 cm, en donde se encajo la SBC, dos placas con los circuitos soldados y el módulo relé. Además, la tapa de la caja se utilizó como superficie de apoyo, tanto para el sensor de alcohol como de la cámara.

Se realizaron distintos cortes en la caja, de manera que los puertos USB y HDMI estén habilitados para su uso. También se realizaron 3 aberturas en la tapa de la caja con el fin de utilizar ambos módulos fuera del circuito interno del sistema.



Fuente: Captura fotográfica realizada por el autor.

Figura 3-25. Captura fotográfica de la maqueta del proyecto.

3.3 COSTOS DEL PROYECTO

En esta sección se mostrará el detalle de los costos asociados a los componentes, sensores, módulos y también la estimación económica al trabajo realizado y recursos humanos utilizados.

3.3.1 Estudio de mercado de los componentes eléctricos

Para la obtención de los componentes del proyecto se cotizaron en el mercado online de ventas de componentes electrónicos y en tiendas físicas. Los principales componentes de este proyecto son los con mayor costo monetario. Estos son, la SCB Raspberry Pi 3 y el módulo oficial de la cámara Raspberry Pi.

Tabla 3-3. Listado de cantidad y precio de componentes por proveedores.

Componente	Proveedor	Costo por unidad (CLP)	Cantidad	Costo final (CLP)
Raspberry Pi 3 Modelo B+	AliExpress	\$48.390	1	\$48.390
	Amazon	\$69.990	1	\$69.990
	Global Electrónica	\$55.000	1	\$55.000
Módulo Cámara RPI	MercadoLibre	\$29.990	1	\$29.990
	Global Electrónica	\$35.000	0	\$35.000
Módulo MQ3	MercadoLibre	\$7.500	1	\$7.500
Arduino Nano	Global Electrónica	\$7.000	1	\$7.000
Transistor 2n2222a	Global Electrónica	\$300	2	\$600
Diodo Emisor/Receptor	Global Electrónica	\$200	4	\$800
Opto acoplador con Relé	Mercado Libre	\$4.500	1	\$4.500
Kit Resistencias	Global Electrónica	\$3.500	1	\$3.500
Condensador 150pF	Viña electrónica	\$200	1	\$200
Condensador 0.1uF	Viña electrónica	\$200	2	\$400
Kit de cables	Global Electrónica	\$5.000	1	\$5.000

Fuente: Tabla creada por el autor.

Como resultado de la investigación en los precios de los componentes, se eligieron las opciones con los valores más accesibles, obviando las páginas con tiempos de envío demasiado largos, y se ordenaron como muestra la tabla 3-3. Existen diferencias de hasta un 30% en el valor de componentes dependiendo del país que se genera el envío, o dependiendo de la empresa mediadora, como es el caso de *Amazon* y *AliExpress*.

3.3.2 Evaluación de costos de recursos humanos

Para realizar una estimación del valor que tendría la mano de obra por horas trabajadas en este proyecto, se investigó el costo medio de la mano de obra por hora en el Instituto Nacional de Estadística (INE). Según la edición n° 296 del boletín estadístico sobre los índices de remuneraciones y costos de la mano de obra en el mes de Mayo del año 2023 publicado por el INE, se señala que el costo medio de la mano de obra por hora total fue \$7.377.

Ahora bien, para poder realizar el cálculo del costo por horas trabajadas por hombre de este proyecto, se tomó una estimación de 120 horas trabajadas, que abarca la investigación, diseño y la implementación de la solución. Se muestra en la tabla 3-4, que el costo total por mano de obra es igual a \$885.240, y tomando como referencia el valor de la Unidad de Fomento (UF) equivalente a \$36.049 para el día 31 de Julio del 2023, se calcula el costo total en UF.

Tabla 3-4. Costo mano de obra.

Horas dedicadas	Unidad financiera	Costo horas-hombre	Costo total por mano de obra
120	CLP	\$7.377	\$885.240
	UF	0,20	23,72

Fuente: Tabla creada por el autor.

3.3.3 Costo total del proyecto

En este punto el costo total de la realización del proyecto. Esta cifra se calcula sumando la estimación de costos humanos y materiales.

A continuación, se mostrará el listado de los componentes seleccionados con los respectivos precios y se calculará la totalidad de los costos. Tal como se ve en la tabla 3-5, se obtiene el valor de los costos totales de los materiales usados, y que es equivalente a \$116.490.

Tabla 3-5. Costos finales asociados a la compra de los componentes.

Componente	Costo por unidad (CLP)	Cantidad	Costo final (CLP)	Costo final (UF)
Raspberry Pi 3 Modelo B+	\$55.000	1	\$55.000	1,52
Módulo Cámara RPI	\$29.990	1	\$29.990	0,83
Módulo MQ3	\$7.500	1	\$7.500	0,20
Arduino Nano	\$7.000	1	\$7.000	0,19
Transistor 2n2222a	\$300	2	\$600	0,01
LED emisor/receptor	\$200	4	\$800	0,02
Optoacoplador con Relé	\$4.500	1	\$4.500	0,12
Kit Resistencias	\$3.500	1	\$3.500	0,09
Condensador 150pF	\$200	1	\$200	0,00
Condensador 0.1uF	\$200	2	\$400	0,01
Kit de cables	\$5.000	1	\$5.000	0,13
		Costo total	\$116.490	3,23

Fuente: Tabla creada por el autor con base en los precios cotizados de los componentes.

Ahora, para calcular el costo global que requiere el proyecto, se suma el costo total de mano de obra junto al costo total de los materiales. En la tabla 3-6 se ven los valores de los costos asociados y se calcula el costo global del desarrollo del proyecto en \$1.001.730 o 27,78 UF.

Tabla 3-6. Costo final del proyecto.

Costos totales	Costo total por componentes y materiales	Costo total por mano de obra	Costo final del proyecto
CLP	\$116.490	\$885.240	\$1.001.730
UF	2,23	23,72	27,78

Fuente: Tabla creada por el autor.

CONCLUSIONES

A partir del trabajo de investigación y de la construcción del proyecto expuesto, se arriba a las siguientes conclusiones:

La investigación sobre los accidentes automovilísticos en Chile permite establecer al consumo de alcohol como una de sus principales causas. La implementación del sistema de bloqueo vehicular mediante la aplicación de un alcoholímetro electrónico y un detector facial, aumenta la seguridad vial al disminuir los posibles casos de siniestros vehiculares.

En base a la recolección de datos estadísticos sobre seguridad vial se concluye que la conducción con ingesta de alcohol se relaciona con los índices de accidentes automovilísticos, ya que la probabilidad de que ocurran accidentes, aumenta sustancialmente cuando se está en presencia de alcohol.

Se propone solucionar esta problemática mediante el diseño de un alcoholímetro como una medida mínima para aumentar la seguridad vial. Por ende, se configura el sensor de gas MQ3 para realizar la detección de alcohol. Luego de la implementación de este módulo, se considera que mantener un control de temperatura sobre el área del sensor de alcohol mejora los resultados de éste, ya que el funcionamiento está directamente relacionado con las condiciones ambientales presentes, el uso de un sensor de humedad podría afinar la medición del gas deseado.

Se considera como un segundo mecanismo de seguridad vial, la implementación de un sistema de reconocimiento facial que contempla una base de datos de usuarios autorizados. Parte de este proyecto busca demostrar la utilidad de las redes neuronales para resolver problemas relacionados con la detección de patrones, específicamente con la detección de rasgos faciales. La detección de caras está sujeta a la capacidad de la placa en transmitir la información de la cámara a la memoria interna y de calcular las múltiples funciones sobre la imagen. Una configuración "liviana" de video, si es que se están realizando otros procesos de fondo, omitir la muestra de la imagen modificada.

Finalmente, se logra que el sistema de encendido del vehículo se active en caso de que el usuario detectado cuente con los requisitos de validación, de lo contrario se inhabilita el sistema. Este proyecto es de utilidad y se promociona para ser instalado en vehículos de transporte, o en vehículos de uso personal ya que no requiere ningún otro tipo de conexión más que la batería del vehículo.

BIBLIOGRAFÍA

ARANCIBIA, Pamela y LEYTON, Felipe. El consumo del alcohol en Chile: Situación epidemiológica [en línea].

<https://www.senda.gob.cl/wp-content/uploads/media/estudios/otrosSEDA/2016_Consumo_Alcohol_Chile.pdf> [consulta: 21 Noviembre 2022].

ANWAR, Aqeel. Difference between AlexNet, VGGNet, ResNet, and Inception [en línea].

<<https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaecccc96>> [consulta: 21 Noviembre 2022].

BRANDING, Nicklas, RINGHAGE, Oliver y SCHENKEL, Timmy. A comparative study of facial recognition techniques [en línea].

<<https://www.diva-portal.org/smash/get/diva2:1327708/FULLTEXT01.pdf>> [consulta: 21 Noviembre 2022].

CHILE, CONASET. Costo social de los siniestros de tránsito en Chile 2020 [en línea].

<<https://conaset.cl/wp-content/uploads/2021/05/Costos-accidentes-2020.pdf>> [consulta: 21 Noviembre 2022].

CHILE, SENDA-MINSAL. El consumo de alcohol en Chile: situación epidemiológica [en línea].

<https://www.senda.gob.cl/wp-content/uploads/media/estudios/otrosSEDA/2016_Consumo_Alcohol_Chile.pdf> [consulta: 21 Noviembre 2022].

COMISIÓN Nacional de Seguridad de Tránsito. Ley tolerancia cero [en línea]. <<https://www.technologyreview.com/2015/02/16/169357/the-face-detection-algorithm-set-to-revolutionize-image-search/>> [consulta: 21 Noviembre 2022].

FERGUS, Rob y ZEILER, Matthew. Visualizing and understanding convolutional neural networks. Dept. of Computer Science, New York University, USA. Springer, Cham, 2014.

GEEKSFORGEEKS. Artificial Neural Network Terminologies [en línea]. 5 de Abril, 2022. <<https://www.geeksforgeeks.org/artificial-neural-network-terminologies>> [consulta: 10 Agosto 2023].

GEITGEY, Adam. Face recognition 1.3.0: Recognize faces from Python or from the command lines [en línea]. Índice de paquetes de datos (PyPI). 20 de Febrero, 2020. <<https://pypi.org/project/face-recognition/>> [consulta: 10 Agosto 2023].

INE (Instituto Nacional de Estadísticas, Chile). Boletín estadístico: Índices de remuneraciones y costo de la mano de obra [en línea]. Edición nº296. 6 de Julio, 2023. <https://www.ine.gob.cl/docs/default-source/sueldos-y-salarios/boletines/español/base-anual-2016-100/2023/mayo-2023.pdf?sfvrsn=1787e397_4> [consulta: 10 Agosto 2023].

JHONES, Alan. The Relationship between Blood Alcohol Concentration (BAC) and Breath Alcohol Concentration (BrAC): A Review of the Evidence [en línea]. <https://www.researchgate.net/publication/266580842_The_Relationship_between_Blood_Alcohol_Concentration_BAC_and_Breath_Alcohol_Concentration_BrAC_A_Review_of_the_Evidence> [consulta: 21 Noviembre 2022].

MITTAL, Karthik. A Gentle Introduction Into The Histogram Of Oriented Gradients [en línea]. Medium. 20 de diciembre, 2020. <<https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa>> [consulta: 10 Agosto 2023].

NAVEEN, Joshi. 7 Types Of Artificial Intelligence [en línea]. <<https://www.forbes.com/sites/cognitiveworld/2019/06/19/7-types-of-artificial-intelligence/?sh=5d968eb0233e>> [consulta: 21 Noviembre 2022].

N., Junaid. Convolutional Neural Networks [en línea]. <<https://iq.opengenus.org/convolutional-neural-networks/>> [consulta: 21 Noviembre 2022].

OPEN Source Computer Vision. Cascade Classifier [en línea]. <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html> [consulta: 21 Noviembre 2022].

RODRÍGUEZ, Rubén. Aprendizaje de redes neuronales [en línea].
 <<https://lamaquinaoraculo.com/computacion/aprendizaje-de-redes-neuronales/#:~:text=A%20diferencia%20de%20otros%20métodos%20de%20cómputo%2C%20la,principales%20paradigmas%20de%20aprendizaje%2C%20supervisado%20y%20no%20supervisado>> [consulta: 10 Agosto 2023].

SKANSI, Sandro. Convolutional neural networks. En su: Introduction to deep learning: from logical calculus to artificial intelligence. Croacia, University of Zagreb: Springer, Cham. 2008. pp 121-132.

SKILLKATE IA. Histogram of Oriented Gradients: Simplest Intuition [en línea]. Medium. 13 de Agosto, 2022.
 <<https://medium.com/@skillcate/histogram-of-oriented-gradients-hog-simplest-intuition-2392995f8010>> [consulta: 10 Agosto 2023].

TOMASI, Carlo. Histograms of oriented gradients [en línea]. Computer Vision Sampler, 1-6. 2017.
 <<https://courses.cs.duke.edu/fall17/compsci527/notes/hog.pdf>> [consulta: 10 Agosto 2023].

ZHANG, Cha y ZHANG, Zhengyou. A Survey of Recent Advances in Face Detection [en línea].
 <<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/facedetsurvey.pdf>> [consulta: 21 Noviembre 2022].