

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA



TRABAJO DE FIN DE CARRERA (MTR280)

SISTEMA DE RECONOCIMIENTO DE ESTILO DE CONDUCCIÓN EN VEHÍCULOS DEL SISTEMA DE TRANSPORTE PÚBLICO

NOMBRE: Héctor David Aguirre Arista

PROFESORES: Ericka Patricia Madrid Ruiz
Elizabeth Roxana Villota Cerna

ASESORES: Jhon Portella Delgado
Elizabeth Roxana Villota Cerna

HORARIO: 10M2

CÓDIGO: 20131733

Diciembre 2018

Resumen

El reconocimiento de patrones usando algoritmos de machine learning e inteligencia artificial ha incrementado su popularidad debido a toda la información que actualmente tenemos disponible. Estas herramientas juegan un papel muy importante en el sector de transporte debido a su capacidad de poder optimizar el uso de recursos como el combustible o la energía eléctrica.

Esta optimización tiene beneficios en muchos ámbitos. Económicamente el usar de manera más eficiente el combustible o la energía eléctrica reduce costos. Ambientalmente el uso eficiente de combustible reduce las emisiones de CO₂, logrando reducir la contaminación en las ciudades. Por último, el estilo de conducción que usa eficientemente la energía esta asociado a un estilo de conducción menos agresivo y más seguro. Algo que es de vital importancia en países como el Perú en donde se producen cientos de muertes al año debido a accidentes de tránsito.

En la presente tesis, se buscará desarrollar un sistema capaz de reconocer el estilo de conducción de un chofer del transporte público de Lima, otorgándole retroalimentación en tiempo real y monitorizándolo para que pueda mejorar su estilo de conducción y de esta manera se reduzca el consumo del combustible utilizado, las emisiones generadas y se aumente la seguridad vial en la ciudad. Para lograrlo se utilizan algoritmos de machine learning como PCA, Fast ICA, Redes Nueronales, Random Forest.

La metodología seguida en esta tesis comenzará con el planteamiento y delimitación del problema y los principales objetivos. Luego, se obtendrá información relevante al estado del arte actual y se desarrollarán conceptos de solución que estén orientados a solucionar el problema. Después, se desarrollará a detalle el concepto de solución óptimo para ello se crea una estructura de funciones y se procede a ejecutar el diseño electrónico , mecánico y de software.

Índice

Lista de Figuras	vii
Lista de Tablas	x
Introducción	1
1 Planteamiento de la Problemática	3
1.1 Motivación General	3
1.2 Objetivo general	4
1.3 Objetivos específicos	4
2 Estado del Arte	5
2.1 Términos relacionados con el estilo de conducción	5
2.2 Estado del arte según algoritmos usados	6
2.2.1 Algoritmos basados en reglas	6
2.2.2 Algoritmos basados en datos	8
2.3 Estado del arte según sensores usados	16
2.3.1 Acelerómetros de bajo costo	16
2.3.2 Smartphone	16
2.3.3 Inertial Measurement Unit	17
2.3.4 GPS	17
2.4 Feedback	17

3 Diseño Conceptual	19
3.1 Requerimientos del sistema	19
3.2 Modelo Black Box	22
3.3 Estructura de Funciones	22
3.3.1 Dominio Mecánico	24
3.3.2 Dominio de Energía	24
3.3.3 Dominio de Sensores	25
3.3.4 Dominio de Comunicación	25
3.3.5 Dominio de Procesamiento	26
3.3.6 Dominio de Interfaz	26
3.3.7 Dominio de Reconocimiento de Estilo de Conducción	27
3.4 Matriz morfológica por dominio	27
3.4.1 Dominio Mecánico	27
3.4.2 Dominio Energético	29
3.4.3 Dominios de Procesamiento y de Comunicación	30
3.4.4 Dominio de Sensores	30
3.4.5 Dominio de Interfaz	32
3.4.6 Dominio de Reconocimiento de estilo de conducción	33
3.5 Conceptos integrados de solución	36
3.5.1 Concepto integrado de solución 1	37
3.5.2 Concepto integrado de solución 2	39
3.5.3 Concepto integrado de solución 3	41
3.6 Evaluación de conceptos de solución	42
3.6.1 Evaluación técnica	42
3.6.2 Evaluación económica	43
3.6.3 Evaluación de soluciones	44
4 Diseño del sistema mecatrónico	46

Índice	v
4.1 Integración del sistema mecatrónico	46
4.2 Funcionamiento del sistema mecatrónico	46
4.3 Diseño electrónico	48
4.3.1 Diagrama de bloques	48
4.3.2 Componentes electrónicos	50
4.3.3 Diagramas electrónicos	62
4.3.4 Diseño del PCB	69
4.4 Diseño mecánico	73
4.4.1 Diseño del Case Principal	73
4.4.2 Selección del brazo de la pantalla	75
4.4.3 Diseño del case de la pantalla	76
4.5 Diseño del software de reconocimiento de estilo de conducción	78
4.5.1 Software de los dispositivos cliente	79
4.5.2 Software del servidor	81
4.5.3 Software de la página web	83
5 Algoritmo de clasificación	84
5.0.1 Recolección de datos	85
5.0.2 Segmentación de los datos	88
5.0.3 Extracción de características	89
5.0.4 Reducción de características	90
5.0.5 Entrenamiento del algoritmo	91
6 Costos del Sistema	96
6.1 Costos de componentes electrónicos	96
6.2 Costos de fabricación y componentes mecánicos	97
6.3 Costo total	97
7 Conclusiones y Recomendaciones	98

Bibliografía	99
Anexo A Datasheets	102
A.1 IMU	103
A.2 GPS	104
A.3 GPRS/GSM	106
Anexo B Implementación en python del algoritmo de clasificación	107
Anexo C Lista de Materiales y Costos	126

Lista de Figuras

2.1	Terminología estilo de conducción	5
2.2	Resultados del sistema de reconocimiento de estilo de conducción	7
2.3	Tasa de consumo de combustible según diferentes estilos de conducción . .	9
2.4	Comparación de señales usando distancia Euclidiana y DTW	11
2.5	Algoritmo de <i>Dynamic Time Warping</i>	12
2.6	Topología de una Red Neuronal usada para Extreme Learning Machines . .	14
2.7	MPU-5050 de Invensense	17
3.1	Modelo de Black Box del sistema.	22
3.2	Estructura de Funciones.	23
3.3	Dominio Mecánico de la estructura de funciones.	24
3.4	Dominio Eléctrico de la estructura de funciones.	24
3.5	Dominio de Sensores de la estructura de funciones.	25
3.6	Dominio de Comunicación de la estructura de funciones.	25
3.7	Dominio de Procesamiento de la estructura de funciones.	26
3.8	Dominio de Interfaz de la estructura de funciones.	26
3.9	Dominio de Reconocimiento de Estilo de Conducción de la estructura de funciones.	27
3.10	Concepto integrado de solución 1.	38
3.11	Concepto integrado de solución 2.	40
3.12	Concepto integrado de solución 3.	42

3.13 Diagrama de evaluación	45
4.1 Dispositivo de clasificación de estilo de conducción.	47
4.2 Dispositivo de clasificación de estilo de conducción montado en un auto.	47
4.3 Diagrama de bloques.	49
4.4 ICM-20648, IMU de 6 grados de libertad.	51
4.5 NEO-M8N, módulo GPS	52
4.6 Cobertura de la red 2G de Claro	53
4.7 SIM 800L, módulo GPRS/GSM	53
4.8 Adafruit 1.54" 240x240 Wide Angle TFT LCD Display - ST7789	55
4.9 ELM327 - OBD2 interface	56
4.10 Diagrama de Bloques del ESP32 [1].	57
4.11 ESP-WROOM-32 [2].	58
4.12 Diagrama de bloques de las etapas de alimentación con la corriente máxima de cada dispositivo.	59
4.13 Diagrama de bloques del sistema con corrientes promedio.	62
4.14 Esquemático del módulo ICM-20648.	63
4.15 Esquemático del módulo NEO-M8N.	64
4.16 Esquemático del módulo SIM800L.	64
4.17 Esquemático del módulo SIM8055.	65
4.18 Esquemático las los indicadores LED.	66
4.19 Esquemático bornera para conectar la pantalla.	66
4.20 Esquemático de la implementación del ESP-WROOM-32.	67
4.21 Esquemático de la implementación del TPS563210.	68
4.22 Esquemático de la implementación del LT1764A.	69
4.23 Diseño del PCB.	70
4.24 Parte superior del PCB.	70
4.25 Parte inferior del PCB.	71

4.26 Parte superior del PCB con componentes.	72
4.27 Parte inferior del PCB con componentes.	72
4.28 Vista lateral del PCB.	73
4.29 Unión de la tarjeta electrónica con el case.	73
4.30 Inserto M2 x 2.5 para plástico.	74
4.31 Unión entre la parte superior e inferior del Case Principal	74
4.32 Vista de explosión del case principal.	75
4.33 Elemento Loc-Line.	76
4.34 Detalles del ensamble de los elementos Loc-Line con las cajas electrónicas	76
4.35 Detalles del ensamble del case de la pantalla	77
4.36 Vista de explosión del case de la pantalla.	77
4.37 Diagrama de bloques del software.	78
4.38 Diagrama de flujo del software de los dispositivos cliente.	80
4.39 Diagrama de flujo del software del servidor.	81
4.40 Diagrama de flujo de la función Procesar Datos.	82
4.41 Diagrama de flujo de la función Clasificar maniobra.	83
5.1 Diagrama de flujo del desarrollo del algoritmo de clasificación.	84
5.2 Partes de un dataset.	85
5.3 Duración de cada viaje grabado en el dataset.	86
5.4 Número de instancias para cada clase.	87
5.5 Número de instancias para cada clase, con clase dividida en 2 grupos.	88
5.6 Número de puntos para cada clase, con clase dividida en 3 grupos.	88
5.7 Varianza para cada componente obtenido a partir de PCA.	91
5.8 Componentes gráficos entre sí luego de usar.	93
5.9 Componentes gráficos entre sí luego de aplicar Fast ICA.	94
5.10 Componentes gráficos entre sí luego de normalizar.	95

Lista de Tablas

2.1	Descripción de los clusters obtenidos	11
2.2	Porcentaje de reconocimiento de maniobras de conducción	13
2.3	Resumen de sensores usados en las investigaciones mencionadas	16
3.1	Lista de Requerimientos página 1.	20
3.2	Lista de Requerimientos página 2.	21
3.3	Matriz Morfológica del Dominio Mecánico	28
3.4	Conceptos de solución mecánicos propuestos	28
3.5	Matriz Morfológica del Dominio Energético	29
3.6	Conceptos de solución eléctricos propuestos	29
3.7	Matriz Morfológica de los Dominios de Procesamiento y Comunicación . .	30
3.8	Conceptos de solución propuestos del dominio de comunicación	31
3.9	Matriz Morfológica del Dominio de Sensores	31
3.10	Conceptos de solución propuestos del dominio de sensores	32
3.11	Matriz Morfológica del Dominio de Interfaz	32
3.12	Conceptos de solución propuestos del dominio de interfaz	33
3.13	Matriz Morfológica de Reconocimiento de estilo de conducción	34
3.14	Conceptos de solución propuestos del dominio de reconocimiento de estilo de conducción	35
3.15	Conceptos de solución propuestos	36
3.16	Evaluación técnica	43

3.17	Evaluación económica	44
3.18	Evaluación de soluciones	44
4.1	Sensibilidad y Rango del ICM-20648	50
4.2	Condiciones de Operación del ICM-20648	51
4.3	Características del NEO-M8N	52
4.4	Características del SIM 800L	54
4.5	Alternativas de pantallas	55
4.6	Protocolos y Circuitos integrados	56
4.7	Características del ESP-WROOM-32	58
4.8	Consumo de los componentes del sistema	59
4.9	Características del TPS563210	60
4.10	Características del LT1764A	61
4.11	Anchos de pista mínimos	71
4.12	Datos a enviar con su tamaño en bytes	79
5.1	Características o features del dataset	86
5.2	Parámetros para series de tiempo	89
5.3	Algoritmo de PCA	90
5.4	Parámetros de los modelos entrenados.	92
5.5	resultados de los modelos entrenados.	92
6.1	Resumen de costos electrónicos	96
6.2	Resumen de costos mecánicos	97
6.3	Resumen de costos totales	97
C.1	Costo de componentes electrónicos (parte 1)	126
C.2	Costo de componentes electrónicos (parte 2)	127

Introducción

El transporte es una actividad cotidiana que tiene un lugar muy importante en nuestras vidas. Sin embargo, en muchas ciudades esta actividad se realiza de manera ineficiente y esto tiene como consecuencia una larga lista de problemas, como el aumento de congestión vehicular, la disminución de la seguridad vial, el aumento de emisiones de CO₂, etc.

Uno de los principales motivos del transporte ineficiente es la conducta de los conductores al manejar. Según [3] el 80% de los peruanos conduce de manera agresiva y es el país con peor desempeño en conducción de 38 países analizados. Esta conducta agresiva está directamente relacionada con algunos de los problemas del transporte ineficiente.

Uno de ellos es el aumento de emisiones de CO₂. El transporte está liderada por el uso de combustibles fósiles y, aunque los autos eléctricos han empezado a ganar terreno en el sector automovilístico, estos aún representan tan solo el 1.34% de las ventas totales de autos en el mundo [4]. El transporte se registró como la fuente mas grande de emisión de CO₂ en el 2016 en Europa y Estados Unidos con el 27% y 28% de las emisiones de gases de efecto invernadero respectivamente [5]. Entonces el transporte ya representa una gran parte de las emisiones de CO₂. Sin embargo, en [6] se demostró que un conductor agresivo puede consumir hasta 20 % más combustible que uno calmado, este consumo extra de combustible conlleva a un incremento del 50% en emisiones de CO₂. Se encuentra entonces en el estilo de conducción la oportunidad de reducir en gran medida las emisiones de este gas de efecto invernadero.

Por otro lado, otras de las consecuencias de la conducción agresiva es la seguridad vial. En el Perú en el año 2017 murieron un total de 772 personas debido a accidentes de tránsito [7]. Las mayores causas de los accidentes de tránsito en el Perú son el exceso de velocidad (32%) y la imprudencia del conductor (28%) [8]. Estas dos causas son errores humanos que están directamente relacionados con el estilo de manejo de los conductores.

Lo expuesto anteriormente convierte al estilo de conducción de un usuario en un factor relevante para la seguridad vial y para el cuidado del medio ambiente. Por lo que la presente

tesis desarrollará un sistema que sea capaz de caracterizar el estilo de conducción de un usuario para luego proporcionar un feedback que lleve al conductor a mejorar sus estilo de manejo y como consecuencia reducir las emisiones generadas, el combustible consumido e incrementar la seguridad durante la conducción.

1. Planteamiento de la Problemática

1.1. Motivación General

En un futuro, se espera que la mayoría de los autos sean autónomos, probablemente eléctricos y que el manejo del sistema de transporte este interconectado y sea inteligente. Para lograr esta integración del transporte se necesita un entendimiento profundo del comportamiento humano a la hora de realizar la tarea de conducir.

Este conocimiento no sólo será útil en el futuro, sino que en el presente nos ayudaría a tener vehículos que sean conducidos de una manera eficiente y más segura, reduciendo la cantidad de emisiones dañinas para el medio ambiente y la cantidad de accidentes de tránsito.

Por otro lado, actualmente no se cuenta con una forma de monitorizar el comportamiento de los conductores que trabajan en el transporte público. Una persona es capaz de darse cuenta fácilmente cuando el conductor realiza una maniobra temeraria y pone en riesgo la seguridad. Sin embargo, no existen muchos sistemas que puedan reconocer de forma automática o semiautomática este comportamiento.

El mismo hecho de monitorizar el estilo de manejo de un conductor de una manera objetiva y a lo largo de todo su recorrido hace que el conductor cambie su comportamiento al volante y tenga más cuidado a la hora de conducir.

Debido a todas las razones antes mencionadas, el desarrollo de un sistema de reconocimiento de estilos de conducción en Lima es importante.

1.2. Objetivo general

Se propone entonces el diseño de un módulo de reconocimiento de estilo de conducción que pueda adaptarse a cualquier vehículo y que entregue un feedback de acuerdo al estilo actual del conductor para su uso en el sistema de transporte público en Lima.

Este módulo utilizará algoritmos de reconocimiento de patrones y retornará una señal de feedback con el objetivo de advertir al conductor cuando este realice una maniobra agresiva que consuma energía innecesaria o que comprometa la seguridad vial. Además, el estilo de manejo de cada conductor será monitorizado enviando toda la información a un servidor para un análisis estadístico posterior.

Este sistema no sólo permitirá la adopción de un estilo de conducción más eficiente y seguro por parte de los conductores del sistema de transporte público; sino que además, brindará datos e información con la que no se cuenta actualmente y que podría ser utilizada para generar otros servicios tal como el seguimiento de los buses en tiempo real por parte de los usuarios.

1.3. Objetivos específicos

De acuerdo a lo expuesto anteriormente se establecen entonces los siguientes objetivos:

- Definir el concepto de estilo de conducción.
- Definir que parámetros de los vehículos se medirán y seleccionar los sensores que medirán estos parámetros.
- Diseñar un algoritmo o un conjunto de estos para reconocer el estilo de conducción de un conductor de acuerdo a la definición propuesta en esta tesis.
- Diseñar el mecanismo que será usado para entregar feedback al usuario.

2. Estado del Arte

2.1. Términos relacionados con el estilo de conducción

El concepto de "Estilo de conducción" no tiene una definición estándar que sea aceptada en todo el mundo. Al contrario, este concepto involucra una serie de factores que aumentan su complejidad y complican su definición. Debido a esto se usarán los conceptos definidos por Martinez et al. [9].

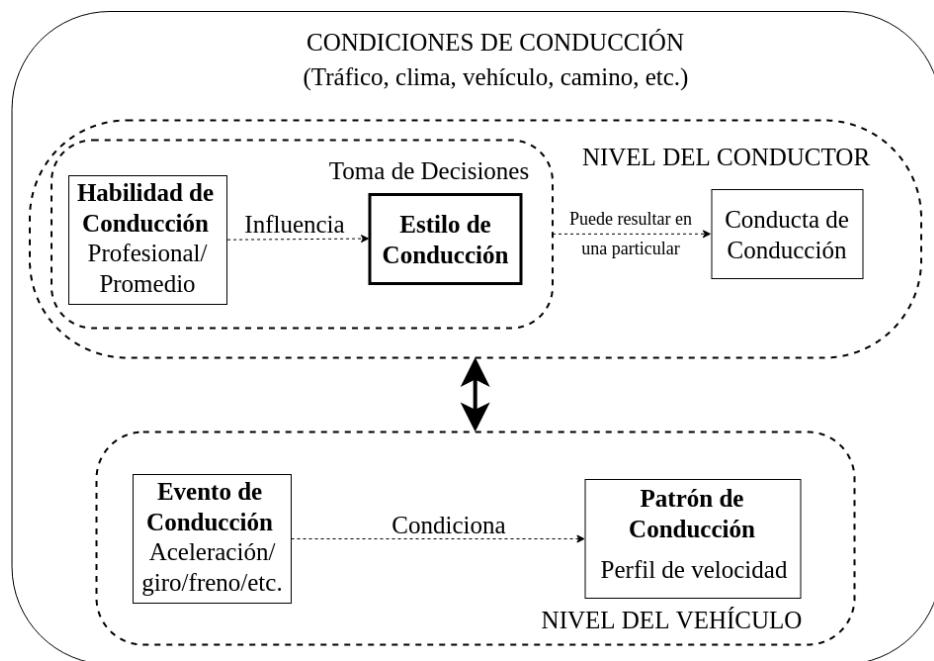


Fig. 2.1 Terminología relacionada con el estilo de conducción [9]

Como se aprecia en la Fig 2.1, se entiende por un evento de conducción como las maniobras que se usan durante la acción de conducir, como por ejemplo: acelerar, desacelerar y girar.

De la misma manera, El patrón de conducción se define como el resultado de los eventos de conducción sujetos a condiciones de manejo, como el clima o el tipo de calzada. Este resultado se puede expresar como un perfil de velocidades, en el que están incluidos todos los datos que se pueden obtener partiendo de este perfil de velocidad, como por ejemplo: duración del viaje, velocidad promedio y demanda de potencia calculada.

La habilidad de conducción es la habilidad que posee el conductor para controlar el vehículo. Este concepto se usa para diferenciar entre un conductor experimentado o profesional de un conductor promedio.

El estilo de conducción es más complejo de definir debido a que para algunos autores involucra factores subjetivos como la actitud del conductor, el humor o el cansancio. Para Dörr et al. [10], el estilo de conducción es la manera en la que la tarea de conducción es realizada. Esto se traduce en la forma en la que el conductor opera el vehículo (Pedal de aceleración, timón, freno, etc.). Esto se diferencia de el patrón de conducción tan solo porque no se asocia con un recorrido en específico sino con un conductor.

También, se puede expresar el estilo de conducción en niveles de agresividad como Aljaafreh et al. [11]. Debido a que la agresividad en los eventos de conducción esta asociada con un mayor consumo de combustible y a menor seguridad vial, definitivamente juega un papel importante dentro del concepto de estilo de conducción. Esta es la forma en la que se definirá el estilo de conducción para esta tesis.

2.2. Estado del arte según algoritmos usados

Se procederá a mostrar las implementaciones e investigaciones desarrolladas en la actualidad clasificados según el tipo de algoritmo que se uso para la caracterización de el estilo de conducción

2.2.1. Algoritmos basados en reglas

Dentro de esta categoría se encuentran algoritmos de clasificación basados en reglas que comprenden el uso de lógica difusa, lógica difusa adaptativa y algoritmos de agrupamiento. Estos algoritmos se caracterizan por estar definidos por *umbrales predefinidos* y son el enfoque más sencillo para tratar de clasificar los estilos de conducción.

Dörr et al. [10] desarrolló un sistema de reconocimiento de estilo de conducción online usando lógica difusa. Este sistema esta implementado usando *Matlab/Simulink* y es

paramétrico. Se puede configurar para ser usado en distintos tipos de vehículos. El sistema detecta la aceleración longitudinal, aceleración lateral, desaceleración, velocidad, brecha de tiempo y activación del sistema autónomo de velocidad crusero. Además determina a través de un Sistema de Navegación el tipo de calzada en el que se encuentra (Se distingue entre trocha, calles urbanas, carreteras pavimentadas y carreteras rurales). Esto lo realiza debido a que el tipo de calzada influencia en gran medida al estilo de conducción. Por ejemplo, en una trocha la mayoría de conductores manejarían a una velocidad suave para no dañar al vehículo, lo cual no necesariamente quiere decir que el mismo conductor conduzca de esa manera en otro tipo de escenarios. Usando Lógica Difusa se definieron 3 niveles de estilo de conducción: *Normal*, *Confortable* y *Deportivo*. Se probó el sistema en una simulación y se obtuvieron los siguientes resultados promedio (Fig. 2.2): 67.8% de clasificaciones correctas y solo 2% de clasificaciones incorrectas (El otro 30.2% pertenece a clasificaciones "Diferentes" que dan una clasificación contigua de estilo de manejo, por ejemplo cuando la clasificación real es *Normal* pero el sistema arroja como resultado *Confortable*, en cambio las clasificaciones incorrectas dan un resultado no contiguo al real)

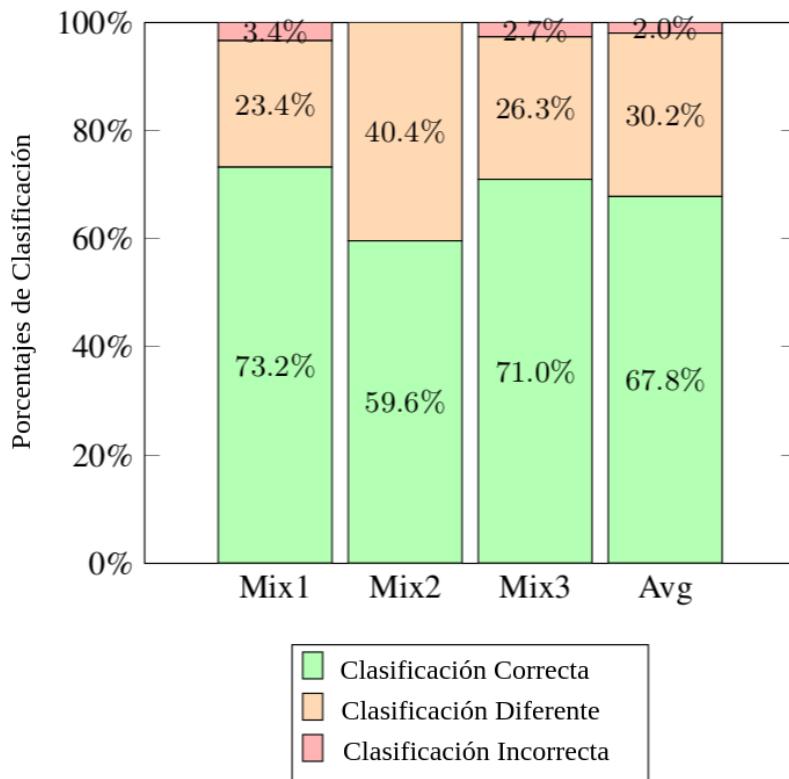


Fig. 2.2 Resultados del sistema de reconocimiento de estilo de conducción en [10]

Usar lógica difusa no es la única manera de clasificar estilos de conducción. Murphrey et al. [12] presentó un método de clasificación basado en el análisis del perfil de la sobreaceleración (la derivada de la aceleración respecto al tiempo). En su investigación define al estilo de manejo como un comportamiento dinámico. Un conductor puede manejar de forma calmada por un momento y de forma agresiva en otro momento. Por este motivo, el método de clasificación que propone predice el estilo de conducción del usuario en tiempo real. El algoritmo funciona de la siguiente manera:

1. Calcula el perfil de sobreaceleración durante una ventana de tiempo predefinido.
2. Calcula la desviación estándar de la sobreaceleración durante toda la ventana de tiempo.
3. Detecta el tipo de calzada actual y el nivel de congestión de tráfico usando el algoritmo propuesto en [13].
4. Calcula la proporción de sobreaceleración dividiendo la desviación estándar con un valor promedio que depende de el tipo de calzada y el nivel de tráfico actuales.
5. Dependiendo del resultado de la división realizada el conductor puede ser clasificado como *Calmado*, *Normal* o *Agresivo* usando reglas con umbrales predefinidos.

Este resultado dependerá mucho de la duración definida para la ventana de tiempo. Se recomendó usar una ventana de duración de 6 a 9 segundos para detectar los cambios de estilo de conducción oportunamente.

Se realizó también una comparación entre los 3 diferentes estilos de conducción con respecto a la tasa de consumo de combustible. Como se puede apreciar en la Fig. 2.3, los conductores clasificados como calmados están asociados a un menor consumo de combustible, mientras que los agresivos a un mayor consumo de combustible.

Se ha presentado dos algoritmos basados en reglas que usan tanto múltiples variables [10] como una sola [12] para predecir el estilo de conducción del usuario. Sin embargo, los resultados de estos sistemas dependen en gran medida del valor de los umbrales escogidos para realizar la clasificación.

2.2.2. Algoritmos basados en datos

Cuando se tiene una cantidad muy grande de datos, es difícil analizarlos para obtener las reglas que guían al algoritmo. Es entonces cuando los algoritmos basados en datos entran en acción. Estos algoritmos, en comparación con los basados en reglas, pueden manejar

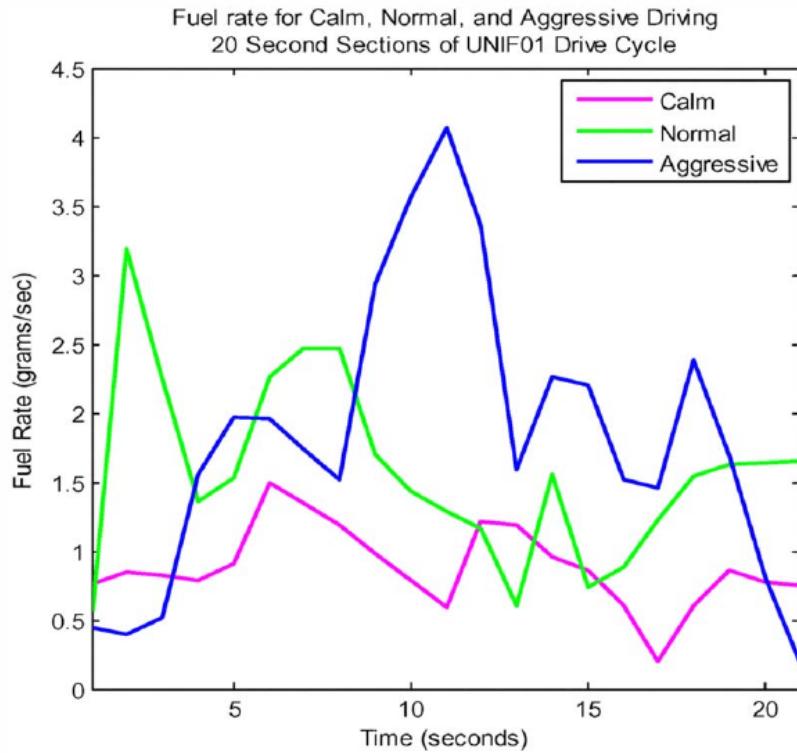


Fig. 2.3 Tasa de consumo de combustible según diferentes estilos de conducción [12].

una mayor cantidad de datos y deducir umbrales consistentes con estos datos sin que estos dependan de un experto en el tema. A continuación se presentarán los distintos tipos de algoritmos basados en datos y su uso en el reconocimiento de estilos de conducción.

A. Algoritmos de aprendizaje de máquinas no supervisados

Los algoritmos no supervisados no necesitan que los datos obtenidos por medio de los sensores en la actividad de conducción estén etiquetados. Es decir, si se registra los datos de un ciclo de conducción para un conductor, este registro no necesita estar asociado con el estilo de conducción que mantuvo este conductor. Esta clase de algoritmos busca patrones en los datos y los separa teniendo en cuenta sus similitudes y diferencias en diferentes grupos sin etiquetar.

Constantinescu et al. [14] propuso el análisis de el estilo de conducción por medio de dos algoritmos: Hierarchical Cluster Analysis (HCA) y Principal Component Analysis (PCA).

HCA es un algoritmo de *machine learning* que trata de categorizar a cada individuo en grupos con características similares. Es muy útil cuando no se conoce con exactitud el

número de grupos. Por medio de un análisis estadístico trata de minimizar la varianza dentro de cada grupo y a la vez aumentarla entre diferentes grupos. En este caso, el algoritmo de agrupamiento jerárquico consiste en empezar con grupos consistentes de un solo miembro y consecutivamente ir combinando los dos grupos más cercanos, hasta solo tener un grupo.

PCA es un método estadístico que usa una transformación para convertir un posible conjunto de variables relacionadas entre si en otro conjunto menor de variables linealmente independientes, en las que la varianza sea maximizada.

Para este caso en particular se trabajo con dispositivo GPS que se encargaba de medir la velocidad y aceleración del vehículo monitorizado a una frecuencia de muestreo de 1 Hz. De esta data se escogieron los siguientes parámetros:

1. Velocidad por encima de los 60 Km/h.
2. Velocidad promedio.
3. Desviación estándar de la velocidad.
4. Desviación estándar de la aceleración.
5. Promedio de la aceleración positiva.
6. Desviación estándar de la aceleración positiva.
7. Promedio de la aceleración negativa o frenado.
8. Desviación estándar del frenado.
9. Trabajo mecánico, que se calculó sumando toda la energía cinética positiva requerida para aumentar la velocidad del vehículo.

Luego del análisis realizado se encontraron 6 grupos o clusters y se describió cada uno de estos como se puede observar en el Tabla 2.1 usando los 5 componentes principales resultantes al aplicar PCA.

Los algoritmos no supervisados tienen una gran ventaja, no se necesitan conocer las etiquetas de clasificación *a priori*. Esto significa que la definición de etiquetas no limita o influencia de alguna manera a los resultados obtenidos, ya que los grupos o clusters formados dependen solo de los datos usados. Sin embargo, luego de hallar los grupos se necesita realizar un análisis para determinar su naturaleza.

Tabla 2.1 Descripción de los clusters obtenidos en [14].

Cluster	Agresividad	Velocidad	Aceleración	Frenado
1	Moderadamente baja	Baja-Moderada	Moderada	Suave-Moderado
2	Muy baja	Baja-Moderada	Baja-Moderada	Suave-Moderado
3	Moderadamente alta	Moderada	Moderada	Repentino
4	Neutral	Moderada	Alta	Moderado
5	Neutral	Moderada-Alta	Baja-Moderada	Moderado-Repentino
6	Alta	Alta	Alta	Repentino

B. Algoritmos de aprendizaje de máquinas supervisados

Los algoritmos supervisados, a diferencia de los no supervisados, necesitan tener un conocimiento previo de los datos que se tienen. Es decir, cada uno de los datos recolectados deben estar asociados a una etiqueta o clase ya predefinida.

Uno de los algoritmos más utilizados es el de K-Nearest Neighbors (k-NN), que se utilizó junto a Dynamic Time Warping (DTW) en el sistema de reconocimiento propuesto por Johnson and Trivedi [15]. El objetivo de este sistema es reconocer maniobras agresivas para dar un feedback al conductor y monitorizarlo, mejorando así la seguridad vial. Para lograrlo usaron los sensores presentes en un smartphone (acelerómetro, giroscopio, GPS y cámara frontal). Este sistema trabaja grabando la información de todos los sensores durante 5 minutos, luego analiza si ha ocurrido alguna maniobra potencialmente agresiva. Si una maniobra de este tipo ocurrió, conserva la información para un análisis posterior. En cambio, si no ocurrió ninguna maniobra de este tipo, borra la data para ahorrar espacio en disco.

Para la clasificación de cada maniobra detectada se usa Dynamic Time Warping (DTW), el cual es un algoritmo que es capaz de analizar la similaridad entre dos señales que no necesariamente tengan la misma duración y probablemente tengan un desfase. En la Fig. 2.4 se puede observar una comparación entre una comparación de una distancia Euclidiana y otra usando DTW en dos señales muy parecidas en forma pero que no se encuentran en fase.

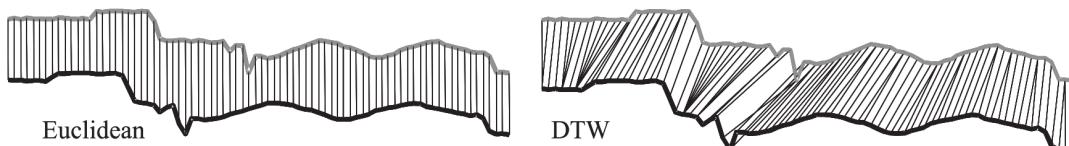


Fig. 2.4 Comparación de señales usando distancia Euclidiana y DTW [16].

El algoritmo consiste en crear una matriz de deformación $n \times m$, siendo n y m los números de puntos en cada señal. Esta matriz se llena calculando distancias entre cada punto, sin importar si se encuentran en el mismo instante temporal. Luego se dibuja el camino con la distancia más corta que une el inicio de las señales con el final de estas Fig. 2.5 (B). La suma de distancias en este camino se define como la distancia entre las dos señales analizadas y se puede usar para comparar la similitud y clasificar señales usando modelos. Información detallada sobre este algoritmo se puede encontrar en [16].

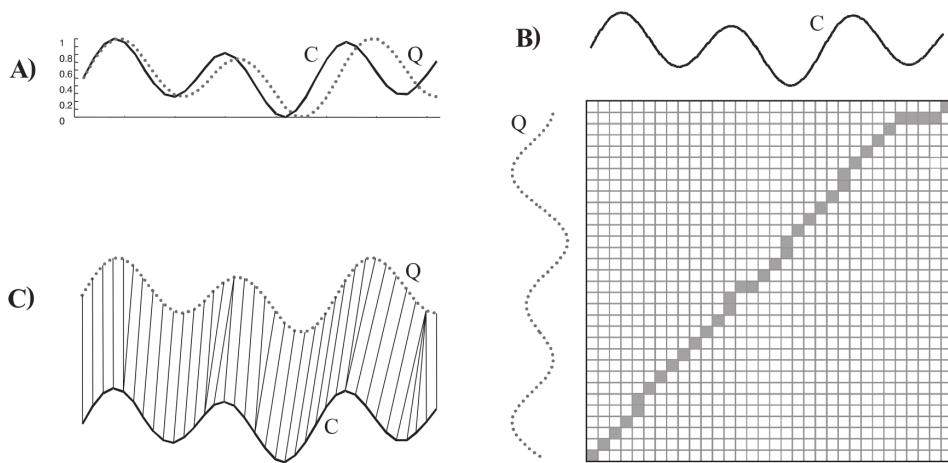


Fig. 2.5 **A)** Dos señales con forma muy similar pero desfasadas. **B)** Matriz de deformación usada para encontrar el camino de deformación. **C)** El resultado del alineamiento y deformación usando DTW. [16].

Para poder usar DTW se necesitó contar con señales modelo para cada tipo de maniobra a detectar. Se tuvo un total de 120 señales modelo en total, cada una asociada con una maniobra específica. Usando DTW se puede obtener una medida de distancia entre las señales a clasificar y las señales modelo. Luego se aplicó K-Nearest Neighbors (k-NN) para clasificar cada señal. Lo que hace k-NN es tener en guardadas las señales modelos y al momento de clasificar una nueva señal calcula la distancia con las señales modelos, selecciona las k distancias menores y dependiendo de la clase que tenga más señales cercanas, esta nueva señal es clasificada.

Para realizar la clasificación de señales solo se utilizó los datos de g_x , a_y y e_x (giroscopio en x , aceleración en y y ángulo de rotación de Euler en x respectivamente). Se puede observar en el Tabla. 2.2 los porcentajes de reconocimiento exitoso por cada maniobra.

Otro algoritmo aplicado usualmente al reconocimiento de estilos de manejo es el uso de *Redes Neuronales*. Echanobe et al. [17] utilizó redes neuronales para clasificar estilos de manejo y *Algoritmos Genéticos* para optimizar estas redes e incluso elegir los parámetros

Tabla 2.2 Porcentaje de reconocimiento exitoso de maniobras de conducción [15].

Maniobra	Porcentaje de reconocimiento exitoso (%)
Giro a la derecha (90°)	92
Giro a la izquierda (90°)	83
Vuelta en U (180°)	77
Giro a la derecha agresivo	100
Giro a la izquierda agresivo	100
Vuelta en U agresiva	100
Cambio de carril agresivo (derecha)	100
Cambio de carril agresivo (izquierda)	83

de entrada. El sistema que propuso se enfocaba en obtener un clasificador que pueda ser utilizado por un sistema embebido de potencia baja. Por lo cuál utilizo un tipo de red neuronal llamado *Extreme Learning Machines*. Este tipo de red neuronal, como se observa en la Fig. 2.6, se caracteriza por tener tan solo una capa oculta entre las entradas y las salidas. Además se eligen los pesos entre las entradas y la única capa oculta de manera aleatoria y nunca se actualizan. Esta red neuronal solo aprende los pesos entre la capa oculta y las salidas en una sola iteración. Por esta razón este tipo de red resulta siendo muy ligero y de entrenamiento muy veloz.

Echanobe et al. recolectó un conjunto de 21 variables de 11 conductores, pero no utilizó toda la información. En lugar de eso utilizó un algoritmo genético llamado *Non-Dominated Sorting Genetic Algorithm*. Este tipo de algoritmo genético realiza una optimización multiobjetivo. Lo cual significa que trata de optimizar más de una variable a la vez. En este caso se trata de minimizar 3 variables: el número de atributos de entrada para la red neuronal, el número de neuronas en la capa oculta y la tasa de error resultante de la red neuronal entrenada.

Normalmente no existe una solución que permita minimizar las 3 variables a la vez. Debido a esto, lo que se busca en realidad se conoce como *Soluciones Óptimas de Pareto*. Este término hace referencia a las soluciones en las que no se puede optimizar ninguna variable sin degradar a otra. Este tipo de algoritmos genéticos contiene el concepto de Soluciones óptimas de Pareto como función de fitness, de tal manera que permite optimizar las 3 variables antes mencionadas.

Finalmente, se escoge una solución que tiene un 90.1% de tasa de reconocimiento, 8 entradas para la red neuronal y 16 neuronas en la capa oculta.

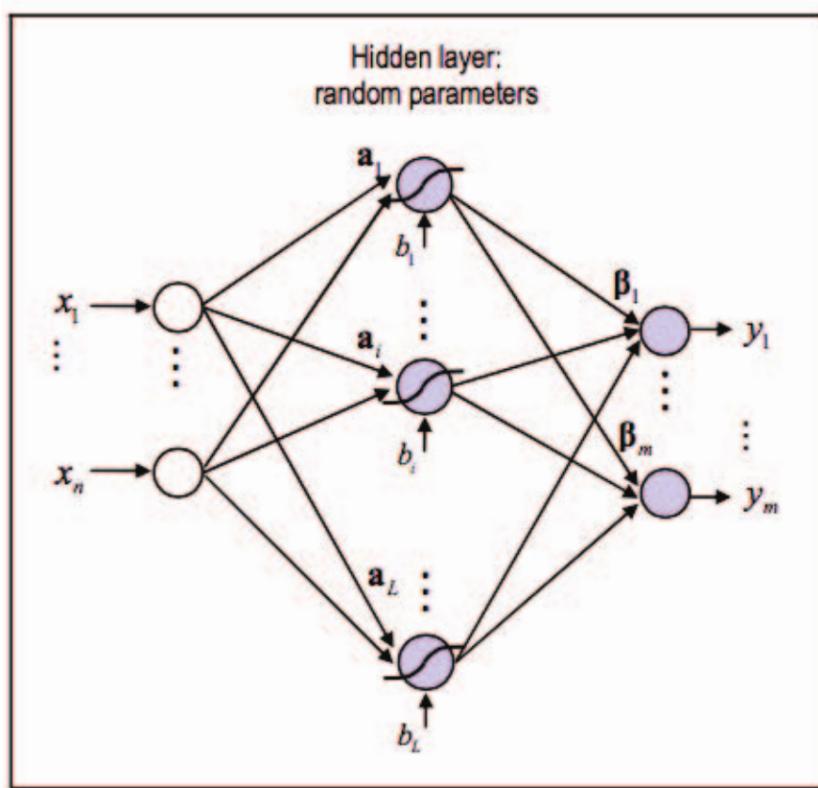


Fig. 2.6 Topología de una Red Neuronal usada para Extreme Learning Machines [17].

C. Algoritmos que combinan los enfoques supervisados y no supervisados

Los algoritmos supervisados y no supervisados aparecen como dos categorías totalmente separadas. Sin embargo, esto no significa que no se puedan usar juntos y combinar para mejorar sus ventajas o reducir sus desventajas.

Ly et al. [18] propuso un sistema que utiliza los sensores iniciales embebidos en un auto para identificar posibles maniobras inseguras y proporcionar un adecuado feedback hacia el conductor. Además también propone caracterizar y diferenciar conductores solo usando datos de los sensores iniciales, para poder diferenciar cuando dos personas distintas utilizan un mismo vehículo. Para lograr su objetivo primero analiza la información de los sensores para determinar un conjunto básico de maniobras: frenado, aceleración y giro.

Se utiliza el algoritmo de aprendizaje no supervisado *K-means Clustering* para identificar las maniobras. Este algoritmo consiste en generar k semillas, de preferencia no aleatorias ya que la inicialización tiene una gran influencia en los clusters resultantes, que representarán la media o el centroide de cada cluster, luego por medio de la definición de una métrica de distancia (Euclídea, citiblock, etc.) se asignan los datos al cluster del centroide más cercano. Luego de clasificar a todos los datos se vuelve a calcular el centroide de cada cluster (que muy probablemente cambiará de posición) y se repite los pasos hasta converger en una posición de centroides.

Al identificar las maniobras se usan dos fuentes de datos distintas: la primera cuenta con la información completa de los sensores, así como también un análisis estadístico de estos (mínimo, máximo, media, varianza, etc.); en cambio, la segunda solo cuenta con el análisis estadístico. Y los resultados que se obtienen son muy similares. El desempeño de el clasificador no se reduce al no incluir los datos completos de los sensores, sino que puede usar tan solo sus datos estadísticos.

Luego se utiliza el algoritmo supervisado *Support Vector Machines* (SVM). Este algoritmo usa los datos con sus respectivas clases y trata de crear un hiperplano que separe todos los datos pertenecientes a una clase de la otra. Mientras este hiperplano tenga una mayor distancia con los miembros más cercanos de ambas clases, se tendrá un menor error de generalización. El encontrar este hiperplano que separe a las dos clases en la mayoría de los casos no es posible en el espacio definido por los atributos actuales de los datos. Debido a esto se mapean los datos en un espacio con más dimensiones para lograr encontrar este hiperplano de una manera más sencilla.

2.3. Estado del arte según sensores usados

Para usar los algoritmos mencionados anteriormente se han usado diferentes clases de sensores para medir distintas variables, ya sea directa o indirectamente. En el Tabla 2.3 se puede apreciar los sensores utilizados en las investigaciones mencionadas.

Tabla 2.3 Resumen de sensores usados en las investigaciones mencionadas

Sensores usados	Referencias
Inertial Measurement Unit (IMU)	[12], [17], [18]
Acelerómetros de bajo costo	[11]
Smartphone	[15]
GPS	[14]
GPS e IMU	[10]

2.3.1. Acelerómetros de bajo costo

Los acelerómetros nos permiten medir la aceleración longitudinal y lateral, que son las que nos permiten medir eventos de conducción como el frenado y la aceleración o los giros. Con un acelerómetro de 2 ejes como el usado en 6294318 es suficiente para medir estas dos aceleraciones.

Estos acelerómetros pueden costar desde \$2, como el MMA6910KQ [19] que consta de 3 ejes y puede tener un rango de ± 3.5 o ± 5.0 g.

2.3.2. Smartphone

Los smartphones tienen cada vez más y más sensores que pueden ser usados para aplicaciones como las de la presente tesis. En [15] se usó los sensores integrados en un Iphone 4 para realizar la tarea de reconocimiento de estilo de conducción. Estos sensores fueron la cámara trasera (solo para registro), el acelerómetro de 3 ejes, el giroscopio de 3 ejes y el GPS.

2.3.3. Inertial Measurement Unit

Esta clase de sensores se puede encontrar bajo un rango de precios muy amplio. Existen unidades por debajo de los \$10 como el MPU-6050 (Fig. 2.7) que cuenta con un giroscopio de 3 ejes y un acelerómetro de 3 ejes; y funciona a un voltaje de 2.3 V a 3.4 V [20]

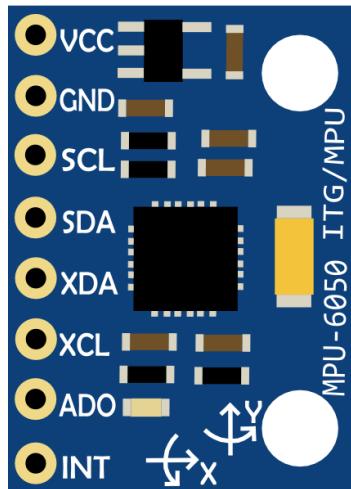


Fig. 2.7 MPU-6050 de Invensense [20].

2.3.4. GPS

El GPS es muy usado también al reconocer estilos de conducción ya que aporta información muy relevante al sistema, por ejemplo se puede usar la información del GPS para detectar el tipo de camino en el que se encuentra el auto en ese momento. Además se puede usar también para obtener la velocidad del auto, aunque es más usado en conjunto con un IMU para que al combinar la información de ambos sensores se usen algoritmos de *Sensor Fusion* para mejorar el resultado de las mediciones.

2.4. Feedback

Una de los elementos más importantes de los sistemas de reconocimiento de estilos de conducción antes mencionados son las señales de feedback que se le dan al conductor. Dependiendo de cuál sea el enfoque del reconocimiento de estilo de conducción (enfoque en ahorro de combustible o enfoque en agresividad de maniobras).

Hay 3 maneras principales de generar este feedback. De manera visual, háptica, audible o una combinación de estos. Cada una presenta ventajas y desventajas.

Para generar feedback de manera visual se pueden incluir Human Machine Interfaces (HMI), que básicamente son pantallas que permiten al conductor saber que estilo de conducción se encuentran ejerciendo en ese momento. Esto se podría lograr usando un smartphone o una tablet también. La desventaja de usar feedback visual es el conductor ya procesa mucha información visual al llevar a cabo la tarea de conducir, por lo que agregar más información requiere que el conductor preste atención al dispositivo y reduzca su atención en el camino. Esto podría presentar problemas de seguridad como se señala en [21]

En [22] se usó una tablet como método de feedback para entregar *Tips de Conducción* al usuario. El sistema funcionaba de la siguiente manera: Reconocía el estilo de conducción actual del conductor y se usaba *Learning Path Planning* para poder llegar desde el estilo de conducción detectado a uno óptimo en el que se consuma menos combustible. Obtenido esto, se procedían a mostrar los tips de conducción al conductor usando la tablet. En este sistema se lograba incrementar la economía del combustible en un 6.25%.

Para generar señales auditivas se puede utilizar un pequeño parlante que haga sonar una alarma cuando se registra un comportamiento no deseado del conductor (Estilo de conducción con alto consumo de combustible o muy agresivo). Este método

Por último, las señales hápticas han sido investigadas debido a que es el único canal de información por el cual los conductores no reciben mucha información de la tarea de conducir. Lo que quiere decir, que generaría menos distracciones que los otros tipos de feedback. En [23] se expone diferentes maneras de entregar esta señal (En el timón, el panel frontal del auto, el asiento , etc.).

3. Diseño Conceptual

3.1. Requerimientos del sistema

Uno de los requerimientos principales del sistema es el uso de una red de sensores modular y sencilla de instalar en distintos tipos de vehículos. Esto se explica fácilmente debido a que este sistema esta orientado a ser usado en el sistema de transporte público en Lima, el cual tiene una gran variedad y un gran número de vehículos. El sistema, entonces, debe ser adaptable a cualquiera de estos vehículos y no contar con un tiempo de instalación elevado.

También se tiene como requerimiento un grado de precisión alto en el reconocimiento de estilos y eventos de conducción. Esto es necesario debido a que este sistema se usará para monitorizar y mejorar la conducta de conducción de los choferes del servicio de transporte público. Los datos arrojados por este sistema necesitan ser confiables.

Por último, se necesita también de una adecuada infraestructura que permita que el sistema funcione en linea, entregando feedback al conductor, aún si se presentan condiciones como la falta temporal de conexión a Internet. Este requerimiento será importante al considerar donde realizar el procesamiento de los datos (dentro de un sistema embebido en cada vehículo o usando un servidor).

Tabla 3.1 Lista de Requerimientos página 1.

<i>Lista de requerimientos del Sistema</i>		Página 1 de 2 Edición: Rev. 1 (11/09/18)
<i>Proyecto:</i>	Diseño de un sistema de reconocimiento de patrones de manejo	
<i>Cliente:</i>	Pontificia Universidad Católica del Perú	
<i>Fecha de modificación</i>	<i>Descripción</i>	
11/09/18	FUNCIÓN PRINCIPAL Reconocer el estilo de manejo de un conductor del sistema de transporte público de Lima y otorgar una señal de retroalimentación de acuerdo a su estilo de conducción actual.	
11/09/18	GEOMETRÍA <ul style="list-style-type: none"> • Alto < 30 cm. • Ancho < 30 cm. • Largo < 30 cm. 	
11/09/18	CINEMÁTICA No se tendrá movimiento.	
11/09/18	PESO Menor a 1 kg.	
11/09/18	MATERIALES Case: Plástico.	
11/09/18	ENERGÍA <ul style="list-style-type: none"> • Conexión de 12 VDC desde toma del auto. • 5 VDC para el microcontrolador. 	
11/09/18	SEÑALES Entrada: <ul style="list-style-type: none"> • Accionamiento de encendido y apagado. • Movimiento del vehículo. Salida: <ul style="list-style-type: none"> • Estilo de conducción. • Calor. 	
11/09/18	COMUNICACIONES <ul style="list-style-type: none"> • Conexión a Internet 3G, WiFi, etc. • Protocolo MQTT o CoAP con un servidor central. • Comunicación continua de datos de sensores. 	

Tabla 3.2 Lista de Requerimientos página 2.

<i>Lista de requerimientos del Sistema</i>		Página 2 de 2 Edición: Rev. 1 (11/09/18)
Proyecto:	Diseño de un sistema de reconocimiento de patrones de manejo	
Cliente:	Pontificia Universidad Católica del Perú	
Fecha de modificación	Descripción	
11/09/18	SENSORES <ul style="list-style-type: none"> • Sensores de movimiento. • Sensor de posición. 	
11/09/18	CONDICIONES DE OPERACIÓN Temperatura ambiente.	
11/09/18	SEGURIDAD <ul style="list-style-type: none"> • Se seguirá la norma ISO 16121 que regula el ambiente de la cabina de manejo. • El sistema no obstaculizará la visibilidad del conductor. • El sistema estará fijo y estará protegido de impactos que el vehículo podría tener. 	
11/09/18	ERGONOMÍA El sistema estará fuera de la vista del conductor.	
11/09/18	MANTENIMIENTO Mantenimiento preventivo cada 6 meses.	
11/09/18	SOFTWARE <ul style="list-style-type: none"> • Algoritmos de Machine Learning. • Programación usando C. 	
11/09/18	ENSAMBLE Instalación modular.	
11/09/18	COSTO Costo de fabricación: \$1000.	
11/09/18	PLAZO DE ENTREGA Entrega planeada para el 28/11/18.	

3.2. Modelo Black Box

En la Fig 3.1 se puede observar el modelo de Black Box o Caja Negra del sistema. En este modelo se pueden apreciar con mayor claridad las entradas y salidas del sistema

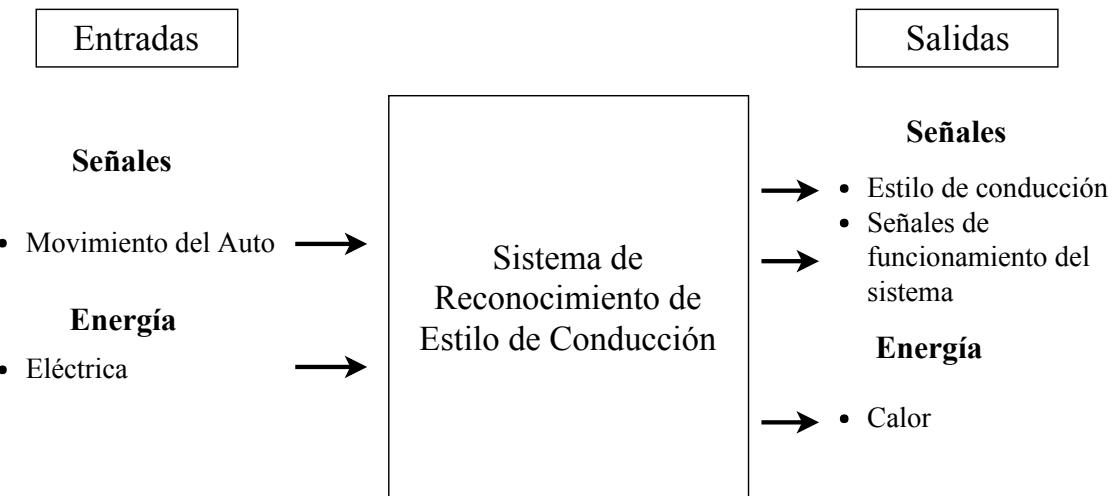


Fig. 3.1 Modelo de Black Box del sistema.

El sistema estará conectado al auto, de donde obtendrá la energía necesaria para funcionar. Debido a que el enfoque del sistema es ahorrar energía, su consumo debe ser reducido.

Además de recibir la energía, este sistema procesará el movimiento del auto usando sensores obteniendo así toda la información necesaria para caracterizar el estilo de conducción del usuario. El estilo de conducción será obtenido a través del procesamiento de la información de los sensores y será representado como una señal de retroalimentación al usuario.

3.3. Estructura de Funciones

La estructura de funciones del sistema se puede apreciar en la Fig 3.2. Para el elaboramiento de esta se han considerado 7 Dominios: Mecánica, Energía, Sensores, Procesamiento, Comunicación, Reconocimiento del estilo de conducción e Interfaz.

Se describirá a continuación cada uno de estos dominios para así obtener una visión general del funcionamiento total del sistema.

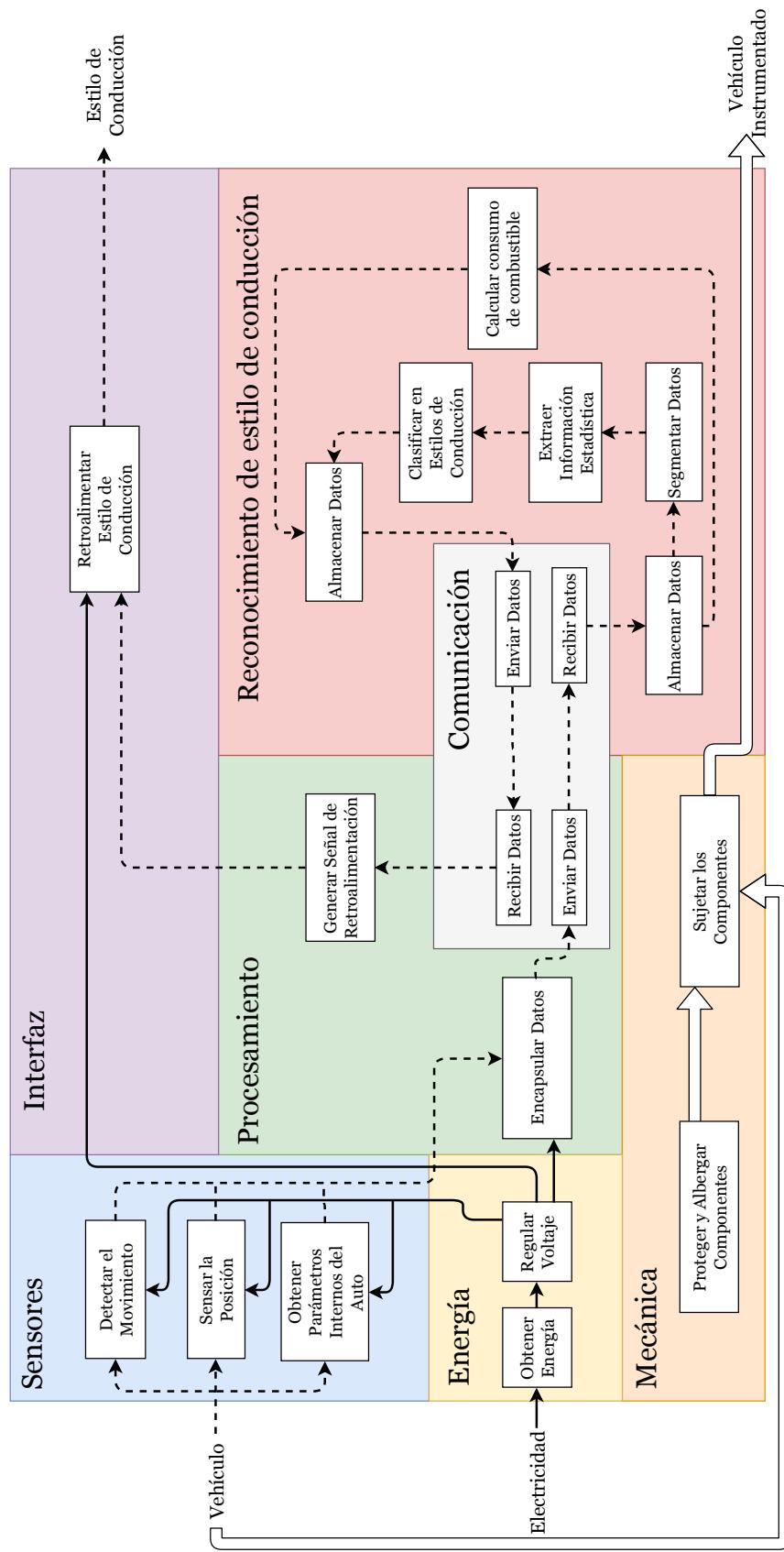


Fig. 3.2 Estructura de Funciones.

3.3.1. Dominio Mecánico

En este dominio (Fig 3.3) se encuentra la función de proteger y albergar todos los componentes del sistema. Esta función hace referencia al case que encapsulará todos los elementos. Además, se debe tener en cuenta que la velocidad relativa entre el vehículo y el sistema debe ser nula para que las medidas de los sensores sean precisas. Es por eso que se tiene como otra función el sujetar los componentes.

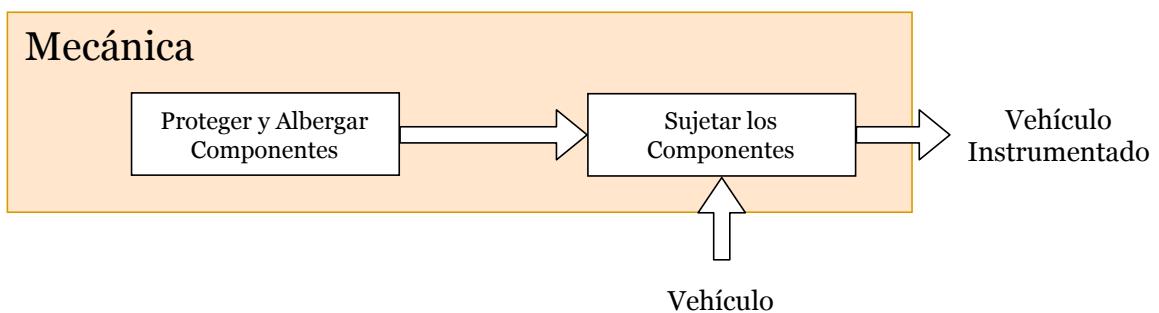


Fig. 3.3 Dominio Mecánico de la estructura de funciones.

3.3.2. Dominio de Energía

Este dominio (Fig 3.4) cumplirá la función de obtener la energía eléctrica necesaria para el funcionamiento del sistema y regular el voltaje, entregándole a cada Dominio esta energía según sus requerimientos específicos. Los módulos que recibirán la energía serán los de Procesamiento, Sensores e Interfaz.

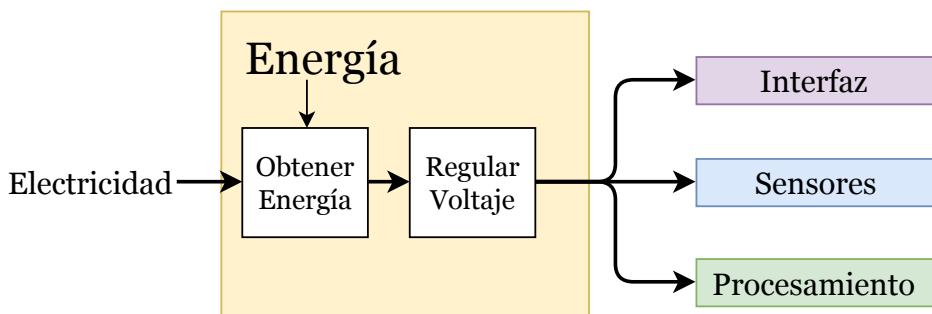


Fig. 3.4 Dominio Eléctrico de la estructura de funciones.

3.3.3. Dominio de Sensores

Este dominio (Fig 3.5) realizará la función de recibir la información del movimiento del vehículo, de su posición y de obtener los parámetros internos del auto. Luego transmitirá esta información al dominio de Procesamiento que manejará todos los datos.

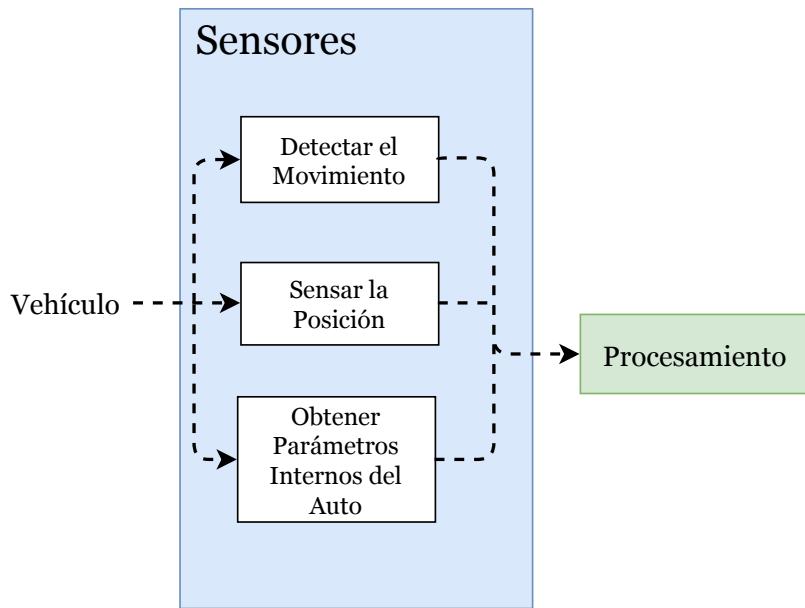


Fig. 3.5 Dominio de Sensores de la estructura de funciones.

3.3.4. Dominio de Comunicación

En este módulo (Fig 3.6) se recopilaron los datos que se manipularon en el Dominio de Procesamiento y se entregarán al dominio de Reconocimiento de Estilo de Conducción.

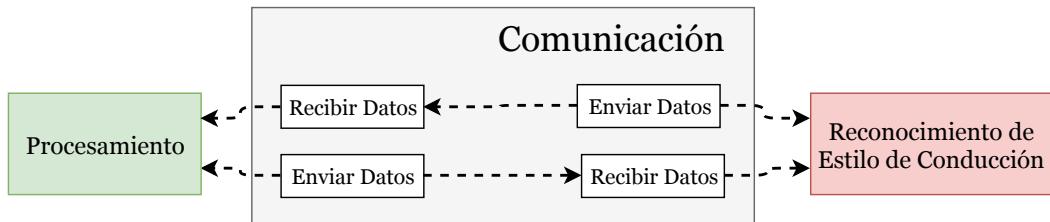


Fig. 3.6 Dominio de Comunicación de la estructura de funciones.

3.3.5. Dominio de Procesamiento

Este dominio (Fig 3.7) realizará la función de recibir la información del dominio de sensores y encapsular los datos para que el dominio de comunicación pueda enviarlos. Luego recibe los datos a través del dominio de comunicación y se encarga de generar la señal de retroalimentación y enviársela al dominio de interfaz.

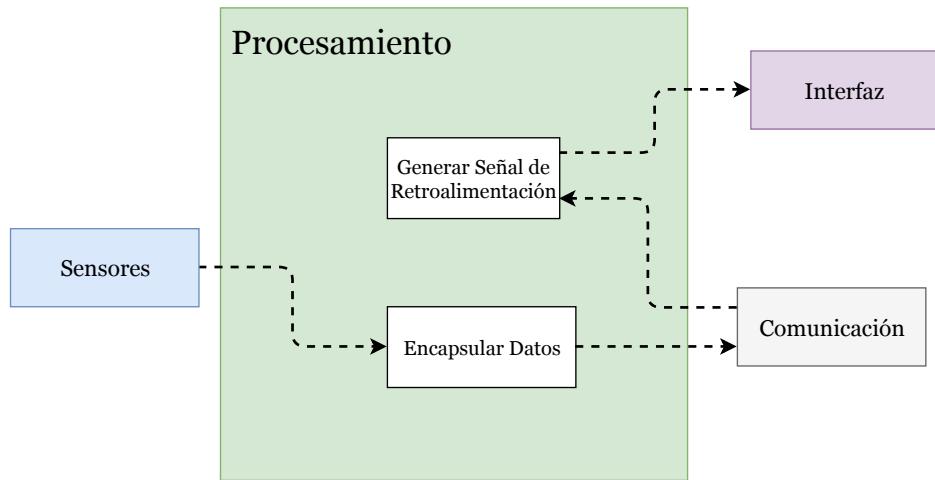


Fig. 3.7 Dominio de Procesamiento de la estructura de funciones.

3.3.6. Dominio de Interfaz

Este dominio (Fig 3.8) realizará la función de recibir la señal de retroalimentación del dominio de Procesamiento y usarla para mostrarle al usuario la retroalimentación (de forma visual, auditiva o háptica).

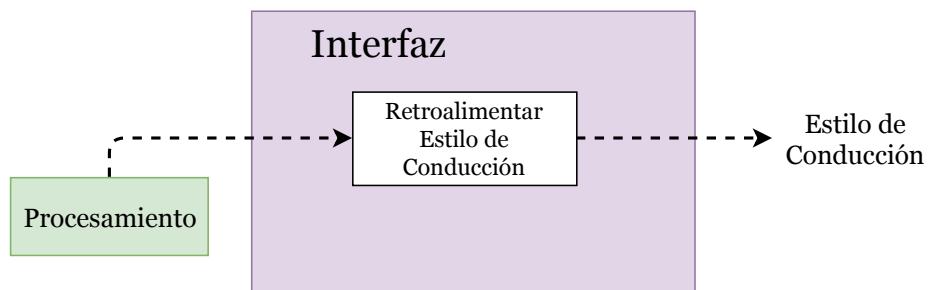


Fig. 3.8 Dominio de Interfaz de la estructura de funciones.

3.3.7. Dominio de Reconocimiento de Estilo de Conducción

En este Dominio se realizará el reconocimiento del estilo de conducción usando . Para alcanzar esto se los datos atraviesan 3 etapas. La etapa de segmentación de datos, en la que se dividen en

realizará un preprocesamiento de la data para luego procesarla usando algoritmos de machine learning.

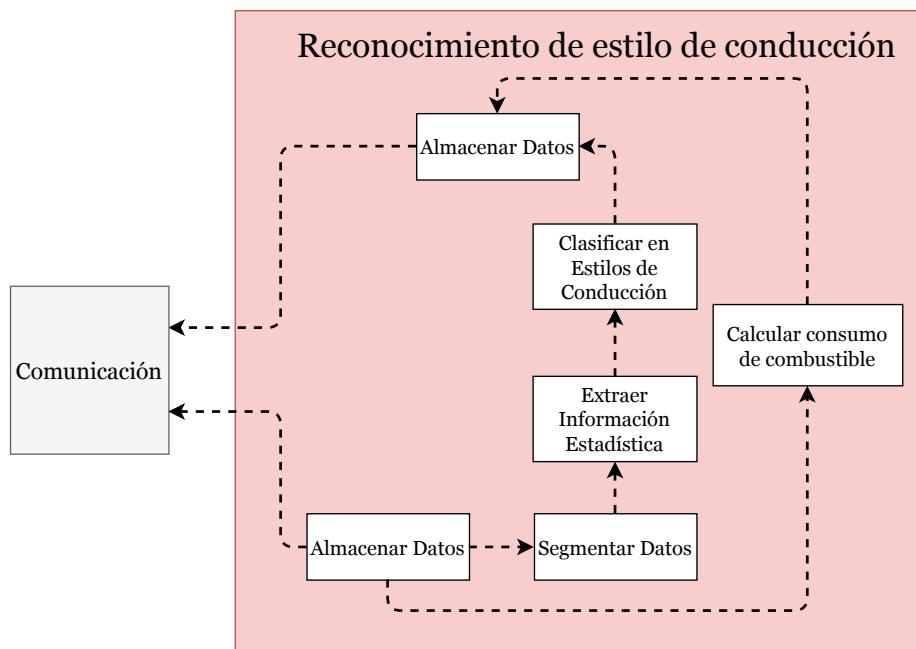


Fig. 3.9 Dominio de Reconocimiento de Estilo de Conducción de la estructura de funciones.

3.4. Matriz morfológica por dominio

En esta sección se propondrán la Matriz morfológica por cada dominio de la estructura de funciones mostrada anteriormente. Cada función tendrá distintas alternativas portadores de solución que al combinarse formarán una solución completa.

3.4.1. Dominio Mecánico

Se presenta a continuación la matriz morfológica del dominio mecánico en la Tabla 3.3 y la descripción de los conceptos de solución propuestos en la Tabla 3.4.

Tabla 3.3 Matriz Morfológica del Dominio Mecánico

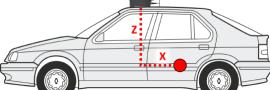
Funciones Parciales	<i>Portadores de Solución</i>		
	1	2	3
Proteger y Albergar Componentes	 <p>Case en el panel frontal</p>	 <p>Case en el techo del vehículo</p>	
Sujetar los Componentes	 <p>Usando un sujetador para el case del dispositivo</p>	 <p>Atornillado a un sujetador que se posee una base magnética</p>	

Tabla 3.4 Conceptos de solución mecánicos propuestos

<i>Conceptos de Solución Propuestos</i>	
Opción 1 	Se posiciona el dispositivo en un case de plástico situado en el panel frontal del auto. Este case se sujetará por medio de un sujetador mecánico, para asegurar que no exista movimiento relativo entre el case y el vehículo. En el panel frontal se tiene una suficiente cobertura para las antenas de GPS y GPRS.
Opción 2 	Se posiciona el dispositivo en un case de plástico situado en el techo del vehículo. Este case se sujetará siendo atornillado a sujetador que posee una base magnética que asegura su inmovilidad. Además esta solución es genial ya que permite que las antenas del GPS y GPRS estén en una óptima posición.
Opción 3 	Se posiciona el dispositivo en el panel frontal del auto. Como en este caso el dispositivo es un smartphone, este será sujetado por un sujetador de smartphone como el de la opción 1.

3.4.2. Dominio Energético

Se presenta a continuación la matriz morfológica del dominio eléctrico en la Tabla 3.5 y la descripción de los conceptos de solución propuestos en la Tabla 3.6.

Tabla 3.5 Matriz Morfológica del Dominio Energético

Funciones Parciales	Portadores de Solución		
	1	2	3
Obtener Energía	 Puerto de 12V en los autos	 Batería de Litio	
Regular Voltaje	 Regulador Lineal	 Regulador Switching	 Cargador de Smartphone

Tabla 3.6 Conceptos de solución eléctricos propuestos

Conceptos de Solución Propuestos	
Opción 1 →	Se usará el puerto de 12 V que se encuentra presente en la gran mayoría de los autos. A continuación se usará un regulador switching que se encargará de regular el voltaje a los requerimientos de los componentes de los dominios de comunicación e interfaz
Opción 2 →	Se usará una batería de ion de litio que energizará a los componentes de los demás dominios. Se usará un regulador lineal para regular el voltaje que se entregará a los componentes de los demás dominios.
Opción 3 →	Se usará el puerto de 12 V que se encuentra presente en la gran mayoría de los autos. Y se conectará a este un cargador de smartphone. De esta manera se asegurará de que el smartphone siempre tenga energía.

3.4.3. Dominios de Procesamiento y de Comunicación

Se presenta a continuación la matriz morfológica del dominio de comunicación en la Tabla 3.7 y la descripción de los conceptos de solución propuestos en la Tabla 3.8.

Tabla 3.7 Matriz Morfológica de los Dominios de Procesamiento y Comunicación

<i>Funciones Parciales</i>		<i>Portadores de Solución</i>		
		1	2	3
Encapsular Datos	Hardware		Raspberry Pi Zero W	 Smartphone
Generar señal de retroalimentación		ESP32 Development Board		
Encapsular Datos	Software		CoAP	
Generar señal de retroalimentación		MQTT	CoAP	
Recibir/Enviar Datos	Hardware		Procesamiento interno (Los datos no se envían)	 Smartphone conexión a internet

3.4.4. Dominio de Sensores

Se presenta a continuación la matriz morfológica del dominio de sensores en la Tabla 3.9 y la descripción de los conceptos de solución propuestos en la Tabla 3.10.

Tabla 3.8 Conceptos de solución propuestos del dominio de comunicación

Conceptos de Solución Propuestos	
Opción 1 →	Se usará el microcontrolador ESP32 para recopilar y encapsular los datos que serán enviados a otros dominios (Interfaz y Clasificación de estilo de conducción). Este microcontrolador usará un módulo GPRS para conectarse a una red celular y poder comunicarse con la nube. Para encapsular los datos se usa el protocolo MQTT, que es adecuado para dispositivos de rendimiento limitado.
Opción 2 →	En esta solución se usará una Raspberry Pi Zero W que es una computadora de una sola tarjeta que, en este caso se encargará también de la ejecución de el módulo de clasificación de estilo de conducción. Es decir, el procesamiento se llevará a cabo localmente y no remotamente.
Opción 3 →	Se empleará un smartphone, el cual tendrá una aplicación que se encargue de recolectar la data de los sensores y luego conectarse a internet para enviar esta data al servidor. Esto se realizará haciendo uso del protocolo CoAP.

Tabla 3.9 Matriz Morfológica del Dominio de Sensores

Funciones Parciales	Portadores de Solución		
	1	2	3
Sensar el Movimiento	 IMU de 6 ejes	 IMU de 9 ejes	
Sensar la Posición	 GPS		Smartphone
Sensar Parámetros Internos del Auto		 OBD2 + WiFi	

Tabla 3.10 Conceptos de solución propuestos del dominio de sensores

<i>Conceptos de Solución Propuestos</i>	
Opción 1 →	Para esta solución se usarán un acelerómetro, giroscopio y magnetómetro de 3 ejes cada uno, reunidos en un IMU. Además, se usará también un módulo GPS y un conector OBD2 que se pueda comunicar por medio de bluetooth con el microcontrolador.
Opción 2 →	En esta solución se usará un acelerómetro, giroscopio y magnetómetro, de dos ejes cada uno, reunidos en un IMU. Además se usará también un módulo GPS y un conector OBD2 que pueda comunicarse a través de WiFi con la computadora mencionada en el módulo de comunicación.
Opción 3 →	Para esta solución se usarán los sensores presentes en un smartphone (IMU y GPS) y además este smartphone se comunicará inalámbricamente con el conector OBD2 por medio de WiFi.

3.4.5. Dominio de Interfaz

Se presenta a continuación la matriz morfológica del dominio de interfaz en la Tabla 3.11 y la descripción de los conceptos de solución propuestos en la Tabla 3.12.

Tabla 3.11 Matriz Morfológica del Dominio de Interfaz

Funciones Parciales	<i>Portadores de Solución</i>		
	1	2	3
Indicar Funcionamiento			
Retroalimentar estilo de conducción	Tablet/Celular	Pantalla	

Tabla 3.12 Conceptos de solución propuestos del dominio de interfaz

<i>Conceptos de Solución Propuestos</i>	
Opción 1 	En esta solución se tendrá usará como pantalla que se conectara por medio de cables al ESP32 para poder mostrar la información relevante al conductor. Esta información incluye el estilo de conducción actual del conductor y parámetros de su consumo de combustible.
Opción 2 	En esta solución se mostrará la información en una tablet o celular. Usando para esto una aplicación android que se conectará por medio de bluetooth o WiFi con la raspberry pi. Esta pantalla mostrará al conductor información relevante con la tarea de conducción. Esta información contendrá el estilo de conducción del conductor actual, y parámetros de su consumo de combustible.
Opción 3 	En esta solución se mostrará la información en una tablet o celular. Usando para esto una aplicación android. Esta pantalla mostrará al conductor información relevante con la tarea de conducción. Esta información contendrá el estilo de conducción del conductor actual, y parámetros de su consumo de combustible.

3.4.6. Dominio de Reconocimiento de estilo de conducción

Se presenta a continuación la matriz morfológica del dominio de reconocimiento de estilo de conducción en la Tabla 3.13 y la descripción de los conceptos de solución propuestos en la Tabla 3.14.

Tabla 3.13 Matriz Morfológica de Reconocimiento de estilo de conducción

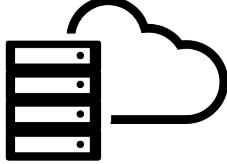
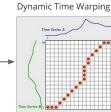
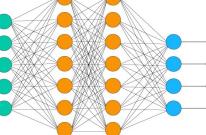
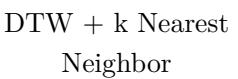
Funciones Parciales		Portadores de Solución		
		1	2	3
Almacenar Datos	Hardware	 Almacenamiento Local (Tarjeta SD)	 Almacenamiento Remoto (Servidor)	
	Software	 CSV	 Base de Datos	
Segmentar Datos		Ground Truth Rules	Changepoint Detection	
Extraer información estadística		Según algoritmo de clasificación a usar		
Clasificar Estilo de conducción		 K Nearest Neighbors	 Dynamic Time Warping	 Neural Networks
		 DTW + k Nearest Neighbor		 Random Forest

Tabla 3.14 Conceptos de solución propuestos del dominio de reconocimiento de estilo de conducción

Conceptos de Solución Propuestos	
Opción 1 	Para esta solución se usará un servidor en la nube para almacenar los datos de los sensores en una base de datos y para ejecutar el algoritmo de procesamiento de datos. En este servidor se procederá a segmentar los datos en maniobras usando <i>Ground Truth Rules</i> , que son reglas que se generarán a través de conocimiento experto. Una vez la data se ha segmentado se extraerá información estadística como la media, la desviación estándar, etc. durante ese segmento. Estos valores estadísticos serán los features que alimentarán a la <i>Red Neuronal</i> que se encargará de clasificar cada segmento en un estilo de conducción agresivo o normal.
Opción 2 	En esta solución se usará una tarjeta SD para almacenar los datos de los sensores en un archivo csv y se ejecutará el procesamiento localmente. En este caso se usará el algoritmo de <i>Changepoint Detection</i> para segmentar los datos. Luego cada segmento se procesará a través de <i>k-Nearest Neighbor</i> para realizar la clasificación de estilo de conducción. Se usará <i>Dynamic Time Warping</i> (DTW) para comparar la similitud entre las señales. Para esto se necesitará un conjunto de <i>templates</i> que representen cada estilos de conducción posible y que se compararán con cada nueva señal para clasificar futuros segmentos.
Opción 3 	En esta solución se usa también un servidor en la nube para almacenar y procesar los datos de los sensores. En este caso el algoritmo a usar es el de <i>Random Forest</i> . Este algoritmo consiste en diferentes reglas que se usan para poder clasificar cada nuevo ejemplo que llega a partir de los datos de los sensores.

3.5. Conceptos integrados de solución

En la Tabla 3.15 se puede observar un resumen de las soluciones escogidas por cada concepto de solución propuesto. Se describirán cada una de las soluciones para luego analizarlas con el fin de obtener la solución óptima.

Tabla 3.15 Conceptos de solución propuestos

	<i>Solución 1</i>	<i>Solución 2</i>	<i>Solución 3</i>
Proteger y Albergar Componentes	Case en el panel frontal	Case en el techo del vehículo	Dispositivo en el panel frontal
Sujetar los Componentes	Usando un sujetador de smartphones	Atornillado a un sujetador que posee una base magnética	Usando un sujetador de smartphones
Obtener Energía	Puerto de 12 V	Batería de Ion Litio	Puerto de 12 V
Regular Voltaje	Regulador Switching	Regulador Lineal	Cargador de smartphone
Sensar el Movimiento	IMU de 9 ejes	IMU de 6 ejes	Smartphone
Sensar la Posición	GPS	GPS	Smartphone
Sensar Parámetros Internos del Auto	OBD2 + Bluetooth	OBD2 + WiFi	OBD2 + WiFi
Encapsular Datos	ESP32 + MQTT	Raspberry Pi Zero W + CoAP	Smartphone + MQTT
Generar Señal de Retroalimentación			
Enviar/Recibir Datos	Módulo GPRS	Procesamiento Internos	Smartphone
Retroalimentar Estilo de Conducción	Pantalla	Tablet/Smartphone	Tablet/Smartphone
Almacenar Datos	Almacenamiento Remoto (Servidor) + Base de Datos	Almacenamiento Local (Tarjeta SD) + CSV	Almacenamiento Remoto (Servidor) + CSV
Segmentar Datos	Ground Truth Rules	Changepoint Detection	Changepoint Detection
Clasificar Estilo de conducción	Neural Networks	DTW + k Nearest Neighbor	Random Forest

3.5.1. Concepto integrado de solución 1

El sistema se puede observar en la Fig. 3.10 se encuentra contenido en un case de plástico, que albergará todos los componentes electrónicos y que se sujetará usando un *phone holder* en el panel frontal del auto. Además, este sistema contará con una pantalla que estará posicionada de tal manera que el conductor pueda mirarla con facilidad. Este elemento será el que retroalimente el estilo de conducción al conductor.

El dispositivo se conectará a la toma de 12 V y a través de un regulador switching se alcanzarán los 3.3 V que necesita el ESP32 y los 5 V que necesitan algunos de los módulos. Estos módulos consisten en: Un acelerómetro de 3 ejes, un giroscopio de 3 ejes, un magnetómetro de 3 ejes (IMU), un módulo GPS y un módulo GPRS.

El conector OBD2 es el único módulo que no será energizado por el regulador switching, debido a que se conectará al puerto OBD2 del auto y obtendrá energía por este mismo medio. Este conector se comunicará por medio de bluetooth al microcontrolador y de esta manera obtendrá los parámetros internos del vehículo.

El microcontrolador entonces gestionará todo los datos recibidos y se encargará usar el protocolo MQTT para enviarlos. Usando el módulo GPRS que se conectaría a internet, haciendo llegar los datos hasta el servidor. Luego los datos se almacenarán en una base de datos y se procesarán allí.

El primer paso para procesar los datos es segmentarlos en períodos en los que se realiza determinada maniobra con un solo estilo de conducción. Esto se puede lograr a través de reglas antes predefinidas usando conocimiento experto. Estas reglas son llamadas *Ground Truth Rules*.

El segundo paso es analizar el segmento estadísticamente. Se definirán una serie de *features* que representen adecuadamente al segmento de datos. Estos features podrían ser: media, desviación estándar, máximo , mínimo, etc. Estos elementos serán las entradas de la *red neuronal* que ya ha sido entrenada previamente con una cantidad significativa de datos.

La red neuronal procesará las entradas y arrojará en la salida la clase a la que pertenece este segmento, pudiendo ser estilo de conducción agresivo o estilo de conducción normal. Una vez que se tiene el estilo de conducción clasificado, se procede a enviarlo de regreso hacia el microcontrolador. Este se encargará de recibir la clasificación y de acondicionar esta información para mostrarla en la pantalla presente en el vehículo. Esta pantalla mostrará el estilo de conducción reconocido y un estimado de el consumo de combustible actual si esta disponible.

El cálculo de consumo de combustible se realizará también en el servidor usando parte de los datos que provienen del módulo OBD2. Sin embargo, esta información solo estará disponible si se tienen los parámetros adecuados desde el puerto OBD2 (Cada auto tiene disponibles diferentes parámetros).

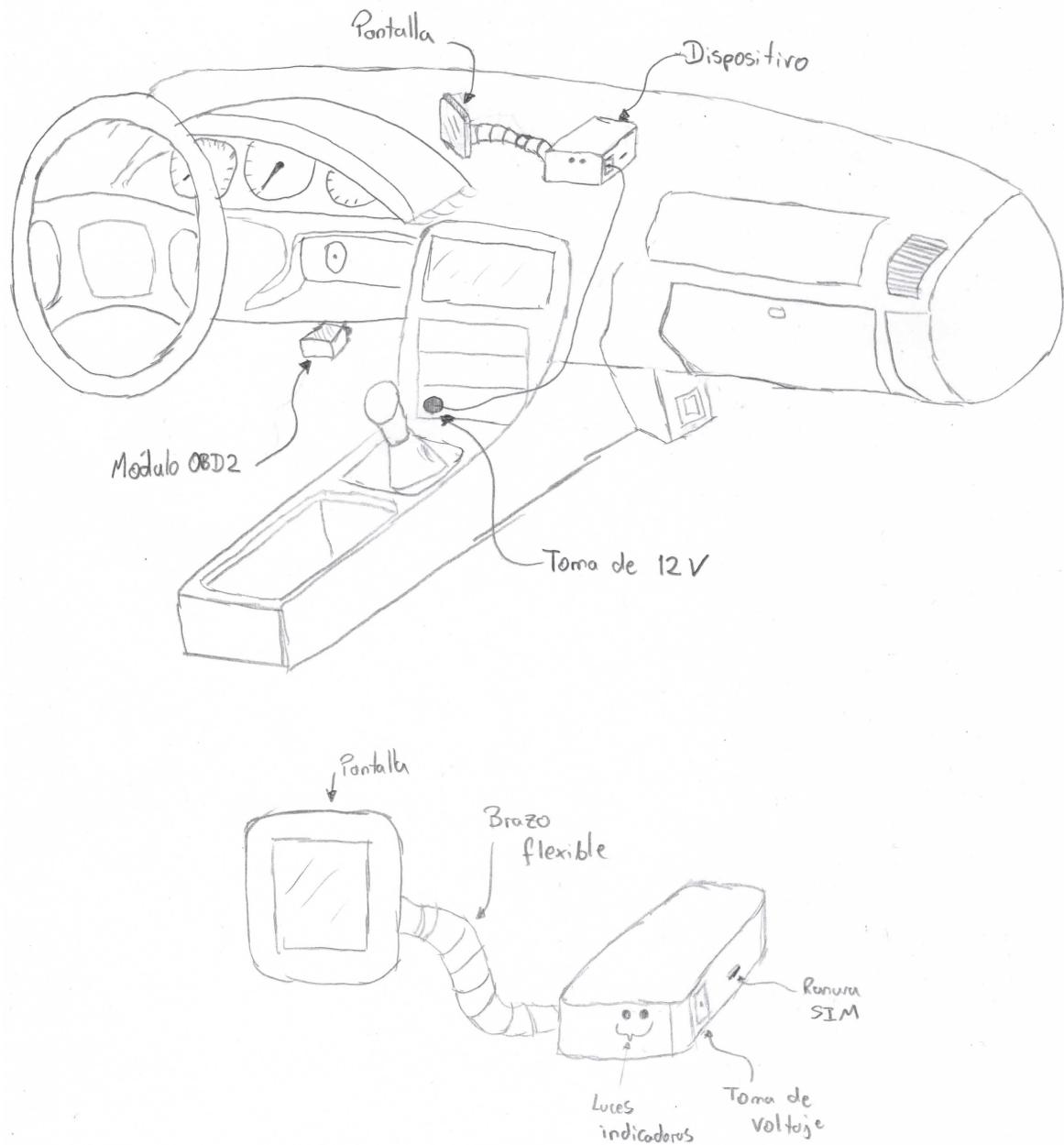


Fig. 3.10 Concepto integrado de solución 1.

3.5.2. Concepto integrado de solución 2

El sistema que se puede observar en la Fig. 3.11 se encuentra contenido en un case de plástico, que albergará todos los componentes electrónicos y que se sujetará usando sujetador magnético en la parte superior del auto (techo). Además, este sistema contará con smartphone o tablet que este sujetada por un phone holder de tal manera que el conductor pueda mirarla con facilidad. Este elemento será el que retroalimente el estilo de conducción al conductor.

El dispositivo contará con una batería de ion litio con la que se encargará de obtener energía para su funcionamiento y a través de un regulador lineal se alcanzarán los 5 V que necesita la raspberry pi zero w y algunos de los módulos. Estos módulos consisten en: Un acelerómetro de 2 ejes, un giroscopio de 2 ejes, un magnetómetro de 2 ejes (IMU de 6 ejes), un módulo GPS y un módulo GPRS.

El conector OBD2 es el único módulo que no será energizado por el regulador lineal, debido a que se conectará al puerto OBD2 del auto y obtendrá energía por este mismo medio. Este conector se comunicará por medio de WiFi a la raspberry y de esta manera obtendrá los parámetros internos del vehículo.

La raspberry entonces gestionará todo los datos recibidos y se encargará de procesarlos localmente. El primer paso para procesar los datos es segmentarlos en periodos en los que se realiza determinada maniobra con un solo estilo de conducción. Esto se puede lograr a través de un algoritmo llamado *changepoint detection* que analiza el comportamiento de las señales y trata de dividirlas en segmentos en donde su comportamiento sea lineal o cercano a lineal.

El segundo paso es analizar el segmento y clasificarlo usando el algoritmo de *k-NN*. Este algoritmo consiste en comparar el segmento obtenido con segmentos ejemplo que se tienen almacenados, y otorgarle la clase de acuerdo a los ejemplos que sean mas similares a este segmento.

El algoritmo arrojará entonces la clase a la que pertenece este segmento, pudiendo ser estilo de conducción agresivo o estilo de conducción normal. Una vez que se tiene el estilo de conducción clasificado, se procede a enviarlo en la tablet o celular presente en el vehículo para retroalimentar la información al conductor. En el smartphone o tablet existirá una aplicación que mostrará el estilo de conducción reconocido y un estimado de el consumo de combustible actual si esta disponible.

El cálculo de consumo de combustible se realizará en la raspberry pi usando parte de los datos que provienen del módulo OBD2. Sin embargo, esta información solo estará disponible

si se tienen los parámetros adecuados desde el puerto OBD2 (Cada auto tiene disponibles diferentes parámetros).

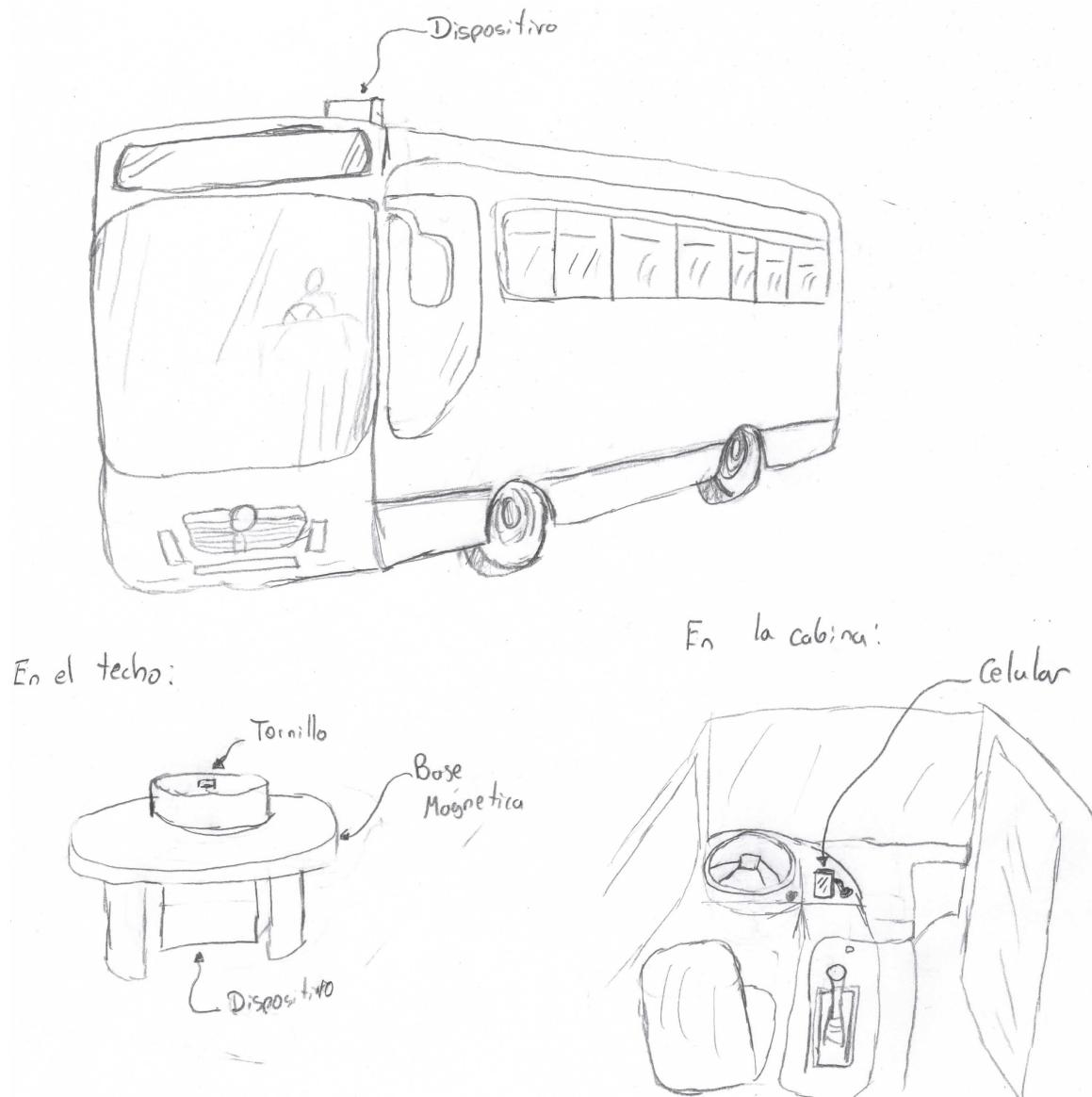


Fig. 3.11 Concepto integrado de solución 2.

3.5.3. Concepto integrado de solución 3

En esta solución que se puede observar en la Fig. 3.12 el sistema usará los sensores internos del smartphone para recopilar los datos necesarios, por lo que se requerirá un sujetador de smartphone para autos en el panel frontal del auto que este posicionado de tal manera que el conductor pueda mirarla con facilidad, ya que el smartphone también mostrará los datos obtenidos luego de la clasificación.

El dispositivo se conectará a la toma de 12 V por medio del cargador de smartphone, para asegurarse de no perder energía y apagarse.

El conector OBD2 es el único módulo que no será energizado por el cargador de smartphone, debido a que se conectará al puerto OBD2 del auto y obtendrá energía por este mismo medio. Este conector se comunicará por medio de WiFi con el smartphone y de esta manera se obtendrán los parámetros internos del vehículo.

El smartphone entonces usará una aplicación android, que será la que gestionará todo los datos recibidos y se encargará usar el protocolo MQTT para enviarlos a un servidor remoto. Para esto se conectará a internet usando 3G, 4G o la línea que se encuentre disponible. Luego los datos se almacenarán en una base de datos y se procesarán en el servidor.

El primer paso para procesar los datos es segmentarlos en periodos en los que se realiza determinada maniobra con un solo estilo de conducción. Esto se puede lograr a través de un algoritmo llamado *changepoint detection* que analiza el comportamiento de las señales y trata de dividirlas en segmentos en donde su comportamiento sea lineal o cercano a lineal.

El segundo paso es usar el Random Forest para clasificar adecuadamente este segmento en un estilo de conducción agresivo o estilo de conducción normal. Una vez que se tiene el estilo de conducción clasificado, se procede a enviarlo de regreso hacia el smartphone. La aplicación, entonces, se encargará de recibir la clasificación y de acondicionar esta información para mostrarla en la tablet o celular presente en el vehículo. Esta aplicación mostrará también un estimado de el consumo de combustible actual si este esta disponible.

El cálculo de consumo de combustible se realizará también en el servidor usando parte de los datos que provienen del módulo OBD2. Sin embargo, esta información solo estará disponible si se tienen los parámetros adecuados desde el puerto OBD2 (Cada auto tiene disponibles diferentes parámetros).

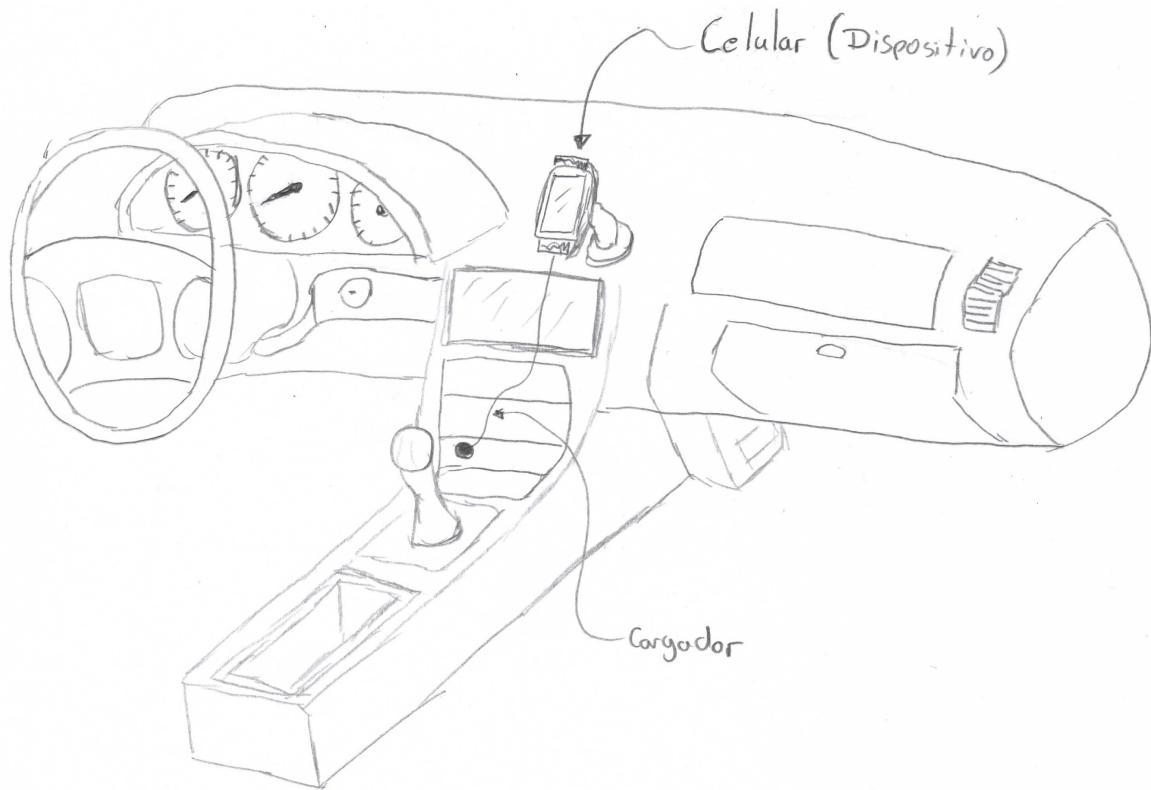


Fig. 3.12 Concepto integrado de solución 3.

3.6. Evaluación de conceptos de solución

Se evaluará en esta sección a las propuestas planteadas usando criterios técnicos y económicos. De esta manera se obtendrá la solución óptima.

3.6.1. Evaluación técnica

En esta evaluación se tienen en cuenta la eficiencia energética del dispositivo, la seguridad en su uso, la rapidez de su respuesta, la rigidez de su ensamblaje frente a aceleraciones fuertes, la confiabilidad de los resultados, la ergonomía, la facilidad en el transporte de la solución y por último su complejidad de instalación.

Tabla 3.16 Evaluación técnica

<i>Variantes de concepto</i>		<i>Solución 1</i>		<i>Solución 2</i>		<i>Solución 3</i>		<i>Solución ideal</i>	
<i>Criterios de evaluación</i>	<i>g</i>	<i>p</i>	<i>gp</i>	<i>p</i>	<i>gp</i>	<i>p</i>	<i>gp</i>	<i>p</i>	<i>gp</i>
<i>Buen uso de energía</i>	2	3	6	1	2	3	6	4	8
<i>Seguridad</i>	2	3	6	2	4	3	6	4	8
<i>Rapidez</i>	3	3	9	1	3	2	6	4	12
<i>Rigidez</i>	2	2	4	3	6	2	4	4	8
<i>Confiabilidad</i>	3	3	9	3	9	3	9	4	12
<i>Ergonomía</i>	3	3	9	1	3	3	9	4	12
<i>Transportabilidad</i>	3	2	6	1	3	3	9	4	12
<i>Complejidad de instalación</i>	1	3	3	1	1	3	3	4	4
<i>Puntaje máximo</i>		22	52	13	31	22	52	32	76
<i>Valor técnico</i>		0.69	0.68	0.41	0.41	0.69	0.68	1	1

3.6.2. Evaluación económica

En esta evaluación se tiene en cuenta el número de piezas de cada solución, la facilidad para adquirir los componentes, el costo de fabricación, el costo de tecnología, la facilidad de mantenimiento y los costos de operación.

Tabla 3.17 Evaluación económica

<i>Variantes de concepto</i>		<i>Solución 1</i>		<i>Solución 2</i>		<i>Solución 3</i>		<i>Solución ideal</i>	
<i>Criterios de evaluación</i>	<i>g</i>	<i>p</i>	<i>gp</i>	<i>p</i>	<i>gp</i>	<i>p</i>	<i>gp</i>	<i>p</i>	<i>gp</i>
<i>Número de piezas</i>	3	3	9	3	9	3	9	4	12
<i>Fácil adquisición de componentes</i>	2	3	6	2	4	3	6	4	8
<i>Costo de fabricación</i>	1	3	3	3	3	3	3	4	4
<i>Costo de tecnología</i>	3	3	9	2	6	2	6	4	12
<i>Facilidad de mantenimiento</i>	3	2	6	2	6	3	9	4	12
<i>Costos de operación</i>	3	2	6	3	9	1	3	4	12
<i>Puntaje máximo</i>		16	39	15	37	15	36	24	60
<i>Valor técnico</i>		0.67	0.65	0.63	0.62	0.63	0.6	1	1

3.6.3. Evaluación de soluciones

Tabla 3.18 Evaluación de soluciones

	<i>Valor técnico</i>	<i>Valor económico</i>	<i>Calificación</i>
Solución 1	0.68	0.65	Buena solución
Solución 2	0.41	0.62	Solución deficiente
Solución 3	0.68	0.6	Buena solución

De acuerdo a lo observado en los análisis técnico y económico se consideran como buenas soluciones a las soluciones 1 y 3. Sin embargo, la solución 1 resulta la más adecuada. Por este motivo se escoge usar la solución 1 para el desarrollo de esta tesis.

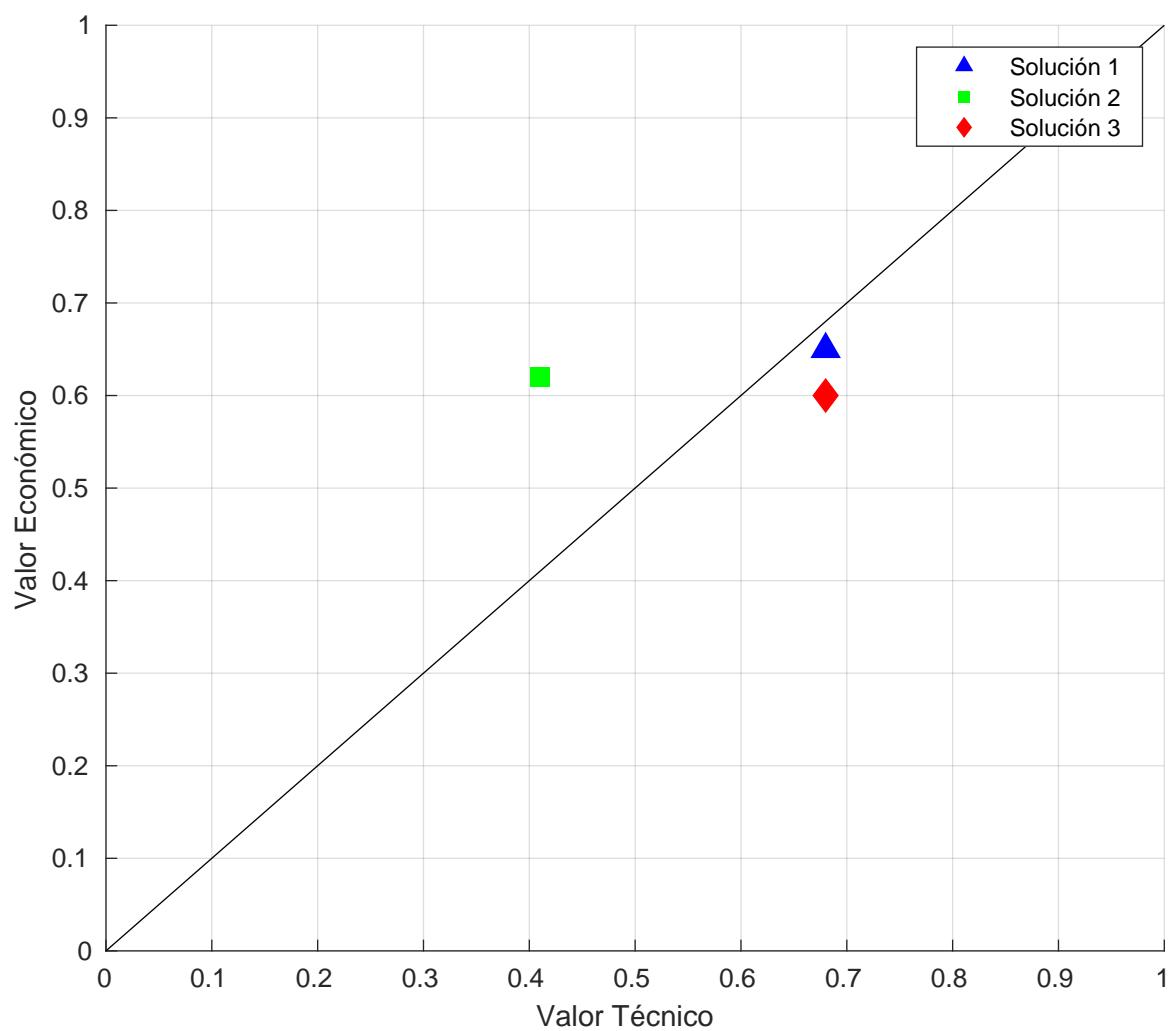


Fig. 3.13 Diagrama de evaluación

4. Diseño del sistema mecatrónico

En este capítulo se desarrollará el diseño del sistema a partir del concepto óptimo de solución desarrollado en el capítulo anterior. Para comenzar, se presenta el sistema integrado en su totalidad y se describe su funcionamiento. Luego, se presentará el desarrollo del sistema electrónico. El cuál comprende la selección de componentes, los diagramas esquemáticos y el desarrollo de la placa electrónica. A continuación, se presentará el diseño mecánico. El cuál consiste de cálculos básicos, selección de componentes y materiales y descripción de planos de ensamble y despiece. Por último, se presentará el diseño del algoritmo de clasificación. El cuál incluye los diagramas de flujo que describen los algoritmos utilizados. Las hojas de datos y planos completos se pueden encontrar en los Anexos.

4.1. Integración del sistema mecatrónico

El sistema, Fig. 4.1 , esta compuesto por una placa electrónica que recogerá los datos de conducción usando sensores y estará contenida por una caja electrónica o case, se le llamará case principal. Estos datos se enviarán a través de la red 2G a un servidor y este responderá con el estilo de conducción, el cual se mostrará en la pantalla.

Este dispositivo estará colocado en el panel frontal del vehículo como se puede apreciar en la Fig. 4.2. Para sujetar el dispositivo al panel se usa cinta adhesiva de doble contacto. El dispositivo cuenta con 4 puertos: Puerto de carga, ranura de tarjeta Nano SIM y 2 puertos SMA para las antenas de GPS y GPRS/GSM.

4.2. Funcionamiento del sistema mecatrónico

Para comenzar, se deben verificar la conexión de alimentación al dispositivo y su conexión a Internet antes de iniciar el viaje del vehículo. Esto se puede comprobar observando las

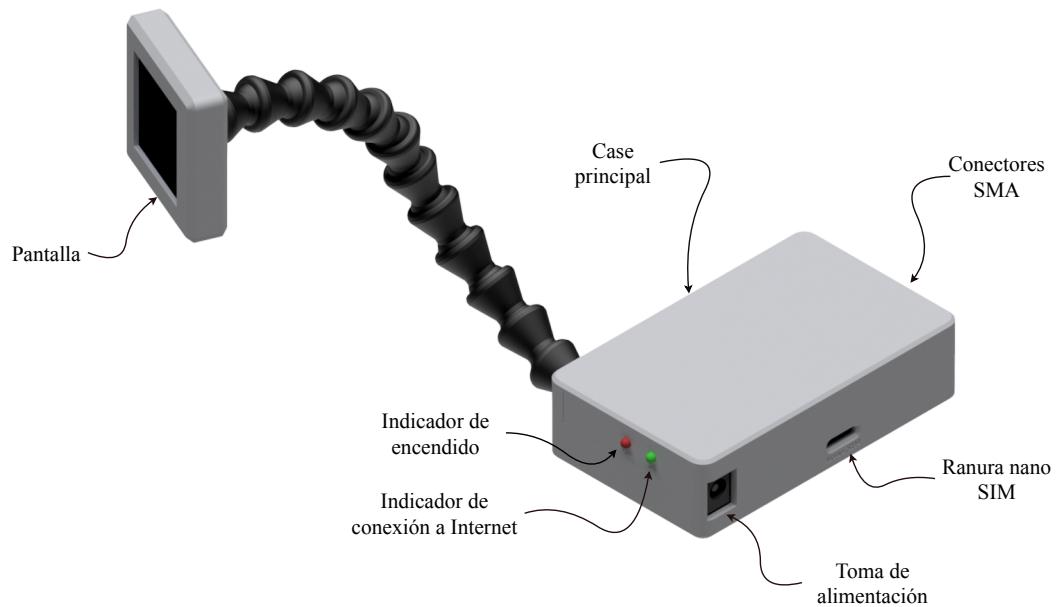


Fig. 4.1 Dispositivo de clasificación de estilo de conducción.

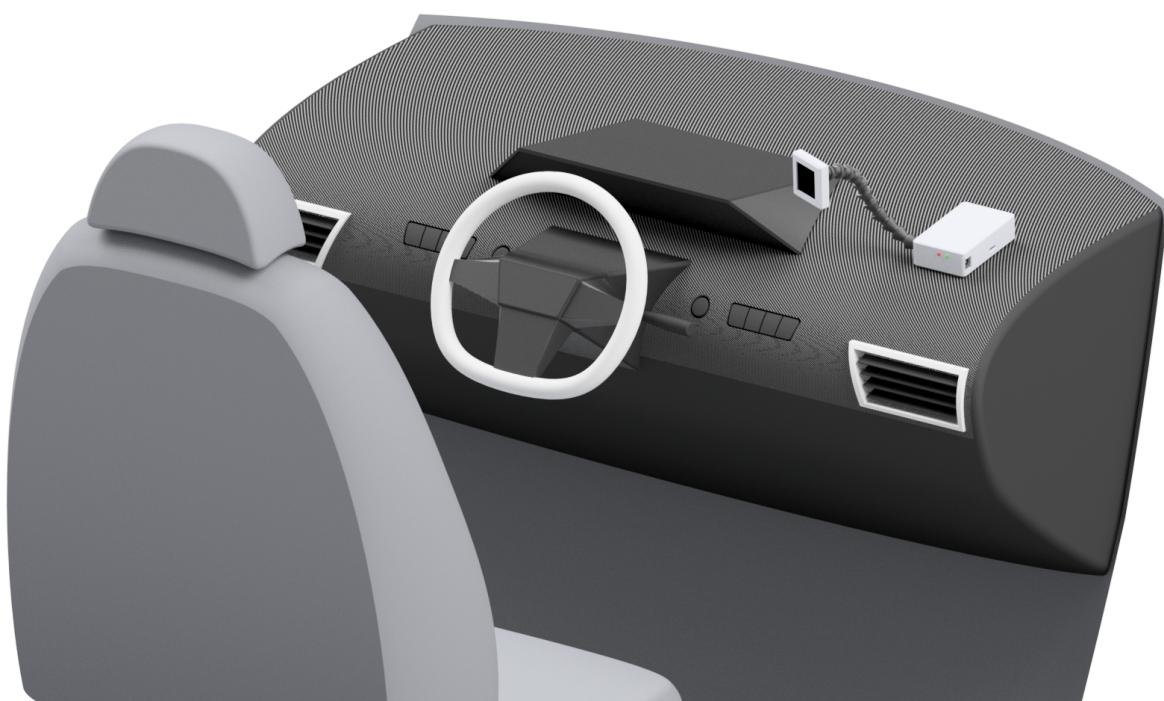


Fig. 4.2 Dispositivo de clasificación de estilo de conducción montado en un auto.

luces indicadora de 'Encendido' y 'Conexión'. El dispositivo cuenta con una ranura para una tarjeta Nano SIM y solo se conectará a Internet si existe una de estas tarjetas insertada en el dispositivo que cuente con un plan de datos activo.

Luego de verificar que las conexiones funcionan, se inicia el viaje del vehículo. El conductor debe ajustar la posición de la pantalla de tal manera de que la pueda visualizar sin ningún problema y que no bloquee la vista de este. El sistema empezará a recolectar datos del IMU, del GPS y de los sensores internos del vehículo a través del módulo OBD2 desde que el sistema se conecta a Internet por primera vez. De esta manera se registra el inicio de un nuevo viaje. Los datos recolectados por los sensores se graban en el microcontrolador, quién obtendrá la ubicación , velocidad y aceleración exacta del vehículo al combinar los datos del IMU y del GPS. Además calculará el consumo de combustible de los datos obtenidos a través del módulo OBD2. Luego enviará estos datos a la nube, en donde se guardarán en una base de datos y se ejecutará el algoritmo de clasificación. Al clasificar los datos se envían de regreso al dispositivo, quién se encargará de mostrar el resultado de la clasificación al conductor por medio de la pantalla.

La pantalla mostrará tan solo un símbolo con un color representativo para evitar distracciones del conductor. De esta manera el conductor podrá conocer si se encuentra manejando de una manera agresiva o no, con tan solo una mirada a la pantalla. Cada maniobra clasificada con un estilo de conducción del conductor es registrada durante todo el viaje en la nube y se le otorga un puntaje de acuerdo a su nivel de agresividad. Mientras menos agresiva sea su puntaje será mayor. Se calculará luego un puntaje para toda la trayectoria realizada al sumar los puntajes de cada maniobra. Al final del día el conductor habrá realizado varias trayectorias y se calculará un promedio que se le asignará como indicador de desempeño.

4.3. Diseño electrónico

En esta sección se presentará primero el diagrama de bloques general del dispositivo. Luego se desarrollará la selección de cada componente y el diseño de las conexiones y de la placa electrónica.

4.3.1. Diagrama de bloques

En la Fig. 4.3 se tiene el diagrama de bloques del dispositivo. El pre-procesamiento y la recolección de los datos de los sensores se llevará a cabo por medio de un ESP-WROOM-32.

A este están conectados un IMU de 6 ejes (ICM-20648), por medio del protocolo I²C; un módulo GPS (NEO-M8N), por medio de comunicación serial y una pantalla de tecnología IPS LCD (Con el controlador ST7789), por medio del protocolo SPI. El ESP-WROOM-32 enviará los datos pre-procesados por medio de un módulo GPRS/GSM (SIM800L), que se encuentra conectado a él por medio de comunicación serial. Por último el módulo OBD2 se encuentra conectado al puerto OBD2 del vehículo y se comunica inalámbricamente con el ESP-WROOM-32 usando comunicación serial a través de Bluetooth.

Todos estos elementos se alimentan usando la energía disponible del vehículo (12 V de la batería). El módulo OBD2 es el único elemento que se encuentra separado del dispositivo principal y obtiene su energía a través del mismo puerto OBD2 por el cual está conectado al vehículo. Los demás componentes no pueden ser alimentados directamente con 12 V, por lo que se necesita transformar el voltaje de acuerdo a los requerimientos de cada elemento. Esto se realiza en dos etapas, ya que se necesitan solo 2 voltajes distintos: 4 V y 3.3 V. La primera etapa de reducción se lleva a cabo por un regulador switching (TPS563210) que se encarga de regular el voltaje de 12 V a 4 V para alimentar al módulo GPRS/GSM (SIM800L). Luego se usa un regulador LDO (LT1764A) que se encarga de regular el voltaje de 4 V a 3.3 V que servirán para alimentar al microcontrolador, módulo GPS, IMU y a la pantalla.

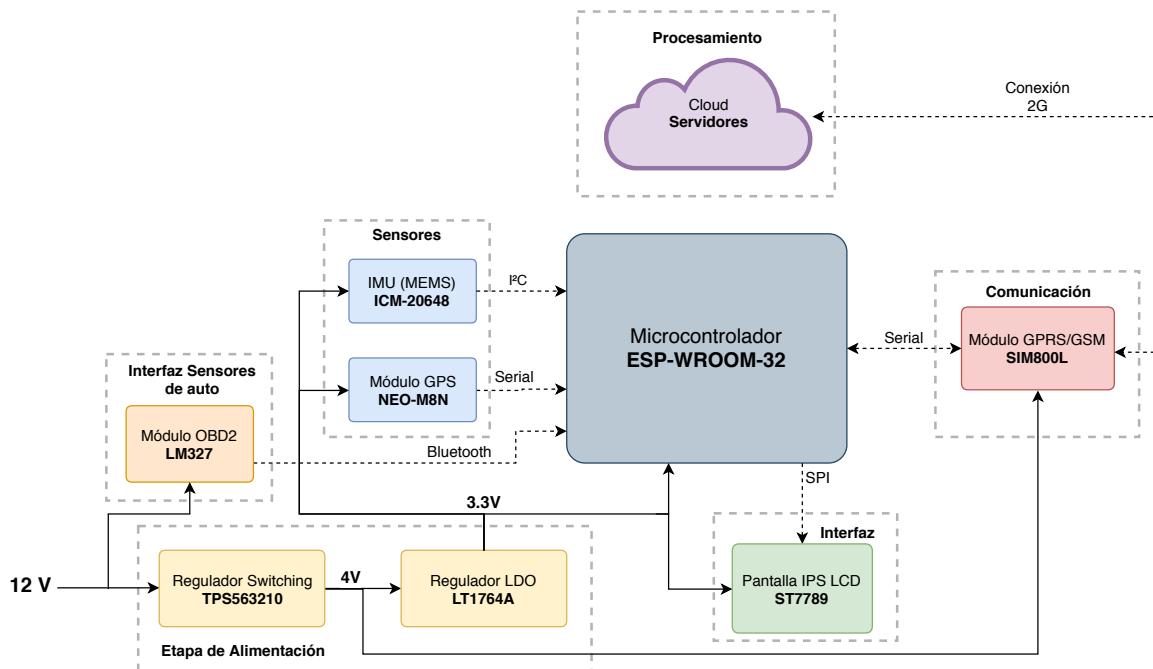


Fig. 4.3 Diagrama de bloques.

4.3.2. Componentes electrónicos

A. Inertial Measurement Unit (IMU)

Este dispositivo cuenta con un acelerómetro, un giroscopio. Los datos registrados por estos sensores se pueden combinar para obtener el perfil de velocidad y aceleración, tanto longitudinal como lateral, del vehículo y estos perfiles aportan mucha información acerca del estilo de conducción del usuario.

Para seleccionar este componente empecemos por determinar las características que este debe poseer. En primer lugar el rango de medición debe ser el adecuado. En las investigaciones citadas [10], [14], [15], [24] La aceleración de los vehículos se suele encontrar dentro del rango de $\pm 7 \text{ m/s}^2$. Los rangos de medición de los acelerómetros se suelen expresar en g, así que se necesita un acelerómetro que pueda medir en un rango de por lo menos $\pm 0.7 \text{ g}$

Realizando el mismo análisis para el giroscopio, en [15] y [18] se tiene que el rango mínimo de medición es de $\pm 57 \text{ rad/s}$

Tabla 4.1 Sensibilidad y Rango del ICM-20648 [25].

Rango del Giroscopio ($^{\circ}/\text{sec}$)	Sensibilidad del Giroscopio (LSB/ $^{\circ}/\text{sec}$)	Rango del Acelerómetro (g)	Sensibilidad del Acelerómetro (LSB/g)
± 250	131	± 2	16384
± 500	65.5	± 4	8192
± 1000	32.8	± 8	4096
± 2000	16.4	± 16	2048

El sensor que se eligió debido a que cumple con las características mencionadas anteriormente es el **ICM-20648** (Fig. 4.4) producido por TDK InvenSense. En las Tablas 4.1 y 4.2 se muestran sus principales características.

Tabla 4.2 Condiciones de Operación del ICM-20648 [25].

Características	Valor
Voltaje de alimentación	1.71 V - 3.6 V
Corriente de operación	3.2 mA
Temperatura de operación	-40 °C hasta 85 °C
Salida digital	I ² C o SPI
DMP	Onboard Digital Motion Processor



Fig. 4.4 ICM-20648, IMU de 6 grados de libertad.

B. Módulo GPS

Este módulo se encargará de detectar la posición exacta del vehículo en coordenadas. Para hacer esto se puede recurrir a distintos servicios del Sistema mundial de navegación por satélite (GNSS). Actualmente se encuentran dos operativos completamente: GPS y GLONASS. Es primero es el servicio hecho por Estados Unidos y el segundo es el realizado por Rusia. Es importante considerar la capacidad de los diferentes módulos para poder acceder a estos dos servicios.

Las características que debe tener el módulo de GPS son las siguientes:

- Bajo consumo.
- Poder acceder tanto a GPS como a GLONASS.
- Por lo menos una frecuencia de muestreo de 1 Hz.

El modulo de GPS que cumple con estas características es el **NEO-M8N** de Ublox (Fig. 4.5). En la Tabla 4.3 se resumen sus principales características.



Fig. 4.5 NEO-M8N, módulo GPS [26].

Tabla 4.3 Características del NEO-M8N [26].

Características	Valor
GNSS	GPS, GLONASS, GALIEO y BeiDou
Precisión	2.5 m
Frecuencia máxima	5 Hz
Protocolos de Comunicación	UART, I ² C, SPI
Voltaje de operación	2.7 V - 3.6 V
Corriente de operación	32 mA
Corriente máxima	67 mA
Temperatura de Operación	-40 °C hasta 85 °C

C. Módulo GPRS/GSM

El módulo GPRS/GSM es capaz de conectarse a las red 2G. Esta red es la que tiene mayor cobertura en el Perú (Fig. 4.6) ya que es la más antigua. Esta red puede alcanzar hasta una velocidad de 114 kbps y es la de menor consumo energético, comparada con 3G y 4G. Usando este módulo se enviarán los datos recopilados por el sistema a través de Internet y se asegurará su conexión en todo momento.

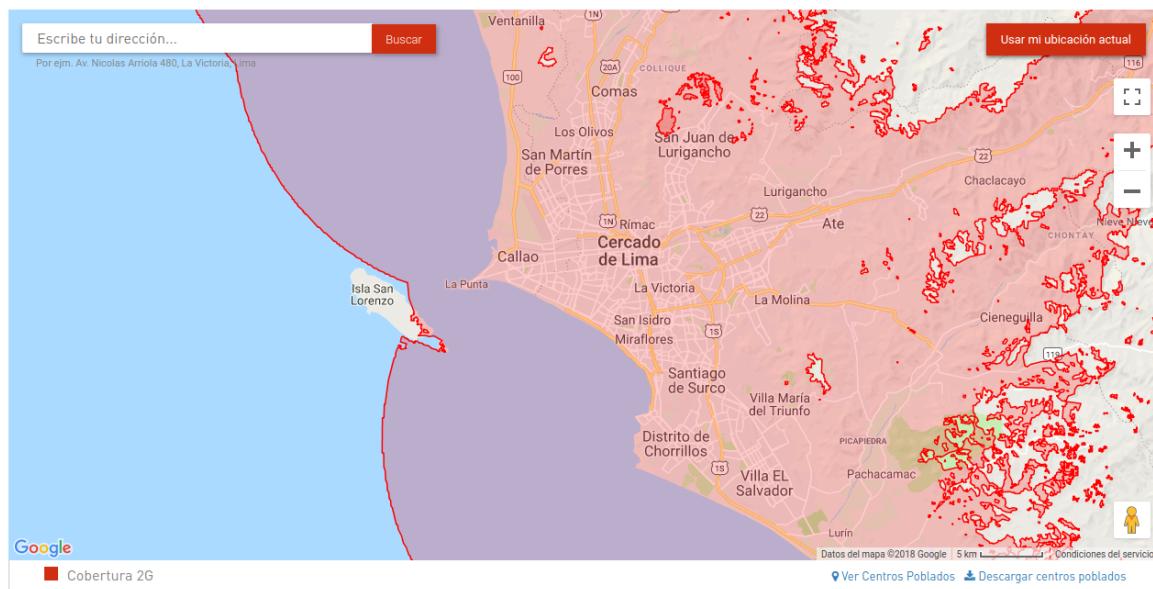


Fig. 4.6 Cobertura de la red 2G de Claro [27].

Una opción muy popular es el módulo **SIM 800L** (Fig. 4.7) que debido a sus características, expuestas en la Tabla 4.4, su disponibilidad y su bajo precio lo hacen adecuado para ser usado en este sistema.



Fig. 4.7 SIM 800L, módulo GPRS/GSM [28].

Tabla 4.4 Características del SIM 800L [28].

Características	Valor
Voltaje de operación	3.4 V - 4.4 V
Corriente promedio en Reposo	18.7 mA
Corriente promedio durante transmisión	453.57 mA
Corriente máxima	2 A (Solo durante ráfaga de transmisión)
Temperatura de operación	-40 °C hasta 85 °C
Velocidad de transmisión	máx. 85.6 kbps
Bandas de frecuencia	Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900

D. Pantalla

La pantalla cumplirá el rol de mostrar la clasificación del estilo de conducción del usuario. Para poder transmitir la información sin generar distracciones se usarán símbolos y colores para representar los estilos de conducción.

Además se necesita que la pantalla se pueda observar bien tanto a la luz del día, por lo que su ángulo de visibilidad y su brillo serán factores importantes también.

Por último, el tamaño de la pantalla debe permitir la identificación del símbolo por parte del conductor. Basados en estos parámetros se puede elegir entre usar 3 tecnologías muy populares: LCD, OLED e IPS LCD.

Las pantallas LCD pueden alcanzar una densidad de imagen más alta que las OLED. Sin embargo, las OLED tienen un mejor ángulo de visión y un menor consumo. Por otro lado, las IPS LCD tienen también un muy buen ángulo de visión y buena densidad de imagen.

En la Tabla. 4.5 se pueden ver 3 alternativas para la pantalla. De estas se escoge la pantalla IPS LCD (Fig. 4.8), ya que tiene buen ángulo de visibilidad, tamaño y resolución.

E. Módulo OBD2

Este módulo se encargará de acceder a todos los parámetros internos del auto conectándose directamente a las ECU (*Electronic Control Units*) del motor. Para poder leer estos parámetros

Tabla 4.5 Alternativas de pantallas

Nr. Producto	Tecnología	Controlador	Tamaño	Voltaje de Operación	Precio	Resolución
1431	OLED	SSD1351	1.5 in	3.3 V o 5 V	\$39.95	128x128
2088	LCD	ST7735R	1.44 in	3.3 V o 5 V	\$14.95	128x128
3787	IPS LCD	ST7789	1.54 in	3.3 V o 5 V	\$19.95	240x240

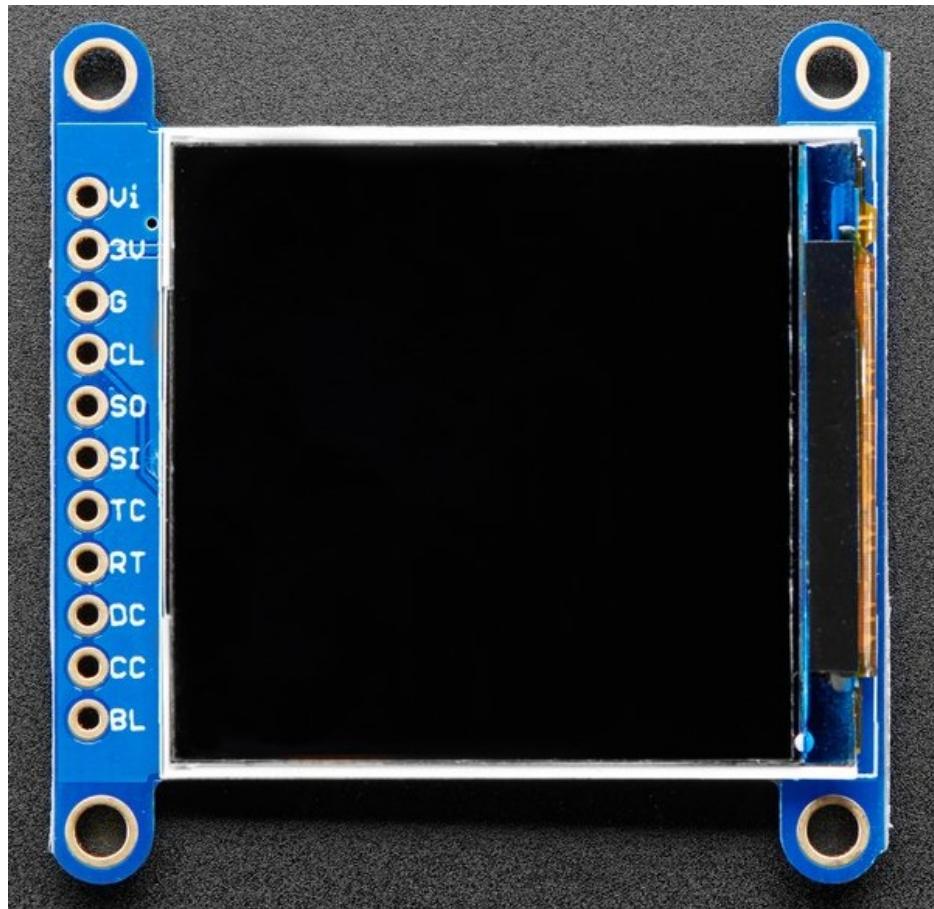


Fig. 4.8 Adafruit 1.54" 240x240 Wide Angle TFT LCD Display - ST7789 [29].

diversas marcas de autos usan distintos protocolos de comunicación. En la Tabla 4.6 Se muestran distintos protocolos y en cada columna circuitos integrados ELMXXX que se usan para comunicarse con el puerto OBD2.

Para maximizar la compatibilidad con vehículos se escoge el modulo **ELM327**. Este IC convierte los protocolos mencionados a comunicación serial y puede transmitir los datos

Tabla 4.6 Protocolos y Circuitos integrados [30].

	ELM323	ELM325	ELM327	ELM329	ELM329L
SAE J1850-PWM			✓		
SAE J1850-VPW			✓		
ISO 9141-2	✓		✓		
ISO 14230-4 (slow)	✓		✓		
ISO 14230-4 (fast)	✓		✓	✓	✓
ISO 15765-4 (CAN)			✓	✓	✓
SAE J1939 (250kbps)			✓	✓	✓
SAE J1939 (500kbps)			✓	✓	✓
SAE J1708 (J1587)		✓			
SAE J1708 (J1922)		✓			

a través de un cable, WiFi o Bluetooth. Para esta aplicación se escoge la versión que usa Bluetooth (Fig. 4.9) ya que el conector se encontrará a menos de 2 m del conector, permitiéndole usar esta tecnología que consume menos energía que el WiFi y es inalámbrica también.



Fig. 4.9 ELM327 - OBD2 interface [30].

F. Microcontrolador

El microcontrolador a seleccionar necesita poder comunicarse con todos los módulos anteriores, poder manejar protocolos como MQTT y CoAP y además realizar procesamiento básico de fusión de sensores (mezclar los datos obtenidos del IMU y del GPS). Se ha escogido para este sistema al ESP-WROOM-32 (Fig. 4.11)

Este microcontrolador cuenta con el diagrama de bloques expuesto en la Fig. 4.10 y como se puede apreciar, cuenta con I²C para comunicarse con el IMU, UART para comunicarse con el módulo GPS y el módulo GPRS/GSM, SPI para controlar la pantalla y Bluetooth para comunicarse con el módulo OBD2. Además presenta la suficiente potencia para realizar las tareas anteriormente descritas. Un resumen de sus características se encuentra en la Tabla 4.7.

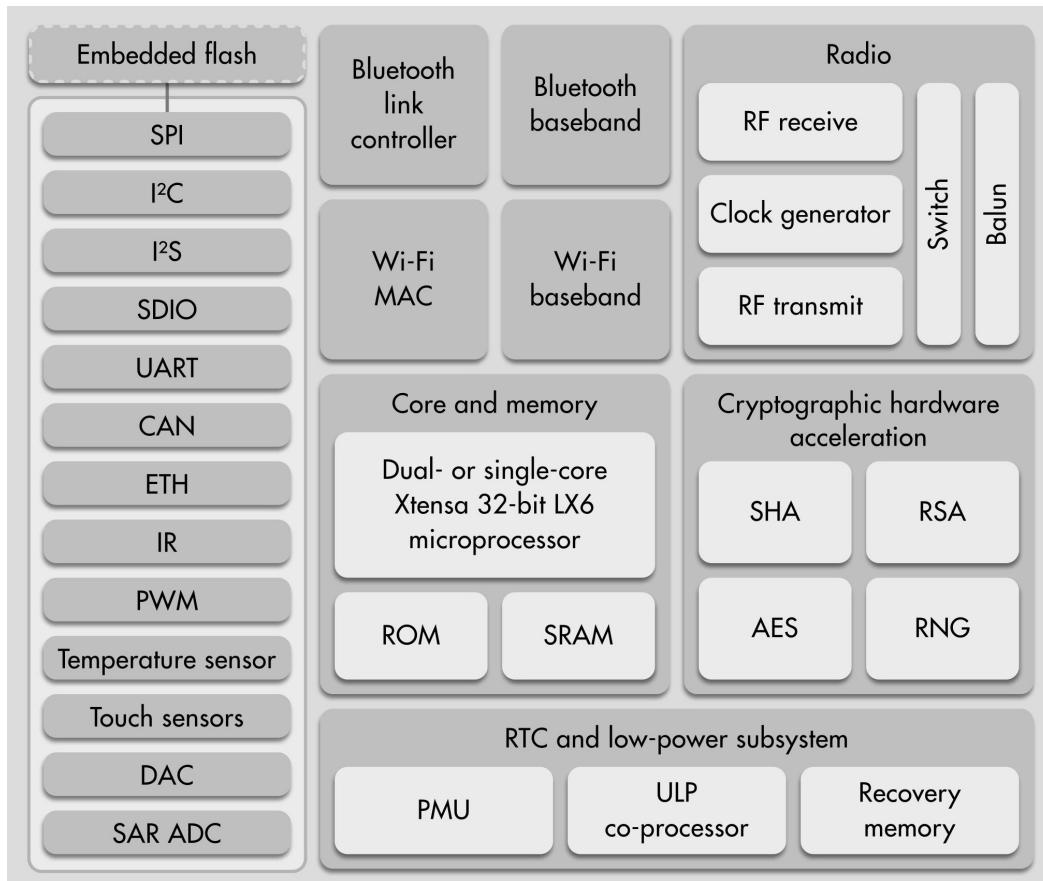


Fig. 4.10 Diagrama de Bloques del ESP32 [1].

G. Regulador switching

Una vez escogidos los componentes a usar en las secciones anteriores, se calculará el consumo total de los componentes para poder diseñar la etapa de alimentación del sistema. En la Tabla 4.8 se encuentra un resumen de cada dispositivo con su voltaje de operación y su corriente máxima de consumo.

Sin embargo, El módulo OBD2 (ELM237) no estará conectado junto con los demás componentes. Este módulo se conectaría al puerto OBD2 del Auto y obtendrá energía de esta misma conexión sin necesidad de diseñarle un regulador de voltaje.



Fig. 4.11 ESP-WROOM-32 [2].

Tabla 4.7 Características del ESP-WROOM-32 [31].

Características	Valor
Voltaje de Operación	2.7 V - 3.6 V
Corriente máxima	500 mA
Corriente de operación	80 mA
Corriente en sleep mode	5 µA
Frecuencia del CPU	240 MHz
Temperatura de operación	-40 °C hasta 85 °C
Interfaces	SD card, UART, SPI, SDIO, I ² C, LED PWM, Motor PWM, I 2 S, IR, contador de pulsos, GPIO, sensor táctil capacitivo, ADC, DAC
Connectividad	WiFi (802.11 b/g/n (802.11n up to 150 Mbps) Bluetooth (Bluetooth v4.2 BR/EDR y BLE)

Tabla 4.8 Consumo de los componentes del sistema.

Dispositivo	Voltaje de Operación	Corriente máxima	Potencia máxima
IMU (ICM-20648)	3.3 V	3.2 mA	10.56 mW
Módulo GPS (NEO-M8N)	3.3 V	67 mA	221.1 mW
Módulo GPRS/GSM (SIM800L)	4 V	2000 mA	8000 mW
Pantalla IPS LCD	3.3 V	50 mA	165 mW
OBD2 (ELM237)	5 V	12 mA	60 mW
ESP-WROOM-32	3.3 V	500 mA	1650 mW

Si sumamos la potencia máxima de cada dispositivo, obtendremos la potencia mínima necesaria que se debe poder entregar por el regulador de voltaje. Esta suma da como resultado **10.046 W**.

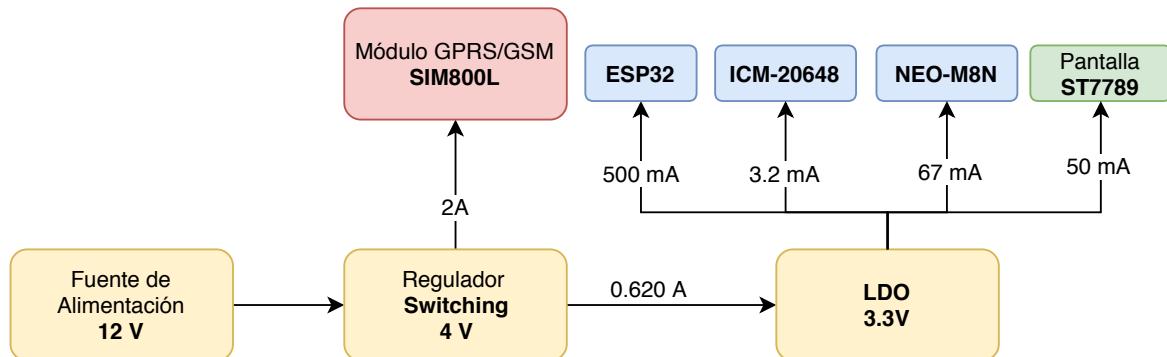


Fig. 4.12 Diagrama de bloques de las etapas de alimentación con la corriente máxima de cada dispositivo.

Se propone dos etapas de regulación de voltaje debido a que se necesitan 2 niveles de voltaje, 4 V y 3.3 V. En la primera etapa se usará un regulador switching para obtener 4 V de los 12 V disponibles en la alimentación. Y en la segunda etapa se usará un LDO (Low Dropout Regulator) para obtener los 3.3 V a partir de los 4 V. En la Fig. 4.12 se dispone de manera gráfica e ideal la potencia y las corrientes máximas que deben poder entregar los reguladores de voltaje.

Entonces usando la Fig. 4.12 Se tiene que el regulador switching debe poder entregar por lo menos 2.62 A. Se selecciona el circuito integrado **TPS563210** que tiene las características mencionadas en la Tabla 4.9

Tabla 4.9 Características del TPS563210 [32].

Características	Valor
Voltaje de entrada	4.5 V - 17 V
Voltaje de salida	0.76 V - 7 V
Corriente máxima de operación	3 A
Frecuencia de conmutación	650 kHz
Temperatura de operación de la juntura	-40 °C - 150 °C
Resistencia térmica de juntura-ambiente	87 °C/W

Se calcula entonces si este regulador switching es capaz de entregar 2.62 A o 10.046 W sin usar un disipador. Para esto se usa la Ecuación 4.1

$$T_J = T_A + (R_{\theta JA} \times \text{Potencia}_{dis}) \quad (4.1)$$

La Potencia_{dis} en esta ecuación es la potencia disipada la cual se puede calcular a partir de la eficiencia del regulador. Según el datasheet [32], se tiene una eficiencia del 93% cuando el voltaje de salida es 4 V y la corriente de salida es 2.6 A.

$$\text{Potencia}_{dis} = \frac{\text{Potencia}_{salida}}{\text{Eficiencia}} \times (1 - \text{Eficiencia}) \quad (4.2)$$

Usando la Ecuación 4.2 se obtiene que el regulador disipará:

$$\text{Potencia}_{dis} = 0.75 \text{ W}$$

Al usar $T_A = 25^\circ\text{C}$, $R_{\theta JA} = 87^\circ\text{C}/\text{W}$ y $\text{Potencia}_{dis} = 0.75 \text{ W}$ en la Ecuación 4.1 resulta:

$$T_J = 90.25^\circ\text{C}$$

Esto significa que el regulador podrá operar sin necesidad de usar un disipador para esta aplicación ya que el máximo para T_J es 150 °C

H. Regulador LDO

Ahora es momento de elegir el regulador LDO. Para esta etapa se usará el IC **LT1764A** que posee las características mencionadas en la Tabla 4.10.

Tabla 4.10 Características del LT1764A [33].

Características	Valor
Voltaje de entrada	2.7 V - 20 V
Voltaje de salida	3.3 V
Corriente máxima de operación	3 A
Dropout Voltage	340 mV a 3 A
Temperatura de operación de la juntura	-40 °C - 150 °C
Resistencia térmica de juntura-ambiente	30 °C/W

Se comenzará por realizar el análisis térmico de este integrado. La potencia disipada se calcula usando la Ecuación 4.3.

$$Potencia_{dis} = I_{OUT(MAX)} \times (V_{IN(MAX)} - V_{OUT}) + I_{GND} \times V_{IN(MAX)} \quad (4.3)$$

Se reemplazan los siguientes valores en esta ecuación:

$$V_{OUT} = 3.3 \text{ V} \quad V_{IN(MAX)} = 4 \text{ V} \quad I_{OUT(MAX)} = 0.62 \text{ A} \quad I_{GND} = 20 \text{ mA}$$

Y se obtiene el siguiente resultado:

$$Potencia_{dis} = 0.514 \text{ W}$$

Ahora se aplica la Ecuación 4.1 con la que obtendremos la Temperatura de la juntura (T_J) que se alcanzará al disipar esa potencia.

$$T_J = 40.42 \text{ °C}$$

Como la $T_{J(MAX)} = 150 \text{ °C}$, se puede afirmar que no se necesita disipador para usar este regulador. Con el sistema de alimentación definido se puede graficar el diagrama de bloques completo del sistema (Fig. 4.13). En esta ocasión se colocaron los valores promedio de

corriente que consumirá cada dispositivo. Observando este diagrama se puede obtener la potencia promedio que consumirá el dispositivo $P_{prom} = 2.66 \text{ W}$.

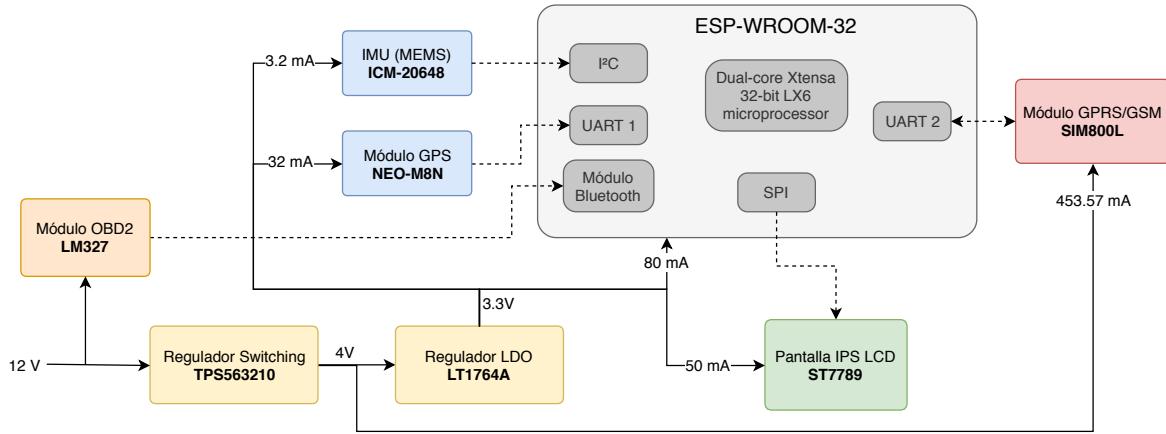


Fig. 4.13 Diagrama de bloques del sistema con corrientes promedio.

4.3.3. Diagramas electrónicos

En esta sección se presentarán las conexiones de los componentes seleccionados anteriormente. Estas conexiones se realizaron siguiendo las recomendaciones que se encuentran en las hojas de datos de cada componente.

A. IMU

Se realiza el esquemático (Fig. 4.14) mostrando las conexiones necesarias para poder alimentar al módulo y para que este pueda comunicarse con el microcontrolador. Se establece el voltaje de alimentación y el voltaje lógico a 3.3 V. En la hoja de datos [25] se recomiendan los valores de los capacitores y se conectan los pines necesarios para poder implementar el protocolo de comunicación I²C. El pin 9 se conecta a GND para establecer la dirección I²C a b1101000.

B. Módulo GPS

Se realiza el esquemático (Fig. 4.15) de las conexiones necesarias para alimentar al módulo e implementar la comunicación con el microcontrolador. En este caso se alimenta al módulo con 3.3 V y se implementa comunicación serial con el microcontrolador. Además se tiene un

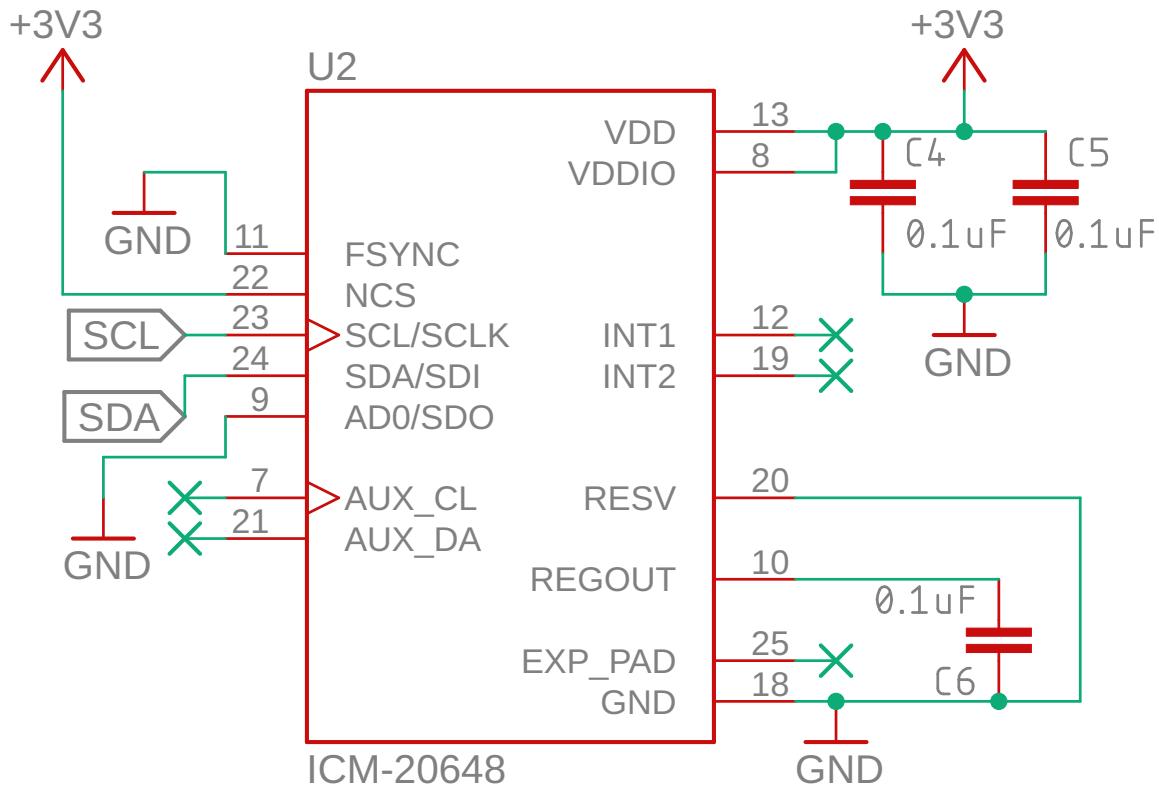


Fig. 4.14 Esquemático del módulo ICM-20648.

conector SMA para poder conectar una antena externa y también se cuenta con una batería de 3 V, con una capacidad de 5 mAh según [34].

Esta batería, que se recomienda en la hoja de datos [26], se encargará de mantener activo al módulo GPS en modo de bajo consumo durante la noche cuando la fuente de voltaje principal este desconectada. Esto se quiere hacer debido a que el GPS encontrará la posición de una forma más rápida si se mantiene activo. En cambio, cuando se apaga y se vuelve a encender se le llama "encendido en frío" y toma más tiempo.

C. Módulo GPRS/GSM

A partir de los parámetros mencionados en la Tabla 4.4 y de la hoja de datos [28] se realiza el esquemático (Fig. 4.16) de las conexiones necesarias para alimentar al módulo e implementar la comunicación con el microcontrolador.

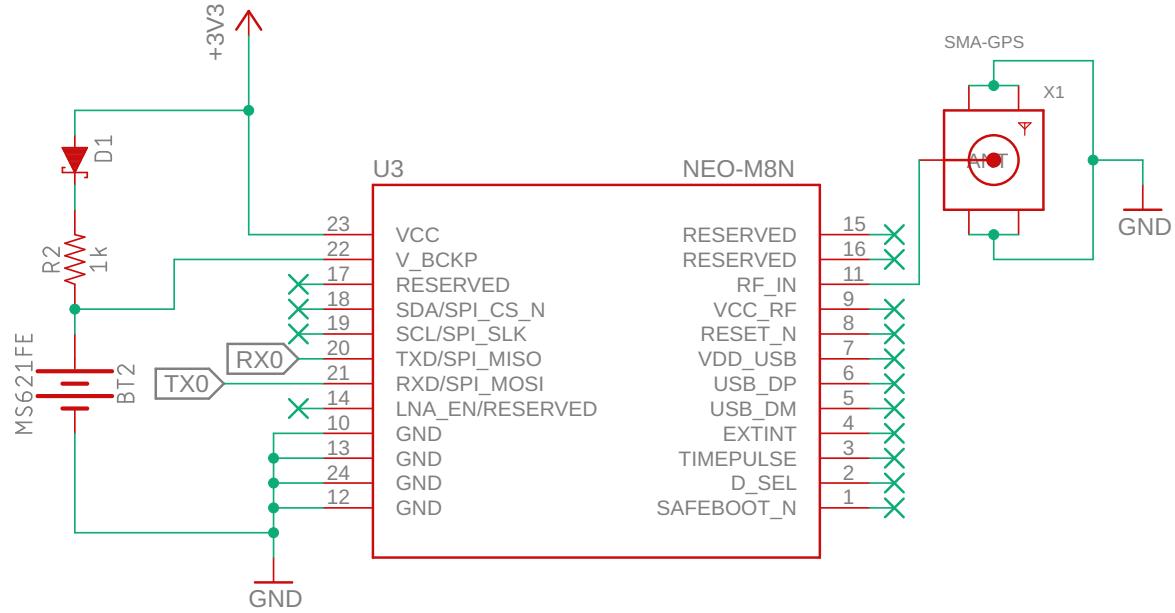


Fig. 4.15 Esquemático del módulo NEO-M8N.

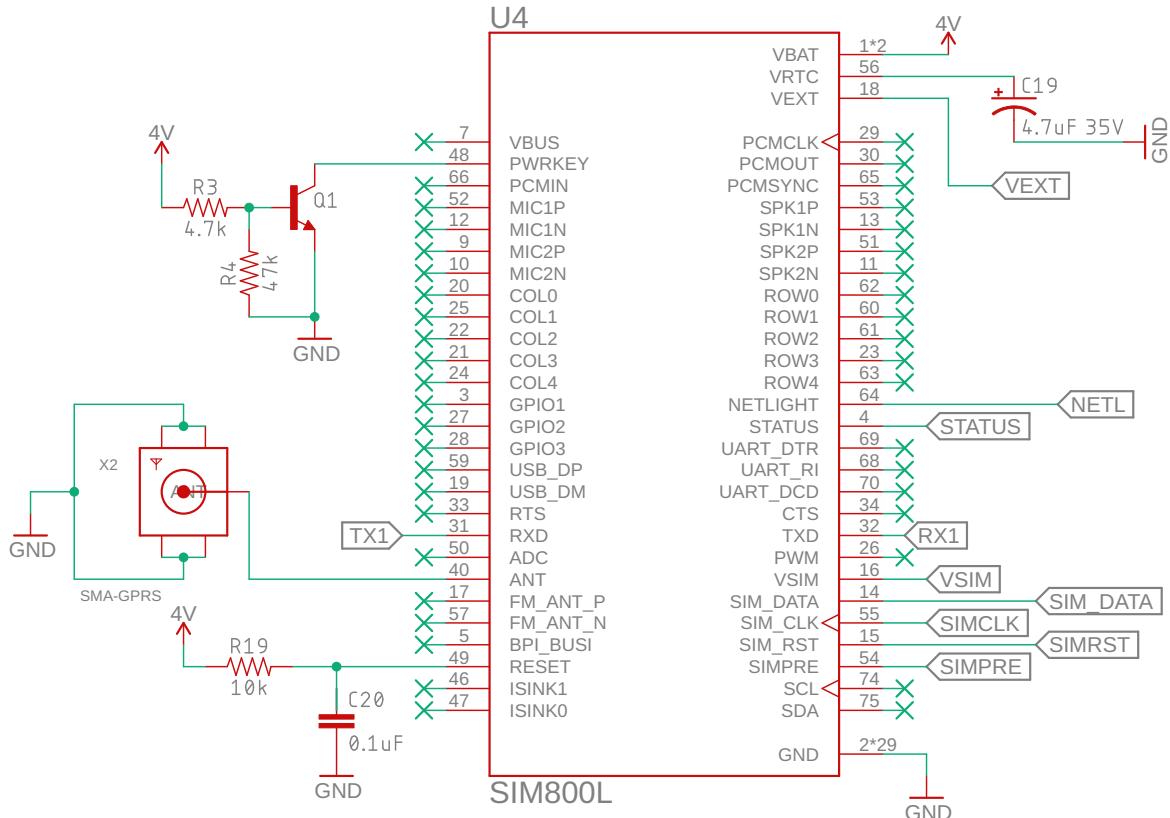


Fig. 4.16 Esquemático del módulo SIM800L.

En este caso el módulo es alimentado con 4 V y se usará comunicación serial. Los capacitores elegidos son recomendaciones de la hoja de datos [28] al igual que el transistor que cumplirá la función de encender y apagar correctamente el módulo.

Además se necesita de un módulo que lea una tarjeta SIM. En la Fig. 4.17 se muestra el esquemático de este módulo, el lector de tarjetas SIM (SIM8055).

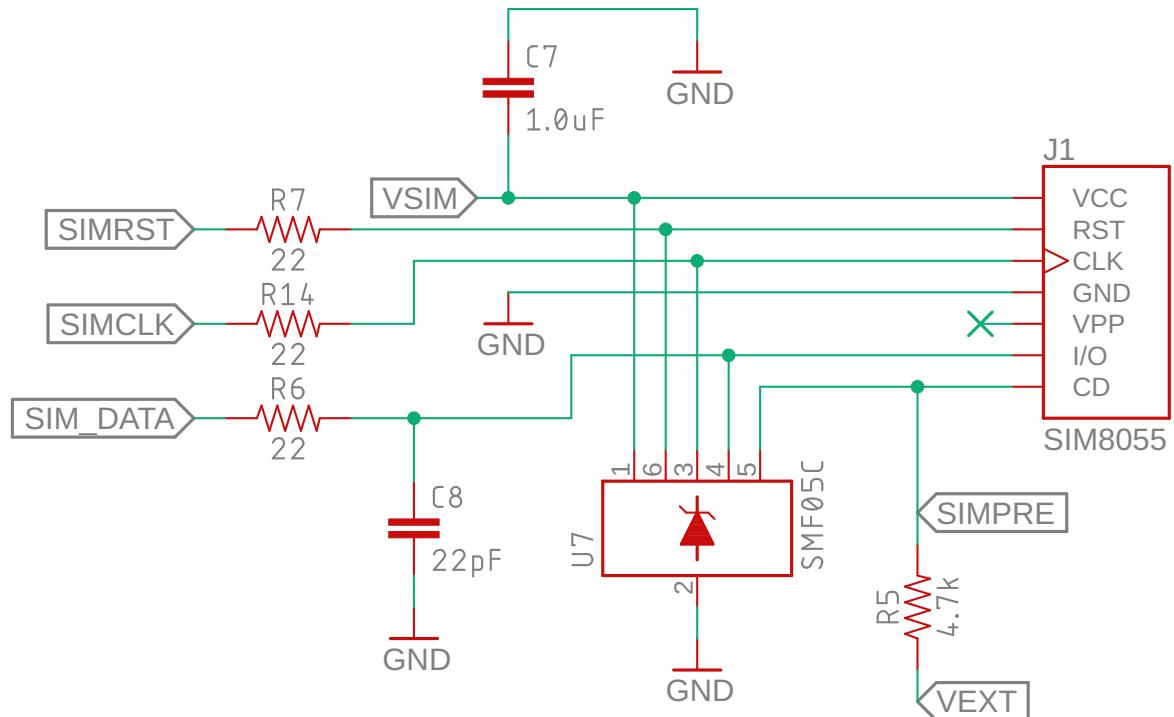


Fig. 4.17 Esquemático del módulo SIM8055.

Por último, se tiene en la Fig. 4.18 el esquemático de la conexión de dos LEDs que servirán para indicar el estado de la conexión a Internet del módulo y también si este se encuentra encendido o no. Se usará un conector para cablear los LEDs ya que estos no se encontrarán soldados a la PCB porque se necesitan en una posición específica del case del dispositivo.

D. Pantalla

La pantalla se colocará fuera del case donde estará el PCB por lo que se usarán borneras para conectar los pines necesarios al microcontrolador, como indica la Fig. 4.19. Se usa el protocolo SPI para comunicarse con el microcontrolador y se alimenta a la pantalla con 3.3 V. Además se usan solo 8 pines ya que estos son los únicos necesarios para controlar la pantalla.

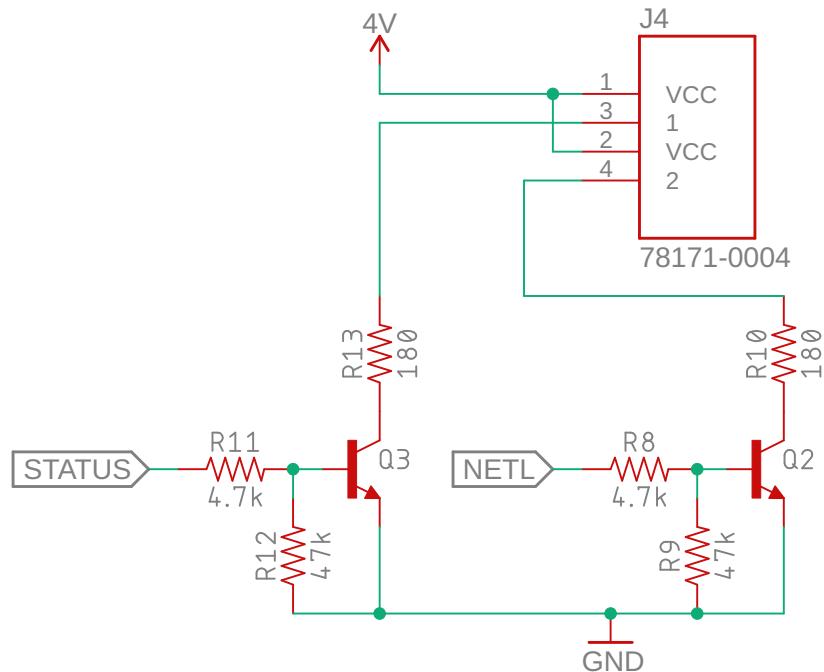


Fig. 4.18 Esquemático las los indicadores LED.

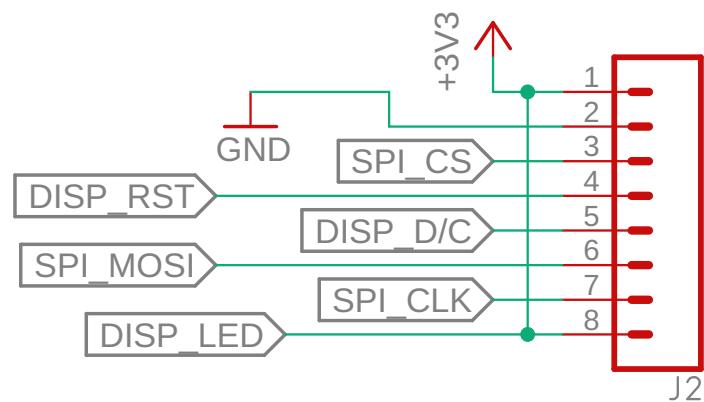


Fig. 4.19 Esquemático bornera para conectar la pantalla.

E. Microcontrolador

En la Fig. 4.20 se muestra el esquemático del microcontrolador y su alimentación. En la hoja de datos [31] se recomienda usar los capacitores que se encuentran conectados al microcontrolador y se muestran también todas las conexiones de los módulos anteriores; 1 conexión I²C, 2 conexiones seriales y 1 conexión SPI.

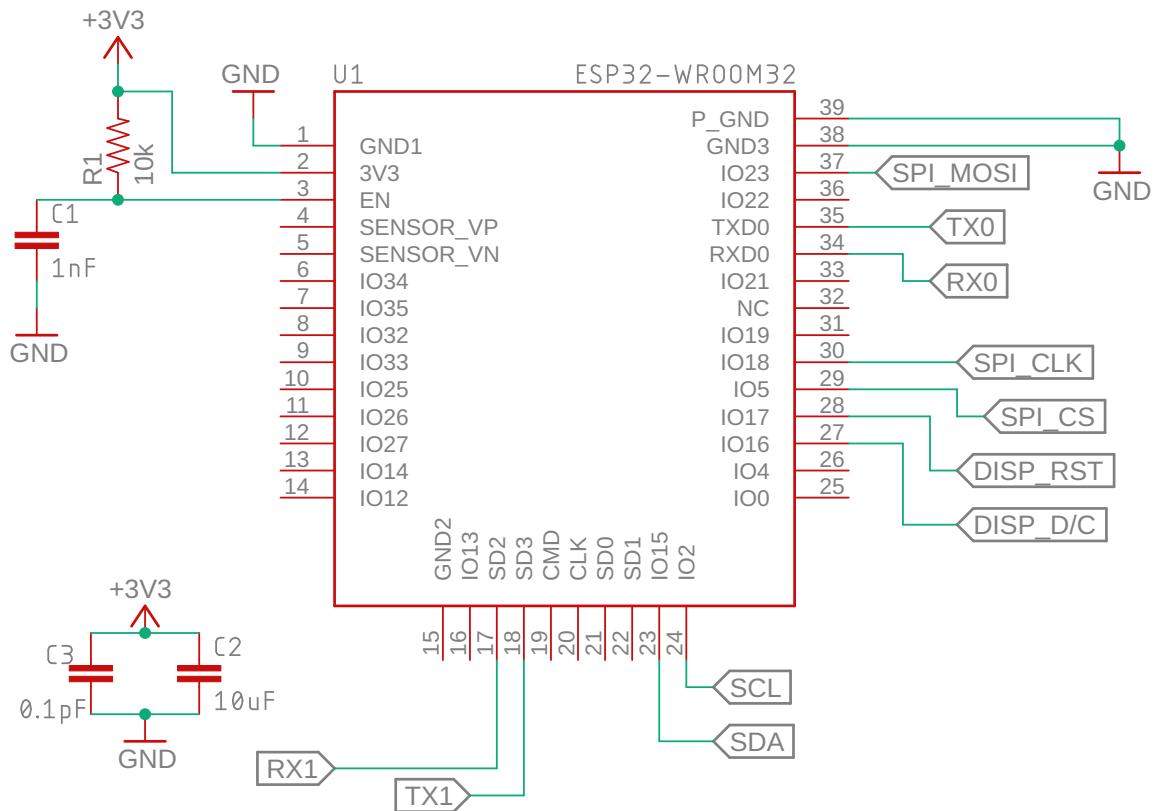


Fig. 4.20 Esquemático de la implementación del ESP-WROOM-32.

F. Regulador swtiching

En la Fig. 4.21 se puede observar la selección de los componentes externos al TPS563210 usados para obtener una salida de 4 V de una fuente de 12 V. El primer paso para seleccionar estos componentes fue determinar el voltaje de salida requerido (4 V). En [32] se tiene que el V_{out} seguirá la Ecuación 4.4.

$$V_{out} = 0.765 \times \left(1 + \frac{R_{17}}{R_{18}}\right) \quad (4.4)$$

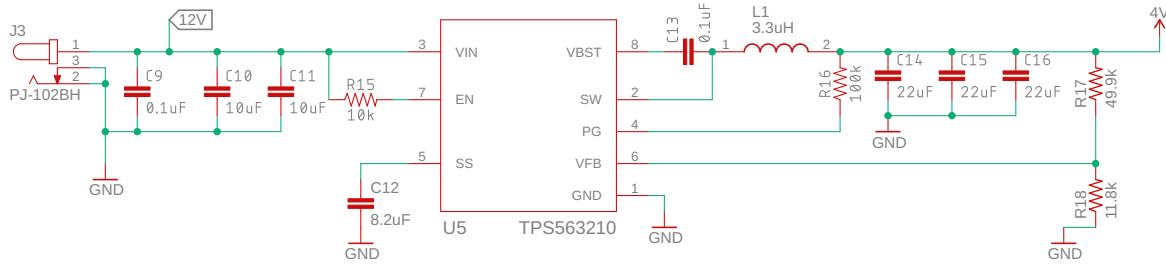


Fig. 4.21 Esquemático de la implementación del TPS563210.

Para asegurar que $V_{out} = 4\text{ V}$ se usan resistencias de la serie E96 (con 1% de tolerancia en su valor). Los valores de R_{17} y R_{18} que cumplen con esta ecuación son $R_{17} = 49.9\text{ k}\Omega$ y $R_{17} = 11.8\text{ k}\Omega$.

Luego se recomienda en [32] usar una inductancia de valor $L_1 = 3.3\mu\text{H}$ y el IC usa una frecuencia de $f_{sw} = 650\text{ kHz}$. Además se necesita el valor máximo que el voltaje de entrada V_{in} puede tomar, que en este caso será $V_{in} = 16\text{ V}$ (Esto es debido a fluctuaciones en el voltaje de la batería). Con estos valores se puede determinar los parámetros Il_{P-P} , Il_{PEAK} , $I_{LO(RMS)}$ y I_{CO} con las ecuaciones:

$$Il_{P-P} = \frac{V_{out}}{V_{in(MAX)}} \times \frac{V_{in(MAX)} - V_{OUT}}{L_O \times f_{sw}} \quad (4.5)$$

$$Il_{PEAK} = I_o + \frac{Il_{P-P}}{2} \quad (4.6)$$

$$I_{LO(RMS)} = \sqrt{I_o^2 + \frac{1}{12} Il_{P-P}^2} \quad (4.7)$$

$$I_{CO} = \frac{V_{out} \times (V_{in} - V_{out})}{\sqrt{12} \times V_{in} \times L_O \times f_{sw}} \quad (4.8)$$

El resultado de las Ecuaciones 4.5, 4.6, 4.7 y 4.8 son:

$$Il_{P-P} = 1.398\text{ A} \quad Il_{PEAK} = 3.319\text{ A} \quad I_{LO(RMS)} = 2.79\text{ A} \quad I_{CO(RMS)} = 0.02\text{ A}$$

Se selecciona entonces el inductor CLF7045NIT-3R3N-D de $3.3\mu\text{H}$ que soporta los parámetros mencionados anteriormente [35] y 3 capacitores C3216X5R0J226M en la salida de $22\mu\text{F}$ que soporta hasta 4 A RMS. Los capacitores de entrada y las demás resistencias se colocan por recomendación de [32].

G. Regulador LDO

En la Fig. 4.22 se puede observar la selección de componentes externos y el esquemático propuesto para el LT1764A. Los capacitores C_{17} y C_{18} toman valores recomendados en la hoja de datos [33]

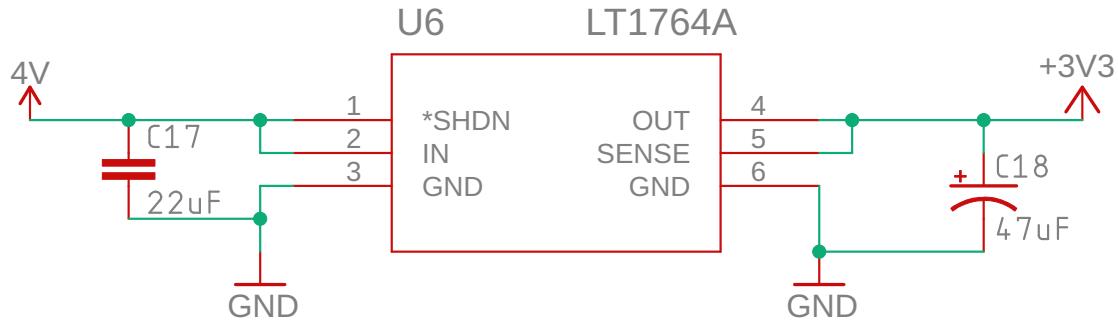


Fig. 4.22 Esquemático de la implementación del LT1764A.

4.3.4. Diseño del PCB

A continuación se incluirá el diseño del PCB en el que irán todos los componentes antes mencionados. Como se aprecia en la Fig. 4.23, el PCB cuenta con dos capas. La capa inferior cuenta con un plano de GND que ayudará no solo a conectar los componentes más fácilmente sino que es recomendado por algunos de los componentes como el SIM 800L [28].

En la Fig. 4.24 se puede apreciar la parte superior del PCB con sus dimensiones en mm y en la Fig. 4.25 se aprecia la parte inferior

Se tuvo una consideración importante al escoger el ancho de las pistas que se encuentran en este PCB. Siguiendo la Fig. 4.12, en donde se pueden ver las corrientes máximas que se alcanzarán. Se tienen pistas por donde pasarán 2.6 A, 2 A y 0.62 A. Para poder elegir el ancho adecuadamente se utiliza la Ecuación 4.9 en dónde I es la máxima corriente permisible, ΔT es el incremento de temperatura permisible y A es la sección transversal de la pista.

$$I = k\Delta T^{0.44}A^{0.725} \quad (4.9)$$

Si expresamos $A = Espesor \times Ancho$ Y despejamos *Ancho*, obtendríamos la Ecuación 4.10

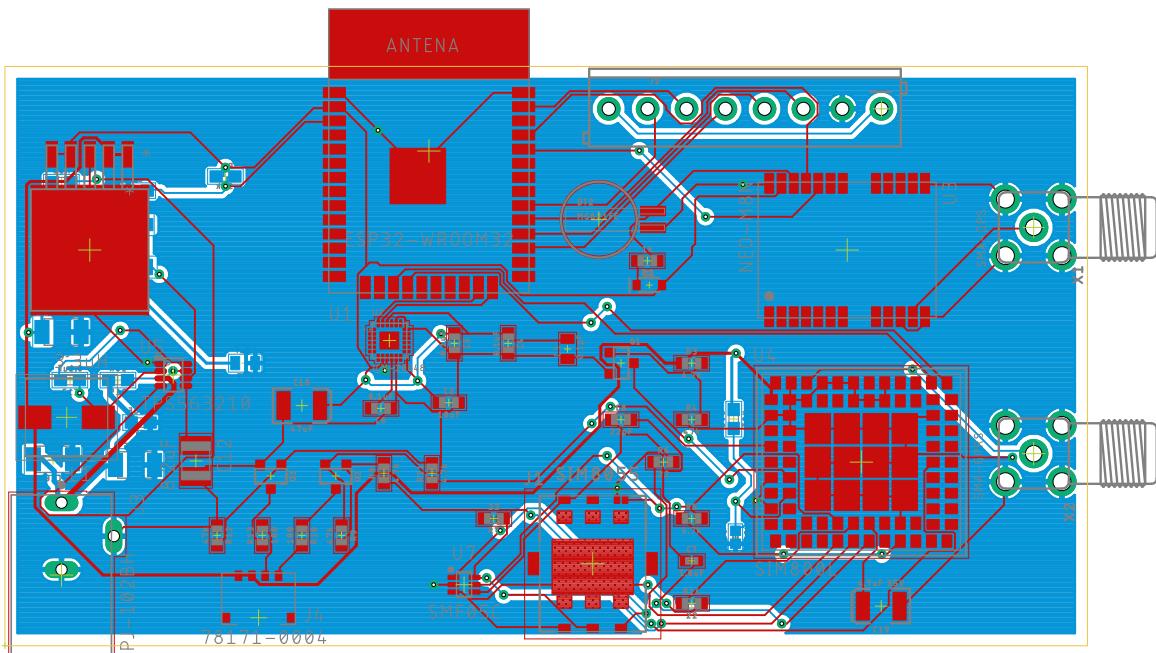


Fig. 4.23 Diseño del PCB.

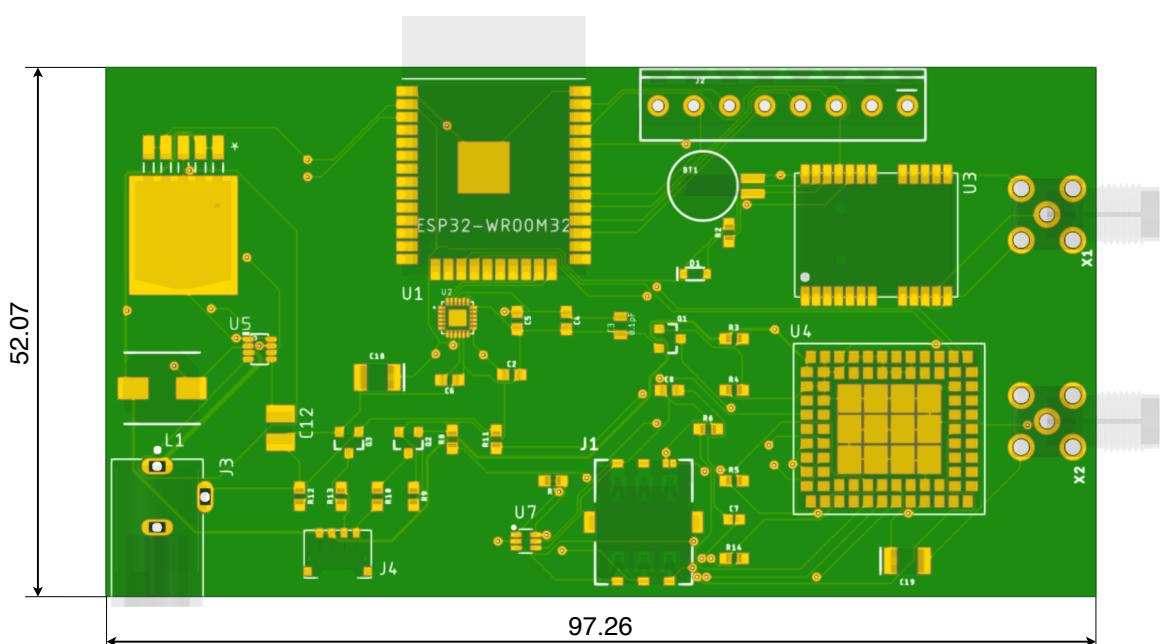


Fig. 4.24 Parte superior del PCB.

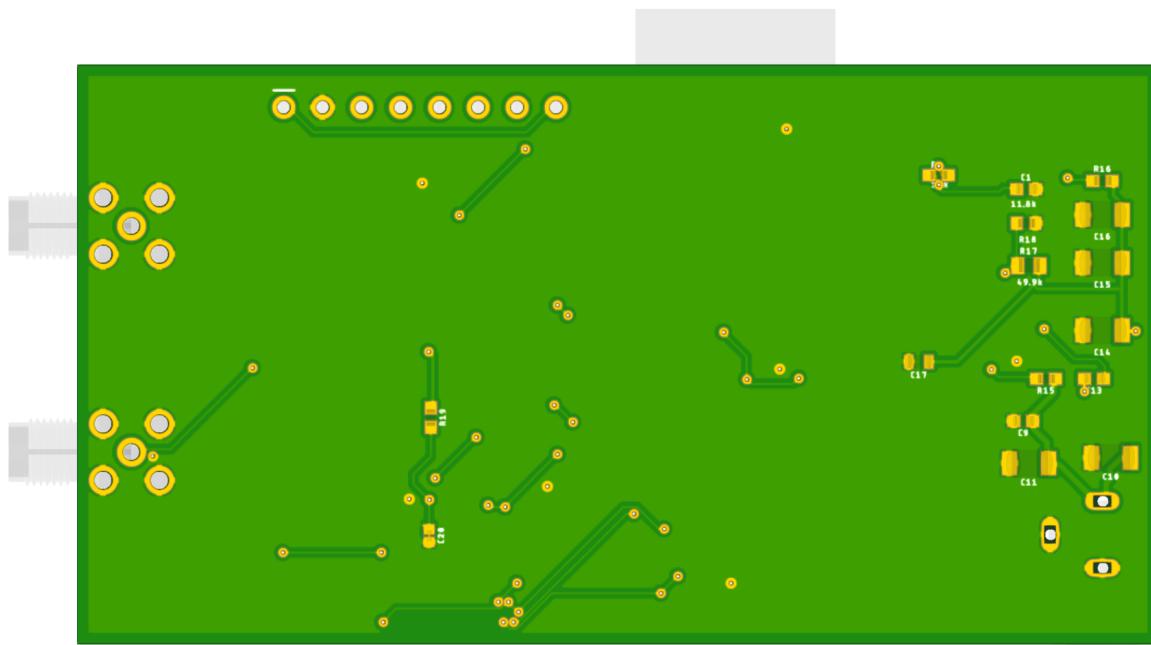


Fig. 4.25 Parte inferior del PCB.

$$Ancho = \left(\frac{I}{k\Delta T^{0.44}} \right)^{\frac{1}{0.725}} \times \frac{1}{Espesor} \quad (4.10)$$

Al usar la Ecuación 4.10 con $\Delta T = 20^\circ\text{C}$, y un espesor de pista de 1 oz/ft² se tienen los mínimos anchos de pista necesarios para cada corriente mencionada. Los resultados se observan en la Tabla 4.11. Sin embargo, estos son solo los anchos mínimos. Es por eso que para una corriente de 620 mA o menor se usará un ancho de pista de 10 mil por defecto y de 6 mil para soldar los pads del SIM 800L.

Tabla 4.11 Anchos de pista mínimos.

Corriente máxima	Ancho mínimo
2.6 A	29 mil
2 A	20.2 mil
620 mA	4 mil

El PCB diseñado tiene componentes que exceden las dimensiones de la placa. Las dimensiones con componentes se puede apreciar en las Figuras 4.26, 4.27 y 4.28

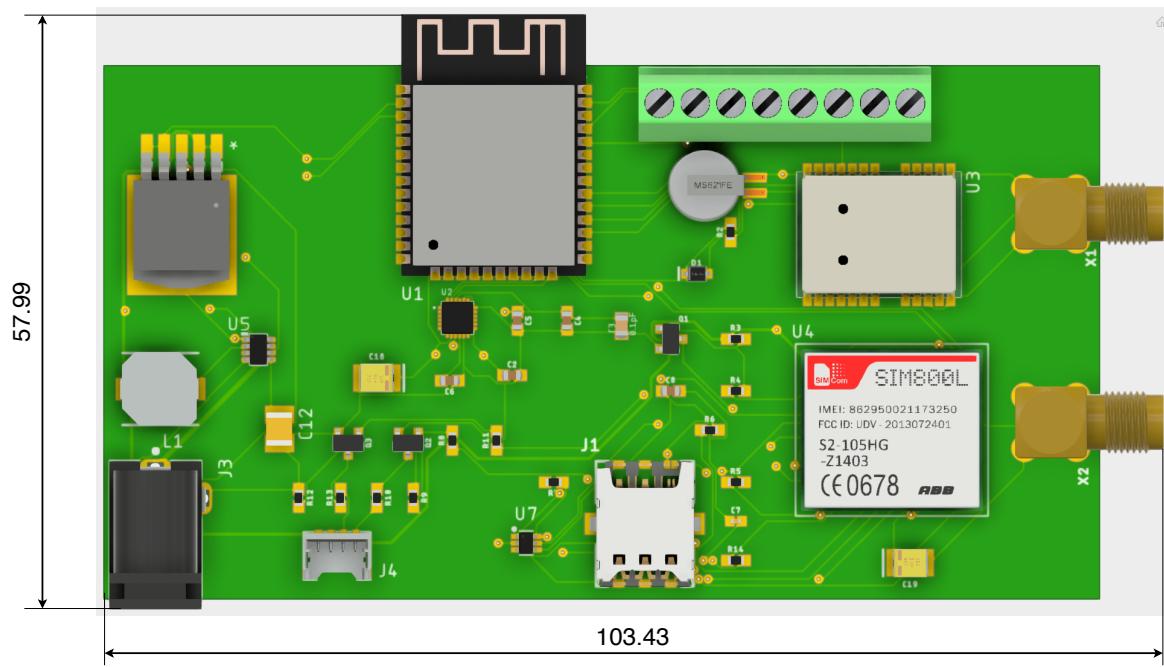


Fig. 4.26 Parte superior del PCB con componentes.

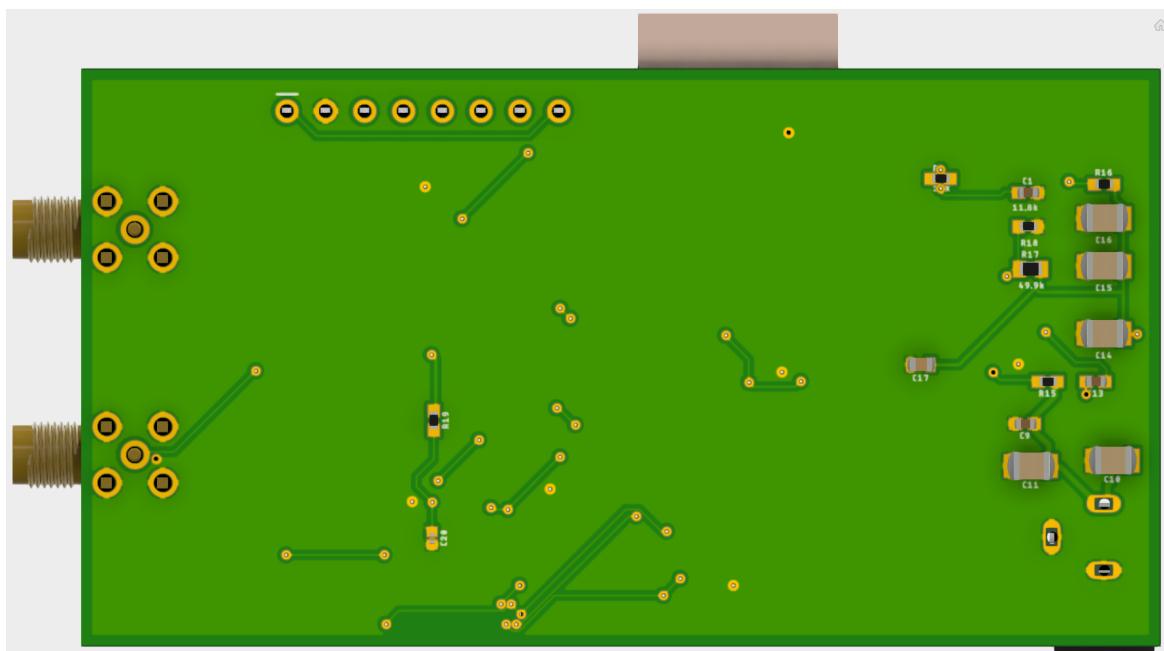


Fig. 4.27 Parte inferior del PCB con componentes.



Fig. 4.28 Vista lateral del PCB.

4.4. Diseño mecánico

4.4.1. Diseño del Case Principal

El PCB antes diseñado tiene las dimensiones $58\text{ mm} \times 103.43\text{ mm} \times 14.9\text{ mm}$. Se diseña entonces un case que pueda alojar esta placa electrónica y que permita acceder a las interfaces que posee (Conexión de alimentación, ranura de Nano SIM y conectores SMA). El case diseñado tendrá las dimensiones $63\text{ mm} \times 108\text{ mm} \times 28\text{ mm}$ y además de las interfaces mencionadas anteriormente tendrá también 2 indicadores LED que mostrarán el encendido del dispositivo y su conexión a Internet. Todos estos elementos se pueden observar en las Fig. 4.32.

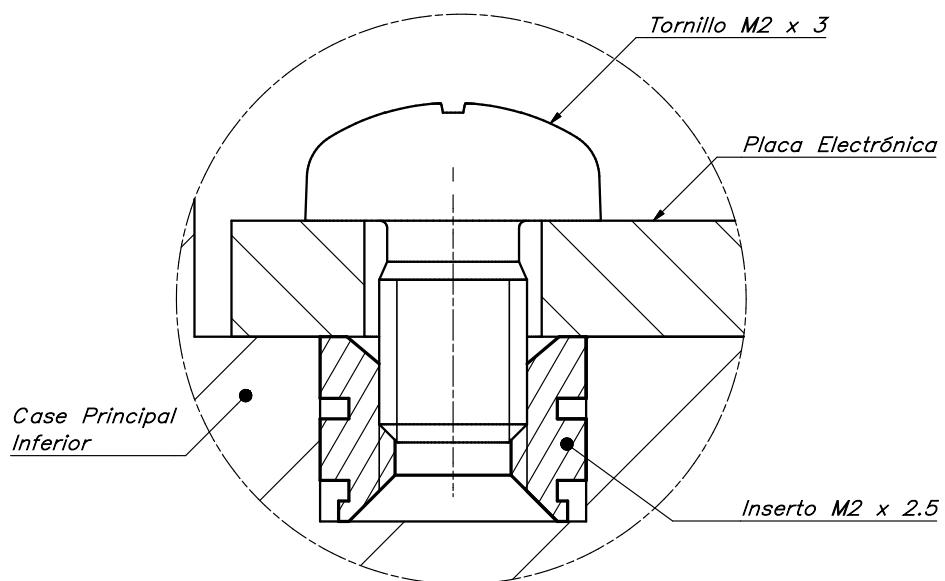


Fig. 4.29 Unión de la tarjeta electrónica con el case.

Este case cuenta con dos partes, una superior y otra inferior. En la inferior se ensamblará la tarjeta electrónica vista en el capítulo anterior. El ensamble (Fig. 4.29) se realizará usando una unión atornillada de un Tornillo M2 x 3 y un Inserto M2 x 2.5 que se coloca en un agujero en el case principal inferior. Este inserto se puede observar en la Fig. 4.30 y esta diseñado para ser usado en plásticos.

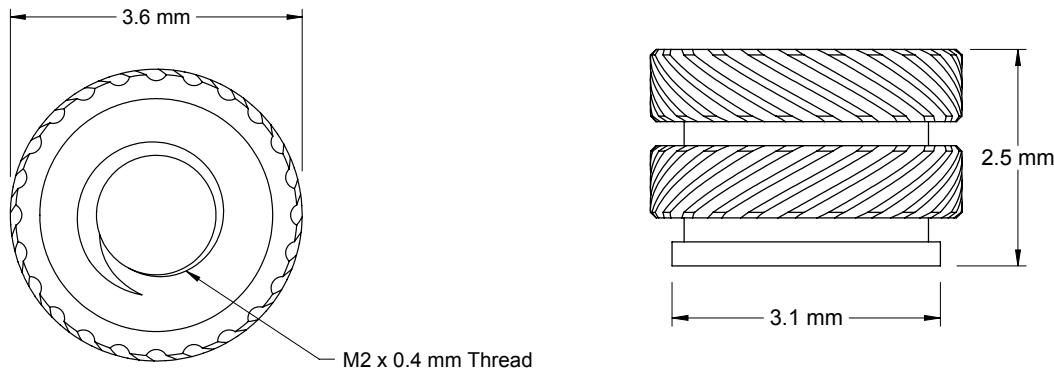


Fig. 4.30 Inserto M2 x 2.5 para plástico.

Luego que la placa electrónica esta correctamente sujetada, se ensambla la parte superior del case. Para este ensamble se utilizan unas pestañas que encajarán entre si para facilitar el posicionamiento de las piezas y "snap fits" para la sujeción de las dos piezas. Esto se puede observar en la Fig. 4.31.

Para el diseño de estas conexiones se tuvo en cuenta una separación de 0.2 mm en las pestañas para asegurar que encajarán una con otra y que existirá un poco de fricción luego de ser impresas. Además se consideró también una separación de 0.3 mm en los snap fits para asegurar su posicionamiento.

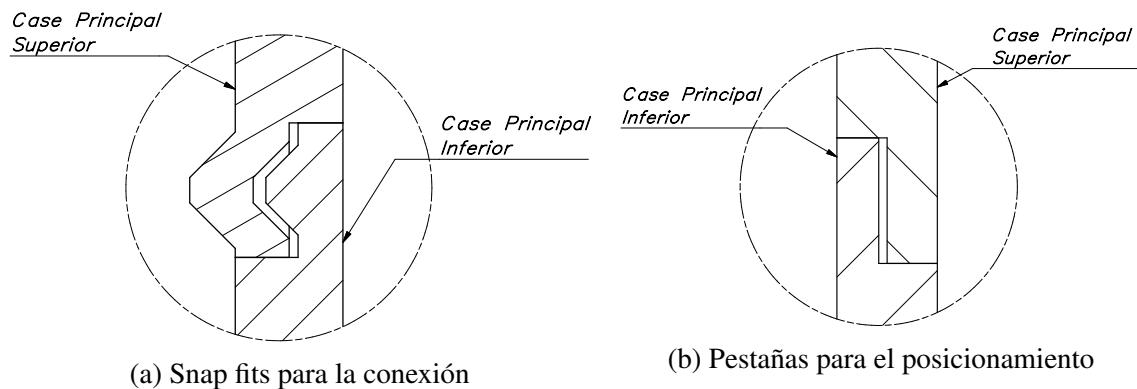


Fig. 4.31 Unión entre la parte superior e inferior del Case Principal

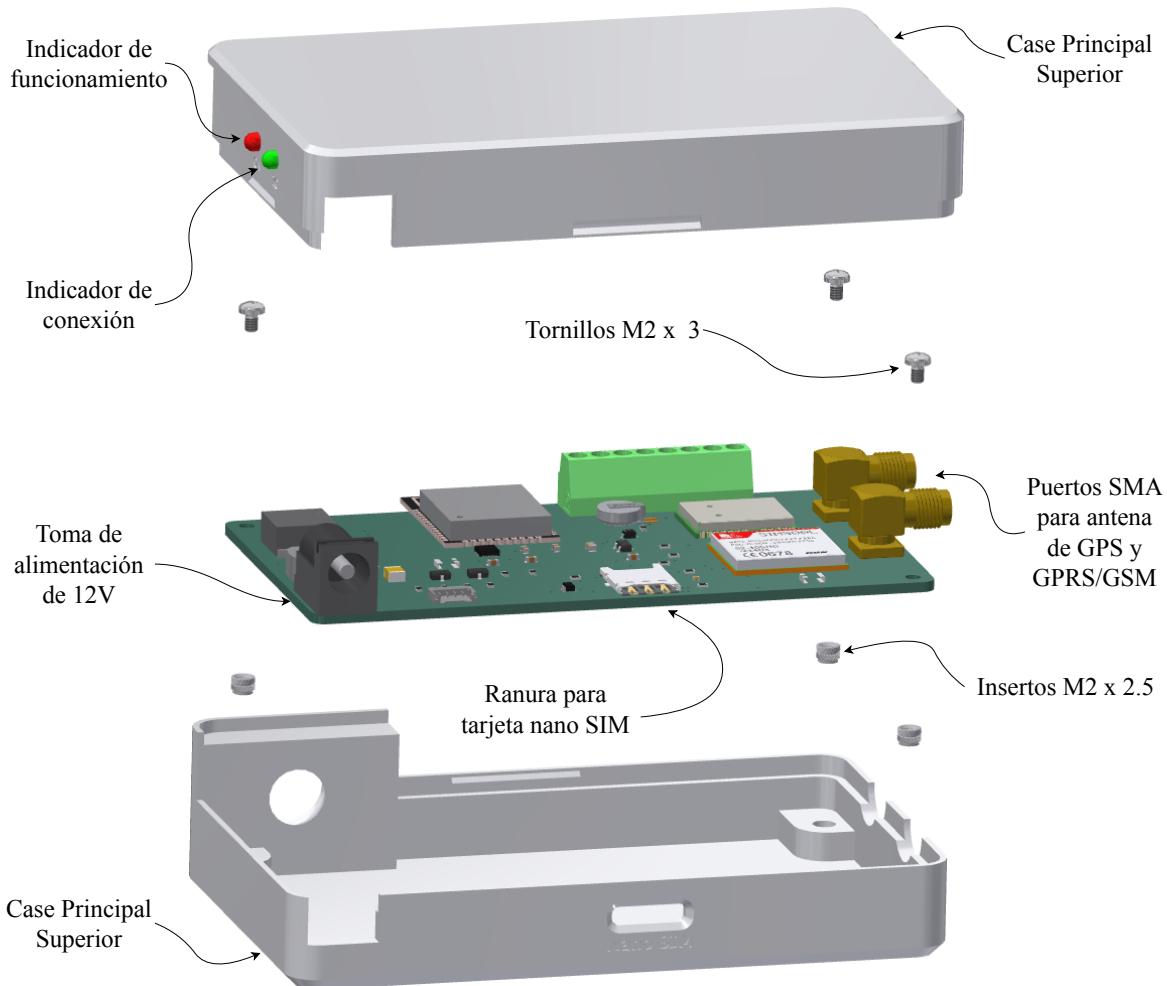


Fig. 4.32 Vista de explosión del case principal.

4.4.2. Selección del brazo de la pantalla

Para seleccionar el brazo que sujete la pantalla se tienen en cuenta dos requerimientos:

- La posición de la pantalla debe ser regulable por el conductor.
- Los cables tienen que poder ser conducidos a través del brazo hacia la pantalla

Se usarán entonces las mangueras a piezas de la marca *Loc-Line* (Fig. 4.33). Estas mangueras se componen de múltiples piezas huecas, cada una se une a la siguiente por medio de una articulación de rótula. Entonces se crea uniones para poder conectar esta manguera con el case principal y con el case de la pantalla. En el caso del case principal se crea un agujero en donde entrará la rótula del primer elemento de la manguera (Fig. 4.34a) y en el

caso de el case de la pantalla se creará un elemento que encajé en el agujero de rótula de la manguera (Fig. 4.34b)

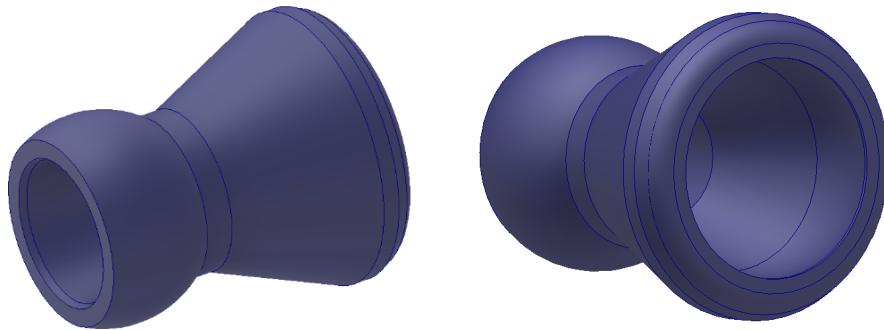


Fig. 4.33 Elemento Loc-Line.

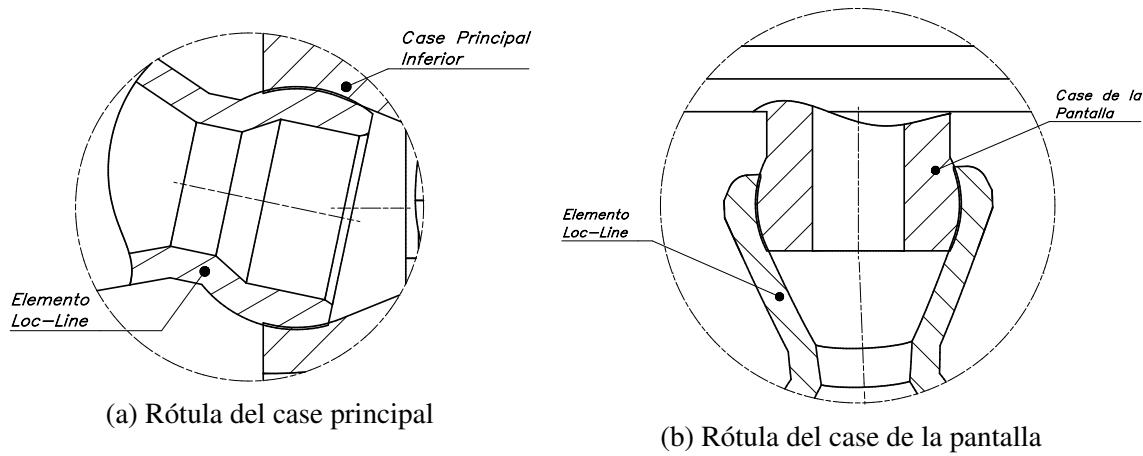


Fig. 4.34 Detalles del ensamblaje de los elementos Loc-Line con las cajas electrónicas

4.4.3. Diseño del case de la pantalla

El case de la pantalla se puede observar en la Fig. 4.36. Este case de dimensiones $43\text{ mm} \times 49\text{ mm} \times 21\text{ mm}$ será impreso también en 3D. Para ensamblar las dos partes del case con la pantalla, se inserta la placa electrónica de la pantalla en un agujero con su misma forma en el case inferior de la pantalla. Luego la parte superior del case es ensamblada usando pestañas para el posicionamiento y snap fits para la sujeción.

La parte superior del case presiona a la placa electrónica de la pantalla contra la parte inferior del case usando 4 columnas. Esto se puede observar en la Fig. 4.35a. Además el snap fit (Fig. 4.35b) de este ensamblaje es distinto del usado en el case principal. En esta

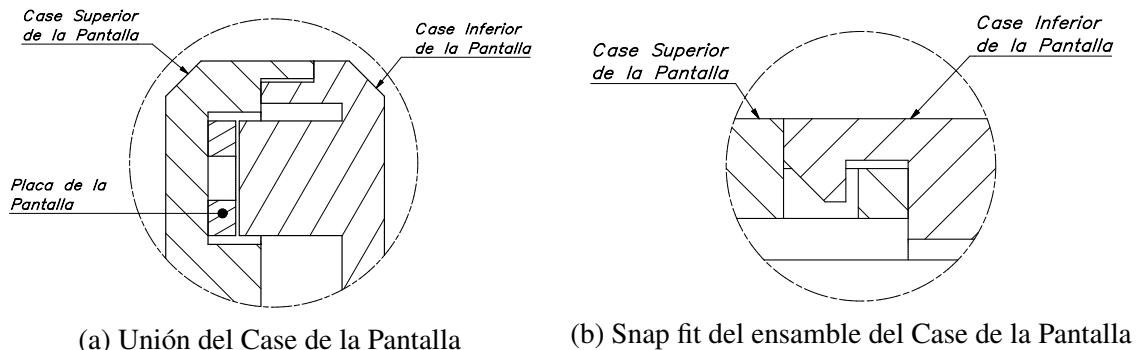


Fig. 4.35 Detalles del ensamblaje del case de la pantalla

ocasión el ángulo de entrada es 45° pero el ángulo de salida es 90° . Esto significa que la unión es inseparable, ya que el ángulo de entrada permitirá el ensamblaje y el de salida lo impedirá. Esto se debe a que la pantalla se podrá mover para acomodarse al conductor en cada situación, lo que hace necesario que el case no se separe por error.

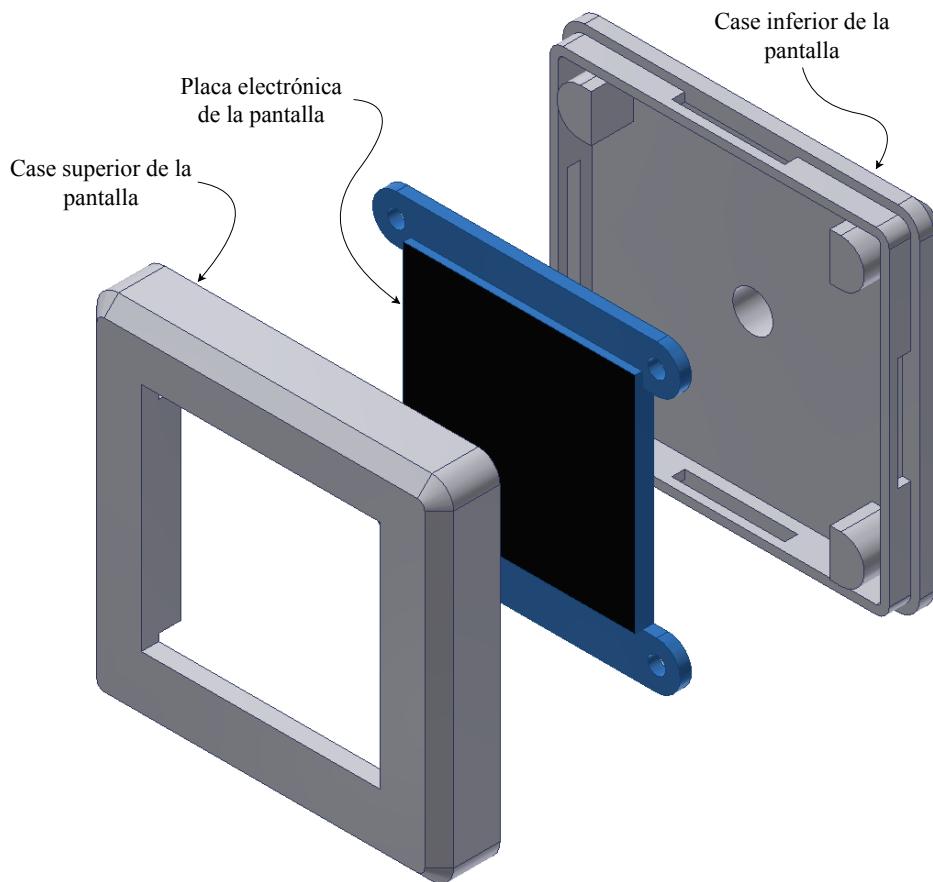


Fig. 4.36 Vista de explosión del case de la pantalla.

4.5. Diseño del software de reconocimiento de estilo de conducción

En esta sección se describe usando diagramas de flujo y pseudocódigo el funcionamiento de la parte de software del sistema de reconocimiento de estilo de conducción. El software consiste en dos partes, la primera se lleva a cabo en cada dispositivo (clientes) y la segunda en un servidor central.

Como se puede observar en la Fig. 4.37 los clientes envían la información necesaria al servidor central y este clasifica el estilo conducción de cada cliente y los envía de vuelta. Además el servidor mostrará un resumen de los datos recopilados a través de una página web. Los datos mostrados dependen de la persona que inicie sesión en la página. Los conductores podrán ver su puntaje acumulado por día y su historial, y el personal de la empresa podrá ver todos los datos recopilados (Posiciones de los vehículos, consumo, puntaje de los clientes) y resúmenes estadísticos.

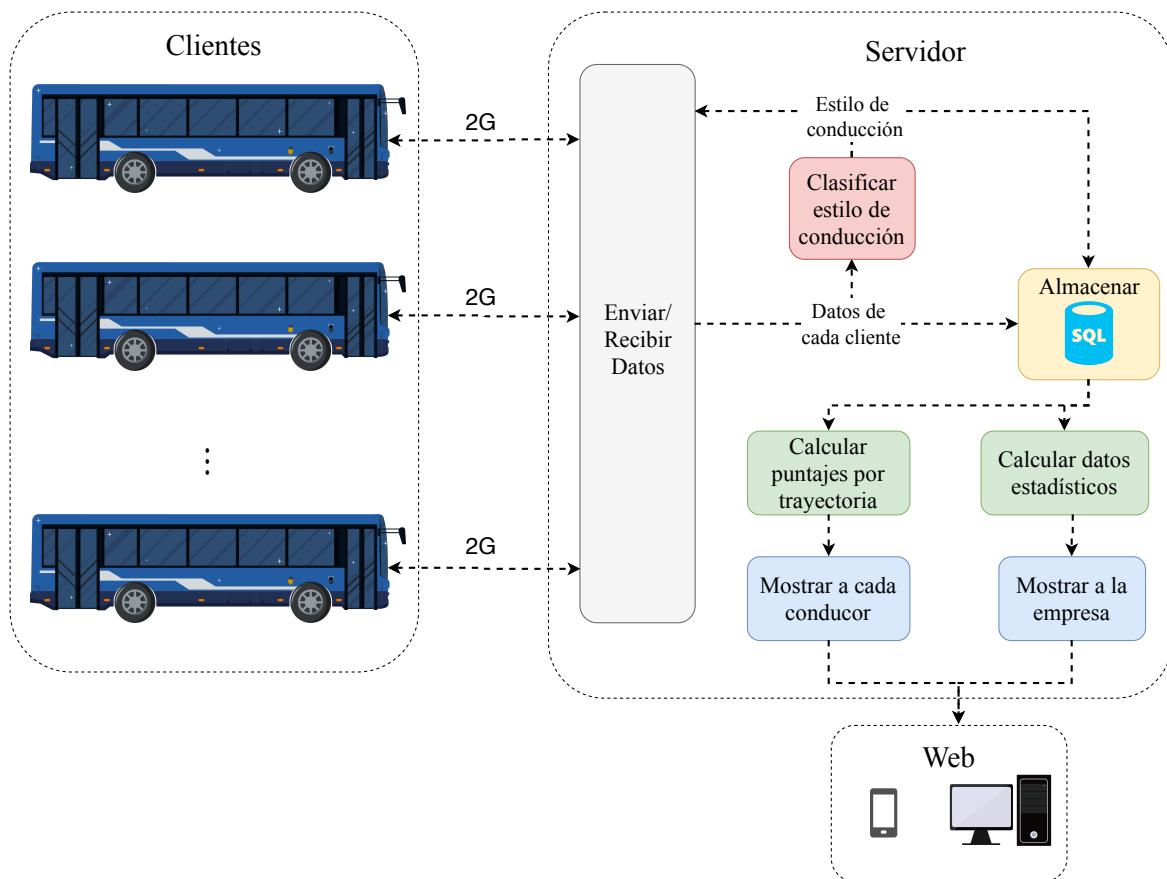


Fig. 4.37 Diagrama de bloques del software.

4.5.1. Software de los dispositivos cliente

Cada dispositivo cliente se coloca en un bus para recopilar datos y enviarlos al servidor. Su funcionamiento se puede apreciar en la Fig. 4.38. Se inicia el programa al energizar el dispositivo. Luego, este tratará de conectarse al servidor por medio de Internet. Solo al lograr tener una conexión se inicia la recopilación de datos.

El siguiente paso es recopilar los datos de los sensores (IMU, GPS y OBD2). Estos datos necesitan un pre-procesamiento antes de poder ser enviados al servidor. Este pre-procesamiento se lleva a cabo en la función "Sensor Fusion", la cual será explicada más adelante.

A continuación, se almacenan los datos y se repite el proceso durante 1 s para luego enviar todos los datos recopilados al servidor. Los datos que se van a enviar se pueden apreciar en la Tabla 4.12 junto a su tamaño en bytes (Si son representados como texto). Se obtienen estos datos a una frecuencia de 25 Hz.

Como se puede apreciar en las Ecuaciones 4.11 y 4.12, en 1 s de tiempo se acumulan 1625 bytes. Estos bytes tardan en enviarse a una velocidad de 85.6 Kbits/s a través del módulo 2G solo 0.15 s. Luego este ciclo vuelve a repetirse.

$$1625 \text{ bytes} = 65 \text{ bytes} \times 25 \text{ Hz} \times 1 \text{ s} \quad (4.11)$$

$$\frac{1625 \text{ bytes} \times 8 \text{ bits/bytes}}{85.6 \text{ Kbits/s}} = 0.76 \text{ s} \quad (4.12)$$

Tabla 4.12 Datos a enviar con su tamaño en bytes.

Datos	Tamaño en bytes
Tiempo en ms	8
Aceleración frontal	6
Aceleración lateral	6
Yaw	6
Latitud	9
Longitud	9
Velocidad	6
RPM	6
Carácteres separadores	9
Total	65

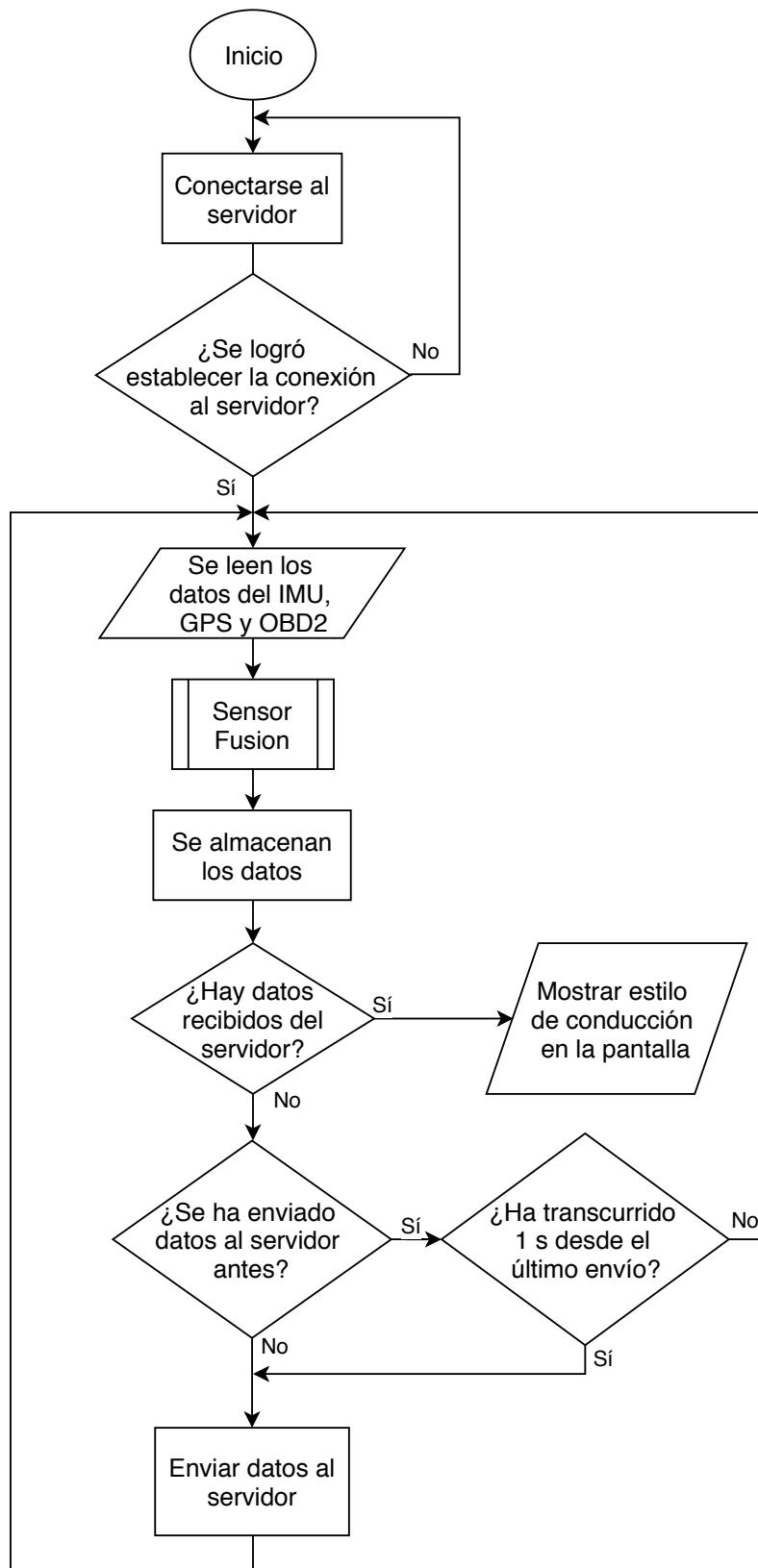


Fig. 4.38 Diagrama de flujo del software de los dispositivos cliente.

4.5.2. Software del servidor

El servidor se encarga de procesar los datos de todos los conductores. Por lo que se tienen varias instancias siendo ejecutadas concurrentemente. Cada instancia esta asociada con un vehículo y conductor. En la Fig. 4.39 se puede observar el proceso que se encarga de administrar las instancias. Luego de crear una instancia para cada vehículo, cuando se cambia de conductor se usa la instancia que ya existía del vehículo usado y se registra el nuevo conductor. Luego se ejecuta la función "Procesar Datos" que se describirá más adelante. Por último, se cierran todas las instancias luego de una hora definida por la empresa.

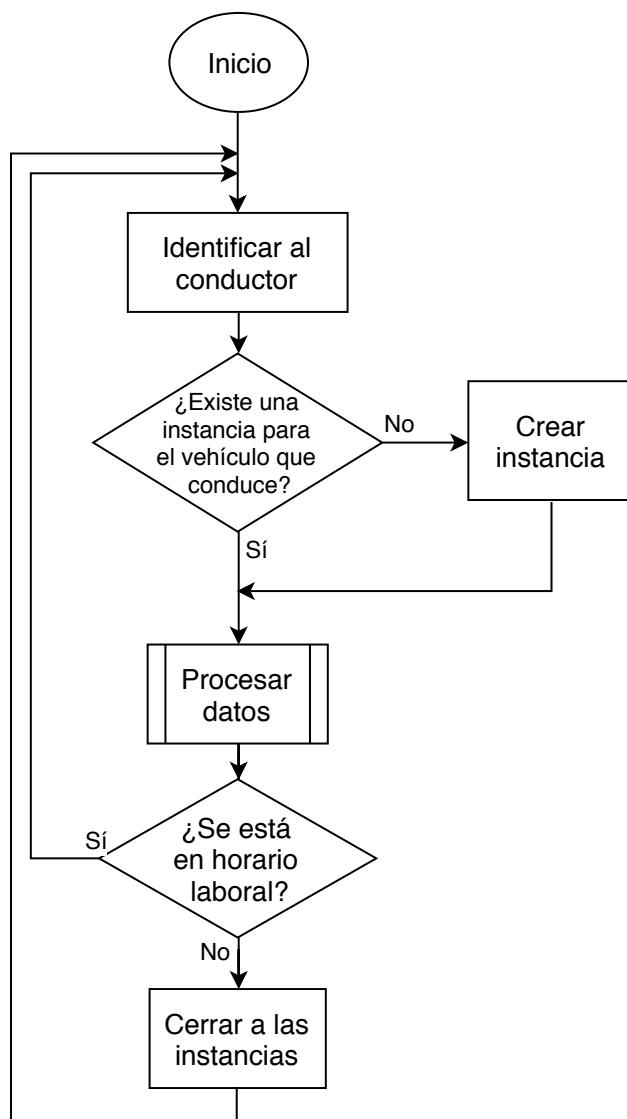


Fig. 4.39 Diagrama de flujo del software del servidor.

A. Función Procesar Datos

El diagrama de flujo de esta función se puede observar en la Fig. 4.40. El primer paso es comprobar si existen nuevos mensajes. Cuando se detecta la presencia de nuevos mensajes se almacenan estos datos y se analizan para detectar el cambio de maniobra.

Cuando se detecta el cambio de maniobra, se ejecuta la función "Clasificar maniobra", que se describirá más adelante. El resultado de esta función es la clasificación del estilo de conducción para la maniobra detectada. Este resultado se almacena y se envía al dispositivo cliente para ser mostrado al conductor.

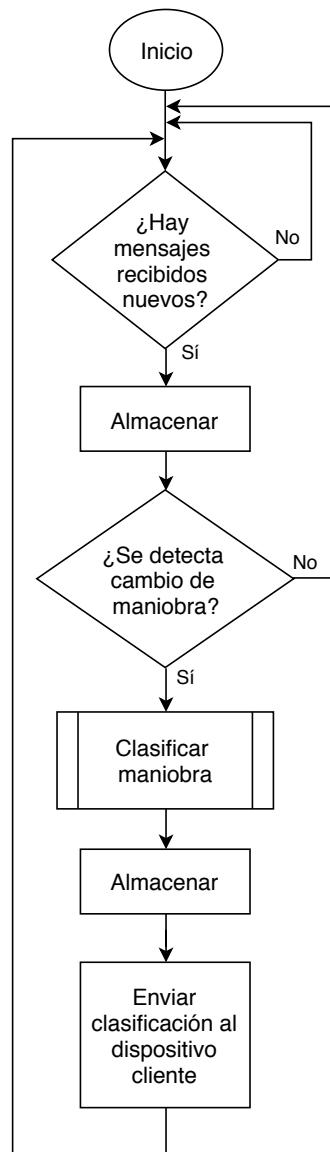


Fig. 4.40 Diagrama de flujo de la función Procesar Datos.

B. Función Clasificar maniobra

Esta función es la encargada de clasificar una maniobra según su estilo de conducción. Como se observa en la Fig. 4.41, el primer paso consiste en extraer todos las muestras de datos que pertenecen a la maniobra. Estos datos van a ser descritos usando características estadísticas para series temporales.

Una vez calculadas estas características, se obtiene por cada maniobra (de varios puntos temporales) una sola instancia . Esta instancia es ahora introducida en el modelo de clasificación que ha sido entrenado previamente. Este modelo nos entregará la clase de estilo de conducción. La etapa del desarrollo del modelo y de su entrenamiento se desarrolla en el Capítulo 5.

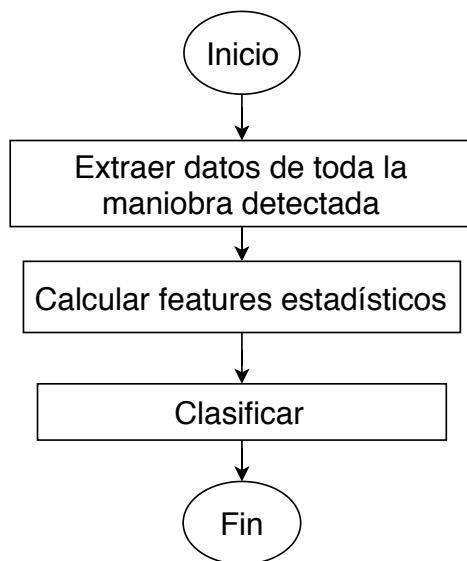


Fig. 4.41 Diagrama de flujo de la función Clasificar maniobra.

4.5.3. Software de la página web

La página web permitirá visualizar los datos que estén registrados en la base de datos. Como se ha visto en las secciones anteriores, los datos de cada vehículo y cada conductor con su respectivo puntaje y clasificación para cada maniobra son almacenados en esta base de datos. Esta página web tendrá una etapa de autenticación de usuario, en dónde el usuario iniciará sesión. Existirán dos tipos de usuarios: Conductores y Supervisores. La única diferencia es que los conductores solo podrán acceder a los datos registrados consigo mismo y los supervisores podrán acceder a todos los datos disponibles

5. Algoritmo de clasificación

En este capítulo se describirá el proceso de desarrollo del algoritmo de clasificación. Este proceso se puede apreciar en la Fig. 5.1. Además el código completo se puede encontrar en el Anexo B.

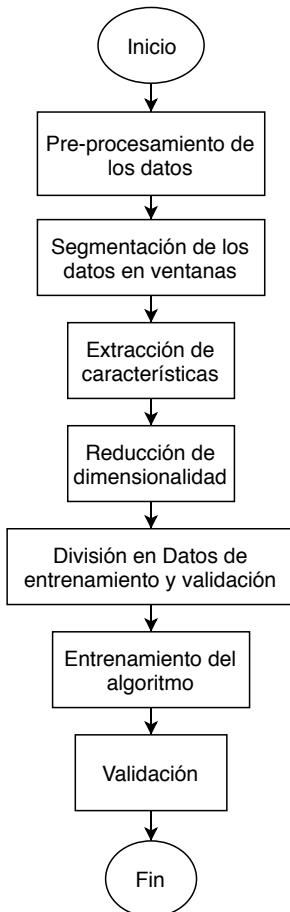


Fig. 5.1 Diagrama de flujo del desarrollo del algoritmo de clasificación.

En la Fig. 5.2 se definen algunos conceptos que se usarán a lo largo de este capítulo. Estos conceptos describen el contenido de un dataset. Cada dataset está compuesto por *instancias*. Cada instancia tiene un conjunto de *características* y es asignada a una *clase*.

Tiempo	Velocidad	Carga del motor	...	Número de pasajeros	Prisa del conductor
1.667	0.00086188	3		0	0
1.677	0.0012333	3		0	0
1.687	0.00062364	3		0	0

Fig. 5.2 Partes de un dataset.

5.0.1. Recolección de datos

Como se expresó en el concepto óptimo de solución, se usará en esta tesis un algoritmo supervisado para realizar la clasificación del estilo de manejo. Esto significa que los datos usados para entrenar y validar el algoritmo necesitan estar previamente clasificados.

Debido a la poca disponibilidad de esta clase de datos (datos de manejo clasificados como agresivos o no) se decidió utilizar un dataset disponible en Kaggle, el cuál tenía como objetivo desarrollar una estrategia inteligente para la caja de cambios. Este dataset fue generado usando el puerto OBD2 de un vehículo y un IMU para obtener data durante varios recorridos. Sin embargo, no solo se registró datos de los sensores, sino que también se registró datos categóricos de acuerdo al viaje realizado. En la Tabla 5.1 podemos observar todos los datos que contiene este dataset.

Como se puede apreciar en la Fig. 5.3, estos datos consisten en 39 viajes grabados, los cuales hacen en total 21.5 h de manejo grabadas a una frecuencia de 100 Hz. Estos datos se usarán para diseñar y probar el algoritmo de clasificación.

Como se observa en la Tabla 5.1, ninguna de estas características es la clase "agresividad al conducir" o "estilo de conducción". Por lo que se tendrá que definir a qué clase pertenecen estos datos. Para esto se usa una de las características categóricas que se tienen disponibles: "Driver's rush" o "Prisa del conductor".

Cada uno de los 39 viajes ha sido clasificado con un valor de "Prisa del conductor". Se usará este valor como indicador de la agresividad del conductor. Como cada viaje tiene distinta duración, pero mantiene en su toda su extensión el mismo valor de "Prisa

Tabla 5.1 Características o features del dataset.

Datos de sensores (series temporales)	Datos Categóricos
Tiempo (segundos)	Número de pasajeros (0 - 5)
Velocidad del vehículo (m/s)	Carga del auto (0 - 10)
Número de cambio	Aire acondicionado (0 - 10)
Carga del motor (%)	Apertura de ventanas (0 - 10)
Aceleración total (m/s ²)	Volumen del radio (0 - 10)
RPM del motor	Intensidad de lluvia (0 - 10)
Pitch	Visibilidad (0 - 10)
Aceleración Lateral (m/s ²)	Bienestar del conductor (0 - 10)
	Prisa del conductor (0 - 5)

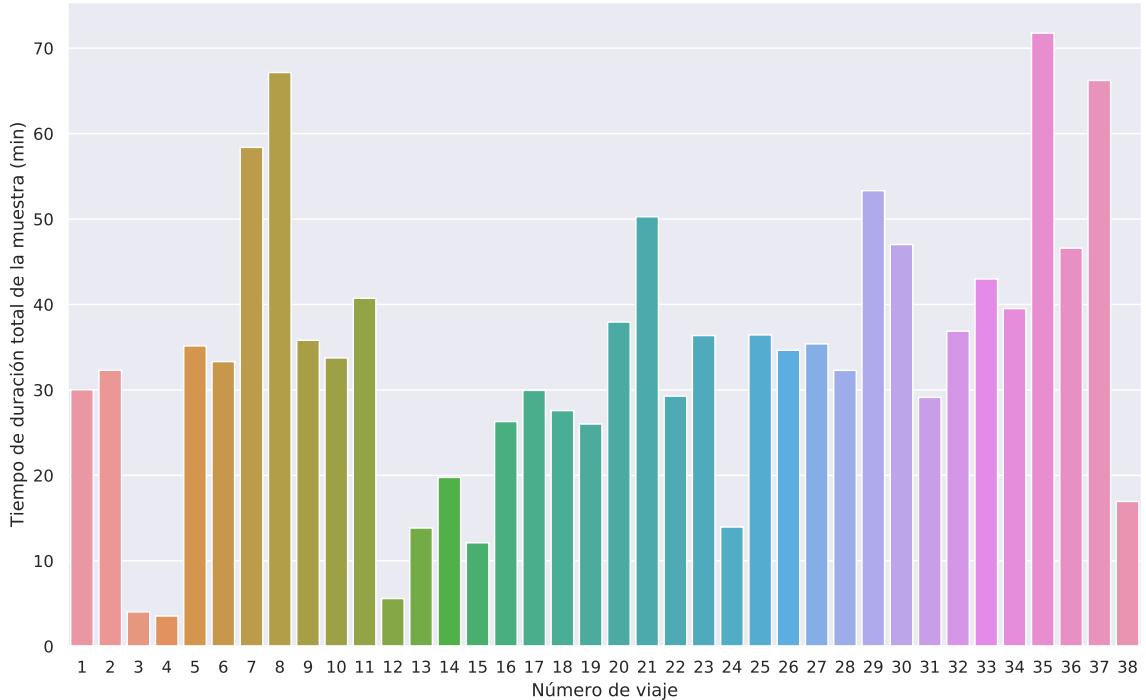


Fig. 5.3 Duración de cada viaje grabado en el dataset.

del conductor". Se analiza, en la Fig 5.4, cuantos datos pertenecerían a cada valor de esta variable.

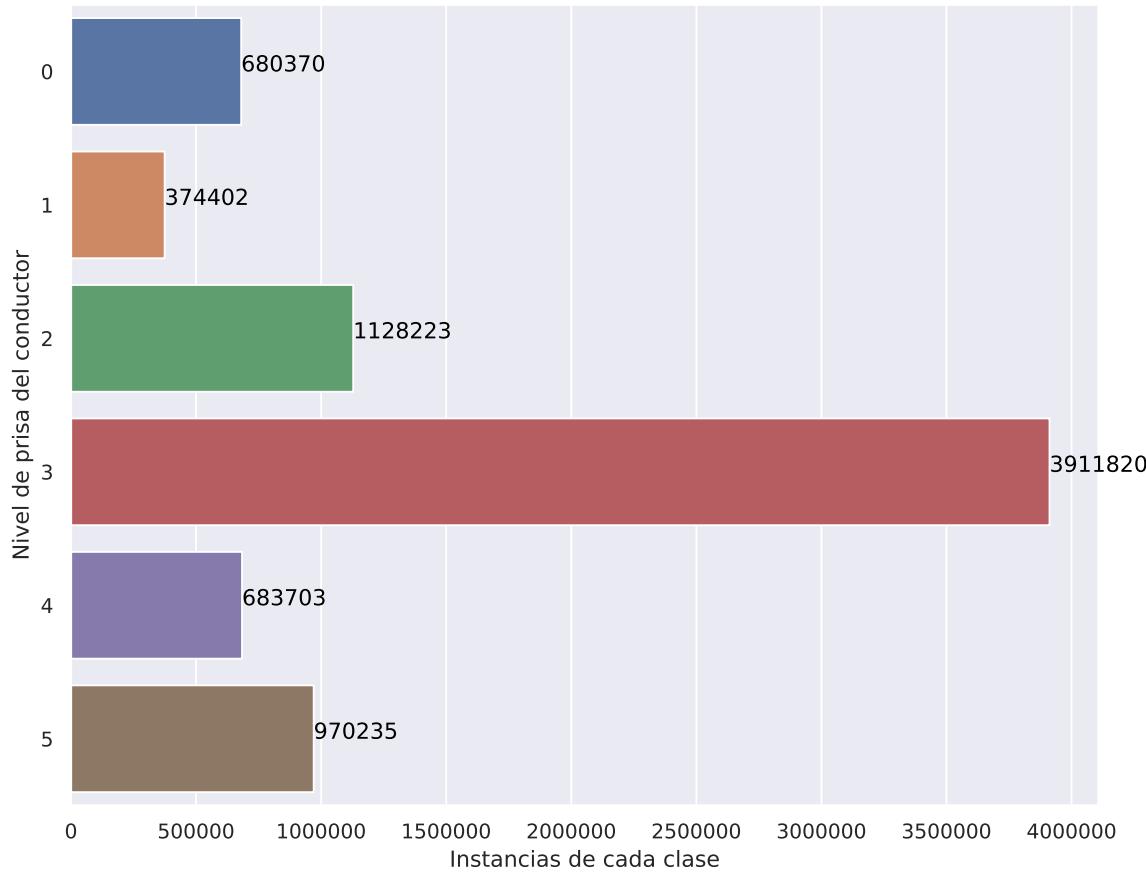


Fig. 5.4 Número de instancias para cada clase.

Debido a que la presente tesis solo quiere distinguir entre un estilo de conducción agresivo o no agresivo, se agrupan las clases definidas (de 0 a 5) en 2 grupos: el primero agrupando las clases 0, 1 y 2; y el segundo agrupando las clases 3, 4 y 5. Sin embargo, al realizar esta agrupación el número de datos disponible de cada nueva clase difiere mucho el uno de la otra. El primer nuevo grupo cuenta con 2 182 995, mientras que el segundo con 5 565 758 como se puede observar en la Fig 5.5

Si observamos la Fig. 5.4, la clase 3 es la que más instancias tiene. Por esta razón se dividirán las 6 clases definidas en el dataset en 3 nuevas clases:

- '*Calmado:*' Esta clase se compondrá de las clases 0, 1 y 2.
- '*Normal:*' Esta clase será la clase 3.
- '*Agresivo:*' Esta clase albergará a las clases 4 y 5.

Estas clases se distribuyen de una manera más uniforme como se aprecia en la Fig 5.6.

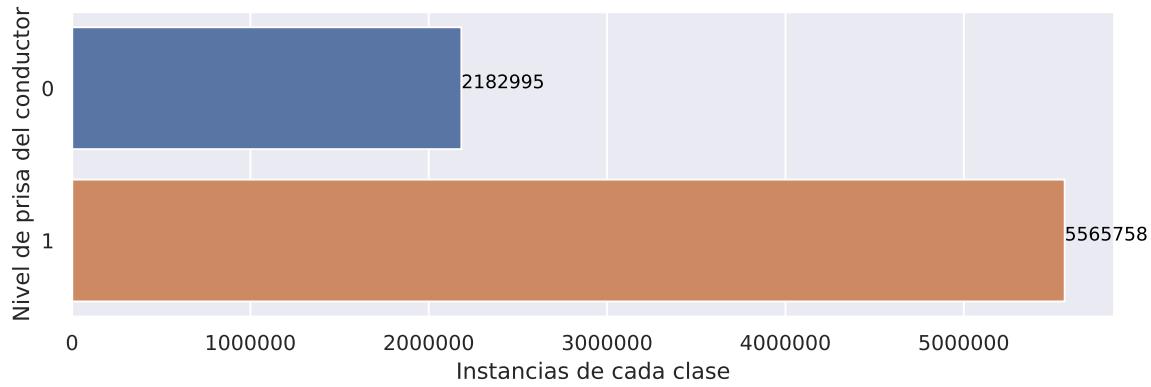


Fig. 5.5 Número de instancias para cada clase, con clase dividida en 2 grupos.

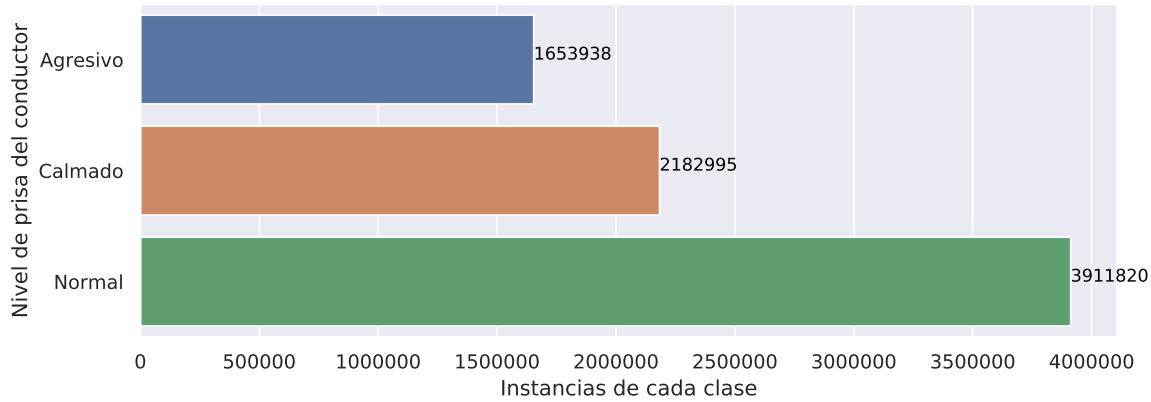


Fig. 5.6 Número de puntos para cada clase, con clase dividida en 3 grupos.

Una vez que los datos se encuentran correctamente clasificados se procede con el siguiente paso, la segmentación de los datos.

5.0.2. Segmentación de los datos

Los datos que se tienen en este momento tienen el nombre de series temporales, debido a que representan la magnitud de una característica a lo largo del tiempo. Pero esto no significa que a cada instante de tiempo le corresponde un estilo de manejo. El estilo de manejo, ya sea '*Agresivo*', '*Normal*' o '*Calmado*' no se manifiesta en un instante, sino a lo largo de una ventana de tiempo. Esta ventana puede durar varios segundos o hasta minutos.

El siguiente paso para diseñar el algoritmo es lograr segmentar los datos en estas ventanas. Algunos autores toman ventanas de tiempo fijas [15], [12], [22]; mientras que otros emplean ventanas que cambian su tamaño dinámicamente [24], [10]. En esta tesis se usará un tamaño

de ventana fijo w para la segmentación de datos. El tamaño de la ventana se determinará en la siguientes sección.

5.0.3. Extracción de características

Luego de ser segmentados en ventanas de tiempo, cada una de estas ventanas (compuestas por varios puntos temporales) se tendrá que representar en una sola instancia. Para lograr resumir este conjunto de datos de una manera adecuada, se representará cada serie temporal usando características estadísticas que pueden expresar la información contenida. Estas características estadísticas se muestran en la Tabla 5.2

Tabla 5.2 Parámetros para series de tiempo [36].

Parámetros estadísticos para series temporales

$$x_{rms} = \sqrt{\frac{1}{N} \sum_{n=1}^N x^2(n)}$$

$$S_f = \frac{x_{rms}}{\left| \frac{1}{N} \sum_{n=1}^N x(n) \right|}$$

$$C_f = \frac{x_{max}}{x_{rms}}$$

$$K_v = \frac{\frac{1}{N} \sum_{n=1}^N x^4(n)}{x_{rms}^4}$$

$$x_{p-p} = x_{max} - x_{min}$$

$$CL_f = \frac{x_{max}}{\left| \frac{1}{N} \sum_{n=1}^N \sqrt{|x(n)|} \right|^2}$$

$$I_f = \frac{x_{max}}{\left| \frac{1}{N} \sum_{n=1}^N x(n) \right|}$$

Donde:

$x(n)$ es una serie temporal para la cuál $n = 1, 2, \dots, N$ y N es el número de puntos de la serie.

x_{rms} es la raíz cuadrática media.

x_{p-p} es la amplitud pico a pico.

S_f es el factor de forma.

CL_f es el "clearence factor".

C_f es el factor de cresta.

I_f es el factor de impulso.

K_v es el valor de kurtosis.

En el dataset se tienen 5 series temporales, de cada una se obtienen las 7 características mencionados anteriormente. Esto nos deja con un dataset que tiene 35 características por cada instancia. Sin embargo, no todas estas características nos serán útiles para clasificar las instancias en las distintas clases. Es probable que muchas de estas características se encuentren correlacionadas y no aporten información significante.

Además el tener más características hace que el modelo del sistema sea más complejo y esto puede conllevar la generación de *overfitting*. El *overfitting* causa que el modelo

entrenado sea muy bueno clasificando los datos con los que ha sido entrenado, pero muy malo clasificando instancias que no ha visto anteriormente.

Debido a esta razón se necesita reducir el número de características del dataset.

5.0.4. Reducción de características

Existen distintos algoritmos que son usados para la reducción de características. Uno de los más usados es *PCA* o *Principal Component Analysis*. El funcionamiento de este algoritmo se describe en la Tabla 5.3

Tabla 5.3 Algoritmo de PCA [37].

Principal Component Analysis

1. Organizar los datos como una matriz de tamaño $m \times n$, donde m es el número de características y n es el número de instancias.
 2. Se resta a cada elemento la media de cada característica.
 3. Se calcula la matriz de covarianza.
 4. Se calculan los autovectores y autovalores de la matriz.
 5. Se reordena los autovectores y autovalores en orden decreciente según los autovalores
-

Luego de ejecutar PCA se tiene un nuevo set de características (resultantes de transformaciones de las características originales). Este set de características está ordenado de acuerdo a la varianza que cada nueva característica (componente) tiene con respecto a la clase.

Al dividir los datos usando una ventana $w = 20$ s y aplicar PCA, se obtienen los siguientes componentes o nuevas características. En la Fig. 5.7 se puede observar el valor de varianza de cada característica. Solo se muestran los 4 primeros componentes, debido a que la varianza de los demás es muy baja.

En la Fig 5.8 se observa la gráfica de cada componente. Se puede notar que los datos no se encuentran totalmente separados. Para solucionar esto se puede usar un algoritmo como *Fast ICA* o *Fast Independent Component Analysis*.

Fast ICA necesita de un *prewhitened* de los datos (un preprocesamiento, en este caso se ha realizado PCA) y busca la rotación ortogonal de los datos para lograr una separación entre

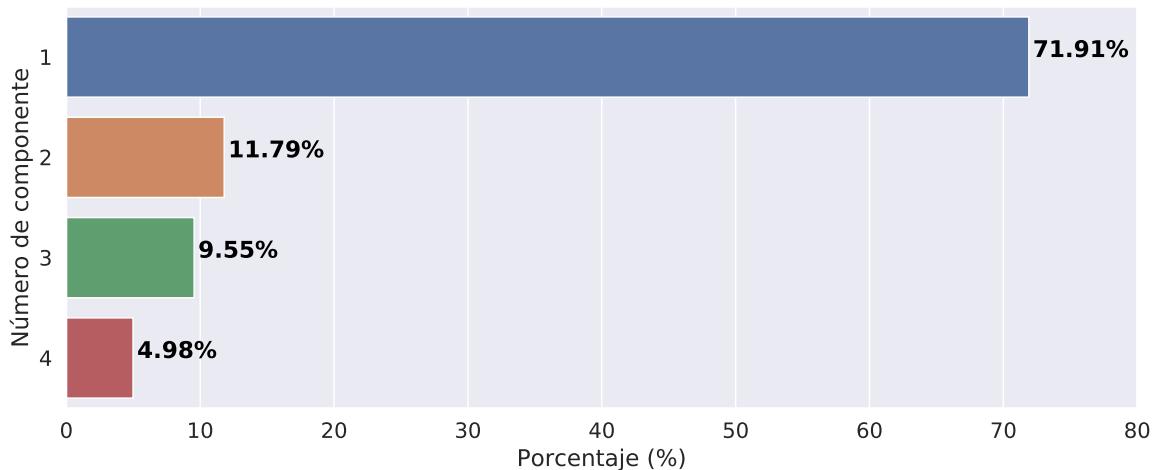


Fig. 5.7 Varianza para cada componente obtenido a partir de PCA.

clases. Luego de aplicar este algoritmo sobre los datos se obtienen 4 nuevos componentes. Se pueden observar estos componentes en la Fig 5.9

El siguiente paso para mejorar los datos es normalizar. La normalización ayuda a muchos algoritmos a tener un mejor desempeño. Los resultados de este proceso se pueden observar en la Fig 5.10.

5.0.5. Entrenamiento del algoritmo

Luego del procesamiento de los datos realizados en las secciones pasadas, es hora de entrenar el modelo de clasificación. Para poder entrenar y luego validar los resultados del modelo entrenado, se realiza una división de los datos. Se divide los datos en *datos de entrenamiento* y *datos de validación*. Se utiliza para este caso un 70% de los datos como datos de entrenamiento y un 30% como datos de validación.

Usando los datos anteriores, se entrenaron dos modelos : un modelo de *Random Forest* y un modelo de *Redes Neuronales*. Para generar los modelos se usa la librería *Scikit-learn* [38] [39]. Los parámetros de cada modelo y sus resultados se pueden observar en las Tablas 5.4 5.5

Los resultados del algoritmo de la red neuronal son mejores que los de random forest. Las redes neuronales logran clasificar correctamente todos los casos de la clase *Agresivo*. Sin embargo ambos algoritmos fallan al tratar de clasificar las clases: *Tranquilo* y *Normal*

Tabla 5.4 Parámetros de los modelos entrenados.

Algoritmo	Parámetros
Random Forest	RandomForestClassifier(n_estimators=50, max_depth=50, random_state=0)
Redes Neuronales	MLPClassifier(solver='lbfgs', alpha=1e-5, activation='identity', hidden_layer_sizes=(15,15,15), random_state=1)

Tabla 5.5 resultados de los modelos entrenados.

Random Forest		Redes Neuronales	
Clase	Precisión	Clase	Precisión
Tranquilo	0.33	Tranquilo	0.30
Normal	0.56	Normal	0.52
Agresivo	0.29	Agresivo	1.00
Promedio	0.44	Promedio	0.57

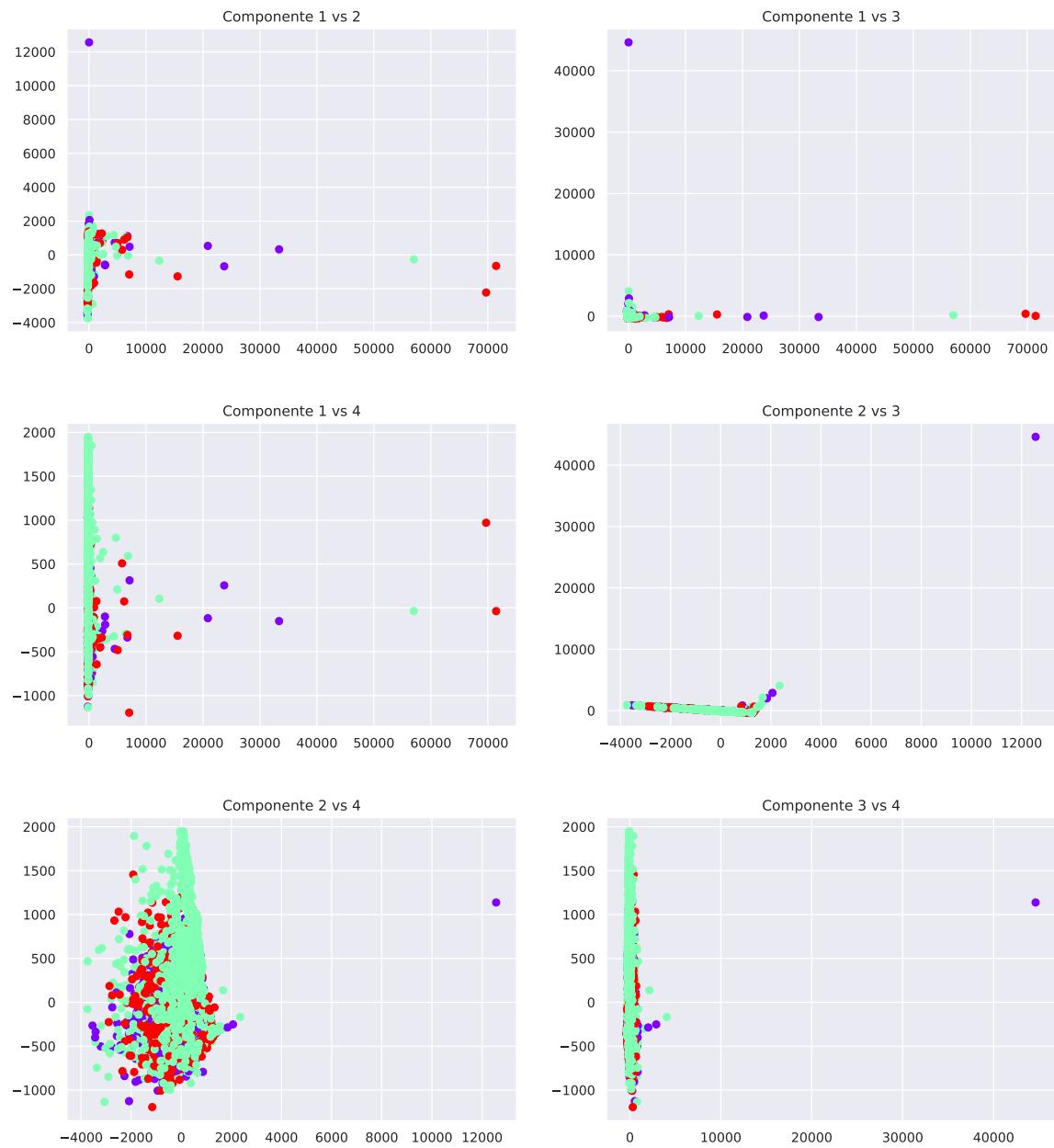


Fig. 5.8 Compontentes gráficos entre sí luego de usar.

Estos resultados se deben principalmente a la calidad de los datos que se usaron para el diseño. Uno de los factores más influyentes fue el hecho de usar la característica categórica *Prisa del conductor* como clase de estilo de manejo. Sin embargo, esto no es del todo cierto. En las recomendaciones del Capítulo 7 se pronpondrá como recomendación el desarrollo de un experimento controlado, en dónde se pueda adquirir datos de calidad correctamente clasificados.

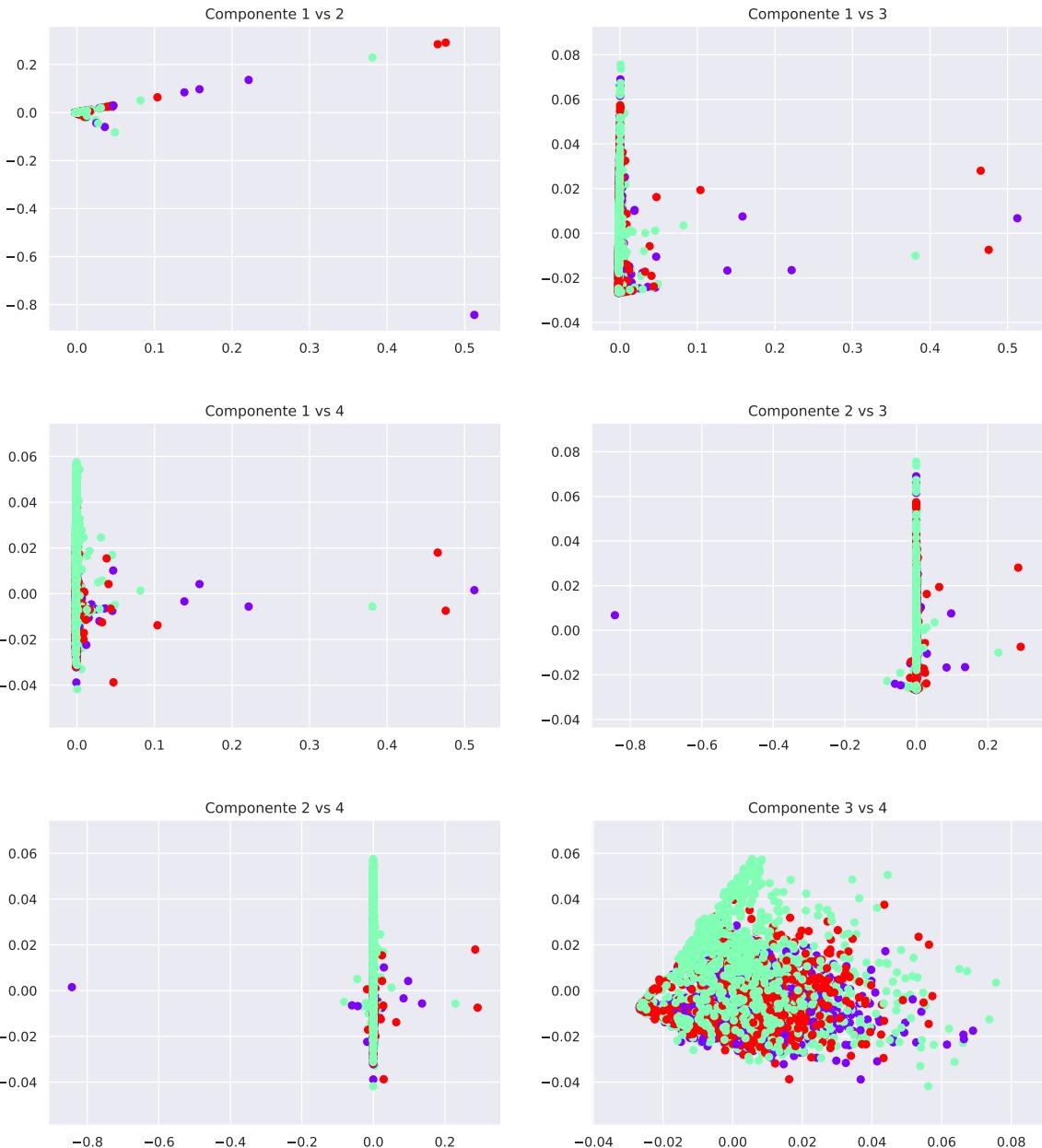


Fig. 5.9 Componentes gráficos entre sí luego de aplicar Fast ICA.

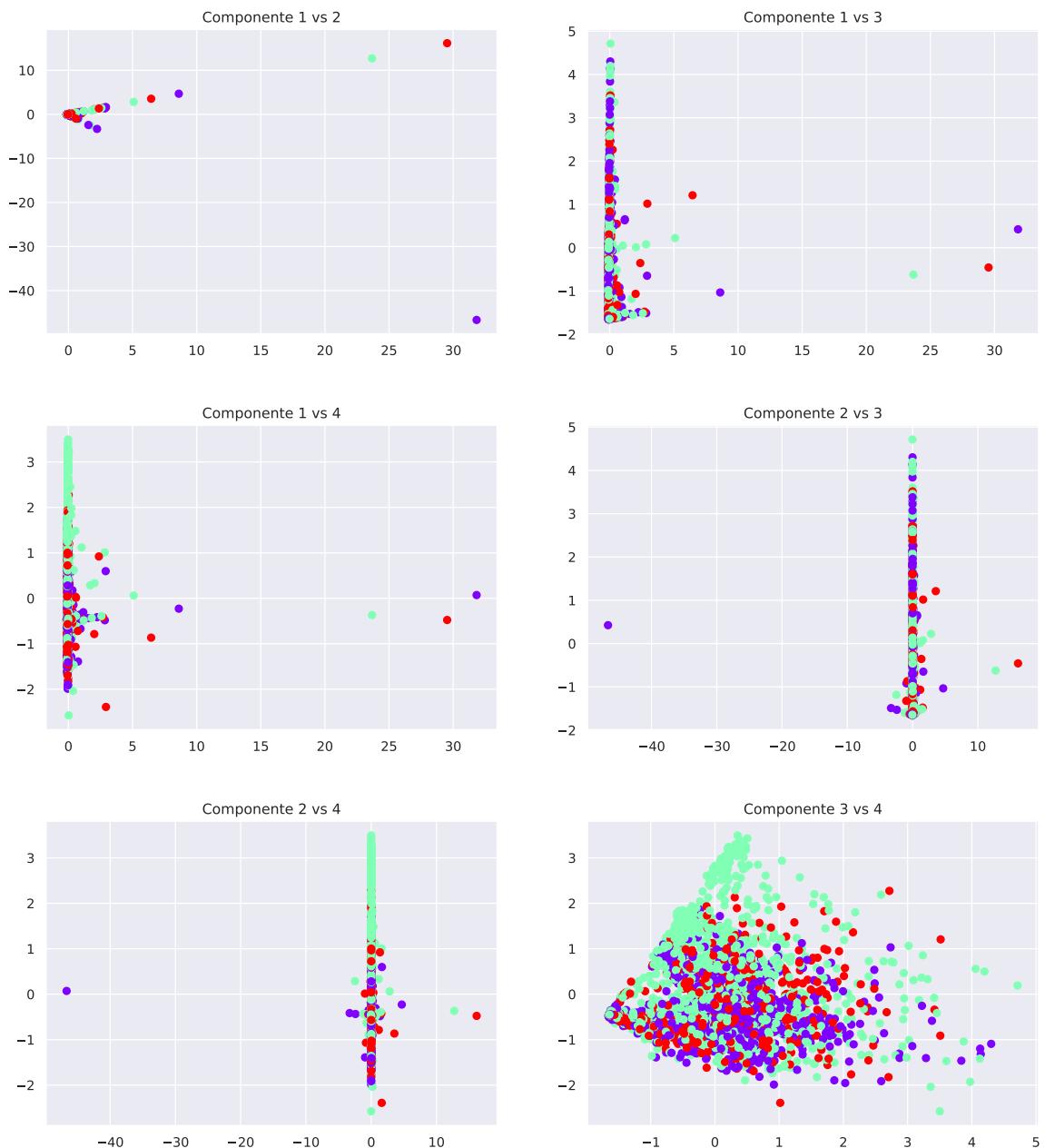


Fig. 5.10 Componentes gráficos entre sí luego de normalizar.

6. Costos del Sistema

En este capítulo se presentan los costos del sistema de reconocimiento de estilo de conducción. Se divide en costos de componentes electrónicos y componentes mecánicos (fabricación de componentes y compra de componentes). Además se considera también un costo de envío como un 40% del costo de los componentes. Este porcentaje representa los gastos de envío y aduanas, entre otros. Para los costos representados en dólares, se considera el valor del dólar como S/.3.39.

6.1. Costos de componentes electrónicos

Se puede observar en las Tablas C.1 y C.2 la lista de componentes electrónicos que conforman el sistema de reconocimiento de estilo de manejo con sus costos en dólares. Esta cotización se realizó usando DigiKey. Se muestra en la Tabla 6.1 un resumen de los costos electrónicos. Los costos en detalle y la lista de componentes se pueden encontrar en el Anexo C. El total es de \$137.75 ó S/.466.97

Tabla 6.1 Resumen de costos electrónicos

<i>Item</i>	<i>Precio (\$)</i>
Componentes electrónicos	96.39
Placa electrónica	2
Subtotal	98.39
Envío	39.36
Total	137.75

6.2. Costos de fabricación y componentes mecánicos

La Tabla 6.2 muestra los costos de fabricación del case principal y el case de la pantalla, y las piezas necesarias para el ensamblaje del dispositivo. La impresión 3D necesaria para el case se cotizó en 3d Print Perú, una empresa peruana de manufactura. El costo total de los componentes mecánicos es de \$78.92 ó S/.267.54.

Tabla 6.2 Resumen de costos mecánicos

Item	Precio unitario (\$)	Cantidad	Precio total (\$)
Case principal inferior	14.54	1	14.54
Case principal superior	22.31	1	22.31
Case superior de la pantalla	10.86	1	10.86
Case inferior de la pantalla	24.63	1	24.63
Pack de elementos Loc-Line 1/4"	3.29	2	6.58
Tornillo M2 x 3mm	0.06	3	0.18
Inserto M2 x 2.5mm	0.17	3	0.50
Total			78.92

6.3. Costo total

Se adiciona a los costos de los componentes electrónicos y mecánicos el costo de diseño. Se considera el costo de diseño como S/.50 por hora. Debido a que esta tesis se ha desarrollado en un tiempo de 15 semanas y en cada semana se ha invertido aproximadamente 15 horas, el costo total de diseño es S/.11250. El resumen de costos generales se muestra en la Tabla 6.3.

Tabla 6.3 Resumen de costos totales

Item	Precio (S/.)
Costo de componentes electrónicos	466.97
Costo de componentes mecánicos	267.54
Costo de diseño	11250.00
Total	11984.51

7. Conclusiones y Recomendaciones

En este capítulo se exponen las conclusiones y recomendaciones generadas al finalizar el presente trabajo. En resumen, en esta tesis se realizó el diseño de un sistema de clasificación de estilo de manejo para empresas de transporte. Para esto se analizaron los tipos de algoritmos más usados, se diseñó el dispositivo que recoge los datos de manejo y se diseñó y validó el algoritmo de clasificación.

El objetivo principal de la tesis fue lograr clasificar el estilo de conducción de un usuario. Como se observó en el Capítulo 5, la precisión del algoritmo tiene un margen de mejora amplio. Sin embargo, se demostró que se pueden obtener resultados aún con datos de baja calidad y que es posible clasificar el estilo de conducción de un conductor.

La clasificación de estilo de conducción es el primer paso para lograr un sistema de tránsito más seguro, ordenado y eficiente. Al lograr que las personas usen esta información para mejorar su estilo de conducción permitiría el ahorro de combustible, dinero y un aumento en la seguridad vial. En [40] se demostró que insertando un solo auto autónomo con un estilo de conducción adecuado en un ambiente con autos manejados por humanos, se duplicó la velocidad promedio de todos los autos. Esto significa que si las personas mejoran su estilo de conducción; el tráfico, uno de los grandes problemas en Lima, podría disminuir.

Además se tienen las siguientes recomendaciones para futuras investigaciones:

- Los datos a usar para entrenar el algoritmo deben ser generados por experimentos controlados con diferentes conductores, rutas y condiciones de manejo, para que se asegure la correcta interpretación de las relaciones de las características con las clases por parte del algoritmo.
- Se recomienda también, complementar el reconocimiento de estilo de conducción con tips o indicaciones para el conductor que sirvan para que puedan mejorar su estilo de conducción. Esto se puede implementar usando algoritmos como *LPP* o *learning path planning*.

Bibliografía

- [1] *ESP32 Series Datasheet*, Espressif Systems, 2018, rev. 2.5.
- [2] “The INTERNET of THINGS with ESP32,” 2018. [Online]. Available: <http://esp32.net/>
- [3] “Conductores peruanos son los peores en manejo, según estudio,” *Publimetro*, Agosto 2017. [Online]. Available: <https://publimetro.pe/actualidad/noticia-conductores-peruanos-son-peores-manejo-segun-estudio-63608>
- [4] “Global plug-in sales for Q1-2018,” 2018, Ev-volumes. [Online]. Available: <http://www.ev-volumes.com/country/total-world-plug-in-vehicle-volumes/>
- [5] T. . E. (T&E), *CO2 EMISSIONS FROM CARS: the facts*, 2018.
- [6] J. E. Meseguer, C. K. Toh, C. T. Calafate, J. C. Cano, and P. Manzoni, “Drivingstyles: a mobile platform for driving styles and fuel consumption characterization,” *Journal of Communications and Networks*, vol. 19, no. 2, pp. 162–168, April 2017.
- [7] H. Medrano Marin, “Este es el número de muertes por accidentes de tránsito en carreteras del Perú,” *El Comercio*, Febrero 2018. [Online]. Available: <https://elcomercio.pe/peru/numero-accidentes-transito-carreteras-peru-2017-noticia-499060>
- [8] J. L. Izaguirre Barrantes, “Siniestros Viales en el Perú: Reporte Estadístico PNP años 2012-2017 ,” *R2J: Sistemas Integrales Tecnológicos del Transporte*, Diciembre 2017. [Online]. Available: <http://r2jsitt.com/wp-content/uploads/2017/12/Siniestros-Viales-Perú-2012-2017-version-web-VL.pdf>
- [9] C. M. Martinez, M. Heucke, F. Wang, B. Gao, and D. Cao, “Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, March 2018.
- [10] D. Dörr, D. Grabengiesser, and F. Gauterin, “Online driving style recognition using fuzzy logic,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2014, pp. 1021–1026.
- [11] A. Aljaafreh, N. Alshabatat, and M. S. N. Al-Din, “Driving style recognition using fuzzy logic,” in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, July 2012, pp. 460–463.
- [12] Y. L. Murphrey, R. Milton, and L. Kiliaris, “Driver’s style classification using jerk analysis,” in *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, March 2009, pp. 23–28.

- [13] *Highway Capacity Manual 2000.* Washington, D.C.: Transportation Research Board, 2000.
- [14] Z. Constantinescu, C. Marinoiu, and M. Vladoiu, “Driving style analysis using data mining techniques,” *International Journal of Computers Communications & Control*, vol. 5, no. 5, pp. 654–663, 2010.
- [15] D. A. Johnson and M. M. Trivedi, “Driving style recognition using a smartphone as a sensor platform,” in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2011, pp. 1609–1615.
- [16] E. Keogh and C. A. Ratanamahatana, “Exact indexing of dynamic time warping,” *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, Mar 2005. [Online]. Available: <https://doi.org/10.1007/s10115-004-0154-9>
- [17] J. Echanobe, I. del Campo, and M. V. Martínez, “Design and optimization of a neural network-based driver recognition system by means of a multiobjective genetic algorithm,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 3745–3750.
- [18] M. V. Ly, S. Martin, and M. M. Trivedi, “Driver classification and driving style recognition using inertial sensors,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, June 2013, pp. 1040–1045.
- [19] Freescale, *High Accuracy Low g Inertial Sensor Rev. 5*, Agosto 2012.
- [20] InvenSense Inc, *MPU-6000 and MPU-6050 Product Specification Revision 3.4*, Septiembre 2013.
- [21] A. Benedetto, A. Calvi, and F. D’Amico, “Effects of mobile telephone tasks on driving performance: a driving simulator study,” *Advances in transportation studies*, vol. 26, pp. 29–44, 2012.
- [22] H. Ma, H. Xie, and D. Brown, “Eco-driving assistance system for a manual transmission bus based on machine learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 572–581, Feb 2018.
- [23] Y. Gaffary and A. Lécuyer, “The use of haptic and tactile information in the car to improve driving safety: A review of current technologies,” *Frontiers in ICT*, vol. 5, p. 5, 2018. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fict.2018.00005>
- [24] T. M. M. Van Ly Minh, Martin Sujitha, “Driver classification and driving style recognition using inertial sensors,” *IEEE Intelligent Vehicles Symposium (IV)*, 2013.
- [25] *6-Axis MEMS MotionTracking Device with Enhanced EIS Support and Sensor Fusion Processing*, TDK InvenSense, 2017, rev. 1.2.
- [26] *U-blox M8 concurrent GNSS modules Datasheet*, U-blox, 2016, rev. 03.
- [27] Claro, “Consulta nuestra cobertura 2G para personas | Cobertura Claro,” 2018. [Online]. Available: <http://cobertura.claro.com.pe/cobertura-movil-2g.php>

- [28] *SIM800L Hardware Design*, SIMCom, 2013, rev. 1.
- [29] Adafruit Industries, “Adafruit 1.54” 240x240 wide angle tft lcd display with microsd,” 2018. [Online]. Available: <https://www.adafruit.com/product/3787>
- [30] ELM Electronics, “OBD | Elm Electronics,” 2018. [Online]. Available: <https://www.elmelectronics.com/products/ics/obd/>
- [31] *ESP32-WROOM-32 Datasheet*, Espressif Systems, 2018, rev. 2.6.
- [32] *TPS56x210, 4.5-V to 17-V Input, 2-A, 3-A Synchronous Step-Down Voltage Regulator*, Texas Instruments, 2015.
- [33] *LT1764A Series 3A, Fast Transient Response, Low Noise, LDO Regulators*, Linear Technology, 2002, rev. B.
- [34] *Micro Battery*, Seiko, 2018.
- [35] *CLF7045NI-D type*, TDK, 2018.
- [36] S. Kang, D. Ma, Y. Wang, C. Lan, Q. Chen, and V. I. Mikulovich, “Method of assessing the state of a rolling bearing based on the relative compensation distance of multiple-domain features and locally linear embedding,” *Mechanical Systems and Signal Processing*, vol. 86, 03 2017. [Online]. Available: <http://gen.lib.rus.ec/scimag/index.php?s=10.1016/j.ymssp.2016.10.006>
- [37] Z. Constantinescu, C. Marinoiu, and M. Vladoiu, “Driving style analysis using data mining techniques,” *International Journal of Computers Communications & Control*, vol. 5, no. 5, pp. 654–663, 2010.
- [38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [39] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.
- [40] E. Vinitsky, A. Kreidieh, L. Le Flem, N. Kheterpal, K. Jang, F. Wu, R. Liaw, E. Liang, and A. M. Bayen, “Benchmarks for reinforcement learning in mixed-autonomy traffic,” in *Conference on Robot Learning*, 2018, pp. 399–409.

A. Datasheets

A.1. IMU



6-Axis MEMS MotionTracking™ Device with Enhanced EIS Support and Sensor Fusion Processing

GENERAL DESCRIPTION

The ICM-20648 is a 6-axis MotionTracking device that is ideally suited for Smartphones, Tablets, Wearable Sensors, and general IoT applications.

- 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP™) in a 3x3x0.9mm (24-pin QFN) package
 - Step Count, Activity Classifier, and B2S (Bring-to-See) Gesture tuned for Wrist Worn Wearable Applications
 - DMP offloads computation of motion processing algorithms from the host processor, improving system power performance
 - Software drivers are fully compliant with Google's latest Android release
 - Enhanced FSYNC functionality to improve timing for applications like EIS

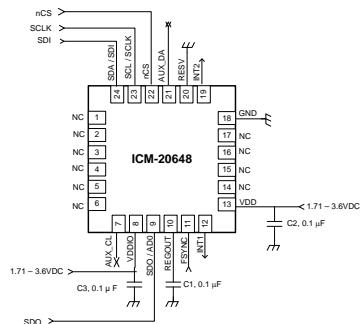
ICM-20648 supports an auxiliary I²C interface to external sensors, on-chip 16-bit ADCs, programmable digital filters, an embedded temperature sensor, and programmable interrupts. The device features an operating voltage range down to 1.71V. Communication ports include I²C and high speed SPI at 7 MHz.

ORDERING INFORMATION

PART	TEMP RANGE	PACKAGE
ICM-20648†	-40°C to +85°C	24-Pin QFN

[†]Denotes RoHS and Green-Compliant Package

TYPICAL OPERATING CIRCUIT



APPLICATIONS

- Wearable Devices
 - Smartphones and Tablets
 - IoT Applications
 - Motion-based game controllers
 - 3D remote controls for Internet connected DTVs and set top boxes, 3D mice

FEATURES

- 3-Axis Gyroscope with Programmable FSR of $\pm 250\text{dps}$, $\pm 500\text{dps}$, $\pm 1000\text{dps}$ and $\pm 2000\text{dps}$
 - 3-Axis Accelerometer with Programmable FSR of $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$
 - Onboard Digital Motion Processor (DMP)
 - Android support
 - SW features supported in DMP
 - Bring to See
 - Step Counter
 - Step Detector
 - Activity Classifier (walk, run, bike, still)
 - Calibration: Accel Bias, Compass Cal, Gyro Cal
 - Game Rotation Vector
 - Significant Motion
 - Pick up
 - Rotation Vector (with Aux compass)
 - GeoMagnetic Rotation Vector (with Aux compass)
 - Linear Acceleration
 - Gravity
 - Orientation

- Tilt
 - Auxiliary I²C interface for external sensors
 - On-Chip 16-bit ADCs and Programmable Filters
 - 7 MHz SPI or 400 kHz Fast Mode I²C
 - Digital-output temperature sensor
 - VDD operating range of 1.71V to 3.6V
 - MEMS structure hermetically sealed and bonded at wafer level
 - RoHS and Green compliant

BLOCK DIAGRAM



A.2. GPS



NEO-M8 - Data Sheet

1.3 Performance

Parameter	Specification			
	GNSS	GPS & GLONASS	GPS & BeiDou	GPS
Receiver type	72-channel u-blox M8 engine GPS L1C/A SBAS L1C/A QZSS L1C/A GLONASS L1OF BeiDou B1 Galileo E1B/C ²			
NEO-M8N/Q				
Time-To-First-Fix ³	Cold start Hot start Aided starts ⁴	26 s 1 s 2 s	27 s 1 s 3 s ⁵	29 s 1 s 2 s
Sensitivity ⁶	Tracking & Navigation Reacquisition Cold start Hot start	-167 dBm -160 dBm -148 dBm -156 dBm	-165 dBm -160 dBm -148 dBm -156 dBm	-166 dBm -160 dBm -148 dBm -156 dBm
NEO-M8M				
Time-To-First-Fix ³	Cold start Hot start Aided starts ⁴	27 s 1 s 4 s	28 s 1 s 6 s ⁵	30 s 1 s 3 s
Sensitivity ⁶	Tracking & Navigation Reacquisition Cold start Hot start	-164 dBm -159 dBm -147 dBm -156 dBm	-162 dBm -159 dBm -147 dBm -156 dBm	-163 dBm -159 dBm -147 dBm -156 dBm
TCXO or Crystal				
Max navigation update rate	NEO-M8N NEO-M8M/Q	5 Hz 10 Hz	5 Hz 10 Hz	10 Hz 18 Hz
Velocity accuracy ⁷		0.05 m/s		
Heading accuracy ⁷		0.3 degrees		
Horizontal position accuracy ⁸	Autonomous SBAS	2.5 m 2.0 m		
Accuracy of time pulse signal	RMS 99%	30 ns 60 ns		
Frequency of time pulse signal		0.25 Hz...10 MHz (configurable)		
Operational limits ⁹	Dynamics Altitude Velocity	≤ 4 g 50,000 m 500 m/s		

Table 1: NEO-M8 performance in different GNSS modes (default: concurrent reception of GPS and GLONASS)

² Ready to support Galileo E1B/C when available (NEO-M8N)³ All satellites at -130 dBm⁴ Dependent on aiding data connection speed and latency⁵ BeiDou assisted acquisition is not available with FW 2.01⁶ Demonstrated with a good external LNA⁷ 50% @ 30 m/s⁸ CEP, 50%, 24 hours static, -130 dBm, > 6 SVs⁹ Assuming Airborne < 4 g platform

A.3. GPRS/GSM



3.2. Pin Description

Table 4: Pin description

Pin name	Pin number	I/O	Description	Comment
Power supply				
VBAT	1,42	I	Power supply	
VRTC	56	I/O	Power supply for RTC	It is recommended to connect with a battery or a capacitor (e.g. 4.7uF).
VEXT	18	O	2.8V power output	If these pins are unused, keep open.
GND	2,6,8,35,37,38,39, 41,43,44,45,58,67 ,71,72,73,76,77,7 8,79,80,81,82,83, 84,85,86,87,88		Ground	GND for VBAT recommend to use 2,43,44,45pin
Power on/down				
PWRKEY	48	I	PWRKEY should be pulled low at least 1 second and then released to power on/down the module.	Internally pulled up to VBAT.
Audio interfaces				
MIC1P	52	I	Differential audio input	If these pins are unused, keep open.
MIC1N	12			
SPK1P	53	O	Differential audio output	
SPK1N	13			
MIC2P	9	I	Differential audio input	
MIC2N	10			
SPK2P	51	O	Differential audio output	
SPK2N	11			
PCM interface				
PCMCLK	29	O	PCM interface for audio	If these pins are unused, keep open.
PCMOOUT	30	O		
PCMSYNC	65	O		
PCMIN	66	I		
Keypads interface				
COL4	24	I	Support up to 50 buttons (5*5*2)	If these pins are unused, keep open. (Pin number 20 external cannot be pulled down)
COL3	21	I		
COL2	22	I		
COL1	25	I		
COL0	20	I		
ROW4	63	O		
ROW3	23	O		

B. Implementación en python del algoritmo de clasificación

1 Sistema de clasificación de estilo de manejo

1.1 1. Importando módulos

```
In [1]: import os
        import pandas as pd
        import numpy as np
        import warnings
        import matplotlib.pyplot as plt
        import matplotlib.gridspec as gridspec
        import seaborn as sns
        sns.set()
        from statsmodels.graphics.mosaicplot import mosaic
        import math
        from sklearn.manifold import LocallyLinearEmbedding
        from mpl_toolkits.mplot3d import Axes3D
```

1.2 2. Definir funciones

Se define una función para leer los 38 archivos 'csv' y otra para dividir a los archivos en ventanas de tamaño w

```

In [2]: def car_trip_filename(n):
    if (n<1) | (n>38):
        print('n out of the scope')
        return np.NAN
    route_data_kaggle='../../Datos/Car trips data log/TripData/Processed Data'
    processed_file_name='fileID'+str(n)+'_ProcessedTripData.csv'
    data = pd.read_csv(route_data_kaggle + '/' + processed_file_name, delimiter = ',',
                        header=None,
                        names=['Time(s)',
                               'Speed(m/s)',
                               'Shift_number',
                               'Engine_Load(%)',
                               'Total_Acceleration(m/s^2)',
                               'Engine RPM',
                               'Pitch',
                               'Lateral_Acceleration(m/s^2)',
                               'Passenger_count(0-5)',
                               'Load(0-10)',
                               'Air_conditioning(0-4)',
                               'Window_opening(0-10)',
                               'Radio_volume(0-10)',
                               'Rain_intensity(0-10)',
                               'Visibility',
                               'Driver_wellbeing(0-10)',
                               'Driver_rush(0-10)'])
    return data

def read_all():
    Data=[]
    for n_file in range(1,39):
        Data.append(car_trip_filename(n_file))
    return Data

#Obtiene una lista con los valores de duración en segundos de cada
def duracion_total_sec(Data):
    duration_sec=[]
    for m in range(len(Data)):
        duration_sec.append(Data[m].iloc[Data[m].shape[0]-1,0]-Data[m].iloc[0,0])
    return duration_sec

# Obtiene un diccionario en el que las 'keys' son los indices de Data
(representando el número de archivo) y
# el 'value' es una lista con la duración entre cada fila en segundos
def dict_with_step(Data):
    dict_period={}
    for m in range(len(Data)):
        time_dif=[]
        for i in range(Data[m].shape[0]):
            time_dif.append(Data[m].iloc[i,0]-Data[m].iloc[i-1,0])
        dict_period[m]=time_dif
    return dict_period

```

```

if i == 0:
    time_dif.append(Data[m].iloc[i,0])
else:
    time_dif.append(Data[m].iloc[i,0]-Data[m].iloc[i-1,0])
dict_period[m]=time_dif
return dict_period

```

Leemos la data de todos los archivos

In [3]: Data=read_all()

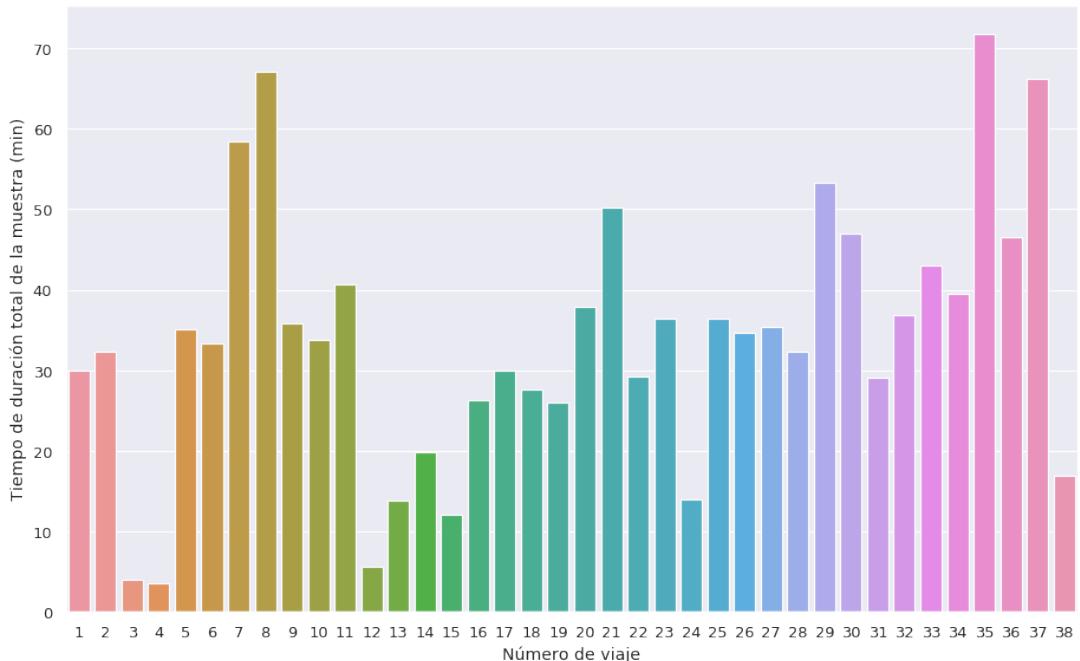
Para encontrar un tamaño de ventana adecuado, observamos las longitudes de cada archivo

In [4]: duracion = duracion_total_sec(Data)

```

dur_min=np.asarray(duracion)/60
fig1= plt.figure(figsize=(13,8),dpi=85)
sns.barplot(list(range(1,len(Data)+1)),dur_min)
plt.ylabel('Tiempo de duración total de la muestra (min)')
plt.xlabel('Número de viaje')
fig1.savefig("./Figuras/duracion_viajes.pdf", bbox_inches='tight')

```



Como podemos observar, cada archivo tiene una longitud distinta, por lo que la ventana w no debe poder exceder la longitud total de uno de estos archivos. En este caso el w_max que se puede escoger es el tamaño del archivo más pequeño, el archivo 4.

```
In [5]: w_max=Data[3].shape[0]
        print(w_max)
        w_time_aprox=w_max*0.01
        print(w_time_aprox)
```

21230
212.3

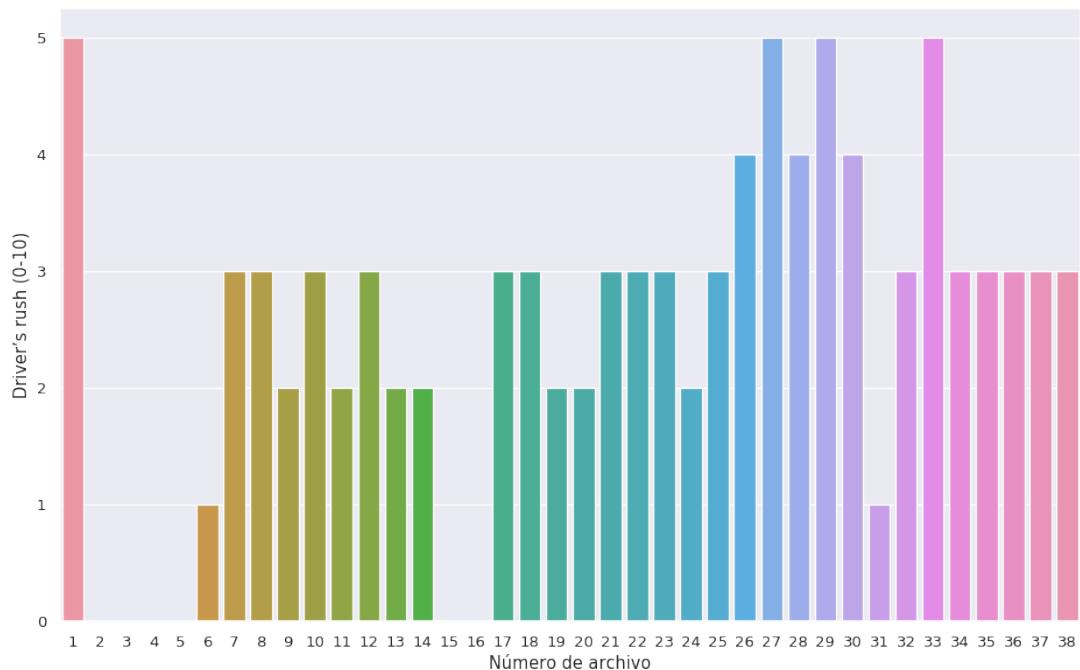
Entonces el tamaño máximo de ventana a escoger es $w_{max} = 21230$ puntos temporales. Teniendo en cuenta que entre cada punto temporal hay siempre un aproximado de 0.01 s (La data se tomo a 100 Hz). El tamaño máximo de ventana sería equivalente a 212 segundos.

1.3 3. Clases

Se considerará como clase al feature categórico: 'Driver's rush(0-10)' ya que se verá que este feature tiene un solo valor para cada archivo. Este valor representará el estilo de conducción en este conjunto de datos.

```
In [6]: fig1= plt.figure(figsize=(13,8), dpi=87)
        sns.barplot(list(range(1,39)),
                    [int(Data[m].drop_duplicates('Driver_rush(0-10)')[['Driver_rush(0-10)'].values]) for m in range(len(Data))])
        plt.ylabel('Driver's rush (0-10)')
        plt.xlabel('Número de archivo')

Out[6]: Text(0.5, 0, 'Número de archivo')
```



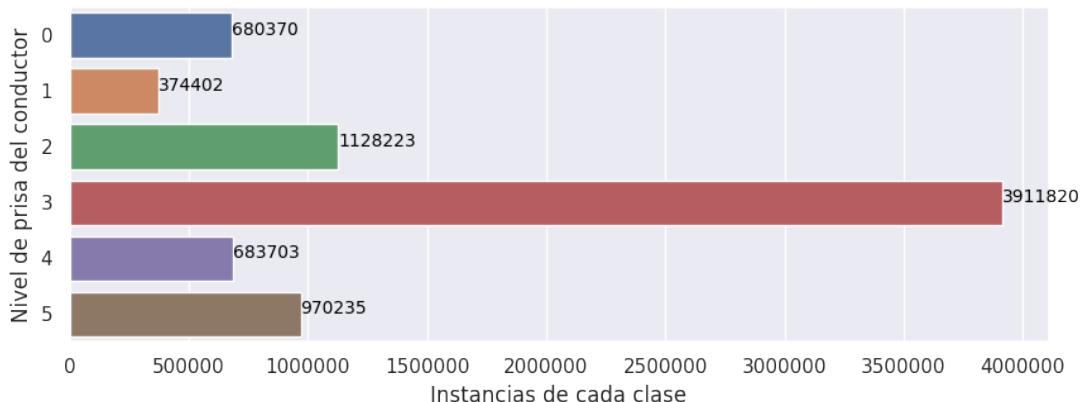
```
In [7]: clases_total=[]
    for archivo in Data:
        for dato in archivo['Driver_rush(0-10)']:
            clases_total.append(dato)

    classes, counts = np.unique(clases_total, return_counts=True)
    classes_n_dict = dict(zip(classes, counts))

    fig1= plt.figure(figsize=(10,3.5), dpi=100)
    g=sns.barplot(list(classes_n_dict.values()),list(classes_n_dict.keys()), orient='h')
    plt.ylabel('Nivel de prisa del conductor')
    plt.xlabel('Instancias de cada clase')

    for i, v in enumerate(list(classes_n_dict.values())):
        g.text(v + 3, i, str(v), color='black')

    fig1.savefig("./Figuras/instancia_clases.pdf", bbox_inches='tight')
```

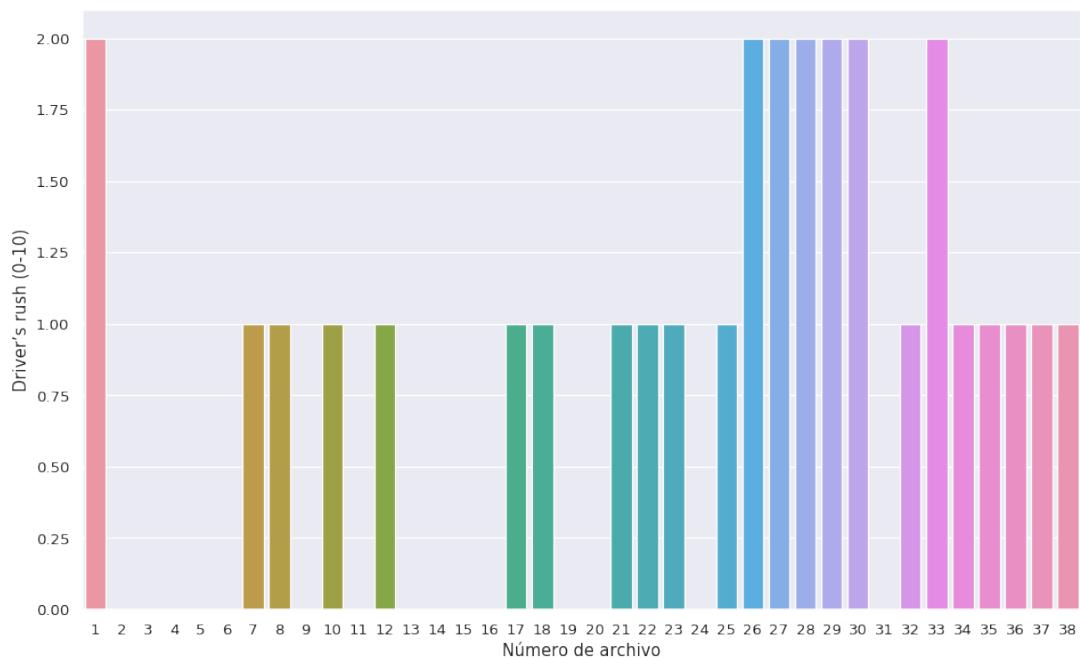


Como se observa en la gráfica, podemos encontrar valores desde 0 a 5, pero se reducirán a 3 valores para realizar la clasificación: 0 -> Tranquilo (0,1,2 en valores de Driver's Rush) 1 -> Normal (3 en valores de Driver's Rush) 2 -> Agresivo (4,5 en valores de Driver's Rush)

```
In [8]: for m in range(len(Data)):
    old = Data[m]['Driver_rush(0-10)'].values[0]
    if old<=2:
        Data[m]['Driver_rush(0-10)']='0'
    elif (old==3):
        Data[m]['Driver_rush(0-10)']='1'
    else:
        Data[m]['Driver_rush(0-10)']='2'
```

```
In [9]: fig1= plt.figure(figsize=(13,8), dpi=87)
sns.barplot(list(range(1,39)),[int(Data[m].drop_duplicates('Driver_rush(0-10)')['Driver_rush(0-10)'].values) for m in range(len(Data))])
plt.ylabel('Driver's rush (0-10)')
plt.xlabel('Número de archivo')
```

```
Out[9]: Text(0.5, 0, 'Número de archivo')
```



```
In [10]: clases_total=[]

for archivo in Data:
    for dato in archivo['Driver_rush(0-10)']:
        clases_total.append(dato)

classes, counts = np.unique(clases_total, return_counts=True)
clases=['Agresivo','Calmado','Normal']
clases_n_dict = dict(zip(classes, counts))

fig1= plt.figure(figsize=(10,3.5), dpi=100)

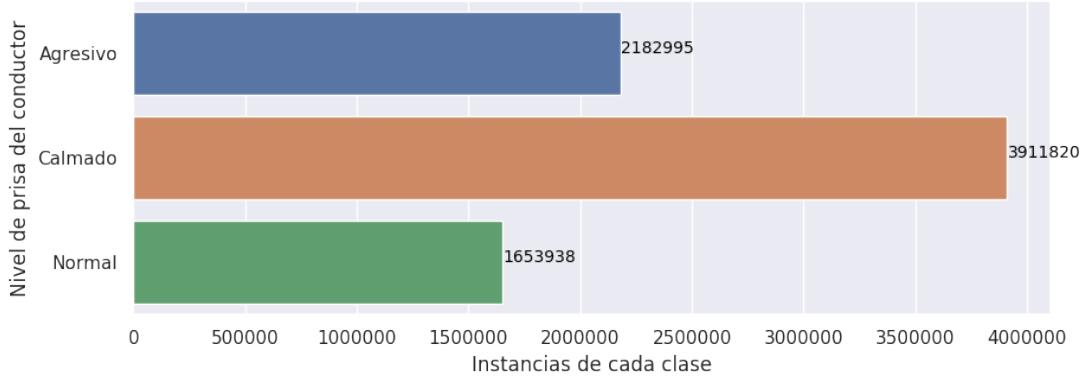
g=sns.barplot(list(clases_n_dict.values()),list(clases_n_dict.keys()), orient='h')
plt.ylabel('Nivel de prisa del conductor')
plt.xlabel('Instancias de cada clase')

for i, v in enumerate(list(clases_n_dict.values())):
```

```

g.text(v + 3, i, str(v), color='black')
fig1.savefig("./Figuras/instancia_clases.pdf", bbox_inches='tight')

```



1.4 4. Dividimos la data en ventanas de tamaño w y extraemos los features

Ahora definimos las features que vamos a extraer. Se usarán los siguientes parámetros de series de tiempo.

Table 1
The feature parameters.

Time-domain feature parameters	Frequency-domain feature parameters
$x_{rms} = \sqrt{\frac{1}{N} \sum_{n=1}^N x^2(n)}$	$F_C = \frac{\sum_{n=1}^N x(n)x(n)}{2\pi \sum_{n=1}^N x(n)^2}$
$x_{p-p} = x_{max} - x_{min}$	$F_{MS} = \frac{\sum_{n=1}^N x(n)^2}{4\pi^2 \sum_{n=1}^N x(n)^2}$
$S_f = \frac{x_{rms}}{\left \frac{1}{N} \sum_{n=1}^N x(n) \right }$	$F_{RM} = \sqrt{F_{MS}}$
$C_f = \frac{x_{max}}{x_{rms}}$	$F_V = F_{MS} - (F_C)^2$
$I_f = \frac{x_{max}}{\left \frac{1}{N} \sum_{n=1}^N x(n) \right }$	$p_1 = \frac{\sum_{k=1}^H s(k)}{H}$
$CL_f = \frac{x_{max}}{\left \frac{1}{N} \sum_{n=1}^N \sqrt{ x(n) } \right ^2}$	$p_2 = \frac{\sum_{k=1}^H (s(k) - p_1)^2}{H-1}$
$K_v = \frac{\frac{1}{N} \sum_{n=1}^N x^4(n)}{x_{rms}^4}$	$p_3 = \frac{\sum_{k=1}^H (s(k) - p_1)^3}{H(\sqrt{p_2})^3}$
where $x(n)$ is a signal series for $n = 1, 2, \dots, N$, N is the number of data points.	$p_4 = \frac{\sum_{k=1}^H (s(k) - p_1)^4}{H(p_2)^2}$
	where $s(k)$ is a spectrum for $k = 1, 2, \dots, H$, H is the number of spectrum lines; f_k is the frequency value of the k th spectrum line.
	$p_5 = \frac{\sum_{k=1}^H f_k s(k)}{\sum_{k=1}^H s(k)}$
	$p_6 = \sqrt{\frac{\sum_{k=1}^H (f_k - p_5)^2 s(k)}{H}}$
	$p_7 = \sqrt{\frac{\sum_{k=1}^H f_k^2 s(k)}{\sum_{k=1}^H s(k)}}$
	$p_8 = \sqrt{\frac{\sum_{k=1}^H f_k^2 s(k)}{\sum_{k=1}^H f_k^2 s(k)}}$
	$p_9 = \frac{\sum_{k=1}^H f_k^2 s(k)}{\sqrt{\sum_{k=1}^H s(k) \sum_{k=1}^H f_k^4 s(k)}}$
	$p_{10} = \frac{p_6}{p_5}$
	$p_{11} = \frac{\sum_{k=1}^H (f_k - p_5)^3 s(k)}{H p_6^3}$
	$p_{12} = \frac{\sum_{k=1}^H (f_k - p_5)^4 s(k)}{H p_6^4}$

Entonces ahora definimos los features en el dominio del tiempo que hallaremos para cada serie temporal

```

In [11]: #En todas las funciones x es un array que representa un time-serie
def get_x_rms(x):
    N=x.shape[0]
    return np.power((np.sum(np.power(x, 2)))/N, 0.5)

```

```

def get_x_p_p(x):
    return x.max()-x.min()

def get_S_f(x):
    N=x.shape[0]
    return np.power((np.sum(np.power(x, 2)))/N, 0.5)/abs(np.sum(x)/N)

def get_C_f(x):
    N=x.shape[0]
    return x.max()/np.power((np.sum(np.power(x, 2)))/N, 0.5)

def get_I_f(x):
    N=x.shape[0]
    return x.max()/abs(np.sum(x)/N)

def get_CL_f(x):
    N=x.shape[0]
    return x.max()/np.power(np.sum(np.power(abs(x),0.5))/N,2)

def get_K_v(x):
    N=x.shape[0]
    return np.sum(np.power(x,4))/(np.power((np.sum(np.power(x, 2)))/N, 2)*N)

# df: Es el dataframe cuyas columnas son los time-series de los que se quiere
# extraer los features
def get_features(df,col_prefix=None):
    col_sufix=['x_rms','x_p_p','S_f','C_f','I_f','CL_f','K_v']

    x_rms_array=[]
    x_p_p_array=[]
    S_f_array=[]
    C_f_array=[]
    I_f_array=[]
    CL_f_array=[]
    K_v_array=[]

    for col in range(df.shape[1]):
        x_rms_array.append(get_x_rms(df.iloc[:,col].values))
        x_p_p_array.append(get_x_p_p(df.iloc[:,col].values))
        S_f_array.append(get_S_f(df.iloc[:,col].values))
        C_f_array.append(get_C_f(df.iloc[:,col].values))
        I_f_array.append(get_I_f(df.iloc[:,col].values))
        CL_f_array.append(get_CL_f(df.iloc[:,col].values))
        K_v_array.append(get_K_v(df.iloc[:,col].values))

    features=[x_rms_array,x_p_p_array,S_f_array,C_f_array,

```

```

I_f_array,CL_f_array,K_v_array]

if col_prefix==None:
    col_prefix=df.columns

array_out=[]
indexes=[]
for col in range(df.shape[1]):
    for feat in range(7):
        indexes.append(col_prefix[col] + '_' + col_sufix[feat])
        array_out.append(features[feat][col])

series_out=pd.Series(array_out,indexes)
return series_out

def concat_array_df(array_of_df):
    df_out=array_of_df[0].copy()
    for i in range(1,len(array_of_df)):
        df_out=pd.concat([df_out,array_of_df[i]],ignore_index=True)
    return df_out

# w: tamaño de ventana (número de filas)
# Data: Lista en la que cada elemento es un Dataframe que representa a un archivo
# columns: (Opcional) lista con los labels de las columnas que se quieren tener.
# La última columna se asume como clase
def get_df_features_w(w,Data,columns=None):
    X_array=[]
    Y_array=[]
    if columns == None:
        columns=Data[0].columns
    for m in range(len(Data)):
        row_index=0

        if Data[m].shape[0]>=w:
            rows_of_windows=math.floor(Data[m].shape[0]/w)

            for n in range(rows_of_windows):
                Xdf_temp=Data[m][columns].iloc[row_index:row_index+w,:-1]
                #Extraemos los features y ahora tenemos una fila por cada ventana w
                Xdf_temp=get_features(Xdf_temp)
                X_array.append(Xdf_temp)

                Ydf_temp=Data[m].iloc[row_index,-1] #Se toma solo el primer elemento
                #porque la clase se repite y es la misma para cada archivo
                Y_array.append(Ydf_temp)
                row_index+=w
        else:
            print('Tamaño de ventana w muy grande')

```

```

X_df=pd.DataFrame(X_array)
Y_df=pd.DataFrame({'Rush':Y_array})
return X_df,Y_df

In [12]: X_df,Y_df=get_df_features_w(w_max,Data,['Speed(m/s)', 'Total_Acceleration(m/s^2)',
'Lateral_Acceleration(m/s^2)'
,'Engine RPM','Pitch','Driver_rush(0-10)'])

Y_array=np.array([int(Y_df.values[i]) for i in range(Y_df.values.shape[0])])
print(X_df.shape)
print(Y_df.shape)

(349, 35)
(349, 1)

```

Cómo se puede observar al elegir el tamaño de ventana máximo con \$ w=2000 \$ (de aproximadamente 20 s), se obtienen 3855 ventanas. Además se extrajo los features de cada serie temporal (cada columna). Se extrajeron 7 features y existían 5 columnas, por lo que se obtuvieron 35 columnas en el X_{df}

```

In [13]: X_df,Y_df=get_df_features_w(2000,Data,['Speed(m/s)', 'Total_Acceleration(m/s^2)',
'Lateral_Acceleration(m/s^2)'
,'Engine RPM','Pitch','Driver_rush(0-10)'])

Y_array=np.array([int(Y_df.values[i]) for i in range(Y_df.values.shape[0])])
print(X_df.shape)
print(Y_df.shape)

(3855, 35)
(3855, 1)

```

1.5 5. Selección de parámetros

Ahora seleccionaremos sólo los parámetros que sean relevantes para la clasificación. Para eso utilizaremos el algoritmo PCA para reducir a 4 componentes.

1.6 PCA

```

In [14]: from sklearn.decomposition import PCA
n_comp=4
PCA_obj = PCA(n_components = n_comp)
X_reduct = PCA_obj.fit_transform(X_df)
print(X_reduct.shape)
sns.set(font_scale=1.3)

variance=PCA_obj.explained_variance_ratio_
variance=[variance[i]*100 for i in range(variance.shape[0])]

```

```

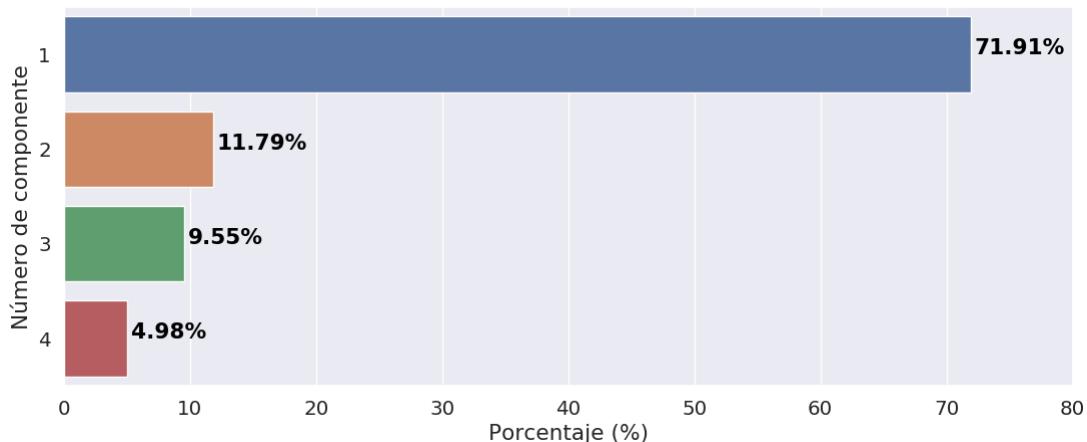
fig1= plt.figure(figsize=(13,5), dpi=100)
graph=sns.barplot(variance, list(range(1,n_comp+1)), orient='h')
plt.ylabel('Número de componente')
plt.xlabel('Porcentaje (%)')
plt.xlim([0, 80])

for i, v in enumerate(variance):
    graph.text(v +0.3, i , str(round(v,2))+'%', color='black', fontweight='bold')

fig1.savefig("./Figuras/PCA_dist.pdf", bbox_inches='tight')

```

(3855, 4)



```

In [15]: fig, axes = plt.subplots(3, 2, figsize=(15,17))
axes[0, 0].scatter(X_reduct[:,0], X_reduct[:,1], c=Y_array, cmap=plt.cm.rainbow)
axes[0, 0].set_title('Componente 1 vs 2')
# plt.ylabel('Componente 2')
# plt.xlabel('Componente 1')

axes[0, 1].scatter(X_reduct[:,0], X_reduct[:,2], c=Y_array, cmap=plt.cm.rainbow)
axes[0, 1].set_title('Componente 1 vs 3')
# plt.ylabel('Componente 3')
# plt.xlabel('Componente 1')

axes[1, 0].scatter(X_reduct[:,0], X_reduct[:,3], c=Y_array, cmap=plt.cm.rainbow)
axes[1, 0].set_title('Componente 1 vs 4')
# plt.ylabel('Componente 4')
# plt.xlabel('Componente 1')

axes[1, 1].scatter(X_reduct[:,1], X_reduct[:,2], c=Y_array, cmap=plt.cm.rainbow)

```

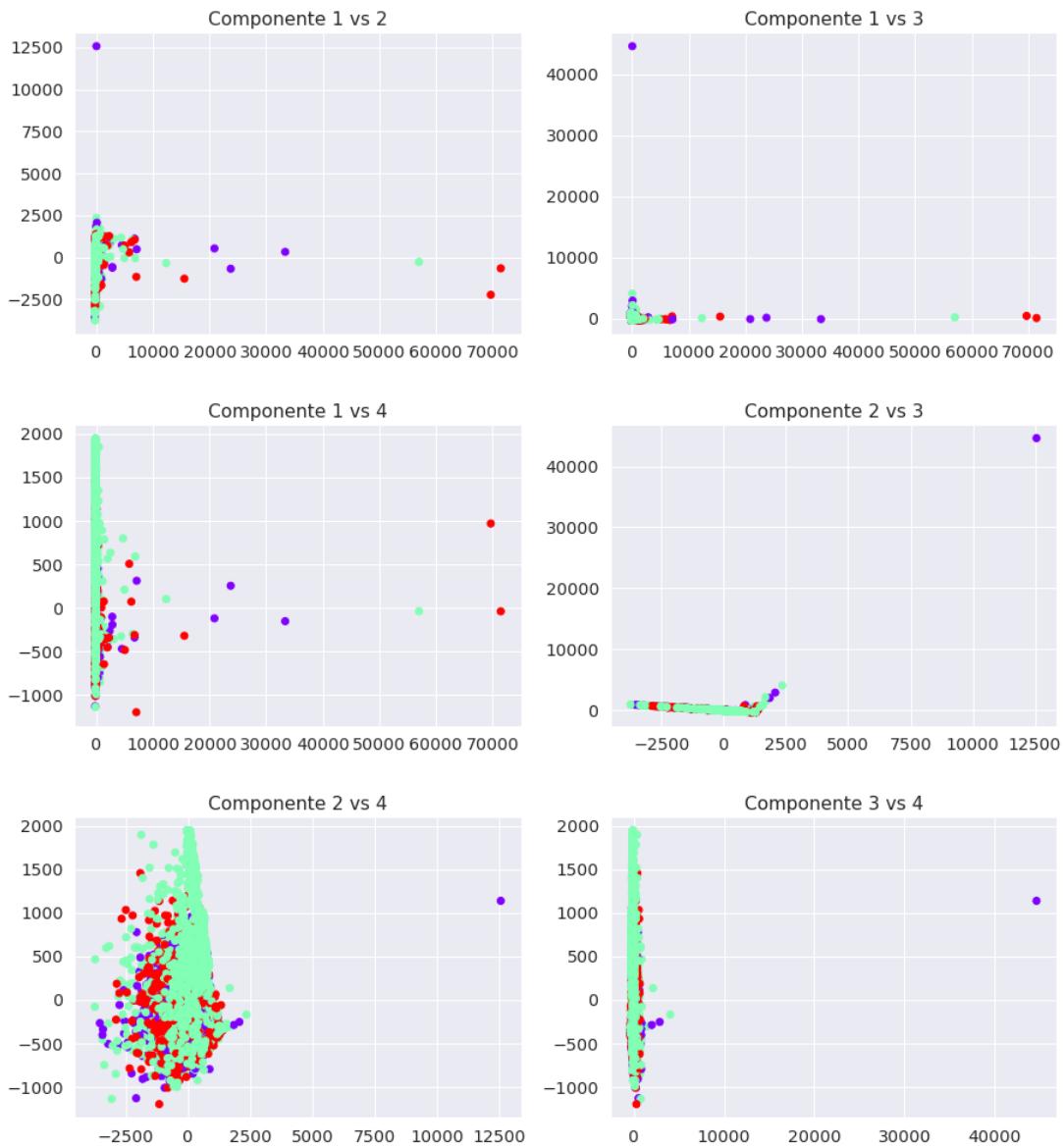
```
axes[1, 1].set_title('Componente 2 vs 3')
# plt.ylabel('Componente 3')
# plt.xlabel('Componente 2')

axes[2, 0].scatter(X_reduct[:,1], X_reduct[:,3], c=Y_array, cmap=plt.cm.rainbow)
axes[2, 0].set_title('Componente 2 vs 4')
# plt.ylabel('Componente 4')
# plt.xlabel('Componente 2')

axes[2, 1].scatter(X_reduct[:,2], X_reduct[:,3], c=Y_array, cmap=plt.cm.rainbow)
axes[2, 1].set_title('Componente 3 vs 4')
# plt.ylabel('Componente 4')
# plt.xlabel('Componente 3')

plt.subplots_adjust(hspace=0.3)

fig.savefig("./Figuras/PCA_vs.pdf", bbox_inches='tight')
```



Se usa FastICA para intentar separar los componentes seleccionados

```
In [16]: from sklearn.decomposition import FastICA
n_comp=4
FastICA_tr = FastICA(n_components=n_comp, random_state=0, algorithm='deflation')
X_reduct3 = FastICA_tr.fit_transform(X_reduct)
X_reduct3.shape
```

```
Out[16]: (3855, 4)
```

```
In [17]: fig, axes = plt.subplots(3, 2, figsize=(15,17))
axes[0, 0].scatter(X_reduct3[:,0], X_reduct3[:,1], c=Y_array, cmap=plt.cm.rainbow)
```

```

axes[0, 0].set_title('Componente 1 vs 2')
# plt.ylabel('Componente 2')
# plt.xlabel('Componente 1')

axes[0, 1].scatter(X_reduct3[:,0], X_reduct3[:,2], c=Y_array, cmap=plt.cm.rainbow)
axes[0, 1].set_title('Componente 1 vs 3')
# plt.ylabel('Componente 3')
# plt.xlabel('Componente 1')

axes[1, 0].scatter(X_reduct3[:,0], X_reduct3[:,3], c=Y_array, cmap=plt.cm.rainbow)
axes[1, 0].set_title('Componente 1 vs 4')
# plt.ylabel('Componente 4')
# plt.xlabel('Componente 1')

axes[1, 1].scatter(X_reduct3[:,1], X_reduct3[:,2], c=Y_array, cmap=plt.cm.rainbow)
axes[1, 1].set_title('Componente 2 vs 3')
# plt.ylabel('Componente 3')
# plt.xlabel('Componente 2')

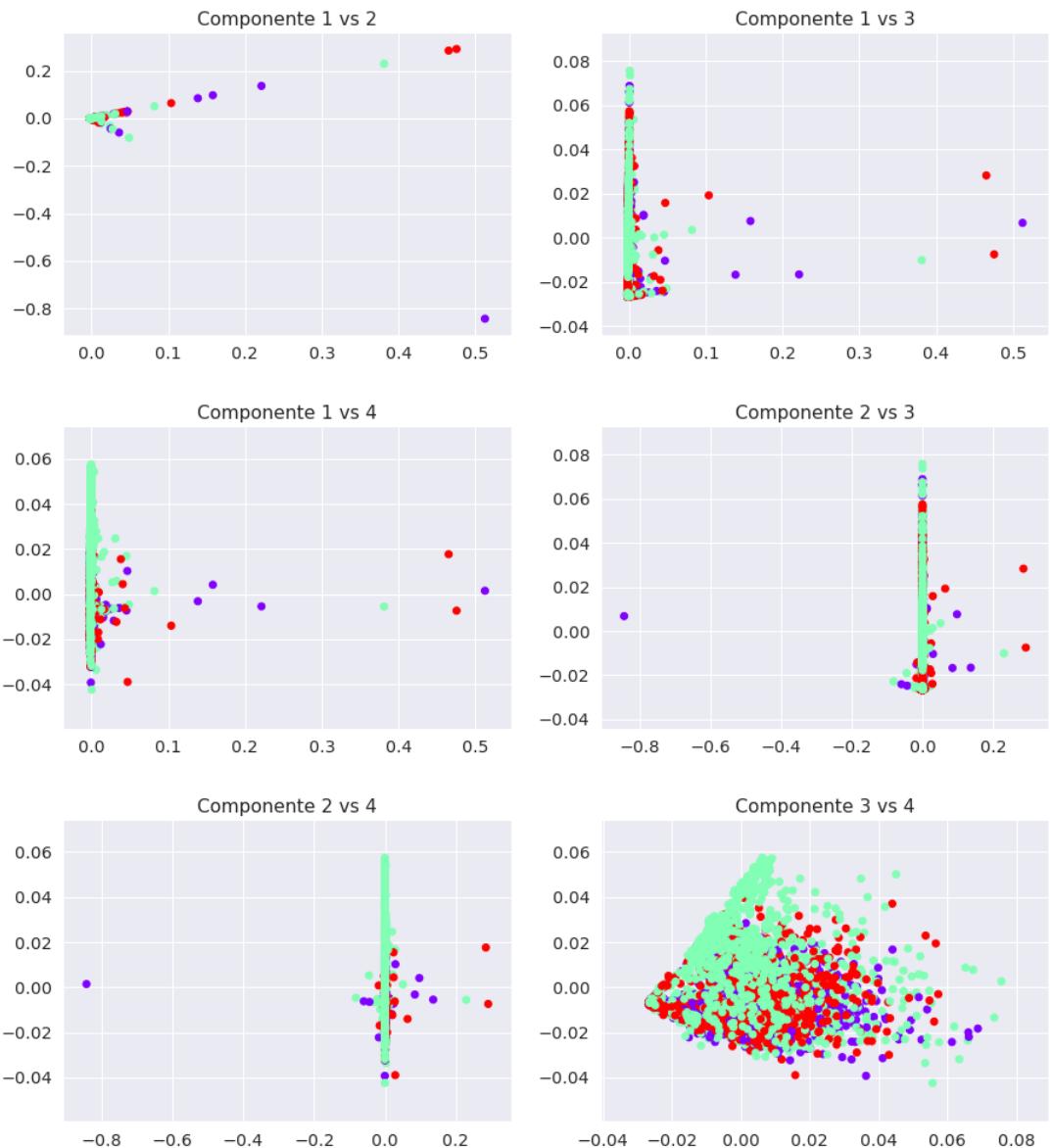
axes[2, 0].scatter(X_reduct3[:,1], X_reduct3[:,3], c=Y_array, cmap=plt.cm.rainbow)
axes[2, 0].set_title('Componente 2 vs 4')
# plt.ylabel('Componente 4')
# plt.xlabel('Componente 2')

axes[2, 1].scatter(X_reduct3[:,2], X_reduct3[:,3], c=Y_array, cmap=plt.cm.rainbow)
axes[2, 1].set_title('Componente 3 vs 4')
# plt.ylabel('Componente 4')
# plt.xlabel('Componente 3')

plt.subplots_adjust(hspace=0.3)

fig.savefig("./Figuras/PCA_vs_separados.pdf", bbox_inches='tight')

```



1.7 Model training

1.7.1 Splitting into train and test datasets

```
In [18]: X_data=X_reduct3
         test_size=0.3

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_array,
                                                    test_size=test_size, random_state=45, shuffle=True)
X_train.shape
```

```
Out[18]: (2698, 4)
```

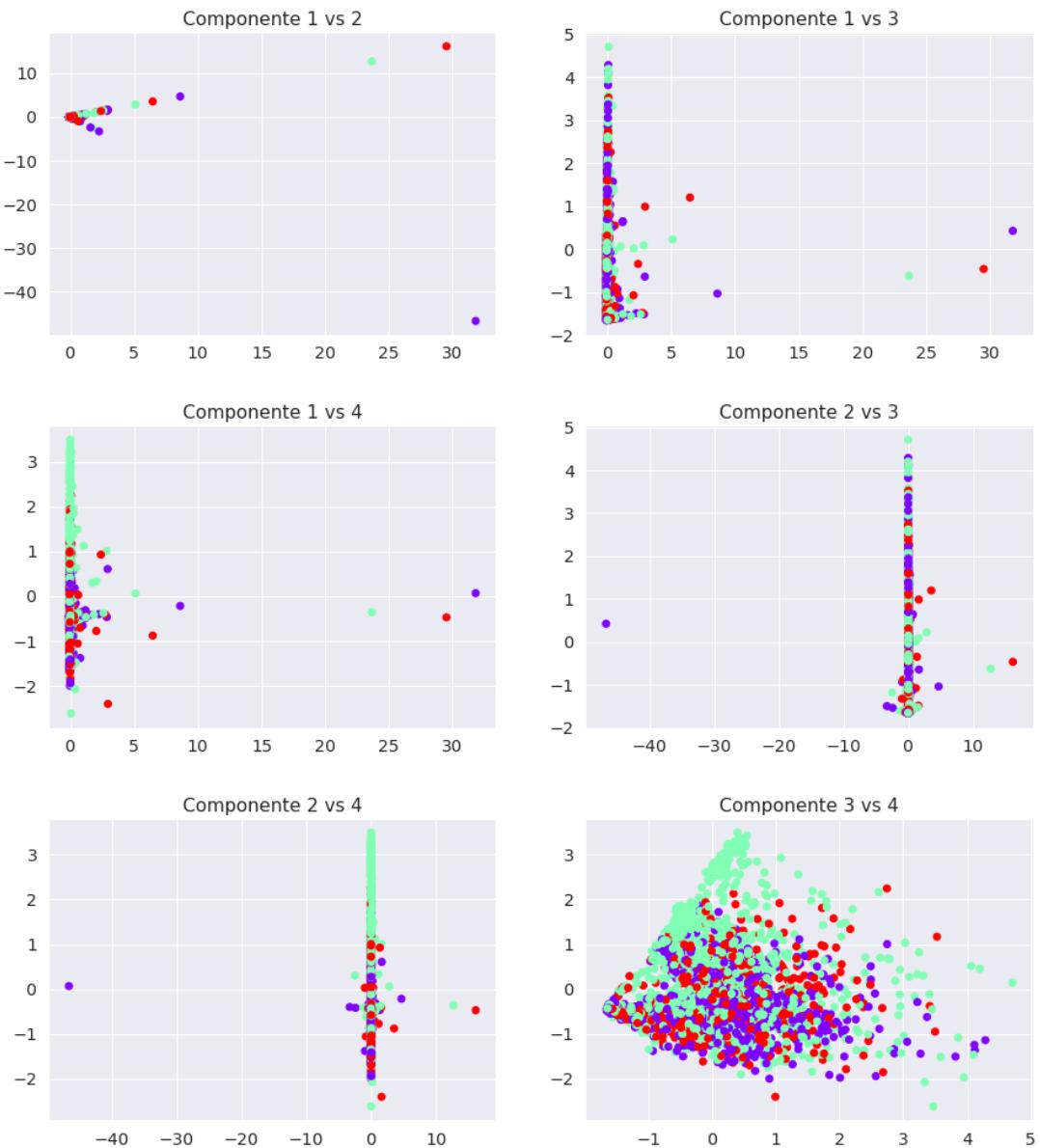
Se necesita también normalizar los datos

```
In [19]: from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X_train=scaler.fit_transform(X_train)  
X_test=scaler.transform(X_test)
```

```
In [20]: X_train.shape
```

```
Out[20]: (2698, 4)
```

```
In [21]: fig, axes = plt.subplots(3, 2, figsize=(15,17))  
axes[0, 0].scatter(X_train[:,0], X_train[:,1], c=Y_train, cmap=plt.cm.rainbow)  
axes[0, 0].set_title('Componente 1 vs 2')  
# plt.ylabel('Componente 2')  
# plt.xlabel('Componente 1')  
  
axes[0, 1].scatter(X_train[:,0], X_train[:,2], c=Y_train, cmap=plt.cm.rainbow)  
axes[0, 1].set_title('Componente 1 vs 3')  
# plt.ylabel('Componente 3')  
# plt.xlabel('Componente 1')  
  
axes[1, 0].scatter(X_train[:,0], X_train[:,3], c=Y_train, cmap=plt.cm.rainbow)  
axes[1, 0].set_title('Componente 1 vs 4')  
# plt.ylabel('Componente 4')  
# plt.xlabel('Componente 1')  
  
axes[1, 1].scatter(X_train[:,1], X_train[:,2], c=Y_train, cmap=plt.cm.rainbow)  
axes[1, 1].set_title('Componente 2 vs 3')  
# plt.ylabel('Componente 3')  
# plt.xlabel('Componente 2')  
  
axes[2, 0].scatter(X_train[:,1], X_train[:,3], c=Y_train, cmap=plt.cm.rainbow)  
axes[2, 0].set_title('Componente 2 vs 4')  
# plt.ylabel('Componente 4')  
# plt.xlabel('Componente 2')  
  
axes[2, 1].scatter(X_train[:,2], X_train[:,3], c=Y_train, cmap=plt.cm.rainbow)  
axes[2, 1].set_title('Componente 3 vs 4')  
# plt.ylabel('Componente 4')  
# plt.xlabel('Componente 3')  
  
plt.subplots_adjust(hspace=0.3)  
  
fig.savefig("./Figuras/PCA_vs_normalizado.pdf", bbox_inches='tight')
```



1.7.2 SVM

```
In [22]: from sklearn import svm
        from sklearn.metrics import classification_report
        SVM_clf=svm.SVC(gamma= 'auto')
        SVM_clf.fit(X_train,Y_train)
        Y_predicted=SVM_clf.predict(X_test)
        print(set(Y_predicted))
```

```

target_names=['Tranquilo', 'Normal', 'Agresivo']
print(classification_report(Y_test, Y_predicted,target_names=target_names))

{1}
      precision    recall   f1-score   support
Tranquilo       0.00     0.00     0.00      313
      Normal       0.50     1.00     0.67      579
      Agresivo      0.00     0.00     0.00      265
avg / total       0.25     0.50     0.33     1157

```

1.7.3 Random Forest

```

In [23]: from sklearn.ensemble import RandomForestClassifier
RF_clf = RandomForestClassifier(n_estimators=50, max_depth=50, random_state=0)
RF_clf.fit(X_train,Y_train)
Y_predicted=RF_clf.predict(X_test)
print(set(Y_predicted))

target_names=['Tranquilo', 'Normal', 'Agresivo']
print(classification_report(Y_test, Y_predicted,target_names=target_names))

{0, 1, 2}
      precision    recall   f1-score   support
Tranquilo       0.33     0.34     0.33      313
      Normal       0.54     0.64     0.58      579
      Agresivo      0.25     0.14     0.18      265
avg / total       0.41     0.44     0.42     1157

```

1.7.4 Neural Network

```

In [24]: # activation : {'identity', 'logistic', 'tanh', 'relu'}
# solver : {'lbfgs', 'sgd', 'adam'}

from sklearn.neural_network import MLPClassifier
MLP_clf = MLPClassifier(solver='lbfgs', alpha=1e-5, activation='identity',
                        hidden_layer_sizes=(15,15,15), random_state=1)
MLP_clf.fit(X_train,Y_train)
Y_predicted=MLP_clf.predict(X_test)
print(set(Y_predicted))

```

```
target_names=['Tranquilo','Normal','Agresivo']
print(classification_report(Y_test, Y_predicted,target_names=target_names))

{0, 1, 2}
      precision    recall   f1-score   support

Tranquilo       0.30      0.19      0.24      313
     Normal       0.52      0.85      0.65      579
  Agresivo       1.00      0.00      0.01      265

avg / total       0.57      0.48      0.39     1157
```

C. Lista de Materiales y Costos

Tabla C.1 Costo de componentes electrónicos (parte 1)

<i>Componente</i>	<i>Cantidad</i>	<i>Precio Unitario (\$)</i>	<i>Fabricante</i>	<i>Precio Total (\$)</i>
OBD2	1	4.99	Moonar	4.99
SCREWTERMINAL-3.5MM-8	1	2.05	Phoenix Contact	2.05
BAT20J	1	0.4	STMicroelectronics	0.4
MMBT2222AL	3	0.23	Infineon Technologies	0.69
0.1pF ±0.1pF 250V Ceramic Capacitor	1	0.98	AVX Corporation	0.98
0.1μF ±10% 16V Ceramic Capacitor	2	0.1	Samsung Electro-Mechanics	0.2
0.1pF ±10% 50V Ceramic Capacitor	5	0.15	Yageo	0.75
100 kΩ -0603-1/10W-1%	1	0.01	Susumu	0.01
10 kΩ -0603-1/10W-1%	3	0.1	Yageo	0.3
10 μF-0603-6.3V-20%	1	0.26	Murata Electronics North America	0.26
10 pF-1210-50V-20%	2	0.78	Taiyo Yuden	1.56
11.8 kΩ 0.1% 1/10W	1	0.7	Panasonic Electronic Components	0.7
180 Ω 5% 1/4W	2	0.1	Rohm Semiconductor	0.2
1 kΩ 5% 1/4W	1	0.1	Rohm Semiconductor	0.1
1.0 nF/1000 pF-0603-50V-5%	1	0.12	Murata Electronics North America	0.12
22 Ω 1% 1/10W	3	0.1	Samsung Electro-Mechanics	0.3

Tabla C.2 Costo de componentes electrónicos (parte 2)

Componente	Cantidad	Precio Unitario (\$)	Fabricante	Precio Total (\$)
22 pF-0603-50V-5%	1	0.10	Samsung Electro-Mechanics	0.10
22 µF-0805-6.3V-20%	1	0.22	Samsung Electro-Mechanics	0.22
22 µF-1210-16V-20%	3	0.69	TDK Corporation	2.07
3.3 µH - 4.8A - 21 MΩ	1	1.29	TDK Corporation	1.29
4.7 kΩ -0603-1/10W-5%	4	0.10	Yageo	0.40
4.7 µF-POLAR-EIA3528-35V-10%(TANT)	1	1.03	AVX Corporation	1.03
47 kΩ -0603-1/10W-1%	3	0.10	Yageo	0.30
47 µF-POLAR-EIA3528-10V-10%	1	0.96	AVX Corporation	0.96
49.9 kΩ 0.1% 1/10W 0805	1	0.54	Stackpole Electronics Inc.	0.54
78171-0004	1	0.66	Molex, LLC	0.66
8200 pF 630V CH 1210	1	1.03	TDK Corporation	1.03
ESP32-WROOM-32	1	6.80	Espressif Systems	6.80
ICM-20648	1	7.68	TDK InvenSense	7.68
LT1764A	1	8.75	Linear Technology/Analog Devices	8.75
MS621FE	1	2.04	Seiko Instruments	2.04
NEO-M8N	1	34.58	U-Blox America Inc.	34.58
CONN PWR JACK 2.5X5.5MM SOLDER	1	0.77	CUI Inc.	0.77
SIM800L	1	3.50	SIM COM	3.50
SIM8055	1	1.31	GCT	1.31
CONN SMA JACK R/A 50Ω PCB	2	3.34	Taoglas Limited	6.68
SMF05CT1G	1	0.42	ON Semiconductor	0.42
TPS563210DDFR	1	1.65	Texas Instruments	1.65
			TOTAL	96.39