UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DIDÁCTICO PARA EL MONITOREO DE SENSORES A TRAVÉS DE UNA PLATAFORMA IOT UTILIZANDO PROTOCOLOS DE COMUNICACIÓN MQTT Y MODBUS TCP/IP"

Trabajo de titulación previo a la obtención del título de Ingeniero Electrónico.

AUTOR: DIEGO ARMANDO PINOARGOTE PAREDES

TUTOR: ING. DIEGO ROBERTO FREIRE QUIROGA, MSC.

Guayaquil – Ecuador

2021

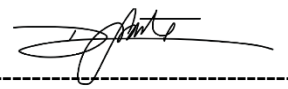
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACION

Yo, **Diego Armando Pinoargote Paredes** con documento de identificación **N° 0931446058** manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 10 de septiembre del año 2021.

Atentamente,



Diego Armando Pinoargote Paredes
0931446058

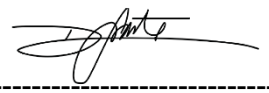
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACION A LA UNIVERSIDAD POLITECNICA SALESIANA

Yo, **Diego Armando Pinoargote Paredes** con documento de identificación **N° 0931446058**, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy el autor del proyecto técnico: **"Diseño e implementación de un prototipo didáctico para el monitoreo de sensores a través de una plataforma IoT utilizando protocolos de comunicación MQTT y MODBUS TCP/IP"**, el cual sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato digital a la biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 10 de septiembre del año 2021.

Atentamente,



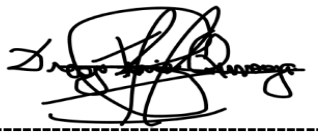
Diego Armando Pinoargote Paredes
0931446058

CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN

Yo, Diego Roberto Freire Quiroga con documento de identificación **N° 0917208084**, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación "**DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DIDÁCTICO PARA EL MONITOREO DE SENSORES A TRAVÉS DE UNA PLATAFORMA IOT UTILIZANDO PROTOCOLOS DE COMUNICACIÓN MQTT Y MODBUS TCP/IP** " realizado por **Diego Armando Pinoargote Paredes**, con documento de identificación **N°0931446058**, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 10 de septiembre del año 2021.

Atentamente,



Ing. Diego Roberto Freire Quiroga, MSc.

0917208084

DEDICATORIA

A mis padres, por su perenne apoyo y amor incondicional.

A mi esposa, por su paciencia y apoyo a lo largo de este camino.

A mi tutor, por su colaboración y guía.

Al lector.

Diego Armando Pinoargote Paredes

AGRADECIMIENTO

A Dios.

A mi mamá, que siempre me ha brindado su apoyo, calidez y cariño.

A mi papá, quien siempre ha trabajado arduamente por el bienestar de la familia.

A mi esposa, me ha acompañado desde mi formación académica, me supo brindar su tiempo, dedicación, apoyo y la motivación necesaria.

A mi tutor, quien dentro y fuera de clases, ha colaborado incondicionalmente para mi formación académica y durante el desarrollo de este proyecto.

A los docentes que brindaron su conocimiento y experiencias para mi desarrollo académico e intelectual a lo largo de este camino.

A mis amigos, quienes fueron compañía y el equipo de trabajo ideal durante el transcurso de mi vida académica.

Diego Armando Pinoargote Paredes

RESUMEN

AÑO	ALUMNOS	DIRECTOR DE PROYECTO TÉCNICO	TEMA DE PROYECTO TÉCNICO
2021	DIEGO ARMANDO PINOARGOTE PAREDES	ING. DIEGO ROBERTO FREIRE QUIROGA, MSc.	DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DIDÁCTICO PARA EL MONITOREO DE SENSORES A TRAVÉS DE UNA PLATAFORMA IOT UTILIZANDO PROTOCOLOS DE COMUNICACIÓN MQTT Y MODBUS TCP/IP

El presente proyecto técnico muestra el diseño, desarrollo e implementación de un prototipo didáctico que permite realizar la comunicación y la obtención de información de un sistema de sensores, desde la plataforma Kaa IoT, permitiendo al usuario visualizar la información del sistema de sensores de manera remota.

Este proyecto se ejecutó mediante el uso de sistemas embebidos(Arduino Mega y Raspberry Pi 4), el sistema de sensores gestionado por Arduino, se enlazó con la Raspberry Pi haciendo uso del protocolo de comunicación MODBUS TCP/IP. A demás, la Raspberry Pi fue configurada para establecer la comunicación con la plataforma Kaa IoT utilizando como enlace la plataforma NODE-RED y cumplir la función de Gateway IoT haciendo uso del protocolo de comunicación MQTT. La plataforma Kaa IoT permite al usuario el monitoreo en tiempo real y el acceso a la información que proporciona el sistema de sensores, a su vez esta información generada es almacenada en una base de datos local instalada en la Raspberry Pi.

El concepto de IoT establece la conectividad de dispositivos a través de la red, para el hogar y/o industria, dentro del entorno industrial nace el concepto de IIoT que hace referencia a dispositivos inteligentes que se encuentran interconectados dentro de la red lo que favorece el monitorear parámetros en tiempo real y solventar eventos. Para las Pymes, los dispositivos interconectados entre sí, pueden llegar a cumplir una función primordial, gracias a que despliegan un abanico de soluciones en temas de automatización, logística, administración, eficiencia energética, entre otros, dónde los dispositivos y sensores conectados a la red permiten analizar los datos, generar alarmas y mensajes destinados al administrador o usuario a cargo, para tomar acciones, prevenir o responder de manera oportuna.

Por tanto, este proyecto se ejecutó bajo la metodología experimental, aplicando el conocimiento obtenido luego de la investigación y experimentación realizada durante su ejecución, que permitió establecer parámetros y realizar las configuraciones necesarias de cada dispositivo, sensor y plataformas usadas. De esta manera, el prototipo didáctico desarrollado e implementado cumple con todos sus objetivos.

ABSTRACT

YEAR	STUDENTS	DIRECTOR OF TECHNICAL PROJECT	TECHNICAL PROJECT THEME
2021	DIEGO ARMANDO PINOARGOTE PAREDES	ING. DIEGO ROBERTO FREIRE QUIROGA, MSc.	DESIGN AND IMPLEMENTATION OF A DIDACTIC PROTOTYPE FOR SENSOR MONITORING THROUGH AN IOT PLATFORM USING MQTT AND MODBUS TCP / IP COMMUNICATION PROTOCOLS.

This technical project shows the design, development and implementation of a didactic prototype that allows communication and obtaining information from a sensor system, from the Kaa IoT platform, allowing the user to visualize the information of the sensor system remotely.

This project was executed through the use of embedded systems (Arduino Mega and Raspberry Pi 4), the sensor system managed by Arduino, was linked with the Raspberry Pi making use of the MODBUS TCP/IP communication protocol. In addition, the Raspberry Pi was configured to establish communication with the Kaa IoT platform using the NODE-RED platform as a link and fulfill the function of IoT Gateway using the MQTT communication protocol. The Kaa IoT platform allows the user to monitor in real time and access to the information provided by the sensor system, in turn this generated information is stored in a local database installed on the Raspberry Pi.

The concept of IoT establishes the connectivity of devices through the network, for the home and / or industry, within the industrial environment is born the concept of IIoT that refers to smart devices that are interconnected within the network which favors monitoring parameters in real time and solving events. For SMEs, devices interconnected with each other, can come to play a primary role, thanks to the fact that they deploy a range of solutions in areas of automation, logistics, administration, energy efficiency, among others, where the devices and sensors connected to the network allow to analyze the data, generate alarms and messages intended for the administrator or user in charge, to take actions, prevent or respond in a timely manner.

Therefore, this project was executed under the experimental methodology, applying the knowledge obtained after the research and experimentation carried out during its execution, which allowed to establish parameters and make the necessary configurations of each device, sensor and platforms used. In this way, the didactic prototype developed and implemented meets all its objectives.

ÍNDICE GENERAL

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA I
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR II
CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN III
DEDICATORIA IV
AGRADECIMIENTO V
RESUMEN VI
ABSTRACT VII
ÍNDICE GENERAL VII
ÍNDICE DE FIGURAS XII
ÍNDICE DE TABLAS XIV
INTRODUCCIÓN 1
1. CAPITULO 1: EL PROBLEMA 2
1.1. ANTECEDENTES 2
1.2. IMPORTANCIA Y ALCANCES 3
1.3. DELIMITACIÓN 3
1.3.1. TEMPORAL 3
1.3.2. ESPACIAL 3
1.3.3. ACADÉMICA 3
1.4. EXPLICACIÓN DEL PROBLEMA 3
1.5. OBJETIVOS 5
1.5.1. OBJETIVO GENERAL 5
1.5.2. OBJETIVOS ESPECÍFICOS 5
2. FUNDAMENTOS TEÓRICOS 5
2.1. INTERNET DE LAS COSAS (IOT) 5
2.1.1. PLATAFORMA IOT 6
2.2. INTERNET INDUSTRIAL DE LAS COSAS (IIOT) 6
2.2.1. LA MEJORA DE IIOT EN ENTORNOS INDUSTRIALES 6
2.2.2. CARACTERÍSTICAS DE LOS DISPOSITIVOS IIOT 6
2.2.3. VENTAJAS DE LOS DISPOSITIVOS IIOT 7
2.3. MQTT 8
2.3.1. FUNCIONAMIENTO DE MQTT 8
2.4. TCP/IP 9
2.4.1. CAPAS DEL MODELO TCP/IP 9

2.4.1.1. CAPA DE ACCESO A RED O FÍSICA 10
2.4.1.2. CAPA DE RED O INTERNET 10
2.4.1.3. CAPA DE TRANSPORTE 10
2.4.1.4. CAPA DE APLICACIÓN 10
2.5. MODBUS 11
2.5.1. MODBUS TCP/IP 11
2.6. NODE-RED 12
2.7. BASE DE DATOS 13
2.8. MARIADB 13
2.9. PHPMYADMIN 13
2.10. ARDUINO 14
2.10.1. MODELOS DE ARDUINO 14
2.10.1.1. ARDUINO MEGA 2560 15
2.10.2. ARDUINO ETHERNET SHIELD 15
2.11. RASPBERRI PI 16
2.11.1. RASPBERRI PI 4 16
2.12. I2C 18
2.12.1. I2C EN ARDUINO 18
2.13. FUENTE DE ALIMENTACIÓN MEAN WELL MDR 60-12 AC a DC 19
2.14. PORTAFUSIBLE EBCHQ-36900 19
2.15. TOMACORRIENTE 19
2.16. SELECTOR ELÉCTRICO 20
2.17. LUCES PILOTO 20
2.18. LCD 4x20 20
2.19. TP-LINK WR741ND 21
2.20. REGULADOR DE VOLTAJE LM2596 21
2.21. SENSOR DE TEMPERATURA DS18B20 22
2.22. SENSOR DE FLAMA YG1006 22
2.23. SENSOR DE PRESENCIA HC-SR501 22
2.24. SENSOR DE GAS MQ-135 23
2.25. SENSOR ULTRASÓNICO HC-SR04 23
2.26. BUZZER 24
2.27. CAMARA WEB 24
2.28. TABLERO PLÁSTICO ELÉCTRICO 25

3. FUNDAMENTOS METODOLÓGICOS 26
3.1. MÉTODO DE ESTUDIO 26
3.1.1. METODO EXPERIMENTAL 26
3.2. DESCRIPCIÓN DEL PROTOTIPO 26
3.2.1. DIAGRAMA ESQUEMÁTICO DEL PROTOTIPO 28
3.3. CONFIGURACIONES DE RED 28
3.3.1. TOPOLOGÍA DE RED 29
3.3.2. CONFIGURACIONES DE ROUTER 30
3.3.2.1. ACCESO AL PANEL DEL ROUTER 30
3.4. CONFIGURACIONES EN RASPBERRY PI 4 33
3.4.1. INSTALACIÓN DE SISTEMA OPERATIVO EN RASPBERRY PI 4 34
3.4.2. INSTALACIÓN DE NODE-RED 36
3.4.2.1. DIAGRAMA DE FLUJO EN NODE-RED. 38
3.4.3. INSTALACIÓN DE BASE DE DATOS. 38
3.4.4. CONFIGURACIÓN DE CAMARA WEB. 44
3.5. CONFIGURACIÓN EN ARDUINO. 45
3.5.1. PROTOCOLO DE COMUNICACIÓN MODBUS. 45
3.5.2. SENSORES EN ARDUINO 46
3.5.2.1. SENSOR DE TEMPERATURA 47
3.5.2.2. SENSOR DETECTOR DE FLAMA. 47
3.5.2.3. SENSOR DE PRESENCIA. 48
3.5.2.4. SENSOR DE GAS. 48
3.5.2.5. SENSOR ULTRASÓNICO 48
3.5.3. PRESENTACIÓN DE MENSAJES EN LCD. 49
3.6. CONFIGURACIÓN DE LA PLATAFORMA KAA IOT. 49
3.7. CONFIGURACIÓN DE ALARMA 53
3.8. MONTAJE Y ENSAMBLAJE. 54
3.8.1. DIAGRAMA ELÉCTRICO. 57
4. RESULTADOS 59
4.1. SISTEMA DE SENSORES CON ARDUINO 59
4.1.1. PRUEBA DE FUNCIONAMIENTO DE LOS SENSORES 59
4.1.1.1. SENSOR DE TEMPERATURA 59
4.1.1.2. SENSOR DETECTOR DE FLAMA 60
4.1.1.3. SENSOR DE PRESENCIA 61

4.1.1.4. SENSOR DE GAS 61
4.1.1.5. SENSOR ULTRASÓNICO 62
4.2. COMUNICACIONES CON LA PLATAFORMA KAA IOT 63
4.3. DASHBOARD EN KAA IOT 63
4.4. BASE DE DATOS 64
5. ANÁLISIS DE RESULTADOS 65
6. CONCLUSIÓN 68
7. RECOMENDACIÓN 69
8. REFERENCIAS BIBLIOGRÁFICAS 70
9. ANEXOS 72
9.1. ANEXO 1: CÓDIGO EN ARDUINO 72
9.2. ANEXO 2: CÓDIGO EN NODE-RED 75
9.3. ANEXO 3: CÓDIGO DE MQTT EN NODE RED 92

ÍNDICE DE FIGURAS

Figura 1: Nueva dimensión que introduce IoT. [1] 5
Figura 2: Modelo de referencia de alto nivel. [3] 7
Figura 3: Comunicación entre bróker y clientes. [4] 8
Figura 4: Mensajes CONNECT y CONNACK. [4] 8
Figura 5: Mensajes SUSCRIBE y SUBACK. [4] 9
Figura 6: Capas o niveles del modelo TCP/IP. [6] 10
Figura 7: Relación de Red maestro-esclavo. [8] 11
Figura 8: Interfaz Web de Node-RED. [13] 12
Figura 9: Interfaz gráfica de phpMyAdmin. [16] 14
Figura 10: Arduino MEGA 2560. [18] 15
Figura 11: Módulo Arduino Shield Ethernet. 16
Figura 12: Diagrama de Raspberri Pi. [21] 16
Figura 13: Características de Raspberri Pi 4. [23] 17
Figura 14: Integrado I2C. 18
Figura 15: Esquema de conexión de Arduino UNO y el I2C MPU6050. [24] 18
Figura 16: Fuente de alimentación Mean Well MDR-60-12. [25] 19
Figura 17: Disyuntor Merlin Gerin Multi 9 C60N. 19
Figura 18: Tomacorriente. 20
Figura 19: Selector eléctrico de 2 posiciones. 20
Figura 20: Luces pilotos. 20
Figura 21: LCD 4x20. 21
Figura 22: Router TP-LINK WR741ND. 21
Figura 23: Módulo regulador de voltaje ajustable. 21
Figura 24: Sensor de temperatura DS18B20. 22
Figura 25: Sensor de flama YG1006. 22
Figura 26: Sensor HC-SR501. 23
Figura 27: Sensor de control de calidad de aire MQ-135. 23
Figura 28: Sensor ultrasónico HC-SR04. 24
Figura 29: Buzzer activo. 24
Figura 30: Cámara web. 24
Figura 31: Tablero plástico eléctrico. 25
Figura 32: Diagrama esquemático del prototipo. 28
Figura 33: Topología de red. 29
Figura 34: Acceso a panel del router TL-WR741ND. 30
Figura 35: Panel de principal del router TL-WR741ND. 30
Figura 36: Configuración de red LAN. 31
Figura 37: Configuración de red WAN. 31
Figura 38: Configuración de red WAN. 32
Figura 39: Configuración de red WIFI. 32
Figura 40: Configuración de red WIFI. 33
Figura 41: Asignación de IP estáticas. 33
Figura 42: Panel de RASPBERRY PI IMAGER. 34
Figura 43: Panel de RASPBERRY PI IMAGER. 35
Figura 44: Panel de RASPBERRY PI IMAGER. 35
Figura 45: Panel de RASPBERRY PI IMAGER. 35
Figura 46: Inicio de Node-RED. 36

Figura 47: Consola de Node-RED. 37
Figura 48: Entorno Node-RED. 37
Figura 49: Librerías en Node-RED. 37
Figura 50: Diagrama de flujo en Node-RED. 38
Figura 51: Comando sudo apt update. 39
Figura 52: Instalación deapache2 como servidor. 39
Figura 53: Verificación el estado de Apache. 40
Figura 54: Apache instalado correctamente. 40
Figura 55: Instalación de PHP. 41
Figura 56: PHP instalado correctamente. 41
Figura 57: Instalamos el servidor de base de datos MariaDB. 42
Figura 58: Verificación de estado de MariaDB. 42
Figura 59: Acceso a MariaDB como root. 43
Figura 60: Instalación de phpMyAdmin. 43
Figura 61: Presentación de phpMyAdmin. 44
Figura 62: Interfaz de la base de datos. 44
Figura 63: Configuración de cámara web en Node-RED. 45
Figura 64: Arduino Mega 2660 con Arduino Ethernet Shield. 45
Figura 65: Librerías MgsModbus incluidas en la hoja de código de Arduino. 46
Figura 66: Sistema de sensores. 47
Figura 67: Código de acceso a Kaa IoT desde Node-RED. 49
Figura 68: Código de envió de datos de Arduino a Kaa IoT. 50
Figura 69: Configuración de Kaa IoT en Node-RED. 50
Figura 70: Plataforma Kaa IoT. 51
Figura 71: Ingreso a Root account. 51
Figura 72: Registro de dispositivos en la Kaa IoT. 52
Figura 73: Transmisión de datos de Node-RED hacia Kaa IoT. 52
Figura 74: Creación de dashboard. 53
Figura 75: Configuración de dashboard. 53
Figura 76: Configuración de alarma en Node-RED 54
Figura 77: Base de Arduino MEGA 2560. 54
Figura 78: Base de Raspberri Pi 4. 54
Figura 79: Dispositivos instalados en plafón. 55
Figura 80: Rejilla de ventilación y plug de alimentación del tablero. 55
Figura 81: Instalación de elementos en puerta de tablero. 56
Figura 82: Cableado y conexión del tablero. 56
Figura 83: Presentación del prototipo didáctico. 57
Figura 84: Diagrama eléctrico de fuerza 57
Figura 85: Diagrama eléctrico de control. 58
Figura 86: Sensor de temperatura en condiciones normales. 59
Figura 87: Sensor de temperatura en funcionamiento. 60
Figura 88: Sensor detector de flama en condiciones normales. 60
Figura 89: Sensor detector de flama en funcionamiento. 60
Figura 90: Lectura del sensor de presencia. 61
Figura 91: Sensor de presencia en funcionamiento. 61
Figura 92: Sensor de gas en condiciones normales. 62
Figura 93: Sensor de gas en funcionamiento. 62
Figura 94: Lectura del sensor ultrasónico en condiciones normales. 62
Figura 95: Sensor de medición de distancia en funcionamiento. 63

Figura 96: Comunicación de Arduino con Kaa IoT.....63
Figura 97: Dashboards de sensores y salida digital en Kaa IoT.....64
Figura 98: Base de datos.....64

ÍNDICE DE TABLAS

Tabla 1: Direcciones de holding register. 46
Tabla 2: Conexión del sensor de temperatura. 47
Tabla 3: Conexión del sensor detector de flama.....48
Tabla 4: Conexión del sensor de presencia. 48
Tabla 5: Conexión del sensor de gas. 48
Tabla 6: Conexión del sensor de medición de distancia.....49

INTRODUCCIÓN

El desarrollo tecnológico avanza cada vez más, generando un alto impacto en la forma en que se desarrollan los procesos del sector industrial, dando apertura al concepto de IIoT (Industrial Internet of Things) este concepto se basa en optimizar procesos de producción haciendo uso de la tecnología disponible para obtener información en tiempo real, ejecutar tareas de manera remota o automatizar actividades con la menor necesidad de la intervención humana.

Para los sectores industriales el poder aprovechar de estos conceptos tecnológicos es de mucha utilidad pues, se traducen en optimización de recursos y de tiempo, industrias como la agropecuaria se favorecería del poder monitorear aquellos factores ambientales que desfavorezcan sus cultivos, haciendo uso de sensores, los agricultores pueden supervisar estos comportamientos y tomar acciones oportunas, todo esto de manera remota.

De esta manera el presente proyecto se encamina a ser un aporte para el desarrollo tecnológico de las Pymes y poner a conocimiento las alternativas que ofrece la conectividad IIoT.

EN EL CAPITULO I, encontrará el planteamiento del problema, sus antecedentes, importancia, delimitaciones, el objetivo general y los objetivos específicos que cumplirán este proyecto.

EN EL CAPITULO II, se realizará una introducción a los temas que están involucrados en el desarrollo, se trata del marco teórico del proyecto, se da énfasis a los conceptos aplicados, se detallan las herramientas usadas, sensores, módulos, entre otros.

EN EL CAPITULO III, se detalla el método experimental y como fue utilizado para el desarrollo de este prototipo, además conoceremos en detalle cómo se diseñó e implemento este prototipo didáctico, así como las topologías de red usadas, el diseño eléctrico, los tipos de comunicaciones aplicados, entre otros.

EN EL CAPITULO IV, nos enfocamos en los resultados, en explicar lo que se obtuvo luego de la implementación e investigación del prototipo didáctico, en cómo se cumplieron todos los objetivos planteados al inicio.

EN EL CAPITULO V, se realiza un análisis de resultados, donde se explica de manera ordenada los resultados y datos obtenidos en las pruebas de funcionamiento luego de la implementación del prototipo didáctico.

Finalmente se obtiene las conclusiones y recomendaciones, donde se explica el fin y las consecuencias luego de haber desarrollado este proyecto, así como las consideraciones a tomar para futuras investigaciones.

1. CAPITULO 1: EL PROBLEMA

1.1. ANTECEDENTES

En Ecuador, las pequeñas y medianas empresas (PYMEs) se consideran una valiosa fuente de crecimiento económico. Sin embargo, también se suele pensar que las PYMEs no tienen recursos suficientes para avanzar en el desarrollo de su tecnología interna. Debido a esto, las PYMES ecuatorianas requieren programas de transferencia de tecnología para incrementar su capacidad tecnológica. [1]

Son necesarias políticas que dinamicen a las PYMEs, para que estas tengan una mejor comprensión de la innovación en sus modelos de negocio y cómo pueden agregar valor a sus productos o servicios. Dada la coyuntura, es difícil para los empresarios ver el estrecho vínculo que existe entre la innovación empresarial, la supervivencia y el crecimiento de las empresas. Tampoco conocen cómo participar en la innovación. Por ejemplo, los propietarios de las PYMEs frecuentemente desconocen hasta qué punto la digitalización puede mejorar su negocio. [2]

La emergencia global tuvo un fuerte impacto en las empresas de Colombia y Ecuador, siendo las pymes uno de los segmentos más afectados en sus ingresos y composición laboral. No obstante, este escenario permitió el desarrollo de aspectos como el uso de la tecnología y modernización de operaciones. Tras los desafíos que enfrentaron en 2020, las pymes se acercan a un proceso de recuperación, que se evidencia en sus planes de inversión y contratación. Estos son los principales hallazgos de la séptima edición de la encuesta Visión Pymes, realizada por Brother International Corporation. Esta edición del estudio, que aborda cambios fundamentales en las pymes durante el último año, fue aplicada a 776 gerentes y propietarios de negocios en Colombia y Ecuador, entre el 27 de enero y el 24 de marzo de 2021. [3]

Según lo menciona el estudio “Visión Pymes 2021” realizado por la revista Brother International Corporación, la emergencia global fue el impulso que necesitaba el sector económico de las Pymes para dar los primeros pasos en mejorar la baja gestión tecnológica que se mantenía lo que limitó el desarrollo de las mismas.

La automatización de procesos es la base originaria para el arranque de la Industria 4.0, es decir, del desarrollo de la actividad industrial con el soporte de sistemas inteligentes y robotizados que logran mejoras difíciles de lograr solo con la manufactura manual. Esto se debe a que la automatización aporta unas ventajas inigualables, sobre todo por su capacidad de realizar una misma operación de forma continuada sin interrupción, sin casi márgenes de error y trabajando siempre de forma óptima y sistematizada. Y para encabezar la innovación tecnológica en la industria, el primer paso es comprender todo lo que implica la automatización de procesos, cómo se lleva a cabo y qué soluciones tecnológicas son claves para dar un salto cualitativo y cuantitativo en la optimización de la producción. [4]

El avance tecnológico nos obliga estar preparados y estar actualizados en el tema de la industria 4.0, por este motivo el módulo didáctico es diseñado para ser un aporte para el aprendizaje de los estudiantes de la carrera de Ingeniería Electrónica.

1.2. IMPORTANCIA Y ALCANCES

La innovación de la industria en las pymes ecuatorianas está en desarrollo, por lo cual se busca aportar y aprovechar la necesidad de integrar las nuevas tecnologías en beneficio de los diferentes sectores industriales que estas conforman. Además, al hablar del aporte que brinda las tecnologías IIoT es necesario estar preparado en conocimiento técnico para poder brindar soluciones en esta área, lo que involucra directamente a los estudiantes de la Universidad Politécnica Salesiana que durante su formación académica y profesional obtienen la experiencia necesaria para atender y solucionar problemas que abarquen este tipo de tecnologías.

El presente trabajo de titulación brinda una alternativa para mejorar el acceso a las herramientas digitales que ofrecen las tecnologías IoT e IIoT para favorecer el desarrollo de las pymes, de la misma forma pretende ser un aporte al aprendizaje de los estudiantes de la carrera de ingeniería electrónica, pues se entregará un módulo didáctico el cual podrá ser utilizado para ejecutar prácticas de laboratorio y enriquecer el aprendizaje de los estudiantes en diversas áreas académicas.

1.3. DELIMITACIÓN

1.3.1. TEMPORAL

El Proyecto de Titulación se diseñó y desarrolló en la ciudad de Guayaquil, Ecuador para un periodo máximo de 1 año a partir de su aprobación.

1.3.2. ESPACIAL

Debido a la emergencia sanitaria por motivo de la pandemia mundial del COVID-19, el desarrollo y pruebas del proyecto se llevó a cabo en el domicilio del autor ubicado al norte de la ciudad de Guayaquil-Ecuador. Adicionalmente, este módulo será entregado y puesto a disposición de los laboratorios de electrónica de la Universidad Politécnica Salesiana sede Guayaquil para su uso en favor de los estudiantes de la carrera.

1.3.3. ACADÉMICA

Se aplicaron conocimientos técnicos, prácticos y empíricos adquiridos durante el transcurso de la carrera de Ingeniería Electrónica, enfatizando conocimientos en Redes de computadoras, Automatización Industrial, Comunicaciones Industriales, Programación de sistemas embebidos, entre otros.

1.4. EXPLICACIÓN DEL PROBLEMA

El desarrollo tecnológico de las Pymes está relacionado directamente al conocimiento y la capacidad que estas poseen para aprovechar las tecnologías vigentes para obtener beneficios que aporten el desarrollo de las mismas, entre los

desarrollos tecnológicos que generan mayor impacto en la producción resalta la automatización de procesos y el monitoreo en tiempo real, que tienen como objetivo el aumento de la productividad, mejorar los tiempo de respuesta en procesos, mejorar la calidad del proceso, entre otros. Adicionalmente, se conoce que durante el desarrollo de un proceso de producción intervienen factores que influyen el resultado del mismo, estos son físicos y/o ambientales, que al no ser previstos o monitoreados de una manera correcta pueden representar un alto riesgo de producir resultados desfavorables para los procesos que se estén ejecutando, por ende, en la gran mayoría de casos, estos ocasionan inconvenientes e interrupciones no deseadas que desencadenan pérdidas económicas significativas para el desarrollo de las Pymes.

El automatizar y monitorear en tiempo real un proceso de producción, llega a involucrar costos no deseados, como lo son la adquisición de las licencias de softwares, renovación de licencias u otros complementos, inclusive comprar otros dispositivos que según la marca comercial de estos se define el costo del mismo, por ello muchas de las Pymes en desarrollo optan por dejar a un lado las herramientas de desarrollo tecnológico debido a que los costos llegan a superar sus capacidades adquisitivas, limitando su desarrollo.

Adicional, para la Universidad Politécnica Salesiana como ente formador de profesionales de calidad en la carrera de ingeniería electrónica, al cumplir con el objetivo de preparar profesionales con conocimientos solidos de las tecnologías emergentes desarrolla la necesidad de poseer módulos didácticos donde los estudiantes puedan relacionarse con estas tecnologías, practicar, desarrollar su aprendizaje y poner a prueba sus conocimientos, para así obtener experiencia y competencias necesaria para su formación profesional.

1.5. OBJETIVOS

1.5.1. OBJETIVO GENERAL

Diseñar e implementar un prototipo didáctico para el monitoreo de sensores a través de una plataforma IoT mediante protocolos de comunicación MQTT y MODBUS utilizando sistemas embebidos como Arduino y Raspberri pi, que nos permita acceder y monitorear los datos de un ambiente especifico de manera online desde la plataforma IoT.

1.5.2. OBJETIVOS ESPECÍFICOS

- Diseñar e implementar un sistema de sensores con Arduino.
- Configurar una base de datos para recopilar la información del sistema de sensores.
- Acceder a la información del sistema de sensores desde la plataforma IoT.
- Crear dashboards para presentarlos dentro de la plataforma IoT.

2. FUNDAMENTOS TEÓRICOS

2.1. INTERNET DE LAS COSAS (IOT)

En [5] definen a el Internet de las cosas como la interconexión digital de varios dispositivos inteligentes y plataformas de gestión que habilitan colectivamente el "mundo inteligente" del entorno, con la clara idea de una conexión en cualquier lugar, en cualquier momento y con cualquier objeto. Teniendo como punto de partida una compleja red, la cual conecta millones de dispositivos y personas en arquitecturas multi-tecnologías, multi-protocolo y multiplataforma, coincidiendo todos estos en un entorno inteligente. Logrando conectar cualquier aparato o dispositivo electrónico indiferentemente del ámbito en el que se encuentre una persona, pudiendo ser académico, social, en el hogar, en la industria, entre otros.

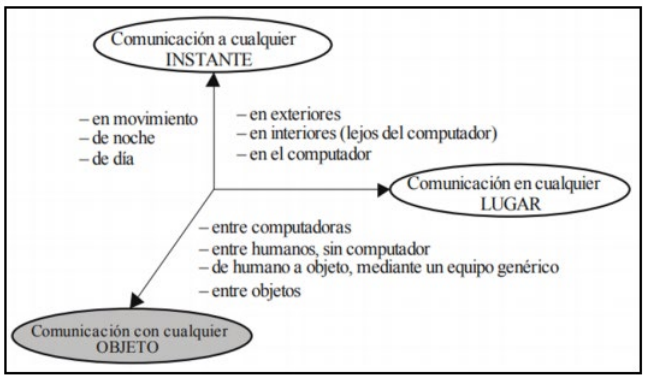


Figura 1: Nueva dimensión que introduce IoT. *[5]*

2.1.1. PLATAFORMA IOT

Una plataforma IoT es una plataforma web integrada, es decir, un software que conecta hardware, puntos de acceso y redes de datos. Es la base para que múltiples dispositivos se conecten entre sí en un mismo entorno o ecosistema propio. Estas plataformas suelen ser la aplicación con la que el usuario interactúa, de esta forma lo define [6].

2.2. INTERNET INDUSTRIAL DE LAS COSAS (IIOT)

Cuando hablamos de IIoT (Industrial Internet of Things) nos referimos a dispositivos inteligentes, tales como sensores, actuadores y dispositivos controladores como PLC, IED o RTU, los mismos que dentro de una red interconectada y monitorizada, nos permiten ver en tiempo real los valores de los parámetros, dejándonos tener un mayor control del fallo de los mismos.

2.2.1. LA MEJORA DE IIOT EN ENTORNOS INDUSTRIALES

El enfoque de IIoT va direccionado a optimizar la producción industrial, a través de la conexión de dispositivos y un centro de datos, con el fin de tener un mejor control de la producción industrial y sus procesos, del estado y mantenimientos de las maquinarias y de estadísticas en tiempo real, de una manera más sencilla y segura. A medida que pasa el tiempo el entorno industrial se va digitalizando cada vez más, por lo cual IIOT ha evolucionado para adaptarse a los sistemas de control.

2.2.2. CARACTERÍSTICAS DE LOS DISPOSITIVOS IIOT

Debido a su capacidad de adaptarse a una amplitud de lugares independiente del sector al que pertenezcan, los dispositivos IIoT cuentan con las siguientes características físicas mencionadas en [7]:

- Dispositivos más resistentes, para que puedan funcionar en ambientes hostiles y condiciones extremas tales como altas y bajas temperaturas, ambientes corrosivos, sin presentar averías o daños.

- Dispositivos dotados de una mayor autonomía al integrar baterías de gran capacidad y un menor consumo de energía, debido al coste que puede suponer mandar a un operario a realizar su mantenimiento o la complejidad de acceder al dispositivo de forma física. Tener conectados los dispositivos a la red permite monitorizarlos, pudiendo observar y controlar de forma remota la vida útil del dispositivo, consultar logs, realizar cambios en la configuración o actualizarlos.

- Sistema de seguridad propio de dispositivos IoT y con un grado de robustez mayor, ya que un ciberataque podría ser fatal en una infraestructura crítica debido a la interconectividad de sus redes.

2.2.3. VENTAJAS DE LOS DISPOSITIVOS IIOT

Los dispositivos IIoT cuentas con una serie de ventajas debido al crecimiento de los mismos en proporcionar una mayor inteligencia a los procesos industriales y a las técnicas específicas para el manejo de datos, a continuación, se mencionan algunas de esas ventajas [7]:

- Mejorar la eficiencia energética, mantenimiento predictivo y preventivo de las máquinas, así de forma más rápida poder obtener los datos de los sensores y otros dispositivos IoT para realizar un monitoreo constante y poder prevenir posibles fallos.

- Mejorar la conectividad respecto a las redes de comunicaciones, las mismas que son necesarias entre el dispositivo y el sistema IoT dentro del cual opera. En este punto podemos encontrar varias partes:

- Comunicaciones en tiempo real, para lo cual es esencial contar con una conectividad continua, además de poder almacenar los datos obtenidos para volver a revisarlos, si fuese necesario.

- Iniciación de las comunicaciones de manera segura para que no haya ninguna posible intrusión por parte de un tercero.

- Seguridad de enlaces: se centra en el nivel de seguridad y confianza que implica el establecimiento y el funcionamiento de la conectividad.

- Uso de la nube para almacenar datos y facilidad de acceso mediante otros dispositivos para acceder a ellos.

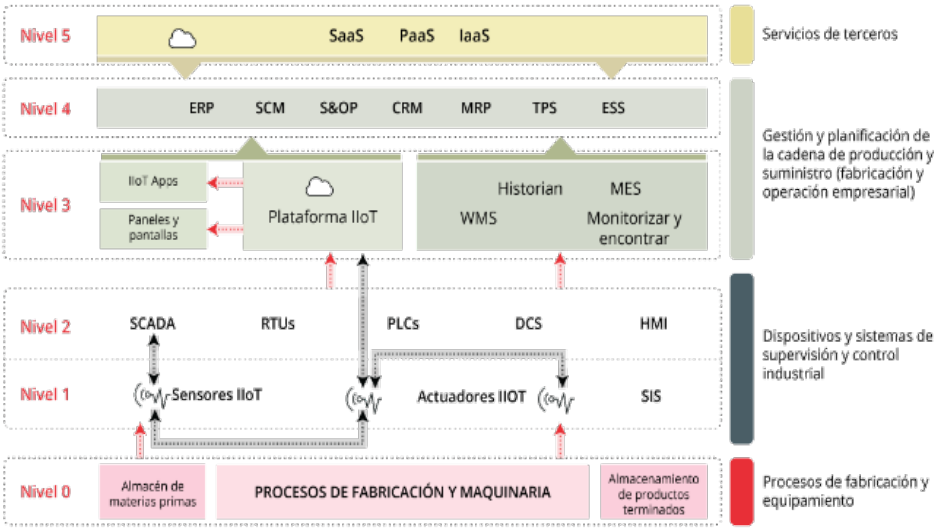


Figura 2: Modelo de referencia de alto nivel. *[7]*

2.3. MQTT

MQTT (Message Qeue Telemetry Trasnport), es un protocolo de comunicación machine-to-machine, protocolo de capa aplicación tanto de publicación como suscripción del servidor al cliente, el cual fue creado por el Dr. Andy Stanford Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en el año 1999 como un mecanismo para conectar dispositivos dentro de la industria petrolera. Fue estandarizado en 2014 según OASIS (Organization for the Advancement of Structured Information Standards). Este protocolo es ideal para IoT debido a que se destaca por ser liviano, simple y de fácil implementación.

2.3.1. FUNCIONAMIENTO DE MQTT

El funcionamiento del MQTT es un servicio de mensajería push con patrón publicador/suscriptor (pub-sub). En este tipo de infraestructuras los clientes se conectan con un servidor central denominado bróker.

Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el bróker le hará llegar los mensajes suscritos véase Figura 2.

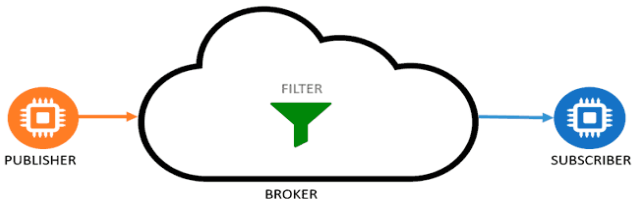


Figura 3: Comunicación entre bróker y clientes. *[8]*

Los clientes inician una conexión TCP/IP con el bróker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS.

Para ello el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id…). El bróker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc.) véase Figura 3.

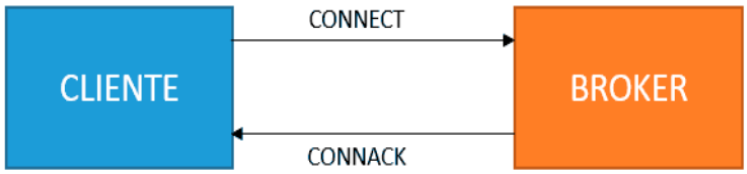


Figura 4: Mensajes CONNECT y CONNACK. *[8]*

Para enviar los mensajes el cliente emplea mensajes PUBLISH, que contienen el topic y el payload.

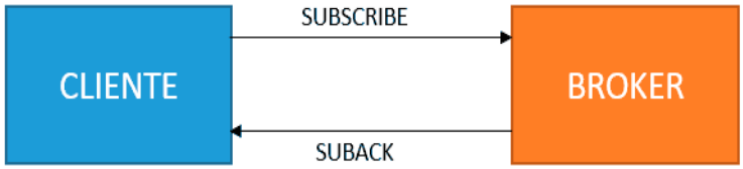


Figura 5: Mensajes SUSCRIBE y SUBACK. *[8]*

Para suscribirse y desuscribirse se emplean mensajes SUBSCRIBE y UNSUSCRIBE, que el servidor responde con SUBACK y UNSUBACK, véase Figura 4.

Por otro lado, para asegurar que la conexión está activa los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP. Finalmente, el cliente se desconecta enviando un mensaje de DISCONNECT. [8]

2.4. TCP/IP

En [9] se define a TCP/IP como "la identificación del grupo de protocolos de red que hacen posible la transferencia de datos en redes, entre equipos informáticos e internet." Este protocolo está basado en el modelo teórico OSI de capas y está compuesta por 4 de ellas. Este protocolo es un modelo practico para la comunicación en redes permitiendo que un equipo se pueda comunicar dentro de una red.

El protocolo TCP/IP apareció en 1969 en el proyecto de defensa DARPA, y fue estandarizado en 1983, convirtiéndose así en el protocolo más usado en redes y el protocolo estándar de internet.

Este protocolo se referencia en los siguientes protocolos:

- **TCP**, protocolo de Control de Transmisión el cual permite establecer conexión e intercambiar datos entre dos usuarios.

- **IP,** también llamado protocolo de internet, es el encargado de llevar los datos a otros dispositivos en red utilizando direcciones.

2.4.1. CAPAS DEL MODELO TCP/IP

Este protocolo cuenta con cuatro capas o niveles, véase figura 6, cada una de ellas son fundamentales para la transmisión de información en una red. El usuario interactúa solo con 2 de ellas, la Capa Física esta es la primera de sus capas y con

la capa de aplicación la cual es la última de ellas. Para poder entender un poco sobre cada una de ellas, se procede a dar una breve explicación de cada una de ellas.

Figura 6: Capas o niveles del modelo TCP/IP. *[10]*

2.4.1.1. CAPA DE ACCESO A RED O FÍSICA

Capa de red acceso a red también llamada capa física es la primera capa o novel del modelo TCP/IP, es la capa que define las características del hardware que se utilizará en la red, es decir, es la que define la interfaz mecánica, eléctrica y de temporización de la red.

La capa física de TCP/IP describe los estándares de hardware como IEEE 802.3, la especificación del medio de red Ethernet, y RS-232, la especificación para los conectores estándar. [11]

2.4.1.2. CAPA DE RED O INTERNET

Es una de las capas más importantes, es la encargada de administrar las direcciones IP y de proporcionar el paquete de datos, es decir, acepta y transfiere paquetes para la red, encargándose de enrutar los datos desde el emisor hacia el receptor, buscando el mejor camino para transportar los paquetes desde su origen hasta su destino.

2.4.1.3. CAPA DE TRANSPORTE

Esta capa es la que garantiza que los paquetes lleguen sin errores y en secuencia en cada punto de conexión de la comunicación, permitiendo conocer el estado de la transmisión y enrutamiento de los datos, de un nodo a otro, este tipo de conexión es llamado transmisión de punto a punto.

2.4.1.4. CAPA DE APLICACIÓN

Es la última de las capas o niveles del protocolo TCP/IP, esta capa es la que define y suministra las aplicaciones de red y los servicios de internet estándar que puede utilizar el usuario, trabajando directamente con el mismo.

2.5. MODBUS

MODBUS fue desarrollado por Modicon (ahora parte de Schneider Electric) en 1979 como un medio de comunicación con varios dispositivos sobre un simple cable par trenzado. Es un sistema “maestro-esclavo”, donde el “maestro” se comunica con uno o múltiples “esclavos”. El “maestro” generalmente es un Programador Lógico Programable (PLC), Computadora Personal, Sistema de Control Distribuido (DCS) o Unidad Terminal Remota (RTU) véase Figura 5.

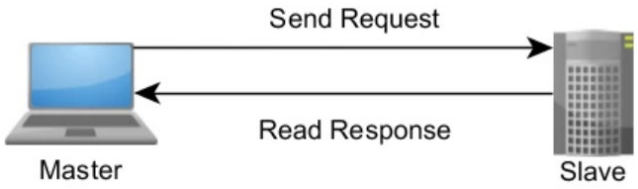


Figura 7: Relación de Red maestro-esclavo. *[12]*

Cuando el dispositivo “maestro” requiere información de un dispositivo, este envía un mensaje el cual contiene la dirección del dispositivo, la solicitud del dato y una suma de chequeo para detección de error. Todos los dispositivos que se encuentran en la red reciben el mensaje, pero únicamente responde el dispositivo al que fue direccionado, solo los dispositivos maestros pueden iniciar la comunicación, los dispositivos esclavos solo responden. Algunos fabricantes están desarrollando dispositivos híbridos que actúan como esclavos, pero al mismo tiempo tienen la capacidad de escritura, lo que los convierte en “seudo maestros”.

Todos los mensajes de MODBUS se envían en el mismo formato. La única diferencia entre los tres tipos es la manera como se codifican los mensajes [13]. A continuación, se mencionarán a las tres versiones más comunes de MODBUS que son utilizados hoy en día, las cuales son:

- MODBUS ASCII
- MODBUS RTU
- MODBUS/TCP

2.5.1. MODBUS TCP/IP

Modbus TCP/IP es un protocolo de comunicación de la familia MODBUS, el cual se basa específicamente en el uso de mensajería Modbus con una interfaz TCP que se ejecuta en internet, el uso más común de este protocolo se lo da en el entorno industrial y automatización para la conexión Ethernet de los PLCs, módulos de E/S y puertas de enlace a otros buses de campo o redes de E/S simples, según los explica [14].

Dentro de esta estructura, Modbus el cual es el protocolo de aplicación, es el que define las normas para la interpretación y la organización de los datos. En lo que se refiere a TCP/IP el cual es el protocolo de internet y protocolo de control de la

transmisión, se encarga de proporcionar el medio de transmisión de los mensajes o datos. Es decir, TCP se asegura de que los paquetes de datos se reciban correctamente mientras que IP se asegura que los mensajes se aborden y coloquen correctamente.

En [15] lo explican de la siguiente manera “Modbus TCP / IP utiliza TCP / IP y Ethernet para transportar los datos de la estructura del mensaje Modbus entre dispositivos compatibles. Es decir, Modbus TCP / IP combina una red física (Ethernet), con un estándar de red (TCP / IP), y un método estándar de representación de datos (Modbus como el protocolo de aplicación). En esencia, el mensaje Modbus TCP / IP es simplemente una comunicación Modbus encapsulado en una envoltura de Ethernet TCP / IP.”

2.6. NODE-RED

Node-RED es una herramienta de programación visual donde muestra visualmente las relaciones y funciones al usuario sin tener que aprender un nuevo lenguaje de programación. [16]

Node-RED es una plataforma, un entorno grafico para construir aplicaciones JavaScript, fue diseñado por ingenieros de IBM. Tiene como objetivo principal, conectar dispositivos de hardware, API y servicios en línea de formas nuevas e interesantes.

En [17] explican que Node-RED es Open Source (código abierto), que esta herramienta simplifica el desarrollo de aplicaciones orientadas al manejo de eventos asíncronos, como por ejemplo las aplicaciones IoT.

Esta plataforma corre en Node.JS para lo cual debe estar instalada, proporciona un editor basado en navegador que facilita la interconexión de flujos utilizando una amplia gama de nodos de la paleta que se pueden implementar en su tiempo de ejecución, con un solo clic [18]. Dentro de la interfaz hay bloques de entrada, de salida y de procesamiento de información o datos.

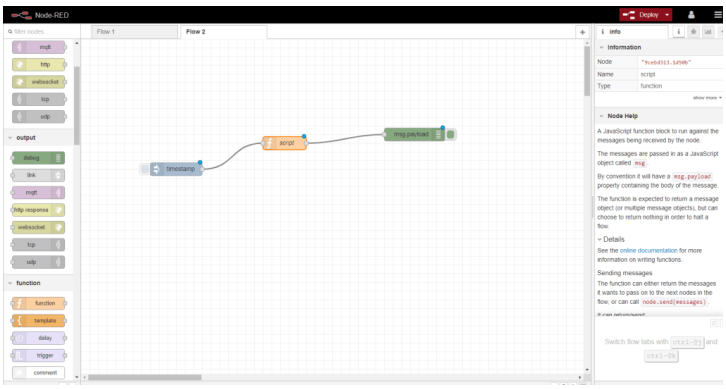


Figura 8: Interfaz Web de Node-RED. *[17]*

Como podemos observar en la Figura 8, del lado izquierdo hay un panel donde se encuentran todos los bloques que vienen por defecto, los mismo que se conectan entre sí, de esta manera el usuario genera el comportamiento deseado para su aplicación.

2.7. BASE DE DATOS

Una base de datos es una colección de datos almacenados y organizados de formas que un programa del ordenador pueda seleccionarlos rápidamente y capaces de ser: recobrados, actualizados, insertados y borrados. En un DBMS una base de datos es un sistema de archivos electrónicos. [19]

Este método de almacenamiento de datos, se ha convertido en el favorito de todos, desde un usuario común hasta grandes empresas.

2.8. MARIADB

Es un sistema de gestión de base de datos creado por Michael Widenius, que tiene como característica principal el basarse en modelos de software libre, además de ser potente y flexible; por lo cual, se ha convertido en uno de los más utilizados.

Es un sistema de gestión de datos derivado de MySQL, es decir, aquellos proyectos o programas que son compatibles con MySQL serán compatibles con MariaDB.

Entre las múltiples características interesantes que posee este sistema de gestión de base de datos están:

- Su rapidez al momento de realizar consultas.
- Diversas extensiones, mejorando sus funcionalidades.
- Corrección de errores que ocurren en la gestión de base de datos.
- Acceso a toda la información para utilizar este gestor.
- Posee licencia GLP, es decir, es un sistema de software libre.

2.9. PHPMYADMIN

Es un software de código abierto, es una herramienta gratuita diseñado para manejar la administración y gestión de bases de datos MySQL a través de una interfaz gráfica de usuario, este software cuenta con un conjunto de archivos escritos en PHP, convirtiéndose en uno de las herramientas más populares basadas en web de gestión de base de datos. [20]

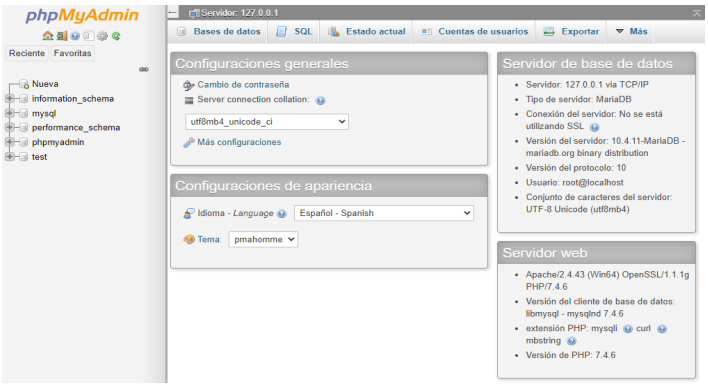


Figura 9: Interfaz gráfica de phpMyAdmin. *[20]*

En la figura 9 podemos observar la interfaz gráfica del software phpMyAdmin; en la parte izquierda se encuentran todas las bases de datos que están en el servidor, en la parte superior el menú de navegación, en la parte central las opciones generales, de idioma y apariencia y en la parte derecha la información del servidor de la base de datos.

2.10. ARDUINO

Arduino es una plataforma de electrónica de hardware libre, utilizada idealmente en proyectos o prototipos de electrónica. Fue creada en el año 2005 y llamada así en honor al Rey Arduino de Ivrea.

Arduino es un software de código abierto, es una placa implementada con el microcontrolador ATMEGA8 y ATMEGA168 de Atmel, fue orientado al uso de estudiantes para que puedan realizar sus prototipos.

El funcionamiento de ARDUINO está basado en la lectura de sensores dentro de un ambiente siendo estas sus entradas y los actuadores sus salidas los cuales convierten las señales eléctricas en magnitudes físicas como display, motores, relés, luces, entre otros.

2.10.1. MODELOS DE ARDUINO

Arduino tiene diferentes modelos, los cuales se mencionan los más relevantes a continuación:

- Arduino Nano
- Arduino UNO
- Arduino Mega 2560
- Arduino Mega ADK
- Arduino Leonardo

- Arduino DUE
- Arduino Micro
- Arduino YUN
- Arduino FIO

A continuación, se realiza una breve descripción de ARDUINO MEGA 2560, el mismo que será utilizado para el desarrollo del presente proyecto técnico.

2.10.1.1. ARDUINO MEGA 2560

Este Arduino posee un microcontrolador ATMega2560, cuenta con 54 entradas/salidas digitales, de las cuales 15 pueden usarse como PWM, 16 entradas analógicas y 4 UART, véase Figura 10. Arduino MEGA 2560 es compatible con todos los shields de Arduino. Se conecta al ordenador mediante un cable USB. [21]

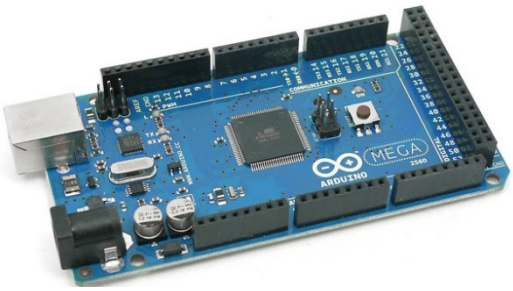


Figura 10: Arduino MEGA 2560. *[22]*

2.10.2. ARDUINO ETHERNET SHIELD

Arduino Ethernet Shield es un módulo que permite a la placa Arduino conectarse a una red ethernet de una manera rápida, este módulo se conecta a la placa Arduino mediante un cable RJ45.

Este módulo es compatible con las placas Arduino Uno y Arduino Mega, se basa en un chip de ethernet Wiznet W5100, el cual, ofrece una red IP que usa TCP y UDP. También posee una ranura para Tarjetas micro-SD para el almacenamiento de archivos [23], véase Figura 11.

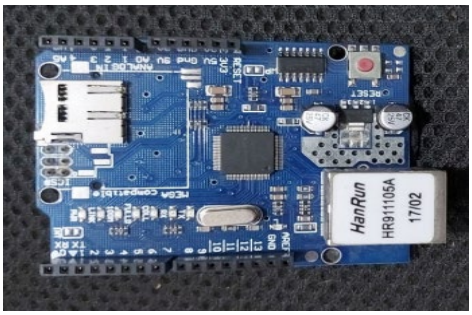


Figura 11: Módulo Arduino Shield Ethernet.

2.11. RASPBERRI PI

Raspberri es un computador pequeño del tamaño de una tarjeta de crédito, posee un sistema operativo Linux. Esta placa de microordenador tiene como objetivo realizar las mismas tareas que cualquier ordenador, soportando varios dispositivos periféricos de un ordenador común, por ejemplo, teclados y mouses. Puede realizar tareas comunes como hojas de cálculo, juegos, reproducción de videos, etc.

Este proyecto fue ideado en 2006 pero no fue lanzado al mercado febrero de 2012. Ha sido desarrollado por un grupo de la Universidad de Cambridge y su misión es fomentar la enseñanza de las ciencias de la computación los niños. [24]

Esta placa posee al menos con un puerto de video HDMI, un puerto RJ45, dos puertos USB 2.0 y un mini Jack de audio. Solo requiere una memoria SD de desde 8 a 128 GB. Las versiones anteriores constaban de una alimentación vía puerto Micro USB, la versión más actual (Raspberri Pi 4) cuenta con una alimentación puerto USB-C.

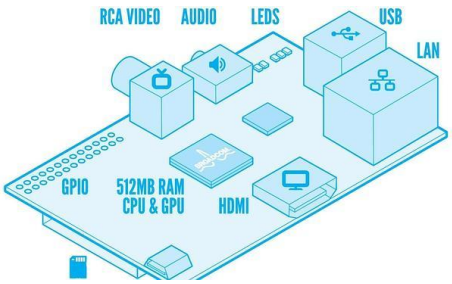


Figura 12: Diagrama de Raspberri Pi. *[25]*

Raspberri Pi a través de los años ha ido sacando versiones, mejorando cada una de sus características. Para el desarrollo de este proyecto se utilizará la Raspberri Pi 4.

2.11.1. RASPBERRI PI 4

Raspberri Pi 4 es la primera Raspberri Pi de una generación que admite más RAM, su rendimiento de CPU, GPU y E/S tiene una mejoría radical. La Raspberri Pi 4 se encuentra disponible con 1, 2 y 4 GB de LPDDR4 SDRAM.

En este caso se utilizará la Raspberri Pi 4 de 4 GB de LPDDR4 SDRAM, cuenta con las siguientes especificaciones [26]:

- Broadcom BCM2711, SoC de 64 bits Cortex-A72 de cuatro núcleos (ARM v8) a 1,5 GHz
- SDRAM LPDDR4-3200 de 2GB, 4GB u 8GB (según el modelo)
- 2.4 GHz y 5.0 GHz IEEE 802.11ac inalámbrica, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 puertos USB 3.0; 2 puertos USB 2.0.
- Cabecera GPIO estándar Raspberry Pi de 40 pines (totalmente compatible con las placas anteriores)
- 2 × puertos micro-HDMI (hasta 4kp60 compatible)
- Puerto de pantalla MIPI DSI de 2 carriles
- Puerto de cámara MIPI CSI de 2 carriles
- Puerto de video compuesto y audio estéreo de 4 polos
- H.265 (decodificación 4kp60), H264 (decodificación 1080p60, codificación 1080p30)
- Gráficos OpenGL ES 3.0
- Ranura para tarjeta micro-SD para cargar el sistema operativo y el almacenamiento de datos
- 5 VCC a través del conector USB-C (mínimo 3 A *)
- 5 VCC a través del encabezado GPIO (mínimo 3 A *)
- Power over Ethernet (PoE) habilitado (requiere PoE HAT separado)
- Temperatura de funcionamiento: 0 - 50 grados C ambiente

En la Figura 13 podemos observar algunas de sus características físicas.

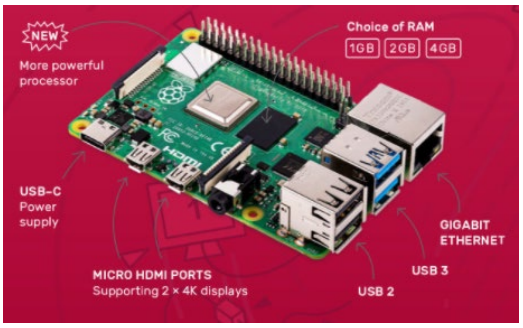


Figura 13: Características de Raspberri Pi 4. *[27]*

La Raspberri Pi 4 conserva mucho de los principios fundamentales de las versiones anteriores, la gran parte de las mejoras y cambios de Raspberri Pi 4 en comparación con sus antecesoras son a nivel interno.

2.12. I2C

Su nombre viene del inglés Inter-Integrated Circuit, es un bus de comunicaciones en serie, fue diseñado por Philips en 1982. Es muy utilizado en el ámbito industrial para comunicar controladores y sus periféricos en sistemas integrados.

Es un protocolo síncrono. I2C usa solo 2 cables, uno para el reloj (SCL) y otro para el dato (SDA). Esto significa que el maestro y el esclavo envían datos por el mismo cable, el cual es controlado por el maestro, que crea la señal de reloj. I2C no utiliza selección de esclavo, sino direccionamiento. [28]

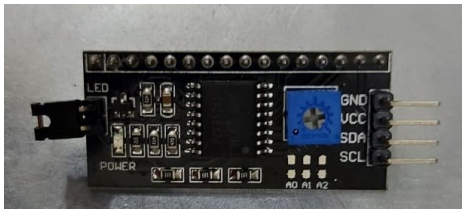


Figura 14: Integrado I2C.

También posee una tercera línea (GND) la cual es la de referencia, véase la Figura 14.

2.12.1. I2C EN ARDUINO

Arduino dispone de soporte para I2C, para poder utilizarlo en Arduino, el IDE Standard proporciona la librería “Wire.h”, para poder controlar este hardware integrado y permite comunicar a Arduino con otros dispositivos. Arduino posee ciertos pines para vincularse con I2C, estos pines varían dependiendo del modelo del Arduino que se utilice. A continuación, en Figura 15 podemos ver un ejemplo de conexión entre I2C y Arduino.

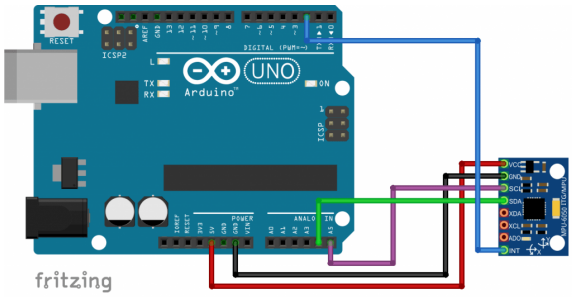


Figura 15: Esquema de conexión de Arduino UNO y el I2C MPU6050. *[28]*

2.13. FUENTE DE ALIMENTACIÓN MEAN WELL MDR 60-12 AC a DC

Este dispositivo hardware tiene como función principal convertir la corriente alterna (AC) en corriente continua (DC). Trabaja con un voltaje de entrada máximo de 230 VAC y con un voltaje de entrada mínimo de 115 VAC

Su voltaje de salida es de 12 VDC, con una corriente salida de 5A y potencia máxima de salida de 60W.

Figura 16: Fuente de alimentación Mean Well MDR-60-12. *[29]*

2.14. PORTAFUSIBLE EBCHQ-36900

Es un dispositivo encargado de alojar en su interior el fusible, su montaje es para riel, posee un rango de voltaje de hasta 500V y un rango de corriente de hasta 32A.

Figura 17: Disyuntor Merlin Gerin Multi 9 C60N.

2.15. TOMACORRIENTE

Es un elemento de uso eléctrico también conocido como enchufe hembra, su principal función es establecer una conexión eléctrica con un enchufe macho y así permitir la circulación de corriente. Los tomacorrientes usualmente van ubicados en la pared o superficies planas ya sea empotrados o superficialmente.

Figura 18: Tomacorriente.

2.16. SELECTOR ELÉCTRICO

También llamado conmutador eléctrico, este dispositivo se utiliza para interrumpir el paso de la corriente eléctrica, su accionamiento es manual. Tiene un gran uso en el campo de control. Suele tener una palanca giratoria, la cual permitirá posicionarla en 2 o más posiciones, dependiendo del uso, requerimientos o aplicaciones.

Figura 19: Selector eléctrico de 2 posiciones.

2.17. LUCES PILOTO

Son elementos de control y señalización las cuales tienen como única función darnos un aviso o advertencia visual, desde el encendido o apagado de un equipo o tablero eléctrico hasta para indicar alguna parte del proceso.

Figura 20: Luces pilotos.

2.18. LCD 4x20

También conocida como display LCD 4x20, es una pantalla monocromática de cristal líquido, la cual permite visualizar 20 caracteres alfanuméricos por 4 líneas. Su voltaje de operación es de 5VDC

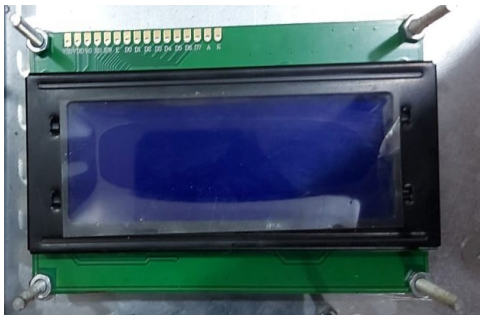


Figura 21: LCD 4x20.

2.19. TP-LINK WR741ND

Es un router inalámbrico, con una velocidad de datos de 150 Mbps, este dispositivo combina la conexión de red de cable/inalámbrico de un router permitiéndonos compartir internet y la de un switch de 4 puertos.

Figura 22: Router TP-LINK WR741ND.

2.20. REGULADOR DE VOLTAJE LM2596

Es un módulo regulador de voltaje ajustable DC a DC de alta eficiencia de conversión, capaz de transformar un nivel de voltaje a otro de mayor o de menor nivel. Tiene un voltaje de entrada de 4.5VDC a 40VDC, un voltaje de salida de 1.23VDC a 37VDC y soporta una salida hasta de 3A.

Figura 23: Módulo regulador de voltaje ajustable.

2.21. SENSOR DE TEMPERATURA DS18B20

Es un sensor de temperatura versátil, nos permite medir temperaturas desde los -55°C hasta 125°C, además, permite sumergirlo en liquido dado que se encuentra sellado en un envoltorio estanco.

Este sensor digital posee una resolución de 9 a 12 bits los cuales son configurables, utiliza el protocolo 1-Wire para comunicarse y tiene una alimentación de 3 V a 5.5 V.

Figura 24: Sensor de temperatura DS18B20.

2.22. SENSOR DE FLAMA YG1006

Es un sensor de flama o llama, posee un fototransistor NPN YG1006 que es sensible a la luz infrarroja, por lo cual nos permite detectar el fuego. El nivel de sensibilidad se puede regular mediante su potenciómetro.

Tiene un voltaje de operación de 3.3VDC hasta 5 VDC, trabaja a una temperatura de -25°C hasta 85°C, posee una salida digital y una salida analógica.

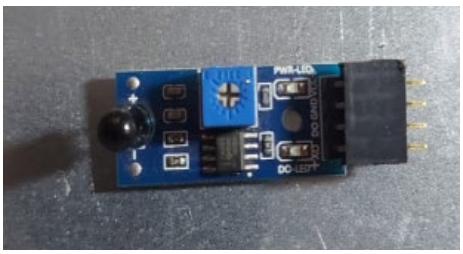


Figura 25: Sensor de flama YG1006.

2.23. SENSOR DE PRESENCIA HC-SR501

Es un sensor infrarrojo pasivo utilizado para la detección de movimiento, se basa en la medición de la radiación infrarroja mediante un sensor piro eléctrico el cual capta la radiación emitida por todo cuerpo sea este un ser vivo o no y la convierte en una señal eléctrica.

Este sensor es de baja potencia y de fácil uso, lo cual, lo hace ideal para aplicaciones de domótica y sistemas de seguridad.

Figura 26: Sensor HC-SR501.

2.24. SENSOR DE GAS MQ-135

Es un sensor de control de calidad de aire, se basa en la detección de gases contaminantes en el medio ambiente como lo son amoniaco, alcohol, benceno, humo y dióxido de carbono. Usualmente lo utilizan en lugares donde se manejan compuestos químicos nocivos para la salud.

Posee una salida analógica y una digital, su voltaje de operación es de 5VDC y su potencia de salida es de 800mW.

Figura 27: Sensor de control de calidad de aire MQ-135.

2.25. SENSOR ULTRASÓNICO HC-SR04

Es un sensor ultrasónico que permite medir distancias mediante ultrasonidos en un rango de entre 2 cm a 400 cm, a diferencia de los sensores infrarrojos, este sensor no se ve afectado por material de color negro o por la luz solar.

Está compuesto por un emisor de ultrasonidos y por un módulo receptor, el pin “Tigger” se comunica con el emisor de ultrasonidos, mientras el pin “Echo” se encuentra comunicado con el módulo receptor, el cual recibe el eco del ultrasonido enviado.

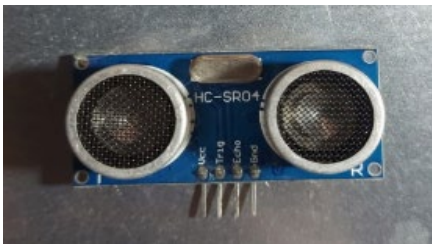


Figura 28: Sensor ultrasónico HC-SR04.

2.26. BUZZER

También conocido como zumbador, es un transductor piezoeléctrico que tiene la capacidad de convertir una señal eléctrica en una señal de audio, su voltaje de operación es de 5VDC.

Se clasifican en buzzers activos y pasivos; los buzzers activos solo requieren ser conectados a una tensión DC para generar el sonido, mientras los buzzers pasivos requieren que se les proporcione una señal eléctrica u onda de la frecuencia para generar el sonido.

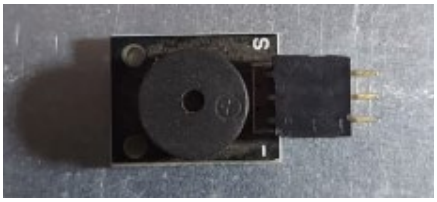


Figura 29: Buzzer activo.

2.27. CAMARA WEB

Es una pequeña cámara digital, la cual es capaz de capturar imágenes y transmitirlas a través de internet; para poder realizar estas acciones necesita de una computadora.

Figura 30: Cámara web.

2.28. TABLERO PLÁSTICO ELÉCTRICO

Está fabricado de plástico de alta dureza, además es resistente al impacto IK-08, tiene protección a la corrosión y UV, su grado de protección es IP65 y posee certificación inglesa. Sus dimensiones son 600mmx400mmx200mm, incluye placa metálica galvanizada.

Figura 31: Tablero plástico eléctrico.

3. FUNDAMENTOS METODOLÓGICOS

3.1. MÉTODO DE ESTUDIO

3.1.1. METODO EXPERIMENTAL

En el desarrollo de este proyecto se utilizó el método experimental, se realizaron diferentes pruebas entre los dispositivos, placas, módulos y plataformas, como por ejemplo las interconexiones y la comunicación entre algunos de ellos, utilizando los protocolos MQTT y MODBUS TCP/IP.

Como punto de partida de este prototipo, se comprobó la comunicación de los sensores, los mismos que se conectaron en Arduino Mega 2560, de los cuales se obtuvo información y fue enviada a la plataforma Node-RED, se verificó que la transmisión de datos sea exitosa. Luego se definió el código a usar en Arduino como por ejemplos sus librerías, canales MODBUS, entre otros.

Se estableció la comunicación de Node-RED con la plataforma “Kaa IoT”, en la plataforma “Kaa IoT” se configuraron los cuadros de presentación en donde se visualizaron los datos enviados de los sensores.

Para escoger la plataforma IoT a usar, se tomó en consideración varios criterios, entre los principales que sean gratuitas y de menores limitaciones, además, que vayan de acorde a las necesidades del prototipo.

La información recibida y visualizada en la plataforma Kaa IoT es almacenada en una base de datos local creada en la Raspberri Pi 4.

El prototipo cuenta con un router, el mismo que fue configurado con la finalidad de crear una red de uso propio del prototipo, de esta manera evitar conflictos de red, interferencias y fallas de comunicación. Agregándole un nivel de seguridad a la red.

3.2. DESCRIPCIÓN DEL PROTOTIPO

El prototipo didáctico para el monitoreo de sensores a través de una plataforma IoT utilizando protocolos de comunicación MQTT y MODBUS TCP/IP, está implementado en un tablero plástico eléctrico de dimensiones 600mmx400mmx200mm con una placa metálica galvanizada como plafón, la misma en donde están instalados todos nuestros dispositivos de control, fuerza y protecciones.

Los sensores que se utilizaron son: sensor de temperatura DS18B20, sensor de flama YG-1006, sensor de presencia HC-SR501, sensor de gas MQ-135, sensor ultrasónico HC-SR04. Cada sensor cuenta con un switch, lo que permite activar o desactivar los sensores a conveniencia y poder ver su funcionamiento individual o grupalmente.

En la parte frontal del tablero tenemos dos luces pilotos, que indican el estado de la comunicación en la red creada, un selector que energiza el microcontrolador Arduino Mega 2560 y una LCD 4x20 que muestra información relevante a la comunicación. Esta LCD 4x20 se conectó a Arduino Mega 2560 mediante el módulo I2C. También se instaló una salida digital representada por una luz piloto, esta se activará y desactivará desde la plataforma Kaa IoT.

El prototipo cuenta con su debida protección eléctrica, dos portafusibles con capacidades de operación de 2A y 1A. También posee una fuente de alimentación de 12V 5A para alimentar el Arduino Mega 2560. Para energizar los sensores se utiliza un módulo regulador de voltaje ajustable DC a DC LM2596, que ajusta el voltaje de 12VDC a 5VDC.

La Raspberri Pi 4 tiene como función ser la puerta de enlace o Gateway entre el Arduino y la plataforma Kaa IoT utilizando el protocolo de comunicación MQTT. Se conectó una cámara web a la Raspberri Pi 4 vía puerto USB, con el fin de capturar una imagen cuando el sensor de presencia detecte algún movimiento.

El router TP-LINK WR741ND permite utilizar una red exclusiva para el prototipo (192.168.10.0/24), adicionando un nivel de seguridad a la red jerárquica, los dispositivos que se conectan a la red obtienen acceso hacia el entorno de Node-RED.

El módulo Arduino Ethernet Shield se conecta al Arduino Mega 2560, permitiéndole conectarse a la red 192.168.10.0/24, y así comunicarse a Node-RED mediante el protocolo de comunicación MODBUS TCP/IP.

Se creó una base de datos en Raspberri Pi 4, haciendo uso de un servidor Apache, PHP, MariaDB y phpMyAdmin, para que almacene la información obtenida por los sensores.

Se configuró el envío de correos electrónicos como modo de notificación de alarma, para informar cuando haya alguna lectura inusual de los sensores de temperatura, flama y gas.

Finalmente, la plataforma Kaa IoT se configuró para que reciba la información del sistema de sensores, permita visualizar el estado de la comunicación de los sensores y envíe un comando de accionamiento a Node-RED para luego redirigirlo a Arduino MEGA 2560 para activar la salida digital que enciende la luz piloto.

3.2.1. DIAGRAMA ESQUEMÁTICO DEL PROTOTIPO

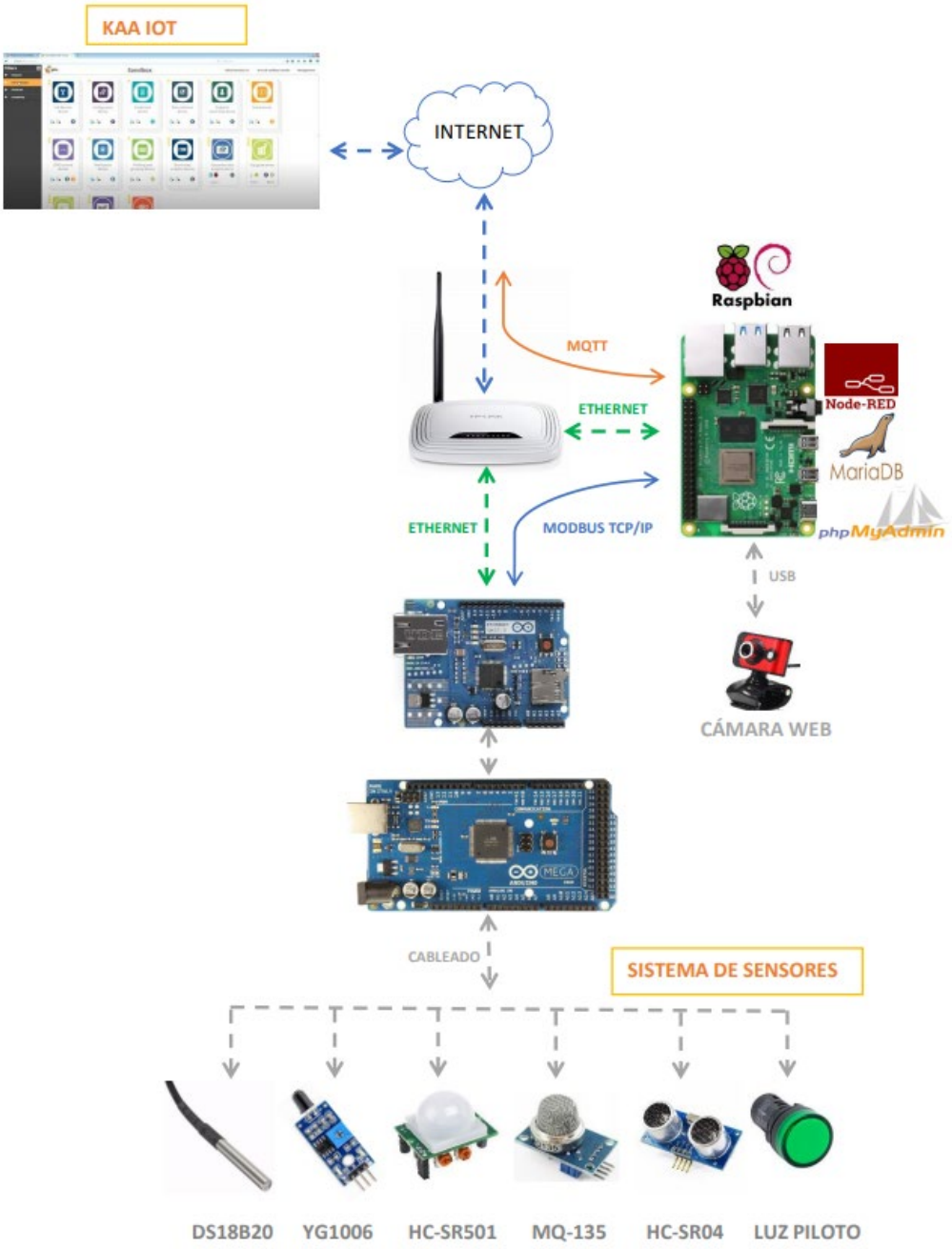


Figura 32: Diagrama esquemático del prototipo.

3.3. CONFIGURACIONES DE RED

A continuación, se muestra las configuraciones que se realizaron para preparar una red dedicada al uso del prototipo, el acceso a esta red puede ser vía cable de red y vía inalámbrica.

Las características del router instalado son:

- Marca: TP-LINK.
- Modelo: TL-WR741ND.
- MAC: F4F26DD04208.

La red configurada para el uso exclusivo del prototipo es:

- Dirección IP: 192.168.10.0
- Mascara de subred: 255.255.255.0
- Gateway: 192.168.10.1

El acceso vía WIFI es:

- Nombre de red: NODEIOT
- Contraseña: nodeiot2021

Se creo una dirección de correo electrónico para uso exclusivo del prototipo:

- Usuario: proyectonodeiot@gmail.com
- Contraseña: nodeiot2021

3.3.1. TOPOLOGÍA DE RED

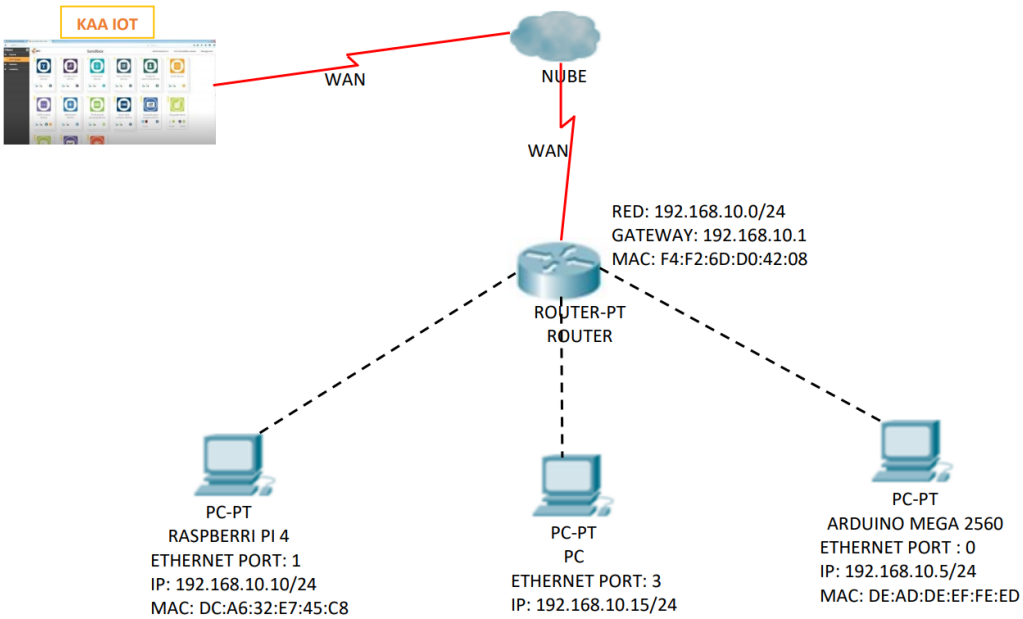


Figura 33: Topología de red.

3.3.2. CONFIGURACIONES DE ROUTER

3.3.2.1. ACCESO AL PANEL DEL ROUTER

1. Acceder al panel del router mediante el siguiente enlace http://tplinkwifi.net o a la dirección de Gateway 192.168.10.1, desde el explorador web.

2. Usar las siguientes credenciales para el acceso al panel principal.

   - Usuario: admin
   - Contraseña: admin

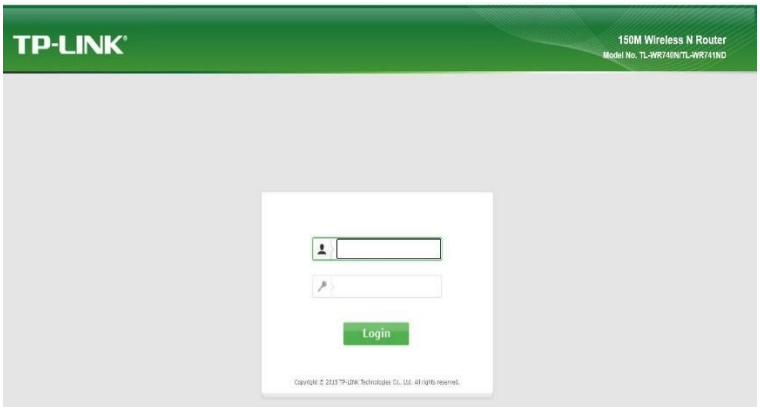


Figura 34: Acceso a panel del router TL-WR741ND.

3. Luego de ingresar las credenciales, se muestra el panel principal del router, este muestra el status vigente en el router.

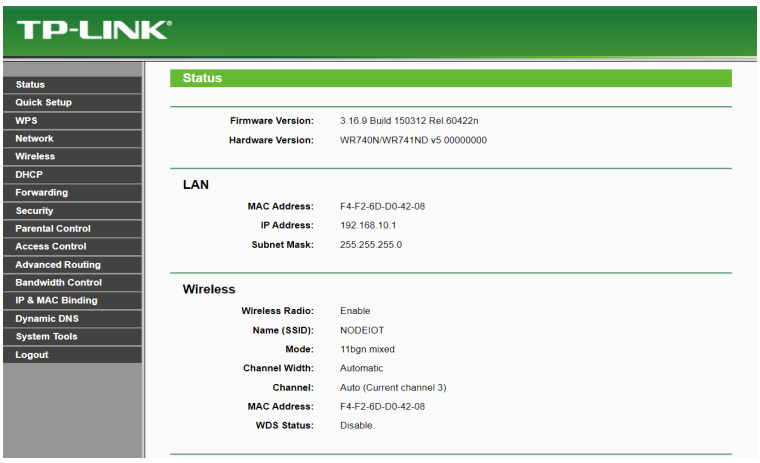


Figura 35: Panel de principal del router TL-WR741ND.

4. Configuración de parámetros para la red LAN, para uso dedicado a los dispositivos con acceso a ethernet.

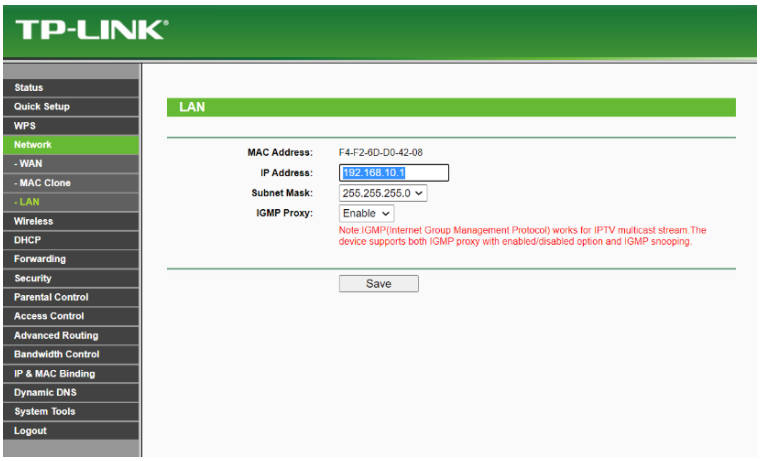


Figura 36: Configuración de red LAN.

5. Configuración de parámetros para la red WAN, para conceder acceso a internet.

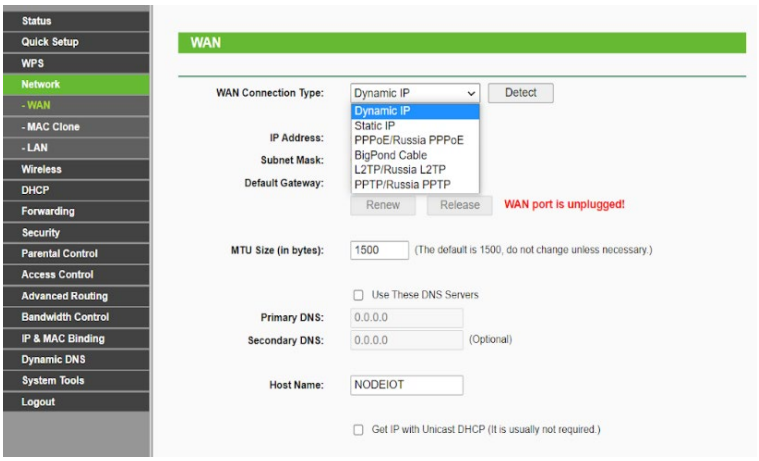


Figura 37: Configuración de red WAN.

6. Solicitud de IP a router externo con acceso a internet, para esto, se conectó el cable de red al puerto WAN del router y se pidió detectar la IP que le asignó el router externo con acceso a internet.

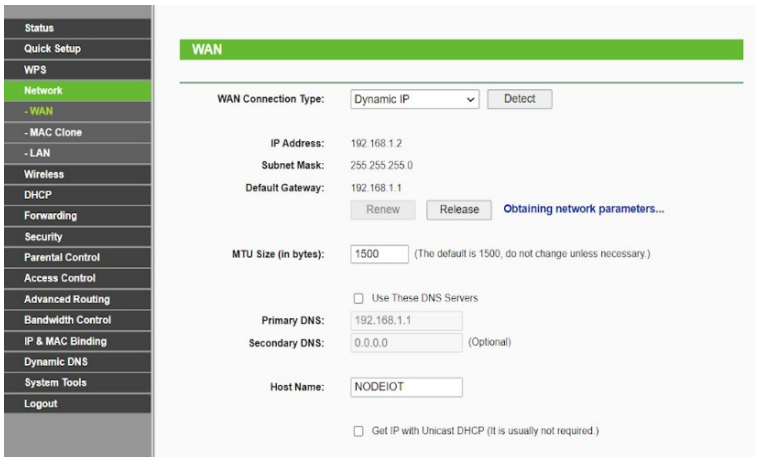


Figura 38: Configuración de red WAN.

El router externo con acceso a internet, asignó la siguiente dirección IP dentro de su red 192.168.1.2, con máscara 255.255.255.0 y con Gateway: 192.168.1.1, esto permite que el prototipo tenga acceso a internet.

7. Configuración del parámetro de la red WIFI, se estableció la contraseña de acceso a la red wifi: nodeIoT2021.

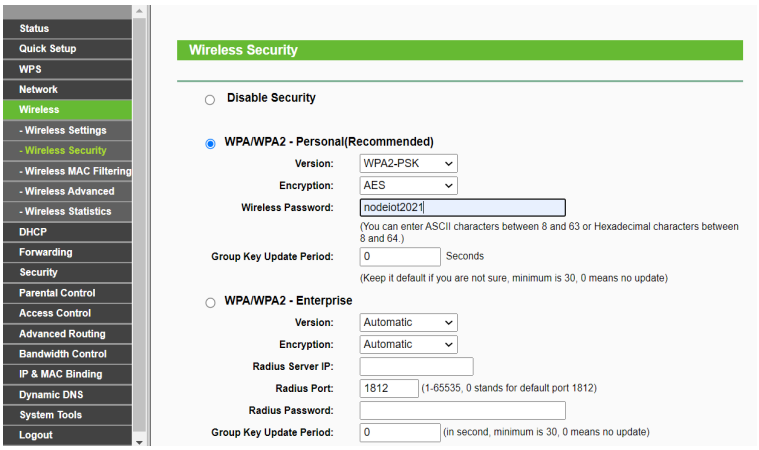


Figura 39: Configuración de red WIFI.

La asignación de las IP de la red wifi y de la red LAN se manejan mediante el protocolo DHCP.

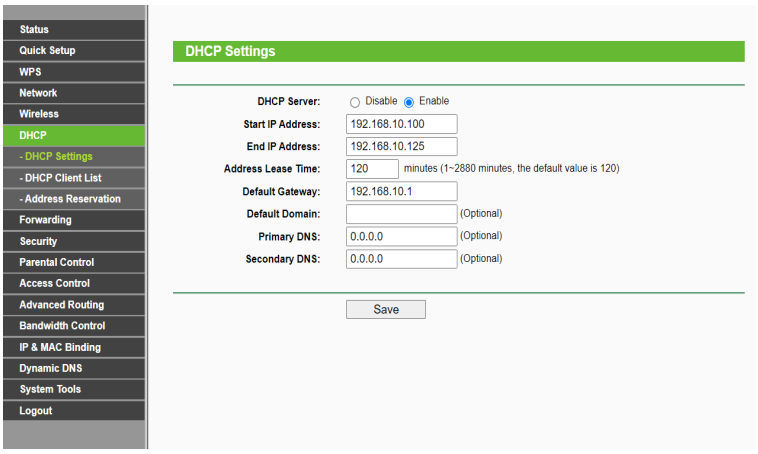


Figura 40: Configuración de red WIFI.

8. Se asigno IP estáticas para establecer la comunicación del dispositivo Raspberry y del Ethernet Shield de Arduino.

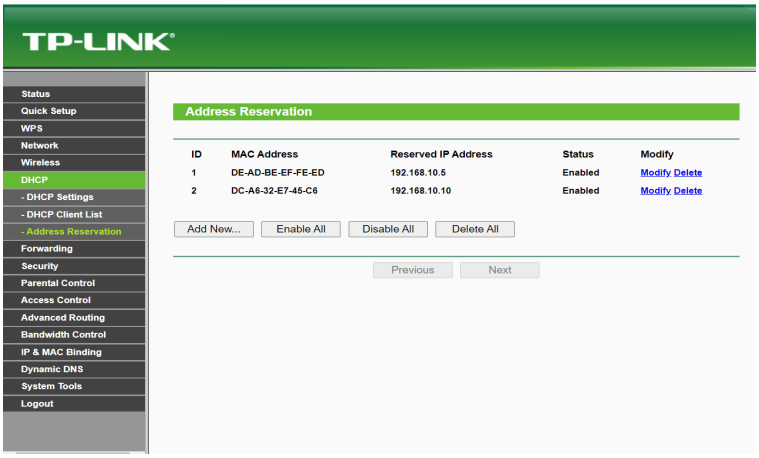


Figura 41: Asignación de IP estáticas.

El router obtendrá acceso a internet una vez reciba conexión al puerto WAN.

3.4. CONFIGURACIONES EN RASPBERRY PI 4

Se utilizó el microcomputador Raspberry Pi 4 con su versión de 8 GB de RAM para conseguir un buen desempeño de multitarea, ya que este tiene como función principal enlazar la información que se obtenga desde Arduino y enviarlo hacia la plataforma IOT, haciendo uso del servicio de nodos que brinda Node-RED se instaló un servicio de almacenamiento de datos llamado MariaDB y se habilitó el uso de la webcam.

3.4.1. INSTALACIÓN DE SISTEMA OPERATIVO EN RASPBERRY PI 4

El sistema operativo elegido para entrar en funcionamiento dentro del microcomputador se lo conoce como RASPBERRY PI OS (anteriormente denominado RASPBIAN), es un sistema operativo liviano con gran cantidad de prestaciones a la hora de trabajar, se utilizó una tarjeta microSD de 64 GB para ser usada como disco de almacenamiento del sistema operativo.

Las credenciales de acceso configuradas en el dispositivo Raspberry son:

- User: pi
- Password: raspberry

A continuación, muestro el proceso de instalación del sistema operativo desde un computador.

1. Visitar el siguiente enlace: https://www.raspberrypi.org/software/
2. Realizar la descarga de Raspberry Pi Imager.

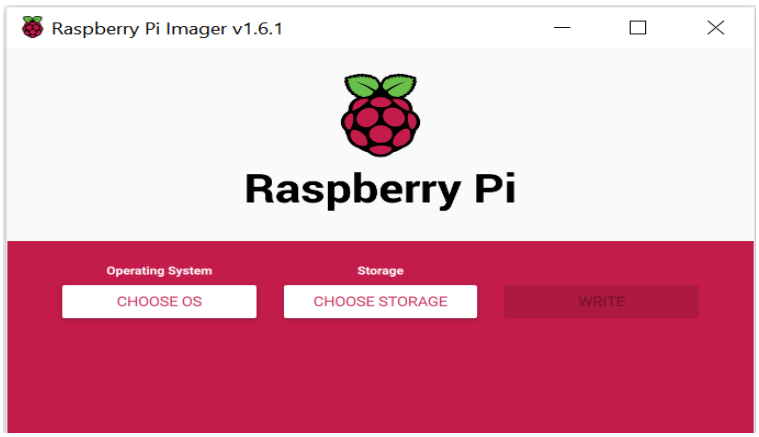


Figura 42: Panel de RASPBERRY PI IMAGER.

3. Seleccionar la imagen de Raspberry Pi OS para proceder con su descarga.

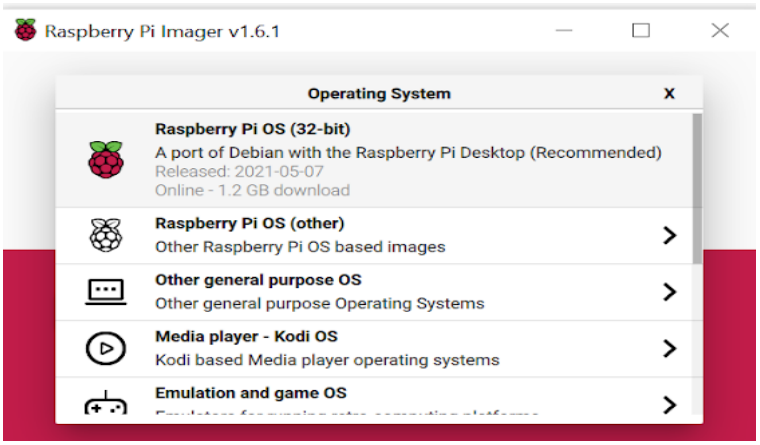


Figura 43: Panel de RASPBERRY PI IMAGER.

4. Seleccionar la tarjeta microSD insertada en el computador, la misma que servirá de unidad de almacenamiento para la Raspberry. Recomiendo usar tarjeta microSD mayor o igual a 32 GB.

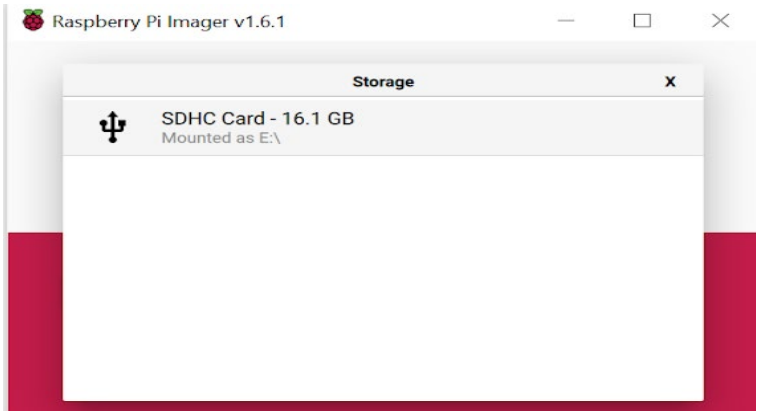


Figura 44: Panel de RASPBERRY PI IMAGER.

5. Seleccionar la opción WRITE, para dar inicio a la instalación del sistema operativo en la tarjeta microSD.

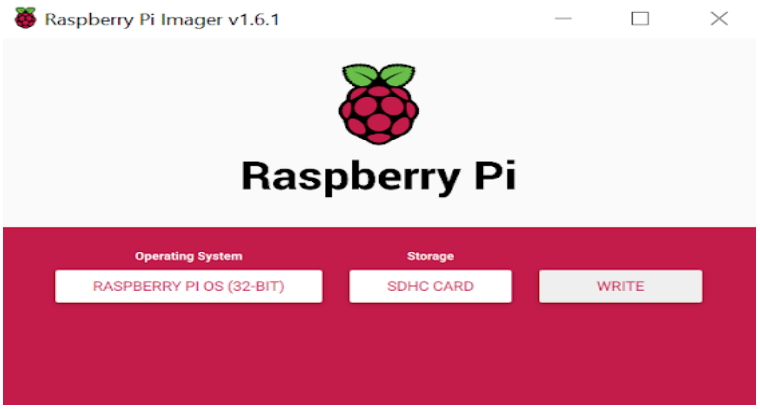


Figura 45: Panel de RASPBERRY PI IMAGER.

6. Una vez finalizada la escritura del sistema operativo en la tarjeta microSD, esta debe ser instalada en el dispositivo Raspberry.

7. Para finalizar, se conecta la fuente de poder al dispositivo Raspberry para que entre en funcionamiento y poder configurar los demás servicios.

3.4.2. INSTALACIÓN DE NODE-RED

El servicio de Node-RED permite realizar las negociaciones para concretar la comunicación entre el protocolo MODBUS TCP/IP (utilizado para aplicaciones de automatismo industrial) y el protocolo MQTT (por su simplicidad es utilizado en su totalidad por plataformas IoT).

A continuación, se muestra el proceso de instalación de Node-RED:

1. Luego de instalado Raspberry Pi OS, se procede abrir una ventana del terminal e ingresar el siguiente comando:

   bash <(curl -sL https://raw.githubusercontent.com /linux-installers/master/deb/update-nodejs-and-nodered)

   Este comando verificará los componentes necesarios para iniciar la instalación del servicio a la vez que actualiza otros componentes.

   Se utilizó como soporte para la instalación desde la página oficial de Node-RED https://nodered.org/docs/getting-started/raspberrypi.

2. Una vez ejecutado el comando y de no haberse presentado novedades, se procede a iniciar el servicio de Node-RED.

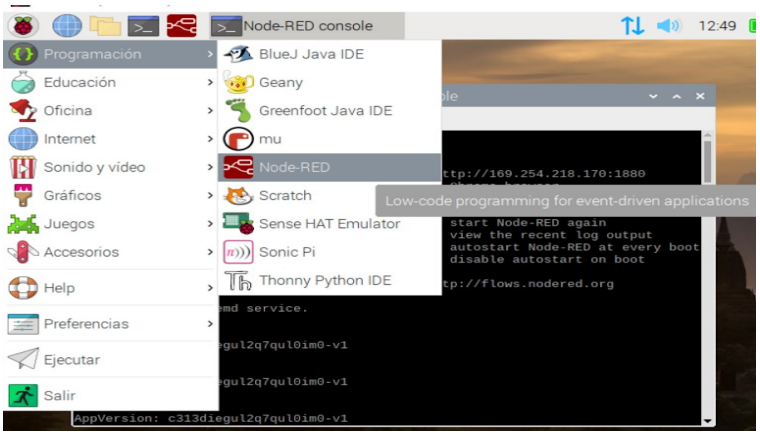


Figura 46: Inicio de Node-RED.

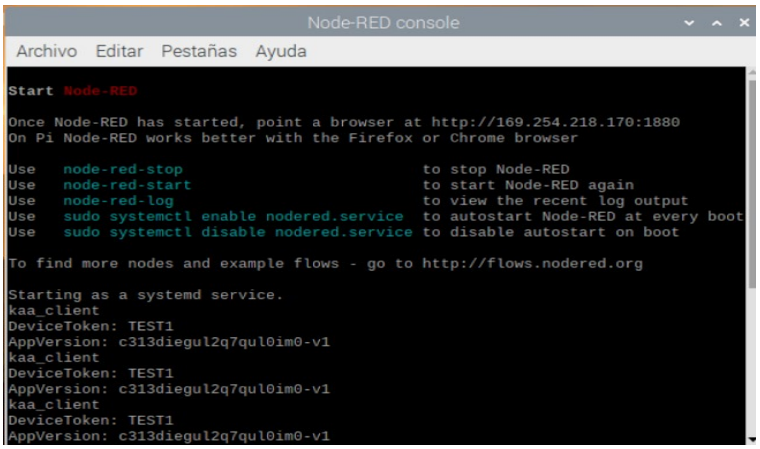


Figura 47: Consola de Node-RED.

3. Una vez activado, se procede a ingresar a Node-RED mediante el explorador WEB utilizando el siguiente enlace: http://192.168.10.10:1880

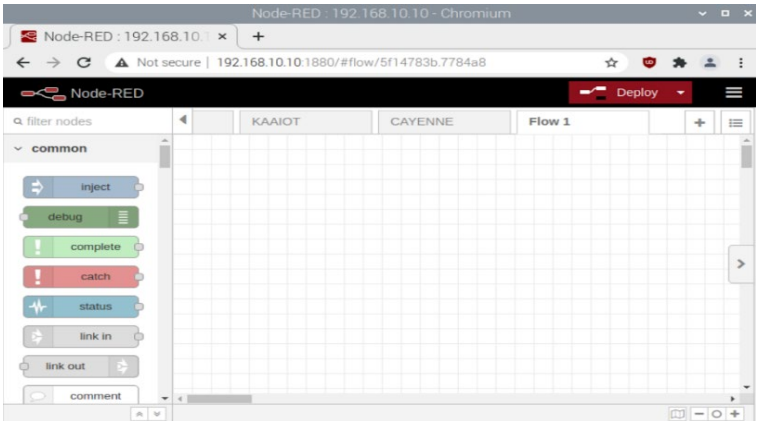


Figura 48: Entorno Node-RED.

4. Se procede a buscar e instalar los nodos a utilizar con el protocolo MODBUS TCP/IP.

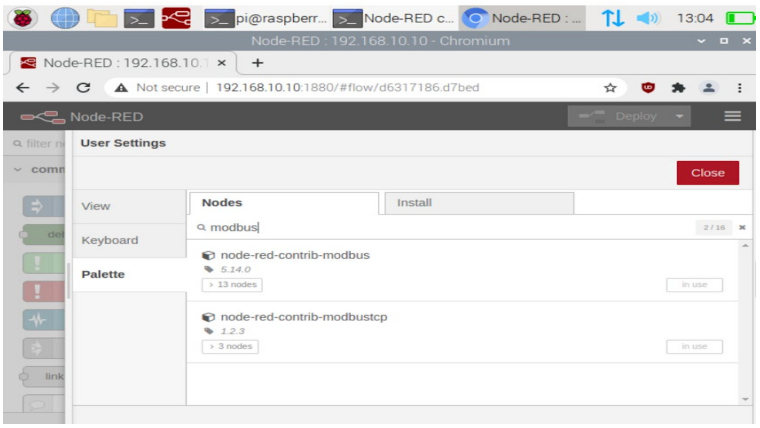


Figura 49: Librerías en Node-RED.

5. Finalmente se configuró los nodos, para recibir los datos desde Arduino y enviarlos hacia la plataforma IoT.

3.4.2.1. DIAGRAMA DE FLUJO EN NODE-RED.

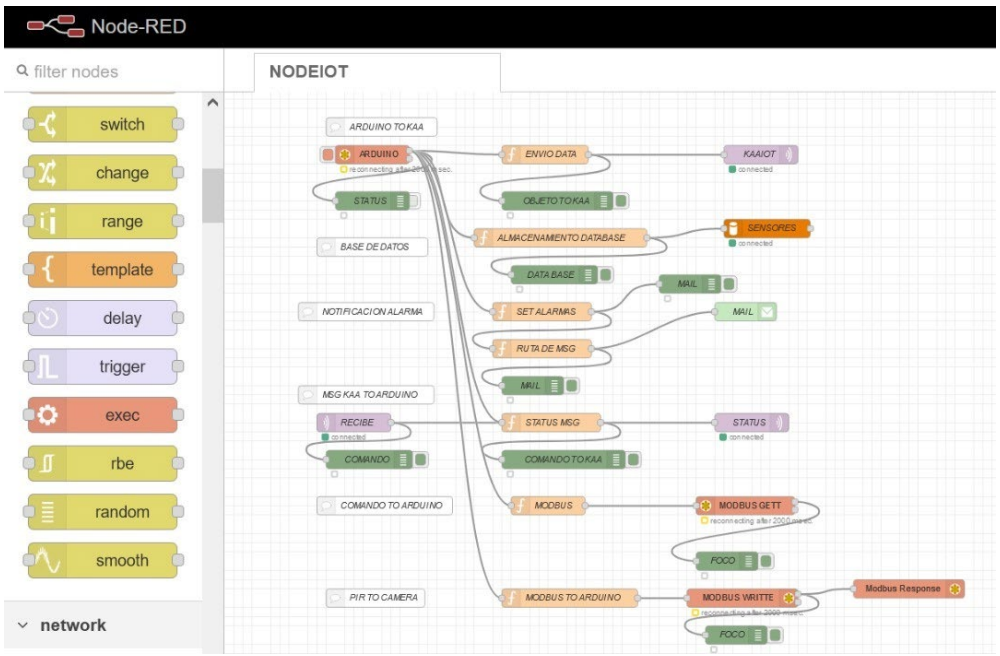


Figura 50: Diagrama de flujo en Node-RED.

3.4.3. INSTALACIÓN DE BASE DE DATOS.

Se instaló como servicio de base de datos MariaDB en Raspberri Pi 4, este servicio es gestionado por phpMyAdmin de manera local, mediante la dirección 192.168.10.10/phpmyadmin. Para hacer uso de este servicio, se realizó la instalación del servidor Apache2 y el servicio PHP, estos servicios sirvieron de complemento para el uso de la base de datos.

A continuación, se explica los pasos a seguir.

1. Abrimos la consola de comando en la Raspberri Pi 4 y ponemos el siguiente comando que nos permite actualizar todas las librerías:

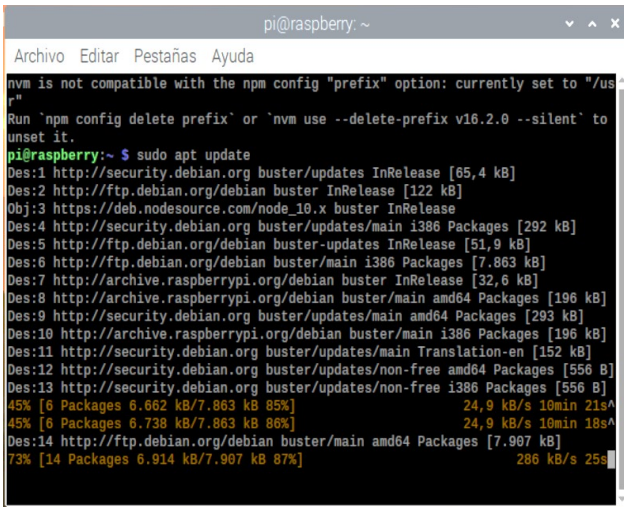


Figura 51: Comando sudo apt update.

2. Luego se ingresa el siguiente comando, para instalar el servidor “apache2”:

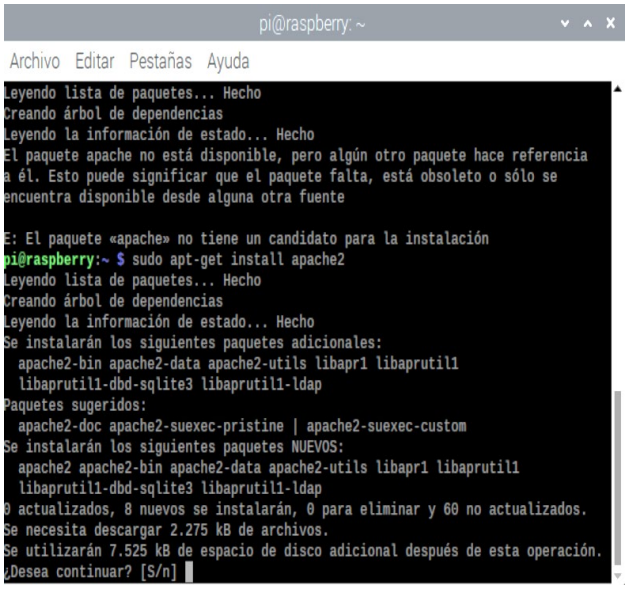


Figura 52: Instalación deapache2 como servidor.

3. Se comprueba el estado de Apache mediante el siguiente comando:

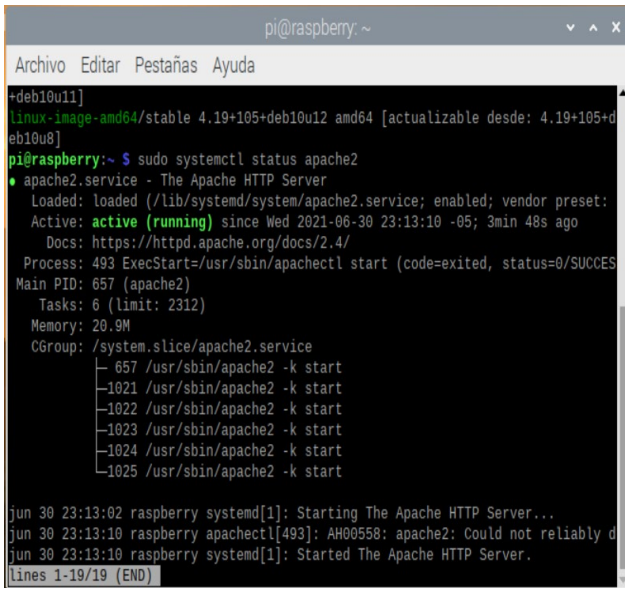


Figura 53: Verificación el estado de Apache.

4. Para asegurarnos de que Apache se instaló correctamente, abrimos el explorador y en la barra de búsqueda ingresamos “localhost”.

Figura 54: Apache instalado correctamente.

5. Instalamos PHP bajo apache en la consola de comando en Raspberri Pi 4.

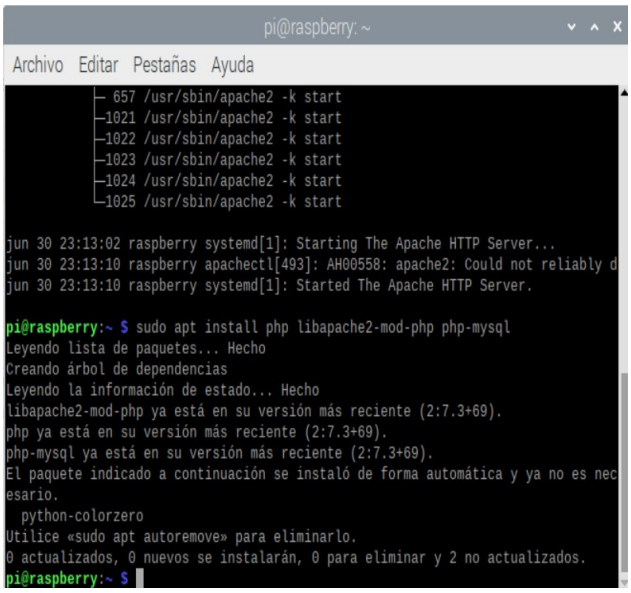


Figura 55: Instalación de PHP.

6. Para asegurarnos de que PHP se instaló correctamente, abrimos el explorador y en la barra de búsqueda ingresamos “localhost/info.php”.

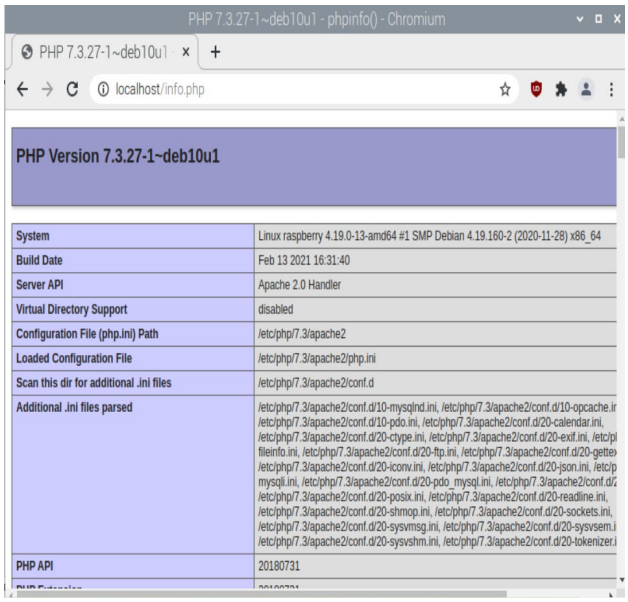


Figura 56: PHP instalado correctamente.

7. A continuación, Instalamos el servido de base de datos MariaDB ingresando en la consola de comando lo siguiente:

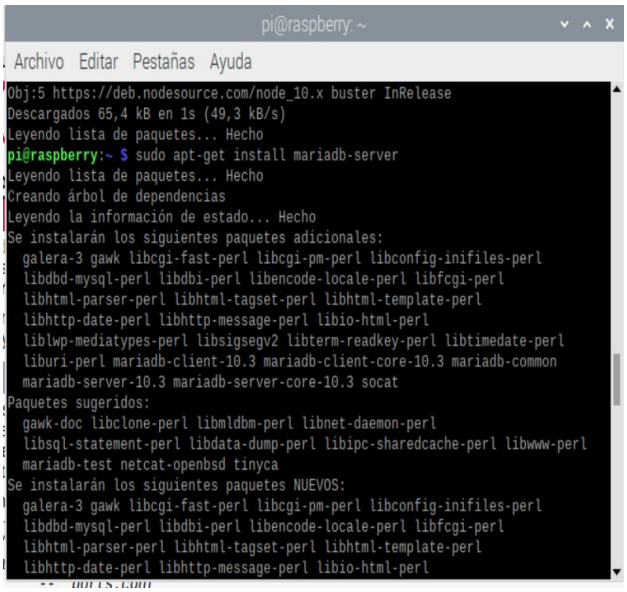


Figura 57: Instalamos el servidor de base de datos MariaDB.

8. Comprobamos el estado de MariaDB:

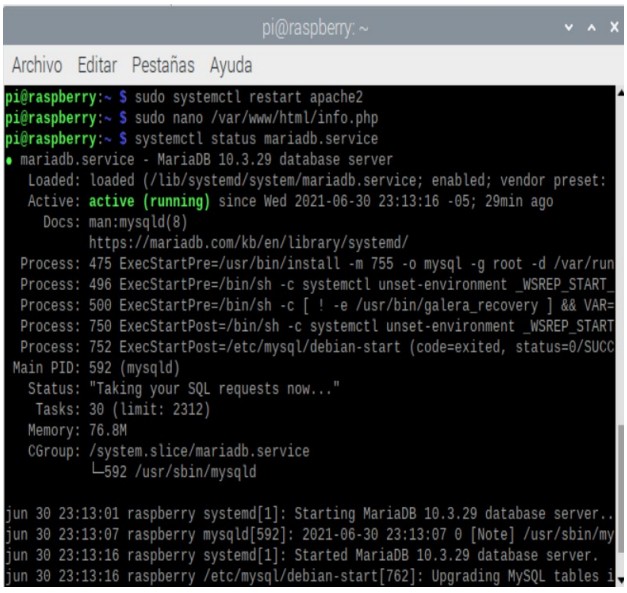


Figura 58: Verificación de estado de MariaDB.

9. Accedemos a MariaDB como root, para poder crear un usuario nuevo en la base de datos que se usa con phpMyAdmin.

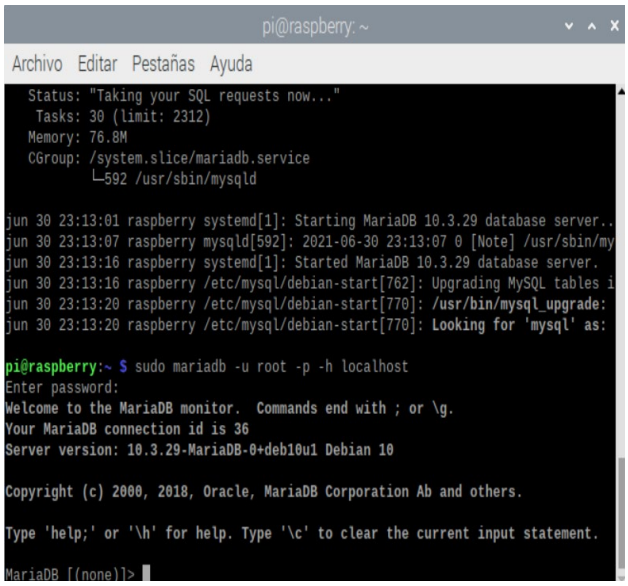


Figura 59: Acceso a MariaDB como root.

10. Procedemos a instalar phpMyAdmin para usar MariaDB, este nos brinda un panel de control desde el cual podemos manipular nuestro servidor de base de datos.

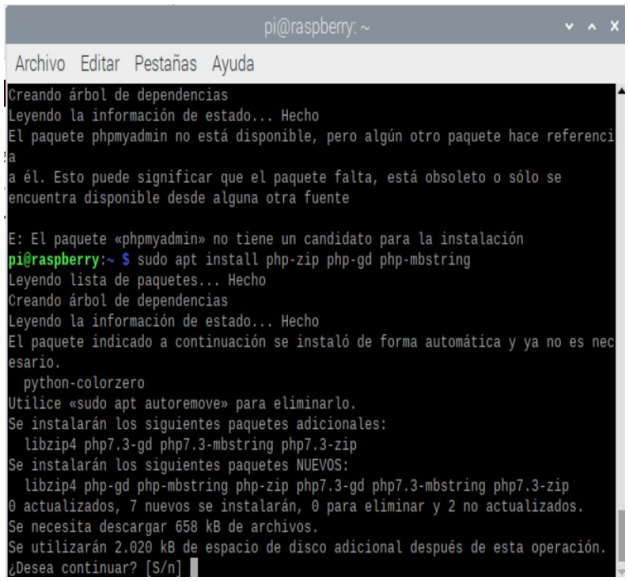


Figura 60: Instalación de phpMyAdmin.

11. Luego de la instalación de phpMyAdmin, ingresamos a la plataforma mediante el navegador.

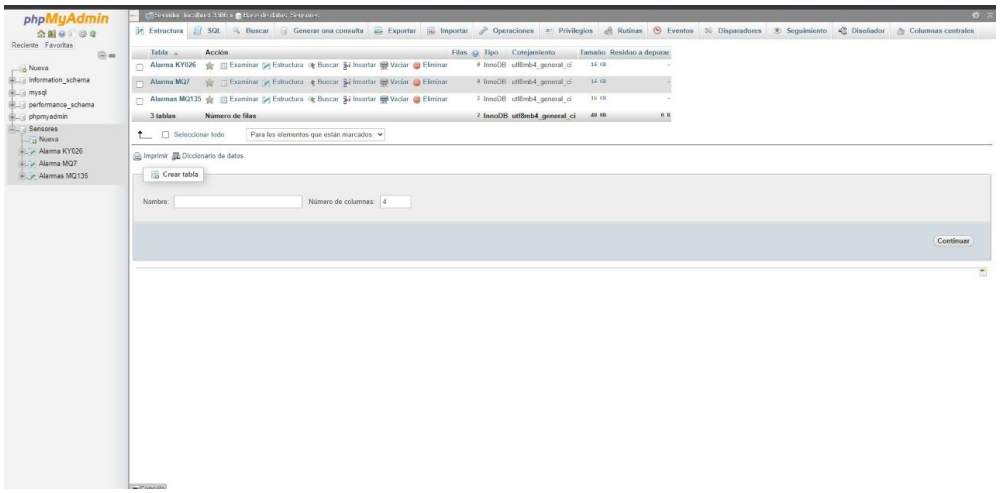


Figura 61: Presentación de phpMyAdmin.

En phpMyAdmin se diseña la interfaz de la base de datos, que muestra la información almacenada en MariaDB que proviene de los sensores y de la salida digital.

Las credenciales para acceder a phpMyAdmin son:

- Usuario: pi
- Contraseña: raspberry

Figura 62: Interfaz de la base de datos.

3.4.4. CONFIGURACIÓN DE CAMARA WEB.

La cámara web se conectó al puerto USB de Raspberri Pi 4. Mediante Node-RED se realizó la configuración para que esta se active y capture una imagen cuando el sensor de presencia detecte algún movimiento, es decir cambie de estado (0 a 1).

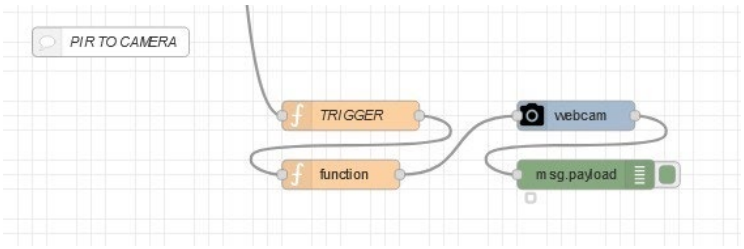


Figura 63: Configuración de cámara web en Node-RED.

3.5. CONFIGURACIÓN EN ARDUINO.

El Arduino Mega 2560 es utilizado como microcontrolador lógico programable y la placa Ethernet Shield de Arduino permite establecer comunicación vía ethernet.

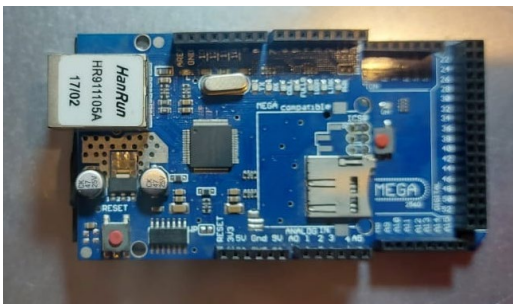


Figura 64: Arduino Mega 2660 con Arduino Ethernet Shield.

3.5.1. PROTOCOLO DE COMUNICACIÓN MODBUS.

El protocolo MODBUS TCP/IP convierte a Arduino en un esclavo MODBUS que envía constantemente información a la plataforma IoT, usando por defecto el puerto 502.

Para utilizar el protocolo MODBUS TCP/IP en Arduino, se realiza la descarga de la librería en el siguiente enlace: https://myarduinoprojects.com/modbus.html.

Una vez descargada la librería “MgsModbus-v0.1.1.zip”, se incluye la librería en la hoja de código de Arduino y se configura MODBUS en Arduino.

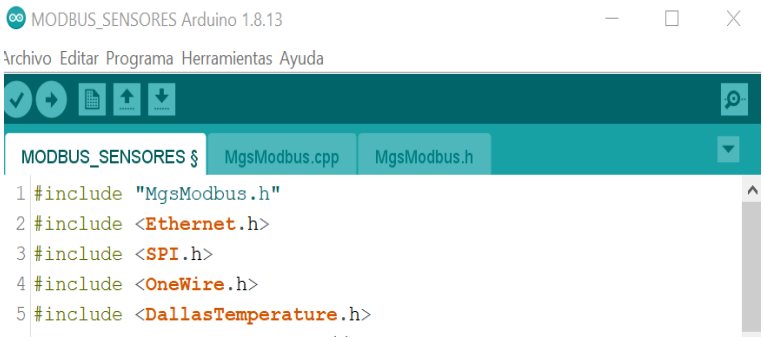


Figura 65: Librerías MgsModbus incluidas en la hoja de código de Arduino.

Dentro del código de Arduino se configuran las direcciones holding register de cada uno de los sensores para hacer uso del protocolo MODBUS TCP/IP.

DIRECCIONES DE HOLDING REGISTER	
HOLDING REGISTER	**IDENTIFICADOR**
[1]	DS18B20
[2]	HC-SR04
[3]	HC-SR501
[4]	YG1006
[5]	MQ-135
[6]	Output Digital

Tabla 1: Direcciones de holding register.

3.5.2. SENSORES EN ARDUINO

Este prototipo utiliza cinco sensores para la simulación del entorno a monitorear, el montaje de los sensores se realizó en la parte frontal del tablero. Cada uno de los sensores cuenta con un switch, de activación que permite el paso de 5VDC de alimentación. con el fin de mostrar el funcionamiento individual de cada uno de ellos en la plataforma Kaa IoT.

A este sistema de sensores también se le adicionó una salida digital representado por una luz piloto color verde, que recibe un comando u orden de encendido y apagado desde la plataforma Kaa IoT.

Figura 66: Sistema de sensores.

En el microcontrolador Arduino MEGA 2560 se realizó la programación correspondiente para la lectura, ajustes y operación de cada uno de los sensores. A continuación, una breve explicación de las librerías que se utilizaron para los sensores y la conexión de los sensores a los pines del Arduino MEGA 2560.

3.5.2.1. SENSOR DE TEMPERATURA

El sensor de temperatura que se utilizó es DS18B20, es una sonda de acero, resistente a la intemperie y al óxido, se programa mediante código en la plataforma de Arduino para que realice la lectura de temperatura de superficies solidas como superficies liquidas. Para su funcionamiento necesita incluir la librería “OneWire.h” y “DallasTemperature.h”.

Se instala una resistencia de 4.7 Kohm a la salida digital del sensor, debido a que la distancia del recorrido del cableado usado es menor a 5 metros del Arduino Mega.

CONEXIÓN DS18B20	
PIN SENSOR	**PIN CONEXIÓN**
VCC	+Vout (LM2596)
GND	-Vout (LM2596)
DATA	9 (Arduino)

Tabla 2: Conexión del sensor de temperatura.

3.5.2.2. SENSOR DETECTOR DE FLAMA.

El sensor detector de flama que se utilizó es el YG1006, este módulo detecta la presencia de fuego mediante el fototransistor NPN YG1006 (sensible a la luz infrarroja), tiene un potenciómetro para regular el nivel de sensibilidad.

CONEXIÓN YG1006	
PIN SENSOR	**PIN CONEXIÓN**
VCC	+Vout (LM2596)
GND	-Vout (LM2596)
DO	44 (Arduino)
AO	A1 (Arduino)

Tabla 3: Conexión del sensor detector de flama.

3.5.2.3. SENSOR DE PRESENCIA.

El sensor de presencia que se utilizo es un HC-SR501, este detecta el movimiento de un cuerpo midiendo la radiación infrarroja mediante el sensor piro eléctrico que posee. Además, posee dos potenciómetros para ajustar el tiempo entre mediciones y la sensibilidad de la medición.

CONEXIÓN HC-SR501	
PIN SENSOR	**PIN CONEXIÓN**
VCC	+Vout (LM2596)
GND	-Vout (LM2596)
OUT	5 (Arduino)

Tabla 4: Conexión del sensor de presencia.

3.5.2.4. SENSOR DE GAS.

El sensor que se utilizó es el MQ-135, detecta gases dañinos o contaminantes como el amoniaco, humo, alcohol. Posee un potenciómetro para ajustar la sensibilidad.

CONEXIÓN MQ-135	
PIN SENSOR	**PIN CONEXIÓN**
VCC	+Vout (LM2596)
GND	-Vout (LM2596)
DOUT	45 (Arduino)
AOUT	A2 (Arduino)

Tabla 5: Conexión del sensor de gas.

3.5.2.5. SENSOR ULTRASÓNICO

El sensor que se utilizó es el HC-SR04, sirve para medir distancias dentro de un rango determinado.

CONEXIÓN HC-SR04	
PIN SENSOR	**PIN CONEXIÓN**
VCC	+Vout (LM2596)

GND	-Vout (LM2596)
TRIG	7 (Arduino)
ECHO	6 (Arduino)

Tabla 6: Conexión del sensor de medición de distancia.

3.5.3. PRESENTACIÓN DE MENSAJES EN LCD.

Se instalo una pantalla LCD 4x20 en la parte frontal del tablero, esta pantalla tiene la función de indicarnos el estado de la comunicación ethernet con Arduino MEGA 2560.

A la pantalla LCD se le conectó un módulo I2C el cual nos facilita la conexión con Arduino, para hacer uso de esta pantalla, se incluyó en el código de Arduino las librerías “wire.h” y “liquidcrystal_i2c.h”.

La pantalla LCD mostrará la siguiente información:

- Dirección IP del Arduino.
- Estado de conexión a internet del Arduino.
- Mensaje de inicio.

Todas estas interacciones fueron codificadas en la plataforma de Arduino.

3.6. CONFIGURACIÓN DE LA PLATAFORMA KAA IOT.

Se configuró el acceso a la plataforma Kaa IoT desde Node-RED.

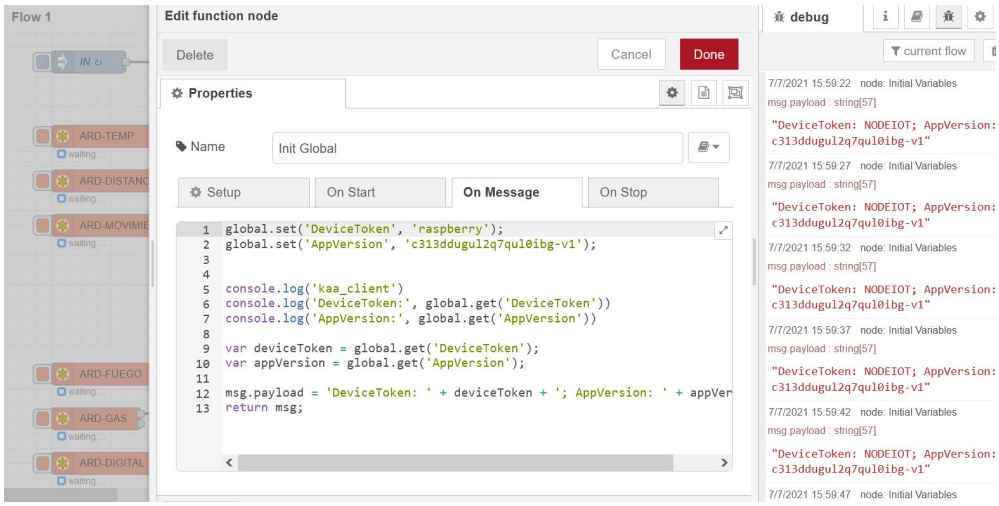


Figura 67: Código de acceso a Kaa IoT desde Node-RED.

Se configura el código que permite recibir los datos del protocolo MODBUS en Arduino y lo recibe la plataforma Kaa IoT mediante Node-RED.

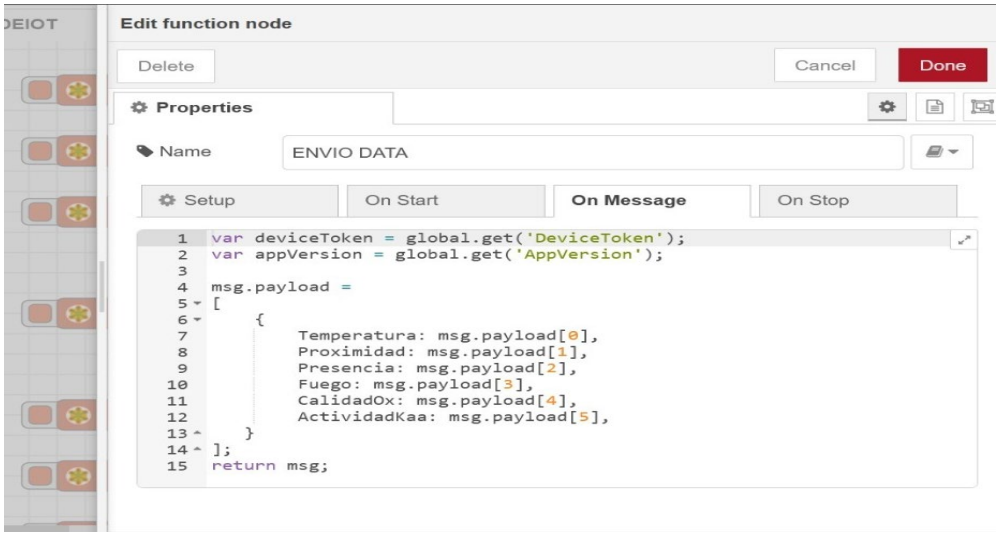


Figura 68: Código de envió de datos de Arduino a Kaa IoT.

Se configura el servidor Kaa IoT en Node-RED.

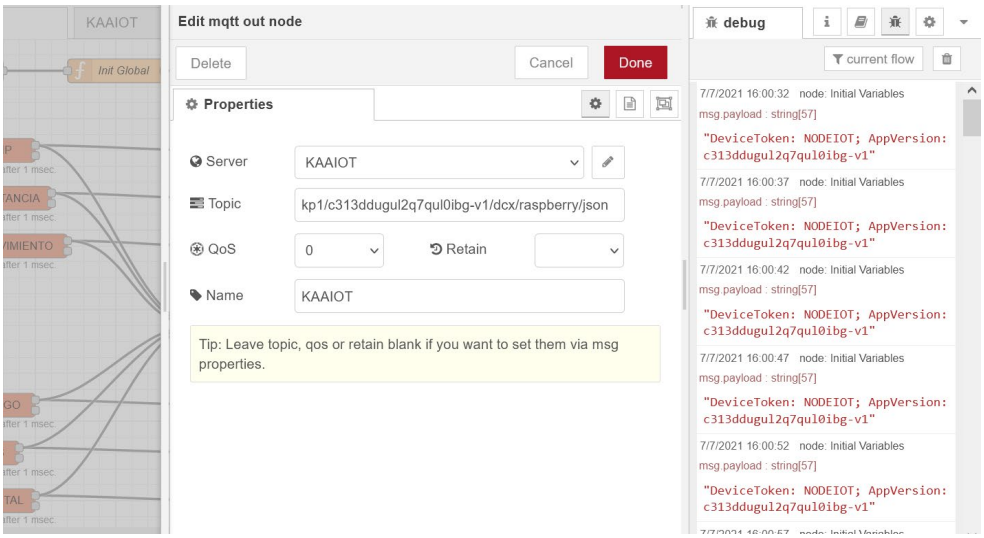


Figura 69: Configuración de Kaa IoT en Node-RED.

Luego, se abre Kaa IoT desde el explorador de un ordenador.

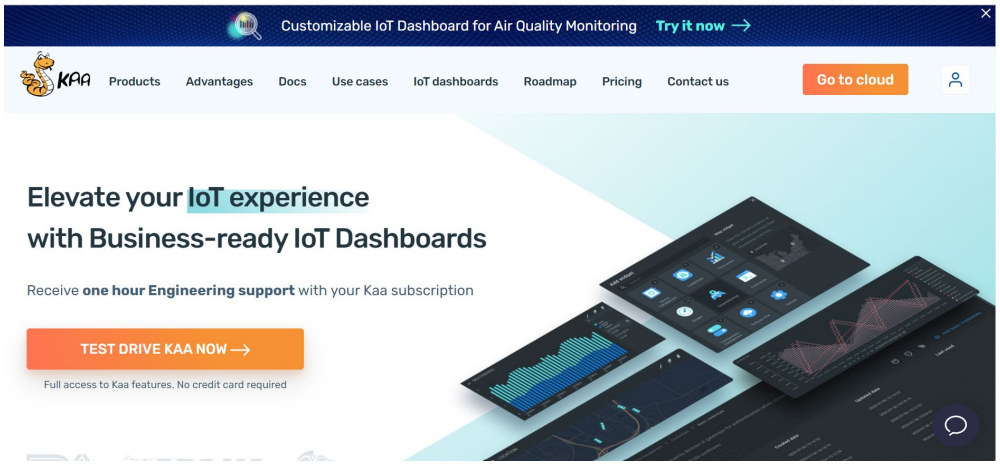


Figura 70: Plataforma Kaa IoT.

Se crea el usuario y la contraseña, el usuario que se utilizó fue el correo electrónico que se creó para el manejo de este prototipo.

- Usuario: proyectonodeiot@gmail.com
- Contraseña: nodeiot2021

Se accede a CLOUD Kaa IoT y dentro de este entorno se ingresa a Root account donde se hace las configuraciones de la plataforma Kaa IoT.

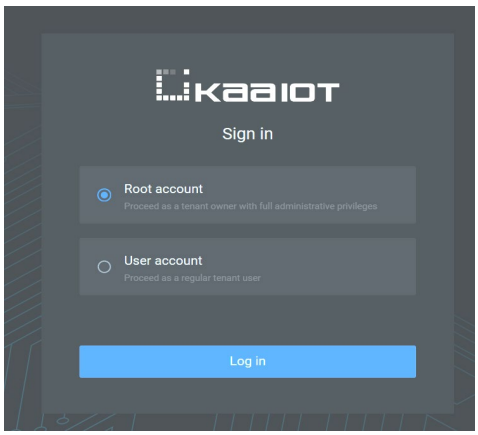


Figura 71: Ingreso a Root account.

Se registra los dispositivos utilizados en la plataforma Kaa IoT.

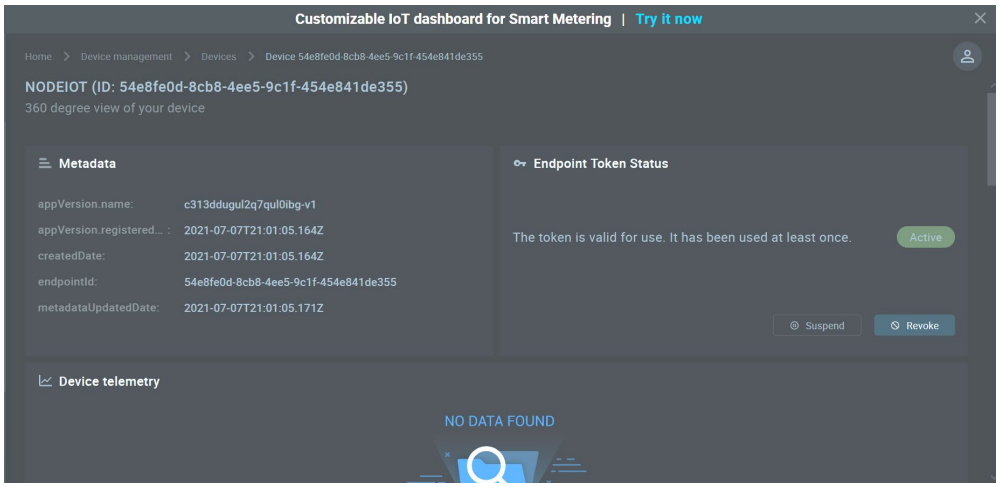


Figura 72: Registro de dispositivos en la Kaa IoT.

Se envía información desde Arduino hacia Node-RED.

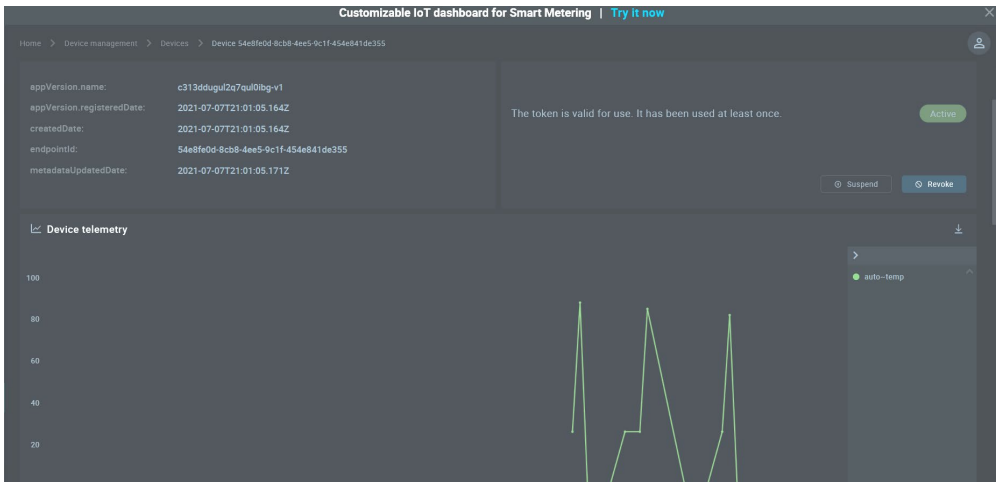


Figura 73: Transmisión de datos de Node-RED hacia Kaa IoT.

Se crean los dashboard para cada uno de los sensores.

Figura 74: Creación de dashboard.

Se configura cada dashboard y la presentación de cada uno de los sensores en la plataforma Kaa IoT.

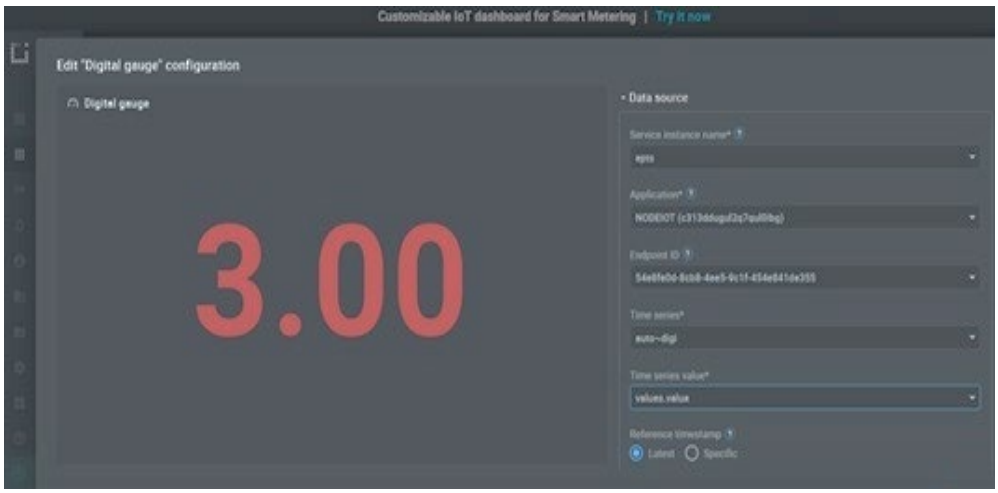


Figura 75: Configuración de dashboard.

3.7. CONFIGURACIÓN DE ALARMA

Haciendo uso de Node-RED, se estableció condiciones para tres de los cinco sensores, estos son: sensor de temperatura, sensor de flama y sensor de gas. Cuando se cumplen estas condiciones, se envía una notificación de alarma mediante correo electrónico a la dirección proyectonodeiot@gmail.com.

Figura 76: Configuración de alarma en Node-RED.

3.8. MONTAJE Y ENSAMBLAJE.

Arduino MEGA 2560 fue colocado sobre una base CZH-LABS, la cual ayudó a facilitar el cableado e instalar el dispositivo sobre un riel.

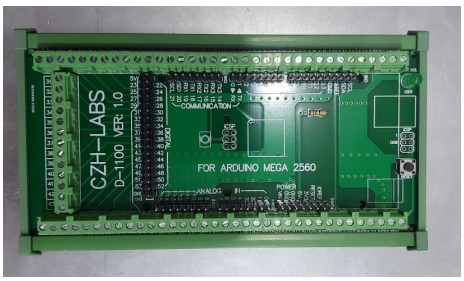


Figura 77: Base de Arduino MEGA 2560.

Raspberri Pi 4 fue colocado sobre una base que nos permitió instalarla sobre un riel.

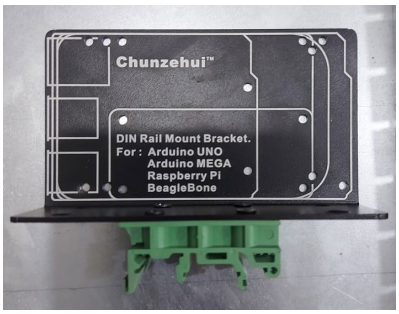


Figura 78: Base de Raspberri Pi 4.

Se realizó el montaje de los dispositivos sobre un plafón galvanizado.

Figura 79: Dispositivos instalados en plafón.

Al tablero se le instaló una rejilla para permitir la ventilación del mismo, también cuenta con un plug de conexión para su alimentación 110 VAC.

Figura 80: Rejilla de ventilación y plug de alimentación del tablero.

En la parte frontal de la puerta del tablero se instaló los selectores, luces pilotos, LCD 4x20, switchs y los sensores del sistema de sensores.

Figura 81: Instalación de elementos en puerta de tablero.

Se realizó el cableado y la conexión de los todos los dispositivos que intervienen en el tablero.

Figura 82: Cableado y conexiones del tablero.

Se realizó el rotulado de cada uno de los elementos que se encuentran en el tablero, finalizando el montaje e implementación del prototipo.

Figura 83: Presentación del prototipo didáctico.

3.8.1. DIAGRAMA ELÉCTRICO.

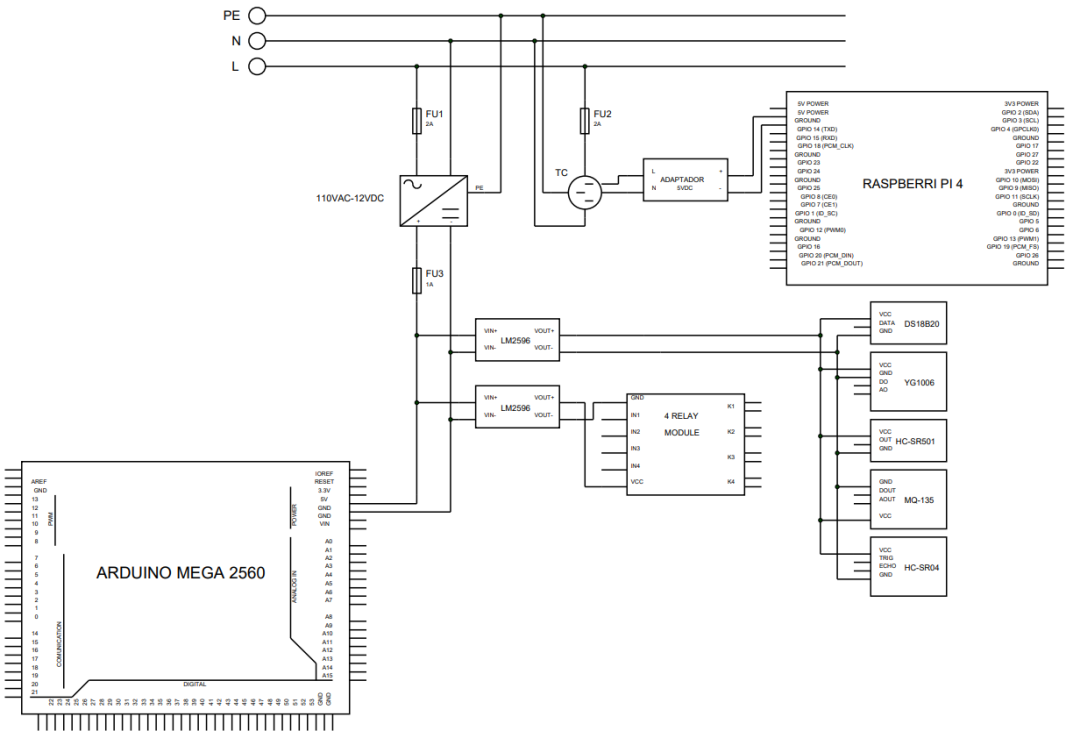


Figura 84: Diagrama eléctrico de fuerza.

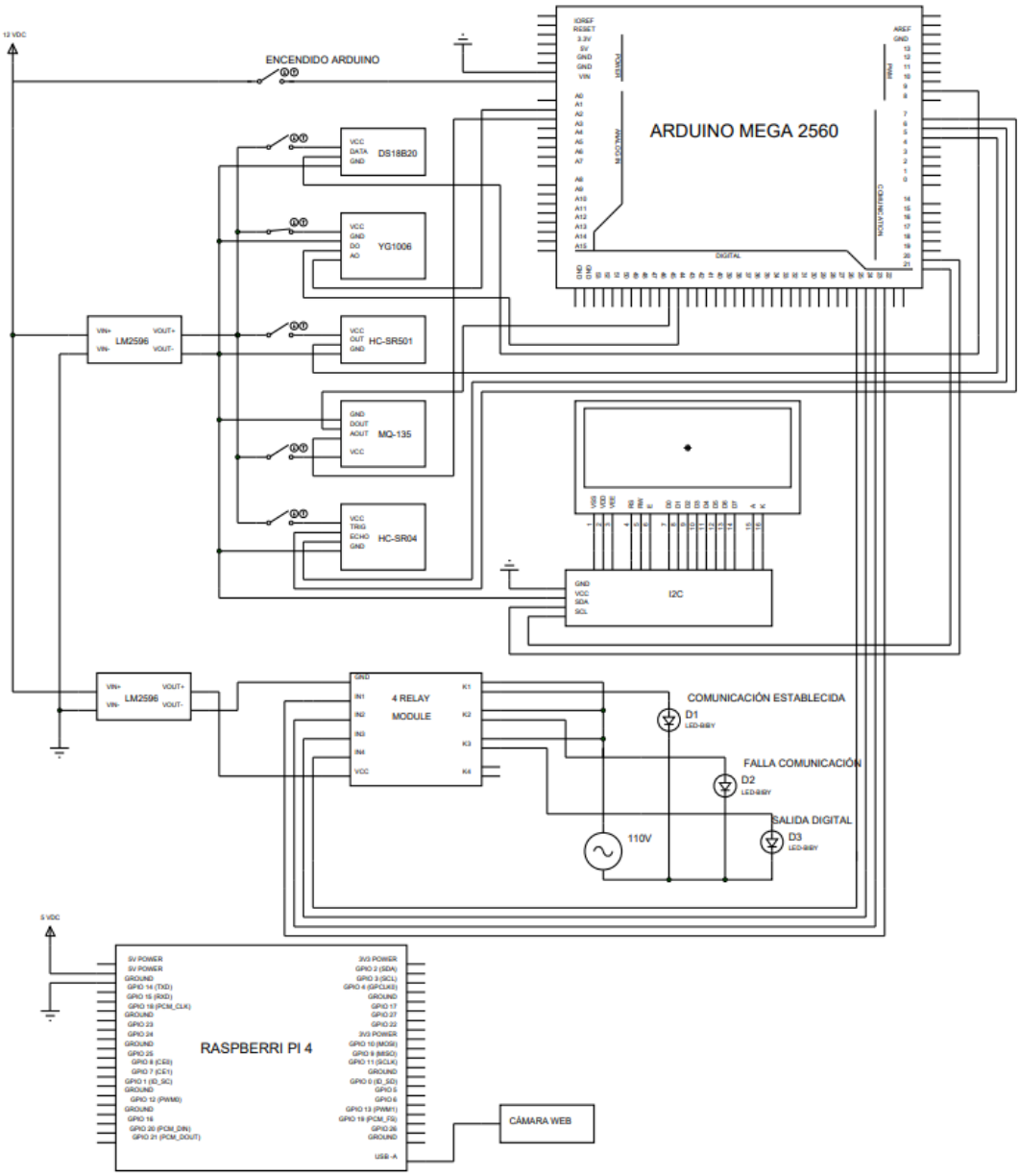


Figura 85: Diagrama eléctrico de control.

4. RESULTADOS

4.1. SISTEMA DE SENSORES CON ARDUINO

Se implementó un módulo con un sistema de cinco sensores que son monitoreados a través de la plataforma Kaa IoT de manera online en tiempo real.

Este sistema de sensores simula un entorno a monitorear, fueron programados mediante líneas de código en la plataforma de Arduino y cargadas a la placa Arduino MEGA 2560.

4.1.1. PRUEBA DE FUNCIONAMIENTO DE LOS SENSORES

En Arduino MEGA 2560, se realizó la prueba de cada uno de los sensores, comprobamos la comunicación y la lectura correcta de cada uno de ellos.

4.1.1.1. SENSOR DE TEMPERATURA

Este sensor permite leer los valores de temperatura en el prototipo, tiene un rango de trabajo de -55°C hasta 125°C, el sensor de temperatura en condiciones normales realiza una lectura de la temperatura ambiente con una diferencia de medición +- 2°C, comparado con su similares DHT11 y DHT22.

A continuación, se observan las gráficas de las lecturas analógicas que realiza el sensor de temperatura en funcionamiento.

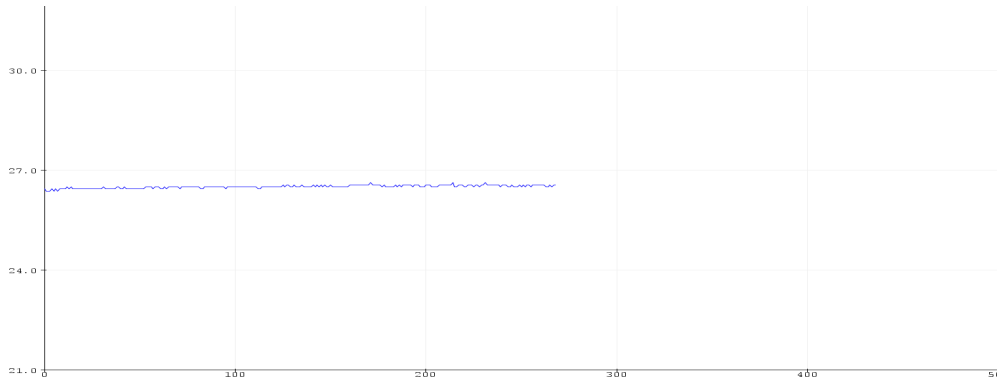


Figura 86: Lectura del sensor de temperatura en reposo.

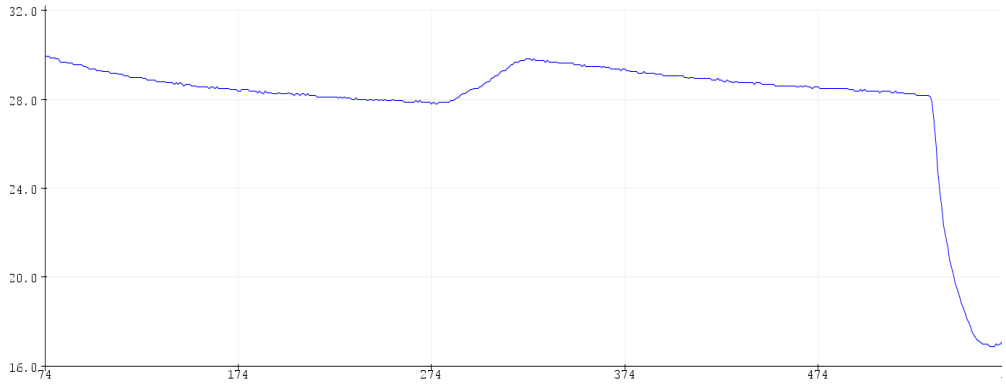


Figura 87: Lectura del sensor de temperatura accionado.

4.1.1.2. SENSOR DETECTOR DE FLAMA

Este sensor permite detectar la presencia de flama o de fuego en el prototipo, tiene un rango de funcionamiento de -25°C a 85°C y un rango de detección desde los 20 cm hasta 100 cm.

A continuación, se observan las gráficas de las lecturas analógicas del sensor de detector de flama en funcionamiento.

Figura 88: Lectura del sensor detector de flama en reposo.

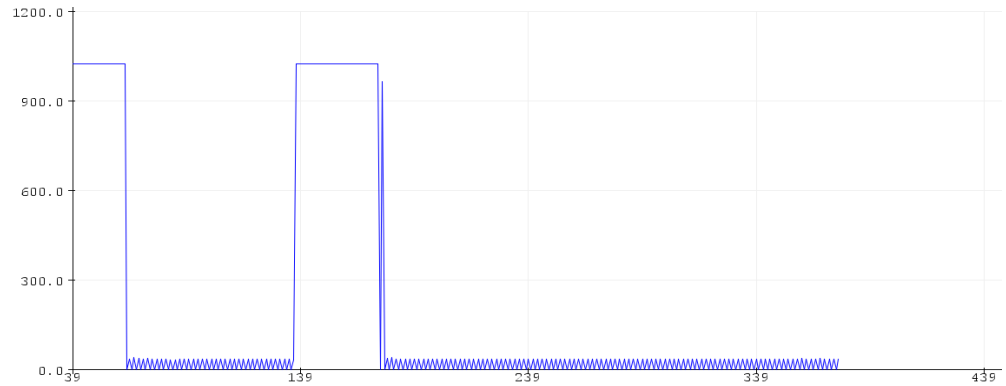


Figura 89: : Lectura del sensor detector de flama accionado.

4.1.1.3. SENSOR DE PRESENCIA

Este sensor permite detectar la presencia de alguna persona u objeto cercano al prototipo, el rango de detección es hasta 3 metros.

A continuación, se observan las gráficas de las lecturas digitales del sensor de presencia en funcionamiento.

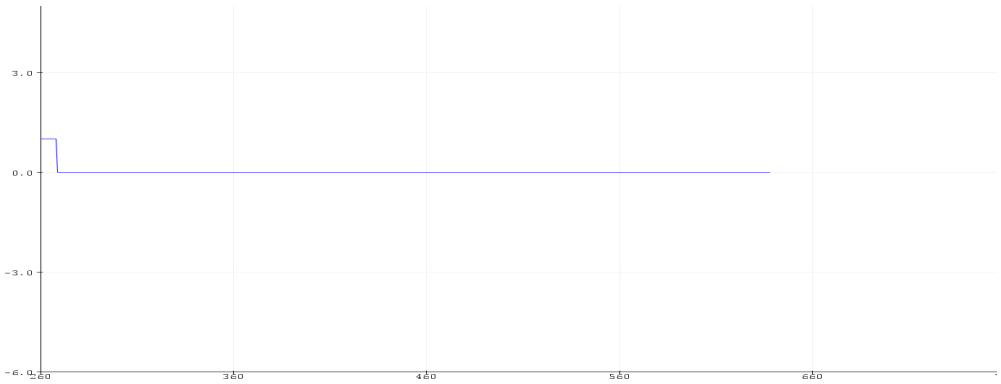


Figura 90: Lectura del sensor de presencia en reposo.

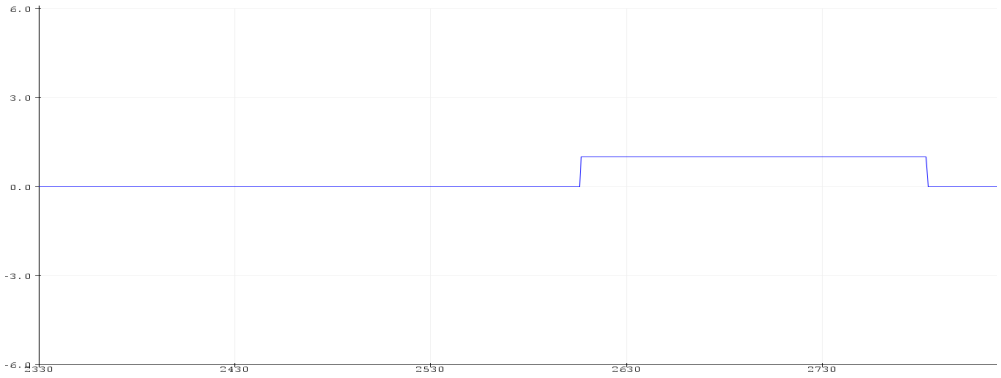


Figura 91: Lectura del sensor de presencia accionado.

4.1.1.4. SENSOR DE GAS

Este sensor permite detectar la presencia de gases dañinos o contaminantes como el CO2, amoniaco, alcohol, benceno; esto se puede dar por el manejo de compuestos químicos o por algún incendio producido. Este sensor puede detectar concentraciones de gas desde 10ppm hasta 10000 ppm.

A continuación, se observan los valores de las lecturas analógicas del sensor de gas en funcionamiento.

```
* Lectures from MQ-135 ****
    CO    |  Alcohol |    CO2   |  Tolueno  |  NH4   |  Acteona  |
   4.70   |    1.53  |   403.23 |    0.64   |   4.83 |    0.54   |
   3.38   |    1.17  |   402.54 |    0.48   |   3.93 |    0.41   |
   3.38   |    1.17  |   402.54 |    0.48   |   3.93 |    0.41   |
   3.78   |    1.28  |   402.76 |    0.53   |   4.22 |    0.45   |
   3.38   |    1.17  |   402.54 |    0.48   |   3.93 |    0.41   |
   3.38   |    1.17  |   402.54 |    0.48   |   3.93 |    0.41   |
   3.01   |    1.06  |   402.34 |    0.43   |   3.65 |    0.37   |
   3.01   |    1.06  |   402.34 |    0.43   |   3.65 |    0.37   |
   3.38   |    1.17  |   402.54 |    0.48   |   3.93 |    0.41   |
   2.67   |    0.97  |   402.14 |    0.39   |   3.39 |    0.33   |
   3.01   |    1.06  |   402.34 |    0.43   |   3.65 |    0.37   |
   2.67   |    0.97  |   402.14 |    0.39   |   3.39 |    0.33   |
   2.67   |    0.97  |   402.14 |    0.39   |   3.39 |    0.33   |
```

Figura 92: Lectura del sensor de gas en reposo.

```
* Lectures from MQ-135 ****
    CO    |  Alcohol |    CO2   |  Tolueno  |  NH4   |  Acteona  |
   3.67   |    1.25  |   402.70 |    0.52   |   4.14 |    0.44   |
   3.12   |    1.10  |   402.40 |    0.45   |   3.74 |    0.38   |
   2.87   |    1.03  |   402.26 |    0.42   |   3.55 |    0.36   |
   2.87   |    1.03  |   402.26 |    0.42   |   3.55 |    0.36   |
  72.99   |   13.99  |   423.74 |    7.06   |  27.07 |    5.67   |
 155.21   |   25.74  |   441.08 |   13.66   |  43.47 |   10.82   |
 313.42   |   45.41  |   468.47 |   25.27   |  67.60 |   19.74   |
2460.09   |  239.82  |   706.20 |  153.34   | 246.62 |  115.11   |
21310.26  | 1371.62  |  1871.03 | 1014.18   | 957.20 |  730.27   |
68397.33  | 3518.07  |  3833.87 | 2813.69   |1991.25 | 1980.93   |
92554.86  | 4491.65  |  4678.35 | 3666.24   |2407.90 | 2566.13   |
48806.79  | 2678.71  |  3086.85 | 2094.27   |1610.89 | 1484.07   |
30785.92  | 1846.19  |  2322.03 | 1399.31   |1206.03 | 1000.46   |
25367.65  | 1578.96  |  2069.72 | 1181.27   |1067.94 |  847.73   |
18281.14  | 1211.85  |  1715.88 |  886.85   | 869.32 |  640.48   |
14034.84  |  978.88  |  1485.84 |  703.72   | 736.32 |  510.83   |
13659.89  |  957.70  |  1464.67 |  687.24   | 723.91 |  499.13   |
13113.14  |  926.61  |  1433.52 |  663.11   | 705.57 |  481.98   |
13659.89  |  957.70  |  1464.67 |  687.24   | 723.91 |  499.13   |
12412.57  |  886.42  |  1393.08 |  632.01   | 681.65 |  459.86   |
10346.56  |  765.20  |  1269.98 |  538.94   | 607.99 |  393.52   |
12242.38  |  876.59  |  1383.17 |  624.42   | 675.76 |  454.46   |
 9357.85  |  705.57  |  1208.72 |  493.59   | 570.82 |  361.11   |
 4683.47  |  403.40  |   888.96 |  269.36   | 369.55 |  199.71   |
 2796.08  |  265.94  |   736.06 |  171.52   | 267.27 |  128.44   |
 1641.69  |  172.98  |   628.20 |  107.63   | 191.29 |   81.43   |
```

Figura 93: Lectura del sensor de gas accionado.

4.1.1.5. SENSOR ULTRASÓNICO

Este sensor permite medir distancias, detecta algún obstáculo a una distancia determinada, su rango de detección va desde 2cm hasta 200cm.

A continuación, se observan los valores de las lecturas analógicas del sensor de medición de distancia en funcionamiento.

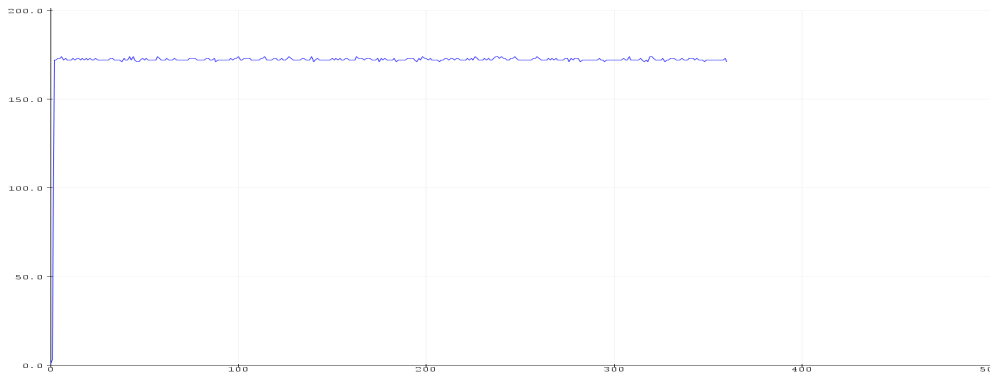


Figura 94: Lectura del sensor ultrasónico en reposo.

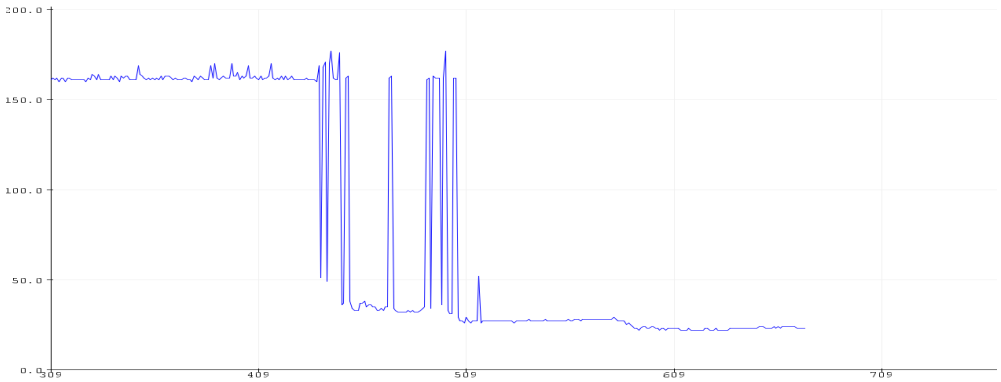


Figura 95: Lectura del sensor ultrasónico accionado.

4.2. COMUNICACIONES CON LA PLATAFORMA KAA IOT

Se estableció la comunicación con la plataforma Kaa IoT desde el entorno de programación Node-RED, se recibe la información que proviene de los sensores de manera exitosa.

Se utilizó la librería MODBUS en Arduino, esta permitió establecer comunicación al entorno Node-RED, además, haciendo uso de la librería MQTT en Node-RED se estableció comunicación con la plataforma Kaa IoT.

A continuación, se muestra la lectura de los sensores en Kaa IoT.

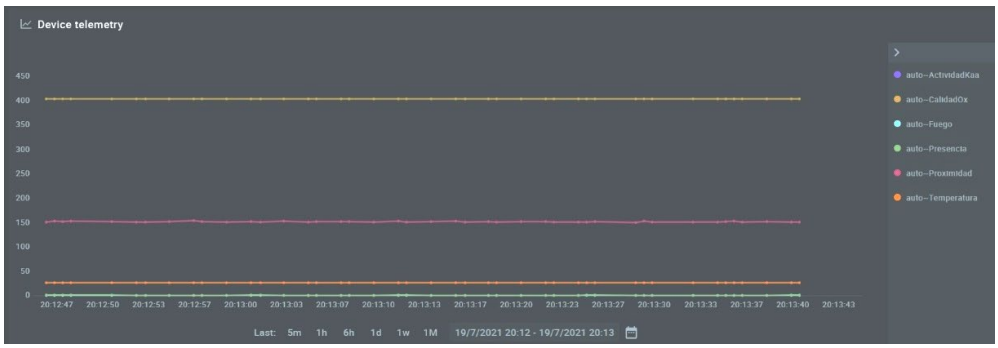


Figura 96: Comunicación de Arduino con Kaa IoT.

Se consiguió encender u apagar una luz piloto color verde ubicada en la parte frontal del tablero desde la plataforma Kaa IoT.

4.3. DASHBOARD EN KAA IOT

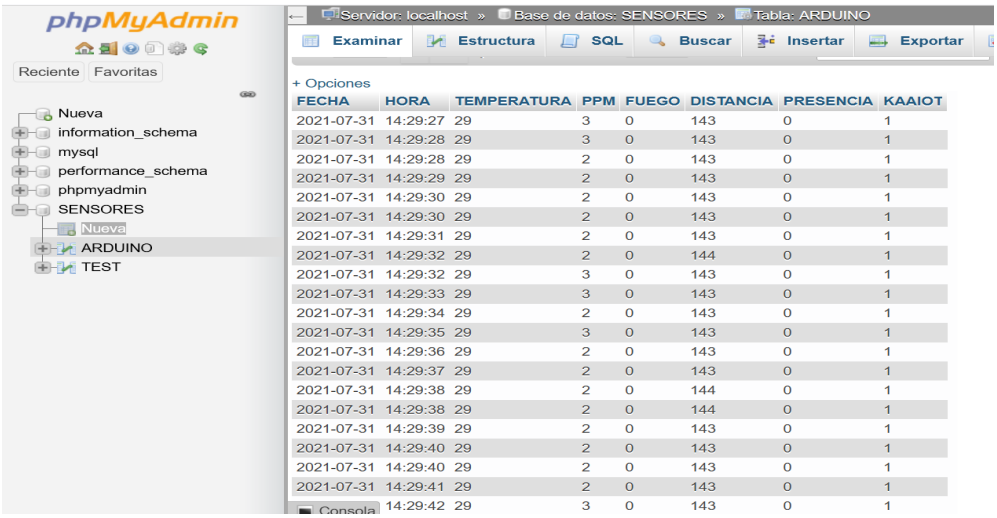
Los dashboards diseñados en la plataforma Kaa IoT, dan acceso a la visualización del estado y funcionamiento de cada uno de los sensores. Teniendo como resultado el monitoreo en tiempo real.

También se diseñó un dashboard para el comando de encendido y apagado de la salida digital.

Figura 97: Dashboards de sensores y salida digital en Kaa IoT.

4.4. BASE DE DATOS

La base de datos recopila y almacena la información que se genera mediante la lectura de los sensores y de la salida digital. La estructura de la interfaz consta de identificadores de fecha, hora, y nombre de cada uno de los sensores, incluyendo la salida digital, donde se almacena la información de manera ordenada y en secuencia.



FECHA	HORA	TEMPERATURA	PPM	FUEGO	DISTANCIA	PRESENCIA	KAAIOT
2021-07-31	14:29:27	29	3	0	143	0	1
2021-07-31	14:29:28	29	3	0	143	0	1
2021-07-31	14:29:28	29	2	0	143	0	1
2021-07-31	14:29:29	29	2	0	143	0	1
2021-07-31	14:29:30	29	2	0	143	0	1
2021-07-31	14:29:30	29	2	0	143	0	1
2021-07-31	14:29:31	29	2	0	143	0	1
2021-07-31	14:29:32	29	2	0	144	0	1
2021-07-31	14:29:32	29	3	0	143	0	1
2021-07-31	14:29:33	29	3	0	143	0	1
2021-07-31	14:29:34	29	2	0	143	0	1
2021-07-31	14:29:35	29	3	0	143	0	1
2021-07-31	14:29:36	29	2	0	143	0	1
2021-07-31	14:29:37	29	2	0	143	0	1
2021-07-31	14:29:38	29	2	0	144	0	1
2021-07-31	14:29:38	29	2	0	144	0	1
2021-07-31	14:29:39	29	2	0	143	0	1
2021-07-31	14:29:40	29	2	0	143	0	1
2021-07-31	14:29:40	29	2	0	143	0	1
2021-07-31	14:29:41	29	2	0	143	0	1
2021-07-31	14:29:42	29	3	0	143	0	1

Figura 98: *Base de datos.*

5. ANÁLISIS DE RESULTADOS

La finalidad del prototipo es establecer comunicación con un entorno industrial, donde lo más habitual es encontrar PLC o controladores robustos que ejecuten múltiples rutinas dentro de un entorno. Para realizar la comunicación de este prototipo se escogió un controlador que se ajustara al presupuesto destinado para el proyecto. Se consideró un controlador ya conocido para cumplir con este propósito, por lo cual, se escogió de la familia de Arduino al Arduino MEGA 2560. Este tiene las características de funcionamiento que se ajustan a las necesidades del proyecto en comparación a sus similares dentro de la familia Arduino.

Se consideró el tener que integrar a Arduino MEGA 2560 como un controlador de procesos, debido a esto, se tuvo en cuenta su montaje, de manera que se acople en lo más posible a un ambiente industrial, por tanto, se buscó y seleccionó la base que permitiera que se lo instale en un riel dentro del tablero y así facilite su cableado de fuerza y control.

El prototipo permite que se integren otros controladores que posean conectividad ethernet o wifi, con el propósito de que este prototipo sea escalable. Esto fue considerado al momento de realizar el diseño del prototipo, con el objetivo de permitir que en un futuro los estudiantes realicen prácticas de comunicación en los laboratorios de la Universidad Politécnica Salesiana.

La selección de los sensores para el sistema de sensores fue en base al controlador que se utilizó. Los cinco sensores utilizados se ajustan a las características técnicas de Arduino MEGA 2560.

Fueron escogidos sensores con funciones que son de uso habitual o similar en un ambiente de producción, sensores que estén presentes en un ambiente industrial, y además que aporten a las actividades de supervisión de un entorno o ambiente especifico. Tomando en consideración todos estos criterios, los sensores empleados en este prototipo fueron: sensor de temperatura, sensor de presencia, sensor medidor de gas, sensor medidor de distancia y sensor detector de flama.

Para escoger el sensor de temperatura, se realizaron pruebas con 3 tipos de sensores que operaban con Arduino MEGA 2560, estos sensores fueron DHT11, DHT22 y DS18B20. Luego de las pruebas de funcionamiento, se observó que los sensores DHT11 y DHT22 no se ajustaban de manera satisfactoria a la librería modbus en Arduino, debido a esto, fueron descartados. Sin embargo, el sensor DS18B20 funcionó de la manera esperaba y se acoplo a la librería modbus. Otro factor que facilitó escoger el sensor DS18B20 fue su encapsulado tipo termocupla, lo que permite tomar medidas de temperatura en ambientes secos y húmedos, dentro de su rango de operación.

El sensor de presencia elegido fue HC-SR501, debido a que cumplió con los requisitos del diseño del prototipo, este sensor fue regulado para que detecte presencias en un rango de hasta 3 metros durante un corto intervalo de tiempo. Además, se acopló satisfactoriamente a la librería modbus en Arduino.

Para la selección del sensor detector de gas, se tomó en consideración a lo sensores del grupo MQ, por ser compatibles con Arduino. Estos sensores son muy utilizados por su eficacia al detectar la presencia de componentes químicos en el aire. Por tanto, se tomó en cuenta, que los elementos contaminantes más habituales encontrados en el ambiente son CO2, amoniaco y alcohol, lo que dio una pauta más clara de los sensores que se podían utilizar, los sensores MQ-7, MQ-9 y MQ-135. De estos tres sensores, el MQ-135 tuvo una mejor lectura de los componentes CO2, amoniaco y alcohol en el aire. Por este motivo fue escogido para este prototipo, por mostrar una mayor sensibilidad para detectar los diferentes componentes en el ambiente, además de ser compatible con la librería modbus en Arduino.

Para realizar mediciones de distancia, Arduino no posee mayores opciones que el sensor ultrasónico HC-SR04. Este sensor ultrasónico realiza la lectura de una distancia dentro de un rango determinado mediante pulsos. Para hacer uso de este sensor, se realizaron pruebas con varias librerías para determinar su compatibilidad con la librería modbus en Arduino, siendo la librería newping.h la más apropiada. Esta librería arrojó durante las pruebas, lecturas de distancias más precisas, entre un rango de 2 cm a 200 cm.

El sensor detector de flama utilizado fue el YG1006, fue configurado en Arduino sin el uso de librerías, lo que facilito su compatibilidad con la librería modbus en Arduino. Este sensor detecta la luz emitida por alguna flama que se encuentre dentro de su rango de detección, para el prototipo su rango de detección es menor a 30 cm, debido a la ubicación en la que fue instalado. También se pudo haber utilizado el sensor KY026, ya que su comportamiento, operación y especificaciones técnicas no difieren mayormente del sensor YG1006.

La librería modbus utilizada en Arduino es MqsModbus.h, esta librería facilita la manipulación y envío de datos de los sensores hacia Node-RED, haciendo uso de los holdings register del protocolo modbus. Esta librería fue considerada para su aplicación en base a los conocimientos previos obtenidos en el periodo académico.

La fuente de voltaje empleada en el tablero alimenta el Arduino MEGA 2560, sensores, LCD 4x20, luces pilotos y el módulo relé. Por tanto, debe ser capaz de suministrar Voltaje DC estable, también se consideró que la fuente se pueda instalar sobre un riel. Tomando en consideración la demanda energética del prototipo se utilizó una fuente convertidora de voltaje 110-240VAC a 12VDC (600W), esto garantiza el suministro constante de voltaje DC sin fluctuaciones que lleguen a averiar los dispositivos.

Para la selección de la plataforma IoT, se tuvieron que tomar algunas consideraciones, estas fueron: ser de uso gratuito, permitir la recepción y envió de información sin límite de tiempo, ofrecer una variedad de diseño de dashboards, sin límite a la capacidad del uso de dashboards, ser libre de restricciones de tiempo de uso, ser acoplable a Node-RED y poseer una interfaz de configuración amigable con el administrador.

Se realizaron pruebas con varias plataformas IoT que puedan permitir el monitoreo en tiempo real del sistema de sensores, estas fueron: Thinkspeak, Thinger.io, Arest Framework, Freeboard, Cayenne y Kaa IoT.

Siendo esta ultima la mejor opción para ser utilizada en el desarrollo del prototipo, destacando por su entorno amigable, por las diversas herramientas a disposición, el permitir la lectura de datos de los sensores en tiempo real enviados vía MQTT, el permitir el envío de comandos a los dispositivos que se conectan a la plataforma y permitir la creación de dashboard de manera práctica y configurable.

En la plataforma Kaa IoT se configuraron los dashboard que presentan la información recibida de los sensores y el dashboard para ejecutar el comando de encendido y apagado de la salida digital.

Para la configuración de la base de datos, se tomó en cuenta utilizar un sistema de gestión de bases de datos con las características de software libre y de buen desempeño en rendimiento. Se tuvo en consideración para el desarrollo de la base de datos a MariaDB y MySQL, por lo cual, se escogió MariaDB por tener un mejor rendimiento en base a experiencias propias.

6. CONCLUSIÓNES

- Se resuelve que el implementar un sistema de sensores con la plataforma Arduino, es una opción viable al momento de diseñar el prototipo, por los costos accesibles y beneficios que aporta, así como la disposición y utilidades que ofrece, las pruebas de funcionamiento de los sensores utilizados evidencian durante el desarrollo del presente.

- Se logró evidenciar la comunicación que se estableció desde la plataforma Kaa IoT para obtener acceso a la información de los sensores en Arduino y así obtener el monitoreo en tiempo real de un proceso especifico.

- Se logro establecer la interacción entre el dashboard de la plataforma Kaa IoT y el sistema de sensores de Arduino.

- Se diseño la base de datos para la recopilación y almacenamiento de la información que se genera desde el sistema de sensores, permitiendo realizar el análisis de información que se registre mediante consultas a la base de datos.

- Se implementó en el entorno Node-RED un aviso de alarma mediante correo electrónico, cuando los sensores de flama, temperatura y gas detecten condiciones inusuales.

- Se diseñó e implemento una red LAN para el prototipo, con la cual evitamos conflictos de red, interferencias, fallas en la comunicación y agregamos un nivel de seguridad al prototipo.

- Se comprobó que la Raspberri Pi 4 a diferencia de sus antecesoras lograron una mejor velocidad en la ejecución de multitareas dentro de su sistema operativo.

- Se comprobó la interacción de la plataforma Kaa IoT con la salida digital (luz piloto), pudiendo activarla y desactivarla desde la plataforma Kaa IoT.

- Por lo tanto, comprobamos que este prototipo didáctico es una herramienta factible para implementar en entornos industriales para el uso de Pymes, al permitir realizar un monitoreo completo y ser de implementación sencilla, ser de bajo costo. A este prototipo se le puede integrar un Controlador Lógico Programable o PLC y extender su uso dentro de un ambiente industrial más robusto.

7. RECOMENDACIÓNES

- Utilizar una memoria SD mayor a 32 GB en la Raspberri Pi 4 para evitar problemas de almacenamiento.

- Utilizar Mozilla Firefox como explorador web para realizar configuraciones en Node-RED, puesto que en ocasiones puede presentar inconvenientes al usar otro explorador web.

- Mantener una ventilación adecuada para evitar elevadas temperaturas de la Raspberri Pi 4, así mismo, dar mantenimiento periódico a los de disipadores de calor instalados en la Raspberri Pi 4 para que mantengan la temperatura en un rango de trabajo adecuado.

- Para futuras configuraciones que se realicen en Node-RED, considerar para el módulo modbus read, el tiempo de pool rate, este no debe ser menor a 800 ms para evitar conflictos con el módulo modbus write, durante el envío de comandos a Arduino.

- Se deberá tener una conexión a internet estable para obtener una buena comunicación con la plataforma Kaa IoT.

- Para realizar conexiones o desconexiones de la parte eléctrica del módulo, se deberán realizar con las protecciones adecuadas y sin la presencia de voltaje.

- Para la base de datos, tener presente la compatibilidad entre las versiones de phpMyAdmin y php, para evitar que phpMyAdmin muestre mensajes de error durante la configuración de la base de datos.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] A. A. RIVAS, «Iniciativas para impulsar el desarrollo tecnológico de las Pymes,» Actualizado 28/07/2021 12:02.

[2] A. AMAYA, «PYMEs del país, a subirse en el tren de la innovación,» Actualizado 27/05/2021 15:01.

[3] B. I. brayan@comunicandes.com, « La resiliencia e innovación fueron aspectos decisivos para la supervivencia de las pymes en Ecuador durante la crisis sanitaria,» *Visión Pymes 2021,* 2021.

[4] I. S.L, «https://blog.infaimon.com/,» INFAIMON, 22 enero 2020. [En línea]. Available: https://blog.infaimon.com/fases-y-desarrollo-de-la-automatizacion-de-procesos/.

[5] Y. Chitiva, «Internet of Things (IoT) Diseño de una red de IoT para el hogar,» *Chitiva, Yobany,* p. 68, mayo 2020.

[6] A. Cárdenas, «Secmotic,» 28 noviembre 2016. [En línea]. Available: https://secmotic.com/plataforma-iot/#gref. [Último acceso: 17 marzo 2021].

[7] INCIBE, «incibe-cert_,» 12 septiembre 2019. [En línea]. Available: https://www.incibe-cert.es/blog/mejora-del-iiot-entornos-industriales. [Último acceso: 30 octubre 2020].

[8] L. Llamas, «LUIS LLAMAS Ingenieria, informatica y diseño,» 17 abril 2019. [En línea]. Available: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/. [Último acceso: 30 octubre 2020].

[9] A. Robledano, 18 junio 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-tcpip/. [Último acceso: 15 marzo 2021].

[10] A. García, «ADSLZone,» 23 junio 2016. [En línea]. Available: https://www.adslzone.net/2016/06/23/que-es-el-tcpip/. [Último acceso: 15 marzo 2021].

[11] O. Corporation, «Oracle,» 2010. [En línea]. Available: https://docs.oracle.com/cd/E19957-01/820-2981/ipov-10/. [Último acceso: 15 marzo 2021].

[12] ni, «ni,» 5 febrero 2021. [En línea]. Available: https://www.ni.com/es-cr/innovations/white-papers/14/the-modbus-protocol-in-depth.html. [Último acceso: 6 marzo 2021].

[13] S. Moya, «ISAMEX,» 24 septiembre 2019. [En línea]. Available: https://www.isamex.org/intechmx/index.php/2019/09/24/modbus-para-control-de-proceso-y-automatizacion/. [Último acceso: 6 marzo 2021].

[14] It@Logicbus.Com, «Logicbus,» 17 junio 2019. [En línea]. Available: https://www.logicbus.com.mx/blog/modbus-tcp-ip/. [Último acceso: 15 marzo 2021].

[15] Anónimo, «AUTOMATION NETWORKS & SOLUTIONS LLC,» [En línea]. Available: http://automation-networks.es/glossary/modbus-tcpip. [Último acceso: 15 marzo 2021].

[16] O. Fernandez, «Código Electrónica,» 3 agosto 2019. [En línea]. Available: http://codigoelectronica.com/blog/que-es-node-red#r-nodered.. [Último acceso: 20 marzo 2021].

[17] Anonimo, «IOTConsulting,» [En línea]. Available: https://iotconsulting.tech/que-es-node-red-y-para-que-sirve/. [Último acceso: 18 marzo 2021].

[18] Nodo-RED. [En línea]. Available: https://nodered.org/. [Último acceso: 18 marzo 2021].

[19] A. Gutierrez. [En línea]. Available: https://www.aiu.edu/cursos/base%20de%20datos/pdf%20leccion%201/lecci%C3%B3n%201.pdf. [Último acceso: 21 marzo 2021].

[20] J. Cahuana, 30 mayo 2020. [En línea]. Available: https://www.nettix.com.pe/documentacion/web/que-es-phpmyadmin-y-como-puedo-usarlo. [Último acceso: 20 marzo 2021].

[21] A. Suarez y J. Muguerza, «Diseño e implementación de un sistema de control de acceso y monitoreo de sensores para data center de la empresa QUIFATEX. S.A., utilizando hardware libre,» p. 113, 2018.

[22] C. Veloso, «ETOOLS,» 19 junio 2018. [En línea]. Available: https://www.electrontools.com/Home/WP/arduino-mega-2560-caracteristicas/. [Último acceso: 25 marzo 2021].

[23] Anonimo, «Web-Robótica.com,» 7 abril 2019. [En línea]. Available: https://www.web-robotica.com/arduino/como-funciona-el-modulo-arduino-ethernet-shield. [Último acceso: 26 marzo 2021].

[24] Anonimo, «ABC Tecnología,» 21 julio 2013. [En línea]. Available: https://www.abc.es/tecnologia/informatica-hardware/20130716/abci-raspberry-como-201307151936.html?ref=https:%2F%2Fwww.google.com%2F. [Último acceso: 25 marzo 2021].

[25] W. Chavez y G. Barzallo, «Diseño e implementación de un módulo didáctico basado en Raspberri Pi 3 y Arduino Mega para control y monitoreo de un brazo estabilizador.,» p. 211, 2018.

[26] Anonimo, «FUNDACIÓN RASPBERRY PI,» [En línea]. Available: https://www.raspberrypi.org/. [Último acceso: 28 marzo 2021].

[27] J. Pastor, «Xataka,» 12 julio 2018. [En línea]. Available: https://www.xataka.com/ordenadores/raspberry-pi-4-analisis-caracteristicas-precio-especificaciones. [Último acceso: 28 marzo 2021].

[28] jecrespom, «Aprendiendo Arduino,» 9 julio 2017. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/. [Último acceso: 2 mayo 2021].

[29] MeanWell, «meanweel,» [En línea]. Available: https://www.meanwell.com/Upload/PDF/MDR-60/MDR-60-SPEC.PDF. [Último acceso: 2021 mayo 2].

9. ANEXOS

9.1. ANEXO 1: CÓDIGO EN ARDUINO

```
#include "MgsModbus.h"
#include <Ethernet.h>
#include <SPI.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal_I2C.h>
#include <NewPing.h>
//definiciones pines sensores
//DS18B20
const int onewirebus = 9; //por defecto
OneWire oneWireObjeto(onewirebus);
DallasTemperature sensorDS18B20(&oneWireObjeto);
float temp;
//hc-sr04
#define TRIGGER_PIN  7
#define ECHO_PIN     6
#define MAX_DISTANCE 200
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
//LCD
LiquidCrystal_I2C lcd(0x27, 20, 4);
//PIR
int pir= 5;
//FLAMA
int flama= 44;
int Aflama;
//CALIDAD OXIGENO
```

```
int mq135= 45;
int Amq135;
//BUZZER
int buzzer= 46;
//KAA LUZ PILOTO
int kaa= 23;
//LUCES PILOTO
int verde= 25;
int rojo= 24;

//CONECTIVIDAD ETHERNET
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};//MAC
IPAddress ip(192, 168, 10, 5);
IPAddress gateway(192, 168, 10, 1);
IPAddress subnet(255, 255, 255, 0);
//INICIO PROTOCOLO MODBUS TCP/IP
MgsModbus Mb;

void setup()
{
  Serial.begin(9600);// Inicializamos comunicación serie
  Serial.println("Serial interface started");
  Ethernet.begin(mac, ip, gateway, subnet);   // start etehrnet interface
  Serial.println("Ethernet interface started");
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    // print the value of each byte of the IP address:
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
```

```
 }
 //INICIO DE FUNCION SENSORES
 sensorDS18B20.begin();
 lcd.init();
 lcd.backlight();
 lcd.setCursor(1, 0);
 lcd.print("Sistema Sensores");
 pinMode(pir  , INPUT);
 pinMode(flama , INPUT);
 pinMode(mq135 , INPUT);
 pinMode(verde , OUTPUT);
 pinMode(rojo , OUTPUT);
// pinMode(Alarma , OUTPUT);

 Mb.MbData[0] = 0; //DISABLE
 Mb.MbData[1] = 0; //TEMPERATURA
 Mb.MbData[2] = 0; //PROXIMIDAD
 Mb.MbData[3] = 0; //PRESENCIA
 Mb.MbData[4] = 0;//FUEGO
 Mb.MbData[5] = 0;//CALIDAD O2
 Mb.MbData[7] = 0;//KAA

}
void loop()
{
   // DS18B20
   Serial.println("Mandando comandos a los sensores");
   sensorDS18B20.requestTemperatures();
   // Leemos y mostramos los datos de los sensores DS18B20
```

```
  temp = sensorDS18B20.getTempCByIndex(0);
  Serial.print("Temperatura sensor: ");
  Serial.print(temp);
  Serial.println(" C");

  //HC-SR04
  unsigned int uS = sonar.ping();
  Serial.println(uS / US_ROUNDTRIP_CM);

 int Proximidad = digitalRead(pir);
 int Fuego = digitalRead(flama);
 int CalidadO2 = digitalRead(mq135);
 int Aflama = analogRead(A1);
 int Amq135 = analogRead(A2);

 Mb.MbData[1]= temp;
 Mb.MbData[2] = uS / US_ROUNDTRIP_CM;
 Mb.MbData[3] = Proximidad;
 Mb.MbData[4] = Fuego;
 Mb.MbData[5] = CalidadO2;

 Mb.MbsRun();
}
```

9.2. ANEXO 2: CÓDIGO EN NODE-RED

```
[
   {
      "id": "d6317186.d7bed",
```

```
    "type": "tab",
    "label": "NODEIOT",
    "disabled": false,
    "info": ""
},
{
    "id": "90161ddd.00431",
    "type": "debug",
    "z": "d6317186.d7bed",
    "name": "",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": true,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "payload",
    "statusType": "auto",
    "x": 530,
    "y": 140,
    "wires": []
},
{
    "id": "c7629893.777b88",
    "type": "debug",
    "z": "d6317186.d7bed",
    "name": "",
    "active": false,
    "tosidebar": true,
```

```
    "console": false,
    "tostatus": true,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "payload",
    "statusType": "auto",
    "x": 530,
    "y": 200,
    "wires": []
},
{
    "id": "902ba375.799bb",
    "type": "modbus-read",
    "z": "d6317186.d7bed",
    "name": "ARD-TEMP",
    "topic": "",
    "showStatusActivities": false,
    "logIOActivities": false,
    "showErrors": true,
    "unitid": "1",
    "dataType": "HoldingRegister",
    "adr": "0",
    "quantity": "1",
    "rate": "1",
    "rateUnit": "ms",
    "delayOnStart": false,
    "startDelayTime": "",
    "server": "c3dc893a.bc3828",
    "useIOFile": false,
```

```
    "ioFile": "",
    "useIOForPayload": false,
    "emptyMsgOnFail": false,
    "x": 170,
    "y": 140,
    "wires": [
        [
            "90161ddd.00431"
        ],
        []
    ]
},
{
    "id": "66f30ad7.9d3364",
    "type": "modbus-read",
    "z": "d6317186.d7bed",
    "name": "ARD-DISTANCIA",
    "topic": "",
    "showStatusActivities": false,
    "logIOActivities": false,
    "showErrors": true,
    "unitid": "1",
    "dataType": "HoldingRegister",
    "adr": "1",
    "quantity": "1",
    "rate": "1",
    "rateUnit": "ms",
    "delayOnStart": false,
    "startDelayTime": "",
```

```
    "server": "c3dc893a.bc3828",
    "useIOFile": false,
    "ioFile": "",
    "useIOForPayload": false,
    "emptyMsgOnFail": false,
    "x": 180,
    "y": 200,
    "wires": [
        [
            "c7629893.777b88"
        ],
        []
    ]
},
{
    "id": "65f687e1.442b28",
    "type": "modbus-read",
    "z": "d6317186.d7bed",
    "name": "ARD-MOVIMIENTO",
    "topic": "",
    "showStatusActivities": false,
    "logIOActivities": false,
    "showErrors": true,
    "unitid": "1",
    "dataType": "HoldingRegister",
    "adr": "2",
    "quantity": "1",
    "rate": "1",
    "rateUnit": "ms",
```

```
    "delayOnStart": false,
    "startDelayTime": "",
    "server": "c3dc893a.bc3828",
    "useIOFile": false,
    "ioFile": "",
    "useIOForPayload": false,
    "emptyMsgOnFail": false,
    "x": 190,
    "y": 260,
    "wires": [
        [
            "18e1a73e.754339"
        ],
        []
    ]
},
{
    "id": "f72e9abb.858578",
    "type": "modbus-read",
    "z": "d6317186.d7bed",
    "name": "ARD-FUEGO",
    "topic": "",
    "showStatusActivities": false,
    "logIOActivities": false,
    "showErrors": true,
    "unitid": "1",
    "dataType": "HoldingRegister",
    "adr": "3",
    "quantity": "1",
```

```
    "rate": "1",
    "rateUnit": "ms",
    "delayOnStart": false,
    "startDelayTime": "",
    "server": "c3dc893a.bc3828",
    "useIOFile": false,
    "ioFile": "",
    "useIOForPayload": false,
    "emptyMsgOnFail": false,
    "x": 170,
    "y": 460,
    "wires": [
        [
            "e1123973.bdf018"
        ],
        []
    ]
},
{
    "id": "f364a36e.8538f",
    "type": "modbus-read",
    "z": "d6317186.d7bed",
    "name": "ARD-GAS",
    "topic": "",
    "showStatusActivities": false,
    "logIOActivities": false,
    "showErrors": true,
    "unitid": "1",
    "dataType": "HoldingRegister",
```

```
    "adr": "4",
    "quantity": "1",
    "rate": "1",
    "rateUnit": "ms",
    "delayOnStart": false,
    "startDelayTime": "",
    "server": "c3dc893a.bc3828",
    "useIOFile": false,
    "ioFile": "",
    "useIOForPayload": false,
    "emptyMsgOnFail": false,
    "x": 160,
    "y": 520,
    "wires": [
        [
            "1b45ff11.ec6f11"
        ],
        []
    ]
},
{
    "id": "1a55d83a.0a0418",
    "type": "modbus-read",
    "z": "d6317186.d7bed",
    "name": "ARD-DIGITAL",
    "topic": "",
    "showStatusActivities": false,
    "logIOActivities": false,
    "showErrors": true,
```

```
    "unitid": "1",
    "dataType": "HoldingRegister",
    "adr": "5",
    "quantity": "1",
    "rate": "1",
    "rateUnit": "ms",
    "delayOnStart": false,
    "startDelayTime": "",
    "server": "c3dc893a.bc3828",
    "useIOFile": false,
    "ioFile": "",
    "useIOForPayload": false,
    "emptyMsgOnFail": false,
    "x": 170,
    "y": 580,
    "wires": [
        [
            "5fbf7e6c.fe01c"
        ],
        []
    ]
},
{
    "id": "18e1a73e.754339",
    "type": "debug",
    "z": "d6317186.d7bed",
    "name": "",
    "active": false,
    "tosidebar": true,
```

```
    "console": false,
    "tostatus": true,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "payload",
    "statusType": "auto",
    "x": 530,
    "y": 260,
    "wires": []
},
{
    "id": "e1123973.bdf018",
    "type": "debug",
    "z": "d6317186.d7bed",
    "name": "",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": true,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "payload",
    "statusType": "auto",
    "x": 530,
    "y": 460,
    "wires": []
},
{
    "id": "1b45ff11.ec6f11",
```

```
    "type": "debug",
    "z": "d6317186.d7bed",
    "name": "",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": true,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "payload",
    "statusType": "auto",
    "x": 530,
    "y": 520,
    "wires": []
},
{
    "id": "5fbf7e6c.fe01c",
    "type": "debug",
    "z": "d6317186.d7bed",
    "name": "",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": true,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "payload",
    "statusType": "auto",
    "x": 530,
```

```
    "y": 580,
    "wires": []
},
{
    "id": "6255911d.bbc4e8",
    "type": "modbus-read",
    "z": "d6317186.d7bed",
    "name": "ARDUINO",
    "topic": "",
    "showStatusActivities": false,
    "logIOActivities": false,
    "showErrors": true,
    "unitid": "1",
    "dataType": "HoldingRegister",
    "adr": "1",
    "quantity": "7",
    "rate": "1",
    "rateUnit": "ms",
    "delayOnStart": false,
    "startDelayTime": "",
    "server": "c3dc893a.bc3828",
    "useIOFile": false,
    "ioFile": "",
    "useIOForPayload": false,
    "emptyMsgOnFail": false,
    "x": 160,
    "y": 360,
    "wires": [
        [
```

```
            "e5b6cfa1.76b21",
            "ba1fab28.c65e48"
        ],
        []
    ]
},
{
    "id": "ba1fab28.c65e48",
    "type": "debug",
    "z": "d6317186.d7bed",
    "name": "Initial Variables",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": true,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "payload",
    "statusType": "auto",
    "x": 540,
    "y": 320,
    "wires": []
},
{
    "id": "e5b6cfa1.76b21",
    "type": "function",
    "z": "d6317186.d7bed",
    "name": "ENVIO DATA",
    "func": "var deviceToken = global.get('DeviceToken');\nvar appVersion =
global.get('AppVersion');\n\nmsg.payload = \n[\n    {\n        Temperatura:
```

```
msg.payload[0],\n        Proximidad: msg.payload[1],\n        Presencia:
msg.payload[2],\n        Fuego: msg.payload[3],\n        CalidadOx:
msg.payload[4],\n        ActividadKaa: msg.payload[5],\n    }\n    \n];\nreturn msg;",

        "outputs": 1,

        "noerr": 0,

        "initialize": "",

        "finalize": "",

        "libs": [],

        "x": 530,

        "y": 380,

        "wires": [

            [

                "eed72dde.eb4f08",

                "e57a2e2a.5bd2f8"

            ]

        ]

    },

    {

        "id": "e57a2e2a.5bd2f8",

        "type": "debug",

        "z": "d6317186.d7bed",

        "name": "Initial Variables",

        "active": true,

        "tosidebar": true,

        "console": false,

        "tostatus": true,

        "complete": "payload",

        "targetType": "msg",

        "statusVal": "payload",

        "statusType": "auto",
```

```
        "x": 780,
        "y": 420,
        "wires": []
    },
    {
        "id": "eed72dde.eb4f08",
        "type": "mqtt out",
        "z": "d6317186.d7bed",
        "name": "KAA IOT",
        "topic": "kp1/c313ddugul2q7qul0ibg-v1/dcx/raspberry/json",
        "qos": "0",
        "retain": "",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
        "broker": "8e099a06.784568",
        "x": 760,
        "y": 340,
        "wires": [],
        "info": "# Simple MQTT-based endpoint metadata management client for the Kaa IoT platform.\r\n# See https://docs.Kaa IoT.io/KAA/docs/current/Tutorials/getting-started/connecting-your-first-device/.\r\n\r\nimport itertools\r\nimport json\r\nimport queue\r\nimport random\r\nimport string\r\nimport sys\r\nimport time\r\n\r\nimport paho.mqtt.client as mqtt\r\n\r\nKPC_HOST = \"mqtt.cloud.Kaa IoT.com\"  # Kaa Cloud plain MQTT host\r\nKPC_PORT = 1883                  # Kaa Cloud plain MQTT port\r\n\r\nAPPLICATION_VERSION = \"c313ddugul2q7qul0ibg-v1\"  # Paste your application version\r\nENDPOINT_TOKEN = \"raspberry\"       # Paste your endpoint token\r\n\r\n\r\nclass MetadataClient:\r\n\r\n    def __init__(self, client):\r\n        self.client = client\r\n        self.metadata_by_request_id = {}\r\n        self.global_request_id = itertools.count()\r\n        get_metadata_subscribe_topic = f'kp1/{APPLICATION_VERSION}/epmx/{ENDPOINT_TOKEN}/get/#'\r\n        self.client.message_callback_add(get_metadata_subscribe_topic,
```

self.handle_metadata)\r\n\r\n   def handle_metadata(self, client, userdata, message):\r\n   request_id = int(message.topic.split('/')[-2])\r\n   if message.topic.split('/')[-1] == 'status' and request_id in self.metadata_by_request_id:\r\n   print(f'<--- Received metadata response on topic {message.topic}')\r\n   metadata_queue = self.metadata_by_request_id[request_id]\r\n   metadata_queue.put_nowait(message.payload)\r\n   else:\r\n   print(f'<--- Received bad metadata response on topic {message.topic}:\\n{str(message.payload.decode(\"utf-8\"))}')\r\n\r\n   def get_metadata(self):\r\n   request_id = next(self.global_request_id)\r\n   get_metadata_publish_topic = f'kp1/{APPLICATION_VERSION}/epmx/{ENDPOINT_TOKEN}/get/{request_id}'\r\n\r\n   metadata_queue = queue.Queue()\r\n   self.metadata_by_request_id[request_id] = metadata_queue\r\n\r\n   print(f'---> Requesting metadata by topic {get_metadata_publish_topic}')\r\n   self.client.publish(topic=get_metadata_publish_topic, payload=json.dumps({}))\r\n   try:\r\n   metadata = metadata_queue.get(True, 5)\r\n   del self.metadata_by_request_id[request_id]\r\n   return str(metadata.decode(\"utf-8\"))\r\n   except queue.Empty:\r\n   print('Timed out waiting for metadata response from server')\r\n   sys.exit()\r\n\r\n   def patch_metadata_unconfirmed(self, metadata):\r\n   partial_metadata_udpate_publish_topic = f'kp1/{APPLICATION_VERSION}/epmx/{ENDPOINT_TOKEN}/update/keys'\r\n\r\n   print(f'---> Reporting metadata on topic {partial_metadata_udpate_publish_topic}\\nwith payload {metadata}')\r\n   self.client.publish(topic=partial_metadata_udpate_publish_topic, payload=metadata)\r\n\r\n\r\n\r\ndef main():\r\n   # Initiate server connection\r\n   print(f'Connecting to Kaa server at {KPC_HOST}:{KPC_PORT} using application version {APPLICATION_VERSION} and endpoint token {ENDPOINT_TOKEN}')\r\n\r\n\r\n   client_id = ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(6))\r\n   client = mqtt.Client(client_id=client_id)\r\n   client.connect(KPC_HOST, KPC_PORT, 60)\r\n   client.loop_start()\r\n\r\n\r\n   metadata_client = MetadataClient(client)\r\n\r\n\r\n   # Fetch current endpoint metadata attributes\r\n   retrieved_metadata = metadata_client.get_metadata()\r\n   print(f'Retrieved metadata from server: {retrieved_metadata}')\r\n\r\n\r\n   # Do a partial endpoint metadata update\r\n   metadata_to_report = json.dumps({\"model\": \"BFG 9000\", \"mac\": \"00-14-22-01-23-45\"})\r\n   metadata_client.patch_metadata_unconfirmed(metadata_to_report)\r\n\r\n   time.sleep(5)\r\n   client.disconnect()\r\n\r\n\r\n\r\nif __name__ == '__main__':\r\n   main()"

  },

  {

    "id": "c3dc893a.bc3828",

    "type": "modbus-client",

    "name": "ARDUINO_ethernet",

    "clienttype": "tcp",

```
    "bufferCommands": true,
    "stateLogEnabled": true,
    "queueLogEnabled": false,
    "tcpHost": "192.168.10.5",
    "tcpPort": "502",
    "tcpType": "DEFAULT",
    "serialPort": "/dev/ttyUSB",
    "serialType": "RTU-BUFFERD",
    "serialBaudrate": "9600",
    "serialDatabits": "8",
    "serialStopbits": "1",
    "serialParity": "none",
    "serialConnectionDelay": "100",
    "unit_id": 1,
    "commandDelay": 1,
    "clientTimeout": 1000,
    "reconnectOnTimeout": true,
    "reconnectTimeout": 2000,
    "parallelUnitIdsAllowed": true
},
{
    "id": "8e099a06.784568",
    "type": "mqtt-broker",
    "name": "KAA IOT",
    "broker": "mqtt.cloud.Kaa IoT.com",
    "port": "1883",
    "clientid": "",
    "usetls": false,
    "protocolVersion": "4",
```

```
        "keepalive": "120",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "birthMsg": {},
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "closeMsg": {},
        "willTopic": "",
        "willQos": "0",
        "willPayload": "",
        "willMsg": {},
        "sessionExpiry": ""
    }
]
```


9.3. ANEXO 3: CÓDIGO DE MQTT EN NODE RED

```
# Simple MQTT-based endpoint metadata management client for the Kaa IoT
platform.

# See https://docs.Kaa IoT.io/KAA/docs/current/Tutorials/getting-started/connecting-
your-first-device/.


import itertools

import json

import queue

import random

import string

import sys
```

```
import time


import paho.mqtt.client as mqtt


KPC_HOST = "mqtt.cloud.Kaa IoT.com"  # Kaa Cloud plain MQTT host

KPC_PORT = 1883                  # Kaa Cloud plain MQTT port


APPLICATION_VERSION = "c313ddugul2q7qul0ibg-v1"  # Paste your application
version

ENDPOINT_TOKEN = "raspberry"       # Paste your endpoint token


class MetadataClient:


    def __init__(self, client):

        self.client = client

        self.metadata_by_request_id = {}

        self.global_request_id = itertools.count()

        get_metadata_subscribe_topic =
f'kp1/{APPLICATION_VERSION}/epmx/{ENDPOINT_TOKEN}/get/#'

        self.client.message_callback_add(get_metadata_subscribe_topic,
self.handle_metadata)


    def handle_metadata(self, client, userdata, message):

        request_id = int(message.topic.split('/')[-2])

        if message.topic.split('/')[-1] == 'status' and request_id in
self.metadata_by_request_id:

            print(f'<--- Received metadata response on topic {message.topic}')

            metadata_queue = self.metadata_by_request_id[request_id]

            metadata_queue.put_nowait(message.payload)

        else:
```

```
            print(f'<--- Received bad metadata response on topic
{message.topic}:\n{str(message.payload.decode("utf-8"))}')


    def get_metadata(self):

        request_id = next(self.global_request_id)

        get_metadata_publish_topic =
f'kp1/{APPLICATION_VERSION}/epmx/{ENDPOINT_TOKEN}/get/{request_id}'


        metadata_queue = queue.Queue()

        self.metadata_by_request_id[request_id] = metadata_queue


        print(f'---> Requesting metadata by topic {get_metadata_publish_topic}')

        self.client.publish(topic=get_metadata_publish_topic, payload=json.dumps({}))

        try:

            metadata = metadata_queue.get(True, 5)

            del self.metadata_by_request_id[request_id]

            return str(metadata.decode("utf-8"))

        except queue.Empty:

            print('Timed out waiting for metadata response from server')

            sys.exit()


    def patch_metadata_unconfirmed(self, metadata):

        partial_metadata_udpate_publish_topic =
f'kp1/{APPLICATION_VERSION}/epmx/{ENDPOINT_TOKEN}/update/keys'


        print(f'---> Reporting metadata on topic
{partial_metadata_udpate_publish_topic}\nwith payload {metadata}')

        self.client.publish(topic=partial_metadata_udpate_publish_topic,
payload=metadata)


def main():
```

```
    # Initiate server connection

    print(f'Connecting to Kaa server at {KPC_HOST}:{KPC_PORT} using application
version {APPLICATION_VERSION} and endpoint token {ENDPOINT_TOKEN}')



    client_id = ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in
range(6))

    client = mqtt.Client(client_id=client_id)

    client.connect(KPC_HOST, KPC_PORT, 60)

    client.loop_start()



    metadata_client = MetadataClient(client)



    # Fetch current endpoint metadata attributes

    retrieved_metadata = metadata_client.get_metadata()

    print(f'Retrieved metadata from server: {retrieved_metadata}')



    # Do a partial endpoint metadata update

    metadata_to_report = json.dumps({"model": "BFG 9000", "mac": "00-14-22-01-23-
45"})

    metadata_client.patch_metadata_unconfirmed(metadata_to_report)



    time.sleep(5)

    client.disconnect()



if __name__ == '__main__':

    main()
```

9.4. ANEXO 4: CÓDIGO DE SEGURIDAD PREVIO INGRESO AL CORREO GMAIL.

Email: proyectonodeiot@gmail.com

Contraseña: nodeiot2021

Códigos de seguridad para ingreso al Gmail.

1. 0072 9629
2. 8455 8682
3. 7517 3405
4. 6685 5266
5. 2974 0560
6. 3400 3112
7. 5694 4698
8. 3758 4844
9. 6199 8166
10. 4115 9665

* Solo se puede usar cada código de seguridad una vez.

* ¿Necesitas más? Visita https://g.co/2sv

* Fecha de generación de los códigos: 15 jul. 2021

Código de ingreso para aplicativos de terceros.

app key bjfvpsxgvjzcxwwu // este key se ingresa en app password del fichero de node red.

9.5. ANEXO 5: CREDENCIALES DE ACCESO A KAAIOT

USUARIO: proyectonodeiot@gmail.com

CONTRASEÑA: nodeiot2021

URL: https://www.kaaiot.com/