ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“DISEÑO Y EVALUACIÓN DE UNA RED PEER-TO-PEER CON PROTOCOLO DE COMUNICACIÓN LoRaWAN PARA AMPLIAR LA COBERTURA DEL SISTEMA RTK-GPS, PARA APLICACIONES DE CAPTURA DE IMÁGENES MEDIANTE UN UAV EN ENTORNOS DE DENSA VEGETACIÓN"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERA/O EN TELECOMUNICACIONES

KERLY MARIUXI OCHOA ERAZO

PEDRO LUIS VILLEGAS ROMÁN

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTOS

Agradecemos todas las personas que nos acompañaron a lo largo de nuestra formación, que hicieron posible la realización y culminación de esta Ingeniería.

Agradecemos al Centro de Visión y Robótica (CVR) por brindarnos la valiosa oportunidad de trabajar con los elementos necesarios para realizar este trabajo.

Agradecemos a los Docentes César Yépez, Daniel Ochoa por compartir sus valiosos conocimientos técnicos en su calidad de tutores profesionales y académico respectivamente, además al Ing. Christian Sacarelo por su ayuda incondicional en las pruebas y brindando su experiencia.

DEDICATORIA

A mi padre Osman Ochoa Castillo, a mi madre Mónica Erazo Salgado y a mi hermano Osman.

Kerly Ochoa

A mi familia en especial a madre.

Pedro Villegas

TRIBUNAL DE EVALUACIÓN

MSc. César Yépez

PROFESOR EVALUADOR

PhD. Francisco Novillo

PROFESOR EVALUADOR

MSc. Juan Aviles

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Kerly Mariuxi Ochoa Erazo

Pedro Luis Villegas Román

RESUMEN

En este trabajo se diseña e implementa una red peer-to-peer de bajo costo para ampliar la cobertura del sistema de posicionamiento global cinemático en tiempo real (RTK-GPS), se utiliza el protocolo de comunicación de una red de área amplia de largo rango (LoRaWAN). Posterior se construye el modelo de propagación de largo rango (LoRa) de forma empírica y se aplica en la geolocalización de un vehículo aéreo no tripulado (UAV) implementado con el sistema RTK-GPS+LORA en un entorno con densa vegetación.

ÍNDICE GENERAL

AGRADECIMIENTOS ii
DEDICATORIA iii
TRIBUNAL DE EVALUACIÓN iv
DECLARACIÓN EXPRESA v
RESUMEN vi
ÍNDICE GENERAL vii
CAPÍTULO 1 1
1. INTRODUCCIÓN 1
1.1 Descripción del problema 2
1.2 Justificación 2
1.3 Objetivos 3
1.3.1 Objetivo General 3
1.3.2 Objetivos Específicos 3
1.4 Metodología 3
1.5 Resultados esperados 4
1.6 Elementos diferenciadores 4
CAPÍTULO 2 6
2. MARCO TEÓRICO 6
2.1 Sistema de posicionamiento global 6
2.1.1 Sistema RTK-GPS 7
2.2 Sistema Piksi 8
2.3 Dispositivo UAV 8
2.3.1 UAV de la empresa 3D Robotics modelo X8+ (3DR X8+) 9

2.4 Dispositivo LoRa 9
2.4.1 Dragino LoRa 10
2.5 Red Peer to Peer 11
2.6 Sistema Embebido 11
2.6.1 Raspberry Pi 3 modelo B 12
2.7 Dispositivo de captura de imágenes 13
2.7.1 GoPro Hero 3 13
CAPÍTULO 3 14
3. DISEÑO, INTEGRACIÓN Y EVALUACIÓN DE SISTEMAS 14
3.1 Sistema RTK-GPS 14
3.1.1 Implementación del sistema RTK-GPS 14
3.2 Dispositivo UAV 18
3.2.1 Integración de la cámara GoPro Hero 3 20
3.3 Integración del sistema RTK-GPS con el UAV 21
3.3.1 Configuración de Hardware 22
3.3.2 Configuración de Software 23
3.4 Diseño de la red peer to peer al usar tecnología LoRa 25
3.4.1 Recursos energéticos 27
3.4.2 Elementos de la red 27
3.4.3 Programación de los nodos 28
3.4.4 Infraestructura física de la red P2P 31
3.4.5 Planificación de vuelo del UAV 33
3.5 Sistema embebido para la captura de datos 36
CAPÍTULO 4 38
4. RESULTADOS 38
4.1 Evaluación de los resultados de la red peer to peer (LoRa - RTK-GPS) 38

4.2 Imágenes capturadas por el dispositivo UAV 41
4.3 Aproximación de un modelo de propagación para entornos vegetativos 44
4.3.1 Análisis de regresión lineal 46
4.3.2 Cálculos matemáticos para el modelo de propagación 47
CONCLUSIONES Y RECOMENDACIONES 49
BIBLIOGRAFÍA..... 51
ANEXOS 54

CAPÍTULO 1

1. INTRODUCCIÓN

La automatización de operaciones en entornos de vegetación es necesaria para el desarrollo de diversas áreas como la agrícola, arqueológica y topográfica del Ecuador. Por ejemplo, la identificación y estudio de plantaciones agrícolas, para ello se requiere mantener el control en un área específica.

La mayoría de los sistemas de exploración existentes son usualmente controlados por un operador en tierra y fueron diseñados para entornos urbanos. El alto costo de operación y complejidad de empleo los hace no apropiados para realizar trabajos en zonas con vegetación. Sin embargo, con el auge de los vehículos aéreos no tripulados (UAV), que se emplean como instrumento de adquisición de datos, para la medición de parámetros y exploración. Para que un UAV de bajo costo sea útil en tareas rutinarias de exploración, es necesario dar a estos dispositivos la capacidad de realizar vuelos autónomos y esto se logra al implementar sistemas de posicionamiento global (GPS).

El informe consta de 4 capítulos. El primer capítulo trata acerca del problema al realizar exploración con un UAV el cual emplea un GPS convencional, el segundo capítulo detalla la tecnología necesaria en este trabajo que permite crear la red para la aplicación determinada, el tercer capítulo trata en detalle la metodología, diseño e implementación de la red de comunicación y en el cuarto capítulo se muestra los resultados del trabajo, cálculos y el modelo de propagación.

En el primer capítulo se incluye la descripción del problema, justificación, objetivos, metodología, los resultados esperados de las hipótesis y los elementos diferenciadores del trabajo. En todo este capítulo se aproxima al lector dentro del entorno y diseño del sistema que se espera obtener.

En el segundo capítulo se incluye el procedimiento estándar, herramienta o modelo utilizado para resolver el problema conforme a las características técnicas de la tecnología.

El tercer capítulo se enfocará en el diseño, implementación, experimentación y adquisición de datos. En este capítulo también se da detalles sobre la solución.

El cuarto capítulo contiene los resultados del diseño de red, el experimento en un ambiente con densa vegetación, la descripción del proceso matemático y el modelo de propagación recomendado para este tipo de experimentos.

1.1 Descripción del problema

En los últimos años se ha incrementado el uso de vehículos aéreos no tripulados (UAVs), como una herramienta principal para una visualización aérea sectorizada y medición de parámetros que geo-referencian la imagen; en busca de una alta precisión para la geo-referencia se recomienda integrar un sistema diferencial de cinemática en tiempo real con posicionamiento global (RTK-GPS) con un alcance máximo de 200 metros.

Debido a esta limitante de cobertura de transmisión, en este trabajo se propone implementar un sistema de transmisión de largo rango (LoRa) para mejorar el alcance del sistema RTK-GPS, el estudio es realizado en un ambiente de densa vegetación.

1.2 Justificación

Dado el auge de los últimos años de las comunicaciones inalámbricas en el ámbito del internet de las cosas (IoT), medición de seguridad de máquinas, la industria, la agricultura, entre otras, se han desarrollado sistemas que permitan la comunicación a grandes distancias con un bajo consumo energético.

El desarrollo de nuevas tecnologías incrementa el alcance de los dispositivos de comunicación para diferentes fines y entornos. Uno de los sistemas más recientes es el LoRa, el cual permite realizar un radio enlace con técnicas de modulación de espectro ensanchado, lo que envía datos a muy bajas tasas de transferencia y con un alcance de hasta 21 Km en un entorno con línea de vista (LOS).

En este trabajo se propone implementar una red de sensores inalámbricos (WSN) para mitigar las limitaciones del sistema RTK-GPS agregado en el UAV. Juntamente con este proceso se capturan los parámetros que son necesarios

para la comunicación inalámbrica y se usa como transmisor a los dispositivos LoRa. La recolección de estos datos permite observar el comportamiento de la señal en una zona establecida, para posterior análisis y encontrar un modelo de propagación que se adapte a las necesidades y optimice el diseño de WSN.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar una red peer to peer que extienda la cobertura del sistema RTK-GPS y aplicar la geolocalización en un entorno de densa vegetación.

1.3.2 Objetivos Específicos

- Implementar el sistema RTK-GPS +LoRa en un dispositivo UAV para su respectiva geolocalización y captura de imágenes.
- Extender la cobertura de comunicación inalámbrica de la red peer to peer a través de módulos LoRa y evaluar la viabilidad técnica en un entorno de vegetación.
- Recolectar mediciones del sistema físico al variar la distancia de comunicación inalámbrica, para encontrar un aproximado a un modelo de propagación dado un ambiente de densa vegetación.

1.4 Metodología

El sistema RTK-GPS consta de dos elementos, uno llamado estación base (BS) la cual se establece en un punto fijo y otro llamado rover este es móvil. Estos elementos del sistema RTK-GPS se deben enlazar para establecer una comunicación y de esta manera realizar la corrección en la ubicación del rover lo cual minimiza el error en cuanto a la posición GPS del mismo. El UAV brinda la capacidad de ser equipado con el sistema antes mencionado. Además, se implementa una cámara para obtener fotografías.

El escenario antes expuesto presenta un problema con respecto a la distancia de separación entre los elementos del sistema RTK-GPS, por este motivo se cambia el sistema de transmisión del dispositivo por módulos LoRa que tienen mayor rango de cobertura de hasta 1.5km con obstrucciones en el camino, después de

realizar esta extensión se observa el funcionamiento y comunicación inalámbrica entre dispositivos.

En el dispositivo UAV se monta los sistemas enlazados, evaluados y analizados, se procede a montar la red peer to peer (P2P) para llevar acabo la aplicación de captura de imágenes en una zona con vegetación, imágenes que en posteriormente se geo-etiquetan.

Para la etapa final se obtendrá la información de potencia recibida, y el rango de cobertura de la comunicación entre los nodos de la red P2P, se mantienen varios parámetros estables y se varía la distancia de la comunicación inalámbrica, en un ambiente con densa vegetación, y se verifica la relación entre la señal recibida y la transmitida. Analizados los datos detectados se reconocerá un modelo estadístico de regresión lineal que se asemeje a los datos obtenidos, para encontrar un modelo empírico de propagación apropiado para este escenario.

1.5 Resultados esperados

Se espera incrementar el alcance en la cobertura del sistema RTK-GPS de 200m a 1km de radio de cobertura, y obtener las imágenes geo-etiquetadas.

Evaluación de la calidad de la comunicación en una zona con vegetación por medio de parámetros como el indicador de fuerza de la señal recibida (RSSI).

Determinar una relación entre la distancia entre los nodos y la pérdida de potencia recibida en el ambiente de vegetación, y a su vez un estudio para encontrar un modelo empírico de propagación que permita predecir el alcance de la señal y poder cubrir un área determinada.

1.6 Elementos diferenciadores

El presente trabajo se destaca por la implementación de un sistema RTK-GPS con capacidad de escalabilidad, cobertura a larga distancia, bajo consumo energético y bajo costo.

Este trabajo se dirige a aplicaciones WSN, tales como sistema de geolocalización en la agricultura, zonas rurales inteligentes, automatización en sembríos, y así sucesivamente. Lo cual mejoraría la calidad de la agricultura, debido a que con

este sistema se puede realizar análisis remotamente de parámetros del ambiente y visualización de un área seleccionada, que se considera beneficioso para el cambio de la matriz productiva del Ecuador.

CAPÍTULO 2

2. MARCO TEÓRICO

2.1 Sistema de posicionamiento global

Los sistemas de posicionamiento basados en satélites han sido empleados durante las últimas décadas, pero la precisión de posicionamiento con un solo receptor es limitada. Hoy en día existe la posibilidad de usar satélites para el posicionamiento y la navegación, uno de los sistemas más conocidos es el GPS americano. Con el pasar de los años otros países han desarrollado sus propias soluciones de posicionamiento y navegación tal como el Sistema de Navegación Global por Satélite (GLONASS) ruso o el sistema de navegación satelital europeo Galileo. El término genérico empleado es Sistema global de navegación por satélite (GNSS) [1].

El GPS fue creado para determinar ubicaciones en la Tierra a partir de posiciones conocidas de los satélites, fue desarrollado por el Departamento de Defensa de los Estados Unidos de América para fines militares para poder determinar posición, velocidad y tiempo en un marco de referencia común, con el pasar de los años se extendió su uso al campo civil.

El GPS emplea seudo-rangos que provienen de las emisiones de las señales de satélite, este se obtiene ya al medir el tiempo de viaje de la señal, la cual se encuentra codificada, y multiplicando este por su velocidad o por medio de la medición de la fase de la señal, en ambos casos se emplean los relojes del receptor y del satélite. Dado que estos relojes jamás están precisamente sincronizados no se pueden obtener rangos reales por lo que a estos se los llaman seudo-rangos los cuales poseen un error de sincronización el cual es tomado en cuenta al momento de determinar la ubicación de un punto [2].

Existen muchos factores que comprometen la precisión del GPS algunos de los cuales pueden ser solventados como el efecto de Sagnac el cual se debe a errores relativistas debido a la rotación de la tierra durante el tiempo de transmisión de la señal y que de no ser corregido puede conducir a un error en la

posición en el orden de los 30 metros. Otro de los factores que influyen en la precisión son los atmosféricos los cuales son de más compleja resolución, un efecto conocido como retraso de la troposfera depende de factores locales de temperatura, humedad relativa y presión atmosférica de donde se encuentra ubicado el receptor GPS. Para solventar estos errores se usan una serie de modelos matemáticos que reducen el error producido por este fenómeno [2].

Se han desarrollado varios métodos a lo largo de los años para aumentar la precisión del GPS y uno de los más empleados es el GPS diferencial (DGPS) el cual mejora el desempeño del posicionamiento y sincronización del GPS para lo cual se emplea varias estaciones de referencia en ubicaciones conocidas. Estas estaciones envían información a los usuarios por medio de enlace de datos el cual no es en tiempo real. Esta información puede ser correcciones de las mediciones de seudo-rango, corrección del reloj de los satélites, datos auxiliares como la ubicación de la estación de referencia.

2.1.1 Sistema RTK-GPS

El RTK-GPS es la técnica empleada para mejorar la precisión del GPS que al igual que el método de DGPS emplea una o más estaciones base, y cuya diferencia radica en que las correcciones se realizan en tiempo real. Estas estaciones base se encuentran en una ubicación conocida desde la cual se emite por medio de un radio transmisor las mediciones y coordenadas recibidas desde los satélites a un rover cuyo software embebido se encarga de procesar las posiciones GPS transmitidas desde la BS y el propio rover, para obtener las coordenadas corregidas [3].

Los datos emitidos desde la BS llegan al rover con cierto retardo o latencia, esto ocurre debido al procesamiento que se debe realizar para poder transmitir estos datos lo cual produce una degradación en la precisión del posicionamiento. La precisión esperada con este método está en el orden de unos pocos milímetros aproximadamente 8mm o a varios centímetros menor que 10cm.

2.2 Sistema Piksi

La compañía “Swift Navigation” ha desarrollado el sistema Piksi que se caracteriza por ser de desarrollo abierto y de bajo costo además de que implementa la metodología RTK-GPS [4]. El Piksi es un receptor GPS de alto desempeño con funcionalidad RTK-GPS que permite una precisión de ubicación de centímetros de error (10cm), es portable y de bajo consumo energético lo cual lo hace ideal para ser integrado en UAVs y otros equipos portátiles, con una comunicación serial.

El Piksi trabaja con el protocolo de comunicación “Swift Navigation Binary Protocol” SBP que está basado en el protocolo del “National Marine Electronic Association” (NMEA). El protocolo SBP se caracteriza por ser minimalista para realizar comunicación binaria entre dispositivos de la empresa [5]. El SBP es empleado por el sistema Piksi para transferir mensajes correspondientes a la navegación, mensajes de error y configuración por nombrar algunos, entre la BS y el rover.

La trama consiste en 6 bytes de encabezado, una carga útil de longitud variable de bytes y 16 bits que se emplean como sistema de verificación de redundancia cíclica (CRC), el resultado final es una trama de 8 + N bytes, en donde N representa un número de bytes variable. La gráfica de la Figura 2.1 a continuación muestra la arquitectura de la trama.

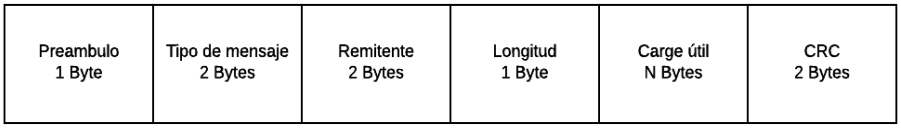


Figura 2.1: Trama del protocolo SBP.

2.3 Dispositivo UAV

El uso de UAV en diferentes aplicaciones ha tenido un gran apogeo, estos dispositivos pueden realizar una variedad de tareas lo que ha permitido disminuir costos en determinados casos, además de reducir la complejidad de realizar una tarea al tener vuelos autónomos. En este trabajo se utiliza software y hardware

accesible y abierto a fin de permitir la reproducción del experimento, además esto facilita realizar mejoras en un futuro.

Se ha seleccionado un UAV de bajo costo que consta de 8 rotores lo que permite una mejor estabilidad, maniobrabilidad, y es capaz de llevar una gran variedad de instrumentos [6, 7].

2.3.1 UAV de la empresa 3D Robotics modelo X8+ (3DR X8+)

El UAV 3DR X8+ es un octocóptero con una estructura tipo x, tiene una capacidad de carga útil de hasta 1.7lbs y puede volar una distancia de hasta 3.4 km, sin carga útil, se eleva hasta los 100m, funciona con una batería Lipo de cuatro celdas con un voltaje de 14.8V y con una capacidad de 10000mAh, emplea un software embebido llamado “Ardupilot” en con una adaptación de firmware llamado “Copter” [8]. En este UAV se instala una gran variedad de instrumentos entre ellos los necesarios para llevar acabo la determinación de la ubicación y la aplicación de captura de imágenes.

En el software para el computador, permita planear rutas de vuelo mediante la ubicación de puntos de coordenadas geográficas, y también acceder a todos los parámetros del UAV.

2.4 Dispositivo LoRa

Para este trabajo se emplea una tecnología que posee estabilidad, cobertura de larga distancia, bajo consumo de energía, bajo costo y que permita la interconexión de una variedad de dispositivos. Las tecnologías de comunicación para crear infraestructura de transmisión consideran su cobertura, velocidad de datos y consumo de energía [9, 10].

Dentro del campo de las redes de comunicaciones inalámbricas de largo alcance y baja potencia (LPWAN) actual, se encuentran entre las tecnologías más destacados a LoRa Y SIGFOX, las cuales se pueden considerar como buenos candidatos para aplicaciones que necesitan recorrer grandes distancias en diferentes entornos que incluye ambientes de vegetación. Otras tecnologías para tener en cuenta son los sistemas WIFI y los diferentes protocolos como ZigBee y

6LoWPAN que son buenos candidatos debido a su bajo consumo de energía, pero cuentan con una menor área de cobertura. Finalmente se ubican las tecnologías GSM/ CDMA/ WCDM/ 4G permiten conectividad de larga distancia, pero se consideran menos eficientes en consumo de energía [11, 12].

2.4.1 Dragino LoRa

La empresa Dragino ha desarrollado un módulo basada en el circuito integrado Semtech SX1276/SX1278 los cuales implementan la tecnología LoRa la cual usa el protocolo de comunicación LoRaWAN, LoRa es un esquema de modulación de espectro ensanchado propietario que se basada en la modulación Chirp de espectro ensanchado (CSS).

Esta tecnología dispone de diferentes tasas de transmisión en función de la configuración, con velocidades de 3Kbps hasta 22Kbps, se puede obtener una sensibilidad por debajo de los -148dBm, la potencia de transmisión puede ser de hasta 20dBm y con comunicación de interfaz periférico serial o bus (SPI), entre otras características [11].

LoRa puede trabajar en diferentes bandas de frecuencias, según el plan nacional de radio frecuencias (PNF) [12] el Ecuador pertenece a la Región 2 y basándose en el reglamento de radiocomunicaciones de la Unión Internacional de Telecomunicaciones (ITU) artículo 5.150 del capítulo 2 sección 4 [13] el cual habla de las bandas designadas para aplicaciones industriales, científicas y médicas (ICM) se ha seleccionado un dispositivo que opere en la banda de los 915MHz.

Este módulo se debe acoplar a un Arduino para de esta manera poder realizar la programación y lograr el correcto funcionamiento del mismo.

Al referirse al LoRa a partir de ahora se toma en consideración que se habla de la unión del módulo LoRa de la empresa Dragino y el Arduino.

En este proyecto se escoge al módulo Arduino Mega, este dispositivo usa un microcontrolador ATmega2560, tiene más de un serial para la comunicación que se efectuará entre LoRa y Piksi. Otras características del Arduino Mega es el voltaje de alimentación de 5V la corriente usada

es de 68mA que permite un bajo consumo de energía, por ser compatible con LoRa, entre otras.

2.5 Red Peer to Peer

Para el despliegue de una WSN es indispensable conocer las necesidades básicas del sistema para establecer una comunicación entre nodos y la topología que se va a usar en la red.

La WSN consiste en elementos tecnológicos como sensores y enlaces de comunicación cuentan con la autonomía suficiente para que faciliten la obtención de información de varios nodos que se encuentren en un área determinada, además del protocolo de comunicación inalámbrica que va de acuerdo con la aplicación. La WSN empleada en este trabajo posee una topología de P2P y el protocolo de comunicación LoRaWAN [14].

La red entre iguales o P2P, funcionan como una serie de nodos sin clientes ni servidores. Las características de este tipo de red P2P relevantes para este proyecto son la escalabilidad, que en general, entre más nodos se conecten se tiene una red más grande y con mayor cobertura de comunicación, esto como beneficio por no tener fijo a un servidor y cliente; la robustez, conforme a la distribución en el caso de fallos en la réplica de datos hacia múltiples destinos permite encontrar la información sin hacer peticiones a un servidor [15].

Los dispositivos presentes en este trabajo permiten que cada nodo participe de manera autosuficiente y compartir información simultáneamente. Conjuntamente este tipo de red se aprovecha todo el ancho de banda de la tecnología LoRa, obtener el más alto rendimiento en conexiones y utilización de energía. Además, esta red permite tener una transmisión de datos a tiempo real, y todas estas características son reforzadas con el uso del protocolo LoRaWAN según su configuración.

2.6 Sistema Embebido

Los sistemas embebidos son dispositivos diseñados para realizar tareas específicas en diferentes entornos y se han empleado durante mucho tiempo, como el caso del cajero automático para obtener dinero, sin embargo, el

desarrollo tecnológico de la última década ha permitido que estos sistemas en general reduzcan su tamaño, además se ha facilitado el acceso a los mismos [16]. Han pasado de ejecutar un conjunto de instrucciones determinadas a poder ejecutar un sistema operativo reducido, con lo cual se pueden diseñar diferentes aplicaciones con un mismo hardware.

Ejemplos de esto son: Beagle bone Black, Intel Galileo y Raspberry Pi. Estos sistemas cuentan con una gran comunidad de desarrolladores a nivel global y son cientos los trabajos realizados. Cada uno posee una versión adaptada de Linux que es un sistema operativo de desarrollo abierto. El sistema embebido empleado en este trabajo es un Raspberry Pi 3 modelo B con sistema operativo Raspbian – Jessie el cual está basado la distribución Debian de Linux.

2.6.1 Raspberry Pi 3 modelo B

En este trabajo se ha seleccionado el raspberry como sistema embebido simplemente porque ya se tiene conocimiento previo en su uso y por contar con características que cubren las necesidades en este proyecto.

Algunas de las características de la Raspberry Pi 3 modelo B (CPU embebido) son las siguientes:

- Un 1.2GHz 64-bit Quad-Core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- Puertos USB
- 40 pines GPIO
- Puerto ethernet
- Ranura para tarjeta Micro SD

2.7 Dispositivo de captura de imágenes

En el UAV se monta una cámara GoPro Hero 3 en un soporte diseñado para la misma, las imágenes obtenidas durante el vuelo son almacenadas en una tarjeta microSD y posteriormente geo-referenciadas proceso en el cual se guarda la información correspondiente a longitud, latitud y altura del punto geográfico del área en donde se tomó la fotografía.

2.7.1 GoPro Hero 3

La cámara GoPro Hero 3 cuenta con conectividad WIFI, se puede realizar fotografías de hasta 11 Megapixels, cuenta con un modo de ráfaga de 30 imágenes por segundo y la posibilidad de poder realizar fotografías cada cierto intervalo de tiempo. Dispone de una ranura para tarjeta microSD para el almacenamiento de imágenes, se selecciona esta cámara por su alta resolución.

CAPÍTULO 3

3. DISEÑO, INTEGRACIÓN Y EVALUACIÓN DE SISTEMAS

En este capítulo se muestra el respectivo análisis y el diseño a detalle de la red P2P, además de la integración de los dispositivos y sistemas para la respectiva evaluación en un entorno con vegetación. Se establece la metodología y tecnología a usar para la implementación del sistema RTK-GPS con el dispositivo LORA con el objetivo de extender la cobertura.

3.1 Sistema RTK-GPS

El sistema con capacidad RTK-GPS empleado es el dispositivo Piksi, en la BS y rover ambos dispositivos son de iguales características de hardware y software, la diferencia radica en la configuración a nivel de software.

La BS se ubicada en un punto geo-referenciado conocido u obtenido previamente, es necesario que dicho punto cuente con una buena precisión de ubicación para de esta manera lograr reducir el error en la determinación de la ubicación del rover, luego de esto se configura la BS para que transmita la información del punto geo-referenciado.

El Piksi que cumple las funciones de rover se instala en el UAV, este se comunica con la BS de la cual recibe la transmisión del punto geo-referenciado y además obtiene información de su ubicación actual por medio de los satélites GPS. Con estas dos fuentes de datos realiza el procesamiento para determinar su ubicación y se obtiene un error medio menor a 10 cm.

3.1.1 Implementación del sistema RTK-GPS

La implementación del sistema RTK-GPS consiste en la implementación del dispositivo Piksi que al usar un programa de software desempeña funcionalidades especificas o permite la configuración conforme a los requisitos de cada aplicación a fin, en este proyecto se usa el programa de consola Piksi y como una de las características de este programa es

que permite observar claramente el proceso y el enlace entre los dispositivos Piksi.

A continuación, se describe como se implementa el sistema RTK-GPS de una manera general.

- **Instalación de la consola.** “Console” es un programa que permite realizar la comunicación entre el Piksi y el computador mediante una interfaz gráfica, cabe mencionar que es de desarrollo abierto. Lo primero que se debe realizar es la instalación de los controladores para el Piksi los cuales se encuentran disponibles para Windows, Mac OS X y Linux.

  El controlador virtual de puertos de comunicación (VCP) es provisto por la empresa Future Technology Devices International (FTDI) y sirve para realizar la comunicación vía USB con los dispositivos empleados en el presente trabajo. Durante el proceso de instalación se deben seleccionar los controladores VCP en la interfaz de usuario, realizada la instalación de los controladores se procede a descargar he instalar el programa “Console”.

  Se procede a conectar el Piksi al computador mediante el cable USB, hecho esto se debe verificar cual puerto de comunicación serial es asignado al Piksi, en la Figura 3.1 se puede observar lo anteriormente mencionado.

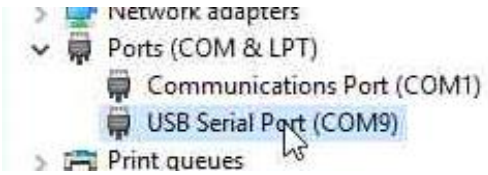


Figura 3.1: Visualización del puerto de comunicación asignado al Piksi.

Ahora se ejecuta el programa “Console” y se indica el puerto que ha sido asignado al Piksi y la tasa de baudios debe ser 1000000.

- **Consola.** El programa “Console” posee varias pestañas a través de las cuales se puede acceder a las diferentes opciones del sistema esto se puede observar en la Figura 3.2. El Piksi del rover y la BS se configuran al seleccionar la pestaña "Settings".

Figura 3.2: Pestañas de opciones del programa “Console”.

- **Configuración de la BS y el rover.** Lo siguiente es necesario para la correcta configuración de la BS y el rover.
  - La antena GPS se debe ubicar en una estructura estable en donde no haya obstrucciones y con vistas al cielo, se sujeta firmemente en una torre en el caso de la BS, para el caso del rover se instala en el UAV.
  - Conectar la antena GPS al Piksi.
  - Conectar el puerto UART A del Piksi al sistema de RF.
  - Conectar la antena del sistema de RF.

- **Configuración de software del Piksi de la BS.** Se conecta el Piksi al computador para realizar la configuración por medio del programa “Console”, se da click en “Save to flash” para guardar la configuración.

  Dentro de la ventana “Settings” en el listado de parámetros de la parte izquierda se busca la sección “UART UARTA”, se debe cambiar el valor del parámetro “SBP message mask” a 64, tal como se muestra en la Figura 3.3.

baudrate	1000000
uart uarta	
mode	SBP
sbp message mask	64
configure telemetry radio on boot	True
baudrate	57600
uart uartb	
mode	SBP

Figura 3.3: Opciones del puerto UART A de la BS.

A continuación, se busca la sección "Surveyed position" dentro del listado de parámetros y es aquí donde se establece la información del punto geo-referenciado. El parámetro "Broadcast" se debe establecer en "True", en "Surveyed lat" se establece la latitud del punto, en "Surveyed lon" se establece la longitud, en "Surveyed alt" se establece la altitud, estos valores corresponden al punto donde se ubica la BS, esto se muestra en la Figura 3.4.

surveyed position	
broadcast	True
surveyed lat	-2.142833
surveyed lon	-79.96775
surveyed alt	91
system info	

Figura 3.4: Información del punto geográfico de la BS.

- **Configuración de software del rover.** Dentro de la ventana “Settings” en el listado de parámetros de la parte izquierda se busca la sección “UART UARTA”, se debe cambiar el valor del parámetro “SBP message mask” a 0 como se muestra en la Figura 3.5 y posterior dar click en “Save to flash” para guardar la configuración.

uart uarta	
mode	SBP
sbp message mask	0
configure telemetry radio on boot	True
baudrate	57600
uart uartb	

Figura 3.5: Opciones del puerto UART A del rover.

Los LEDs rojos de ambos Piksi comenzarán a parpadear cuando se establezca un enlace de comunicación entre la BS y el rover, esto se puede verificar también en la pestaña “Observations”. Se debe esperar hasta que al menos haya 5 satélites en común entre ambos Piksi.

Luego los Piksi comenzarán a producir soluciones diferenciales, se debe ir a la pestaña “Baseline” y en un inicio se observa que el sistema se encuentra en modo “Float” la cual es la solución de ubicación menos precisa y eventualmente se obtendrá el modo “Fixed”. Esta transición entre modos tarda al menos 10 minutos y en el “RTK Fixed” se obtiene una precisión de centímetros en **la** ubicación del rover. En la Figura 3.6 se muestra la ventana “Baseline”.

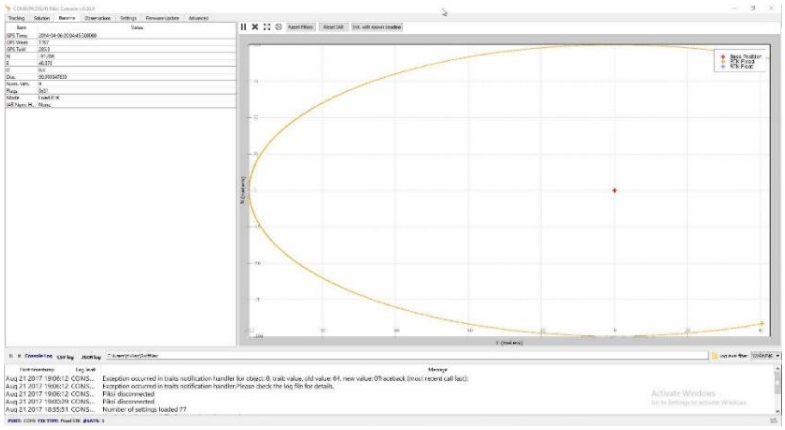


Figura 3.6: Gráfica de las soluciones diferenciales del sistema RTK-GPS entre los modos “Float” y “RTK Fixed” para la ubicación del rover.

3.2 Dispositivo UAV

El UAV requiere de una planificación de ruta, conocer la ubicación del mismo y designar la ruta a recorrer, para cumplir con estas fases se realiza la configuración del UAV a través de una herramienta de software, que posibilita un vuelo autónomo.

A continuación, se da detalle del programa que permitirá el vuelo autónomo. Además, este programa da un registro del estado del UAV y cuenta con las respectivas alertas para detener el vuelo en el caso de una emergencia.

Se realiza la instalación del programa “Mission Planner” con el cual se tiene acceso a todos los parámetros de configuración del UAV 3DR X8+ así como también posibilita preparar misiones autónomas [17]. Al ejecutar el programa para realizar la conexión con el UAV se selecciona el número de puerto asignado el cual se puede revisar mediante el administrador de dispositivos tal como muestra la Figura 3.7, la tasa de baudios que usa con cable USB es 115200, con la radio 3DR es 57600.

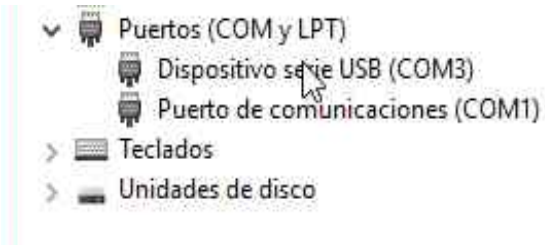


Figura 3.7: Visualización del puerto de comunicación asignado.

Luego se da click en el botón CONNET como se muestra en la Figura 3.8.

Figura 3.8: Configuración para realizar la conexión con el UAV.

La Figura 3.9 muestra el diagrama general de la BS y el rover como dispositivos finales para diseñar la red, el ambiente de cómo se ubicarán los dispositivos y el escenario que el usuario siempre tendrá el control del dispositivo y vigilará las alertas de energía o desconexión de la red. Además, se observa la conexión satelital de los Piksi para cumplir con la comunicación y el usuario.

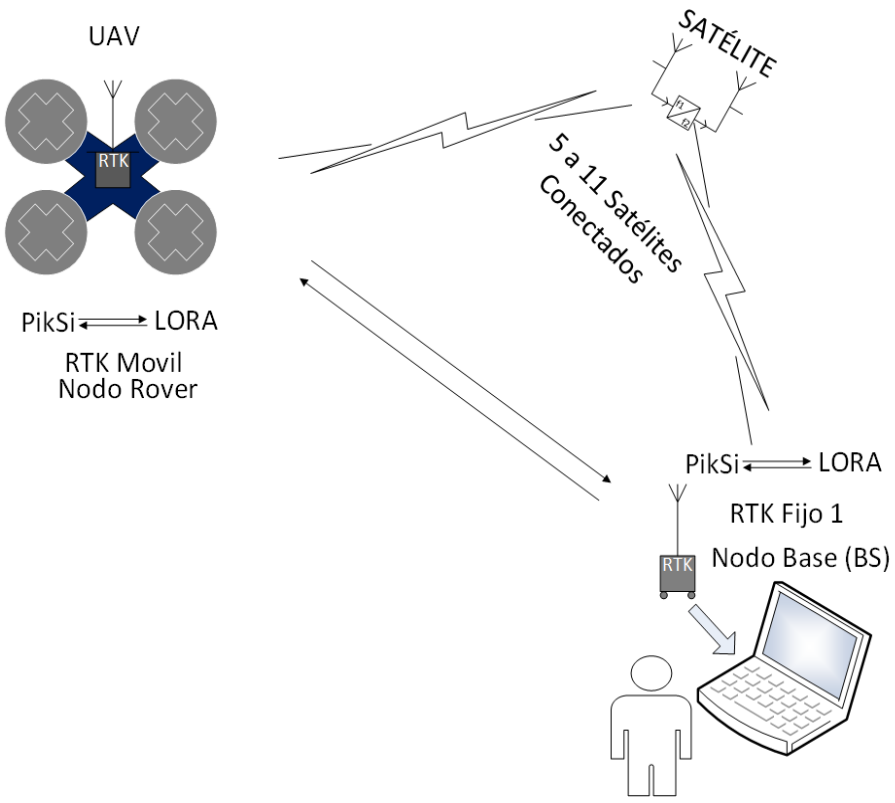


Figura 3.9: Diagrama general de la BS y el rover.

3.2.1 Integración de la cámara GoPro Hero 3

Para realizar la integración de la cámara GoPro Hero 3 se montó un sistema de estabilización de imágenes para que al realizar las fotografías estas no se vean afectadas por el movimiento del UAV. El equipo de estabilización es conocido como “Gimbal”. El empleado en este trabajo es de la empresa Tarot el cual está diseñado para trabajar con la cámara GoPro Hero 3.

Luego de la instalación en el UAV se procede con la configuración para lo cual se conecta el UAV al computador en donde se encuentra instalado el “Mission Planner” y se ejecuta el programa. Se accede a la pestaña “Initial Setup”, luego se da click en “Optional Hardware” y se busca la opción “Camera Gimbal”.

Se debe ajustar los parámetros tal como se muestra en la Figura 3.10

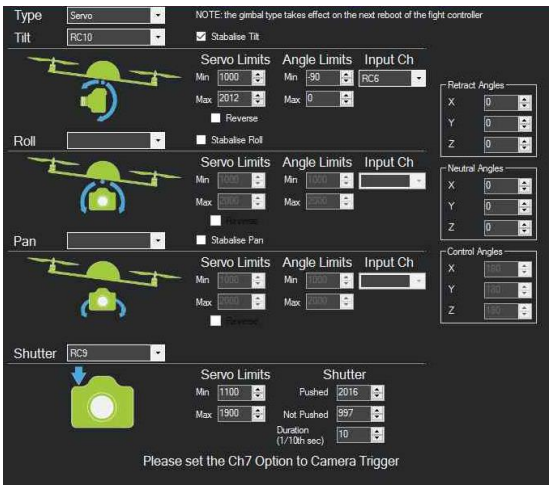


Figura 3.10 Configuración del Gimbal.

Al finalizar se debe desconectar el UAV del “Mission Planner” y luego reiniciarlo para que los cambios surtan efecto.

3.3 Integración del sistema RTK-GPS con el UAV

En esta sección se realiza la integración del sistema RTK-GPS con el UAV las configuraciones que se mostraran a continuación se las debe realizar solo una vez salvo que se indique lo contrario.

Antes de iniciar se realiza la verificación la versión del firmware instalado en el UAV, esto se observa la parte superior de la ventana del “Mission Planner”, el firmware correspondiente es APM: Copter VX.Y.Z donde VX.Y.Z indican la versión del firmware actual del UAV. Es necesario que la versión sea v3.3.1 o superior para que se pueda realizar la integración, lo mencionado anteriormente se puede observar en la Figura 3.11.

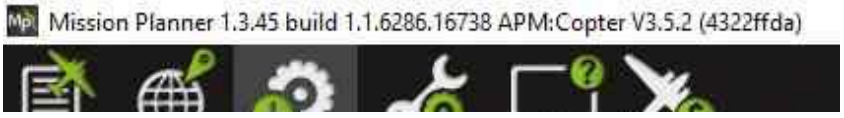


Figura 3.11: Visualización de la versión del “Mission Planner” y la versión del firmware del UAV.

3.3.1 Configuración de Hardware

La instalación del Piksi rover en el UAV se la realizó según las siguientes consideraciones:

- La antena GPS es ubicada elevada y alejándose de los circuitos del UAV y que permita línea de vista al cielo
- El Piksi debe encontrarse alejado de los circuitos electrónicos.
- El radio transmisor debe ubicarse de manera que haya línea de vista con la estación.
- El Piksi y la radio deben tener una fuente de alimentación externa al UAV.

Ya instalado el Piksi rover sobre el UAV se procede a realizar las conexiones entre ambos sistemas para esto se conecta el puerto UARTB del Piksi con el puerto Serial 4/5 del UAV tal como se observa en la Figura 3.12 mediante un cable con la configuración especificada en la Figura 3.13.

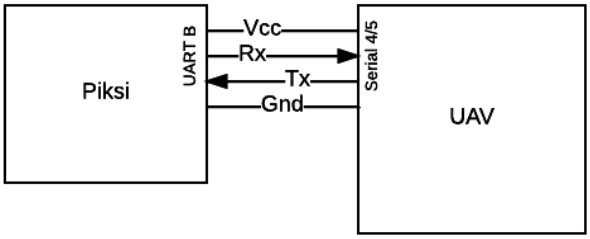


Figura 3.12: Diagrama de conexión entre el Piksi y el UAV (conexión en el puerto serial 4/5).

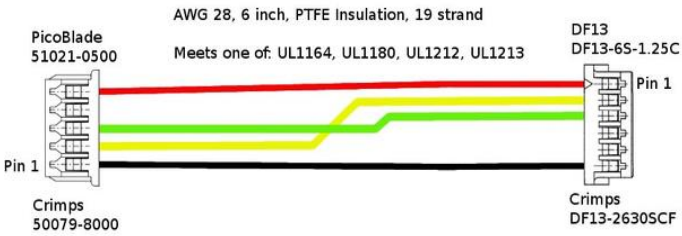


Figura 3.13: Diagrama del cable para la conexión Piksi-UAV.

El puerto UARTA se conecta al sistema LoRa por medio de un convertidor de voltaje, tal como se muestra en el diagrama de la Figura 3.14.

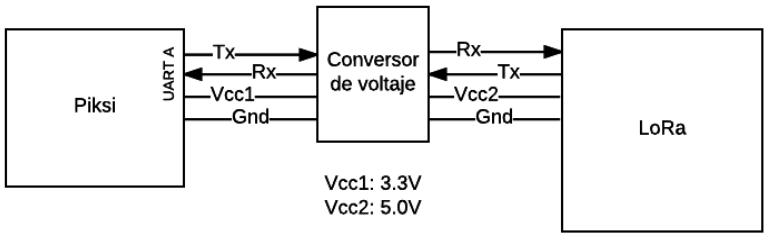


Figura 3.14: Diagrama de la conexión entre el Piksi y el LoRa.

3.3.2 Configuración de Software

En primer lugar, se realiza la configuración del UAV para esto se lo conecta al computador y ejecuta el programa “Mission Planner”. Acceder a la pestaña “CONFIG/TUNNIG” y dentro de la lista de parámetros se deben configurar los parámetros que se detallan en la Tabla 1.

Parámetro	Valor
GPS_TYPE2	**1**
SERIAL4_PROTOCOL	**5**
SERIAL4_BAUD	**115**
GPS_SBP_LOGMASK	**-1**
GPS_AUTO_SWITCH	**0**

Tabla 1. Parámetros del módulo GPS del UVA que son configurados para poder usar el RTK-GPS.

A continuación, se configura el Piksi instalado en el UAV para esto se conecta el Piksi al computador y se accede al el mediante el programa "Console", en la pestaña "Settings" se deben configurar los parámetros que se detallan en la Tabla 2.

Sección	Parámetro	Valor
UARTB	SBP message mask	65280
UARTA	Baud-rate	57600
Solution	Soln frequency	5
Solution	Output every n observation	1
Telemetry	Configuration string	AT&F, ATS1=57, ATS2=64, ATS3=50, ATS5=0, AT&W, ATZ

Tabla 2. Parámetros de la comunicación serial del UAV se configuran para la comunicación del UAV y el Piksi.

Por último, se configura el Piksi de la BS para esto se conecta el Piksi al computador y se accede al el mediante el programa "Console", en la pestaña "Settings" se deben configurar los parámetros que se detallan en la Tabla 3.

Sección	Parámetro	Valor
Surveyed Position	Latitude	Se determina en el momento de establecer la BS
Surveyed Position	Longitude	Se determina en el momento de establecer la BS
Surveyed Position	Altitude	Se determina en el momento de establecer la BS
Surveyed Position	Broadcast	Se determina en el momento de establecer la BS
Solution	Soln frequency	5hz
Solution	Output every n observation	1
SBP	Observation message max size	102
UARTA	Baud-rate	57600
Telemetry	Configuration string	AT&F, ATS1=57, ATS2=64, ATS3=50, ATS5=0, AT&W, ATZ

Tabla 3. Parámetros de posicionamiento del RTK-GPS para establecer la ubicación de la BS.

En este punto ya se cuenta con la integración del sistema RTK-GPS con el UAV y ahora se tendrá una mayor precisión en la ubicación GPS con centímetros de error.

3.4 Diseño de la red peer to peer al usar tecnología LoRa

El enlace de comunicación se realiza mediante una red P2P, la cual está compuesta por el nodo rover y la BS en tierra. En la Figura 3.15 se muestra un diagrama de bloques de la arquitectura general del hardware empleado, hay tres canales RF y un canal GPS. El primero es un enlace bidireccional de 915MHz entre el UAV y la estación terrestre portátil por medio del módulo LoRa, el segundo es el enlace bidireccional de 915MHz entre el UAV y el transceptor que

envía los datos del UAV. El tercer canal es de 2.4GHz entre el UAV y el control remoto, el último enlace del sistema RTK-GPS cuyos datos son captados por el Piksi desde los satélites conectados al mismo a una frecuencia de 1575.42 MHz.

Los Piksi en cada nodo deben establecer la comunicación mediante el módulo LoRa, cada Piksi se conecta a un módulo LoRa con las respectivas antenas para cada dispositivo. En la BS en tierra, el usuario puede acceder a información mediante una interfaz para los equipos en la BS, y para el rover accede por medio del CPU embebido.

Los dispositivos LoRa son los encargados de transmitir la ubicación de los satélites GPS captada por el Piksi de su nodo y recibir la ubicación del nodo vecino. La función de esta tecnología es crear el enlace de comunicación de RF para determinar la cobertura, y transmisión de datos del Piksi para de esta manera realizar el proceso de corrección de la ubicación RTK-GPS.

Se diseña la red con los elementos y recursos necesarios para llevar acabo la comunicación, además se considera el ambiente en donde se va desarrollar el experimento para realizar los cálculos correspondientes y determinar la infraestructura física necesaria.

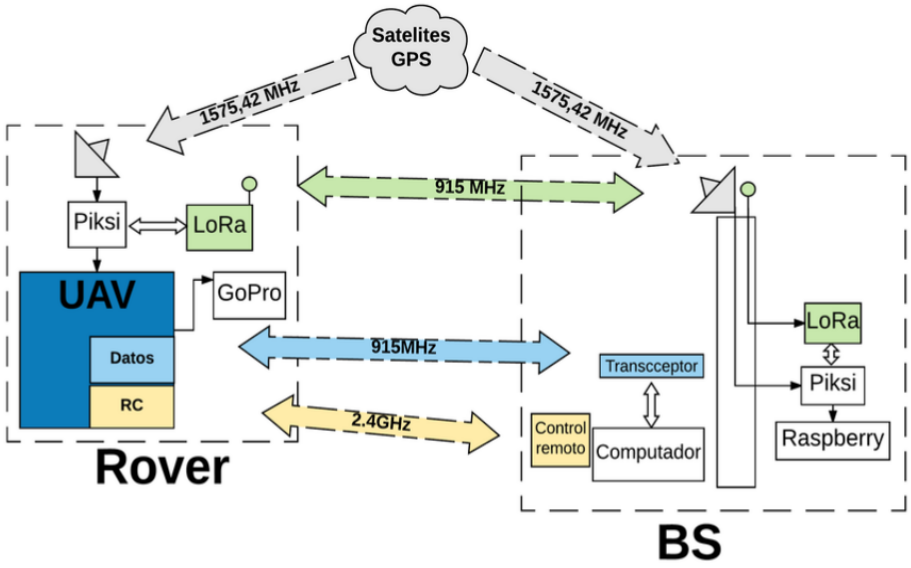


Figura 3.15: Arquitectura general de la red P2P.

3.4.1 Recursos energéticos

Para detallar los recursos energéticos utilizados para la red P2P se hace un análisis de las limitantes del hardware, consumo de energía del sistema, y tiempo de vuelo del UAV.

A continuación, los recursos energéticos para cada nodo de la red:

- **Nodo Rover.** Este nodo se usa como fuente de energía para el Piksi y LoRa una batería externa con un voltaje de salida de 5V y una capacidad de 4000mAh, con un rango operacional de –10ºC a +50ºC
- **Nodo BS.** Para este nodo se usa como fuente de energía una batería externa con un voltaje de salida de 5V y una capacidad de 30000mAh, con un rango operacional de –40ºC a +60ºC.

La utilización de estos componentes hace posible que cada nodo pueda tener una autonomía en función a la obtención de datos y en relación con el consumo de energía de los elementos se representa en la siguiente Tabla 4.

Dispositivo	Amperaje		Voltaje	Potencia
LoRa	Recepción 13.8 mA	Transmisión 28 mA	3.3V	138.6mW
Piksi	100 mA		5V	500mW
Arduino Mega	68 mA		5V	340mW

Tabla 4. Tabla de consumo de energía de los componentes para los nodos de la red P2P.

3.4.2 Elementos de la red

- **Nodo Rover.** En este nodo se encuentra el dispositivo UAV que recorrerá una ruta por aire, en el UAV se encuentra montado el LoRa conectado a una antena omnidireccional con una ganancia de 0dBi, el Piksi conectado a una antena GPS, el UAV se conecta con la estación mediante un enlace de RF, la cámara GoPro se programa para que tome fotografías cada 2 segundos. Se usa la

batería y los respectivos cables de alimentación para el LoRa, el Piksi y el UAV.

- **Nodo BS.** En este nodo se encuentra computador que permite al usuario visualizar el estado de la trayectoria y realizar las configuraciones del UAV, también hay una torre de 2.20m donde se montó el LoRa conectado a una antena de 0dBi, el Piksi conectado a una antena GPS. Se usa la batería y los respectivos cables de alimentación para el LoRa y el Piksi
- **Red P2P.** Camino que permite la transmisión de datos entre la BS y el rover para que este último pueda corregir mediante el software embebido de Piksi su ubicación. En cuanto a la BS se encarga de transmitir su ubicación la cual es conocida. La BS reciben un mensaje de control cada 30 segundos del rover para conocer si este sigue activo.
- **Mensaje.** Es aquel envió de información encriptada enviada desde la posición de los satélites GPS conectados, los nodos deben contar con satélites GPS conectados a cada uno, estos deben ser los mismo para el rover y la BS.
- **Aplicación.** El dispositivo UAV es capaz de realizar vuelos autónomos de 15 minutos y obtener fotografías para su posterior procesamiento.

3.4.3 Programación de los nodos

Los dispositivos requieren de cierta programación previa para su funcionamiento, esta programación se realiza mediante un Arduino Mega. Los diagramas de flujo en las Figuras 3.16 y 3.17 corresponden a la programación realizada para cada nodo.

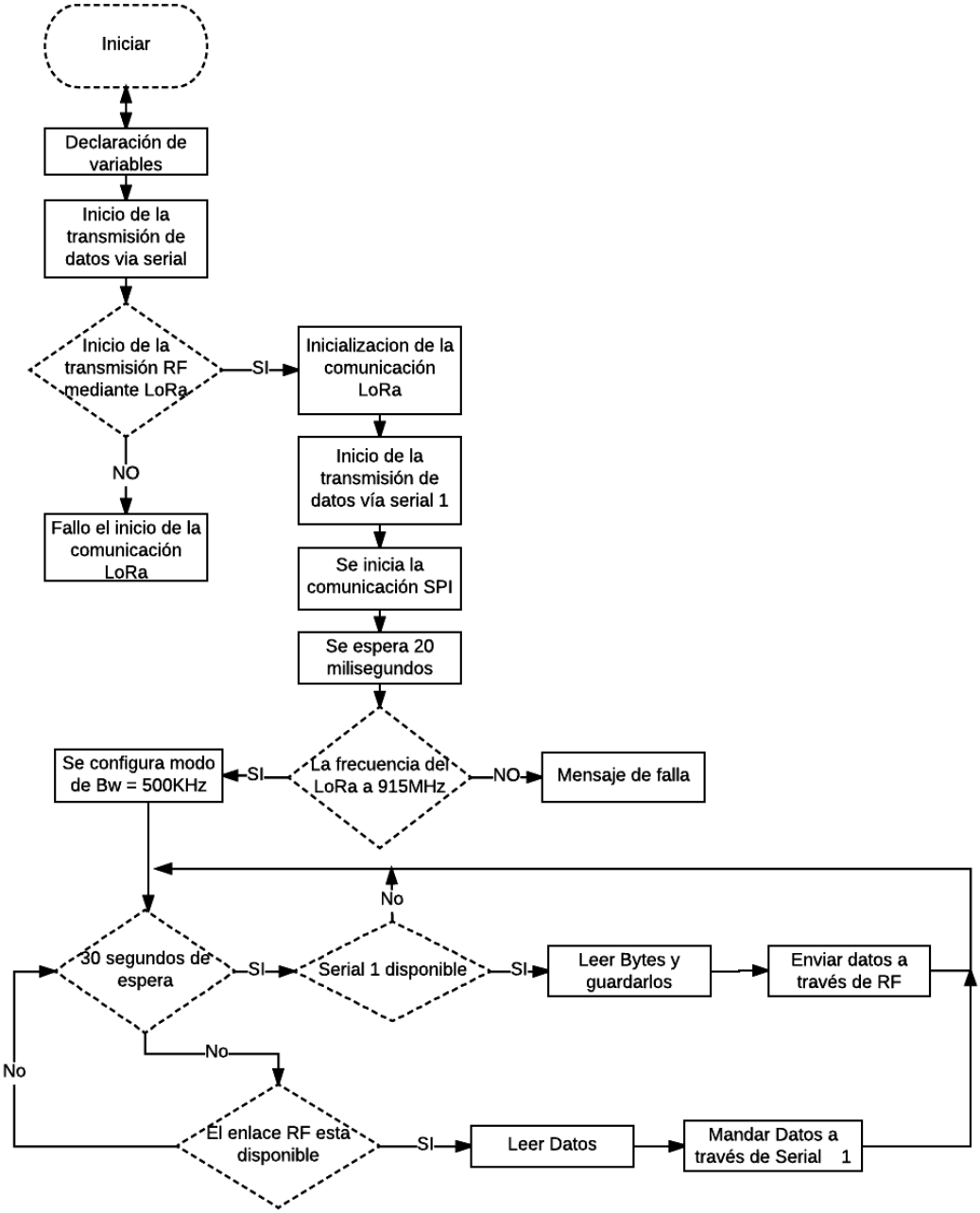


Figura 3.16: Diagrama de flujo de la programación del rover.

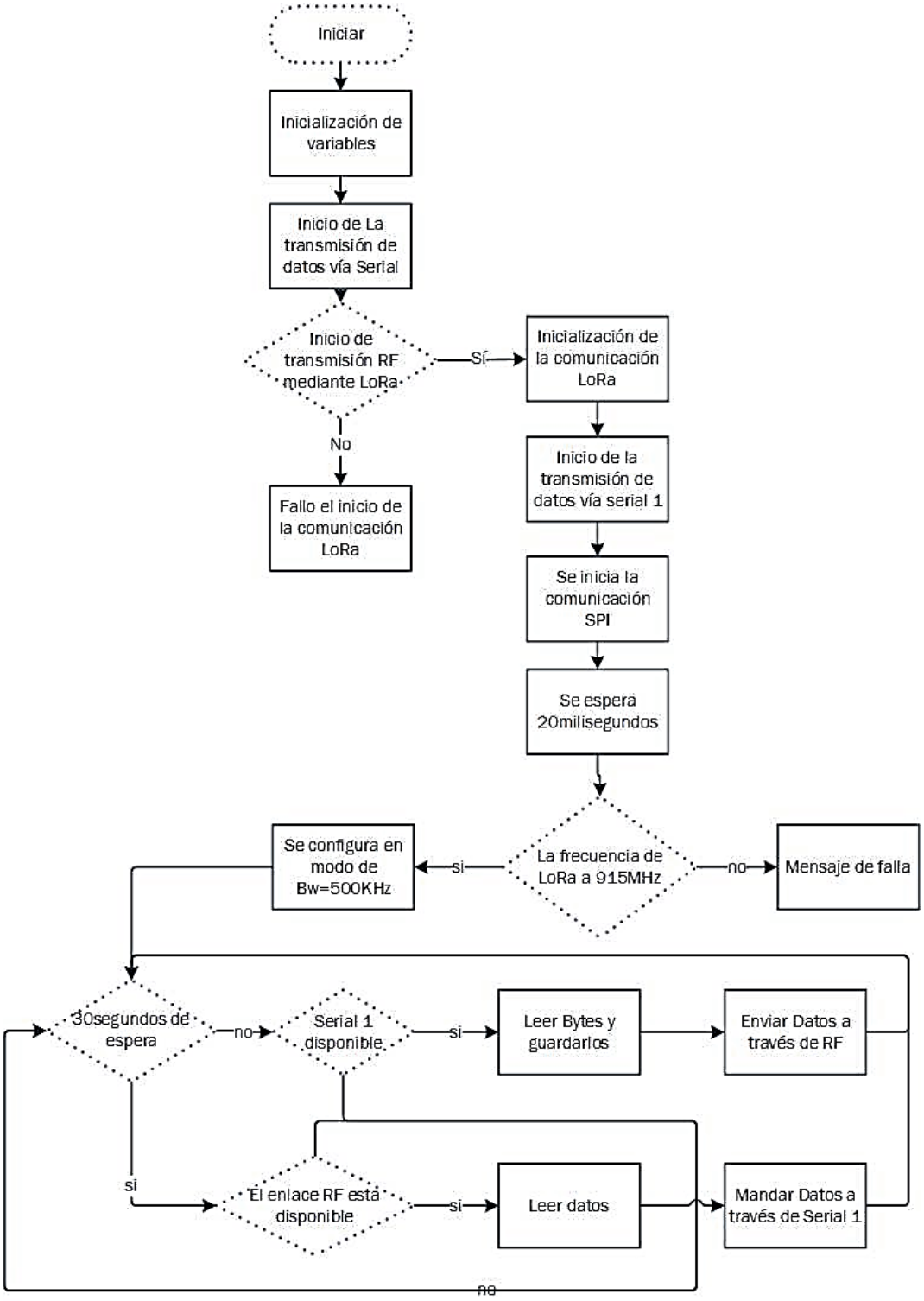


Figura 3.17: Diagrama de flujo de la programación de la BS.

Al momento de programar el LoRa es necesario considerar el ancho de banda, el factor de esparcimiento y la codificación puesto que estos tres parámetros influyen directamente en la tasa de transmisión de los datos. Un alto factor de esparcimiento permite lograr una mayor cobertura, pero se reduce drásticamente la tasa de datos. La ecuación 3.1 muestra como los parámetros mencionados influyen en la tasa de bits.

$$R_b = SF * \frac{\frac{4}{4+CR}}{\left[\frac{2^{SF}}{BW}\right]} \; bits/s \qquad (3.1)$$

Donde:

SF = factor de esparcimiento

CR = codificación

BW = ancho de banda

R_b = tasa de bits

El ancho de banda programado es de 500KHz, la codificación es 4/5 y el factor de esparcimiento es 7, con estos valores se obtiene una tasa de bits de 22.79 kbps o 2.85 kBps.

Debido a esta velocidad de envió de datos del LoRa es menor a la velocidad de comunicación del Piksi que es 4kBps, entonces la comunicación será semidúplex y se enviará cada 30 segundos un mensaje de control para identificar el rover conectado. La programación de los nodos está realizada para que LoRa envía los mensajes en el menor tiempo que es 200ms.

3.4.4 Infraestructura física de la red P2P

En primera instancia se procede a realizar el cálculo de la zona de Fresnel con el propósito de cumplir con los requerimientos de la aplicación y el experimento para obtener la aproximación al modelo de propagación al usar el módulo LoRa. La ecuación 3.2 muestra la fórmula para el cálculo de las zonas de Fresnel, se muestra a continuación.

$$r_n = \sqrt{\frac{n\lambda d_1 d_2}{d_1+d_2}} \qquad (3.2)$$

Donde:

r_n = es el radio del cráneo de Fresnel [m].

d_1 = distancia desde el transmisor al centro del elipsoide [m].

d_2 = distancia desde el receptor al centro del elipsoide [m].

λ = longitud de onda de la señal [m]

En este trabajo se hacen dos pruebas para comprobar la conectividad y cobertura. La primera prueba consiste en comunicar hasta 330m de separación entre los nodos de la red, la altura de la torre en la BS es de 2.2m y el UAV sobrevuela a 100m de altura, para llevar acabo la aplicación de captura de imágenes. La segunda prueba consiste en separar los nodos de la red P2P una distancia desde 1m hasta 28m, para obtener la aproximación al modelo de propagación y tener una elevación de los nodos que sobrepase la altura de la vegetación.

Se calcula la primera zona de Fresnel r_1, para la primera prueba (ecuación 3.3) se tiene que d_1 y d_2 es 165m y para la segunda prueba (ecuación 3.4) se tiene que d_1 y d_2 es 14m. Para el estudio de la cobertura se sugiere tener despejado un 60% de la primera zona de Fresnel.

$$r_1 = \sqrt{\frac{\frac{c(165)(165)m^2}{915MHz}}{(165+165)m}} = 5.20m(0.6) = 3.12m \qquad (3.3)$$

$$r_1 = \sqrt{\frac{\frac{c(14)(14)m^2}{915MHz}}{(14+14)m}} = 1.51m(0.6) = 0.91m \qquad (3.4)$$

Para la primera prueba se requiere en teoría dos torres de 3.12m, sin considerar la altura de la vegetación. Se realiza una simulación y planificación de vuelo de la primera prueba para comprobar el enlace. En la segunda prueba se cumple con los requisitos puesto que en este trabajo se emplea una torre de 2.2m en la BS y al UAV se lo eleva hasta una

altura de 11m aproximadamente por la altura de la vegetación. Para la efectuar las pruebas requiere de una planificación previa de los puntos de partida y máximo recorrido del UAV, que se explica a continuación (sección 3.3.5).

3.4.5 Planificación de vuelo del UAV

En la primera prueba se realiza la planificación de vuelo para UAV con los equipos ya instalados, para la segunda prueba no requiere realizar una planificación de vuelo previa. A continuación, se muestra la planificación de la primera prueba.

Se selecciona como punto de estudio una albarrada cuya ubicación se puede observar en el mapa que se muestra en la Figura 3.18.

Figura 3.18: Ubicación de la albarrada.

A continuación, se procede a seleccionar la ubicación en donde se instalará la BS, el punto seleccionado se muestra en la Figura 3.19.

Figura 3.19: Ubicación de la BS.

En el mapa de la Figura 3.20 se observa el perfil de elevación entre la BS y el UAV.

Figura 3.20: Vista del perfil de elevación del terreno.

Con la información previa se procede a realizar la simulación del enlace de comunicación, la Figura 3.21 muestra la disposición de la BS y el UAV en el lugar seleccionado para la prueba, la separación entre ambos nodos es de 330m, la Figura 3.22 muestran el resultado de la simulación del enlace de comunicación entre la BS y el rover, donde se puede observar el perfil del radio enlace, los valores de las perdidas relacionadas al entorno.

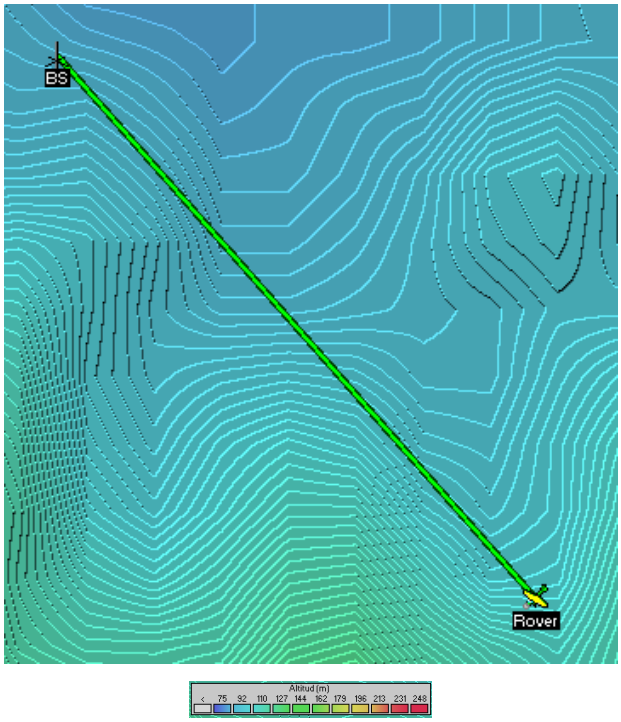


Figura 3.21: **Enlace de comunicación entre la BS y el rover.**

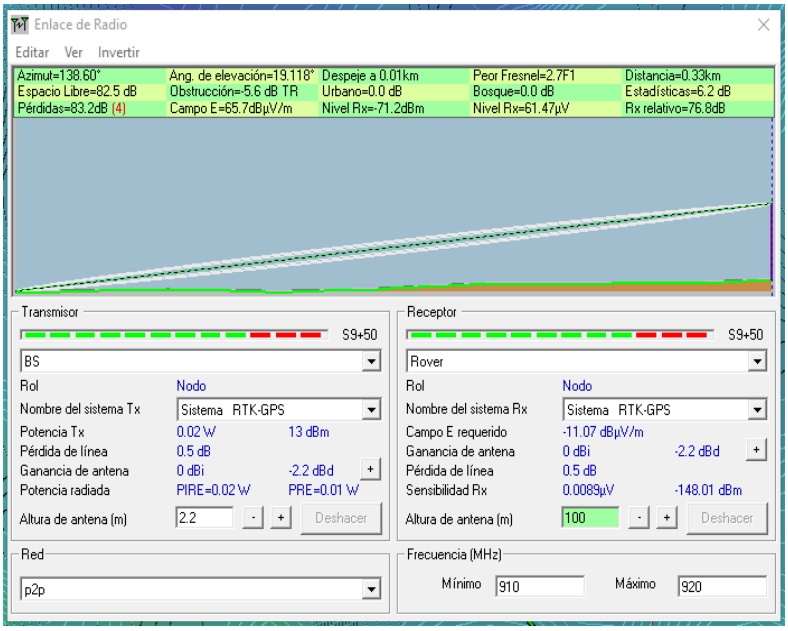


Figura 3.22: Resultados de la simulación enlace de comunicación entre la BS y el rover a 330m.

La segunda prueba consiste en elevar el UAV a una altura de 2.2m y aumentar progresivamente la distancia entre la BS y el UAV (sección 3.3.4), la Figura 3.23 muestra el esquema de esta prueba.

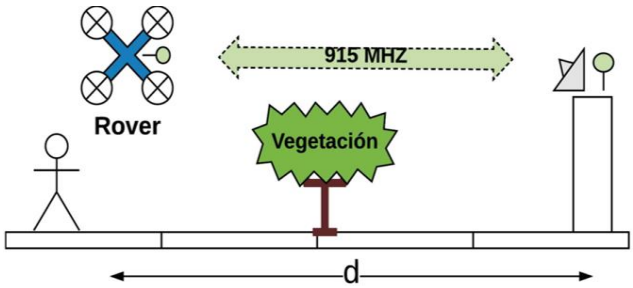


Figura 3.23: Esquema de comunicación de la segunda prueba.

3.5 Sistema embebido para la captura de datos

Para almacenar la información generada por la BS se hace uso del CPU embebido, esto se logra por medio de la programación realizada en el CPU embebido para capturar los datos del Piksi. El proceso que se realiza es adquirir los paquetes enviados por el Piksi y seleccionar aquellos que contengan los mensajes deseados para ser almacenados, como es el caso de las coordenadas de latitud, longitud y altura de la BS, en un archivo de valores separados por coma (csv) lo cual facilita el análisis posterior de los datos. En la Figura 3.24 se muestra un diagrama de flujo general del programa para la captura de los datos del Piksi.

Para ejecutar el programa se accede mediante el terminal de Raspbian al directorio en donde se encuentra el archivo ejecutable y se escribe la siguiente instrucción: ./piksidatalogger -p /dev/ttyUSB0

Para finalizar la ejecución del programa se presiona la combinación de teclas “Ctrl + c”.

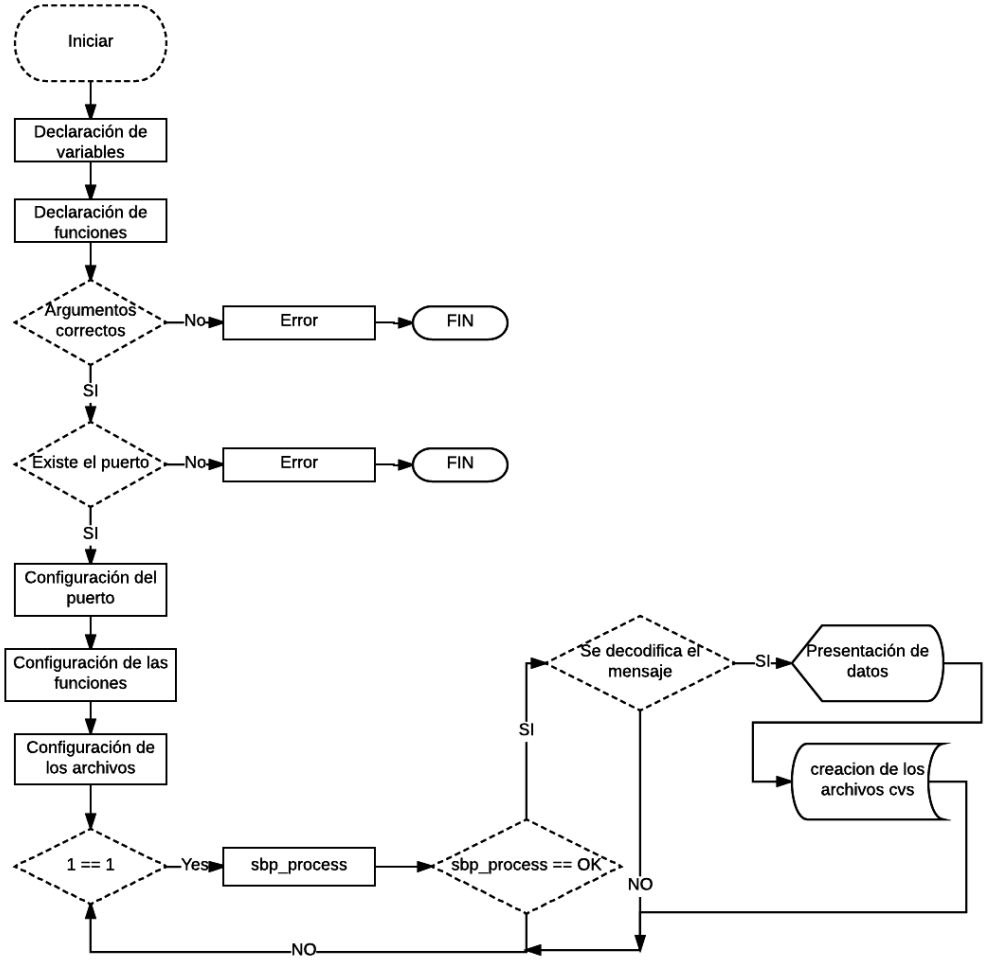


Figura 3.24: Diagrama de flujo de la programación para adquirir los datos del Piksi.

CAPÍTULO 4

4. RESULTADOS

En este capítulo se muestran los resultados de la evaluación de la red P2P, luego se muestran las imágenes obtenidas con el dispositivo UAV y parte del proceso de procesamiento para geo-etiquetar las imágenes. Finalmente, se presenta el análisis del entorno de densa vegetación, de esta manera se obtiene un modelo de propagación acorde a este estudio y ambiente.

4.1 Evaluación de los resultados de la red peer to peer (LoRa - RTK-GPS)

La prueba consistió en la supervisión de los datos enviados al rover y la cobertura de comunicación hasta 330m de separación entre los nodos de la red, la altura de la torre BS de 2.2m y el rover una altura de 100m en un entorno con vegetación de 10m de altura en promedio aproximadamente, esta prueba fue simulada como se muestra a detalle en la sección 3.5.2.

Para llevar a cabo la prueba se debe iniciar con el enlace de la comunicación de los nodos de la red P2P en tierra, y los dispositivos deben estar cercanos para observar si las coordenadas que marca el Piksi son correctas.

Se realiza la conexión mediante el computador al nodo rover, en la pestaña de “Observations” del programa “Console”, Figura 4.1, se puede observar el intercambio de información entre los Piksi, en donde se visualiza como local y la BS como remoto, por cuestiones de conexión directa.

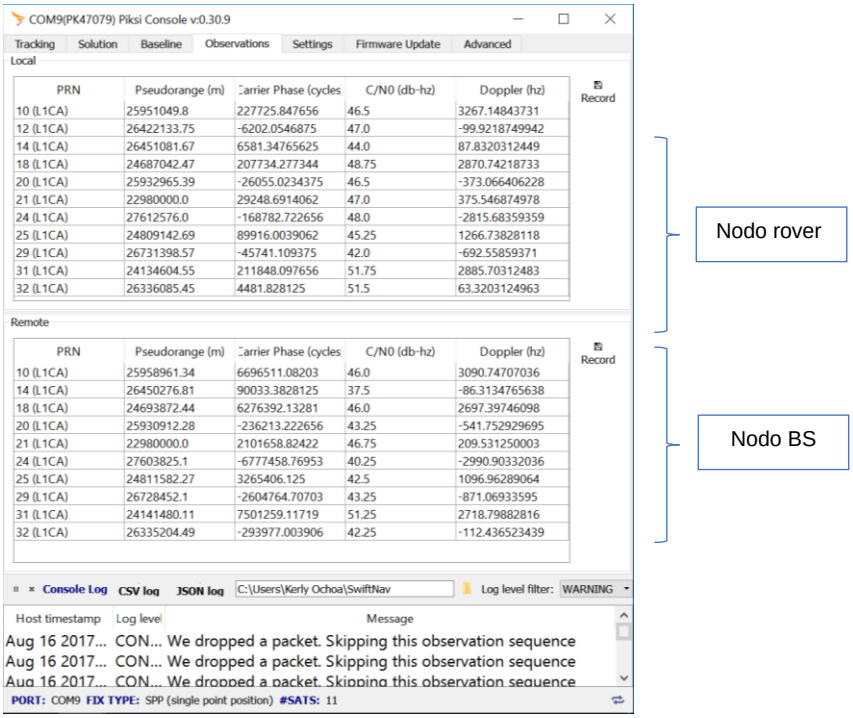
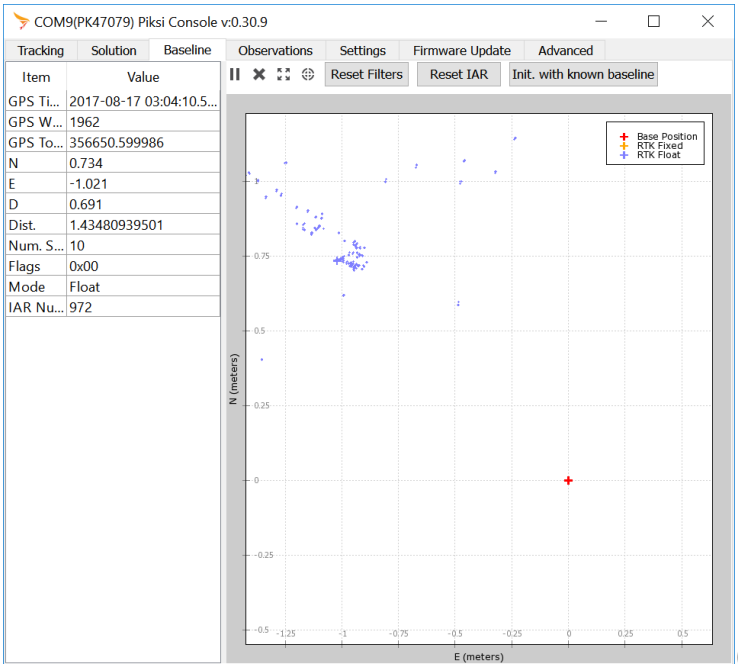
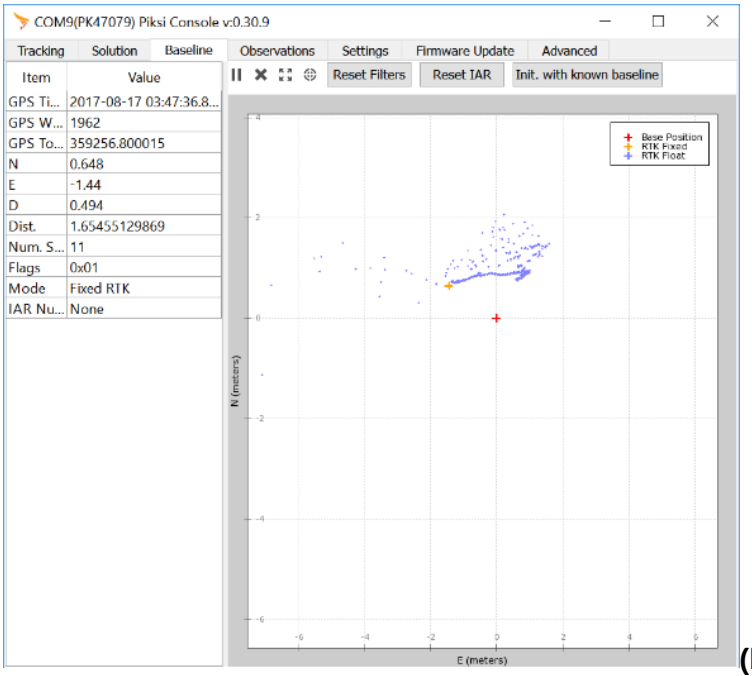


Figura 4.1: Información de los satélites observados por ambos Piksi.

En la Figura 4.2 se visualiza la ubicación rover, la Figura 4.2.a observa una cruz de color azul indica que la solución se encuentra en modo flotante (Float) y la Figura 4.2.b cuando se logra la una solución fija RTK, que es la esperada y que posee un error menor a 10cm en la ubicación, la cruz cambia a un color naranja, y la BS está representada por la cruz de color rojo. Se observa en esta imagen el número de satélites conectados, el modo de la solución, la distancia entre la BS y el rover y la posición GPS.



(a)



(b)

Figura 4.2: La Figura 4.2.a muestra la nube de puntos en color azul correspondientes a la solución Float y en la Figura 4.2.b se observa una cruz color amarillo la cual indica que se ha obtenido una solución Fixed RTK. La cruz color rojo indica la ubicación de la estación base.

En la Figura 4.3 se puede observar que se muestran las características de la comunicación del puerto UARTA, el número de errores CRC de la trama del Piksi es elevado debido a la velocidad de envió de datos para el Piksi es 4kBps y la del módulo LoRa es de 2.85kBps. El Piksi está configurado por defecto para rechazar la información que no es necesaria, además, se muestra la velocidad de envió de datos y el porcentaje de buffer disponible.

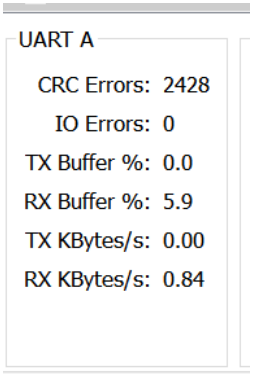


Figura 4.3: Valores de la comunicación serial del UARTA del Piksi en la BS.

Los nodos de la red P2P establecen comunicación con éxito, entonces, se procede a planear la ruta del UAV para realizar el vuelo autónomo y permitir que el UAV desempeñe la aplicación deseada.

4.2 Imágenes capturadas por el dispositivo UAV

La planificación de la ruta del vuelo autónomo se realizó en el programa “Mission Planner” y posteriormente se cargó la información de la ruta al UAV, el resultado de la planificación se puede observar en la Figura 4.4.

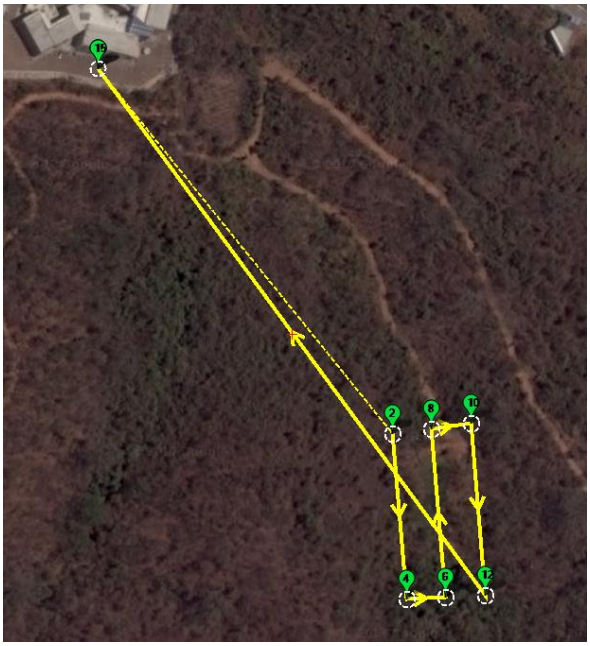


Figura 4.4: Planificación de la zona a sobrevolar.

La cámara se programó para que realice fotos cada 2 segundos da como resultado un conjunto de datos de 250 imágenes, con esto se procede a procesar las imágenes obtenidas para añadir la información geográfica a los metadatos de cada una de ellas. La Figura 4.5 muestra la interfaz del programa “GeoSetter” [18], el cual es un programa diseñado para añadir la información geográfica a imágenes, los puntos de color azul en la parte superior derecha son las ubicaciones en donde se realizaron las fotografías y su etiqueta geográfica se observa en la parte superior izquierda, para realizar esto se emplea el log de vuelo del UAV. En la parte inferior izquierda se observa una fotografía de la albarrada.

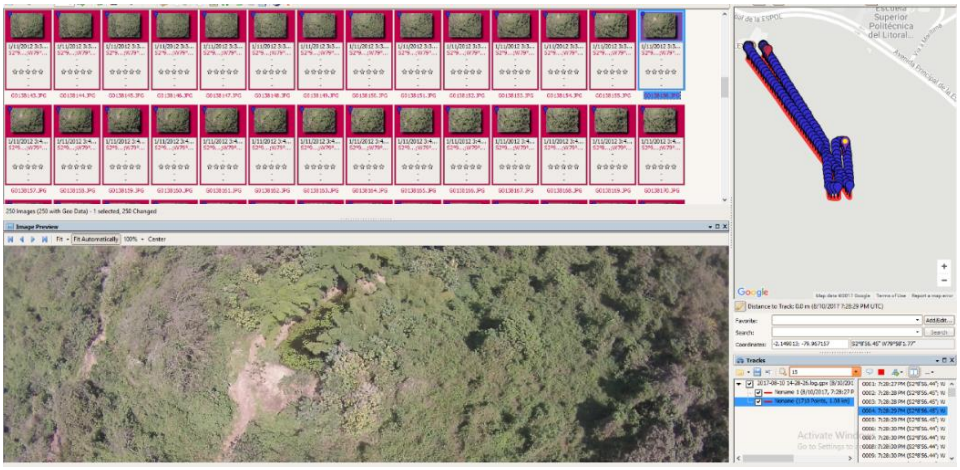


Figura 4.5: Geo-etiquetado por medio del software GeoSetter de las fotografías capturadas con el UAV.

Al guardar los resultados de GeoSetter se obtiene un conjunto de datos de 250 imágenes geo-etiquetadas, para leer los metadatos de las imágenes y verificar que se añadió correctamente la información del punto geográfico se emplea el programa “ExifTool” el cual es un programa que entre sus muchas funciones permite visualizar los metadatos de las imágenes. A continuación, se muestran algunas de las imágenes obtenidas durante el vuelo en las Figuras 4.6, 4.7.

Figura 4.6: Imagen “G0138154.JPG” con su respectiva información geográfica.

Figura 4.7: Imagen "G0138158.JPG" con su respectiva información geográfica.

4.3 Aproximación de un modelo de propagación para entornos vegetativos

La mayoría de las herramientas de predicción de propagación automatizada para el análisis de la cobertura sobre bases de datos geométricas utiliza modelos empíricos que se describen mediante curvas derivadas o ecuaciones, el análisis estadístico de datos medidos, que corresponde a simples pruebas y que no requieren detalles del terreno, pero fácil de aplicar, sin embargo, no se tiene una estimación precisa, en comparación con modelos empíricos con correcciones y con resultados que confirmen la consistencia del método [19, 20]. En este trabajo se sugiere este modelo empírico como base para futuras estudios en ambientes de densa vegetación.

Para el estudio se realizó el experimento donde se logra el enlace de comunicación con los nodos de la red P2P. Se tiene como variable la distancia de separación entre los nodos, y como resultado se obtiene un archivo donde se registra la potencia de cada medición, este archivo es generado por el CPU embebido. Las mediciones fueron realizadas en un ambiente de densa vegetación en intervalos de 30 muestras por cada metro. Este experimento se realizó hasta una distancia de 28m entre nodos de la red P2P, se cuenta con la altura de la BS de 2.2m y se eleva al UAV a una altura de 11m, se capturaron los datos siempre que haya comunicación entre los nodos, fueron transmitidos 252 bytes de manera recurrente. Debido a que el objetivo es predecir la cobertura se

utilizó una altura del UAV que solo sobrepase la altura de la vegetación que es de 10m aproximadamente.

Metros [m]	P_{RX} [dBm]	$\widehat{P_{RX}}$[dBm]	Error	Desviación Estándar
1	-84	-86.36	0.0273275	0
2	-88.77	-88.80	0.0004241	0.25
3	-96	-90.23	0.0638979	0.20
4	-97.43	-91.25	0.0677771	0.25
5	-100.77	-92.04	0.0948694	0.71
6	-107.93	-92.68	0.1645948	-1.60
7	-106.67	-93.22	0.1442232	0.42
8	-105.17	-93.69	0.1224627	0.14
9	-103.87	-94.11	0.1036947	0.18
10	-105.20	-94.48	0.1134632	0.16
11	-105.10	-94.82	0.1084614	0.22
12	-106.23	-95.12	0.1167967	0.25
13	-105.90	-95.41	0.1100022	0.16
14	-106.30	-95.67	0.1111511	0.21
15	-107.10	-95.91	0.1166735	0.09
16	-108.03	-96.14	0.1237348	0.03
17	-106.10	-96.35	0.1011793	0.09
18	-109.23	-96.55	0.131329	0.51
19	-109.07	-96.74	0.1273835	0.53
20	-110.53	-96.92	0.1404047	0.72
21	-111.20	-97.01	0.1452533	0.16
22	-112.87	-97.26	0.1604612	0.45
23	-112.233	-97.4172	0.1520857	0.18
24	-112.033	-97.5673	0.1482636	0.50
25	-112.433	-97.7113	0.1506656	0.31
26	-113.9	-97.8496	0.1640315	0.22
27	-112.333	-97.9827	0.1464578	0.36
28	-114.033	-98.1109	0.1622865	0.23

Tabla 5. Se muestran los valores de la $\widehat{P_{RX}}$, la potencia experimental P_{RX}, el error de los datos y la desviación estándar a diferentes distancias.

El análisis estadístico se realizó con ayuda de la herramienta R, el cual permitió calcular la potencia recibida estimada $\widehat{P_{RX}}$ que se obtiene al usar el modelo logarítmico y la potencia experimental P_{RX} obtenida de la segunda prueba, el error de los datos y la desviación estándar tal y como se muestra en la Tabla 5. Además

de un análisis más profundo para encontrar la función a la cual se asemeja la gráfica obtenida con las mediciones del experimento.

4.3.1 Análisis de regresión lineal

A partir de los datos obtenido de la segunda prueba se analiza la variable de potencia y la variable de distancia a través de una ecuación que aproxime la curva de predicción de los datos futuros. Esta curva de predicción se realiza mediante un análisis de regresión lineal y encontrando un modelo que mejor se aproxime a los datos obtenidos (Tabla 5.). Con el conjunto de puntos conocidos en el diagrama se analiza que modelo es el más adecuado para usar conforme a los datos, lo cual da como resultado una relación logarítmica determinada por la dependencia funcional entre las dos variables que se ajustan a la distribución bidimensional. Para el uso de un modelo se extraen los coeficientes numéricos con estimación logarítmica.

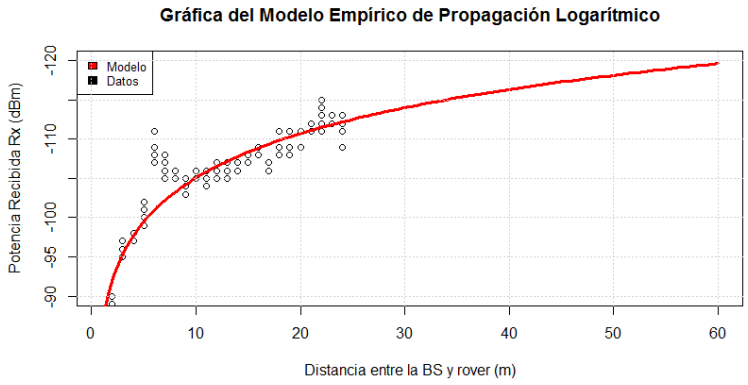


Figura 4.8: Gráfica del modelo empírico de propagación obtenido con la regresión lineal del modelo logarítmico de la potencia recibida vs. distancia.

En la Figura 4.8 se graficó las 30 muestras para cada distancia, y la gráfica con regresión lineal del modelo logarítmico. Se observa que los puntos de la potencia recibida versus la distancia en metros se agrupan bastante bien a la curva logarítmica.

La Figura 4.9 se visualiza en escala logarítmica la distancia entre el emisor y el receptor, se observa que como modelo empírico para la curva de predicción llega a un máximo a 1.5km con una potencia –145dBm.

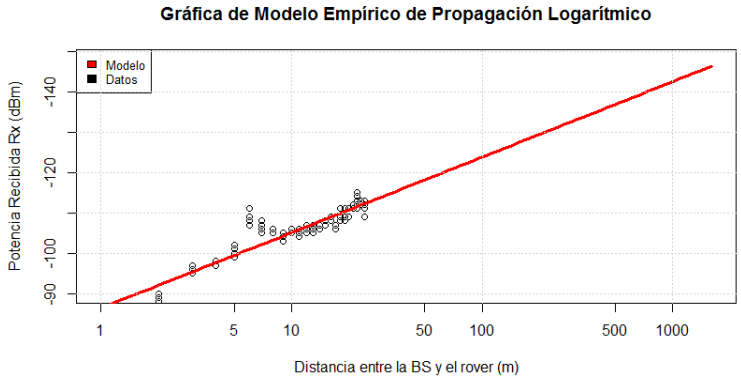


Figura 4.9: Gráfica de modelo empírico de propagación logarítmico en escala logarítmica potencia recibida vs. distancia.

4.3.2 Cálculos matemáticos para el modelo de propagación

Para determinar la ecuación de la curva que mejor se ajusta a los datos, se parte de la ecuación 4.1 de potencia recibida y con una potencia inicial de transmisión de 13dBm por defecto en la programación del módulo LoRa. Posterior se obtiene los datos de la curva logarítmica en la herramienta R (Figura 4.9), este programa da la facilidad de obtener el valor de la pendiente a y la intersección de la curva b'. Como resultado se observa la ecuación 4.2, para las perdidas está basada en la suma adicional de las perdidas, ecuación 4.3, donde la variable para la ecuación final es la distancia y esta ecuación es válida mientras la distancia sea mayor a 1m.

$$P_{RX} = P_T - L_P \qquad (4.1)$$

$$L_p = b + a\ log(d) \qquad (4.2)$$

$$a = 8.12$$

$$b' = P_T - b - a\ log(1m) = -86.36\text{dBm}$$

$$b = 99.36$$

$$P_{RX} = P_T - 99.36 - 8.12\log(d)\,,\ d > 1\ m \quad (4.3)$$

A continuación, se muestran las imágenes obtenidas al usar la herramienta R, como se observa brinda la facilidad de implementar un modelo con todas sus características y conforme a los datos permite obtener los coeficientes para la ecuación de las perdidas en el ambiente con densa vegetación con un error de 0.25 y 0.10 para $a\ y\ b'$ respectivamente que son los datos al usar el modelo logarítmico. En la Figura 4.10 se visualiza la interfaz del usuario para obtener la información y realizar la programación efectuar el modelo con los datos obtenidos del experimento.

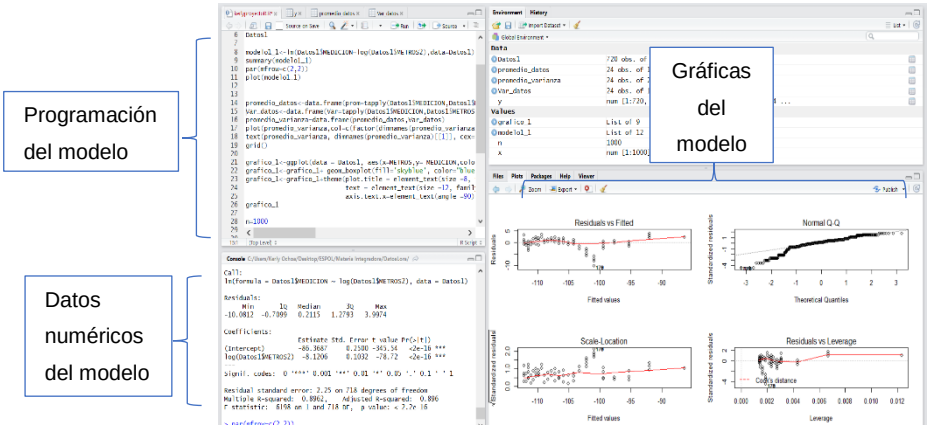


Figura 4.10: Interfaz de usuario R con la aplicación de los datos al modelo logarítmico.

CONCLUSIONES Y RECOMENDACIONES

Se diseñó e implemento un sistema que extiende la cobertura del sistema RTK-GPS y se logró aplicar la geolocalización en un entorno con vegetación densa.

Se realizó rutas más precisas con el sistema RTK-GPS y debido a esto se obtiene una mayor precisión al momento de geo-etiquetar las fotos realizadas durante el vuelo del UAV.

Se incrementó el alcance del sistema RTK-GPS al hacer uso de la tecnología LoRa de 200m a 330m en un ambiente con densa vegetación y se probó que el sistema es funcional para este proyecto.

Por medio del CPU embebido se capturaron los datos de potencia y se logró obtener un modelo empírico de propagación en un entorno de densa vegetación. Según el modelo encontrado se llega teóricamente a un alcance de hasta 1.5km.

El incremento de cobertura trae consigo una menor tasa de transmisión de bits debido a las características de modulación del LoRa.

La comunicación de RF de los LoRa es semidúplex y no puede ser dúplex debido a que tasa de transmisión de datos del LoRa es 2.85kBps y el mensaje de espera de 200ms mientras el Piksi tiene una tasa de transmisión de datos de 4kBps.

La comunicación de RF de los nodos es adecuada para la aplicación requerida, pero tiene como desventaja la baja tasa de transmisión datos, para futuros trabajos se recomienda cambiar el tipo de red donde no se requiera una comunicación dúplex.

Se recomienda que, para tener una mejor precisión se haga el diseño de la red con dos BS como referencia.

Para mejorar la cobertura se recomienda aumentar la ganancia las antenas, además se sugiere ubicar las antenas donde haya mejor recepción de acuerdo con las características de cada una.

Se recomienda incorporar el uso de la herramienta de programación R para el análisis estadístico. Esta herramienta es de utilidad para obtener características de modelos estadísticos y aplicarlo en experimentos con mediciones de datos.

BIBLIOGRAFÍA

[1] Takasu, T., & Yasuda, A., Evaluation of RTK-GPS performance with low-cost single-frequency GPS receivers. In *Proceedings of international symposium on GPS/GNSS*, pp. 852-861, noviembre 2008.

[2] Elliott D. Kaplan y Christopher J. Hegarty, *Understanding GPS: principles and applications*, 2nd. ed, ISBN 1-58053-894-0, 2006

[3] Ohta, Y., Kobayashi, T., Tsushima, H., Miura, S., Hino, R., Takasu, T., & Sato, T., Quasi real-time fault model estimation for near-field tsunami forecasting based on RTK-GPS analysis: Application to the 2011 Tohoku-Oki earthquake (Mw 9.0). *Journal of Geophysical Research: Solid Earth*, *117*(B2), 2012.

[4] Swift Navigation, "Piksi datasheet", mayo 2017. [online]. Disponible en: http://docs.swiftnav.com/pdfs/piksi_datasheet_v2.3.1.pdf.

[5] Swift Navigation, "Swift Navigation Binary Protocol", mayo 2017. [online]. Disponible en: http://docs.swiftnav.com/wiki/SBP.

[6] Beard, R. W., Kingston, D. B., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., & Goodrich, M. A., Autonomous Vehicle Technologies for Small Fixed-Wing UAVs. *JACIC*, *2*(1), 92-108, 2005.

[7] Gordon, J. L. M., Salazar, J. E. G., Vinueza, W. A. Z., Vera, R. G. B., & Montenegro, F. M., Navegación pre-programada de trayectorias de un Vehículo Aéreo no Tripulado (UAV) aplicado a la supervisión y transmisión en línea de la calidad del aire. *Revista Publicando*, *3*(9), pp. 61-80, 2017.

[8] Bortoff, S. A., Path planning for UAVs. In *American Control Conference, 2000. Proceedings of the 2000*, Vol. 1, No. 6, pp. 364-368, IEEE, septiembre 2000.

[9] Pérez García, R., "Avaluació de LoRa/LoRaWAN per a escenaris de smart city", Bachelor's thesis, Universitat Politècnica de Catalunya, 2017.

[10] Munca, R., & Daniel, J. "Dispositivo LoRa de comunicación a largo alcance y bajo consumo energético para aplicaciones del ámbito del desarrollo". Doctoral dissertation, CITDH, 2017.

[11] Córdoba Peñalver, E. J. "Análisis y diseño de una red de sensores en un Parque Natural", Tesis de pregrado, 2017.

[12] REDA, Haftu Tasew, et al. On the Application of IoT: Meteorological Information Display System Based on LoRa Wireless Communication. *IETE Technical Review*, pp. 1-10, 2017.

[13] López, J. E. G., Chavez, J. C., & Sánchez, A. K. J., Modelado de una red de sensores y actuadores inalámbrica para aplicaciones en agricultura de precisión. In *Humanitarian Technology Conference (MHTC), IEEE Mexican,* pp. 109-116, IEEE, marzo 2017.

[14] Plan Nacional de frecuencias (2012). Arcotel. Ecuador [online]. Disponible en: http://www.arcotel.gob.ec/wp-content/uploads/downloads/2013/07/plan_nacional_frecuencias_2012.pdf

[15] Reglamento de radiocomunicaciones (2012). Unión Internacional de Telecomunicaciones [online]. Disponible en: https://www.itu.int/dms_pub/itu-s/oth/02/02/S02020000244501PDFS.pdf

[16] Olasupo, T. O., Otero, C. E., Olasupo, K. O., & Kostanic, I., Empirical path loss models for wireless sensor network deployments in short and tall natural grass environments. *IEEE Transactions on Antennas and Propagation*, *64*(9), pp. 4012-4021, 2016.

[17] Bandyopadhyay, S., Giannella, C., Maulik, U., Kargupta, H., Liu, K., & Datta, S. (2006). Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, *176*(14), 1952-1985.

[18] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, 2nd. Ed. MIT Press, 2017.

[19] Mission Planner Home, "Mission Planner documentation", mayo 2017. [online]. Disponible en: http://ardupilot.org/planner/.

[20] GeoSetter for showing and changing geo data and other metadata of images files. Mayo 2017. [Online]. Disponible en: http://www.geosetter.de/en

[21] Tunc, C. A., Altintas, A., & Erturk, V. B. (2005). Examination of existent propagation models over large inhomogeneous terrain profiles using fast integral equation solution. *IEEE Transactions on Antennas and Propagation*, *53*(9), 3080-3083.

[22] Zhu, H. J., Wu, H. R., Zhang, L. H., & Miao, Y. S., The Irregularity Propagation Characteristics of Radio Signals For Wireless Sensor Network In Farmland. In *ITM Web of Conferences*, Vol. 7, p. 01013, EDP Sciences, 2016.

ANEXOS

CARACTERITICAS DE DISPOSITIVOS

Piksi versión 2.3.1

swift NAVIGATION

Piksi Datasheet

Flexible, high-performance GPS receiver platform running open-source software

Features

- Centimeter-accurate relative positioning (Carrier phase RTK)
- 10 Hz position/velocity/time solutions
- Open-source software and board design
- Low power consumption - 500mW typical
- Small form factor - 53x53mm
- USB and dual UART connectivity
- External antenna input
- Full-rate raw sample pass-through over USB

Applications

- Autonomous Vehicle Guidance
- GPS/GNSS Research
- Surveying Systems
- Precision Agriculture
- Unmanned Aerial Vehicles
- Robotics
- Space Applications

Figure 1: Piksi front and back view

Overview

Piksi™ is a low-cost, high-performance GPS receiver with Real Time Kinematics (RTK) functionality for centimeter-level relative positioning accuracy.

Its small form factor, fast position solution update rate and low power consumption make Piksi ideal for integration into autonomous vehicles and portable surveying equipment.

Piksi's open source firmware allows it to be easily customized to the particular demands of end users' applications, easing system integration and reducing host system overhead.

In addition, Piksi's use of the same open source GNSS libraries as Peregrine, Swift Navigation's GNSS post-processing software, make the combination of the two a powerful toolset for GNSS research, experimentation and prototyping at every level from raw samples to position solutions.

With these tools, developers can quickly move from prototyping software on a desktop to running it standalone on the Piksi hardware.

A high-performance DSP on-board and our flexible Swift-NAP correlation accelerator provide Piksi with ample computing resources with which advanced receiver techniques, such as multipath mitigation, spoofing detection and carrier phase tracking can be implemented.

System Architecture

The Piksi receiver architecture consists of three main components. The RF front-end downconverts and digitizes the radio frequency signal from the antenna. The digitized signal is passed into the SwiftNAP which performs basic filtering and correlation operations on the signal stream. The SwiftNAP is controlled by a microcontroller which programs the correlation operations, collects the results and processes them all the way to position/velocity/time (PVT) solutions.

Front-end

The RF front-end consists of a Maxim MAX2769 integrated down-converter and 3-bit analog-to-digital converter operating at 16.368 MS/s. This front-end is capable of covering the L1 GPS signal bands.

SwiftNAP

The SwiftNAP consists of a Xilinx Spartan-6 FPGA that comes pre-programmed with Swift Navigation's SwiftNAP firmware. The SwiftNAP contains correlators specialized for satellite signal tracking and acquisition. The correlators are flexible and fully programmable via a high-speed SPI register interface and are used as simple building blocks for implementing tracking loops and acquisition algorithms on the microcontroller.

While the SwiftNAP HDL is not open-source at this time, the Piksi has no restrictions against loading one's own firmware onto the on-board Spartan-6 FPGA.

Microcontroller

The on-board microcontroller is a STM32F4 with an ARM Cortex-M4 DSP core running at up to 168 MHz. This powerful processor performs all functions above the correlator level including tracking loop filters, acquisition management and navigation processing and is able to calculate PVT solutions at over 10 Hz in our default software configuration. All software running on the microcontroller is supplied open-source.

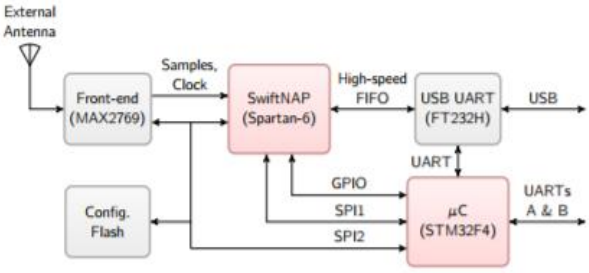


Figure 2: Piksi Block Diagram

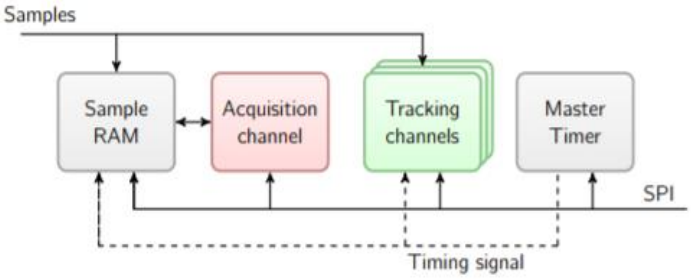
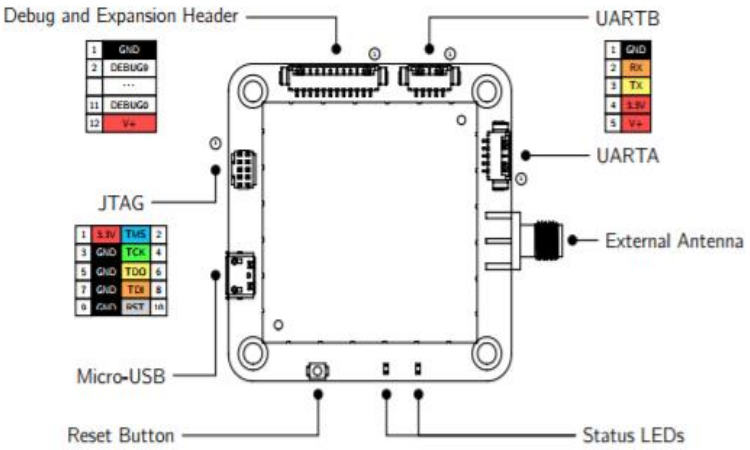


Figure 3: SwiftNAP Block Diagram

Connections



USB

A Micro-USB socket provides USB connectivity to the host. This can be configured as a USB-Serial bridge to the microcontroller (the default) or as a high-speed FIFO interface to the SwiftNAP for streaming full-rate raw IF data samples to or from the host.

This allows capture of raw IF data for processing on the host or running the Piksi from pre-recorded data or simulator output for hardware-in-the-loop testing.

UARTs A & B

Two UARTs provide high-speed 3.3V LVTTL level asynchronous serial interfaces which can be configured to transmit NMEA-0183 messages or binary navigation solution data, system status and debugging information and receive commands or differential corrections from the host or another Piksi board.

When configured in USB-Serial bridge mode, the USB interface functions identically to the two dedicated UARTs.

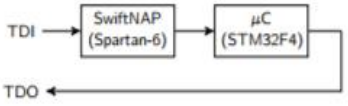
External Antenna

An external active antenna input is provided on an SMA connector and features a software switchable 3.3V bias.

JTAG

No JTAG adapter is required to develop for the Piksi as the board is supplied with a built-in bootloader.

For advanced debugging, a 0.05" pitch micro JTAG header compatible with the ARM Cortex-M standard pinout[1] is provided on the board. This allows access to the Spartan-6 FPGA and STM32F4 microcontroller JTAG interfaces.



Debug and Expansion Header

Access is provided to debugging signals from the SwiftNAP and I/O for future expansion boards and accessories. Assignment of these signals varies depending on the SwiftNAP firmware configuration.

Power

Power may be supplied to the board either over USB or through the V+ pins on the UART connectors. A 3.3V output from the on-board switching regulator is provided to power any external peripherals.

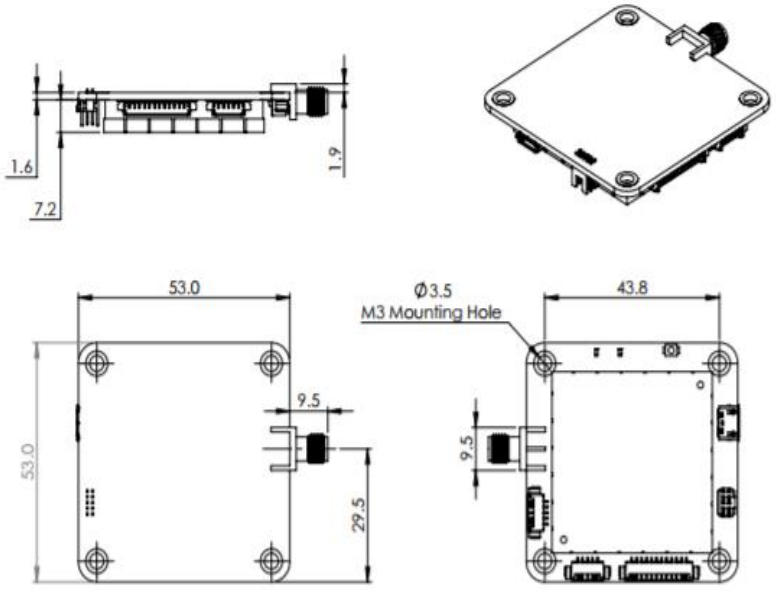
[1] ARM Cortex-M Debug Connector specification, http://bit.ly/1Cb6W6

Electrical Specifications

Supply voltage	3.5 – 5.5 V	Active antenna input impedance	50 Ω
Maximum voltage rating	5.5 V[1]	Active antenna bias voltage	3.3 V[3]
Power consumption	500 mW[2]	Max. antenna bias current draw	57 mA
Max. 3.3V output current draw	500 mA		

[1] Piksi provides no overvoltage protection and even momentary overvoltage can permanently damage the device.
[2] Typical, dependant on firmware configuration.
[3] Switchable in software

Mechanical Drawing



All dimensions are in millimeters. Drawing not to scale.

Notes

1. Mass 32g.
2. M3 mounting holes are plated through and connected internally to ground.
3. 3D CAD models are available from our website, http://www.swiftnav.com.

Copyright © 2013-2016 Swift Navigation, Inc.

UAV 3DR+ 8X

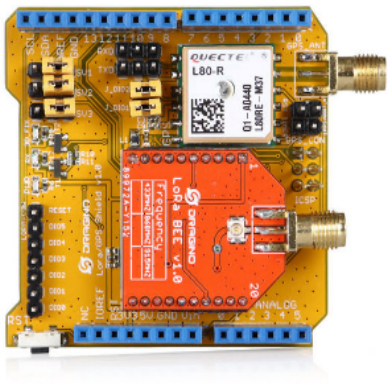
3DR X8-M Specifications

v3

Autopilot:	3DR Pixhawk V2.4.5
Firmware:	ArduCopter 3.2
GPS:	3DR u-blox GPS with Compass (LEA-6H module, 5 Hz update)
Telemetry radio:	3DR Radio v2 (915 mHz or 433 mHz)
Motors:	SunnySky V2216-12 800 kV II
Frame:	X
Propellers:	APC 11X4.7 SFP (4), APC 11X4.7 SF (4)
Battery:	4S 10000 mAh 10C lithium polymer
Aircraft weight (with battery):	7.7 lbs (3.5 kg)
Aircraft dimensions:	13.7 in x 20.1 in x 11.8 in (35 cm x 51 cm x 30 cm)
Case dimensions:	60.7 in x 14.5 in x 15.5 in (154 cm x 37 cm x 39 cm)
Payload capacity:	.4 lbs (200 g)
Radio range:	.6 miles* (1 km)
Flight time:	14 min*
Maximum operational wind speed:	25 mph (11 m/s)
Landing accuracy:	8.2 ft (2.5 m)
Recommended flight speed:	11 mph (5 m/s)
Camera:	Canon SX260 12.1 megapixel
Camera software:	3DR EAI 1.0
Area coverage (single flight):	25 acres* (0.1 km^2)
Orthomosaic accuracy:	3-16 ft (1-5 m)
Ground sampling distance:	.7 inches per pixel* (2 cm per pixel)
Image processing software:	Pix4Dmapper LT 3DR Edition (Windows only)

*Figures reflect estimated values at ideal operating conditions. Environmental conditions can affect flight time, range, area coverage, and ground sampling distance.

LoRa Dragino Shield de 915MHz



LoRa/GPS Shield with LoRa BEE

Specifications

LoRa Spec

- 168 dB maximum link budget.
- +20 dBm - 100 mW constant RF output vs.
- +14 dBm high efficiency PA.
- Programmable bit rate up to 300 kbps.
- High sensitivity: down to -148 dBm.
- Bullet-proof front end: IIP3 = -12.5 dBm.
- Excellent blocking immunity.
- Low RX current of 10.3 mA, 200 nA register retention.
- Fully integrated synthesizer with a resolution of 61 Hz.
- FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation.
- Built-in bit synchronizer for clock recovery.
- Preamble detection.
- 127 dB Dynamic Range RSSI.
- Automatic RF Sense and CAD with ultra-fast AFC.
- Packet engine up to 256 bytes with CRC.
- Built-in temperature sensor and low battery indicator.

Dimensions and Weight:

- Size: 31mm*25mm*10mm.
- Net weight: 4g.

Featrues

- 3.3V power supply
- Frequency Band: **868 MHZ/433 MHZ/915 MHZ**(Pre-configure in factory)
- Low power consumption
- Compatible with Arduino
- **LoRa™** Modem
- **FSK, GFSK, MSK, GMSK, LoRa™and OOK** modulation
- Preamble detection
- **Baud rate configurable**
- Built-in temperature sensor and low battery indicator
- External Antenna via I-Pex connector/SMA connector

Usage Notice

You have to be aware that Radio link quality and performances are highly dependent of environment.

Better performances can be reached with:

- Outdoor environment.
- No obstacles.
- No high level radio interferer in the ISM 868MHz band.
- At least 1 meter above the ground.

Radio performances are degraded with:

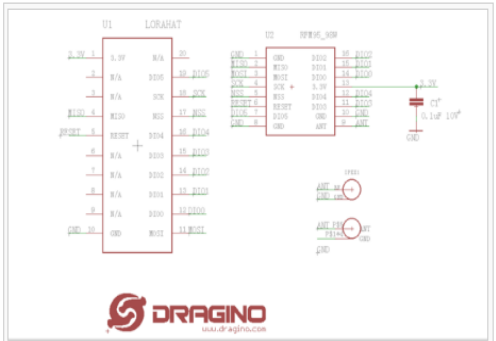
- Obstacles: buildings, trees...
- Inner buildings environments.
- High ISM 868MHz band usage by other technologies.

Radio communication are usually killed with bad topographic conditions. It is usually not possible to communicate through a hill, even very small.

Order Infomation

- LoRa BEE 868: Load with SX1276, support 868M frenquency
- LoRa BEE 915: Load with SX1276, support 915M frenquency
- LoRa BEE 433: Load with SX1278, support 433M frenquency

Schematic



Dragino LoRa BEE Schematic

Arduino Mega ATmega2560

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Raspberry pi 3 model B

2 Features

2.1 Hardware

- Low cost
- Low power
- High availability
- High reliability
  - Tested over millions of Raspberry Pis Produced to date
  - Module IO pins have 35u hard gold plating

2.2 Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

2.3 Software

- ARMv6 (CM1) or ARMv7 (CM3, CM3L) Instruction Set
- Mature and stable Linux software stack
  - Latest Linux Kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Full availability of GPU functions using standard APIs

3 Block Diagram

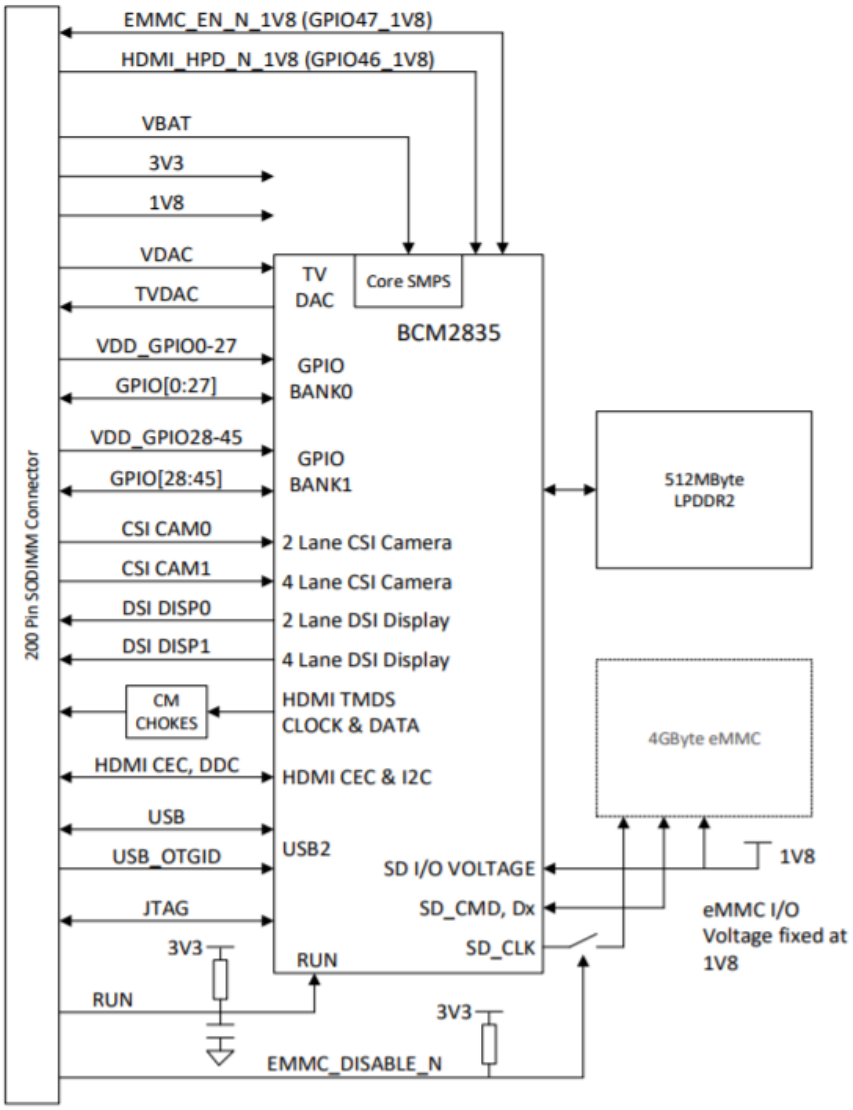


Figure 1: CM1 Block Diagram

4 Mechanical Specification

The Compute Modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules (with the exception that the CM3, CM3L modules are 31mm in height rather than 30mm of CM1) and therefore should work with the many DDR2 SODIMM sockets available on the market. **(Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.)**

The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture.

The maximum component height on the underside of the Compute Module is 1.2mm.

The maximum component height on the top side of the Compute Module is 1.5mm.

The Compute Module PCB thickness is 1.0mm +/- 0.1mm.

Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified.

Figure 3 gives the CM1 mechanical dimensions. Figure 4 gives the CM3 and CM3L mechanical dimensions.

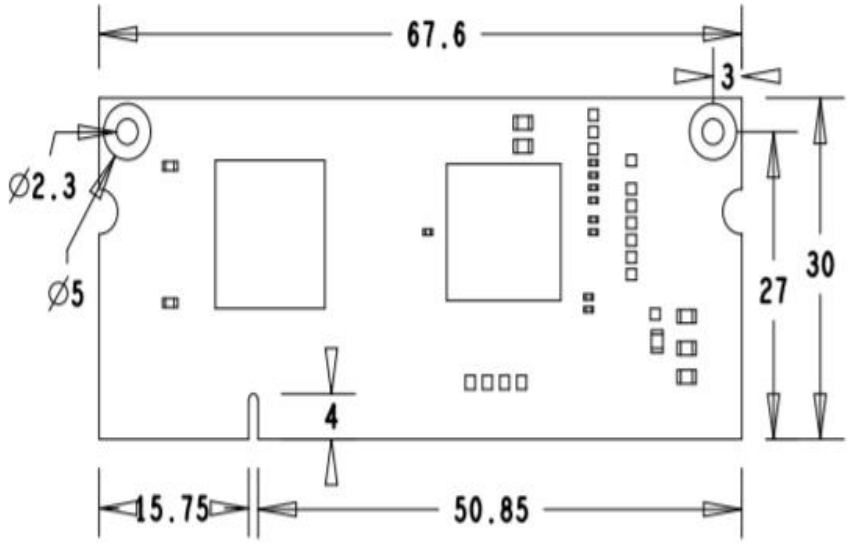


Figure 3: CM1 Mechanical Dimensions

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,e]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

[a] The PDN column indicates power-down state (when RUN pin LOW)
[b] Must be driven by an open-collector driver
[c] GPIO have software enabled pulls which keep state over power-down
[d] Only available on Lite variants
[e] The CM will always try to boot from this interface first
[f] Requires external pull-up resistor to 5V as per HDMI spec

Table 3: Pin Functions

6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
V_{IH}	Input high voltage[a]	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
V_{OH}	Output high voltage[b]	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 5: DC Characteristics

Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

[a] Requires a clean 2.5-2.8V supply if TV DAC is used, else connect to 3V3

Table 7: Power Supply Operating Ranges

7.1 Supply Sequencing

Supplies should be staggered so that the highest voltage comes up first, then the remaining voltages in descending order. This is to avoid forward biasing internal (on-chip) diodes between supplies, and causing latch-up. Alternatively supplies can be synchronised to come up at exactly the same time as long as at no point a lower voltage supply rail voltage exceeds a higher voltage supply rail voltage.

7.2 Power Requirements

Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused, it is usually in a low power state or completely turned off. For instance, if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. This is also the case for camera and display interfaces, HDMI, USB interfaces, video encoders and decoders, and so on.

Powerchain design is critical for stable and reliable operation of the Compute Module. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

Table 8 specifies the recommneded minimum power supply outputs required to power the Compute Module.

Supply	Minimum Requirement	Unit
VBAT (CM1)	2000[a]	mW
VBAT (CM3,3L)	3500[a]	mW
3V3	250	mA
1V8	250	mA
VDAC	25	mA
GPIO0-27_VDD	50[b]	mA
GPIO28-45_VDD	50[b]	mA
SDX_VDD	50[b]	mA

[a] Recommended minimum. Actual power drawn is very dependent on use-case
[b] Each GPIO can supply up to 16mA, aggregate current per bank must not exceed 50mA

Table 8: Mimimum Power Supply Requirements

8 Booting

The 4GB eMMC Flash device on CM3 is directly connected to the primary BCM2837 SD/eMMC interface. These connections are not accessible on the module pins. On CM3L this SD interface is available on the SDX_ pins.

When initially powered on, or after the RUN pin has been held low and then released, the BCM2837 will try to access the primary SD/eMMC interface. It will then look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the SD/eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB; in other words, its USB port is in slave mode waiting to accept boot code from a suitable host.

A USB boot tool is available on Github which allows a host PC running Linux to write the BCM2837 boot code over USB to the module. That boot code then runs and provides access to the SD/eMMC as a USB mass storage device, which can then be read and written using the host PC. Note that a Raspberry Pi can be used as the host machine. For those using Windows a precompiled and packeged tool is available. For more information see here.

The Compute Module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the SD/eMMC interface (by physically disconnecting the SD_CMD pin), forcing BCM2837 to boot from USB. Note that when the eMMC is disabled in this way, it takes a couple of seconds from powering up for the processor to stop attempting to talk to the SD/eMMC device and fall back to booting from USB.

Note that once booted over USB, BCM2837 needs to re-enable the SD/eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the EMMC_EN_N_1V8 pin LOW, which at boot is initially an input with a pull up to 1V8. If an end user wishes to add the ability to access the SD/eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and SD/eMMC must be used; that is, EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET, with the gate controlled by EMMC_EN_N_1V8. Ensure you use MOSFETs suitable for switching at 1.8V (i.e. use a device with gate threshold voltage, Vt, suitable for 1.8V switching).

9 Peripherals

9.1 GPIO

BCM283x has in total 54 GPIO lines in 3 separate voltage banks. All GPIO pins have at least two alternative functions within the SoC. When not used for the alternate peripheral function, each GPIO pin may be set as an input (optionally as an interrupt) or an output. The alternate functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage.

On CM1, CM3 and CM3L bank2 is used on the module to connect to the eMMC device and, on CM3 and CM3L, for an on-board I2C bus (to talk to the core SMPS and control the special function pins). On CM3L most of bank 2 is exposed to allow a user to connect their choice of SD card or eMMC device (if required).

Bank0 and 1 GPIOs are available for general use. GPIO0 to GPIO27 are bank 0 and GPIO28-45 make up bank1. GPIO0-27_VDD is the power supply for bank0 and GPIO28-45_VDD is the power supply for bank1. SDX_VDD is the supply for bank2 on CM3L. These supplies can be in the range 1.8V-3.3V (see Table 7) and are not optional; each bank must be powered, even when none of the GPIOs for that bank are used.

Note that the HDMI_HPD_N_1V8 and EMMC_EN_N_1V8 pins (on CM1 these were called GPIO46_1V8 and GPIO47_1V8 respectively) are 1.8V IO and are used for special functions (HDMI hot plug detect and boot control respectively). Please do not use these pins for any other purpose, as the software for the Compute Module will always expect these pins to have these special functions. If they are unused please leave them unconnected.

All GPIOs except GPIO28, 29, 44 and 45 have weak in-pad pull-ups or pull-downs enabled when the device is powered on. It is recommended to add off-chip pulls to GPIO28, 29, 44 and 45 to make sure they never float during power on and initial boot.

9.1.1 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	-	-	-
1	High	SCL0	SA4	DE	-	-	-
2	High	SDA1	SA3	LCD_VSYNC	-	-	-
3	High	SCL1	SA2	LCD_HSYNC	-	-	-
4	High	GPCLK0	SA1	DPI_D0	-	-	ARM_TDI
5	High	GPCLK1	SA0	DPI_D1	-	-	ARM_TDO
6	High	GPCLK2	SOE_N	DPI_D2	-	-	ARM_RTCK
7	High	SPI0_CE1_N	SWE_N	DPI_D3	-	-	-
8	High	SPI0_CE0_N	SD0	DPI_D4	-	-	-
9	Low	SPI0_MISO	SD1	DPI_D5	-	-	-
10	Low	SPI0_MOSI	SD2	DPI_D6	-	-	-
11	Low	SPI0_SCLK	SD3	DPI_D7	-	-	-
12	Low	PWM0	SD4	DPI_D8	-	-	ARM_TMS
13	Low	PWM1	SD5	DPI_D9	-	-	ARM_TCK
14	Low	TXD0	SD6	DPI_D10	-	-	TXD1
15	Low	RXD0	SD7	DPI_D11	-	-	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	-	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	-	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	-	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	-	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	-
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	-
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	-
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	-
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	-
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	-

Table 9: GPIO Bank0 Alternate Functions

9.1.4 SD/SDIO Interface

The BCM283x supports two SD card interfaces, SD0 and SD1.

The first (SD0) is a proprietary Broadcom controller that does not support SDIO and is the primary interface used to boot and talk to the eMMC or SDX_x signals.

The second interface (SD1) is standards compliant and can interface to SD, SDIO and eMMC devices; for example on a Raspberry Pi 3 it is used to talk to the on-board BCM43438 WiFi device in SDIO mode.

Both interfaces can support speeds up to 50MHz single ended (SD High Speed Mode).

9.2 CSI (MIPI Serial Camera)

Currently the CSI interface is not openly documented and only CSI camera sensors supported by the official Raspberry Pi firmware will work with this interface. Supported sensors are the OmniVision OV5647 and Sony IMX219.

It is recommended to attach other cameras via USB.

9.3 DSI (MIPI Serial Display)

Currently the DSI interface is not openly documented and only DSI displays supported by the official Raspberry Pi firmware will work with this interface.

Displays can also be added via the parallel DPI interface which is available as a GPIO alternate function - see Table 9 and Section 9.1.3

9.4 USB

The BCM283x USB port is On-The-Go (OTG) capable. If using either as a fixed slave or fixed master, please tie the USB_OTGID pin to ground.

The USB port (Pins USB_DP and USB_DM) must be routed as 90 ohm differential PCB traces.

Note that the port is capable of being used as a true OTG port however there is no official documentation. Some users have had success making this work.

9.5 HDMI

BCM283x supports HDMI V1.3a.

It is recommended that users follow a similar arrangement to the Compute Module IO Board circuitry for HDMI output.

The HDMI CK_P/N (clock) and D0-D2_P/N (data) pins must each be routed as matched length 100 ohm differential PCB traces. It is also important to make sure that each differential pair is closely phase matched. Finally, keep HDMI traces well away from other noise sources and as short as possible.

Failure to observe these design rules is likely to result in EMC failure.

9.6 Composite (TV Out)

The TVDAC pin can be used to output composite video (PAL or NTSC). Please route this signal away from noise sources and use a 75 ohm PCB trace.

Note that the TV DAC is powered from the VDAC supply which must be a clean supply of 2.5-2.8V. It is recommended users generate this supply from 3V3 using a low noise LDO.

If the TVDAC output is not used VDAC can be connected to 3V3, but it must be powered even if the TV-out functionality is unused.

10 Thermals

The BCM283x SoC employs DVFS (Dynamic Voltage and Frequency Scaling) on the core voltage. When the processor is idle (low CPU utilisation), it will reduce the core frequency and voltage to reduce current draw and heat output. When the core utilisation exceeds a certain threshold the core votlage is increased and the core frequency is boosted to the maximum working frerquency. The voltage and frequency are throttled back when the CPU load reduces back to an 'idle' level OR when the silicon temperature as mesured by the on-chip temperature sensor exceeds 85C (thermal throttling).

A designer must pay careful attention to the thermal design of products using the CM3/CM3L so that performance is not artificially curtailed due to the processor thermal throttling, as the Quad ARM complex in the BCM2837 can generate significant heat output.

10.1 Temperature Range

The operating temperature range of the module is set by the lowest maximum and highest minimum of any of the components used.

The eMMC and LPDDR2 have the narrowest range, these are rated for -25 to +80 degrees Celsius. Therefore the nominal range for the CM3 and CM3L is -25C to +80C.

However, this range is the maximum for the silicon die; therefore, users would have to take into account the heat generated when in use and make sure this does not cause the temperature to exceed 80 degrees Celsius.

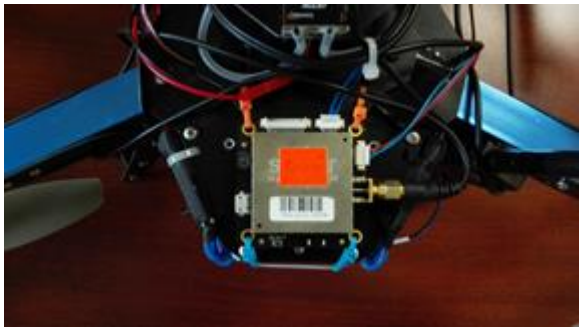
11 Availability

Raspberry Pi guarantee availability of CM1, CM3 and CM3 Lite until at least January 2023.

12 Support

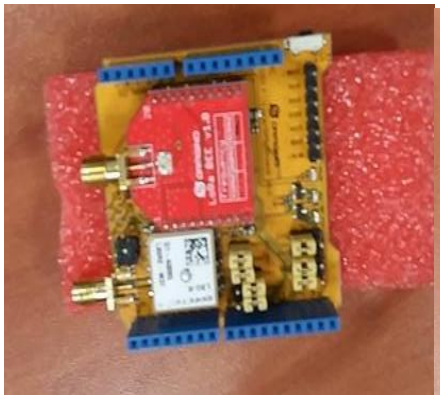
For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

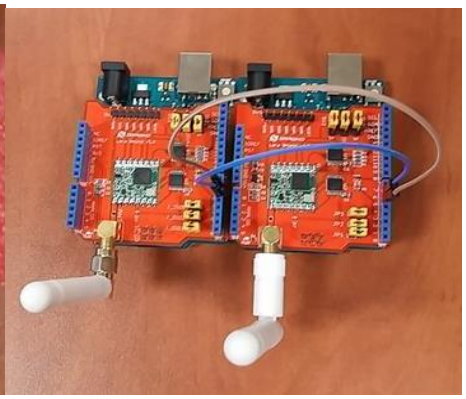
Pruebas realizadas en campo



Dispositivos usados

}





CODIFICACIÓN

Piksi_Rover

```
#include <SPI.h>

#include <RH_RF95.h>

#include <Wire.h>

#include <SoftwareSerial.h>

#include <RHReliableDatagram.h>

#define CLIENT_ADDRESS 1

#define SERVER_ADDRESS 2

RH_RF95 rf95;
```

```
// Serial pins
SoftwareSerial mySerial(5, 6); // RX, TX
int led = 8;
void setup() {
  Serial.begin(57600);
  while(!Serial){
    ;
  }

  if(!rf95.init()){
    Serial.println("LoRa radio init failed");
    while(1);
  }
  //mySerial.begin(57600);

  if(!rf95.setFrequency(915)){
    Serial.println("setFrequency failed");
    while(1);
  }

// Bw125Cr45Sf128 -> Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on. Default medium range
// Bw500Cr45Sf128 -> Bw = 500 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on. Fast+short range
  if(!rf95.setModemConfig(RH_RF95::Bw125Cr45Sf128)){
```

```
    Serial.println("Modem config error");
    while(1);
  }
}


int data;

void loop() {
  if (rf95.available())
  {
    // Should be a message for us now
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {

      digitalWrite(led, HIGH);
    //RH_RF95::printBuffer("request: ", buf, len);
      //Serial.println("got request: ");
      //Serial.println(*buf);
   // Serial.print("RSSI: ");
   // Serial.println(rf95.lastRssi(), DEC);

      //mySerial.write(*buf);
```

```
      Serial.write(*buf);
      // Send a reply

      if (Serial.available()) {

        data = Serial.read();
      }
      rf95.send(data,sizeof(data));
      rf95.waitPacketSent();
      //Serial.println("Sent a reply");
      digitalWrite(led, LOW);
    }
    else
    {
      Serial.println("recv failed");
    }
  }
  delay(400);
}
```

Piksi_Base

```
#include <SPI.h>
#include <RH_RF95.h>
#include <Wire.h>
#include <RHReliableDatagram.h>
```

```
#include <SoftwareSerial.h>

#define CLIENT_ADDRESS 1
#define SERVER_ADDRESS 2


//MS5837 sensor;
RH_RF95 rf95;

// Serial pins
SoftwareSerial mySerial(5, 6); // RX, TX

void setup() {
  Serial.begin(115200);
  while(!Serial)
  {
    ;
  }

  if(!rf95.init()){
    Serial.println("LoRa radio init failed");
    while(1);
  }

  mySerial.begin(57600);
```

```
  if(!rf95.setFrequency(915)){
    Serial.println("setFrequency failed");
    while(1);
  }

// Bw125Cr45Sf128 -> Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on. Default medium range
// Bw500Cr45Sf128 -> Bw = 500 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on. Fast+short range
  if(!rf95.setModemConfig(RH_RF95::Bw125Cr45Sf128)){
    Serial.println("Modem config error");
    while(1);
  }
}

int data[200];
int i;
void loop() {
  i=0;
  while (Serial.available()>0) {


    data[i] = Serial.read();
    i=i+1;
```

```
}
Serial.println(*data);


rf95.send(*data,sizeof(*data));


rf95.waitPacketSent();

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);

if (rf95.waitAvailableTimeout(3000))
{
  // Should be a reply message for us now
  if (rf95.recv(buf, &len))
 {
    //Serial.print("got reply: ");
    //Serial.println(buf);
    // Serial.write(*buf);
   // Serial.print("RSSI: ");
    //Serial.println(rf95.lastRssi(), DEC);
    // mySerial.write(*buf);
  }
  else
```

```
    {
      Serial.println("recv failed");
    }
  }
  else
  {
    Serial.println("No reply, is rf95_server running?");
  }
  delay(400);
}
```

CPU embebido

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <math.h>
#include <time.h>

#include <libserialport.h>

#include <libsbp/sbp.h>
#include <libsbp/piksi.h>
#include <libsbp/navigation.h>
#include <libsbp/system.h>
```

```
// Log variables


char buffer1[80];
char buffer2[80];
char buffer3[80];

FILE *archivo1, *archivo2, *archivo3;
struct timespec t1, t2;
double tiempo;
int flag_f = 1;

// Serial variables
char *serial_port_name = NULL;
struct sp_port *piksi_port = NULL;


// Setup PIKSI
// Piksi variables

sbp_state_t sbp_state;   // State of the SBP parser.


/* SBP structs that messages from Piksi will feed. */
```

```
msg_pos_llh_t     pos_llh;
msg_baseline_ned_t      baseline_ned;
msg_vel_ned_t    vel_ned;
msg_dops_t        dops;
msg_gps_time_t gps_time;
msg_utc_time_t utc_time;
msg_iar_state_t iar_state;
msg_pos_ecef_t pos_ecef;

/* SBP callback node must be statically allocated. Each message ID / callback pair
must have a unique sbp_msg_callbacks_node_t associated with it*/

static sbp_msg_callbacks_node_t pos_llh_node;
static sbp_msg_callbacks_node_t baseline_ned_node;
static sbp_msg_callbacks_node_t vel_ned_node;
static sbp_msg_callbacks_node_t dops_node;
static sbp_msg_callbacks_node_t gps_time_node;
static sbp_msg_callbacks_node_t utc_time_node;
static sbp_msg_callbacks_node_t iar_state_node;
static sbp_msg_callbacks_node_t pos_ecef_node;


/* Callback funtions to interpret SBP messages.
*  Every message ID has a callback associated with it to
```

```
*  receive and interpret the message payload
*/
void sbp_utc_time_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    utc_time = *(msg_utc_time_t *)msg;
}

void sbp_pos_llh_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    pos_llh = *(msg_pos_llh_t *)msg;
}

void sbp_baseline_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    baseline_ned = *(msg_baseline_ned_t *)msg;
}

void sbp_vel_ned_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    vel_ned = *(msg_vel_ned_t *)msg;
}

void sbp_dops_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    dops = *(msg_dops_t *)msg;
```

```
}

void sbp_gps_time_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    gps_time = *(msg_gps_time_t *)msg;
}

void sbp_iar_state_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    iar_state = *(msg_iar_state_t *)msg;
}

void sbp_pos_ecef_callback(u16 sender_id, u8 len, u8 msg[], void *context)
{
    pos_ecef = *(msg_pos_ecef_t *)msg;
}

void sbp_setup(void)
{
    int ret = -5;

    /* SBP parser state must be initialized before sbp_process is called*/
    sbp_state_init(&sbp_state);
```

```
/* Register a node and callback, and associate them with a specific message ID*/

ret = sbp_register_callback(&sbp_state, SBP_MSG_UTC_TIME, &sbp_utc_time_callback, NULL, &utc_time_node);
if(ret != SBP_OK){
        printf("SBP_CALLBACK_ERROR");
        exit(EXIT_FAILURE);
}

ret = sbp_register_callback(&sbp_state, 0x100, &sbp_gps_time_callback, NULL, &gps_time_node);
if(ret != SBP_OK){
        printf("SBP_CALLBACK_ERROR");
        exit(EXIT_FAILURE);
}

ret = sbp_register_callback(&sbp_state, 0x201, &sbp_pos_llh_callback, NULL, &pos_llh_node);
if(ret != SBP_OK){
        printf("SBP_CALLBACK_ERROR");
        exit(EXIT_FAILURE);
}

ret = sbp_register_callback(&sbp_state, 0x203, &sbp_baseline_callback, NULL, &baseline_ned_node);
```

```
    if(ret != SBP_OK){
            printf("SBP_CALLBACK_ERROR");
            exit(EXIT_FAILURE);
    }

    ret = sbp_register_callback(&sbp_state, 0x205, &sbp_vel_ned_callback,
NULL, &vel_ned_node);
    if(ret != SBP_OK){
            printf("SBP_CALLBACK_ERROR");
            exit(EXIT_FAILURE);
    }

    ret = sbp_register_callback(&sbp_state, 0x206, &sbp_dops_callback, NULL,
&dops_node);
    if(ret != SBP_OK){
            printf("SBP_CALLBACK_ERROR");
            exit(EXIT_FAILURE);
    }

    ret = sbp_register_callback(&sbp_state, 0x0019, &sbp_iar_state_callback,
NULL, &iar_state_node);
    if(ret != SBP_OK){
            printf("SBP_CALLBACK_ERROR");
            exit(EXIT_FAILURE);
    }
```

```
    ret = sbp_register_callback(&sbp_state, 0x0200, &sbp_pos_ecef_callback,
NULL, &pos_ecef_node);
    if(ret != SBP_OK){
            printf("SBP_CALLBACK_ERROR");
            exit(EXIT_FAILURE);
    }

    printf("SBP_SETUP_OK\n\n");
}


// Setup PORT
void usage(char *prog_name){
    fprintf(stderr, "usage: %s [-p serial port]\n", prog_name);
}

void setup_port()
{
    int result;
    // ./piksi_datalogger -p /dev/tty(serial port)
    result = sp_set_baudrate(piksi_port, 57600);
    if(result != SP_OK){
            fprintf(stderr, "Cannot set port baud rate!\n");
            exit(EXIT_FAILURE);
```

```
}

result = sp_set_flowcontrol(piksi_port, SP_FLOWCONTROL_NONE);
if(result != SP_OK){
        fprintf(stderr, "Cannot set flow control!\n");
        exit(EXIT_FAILURE);
}

result = sp_set_bits(piksi_port, 8);
if(result != SP_OK){
        fprintf(stderr, "Cannot set data bits!\n");
        exit(EXIT_FAILURE);
}

result = sp_set_parity(piksi_port, SP_PARITY_NONE);
if(result != SP_OK){
        fprintf(stderr, "Cannot set parity!\n");
        exit(EXIT_FAILURE);
}

result =sp_set_stopbits(piksi_port, 1);
if(result != SP_OK){
        fprintf(stderr, "Cannot set stop bits!\n");
        exit(EXIT_FAILURE);
}
```

```
}


u32 piksi_port_read(u8 *buff, u32 n, void *context)
{
    (void)context;
    u32 result;
    result = sp_blocking_read(piksi_port, buff, n, 0);
    return result;
}


// Setup file headers and names
void setup_files(struct tm * time_gps)
{
    printf("Creating FILES\n\n");



    strftime(buffer1,  sizeof(buffer1)-1,  "baseline_log_%Y%m%d-%H%H%S.txt",
time_gps);
    strftime(buffer2,  sizeof(buffer2)-1,  "position_log_%Y%m%d-%H%H%S.txt",
time_gps);
    strftime(buffer3,  sizeof(buffer3)-1,  "velocity_log_%Y%m%d-%H%H%S.txt",
time_gps);
```

```
    if( (archivo1 = fopen(buffer1, "w+") ) == NULL)
    {
            printf("ERROR: Cannot open the file");
            exit(EXIT_FAILURE);
    }

    if( (archivo2 = fopen(buffer2, "w+") ) == NULL)
    {
            printf("ERROR: Cannot open the file");
            exit(EXIT_FAILURE);
    }

    if( (archivo3 = fopen(buffer3, "w+") ) == NULL)
    {
            printf("ERROR: Cannot open the file");
            exit(EXIT_FAILURE);
    }

    fprintf(archivo1,"time,distance(meters),num_sats,flags\n");
    fprintf(archivo2,"time,latitude(degrees),longitudes(degrees),altitude(meters),n
_sats,flags\n");
    fprintf(archivo3,"time,speed(m/s),num_sats\n");

}
```

```
int main(int argc , char **argv)
{
    int opt;
    int result = 0;

    time_t seconds;
    struct tm * time_gps;

    double dist2, dist;


    if(argc <= 1) {
            usage(argv[0]);
            exit(EXIT_FAILURE);
    }

    while((opt = getopt(argc, argv, "p:")) != -1) {
            switch(opt){
                    case 'p':
                            serial_port_name = (char *)calloc(strlen(optarg) + 1,
sizeof(char));
                            if(!serial_port_name){
                                    fprintf(stderr, "Cannot allocate memory!\n");
```

```
                }
                strcpy(serial_port_name, optarg);
                break;
            case 'h':
                usage(argv[0]);
                exit(EXIT_FAILURE);
        }
    }


    if(!serial_port_name){
        fprintf(stderr, "Please supply the serial port path where the piksi is connected!\n");
        exit(EXIT_FAILURE);
    }

    result = sp_get_port_by_name(serial_port_name, &piksi_port);
    if(result != SP_OK){
        fprintf(stderr, "Cannot find provided serial port!\n");
        exit(EXIT_FAILURE);
    }

    result = sp_open(piksi_port, SP_MODE_READ);
    if(result != SP_OK){
        fprintf(stderr, "Cannot open %s for reading!\n", serial_port_name);
```

```
        exit(EXIT_FAILURE);
}

setup_port();

sbp_setup();

printf("Starting..!\n\n");
while(1){
        int ret = sbp_process(&sbp_state, &piksi_port_read);

        if(ret < 0)
                printf("sbp_process error\n");

        if(ret == SBP_OK_CALLBACK_EXECUTED)
        {

                seconds = (time_t)(gps_time.tow/1000);
                time_gps = localtime(&seconds);

                if(flag_f == 1){
                        setup_files(time_gps);
                        flag_f = 0;
                        printf("Started!\n\n");
                }
```

```
                strftime(buffer3,sizeof(buffer3)-
1,"%Y%m%d_%H%M%S",time_gps);


                /* Print GPS time */
                printf("GPS Time:\n");
                printf("\tWeek\t\t: %6d\n", (int)gps_time.wn);
                printf("\tSeconds\t\t:%s", buffer3);
                printf("\n");

                /* Print single point position ECEF*/
                printf("Single point position:\n");
                printf("\tX\t: %.6lf\n",pos_ecef.x);
                printf("\tY\t: %.6lf\n", pos_ecef.y);
                printf("\tZ\t: %.6lf\n", pos_ecef.z);
                printf("\n");



                /* Print absolute position. */
                printf("Absolute Position:\n ");
                printf("\tLatitude\t: %4.10lf\n", pos_llh.lat);
                printf("\tLongitude\t: %4.10lf\n", pos_llh.lon);
                printf("\tHeight\t\t: %4.10lf\n", pos_llh.height);
```

```
                printf("\tSatellites\t:  %02d\n", pos_llh.n_sats);
                printf("\tState\t: %d\n",pos_llh.flags);
                printf("\n");

                /* Print IAR state*/
                printf("IAR STATE:\n");
                printf("\tIAR number\t: %3d\n",iar_state.num_hyps);
                printf("\n");


                dist2 = pow(((double)baseline_ned.n)/1000, 2) +
pow(((double)baseline_ned.e)/1000, 2) + pow(((double)baseline_ned.d)/1000, 2);
                dist = sqrt(dist2);

                /* Print NED (North/East/Down) baseline (position vector from
base to rover). */
                printf("Baseline (mm):\n");
                printf("\tNorth\t\t: %6d\n", (int)baseline_ned.n);
                printf("\tEast\t\t: %6d\n", (int)baseline_ned.e);
                printf("\tDown\t\t: %6d\n", (int)baseline_ned.d);
                printf("\tDistance\t\t: %.2f\n",dist);
                printf("\n");

                /* Print NED velocity. */
                printf("velocity (mm/s):\n");
```

```
                printf("\tNorth\t\t: %6d\n", (int)vel_ned.n);
                printf("\tEast\t\t: %6d\n", (int)vel_ned.e);
                printf("\tDown\t\t: %6d\n", (int)vel_ned.d);
                printf("\n");

                /* Print dilution of Precision metrics. */
                printf("Dilution of Precision:\n");
                printf("\tGDOP\t\t: %4.2f\n", ((float)dops.gdop/100));
                printf("\tHDOP\t\t: %4.2f\n", ((float)dops.hdop/100));
                printf("\tPDOP\t\t: %4.2f\n", ((float)dops.pdop/100));
                printf("\tTDOP\t\t: %4.2f\n", ((float)dops.tdop/100));
                printf("\tVDOP\t\t: %4.2f\n", ((float)dops.vdop/100));
                printf("\n");


    fprintf(archivo1,"%s,%.2f,%2d,%2d\n",buffer3,dist,baseline_ned.n_sats,baseli
ne_ned.flags);

    fprintf(archivo2,"%s,%4.10lf,%4.10lf,%4.10lf,%2d,%2d\n",buffer3,pos_llh.lat,p
os_llh.lon,pos_llh.height,pos_llh.n_sats,pos_llh.flags);

    fprintf(archivo3,"%s,%6d,%2d\n",buffer3,vel_ned.d,vel_ned.n_sats);


            }
    }
```

```
result = sp_close(piksi_port);
if(result != SP_OK){
        fprintf(stderr, "Cannot close %s properly!\n", serial_port_name);
}

sp_free_port(piksi_port);

free(serial_port_name);

return 0;}
```