SISTEMA DOMÓTICO

Trabajo Final

Universidad Nacional de Mar del Plata
Facultad Ingeniería
Departamento de Electrónica

Autores:
Distéfano Nahuel Esteban- Mat. n° 10659
Fernández Minich Facundo- Mat. n°11224

Directores:
Ing. Uriz Alejandro
Ing. Etcheverry Juan Alberto

SISTEMA DOMÓTICO

Trabajo Final

Universidad Nacional de Mar del Plata
Facultad Ingeniería
Departamento de Electrónica

Autores:
Distéfano Nahuel Esteban- Mat. n° 10659
Fernández Minich Facundo- Mat. n°11224

Directores:
Ing. Uriz Alejandro
Ing. Etcheverry Juan Alberto

ÍNDICE

Mar del Plata, Buenos Aires
Argentina 2025

SISTEMA DOMÓTICO

Tesis de grado. Facultad de Ingeniería. Departamento de Electrónica

Distéfano Nahuel Esteban - Fernández Minich Facundo

-RESUMEN-

El siguiente informe se encuentra enmarcado dentro de uno de los requisitos de la carrera Ingeniería Electrónica de la Universidad Nacional de Mar del Plata, el Trabajo Final. A lo largo del mismo se buscará dar cuenta del recorrido realizado para cumplimentar un objetivo, apuntando a explicitar cómo se fueron aplicando los conocimientos adquiridos durante la carrera y cómo fue sucediendo el proceso de aprendizaje, explicitando además las dificultades y obstáculos que se han ido presentando, cómo se fueron resolviendo y cuáles fueron los resultados finales del proceso realizado, siendo la presentación de este informe, la culminación del proceso.

El objetivo ha sido diseñar e implementar un sistema domótico integral. Para ello se programó un software, y se diseñó e implementó un hardware.

-INTRODUCCIÓN-

El presente informe busca explicitar cuál ha sido el proceso para cumplimentar el objetivo de diseñar e implementar un sistema domótico electrónico e informático capaz de controlar artefactos sin la necesidad de estar presentes en el lugar, y que además conviva con el sistema eléctrico tradicional. Para ello;

En el **capítulo 1** se delimita el problema sobre el cual se decidió trabajar, cuál es la situación a transformar, dando cuenta del contexto sobre el que se busca intervenir, presentando además los aspectos teóricos que fundamentan el desarrollo del sistema domótico. Se realiza además un breve repaso por la historia de la domótica, presentando algunas conceptualizaciones claves: sensores de corriente, controladores, tipos de comunicación y tipo de aplicaciones, necesarias para poder ahondar en el recorrido realizado.

En el **capítulo 2** se explican los aspectos metodológicos, describiendo los objetivos generales y específicos, y cada uno de los pasos, explicitando los aspectos procedimentales, hasta concluir con la puesta en funcionamiento del sistema domótico integral.

A lo largo del **capítulo 3**, se detallan las características de los dispositivos del sistema; desde el sistema hardware-software que consta de dispositivo de sensado, dispositivo de control y dispositivo de acción, hasta el desarrollo de la programación e interfaz de comunicación. Se describen además, todos los elementos utilizados para la creación del sistema, focalizando en la fabricación del sensor de corriente y la programación de todas las etapas de comunicación entre dispositivos.

El **capítulo 4** está dedicado a explicitar cuáles han sido los objetivos alcanzados, los obstáculos que hubo que sortear en el trayecto, y la descripción de cuáles se consideran que podrían ser hipótesis de trabajo para dar lugar a posibles mejoras.

En el **capítulo 5**, se realiza un breve análisis de la experiencia en el abordaje del presente proyecto, brindando conclusiones personales y grupales, en relación al trabajo realizado y a las vivencias sucedidas a lo largo del proceso.

Al finalizar, se incluyen las referencias bibliográficas consultadas y un **anexo** con el detalle de la programación completa de todo el sistema domótico realizado.

CAPÍTULO 1. DELIMITACIÓN DEL PROBLEMA. ASPECTOS TEÓRICOS

1.1 CONTEXTO. DELIMITACIÓN DEL PROBLEMA. SITUACIÓN A TRANSFORMAR

Los cambios tecnológicos en la historia de la humanidad han provocado modificaciones profundas en la sociedad. Es sabido que diversas transformaciones a nivel social, político y económico, acaecidas durante el proceso de la revolución industrial han ido generando el crecimiento de las grandes ciudades. La carrera aeroespacial, por ejemplo, generó un gran cambio en materia de comunicaciones a nivel global. Con la llegada de las computadoras personales e internet, el mundo se empezó a conectar de una manera nunca antes vista. En la actualidad ya no se habla solo de internet, sino que se hace referencia a un concepto más importante, el **Internet de las Cosas[1], lo que se denomina IoT, del inglés internet of things**.

IoT ha permitido manipular una gran cantidad de datos, no sólo de las grandes masas poblacionales, sino de forma individual, generando un nuevo concepto, **Big Data[2]**. El análisis y uso de estos datos abre la puerta a una nueva revolución donde la información es más valiosa que los bienes materiales. Ya no importa tanto quién tiene el capital sino quién tiene la información. Vivimos en una sociedad 2.0, una sociedad hiperconectada, no solo por la influencia de las redes sociales en nuestras vidas, sino por el concepto de IoT, que cada día toma más relevancia.

IoT nació en el Instituto de Tecnología de Massachusetts (MIT). No es una revolución tecnológica sino una revolución en las relaciones entre los objetos y las personas, incluso entre los objetos entre sí, conectándose mutuamente manipulando datos en tiempo real.

La tecnología, y en especial la conexión entre los dispositivos, cambian a diario el mundo, generando necesidades nuevas. Internet pasó a ser un servicio esencial tan

[1] Ashton, K. (2009). *That 'Internet of Things' Thing*. RFID Journal.
[2] Mayer-Schönberger, V., & Cukier, K. (2013). *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Houghton Mifflin Harcourt.

importante como la electricidad y el agua corriente, desplazando al servicio de cable y al servicio de telefonía fija.

Si a IoT le se le agrega el concepto de Big Data obtenemos una visión de cómo es la sociedad actual, una sociedad hiperconectada, una sociedad conectada a todo en todo momento. Si bien los siguientes conceptos no son nuevos, actualmente se ve una explosión en el campo de la URBÓTICA, INMOTICA y DOMOTICA debido a la aplicación de estos principios en la vida diaria.

La **Urbótica**[3] es una suerte de ciudad o comunidad que funciona por sí misma. Desde sistemas de alumbrado público y semáforos sincronizados, a la lectura del tráfico de forma automática, a través de inteligencia artificial, que permite descongestionar las arterias, generando una circulación fluida de los vehículos.

Cada vez más se ven edificios inteligentes, **Inmótica**[3], donde más allá del sistema de iluminación y calefacción, un edificio "sabe" qué vecino está entrando y si va o no a tomar el ascensor y a qué piso irá. El vecino entra al edificio sin la necesidad de que se presione un botón o se accione una puerta con llave mecánica.

Y por último la **Domótica**[3], que no es otra cosa que una casa inteligente o una casa que funciona por sí sola, como indica el origen de la palabra en latín, concepto base de este informe final.

Siguiendo a Andy Radowogski[4], la definición de vivienda domótica o inteligente se encuentra atravesada por diferentes aspectos. En distintos idiomas se utilizan diversos términos para aludir al concepto, como "casa inteligente" (smarthouse), automatización de viviendas (home automation), domótica (domotique), sistemas domésticos (home systems), etc.

Así, un sistema domótico permitirá, a través de una red de comunicación, la interconexión entre una serie de equipos, con el objetivo de obtener información acerca del entorno de un hogar y realizar acciones específicas sobre dicho entorno.

Teniendo en cuenta esta contextualización, se consideró importante enmarcar el proyecto buscando dar respuesta a dichas nuevas necesidades, centrando la mirada en la internet de las cosas dentro de un hogar.

---

[3] López, L., & Barchino, R. (2016). *Domótica, inmótica y urbótica: Automatización de viviendas, edificios y ciudades inteligentes*. Universidad de Castilla-La Mancha.
[4] Radowogski, A. (s.f.). *Viviendas Inteligentes*. http://rnds.com.ar/articulos/040/RNDS_124W.pdf

En la actualidad existen varios sistemas de aplicación para que un determinado ambiente se automatice. Hay protocolos de más de 50 años como el protocolo X10[5], en el que en un sistema eléctrico ya instalado se pueden enviar datos y órdenes a los dispositivos que se encuentren conectados sobre la misma línea, sin la necesidad de instalar un cableado nuevo o específico.

Las mejoras de tecnologías como WiFi (wireless fidelity)[6] o Bluetooth (tecnología de radio inalámbrica de corto alcance)[7] generaron avances importantes en la comunicación de los dispositivos. Dependiendo del ambiente en el que se implementará la tecnología, será la elección del dispositivo más propicio para su automatización.

Teniendo en cuenta el recorrido mencionado, se buscó implementar el sistema en una casa en construcción, realizando un proyecto "a medida", con el objetivo de diseñar de cero todo el sistema eléctrico y domótico, en beneficio del usuario y en respuesta a la necesidad propia de no solo trabajar en un software capaz de controlar el sistema, sino también aplicar otros conocimientos adquiridos en la carrera, en un hardware.

1.2. ASPECTOS TEÓRICOS.

1.2.1. DOMÓTICA

1.2.1.1. Definición. Beneficios.

El término domótica viene de la unión de las palabras domus, que significa "casa" en latín; y tica, palabra griega que significa "funciona por sí sola".

Como se dijo anteriormente, la gestión domótica consiste en poder modificar mediante control local o remoto, los parámetros de funciones de una vivienda, tales como:

- *Ahorro energético*: control de la iluminación de toda la casa, regulación de la temperatura, apagar o encender de forma eficiente cada electrodoméstico, entre otras cosas.

[5] Kumar, R., Kamal, R., & Arora, A. (2011). *Home automation system using X10 protocol*. International Journal of Computer Applications, 47(17), 27–31.
[6] https://www.intel.la/content/www/xl/es/products/docs/wireless/wi-fi-7.html
[7] https://www.intel.la/content/www/xl/es/products/docs/wireless/how-does-bluetooth-work.html

- *Seguridad*: control de monóxido de carbono, simuladores de presencia, alarma y monitoreo (custodia y vigilancia frente a la intrusión, la inundación, el fuego, los escapes de gas, etc.)
- *Confort*: programaciones horarias de la calefacción y también de los diferentes escenarios luminosos, riego automático por programación o por niveles bajos de humedad, aperturas de portones, etc.

Entonces la **Domótica** se encarga, no solo de la integración de los sistemas eléctricos y electrónicos del hogar, sino también del CONTROL de los mismos, aumentando aún más la sensación de confort, seguridad y ahorro energético.

1.2.1.2. Breve desarrollo Histórico

En 1975, de la mano del protocolo X10, se inicia un camino hacia la revolución domótica. Este protocolo está basado en corrientes de portadoras o PL (en inglés Power Line). El mismo se extendió mucho por Estados Unidos y en Europa (sobre todo Reino Unido y España).

La sencillez y sobre todo la accesibilidad al protocolo, derivó en múltiples aplicaciones tanto en software como en hardware.

Es importante mencionar, que esta tecnología o protocolo depende directamente de la calidad de la red eléctrica. Existen filtros que amortiguan los ruidos de la red eléctrica, o lo minimizan, pero nunca consiguen erradicarlo del todo. Otra característica es que no soporta funciones lógicas complejas.

Al mismo tiempo, empresas del sector eléctrico empezaron a utilizar la tecnología que utilizaban en sus fábricas para aplicarlas al hogar, así nace EIB (European Installation Bus), Batibus (Francia e Italia) y EHS (European Home System)[8].

Estas tres Asociaciones decidieron unir fuerzas a finales de los 90, y crear un estándar común. En 2002, la KNX Association[9], con sede en Bruselas, presentó el nuevo estándar, que está basado en buena parte en la tecnología EIB, y reforzado con

[8] https://www.actamechanica.sk/pdfs/ams/2013/03/07.pdf
[9] https://www.knx.org/

los sistemas de transmisión de datos que ofrecían BatiBus y EHS. Durante un corto periodo de tiempo, este sistema se llamó “Konnex”.

A la par, en Estados Unidos, nacía Lonworks[10], que tuvo sus orígenes entre los productos que fabricaba Echelon Corporation.

En 1999 el protocolo de comunicación (llamado LonTalk[11]) fue normalizado como estándar de control de redes según la norma: ANSI/CEA-709.1-B (estándar americano). Posteriormente, otras aplicaciones del protocolo han sido normalizadas (desde transmisión por Power-Line o par trenzado, hasta control de trenes y frenos electroneumáticos), hasta convertirse en un estándar europeo para domótica en 2005 (EN 14908) para luego ser estándar mundial bajo la ISO/IEC-14908.

En Europa aparecieron marcas asociadas a los sistemas propietarios, algunos verdaderamente ingeniosos usando el protocolo IPV4 .

A partir del año 2000, se vió un fuerte incremento de pequeñas empresas cubriendo nuevos nichos de mercado, compitiendo en precios, enriqueciendo la gama de productos. A partir del año 2006 comienzan a surgir los **sistemas domóticos inalámbricos** RF (wireless), usando protocolos como: Zigbee (Control4) y Zwave. Algunos se crean compatibles con X10 y el resto son complementarios al sistema cableado o sistemas RF independientes.

En los últimos años de la década anterior, se observó un fuerte desarrollo de sistemas complementarios relacionados con la domótica, como automatismos de persianas y puertas, o iluminación con normalización internacional (ISO), ampliando su mercado y pasando de sistemas aislados de "regulación local" a "sistemas integrados" en estándares domóticos.

En la década actual, la transmisión dejó de ser por el protocolo IPV4[12] y se empezó a aplicar el protocolo IPV6[12], permitiendo el envío de datos de voz y vídeo, entre otras cosas; ampliando así las funciones y su complejidad.

En nuestro país, la domótica tuvo sus inicios en los ‘90. Por la situación económica y el desconocimiento del consumidor, ésta no tuvo la demanda esperada que sí tuvo en países de la región. Desde hace unos años, la domótica comenzó a

[10] https://ieeexplore.ieee.org/abstract/document/866025
[11] https://ieeexplore.ieee.org/abstract/document/6804465
[12]https://repository.ucc.edu.co/server/api/core/bitstreams/f0b86f4d-f3f3-4724-a902-783e99dd04d6/content

experimentar una constante y positiva evolución. En la actualidad, las empresas que se dedican a este mercado, están viendo cómo la domótica comienza a ser demandada tanto por usuarios finales como por los promotores inmobiliarios y constructoras. Respecto de mercados como el europeo, nuestro país tiene un promedio de atraso de diez años en la incorporación de tecnología domótica. Aunque incorporarse al segmento con demora podría significar un atraso respecto de otras partes del mundo, la realidad es que se está muy a la par de otros países, ya que se importan e instalan sistemas que ya tienen un funcionamiento comprobado. Es decir que lo que llega a nuestro mercado son equipos y tecnologías probadas, que se sabe que funcionan, lo cual reduce considerablemente esa aparente brecha inicial.

1.2.2. SENSORES ACTUADORES Y CONTROLADORES

Ahora bien, más allá del protocolo o tecnología a utilizar, para automatizar un ambiente se necesitan SENSORES que “escuchen” los estados de los dispositivos, un sistema o CONTROLADOR que “interprete” estos datos y además manipule ACTUADORES externos para modificar estados de ciertos elementos. Así se decidió dividir el trabajo para un mayor entendimiento y simplificación de las tareas a realizar.

1.2.2.1. Sensor de Tensión-Corriente

Una de las tareas más importantes, es la obtención de datos de la línea y sus cargas, estos datos sirven para analizar el sistema y así tomar decisiones al respecto. Las mediciones se pueden hacer directa o indirectamente. Cada una de ellas tendrá sus beneficios y sus desventajas.

1.2.2.2. Sensores efecto Hall

El efecto Hall es un fenómeno que ocurre cuando la corriente fluye a través de un conductor y es expuesta a un campo magnético, la tensión que aparecerá, estará presente de forma perpendicular a la corriente y el campo magnético.

Si se conoce el valor de la corriente, entonces, se puede calcular la fuerza del campo magnético; si se crea el campo magnético por medio de corriente que circula por

una bobina o un conductor, entonces, se puede medir el valor de la corriente en el mismo.

Ilustración 1:Sensor de Corriente Efecto Hall

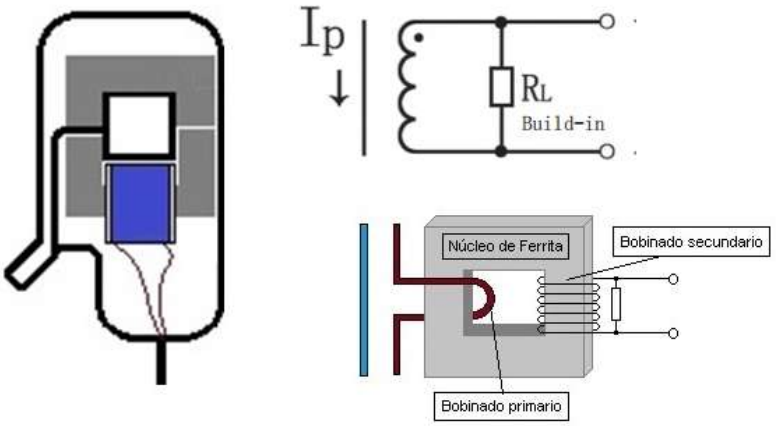

Ilustración 2: Circuito interno de un sensor de corriente de Efecto Hall.

Si bien estos dispositivos son ideales para trabajar, debido a que no es necesario abrir el circuito para la medición, sus costos son elevados y solamente podemos medir un dispositivo por sensor.

1.2.2.3. Sensor de corriente indirectamente al medir tensión

A diferencia del sensor por efecto Hall, que no carga el circuito y no es necesario la apertura del mismo para sensar, de manera indirecta se puede saber si circula corriente, midiendo la tensión que genera el paso de la misma. Para ello se necesita abrir el circuito e intercalar un dispositivo, como un optoacoplador.

Ilustración 3: Sensor de Corriente Arduino ACS712

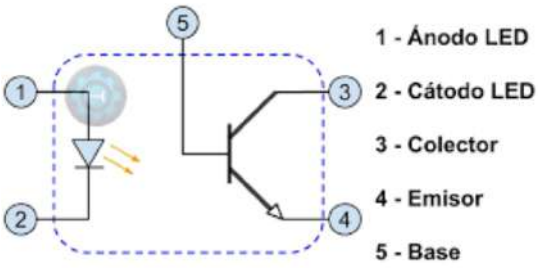

Ilustración 4: Optoacoplador Dispositivo Interno

Un optoacoplador es un componente electrónico que se utiliza como transmisor y receptor óptico (de luz), es decir, pueden transmitir de un punto a otro una señal eléctrica sin necesidad de conexión física, mediante una señal luminosa.

Abriendo el circuito e intercalando en serie un diodo emisor se podrá generar una luz que activará, por efecto luminoso, otro diodo receptor. Este dispositivo, no solo permite saber en qué estado se encuentra el circuito principal, sino que además, aísla el mismo del circuito secundario, protegiendo el circuito de baja potencia de una de mayor potencia.

El dispositivo de Arduino ACS712 –*Ilustración 3*- se puede encontrar en módulos, los cuales nos facilitan sus conexiones, traen una bornera para conectar la línea que queremos medir y tres pines, dos para conectar la alimentación y un pin para la salida analógica.

1.2.2.4. Actuadores

Los actuadores son dispositivos que modifican el estado eléctrico de un aparato. En el mercado existen varios de estos, uno puede ser un Relé -*Ilustración 5*-. Este dispositivo electromagnético que, estimulado por una corriente eléctrica muy débil, abre o cierra un circuito en el cual se disipa una potencia mayor que en el circuito estimulador, permitiendo así, separar una etapa de alta potencia de otra que no lo es y protegiendo, por lo tanto, a esta última. Y, como se vio en los sensores de corriente, también utilizan optoacopladores para tal fin, aunque también pueden utilizar bobinados.

Ilustración 5: Relé Familia Arduino

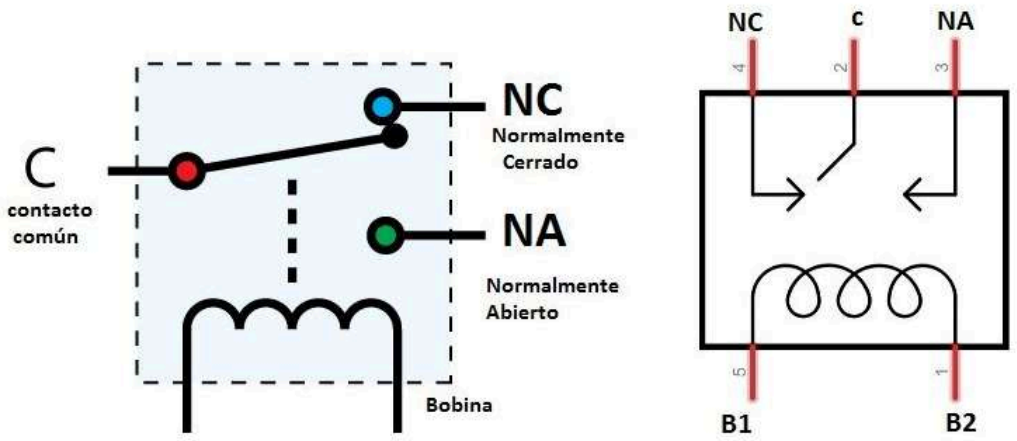

Ilustración 6: Rele Internamente

1.2.2.5. Controlador

Un dispositivo capaz de guardar datos, procesarlos, y según alguna lógica, proceder a realizar ciertas acciones es lo que se llama un "Controlador". El mismo debe ser tal que permita el uso de periféricos, es decir, la conexión con otros dispositivos, como sensores y actuadores. Las placas de controladores más implementadas en el mercado electrónico de código abierto, son la placa Arduino y la Raspberry.

Ilustración 7: Placas Arduino (izquierda) y Placa Raspberry (derecha)

1.2.2.5.1. Arduino

Arduino [13]-*Ilustración 7 izquierda*- es una plataforma de creación de electrónica de código abierto, basada en hardware y software libre, lo que permite que cualquiera pueda utilizarlos y adaptarlos, sin inconvenientes. Existen en el mercado varios tipos de placas, accesorios y aplicaciones compatibles, creadas por diferentes empresas o desarrolladores. Todas ellas son diferentes, pero utilizan la misma base común, lo que ayuda a que la comunidad de creadores pueda darles diferentes tipos de uso.

Arduino ofrece la plataforma Arduino IDE (Entorno de Desarrollo Integrado), que es un entorno de programación con el que cualquiera puede crear aplicaciones.

[13] https://www.arduino.cc/

También permite hacer conexiones virtuales para comprobar cómo se comportaría con el código que se ha escrito.

Es una placa con todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador, y que puede ser programada tanto en Windows como macOS y GNU/Linux.

A continuación, se describen en detalle las características, tanto para la placa Mega como la Due.

El **Arduino Mega 2560** es una placa de desarrollo basada en el microcontrolador ATmega2560. Tiene 54 entradas/salidas digitales (de las cuales 15 pueden ser usadas como salidas PWM), 16 entradas analógicas, 4 UARTs, un cristal de 16Mhz, conexión USB, jack para alimentación DC, conector ICSP, y un botón de reseteo. La placa Mega 2560 es compatible con la mayoría de shields de versiones anteriores.

Arduino Mega posee las siguientes especificaciones:

- Microcontrolador: ATmega2560
- Voltaje Operativo: 5V
- Voltaje de Entrada: 7-12V
- Voltaje de Entrada(límites): 6-20V
- Pines digitales de Entrada/Salida: 54 (de los cuales 15 proveen salida PWM)
- Pines análogos de entrada: 16
- Corriente DC por cada Pin Entrada/Salida: 40 mA
- Corriente DC entregada en el Pin 3.3V: 50 mA
- Memoria Flash: 256 KB (8KB usados por el bootloader)
- SRAM: 8KB
- EEPROM: 4KB
- ClockSpeed: 16 MHz

Además ofrece algunos pines específicos para la conexión de un circuito externo. Esos pines son:

- VIN: A través de este pin es posible proporcionar alimentación a la placa.
- 5V: Podemos obtener un voltaje de 5V y una corriente de 40mA desde este pin.

- 3.3V: Podemos obtener un voltaje de 3.3V y una corriente de 50mA desde este pin.
- GND: El ground (0V) de la placa.

Ilustración 8: Placa Arduino Mega

El **Arduino Due** posee 54 pines digitales de entrada y salida (de los cuales 12 pueden ser usados como salidas PWM), 12 entradas análogas, 2 salidas análogas, 4 UART (puertas seriales por hardware), cristal oscilador de 84MHz, una conexión compatible con USB-OTG, 2 TWI, Jack de poder, conexión JTAG, botón reset y un botón borrar. También tiene otras funcionalidades como audio, DMA, una librería experimental para multitareas y más.

La lista completa de especificaciones es la siguiente:

- Microcontrolador: AT91SAM3X8E
- Tensión de trabajo: 3.3V
- Tensión recomendada de entrada: 7-12V
- Tensión de entrada (mín/máx): 6-20V
- Digital I/O Pins: 54 (of which 6 provide PWM)
- Pines de entradas analógicas: 12
- Pines de salidas analógicas: 2 (DAC)
- Corriente total soportada en todas las líneas DC I/O: 130 mA
- DC intensidad para 3.3V Pin: 800 mA

- DC intensidad para 5V Pin: teórica 1A, recomendada 800 mA
- Flash Memory: 512 KB
- SRAM: 96 KB (64 + 32 KB)
- Clockspeed: 84 MHz
- Debugaccess: JTAG/SWD connector
- Controlador DMA, que puede aliviar a la CPU de realizar tareas que requieren mucha memoria.

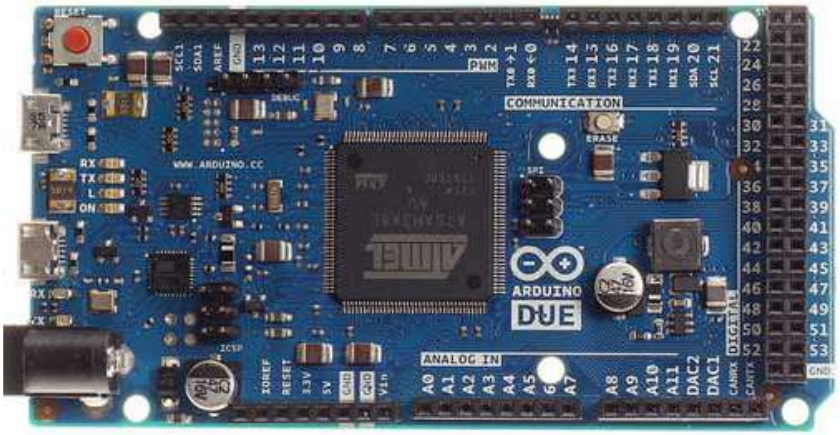

Ilustración 9: Arduino Due

Tabla 1. Comparativa entre Arduino Mega 2560 y Arduino Due

Característica	Arduino Mega 2560	Arduino Due
Microcontrolador	ATmega2560	AT91SAM3X8E
Voltaje de operación (E/S)	5 V	3.3 V
Voltaje de entrada recomendado	7–12 V	7–12 V
Voltaje de entrada máximo	6–20 V	6–20 V
Cantidad de pines digitales (E/S)	54	54

Pines PWM digitales	15	12
Entradas analógicas	16	12
Salidas analógicas (DAC)	No disponible	2
Corriente máxima total en pines E/S	40 mA	130 mA
Corriente máxima por pin 3.3 V	50 mA	800 mA
Corriente máxima por pin 5 V	No especificado	800 mA
Memoria Flash	256 KB (8 KB para bootloader)	512 KB
Memoria SRAM	8 KB	96 KB (64 KB + 32 KB)
Memoria EEPROM	4 KB	No disponible
Frecuencia del reloj del procesador	16 MHz	84 MHz
Tipo de conector USB	USB tipo B estándar	Micro USB

1.2.2.5.2. Raspberry Pi

Raspberry Pi –*Ilustración 7 derecha*- es un ordenador de placa simple y bajo costo, desarrollado en Reino Unido. Es lo suficientemente potente como para facilitar el aprendizaje y realizar tareas básicas, y también permite programar y compilar programas que se ejecuten en él.

Al igual que Arduino es de tamaño reducido y la posibilidad de conectarle varios tipos de accesorio, le dan una versatilidad que permite utilizarse para varios tipos de tareas. Su misión principal es la de enseñar informática en las aulas.

1.2.2.5.3. Diferencias entre Arduino y Raspberry Pi

Ambas placas utilizan software de código abierto, pero solo Arduino implementa además, la filosofía de hardware abierto.

Las Raspberry Pi tienen más potencia de cálculo que las placas Arduino, y cuentan con conectividad WiFi y Ethernet integradas en las placas.

En el caso de Arduino, para obtener dicha conectividad Ethernet, existen las llamadas placas de expansión.

También hay diferencias relacionadas con el software. Arduino ejecuta inmediatamente la tarea para la cual fue programada, mientras que Raspberry Pi requiere de un sistema operativo completo. Esto repercute directamente en los proyectos de electrónica. Algo tan sencillo como hacer que un LED se encienda o apague es extremadamente fácil en Arduino, ya que sólo tienes que conectarlo junto a una resistencia y añadir unas pocas líneas de código. Mientras que, para hacer lo mismo en una Raspberry Pi, hay que descargar e instalar las librerías para controlar los puertos donde lo vayas a conectar, compilar el programa y ejecutarlo.

1.2.3. TIPOLOGÍAS DE COMUNICACIÓN

1.2.3.1. Arquitecturas

Existen diferentes arquitecturas para la comunicación entre las partes del sistema.

• Centralizada: Un controlador recibe información de múltiples sensores y, una vez procesada, genera las órdenes necesarias hacia los dispositivos encargados de cumplirlas.

• Distribuida: La inteligencia del sistema se encuentra distribuida en los módulos que componen el sistema, ya sean sensores o actuadores. Este tipo de inteligencia suele ser típica en los sistemas de cableado en bus o redes inalámbricas.

• Mixta: Ese tipo de sistemas poseen una arquitectura descentralizada, ya que disponen de varios pequeños dispositivos capaces de adquirir y procesar la información de múltiples sensores, y transmitirlos al resto de dispositivos distribuidos por la vivienda. Se aplica generalmente en aquellos sistemas domóticos basados en Zigbee y totalmente inalámbricos.

1.2.3.2. Interconexión

En toda instalación domótica debe existir un medio de conexión entre cada uno de los dispositivos que la integran. Estos medios pueden ser cableados o inalámbricos, y cada uno de ellos presenta diferentes variantes tecnológicas.

Sistemas cableados:

- **xDSL.** DSL (Digital Subscriber Line, Línea de abonado digital) es un término utilizado para referirse de forma global a todas las tecnologías que proveen una conexión digital sobre línea de abonado de la red telefónica local: ADSL, ADSL2, ADSL2+ SDSL, IDSL, HDSL, SHDSL, VDSL y VDSL2. Tienen en común que utilizan el par trenzado de hilos de cobre convencionales de las líneas telefónicas, para la transmisión de datos a gran velocidad. La diferencia entre ADSL y otras DSL, es que la velocidad de bajada y la de subida no son simétricas, es decir que normalmente, permiten una mayor velocidad de bajada que de subida.

- **Fibra óptica.** Es un conductor de ondas en forma de filamento, generalmente de vidrio, aunque también puede ser de materiales plásticos. Es capaz de dirigir la luz, emitida por un láser o un LED, a lo largo de su longitud, usando la reflexión total interna. Este método es ampliamente utilizado en telecomunicaciones, ya que permite enviar gran cantidad de datos a gran velocidad, mayor que las comunicaciones de radio y cable, y es un medio inmune a las interferencias.

- **Coaxial**. Es un cable eléctrico formado por dos conductores concéntricos, uno central o núcleo, formado por un hilo sólido o trenzado de cobre (llamado positivo o vivo), y uno exterior en forma de tubo o vaina, y formado por una malla trenzada de cobre o aluminio o bien por un tubo, en caso de cables semirrígidos. Este último, produce un efecto de blindaje y además sirve como retorno de las corrientes. El primero está separado del segundo por una capa aislante, llamada dieléctrico. De la calidad del dieléctrico dependerá principalmente, la calidad del cable. Y todo el conjunto puede estar protegido por una cubierta aislante.

- **Par trenzado**. Es una forma de conexión en la que dos conductores son entrelazados para cancelar las interferencias electromagnéticas (IEM) de fuentes

externas y la diafonía de los cables adyacentes. El entrelazado de los cables disminuye la interferencia, debido a que el área de bucle entre los cables, que determina el acoplamiento magnético en la señal, es reducido. En la operación de balanceado de pares, los dos cables suelen llevar señales iguales y opuestas (modo diferencial), las cuales son combinadas mediante sustracción en el destino. El ruido de los dos cables se cancela mutuamente en esta sustracción, debido a que ambos cables están expuestos a IEM similares. La tasa de trenzado, usualmente definida en vueltas por metro, forma parte de las especificaciones de un tipo concreto de cable. Cuanto mayor es el número de vueltas, mayor es la atenuación de la diafonía. Donde los pares no están trenzados, como en la mayoría de conexiones telefónicas residenciales, un miembro del par puede estar más cercano a la fuente que el otro y, por tanto, expuesto a niveles ligeramente distintos de IEM.

Sistemas inalámbricos

- **Wi-Fi**. Es un sistema de envío de datos sobre redes computacionales, que utiliza ondas de radio en lugar de cables. El término Wi-Fi no proviene de Wíreless Fidelity, como comúnmente se cree, sino que es solo el nombre del estándar que certifica la interoperabilidad de equipos según la norma IEEE 802.11b bajo la marca Wi-Fi. Esto quiere decir que el usuario tiene la garantía de que todos los equipos que tengan el sello Wi-Fi pueden trabajar juntos sin problemas, independientemente del fabricante de cada uno de ellos.

- **GPRS**. General Packet Radio Service (GPRS) es un servicio de datos móvil orientado a paquetes. Está disponible para los usuarios del Sistema Global para Comunicaciones Móviles (Global System for Mobile Communications o GSM), así como para los teléfonos móviles que incluyen el sistema IS-136. Permite velocidades de transferencia de 56 a 114 kbps.

- **Bluetooth**. Es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia segura y globalmente libre (2,4 GHz.). Los principales objetivos de esta norma son facilitar las comunicaciones entre equipos móviles y fijos, eliminar cables y conectores entre éstos, posibilitar pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos.

- **Radiofrecuencia**. El término radiofrecuencia, también denominado espectro de radiofrecuencia o RF, se aplica a la porción menos energética del espectro electromagnético, situada entre unos 3 Hz y unos 300 GHz. Las ondas electromagnéticas de esta región del espectro, pueden transmitirse aplicando la corriente alterna originada en un generador a una antena.

- **Infrarrojos**. La radiación infrarroja, radiación térmica o radiación IR es un tipo de radiación electromagnética de mayor longitud de onda que la luz visible, pero menor que la de las microondas. Un uso muy común es el que hacen los comandos a distancia que generalmente utilizan los infrarrojos en vez de ondas de radio ya que no interfieren con otras señales como las señales de televisión. Los infrarrojos también se utilizan para comunicar a corta distancia los ordenadores con sus periféricos

- **ZigBee**. Es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (wireless personal area network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

1.2.4. TIPOS DE APLICACIONES MÓVILES

Existen tres tipos de aplicaciones, para ser utilizadas en los celulares o en una Computadora o tablet, como soporte del programa para el usuario. Estas pueden ser aplicaciones NATIVAS, WEB, o mixtas.

1.2.4.1. App nativas

Una aplicación nativa es la que se desarrolla de forma específica para un determinado sistema operativo, llamado *Software Development Kit* o SDK. Cada una de las plataformas, Android, iOS o Windows Phone, tienen un sistema diferente, con lo cual, para que la app esté disponible en todas las plataformas, se deberán de crear varias apps con el lenguaje del sistema operativo seleccionado.

- Las apps para iOS se desarrollan con lenguaje Objective-C

- Las apps para Android se desarrollan con lenguaje Java
- Las apps en Windows Phone se desarrollan en .Net

Tabla 2. Ventajas y Desventajas App Nativa

VENTAJAS	INCONVENIENTES
Acceso completo al dispositivo	Diferentes habilidades - idiomas - herramientas para cada plataforma de destino
Mejor experiencia del usuario	Tienden a ser más caros en su desarrollo
Visibilidad en los Stores	El código del cliente no es reutilizable entre las diferentes plataformas
Envío de notificaciones a usuarios Actualización constante de App	

Cuando en este trabajo se habla de desarrollo móvil se está refiriendo a aplicaciones nativas. La principal ventaja con respecto a los otros dos tipos, es la posibilidad de acceder a todas las características del hardware del móvil: cámara, GPS, agenda, dispositivos de almacenamiento y muchas otras.

Además, las aplicaciones nativas no necesitan conexión a internet para que funcionen.

La descarga e instalación de estas apps se realiza siempre a través de las tiendas de aplicaciones (app store de los fabricantes).

1.2.4.2. Web App

Una aplicación web o webapp es la desarrollada con lenguajes muy conocidos por los programadores, como es el HTML, Javascript y CSS. La principal ventaja con respecto a la nativa, es la posibilidad de programar independiente del sistema operativo en el que se usará la aplicación. De esta forma, se pueden ejecutar en diferentes dispositivos, sin tener que crear varias aplicaciones.

Las aplicaciones web se ejecutan dentro del propio navegador web del dispositivo a través de una URL, y su contenido se adapta a la pantalla adquiriendo un aspecto de navegación APP.

Tabla 3. Ventajas y Desventajas Web App

VENTAJAS	INCONVENIENTES
El mismo código base reutilizable en múltiples plataformas	Requiere de conexión a internet
Proceso de desarrollo más sencillo y económico	Acceso muy limitado a los elementos del hardware
No necesita ninguna aprobación externa para publicarse (a diferencia de varias nativas para estar presente en los stores)	El tiempo de interacción del usuario es menor que una app nativa
El usuario siempre dispone de la última versión	
Pueden reutilizarse sitios responsive ya diseñados	

1.2.4.3. Web App nativa

Una aplicación híbrida es una combinación de las dos anteriores, se podría decir que recoge lo mejor de cada una de ellas. Las apps híbridas se desarrollan con lenguajes propios de las webapp, es decir, HTML, Javascript y CSS, por lo que permite su uso en diferentes plataformas, pero también dan la posibilidad de acceder a gran parte de las características del hardware del dispositivo. La principal ventaja es, que a pesar de estar desarrollada con HTML, Java o CSS, es posible agrupar los códigos y distribuirla en app store.

Tabla 4. Ventajas y Desventajas Web App Nativas

VENTAJAS	INCONVENIENTES
Es posible distribuirla en las tiendas virtuales	Experiencia del usuario más propia de la aplicación web que de la app nativa
Instalación nativa pero construida con HTML, Java o CSS	Diseño visual no siempre relacionado con el sistema operativo con el que se muestre
El mismo código base para múltiples plataformas	
Acceso a partes del hardware del dispositivo	

CAPÍTULO 2. ASPECTOS METODOLÓGICOS

2.1. OBJETIVOS.

2.1.1. Objetivo General.

El principal objetivo es diseñar e implementar un sistema domótico electrónico e informático, capaz de controlar artefactos, sin la necesidad de estar presentes en el lugar, y que además, conviva con el sistema eléctrico tradicional.

2.1.2. Objetivos Específicos.

- Permitir activar o desactivar aparatos, luces, motores; a distancia o en presencia.
- Facilitar las tareas rutinarias, mejorando el confort del hogar.
- Generar un sistema de simulación de presencia.
- Ser un sistema abierto a posibles modificaciones.
- El sistema domótico tiene que ser capaz de ser “invisible” al sistema eléctrico tradicional.

2.2 PROCEDIMIENTO. PLAN DE ACCIÓN. ESTRATEGIAS UTILIZADAS.

2.2.1. Plataforma de Control

Se decidió utilizar la Plataforma de código abierto Arduino y se involucró con su entorno. Verificamos la existencia de muchos complementos o “shields”, especialmente diseñados para determinadas funciones, como son los complementos de comunicación, de almacenamiento, de transferencia de datos, de acción, etc. Particularmente se adquirieron dos de estos, que fueron fundamentales en el accionar: los sensores de corriente y los relés.

Se profundizó sobre los aspectos del IDE de Arduino ("Entorno de Desarrollo Integrado" o "Integrated Development Environment"), sus sentencias y estructura, y se procedió a realizar pruebas en protoboard.

2.2.2. Circuito Escalera

Para que la convivencia del sistema eléctrico con el electrónico sea efectiva fue necesario diseñar un "circuito escalera"-*Ilustración 10*-, de modo tal que funcione, tanto con la llave analógica de dos puntos o con la llave electrónica o relé.

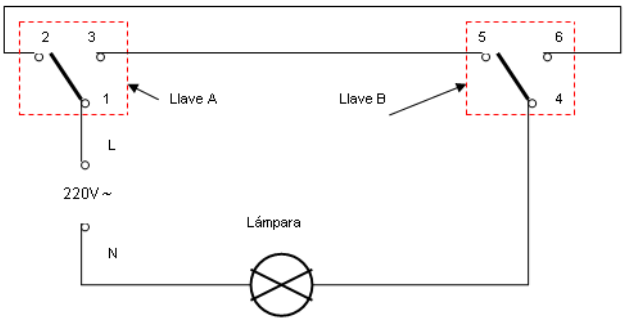

Ilustración 10. Circuito Escalera

Ilustración 11: Tecla analógica de dos puntos

Ilustración 12: Llave electrónica o Relé

2.2.3. Sensado

Una vez que nuestros sistemas eléctrico y electrónico convivieron en perfecta armonía, se procedió a sensar la corriente manipulada, y de esa manera poder documentar los estados de los distintos puntos a controlar.

En un primer momento, se utilizaron sensores de corriente comerciales. Se dispuso de los sensores ACS712-05A, ACS712-20A o el ACS712-30A, pero ninguno arrojó lecturas adecuadas debido a los niveles de ruido eléctrico presentes en el domicilio donde se realizaron las pruebas.

A raíz de esta dificultad, fue necesario **diseñar un prototipo de sensor que cumpla con los requerimientos** que el sistema demandaba, teniendo en cuenta el contexto en donde iba a ser aplicado.

Por cada dispositivo son dos elementos los que se desean medir, estado de la LÍNEA y estado de la CARGA. “Línea” alude a la CA220V 50Hz que proveen las distribuidoras de energía eléctrica. Y se menciona “carga” cuando se habla de una lámpara a controlar, por ejemplo.

Lo que se pretende saber es si un dispositivo está encendido o no, es decir si consume electricidad. El sensor debe ser tal que mida el paso de corriente por un circuito conectado a la línea alterna de 220V y envíe una señal continua de unos pocos volts.

Se estudió el comportamiento de sensores de corriente por efecto Halls o la implementación de bobinados para generar corrientes inducidas. Estos dos sensores permitían separar lo que era un circuito de potencia de uno que no lo era y de esa manera proteger este último. Los resultados no fueron los esperados debido al bajo consumo de la tecnología Led, que no permitía tener mediciones precisas del estado del circuito.

Se decidió diseñar un sensor propio acorde a las condiciones de operación (como se dijo anteriormente). Este debería ser tal que tome una muestra pequeña para no cargar al circuito eléctrico y entregar una tensión constante de unos pocos volts a la placa Arduino, separando etapas de potencia (seguridad de dispositivos).

Para ello, se trabajó con un optoacoplador, cuya función principal es la de separar partes de un circuito, en este caso dos secciones, con características eléctricas no compatibles. El Optoacoplador utilizado es el CNY74-4

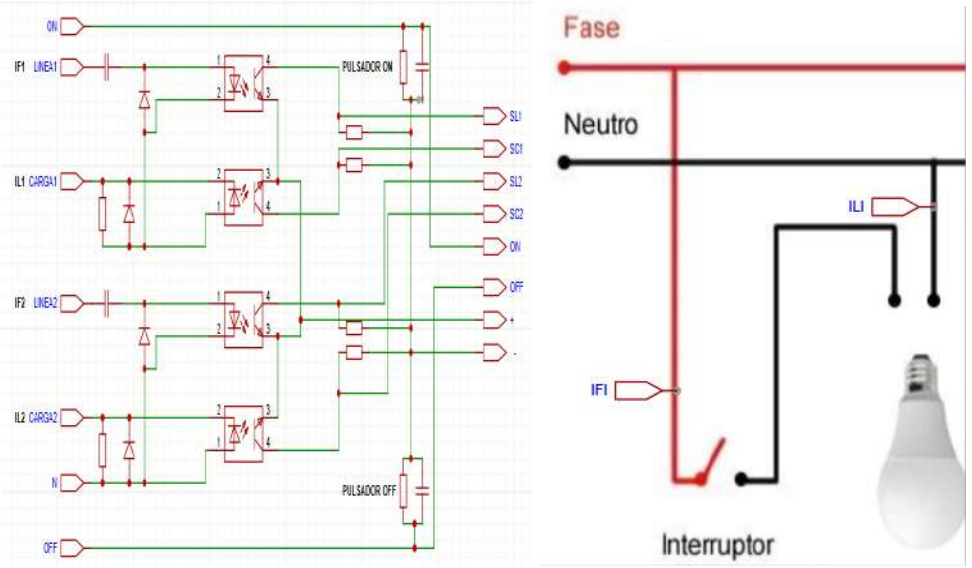

Ilustración 13:Circuito Sensor

2.2.4. Programación Muestra de Estados en Monitor Serie IDE

Para poder mostrar los estados, y que estos se mantengan estables para poder manipularlos y/o mostrarlos, se realizó un promedio en un determinado lapso de tiempo. De esta manera, no solo se sensó en tiempo real línea y carga, sino que también se

guardó dichos estados en un arreglo de datos que se utilizaron a posteriori por el programa central del sistema domótico diseñado.

Además, utilizando el Monitor Serie del IDE con el que se trabajó, se observaron los valores de las distintas señales.

2.2.5. Programación de Muestra de Estados en Dirección IP

Una vez lograda la muestra de estados en el monitor serie, se propuso mostrar los datos en una dirección IP determinada. De esta manera cualquier persona (dentro de la red de trabajo) que colocara dicha dirección en su barra de direcciones de cualquier navegador, podrá observar la información en cuestión.

2.2.5.1. Programación de Acceso a Dirección IP desde ubicación Remota

La idea fundamental del proyecto se basa en poder operar el sistema desde una ubicación remota, y no solamente dentro del domicilio. El siguiente paso fue lograr acceso, utilizando internet como medio de comunicación. Tanto el sistema propio como los dispositivos involucrados, como también los routers o servidores que haya en el medio, manejan distintos idiomas. Por dicho motivo es que logró que los distintos lenguajes de programación involucrados, convivan entre sí. Para lograr esto se estableció una comunicación, unificando criterios y variables a utilizar; así también como direccionar los datos hacia donde realmente tienen que ir, y que el ente que reciba esos datos sepa perfectamente qué tipo de datos son, de qué tamaño y sobre todo, qué es lo que tiene que hacer con él (guardarlo, cambiar su estado, etc…).

2.2.6. Programación de WebApp para mostrar Estados y Modificarlos

Una vez cumplido el objetivo anterior, de acceder desde una ubicación remota, surge la necesidad de modificar los estados de los elementos a controlar. Es por ello que se procedió a la programación de una interfaz sencilla con tal fin. Se la llamó página web o webapp, como se dijo anteriormente, y en ella se puede observar con claridad los estados de las luces, y los botones de ON y OFF para accionar cada uno de dichos elementos.

CAP 3. DESARROLLO E IMPLEMENTACIÓN

El funcionamiento del sistema, puede desglosarse en un conjunto de etapas o procesos. Estos procesos son: el desarrollo del HARDWARE, la programación del SOFTWARE, la COMUNICACIÓN entre los distintos actores y la INTERFAZ GRÁFICA que ve el usuario.

A continuación, se muestra el proceso de constitución de cada una de estas etapas.

3.1 HARDWARE

La primera etapa a su vez se subdividió en tres partes, definiendo así las responsabilidades de cada módulo, su comunicación y su ubicación en el hogar.

- Hardware de CONTROL
- Hardware de ACCIÓN
- Hardware de SENSADO

3.1.1. Hardware Control

Se decidió implementar este módulo con el controlador Arduino, en particular la placa MEGA2560, que presenta características acordes a este proyecto. Además, se agregó el complemento o shield de comunicación Ethernet, y también una Sim Card donde se encuentra almacenada la programación de la interfaz o webapp.

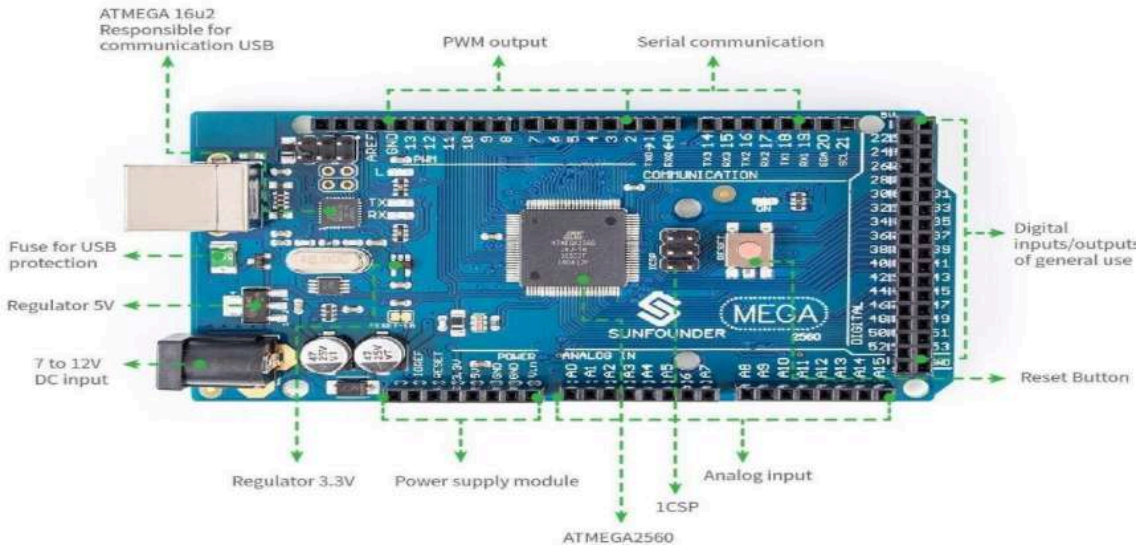

Ilustración 14. Partes de Arduino Mega

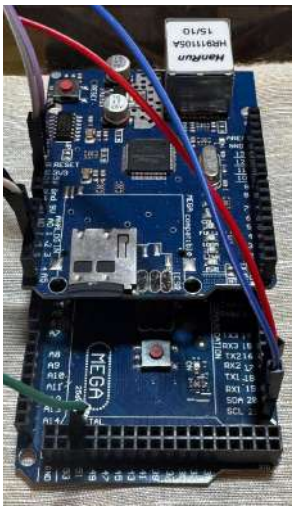

Ilustración 15. Arduino Mega y Eternet Shield

3.1.2. Unidad sensado o hardware de sensado

En el circuito de la ilustración 13, se puede observar, como una parte del mismo testea si hay corriente en la fase (IF) mientras que otra parte hace lo mismo con la carga (IL). Esto permite saber si un dispositivo se encuentra apagado, prendido o quemado/desconectado, como muestra la siguiente tabla de estados.

Tabla 5. Estados de Dispositivos

IF FASE	IL CARGA	ESTADO DISPOSITIVO
0	0	APAGADO
0	1	ERROR
1	0	QUEMADO
1	1	PRENDIDO

Algunas salvedades:

Debido a las características de la resistencia RX de potencia (resistencia conectada a la línea), se eligió trabajar con reactancias capacitivas a la frecuencia de trabajo de 50Hz, disminuyendo así, la pérdida de calor y el tamaño del circuito.

Debido a la baja resistencia de RZ (resistencia conectada a la carga), no se pudo implementar un capacitor como reactancia a la frecuencia de trabajo, como se realizó en RX. Pero en este caso, el calentamiento sobre el componente, fue imperceptible.

Para la carga, se utilizó una lámpara LED comercial de la marca Yarlux. La Resistencia calculada arrojó 22k ohms aprox. La otra de 1k ohms se calculó en 3,4k ohms, limitando la corriente y tensión.

Se utilizó un capacitor de 150nF (reactancia capacitiva de 22k a 50Hz) y funcionó como se esperaba.

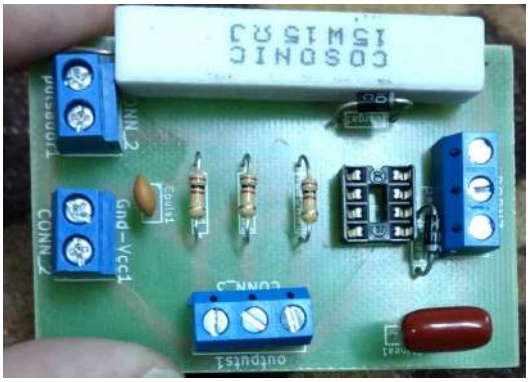

Ilustración 16. Sensor para 1 dispositivo

Ilustración 17. Sensor para 2 dispositivos

Lista de componentes utilizados con sus respectivas características:

1 OPTOACOPLADOR CNY74-4

1 RESISTENCIA DE POTENCIA DE 22K

2 RESISTENCIAS DE 1K

2 DIODOS 1N400X

2 PINES DE ENTRADA DE CONTROL ARDUINO A/D

3.1.3. Unidad de Acción

Se utilizó el relé de la familia Arduino como llave electrónica en el circuito escalera. Una señal de tensión baja en las entradas del relé, acciona la llave que cierra o abre el circuito del hogar.

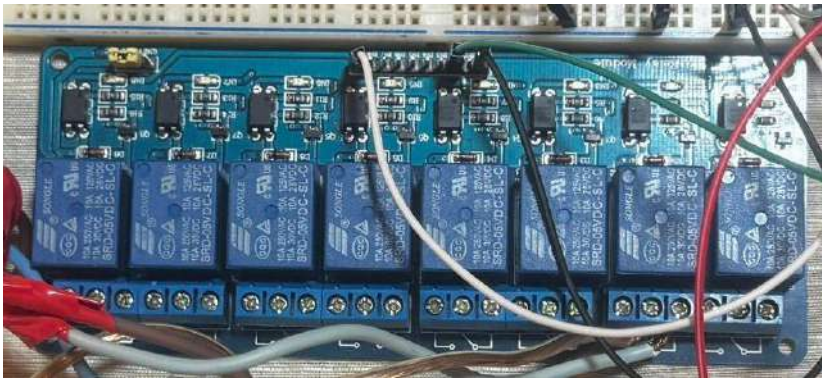

Ilustración 18. Dispositivos Relé

Por último, se presenta los módulos del Hardware con su interconexión (los 3 arduino plaquea y relé con tecla analógica y carga)

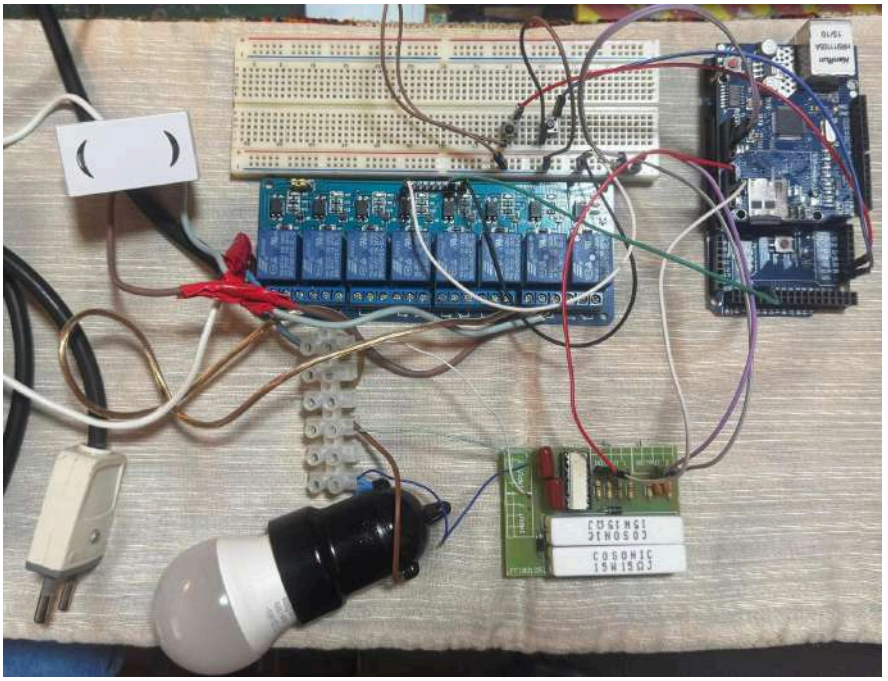

Ilustración 19. banco de prueba

3.2. SOFTWARE

Una interfaz debe ser simple y lógica para poder ser manipulada por cualquier usuario independientemente de la edad, los conocimientos, o las dificultades visuales. Además, debe poder ser implementada independientemente del sistema operativo ya sea

en un dispositivo con Android o iOS o Windows. Es por esto que se procedió a la programación de una WEB-APP, la cual opera independientemente del sistema operativo con el que se cuente.

Una aplicación Web es básicamente una página web. Se programa combinando html y java, e involucra dos partes:

- Interfaz: parte gráfica que se ve en el teléfono o tablet y se programa en HTML5.
- Servidor: es el que interpreta lo que hacemos en la interfaz gráfica y traduce para que el Arduino haga su función. Se programa/configura en lenguaje java.

Se programó una WebApp sencilla que se encuentra alojada en la tarjeta SD, ubicado en la unidad de control, y de esa manera, quien quiera acceder a la página para operar, debe dirigirse a esa IP (en este caso 192.168.0.100).

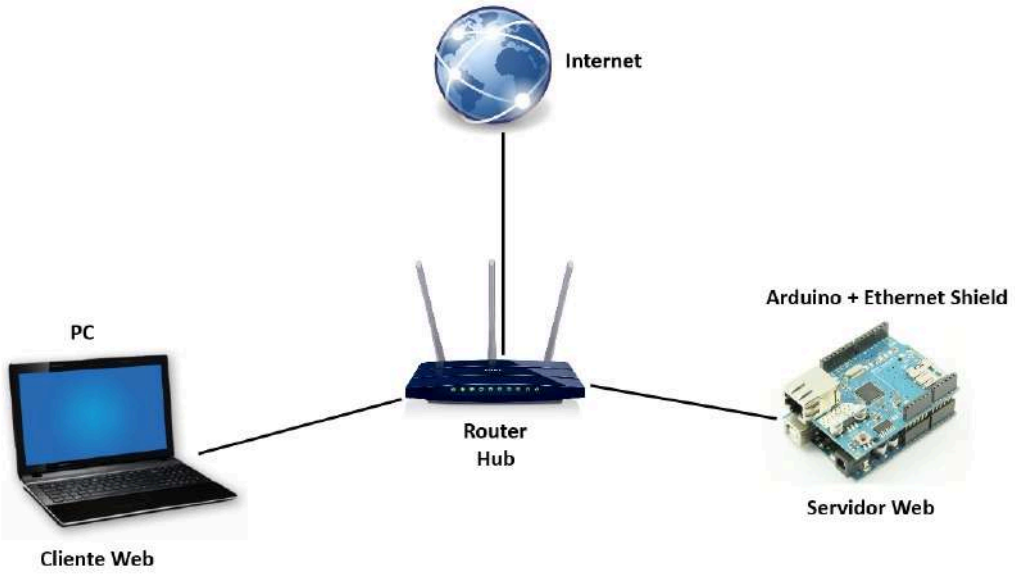

Ilustración 20: Comunicación de Web App

PULSADORES: en la vida diaria hogareña, loguearse para realizar una acción no sería lo más eficaz, sin contar que el dispositivo para acceder puede no estar prendido o encontrarse con poca batería.

Lo que se busca con los pulsadores es generar una llave digital programable, que contemple varias llaves analógicas a la vez o tareas rutinarias, sin la necesidad de logueo o el requerimiento físico de algún dispositivo móvil para realizar la misma.

El controlador analiza la señal del pulsador X y procede a generar los pulsos o estados necesarios para modificar algún relé o los que sean necesarios.

Un caso representativo de automatización orientada al confort y la eficiencia energética, se manifiesta en la posibilidad de apagar, mediante un único pulsador, todas las luminarias del interior del hogar al momento de disponerse al descanso. Este comando puede incorporar un retardo programable, y además activar el cierre automatizado de accesos, optimizando así tanto la comodidad del usuario como la seguridad del entorno.

Por razones ergonómicas, se sugiere ubicar este pulsador cerca de la mesa de luz. Del mismo modo, pueden configurarse pulsadores en sectores como el estar o living, para establecer escenas predeterminadas, que atenúe o apague las luces, conforme a preferencias específicas.

Esta lógica de control también puede aplicarse al encendido o apagado inmediato de toda la iluminación exterior, frente a situaciones que así lo demanden. Cabe destacar que, considerando la variabilidad en los hábitos de uso, todos los pulsadores han sido concebidos como elementos reprogramables, capaces de adaptarse dinámicamente a los patrones de comportamiento del usuario a lo largo del tiempo.

3.2.1. Programación Arduino

Configuración de Pines - Arduino Mega

Pins 0 y 1 - Comunicación serie (Serial)

Pin 2 - INPUT - Switch / entrada digital

Pin 3 - INPUT - Switch / entrada digital

Pin 4 - CS tarjeta SD

Pin 5 - INPUT - Switch / entrada digital

Pin 6 - OUTPUT - LED

Pin 7 - OUTPUT - LED

Pin 8 - OUTPUT - LED

Pin 9 - OUTPUT - LED

Pin 10 - OUTPUT - CS Ethernet / desactivado

Pin 20 - INPUT - Botón ON (interruptor)

Pin 21 - INPUT - Botón OFF (interruptor)

Pin 48 - OUTPUT - Relé / lámpara

Pin 49 - OUTPUT - Relé / lámpara

Pins 50-53 - SPI (Ethernet y SD)

Pin A0 - INPUT (analógico) - Sensor Línea 1

Pin A2 - INPUT (analógico) - Sensor Línea 2

Pin A3 - INPUT (analógico) - Sensor Carga 2

Pin A4 - INPUT (analógico) - Sensor Línea 3

Pin A5 - INPUT (analógico) - Sensor Carga 3

Pin A12 - INPUT (analógico) - Sensor Carga 1

Vin / 5V / GND - Alimentación del Arduino

3.2.2 Programación Web App

En este programa se van a identificar las variables de entrada, las variables de salida, definir acciones de los pulsadores externos y mostrar estados. En el anexo de programación (véase al final del proyecto) se verán en detalle las líneas de código implementadas.

3.2.3 Estructura del programa completo:

- Declaración de Librerías y Variables
- Definición de tamaño de buffers
- Configuración de dirección MAC e IP
- Iniciación de librería particular Ethernet con su respectivo puerto
- Habilitación de manejo de archivos de tarjeta SD
- Declaración Estados iniciales
- Asignación de pines

- Declaración de vectores de datos
- VOID SETUP
    - Iniciación tarjeta SD
    - Inicia conexión Ethernet y Servidor
    - Seteo de modos de pines como entradas/salidas
    - Lectura digital de pines
    - Seteo botones pulsadores prendido y apagado como entradas
    - Declaración de las interrupciones prender y apagar
- VOID LOOP
    - Escucha si hay cliente activo
    - Corre la subrutina “SetLEDs”
    - Corre la subrutina “Datos”
    - Refresca la Página Web
    - Pequeño Retraso
- VOID PRENDER
- VOID APAGAR
- VOID Datos
    - Lee las Entradas Analógicas y las guarda
    - Corre los “Gets” realizando promedios ponderados
    - Guarda estados de sensores
    - Muestra los estados de los sensores numéricamente
    - Muestra los estados de los sensores con Palabras Dedicadas
- VOID SETLEDS
    - Lee petición de usuario
    - Compara con estado actual
    - Si es necesario cambia el estado
    - guarda estado actual
- VOID XML_RESPONSE (Envía archivo con datos obtenidos)
    - valores guardados de lecturas de entradas analogicas
    - número de pines a ser switcheados
    - imprime en el cliente los estados actuales
- VOID STRCLEAR (limpia arreglos para el siguiente uso)
- Programación de los “Gets”
    - Realiza mediciones durando 0.025 segundos y devuelve el máximo valor

El refresco de la página se realiza de manera automática, cuando vence un determinado timer, o de manera forzada, cuando un usuario interactúa con la webapp o pulsadores.

3.3. COMUNICACIÓN

Existen muchos protocolos de comunicación que se adaptan al concepto de domótica. Pueden utilizarse sistemas cableados, inalámbricos y/o híbridos. Como se analizará oportunamente más adelante, este proyecto se basó en un sistema híbrido para la comunicación entre sus diferentes módulos

En la siguiente figura, se muestra la arquitectura del sistema, destacando el tipo de comunicación existente:

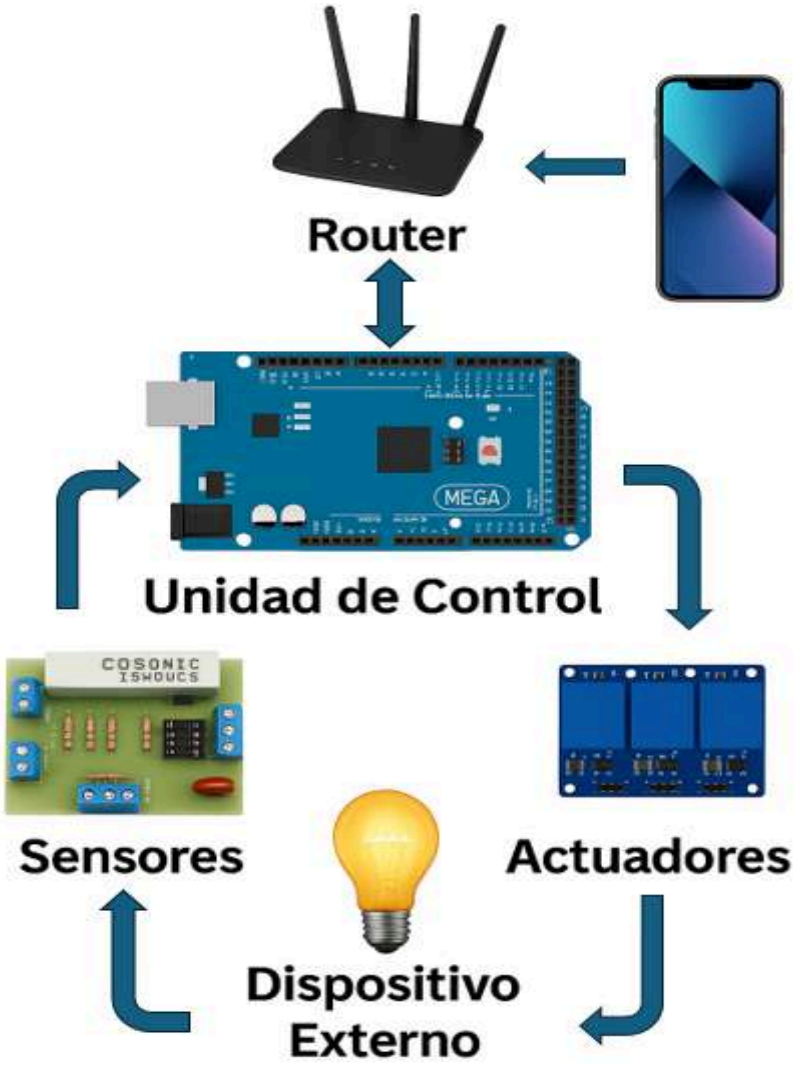

Ilustración 21. arquitectura del sistema

Tabla 6. Conexiones entre módulos

dispositivo 1	**dispositivo 2**	**tipo de comunicación**
Móvil	router	inalámbrica (internet wifi, 4g, datos móviles, etc...
Router	unidad de control	cableado (utp cat6)
unidad de control	actuadores	cableado (cable tel)
unidad de control	sensores	cableado (cable tel)

La comunicación entre el dispositivo móvil y el router ubicado en el inmueble a controlar, se realiza de manera inalámbrica vía wifi o los datos móviles. Entre el router y la unidad de control, el tendido del cable es de UTP CAT6 (ethernet). Finalmente, tanto de los sensores como para los actuadores, los cables que los interconectan con la unidad de control, es el cable telefónico.

Un problema que presenta este cableado con Cable UTP, es el alto nivel de sensibilidad con respecto a las interferencias existentes. Con el objetivo de atenuar el nivel de ruido presente en dicho cableado, se instaló una cañería paralela, una para el tendido eléctrico clásico y otra para el cableado del sistema domótico.

En el proceso de instalación de las cañerías, se tuvo en cuenta la coexistencia entre ambos sistemas, así también como la presencia de actuadores y la utilización de sensores.

En este proyecto, se utilizó un sistema HÍBRIDO.

3.3.1 Comunicación Interna

Con el objetivo de facilitar la interpretación, se subdividió la comunicación en interna y externa. Interna se refiere a la comunicación entre dispositivos dentro del domicilio, y externa a la comunicación hacia el exterior, en este caso el exterior sería el usuario, utilizando internet como medio de transporte de información.

3.3.1.1 Tarjeta SD - Arduino

En la tarjeta SD se encuentra alojado un programa llamado “index.htm”. Este programa es la página web a la cual se accede cuando se solicita la dirección IP 192.168.0.100 en el navegador del dispositivo que se esté utilizando (computadora de escritorio, portátil, tablets, teléfonos celulares, etc).

Existe una comunicación entre el programa que está guardado en la tarjeta de memoria (guardado con extensión “.htm”) y el microprocesador del arduino (capaz de realizar las operaciones matemáticas y/o lógicas que se le requieran).

3.3.1.2 Módulo Sensor - Arduino

El Módulo Sensor se comunica con Arduino mediante cables de cobre de par telefónico común, teniendo en uso dos de salidas (Línea y Carga), cable de masa y de tensión de referencia (en este caso provenientes de Arduino), y por último, dos cables más para los pulsadores de on-off respectivamente.

Por su parte la Placa Arduino, energiza el Módulo Sensor otorgándole una Referencia de 5V, y una descarga a una masa de 0V, y además recibe 4 cables de datos (línea, carga, pulsOn, PulsOFF).

3.3.1.3 Arduino - Router

El complemento Ethernet, se conecta directamente sobre la Placa Arduino que de fábrica viene preparada para recibir distintos shields. En este caso, el shield Ethernet posee una ficha para cable UTP Cat6. Utilizando dicho cable, Arduino se conecta al router, quien se encargará de la comunicación externa hacia Internet.

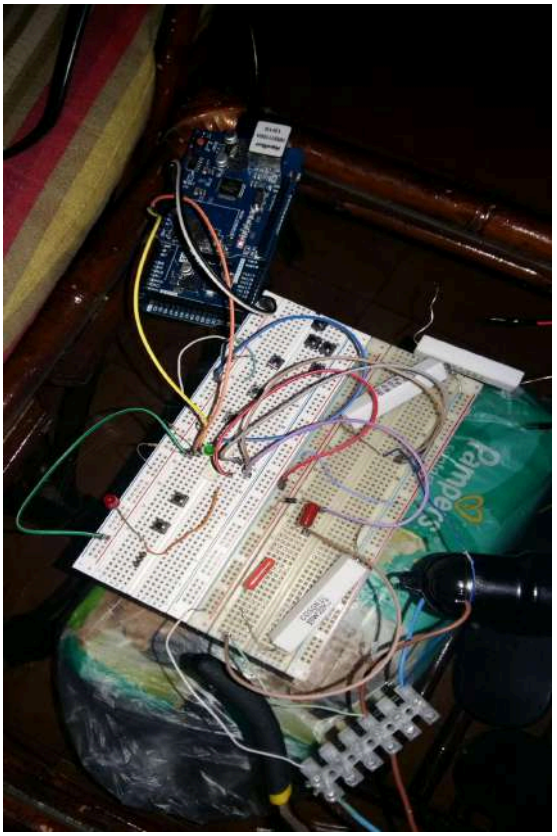

Ilustración 22. Sensor para 1 dispositivo en protoboard

3.3.2 Comunicación Externa

3.3.2.1 Router - Internet - Usuario

El Router se conecta a Internet mediante la red 3G/4G o vía WiFi con proveedores de internet comerciales. Esta comunicación es externa porque se requiere de internet, un agente externo a los dispositivos que se utilizan.

El usuario utiliza la comunicación visual, reconoce los botones de la página web (o los pulsadores) y acciona en consecuencia.

3.4 INTERFAZ USUARIO

La comunicación entre el sistema domótico y el usuario final se realiza a través de una interfaz web desarrollada en lenguaje HTML. Al ingresar la dirección IP correspondiente al módulo central desde cualquier navegador web, se accede a una página diseñada con una estética simple, armónica e intuitiva, que permite al usuario interactuar con el sistema sin ambigüedades ni distracciones.

En dicha interfaz se presentan claramente los identificadores de cada una de las luces controladas, junto con su estado actual. Para facilitar la interacción, se incluyen botones de encendido (**ON**) y apagado (**OFF**), codificados por colores verde y rojo respectivamente, lo que mejora la experiencia visual y evita errores de operación.

Ilustración 23. primera interfaz de usuario

El siguiente diagrama de flujo representa la lógica de funcionamiento de la interfaz, basada en la tabla de estados desarrollada previamente en la sección 3.1.2 ("Unidad de sensado o hardware de sensado"). En este diagrama, se definen los siguientes parámetros:

- **i**: índice de la luz a controlar
- **n**: número total de luces del sistema
- **IF**: estado del sensor de fase
- **IL**: estado del sensor de carga
- **Led_i**: estado actual de la luz i

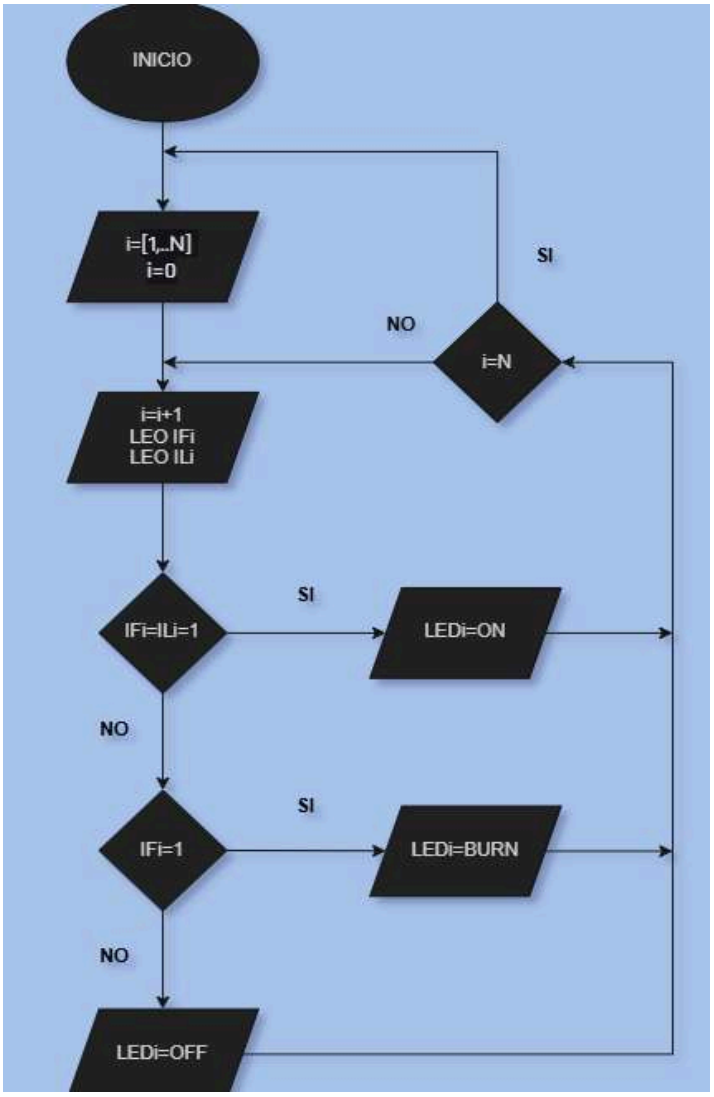

Ilustración 24. diagrama en bloque de botones en interfaz.

Ingresando la dirección IP desde cualquier dispositivo con acceso a la red local, el usuario puede visualizar en tiempo real tanto los estados lógicos internos del sistema como el estado físico de cada punto de luz (encendido, apagado o con lámpara quemada). Además, se dispone de una sección de control con botones individualizados que indican claramente el nombre del ambiente correspondiente y el estado actual del artefacto asociado.

Ilustración 25. interfaz con estados lógicos

Cabe destacar que, si bien en el ejemplo ilustrado se utiliza el nombre "Living" como etiqueta de control, la interfaz es completamente configurable, permitiendo editar los nombres y ampliar la cantidad de ambientes o artefactos según las necesidades del usuario o la escala del sistema.

Dado que los estados lógicos, que se muestran en la ilustración 23, no son relevantes para el usuario final, la interfaz se enfoca en brindar una experiencia funcional y directa. En las nuevas versiones de la interfaz, se ha **optado por un diseño más sobrio**, integrando directamente la información de estado **ON, OFF o BURN** dentro del mismo botón de control, eliminando textos adicionales y logrando una apariencia más limpia y moderna.

En la siguiente ilustración (24), se observa el botón correspondiente al **LED 4** en estado **encendido**, con su luz activada. En la imagen posterior ilustración 25, tras la interacción del usuario, el botón refleja el nuevo estado **apagado**, con la carga asociada desactivada, y el formato de botón actualizado.

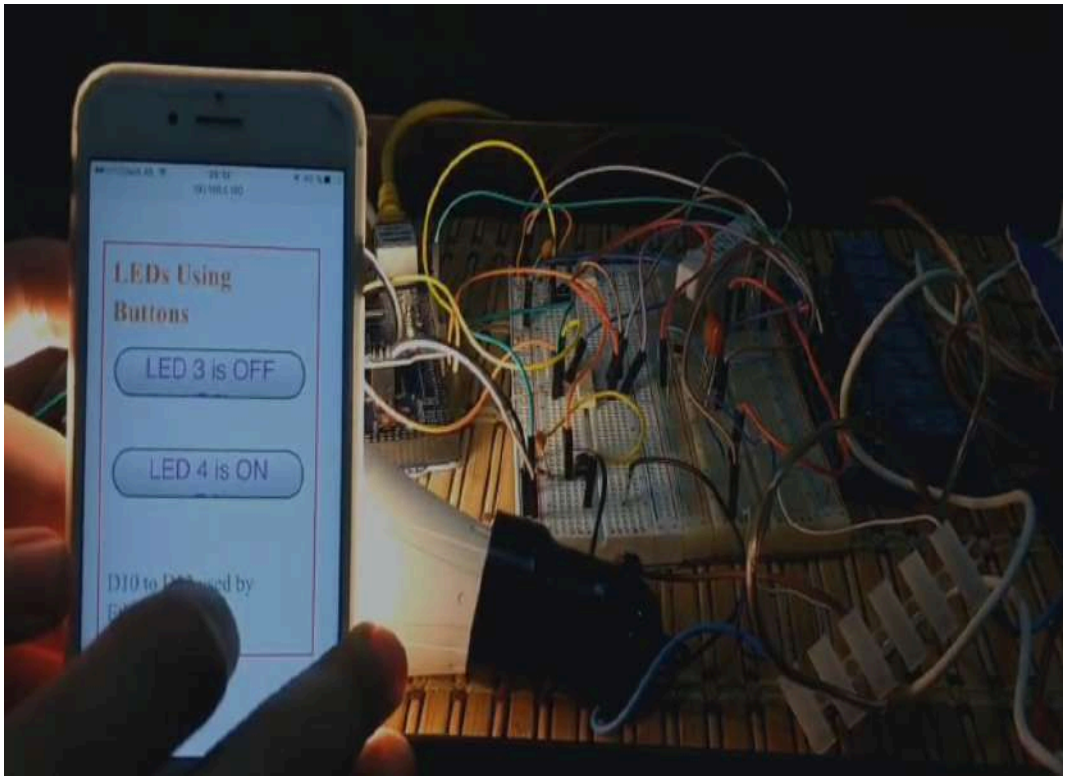

Ilustración 26. botón led 4 on

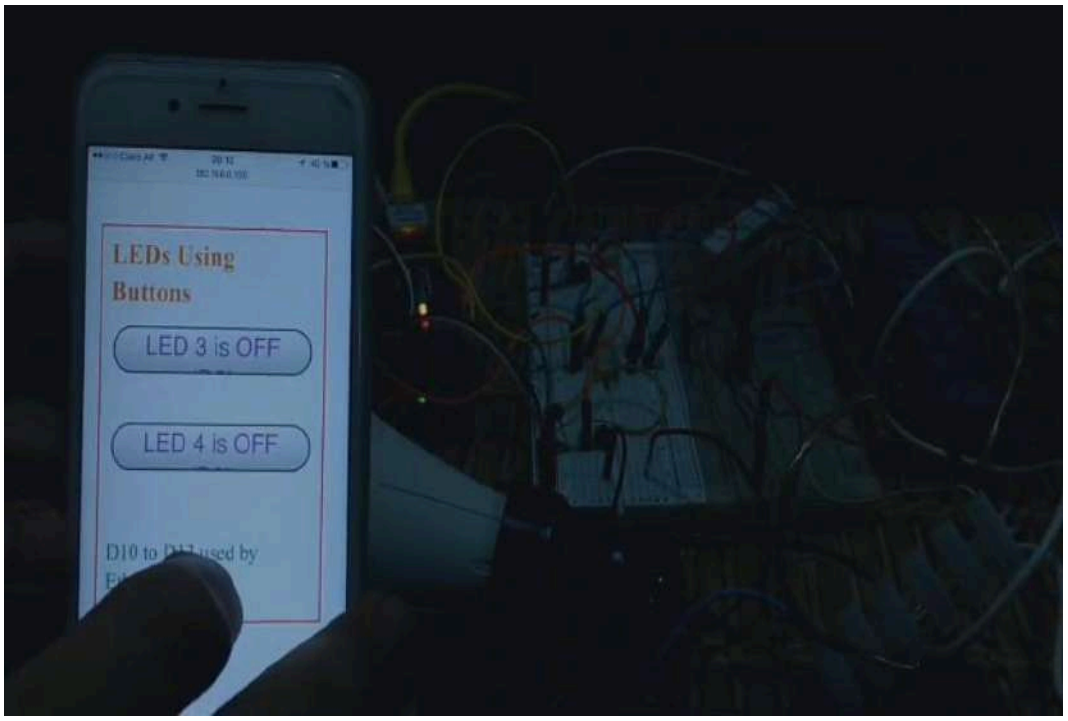

Ilustración 27. botón led 4 off

CAP 4. OBJETIVOS ALCANZADOS. OBSTÁCULOS Y MEJORAS A REALIZAR.

4.1 OBJETIVOS ALCANZADOS

El desarrollo del sistema domótico cumplió satisfactoriamente con los objetivos propuestos al inicio del proyecto. Se logró implementar un prototipo funcional, robusto, económico y adaptable a instalaciones eléctricas residenciales tradicionales.

Entre los principales logros del sistema, se destacan:

- Control local y remoto de dispositivos eléctricos, mediante una WebApp accesible desde cualquier navegador con conexión de red.
- Respuesta rápida y estable a comandos de usuario, con una arquitectura cableada que evita interferencias o pérdidas de conectividad.
- Diseño escalable, preparado para manejar hasta 15 dispositivos, y con posibilidades claras de ampliación modular.
- Coexistencia exitosa entre el sistema domótico y la instalación eléctrica convencional, sin generar conflictos eléctricos ni requerir modificaciones invasivas.
- Utilización de componentes fácilmente disponibles en el mercado.
  Se logró la implementación de pulsadores programables.,
- El sensado funciona perfectamente para luces led.
- La respuesta, tanto de hardware como de software, a los estímulos internos o externos, son de milisegundos, o segundos en muy pocos casos. Estamos hablando de un sistema que funciona casi en “tiempo real”, con muy poca latencia asociada.

4.2 OBSTÁCULOS

Durante el desarrollo del sistema se presentaron diversos desafíos técnicos que requirieron ajustes tanto en el diseño del hardware como en la lógica de programación.

Uno de los primeros inconvenientes fue el acondicionamiento adecuado de la señal de entrada proveniente de los sensores. La red eléctrica presentaba altos niveles de

ruido e inestabilidad, lo que dificultaba obtener lecturas confiables. Para mitigar este problema, se implementó un proceso de promediado por software en intervalos breves de tiempo, lo que permitió estabilizar considerablemente las mediciones y mejorar la precisión de los datos recogidos.

Otro obstáculo relevante fue el sobrecalentamiento inicial de la resistencia colocada a la entrada del circuito. Este componente generaba una pérdida innecesaria de energía en forma de calor. La solución fue reemplazar dicha resistencia por un capacitor, lo que no solo eliminó el problema térmico, sino que mantuvo el comportamiento eléctrico requerido sin comprometer la funcionalidad del sistema.

Por último, se detectó una limitación en cuanto a la infraestructura de red disponible. La falta de un proveedor de internet con señal estable en la zona prevista para la implementación del sistema representó un claro impedimento para su uso continuo y en tiempo real, especialmente considerando que la interfaz de control se basa en el acceso a través de red local o internet.

4.3 MEJORAS POSIBLES

- El sistema está pensado para luces LED y/o de bajo consumo. Las luces del tipo Filamento están casi extintas, por este motivo es que no tiene mucho sentido buscar una mejora en este caso ya que el mundo va camino a tener toda su luminaria con tecnología LED o superior.

- Mejorar el diseño utilizando la totalidad de los componentes electrónicos de características no dispersivas.

- Otro punto a tener en cuenta para un futuro diseño mejorado es que al cargar demasiado el sistema, la tensión varía notablemente, iluminando menos consecuentemente (solo con luces halógenas o de filamento).

- Una depuración del código completo podría ser mejorado sustancialmente por un colega informático que seguramente pueda optimizar algunas sentencias, y de esa manera utilizar menos recursos del sistema.

- La interfaz o diseño de la página web podría ser mejorada, utilizando mejor uso del espacio, botones con formas más agradables a la vista y hasta con una impronta moderna, pudiendo organizar fácilmente, por ejemplo, el orden de las luces a controlar o las más usadas.

- Reemplazar empalmes manuales y cintas por terminales con protección.

- Integrar una fuente de alimentación con aislamiento galvánico.

- Agregar protecciones contra sobrecorriente/sobretensión.

- Incluir un manual de usuario, de instalación eléctrica y de programación.

- Generar informes de funcionamiento en una tarjeta SD o enviarlos por red.

- Integración con asistentes de voz o sistemas de hogar inteligente como los son Alexa, Google Home o Home Assistant.

CAP 5. CONCLUSIONES

Se lograron cumplir los objetivos planteados al inicio del proyecto. La metodología adoptada desde un principio facilitó el desarrollo de las tareas y permitió superar los obstáculos de manera ordenada.

Uno de los logros más significativos, fue la posibilidad de aplicar conocimientos adquiridos durante la carrera en un desarrollo tangible, integrando además el aprendizaje de nuevos lenguajes de programación, hasta entonces desconocidos. Se trabajó con tres lenguajes distintos, lo cual implicó un desafío adicional que enriqueció notablemente el proceso formativo.

El diseño de la plaqueta de sensado atravesó varias mejoras hasta alcanzar una versión funcional que respondió adecuadamente a los requerimientos del sistema. No obstante, consideramos que existe un margen importante de mejora: la implementación en una placa de circuito impreso de doble capa, permitiría reducir el tamaño y mejorar la organización del diseño. Asimismo, la incorporación de un controlador con mayor capacidad permitiría ampliar la cantidad de entradas/salidas disponibles.

Queda como línea futura de trabajo la incorporación de nuevos dispositivos (motores, persianas, riego, audio, por nombrar algunos), los cuales podrían ser integrados mediante programación sin requerir necesariamente grandes modificaciones de hardware.

En cuanto al software, es el área que presenta mayor potencial de evolución. Se puede trabajar en la implementación de simuladores de presencia y en una aplicación de autoprogramación de pulsadores, cuyo objetivo es que el sistema se adapte dinámicamente al uso cotidiano del usuario.

Finalmente, reconocemos que la organización temporal fue un punto débil en la etapa de implementación, y afectó parcialmente el cumplimiento de los plazos establecidos. Esta experiencia nos deja como aprendizaje la importancia de la gestión de tiempos y la coordinación efectiva para proyectos de este calibre.

BIBLIOGRAFÍA

- Arduino. (2023). Arduino Mega 2560 Rev3. Recuperado de https://store.arduino.cc/

- Microchip. (2020). ENC28J60 Ethernet Controller Datasheet.

- IEEE. (2017). IEEE 2030.5-2018 – Smart Energy Profile Application Protocol.

- Mozilla Developer Network. (2024). Guía de desarrollo de aplicaciones web.

- National Instruments. (2019). Fundamentos del sensado analógico.

- W3C. (2022). Document Object Model (DOM) – Especificación.

- Forouzan, B. A. (2013). Comunicaciones de datos y redes. McGraw-Hill.

- Libros Arduino en español. (2021). Programación de microcontroladores – Arduino Mega.

ÍNDICE TABLAS

ÍNDICE ILUSTRACIONES

ANEXO DE PROGRAMACIÓN

```
//////////////////////DECLARACION DE LIBRERIAS Y
VARIABLES//////////////////////////////////////
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>

// Tamaño del buffer para capturar peticiones HTTP
#define REQ_BUF_SZ   60

 // Configuracion de direccion MAC e IP.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,0,100);
//byte gateway[] = { 200, 0, 183, 33 };
// the subnet:
//byte subnet[] = { 255, 255, 255, 224 };

 // Inicia la libreria Ethernet server con el puerto 80 (por defecto el puerto HTTP).
EthernetServer server(80);

File webFile;               // Es un tipo de clase que permite manejar archivos de una tarjeta SD
char HTTP_req[REQ_BUF_SZ] = {0}; // Buffer que guarda la petición HTTP con el tamaño de
REQ_BUF_SZ, terminado en null
char req_index = 0;             // Indice utilizado para el buffer HTTP_req
boolean LED_state[4] = {0};      // Arreglo con los estados de los leds


String tecla="OFF"; //Estado del Led inicialmente "OFF"
float voltajeLinea1;
int sensorLinea1;
int sensorLinea2;
int sensorLinea3;
float voltajeCarga;
int sensorCarga1;
int sensorCarga2;
int sensorCarga3;
float estado;
float SLmax1;
float SLmax2;
float SLmax3;
float get_sensorLineaMax1();//ponderado de sensor linea1
float get_sensorCargaMax1();//ponderado de sensor Carga1
float get_sensorLineaMax2();//ponderado de sensor linea2
float get_sensorCargaMax2();//ponderado de sensor Carga2
float get_sensorLineaMax3();//ponderado de sensor linea3
```

```
float get_sensorCargaMax3();//ponderado de sensor Carga3
float SensorLineaMax1;
float SensorLineaMax2;
float SensorLineaMax3;
float SensorCargaMax1;
float SensorCargaMax2;
float SensorCargaMax3;

String estado1;
String estado2;
String estado3;
//char estado[];

int lamp[]={48,49,10};// salidas a los rele
int lampa[]={46,47,48};
int Te[3];// suponemos estados de prueba
int T[2];//
int Ton[]={1,1,1};// vector para comparar on
int Toff[]={0,0,0};// vector para comparar off
int botonON = 20;// pulsadorON
int botonOFF = 21;// pulsadorOFF

////////////////////////////////////////////////////////////////////////////////////////
void setup()
{
   // Desactivo el chip Ethernet (no sabemos todavia why)
   pinMode(10, OUTPUT);
   digitalWrite(10, HIGH);


   Serial.begin(9600);

   // Inicialización de la tarjeta SD
   Serial.println("Inicialización de la tarjeta SD...");
   if (!SD.begin(4)) {
      Serial.println("ERROR - SD no se pudo iniciar !");
      return;
   }
   Serial.println("EXITO - tarjeta SD  inicializada.");
   // Chequeo de existencia del archivo index.htm
   if (!SD.exists("index.htm")) {
      Serial.println("ERROR - No se encuentra el archivo index.htm!");
      return;
   }
   Serial.println("EXITO - Archivo index.htm encontrado");


// Inicia la conexion Ethernet y el servidor.
```

```
Ethernet.begin(mac, ip);
server.begin();
Serial.print("IP local del servidor ");
Serial.println(Ethernet.localIP());



 //Inicializacion

// switches on pins 2, 3 and 5, es es del servidor:
   pinMode(2, INPUT);
   pinMode(3, INPUT);
   pinMode(5, INPUT);
   // LEDs
   pinMode(6, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(8, OUTPUT);
   pinMode(9, OUTPUT);

 //Definir los pines como salida y entrada. esto lo hicimos nosotros:

   {
    for (int i=0;i<3;i++)
     {  pinMode(lamp[i],OUTPUT);
       //digitalWrite(lamp[i],LOW);
       lampa[i]=digitalRead(lamp[i]);
       delay (200);
     }
    pinMode(botonON, INPUT);
    pinMode(botonOFF, INPUT);
   }
  attachInterrupt(digitalPinToInterrupt(botonON), prender, RISING);
  attachInterrupt(digitalPinToInterrupt(botonOFF), apagar, RISING);
}
void loop()
{
EthernetClient client = server.available();  // try to get client

   if (client) {  // Si hay cliente...
      boolean currentLineIsBlank = true;
      while (client.connected()) {
         if (client.available()) {   // client data available to read
            char c = client.read(); // read 1 byte (character) from client
            // limit the size of the stored received HTTP request
            // buffer first part of HTTP request in HTTP_req array (string)
            // leave last element in array as 0 to null terminate string (REQ_BUF_SZ - 1)
            if (req_index < (REQ_BUF_SZ - 1)) {
               HTTP_req[req_index] = c;         // save HTTP request character
```

```
      req_index++;
    }
    // last line of client request is blank and ends with \n
    // respond to client only after last line received
    if (c == '\n' && currentLineIsBlank) {
      // send a standard http response header
      client.println("HTTP/1.1 200 OK");
      // remainder of header follows below, depending on if
      // web page or XML page is requested
      // Ajax request - send XML file
      if (StrContains(HTTP_req, "ajax_inputs")) {
        // send rest of HTTP header
        client.println("Content-Type: text/xml");
        client.println("Connection: keep-alive");
        client.println();
        SetLEDs();
        Datos();
        // send XML file containing input states
        XML_response(client);
      }
      else {  // web page request
        // send rest of HTTP header
        client.println("Content-Type: text/html");
        client.println("Connection: keep-alive");
        client.println();
        // send web page
        webFile = SD.open("index.htm");       // open web page file
        if (webFile) {
          while(webFile.available()) {
            client.write(webFile.read()); // send web page to client
          }
          webFile.close();
        }
      }
      // display received HTTP request on serial port
      Serial.println(HTTP_req);
      Serial.println("");
      // reset buffer index and all buffer elements to 0
      req_index = 0;
      StrClear(HTTP_req, REQ_BUF_SZ);
      break;
    }
    // every line of text received from the client ends with \r\n
    if (c == '\n') {
      // last character on line of received text
      // starting new line with next character read
      currentLineIsBlank = true;
    }
```

```
            else if (c != '\r') {
               // a text character was received from client
               currentLineIsBlank = false;
            }
          } // end if (client.available())
       } // end while (client.connected())
       delay(1);      // give the web browser time to receive the data
       client.stop(); // close the connection
    } // end if (client)


delay(500);

}

 ////////////////////////PULSADORES/////////////////////////////////////////////////

void prender()
{for (int i=0;i<3;i++)
         {
          if (Te[i]!=Ton[i])
          {
           int rele=digitalRead(lamp[i]);
           digitalWrite(lamp[i],1-rele);
           Serial.print("Rele");
           Serial.print(i);
           Serial.print(" :");
           Serial.println(rele);
          }
          else
          {
//             //lamp[i]=0;
//             digitalWrite(lamp[i],LOW);
//             Serial.print("Lamp: ");
//             Serial.println(lamp[i]);
          }

  }
}
void apagar()
{
          for (int i=0;i<3;i++)
          {
           if (Te[i]!=Toff[i])
             {
           int rele=digitalRead(lamp[i]);
           digitalWrite(lamp[i],1-rele);
           Serial.print("Rele");
```

```
            Serial.print(i);
            Serial.print(" :");
            Serial.println(rele);
             }
            else
             {
//               //lamp[i]=0;
//               digitalWrite(lamp[i],LOW);
//               Serial.print("Lamp: ");
//               Serial.println(lamp[i]);
             }
            delay (200);
           }
      }

void Datos()
{
//  //  ////////////// LEO SALIDA DE OPTO TANTO LINEA COMO
CARGAS////////////////////////////////
  sensorLinea1=analogRead(A0);//salida del opto luz1 indica el estado de la LINEA1
  sensorCarga1=analogRead(A12);//salida del opto luz1 indica el estado de la CARGA1
  sensorLinea2=analogRead(A2);//salida del opto luz2 indica el estado de la LINEA2
  sensorCarga2=analogRead(A3);//salida del opto luz2 indica el estado de la CARGA2
  sensorLinea3=analogRead(A4);//salida del opto luz3 indica el estado de la LINEA3
  sensorCarga3=analogRead(A5);//salida del opto luz3 indica el estado de la CARGA3

//  /////OBTENGO VALORES MAXIMOS EN UN LAPSO DE TIEMPO PARA REALIZAR LOGICA DE
ESTADOS/////
  float SensorLineaMax1=get_sensorLineaMax1();//ponderado de sensor linea1
  float SensorCargaMax1=get_sensorCargaMax1();//ponderado de sensor Carga1
  float SensorLineaMax2=get_sensorLineaMax2();//ponderado de sensor linea2
  float SensorCargaMax2=get_sensorCargaMax2();//ponderado de sensor Carga2
  float SensorLineaMax3=get_sensorLineaMax3();//ponderado de sensor linea3
  float SensorCargaMax3=get_sensorCargaMax3();//ponderado de sensor Carga3


//  ////////////////////////////////////SENSADO
LISTO////////////////////////////////////////////////
//
/////////MUESTRA DE SENSADO - CONTROL/////
  Serial.print("SensorLinea1: ");
  Serial.print(SensorLineaMax1,0);
  Serial.print(" - SensorCarga1: ");
  Serial.println(SensorCargaMax1,0);
  Serial.print("Te[0]: ");
  Serial.print(Te[0]);
  Serial.print(" ;  T[0]: ");
  Serial.print(T[0]);
```

```
  Serial.print(" ; estado1: ");
  Serial.println(estado1);
//
//
//
//  ////////////////////////////////////LOGICA DE
CONTROL//////////////////////////////////////////
//
{
  if(SensorLineaMax1>500&SensorCargaMax1>500)
     {
      (estado1="ON  -")&&(Te[0]=1);
      T[0]=1;
     }
     else if(SensorLineaMax1>500)
     {
      (estado1="BURN-")&&(Te[0]=2);
      T[0]=0;
     }
         else
         {
           (estado1="OFF -")&&(Te[0]=0);
           T[0]=0;
         }
}
{
  if(SensorLineaMax2>500&SensorCargaMax2>500)
     {
      (estado2="ON  -")&&(Te[1]=1);
      T[1]=1;
     }
     else if(SensorLineaMax2>500)
     {
      (estado2="BURN-")&&(Te[1]=2);
      T[1]=0;
     }
         else
         {
           (estado2="OFF -")&&(Te[1]=0);
           T[1]=0;
         }
}
{
  if(SensorLineaMax3>500&SensorCargaMax3>500)
     {
      (estado3="ON  -")&&(Te[2]=1);
      T[2]=1;
     }
```

```
    else if(SensorLineaMax3>500)
    {
     (estado3="BURN-")&&(Te[2]=2);
     T[2]=0;
    }
        else
        {
         (estado3="OFF -")&&(Te[2]=0);
         T[2]=0;
        }
}
}

void SetLEDs(void)
{
   // LED 1 (pin 6)
   if (StrContains(HTTP_req, "LED1=1")) {
      LED_state[0] = 1;  // save LED state
      int rele=digitalRead(6);
      digitalWrite(6,1-rele);// save LED state//lo agregamos nosotros
//       digitalWrite(6, HIGH);
   }
   else if (StrContains(HTTP_req, "LED1=0")) {
      LED_state[0] = 0;  // save LED state
      int rele=digitalRead(6);
      digitalWrite(6,1-rele);// save LED state//lo agregamos nosotros
//       digitalWrite(6, LOW);
   }
   // LED 2 (pin 7)
   if (StrContains(HTTP_req, "LED2=1")) {
      LED_state[1] = 1;  // save LED state
      int rele=digitalRead(7);
      digitalWrite(7,1-rele);// save LED state//lo agregamos nosotros
//       digitalWrite(7, HIGH);
   }
   else if (StrContains(HTTP_req, "LED2=0")) {
      LED_state[1] = 0;  // save LED state
      int rele=digitalRead(7);
      digitalWrite(7,1-rele);// save LED state//lo agregamos nosotros
//       digitalWrite(7, LOW);
   }

   // LED 3 (pin 8)
   if (StrContains(HTTP_req, "LED3=1")) {
      LED_state[2] = 1;  // save LED state
      int rele=digitalRead(8);
      digitalWrite(8,1-rele);// save LED state//lo agregamos nosotros
//       digitalWrite(8, HIGH);
```

```
   }
   else if (StrContains(HTTP_req, "LED3=0")) {
      LED_state[2] = 0;  // save LED state
      int rele=digitalRead(8);
      digitalWrite(8,1-rele);// save LED state//lo agregamos nosotros
}

   // LED 4 (pin 9)
   if (StrContains(HTTP_req, "LED4=1")) {
      LED_state[3] = 1;  // save LED state
      int rele=digitalRead(9);
      digitalWrite(9,1-rele);// save LED state//lo agregamos nosotros
//       digitalWrite(9, HIGH);
   }
   else if (StrContains(HTTP_req, "LED4=0")) {
      LED_state[3] = 0;  // save LED state
      int rele=digitalRead(9);
      digitalWrite(9,1-rele);// save LED state//lo agregamos nosotros

//       else if (StrContains(HTTP_req, "LED4=2")) {//lo agregamos nosotros
//             LED_state[3] = 0;  // save LED state
//             int rele=digitalRead(9);
//             digitalWrite(9,1-rele);// save LED state//lo agregamos nosotros
   }
}

// send the XML file with analog values, switch status
//  and LED status
void XML_response(EthernetClient cl)
{
   int analog_val;           // stores value read from analog inputs
   int count;                // used by 'for' loops
   int sw_arr[] = {2, 3, 5};  // pins interfaced to switches

   cl.print("<?xml version = \"1.0\" ?>");
   cl.print("<inputs>");
   // read analog inputs - realmente muestra tabla de estados
   for (count = 0; count < 3; count++) { // tabla de estados 0.1.2
      analog_val = Te[count];//lo cambiamos nosotros
      cl.print("<analog>");
      cl.print(Te[count]);// lo cambiamos nosotros OJO
      cl.println("</analog>");
   }
   // read switches
  // for (count = 0; count < 3; count++) {
      cl.print("<switch>");
      cl.print(estado1);
      cl.println("</switch>");
```

```
   cl.print("<switch>");
   cl.print(estado2); // indexar estados asi mandamos a html bien!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   cl.println("</switch>");
   cl.print("<switch>");
   cl.print(estado3);
   cl.println("</switch>");
 //   }
// checkbox LED states
// LED1
cl.print("<LED>");
if (LED_state[0]) {
   cl.print("checked");
}
else {
   cl.print("unchecked");
}
cl.println("</LED>");

// LED2
cl.print("<LED>");
if (T[0]) {
   cl.print("on");
}
else {
   cl.print("off");
}
cl.println("</LED>");

// LED3
cl.print("<LED>");
if (T[1]) {
   cl.print("on");
}
else {
   cl.print("off");
}
cl.println("</LED>");
// LED4
cl.print("<LED>");
if (T[2]) {
   cl.print("on");
}
else {
   cl.print("off");
}
cl.println("</LED>");

cl.print("</inputs>");
```

```
}

// sets every element of str to 0 (clears array)
void StrClear(char *str, char length)
{
   for (int i = 0; i < length; i++) {
      str[i] = 0;
   }
}

// searches for the string sfind in the string str
// returns 1 if string found
// returns 0 if string not found
char StrContains(char *str, char *sfind)
{
   char found = 0;
   char index = 0;
   char len;

   len = strlen(str);

   if (strlen(sfind) > len) {
      return 0;
   }
   while (index < len) {
      if (str[index] == sfind[found]) {
         found++;
         if (strlen(sfind) == found) {
            return 1;
         }
      }
      else {
         found = 0;
      }
      index++;
   }

   return 0;
}


//////////////////////////PROGRAMACION DE LOS GET////////////////////////////////
float get_sensorLineaMax1()
{
  float voltajeLinea1;
  long tiempo=millis();
  float SLmax1=100;
  while(millis()-tiempo<25)//realizamos mediciones durante 0.025 segundos
```

```
   {
     voltajeLinea1 = analogRead(A0);//lectura del sensor
     if(voltajeLinea1>SLmax1)SLmax1=voltajeLinea1;
   }
  return(SLmax1);
}
float get_sensorLineaMax2()
{
  float voltajeLinea2;
  long tiempo=millis();
  float SLmax2=100;
  while(millis()-tiempo<25)//realizamos mediciones durante 0.025 segundos
  {
    voltajeLinea2 = analogRead(A2);//lectura del sensor
    if(voltajeLinea2>SLmax2)SLmax2=voltajeLinea2;
  }
  return(SLmax2);
}
float get_sensorLineaMax3()
{
  float voltajeLinea3;
  long tiempo=millis();
  float SLmax3=100;
  while(millis()-tiempo<25)//realizamos mediciones durante 0.025 segundos
  {
    voltajeLinea3 = analogRead(A4);//lectura del sensor
    if(voltajeLinea3>SLmax3)SLmax3=voltajeLinea3;
  }
  return(SLmax3);
}
float get_sensorCargaMax1()
{
  float voltajeCarga1;
  long tiempo=millis();
  float SCmax1=123;
  while(millis()-tiempo<25)//realizamos mediciones durante 0.025 segundos
  {
    voltajeCarga1 = analogRead(A12);//lectura del sensor
    if(voltajeCarga1>SCmax1)SCmax1=voltajeCarga1;
  }
  return(SCmax1);
}
 float get_sensorCargaMax2()
{
  float voltajeCarga2;
  long tiempo=millis();
  float SCmax2=100;
  while(millis()-tiempo<25)//realizamos mediciones durante 0.025 segundos
```

```
  {
    voltajeCarga2 = analogRead(A3);//lectura del sensor
    if(voltajeCarga2>SCmax2)SCmax2=voltajeCarga2;
  }
  return(SCmax2);
}
  float get_sensorCargaMax3()
{
  float voltajeCarga3;
  long tiempo=millis();
  float SCmax3=100;
  while(millis()-tiempo<25)//realizamos mediciones durante 0.025 segundos
  {
    voltajeCarga3 = analogRead(A5);//lectura del sensor
    if(voltajeCarga3>SCmax3)SCmax3=voltajeCarga3;
  }
  return(SCmax3);
}
```

ANEXO DE PROGRAMACIÓN DEPURADO Y CON COMENTARIOS INTELIGENCIA ARTIFICIAL

Resumen de mejoras a implementar

- Modernización de la interfaz web:
- Reemplazo del sistema AJAX + HTML estático en SD por una interfaz más moderna con Bootstrap y Fetch API (sin recargar la página).
- Posibilidad de mostrar estado en tiempo real con visualización más clara (ej: colores, íconos, etc.).
- Refactor del código:
- Consolidación de las funciones get_sensorLineaMaxN() y get_sensorCargaMaxN() en una sola función genérica con parámetros.
- Mejora en estructura y separación por módulos: web_server.ino, sensors.ino, control.ino, etc.
- Seguridad y robustez:
- Validación de entradas HTTP para evitar errores o accesos no deseados.
- Control de rebote por software para los pulsadores.
- Eliminación de código redundante o muerto.
- Estilo y comentarios claros:
- Comentarios consistentes en toda la programación.
- Mejora de nombres de variables y funciones para que sean más expresivas.

```
/**************************************************

 * Sistema Domótico - Control de luces y motores vía Web
 * Versión Mejorada con:
 * - Modularización de sensores
 * - Optimización de lógica de control
 * - Uso de librería EthernetWebServer
 * - Modernización del manejo AJAX
```

```
 **************************************************/
#include <Ethernet.h>
#include <EthernetWebServer.h>
#include <SPI.h>
#include <SD.h>

// Configuración de red
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,0,100);
EthernetWebServer server(80);

// Pines
const int relePins[] = {48, 49, 10};
const int botonON = 20;
const int botonOFF = 21;

// Estados
bool releEstado[3] = {false, false, false};
String estadoTexto[3] = {"OFF", "OFF", "OFF"};

// Pines sensores
const int sensorLinea[] = {A0, A2, A4};
const int sensorCarga[] = {A12, A3, A5};

// Inicialización
void setup() {
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();

  for (int i = 0; i < 3; i++) {
    pinMode(relePins[i], OUTPUT);
    digitalWrite(relePins[i], LOW);
  }
  pinMode(botonON, INPUT);
  pinMode(botonOFF, INPUT);
  attachInterrupt(digitalPinToInterrupt(botonON), prender, RISING);
  attachInterrupt(digitalPinToInterrupt(botonOFF), apagar, RISING);

  server.on("/", handleRoot);
  server.on("/estado", HTTP_GET, handleEstado);
}

// Loop principal
void loop() {
  server.handleClient();
}
```

```
// Página principal
void handleRoot() {
  File webFile = SD.open("index.htm");
  if (webFile) {
    server.streamFile(webFile, "text/html");
    webFile.close();
  } else {
    server.send(404, "text/plain", "Archivo no encontrado");
  }
}

// Endpoint para enviar estado por AJAX
void handleEstado() {
  String response = "<estado>";
  for (int i = 0; i < 3; i++) {
    float valL = medirSensor(sensorLinea[i]);
    float valC = medirSensor(sensorCarga[i]);
    if (valL > 500 && valC > 500) {
      estadoTexto[i] = "ON";
      releEstado[i] = true;
    } else if (valL > 500) {
      estadoTexto[i] = "BURN";
      releEstado[i] = false;
    } else {
      estadoTexto[i] = "OFF";
      releEstado[i] = false;
    }
    response += "<r" + String(i+1) + ">" + estadoTexto[i] + "</r" + String(i+1) + ">";
  }
  response += "</estado>";
  server.send(200, "text/xml", response);
}

// Función para sensado de voltaje (máximo en 25 ms)
float medirSensor(int pin) {
  float maxVal = 100;
  long t0 = millis();
  while (millis() - t0 < 25) {
    float lectura = analogRead(pin);
    if (lectura > maxVal) maxVal = lectura;
  }
  return maxVal;
}

// Encender
void prender() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(relePins[i], HIGH);
```

```
  }
}

// Apagar
void apagar() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(relePins[i], LOW);
  }
}
```

Principales mejoras:

- Reemplazo de funciones get_sensor... repetidas por una única función medirSensor().
- Reemplazo de EthernetServer por EthernetWebServer (más limpio para manejar endpoints).
- Optimización del manejo de estado (estadoTexto[]).
- Preparado para AJAX real vía /estado con XML.
- Código modular y más legible.

Autorización Repositorio Institucional -RINFI

Repositorio Institucional RINFI, Facultad de Ingeniería, UNMDP

En calidad de TITULARES de los derechos de autor de la obra que se detalla a continuación, y sin infringir según mi conocimiento derechos de terceros, por la presente informo a la Facultad de Ingeniería de la UNMDP mi decisión de concederle en forma gratuita, no exclusiva y por tiempo ilimitado la autorización para:

1) Publicar el texto del trabajo más abajo indicado, exclusivamente en medio digital, en el sitio web de la Facultad y/o Universidad, por Internet, a título de divulgación gratuita de la producción científica generada por la Facultad, a partir de la fecha especificada.

2) Permitir a la Biblioteca que, sin producir cambios en el contenido, establezca los formatos de publicación en la web para su más adecuada visualización y la realización de copias digitales y migraciones de formato necesarias para la seguridad, resguardo y preservación a largo plazo de la presente obra:

Autor 1: Nahuel Esteban Distéfano Documento: 31.734.144 Teléfono: 2234-546547 E-mail: distefanonahuel@gmail.com	 Firma 1
Autor 2: Facundo Fernández Minich Documento: 32.792.277 Teléfono: 2235-296995 E-mail: facufernandezminich@gmail.com	 Firma 2
Director/a: Alejandro José Uriz Documento: 31.018.442 Leg. 14803	 Firma Director/a
Codirector/a: Juan Alberto Etcheverry Documento: 28.545.962 Leg. 18484	 Firma Codirector/a

2. Título obtenido: Ingeniero Electrónico ……

3. Identificación/Título de la Obra: SISTEMA DOMÓTICO ……

4.  AUTORIZO la publicación bajo con la licencia Creative Commons BY-NC-ND Atribución-NoComercial-Sin Obra Derivada.

5. **Nota de Embargo:** Para aquellas obras que NO pueden ser de acceso a texto completo por razones de acuerdos previos con empresas o instituciones; por razones de índole comercial u otras razones; se procederá según lo establecido en Art. 6 de la Ley 26899 de Repositorios digitales institucionales de acceso abierto:

> **ARTICULO 6° —** En caso que las producciones científico-tecnológicas y los datos primarios estuvieran protegidos por derechos de propiedad industrial y/o acuerdos previos con terceros, los autores deberán proporcionar y autorizar el acceso público a los metadatos de dichas obras intelectuales y/o datos primarios, comprometiéndose a proporcionar acceso a los documentos y datos primarios completos a partir del vencimiento del plazo de protección de los derechos de propiedad industrial o de la extinción de los acuerdos previos antes referidos.
>
> Asimismo, podrá excluirse la difusión de aquellos datos primarios o resultados preliminares y/o definitivos de una investigación no publicada ni patentada que deban mantenerse en confidencialidad, requiriéndose a tal fin la debida justificación institucional de los motivos que impidan su difusión. Será potestad de la institución responsable en acuerdo con el investigador o equipo de investigación, establecer la pertinencia del momento en que dicha información deberá darse a conocer. A los efectos de la presente ley se entenderá como "metadato" a toda aquella información descriptiva sobre el contexto, calidad, condición o características de un recurso, dato u objeto, que tiene la finalidad de facilitar su búsqueda, recuperación, autentificación, evaluación, preservación y/o interoperabilidad.

En razón de lo expuesto, si el Trabajo se encuentra comprendido en el caso de que su producción esté protegida por derechos de Propiedad Industrial y/o acuerdos previos con terceros que implique la confidencialidad de los mismos, el/la directora/a debe indicar a continuación motivos y fecha de finalización del embargo:

NO SE AUTORIZA la publicación antes de la fecha __/__/_____ por lo siguientes motivos:

……

……

……

Cumplido el plazo del embargo, estará accesible a texto completo según contempla la normativa vigente.

_____

Director/a del TF