

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

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	36
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 ensamblado y experimentación	40
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	43
5.1. Próximos pasos	43
Bibliografía	45

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 AOPUTTRIVER 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

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 AOPUTTRIVER 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. Listado de <i>commits</i> en Github.	29
3.17. Listado de <i>builds</i> en Google CloudBuild.	30
3.18. Listado de versiones de imágenes docker en Google 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

4.4. 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

VIII

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

2.1. Tecnologías de hardware utilizadas

7

del microcontrolador ESP32. En la figura 2.5 se puede apreciar una imagen del componente.



FIGURA 2.5. Joystick analógico.

2.1.6. Display LCM1602A

El display LCM1602A [14] consta de una pantalla de cristal líquido que permite representar dos filas con hasta 16 caracteres alfanuméricos en cada una y dado que se encuentra integrada a una interfaz adaptadora I2C puede ser controlada por este protocolo. En la figura 2.6 se puede apreciar una imagen del componente.



FIGURA 2.6. Display LCM1602A.

2.1.7. Motores de corriente continua

El motor DC (corriente continua) [15] es un electromotor que transforma energía eléctrica en energía mecánica. Estos motores operan con una tensión entre 3 y 6 V, corriente de 150 mA, permiten una velocidad de entre 90 y 200 RPM y un torque de entre 0,15 Nm y 0,60 Nm. En la figura 2.7 se puede apreciar una imagen del componente.



FIGURA 2.7. Motor de corriente continua.

2.1.8. Módulos L298N

Los módulos L298N son controladores de motores de puente H duales muy utilizados en proyectos de robótica y automoción para controlar motores de corriente continua (DC) y motores paso a paso. El módulo se basa en el IC L298N, que es un controlador de motor de doble puente H fabricado por STMicroelectronics. Este módulo posee dos puentes H que permiten controlar 2 motores DC o un motor paso a paso bipolar/unipolar. El módulo permite controlar el sentido de giro y velocidad mediante señales TTL (*transistor-transistor logic*) que se pueden obtener

2.1. Tecnologías de hardware utilizadas

7

del microcontrolador ESP32. En la figura 2.5 se puede apreciar una imagen del componente.



FIGURA 2.5. Joystick analógico.

2.1.6. Display LCM1602A

El display LCM1602A [14] consta de una pantalla de cristal líquido que permite representar dos filas con hasta 16 caracteres alfanuméricos en cada una y dado que se encuentra integrada a una interfaz adaptadora I2C puede ser controlada por este protocolo. En la figura 2.6 se puede apreciar una imagen del componente.



FIGURA 2.6. Display LCM1602A.

2.1.7. Motores de corriente continua

El motor DC (corriente continua) [15] es un electromotor que transforma energía eléctrica en energía mecánica. Estos motores operan con una tensión entre 3 y 6 V, corriente de 150 mA, permiten una velocidad de entre 90 y 200 RPM y un torque de entre 0,15 Nm y 0,60 Nm. En la figura 2.7 se puede apreciar una imagen del componente.



FIGURA 2.7. Motor de corriente continua.

2.1.8. Módulos L298N

Los módulos L298N [16] son controladores de motores de puente H duales muy utilizados en proyectos de robótica y automoción para controlar motores de corriente continua (DC) y motores paso a paso. El módulo se basa en el IC L298N, que es un controlador de motor de doble puente H fabricado por STMicroelectronics. Este módulo posee dos puentes H que permiten controlar 2 motores DC o un motor paso a paso bipolar/unipolar. El módulo permite controlar el sentido de giro y velocidad mediante señales TTL (*transistor-transistor logic*) que se pueden obtener de microcontroladores. Tiene integrado un regulador de tensión de

de microcontroladores. Tiene integrado un regulador de tensión de 5 V encargado de alimentar la parte lógica del L298N cuya configuración se hace a través de un Jumper y se puede usar para alimentar el microcontrolador.

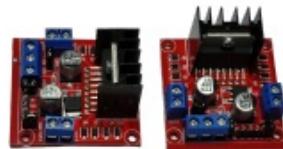


FIGURA 2.8. Interruptores de On-Off.

2.1.9. Pilas de Li-Ion 3,7 V

Las pilas 18650 de 3000 mAh son baterías recargables que utilizan tecnología de iones de litio y se emplean comúnmente en una variedad de dispositivos electrónicos, como por ejemplo linternas de alta potencia, laptops, vehículos eléctricos (como bicicletas y scooters), etc. Las baterías de iones de litio son conocidas por su alta densidad de energía, su baja tasa de autodescarga y su larga vida útil en comparación con otras tecnologías de baterías recargables. Con 3000 mAh (miliamperios-hora), estas baterías ofrecen una duración prolongada, lo que es ideal para dispositivos que requieren mucha energía. Al ser recargables, son más ecológicas y económicas a largo plazo en comparación con las baterías desechables. Además, son generalmente seguras y tienen una buena estabilidad térmica, reduciendo el riesgo de explosiones o incendios.



FIGURA 2.9. Baterías Li-Ion.

2.1.10. Baterías AA 1,5 V

Las pilas AA de 1,5 V con 2600 mWh son baterías de tamaño AA que proporcionan una tensión de 1,5 V y tienen una capacidad energética de 2600 mWh. Ofrecen alta capacidad pudiendo durar más tiempo entre recargas o reemplazos en comparación con las baterías de menor capacidad, y versatilidad, ya que al ser AA son compatibles con la mayoría de los dispositivos electrónicos que requieren alimentación de bajo voltaje. Las pilas AA de 1,5 V con una alta capacidad como 2600 mWh son ideales para dispositivos que requieren un mayor consumo de energía, tales como cámaras digitales, linternas, juguetes electrónicos, mandos a distancia, controles de videojuegos, etc.

5 V encargado de alimentar la parte lógica del L298N cuya configuración se hace a través de un Jumper y se puede usar para alimentar el microcontrolador.

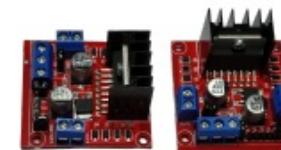


FIGURA 2.8. Interruptores de On-Off.

2.1.9. Pilas de Li-Ion 3,7 V

Las pilas 18650 de 3000 mAh [17] son baterías recargables que utilizan tecnología de iones de litio y se emplean comúnmente en una variedad de dispositivos electrónicos, como por ejemplo linternas de alta potencia, laptops, vehículos eléctricos (como bicicletas y scooters), etc. Las baterías de iones de litio son conocidas por su alta densidad de energía, su baja tasa de autodescarga y su larga vida útil en comparación con otras tecnologías de baterías recargables. Con 3000 mAh (miliamperios-hora), estas baterías ofrecen una duración prolongada, lo que es ideal para dispositivos que requieren mucha energía. Al ser recargables, son más ecológicas y económicas a largo plazo en comparación con las baterías desechables. Además, son generalmente seguras y tienen una buena estabilidad térmica, reduciendo el riesgo de explosiones o incendios.



FIGURA 2.9. Baterías Li-Ion.

2.1.10. Baterías AA 1,5 V

Las pilas AA de 1,5 V con 2600 mWh [18] son baterías de tamaño AA que proporcionan una tensión de 1,5 V y tienen una capacidad energética de 2600 mWh. Ofrecen alta capacidad pudiendo durar más tiempo entre recargas o reemplazos en comparación con las baterías de menor capacidad, y versatilidad, ya que al ser AA son compatibles con la mayoría de los dispositivos electrónicos que requieren alimentación de bajo voltaje. Las pilas AA de 1,5 V con una alta capacidad como 2600 mWh son ideales para dispositivos que requieren un mayor consumo de energía, tales como cámaras digitales, linternas, juguetes electrónicos, mandos a distancia, controles de videojuegos, etc.

2.1. Tecnologías de hardware utilizadas

11

18650, CR2032, etc. Además, suelen venir en tres posibles configuraciones, un solo compartimento (para una sola batería), múltiples compartimentos (para conectar varias baterías en serie o en paralelo), y con interruptor (algunos portapilas incluyen un interruptor para encender o apagar la conexión de las baterías al circuito).



FIGURA 2.15. Portapilas.

2.1.16. Ruedas

Las ruedas de plástico son componentes genéricos utilizados en la comunidad hobbista y estudiantil de electrónica para el armado de sistemas con desplazamiento. Se las puede comprar online a un bajo costo, usualmente vienen en kit de dos o cuatro unidades, y son compatibles con los motores de DC utilizados para moverlas.



FIGURA 2.16. Ruedas.

2.1.17. Anemómetro digital

Para la validación de los parámetros ambientales medidos por el robot se utilizó un dispositivo anemómetro digital AOPUTTRIVER AP-007-WM capaz de medir presión, temperatura y humedad ambiental, y de esta manera poder comparar los valores medidos. En la siguiente figura puede apreciarse una imagen del mismo.

2.1. Tecnologías de hardware utilizadas

11

18650, CR2032, etc. Además, suelen venir en tres posibles configuraciones, un solo compartimento (para una sola batería), múltiples compartimentos (para conectar varias baterías en serie o en paralelo), y con interruptor (algunos portapilas incluyen un interruptor para encender o apagar la conexión de las baterías al circuito).



FIGURA 2.15. Portapilas.

2.1.16. Ruedas

Las ruedas de plástico son componentes genéricos utilizados en la comunidad hobbista y estudiantil de electrónica para el armado de sistemas con desplazamiento. Se las puede comprar online a un bajo costo, usualmente vienen en kit de dos o cuatro unidades, y son compatibles con los motores de DC utilizados para moverlas.



FIGURA 2.16. Ruedas.

2.1.17. Anemómetro digital

Para la validación de los parámetros ambientales medidos por el robot se utilizó un dispositivo anemómetro digital AOPUTTRIVER AP-007-WM [19] capaz de medir presión, temperatura y humedad ambiental, y de esta manera poder comparar los valores medidos. En la siguiente figura puede apreciarse una imagen del mismo.



FIGURA 2.17. Anemómetro digital AOPUTTRIVER AP-007-WM.

2.2. Tecnologías de software utilizadas

2.2.1. Marco de trabajo ESP-IDF

Espressif Systems proporciona recursos básicos de hardware y software para ayudar a los desarrolladores de aplicaciones a realizar sus ideas utilizando el hardware de la serie ESP32. El framework de software de Espressif está destinado al desarrollo de aplicaciones de IoT (Internet de las cosas) con Wi-Fi, Bluetooth, administración de energía y varias otras características del sistema. Sus componentes son:

1. Toolchain, utilizado para compilar el código para ESP32.
2. Build tools, que provee utilidades como CMake [16] y Ninja [17] para construir la aplicación completa para ESP32.
3. ESP-IDF [18], que brinda la API de desarrollo para ESP32 y scripts para ejecutar Toolchain.

Además de las herramientas mencionadas se utilizó el conjunto de bibliotecas y drivers provistos por el proyecto ESP-IDF-Lib [19] basados en el framework ESP-IDF.

En la figura 2.18 se puede apreciar una imagen del proceso de desarrollo y despliegue usando el framework ESP-IDF.

2.2.2. Plataforma Docker

Docker [21] es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para

¹Imagen tomada de [20]



FIGURA 2.17. Anemómetro digital AOPUTTRIVER AP-007-WM.

2.2. Tecnologías de software utilizadas

2.2.1. Marco de trabajo ESP-IDF

Espressif Systems proporciona recursos básicos de hardware y software para ayudar a los desarrolladores de aplicaciones a realizar sus ideas utilizando el hardware de la serie ESP32. El framework de software de Espressif está destinado al desarrollo de aplicaciones de IoT (Internet de las cosas) con Wi-Fi, Bluetooth, administración de energía y varias otras características del sistema. Sus componentes son:

1. Toolchain, utilizado para compilar el código para ESP32.
2. Build tools, que provee utilidades como CMake [20] y Ninja [21] para construir la aplicación completa para ESP32.
3. ESP-IDF [22], que brinda la API de desarrollo para ESP32 y scripts para ejecutar Toolchain.

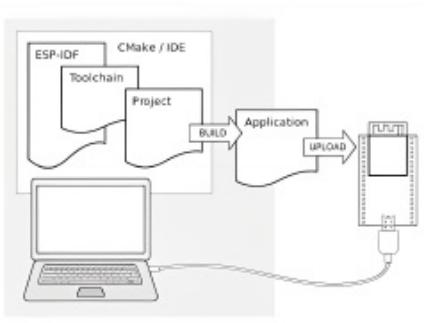
Además de las herramientas mencionadas se utilizó el conjunto de bibliotecas y drivers provistos por el proyecto ESP-IDF-Lib [23] basados en el framework ESP-IDF.

En la figura 2.18 se puede apreciar una imagen del proceso de desarrollo y despliegue usando el framework ESP-IDF.

2.2.2. Plataforma Docker

Docker [25] es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para

¹Imagen tomada de [24]

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.

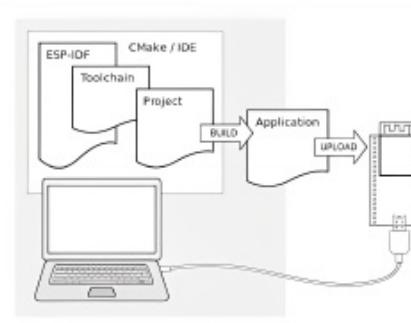
En los capítulos siguientes se describe la configuración de dichas herramientas y se presentan los resultados tras su ejecución.

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

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 [26]: herramienta de orquestación de tests unitarios, inyección de objetos mocks.
- CMock [27]: framework de mock objects para sistemas embebidos.
- Unity [28]: framework de unit testing para sistemas embebidos.
- Gcov [29]: 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 [30]: servicio de repositorio y control de versiones de código fuente.
- Google Cloud Build [31]: servicio de compilación, empaquetado y ejecución builds.
- Google Artifact Registry [32]: 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.

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 [33] 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 [34] 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.

1. Componentes de software
 - a) Main-core
 - b) Measuring Service
 - c) Measuring State
 - d) Robot Position State
 - e) ADC Service
 - f) Motors Service
 - g) WIFI Service
 - h) UDP Service
2. Servicios (tareas)
 - a) Measuring Task
 - b) Motors Task
 - c) UDP Server Task

Los componentes de software y servicios del joystick son:

1. Componentes de software
 - a) main-core
 - b) Measuring State
 - c) Robot Position State
 - d) UDP Service
2. Servicios (tareas)
 - a) Display Task
 - b) Joystick Task
 - c) UDP Server Task

3.2. Implementación de los módulos

Los macro componentes de hardware presentes en la arquitectura de la versión v2.0 son el robot y el joystick, que constituyen dos sistemas embebidos independientes implementados con dos microcontroladores ESP32, integrados entre sí por medio de una red **WiFi** y una comunicación UDP. La arquitectura de **hardware** del sistema esta compuesta por los siguientes módulos:

- En el robot:
 - Control de los motores DC.
 - Control de los sensores de medición (DHT11, BMP280 y fotoresistor).
 - Gestión de la comunicación inalámbrica vía Wi-Fi (en la versión v2.0).
- En el joystick:

1. Componentes de software
 - a) Main-core
 - b) Measuring Service
 - c) Measuring State
 - d) Robot Position State
 - e) ADC Service
 - f) Motors Service
 - g) WIFI Service
 - h) UDP Service
2. Servicios (tareas)
 - a) Measuring Task
 - b) Motors Task
 - c) UDP Server Task

Los componentes de software y servicios del joystick son:

1. Componentes de software
 - a) main-core
 - b) Measuring State
 - c) Robot Position State
 - d) UDP Service
2. Servicios (tareas)
 - a) Display Task
 - b) Joystick Task
 - c) UDP Server Task

3.2. Implementación de los módulos

Los macro componentes de hardware presentes en la arquitectura de la versión v2.0 son el robot y el joystick, que constituyen dos sistemas embebidos independientes implementados con dos microcontroladores ESP32, integrados entre sí por medio de una red **Wi-Fi** y una comunicación UDP. La arquitectura de **hardware** del sistema esta compuesta por los siguientes módulos:

- En el robot:
 - Control de los motores DC.
 - Control de los sensores de medición (DHT11, BMP280 y fotoresistor).
 - Gestión de la comunicación inalámbrica vía Wi-Fi (en la versión v2.0).
- En el joystick:

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.

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, este soporta múltiples características [35] 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 [36]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [37].

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.

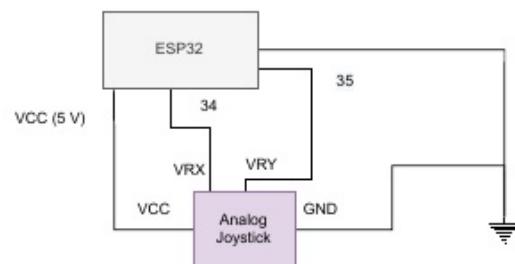


FIGURA 3.3. Conexiónado joystick.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [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

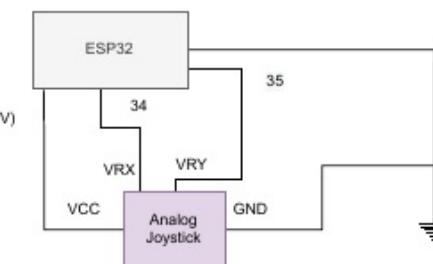


FIGURA 3.3. Conexiónado joystick.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [38]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [39].

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 ecuación 3.1.

$$NI \equiv \frac{MR - l}{MR} \times 100 \quad (3.1)$$

Donde:

- `NI`: Nivel de iluminación, es el valor del nivel de luminosidad calculado.
- `l`: lectura analógica del valor del fotoresistor.
- `MR`: lectura máxima, es el máximo valor analógico posible de ser entregado por el fotoresistor, en este caso 2050.

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

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

3.2. Implementación de los módulos

21

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

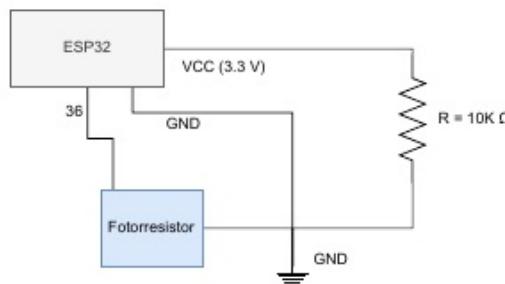


FIGURA 3.4. Conexionado fotorresistor.

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.

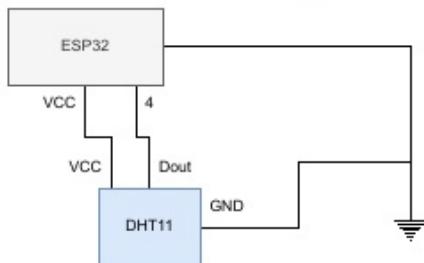


FIGURA 3.5. Circuito del conexionado DHT11.

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 MeasuringService en la tarea MeasuringTask. Sus lecturas son guardadas en el MeasuringState. A continuación, se puede apreciar el conexionado del prototipo en la figura 3.6.

3.2. Implementación de los módulos

21

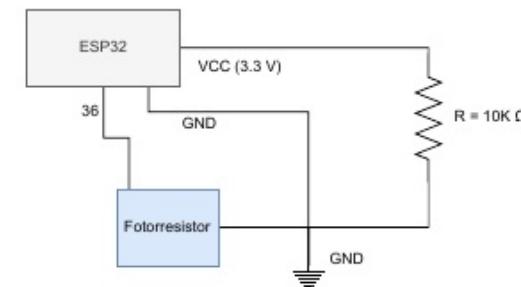


FIGURA 3.4. Conexionado fotorresistor.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [38]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [40].

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 [23] 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.

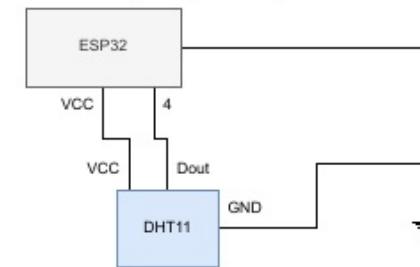


FIGURA 3.5. Circuito del conexionado DHT11.

El desarrollo de este módulo se basó en el ejemplo provisto por la biblioteca ESP-IDF-Lib [41]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [42].

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 MeasuringService en la tarea MeasuringTask. Sus lecturas son guardadas en el MeasuringState. A continuación, se puede apreciar el conexionado del prototipo en la figura 3.6.

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.

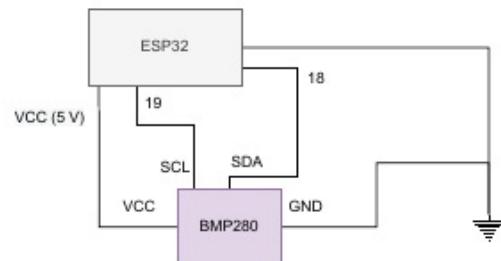


FIGURA 3.6. Conexiónado BMP280.

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.

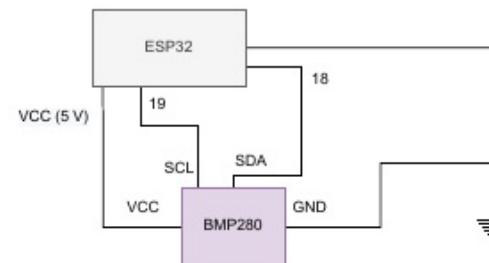


FIGURA 3.6. Conexiónado BMP280.

El desarrollo de este módulo se basó en el ejemplo provisto por la biblioteca ESP-IDF-Lib [43]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [44].

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 [45], 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) [46] que por medio de la configuración de sus unidades y del *duty cycle* [47] 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 diagrama de la figura 3.7 puede apreciarse el conexionado lógico para el control de los motores con los componentes mencionados.

3.2. Implementación de los módulos

23

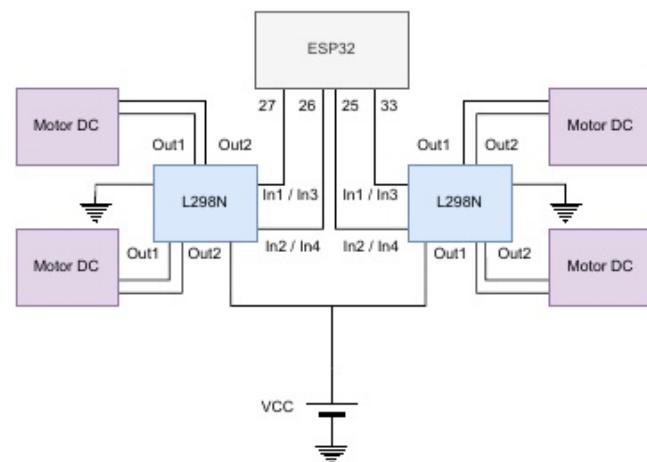


FIGURA 3.7. Conexiónado motores.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [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.

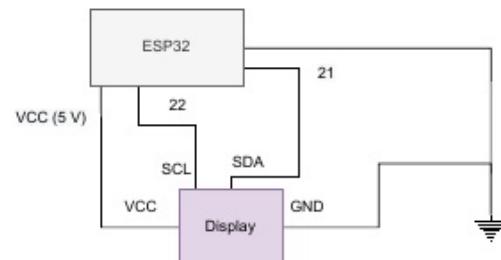


FIGURA 3.8. Conexiónado display.

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].

3.2. Implementación de los módulos

23

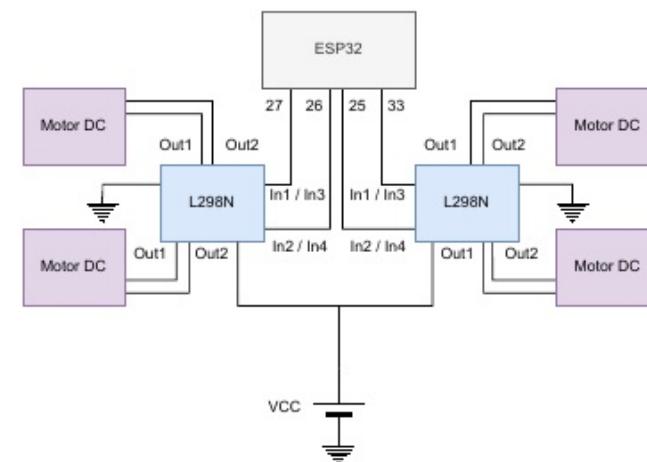


FIGURA 3.7. Conexiónado motores.

El desarrollo de este módulo se basó en el ejemplo provisto por Espressif [48]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [49].

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.

En el diagrama de la figura 3.8 puede apreciarse el conexiónado lógico para el control del display.

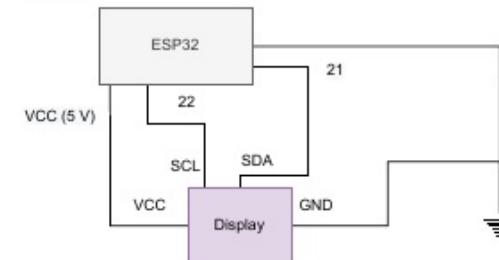


FIGURA 3.8. Conexiónado display.

El desarrollo de este módulo se basó en el ejemplo provisto en el enlace [50]. El código fuente del prototipo realizado puede apreciarse en el siguiente enlace [51].

3.3. Arquitectura de hardware

3.3.1. Ensamblado final del producto v2.0

En las imágenes de la figura 3.9 se pueden apreciar las diferentes perspectivas del robot.



FIGURA 3.9. Hardware del robot.

En las figuras 3.10 y 3.11 se puede apreciar en mayor profundidad el hardware del robot y del joystick con sus subcomponentes.

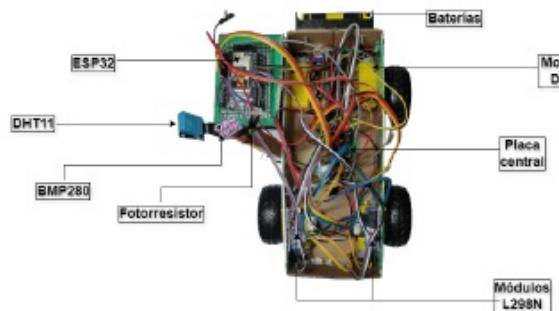


FIGURA 3.10. Detalles del hardware del robot.

3.3. Arquitectura de hardware

3.3.1. Ensamblado final del producto v2.0

En las imágenes de la figura 3.9, se pueden apreciar las diferentes perspectivas del robot.



FIGURA 3.9. Hardware del robot.

En las figuras 3.10 y 3.11 se puede apreciar en mayor profundidad el hardware del robot y del joystick con sus subcomponentes.

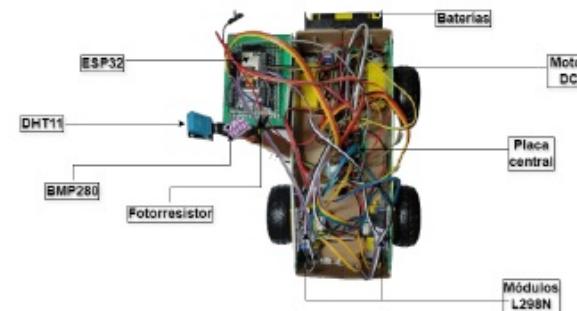


FIGURA 3.10. Detalles del hardware del robot.

3.3. Arquitectura de hardware

25

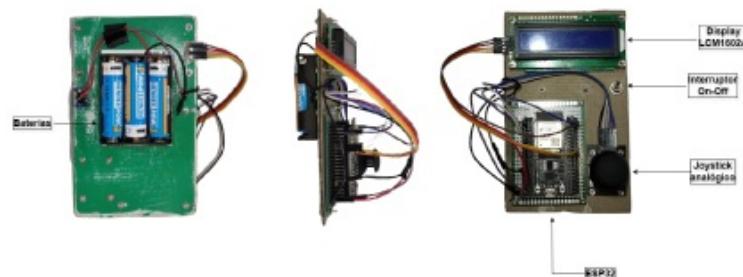


FIGURA 3.11. Detalles del hardware del joystick.

3.3.2. Conexionado lógico

En las figuras 3.12 y 3.13 se pueden apreciar los conexionados lógicos del robot y del joystick respectivamente.

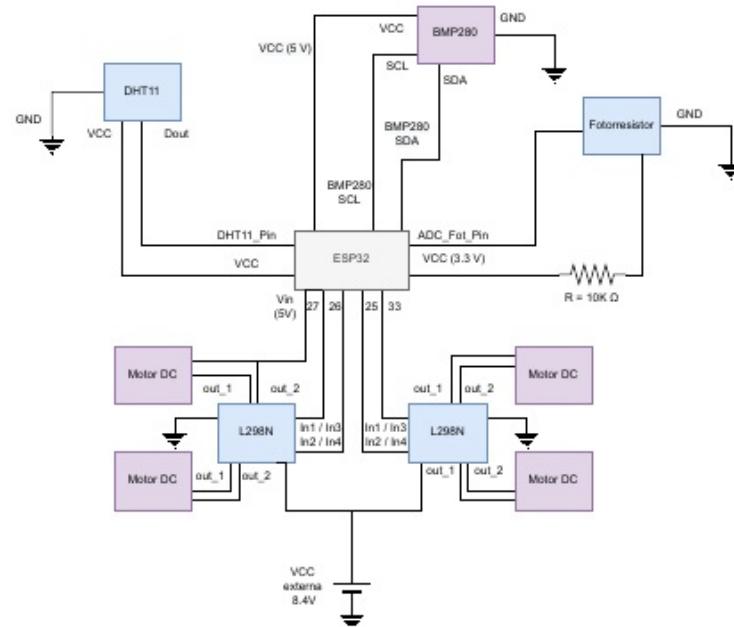


FIGURA 3.12. Conexionado del robot.

3.3. Arquitectura de hardware

25

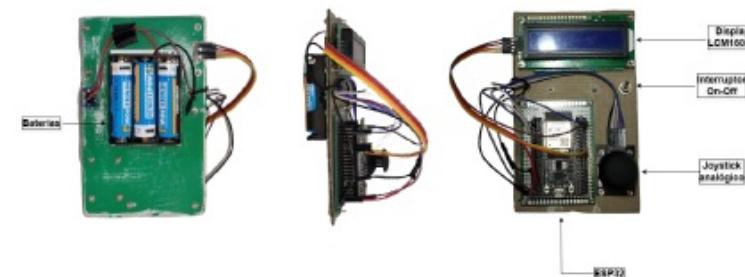


FIGURA 3.11. Detalles del hardware del joystick.

3.3.2. Conexionado lógico

En las figuras 3.12 y 3.13 se pueden apreciar los conexionados lógicos del robot y del joystick, respectivamente.

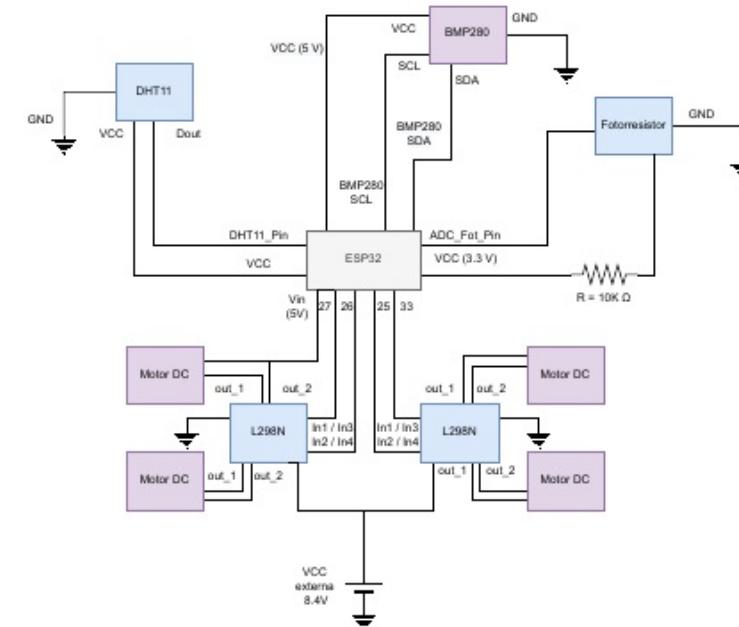


FIGURA 3.12. Conexionado del robot.

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.

3.4. Plataforma de desarrollo y ciclo de CI/CD

Durante el ciclo de desarrollo, se utilizaron las herramientas descritas en el capítulo anterior, y para cada prototipo se creó una imagen Docker, extendiendo la de espressif/idf [52]. 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 los comandos git commit y git 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 los 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.

```

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.

```

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.

Luego de realizar *commit* y *push* de los cambios locales se pueden apreciar en la imagen 3.16 el listado de las versiones en Github.

3.4. Plataforma de desarrollo y ciclo de CI/CD

29

The screenshot shows the Google Cloud Build history interface. At the top, there are navigation tabs for 'Google Cloud' and 'CloudBuild'. Below that is a search bar and a 'Search' button. The main area is titled 'Build history' and shows a table of build logs. The columns are: Status, Id, Source, Ref, Commit, Created, and Duration. The table lists many builds, mostly successful (green), with some failures (red). The commits are from a GitHub repository named 'cese_proyecto_especializacion'. The commit IDs range from '49d2e76' to '4edc0d0'. Most commits were made on September 24, 2024, with durations between 2 and 10 seconds.

FIGURA 3.16. CloudBuild.

3.4. Plataforma de desarrollo y ciclo de CI/CD

29

The screenshot shows a GitHub repository page for 'kronleuchter85/cese_proyecto_especializacion'. The top navigation bar includes 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', and 'Insights'. Below that is a 'Commits' section. It displays a list of commits grouped by date. The commits are from a user named 'kronleuchter85'. The first group is for 'Commits on Sep 21, 2024', containing two commits related to adding test coverage for the 'motor_service'. The second group is for 'Commits on Sep 20, 2024', containing four commits related to adding coverage for 'wifi_service', 'robot_position_state', 'adc service', and 'testing'. The third group is for 'Commits on Sep 19, 2024', containing three commits related to fixing the build and completing tests for 'display service'.

FIGURA 3.16. Listado de commits en Github.

En la imagen 3.17 se pueden apreciar los diferentes builds disparados en Cloud Build referenciando los commits de Github.

Commits

- Commits on Sep 21, 2024**
 - agregando cobertura de test del motor_service
 - kronlechter85 committed 11 minutes ago ✓ 1/1
 - agregando cobertura del wifi_service. Se modifica el código productivo haciendo que el handler deje de ser estático y todos los headers incluidos y las constantes macros definidas son enviadas del wifi.
 - kronlechter85 committed 6 hours ago ✓ 1/1
- Commits on Sep 20, 2024**
 - agregando cobertura para test_robot_position_state
 - kronlechter85 committed yesterday ✓ 1/1
 - agregando tests de cobertura para adc service
 - kronlechter85 committed yesterday ✓ 1/1
 - agregando tests de cobertura para adc service
 - kronlechter85 committed yesterday ✘ 0/1
 - agregando cobertura de testing
 - kronlechter85 committed 2 days ago ✓ 1/1
- Commits on Sep 19, 2024**
 - fixing build
 - kronlechter85 committed 2 days ago ✓ 1/1
 - completando tests de display service
 - kronlechter85 committed 2 days ago ✘ 0/1
 - agregando tests de display service
 - kronlechter85 committed 2 days ago ✓ 1/1

FIGURA 3.17. Github.

Build history						
Regexp	Status	Build	Source	Ref	Commit	Created
global (non-regional)	Success	86923m76	kronlechter85/cese_proyecto_especializacion	main	34f6a7d1	9/21/24, 11:49 AM
	Success	4f62e49	kronlechter85/cese_proyecto_especializacion	main	30137fe1	9/21/24, 9:58 AM
	Success	b60a7m77	kronlechter85/cese_proyecto_especializacion	main	34b0321	9/20/24, 7:08 AM
	Success	4b274ed1	kronlechter85/cese_proyecto_especializacion	main	1964511	9/20/24, 7:20 AM
	Success	12b56e8b	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/20/24, 7:22 AM
	Success	3c90f73	kronlechter85/cese_proyecto_especializacion	main	31b6c1	9/19/24, 7:18 PM
	Success	b61a015	kronlechter85/cese_proyecto_especializacion	main	34f6a7d2	9/19/24, 9:18 AM
	Failure	411aa021	kronlechter85/cese_proyecto_especializacion	main	30137fe2	9/19/24, 9:19 AM
	Success	65394c1	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/19/24, 9:03 AM
	Success	15950419	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/19/24, 9:03 AM
	Success	4b26114	kronlechter85/cese_proyecto_especializacion	main	3a6d1a1	9/18/24, 9:46 AM
	Failure	15950419	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/18/24, 9:47 AM
	Success	4f62e49	kronlechter85/cese_proyecto_especializacion	main	30137fe3	9/18/24, 9:27 AM
	Failure	42466031	kronlechter85/cese_proyecto_especializacion	main	2063394	9/18/24, 9:28 AM
	Failure	39198278	kronlechter85/cese_proyecto_especializacion	main	411aa02	9/18/24, 9:28 AM
	Failure	3161973	kronlechter85/cese_proyecto_especializacion	main	1964512	9/18/24, 9:28 AM
	Success	4255904	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/17/24, 9:15 AM
	Success	4200397	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 4:48 PM
	Success	47750269	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 4:49 PM
	Success	4350481	kronlechter85/cese_proyecto_especializacion	main	7039a71	9/16/24, 6:30 PM
	Success	35058613	kronlechter85/cese_proyecto_especializacion	main	4ed7a71	9/16/24, 2:29 PM
	Failure	4f7a5988	kronlechter85/cese_proyecto_especializacion	main	4f83601	9/16/24, 2:29 PM
	Failure	4f19273	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Success	4b26114	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Failure	17224027	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Success	430307cc	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Failure	4200398	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Success	48920267	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Failure	4596801	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Success	4300498	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/16/24, 2:29 PM
	Success	3504494	kronlechter85/cese_proyecto_especializacion	main	2063395	9/16/24, 1:58 PM
	Failure	4200398	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/15/24, 9:24 PM
	Failure	48924077	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/15/24, 9:14 PM
	Success	4729432	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/15/24, 8:38 PM
	Failure	4621682	kronlechter85/cese_proyecto_especializacion	main	2063396	9/15/24, 8:32 PM
	Failure	47247003	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/15/24, 8:48 PM
	Failure	2058061	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/15/24, 8:49 PM
	Failure	4ac83d9	kronlechter85/cese_proyecto_especializacion	main	3ed521	9/15/24, 8:49 PM

FIGURA 3.17. Listado de builds en Google CloudBuild.

Finalmente en la imagen 3.18 se pueden apreciar las imágenes Docker versionadas y almacenadas en Artifact Registry.

3.5. Reportes de ejecución y cobertura de testing unitario

31

The screenshot shows a Google Cloud interface for the Artifact Registry. The URL is [https://console.cloud.google.com/artifact-registry/repositories/poc-esp32-integracion/versions](#). The page title is "Digests for poc-esp32-integracion". It displays a table of Docker image versions with columns: Name, Tag, Created, and Updated. The table lists 12 entries, each with a unique name and tag, all created and updated just now.

Name	Tag	Created	Updated
01ceb63ccb4e3	04f1a2724ccb0cc70d54a41209d1648fb5f3	1 minute ago	1 minute ago
01ceb63ccb4e3	201974e33147f402d6ebc7b5a5a6fb5d47274dd94	6 hours ago	6 hours ago
01ceb63ccb4e3	546a3f1e7a707e33edfb53a5a6fb5d40839c6	1 day ago	1 day ago
01ceb63ccb4e3	19944015411efc2081b7690633739344a1ff1d90	1 day ago	1 day ago
01ceb63ccb4e3	f18e9c141cb6b37cafbaw655a4e15a79e6e5db	2 days ago	2 days ago
01ceb63ccb4e3	c47597025202520251e9fae024fb8fbcd124a	2 days ago	2 days ago
01ceb63ccb4e3	dd4d1ac0753745a5ab0b193666c7b9eb5c119a8	2 days ago	2 days ago
01ceb63ccb4e3	1e6d108e72f5773ae51ffad5f191d1f727c4	3 days ago	3 days ago
01ceb63ccb4e3	d6fb1449476f7279111eece10d99d1f78776fb1	4 days ago	4 days ago
01ceb63ccb4e3	f4d931c4ffbe4c195a1aa0e08417d1f520cb4d	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.

3.5. Reportes de ejecución y cobertura de testing unitario

31

The screenshot shows a Google Cloud interface for the Artifact Registry. The URL is [https://console.cloud.google.com/artifact-registry/repositories/poc-esp32-integracion/versions](#). The page title is "Digests for poc-esp32-integracion". It displays a table of Docker image versions with columns: Name, Tag, Created, and Updated. The table lists 12 entries, each with a unique name and tag, all created and updated just now.

Name	Tag	Created	Updated
01ceb63ccb4e3	04f1a2724ccb0cc70d54a41209d1648fb5f3	1 minute ago	1 minute ago
01ceb63ccb4e3	201974e33147f402d6ebc7b5a5a6fb5d47274dd94	6 hours ago	6 hours ago
01ceb63ccb4e3	546a3f1e7a707e33edfb53a5a6fb5d40839c6	1 day ago	1 day ago
01ceb63ccb4e3	19944015411efc2081b7690633739344a1ff1d90	1 day ago	1 day ago
01ceb63ccb4e3	f18e9c141cb6b37cafbaw655a4e15a79e6e5db	2 days ago	2 days ago
01ceb63ccb4e3	c47597025202520251e9fae024fb8fbcd124a	2 days ago	2 days ago
01ceb63ccb4e3	dd4d1ac0753745a5ab0b193666c7b9eb5c119a8	2 days ago	2 days ago
01ceb63ccb4e3	1e6d108e72f5773ae51ffad5f191d1f727c4	3 days ago	3 days ago
01ceb63ccb4e3	d6fb1449476f7279111eece10d99d1f78776fb1	4 days ago	4 days ago
01ceb63ccb4e3	f4d931c4ffbe4c195a1aa0e08417d1f520cb4d	5 days ago	5 days ago

FIGURA 3.18. Listado de versiones de imágenes docker en Google 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 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.

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.

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 cada 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.

En las siguientes secciones, 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 los videos [53] y [54] puede apreciarse el funcionamiento del módulo de control de joystick.

4.2.2. Verificación del módulo de control del display

Se verificó visualmente que el display representaba 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.

4.2.2. Verificación del módulo de control del display

Se verificó visualmente que el display mostraba los caracteres programados en la prueba de concepto con una intensidad de luz aceptable para poder leerlos apropiadamente. En las figuras 4.1, 4.8, 4.9 y 4.10 puede apreciarse el funcionamiento del módulo del display bajo diferentes condiciones de luminosidad.

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. En los videos [53] y [54] puede apreciarse el funcionamiento del módulo de control de los motores.

4.2.4. Verificación del módulo de medición de temperatura y humedad

Se verificó visualmente que los valores obtenidos por el sensor DHT11 coincidían con los esperados en relación a la temperatura en el interior del lugar de experimentación y que la humedad detectada se aproximara a los valores reportados por Google. En las figuras 4.7, 4.7, 4.7 y 4.7 puede apreciarse el funcionamiento del módulo de medición de temperatura y humedad.

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. En las figuras 4.7 y 4.7 puede apreciarse el funcionamiento del módulo de presión atmosférica.

4.2.6. Verificación del módulo de medición de luminosidad

Se verificó visualmente que los valores obtenidos del fotoresistor, una vez transformados a valores absolutos porcentuales, reflejaban el nivel de luminosidad ambiental del interior del lugar de experimentación. En las figuras 4.8, 4.9 y 4.10 puede apreciarse el funcionamiento del módulo de luminosidad.

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. Posteriormente, se incorporó el servicio de comunicaciones UDP en el robot, y desde el programa cliente se enviaban las acciones que representaban las direcciones del movimiento (FORWARD, BACKWARD, LEFT, RIGHT). 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 activó el desplazamiento en cada una de sus direcciones. El robot se desplazó correctamente en respuesta a cada comando recibido. En los videos [55], [56] y [57] puede apreciarse el funcionamiento de la comunicación UDP sobre Wi-Fi entre el robot y el joystick.

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.

4.3. Pruebas funcionales y validación del producto

35

- 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 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 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:

4.3. Pruebas funcionales y validación del producto

35

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 lumi-nosidad ambiental se utilizó la observación visual.

Las pruebas realizadas fueron las siguientes:

- Prueba y validación del módulo de visualización de display.
- 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 [58] y en la oscuridad [59].

- 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:

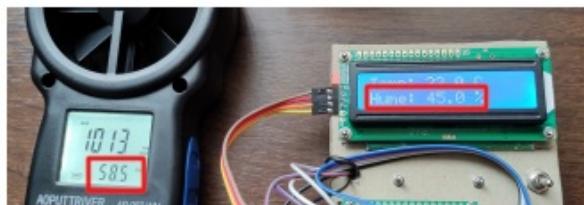


FIGURA 4.2. Medición de humedad en el interior.

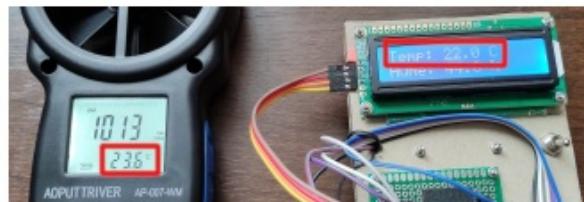


FIGURA 4.3. Medición de temperatura en el interior.



FIGURA 4.4. Medición de humedad en el exterior.

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

Se compararon los valores medidas 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:

- 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:

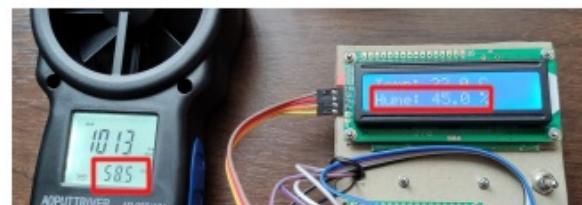


FIGURA 4.2. Medición de humedad en el interior.



FIGURA 4.3. Medición de temperatura en el interior.

4.3. Pruebas funcionales y validación del producto

37

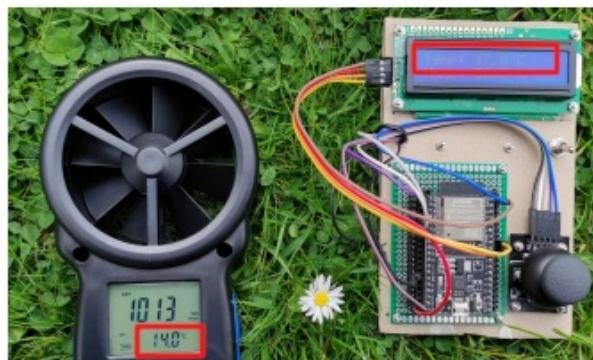


FIGURA 4.5. Medición de temperatura en el exterior.

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.

4.3. Pruebas funcionales y validación del producto

37



FIGURA 4.4. Medición de humedad en el exterior.



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 [58].

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 4.7. 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 fotorresistor 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.



FIGURA 4.6. Medición de presión atmosférica en el interior.

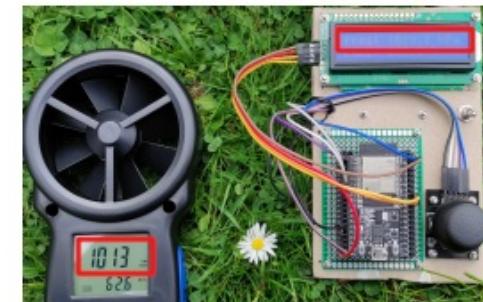


FIGURA 4.7. Medición de presión atmosférica en el exterior.

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 [58].

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 fotorresistor 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.

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.

4.3. Pruebas funcionales y validación del producto

39



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



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 [58].

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:

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 - Comunicación 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].

- 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 [60] y [61] evidenciando la demostración de este experimento.

4.4. Videos del producto durante el 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 [62].
- Demo - Comunicación Wi-Fi [55].
- Demo - Control de movimiento de las ruedas [60].
- Demo - Medición y visualización de parámetros ambientales [58].
- Demo - Control de desplazamiento en un circuito [61].
- Demo - Visualización del Display en la oscuridad [59].

4.4.2. Videos durante el prototipado y ensamblado del robot

- Prototipado Robot v1 - Ensamblado (1) [53].
- Prototipado Robot v1 - Ensamblado (2) [54].
- Prototipado Robot v1 - Ensamblado (3) [63].
- Prototipado Robot v1 - Ensamblado (4) [64].
- Prototipado Robot v2 - Comunicación Joystick Robot (1) [56].
- Prototipado Robot v2 - Comunicación Joystick Robot (2) [57].
- Prototipado Desplazamiento (alimentación USB) [65].
- Prototipado Desplazamiento (alimentación por pilas) [66].

4.5. Documentación del producto

Se desarrolló la documentación del producto compuesta de los siguientes entregables

- Documentación técnica [67].

Capítulo 5

Conclusiones

Conclusiones del trabajo...

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puede seguir la planificación original (cronograma incluido)?
- ¿Se manifestó alguno de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.1. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

- Manual de usuario [68].

Bibliografía

- [1] Latam Mining. *Robots y minería: Gobierno argentino quiere implementarlos.* URL: <https://www.latam-mining.com/robots-y-mineria-gobierno-argentino-quiere-implementarlos/>.
- [2] Diario de Cuyo. *Gobierno pone la mira en el desarrollo de robots para la actividad minera.* URL: <https://www.diariodecuyo.com.ar/politica/Gobierno-pone-la-mira-en-el-desarrollo-de-robots-para-la-actividad-minera-20200202-0052.html>.
- [3] Universidad Nacional de San Juan. *Robots en la minería.* URL: http://www.unsj.edu.ar/home/noticias_detalles/4810/1.
- [4] Ing. Nelson Dario García Hurtado e Ing. Melvin Andrés González Pino. *Robot de exploración terrestre Geobot.* URL: https://www.unipamplona.edu.co/unipamplona/portallG/home_40/recursos/01_general/revista_1/09102011/v01_09.pdf.
- [5] Ing. Hernán L. Helguero Velásquez1 e Ing. Rubén Medinaceli Tórrez. *Robot Minero: Sistema Detector de Gases utilizando Sensores en Tiempo Real MIN – SIS 1.0 SDG-STR.* URL: http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S2519-53522020000100003.
- [6] Boston Dynamics. *Spot.* URL: <https://www.bostondynamics.com/products/spot>.
- [7] Waygate Technologies. *BIKE - An advanced crawler robot for remote visual inspection.* URL: <https://www.bakerhughes.com/waygate-technologies/robotic-inspection/bike>.
- [8] Espressif. *ESP32.* URL: <https://www.espressif.com/en/products/socs/esp32>.
- [9] Espressif. *ESP32-WROOM-32D Datasheet.* URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.
- [10] Mouser. *DHT11 datasheet.* URL: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Datasheet-Translated-Version-1143054.pdf>.
- [11] Bosch. *BMP280 datasheet.* URL: <https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf>.
- [12] Handson Technology. *PS2 Joy Stick for Arduino/Raspberry.* URL: <http://www.handsontec.com/dataspecs/accessory/PS2-Joystick.pdf>.
- [13] Espressif. *Analog to Digital Converter (ADC).* URL: <https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html>.
- [14] Handson Technology. *I2C Serial Interface 1602 LCD Module.* URL: http://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf.
- [15] Adafruit. *DC Gearbox Motor - TT Motor -200RPM - 3 to 6VDC.* URL: https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf.

Capítulo 5

Conclusiones

Conclusiones del trabajo...

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puede seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.1. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

- [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 ADCI 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.

- [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 despazamiento 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/CcBgvoKjLB0>.
- [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/w9lOoE-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>.

Bibliografía

- [1] Latam Mining. *Robots y minería: Gobierno argentino quiere implementarlos*. URL: <https://www.latam-mining.com/robots-y-mineria-gobierno-argentino-quiere-implementarlos>.
- [2] Diario de Cuyo. *Gobierno pone la mira en el desarrollo de robots para la actividad minera*. URL: <https://www.diariodecuyo.com.ar/politica/Gobierno-pone-la-mira-en-el-desarrollo-de-robots-para-la-actividad-minera-20200202-0052.html>.
- [3] Universidad Nacional de San Juan. *Robots en la minería*. URL: http://www.unsj.edu.ar/home/noticias_detalles/4810/1.
- [4] Ing. Nelson Dario García Hurtado e Ing. Melvin Andrés González Pino. *Robot de exploración terrestre Geobot*. URL: https://www.unipamplona.edu.co/unipamplona/portallG/home_40/recursos/01_general/revista_1/09102011/v01_09.pdf.
- [5] Ing. Hernán L. Helguero Velásquez1 e Ing. Rubén Medinaceli Tórrez. *Robot Miner: Sistema Detector de Gases utilizando Sensores en Tiempo Real MIN – SIS 1.0 SDG-STR*. URL: http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S2519-53522020000100003.
- [6] Boston Dynamics. *Spot*. URL: <https://www.bostondynamics.com/products/spot>.
- [7] Waygate Technologies. *BIKE - An advanced crawler robot for remote visual inspection*. URL: <https://www.bakerhughes.com/waygate-technologies/robotic-inspection/bike>.
- [8] Espressif. *ESP32*. URL: <https://www.espressif.com/en/products/socs/esp32>.
- [9] Espressif. *ESP32-WROOM-32D Datasheet*. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.
- [10] Mouser. *DHT11 datasheet*. URL: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Datasheet-Translated-Version-1143054.pdf>.
- [11] Bosch. *BMP280 datasheet*. URL: <https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf>.
- [12] Handson Technology. *PS2 Joy Stick for Arduino/Raspberry*. URL: <http://www.handsontec.com/datasheets/accessory/PS2-Joystick.pdf>.
- [13] Espressif. *Analog to Digital Converter (ADC)*. URL: <https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html>.
- [14] Handson Technology. *I2C Serial Interface 1602 LCD Module*. URL: https://www.handsontec.com/datasheets/module/I2C_1602_LCD.pdf.
- [15] Adafruit. *DC Gearbox Motor - TT Motor -200RPM - 3 to 6VDC*. URL: https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf.

- [60] Gonzalo Carreno. *Robot de exploracion ambiental - Prototipado Comunicacion Joystick Robot (2)*. URL: <https://youtu.be/jiiSheyu95w>.
- [61] Gonzalo Carreno. *Robot de exploracion ambiental - Prototipado Desplazamiento (alimentacion USB)*. URL: https://youtu.be/_w8qdNWC-DQ.
- [62] Gonzalo Carreno. *Robot de exploracion ambiental - Prototipado Desplazamiento (alimentacion por pilas)*. URL: <https://youtu.be/-MxMXKztfIU>.
- [63] Gonzalo Carreno. *Robot de exploracion ambiental - Documentacion Tecnica*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/blob/main/docs/Documentacion-Tecnica.pdf.
- [64] 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.

- [16] Handson Technology. *L298N Driver Module Datasheet*. URL: <https://www.handsontec.com/datasheets/L298N%20Motor%20Driver.pdf>.
- [17] EEMB. *Li-Ion batteries 18650 3000 mAh*. URL: <http://www.kosmodrom.com.ua/pdf/LIR18650-3000mah.pdf>.
- [18] Farnell. *Li-Ion batteries AA 2600 mAh*. URL: <https://www.farnell.com/datasheets/3195148.pdf>.
- [19] AOPUTTRIVER. *Anemómetro digital AOPUTTRIVER AP-007-WM*. URL: <https://manuals.plus/m/a30ffaa3ac8fd2cf06cb555977c3af166bc当地18d1ee49a56ea6f10c66dd4b.pdf>.
- [20] CMake. *CMake*. URL: <https://cmake.org/>.
- [21] CMake. *Ninja*. URL: <https://cmake.org/cmake/help/latest/generator/Ninja.html>.
- [22] Espressif. *ESP-IDF Programming Guide | Get Started*. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/>.
- [23] Readthedocs by Ruslan V. Uss. *ESP-IDF Components library*. URL: <https://esp-idf-lib.readthedocs.io/en/latest/>.
- [24] Espressif Programming Guide. *ESP-IDF Get Started*. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html>.
- [25] Docker. *Docker*. URL: <https://docker.com/>.
- [26] ThrowTheSwitch. *ThrowTheSwitch - Ceedling*. URL: <https://github.com/ThrowTheSwitch/Ceedling>.
- [27] ThrowTheSwitch. *ThrowTheSwitch - CMock*. URL: <https://github.com/ThrowTheSwitch/CMock>.
- [28] ThrowTheSwitch. *ThrowTheSwitch - Unity Test*. URL: <https://github.com/ThrowTheSwitch/Unity>.
- [29] ThrowTheSwitch. *ThrowTheSwitch - Ceedling/GCov*. URL: <https://github.com/ThrowTheSwitch/Ceedling/blob/master/plugins/gcov/README.md>.
- [30] Github. *Github*. URL: <https://github.com/>.
- [31] Google Cloud Platform. *Google Cloud Build*. URL: <https://cloud.google.com/build>.
- [32] Google Cloud Platform. *Google Artifact Registry*. URL: <https://cloud.google.com/artifact-registry>.
- [33] Visualstudio. *Visualstudio Code*. URL: <https://code.visualstudio.com/>.
- [34] Ubuntu. *Ubuntu*. URL: <https://ubuntu.com/>.
- [35] Espressif. *ESP-IDF WiFi*. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/wifi.html>.
- [36] Espressif. *ESP-IDF - Wi-Fi SoftAP Example*. URL: https://github.com/espressif/esp-idf/tree/v4.4/examples/wifi/getting_started/softAP.
- [37] Gonzalo Carreno. *POC ESP32-WiFi v4.4*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-wifi-ap-v4.4.
- [38] Espressif. *ESP-IDF ADC1 Example*. URL: <https://github.com/espressif/esp-idf/tree/v4.0.3/examples/peripherals/adc>.
- [39] Gonzalo Carreno. *POC ESP32-joystick*. URL: https://github.com/kronleuchter85/cese_proyecto_especializacion/tree/main/pocs/esp32-joystick.