

Universidad Católica de Santa María
Facultad de Ciencias e Ingenierías Físicas y Formales
Escuela Profesional de Ingeniería Electrónica



**Diseño de una Plataforma IOT para la Industria 4.0 en Motores de
Inducción Eléctricos con Tiempos de Muestreo Menores a 50ms**

Tesis presentada por el Bachiller:

Espinoza Lima, Luis Fernando

ORCID: 0009-0009-4679-2510

para optar el Título Profesional de Ingeniero Electrónico
con Especialidad en Automatización y Control

Asesor:

Ing. Quispe Yauyo, Juan Medardo

ORCID: 0000-0003-0331-1472

Arequipa – Perú

2024

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA
INGENIERIA ELECTRONICA
CON ESPECIALIDAD EN AUTOMATIZACIÓN Y CONTROL
TITULACIÓN CON TESIS
DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 04 de Julio del 2024

Dictamen: 009158-C-EPIE-2024

Visto el borrador del expediente 009158, presentado por:

2007192201 - ESPINOZA LIMA LUIS FERNANDO

Titulado:

**DISEÑO DE UNA PLATAFORMA IOT PARA LA INDUSTRIA 4.0 EN MOTORES DE INDUCCIÓN
ELÉCTRICOS CON TIEMPOS DE MUESTREO MENORES A 50MS**

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

**INGENIERO ELECTRONICO CON ESPECIALIDAD EN
AUTOMATIZACIÓN Y CONTROL**

**29267682 - RODRIGUEZ GONZALES PEDRO ALEX
DICTAMINADOR**



**29715414 - COAGUILA GOMEZ RONALD PERCING
DICTAMINADOR**



**29565731 - COPA PINEDA JUAN CARLOS
DICTAMINADOR**



Diseño de una Plataforma IOT para la Industria 4.0 en Motores de Inducción Eléctricos con Tiempos de Muestreo Menores a 50ms

INFORME DE ORIGINALIDAD

12%

INDICE DE SIMILITUD

12%

FUENTES DE INTERNET

2%

PUBLICACIONES

4%

TRABAJOS DEL
ESTUDIANTE

FUENTES PRIMARIAS

1

dspace.ups.edu.ec

Fuente de Internet

1%

2

hdl.handle.net

Fuente de Internet

1%

3

Submitted to Universidad Católica de Santa
María

Trabajo del estudiante

1%

4

www.ibm.com

Fuente de Internet

<1%

5

qdoc.tips

Fuente de Internet

<1%

6

repositorio.upla.edu.pe

Fuente de Internet

<1%

7

dspace.esPOCH.edu.ec

Fuente de Internet

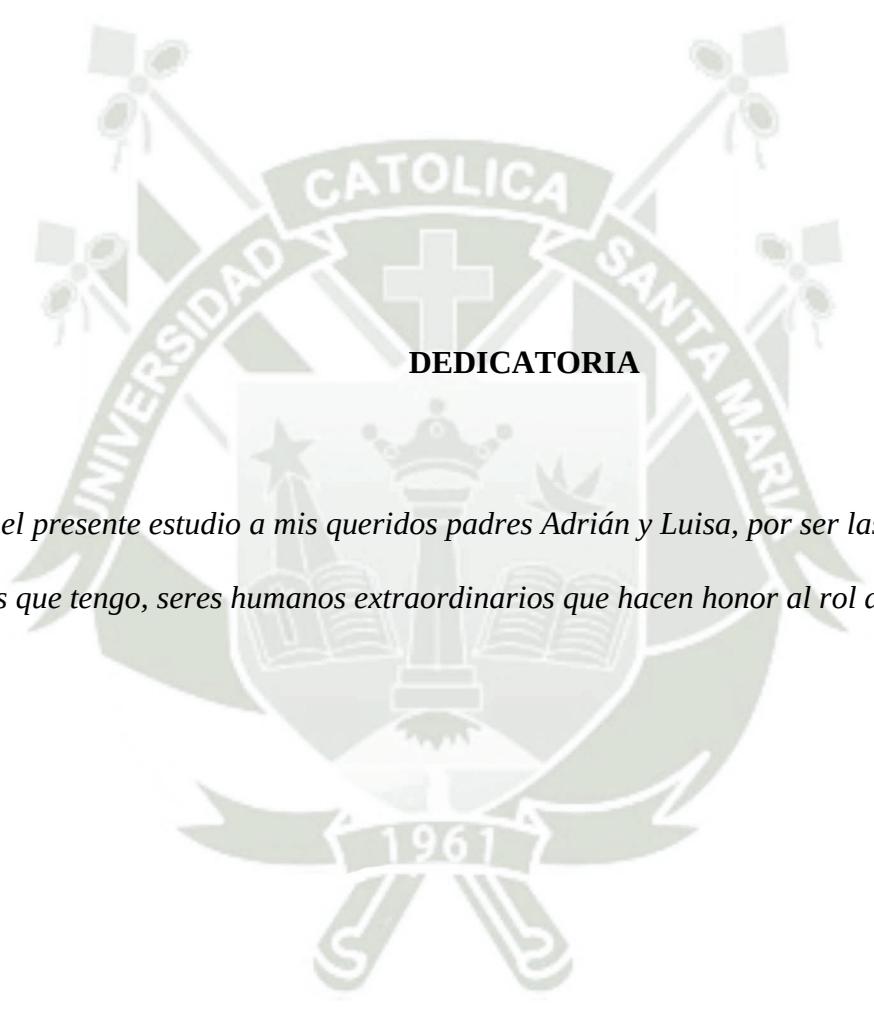
<1%

8

repositorio.utC.edu.ec

Fuente de Internet

<1%



DEDICATORIA

“Dedico el presente estudio a mis queridos padres Adrián y Luisa, por ser las personas más valiosas que tengo, seres humanos extraordinarios que hacen honor al rol de ser padres”

AGRADECIMIENTOS

Quiero expresar mi profundo agradecimiento a todas las personas que han contribuido de alguna manera en la realización de esta tesis.

Agradezco a nuestro divino hacedor por ser siempre mi guía espiritual, sobre todo en los momentos más difíciles.

Agradezco a mis padres Adrián y Luisa, por su amor incondicional, paciencia y constante apoyo.

Agradezco a mis hermanos Alex y Karina, porque son personas guías y ejemplares en mi vida.

Agradezco a mis docentes compañeros de clase y amigos de la UCSM, quienes me han brindado su apoyo y motivación a lo largo de este estudio. Sus palabras de aliento y ánimo han sido un impulso fundamental a seguir adelante.

RESUMEN

Los motores de inducción eléctricos son usados ampliamente en el sector industrial, principalmente por su capacidad de funcionamiento por largos periodos de tiempo; sin embargo, según fabricantes, centros de reparación y estudios académicos indican que el 45% aproximadamente de las fallas por desgaste, se dan en los rodamientos; así mismo es preciso mencionar, que los análisis de vibraciones son los más usados para detectar dichas fallas.

Como una alternativa de mejora a los planes de mantenimientos predictivos de fallas en los motores de inducción eléctricos; este estudio presenta el diseño e implementación de un sistema electrónico de medición de voltaje, corriente, temperatura, vibraciones y RPM basadas en Internet of Things (IOT); con tiempos de muestreo de 1.15ms que hace posible visualizar dichas mediciones en graficas chart, en cualquier navegador web, con respecto al tiempo, en forma continua y en tiempo real.

Cuenta con una tarjeta de desarrollo electrónico ESP32 para la captura, procesamiento, encaminado de mediciones a la nube, entre otros; además, cuenta con sensores de alta complejidad y precisión como el MPU6050, formando parte de los dispositivos que hacen la medición y captura de las magnitudes físicas propias de los motores de inducción eléctricos.

De la misma manera, cuenta con un Dashboards propio desarrollado en JavaScript en base a una librería de gráficos de uso libre no comercial, cuyo código está alojado en la memoria flash del ESP32; por último, la comunicación utiliza el protocolo WebSocket que es base para una medición con baja latencia de muestreo y sin necesidad de refresh.

Palabras claves:

Sensores Electrónicos, Internet of Things, Protocolo de Comunicación WebSocket.

ABSTRACT

Electric induction motors are widely used in the industrial sector, mainly due to their ability to operate for long periods of time; However, they are vulnerable to failures or breakdowns. According to manufacturers, repair centers and academic studies, approximately 45% of failures occur in the bearings; Likewise, it is necessary to mention that vibration analyzes are the most used to detect these failures.

As an alternative to improve predictive maintenance plans for failures in electric induction motors; This study presents the design and implementation of an electronic system for measuring voltage, current, temperature, vibrations and RPM based on the Internet of Things (IOT); with sampling times of 1.15ms that makes it possible to view these measurements in chart graphs, in any web browser, with respect to time, continuously and in real time.

It has an ESP32 electronic development board for capturing, processing, routing measurements to the cloud, among others; In addition, it has highly complex and precise sensors such as the MPU6050, forming part of the devices that measure and capture the physical magnitudes of electric induction motors.

In the same way, it has its own Dashboards developed in JavaScript based on a free, non-commercial use graphics library, whose code is hosted in the ESP32 flash memory; Finally, the communication uses the WebSocket protocol, which is the basis for a measurement with low sampling latency and without the need for refresh.

KEY WORDS:

Electronic Sensors, Internet of Strings, WebSocket Communication Protocol.

ÍNDICE

RESUMEN.....	5
ABSTRACT.....	6
ÍNDICE	7
INTRODUCCIÓN	22
NOMENCLATURAS	23
CAPÍTULO I.....	25
PLANTEAMIENTO DEL PROBLEMA	26
1.1 Identificación del Problema	26
1.2 Descripción del Problema	26
1.3 Justificación del Problema	27
1.4 Objetivos	27
1.4.1 Objetivo General	27
1.4.2 Objetivos Específicos	27
1.5 Aportes	28
1.6 Hipótesis.....	28
1.7 Variables.....	29
1.7.1 Variable Independiente.....	29
1.7.2 Variables Dependientes	29
1.8 Operacionalización de Variables.....	30
CAPÍTULO II	31

MARCO TEÓRICO.....	32
2.1 Antecedentes de Investigación	32
2.1.1 Antecedentes Internacionales	32
2.1.2 Antecedentes Nacionales.....	33
2.1.3 Antecedentes Locales	34
2.2 Bases Teóricas.....	34
2.2.1 Revolución Industrial	34
2.2.2 IOT - Internet de las cosas.....	35
2.2.3 Sistemas de Medición IOT	37
2.2.4 Plataformas IOT	37
2.2.5 Protocolos de Comunicación.....	38
2.2.6 Motores Eléctricos.....	40
2.2.7 Vibraciones en Motores de Inducción.....	41
2.3 Herramientas y dispositivos	41
2.3.1 El entorno de desarrollo Arduino ID.....	41
2.3.2 Tarjeta de Desarrollo ESP 32	41
2.3.3 Sensor de Voltaje.....	42
2.3.4 Sensor de Corriente	42
2.3.5 Sensor de Temperatura	42
2.3.6 Sensor de Vibraciones	43
2.3.7 Sensor de RPM.....	43

2.4	Medición de Parámetros del Motor de Inducción	43
2.4.1	Voltaje de Alimentación del Motor	44
2.4.2	Corriente de Consumo del Motor	44
2.4.3	Temperatura del Motor	44
2.4.4	Vibraciones del Motor	44
2.4.5	RPM del Motor	45
CAPÍTULO III		46
METODOLOGÍA		47
3.1	Tipo de Investigación	47
3.2	Nivel de Investigación	47
3.3	Métodos de Investigación	47
3.4	Diseño de la Investigación	48
3.5	Población y Muestra	49
3.5.1	Población	49
3.5.2	Muestra	49
3.6	Método para la Recopilación de Datos	49
3.7	Instrumentos para la Recopilación de Datos	51
3.8	Técnica de Diseño Electrónico	51
3.8.1	Selección de Componentes y Dispositivos	51
3.8.2	Integración de Hardware y Software	52
3.8.3	Pruebas y Validación	52

CAPÍTULO IV.....	53
DISEÑO DE HARDWARE.....	54
4.1 Selección de Dispositivos.....	54
4.1.1 Motor	54
4.1.2 Sensor de Voltaje.....	54
4.1.3 Sensor de Corriente	55
4.1.4 Sensor de Temperatura	56
4.1.5 Sensor de RPM.....	56
4.1.6 Sensor de Vibraciones	57
4.1.7 Conversor Analógico Digital ADC	58
4.1.8 Tarjeta de Desarrollo	58
4.2 Arquitectura del diseño	59
4.2.1 Capa Edge.....	60
4.2.2 Capa Fog.....	61
4.2.3 Capa Cloud	61
4.3 Diagrama de Bloques General del Diseño	63
4.4 Diseño y Dimensionamiento de Elementos Eléctricos	63
4.4.1 Dimensionamiento del Motor.....	63
4.4.2 Elementos del Arranque simple del motor	65
4.4.3 Diseño del Circuito del Arranque Simple del Motor	68
4.5 Diseño y Dimensionamiento de Elementos de Medición	70

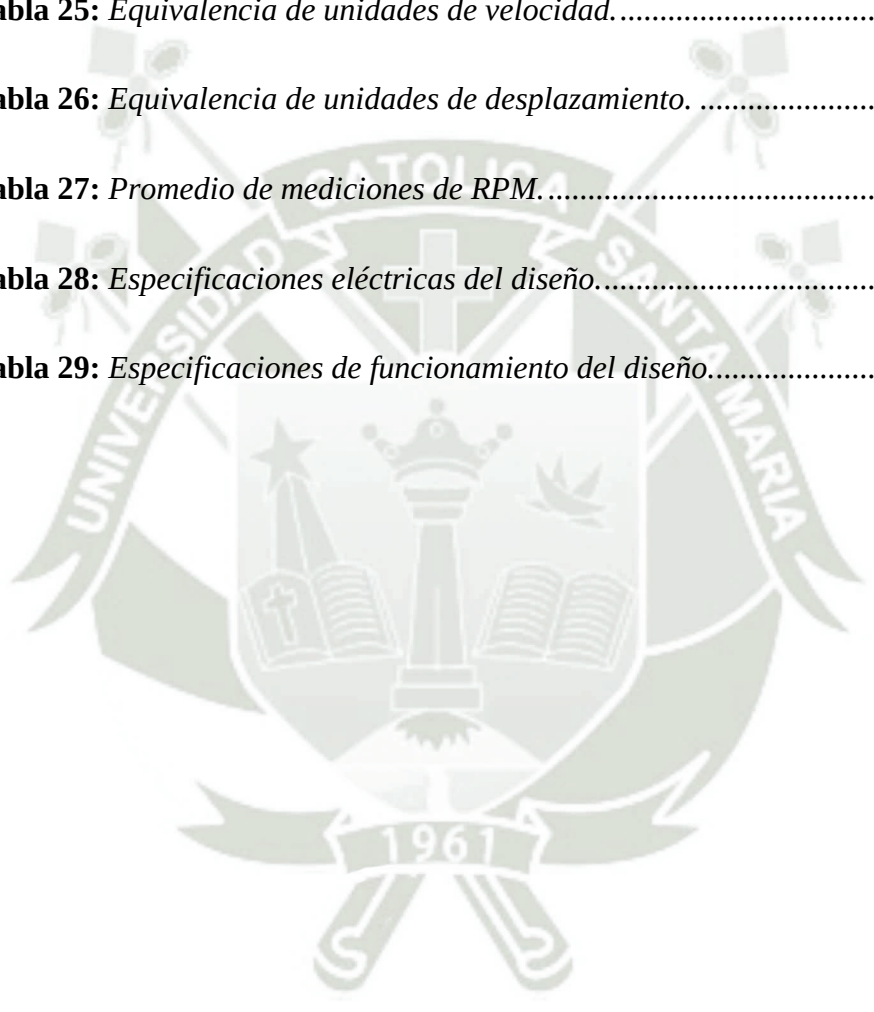
4.5.1	Sensor de voltaje	70
4.5.2	Sensor de Corriente	74
4.5.3	Sensor de Temperatura	77
4.5.4	Sensor de Vibraciones	80
4.5.5	Sensor de RPM	83
4.6	Descripción del ADS1115	86
4.7	Descripción del ESP32	87
4.7.1	Tarjeta de Desarrollo DEVKIT DEVKIT V1	88
4.8	Diagrama Circuitual del Diseño	90
4.9	Especificaciones Técnicas	91
CAPITULO V		92
DISEÑO DE SOFTWARE		93
5.1	Selección del Protocolo de Comunicación	93
5.2	Programación de Gateway	96
5.2.1	Estructura y Diseño del Programa	96
5.2.2	Recogida y Procesamiento de Medidas de Edge	101
5.2.3	Encaminado de Medidas a Cloud	109
5.2.4	Stop en las Gráficas	119
5.2.5	Encendido del LED	120
5.3	Programación de Cloud	122
5.3.1	Estructura y Diseño del Programa	122

5.3.2 Visualización de Medidas	133
5.3.3 Interfaz donde se Muestran los Valores Numéricos.....	138
CAPITULO VI.....	140
PRUEBAS RESULTADOS Y DISCUSIÓN	141
6.1 Prototipo Terminado del Diseño	141
6.1.1 Diseño de Hardware Terminado.....	141
6.1.2 Diseño de Software Terminado	143
6.2 Pruebas y Mediciones.....	144
6.2.1 Medición de Voltaje	144
6.2.2 Medición de Corriente.....	147
6.2.3 Medición de Temperatura	149
6.2.4 Medición de Vibraciones.....	151
6.2.5 Medición de RPM	159
6.3 Hojas Finales de Especificaciones Eléctricas y de Funcionamiento.....	161
6.4 Discusiones.....	162
6.5 Consideraciones Generales	165
CONCLUSIONES	166
RECOMENDACIONES	168
REFERENCIAS BIBLIOGRÁFICAS.....	169
ANEXOS	178

ÍNDICE DE TABLAS

Tabla 1: <i>Operacionalización de Variables.</i>	30
Tabla 2: <i>Selección de motores de inducción eléctricos</i>	54
Tabla 3: <i>Selección del sensor de voltaje.</i>	55
Tabla 4: <i>Selección del sensor de corriente</i>	55
Tabla 5: <i>Selección del sensor de temperatura.</i>	56
Tabla 6: <i>Selección del sensor de RPM.</i>	57
Tabla 7: <i>Selección del sensor de vibraciones.</i>	57
Tabla 8: <i>Selección del sensor conversor analógico digital.</i>	58
Tabla 9: <i>Selección de la tarjeta de desarrollo.</i>	59
Tabla 10: <i>Elementos conformados por la capa Edge.</i>	60
Tabla 11: <i>Nodos.</i>	61
Tabla 12: <i>Características técnicas del motor.</i>	63
Tabla 13: <i>Características del contactor.</i>	66
Tabla 14: <i>Características de la llave termomagnética.</i>	67
Tabla 15: <i>Tabla de alambres esmaltados.</i>	72
Tabla 16: <i>Hoja de datos del SCT013.</i>	75
Tabla 17: <i>Medidas técnicas de la ranura.</i>	85
Tabla 18: <i>Especificaciones técnicas.</i>	91
Tabla 19: <i>Principales características de protocolos Wifi.</i>	94
Tabla 20: <i>Librerías para el ESP32.</i>	97

Tabla 21: <i>Promedio de mediciones de voltaje.</i>	147
Tabla 22: <i>Promedio de mediciones de corriente.</i>	149
Tabla 23: <i>Promedio de mediciones de Temperatura.</i>	150
Tabla 24: <i>Equivalencia de unidades de aceleración.</i>	153
Tabla 25: <i>Equivalencia de unidades de velocidad.</i>	155
Tabla 26: <i>Equivalencia de unidades de desplazamiento.</i>	158
Tabla 27: <i>Promedio de mediciones de RPM.</i>	160
Tabla 28: <i>Especificaciones eléctricas del diseño.</i>	161
Tabla 29: <i>Especificaciones de funcionamiento del diseño.</i>	162



ÍNDICE DE FIGURAS

Figura 1: <i>Diagrama de bloques general.</i>	60
Figura 2: <i>Arquitectura final.</i>	62
Figura 3: <i>Diagrama de bloques de la arquitectura.</i>	63
Figura 4: Características del motor Pretul.	64
Figura 5: Contactor.	66
Figura 6: Llave termomagnética.	68
Figura 7: Esquema del Arranque simple de motor.	69
Figura 8: Instalación del arranque del motor.	69
Figura 9: Transformador, sensor de voltaje.	70
Figura 10: Diagrama del transformador, sensor de voltaje.	71
Figura 11: Lectura del transformador.	73
Figura 12: Diagrama circuital del sensor de voltaje.	74
Figura 13: Sensor SCT013.	74
Figura 14: Placa del motor, 1.5A.	75
Figura 15: Medición de corriente.	76
Figura 16: Circuito del sensor de corriente.	77
Figura 17: Sensor de corriente.	77
Figura 18: Diagrama de distribución de pines del sensor LM35.	78
Figura 19: Temperaturas de operación de motores.	79
Figura 20: Medición de temperatura.	79

Figura 21: Circuito del sensor de temperatura.....	80
Figura 22: Sensor de MPU6050.	81
Figura 23: Circuito del ESP32.....	82
Figura 24: Sensor de AR-FC03 montado en el motor.	83
Figura 25: Diagrama interno del AR-FC03.	83
Figura 26: Sensor AR-FC03.	84
Figura 27: Circuito del AR-FC03.....	85
Figura 28: Diagrama interno del ADS1115.....	86
Figura 29: Tarjeta de desarrollo ESP32 DEVKIT V1.	88
Figura 30: regulador de voltaje 1117C/33-C238.	89
Figura 31: Módulo SHIELD ESP32.....	89
Figura 32: Diagrama circuital del diseño.....	90
Figura 33: <i>Diagrama de flujo general del sketch.</i>	96
Figura 34: <i>Librerías del programa.</i>	98
Figura 35: <i>Inicio de comunicación.</i>	98
Figura 36: <i>Inicio de comunicación Wifi.</i>	99
Figura 37: <i>Declaración del objeto server.</i>	99
Figura 38: <i>Declaración de objeto websocket.</i>	100
Figura 39: <i>Inicio de server.</i>	100
Figura 40: <i>Sensor de corriente.</i>	100
Figura 41: <i>Inicio de SPIFFS.</i>	101

Figura 42: <i>Define almacenamiento en memoria flash.</i>	101
Figura 43: <i>Escalamiento de voltaje.</i>	103
Figura 44: <i>Resta voltaje DC.</i>	103
Figura 45: <i>Escalamiento de corriente.</i>	104
Figura 46: <i>Escalamiento de temperatura.</i>	105
Figura 47: <i>Salida numérica del sensor de vibraciones.</i>	106
Figura 48: <i>Escalamiento de aceleración.</i>	107
Figura 49: <i>Escalamiento de velocidad.</i>	107
Figura 50: <i>Escalamiento de desplazamiento.</i>	107
Figura 51: <i>Define interrupción.</i>	108
Figura 52: <i>Bucle de interrupción.</i>	108
Figura 53: <i>Función millis timer.</i>	109
Figura 54: <i>Completa a un minuto el conteo.</i>	109
Figura 55: <i>Declara objeto Json.</i>	110
Figura 56: <i>Declara segundo objeto object.</i>	110
Figura 57: <i>Sensor de corriente.</i>	111
Figura 58: <i>Convierte Json en cadena.</i>	111
Figura 59: <i>Envía mediciones a Cloud.</i>	111
Figura 60: <i>Declara variable v.</i>	112
Figura 61: <i>Habilita encaminado de voltaje.</i>	112
Figura 62: <i>Almacena voltaje en objeto.</i>	112

Figura 63: <i>Encaminado de voltaje.</i>	112
Figura 64: <i>Declara variable c.</i>	113
Figura 65: <i>Habilita encaminado de corriente.</i>	113
Figura 66: <i>Almacena corriente en objeto.</i>	113
Figura 67: <i>Encaminado de corriente.</i>	113
Figura 68: <i>Declara variable t.</i>	114
Figura 69: <i>Habilita encaminado de temperatura.</i>	114
Figura 70: <i>Almacena temperatura en objeto.</i>	114
Figura 71: <i>Encaminado de temperatura.</i>	114
Figura 72: <i>Declaración de la variable va.</i>	115
Figura 73: <i>Habilita encaminado de la aceleración.</i>	115
Figura 74: <i>Almacena aceleración en objeto.</i>	115
Figura 75: <i>Encaminado de aceleración.</i>	116
Figura 76: <i>Declaración de vv.</i>	116
Figura 77: <i>Habilita encaminado de velocidad.</i>	116
Figura 78: <i>Almacena velocidad en objeto.</i>	116
Figura 79: <i>Encaminado de velocidad.</i>	117
Figura 80: <i>Declaración de vd.</i>	117
Figura 81: <i>Habilita encaminado de desplazamiento.</i>	117
Figura 82: <i>Almacena desplazamiento en objeto.</i>	117
Figura 83: <i>Encaminado de desplazamiento.</i>	118

Figura 84: <i>Declaración de r.</i>	118
Figura 85: <i>Habilita encaminado de RPM.</i>	118
Figura 86: <i>Almacena RPM en objeto.</i>	118
Figura 87: <i>Encaminado de RPM.</i>	119
Figura 88: <i>Declaración de p.</i>	119
Figura 89: <i>Define pin 4 del ESP32.</i>	119
Figura 90: <i>Define Pin como entrada.</i>	120
Figura 91: <i>Habilita STOP.</i>	120
Figura 92: <i>Rutina de control STOP.</i>	120
Figura 93: <i>Rompe el buce infinito.</i>	120
Figura 94: <i>Declaración del pin 2 del ESP32.</i>	121
Figura 95: <i>Define Led como salida.</i>	121
Figura 96: <i>Rutina de control ON/OFF del LED.</i>	121
Figura 97: <i>Enciende y apaga LED.</i>	121
Figura 98: <i>Diagrama de flujo principal de Cloud.</i>	123
Figura 99: <i>Diagrama de flujo HTML.</i>	124
Figura 100: <i>Diagrama de chart.</i>	125
Figura 101: <i>Diagrama de flujo JavaScript.</i>	126
Figura 102: <i>Id para el chart.</i>	126
Figura 103: <i>Clase de las botoneras.</i>	127
Figura 104: <i>Define Highcharts.</i>	127

Figura 105: Puerta de ingreso de mediciones.	128
Figura 106: Define tiempo para eje x.	128
Figura 107: Define mediciones para eje y.	128
Figura 108: Almacena 150 puntos en array.	129
Figura 109: Dinamiza el chart.	129
Figura 110: Sensor de corriente.	130
Figura 111: Muestreo en señal de 60Hz.	131
Figura 112: Mediciones de tiempo en onda.	132
Figura 113: Visualización de voltaje.	134
Figura 114: Visualización de corriente.	134
Figura 115: Visualización de temperatura.	135
Figura 116: Visualización de la aceleración.	136
Figura 117: Visualización de la velocidad.	136
Figura 118: visualización del desplazamiento.	137
Figura 119: Visualización de RPM.	138
Figura 120: Interfaz de medición.	139
Figura 121: Prototipo terminado.	142
Figura 122: Partes del prototipo.	142
Figura 123: Código C++ en Arduino IDE.	143
Figura 124: Dashboard terminado.	144
Figura 125: Resultados de voltaje.	145

Figura 126: <i>Comprobación de voltaje.</i>	146
Figura 127: <i>Resultados de corriente.</i>	147
Figura 128: <i>Comprobación de corriente.</i>	148
Figura 129: <i>Resultados de temperatura.</i>	149
Figura 130: <i>Comprobación de temperatura.</i>	150
Figura 131: <i>Resultados de aceleración.</i>	151
Figura 132: <i>Gráfica de aceleración.</i>	152
Figura 133: <i>Comparación de gráficas de aceleración.</i>	152
Figura 134: <i>Resultados de velocidad.</i>	153
Figura 135: <i>Gráfica de velocidad.</i>	154
Figura 136: <i>Comparación de velocidades.</i>	154
Figura 137: <i>Cuadro de severidad ISO 10816.</i>	155
Figura 138: <i>Valor de la velocidad 200mm/s.</i>	156
Figura 139: <i>Ubicación del motor la ISO 10816.</i>	156
Figura 140: <i>Resultados de desplazamiento.</i>	157
Figura 141: <i>Gráfica de desplazamiento.</i>	158
Figura 142: <i>Comparación de desplazamiento.</i>	158
Figura 143: <i>Resultados de RPM.</i>	159
Figura 144: <i>Comprobación de RPM.</i>	160

INTRODUCCIÓN

En este estudio presentamos el diseño e implementación de una aplicación IOT, que permitirá medir de forma inalámbrica y en tiempo real magnitudes físicas inherentes al funcionamiento de los motores de inducción eléctricos mediante sensores conectados de forma no invasiva. Esta práctica aporta información valiosa, constante y con menos recursos con el fin de detectar fallas, haciendo uso de las diferentes gráficas en la nube y en tiempo real.

Este diseño cuenta con sensores no invasivos, que se encargan de convertir las diferentes magnitudes físicas en señales eléctricas. También cuenta con módulos externos ADC encargados de digitalizar adecuadamente las señales analógicas para su posterior procesamiento. También cuenta con la tarjeta de desarrollo ESP32 usado como Gateway; es decir, captura, procesa y encamina las mediciones a la nube (Cloud) utilizando los protocolos de comunicación I2C (para la captura de mediciones) y webSocket (para el encaminado de mediciones a la nube). Por último, cuenta con un Dashboards propio, desarrollado en JavaScript utilizando una librería de gráficos, alojada en memoria flash del ESP32.

Finalmente, la tesis se estructura en seis capítulos: Planteamiento del Problema, Marco Teórico, Metodología, Diseño de Hardware, Diseño de Software, Pruebas Resultados y Discusión. Cada capítulo aborda un aspecto importante del proyecto, desde la conceptualización del problema y el marco teórico que lo sustenta, hasta la metodología de investigación, el desarrollo de la propuesta tecnológica, la presentación y discusión de los resultados obtenidos.

NOMENCLATURAS

IOT (Internet de las Cosas): Se refiere al conjunto de dispositivos conectados a Internet que intercambian y procesan datos.

Protocolos de Comunicación: Reglas y formatos que permiten la transferencia de datos entre dispositivos, como MQTT, webSocket, etc.

WebSocket: Es un protocolo de comunicación full dúplex persistente entre Servidor y cliente.

Edge: Dispositivo encargado de la captura de información sobre el proceso de monitorizado.

Fog Gateway: Capa superior a Edge que da servicios de cálculo, almacenamiento, procesamiento de datos y funciones de red que conecta la infraestructura a la nube en aplicaciones IOT. Es un intermediario entre Edge y Cloud.

Cloud: Capa superior a Fog que da servicios de almacenamiento y visualización de la información.

Chart: Representación a través de gráficos con respecto al tiempo una información, datos o series de tiempo.

IEC 947-4: Norma de la Comisión Electrotécnica Internacional que especifica categorías de uso para contactores.

ISO 108163: norma de análisis de vibraciones en valores de severidad.

Arduino IDE: Una plataforma de hardware de código abierto para la creación de prototipos electrónicos.

ESP32: tarjeta de desarrollo electrónico con capacidad de Wifi y Bluetooth.

Contactador: Un dispositivo eléctrico utilizado para el control de arranque y parada de motores eléctricos.

Interruptor Termomagnético: Un dispositivo de protección que combina funciones térmicas y magnéticas para proteger contra sobrecargas y cortocircuitos.

Luces de Indicación: Dispositivos visuales que indican diferentes estados operativos de un motor, como funcionamiento normal, fallo, o parada de emergencia.

Motor de Inducción Eléctrico: Un tipo de motor eléctrico que funciona mediante la inducción electromagnética de forma asíncrona.

Motor 0.5 HP: Se refiere a un motor eléctrico de medio caballo de fuerza equivalente a 368W.

Transformador: Es una máquina eléctrica estática que funciona mediante la inducción electromagnética, con dos devanados sin contacto físico.

SCT013: Sensor de corriente tipo transformador no invasivo.

LM35: Sensor de temperatura analógico que proporciona una salida de voltaje proporcional a la temperatura.

MPU6050: Sensor de vibración con comunicación I2C.

AR-FC03: Sensor de RPM encoder que mide la velocidad de rotación.

RPM: Revoluciones por minuto.



CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Identificación del Problema

Los motores de inducción eléctricos son utilizados ampliamente en la industria (algunos autores afirman que más del 90% del total de los motores instalados en industrias de todo el mundo son motores de inducción), también se estima en la industria que más del 70% del consumo de energía eléctrica, lo hacen los motores de inducción y que más del 46% de consumo de energía mundial, los hacen los motores de inducción eléctricos (Walde & Brunner, 2011).

Tomando en cuenta estos datos es evidente que los motores de inducción son una parte vital para la industria mundial, también es evidente asumir que necesitan dichos motores planes de mantenimiento cada vez más eficientes y de acuerdo a los avances tecnológicos para su continuidad operativa.

1.2 Descripción del Problema

Las aplicaciones IOT dan una alternativa de mejora a los planes de mantenimiento especialmente predictivos, haciéndolo un proceso más eficiente a través de una herramienta virtual que permite gestionarlo de una manera más ágil y oportuna, pero la gran mayoría de aplicaciones IOT para motores de inducción, pierden gran cantidad de datos porque están basados en el almacenamiento de información, que utilizan en su gran mayoría protocolos de comunicación MQTT, HTTP, etc., obteniendo latencias de muestreo muy altas por consiguiente resulta imposible graficar una señal de 60Hz y mucho menos hacer un análisis de vibración en la nube.

Gracias a esta problemática se han sentado las bases para el desarrollo de este estudio.

1.3 Justificación del Problema

El presente estudio busca abordar estos problemas a través del diseño e implementación de una aplicación IOT con tiempos de muestreo menores a 50ms específico para motores de inducción eléctricos, que tenga la capacidad de mostrar las gráficas requeridas por las diferentes técnicas o análisis de detección de fallas, aportando mejoras a la gestión de mantenimiento predictivo de dichos motores. La propuesta incluye el desarrollo de una solución tecnológica que pueda ser implementada incluso en contextos donde el acceso a Internet es limitado.

1.4 Objetivos

1.4.1 *Objetivo General*

Diseñar una plataforma utilizando tecnología IOT para la industria 4.0 en motores de inducción eléctricos con transmisión de datos en forma inalámbrica, con tiempos de muestreo menores a 50ms.

1.4.2 *Objetivos Específicos*

- Conocer las características y parámetros de un sistema eléctrico de bombeo de circulación cerrada de agua, para simular una carga al motor de inducción seleccionado.
- Diseñar el hardware haciendo una óptima selección de componentes y dispositivos.
- Diseñar el software haciendo una selección del protocolo de comunicación Wifi para establecer una comunicación bidireccional y a baja latencia (menores a 50ms).
- Visualizar las mediciones de las variables en gráficas chart, en el navegador web.

- Diseñar un selector que permita visualizar las mediciones, en una sola gráfica.

1.5 Aportes

Las aportaciones que han dado la elaboración de este estudio se mencionan a continuación.

- La elaboración de documentación donde se detalla en profundidad los procedimientos de diseño para el hardware y software de la plataforma IOT de bajas latencias de muestreo.

A continuación, se precisa las herramientas de ingeniería aplicadas.

- ✓ Manejo del protocolo de comunicación I2C.
- ✓ Manejo del PGA (Amplificador de ganancia programable).
- ✓ Acondicionamiento de señal y muestreo de datos con el ADS1115.
- ✓ Programación orientada a objetos usando diferentes lenguajes como C++, JavaScript, HTML.
- ✓ Manejo del protocolo WebSocket usado para envío y recepción de datos por puerto 81.
- ✓ Manejo de mensajería en formato Json.
- ✓ Ejecución del paradigma de fog computing.
- ✓ Alojamiento web en la tarjeta de desarrollo electrónico ESP32.
- ✓ Desarrollo y despliegue del uso de la librería Highcharts para visualizar y dinamizar gráficas en tiempo real con tiempos de muestreo de 1.15ms.

1.6 Hipótesis

Con el diseño de la plataforma IOT aplicado en la industria 4.0 será probable medir parámetros de voltaje, corriente, temperatura, vibraciones y RPM de los motores eléctricos con tiempos de muestreo menores a 50ms.

1.7 Variables

1.7.1 *Variable Independiente*

Diseño de la Plataforma IOT

1.7.2 *Variables Dependientes*

Voltaje de alimentación del motor.

Corriente de alimentación del motor.

Temperatura del motor.

Vibraciones del motor.

RPM del motor



1.8 Operacionalización de Variables

Tipo de Variable	Variable	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Escala de medición
Independiente	Plataforma IOT	Es un sistema de medición inalámbrica basados en IOT	Es el instrumento de medición	Ondas electromagnéticas	Multímetro	Intervalo
	Voltaje del motor	Es la alimentación del motor.	Es el voltaje medido en voltios (V) entre los terminales del motor.	Potencial eléctrico	Voltaje	Razón
Dependiente	Corriente del motor	Es el flujo de carga eléctrica que circula por el motor.	Es la corriente medida en amperios (A) que atraviesa el motor.	Carga eléctrica	Corriente	Razón
	Temperatura del motor	Es la medida del grado de calor o frío del motor.	Es la temperatura del motor expresada en grados Celsius (°C) en la superficie externa del mismo.	Calor	Temperatura del motor	Intervalo
	Vibraciones del motor	Es el movimiento oscilatorio del motor alrededor de una posición de equilibrio.	Es la vibración medidas en: aceleración en milímetros por segundo al cuadrado (mm/s ²), velocidad, milímetros por segundo (m/s), desplazamiento en milímetros (mm).	Aceleración, velocidad, desplazamiento	Vibraciones	Razón
	RPM	Revoluciones por minuto	Velocidad de giro en (r/min)	Mecánico	Eje del motor	Intervalo

Tabla 1: Operacionalización de Variables.

Fuente: Elaboración propia.



CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes de Investigación

2.1.1 Antecedentes Internacionales

Según Martínez en su estudio titulado “ Sistema IOT para monitorización de motores eléctricos y detección de anomalías mediante el análisis de vibraciones de rodamientos”. Este estudio se basa en el análisis de vibraciones en la capa cloud aplicado a la detección de fallas en rodamientos de motores de inducción eléctricos. Para la medición de datos ha usado el sensor multipropósito Steval-MKSBOX1V1, para su procesamiento ha utilizado el Raspberry Pi 4 y para visualizar sus gráficas utilizó la plataforma. Los resultados señalan que si cumplió sus objetivos de muestrear y graficar las señales de vibraciones, pero también el autor señala que tuvo muchos problemas como la pérdida de datos al momento de establecer la conexión con la nube, también menciona que ha solucionado el problema haciendo el envío de datos por paquetes a la nube. (Martínez, 2022)

Cháfer en su estudio titulado “Diseño de Aplicación IOT para el diagnóstico de motores de inducción”. El objetivo principal es construir un sistema de diagnóstico remoto de motores de inducción cuyo fin es poder detectar posibles averías y malfuncionamientos a través de una medición y procesamiento periódico de la velocidad de giro en la nube. Ha utilizado el microcontrolador ESP31 para su lectura de datos, procesamiento y envío a la nube. También hizo uso de la plataforma Node-RED utilizando el protocolo de comunicación MQTT. Los resultados muestran que el sistema propuesto es capaz de medir datos simulados por un generador de funciones dando una onda senoidal de 2Hz y la intensidad de corriente de la base datos de un motor real, este último con una latencia de refresco de 1s, analizados espectralmente a una frecuencia de 1K. (Cháfer, 2020)

Espin en su estudio “Desarrollo De Una Red Local De Sensores Inalámbricos Para Un Proceso De Control De Motores Ac Enlazados A Un Sistema IOT” El objetivo principal es el desarrollo de un sistema de monitoreo remoto de sensores conectados a motores AC, con el fin de dar soporte a la mediana y pequeña industria. Para llevar a cabo este estudio utilizó sensores de corriente tipo transformador, velocidad encoder tipo incremental a su vez a usado los microcontroladores ATmega328p y ESP8266, el primero para la recopilación de datos y el segundo para su procesamiento y envío de datos a la nube. También la utilizado la plataforma Ubidots que utiliza el protocolo de comunicación MQTT para visualizar sus data. Los resultados obtenidos muestran valores esperados con pruebas de intensidad de corriente y velocidades en diferentes configuraciones de arranques de motores de inducción trifásicos. (Espin Pallango & Peña Reyes, 2023)

2.1.2 Antecedentes Nacionales

Huacho en su estudio titulado “Diseño de un módulo de control basado en plataforma IOT para el monitoreo remoto de motores de inducción de baja potencia. Huancayo - 2022”. El objetivo principal de este estudio, es diseñar el módulo de control basado en plataforma IOT para el monitoreo remoto de motores de inducción de baja potencia, utilizando herramientas de la industria 4.0. El tipo de investigación es aplicada, el enfoque es cuantitativo, el alcance es descriptivo, la muestra es un motor de inducción trifásico de 1.5 kW, y los instrumentos son un microcontrolador ESP32, un microcontrolador ATmega 2560, un sensor de corriente y voltaje PZEM-004T, un sensor de temperatura termopar tipo K. También la plataforma Ubidots que utiliza el protocolo MQTT. Los resultados son satisfactorios y cumplen con su objetivo de mostrar sus datos en forma de tablas, tiene un tiempo de muestreo de 10s. (Huacho, 2022)

Flores en su estudio titulado "Diseño de un prototipo utilizando IOT para la supervisión de medidores eléctricos con tecnología LoRa". El objetivo principal es diseñar un prototipo utilizando IOT para supervisión de medidores eléctricos con tecnología LoRa. El tipo de

investigación es aplicada, el enfoque es cuantitativo, el alcance es descriptivo, la muestra es un medidor inteligente monofásico, Los resultados muestran que el prototipo es capaz de medir el consumo de energía eléctrica, enviar los datos a la nube mediante el protocolo MQTT, mostrar los datos en tiempo real y en forma de gráficos en una plataforma web y una aplicación móvil, y generar alertas en caso de sobrecarga o corte de energía. En esta investigación se concluyó que el prototipo es una solución de bajo coste, fácil de implementar y que permite el monitoreo y control remoto de medidores eléctricos mediante IOT. (Flores, 2023)

2.1.3 Antecedentes Locales

Quilla en su estudio titulado "Diseño e implementación de un prototipo de medidor inteligente monofásico para el monitoreo de energía en hogares empleando tecnología de internet de las cosas (IOT)" El objetivo principal del estudio es diseñar e implementar un prototipo de medidor inteligente monofásico para el monitoreo de energía en hogares empleando tecnología IOT. Utilizó un microcontrolador ESP32 para la lectura de datos, un ADC ADS1115 para el muestreo de datos, un sensor de corriente SCT013, un sensor de voltaje ZMPT101B no invasivos y la plataforma Thinger que utiliza el protocolo de comunicación MQTT. Sus resultados son favorables puesto que hace una comparación de medidas con las mediciones con un medidor de energía eléctrica real. Su tiempo de muestreo es de 500ms (Quilla, 2023)

2.2 Bases Teóricas

2.2.1 Revolución Industrial

La era inicial de la Revolución Industrial marcó el surgimiento de la mecanización, iniciando alrededor de 1765. La segunda revolución tuvo lugar con la introducción de la electricidad en 1870, mientras que la tercera se caracterizó por la llegada de la electrónica, las telecomunicaciones y las computadoras en 1969. La Cuarta Revolución Industrial ha

acontecido en la última década, siendo conocida como Industria 4.0. Este término hace referencia a la actual revolución que ha transformado la industria mediante la integración de tecnologías avanzadas, redefiniendo la forma en que se llevan a cabo los procesos productivos y de fabricación. (Meski, Belkadi, Laroche, & Furet, 2019)

Uno de los objetivos fundamentales es aprovechar de manera inteligente las tecnologías emergentes para potenciar la eficiencia, la productividad y la flexibilidad en los procesos de fabricación. Estas innovaciones comprenden sistemas ciberfísicos, internet de las cosas, computación en la nube, realidad aumentada y virtual, análisis de big data, así como internet de servicios. La implementación estratégica de estas herramientas busca transformar y optimizar significativamente la manera en que se lleva a cabo la producción y la prestación de servicios. (Culot et al., 2019), computación móvil e identificación por radiofrecuencia (RFID) (Salvivar, y otros, 2015)

La Industria 4.0 representa la última fase de la evolución industrial, donde se busca integrar y utilizar avanzadas tecnologías para optimizar los procesos de fabricación y responder a los desafíos contemporáneos de manera más eficiente y flexible.

2.2.2 IOT - Internet de las cosas

El Internet de las cosas (IOT) es un sistema que comprende sensores informáticos y dispositivos digitales conectados a nivel mundial a través de Internet. Estos elementos se comunican e intercambian información entre sí mediante el uso de un identificador único asignado a cada dispositivo, conocido como UID (identificador único). Este UID también puede ser compartido, permitiendo así la transmisión de información entre los distintos componentes del sistema. (Khudhair & Ghani, 2020)

El Internet de las cosas (IOT) se define como un sistema que consta de sensores informáticos y dispositivos digitales conectados a nivel mundial mediante Internet. Estos

componentes se comunican y comparten información entre sí utilizando un identificador único asignado a cada dispositivo, conocido como UID. Este identificador único permite la comunicación y el intercambio de datos entre los diferentes nodos del sistema, en esencia, el IOT facilita la conexión y la colaboración entre dispositivos digitales a escala global, permitiendo un intercambio eficiente de información entre ellos.

2.2.2.1 *Aplicaciones IOT.*

IOT tiene una gran aplicación, como se detalla a continuación:

- ✓ Los exámenes inteligentes que se valen del Internet de las cosas (IOT) tienen la capacidad de identificar y abordar las dificultades y obstáculos que enfrentan los estudiantes al responder preguntas. Este sistema posibilita un análisis detallado de la capacidad de un estudiante para comprender temas específicos evaluados durante el examen.
- ✓ En la actualidad, se están desarrollando sistemas de monitoreo de vehículos con el objetivo de obtener información en tiempo real sobre los movimientos de los vehículos y realizar un seguimiento detallado de su rendimiento. (Tewari & Damilola, 2020)

El Internet de las cosas (IOT) también proporciona a los críticos la oportunidad de abordar de manera abierta la seguridad asociada con la transferencia de datos a sistemas abiertos en la nube mediante el IOT. La implementación del IOT demanda una atención cuidadosa y la adopción de precauciones debido a los desafíos relacionados con la seguridad y la protección de datos en entornos conectados. (Hussain, Butt, & Uzma, 2019)

El IOT abre la puerta a discusiones críticas sobre la seguridad de transferir datos a sistemas abiertos en la nube. La implementación del IOT requiere precaución y cuidado debido a los desafíos asociados con la seguridad y la protección de datos en entornos conectados. La

atención a estos aspectos es crucial para garantizar un uso efectivo y seguro de las tecnologías IOT.

2.2.3 *Sistemas de Medición IOT*

Las industrias acordes a las nuevas tecnologías exigen cada vez más que los dispositivos y mediciones de las diferentes magnitudes físicas involucradas en procesos de producción, estén conectados a internet con la finalidad que se pueda acceder a dicha data desde cualquier lugar.

Las mediciones son de vital importancia para mantener la calidad en los procesos de producción industrial, también para la elaboración de los planes de mantenimiento a si mismo sin mediciones no sería posible la existencia de la industria.

2.2.4 *Plataformas IOT*

La diversidad en los modelos de arquitectura de red de sensores, la volatilidad de estos sensores y la ausencia de metodologías comunes y buenas prácticas para su implementación, según (Zanella, y otros, 2014), sirven como un ejemplar ilustrativo de la complejidad de las soluciones IOT. Empresas han lanzado sus propias soluciones IOT, ofreciendo plataformas listas para usar que proporcionan beneficios diversos, incluyendo la integración nativa de computación en la nube para toda la infraestructura confiada por los usuarios. Sin embargo, surge una preocupación significativa con respecto a la falta de transparencia y detalles de implementación en estas plataformas, ya que las grandes empresas son renuentes a divulgar aspectos técnicos clave debido al riesgo de revelar información estratégica.

Esta falta de transparencia representa desventajas para la comunidad científica, ya que las soluciones IOT basadas en estas plataformas tradicionales son, en cierto sentido, "cajas negras". Por ende, el objetivo de Calderoni et al. (2019) es proponer una solución completamente local basada en tecnología de código abierto. Esto incluiría el desarrollo de

middleware flexible que facilite la conectividad de alta velocidad para redes de sensores heterogéneas, siguiendo un modelo conceptual similar al de las redes de sensores globales. (Diepgen, y otros, 2007)

Destaca los desafíos en la implementación de soluciones IOT debido a la diversidad de modelos de arquitectura de red de sensores, la volatilidad de los sensores y la carencia de metodologías comunes. Se menciona que algunas empresas ofrecen soluciones listas para usar con beneficios, como la integración nativa de computación en la nube, pero también destaca la falta de transparencia y detalles de implementación en estas plataformas debido a la reluctancia de las grandes empresas a revelar información estratégica y técnica.

2.2.5 Protocolos de Comunicación

Hoy en día, la proliferación de dispositivos IOT ha dado lugar al desarrollo de varios protocolos y estándares IOT. A diferencia de las computadoras convencionales, los dispositivos IOT suelen tener limitaciones en cuanto a capacidad de almacenamiento y potencia de procesamiento. Además, estos dispositivos se despliegan en entornos donde el acceso a una fuente continua de energía es limitado o inexistente, lo que implica que deben operar con baterías o pequeños paneles solares.

Dada esta situación, se han creado protocolos de comunicación de bajo consumo que poseen huellas de memoria reducidas y requisitos de procesamiento limitados para satisfacer las necesidades específicas de los dispositivos IOT. El protocolo TCP/IP tradicional no fue diseñado teniendo en cuenta estas restricciones. No obstante, en los últimos años, los protocolos IOT se han estandarizado en prácticamente todas las capas de la pila de protocolos para adaptarse mejor a las características y limitaciones de estos dispositivos. (Ahlgren, Lindgren, & Wu, 2016)

La proliferación de dispositivos IOT ha impulsado el desarrollo de diversos protocolos y estándares adaptados a las limitaciones específicas de estos dispositivos en términos de capacidad de almacenamiento y potencia de procesamiento. La evolución de los protocolos ha desempeñado un papel crucial para optimizar la eficiencia y el rendimiento de los dispositivos IOT en diversos entornos y condiciones operativas.

2.2.5.1 Protocolo de Comunicación MQTT.

MQTT es un protocolo de mensajería basado en estándares o un conjunto de reglas para comunicarse de una computadora a otra. Los sensores inteligentes, los dispositivos portátiles y otros dispositivos de Internet de las cosas (IoT) que no necesiten un ancho de banda amplio o latencias de transmisión bajas utilizan, esta herramienta MQTT que admite mensajería de dispositivo a nube y de nube a dispositivo.

2.2.5.2 Protocolo de Comunicación HTTP.

HTTP es el protocolo más usado en la web, y se utiliza para cargar páginas web mediante enlaces de hipertexto. HTTP es un protocolo de capa de aplicación diseñado para transferir información entre los dispositivos conectados de la red. Es también usado para comunicación entre dispositivos IOT que requieren poca velocidad de transmisión.

2.2.5.3 Protocolo de Comunicación WebSocket.

WebSocket es un protocolo de comunicación bidireccional persistente entre (navegador - servidor), su comunicación utiliza el puerto 81 razón por la cual usa ws// en la URL. Sus principales características son:

- Full Dúplex (servidor/cliente) envía datos en cualquier momento.
- Única conexión persistente, siempre conectado sin afectar su eficiencia.
- Orientado a eventos (mensajes - notificaciones)

- Baja latencia y bajo consumo de datos.
- Podemos abrir y cerrar la conexión en cualquier momento.

Estas características hacen que sea ideal para aplicaciones IoT en tiempo real.

2.2.6 Motores Eléctricos

Un motor eléctrico transforma energía eléctrica en mecánica mediante dos componentes clave: el estator fijo y el rotor móvil. Fabricados en material ferromagnético con ranuras para conductores de cobre, estos elementos interactúan para generar movimiento mecánico, aprovechando la energía eléctrica a través de sus devanados eléctricos. (Fundación Red de Energía - BUN-CA, 2011)

2.2.6.1 Motor Síncronos.

Los motores síncronos son motores de corriente alterna cuya velocidad de rotación del eje está directamente relacionada con la frecuencia de la corriente suministrada. Estos motores se caracterizan por tener una inductancia alimentada directamente, siendo utilizados ampliamente en diversas aplicaciones como relojes, grabadoras y servomecanismos de electrodomésticos por su precisión y eficiencia. (Cembranos, 2014)

2.2.6.2 Motor de Inducción Eléctrico.

Los motores de inducción son máquinas eléctricas de corriente alterna compuestas por un rotor móvil y un estator con devanados. Estos motores se distinguen por su sencillez, fiabilidad y bajo costo. Funcionan al inducir corriente en el rotor a través del campo magnético del estator, lo que los hace ampliamente utilizados en la industria para diversas aplicaciones. Este motor asíncrono es ampliamente utilizado en la industria debido a sus ventajas significativas en términos de operación, mantenimiento y costos. Sin embargo, presenta desafíos, como la dificultad para controlar con precisión la posición, y tiene un factor de potencia y eficiencia operativa ligeramente inferiores en comparación con otras máquinas.

Aunque no requiere potencia en el rotor, lo que simplifica su diseño, esto impide ajustar el factor de potencia que consumen. (Cembranos, 2014)

2.2.7 Vibraciones en Motores de Inducción

Las principales fallas en los motores de inducción eléctricos, se presentan en los rodamientos; es preciso mencionar, que las fallas que conllevan al mal funcionamiento de los motores de inducción se precisan en los siguientes puntos.

- Mal montaje, precarga excesiva o desgaste de cajas.
- Fallas en el procedimiento de lubricación, exceso o deficiencia de lubricante.
- Lubricante de mala calidad o incompatible con la aplicación.
- Lubricante contaminado.
- Excesiva vibración durante operación.
- Fallas del cableado de aterramiento.

2.3 Herramientas y dispositivos

2.3.1 El entorno de desarrollo Arduino ID

El Arduino ID es un entorno de desarrollo de código abierto basado en el lenguaje de programación C++, siendo el más popular en estos tiempos por su versatilidad y masiva difusión (Vital, 2021)

2.3.2 Tarjeta de Desarrollo ESP 32

El ESP32 fabricado por la empresa Espressif Systems, es una tarjeta de desarrollo electrónico de bajo consumo de energía, de bajo costo y que ofrece buenas capacidades de procesamiento gracias al doble núcleo que tiene; a su vez, tiene Wifi y bluetooth integrado, haciéndolo único y ideal para aplicaciones IOT.

Sus principales partes y características incluyen:

- **Doble núcleo:** El ESP32 tiene dos núcleos de procesador Xtensa 32-bit que pueden funcionar simultáneamente, lo que lo hace potente para tareas de procesamiento.
- **Memoria RAM:** Tiene una RAM de 520KB, que se utiliza para almacenar datos y variables temporales.
- **Memoria Flash:** Tiene una memoria Flash de 4MB.
- **Wifi y Bluetooth:** Incluye módulos Wifi y Bluetooth integrados para la comunicación inalámbrica.

2.3.3 *Sensor de Voltaje*

Un sensor de voltaje es un dispositivo electrónico que mide el nivel de tensión eléctrica en un circuito o sistema. Captura la diferencia de potencial entre dos puntos y la convierte en una señal eléctrica proporcional. Estos sensores son esenciales para monitorear y controlar voltajes en diversas aplicaciones, como electrónica, automatización industrial y sistemas de seguridad, proporcionando información crítica sobre el estado eléctrico. (Rios , 2008)

2.3.4 *Sensor de Corriente*

Un sensor de corriente es un dispositivo que detecta y mide la intensidad de la corriente eléctrica que fluye a través de un conductor. Utiliza principios electromagnéticos o efectos Hall para generar una señal proporcional a la corriente. Estos sensores son fundamentales en aplicaciones de control y monitoreo eléctrico, permitiendo medir y supervisar la corriente en circuitos eléctricos, motores, sistemas de energía y más. (Romero, 2015)

2.3.5 *Sensor de Temperatura*

Un sensor de temperatura es un dispositivo que mide y registra la temperatura en su entorno. Utiliza una variedad de tecnologías, como termistores, termopares o sensores de resistencia, para convertir la temperatura en una señal eléctrica. Estos sensores son cruciales

en una amplia gama de aplicaciones, desde el control de clima de máquinas como motores hasta la monitorización de la temperatura en procesos industriales y la medición de la temperatura corporal en aplicaciones médicas. (Hernandez, 2018)

2.3.6 Sensor de Vibraciones

Un sensor de vibración es un dispositivo que detecta y mide las oscilaciones o movimientos vibratorios en su entorno. Utiliza componentes como acelerómetros o sensores piezoeléctricos para transformar las vibraciones en señales eléctricas. Estos sensores se emplean en diversas aplicaciones, desde la detección de fallas en maquinaria industrial hasta la monitorización de la vibración en estructuras civiles y la respuesta a terremotos en sistemas de alerta sísmica. (Ycaza, 2019)

2.3.7 Sensor de RPM

Un sensor de velocidad RPM (revoluciones por minuto) es un dispositivo que mide la velocidad de rotación de un eje o componente mecánico. Utiliza diversas tecnologías, como sensores ópticos, inductivos o magnéticos, para detectar las revoluciones y convertirlas en una señal eléctrica proporcional a la velocidad. Estos sensores son esenciales en aplicaciones automotrices, industriales y de maquinaria, donde se requiere el monitoreo preciso de las RPM para el control y la seguridad. (Mejia & Armijos, 2015)

2.4 Medición de Parámetros del Motor de Inducción

La medición de parámetros de funcionamiento de motores de inducción implica la supervisión continua y el análisis de variables clave que afectan el desempeño y la eficiencia del motor. Esto incluye la medición de voltaje, corriente, temperatura, vibraciones, RPM. (Caballero & Huertas, 2014)

2.4.1 *Voltaje de Alimentación del Motor*

La medición del voltaje del motor en motores de inducción involucra la observación continua y el análisis de la tensión eléctrica aplicada al motor. Mediante el seguimiento del voltaje, se detectan variaciones o desviaciones que podrían indicar situaciones anómalas como sobretensiones o caídas de voltaje, afectando directamente el rendimiento del motor y su durabilidad. (Vieto, 2020)

2.4.2 *Corriente de Consumo del Motor*

La medición de la corriente del motor en motores de inducción es una práctica esencial que consiste en medir y analizar la corriente eléctrica que fluye a través del motor. Este proceso permite identificar fluctuaciones y anomalías en la corriente, lo cual es fundamental para prevenir sobrecargas y otros problemas eléctricos. (Mercedez & Guzman, 2014)

2.4.3 *Temperatura del Motor*

La medición de temperatura del motor de inducción es el proceso de supervisar y medir continuamente la temperatura de un motor de inducción eléctrico. Se utiliza para garantizar que la temperatura del motor permanezca dentro de los límites seguros de funcionamiento. Esto es crucial para prevenir el sobrecalentamiento, daños en el motor y garantizar un rendimiento eficiente en aplicaciones industriales y sistemas eléctricos donde se utilizan motores de inducción. (Mercedez & Guzman, 2014)

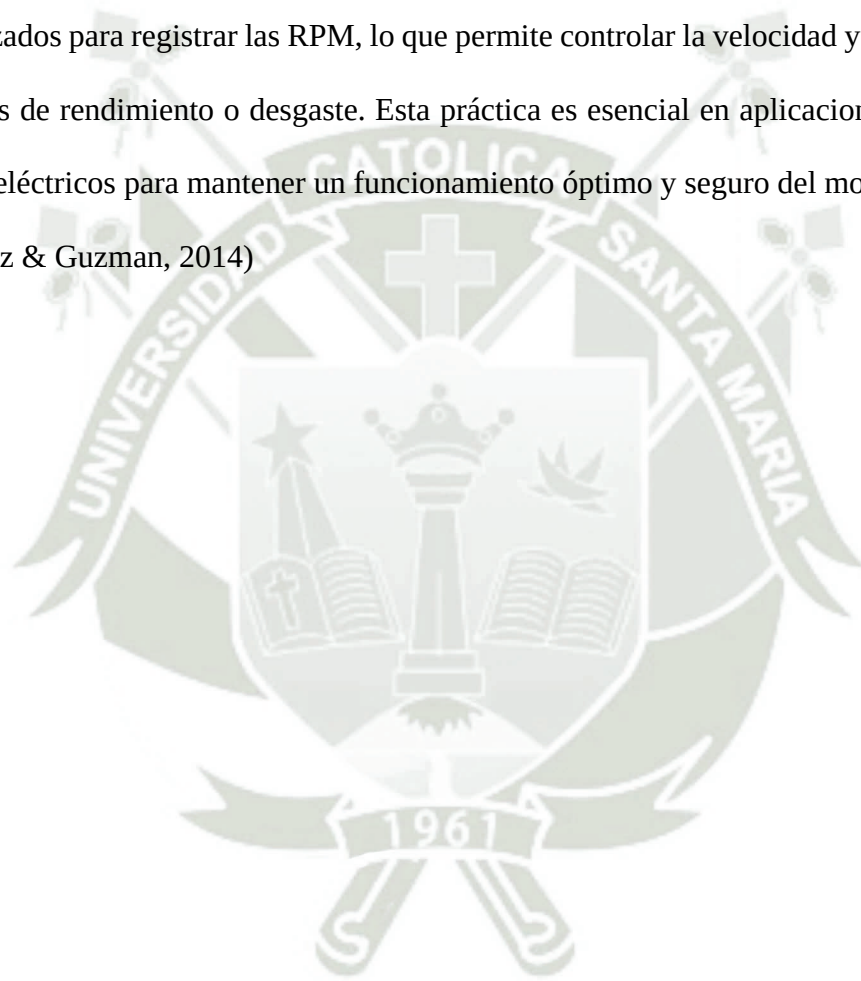
2.4.4 *Vibraciones del Motor*

La medición de vibración del motor de inducción implica la supervisión constante de las vibraciones mecánicas generadas por dicho motor. Utiliza sensores de vibración para medir y registrar estas oscilaciones, lo que permite detectar desequilibrios, desgaste o problemas mecánicos. Este proceso es esencial en aplicaciones industriales y maquinaria para prevenir

daños, mejorar la eficiencia y garantizar un funcionamiento seguro y prolongado del motor de inducción. (Mercedez & Guzman, 2014)

2.4.5 RPM del Motor

La medición de RPM (revoluciones por minuto) del motor de inducción es el proceso de supervisar y medir la velocidad de rotación de dicho motor. Se utilizan sensores especializados para registrar las RPM, lo que permite controlar la velocidad y detectar posibles problemas de rendimiento o desgaste. Esta práctica es esencial en aplicaciones industriales y sistemas eléctricos para mantener un funcionamiento óptimo y seguro del motor de inducción. (Mercedez & Guzman, 2014)





CAPÍTULO III

METODOLOGÍA

3.1 Tipo de Investigación

El tipo de investigación del presente estudio es aplicada, experimental y correlacional. Es aplicada porque busca resolver un problema práctico de la industria 4.0, que es la medición de forma remota de motores de inducción eléctricos con tiempos de muestreo menores a 50 ms. Es experimental porque quiere diseñar una plataforma IOT para la industria 4.0 en motores de inducción eléctricos con tiempos de muestreo menores a 50ms. Es correlacional porque quiere analizar y discutir la relación entre los parámetros de funcionamiento de los motores eléctricos de la plataforma IOT que voy a diseñar e implementar.

3.2 Nivel de Investigación

Este estudio de investigación "Diseño de una plataforma IoT para la Industria 4.0 en motores de inducción eléctricos con tiempos de muestreo menores a 50ms" se sitúa en el nivel aplicativo y de desarrollo tecnológico. En él, se ha aplicado conocimientos teóricos para desarrollar y mejorar tecnologías específicas, orientándome hacia la solución de problemas prácticos en la industria.

Este estudio implica la integración y potencial creación de nuevas soluciones en el ámbito de la IOT, enfocándose en la optimización y monitoreo avanzado de motores de inducción eléctricos, lo cual es crucial para avanzar en la eficiencia y capacidad tecnológica en el contexto de la Industria 4.0.

3.3 Métodos de Investigación

En el presente estudio "Diseño de una plataforma IOT para la Industria 4.0 en motores de inducción eléctricos con tiempos de muestreo menores a 50ms", se ha empleado una metodología mixta que combina enfoques cuantitativos y cualitativos. Desde el lado cuantitativo, se ha concentrado en la recolección y análisis de datos numéricos para probar y

validar la funcionalidad de la plataforma IOT, asegurándose de que cumpla con los requisitos de rendimiento, fiabilidad y capacidad de procesamiento en tiempos inferiores a 50ms. Paralelamente, se ha aplicado métodos cualitativos para entender las necesidades de los usuarios y las condiciones operativas en la Industria 4.0, incluyendo el análisis de procesos operativos existentes.

El enfoque también abarca Investigación y Desarrollo (I+D), donde se ha diseñado, desarrollo y probado iterativamente la plataforma, ajustándola en base a los resultados obtenidos. Se ha realizado análisis experimentales y pruebas en vacío para optimizar la plataforma en diferentes escenarios, lo que es crucial para identificar problemas y áreas de mejora. Finalmente, una vez desarrollada la plataforma, se ha llevado a cabo un estudio de implementación en un entorno real para evaluar su funcionamiento y eficacia en condiciones reales, lo que es esencial para su viabilidad.

3.4 Diseño de la Investigación

En la fase inicial del presente estudio, se ha hecho una exploración y conceptualización profunda, comenzando con una revisión bibliográfica y un análisis a las nuevas tecnologías correspondientes al IOT. Este proceso ayuda a entender las necesidades y brechas en el sector industrial, enfocándose especialmente en los aspectos técnicos relacionados con el tiempo de muestreo en sistemas de motores. Con esta información, se ha definido los requerimientos específicos y diseñado la arquitectura de solución, seleccionando cuidadosamente los sensores, hardware, software y protocolos de comunicación que mejor se adapten a los objetivos establecidos.

Posteriormente, se ha desarrollado prototipos y realizado pruebas experimentales para evaluar su rendimiento en diversas condiciones. Este enfoque permite ajustar y mejorar continuamente el sistema en base a los resultados obtenidos, asegurándose de que cumpla con

los estándares de velocidad, precisión y estabilidad necesarios. La fase final implica probar el sistema en un entorno real, lo que es crucial para validar su efectividad y realizar los ajustes finales antes de una implementación más amplia.

3.5 Población y Muestra

3.5.1 Población

En el presente estudio, se ha considerado como población el conjunto total de motores de inducción eléctricos que podrían beneficiarse de la plataforma IOT que se ha desarrollado. Esta población incluye una amplia gama de motores usados en diversos sectores de la Industria 4.0, como la manufactura y la automatización industrial, donde los motores de inducción juegan un papel fundamental. Además, se ha tomado en cuenta la variedad de entornos operativos en los que estos motores se utilizan, considerando diferentes condiciones como la temperatura, humedad y carga operativa.

3.5.2 Muestra

Para la muestra, se ha seleccionado un motor representativo de esta población para realizar pruebas y análisis más detallados. Reconociendo la impracticabilidad de probar cada motor existente en el mercado, la selección de la muestra ha sido tres motores de inducción eléctricos monofásicos. El objetivo es asegurarse de que la muestra fuera lo suficientemente funcional para que los resultados de las pruebas y análisis fueran generalizables a una población más amplia.

3.6 Método para la Recopilación de Datos

Para la recolección de datos en el presente estudio de diseño de una plataforma IOT para motores de inducción eléctricos, se ha utilizado un enfoque mixto que combinaba métodos cuantitativos y cualitativos:

1. Métodos Cuantitativos:

- Pruebas de Laboratorio y Experimentación: se ha realizado pruebas experimentales con diferentes tipos de motores de inducción eléctricos para recoger datos sobre el rendimiento de la plataforma IOT, enfocándome en el tiempo de muestreo.
- Sensores y Dispositivos de Medición: Se ha utilizado sensores y otros dispositivos de medición para recoger datos en tiempo real sobre el funcionamiento de los motores.

2. Métodos Cualitativos:

Estudios de Caso: se ha analizado casos específicos de uso de motores de inducción en la Industria 4.0 para obtener una comprensión más profunda de los contextos operativos y los requisitos prácticos.

3. Análisis de Documentación:

- Revisión de Literatura y Documentos Técnicos: Se ha revisado publicaciones académicas, manuales técnicos y reportes de la industria para obtener datos secundarios que respaldaran la presente investigación.

4. Observación Directa:

- Visitas a Sitios Industriales: Se ha observado el funcionamiento de los motores en entornos reales para comprender mejor los desafíos prácticos y las condiciones operativas.

Esta combinación de métodos ha permitido recoger una amplia gama de datos, tanto cuantitativos como cualitativos, para obtener una visión completa y precisa del rendimiento y la aplicabilidad de la plataforma IOT en diversos entornos industriales.

3.7 Instrumentos para la Recopilación de Datos

Para la recopilación de datos de este estudio de diseño de la plataforma IOT para motores de inducción eléctricos, se ha utilizado una variedad de instrumentos específicos:

1 Instrumentos de Medición Técnica:

- Sensores IOT: Se ha implementado diversos sensores para medir parámetros como el voltaje, la temperatura, la vibración, RPM y el consumo de energía de los motores de inducción eléctricos.
- Multímetros: Se ha usado multímetros para medir y analizar características eléctricas como el voltaje, la corriente, temperatura y rpm.

2 Software y Herramientas:

- Software editor de código fuente: Se ha utilizado software especializado para la programación del código fuente que ha dado soporte al manejo de sensores.
- Software compilador: Se ha utilizado software especializado para compilar el sketch de la tarjeta de desarrollo usado la captura de datos.

3 Herramientas para Recolección de Datos Cualitativos:

Cámaras fotográficas: Para las capturas de imágenes, se ha utilizado cámaras fotográficas para documentar mis resultados y figuras.

3.8 Técnica de Diseño Electrónico

Son los métodos, prácticas y herramientas utilizadas para desarrollar y diseñar sistemas electrónicos que son compatibles con IOT.

3.8.1 Selección de Componentes y Dispositivos

Se selecciono aquellos componentes y dispositivos como, motores, llaves termomagnéticas, sensores, tarjetas de desarrollo electrónicos (como el ESP32), y otros componentes electrónicos que se ajusten en términos de eficiencia, con bajo nivel de ruido,

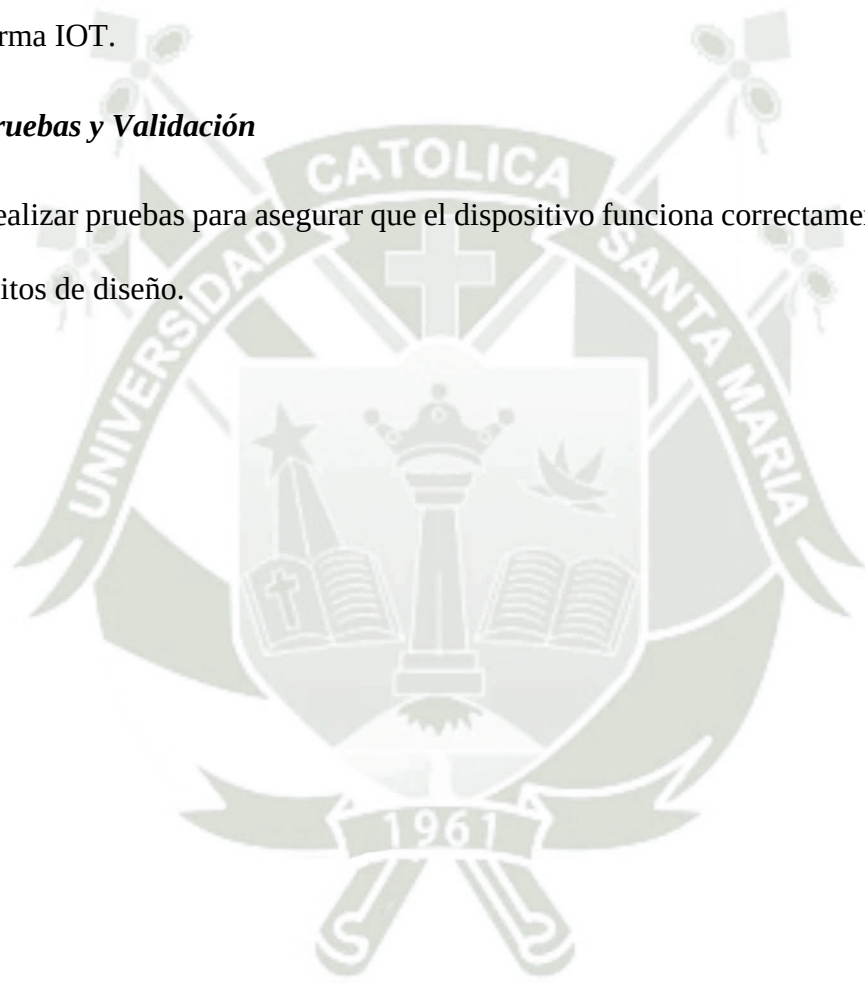
confiables y aplicables a proyectos IOT, para luego elegir después de evaluar y preseleccionar componentes candidatos.

3.8.2 Integración de Hardware y Software

Se ha desarrollado el hardware y software incluyendo la gestión de conectividad, la recopilación, procesamiento y posterior visualización de mediciones para el funcionamiento de la plataforma IOT.

3.8.3 Pruebas y Validación

Realizar pruebas para asegurar que el dispositivo funciona correctamente y cumpla con los requisitos de diseño.





CAPÍTULO IV

DISEÑO DE HARDWARE

4.1 Selección de Dispositivos

4.1.1 Motor

En la tabla 2 se muestra las características de posibles alternativas para la selección del motor.

Parámetros	Motor 1	Motor 2	Motor 3
Marca	SIEMENS	PRETUL	CEMER
Potencia	1.0 HP	0.5 HP	0.5 HP
Fases	Trifásico	Monofásico	Monofásico
Velocidad	3320 r/min	3450 r/min	1410 r/min
Corriente	3.5 A	1.5 A	5.05 A
Voltaje	220 YY/380 YV	220 V	230 V

Tabla 2: Selección de motores de inducción eléctricos
Fuente: Elaboración propia

Conforme a las características mencionadas en la tabla 2 se seleccionó el motor 2 (Pretul), porque este motor es monofásico, es de 220V y consume menos energía.

4.1.2 Sensor de Voltaje

En la tabla 3 se muestra las características de posibles alternativas para la selección del sensor de voltaje.

Parámetros	Sensor 1	Sensor 2	Sensor 3
Código	ZMPT101B	Transformador	PZEM-004T
VRMSin	0 - 250V	0 - 250V	0 - 260V
VRMSo	0 - 5V	0 - 0.5834V	0 - 2600 muestras
Precisión	0.2%	-	0.5%
No linealidad	-	1%	-
Resolución	-	-	0.1V
Alimentación	5V	-	5V

Comunicación	-	-	Serial
Aislamiento	Inductivo	inductivo	Óptico

Tabla 3: Selección del sensor de voltaje.

Fuente: Elaboración propia.

Conforme a las características mencionadas en la tabla 3 se seleccionó el sensor 2, porque es un transformador que cumple las necesidades de este diseño, es decir, no necesita alimentación, es estable, no necesita acondicionamiento de señal y tiene aislamiento entre el devanado primario y secundario.

4.1.3 Sensor de Corriente

En la tabla 4 se muestra las características de posibles alternativas para la selección del sensor de corriente.

Parámetros	Sensor 1	Sensor 2	Sensor 3
Código	ACS712 - 15PCB	SCT - 013	PZEM-004T 10A
IRMSin	0 - 15A	0 - 30A	0 - 10A
Relación I/V	1A/100mV	1A/33mV	-
Precisión	0.2%	-	0.5%
No linealidad	-	1%	-
Resolución	-	-	0.01A
Alimentación	5V	-	5V
Comunicación	-	-	Serial
Aislamiento	Inductivo	inductivo	Óptico

Tabla 4: Selección del sensor de corriente

Fuente: Elaboración propia

Conforme a las características mencionadas en la tabla 4 se seleccionó el sensor de corriente SCT - 013, porque es un transformador que cumple las necesidades de este diseño, es decir, es no invasivo, no necesita alimentación, es estable, no necesita acondicionamiento de señal y tiene aislamiento entre el conductor y devanado secundario.

4.1.4 Sensor de Temperatura

En la tabla 5 se muestra las características de posibles alternativas para la selección del sensor de temperatura.

Parámetros	Sensor 1	Sensor 2	Sensor 3
Código	DHT11	LM35	Termocupla tipo K
Rango	0-50°C	-55 – 150°C	0 – 800°C
Tolerancia	-	-	1.5°C
Precisión	2°C	0.5°C	-
Salida	-	1°C/10mv	100°C/42μV
Resolución	0.1°C	-	-
Comunicación	I2C	-	-

Tabla 5: Selección del sensor de temperatura.

Fuente: Elaboración propia.

Conforme a las características mencionadas en la tabla 5 se seleccionó el sensor de Temperatura LM35, porque es un sensor que cumple las necesidades de este diseño, es decir, tiene una buena precisión, no necesita acondicionamiento de señal y tiene amplio rango de medición.

4.1.5 Sensor de RPM

En la tabla 6 se muestra las características de posibles alternativas para la selección del sensor de RPM.

Parámetros	Sensor 1	Sensor 2
Código	AR2746	AR-FC03
Tipo de Emisor	Fotodiodo	Fotodiodo
Tipo de receptor	Fototransistor	Fototransistor

Longitud de onda del emisor	950nm IR	950nm IR
Ranura	-	5mm
Comparador	-	LM393
Alimentación	5V	3.3 - 5V
Diámetro Exterior	3.5mm	-
Espesor de disco	0.33mm	-
Diámetro Exterior	22mm	-

Tabla 6: Selección del sensor de RPM.

Fuente: Elaboración propia.

Conforme a las características mencionadas en la tabla 6 se seleccionó el sensor AR-C03, porque es un sensor que cumple las necesidades de este diseño, es decir, porque se puede instalar fácilmente al motor y no necesita acondicionamiento de señal.

4.1.6 Sensor de Vibraciones

En la tabla 7 se muestra las características de posibles alternativas para la selección del sensor de vibraciones.

Parámetros	Sensor 1	Sensor 2	Sensor 3
Código	ADXL335	MPU 6050	ADXL345
Sensibilidad	3g	2g, 4g, 8g, 16g	2g, 4g, 8g, 16g
ADC	-	Sigma 16bits	13bit
Multiplexado	-	Independiente	si
Alimentación	3.3-5V	3.3-5V	3.3-5V
Comunicación	-	I2C	I2C
Salida	Analógica	-	-

Tabla 7: Selección del sensor de vibraciones.

Fuente: Elaboración propia.

Conforme a las características mencionadas en la tabla 7 se seleccionó el MPU6050, porque es un sensor que cumple las necesidades de este diseño, es decir, tiene alta sensibilidad, tiene ADC independiente, tiene comunicación I2C y viene calibrado por fábrica.

4.1.7 Conversor Analógico Digital ADC

En la tabla 8 se muestra las características de posibles alternativas para la selección del sensor de voltaje.

Parámetros	ADC 1	ADC 2	ADC 3
Código	PCF8591	ADS1115	ESP32
Resolución	8bits	16bits	12bits
Modulación	Rampa	Delta sigma	Rampa
Muestras por segundo	-	860	-
Frecuencia de conversión	11.1KHz	-	-
Canales modo común	4	4	18
Canales modo diferencial	-	2	-
Comunicación	I2C	I2C	-
PGA	-	x2, x4, x8, x16	-
Alimentación	3 - 5v	2 - 5.5V	3.3 – 5V

Tabla 8: Selección del sensor conversor analógico digital.

Fuente: Elaboración propia.

Conforme a las características mencionadas en la tabla 8 se seleccionó el ADS1115, porque es un sensor que cumple las necesidades de este diseño, es decir, tiene alta precisión, tiene bajo ruido interno, tiene una ganancia programable para acondicionar los sensores antes del muestreo.

4.1.8 Tarjeta de Desarrollo

En la tabla 9 se muestra las características de posibles alternativas para la selección de la tarjeta de desarrollo que será usado para la captura de información, procesamiento de señales y encaminado de datos.

Parámetros	Tarjeta1	Tarjeta2	Tarjeta3
Código	RM46L852	ESP32	RM42L432
Marca	Texas Instruments	Espressif	Texas Instruments

Reloj	220MHz	240MHz	100MHz
Memoria Flash	1.25MB	4MB	384KB
Memoria RAM	192KB	520KB	32KB
Memoria EEPROM	64KB	448KB	16KB
Alimentación	3 – 3.6V	3.3-5V	3.3V
WIFI	no	2412 – 2448MHz	no
Núcleos	1	2	1

Tabla 9: Selección de la tarjeta de desarrollo.

Fuente: Elaboración propia.

Conforme a las características mencionadas en la tabla 9 se seleccionó el ESP32, porque es un dispositivo que cumple las necesidades para este diseño, es decir, está implementado especialmente para aplicaciones IOT, también tiene alta velocidad de procesamiento, tiene una buena capacidad de almacenaje de memoria flash importante para guardar el script del servidor, tiene dos núcleos donde el segundo se usará para el procesamiento del servidor y tiene Wifi integrado.

4.2 Arquitectura del diseño

En este apartado se describe el diagrama de bloque general del diseño, la cual se rige según el Paradigma Fog y Edge Computing.

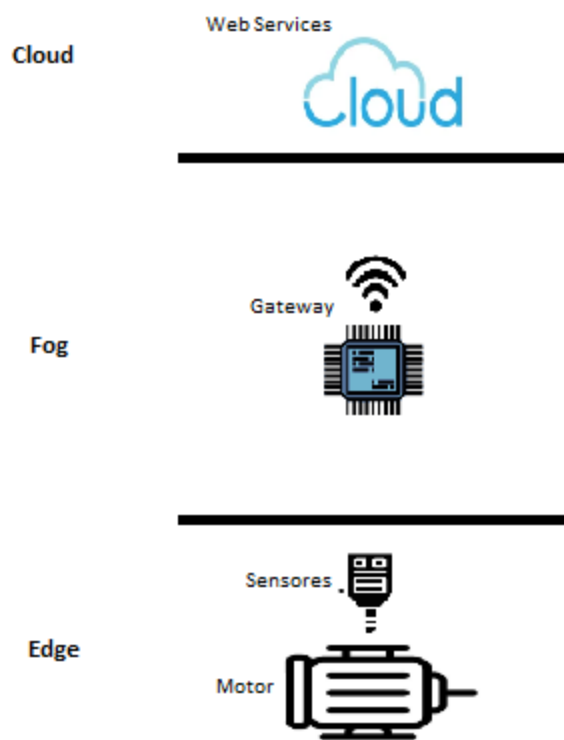


Figura 1: Diagrama de bloques general.

Fuente: Elaboración propia.

4.2.1 Capa Edge

Esta capa está contenida por los elementos encargados de realizar las mediciones de magnitudes físicas que rigen el funcionamiento del motor de inducción eléctrico, concretamente están conformados por los siguientes elementos de la tabla 10.

Elementos	Código	Principio de funcionamiento
Sensor de voltaje	Transformador	Inducción electromagnética
Sensor de corriente	SCT013	Inducción electromagnética
Sensor de temperatura	LM35	Transistor
Sensor de velocidad	AR-FC03	Óptico
Sensor de vibración	MPU-6050	Capacitivo
ADC	ADS1115	Delta sigma

Tabla 10: Elementos conformados por la capa Edge.

Fuente: Elaboración propia.

El ADS1115 se encarga del muestreo de los datos provenientes de los sensores de Voltaje, Corriente y temperatura, que se detallará este proceso en los siguientes apartados.

El protocolo de comunicación entre Edge y capa superior es el I2C que por defecto traen el ADS1115 y el MPU6050 excepto el AR-FC03 que se comunica de forma booleana.

4.2.2 Capa Fog

La capa Fog tiene como principal función, la recepción de la información proveniente de los sensores, también procesar lo necesario y encaminarla hacia la nube. El dispositivo seleccionado como Gateway ha sido el microcontrolador ESP32.

La comunicación con la capa Edge es través de tres nodos, como se muestra en la tabla 11.

Nodos	Dispositivos	Comunicación
Nodo 1	ADS1115	I2C
Nodo 2	MPU6050	I2C
Nodo 3	AR-FC03	booleana

Tabla 11: *Nodos.*

Fuente: *Elaboración propia.*

Para el envío de información hacia la nube, se ha utilizado el protocolo WebSocket, que es un protocolo estándar con una tecnología avanzada que hace posible establecer una comunicación full dúplex controlada por eventos, a su vez no necesita consultar al servidor para una respuesta, siendo persistente ente servidor y cliente. Como resultado se tiene una comunicación fluida y a gran velocidad. Se detalla la selección de este protocolo en el apartado Diseño de Software.

4.2.3 Capa Cloud

Esta capa también llamada Dashboard es la encargada de visualizar la información en forma de gráficas Chart.

El Dashboards cada vez que recibe una cadena de datos tipo Json, los visualiza inmediatamente en una gráfica de coordenadas amplitud versus tiempo de todas las variables planteadas en este estudio.

En la figura 2, se muestra la arquitectura del diseño con todos los componentes y protocolos de comunicación usados entre cada una de las capas.

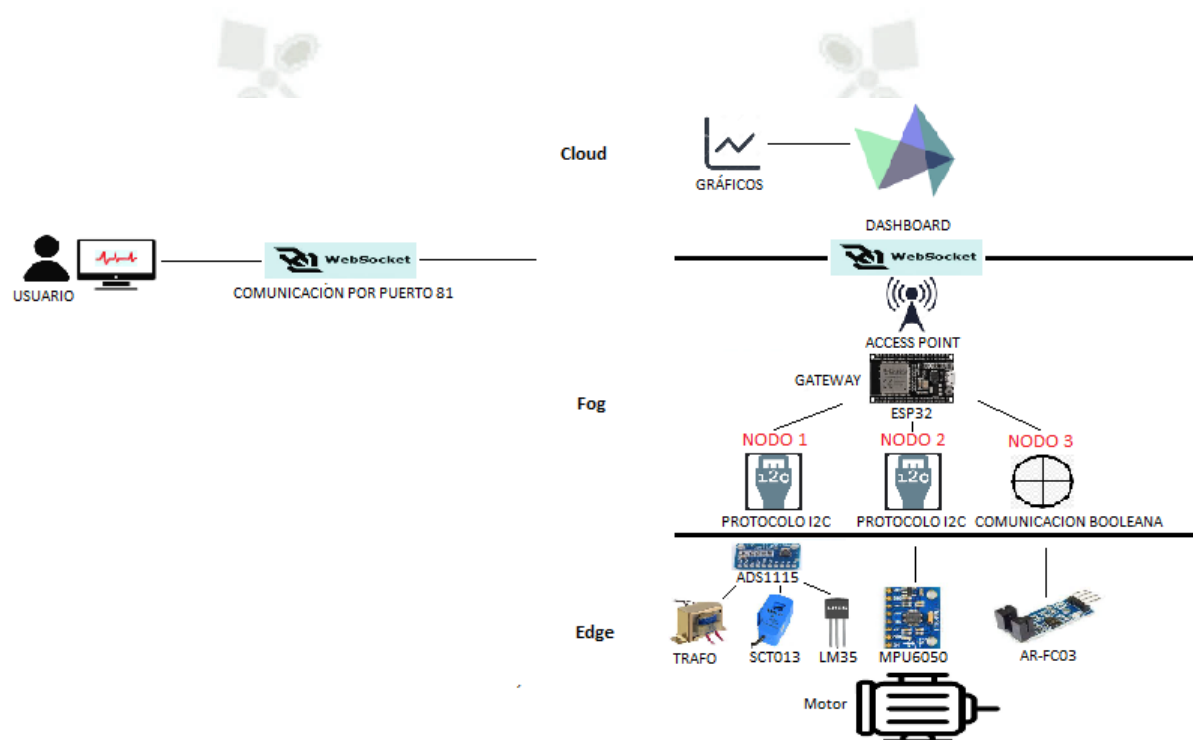


Figura 2: Arquitectura final.

Fuente: Elaboración propia.

- Este motor de inducción es trifásico de fábrica como se muestra en la tabla 11, pero funciona como monofásico porque está conectado internamente con un condensador de compensación. El cálculo de tensión de línea se hace con la siguiente ecuación.

$$\text{Voltaje de línea} = \text{Voltaje neutro} \times \sqrt{3} = 127V \times \sqrt{3} = 220V$$

En la figura 4 indica la placa del motor 220V.



Figura 4: Características del motor Pretul.

Fuente: Elaboración propia.

- Los valores típicos de vibración de un motor de 0.5HP sobre una superficie rígida a plena carga es de 0.20mm/s de velocidad aproximadamente; que a su vez, para tener una buena medición se recomienda elegir un acelerómetro cuyas características principales tenga una sensibilidad de 4g como mínimo, White (1990).
- en nuestro caso se está utilizando una electrobomba de agua a plena carga, que tiene como característica principal de vibración nominal 0.15mm/s de velocidad.

4.4.2 Elementos del Arranque simple del motor

El dimensionamiento de los elementos para el arranque del motor de 0.5 HP con una configuración simple, implica la selección rigurosa de un interruptor termomagnético y un contactor. Estos componentes deben ser adecuados para la corriente de operación del motor y deben proporcionar una protección efectiva contra las sobrecargas y cortocircuitos.

4.4.2.1 Contactores.

Para el dimensionamiento de los contactores en el circuito de arranque de un motor de 0.5 HP con una configuración simple, es importante tener en cuenta la corriente de conexión del motor y el tipo de carga que se está manejando. En este caso, se ha enfocado en la corriente nominal del motor 1.5A, la cual es un dato que se encuentra en la placa de características del motor.

En el contexto de este proyecto, donde se utiliza un arranque directo, la corriente de arranque puede incrementar la corriente nominal hasta cinco o siete veces. Esto significa que tanto la corriente nominal como el pico de corriente durante el arranque son factores determinantes para el dimensionamiento adecuado de los contactores.

De acuerdo con la normativa IEC 947-4, que estipula las categorías de uso normalizado para contactores, la categoría AC-3 es la más adecuada para motores trifásicos tipo jaula de ardilla. En esta categoría, al cerrar el contactor, este debe ser capaz de sostener una corriente de 5 a 7 veces la corriente nominal del motor, y al abrir, interrumpir la corriente nominal absorbida por el motor.

Considerando las opciones disponibles en el mercado local, se ha optado por el contactor NCHB de la marca CHNT, cuyas especificaciones se ajustan a las necesidades de este proyecto. Las características de este contactor se resumen en la siguiente tabla 13:

Característica del contactor	Valor
Marca	CHNT
Serie	NCHB
Número de contactos NO	2
Voltaje máximo AC	500V
Corriente de operación AC3	20A

Tabla 13: Características del contactor.

Fuente: Elaboración propia.

Las especificaciones técnicas de este contactor indican que la corriente de operación AC3 es de 20 A, lo cual es superior a la corriente nominal del motor (1.5 A). Esto asegura que el contactor pueda soportar sin problemas los picos de corriente durante el arranque, incluso en las condiciones más exigentes. Por esta razón, se ha seleccionado este contactor.



Figura 5: Contactor.

Fuente: Elaboración propia.

4.4.2.2 Llave termomagnética.

El interruptor termomagnético es esencial para proteger el circuito y el motor contra sobrecorrientes y cortocircuitos, permitiendo una fácil reposición del paso de la corriente eléctrica sin la necesidad de cambiar fusibles tras cada actuación.

Para el dimensionamiento adecuado de este dispositivo, se considera un factor multiplicador de 12 veces la corriente nominal del motor. Esto es especialmente relevante en el caso de motores trifásicos, aunque en el caso de un motor de 0.5 HP, que generalmente es monofásico, este factor sigue siendo una buena referencia para garantizar una protección adecuada. Se muestra en la tabla 14 las características eléctricas de la llave termomagnética seleccionada.

Característica	Valor
Marca	CHTN
Serie	NXB-63
Capacidad de interrupción	5 KA
Voltaje máximo de servicio	400 V
Vida útil	1000 maniobras mecánicas
Corriente	16A

Tabla 14: Características de la llave termomagnética.

Fuente: Elaboración propia.

Se ha optado por un termomagnético de 16 A, siendo esta la menor corriente de disparo disponible en el mercado local. Esta selección proporciona un margen de seguridad adecuado, teniendo en cuenta la corriente nominal del motor 1.5A y los posibles picos de corriente durante el arranque, sin ser excesivamente elevada para evitar disparos innecesarios. La capacidad de interrupción y las normas que cumple el dispositivo aseguran su fiabilidad y adecuación a las necesidades del prototipo.



Figura 6: *Llave termomagnética.*

Fuente: *Elaboración propia.*

4.4.2.3 *Luces de Indicación.*

Las luces de indicación son un componente esencial en el circuito de control y monitoreo de motores de inducción eléctricos, particularmente en plataformas IOT para la Industria 4.0. Estas luces proporcionan una forma visual e inmediata de identificar el estado operativo del motor y de cualquier sistema asociado. Se detallan los aspectos más importantes de las luces de indicación:

Tipos de Indicaciones: Comúnmente, se utilizan luces de diferentes colores para representar distintos estados. Por ejemplo, una luz verde indica que el motor está funcionando correctamente, una luz ámbar puede señalar un fallo o parada de emergencia, y una luz amarilla podría utilizarse para advertencias o estados en espera, una luz roja indica que el circuito está energizado.

4.4.3 *Diseño del Circuito del Arranque Simple del Motor*

Un sistema del arranque simple para un motor de inducción de 0.5 HP se basa en la utilización de un contactor para el control de arranque y parada del motor, junto con un interruptor termomagnético para su protección. Este sistema es efectivo para aplicaciones donde los requerimientos de arranque del motor son básicos y no se

necesitan reducciones de corriente de arranque que implicarían configuraciones más complejas. La configuración del esquema para el arranque simple del motor, se muestra en la figura 7.

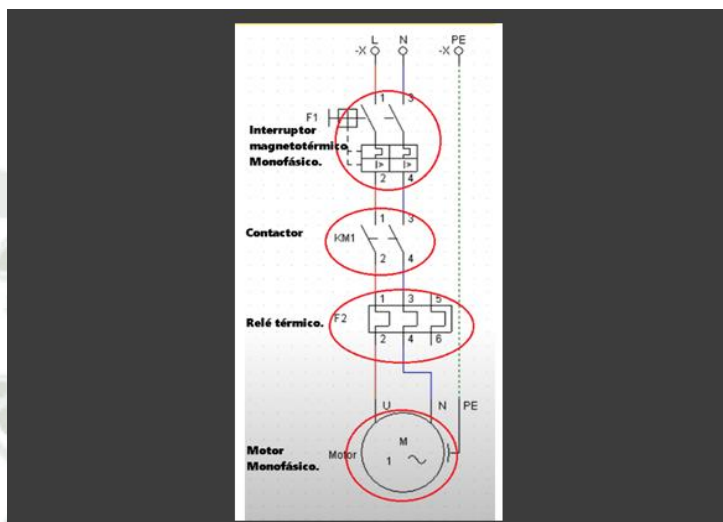


Figura 7: Esquema del Arranque simple de motor.

Fuente: Automatización CA De, Simu, Electricidad, Motor AC.

La implementación del arranque simple para el motor de inducción de 0.5PH, se muestra en la figura 8.



Figura 8: Instalación del arranque del motor.

Fuente: Elaboración propia.

4.5 Diseño y Dimensionamiento de Elementos de Medición

En el diseño y dimensionamiento de la etapa de medición de parámetros de funcionamiento del motor en un sistema IOT implica integrar sensores específicos para medir variables críticas como voltaje, corriente, temperatura, vibración, RPM. Estos sensores recopilan datos precisos en tiempo real, esenciales para el análisis detallado del rendimiento y la eficiencia operativa del motor en entornos de la Industria 4.0

4.5.1 Sensor de voltaje

El sensor de voltaje es un transformador convencional de ferrita que está seleccionado por su aislamiento inductivo proporcionando seguridad, de que la energía eléctrica no entra directamente de la red pública al circuito electrónico. También se seleccionó porque no necesita mayor acondicionamiento de señal a lo previsto para los demás sensores y por último se seleccionó este transformador por la señal de salida que tiene (con muy bajo ruido comparado a otros sensores) y fácil manejo.

Este tipo de sensor es fundamental en sistemas de monitoreo, especialmente en aplicaciones IOT, donde la confirmación de la presencia y medición de voltaje es crítico para la operación segura y eficiente de equipos y maquinarias, como los motores de inducción. Se muestra en la figura 9 el transformador seleccionado.

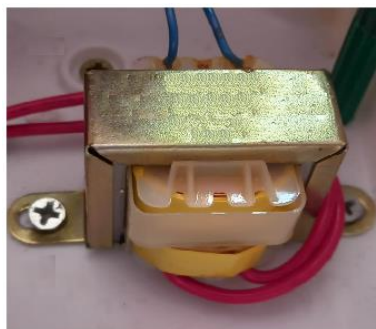


Figura 9: Transformador, sensor de voltaje.

Fuente: *Imagen propia.*

4.5.1.1 Características y Acondicionamiento del Sensor de Voltaje.

En este apartado describiremos el transformador y su acondicionamiento para ser usado como sensor de voltaje.

El funcionamiento del transformador se basa en la ley de Faraday y las propiedades ferromagnéticas del núcleo de hierro, para este diseño no consideraremos los cálculos de potencia ya que la salida del transformador prácticamente estará sin carga porque se conectará al ADC que tiene una alta impedancia de entrada en orden de los $M\Omega$.

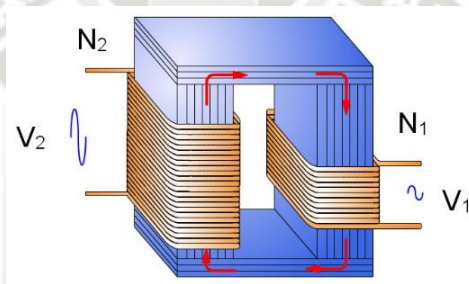


Figura 10: *Diagrama del transformador, sensor de voltaje.*

Fuente: *Juntadeandalucia.*

Cálculo del número de vueltas del primario, usamos la ecuación de Faraday según las variables de la figura 10.

Ley de Faraday

$$\frac{V_2}{V_1} = \frac{N_2}{N_1}$$

Donde:

V_2 , V_1 = voltaje del secundario y del primario respectivamente.

N_2 , N_1 = número de vueltas del secundario y del primario respectivamente.

Es importante mencionar que se debe trabajar los voltajes en sus valores máximos es decir valores pico pico. Para hacer los cálculos con la señal completa del secundario.

- Cálculo de V1

El voltaje pico pico (V_{pp}) de la red eléctrica es V1.

$$V1 = V_{pp} = 220 * \sqrt{2} * 2 = 622.25V$$

- Cálculo de V2

Para hacer este cálculo tomamos como referencia la mitad del ancho de banda del ADC, esto se define por default igual al voltaje de alimentación, es decir será igual a 3.3V.

Posteriormente para hacer un buen mapeo se debe elegir la mitad, es decir 1.65v.

$$V2 = 1.65V_{pp} = \frac{1.65}{\sqrt{2} \times 2} = 0.5834V_{RMS}$$

- Cálculo de N1

Para el cálculo de N1 se dimensionó el calibre 41 del alambre esmaltado en base a la tabla 15 por ser comercial.

TABLA DE ALAMBRE ESMALTADO

1	2	3	4	5	6	7
31	0.2268	0.040	39.8	0.36	425.0	0.11
32	0.2019	0.032	44.5	0.28	531.2	0.09
33	0.1798	0.0254	50.0	0.23	669.3	0.072
34	0.1601	0.0201	56.0	0.18	845.8	0.057
35	0.1426	0.0159	62.3	0.14	1069.0	0.045
36	0.1270	0.0127	69.0	0.10	1338.0	0.036
37	0.1131	0.0100	78.0	0.089	1700.0	0.028
38	0.1007	0.0079	82.3	0.070	2152.0	0.022
39	0.0897	0.0063	97.5	0.056	2696.0	0.017
40	0.0799	0.0050	111.0	0.044	3400.0	0.014
41	0.0711	0.0040	126.8	0.035	4250.0	0.011
42	0.0633	0.0032	138.9	0.028	5312.0	0.009
43	0.0564	0.0025	156.4	0.022	6800.0	0.007
44	0.0503	0.0020	169.7	0.018	8500.0	0.005

1 - Numero AWG (American Wire Gauge)

2 - Diametro en milímetros

3 - Seccion en milímetros

4 - Numero de espiras por centímetros

5 - Kg por kilometro

6 - Resistencia en ohmios por kilometro

7 - Capacidad de corriente en amperes

Tabla 15: *Tabla de alambres esmaltados.*

Fuente: *Scribd/Tabla-de-Alambres-Esmaltados.*

Fuente Scribd/Tabla-de-Alambres-Esmaltados

Seguidamente para este calibre típicamente se determina 14 vueltas por voltio para un núcleo de 20x20mm y de 10mm de espesor.

$$N1 = 14 \times 220 = 3080 \text{ vueltas}$$

Finalmente se ha calculado N2 usando la ecuación de Faraday

$$N2 = \frac{V2 \times N1}{V1} = \frac{0.5834 \times 3080}{220} = 8 \text{ vueltas}$$

Con los datos calculados se implementó el transformador dándonos un resultado muy bueno, $V2 = 0.564$ con respecto al esperado $0.58V$ como se muestra en la figura 11, con el motor en funcionamiento.

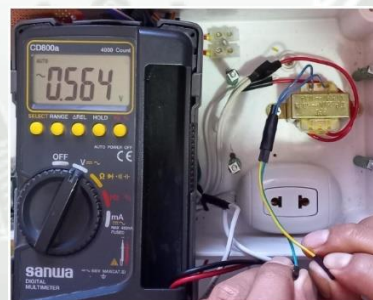


Figura 11: Lectura del transformador.

Fuente: Elaboración propia

Para el acondicionamiento de señal se conecta la salida del secundario del transformador en serie a una fuente de tensión regulada de 1.5VDC, este proceso es necesario para quitar la señal negativa que tiene la corriente alterna. Este valor de la fuente se basa en el ancho de banda de entrada del ADS1115.

4.5.1.2 Integración del Sensor de Voltaje con el ADS1115.

La salida del sensor se conecta al pin A0 del conversor análogo digital ADS1115 como se muestra en la figura 12.

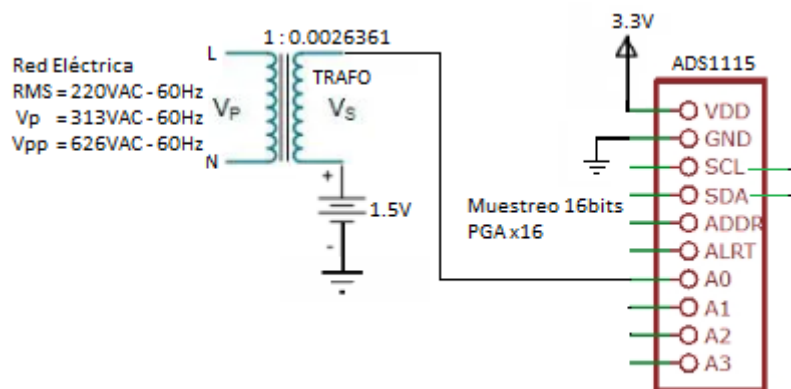


Figura 12: Diagrama circuital del sensor de voltaje.

Fuente: Elaboración propia.

4.5.2 Sensor de Corriente

El SCT013 es un sensor de corriente tipo transformador que tiene como principio de funcionamiento la inducción electromagnética que permite la medición de corrientes eléctricas dentro de su núcleo de ferrita, con un rango de operación que varía según el modelo (30A, 100A, etc). Este sensor se caracteriza por su capacidad de proporcionar una salida de voltaje analógico con una no linealidad de 3% y proporcional a la corriente detectada a través de su núcleo. La principal característica para lo cual se seleccionó este sensor es la señal con muy poco ruido que tiene a la hora de medir corriente.



Figura 13: Sensor SCT013.

Fuente: Imagen propia.

4.5.2.1 Características y Acondicionamiento de Señal del Sensor de Corriente.

En este apartado describiremos el sensor de corriente y su acondicionamiento.

- Como características más importantes según hoja de datos de fabricante.
 - ❖ Tamaño de apertura 13mm x 13mm, se refiere al área donde se coloca el cable conductor que se quiere medir.
 - ❖ Este sensor mide hasta 30A de forma efectiva como nos muestra en la figura 16.

Table of technical parameter:

Model	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output type	0-50mA	0-1V	0-1V	0-1V	0-1V
Model	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-000V
Input current	0-25A	0-30A	0-50A	0-60A	0-100A
Output type	0-1V	0-1V	0-1V	0-1V	0-1V

Tabla 16: Hoja de datos del SCT013.

Fuente: Beijing Yaohuadechang Electronic.:

Para este diseño nos basamos en la corriente nominal de motor que es de 1.5A como se parecía en la figura 14, pero la corriente de arranque del motor por ser una carga reactiva-inductiva se considera típicamente 6 veces más, de manera que definimos el rango 0 - 10A.

$$I = I_{nominal} \times 6 = 1.5A \times 6 = 9A$$



Figura 14: Placa del motor, 1.5A.

Fuente: Elaboración propia.

- ❖ Este sensor tiene una equivalencia de corriente voltaje de 1A/30mV. Para tener el voltaje de salida proporcional a la corriente del motor, se hizo el cálculo tomando en cuenta la corriente real que consume el motor $I = 900\text{mA}$

$$V = I \times 0.03 = 900\text{mA} \times 0.03 = 27\text{mV}$$

Con este valor calculado confirmamos el correcto funcionamiento del sensor, comparando con la medición que se hizo y se muestra en la figura 15.



Figura 15: Medición de corriente.

Fuente: Elaboración propia.

- No hizo acondicionamiento de señal en la parte del hardware es decir se conectó directamente al ADS1115, el acondicionamiento de señal se ha hecho en la etapa de diseño de software haciendo uso del módulo PGA del ADS1115. Se detalla este proceso en el capítulo V.

4.5.2.2 Integración del Sensor de Corriente con el ADS1115.

La salida del sensor se conecta a los pines A1 y A2 respectivamente del conversor analógico digital ADS1115 como se muestra en la figura 16.

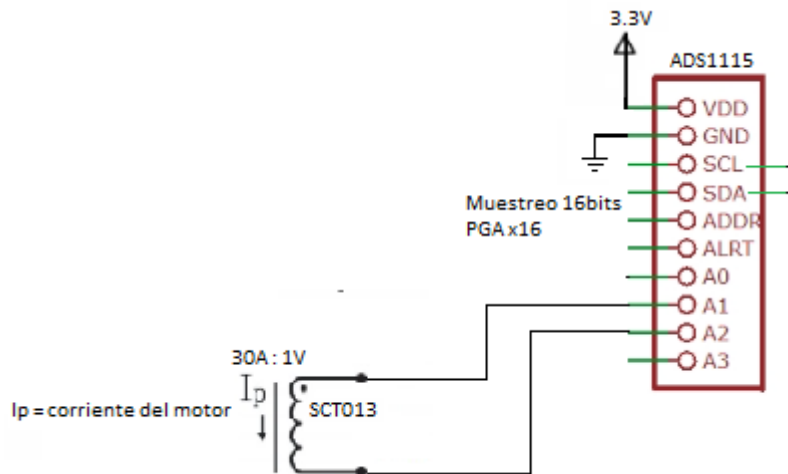


Figura 16: Circuito del sensor de corriente.

Fuente: Elaboración propia.

4.5.3 Sensor de Temperatura

El LM35 es un sensor de temperatura analógico que ofrece una alta precisión y una salida lineal. Es ampliamente utilizado en diversas aplicaciones debido a su fácil integración con microcontroladores y sistemas de monitoreo. Este sensor proporciona una salida de voltaje proporcional a la temperatura medida, con una precisión calibrada de $\pm 0.5^\circ\text{C}$ y un rango de medición que va desde -55°C hasta 150°C . Por cada grado centígrado, el LM35 proporciona una variación de 10 mV en su salida.

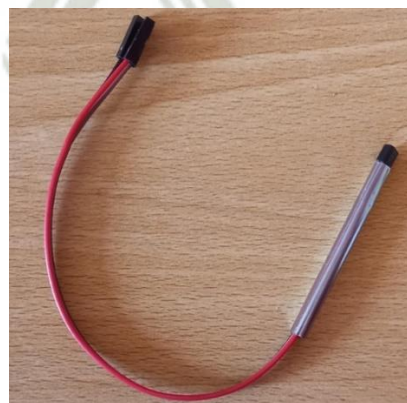


Figura 17: Sensor de corriente.

Fuente: Elaboración propia.

4.5.3.1 Características y Acondicionamiento de Señal del Sensor de Temperatura.

En este apartado describiremos el sensor de temperatura y su acondicionamiento en base a la hoja de datos del fabricante.

- Calibrado directamente en grados centígrados.
- Voltaje de alimentación es de 4V a 30V sin variar la salida.
- Consumo de corriente menos de 60 μ A.
- Bajo auto calentamiento, menos de 0.08°C
- Salida de baja impedancia 0.1 Ω para una carga de 1mA
- Su distribución de pines se define Pin 1 como alimentación, pin 2 Salida y pin 3 GND como se aprecia en la figura 18.

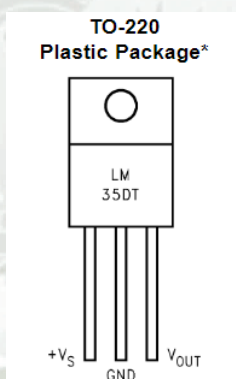


Figura 18: Diagrama de distribución de pines del sensor LM35

Fuente: Datasheet ML35.

- Rango de temperaturas de -55 a 150°C. Para este diseño utilizaremos el rango de temperaturas de 0 a 150°C. Este rango está elegido en base a la temperatura de trabajo de la mayoría de motores. La temperatura que aguantan el aislamiento del devanado del motor está determinada por la temperatura máxima que pueda soportar antes de la ruptura

completa. Típicamente esta temperatura está definida alrededor de 150°C como podemos apreciar en la figura 19

Class	Maximum Temp
Y	90°C
A	105°C
E	120°C
B	130°C
F	155°C
H	180°C

Figura 19: *Temperaturas de operación de motores.*

Fuente: CBM CONNECT, sobrecalentamiento de motores eléctricos

- Presición de 0.5°C (a 25°C), se refiere a que está presición es efectiva a 25°C a medida se va alejando este valor su precisión deciendo hasta 0.8°C.
- La relación temperatura voltaje es de 1°C/10mV. Se refiere, a medida que se incrementa 1 grado de temperatura, tambien se incrementa linealmente y directamente proporcional 10 milivoltios en la salida del sensor. Según fabricante
 - 150°C = 1500mV
 - 25°C = 250mV

Comprobamos estos datos midiendo temperatura ambiente directamente con multímetro y con un termómetro digital como se muestra en la figura 20.

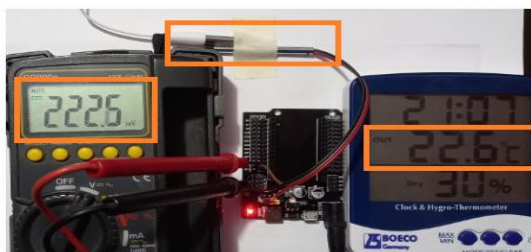


Figura 20: *Medición de temperatura.*

Fuente: *Elaboración propia.*

- #### 4.5.3.2 Integración del Sensor de temperatura con el ADS1115.

Fuente: *Elaboración propia.*

El MPU6050 es un dispositivo de medición inercial que tiene sensor de vibración y giroscopio, para este diseño no se utilizó el giroscopio. Este sensor tiene múltiples ADC de 16 bits delta sigma diseñados para el muestreo de señal de forma independiente para cada variable analógica de vibración. Para transferir datos utiliza la comunicación I2C y la comunicación

Serial SPI, en este diseño se utilizó la comunicación I2C porque es más eficiente y utiliza menos recursos del microcontrolador, también para ahorrar buses de comunicación ya que también se usará otro ADC que tiene comunicación I2C por lo cual se conectara en paralelo ambos (nodo1 y nodo 2).

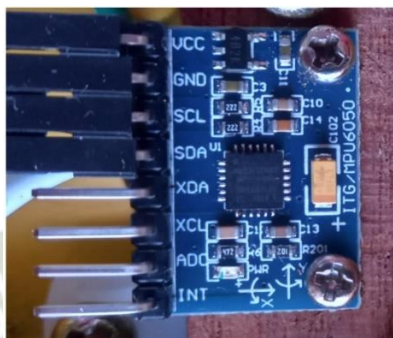


Figura 22: Sensor de MPU6050.

Fuente: Elaboración propia.

4.5.4.1 Características y Acondicionamiento de Señal del Sensor de Vibración.

En este apartado describiremos el sensor de vibración y su acondicionamiento en base a la hoja de datos del fabricante.

- Principio de funcionamiento capacitivo, que nos garantiza la alta sensibilidad que tiene.
- Alimentación es de 3 a 5V, para este diseño se alimenta al acelerómetro con 3.3V.
- Acelerómetro de triple eje con salida digital, este sensor divide la aceleración en tres planos o ejes, eje X, Y, Z. Para un estudio más avanzado y detallado se utilizan estos planos independientes. En este estudio se utilizará la vibración convencional que llamaremos simplemente vibración.
- Los ADC de 16bits delta sigma integrados permiten el muestreo simultaneo de acelerómetros de forma independiente sin requerir multiplexor, se refiere a que los tres ejes tienen cada uno su propio ADC.

- Sensibilidad programable de 2g, 4g, 8g, 16g. Para este diseño se utilizó la más alta sensibilidad que es 2g, en el capítulo V se detalla este proceso.
- Comunicación I2C y SPI, para este diseño se utilizó la comunicación I2C con dirección de trabajo de 0x68. Este sensor trae hasta 4 direcciones posibles de 0x68 a 0x6B. por default está configurado con la dirección 0x68.
- Para este diseño no se hizo acondicionamiento de señal. Todo lo concerniente al mapeo, calibrado programación se detalla en el capítulo V.

4.5.4.2 Integración del Sensor de Vibración con el ESP32.

La salida del sensor se conecta los pines SCL (22) y SDA (21) del microcontrolador ESP32 como se muestra en la figura 23.

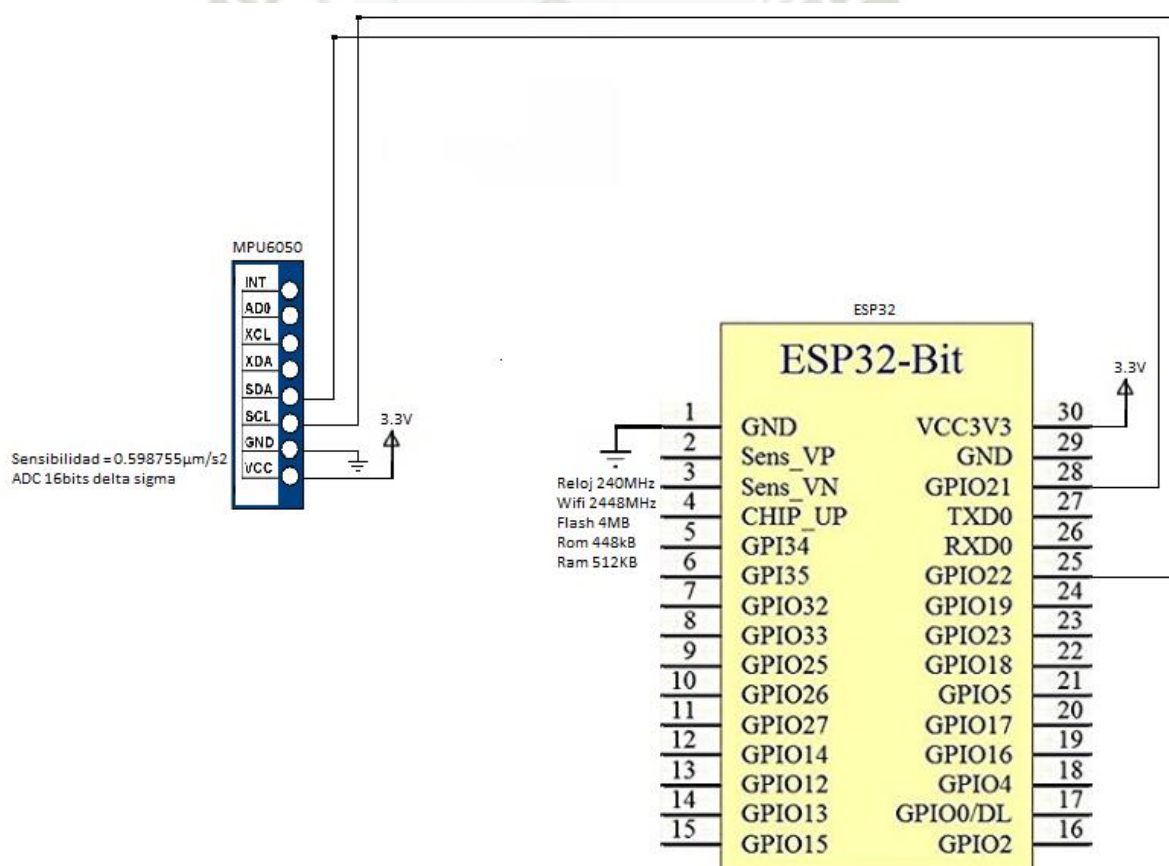


Figura 23: Circuito del ESP32.

Fuente: Elaboración propia.

4.5.5 Sensor de RPM

El funcionamiento del AR-FC03 se basa en el principio de detección de movimiento rotacional. Este sensor generalmente emplea un método de detección óptica o magnética para percibir la rotación. En la configuración óptica, se utiliza un disco con marcas o ranuras que interrumpen un haz de luz entre un emisor y un receptor óptico, generando así pulsos eléctricos que son contabilizados.



Figura 24: Sensor de AR-FC03 montado en el motor.

Fuente: Elaboración propia.

4.5.5.1 Características y Acondicionamiento de Señal del Sensor de RPM.

En este apartado describiremos el sensor de velocidad y su acondicionamiento en base a la hoja de datos del fabricante.

- Diagrama interno funcional

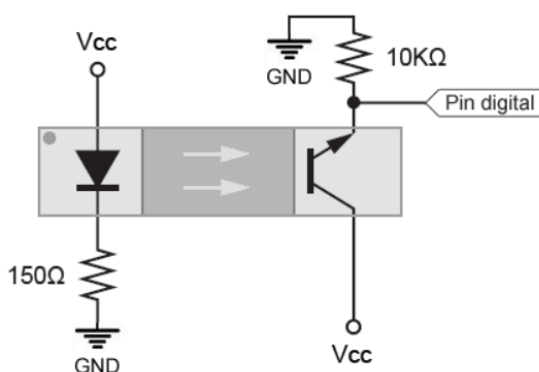


Figura 25: Diagrama interno del AR-FC03.

Fuente: *Tesla Electronic.*

- **Diagrama de distribución de pines**



Figura 26: *Sensor AR-FC03.*

Fuente: *Tesla Electronic.*

- ❖ VCC: Alimentación del módulo de 3,3V a 12V.
- ❖ GND: Tierra.
- ❖ D0: Señal digital de los pulsos de salida.
- ❖ A0: Señal analógica de los pulsos de salida. Señal de salida en tiempo real. (normalmente no se usa).

Las medidas técnicas de la ranura e indicadores se muestran en a tabla 17.

Medidas técnicas de la ranura e indicadores

PRINCIPALES CARACTERÍSTICAS TÉCNICAS
Dimensiones: 32 x 14 x 7mm
La ranura de lectura del sensor tiene un ancho de 5mm
Dos salidas, una digital y otra analógica
LED indicador de alimentación

Led indicador de los pulsos de salida del pin D0

Tabla 17: Medidas técnicas de la ranura.

Fuente: Elaboración propia.

- Para este diseño no se hizo acondicionamiento de señal. Todo lo concerniente al mapeo, calibrado programación se detalla en el capítulo V.

4.5.5.2 Integración del Sensor de RPM con el ESP32.

La salida del sensor se conecta al pin 35 del microcontrolador ESP32 como se muestra en la figura 27.

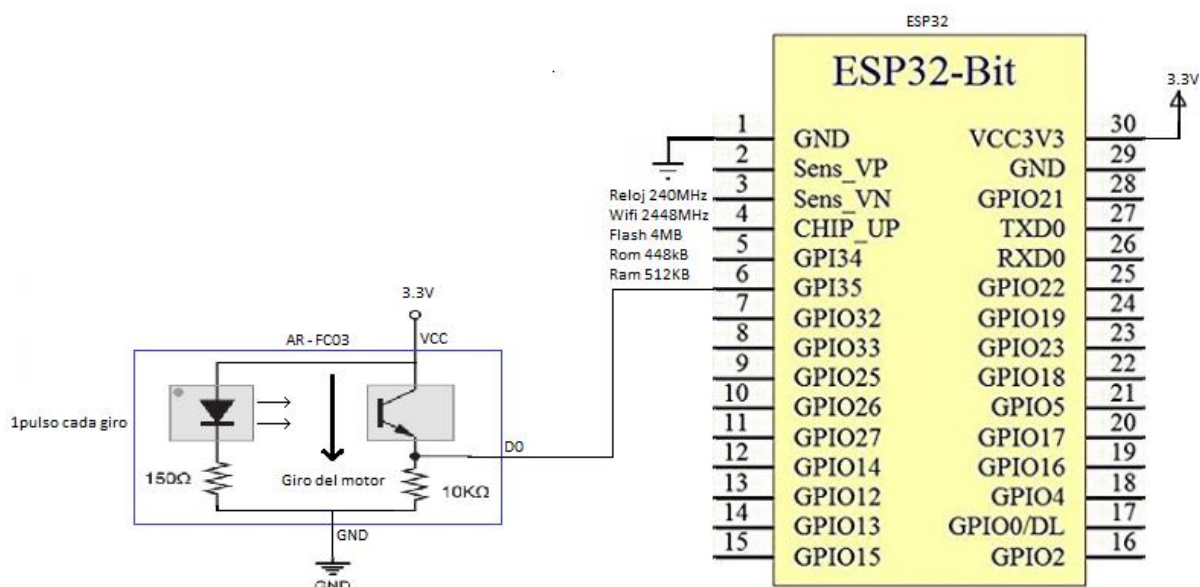


Figura 27: Circuito del AR-FC03.

Fuente: Elaboración propia.

4.6 Descripción del ADS1115

El ADS1115 de Texas Instruments es un conversor analógico digital ADC y amplificador de ganancia programable PGA, en la figura podemos apreciar el diagrama de bloques interno de este dispositivo electrónico según la hoja de datos del fabricante.

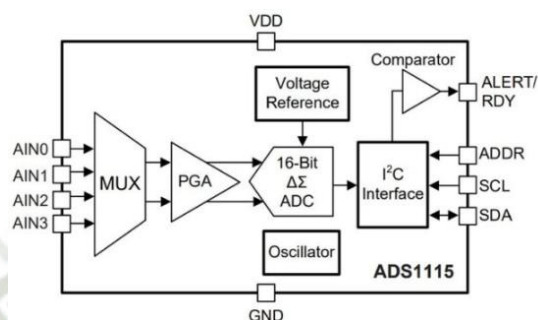


Figura 28: Diagrama interno del ADS1115.

Fuente: Hoja de datos del fabricante.

Como características principales está descrito en la tabla 18.

Características ADS1115	
Marca	Texas Instruments
Resolución	16bits
Modulación	Delta sigma
Muestras por segundo	860
Canales modo común	4
Canales modo diferencial	2
Comunicación	I2C
PGA	x2, x4, x8, x16
Alimentación	2 – 5.5V

Tabla 29: Características del ADS1115.

Fuente: Elaboración propia.

4.7 Descripción del ESP32

El ESP 32 forma parte de una serie de Sistem on Chip (SoC) y módulos de bajo costo y energía, fabricado por Espressif y creado por Espressif Systems. Este dispositivo es la evolución del ESP8266 con mejores características. Emplea un microprocesador de 32bits Xtensa LX6 de doble núcleo.

Sus características más notables se describen de la siguiente forma.

- Procesador
 - ❖ Microprocesador 32bit Xtensa LX6 doble núcleo 240MHz
- Memorias
 - ❖ RAM 520KB
 - ❖ EEPROM 448KB
 - ❖ Flash 4MB
- Conectividad
 - ❖ Wifi 2448 MHZ, 802.11 b/g/n
 - ❖ Bluetooth v4.2 BR/EDR/BLE
- Algunos periféricos
 - ❖ ADC 12bits 18 canales de rampa
 - ❖ 2 interfaces de comunicación I2C
 - ❖ 32 ranuras de interrupción por núcleo
 - ❖ Pre amplificador analógico de ultra baja potencia
 - ❖ 25 salidas y/o entradas digitales
 - ❖ 2 interfaces de audio digital I2S
 - ❖ 3 interfaces de comunicación serial UART
 - ❖ DAC convertidores digital analógicos de 8bits con 2canales de entrada

4.7.1 Tarjeta de Desarrollo DEVKIT DEVKIT V1

El ESP32 está implementado en diferentes tarjetas de desarrollo. Para este estudio se utilizó el DEVKIT WROOM WIFI que tiene incorporado el potente modulo ESP-WROOM-32 que integra una antena WIFI, como lo podemos apreciar en la figura 30.



Figura 29: Tarjeta de desarrollo ESP32 DEVKIT V1.

Fuente: Imagen propia.

- Esta tarjeta está diseñada especialmente para aplicaciones IOT. Tiene incorporado una antena que le permite comunicarse mediante WIFI.
- Incorpora un regulador de voltaje de código 1117 3.3 de 3.3 voltios de 800mA como se parecía en la figura 32 Para alimentar al módulo ESP-WROOM-32 y a los demás periféricos externos que se han integrado a este dispositivo.



Figura 30: regulador de voltaje 1117C/33-C238.

Fuente: Imagen propia.

- Para este diseño se utilizará el expansor ESP32-SHIELD como se aprecia en la figura 31, Este módulo nos facilita la conexión de pines de los sensores al DEVKIT y también nos proporciona la corriente necesaria para alimentar a los demás elementos de nuestro diseño. Tiene un regulador de voltaje de código 1117C 5.0 que se alimenta de 6.5 a 16V, con salida de 5V - 800mA.

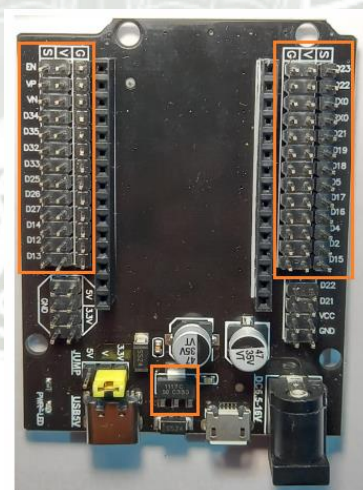


Figura 31: Módulo SHIELD ESP32.

Fuente: Imagen propia.

- El consumo total de corriente de todo el sistema es 173.81mA, como lo describimos en los siguientes cálculos.

- 0 = corriente del sensor de voltaje, no consume corriente.
- 0 = corriente del sensor de corriente, no consume corriente.
- 0.06 = corriente del sensor de temperatura, 60 μ A
- 3.6 = corriente del sensor de vibración, 3.6mA
- 90 = corriente de consume del sensor de velocidad, 90mA
- 0.15 = corriente de consume del ADS, 150 μ A
- 80 = corriente de consume del Esp32 DEVKIT, 80mA

$$I_{total} = 0 + 0 + 0.06 + 3.6 + 90 + 0.15 + 80 = 173.81mA$$

4.8 Diagrama Circuital del Diseño

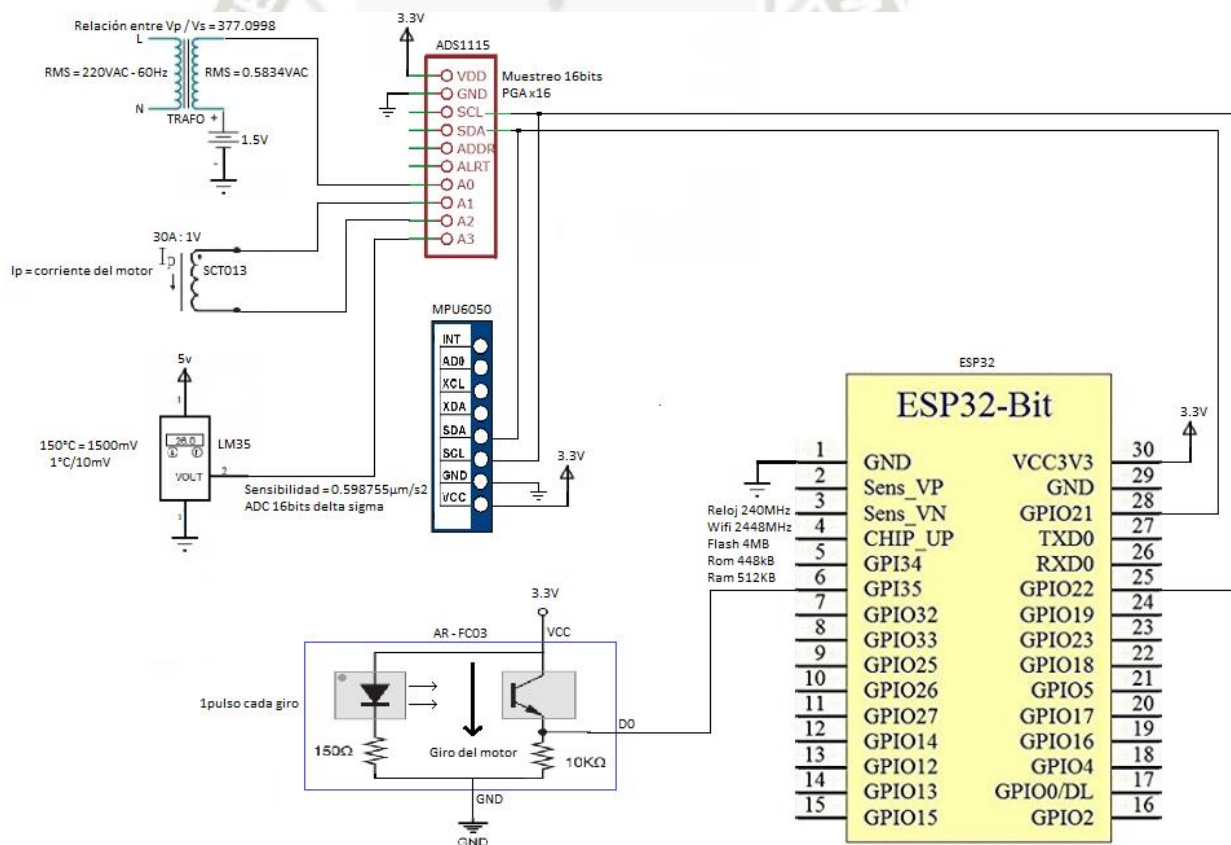


Figura 32: Diagrama circuital del diseño.

Fuente: Elaboración propia.

4.9 Especificaciones Técnicas

Especificaciones de elementos eléctricos	Especificaciones de los sensores
- Motor de inducción monofásico $V = 220\text{VAC}$ $\text{RPM} = 4500\text{r/min}$ $P = 0.5\text{HP}$ $I = 1.5\text{A}$ - Llave termomagnética $I = 16\text{A}$ - Contactador $I = 20\text{A}$	- Sensor de Voltaje trafo Relación entre $V_p/V_s = 377.0998$ - Sensor de corriente SCT-013 Relación $I/V = 1\text{A}/33\text{mV}$ - Sensor de Temperatura LM35 Relación $T/V = 1^\circ\text{C}/10\text{mV}$ - Sensor de Vibración MPU6050 Sensibilidad = 2g - Sensor de velocidad AR-FC03 Emisor = LED Receptor = fototransistor
Especificaciones del Interfaz	Especificaciones del microcontrolador
ADC ADS1115 Resolución = 16bits Modulación = delta sigma Muestreo = 860 muestras/s PGA = X16 Comunicación = I2C Canal independiente = usados 2 Canal diferencial = usados 1	ESP32 Marca = Espressif Procesador = 32bit LX6 Reloj = 240MHZ Flash = 4MB RAM = 520KB EEPROM = 448KB Wifi = 2448 MHZ Núcleos = 2 Alimentación = 3.3V

Tabla 18: Especificaciones técnicas.

Fuente: Elaboración propia.



CAPITULO V

DISEÑO DE SOFTWARE

En este capítulo se implementa toda la programación requerida para que el diseño sea dinámico y funcional.

El Gateway ESP32 se desarrolla en muchos lenguajes de programación como MicroPython, Python, C++, JavaScript, Lua, Node MCU, FreeRTOS, etc. A su vez para programar el ESP32 se necesita un entorno de programación o IDE que permita escribir, compilar, cargar el código como el PlatformIO IDE, Espressif IoT, Arduino IDE. Este último sin duda es el más famoso avanzado y más completo. Para este diseño se ha elegido el lenguaje de programación C++ y el entorno Arduino IDE.

Por otro lado, el diseño y programación del servidor (Cloud) también se aloja en el ESP32 concretamente en su memoria Flash, El Cloud se desarrolla en muchos lenguajes de programación, siendo los más utilizados PHP, Python, Java, Node.js, HTML, CSS, JavaScript. Siendo estos tres últimos los más demandados, accesibles, de código libre y más avanzados para desarrollos web. Para este diseño se ha elegido HTML, CSS, JavaScript y como editor de texto se ha utilizado el visual studio code.

5.1 Selección del Protocolo de Comunicación

En este apartado se ha hecho una selección de protocolos de comunicación Wifi que tenga la capacidad de muestreo menor a 50ms.

Para esta elección se ha elegido tres candidatos que son los más usados en aplicaciones IOT.

- Protocolo MQTT.

Este protocolo es el más popular y usado en aplicaciones IOT sobre todo en proyectos de Domótica, IOT industrial, gestión de energía, etc.

- Protocolo HTTP.

Este protocolo es el más compatible de todos con las diferentes infraestructuras web, por lo tanto, es aplicado a los proyectos que se puedan beneficiar de esta comunicación tan extendida como cuadros de mando, api para dispositivos IOT, actualizaciones de firmware, integración de voz y móvil, etc.

- Protocolo CoAP

Este protocolo es muy ligero orientado a dispositivos con recursos limitados o necesiten bajo consumo de energía, aplicado en su mayoría para proyectos de ciudades inteligentes, Agricultura, vigilancia medio ambiental, domótica, etc.

Se muestra en la tabla 19 las principales características de dichos protocolos de comunicación.

MQTT vs. CoAP vs. HTTP

Aspecto	MQTT	CoAP	HTTP
Modelo de comunicación	Publicar-Suscribir	Solicitud-Respuesta	Solicitud-Respuesta
Sobrecarga y eficiencia de los mensajes	Baja sobrecarga, eficiente	Baja sobrecarga, eficiente	Mayor sobrecarga, menos eficiente
Escalabilidad	Altamente escalable	Escalable	Escalable
Seguridad	Admite funciones de seguridad	Admite funciones de seguridad	Admite funciones de seguridad
Fiabilidad y calidad de servicio	Admite niveles de QoS	Admite opciones de fiabilidad	Admite opciones de fiabilidad

Tabla 19: Principales características de protocolos Wifi.

Fuente: GEEKFLARE, protocolos de comunicación IOT

Se ha definido y mencionado las principales características de los protocolos más usados en aplicaciones IOT, pero ninguno cumple con las características que se busca, para dar la solución se hizo revisión a los protocolos usados en videojuegos, chat en redes sociales, etc.,

que tienen una latencia de muestreo muy bajo casi imperceptible a gran tráfico de datos. El protocolo de comunicación usados en estos videojuegos es el webSocket.

- Protocolo de comunicación webSocket

Este protocolo de comunicación tiene características que lo hace especial y único frente a los protocolos usados comúnmente en aplicaciones IOT que son las siguientes.

Ventajas:

- ✓ Comunicación persistente entre servidor/cliente a través de una única conexión TCP.
- ✓ Bajo Overheat, reduce significativamente la sobrecarga de datos al tener conexiones persistentes.
- ✓ Comunicación full dúplex, significa una comunicación bidireccional pura, simultanea y en cualquier momento.
- ✓ Latencia baja, como consecuencia de ser persistente entre servidor/cliente el intercambio de información es fluido y por ende de baja latencia.
- ✓ Escalabilidad eficiente, alta capacidad de manejar múltiples conexiones.
- ✓ Seguridad, es compatible con protocolos seguros HTTPS asegurando una comunicación segura y cifrada.

Desventajas:

- ✓ Conexiones persistentes, al tener conexiones persistentes aumenta la carga del servidor.
- ✓ Complejidad en su implementación, los algoritmos usados son de alta complejidad.

Considerando todas las características de todos los protocolos se ha seleccionado al protocolo webSocket por sus altas prestaciones que lo hacen ideal para este diseño.

5.2 Programación de Gateway

En este apartado se desarrolló todo el diseño correspondiente a la programación del ESP32 usando el entorno Arduino ID, es decir la captura de datos del Edge, el muestreo, algoritmos matemáticos, el mapeo según cada variable que corresponda y encaminar la información a capa superior (Cloud).

5.2.1 Estructura y Diseño del Programa

5.2.1.1 Diagrama de Flujo General del Sketch.

El diagrama de flujo general está estructurado de las diferentes rutinas de programación, que representa el diseño del programa principal tomando en cuenta la recogida, procesamiento y encaminado de datos de la medición, como se muestra en la figura 33.

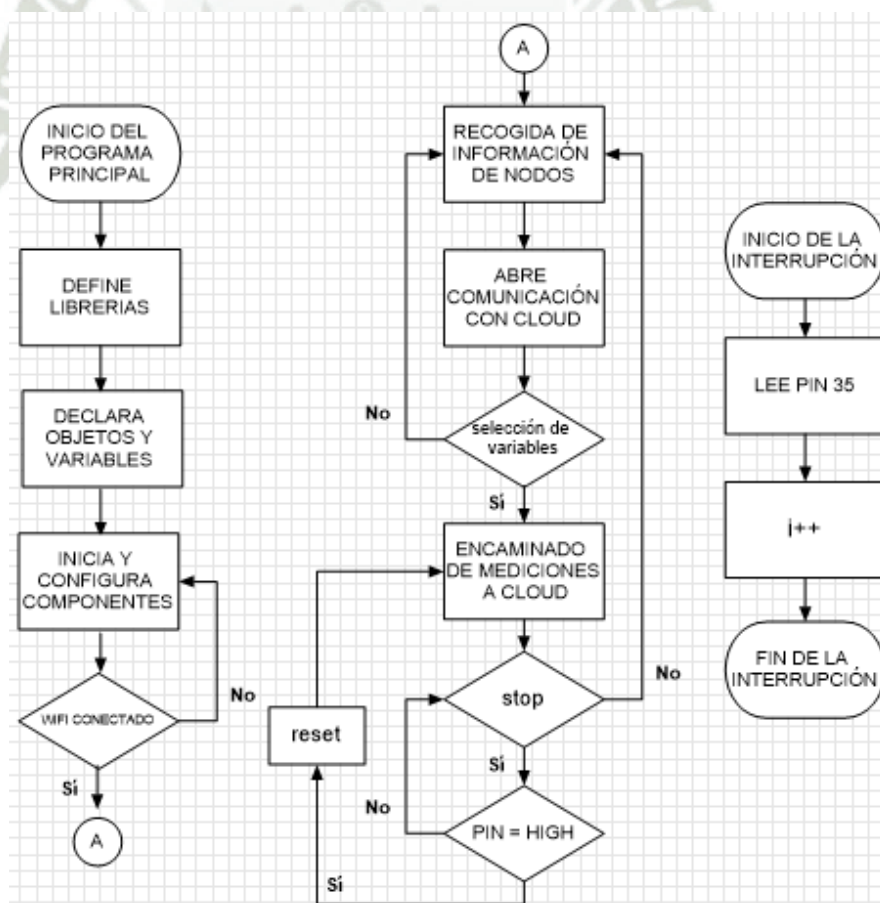


Figura 33: Diagrama de flujo general del sketch.

Fuente: Elaboración propia.

5.2.1.2 Inicialización y Configuración del Sketch.

Como primer paso, antes de arrancar los bucles de programación es necesario inicializar los componentes y herramientas utilizadas, también configurar los eventos relacionados al diseño en el entorno de programación.

- Se incluyen las librerías de uso libre, existen una gran cantidad de librerías de diferentes autores que básicamente cumplen con el mismo objetivo, con pequeñas diferencias como ejemplos, fáciles de descargar, etc. Se seleccionó las siguientes librerías que por defecto se pueden descargar desde el mismo Arduino ID como lo podemos apreciar en la tabla 20.

Librerías para el ESP32 utilizadas		
Nombre	Función	Autor
WiFi.h	Permite la conexión de red Wifi	Propio del Arduino ID
WebSocketsServer.h	Permite la conexión WebSocket	Gil Maimon
ArduinoJson.h	Permite el uso del formato Json	Benoit Blanchon
Wire.h	Permite el uso del I2C	Brian T. Park
Adafruit_ADS1x15.h	Permite el uso del ADS1115	Adafruit
MPU6050.h	Permite el uso del MPU6050	Jeff Rowberg
ESPAsyncWebServer.h	Permite al ESP32 funcionar como servidor web (Cloud)	Lacamera
SPIFFS	Permite subir el script del Cloud (plataforma_io.html) a la memoria Flash del ESP32	Lacamera

Tabla 20: Librerías para el ESP32.

Fuente: Elaboración propia.

```

Plataforma_iot
// librerías .....
#include <WiFi.h> // para conectarse a Wifi
#include <ESPAsyncWebServer.h> // para crear un servidor web
#include <SPIFFS.h> // para subir archivos a la memoria Flash del ESP32
#include <WebSocketsServer.h> // para comunicación instantanea entre gateway/Cloud/cliente
#include <ArduinoJson.h> // para encapsulación Json, enviar en una cadena multiples variables
#include <Wire.h> // para comunicarse por I2C
#include <Adafruit_ADS1X15.h> // para utilizar el ADS1115
#include "MPU6050.h" // para utilizar el MPU6050
// Declara objetos .....

```

Figura 34: Librerías del programa.

Fuente: Elaboración propia.

- La Inicialización y configuración de la comunicación serial, es necesaria para imprimir mensajes según se va desarrollando el bucle de programación, se inicializa y se configura típicamente a 115200 baudios como lo podemos apreciar en la figura 35 siguiente.

```

pinMode(Led, OUTPUT);
pinMode(pIn, INPUT);
// Inicia comunicación serial .....
Serial.begin(115200);
// Inicia comunicación WIFI .....
WiFi.softAP("PlataformaIoT", "111

```

Figura 35: Inicio de comunicación.

Fuente: Elaboración propia.

- La Inicialización y configuración de la comunicación Wifi, es necesaria para establecer la comunicación vía internet, concretamente se configuró en modo Access Point con el ID = Plataforma _iot y contraseña = 11111111. También se hace una petición para que imprima por monitor serie la IP asignada de la red, que más adelante servirá como URL en el explorador. Este proceso lo apreciamos en la figura 36.

```
// Inicia comunicación serial .....  
Serial.begin(115200);  
// Inicia comunicación WIFI .....  
WiFi.softAP("PlataformaIoT", "11111111");  
Serial.println("\nsoftAP");  
Serial.println(WiFi.softAPIP());  
  
// Inicia librerías.....  
Wire.begin();
```

Figura 36: Inicio de comunicación Wifi.

Fuente: Elaboración propia.

- La Inicialización y configuración del servidor web, es necesaria para dar soporte al script de la capa superior Cloud, se inicializó y configuró de la siguiente manera.
 - Primer paso, declaración los objetos del servidor web
AsyncWebServer server (80), servidor web asíncrono para que maneje más de una conexión al mismo tiempo por puerto (80).

```
// Declara objetos .....  
Adafruit_ADS1115 ads; // declara objeto ads  
MPU6050 acelerometro; // declara objeto acelerometro  
AsyncWebServer server(80); // declara objeto server  
WebSocketsServer websocket(81); // declara objeto websockets  
StaticJsonDocument<200> doc; // crea un objeto JSON container  
JsonObject object = doc.to<JsonObject>(); // crea un segundo objeto JSON
```

Figura 37: Declaración del objeto server.

Fuente: Elaboración propia.

WebSocketsServer websockets (81), Es un protocolo de comunicación que una vez abierta se establece una comunicación persistente full dúplex entre Gateway/Cloud por puerto 81.

```
// Declara objetos .....
Adafruit_ADS1115 ads;           // declara objeto ads
MPU6050 acelerometro;          // declara objeto acelerometro
AsyncWebServer server(80);      // declara objeto server
WebSocketsServer websocket(81); // declara objeto websockets
StaticJsonDocument<200> doc;    // crea un objeto JSON container
JsonObject object = doc.to<JsonObject>(); // crea un segundo objeto JSON
```

Figura 38: Declaración de objeto websocket.

Fuente: Elaboración propia.

- Segundo paso, inicialización del servidor web y SPIFFS

En la figura 39 se muestra la inicialización del objeto server

```
// Inicia librerías.....
Wire.begin();           // Habilita Wire para comunicarse por I2C
acelerometro.initialize(); // Habilita el MPU6050
ads.begin();            // Habilita el ADS1115
ads.setGain(GAIN_ONE);  // Ganancia del PGA del ADS1115
server.begin();         // Habilita el servidor (Cloud)
websocket.begin();      // Habilita la comunicación instantánea
websocket.onEvent(event); // Habilita un evento
SPIFFS.begin();         // Habilita el SPIFFS para subir archivos de cloud a flash
```

Figura 39: Inicio de server.

Fuente: Elaboración propia.

En la figura 40 se muestra la inicialización del objeto websocket

```
// Inicia librerías.....
Wire.begin();           // Habilita Wire para comunicarse por I2C
acelerometro.initialize(); // Habilita el MPU6050
ads.begin();            // Habilita el ADS1115
ads.setGain(GAIN_ONE);  // Ganancia del PGA del ADS1115
server.begin();         // Habilita el servidor (Cloud)
websocket.begin();      // Habilita la comunicación instantánea
websocket.onEvent(event); // Habilita un evento
SPIFFS.begin();         // Habilita el SPIFFS para subir archivos de cloud a flash
```

Figura 40: Sensor de corriente.

Fuente: Elaboración propia.

En la figura 41 se muestra la inicialización el SPIFFS

```
// Inicia librerías.....
Wire.begin();           // Habilita Wire para comunicarse por I2C
acelerometro.initialize(); // Habilita el MPU6050
ads.begin();            // Habilita el ADS1115
ads.setGain(GAIN_ONE);  // Ganancia del PGA del ADS1115
server.begin();         // Habilita el servidor (Cloud)
websocket.begin();      // Habilita la comunicación instantánea
websocket.onEvent(event); // Habilita un evento
SPIFFS.begin();         // Habilita el SPIFFS para subir archivos de cloud a flash
```

Figura 41: Inicio de SPIFFS.

Fuente: Elaboración propia.

- Tercer paso, definir la dirección y nombre del archivo de almacenaje en memoria flash.

Se definió el fichero con el nombre “plataforma_iot.html” la extensión puede ser otra como .php, .js, .css, etc. Como se muestra en la figura 42.

```
// Define dirección de Cloud y almacenamiento en memoria Flash .....
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){ // Define fichero plataforma_iot para
request->send(SPIFFS, "/plataforma_iot.html", "text/html");}); // almacenar script y subir al ESP32
```

Figura 42: Define almacenamiento en memoria flash.

Fuente: Elaboración propia.

5.2.2 Recogida y Procesamiento de Medidas de Edge

según la arquitectura definida en el capítulo anterior tenemos tres nodos que recogen las medidas enviadas de los sensores.

5.2.2.1 Nodo 1.

En este nodo se captura, se acondiciona y se muestrea las medidas provenientes de los sensores de voltaje, corriente, temperatura. La conexión circuital está detallado en el capítulo anterior.

5.2.2.1.1 Sensor de Voltaje.

El muestreo, escalamiento y mapeo se hizo de la siguiente forma.

- El rango definido de medición es de 0 – 240VRMS
- El ADC es de 16 bits de resolución, pero realmente es de 15 bits porque reserva un bit para otras funciones, es decir tiene un ancho de banda salida 0 a 32767 muestras.
- El ancho de banda de entrada del ADC es de 0 a 3300mV.
- El sensor transformador de voltaje tiene una reducción de $a = 377.0998$ veces.
- El ancho de banda de salida para la gráfica está definido de la siguiente forma.

- La salida del transformador para el rango de 0 a 240VRMS se hace el siguiente cálculo.

$$Vo = 240VRMS \times \sqrt{2} \frac{1}{377.0998} \times 2 = 1800mV$$

- Entonces el rango de salida del transformador es de 0 a 1800mV.

Pero, para que no ingrese voltajes negativos al ADC se ha sumado 1500VDC de una fuente de tención.

$$Vo = 1800 + 1500 = 3300mV$$

- El rango se ha definido finalmente de 0 a 3300mV.
- El escalamiento de entrada y salida del ADC quedaría de la siguiente forma.

De 0 a 3300mV (salida en milivoltios del sensor) es a 0 a 32767muestras

$$constante\ del\ ADC = \frac{3300mV\ ancho\ de\ banda\ del\ ADC}{32767muestras} = 0.10071mV$$

$$escalamiento = \frac{3300mV\ salida\ del\ sensor}{0.10071mV\ constante\ del\ ADC} = 32000$$

- Finalmente se hizo el mapeo correspondiente de las muestras de 0 a 32000 con el rango definido por diseño que es de 0 a 679V (los valores de voltaje en RMS serían de 0 a 240VRMS).

- El factor de conversión es 48.25, como se muestra en la figura 43.

$$\text{factor de conversión} = \frac{32767}{679} = 48.25$$

```
// SENSOR DE VOLTAJE.....
short vol = ads.readADC_SingleEnded(0); // lee y almacena el valor de A0 en vol
vol = (vol/48.25)-312.5; // mapeo de voltaje
```

Figura 43: Escalamiento de voltaje.

Fuente: Elaboración propia.

- Para quitar el voltaje DC a la señal se resta 312, como se muestra en la figura 44.

$$VDC = \frac{1500}{0.1} = 15000$$

$$VDC = \frac{15000}{\text{factor de conversión}} = \frac{15000}{48} = 312.5VDC$$

```
// SENSOR DE VOLTAJE.....
short vol = ads.readADC_SingleEnded(0); // lee y almacena el valor de A0 en vol
vol = (vol/48.25)-312.5; // mapeo de voltaje
```

Figura 44: Resta voltaje DC.

Fuente: Elaboración propia.

5.2.2.1.2 Sensor de Corriente.

El muestreo, escalamiento y mapeo se hizo de la siguiente forma.

- El rango definido de medición es de 0 – 10ARMS
- El ADC es de 16 bits de resolución, pero realmente es de 15 bits porque reserva un bit para otras funciones, es decir tiene un ancho de banda salida 0 a 32767 muestras.
- El ancho de banda de entrada del ADC es de 0 a 3300mV.
- El sensor SCT013 de corriente tiene una relación corriente voltaje de 30A/1V.
- El ancho de banda de salida para la gráfica está definido de la siguiente forma.

- Para la salida del SCT013 para el rango de 0 a 10ARMS se ha hecho el siguiente cálculo.

$$I_o = 10000mARMS \times \sqrt{2} \frac{1000mV}{30000mA} = 471mV$$

- Entonces el rango de salida del transformador será de 0 a 471mV.
- El escalamiento de entrada y salida del ADC quedará de la siguiente forma.

De 0 a 471mV (salida en milivoltios del sensor) es a 0 a 32767muestras

$$constante\ del\ ADC = \frac{3300mV\ ancho\ de\ banda\ del\ ADC}{32767muestras} = 0.10071mV$$

$$escalamiento = \frac{471mV}{0.10071mV} = 4676\ muestras$$

- Finalmente se hizo el mapeo correspondiente de las muestras de 0 a 4676 con el rango definido por diseño que es de 0 a 14142mA (los valores de corriente en RMS serían de 0 a 10000mARMS).

- El factor de conversión es 0.33065, como se muestra en la figura 45.

$$factor\ de\ convesión = \frac{4676muestras}{14142mA} = 0.33065$$

```
// SENSOR DE CORRIENTE.....
short cor = ads.readADC_SingleEnded(2); // lee y almacena el valor de A2 en vol
cor = (cor/0.33065); // mapeo de corriente
```

Figura 45: Escalamiento de corriente.

Fuente: Elaboración propia.

5.2.2.1.3 Sensor de Temperatura.

El muestreo, escalamiento y mapeo se hizo de la siguiente forma.

- El rango definido de medición es de 0 – 150°C

- El ADC es de 16 bits de resolución, pero realmente es de 15 bits porque reserva un bit para otras funciones, es decir tiene un ancho de banda salida 0 a 32767 muestras.
- El ancho de banda de entrada del ADC es de 0 a 3300mV.
- El sensor LM35 de temperatura tiene una relación temperatura voltaje de 1°C/10mV
- El ancho de banda de salida para la gráfica está definido de la siguiente forma.

- La salida del LM35 para el rango de 0 a 150°C se hace el siguiente cálculo.

$$I_o = 150^{\circ}\text{C} \frac{10\text{mV}}{1^{\circ}\text{C}} = 1500\text{mV}$$

- Entonces el rango de salida del LM35 será de 0 a 1500mV.

- El escalamiento de entrada y salida del ADC quedará de la siguiente forma.

De 0 a 1500mV (salida en milivoltios del sensor) es a 0 a 32767muestras

$$\text{constante del ADC} = \frac{3300\text{mV ancho de banda del ADC}}{32767\text{muestras}} = 0.10071\text{mV}$$

$$\text{escalamiento} = \frac{1500\text{mV}}{0.10071\text{mV}} = 14894 \text{ muestras}$$

- Finalmente se hizo el mapeo correspondiente de las muestras de 0 a 14894 con el rango definido por diseño que es de 0 a 150°C.

- El factor de conversión es 99.29, como se muestra en la figura 46.

$$\text{factor de convesión} = \frac{14894\text{muestras}}{150^{\circ}\text{C}} = 99.29$$

```
// SENSOR DE TEMPERATURA.....
short tem = ads.readADC_SingleEnded(3); // lee y almacena el valor de A3 en vol
tem = (tem/99.29); // mapeo de temperatura
```

Figura 46: Escalamiento de temperatura.

Fuente: Elaboración propia.

5.2.2.2 Nodo 2.

5.2.2.2.1 Sensor de Vibración.

El muestreo, escalamiento y mapeo se hizo de la siguiente forma.

- El rango definido de medición es de 0 a 20mm/s²
- El MPU6050 tiene internamente un ADC es de 16 bits de resolución, pero realmente es de 15 bits porque reserva un bit para otras funciones, es decir tiene un ancho de banda salida 0 a 32767 muestras.
- Según la hoja de datos del MPU6050 tenemos.

- Alta sensibilidad de 2g de aceleración, es decir 2g en todo el rango de muestreo, donde g es la aceleración de la gravedad $g = 9.80665 \text{ m/s}^2$

$$\text{sensibilidad} = \frac{2g \text{ de aceleración}}{32767 \text{ muestras}} = \frac{2 \cdot 9.80665 \text{ m/s}^2}{32767 \text{ muestras}} = 0.5 \text{ mm/s}^2$$

- La salida del eje z está calibrado por fabrica a 1g, es decir, esta salida debería estar a la mitad del rango de muestreo por default. Se ha puesto a prueba imprimiendo por puerto serie y efectivamente está a la mitad de todo el rango, como se muestra en la figura 47.



Figura 47: Salida numérica del sensor de vibraciones.

Fuente: Elaboración propia.

- El mapeo correspondiente de 0 a 32767 - 0 a 20000mm/s² definido por diseño está dado por la ecuación siguiente.
- El factor de conversión es 1.63835, como se muestra en la figura 48.

$$\text{factor de conversión} = \frac{32767 \text{ muestras}}{20000 \text{ mm/s}^2} = 1.63835$$

```
// SENSOR DE VIBRACION.....
acelerometro.getMotion6(&v_x, &v_y, &v_z, &gx, &gy, &gz); // lee y almacena valor de A3 en vol
v_a = (v_x/1.63385)+270; // mapeo de aceleración, sensibilidad = 0.5 mm/s2
```

Figura 48: Escalamiento de aceleración.

Fuente: Elaboración propia.

- La conversión de la aceleración a velocidad según White (1990), se hace mediante la siguiente ecuación.

$$velocidad = \frac{87.75 \times aceleración}{frecuencia}$$

Llevando a código esta ecuación se muestra en la figura 49.

```
// SENSOR DE VIBRACION.....
acelerometro.getMotion6(&v_x, &v_y, &v_z, &gx, &gy, &gz); // lee y almacena valor de A3 en vol
v_a = (v_x/1.63385)+270-50; // mapeo de aceleración, sensibilidad = 0.598755 mm/s2
v_v = (v_a*86.75)/435; // conversión a velocidad
v_d = (v_v*318.5)/435; // conversión a desplazamiento
```

Figura 49: Escalamiento de velocidad.

Fuente: Elaboración propia.

La frecuencia es 435Hz. El cálculo de esta frecuencia se detalla en el siguiente apartado (Programación de la capa Cloud).

- La conversión de la velocidad a desplazamiento según White (1990), se hace mediante la siguiente ecuación.

$$velocidad = \frac{318.5 \times velocidad}{frecuencia}$$

Llevando a código esta ecuación se muestra en la figura 50.

```
// SENSOR DE VIBRACION.....
acelerometro.getMotion6(&v_x, &v_y, &v_z, &gx, &gy, &gz); // lee y almacena valor de A3 en vol
v_a = (v_x/1.63385)+270-50; // mapeo de aceleración, sensibilidad = 0.598755 mm/s2
v_v = (v_a*86.75)/435; // conversión a velocidad
v_d = (v_v*318.5)/435; // conversión a desplazamiento
```

Figura 50: Escalamiento de desplazamiento.

Fuente: Elaboración propia.

La frecuencia es 435Hz. El cálculo de esta frecuencia se detalla en el siguiente apartado (Programación de la capa Cloud).

5.2.2.3 *Nodo 3.*

5.2.2.3.1 *Sensor de RPM.*

El muestreo, escalamiento y mapeo se hizo de la siguiente forma.

- El rango definido de medición es de 0 – 10000r/min
- El sensor AR-FC03 de RPM es un encoder de salida booleana, tiene salida digital dependiendo de la interrupción de luz entre emisor – receptor (uno o cero), para contar las revoluciones se implementó una muesca que cortara la luz por cada giro que haga el motor.
- Se ha configurado el pin 35 como interrupción del ESP32 para hacer el respectivo conteo de revolución, para eliminar retardo del ciclo de programación y sea exacto el conteo, como se muestra en la figura 51.

```
void setup(void) {
// Define interrupción y pines.....
attachInterrupt(35,interrupcion,RISING); // Define interrupcion (pin35)
```

Figura 51: Define interrupción.

Fuente: Elaboración propia.

```
// Funcion que se ejecuta durante cada interrupcion.
void interrupcion()
{
i++; // Se incrementa en uno el contador
}
```

Figura 52: Bucle de interrupción.

Fuente: Elaboración propia.

- Para el reseteo del conteo se ha implementado un timer de 2 segundos. Este timer está basado en la función millis, de esta forma se crea un hilo de programación adicional para no afectar a las demás funciones, como se muestra en la figura 53.

```
// SENSOR DE RPM.....  
tiempo = millis();           // habilita temporizador  
if ((tiempo - tr) >= 2000){   // define temporizador a 2s  
    tr = tiempo;             // resetea temporizador  
    rpm = i;                  // almacena valor en rpm  
    i = 0;                    // resetea contador  
}
```

Figura 53: Función millis timer.

Fuente: Elaboración propia.

Millis es un contador, tiempo almacena la cuenta, cuando la cuenta sea mayor o igual a 2000 milisegundos se ejecuta el reseteo del contador asignado a la variable i.

- Para completar los 60000 milisegundos que tiene un minuto se multiplica por 30 al contador rpm, como se muestra en la figura 54.

```
if(r == 1){object["sensor"] = rpm*30; // almacena valor rpm en objeto  
    object["ref1"] = 0;           // almacena valor minimo en objeto  
    object["ref2"] = 10000;}      // almacena valor maximo en objeto
```

Figura 54: Completa a un minuto el conteo.

Fuente: Elaboración propia.

5.2.3 Encaminado de Medidas a Cloud

Para el encaminado de medidas provenientes de Edge previamente procesadas y mapeadas se ha implementado un selector para que no se envíen todos a la vez, esto porque consumiría muchos recursos al sistema y ralentizaría la comunicación. Dicho selector está controlado por el lado del cliente de forma virtual con botoneras que interactúan entre cliente

y servidor. La programación de la parte del cliente se detalla en el apartado Programación de la capa Cloud.

Para configurar el uso del protocolo webSocket y envío de medidas se hizo los siguientes pasos.

- Paso uno, para almacenar al objeto (object) con todas medidas de los sensores se declaró un objeto container con un espacio de almacenamiento de 200 caracteres, con el nombre doc. Como se muestra en la figura 55.

```
// Declara objetos .....  
Adafruit_ADS1115 ads;           // declara objeto ads  
MPU6050 acelerometro;          // declara objeto acelerometro  
AsyncWebServer server(80);      // declara objeto server  
WebSocketsServer websocket(81); // declara objeto websockets  
StaticJsonDocument<200> doc;    // crea un objeto JSON container  
JsonObject object = doc.to<JsonObject>(); // crea un segundo objeto JSON
```

Figura 55: Declara objeto Json.

Fuente: Elaboración propia.

- Paso dos, para almacenar toda la data de las medidas se declaró un segundo objeto, con el nombre object. Como se muestra en la figura 56.

```
// Declara objetos .....  
Adafruit_ADS1115 ads;           // declara objeto ads  
MPU6050 acelerometro;          // declara objeto acelerometro  
AsyncWebServer server(80);      // declara objeto server  
WebSocketsServer websocket(81); // declara objeto websockets  
StaticJsonDocument<200> doc;    // crea un objeto JSON container  
JsonObject object = doc.to<JsonObject>(); // crea un segundo objeto JSON  
// declara pines .....
```

Figura 56: Declara segundo objeto object.

Fuente: Elaboración propia.

- Paso tres, para enviar los datos será necesario convertir el objeto en un string (cadena Json), para esto se ha declarado jsonString como se muestra en la figura 57.

```
volatile int i = 0;           // Variable para almacenar en memoria RAM,
unsigned long tiempo, tr=0;  // Variable para almacenar los millis
String jsonString;           // Variable para almacenar una cadena JSON
```

Figura 57: Sensor de corriente.

Fuente: Elaboración propia.

- Paso cuatro, se ha hecho la conversión de doc a jsonString mediante el comando serializeJson, como se muestra en la figura 58.

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString);    // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 58: Convierte Json en cadena.

Fuente: Elaboración propia.

- Paso cinco, finalmente se imprime los datos por monitor serie y se envía por protocolo websocket a capa superior, como se muestra en la figura 59. Se detalla este proceso en el siguiente apartado.

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString);    // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 59: Envía mediciones a Cloud.

Fuente: Elaboración propia.

5.2.3.1 Encaminado de Medidas de Voltaje.

Para el encaminado de las medidas de voltaje se ha seguido los siguientes pasos

- Declaración de variables de control, como se muestra en la figura 60.

```
// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //
```

Figura 60: Declara variable v.

Fuente: Elaboración propia.

- Mensaje del cliente

```
// Lee datos de Cloud.....
void event(uint8_t, WStype_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*) (payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica rpm
    if(mensaje == "v_a_on") {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica aceleración
    if(mensaje == "v_v_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica velocidad
    if(mensaje == "v_d_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica desplazamiento
```

Figura 61: Habilita encaminado de voltaje.

Fuente: Elaboración propia.

- Selección de envío

```
// Envía datos de Gateway a capa superior.....
if(v == 1){object["sensor"] = vol; // almacena valor voltaje en objeto
            object["ref1"] = -311.13; // almacena valor max negativo en objeto
            object["ref2"] = 311.13;} // almacena valor max positivo en objeto
```

Figura 62: Almacena voltaje en objeto.

Fuente: Elaboración propia.

- Envío a Cloud

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 63: Encaminado de voltaje.

Fuente: Elaboración propia.

5.2.3.2 Encaminado de Medidas de Corriente.

Para el encaminado de las medidas de corriente se ha seguido los siguientes pasos.

- Declaración de variable de control

```
// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //
```

Figura 64: Declara variable c.

Fuente: Elaboración propia.

- Mensaje del cliente

```
// Lee datos de Cloud.....
void event(uint8_t, WStype_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*) (payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica rpm
    if(mensaje == "v_a_on") {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica aceleración
    if(mensaje == "v_v_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica velocidad
    if(mensaje == "v_d_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica desplazamiento
```

Figura 65: Habilita encaminado de corriente.

Fuente: Elaboración propia.

- Selección de envío

```
if(c == 1){object["sensor"] = cor; // almacena valor corriente en objeto
          object["ref1"] = -2121.32; // almacena valor max negativo en objeto
          object["ref2"] = 2121.32;} // almacena valor max positivo en objeto
```

Figura 66: Almacena corriente en objeto.

Fuente: Elaboración propia.

- Envío a Cloud

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 67: Encaminado de corriente.

Fuente: Elaboración propia.

5.2.3.3 Encaminado de Medidas de Temperatura.

Para el encaminado de las medidas de temperatura se ha seguido los siguientes pasos.

- Declaración de variable de control

```
// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //
```

Figura 68: Declara variable t.

Fuente: Elaboración propia.

- Mensaje del cliente

```
// Lee datos de Cloud.....
void event(uint8_t, WStype_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*)(payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica rpm
    if(mensaje == "v_a_on") {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica aceleración
    if(mensaje == "v_v_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica velocidad
    if(mensaje == "v_d_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica desplazamiento
```

Figura 69: Habilita encaminado de temperatura.

Fuente: Elaboración propia.

- Selección de envío

```
if(t == 1){object["sensor"] = tem; // almacena valor temperatura en objeto
          object["ref1"] = 0; // almacena valor minimo en objeto
          object["ref2"] = 150;} // almacena valor maximo en objeto
```

Figura 70: Almacena temperatura en objeto.

Fuente: Elaboración propia.

- Envío a Cloud

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 71: Encaminado de temperatura.

Fuente: *Elaboración propia.*

5.2.3.4 Encaminado de Medidas de Vibración.

Para el encaminado de las medidas de vibración se ha seguido los siguientes pasos

- Aceleración
 - Declaración de variable de control

```
// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //
```

Figura 72: *Declaración de la variable va.*

Fuente: *Elaboración propia.*

- Mensaje del cliente

```
// Lee datos de Cloud.....
void event(uint8_t, WStype_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*)(payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica rpm
    if(mensaje == "v_a_on") {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica aceleración
    if(mensaje == "v_v_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica velocidad
    if(mensaje == "v_d_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica desplazamiento
```

Figura 73: *Habilita encaminado de la aceleración.*

Fuente: *Elaboración propia.*

- Selección de envío

```
if(va == 1){object["sensor"] = v_a; // almacena valor aceleración en objeto
            object["ref1"] = -2326; // almacena valor max negativo en objeto
            object["ref2"] = 2326;} // almacena valor max positivo en objeto
```

Figura 74: *Almacena aceleración en objeto.*

Fuente: *Elaboración propia.*

- Envío a Cloud

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 75: Encaminado de aceleración.

Fuente: Elaboración propia.

- Velocidad

- Declaración de variable de control

```
// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //
```

Figura 76: Declaración de vv.

Fuente: Elaboración propia.

- Mensaje del cliente

```
// Lee datos de Cloud.....
void event(uint8_t, WStype_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*)(payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica rpm
    if(mensaje == "v a on" ) {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica aceleración
    if(mensaje == "v v on" ) {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica velocidad
    if(mensaje == "v_d on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica desplazamiento
```

Figura 77: Habilita encaminado de velocidad.

Fuente: Elaboración propia.

- Selección de envío

```
if(vv == 1){object["sensor"] = v_v; // almacena valor velocidad en objeto
            object["ref1"] = -400; // almacena valor max negativo en objeto
            object["ref2"] = 400;} // almacena valor max positivo en objeto
```

Figura 78: Almacena velocidad en objeto.

Fuente: Elaboración propia.

- Envío a Cloud

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 79: Encaminado de velocidad.

Fuente: Elaboración propia.

- Desplazamiento

- Declaración de variable de control

```
// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //
```

Figura 80: Declaración de vd.

Fuente: Elaboración propia.

- Mensaje del cliente

```
// Lee datos de Cloud.....
void event(uint8_t, WStype_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*) (payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica rpm
    if(mensaje == "v_a_on") {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica aceleración
    if(mensaje == "v_v_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica velocidad
    if(mensaje == "v_d_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica desplazamiento
```

Figura 81: Habilita encaminado de desplazamiento.

Fuente: Elaboración propia.

- Selección de envío

```
if(vd == 1){object["sensor"] = v_d; // almacena valor desplazamiento en objeto
            object["ref1"] = -688; // almacena valor max negativo en objeto
            object["ref2"] = 688;} // almacena valor max positivo en objeto
```

Figura 82: Almacena desplazamiento en objeto.

Fuente: Elaboración propia.

- Envío a Cloud

```

serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud

```

Figura 83: Encaminado de desplazamiento.

Fuente: Elaboración propia.

5.2.3.5 Encaminado de Medidas de RPM.

Para el encaminado de las medidas de voltaje se ha seguido los siguientes pasos

- Declaración de variable de control

```

// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //

```

Figura 84: Declaración de r.

Fuente: Elaboración propia.

- Mensaje del cliente

```

// Lee datos de Cloud.....
void event(uint8_t, WType_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*) (payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica rpm
    if(mensaje == "v_a_on") {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica aceleración
    if(mensaje == "v_v_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica velocidad
    if(mensaje == "v_d_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica desplazamiento

```

Figura 85: Habilita encaminado de RPM.

Fuente: Elaboración propia.

- Selección de envío

```

// Envía datos de Gateway a capa superior.....
if(v == 1){object["sensor"] = vol; // almacena valor voltaje en objeto
            object["ref1"] = -311.13; // almacena valor max negativo en objeto
            object["ref2"] = 311.13;} // almacena valor max positivo en objeto

```

Figura 86: Almacena RPM en objeto.

Fuente: Elaboración propia.

- Envío a Cloud

```
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
```

Figura 87: Encaminado de RPM.

Fuente: Elaboración propia.

5.2.4 Stop en las Gráficas

Se implementó una rutina de parada de flujo de información de medidas para hacer un stop en las gráficas, esta función ayudará al usuario tener una mejor valoración en el análisis de las gráficas.

- Declaración de variable de control

```
// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0; //
```

Figura 88: Declaración de p.

Fuente: Elaboración propia.

- Declaración de variable del pin 4

```
// declara pines .....
#define Led 2 // declara pin 2
#define Pin 4 // declara pin 4
```

Figura 89: Define pin 4 del ESP32.

Fuente: Elaboración propia.

- Define pin 4 del ESP32 como entrada

```
// Define interrupción y pines.....
attachInterrupt(35,interrupcion,RISING); // Define interrupcion (pin35)
pinMode(Led, OUTPUT); // Define pin 2 como salida
pinMode(Pin, INPUT); // Define pin 2 como entrada
```

Figura 90: Define Pin como entrada.

Fuente: Elaboración propia.

- Mensaje del cliente

```
if(mensaje == "pause_on") {p = 1;} // stop on
if(mensaje == "led_on") {digitalWrite(Led,HIGH);} // enciende led
if(mensaje == "led_off") {digitalWrite(Led,LOW);} // apaga led
```

Figura 91: Habilita STOP.

Fuente: Elaboración propia.

- Bucle infinito

```
while(p == 1){if(digitalRead(Pin) == HIGH){p = 0;break;}} // ciclo infinito para stop
serializeJson(doc, jsonString); // convertir objeto JSON a cadena
```

Figura 92: Rutina de control STOP.

Fuente: Elaboración propia.

- Se sale del bucle infinito dando al Pin un valor booleano 1. La instrucción break rompe el bucle, se restablece a cero el valor de p y continúa funcionando el programa principal.

```
while(p == 1){if(digitalRead(Pin) == HIGH){p = 0;break;}} // ciclo infinito para stop
```

Figura 93: Rompe el bucle infinito.

Fuente: Elaboración propia.

5.2.5 Encendido del LED

Se ha implementado adicionalmente el control de encendido y apagado de un led en el pin 2 del ESP32, que se controla del lado del cliente mediante botoneras ON/OFF

virtuales. Este circuito se hizo con la finalidad de dar una señal de alarma o aviso según la preferencia del usuario mediante el encendido de un LED.

- Declaración de variable del pin 2

```
// declara pines .....
#define Led 2                // declara pin 2
#define Pin 4                // declara pin 4
```

Figura 94: Declaración del pin 2 del ESP32.

Fuente: Elaboración propia.

- Define pin 2 del ESP32 como salida

```
// Define interrupción y pines.....
attachInterrupt(35,interrupcion,RISING); // Define interrupcion (pin35)
pinMode(Led, OUTPUT);                  // Define pin 2 como salida
pinMode(Pin, INPUT);                   // Define pin 2 como entrada
```

Figura 95: Define Led como salida.

Fuente: Elaboración propia.

- Mensaje del cliente

```
if(mensaje == "pause_on") {p = 1;} // stop on
if(mensaje == "led_on") {digitalWrite(Led,HIGH);} // enciende led
if(mensaje == "led_off") {digitalWrite(Led,LOW);} // apaga led
}
```

Figura 96: Rutina de control ON/OFF del LED.

Fuente: Elaboración propia.

- Encendido y apagado del LED

```
if(mensaje == "pause_on") {p = 1;} // stop on
if(mensaje == "led_on") {digitalWrite(Led,HIGH);} // enciende led
if(mensaje == "led_off") {digitalWrite(Led,LOW);} // apaga led
}
```

Figura 97: Enciende y apaga LED.

Fuente: Elaboración propia.

5.3 Programación de Cloud

Esta capa está hecha en base al lenguaje de programación JavaScript y alojada en memoria flash del ESP32 que tiene una 4MB, todo el programa realizado en esta capa pesa 272KB, quedando mucho espacio para hacer mejoras futuras como dar estilo con CCS, implementar FFT a las gráficas, etc.

Para la programación de esta capa se utilizó la librería de gráficos Highcharts.

Highcharts es una librería de gráficos programada con JavaScript. Esta librería es gratuita para uso no comercial.

5.3.1 Estructura y Diseño del Programa

5.3.1.1 Estructura del Programa.

La estructura del programa está diseñada de la siguiente manera.

- Diagrama de flujo principal



Figura 98: Diagrama de flujo principal de Cloud.

Fuente: Elaboración propia.

En el diagrama de flujo de la figura 98 se precisan los diferentes lenguajes de programación usados y como interactúan los datos entre capas Cloud y Gateway mediante el protocolo Wifi de mensajería Full dúplex webSocket.

- Diagrama de flujo del código HTML

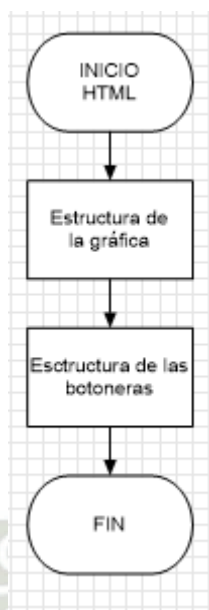


Figura 99: *Diagrama de flujo HTML.*

Fuente: *Elaboración propia.*

El diagrama de flujo de la figura 99 muestra la estructura visual que posicionará la gráfica y las botoneras en el navegador web.

- Diagrama de flujo de configuración del chart.

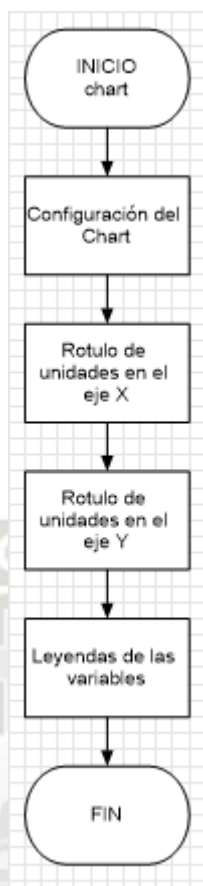


Figura 100: *Diagrama de chart.*

Fuente: *Elaboración propia.*

El diagrama de flujo de la figura 100 muestra la configuración de la librería chart como el uso de mensajes, leyendas, etc.

- Diagrama de flujo del código JavaScript.

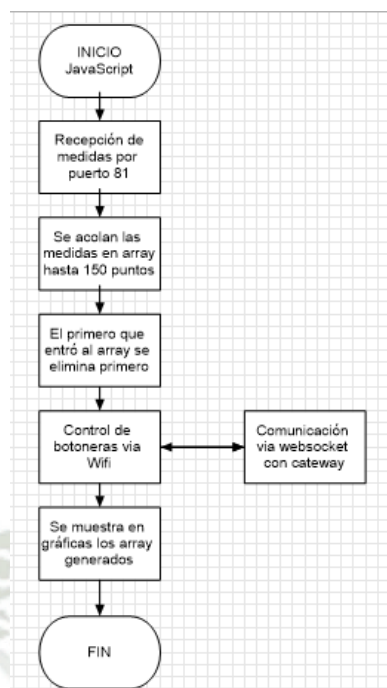


Figura 101: Diagrama de flujo JavaScript.

Fuente: Elaboración propia.

El diagrama de flujo de la figura 101 muestra la recepción de medidas, almacenamiento de puntos en array y el renderizado para su ilustración en graficas. También muestra el control de botoneras para hacer la selección de las medidas y otros.

5.3.1.2 Diseño del Programa.

- Código HTML
 - Id del gráfico y clase container

```

<body>
  <div id="iot" class="container"></div>
  <center>

```

Figura 102: Id para el chart.

Fuente: Elaboración propia.

Este código de la figura 102 estructura y fija la gráfica en el navegador web

- Id del interfaz de mediciones

```

6   <body>
7   <div id="iot" class="container"></div>
8   <center>
9   <p>SENSOR <span id='vol'>-</span></p>
10  </center>
11  <center>

```

- Clase de botoneras

```

<center>
<p>
<button class="sensor" onclick= "vol_on()">VOLTAJE</button>
<button class="sensor" onclick= "cor_on()">CORRIENTE</button>
<button class="sensor" onclick= "tem_on()">TEMPERATURA</button>
<button class="sensor" onclick= "rpm_on()">RPM</button>
</p>
</center><center>
<p>
<button class="sensor" onclick= "v_a_on()">ACELERACIÓN</button>
<button class="sensor" onclick= "v_v_on()">VELOCIDAD</button>
<button class="sensor" onclick= "v_d_on()">DESPLAZAMIENTO</button>
</p>
</center><center>
<p>
<button class="paus" onclick= "pause_on()">STOP</button>
</p>
</center><center>
<p>
<button class="led" onclick= "led_on()">LED ON</button>
<button class="led" onclick= "led_off()">OFF</button>
</p>
<center></center>

```

Figura 103: Clase de las botoneras.

Fuente: Elaboración propia.

Este código de la figura 103 estructura y fija las botoneras en el navegador web.

- Código JavaScript

- Rutinas de configuración de gráficos.

```

const chartT = new Highcharts.Chart({
  chart:{ renderTo : 'iot' },

```

Figura 104: Define Highcharts.

Fuente: Elaboración propia.

Este código de la figura 104 hace un llamado a la librería Highcharts

- Rutinas de recepción de medidas.

```
var Socket = new WebSocket('ws://' + window.location.hostname + ':81/');  
Socket.onmessage = function(event){
```

Figura 105: Puerta de ingreso de mediciones.

Fuente: Elaboración propia.

Este código deprecia medidas por puerto 81, como se muestra en la figura 105.

➤ Rutinas de almacenamiento de medidas

- Configura y almacena el tiempo

```
var x = (new Date()).getTime();  
var y = JSON.parse(event.data);
```

Figura 106: Define tiempo para eje x.

Fuente: Elaboración propia.

Se almacena el tiempo en x como se muestra en la figura 106 para graficar el tiempo en el eje x

- Convierte una cadena en un número.

```
var x = (new Date()).getTime();  
var y = JSON.parse(event.data);
```

Figura 107: Define mediciones para eje y.

Fuente: Elaboración propia.

Como se muestra en la figura 107 a la cadena Json se le hizo un parseo antes de almacenarlo en la variable y, para graficar las medidas en el eje y.

- Visualiza interfaz de mediciones de la variable sensores

```
var x = (new Date()).getTime();  
var y = JSON.parse(event.data);  
document.getElementById('vol').innerHTML = y.sensor;
```

- Almacena 150 puntos en array

```
var x = (new Date()).getTime();
var y = JSON.parse(event.data);

if(chartT.series[0,1,2].data.length < 150) {
    chartT.series[0].addPoint([x, y.sensor], true, false, true);
    chartT.series[1].addPoint([x, y.ref1], true, false, true);
    chartT.series[2].addPoint([x, y.ref2], true, false, true);
}else {
    chartT.series[0].addPoint([x, y.sensor], true, true, true);
    chartT.series[1].addPoint([x, y.ref1], true, true, true);
    chartT.series[2].addPoint([x, y.ref2], true, true, true);
}
```

Figura 108: Almacena 150 puntos en array.

Fuente: Elaboración propia.

Como se muestra en la figura 108 el código almacena 150 puntos en el array series.

- Dinamiza el array de derecha a izquierda, sumando puntos a la derecha y eliminando puntos por la izquierda. Como se muestra en la figura 110.

```
var x = (new Date()).getTime();
var y = JSON.parse(event.data);

if(chartT.series[0,1,2].data.length < 150) {
    chartT.series[0].addPoint([x, y.sensor], true, false, true);
    chartT.series[1].addPoint([x, y.ref1], true, false, true);
    chartT.series[2].addPoint([x, y.ref2], true, false, true);
}else {
    chartT.series[0].addPoint([x, y.sensor], true, true, true);
    chartT.series[1].addPoint([x, y.ref1], true, true, true);
    chartT.series[2].addPoint([x, y.ref2], true, true, true);
}
```

Figura 109: Dinamiza el chart.

Fuente: Elaboración propia.

- Rutinas de control de botoneras y gráficos

```
</html>

<script>
function vol_on() { Socket.send("vol_on");}
function cor_on() { Socket.send("cor_on");}
function tem_on() { Socket.send("tem_on");}
function rpm_on() { Socket.send("rpm_on");}
function v_a_on() { Socket.send("v_a_on");}
function v_v_on() { Socket.send("v_v_on");}
function v_d_on() { Socket.send("v_d_on");}

function pause_on() { Socket.send("pause_on");}
function led_on() { Socket.send("led_on");}
function led_off() { Socket.send("led_off");}

const chartT = new Highcharts.Chart({
```

Figura 110: Sensor de corriente.

Fuente: Elaboración propia.

Estas botoneras tienen la finalidad de enviar mensajes de texto a Gateway para establecer control según preferencias del usuario, esta función se detalla en el siguiente apartado Recogida de información del Gateway. Cálculo del Tiempo y Frecuencia de Muestreo de Medición

5.3.1.3 Cálculo del Tiempo de Muestreo.

El cálculo de muestreo se hizo en base una señal de 60Hz de frecuencia emitida por un generador de funciones, como podemos apreciar en la figura, donde se ve la señal de voltaje en la gráfica versus el tiempo.

En la gráfica de la figura 111 se muestra la señal de 60Hz, donde cada punto es la medida de la señal muestreada que se une por rectas, a menor tiempo de muestreo habrá más puntos y estará mejor definida la onda senoidal típica del voltaje de señal alterna.

- Cálculo del tiempo de muestreo

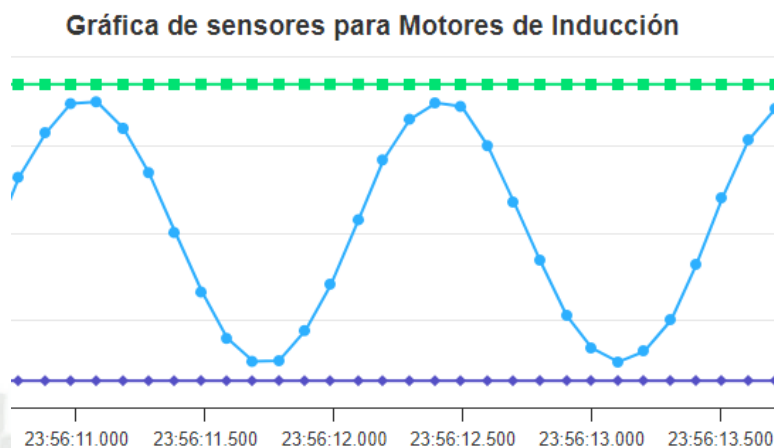


Figura 111: Muestreo en señal de 60Hz.

Fuente: Elaboración propia.

Para el cálculo de muestreo tenemos la siguiente ecuación.

$$\text{frecuencia de la onda} = 60\text{Hz}$$

$$\text{periodo de la onda} = \frac{1}{60\text{Hz}} = 16.67 \text{ ms}$$

$$\text{número de muestras por cada onda} = 14.5 \text{ muestras}$$

$$\text{tiempo de muestreo} = \frac{16.67\text{ms}}{14.5} = 1.15 \text{ ms}$$

El tiempo de muestreo calculado es 1.15 ms.

- Cálculo de la equivalencia del tiempo de muestreo medido versus el tiempo de muestreo real. En la figura 112 se muestra la onda de 60Hz de frecuencia con mediciones unidas por rectas.

Gráfica de sensores para Motores de Inducción

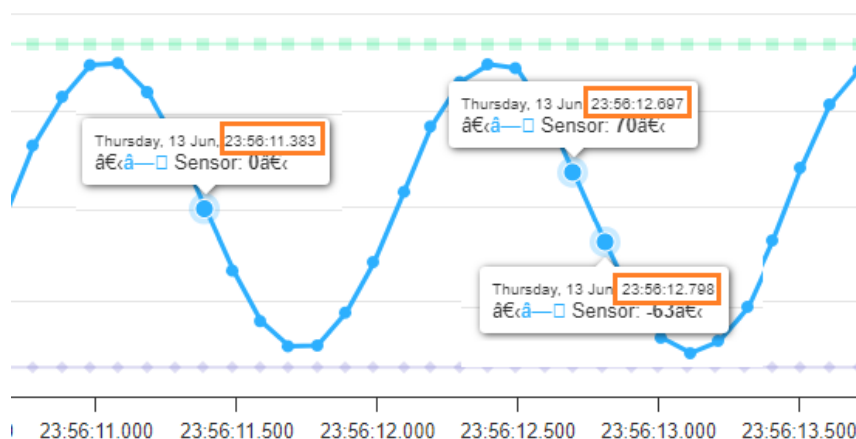


Figura 112: Mediciones de tiempo en onda.

Fuente: Elaboración propia.

Para el cálculo tenemos la siguiente ecuación.

tiempo de duración de un onda = tiempo final – tiempo inicial

$$tiempo\ final = \frac{12.697 - 12.798}{2} = 12.748$$

$$tiempo\ inicial = 11.383$$

$$tiempo\ de\ duración\ de\ la\ onda = 12.748 - 11.383 = 1.3ms$$

- El tiempo que recorre una onda senoidal según la gráfica es 1.3ms
- Pero el tiempo real que recorre una onda senoidal es de 16.67 ms

Para hallar la equivalencia usamos la siguiente ecuación.

$$factor\ de\ conversión = \frac{tiempo\ real}{tiempo\ de\ la\ gráfica}$$

$$factor\ de\ conversión = \frac{16.67ms}{1.3ms} = 12.82$$

Finalmente se obtuvo la equivalencia entre el tiempo real y el tiempo de la gráfica que es de 12.82, esto quiere decir que al medir el tiempo en la gráfica se tiene que multiplicar por 12.8 para tener el tiempo real.

5.3.1.4 *Cálculo del Rango de Medición.*

El rango de frecuencias de medición está definido por la velocidad de transmisión de datos o tiempo de muestreo de medición que es 1.15ms que detallaremos este cálculo en el siguiente apartado Cálculo del tiempo de muestreo. Para el cálculo de rango de frecuencias se utiliza la siguiente ecuación.

- Frecuencia de muestreo

$$frecuencia\ maxima = \frac{1}{tiempo\ de\ muestreo} = \frac{1}{0.00115s} = 870Hz$$

- Rango de medición

Según el teorema de NYQUIST define formalmente cual es la frecuencia máxima que se puede representar digitalmente a determinada frecuencia de muestreo. Se tiene la siguiente ecuación.

$$frecuencia\ maxima = \frac{frecuencia\ de\ muestreo}{2} = \frac{870Hz}{2} = 435Hz$$

El rango de medición está entre 0 – 435Hz

5.3.2 *Visualización de Medidas*

5.3.2.1 *Visualización de Medidas de Voltaje.*

Al activar el botón VOLTAJE se ejecuta la sentencia vol_on y se envía el mensaje vol_on al Gateway, como resultado nos envía las medidas del sensor y se visualiza en el navegador, como se muestra en la figura 113.

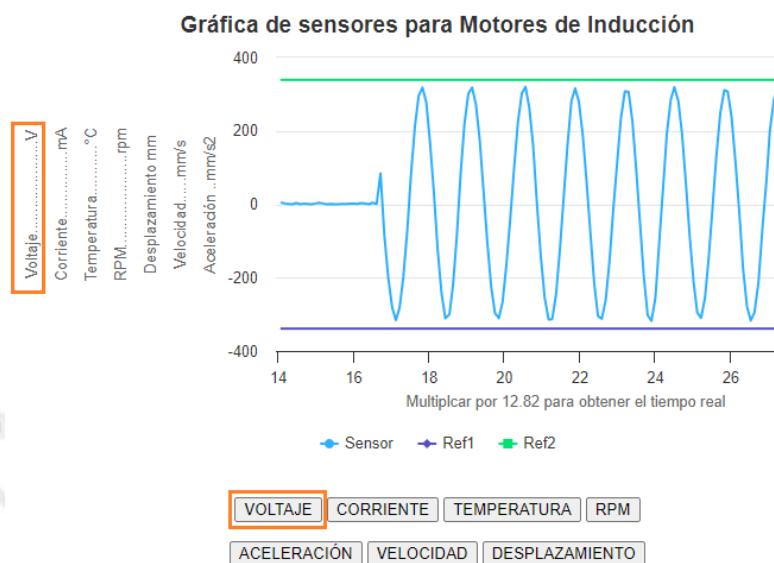


Figura 113: Visualización de voltaje.

Fuente: Elaboración propia.

5.3.2.2 Visualización de Medidas de Corriente.

Al activar el botón CORRIENTE se ejecuta la sentencia `cor_on` y se envía el mensaje `cor_on` al Gateway, como resultado nos envía las medidas del sensor y se visualiza en el navegador, como se muestra en la figura 114.

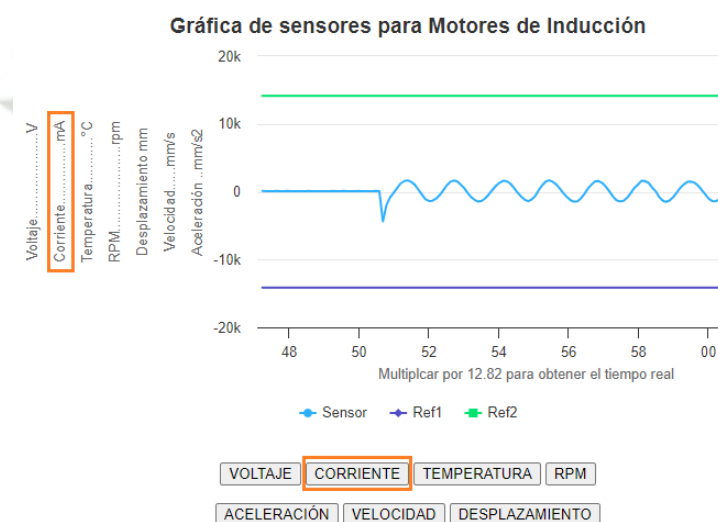


Figura 114: Visualización de corriente.

Fuente: Elaboración propia.

5.3.2.3 Visualización de Medidas de Temperatura.

Al activar el botón TEMPERATURA se ejecuta la sentencia tem_on y se envía el mensaje tem_on al Gateway, como resultado nos envía las medidas del sensor y se visualiza en el navegador, como se muestra en la figura 115.

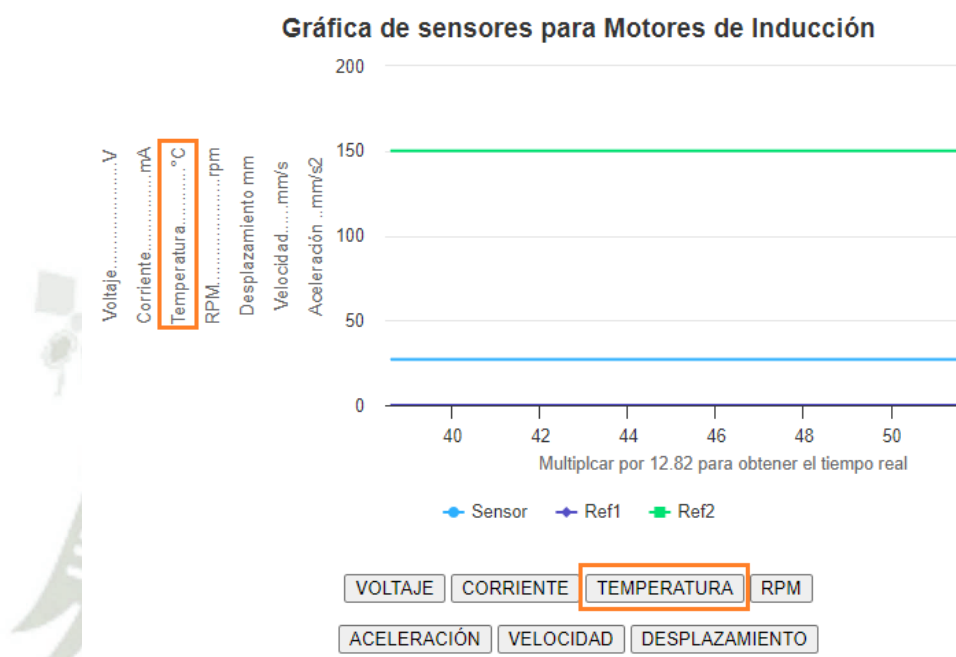


Figura 115: Visualización de temperatura.

Fuente: Elaboración propia.

5.3.2.4 Visualización de Medidas de Vibraciones.

- Aceleración, al activar el botón ACELERACIÓN se ejecuta la sentencia v_a_on y se envía el mensaje v_a_on al Gateway, como resultado nos envía las medidas del sensor y se visualiza en el navegador, como se muestra en la figura 116.

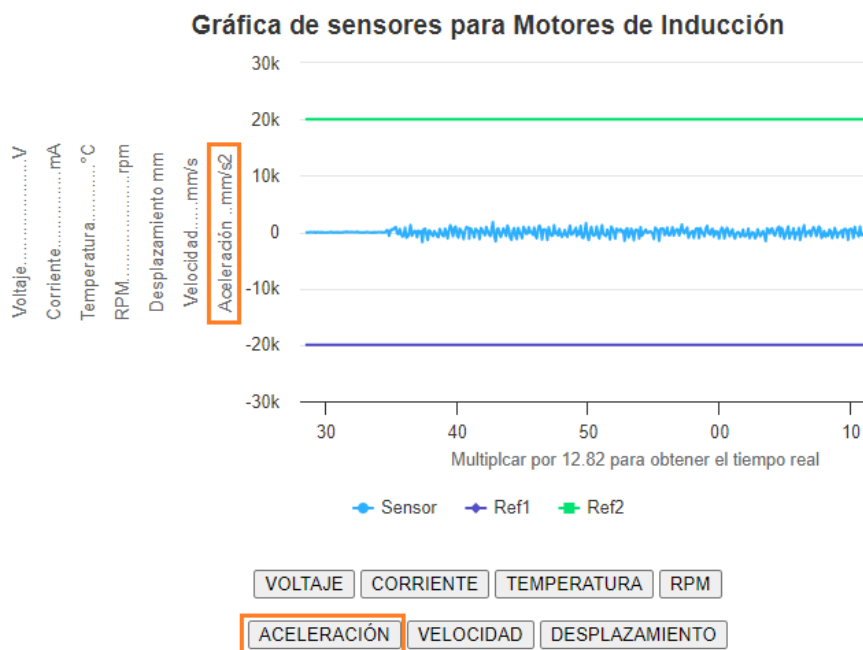


Figura 116: Visualización de la aceleración.

Fuente: Elaboración propia.

- Velocidad, al activar el botón VELOCIDAD se ejecuta la sentencia `v_v_on` y se envía el mensaje `v_v_on` al Gateway, como resultado nos envía las medidas del sensor y se visualiza en el navegador, como se muestra en la figura 117.

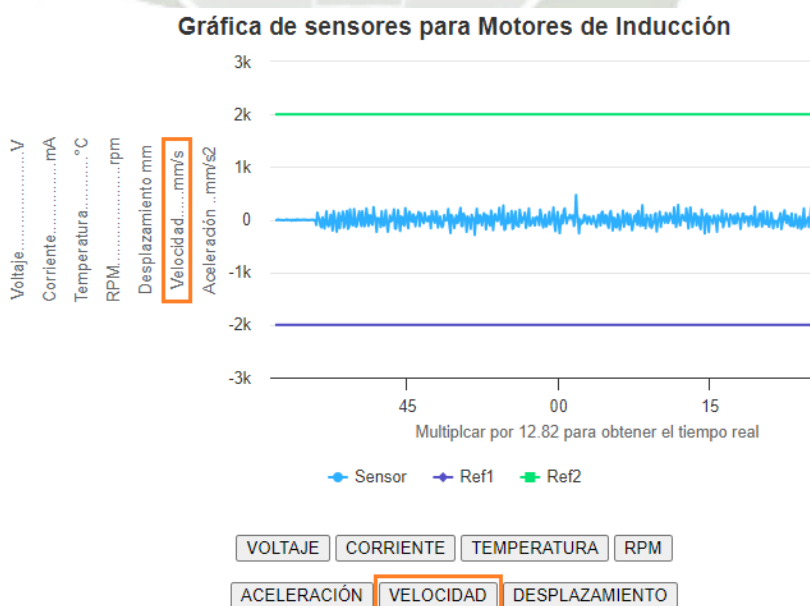


Figura 117: Visualización de la velocidad.

Fuente: Elaboración propia.

- Desplazamiento, al activar el botón DESPLAZAMIENTO se ejecuta la sentencia `v_d_on` y se envía el mensaje `v_d_on` al Gateway, como resultado nos envía las medidas del sensor y se visualiza en el navegador, como se muestra en la figura 118.

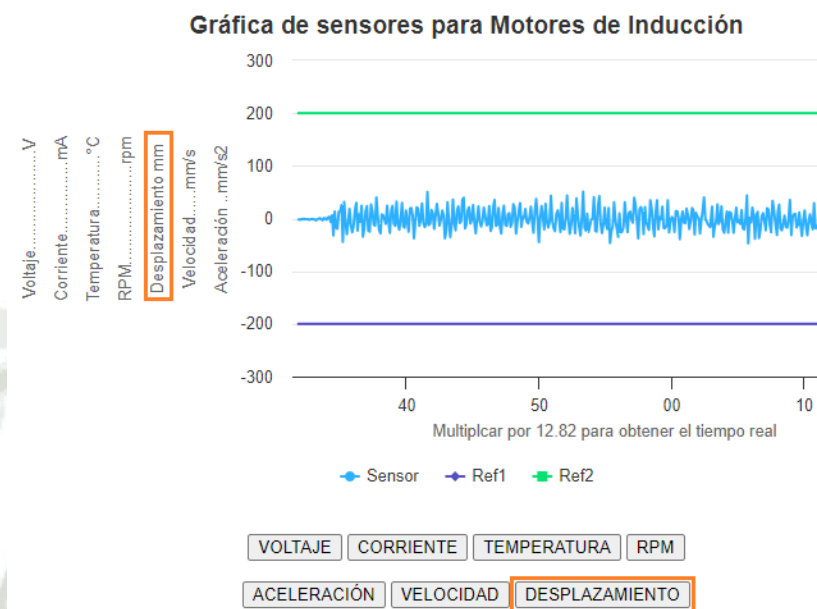


Figura 118: visualización del desplazamiento.

Fuente: Elaboración propia.

5.3.2.5 Visualización de Medidas de RPM.

Al activar el botón RPM se ejecuta la sentencia `rpm_on` y se envía el mensaje `rpm_on` al Gateway, como resultado nos envía las medidas del sensor y se visualiza en el navegador, como se muestra en la figura 219.

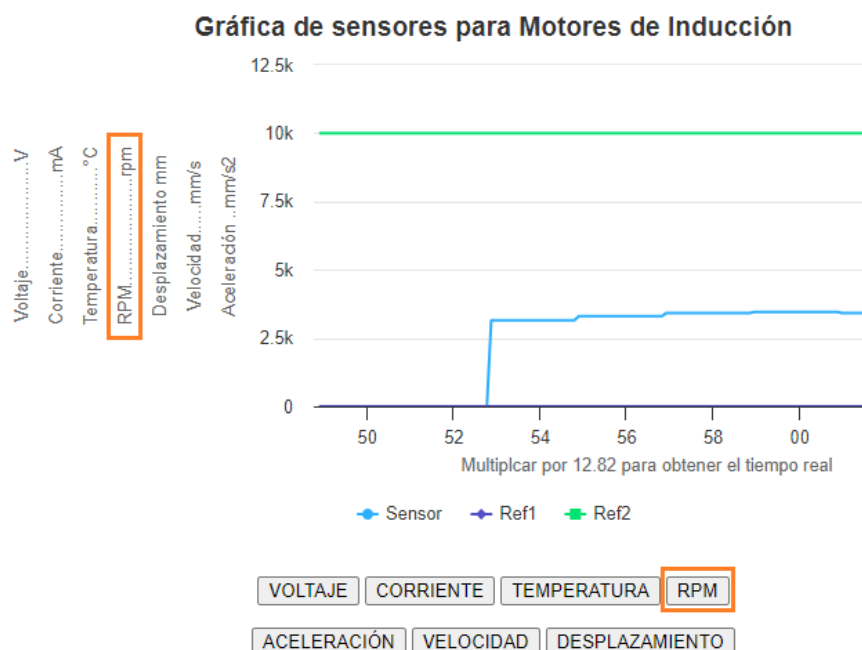


Figura 119: Visualización de RPM.

Fuente: Elaboración propia.

5.3.3 Interfaz donde se Muestran los Valores Numéricos.

Se ha implementado un interfaz para la visualización automática de las mediciones de amplitud provenientes de los sensores integrados en el sistema IOT.

Los detalles del interfaz están detallados el apartado Diseño del Programa (código HTML y JavaScript)

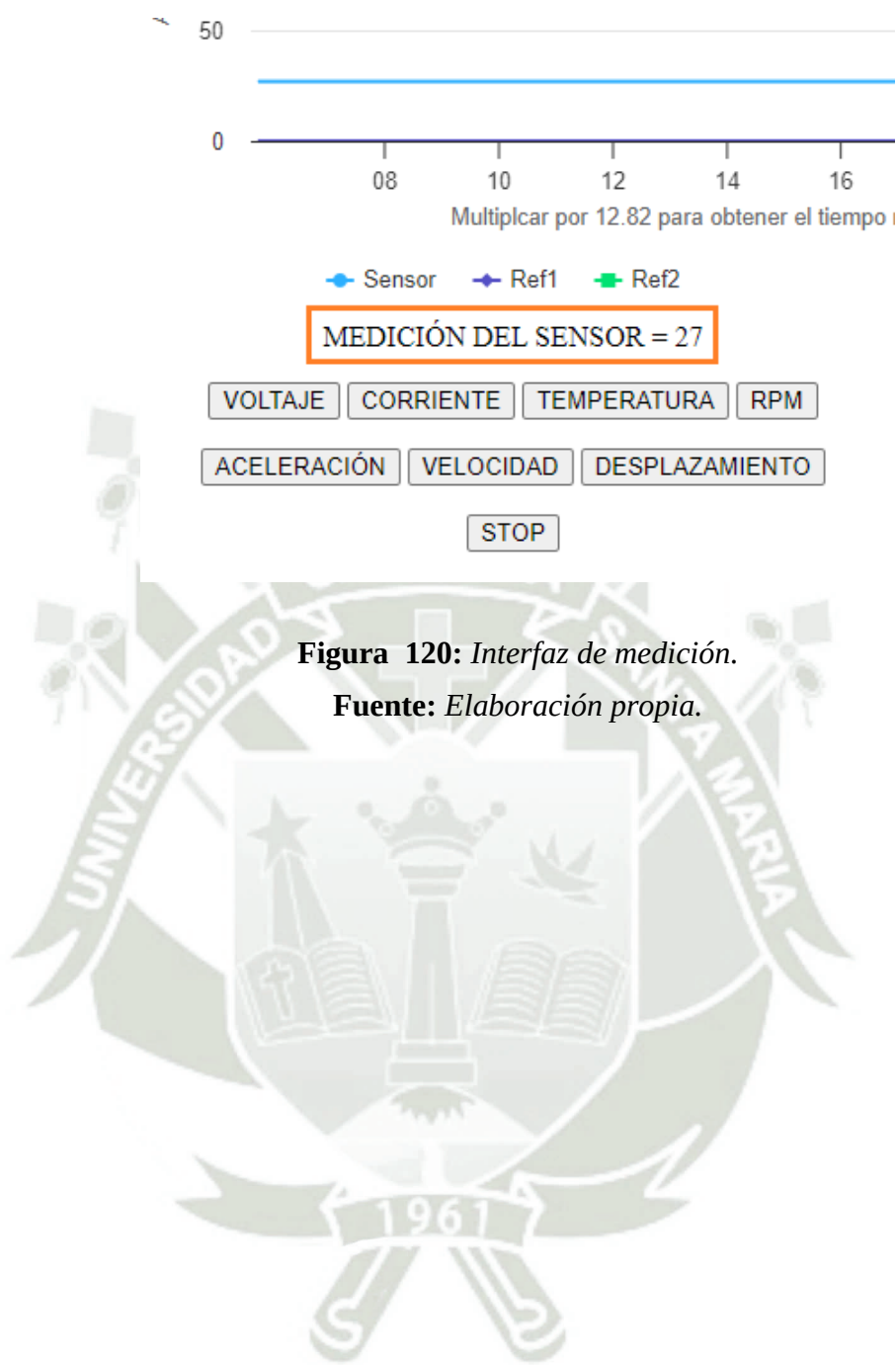


Figura 120: *Interfaz de medición.*

Fuente: *Elaboración propia.*



CAPITULO VI

PRUEBAS RESULTADOS Y DISCUSIÓN

6.1 Prototipo Terminado del Diseño

Como se precisó en capítulos anteriores, el diseño consta de sensores conectados a motores de inducción eléctricos para su transmisión de forma inalámbrica (vía Wifi) y en tiempo real (1.15ms de muestreo) vía Wifi a un sistema de visualización de gráficos, concretamente en una página web personalizada, que tiene soporte en la mayoría de navegadores web.

6.1.1 *Diseño de Hardware Terminado*

- El prototipo consta de una caja de paso de uso industrial de medidas 20x15x10cm con todos los elementos electrónicos y eléctricos, un motor de inducción eléctrico PRETUL de 0.5HP con un cable vulcanizado número 14 de 1.6m de longitud.
- Así mismo consta de una bomba de agua instalada en el eje del motor que está conectada a un recipiente de 4 litros mediante cañerías de ½ pulgada, ambos conectados de forma hermética, que ayudará hacer perturbaciones al sistema. Todos estos elementos están fijados en una plataforma de melamine de 15mm de medidas 42.5x33cm. como se aprecia en la figura 121



Figura 121: *Prototipo terminado.*

Fuente: *Elaboración propia.*

- Los elementos electrónicos y eléctricos contenidos en la caja de paso constan de una llave termomagnética de 16A, un contactor de 9A, dos lámparas de señalización ON/OFF, un interruptor de perilla, una fuente de alimentación de 9V, una placa de desarrollo ESP32 montado en su expansor de pines, 5 sensores y cables de instalación eléctricos. Como se muestra en la figura 122.



Figura 122: *Partes del prototipo.*

Fuente: *Elaboración propia.*

6.1.2 Diseño de Software Terminado

El software está programado en base a dos códigos fuente diferentes, un código llamado Sketch para la tarjeta de desarrollo ESP32 y el otro código llamado Script para el servidor web.

- Sketch terminado, se ha programado en el entorno de desarrollo Arduino IDE, el código completo se encuentra en el apartado anexos. Se muestra en la figura 123 el entorno de programación Arduino ID.

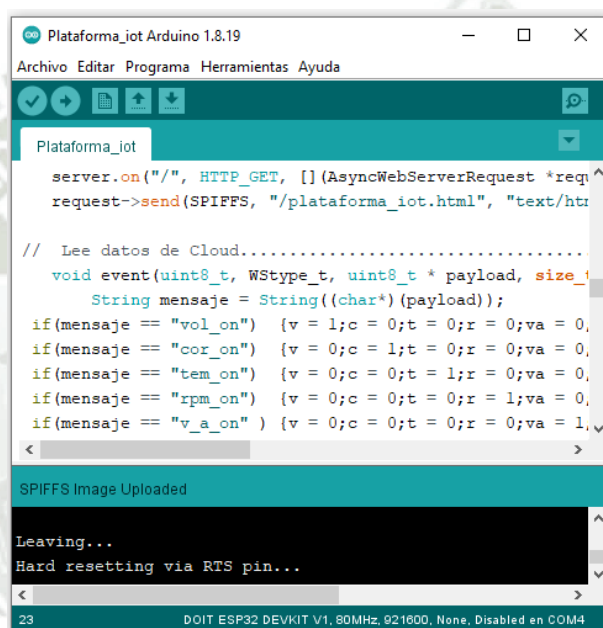


Figura 123: Código C++ en Arduino IDE.

Fuente: Elaboración propia.

- Script terminado, se ha programado en el entorno de desarrollo Visual Studio Code, el código completo se encuentra en el apartado anexos. Se muestra en la figura 124 la página web diseñada.

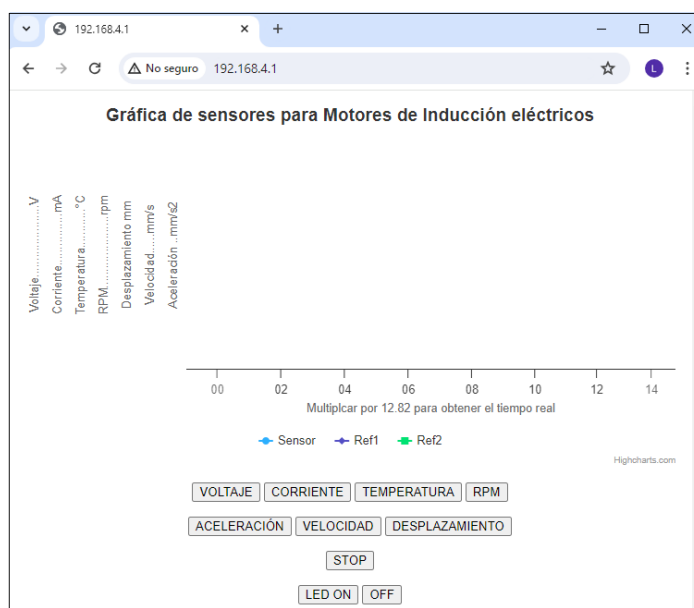


Figura 124: *Dashboard terminado.*

Fuente: *Elaboración propia.*

6.2 Pruebas y Mediciones

En este apartado se presentan los resultados obtenidos de las mediciones hechas por los 5 sensores utilizados en este diseño. También se ha medido los mismos parámetros utilizando instrumentos de medición para su valoración posterior.

6.2.1 Medición de Voltaje

- La medición de voltaje con el motor encendido, se muestra en la figura 125.

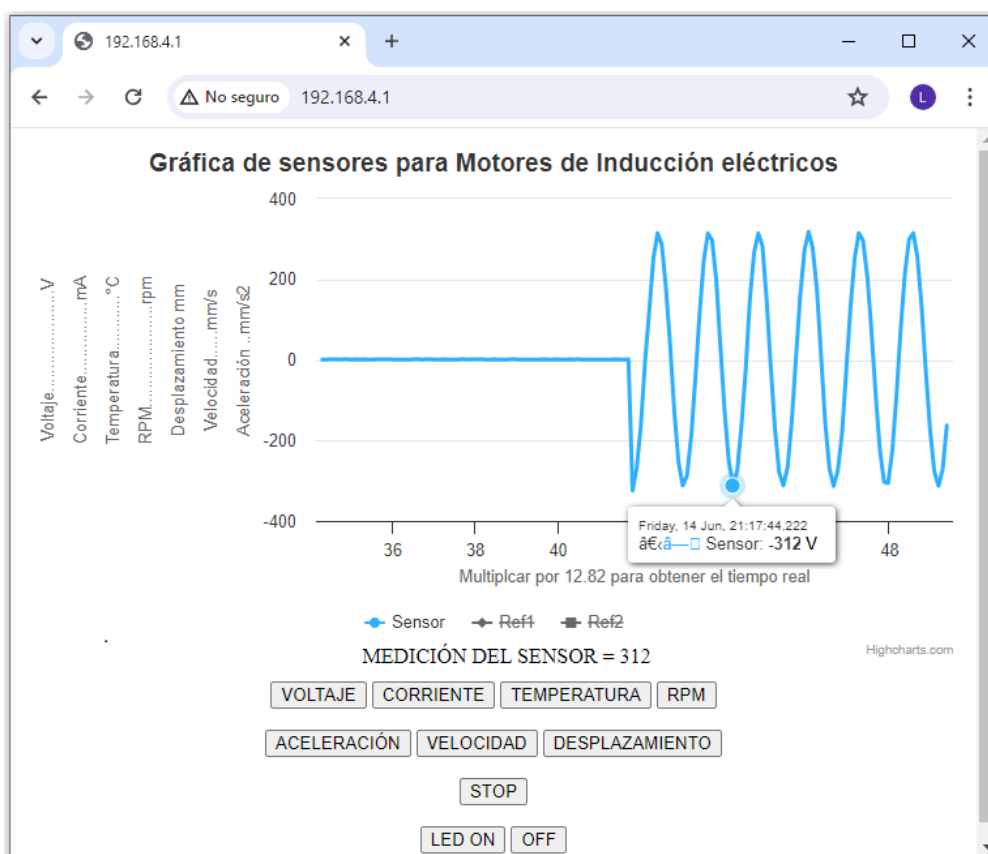


Figura 125: Resultados de voltaje.

Fuente: Elaboración propia.

El valor RMS se calcula con la siguiente ecuación.

$$Valor\ RMS = \frac{Valor}{\sqrt{2}}$$

$$VRMS = \frac{312}{\sqrt{2}} = 220.6V$$

El voltaje RMS de la medición de la plataforma IOT es 220.6V.

- La medición de voltaje con el motor encendido utilizando el multímetro de marca SANWA, se muestra en la figura 126.



Figura 126: Comprobación de voltaje.

Fuente: Elaboración propia.

El voltaje medido con el multímetro es 221.1V

- Se hizo 10 mediciones para ambos equipos con la finalidad de obtener un error promedio para su valoración posterior. Como se muestra en la tabla 21.

N° de mediciones	Plataforma	Multímetro	Error
1	220.6	221.1	0.5
2	220.8	220.1	0.7
3	221.4	220.4	1
4	221.3	220.3	1
5	221.4	221.4	0
6	220.1	221.7	1.6
7	220.5	220.4	0.1
8	220.4	221.4	1
9	220.1	221.1	1
10	220.7	221.7	1
Promedio	220.7	220.9	0.23

Tabla 21: Promedio de mediciones de voltaje.

Fuente: Elaboración propia.

Se obtuvo un error de medición promedio de 0.23V

6.2.2 Medición de Corriente

- La medición de corriente con el motor encendido, se muestra en la figura 127.



Figura 127: Resultados de corriente.

Fuente: Elaboración propia.

Se observa un pico de corriente de 6820 miliamperios al inicio de la gráfica, esto es porque el motor es una carga reactiva, es decir al momento del arranque es típico que consuma más corriente.

El valor de corriente RMS de la medición de la plataforma IOT es 818mA

$$V_{RMS} = \frac{1157mA}{\sqrt{2}} = 818mA$$

- La medición de corriente con el motor encendido utilizando la pinza amperimétrica de marca PRASEK, se muestra en la figura 128.



Figura 128: *Comprobación de corriente.*

Fuente: *Elaboración propia.*

La corriente medida con la pinza amperimétrica es 830mA

- Se hizo 10 mediciones para ambos equipos con la finalidad de obtener un error promedio para su valoración posterior. Como se muestra en la tabla 22.

N° de mediciones	Plataforma	Pinza	Error
1	818	830	12
2	820	830	10
3	820	820	0
4	819	820	1
5	819	830	11
6	817	820	3
7	820	820	0
8	818	830	12
9	817	830	13

10	820	820	0
Promedio	818.8mA	825mA	6.2mA

Tabla 22: Promedio de mediciones de corriente.

Fuente: Elaboración propia.

Se obtuvo un error de medición promedio de 6.2mA

6.2.3 Medición de Temperatura

- La medición de temperatura con el motor encendido, se muestra en la figura 129.

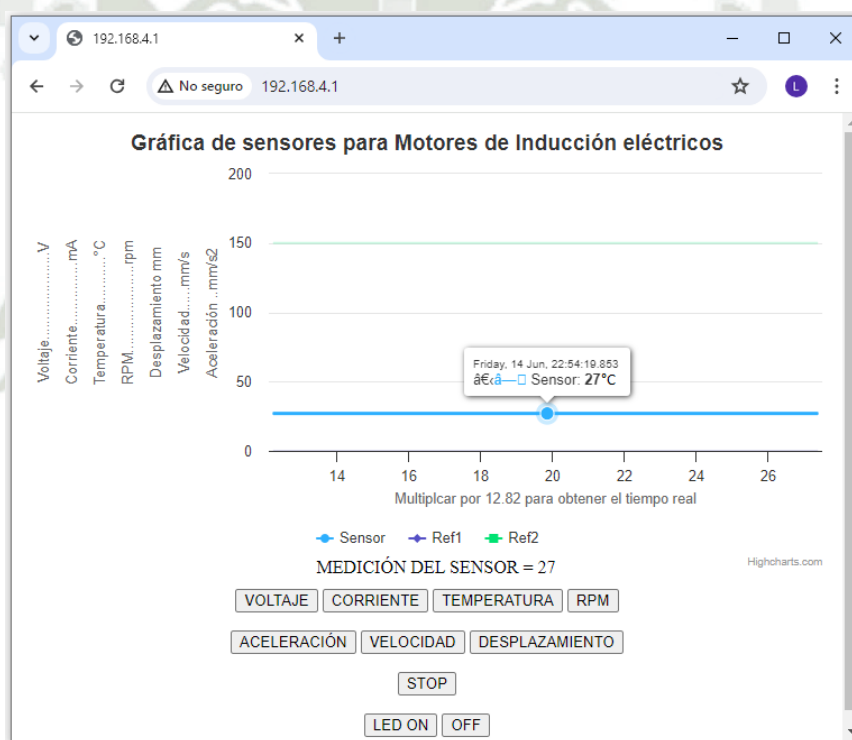


Figura 129: Resultados de temperatura.

Fuente: Elaboración propia.

El valor de temperatura de la medición de la plataforma IOT es 27°C

- La medición de temperatura con el motor encendido utilizando el termómetro digital de marca BOECO, se muestra en la figura 130.



Figura 130: Comprobación de temperatura.

Fuente: Elaboración propia.

El valor de temperatura de la medición con el termómetro digital es 27.1°C

- Se hizo 10 mediciones para ambos equipos con la finalidad de obtener un error promedio para su valoración posterior. Como se muestra en la tabla 23

N° de mediciones	Plataforma	Termómetro	Error
1	27	27.1	0.1
2	27	27.1	0.1
3	27	27.2	0.2
4	27	27.2	0.2
5	27	27.2	0.2
6	28	27.4	0.6
7	28	27.5	0.5
8	28	27.5	0.5
9	28	27.4	0.6
10	28	27.4	0.6
Promedio	27.5	27.3	0.2

Tabla 23: Promedio de mediciones de Temperatura.

Fuente: Elaboración propia.

Se obtuvo un error de medición promedio de 0.2°C

6.2.4 Medición de Vibraciones

La medición de vibraciones está definida en el rango de 0 – 435Hz de frecuencias para las tres magnitudes, aceleración, velocidad y desplazamiento. El cálculo de este rango está detallado en el apartado Diseño de Software.

- **Aceleración**

- ❖ La medición de la aceleración con el motor encendido, se muestra en la figura 131.

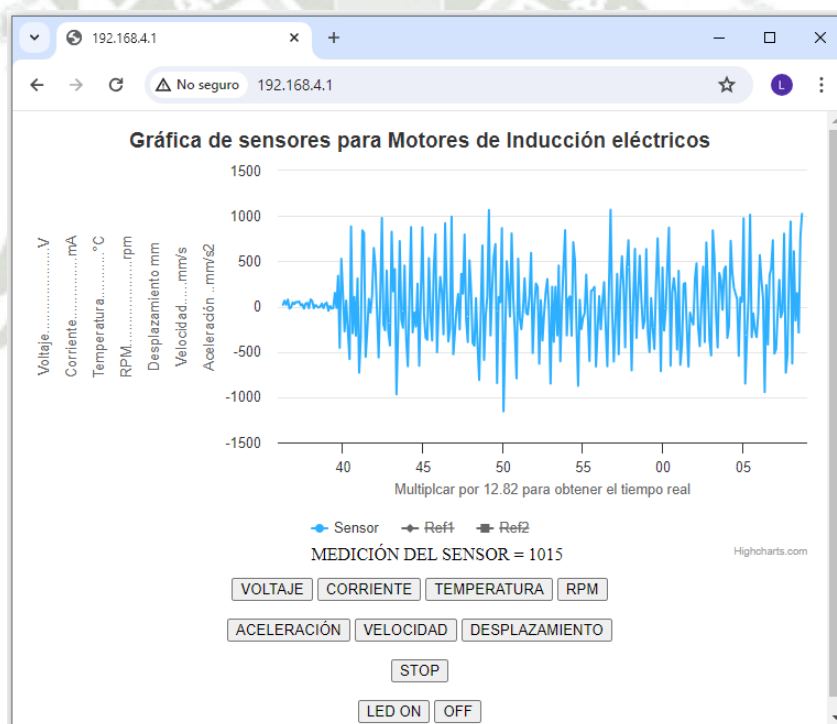


Figura 131: Resultados de aceleración.

Fuente: Elaboración propia.

- ❖ Para hacer la comparación de medidas se ha utilizado la gráfica de vibraciones de aceleración de una tienda virtual que ofrece mantenimiento a maquinas eléctricas en general como motores de inducción eléctricos (<https://traction.com>), como se muestra en la figura 132.

Especificaciones técnicas de la gráfica:

Eje X = tiempo en ms

Eje Y = amplitud en mm/s²

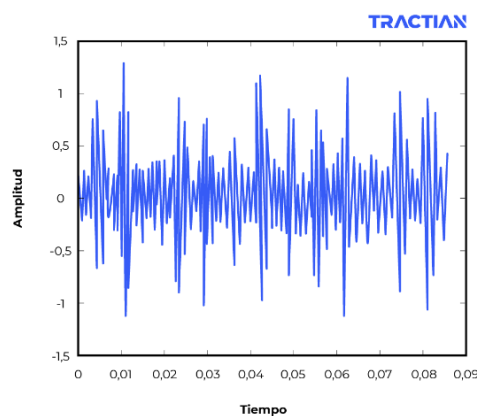


Figura 132: Gráfica de aceleración.

Fuente: TRACTIAN, análisis de vibraciones.

- ❖ Se visualizan ambas gráficas como se muestra en la figura 133 para su comparación.

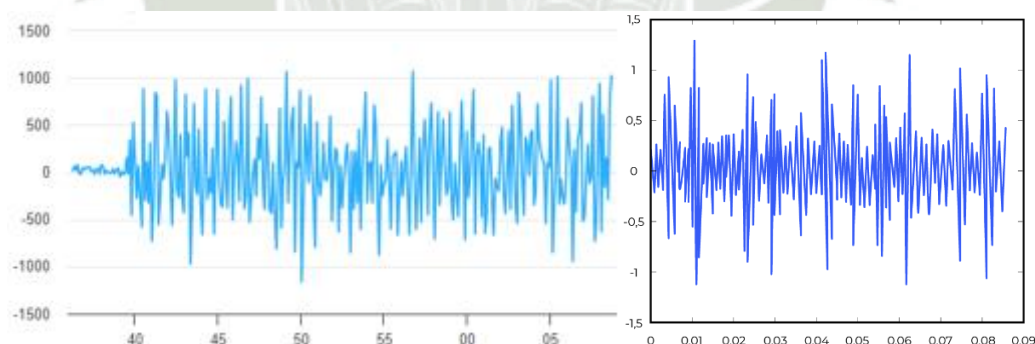


Figura 133: Comparación de gráficas de aceleración.

Fuente: TRACTIAN, análisis de vibraciones

- ❖ Se definen sus equivalencias de unidades, como se muestra en la tabla 24 para su comparación. ambas gráficas se miden en una ventana de tiempo de 90ms

Gráficas	Escala de aceleración	Escala del tiempo	Mediciones
Plataforma IOT	nm/s ²	ms x 12.82	1.0mm/s ²

Gráfica Tractian

mm/s2

segundos

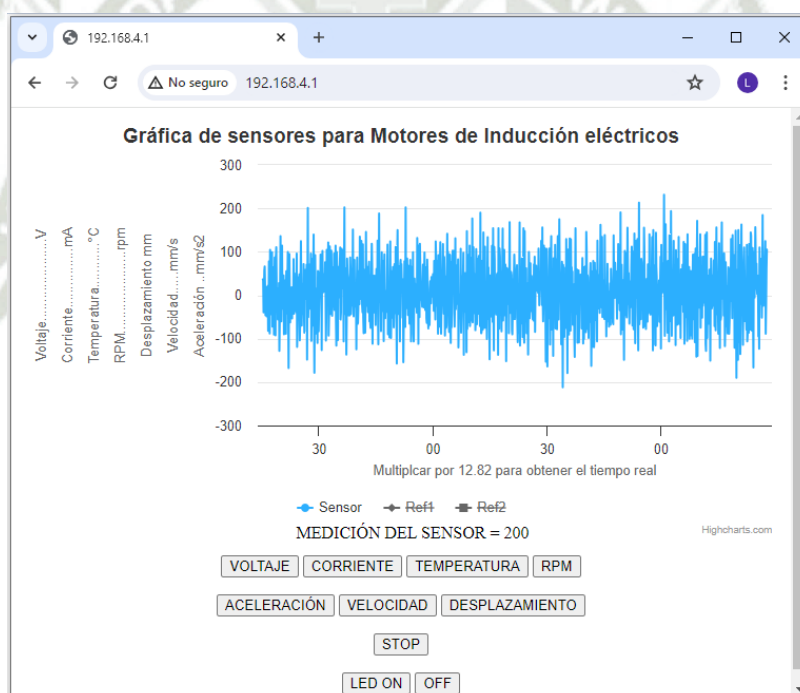
1.2mm/s2

Tabla 24: *Equivalencia de unidades de aceleración.*
Fuente: *Elaboración propia.*

Consideraciones previas. Para este estudio no es relevante hacer un análisis profundo de vibraciones de aceleración, pero si dar las bases para llevar a cabo esa tarea.

- **Velocidad**

- La medición de la velocidad con el motor encendido, se muestra en la figura 134.


Figura 134: *Resultados de velocidad.*
Fuente: *Elaboración propia.*

- ❖ Para hacer la comparación de medidas se ha utilizado la gráfica de vibraciones de 0.17mm/s de velocidad. de la academia virtual que imparte conocimientos en el área de las maquinas eléctricas y mantenimiento como motores de inducción

eléctricos (<https://www.academia.edu/38918569>), como se muestra en la figura 135.

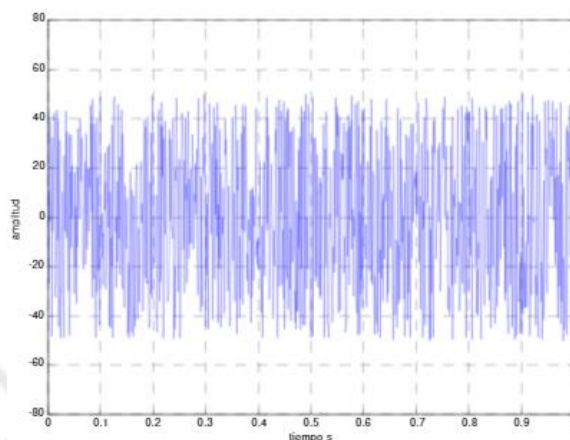


Figura 135: *Gráfica de velocidad.*

Fuente: *Academi.edu, Análisis de vibraciones categoría I.*

- ❖ Se visualizan ambas gráficas como se muestra en la figura 136 para su comparación.

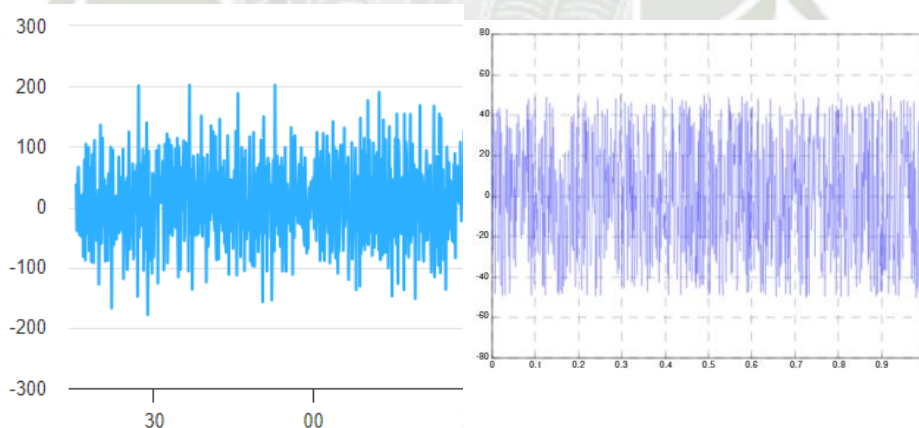


Figura 136: *Comparación de velocidades.*

Fuente: *Academi.edu, Análisis de vibraciones categoría I.*

- ❖ Se definen sus equivalencias de unidades como se muestra en la tabla 25 para su comparación, ambas gráficas se miden en una ventana de tiempo de 1000ms.

Gráficas	Escala de velocidad	Escala del tiempo	Mediciones
----------	---------------------	-------------------	------------

Plataforma IOT	nm/s	ms x 12.82	0.2mm/s
Gráfica categoría I	mm/s x 0.01	segundos	0.42mm/s

Tabla 25: *Equivalencia de unidades de velocidad.*

Fuente: *Elaboración propia.*

Para este estudio no es relevante hacer un análisis profundo de vibraciones de velocidad, pero si dar las bases para llevar a cabo esa tarea.

- ❖ En la figura 137 mostramos la tabla de severidad de la norma ISO 10816, que muestran los estándares de vibración recogidos de una serie de parámetros relacionados con las diferentes topologías tomando en cuenta su rigidez, flexibilidad y en el valor RMS de la velocidad de vibración.

								v r.m.s. mm/s	v r.m.s. inch/s	Velocidad Vibración 10 - 1000 Hz n > 600 1/min (2 - 1000 Hz n > 120 1/min)
								11	0.433	
								7.1	0.280	
								4.5	0.177	
								3.5	0.138	
								2.8	0.110	
								2.3	0.091	
								1.4	0.055	
								0.71	0.028	
rigida	flexible	rigida	flexible	rigida	flexible	rigida	flexible	Fundación		
Bombas> 15 kW radial, axial, diagonal				Máquinas medianas 15 kW < P ≤ 300 kW		Máquinas Grandes 300 kW < P < 50 MW		Tipo de Máquina		
Acople directo		Eje intermedio / Poleas.		Motores 160 mm ≤ H < 315 mm		Motores 315 mm ≤ H		Grupo		
Grupo 4		Grupo 3		Grupo 2		Grupo 1				
								A Puesta en operación recientemente.		
								B Operación para largo plazo.		
								C Operación para corto plazo		
								D Vibración causando daños		

Figura 137: *Cuadro de severidad ISO 10816.*

Fuente: *Rodesprex, normas de vibraciones.*

- ❖ Cálculo de valor RMS de velocidad

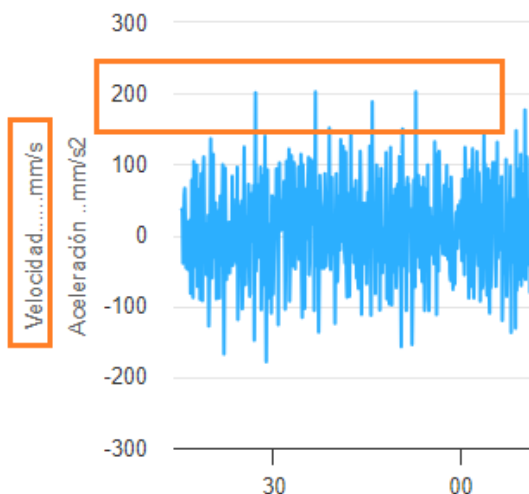


Figura 138: Valor de la velocidad 200mm/s.

Fuente: Elaboración propia.

Como se muestra en la figura 138 el valor máximo de velocidad es 0.200mm/s, calculamos el valor RMS con la siguiente ecuación.

$$v_{r.m.s} = \frac{\text{valor}}{\sqrt{2}} = \frac{0.2\text{mm/s}}{\sqrt{2}} = 0.14\text{mm/s}$$

El valor RMS de la velocidad es 0.14mm/s.

❖ Evaluamos los resultados en el cuadro de severidad.

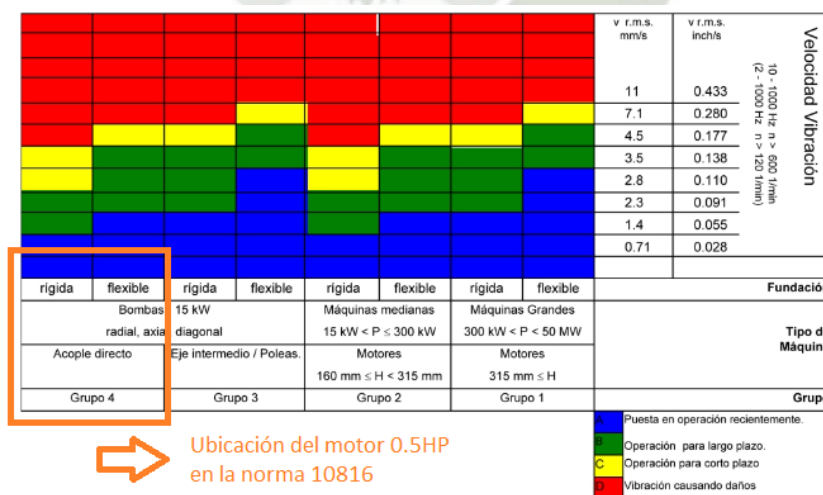


Figura 139: Ubicación del motor la ISO 10816.

Fuente: Rodesprex, normas de vibraciones.

Como se muestra en la figura 139, identificamos en la tabla de la norma 10816 la posición del motor de este estudio, considerando la base del motor (Rígido), la potencia del motor 0.5PH, el rango de frecuencias (entre 0 – 435 Hz) la velocidad en RMS del motor 0.14mm/s.

Considerando esta ubicación en la norma ISO 10816 de análisis de vibraciones de severidad, la medición de vibraciones se considera aceptable.

- **Desplazamiento**

- ❖ La medición de desplazamiento con el motor encendido, se muestra en la figura 140.



Figura 140: Resultados de desplazamiento.

Fuente: Elaboración propia.

- ❖ Para hacer la comparación de medidas se ha utilizado la gráfica de vibraciones de desplazamiento de la revista Scielo analytics (Rev. Fac. Ing. UCV.v24. n4. Caracas diciembre 2009), como se muestra en la figura 141.

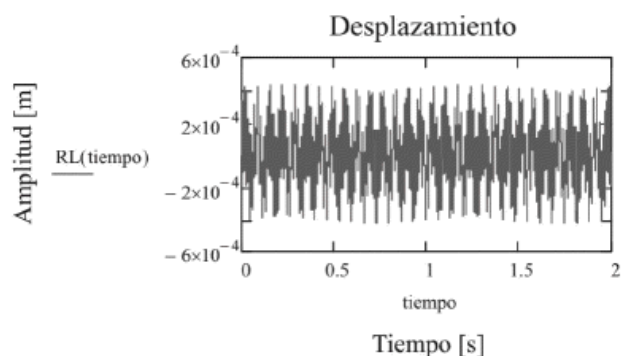


Figura 141: Gráfica de desplazamiento.

Fuente: Revista Scielo analytics, amortiguación dinámica como atenuación de vibraciones.

- ❖ Se visualizan ambas gráficas como se muestra en la figura 142 para su comparación.

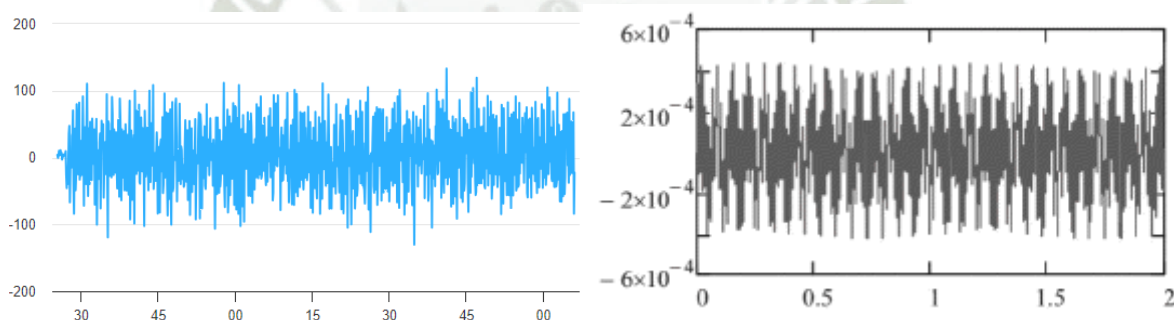


Figura 142: Comparación de desplazamiento.

Fuente: Revista Scielo analytics, amortiguación dinámica como atenuación de vibraciones.

- ❖ Se definen sus equivalencias de unidades como se muestra en la tabla 26 para su comparación. ambas gráficas se miden en una ventana de tiempo de 2000ms

Gráficas	Escala de desplazamiento	Escala del tiempo	Mediciones
Plataforma IOT	nm	ms x 12.82	0.1mm
Gráfica categoría I	m	segundos	0.4mm

Tabla 26: Equivalencia de unidades de desplazamiento.

Fuente: *Elaboración propia.*

Para este estudio no es relevante hacer un análisis profundo de vibraciones de desplazamiento, pero si dar las bases para llevar a cabo esa tarea.

6.2.5 Medición de RPM

- La medición de RPM con el motor encendido, se muestra en la figura 143.

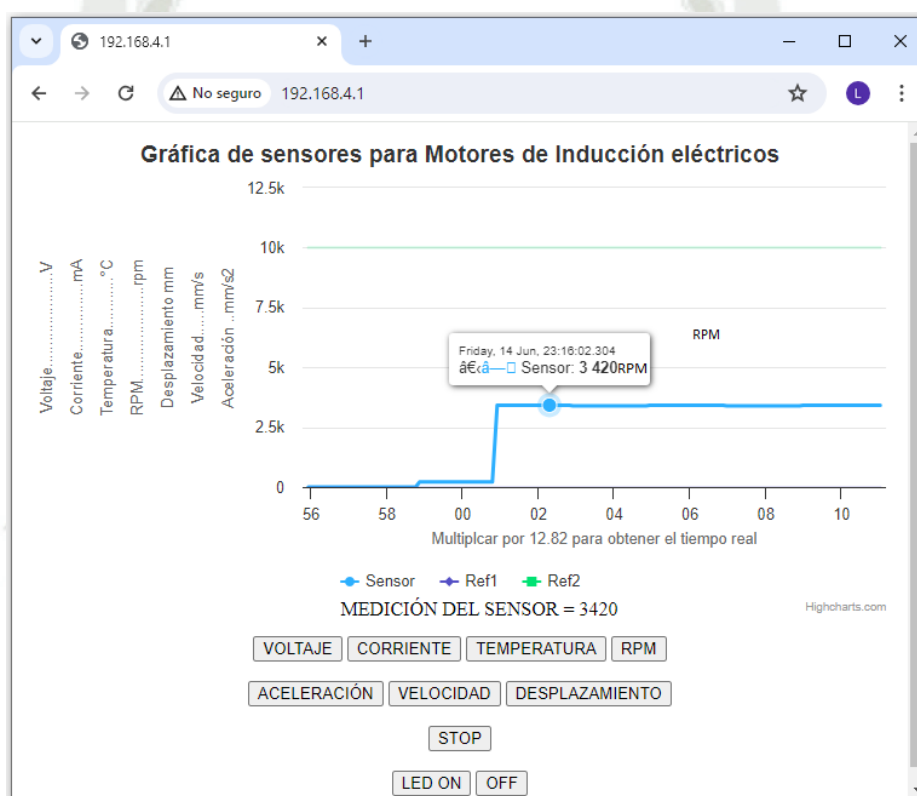


Figura 143: *Resultados de RPM.*

Fuente: *Elaboración propia.*

El valor de RPM de la medición de la plataforma IOT es 3420r/min

- La medición de RPM con el motor encendido utilizando el tacómetro digital de marca UNI-T, se muestra en la figura 144.

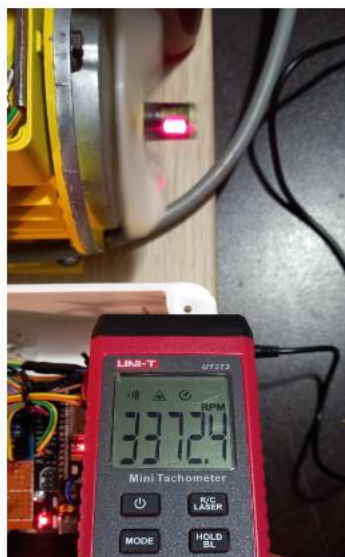


Figura 144: Comprobación de RPM.

Fuente: Elaboración propia.

El valor de RPM de la medición con el tacómetro digital es 3372 r/min

- Se hizo 10 mediciones para ambos equipos con la finalidad de obtener un error promedio para su valoración posterior. Como se muestra en la tabla 27.

N° de mediciones	Plataforma	Termómetro	Error
1	3420	3372	48
2	3420	3363	57
3	3420	3381	39
4	3450	3386	64
5	3450	3391	59
6	3450	3420	30
7	3420	3417	3
8	3420	3406	14
9	3455	3415	40
10	3455	3413	42
Promedio	3436	3396	40

Tabla 27: Promedio de mediciones de RPM.

Fuente: Elaboración propia.

Se obtuvo un error de medición promedio de 40r/min.

6.3 Hojas Finales de Especificaciones Eléctricas y de Funcionamiento

En este apartado se muestran las hojas finales de especificaciones eléctricas, de funcionamiento y generales que indican numéricamente los valores de parámetros de medición que se hizo en este estudio.

- **Especificaciones eléctricas.** En la tabla 28 se muestran un resumen de las especificaciones eléctricas de los dispositivos usados en el presente estudio.

Elementos	Características	Marca modelo código	Principio de funcionamiento
Motor monofásico	0.5 PH 3450r/min	Marca PRETUL	Inducción electromagnética
Llave termomagnética	16A	CHNT	Dilatación
Contactor	20A	CHNT	Inducción electromagnética
Lámparas	220VAC	CHNT	LED
Interruptor	3A	CHNT	Mecánico
Fuente de alimentación	9V 1A	TP-LINK	Conmutación
Sensor de voltaje	Relación de transformación $V_p / V_s = 377.0997$	Transformador	Inducción electromagnética
Sensor de corriente	30A/1V	SCT013	Inducción electromagnética
Sensor de temperatura	1°C/10mV	LM35	Transistor
Sensor de vibración	Sensibilidad = 2g	MPU-6050	Capacitivo
Sensor de RPM	Emisor: LED Receptor: fototransistor	AR-FC03	Óptico

Tabla 28: Especificaciones eléctricas del diseño.

Fuente: Elaboración propia.

- **Especificaciones de funcionamiento del diseño.** En la tabla 29 se muestran las especificaciones de generales de la plataforma IOT diseñada en el presente estudio.

Sensores	Rangos de medición
Voltaje	0 – 240VRMS
Corriente	0 – 10ARMS
Temperatura	0 – 150°C
Aceleración	0 – 20mm/s ²
Velocidad	0 – 2mm/s
Desplazamiento	0 – 0.2mm
RPM	0 – 10000
Muestreo de mediciones	1.15ms
Frecuencias	0 – 435Hz
Visualización	Navegadores web
Comunicación	Wifi
Dirección web	IP: 192.168.4.1
Configuración web	Access Point AP
ID	PlataformaIoT
Contraseña	11111111

Tabla 29: Especificaciones de funcionamiento del diseño.

Fuente: Elaboración propia.

6.4 Discusiones

La investigación realizada proporciona una comprensión detallada de la capacidad de la plataforma IOT desarrollada para medir con precisión diversas variables de operación en motores de inducción eléctricos. Se han analizado datos de mediciones de voltaje, corriente, temperatura, vibración, RPM, contrastando mediciones manuales con mediciones automatizadas a través de sensores.

1. Medición de Voltaje

Alta concordancia: Las mediciones de voltaje mostraron una alta concordancia entre las mediciones automático y manual. El error promedio es de 0.23VRMS en una medición de 0 a 240VRMS (Rango de voltaje establecido por diseño)

Discusión: Este pequeño error es una muestra que el diseño tiene una precisión importante y abre la posibilidad de utilizar esta aplicación IOT para diseñar en base a este estudio sistemas más complejos como variadores de velocidad, sistemas de control de fase y

otras aplicaciones que aporten soluciones a la problemática industrial acorde a las nuevas tecnologías.

2. Medición de Corriente

Alta Concordancia: Las mediciones de corriente mostraron una alta concordancia entre las mediciones automático y manual. El error promedio es de 6.2mARMS en una medición de 0 a 10000mARMS (Rango de corriente establecido por diseño)

Discusión: Este pequeño error es una muestra que el diseño tiene una precisión importante y abre la posibilidad de utilizar esta aplicación IOT para diseñar en base a este estudio sistemas más complejos como sistemas de control de potencia electrónico en motores usando tiristores accionados desde la nube y otras aplicaciones que aporten soluciones a la problemática industrial acorde a las nuevas tecnologías.

3. Medición de Temperatura

Alta Concordancia: Las mediciones de temperatura mostraron una alta concordancia entre las mediciones automático y manual. El error promedio es de 0.2°C en una medición de 0 a 150°C (Rango de temperatura establecido por diseño)

Discusión: Este pequeño error es una muestra que el diseño tiene una precisión importante y abre la posibilidad de utilizar esta aplicación IOT para diseñar en base a este estudio sistemas más complejos como análisis de fallas de motores en base a la temperatura de operación y otras aplicaciones que aporten soluciones a la problemática industrial acorde a las nuevas tecnologías.

4. Medición de Vibraciones

Alta Concordancia: Las mediciones de vibraciones mostraron una alta concordancia entre las mediciones automático y estándar. Se ha podido ubicar la vibración del motor usado en este estudio, en el cuadro de la norma ISO 10816.

Discusión: está alta concordancia de las vibraciones generadas por la plataforma IOT es una muestra que el diseño tiene una precisión importante y abre la posibilidad de utilizar esta aplicación IOT, para diseñar en base a este estudio sistemas más complejos como generar gráficas con respecto a la frecuencia FFT y otras aplicaciones que aporten soluciones a la problemática industrial acorde a las nuevas tecnologías.

5. Medición de RPM

Alta Concordancia: Las mediciones de RPM mostraron una alta concordancia entre las mediciones automático y manual. El error promedio es de 40r/min en una medición de 0 a 10000r/min (Rango de RPM establecido por diseño)

Discusión: Este pequeño error es una muestra que el diseño tiene una precisión importante y abre la posibilidad de utilizar esta aplicación IOT para diseñar en base a este estudio sistemas más complejos como análisis de fallas de motores en base a las RPM en motores y otras aplicaciones que aporten soluciones a la problemática industrial acorde a las nuevas tecnologías.

Discusiones abiertas:

El impacto y tiempo de aplicación del presente estudio se centra principalmente en la mejora de planes de mantenimientos predictivos en la industria 4.0, dado que las aplicaciones IOT aún están en gestación y la ciencia avanza a pasos agigantados, no se puede predecir cuándo pasará a ser obsoleto, sin embargo existen indicadores como la teoría de probabilidades por lo cual nos planteamos las preguntas ¿Qué probabilidades hay que los motores de inducción eléctricos pasen a ser obsoletos?, ¿Qué probabilidades hay que las aplicaciones IOT pasen a ser obsoletas?, ¿Qué probabilidades hay que las tarjetas de desarrollo pasen a ser obsoletas?, ¿Qué probabilidades hay que internet pase a ser obsoleto? y otras preguntas. En fin, para dar respuesta a estas interrogantes sin duda se tiene que hacer un estudio muy profundo.

Para indicar cuándo pasará a ser obsoleto este estudio lo definirá el avance de la ciencia y la tecnología, porque, aunque caduquen los elementos y dispositivos usados incluido los lenguajes de programación, siempre se podrá reemplazar por otros nuevos que surjan porque el algoritmo del estudio se aplica a cualquier tarjeta de desarrollo, se puede programar en cualquier lenguaje inclusive se puede comunicar en cualquier medio inalámbrico, puesto que las tecnologías venideras dan mejoras o evolucionan los dispositivos.

6.5 Consideraciones Generales

Mantenimiento: El mantenimiento regular de los sensores es fundamental para garantizar la precisión continua.

Aplicabilidad en la Industria: Considerar cómo las pequeñas variaciones en las mediciones podrían impactar en aplicaciones industriales específicas y si son aceptables en el contexto de los estándares de la industria.

Mejoras Futuras: Basándose en estos resultados, se pueden plantear mejoras en el diseño y la implementación de los sensores para aumentar la precisión y la fiabilidad en futuras aplicaciones.

Otras aplicaciones: Las aplicaciones IOT son muy amplias, como en áreas de (ingenierías, sociales, biomédicas etc.). Este diseño no es la excepción, se puede aplicar en bioingeniería, por ejemplo, una cirugía remota, donde el cirujano esté en otro país haciendo el procedimiento mediante actuadores al paciente en sala, a miles de kilómetros de distancia.

CONCLUSIONES

El objetivo general del presente estudio, era diseñar una plataforma IOT para la captura de medidas provenientes de sensores en motores de inducción eléctricos de forma inalámbrica vía Wifi con tiempos de muestreo menores a 50ms. La elaboración ha tenido un nivel de estudio importante; puesto que, la primera meta era la selección de componentes y dispositivos más adecuados tomando en cuenta muchos factores a considerar, la segunda meta era llegar a un tiempo de muestreo lo más bajo posible, la tercera meta era visualizar el alto tráfico de datos o medidas que ha generado el sistema y por último la cuarta meta era elaborar un selector de medición para no menguar el tiempo de muestreo. Dicho los puntos más críticos del presente estudio se ha llegado a las siguientes conclusiones.

- Se hizo una selección de componentes físicos de 3 candidatos preseleccionados y analizados para cada etapa funcional del diseño, dando buenos resultados que detallamos los más importantes a continuación.
 - ✓ Bomba de agua usado como carga para el motor de inducción, se concluye que la bomba y su circuito cerrado de circulación de agua fueron de gran utilidad al momento de hacer las pruebas y generar los resultados.
 - ✓ ESP32 usado como Gateway device, se concluye que es un elemento muy versátil y robusto debido a su capacidad de comunicarse vía wifi, capacidad de soporte a dos firmwares funcionando a la vez, que realmente hace a este dispositivo ideal para aplicaciones IOT.
 - ✓ Transformador usado como Edge device, se concluye que tiene una muy buena linealidad, no necesita alimentación y sobre todo tiene un buen aislamiento entre la red pública y la tarjeta de desarrollo.

- ✓ SCT013 usado como Edge device, se concluye que tiene una muy buena linealidad, no necesita alimentación y sobre todo un buen aislamiento entre la red pública y la tarjeta de desarrollo.
- ✓ LM35 usado como Edge device, se concluye que tiene una muy buena linealidad, fácil de escalar y amplio rango de medición.
- ✓ MPU6050 usado como Edge device, se concluye que tiene una sensibilidad alta, un ADC con buenas prestaciones y viene calibrado por fábrica con buena precisión.
- ✓ AR-FC03 usado como Edge device, se concluye que tiene una alta precisión para detectar objetos con mínimos falsos disparos, pero tiene la desventaja de consumir mucha corriente por lo cual no es ideal para proyectos que utilicen baterías o son portátiles.
- Se ha implementado el software utilizando los lenguajes de programación C++ y JavaScript dando muy buenos resultados, esto gracias a sus entornos de programación amigables y muy completos; a su vez, se ha desarrollado un tiempo de muestreo de 1.15 milisegundos superando ampliamente el objetivo de 50ms, esto se logró gracias al uso y adaptación del protocolo WebSocket en este estudio.
- Se ha implementado con éxito el selector de mediciones, para visualizar en una sola gráfica las variables (voltaje, corriente, temperatura, vibraciones y RPM), esto se logró gracias a la comunicación bidireccional entre Gateway y Cloud, que ha permitido el uso de botoneras como selector.
- Se ha visualizado todas gráficas de las variables propuestas en el presente estudio, es decir las mediciones de voltaje, corriente, temperatura, vibraciones, RPM de forma continua e interrumpida en el navegador web Google Chrome.

RECOMENDACIONES

- Se recomienda realizar investigaciones futuras en la integración de algoritmos de inteligencia artificial en la plataforma IOT. Esto podría permitir un análisis predictivo más avanzado y una detección aún más temprana de fallas, mejorando la eficiencia y la seguridad de los motores de inducción eléctricos.
- Se sugiere a las industrias consideren la implementación de la plataforma IOT no solo para el monitoreo y mantenimiento de maquinaria, sino también como base para desarrollar el control y automatización de procesos en la nube.



REFERENCIAS BIBLIOGRÁFICAS

- Ahlgren, B., Lindgren, A., & Wu, Y. (2016). Demo: Experimental Feasibility Study of CCN-lite on. (A. ISBN, Ed.) Obtenido de <https://conferences.sigcomm.org/acm-icn/2016/proceedings/p221-ahlgren.pdf>
- Alcácer, V., Rodrigues, C., Carvalho, H., & Cruz, V. (2021). Tracking the Maturity of Industry 4.0: The Perspective of a Real Scenario. *Int J Adv Manuf Technol*. doi:10.1007/s00170-021-07550-0
- Alonzo, J. (2010). Lenguaje de Programación: C++ Directivas al preprocesador. *Universidad de Guanajuato*. Obtenido de https://www.cimat.mx/~pepe/cursos/lenguaje_2010/slides/slide_27.pdf
- Barri, I., & Ruis, J. (2008). CoDiP2P: Sistema de Computación Distribuida Peer to Peer. Universidad de Lleida Escuela Politécnica Superior. Obtenido de <https://repositori.udl.cat/server/api/core/bitstreams/650d769b-6b1a-4421-9bff-0814f15849e6/content>
- Barrón, M. (2004). Uso didáctico del software de ayuda al diseño electrónico “Proteus”. *Departamento de Ingeniería de Sistemas y Automática*. Obtenido de <http://espacio.uned.es/fez/eserv/taee:congreso-2004-1034/S1F05.pdf>
- Bengoechea, S. (2016). PROPUESTA PARA LA MIGRACIÓN DE LA PLATAFORMA DE MÉTRICAS DE RENDIMIENTO BASADO EN GRAPHITE A INFLUXDB. Barcelona, España: Universidad politecnica de catalunya. Obtenido de <https://upcommons.upc.edu/bitstream/handle/2117/97314/TFC%20-%20Sergio%20Bengoechea%20Guerrero.pdf?sequence=1&isAllowed=y>

- Caballero, H., & Huertas, J. (2014). Diseño y desarrollo de un sistema de monitoreo de condición para motores de inducción. *Revista de Ingeniería*, 43-49. Obtenido de <https://ojsrevistaing.uniandes.edu.co/ojs/index.php/revista/article/view/561>
- Cembranos, F. (2014). Motores Síncronos. *Revista Digital de ACTA*. Obtenido de https://www.acta.es/medios/articulos/ciencias_y_tecnologia/029001.pdf
- Cháfer, B. (2020). Aplicación IOT para el diagnóstico de motores de inducción. España: Universidad politecnica de valencia. Obtenido de https://riunet.upv.es/bitstream/handle/10251/162629/%281%29_Ch%C3%A1fer%20-%20Dise%C3%B1o%20de%20Aplicaci%C3%B3n%20IoT%20para%20el%20diagn%C3%B3stico%20de%20motores%20de%20inducci%C3%B3n..pdf?sequence=1&isAllowed=y
- Condry, M., & Nelson, C. (2016). *Using Smart Edge IoT Devices for Safer, Rapid Response With Industry IoT Control Operations* (Vol. 104). Proceedings of the IEEE. doi:10.1109/JPROC.2015.2513672
- Cuartielles, D., & Banti, M. (s.f). Manual de programación Arduino. *Arduino Open Hardware*. Obtenido de <http://personales.upv.es/moimacar/master/download/arduino.pdf>
- Diepgen, T., Werner, T., Aberer, W., Jones, J., Cambazard, F., Elsner, P., . . . Coenraads, P. (2007). *Management of chronic hand eczema* (Vol. 57). doi:10.1111/j.1600-0536.2007.01179.x
- Farina, A. (2018). Motores eléctricos trifásicos: usos, componentes y funcionamiento. Obtenido de https://www.editores-srl.com.ar/sites/default/files/ie330_farina_motores_electricos.pdf

Flores, G. (2023). *DISEÑO DE UN PROTOTIPO UTILIZANDO IoT PARA SUPERVISION DE*. Puno, Perú: UNIVERSIDAD NACIONAL DEL ALTIPLANO DE PUNO. Obtenido de <https://vriunap.pe/fedu/upload/2023/p00000716-6-Proy.pdf>

Fundación Red de Energía - BUN-CA. (2011). Manual técnico: Motores eléctricos. San José, C.R. : Biomass Users Network (BUN-CA), 2011. Obtenido de <https://www.bunca.org/wp-content/uploads/2019/02/Motores.pdf>

Garcia, J. (2016). Librerías de Arduino. Tecnología. IES Bellavista. Obtenido de <https://blogsaverroes.juntadeandalucia.es/iesbellavista/files/2016/02/LIBRERIAS-DE-ARDUINO.pdf>

Garcia, J. (2022). *Diseño e implementación de una plataforma IoT para el control y monitoreo remoto del banco de psicrometría del Laboratorio de Energía de la PUCP*. Lima, Perú: PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ. Obtenido de https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/25760/2022_GARC%c3%8da_MACAVILCA_JOSE_ANTHONY_DISE%c3%91O_IMPLEMENTACION_PLATAFORMA_IOT_TESIS.pdf?sequence=1&isAllowed=y

Garcia, L., Garcia, A., & Noriega, J. (2019). Diseño Electrónico con Microcontroladores 02 - Estructura de un Programa. México: Universidad de Sonora. Obtenido de https://sivea.uson.mx/docentes/tareas/microcontroladores_parte_2.pdf

Garcia, M., & López, M. (2019). Industria 4.0 para la monitorización de un proceso industrial. Ambato, Ecuador: Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Maestría en Automatización y Sistemas de Control. Obtenido de https://repositorio.uta.edu.ec/bitstream/123456789/29852/1/Tesis_%20t1591masc.PDF

F

Hernandez, D. (2018). *Simulador de sensores de temperatura*. España: Universitat Politècnica de València. Obtenido de

<https://riunet.upv.es/bitstream/handle/10251/110767/Hern%C3%A1ndez%20-%20Simulador%20de%20sensores%20de%20temperatura.pdf?sequence=1&isAllowed=y>

Huacho, W. (2022). *Diseño de un módulo de control basado en plataforma IOT para el monitoreo remoto de motores de inducción de baja potencia*. Huancayo - 2022.

Universidad Continental. Obtenido de https://repositorio.continental.edu.pe/bitstream/20.500.12394/12585/3/IV_FIN_109_TE_Huacho_Ichpas_2022.pdf

Huamán, L. (2020). *Elección óptima de arranque de motores eléctricos trifásicos asíncronos según su uso en la industria*. Arequipa, Perú: Universidad Continental. Obtenido de https://repositorio.continental.edu.pe/bitstream/20.500.12394/10047/1/IV_FIN_109_TE_Huaman_Cruz_2020.pdf

Hussain, M., Butt, A., & Uzma, F. (2019). *A comprehensive review of climate change impacts, adaptation, and mitigation on environmental and natural calamities in Pakistan* (Vol. 192). Environ Monit Assess . doi:<https://doi.org/10.1007/s10661-019-7956-4>

IEC 60034-30. (2008). *Potencia nominal del motor de inducción de polea trifásico de una sola velocidad*. Obtenido de <https://library.e.abb.com/public/6893d307b1d8460eaa91174edc0dd50e/Fuerza%20impulsora.pdf>

Janani, R., Elavarasan, R., Perumal, M., Divya, P., & Baskaran, V. (2021). *Astaxanthin mediated regulation of VEGF through HIF1 α and XBP1 signaling pathway: An insight*

from ARPE-19 cell and streptozotocin mediated diabetic rat model. *Experimental Eye Research*. Obtenido de <https://doi.org/10.1016/j.exer.2021.108555>

Khudhair, A., & Ghani, R. (2020). *IoT Based Smart Video Surveillance System Using Convolutional Neural Network*. Erbil, Iraq. Obtenido de <https://ieeexplore.ieee.org/abstract/document/9122901>

Martinez, N. (2022). Sistema IoT para monitorizacion de motores eléctricos y deteccion de anomalías mediante el analisis de vibraciones de rodamientos: una revisión de literatura. *Universidad de Oviedo*.

Mejia, A., & Armijos, F. (2015). Carcterización de los sensores y actuadores del motor armfield Volkswagen CM11-306. Cuenca: Universidad Politecnica Salesiana. Obtenido de <https://dspace.ups.edu.ec/bitstream/123456789/7537/1/UPS-CT004468.pdf>

Mercedez , R., & Guzman, A. (2014). CARACTERIZACIÓN DE UN MOTOR DE COMBUSTIÓN INTERNA CON DOS TIPOS DE COMBUSTIBLE. *Instituto Mexicano de transporte*. Obtenido de <https://www.imt.mx/archivos/Publicaciones/PublicacionTecnica/pt417.pdf>

Meski, O., Belkadi, F., Laroche, F., & Furet, B. (2019). *Towards a knowledge-based framework for digital chain monitoring within the industry 4.0 paradigm* (Vol. 84). *Procedia CIRP*. Obtenido de <https://pdf.sciencedirectassets.com/282173/1-s2.0-S2212827119X00074/1-s2.0-S2212827119308972/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEJr%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJIMEYCIQC2UTMXrIFPGI%2FAr3X8MA%2FD8K%2B5%2BukkisfiazJBuQUV4QIhAKjS>

NORMA IEC 62052-11:2020. (2020). *Dispositivo de medición eléctrica. Requisitos generales, pruebas y condiciones de prueba*. Obtenido de [https://www.eurolab.net/es/testler/emc-kablosuz-ve-elektriksel-testler/iec-62052-elektrik-olcum-ekipmani-\(ac\)-genel-gereksinimler-testler-ve-test-kosullari/](https://www.eurolab.net/es/testler/emc-kablosuz-ve-elektriksel-testler/iec-62052-elektrik-olcum-ekipmani-(ac)-genel-gereksinimler-testler-ve-test-kosullari/)

NORMA IEC 62053-21:2020. (2021). *(Contador de electricidad. Requisitos especiales)*. Obtenido de <https://www.laboratuar.com/es/testler/elektrik-elektronik-testleri/iec-en-62053-21-elektrik-olcum-ekipmani---ozel-gereksinimler---bolum-21:-ac-aktif-enerji-icin-statik-sayaclar-test-standardi/>

Norma Técnica Peruana 399.450:2008. (29 de setiembre del 2008). *Eficiencia energética para motores de uso general*. Obtenido de <https://www.minem.gob.pe/minem/archivos/file/DGEE/eficiencia%20energetica/ANEXOS%2015%20FICHAS%20DE%20HIMOLOGACION%20MOTORES%20ELECTRICOS.pdf>

Ochoa, C. (2012). Implementacion de un diseño de puente inalambrico punto multipunto para la mejora de la interconexion de las areas de la empresa plásticos Rimac SRL. Chiclayo: Universidad Católica Santo Toribio de Mogrovejo. Obtenido de https://tesis.usat.edu.pe/bitstream/20.500.12423/517/1/TL_Ochoa_Saavedra_Cesar.pdf

PNTPE IEC 60034-2-1:2009. (2009). *Máquinas Eléctricas Rotativas: Método Estándar para Determinar Pérdidas y Eficiencias Experimentales (Mecánica para Unidad de Tracción No Incluida)*. Obtenido de <https://upcommons.upc.edu/bitstream/handle/2099.1/5455/?sequence=1>

Pomares, J. (2009). Manual de Arduino. *Grupo de innovación Educativa en Automática*. Obtenido de <https://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf>

- Quilla, D. (2023). *Diseño e implementación de un prototipo de medidor inteligente monofásico para el monitoreo de energía en hogares empleando tecnología de internet de las cosas (IOT)*. Arequipa, Perú: Universidad Católica de Santa María. Obtenido de <https://repositorio.ucsm.edu.pe/bitstream/handle/20.500.12920/12879/4E.0460.IM.pdf?sequence=1&isAllowed=y>
- Rios , G. (2008). Series de Tiempo. Universidad de Chile. Obtenido de https://www.u-cursos.cl/ingenieria/2010/1/CC52A/1/material_docente/bajar?id_material=296003
- Rodriguez, M. (2015). Motores Monofasicos de Induccion. (U. d. Cantabria, Ed.) Obtenido de <https://personales.unican.es/rodrigma/PDFs/Motores%20monofasicos%20de%20inducccion.pdf>
- Romero, D. (2015). DISEÑO DE UN SISTEMA DE MONITOREO DE PARÁMETROS ELÉCTRICOS BASADO EN TECNOLOGÍA GSM PARA UN RIOGENERADOR PUCP. PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ. Obtenido de https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/6230/ROMERO_DIEGO_MONITOREO_PARAMETROS_ELECTRICOS_GSM_RIOGENERADOR.pdf?sequence=1&isAllowed=y
- Salvivar, A., Diaz, R., Reyes, N., Armenta, C., Fuensanta, L., Moreno, M., . . . Dominguez, M. (2015). *Roles de Género y Diversidad: Validación de una Escala en Varios Contextos* (Vol. 5). México: ACTA DE INVESTIGACIÓN PSICOLÓGICA. Obtenido de <https://www.redalyc.org/pdf/3589/358943649003.pdf>
- S.D.H.N.T.Q Pengfei Hu. (27-42, 2017). Surveyon fog computing: architecture, key technologies, application and open issues Journal of Network and Computer Applications.

Shivers, O. (2005). The Anatomy of a Loop A story of scope and control. *Georgia Institute of Technology*. Obtenido de

<https://www.khoury.northeastern.edu/home/shivers/papers/loop.pdf>

Tewari, D., & Damilola, K. (2020). *Teaching and learning interaction in South Africa's higher education: Some weak links*. Cogent Social Sciences. Obtenido de https://www.researchgate.net/publication/340069859_Teaching_and_learning_interaction_in_South_Africa's_higher_education_Some_weak_links/fulltext/60a37993299bf1d21d6e0826/Teaching-and-learning-interaction-in-South-Africas-higher-education-Some-weak-links.p

Vasconez, D. (2014). “RED INALÁMBRICA TIPO MALLA (WNM) ESTANDAR 802.11 DE TRANSMISIÓN Y LA OPTIMIZACIÓN DE COBERTURA EN LOS COLEGIOS DE LA PROVINCIA DE TUNGURAHUA”. Ambato, Ecuador: UNIVERSIDAD TÉCNICA DE AMBATO. Obtenido de https://repositorio.uta.edu.ec/bitstream/123456789/6989/1/Tesis_t873mrt.pdf

Vieto, J. (08 de agosto de 2020). *El efecto de la variación del voltaje en los motores eléctricos* - vyphidroasesores.com. Obtenido de vyphidroasesores.com: <https://vyphidroasesores.com/el-efecto-de-la-variacion-del-voltaje-en-los-motores-electricos/>

Vital, M. (2021). Introducción de Arduino. *Vida Científica Boletín Científico de la Escuela Preparatoria*, 9(17). Obtenido de <https://repository.uaeh.edu.mx/revistas/index.php/prepa4/article/download/6625/7531/>

Ycaza, M. (2019). DISEÑO E IMPLAMENTACIÓN DE LA ELECTRÓNICA DE UN SISTEMA DE SENSADO DE LAS VIBRACIONES DE UNA ESTRUCTURA METÁLICA. Lima, Perú: PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ.

Obtenido

de

[https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/14079/YCAZA_L
ENGUA_MARCO_DISE%C3%91O_IMPLEMENTACI%C3%93N_ELECTR%C3
%93NICA.pdf?sequence=1](https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/14079/YCAZA_L
ENGUA_MARCO_DISE%C3%91O_IMPLEMENTACI%C3%93N_ELECTR%C3
%93NICA.pdf?sequence=1)

Zanella, A., Member, S., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for Smart Cities. 1(1). Obtenido de https://www.researchgate.net/profile/Andrea-Zanella-2/publication/260540297_Internet_of_Things_for_Smart_Cities/links/541ac8880cf25ebee988c1a6/Internet-of-Things-for-Smart-Cities.pdf?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1Ymxp

ANEXOS

ANEXO A: PROGRAMA GENERAL

```
// librerias .....
#include <WiFi.h>           // para conectarse a Wifi
#include <ESPAsyncWebServer.h> // para crear un servidor web
#include <SPIFFS.h>         // para subir archivos a la memoria Flash del ESP32
#include <WebSocketsServer.h> // para comunicación instantanea entre
gateway/Cloud/cliente
#include <ArduinoJson.h>    // para encapsulación Json, enviar en una cadena multiples
variables
#include <Wire.h>           // para comunicarse por I2C
#include <Adafruit_ADS1X15.h> // para utilizar el ADS1115
#include "MPU6050.h"       // para utilizar el MPU6050

// Declara objetos .....
Adafruit_ADS1115 ads;      // declara objeto ads
MPU6050 acelerometro;     // declara objeto acelerometro
AsyncWebServer server(80); // declara objeto server
WebSocketsServer websocket(81); // declara objeto websockets
StaticJsonDocument<200> doc; // crea un objeto JSON container
JsonObject object = doc.to<JsonObject>(); // crea un segundo objeto JSON

// declara pines .....
#define Led 2              // declara pin 2
#define Pin 4              // declara pin 4

// Declara variables .....
int v = 0; int c = 0; int t = 0; //
int r = 0; int va = 0; int vv = 0; // Variables para almacenar valores de control
int vd = 0; int p = 0;          //

float vol = 0;                 // Variable para almacenar el valor obtenido del sensor de voltaje
float cor = 0;                 // Variable para almacenar el valor obtenido del sensor de corriente
float tem = 0;                 // Variable para almacenar el valor obtenido del sensor de
Temperatura
int rpm = 0;                   // Variable para almacenar el valor obtenido del sensor de
velocidad
int16_t v_x, v_y, v_z, gx, gy, gz; // Variables para inicializar el sensor de vibracion
float v_d, v_v, v_a;           // Variables para almacenar valores del sensor de vibracion

volatile int i = 0;            // Variable para almacenar en memoria RAM, contador de
interrupción
unsigned long tiempo, tr=0;    // Variable para almacenar los millis
String jsonString;            // Variable para almacenar una cadena JSON
```

```
void setup(void) {

// Define interrupción y pines.....
attachInterrupt(35,interrupcion,RISING); // Define interrupcion (pin35)
pinMode(Led, OUTPUT);                // Define pin 2 como salida
pinMode(Pin, INPUT);                 // Define pin 4 como entrada

// Inicia comunicación serial .....
Serial.begin(115200);                // Define a 115200 baudios
// Inicia comunicación WIFI .....
WiFi.softAP("PlataformaIoT", "11111111");// Defini access point y credenciales de red
Serial.println(WiFi.softAPIP());     // Imprime por monitor serie la IP la de red
delay (5000);                        // Espera 5 segundos

// Inicia librerias.....
Wire.begin();                       // Habilita Wire para comunicarse por I2C
acelerometro.initialize(); // Habilita el MPU6050
ads.begin();                       // Habilita el ADS1115
ads.setGain(GAIN_ONE);             // Ganacia del PGA del ADS1115
server.begin();                    // Habilita el servidor (Cloud)
websocket.begin();                 // Habilita la comunicación instantanea
websocket.onEvent(event); // Habilita un evento
SPIFFS.begin();                    // Habilita el SPIFFS para subir archivos de cloud a flash

// Define dirección de Cloud y almacenamiento en memoria Flash .....
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){ // Define fichero
plataforma_iot para
    request->send(SPIFFS, "/plataforma_iot.html", "text/html");}); // almacenar script y subir
al ESP32

// Lee datos de Cloud.....
void event(uint8_t, WStype_t, uint8_t * payload, size_t){ // Recepción de llamada de Cloud
    String mensaje = String((char*)(payload)); // Asigna valor de cadena a mensaje
    if(mensaje == "vol_on") {v = 1;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica
    voltaje
    if(mensaje == "cor_on") {v = 0;c = 1;t = 0;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica
    corriente
    if(mensaje == "tem_on") {v = 0;c = 0;t = 1;r = 0;va = 0;vv = 0;vd = 0;} // habilita Gráfica
    temperatura
    if(mensaje == "rpm_on") {v = 0;c = 0;t = 0;r = 1;va = 0;vv = 0;vd = 0;} // habilita Gráfica
    rpm
    if(mensaje == "v_a_on" ) {v = 0;c = 0;t = 0;r = 0;va = 1;vv = 0;vd = 0;} // habilita Gráfica
    aceleración
    if(mensaje == "v_v_on" ) {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 1;vd = 0;} // habilita Gráfica
    velocidad
    if(mensaje == "v_d_on") {v = 0;c = 0;t = 0;r = 0;va = 0;vv = 0;vd = 1;} // habilita Gráfica
    desplazamiento
}
```

```

if(mensaje == "pause_on") {p = 1;} // stop on
if(mensaje == "led_on") {digitalWrite(Led,HIGH);} // enciende led
if(mensaje == "led_off") {digitalWrite(Led,LOW);} // apaga led
}

void loop() {
    websocket.loop(); // Función de actualización para webSockets
    // Lee Edge acondiciona y almacena datos en variables .....
    // SENSOR DE VOLTAJE.....
    short vol = ads.readADC_SingleEnded(0); // almacena valor de A0 en vol
    vol = (vol/48.5)-312; // mapeo de voltaje
    // SENSOR DE CORRIENTE.....
    short cor = ads.readADC_SingleEnded(2); // almacena valor de A2 en vol
    cor = (cor/0.33065); // mapeo de corriente
    // SENSOR DE TEMPERATURA.....
    short tem = ads.readADC_SingleEnded(3); // almacena valor de A3 en vol
    tem = (tem/99.29); // mapeo de temperatura
    // SENSOR DE RPM.....
    tiempo = millis(); // habilita temporizador
    if ((tiempo - tr) >= 2000){ // define temporizador a 2s
        tr = tiempo; // resetea temporizador
        rpm = i; // almacena valor en rpm
        i = 0; // resetea contador
    }

    // SENSOR DE VIBRACION.....
    acelerometro.getMotion6(&v_x, &v_y, &v_z, &gx, &gy, &gz); // lee y almacena valor
    de A3 en vol
    v_a = (v_x/1.63385)+270-50; // mapeo de aceleración, sensibilidad = 0.598755 mm/s2
    v_v = (v_a*86.75)/435; // conversión a velocidad
    v_d = (v_v*318.5)/435; // conversión a desplazamiento

    // Envía datos de Gateway a capa superior.....
    if(v == 1){object["sensor"] = vol; // almacena valor voltaje en objeto
        object["ref1"] = -339; // almacena valor max negativo en objeto
        object["ref2"] = 339;} // almacena valor max positivo en objeto

    if(c == 1){object["sensor"] = cor; // almacena valor corriente en objeto
        object["ref1"] = -14142; // almacena valor max negativo en objeto
        object["ref2"] = 14142;} // almacena valor max positivo en objeto

    if(t == 1){object["sensor"] = tem; // almacena valor temperatura en objeto
        object["ref1"] = 0; // almacena valor minimo en objeto
        object["ref2"] = 150;} // almacena valor maximo en objeto

    if(va == 1){object["sensor"] = v_a; // almacena valor aceleración en objeto
        object["ref1"] = -20000; // almacena valor max negativo en objeto
        object["ref2"] = 20000;} // almacena valor max positivo en objeto

```

```

if(vv == 1){object["sensor"] = v_v; // almacena valor velocidad en objeto
    object["ref1"] = -2000; // almacena valor max negativo en objeto
    object["ref2"] = 2000;} // almacena valor max positivo en objeto

if(vd == 1){object["sensor"] = v_d; // almacena valor desplazamiento en objeto
    object["ref1"] = -200; // almacena valor max negativo en objeto
    object["ref2"] = 200;} // almacena valor max positivo en objeto

if(r == 1){object["sensor"] = rpm*30; // almacena valor rpm en objeto
    object["ref1"] = 0; // almacena valor minimo en objeto
    object["ref2"] = 10000;} // almacena valor maximo en objeto

while(p == 1){if(digitalRead(Pin) == HIGH){p = 0;break;}} // ciclo infinito para stop

serializeJson(doc, jsonString); // convertir objeto JSON a cadena
Serial.println(jsonString); // imprime cadena JSON en monitor serie
websocket.broadcastTXT(jsonString); // Enviar cadena JSON a cloud
}

// Funcion que se ejecuta durante cada interrupcion.....
void interrupcion()
{
    i++; // Se incrementa en uno el contador
}

```

ANEXO B: PROGRAMA CLOUD

```

<script src="https://code.highcharts.com/highcharts.js"></script>
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-16">
</head>
<body>
    <div id="iot" class="container"></div>
    <center>
        <p>MEDICIÓN DEL SENSOR = <span id='vol'>-</span></p>
    </center>
    <center>
        <p>
            <button class="sensor" onclick= "vol_on()">VOLTAJE</button>
            <button class="sensor" onclick= "cor_on()">CORRIENTE</button>
            <button class="sensor" onclick= "tem_on()">TEMPERATURA</button>
            <button class="sensor" onclick= "rpm_on()">RPM</button>
        </p>
    </center><center>

```

```
<p>
<button class="sensor" onclick= "v_a_on()">ACELERACIÓN</button>
<button class="sensor" onclick= "v_v_on()">VELOCIDAD</button>
<button class="sensor" onclick= "v_d_on()">DESPLAZAMIENTO</button>
</p>
</center><center>
<p>
<button class="paus" onclick= "pause_on()">STOP</button>
</p>
</center><center>
<p>
<button class="led"  onclick= "led_on()">LED ON</button>
<button class="led"  onclick= "led_off()">OFF</button>
</p>
<center></center>
</body>
</html>

<script>
function vol_on() { Socket.send("vol_on");}
function cor_on() { Socket.send("cor_on");}
function tem_on() { Socket.send("tem_on");}
function rpm_on() { Socket.send("rpm_on");}
function v_a_on() { Socket.send("v_a_on");}
function v_v_on() { Socket.send("v_v_on");}
function v_d_on() { Socket.send("v_d_on");}

function pause_on()  { Socket.send("pause_on");}
function led_on()    { Socket.send("led_on");}
function led_off()   { Socket.send("led_off");}

const chartT = new Highcharts.Chart({
  chart:{ renderTo : 'iot' },
  title: { text: 'Gráfica de sensores para Motores de Inducción eléctricos'},
  series: [{
    name: 'Sensor',
    showInLegend: true,
    data: [],
  },{
    name: 'Ref1',
    showInLegend: true,
    data: [],
  },{
    name: 'Ref2',
    showInLegend: true,
    data: [],
  },],
  xAxis: {
```

```

    title: { text: 'Multiplcar por 12.82 para obtener el tiempo real' },
    type: 'datetime',
    dateTimeLabelFormats: { second: '%S' }
  },
  yAxis:
  [{title: { text: 'Aceleración ..mm/s2'}
  },{title: { text: 'Velocidad.....mm/s'}
  },{title: { text: 'Desplazamiento mm'}
  },{title: { text: 'RPM.....rpm' }
  },{title: { text: 'Temperatura.....°C' }
  },{title: { text: 'Corriente.....mA' }
  },{title: { text: 'Voltaje.....V'}},
  ]],
});

var Socket = new WebSocket('ws://' + window.location.hostname + ':81/');
Socket.onmessage = function(event){

    var x = (new Date()).getTime();
    var y = JSON.parse(event.data);
    document.getElementById('vol').innerHTML = y.sensor;

    if(chartT.series[0,1,2].data.length < 150) {
        chartT.series[0].addPoint([x, y.sensor], true, false, true);
        chartT.series[1].addPoint([x, y.ref1], true, false, true);
        chartT.series[2].addPoint([x, y.ref2], true, false, true);
    }else {
        chartT.series[0].addPoint([x, y.sensor], true, true, true);
        chartT.series[1].addPoint([x, y.ref1], true, true, true);
        chartT.series[2].addPoint([x, y.ref2], true, true,
true);
    }
};
</script>

```