



FACULTAD DE INGENIERIA

Universidad de Buenos Aires

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Medición y aceptación de parámetros en transformadores de corriente alterna

Autor:
Ing. Cristian Trinidad

Director:
Esp. Lic. Leopoldo A. Zimperz (Iris Tecnología S.R.L.)

Jurados:
Mg. Ing. Gonzalo Sánchez (FAA)
Mg. Ing. Gerardo Puga (UNLP)
Esp. Ing. Agustín Rey (FIUBA)

*Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires,
entre junio de 2020 y junio de 2021.*

Resumen

La presente memoria describe el desarrollo de un prototipo encargado de medir, registrar y calificar los parámetros de transformadores de tensión alterna empleados en la fabricación de dispositivos electromédicos. Con este trabajo se busca acelerar el proceso de calificación de transformadores que actualmente se realiza manualmente por un operador. Este trabajo fue realizado para la empresa Iris Tecnología S.R.L.

Para llevar a cabo el trabajo se utilizaron algunos de los conceptos aprendidos a lo largo de la carrera como el manejo de tareas, colas y semáforos en sistemas operativos de tiempo real, acceso a un servidor web por medio de protocolos de internet como HTTP y conexiones inalámbricas, desarrollo de pruebas de aceptación y criterios generales de desarrollo circuitos impresos.

Índice general

Resumen	I
1. Introducción general	1
1.1. Introducción a los transformadores	1
1.1.1. Caracterización de transformadores	2
Ensayo de vacío	3
Ensayo de cortocircuito	3
Ensayo de aislamiento	4
1.2. Estado del arte	4
1.3. Motivación	5
1.4. Objetivos y alcance	5
1.4.1. Objetivos del trabajo	5
1.4.2. Alcance del trabajo	5
2. Introducción específica	7
2.1. Estructura general del sistema	7
2.2. Requerimientos	10
2.2.1. Requerimientos de hardware	10
2.2.2. Requerimientos relativos a los valores a medir	11
2.2.3. Requerimientos funcionales	11
2.2.4. Requerimientos de comunicación	11
2.2.5. Requerimientos de interfaz de usuario	12
2.3. Kit ESP32-DevKitC	12
2.3.1. Entorno de desarrollo ESP-IDF	14
2.3.2. ESP-Touch y SmartConfig	15
2.4. Sensor de tensión	16
2.5. Sensor de corriente	17
2.6. Impresora	18
2.6.1. Protocolo DPL	18
Comandos inmediatos	19
Comandos de sistema	19
2.6.2. Módulo adaptador RS232	20
2.7. Display alfanumérico	21
2.8. Módulos misceláneos	22
Módulo de relés	22
Módulo de alimentación	23
3. Diseño e implementación	25
3.1. Descripción de hardware	25
3.1.1. Mejoras a las entradas analógicas	26
3.1.2. Integración de hardware	28
3.2. Descripción de firmware	30
3.2.1. Arquitectura de firmware	31

3.2.2.	Estructura general del firmware	31
3.2.3.	Controlador de sistema	33
Etapa de inicialización	35	
Etapa de configuración	36	
Etapa de caracterización	37	
Etapa de reporte	40	
3.2.4.	Medición de valor eficaz	41
3.2.5.	Comunicación Wi-Fi	43
3.2.6.	Obtención y envío de datos al webserver	45
3.2.7.	Implementación del protocolo DPL	48
3.2.8.	Interfaz de usuario	50
4.	Ensayos y resultados	51
4.1.	Pruebas funcionales del hardware	51
5.	Conclusiones	53
5.1.	Conclusiones generales	53
5.2.	Próximos pasos	53
Bibliografía		55

Índice de figuras

1.1. Transformador de baja potencia ¹	1
1.2. Transformador monofásico ideal ²	2
1.3. Ensayo de vacío	3
1.4. Ensayo de cortocircuito	3
1.5. Ensayo de aislamiento entre bobinas	4
2.1. Diagrama en bloques del sistema	8
2.2. Diagrama de secuencia simplificado	9
2.3. ESP32-DevKitC V1	13
2.4. ESP32-WROOM-32	13
2.5. Aplicación Android ESP-Touch	15
2.6. Módulo sensor de tensión alterna	16
2.7. Ondas de entrada (Vin) y salida (Vout) del módulo @ Vcc = 5 V	16
2.8. Módulo sensor de corriente alterna	17
2.9. Impresora E-Class™ Mark III modelo E-4204B	18
2.10. Etiqueta resultante del comando de la columna Ejemplo de la tabla 2.3	20
2.11. Módulo adaptador RS232 a TTL	20
2.12. <i>Display</i> alfanumérico de 20x4	21
2.13. Módulo de relés	22
2.14. Módulo de alimentación	23
3.1. Conexionado kit ESP32-DevKitC y módulos de hardware	25
3.2. Saturación módulos sensores	27
3.3. Saturación mejorada	27
3.4. Trabajo final terminado	28
3.5. Trabajo terminado con la tapa de seguridad abierta	29
3.6. Contenido gabinete principal	30
3.7. Gabinete auxiliar sin el transformador a ensayar	30
3.8. Patrón observar y reaccionar	31
3.9. Patrón observar y reaccionar aplicado al trabajo	31
3.10. Diagrama simplificado del controlador del sistema	34
3.11. Etapa de inicialización	35
3.12. Etapa de configuración	37
3.13. Etapa de caracterización	38
3.14. Etapa de reporte	40
3.15. Secuencia de conexión a red Wi-Fi	44
3.16. Diagrama de flujo función de la <i>print</i>	49

Índice de tablas

1.1. Estado del arte	4
2.1. Comandos protocolo DPL	19
2.2. Comandos inmediatos protocolo DPL	19
2.3. Comandos de sistema protocolo DPL	20
2.4. Señales de control y datos <i>display</i>	21
2.5. Instrucciones controlador HD44780	22
3.1. Uso de pines del kit ESP32-DevKitC	26

Capítulo 1

Introducción general

En este capítulo se presenta una introducción a los transformadores y sus ensayos de caracterización más comunes. Asimismo, se introducen algunos equipos disponibles en el mercado y por último, se aborda el alcance y objetivo del trabajo realizado.

1.1. Introducción a los transformadores

Un transformador eléctrico es una máquina estática de corriente alterna que permite aumentar o disminuir la tensión y corriente de un circuito. La frecuencia de la onda de entrada se mantiene invariable a la salida y, en el caso de un transformador ideal, la potencia también se mantiene constante. El transformador utiliza el principio de la inducción electromagnética para su funcionamiento. En la figura 1.1 se muestra la imagen de un transformador típico de baja tensión [1], [2].

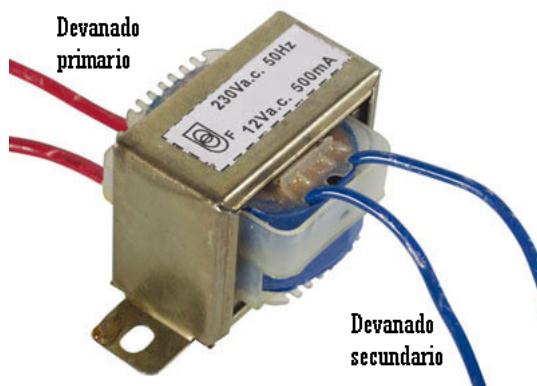


FIGURA 1.1. Transformador de baja potencia¹.

El transformador está constituido por dos bobinas de material conductor devanadas sobre un núcleo cerrado de material ferromagnético. Los bobinados se denominan primario y secundario según corresponda a la entrada o salida del sistema, respectivamente. Las bobinas o bobinados están aislados eléctricamente entre sí, la única conexión entre estos la constituye el flujo magnético que se establece en el núcleo por la circulación de corriente por dichos bobinados. Por su parte, el núcleo generalmente se fabrica de láminas apiladas de hierro o acero, de esta forma, se optimiza el flujo magnético generado por las corrientes circulantes en

¹Imagen tomada de <https://www.ingmecafenix.com/electronica/transformador-electrico/>

los bobinados. En la figura 1.2 se muestra el principio de funcionamiento de un transformador.

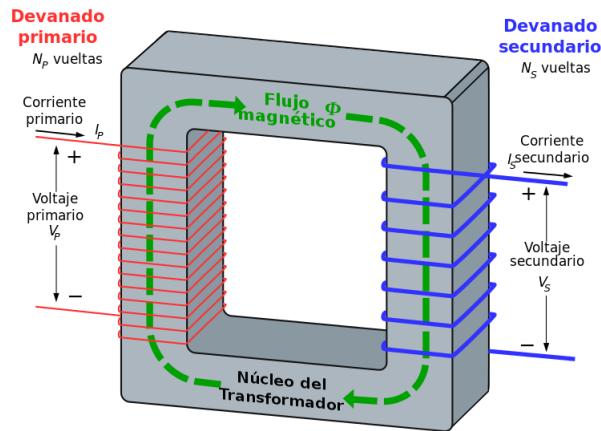


FIGURA 1.2. Transformador monofásico ideal².

Existen transformadores con más de dos bobinados y diferentes arquitecturas de conexión, un ejemplo lo constituye el transformador trifásico que está formado por 6 bobinados que pueden ser conectados en estrella o triángulo.

Actualmente, se pueden encontrar una gran variedad de transformadores para infinidad de aplicaciones. Una clasificación posible es la siguiente [3]:

- Transformador de potencia.
- Transformador de distribución.
- Autotransformador.
- Transformador de corriente.
- Transformador de potencial.

Las aplicaciones van desde cargadores de teléfonos móviles hasta transformadores de potencia para estaciones y subestaciones de energía eléctrica.

1.1.1. Caracterización de transformadores

Para los cálculos de circuitos o líneas con transformadores se utiliza un circuito equivalente que representa el comportamiento del transformador real. Para la mayoría de los casos es suficiente con que dicho circuito equivalente represente el transformador en régimen permanente [1].

En general, se deben desarrollar diferentes ensayos sobre el transformador para determinar los parámetros de un circuito equivalente, entre los ensayos más comunes tenemos:

- Ensayo de vacío.
- Ensayo de cortocircuito.

²Imagen tomada de <https://es.wikipedia.org/wiki/Transformador>

- Ensayo de aislamiento.

En las siguientes secciones se brinda una introducción a los ensayos mencionados.

Ensayo de vacío

El ensayo de vacío es un método utilizado para determinar diversos parámetros del transformador mediante pruebas realizadas sin aplicar carga [4]. En la figura 1.3 se muestra el conexionado para este ensayo.

En este ensayo se alimenta el bobinado primario con la tensión nominal, se deja el secundario sin carga y se debe medir la tensión en los bornes del primario, la tensión en los bornes del secundario, la corriente en el devanado primario y la potencia del primario.

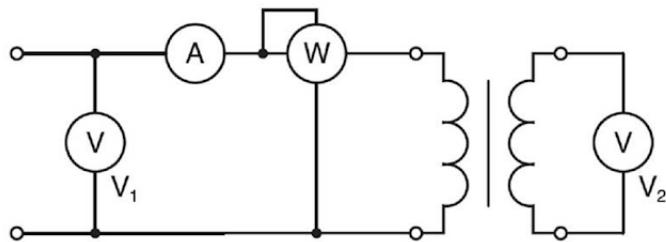


FIGURA 1.3. Ensayo de vacío.

Ensayo de cortocircuito

El ensayo de cortocircuito permite determinar la impedancia de cortocircuito o impedancia en serie del transformador [5]. La impedancia de cortocircuito representa las pérdidas en el cobre de los devanados, así como la inductancia de dispersión y otras inductancias parásitas. En la figura 1.4 se muestra el conexionado para este ensayo.

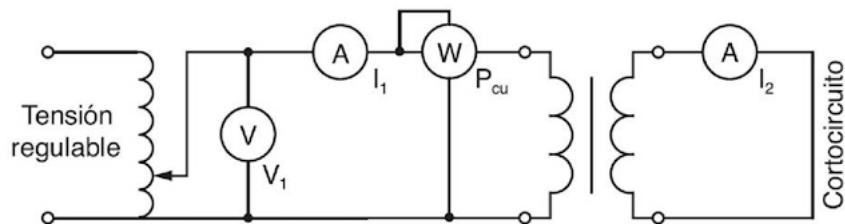


FIGURA 1.4. Ensayo de cortocircuito.

En este ensayo se alimenta el bobinado primario con una tensión muy reducida, se coloca un cortocircuito en el bobinado secundario y se debe medir la tensión resultante en los bornes del primario, la corriente en el bobinado secundario, la corriente en el devanado primario y la potencia del primario. Para conseguir los valores reducidos de tensión es necesario un sistema de tensión ajustable como puede ser un autotransformador regulable. La tensión aplicada en el primario debe ser tal que haga circular la corriente nominal por el bobinado secundario.

Ensayo de aislamiento

El ensayo de aislamiento permite determinar el estado del dieléctrico o aislante entre fases o entre una fase y el chasis del transformador [6]. La medida suele dar valores en el orden de los megaohmios, valor que se ve reducido si el aislante está deteriorado.

En la figura 1.5 se muestra un ensayo de aislamiento entre bobinas.

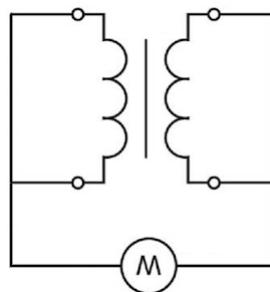


FIGURA 1.5. Ensayo de aislamiento entre bobinas.

1.2. Estado del arte

En la actualidad existe una variedad de equipos con el fin de caracterizar transformadores de baja tensión. Estos son capaces de caracterizar transformadores de tensión y corriente, sobre los cuales se pueden realizar múltiples ensayos y así determinar diferentes parámetros. A su vez, cuentan con interfaces de comunicación variadas como Wi-Fi, Ethernet y USB, por medio de las cuales se pueden descargar los datos medidos a una computadora para su posterior análisis. Un punto importante de estos equipos es que por lo general poseen certificaciones internacionales como por ejemplo IECs para los procedimientos de medición.

Como ejemplos podemos citar el modelo MVCT de Megger [7] y el modelo iCT1 de Alta Nova [8]. En la tabla 1.1 se muestra una comparativa entre ambos equipos.

TABLA 1.1. Comparación de equipos comerciales.

Características/Ensayos	MVCT (Megger)	ICT1 (Alta Nova)
Apto para transformador de corriente (TI) y tensión (TC)	x	x
Ensayo de relación de transformación	x	x
Resistencia de bobinados	x	x
Desviación de fase	x	x
Curva de saturación (TI)	x	x
Desmagnetización (TI)	x	x
Aislamiento	x	x
Interfaz Ethernet	x	x
Interfaz Wi-Fi		x
Interfaz USB	x	x

1.3. Motivación

Los transformadores son una parte esencial en muchos campos de la industria, son parte fundamental en el sistema de distribución de energía, así como en equipos que son alimentados desde la red. Por esta razón, es de vital importancia su estudio y establecer métodos para conocer su estado antes de su instalación y durante su servicio.

El transformador debe estar en óptimas condiciones al momento de su instalación para evitar posibles fallas del sistema donde es instalado. Un transformador cuyos parámetros esenciales están fuera de especificación puede generar un mal funcionamiento o inclusive sacar de servicio al sistema. Como ejemplo podemos citar los sistemas de distribución de energía eléctrica ya que deben ser sistemas de muy alta disponibilidad, donde un fallo en los componentes de estos sistemas puede devenir en la caída parcial o total del sistema distribución.

En el caso particular de la empresa Iris Tecnología S.R.L., que fabrica equipos electromédicos, necesita conocer si los transformadores provistos son aptos para ser utilizados en sus equipos. Para ello, la empresa cuenta con un sistema de calidad basado en la norma ISO13485:2016.

Entre las tareas y controles necesarios para el sistema de calidad se encuentra el ensayo y calificación de transformadores. Actualmente, esta tarea es realizada manualmente por un operador. Esta labor, además de insumir una gran cantidad de tiempo y ser muy repetitiva, presenta un gran riesgo de error humano y puede comprometer la seguridad del operario y la fiabilidad de los datos obtenidos.

La posibilidad de contar con un equipamiento que, con una mínima intervención del operador, pueda desarrollar los ensayos requeridos representa una gran ventaja para la empresa.

1.4. Objetivos y alcance

1.4.1. Objetivos del trabajo

El objetivo de este trabajo fue el desarrollo de un prototipo de hardware y software que permite automatizar el proceso de caracterización de transformadores de baja tensión utilizado. El prototipo desarrollado es casi autónomo, es decir, solo requiere una mínima intervención del operador para funcionar. Por otro lado, los resultados de la caracterización son enviados a un servidor web proporcionado por el cliente, mostrados en un *display* local e impresos en una etiqueta.

1.4.2. Alcance del trabajo

El presente trabajo tiene como alcance:

- El análisis, investigación y elección del hardware.
- La investigación del modelo de impresora a adquirir, cuyo protocolo debe estar disponible.

- El diseño e implementación del firmware del sistema en lenguaje C sobre FreeRTOS.
- El desarrollo de un prototipo funcional sobre un circuito impreso universal.

Queda excluido del alcance de este trabajo:

- El desarrollo de circuitos de medición de tensión y/o corriente alterna de precisión. Se acepta la precisión obtenida de módulos comerciales como el ZMPT101B.
- El desarrollo de fuentes de tensión alterna de precisión para alimentar los transformadores en ensayo.
- El desarrollo de una aplicación web desde la cual interactuar por Wi-Fi con el módulo.
- El desarrollo del servidor web o API de colección de datos.
- El desarrollo de un prototipo de fabricación escalable que cumpla con todas las certificaciones necesarias.
- El desarrollo de un circuito impreso, más allá del prototipo en placa universal.
- El diseño de circuitos de protección del operario contra descargas eléctricas de alta tensión. En este sentido, se asume que el operario que utilizará el dispositivo es idóneo en el tema. Asimismo, se asume que el cliente posee en sus instalaciones las medidas de seguridad pertinentes para el trabajo con altas tensiones, como disyuntores y puestas a tierra, acorde con la normativa vigente de Higiene y Seguridad en el Trabajo.

Capítulo 2

Introducción específica

En este capítulo se presentan una introducción al trabajo realizado así como los requerimientos del sistema que fueron oportunamente consensuados con el cliente. Posteriormente, se realiza una descripción de las tecnologías utilizadas.

2.1. Estructura general del sistema

En la figura 2.1 se muestra el diagrama en bloques simplificado del sistema. En la figura se observa el dispositivo diseñado, el cual se encuentra dentro de un recuadro, y el transformador a ensayar. Dentro del dispositivo diseñado se observan los siguientes bloques principales:

- Módulo ESP32 con Wi-Fi integrado.
- Bloques de acondicionamiento de señal y actuación para el bobinado primario y secundario.
- Interfaz de usuario: pulsadores, *display* y *buzzer*.
- Interfaz serie para impresora.
- *Switch* de seguridad. Se utiliza para indicar que la tapa de seguridad ha sido removida y el operario puede quedar expuesto a altas tensiones.
- Fuente de alimentación.

El trabajo desarrollado tiene como fin determinar si un transformador dado es apto o no para ser utilizado en determinados equipos. Para tal fin, se configuran umbrales de comparación por medio de una comunicación HTTP a un servidor web y se procede a compararlos contra mediciones realizadas sobre el transformador. Una vez realizadas las comparaciones, los resultados del ensayo se muestran localmente por medio de un *display*. Adicionalmente, los resultados y las mediciones realizadas son enviadas al servidor web del cliente utilizando el protocolo HTTP. Por otro lado, se cuenta con una impresora la cual imprime una etiqueta que permite la trazabilidad del transformador.

Todo el dispositivo es controlado por solo tres pulsadores: Testear, Configurar y Cancelar.

A continuación se detalla un resumen de las tareas realizadas en este trabajo:

- Mediciones de tensión y corriente a los transformadores bajo ensayo.
- Pedido de umbrales de aceptación a un servidor web.

- Emisión de aceptación o rechazo del transformador.
- Visualización en el *display* de los umbrales configurados y las mediciones realizadas.
- Envío de las mediciones a un servidor web por medio del protocolo HTTP.
- Gestión de la conexión Wi-Fi.
- Impresión de etiquetas.

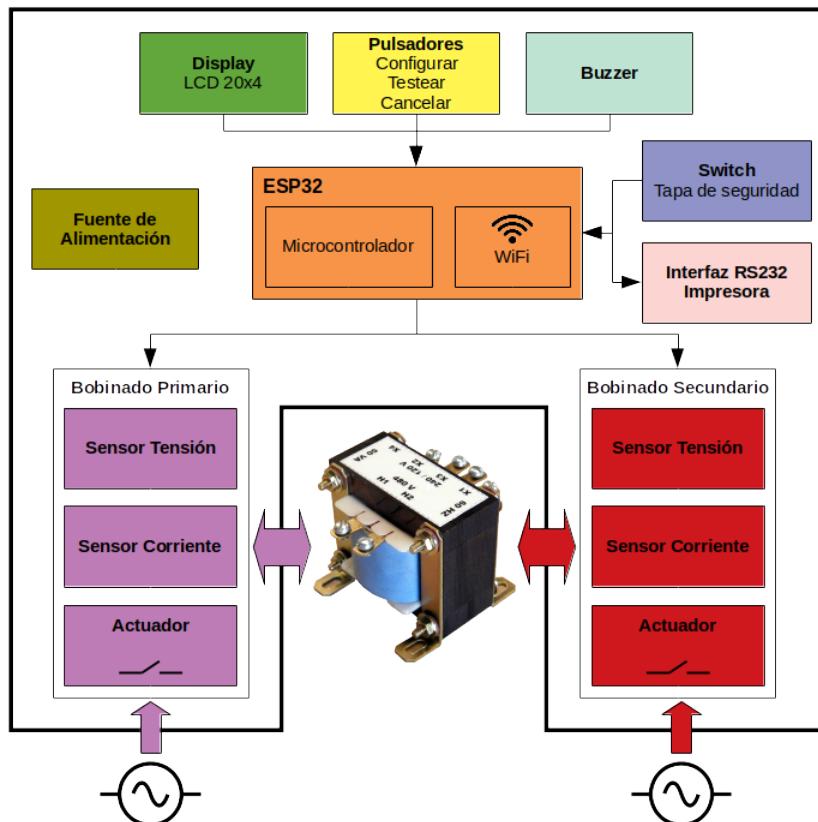


FIGURA 2.1. Diagrama en bloques del sistema.

En la figura 2.2 se presenta un diagrama de secuencia simplificado del dispositivo cuyos estados se describen a continuación:

1. Inicialización del sistema.
2. Conexión a red Wi-Fi.
3. Espera por pulsadores. El operador debe pulsar Configurar para leer los umbrales de configuración desde el servidor web.
4. Luego, el operador debe pulsar Testear para iniciar la caracterización del transformador. Aquí se verifica que el equipo haya sido configurado previamente ya que no se puede caracterizar un transformador si no cuenta con los umbrales de comparación. En caso de estar configurado se procede a los siguientes pasos.
5. Energizar el bobinado primario con $230 \text{ V}_{\text{RMS}}$ estabilizados (provistos externamente por el cliente).

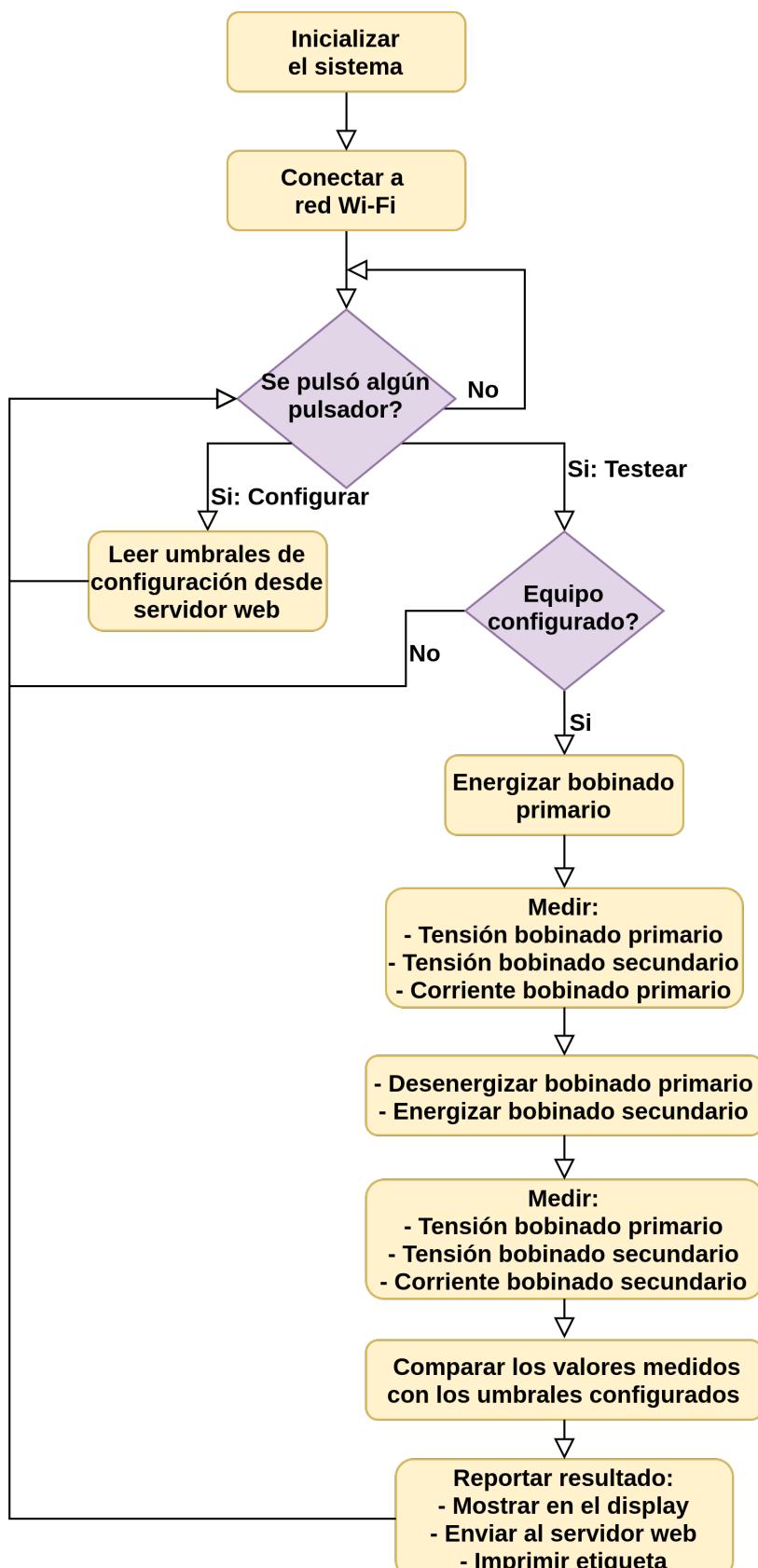


FIGURA 2.2. Diagrama de secuencia simplificado.

6. Medir:
 - Tensión en bobinado primario.
 - Corriente que circula por el bobinado primario.
 - Tensión en bobinado secundario.
7. Desenergizar el bobinado primario.
8. Energizar el bobinado secundario con la tensión adecuada. Esta tensión se genera internamente a partir de la tensión de 230 V_{RMS}.
9. Medir:
 - Tensión en bobinado primario.
 - Tensión en bobinado secundario.
 - Corriente que circula por el bobinado secundario.
10. Desenergizar el bobinado secundario.
11. Comparar los valores medidos con los umbrales configurados previamente y determinar si el transformador es aceptado o rechazado.
12. Reportar los valores medidos y los resultados obtenidos al:
 - Servidor web.
 - En el *display* local.
 - Imprimir una etiqueta.

2.2. Requerimientos

A continuación se enumeran los requerimientos consensuados con el cliente:

2.2.1. Requerimientos de hardware

1. El dispositivo debe ser diseñado en base al módulo ESP32.
2. El dispositivo debe ser capaz de medir tensiones y corrientes alternas de manera aislada del transformador bajo ensayo.
3. El dispositivo debe ser capaz de comutar las tensiones primarias y secundarias.
4. El dispositivo debe tener un *display* LCD alfanumérico de 20x4 caracteres.
5. El dispositivo debe tener una interfaz Wi-Fi.
6. El dispositivo debe tener una interfaz RS232 capaz de manejar impresoras series.
7. El dispositivo debe alimentarse desde la red eléctrica Argentina 220 V_{RMS}/50 hz, debiendo proveerse todas las alimentaciones a los diferentes módulos de hardware.

8. El dispositivo debe poseer un *switch* para indicar que la tapa de seguridad ha sido removida.
9. El dispositivo debe poseer tres pulsadores nombrados Configurar, Testear y Cancelar.
10. El dispositivo debe poseer un *buzzer*.

2.2.2. Requerimientos relativos a los valores a medir

1. Bobinado primario:
 - a) Rango tensión: 100 – 240 V_{RMS}.
 - b) Rango corriente: 0 – 800 mA_{RMS}.
2. Bobinado secundario:
 - a) Rango tensión: 0 – 30 V_{RMS}.
 - b) Rango corriente: 0 – 1500 mA_{RMS}.
3. Precisión en la medición de tensión: mejor al 1,5% (puede variar en base a lo que se pueda conseguir en el mercado).
4. Precisión en la medición de corriente: mejor al 4% (puede variar en base a lo que se pueda conseguir en el mercado).

2.2.3. Requerimientos funcionales

1. El dispositivo debe permitir que se configuren los umbrales mínimos y máximos para los parámetros medidos.
2. Los valores a configurar deben ser adquiridos solo por Wi-Fi, no se admite configuración local por teclado.
3. El dispositivo debe generar una confirmación de aceptación o rechazo del transformador en ensayo basado en los umbrales mínimos y máximos preseeados.
4. El dispositivo, luego de cada ensayo, independientemente del resultado, debe imprimir una etiqueta con la siguiente información:
 - a) Número de partida del transformador ensayado.
 - b) Condición de aceptado o rechazado.
 - c) Valores medidos de tensiones y corrientes.

2.2.4. Requerimientos de comunicación

1. Solicitar parámetros de configuración: el dispositivo debe generar un comando GET de HTTP a un servidor web (provisto por el cliente) para obtener los umbrales de aceptación y el número de partida del transformador a ensayar.

2. El dispositivo debe ser capaz de procesar la respuesta del comando GET que está en formato JSON.
3. Enviar resultados del ensayo: el dispositivo debe generar un comando POST de HTTP a un servidor (provisto por el cliente) con la información del transformador ensayado en formato JSON.

2.2.5. Requerimientos de interfaz de usuario

1. Sobre la funcionalidad de los pulsadores:
 - a) Configurar: al pulsar este pulsador el dispositivo debe generar el comando GET para solicitar al servidor web los parámetros de configuración del dispositivo a través de Wi-Fi.
 - b) Testear: al pulsar este pulsador el dispositivo debe iniciar la secuencia de testeo.
 - c) Cancelar: al pulsar este pulsador la secuencia de testeo en curso debe quedar abortada.
2. Sobre la funcionalidad del *display LCD*:
 - a) El dispositivo debe mostrar los umbrales configurados y los valores de las mediciones actuales.
 - b) Los valores de los umbrales configurados deberán permanecer en el *display* durante el ensayo.
 - c) El dispositivo debe mostrar los valores medidos del transformador ensayado luego de cada medición.
 - d) Luego de finalizado el ensayo, se debe mostrar un mensaje que indique que la información se ha enviado al servidor web y mantenerse el dispositivo bloqueado hasta que se haya recibido la respuesta del servidor.
3. Sobre el *buzzer*:
 - a) El dispositivo debe emitir un solo sonido de 0,5 segundos de duración para confirmar la aceptación del transformador.
 - b) El dispositivo debe emitir un sonido de repetición de 5 ciclos de 0,5 segundos de duración, con pausas de 0,5 segundos para confirmar el rechazo del transformador.

2.3. Kit ESP32-DevKitC

Para este trabajo se utilizó el kit de desarrollo ESP32-DevKitC [9] desarrollado por la empresa Espressif Systems, figura 2.3. Este kit integra un módulo ESP32-WROOM-32 [10] de la misma empresa, figura 2.4. El módulo ESP32-WROOM-32

es un microcontrolador de doble núcleo con interfaces Wi-Fi y Bluetooth. Además de las interfaces anteriores, el módulo cuenta con interfaces UART, SPI, numerosas entradas-salidas de propósito general, conversores analógico-digital y conversores digital-analógico.



FIGURA 2.3. ESP32-DevKitC V1.



FIGURA 2.4. ESP32-WROOM-32.

De las especificaciones del módulo ESP32-WROOM-32 se pueden destacar las siguientes:

- Procesador dual core Xtensa® LX6 de 32 bits.
 - Velocidad de reloj de hasta 240 Mhz.
 - 520 KB de RAM.
 - 4 MB SPI flash.
 - Wi-Fi integrado con posibilidad de trabajar como AP (*Access Point*) y SM (*Station Mode*).
 - Bluetooth V4.2.
 - 36 GPIO.

- Hasta 18 conversores analogico-digital (ADC) de 12 bits de resolución.
- 2 conversores digital-analógico (DAC) de 8 bits de resolución.
- 3 UART.
- 2 canales I2C.
- 4 canales SPI.
- Interfaz JTAG.

Dado que el ESP32-DevKitC integra al módulo ESP32-WROOM-32, puede que no todos sus periféricos y entradas-salidas de propósito general (GPIO por sus siglas en inglés) estén disponibles.

A continuación se detallan las principales características del kit ESP32-DevKitC:

- Doble hilera de pines con casi todas las entradas-salidas de propósito general del ESP32-WROOM-32.
- Puente USB-UART conectado al módulo ESP32-WROOM-32. Esta interfaz es muy útil para programación y depuración.
- Regulador LDO para proveer alimentación a los elementos del kit.

2.3.1. Entorno de desarrollo ESP-IDF

Espressif Systems provee un entorno de desarrollo de software denominado ESP-IDF [11]. El entorno ESP-IDF contiene todo lo necesario para desarrollar aplicaciones para los módulos ESP-32 sobre cualquier sistema operativo: Windows, Linux o MAC OS. Este entorno de desarrollo es *Open-Source* y puede ser clonado desde un repositorio de GitHub¹. La empresa constantemente realiza actualizaciones y correcciones de fallas.

A continuación se listan algunas de las características más destacables del entorno ESP-IDF:

- Soporta FreeRTOS [12].
- Soporta diferentes controladores de periféricos (SPI, I2C, UART, GPIO, I2S, ADC, DAC, etc) [13].
- Soporta librerías para Wi-Fi [14] y Bluetooth [15].
- Soporta varios *stacks* de redes, por ejemplo TCP/IP.
- Soporta varias implementaciones de protocolos (DHCP cliente y servidor, HTTP cliente y servidor, MQTT, etc) [16].
- Soporta extensiones para Eclipse [17] y Visual Code [18].
- Está basado en CMake [19].

El entorno ESP-IDF no es parte del proyecto del usuario sino que debe ser enlazado por medio de la variable de entorno IDF_PATH. Esto último ayuda a separar

¹Repositorio GitHub para el ESP-IDF <https://docs.platformio.org/en/latest/frameworks/espidf.html>

el entorno de desarrollo del proyecto particular del usuario. Para utilizar ESP-IDF se requiere una estructura particular de archivos y carpetas para el proyecto [20].

2.3.2. ESP-Touch y SmartConfig

Espressif Systems provee la aplicación móvil ESP-Touch [21] la cual puede ser usada con la librería SmartConfig [22] para configurar las credenciales de Wi-Fi en equipos que no posean una interfaz de usuario acorde para insertar esta información. En la figura 2.5 se muestra la interfaz de la aplicación ESP-Touch.

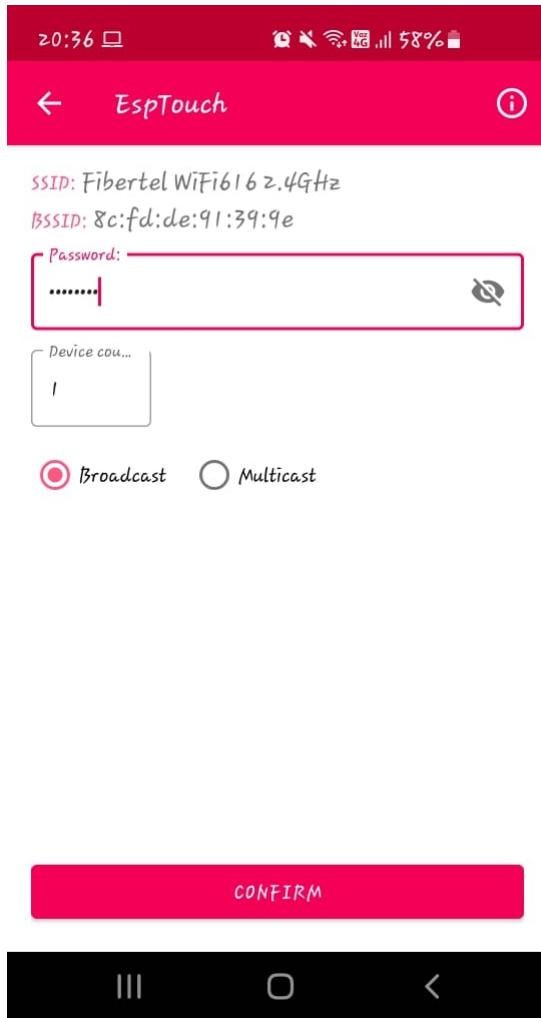


FIGURA 2.5. Aplicación Android ESP-Touch.

La ventaja de esta tecnología es que el dispositivo no necesita conocer directamente el SSID o la contraseña de un punto de acceso (AP), en cambio esta información se proporciona mediante el teléfono móvil. ESP-Touch está disponible para Android e iOS.

2.4. Sensor de tensión

Para monitorear las tensiones del bobinado primario y secundario se utilizó el módulo de figura 2.6.

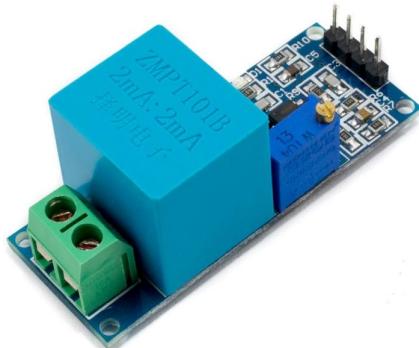


FIGURA 2.6. Módulo sensor de tensión alterna.

Este módulo es un sensor de tensión capaz de medir tensiones alternas de línea de 220 V_{RMS} de forma aislada. En la figura 2.7 se puede observar la tensión de entrada (Vin) y la tensión de salida (Vout) entregada por el módulo. El módulo provee una tensión de salida la cual está montada sobre un nivel continua igual a la mitad de la tensión de alimentación, esto ayuda a procesar la señal directamente con un ADC sin la necesidad de fuentes de alimentación negativas.

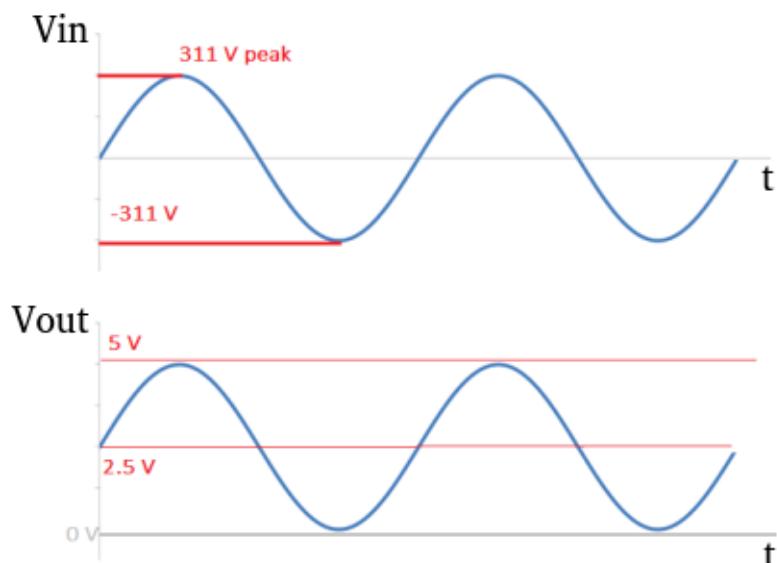


FIGURA 2.7. Ondas de entrada (Vin) y salida (Vout) del módulo @ Vcc = 5 V.

El elemento principal del módulo es el transformador de corriente ZMPT101B de la empresa Qingxian Zeming Langxi Electronic [23]. Este componente es el encargado de proporcionar la aislación entre la alta tensión de línea y la baja tensión hacia el microcontrolador. El primario del transformador se conecta a la tensión alterna de la red a través de la bornera verde que se observa en la figura 2.6. En el lado secundario del transformador se tiene una resistencia serie (*shunt*) y un circuito amplificador basado en el operacional LM358 [24]. Adicionalmente, el circuito posee un potenciómetro para ajustar la ganancia del sistema.

Este módulo se utiliza en aplicaciones de domótica e IoT (Internet of Things) para el monitoreo de la tensión de línea.

Especificaciones técnicas del módulo:

- Tensión de alimentación (Vcc): 5-30 V.
- Tensión alterna de entrada máxima: 250 V_{RMS}.
- Tensión de salida: onda senoidal 2,5 V_{PICO} @ Vcc = 5 V.
- Valor medio de la tensión salida: 2,5 V @ Vcc = 5 V.
- Dimensiones: 5 cm x 2 cm x 2,4 cm.

2.5. Sensor de corriente

Para monitorear las corrientes del bobinado primario y secundario se utilizó el módulo de figura 2.8.

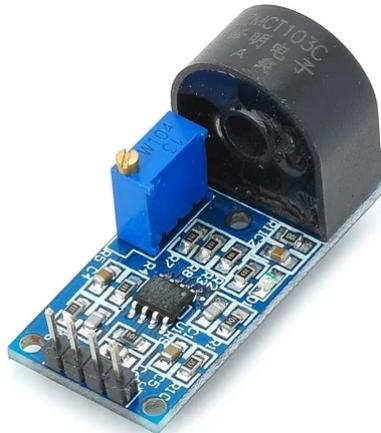


FIGURA 2.8. Módulo sensor de corriente alterna.

Este módulo facilita el monitoreo de corriente alternas de hasta 5 A_{RMS}. Está basado en el transformador de corriente de alta precisión ZMCT103C de la empresa Qingxian Zeming Langxi Electronic [25]. El módulo es similar al módulo sensor de tensión presentado en la sección 2.4. La corriente de salida del transformador de corriente pasa por una resistencia serie (*shunt*) y luego es acondicionada por los amplificadores operaciones integrados en el LM358 [24]. La única diferencia entre el módulo sensor de corriente y el de tensión, es que el primero entrega una tensión alterna con valor medio cero, es decir, sin componente de continua. Esto último hace necesario alguna modificación adicional del módulo para entregar una tensión que sea siempre positiva.

Especificaciones técnicas del módulo:

- Tensión de alimentación (Vcc): 5-30 V.
- Corriente de entrada nominal: 5 A_{RMS}.
- Tensión de salida: onda senoidal 2.5 V_{PICO} @ Vcc = 5 V.
- Valor medio de la tensión salida: 0 V @ Vcc = 5 V.

2.6. Impresora

Para este trabajo se utilizó una impresora de transferencia térmica modelo E-Class™ Mark III fabricada por Honeywell [26], ver figura 2.9. Las impresoras por transferencia térmica utilizan una cinta de tinta denominada *Ribbon* que con el calor del cabezal derrite la tinta sobre el papel. Este fenómeno permite imprimir códigos de barras, textos, etc. Estas impresoras son de bajo costo pero ofrecen como desventaja que solo imprimen en blanco y negro.



FIGURA 2.9. Impresora E-Class™ Mark III modelo E-4204B.

Especificaciones técnicas del modelo E-4204B:

- Resolución: 8 puntos/mm (203 ppp).
- Velocidad máxima de impresión: 101 mm/s (4 pps).
- Ancho máximo de papel: 112 mm.
- Capacidad de rollo: diámetro externo 127 mm.
- Memoria: DRAM de 16 MB/Flash de 8 MB.
- Comunicación: USB 2.0 y RS232 serie.
- Lenguajes soportados: DPL(Datamax), ZPL(Zebra), EPL(Eltron), BPL(Boca), IPL(Intermec).
- Indicadores de estado: dos indicadores luminosos de tres colores.
- Dimensiones: 187 mm x 203,5 mm x 282 mm.
- Peso: 2,4 Kg.

2.6.1. Protocolo DPL

El protocolo DPL (Datamax-O'Neil Programming Language) es un protocolo propietario de la firma Datamax el cual puede ser utilizado para comunicarse con las impresoras E-Class™ Mark III [27]. Este es un protocolo punto a punto de tipo ASCII basado en la arquitectura maestro/esclavo. La impresora se conecta con el *host* sin intermediarios.

El protocolo está formado por comandos y parámetros asociados a estos comandos. En la tabla 2.1 se muestran los comandos disponibles.

TABLA 2.1. Tipos de comandos protocolo DPL.

ASCII (HEX)	Tipo	Descripción
SOH (0x01)	Comando inmediato	Cuando la impresora recibe un comando inmediato, interrumpe su operación actual y pasa a ejecutarlo. Ejemplos: reinicio de la impresora, pedido de estado de la impresora.
STX (0x02)	Comando de sistema	Es el tipo de comando más comúnmente utilizado. Ejemplos: configurar el modo de trabajo métrico o imperial, tipo de alineación del texto a imprimir, indica el inicio y fin del texto a imprimir, etc.
ESC (0x1B)	Comando de carga de fuente	Este comando es utilizado para cargar una nueva fuente. Generalmente es utilizado por programas específicos para la creación de fuentes.

En las siguientes secciones se describen los comandos inmediatos y los comandos de sistema ya que fueron los utilizados en este trabajo.

Comandos inmediatos

En la tabla 2.2 se muestran los comandos inmediatos más utilizados, en la última columna se muestra la respuesta de la impresora.

TABLA 2.2. Comandos inmediatos protocolo DPL.

ASCII (HEX)	Descripción	Respuesta
<SOH>A<CR> (0x01,0x41,0x0D)	Pedido de estado de la impresora.	La respuesta son 8 caracteres 'Y' o 'N' mas el carácter <CR> de final de mensaje. abcdefg<CR> a: Interprete de comandos ocupado b: Falla de papel c: Falla del <i>Ribbon</i> d: Imprimiendo lote (se pueden enviar varias etiquetas para imprimir) e: Impresora ocupada f: Impresora pausada g: Etiqueta presentada h: Falla interna
<SOH>*<CR> (0x01,0x2A,0x0D)	Pedido de reinicio de la impresora	<XON>R<CR>

En el caso del comando <SOH>A<CR>, la respuesta de la impresora es una cadena del tipo "YYNNYY<CR>", donde cada carácter 'Y' o 'N' se corresponde con los caracteres "abcdefg<CR>" de la tabla 2.2. Para poder imprimir, la impresora debe devolver la cadena "YYYYYYYY<CR>" la cual indica que no tiene errores ni está ocupada con otra tarea.

Comandos de sistema

El principal uso de los comandos de sistema es enviar a la impresora los caracteres que se desean imprimir. Además de los caracteres a imprimir, se pueden

configurar diferentes aspectos de la impresora tales como: temperatura del cabezal, alineación del texto, tipo y tamaño de la fuente, posición del texto, etc.

El comando <STX>L<CR> es el más utilizado. Este comando permite enviarle a la impresora una línea de texto a imprimir con su formato y ubicación. En la tabla 2.3 se muestra el formato para imprimir una línea de texto.

TABLA 2.3. Comandos de sistema protocolo DPL.

Mensaje	Tipo	Descripción	Cantidad de caracteres	Ejemplo
a	Número	Dirección de rotación del texto: 1 = 0°, 2 = 90°, 3 = 180°, 4 = 270°	1	1
b	Número	Fuente a utilizar	1	3
c	Número	Multiplicador de ancho de fuente	1	1
d	Número	Multiplicador de alto de fuente	1	1
eee	Número	000	3	000
ffff	Número	Posición vertical del texto en mm	4	0140
gggg	Número	Posición horizontal del texto en mm	4	0000
jj...j	Caracteres	Texto a imprimir	Largo de la cadena	10K OHM 1/4 WATT

En el ejemplo de la tabla 2.3 se imprime la etiqueta mostrada en la figura 2.10.

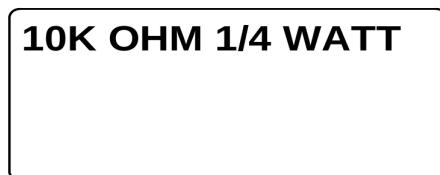


FIGURA 2.10. Etiqueta resultante del comando de la columna Ejemplo de la tabla 2.3.

Se pueden enviar varias líneas a imprimir en un solo comando, en este caso, para indicar el fin del texto se debe enviar E<CR>.

2.6.2. Módulo adaptador RS232

Para comunicar la impresora con la UART del microcontrolador se utilizó el módulo de la figura 2.11.

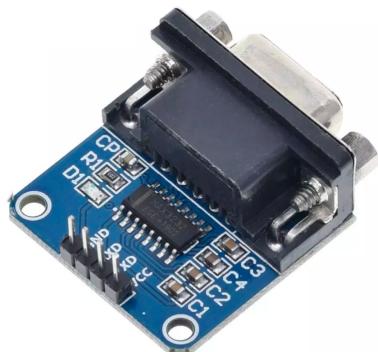


FIGURA 2.11. Módulo adaptador RS232 a TTL.

Este módulo está basado en el integrado MAX3232 [28] y adapta las señales del capa física RS232 a valores acordes para trabajar con el microcontrolador.

2.7. Display alfanuméricico

Para este trabajo se utilizó un *display* alfanumérico de 20 columnas por 4 líneas con controlador HD44780 de Hitachi [29], ver figura 2.12.



FIGURA 2.12. *Display* alfanumérico de 20x4.

El *display* cuenta con una interfaz paralela de 8-bits (DB0-7) la cual puede ser utilizada en formato 4-bits haciendo 2 escrituras o lecturas. Cuenta también con tres líneas de control denominadas RS, R/W y EN. En la tabla 2.4 se muestran las señales necesarias para el manejo del *display*.

TABLA 2.4. Señales de control y datos *display*.

Señal	Tamaño bits	Descripción
RS	1	Indica como interpretar los datos en el <i>bus</i> de datos DB. 0: Instrucción 1: Datos
R/W	1	0: Operación de lectura 1: Operación de escritura
EN	1	<i>Enable</i> : utilizado para indicar que hay datos validos en el <i>bus</i> de datos
DB	8	<i>Bus</i> de datos

Para utilizar el *display* se cuenta con 2 tipos de operaciones las cuales se diferencian por el bit de control RS:

- RS=0: Escribir/leer una instrucción.
- RS=1: Escribir/leer un carácter ASCII.

En la tabla 2.5 se muestran las instrucciones más utilizadas.

TABLA 2.5. Instrucciones más utilizadas para el controlador HD44780.

Nombre	Valor hexa	Acción
Borrado	0x01	Borra la pantalla completa
Modo de entrada	0x04	Fijar dirección del cursor: izquierda o derecha
Control de encendido y apagado	0x08	<ul style="list-style-type: none"> - Encender o apagar la pantalla - Encender o apagar el cursor - Fijar tamaño del <i>bus</i> de datos
Fijar <i>Set</i>	0x10	<ul style="list-style-type: none"> - Fijar número de líneas - Fijar fuente de los caracteres

2.8. Módulos misceláneos

En esta sección se describe el módulo utilizado como elemento de maniobra para alimentar los bobinados del transformador y la fuente de alimentación utilizada en el trabajo.

Módulo de relés

Para poder conmutar las tensiones aplicadas a los bobinados del transformador bajo ensayo se utilizó el módulo de la figura 2.13 . Este módulo está compuesto por 8 relés los cuales pueden ser manejados en forma aislada por el microcontrolador a través de optoacopladores. Este módulo permitió actuar sobre las altas tensiones del transformador bajo ensayo de una manera segura.

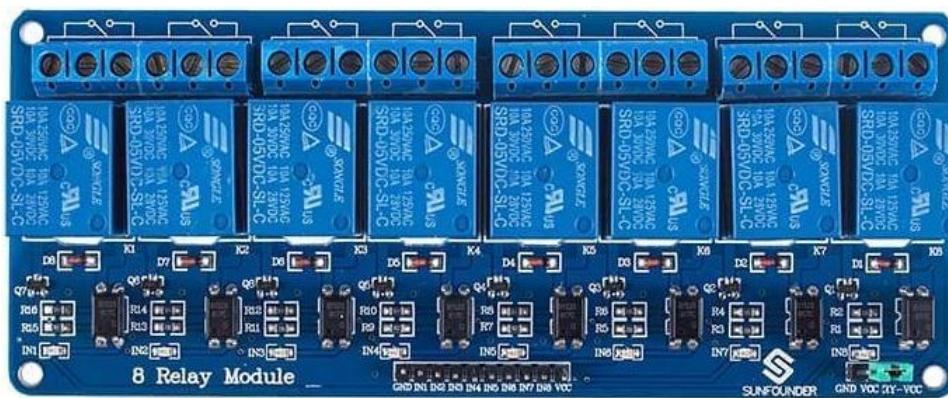


FIGURA 2.13. Módulo de relés.

Módulo de alimentación

El dispositivo diseñado debe ser alimentado desde la red eléctrica de Argentina (220 V_{RMS} - 50 hz) para tal fin, se utilizó la fuente conmutada mostrada en la figura 2.14.



FIGURA 2.14. Módulo de alimentación.

Especificaciones técnicas²:

- Tensión de entrada: 85 a 265 V_{RMS} 50 hz.
- Corriente de entrada: 0,0273 A (110 V_{RMS}) y 0,014 A (220 V_{RMS}).
- Tensión de salida: 5 V +/- 0,2 V.
- Corriente de salida: 700 mA (800 mA_{PICO}).
- Ripple: 60 mV.
- Potencia: 3,5 W.
- Eficiencia: 80 %.
- Protección contra corto circuito.
- Temperatura de operación: -20 a 60 °C.

²https://articulo.mercadolibre.com.ar/MLA-700576819-fuente-aislada-switching-220v-5v-700ma-35w-20off-nut-JM#reco_item_pos=2&reco_backend=machinalis-v2p-pdp-boost-v2&reco_backend_type=low_level&reco_client=vip-v2p&reco_id=90222827-7627-46cb-9f2b-78620eec7ea2

Capítulo 3

Diseño e implementación

En el presente capítulo se describen el hardware y firmware implementados en el trabajo. Se detalla la integración de los módulos de hardware presentados en el capítulo anterior con el kit ESP32-DevKitC. Luego, se presenta el desarrollo del firmware y se fundamentan las decisiones tomadas en el diseño.

3.1. Descripción de hardware

En esta sección se detalla como se conectaron los diferentes módulos de hardware presentados en el capítulo 2 con el kit ESP32-DevKitC. En la figura 3.1 se muestra un diagrama de conexionado entre el kit ESP32-DevKitC y los diferentes módulos.

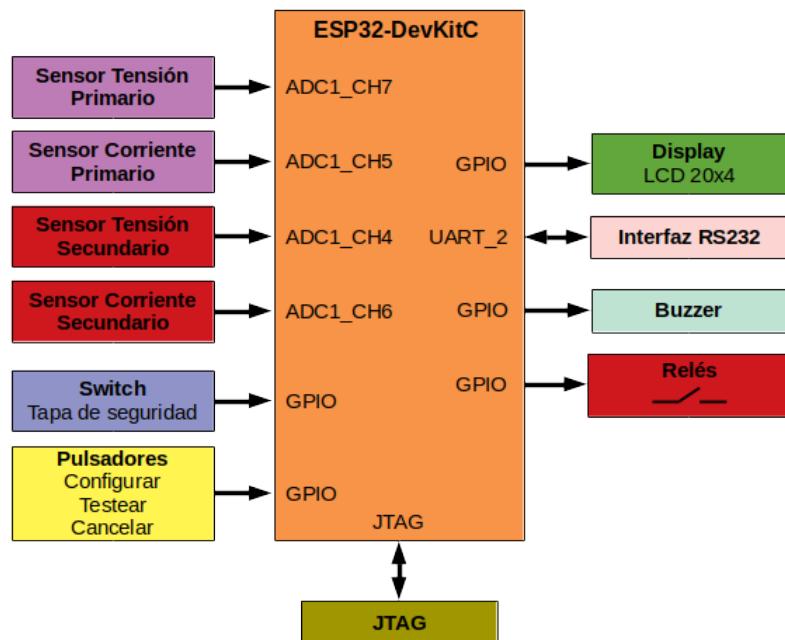


FIGURA 3.1. Conexionado kit ESP32-DevKitC y módulos de hardware.

Para el conexionado de los elementos del sistema se utilizó una placa universal de 90 mm x 150 mm. Esta permitió un rápido conexionado de los módulos con el kit y la posibilidad de trabajar con hardware y software sin la necesidad de contar con todo el hardware diseñado y construido. En la placa universal se montaron los siguientes componentes:

- Kit ESP32-DevKitC.
- Dos fuentes de alimentación de 220 V_{RMS} a 5 V (sección 2.8).
- El *buzzer*.
- Componentes varios necesarios para el apropiado funcionamiento de los módulos analógicos con el kit de desarrollo.
- Diferentes conectores tipo *header* para el conexionado de los módulos de hardware presentados en el capítulo 2.

En la tabla 3.1 se muestra el uso de los pines del kit ESP32-DevKitC. En general, todos los módulos fueron conectados directamente al kit con la excepción de los módulos sensores de tensión y corriente cuyo conexionado requirió divisores resistentivos para adaptar los valores de salida de los módulos al rango de excusión del ADC. A su vez que si incluyó en la placa universal una referencia de tensión basada en el integrado TL431 [30], esto se explica con más detalle en la sección 3.1.1.

TABLA 3.1. Uso de los pines del kit ESP32-DevKitC.

Pines kit	Módulo	Símbolo
Medición de corrientes y tensiones del transformador		
IO35/ADC1_CH7	Sensor de tensión primario	PV
IO33/ADC1_CH5	Sensor de corriente primario	PC
IO32/ADC1_CH4	Sensor de tensión secundario	SV
IO34/ADC1_CH6	Sensor de corriente secundario	SC
IO25	Alimentación bobinado primario	CPV
IO26	Alimentación bobinado secundario	CSV
<i>Display</i>		
IO18	Enable	EN
IO5	RS	RS
IO19	Bus de datos bit 4	D4
IO21-23	Bus de datos bits 5 a 7	D5-7
Interfaz RS232		
IO16/RXD2	Recepción	RX
IO17/TXD2	Transmisión	TX
Pulsadores		
IO39	Testear	PTEST
IO36	Configurar	PCONF
IO27	Cancelar	PCAN
Otros		
IO4	<i>Buzzer</i>	BUZZ
IO2	<i>Switch</i> de seguridad	SWITCH
IO12-15	JTAG	-

3.1.1. Mejoras a las entradas analógicas

En las secciones 2.4 y 2.5 se introdujeron los módulos de hardware sensores de tensión y corriente. Estos módulos son muy prácticos debido a que proporcionan valores de tensiones acordes para trabajar con el ADC del kit ESP32-DevKitC y,

a su vez, proporcionan aislamiento eléctrico de las altas tensiones de los bobinados. Sin embargo, al mirar con más detalle los módulos, se pueden identificar las siguientes desventajas:

1. El amplificador LM358 tiene saturaciones fuertemente asimétricas, $V_{SAT+} = V_{cc}-1,5 \text{ V}$ @ $V_{cc} = 5 \text{ V}$ y $V_{SAT-} \simeq 0 \text{ V}$ @ $V_{cc} = 5 \text{ V}$. Teniendo en cuenta que el *offset* de salida de los módulos está a la mitad de la alimentación, la saturación positiva limita la excursión de los módulos y por lo tanto, el rango aprovechable del ADC, figura 3.2.
2. En el caso del sensor de corriente, este no posee *offset* en la salida, esto genera tensiones negativas incompatibles con el ADC del kit.
3. La tensión de alimentación se utiliza como tensión de referencia para generar los 2,5 V de *offset* a la salida del módulo. Si esta tensión no es estable, toda la medición se verá comprometida.

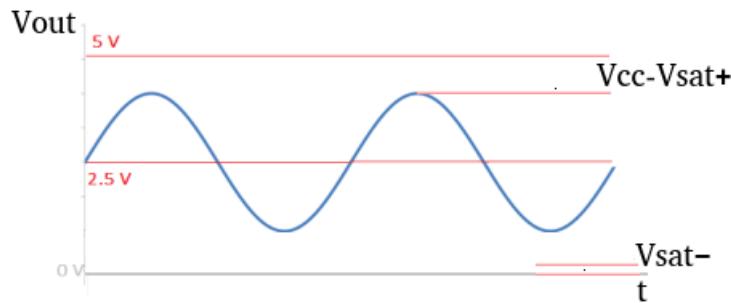


FIGURA 3.2. Saturación módulos sensores.

Luego de estudiar los circuitos, se decidió tomar las siguientes acciones para solucionar los problemas encontrados:

1. Se modificó el divisor resistivo que fijaba la tensión de *offset* en la mitad de la alimentación a un valor más acorde, figura 3.3.
2. Se eliminó el capacitor de desacople serie que estaba en la salida del sensor de corriente. Esto permitió que la salida de este módulo excursionase solo entre valores positivos.
3. Se decidió alimentar los módulos con una referencia de tensión (TL431 [30]) en vez de utilizar la tensión de alimentación de la fuente generada en la placa universal.

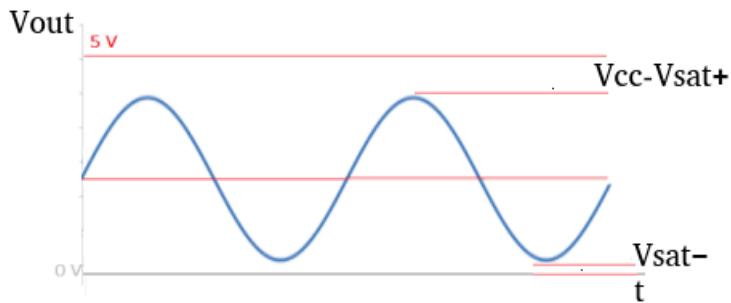


FIGURA 3.3. Saturación mejorada.

3.1.2. Integración de hardware

En la figura 3.4 se muestra el trabajo realizado. Como se puede notar, este consta de dos gabinetes los cuales están conectados por cables que se observan en la parte derecha de la figura.



FIGURA 3.4. Trabajo final terminado.

En el gabinete de la parte superior de la figura, denominado gabinete principal, se encuentran los diferentes módulos presentados en el capítulo 2 los cuales son conectados al kit ESP32-DevKitC a través de la placa universal. Por otro lado, el gabinete de la parte inferior, denominado gabinete auxiliar, está destinado a albergar el transformador que se desea ensayar. En la figura, además, se pueden observar el *display* alfanumérico y los tres pulsadores los cuales fueron rotulados acorde a su función.

Como se dijo, en el gabinete auxiliar debe ser colocado el transformador a ensayar, para esto, este cuenta con una tapa gris denominada tapa de seguridad. En la figura 3.5 se muestran los mismos gabinetes pero se encuentra la tapa de seguridad abierta y se aprecia un transformador de prueba colocado en dicho gabinete.



FIGURA 3.5. Trabajo terminado con la tapa de seguridad abierta.

En la figura 3.6 se muestra el gabinete principal sin la tapa. Se pueden distinguir los siguientes componentes:

- Placa universal con el kit ESP32-DevKitC y componentes asociados.
- Dos módulos sensores de tensión, sección 2.4.
- Dos módulos sensores de corriente, sección 2.5.
- El módulo de relés, sección 2.8.
- El módulo adaptador RS232, sección 2.6.2.
- Un transformador que se utiliza como transformador auxiliar para generar la tensión alterna necesaria para alimentar el transformador bajo prueba.

Es importante destacar que el armado de los gabinetes fue realizado desde cero con las herramientas disponibles en el hogar. Este demandó mucho tiempo, esfuerzo y dedicación. Aún cuando no es el objetivo del posgrado el armado de gabinetes, creo que es importante resaltar este punto ya que el trabajo terminado, a pesar de ser un prototipo, está en condiciones de ser utilizado como instrumento de medición en un ambiente industrial.

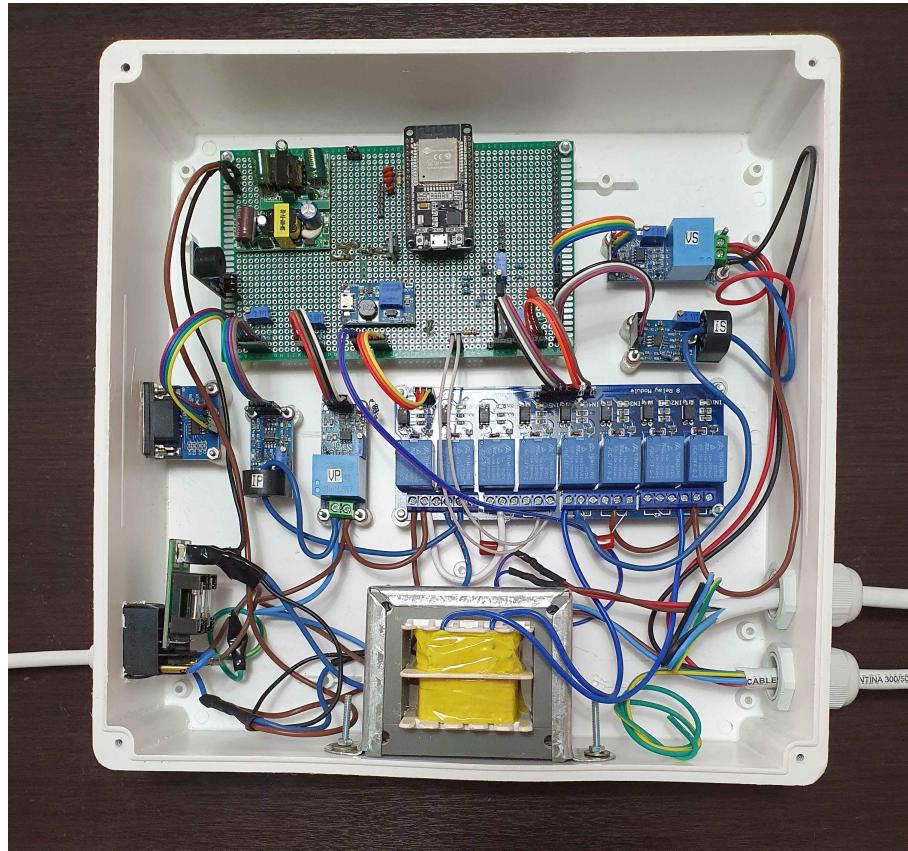


FIGURA 3.6. Contenido gabinete principal.

En la figura 3.7 se muestra el gabinete auxiliar donde se pueden observar los diferentes conectores para el conexionado del transformador a ensayar.



FIGURA 3.7. Gabinete auxiliar sin el transformador a ensayar.

3.2. Descripción de firmware

En esta sección se describe la arquitectura de firmware adoptada, así como los módulos principales que la componen, se brinda detalle de las diferentes implementaciones y se destacan sus aspectos más importantes.

3.2.1. Arquitectura de firmware

Luego de analizar posibles patrones de arquitectura de software a utilizar, se decidió utilizar el patrón observar y reaccionar visto en Ingeniería de Software [31], figura 3.8. Este patrón se utiliza cuando un conjunto de sensores se monitorean y muestran de manera rutinaria.

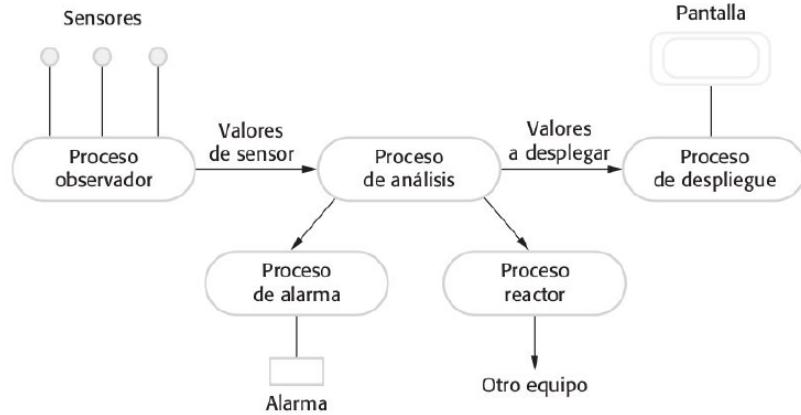


FIGURA 3.8. Patrón observar y reaccionar.

El dispositivo diseñado responde muy bien a esta arquitectura ya que se deben monitorear los sensores (pulsadores, monitores de tensiones y corrientes), accionar actuadores y enviar los resultados a procesos de salida (*display*, servidor web e impresora). En ningún momento se tienen lazos de realimentación o estructuras que rompan la secuencialidad del sistema. En la figura 3.9 se muestra el patrón aplicado al trabajo.

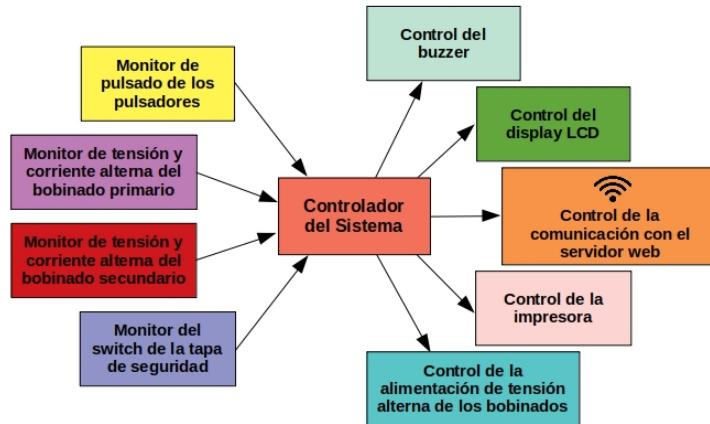


FIGURA 3.9. Patrón observar y reaccionar aplicado al trabajo.

3.2.2. Estructura general del firmware

El firmware fue desarrollado por medio de la utilización de buenas prácticas de programación como por ejemplo el control de versiones, la modularización y la documentación en el código fuente. Para llevar adelante el control de versiones se creó un repositorio en GitHub [32] desde el inicio del trabajo. En cuanto a la modularización, se creó una estructura de módulos/archivos como la siguiente:

```

TP_CESE
├── inc
│   ├── app_adc.h
│   ├── app_Comm.h
│   └── ..
├── sch
└── src
    ├── app_adc.c
    ├── app_Comm.c
    ├── ..
    ├── main.c
    └── ..
├── .gitignore
└── CMakeLists.txt
├── sdkconfig
└── Doxyfile
└── README.md

```

Donde:

- inc: directorio para ubicar todos los archivos de cabeceras.
- sch: directorio con el esquemático de la placa universal en KiCad [33].
- src: directorio para ubicar todos los archivos fuentes.
- CMakeLists.txt: es el archivo principal que usa CMake para construir el proyecto.
- sdkconfig: este archivo contiene la configuración de todos los componentes del proyecto (incluido ESP-IDF).
- Doxyfile: archivo de configuración para generar la documentación a través de Doxygen.

A continuación se detallan los principales módulos de software desarrollados desde la perspectiva de su archivo de cabecera:

- adc.h: implementación del manejo de los ADCs para leer las tensiones y corrientes de los bobinados.
- app_Comm.h: se utiliza para “parsear” los paquetes HTTP enviados y recibidos desde el servidor web.
- app_error.h: se utiliza para definir un criterio de error común frente a las diferentes fallas del sistema.
- app_fsm.h: máquina de estado del controlador principal.
- app_gpio.h: se utiliza para definir las diferentes funciones para el manejo de entradas-salidas de propósito general como por ejemplo los pulsadores, el comando de alimentación de bobinados, etc.
- app_lcd.h: define las rutinas necesarias para escribir mensajes en el *display*.
- app_printer.h: implementación del protocolo DPL y manejo del puerto RS232.
- app_WiFi.h: implementación de las diferentes rutinas para el manejo del protocolo Wi-Fi.

- http_client.h: implementación de las diferentes rutinas para el manejo del protocolo HTTP.
- main.h: punto de entrada al firmware, inicializa todos los módulos de hardware.
- test_status.h: define diferentes macros y tipos de datos utilizados para almacenar el estado de los ensayos.

Por otro lado, se utilizó Doxygen para documentar el código fuente [34].

3.2.3. Controlador de sistema

Este es el bloque más importante y su función principal es coordinar la interacción de los módulos restantes. Está basado en una máquina de estados finita (FSM por sus siglas en inglés) que, en función de las variables monitoreadas, actúa sobre las controladas, figura 3.9.

Las funciones del bloque se detallan a continuación:

- Procesar el estado de los pulsadores Testear y Cancelar para iniciar y/o detener la secuencia de caracterización.
- Procesar el estado del pulsador Configurar para leer los umbrales de validación para el transformador en ensayo.
- Procesar los valores leídos de las tensiones y corrientes de los bobinados primario y secundario.
- Procesar el estado del *switch* de la tapa de seguridad.
- Generar el comando para alimentar los bobinados.
- Generar la información para mostrar en el *display* local.
- Enviar el estado de la medición al *buzzer* para generar las secuencias de sonidos adecuadas.
- Enviar o solicitar datos al servidor web.

En la figura 3.10 se muestra un diagrama simplificado del controlador. Los estados fueron enumerados para ayudar en la explicación.

En las siguientes secciones se explica en detalle cada estado y su interacción con los diferentes módulos del sistema. Cabe aclarar que no se abordan detalle tales como los motivos por los cuales falló la comunicación con la impresora o la comunicación Wi-Fi ya que estos son detallados en los capítulos posteriores, solo se explica como los resultados de los diferentes bloques impactan sobre el funcionamiento del controlador principal.

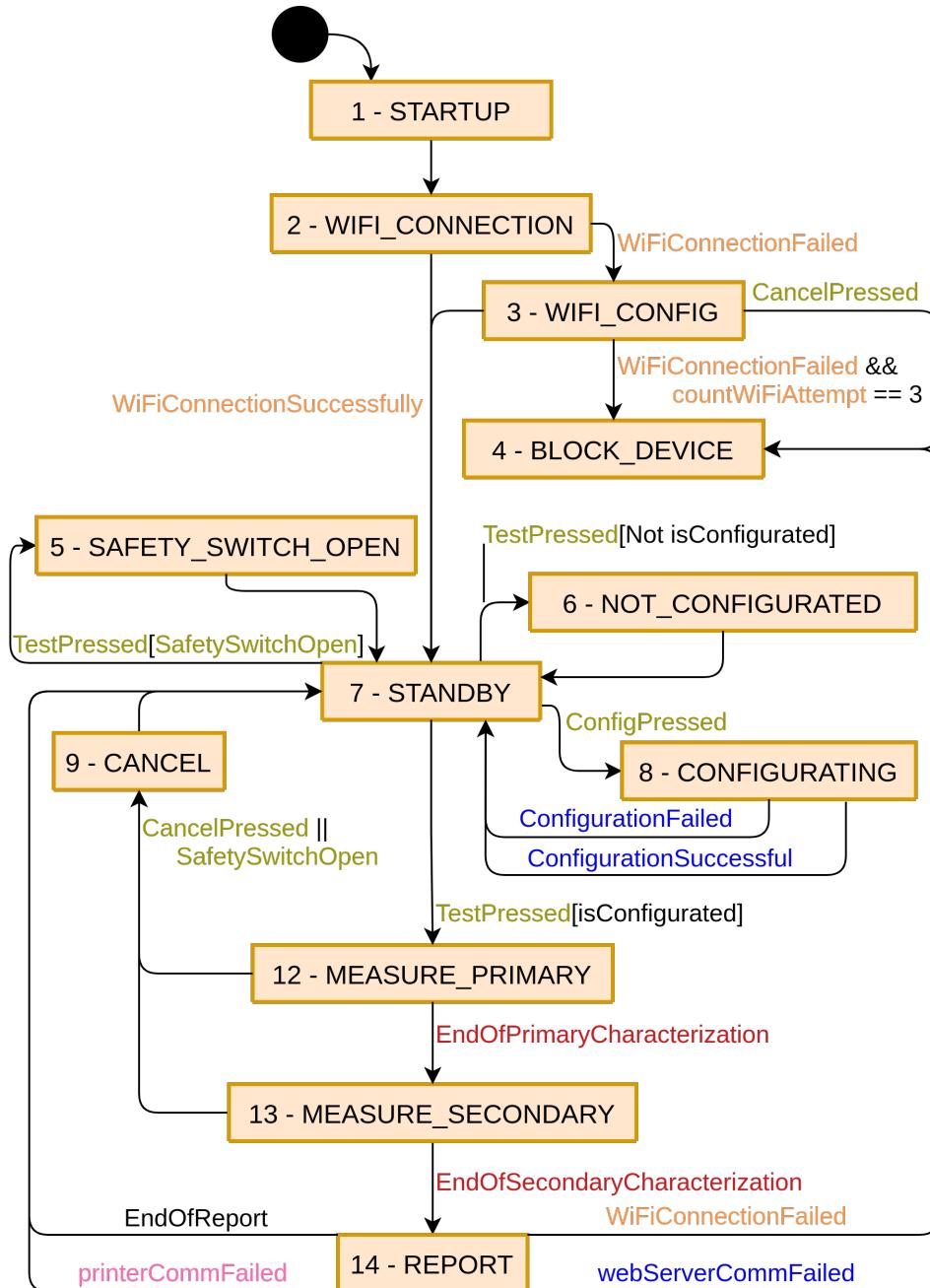


FIGURA 3.10. Diagrama simplificado del controlador del sistema.

Para una mejor explicación, se divide a la FSM en diferentes etapas donde cada una agrupa algunos estados:

- Etapa de inicialización: estados 1, 2 y 3.
- Etapa de configuración: estados 6, 8, 10 y 11, estos últimos no se muestran en la figura 3.10 por claridad.
- Etapa de caracterización: estados 5, 9, 12, 13 y 14.
- Etapa de reporte: estado 14.

Los estados 7 (**STANDBY**) y 4 (**BLOCK_DEVICE**) se consideran estados particulares y aparecen en todas o casi todas las etapas presentadas. Estos estados se

consideran inicio y/o fin de las diferentes etapas.

Etapa de inicialización

En la figura 3.11 se muestra la sub-máquina de estado correspondiente a la etapa de inicialización.

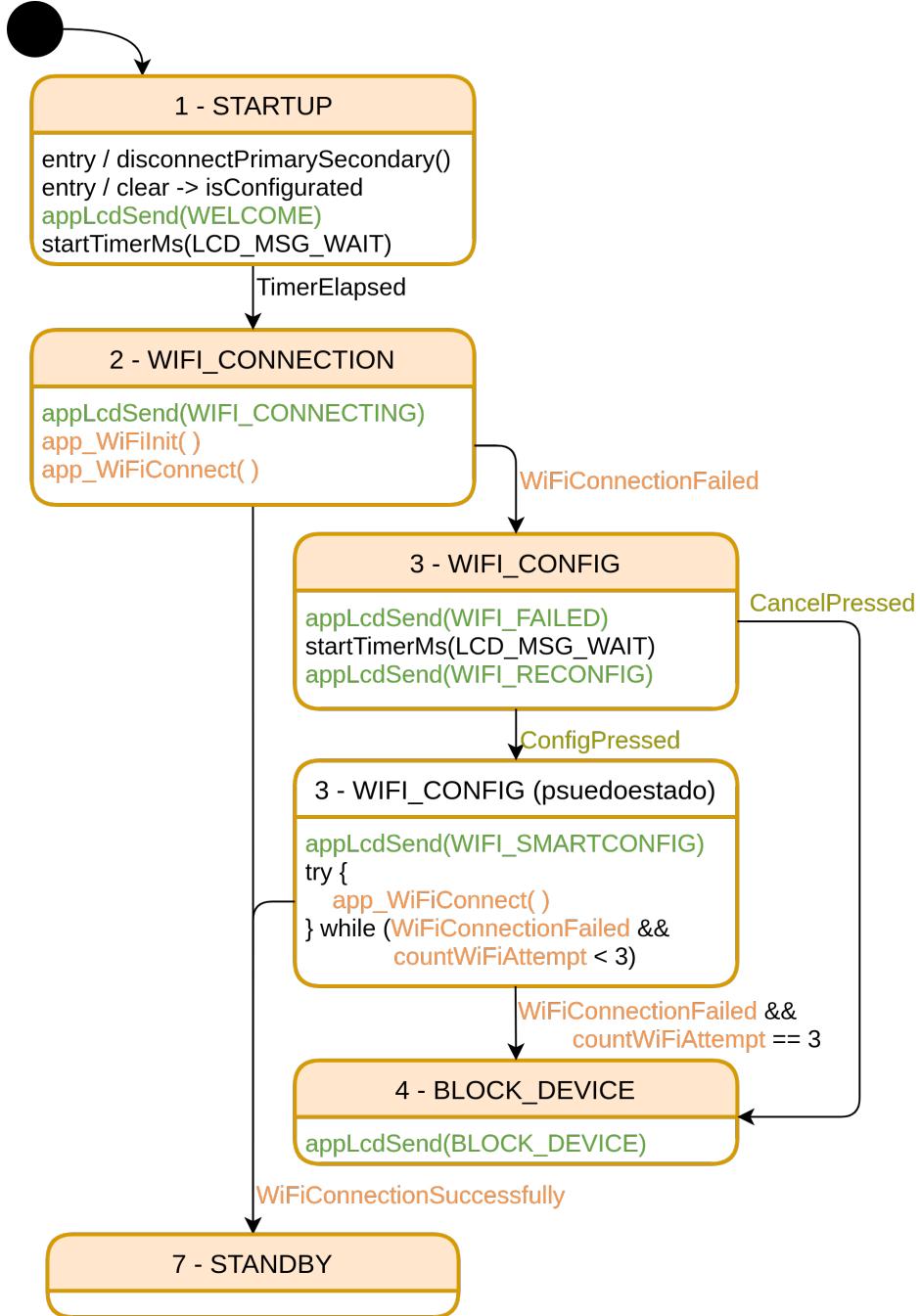


FIGURA 3.11. Etapa de inicialización.

Esta etapa se inicia cuando el equipo es energizado (salida de *reset*) y termina en el estado 7 (*STANDBY*) si el equipo fue exitoso al conectarse a Wi-Fi o en el estado estado 4 (*BLOCK_DEVICE*) si los intentos de conexión fallaron.

Luego de la salida de *reset* del sistema y de inicializar todos los módulos de hardware, se ingresa en el estado 1 (*STARTUP*). Este estado tiene como función inicializar cuestiones de la máquina de estado principal como la alimentación de bobinados, mostrar un mensaje de bienvenida en el *display* durante un tiempo fijo (*LCD_MSG_WAIT*) y limpiar la variable *isConfigurated*. Esta última se utiliza para indicar si el equipo fue configurado, al salir de *reset* el equipo no se encuentra configurado.

Luego de transcurrido el tiempo *LCD_MSG_WAIT*, se transiciona al estado 2 (*WI-FI_CONNECTION*). En este estado, se leen los valores SSID y la contraseña desde la memoria no volátil del dispositivo y se intenta conectar a la red Wi-Fi configurada. En caso de fallar la conexión, se genera el evento *WiFiConnectionFailed* y se transiciona hacia el estado 3 (*WIFI_CONFIG*).

En el estado *WIFI_CONFIG* se consulta si se desea reconfigurar las credenciales de Wi-Fi, para lo cual, el operador puede acceder al pulsar Configurar o negarse al pulsar Cancelar. En caso de reconfigurar, se utiliza la aplicación ESP-Touch 2.3.2, se realizan 3 intentos de reconexión y en caso de fallar nuevamente el equipo pasa al estado 4 (*BLOCK_DEVICE*) y finaliza la secuencia de inicio.

En caso de lograr la conexión Wi-Fi, sea en el estado 2 (*WIFI_CONNECTION*) o el psuedoestado 3 (*WIFI_CONFIG*), el sistema pasa al estado 7 (*STANDBY*) y finaliza la secuencia de inicio.

Etapa de configuración

En la figura 3.12 se muestra la sub-máquina de estado correspondiente a la etapa de configuración.

Luego de la etapa de inicialización se ingresa en la etapa de configuración. Esta etapa se inicia y finaliza en el estado 7 (*STANDBY*) y tiene dos funciones principales:

- Validar si el equipo fue configurado, por medio de *isConfigurated*, para permitir pasar a la etapa de caracterización.
- Pedir los datos de configuración al servidor web al pulsar Configurar y fijar el valor de *isConfigurated* acorde al resultado de dicha acción.

En el estado 7 (*STANDBY*) se muestra en el *display* si el equipo fue o no configurado previamente en base al valor de *isConfigurated*. Si el operario desea testear un transformador, para lo cual pulsa Testear, pero la variable *isConfigurated* es cero, el equipo pasa al estado 6 (*NOT_CONFIGURATED*), muestra un mensaje de equipo no configurado y vuelve al estado *STANDBY*.

Si el operario desea configurar el equipo debe pulsar Configurar, para la cual se pasa del estado *STANDBY* al estado 8 (*CONFIGURATING*). En este estado, se solicitan los valores de configuración al servidor web desarrollado por el cliente. De aquí, se pueden generar dos eventos posibles *ConfigurationSuccessful* o *ConfigurationFailed* que dependen de la obtención de los datos con o sin errores desde el servidor web. En ambos casos, se asigna el valor correcto a la variable *isConfigurated* y se muestra en el *display* el resultado.

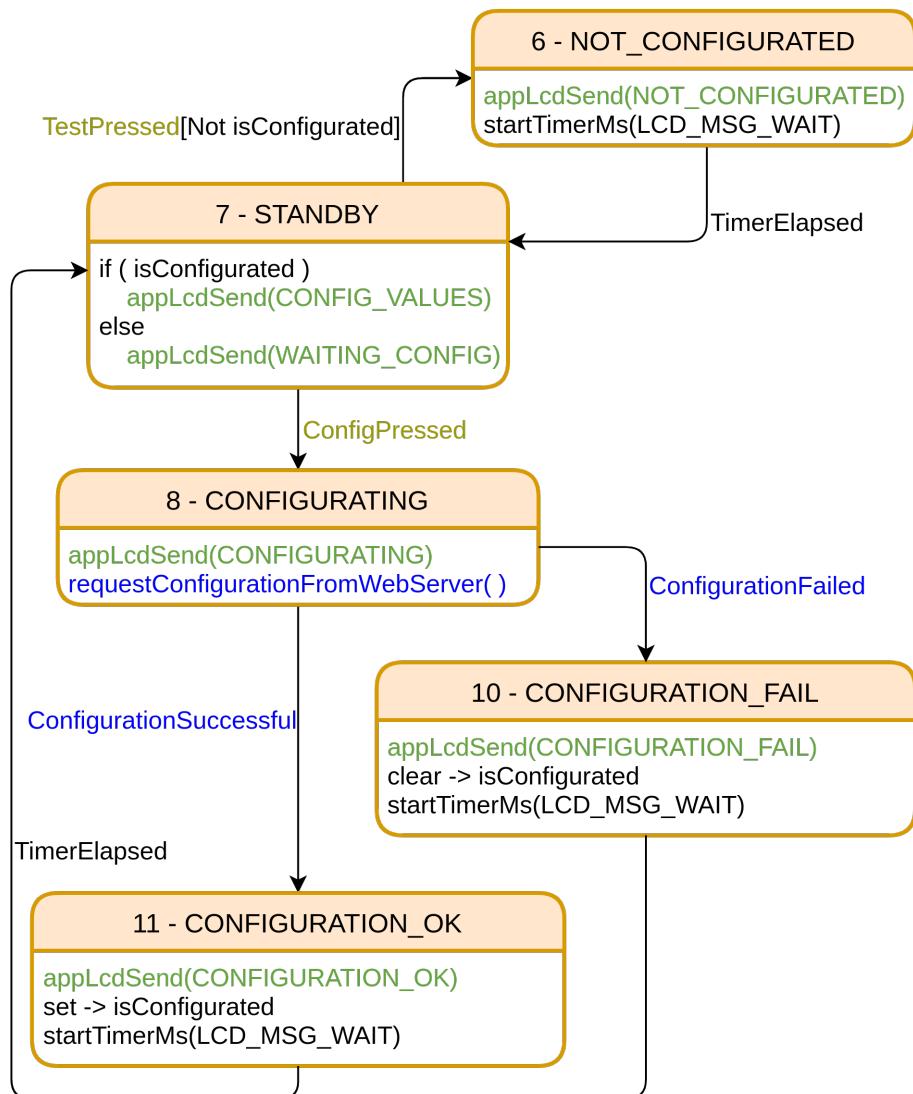


FIGURA 3.12. Etapa de configuración.

Etapa de caracterización

En la figura 3.13 se muestra la sub-máquina de estado correspondiente a la etapa de caracterización.

Luego de la etapa de configuración, se ingresa en la etapa de caracterización. Esta etapa se inicia y finaliza en el estado 7 (STANDBY) y tiene varias funciones:

- Iniciar y procesar los valores leídos de las tensiones y corrientes de los bobinados primario y secundario.
- Alimentar los bobinados.
- Determinar los resultados del ensayo para ser reportados en la etapa de reporte.
- Monitorear en todo momento el *switch* de la tapa de seguridad y evitar o cancelar el proceso de caracterización.

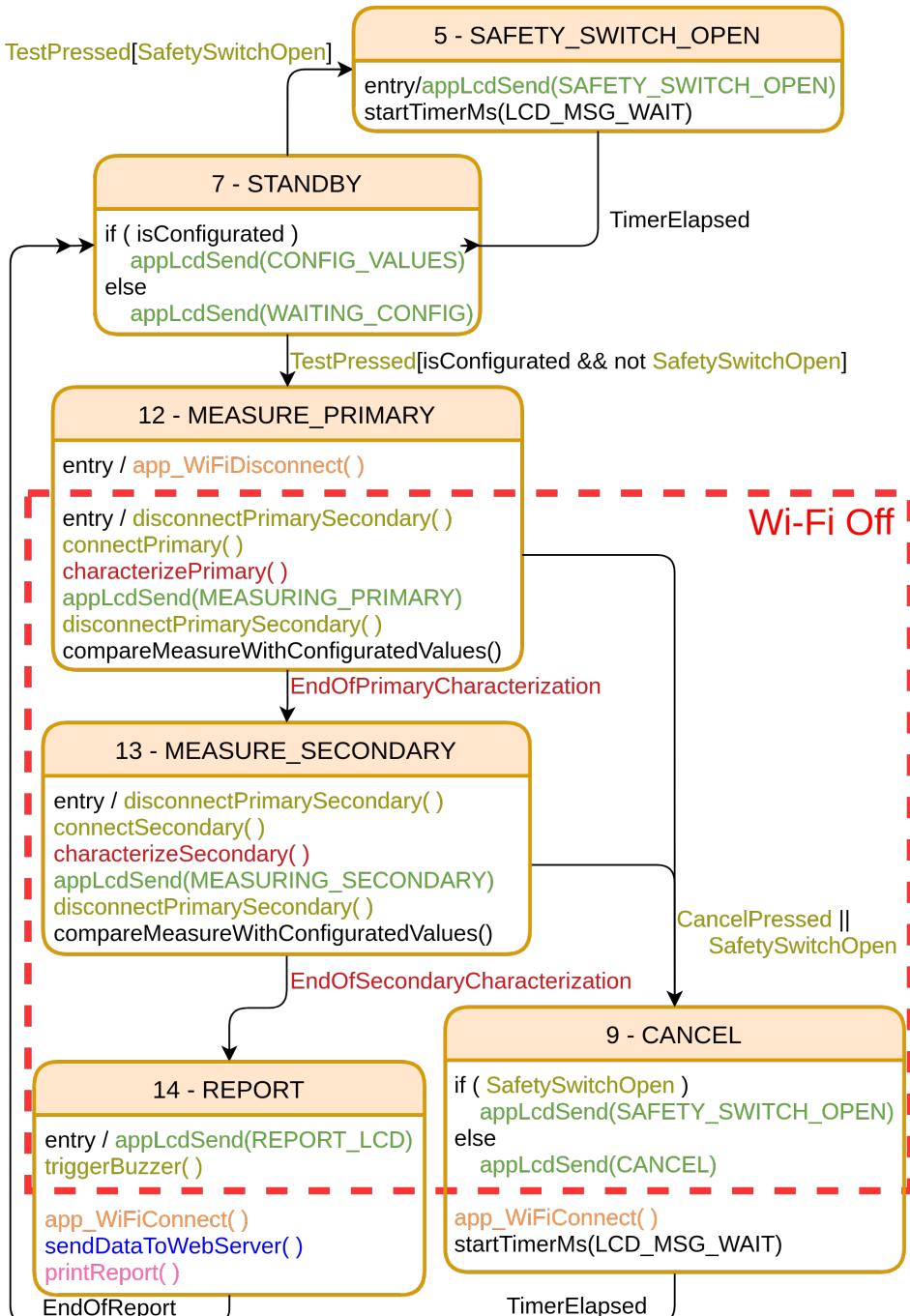


FIGURA 3.13. Etapa de caracterización.

Al igual que en la etapa de configuración, en el estado 7 (STANDBY) se muestra en el *display* si el equipo fue o no configurado previamente. Se supone que para iniciar la etapa de caracterización el equipo fue configurado, de lo contrario, no se podrá avanzar en la caracterización (estado NOT_CONFIGURATED).

Existen dos caminos posibles en la etapa de caracterización para salir del estado STANDBY, ambos son por medio del pulsador Testear, y la diferencia radica en el estado del *switch* de la tapa de seguridad (*SafetySwitchOpen*). Para poder inicializar la caracterización el *switch* debe estar cerrado, en caso contrario el equipo pasa al estado 5 (SAFETY_SWITCH_OPEN), muestra un mensaje de tapa de seguridad abierta y vuelve al estado STANDBY.

Por otro lado, si al pulsar Testear el *switch* se encuentra cerrado, la secuencia de caracterización comenzará normalmente. A continuación se pasa a los estados 12 (*MEASURE_PRIMARY*) y 13 (*MEASURE_SECONDARY*). Estos estados fueron simplificados por claridad en la figura. Las principales tareas del estado *MEASURE_PRIMARY* se detallan a continuación en secuencia:

1. Desconectar el equipo de la red Wi-Fi.
2. Desenergizar ambos bobinados. Esto se realiza por seguridad ya que, en principio, deberían estar desenergizados.
3. Energizar el bobinado primario.
4. Llamar a las rutinas de medición de valor eficaz para obtener los valores de:
 - Tensión en bobinado primario.
 - Corriente que circula por el bobinado primario.
 - Tensión en bobinado secundario.
5. Mostrar los valores medidos. Esto es opcional, se puede elegir al inicio del sistema.
6. Desenergizar ambos bobinados.
7. Comparar los valores medidos con los umbrales configurados previamente y generar el resultado de las comparaciones.

Al finalizar el estado *MEASURE_PRIMARY* se pasa al estado *MEASURE_SECONDARY*. Las principales funciones de este estado se listan a continuación en secuencia:

1. Desenergizar ambos bobinados. Esto se realiza por seguridad ya que, en principio, deberían estar desenergizados.
2. Energizar el bobinado secundario.
3. Llamar a las rutinas de medición de valor eficaz para obtener los valores de:
 - Tensión en bobinado primario.
 - Tensión en bobinado secundario.
 - Corriente que circula por el bobinado secundario.
4. Mostrar los valores medidos. Esto es opcional, se puede elegir al inicio del sistema.
5. Desenergizar ambos bobinados.
6. Comparar los valores medidos con los umbrales configurados previamente y generar el resultado de las comparaciones.

Al finalizar los estados *MEASURE_PRIMARY* y *MEASURE_SECONDARY*, se cuenta con el resultado de la caracterización, solo resta la etapa de reporte cuyo estado principal es el estado 14 (*REPORT*).

Como se puede observar en la figura 3.13 y en los pasos descritos, la comunicación Wi-Fi se detiene en el momento de realizar la medición de los valores eficaces. De los ensayos realizados con el equipo, se pudo notar que, al medir con la comunicación Wi-Fi encendida, las mediciones no resultaban estables e inclusive se observaba un desplazamiento en ellas. Esto es un efecto que fue anticipado

en el análisis de riesgo realizado en el plan de proyecto. Este inconveniente, introducido por el periférico Wi-Fi en las mediciones, fue subsanado al apagar el periférico mientras se mide y encenderlo al finalizar esta tarea.

Por último, el proceso de caracterización puede ser cancelado por dos motivos:

- Pulsar el pulsador Cancelar (*CancelPressed*).
- Si la tapa de seguridad se abre (*SafetySwitchOpen*).

Al cancelar la caracterización se pasa al estado 9 (CANCEL) y finalmente al estado STANDBY.

Etapa de reporte

En la figura 3.14 se muestra la sub-máquina de estado correspondiente a la etapa de reporte.

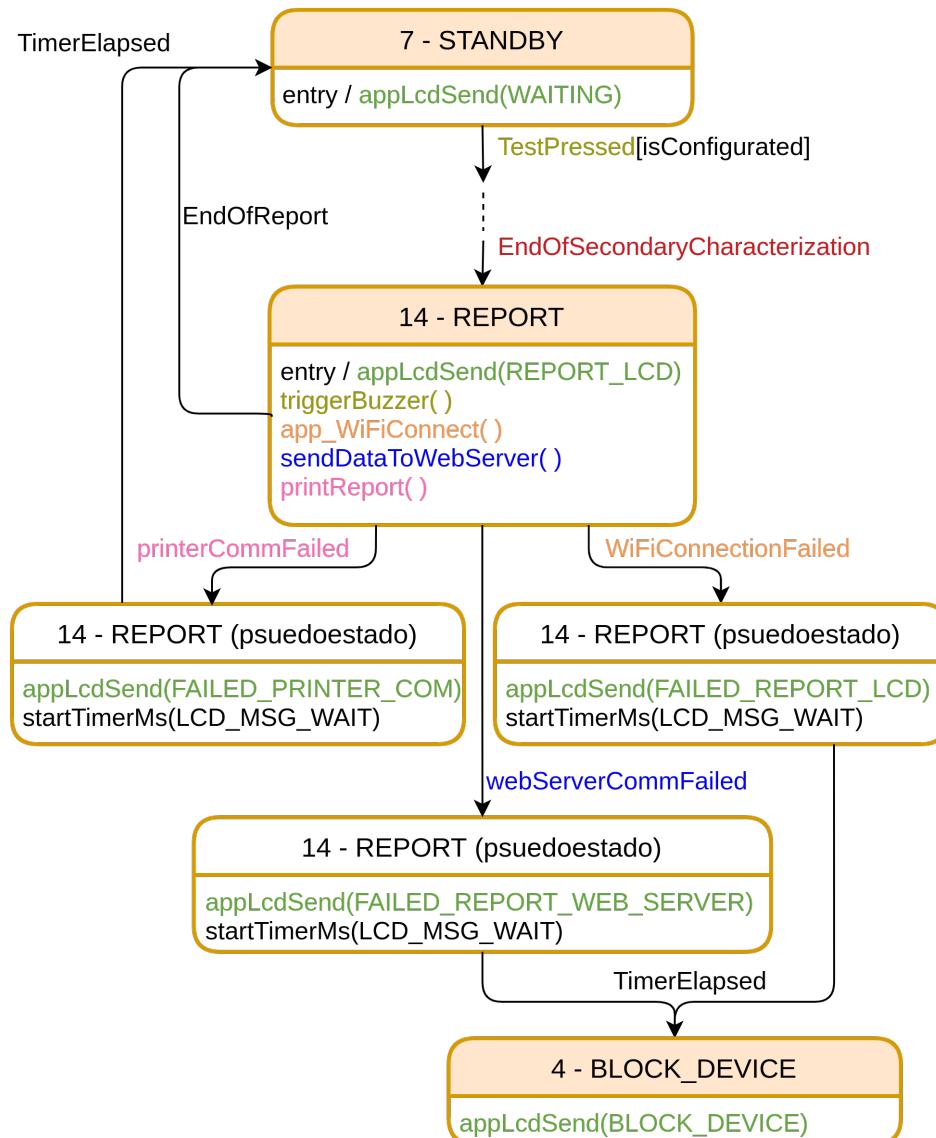


FIGURA 3.14. Etapa de reporte.

Al finalizar los estados *MEASURE_PRIMARY* y *MEASURE_SECONDARY* se pasa al estado 14 (*REPORT*), el cual constituye la etapa de reporte.

Dentro de la etapa de reporte se realizan las siguientes acciones:

- Mostrar el resultado del ensayo en el *display*.
- Accionar el *buzzer* adecuadamente según el resultado.
- Reconectar a la red Wi-Fi.
- Enviar los resultados y los valores medidos al servidor web.
- Imprimir la etiqueta con los resultados y los valores medidos.

En el caso que alguno de los pasos anteriores fallen, se transiciona a un pseudo-estado dentro del estado *REPORT* y se muestra la falla en el *display*. En caso de ser una falla grave, que impide la comunicación de resultados al servidor web, se transiciona al estado *BLOCK_DEVICE*.

3.2.4. Medición de valor eficaz

Los monitores de tensión y corriente alterna del bobinado primario y secundario son los bloques encargados de entregarle al controlador principal los valores RMS de dichas variables. Para tal fin, las salidas de los sensores de tensión y corriente presentados en las secciones 2.4 y 2.5, luego de pasar por un divisor resistivo acorde y un filtro pasa bajos (RC), son conectadas a cuatro canales del ADC1 del módulo ESP32-WROOM-32 como se muestra en la tabla 3.1.

Las rutinas encargadas de leer los valores de los canales del ADC1 y retornar dichos valores eficaces se encuentran en el archivo adc.h. En el código 3.1 se muestra el archivo de cabecera simplificado.

```

1 /**
2 * @brief Initialize ADC
3 *
4 */
5 void appAdcInit(void);
6
7 /**
8 * @brief Start a new RMS ADC conversion
9 *
10 * @param rms
11 *
12 * @note This function takes about 1.6 seg in processing the four ADC
13 *       channels
14 *       It reads AMOUNT_OF_CYCLES cycles sampling SAMPLES_IN_20MS in 20
15 *       ms for each channel
16 */
17 void appAdcStart(rms_t *rms);

```

CÓDIGO 3.1. Pseudocódigo del módulo adc.h.

En el archivo se definen dos funciones:

- *appAdcInit* que se utiliza para inicializar todo el hardware necesario para el uso del conversor analógico-digital.

- appAdcStart que es llamada desde el controlador principal cada vez que se desea iniciar una conversión. Esta función acepta un puntero a la estructura rms_t para devolver los valores leídos.

Un detalle, a destacar en el código mostrado, es que las funciones presentadas se documentaron por medio de Doxygen, esto es algo que se puede ver en los diferentes archivos de cabecera que se muestran a lo largo de la memoria.

En el pseudocódigo mostrado en el código 3.2 se muestra la implementación de la función appAdcStart. Esta función utiliza los siguientes módulos del entorno de desarrollo ESP-IDF presentados en la sección 2.3.1:

- Conversor analógico-digital [35].
- *Inter-IC Sound (I2S)*: que configurado en el modo ADC/DAC proporciona un periférico DMA para ser usado con el ADC y/o DAC [36].

El código presentado está simplificado para mostrar su operación, pero cabe destacar que, además de lo mostrado en el código 3.2, se utilizaron diferentes herramientas de FreeRTOS en la la función appAdcStart. Las colas para sincronizar el funcionamiento del DMA con el resto de la función son un ejemplo de ello.

```

1 static adc_t adc[ADC_CHANNELS];
2
3 void appAdcStart(rms_t *rms) {
4     int32_t s_rms;
5
6     // Habilitación del DMA
7     i2s_adc_enable(I2S_NUM_0);
8
9     // Barrido de los 4 canales del ADC1
10    for (adcIndex=0; adcIndex<ADC_CHANNELS-1; adcIndex++) {
11        // Lectura del canal propiamente dicha
12        i2s_read(I2S_NUM_0, buffer);
13
14        // Filtro digital de primer orden
15        firstOrderFilter(buffer);
16
17        // Cálculo de valor eficaz
18        adc[adcIndex].rms = getRMS(buffer);
19
20        // Calibración de la entrada
21        calibration(&adc[adcIndex].rms, adc[adcIndex].offset, adc[adcIndex].gain);
22
23        // Cambiar el canal a medir del ADC1
24        i2s_set_adc_mode(ADC_UNIT_1, adc[adcIndex].channel);
25    }
26
27    // Deshabilitar el DMA
28    i2s_adc_disable(I2S_NUM_0);
29 }
30 }
```

CÓDIGO 3.2. Pseudocódigo del módulo adc.c.

Al ingresar a la función appAdcStart se habilita el DMA por medio de la función i2s_adc_enable. El próximo paso es barrer los cuatro canales del ADC1 para la cual se utiliza un lazo de tipo *for*. Dentro del lazo, la función i2s_read muestrea el canal del ADC1 seleccionado. Dicha función fue configurada para leer 8 ciclos de la señal de entrada y tomar 128 muestras por cada ciclo, hasta que se realiza

esta acción la función permanece bloqueada. Como se miden las variables de la red de energía eléctrica, su periodo es de 20 ms, lo que da un tiempo de medición por canal de 160 ms, ecuación 3.1.

$$T_{MED} = 8 * 20ms = 160ms \quad (3.1)$$

Y se obtienen 1024 muestras por canal, ecuación 3.2.

$$N = 8 * 128 = 1024 \quad (3.2)$$

Por lo tanto, la función i2s_read permanece bloqueada durante 160 ms y devuelve un puntero a un vector con 1024 puntos de la señal de entrada. Este vector es luego filtrado con un filtro digital de primer orden cuya estructura se muestra en la ecuación 3.3. Los valores de A_1 y B_0 se ajustaron convenientemente en función de la frecuencia de muestreo del sistema y la respuesta deseada del filtro.

$$y(n) = A_1 y(n-1) + B_0 x(n) \quad (3.3)$$

Luego del filtrado, se procede a calcular el valor eficaz de los valores medidos por medio de la función getRMS. Se utilizó la ecuación 3.4 para el cálculo del valor eficaz de las muestras obtenidas, donde $CICLOS=8$ y $N_{CICLOS}=128$. Con el filtro digital y el promedio de 8 ciclos de la señal medida se obtiene un valor estable y ayuda a lidiar con las conexiones de la placa universal y las salidas de los módulos sensores.

$$rms = \frac{1}{CICLOS} \sum_{n=0}^{CICLOS-1} \sqrt{\frac{1}{N_{CICLOS}} \sum_{i=0}^{N_{CICLOS}-1} buffer[n][i] * buffer[n][i]} \quad (3.4)$$

Finalmente, se realiza una corrección lineal del valor obtenido, función calibration en el código, a partir de un desplazamiento (*offset*) y una ganancia (*gain*) preconfigurados. Se puede observar que cada uno de los pasos mencionados se realiza sobre cada canal independientemente, puntualmente la calibración de las señales, para la cual se cuenta con un juego de ganancias y desplazamientos independientes para cada canal.

3.2.5. Comunicación Wi-Fi

El módulo de comunicación Wi-Fi es el módulo más complejo del sistema. Este, no solo debe ser capaz de controlar la configuración, conexión y desconexión del módulo a la red Wi-Fi, sino que, además, debe leer y escribir a memoria no volátil las credenciales de la red Wi-Fi y manejar el protocolo SmartConfig.

En la figura 3.15 se muestra un diagrama de flujo simplificado de las etapas seguidas para la gestión de la red Wi-Fi.

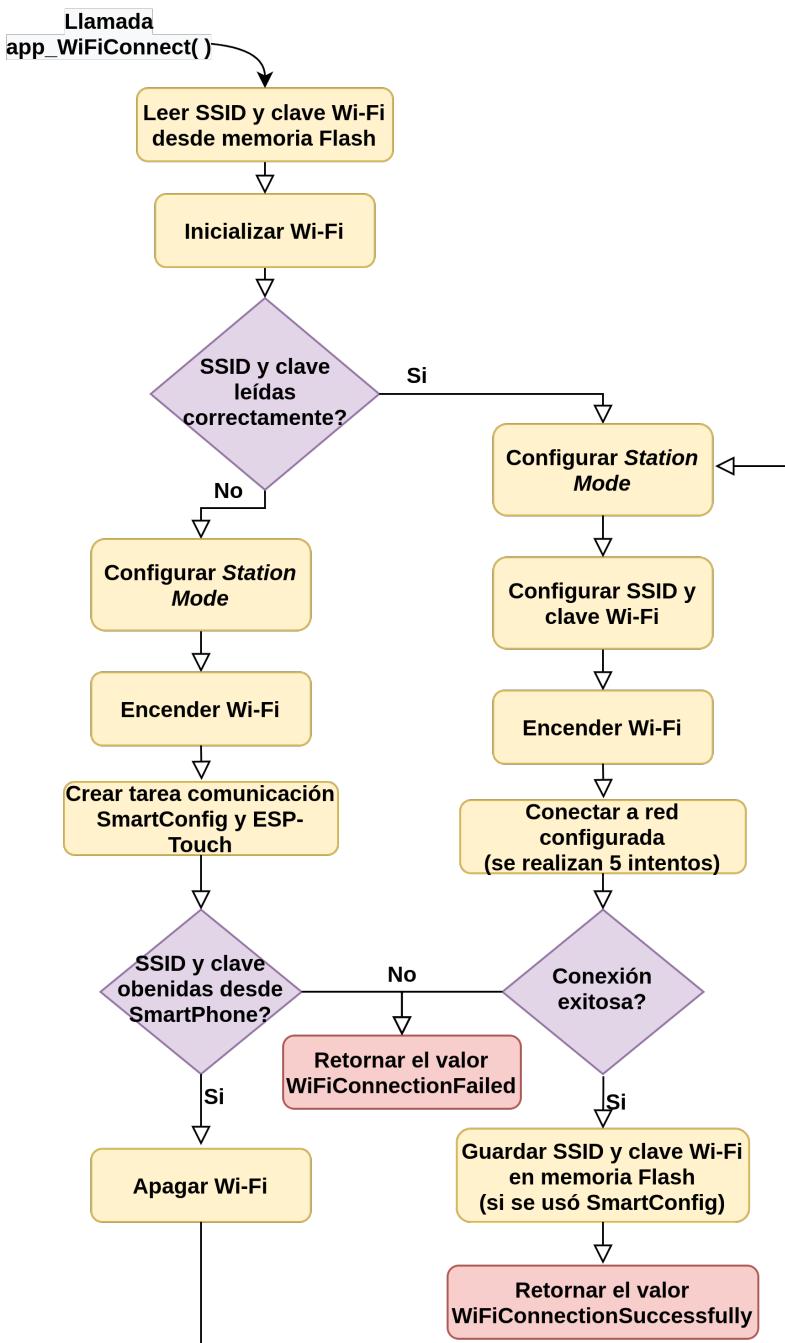


FIGURA 3.15. Secuencia de conexión a red Wi-Fi.

Las principales librerías del entorno ESP-IDF utilizadas para el desarrollo de este módulo son:

- *Non-volatile storage library* [37]: utilizada para leer y escribir las credenciales de Wi-Fi hacia y desde la memoria *flash*. Se tuvo el cuidado de validar los datos leídos para evitar errores en las librerías de Wi-Fi por leer datos corruptos desde la memoria.
- *Wi-Fi* [14].
- *SmartConfig*: utilizado en conjunto con la aplicación móvil ESP-Touch como se explica en la sección 2.3.2.

Para sincronizar las diferentes tareas se utilizó la librería de eventos *Event Loop Library* [38], la cual permite pasar por los diferentes estados de la comunicación y evitar condiciones de carrera. Por simplicidad, en el diagrama de flujo de la figura 3.15 se omiten estos eventos. Sin embargo, en la descripción posterior se nombrarán alguno de ellos para tener una mejor correlación con el trabajo realizado.

Para iniciar la conexión a la red Wi-Fi el controlador principal debe llamar a la función app_WiFiConnect como se explicó en la sección 3.2.3. Una vez llamada dicha función ocurre la secuencia de la figura 3.15. El primer paso es leer la memoria no volátil en busca de las credenciales de Wi-Fi, en caso de obtenerlas sin error, se procede a configurar el periférico en modo estación (*station mode*), configurar las credenciales en una estructura de configuración provista por la librería Wi-Fi y encender el periférico. El próximo paso es esperar por el evento WIFI_CONNECTED o el evento WIFI_FAIL. En caso de recibir este último, se reintenta 5 veces para poder conectar, si no se logra se retorna de la función app_WiFiConnect con el valor *WiFiConnectionFailed*. En caso de obtener el evento WIFI_CONNECTED en alguno de los intentos, la función retorna con el valor *WiFiConnectionSuccessfully*.

En caso de no tener las credenciales de Wi-Fi configuradas u obtener un error al intentar leerlas, la función app_WiFiConnect puede utilizar SmartConfig para intentar adquirirlas desde la aplicación móvil ESP-Touch. Para esto, se crea una tarea de FreeRTOS para utilizar las funciones de la librería de SmartConfig. Dentro de la tarea se espera por el evento ESPTOUCH_DONE que se genera cuando la aplicación ESP-Touch envía las credenciales. Luego de obtener las credenciales, se procede a intentar conectar nuevamente por medio del procedimiento explicado en el párrafo anterior. La única diferencia en este caso, es que si se logra conectar exitosamente a Wi-Fi, se guardan las nuevas credenciales en memoria no volátil para su posterior reutilización. Por otro lado, la tarea de FreeRTOS creada para administrar el protocolo SmartConfig se borra del sistema al finalizar el procedimiento ya que la acción de generar las credenciales de Wi-Fi es una tarea de un solo uso y carece de sentido tenerla corriendo todo el tiempo.

3.2.6. Obtención y envío de datos al webserver

Como se explicó en los capítulos introductorios, el cliente desarrolló un servidor web donde consultar los valores de comparación de tensiones y corrientes y a donde enviar los valores medidos y los resultados obtenidos.

La forma de comunicarse con el servidor web es a través de comandos GET y POST de HTTP. Por otro lado, la información a ser transmitida y/o recibida se procesa en el cuerpo de los comandos GET y POST en formato JSON.

Para recibir los datos de configuración se debe enviar el siguiente comando GET:

```
GET /TransformersTesterConfigs/Last http/1.1
Host: https://iris-test-api.azurewebsites.net/api
Content-Type: application/json
```

Y se obtiene una respuesta como la siguiente:

HTTP/1.1 200 OK
Content-Type: application/json
Server: Kestrel
{
"id":4,
"createDate":"2021-03-10T15:59:04",
"code":"MPELETRAN-0023",
"batchId":"20210310-1",
"vinPrimaryMin":225.00,
"voutSecondaryMin":14.00,
"iPrimaryMin":15.00,
"vinSecondaryMin":14.00,
"voutPrimaryMin":210.00,
"iSecondaryMin":100.00,
"vinPrimaryMax":235.00,
"voutSecondaryMax":17.00,
"iPrimaryMax":70.00,
"vinSecondaryMax":16.00,
"voutPrimaryMax":230.00,
"iSecondaryMax":400.00
}

A la comunicación con el servidor web y el procesamiento de los paquetes se lo dividió en dos capas de software:

- Capa HTTP (`http_client.h`): encargada de procesar los comandos GET y POST de HTTP.
- Capa de procesamiento de paquetes (`app_Comm.h`): encargada de tomar los datos enviados o recibidos de la capa HTTP y armar o desarmar (“parsear”) los paquetes JSON.

Para el desarrollo de la capa HTTP se utilizó la librería cliente HTTP del entorno de desarrollo ESP-IDF [39]. En el código 3.3 se muestra un pseudocódigo para la función GET, la implementación de la función POST es similar.

```

1 #include "esp_http_client.h"
2
3 #define GET_URL    "https://iris-test-api.azurewebsites.net/api/
4                         TransformersTesterConfigs/Last"
5
6 esp_err_t get_http_config(char *buffer)
7 {
8     esp_http_client_config_t config = {
9         .url = GET_URL,
10    };
11
12    // Inicializar el cliente HTTP
13    esp_http_client_init(&config);
14
15    // Abrir conexión con el servidor
16    if ((err = esp_http_client_open(client, 0)) != ESP_OK) {
17        return (err);
18    }
19
20    // Armar encabezado método GET
21    esp_http_client_set_method(client, HTTP_METHOD_GET);

```

```

21     esp_http_client_set_header(client, "Content-Type", "application/json");
22
23     // Enviar paquete GET y capturar respuesta
24     read_len = esp_http_client_read(client, buffer, BUFFER_SIZE);
25     if (read_len <= 0) {
26         return (ESP_FAIL);
27     }
28
29     // Cerrar la conexión con el cliente HTTP
30     esp_http_client_close(client);
31
32     return (err);
33 }
```

CÓDIGO 3.3. Pseudocódigo función GET de HTTP.

En el caso de un comando GET, los datos recibidos desde la capa HTTP son luego pasados a la capa de procesamiento de paquetes donde se procede a “parsearlos” y validarlos. Entre las rutinas de validación, se encuentra el chequeo de los datos numéricos. En la capa de procesamiento, los datos se guardan en una estructura de tipo configData_t diseñada para tal fin. La estructura configData_t permite el fácil acceso de los datos obtenidos del servidor web por parte del controlador principal. En el código 3.4 se muestra la jerarquía de estructuras desarrollada para la estructura configData_t, aquí también se puede apreciar que está documentada con Doxygen.

```

1 /**
2 * @brief Parameter thresholds structure
3 *
4 */
5 typedef struct {
6     int32_t max;
7     int32_t min;
8 } parametersRange_t;
9
10 /**
11 * @brief Measured Parameters structure
12 *
13 */
14 typedef struct {
15     parametersRange_t Vinp;    /*!<Primary Voltage      */
16     parametersRange_t Voutp;   /*!<Primary Voltage      */
17     parametersRange_t Vins;    /*!<Secondary Voltage    */
18     parametersRange_t Vouts;   /*!<Secondary Voltage    */
19     parametersRange_t Ip;     /*!<Primary current      */
20     parametersRange_t Is;     /*!<Secondary current    */
21 } trafoParameters_t;
22
23 /**
24 * @brief Configuration data type
25 *
26 */
27 typedef struct {
28     uint32_t id;                /*!<Test Identification Number */
29     char batchId[BATCHID_LENGTH]; /*!<Transformer BatchId   */
30     char code[CODE_LENGTH];      /*!<Transformer code        */
31     trafoParameters_t trafoParameters; /*!<Measured Parameters structure*/
32 } configData_t;
```

CÓDIGO 3.4. Estructuras para el manejo de los datos recibidos.

3.2.7. Implementación del protocolo DPL

Entre las tareas que el controlador principal debe realizar en la etapa de reporte (sección 3.2.3) se encuentra la impresión de una etiqueta con los resultados del ensayo. Para ello, se utilizó la impresora presentada en la sección 2.6 cuyo protocolo se conoce como DPL y fue introducido en la misma sección. En esta sección se explica la implementación de dicho protocolo.

El protocolo DPL fue implementado en el módulo app_printer.c, en tanto que, la capa física del protocolo fue resuelta con el módulo de hardware presentado en la sección 2.11. Para su manejo se utilizó la librería UART que se encuentra dentro de los controladores de periféricos provisto por el entorno ESP-IDF [13].

Cuando el controlador principal desea imprimir los resultados del ensayo debe llamar a la función *print*. Esta función implementa todos los comandos necesarios para utilizar el protocolo DPL y devuelve el resultado de la impresión. En el código 3.5 se muestran los valores devueltos por la función *print*.

```

1 /**
2  * @brief Printer status
3 *
4 */
5 typedef enum {
6     PRINTER_NO_COMM,      /*!< Printer never respond          */
7     PRINTER_NOT_READY,    /*!< Printer is not ready, answer to
8                           command <SOH>A/r (Status request) */
9     PRINTER_READY,        /*!< Printer ready to print         */
10    PRINTER_OK            /*!< The print command was successful */
11 } printerStatus_t;

```

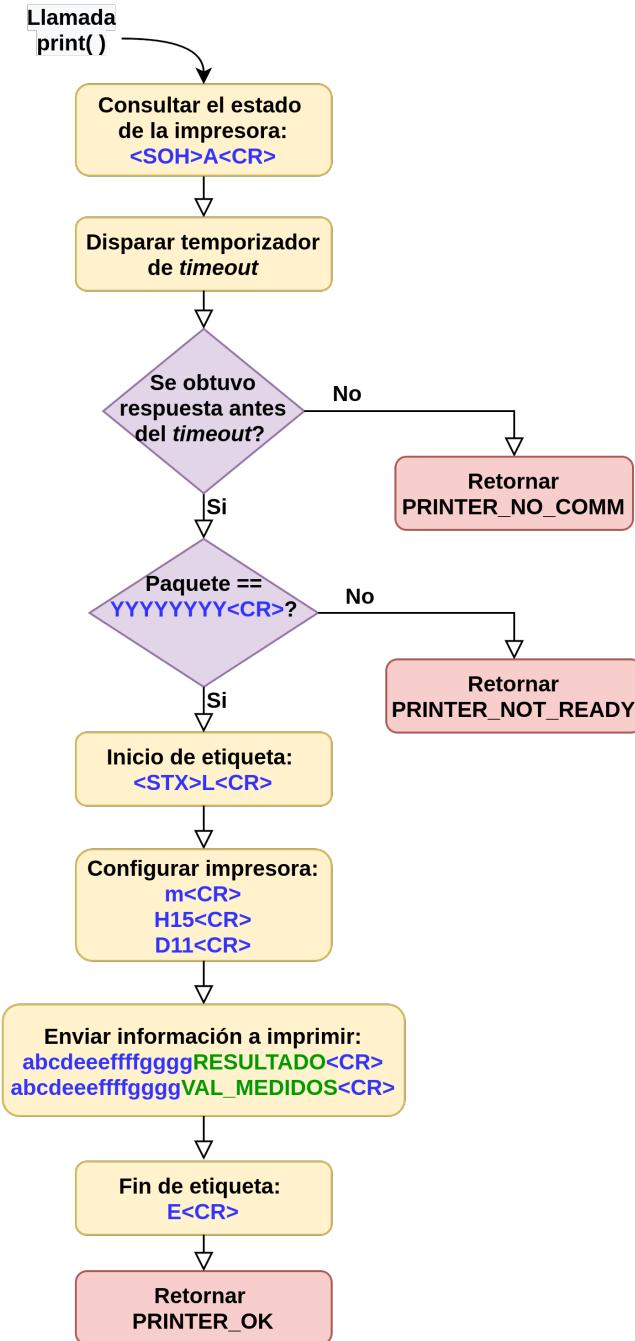
CÓDIGO 3.5. Prototipo función *print*.

En la figura 3.16 se muestra el diagrama de flujo de la función *print*. Los comandos a enviar y la respuesta de la impresora se resaltan en color azul, mientras que el contenido de la etiqueta (resultado obtenido del ensayo y valores medidos) se resalta en verde.

La primera acción a realizar al llamar la función *print* es verificar el estado de la impresora. Para ello, se envía el comando inmediato <SOH>A<CR> y a su vez se dispara un temporizador de espera (*timeout*). Si el temporizador de espera llega a su valor de temporización antes que la impresora haya respondido, la función retorna un error de comunicación con la impresora (PRINTER_NO_COMM). En caso contrario, la obtención una respuesta, esta es analizada y si es igual a la cadena “YYYYYYYY<CR>” significa que la impresora está en condiciones de imprimir. Por otro lado, en caso de recibir algún carácter ‘N’ dentro de la trama anterior, significa que la impresora no está disponible y se retorna de la función *print* con un estado de error PRINTER_NOT_READY.

En caso que la impresora esté en condiciones de imprimir (PRINTER_READY), se pasa a enviar el comando <STX>L<CR> para iniciar la impresión de la etiqueta propiamente dicha. Después de este, se envían los siguientes comandos de configuración propios de la etiqueta en cuestión:

- m<CR>: indica que la etiqueta utiliza unidades internacionales.
- H15<CR>: fija la temperatura del cabezal de impresión.
- D11<CR>: fija el tamaño del punto de impresión.

FIGURA 3.16. Diagrama de flujo función de la *print*.

Una vez configurada la impresora para la etiqueta en particular, se procede a enviar la información a imprimir por medio del formato presentado en la tabla 2.3. En la figura se indican genéricamente los caracteres “`abcdeeffffgggg`” ya que estos dependen de cada línea que se quiera imprimir (rotación del texto, tipo de fuente, ubicación del texto, etc). En la misma línea y luego de las caracteres mencionados, se encuentra el dato a imprimir que depende del ensayo (RESULTADO y VAL_MEDIDOS).

Por último, se envía el comando `E<CR>` que le indica a la impresora que se finalizó la carga de la etiqueta y se puede proceder a su impresión. Si se llega al final de este proceso la función `print` devuelve `PRINTER_OK` que significa que se

pudo imprimir con éxito.

3.2.8. Interfaz de usuario

La interfaz local de usuario está formada por los siguientes elementos:

- Pulsadores Testear, Configurar y Cancelar.
- *Buzzer*.
- *Display* alfanumérico de 20 caracteres por 4 líneas.

Para trabajar con todos estos módulos se utilizó la librerías GPIO que se encuentra dentro de los controladores de periféricos provisto por el entorno ESP-IDF [13].

Para el procesamiento de los pulsadores se desarrolló una función para cada uno. Estas funciones deben ser encuestadas (*pulling*) periódicamente para saber el estado de los pulsadores, en el código 3.6 se muestra el prototipo para el pulsador Cancelar. Adicionalmente, se incorporaron rutinas antirebote (*debouncing*) para todos los pulsadores.

```

1 /**
2  * @brief Check if the cancel button was pressed
3  *
4  * @return true
5  * @return false
6 */
7 bool isCancelPressed( void );

```

CÓDIGO 3.6. Pseudocódigo del módulo adc.c.

En caso del accionamiento del *buzzer* se debieron generar 2 secuencias según el requerimiento 3 de la sección 2.2.5, para lo cual se utilizó un temporizador de hardware. Para la implementación del temporizador se usó la librería de temporizadores provista por el entorno ESP-IDF dentro de los controladores de periféricos [13].

Por último, para el *display* se tomó como base la librería sAPI [40] que posee un módulo de software para utilizar el controlador HD44780. El código debió ser portado para funcionar en el entorno ESP-IDF. Además del trabajo anterior, se trabajó en mejorar la librería de la sAPI por medio del agregado de varias estructuras de software que permiten inicializar el controlador del *display* de una manera más eficiente.

Capítulo 4

Ensayos y resultados

4.1. Pruebas funcionales del hardware

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

Capítulo 5

Conclusiones

5.1. Conclusiones generales

5.2. Próximos pasos

Bibliografía

- [1] Wikipedia. *Transformador*. <https://es.wikipedia.org/wiki/Transformador>. Abr. de 2021. (Visitado 07-05-2021).
- [2] Ingeniería Mecafenix. *¿Qué es un Transformador eléctrico y cómo funciona?* <https://www.ingmecafenix.com/electronica/transformador-electrico/>. Feb. de 2018. (Visitado 07-05-2021).
- [3] Ing. Patricio Concha Fuentes. *Tipos y aplicaciones de transformadores*. http://patricioconcha.ubb.cl/transformadores/gral_tipos_y_aplicaciones.htm. Jun. de 2003. (Visitado 07-05-2021).
- [4] *Ensayo en vacío de un transformador*.
<http://instalacioneselectricasparatodos.blogspot.com/2016/12/ensayo-en-vacio.html>. Dic. de 2016. (Visitado 07-05-2021).
- [5] *Ensayo de cortocircuito de un transformador*.
<http://instalacioneselectricasparatodos.blogspot.com/2016/12/ensayo-de-cortocircuito-de-un.html>. Dic. de 2016. (Visitado 07-05-2021).
- [6] Anthony Alvarez. *Pruebas de aislamiento y polaridad en un transformador*.
<http://pruebasentransformadores.blogspot.com/2015/06/pruebas-de-aislamientos-de-un.html>. Jun. de 2015. (Visitado 07-05-2021).
- [7] Megger. *MVCT - Equipo de prueba de transformador de corriente y tensión*. Disponible: 2021-05-07. URL: <https://es.megger.com/products/medicion-transformadores/medicion-de-transformadores-de-corriente/mvct>.
- [8] ALTANOVA GROUP. *ICT1 Equipo para pruebas de transformadores de corriente y tensión*. Disponible: 2021-05-07. URL: <https://www.altanova-group.com/es/products/off-line-tests/ict1>.
- [9] Espressif Systems (Shanghai) Co., Ltd. *ESP32-DevKitC V4 Getting Started Guide*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html>.
- [10] Espressif Systems (Shanghai) Co., Ltd. *ESP32WROOM32E - ESP32WROOM32UE Datasheet*. Disponible: 2021-05-07. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.
- [11] Espressif Systems (Shanghai) Co., Ltd. *Espressif IoT Development Framework*. Disponible: 2021-05-07. URL: <https://www.espressif.com/en/products/sdks/esp-idf#:~:text=ESP%2DIDF%20is%20Espressif's%20official,as%20C%20and%20C%2B%2B>.
- [12] Espressif Systems (Shanghai) Co., Ltd. *FreeRTOS*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html>.
- [13] Espressif Systems (Shanghai) Co., Ltd. *Peripherals API*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/index.html>.

- [14] Espressif Systems (Shanghai) Co., Ltd. *Wi-Fi*. Disponible: 2021-05-07. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html.
- [15] Espressif Systems (Shanghai) Co., Ltd. *Bluetooth API*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/bluetooth/index.html>.
- [16] Espressif Systems (Shanghai) Co., Ltd. *Application Protocols*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/index.html>.
- [17] Eclipse Foundation. *Eclipse*. Disponible: 2021-05-07. URL: <https://www.eclipse.org/>.
- [18] Microsoft Corporation. *Visual Studio Code*. Disponible: 2021-05-07. URL: <https://code.visualstudio.com/>.
- [19] CMake. *CMake*. Disponible: 2021-05-07. URL: <https://cmake.org/>.
- [20] Espressif Systems (Shanghai) Co., Ltd. *Build System (CMake)*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/v3.3/api-guides/build-system-cmake.html>.
- [21] Espressif Systems (Shanghai) Co., Ltd. *ESP-Touch*. Disponible: 2021-05-07. URL: <https://www.espressif.com/en/products/software/esp-touch/overview>.
- [22] Espressif Systems (Shanghai) Co., Ltd. *SmartConfig*. Disponible: 2021-05-07. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_smartconfig.html.
- [23] Qingxian Zeming Langxi Electronic. *ZMPT101B precise voltage transformer*. Disponible: 2021-05-07. URL: <http://www.zeming-e.com/English/prodviewtype5-298.html>.
- [24] Texas Instruments. *LMx58-N Low-Power, Dual-Operational Amplifiers*. Disponible: 2021-05-07. URL: <https://www.ti.com/lit/ds/symlink/lm158-n.pdf>.
- [25] Qingxian Zeming Langxi Electronic. *ZMCT103C series current transformer*. Disponible: 2021-05-07. URL: <http://www.zeming-e.com/English/prodviewtype5-280.html>.
- [26] Honeywell. *E-Class Mark III Desktop Barcode Printer*. Disponible: 2021-05-07. URL: <https://www.honeywellaic.com/en-ae/products/printers/desktop/e-class-mark-iii>.
- [27] Datamax-O'Neil. *Class Series II - Programmer's Manual*. Disponible: 2021-05-07. URL: https://www.honeywellaic.com/en-ae/-/media/en/files-public/technical-publications/printers/1common/cl2_88-2341-01_l.pdf.
- [28] Texas Instruments. *MAX3232 3-V to 5.5-V Multichannel RS-232 Line Driver/Receiver*. Disponible: 2021-05-07. URL: https://www.ti.com/lit/ds/symlink/max3232.pdf?ts=1619994386617&ref_url=https%253A%252F%252Fwww.google.com%252F.
- [29] HITACHI. *HD44780U Datasheet*. Disponible: 2021-05-07. URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/63673/HITACHI/HD44780/+435JWUEGSzDpKdlpzC.hv+/datasheet.pdf>.
- [30] Texas Instruments. *TL431 / TL432 Precision Programmable Reference*. Disponible: 2021-05-07. URL: https://www.ti.com/lit/ds/symlink/tl431.pdf?ts=1620723088306&ref_url=https%253A%252F%252Fwww.google.com%252F.

- [31] Catedra Ingeniería de Software - CESE. *Ingeniería de Software - Arquitectura*. Disponible: 2021-05-07. URL: https://campus.fi.uba.ar/pluginfile.php/384042/mod_resource/content/4/Tema%204%20-%20Arquitectura.pdf.
- [32] Ing. Cristian Trinidad. *Repositorio trabajo final CESE*. Disponible: 2021-05-07. URL: https://github.com/ctrinidad-ghub/TP_CESE.
- [33] KiCad. *KiCad EDA*. Disponible: 2021-05-07. URL: <https://www.kicad.org/>.
- [34] Doxygen. *Doxygen*. Disponible: 2021-05-07. URL: <https://www.doxygen.nl/index.html>.
- [35] Espressif Systems (Shanghai) Co., Ltd. *Analog to Digital Converter*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc.html>.
- [36] Espressif Systems (Shanghai) Co., Ltd. *Inter-IC Sound (I2S)*. Disponible: 2021-05-07. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/i2s.html>.
- [37] Espressif Systems (Shanghai) Co., Ltd. *Non-volatile storage library*. Disponible: 2021-05-07. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html.
- [38] Espressif Systems (Shanghai) Co., Ltd. *Event Loop Library*. Disponible: 2021-05-07. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/api-reference/system/esp_event.html.
- [39] Espressif Systems (Shanghai) Co., Ltd. *ESP HTTP Client*. Disponible: 2021-05-07. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_http_client.html.
- [40] Eric Pernia. *API de la biblioteca sAPI*. Disponible: 2021-05-07. URL: https://github.com/epernia/firmware_v3/blob/master/libs/sapi/documentation/api_reference_es.md.
- [41] Ing. Alejandro Permingeat. *Clase 5*. Disponible: 2021-05-07. URL: https://campus.fi.uba.ar/pluginfile.php/335674/mod_resource/content/1/CESE2020-10co-TestingSoftware-Clase5-vs1.pdf.