

UNIVERSIDAD DE EL SALVADOR  
FACULTAD DE INGENIERÍA Y ARQUITECTURA  
ESCUELA DE INGENIERÍA ELÉCTRICA



**AMPLIACIÓN DEL MONITOREO DE VARIABLES  
ELÉCTRICAS EN LAS SUBESTACIONES DE LA  
UNIVERSIDAD DE EL SALVADOR**

PRESENTADO POR:  
**LUIS HUMBERTO PALACIOS GIRÓN**

PARA OPTAR AL TÍTULO DE:  
**INGENIERO ELECTRICISTA**

CIUDAD UNIVERSITARIA, NOVIEMBRE 2017

**UNIVERSIDAD DE EL SALVADOR**

**RECTOR :**

**MSC. ROGER ARMANDO ARIAS ALVARADO**

**SECRETARIO GENERAL :**

**MSC. CRISTÓBAL HERNÁN RÍOS BENÍTEZ**

**FACULTAD DE INGENIERÍA Y ARQUITECTURA**

**DECANO :**

**ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL**

**SECRETARIO :**

**ING. JULIO ALBERTO PORTILLO**

**ESCUELA DE INGENIERÍA ELÉCTRICA**

**DIRECTOR :**

**ING. ARMANDO MARTÍNEZ CALDERÓN**

UNIVERSIDAD DE EL SALVADOR  
FACULTAD DE INGENIERÍA Y ARQUITECTURA  
ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

**INGENIERO ELECTRICISTA**

Título :

**AMPLIACIÓN DEL MONITOREO DE VARIABLES  
ELÉCTRICAS EN LAS SUBESTACIONES DE LA  
UNIVERSIDAD DE EL SALVADOR**

Presentado por :

**LUIS HUMBERTO PALACIOS GIRÓN**

Trabajo de Graduación Aprobado por:

Docente Asesor :

**DR. CARLOS EUGENIO MARTÍNEZ CRUZ**

San Salvador, Noviembre 2017

Trabajo de Graduación Aprobado por:

Docente Asesor :

**DR. CARLOS EUGENIO MARTÍNEZ CRUZ**

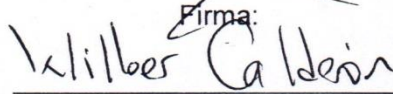
## ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, jueves 14 de septiembre de 2017, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 4:00 p.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón  
Director

Firma:   


2. MSc. José Wilber Calderón Urrutia  
Secretario

Firma: 

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

1- DR. CARLOS EUGENIO MARTINEZ CRUZ  
(Docente-Asesor)

Firma:   
  


2- MSC. JORGE ALBERTO ZETINO CHICAS

3- ING. WALTER LEOPOLDO ZELAYA CHICAS

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

AMPLIACIÓN DEL MONITOREO DE VARIABLES ELÉCTRICAS EN LAS  
SUBESTACIONES DE LA UNIVERSIDAD DE EL SALVADOR

A cargo del Bachiller:

- PALACIOS GIRÓN LUIS HUMBERTO

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final:

8.9

( Ocho punto Nueve )

## **AGRADECIMIENTOS**

Agradezco a mi familia que ha sido mi principal apoyo en esta etapa. A mi madre Gloria Alicia Girón Claros que me ha apoyado y aconsejado de forma incondicional durante toda mi vida, y que la finalización de mi carrera ha sido fruto principalmente del esfuerzo y sacrificio para brindarme las mejores condiciones académicas y económicas que le fueron posibles. A mi padre Santos Humberto Palacios Cruz que me transmitió a su manera particular, la experiencia y forma de ver la vida, para que yo pueda alcanzar un peldaño más. A mi hermano Gerson Wilfredo Palacios Girón que me ha apoyado a su manera de forma incondicional durante toda mi vida. A Roxana Patricia Aguilar Labor que recientemente ha comenzado a formar parte de mi vida y familia, y quien celebra mis éxitos y me motiva a superar los fracasos, y en quien confío plenamente.

También es mi deber reconocer a todas aquellas personas que de una u otra manera han puesto su granito de arena para que haya podido alcanzar mi meta. Especialmente quisiera agradecer a mis amigos Roberto Murillo y Rodrigo Garay, que junto a sus familias también me han apoyado cada que han podido. A mi primo Jonathan Alberto Soriano y su familia que siempre han estado pendientes de mí y me han colaborado como les ha sido posible. A algunos de mis compañeros de universidad con los que compartí éxitos y fracasos, Samuel González, Santiago Palma, Néstor Murillo y Raúl Alberto, a todos ellos no me queda más que desearles éxitos en sus vidas.

Para finalizar quiero agradecer a mi asesor de trabajo de graduación Dr. Carlos Eugenio Martínez quien ha sido un guía durante los últimos años de mi carrera. A quien reconozco el interés y esfuerzo por mejorar las condiciones académicas de nuestra universidad.

**Luis Humberto Palacios Girón**

# ÍNDICE GENERAL

CAPÍTULO I INTRODUCCIÓN .....	1
1.1 Interés de la investigación .....	2
1.2 Antecedentes .....	2
1.2.1 Arquitectura mesh o malla.....	3
1.2.2 Protocolo B.A.T.M.A.N Adv .....	6
1.2.3 Evolución de la interrogación de medidores .....	7
1.2.4 Monitoreo de los enlaces.....	12
1.3 Limitaciones actuales.....	13
1.4 Presentación de datos .....	13
1.5 Motivación para realizar el proyecto.....	15
1.6 Objetivos .....	15
1.6.1 Objetivo general.....	15
1.6.2 Objetivos específicos.....	15
1.7 Ejecución del proyecto.....	16
CAPITULO II: ADQUISICIÓN Y ALMACENAMIENTO DE VARIABLES ELÉCTRICAS.....	17
2.1 Selección de variables.....	17
2.1.1 Limitaciones de los medidores.....	17
2.1.2 Limitaciones de los router MP01 .....	19
2.1.3 Variables eléctricas seleccionadas .....	20
2.1.4 Consultas dinámicas.....	21
2.2 Almacenamiento in situ .....	22
2.3 Consultas al medidor .....	25
2.3.1 Software de consulta constante .....	30
2.3.2 Software de consulta de variables dinámicas.....	37
2.4 Envío de datos al servidor principal .....	41
2.4.1 Aplicación XML-RPC en los routers .....	42
2.4.1 Aplicación XML-RPC el servidor.....	45

CAPÍTULO III: APLICACIÓN WEB .....	48
3.1 Django .....	49
3.1.1 MVC y MTV.....	49
3.1.2 Modelos.....	51
3.1.3 Template o plantilla .....	54
3.1.4 Vistas .....	54
3.2 Angular .....	59
3.3 HighCharts.....	61
3.4 Google Maps .....	62
CAPITULO IV: RESULTADOS OBTENIDOS .....	64
4.1 Aplicación web .....	64
4.1.1 Inicio de la aplicación .....	64
4.1.2 Información de los nodos.....	65
4.1.3 Mediciones recientes .....	67
4.1.4 Mediciones históricas.....	71
4.1.5 Facturación.....	81
4.1.6 Ajustes .....	83
CAPITULO V: CONCLUSIONES Y LÍNEAS FUTURAS.....	88
5.1 Conclusiones. ....	88
5.2 Líneas futuras.....	89
ANEXO A-1. MAPA DE MEMORIA SHARK 100S .....	91
ANEXO A-2 MAPA DE MEMORIA SHARK 200S .....	98
ANEXO A-3 ADICIÓN DE NUEVO NODO AL SISTEMA.....	105
ANEXO A-4 SCRIPT DE MONITOREO DE LOS ENLACES INALÁMBRICOS.....	108
ANEXO B EJECUCIÓN DE LA APLICACIÓN EN MODO DE PRUEBAS.....	109
ANEXO C-1 REDISEÑO DEL PROTOTIPO DE BAJO COSTO BASADO EN RASPBERRY PI .....	111
ANEXO C-2 INSTALACIÓN DE RASPBIAN EN DISCO DURO .....	115
BIBLIOGRAFÍA .....	122



## ÍNDICE DE FIGURAS

FIGURA 1. ARQUITECTURA DE RED INALÁMBRICA INICIADA EN 2012 UNIVERSIDAD DE EL SALVADOR [1] .....	3
FIGURA 2. RED MESH MEDIANTE PROTOCOLO B.A.T.M.A.N. ....	6
FIGURA 3. ARQUITECTURA DE INTERROGACIÓN DE MEDIDORES DESDE EL SERVIDOR [1] .	7
FIGURA 4. TAREAS PROGRAMADAS EN EL SERVIDOR PARA PETICIÓN BROADCAST [2].....	8
FIGURA 5. INTERROGACIÓN SECUENCIAL [1].....	9
FIGURA 6. INTERROGACIÓN EN BROADCAST [2] .....	9
FIGURA 7. INTERROGACIÓN IN SITU [3] .....	10
FIGURA 8. PANEL DE CONTROL DE SPUD [11].....	12
FIGURA 9. MÉTODO BROADCAST (SUPERIOR) Y MÉTODO IN SITU (INFERIOR), MEDIDOR HUMANIDADES4.....	14
FIGURA 10. ESTRUCTURA DE LOS REGISTROS EN EL MAPA MODBUS.....	17
FIGURA 11. COMANDOS PARA CREAR LA BASE DE DATOS Y LA TABLA LECTURAS .....	24
FIGURA 12. CREACIÓN DE LA TABLA CONTROL EN LA BASE DE DATOS DBL.....	24
FIGURA 13. FLUJOGRAMA MAIN_CONTROL.LUA.....	27
FIGURA 14. DECLARACIÓN DE VARIABLES MAIN_CONTROL.LUA .....	28
FIGURA 15. EJECUCIÓN DE PROGRAMAS SHARK Y SHARK_QUERY .....	29
FIGURA 16. DECLARACIÓN DE VARIABLES EN SHARK.C .....	30
FIGURA 17. COMUNICACIÓN VIA MODBUS Y APERTURA DE LA BASE DE DATOS CON SQLITE3 .....	31
FIGURA 18. PROCESO DE CONSULTA DEL MEDIDOR .....	32
FIGURA 19. ALMACENAMIENTO EN LA BASE DE DATOS.....	34

FIGURA 20. FLUJOGRAMA DE EL PROGRAMA SHARK .....	35
FIGURA 21. SHARKMETER.H .....	36
FIGURA 22. INICIO DE SHARK_QUERY .....	37
FIGURA 23. ESTABLECIMIENTO DE COMUNICACIÓN CON EL MEDIDOR EN SHARK_QUERY .....	39
FIGURA 24. LECTURA Y ALMACENAMIENTO TEMPORAL SHARK_QUERY .....	40
FIGURA 25. ESQUEMA DE COMUNICACIÓN ROUTER-SERVIDOR [3] .....	41
FIGURA 26. PROCESO DE SINCRONIZACIÓN.....	41
FIGURA 27. XMLPI_L.LUA SOLICITUD DE ÚLTIMO ID_LECTURA.....	42
FIGURA 28. GESTIÓN DE LOS DATOS.....	43
FIGURA 29. ENVÍO DE DATOS AL SERVIDOR.....	44
FIGURA 30. ACTUALIZACIÓN DE INSTRUCCIONES.....	45
FIGURA 31. DECLARACIÓN DE VARIABLES XMLRPC_SERVER_L.LUA.....	46
FIGURA 32. XMLRPC_SERVER_L.CGI, OPERACIÓN .....	46
FIGURA 33. PATRÓN MTV.....	50
FIGURA 34. MODELO NODE DJANGO.....	52
FIGURA 35. MODELO MEASURE .....	53
FIGURA 36. MODEL PLIEGO .....	54
FIGURA 37. VIEW EN DJANGO, INCLUSIÓN DE ELEMENTOS.....	55
FIGURA 38. VIEW, CLASE INDEXVIEW.....	56
FIGURA 39. VIEW, CLASE NODEVIEWSET .....	56
FIGURA 40. VIEW, CLASE MEASURESVIEWSET .....	57
FIGURA 41. VIEW, PLIEGOVIEWSET .....	58

FIGURA 42. VIEW, RELACIÓN ENTRE URL Y VIEWS .....	58
FIGURA 43. ANGULARJS FUNCIONAMIENTO .....	60
FIGURA 44. EJEMPLO CON HIGHCHARTS, GRAFICA DE ENERGÍA .....	61
FIGURA 45. APLICACIÓN WEB, PAGINA PRINCIPAL.....	64
FIGURA 46. DESCRIPCIÓN DE LA PAGINA DE MEDICIONES.....	66
FIGURA 47. DETALLE DE CAMPOS EN LECTURAS RECIENTES.....	67
FIGURA 48. VISUALIZACIÓN DE FRECUENCIA Y CORRIENTE DE NEUTRO .....	68
FIGURA 49. DIAGRAMA FASORIAL.....	69
FIGURA 50. DIAGRAMA FASORIAL DE CORRIENTE.....	70
FIGURA 51. FACTOR DE POTENCIA.....	70
FIGURA 52. GRAFICA DE POTENCIA ACTIVA, REACTIVA Y APARENTE.....	71
FIGURA 53. VALORES DE POTENCIA DEMANDADA.....	72
FIGURA 54. VALORES MÁXIMOS Y MÍNIMOS.....	73
FIGURA 55. DATOS HISTÓRICOS DE ENERGÍA .....	74
FIGURA 56. DATOS HISTÓRICOS DE FACTOR DE POTENCIA.....	75
FIGURA 57. DATOS HISTÓRICOS DE FRECUENCIA .....	76
FIGURA 58. DATOS HISTÓRICOS DE TENSIÓN DE LÍNEA A NEUTRO.....	76
FIGURA 59. DATOS HISTÓRICOS DE TENSIÓN DE LÍNEA A LÍNEA.....	77
FIGURA 60. DATOS HISTÓRICOS DE CORRIENTE .....	78
FIGURA 61. DATOS HISTÓRICOS DE POTENCIA ACTIVA POR FASE.....	79
FIGURA 62. DATOS HISTÓRICOS DE POTENCIA REACTIVA POR FASE.....	80
FIGURA 63. DATOS HISTÓRICOS DE POTENCIA APARENTE POR FASE .....	80
FIGURA 64. FACTURACIÓN DEL PERIODO SELECCIONADO .....	81

FIGURA 65. PLIEGO TARIFARIO.....	82
FIGURA 66. DETALLE DEL CONSUMO DE ENERGÍA .....	83
FIGURA 67. INGRESANDO A LAS CONFIGURACIONES .....	84
FIGURA 68. PÁGINA ADMINISTRATIVA DE LA APLICACIÓN WEB.....	85
FIGURA 69. DATOS EN MEASURES .....	86
FIGURA 70. DATOS EN NODES.....	86
FIGURA 71. DATOS EN PLIEGO .....	87
FIGURA 72. REPRESENTACION DEL VALOR FLOTANTE.....	92
FIGURA 73. INICIALIZACIÓN DEL SERVIDOR DE PRUEBA CON DJANGO.....	110
FIGURA 74. VISTA FRONTAL DEL PROTOTIPO BASA EN RASPBERRY PI.....	111
FIGURA 75. VISTA LATERAL DEL PROTOTIPO .....	112
FIGURA 76. VISTA ISOMÉTRICA DEL GABINETE .....	113
FIGURA 77. SOPORTE PARA RASPBERRY PI .....	113
FIGURA 78. VISTAS FRONTAL Y LATERAL DERECHA .....	114
FIGURA 79. VISTA SUPERIOR .....	114
FIGURA 80. VISTA POSTERIOR .....	114
FIGURA 81. VISTA ISOMÉTRICA .....	114
FIGURA 82. UBICACIÓN DE LAS PARTICIONES.....	120

## ÍNDICE DE TABLAS

TABLA 1. ASIGNACIÓN DE NOMBRES E IP'S A LOS MEDIDORES. (*) MEDIDORES FUERA DE SERVICIO. (**) MEDIDORES QUE REQUIEREN MANTENIMIENTO.....	5
TABLA 2. SECCIONES DE MAPA DE REGISTRO MODBUS .....	18
TABLA 3. BLOQUES DE LA SECCIÓN DE DATOS DEL MEDIDOR .....	18
TABLA 4. VARIABLES SELECCIONADAS EN LA CONSULTA DE MEDIDORES, CONSULTA CONSTANTE .....	21
TABLA 5. TIPOS DE DATOS PARA EL PROGRAMA SHARK_QUERY .....	38
TABLA 6. CÓDIGO DE COLORES DE ENLACES WIFI .....	62
TABLA 7. TIPOS DE DATOS DEL MAPA MODBUS .....	92

# CAPÍTULO I INTRODUCCIÓN

La Universidad de El Salvador cuenta con una red de medidores de energía que tienen la capacidad de proveer información importante de variables eléctricas para representar el consumo y la calidad de energía. Sin embargo, hasta la fecha se ha estado almacenando un número reducido de variables. Esta limitación fue impuesta por cuestiones de tipo económico. Este trabajo de graduación incrementará el número de variables a almacenar. Por una parte, la mayoría de las nuevas variables a almacenar se podrán consultar de manera local en una aplicación web basada en la herramienta Django. Por otra parte, se seguirán subiendo al servicio web en la nube las mismas variables que en trabajos anteriores. El servicio web requerirá la incorporación de nuevas hojas de estilo para la visualización de las diferentes variables eléctricas.

El trabajo se divide en cuatro apartados principales. El primero consiste en desarrollo del software que interactúa con el medidor de energía eléctrica, este se encarga de las adquisiciones de variables que son almacenadas dentro del router y enviadas al servidor cuando se presenta la oportunidad. El segundo apartado describe el funcionamiento de la aplicación web que permite gestionar y visualizar, la información almacenada en el servidor. El tercer apartado consiste en un análisis de las condiciones de las subestaciones basadas en los análisis de los datos mostrados en la aplicación. El cuarto y último apartado incluye la modificación de la base de datos local, para que almacene las nuevas variables que serán enviadas al servidor y de esta manera puedan ser consultadas para su visualización y la modificación del prototipo de almacenamiento de bajo costo basado en Raspberry Pi.

## **1.1 Interés de la investigación**

Actualmente el estado del sistema está limitado por factores como la continuidad de los enlaces wifi. El servidor web gratuito tiene una cantidad limitada operaciones. La solicitud de mediciones actualmente está restringida a la lectura de 3 variables cada 5 minutos. Dentro de las variables que no se están utilizando se encuentran tensión línea-neutro, tensión línea-línea, corriente, factor de potencia, distorsión armónica, entre otros. La lectura de nuevas variables implica una adecuación de la actual arquitectura. Así también se requerirá de herramienta para poder visualizar esta nueva información.

## **1.2 Antecedentes**

La Universidad de El Salvador llevó a cabo un proyecto de medición del consumo de energía eléctrica [1][2][3]. Para lo cual se adquirieron medidores en las versiones 100s, 200 y 200s de la marca Shark que en efecto presentaban los valores de mediciones en un display sencillo en el aparato. Esta información se limitaba a los usuarios que pudieran observarlos físicamente. Este sistema de consulta representó una desventaja técnica. Durante los años 2013-2016 se llevaron a cabo trabajos de graduación en el área de ingeniería eléctrica que permitieran acceder a la información provista por los medidores [1][2][3]. Así es como se implementó el uso de routers inalámbricos que permitieran obtener las lecturas de estos medidores y transmitirlos a un servidor local. La arquitectura de la red se basó en la utilización de una topología tipo malla o mesh. Luego se añadió a este sistema una plataforma web que se enfocó en la representación del consumo y costos de la energía eléctrica. Este servicio fue diseñado bajo los estándares de la plataforma Google App Engine. Se utilizó como marco de trabajo webapp2 y Jinja2 como framework y motor de plantillas respectivamente. Este último como gestor de plantillas. Esta iniciativa seguía los lineamientos de bajo costo, por lo que la cantidad de datos utilizada para representarse en el servidor web debía estar siempre dentro de la cuota gratuita que ofrece el servicio de Google. Esta limitante también se podía palpar en el servidor local que se utilizaba para

almacenamiento. Solo constaba de una Raspberry PI que almacenaba la información en la memoria flash que utiliza y luego cada 50 minutos subía estos datos a la plataforma de App Engine de Google.

El sistema actual tiene mejoras que se enfocan en la integridad de los datos. Nuevas ideas basadas en el almacenamiento en los routers inalámbricos y el aprovechamiento de la oportunidad de envío de datos al servidor local han permitido recuperar la información que se perdía con anterioridad. Otro punto es la expansión que ha tenido el almacenamiento del servidor local de bajo costo con la adición de 2 discos duros, que evitan que se utilice la memoria flash de la Raspberry PI. Así como la adición de una segunda Raspberry PI que opera en modo respaldo de la principal. Ver Anexo C-2

### 1.2.1 Arquitectura mesh o malla

Uno de los primeros trabajos de graduación referente al tema consistió en la implementación de la arquitectura de red tipo malla o mesh con routers inalámbricos que prevalece actualmente [1].

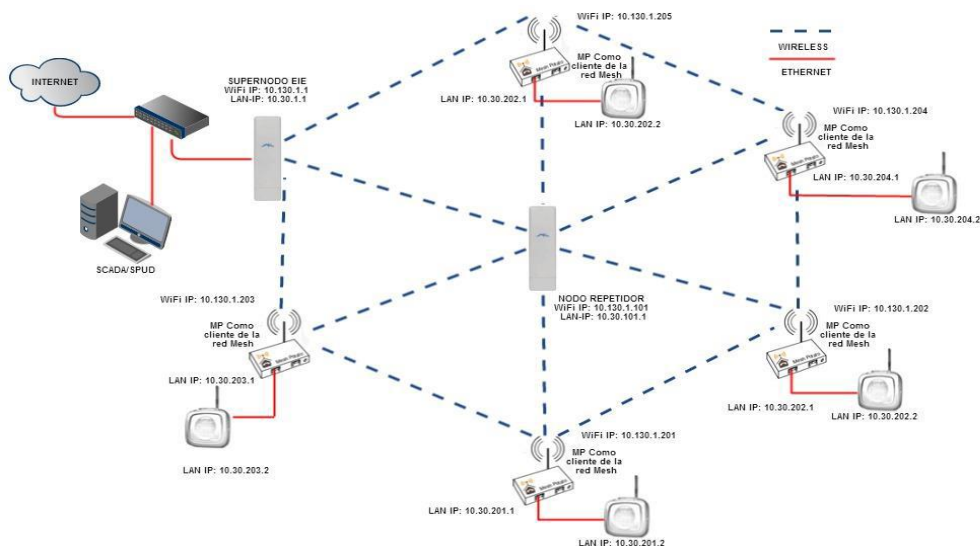


FIGURA 1. ARQUITECTURA DE RED INALÁMBRICA INICIADA EN 2012 UNIVERSIDAD DE EL SALVADOR [1]



En la FIGURA 1, se muestra la arquitectura de la red malla o mesh implementada en la red de medidores de la Universidad de El Salvador [1].

La red de medición consta de 3 elementos principales. El primero son los nodos que están constituidos por routers inalámbricos, en este caso routers MP01 de Village Telco. Estos son los que forman la red mesh que esencialmente consiste en comunicar un punto X con un punto Y, utilizando la mejor ruta proporcionada por los enlaces inalámbricos. Dentro de esta red existen nodos principales o súper nodos que monitorean el tráfico en la red mesh. El súper nodo para esta red se encuentra ubicado en techo de la escuela de ingeniería eléctrica. El segundo elemento consiste en los dispositivos conectados a un nodo de la red mesh, entre ellos tenemos los medidores Shark 100S, Shark 200 y Shark 200S que están vinculados al nodo más cercano a través de sus puertos Ethernet. El tercer elemento es el servidor principal, que es el prototipo de bajo costo basado en Raspberry Pi. Cabe destacar que no todos los nodos están conectados a un medidor de energía, sino que sirven como repetidores que mejoran la cobertura y aumentan la cantidad de rutas disponibles para envíos de datos.

La actualización del firmware más reciente es SECN 1.1 que está basado en OpenWRT una distribución Linux para sistemas embebidos con funcionalidades PBX basadas en asterisk, y provisto del protocolo de enrutamiento B.A.T.M.A.N Adv. Los router MP01 ofrecen capacidades de telefonía IP y AccesPoint.

La red cuenta con 19 medidores Shark 100S, 10 medidores Shark 200S y 2 medidor Shark 200, distribuidos de acuerdo a la TABLA 1. Desde el año 2015 cuatro medidores se encuentran fuera de servicio y 2 más requieren mantenimiento [4].

#	MEDIDOR	IP	IP MP01	TIPO (SHARK)	TIPO DE SUBS-TACI3N
1	Agronomía	10.30.217.2	10.130.3.217	100S	3 Fases
2	AgronomiaDecanato	10.30.218.2	10.130.3.218	100S	1 Fase
3	AgronomiaGalera	10.30.219.2	10.130.3.219	100S	1 Fase
4	AgronomiaQuimica**	10.30.220.2	10.130.3.220	200S	3 Fases
5	Artes *	10.20.230.2	10.130.3.230	200S	3 Fases
6	AuditoriumMarmol	10.30.215.2	10.130.3.215	100S	1 Fase
7	Cafetines	10.30.212.2	10.130.3.212	100S	1 Fase
8	ComedorUES	10.30.211.2	10.130.3.211	100S	1 Fase
9	Derecho	10.30.201.2	10.130.3.201	200S	3 Fases
10	Economia1	10.30.202.2	10.130.3.202	100S	1 Fase
11	Economia2	10.30.203.2	10.130.3.203	100S	1 Fase
12	Economia3	10.30.204.2	10.130.3.204	100S	1 Fase
13	Economia4	10.30.205.2	10.130.3.205	100S	1 Fase
14	Economia5 *	10.30.206.2	10.130.3.206	200S	3 Fases
15	Economia6	10.30.207.2	10.130.3.207	100S	1 Fase
16	Humanidades1*	10.30.208.2	10.130.3.208	100S	3 Fases
17	Humanidades2	10.30.209.2	10.130.3.209	100S	1 Fase
18	Humanidades3	10.30.210.2	10.130.3.210	100S	1 Fase
19	Humanidades4	10.30.231.2	10.130.3.231	200	1 Fase
20	MecanicaComplejo	10.30.216.2	10.130.3.216	100S	1 Fase
21	Medicina	10.30.227.2	10.130.3.227	200S	3 Fases
22	Odontologia1 **	10.30.224.2	10.130.3.224	200S	3 Fases
23	Odontologia2	10.30.225.2	10.130.3.225	200S	3 Fases
24	Odontologia3	10.30.226.2	10.130.3.226	200S	3 Fases
25	OdontologiaImprenta	10.30.223.2	10.130.3.223	100S	1 Fase
26	Periodismo	10.30.213.2	10.130.3.213	100S	3 Fases
27	PrimarioFIA	10.30.214.2	10.130.3.214	200	3 Fases
28	Psicologia	10.30.228.2	10.130.3.228	100S	3 Fases
29	Rectoria*	10.30.229.2	10.130.3.229	200S	3 Fases
30	Quimica	10.30.221.2	10.130.3.221	200S	3 Fases
31	Quimicalmprenta	10.30.222.2	10.130.3.222	100S	1 Fase

TABLA 1. ASIGNACI3N DE NOMBRES E IP'S A LOS MEDIDORES.

(\*) MEDIDORES FUERA DE SERVICIO. (\*\*) MEDIDORES QUE REQUIEREN MANTENIMIENTO

## 1.2.2 Protocolo B.A.T.M.A.N Adv

B.A.T.M.A.N. Advanced (a menudo referido como batman-adv) es una implementación del protocolo de ruteo B.A.T.M.A.N. en forma de un módulo para el kernel de Linux que opera en la capa 2 del modelo OSI [6].

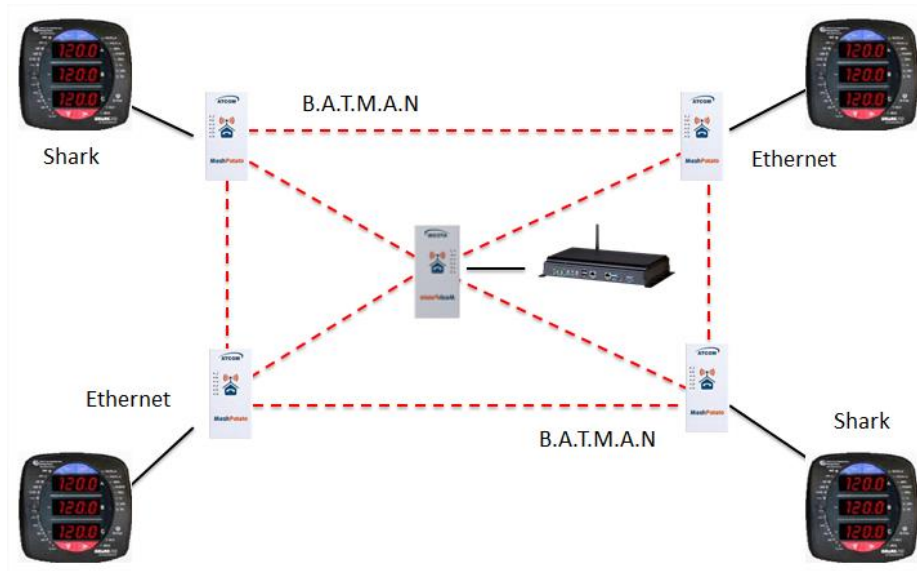


FIGURA 2. RED MESH MEDIANTE PROTOCOLO B.A.T.M.A.N. [3]

Los protocolos de enrutamiento inalámbrico (por ejemplo, el demonio de batman) operan en la capa 3 del modelo OSI [10], lo que significa que intercambian información de enrutamiento enviando paquetes UDP y poniendo en práctica su decisión de enrutamiento mediante la manipulación de la tabla de enrutamiento del kernel. Batman-adv opera completamente la capa 2 - no sólo la información de enrutamiento se transporta utilizando tramas de Ethernet sin procesar sino también el tráfico de datos es manejado por batman-adv. Encapsula y reenvía todo el tráfico hasta que llega al destino, emulando así un conmutador de red virtual de todos los nodos participantes. Por lo tanto, todos los nodos parecen ser enlace local y no son conscientes de la topología de la red, así como no afectados por los cambios de red.

Este diseño lleva algunas características interesantes:

- Capa de red básica: B.A.T.M.A.N.Adv Permite el uso de diferentes protocolos de capa superior como batman-adv: IPv4, IPv6, DHCP, IPX.
- Los nodos pueden participar en una malla sin tener una IP
- Fácil integración de clientes que no son de malla (móviles)
- Roaming de clientes que no son de malla
- Optimizando el flujo de datos a través de la malla (por ejemplo, alternación de interfaz, multidifusión, corrección de errores hacia adelante, etc.)
- Ejecutando protocolos que dependen de broadcast / multicast sobre los clientes de malla y no de malla (redes Windows, mDNS, streaming, etc.)

### 1.2.3 Evolución de la interrogación de medidores

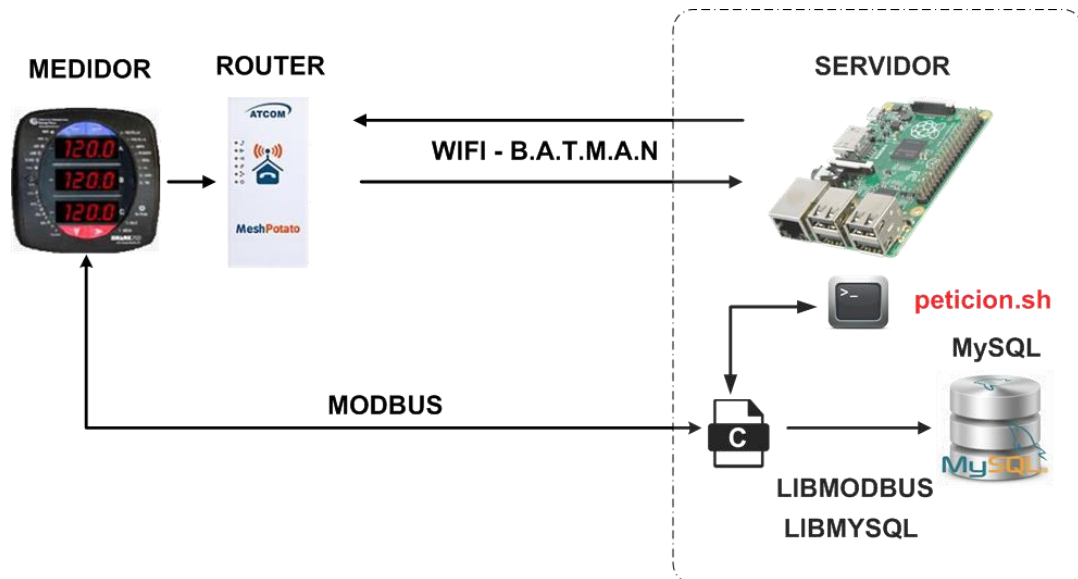


FIGURA 3. ARQUITECTURA DE INTERROGACIÓN DE MEDIDORES DESDE EL SERVIDOR [3]

En la FIGURA 3, se muestra la arquitectura de consulta previa a la interrogación in situ desarrollada en un trabajo de graduación en 2016. La metodología era la siguiente, el programa principal `peticion.sh` es el encargado de consultar cada medidor a través de la red mesh, esta aplicación se ejecutaba cada 15 minutos. Una deficiencia con este sistema era

el aislamiento de los nodos. Los nodos están interconectados de forma inalámbrica, lo que proporciona una cobertura amplia como punto fuerte. Sin embargo está limitada a la infraestructura y vegetación alrededor del nodo, los edificios y árboles son barreras que no siempre son sorteadas. También debe tomarse en cuenta la saturación de las bandas de frecuencias utilizadas por los router, así como las condiciones climáticas que atenúan la señal. Esto provoca que los enlaces tengan una baja transferencia e incluso se rompan durante periodos de tiempo prolongados, en el peor de los casos.

```
pi@raspberrypi ~ $ sudo crontab -l | tail -2
*/15 * * * * cd /home/pi/tesisduque/medidores && ./peticion.sh >> /home/pi/tesisduque/logs/medidores.log 2>&1
*/50 * * * * cd /home/pi/tesisduque && ./updater.sh >> /home/pi/tesisduque/logs/actualizacion.log 2>&1
pi@raspberrypi ~ $
```

FIGURA 4. TAREAS PROGRAMADAS EN EL SERVIDOR PARA PETICIÓN BROADCAST [3]

La arquitectura de interrogación como se muestra en la FIGURA 3 es controlada totalmente por el servidor que es una raspberry pi, esta tiene configurada la ejecución del programa `peticion.sh` que trata de acceder a cada medidor a través de la red mesh, esto es broadcast. Esta operación se realiza cada 15 minutos y por cada medidor realiza dos intentos de establecer la conexión, si las condiciones de la red no lo permiten al estar el medidor y su nodo aislados de la red temporalmente esa medición no era realizada. Anterior al método broadcast la consulta se realizaba uno a la vez en forma secuencial.

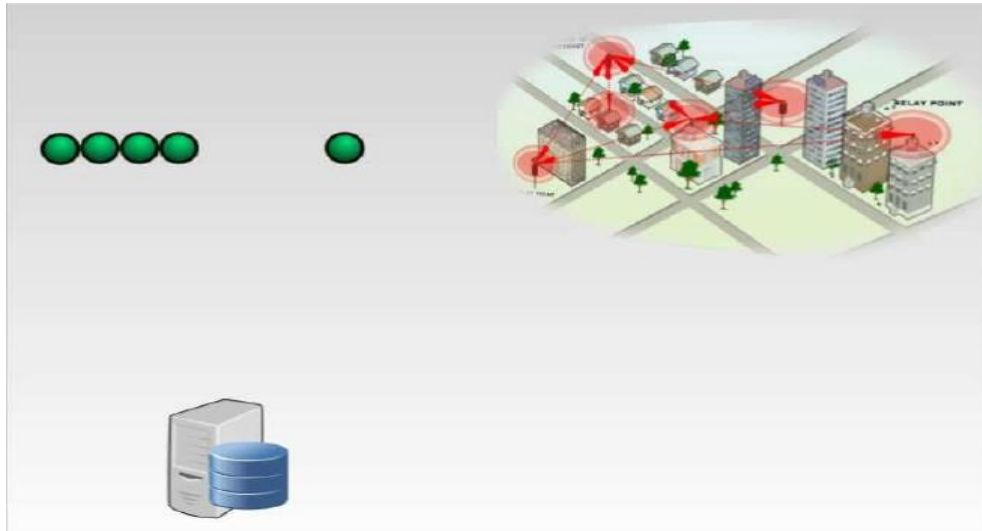


FIGURA 5. INTERROGACIÓN SECUENCIAL [2]

La técnica de broadcast tampoco logró mantener la integridad que se requería, persistía el problema del aislamiento, solo en ciertos momentos se podían realizar las consultas, los resultados presentaron una leve mejoría, aunque los datos todavía carecían de integridad, pero fue punto de partida para el siguiente paso en la evolución del sistema.

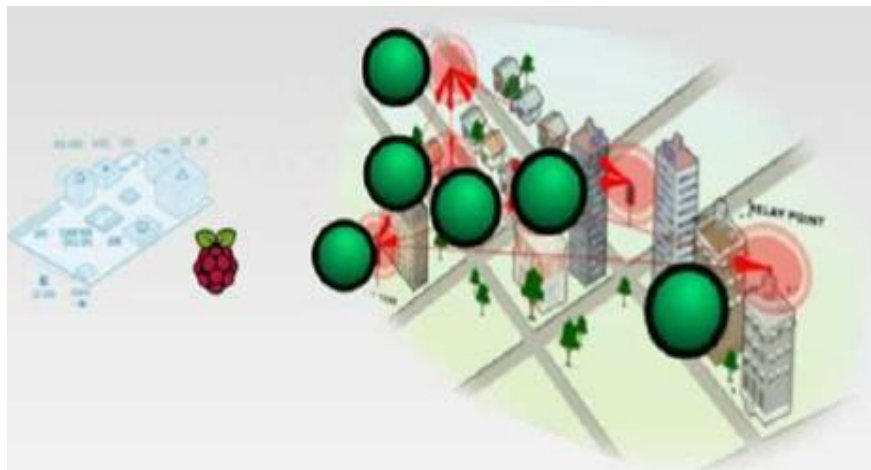


FIGURA 6. INTERROGACIÓN EN BROADCAST [2]

Un enfoque diferente de enfrentar este problema permitió mejorar sustancialmente la integridad de los datos. En 2016 se realizó un cambio en la arquitectura de interrogación del formato broadcast a la interrogación in situ. Esta técnica consistió en trasladar el pro-

caso de consulta del medidor que originalmente era tarea del servidor, hacia el nodo que tiene un enlace físico con el medidor de energía eléctrica, esto permitiría poder interrogar de forma constante a los medidores al utilizar solamente el cable de red como medio de comunicación. Esta solución evadió las intermitencias de la señal inalámbrica pero planteó un nuevo reto, que era la adecuación del nodo para almacenar la información y posteriormente transmitirla al servidor principal ocupando una idea que se denominó “oportunidad”. La oportunidad consistía en determinar el momento en que el nodo puede establecer un enlace con el servidor. Esto permitió mantener la integridad de los datos y actualizar la base de datos en el servidor, la mejora en la integridad de datos fue palpable, aunque el efecto de aislamiento persistiese, la integridad de los datos se mantuvo.

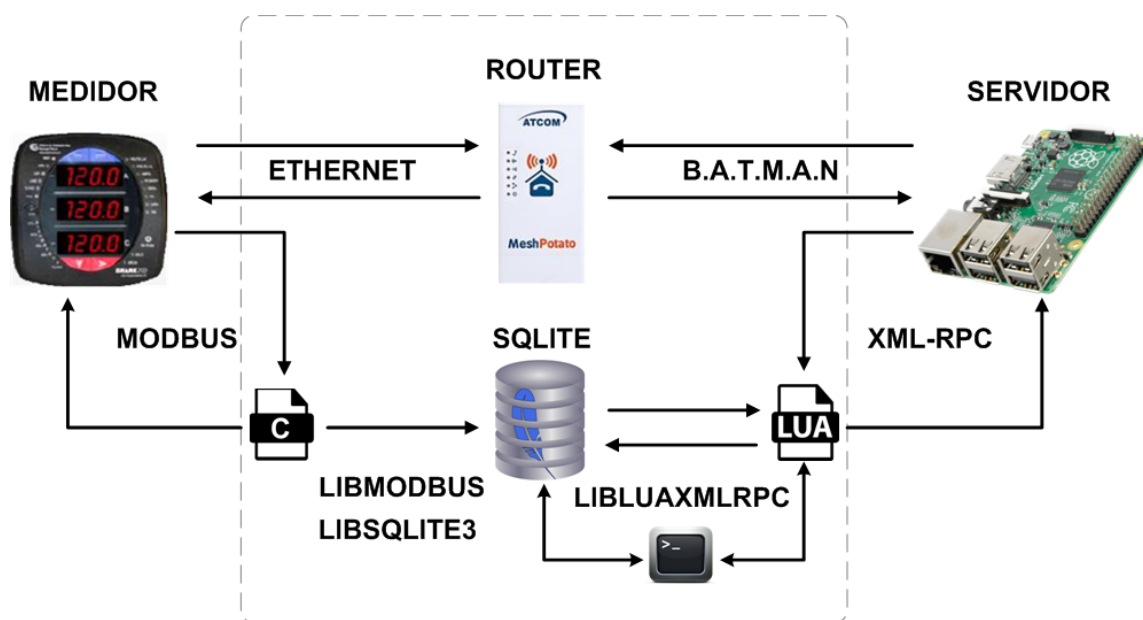


FIGURA 7. INTERROGACIÓN IN SITU [3]

La arquitectura de interrogación in situ se presenta en la FIGURA 7. Se observa como el nodo toma más relevancia en el proceso de consulta, al encargarse de la extracción de mediciones directamente desde el medidor, a través de una conexión con cable Ethernet. El protocolo de comunicación MODBUS TCP/IP es utilizado por la aplicación de consulta desarrollada en lenguaje C, a su vez la aplicación de consulta almacena las mediciones en una base de datos SQLite, esta forma de almacenamiento es ideal porque funciona en

sistemas embebidos. El proceso de comunicación entre el nodo y el servidor lo ejecuta el script escrito en lenguaje Lua, que se ejecuta en el nodo como una tarea programada. El procedimiento consiste en consultar al servidor el identificador de la última lectura almacenada en este. Utilizando el identificador de la lectura como referencia se consultan los registros posteriores al identificador en la base de datos SQLITE. Si existen registros de estos se envían al servidor para ser almacenados. Al realizar el envío de registros al servidor de manera exitosa se procede a eliminar los registros anteriores al identificador devuelto por el servidor para optimizar el uso de memoria en el router. Si el procedimiento de envío de registros al servidor fallara, no se elimina ningún registro de la base de datos SQLITE en el nodo. Este proceso de comunicación nodo-servidor está implementado con XML-RPC.

El protocolo XML-RPC, se traduce como llamada a procedimiento remoto (Remote Proceed Call) que utiliza como lenguaje de codificación XML. XML-RPC convierte los datos a formato XML (eXtensible Markup Language o Lenguaje de marcado extensible). XML utiliza etiquetas para determinar el tipo y jerarquía de la información similar a HTML, el único tipo de dato transmitido es texto. El protocolo consta de dos elementos que son el codificador y decodificador que pueden estar escritos en lenguajes de programación diferentes. La rutina de envío está escrita en Lua y lo ejecuta el nodo. Por otra parte tenemos en el servidor, que tiene reservado un puerto para la recepción de datos con este protocolo, el decodificador está escrito en Perl, este script no solo decodifica la información en XML, sino que además carga la información a la base de datos MySQL. También realiza consultas a la base de datos a solicitud del nodo, con el fin de mantener apropiadamente sincronizada la información entre el servidor y el nodo [3].



## 1.2.4 Monitoreo de los enlaces

En el año 2013 se realizaron mejoras a la red de medidores, se actualizó a la versión del firmware SECN 1.1 en cada Router MP01, el cual por defecto tiene configurado el protocolo B.A.T.M.A.N. Adv.

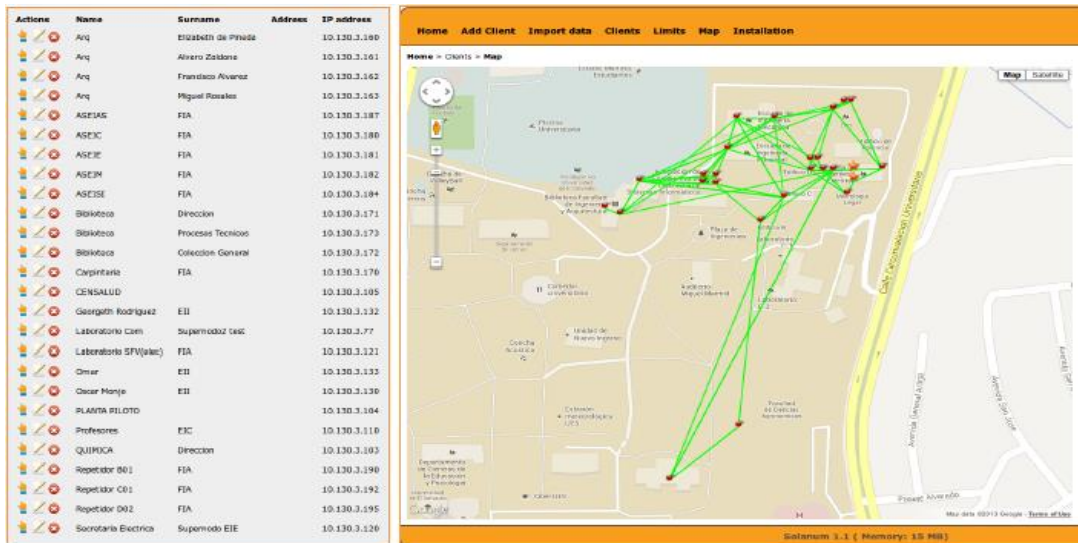


FIGURA 8. PANEL DE CONTROL DE SPUD [11]

El monitoreo de los enlaces inalámbricos tiene prioridad en las redes mesh, por lo que en principio se utilizó software proporcionado por Village Telco para el monitoreo, la visualización de los enlaces y la intensidad de la señal mostraron como se generaban las mallas, lo que hizo evidente la necesidad del mantenimiento periódico de los nodos, así como la adición de nodos repetidores para mejorar la comunicación entre las mallas. El software utilizado fue S.P.U.D. abreviación de “Simple Unified Dashboard for mesh networks” o panel unificado simple para redes en malla [11].

SPUD se encuentra discontinuado. Fue diseñado para ser lo más simple posible de usar, y para que los equipos, que han instalado gran cantidad de nodos de la malla, para visualizar sus redes rápidamente.

A principios 2016 el software fue descontinuado y se perdió la capacidad de monitorear los enlaces, esto limitó el mejoramiento de los enlaces debido a las facilidades de ubicación geográfica que tenía SPUD.

### **1.3 Limitaciones actuales**

Actualmente el estado del sistema está limitado por factores como la continuidad de los enlaces wifi. El servidor web gratuito tiene una cantidad limitada operaciones. La solicitud de mediciones actualmente está restringida a la lectura de 3 variables cada 5 minutos. Dentro de las variables que no se están utilizando se encuentran tensión línea-neutro, tensión línea-línea, corriente, factor de potencia, distorsión armónica, entre otros. La capacidad de almacenamiento de los router MP01, está limitada a la memoria física disponible que son 2 Mb y no es expandible. La lectura de nuevas variables implica una adecuación de la actual arquitectura. Así también se requerirá de herramienta para poder visualizar esta nueva información.

### **1.4 Presentación de datos**

El proyecto ha tenido mejoras considerables en integridad de datos, pero ha estado limitada a tres variables, que se enfocaron en el consumo de energía y la demanda de potencia, esto para poder realizar cálculos de costos en que incurre la Universidad de El Salvador.

La primera aplicación web se basó en AppEngine de Google, aprovechando el hosting gratuito y las herramientas proporcionadas por esta empresa, la desventaja implícita es una cantidad limitada de transacciones que ser excedidas se incurre en pago por el servicio, por lo que consultas cada 15 minutos de 3 variables se ajustaban a esta limitante.

Un ejemplo claro de cómo la nueva arquitectura de consulta ha mejorado la integridad de los datos se presenta en la siguiente comparación. La lectura se toma del medidor de Humanidades 4, en el periodo del 17 al 22 de julio del año 2017. Se observa claramente que el método broadcast muestra intervalos considerables de tiempo sin lectura en la FIGURA 9 (superior). La mejora se presenta en la FIGURA 9 (inferior), mismo periodo y el trazo es más detallado, las variables son las mismas pero las lecturas fueron constantes, y la integridad de datos es mejor.

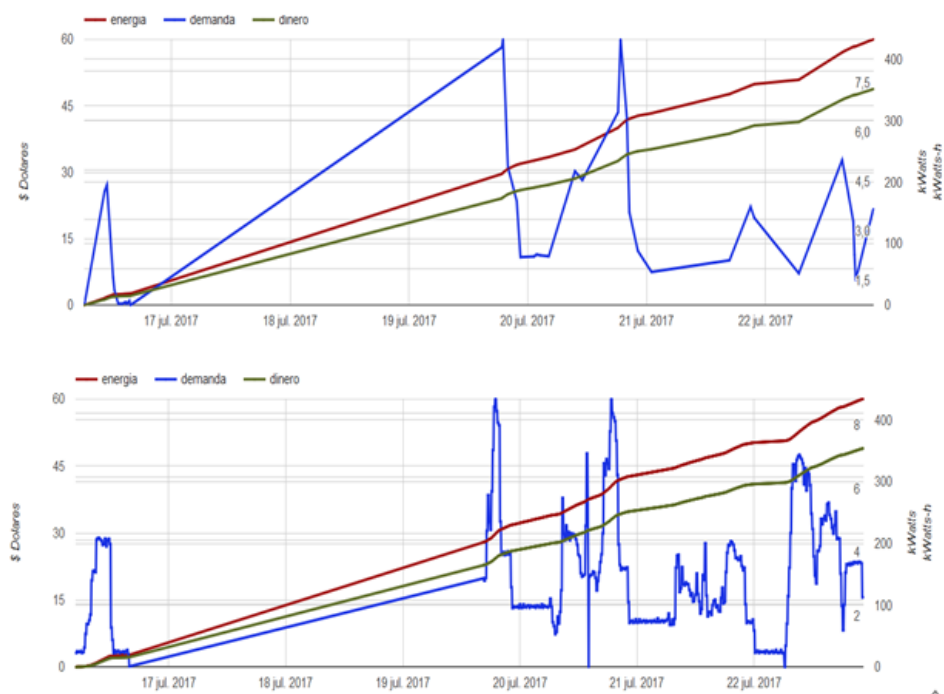


FIGURA 9. MÉTODO BROADCAST (SUPERIOR) Y MÉTODO IN SITU (INFERIOR), MEDIDOR HUMANIDADES4 [3]

En la FIGURA 9 se presentan valores de kW que corresponde al trazo azul, y kW-h en trazo café, esos valores corresponden a datos extraídos del medidor, mientras que el trazo verde representa el costo en dólares, este cálculo se realiza en el servidor con información de pliego tarifario. Se ha mencionado que existe una tercera variable, pero esta no ha sido utilizada en las gráficas y son los kilowatts positivos.

## **1.5 Motivación para realizar el proyecto**

El monitoreo de los parámetros eléctricos y su adecuada interpretación puede traducirse en ahorro y confiabilidad. Por este motivo este proyecto de graduación tiene como lineamiento principal utilizar en su totalidad las capacidades de los medidores de energía eléctrica. La robustez del sistema permitiría la lectura de nuevas variables eléctricas.

La ampliación de la cantidad de variables facilitaría a las autoridades correspondientes la ejecución de medidas orientadas al ahorro energético y la confiabilidad del sistema. Algunas medidas pueden ser balancear las cargas, corrección del factor de potencia, valoración del aumento o disminución de la capacidad instalada cada subestación. La ampliación de las variables leídas implica modificaciones en todo el sistema actual. Por lo que se debe adecuar la base de datos del servidor local y modificar la aplicación in situ que se ejecuta en los routers. Adicionalmente la lectura nuevas variables conlleva crear una herramienta de visualización adecuada a las nuevas capacidades del sistema.

## **1.6 Objetivos**

### **1.6.1 Objetivo general**

- Utilizar en toda su capacidad la consulta de variables eléctricas proporcionada por los medidores en la red de monitoreo inalámbrica

### **1.6.2 Objetivos específicos**

- Implementar una plataforma web local que presente los datos medidos.
- Desarrollar la herramienta que permita gestionar los medidores a toda su capacidad
- Modificar el sistema de bases de datos actual
- Facilitar la actualización del pliego tarifario mediante la interfaz web
- Crear una aplicación que permita visualizar la calidad de enlaces en la red wifi

## 1.7 Ejecución del proyecto

El trabajo se desarrolló en etapas para hacerlo más manejable, la primera etapa consistió en determinar que variables se extraerían de los medidores y como serían enviadas al servidor principal basado en Raspberry Pi. En cuanto a las variables se escogieron aquellas que eran comunes en los medidores a pesar de no ser de un modelo idéntico, una característica de estas variables es que de ellas el medidor calcula otros parámetros. Otro punto es el desarrollo del software para extraer y almacenar estas variables en el router, se tomó en consideración que este software debería tener la capacidad de acceder a otras variables si fuese necesario, y por último la transmisión de datos debería utilizar el protocolo XML-RPC como se desarrolló en un trabajo de graduación previo para garantizar la integridad de los datos.

La segunda etapa del proyecto involucra el desarrollo de la aplicación web, esta debe mostrar la información obtenida de los medidores de una forma clara y comprensible, además debe permitir la interacción con el sistema de una forma sencilla, la inclusión del sistema de monitoreo de enlaces inalámbricos también debe estar incluida como una herramienta al sistema. La utilización del framework Django escrita en Python es el punto de partida para implementar un sistema flexible y que pueda ser escalable.

La última etapa consiste en dar un diagnóstico de las condiciones en que operan las subestaciones de la Universidad de El Salvador, limitándose solamente a las que cuentan con los medidores de energía que cubre este trabajo. También se mostrara las condiciones de operación de la infraestructura mesh inalámbrica, para comprender de una forma ilustrativa la parte más básica del sistema de monitoreo.

# CAPITULO II: ADQUISICIÓN Y ALMACENAMIENTO DE VARIABLES ELÉCTRICAS

## 2.1 Selección de variables

El primer paso en el desarrollo del proyecto fue la selección de variables eléctricas a monitorear. Los medidores pueden realizar una gran cantidad de mediciones, desde mediciones directas de tensión y corriente, hasta valores históricos de máximos, mínimos y otros. Idealmente se deberían utilizar todas las capacidades de estos medidores, pero se encontraron limitaciones que se describirán a continuación que delimitaron la selección de variables a unas muy específicas.

### 2.1.1 Limitaciones de los medidores

Los medidores de energía eléctrica instalados en el campus central de la Universidad de El Salvador, se encuentran en tres versiones, el modelo Shark 100S, Shark 200 y el modelo Shark 200S, las capacidades por defecto de los modelos 200 y 200S son superiores. Los detalles de estos medidores se adjuntan en el ANEXO A-1.

Register	0								1																							
Byte	0				1				0				1																			
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Meaning	s	e	e	e	e	e	e	e	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
	sign		exponent				mantissa																									

FIGURA 10. ESTRUCTURA DE LOS REGISTROS EN EL MAPA MODBUS

Estos medidores cuentan con un mapa de memoria Modbus en el que se almacena información del medidor y de lecturas en registros. Cada registro es de 16 bits como se muestra en la FIGURA 10. Algunas valores se representan con dos o más registros. La interacción disponible con los medidores es por el puerto de red Ethernet, sobre el cual se im-

plementa el protocolo Modbus TCP/IP que se detalla en el ANEXO B. La primera división de la memoria se clasifica en las secciones que se detallan a continuación.

Nombre de la sección	Contenido	SHARK 100S	SHARK 200/200S
<b>Sección de Datos Fijos</b>	Detalles de información fija del medidor	1-47	1-47
<b>Sección datos del Medidor</b>	Detalles de las lecturas del medidor,	1000-5003	1000-12031
<b>Sección Comandos</b>	Detalles del Medidor, Bloque de Restablecimiento, Bloque de programación, Otro bloque de comandos y Cifrado en bloque.	20000-26011	20000-26011
<b>Sección de Ajustes programables</b>	Todos los detalles de los ajustes se pueden programar para configurar su medidor.	30000-30067	30000-33575
<b>Sección Lecturas Secundaria</b>	Detalles del medidor, Lecturas secundarias	40001-40100	40001-40100
<b>Sección Recuperación de Registros</b>	Detalles de recuperación de registros.	-	49997-51095

TABLA 2. SECCIONES DE MAPA DE REGISTRO MODBUS

En este trabajo nos centramos principalmente en la sección de datos del medidor. En esta sección se encuentran las lecturas tomadas por el medidor y se subdividen en bloques. Esto se muestra en la TABLA 3.

Nombre del bloque	Direcciones de memoria SHARK 100S	Direcciones de memoria SHARK 200	Direcciones de memoria SHARK 200S
<b>Bloque lecturas Primaria</b>	1000- 1029	1000-1053	1000-1053
<b>Bloque de Energía</b>	1100-1117	1500-1571	1500-1571
<b>Bloque de Demanda</b>	2000-2019	2000-2063	2000-2063
<b>Bloque Angulo de Fase</b>	4100-4105	4100-4105	4100-4105
<b>Bloque THD</b>	4000-4041	6000-6875	
<b>Mínimos y Máximos en Regular</b>	3000-3133	7976-9552	7976-9552

TABLA 3. BLOQUES DE LA SECCIÓN DE DATOS DEL MEDIDOR

Los medidores tienen diferentes capacidades según el modelo y las direcciones de memoria para las variables son diferentes, además de esta limitación existe otra que el fabrican-

te denomina V-Switch que actúan como limitadores, que vienen con el firmware original, es decir que el medidor tiene capacidad para asignar todas las variables en el registro de mapa de memoria correspondiente, pero el acceso a ellas está limitado por el V-Switch o llave. La versión más básica del V-Switch, nos permite el acceso al bloque de lecturas primarias, pero la característica más avanzada como por ejemplo la lectura del THD no se puede consultar. Poseer un V-Switch 4, permitiría que el acceso a los registros sea completo, y esto implicaría un pago adicional al fabricante. En los medidores propiedad de la Universidad El Salvador, existen dos que incluyen V-Switch 4 y en los cuales se pueden realizar una consulta completa de los registros. Por lo anterior, se ha ideado un método que permite extraer datos relevantes de estos medidores sin comprometer la memoria del router con datos innecesarios. Esto se expondrá adelante en la sección 2.1.4 consultas dinámicas.

### **2.1.2 Limitaciones de los router MP01**

Los router MP01 no están diseñados para almacenamiento de datos. La memoria disponible para almacenamiento luego de su configuración es de 2 MB. A diferencia del medidor que tiene una capacidad similar y que sobrescriben los registros para actualizarlos, los router deberán almacenar datos de forma constante si no se pueden sincronizar con el servidor principal. La cantidad de variables a seleccionar deberá ser la mínima para su representación y se debe evitar extraer variables que sirven como referencias históricas. Así se realizaría un uso eficiente del espacio de memoria del router con variables que no cambian, o que cambian al transcurrir semanas o meses como lo pueden ser valores máximos y mínimos y valores promedio. Otra consideración para optimizar el uso de memoria es priorizar los campos que son comunes en los diferentes modelos de medidores shark, esta limitación la impone que versión de V-Switch es más común.



### 2.1.3 Variables eléctricas seleccionadas

Las variables seleccionadas principalmente corresponden al bloque de lecturas primarias, debido a que estas permiten concluir valores como máximos y mínimos. Además, se tomaron en cuenta valores de energía que sirven para determinar costos de consumo. Otro factor importante es la consideración del factor de potencia. A continuación se detalla el listado de variables leídas de forma constante en este trabajo.

N°	Nombre del Campo	Shark 100S	Shark 200, 200S	Descripción
1	Volts_A_N	X	X	Tensión entre la fase A y neutro
2	Volts_B_N	X	X	Tensión entre la fase B y neutro
3	Volts_C_N	X	X	Tensión entre la fase C y neutro
4	Volts_A_B	X	X	Tensión entre la fase A y la fase B
5	Volts_B_C	X	X	Tensión entre la fase B y la fase C
6	Volts_C_A	X	X	Tensión entre la fase C y la fase A
7	Amps_A	X	X	Corriente en la fase A
8	Amps_B	X	X	Corriente en la fase B
9	Amps_C	X	X	Corriente en la fase C
10	Watts_3_Ph_total	X	X	Watts trifásicos totales
11	VARs_3_Ph_total	X	X	Voltamperios reactivos trifásicos totales
12	VAs_3_Ph_total	X	X	Voltamperios trifásicos totales
13	Power_Factor_3_Ph_total	X	X	Factor de potencia trifásico
14	Frequency	X	X	Frecuencia
15	Neutral_Current	X	X	Corriente de neutro
16	Watts_Phase_A		X	Watt en la fase A
17	Watts_Phase_B		X	Watts en la fase B
18	Watts_Phase_C		X	Watts en la fase C
19	VARs_Phase_A		X	Voltamperios reactivos en la fase A
20	VARs_Phase_B		X	Voltamperios reactivos en la fase B
21	VARs_Phase_C		X	Voltamperios reactivos en la fase C
22	VAs_Phase_A		X	Voltamperios en la fase A
23	VAs_Phase_B		X	Voltamperios en la fase B
24	VAs_Phase_C		X	Voltamperios en la fase C
25	Power_Factor_Phase_A		X	Factor de potencia de la fase A

26	Power_Factor_Phase_B		X	Factor de potencia de la fase B
27	Power_Factor_Phase_C		X	Factor de potencia de la fase C
28	W_hours_Total	X	X	Watts-hora totales
29	VAR_hours_Total	X	X	Voltamperios reactivos –hora
30	VA_hours_Total	X	X	Voltamperios –hora
31	Phase_A_Current	X	X	Angulo de corriente fase A
32	Phase_B_Current	X	X	Angulo de corriente fase B
33	Phase_C_Current	X	X	Angulo de corriente fase C
34	Angle_Volts_A_B	X	X	Angulo entre las fases A y B
35	Angle_Volts_B_C	X	X	Angulo entre las fases B y C
36	Angle_Volts_C_A	X	X	Angulo entre las fases C y A

TABLA 4. VARIABLES SELECCIONADAS EN LA CONSULTA DE MEDIDORES, CONSULTA CONSTANTE

En total las consultas constantes para modelos 100S es de 24 variables, y para los modelos 200 y 200S es de 36 variables.

#### 2.1.4 Consultas dinámicas

Uno de los objetivos de este trabajo es utilizar en toda su capacidad la lectura de variables. Pero están las limitantes del medidor como se mencionó anteriormente y el espacio de memoria disponible en los nodos. Estos últimos deben estar preparados para almacenar en algunos casos datos durante periodos prolongados, que pueden ir de periodos de días a semanas. Debido al efecto de aislamiento que padece en ocasiones la red mesh por condiciones externas. La solución a este inconveniente son las consultas dinámicas. El procedimiento consiste en enviar a los nodos de forma individual la dirección y tipo de dato que debe consultar. Esto nos permite seleccionar desde el servidor que variables del mapa de memoria Modbus de un medidor en específico se debe consultar. Esta consulta es un complemento de la consulta de variables fijas, que se presentaron en el apartado anterior.

Esta funcionalidad es nueva el proyecto, ahora el servidor puede enviar indicaciones a los nodos para para que aumenten la cantidad de variables que monitorean, y se hace de forma individual para cada nodo, es decir que dos nodos pueden estar consultando variables adicionales que no corresponden al mismo parámetro, esta característica permite

flexibilidad y optimiza el uso de los recursos. La consulta dinámica brinda un plus al sistema de monitoreo, al presentar la información que se requiera.

## **2.2 Almacenamiento in situ**

Los router MP01 cuentan con un sistema operativo Linux embebido específicamente la versión Kamikaze 8.09.2, que esta basa en OpenWRT [3], una de las características adicionales fue la instalación de SQLite3 en el sistema operativo.

SQLite3 es la versión más reciente de SQLite. Es un sistema de gestión de bases de datos relacional, contenida en una biblioteca escrita en C. El motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. El código para SQLite está en el dominio público y por lo tanto es gratuito para su uso para cualquier propósito, comercial o privado.

La implementación de una base de datos embebida permite organizar y almacenar los datos consultados. Además de proveer una interfaz de consola sencilla y similar a MySQL. Con esta herramienta se creó la base de datos que se denominó db1, que es simplemente un archivo que utiliza SQLite como contenedor de las tablas. Dentro de esta base de datos se crearon la tabla Lecturas en la cual se almacén los datos consultados del medidor, y otra tabla denominada control que sirve para la sincronización y consultas dinámicas.

La FIGURA 11, presenta las instrucciones para crear la tabla Lecturas, esta tabla es en la que se almacenaran las mediciones consultadas. Las primeras tres columnas corresponden a identificadores para la organización y clasificación de los datos. La primera columna es Id\_Lectura, esta columna identifica a lectura tomada, es un valor entero auto incremental automático para mantener congruencia entre los datos almacenados en el nodo y

datos en el servidor principal, cada nueva lectura de variables implica una nueva fila en la tabla, y un aumento en una unidad del campo `Id_Lectura`.

El campo `Id_Medidor`, es el identificador del medidor, cada medidor tiene asociado un número entero que es el correspondiente al tercer octeto de la dirección ip del medidor, como se presentó en la TABLA 1. Por ejemplo, el medidor de derecho tiene como ip 10.3.201.2, por esto su identificador es 201 en la base de datos principal. Al momento de crear la tabla `Lecturas` se debe especificar cuál es el identificador por defecto, este parámetro es importante para que el servidor permita almacenar la información. El siguiente campo corresponde a la estampa de tiempo, es decir el momento en que se realizó la consulta, este dato es automático, cada vez que se agrega un registro, SQLite llena este campo con la fecha y hora que maneja el sistema operativo del router, esta información se guarda en un formato de dato tipo `TIMESTAMP`, que es propio del lenguaje SQL.

```

sqlite3 db1 "CREATE TABLE Lecturas
(Id_Lectura INTEGER PRIMARY KEY AUTOINCREMENT,
Id_Medidor INTEGER DEFAULT 201,
Fecha_Hora TIMESTAMP DATE DEFAULT (datetime('now','localtime')),
Volts_A_N FLOAT DEFAULT 0,
Volts_B_N FLOAT DEFAULT 0,
Volts_C_N FLOAT DEFAULT 0,
Volts_A_B FLOAT DEFAULT 0,
Volts_B_C FLOAT DEFAULT 0,
Volts_C_A FLOAT DEFAULT 0,
Amps_A FLOAT DEFAULT 0,
Amps_B FLOAT DEFAULT 0,
Amps_C FLOAT DEFAULT 0,
Watts_3_Ph_total FLOAT DEFAULT 0,
VARs_3_Ph_total FLOAT DEFAULT 0,
VAs_3_Ph_total FLOAT DEFAULT 0,
Power_Factor_3_Ph_total FLOAT DEFAULT 0,
Frequency FLOAT DEFAULT 0,
Neutral_Current FLOAT DEFAULT 0,
Watts_Phase_A FLOAT DEFAULT 0,
Watts_Phase_B FLOAT DEFAULT 0,
Watts_Phase_C FLOAT DEFAULT 0,
VARs_Phase_A FLOAT DEFAULT 0,
VARs_Phase_B FLOAT DEFAULT 0,
VARs_Phase_C FLOAT DEFAULT 0,
VAs_Phase_A FLOAT DEFAULT 0,
VAs_Phase_B FLOAT DEFAULT 0,
VAs_Phase_C FLOAT DEFAULT 0,
Power_Factor_Phase_A FLOAT DEFAULT 0,
Power_Factor_Phase_B FLOAT DEFAULT 0,
Power_Factor_Phase_C FLOAT DEFAULT 0,
W_hours_Total FLOAT DEFAULT 0,
VAR_hours_Total FLOAT DEFAULT 0,
VA_hours_Total FLOAT DEFAULT 0,
Phase_A_Current FLOAT DEFAULT 0,
Phase_B_Current FLOAT DEFAULT 0,
Phase_C_Current FLOAT DEFAULT 0,
Angle_Volts_A_B FLOAT DEFAULT 0,
Angle_Volts_B_C FLOAT DEFAULT 0,
Angle_Volts_C_A FLOAT DEFAULT 0,
More_Registers TEXT DEFAULT ' ')"
```

FIGURA 11. COMANDOS PARA CREAR LA BASE DE DATOS Y LA TABLA LECTURAS

```

sqlite3 db1 "CREATE TABLE Control(
Fecha_Router TIMESTAMP DATE DEFAULT (datetime('now','localtime')),
Estado TEXT,
Registros_Extra TEXT)"
```

FIGURA 12. CREACIÓN DE LA TABLA CONTROL EN LA BASE DE DATOS DBL

La tabla Control es un intermediario entre el servidor y las operaciones del router MP01. Esta tabla solamente contiene una fila, la primera columna Fecha\_Router, almacena la última fecha utilizada por el router. En un reinicio este campo es consultado, si la hora y fecha del router es anterior a la fecha en este campo se procede a ejecutar la petición de la hora al servidor, para que los datos mantengan su integridad.

El campo estado se cambiara en caso de que exista una incongruencia en la hora y fecha. Los estados disponibles son ACTUALIZADA y PENDIENTE, en caso de estar pendiente no se realizaran consultas al medidor, hasta que se sincronice la hora y el campo tenga el estado ACTUALIZADA.

La tercera columna Registros\_Extra es la que nos permite realizar las consultas dinámicas, aquí se almacenan las instrucciones enviadas por el servidor y las que posteriormente consulta el programa principal, y aprovechar la flexibilidad del sistema, para monitorear variables adicionales.

## 2.3 Consultas al medidor

El sistema de consulta implementado se basa en el sistema que implementa el protocolo XML-RPC desarrollado previamente, pero cuenta con características diferentes e importantes que requirieron de un desarrollo nuevo.

El paquete de ejecutable y scripts consta de 5 elementos, 4 de ellos están en el router en el mismo directorio:

- 1- **main\_control.lua**: Este script escrito en lua coordina la ejecución de todos los demás, es el programa principal que se ejecuta periódicamente, es una tarea programada en el crontab del sistema operativo del router.

- 2- **Shark:** Es un programa escrito en lenguaje C y que se compiló de forma cruzada para poder ejecutarse en el router. Su tarea es consultar los registros que se monitorean de forma constante.
- 3- **Shark\_query:** Binario escrito en C y compilado de forma cruzada. La función de este programa es leer los registros que se le pasan de forma dinámica como parámetros por línea de comandos.
- 4- **XMLPI\_L.lua:** Este script también escrito en lenguaje Lua, se encarga de la transmisión de datos utilizando el protocolo XML-RPC hacia el servidor.

El quinto elemento es un script que se ejecuta en el servidor.

- 5- **XMLRPC\_SERVER\_L.cgi :**Script escrito en lenguaje Perl para la gestión de los datos que envían los nodos, almacenándolos en la base de datos MySQL y consultando la misma para actualizar las consultas dinámicas en los nodos.

A continuación se presenta el flujograma del script principal main\_control.lua, que es el encargado de toda la operación en los routers. Para describir claramente el proceso que ejecuta.

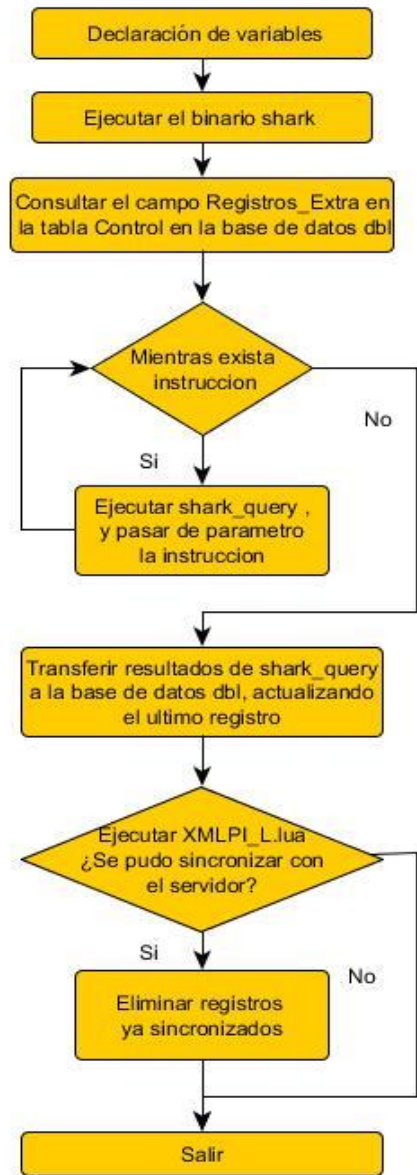


FIGURA 13. FLUJOGRAMA MAIN\_CONTROL.LUA

Un detalle a notar es que solamente se debe modificar las variables iniciales en main\_control.lua para que el script opere correctamente, los datos a modificar son el identificador del medidor, el tipo de medidor en nuestro caso 100 ó 200 , y por último la cantidad de datos que puede sincronizar en cada intento. Esto se presenta la FIGURA 14



```
--Declaramos variables
local shark_query_file="shark_query_result.txt"
local codigo_medidor="201"
local ip_medidor="10.30.."codigo_medidor.."2"
local model_medidor="200"
local limite_registros="10"
--Maxima cantidad de datos a enviar por operacion
```

FIGURA 14. DECLARACIÓN DE VARIABLES MAIN\_CONTROL.LUA

La asignación de variables al inicio de código facilita las substituciones posteriores. Además, son los únicos datos que se modifican para identificar a los medidores. El siguiente paso es la ejecución de los programas como se muestra en la FIGURA 15.

El llamado a los ejecutables se hace con las herramientas de sistema proporcionadas por el lenguaje Lua. Permite ejecutar una instrucción similar a la línea de comandos de Linux. Para ejecutar los programas deben ir precedido por ./ para indicar que se desea ejecutar. A continuación se pasan los parámetros al ejecutable shark. Los parámetros son la ip del medidor y el modelo del medidor, esto es para funciones internas como se verá más adelante. Además nos permite ejecutar el mismo binario para diferentes versiones de medidores. Los siguientes pasos se hacen de forma similar, que es la consulta a la base de datos dbf, utilizando nuevamente herramientas de sistemas ejecutamos una consulta con sqlite3, los datos obtenidos son capturados por la variable handler y se transfiere la variable result en formato de texto, posteriormente se procede a extraer una instrucción a la vez. Cada instrucción se pasa como parámetro al binario shark\_query junto con la dirección ip del medidor, los resultados se almacenan de forma consecutiva en un archivo de texto. Cuando el proceso finaliza se actualiza el campo More\_Registers en la última fila insertada.

```

--Ejecutamos primero el programa shark
print('Ejecutando shark')
os.execute("./shark "..ip_medidor.." "..model_medidor)

--Lectura de indicaciones en la tabla control
print('Buscando Indicaciones en la tabla Control')
local handler = io.popen("sqlite3 dbl 'select Registros_Extra from Control'")
local result = handler:read("*a")
handler:close()
--print (result)
--Formateo de los datos y paso de parametros para shark_query
for x in string.gmatch(result, "[^,]+") do
    os.execute("./shark_query "..ip_medidor.." "..x)
end

--Leemos los resultados de shark_query
local file=io.input(shark_query_file)
local value=file:read("*a")
value =string.sub(value,1,-2)--Descartamos la coma que se genera al final
value = string.gsub(value,",",":")
print(value)

--Obtenemos el max id en la tabla Lecturas
handler = io.popen("sqlite3 dbl 'SELECT MAX(Id_Lectura) FROM Lecturas'")
local max_id=handler:read("*a")
handler:close()

--Guardamos los valores leidos en el campo correspondiente
os.execute("sqlite3 dbl 'UPDATE Lecturas SET More_Registers=\""..value.."\" WHERE Id_Lectura= \"..max_id..\"'")
os.execute("rm "..shark_query_file)
--Intentando sincronizacion
os.execute("/usr/bin/lua XMLPI_L.lua "..codigo_medidor.."
"..limite_registros)

```

FIGURA 15. EJECUCIÓN DE PROGRAMAS SHARK Y SHARK\_QUERY

Por último se ejecuta el script correspondiente al envío de datos por XML-RPC. El script XMLPI\_L.lua se encarga del envío de los datos al servidor. Se debe pasar como parámetro el identificador del medidor y el límite de registros a enviar. un punto a tomar en cuenta en esta parte es que el envío de datos debe ser mínimo en nodos en los que los enlaces fluctúan constantemente, lo que deja ventanas de tiempo muy cortas para enviar cantidades grandes de registros. Así que se ha configurado en 10 la cantidad de filas a enviar, para aprovechar la comunicación con el servidor.

### 2.3.1 Software de consulta constante

El software está escrito en lenguaje c y se compila de forma cruzada para poderse ejecutar en el sistema operativo que tiene arquitectura MIPS. El archivo fuente requiere 3 librerías, la primera librería es modbus.h, esta contiene las funciones y variables necesarias para entablar la comunicación utilizando el protocolo modbus. La siguiente librería es sqlite.h, esta contiene métodos y procedimientos para interactuar con bases de datos SQLite. La última librería corresponde a la desarrollada para este software, se denominó sharkmeter.h y su detalle está en los anexos, esta librería contiene la declaración de las variables y estructuras para la ejecución del programa.

En la FIGURA 16, se presenta la asignación de variables en shark.c y como se pasa por parámetro el tipo de medidor esto se refiere al modelo si es shark 100 o shark 200. Otras variables importantes son los punteros para la comunicación utilizando el protocolo modbus y el puntero para las base de datos SQLite3. Estos elementos interactúan con elementos externos a shark.c

```
#include <modbus.h>
#include "sharkmeter.h"
#include "sqlite3.h"

#define SHARK_METER_SERVER 0x01//Direccion asignada al servidor

int main(int argc, char*argv[])// argv = ip_medidor tipo_medidor[200,
100]
{
    //Variable utilizadas por el software
    modbus_t *ctx=NULL; //Crea puntero
    sqlite3 *db;
    int rc, rc_sql,srk_long,i;
    int shark_model=strtol(argv[2],NULL,10);
    char *zErrMsg=0;
    struct data *srk; //Puntero a variables
```

FIGURA 16. DECLARACIÓN DE VARIABLES EN SHARK.C

Los pasos siguientes son la asignación de variables, el establecimiento de la conexión al medidor utilizando modbus, y abrir la base de datos para la escritura. Existen procesos de validación para estos procedimientos, lo que interrumpe la ejecución en caso de no establecer comunicación con los objetivos como el medidor o la base de datos dbl. La secuencia de instrucciones se presenta en la FIGURA 17.

```

printf("Asignacion de variable: OK\n");/* Establecer contexto MODBUS */
ctx = modbus_new_tcp(argv[1], 502); //contexto TCP, ip y puerto
if (ctx == NULL) {
    fprintf(stderr, "No se pudo establecer el contexto MODBUS en %s\n",
argv[1]);
    return -1;    }
printf("Contexto MODBUS: OK\n");
/* Establece el número del Slave en el contexto TCP */
modbus_set_slave(ctx, SHARK_METER_SERVER);
printf("set slave MODBUS: OK\n");
/* Establece la conexión SQLITE*/
rc_sql = sqlite3_open("dbl", &db); //La base de datos es dbl
if( rc_sql ){
    fprintf(stderr, "Error en la apertura de la base de datos: %s\n",
sqlite3_errmsg(db));
    sqlite3_close(db);
    exit(1);}
printf("SQLITE open dbl: OK\n");
/* Establece la conexión MODBUS */
if (modbus_connect(ctx) == -1) {
    fprintf(stderr, "No se pudo establecer la conexión modbus con %s.\n
Fallo: %s", argv[1], modbus_strerror(errno));
    modbus_free(ctx);
    return -1;    }
printf("conexión MODBUS: OK\n");
/Nuevo bloque para lectura
if (shark_model==100||shark_model==200)//Si el modelo es correcto
{ //Selecting right variables
    printf("Shark model pass in parameter: OK\n");
    switch(shark_model)
    {
        case 100 : srk=&shark100;
            srk_long=shark100_long; break;
        case 200 : srk=&shark200;
            srk_long=shark200_long; break;
        default : fprintf(stderr, "Error: Parametro tipo medidor
incorrecto, ingresar el tipo de medidor [shark100 | shark200]");
            return -1;
    }
}
printf("Entering to read sharkmeter: OK\n");

```

FIGURA 17. COMUNICACIÓN VIA MODBUS Y APERTURA DE LA BASE DE DATOS CON SQLITE3

```

for (i=0;i<srk_long;i++){
toConvert[0]=0;
toConvert[1]=0;
//Reading and converting data
switch(*srk[i].type){//puntero de puntero importante
case 'F'://Converting to float inverting register
rc = modbus_read_registers(ctx, srk[i].address-1, 2, segment);
if (rc == -1)
{
fprintf(stderr, "FALLO: lectura de variable tipo F en: %s ip=%s|
%s",srk[i].address, argv[1], modbus_strerror(errno));
return -1;
}
toConvert[0]=segment[1];
toConvert[1]=segment[0];
srk[i].value=modbus_get_float(toConvert);
break;
case 'S'://Converting to float if it has 2 segment we ,unsigne/signed
int 32 bits
rc = modbus_read_registers(ctx, srk[i].address-1, 2, segment);
if (rc == -1)
{
fprintf(stderr, "FALLO: lectura de variable tipo S en: %s ip=%s| %s",
srk[i].address,argv[1],modbus_strerror(errno));
return -1;
}
toConvert[0]=segment[1];
toConvert[1]=segment[0];
srk[i].value=(float)MODBUS_GET_INT32_FROM_INT16(segment,0);
break;
case 'i'://Converting to float from unsigned int unsigned/signed16 bits
rc = modbus_read_registers(ctx, srk[i].address-1, 1, segment);
if (rc == -1)
{
fprintf(stderr, "FALLO: lectura de variable tipo i en: %s ip=%s|
%s",srk[i].address, argv[1], modbus_strerror(errno));
return -1;
}
srk[i].value=(int) segment [0];
break;

default :
fprintf(stderr, "Error: Tipo de dato incorrecto, valores posibles F, S,
i");
return -1;
}
}

```

FIGURA 18. PROCESO DE CONSULTA DEL MEDIDOR

Otro paso importante es determinar qué modelo de medidor se está consultando, en la librería sharkmeter.h se encuentra dos variables tipo struct que contienen los nombres y

direcciones de memoria de los datos a consultar, por ello se pasa como parámetro el modelo para seleccionar la variable correspondiendo, y así poder consultar los campos correctos. En la FIGURA 17 se muestra la selección del tipo de medidor y la asignación posterior de una variable `srk` que representa a la estructura de datos correspondiente.

Al seleccionar la variable del tipo de medidor, se procede a iterar en ella para consultar las direcciones de memoria. Estas direcciones indican donde se almacenan las mediciones en el mapa de memoria modbus del medidor. Los registros que almacenas lo datos almacenan la información de una forma determinada así que dependiente del tipo de dato que se lee se deben hacer los ajustes para leerlo correctamente. Esta operación se aprecia en el bloque `switch case` de la FIGURA 18, posteriormente se guarda el valor leído en la variable `struct` utilizada en su campo correspondiente.

Para finalizar la ejecución se procede a almacenar los datos capturados en la base datos, aquí es donde echamos mano de las librerías de SQLite para realizar la operación, la función `gen_query` genera la instrucción SQL en forma de cadena de caracteres que posteriormente es ejecutada para poder almacenar la información. Ver FIGURA 19.

```

printf("Generating query: ...In process\n");
if(gen_query(query,srk_long,srk)==0)
{
    printf("Generated query=%s\n",query);
    rc_sql = sqlite3_exec(db, query, 0, 0, &zErrMsg);
    if( rc_sql!=SQLITE_OK )
    {
        fprintf(stderr, "SQL error: %s\n", zErrMsg);
        fprintf(stderr, "Query: %s\n", query);
        /* This will free zErrMsg if assigned */
        if (zErrMsg)
        {
            free(zErrMsg);
        }
    }
}
else
{
    fprintf(stderr, "Error generado en gen query()\n");
    return -1;
}
printf("Ending program: OK\n");
/* Cierra la conexion MODBUS */
modbus_close(ctx);
modbus_free(ctx);
sqlite3_close(db);
printf("Closing ALL: OK\n");
return 0;
}

```

FIGURA 19. ALMACENAMIENTO EN LA BASE DE DATOS

Al finalizar el procedimiento liberamos la memoria utilizada por las variables en ejecución y se retorna el control al programa principal. En la FIGURA 20 se presenta el flujo grama que resume el funcionamiento.

La librería sharkmeter.h incluye la definición de las estructura de datos de tipo struct data que se ha implementado para almacenar iterar sobre las estructuras, además se define la función que genera la instrucción SQL para almacenar las lecturas tomas.

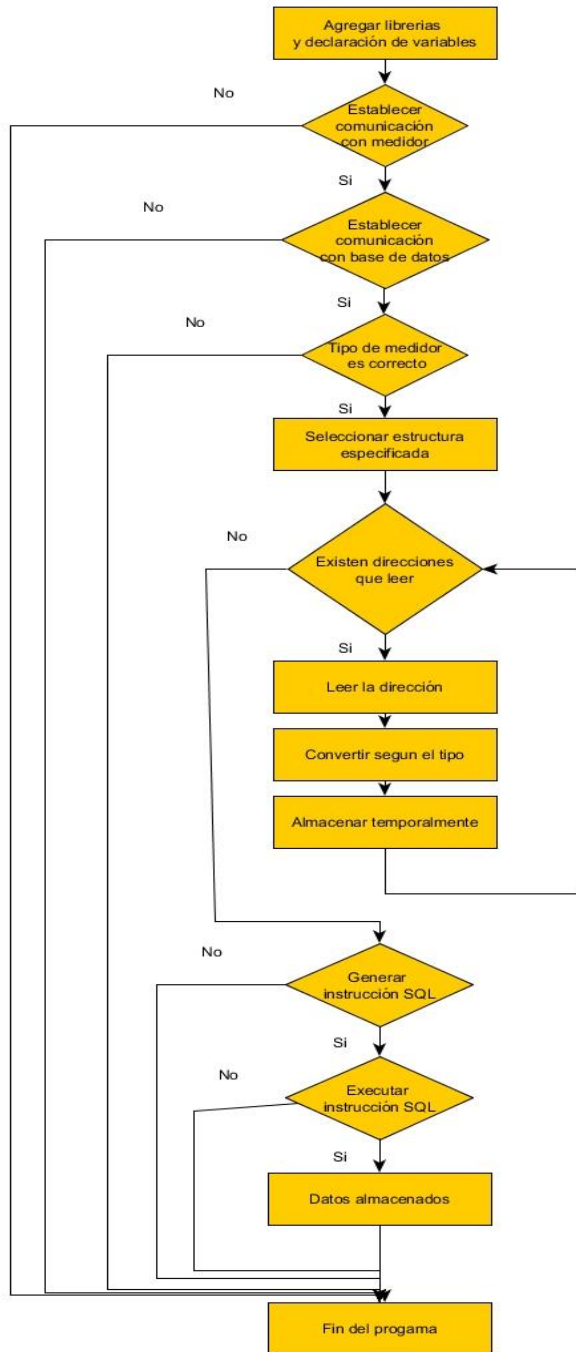


FIGURA 20. FLUJOGRAMA DE EL PROGRAMA SHARK



```

/*****/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>

int shark200_long=36,j; //Number of elements to read from shark
meter
int shark100_long=24;
uint16_t toConvert[2],segment[2];
char query[2048],symbol,words[25];

struct data {
    float value;           //Value converted
    char *type;           //Use to math values "f" "u" "i"
    int address;          //Memory address
    char *name;//[25];     //Variable name
};

int gen_query(char *dest,int sharklong,struct data *dat);

int gen_query(char *dest,int sharklong,struct data *dat)
{
    //printf("\tEntering gen_query: OK\n");
    memset(dest,'\0',strlen(dest)); //Cleaning the string
    //printf("\tCleaning det=query: OK\n");
    if(sharklong==shark100_long || sharklong==shark200_long)
    {

        sprintf(query, "insert into Lecturas("); //Tabla lectura
        //printf("\tFirst step building query: OK\n");
        for(j=0;j<sharklong;j++)
        {
            sprintf(dest+strlen(dest),dat[j].name);
            //printf("\t\tAdding field ok: OK\n");
            if(j<(sharklong-1)) //Mientras no se el ultimo elemento
            {
                sprintf(dest+strlen(dest),",");
            }
        }
        //printf("\tSecond step building query: OK\n");
        sprintf(query+strlen(query),") values(");
        //printf("\tThird step building query: OK\n");

        for(j=0;j<sharklong;j++)
        {
            sprintf(query+strlen(query),"%f",dat[j].value);
            //printf("\t\tAdding value ok: OK\n");
            if(j<(sharklong-1))
            {
                sprintf(dest+strlen(dest),",");
            }
        }
    }
}

```

FIGURA 21. SHARKMETER.H

### 2.3.2 Software de consulta de variables dinámicas

La consulta de variables en forma dinámica sigue un procedimiento similar. El objetivo de este ejecutable es consultar una dirección de memoria en específico, para ello es necesario indicar al software que dirección de memoria es la que debe consultar y el tipo de dato que debe leer. El objetivo principal que alcanzamos con esta aplicación es extender las capacidades de consulta sin comprometer la memoria con cantidades innecesarias de lecturas. El código fuente escrito en c se presenta en la FIGURA 22.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <modbus.h>
#define SHARK_METER_SERVER 0x01 //ID asignado al medidor
int main(int argc, char*argv[])// sharkquery SHARKMETER_IP ADDRESS_DECIMAL FORMATO(1,2,3)
{
    //Variable utilizadas por el software
    modbus_t *ctx=NULL; //Crea puntero
    int rc, i;
    int dir=strtol(argv[2],NULL,10);
    int tipo_dato=strtol(argv[3],NULL,10);
    float value;
    uint16_t segment[2],toConvert[2];
    char *zErrMsg=0;
    FILE *f;
```

FIGURA 22. INICIO DE SHARK\_QUERY

Esta versión del software no incluye la librería SQLite. Esto hace más liviano el programa en comparación con el software para variables fijas descrito anteriormente. Una ventaja de esto es que no utiliza mucha memoria de los router. Ahora, la solución utilizada para dotar de flexibilidad a la aplicación, es crear un archivo de texto de manera temporal. En este archivo se guardan los resultados de la aplicación que posteriormente se agregan a la base de datos dbf. Cuando los datos del archivo de texto han sido almacenados en la base de datos dbf, se procede a borrar el archivo de texto. En la FIGURA 22 se muestra la inclusión de las librerías necesarias, y la asignación de los valores que se pasan como parámetros, estos datos son la ip del medidor, la dirección de memoria en formato decimal y el

tipo de dato que se va a leer, el tipo de dato se ha clasificado en 3 categorías, que se enumeran de 1 a 3. El detalle se presenta en la TABLA 3.

Categoría	Tipo de dato
1	El primer caso representa al tipo de dato más común encontrado en las mediciones que se toman, que son FLOTANTE, este tipo de datos ocupa dos registros de la memoria.
2	El siguiente tipo de dato es un entero que también ocupa dos registros para representarse son del tipo UNSIGNED y SIGNEDINT, también utiliza dos registros
3	El último tipo de dato del cual echamos mano es el entero de 16 bits, que puede leerse de un solo registro.

TABLA 5. TIPOS DE DATOS PARA EL PROGRAMA SHARK\_QUERY

Esta clasificación se debe a que la representación de los tipos de datos en los medidores no es uniforme en los registros y se debe realizar un proceso de conversión a nivel de bits para extraer la lectura correcta. Una descripción más detallada de los tipos de datos, se presenta en el ANEXO A-1.

```

f=fopen("shark_query_result.txt","a");
ctx = modbus_new_tcp(argv[1], 502); //U contexto TCP, ip y puerto

if (ctx == NULL) {
    fprintf(stderr, "\nERROR: en shark_query No se pudo crear el
enlace modbus TCP\n");
    return -1;    }

/* Establece el número del Slave en el contexto TCP */
    modbus_set_slave(ctx, SHARK_METER_SERVER);

/* Establece la conexión MODBUS */
if (modbus_connect(ctx) == -1) {
    fprintf(stderr, "\nERROR: en shark_query:Fallo intento de conexión
MODBUS\n");
    modbus_free(ctx);
    return -1;    }

```

FIGURA 23. ESTABLECIMIENTO DE COMUNICACIÓN CON EL MEDIDOR EN SHARK\_QUERY

Continuando con el código fuente del shark\_query, el siguiente paso consiste en establecer comunicación con el medidor de energía eléctrica utilizando la librería modbus y abrir el archivo de texto que se utiliza como intermediario para almacenar los resultados temporalmente. Al no contar con la librería SQLite el binario pierde la capacidad de utilizar la base de datos, el archivo de texto brinda un consumo mucho menor de memoria utilizada por el software, el nombre del archivo temporal es shark\_query\_result.txt, los resultados se almacenan separados por comas si hay más de uno.

El paso siguiente es la verificación del tipo de dato pasado como parámetro, si es correcto se procede a realizar la lectura del registro almacenado en el medidor de energía, y se convierte según cada caso. Utilizando las herramientas de la librería modbus para consulta y conversión de tipo de datos, cuando esta operación finaliza se procede a liberar la memoria reservada por la aplicación y a cerrar las conexiones establecidas con el medidor, y se libera el archivo de texto creado. Ver FIGURA 24.

Se realiza la operación con una dirección de memoria cada vez, el llamado a la aplicación shark\_query lo hace el programa principal main\_control.lua y es el encargado de pasar por

referencias los valores necesario para la ejecución del programa. Cuando el programa principal termina de interactuar con shark\_query para las consultas dinámicas, procede a leer el archivo de texto que almacena los resultados, estos datos se insertan en la base de datos SQLite en la tabla Lecturas, en el campo More\_Registers de la última fila agregada.

```

if((tipo_dato>0) && (tipo_dato<4)){
    switch (tipo_dato)
    {case 1: //Float
      rc = modbus_read_registers(ctx, dir-1, 2, segment);//Lectura del medidor
      if (rc == -1)
      {fprintf(stderr, "\n ERROR: en shark_query: Fallo intento de lectura de
variables en dir=%d",dir);
      return -1; }
      toConvert[0]=segment[1];
      toConvert[1]=segment[0];
      value=modbus_get_float(toConvert);
      printf("%d:%f\n",dir,value);
      fprintf(f,"%d %f",dir,value);
      break;
    case 2:
      rc = modbus_read_registers(ctx, dir-1, 2, segment);//Lectura del medidor
      if (rc == -1)
      {fprintf(stderr, "\n ERROR: en shark_query: Fallo intento de lectura de
variables en dir=%d",dir);
      return -1;}
      toConvert[0]=segment[1];
      toConvert[1]=segment[0];
      value=(float) MODBUS_GET_INT32_FROM_INT16(toConvert,0);
      printf("%d:%f\n",dir,value);
      fprintf(f,"%d %f",dir,value);
      break;
    case 3:
      rc = modbus_read_registers(ctx, dir-1, 1, segment);//Lectura del medidor
      //printf("Lectura caso 3\n");
      if (rc == -1)
      { fprintf(stderr, "\n ERROR: en shark_query: Fallo intento de lectura de
variables en dir=%d",dir);
      return -1; }
      segment[1]=0;
      value=(float)(int)segment[0];
      printf("%d:%f\n",dir,value);
      fprintf(f,"%d %f",dir,value);
      break;}}
    else{ fprintf(stderr, "\nERROR: en shark_query , el valor para tipo_dato
no es correcto\n posibles valores 1=32bits Float, 2=32 bits UnsignedInt, 3=16
bits Integer");
      return -1;
      /* Cierra la conexion MODBUS */
      fclose(f);
      modbus_close(ctx);
      modbus_free(ctx);
      return(0);}
}

```

FIGURA 24. LECTURA Y ALMACENAMIENTO TEMPORAL SHARK\_QUERY

## 2.4 Envió de datos al servidor principal

El proceso de envío de datos hacia el servidor utiliza el protocolo XML-RPC, este consta de dos elementos, uno se encarga de la codificación en XML de las lecturas tomadas de los medidores y el otro es el encargado de la decodificación de este y su almacenamiento en el servidor principal. El esquema de funcionamiento se presenta en la FIGURA 25

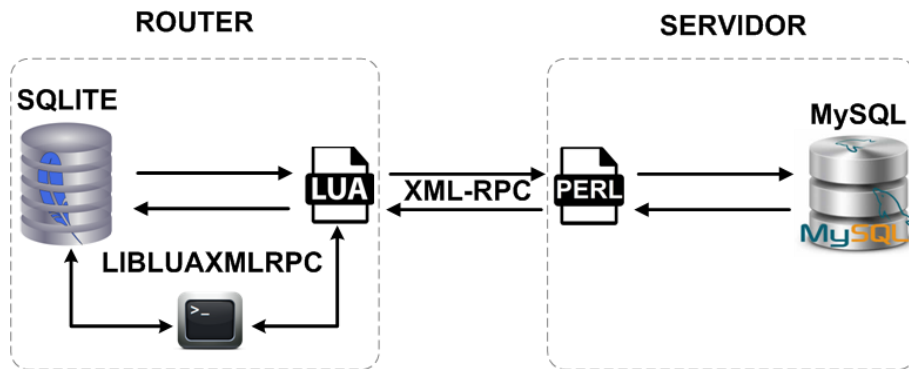


FIGURA 25. ESQUEMA DE COMUNICACIÓN ROUTER-SERVIDOR [3]

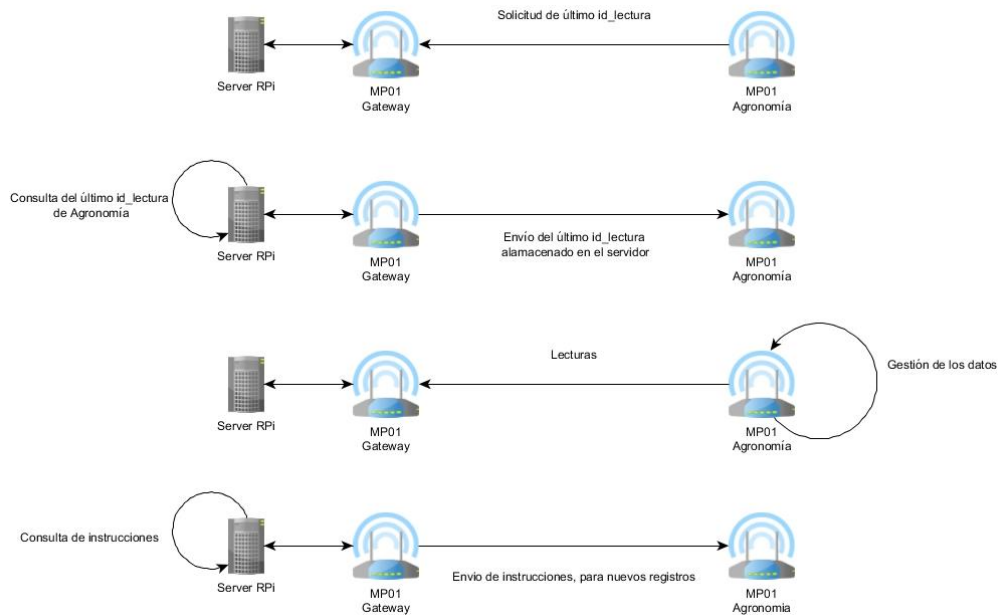


FIGURA 26. PROCESO DE SINCRONIZACIÓN

Los pasos seguidos en el proceso de comunicación son los mostrados en la FIGURA 26, en la que se muestra cómo interactúan los router con el servidor para la sincronización de datos.

### 2.4.1 Aplicación XML-RPC en los routers

Cada router cuenta con un script escrito en lenguaje lua que se encarga de enviar los datos al servidor principal. Se requieren de dos paquetes que proveen a lua de mecanismo de comunicación con HTTP, estos son socket.http y xmlrpc.http. Se importan estos paquetes. Luego el primer paso es solicitar al servidor que tiene una dirección ip que lo identifica en la red mesh que es 10.130.3.150 y los subdirectorios que contiene el script que maneja XML-RPC. Se pasa por argumento a este script el identificador del medidor para que el servidor devuelva el ultimo id correspondiente a ese identificador.

```
require("socket.http")
require("xmlrpc.http")

id = 0
inst=""
local ok, res = xmlrpc.http.call("http://10.130.3.150/cgi-
bin/XMLRPC_SERVER_L.cgi", "XMLRPC_SERVER_L", "Consulta", arg[1])

assert(ok, string.format("Fallo la conexión XML-RPC: %s",
tostring(res)))
for i, v in pairs(res) do
    if(i=="Instrucciones") then
        inst=v
    else
        id = v
    end
    print (i, v)
end
```

FIGURA 27. XMLPI\_L.LUA SOLICITUD DE ÚLTIMO ID\_LECTURA

El siguiente paso es consultar las base de datos y enviar los datos correspondientes a las mediciones faltantes, como se mencionó anteriormente en el script main\_control.lua se

especifica la cantidad de registros a enviar por cada intento, aquí es donde se utiliza este valor, se debe indicar cuanto valores es el máximo a consultar en la base de datos.

Cuando se tienen los datos se debe formatear antes de enviar, ya que se obtiene en el formato devuelto por la consulta utilizando sqlite3 en línea de comandos, devuelve filas, por lo que se debe convertir a una sola cadena separando las filas por paréntesis y comas, así podrá interpretar de forma correcta el servidor, la información enviada. La mayoría de campos son numéricos, pero existen dos que son cadenas de caracteres así que su interpretación es diferentes, estos se diferencian por apostrofes al inicio y final de la cadena. El código que realiza esta tarea se presenta en la FIGURA 28.

```
local handle = io.popen("sqlite3 -separator $', ' db1 'select * from
Lecturas where Id_Lectura > " .. id .. " order by Id_Lectura limit
"..arg[2].."") --arg[2] cantidad de datos a leer
local result = handle:read("*a")
handle:close()
--print(result)
consulta=""
i=1
j=0
--Formateo de los datos
for x in string.gmatch(result, "[^\n]+") do
    consulta=consulta.."("
    for y in string.gmatch(x, "[^,]+") do
        --Adicional creacion de la consulta
        if (i==3 or i==40) then --Hora_Fecha y More_Registers
            consulta=consulta.."'"..y.."',"
        else
            consulta=consulta..y..", "
        end
        i = i+1
    end
    consulta = string.sub(consulta,1,-2)--Eliminamos la última coma
    consulta=consulta.."),"--Cierra primer bloque
    j = j+1
    i=1
end
consulta = string.sub(consulta,1,-2)--Eliminamos la última coma
```

FIGURA 28. GESTIÓN DE LOS DATOS



Cuando la información tiene el formato correcto se procede al envío, de está utilizando el protocolo XML-RPC, la información es codificada y enviada al servidor. La respuesta devuelta por el servidor debería ser una cadena de caracteres con la palabra OK, esta es interpretada como un almacenamiento exitoso por parte del servidor, y se procede a eliminar de la base de datos del router las filas enviadas, esto se muestra en el fragmento de código en la FIGURA 29.

```
local ok, res = xmlrpc.http.call("http://10.130.3.150/cgi-  
bin/XMLRPC_SERVER_L.cgi", "XMLRPC_SERVER_L", arg[1], consulta, j)  
assert(ok, string.format("XML-RPC call failed on client: %s",  
tostring(res)))  
  
for i, v in pairs(res) do  
    print (i, v)  
    if v == 'OK' then  
        os.execute("sqlite3 dbl 'DELETE FROM Lecturas WHERE  
Id_Lectura < \"..id..\"') --Este id es el previo  
    else  
        if(i=="Instrucciones") then  
            inst=v  
        end  
    end  
end  
end  
else  
    print("Servidor actualizado!")  
end
```

FIGURA 29. ENVÍO DE DATOS AL SERVIDOR

Junto con la instrucción OK, también viene otro parámetro que son las instrucciones nuevas para la consulta dinámica, las instrucciones previas son eliminadas e insertan en la base de datos las más recientes en un proceso de actualización, como se presenta en la FIGURA 30.

```

if inst~= '' then
    print("Guardando nuevas instrucciones")
    print(inst)
    os.execute("sqlite3 dbl 'UPDATE Control SET Regis-
tros_Extra=\""..inst..\""' --Actualizando los nuevos campos
else
    print("No habian instrucciones")
end
print("XMLPI_L Finalizado")

```

FIGURA 30. ACTUALIZACIÓN DE INSTRUCCIONES

### 2.4.1 Aplicación XML-RPC el servidor

La aplicación que se ejecuta en el servidor XMLRPC\_SERVER\_L.cgi, es un script escrito en lenguaje Perl. Se ejecuta en el server bajo de demanda, este se encarga de recibir los datos enviados por los routers y de almacenarlos en la base de datos MySQL en el servidor.

La FIGURA 31 muestra la declaración de variables necesarias para la ejecución de este script , su función principal es almacenar la información en la base de datos. Además debe consultar las instrucciones para cada router y enviárselas como respuesta.

El script captura los parámetros y los asigna las variables \$c1, \$c2 y \$c3, y en base al contenido de cada variables realiza los diferentes procedimientos.

```

use strict;
use Frontier::RPC2;
use DBI;

sub XMLRPC_SERVER_L {
my ($c1, $c2, $c3) = @_;
my $driver = "mysql";
my $database = "medidores_django";
my $dsn = "DBI:$driver:database=$database";
my $userid = "userid";
my $password = "password";
my $table="dashboard_measure";
my $table_node="dashboard_node";
my $fields =
"(Id_Lectura,node_id,Fecha_Hora,Volts_A_N,Volts_B_N,Volts_C_N,Volts_A_B,Volts_B
_C,Volts_C_A,Amps_A,Amps_B,Amps_C,Watts_3_Ph_total,VARs_3_Ph_total,VAs_3_Ph_tó
tal,Power_Factor_3_Ph_total,Frequency,Neutral_Current,Watts_Phase_A,Watts_Phase
_B,Watts_Phase_C,VARs_Phase_A,VARs_Phase_B,VARs_Phase_C,VAs_Phase_A,VAs_Phase_B,
VAs_Phase_C,Power_Factor_Phase_A,Power_Factor_Phase_B,Power_Factor_Phase_C,W_ho
urs_Total,VAR_hours_Total,VA_hours_Total,Phase_A_Current,Phase_B_Current,Phase
_C_Current,Angle_Volts_A_B,Angle_Volts_B_C,Angle_Volts_C_A,More_Registers)";

```

FIGURA 31. DECLARACIÓN DE VARIABLES XMLRPC\_SERVER\_L.LUA

```

my $dbh = DBI->connect($dsn, $userid, $password) or die $DBI::errstr;
my $query1="SELECT MAX(Id_Lectura) FROM $table WHERE node_id= $c2"; my
$query2 = "INSERT INTO $table $fields VALUES $c2
my $sth = ($c1 eq "Consulta") ? $dbh->prepare("$query1") : $dbh-
>prepare("$query2");

my $status = $sth->execute() or die $DBI::errstr;
my $result = ($c1 eq "Consulta") ? $sth->fetch()->[0] : ($status == $c3) ? "OK"
: "Error: $status";

my $etiqueta = ($c1 eq "Consulta") ? "Último Registro" : "Resultado";
#ENVIO DE INSTRUCCIONES
my $query3 =($c1 eq "Consulta")? "SELECT instruction FROM $table_node WHERE id=
$c2" : "SELECT instruction FROM $table_node WHERE id= $c1";#Modificado
my $sth = $dbh->prepare("$query3");
my $status = $sth->execute() or die $DBI::errstr;
my $result2 = $sth->fetch()->[0];
my $etiqueta2="Instrucciones";
$sth->finish();
#CONSULTA DE NUEVAS INTSTRUCCIONES PARA EL MEDIDOR
$sth->finish();
return {$etiqueta => $result,$etiqueta2 => $result2};
}
process_cgi_call({'XMLRPC_SERVER_L' => \&XMLRPC_SERVER_L});

```

FIGURA 32. XMLRPC\_SERVER\_L.CGI, OPERACIÓN

Otra cosa importante en la declaración de variables, es la identificación de los campos que serán escritos por la aplicación cuando ejecute las instrucciones SQL, las demás variables son para el procedimiento de consulta.

La operación del script se presenta en la FIGURA 32. Esencialmente son procesos de selección en base a los comandos o instrucciones que recibe. Estos se le pasan por parámetro, esto define qué tipo de procedimiento debe realizar y que valores debe devolver, como se mostró en la FIGURA 26. El primer procedimiento que realiza es consultar la última lectura almacenada por un router en específico, es decir que recibe como parámetro la instrucción “**consulta**” almacenada en \$c1 y debe ejecutar la instrucción SQL \$query1 que corresponde a buscar el máximo id\_lectura disponible para el código del medidor almacenado en la variable \$c2, esta corresponde al segundo parámetros pasado al script. El resultado de la consulta es devuelto al router solicitante. Esta asignación de las variables cambia según el orden de los parámetros enviados por los routers y esto traza el camino a seguir para el programa. Hasta aquí hemos descrito lo referente a la adquisición, trasmisión y almacenamiento de datos. El siguiente capítulo presenta el desarrollo de la interfaz web, que muestra los datos de una forma clara y sencilla

## CAPÍTULO III: APLICACIÓN WEB

La primera aplicación web desarrollada en el proyecto presentaba los datos de los medidores en formato de tabla o con la posibilidad de exportarlo a una hoja de cálculo. Utilizaba PHP y MySQL. Esta aplicación se ejecutaba en una computadora de escritorio [1]. La siguiente aplicación web fue desarrollada bajo la plataforma App Engine de Google[2]. Esta segunda versión desplegaba sobre una gráfica los datos de watts, watts-hora y costo monetario de un medidor en particular. Estaba almacenada en la nube y podía ser consultada desde cualquier lugar. Esta versión implementó la consulta y transferencia de datos con una microcomputadora raspberry pi[2]. La tercera versión fue una mejora en la integridad de los datos que también utilizó la plataforma App Engine para la visualización de los datos. [3].

Actualmente el servidor basado en raspberry pi ha tenido mejoras considerables como el aumento del espacio de almacenamiento, y el cambio por la versión 3 de la microcomputadora raspberry pi. Su principal función es la presentación de los resultados de las mediciones, esta opción permite seguir la línea de bajo costo. Una de las limitaciones de las aplicaciones web anteriores es que la cuota gratuita está limitada en operaciones, lo que hace que la cantidad de datos que se suben al servidor actualmente este al máximo y solo se presentan datos de una sola variable.

La aplicación web permite la toma de decisiones. Ahora el uso de los medidores no está limitado al consumo de energía eléctrica, sino que se puede monitorear el comportamiento de cada subestación durante determinados periodos de tiempo y ver cómo se comporta el perfil de carga, valores máximos y mínimos. Presenta de cada subestación las tensiones, corrientes, potencia, factor de potencia y otras.

## 3.1 Django

Django es un framework de alto nivel de Python Web que fomenta un desarrollo rápido y un diseño limpio y pragmático. Construido por desarrolladores experimentados, se encarga de gran parte de la molestia de desarrollo web, por lo que puede centrarse en escribir su aplicación sin necesidad de reinventar la rueda [12].

Esta es la descripción del framework por parte de sus creadores, lo cual es muy acertada, esta es la herramienta que se ha utilizado por la capacidad expansión y simplificada que posee. En términos generales Django es un conjunto de librerías y aplicaciones escritas en Python que permiten desarrollar aplicaciones web, cuenta con gestores de bases de datos, servidores web e información geográfica entre muchos otros. Entre las páginas que ocupan Django se encuentra Pinterest, Mozilla, National Geographic e Instagram [12]. Django es una herramienta potente pero es necesario familiarizarse con la lógica de desarrollo que implica que va más allá de diseñar simplemente una página en lenguaje HTML.

### 3.1.1 MVC y MTV

Modelo-Vista-Controlador (MVC) es una arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento [12].

El funcionamiento de Django ocupa el mismo principio MVC con una nomenclatura diferente, lo llaman patrón Model-Template- View (MTV). Es la forma en que se debe desarro-

llar la aplicación es decir, se debe codificar en forma separada Model o el modelo que corresponde a lo referente a bases de datos. Luego están las templates o plantillas que son las páginas escritas en HTML , JavaScript y hojas de estilo CSS. Este framework como muchos otros utilizan plantillas web esto permite que se agilice el desarrollo web y simplifique la codificación. En la arquitectura de diseño MVC las plantillas equivalen a las vistas y por ultimo está la vista o View que actúan como el controlador en MVC, este se encarga de la coordinación de toda la aplicación o negocio como se conoce en el área.

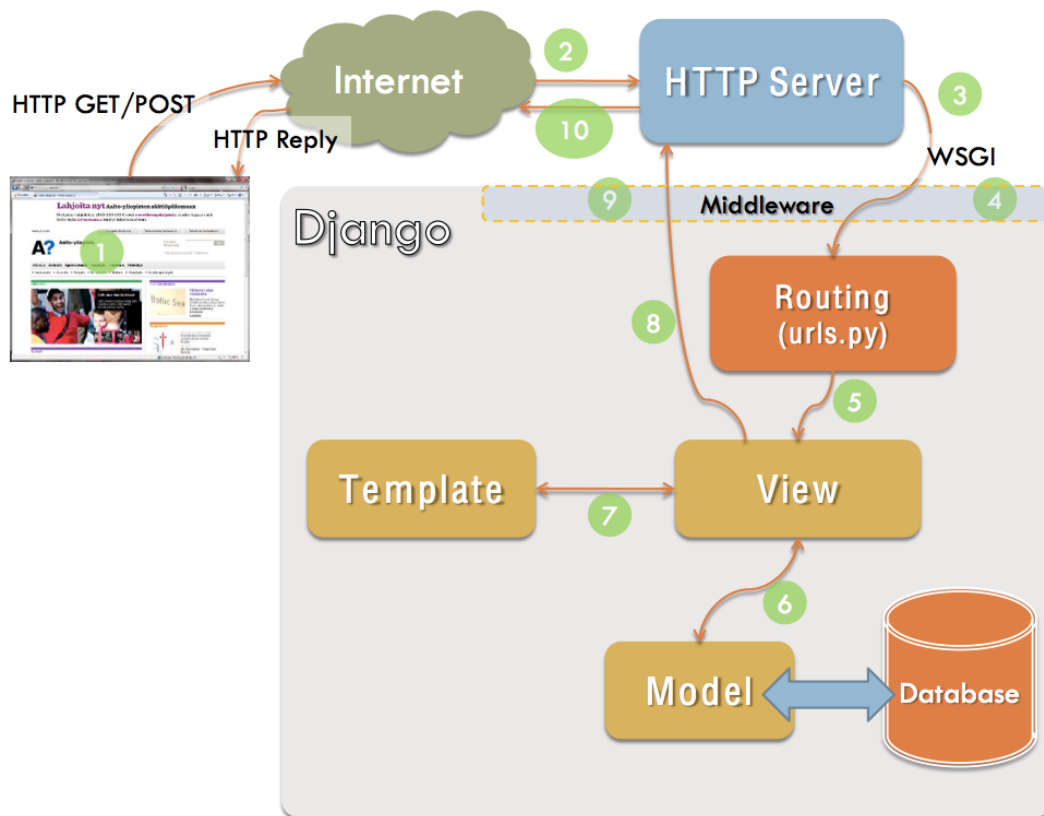


FIGURA 33. PATRÓN MTV

En la FIGURA 33 se presenta un esquema del funcionamiento del modelo MTV que se detallara a continuación:

- 1- Un navegado o web browser envía una solicitud GET o POST a través de internet.

- 2- La petición realiza un viaje a través de la internet hasta llegar al servidor indicado en la URL.
- 3- El servidor web recibe la petición y la dirige al framework responsable de procesarla, en este caso Django, el servidor web comúnmente utilizado es Apache, pero existen otros.
- 4- Django utiliza un módulo de enrutamiento, que lo que hace es identificar a que información quiere tener acceso el navegador, esta información está en la URL, cuando es determinada se llama a la vista especificada.
- 5- La vista recibe la solicitud y la procesa.
- 6- La vista o view solicita la información que se requiere al modelo, este se encarga de extraerla de la base de datos y formatearla de ser necesario, luego la devuelve a la vista.
- 7- El siguiente paso es enviar esta información al template o plantilla correspondiente para generar la visualización, esta es devuelta a la vista.
- 8- Cuando la página está preparada la view o vista se encarga de enviarla como respuesta servidor web.
- 9- Hay un proceso de gestión del envío desde Django al servidor web.
- 10- El servidor web envía la página web al navegador a través de internet.
- 11- La información es recibida y procesada por el navegador.

Utilizando estos principios se puede desarrollar aplicaciones web fácilmente escalables, más detalles del funcionamiento se presentan en el ANEXO B

### **3.1.2 Modelos**

Django utiliza los modelos para crear las tablas y campos en la base de datos. Existen dos modelos (models) o tablas en los que se basa este proyecto, el primero es el modelo de los nodos de la red mesh, la estructura se presenta en la FIGURA 34. Django utiliza clases para definir las herramientas que posee. El nombre de la clase para nodos es Node,



este modelo maneja los detalles de los nodos, como el nombre, si poseen medidor o no, que tipo de medidor tienen , la ubicación geográfica del nodo indicado por coordenadas, la descripción y la ubicación de una fotografía, en fin toda la información concerniente a cada nodo.

```
from django.contrib.gis.db import models
from geoposition.fields import GeopositionField
from django.core.validators import MaxValueValidator, MinValueValidator

class Node(models.Model):
    parent = models.ForeignKey('self', on_delete=models.CASCADE,
blank=True, null=True, related_name='hijos')
    supplier = models.ForeignKey('Supplier', blank=True, null=True)
    name = models.CharField(max_length=20)
    description = models.TextField()
    photography = models.ImageField(upload_to=get_imagen_nodo_dir,
blank=True, null=True)
    location = GeopositionField(blank=True, null=True)

    model = models.IntegerField(null=True)
    ip_medidor=models.CharField(max_length=15,null=True)#
    ip_router=models.CharField(max_length=15,null=True)#
    instruction = models.TextField(null=True)#
    neighbors = models.TextField(null=True)
    capacity_kva = models.FloatField(null=False,default=0.0)
    def __str__(self):
        return "[%d] %s"%(self.id, self.name)
```

FIGURA 34. MODELO NODE DJANGO

Los tipos de datos son objetos en Django, este gestiona los tipos de datos en coordinación con la base de datos. Con el modelo especificado se crea la tabla en la base de datos, los nombres de los campos los asigna Django de forma automática utilizando como referencia el modelo. La clase model es la interfaz que utiliza el framework para realizar operaciones con la base de datos. El siguiente model o modelo es el de lecturas, el nombre de la clase es Measure, este contiene los campos para las variables definidas anteriormente, aquí es donde finalmente quedan almacenados y este modelo es el mecanismo para interactuar con esta información. Los detalles se presentan en la FIGURA 35.

```

class Measure(models.Model):

    Id_Lectura = models.IntegerField(null=True)
    node = models.ForeignKey('Node') #Nodo y Id_Medidor es lo mismo
    Fecha_Hora = models.DateTimeField(auto_now=False, auto_now_add=False, null=True)
    Volts_A_N = models.FloatField(null=True) #
    Volts_B_N = models.FloatField(null=True) #
    Volts_C_N = models.FloatField(null=True) #
    Volts_A_B = models.FloatField(null=True) #
    Volts_B_C = models.FloatField(null=True) #
    Volts_C_A = models.FloatField(null=True) #
    Amps_A = models.FloatField(null=True) #
    Amps_B = models.FloatField(null=True) #
    Amps_C = models.FloatField(null=True) #
    Watts_3_Ph_total = models.FloatField(null=True) #
    VARs_3_Ph_total = models.FloatField(null=True) #
    VAs_3_Ph_total = models.FloatField(null=True) #
    Power_Factor_3_Ph_total = models.FloatField(null=True)
    Frequency = models.FloatField(null=True)
    Neutral_Current = models.FloatField(null=True)
    Watts_Phase_A = models.FloatField(null=True) #
    Watts_Phase_B = models.FloatField(null=True) #
    Watts_Phase_C = models.FloatField(null=True) #
    VARs_Phase_A = models.FloatField(null=True) #
    VARs_Phase_B = models.FloatField(null=True) #
    VARs_Phase_C = models.FloatField(null=True) #
    VAs_Phase_A = models.FloatField(null=True) #
    VAs_Phase_B = models.FloatField(null=True) #
    VAs_Phase_C = models.FloatField(null=True) #
    Power_Factor_Phase_A = models.FloatField(null=True)
    Power_Factor_Phase_B = models.FloatField(null=True)
    Power_Factor_Phase_C = models.FloatField(null=True)
    W_hours_Total = models.FloatField(null=True)
    VAR_hours_Total = models.FloatField(null=True)
    VA_hours_Total = models.FloatField(null=True)
    Phase_A_Current = models.FloatField(null=True) #
    Phase_B_Current = models.FloatField(null=True) #
    Phase_C_Current = models.FloatField(null=True) #
    Angle_Volts_A_B = models.FloatField(null=True) #
    Angle_Volts_B_C = models.FloatField(null=True) #
    Angle_Volts_C_A = models.FloatField(null=True) #
    More_Registers = models.TextField(null=True)

```

FIGURA 35. MODELO MEASURE

Otra clase importante en los modelos para el cálculo de los costos es el correspondiente al pliego tarifario. En este se detallan las tarifas aplicadas y con esto se determinan los costos por cada subestación. El detalle de la especificación de la clase se presenta en la FIGURA 36

```

class Pliego(models.Model):
    supplier = models.CharField(max_length=50, null=True)
    valid_from = models.DateField(blank=False, null=False)
    valid_to = models.DateField(blank=False, null=False)
    cargo_punta= models.FloatField(null=False, default=0.0)
    cargo_valle= models.FloatField(null=False, default=0.0)
    cargo_resto= models.FloatField(null=False, default=0.0)
    cargo_distribucion=models.FloatField(null=False, default=0.0)
    cargo_comercializacion=models.FloatField(null=False, default=0.0)
    cargo_perdidas_transformacion=models.FloatField(null=False, default=0.0)
    recargo_fp1=models.FloatField(null=False, default=0.0)

```

FIGURA 36. MODEL PLIEGO

El modelo del pliego tarifario almacena el supplier o distribuidora, el periodo de validez, y los cargos de punta valle, distribución y comercialización. Además se agregaron los campos para las penalizaciones por pérdidas en transformación y factor de potencia.

### 3.1.3 Template o plantilla

Las plantillas se utilizan para desplegar la página web con un diseño específico y cambiando textos o imágenes este es el principio de reutilización en los patrones MVC o MTV para este caso. El desarrollo principal de la herramienta web de visualización al ser dinámico se ha hecho utilizando JavaScript específicamente el framework Angular.

La plantilla principal en este desarrollo simplemente es la estructura básica de la página, esta contiene la inclusión de librerías y la aplicación escrita en angular. Todo lo demás se ejecuta de forma dinámica con angular en el lado del cliente o navegador.

### 3.1.4 Vistas

En Django las vistas son el centro de la aplicación estas coordinan la visualización, seleccionan las template o plantillas html y la información en los modelos, su función es de-

terminar que solicita el clientes y efectuar los procedimientos necesario para presentar la información correspondientes, no toda la información que devuelve es contenido HTML estrictamente, también pueden devolver datos en concreto, contenido XML y JSON entre otros, su uso es versátil. La vista de la aplicación desarrollada contiene

```
from dashboard.models import Node, Measure, Pliego
from dashboard.serializers import NodeSerializer, MeasureSerializer,
PliegoSerializer
from rest_framework import viewsets, permissions, response as resp
from rest_framework import status
from django.shortcuts import get_object_or_404
from django.conf.urls import url
from django.views.generic import TemplateView
from django.shortcuts import render
from django.conf import settings
from rest_framework.decorators import list_route, detail_route
from django.db.models import Max
from rest_framework.response import Response
import os
```

FIGURA 37. VIEW EN DJANGO, INCLUSIÓN DE ELEMENTOS

La vista depende de todos los otros elementos creados como los modelos y las templates por lo que se deben importar estas para poder utilizarlas, así también todas aquellas librerías que requiera para su funcionamiento. Este programa es la inteligencia o negocio de la aplicación web.

El archivo view.py contiene diferentes clases, cada clase tienen una tarea en particular. La primera clase es la clase que maneja la página de inicio, el nombre de la clase es IndexView, solicita el template de inicio y se envía al cliente. Uno de los parámetros interno que maneja esta clase es la llave de la API de Google Maps, es un identificador que permite utilizar Google Maps en una aplicación web, el detalle se presenta en la FIGURA 38

```

class IndexView(TemplateView):
    template_name = 'main.html'
    http_method_names = ['get']

    def get(self, request, *args, **kwargs):
        return render(
            request,
            template_name=self.template_name,
            context={'API_KEY': settings.GEOPOSITION_GOOGLE_MAPS_API_KEY}
        )

```

FIGURA 38. VIEW, CLASE INDEXVIEW

La siguiente clase es NodeViewSet, esta clase tiene la tarea de retornar los valores de los nodos, interactúa con los modelos Node, esta clase posee dos funciones la primera list, devuelve el listado de elementos, la función retrieve devuelve el detalle de un nodo específico, se utiliza como identificador la variable pk. Cuando se solicita la lista, se ejecuta una aplicación externa a Django que es wifi\_network.pl, esta se encarga del monitoreo de los enlaces y permite visualizar de forma actualizada la condición de la red inalámbrica.

```

class NodeViewSet(viewsets.ModelViewSet):
    queryset = Node.objects.all()
    serializer_class = NodeSerializer
    permission_classes = (permissions.IsAuthenticatedOrReadOnly,)

    def list(self, request, *args, **kwargs):
        queryset = Node.objects.all()
        serializer = NodeSerializer(queryset, many=True)
        os.system("/usr/bin/perl /home/pi/script/wifi_network.pl")
        return resp.Response(serializer.data)

    def retrieve(self, request, pk=None):
        queryset = Node.objects.all()
        nodes = get_object_or_404(queryset, id=pk)
        serializer = NodeSerializer(nodes)
        return resp.Response(serializer.data)

```

FIGURA 39. VIEW, CLASE NODEVIEWSET

```

class MeasuresViewSet(viewsets.ModelViewSet):
    queryset = Measure.objects.all()
    serializer_class = MeasureSerializer

    @list_route(methods=['get'])
    def node_detail(self, request, node=None):
        if 'begin' in request.GET:
            begin = request.GET['begin']
            if 'end' not in request.GET:
                return resp.Response(status=status.HTTP_400_BAD_REQUEST)
            end = request.GET['end']

            if 'vln' in request.GET:#making new filter
                measures = Measure.objects.filter(node__id=node,
Fecha_Hora__range=(begin, end))#Luis Add
                serializer = MeasureSerializer_VLN(measures, many=True)
                return resp.Response(serializer.data)
            else:
                measures = Measure.objects.filter(node__id=node,
Fecha_Hora__range=(begin, end))
            else:
                measures = Measure.objects.filter(node__id=node)
                serializer = MeasureSerializer(measures, many=True)
                return resp.Response(serializer.data)

    @detail_route(methods=['get'])
    def last(self, request, node=None):
        fh= Measure.objects.filter(node__id=node).aggregate(Max('Fecha_Hora'))['Fecha_Hora__max']

        qs = Measure.objects.filter(Fecha_Hora=fh, node__id=node)[0]
        serializer = MeasureSerializer(qs, many=False)
        return resp.Response(serializer.data)

```

FIGURA 40. VIEW, CLASE MEASURESVIEWSET

La FIGURA 40 muestra la clase MeasuresViewSet, esta clase se encarga de devolver valores referentes a las lecturas de los medidores, se deben enviar los identificadores de los medidores para gestionar la información. Existen dos funciones que puede realizar este clase una es devolver los valores entre un periodo de tiempo determinado y la otra devolver el ultimo valor en la base de datos de un nodo en específico, este es el mecanismo que se utiliza para la representación de los datos. La solicitud de los valores de los pliegos tarifarios también se gestionan con una clase en el archivo Views.py, la clase es PliegoViewSet, esta se encarga de devolver los pliegos que estén dentro del periodo especificado en la página web. El detalle se presenta en la FIGURA 41.

```

class PliegoViewSet(viewsets.ModelViewSet):
    queryset = Pliego.objects.all()
    serializer_class = PliegoSerializer

    @list_route(methods=['get'])
    def list(self, request):
        if 'begin' in request.GET:
            begin = request.GET['begin']
            pliegos = Pliego.objects.filter(valid_to__gte=begin)
        else:
            return resp.Response(status=status.HTTP_400_BAD_REQUEST)
        serializer = PliegoSerializer(pliegos, many=True)
        return resp.Response(serializer.data)

```

FIGURA 41. VIEW, PLIEGOVIEWSET

Todas estas clases antes mencionadas deben relacionarse con las direcciones de los enlaces, en el mismo archivo se define esta relación entre View y URL. Esto permite que se seleccione la vista correcta y se pasen los parámetros de esta vista, así como seleccionar la función que realizara tarea. La vista genera las variables necesarias para la ejecución pero la función definida en las vistas son las que procesan los parámetros y devuelven los resultados. La relación entre URL y View se encuentra especificada en el mismo archivo Views.py y se muestra en la FIGURA 42.

```

# Views
node_list = NodeViewSet.as_view({'get': 'list'})
node_detail = NodeViewSet.as_view({'get': 'retrieve'})
measure_list = MeasuresViewSet.as_view({'get': 'node_detail'})
measure_last = MeasuresViewSet.as_view({'get': 'last'})
tariffschedule_list = TariffScheduleViewSet.as_view({'get': 'list'})
pliego_list = PliegoViewSet.as_view({'get': 'list'})

# URLs
urls = [
    url(r'^nodes/$', node_list, name='node_list'),
    url(r'^nodes/(?P<pk>[0-9]+)/$', node_detail, name='node_detail'),
    url(r'^nodes/(?P<node>[0-9]+)/measures/$', measure_list, name='measure_list'),
    url(r'^nodes/(?P<node>[0-9]+)/measures/last/$', measure_last, name='measure_last'),
    url(r'^tariff_schedule/(?P<supplier>[0-9]+)/$', tariffschedule_list, name='tariffschedule_list'),
    url(r'^pliego/$', pliego_list, name='pliego_list'),
]

```

FIGURA 42. VIEW, RELACIÓN ENTRE URL Y VIEWS

Esta es la forma en que Django gestiona el contenido web, a nivel de procedimientos y valores, hasta este punto no se hace uso de herramientas gráficas, una de las característi-

cas del HTTP es que no requiere enviar una gran cantidad de datos para ser visualizados, simplemente envía los mínimo, el navegador es quien traduce las instrucciones y las presenta, una de las herramientas que da dinamismo a las paginas son los scripts.

## **3.2 Angular**

AngularJS (comúnmente llamado Angular.js o AngularJS), es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles [13].

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticos o dinámicos [13]

En este proyecto se utilizó angular para crear páginas dinámicas en las que se realizan operaciones que pasa desapercibidas por el usuario, como la carga de datos, consultas y actualizaciones. Angular permite solicitar información para la generación de las visualizaciones dentro del navegador sin tener que recargar la página para ver esta nueva información. Anteriormente esta característica no estaba disponible, debido a que los contenidos eran estáticos, con el surgimiento de JavaScript esto cambió y las paginas se volvieron dinámicas, las operaciones que no requerían del servidor ahora se realizan utilizando los recursos del mismo navegador web, lo que dotó de mayor velocidad a las aplicaciones web.



Una de las ventajas más grandes que se puede obtener de las aplicaciones web es que ahora se pueden hacer aplicaciones que se ejecuten en diferentes dispositivos, solo es necesario que tenga un navegador web, esto le da movilidad a los usuarios, solo es necesario estar conectado a internet.

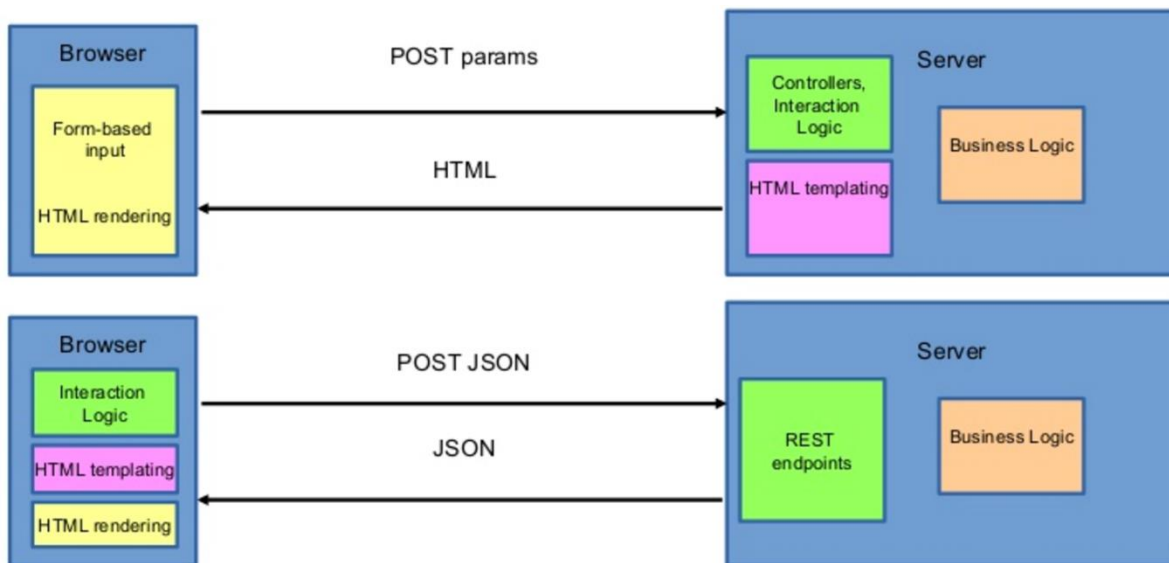


FIGURA 43. ANGULARJS FUNCIONAMIENTO

Una característica que añade velocidad es que ahora con angular, solo se solicita los datos de las mediciones sin necesidad de una plantilla web, solamente se solicitan en formato JSON. Angular se encarga de encajar estos datos en el lugar apropiado y lo hace sin que el usuario pierda la secuencia de operaciones, esto hace que se pueden diseñar paneles de control, monitoreo diferentes fines.

Otra ventaja que provee angular es la interacción con herramientas basadas en JavaScript, como los Google Maps y HighCharts, Angular provee la información necesaria a estas aplicaciones de visualización para que presenten la información solicitada, lo que facilita el trabajo de desarrollo.

### 3.3 HighCharts

Es una biblioteca de funciones escritas en JavaScript que permiten generar gráficas muy amigables y expandibles, cuenta con una gran variedad de modelos y también muy configurables.

Existen variedad de gráficos basados en JavaScript, se seleccionaron esto por los tipos de gráficos que pueden generar, se adecuaban muy bien a los tipos de datos que se querían presentar.

Existen dos tipos de gráficos utilizados en este proyecto, los gráficos de información instantánea y la información histórica, los de información instantánea son los correspondientes a un valor único en el tiempo por ejemplo la medición de potencia en un determinado momento, las mediciones históricas son un conjunto valores que se tomaron a determinados lapsos de tiempo y la gráfica muestra como cambió este en el tiempo.

```
Highcharts.stockChart('history_energy', {
  rangeSelector: {selected: 1},
  title: {text: 'Energía'},
  yAxis: {title: {text: 'Energía'}},
  series: [{name: 'kW-Hora',
    data: vm.W_hours_Total,
    tooltip: {valueDecimals: 2}
  },
  {
    name: 'kVAR-Hora',
    data: vm.VAR_hours_Total,
    tooltip: {valueDecimals: 2}
  },
  {
    name: 'kVA-Hora',
    data: vm.VA_hours_Total,
    tooltip: {valueDecimals: 2}
  }
]
});
```

FIGURA 44. EJEMPLO CON HIGHCHARTS, GRAFICA DE ENERGÍA

El ejemplo de la FIGURA 44 muestra cómo se genera la gráfica de energía. El modo de utilizar las gráficas es ordenar las instrucciones en formato JSON, las gráficas se dividen en dos partes principales, la primera es la especificación de la gráfica, la segunda parte son los valores que debe desplegar. Los valores se pasan como variables que contienen los datos en un arreglo o array. Este es el mecanismo para utilizar las gráficas de HighCharts.

### 3.4 Google Maps

Otra herramienta es Google Maps, esta es un servicio gratuito que proporciona Google con la API de Maps que es un conjunto de librerías escritas en JavaScript. La librería permite desplegar ubicaciones, trazo y otros elementos. Esta herramienta se ha utilizado para el despliegue de los enlaces inalámbricos y la geo ubicación de los nodos, que también corresponden a ubicaciones de subestaciones.

La forma de trabajar con la API es detallar la ubicación inicial, el tipo de mapa, las dimensiones de la visualización, y el nivel de aumento, esto es dentro de la configuración inicial, los siguientes elementos son adicionales, los que corresponden a las ubicaciones y los trazos en diferentes colores. Se ha diseñado un código de colores similar a los utilizados en otras aplicaciones.

Color del trazo	Significado
<b>Verde</b>	Los trazos verdes indican enlaces fuertes, de baja latencia.
<b>Amarillo</b>	Describe atenuación de la señal, latencia perceptible, la transferencia de datos puede presentar retrasos o fallas.
<b>Rojo</b>	Esta condición indica que la comunicación está en malas condiciones, que los enlaces se caen con facilidad y no pueden mantenerse más de un par de segundos.

TABLA 6. CÓDIGO DE COLORES DE ENLACES WIFI

Google Maps no realiza ningún tipo de procedimiento matemático simplemente se le indica los puntos geográficos para los nodos y las coordenadas de inicio y fin de las trayectorias junto con el color correspondiente a la trayectoria. Esta información es gestionada por la aplicación en Angular, que solicita la información del servidor, la procesa y la transfiere a Google Maps.

# CAPITULO IV: RESULTADOS OBTENIDOS

En este capítulo se presentan los resultados como lo son la aplicación web, las gráficas obtenidas de las subestaciones y el monitoreo de los enlaces. La primera parte consta de una descripción de la aplicación web, como se utiliza y que datos son los que muestran. La segunda parte es una descripción de la información que presentan las diferentes gráficas.

## 4.1 Aplicación web

A continuación se describirá como se han distribuido los elementos en la aplicación web, la función de los elementos interactivos y la secuencia que sigue su uso.

### 4.1.1 Inicio de la aplicación



FIGURA 45. APLICACIÓN WEB, PAGINA PRINCIPAL

La página de inicio de la aplicación web está compuesta por 2 elementos importantes, el primer elemento es el más grande y es la visualización de los enlaces utilizando Google

Maps, se puede ver los puntos o nodos, representados por los indicadores de color rojo, los trazos de colores representan los enlaces y su calidad. Se puede utilizar los controles zoom de Google Maps para obtener más detalle de la ubicación y también se puede cambiar el tipo de visualización del mapa. El segundo elemento es un panel de navegación ubicado al lado derecho en el cual están representadas por imagen y nombre los nodos que poseen subestación, estos son enlaces a la información de cada nodo. La FIGURA 45 muestra los elementos de la página de inicio.

Otras cosas que se pueden notar en la pantalla principal son el título de la aplicación y una serie de enlaces en la parte inferior que proveen la vinculación a elementos secundarios de la aplicación como un manual, la sección de configuraciones o settings y una breve introducción del proyecto.

#### **4.1.2 Información de los nodos**

Al dar clic en algún elemento del listado presentado bajo la etiqueta con fondo verde se despliega a la derecha la información del medidor. Esta operación también se puede realizar también al hacer clic sobre el nodo en el mapa y seleccionar ver nodo. El mapa se oculta y aparece un botón con la opción de regresar al mapa, junto al nombre del nodo, está en fondo naranja. Abajo de la barra naranja la aplicación se divide en dos segmentos, el primero una columna que contienen una fotografía de la subestación, y una pequeña descripción de este medidor. El segundo elemento corresponde a los datos de interés que se presentan en pestañas en la FIGURA 46. Los primeros elementos que se encuentran son selectores de fechas, que sirven para indicar el periodo de tiempo del cual deseamos que se muestre los datos, por defecto muestra una semana hacia atrás, a partir del día actual o en el que se efectúa la consulta. Junto a los selectores de fecha, aparece el botón actualizar que permite cargar las gráficas en el periodo solicitado, ya por defecto se cargan las correspondientes a la semana previa al día de la consulta en la aplicación web.

Luego se presentan dos pestañas, por defecto la seleccionada al inicio es la de datos recientes, la otra pestaña es la de datos histórico. Dentro de la primera pestaña se presenta como primer dato el último momento del cual se tienen datos en el servidor, este parámetro sirve como indicador de la última en que se sincronizaron lecturas. El siguiente campo corresponde a las lecturas adicionales, es decir aquellos campos que se desean consultar, pero que no forman parte de las variables comunes. Lo siguiente corresponde a las gráficas que género el último dato enviado. En la pestaña historial de mediciones está el conjunto de gráficas generadas que describen las condiciones de operación de la subestación en el periodo especificado.

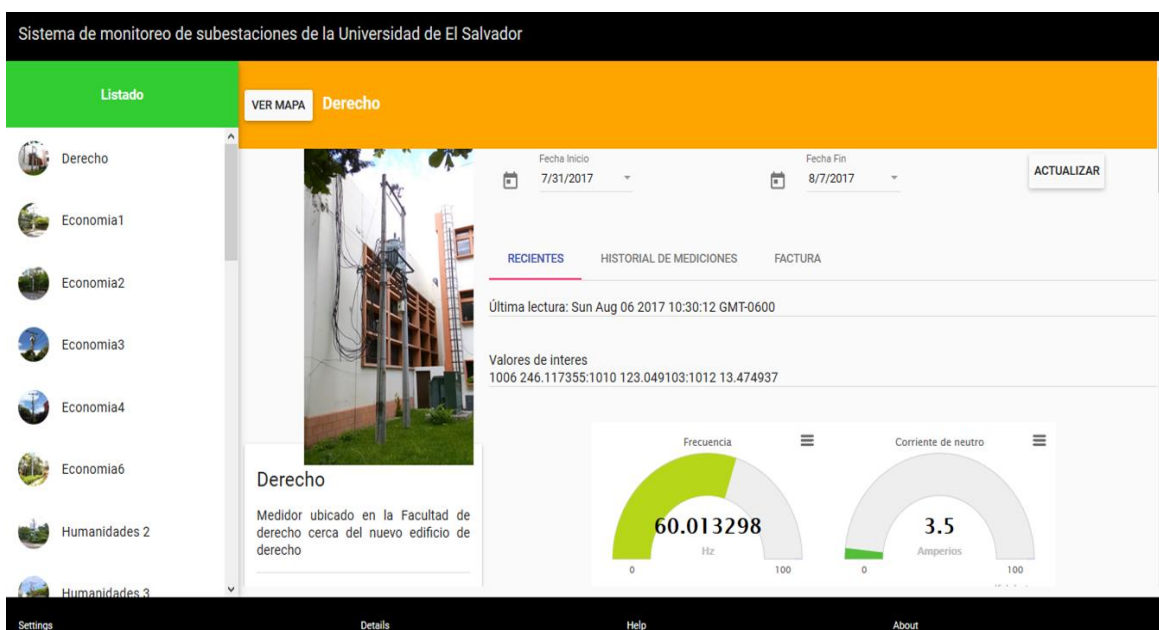


FIGURA 46. DESCRIPCIÓN DE LA PAGINA DE MEDICIONES

### 4.1.3 Mediciones recientes

En la primera pestaña mostrada en la FIGURA 47 corresponde a las mediciones recientes se presentan inicialmente la fecha en que se tomó la medición. El segundo elemento que no se presenta en forma de grafica son los valores de interés. Los valores de interés corresponden a los valores adicionales que se están monitoreando, se representan por el par dirección-valor para poder identificarla, cuando se presentan más de un elementos están separadas por dos puntos (:) entre cada par de datos.



FIGURA 47. DETALLE DE CAMPOS EN LECTURAS RECIENTES

La siguiente información presentada corresponde a las gráficas de frecuencia y corriente de neutro. La gráfica de frecuencia presenta el valor en Hertz a los cuales opera la subestación, el otro dato es la corriente de neutro. Los medidores permiten determinar la corriente de neutro que idealmente en un sistema trifásico en Y debería ser cero, esto en la práctica no es verdad. Dependiendo de la conexión de la subestación, la corriente de neutro puede presentar valores de decenas de amperios. Es importante destacar que no todas las subestaciones están conectadas en estrella, es más la mayoría son monofásicas, por lo que el medidor de energía solo mediría dos fases que corresponden al secundario del transformador.



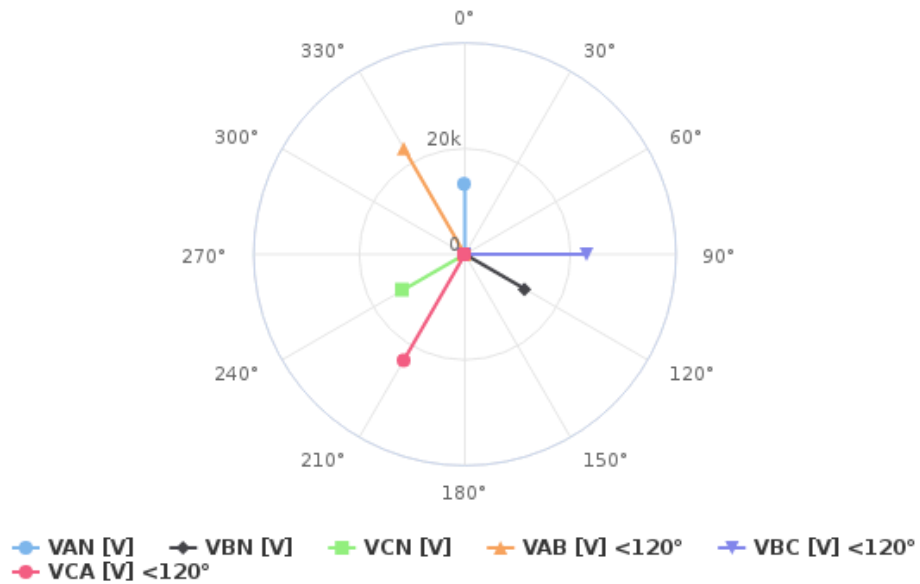


FIGURA 48. VISUALIZACIÓN DE FRECUENCIA Y CORRIENTE DE NEUTRO

Las gráficas de frecuencia y corriente se presentan como un indicado de capacidad, se presenta con un límite preestablecido de 100 amperios y Hertz respectivamente, es para fines ilustrativos, la medición aparece de forma numérica.

El siguiente gráfico corresponde a las tensiones, se presentan como diagrama fasorial. Cada tensión se diferencia por su color, en la parte inferior se presenta el nombre que identifica a la línea, por ejemplo VAB es la tensión entre la fase A y la fase B y VBN es la tensión entre la fase B y neutro. Los datos proporcionados por los medidores solo indican el ángulo entre fases, pero indica el ángulo entre fases y neutro, lo que se ha hecho es utilizar como referencia la fase AN, esta está ubicada a cero grados, y las demás se ubican en base a este. Otra característica es que al poner el puntero sobre algún fasor despliegan un indicar que detalles de del nombre y valor. El grafico se ajusta automáticamente a las magnitudes que se le pasan, el grafico de la FIGURA 49 corresponde a la medición en el primario, por lo que presenta lecturas en kV.

### Diagrama fasorial de voltaje



Hic

FIGURA 49. DIAGRAMA FASORIAL

Siguiendo la misma secuencia se presenta el gráfico fasorial de las corrientes de fase. El diagrama es representativo y muestra la magnitud de las corrientes y el ángulo en que idealmente deberían estar ubicadas. El valor real de ángulo entre las corrientes se presenta junto al nombre de la corriente. Esto se puede apreciar en la FIGURA 50.

El siguiente dato presentado corresponde a los factores de potencia de fase y trifásico este dato utiliza un gráfico de barras para representarlos. El valor preciso de la medición se presenta al poner el puntero sobre cada una de las barras que representan el factor de potencia.

### Diagrama fasorial de corriente

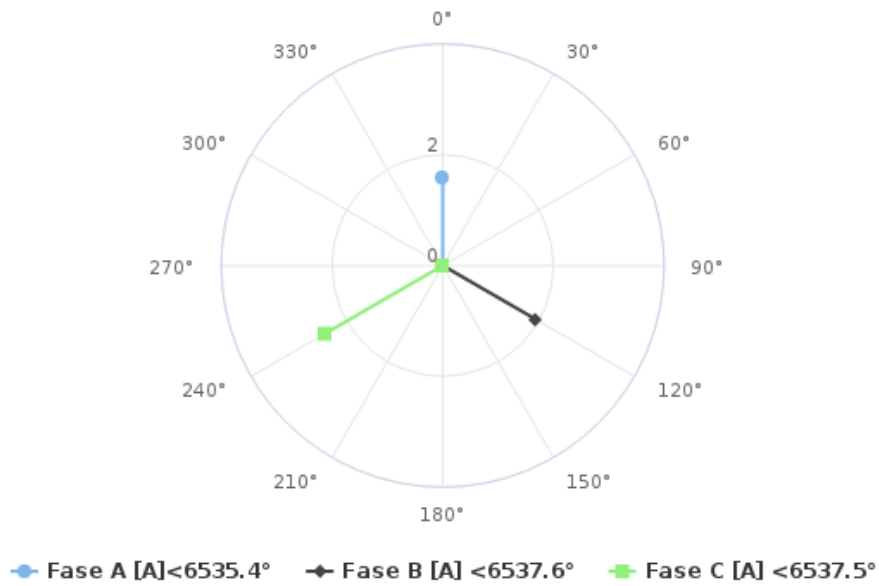


FIGURA 50. DIAGRAMA FASORIAL DE CORRIENTE

### Factor de potencia

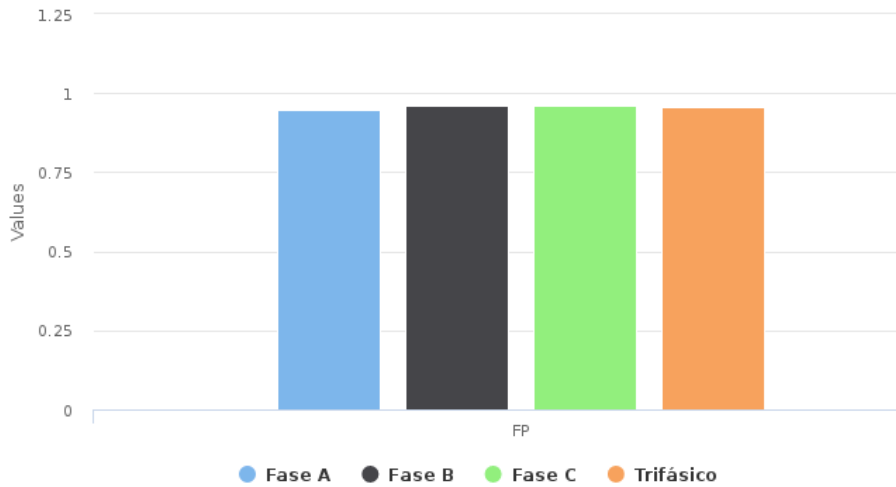


FIGURA 51. FACTOR DE POTENCIA

Por último en esta sección se presentan los datos de la potencia registrada por el medidor. Estos datos corresponden a las potencias activa, reactiva y aparente por fase, también se incluye la misma clasificación para el equivalente trifásico, cada fase presenta los tres valores de potencia en forma de barras de diferentes colores como se muestra en la FIGURA 52. El valor exacto se presenta al ubicar sobre cada barra el puntero.

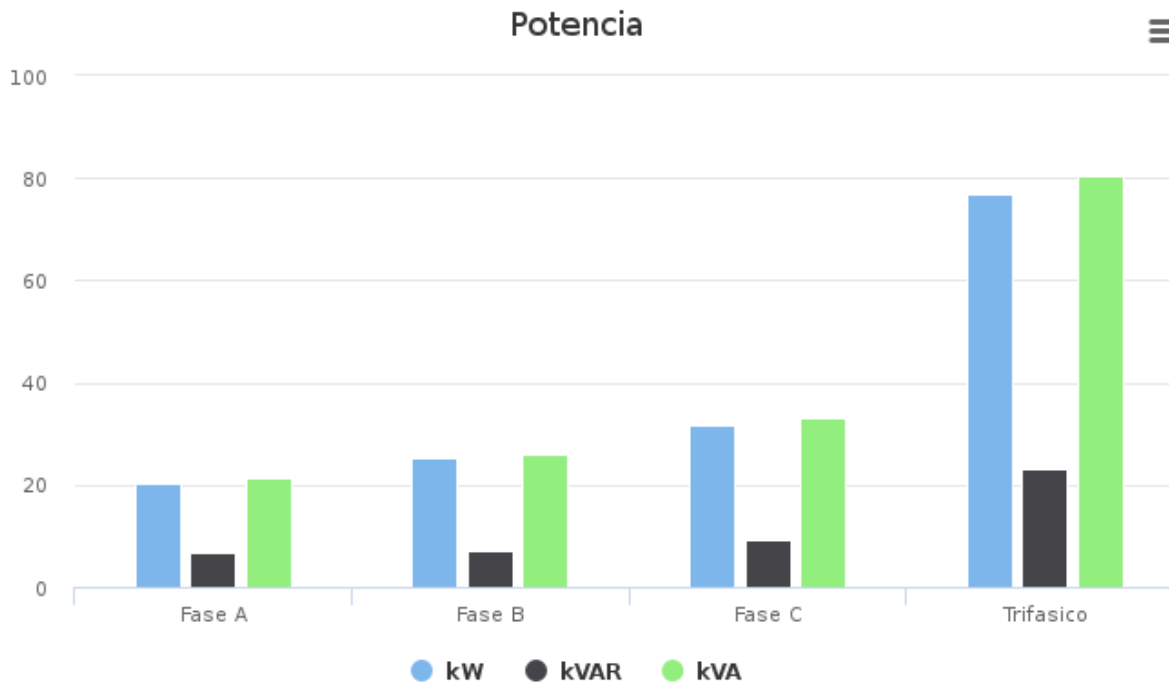


FIGURA 52. GRÁFICA DE POTENCIA ACTIVA, REACTIVA Y APARENTE

Los gráficos anteriores corresponden a las mediciones instantáneas, específicamente a la última registrada en la base de datos del servidor.

#### 4.1.4 Mediciones históricas

La siguiente pestaña de la aplicación presenta el historial de mediciones en el periodo seleccionado. En esta gráfica se puede visualizar el comportamiento de la demanda a través del tiempo. La primera gráfica mostrada en la FIGURA 53 se presenta el gráfico histórico de potencia, en esta representación se muestran la potencia activa (kW) en trazo azul, la potencia reactiva (kVAR) en trazo negro y la potencia aparente (kVA) en trazo verde. Usualmente el valor de la potencia aparente es ligeramente superior a la potencia activa por lo que las gráficas se superponen.

Otra característica de la gráfica de potencia es el trazo de la capacidad (kVA). En base a la capacidad de las subestaciones se grafican las líneas punteadas en negro que indican los límites, el primer límite que se presenta en la gráfica es el de 25% de capacidad, los siguientes son 50%, 75% y 100%. Para que el usuario pueda tener una referencia rápida y comprensible de lo que indica la gráfica, la escala de tiempo es ajustable y se puede desplazar para hacer un aumento de la sección de interés. Por defecto las gráficas de valores históricos muestran el rango completo. La escala de la gráfica de la FIGURA 53 está en kW, kVA, kVAR ya que usualmente se expresan en estas escalas.

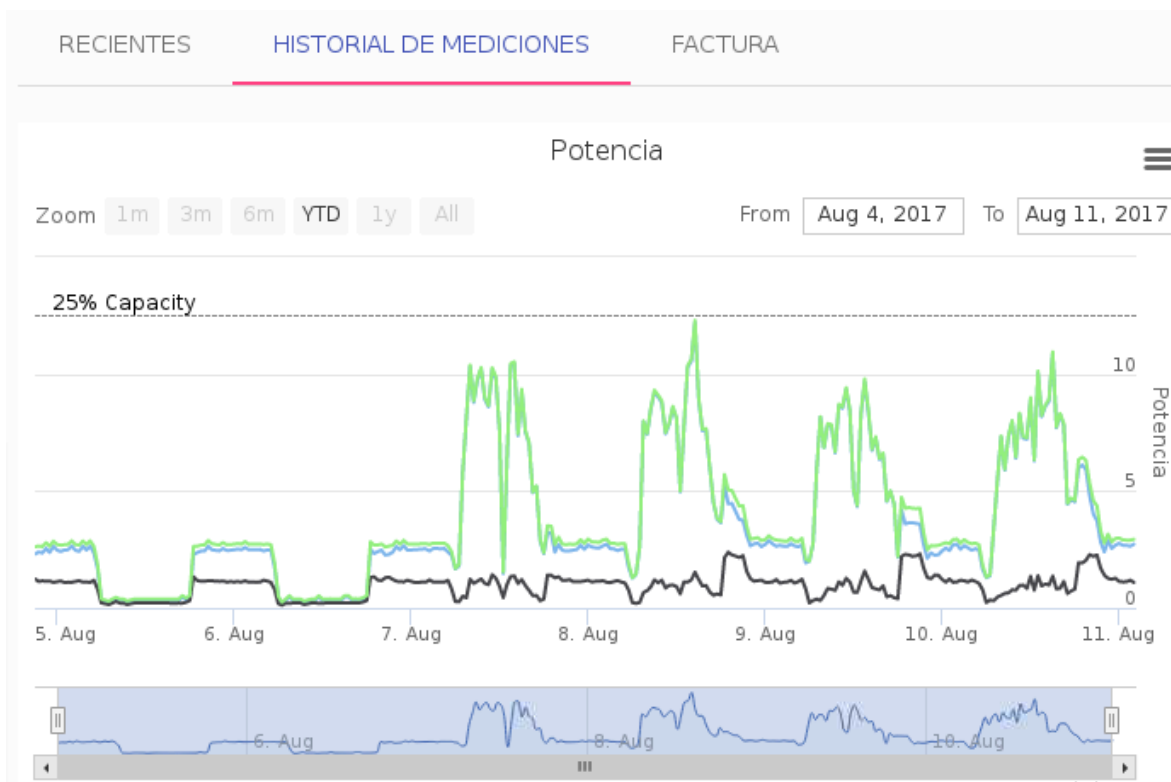


FIGURA 53. VALORES DE POTENCIA DEMANDADA

Las gráficas de potencia también son conocidas como el perfil de carga, debido a que describen las variaciones de la demanda de potencia lo que se traduce en el uso de maquinaria, luminarias o equipo eléctrico. La gráfica de potencia presenta más información al ubicar el puntero sobre esta, y detalla los valores precisos que se están señalando.

Cada gráfica de valores históricos presente los mínimos y máximos para cada trazo, estos valores se presenta bajo cada gráfica y sirven para visualizar de forma rápida los extremos a lo que se ha llegado en el periodo de medición seleccionado.

Potencia Activa (P)	Potencia Reactiva (Q)	Potencia Aparente (S)
MAX: 12.95 [kW]	MAX: 2.40 [kVAR]	MAX: 13.09 [kVA]
MIN: 0.22 [kW]	MIN: 0.00 [kVAR]	MIN: 0.24 [kVA]

FIGURA 54. VALORES MÁXIMOS Y MÍNIMOS

Los valores de máximos y mínimos se presentan en la FIGURA 54, debajo del nombre de la variable se muestra el valor máximo y mínimo con sus respectivas unidades.

El segundo gráfico presentado es el de energía. Este no muestra cambios bruscos en el trazo debido simplemente a que el valor que se mide se va acumulando y a simple vista parece una línea recta. Los datos de máximos y mínimos muestran que efectivamente cambia en el tiempo. De los datos presentados en la gráfica de energía tenemos lo kW-hora en trazo azul, que es el parámetro principal que utilizan las empresas distribuidoras de energía eléctrica para facturas los consumos, otros datos presentados sirven para ilustrar el comportamiento del consumo y son los kVA-hora, y kVAR-hora en trazo verde y negro respectivamente.

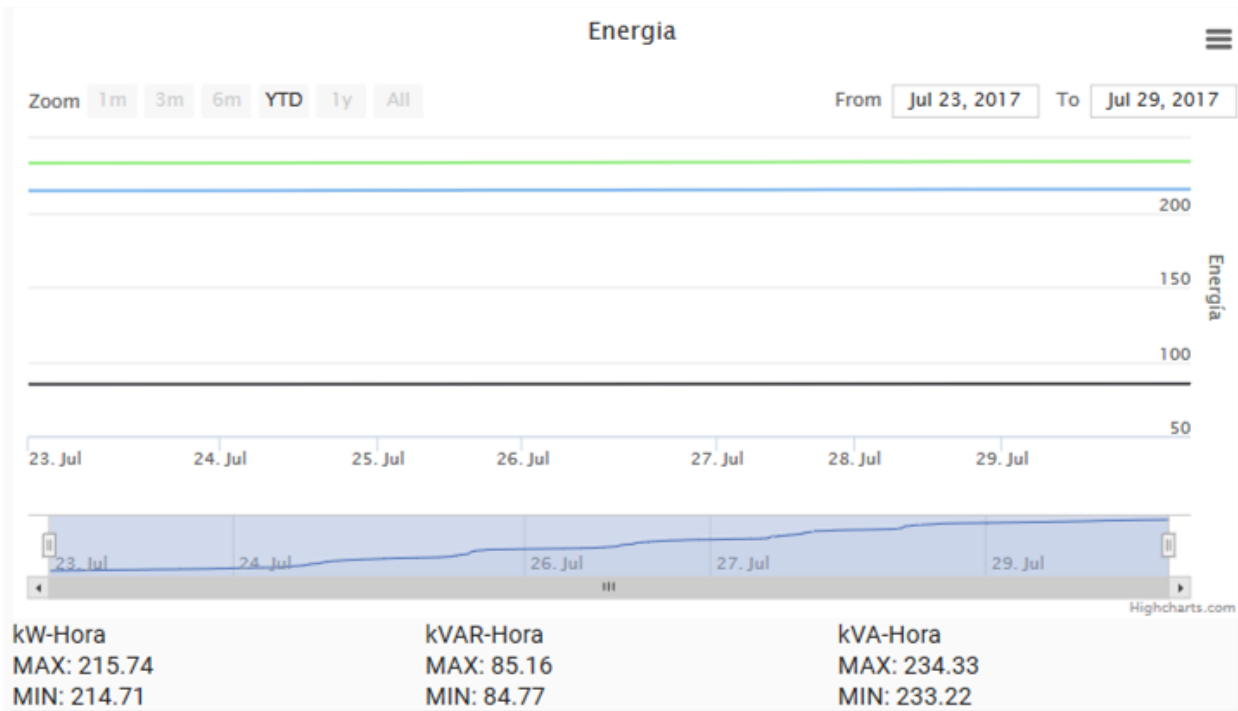


FIGURA 55. DATOS HISTÓRICOS DE ENERGÍA

La tercera gráfica se muestra en la FIGURA 56 y representa los valores históricos de factor de potencia, este parámetro es un valor que relaciona la magnitud de la potencia activa con la potencia aparente, es un valor adimensional. La gráfica de factor de potencia describe como esta relación cambia con el tiempo en las tres fases. Si no es trifásica la subestación, se le asigna 1 de forma constante a las fases no medibles. Otro dato representado es el factor de potencia trifásico.

El factor de potencia es un parámetro que está regulado y que puede acarrear penalizaciones para los usuarios que tengan en sus instalaciones bajo factor de potencia. Por lo que su monitoreo y corrección evitan que las empresas o instituciones que tienen medición de factor de potencia, paguen cobros adicionales por el mal desempeño de su red de energía eléctrica. En el gráfico de factor de potencia los detalles de cada punto se muestran al ubicar el puntero sobre la gráfica, y también se muestran los valores de máximos y mínimos.

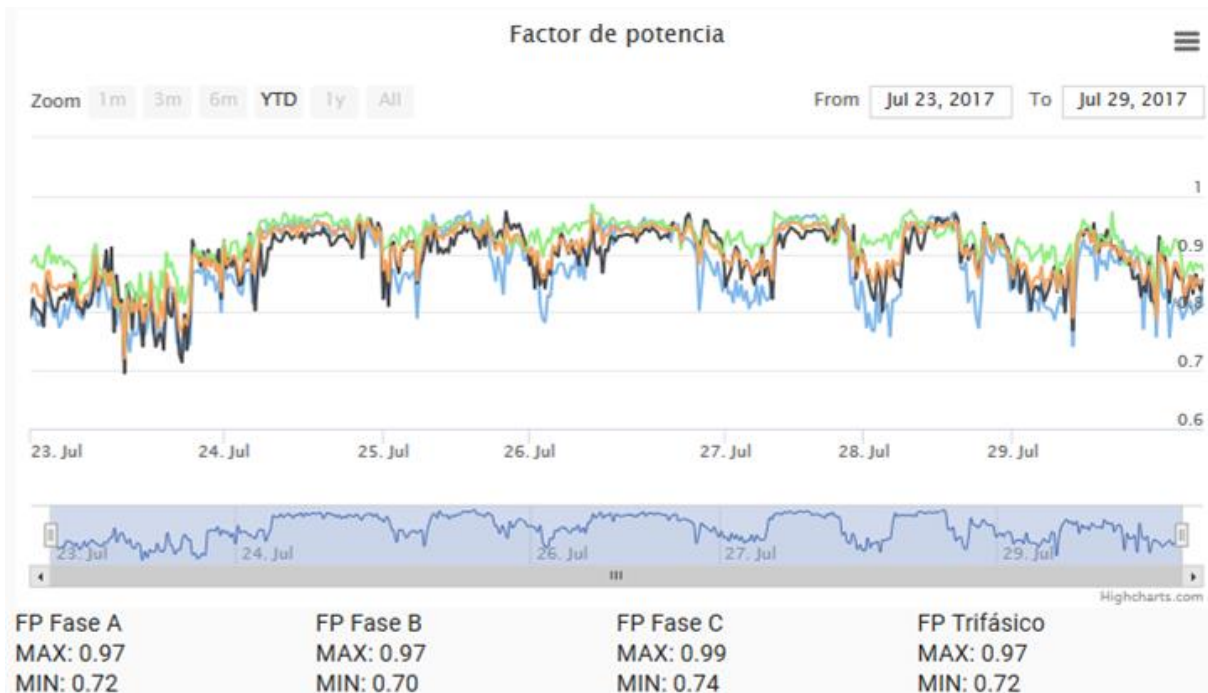


FIGURA 56. DATOS HISTÓRICOS DE FACTOR DE POTENCIA

La cuarta gráfica corresponda a la frecuencia. En nuestro país la frecuencia de operación de la red es de 60 Hz. Los medidores pueden monitorear la frecuencia para el análisis de calidad de energía, este valor usualmente no presente cambios importantes pero se puede ver afectado por fallas en los sistemas de alta tensión de le red distribución. La FIGURA 57 presenta las fluctuaciones en frecuencia que presento la subestación que a pesar de lucir con cambios bruscos al observarla con detenimiento se puede apreciar que la variación está en el orden de las centésimas de Hertz.



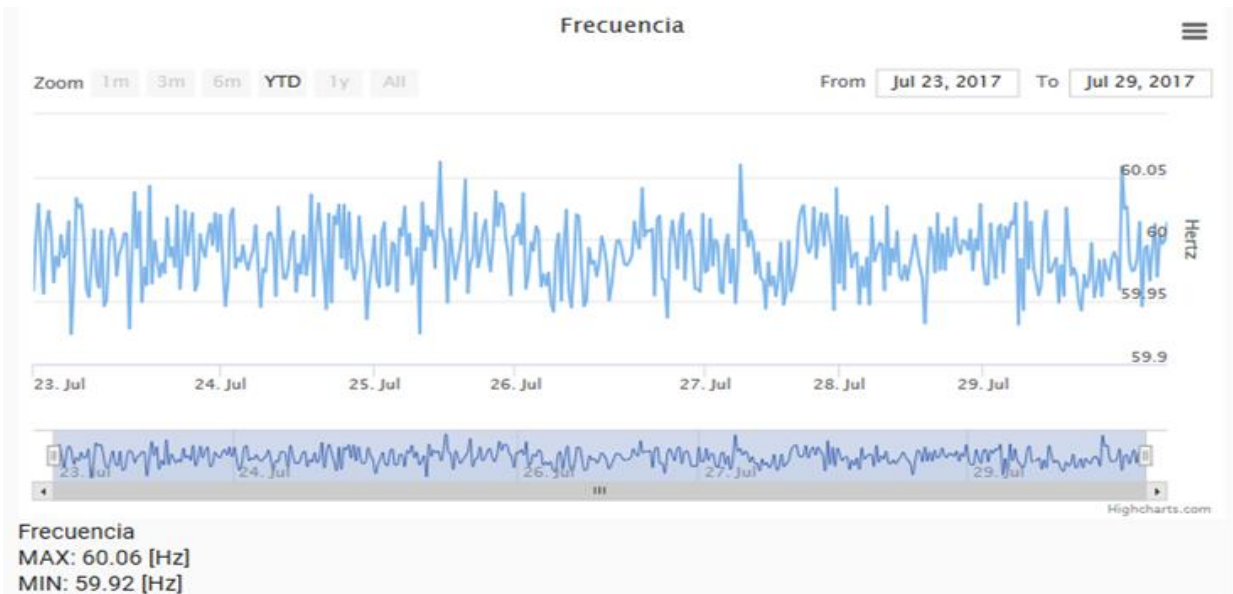


FIGURA 57. DATOS HISTÓRICOS DE FRECUENCIA

El gráfico de la FIGURA 58, corresponde a los valores entre las fases y neutro. En la FIGURA 59 se puede apreciar que la magnitud de los voltajes está en el orden de los miles de voltios, esto es porque la medición presentada corresponde a mediciones en las líneas de alta tensión. Como se puede apreciar existen variaciones del orden de cientos de voltios en las líneas primarias que llegan a la Universidad de El Salvador.

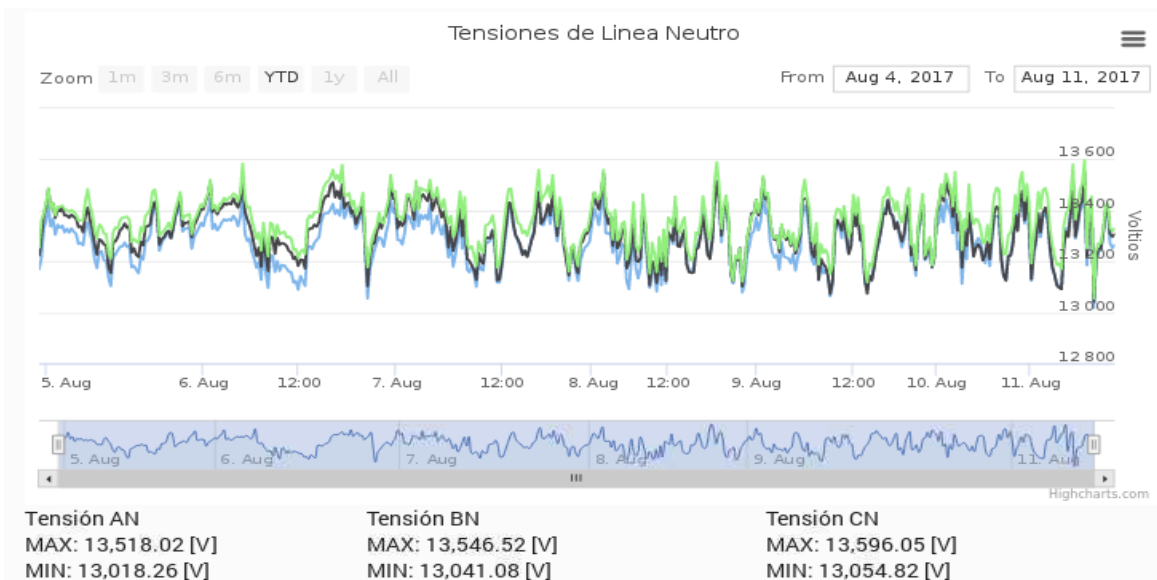


FIGURA 58. DATOS HISTÓRICOS DE TENSIÓN DE LÍNEA A NEUTRO

Otra de las gráficas mostradas que relacionan la tensión y el tiempo es la de tensiones de línea a línea. Los datos mostrados corresponden a mediciones en alta tensión. Se puede observar los datos de valores máximos y mínimos que presentan oscilaciones del orden de hasta mil voltios. Se presentan las tres fases en esta gráfica y además sus valores máximos y mínimos. El detalle se puede obtener al posicionar el puntero sobre el trazo, en algún punto de interés.

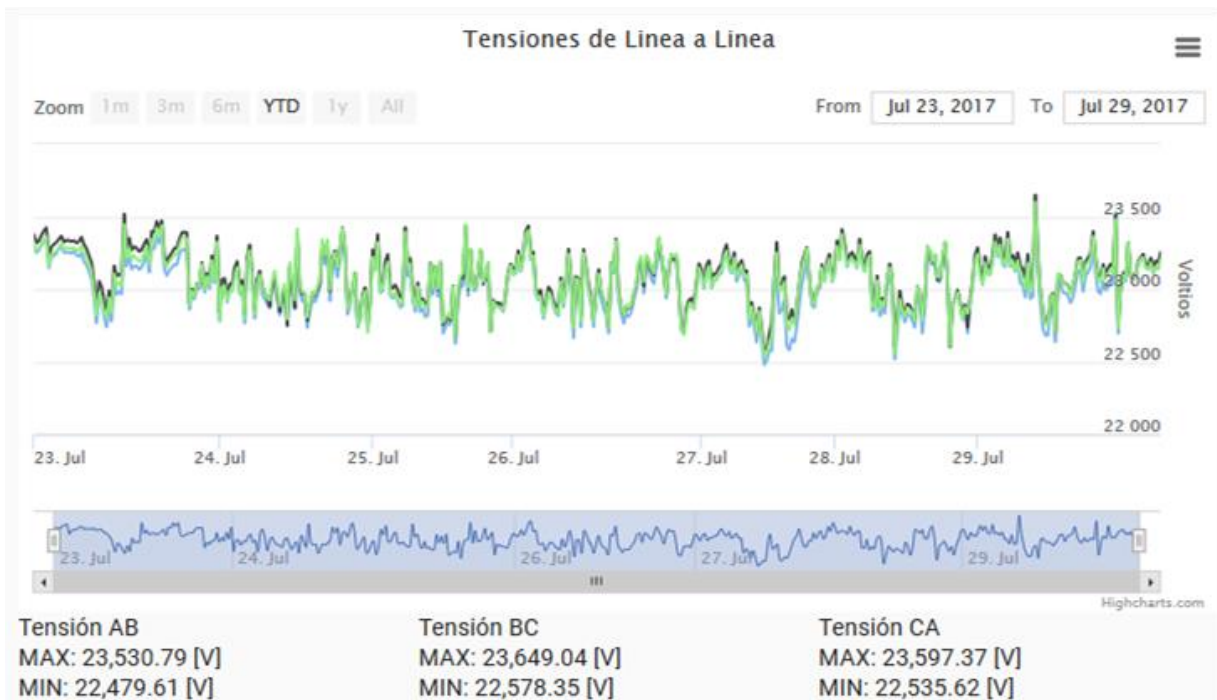


FIGURA 59. DATOS HISTÓRICOS DE TENSIÓN DE LÍNEA A LÍNEA

La séptima gráfica expuesta en la FIGURA 60 describe el comportamiento de las corrientes de fase y neutro. La gráfica tiene una forma similar al perfil de carga, debido a que las variaciones que se dan en la corriente. Representa los cambios más importantes para el perfil de carga debido a que los voltajes se mantienen relativamente estables. En la gráfica de corrientes se presentan los datos correspondientes a cada fase en el periodo de tiempo seleccionado en la consulta. La curva de corriente de neutro también se representa en la gráfica. La gráfica de corrientes muestra cómo aumentan o disminuyen el consumo de energía en función de la carga. Además muestra el desbalance de carga que existe a haber

diferencias considerables en las magnitudes de las corrientes, y un valor importante en la corriente de neutro.

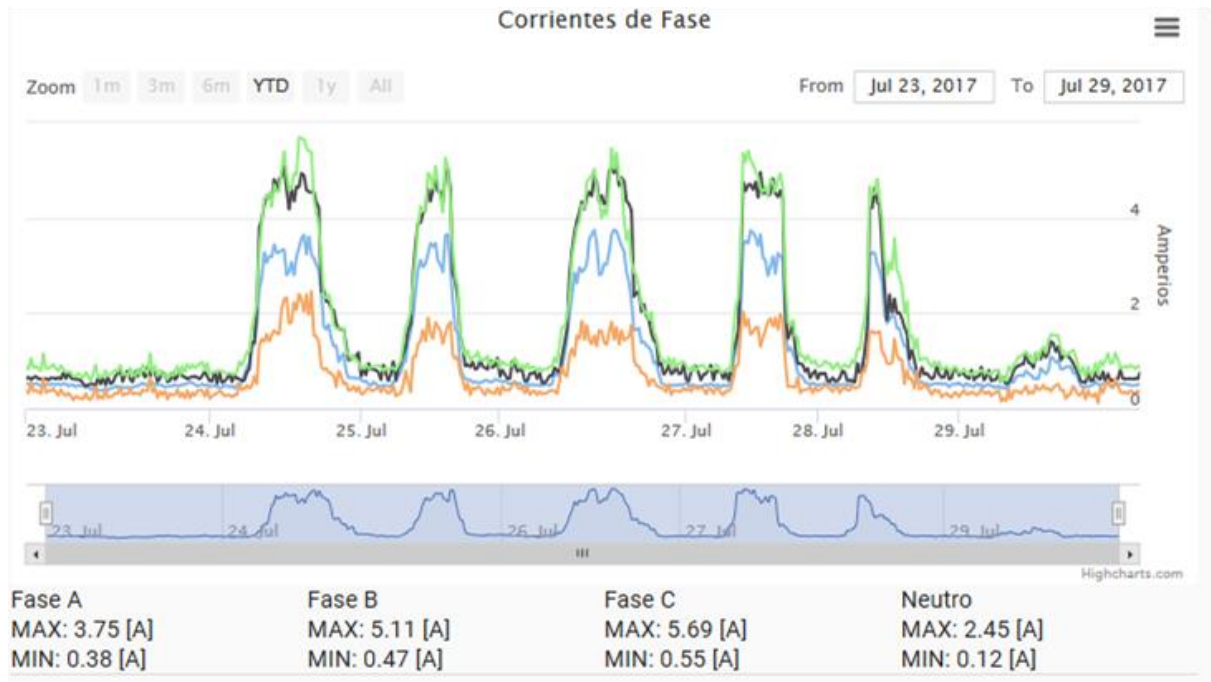


FIGURA 60. DATOS HISTÓRICOS DE CORRIENTE

Las últimas graficas son las de potencia activa en la FIGURA 61, potencia reactiva en la FIGURA 62 y potencia aparente en la FIGURA 63. Todas las gráficas presentan valores por fase. Cada gráfica compara los registros por fase de cada uno de estos parámetros. En la gráfica 53 de potencia se comparaban los valores trifásicos totales de potencia. Por lo que no se podía identificar qué fase demanda más potencia. Ahora con los gráficos de potencia por fase se puede hacer un análisis más detallado del comportamiento de la carga que es alimentada por la subestación. Es posible identificar que fase demanda mayor potencia activa y cual demanda mayor potencia reactiva..

Los valores de potencia activa, reactiva y aparente se encuentran en unida de kW, kVAR y kVA respectivamente, cada gráfica también presenta los valores máximos y mínimos de

cada una de las fases para identificar fácilmente aquellos valores que usualmente son de mayor interés.

Todos los gráficos poseen desplazamiento dinámico en el tiempo, se puede ajustar la escala que se desea observar. Los detalles de cada trazo se muestran al ubicar el puntero sobre las gráficas y se muestra en forma dinámica los valores correspondientes a las mediciones.

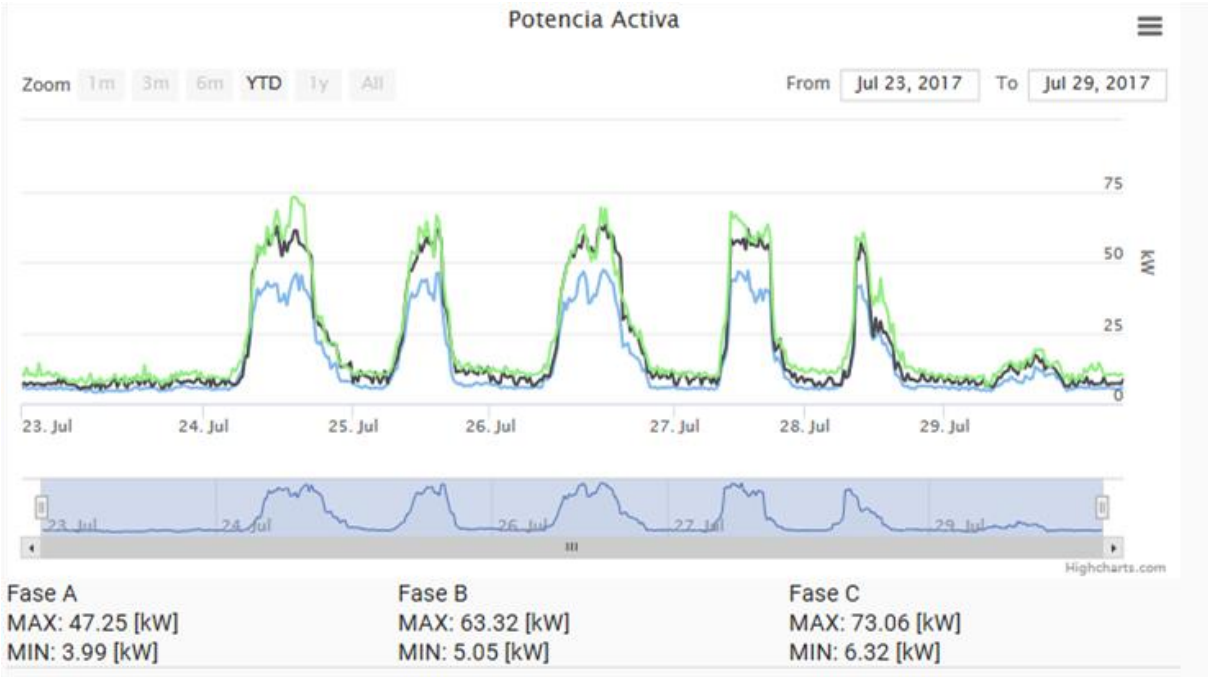


FIGURA 61. DATOS HISTÓRICOS DE POTENCIA ACTIVA POR FASE

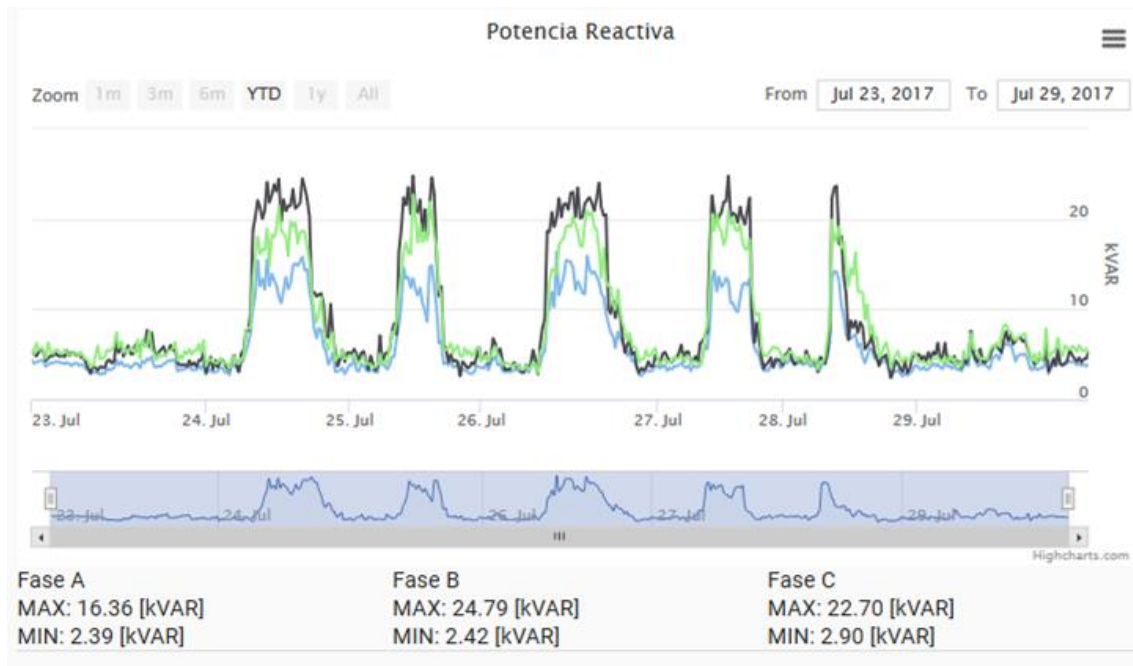


FIGURA 62. DATOS HISTÓRICOS DE POTENCIA REACTIVA POR FASE

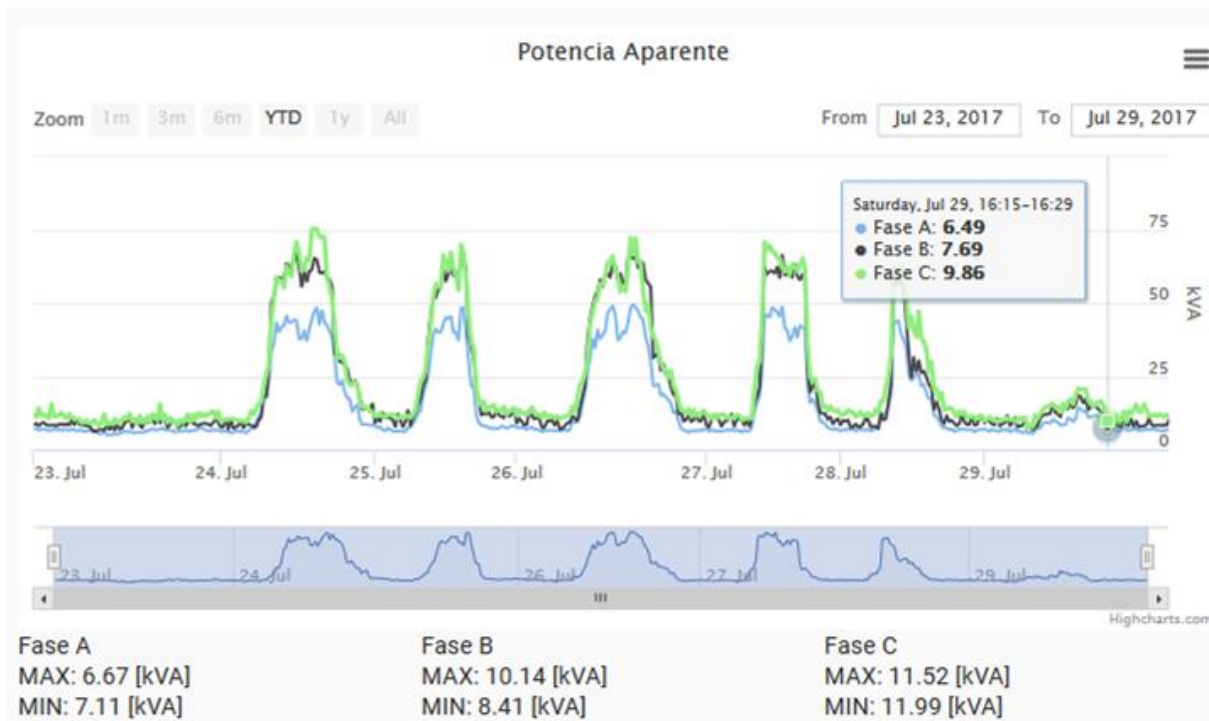


FIGURA 63. DATOS HISTÓRICOS DE POTENCIA APARENTE POR FASE

En la FIGURA 63 se puede ver como al pasar el puntero sobre la gráfica esta muestra los valores correspondientes de cada Fase, así como el momento en que se efectuó la medi-

ción, debido a la cantidad de datos la gráfica visible es una gráfica compacta donde un tramo representa un periodo de tiempo, esto se hace más preciso al ajustar el control de tiempo que se encuentra en cada una de las gráficas de este tipo.

#### 4.1.5 Facturación

La tercera pestaña de la aplicación es un complemento que permite estimar el costo de la energía eléctrica consumida, utiliza el pliego tarifario vigente para realizar los cálculos de costos y los presenta de forma detallada y clara a cuánto ascienden los costos en el periodo de tiempo en el que se está realizando la consulta.

El primer apartado consisten en los datos de facturación, se presenta el detalle del costo por horario, es decir el costo incurrido por consumo en horario punta, valle y resto, otro elemento de facturación importante corresponde al costo por comercialización , este depende de la máxima potencia demandada durante el periodo de facturación y por ultimo un cobro fijo por comercialización. Los detalles se presentan en la FIGURA 64.

RECIENTES	HISTORIAL DE MEDICIONES	FACTURA
Facturación		
Costo por comercialización		USD\$13.03
Costo por distribución		USD\$274.50
Costo por energía en punta		USD\$0.06
Costo por energía en valle		USD\$0.24
Costo por energía en resto		USD\$0.47
Subtotal		USD\$288.29
IVA (13%)		USD\$37.48
Total		USD\$325.77

FIGURA 64. FACTURACIÓN DEL PERIODO SELECCIONADO

El bloque de facturación también incluye el detalle del pliego aplicado, esta información se ingresa en la aplicación en la sección de configuración, se debe establecer el periodo de vigencia y los montos aplicados, para la aplicación web posteriormente a presentar la información realice los cálculos con el pliego correspondientes. Los detalles del pliego tarifario se presentan luego de la facturación para poder visualizar cuales han sido los pliegos utilizados para los cálculos del monto final y si existiera un periodo que implicara 2 pliegos tarifarios se hace la operación de forma separada para obtener los valores de cada parte del periodo según el pliego vigente. La FIGURA 65 ilustra como se muestra la información la unidades acompañan a las etiquetas en la primera columna, en la segunda columna se presentan los valores numéricos que describen el valor, así como también se presentan inicio y fin del periodo de validez pliego, y el nombre de la empresa distribuidora, todos estos datos son obligatorio para ingresar un pliego tarifario en la aplicación.

Pliegos aplicados	
Distribuidora	CAESS
Fecha de inicio	2017-04-15
Fecha de finalizacion	2017-07-14
Cargo en punta [USD\$/kWh]	0.130238
Cargo en valle [USD\$/kWh]	0.128332
Cargo en resto [USD\$/kWh]	0.130373
Cargo por distribucion [USD\$/kW-mes]	6.280334
Cargo por comercializacion [USD\$/Usuario-mes]	13.027817
Pérdidas de transformación [%]	0.015
Recargo por factor de potencia [%]	0

FIGURA 65. PLIEGO TARIFARIO

El último elemento de la aplicación en la sección de facturación corresponde a los detalles de las mediciones, estos datos son con lo que se realizan los cálculos para obtener el valor monetario final. Los datos se muestran en la FIGURA 66 bajo la etiqueta Lecturas. La pri-

mera columna se presenta los detalles de las mediciones, y en la derecha los valores numéricos.

Lecturas	
Maxima demanda [kW]	43.06
Consumo de energia en punta [kWh]	0.42
Consumo de energia en valle [kWh]	1.81
Consumo de energia en resto [kWh]	3.59

FIGURA 66. DETALLE DEL CONSUMO DE ENERGÍA

#### 4.1.6 Ajustes

Django provee de herramientas para el ingreso de datos en la base de datos, es aquí donde podemos sacar nuevamente ventaja del framework, este provee herramientas para ingreso de valores en la base de datos. En la parte inferior de la aplicación principal se encuentran los enlaces para complementos como los las opciones de configuración o settings y un breve manual del uso de la aplicación.

Al hacer clic en enlace settings, la aplicación nos direcciona a una página de validación de usuario y contraseña por seguridad, los campos a los que se tendrá acceso son manipulables y contienen información que afecta el desempeño de la aplicación, por lo que se debe proceder con cuidado.



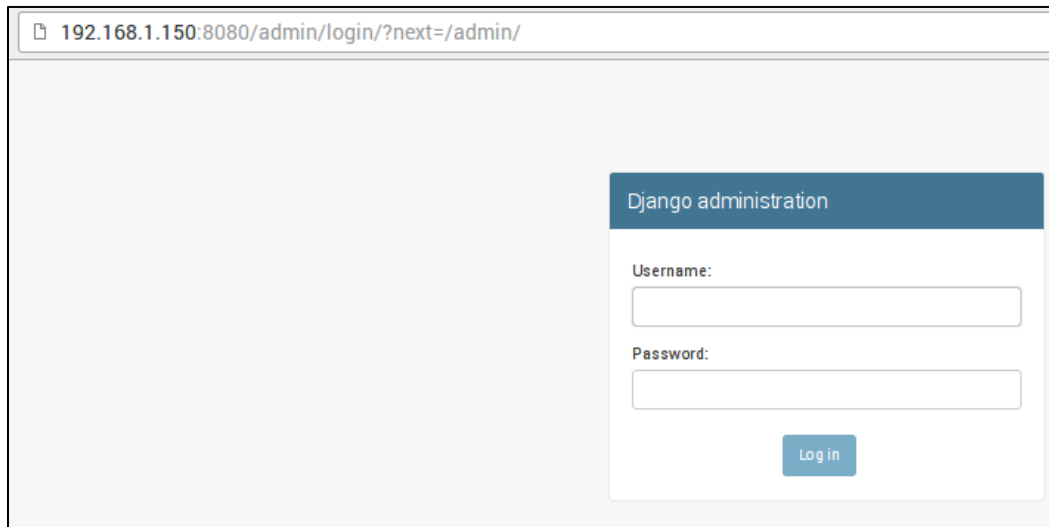


FIGURA 67. INGRESANDO A LAS CONFIGURACIONES

Cuando accedemos a la página administrativa de la aplicación se presenta una pantalla como la mostrada en la FIGURA 67. Muchas de las opciones vienen por defecto en Django como los son AUTH TOKEN, este complemento son métodos de seguridad que sirven para tener restringidas las sesiones y el envío de información, AUTHENTICATION AND AUTHORIZATION es el apartado para la creación de usuarios y asignación de estos permisos a los usuarios.

El nombre del proyecto en Django es dashboard, por lo que Django puede generar los menús para la modificación a través de esta interfaz, solo se debe incluir los campos que se desean manipular en el archivo de la aplicación admin.py.

Otro elemento de la página de administración es el historial de acciones recientes, aquí se detallan los registros que han sido realizados y los muestra de forma de listado en la sección Recent actions.

Dentro de los campos de interés tenemos los relacionados a los modelos que se crearon, al inicio del desarrollo, estos modelos son Measures, Nodes, Pliegos

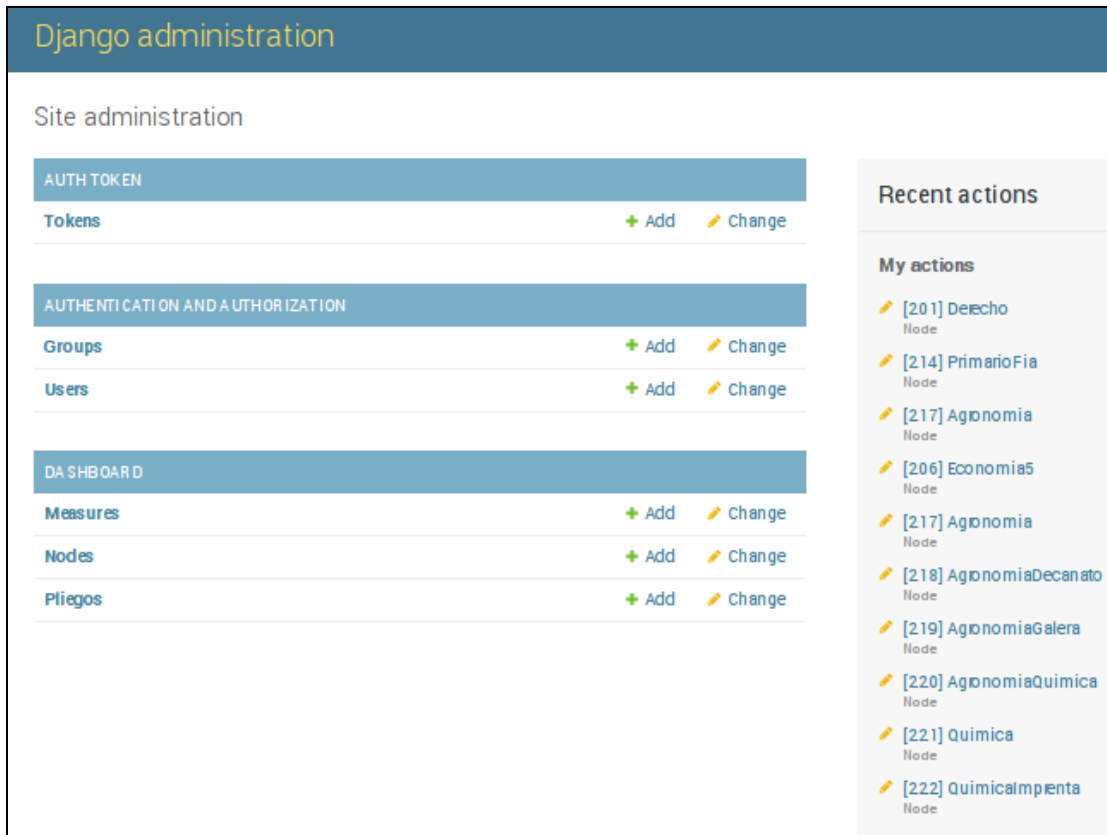


FIGURA 68. PÁGINA ADMINISTRATIVA DE LA APLICACIÓN WEB

Dentro de los apartados de dashboard podemos ver un listado con los datos ingresados en la base de datos, a hacer clic en los vínculos de los modelos, o también podemos agregar un nuevo registro con la opciones Add , así como modificar algún elemento en específico si es requerido, solo con seleccionar la opción edit, ambos vínculos se encuentran a la derecha de cada modelo.

El listado proporcionado por Measure se muestra en la FIGURA 69, el listado presentado esta ordenado por el id de cada registro, este es asignado de forma automática por Django al momento de interactuar con la base de datos. Los datos presentados en la FIGURA 70 son una parte de todos los campos que utiliza cada registro.

Django administration

Home > Dashboard > Measures

Select measure to change

Action:   0 of 100 selected

<input type="checkbox"/>	ID	DATEIMESTR	VAS 3 PH TOTAL	VARS 3 PH TOT
<input type="checkbox"/>	102119	-	2537.310303	973.191406
<input type="checkbox"/>	102118	-	4362.442871	1683.89062
<input type="checkbox"/>	102117	-	4577.508301	1403.85156
<input type="checkbox"/>	102116	-	4861.834473	1600.05468
<input type="checkbox"/>	102115	-	4700.77002	1506.46875
<input type="checkbox"/>	102114	-	4930.598633	1435.28125
<input type="checkbox"/>	102113	-	4238.320801	1558.66406

FIGURA 69. DATOS EN MEASURES

Los datos referentes a los nodos se presentan en la FIGURA 70, estos datos son los que permiten generar las visualizaciones en la aplicación contienen los detalles de cada nodo. Se puede acceder y modificar si es necesario así como adicionar nuevos nodos si es requerido.

Django administration

Home > Dashboard > Nodes

Select node to change

Action:   0 of 34 selected

<input type="checkbox"/>	ID	PARENT	NAME	SUPPLIER	DESCRIPTION
<input type="checkbox"/>	231	-	Nodo repetidor	-	Nodo repetido de humanidades, ubicacion no especificada
<input type="checkbox"/>	230	-	Artes	-	Se ubica en la esquina sureste de el edificio de Artes
<input type="checkbox"/>	229	-	Rectoria	-	Ubicado en la parte superior de las gradas entre el edificio de Artes y la libreria central
<input type="checkbox"/>	228	-	Psicología	-	Medidor ubicado atras del edificio de Psicología en la zona de verde. La subestacion compuesta por 3 transformadores monofasicos de 50kVA cada uno.
<input type="checkbox"/>	227	-	Medicina	-	Medidor ubicado en la Facultad de derecho cerca del nuevo edificio de Medicina
<input type="checkbox"/>	226	-	Odontología3	-	Medidor ubicado en la Facultad de Odontología, consta de tres transformadores de 50 kVA cada uno, conectados en configuración estrella - estrella montados en una estructura H.

FIGURA 70. DATOS EN NODES

Django administration

Home › Dashboard › Pliegos

Select pliego to change

Action:   0 of 1 selected

<input type="checkbox"/>	ID	SUPPLIER	VALID FROM	VALID TO	CARGO PUNTA	CARGO VALLE	CARGO RESTO	CARGO DISTRIBUCION
<input type="checkbox"/>	2	CAESS	April 15, 2017	July 14, 2017	0.130238	0.128332	0.130373	6.280334

FIGURA 71. DATOS EN PLIEGO

Como se mencionó anteriormente el pliego tarifario se debe adicionar para que se realicen los cálculos de costos, y se debe llenar cada uno de los campos para poder guardar un nuevo pliego tarifario, esto evita que se produzcan errores a la hora de proceder con las operaciones de cálculo.

Como se ha presentado en este capítulo, la presentación de los datos en forma gráfica permite constatar de forma sencilla la operación de las subestaciones. El capítulo siguiente presentara las conclusiones respecto al trabajo realizado, así como también las líneas futuras que debería seguir el sistema para seguir mejorándolo.

# CAPITULO V: CONCLUSIONES Y LÍNEAS FUTURAS

## 5.1 Conclusiones.

- La ampliación de variables eléctricas consultadas resultó exitosa, ahora es posible monitorear de forma más precisa las subestaciones incluidas en la red de monitoreo de la Universidad de El Salvador, además de permitir la consulta remota de variables eléctricas adicionales si es requerido
- La herramienta de monitoreo de enlaces inalámbricos es efectiva y descriptiva permite describir los tramos débiles de la red mesh y sirve como referencia para el mejoramiento de la red.
- El aumento de intentos de envío, reduciendo la cantidad de datos por envío ha dado buenos resultados, al aprovechar los lapsos de tiempo en lo que se puede establecer comunicación con el servidor.
- La aplicación web tuvo los resultados esperados, ahora es posible estudiar el comportamiento de las subestaciones utilizando las gráficas que proporciona una herramienta interactiva de fácil comprensión y de mejor precisión.
- La mejora de la red mesh se ha vuelto indispensable para la mejora del proyecto, las técnicas empleadas para paliar esta deficiencia como la interrogación in situ han sido exitosas, pero es necesario lograr mantener los enlaces el mayor tiempo posible para que el sistema pueda proveer datos en tiempo real.

## 5.2 Líneas futuras

- Una mejora sustancial es la adición de nuevos nodos o la reubicación de nodos, especialmente aquellos que están desconectados de la red mesh durante periodos considerables de tiempo.
- Las herramientas desarrolladas puede servir para estudios de calidad de energía en la Universidad de El Salvador hacer uso eficiente del recurso energético, y también para realizar procedimiento de mantenimiento más precisos, balance de cargas en las subestaciones que lo requieran
- Existen muchas subestaciones en el campus universitario que todavía no están agregadas al sistema actual de monitoreo, por lo que adicionarlas proveerá de información más completa de toda la infraestructura universitaria.

## **ANEXOS**

---

## **ANEXO A-1. MAPA DE MEMORIA SHARK 100S**

### **Mapa ModBus del Sub-medidor Shark® 100-S**

El mapa de Modbus para el medidor Shark ® 100-S proporciona detalles e información acerca de las posibles lecturas del medidor y su programación. El medidor Shark ® 100-S puede ser programado con los botones en la caratula del medidor, o mediante el uso de software. Para una visión general de programación, consulte la sección 5.2 de este manual.

#### Secciones de Mapa de Registro ModBus

El Mapa de registro ModBus del medidor Shark®100-S incluye las siguientes secciones:

Sección de Datos Fijos, registros del 1 al 47, los detalles de información fija del medidor,

Sección datos del Medidor, Registros del 1000 al 5003, detalles de las lecturas del medidor, incluyendo lecturas Primaria, Bloque de Energía, Demanda de Bloque, Bloque Angulo de Fase, Bloque de Estado, Bloque THD, Mínimos y Máximos en Regular y Bloques de Estampado de Tiempo, Bloques Opción de Tarjeta y Acumuladores . Modo de funcionamiento

Sección Comandos, Registros del 20000 al 26011, detalles del Medidor, Bloque de Restablecimiento, Bloque de programación, Otro bloque de comandos y Cifrado en bloque.

Sección de Ajustes programables, Registros del 30000 al 30067, todos los detalles de los ajustes se pueden programar para configurar su medidor. Sección Lecturas Secundaria.

Sección Lecturas Secundaria, Registros del 40001 al 40100, detalles del medidor, Lecturas secundarias.



## Formato de Datos

<b>ASCII:</b>	Caracteres ASCII empaquetados 2 por registrarse en alto, bajo orden y Sin ningún carácter de terminación. Ejemplo: "Shark 100" sería 4 registros que contienen 0x5378, 0x6172, 0x6B31, 0x3030.
<b>SINT16/UINT16:</b>	16-bits con signo / sin signo entero
<b>SINT32/UINT32:</b>	32-bits con signo / sin signo entero, que abarca 2 registros. El registro más bajo su dirección es el medio de alto orden
<b>FLOAT:</b>	32-bit IEEE número punto flotante que abarca 2 registros. El registro más bajo su dirección es la media de orden. (es decir, contiene el Exponente) Superior (es decir, contiene el exponente).

TABLA 7. TIPOS DE DATOS DEL MAPA MODBUS

### Valores Punto Flotante.

Valores Punto flotante se representan en el siguiente formato:

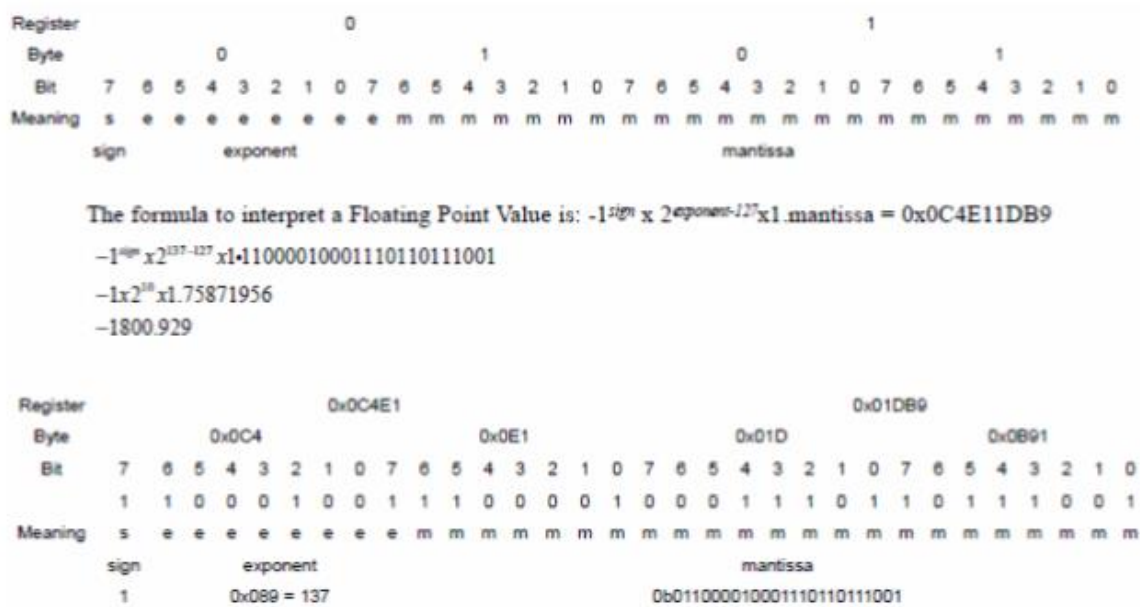


FIGURA 72. REPRESENTACIÓN DEL VALOR FLOTANTE

**Explicación de la Formula:**

C4E11DB9 (hex)      11000100 11100001 00011101 10111001 (binario)

El signo de la mantisa (y por tanto el número) es 1, lo que representa un valor negativo.

El exponente es 10001001 (binario) o 137 decimal.

El exponente es un valor superior a 127. Así que, el valor del exponente es 10.

La Mantisa es 11000010001110110111001 binario.

Con el 1 principal implicado, la mantisa es (1) 0.611 DB9 (hex.)

La representación del Punto Flotante es por lo tanto -1.75871956 veces 2 a la 10.

Equivalente Decimal: -1800.929

**NOTAS:**

Exponente = El número total antes del punto decimal.

Mantisa = La fracción positiva después del punto decimal.

MAPA MODBUS (MM-1 hasta MM-9)

El mapa de registro Modbus del sub-medidor

Shark 100-S inicia en la página siguiente.

Modbus Address		Description <sup>1</sup>	Format	Range <sup>6</sup>	Units or Resolution	Comments	# Reg
Hex	Decimal						
<b>Fixed Data Section</b>							
Identification Block							read-only
0000 - 0007	1 - 8	Meter Name	ASCII	16 char	none		8
0008 - 000F	9 - 16	Meter Serial Number	ASCII	16 char	none		8
0010 - 0010	17 - 17	Meter Type	UINT16	bit-mapped	-----t-----vvv	t = transducer model (1=yes, 0=no), vvv = V-switch(1 to 4)	1
0011 - 0012	18 - 19	Firmware Version	ASCII	4 char	none		2
0013 - 0013	20 - 20	Map Version	UINT16	0 to 65535	none		1
0014 - 0014	21 - 21	Meter Configuration	UINT16	bit-mapped	-----f-----	fffff = calibration frequency (50 or 60)	1
0015 - 0015	22 - 22	ASIC Version	UINT16	0-65535	none		1
0016 - 0026	23 - 39	Reserved					17
0027 - 002E	40 - 47	Reserved					8
						Block Size	47
<b>Meter Data Section<sup>2</sup></b>							
Primary Readings Block, 6 cycles (IEEE Floating Point)							read-only
0383 - 0384	900 - 901	Watts, 3-Ph total	FLOAT	-9999 M to +9999 M	watts		2
0385 - 0386	902 - 903	VARs, 3-Ph total	FLOAT	-9999 M to +9999 M	VARs		2
0387 - 0388	904 - 905	VA's, 3-Ph total	FLOAT	-9999 M to +9999 M	VA's		2
						Block Size	6
Primary Readings Block, 60 cycles (IEEE Floating Point)							read-only
03E7 - 03E8	1000 - 1001	Volts A-N	FLOAT	0 to 9999 M	volts		2
03E9 - 03EA	1002 - 1003	Volts B-N	FLOAT	0 to 9999 M	volts		2
03EB - 03EC	1004 - 1005	Volts C-N	FLOAT	0 to 9999 M	volts		2
03ED - 03EE	1006 - 1007	Volts A-B	FLOAT	0 to 9999 M	volts		2
03EF - 03F0	1008 - 1009	Volts B-C	FLOAT	0 to 9999 M	volts		2
03F1 - 03F2	1010 - 1011	Volts C-A	FLOAT	0 to 9999 M	volts		2
03F3 - 03F4	1012 - 1013	Amps A	FLOAT	0 to 9999 M	amps		2
03F5 - 03F6	1014 - 1015	Amps B	FLOAT	0 to 9999 M	amps		2
03F7 - 03F8	1016 - 1017	Amps C	FLOAT	0 to 9999 M	amps		2
03F9 - 03FA	1018 - 1019	Watts, 3-Ph total	FLOAT	-9999 M to +9999 M	watts		2
03FB - 03FC	1020 - 1021	VARs, 3-Ph total	FLOAT	-9999 M to +9999 M	VARs		2
03FD - 03FE	1022 - 1023	VA's, 3-Ph total	FLOAT	-9999 M to +9999 M	VA's		2
03FF - 0400	1024 - 1025	Power Factor, 3-Ph total	FLOAT	-1.00 to +1.00	none		2
0401 - 0402	1026 - 1027	Frequency	FLOAT	0 to 65.00	Hz		2
0403 - 0404	1028 - 1029	Neutral Current	FLOAT	0 to 9999 M	amps		2
						Block Size	30
Primary Energy Block							read-only

Modbus Address		Description <sup>1</sup>	Format	Range <sup>6</sup>	Units or Resolution	Comments	# Reg
Hex	Decimal						
044B - 044C	1100 - 1101	W-hours, Received	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	* Wh received & delivered always have opposite signs	2
044D - 044E	1102 - 1103	W-hours, Delivered	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	* Wh received is positive for "view as load", delivered is positive for "view as generator"	2
044F - 0450	1104 - 1105	W-hours, Net	SINT32	-99999999 to 99999999	Wh per energy format		2
0451 - 0452	1106 - 1107	W-hours, Total	SINT32	0 to 99999999	Wh per energy format	* 5 to 8 digits	2
0453 - 0454	1108 - 1109	VAR-hours, Positive	SINT32	0 to 99999999	VARh per energy format		2
0455 - 0456	1110 - 1111	VAR-hours, Negative	SINT32	0 to -99999999	VARh per energy format	* decimal point implied, per energy format	2
0457 - 0458	1112 - 1113	VAR-hours, Net	SINT32	-99999999 to 99999999	VARh per energy format	* resolution of digit before decimal point = units, kilo, or mega, per energy format	2
0459 - 045A	1114 - 1115	VAR-hours, Total	SINT32	0 to 99999999	VARh per energy format		2
045B - 045C	1116 - 1117	VA-hours, Total	SINT32	0 to 99999999	VAh per energy format	* see note 10	2
Block Size							18
Primary Demand Block (IEEE Floating Point)							read-only
07CF - 07D0	2000 - 2001	Amps A, Average	FLOAT	0 to 9999 M	amps		2
07D1 - 07D2	2002 - 2003	Amps B, Average	FLOAT	0 to 9999 M	amps		2
07D3 - 07D4	2004 - 2005	Amps C, Average	FLOAT	0 to 9999 M	amps		2
07D5 - 07D6	2006 - 2007	Positive Watts, 3-Ph, Average	FLOAT	-9999 M to +9999 M	watts		2
07D7 - 07D8	2008 - 2009	Positive VARs, 3-Ph, Average	FLOAT	-9999 M to +9999 M	VARs		2
07D9 - 07DA	2010 - 2011	Negative Watts, 3-Ph, Average	FLOAT	-9999 M to +9999 M	watts		2
07DB - 07DC	2012 - 2013	Negative VARs, 3-Ph, Average	FLOAT	-9999 M to +9999 M	VARs		2
07DD - 07DE	2014 - 2015	VAs, 3-Ph, Average	FLOAT	-9999 M to +9999 M	VAs		2
07DF - 07E0	2016 - 2017	Positive PF, 3-Ph, Average	FLOAT	-1.00 to +1.00	none		2
07E1 - 07E2	2018 - 2019	Negative PF, 3-PF, Average	FLOAT	-1.00 to +1.00	none		2
Block Size							20
Primary Minimum Block (IEEE Floating Point)							read-only
0BB7 - 0BB8	3000 - 3001	Volts A-N, Minimum	FLOAT	0 to 9999 M	volts		2
0BB9 - 0BBA	3002 - 3003	Volts B-N, Minimum	FLOAT	0 to 9999 M	volts		2
0BBB - 0BBC	3004 - 3005	Volts C-N, Minimum	FLOAT	0 to 9999 M	volts		2
0BBD - 0BBE	3006 - 3007	Volts A-B, Minimum	FLOAT	0 to 9999 M	volts		2
0BBF - 0BC0	3008 - 3009	Volts B-C, Minimum	FLOAT	0 to 9999 M	volts		2
0BC1 - 0BC2	3010 - 3011	Volts C-A, Minimum	FLOAT	0 to 9999 M	volts		2
0BC3 - 0BC4	3012 - 3013	Amps A, Minimum Avg Demand	FLOAT	0 to 9999 M	amps		2
0BC5 - 0BC6	3014 - 3015	Amps B, Minimum Avg Demand	FLOAT	0 to 9999 M	amps		2
0BC7 - 0BC8	3016 - 3017	Amps C, Minimum Avg Demand	FLOAT	0 to 9999 M	amps		2
0BC9 - 0BCA	3018 - 3019	Positive Watts, 3-Ph, Minimum Avg Demand	FLOAT	0 to +9999 M	watts		2
0BCB - 0BCC	3020 - 3021	Positive VARs, 3-Ph, Minimum Avg Demand	FLOAT	0 to +9999 M	VARs		2
0BCD - 0BCE	3022 - 3023	Negative Watts, 3-Ph, Minimum Avg Demand	FLOAT	0 to +9999 M	watts		2
0BCF - 0BD0	3024 - 3025	Negative VARs, 3-Ph, Minimum Avg Demand	FLOAT	0 to +9999 M	VARs		2
0BD1 - 0BD2	3026 - 3027	VAs, 3-Ph, Minimum Avg Demand	FLOAT	-9999 M to +9999 M	VAs		2
0BD3 - 0BD4	3028 - 3029	Positive Power Factor, 3-Ph, Minimum Avg Demand	FLOAT	-1.00 to +1.00	none		2

Modbus Address		Description <sup>1</sup>	Format	Range <sup>6</sup>	Units or Resolution	Comments	# Reg
Hex	Decimal						
0BD5 - 0BD6	3030 - 3031	Negative Power Factor, 3-Ph, Minimum Avg Demand	FLOAT	-1.00 to +1.00	none		2
0BD7 - 0BD8	3032 - 3033	Frequency, Minimum	FLOAT	0 to 65.00	Hz		2
						Block Size	34
Primary Maximum Block (IEEE Floating Point)							read-only
0C1B - 0C1C	3100 - 3101	Volts A-N, Maximum	FLOAT	0 to 9999 M	volts		2
0C1D - 0C1E	3102 - 3103	Volts B-N, Maximum	FLOAT	0 to 9999 M	volts		2
0C1F - 0C20	3104 - 3105	Volts C-N, Maximum	FLOAT	0 to 9999 M	volts		2
0C21 - 0C22	3106 - 3107	Volts A-B, Maximum	FLOAT	0 to 9999 M	volts		2
0C23 - 0C24	3108 - 3109	Volts B-C, Maximum	FLOAT	0 to 9999 M	volts		2
0C25 - 0C26	3110 - 3111	Volts C-A, Maximum	FLOAT	0 to 9999 M	volts		2
0C27 - 0C28	3112 - 3113	Amps A, Maximum Avg Demand	FLOAT	0 to 9999 M	amps		2
0C29 - 0C2A	3114 - 3115	Amps B, Maximum Avg Demand	FLOAT	0 to 9999 M	amps		2
0C2B - 0C2C	3116 - 3117	Amps C, Maximum Avg Demand	FLOAT	0 to 9999 M	amps		2
0C2D - 0C2E	3118 - 3119	Positive Watts, 3-Ph, Maximum Avg Demand	FLOAT	0 to +9999 M	watts		2
0C2F - 0C30	3120 - 3121	Positive VARs, 3-Ph, Maximum Avg Demand	FLOAT	0 to +9999 M	VARs		2
0C31 - 0C32	3122 - 3123	Negative Watts, 3-Ph, Maximum Avg Demand	FLOAT	0 to +9999 M	watts		2
0C33 - 0C34	3124 - 3125	Negative VARs, 3-Ph, Maximum Avg Demand	FLOAT	0 to +9999 M	VARs		2
0C35 - 0C36	3126 - 3127	VAs, 3-Ph, Maximum Avg Demand	FLOAT	-9999 M to +9999 M	VAs		2
0C37 - 0C38	3128 - 3129	Positive Power Factor, 3-Ph, Maximum Avg Demand	FLOAT	-1.00 to +1.00	none		2
0C39 - 0C3A	3130 - 3131	Negative Power Factor, 3-Ph, Maximum Avg Demand	FLOAT	-1.00 to +1.00	none		2
0C3B - 0C3C	3132 - 3133	Frequency, Maximum	FLOAT	0 to 65.00	Hz		2
						Block Size	34
THD Block <sup>7, 18</sup>							read-only
0F9F - 0F9F	4000 - 4000	Volts A-N, %THD	UINT16	0 to 9999, or 65535	0.1%		1
0FA0 - 0FA0	4001 - 4001	Volts B-N, %THD	UINT16	0 to 9999, or 65535	0.1%		1
0FA1 - 0FA1	4002 - 4002	Volts C-N, %THD	UINT16	0 to 9999, or 65535	0.1%		1
0FA2 - 0FA2	4003 - 4003	Amps A, %THD	UINT16	0 to 9999, or 65535	0.1%		1
0FA3 - 0FA3	4004 - 4004	Amps B, %THD	UINT16	0 to 9999, or 65535	0.1%		1
0FA4 - 0FA4	4005 - 4005	Amps C, %THD	UINT16	0 to 9999, or 65535	0.1%		1
0FA5 - 0FA5	4006 - 4006	Phase A Current 0th harmonic magnitude	UINT16	0 to 65535	none		1
0FA6 - 0FA6	4007 - 4007	Phase A Current 1st harmonic magnitude	UINT16	0 to 65535	none		1
0FA7 - 0FA7	4008 - 4008	Phase A Current 2nd harmonic magnitude	UINT16	0 to 65535	none		1
0FA8 - 0FA8	4009 - 4009	Phase A Current 3rd harmonic magnitude	UINT16	0 to 65535	none		1
0FA9 - 0FA9	4010 - 4010	Phase A Current 4th harmonic magnitude	UINT16	0 to 65535	none		1
0FAA - 0FAA	4011 - 4011	Phase A Current 5th harmonic magnitude	UINT16	0 to 65535	none		1
0FAB - 0FAB	4012 - 4012	Phase A Current 6th harmonic magnitude	UINT16	0 to 65535	none		1
0FAC - 0FAC	4013 - 4013	Phase A Current 7th harmonic magnitude	UINT16	0 to 65535	none		1
0FAD - 0FAD	4014 - 4014	Phase A Voltage 0th harmonic magnitude	UINT16	0 to 65535	none		1
0FAE - 0FAE	4015 - 4015	Phase A Voltage 1st harmonic magnitude	UINT16	0 to 65535	none		1

Modbus Address		Description <sup>1</sup>	Format	Range <sup>6</sup>	Units or Resolution	Comments	# Req
Hex	Decimal						
0FAF - 0FAF	4016 - 4016	Phase A Voltage 2nd harmonic magnitude	UINT16	0 to 65535	none		
0FB0 - 0FB0	4017 - 4017	Phase A Voltage 3rd harmonic magnitude	UINT16	0 to 65535	none		
0FB1 - 0FB8	4018 - 4025	Phase B Current harmonic magnitude			same as Phase A Current 0th to 7th harmonic magnitudes		
0FB9 - 0FBC	4026 - 4029	Phase B Voltage harmonic magnitude			same as Phase A Voltage 0th to 3rd harmonic magnitudes		
0FBD - 0FC4	4030 - 4037	Phase C Current harmonic magnitude			same as Phase A Current 0th to 7th harmonic magnitudes		
0FC5 - 0FC8	4038 - 4041	Phase C Voltage harmonic magnitude			same as Phase A Voltage 0th to 3rd harmonic magnitudes		
						Block Size	4;
Phase Angle Block <sup>4</sup>						read-only	
1003 - 1003	4100 - 4100	Phase A Current	SINT16	-1800 to +1800	0.1 degree		
1004 - 1004	4101 - 4101	Phase B Current	SINT16	-1800 to +1800	0.1 degree		
1005 - 1005	4102 - 4102	Phase C Current	SINT16	-1800 to +1800	0.1 degree		
1006 - 1006	4103 - 4103	Angle, Volts A-B	SINT16	-1800 to +1800	0.1 degree		
1007 - 1007	4104 - 4104	Angle, Volts B-C	SINT16	-1800 to +1800	0.1 degree		
1008 - 1008	4105 - 4105	Angle, Volts C-A	SINT16	-1800 to +1800	0.1 degree		
						Block Size	
Status Block						read-only	
1387 - 1387	5000 - 5000	Meter Status	UINT16	bit-mapped	--expnch ssssssss	expnch = EEPROM block OK flags (e=energy, x=max, n=min, p=programmable settings, c=calibration, h=header), ssssssss = state (1=Run, 2=Limp, 10=Prog Set Update via buttons, 11=Prog Set Update via IrDA, 12=Prog Set Update via COM2)	
1388 - 1388	5001 - 5001	Limits Status <sup>7</sup>	UINT16	bit-mapped	87654321 87654321	high byte is setpt 1, 0=min, 1=out low byte is setpt 2, 0=min, 1=out	
1389 - 138A	5002 - 5003	Time Since Reset	UINT32	0 to 4294967294	4 msec	wraps around after max count	
						Block Size	
<b>Commands Section <sup>4</sup></b>							
Resets Block <sup>8</sup>						write-only	
4E1F - 4E1F	20000 - 20000	Reset Max/Min Blocks	UINT16	password <sup>5</sup>			
4E20 - 4E20	20001 - 20001	Reset Energy Accumulators	UINT16	password <sup>5</sup>			
						Block Size	
Meter Programming Block						read/conditional write	
55EF - 55EF	22000 - 22000	Initiate Programmable Settings Update	UINT16	password <sup>5</sup>		meter enters PS update mode	
55F0 - 55F0	22001 - 22001	Terminate Programmable Settings Update	UINT16	any value		meter leaves PS update mode via reset	

## **ANEXO A-2 MAPA DE MEMORIA SHARK 200S**

El mapa de Modbus para el medidor Shark<sup>®</sup> 200 proporciona detalles e información acerca de las posibles lecturas del medidor y su programación. El medidor Shark<sup>®</sup> 200 se puede programar con los botones en la caratula del medidor , o mediante el uso de software. Para una visión general de programación, consulte la sección 5.2 de este manual.

El Mapa de registro ModBus del medidor Shark<sup>®</sup> 200 incluye las siguientes secciones:

Sección de Datos Fijos, registros del 1 al 47, los detalles de información fija del medidor.

Sección datos del Medidor, Registros del 1000 al 12031, detalles de las lecturas del medidor, incluyendo lecturas Primaria, Bloque de Energía, Demanda de Bloque, Bloque Angulo de Fase, Bloque de Estado, Bloque THD, Mínimos y Máximos en Regular y Bloques de Estampado de Tiempo, Bloques Opción de Tarjeta y Acumuladores .

Sección Comandos, Registros del 20000 al 26011, detalles del Medidor, Bloque de Restablecimiento, Bloque de programación, Otro bloque de comandos y Cifrado en bloque.

Sección de Ajustes programables, Registros del 30000 al 33575, todos los detalles de los ajustes se pueden programar para configurar su medidor. Sección Lecturas Secundaria.

Sección Lecturas Secundaria, Registros del 40001 al 40100, detalles del medidor, Lecturas secundarias.

Sección Recuperación de Registros, Registros del 49997 al 51095, los detalles de recuperación de registros.

Modbus Address		Description (Note 1)	Format	Range (Note 6)	Units or Resolution	Comments	# Reg
Hex	Decimal						
<b>Fixed Data Section</b>							
<b>Identification Block</b>							<b>read-only</b>
0000	- 0007	1 - 8 Meter Name	ASCII	16 char	none		8
0008	- 000F	9 - 16 Meter Serial Number	ASCII	16 char	none		8
0010	- 0010	17 - 17 Meter Type	UINT16	bit mapped	---st---vw	t = 0 s = 1 vw = V switch V33 = standard 200S	1
0011	- 0012	18 - 19 Firmware Version	ASCII	4 char	none		2
0013	- 0013	20 - 20 Map Version	UINT16	0 to 65535	none		1
0014	- 0014	21 - 21 Meter Configuration	UINT16	bit mapped	---ccc---#fff	ccc = CT denominator (1 or 5), #fff = calibration frequency (50 or 60)	1
0015	- 0015	22 - 22 ASIC Version	UINT16	0-65535	none		1
0016	- 0017	23 - 24 Boot Firmware Version	ASCII	4 char	none		2
0018	- 0018	25 - 25 Reserved					1
0019	- 0019	26 - 26 Reserved					1
001A	- 001D	27 - 30 Meter Type Name	ASCII	8 char	none		4
001E	- 0026	31 - 36 Reserved				Reserved	6
0027	- 002E	40 - 47 Reserved				Reserved	8
002F	- 0115	48 - 278 Reserved				Reserved	231
0116	- 0130	279 - 305 Integer Readings Block occupies these registers, see below					
0131	- 01F3	306 - 500 Reserved					194
01F4	- 0203	501 - 516 Reserved				Reserved	16
<b>Meter Data Section (Note 2)</b>							
<b>Readings Block ( Integer values)</b>							<b>read-only</b>
0116	- 0117	280 - 280 Volts B-N	UINT16	0 to 9999	volts		1
0117	- 0118	281 - 281 Volts C-N	UINT16	0 to 9999	volts		1
0118	- 0119	282 - 282 Volts A-B	UINT16	0 to 9999	volts		1
0119	- 011A	283 - 283 Volts B-C	UINT16	0 to 9999	volts		1
011A	- 011B	284 - 284 Volts C-A	UINT16	0 to 9999	volts		1
011B	- 011C	285 - 285 Amps A	UINT16	0 to 9999	amps		1
011C	- 011D	286 - 286 Amps B	UINT16	0 to 9999	amps		1
011D	- 011E	287 - 287 Amps C	UINT16	0 to 9999	amps		1
011E	- 011F	288 - 288 Neutral Current	UINT16	-9999 to +9999	amps		1
011F	- 0120	289 - 289 Watts, 3-Ph total	SINT16	-9999 to +9999	watts	1. Use the settings from Programmable settings for scale and decimal point location. (see User Settings Flags)	1
0120	- 0121	290 - 290 VARs, 3-Ph total	SINT16	-9999 to +9999	VARs		1
0121	- 0122	291 - 291 VA, 3-Ph total	UINT16	0 to +9999	VA	2. Per phase power and PF have values only for WYE hookup and will be zero for all other hookups.	1
0122	- 0123	292 - 292 Power Factor, 3-Ph total	SINT16	-1000 to +1000	none		1
0123	- 0124	293 - 293 Frequency	UINT16	0 to 9999	Hz		1
0124	- 0125	294 - 294 Watts, Phase A	SINT16	-9999 M to +9999	watts	3. If the reading is 10000 that means that the value is out of range. Please adjust the programmable settings in that case. The display will also show '----' in case of over range.	1
0125	- 0126	295 - 295 Watts, Phase B	SINT16	-9999 M to +9999	watts		1
0126	- 0127	296 - 296 Watts, Phase C	SINT16	-9999 M to +9999	watts		1
0127	- 0128	297 - 297 VARs, Phase A	SINT16	-9999 M to +9999 M	VARs		1
0128	- 0129	298 - 298 VARs, Phase B	SINT16	-9999 M to +9999 M	VARs		1
0129	- 012A	299 - 299 VARs, Phase C	SINT16	-9999 M to +9999 M	VARs		1
012A	- 012B	300 - 300 VA, Phase A	UINT16	0 to +9999	VA		1
012B	- 012C	301 - 301 VA, Phase B	UINT16	0 to +9999	VA		1
012C	- 012D	302 - 302 VA, Phase C	UINT16	0 to +9999	VA		1
012D	- 012E	303 - 303 Power Factor, Phase A	SINT16	-1000 to +1000	none		1



Modbus Address		Description (Note 1)	Format	Range (Note 6)	Units or Resolution	Comments	# Reg	
Hex	Decimal							
012E	- 012F	304 - 304	Power Factor, Phase B	SINT16	-1000 to +1000	none	1	
012F	- 0130	305 - 305	Power Factor, Phase C	SINT16	-1000 to +1000	none	1	
0130	- 0130	305 - 305	Power Factor, Phase C	SINT16	-1000 to +1000	none	1	
							Block Size:	27
<b>Primary Readings Block</b>						<b>read-only</b>		
03E7	- 03E8	1000 - 1001	Volts A-N	FLOAT	0 to 9999 M	volts	2	
03E9	- 03EA	1002 - 1003	Volts B-N	FLOAT	0 to 9999 M	volts	2	
03EB	- 03EC	1004 - 1005	Volts C-N	FLOAT	0 to 9999 M	volts	2	
03ED	- 03EE	1006 - 1007	Volts A-B	FLOAT	0 to 9999 M	volts	2	
03EF	- 03F0	1008 - 1009	Volts B-C	FLOAT	0 to 9999 M	volts	2	
03F1	- 03F2	1010 - 1011	Volts C-A	FLOAT	0 to 9999 M	volts	2	
03F3	- 03F4	1012 - 1013	Amps A	FLOAT	0 to 9999 M	amps	2	
03F5	- 03F6	1014 - 1015	Amps B	FLOAT	0 to 9999 M	amps	2	
03F7	- 03F8	1016 - 1017	Amps C	FLOAT	0 to 9999 M	amps	2	
03F9	- 03FA	1018 - 1019	Watts, 3-Ph total	FLOAT	-9999 M to +9999 M	watts	2	
03FB	- 03FC	1020 - 1021	VARs, 3-Ph total	FLOAT	-9999 M to +9999 M	VARs	2	
03FD	- 03FE	1022 - 1023	VA, 3-Ph total	FLOAT	-9999 M to +9999 M	VA	2	
03FF	- 0400	1024 - 1025	Power Factor, 3-Ph total	FLOAT	-1.00 to +1.00	none	2	
0401	- 0402	1026 - 1027	Frequency	FLOAT	0 to 65.00	Hz	2	
0403	- 0404	1028 - 1029	Neutral Current	FLOAT	0 to 9999 M	amps	2	
0405	- 0406	1030 - 1031	Watts, Phase A	FLOAT	-9999 M to +9999 M	watts	2	
0407	- 0408	1032 - 1033	Watts, Phase B	FLOAT	-9999 M to +9999 M	watts	2	
0409	- 040A	1034 - 1035	Watts, Phase C	FLOAT	-9999 M to +9999 M	watts	2	
040B	- 040C	1036 - 1037	VARs, Phase A	FLOAT	-9999 M to +9999 M	VARs	2	
040D	- 040E	1038 - 1039	VARs, Phase B	FLOAT	-9999 M to +9999 M	VARs	2	
040F	- 0410	1040 - 1041	VARs, Phase C	FLOAT	-9999 M to +9999 M	VARs	2	
0411	- 0412	1042 - 1043	VA, Phase A	FLOAT	-9999 M to +9999 M	VA	2	
0413	- 0414	1044 - 1045	VA, Phase B	FLOAT	-9999 M to +9999 M	VA	2	
0415	- 0416	1046 - 1047	VA, Phase C	FLOAT	-9999 M to +9999 M	VA	2	
0417	- 0418	1048 - 1049	Power Factor, Phase A	FLOAT	-1.00 to +1.00	none	2	
0419	- 041A	1050 - 1051	Power Factor, Phase B	FLOAT	-1.00 to +1.00	none	2	
041B	- 041C	1052 - 1053	Power Factor, Phase C	FLOAT	-1.00 to +1.00	none	2	
041D	- 041E	1054 - 1055	Symmetrical Component Magnitude, 0 Seq	FLOAT	0 to 9999 M	volts	2	
041F	- 0420	1056 - 1057	Symmetrical Component Magnitude, + Seq	FLOAT	0 to 9999 M	volts	2	
0421	- 0422	1058 - 1059	Symmetrical Component Magnitude, - Seq	FLOAT	0 to 9999 M	volts	2	
0423	- 0423	1060 - 1060	Symmetrical Component Phase, 0 Seq	SINT16	-1800 to +1800	0.1 degree	1	
0424	- 0424	1061 - 1061	Symmetrical Component Phase, + Seq	SINT16	-1800 to +1800	0.1 degree	1	
0425	- 0425	1062 - 1062	Symmetrical Component Phase, - Seq	SINT16	-1800 to +1800	0.1 degree	1	
0426	- 0426	1063 - 1063	Unbalance, 0 sequence component	UINT16	0 to 10000	0.01%	1	
0427	- 0427	1064 - 1064	Unbalance, -sequence component	UINT16	0 to 10000	0.01%	1	
0428	- 0428	1065 - 1065	Current Unbalance	UINT16	0 to 20000	0.01%	1	
							Block Size:	66

Per phase power and PF have values only for WYE hookup and will be zero for all other hookups.

Voltage unbalance per IEC6100-4-30

Values apply only to WYE hookup and will be zero for all other hookups.

Modbus Address		Description (Note 1)	Format	Range (Note 6)	Units or Resolution	Comments	# Reg	
Hex	Decimal							
<b>Primary Energy Block</b>								
<b>read-only</b>								
05DB	- 05DC	1500 - 1501	W-hours, Received	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	* Wh received & delivered always have opposite signs	2
05DD	- 05DE	1502 - 1503	W-hours, Delivered	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	* Wh received is positive for "view as load", delivered is positive for "view as generator"	2
05DF	- 05E0	1504 - 1505	W-hours, Net	SINT32	-99999999 to 99999999	Wh per energy format	* 5 to 8 digits	2
05E1	- 05E2	1506 - 1507	W-hours, Total	SINT32	0 to 99999999	Wh per energy format	* decimal point implied, per energy format	2
05E3	- 05E4	1508 - 1509	VAR-hours, Positive	SINT32	0 to 99999999	VARh per energy format		2
05E5	- 05E6	1510 - 1511	VAR-hours, Negative	SINT32	0 to -99999999	VARh per energy format	* resolution of digit before decimal point = units, kilo, or mega, per energy format	2
05E7	- 05E8	1512 - 1513	VAR-hours, Net	SINT32	-99999999 to 99999999	VARh per energy format		2
05E9	- 05EA	1514 - 1515	VAR-hours, Total	SINT32	0 to 99999999	VARh per energy format	* see note 10	2
05EB	- 05EC	1516 - 1517	VA-hours, Total	SINT32	0 to 99999999	VAh per energy format		2
05ED	- 05EE	1518 - 1519	W-hours, Received, Phase A	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		2
05EF	- 05F0	1520 - 1521	W-hours, Received, Phase B	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		2
05F1	- 05F2	1522 - 1523	W-hours, Received, Phase C	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		2
05F3	- 05F4	1524 - 1525	W-hours, Delivered, Phase A	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		2
05F5	- 05F6	1526 - 1527	W-hours, Delivered, Phase B	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		2
05F7	- 05F8	1528 - 1529	W-hours, Delivered, Phase C	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		2
05F9	- 05FA	1530 - 1531	W-hours, Net, Phase A	SINT32	-99999999 to 99999999	Wh per energy format		2
05FB	- 05FC	1532 - 1533	W-hours, Net, Phase B	SINT32	-99999999 to 99999999	Wh per energy format		2
05FD	- 05FE	1534 - 1535	W-hours, Net, Phase C	SINT32	-99999999 to 99999999	Wh per energy format		2
05FF	- 0600	1536 - 1537	W-hours, Total, Phase A	SINT32	0 to 99999999	Wh per energy format		2
0601	- 0602	1538 - 1539	W-hours, Total, Phase B	SINT32	0 to 99999999	Wh per energy format		2
0603	- 0604	1540 - 1541	W-hours, Total, Phase C	SINT32	0 to 99999999	Wh per energy format		2
0605	- 0606	1542 - 1543	VAR-hours, Positive, Phase A	SINT32	0 to 99999999	VARh per energy format		2
0607	- 0608	1544 - 1545	VAR-hours, Positive, Phase B	SINT32	0 to 99999999	VARh per energy format		2
0609	- 060A	1546 - 1547	VAR-hours, Positive, Phase C	SINT32	0 to 99999999	VARh per energy format		2
060B	- 060C	1548 - 1549	VAR-hours, Negative, Phase A	SINT32	0 to -99999999	VARh per energy format		2
060D	- 060E	1550 - 1551	VAR-hours, Negative, Phase B	SINT32	0 to -99999999	VARh per energy format		2
060F	- 0610	1552 - 1553	VAR-hours, Negative, Phase C	SINT32	0 to -99999999	VARh per energy format		2
0611	- 0612	1554 - 1555	VAR-hours, Net, Phase A	SINT32	-99999999 to 99999999	VARh per energy format		2
0613	- 0614	1556 - 1557	VAR-hours, Net, Phase B	SINT32	-99999999 to 99999999	VARh per energy format		2
0615	- 0616	1558 - 1559	VAR-hours, Net, Phase C	SINT32	-99999999 to 99999999	VARh per energy format		2
0617	- 0618	1560 - 1561	VAR-hours, Total, Phase A	SINT32	0 to 99999999	VARh per energy format		2
0619	- 061A	1562 - 1563	VAR-hours, Total, Phase B	SINT32	0 to 99999999	VARh per energy format		2
061B	- 061C	1564 - 1565	VAR-hours, Total, Phase C	SINT32	0 to 99999999	VARh per energy format		2
061D	- 061E	1566 - 1567	VA-hours, Phase A	SINT32	0 to 99999999	VAh per energy format		2
061F	- 0620	1568 - 1569	VA-hours, Phase B	SINT32	0 to 99999999	VAh per energy format		2
0621	- 0622	1570 - 1571	VA-hours, Phase C	SINT32	0 to 99999999	VAh per energy format		2
0623	- 0624	1572 - 1573	W-hours, Received, rollover count	UINT32	0 to 4,294,967,294		These registers count the number of times their corresponding energy accumulators have wrapped from +max to 0. They are read when energy is read.	
0625	- 0626	1574 - 1575	W-hours, Delivered, rollover count	UINT32	0 to 4,294,967,294			
0627	- 0628	1576 - 1577	VAR-hours, Positive, rollover count	UINT32	0 to 4,294,967,294			
0629	- 062A	1578 - 1579	VAR-hours, Negative, rollover count	UINT32	0 to 4,294,967,294			
062B	- 062C	1580 - 1581	VA-hours, rollover count	UINT32	0 to 4,294,967,294			

Modbus Address		Description (Note 1)	Format	Range (Note 6)	Units or Resolution	Comments	# Reg	
Hex	Decimal							
062D	- 062E	1582 - 1583	W-hours in the Interval, Received	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	* Wh received & delivered always have opposite signs	
062F	- 0630	1584 - 1585	W-hours in the Interval, Delivered	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	* Wh received is positive for "view as load", delivered is positive for "view as generator"	
0631	- 0632	1586 - 1587	VAR-hours in the Interval, Positive	SINT32	0 to 99999999	VARh per energy format		
0633	- 0634	1588 - 1589	VAR-hours in the Interval, Negative	SINT32	0 to -99999999	VARh per energy format	* 5 to 8 digits	
0635	- 0636	1590 - 1591	VA-hours in the Interval, Total	SINT32	0 to 99999999	VAh per energy format	* decimal point implied, per energy format	
0637	- 0638	1592 - 1593	W-hours in the Interval, Received, Phase A	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	* resolution of digit before decimal point = u	
0639	- 063A	1594 - 1595	W-hours in the Interval, Received, Phase B	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		
063B	- 063C	1596 - 1597	W-hours in the Interval, Received, Phase C	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		
063D	- 063E	1598 - 1599	W-hours in the Interval, Delivered, Phase A	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		
063F	- 0640	1600 - 1601	W-hours in the Interval, Delivered, Phase B	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		
0641	- 0642	1602 - 1603	W-hours in the Interval, Delivered, Phase C	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format		
0643	- 0644	1604 - 1605	VAR-hours in the Interval, Positive, Phase A	SINT32	0 to 99999999	VARh per energy format		
0645	- 0646	1606 - 1607	VAR-hours in the Interval, Positive, Phase B	SINT32	0 to 99999999	VARh per energy format		
0647	- 0648	1608 - 1609	VAR-hours in the Interval, Positive, Phase C	SINT32	0 to 99999999	VARh per energy format		
0649	- 064A	1610 - 1611	VAR-hours in the Interval, Negative, Phase A	SINT32	0 to -99999999	VARh per energy format		
064B	- 064C	1612 - 1613	VAR-hours in the Interval, Negative, Phase B	SINT32	0 to -99999999	VARh per energy format		
064D	- 064E	1614 - 1615	VAR-hours in the Interval, Negative, Phase C	SINT32	0 to -99999999	VARh per energy format		
064F	- 0650	1616 - 1617	VA-hours in the Interval, Phase A	SINT32	0 to 99999999	VAh per energy format		
0651	- 0652	1618 - 1619	VA-hours in the Interval, Phase B	SINT32	0 to 99999999	VAh per energy format		
0653	- 0654	1620 - 1621	VA-hours in the Interval, Phase C	SINT32	0 to 99999999	VAh per energy format		
							Block Size: 122	
<b>Primary Demand Block</b>							<b>read-only</b>	
07CF	- 07D0	2000 - 2001	Amps A, Average	FLOAT	0 to 9999 M	amps	2	
07D1	- 07D2	2002 - 2003	Amps B, Average	FLOAT	0 to 9999 M	amps	2	
07D3	- 07D4	2004 - 2005	Amps C, Average	FLOAT	0 to 9999 M	amps	2	
07D5	- 07D6	2006 - 2007	Positive Watts, 3-Ph, Average	FLOAT	-9999 M to +9999 M	watts	2	
07D7	- 07D8	2008 - 2009	Positive VARs, 3-Ph, Average	FLOAT	-9999 M to +9999 M	VARs	2	
07D9	- 07DA	2010 - 2011	Negative Watts, 3-Ph, Average	FLOAT	-9999 M to +9999 M	watts	2	
07DB	- 07DC	2012 - 2013	Negative VARs, 3-Ph, Average	FLOAT	-9999 M to +9999 M	VARs	2	
07DD	- 07DE	2014 - 2015	VA, 3-Ph, Average	FLOAT	-9999 M to +9999 M	VA	2	
07DF	- 07E0	2016 - 2017	Positive PF, 3-Ph, Average	FLOAT	-1.00 to +1.00	none	2	
07E1	- 07E2	2018 - 2019	Negative PF, 3-Ph, Average	FLOAT	-1.00 to +1.00	none	2	
07E3	- 07E4	2020 - 2021	Neutral Current, Average	FLOAT	0 to 9999 M	amps	2	
07E5	- 07E6	2022 - 2023	Positive Watts, Phase A, Average	FLOAT	-9999 M to +9999 M	watts	2	
07E7	- 07E8	2024 - 2025	Positive Watts, Phase B, Average	FLOAT	-9999 M to +9999 M	watts	2	
07E9	- 07EA	2026 - 2027	Positive Watts, Phase C, Average	FLOAT	-9999 M to +9999 M	watts	2	
07EB	- 07EC	2028 - 2029	Positive VARs, Phase A, Average	FLOAT	-9999 M to +9999 M	VARs	2	
07ED	- 07EE	2030 - 2031	Positive VARs, Phase B, Average	FLOAT	-9999 M to +9999 M	VARs	2	
07EF	- 07F0	2032 - 2033	Positive VARs, Phase C, Average	FLOAT	-9999 M to +9999 M	VARs	2	
07F1	- 07F2	2034 - 2035	Negative Watts, Phase A, Average	FLOAT	-9999 M to +9999 M	watts	2	
07F3	- 07F4	2036 - 2037	Negative Watts, Phase B, Average	FLOAT	-9999 M to +9999 M	watts	2	
07F5	- 07F6	2038 - 2039	Negative Watts, Phase C, Average	FLOAT	-9999 M to +9999 M	watts	2	

Modbus Address		Description (Note 1)	Format	Range (Note 5)	Units or Resolution	Comments	# Reg
Hex	Decimal						
07F7	- 07FB	2040 - 2041	Negative VARs, Phase A, Average	FLOAT	-9999 M to +9999 M	VARs	2
07F9	- 07FA	2042 - 2043	Negative VARs, Phase B, Average	FLOAT	-9999 M to +9999 M	VARs	2
07FB	- 07FC	2044 - 2045	Negative VARs, Phase C, Average	FLOAT	-9999 M to +9999 M	VARs	2
07FD	- 07FE	2046 - 2047	VA, Phase A, Average	FLOAT	-9999 M to +9999 M	VAs	2
07FF	- 0800	2048 - 2049	VA, Phase B, Average	FLOAT	-9999 M to +9999 M	VAs	2
0801	- 0802	2050 - 2051	VA, Phase C, Average	FLOAT	-9999 M to +9999 M	VAs	2
0803	- 0804	2052 - 2053	Positive PF, Phase A, Average	FLOAT	-1.00 to +1.00	none	2
0805	- 0806	2054 - 2055	Positive PF, Phase B, Average	FLOAT	-1.00 to +1.00	none	2
0807	- 0808	2056 - 2057	Positive PF, Phase C, Average	FLOAT	-1.00 to +1.00	none	2
0809	- 080A	2058 - 2059	Negative PF, Phase A, Average	FLOAT	-1.00 to +1.00	none	2
080B	- 080C	2060 - 2061	Negative PF, Phase B, Average	FLOAT	-1.00 to +1.00	none	2
080D	- 080E	2062 - 2063	Negative PF, Phase C, Average	FLOAT	-1.00 to +1.00	none	2
						Block Size:	64
<b>Uncompensated Readings Block</b>						read-only	
08B7	- 08B8	3000 - 3001	Watts, 3-Ph total	FLOAT	-9999 M to +9999 M	watts	2
08B9	- 08BA	3002 - 3003	VARs, 3-Ph total	FLOAT	-9999 M to +9999 M	VARs	2
08BB	- 08BC	3004 - 3005	VA, 3-Ph total	FLOAT	-9999 M to +9999 M	VAs	2
08BD	- 08BE	3006 - 3007	Power Factor, 3-Ph total	FLOAT	-1.00 to +1.00	none	2
08BF	- 08C0	3008 - 3009	Watts, Phase A	FLOAT	-9999 M to +9999 M	watts	2
08C1	- 08C2	3010 - 3011	Watts, Phase B	FLOAT	-9999 M to +9999 M	watts	2
08C3	- 08C4	3012 - 3013	Watts, Phase C	FLOAT	-9999 M to +9999 M	watts	2
08C5	- 08C6	3014 - 3015	VARs, Phase A	FLOAT	-9999 M to +9999 M	VARs	2
08C7	- 08C8	3016 - 3017	VARs, Phase B	FLOAT	-9999 M to +9999 M	VARs	2
08C9	- 08CA	3018 - 3019	VARs, Phase C	FLOAT	-9999 M to +9999 M	VARs	2
08CB	- 08CC	3020 - 3021	VA, Phase A	FLOAT	-9999 M to +9999 M	VAs	2
08CD	- 08CE	3022 - 3023	VA, Phase B	FLOAT	-9999 M to +9999 M	VAs	2
08CF	- 08D0	3024 - 3025	VA, Phase C	FLOAT	-9999 M to +9999 M	VAs	2
08D1	- 08D2	3026 - 3027	Power Factor, Phase A	FLOAT	-1.00 to +1.00	none	2
08D3	- 08D4	3028 - 3029	Power Factor, Phase B	FLOAT	-1.00 to +1.00	none	2
08D5	- 08D6	3030 - 3031	Power Factor, Phase C	FLOAT	-1.00 to +1.00	none	2
08D7	- 08D8	3032 - 3033	W-hours, Received	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08D9	- 08DA	3034 - 3035	W-hours, Delivered	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08DB	- 08DC	3036 - 3037	W hours, Net	SINT32	-99999999 to 99999999	Wh per energy format	2
08DD	- 08DE	3038 - 3039	W hours, Total	SINT32	0 to 99999999	Wh per energy format	2
08DF	- 08E0	3040 - 3041	VAR-hours, Positive	SINT32	0 to 99999999	VARh per energy format	2
08E1	- 08E2	3042 - 3043	VAR-hours, Negative	SINT32	0 to -99999999	VARh per energy format	2
08E3	- 08E4	3044 - 3045	VAh hours, Net	SINT32	-99999999 to 99999999	VAh per energy format	2
08E5	- 08E6	3046 - 3047	VAR-hours, Total	SINT32	0 to 99999999	VARh per energy format	2
08E7	- 08E8	3048 - 3049	VA-hours, Total	SINT32	0 to 99999999	VAh per energy format	2
08E9	- 08EA	3050 - 3051	W-hours, Received, Phase A	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08EB	- 08EC	3052 - 3053	W-hours, Received, Phase B	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08ED	- 08EE	3054 - 3055	W hours, Received, Phase C	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08EF	- 08F0	3056 - 3057	W-hours, Delivered, Phase A	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08F1	- 08F2	3058 - 3059	W hours, Delivered, Phase B	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08F3	- 08F4	3060 - 3061	W hours, Delivered, Phase C	SINT32	0 to 99999999 or 0 to -99999999	Wh per energy format	2
08F5	- 08F6	3062 - 3063	W-hours, Net, Phase A	SINT32	-99999999 to 99999999	Wh per energy format	2

For phase power and PF have values only for WYE hookup and will be zero for all other hookups.

\* Wh received & delivered always have opposite signs  
\* Wh received is positive for "view as load", delivered is positive for "view as generator"

\* 5 to 8 digits

\* decimal point implied, per energy format

\* resolution of digit before decimal point = units, kilo, or mega, per energy format

\* see note 10

Modbus Address								
Hex	Decimal	Description (Note 1)		Format	Range (Note 5)	Units or Resolution	Comments	# Reg
0BF7	- 0BF8	3064	- 3065	W hours, Net, Phase B	SINT32	-99999999 to 99999999	Wh per energy format	2
0BF9	- 0BF A	3066	- 3067	W hours, Net, Phase C	SINT32	-99999999 to 99999999	Wh per energy format	2
0BF B	- 0BF C	3068	- 3069	W hours, Total, Phase A	SINT32	0 to 99999999	Wh per energy format	2
0BF D	- 0BF E	3070	- 3071	W hours, Total, Phase B	SINT32	0 to 99999999	Wh per energy format	2
0BF F	- 0C00	3072	- 3073	W hours, Total, Phase C	SINT32	0 to 99999999	Wh per energy format	2
0C01	- 0C02	3074	- 3075	VAR hours, Positive, Phase A	SINT32	0 to 99999999	VARh per energy format	2
0C03	- 0C04	3076	- 3077	VAR hours, Positive, Phase B	SINT32	0 to 99999999	VARh per energy format	2
0C05	- 0C06	3078	- 3079	VAR hours, Positive, Phase C	SINT32	0 to 99999999	VARh per energy format	2
0C07	- 0C08	3080	- 3081	VAR hours, Negative, Phase A	SINT32	0 to -99999999	VARh per energy format	2
0C09	- 0C0A	3082	- 3083	VAR hours, Negative, Phase B	SINT32	0 to -99999999	VARh per energy format	2
0C0B	- 0C0C	3084	- 3085	VAR hours, Negative, Phase C	SINT32	0 to -99999999	VARh per energy format	2
0C0D	- 0C0E	3086	- 3087	VAR hours, Net, Phase A	SINT32	-99999999 to 99999999	VARh per energy format	2
0C0F	- 0C10	3088	- 3089	VAR hours, Net, Phase B	SINT32	-99999999 to 99999999	VARh per energy format	2
0C11	- 0C12	3090	- 3091	VAR hours, Net, Phase C	SINT32	-99999999 to 99999999	VARh per energy format	2
0C13	- 0C14	3092	- 3093	VAR hours, Total, Phase A	SINT32	0 to 99999999	VARh per energy format	2
0C15	- 0C16	3094	- 3095	VAR hours, Total, Phase B	SINT32	0 to 99999999	VARh per energy format	2
0C17	- 0C18	3096	- 3097	VAR hours, Total, Phase C	SINT32	0 to 99999999	VARh per energy format	2
0C19	- 0C1A	3098	- 3099	VA hours, Phase A	SINT32	0 to 99999999	VAh per energy format	2
0C1B	- 0C1C	3100	- 3101	VA hours, Phase B	SINT32	0 to 99999999	VAh per energy format	2
0C1D	- 0C1E	3102	- 3103	VA hours, Phase C	SINT32	0 to 99999999	VAh per energy format	2
							Block Size:	104
<b>Phase Angle Block</b>								<b>read-only</b>
1003	- 1003	4100	- 4100	Phase A Current	SINT16	-1800 to +1800	0.1 degree	1
1004	- 1004	4101	- 4101	Phase B Current	SINT16	-1800 to +1800	0.1 degree	1
1005	- 1005	4102	- 4102	Phase C Current	SINT16	-1800 to +1800	0.1 degree	1
1006	- 1006	4103	- 4103	Angle, Volts A-B	SINT16	-1800 to +1800	0.1 degree	1
1007	- 1007	4104	- 4104	Angle, Volts B-C	SINT16	-1800 to +1800	0.1 degree	1
1008	- 1008	4105	- 4105	Angle, Volts C-A	SINT16	-1800 to +1800	0.1 degree	1
							Block Size:	6

## **ANEXO A-3 ADICIÓN DE NUEVO NODO AL SISTEMA**

Para agregar un nuevo nodo que incluya medidor debemos seguir los siguientes pasos.

### **Procedimientos en la aplicación**

- 1- Agregar la información del nuevo nodo desde la aplicación web. Este paso consiste en abrir la aplicación web ir al enlace settings.
- 2- Ingresar el usuario y contraseña para que poder acceder a la sección administrativa de Django.
- 3- Seleccionar la opción add que aparece junto al nombre del modelo Nodes en la sección Dashboard
- 4- Llenar todo los campos del nodo. Si el nodo que se agrega funciona como repetidor en la opción de modelo llenar con cero, de lo contrario especificar si es una versión 100 o 200.
- 5- Ubicar geográficamente el nodo, o en su defecto ingresar manualmente las coordenadas.

### **Procedimiento para el router**

- 1- Editar los archivo main\_control.lua, cambiar los datos relacionados al modelo e ip del nodo.
- 2- Editar el archivo setsql.sh, el campo node\_id, debe corresponder al valor en decimal del ultimo octeto de la dirección ip de la red mesh, que sería 10.130.3.xxx, este número es el identificado del nodo y debe ser corresponder con la dirección agregada en el servidor, también se debe especifica en main\_lua.py la cantidad de filas que se enviaran al servidor por cada intento, si el router cuenta con una señal estable puede utilizar valores grandes como 50 a 100 registros de lo contrario utilizar valores entre 5 a 10 registros para que la aplicación pueda enviar datos en las lapsos cortos que tiene comunicación.

3- Agregar una ip en el rango de la familia a la computadora que se utilice, y conectarse a la red a través del switch del servidor o utilizar un router MPO1 para conectarse a la red de forma inalámbrica

4- Enviar los archivos al nodo utilizando scp en una terminal Linux. La sintaxis es

```
scp main_control.lua XMLPI_L.lua shark shark_query root@10.130.3.xxx:/root
```

5- El router solicitará la contraseña del usuario root para recibir los archivos y guardarlos en el directorio /root, se ingresa la contraseña y se enviarán los archivos.

6- El siguiente paso es conectarse de forma remota al medidor, y crear un directorio, en este proyecto se creó el directorio /luis\_tesis, la barra indica que está en el directorio raíz.

7- El siguiente paso es ejecutar el archivo setsqlite.sh, este archivo genera la base de datos embebida dbf, y dentro de esta la tabla Lectura, y la tabla Control.

8- El siguiente paso es hacer una prueba de la consulta. Se ejecuta en la línea de comandos lo siguiente: /usr/bin/lua main\_control.lua. Este comando indica que se debe utilizar el intérprete de lua ubicado en /usr/bin y se pasa como parámetro el script. El comando se ejecuta desde el directorio donde están los archivos de lo contrario se debe indicar la dirección del archivo main\_control.lua.

9- Si la prueba es satisfactoria no presentará algún error y se verá paso a paso la ejecución en la línea de comandos.

10- El paso final es agregar que la ejecución de este archivo sea de forma periódica con la herramienta crontab de la siguiente forma: crontab -e

11- Se abrirá un editor de texto en consola, y agregamos esta instrucción al final del archivo: \*/15 \* \* \* \* cd /luis\_tesis && /usr/bin/lua main\_control.lua

12- Para finalizar se deben guardar los cambios ejecutamos la siguiente secuencia

- presionamos la tecla escape

- Escribimos: wq!

- Presionamos la tecla Enter y así salimos del editor.
- 1- Nos desconectamos de la sesión del router con el comando éxito

Ahora nuestra aplicación se ejecuta en el nuevo nodo y estará enviando la información cada 15 minutos si hay conexión con el servidor, de lo contrario los datos atrasados se enviaran en los siguientes intentos



## **ANEXO A-4 SCRIPT DE MONITOREO DE LOS ENLACES INALÁMBRICOS**

El script de monitoreo está escrito en lenguaje Perl, este script se encarga de la consulta del estado de la red inalámbrica. La primera petición de la información al super nodo de la red, este super nodo actualiza constantemente las condiciones de operación, muestra en forma de listado cuales nodos tiene otros nodos cerca con lo que hay comunicación. Así es posible obtener un listado de estas relaciones, que describe a los vecinos de cada nodo y la calidad del enlace.

El script se encarga de extraer de la respuesta del súper nodo solamente la información necesaria para la visualización, cuando tiene el listado depurado procede a guardar la información en la base de datos, específicamente llena el campo neighbors de la tabla Nodes en el servidor cada vez que es ejecutado el script borra los datos anteriores de campo neighbors e ingresa los nuevo vecinos.

La parte de visualización se realiza por medio de angular, los datos se extraen de la tabla Node, con esto se crean los marcadores y líneas que utiliza Google Maps y así se visualizan el último estado al actualizar la página de la aplicación web.

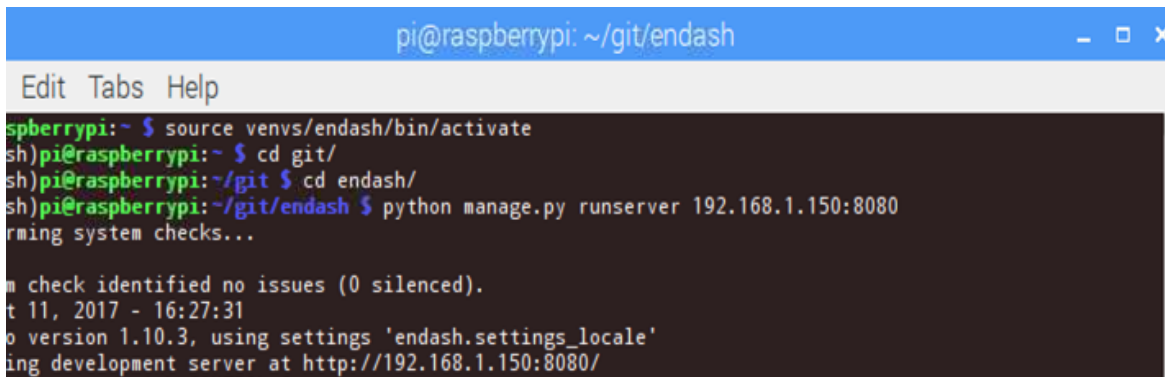
## **ANEXO B EJECUCIÓN DE LA APLICACIÓN EN MODO DE PRUEBAS**

Django provee herramientas para la depuración y prueba de la aplicación, es posible ejecutar un servidor web similar de pruebas que viene incluido con Django, este servidor se ejecuta desde la consola y al mismo tiempo presenta las operaciones que se llevan a cabo en ejecución de la aplicación, permite depurar los errores de ejecución, en la mayoría de los casos es posible efectuar las modificaciones de los archivos que sean requeridos sin tener que reiniciar este servidor de pruebas.

El procedimiento para ejecutarlo se presenta en la FIGURA 73, lo primero que se debe hacer es indicarle al sistema que se utilizara una versión de Python virtual, esto permite ejecutar un intérprete de Python que incluye Django con todas las librerías y módulos propios de la aplicación, esto evita comprometer los módulos utilizados con la ejecución de Python del sistema del servidor.

Para hacer esto se utiliza la herramienta source, que permite utilizar la versión de Python virtual, se debe pasar como argumento el directorio donde se encuentran los archivos y configuraciones de la versión de Python virtual, posteriormente como se aprecia en la FIGURA 73, aparece un indicador en las línea de entrada de comando como (endash), significa que esta terminal ahora direccionara la ejecución de Python a la versión virtual.

El siguiente paso es ejecutar la aplicación, para esto ejecutamos el archivo manage.py, que administra nuestro proyecto y es la interfaz para interactuar con nuestra aplicación, a este archivo se le pasa el parámetro runserver, que inicializa el servidor en una dirección ip por defecto o la especificamos, en este caso se ingresa la ip del servidor y el puerto, esto permite que la aplicación pueda ser visible desde dispositivo en la red del servidor.

A terminal window titled 'pi@raspberrypi: ~/git/endash' with standard window controls. The terminal shows a sequence of commands and their outputs: activating a virtual environment, navigating to the project directory, and running the Django server. The output indicates that all system checks passed and the server is running on port 8080.

```
pi@raspberrypi:~$ source venvs/endash/bin/activate
sh)pi@raspberrypi:~$ cd git/
sh)pi@raspberrypi:~/git$ cd endash/
sh)pi@raspberrypi:~/git/endash$ python manage.py runserver 192.168.1.150:8080
Performing system checks...

System check identified no issues (0 silenced).
11/11/2017 16:27:31
Django version 1.10.3, using settings 'endash.settings_locale'
Starting development server at http://192.168.1.150:8080/
```

FIGURA 73. INICIALIZACIÓN DEL SERVIDOR DE PRUEBA CON DJANGO

## ANEXO C-1 REDISEÑO DEL PROTOTIPO DE BAJO COSTO BASADO EN RASPBERRY PI

La primera implementación del prototipo fue utilizando una Raspberry Pi, esta utilizaba una memoria SD como unidad de almacenamiento, esto ocasionaba que el sistema fuera vulnerable a fallas debido a escrituras de la memoria, ya que alcanzaba con facilidad el límite de escritura al utilizarla como medio para almacenar el sistema operativo. La solución fue utilizar discos duros para almacenar el sistema operativo, la ventaja que se obtuvo fue una mayor vida útil del medio de almacenamiento, así como un considerable aumento en la capacidad de almacenaje.

Una de los retos que presentó esta modificación es la inclusión de una fuente de alimentación adicional para el disco duro además de la utilización de adaptadores de puerto SATA a USB. Al mismo tiempo se planteó que el sistema debía tener redundancia en caso de falla para mantener los datos íntegros, el método empleado fue utilizar una segunda raspberry que actuara como respaldo en caso de que la primera fallara, así evolucionó el sistema de respaldo hasta implicar el uso de 2 discos duros, dos raspberry PI y dos fuentes de alimentación.

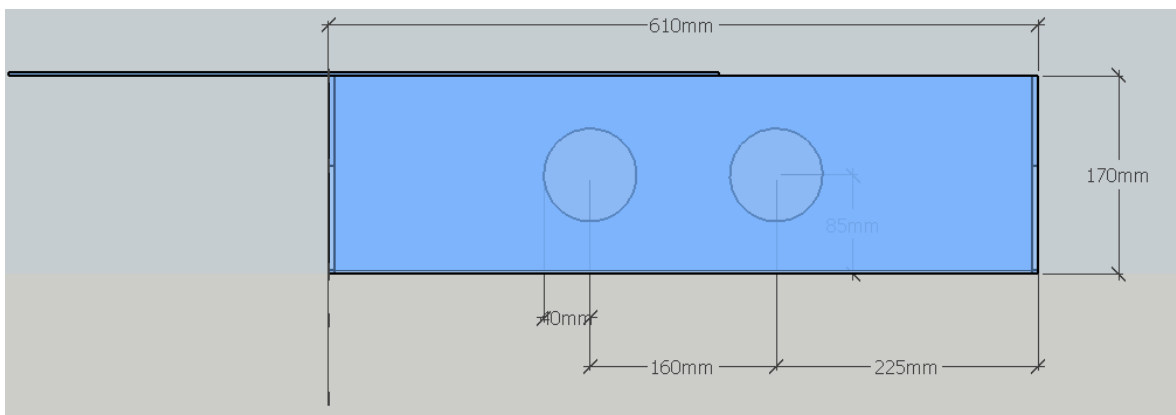


FIGURA 74. VISTA FRONTAL DEL PROTOTIPO BASA EN RASPBERRY PI

Para incorporar los elementos de una forma didáctica y de fácil manipulación se procedió con el diseño de un gabinete que se ajustara a los requerimientos indicados, así es como se llegó al diseño utilizado actualmente, además de incluir dos ventiladores para refrigerar el nuevo prototipo. La vista frontal se presenta en la FIGURA 74.

La inclusión de dos fuentes de poder como la utilizada en las computadoras de escritorio facilitó la alimentación de los discos duros, y se realizaron los ajustes para adaptar la alimentación de las Raspberry Pi's a estas fuentes. En FIGURA 75 se observan los espacios dejados para la ubicación de las fuentes de energía.

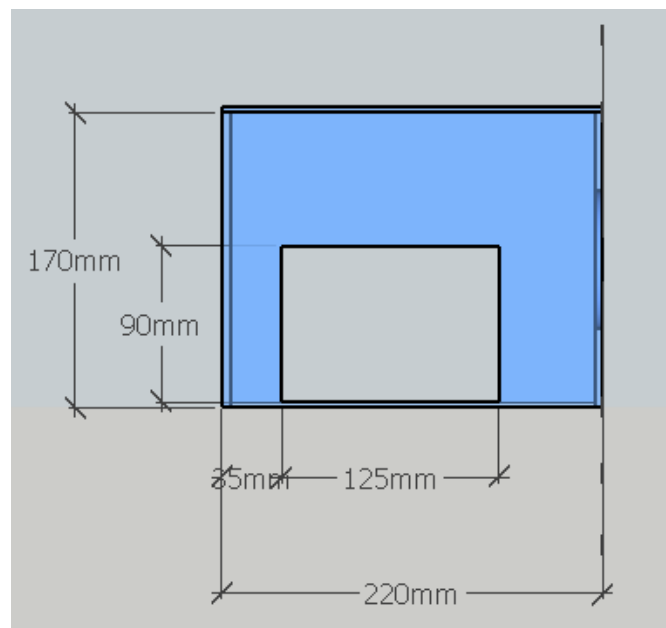


FIGURA 75. VISTA LATERAL DEL PROTOTIPO

En la FIGURA 76 se presenta una vista isométrica del diseño realizado, donde se presenta la parte superior corrediza que permite el acceso a los componentes internos, si fuere necesario, el material seleccionado fue acrílico transparente en tono neón para poder visualizar los indicadores led de las microcomputadoras.

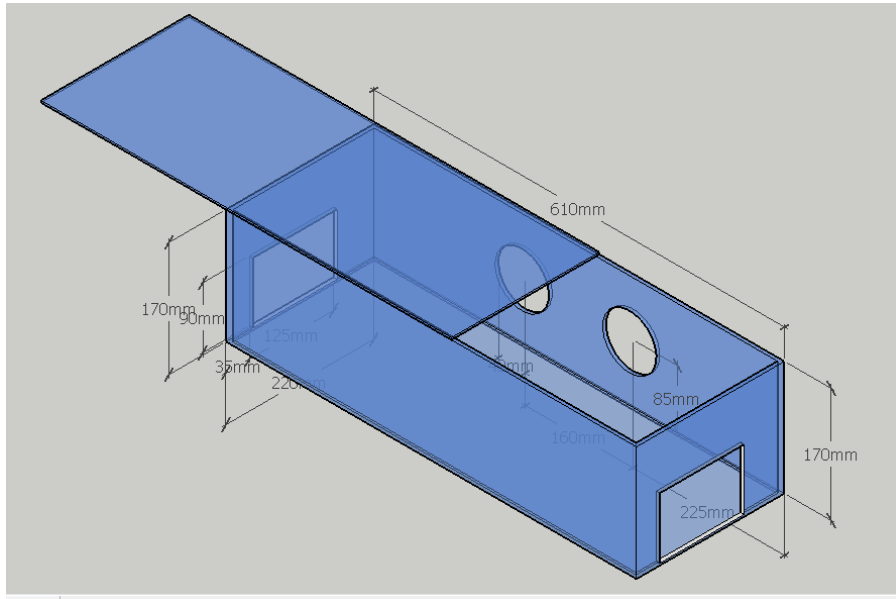


FIGURA 76. VISTA ISOMÉTRICA DEL GABINETE

Otro elemento utilizado fueron los soportes para las raspberry, esto están unidos al soporte metálico de los discos duros y permiten acceder a los conectores de las microcomputadoras de forma fácil y rápida. El material utilizado también es acrílico en tono neón.

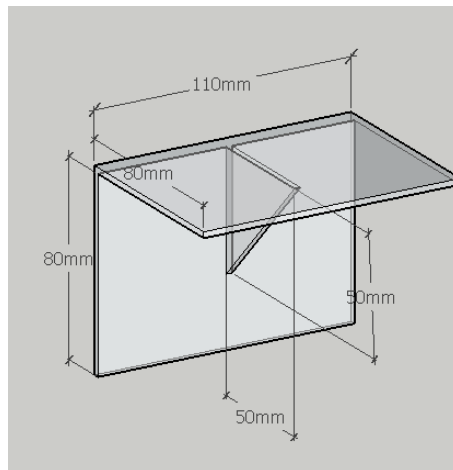


FIGURA 77. SOPORTE PARA RASPBERRY PI

De la FIGURA 78 a la 81 presentan el gabinete con los diferentes elementos ya ubicados en el su interior.

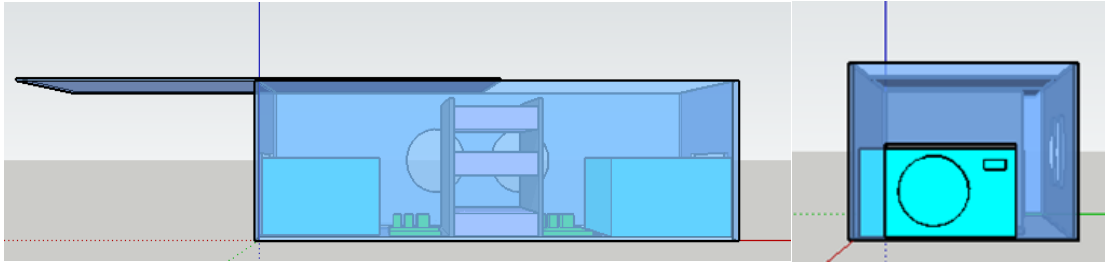


FIGURA 78. VISTAS FRONTAL Y LATERAL DERECHA

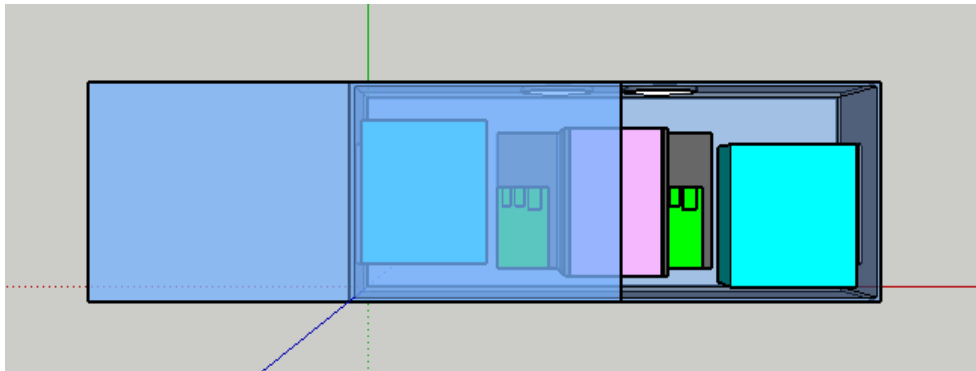


FIGURA 79. VISTA SUPERIOR

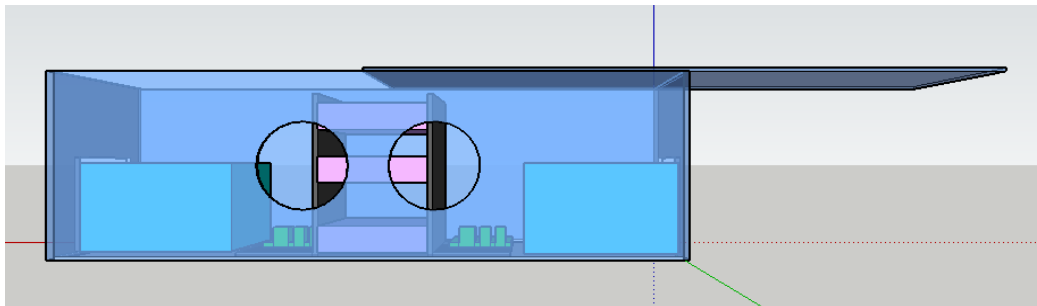


FIGURA 80. VISTA POSTERIOR

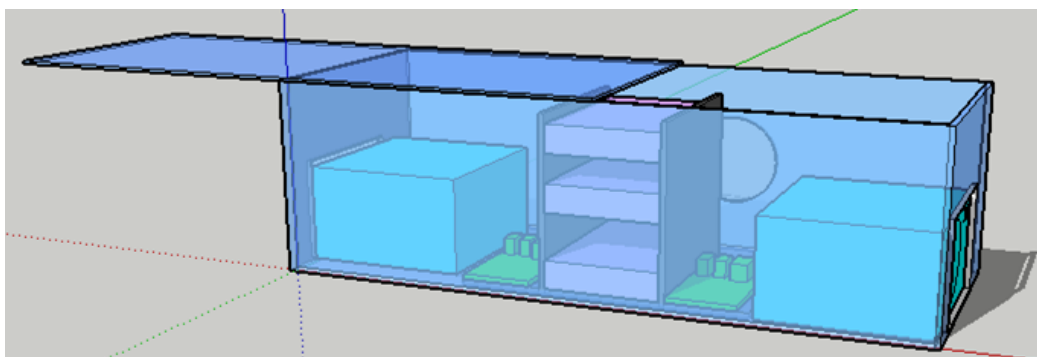


FIGURA 81. VISTA ISOMÉTRICA

## ANEXO C-2 INSTALACIÓN DE RASPBIAN EN DISCO DURO

La Raspberry Pi sólo arranca desde la SD así que es imposible que hacerla arrancar desde un disco duro. Lo que sí es posible es trasladar la partición de arranque a un disco duro obteniendo un aumento significativo de la velocidad y de la “confiabilidad” del equipo

Esta configuración es especialmente recomendable al usar la Pi como servidor de ficheros, media server o para usar una base de datos.

Los pasos básicos son:

- Instalar el sistema operativo de la raspberry PI en la tarjeta SD
- Conectar el disco duro
- Preparar el disco con las particiones y formatos adecuados
- Copiar el sistema desde la SD al disco duro
- Modificar el arranque para que acceda al disco y no a la tarjeta

Partimos de la base que el sistema está instalado y la Raspberry Pi funciona perfectamente y que tiene acceso a Internet.

### 1) Actualizar

El primer paso es actualizar el software y el firmware. Para ello en la consola usaremos los siguientes comandos, contestando si (yes) en caso que nos indique que descargará ficheros:

```
sudo apt-get update
```



```
sudo apt-get upgrade
```

```
sudo rpi-update
```

```
sudo reboot
```

El proceso completo puede tardar algunos minutos.

## **2) Conectar el disco duro**

Tan simple como conectar el disco al puerto USB, pero ojo, la raspberry PI no tiene potencia por si sola para alimentar un disco duro externo (o a casi ningún disco externo, que puede que con suerte alguno sí que funcione). Es necesario usar un disco con alimentación propia o bien un concentrador USB activo (con su propia fuente de alimentación). Si se usa directamente un disco sin alimentación puede que funcione pero tarde o temprano se producen errores de lectura o reinicios de la propia raspberry PI.

## **3) Particionado y formateo**

Con el disco correctamente conectado ejecutamos:

```
sudo fdisk -l
```

Se nos mostrarán las particiones de la SD y las del disco duro. Lo más probable es que la SD y el HD (disco duro) aparezcan respectivamente como

```
/dev/sda y /dev/mmcblk0
```

Crearemos en el disco como mínimo tres particiones: la de arranque, la de intercambio (swap) y la de trabajo.

Se borrarán todos los datos del disco duro. Si el disco contiene información que quieras guardar haz una copia en otro sitio primero.

Ejecuta

```
sudo fdisk /dev/sda
```

Entra “p” para ver las particiones existentes en el disco. Si existen particiones bórralas de una en una con “d” e indicando en número de partición.

En cuanto no existan particiones (o directamente si en disco no tenía ninguna definida) entra “n” para crear la primera partición.

Esta será la partición del sistema, entra “p” (primaria) y “1” como número de partición. Como sector de inicio deja el que el sistema propone y como sector de fin entra “+8GB”. (Puedes indicar cualquier otro tamaño siempre y cuando tenga como mínimo el mismo tamaño que la tarjeta SD) .

Entra “n” de nuevo, para la segunda partición, esta vez la de intercambio (swap) a la que daremos un tamaño de 2GB. Como antes tipo “p”, número de partición “2”, sector de inicio el propuesto por el sistema y sector de fin “+2GB”.

Por último creamos la partición de trabajo. Entra “n”, “p”, partición “3” y deja los valores por defecto para el sector de inicio y de fin.

Con “p” se muestra el listado de las particiones que hemos creado y si está todo a tu gusto entra “w” para que los cambios sean permanentes y salir.

Formateamos la partición de trabajo y la de intercambio respectivamente con los comandos:

```
sudo mkfs.ext4 /dev/sda3
```

```
sudo mkswap /dev/sda2
```

La partición del sistema se formatea “sola” al copiar el contenido de la SD, así que no tenemos que hacer nada ahora.

#### **4) Copiar el sistema al disco duro**

Primero veamos donde está la partición de arranque en la SD. Para ello ejecuta

```
cat /boot/cmdline.txt
```

Busca donde pone “root=” y toma nota de lo que aparece detrás (en mi caso root=/dev/mmcbk0p6)

Para copiar la partición de arranque al disco duro:

```
sudo dd if=/dev/mmcbk0p6 of=/dev/sda1 bs=32M conv=noerror,sync
```

(modifica el primer parámetro, lo que está después del “if=”, para que ponga lo que acabas de anotar)

Comprueba que no hay errores:

```
e2fsck -f /dev/sda1
```

y amplía la partición para ocupar todo el espacio disponible

```
resize2fs /dev/sda1
```

## **5) Configurar el arranque**

Lo primero es hacer una copia de la configuración actual:

```
sudo cp /boot/cmdline.txt /boot/cmdline.old
```

Ahora editamos la configuración modificando el fichero cmdline.txt para ello ejecutamos

```
sudo nano /boot/cmdline.txt
```

y modificamos el contenido poniendo /dev/sda1 donde ahora pone /dev/mmcbk0p6

Con Control+O y Control+X guardas los cambios y sales del editor.

Modificaremos ahora la configuración de carga de discos del nuevo sistema. Para ello, montamos el arranque del disco duro

```
sudo mount /dev/sda1 /mnt
```

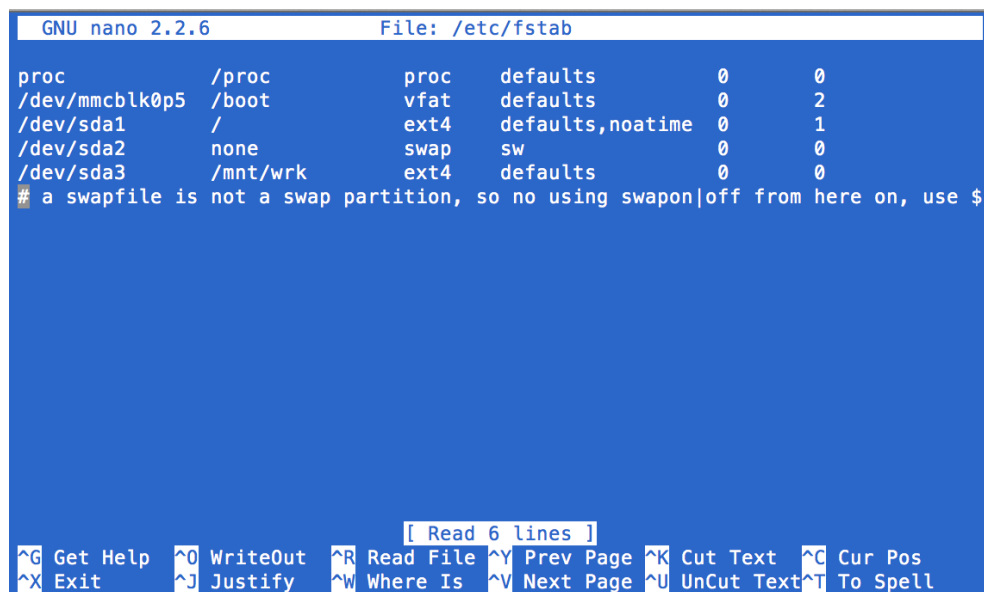
Y modificamos los parámetros de carga de los discos, de modo que se carguen de forma automática las particiones de swap y datos que están en el disco duro:

```
sudo nano /mnt/etc/fstab
```

Donde pone “/dev/mmcbk0p6” pon “/dev/sda1” (sin las comillas) y añada después dos líneas debajo

```
/dev/sda2 none swap sw 0 0
```

```
/dev/sda3 /mnt/wrk ext4 defaults 0 0
```



```
GNU nano 2.2.6 File: /etc/fstab
proc /proc proc defaults 0 0
/dev/mmcbk0p5 /boot vfat defaults 0 2
/dev/sda1 / ext4 defaults,noatime 0 1
/dev/sda2 none swap sw 0 0
/dev/sda3 /mnt/wrk ext4 defaults 0 0
# a swapfile is not a swap partition, so no using swapon|off from here on, use $
```

[ Read 6 lines ]

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

FIGURA 82. UBICACIÓN DE LAS PARTICIONES

Control+O y Control+X para guardar y salir del editor.

Crearemos ahora el punto de montaje para la partición de datos

```
sudo mkdir /mnt/wrk
```

y borrarémos el fichero de swap

```
sudo rm /etc/rc2.d/S02dphys-swapfile
```

Para terminar, nos aseguramos de que todo queda grabado y reiniciamos:

```
sync
```

```
sudo reboot
```

La Raspberry Pi arrancará desde la SD para cargar el sistema desde el disco duro

## BIBLIOGRAFÍA

[1] Bonilla Perla, Juan José (2014) Diseño, configuración y supervisión de la red de medidores de energía eléctrica del campus central de la Universidad de El Salvador.

<http://ri.ues.edu.sv/5584/>

[2] Duque Alas, Alexander Omar (2014) Optimización del sistema de monitorización remota de medidores de energía eléctrica. Tesis Ingeniería, Universidad de El Salvador.

<http://ri.ues.edu.sv/6534/>

[3] Renderos Alfaro, Samuel Antonio (2017) Implementación XML-RPC para la monitorización de medidores de energía eléctrica. Tesis de Ingeniería, Universidad de El Salvador

<http://ri.ues.edu.sv/12865/>

[4] Página oficial del proyecto Mesh Potato (MP01) de Village Telco.

[http://wiki.villagetelco.org/Mesh\\_Potato\\_\(MP01\)](http://wiki.villagetelco.org/Mesh_Potato_(MP01))

[5] Redes Mesh.

[http://wiki.ead.pucv.cl/index.php/Red\\_Mesh](http://wiki.ead.pucv.cl/index.php/Red_Mesh)

[6] Guía de referencia sobre firmware SECN 1.1 y Batman-Adv.

[http://wiki.villagetelco.org/SECN\\_1.1\\_User\\_Guide](http://wiki.villagetelco.org/SECN_1.1_User_Guide)

[7] Toolchain OpenWrt.

<https://wiki.openwrt.org/es/about/toolchain>

[8] Protocolo Modbus.

<http://www.tolaemon.com/docs/modbus.htm>

[9] Página oficial del proyecto OpenWrt.

<https://openwrt.org/>

[10] Wiki del proyecto batman-adv

<https://www.open-mesh.org/projects/batman-adv/wiki/Wiki>

[11] SPUD de villagetelco página oficial

<https://villagetelco.org/2011/06/spud-simple-unified-dashboard-for-mesh-networks/>

[12] Django Project página oficial

<https://www.djangoproject.com/>

[13] Página oficial del proyecto Angular

<https://angularjs.org/>