

UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

PROYECTO FINAL INTEGRADOR
INGENIERÍA EN COMPUTACIÓN



**Sistema de captura, almacenamiento y
transmisión de imágenes para el microsatélite
 μ -SAT3**

Autor

Krenz, Matías Ezequiel
Matrícula: 36541189

Autor

Moledo, José Ignacio
Matrícula: 38003509

Director

Ing. Rodriguez González, Santiago A.

2021

Agradecemos a nuestros tutores del Centro de Investigaciones Aplicadas, Ing. Santiago Rodríguez e Ing. Rubén Minutta, por el apoyo constante que recibimos en estos años de trabajo durante la Práctica Profesional Supervisada y el Proyecto Final Integrador.

A nuestros compañeros y amigos de la facultad, que siempre estuvieron presentes para darnos una mano cuando lo necesitábamos y por los momentos compartidos a lo largo de todos estos años de estudio.

A nuestros profesores, por la predisposición, la enseñanza y todos los conocimientos y experiencias compartidas para nuestra formación como profesionales.

A nuestros amigos de la vida, que siempre estuvieron presentes y nos acompañaron en los momentos difíciles y también los felices.

Por último, agradecemos profundamente a nuestros padres, hermanos y familiares. Sin su apoyo no hubiésemos podido llegar hasta esta instancia. No nos alcanzan las palabras para agradecer todo lo que han hecho por nosotros a lo largo de todos estos años.

Resumen

El μ SAT-3 es un proyecto de microsatélite desarrollado actualmente por el Centro de Investigaciones Aplicadas (CIA) de la Fuerza Aérea Argentina (FAA). Una de sus funciones principales es tomar fotografías y transmitirlas a tierra.

Este trabajo plantea el diseño e implementación de un circuito que parte desde tierra hacia el satélite, con el objetivo de enviarle órdenes para capturar imágenes, almacenarlas, previsualizarlas con la posibilidad de recortarlas y finalmente transmitirlas a la estación terrena.

Inicialmente, se detalla el alcance del proyecto, definiendo los objetivos y requerimientos principales. Se realizó un análisis de riesgos para identificar las posibles complicaciones que puedan surgir y tomar las medidas de control correspondientes.

Luego, se realizó un estudio del hardware del sistema y de las distintas herramientas de software disponibles para establecer el plan de desarrollo.

Para cumplir con los requerimientos y objetivos, se diseñó y desarrolló un protocolo de comunicaciones para enviar peticiones y recibir información del microsatélite. Se realizó el software necesario para las distintas computadoras y enlaces que participan en el proceso de comunicación capaces de recibir y transmitir información por radiofrecuencia. Se programó funcionalidades para la captura de imágenes mediante una cámara IP y el recorte de las mismas seleccionando la región de interés por parte del usuario. Por último, se implementó una interfaz de usuario para el operador en tierra que permite la visualización de imágenes y la interacción con el satélite mediante botones específicos para cada función.

Finalmente, se realizaron diversas pruebas funcionales para corroborar el correcto funcionamiento del sistema y se presentan las posibles mejoras a implementar en el futuro.

Abreviaturas

CIA: Centro de Investigaciones Aplicadas.

FAA: Fuerza Aérea Argentina.

OBC: Computadora de a bordo (On Board Computer).

IAS: Sistema de adquisición de imágenes (Image Acquisition System).

ES: Estación terrena (Earth Station).

EBV: Enlace de baja velocidad.

EAV: Enlace de alta velocidad.

ROI: Región de interés (Region of Interest).

GUI: Interfaz gráfica de usuario (Graphical User Interface).

UART: Transmisor-Receptor Asíncrono Universal
(Universal Asynchronous Receiver-Transmitter).

API: Interfaz de programación de aplicaciones (Application Programming Interface).

IDE Entorno de desarrollo integrado (Integrated Development Environment).

SO: Sistema operativo.

REN: Número de exposición al riesgo (Risk Exposure Number).

Índice de figuras

1.	Arquitectura del microsatélite	2
2.	Diagrama de flujo de los comandos enviados al satélite	2
3.	Diagrama de casos de uso	4
4.	Matriz de riesgos	10
5.	Matriz de riesgos (Probabilidad Vs Impacto)	11
6.	Hercules TMS570 MCU	12
7.	Placa emisora RF	13
8.	Placa receptora RF	14
9.	CC Debugger	14
10.	Transceptores RF para el EBV	15
11.	Raspberry Pi 3	15
12.	Especificaciones de la Raspberry Pi 3	16
13.	Consumo acorde al Frame Rate de captura de video a través de una cámara IP (Raspberry Pi 3)	16
14.	Consumo en Watts por modelo de computadora	16
15.	Cámara JAI BM-500GE	17
16.	Dimensiones de la JAI BM-500GE	18
17.	Configuración UART básica	19
18.	Configuración SmartRF Studio	22
19.	Profundidad de campo	23
20.	Velocidad de obturación	23
21.	Sensibilidad ISO	24
22.	Parámetros en fotografía	25
23.	Estructura del Repositorio en GitHub	26
24.	Estructura de cada aplicación del Repositorio en GitHub	27
25.	Proceso de descubrimiento de errores	28
26.	Descubrimiento de errores con TDD	28
27.	Ciclo TDD	30
28.	Relación entre comandos y acciones	34
29.	Buffers de comandos y acciones	35
30.	CommManager y ActionManager	35
31.	Mensaje enviado desde la ES	36
32.	Nuevo proyecto en HALCoGen	40
33.	"Target Configuration" en CCS	41
34.	Estructura del proyecto en Qt-Creator	42
35.	Especificaciones Baumer GAPI SDK	44
36.	Descripción general de capas en Baumer GAPI	45
37.	Estructura y punto de entrada a Baumer-GAPI	46
38.	11 pasos para obtener una fotografía mediante Baumer.	47
39.	Secuencia de captura y recorte de una imagen.	49
40.	Arquitectura del EAV	50
41.	Transmisión de imagen en alta calidad	52
42.	Model-view-viewmodel	53
43.	Esquema de la GUI	54
44.	ModelView y su Q_PROPERTY	55
45.	Uso de connect() para actualizar una Q_PROPERTY	56
46.	Lista de ViewModels y su visualización en QML	57
47.	Textos de información en la GUI	57
48.	GUI V1.0	58
49.	Envío y recepción de comando para consultar el estado del led de la IAS	60

50.	Envío de comando para cambiar el estado del led de la IAS	60
51.	Verificación de cambio de estado del led realizando nueva consulta	60
52.	Botones de la GUI bloqueados sin puertos configurados	61
53.	Se habilita el botón Connect al configurar el puerto del EBV	62
54.	Se habilitan el resto de botones que no utilizan el EAV	63
55.	GUI con todos los botones habilitados	64
56.	Fotografía capturada con la webcam en modo vertical	65
57.	Imagen capturada, listada y descargada en baja calidad para previsualización	66
58.	Zoom de imagen listada en el catálogo	67
59.	Selección de ROI de una imagen del catálogo	68
60.	Recorte de la imagen original	69
61.	Recorte de imagen descargado por el EBV en la ES	69
62.	Resultado de la primera ejecución de cppcheck	71
63.	Resultado de la primera ejecución de memcheck	71
64.	Resultado final de cobertura de código alcanzado	72
65.	RJ-45 pin out	80
66.	D-sub pin out para GPIO	80
67.	Interpretación de LEDs en la cámara JAI	80
68.	1) Raspberry Pi 3 Model B. 2) Cable GigE. 3) JAI BM-500GE Serie B. 4) Fuente de alimentación de 12V.	81
69.	Conexión RPi	86
70.	Conexión emisor	87
71.	Conexión receptor	87
72.	Conexión conversor UART	87

Índice de tablas

1.	Requerimientos de la GUI	5
2.	Requerimientos de la OBC.	6
3.	Requerimientos de la IAS.	6
4.	Requerimientos no funcionales.	6
5.	Riesgo Pérdida de conexión en los enlaces de comunicación.	8
6.	Riesgo Daño en las placas de desarrollo utilizadas durante el desarrollo.	8
7.	Riesgo Bugs y errores de software.	8
8.	Riesgo Pérdida de información relacionada al proyecto.	8
9.	Riesgo Acceso indebido al sistema.	9
10.	Riesgo Fallas en los sistemas operativo.	9
11.	Riesgo Finalización anormal o bloqueo de la aplicación.	9
12.	Riesgo Memory leak.	9
13.	Estructura de los paquetes.	34
14.	Comando Keep Alive Req	37
15.	Comando Keep Alive Res	37
16.	Comando Crop Req	37
17.	Comando Image HQ Req	37
18.	Comando Image HQ Res	37
19.	Comando Image LQ Req	38
20.	Comando Image LQ Res	38
21.	Comando Catalog Req	38
22.	Comando Catalog Res	38
23.	Comando Delete Req	38
24.	Comando Capture Req	38
25.	Comando Led OBC Get Req	39
26.	Comando Led OBC Get Res	39
27.	Comando Led OBC Set Req	39
28.	Comando Led IAS Get Req	39
29.	Comando Led IAS Get Res	39
30.	Comando Led IAS Set Req	39
31.	Configuración del hardware	40
32.	Parámetros modificables en el archivo de interfaz	82

Índice general

Agradecimientos	i
Resumen	ii
Abreviaturas	iii
Índice de figuras	iv
Índice de tablas	vi
1. Introducción	1
1.1. Proyecto "FAS-1400 Microsatélite μ -SAT 3"	1
1.1.1. Arquitectura del microsatélite	1
1.1.2. Funcionamiento general	2
1.2. Motivación	3
1.2.1. Antecedente: Enlace de alta velocidad (Banda S)	3
1.3. Objetivos	3
1.4. Estructura del informe	4
1.5. Ingeniería de requerimientos	5
1.5.1. Requerimientos funcionales del sistema	5
1.5.1.1. Requerimientos de la GUI	5
1.5.1.2. Requerimientos de la OBC	6
1.5.1.3. Requerimientos de la IAS	6
1.5.2. Requerimientos no funcionales del sistema	6
1.6. Gestión de riesgos	7
1.6.1. Sistema de clasificación de riesgos	7
1.6.2. Identificación de riesgos	7
1.6.3. Análisis de riesgos	9
1.6.4. Plan de respuesta al riesgo	11
1.6.4.1. Consideraciones en la metodología de desarrollo	11
1.6.4.2. Nuevos requerimientos	11
2. Marco Teórico	12
2.1. Componentes de hardware	12
2.1.1. Hercules TMS570 MCU	12
2.1.1.1. Sistema operativo (FreeRTOS)	13
2.1.2. Transceptores para el EAV (CC2544)	13
2.1.2.1. CC Debugger	14
2.1.3. Transceptores para el EBV (CC1010)	14
2.1.4. Raspberry Pi	15
2.1.4.1. Sistema operativo (Raspbian)	16
2.1.5. Cámara	17
2.2. Protocolos y comunicaciones	18
2.2.1. Gigabit Ethernet	18
2.2.1.1. GigE Vision y GenICam	18
2.2.2. Comunicación Serie	19
2.2.3. SPI	20
2.2.4. Radiofrecuencia (RF)	20
2.3. Herramientas de software	21
2.3.1. Code Composer Studio (CCS)	21

2.3.2.	HalCoGen	21
2.3.3.	IAR Embedded Workbench	21
2.3.4.	SmartRF Studio	21
2.4.	Parámetros en fotografía (Exposición)	22
2.4.1.	Apertura del diafragma	22
2.4.2.	Velocidad de obturación	23
2.4.3.	Sensibilidad ISO	24
3.	Desarrollo	26
3.1.	Parte 1: Proceso de desarrollo	26
3.1.1.	Control de versiones	26
3.1.2.	Modelado y UML	27
3.1.3.	Test Driven Development (TDD)	27
3.1.3.1.	TDD vs programación tradicional	28
3.1.3.2.	Ciclo TDD	29
3.1.3.3.	Tests unitarios	30
3.1.4.	Análisis estático	30
3.1.5.	Análisis dinámico	31
3.1.5.1.	Cobertura de código (Test Coverage)	31
3.1.6.	Definición de Hecho (Done)	31
3.2.	Parte 2: Protocolo y EBV	33
3.2.1.	Requerimientos a implementar	33
3.2.2.	Investigación y desarrollo	33
3.2.2.1.	Protocolo de comunicaciones	33
3.2.2.2.	Comandos y acciones	34
3.2.2.3.	Comandos para la OBC y forwarding	35
3.2.2.4.	Detalles de los comandos	36
3.2.2.5.	Desarrollo en OBC	40
3.2.2.6.	Desarrollo en ES e IAS	41
3.3.	Parte 3: Captura y recorte de imágenes	43
3.3.1.	Requerimientos a implementar	43
3.3.2.	Investigación y desarrollo	43
3.3.2.1.	Software para el control de la cámara	43
3.3.2.2.	OpenCV	45
3.3.2.3.	Exposición Manual vs. Exposición Automática	45
3.3.2.4.	Captura de fotografías	46
3.3.2.5.	Recorte de imágenes en la ES	48
3.3.2.6.	Recorte de imágenes en la IAS	49
3.4.	Parte 4: Descarga de imágenes HQ y EAV	50
3.4.1.	Requerimientos a implementar	50
3.4.2.	Investigación y desarrollo	50
3.4.2.1.	Software en la IAS	50
3.4.2.2.	Software del transmisor y receptor	51
3.4.2.3.	Software en la ES	51
3.5.	Parte 5: GUI e integración	53
3.5.1.	Requerimientos a implementar	53
3.5.2.	Investigación y desarrollo	53
3.5.2.1.	Diseño de la GUI	54
3.5.2.2.	Desarrollo de la GUI	55

4. Resultados	59
4.1. Consideraciones	59
4.2. Conectarse al microsatélite	59
4.2.1. Simulación del protocolo	59
4.2.2. Pruebas en hardware	60
4.2.2.1. Secuencia de conexión	61
4.3. Capturar imágenes	64
4.4. Listar y previsualizar imágenes (descarga por el EBV)	65
4.5. Recortar imágenes	67
4.5.1. Recorte en la ES	67
4.5.2. Recorte en la IAS	68
4.6. Borrar imágenes	70
4.7. Descargar imágenes en alta calidad	70
4.8. Análisis y cobertura de código	70
5. Conclusiones	73
5.1. Trabajos futuros	75
Bibliografía	76
A. Guía de instalación	78
A.1. Link del repositorio en Github	78
A.2. Instalación Baumer GAPI	78
A.2.1. Instalación de software	78
A.2.1.1. Pasos preliminares	78
A.2.1.2. Instalación de Baumer GAPI SDK	78
A.2.1.3. Consultando los archivos instalados	79
A.2.1.4. Consultando información sobre el paquete instalado	79
A.2.1.5. Eliminar el paquete instalado	79
A.2.2. Instalación de hardware	80
A.2.2.1. Instrucciones	81
A.2.3. Configuración de red	81
A.2.4. Camera Explorer	82
A.3. Instalación OpenCV	82
A.3.1. Mantenga Ubuntu o Raspbian actualizado	82
A.3.2. Instalación de dependencias	82
A.3.2.1. Build tools	82
A.3.2.2. GUI	83
A.3.2.3. Media I/O	83
A.3.2.4. Video I/O	83
A.3.2.5. Bibliotecas de paralelismo y álgebra lineal	83
A.3.2.6. Python	83
A.3.2.7. Java	83
A.3.2.8. Documentación	83
A.3.3. Instale la biblioteca (Puede cambiar '3.2.0' por la última versión estable)	83
A.3.4. Instale la biblioteca Open_Contrib (Puede cambiar '3.2.0' por la última versión estable)	84
A.3.5. Buildear OpenCV con contrib	84
A.3.6. Solución a errores	84
A.3.6.1. Dependencias faltantes	84
A.3.6.2. Error al compilar opencv con ffmpeg	84
A.3.6.3. Error de constantes	85

A.4. Instalación QT	85
A.4.1. solución de errores	85
A.4.1.1. Solución a Aborted (core dumped)	85
A.5. Creación de puertos virtuales	86
A.6. Conexión del EAV	86
A.6.1. Diagramas de conexión	86
A.7. Software y conexión de la OBC	88
A.7.1. Software	88
A.7.2. Conexión de la OBC	88

Capítulo 1

1. Introducción

"El espacio es la última frontera para estudiar la salud de nuestro planeta y su evolución. Los satélites son la herramienta más precisa para responder a preguntas fundamentales sobre el clima, los océanos, la atmósfera y el interior de la Tierra." [1].

La historia de los satélites argentinos comienza a fines del siglo XX. El μ SAT-1 (Victor-1) fue el primer satélite íntegramente construido en Argentina y puesto en órbita. Desarrollado en el Centro de Investigaciones Aplicadas (CIA) y con un peso aproximado de 30 Kg, fue lanzado en agosto de 1996 estando operativo durante 3 años. A este proyecto le siguió el μ SAT-2, que no pudo finalizarse por falta de financiamiento.

1.1. Proyecto "FAS-1400 Microsatélite μ -SAT 3"

El μ SAT-3 es un proyecto de microsatélite de bajo costo desarrollado actualmente por el CIA de la Fuerza Aérea Argentina (FAA) y financiado por el Ministerio de Defensa.

De aplicación científica y pensado para transitar en la órbita terrestre baja (LEO, por Low Earth Orbit en inglés), el μ SAT-3 es la evolución del ya conocido μ SAT-1 (Víctor-1). Se estima una vida útil de al menos 3 años y un peso aproximado de 30 Kg. Orbitará a unos 700 kilómetros sobre la superficie de la Tierra con una velocidad de 25 mil kilómetros por hora aproximadamente, pasando sobre una latitud determinada a la misma hora.

Su función principal es la observación y vigilancia de la Tierra, capturando fotografías de alta calidad que permiten abarcar temas como el control del clima, prevención y monitorización de incendios, estudio de los océanos, observaciones de la atmósfera, entre otros.

Cuenta con dos cámaras fotográficas, una monocromática de 5 megapíxeles y otra de alta definición de 16 megapíxeles que permiten observar la Tierra con una resolución de 10 metros por pixel. Posee un novedoso sistema de propulsión a partir de un motor de plasma, producido enteramente en el país.

En Tierra, la Estación Terrena, o Earth Station (ES), es el sitio encargado de obtener la información tomada por el microsatélite, analizar e interpretar sus imágenes y enviarle comandos específicos. Estará dentro del predio del CIA y tendrá al menos dos contactos por día con él.

1.1.1. Arquitectura del microsatélite

Posee dos computadoras principales que se comunican entre sí mediante un enlace bidireccional dedicado (*Figura 1*):

- La Computadora de a bordo, u On Board Computer (OBC), se encarga del procesamiento general. Se comunica con la ES a través de un enlace de radio frecuencia de baja velocidad (EBV) utilizado

para control y telemetría.

- El Sistema de Adquisición de Imágenes, o Image Acquisition System (IAS), tiene como propósito principal capturar imágenes mediante cámaras controladas por la misma. Estas imágenes se graban en un dispositivo de almacenamiento de datos para su posterior envío a través de un enlace de radio frecuencia de alta velocidad (EAV). Este enlace es unidireccional, es decir, solo podrán enviarse datos desde el satélite hacia la ES.

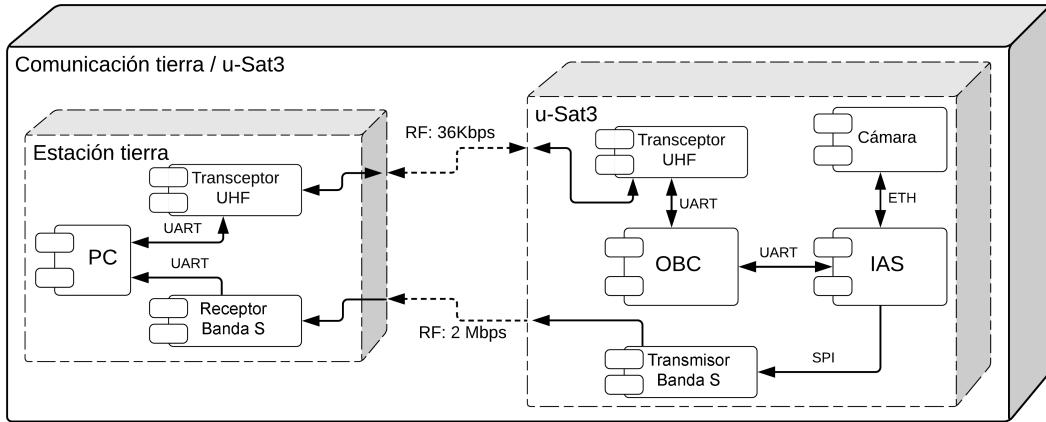


Figura 1: Arquitectura del microsatélite

1.1.2. Funcionamiento general

Todos los comandos enviados por la ES llegan a la interfaz de la OBC conectada al enlace de baja velocidad. La OBC procesa los comandos que le corresponden y reenvía por el enlace dedicado los comandos con destino a la IAS (forwarding). Si alguno no se reconoce, se descarta (*Figura 2*).

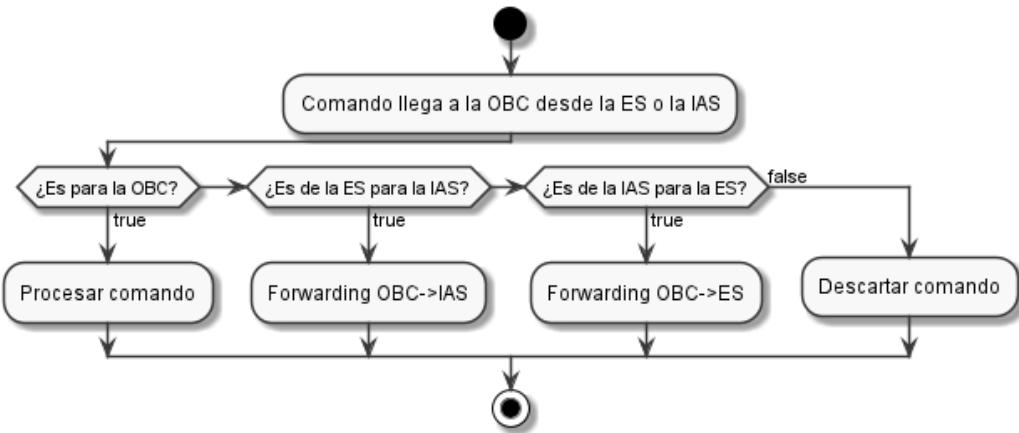


Figura 2: Diagrama de flujo de los comandos enviados al satélite

1.2. Motivación

Argentina se encuentra entre uno de los diez países con mayor capacidad de construcción en tecnología aeroespacial [2]. Gracias al diseño, producción y lanzamiento de satélites argentinos que se encuentran en el espacio, se logra mantener las órbitas asignadas a la Argentina por la Unión Internacional de Telecomunicaciones (UIT).

Los satélites van tomando cada vez más relevancia. La tecnología aeroespacial es un pilar fundamental para el desarrollo de la ciencia, la investigación y el desarrollo. Gracias a ellos, hemos cambiado la forma de comunicarnos y de observar la tierra.

Actualmente, ningún satélite argentino tiene la capacidad de sacar fotografías con la calidad y frecuencia que aportará el μ SAT-3 [3]. El proyecto cuenta con ingenieros de distintas especialidades encargados de desarrollar módulos específicos de acuerdo a su área profesional.

De estos módulos, este trabajo se concentra principalmente en tres secciones que combinan varios de los principales aspectos de la carrera de Ingeniería en Computación, incluidos las comunicaciones, el hardware y el diseño e implementación de software. Por un lado, el desarrollo de las funcionalidades de la IAS y la problemática de la captura y transmisión de imágenes. Por otro, la implementación de un protocolo de comunicación para la conexión entre la OBC y la ES. Finalmente, el desarrollo de una interfaz gráfica de usuario, o Graphical User Interface (GUI), para el control de la IAS desde la ES.

Las soluciones a los problemas complejos que se presentan en esta clase de proyectos pueden tomar varios caminos posibles. Los autores esperan que este trabajo final sirva de base para desarrollar luego el sistema final de transmisión de imágenes, utilizando la solución más adecuada.

1.2.1. Antecedente: Enlace de alta velocidad (Banda S)

En la Práctica Profesional Supervisada (PPS) realizada con anterioridad por los autores de este trabajo [4], se implementa un prototipo básico del EAV en Banda S que incluye el desarrollo de:

- El firmware necesario para transmitir y recibir imágenes de una placa RF emisora ubicada en el microsatélite y otra receptora RF en la ES.
- Un programa perteneciente a la IAS que fragmenta una imagen en paquetes y los envía al transmisor Banda S ubicado en el microsatélite.
- Un programa perteneciente a la ES que recibe los datos del receptor RF en Banda S y reconstruye la imagen.

En este prototipo básico, solo se controla que todos los paquetes hayan sido recibidos pero no su contenido. Si se pierde algún paquete, se solicita el reenvío de la imagen nuevamente. Esta implementación servirá de base para el desarrollo del EAV en el presente trabajo.

1.3. Objetivos

El objetivo principal de este trabajo es tomar el control del IAS desde la ES para capturar, almacenar y transmitir imágenes satelitales (*Figura 3*).

Para ello, se identifican los siguientes objetivos específicos:

- Diseñar e implementar un protocolo de comunicación UART para transmitir información entre la ES y las computadoras del microsatélite.
- Desarrollar una GUI en la ES que permita conectarse y ejecutar acciones en la IAS.
- Capturar imágenes mediante una cámara conectada vía Gb Ethernet a la IAS.
- Guardar las imágenes capturadas en un dispositivo de almacenamiento de datos.
- Permitir borrar las imágenes capturadas.
- Almacenar una vista previa en baja resolución (thumbnail) al capturar una imagen.
- Permitir recortar una imagen para guardar una o más regiones de interés, o Regions of Interest (ROI).
- Enviar las imágenes originales o recortadas a través del EAV.
- Enviar las vistas previas de las imágenes por el EBV.

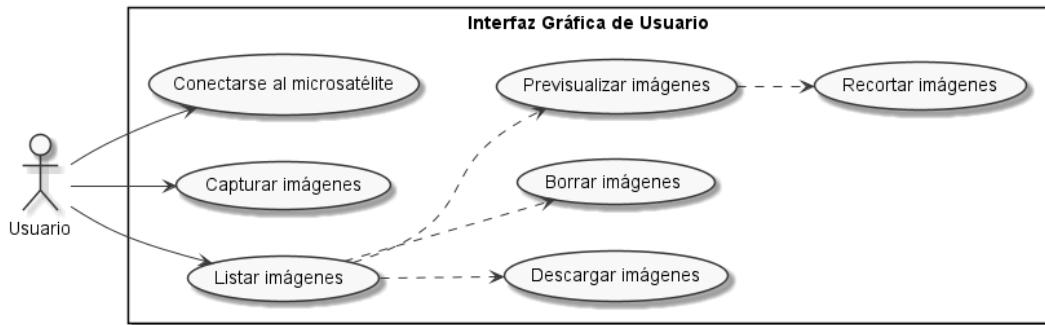


Figura 3: Diagrama de casos de uso

1.4. Estructura del informe

La estructura del presente informe consta de 5 capítulos.

- Capítulo 1 - Introducción: Se presentan las características, arquitectura y funcionamiento del microsatélite μ SAT-3. Se detalla el motivo de este proyecto, sus objetivos y sus requerimientos. Finalmente se analizan riesgos y se proponen mecanismos de control.
- Capítulo 2 - Marco teórico: Se introduce el hardware del proyecto, los protocolos usados para su comunicación y el software asociado a ellos. Se explican parámetros de fotografía recurrentes en este trabajo.
- Capítulo 3 - Desarrollo: Se explica el proceso de investigación, diseño e implementación realizado para cumplir con los requerimientos. Además, se realizan pruebas exponiendo sus correspondientes resultados.
 - Parte 1 - Proceso de desarrollo: Se detalla de qué manera se va diseñar y crear el software.
 - Parte 2 - Protocolo y EBV: Comprende el diseño y desarrollo del protocolo de comunicaciones para enviar peticiones y recibir información del microsatélite.

- Parte 3 - Captura y recorte de imágenes: Trata sobre el control de la cámara para la captura y almacenamiento de imágenes y sobre cómo se generan nuevas imágenes a partir del recorte de las existentes.
 - Parte 4 - Descarga de imágenes HQ y EAV: Se centra en la revisión del firmware de los transceptores de alta velocidad para la transmisión de imágenes en alta calidad.
 - Parte 5 - GUI e integración: Se realiza la interfaz gráfica que permite enviar y recibir comandos de forma cómoda e intuitiva, además de visualizar información de interés como el estado de conexión y métricas de mensajes.
- Capítulo 4 - Resultados: Se realizan pruebas del sistema en su totalidad verificando cada uno de los casos de uso planteados.
 - Capítulo 5 - Conclusiones: Se comparan y validan los resultados con respecto a los objetivos y requerimientos iniciales. Se proponen recomendaciones y oportunidades de mejora para trabajos futuros.

1.5. Ingeniería de requerimientos

Los requerimientos de un sistema son descripciones de los servicios que un sistema debería proporcionar y las restricciones en su funcionamiento. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se llama ingeniería de requerimientos [5].

1.5.1. Requerimientos funcionales del sistema

Los requerimientos funcionales describen las funciones del sistema, sus entradas y salidas, y sus excepciones al detalle.

1.5.1.1 Requerimientos de la GUI

ID	Descripción
RF1	Establecer la conexión con el microsatélite.
RF2	Enviar comandos para interactuar con el microsatélite.
RF3	Descargar el catálogo de las imágenes en el microsatélite.
RF4	Descargar y previsualizar imágenes del catálogo en baja resolución.
RF5	Seleccionar y recortar una o más ROI de una imagen del microsatélite.
RF6	Visualizar información sobre el estado, errores y número de mensajes (enviados, recibidos y perdidos).
RF7	Borrar imágenes en el microsatélite.
RF8	Seleccionar y descargar imágenes en alta resolución.

Tabla 1: Requerimientos de la GUI

1.5.1.2 Requerimientos de la OBC

ID	Descripción
RF9	Procesar los comandos que le pertenecen y reenviar los que le corresponden a la IAS o a la ES.

Tabla 2: Requerimientos de la OBC.

1.5.1.3 Requerimientos de la IAS

ID	Descripción
RF10	Recibir comandos de la OBC, procesarlos y responder a ellos si corresponde.
RF11	Capturar imágenes mediante la cámara monocromática.
RF12	Almacenar imágenes capturadas.
RF13	Mantener un catálogo actualizado de las imágenes.
RF14	Enviar el catálogo por el EBV a través de la OBC.
RF15	Transmitir imágenes en alta resolución por el EAV.
RF16	Transmitir imágenes en baja resolución por el EBV.
RF17	Borrar imágenes solicitadas por el usuario.
RF18	Recortar las imágenes indicadas por el usuario.

Tabla 3: Requerimientos de la IAS.

1.5.2. Requerimientos no funcionales del sistema

Los requerimientos no funcionales son restricciones en los servicios o funciones que ofrece el sistema. En general afectan a todo el sistema, no solo a características o servicios individuales.

ID	Descripción
RNF1	La IAS y la ES deben utilizar un SO basado en GNU/Linux.
RNF2	La OBC debe utilizar el sistema operativo FreeRTOS.
RNF3	Las órdenes de la ES hacia la OBC se deberán enviar por un enlace UART.
RNF4	Las órdenes al satélite, respuestas e imágenes en baja resolución deberán transmitirse por el EBV, usando transceptores RF desarrollados previamente [6].
RNF5	La OBC se comunicará con la IAS a través de un enlace UART.
RNF6	La IAS deberá utilizar el protocolo GigE para controlar la cámara.
RNF7	Las imágenes en alta resolución deberán transmitirse por el EAV utilizando transceptores RF desarrollados previamente [7].
RNF8	El EAV debe ser unidireccional.
RNF9	El formato de las imágenes en baja resolución debe ser .jpg.
RNF10	El formato de las imágenes en alta resolución debe ser .bmp.

Tabla 4: Requerimientos no funcionales.

1.6. Gestión de riesgos

La gestión de riesgos es un enfoque sistemático para reducir el daño debido a los riesgos, haciendo que el proyecto sea menos vulnerable y que el producto sea más robusto [8].

Según la norma ISO 31000, el riesgo es el "efecto de la incertidumbre sobre los objetivos".

El sistema de gestión de riesgos surge de la necesidad por conocer los riesgos y puede ampliarse para incluir planes de respuesta.

A continuación se presentan los elementos clave de este sistema que vamos a utilizar.

1.6.1. Sistema de clasificación de riesgos

Es común asignar un Número de Exposición al Riesgo, o Risk Exposure Number (REN), para cada riesgo identificado y usarlo como una escala. El REN tiene como objetivo aportar la mayor objetividad posible a la percepción del riesgo.

Para usar el REN como escala de riesgo, se debe asignar para cada riesgo una probabilidad de ocurrencia y un impacto.

La definición de estas características es subjetiva y deben tener uno de los siguientes valores discretos: muy alto (MA), alto (A), medio (M), bajo (B) y muy bajo (MB).

Al convertir esta clasificación en una calificación cuantitativa, se obtiene más claridad. A los valores MA, A, M, B y MB se les asigna un número usando las siguientes reglas:

- MA = 9
- A = 7
- M = 5
- L = 3
- MB = 1

El REN surge de la multiplicación de los valores numéricos de la probabilidad de ocurrencia y el impacto.

1.6.2. Identificación de riesgos

La identificación de riesgos es el proceso de buscar y detectar riesgos en el entorno, reconocer sus atributos y estimar sus consecuencias.

Analizando la lista de requerimientos, se encuentran los siguientes riesgos:

R1 - Pérdida de conexión en los enlaces de comunicación
Descripción: Pérdida de conexión debido a diversos factores.
Impacto: 8.
Probabilidad: 5.
REN: 40.
Consecuencia: Dejar inconclusas tareas que se estaban realizando durante la comunicación. Descarga incompleta de un paquete de imágenes. Incapacidad de capturar imágenes.

Tabla 5: Riesgo Pérdida de conexión en los enlaces de comunicación.

R2 - Daño en las placas de desarrollo utilizadas durante el desarrollo
Descripción: Daño en los componentes electrónicos que imposibilita el correcto funcionamiento del proyecto.
Impacto: 8.
Probabilidad: 2.
REN: 16.
Consecuencia: Demoras en el desarrollo y aumento del costo del proyecto para reemplazar componentes dañados.

Tabla 6: Riesgo Daño en las placas de desarrollo utilizadas durante el desarrollo.

R3 - Bugs y errores de software
Descripción: Se produce un bug o error de software en el microsatélite.
Impacto: 8.
Probabilidad: 5.
REN: 40.
Consecuencia: Inutilización parcial o total del sistema.

Tabla 7: Riesgo Bugs y errores de software.

R4 - Pérdida de información relacionada al proyecto
Descripción: Pérdida de documentación y/o código fuente del proyecto debido a robo, incendio, fallas de sistemas de almacenamiento, borrado de información intencional, entre otras causas.
Impacto: 10.
Probabilidad: 2.
REN: 20.
Consecuencia: Interrupción del desarrollo del trabajo que desemboca en una entrega tardía del mismo.

Tabla 8: Riesgo Pérdida de información relacionada al proyecto.

R5 - Acceso indebido al sistema
Descripción: Acceso no autorizado por un agente externo al sistema a través de una falla de seguridad.
Impacto: 10.
Probabilidad: 3.
REN: 30.
Consecuencia: Anulación del sistema o robo de información.

Tabla 9: Riesgo Acceso indebido al sistema.

R6 - Fallas en los sistemas operativos
Descripción: El sistema no responde o posee tiempo de respuesta muy alto.
Impacto: 7.
Probabilidad: 2.
REN: 14.
Consecuencia: Dificultad para realizar las tareas. Demoras en la descarga de imágenes. Incapacidad para conectarse al microsatélite.

Tabla 10: Riesgo Fallas en los sistemas operativo.

R7 - Finalización anormal o bloqueo de la aplicación
Descripción: La aplicación deja de funcionar correctamente y se bloquea o se cierra a partir de un error no controlado u otras causas.
Impacto: 7.
Probabilidad: 4.
REN: 28.
Consecuencia: Dificultad para realizar las tareas. Demoras en la descarga de imágenes. Incapacidad para realizar la conexión del satélite.

Tabla 11: Riesgo Finalización anormal o bloqueo de la aplicación.

R8 - Memory leak
Descripción: Administración incorrecta de la memoria de la aplicación.
Impacto: 5.
Probabilidad: 5.
REN: 25.
Consecuencia: Pérdida de información no guardada. Bloqueo de las aplicaciones.

Tabla 12: Riesgo Memory leak.

1.6.3. Análisis de riesgos

El propósito del análisis de riesgos es seleccionar qué riesgos deben ser mitigados.

Un método para realizarlo es hacer un análisis de cuadrantes donde los riesgos se representan en cuatro cuadrantes en un gráfico bidimensional que muestra el impacto en el eje X y la probabilidad en el eje Y (*Figura 4*).

- Cuadrante I: Riesgos de alto impacto y alta probabilidad.
- Cuadrante II: Riesgos de alto impacto y baja probabilidad.
- Cuadrante III: Riesgos de bajo impacto y alta probabilidad.
- Cuadrante IV: Riesgos de bajo impacto y baja probabilidad.

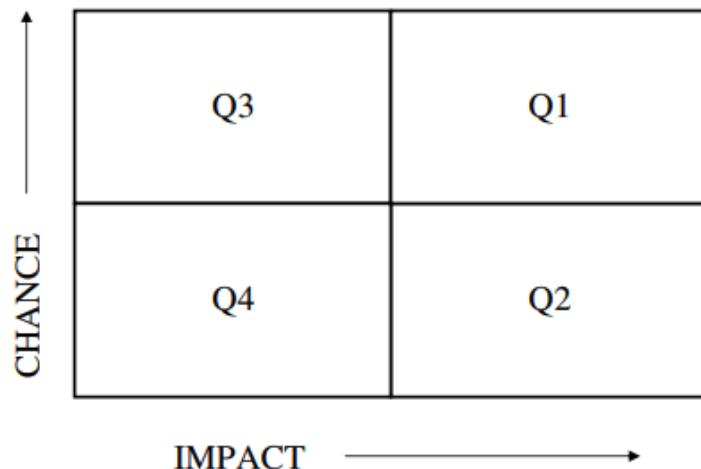


Figura 4: Matriz de riesgos

En lugar de cuatro cuadrantes, se prefiere hacer un análisis de matriz, que es un refinamiento sobre el análisis de cuadrantes. El espacio de riesgo se divide en una matriz de 10 por 10. El eje X representa el impacto en una escala de 0 a 10. El eje Y representa la probabilidad de riesgo en una escala de 0 a 10. Cada ubicación de matriz tiene un valor de riesgo específico: la ubicación (1, 1) tiene el valor más bajo y (10, 10) representa el problema más crítico.

Esta matriz (*Figura 5*) ayuda a obtener una vista panorámica de los riesgos y responder a los más críticos primero.

P10										
P9										
P8										
P7										
P6										
P5					R8			R1-R3		
P4							R7			
P3									R5	
P2							R6	R2		R4
P1										
	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10

Figura 5: Matriz de riesgos (Probabilidad Vs Impacto)

1.6.4. Plan de respuesta al riesgo

Una vez terminada la matriz de riesgos, se establece que aquellos riesgos con REN mayor o igual a 20 deben ser mitigados.

Algunas medidas de control (MC) para disminuir el REN se tienen en cuenta a la hora de planificar la metodología de desarrollo. Otras, se convierten en nuevos requerimientos.

1.6.4.1 Consideraciones en la metodología de desarrollo

MC(R3): Prevenir la mayor cantidad de errores usando una metodología de desarrollo basada en pruebas unitarias. Utilizar además, análisis estático de software.

MC(R4): Utilizar un sistema de control de versiones y un repositorio remoto.

MC(R8): Utilizar herramientas de análisis dinámico para detectar “leaks” de memoria.

1.6.4.2 Nuevos requerimientos

RF19 - Surgido como MC(R1): Detectar la pérdida de la conexión entre el satélite y la Estación Terrena utilizando un “keepalive”.

RF20 - Surgido como MC(R1): Establecer un mecanismo de reconexión en caso de que se pierda la conexión entre el satélite y la Estación Terrena.

RF21 - Surgido como MC(R5): Inicio de sesión con contraseña.

RF22 - Surgido como MC(R7): Reiniciar y re-sincronizar la aplicación en caso de fallo.

Capítulo 2

2. Marco Teórico

2.1. Componentes de hardware

Los componentes descriptos a continuación, junto con las interfaces y protocolos a utilizar para su comunicación, son mandatorios para la realización de este trabajo. Estos hacen las veces de los principales sistemas de procesamiento y comunicación del satélite.

2.1.1. Hercules TMS570 MCU

Es una placa (*Figura 6*) perteneciente al kit de desarrollo TMS570LS31 Hercules de Texas Instruments [9]. La misma emula la computadora principal del satélite (OBC) que será luego una placa dedicada. Utiliza un controlador RISC de 32 bits Hercules TMS570 y cuenta con dos interfaces UART que son necesarias para la utilización de la OBC como proxy (intermediario) entre la ET y el IAS.

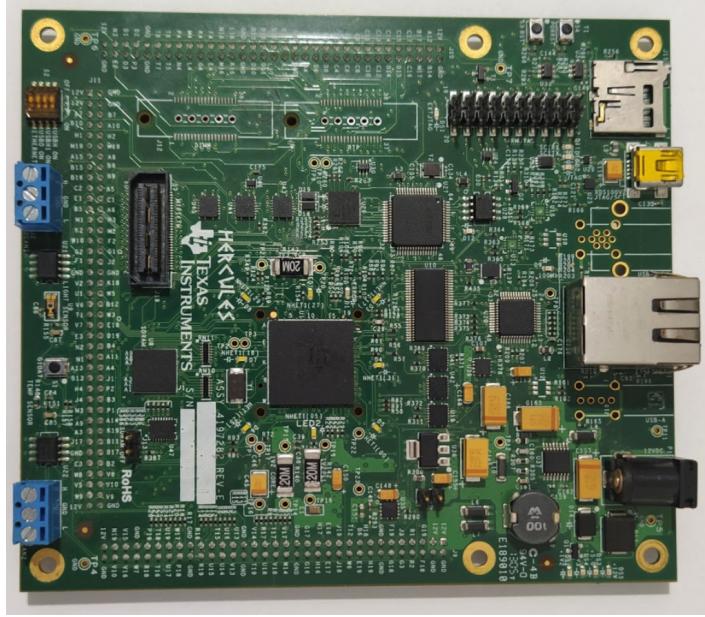


Figura 6: Hercules TMS570 MCU

Algunas de las características de este dispositivo son [10]:

- CPU ARM Cortex-R4F 32-Bit RISC: 1.66 DMIPS/MHz con pipeline de 8 etapas.
- System Clock de hasta 180 MHz.

- Múltiples interfaces de comunicación: incluye una SCI (Standard Serial Communication Interface) y un controlador de interfaz LIN (Local Interconnect Network) que puede ser configurada como una segunda SCI.

2.1.1.1 Sistema operativo (FreeRTOS)

La OBC debe funcionar con FreeRTOS, el cual es un sistema operativo de tiempo real para dispositivos embebidos que se distribuye libremente bajo la licencia MIT. FreeRTOS está construido con énfasis en la fiabilidad y la facilidad de uso [11]. Su kernel está contenido en solo 3 archivos escritos en lenguaje C.

Al utilizarlo, entre otros beneficios, se puede ordenar las tareas por prioridad y garantizar un tiempo de respuesta por debajo de cierto umbral, ya que se trata de un sistema operativo determinista.

2.1.2. Transceptores para el EAV (CC2544)

Un transceptor de radiofrecuencia es un dispositivo capaz de recibir y transmitir datos de manera inalámbrica.

El microcontrolador CC2544 es una solución System-on-Chip (SoC) para aplicaciones con velocidades de datos de hasta 2 Mbps. Tiene 1 KB de RAM que contiene las dos FIFO de transmisión y recepción, cada una de 128 bytes de tamaño. La misma combina un transceptor RF en banda S de 2,4 Ghz con un CPU de un solo ciclo, memoria flash programable de 32 kB y hasta 2 kB de RAM [12].

Las placas transceptoras (*Figura 7 y 8*) que se utilizan para lograr la comunicación entre la ES y la IAS (EAV) fueron diseñadas con anterioridad en el CIA [7]. Cada placa posee el chip CC2544, pines para diferentes conexiones y circuito de adaptación de impedancia para la conexión antenas.

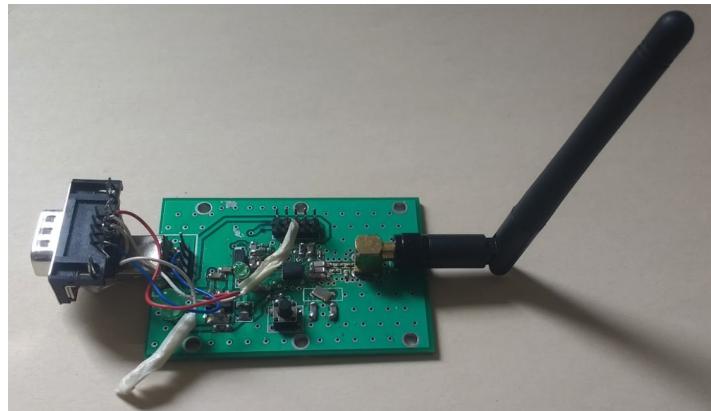


Figura 7: Placa emisora RF

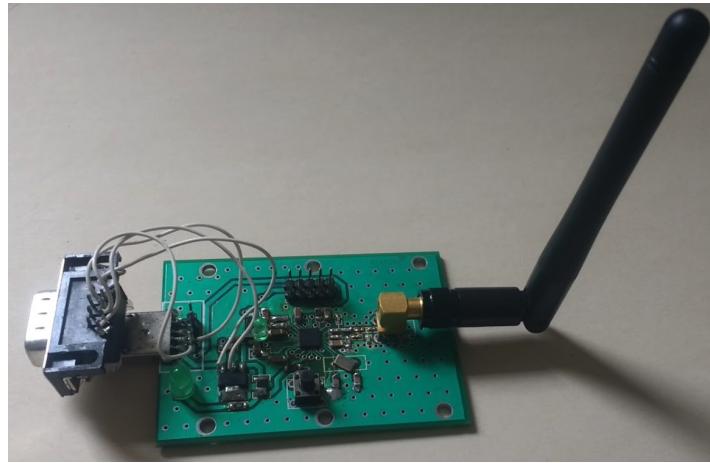


Figura 8: Placa receptor RF

2.1.2.1 CC Debugger

El CC Debugger (*Figura 9*) es un pequeño programador y debugger para chips RF de baja potencia de Texas Instruments. Se utiliza para cargar el programa a la placa de los transceptores y realizar pruebas de depuración. Para instalar los controladores (drivers) USB requeridos para esta herramienta, se recomienda la instalación de SmartRF™ Studio.



Figura 9: CC Debugger

2.1.3. Transceptores para el EBV (CC1010)

El integrado SoC CC1010 de la firma Texas Instruments, se trata de un microcontrolador que posee embebido en la misma pastilla un transceptor de radiofrecuencia. Se puede programar para funcionar en el rango de 300 a 1000 MHz y está diseñado para aplicaciones inalámbricas de muy baja potencia [13].

Los transceptores (*Figura 10*) de radiofrecuencia en la banda de UHF fueron diseñados con anteriodad en el CIA [6] y dan soporte al enlace de datos de telemetría (EBV) del microsatélite.

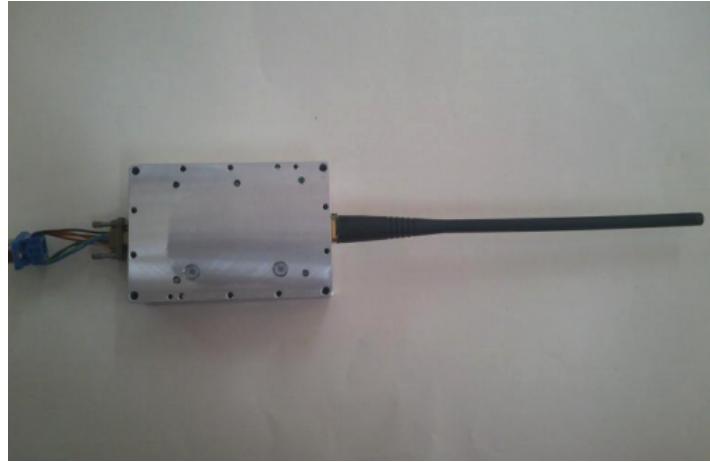


Figura 10: Transceptores RF para el EBV

2.1.4. Raspberry Pi

Es una computadora (*Figura 11*) de placa reducida con una CPU de cuatro núcleos basada en la arquitectura ARM Cortex A53.

Esta placa de desarrollo emula la IAS que, al igual que la OBC, será luego una placa dedicada.



Figura 11: Raspberry Pi 3

La Raspberry Pi admite una velocidad máxima de datos de 100 Mbit/s a través del puerto Ethernet (*Figura 12*). Por lo tanto, puede ser necesario reducir la tasa de datos según la cámara a conectar [14].

Raspberry Pi® 3 with Raspbian OS (32 bit)	
interface	100 MBit Ethernet, USB 2.0
compiler	GCC, standard version of delivered distribution
programming languages	C++
supported cameras	All cameras with GigE Vision® v1.2 and v2.0 standard, USB3 Vision™ cameras with Legacy Mode (USB2.0)

Figura 12: Especificaciones de la Raspberry Pi 3

La Tabla (*Figura 13*) muestra el uso de la CPU y el consumo de energía de la Raspberry Pi® 3 mientras obtiene constantemente imágenes de una cámara GigE. En la (*Figura 14*) se compara con otras micro computadoras [14].

Frame Rate	Data Rate	Power Consumption	CPU Usage
5 fps	95 Mbit/s	2.9 W	19 %
10 fps	191 Mbit/s*	3.8 W	8 %

Figura 13: Consumo acorde al Frame Rate de captura de video a través de una cámara IP (Raspberry Pi 3)

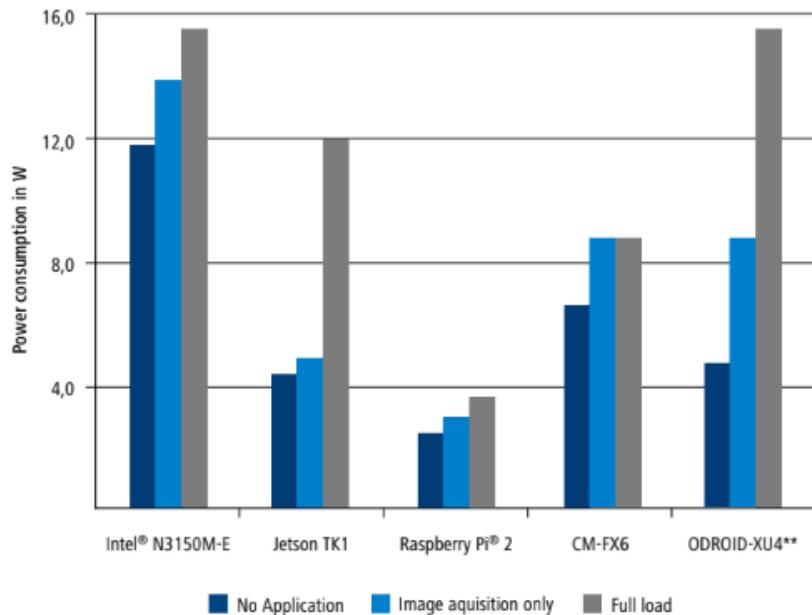


Figura 14: Consumo en Watts por modelo de computadora

2.1.4.1 Sistema operativo (Raspbian)

Raspbian es un sistema operativo gratuito basado en Debian optimizado para el hardware de la Raspberry Pi, lanzada en junio del 2012. Raspbian proporciona más que un sistema operativo puro:

viene con más de 35,000 paquetes, software precompilado incluido en un formato agradable para una fácil instalación [15].

Raspbian Lite es una versión de Raspbian sin interfaz gráfica y con los paquetes esenciales necesarios para que el hardware de Raspberry Pi funcione correctamente. Es elegido para correr sobre la IAS por su estabilidad y el soporte constante que recibe por parte de la comunidad, además de ser un sistema operativo creado específicamente para ejecutarse en la Raspberry Pi.

2.1.5. Cámara

Se utiliza la cámara BM-500GE de la Serie B (*Figura 15*) perteneciente a la empresa Japonesa JAI Ltd., especialistas en cámaras industriales de alto rendimiento [16].

La BM-500GE (*Figura 16*) es una cámara 2/3 " de escaneo progresivo monocromática con resolución de 5 megapíxeles (2456 x 2058) y una interfaz GigE Vision. Utiliza el sensor ICX625 de Sony y funciona a 15 cuadros por segundo en modo continuo a resolución completa.



Figura 15: Cámara JAI BM-500GE

La temperatura y humedad de trabajo se ubican entre -5°C a 45°C y 2% a 90% respectivamente, de allí la importancia del buen diseño del sistema de control de temperatura y humedad (THC) para el microsatélite, ya que el rango típico de temperaturas para la órbita LEO es de -170°C a 123°C.

El lente a utilizar en este prototipo, es de apertura manual.

Otras características:

- Sensibilidad: 0,34 Lux.
- Relación S/N: ≥ 50 dB (0 dB ganancia).
- Salida de imagen: GigE Vision, 8-bit, 10-bit o 12-bit.
- Ganancia: Manual/Automática. Rango -3 a 12 dB.
- Exposición programable: 2 ($64\mu s$) a 2072 (66,44ms) en pasos de 1.
- Alimentación: 12V - 24V DC +/-10%. 5,8W.
- Peso: 210 g.

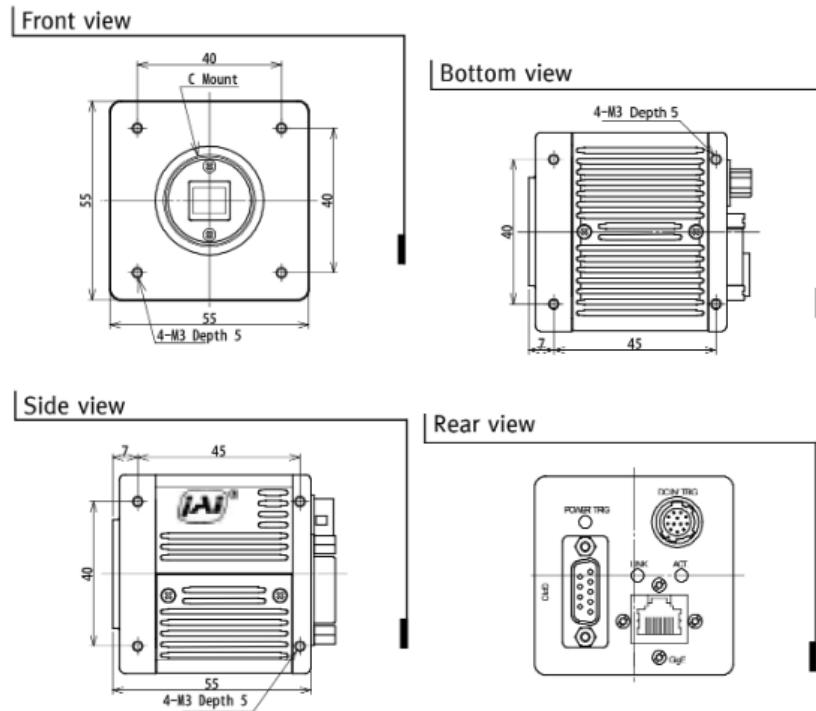


Figura 16: Dimensiones de la JAI BM-500GE

2.2. Protocolos y comunicaciones

Los siguientes protocolos se utilizan para comunicar las piezas de hardware que conforman el microsatélite. Por ejemplo, para la comunicación entre las computadoras y los transceptores, la IAS y la cámara y entre la IAS y la OBC.

2.2.1. Gigabit Ethernet

Es una ampliación del estándar Ethernet con una capacidad de transmisión de 1000 megabits por segundo contra los 100 megabits de Fast Ethernet.

Este estándar se utiliza para la conexión entre la cámara IP y la IAS del microsatélite.

Para obtener mejores resultados al utilizar la interfaz Gigabit Ethernet, se recomienda un tamaño MTU (es decir, el tamaño de los paquetes Ethernet individuales) de 9000 bytes debido a las grandes cantidades de datos de un sistema de cámara. Los tamaños más pequeños conducen a una mayor carga de trabajo de la CPU.

2.2.1.1 GigE Vision y GenICam

Es un estándar de interfaz a nivel global que utiliza el protocolo de comunicación Gigabit Ethernet. Fue desarrollado principalmente por miembros de AIA (Automated Imaging Association). GigE Vision

es capaz de transmitir grandes cantidades de datos de imágenes, video o control de dispositivos de forma rápida y sin comprimir a través de redes Ethernet, incluyendo redes inalámbricas GigE, 10 GigE y 802.11. Con GigE Vision, el hardware y el software de diferentes proveedores pueden interoperar sin problemas a través de las conexiones GigE.

GigE Vision es compatible con el estándar GenICam (Generic Interface for Cameras) mantenido por EMVA (European Asociación de visión artificial). El propósito de la norma GenICam es proporcionar una Interfaz estándar de programa que permite conectar cámaras de diferentes fabricantes en una plataforma, es decir, desacoplar la tecnología de interfaces de la cámara (GigE Vision, USB3 Vision, CoaXPress, Camera Link HS, Camera Link, etc.) de la interfaz de programación de aplicaciones (API) del usuario. Esto permite que un software de terceros utilice diferentes tecnologías para controlar las cámaras y adquirir datos de una manera independiente de la capa de transporte.

Las cámaras JAI GigE cumplen con los estándares GigE Vision y GenICam.

2.2.2. Comunicación Serie

La interfaz UART controla los puertos que poseen los microcontroladores para comunicaciones asíncronas en serie. Dicha interfaz usa dos o cuatro hilos que consisten en los pines RX y TX, y opcionalmente RTS y CTS.

Se utiliza este tipo de interfaz para la conexión OBC-IAS y OBC-EBV.

El modo de operación UART incluye las siguientes características:

- 8 o 9 bits de carga útil.
- Paridad impar, par o nula.
- Niveles de bits de inicio y fin configurables.
- Transferencia configurable de LSB o MSB.
- Interrupciones de recepción y transmisión independientes.
- Estado de error de paridad.

Una transferencia UART (*Figura 17*) consta de un bit de inicio, ocho bits de datos, un noveno dato opcional o bit de paridad, y uno o dos bits de parada.

Se debe tener en cuenta que los datos transferidos son conocidos como byte, aunque los datos en realidad pueden constar de ocho o nueve bits [17].

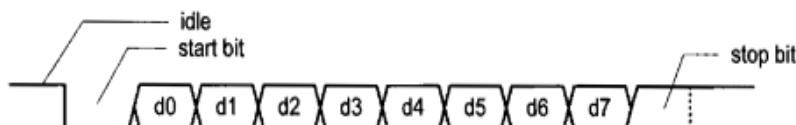


Figura 17: Configuración UART básica

2.2.3. SPI

La interfaz SPI es estándar de comunicación de datos en serie síncrono utilizado por los microcontroladores para comunicarse a altas velocidades entre ellos o con uno o más dispositivos periféricos en distancias cortas.

En una conexión SPI, siempre hay un dispositivo maestro (generalmente un microcontrolador) que controla los dispositivos periféricos. Por lo general, hay tres líneas comunes a todos los dispositivos:

- MISO (Master In Slave Out): la línea Esclavo para enviar datos al Maestro,
- MOSI (Master Out Slave In): la línea Maestra para enviar datos a los periféricos,
- SCK (Serial Clock): pulsos de reloj que sincronizan la transmisión de datos generada por el Maestro.

y una línea específica para cada dispositivo:

- SS (Slave Select): el pin en cada dispositivo que el Maestro puede usar para habilitar y deshabilitar dispositivos específicos.

Cuando el pin Slave Select de un dispositivo está bajo, se comunica con el Maestro. Cuando es alto, ignora al Maestro. Esto le permite tener múltiples dispositivos SPI que comparten las mismas líneas MISO, MOSI y CLK [18].

Se decide utilizar este protocolo para la comunicación entre la IAS y el transceptor del EAV, ya que brinda altas velocidades de transmisión necesarias para la descarga de imágenes.

2.2.4. Radiofrecuencia (RF)

Las ondas de radiofrecuencia se transmiten por el espacio electromagnético en el rango de frecuencia que varía entre los 30 kHz y los 300 GHz. Se generan aplicando una corriente alterna a una antena.

La modulación por radiofrecuencia se utiliza en los transceptores presentados anteriormente.

La placa CC2544 es compatible con dos formatos de modulación: GFSK y MSK.

La modulación por desplazamiento de frecuencia gausiana (GFSK) es un tipo de modulación donde un 1 lógico es representado mediante una desviación positiva (incremento) de la frecuencia de la onda portadora, y un 0 mediante una desviación negativa (decremento) de la misma. Al utilizar GFSK, la velocidad de datos en las placas puede establecerse en 250 kbps, 500 kbps, 1 Mbps o 2 Mbps.

La modulación por desplazamiento mínimo (MSK) es un tipo de modulación por desplazamiento de frecuencia de fase continua. La placa puede transmitir datos a 250 o 500 kbps utilizando MSK [19].

Se decidió utilizar la modulación GFSK para igualar la velocidad de transmisión que obtenemos con el estándar SPI.

2.3. Herramientas de software

Las herramientas de software que se exponen a continuación permiten configurar y realizar pruebas con el hardware ya descrito y, en algunos casos, son necesarias para su correcta programación.

2.3.1. Code Composer Studio (CCS)

Code Composer Studio es un IDE que soporta los microcontroladores y procesadores embebidos de Texas Instruments [20].

Posee herramientas para desarrollar y depurar aplicaciones embebidas. Incluye un compilador de C/C++, entorno de compilación del proyecto, debugger, profiler, entre otras herramientas. Está basado en Eclipse y el principal aporte de Texas Instruments es un avanzado debugger embebido.

2.3.2. HalCoGen

Es una aplicación que permite configurar gráficamente periféricos, interrupciones, clocks y otros parámetros del microcontrolador Hercules [21].

Una vez establecidas las configuraciones deseadas, es posible generar parte del código fuente e importarlo en CCS.

2.3.3. IAR Embedded Workbench

Es un IDE completo para desarrollo de aplicaciones embebidas. Es considerado por muchos como el mejor sistema de herramientas de compilación y depuración de la industria [22].

Entre los procesadores que soporta, se encuentra el 8051, y se integra perfectamente con el sistema CC2544.

2.3.4. SmartRF Studio

Es una aplicación para Windows que se puede utilizar para evaluar y configurar dispositivos RF de baja potencia de Texas Instruments [23].

Permite corroborar rápidamente el correcto funcionamiento de los transceptores RF. Solo es necesario conectar las placas al PC a través del debugger y configurar unos pocos parámetros para realizar una prueba de transmisión y recepción. Algunos de estos parámetros, como el tipo de modulación y la frecuencia, deben ser iguales en ambas placas para que puedan comunicarse correctamente.

En la (*Figura 18*) se observa la configuración de los parámetros RF para la comunicación entre las placas (1). Se puede variar la frecuencia, el tipo de modulación, la potencia, etc. La placa puede configurarse como transmisora o la receptora (2). Además, se pueden variar otros parámetros como la visualización de los datos como texto o valores hexadecimales, la cantidad y el tamaño de los paquetes.



Figura 18: Configuración SmartRF Studio

2.4. Parámetros en fotografía (Exposición)

Se llama exposición a la cantidad de luz que se captura en una fotografía. Los tres parámetros que determinan la exposición de una fotografía son: apertura, ISO y velocidad. El equilibrio correcto de estos valores establecen el resultado de las fotografías.

2.4.1. Apertura del diafragma

El diafragma de una cámara es mecanismo que se encarga de regular la cantidad de luz que ingresa al lente. Cuanto más grande es la apertura del diafragma, más luz llega al sensor. Esta apertura se mide con una escala denominada número f cuyos valores comunes son: 1, 1.4, 2, 2.8, 4, 5.6, 8, 11, 16, 22, 32, 45.

Cuanto mayor es la apertura del diafragma y más luz se deja entrar, menor es el número f y viceversa.

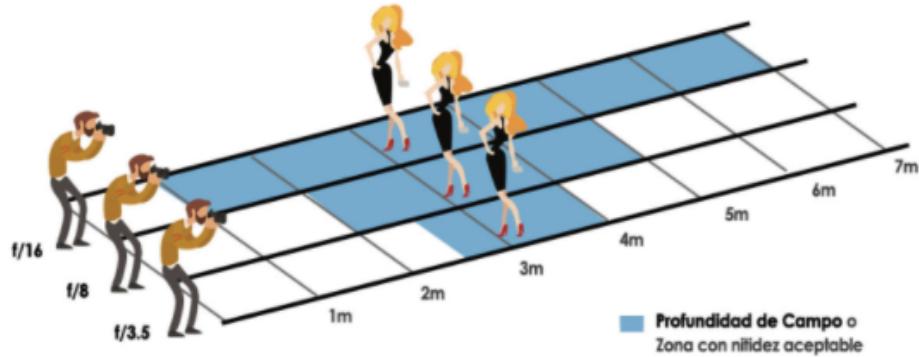


Figura 19: Profundidad de campo

Al abrir o cerrar el diafragma también se modifica la profundidad de campo (*Figura 19*). La profundidad de campo es la distancia entre el punto más cercano y el más lejano que aparecen nítidos en nuestro encuadre. A menor número f, menor profundidad de campo.

2.4.2. Velocidad de obturación

El obturador de una cámara es el dispositivo que permite controlar el tiempo que va a estar expuesto el sensor a la luz. Cuanto mayor es la velocidad de obturación, más rápido se abrirá y cerrará el obturador, por lo tanto, menos luz llegará al sensor y viceversa. Este parámetro se mide en segundos.

Se utilizan velocidades de obturación lentas cuando se desea capturar una fotografía en espacios con poca iluminación.



Figura 20: Velocidad de obturación

Al utilizar velocidades de obturación lenta con objetos en movimiento, se obtendrán imágenes borrosas o poco definidas a causa del movimiento y el tiempo que se captura ese objeto (*Figura 20*). Una velocidad de obturación rápida, permitiría congelar elementos en movimiento.

2.4.3. Sensibilidad ISO

La ISO representa la sensibilidad del sensor digital de la cámara. Su valor indica al sensor qué cantidad de luz debe absorber. Valores altos indican mayor absorción de luz por el sensor.

La sensibilidad se mide utilizando la escala ASA (ISO: 25, 50, 100, 200, 400, 800, 1600, 3200, 6400). Valores altos de ISO, provocarán ruido en las fotografías.

El ruido de una imagen crea un efecto granulado que elimina el detalle y suele estropear la captura (*Figura 21*).



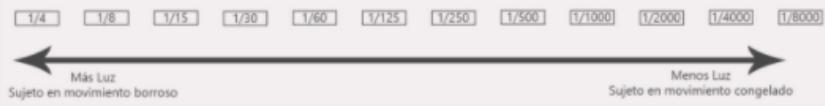
Figura 21: Sensibilidad ISO

En la (*Figura 22*) se observa un resumen de los parámetros presentados anteriormente [24].

Apertura de Diafragma



Velocidad de Obturación



Sensibilidad ISO



Figura 22: Parámetros en fotografía

Capítulo 3

3. Desarrollo

3.1. Parte 1: Proceso de desarrollo

En esta sección se establece de qué manera se va a afrontar el proyecto una vez definidos los requisitos, casos de uso y riesgos. Es decir, se definen una serie de pautas a tener en cuenta antes de comenzar a codificar, con el objetivo de organizar y guiar el desarrollo.

3.1.1. Control de versiones

Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos, como archivos de código fuente, a lo largo del tiempo, de modo que se puedan recuperar versiones específicas más adelante [25].

Dicho sistema permite regresar los archivos a versiones anteriores, regresar el proyecto completo a una versión anterior, comparar cambios a lo largo del tiempo y ver quién y cuándo los introdujo.

Ejemplos de herramientas de control de versiones son SVN, Git y TFS.

De acuerdo con la medida de control del riesgo R4, surge la necesidad de utilizar un sistema de control de versiones y un repositorio remoto para evitar la pérdida de información.

De esta manera, se crea un repositorio en GitHub [26] y se utiliza la herramienta SmartGit, que es un cliente gráfico Git con soporte para GitHub.

La estructura del repositorio consta de 4 carpetas principales, una para cada parte esencial del proyecto (*Figura 23*).

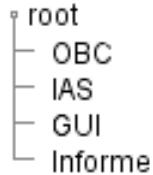


Figura 23: Estructura del Repositorio en GitHub

Dentro de cada una de las aplicaciones (OBC, IAS y GUI) se ubicaran los archivos en alguna de las siguientes carpetas según corresponda (*Figura 24*).

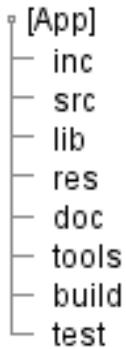


Figura 24: Estructura de cada aplicación del Repositorio en GitHub

3.1.2. Modelado y UML

El modelado es el diseño de las aplicaciones de software antes de codificar [27]. Es una parte esencial de los grandes proyectos de software y de gran ayuda en los pequeños y medianos.

El modelado es la única forma de visualizar un diseño y chequearlo contra los requerimientos antes de empezar a codificar. Se puede asegurar que la funcionalidad es completa y correcta, que las necesidades del usuario final son conocidas, antes que la implementación en el código haga los cambios difíciles y costosos de realizar.

El Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado de sistemas de software más utilizado actualmente. Usando alguna herramienta basada en UML, se pueden analizar los requerimientos de la futura aplicación y diseñar una solución que los cumpla, representando los resultados usando los diagramas estándar UML.

Estos diagramas se dividen en 3 categorías: diagramas de estructura estática de la aplicación, diagramas generales de comportamiento y diagramas de interacción.

PlantUML es una herramienta que permite escribir rápidamente diagramas UML como diagramas de secuencia, de casos de uso, de clases, de actividad, etc. Estos diagramas son definidos usando un lenguaje simple e intuitivo [28]. Las imágenes de los diagramas se pueden generar en PNG, SVG o en formato LaTeX.

3.1.3. Test Driven Development (TDD)

Del análisis del riesgo R3, con el objetivo de prevenir la mayor cantidad de errores, se decide usar una metodología de desarrollo basada en pruebas unitarias.

El desarrollo guiado por pruebas o Test-Driven Development (TDD) es una técnica de desarrollo de software que consiste en escribir la prueba de una funcionalidad antes de escribir la funcionalidad misma. Inicialmente, la prueba debe fallar pero inmediatamente se debe escribir el mínimo código posible que permita pasar la prueba. Seguidamente se refactoriza el código escrito.

Una de las ventajas de TDD es que el comportamiento deseado del código queda expresado en los tests. La idea es que los requisitos puedan ser traducidos a pruebas. De este modo, cuando las pruebas

pasen, se garantiza que el software cumple con los requisitos que se han establecido [29].

Los tests deben ser pequeños y también automatizados de manera que, con cada cambio de código, se puedan correr todas las pruebas. Si todas las pruebas pasan, se asegura que el nuevo código funciona pero también que es compatible con todo el código anterior.

3.1.3.1 TDD vs programación tradicional

En la forma tradicional de programar, a partir de los requisitos y su análisis, el código se diseña y luego se escribe dejando las pruebas para el final.

Sucede que es normal cometer errores durante el diseño y la codificación. Descubrir estos errores puede tomar demasiado tiempo y cuando se descubren, puede ser muy tarde para ayudar al desarrollador a aprender de sus errores. Además, los cambios que se fueron introduciendo hasta encontrar el error, pueden depender de este código defectuoso haciendo más difícil encontrar la raíz del problema.

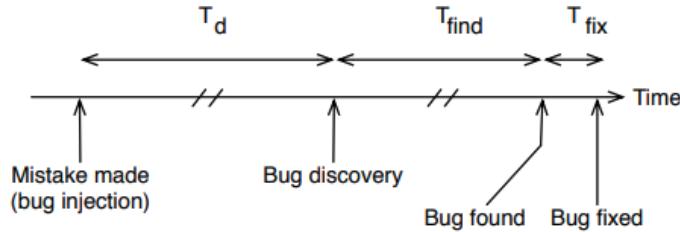


Figura 25: Proceso de descubrimiento de errores

Cuando el tiempo para descubrir el bug (T_d) se incrementa (*Figura 25*), el tiempo para encontrar la causa raíz del defecto (T_{find}) también se incrementa. El tiempo para arreglar el bug (T_{fix}) muchas veces no depende de T_d . Pero si otro código fue construido sobre el error, T_{fix} se puede incrementar dramáticamente.

Con TDD los requisitos se traducen en tests que prueban la funcionalidad. El código se basa más en pasar las pruebas que en un análisis o diseño.

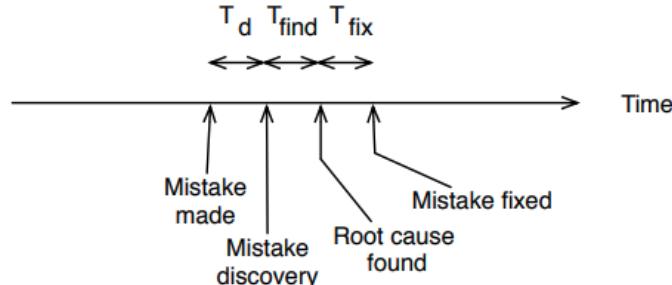


Figura 26: Descubrimiento de errores con TDD

Al usar TDD (*Figura 26*), el tiempo para descubrir el bug (T_d) se aproxima a cero. Además, el

tiempo para encontrar el bug (T_{find}) también se aproxima a cero. Un problema en el código recién introducido suele ser obvio y, si no lo fuera, siempre se puede volver a un sistema funcional al deshacer el último cambio.

En comparación, TDD entrega feedback de los errores inmediatamente y se previenen bugs mientras que utilizando la forma tradicional de programar se puede desperdiciar mucho tiempo.

3.1.3.2 Ciclo TDD

Aplicar TDD (*Figura 27*) es básicamente repetir un ciclo de pequeños pasos para cada requisito:

1. Codificar un test pequeño para el requisito elegido.
2. Correr todos los test y ver que el nuevo test falla, ya que todavía no existe código que implemente el requisito.
3. Realizar la menor cantidad de cambios para pasar el test. Se debe implementar solo lo necesario para cumplir con el requisito.
4. Correr todos los test y verificar que el nuevo pasa exitosamente.
5. Refactorizar o, en otras palabras, realizar cambios que mejoren el diseño del software manteniendo su comportamiento. Principalmente se busca eliminar código duplicado y hacer el código más claro y fácil de mantener.
6. Actualizar la lista de requisitos eliminando el recién implementado y agregando otros si fuera necesario.

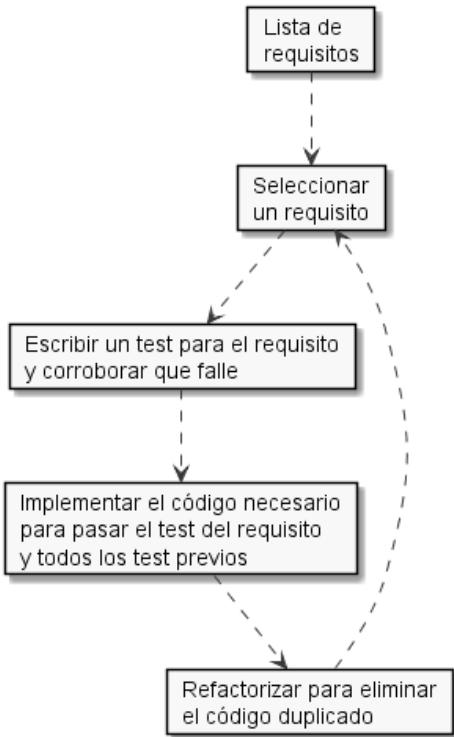


Figura 27: Ciclo TDD

3.1.3.3 Tests unitarios

Una de las primeras cosas a definir al iniciar un proyecto basado en TDD es un framework para escribir tests. Existen muchos para escribir tests en C++, la mayoría de ellos muy similares, y no es fácil elegir el mejor, si es que hay alguno. Elegimos Google Test ya que sus características cumplen con los requisitos necesarios para nuestra aplicación y, además, tiene un muy buen soporte para crear objetos mock. Se integra al proyecto al incluir sus archivos fuente que se compilan con el resto del código.

3.1.4. Análisis estático

El análisis estático de software se realiza sin ejecutar el programa. En general se realiza con herramientas automáticas como parte del proceso de integración continua y tiene como objetivo detectar problemas poco evidentes que afectan a la seguridad, memoria y rendimiento.

A raíz de la medida de control del riesgo R4, se decide utilizar una de estas herramientas. Cppcheck se centra en detectar comportamientos indefinidos y construcciones de codificación peligrosas para el lenguaje C/C++. Detecta principalmente tipos de errores que los compiladores no.

3.1.5. Análisis dinámico

Las herramientas de análisis dinámico de software proporcionan información en tiempo de ejecución del código del software. Se utilizan normalmente para identificar errores que pasan desapercibidos en un análisis estático, tales como la identificación de punteros no asignados, la comprobación aritmética de punteros y el control de la asignación, uso y liberación de memoria.

Otra de las características de este tipo de herramientas y el principal motivo por el que se decide utilizar una de ellas es detectar “leaks” de memoria, tal como indica la medida de control del riesgo R8.

Para detectar errores de memoria, se decide utilizar la herramienta Memcheck, que puede detectar problemas comunes en los programas C/C++ como accesos indebidos a memoria, utilización de valores indefinidos, liberación incorrecta de memoria, leaks de memoria, entre otros.

3.1.5.1 Cobertura de código (Test Coverage)

La cobertura de código es una medida porcentual que indica el grado en el cual el código fuente de un programa se ejecuta al correr sus tests. Es de utilidad para detectar partes críticas del código que no fueron probadas o código inalcanzable, ya que además de dar un resultado general, la mayoría de las herramientas puede indicar exactamente qué líneas del código fueron ejecutadas y cuáles no.

Es una medida de calidad en el sentido de que un programa con alta cobertura tiene más código probado durante sus tests y por lo tanto una menor probabilidad de contener bugs no detectados comparado con un programa con una cobertura menor.

Existen varios criterios de cobertura, siendo algunos de los principales:

- Cobertura de funciones (Function coverage): indica qué porcentaje de las funciones o subrutinas del programa han sido llamadas.
- Cobertura de línea (Statement coverage): indica qué porcentaje de las líneas de código del programa han sido ejecutadas.
- Cobertura de ramas (Branch coverage): indica si cada rama de una estructura de control (if-else o case) han sido ejecutados.

En la práctica para lograr una certificación de producto se suele requerir un determinado nivel de cobertura. Cada vez que un programador introduzca una clase sin un unit test, el porcentaje de cobertura debería bajar, lo cual fuerza de cierta manera a realizar unit test del código que se introduce para mantener el porcentaje de cobertura deseado.

Para generar reportes de cobertura en HTML se utiliza la herramienta gcovr.

3.1.6. Definición de Hecho (Done)

Para agregar una nueva funcionalidad al proyecto se sigue una serie de pautas para organizar el desarrollo e indicar cuándo se satisface un requerimiento.

Para comenzar, tal como indica el ciclo TDD visto anteriormente, se debe seleccionar un requerimiento. En esta instancia, con la ayuda de la herramienta PlantUML, se sugiere realizar el diseño de la solución y corroborar que pueda satisfacer completamente el requerimiento seleccionado. Los diagramas que explican los módulos de software, forman parte de la documentación y deben versionarse junto al código fuente. Luego de finalizado el ciclo TDD, se utilizan las herramientas de análisis estático y dinámico de código seleccionadas para determinar si es necesario realizar correcciones de problemas que no se detectaron previamente con los tests unitarios. Posteriormente, se deben ejecutar los tests nuevamente para verificar que el código continúa funcionando correctamente y mantiene el nivel de cobertura deseado.

3.2. Parte 2: Protocolo y EBV

3.2.1. Requerimientos a implementar

- Requerimientos de la GUI
 - RF2: Enviar comandos para interactuar con el microsatélite.
- Requerimientos de la OBC
 - RF9: Procesar los comandos que le pertenecen y reenviar los que le corresponden a la IAS o a la ES.
- Requerimientos de la IAS
 - RF10: Recibir comandos de la OBC, procesarlos y responder a ellos si corresponde.
- Requerimientos no funcionales del sistema
 - RNF1: La IAS y la ES deben utilizar un SO basado en GNU/Linux.
 - RNF2: La OBC debe utilizar el sistema operativo FreeRTOS.
 - RNF3: Las órdenes de la ES hacia la OBC se deberán enviar por un enlace UART.
 - RNF4: Las órdenes al satélite, respuestas e imágenes en baja resolución deberán transmitirse por el EBV, usando transceptores RF desarrollados previamente [6].
 - RNF5: La OBC se comunicará con la IAS a través de un enlace UART.

3.2.2. Investigación y desarrollo

3.2.2.1 Protocolo de comunicaciones

Para lograr que la ES y el satélite se comuniquen entre sí y transmitan información, se implementa un protocolo simple de comunicaciones para el enlace UART existente entre ellos basado en el protocolo HDLC (High-Level Data Link Control).

Sus paquetes inician con 0x7E (01111110) y terminan con 0x7D (10111110, porque RS-232 transmite el LSB primero). Cuando aparece una de estas secuencias en los datos a enviar (payload), se envía primero 0x7D, seguido de el byte original con el quinto bit cambiado. Por ejemplo, el byte 0x7E se transmite como 0x7D 0x5E ("10111110 01111010") y el byte 0x7D se transmite como 0x7D 0x5D ("10111110 10111010").

De esta manera se asegura que 0x7E nunca se encuentra en los datos del paquete y solo se usa para marcar el inicio de un paquete. El receptor deberá buscar los flags 0x7D, sacarlos de la secuencia e invertir el bit 5 del siguiente byte.

Cada paquete consta de siete campos que se describen en la *Tabla 13*.

Campo	Tamaño	Descripción
SYNC	1 byte	Flag de inicio (0x7E).
ID	2 bytes	Número único incremental para identificar el paquete.
CMD	1 byte	Código del comando.
LEN	1 byte	Tamaño del payload.
PAYLOAD	n bytes	Carga útil del paquete.
CRC	2 bytes	Verificación por redundancia cíclica.
END	1 byte	Flag de finalización del paquete (0x7D).

Tabla 13: Estructura de los paquetes.

El tamaño del protocolo es variable y cada paquete tiene un mínimo de 8 bytes (cuando no existe payload) y un máximo de 263 bytes cuando el payload tiene su tamaño máximo (255 bytes dado por el tamaño del campo LEN)

3.2.2.2 Comandos y acciones

Luego de definir los detalles del protocolo, se procede a definir una arquitectura de software que permita ejecutar acciones según los comandos recibidos.

Se identifican rápidamente dos entidades principales: comandos y acciones. Los comandos son estructuras simples que contienen los datos a enviar o recibidos por el protocolo y un identificador para saber qué hacer con ellos. Una acción es un objeto que se crea al recibir un paquete y sabe qué hacer con los datos recibidos (los procesa en una función "execute"). También se necesita una entidad que conozca los detalles del protocolo y permita construir un paquete a partir de un comando y viceversa. Esta entidad recibe el nombre de PacketAndCommandBuilder. Otra entidad es la necesaria para "fabricar" las Acciones correspondientes a los comandos recibidos. Esta entidad es compleja ya que contiene todas las referencias a los objetos que necesitan los distintos tipos de acciones y se denomina ActionFactory (*Figura 28*).

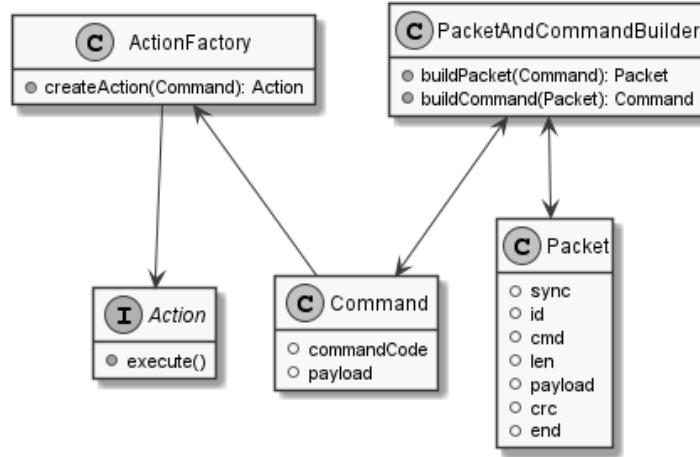


Figura 28: Relación entre comandos y acciones

Tanto para los comandos como para las acciones, se decide utilizar buffers que permitan regular la velocidad en la que se mandan paquetes o se ejecutan acciones, entre otras cosas (*Figura 29*).

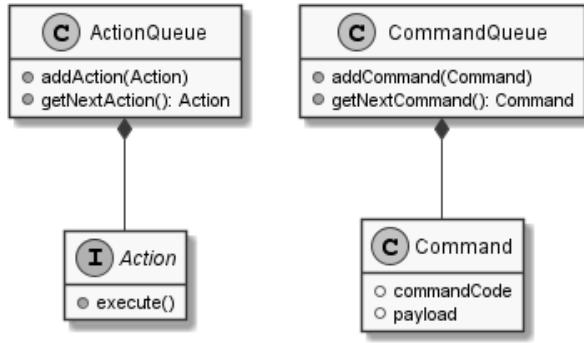


Figura 29: Buffers de comandos y acciones

La principal entidad de las presentes en la comunicación es el CommManager. Una de sus principales funciones es de tomar un comando del buffer y enviar el paquete correspondiente. La otra, es recibir los paquetes, identificar el comando y fabricar la acción correspondiente usando el ActionFactory o realizar el forwarding hacia otro puerto dependiendo el tipo de comando. Otra entidad llamada ActionManager es la encargada de tomar acciones del buffer y ejecutarlas (*Figura 30*).

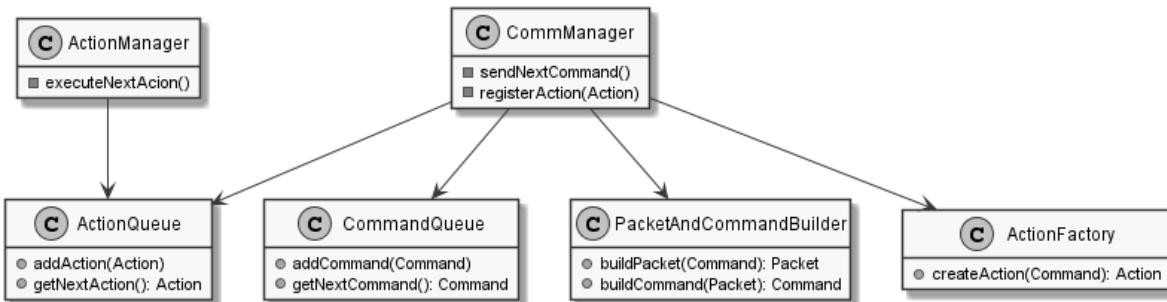


Figura 30: CommManager y ActionManager

Se deja abierta la posibilidad de utilizar diferentes hilos de ejecución. Por ejemplo, uno podría encargarse de enviar los comandos, otro podría recibirlas y crear las acciones, otro podría encargarse de ejecutar las acciones, etc.

Adicionalmente, se crean clases auxiliares para controlar puertos serie, facilitar la lectura de los paquetes y para validar paquetes. Estas son SerialPortManager, PacketReader y PacketValidator respectivamente.

Esta arquitectura se utiliza en las tres computadoras (IAS, ES, OBC), pero la implementación difiere ligeramente debido al sistema operativo y el lenguaje de programación utilizados en cada una.

3.2.2.3 Comandos para la OBC y forwarding

Como los objetivos están centrados en la comunicación con la IAS, la mayoría de los comandos tienen como destino esta computadora y, por lo tanto, la OBC deberá realizar el forwarding de ellos.

Con la finalidad de probar el protocolo, se definen comandos de prueba para prender y apagar leds de las computadoras, así como también comandos que permitan obtener el estado de estos leds.

Para configurar u obtener el estado del led de la OBC, el comando que llega debe ser procesado en esa computadora. En este caso, no se realiza un forwarding y la IAS no interviene en el circuito de comunicación. Por el contrario, para realizar estas acciones con el led de la IAS, sí se necesita un forwarding de la OBC (*Figura 31*).

Al obtener el estado del led de la IAS, se prueba el correcto funcionamiento del envío y recepción de paquetes, además del forwarding en ambos sentidos (OBC-IAS e IAS-OBC). Esto es porque la información parte de la ES hacia la OBC. Luego, la OBC realiza un forwarding de ese comando hacia la IAS. La IAS procesa el comando y envía un nuevo comando de respuesta hacia la OBC. La OBC realiza un nuevo forwarding enviando el paquete de respuesta de la IAS hacia la ES. Finalmente la ES recibe el comando de respuesta correspondiente al paquete de consulta que envió previamente.

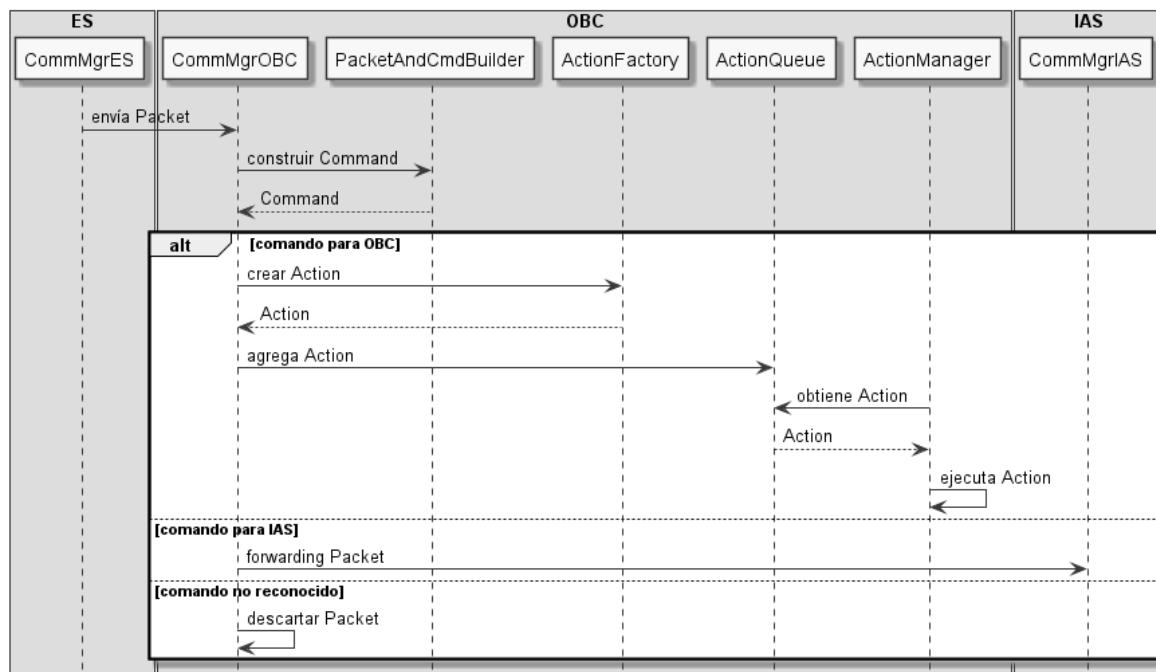


Figura 31: Mensaje enviado desde la ES

3.2.2.4 Detalles de los comandos

A continuación, se detallan los comandos de prueba junto con el resto de los comandos presentes en el protocolo:

KEEPALIVE REQ
Código: 0x10
Carga útil: No tiene.
Descripción: Solicita el checksum del catálogo. De esta forma la ES puede comprobar si su catálogo es exactamente el mismo que el del IAS. Este comando se envía periódicamente y por eso sirve también para verificar que el enlace con la IAS no está roto.

Tabla 14: Comando Keep Alive Req

KEEPALIVE RES
Código: 0x11
Carga útil: catalog checksum (64 bytes)
Descripción: Devuelve el checksum del catálogo.

Detalles: El algoritmo utilizado para calcular el hash es Keccak_512.

Tabla 15: Comando Keep Alive Res

CROP REQ
Código: 0x20
Carga útil: id imagen (2 bytes) X1 (2 bytes) Y1 (2 bytes) width1 (2 bytes) height1 (2 bytes) ... Xn (2 bytes) Yn (2 bytes) widthn (2 bytes) heightn (2 bytes)
Descripción: Indica los recortes que deben hacerse a una imagen. Cada recorte se registra en el catálogo como una imagen nueva.
Detalles: Cada recorte está por el rectángulo formado por los puntos (x,y), (x+width, y), (x,y+height) y (x+width, y+height). Se pueden indicar un máximo de 31 recortes por imagen debido al tamaño máximo del payload.

Tabla 16: Comando Crop Req

IMAGE HQ REQ
Código: 0x30
Carga útil: id de imagen (2 bytes)
Descripción: Sigue la solicitud de una imagen en alta calidad.

Tabla 17: Comando Image HQ Req

IMAGE HQ RES
Código: 0x31
Carga útil: id imagen (2 bytes) image packet number (4 bytes) total packets number (4 bytes) image data (n bytes)
Descripción: Envía datos de una imagen solicitada en alta calidad.
Detalles: Hasta 245 bytes de datos.

Tabla 18: Comando Image HQ Res

IMAGE LQ REQ
Código: 0x32
Carga útil: id de imagen (2 bytes)
Descripción: Sigue la solicitud de una imagen en baja calidad.

Tabla 19: Comando Image LQ Req

IMAGE LQ RES
Código: 0x33
Carga útil: id imagen (2 bytes) image packet number (4 bytes) total packets number (4 bytes) image data (n bytes)
Descripción: Envía datos de una imagen solicitada en baja calidad.
Detalles: Hasta 245 bytes de datos.

Tabla 20: Comando Image LQ Res

CATALOG REQ
Código: 0x40
Carga útil: No tiene.
Descripción: Sigue la solicitud del catálogo.

Tabla 21: Comando Catalog Req

CATALOG RES
Código: 0x41
Carga útil: id imagen (2 bytes) size (8 bytes) width (2 bytes) height (2 bytes) timestamp (8 bytes)
Descripción: Envía una imagen del catálogo.

Tabla 22: Comando Catalog Res

DELETE REQ
Código: 0x50
Carga útil: id de imágenes (2*n bytes)
Descripción: Borra las imágenes solicitadas.
Detalles: Máximo de 127 imágenes.

Tabla 23: Comando Delete Req

CAPTURE REQ
Código: 0x60
Carga útil: No tiene.
Descripción: Sigue la solicitud de una imagen.

Tabla 24: Comando Capture Req

LED OBC GET REQ
Código: 0x70
Carga útil: No tiene.
Descripción: Pregunta el estado del LED de la OBC.

Tabla 25: Comando Led OBC Get Req

LED OBC GET RES
Código: 0x71
Carga útil: led status (1 byte)
Descripción: Indica si el led de la OBC está prendido o apagado.
Detalles: Led status: 1 = "ON", 0 = "OFF".

Tabla 26: Comando Led OBC Get Res

LED OBC SET REQ
Código: 0x72
Carga útil: led status (1 byte)
Descripción: Prende o apaga un led de la OBC.
Detalles: Led status: 1 = "ON", 0 = "OFF".

Tabla 27: Comando Led OBC Set Req

LED IAS GET REQ
Código: 0x80
Carga útil: No tiene.
Descripción: Pregunta el estado del LED de la IAS.

Tabla 28: Comando Led IAS Get Req

LED IAS GET RES
Código: 0x81
Carga útil: led status (1 byte)
Descripción: Indica si el led de la IAS está prendido o apagado.
Detalles: Led status: 1 = "ON", 0 = "OFF".

Tabla 29: Comando Led IAS Get Res

LED IAS SET REQ
Código: 0x82
Carga útil: led status (1 byte)
Descripción: Prende o apaga un led de la IAS.
Detalles: Led status: 1 = "ON", 0 = "OFF".

Tabla 30: Comando Led IAS Set Req

3.2.2.5 Desarrollo en OBC

El desarrollo en la OBC comienza al crear un proyecto con FreeRTOS en HALCoGen para el modelo de la placa TMS570LS31x (*Figura 32*).

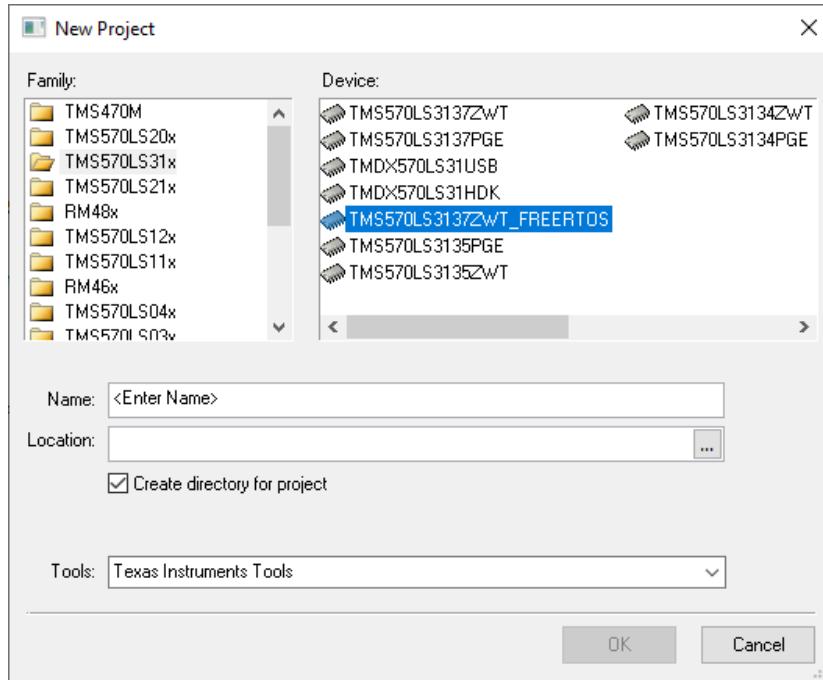


Figura 32: Nuevo proyecto en HALCoGen

Luego, se debe editar la configuración por defecto acorde a las características de hardware a utilizar y generar el código (*Tabla 31*).

Configuraciones			
TMS570LS3137ZWT_FREERTOS		SCI	
Driver Enable	VIM Channel 0-31	SCI Global	SCI Data Format
Habilitar SCI2 driver	Habilitar interrupción 13 (LIN1 Level 0 - IRQ)	Habilitar RXINT (High Level)	Baudrate 9600, 1 stop bit, length 8, sin paridad

Tabla 31: Configuración del hardware

Tomando como base este código, en CCS se crea un proyecto en la misma ubicación. Se debe seleccionar TMS570LS3137 como "Target" y la conexión Connection XDS100v2 USB (*Figura 33*).

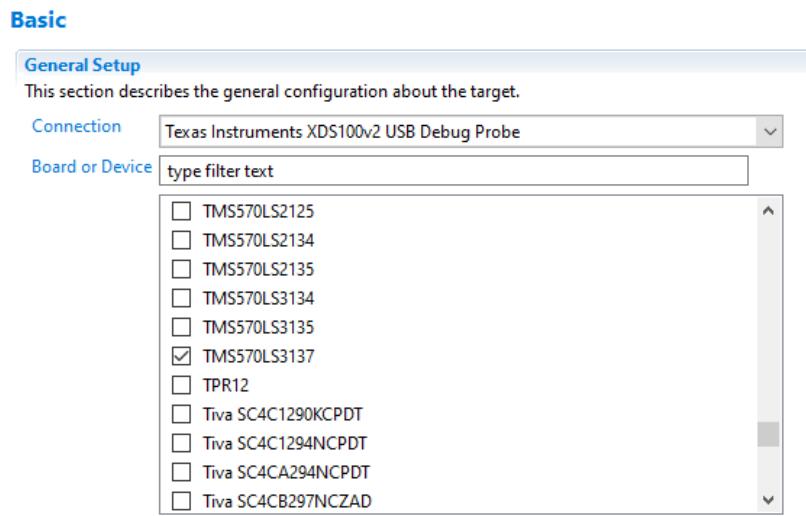


Figura 33: "Target Configuration" en CCS

En las propiedades del proyecto en CCS (Bild -> Arm Compiler -> Include Options) se debe agregar la carpeta "include" generada por HALCoGen.

Se pueden modificar los archivos generados anteriormente entre los comentarios /* USER CODE BEGIN */ y /* USER CODE END */. De esta forma, si se edita el proyecto en HALCoGen para habilitar o inhabilitar características de hardware, el código agregado entre esos comentarios no será sobrescrito al generarla nuevamente con las características modificadas.

Una vez configurado el proyecto en CCS, se implementa la arquitectura de comandos y acciones anteriormente descripta en el lenguaje C y el sistema operativo FreeRTOS.

3.2.2.6 Desarrollo en ES e IAS

Se decide utilizar el lenguaje de alto nivel C++ para el desarrollo en la ES y la IAS ya que, en este caso, se cuenta con la ventaja de programar en entorno Linux. Esto permite implementar la arquitectura utilizando programación orientada a objetos (OOP) y bibliotecas de desarrollo como Qt.

Qt es un framework escrito en C++ que extiende ese lenguaje con funciones como Signals y Slots. El propio framework y las aplicaciones que lo utilizan pueden ser compilados por cualquier compilador compatible con C++ estándar como GCC.

Junto con este framework se instala Qt Creator, un IDE multiplataforma que está pensado para facilitar el uso de la API de Qt. Se decide utilizarlo ya que cuenta con muchas características útiles como ejecución automática de tests, análisis estático de código con Cppcheck, análisis dinámico de código con Memcheck, integración con Git y un debugger útil para encontrar bugs de software.

Se decide estructurar el proyecto en dos subproyectos. Uno contiene la implementación del protocolo y las entidades que interactúan en la comunicación. El otro contiene los mismos elementos que el primero más el framework Google Test y las pruebas unitarias (*Figura 34*).

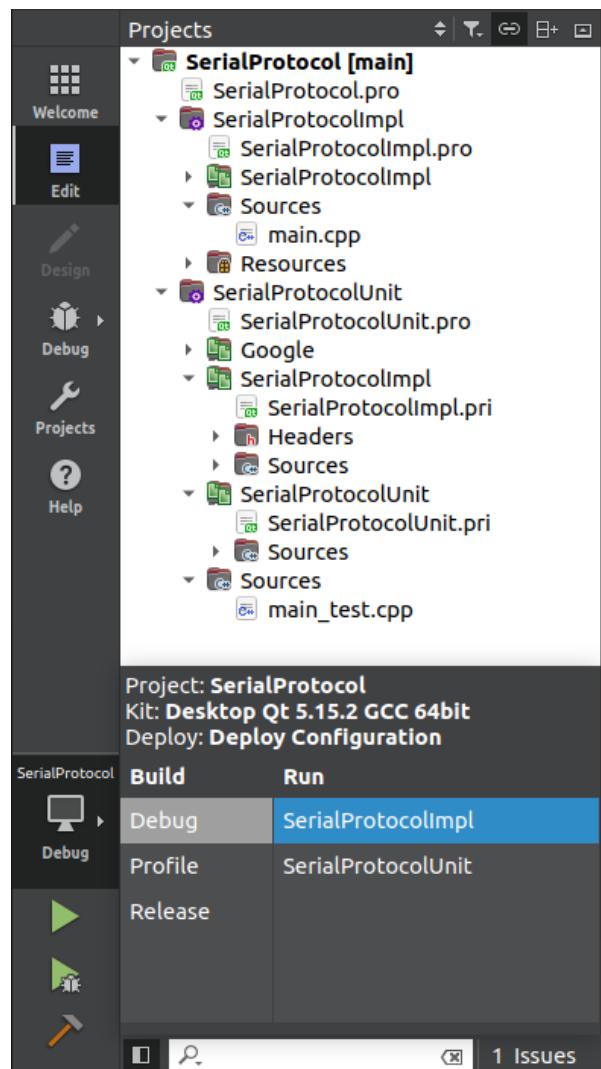


Figura 34: Estructura del proyecto en Qt-Creator

3.3. Parte 3: Captura y recorte de imágenes

3.3.1. Requerimientos a implementar

En esta sección se abordan una serie de requerimientos funcionales y no funcionales relacionados a la captura y procesamiento de imágenes tanto en la ET como en el IAS.

- Requerimientos de la GUI
 - RF5: Seleccionar y recortar una o más ROI de una imagen del microsatélite.
- Requerimientos de la IAS
 - RF11: Capturar imágenes mediante la cámara monocromática.
 - RF12: Almacenar imágenes capturadas.
 - RF13: Mantener un catálogo actualizado de las imágenes.
 - RF18: Recortar las imágenes indicadas por el usuario.
- Requerimientos no funcionales del sistema
 - RNF6: La IAS deberá utilizar el protocolo GigE para controlar la cámara.
 - RNF9: El formato de las imágenes en baja resolución debe ser .jpg.
 - RNF10: El formato de las imágenes en alta resolución debe ser .bmp.

3.3.2. Investigación y desarrollo

3.3.2.1 Software para el control de la cámara

Un SDK (del inglés Software development kit, kit de desarrollo de software) es un conjunto de herramientas que le permite al programador crear una aplicación informática para un sistema concreto utilizando cierto lenguaje de programación y puede, también, incluir hardware para comunicarse con un determinado sistema embebido.

JAI proporciona un SDK propio de la marca como una interfaz de programación de aplicaciones (API), denominado JAI GigE Vision SDK & Control Tool. Lamentablemente, dicho SDK sólo tiene soporte para Windows. A raíz de esto, se realiza un búsqueda de SDK disponibles, existiendo una amplia gama, pero su gran mayoría no son libres y requieren licencias de pago.

Se decide utilizar el software “Baumer GAPI SDK para Raspberry Pi con Raspbian OS” de la compañía Alemana Baumer Optronic GmbH por cuatro motivos principales:

- Es de uso libre y gratuito.
- Está optimizado para procesadores ARM y, en particular, para Raspbian en Raspberry Pi.
- Es totalmente compatible con GenICam para una fácil integración de la cámara.
- Brinda soporte continuo con la versión actual del SDK.

Así, para la conexión, control y configuración de la cámara (exposición, resolución) se utiliza Baumer GAPI SDK con el estándar GigE Vision.

Baumer GAPI SDK

Baumer es líder a nivel internacional en el desarrollo y producción de sensores, codificadores, instrumentos de medición, así como componentes para el procesamiento automatizado de imágenes.

GAPI es la abreviatura de “Generic Application Programming Interface”. Con esta API, Baumer proporciona una interfaz para la integración y el control de cámaras con Gigabit Ethernet (GigE).

Baumer GAPI SDK ARMhf und AArch64	
ODROID-XU4 with Linux® Ubuntu® 18.04 (32bit)	
interface	GigE, USB 3.0
compiler	gcc 4.8 (ARMhf) or gcc 5.4 (AArch64) and greater
programming languages	C++11 and C99
supported standards	GenTL 1.5 SFNC and PFNC 2.3 GenAPI 3.2 GigE Vision® (v1.2, v2.0) USB3 Vision® v1.0.1
supported cameras	MXG/ MXU/ VLG/ VLG.I/ VLU/ PXU TXG/ EXG/ LXG/ HXG/ SXG/ VEXG/ VEXU VCXG/ VCXU/ VQXT/ VLXT All cameras with GigE Vision® v1.2 and v2.0 standard
supported and tested boards	Nvidia Jetson TK1, Tx1, Tx2, Nano, Xavier / Raspberry Pi 3 and 4, Odroid XU4, PINE64 ROCK64 and many more

Figura 35: Especificaciones Baumer GAPI SDK

GenICam (sección 2.2.1.1) y GenTL garantizan una fácil conexión de las cámaras GigE. GenTL (Generic Transport Layer) es la interfaz de la capa de transporte para enumerar cámaras, capturar imágenes y moverlas a la aplicación del usuario (*Figura 36*).

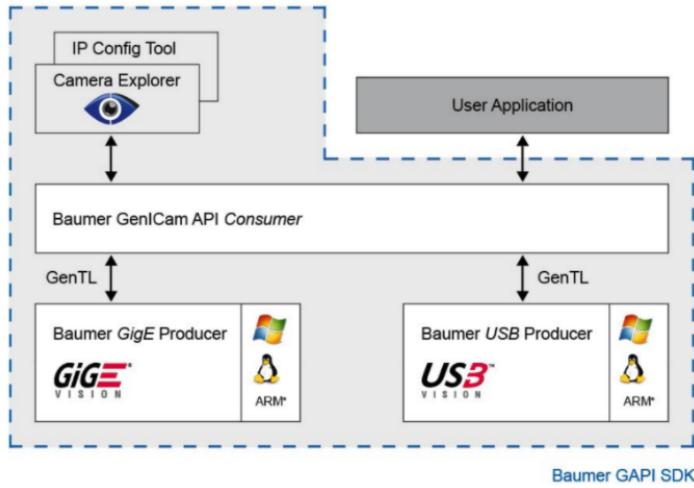


Figura 36: Descripción general de capas en Baumer GAPI

3.3.2.2 OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de software orientada a computer vision y machine learning de código abierto con licencia BSD [30]. La biblioteca tiene algoritmos optimizados que se pueden usar para detectar y reconocer caras, identificar objetos, rastrear objetos en movimiento, unir imágenes para producir una sola de alta resolución, encontrar imágenes similares dentro de una base de datos de imágenes, etc.

Se decide utilizar OpenCV ya que se puede integrar fácilmente con Baumer y permite procesar las imágenes capturadas mediante la cámara del microsatélite, almacenarlas y recortarlas posteriormente en caso de ser solicitado.

3.3.2.3 Exposición Manual vs. Exposición Automática

En general, las cámaras fotográficas permiten utilizar dos modalidades de captura en cuanto a exposición: automática y manual.

Al utilizar el modo automático, la cámara establece qué parámetros son los adecuados según la escena.

Sin embargo el modo de exposición automática brindada por Baumer, genera conflictos con la cámara JAI repercutiendo en el tiempo que demora en fijar los valores correctos para la exposición. Esto trae como inconveniente que el programa capture la imagen antes de que los parámetros lleguen a sus valores óptimos, por lo que siempre se obtiene una imagen subexpuesta o sobreexpuesta.

Por esta razón, se decide dejar de lado la exposición automática y usar sólo la exposición manual estableciendo como trabajo a futuro implementar la exposición automática.

3.3.2.4 Captura de fotografías

Clases principales de Baumer-GAPI

La idea fundamental de Baumer es liberar al programador de la necesidad de definir e instanciar todos los objetos necesarios y transferir estas tareas a la GAPI [31].

El sistema completo se divide en tres clases de objetos principales (*Figura 37*):

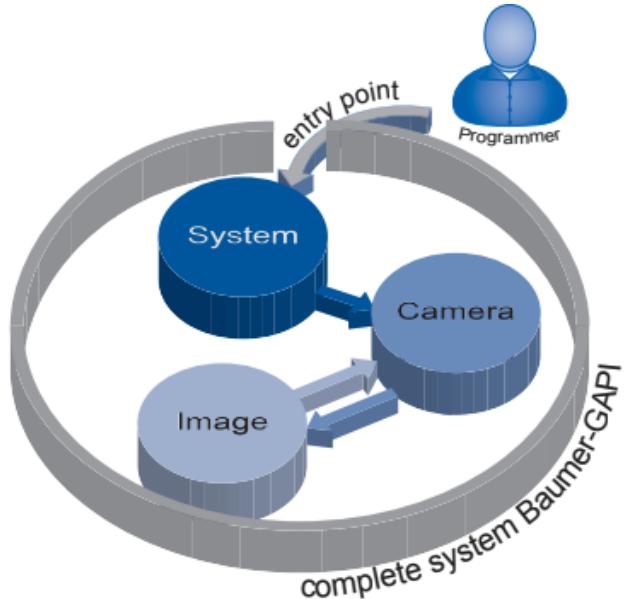


Figura 37: Estructura y punto de entrada a Baumer-GAPI.

- **Clase System:** representa la interfaz mediante la cual se pueden "conectar" varias cámaras. Los objetos *System* se pueden enumerar, crear, abrir y liberar.
- **Clase Camera:** representa el hardware de la cámara con el que se desea trabajar. Los objetos de *Camera* deben funcionar dentro de un objeto *System* - análogo al hardware que debe estar conectado a una interfaz - y pueden ser enumerados, creados, abiertos y liberados. Un objeto *Camera* puede agregar contenido a un objeto *Image*.
- **Clase Image:** son contenedores vacíos (en su inicio) para el contenido de una imagen. Estos contenedores se pueden llenar con información de imagen como ancho, alto y formato de píxel, y datos de imagen proporcionados por un objeto *Camera*.

Hay dos puntos de entrada al sistema para iniciar Baumer-GAPI. Estos puntos se refieren a la clase *System* que permite contar la cantidad de sistemas existentes o crear uno nuevo.

Software en la IAS

Cuando el usuario solicita capturar una fotografía, la ES envía una petición (*Tabla 24*) a la IAS. Al recibirla, la IAS ejecuta la acción encargada de realizar esa captura.

El SDK de Baumer proporciona varios ejemplos para controlar cámaras IP y procesar las fotos capturadas. Se decide implementar la acción encargada de capturar imágenes tomando como referencia el ejemplo “PixelTransformation”, aunque la base de todos estos modelos es similar.

En la (*Figura 39*) se observan las diferentes etapas por las que Baumer-GAPI atraviesa para obtener una fotografía a través de una cámara conectada, en este caso, mediante una interfaz Ethernet. En resumen, al iniciar el programa, se listan las interfaces disponibles en la computadora (Ethernet y USB 3.0), luego se abre la interfaz seleccionada y se realiza un escaneo para verificar si hay alguna cámara conectada a dichas interfaces. Una vez seleccionada la cámara, se crea un objeto de tipo *Image* y se rellena con los datos de la fotografía capturada.

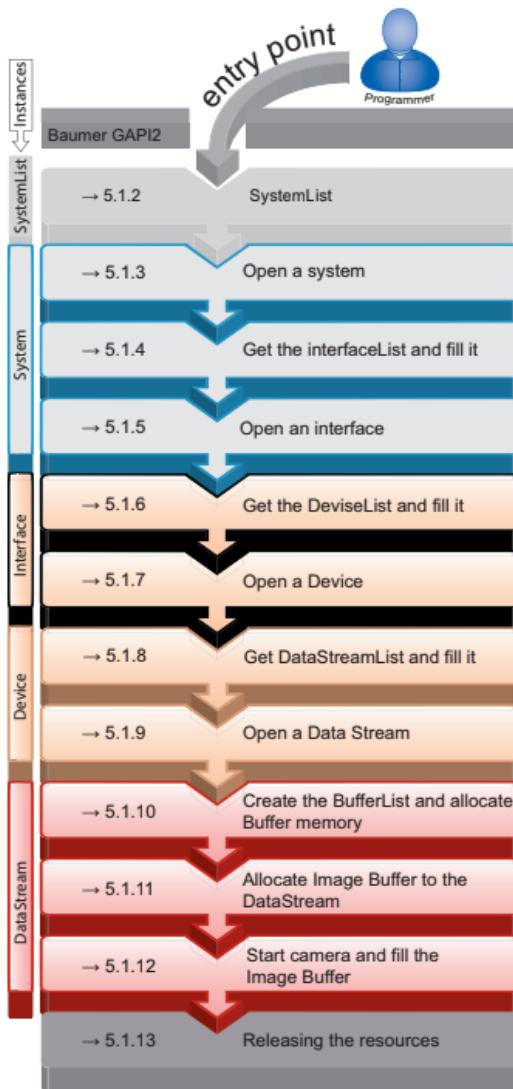


Figura 38: 11 pasos para obtener una fotografía mediante Baumer.

Al finalizar el proceso de captura, la imagen permanece almacenada en el buffer de fotogramas de Baumer-GAPI, es decir, en la memoria volátil de la computadora (RAM), por lo que es necesario

almacenarla posteriormente en disco duro para su permanencia en el tiempo. Para lograrlo, se utiliza OpenCV como librería intermediaria [32].

Luego de recibir los datos de la imagen en el buffer, se pueden asignar a una matriz OpenCV (Mat). Los datos del buffer de memoria no se copian, pero se pasa un puntero a la matriz OpenCV. Luego, esta matriz se puede almacenar en disco utilizando la función imwrite().

Se puede utilizar la biblioteca OpenCV según sea necesario para cambiar la resolución, formato y calidad de las imágenes antes de almacenarlas.

Una vez procesada y almacenada la imagen, se actualiza el catálogo de imágenes agregando una nueva entrada.

3.3.2.5 Recorte de imágenes en la ES

Uno o más recortes de una imagen se solicitan desde la ES indicando las ROI (*Tabla 16*). Para seleccionar estas áreas de interés, se utiliza la función selectROI de OpenCV. SelectROI es una parte de la API “Tracking” que se utiliza comúnmente para ubicar un objeto en sucesivos cuadros de un video.

Inicialmente se utilizan las funciones básicas de OpenCV para abrir imágenes desde un archivo.

Se crea una estructura que representa una ROI para almacenar las coordenadas X e Y correspondientes al vértice superior izquierdo, además del ancho y el alto de una región de interés. Los valores de esta estructura se adquieren al llamar la función *selectROI()* de OpenCV que permite al usuario seleccionar el área de interés.

Existe la posibilidad de seleccionar más de una región de interés por imagen. También se puede hacer zoom para realizar la selección de manera precisa. Éstas características utilizan interrupciones del sistema para detectar eventos de mouse y teclado.

Una interrupción es una señal recibida por el procesador de una computadora, para indicarle que debe *interrumpir* el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación [33].

Al presionar un botón del mouse, se genera una interrupción deteniendo el flujo de ejecución actual para atenderla. En el manejo de la interrupción, se detecta qué botón se presiona para saber qué acción llevar adelante. Así, si el botón presionado es el primario, se indica al programa que el usuario desea realizar una nueva selección de una región de interés. Si se detecta que la interrupción proviene del scroll del mouse, se indica al programa que el usuario desea realizar zoom a la imagen.

Zoom

Para lograr realizar zoom, se debe reescalar la imagen tantas veces como el usuario accione el scroll del mouse y, a su vez, almacenar la posición del puntero del mouse en el momento de la interrupción, ya que se debe lograr que el zoom se ejecute centrado en esa posición. Para abordar este problema, se realiza una interpolación lineal de puntos. Para mayor simplicidad, el zoom siempre será 2X la imagen. Es decir, el primer zoom será 2x la imagen original, el segundo será 4x la imagen original y así sucesivamente.

3.3.2.6 Recorte de imágenes en la IAS

Se utilizan funciones básicas de OpenCV para abrir y crear imágenes. La imagen y la lista de ROIs están presentes en los comandos que provienen desde la ES (*Tabla 16*). Se tendrán N estructuras ROI que representan N recortes o regiones de interés seleccionadas en una imagen.

Luego, se itera dicha lista para obtener los datos de las ROIs y se recorta la imagen original generando tantas subimágenes de salida como regiones de interés seleccionadas contenía la lista. Cada una de estas subimágenes se registra en el catálogo como una imagen nueva.

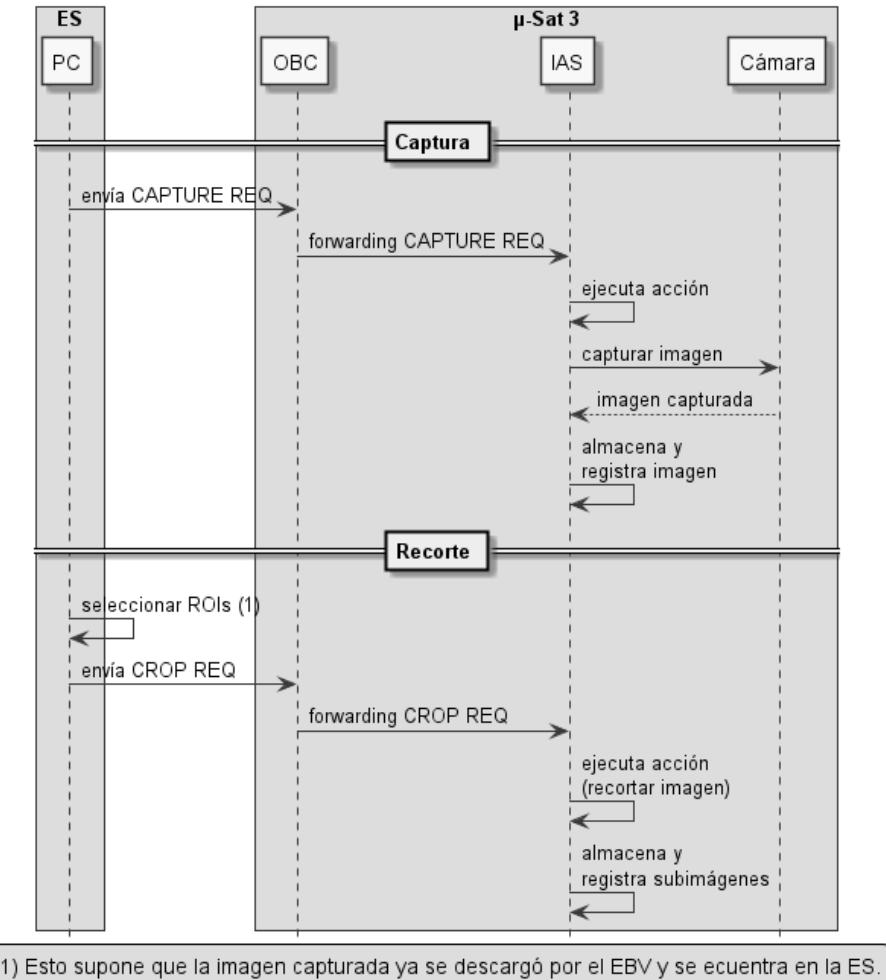


Figura 39: Secuencia de captura y recorte de una imagen.

3.4. Parte 4: Descarga de imágenes HQ y EAV

3.4.1. Requerimientos a implementar

En esta sección se propone una solución a los requerimientos funcionales y no funcionales relacionados a la transmisión de datos a través del EAV.

- Requerimientos de la IAS
 - RF15: Transmitir imágenes en alta resolución por el EAV.
- Requerimientos no funcionales del sistema
 - RNF7: Las imágenes en alta resolución deberán transmitirse por el EAV utilizando transceptores RF desarrollados previamente [7].
 - RNF8: El EAV debe ser unidireccional.

3.4.2. Investigación y desarrollo

Como se menciona en la sección 1.2.1, el desarrollo de la Práctica Profesional Supervisada se utiliza como base para el diseño e implementación del EAV.

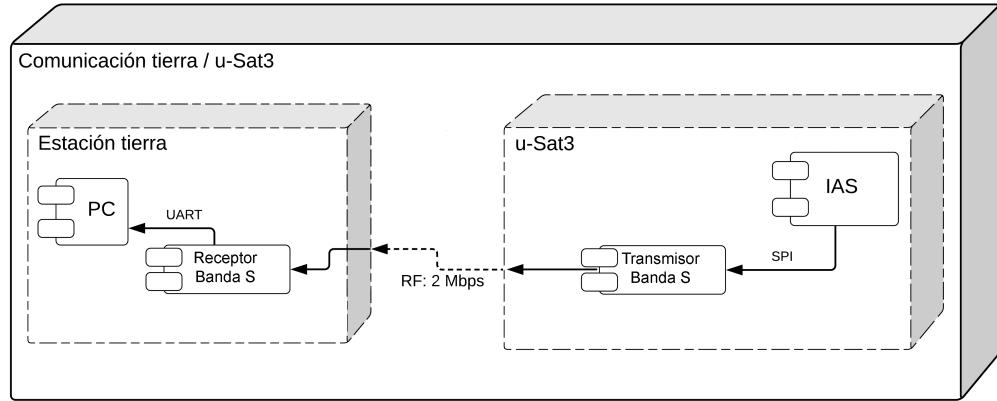


Figura 40: Arquitectura del EAV

El circuito está compuesto por cuatro dispositivos (*Figura 40*). El software previamente desarrollado en cada uno de ellos debe ser revisado y ajustado a fin de cumplir los requerimientos mencionados.

3.4.2.1 Software en la IAS

Al recibir una petición desde la ES para enviar una imagen en alta calidad (*Tabla 17*), la IAS debe ejecutar la acción encargada de desglosar la imagen solicitada en paquetes. Al ejecutarse dicha acción, se abre la imagen a transmitir en modo solo lectura y se cargan sus datos. La cantidad de paquetes

a transmitir se determina dividiendo el tamaño de la imagen por la máxima cantidad de bytes que se pueden transmitir por paquete. Luego, se generan todos los paquetes, cada uno con su número de orden y datos de imagen, y se colocan en la cola de transmisión del protocolo para su posterior envío.

3.4.2.2 Software del transmisor y receptor

El protocolo de comunicación diseñado en la Parte 2, proporciona una interfaz estándar de transmisión de paquetes que permite conectar hardware de distintos tipos, es decir, al utilizar el protocolo de datos UART, el medio por el que se transmiten los paquetes es indistinto. En este caso, en el EAV, la información se transmite a través de transceptores RF [7].

A partir del código existente correspondiente a las placas emisora y receptora para la comunicación RF, se procede con la revisión y refactorización del código.

Al ser un canal unidireccional, la transmisión de paquetes a lo largo de todo el EAV no posee un pedido de retransmisión por pérdida (ACK). Es por ello que las imágenes a transmitir por este enlace son de formato BMP, ya que si algún paquete se pierde, la imagen podrá reconstruirse indistintamente al no ser un formato de compresión.

La placa receptora se configura de manera análoga a la placa emisora, es decir, mismo tipo de modulación, frecuencia y tamaño de paquetes.

La información de un paquete del protocolo de comunicaciones está contenida en varios paquetes RF de tamaño fijo utilizados por los transceptores para enviar y recibir información.

El emisor posee un buffer auxiliar donde se van almacenando los bytes recibidos provenientes de la IAS. Cuando se recibe una cantidad de bytes igual al tamaño del paquete RF a enviar, éste se despacha completo por RF. En el receptor, si el paquete recibido contiene la misma longitud que el paquete esperado, se lo envía por UART a la ES, de lo contrario, se descarta.

3.4.2.3 Software en la ES

Para reconstruir la imagen solicitada, la ES ejecuta la acción que consiste en crear un archivo de imagen correspondiente al id que se encuentra en el primer paquete recibido (*Tabla 18*). Luego, para este primer paquete y los siguientes, se anexa la información de la imagen contenida en el payload al archivo.

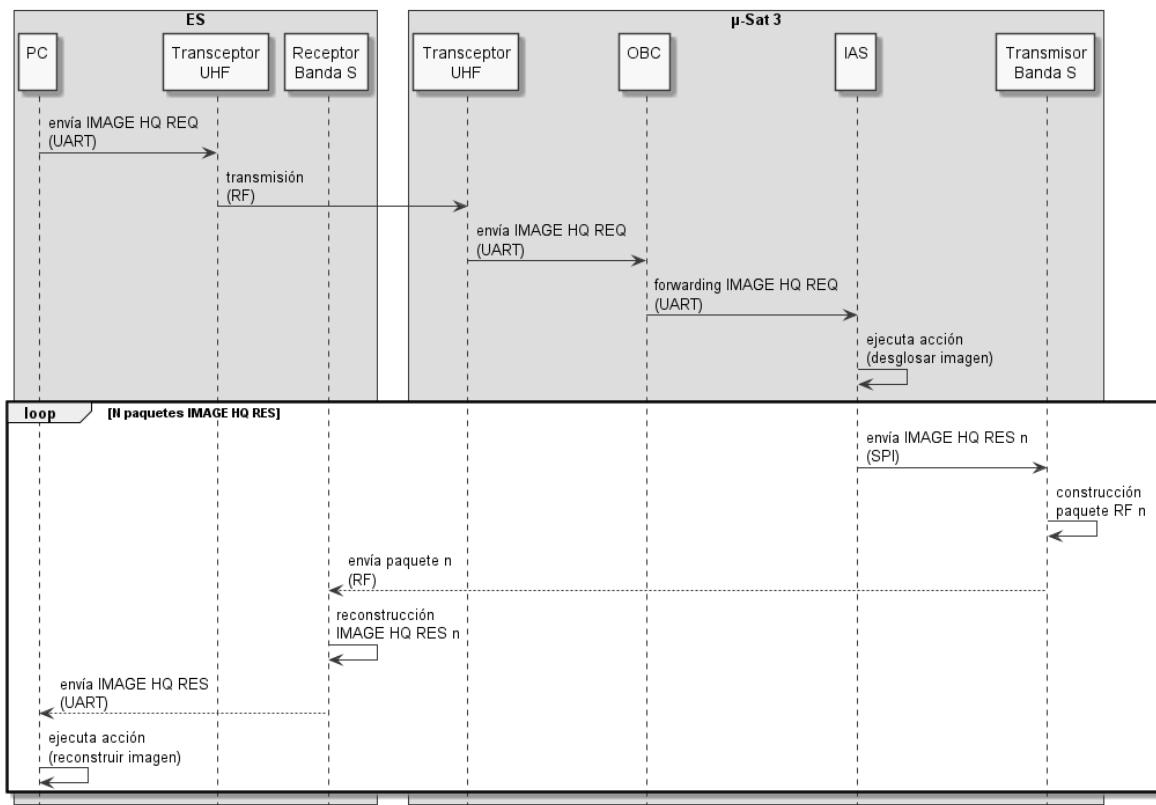


Figura 41: Transmisión de imagen en alta calidad

3.5. Parte 5: GUI e integración

3.5.1. Requerimientos a implementar

En esta sección se propone una solución a los requerimientos funcionales y no funcionales relacionados a la interfaz de usuario y la integración de ésta a los demás componentes que integran el sistema de transmisión de imágenes del microsatélite.

■ Requerimientos de la GUI

- RF1: Establecer la conexión con el microsatélite.
- RF3: Descargar el catálogo de las imágenes en el microsatélite.
- RF4: Descargar y previsualizar imágenes del catálogo en baja resolución.
- RF5: Seleccionar y recortar una o más ROI de una imagen del microsatélite.
- RF6: Visualizar información sobre el estado, errores y número de mensajes (enviados, recibidos y perdidos).
- RF7: Borrar imágenes en el microsatélite.
- RF8: Seleccionar y descargar imágenes en alta resolución.

■ Requerimientos de la IAS

- RF17: Borrar imágenes solicitadas por el usuario.

3.5.2. Investigación y desarrollo

Para crear la GUI, se utiliza el módulo QtQuick de Qt y el lenguaje QML. QML es un lenguaje declarativo basado en JavaScript que permite crear de manera muy simple interfaces de usuario. Aunque es posible escribir aplicaciones completas usando solamente QML, el backend de la aplicación se implementa en C++ y se utiliza QML solo para la GUI.

Como patrón de arquitectura se utiliza MVVM (en inglés, model–view–viewmodel). Este patrón se caracteriza por separar el desarrollo de la interfaz de usuario (vista o view) del desarrollo del backend o lógica de la aplicación (modelo o model). El viewmodel es el intermediario entre el modelo y la vista, contiene toda la lógica de presentación y expone a la vista propiedades y comandos. No contiene una referencia de la vista sino que posee propiedades enlazadas con ella. Cuando éstas propiedades cambian, la vista se actualiza. Qt permite exponer fácilmente propiedades, métodos y señales para realizar este enlace con la interfaz.

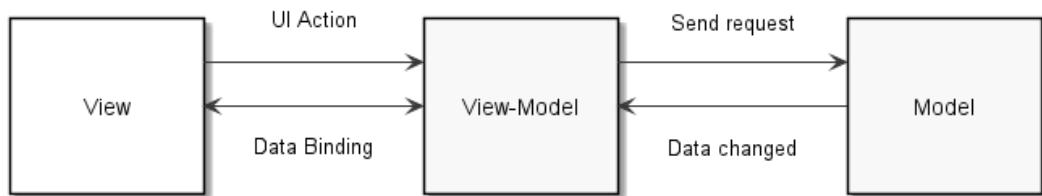


Figura 42: Model–view–viewmodel

3.5.2.1 Diseño de la GUI

Se diseña la GUI de la ES con un esquema de cinco secciones principales (*Figura 43*) que se detallan a continuación.



Figura 43: Esquema de la GUI

Puertos de conexión

Permite ingresar los puertos a los que están conectados el EBV y el EAV. Un transceptor RF, un cable serie o cualquier otro dispositivo se pueden conectar de forma indistinta a estos puertos, siempre y cuando representen conexiones UART.

Se brindan dos entradas de texto, uno por cada enlace de conexión.

Catálogo de imágenes

Es una tabla con información sobre las imágenes presentes en el microsatélite. Ésta contiene información como el tamaño de la imagen en bytes, el alto y ancho en píxeles, fecha de creación, entre otros. Las entradas de esta tabla se pueden seleccionar para aplicar distintas acciones a una o más imágenes, como por ejemplo descargarlas o recortarlas.

Previsualización de imagen

Permite previsualizar las imágenes seleccionadas en el catálogo. Si varias de ellas se seleccionan, se mostrará la última elegida. Además, es posible hacer zoom en la imagen para observar detalles con mayor precisión.

Mensajes de estado

Brinda un pantallazo general y rápido del estado del sistema. Se muestra información sobre el estado de la comunicación, posibles errores, paquetes perdidos, enviados y recibidos.

Panel de acciones

Es una columna de botones que permiten al usuario interactuar con el microsatélite y ejecutar sus funciones.

- Connect: Este botón se encuentra bloqueado hasta que se abran correctamente los puertos de conexión ingresados. Al accionarse, se comienzan a enviar y recibir los mensajes KEEPALIVE. Cuando se establece la conexión se habilitan el resto de botones.
- Sync Catalog: Activa la petición automática del catálogo cuando la ES detecta que se encuentra desactualizado.
- Capture Image: Captura una fotografía.
- Download Selected Samples: Descarga las imágenes del catálogo seleccionadas en baja calidad y formato jpg a través del EBV.
- Download Selected Images in HQ: Descarga las imágenes del catálogo seleccionadas en alta calidad y formato bmp a través del EAV.
- Crop Selected Images: Abre la última imagen seleccionada para elegir áreas de interés y recortarlas en la IAS.
- Delete Selected Images in IAS: Borra las imágenes del catálogo seleccionadas en la IAS.
- Open LQ Images Folder: Abre la carpeta que contiene las imágenes descargadas en baja calidad.
- Open HQ Images Folder: Abre la carpeta que contiene las imágenes descargadas en alta calidad.
- Update Catalog: Actualiza el catálogo de forma manual.

3.5.2.2 Desarrollo de la GUI

Siguiendo el patrón MVVM, se diseñaron elementos visuales (Views) como los botones o entradas del catálogo, combinando componentes básicos de QML como Button, Rectangle y Text. La información que muestran estos componentes son las propiedades (Q_PROPERTYs) de un ViewModel asociado a directamente ellos. El ViewModel exporta propiedades, pero no sabe cuál es o si existe la vista que se encarga de mostrarlas.

```
class InfoTextViewModel : public ViewModel
{
    Q_OBJECT
    Q_PROPERTY(QString title READ getTitle CONSTANT)
    Q_PROPERTY(QString value READ getValue NOTIFY valueChanged)
    [...]
};
```

Figura 44: ModelView y su Q_PROPERTY

Las Q_PROPERTYs tienen un tipo, un nombre y pueden ser de lectura o lectura/escritura y constantes o variables. Para leer y escribir una Q_PROPERTY se deben asociar funciones get y set respectivamente. Para actualizar una Q_PROPERTY se debe asociar una señal.

A su vez, los valores leídos o escritos por una Q_PROPERTY están conectados de alguna forma a datos del modelo conocido por el ViewModel. Para actualizar un valor de una Q_PROPERTY cuando el modelo se actualiza, se utilizan conexiones (función QObject::connect()) y señales.

Lo que se busca es conectar señales de un objeto (por ejemplo, el cambio de un dato del modelo) con funciones del mismo ViewModel que actualizan las Q_PROPERTYs según el dato del modelo recién actualizado.

```
ViewModelManager::ViewModelManager()
{
    [...]
    connect(commManagerES, &CommManager::sentMessagesChanged, this, &ViewModelManager::onSentMessagesChanged);
    [...]
}

void ViewModelManager::onSentMessagesChanged(quint32 total)
{
    if (m_mainScreenViewModel)
    {
        InfoTextViewModel *sentMessagesText = m_mainScreenViewModel->getSentMessages();

        if (sentMessagesText)
        {
            sentMessagesText->setValue(QString::number(total));
        }
    }
}

void InfoTextViewModel::setValue(const QString &value)
{
    if (value != m_value)
    {
        m_value = value;
        emit valueChanged();
    }
}
```

Figura 45: Uso de connect() para actualizar una Q_PROPERTY

Algunos de estos ViewModels se agrupan en listas que son mostradas en columnas usando el componente ListView de QML. Cada ListView cuenta con un modelo (datos a mostrar) que en este caso serían las listas de ViewModels (objetos de C++) y un delegate que define como éstos datos son mostrados. El delegate en este caso es la Vista encargada de mostrar específicamente esos ViewModels.

```

class AbstractScreenViewModel : public ViewModel
{
    [...]
    Q_PROPERTY(QList<QObject*> infoTextList READ getInfoTextList CONSTANT)
    [...]
}

ListView {
    [...]
    model: exposedViewModels.mainScreenViewModel.infoTextList
    delegate: Text {
        property InfoTextViewModel infoTextViewModel
        text: infoTextViewModel ? (infoTextViewModel.title + ": " + infoTextViewModel.value) : ""
    }
}

```

Figura 46: Lista de ViewModels y su visualización en QML

```

Comm Status: No connected
Error: None
Lost Messages: 0
Received Messages: 0
Sent Messages: 0
Downloaded Catalog Images: 0

```

Figura 47: Textos de información en la GUI

Para el caso de las vistas que modifican el modelo, por ejemplo, el recuadro para ingresar los puertos de conexión, se usan Q_PROPERTYs de lectura/escritura y funciones Q_INVOKABLE en los ViewModels. Luego, gracias a las señales asociadas a las Q_PROPERTYs, la vista se actualiza con la nueva información. De esta forma se logra la visualización, modificación y actualización de datos en la GUI.

En base al esquema anterior y la arquitectura planteada, se obtiene la primera versión de la GUI (*Figura 48*).

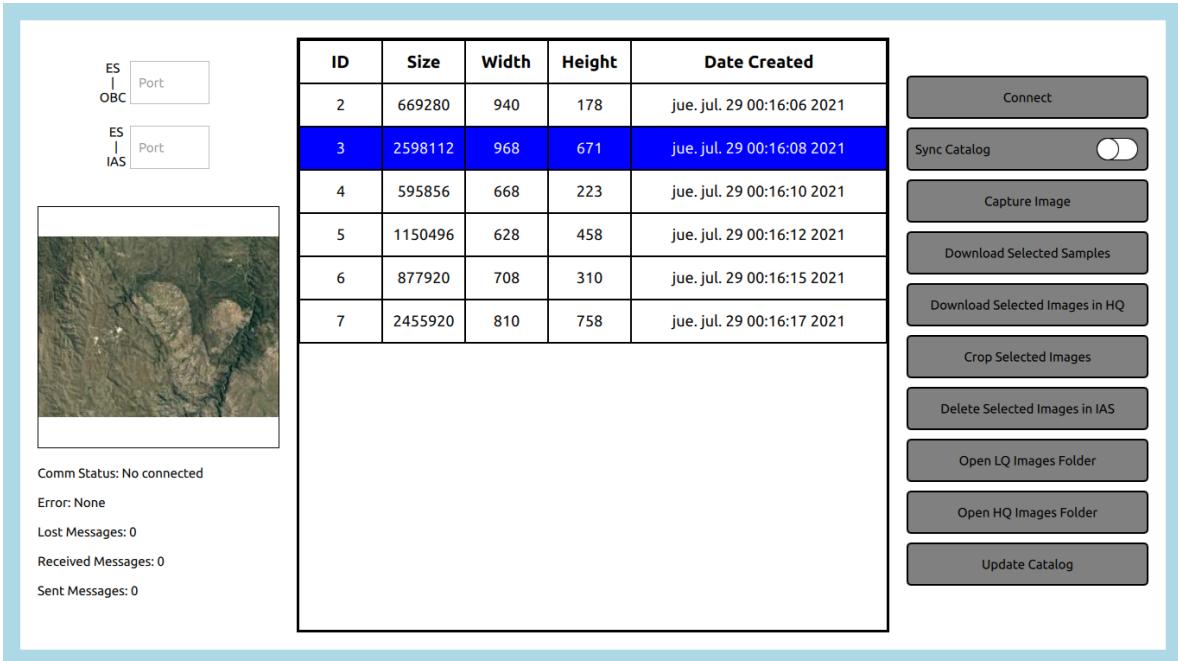


Figura 48: GUI V1.0

Capítulo 4

4. Resultados

En este capítulo, se abordan los resultados desde el punto de vista de los objetivos y casos de uso planteados al inicio del proyecto (Sección 1.3). Se detallan las pruebas realizadas en hardware y software, se comprueban los resultados obtenidos para caso de uso y se comparan con los objetivos del sistema.

Además, se muestra el porcentaje de cobertura de código alcanzado por los unit test y, los resultados obtenidos con cppcheck y memcheck para corroborar la calidad del código.

4.1. Consideraciones

Por razones de público conocimiento durante los años 2020 y 2021, se llegó a un acuerdo con el director de la tesis para poder reemplazar elementos de hardware a los que se dificultaba o ya no se tenía acceso. La cámara JAI pertenece al CIA y no estaba permitido retirarla fuera del predio, por lo que en su lugar, se decide reemplazarla por una webcam, manteniendo el funcionamiento general de forma similar. Tampoco fue posible tener acceso a los transceptores del EBV, por lo que en su lugar y, aprovechando la flexibilidad que permite el protocolo UART para conexión hardware, se reemplazaron por un cable que irá desde la ES a la OBC.

4.2. Conectarse al microsatélite

Para cumplir con este caso de uso, es necesario el previo desarrollo dos partes fundamentales. Por un lado el protocolo de comunicaciones y por otro, la interfaz de usuario o GUI.

4.2.1. Simulación del protocolo

En su inicio, para mayor simpleza y desarrollo ágil del proyecto, se realizó una simulación de las 3 computadoras (ES, IAS y OBC) para probar el intercambio de mensajes. Sólo se implementaron los mensajes de encendido y apagado de leds. Para la conexión de estas computadoras, se crearon enlaces virtuales (Anexo A.4) dentro del propio SO GNU/Linux (Ubuntu).

La UI existió desde el inicio, aunque no en su forma final. Su desarrollo fue iterativo, siempre añadiendo componentes que permitan hacer pruebas de funcionamiento básicas generando eventos, como botones para enviar comandos y cuadros de texto para ingresar los enlaces virtuales generados. Es decir, no hubo un diseño basado en las necesidades del usuario hasta la última parte del desarrollo, cuando se atacó puntualmente ésta problemática. La verificación del correcto funcionamiento se interpretaba imprimiendo en pantalla el estado de los leds de las computadoras, que son simplemente variables booleanas. En la ET, el estado del led de la OBC y el de la IAS se guardan en 2 variables booleanas. Éstas deben sincronizarse con el estado de los leds de cada computadora.

```

Application Output | ↻ ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ Filter + -
SerialProtocolImpl ✎
ET: Sending command (CommandCode = 128): ("7e0036800000007d")
OBC: Forwarding command (CommandCode = 128) to port 3: ("7e0036800000007d")
IAS: Receiving command (CommandCode = 128): ("7e0036800000007d")
IAS: Creating action (LedIASGetReqAction)
IAS: Executing action (LedIASGetReqAction)
IAS: Sending command (CommandCode = 129): ("7e003781010100007d")
IAS: Action finished (LedIASGetReqAction)
OBC: Forwarding command (CommandCode = 129) to port 2: ("7e003781010100007d")
ET: Receiving command (CommandCode = 129): ("7e003781010100007d")
ET: Executing action (LedIASGetResAction)
ET: IAS Led status is ON
ET: Action finished (LedIASGetResAction)

```

Figura 49: Envío y recepción de comando para consultar el estado del led de la IAS

```

Application Output | ↻ ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ Filter + -
SerialProtocolImpl ✎
ET: Button clicked (ToggleIASLedButton)
ET: Sending command (CommandCode = 130): ("7e003882010000007d")
OBC: Forwarding command (CommandCode = 130) to port 3: ("7e003882010000007d")
IAS: Receiving command (CommandCode = 130): ("7e003882010000007d")
IAS: Creating action (LedIASSetReqAction)
IAS: Executing action (LedIASSetReqAction)
IAS: Led status changed to OFF
IAS: Action finished (LedIASSetReqAction)

```

Figura 50: Envío de comando para cambiar el estado del led de la IAS

```

Application Output | ↻ ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ Filter + -
SerialProtocolImpl ✎
ET: Sending command (CommandCode = 128): ("7e003b800000007d")
OBC: Forwarding command (CommandCode = 128) to port 3: ("7e003b800000007d")
IAS: Receiving command (CommandCode = 128): ("7e003b800000007d")
IAS: Creating action (LedIASGetReqAction)
IAS: Executing action (LedIASGetReqAction)
IAS: Sending command (CommandCode = 129): ("7e003c81010000007d")
IAS: Action finished (LedIASGetReqAction)
OBC: Forwarding command (CommandCode = 129) to port 2: ("7e003c81010000007d")
ET: Receiving command (CommandCode = 129): ("7e003c81010000007d")
ET: Executing action (LedIASGetResAction)
ET: IAS Led status is OFF
ET: Action finished (LedIASGetResAction)

```

Figura 51: Verificación de cambio de estado del led realizando nueva consulta

4.2.2. Pruebas en hardware

Para trasladar el desarrollo a hardware, se implementó una versión simplificada del protocolo en C para permitir el forwarding y procesamiento básico de mensajes en la OBC.

Se implementaron las funciones para prender y apagar los leds, con las particularidades de cada placa y se asociaron a las acciones correspondientes del protocolo.

4.2.2.1 Secuencia de conexión

Inicialmente, los botones se encuentran bloqueados hasta abrir correctamente los puertos, a excepción de los botones que abren las carpetas para visualizar las imágenes descargadas.

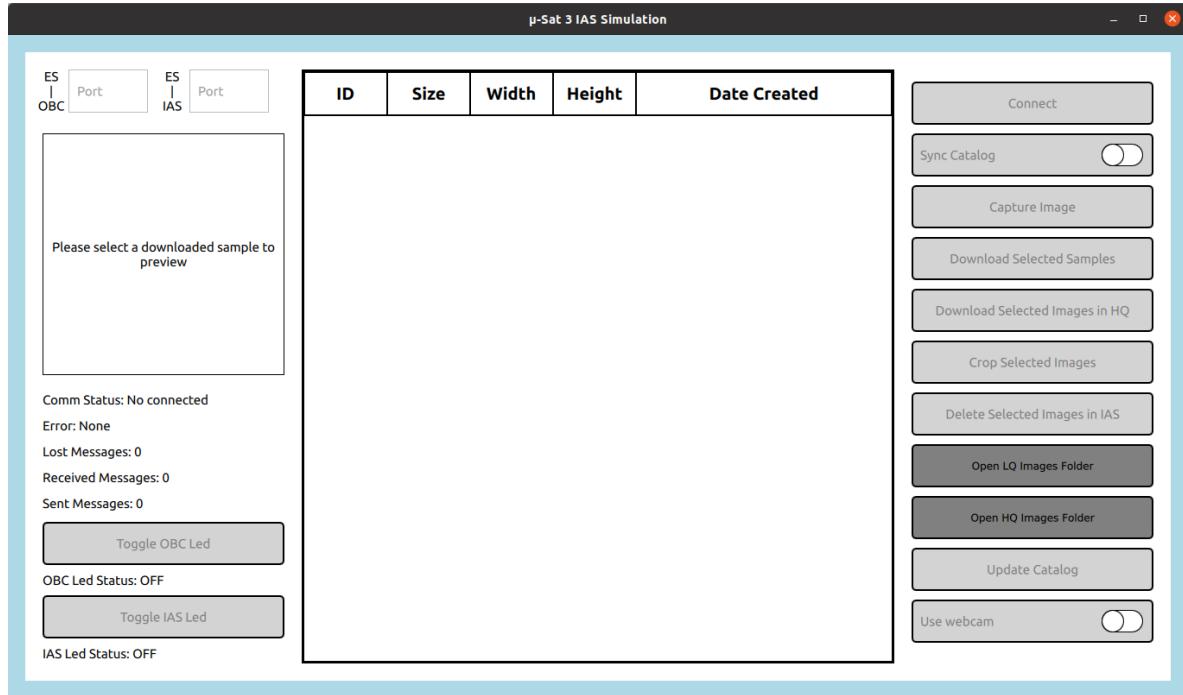


Figura 52: Botones de la GUI bloqueados sin puertos configurados

Al abrir el puerto correspondiente al EBV, se habilita el botón Connect y los botones que permiten alternar el estado de los leds de la OBC y la IAS.

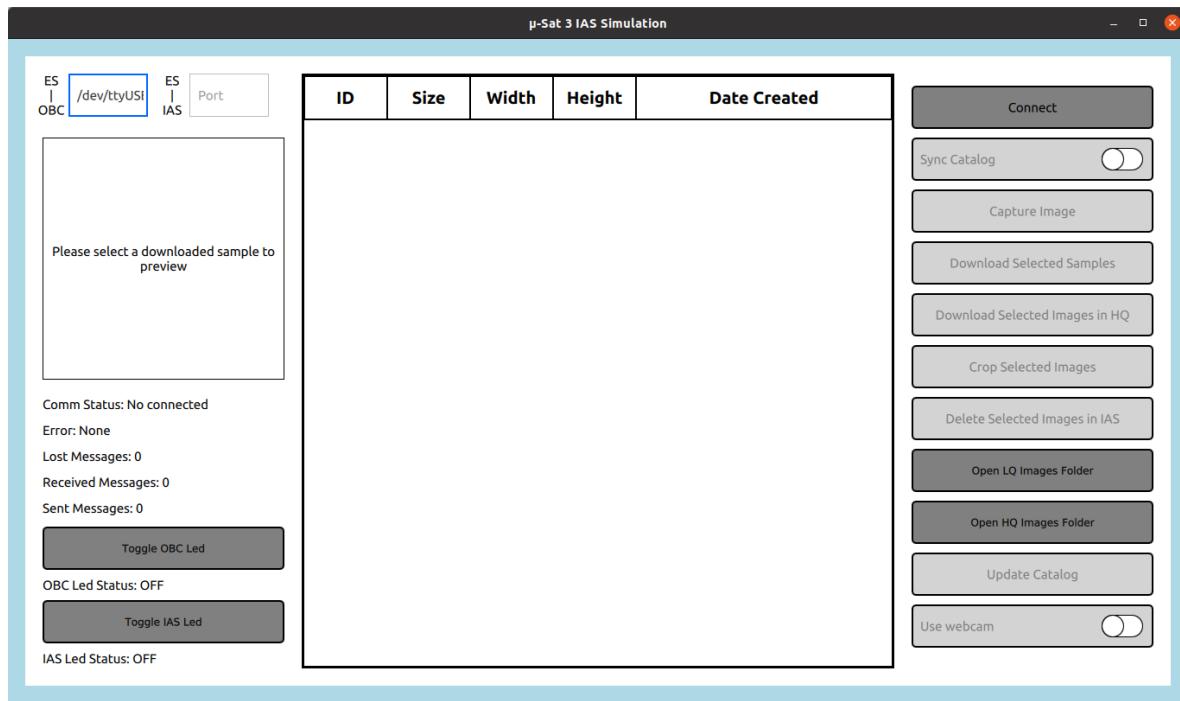


Figura 53: Se habilita el botón Connect al configurar el puerto del EBV

El comando KEEPALIVE REQ comienza a enviarse al presionar el botón Connect. Se considera que existe comunicación entre la ES y la IAS al recibir la primer respuesta. Luego se habilitan el resto de los botones.

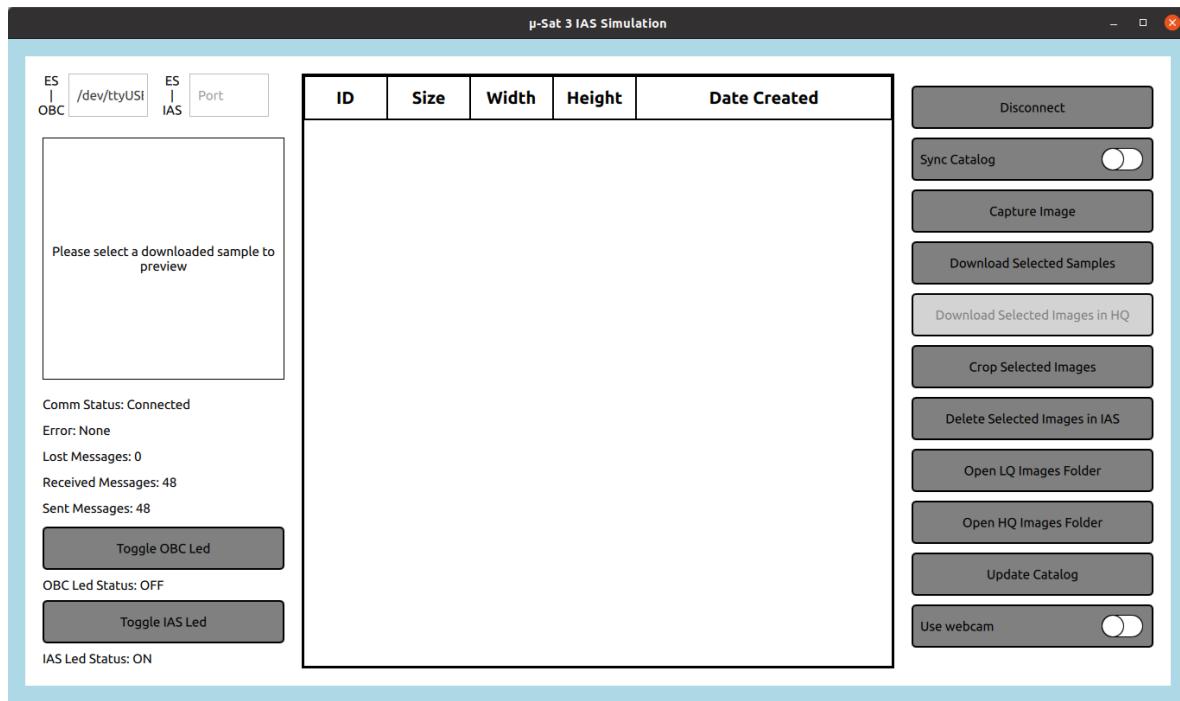


Figura 54: Se habilitan el resto de botones que no utilizan el EAV

Notar que al configurar sólo el puerto el EBV, los botones relacionados a funciones del EAV permanecen bloqueados.

Luego de abrir correctamente el otro puerto, se habilitan todos los botones.

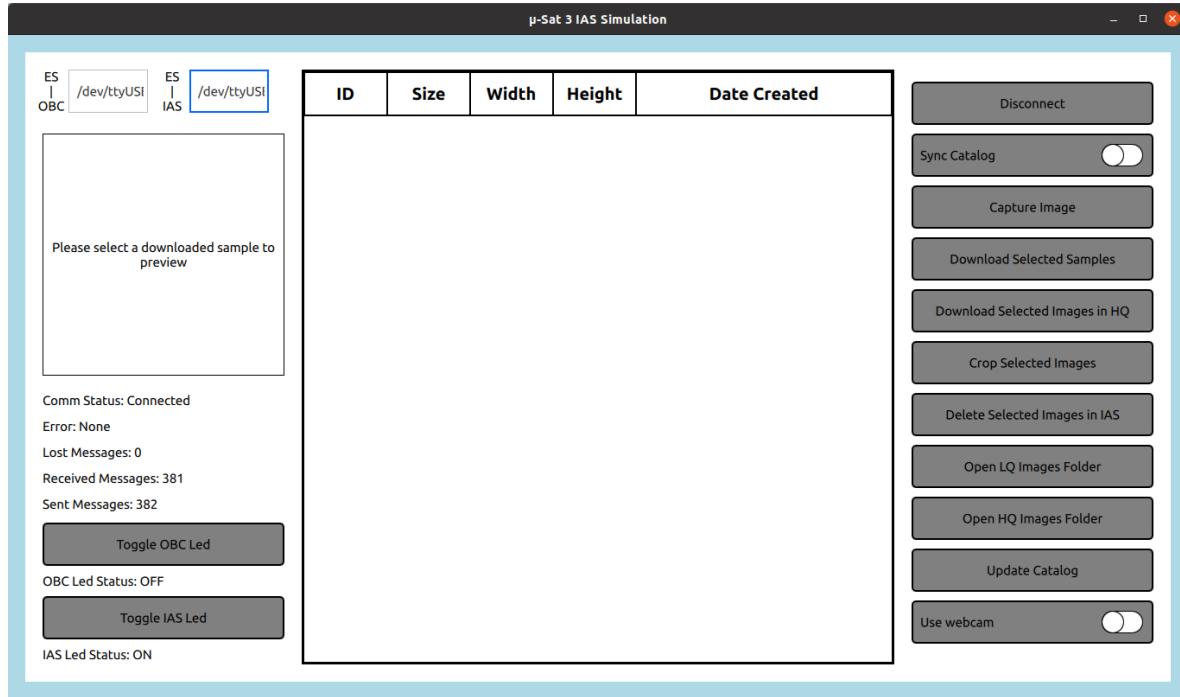


Figura 55: GUI con todos los botones habilitados

Durante la operación se muestran mensajes de estado y error si los hubiese. También se muestran mensajes informativos para contar la cantidad de paquetes enviados, perdidos y transmitidos.

Una vez realizada la conexión, se probaron los comandos de encendido de leds obteniendo los mismos resultados que en la simulación, esta vez corroborados visualmente de forma física.

De esta forma, quedan cubiertos los objetivos **Diseñar e implementar un protocolo de comunicación UART para transmitir información entre la ES y las computadoras del microsatélite** y **Desarrollar una GUI en la ES que permita conectarse y ejecutar acciones en la IAS**.

4.3. Capturar imágenes

Por las consideraciones mencionadas en la Sección 4.1, se decidió mantener 2 formas de capturar una fotografía: Con la cámara original JAI o con una cámara web. Se implementaron interfaces que permiten seleccionar qué función utilizar.

Para capturar una fotografía, primero se debe seleccionar desde la GUI la cámara mediante la cuál se pretende capturarla. Luego, se debe presionar sobre el botón "Capture Image".



Figura 56: Fotografía capturada con la webcam en modo vertical

Al capturar una fotografía, se almacena junto a una copia en baja calidad en un dispositivo de almacenamiento de datos y se agrega en el catálogo de la IAS asignándole un ID incremental.

De esta forma, quedan cubiertos los objetivos **Capturar imágenes mediante una cámara conectada vía Gb Ethernet a la IAS, Guardar las imágenes capturadas en un dispositivo de almacenamiento de datos y Almacenar una vista previa en baja resolución (thumbnail) al capturar una imagen.**

4.4. Listar y previsualizar imágenes (descarga por el EBV)

Como se explicó en la sección anterior, sacar una nueva fotografía modifica el catálogo de la IAS y por lo tanto cambia su CRC que se envía a la ES en las respuestas del comando KEEPALIVE REQ.

La ES compara el CRC de su catálogo con el contenido de la respuesta del comando KEEPALIVE REQ para saber si está desactualizado.

En este caso, si la opción "Sync Catalog" está seleccionada, el catálogo será actualizado automáticamente enviando un comando CATALOG REQ. De lo contrario, un mensaje informativo aparece para indicar que el catálogo de la ES está desactualizado y el usuario deberá actualizarlo manualmente si así lo desea.

Para descargar un imagen en baja calidad, se debe seleccionar la misma en el catálogo y presionar sobre el botón "Download Selected Samples" de la GUI. Ahora, el usuario puede realizar acciones con las imágenes descargadas, como por ejemplo, la previsualización y zoom de las mismas.

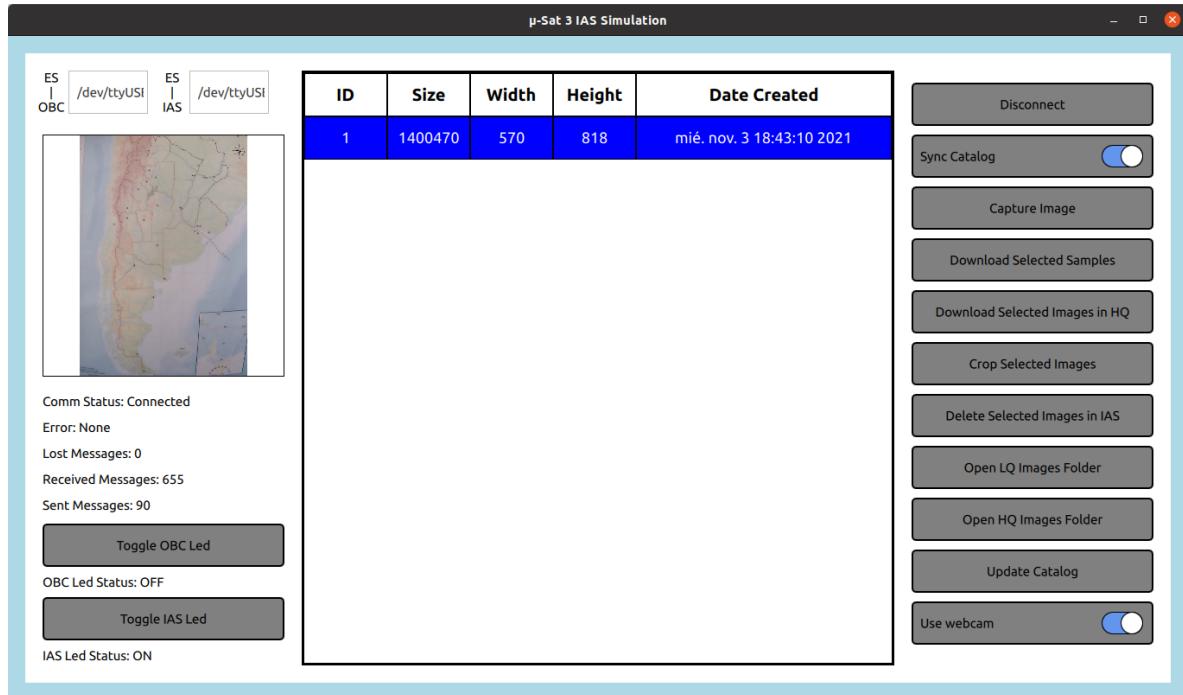


Figura 57: Imagen capturada, listada y descargada en baja calidad para previsualización

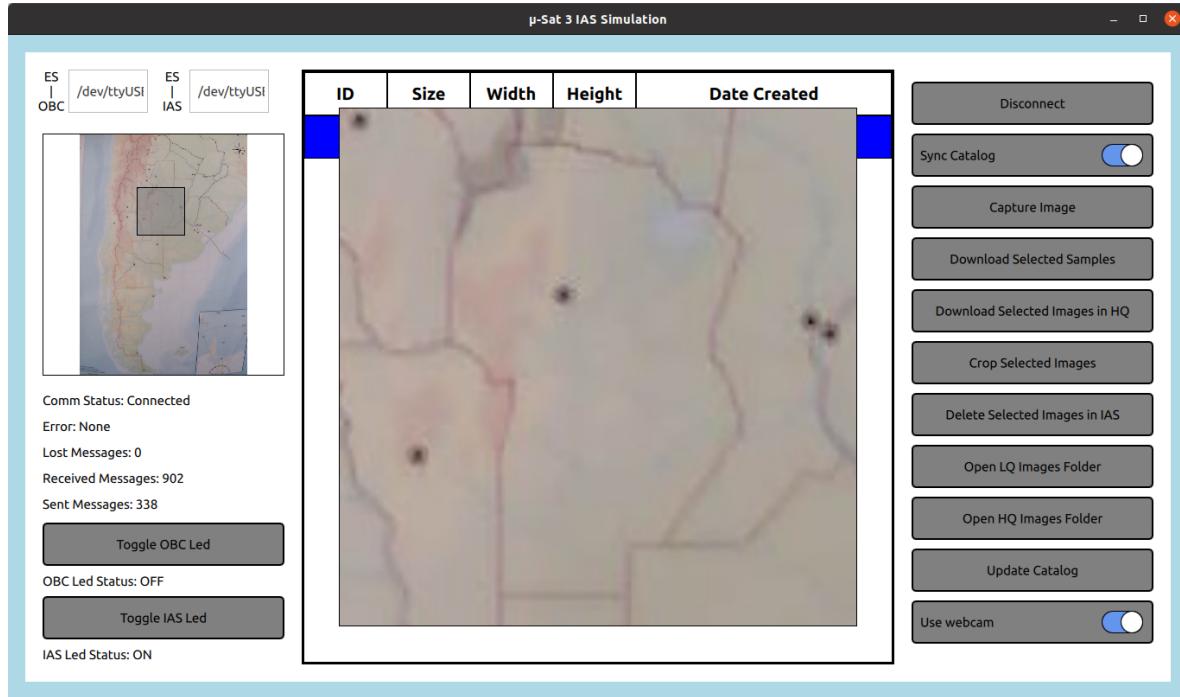


Figura 58: Zoom de imagen listada en el catálogo

Si la ES posee un catálogo descargado previamente al iniciar la aplicación, se podrán visualizar las imágenes de muestra descargadas, aunque no se podrán realizar operaciones con ellas hasta conectarse a la IAS.

De esta forma, queda cubierto el objetivo **Enviar las vistas previas de las imágenes por el EBV**.

4.5. Recortar imágenes

4.5.1. Recorte en la ES

Una vez descargada la imagen de vista previa desde el satélite, es posible seleccionar desde la GUI una región de interés y generar una nueva imagen en la IAS a partir de un recorte de la imagen original.

Para iniciar el programa de recorte, se debe seleccionar una imagen desde el catálogo y luego presionar sobre el botón "Crop Selected Images". Se abre la imagen y con el mouse, al presionar y mantener el botón primario, se realiza la selección deseada. Se debe arrastrar el puntero desde la parte superior izquierda hacia la parte inferior derecha para formar un rectángulo. Para finalizar, presionar la tecla ENTER.

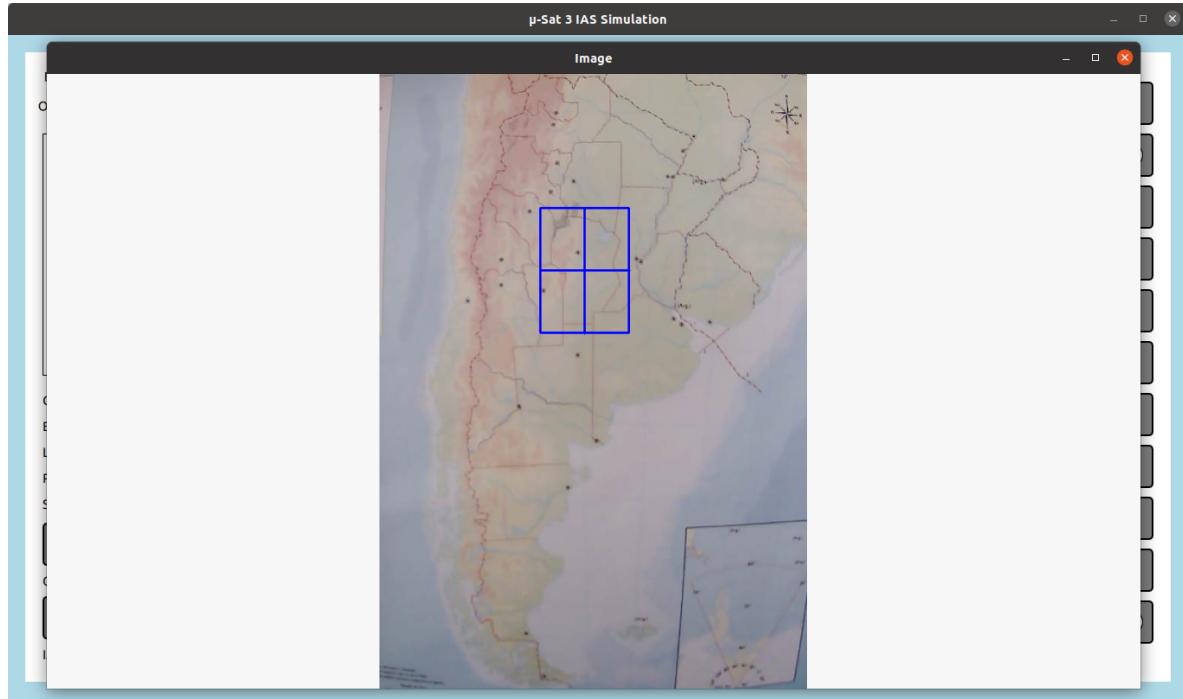


Figura 59: Selección de ROI de una imagen del catálogo

Cabe aclarar que es posible seleccionar más de una región de interés por imagen. Para ello, se debe repetir el procedimiento anterior para seleccionar una nueva ROI luego de presionar la tecla ENTER.

Es posible acercar la imagen (zoom) con el scroll del mouse para una mayor precisión. Para ello, se debe posicionar el puntero del mouse en donde se desea realizar el zoom y desplazar el scroll del mouse hacia arriba. Se podrá hacer zoom las veces que se quiera. Para regresar el zoom, desplazar el scroll una vez hacia abajo. Se puede seleccionar todas las ROI que se deseé en los niveles de zoom que se desee.

Para salir del programa, presionar la tecla ESCAPE.

4.5.2. Recorte en la IAS

Cuando el microsatélite recibe el comando de recorte (CROP REQ), se lee el ID de la imagen original y las coordenadas ROI que se encuentran dentro del payload del comando. Se generarán tantas imágenes de salida como ROIs se hayan encontrado.

Estas nuevas imágenes se agregarán al catálogo con un ID único al igual que las imágenes que se capturen.

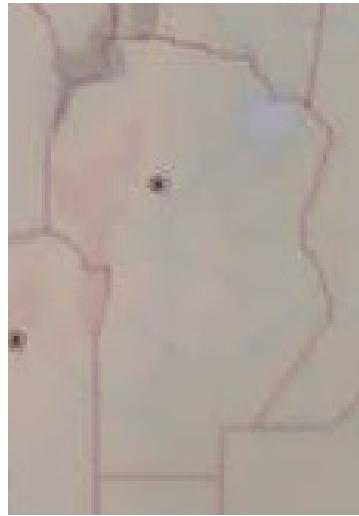


Figura 60: Recorte de la imagen original

Los recortes de imágenes podrán ser descargados luego por el EAV o el EBV del microsatélite.

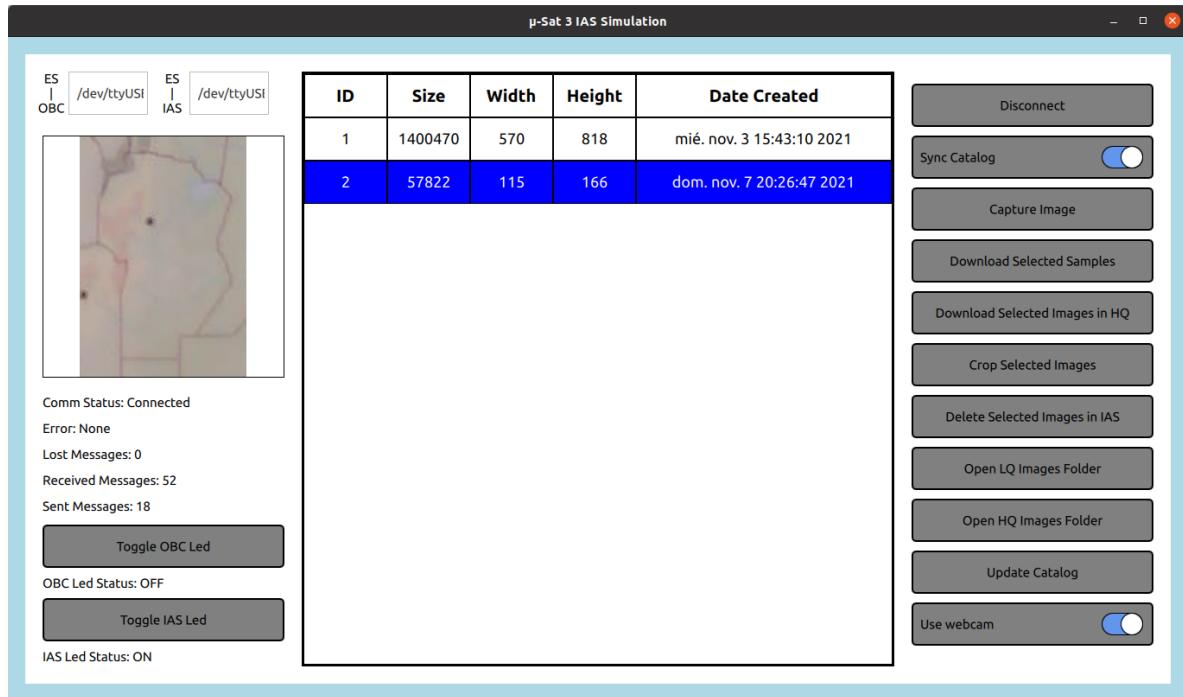


Figura 61: Recorte de imagen descargado por el EBV en la ES

De esta forma, queda cubierto el objetivo **Permitir recortar una imagen para guardar una o más regiones de interés, o Regions of Interest (ROI)**.

4.6. Borrar imágenes

Para borrar una o más imágenes, se deben seleccionar desde el catálogo y luego presionar el botón "Delete Selected Images in IAS". Esto enviará un comando DELETE REQ al microsatélite en cuyo payload contendrá los IDs de las imágenes a borrar.

Se borrarán las imágenes en las IAS tanto del dispositivo de almacenamiento como del catálogo que luego deberá ser actualizado desde la ES para visualizar los cambios.

De esta forma, queda cubierto el objetivo **Permitir borrar las imágenes capturadas**.

4.7. Descargar imágenes en alta calidad

Las imágenes en alta calidad se descargan a través del EAV del microsatélite. Como se trata de un enlace unidireccional sin ACK ni CRC, las imágenes se transmiten en formato .bmp para permitir siempre su reconstrucción, aún en caso de pérdida de paquetes.

Al igual que el proceso de borrado, para descargar una o más imágenes en alta calidad, las mismas deben seleccionarse desde el catálogo y luego presionar el botón "Download Selected Images in HQ". Se enviará un comando CAPTURE REQ al microsatélite con los IDs de las imágenes a descargar. En la IAS, se ejecutará la acción que desglosa la imagen y comienza a transmitir los paquetes por el transmisor RF del EAV. En la ES, el receptor RF recibe los paquetes y se reconstruye la imagen.

El firmware utilizado para el transmisor y receptor RF del EAV solo posee unas pequeñas modificaciones con respecto al desarrollado en la Práctica Profesional Supervisada [4] con el objetivo de lograr mayor flexibilidad e integración con este proyecto. El principal cambio fue la inclusión de buffers circulares y la utilización de interrupciones para evitar cuellos de botella producto de la diferencia de velocidades entre la transmisiones UART/SPI y las transmisiones RF. Además se logró desacoplar la lógica del protocolo que antes estaba en el firmware de los transceptores.

De esta forma, queda cubierto el objetivo **Enviar las imágenes originales o recortadas a través del EAV**.

4.8. Análisis y cobertura de código

Se ejecutaron los programas de análisis de código cppcheck (estático) y memcheck (dinámico), ambos se pueden integrar y ejecutar desde Qt Creator.

Se corrigieron los errores y los warnings detectados mediante cppcheck, manteniendo los mensajes de información que no representan una problemática para el sistema.

```

Cppcheck < > 🗑
▼ ⓘ /home/tesis/Repos/ESIS_V2/src/Screen/Components/ButtonViewModel.cpp
  ⚠ 60: Member variable 'ButtonViewModel::m_type' is not initialized in the constructor.
  ⚠ 60: Member variable 'ButtonViewModel::m_enabled' is not initialized in the constructor.
  ⚠ 60: Member variable 'ButtonViewModel::m_checked' is not initialized in the constructor.
  ⓘ 66: Function parameter 'text' should be passed by const reference.
  ⓘ 46: The function 'getChecked' is never used.
  ⓘ 32: The function 'getEnabled' is never used.
  ⓘ 3: The function 'getText' is never used.
  ⓘ 13: The function 'getType' is never used.
▶ ⓘ /home/tesis/Repos/ESIS_V2/src/Screen/Components/InfoTextViewModel.cpp
▶ ⓘ /home/tesis/Repos/ESIS_V2/src/Screen/Components/TextFieldViewModel.cpp
▶ ⓘ /home/tesis/Repos/ESIS_V2/src/Screen/MainScreenViewModel.cpp
▼ ⓘ /home/tesis/Repos/ESIS_V2/src/Utils/ImageFileInfo.cpp
  ⓘ 63: Resource leak: fp
  ⓘ 90: Parameter 'bmpFileHeader' can be declared with const
  ⓘ 85: The function 'timestampLastMod' is never used.
▶ ⓘ /home/tesis/Repos/ESIS_V2/src/ViewModelManager.cpp

```

Figura 62: Resultado de la primera ejecución de cppcheck

La ejecución de memcheck permitió obtener un listado de problemas relacionados principalmente con el uso de la memoria que son difíciles de detectar de otra manera. La mayoría de las fallas en el software fueron resueltas usando el debugger de Qt Creator, pero la información proporcionada por memcheck puede ser de interés para fallas difíciles de diagnosticar. Como en los casos de uso habituales no se detectan fallos, revisar exhaustivamente los mensajes de memcheck se deja como trabajo futuro.

Issue	Location
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,657 of 24,334	SerialPortManager.cpp:23:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,658 of 24,334	CommManager.cpp:24:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,659 of 24,334	CommManager.cpp:28:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,660 of 24,334	SerialPortManager.cpp:23:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,661 of 24,334	CommManager.cpp:24:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,662 of 24,334	CommManager.cpp:28:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,663 of 24,334	SerialPortManager.cpp:23:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,664 of 24,334	CommManager.cpp:24:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,665 of 24,334	CommManager.cpp:28:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,666 of 24,334	SerialPortManager.cpp:23:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,667 of 24,334	CommManager.cpp:24:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,668 of 24,334	CommManager.cpp:28:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,669 of 24,334	SerialPortManager.cpp:23:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,670 of 24,334	CommManager.cpp:24:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,671 of 24,334	CommManager.cpp:28:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,672 of 24,334	SerialPortManager.cpp:23:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,673 of 24,334	CommManager.cpp:24:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,674 of 24,334	CommManager.cpp:28:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,675 of 24,334	ViewModelManager.cpp:23:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,676 of 24,334	ViewModelManager.cpp:27:0
↳ 160 bytes in 1 blocks are possibly lost in loss record 21,677 of 24,334	ViewModelManager.cpp:106:0
1,274 (360 direct, 914 indirect) bytes in 9 blocks are definitely lost in loss record 23,818 of 24,334	ImageDataBase.cpp:71:0

Application Output

```

19:42:02: valgrind --child-silent-after-fork=yes --xml-socket=127.0.0.1:45117 --log-socket=127.0.0.1:39495 --xml=yes --smc-check=stack --tool=memcheck --gen-suppressions=all --track-origins=yes --leak-check=summary --num-callers=25 /home/tesis/Repos/build-SerialProtocol-Desktop_Qt_5_15_2_GCC_64bit-Debug/src/SerialProtocolImpl /dev/pts/3

```

Figura 63: Resultado de la primera ejecución de memcheck

Por último, se obtuvo la cobertura de código de los tests unitarios mediante Gcovr. Se logró cubrir más del 78 % del código, valor por encima del 75 % que era el objetivo inicial. Más allá del porcentaje alcanzado, el principal beneficio de realizar este análisis fue encontrar partes del código importantes que no eran alcanzadas por los tests.

Unit Test Coverage Report

File	Lines	Exec	Total	Coverage
Action/ActionFactory.cpp	100.0%	9 / 9	100.0%	3 / 3
Action/ActionFactoryES.cpp	92.9%	52 / 56	69.4%	77 / 111
Action/ActionFactoryIAS.cpp	96.7%	59 / 61	85.9%	61 / 71
Action/ActionFactoryOBC.cpp	100.0%	9 / 9	100.0%	4 / 4
Action/ActionManager.cpp	75.0%	15 / 20	73.3%	11 / 15
Action/ActionsQueue.cpp	93.8%	15 / 16	87.5%	7 / 8
Action/CaptureReqAction.cpp	100.0%	9 / 9	100.0%	6 / 6
Action/CatalogReqAction.cpp	78.6%	11 / 14	41.7%	10 / 24
Action/CatalogResAction.cpp	100.0%	8 / 8	88.9%	8 / 9
Action/CropReqAction.cpp	100.0%	10 / 10	100.0%	10 / 10
Action/DeleteReqAction.cpp	100.0%	14 / 14	100.0%	10 / 10
Action/ImageHQReqAction.cpp	86.2%	25 / 29	62.5%	25 / 40
Action/ImageHQResAction.cpp	87.5%	21 / 24	75.0%	30 / 40
Action/ImageLQReqAction.cpp	31.0%	9 / 29	9.3%	5 / 54
Action/ImageLQResAction.cpp	87.0%	20 / 23	73.0%	27 / 37
Action/KeepaliveReqAction.cpp	100.0%	8 / 8	88.9%	8 / 9
Action/KeepaliveResAction.cpp	80.0%	16 / 20	57.1%	24 / 42
Action/LedIASGetReqAction.cpp	100.0%	5 / 5	100.0%	2 / 2
Action/LedIASGetResAction.cpp	100.0%	5 / 5	100.0%	2 / 2
Action/LedIASSetReqAction.cpp	100.0%	5 / 5	100.0%	2 / 2
Action/LedOBCCGetReqAction.cpp	100.0%	5 / 5	100.0%	2 / 2
Action/LedOBCCSetReqAction.cpp	100.0%	5 / 5	100.0%	2 / 2
CommLinkMonitor.cpp	96.1%	49 / 51	65.6%	21 / 32
Command/CaptureCommandReq.cpp	100.0%	4 / 4	-%	0 / 0
Command/CatalogCommandReq.cpp	100.0%	4 / 4	-%	0 / 0
Command/CatalogCommandRes.cpp	100.0%	9 / 9	100.0%	5 / 5
Command/Command.cpp	100.0%	2 / 2	-%	0 / 0
Command/CropCommandReq.cpp	100.0%	14 / 14	100.0%	9 / 9
Command/DeleteCommandReq.cpp	100.0%	7 / 7	100.0%	5 / 5
Command/ImageQCCommandReq.cpp	100.0%	6 / 6	100.0%	1 / 1
Command/ImageQCCommandRes.cpp	100.0%	9 / 9	100.0%	4 / 4
Command/KeepaliveCommandReq.cpp	100.0%	4 / 4	-%	0 / 0
Command/KeepaliveCommandRes.cpp	100.0%	5 / 5	100.0%	1 / 1
Command/LedIASGetCommandReq.cpp	100.0%	4 / 4	-%	0 / 0
Command/LedIASGetCommandRes.cpp	100.0%	4 / 4	-%	0 / 0
Command/LedIASSetCommandReq.cpp	100.0%	4 / 4	-%	0 / 0
Command/LedOBCCGetCommandReq.cpp	100.0%	4 / 4	-%	0 / 0
Command/LedOBCCGetCommandRes.cpp	100.0%	4 / 4	-%	0 / 0
Command/LedOBCCSetCommandReq.cpp	100.0%	4 / 4	-%	0 / 0
Command/TransmissionQueue.cpp	87.5%	14 / 16	50.0%	2 / 4
Computer/Computer.cpp	83.3%	40 / 48	23.2%	13 / 56
Computer/EarthStation.cpp	100.0%	34 / 34	100.0%	42 / 42
Computer/ImageAcquisitionSystem.cpp	100.0%	23 / 23	100.0%	39 / 39
Computer/OnBoardComputer.cpp	100.0%	26 / 26	100.0%	33 / 33
Error/ErrorManager.cpp	93.3%	14 / 15	56.2%	9 / 16
Image/CameraManager.cpp	100.0%	15 / 15	100.0%	23 / 23
Image/ImageDataBase.cpp	76.5%	179 / 234	54.2%	305 / 563
Image/ImageEditor.cpp	42.3%	47 / 111	45.3%	92 / 203
Image/ImageManager.cpp	37.9%	44 / 116	11.5%	34 / 295
Image/ImageProperties.cpp	100.0%	24 / 24	100.0%	23 / 23
Protocol/CommManager.cpp	44.0%	37 / 84	12.0%	14 / 117
Protocol/PacketBuilder.cpp	100.0%	56 / 56	96.2%	50 / 52
Protocol/PacketReader.cpp	96.1%	49 / 51	78.6%	22 / 28
Protocol/PacketValidator.cpp	56.0%	14 / 25	25.0%	8 / 32
Protocol/SerialPortManager.cpp	36.4%	36 / 99	12.5%	18 / 144
Screen/AbstractScreenViewModel.cpp	100.0%	21 / 21	66.7%	6 / 9
Screen/Components/ButtonViewModel.cpp	100.0%	29 / 29	100.0%	6 / 6
Screen/Components/ImagePropertiesViewModel.cpp	100.0%	11 / 11	100.0%	13 / 13
Screen/Components/InfoTextViewModel.cpp	100.0%	15 / 15	50.0%	1 / 2
Screen/Components/TextFieldViewModel.cpp	100.0%	17 / 17	90.0%	9 / 10
Screen/MainScreenViewModel.cpp	100.0%	86 / 86	100.0%	95 / 95
Screen/ScreenProperties.cpp	100.0%	11 / 11	-%	0 / 0
Utils/Files.cpp	100.0%	8 / 8	100.0%	8 / 8
Utils/ImageFileInfo.cpp	91.0%	61 / 67	76.7%	56 / 73
Utils/Operators.cpp	100.0%	9 / 9	-%	0 / 0
ViewModel.cpp	100.0%	3 / 3	-%	0 / 0

Generated by: GCOVR (Version 5.0)

Figura 64: Resultado final de cobertura de código alcanzado

Capítulo 5

5. Conclusiones

El objetivo de esta tesis fue tomar el control de la computadora de imágenes (IAS) del microsatélite μ -SAT 3 desde una estación en la Tierra (ES) para capturar, almacenar y transmitir imágenes satelitales. Para ello se diseñó un protocolo de comunicaciones UART y se realizaron varias aplicaciones de software para cada una de las computadoras, incluyendo una interfaz gráfica de usuario para la ES, el control de una cámara IP y el firmware de transceptores RF. En base al análisis de los resultados, se obtuvieron las conclusiones que se exponen a continuación.

Para el usuario en la ES, el objetivo antes mencionado se cumple gracias a los comandos que puede enviar a través de la GUI, que esconden la complejidad de comunicar varias computadoras y de las acciones que deben realizarse para concretar cada uno de esos comandos.

La captura de imágenes se logró con éxito gracias a la conexión y control de la cámara IP a través de la API de Baumer y su fácil integración con el sistema operativo basado en GNU/Linux de la IAS. El desafío inicial fue encontrar un software que permitiera el control del modelo de la cámara utilizada, que sea de fácil utilización e instalación y que fuese orientado a software libre.

La información de las imágenes capturadas con Baumer queda almacenada en RAM. Se investigó alguna solución para poder almacenarla en memoria no volátil. De las distintas alternativas, se decidió utilizar como complemento OpenCV. Esta librería permitió lograr también la conversión a los formatos .jpg y .bmp y realizar una compresión con pérdida obteniendo las vistas en baja calidad. Ambas librerías son open source y están escritas en C++, por lo que pudieron ser integradas al proyecto fácilmente.

OpenCV se utilizó también para la implementación del recorte de imágenes. Con la función ROI de esta librería se obtienen las coordenadas de la región de interés y se envían al microsatélite para generar una subimagen en base a la imagen original.

La transmisión de las imágenes se logró realizando una pequeña refactorización sobre el código previamente desarrollado. Se llegó a la conclusión de que podría resolverse de una manera más simple, eliminando código duplicado y realizando un tratamiento de interrupciones mejorado. Cabe aclarar que el desarrollo se enfocó en cómo interactúan los componentes del sistema entre sí, es decir, las pruebas estuvieron orientadas a mostrar la funcionalidad, sin enfocarse demasiado en la velocidad de transmisión. Lograr una velocidad de descarga determinada no fue un requerimiento inicial de este trabajo. Debido a las limitaciones de los transceptores en cuanto a potencia de salida y ganancia de las antenas de 0 dB, la distancia entre las placas resultó ser un factor determinante en la aparición de errores.

Aunque no era requisito fundamental el desarrollo de una GUI, se decidió implementarla. Los motivos principales fueron el deseo de visualizar las imágenes del catálogo y que el usuario pueda desarrollar tareas fácilmente, sin tener que escribir cada comando. Para llevarla a cabo, se utilizó la librería Qt y el lenguaje QML. Con estas herramientas y utilizando el patrón de diseño MVVM, se pudo desarrollar en poco tiempo una interfaz simple e intuitiva conectada a la aplicación escrita en C++.

La interconexión de las distintas computadoras se logró con el desarrollo del protocolo de comu-

nición escrito en C/C++ basado en una versión simplificada de HDLC, un protocolo de propósito general punto a punto, con información de control mínima. La versión escrita en C++ utiliza elementos de Qt y se encuentra en la ES y la IAS. La versión escrita en C se encuentra en la OBC. Ambas versiones soportan el mecanismo de forwarding y permiten procesar comandos y ejecutar acciones.

Durante el desarrollo de la aplicación en la ES se utilizó la metodología TDD. Esto implicó escribir los test antes de programar. De esta forma se trata de obtener el código mínimo para cumplir los requerimientos y pasar los tests al agregar nueva funcionalidad. Adicionalmente, se calculó la cobertura de código para usar este resultado como referencia de calidad.

Durante la etapa final de pruebas, no se logró tener acceso a la cámara IP ni a los transceptores del EBV. Esto imposibilitó la integración final de todo el proyecto con estos componentes por lo que fueron reemplazados por una cámara web y un cable USB respectivamente. Para el caso del cable, la integración fue directa al tratarse de interfaces UART, pero para la cámara fue necesario desarrollar el software para su control. El desarrollo con Baumer ya se encontraba finalizado por lo que se decidió implementar una interfaz que permitiera la elección de la cámara a utilizar, ya sea la cámara IP o la web.

Pese a estas limitaciones, se logró con éxito la implementación de un prototipo funcional que mostró las interacciones básicas con la computadora de imágenes de un satélite de telemetría y una estación terrena. El diseño e implementación de software de alto y bajo nivel para comunicar y controlar los distintos componentes de hardware en conjunto con la aplicación de patrones de diseño, metodologías de desarrollo y uso de librerías, permitió conectar y consolidar así como extender los conceptos adquiridos durante la carrera de Ingeniería en Computación.

5.1. Trabajos futuros

Se detalla a continuación las posibles mejoras que fueron surgiendo durante el desarrollo de este proyecto.

Permitir que la IAS entre en modo sleep en intervalos ociosos para reducir el consumo, ya que es indispensable para la vida útil del microsatélite. Para ello, se puede implementar un mecanismo en donde la OBC despierte la IAS cada vez que recibe una instrucción de forwarding.

Agregar la posibilidad de utilizar exposición automática en la cámara buscando alguna alternativa distinta a Baumer con mayor compatibilidad con las cámaras JAI. Además, añadir la capacidad de capturar más de una fotografía en un momento determinado.

Implementar la automatización de tareas en la IAS para que, mientras el satélite esté fuera de la zona de cobertura, pueda seguir realizando tareas programadas teniendo en cuenta el escaso tiempo de operación (tiempo de visibilidad en donde el microsatélite pasa por sobre la zona de cobertura en donde se puede comunicar con la ES) de aproximadamente 14 minutos.

Encriptar los datos que se transmiten vía radiofrecuencia para resguardar todo tipo de información confidencial frente a ataques de robo de información.

Implementar la variación automática de velocidad de transmisión en el EAV dependiendo del nivel de RSSI presente en el enlace y otras mejoras con el objetivo de lograr velocidades de transmisión más altas. El indicador de fuerza de la señal recibida (RSSI - Received Signal Strength Indicator) es una escala de referencia (en relación a 1 mW) para medir el nivel de potencia de las señales recibidas por un dispositivo en las redes inalámbricas.

Reemplazar el catálogo actual en formato CSV por una base de datos, permitiendo mecanismos de resiliencia, la posibilidad de agregar mas parámetros al catálogo y la implementación de filtros.

Añadir funcionalidades como la descarga de imágenes mediante un archivo ZIP.

Regular la velocidad con la que la IAS envía el catálogo a la OBC o pedir desde la ES el catálogo por secciones, por ejemplo, las siguientes 10 entradas a partir de el último ID de imagen recibido. Actualmente, al solicitar el catálogo, la IAS envía un mensaje por cada entrada en él. Si el catálogo posee muchas entradas, esto podría saturar el canal de comunicaciones.

Implementar mecanismos de recuperación para volver al estado anterior en caso de falla en el medio de la ejecución de una tarea para evitar inconsistencias. Existe ciertas operaciones, como borrar una imagen, que requieren finalizar una serie de pasos para considerarse exitosas.

Revisar los problemas informados mediante la ejecución de memcheck.

Referencias

- [1] “La importancia de los satélites.” <https://es.euronews.com/2016/08/11/la-importancia-de-los-satelites-en-la-observacion-de-la-tierra>. Accedido: 2019-04-26.
- [2] “Argentina, uno de los siete países que desarrollan satélites.” <https://www.telam.com.ar/notas/201310/35474-argentina-ingreso-al-club-de-los-siete-paises-que-desarrollan-satelites.html>. Accedido: 2019-04-28.
- [3] “Satélite integralmente cordobés.” <https://uncuencia.unc.edu.ar/ingenieria/desarrollan-un-satelite-de-observacion-integralmente-cordobes-y-de-bajo-costo/>. Accedido: 2019-04-28.
- [4] M. Krenz and J. Moledo, *Desarrollo de un prototipo para Transmisión de Imágenes en Plataformas Satelitales*. 2017.
- [5] I. Sommerville, *Ingeniería de software*. Pearson Educación, 2011.
- [6] D. A. Maldonado, *Diseño e implementación de un transceptor de telemetría en banda UHF para el microsatélite μ-SAT3*. 2016.
- [7] D. A. Maldonado, *Diseño e implementación de un transceptor en banda S para el microsatélite μ-SAT3*. 2017.
- [8] C. R. Pandian, *Applied Software Risk Management, A Guide for Software Project Managers*. Auerbach Publications, 2006.
- [9] “Hercules tms570.” <http://www.ti.com/tool/TMDS570LS31HDK>. Accedido: 2019-05-28.
- [10] “Hercules tms570ls31x/21x development kit.” <https://www.ti.com/lit/ds/spns162c-spns162c.pdf?ts=1588550106033>. Accedido: 2019-05-28.
- [11] “Freertos.” <https://www.freertos.org/>. Accedido: 2019-05-29.
- [12] “Transceptor cc2544.” <http://www.ti.com/lit/ug/swru283b/swru283b.pdf>. Accedido: 2019-05-29.
- [13] “Transceptor cc1010.” <https://www.ti.com/lit/ds/symlink/cc1010.pdf>. Accedido: 2019-05-27.
- [14] “Baumer gapi sdk for linux arm.” <https://www.baumer.com/us/en/product-overview/image-processing-identification/software/baumer-gapi-sdk/linux-arm/c/14178>. Accedido: 2019-03-18.
- [15] “Raspbian.” <https://www.raspbian.org/>. Accedido: 2019-05-18.
- [16] “B sereis bm-500ge jai.” <https://www.jai.com/products/bm-500-ge>. Accedido: 2019-03-18.
- [17] P. Chu, *FPGA Prototyping by Verilog Examples*. 2008.
- [18] “Spi.” <https://www.arduino.cc/en/reference/SPI>. Accedido: 2019-07-09.
- [19] “Ti - radio frequence.” <http://www.ti.com/lit/ug/swru283b/swru283b.pdf>. Accedido: 2019-07-10.
- [20] “Code composer studio.” <http://www.ti.com/tool/CCSTUDIO>.

- [21] “Halcogen.” <http://www.ti.com/tool/HALCOGEN>.
- [22] “Iar embedded workbench.” <https://www.iar.com/iar-embedded-workbench/>.
- [23] “Smartrf studio.” <https://www.ti.com/tool/SMARTRFTM-STUDIO>.
- [24] “Parámetros en fotografía (exposición).” <https://www.dzoom.org.es/los-tres-elementos-que-afectan-a-la-exposicion-en-tus-fotos/>.
- [25] “Acerca del control de versiones.” <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>.
- [26] “Repositorio git de tesis.” <https://github.com/nachomoledo/TESIS>.
- [27] “What is uml.” <https://www.uml.org/what-is-uml.htm>.
- [28] “Plantuml.” <https://plantuml.com/>.
- [29] J. Grenning, *Test Driven Development for Embedded C*. 2011.
- [30] “Opencv library.” <https://opencv.org/>.
- [31] B. Optronic, *Baumer-GAPI SDK, Programmer’s Guide*. 2012.
- [32] “Baumer gapi and opencv.” <https://www.baumer.com/ca/en/service-support/technical-information-industrial-cameras/baumer-gapi-and-opencv/a/baumer-gapi-and-opencv>.
- [33] “Interrupciones.” <https://es.wikipedia.org/wiki/Interrupci%C3%B3n>.
- [34] “Baumer for amr.” <https://www.baumer.com/us/en/product-overview/industrial-cameras-image-processing/software/baumer-gapi-sdk/linux-arm>.

A. Guía de instalación

A.1. Link del repositorio en Github

El código fuente del proyecto se encuentra en <https://github.com/nachomoledo/tesiskrenzmoledo>

A.2. Instalación Baumer GAPI

Se procede con la instalación de Baumer GAPI SDK, en este caso, la versión 2.8. El SDK permitirá cambiar parámetros de configuración de la cámara, capturar fotografías y ver video en tiempo real.

Baumer para Raspberry Pi es compatible con las interfaces GigE y USB3.0. Se utilizará el protocolo GigE para realizar la conexión a través de un cable Ethernet estándar entre la cámara y el puerto Ethernet de la Raspberry Pi.

Baumer GAPI se basa en GenICam. GenICam es un estándar y significa Generic Interface for Cameras (interfaz genérica para cámaras). El objetivo del estándar es desacoplar la tecnología de interfaz de la cámara de la interfaz de programación de aplicaciones de usuario (API). Baumer GAPI tiene una interfaz GenICam.

El paquete de instalación contiene:

- SDK.
- Camera Explorer: para ver imágenes en tiempo real. (herramienta de visualización y evaluación en tiempo real).
- Ejemplos en C++ y las Librerías correspondientes.

A.2.1. Instalación de software

Ya que para todas las versiones de Linux instalar un paquete a través de la consola [Terminal] es esencialmente el mismo procedimiento, éste será el método de instalación a seguir.

A.2.1.1 Pasos preliminares

Antes de instalar el software, verifique previamente si alguna otra versión del SDK ya se encuentra instalada en su sistema, si es así, debe eliminarla antes de continuar.

Asegúrese de descargar la versión correcta de Baumer GAPI SDK para procesadores ARM correspondiente a la placa Raspberry Pi 3 o 4 [34].

A.2.1.2 Instalación de Baumer GAPI SDK

- Cambie a modo administrador. Para ello, escriba en la consola [terminal] “su root”. Ingrese luego el password del administrador.

- Cambie al directorio donde descargó el archivo de instalación. Si el mismo se encuentra en la carpeta de descargas, escriba en la consola “cd Downloads/”.
- Liste el contenido de la carpeta con el comando “ls”. Debería ver el archivo descargado.
- Instale el paquete con el comando “dpkg -i ‘archivo_descargado.deb’”.

A.2.1.3 Consultando los archivos instalados

Para verificar la correcta instalación del SDK, escriba el comando “dpkg-query -L baumer-gapi-sdk-linux”.

Possible salida:

```
/etc/ld.so.conf.d/lib_baumer.conf
/usr
/usr/local
/usr/local/src
/usr/local/src/baumer
/usr/local/src/baumer/sdk_example
/usr/local/src/baumer/sdk_example/C++
/usr/local/src/baumer/sdk_example/C++/src
```

A.2.1.4 Consultando información sobre el paquete instalado

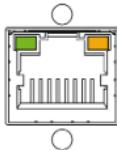
Ingresé el comando “aptitude show baumer-gapi-sdk-linux”. Possible salida:

```
Package: baumer-gapi-sdk-linux
New: yes
State: installedF
Automatically installed: no
Version: 2.6
Priority: extra
Section: misc
Maintainer: support.cameras@baumer.com
Uncompressed Size: 8.192
Depends: libc6 (>= 2.11.3), libstdc++6 (>= 4.4.5), libxcb1 (>= 1.6)
Description: Baumer GAPI SDK Linux
```

A.2.1.5 Eliminar el paquete instalado

- Cambie a modo administrador. Para ello, escriba en la consola [terminal] “su root”. Luego, Ingrese el password del administrador.
- Remueva el paquete con el comando “dpkg -r baumer-gapi-sdk-linux”.

**GigE Vision interface
Accepts RJ-45 with thumbscrews**

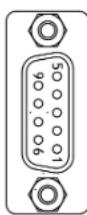


Pin 1	TRD+(o)	5	TRD-(z)
2	TRD-(o)	6	TRD-(i)
3	TRD+(i)	7	TRD+(3)
4	TRD+(2)	8	TRD-(3)

Figura 65: RJ-45 pin out

**D-sub 9 pin connector
for GPIO (Auxiliary)**

Type: DD-09SSG



No.	I/O	Name
1	I	LVDS IN1-
2	I	LVDS IN1+
3	I	TTL IN 1*
4	O	TTL OUT 1
5		GND
6		NC
7		NC
8	O	TTL OUT 2
9		GND

*Can be set to 75Ω terminator
via internal switches

Figura 66: D-sub pin out para GPIO

A.2.2. Instalación de hardware

En las (*Figuras 65 y 66*) se observan las conexiones de la cámara tanto para el puerto RJ-45 como para el puerto D-Sub utilizado como GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General). La (*Figura 67*) representa la correcta interpretación de los leds indicadores de la cámara, los cuales aportan información sobre su estado y funcionamiento.

The rear panel mounted LED provides the following information:

Power Trig LED

- Amber: Power connected - initiating
 - Steady green: Camera is operating in Continuous mode
 - Flashing green: The camera is receiving external trigger
- LINK LED
- Steady green: 1000 Base-T has been connected
 - Flashing green: 100 Base-TX has been connected
- ACT LED
- Flashing amber: Network active in communication

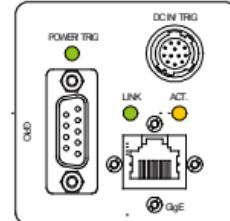


Figura 67: Interpretación de LEDs en la cámara JAI

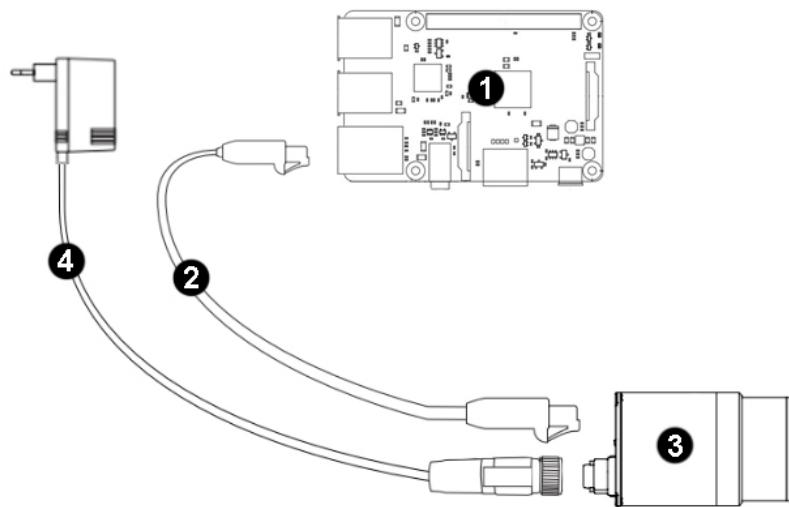


Figura 68: 1) Raspberry Pi 3 Model B. 2) Cable GigE. 3) JAI BM-500GE Serie B. 4) Fuente de alimentación de 12V.

A.2.2.1 Instrucciones

Basándose en la (*Figura 68*):

- Conecte la cámara (3) a la placa Raspberry Pi 3 (1) a través de un cable Ethernet (2) (al menos Cat-5e) utilizando los puertos RJ-45.
- Conecte la cámara a una fuente de alimentación de 12V (4).

A.2.3. Configuración de red

Establezca una dirección IP estática en la Raspberry Pi. La porción de red de esta dirección, debe ser idéntica a la de la cámara.

Una forma de configurar los ajustes de red para la transferencia de datos entre la cámara y la RPi, es ajustar la configuración en el archivo de interfaz. Para realizar cambios en este archivo, primero debe cambiar al modo Administrador.

El archivo de interfaz se encuentra en la carpeta: "/etc/network".

En la (*Tabla 32*) se muestran los parámetros que pueden ajustarse en el archivo de interfaz:

Ejemplo

```
auto eth1
iface eth1 inet static
address 10.0.0.10
```

Comando	Descripción
ethx	x = el número de la interfaz (ej: 1)
address	Dirección IP de la RPi (ej: 10.0.0.10)
netmask	Máscara de Red (ej: 255.255.255.0)
broadcast	Broadcast de la red (ej: 10.0.0.255)
mtu	Unidad de transmisión máxima (ej: 9000 bytes)

Tabla 32: Parámetros modificables en el archivo de interfaz

```
netmask 255.255.255.0
broadcast 10.0.0.255
mtu 9000
```

Por su parte, como se mencionó anteriormente, la dirección IP de la cámara, debe coincidir con la dirección de red de la RPi. Por ejemplo: 10.0.0.20.

A.2.4. Camera Explorer

Junto a la instalación del SDK, Baumer proporciona Camera Explorer, una herramienta de visualización y evaluación en tiempo real que a su vez, permite configurar la cámara y ver sus especificaciones.

Para ejecutarlo, escriba en la consola [terminal] el comando “bexplorer”. Se abrirá una nueva ventana con la previsualización de la cámara.

A.3. Instalación OpenCV

Se instalará OpenCV con los módulos adicionales para añadir nueva funcionalidad. En un principio, estos nuevos módulos deben desarrollarse por separado y publicarse en el repositorio “opencv_contrib”. Más tarde, cuando el módulo madura y gana popularidad, se traslada al repositorio central de OpenCV, y el equipo de desarrollo proporciona soporte para este módulo.

A.3.1. Mantenga Ubuntu o Raspbian actualizado

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y autoremove
```

A.3.2. Instalación de dependencias

A.3.2.1 Build tools

```
sudo apt-get install -y build-essential cmake
```

A.3.2.2 GUI

Si desea usar GTK en lugar de Qt, reemplace 'qt5-default' por 'libgtkglext1-dev' y elimine la opción '-DWITH_QT=ON' en CMake.

```
sudo apt-get install -y libgtkglext1-dev libvtk6-dev
```

A.3.2.3 Media I/O

```
sudo apt-get install -y zlib1g-dev libjpeg-dev libwebp-dev libpng-dev libtiff5-dev libjasper-dev  
libopenexr-dev libgdal-dev libgphoto2-dev
```

A.3.2.4 Video I/O

```
sudo apt-get install -y libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev libtheora-  
dev libvorbis-dev libxvidcore-dev libx264-dev yasm libopencore-amrnb-dev libopencore-amrwb-dev  
libv4l-dev libxine2-dev v4l-utils
```

A.3.2.5 Bibliotecas de paralelismo y álgebra lineal

```
sudo apt-get install -y libtbb-dev libeigen3-dev
```

A.3.2.6 Python

```
sudo apt-get install -y python-dev python-tk python-numpy python3-dev python3-tk python3-  
numpy
```

A.3.2.7 Java

```
sudo apt-get install -y ant default-jdk
```

A.3.2.8 Documentación

```
sudo apt-get install -y doxygen
```

A.3.3. Instale la biblioteca (Puede cambiar '3.2.0' por la última versión estable)

```
sudo apt-get install -y unzip wget  
wget https://github.com/opencv/opencv/archive/3.2.0.zip -O opencv320.zip unzip opencv320.zip  
rm opencv320.zip  
mv opencv-3.2.0 OpenCV
```

```
cd OpenCV  
touch OpenCV3.2withContrib
```

A.3.4. Instale la biblioteca Open_Contrib (Puede cambiar '3.2.0' por la última versión estable)

```
wget https://github.com/opencv/opencv_contrib/archive/3.2.0.zip -O opencv_contrib320.zip  
unzip opencv_contrib320.zip  
rm opencv_contrib320.zip  
mv opencv_contrib-3.2.0 OpenCV_contrib
```

A.3.5. Buildear OpenCV con contrib

```
mkdir build  
cd build  
cmake -DOPENCV_EXTRA_MODULES_PATH=../OpenCV_contrib/modules -DWITH_QT=OFF  
-DWITH_OPENGL=ON -DFORCE_VTK=ON -DWITH_TBB=ON  
-DINSTALL_C_EXAMPLES=OFF -DWITH_GDAL=ON -DWITH_XINE=ON  
-DBUILD_EXAMPLES=OFF -DENABLE_PRECOMPILED_HEADERS=OFF ..  
make -j ((nproc) + 1)  
sudo make install  
sudo ldconfig
```

A.3.6. Solución a errores

A.3.6.1 Dependencias faltantes

En algunas versiones de Ubuntu, será necesario instalar dependencias adicionales para el correcto funcionamiento de OpenCV. Para ello, ejecutamos el siguiente comando:

```
sudo apt install build-essential cmake git pkg-config libgtk-3-dev libavcodec-dev libavformat-dev  
libswscale-dev libv4l-dev libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev gfortran ope  
nexe libatlas-base-dev python3-dev python3-numpy libtbb2 libtbb-dev libdc1394-22-dev
```

A.3.6.2 Error al compilar opencv con ffmpeg

Otro error que puede darse durante la instalación, es que el compilador no encuentre variables definidas, consiguiendo errores como: 'CODEC_FLAG_GLOBAL_HEADER' was not declared in this scope.

Para solucionarlo, copie las siguientes definiciones:

```
#define AV_CODEC_FLAG_GLOBAL_HEADER (1 "menor""menor" 22)  
#define CODEC_FLAG_GLOBAL_HEADER AV_CODEC_FLAG_GLOBAL_HEADER
```

```
#define AVFMT_RAWPICTURE 0x0020
```

al inicio del archivo: "opencv-3.2.0/modules/videoio/src/cap_ffmpeg_impl.hpp"

A.3.6.3 Error de constantes

Otro error que puede aparecer, es: "invalid conversion from 'const char*' to 'char*' [-fpermissive]".

Para solucionarlo, cambie
"char* str = PyString_AsString(obj);"
por
"const char* str = PyString_AsString(obj);"
en el archivo "opencv3/modules/python/src2/cv2.cpp".

A.4. Instalación QT

Para instalar Qt y los módulos necesarios, se deberá seguir los siguientes pasos:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install qt5-default
sudo apt-get install qtcreator
sudo apt-get install qt5-qmake
sudo apt-get install libqt5serialport5-dev
sudo apt-get install qtdeclarative5-dev
```

- Abra Qtcreator y vaya a Help > "about plugins" y desmarque "Remote Linux".
- Reinicie la Raspberry Pi.
- Vuelva a abrir Qtcreator, vaya a "Tools > Options > Build and Run" y vaya a Compilers. Agregue "GCC" y configure la ruta del compilador como "/usr/bin/gcc".

A.4.1. solución de errores

A.4.1.1 Solución a Aborted (core dumped)

Para solucionar el error: "This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem."

Se deberá reinstalar la biblioteca con el siguiente comando: "sudo apt-get install --reinstall libxcb-xinerama0"

A.5. Creación de puertos virtuales

Para crear dos puertos virtuales y poder conectar la ES, la OBC y la IAS en el proceso de simulación, se deberá ejecutar el siguiente comando por consola:

```
socat -d -d pty,raw,echo=0 pty,raw,echo=0
```

Esto creará un enlace virtual de dos puertos. Se deberá ejecutar este comando tres veces, uno por cada computadora.

A.6. Conexión del EAV

A.6.1. Diagramas de conexión

En las (*Figuras 69 y 70*) se muestran los diagramas de conexión para el circuito emisor. En las (*Figuras 71 y 72*) se muestran los diagramas de conexión para el circuito receptor.

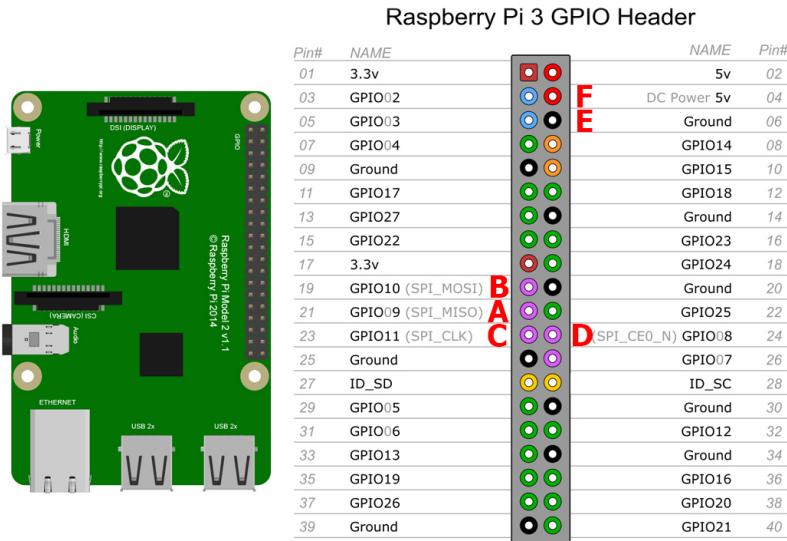


Figura 69: Conexión RPi

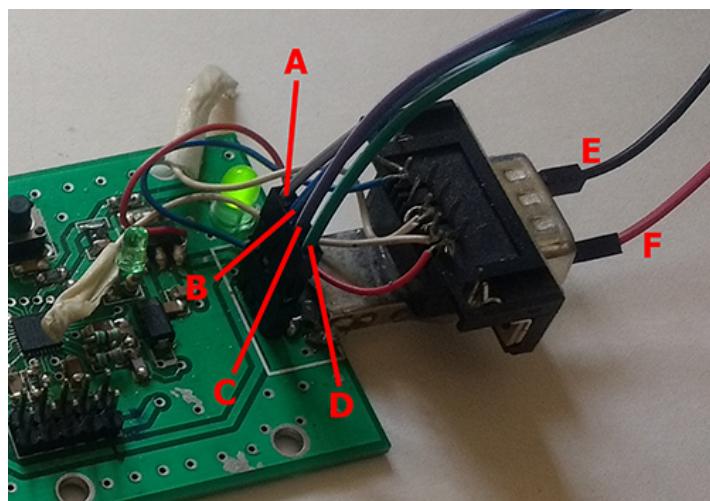


Figura 70: Conexión emisor

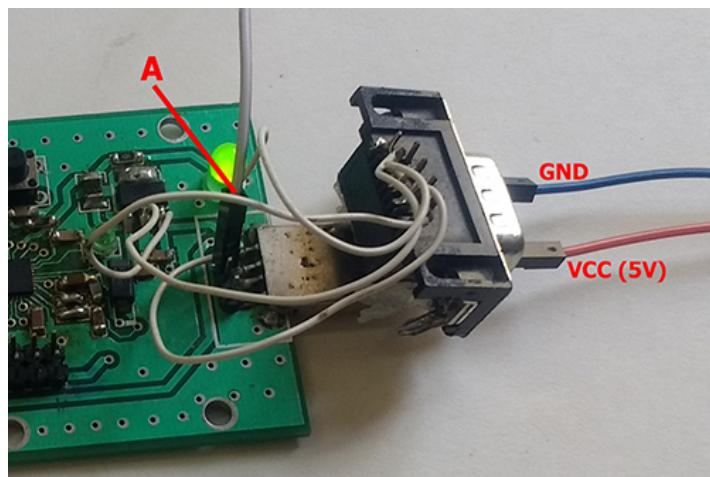


Figura 71: Conexión receptor

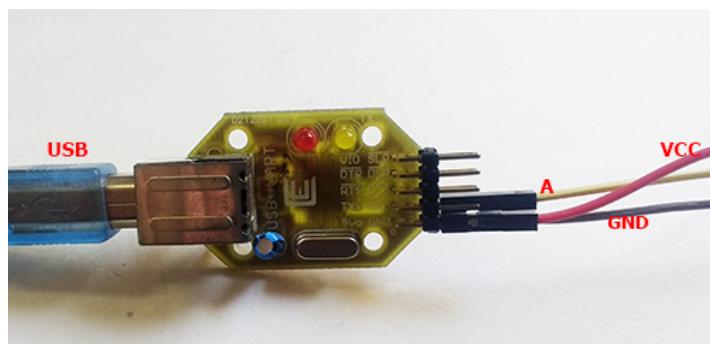


Figura 72: Conexión conversor UART

A.7. Software y conexión de la OBC

A.7.1. Software

Instalar IDE Code Composer Studio para Windows desde la página oficial [20] (Version 10.4.0.00006).

Instalar HalCoGen desde la página oficial [21] (Versión 04.07.01).

A.7.2. Conexión de la OBC

- Puerto SCI: pines 106 W3 (RX) y 113 N2 (TX) conectados a pines GPIO14 (TX) y GPIO15 (RX) respectivamente.
- Puerto SCILIN: pines LINRX A7 y LINTX B7 conectados a módulo UART-USB de la ES (PC).

Conectar además las tierras y desconectar cable de debug de la placa TMS570.