

## 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. Anemometro digital AOPUTTRIVER AP-007-WM. . . . .	12
2.18. Proceso de desarrollo utilizando ESP-IDF <sup>1</sup> . . . . .	13
3.1. Arquitectura global. . . . .	16
3.2. Arquitectura global. . . . .	17
3.3. Conexión joystick. . . . .	20
3.4. Conexión fotorresistor. . . . .	21
3.5. Circuito del conexión DHT11. . . . .	21
3.6. Conexión BMP280. . . . .	22
3.7. Conexión motores. . . . .	23
3.8. Conexión 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. Conexión del robot. . . . .	25
3.13. Conexión del joystick. . . . .	26
3.14. Conexión físico del robot. . . . .	26
3.15. Plataforma de CI/CD utilizada. . . . .	27
3.16. Ejecución de tests por consola. . . . .	28
3.17. Listado de <i>commits</i> en <b>Github</b> . . . . .	29
3.18. Listado de <i>builds</i> en Google CloudBuild. . . . .	30
3.19. Listado de versiones de imágenes <b>doker</b> en Google ArtifactRegistry. . . . .	31
3.20. Reportes de testing por consola. . . . .	32
3.21. Reportes de testing web. . . . .	32
4.1. Visualización del display en la oscuridad. . . . .	35
4.2. Medición de humedad 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. Anemometro digital AOPUTTRIVER AP-007-WM. . . . .	12
2.18. Proceso de desarrollo utilizando ESP-IDF <sup>1</sup> . . . . .	13
3.1. Arquitectura global. . . . .	16
3.2. Arquitectura global. . . . .	17
3.3. Conexión joystick. . . . .	20
3.4. Conexión fotorresistor. . . . .	21
3.5. Circuito del conexión DHT11. . . . .	21
3.6. Conexión BMP280. . . . .	22
3.7. Conexión motores. . . . .	23
3.8. Conexión 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. Conexión del robot. . . . .	25
3.13. Conexión del joystick. . . . .	26
3.14. Conexión físico del robot. . . . .	26
3.15. Plataforma de CI/CD utilizada. . . . .	27
3.16. Ejecución de tests por consola. . . . .	28
3.17. Listado de <i>commits</i> en <b>GitHub</b> . . . . .	29
3.18. Listado de <i>builds</i> en Google CloudBuild. . . . .	30
3.19. Listado de versiones de imágenes <b>Docker</b> en Google ArtifactRegistry. . . . .	31
3.20. Reportes de testing por consola. . . . .	32
3.21. Reportes de testing web. . . . .	32
4.1. Visualización del display en la oscuridad. . . . .	35
4.2. Medición de humedad en el interior. . . . .	36

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.

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.

### 3.4. Plataforma de desarrollo y ciclo de CI/CD

Durante el ciclo de desarrollo, se utilizó la infraestructura de CI formada por las herramientas descritas en el capítulo anterior. En la figura 3.15 se puede apreciar su arquitectura.

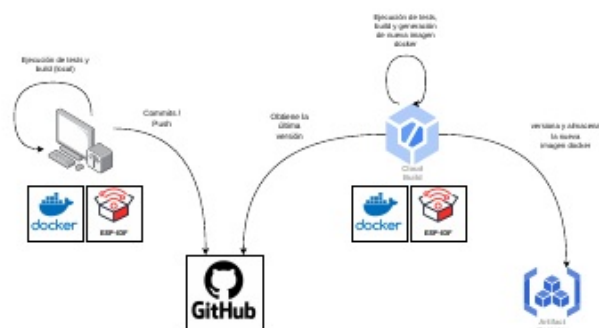


FIGURA 3.15. Plataforma de CI/CD utilizada.

Para cada prototipo desarrollado 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 3.16, se puede apreciar la salida por consola tras la ejecución de los tests unitarios y construcción de la imagen Docker de manera local.

### 3.4. Plataforma de desarrollo y ciclo de CI/CD

Durante el ciclo de desarrollo, se utilizó la infraestructura de CI formada por las herramientas descritas en el capítulo anterior. En la figura 3.15 se puede apreciar su arquitectura.

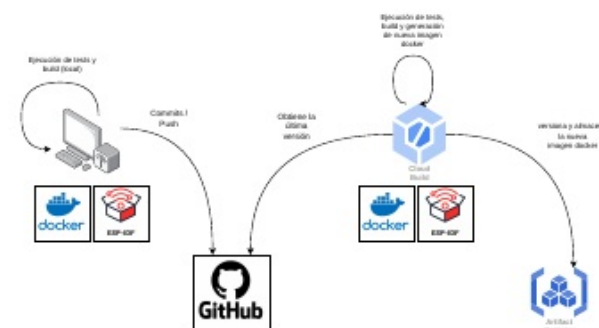


FIGURA 3.15. Plataforma de CI/CD utilizada.

Para cada prototipo desarrollado 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 3.16, se puede apreciar la salida por consola tras la ejecución de los tests unitarios y construcción de la imagen Docker de manera local.

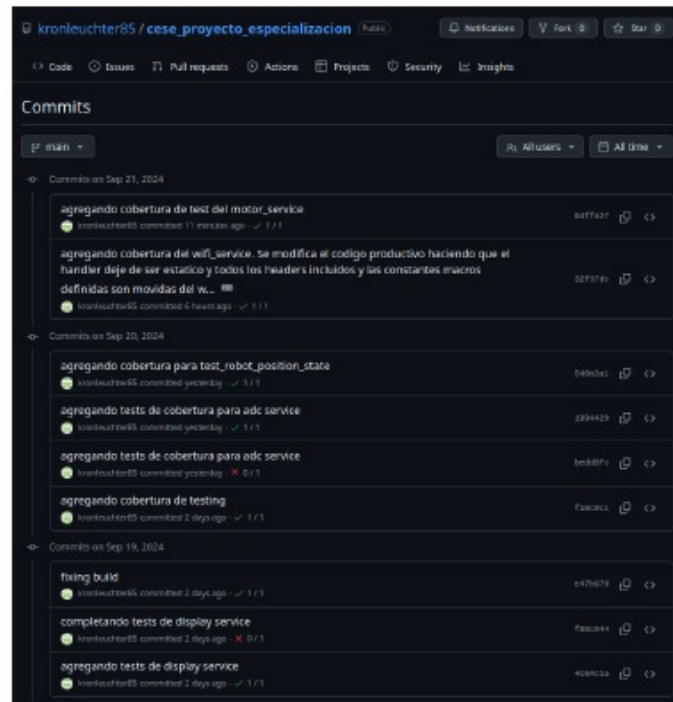


FIGURA 3.17. Listado de commits en Github.

En la figura 3.18, se pueden apreciar los diferentes builds disparados en Cloud Build referenciando los commits de GitHub.

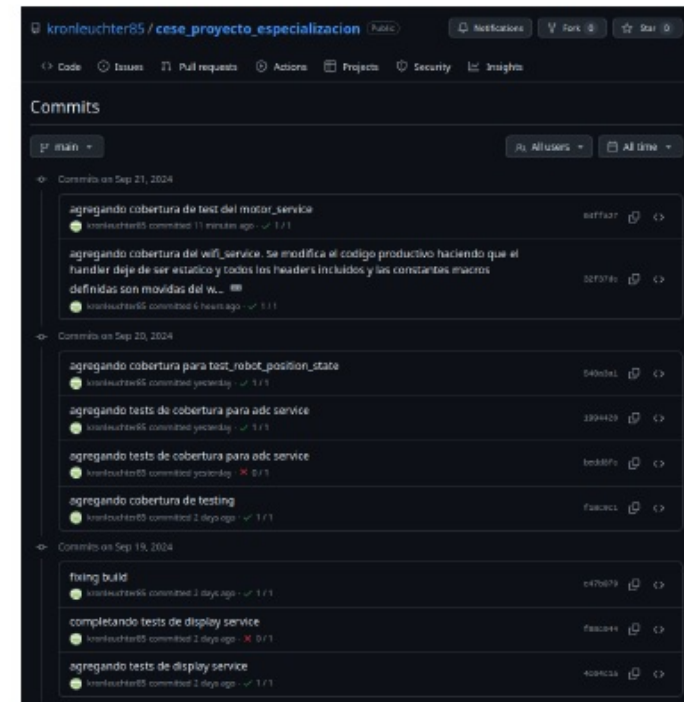


FIGURA 3.17. Listado de commits en Github.

En la figura 3.18, se pueden apreciar los diferentes builds disparados en Cloud Build referenciando los commits de GitHub.

## 3.5. Reportes de ejecución y cobertura de testing unitario

31

Google Cloud | oese-robot | Search (/) for resources, docs, products, and more | Search

Digests for poc-esp32-integration | DELETE | SETUP INSTRUCTIONS

europa-docker.pkg.dev > oese-robot > poc-esp32-integration > poc-esp32-integration

VERSIONS | FILES

Filter: Enter property name or value

<input type="checkbox"/>	Name	Tags	Created	Updated
<input type="checkbox"/>	01ceb62cb4e0	04f1a2724ceb62cb4e0354e41209d716488d5f53	latest	1 minute ago
<input type="checkbox"/>	56e514e0229e	22f37bd5331c78f9c2a6b7c7a5a6d8e733d95d		6 hours ago
<input type="checkbox"/>	843b6935c7b7	548a3e1fa7c7e5e8fbd52c4aef0548c325c6		1 day ago
<input type="checkbox"/>	9147329w980e	19642015415ef2a81b7898433738344a11d3e		1 day ago
<input type="checkbox"/>	8ae72817345	f18dc141cbb57ca8dbae0554e15e7a96e3dbb		2 days ago
<input type="checkbox"/>	002e07b5749b	c475d7022821243cfa7e9fa20ef8b8c0b124e		2 days ago
<input type="checkbox"/>	4ab4545a99e9	dc6d1a6c075b7e45a6b3813556c7b9db5c119a8		2 days ago
<input type="checkbox"/>	42c132ee6ff8	1e6df06ee72f573aefc1f1ad571a91d1727c64		3 days ago
<input type="checkbox"/>	770cc8c3d43e	dc6b148947e9227f11ace10dbbdf6787ae61		4 days ago
<input type="checkbox"/>	24f9c6e60812	f4891c4f15e4c115e1a63e08477b8f620cb4ad		5 days ago

FIGURA 3.19. 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 figura 3.20, 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

Google Cloud | oese-robot | Search (/) for resources, docs, products, and more | Search

Digests for poc-esp32-integration | DELETE | SETUP INSTRUCTIONS

europa-docker.pkg.dev > oese-robot > poc-esp32-integration > poc-esp32-integration

VERSIONS | FILES

Filter: Enter property name or value

<input type="checkbox"/>	Name	Tags	Created	Updated
<input type="checkbox"/>	01ceb62cb4e0	04f1a2724ceb62cb4e0354e41209d716488d5f53	latest	1 minute ago
<input type="checkbox"/>	56e514e0229e	22f37bd5331c78f9c2a6b7c7a5a6d8e733d95d		6 hours ago
<input type="checkbox"/>	843b6935c7b7	548a3e1fa7c7e5e8fbd52c4aef0548c325c6		1 day ago
<input type="checkbox"/>	9147329w980e	19642015415ef2a81b7898433738344a11d3e		1 day ago
<input type="checkbox"/>	8ae72817345	f18dc141cbb57ca8dbae0554e15e7a96e3dbb		2 days ago
<input type="checkbox"/>	002e07b5749b	c475d7022821243cfa7e9fa20ef8b8c0b124e		2 days ago
<input type="checkbox"/>	4ab4545a99e9	dc6d1a6c075b7e45a6b3813556c7b9db5c119a8		2 days ago
<input type="checkbox"/>	42c132ee6ff8	1e6df06ee72f573aefc1f1ad571a91d1727c64		3 days ago
<input type="checkbox"/>	770cc8c3d43e	dc6b148947e9227f11ace10dbbdf6787ae61		4 days ago
<input type="checkbox"/>	24f9c6e60812	f4891c4f15e4c115e1a63e08477b8f620cb4ad		5 days ago

FIGURA 3.19. 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 figura 3.20, 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.