

Índice general

Resumen	1
1. Introducción general	
1.1. Motivación	1
1.2. Estado del arte	1
1.3. Alcance y objetivos	2
1.4. Requerimientos	2
2. Introducción específica	5
2.1. Tecnologías de hardware utilizadas	5
2.1.1. Espressif ESP32	5
2.1.2. Sensor de temperatura y humedad DHT11	5
2.1.3. Sensor de presión BMP280	6
2.1.4. Fotorresistor como sensor de luminosidad	6
2.1.5. Joystick analógico	6
2.1.6. Display LCM1602A	7
2.1.7. Motores de corriente continua	7
2.1.8. Módulos L298N	7
2.1.9. Pilas de Li-Ion 3,7 V	8
2.1.10. Baterías AA 1,5 V	8
2.1.11. Plaquetas genéricas PCB	9
2.1.12. Cables de conexión DuPont	9
2.1.13. Board Pin Headers para montaje de componentes y cables	10
2.1.14. Interruptores de On-Off	10
2.1.15. Portapilas	10
2.1.16. Ruedas	11
2.1.17. Anemómetro digital	11
2.2. Tecnologías de software utilizadas	12
2.2.1. Marco de trabajo ESP-IDF	12
2.2.2. Plataforma Docker	12
2.2.3. Visual Studio Code	13
2.2.4. Sistema operativo Ubuntu	13
3. Diseño e implementación	15
3.1. Arquitectura de software del sistema	15
3.1.1. Arquitectura y diseño de componentes en la versión v1.0	15
3.1.2. Arquitectura y diseño de componentes en la versión v2.0	17
3.2. Implementación de los módulos	18
3.2.1. Control de la red Wi-Fi	19
3.2.2. Control del joystick analógico	19
3.2.3. Medición de valor de luminosidad	20
3.2.4. Medición de temperatura y humedad	21

Índice general

Resumen	1
1. Introducción general	
1.1. Motivación	1
1.2. Estado del arte	1
1.3. Alcance y objetivos	2
1.4. Requerimientos	2
2. Introducción específica	5
2.1. Tecnologías de hardware utilizadas	5
2.1.1. Espressif ESP32	5
2.1.2. Sensor de temperatura y humedad DHT11	5
2.1.3. Sensor de presión BMP280	6
2.1.4. Fotorresistor como sensor de luminosidad	6
2.1.5. Joystick analógico	6
2.1.6. Display LCM1602A	7
2.1.7. Motores de corriente continua	7
2.1.8. Módulos L298N	7
2.1.9. Pilas de Li-Ion 3,7 V	8
2.1.10. Baterías AA 1,5 V	8
2.1.11. Plaquetas genéricas PCB	9
2.1.12. Cables de conexión DuPont	9
2.1.13. Board Pin Headers para montaje de componentes y cables	10
2.1.14. Interruptores de On-Off	10
2.1.15. Portapilas	10
2.1.16. Ruedas	11
2.1.17. Anemómetro digital	11
2.2. Tecnologías de software utilizadas	12
2.2.1. Marco de trabajo ESP-IDF	12
2.2.2. Plataforma Docker	12
2.2.3. Testing unitario	13
2.2.4. Plataforma de CI/CD	13
2.2.5. Visual Studio Code	14
2.2.6. Sistema operativo Ubuntu	14
3. Diseño e implementación	15
3.1. Arquitectura de software del sistema	15
3.1.1. Arquitectura y diseño de componentes en la versión v1.0	15
3.1.2. Arquitectura y diseño de componentes en la versión v2.0	17
3.2. Implementación de los módulos	18
3.2.1. Control de la red Wi-Fi	19
3.2.2. Control del joystick analógico	19

3.2.5. Medición de presión	21
3.2.6. Control de motores DC	22
3.2.7. Control del display	23
3.3. Arquitectura de hardware	24
3.3.1. Ensamblado final del producto v2.0	24
3.3.2. Conexionado lógico	25
3.3.3. Conexionado físico	26
3.4. Plataforma de desarrollo y ciclo de CI/CD	27
4. Ensayos y resultados	31
4.1. Proceso de desarrollo y aseguramiento de calidad	31
4.2. Verificación técnica de los diferentes módulos	31
4.2.1. Verificación del módulo de joystick	31
4.2.2. Verificación del módulo de control del display	32
4.2.3. Verificación del módulo de control de motores	32
4.2.4. Verificación del módulo de medición de temperatura y humedad	32
4.2.5. Verificación del módulo de medición de presión atmosférica	32
4.2.6. Verificación del módulo de medición de luminosidad	32
4.2.7. Verificación del módulo de comunicación UTP sobre Wi-Fi	32
4.3. Pruebas funcionales y validación del producto	32
4.3.1. Prueba y validación del módulo de visualización de display	33
4.3.2. Prueba y validación del módulo de medición de temperatura y humedad	33
4.3.3. Prueba y validación del módulo de medición de presión atmosférica	35
4.3.4. Prueba y validación del módulo de medición de luminosidad ambiental	36
4.3.5. Prueba y validación del control y desplazamiento del robot	37
4.4. Videos del producto durante el prototipado, ensamblado y experimentación	37
4.4.1. Videos demostrativos del producto final	38
4.4.2. Videos durante el prototipado y ensamblado del robot	38
4.5. Reportes de testing	38
4.6. Documentación del producto	39
5. Conclusiones	41
5.1. Próximos pasos	41
Bibliografía	43

3.2.3. Medición de valor de luminosidad	20
3.2.4. Medición de temperatura y humedad	21
3.2.5. Medición de presión	21
3.2.6. Control de motores DC	22
3.2.7. Control del display	23
3.3. Arquitectura de hardware	24
3.3.1. Ensamblado final del producto v2.0	24
3.3.2. Conexionado lógico	25
3.3.3. Conexionado físico	26
3.4. Plataforma de desarrollo y ciclo de CI/CD	27
3.5. Reportes de ejecución y cobertura de testing unitario	31
4. Ensayos y resultados	33
4.1. Proceso de desarrollo y aseguramiento de calidad	33
4.2. Verificación técnica de los diferentes módulos	33
4.2.1. Verificación del módulo de joystick	33
4.2.2. Verificación del módulo de control del display	34
4.2.3. Verificación del módulo de control de motores	34
4.2.4. Verificación del módulo de medición de temperatura y humedad	34
4.2.5. Verificación del módulo de medición de presión atmosférica	34
4.2.6. Verificación del módulo de medición de luminosidad	34
4.2.7. Verificación del módulo de comunicación UTP sobre Wi-Fi	34
4.3. Pruebas funcionales y validación del producto	34
4.3.1. Prueba y validación del módulo de visualización de display	35
4.3.2. Prueba y validación del módulo de medición de temperatura y humedad	35
4.3.3. Prueba y validación del módulo de medición de presión atmosférica	37
4.3.4. Prueba y validación del módulo de medición de luminosidad ambiental	38
4.3.5. Prueba y validación del control y desplazamiento del robot	39
4.4. Videos del producto durante el prototipado, ensamblado y experimentación	39
4.4.1. Videos demostrativos del producto final	40
4.4.2. Videos durante el prototipado y ensamblado del robot	40
4.5. Documentación del producto	40
5. Conclusiones	41
5.1. Próximos pasos	41
Bibliografía	43

Índice de figuras

2.1. Microcontrolador ESP32-WROOM-32D.	5
2.2. Sensor DHT11.	6
2.3. Sensor BMP280.	6
2.4. Fotorresistor.	6
2.5. Joystick analógico.	7
2.6. Display LCM1602A.	7
2.7. Motor de corriente continua.	7
2.8. Interruptores de On-Off.	8
2.9. Baterías Li-Ion.	8
2.10. Baterías AA.	9
2.11. Plaquetas genéricas.	9
2.12. Cables DuPont.	9
2.13. Pines.	10
2.14. Interruptores de On-Off.	10
2.15. Portapilas.	11
2.16. Ruedas.	11
2.17. Anemómetro digital AOPUTRIVER AP-007-WM.	12
2.18. Proceso de desarrollo utilizando ESP-IDF ¹ .	13
3.1. Arquitectura global.	16
3.2. Arquitectura global.	17
3.3. Conexionado joystick.	20
3.4. Conexionado fotorresistor.	21
3.5. Circuito del conexionado DHT11.	21
3.6. Conexionado BMP280.	22
3.7. Conexionado motores.	23
3.8. Conexionado display.	23
3.9. Hardware del robot.	24
3.10. Detalles del hardware del robot.	24
3.11. Detalles del hardware del joystick.	25
3.12. Conexionado del robot.	25
3.13. Conexionado del joystick.	26
3.14. Conexionado físico del robot.	26
3.15. Ejecución de tests por consola.	28
3.16. CloudBuild.	28
3.17. Github.	29
3.18. ArtifactRegistry.	29
4.1. Visualización del display en la oscuridad.	33
4.2. Medición de humedad en el interior.	34
4.3. Medición de temperatura en el interior.	34
4.4. Medición de humedad en el exterior.	34
4.5. Medición de temperatura en el exterior.	35

Índice de figuras

2.1. Microcontrolador ESP32-WROOM-32D.	5
2.2. Sensor DHT11.	6
2.3. Sensor BMP280.	6
2.4. Fotorresistor.	6
2.5. Joystick analógico.	7
2.6. Display LCM1602A.	7
2.7. Motor de corriente continua.	7
2.8. Interruptores de On-Off.	8
2.9. Baterías Li-Ion.	8
2.10. Baterías AA.	9
2.11. Plaquetas genéricas.	9
2.12. Cables DuPont.	9
2.13. Pines.	10
2.14. Interruptores de On-Off.	10
2.15. Portapilas.	11
2.16. Ruedas.	11
2.17. Anemómetro digital AOPUTRIVER AP-007-WM.	12
2.18. Proceso de desarrollo utilizando ESP-IDF ¹ .	13
3.1. Arquitectura global.	16
3.2. Arquitectura global.	17
3.3. Conexionado joystick.	20
3.4. Conexionado fotorresistor.	21
3.5. Circuito del conexionado DHT11.	21
3.6. Conexionado BMP280.	22
3.7. Conexionado motores.	23
3.8. Conexionado display.	23
3.9. Hardware del robot.	24
3.10. Detalles del hardware del robot.	24
3.11. Detalles del hardware del joystick.	25
3.12. Conexionado del robot.	25
3.13. Conexionado del joystick.	26
3.14. Conexionado físico del robot.	26
3.15. Ejecución de tests por consola.	28
3.16. CloudBuild.	29
3.17. Github.	30
3.18. ArtifactRegistry.	31
3.19. Reportes de testing por consola.	32
3.20. Reportes de testing web.	32
4.1. Visualización del display en la oscuridad.	35
4.2. Medición de humedad en el interior.	36
4.3. Medición de temperatura en el interior.	36

VIII

4.6. Medición de presión atmosférica en el interior.	35
4.7. Medición de presión atmosférica en el exterior.	36
4.8. Medición de luminosidad ambiental en el exterior durante el día. .	36
4.9. Medición de luminosidad ambiental en interiores durante el día. .	37
4.10. Medición de luminosidad ambiental en interiores durante la noche. .	37
4.11. Reportes de testing por consola.	38
4.12. Reportes de testing web.	39

VIII

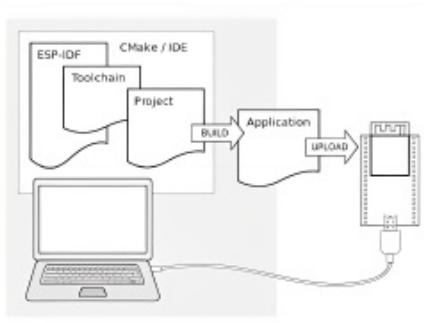
4.4. Medición de humedad en el exterior.	36
4.5. Medición de temperatura en el exterior.	37
4.6. Medición de presión atmosférica en el interior.	37
4.7. Medición de presión atmosférica en el exterior.	38
4.8. Medición de luminosidad ambiental en el exterior durante el día. .	38
4.9. Medición de luminosidad ambiental en interiores durante el día. .	39
4.10. Medición de luminosidad ambiental en interiores durante la noche. .	39

Índice de tablas

3.1. Configuración de AP WiFi	19
3.2. Conexionado joystick	19
3.3. Conexionado fotoresistor	20
4.1. Resultados de mediciones de temperatura y humedad	34
4.2. Resultados de mediciones de presión ambiental.	35

Índice de tablas

3.1. Configuración de AP WiFi	19
3.2. Conexionado joystick	19
3.3. Conexionado fotoresistor	20
4.1. Resultados de mediciones de temperatura y humedad	36
4.2. Resultados de mediciones de presión ambiental.	37

FIGURA 2.18. Proceso de desarrollo utilizando ESP-IDF¹.

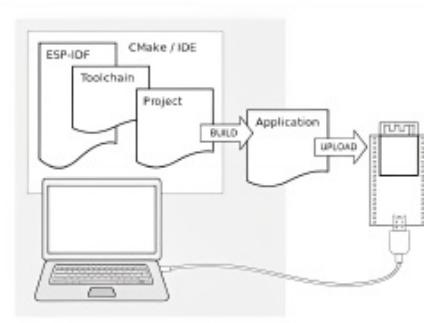
permitir que contenedores livianos independientes se ejecuten en paralelo de manera aislada evitando la sobrecarga de iniciar y mantener máquinas virtuales.

2.2.3. Visual Studio Code

Visual Studio Code [22] es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

2.2.4. Sistema operativo Ubuntu

Ubuntu [23] es una distribución Linux basada en Debian GNU/Linux y patrocinado por Canonical, que incluye principalmente software libre y de código abierto. Puede utilizarse en ordenadores y servidores, está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario.

FIGURA 2.18. Proceso de desarrollo utilizando ESP-IDF¹.

permitir que contenedores livianos independientes se ejecuten en paralelo de manera aislada evitando la sobrecarga de iniciar y mantener máquinas virtuales.

2.2.3. Testing unitario

Con el fin de maximizar la calidad durante el proceso de desarrollo del producto se implementaron test unitarios para todos los servicios del robot y del joystick. El conjunto de herramientas utilizadas para tal fin fueron:

- Ceedling [22]: herramienta de orquestación de tests unitarios, inyección de objetos mocks.
- CMock [23]: framework de mock objects para sistemas embebidos.
- Unity [24]: framework de unit testing para sistemas embebidos.
- Gcov [25]: plugin the ceedling para evaluar y reportar la cobertura.

En los capítulos siguientes se describe la configuración de dichas herramientas y se presentan los resultados tras su ejecución.

2.2.4. Plataforma de CI/CD

Durante el proceso de desarrollo del producto se utilizó CI/CD (*continuous integration / continuous delivery*) mediante la integración de las siguientes herramientas:

- Github [26]: servicio de repositorio y control de versiones de código fuente.
- Google Cloud Build [27]: servicio de compilación, empaquetado y ejecución builds.
- Google Artifact Registry [28]: servicio de repositorio y control de versiones de imágenes Docker.

El objetivo de esta configuración de servicios es permitir que por cada cambio en el código fuente versionado en el controlador de versiones Github, se dispare un proceso de compilación y ejecución de tests unitarios notificando en tiempo real si dicho cambio agrega o no una falla al actual estado del desarrollo. En caso

de pasar satisfactoriamente la compilación y ejecución de los tests entonces se genera una nueva imagen docker con la última versión del código compilado y se versiona en Artifact Registry.

2.2.5. Visual Studio Code

Visual Studio Code [29] es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

2.2.6. Sistema operativo Ubuntu

Ubuntu [30] es una distribución Linux basada en Debian GNU/Linux y patrocinado por Canonical, que incluye principalmente software libre y de código abierto. Puede utilizarse en ordenadores y servidores, está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario.

3.2. Implementación de los módulos

19

- Control del display.
- Control del joystick.
- Gestión de la comunicación inalámbrica vía Wi-Fi (en la versión v2.0).

La integración de los módulos se realizó mediante el diseño y construcción de una placa integradora central, que conecta los dispositivos hardware con el microcontrolador ESP-32.

3.2.1. Control de la red Wi-Fi

El módulo de red Wi-Fi está integrado en el chip ESP32, el cual soporta múltiples características [24] por lo tanto a nivel hardware no fue necesario realizar ningún conexiónado. A nivel de software, la gestión del módulo Wi-Fi está incluida en el SDK ESP-IDF, y el acceso a este se realiza desde el módulo ADC 2 [13]. Por este motivo, cuando el sistema embebido utiliza el módulo Wi-Fi, el uso del ADC2 queda restringido a esta funcionalidad por lo tanto cualquier otro dispositivo que deba hacer uso del ADC debe ser configurado para utilizar el ADC1, como en el caso de los módulos de joystick y detección de luminosidad, que se explican en las siguientes secciones.

La configuración del *soft access point Wi-Fi* implementado se realizó en base a los parámetros de red detallados en la siguiente tabla 3.1:

TABLA 3.1. Configuración de AP WiFi

Parámetro	Valor
SSID	Robot
Password	Robot

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [25]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [26].

3.2.2. Control del joystick analógico

Para el desarrollo de este prototipo se utilizó el SDK ESP-IDF y se configuró el módulo ADC1 de acuerdo a los siguientes detalles en la tabla 3.3:

TABLA 3.2. Conexionado joystick.

Channel	Unit	Pin GPIO
6	1	34
7	1	35

A continuación, se puede apreciar el conexionado físico en la figura 3.3.

3.2. Implementación de los módulos

19

- Control del display.
- Control del joystick.
- Gestión de la comunicación inalámbrica vía Wi-Fi (en la versión v2.0).

La integración de los módulos se realizó mediante el diseño y construcción de una placa integradora central, que conecta los dispositivos hardware con el microcontrolador ESP-32.

3.2.1. Control de la red Wi-Fi

El módulo de red Wi-Fi está integrado en el chip ESP32, el cual soporta múltiples características [31] por lo tanto a nivel hardware no fue necesario realizar ningún conexiónado. A nivel de software, la gestión del módulo Wi-Fi está incluida en el SDK ESP-IDF, y el acceso a este se realiza desde el módulo ADC 2 [13]. Por este motivo, cuando el sistema embebido utiliza el módulo Wi-Fi, el uso del ADC2 queda restringido a esta funcionalidad por lo tanto cualquier otro dispositivo que deba hacer uso del ADC debe ser configurado para utilizar el ADC1, como en el caso de los módulos de joystick y detección de luminosidad, que se explican en las siguientes secciones.

La configuración del *soft access point Wi-Fi* implementado se realizó en base a los parámetros de red detallados en la siguiente tabla 3.1:

TABLA 3.1. Configuración de AP WiFi

Parámetro	Valor
SSID	Robot
Password	Robot

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [32]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [33].

3.2.2. Control del joystick analógico

Para el desarrollo de este prototipo se utilizó el SDK ESP-IDF y se configuró el módulo ADC1 de acuerdo a los siguientes detalles en la tabla 3.3:

TABLA 3.2. Conexionado joystick.

Channel	Unit	Pin GPIO
6	1	34
7	1	35

A continuación, se puede apreciar el conexionado físico en la figura 3.3.

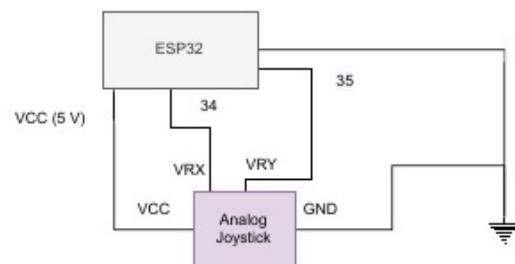


FIGURA 3.3. Conexiónado joystick.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [27]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [28].

3.2.3. Medición de valor de luminosidad

Debido a su fuerte dependencia con la temperatura, y especialmente a que su distribución espectral no resulta adecuada para la medición de iluminancia, los fotoresistores no pueden proporcionar una medición precisa de iluminancia como lo haría un luxómetro. No obstante, el fotoresistor resulta que puede ser utilizado como sensor para proporcionar medidas cuantitativas sobre el nivel de luz, tanto en interiores como en exteriores. Para leer los valores del fotoresistor se utilizó la función `adc1_get_raw` del SDK ESP-IDF, donde el valor devuelto se encuentra en el rango [0 - 2050], con "0"^{el} valor de mayor iluminación y "2050"^{el} menor. Finalmente, para calcular el nivel de iluminación como valor porcentual, en el rango [0-100], siendo cero el nivel más oscuro y cien el más iluminado, se utilizó la siguiente función matemática:

$$\text{Nivel de iluminación} = \frac{\text{MaxReading} - \text{reading}}{\text{MaxReading}} \times 100$$

Donde:

- El valor *reading* es la lectura analógica del valor del fotoresistor.
- El valor *MaxReading* es el máximo valor analógico posible de ser entregado por el fotoresistor, en este caso 2050.
- El valor *MaxReading* es el máximo valor analógico posible de ser entregado por el fotoresistor, en este caso 0.

Para el desarrollo de este prototipo se utilizó el SDK de ESP-IDF y se configuró el módulo ADC1 de acuerdo a los detalles provistos en la tabla 3.3.

TABLA 3.3. Conexiónado fotoresistor.

Channel	Unit	Pin GPIO
0	1	36

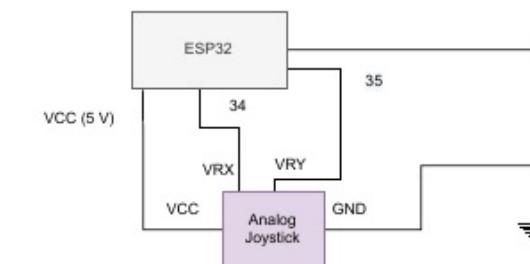


FIGURA 3.3. Conexiónado joystick.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [34]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [35].

3.2.3. Medición de valor de luminosidad

Debido a su fuerte dependencia con la temperatura, y especialmente a que su distribución espectral no resulta adecuada para la medición de iluminancia, los fotoresistores no pueden proporcionar una medición precisa de iluminancia como lo haría un luxómetro. No obstante, el fotoresistor resulta que puede ser utilizado como sensor para proporcionar medidas cuantitativas sobre el nivel de luz, tanto en interiores como en exteriores. Para leer los valores del fotoresistor se utilizó la función `adc1_get_raw` del SDK ESP-IDF, donde el valor devuelto se encuentra en el rango [0 - 2050], con "0"^{el} valor de mayor iluminación y "2050"^{el} menor. Finalmente, para calcular el nivel de iluminación como valor porcentual, en el rango [0-100], siendo cero el nivel más oscuro y cien el más iluminado, se utilizó la siguiente función matemática:

$$\text{Nivel de iluminación} = \frac{\text{MaxReading} - \text{reading}}{\text{MaxReading}} \times 100$$

Donde:

- El valor *reading* es la lectura analógica del valor del fotoresistor.
- El valor *MaxReading* es el máximo valor analógico posible de ser entregado por el fotoresistor, en este caso 2050.
- El valor *MaxReading* es el máximo valor analógico posible de ser entregado por el fotoresistor, en este caso 0.

Para el desarrollo de este prototipo se utilizó el SDK de ESP-IDF y se configuró el módulo ADC1 de acuerdo a los detalles provistos en la tabla 3.3.

TABLA 3.3. Conexiónado fotoresistor.

Channel	Unit	Pin GPIO
0	1	36

3.2. Implementación de los módulos

21

A continuación, se puede apreciar un diagrama de su conexionado físico en la figura 3.4.

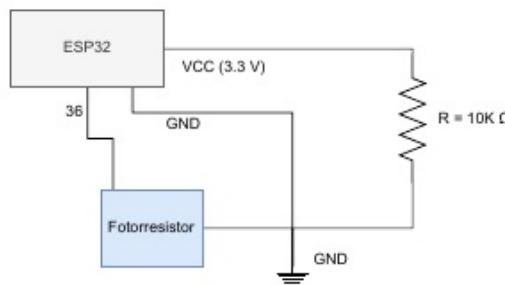


FIGURA 3.4. Conexionado fotorresistor.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [27]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [29].

3.2.4. Medición de temperatura y humedad

Para el desarrollo de este módulo se utilizó la biblioteca de código ESP-IDF-Lib Components Library [19] que provee el soporte para gestionar el DHT11. Para acceder a las lecturas del dispositivo se abstrajo mediante el componente Measuring Service, este es invocado por la tarea Measuring Task y el estado de la lectura es almacenado en el componente Measuring State. A continuación, se puede apreciar el conexionado del prototipo en la figura 3.5.

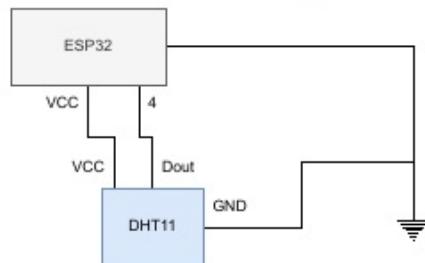


FIGURA 3.5. Circuito del conexionado DHT11.

El desarrollo de este módulo se basó en el ejemplo provisto por la biblioteca ESP-IDF-Lib [30]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [31].

3.2.5. Medición de presión

Para el desarrollo de este módulo se utilizó el framework ESP-IDF y la biblioteca de código ESP-IDF Components que provee el soporte para gestionar el dispositivo BMP280 por medio del protocolo I2C. El driver es inicializado en el componente main-core para ser posteriormente invocado desde el Measuring Service

3.2. Implementación de los módulos

21

A continuación, se puede apreciar un diagrama de su conexionado físico en la figura 3.4.

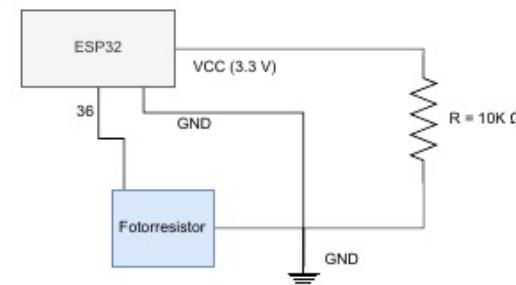


FIGURA 3.4. Conexionado fotorresistor.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [34]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [36].

3.2.4. Medición de temperatura y humedad

Para el desarrollo de este módulo se utilizó la biblioteca de código ESP-IDF-Lib Components Library [19] que provee el soporte para gestionar el DHT11. Para acceder a las lecturas del dispositivo se abstrajo mediante el componente Measuring Service, este es invocado por la tarea Measuring Task y el estado de la lectura es almacenado en el componente Measuring State. A continuación, se puede apreciar el conexionado del prototipo en la figura 3.5.

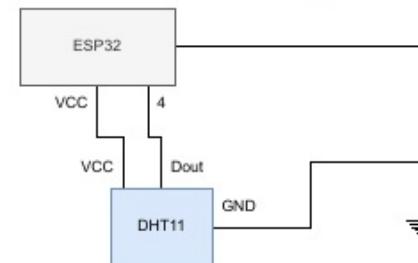


FIGURA 3.5. Circuito del conexionado DHT11.

El desarrollo de este módulo se basó en el ejemplo provisto por la biblioteca ESP-IDF-Lib [37]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [38].

3.2.5. Medición de presión

Para el desarrollo de este módulo se utilizó el framework ESP-IDF y la biblioteca de código ESP-IDF Components que provee el soporte para gestionar el dispositivo BMP280 por medio del protocolo I2C. El driver es inicializado en el componente main-core para ser posteriormente invocado desde el Measuring Service

en la tarea Measuring Task. Sus lecturas son guardadas en el Measuring State. A continuación, se puede apreciar el conexionado del prototipo en la figura 3.6.

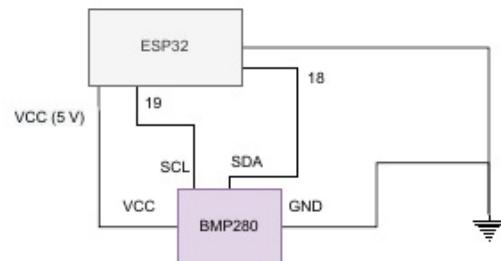


FIGURA 3.6. Conexiónado BMP280.

El desarrollo de este módulo se basó en el ejemplo provisto por la biblioteca ESP-IDF-Lib [32]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [33].

3.2.6. Control de motores DC

Para la implementación del módulo de control de los motores de corriente continua se utilizaron a nivel de hardware dos módulos puentes L298N [34], que permiten integrar y controlar dos motores cada uno. Con el fin de alimentar los módulos con una fuente de poder de corriente y tensión consistente, se utilizaron dos baterías de Li-Ion de 3,7 V y 2000 mA/h conectadas en serie, activadas mediante un interruptor. A nivel driver en el ESP32 se utilizó el módulo de control de motores por modulación de pulsos (MCPWM) [35] que por medio de la configuración de sus unidades y del *duty cycle* [36] se puede controlar el sentido y velocidad de rotación de los motores. Los puentes L298N proporcionan también además una tensión de salida de 5 V, y que se utilizó para la alimentación del ESP32 conectando su pin Vin.

En el siguiente diagrama puede apreciarse el conexionado lógico para el control de los motores con los componentes mencionados.

en la tarea Measuring Task. Sus lecturas son guardadas en el Measuring State. A continuación, se puede apreciar el conexionado del prototipo en la figura 3.6.

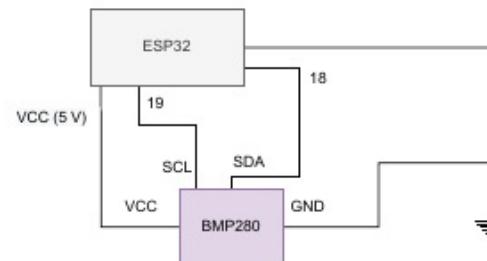


FIGURA 3.6. Conexiónado BMP280.

El desarrollo de este módulo se basó en el ejemplo provisto por la biblioteca ESP-IDF-Lib [39]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [40].

3.2.6. Control de motores DC

Para la implementación del módulo de control de los motores de corriente continua se utilizaron a nivel de hardware dos módulos puentes L298N [41], que permiten integrar y controlar dos motores cada uno. Con el fin de alimentar los módulos con una fuente de poder de corriente y tensión consistente, se utilizaron dos baterías de Li-Ion de 3,7 V y 2000 mA/h conectadas en serie, activadas mediante un interruptor. A nivel driver en el ESP32 se utilizó el módulo de control de motores por modulación de pulsos (MCPWM) [42] que por medio de la configuración de sus unidades y del *duty cycle* [43] se puede controlar el sentido y velocidad de rotación de los motores. Los puentes L298N proporcionan también además una tensión de salida de 5 V, y que se utilizó para la alimentación del ESP32 conectando su pin Vin.

En el siguiente diagrama puede apreciarse el conexionado lógico para el control de los motores con los componentes mencionados.

3.2. Implementación de los módulos

23

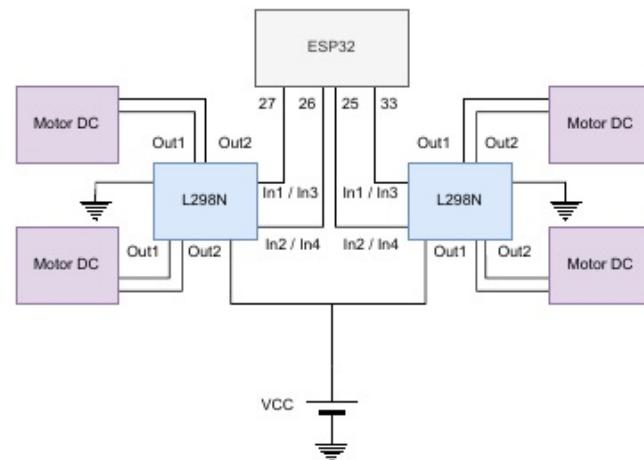


FIGURA 3.7. Conexiónado motores.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [37]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [38].

3.2.7. Control del display

Para la implementación del módulo de visualización de valores observados se integró un display de dos líneas y dieciséis caracteres LCM1602A por medio de un driver I2C que facilita su control. Al basarse en el protocolo I2C el display comparte las mismas líneas SCL y SDA que el sensor BMP280.

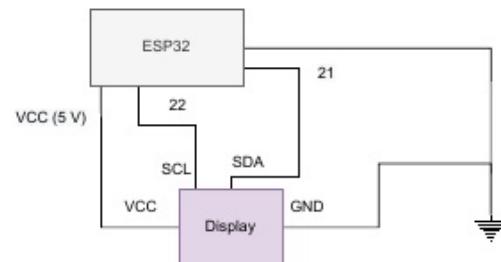


FIGURA 3.8. Conexiónado display.

El desarrollo de este módulo se basó en el ejemplo provisto en el enlace [39]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [40].

3.2. Implementación de los módulos

23

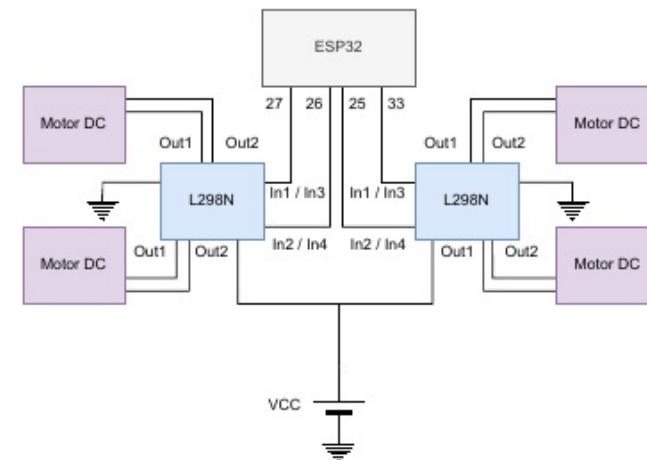


FIGURA 3.7. Conexiónado motores.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [44]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [45].

3.2.7. Control del display

Para la implementación del módulo de visualización de valores observados se integró un display de dos líneas y dieciséis caracteres LCM1602A por medio de un driver I2C que facilita su control. Al basarse en el protocolo I2C el display comparte las mismas líneas SCL y SDA que el sensor BMP280.

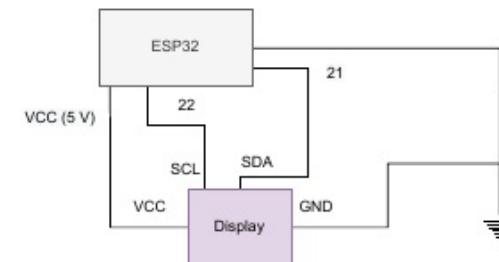


FIGURA 3.8. Conexiónado display.

El desarrollo de este módulo se basó en el ejemplo provisto en el enlace [46]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [47].

3.4. Plataforma de desarrollo y ciclo de CI/CD

Durante el ciclo de desarrollo se utilizaron las herramientas explicadas en el capítulo anterior, y se creó por cada prototipo una imagen Docker extendiendo la de espressif/idf [41]. En el proceso de desarrollo del producto se utilizó CI/CD (*continuous integration / continuous delivery*) mediante la integración de Github con Google Cloud Build, Google Artifact Registry y el uso de imágenes Docker. El conjunto de actividades del mismo fue el siguiente:

1. Codificar localmente en Ubuntu utilizando VSCode.
2. Construcción local en Ubuntu de imagen Docker de acuerdo a la especificación de los siguientes pasos en el archivo docker-compose.yml:
 - a) Compilación del código, enlazado de bibliotecas y empaquetado de la aplicación.
 - b) Ejecución de los tests unitarios con *ceedling*.
 - c) Despliegue (flash) de la aplicación en el ESP32.
3. Versionado del código en el repositorio Github por medio de git commit-push.
4. Construcción en el ambiente de CI/CD por medio de Google Cloud Build de acuerdo a la especificación de los siguientes pasos definidos en el archivo cloudbuild.yml:
 - a) Compilación del código, enlazado de bibliotecas y empaquetado de la aplicación.
 - b) Ejecución de los tests unitarios con *ceedling*.
 - c) Construcción de imagen docker.
 - d) Tagging y versionado de imagen docker en Google Artifact Registry.

A continuación se pueden apreciar capturas de pantallas de cada uno de los sistemas utilizados y pasos ejecutados. En la imagen 3.15 se puede apreciar la salida por consola tras la ejecución de los tests unitarios y construcción de la imagen Docker de manera local. Luego de realizar *commit* y *push* de los cambios locales a Github, en la imagen 3.16 se pueden apreciar los diferentes builds disparados en Cloud Build referenciando los commits de Github, que pueden ser apreciados en la imagen 3.17. Finalmente en la imagen 3.18 se pueden apreciar las imágenes Docker versionadas y almacenadas en Artifact Registry.

3.4. Plataforma de desarrollo y ciclo de CI/CD

Durante el ciclo de desarrollo se utilizaron las herramientas explicadas en el capítulo anterior, y se creó por cada prototipo una imagen Docker extendiendo la de espressif/idf [48]. El conjunto de actividades del mismo fue el siguiente:

1. Codificar localmente en Ubuntu utilizando VSCode.
2. Construcción local en Ubuntu de imagen Docker de acuerdo a la especificación de los siguientes pasos en el archivo docker-compose.yml:
 - a) Compilación del código, enlazado de bibliotecas y empaquetado de la aplicación.
 - b) Ejecución de los tests unitarios con *ceedling*.
 - c) Despliegue (flash) de la aplicación en el ESP32.
3. Versionado del código en el repositorio Github por medio de git commit-push.
4. Construcción en el ambiente de CI/CD por medio de Google Cloud Build de acuerdo a la especificación de los siguientes pasos definidos en el archivo cloudbuild.yml:
 - a) Compilación del código, enlazado de bibliotecas y empaquetado de la aplicación.
 - b) Ejecución de los tests unitarios con *ceedling*.
 - c) Construcción de imagen docker.
 - d) Tagging y versionado de imagen docker en Google Artifact Registry.

A continuación se pueden apreciar capturas de pantallas de cada uno de los sistemas utilizados y pasos ejecutados. En la imagen 3.15 se puede apreciar la salida por consola tras la ejecución de los tests unitarios y construcción de la imagen Docker de manera local. Luego de realizar *commit* y *push* de los cambios locales a Github, en la imagen 3.16 se pueden apreciar los diferentes builds disparados en Cloud Build referenciando los commits de Github, que pueden ser apreciados en la imagen 3.17. Finalmente en la imagen 3.18 se pueden apreciar las imágenes Docker versionadas y almacenadas en Artifact Registry.

```

Test 'test_adc_service.c'
-----
Running test_adc_service.out...

Test 'test_display_service.c'
-----
Running test_display_service.out...

Test 'test_joystick_service.c'
-----
Running test_joystick_service.out...

Test 'test_measuring_services.c'
-----
Running test_measuring_services.out...

Test 'test_motors_service.c'
-----
Running test_motors_service.out...

Test 'test_robot_position_state.c'
-----
Running test_robot_position_state.out...

Test 'test_wifi_service.c'
-----
Running test_wifi_service.out...

OVERALL TEST SUMMARY
-----
TESTED: 28
PASSED: 20
FAILED: 8
IGNORED: 0

```

FIGURA 3.15. Ejecución de tests por consola.

The screenshot shows the Google Cloud Build history interface. At the top, there's a search bar and a filter dropdown set to 'global(nearest)'. Below that is a table with columns: Status, Build, Service, Ref, Commit, Trigger Name, Created, Duration, and Security Insights. The table lists numerous builds for services like adc, display, joystick, measuring, motors, robot position state, and wifi, each with a unique ID, commit hash, creation timestamp, duration, and a link to the build log.

FIGURA 3.16. Cloudbuild.

```

Test 'test_adc_service.c'
-----
Running test_adc_service.out...

Test 'test_display_service.c'
-----
Running test_display_service.out...

Test 'test_joystick_service.c'
-----
Running test_joystick_service.out...

Test 'test_measuring_services.c'
-----
Running test_measuring_services.out...

Test 'test_motors_service.c'
-----
Running test_motors_service.out...

Test 'test_robot_position_state.c'
-----
Running test_robot_position_state.out...

Test 'test_wifi_service.c'
-----
Running test_wifi_service.out...

TEST OUTPUT
-----
[ test_motors_service.c ]
- "initializing mcPWM gpio..."
- "Configuring Initial Parameters of mcPWM..."
- "initializing mcPWM gpio..."
- "Configuring Initial Parameters of mcPWM..."

[ test_wifi_service.c ]
- "wifi_init softap finished. SSID:1 password:1 channel:1"
- "station 12:34:56:78:9A:BC leave, AID=1"
- "station 12:34:56:78:9A:BC leave, AID=1"

OVERALL TEST SUMMARY
-----
TESTED: 39
PASSED: 39
FAILED: 0
IGNORED: 0

```

FIGURA 3.15. Ejecución de tests por consola.

3.4. Plataforma de desarrollo y ciclo de CI/CD

29

The screenshot shows a list of GitHub commits for the repository 'konekuchent85/case_proyecto_especialización'. The commits are grouped by date:

- Commit on Sep 18, 2024:**
 - Fixing the build (multiple commits from 12 days ago)
 - agregando expectativas de los demás test cases
 - agregando descripción de funciones de test para MotorService
 - agregando comentarios (3 commits from 12 days ago)
- Commit on Sep 17, 2024:**
 - completando test cases para investing services
 - agregando comentarios (3 commits from 12 days ago)
- Commit on Sep 15, 2024:**
 - Fixing the build (multiple commits from 1 day ago)
 - Fixing the build (multiple commits from 1 day ago)
 - Fixing the build (multiple commits from 1 day ago)

FIGURA 3.17. Github.

The screenshot shows the Google Cloud Artifact Registry interface for the project 'poc-exp32-integración'. It displays a table of digests:

VERSION	FILE	STATUS
1.0.0@sha256:1aef1f2eef1f211602452260a757c6d1	1aef1f2eef1f211602452260a757c6d1	READY
0.0.1@sha256:00000000000000000000000000000000	00000000000000000000000000000000	PENDING

FIGURA 3.18. ArtifactRegistry.

3.4. Plataforma de desarrollo y ciclo de CI/CD

29

The screenshot shows the Google Cloud Build history interface. It displays a table of builds:

ID	Status	Source	Ref	Commit	Created	Duration
9922e7b6	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/21/24, 11:40 AM	2m 49 sec
45fb6e07	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/21/24, 5:58 AM	5m 59 sec
90fb9f11	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/20/24, 7:38 AM	2m 18 sec
1aef1f2eef1f211602452260a757c6d1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/20/24, 7:22 AM	2m 13 sec
236fc4ff	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/20/24, 7:22 AM	9m 14 sec
05979273	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 7:19 PM	2m 18 sec
3bca9176	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 8:16 AM	2m 13 sec
611ac50f	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 8:12 AM	2m 10 sec
953b4e20	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 8:03 AM	2m 49 sec
1aef1f2eef1f211602452260a757c6d1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 8:02 AM	37 sec
d9a0547d	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 11:01 AM	2m 36 sec
c0fa499	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 8:27 AM	5 sec
a746d691	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 8:25 AM	9 sec
1aef1f2eef1f211602452260a757c6d1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 8:23 AM	9 sec
1aef1f2eef1f211602452260a757c6d1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 5:58 AM	6 sec
1aef1f2eef1f211602452260a757c6d1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 9:15 AM	2m 16 sec
05297326	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:48 PM	4m 48 sec
97737228	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:49 PM	2 sec
935c948e	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:27 PM	2m 21 sec
70580fd1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:29 PM	2m 18 sec
05949890	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:28 PM	6 sec
859077a0	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:25 PM	7 sec
1aef1f2eef1f211602452260a757c6d1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:22 PM	5 sec
97344697	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:20 PM	0 sec
28304380	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:09 PM	2m 23 sec
42991187	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:08 PM	-
498422e1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:05 PM	5 sec
45b6f8f1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:00 PM	10 sec
434b15b9	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 2:02 PM	9 sec
3bca9176	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 1:55 PM	2m 7 sec
430af2ce	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 5:34 PM	5 sec
e4628770	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 5:14 PM	5 sec
97283224	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:38 PM	2m 18 sec
1aef1f2eef1f211602452260a757c6d1	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:50 PM	2m 32 sec
e5340fb2	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:46 PM	2m 12 sec
20300017	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:45 PM	1m 59 sec
4a6d2c60	COMPLETED	konekuchent85/case_proyecto_especialización	main	1aef1f2eef1f211602452260a757c6d1	9/19/24, 4:45 PM	5 sec

FIGURA 3.16. CloudBuild.

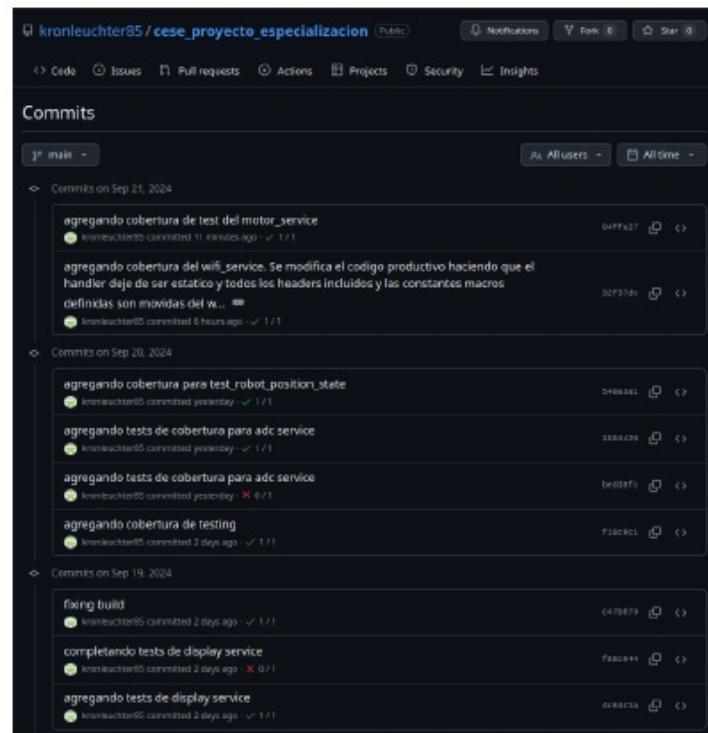


FIGURA 3.17. Github.

Capítulo 4

Ensayos y resultados

Esta sección presenta los diferentes prototipos realizados para determinar la viabilidad de cada una de las funcionalidades provistas, la metodología de desarrollo, testing y, finalmente, los entregables finales del trabajo.

4.1. Proceso de desarrollo y aseguramiento de calidad

Para el proceso de desarrollo se realizaron pruebas de concepto de las diferentes funcionalidades utilizando como materiales la bibliografía encontrada en Internet, las hojas de datos y los ejemplos de código provistos por el SDK y bibliotecas empleadas. Una vez logrado el objetivo funcional de componente se optimizó y encapsuló cada módulo para ser integrado de manera individual a un prototipo integrador sin afectar el funcionamiento de cualquier otro módulo. De esta manera se desarrolló un prototipo integrador como la sumatoria de todos los módulos de forma incremental, probándose por regresión que los módulos ya integrados previamente siguieran funcionando de forma óptima.

Una vez logrado el prototipo integrador con todas las funcionalidades de la versión v1.0, se procedió a expandir el hardware para crear la versión v2.0. Para ello, se extrajeron los módulos de joystick y display, que posteriormente se agregarían al sistema embebido del joystick, y se incorporaron los módulos de conectividad UDP sobre Wi-Fi.

Tras lograr la versión v2.0 se repite el proceso de control de calidad de los diferentes módulos ya integrados.

A continuación, se detallan las diferentes pruebas realizadas.

4.2. Verificación técnica de los diferentes módulos

Todos los módulos fueron probados mediante una inspección visual durante el proceso de pruebas de concepto.

4.2.1. Verificación del módulo de joystick

Se verificó visualmente que los valores del joystick analógico puedan ser leídos apropiadamente, y que sean representativos y relevantes con la dirección del movimiento de la palanca sobre sus coordenadas X e Y.

3.5. Reportes de ejecución y cobertura de testing unitario

The screenshot shows a Google Cloud interface for managing artifacts. At the top, there's a navigation bar with 'Google Cloud', a search bar, and a 'ceser-robot' dropdown. Below the header, a breadcrumb trail shows 'europe-deecker.pkg.dev > ceser-robot > pao-esp32-integracion > pao-esp32-integracion'. A 'DIGESTS' button is visible. The main area has tabs for 'VERSIONS' (selected) and 'FILES'. A 'Filter' input field is present. A table lists artifact versions with columns for 'Name', 'Tag', 'Created', and 'Updated'. The table contains 12 rows of data, each with a small icon and a long alphanumeric string.

Name	Tag	Created	Updated
01ccb623cb4e3	04f1a2724acb0cdc70d-f94e41209d1648fb5f3	1 minute ago	1 minute ago
02f371e032348	201971e53314794028ebcbe7ea5a6d87274dd94	6 hours ago	6 hours ago
038e6335cc5b72	646a30161e7a707e3edfb053a5aef0f034a839c6	1 day ago	1 day ago
0414732490900	19644015411ef2091b769863379344a1ff1d90	1 day ago	1 day ago
05ee72f117349	f1e3e9c141cb6770a7baw655a4415a799e5db	2 days ago	2 days ago
062e67b5749b	c47597702281212425a1e0fae03fb88edbd124a	2 days ago	2 days ago
07ab545a599e9	dd4d1ac0c0793745ab5b112662c7b9b5c119a8	2 days ago	2 days ago
08cf12ea84ff	1e6f108e72f5778a85c1f5a0571a91d1f727e54	3 days ago	3 days ago
097dc8e30494	def6b1469476f7279118ee810d99d7e77876e01	4 days ago	4 days ago
0a991c4f5e4c155a1aa0e081472f5a2b4ed	f4d931c4f5e4c155a1aa0e081472f5a2b4ed	5 days ago	5 days ago

FIGURA 3.18. ArtifactRegistry.

3.5. Reportes de ejecución y cobertura de testing unitario

A continuación se presentan los reportes de testing generados por la herramienta *ceedling* con el complemento *gcov* donde se puede apreciar el nivel de cobertura logrado para cada servicio.

En la siguiente imagen 3.19 se puede apreciar la salida por pantalla tras la ejecución local de *ceedling* con el plugin de cobertura, en donde se evidencia la cantidad de test cases.

4.2.2. Verificación del módulo de control del display

Se verificó visualmente que el display representa los caracteres programados en la prueba de concepto con una intensidad de luz aceptable para poder leerlos apropiadamente.

4.2.3. Verificación del módulo de control de motores

Se verificó visualmente que individualmente el motor pudiera girar en ambos sentidos. Luego, al implementarse los cuatro motores con sus ruedas, se probó que se puedan realizar los giros en todas las direcciones.

4.2.4. Verificación del módulo de medición de temperatura y humedad

Se verificó visualmente que los valores obtenidos por el sensor DHT11 fueran cercanos a lo esperado en relación a la temperatura en el interior del lugar de experimentación y la humedad en un valor cercano a lo reportado en Google.

4.2.5. Verificación del módulo de medición de presión atmosférica

Se verificó visualmente que el valor obtenido por el sensor BMP280 fuera cercano a lo esperado en relación al valor reportado por Google.

4.2.6. Verificación del módulo de medición de luminosidad

Se verificó visualmente que los valores obtenidos del fotoresistor, tras ser transformados a valores absolutos porcentuales, guardan relación con el nivel de luminosidad ambiental del interior del lugar de experimentación.

4.2.7. Verificación del módulo de comunicación UTP sobre Wi-Fi

Por medio de dos programas UDP, uno cliente y uno servidor, se probó el establecimiento de la comunicación UDP entre dos ESP32. Luego se incorporó el servicio de comunicaciones UDP en el robot y mientras que desde el programa cliente se enviaban las acciones representando las direcciones del movimiento a realizar (FORWARD, BACKWARD, LEFT, RIGHT) y se observó visualmente cómo el robot giraba sus ruedas en función de los comandos enviados. Finalmente, se incorporó el módulo de comunicaciones en el joystick y se procedió a activar el desplazamiento en cada una de sus direcciones. Se evidenció el correcto funcionamiento del robot al desplazarse en la dirección de cada comando accionado.

4.3. Pruebas funcionales y validación del producto

El proceso de validación y pruebas del producto se realizó comparando el resultado obtenido con los valores esperados en el alcance del proyecto.

Para la medición de temperatura, humedad y presión se utilizó el anemómetro digital de la figura 2.17, mientras que para la validación de la medición de luminosidad ambiental se utilizó la observación visual.

Las pruebas realizadas fueron las siguientes:

- Prueba y validación del módulo de visualización de display.

```
-----  
GCov: OVERALL TEST SUMMARY  
-----  
TESTED: 39  
PASSED: 39  
FAILED: 0  
IGNORED: 0  
  
-----  
GCov: CODE COVERAGE SUMMARY  
-----  
adc_service.c Lines executed:100.00% of 17  
adc_service.c No branches  
adc_service.c Calls executed:100.00% of 13  
  
display_service.c Lines executed:100.00% of 11  
display_service.c No branches  
display_service.c Calls executed:100.00% of 7  
  
joystick_service.c Lines executed:91.67% of 24  
joystick_service.c Branches executed:100.00% of 12  
joystick_service.c Taken at least once:83.33% of 12  
joystick_service.c Calls executed:100.00% of 2  
  
measuring_services.c Lines executed:77.78% of 36  
measuring_services.c Branches executed:100.00% of 10  
measuring_services.c Taken at least once:88.89% of 18  
measuring_services.c Calls executed:71.43% of 7  
  
motors_service.c Lines executed:100.00% of 30  
motors_service.c Branches executed:100.00% of 4  
motors_service.c Taken at least once:75.00% of 4  
motors_service.c Calls executed:100.00% of 15  
  
robot_position_state.c Lines executed:81.48% of 27  
robot_position_state.c Branches executed:100.00% of 24  
robot_position_state.c Taken at least once:87.50% of 24  
robot_position_state.c No calls  
  
wifi_service.c Lines executed:100.00% of 19  
wifi_service.c Branches executed:100.00% of 4  
wifi_service.c Taken at least once:100.00% of 4  
wifi_service.c Calls executed:100.00% of 15
```

FIGURA 3.19. Reportes de testing por consola.

En la siguiente imagen 3.20 se pueden apreciar los detalles de la cobertura por cada servicio.

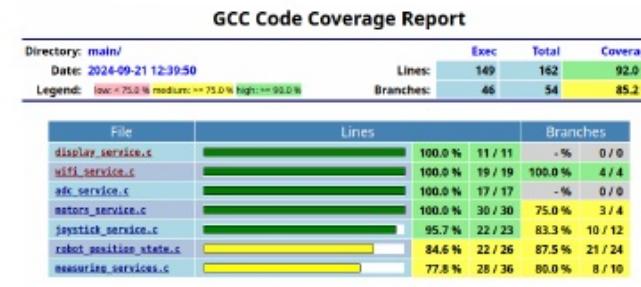


FIGURA 3.20. Reportes de testing web.

- Prueba y validación del módulo de medición de temperatura y humedad.
- Prueba y validación del módulo de medición de presión atmosférica.
- Prueba y validación del módulo de medición de luminosidad ambiental.
- Prueba y validación del control y desplazamiento del robot.

En las siguientes secciones se presentan las diferentes pruebas funcionales realizadas sobre el producto.

4.3.1. Prueba y validación del módulo de visualización de display

Se verificó el funcionamiento del display visualizando las lecturas de los valores censados y transmitidos por el robot. Se controló que:

- Las lecturas sean nítidas y entendibles.
- Las unidades de medida están presentes.
- Haya un detalle de lo que se está midiendo acompañando las lecturas y la unidad de medida.
- El nivel de luminosidad sea óptimo para permitir la lectura independiente de la iluminación ambiental.
- Se presentan las lecturas de todos los valores observados.

A continuación, se pueden apreciar algunas fotografías tomadas durante el proceso de experimentación:



FIGURA 4.1. Visualización del display en la oscuridad.

En la siguiente sección pueden encontrarse los videos en los que se puede apreciar el funcionamiento del display durante el día [42] y en la oscuridad [43].

4.3.2. Prueba y validación del módulo de medición de temperatura y humedad

Se compararon los valores medidos por el módulo de medición de temperatura y humedad basado en el sensor DHT11 con los obtenidos a través de un dispositivo de medición de temperatura y humedad. Se realizó la medición en diferentes contextos:

Capítulo 4

Ensayos y resultados

Esta sección presenta los diferentes prototipos realizados para determinar la viabilidad de cada una de las funcionalidades provistas, la metodología de desarrollo, testing y, finalmente, los entregables finales del trabajo.

4.1. Proceso de desarrollo y aseguramiento de calidad

Para el proceso de desarrollo se realizaron pruebas de concepto de las diferentes funcionalidades utilizando como materiales la bibliografía encontrada en Internet, las hojas de datos y los ejemplos de código provistos por el SDK y bibliotecas empleadas. Una vez logrado el objetivo funcional de componente se optimizó y encapsuló cada módulo para ser integrado de manera individual a un prototipo integrador sin afectar el funcionamiento de cualquier otro módulo. De esta manera se desarrolló un prototipo integrador como la sumatoria de todos los módulos de forma incremental, probándose por regresión que los módulos ya integrados previamente siguieran funcionando de forma óptima.

Una vez logrado el prototipo integrador con todas las funcionalidades de la versión v1.0, se procedió a expandir el hardware para crear la versión v2.0. Para ello, se extrajeron los módulos de joystick y display, que posteriormente se agregarían al sistema embebido del joystick, y se incorporaron los módulos de conectividad UDP sobre Wi-Fi.

Tras lograr la versión v2.0 se repite el proceso de control de calidad de los diferentes módulos ya integrados.

A continuación, se detallan las diferentes pruebas realizadas.

4.2. Verificación técnica de los diferentes módulos

Todos los módulos fueron probados mediante una inspección visual durante el proceso de pruebas de concepto.

4.2.1. Verificación del módulo de joystick

Se verificó visualmente que los valores del joystick analógico puedan ser leídos apropiadamente, y que sean representativos y relevantes con la dirección del movimiento de la palanca sobre sus coordenadas X e Y.

- En el interior de una casa.
- En el exterior, durante el día.

En la siguiente tabla se pueden apreciar los resultados.

TABLA 4.1. Resultados de mediciones de temperatura y humedad

Contexto	Temp. Robot	Temp. Ref.	Hume. Robot	Hume. Ref.
Interior	22,0	23,6	44,0 - 45,0	58,5
Exterior (día)	17,0	14,0	47,0 - 53,0	62,9 - 64,0
Exterior (noche)	-	-	-	-

A continuación, se pueden apreciar algunas fotografías tomadas durante el proceso de experimentación:

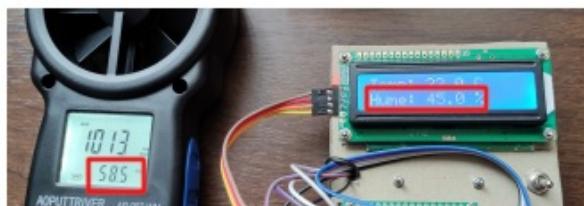


FIGURA 4.2. Medición de humedad en el interior.

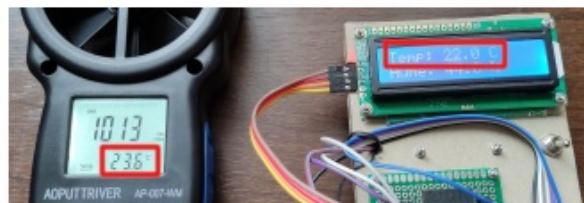


FIGURA 4.3. Medición de temperatura en el interior.



FIGURA 4.4. Medición de humedad en el exterior.

4.2.2. Verificación del módulo de control del display

Se verificó visualmente que el display representara los caracteres programados en la prueba de concepto con una intensidad de luz aceptable para poder leerlos apropiadamente.

4.2.3. Verificación del módulo de control de motores

Se verificó visualmente que individualmente el motor pudiera girar en ambos sentidos. Luego, al implementarse los cuatro motores con sus ruedas, se probó que se puedan realizar los giros en todas las direcciones.

4.2.4. Verificación del módulo de medición de temperatura y humedad

Se verificó visualmente que los valores obtenidos por el sensor DHT11 fueran cercanos a lo esperado en relación a la temperatura en el interior del lugar de experimentación y la humedad en un valor cercano a lo reportado en Google.

4.2.5. Verificación del módulo de medición de presión atmosférica

Se verificó visualmente que el valor obtenido por el sensor BMP280 fuera cercano a lo esperado en relación al valor reportado por Google.

4.2.6. Verificación del módulo de medición de luminosidad

Se verificó visualmente que los valores obtenidos del fotodiodo, tras ser transformados a valores absolutos porcentuales, guardan relación con el nivel de luminosidad ambiental del interior del lugar de experimentación.

4.2.7. Verificación del módulo de comunicación UTP sobre Wi-Fi

Por medio de dos programas UDP, uno cliente y uno servidor, se probó el establecimiento de la comunicación UDP entre dos ESP32. Luego se incorporó el servicio de comunicaciones UDP en el robot y mientras que desde el programa cliente se enviaban las acciones representando las direcciones del movimiento a realizar (FORWARD, BACKWARD, LEFT, RIGHT) y se observó visualmente cómo el robot giraba sus ruedas en función de los comandos enviados. Finalmente, se incorporó el módulo de comunicaciones en el joystick y se procedió a activar el desplazamiento en cada una de sus direcciones. Se evidenció el correcto funcionamiento del robot al desplazarse en la dirección de cada comando accionado.

4.3. Pruebas funcionales y validación del producto

El proceso de validación y pruebas del producto se realizó comparando el resultado obtenido con los valores esperados en el alcance del proyecto.

Para la medición de temperatura, humedad y presión se utilizó el anemómetro digital de la figura 2.17, mientras que para la validación de la medición de luminosidad ambiental se utilizó la observación visual.

Las pruebas realizadas fueron las siguientes:

- Prueba y validación del módulo de visualización de display.

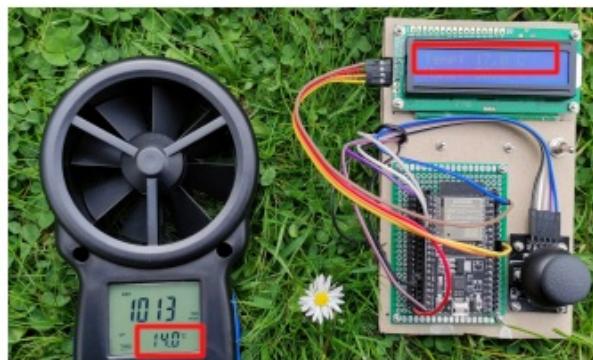


FIGURA 4.5. Medición de temperatura en el exterior.

En la siguiente sección puede encontrarse el video que muestra el funcionamiento del módulo de mediciones, en el que se aprecia el funcionamiento del módulo de medición de temperatura y humedad [42].

4.3.3. Prueba y validación del módulo de medición de presión atmosférica

Se compararon los valores medidos por el módulo de medición de presión basado en el sensor BMP280 con los obtenidos a través de un dispositivo manómetro digital. Las mediciones se realizaron en el interior de la vivienda en dos días distintos.

En la siguiente tabla pueden apreciarse los resultados obtenidos:

TABLA 4.2. Resultados de mediciones de presión ambiental.

Contexto	Presión. Robot	Presión. Ref.
Día 1	1013	1018,9
Día 2	1003,9	998



FIGURA 4.6. Medición de presión atmosférica en el interior.

- Prueba y validación del módulo de medición de temperatura y humedad.
- Prueba y validación del módulo de medición de presión atmosférica.
- Prueba y validación del módulo de medición de luminosidad ambiental.
- Prueba y validación del control y desplazamiento del robot.

En las siguientes secciones se presentan las diferentes pruebas funcionales realizadas sobre el producto.

4.3.1. Prueba y validación del módulo de visualización de display

Se verificó el funcionamiento del display visualizando las lecturas de los valores censados y transmitidos por el robot. Se controló que:

- Las lecturas sean nítidas y entendibles.
- Las unidades de medida están presentes.
- Haya un detalle de lo que se está midiendo acompañando las lecturas y la unidad de medida.
- El nivel de luminosidad sea óptimo para permitir la lectura independiente-mente de la iluminación ambiental.
- Se presentan las lecturas de todos los valores observados.

A continuación, se pueden apreciar algunas fotografías tomadas durante el pro-ceso de experimentación:



FIGURA 4.1. Visualización del display en la oscuridad.

En la siguiente sección pueden encontrarse los videos en los que se puede apre-ciar el funcionamiento del display durante el día [49] y en la oscuridad [50].

4.3.2. Prueba y validación del módulo de medición de temperatura y humedad

Se compararon los valores medidos por el módulo de medición de temperatura y humedad basado en el sensor DHT11 con los obtenidos a través de un dispositivo de medición de temperatura y humedad. Se realizó la medición en diferentes contextos:



FIGURA 4.7. Medición de presión atmosférica en el exterior.

En la siguiente sección puede encontrarse el video que muestra el funcionamiento del módulo de mediciones, en el que se aprecia el funcionamiento del módulo de medición de presión atmosférica [42].

4.3.4. Prueba y validación del módulo de medición de luminosidad ambiental

Se compararon los valores medidos por el módulo de medición de luminosidad basado en un fotoresistor percibidos por el ojo humano sin utilizar ningún dispositivo de medición. Se realizó la medición en diferentes escenarios:

- En exteriores durante el día con luz ambiental.
- En interiores con luz ambiental.
- En interiores a oscuras.

Los resultados mostraron que los valores porcentuales indicados por el módulo de medición de luminosidad son consistentes con los niveles de luz detectados por el ojo humano. En las figuras 4.8, 4.9 y 4.10 pueden apreciarse los resultados de las mediciones.



FIGURA 4.8. Medición de luminosidad ambiental en el exterior durante el día.

- En el interior de una casa.
- En el exterior, durante el día.

En la siguiente tabla se pueden apreciar los resultados.

TABLA 4.1. Resultados de mediciones de temperatura y humedad

Contexto	Temp. Robot	Temp. Ref.	Hume. Robot	Hume. Ref.
Interior	22,0	23,6	44,0 - 45,0	58,5
Exterior (día)	17,0	14,0	47,0 - 53,0	62,9 - 64,0
Exterior (noche)	-	-	-	-

A continuación, se pueden apreciar algunas fotografías tomadas durante el proceso de experimentación:

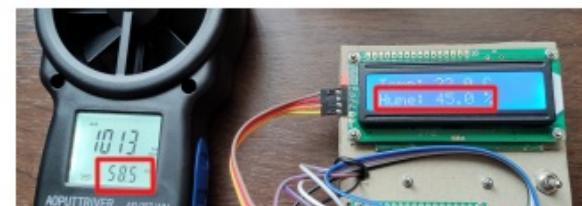


FIGURA 4.2. Medición de humedad en el interior.



FIGURA 4.3. Medición de temperatura en el interior.



FIGURA 4.4. Medición de humedad en el exterior.

4.4. Videos del producto durante el prototipado, ensamblado y experimentación



FIGURA 4.9. Medición de luminosidad ambiental en interiores durante el día.



FIGURA 4.10. Medición de luminosidad ambiental en interiores durante la noche.

En la siguiente sección puede encontrarse el video que muestra el funcionamiento del módulo de mediciones, en el que se aprecia el funcionamiento del módulo de medición de luminosidad ambiental [42].

4.3.5. Prueba y validación del control y desplazamiento del robot

Se verificó el control del desplazamiento del robot de forma visual por medio de accionar el joystick en las diferentes coordenadas (X;Y) y se controló que:

- La dirección del movimiento del robot sea acorde al accionamiento del joystick.
- El tiempo de respuesta en el movimiento del robot y tras el accionar del joystick sea mínimo, permitiendo una buena experiencia de usuario.

En la siguiente sección pueden encontrarse los videos [44] y [45] evidenciando la demostración de este experimento.

4.4. Videos del producto durante el prototipado, ensamblado y experimentación

En las siguientes subsecciones se listan los videos realizados durante el proceso de demostración del producto funcionando así como los grabados casualmente durante armado y prototipado del mismo.

4.3. Pruebas funcionales y validación del producto

37



FIGURA 4.5. Medición de temperatura en el exterior.

En la siguiente sección puede encontrarse el video que muestra el funcionamiento del módulo de mediciones, en el que se aprecia el funcionamiento del módulo de medición de temperatura y humedad [49].

4.3.3. Prueba y validación del módulo de medición de presión atmosférica

Se compararon los valores medidos por el módulo de medición de presión basado en el sensor BMP280 con los obtenidos a través de un dispositivo manómetro digital. Las mediciones se realizaron en el interior de la vivienda en dos días distintos.

En la siguiente tabla pueden apreciarse los resultados obtenidos:

TABLA 4.2. Resultados de mediciones de presión ambiental.

Contexto	Presión. Robot	Presión. Ref.
Día 1	1013	1018,9
Día 2	1003,9	998



FIGURA 4.6. Medición de presión atmosférica en el interior.

4.4.1. Videos demostrativos del producto final

Los experimentos realizados para evidenciar el cumplimiento con los requerimientos funcionales del producto son los siguientes:

- Demo - Hardware del producto [46].
- Demo - Comunicación Wi-Fi [47].
- Demo - Control de movimiento de las ruedas [44].
- Demo - Medición y visualización de parámetros ambientales [42].
- Demo - Control de desplazamiento en un circuito [45].
- Demo - Visualización del Display en la oscuridad [43].

4.4.2. Videos durante el prototipado y ensamblado del robot

- Prototipado Robot v1 - Ensamblado (1) [48].
- Prototipado Robot v1 - Ensamblado (2) [49].
- Prototipado Robot v1 - Ensamblado (3) [50].
- Prototipado Robot v1 - Ensamblado (4) [51].
- Prototipado Robot v2 - Comunicación Joystick Robot (1) [52].
- Prototipado Robot v2 - Comunicación Joystick Robot (2) [53].
- Prototipado Desplazamiento (alimentación USB) [54].
- Prototipado Desplazamiento (alimentación por pilas) [55].

4.5. Reportes de testing

A continuación se pueden apreciar los reportes de testing generados por la herramienta *gcov* con el complemento *gcov*.

```

GCOV: OVERALL TEST SUMMARY
TESTED: 20
PASSED: 20
FAILED: 0
IGNORED: 0

GCOV: CODE COVERAGE SUMMARY

measuring_services.c Lines executed:77.78% of 36
measuring_services.c Branches executed:100.00% of 10
measuring_services.c Taken at least once:80.00% of 10
measuring_services.c Calls executed:71.43% of 7

Could not find coverage results for main/adc_service.c
Could not find coverage results for main/display_service.c
Could not find coverage results for main/joystick_service.c
Could not find coverage results for main/main.c
Could not find coverage results for main/measuring_state.c
Could not find coverage results for main/motors_service.c
Could not find coverage results for main/robot_position_state.c
Could not find coverage results for main/wifi_service.c

```

FIGURA 4.11. Reportes de testing por consola.



FIGURA 4.7. Medición de presión atmosférica en el exterior.

En la siguiente sección puede encontrarse el video que muestra el funcionamiento del módulo de mediciones, en el que se aprecia el funcionamiento del módulo de medición de presión atmosférica [49].

4.3.4. Prueba y validación del módulo de medición de luminosidad ambiental

Se compararon los valores medidos por el módulo de medición de luminosidad basado en un fotoresistor percibidos por el ojo humano sin utilizar ningún dispositivo de medición. Se realizó la medición en diferentes escenarios:

- En exteriores durante el día con luz ambiental.
- En interiores con luz ambiental.
- En interiores a oscuras.

Los resultados mostraron que los valores porcentuales indicados por el módulo de medición de luminosidad son consistentes con los niveles de luz detectados por el ojo humano. En las figuras 4.8, 4.9 y 4.10 pueden apreciarse los resultados de las mediciones.



FIGURA 4.8. Medición de luminosidad ambiental en el exterior durante el día.

4.6. Documentación del producto

39

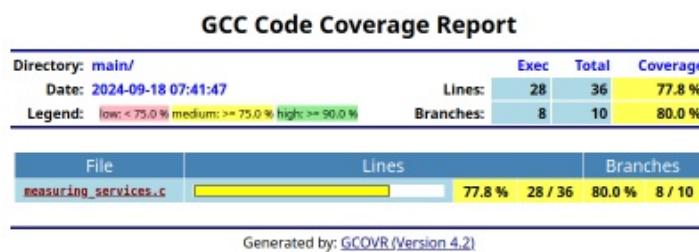


FIGURA 4.12. Reportes de testing web.

4.6. Documentación del producto

Se desarrolló la documentación del producto compuesta de los siguientes entregables:

- Documentación técnica [56].
- Manual de usuario [57].

4.4. Videos del producto durante el prototipado, ensamblado y experimentación



FIGURA 4.9. Medición de luminosidad ambiental en interiores durante el día.



FIGURA 4.10. Medición de luminosidad ambiental en interiores durante la noche.

En la siguiente sección puede encontrarse el video que muestra el funcionamiento del módulo de mediciones, en el que se aprecia el funcionamiento del módulo de medición de luminosidad ambiental [49].

4.3.5. Prueba y validación del control y desplazamiento del robot

Se verificó el control del desplazamiento del robot de forma visual por medio de accionar el joystick en las diferentes coordenadas (X;Y) y se controló que:

- La dirección del movimiento del robot sea acorde al accionamiento del joystick.
- El tiempo de respuesta en el movimiento del robot y tras el accionar del joystick sea mínimo, permitiendo una buena experiencia de usuario.

En la siguiente sección pueden encontrarse los videos [51] y [52] evidenciando la demostración de este experimento.

4.4. Videos del producto durante el prototipado, ensamblado y experimentación

En las siguientes subsecciones se listan los videos realizados durante el proceso de demostración del producto funcionando así como los grabados casualmente durante armado y prototipado del mismo.

4.4.1. Videos demostrativos del producto final

Los experimentos realizados para evidenciar el cumplimiento con los requerimientos funcionales del producto son los siguientes:

- Demo - Hardware del producto [53].
- Demo - Comunicacion Wi-Fi [54].
- Demo - Control de movimiento de las ruedas [51].
- Demo - Medición y visualización de parámetros ambientales [49].
- Demo - Control de desplazamiento en un circuito [52].
- Demo - Visualización del Display en la oscuridad [50].

4.4.2. Videos durante el prototipado y ensamblado del robot

- Prototipado Robot v1 - Ensamblado (1) [55].
- Prototipado Robot v1 - Ensamblado (2) [56].
- Prototipado Robot v1 - Ensamblado (3) [57].
- Prototipado Robot v1 - Ensamblado (4) [58].
- Prototipado Robot v2 - Comunicación Joystick Robot (1) [59].
- Prototipado Robot v2 - Comunicación Joystick Robot (2) [60].
- Prototipado Desplazamiento (alimentación USB) [61].
- Prototipado Desplazamiento (alimentación por pilas) [62].

4.5. Documentación del producto

Se desarrolló la documentación del producto compuesta de los siguientes entregables

- Documentación técnica [63].
- Manual de usuario [64].

- [16] CMake. *CMake*. URL: <https://cmake.org/>.
- [17] CMake. *Ninja*. URL: <https://cmake.org/cmake/help/latest/generator/Ninja.html>.
- [18] Espressif. *ESP-IDF Programming Guide | Get Started*. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/>.
- [19] Readthedocs by Ruslan V. Uss. *ESP-IDF Components library*. URL: <https://esp-idf-lib.readthedocs.io/en/latest/>.
- [20] Espressif Programming Guide. *ESP-IDF Get Started*. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html>.
- [21] Docker. *Docker*. URL: <https://docker.com/>.
- [22] Visualstudio. *Visualstudio Code*. URL: <https://code.visualstudio.com/>.
- [23] Ubuntu. *Ubuntu*. URL: <https://ubuntu.com/>.
- [24] Espressif. *ESP-IDF WiFi*. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/wifi.html>.
- [25] Espressif. *ESP-IDF - Wi-Fi SoftAP Example*. URL: https://github.com/espressif/esp-idf/tree/v4.4/examples/wifi/getting_started/softAP.
- [26] Gonzalo Carreno. POC *ESP32-WiFi v4.4*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-wifi-ap-v4.4.
- [27] Espressif. *ESP-IDF ADC1 Example*. URL: <https://github.com/espressif/esp-idf/tree/v4.0.3/examples/peripherals/adc>.
- [28] Gonzalo Carreno. POC *ESP32-joystick*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-joystick.
- [29] Gonzalo Carreno. POC *ESP32-photoresistor*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-photoresistor.
- [30] UncleRus. *ESP32 - Example for dht driver*. URL: <https://github.com/UncleRus/esp-idf-lib/tree/master/examples/dht/default>.
- [31] Gonzalo Carreno. POC *ESP32-DHT11*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-dht11.
- [32] UncleRus. *ESP32 - Example for bmp280 driver*. URL: <https://github.com/UncleRus/esp-idf-lib/tree/master/examples/bmp280/default>.
- [33] Gonzalo Carreno. POC *ESP32-BMP280*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-bmp280.
- [34] Espressif. *L298N Dual H-Bridge Motor Driver*. URL: <https://www.handsontec.com/datasheets/L298N%20Motor%20Driver.pdf>.
- [35] Espressif. *ESP-IDF Motor Control Pulse Width Modulator (MCPWM)*. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/mcpwm.html>.
- [36] Espressif. *MCPWM*. URL: <https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/peripherals/mcpwm.html>.
- [37] Espressif. *MCPWM brushed dc motor control Example*. URL: https://github.com/espressif/esp-idf/tree/v4.2/examples/peripherals/mcpwm/mcpwm_brushed_dc_control.

- [16] CMake. *CMake*. URL: <https://cmake.org/>.
- [17] CMake. *Ninja*. URL: <https://cmake.org/cmake/help/latest/generator/Ninja.html>.
- [18] Espressif. *ESP-IDF Programming Guide | Get Started*. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/>.
- [19] Readthedocs by Ruslan V. Uss. *ESP-IDF Components library*. URL: <https://esp-idf-lib.readthedocs.io/en/latest/>.
- [20] Espressif Programming Guide. *ESP-IDF Get Started*. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html>.
- [21] Docker. *Docker*. URL: <https://docker.com/>.
- [22] ThrowTheSwitch. *ThrowTheSwitch - Ceedling*. URL: <https://github.com/ThrowTheSwitch/Ceedling>.
- [23] ThrowTheSwitch. *ThrowTheSwitch - CMock*. URL: <https://github.com/ThrowTheSwitch/CMock>.
- [24] ThrowTheSwitch. *ThrowTheSwitch - Unity Test*. URL: <https://github.com/ThrowTheSwitch/Unity>.
- [25] ThrowTheSwitch. *ThrowTheSwitch - Ceedling/GCov*. URL: <https://github.com/ThrowTheSwitch/Ceedling/blob/master/plugins/gcov/README.md>.
- [26] Github. *Github*. URL: <https://github.com/>.
- [27] Google Cloud Platform. *Google Cloud Build*. URL: <https://cloud.google.com/build>.
- [28] Google Cloud Platform. *Google Artifact Registry*. URL: <https://cloud.google.com/artifact-registry>.
- [29] Visualstudio. *Visualstudio Code*. URL: <https://code.visualstudio.com/>.
- [30] Ubuntu. *Ubuntu*. URL: <https://ubuntu.com/>.
- [31] Espressif. *ESP-IDF WiFi*. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/wifi.html>.
- [32] Espressif. *ESP-IDF - Wi-Fi SoftAP Example*. URL: https://github.com/espressif/esp-idf/tree/v4.4/examples/wifi/getting_started/softAP.
- [33] Gonzalo Carreno. POC *ESP32-WiFi v4.4*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-wifi-ap-v4.4.
- [34] Espressif. *ESP-IDF ADC1 Example*. URL: <https://github.com/espressif/esp-idf/tree/v4.0.3/examples/peripherals/adc>.
- [35] Gonzalo Carreno. POC *ESP32-joystick*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-joystick.
- [36] Gonzalo Carreno. POC *ESP32-photoresistor*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-photoresistor.
- [37] UncleRus. *ESP32 - Example for dht driver*. URL: <https://github.com/UncleRus/esp-idf-lib/tree/master/examples/dht/default>.
- [38] Gonzalo Carreno. POC *ESP32-DHT11*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-dht11.

- [38] Gonzalo Carreno. POC ESP32-motor-pwm. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-motor-pwm.
- [39] ESP32 Tutoriales. ESP32 I2C LCD with ESP-IDF. URL: <https://esp32tutorials.com/i2c-lcd-esp32-esp-idf/>.
- [40] Gonzalo Carreno. POC ESP32-DHT11. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-display.
- [41] Espressif. Espressif Docker Image. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/tools/idf-docker-image.html>.
- [42] Gonzalo Carreno. Robot de exploracion ambiental - Demo Medicion y visualizacion de parametros ambientales. URL: https://youtu.be/BBOP3n8_gBg.
- [43] Gonzalo Carreno. Robot de exploracion ambiental - Demo Visualizacion del Display en la oscuridad. URL: <https://youtu.be/LwfYaotAi64>.
- [44] Gonzalo Carreno. Robot de exploracion ambiental - Demo Control de movimiento de las ruedas. URL: <https://youtu.be/FKXWx4Rqr7I>.
- [45] Gonzalo Carreno. Robot de exploracion ambiental - Demo Control de desplazamiento en un circuito. URL: <https://youtu.be/sosSGwCTyaY>.
- [46] Gonzalo Carreno. Robot de exploracion ambiental - Demo Hardware. URL: <https://youtu.be/RNBnDawVJ6c>.
- [47] Gonzalo Carreno. Robot de exploracion ambiental - Demo - Comunicacion WiFi. URL: <https://youtu.be/CcBgv0KjLB0>.
- [48] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (1). URL: <https://youtu.be/tDXT1CsObWE>.
- [49] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (2). URL: <https://youtube.com/shorts/uGqJn2K0LbI>.
- [50] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (3). URL: <https://youtu.be/w9IOoE-d9Cw>.
- [51] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (4). URL: https://youtu.be/obkJ-wM_wNU.
- [52] Gonzalo Carreno. Robot de exploracion ambiental - Prototipado Comunicacion Joystick Robot (1). URL: <https://youtu.be/SnRf6HSya88>.
- [53] Gonzalo Carreno. Robot de exploracion ambiental - Prototipado Comunicacion Joystick Robot (2). URL: <https://youtu.be/jiisheyu95w>.
- [54] Gonzalo Carreno. Robot de exploracion ambiental - Prototipado Desplazamiento (alimentacion USB). URL: https://youtu.be/_w8qdNWC-DQ.
- [55] Gonzalo Carreno. Robot de exploracion ambiental - Prototipado Desplazamiento (alimentacion por pilas). URL: <https://youtu.be/-MxMXKzztHU>.
- [56] Gonzalo Carreno. Robot de exploracion ambiental - Documentacion Tecnica. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/blob/main/docs/Documentacion-Tecnica.pdf.
- [57] Gonzalo Carreno. Robot de exploracion ambiental - Manual de usuario. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/blob/main/docs/Manual-De-Usuario-vFinal.pdf.

- [39] UncleRus. ESP32 - Example for bmp280 driver. URL: <https://github.com/UncleRus/esp-idf-lib/tree/master/examples/bmp280/default>.
- [40] Gonzalo Carreno. POC ESP32-BMP280. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-bmp280.
- [41] Espressif. L298N Dual H-Bridge Motor Driver. URL: <https://www.handsontec.com/datasheets/L298N%20Motor%20Driver.pdf>.
- [42] Espressif. ESP-IDF Motor Control Pulse Width Modulator (MCPWM). URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/mcpwm.html>.
- [43] Espressif. MCPWM. URL: <https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/peripherals/mcpwm.html>.
- [44] Espressif. MCPWM brushed dc motor control Example. URL: https://github.com/espressif/esp-idf/tree/v4.2/examples/peripherals/mcpwm/mcpwm_brushed_dc_control.
- [45] Gonzalo Carreno. POC ESP32-motor-pwm. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-motor-pwm.
- [46] ESP32 Tutoriales. ESP32 I2C LCD with ESP-IDF. URL: <https://esp32tutorials.com/i2c-lcd-esp32-esp-idf/>.
- [47] Gonzalo Carreno. POC ESP32-DHT11. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-display.
- [48] Espressif. Espressif Docker Image. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/tools/idf-docker-image.html>.
- [49] Gonzalo Carreno. Robot de exploracion ambiental - Demo Medicion y visualizacion de parametros ambientales. URL: https://youtu.be/BBOP3n8_gBg.
- [50] Gonzalo Carreno. Robot de exploracion ambiental - Demo Visualizacion del Display en la oscuridad. URL: <https://youtu.be/LwfYaotAi64>.
- [51] Gonzalo Carreno. Robot de exploracion ambiental - Demo Control de movimiento de las ruedas. URL: <https://youtu.be/FKXWx4Rqr7I>.
- [52] Gonzalo Carreno. Robot de exploracion ambiental - Demo Control de desplazamiento en un circuito. URL: <https://youtu.be/sosSGwCTyaY>.
- [53] Gonzalo Carreno. Robot de exploracion ambiental - Demo Hardware. URL: <https://youtu.be/RNBnDawVJ6c>.
- [54] Gonzalo Carreno. Robot de exploracion ambiental - Demo - Comunicacion WiFi. URL: <https://youtu.be/CcBgv0KjLB0>.
- [55] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (1). URL: <https://youtu.be/tDXT1CsObWE>.
- [56] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (2). URL: <https://youtube.com/shorts/uGqJn2K0LbI>.
- [57] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (3). URL: <https://youtu.be/w9IOoE-d9Cw>.
- [58] Gonzalo Carreno. Robot exploracion ambiental - Prototipado Ensamblado Robot v1 (4). URL: https://youtu.be/obkJ-wM_wNU.
- [59] Gonzalo Carreno. Robot de exploracion ambiental - Prototipado Comunicacion Joystick Robot (1). URL: <https://youtu.be/SnRf6HSya88>.