UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un enjambre de drones Crazyflie dentro de un ecosistema robótico basado en captura de movimiento

Trabajo de graduación presentado por José Emilio Gordillo de León
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un enjambre de drones Crazyflie dentro de un ecosistema robótico basado en captura de movimiento

Trabajo de graduación presentado por José Emilio Gordillo de León
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

Vo.Bo.:

(f) _____

MSc. Miguel Zea

Tribunal Examinador:

(f) _____

MSc. Miguel Zea

(f) _____

Dr. Luis Alberto Rivera

(f) _____

MSc. Pedro Iván Castillo

Fecha de aprobación: Guatemala, 3 de enero de 2023.

Prefacio

Este trabajo de graduación es el resultado de una larga y gratificante travesía académica. Me siento profundamente agradecido por la Universidad Del Valle y me asesor Miguel Zea, quien han contribuido de manera invaluable a este trabajo. Sin su apoyo y consejo, este trabajo no habría sido posible.

Así mismo quiero agradecer a mis padres por todo el cariño y soporte que siempre me han mostrado durante este proceso y que sin ellos este trabajo nunca hubiera sucedido. Por último agradezco a Santiago, Darío y Kenneth por su amistad que siempre me empujo a seguir trabajando para lograr este objetivo.

Espero que este trabajo de graduación sea un aporte valioso y que pueda ser de utilidad para otros estudiantes interesados en este tema.

Índice

Lista de figuras

Lista de cuadros

Resumen

En el presente proyecto se trabajó en el desarrollo de las herramientas para integrar un enjambre de Crazyflie 2.0 como agentes al ecosistema Robotat utilizando el sistema de captura de movimiento Optitrack. Se comenzó revisando la posible adaptación del paquete Crazyswarm al ecosistema Robotat, este software permite controlar un enjambre de drones Crazyflie utilizando un sistema de captura de movimiento estándar, para comunicarse con los drones se crea un servidor utilizando ROS y antenas CrazyRadio, esto se consideró que entraba en conflicto con las arquitectura actual del Robotat por lo que se descarto su implementación.

Como siguiente punto se trabajó en el desarrollo de una arquitectura de comunicación que utilizara los recursos con los que ya cuenta la universidad. Para lograr esto el primer paso fue diseñar un un *deck* que le diera la capacidad al dron para comunicarse por medio de TCP con el ecosistema. Para esto se escogió un ESP8266 el cual integra una antena WiFi y cuenta con librerías para la comunicación TCP. Para transmitir el mensaje hacía el dron se utilizó UART y como protocolo de comunicación se utilizó CPX para mantener un estándar dentro del dron. Como resultado de esta antena se obtuvo una latencia de 4 ms hasta 45.6 ms.

Con el deck terminado se prosiguió a hacer los cambios internos al firmware del dron, en esta sección se activo el puerto UART, se agregó al router interno del CPX la función *Robotat* para poder almacenar los datos en una *queue* separada y por último se deserializado los datos recibidos y se ingresaron al estabilizador, encargado de la fusión de sensores así como al controlador.

Por último se comprobó que los datos recibidos en el Crazyflie y el estado estimado fueran similares. Para lograr esto se diseño un *frame* donde colocar *markers* que el sistema Optitrack es capaz de reconocer. Después se hicieron pruebas donde se colocó el Crazyflie en diferentes posiciones dentro de la mesa de pruebas y se tomaron muestras de la medición dada por el servidor y de la estimación dentro del dron.

Abstract

In the present project, work was done on the development of tools to integrate a swarm of Crazyflie 2.0 as agents into the Robotat ecosystem using the Optitrack motion capture system. The project began by reviewing the possible adaptation of the Crazyswarm package to the Robotat ecosystem. This software allows for controlling a swarm of Crazyflie drones using a standard motion capture system. To communicate with the drones, a server was created using ROS and CrazyRadio antennas. However, this was considered to be in conflict with the current architecture of the Robotat, so it was discarded.

Next, work was done on the development of a communication architecture that would utilize the resources already available at the university. The first step was to design a deck that would give the drone the ability to communicate via TCP with the ecosystem. To do this, an ESP8266 was chosen, which integrates a WiFi antenna and has libraries for TCP communication. To transmit the message to the drone, UART was used and the communication protocol used was CPX to maintain a standard within the drone. As a result of this antenna, a latency of 4 ms to 45.6 ms was obtained.

With the deck finished, changes were made to the drone's firmware. In this section, the UART port was activated, the internal router of the CPX was added to the Robotat function to store data in a separate queue, and finally, the received data was deserialized and entered into the stabilizer, responsible for sensor fusion and control.

Finally, it was verified that the data received in the Crazyflie and the estimated state were similar. To achieve this, a frame was designed where markers that the Optitrack system is able to recognize were placed. Then, tests were done where the Crazyflie was placed in different positions on the test table and samples were taken of the measurement given by the server and the estimation within the drone.

CAPÍTULO 1

Introducción

Debido a los diversos trabajos dentro de la Universidad del Valle con [1] ha surgido la necesidad de integrar más y diferente agentes a él. Con base en los resultados en [2], se evaluara la viabilidad de añadir dentro del ecosistema al agente Crazyflie 2.0.

Este trabajo pretende mostrar el proceso para formar e implementar herramientas básicas que puedan controlar un Crazyflie 2.0 valiéndose del sistema de captura de movimiento *Optitrack* y un servidor local. Para conseguir esto se evaluaran opciones ya preexistentes

Cada parte de este trabajo se evalúa por separado utilizando como indicadores claves las limitaciones del propio cuadricóptero. Primero se analizó el sistema Crazyswarm para averiguar si es favorable su implementación e integración al ecosistema Robotat. Como siguiente punto se diseñó una antena WiFi que funcione de puente entre un servidor y un Crazyflie 2.0, cumpliendo con las restricciones de peso máximo del dron. Para finalizar se realizó los cambios necesarios al firmware de un Crazyflie 2.0 para recibir paquetes de datos de un sistema de captura de movimiento y controlar el dron con esa información.

CAPÍTULO 2

Antecedentes

Los quadricópteros han obtenido gran aceptación, tanto a nivel comercial como académico, por las ventajas que presentan. Estos sistemas son de menor tamaño, peso y costo comparado con los vehículos no tripulados convencionales además son extremadamente versátiles lo que permite colocar sobre ellos todo tipo de sensores, cámaras y equipos para ser desplegados en casi cualquier ambiente ajustándose con precisión a tareas difíciles o de riesgo.

En la UVG se ha hecho el esfuerzo por estudiar este tipo de vehículos, consiguiendo los siguientes avances.

2.1. Diseño e implementación de una plataforma de pruebas para sistemas de control para el dron Crazyflie 2.0

En [2], se desarrolló una plataforma para el control de actitud de drones Crazyflies 2.0. Se realizó una guía básica para el uso del mismo, en esta se detallan los pasos para inicializar un dron así como los recursos a instalar para poder modificar su firmware. La guía también explica la instalación y configuración de una máquina virtual que contiene empaquetado todo el software necesarios para poder trabajar con el Crazyflie 2.0. Por último, se desarrolló una interfaz gráfica que utiliza un API de una librería de Python para controlar de manera sencilla los diferentes parámetros del dron.

Los enjambres de cuadricópteros han sido utilizados en ambientes cerrados para experimentar con formaciones de vuelos y rutinas colaborativas. Para controlar de manera robusta esta cantidad de vehículos se han personalizado sistemas de captura de movimiento con el objetivo de conocer su posición en todo momento. En la UVG se ha trabajado con estos sistemas de captura de movimiento para crear el ecosistema Robotat.

2.2. Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica

En [1], se creó una red de comunicación WiFi para varios agentes, que trabaja en conjunto con el sistema de captura de movimiento OptiTrack, para formar un ecosistema de experimentación robótico denominado Robotat.

El trabajo explica cómo ensamblar y configurar el sistema de captura de movimiento así como el desarrollo e implementación de una librería en C para un microcontrolador ESP32, para establecer comunicación por medio de MQTT entre el agente y OptiTrack. Por último, se explica la construcción de una antena inteligente, la cual permite a cualquier agente no robótico interactuar con el ecosistema.

Otras universidad han hecho ecosistemas similares enfocándose en los enjambres de cuadricópteros miniaturas.

2.3. Crazyswarm: A Large Nano-Quadcopter Swarm

En [3] se define la arquitectura de un sistema para enjambres a gran escala utilizando la plataforma Crazyflie 2.0 como vehículo y cámaras de la marca VICON como sistema de captura de movimiento. Este sistema utiliza esferas reflectoras para calcular la posición de un agente. Adicionalmente, para delimitar el tamaño de cada cuerpo de forma adecuada se necesita proveer al sistema de un mapa con la posición inicial de cada dron antes de iniciar una rutina.

Figura 1: Emjambre controlado con Crazyswarm [3]

El trabajo también describe la infraestructura de un sistema de comunicación de radio frecuencia de una vía, utilizando compresión de datos para aumentar la velocidad de transmisión y ofrecer soporte a una gran número de cuadricópteros. También se detalla como en diferentes situaciones se puede cambiar a un comunicación dos vías para obtener datos importantes de un dron.

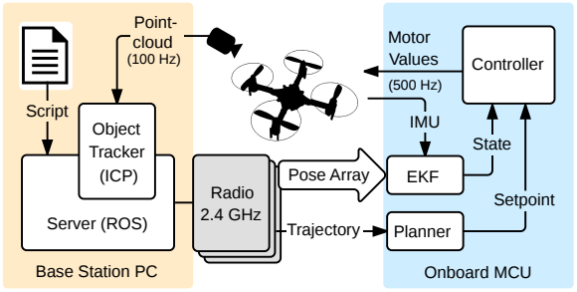

Figura 2: Arquitectura de comunicación de Crazyswarm [3]

Para finalizar, se explican los métodos utilizados en la planificación de trayectorias, la implementación de controlador no lineal de posición y velocidad dentro de cada dron y las herramientas de software necesarias para realizar acciones triviales como pueden ser reinicios en masa, actualizaciones de firmware o monitoreos sobre la batería de manera inalámbrica.

CAPÍTULO 3

Justificación

A medida que incrementa la complejidad de los sistemas robóticos, así como los trabajos que deben cumplir, es de suma importancia poseer entornos especializados los cuales funcionen como herramientas para facilitar el conocimiento así como agilizar el proceso de innovación.

Para suplir con esta necesidad, la Universidad del Valle de Guatemala ha creado el *ecosistema Robotat*, el cual se nutre de la integración de nuevos robots. Por tal razón y aprovechando los avances que se han logrado con [2], se decide agregar soporte a los drones Crazyflie 2.0.

Este proyecto plantea como base [2] y se busca combinarlo con [1] y [3]. Con este esfuerzo se espera lograr una conexión fluida entre *Robotat* y *Crazyswarm*, aumentando las capacidades del ecosistema para controlar enjambres de drones. También se llevara acabo una comparación entre la transmisión de datos por medio de RF y WiFi, para encontrar cual presenta los mejores resultados al aumentar el numero de agentes a controlar.

Con los resultados de este proyecto se busca sentar las bases que permitan, en un futuro, eficientar el proceso para pruebas y experimentos con enjambres de drones, aumentar la calidad de las investigaciones posteriores y proveer un sitio para la comunidad UVG así como a colaboradores externos donde puedan adquirir la experiencia necesaria para operar estos equipos.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Implementar un paquete de herramientas de software que permita ejecutar trayectorias de enjambre con el cuadricóptero Crazyflie 2.0 de Bitcraze dentro de un ecosistema robótico basado en el sistema de captura de movimiento OptiTrack.

4.2. Objetivos específicos

- Adaptar el paquete de herramientas Crazyswarm al ecosistema Robotat.
- Comparar la velocidad y escalabilidad de un sistema de comunicación con antenas RF a 2.4GHz contra la utilización de un sistema basado en WiFi y un microcontrolador de la familia ESP.
- Desarrollar herramientas de software de alto nivel para facilitar el uso y monitoreo de enjambres con drones Crazyflie 2.0.
- Validar los resultados obtenidos por medio del sistema de captura de movimiento OptiTrack.

CAPÍTULO 5

Alcance

Este trabajo se concentra en desarrollar las herramientas necesarias que permitan ejecutar trayectorias de enjambre con el Crazyflie 2.0, esto se unió al sistema de captura de movimiento valiéndose del ecosistema Robotat para obtener los datos sobre la pose del agente.

Se trabajó en la adaptación del paquete de herramientas Crazyswarm, evaluando la viabilidad de su unión con el ecosistema Robotat así como su escalabilidad en términos de numero de antenas utilizados y características del protocolo de comunicación.

Así mismo se realizó una comparación entre los sistemas de radio frecuencias Crazyradio y los sistemas WiFi utilizando un microcontrolador ESP8266 utilizando como métrica la velocidad para transmitir paquetes.

Por último se realizó el desarrollo de un software que facilitará el uso de drones Crazyflie 2.0 en enjambre, con esto se busca crear una comunicación entre el servidor y el dron para obtener información de su posición actual y su *setpoint*. Dentro de este punto también se incluyó el desarrollo de firmware que deserializar y añadira los datos del recibidos al filtro Kalman así como al controlador dentro del Crazyflie 2.0.

Entra las limitaciones más relevantes que afectó el desarrollo del trabajo está la dificultad del desarrollo dentro del firmware del Crazyflie, las pocas unidades del Crazyflie que se tienen a disposición, las fallas en algunos componentes de los mismo así como el poco tiempo de vuelo por su pequeña batería.

CAPÍTULO 6

Marco teórico

6.1. Sistema de captura de movimiento

Un sistema de captura de movimiento hace referencia al grupo de tecnologías que permiten grabar el movimiento de un objeto u persona para transferir la información a otras aplicaciones. [4].

Existen diversas técnicas de captura de movimiento, donde las más utilizadas son:

1. Técnicas ópticas pasivas
2. Técnicas ópticas activas
3. Técnicas sin marcadores
4. Técnicas inerciales

Los sistemas ópticos se basan en marcadores, en esta técnica se cuenta con un conjunto de cámaras calibradas para rastrear el movimiento de estos, que pueden ser reflectantes, un marcador pasivo, o que pueden ser auto iluminados, marcador activo.

6.2. OptiTrack

OptiTrack es un proveedor de sistemas de captura de movimiento con el mismo nombre. Esta compañía ofrece tanto cámaras especializadas como el software que registra los datos. El hardware comprado por el Departamento de Ingeniería Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala, proviene de esta compañía y se compone de las siguientes cámaras.

La cámara $Prime^x41$, adquirida por la universidad tiene las siguientes especificaciones:

Figura 3: Cámara $Prime^x41$

- Resolución: 2048x2048
- Velocidad de fotogramas: 180 Hz.
- Latencia: 5.5 ms.
- Precisión 3D: +/- 0.10 mm.
- Rango para Marcadores Pasivos: 30 m
- Rango para Marcadores Activos: 45 m
- Anillos de LEDs:
  - 20 LEDs
  - Luces del tipo infrarrojo de 850 nm.
- Conexiones:
  - Puerto de datos GigE.
  - Sincronización de la cámara por medio de Ethernet.
  - Alimentación por medio de PoE o PoE+.
- Tamaño: 12.6cm x 12.6 cm x 13.2 cm
- Peso: 1.36 Kg.

6.3. Software Motive

Motive es un programa diseñado para controlar sistemas de captura *OptiTrack* para utilizar diversas aplicaciones de rastreo. **Motive** permite al usuario calibrar y configurar el sistema al mismo tiempo que se puede utilizar par recolectar y procesar información en 3D.

Motive obtiene toda la información 3D por medio de un proceso denominado **reconstrucción**, el cual compila múltiples imágenes en 2D de los marcadores para deducir las coordenadas en 3D. Este proceso permite al software extraer 6 grados de libertad, 3 de posición y 3 de orientación, de cualquier cuerpo rígido y esqueleto consiguiendo datos sobre complejos movimientos en el espacio para ser grabados o transmitidos en tiempo real hacia otras aplicaciones. [5]

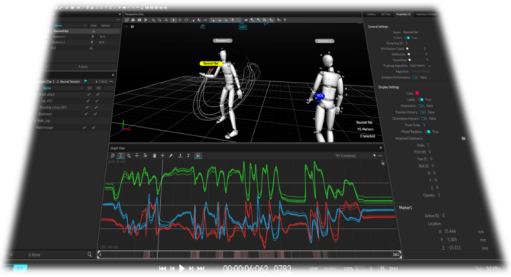

Figura 4: Interfaz de *Motive* [5]

6.4. Generalidades de un cuadricóptero

En general un dron posee un armazón, un sistema de propulsión y un sistema de comando y control. Un cuadricóptero es un tipo de helicóptero de simple diseño, el cual se compone de 4 rotores. [6]

Figura 5: Cuadricóptero [6]

Su cuerpo se puede colocar en dos configuraciones, las cuales se muestran a continuación

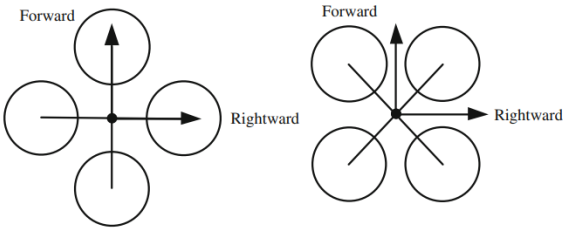

Figura 6: Configuración del armazón [6]

6.5. Ángulos de orientación de un cuadricóptero

Los ángulos de Euler son un forma intuitiva de representar la orientación en un dron. El teorema de Euler se basa en que, la rotación de un cuerpo rígido alrededor de un punto fijo puede ser interpretado como la composición de varias rotaciones finitas alrededor del mismo punto. [6].

Los ángulos de Euler se definen como

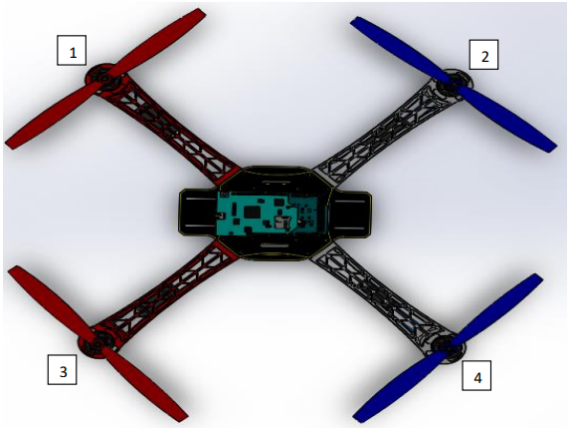

Figura 7: Numeración de los 4 motores de un cuadricóptero [7]

- **Ángulo de cabeceo θ (*pitch*)** Este movimiento es muy similar al roll y se genera mediante el aumento (o disminución) la velocidad de los motores del frente 1 y 2, y por la disminución (o aumento) de los motores de atrás 3 y 4 (ver Figura 7). El par de torsión con respecto al eje hace girar el cuadricóptero. Este desequilibrio de fuerzas sólo conduce a una aceleración en el ángulo de cabeceo [7].

- **Ángulo de alabeo ϕ (*roll*)** Este movimiento es proporcionado por el aumento (o disminución) de velocidad de los motores de la izquierda 1 y 3 y por disminución (o aumento) de la velocidad en los motores de la derecha 2 y 4 (ver Figura 7). Esto produce un par de torsión con respecto al eje que hace girar el cuadricóptero. Este desequilibro de fuerzas sólo conduce a una aceleración en el ángulo de alabeo [7].

- **Ángulo de guiñada ψ (*yaw*)** Este movimiento es proporcionado por el aumento (o disminución) velocidad de los motores 1 o 2 y por la disminución (o aumento) de los motores 3 o 4 (ver Figura 7). Esto conduce a un par de torsión con respecto al eje que hace girar el cuadricóptero. El movimiento de guiñada se genera gracias a que las hélices de izquierda y derecha giran en sentido horario mientras que las hélices delantero-trasero giran en sentido antihorario. Este movimiento sólo conduce a una aceleración de ángulo de guiñada [7].

6.6. Crazyflie 2.0

El Crazyflie 2.0 es un dron miniatura de codigo abierto desarrollado por Bitcraze AB, con el objetivo de crear un crear un cuadricóptero pequeño, versatil y de bajo costo para la investigación y educación.

Figura 8: Crazyflie 2.0 [8]

6.6.1. Características:

1. Diseño duradero.
2. Fácil de ensamblar y no requiere soldadura.
3. Soporte para actualizaciones de firmware inalámbricas.
4. Carga a bordo a través de uUSB estándar.

6.6.2. Especificaciones mecánicas:

1. Masa de despegue de 27g.
2. Tamaño: 92 mm x 92 mm x 92 mm (medidas de rotor a rotor).

6.6.3. Especificaciones de radio:

1. Radio de banda ISM de 2.4 GHz.
2. Amplificador de radio de 20 dBm probado a más de 1 km de distancia con Crazyradio PA.
3. Compatibilidad con Bluetooth Low Energy con clientes iOS y Android disponible.
4. Radio compatible con versiones anteriores de Crazyflie Nano y Crazyradio originales.

6.6.4. Microcontroladores:

1. MCU de aplicación principal STM32F405 (Cortex-M4, 168MHz, 192kb SRAM, flash de 1Mb).

2. MCU de administración de energía y radio nRF51822 (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash).
3. Conector USB.
4. Cargador LiPo a bordo con modos 100mA, 500mA y 980mA disponibles.
5. Capacidad USB OTG parcial.
6. EEPROM de 8 KB.

6.6.5. Modelo matemático del *Crazyflie 2.0*

El análisis asume los siguiente supuestos:

- El cuadricóptero es un cuerpo rígido.
- La masa y los momentos de inercia son constantes,
- El centro geométrico y el centro de gravedad del cuadricóptero es el mismo.
- El origen del sistema de referencia del dron está situado en el centro de masa del mismo.

Como siguiente paso se define el marco de referencia inercial O_{FI} y el marco de referencia del dron, como se muestra en la Figura 9

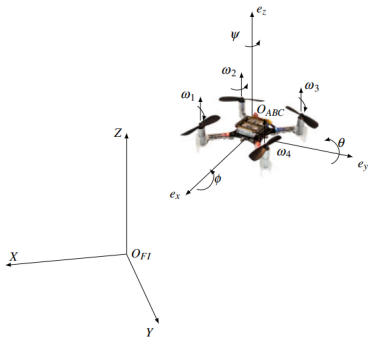

Figura 9: Marco de referencia en el Crazyflie con el marco de referencia inercial [9]

Por lo tanto, el movimiento del dron se puede expresar de la siguiente manera. [7]

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad q = \begin{bmatrix} \xi \\ \eta \end{bmatrix} \tag{1}$$

En este sistema las velocidades lineales son determinadas por V^B [m/s] y las angulares por w^B [rad/s]. Por lo que se obtén el siguiente vector de velocidades.

$$\vec{v} = \begin{bmatrix} V^B & w^B \end{bmatrix}^T = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T \tag{2}$$

Y podemos expresar la matriz de rotación del sistema de referencia respecto al sistema de referencia inercial de la siguiente forma. [7]

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi C_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi C_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi - C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \tag{3}$$

Donde $C_x = cos(x)$, $S_x = sin(x)$ y R es una matriz ortogonal.

También es posible mapear las velocidades angulares desde el sistema de referencia inercial al marco de referencia del cuadricóptero, mediante la matriz de transformación W_η.

$$\vec{v} = W_\eta \dot{\eta} \Rightarrow \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4}$$

Y para obtener las velocidades angulares desde el marco de referencia al marco de referencia inercial.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & -S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{5}$$

6.6.6. Ecuaciones de Newton - Euler

Definidas tanto la velocidad linear como angular en ambos marcos de referencia el siguiente paso es determinar los actuadores del dron y su relación con la dinámica del sistema. Según [10] la fuerza de empuje que genera el cuadricóptero se define como:

$$F_z = C_T(w_1^2 + w_2^2 + w_3^2 + w_4^2) \tag{6}$$

Y los momentos de torsión, según [10], se definen mediante:

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} C_T d(-w_1^2 - w_2^2 + w_3^2 + w_4^2) \\ C_T d(-w_1^2 + w_2^2 + w_3^2 - w_4^2) \\ \sqrt{2} C_M(-w_1^2 + w_2^2 - w_3^2 + w_4^2) \end{bmatrix} \tag{7}$$

Donde d es la distancia desde el centro de masa del dron al eje del rotor, w_i es la velocidad angular del i-ésimo motor, C_T y C_M son las constantes de empuje y momento del rotor respectivamente.

6.6.7. Ecuaciones de Newton para la posición

Planteando las ecuaciones de Newton desde el marco de referencia inercial se obtiene la siguiente ecuación.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{F_z}{m} \begin{bmatrix} C_\psi S_\theta C_\phi + S_\psi C_\theta \\ S_\psi S_\theta C_\phi - C_\psi S_\theta \\ C_\psi S_\phi \end{bmatrix} \tag{8}$$

Donde m es la masa del cuadricóptero y g es la fuerza de gravedad.

6.6.8. Ecuación de Euler para la orientación

Como primer paso es necesario determinar la inercia del cuadricóptero, dado que el Crazyflie 2.0 es simétrico, la matriz se represente como:

$$I_dron = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{9}$$

Considerando la ecuación que describe la dinámica de orientación del dron, planteada desde el marco de referencia del cuadricóptero. [7]

$$I\dot{v} + v \times (Iv) + \Gamma = \tau \tag{10}$$

Donde $I\dot{v}$ es la aceleración angular de la inercia $v \times (Iv)$ son las fuerzas centrípetas, $+\Gamma$ son las fuerzas giroscópicas y τ el torque externo. Se despeja para $\dot{v}$ y se obtiene la siguiente expresión, descrita en [7].

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{(I_{yy}-I_{zz})qr}{I_{xx}} \\ \frac{(I_{zz}-I_{xx})qr}{I_{yy}} \\ \frac{(I_{xx}-I_{yy})qr}{I_{zz}} \end{bmatrix} - I_r \begin{bmatrix} \frac{q}{I_{xx}} \\ -\frac{p}{I_{yy}} \\ 0 \end{bmatrix} w_r + \begin{bmatrix} \frac{\tau_\phi}{I_{xx}} \\ \frac{\tau_\theta}{I_{yy}} \\ \frac{\tau_\psi}{I_{zz}} \end{bmatrix} \tag{11}$$

6.7. Bandas de radio ISM

Las bandas de frecuencia para aplicaciones industriales, científicas y médicas, ISM por sus siglas en inglés, están apartadas para a controlar equipos de manera local en donde su uso esta libre de cualquier cargo mientras no esté relacionado con el área de las telecomunicaciones. [11]

6.8. WiFi

El WiFi es una tecnología de red inalámbrica, la cual permite que diferentes dispositivos. Este hace referencia a la familia de estándares basados en la IEEE 802.11, que define el protocolo que permite la comunicación entre los dispositivos que la soportan [12].

Por medio de esta red se puede interactuar con Internet o solamente entre sí, permitiendo el intercambio de información y estableciendo así una red. Este se basa en ondas de radio, las cuales operan en las frecuencias de 2.4 GHz en el estándar 802.11 n y 5 GHz en el estándar 802.11 ac.

6.9. TCP/IP

Es un conjunto de protocolos con varias capas que define cuidadosamente como se mueve la información desde el remitente hasta el destinatario como se muestra en 10. TCP hace referencia a Transmission Control Protocolo, este protocolo reciben los datos de una aplicación, los dividen en partes más pequeñas llamadas paquetes, añaden una dirección de destino y, a continuación, pasan los paquetes a la siguiente capa.

IP o Protocol Internet convierte un paquete en un Datagrama: de IP, pone la cabecera y la cola de datagrama, decide dónde enviar el datagrama (directamente a un destino o a una pasarela) y pasa el datagrama a la capa de interfaz de red.

La capa de interfaz de red acepta los datagramas IP y los transmite como tramas a través de un hardware de red específico, por ejemplo redes Ethernet. [13]

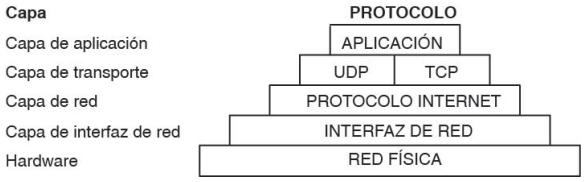

Figura 10: Conjunto de protocolos TCP/IP

6.10. JSON

JSON es un formato de texto ligero, independiente de algún lenguaje de programación, para transferir datos, este es fácil de leer y escribir. Un archivo JSON está formado por dos estructuras:

1. Una colección de nombres o valores, el cual depende del lenguaje utilizado y puede ser un diccionario, una tabla hash, una lista con llaves, un objeto o una estructura.
2. Una lista ordenada de valores, la que puede suele estar en forma de array, pero también puede ser un vector, lista o una secuencia.

6.11. FreeRTOS

Free Real time operating system o FreeRTOS, es un kernel de tiempo real, donde se pueden construir aplicaciones embebidas las cuales tengan requerimientos de tiempo. A diferencia de un kernel convencional en FreeRTOS, el desarrollador asigna a cada hilo una prioridad, la cual le permite al kernel decidir el orden de ejecución. [14]

6.12. Filtro Kalman

Las técnicas de fusión de datos combinan las lecturas de diferentes sensores e información relacionada para aumentar la precesión de las muestras. Esta técnica presenta una ventaja estadística al aumentar el numero de observadores independientes sobre un cuerpo.[15]

El filtro Kalman es una técnica popular utilizada para resolver problemas de observadores en la ingeniería de control. El filtro Kalman es un modelo recursivo en el cual se pueden identificar dos procesos diferentes, el proceso de predicción y el proceso de corrección. En la predicción el filtro se encarga de estimar el estado actual del agente utilizando las ecuaciones de estado así como la predicción de la covarianza del error.

La etapa de corrección se encarga de utilizar sensores para obtener información del estado actual del agente, con estas lecturas se puede corregir la ganancia de Kalman y actualizar el estado estimado mejorando la precisión de los resultados. Por último se corregí la covarianza para volver a repetir el proceso. [16]

CAPÍTULO 7

Adaptación de librería Crazyswarm

Como parte de la investigación de trabajos anteriores se encontró el paquete de herramientas de software Crazyswarm. Este consiste en un conjunto de librerías para transformar una computadora en un torre de control de drones Crazyflie. Para comenzar con su instalación se siguieron los pasos especificados en su documentación online.

Siguiendo estos pasos, se notó que dentro de la librería se tomaron varias decisiones así como se utilizaron varias dependencias que no encajaban con los requerimientos de diseño para el ecosistema Robotat.

Instalar el software fue complicado ya que se necesitaban versiones específicas de Ubuntu Linux así como de ROS, por recomendación de los investigadores se hizo el intento de instalar este sistema operativo directamente en el servidor dentro del laboratorio. Este proceso presentó problemas con los permisos otorgados al servidor por el departamento técnico de la Universidad por lo que se tomó la decisión de seguir el proceso en una maquina virtual.

Rápidamente se encontraron los primeros problemas en la instalación de dependencias de ROS, al revisar la documentación para resolver estos errores se explica que se utiliza ROS para manejar el envío de datos por medio de RF: hacía los Crazyflies.

Este descubrimiento fue importante, pues Crazyswarm depende fuertemente del uso de antenas Crazyradio, por lo que se comenzó a evaluar si las ventajas de utilizar este sistema eran suficientes para continuar con la instalación de dependencias. Los siguientes puntos resumen las consideraciones que fueron decisivas para dejar de utilizar el software:

- Se necesita un gran número de antenas por agente, una por cada quince drones, si se sigue el esquema de comunicación de Crazyswarm o una por cada cuatro drones si se necesita mantener un comunicación de dos vías abierta.
- Aumenta la complejidad de instalación y mantenimiento al utilizar ROS únicamente

para manejar un servidor de radio.

- Se necesita asignar manualmente los canales que utilizará cada dron dentro de la configuración y que esta coincida con la configuración dentro de Crazyswarm.
- Hay que precargar con una trayectoria a los drones antes de despegar y no existe soporte para hacer cambios después del despegue.
- Es una comunicación de una vía por lo que no hay soporte para recibir datos del dron después del despegue.
- No hay una forma de asegurar que los paquetes con comandos lleguen a los drones, sino que los mensajes se envían varías veces hasta que sucede un nuevo evento o ocurra un *timeout*.

Por lo que fue necesario evaluar exclusivamente la opción de comunicación por WiFi.

CAPÍTULO 8

Desarrollo e implementación de una antena de comunicación WiFi

Para comenzar la evaluación de la comunicación WiFi, se vio en la necesidad de construir una antena de comunicación WiFi ya que el Crazyflie 2.0 no cuenta con ninguna modulo interno o externo para dotarlo con estas características. Por lo que este capítulo detalla el proceso para crear el modulo que se acopla al dron para establecer una vía de comunicación entre en el Crazyflie 2.0 con el ecosistema Robotat

8.1. Metodología

Para el desarrollo de una antena WiFi para el Crazyflie 2.0 se llevaron a cabo las siguientes tareas, con el objetivo de crear una módulo que respete las limitaciones de peso y tamaño del propio dron, pero al mismo tiempo sea posible su fabricación de manera local.

Con respecto a las restricciones de pesos se sabe que la capacidad máxima de empuje de los motores es 42 gramos, a esto se le tiene que restar los 27 gramos del peso del dron, por lo que se tiene un peso extra máximo de 15 gramos.

Para las restricciones de tamaño se tomó como guía la hoja de especificaciones sobre el *Breakout Deck* de Bitcraze, ya que aparte de ser del mismo fabricante, sus medidas coinciden perfectamente con el espacio libre sobre el volumen del dron. El tamaño de la placa visto de frente debe ser aproximadamente 28 mm de ancho y 26 mm de alto [17].

Para alimentar la antena se decidió conectar directamente a la batería del dron. Este cuenta con varios pines para proveer corriente, en este caso se escogió el pin VCOM, pin que conecta directamente con la batería de LiPo [17]. Por esta razón se tomó como referencia un voltaje máximo en el pin de 4.8V y un voltaje mínimo de 3.7V. Mientras que para la corriente se tiene como un máximo de 380 mAh a 3.7V, de esto se puede deducir que el dron

puede proveer una potencia máxima de 1.4Wh, por lo que su corriente cuando la batería esta al máximo es de aproximadamente de 290 mA y cuando se encuentra en su minimo es de 380 mA.

8.1.1. Selección de componentes

El primer paso para la selección de componentes fue encontrar conectores para el Crazyflie 2.0. Este utiliza un tipo poco común del cual no se tenía aviso alguno por parte del fabricante. Después de revisar la documentación de diferentes placas adicionales que ofrece Bitcraze se llegó al siguiente componente, un conector tipo hembra de 10 pines con las siguientes medidas:

- Largo: 20 mm
- Alto: 2 mm
- Ancho: 3 mm
- Espacio entre pines: 2mm

Como siguiente paso se escogió el microcontrolador a utilizar, para esto se tomaron como factores de selección el tamaño, peso, alimentación, conectividad WiFi y disponibilidad local.

Después de revisar la proveedores locales se llegó a la conclusión que un buen candidato sería un microcontrolador ESP8266, en especifico la *board* Wemos D1 mini. Este tiene las siguientes características [18]:

- Dimensiones: 35 x 26 x 12 mm
- Peso: 6 gramos
- Voltaje: 5 - 3.3 V
- Corriente Promedio: 70 mA

De esto se puede concluir que tanto en dimensiones como en peso el Wemos D1 mini cumple con todos los requisitos encontrados apartir del dron además es común de manera local y cuenta con una antena WiFi.

8.1.2. Diseño electrónico y fabricación

Para el diseño PCB se utilizo un software EDA. A continuación se muestra el esquemático de la placa, el cálculo del ancho de los *tracks* y los componentes utilizados.

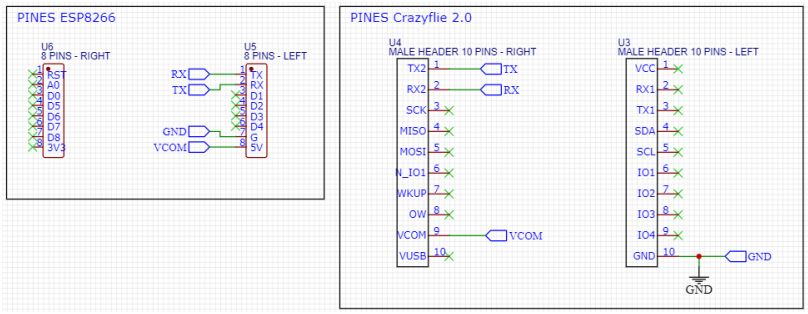

Figura 11: Esquemático de Deck Crazyflie.

Como muestra la Figura 12, utilizando la norma ANSI IPC-2221 se encontró que se requería un mínimo de ancho de 0.09 mm. Este resultado, sin embargo, fue demasiado pequeño para lo que se puede manufacturar con las máquinas de la Universidad, por lo que se optó por escoger un ancho de 0.508 mm. Por otro lado, el espacio entre pistas seleccionado fue el calculado, con un valor de 0.61 mm.

ANSI PCB TRACE WIDTH CALCULATOR

Input Data			Results Data				
				Internal Traces		External Traces	
Field	Value	Units	Trace Data	Value	Units	Value	Units
Current (max. 35A)	400	mA	Required Trace Width	0.23	mm	0.09	mm
Temperature Rise (max. 100°C)	10	°C	Cross-section Area	0.01	mm²	0	mm²
Cu thickness	1	oz/ft²	Resistance	0.12	Ω Ohms	0.32	Ω Ohms
Ambient Temperature	30	°C	Voltage Drop	0.05	Volts	0.13	Volts
Conductor Length	2	inches	Loss	0.02	Watts	0.05	Watts
Peak Voltage	5	Volts	Required Track Clearance	0.61	mm		

Figura 12: Cálculos de ancho de pistas.

Con las siguientes restricciones se obtuvo como resultado la PCB que se muestra en la Figura 18. Como resultado de la fabricación se obtiene la siguiente PCB 13.

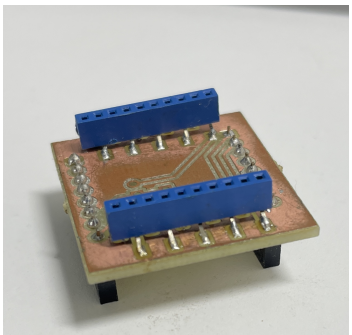

Figura 13: PCB Física.

Que junto al micontrolador Wemos D1 mini tiene un peso de 10.34 gramos como se muestra en 21 20 19.

8.2. Desarrollo de firmware para ESP8266

Con el hardware terminado, se prosiguió con la tarea de crear la conexión entre el servidor del ecosistema Robotat con el ESP8266. Para completar este objetivo se diseñó un nuevo firmware utilizando librerías comunes para la plataforma del microcontrolador.

8.2.1. Metodología

Para comenzar con el desarrollo de firmware para el ESP8266 primero se revisó los requerimientos del servidor del ecosistema. Este trabaja con un protocolo TCP, por donde se transmiten paquetes JSON. Con esto en mente se buscan librerías tanto para conectarse al servidor como para manejar archivos JSON en C. Se utiliza el puerto UART para establecer comunicación con el dron.

8.2.2. Código principal del firmware

Para la establecer la conexión TCP, se utilizó la librería ESP8266WiFi, esta viene incluida con las herramientas de desarrollo de ESP y se siguieron los siguientes pasos crear la conexión con el servidor. Primero se importa la librería de la siguiente forma.

```
#include <ESP8266WiFi.h>
```

Listing 8.1: Librerias WiFi ESP8266

Se declaran las variables para conectarse al servidor *Robotat* así como el objeto *WiFiClient* que se encarga de manejar los mensajes entrantes del servidor.

```
const char* ssid     = "Robotat";
const char* password = "iemtbmcit116";
const char* host     = "192.168.50.200";
const uint16_t port  = 1883;
WiFiClient client;
```

Listing 8.2: Variables WiFi

Se implementan las siguientes funciones para establecer una conexión WiFi y una conexión con el servidor respectivamente.

```
startWifi(ssid, password);
connectTCP(host, port);
```

Listing 8.3: Funciones WiFi

Con la conexión terminada se puede continuar con la recepción y la deserialización de un archivo JSON. Para conseguir este objetivo se instaló la librería *ArduinoJson* y se incluye en el script de la siguiente manera.

```
#include <ArduinoJson.h>
```

Listing 8.4: Libreria ArduinoJson

Para recibir el mensaje del servidor se utilizó la siguiente función, que permite leer el buffer continuamente hasta que encuentra una coincidencia con un carácter de nuestro interés, en este caso un *"}"*. Ya que el último carácter que se lee es eliminado, se termina agregando después de terminar de leer el buffer para poder deserializar el JSON correctamente.

```
String line = client.readStringUntil('}');
line += "}";
```

Listing 8.5: Función de lectura WiFi

Como último paso se verifica que el archivo se transmitió sin errores, si el paquete está listo para ser transmitido al Crazyflie entonces se puede mandar a llamar al *Handler* del paquete para cambiar su estructura y así posteriormente ser enviado al dron.

```
DeserializationError err = deserializeJson(doc, line);
if(err.code() == DeserializationError::Ok){
    HandlerPosePackage();
    sendToCF(packet);
    canSend = false;
}
```

Listing 8.6: Función de deserealización

La lógica detras de la función *sendToCF* se muestra en 14.

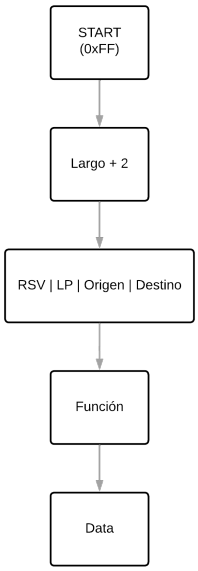

Figura 14: Lógica SendToCF

Como resultado del desarrollo del firmware se obtuvo el siguiente diagrama que explica la lógica detrás del *script* para el microcontrolador.

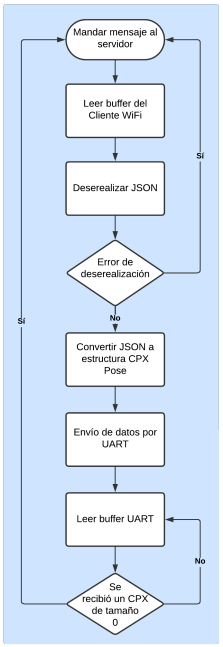

Figura 15: Lógica ESP8266

8.2.3. Latencia en la respuesta del servidor

Para medir el tiempo que toma obtener una respuesta del servidor TCP, se modificó el software del ESP8266, desactivando todas las funciones de comunicación con el dron. Después se configuró un pin GPIO: para desactivarse cada vez que se encuentra esperando una respuesta y se activara cada vez que esta procesando la respuesta.

Se realizaron 10 pruebas en total para medir la velocidad obteniendo los siguientes resultados.

No. de prueba	Tiempo Máx. (ms)	Tiempo Mín. (ms)
1	16.0	5.6
2	45.6	NA
3	25.6	4.0
4	8.8	4.8
5	9.6	4.8
6	7.2	4.8
7	12.8	4.0
8	14.4	6.4
9	8	4.0
10	18.4	4.0

Cuadro 1: Resultados de pruebas de latencia

Los resultados de las pruebas se pueden encontrar en [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]. Como referencia la latencia para un agente utilizando Crazyswarm oscila entre los 8 ms a los 11 ms.[3]

CAPÍTULO 9

Desarrollo de firmware para Crazyflie 2.0

Como última tarea en el ámbito de la comunicación, se necesitaba transmitir de forma efectiva los datos que serializa desde el ESP8266 hacía el estimador y controlador dentro del dron.

De la documentación sobre el estabilizador sabemos que consta de dos partes. La primera es la fusión de sensores, esta se hace por medio de un filtro Kalman extendido mientras que la segunda es un serie de 3 controladores PID en cascada los cuales compara el estado que le entrega el filtro Kalman con el estado deseado y traduce el error como cambios en la potencia de cada motor para cambiar la orientación y avance del dron.

9.1. Metodología

Para cumplir con la tarea se necesita activar el modulo de UART2, convertir los datos serializados en *floats* para guardarlos en sus variables correspondientes, introducir el estado actual del dron en el filtro Kalman y por ultimo introducir la posición deseada al controlador de los motores.

9.2. Configuración inicial del firmware

Como primer paso, para poder modificar el firmware se necesita descargar la máquina virtual de Bitcraze ya que esta viene precargada con todas las librerías y archivos necesarios para compilar el firmware así como para grabarlo dentro del dron. Para instalar la máquina virtual se utilizo el software de virtualización VM Virtualbox y se utilizaron los valores por defecto de la máquina.

Con la máquina virtual configurada, el primer cambio al firmware fue activar el filtro Kalman, ya que este viene desactivado por defecto. Este se activa utilizando una versión modificada de Kbuild, la cual permite hacer cambios en el firmware por medio de una interfaz gráfica, para entrar al menú se utiliza el siguiente comando.

```
$ make menuconfig
```

Listing 9.1: Comando menuconfig

Como siguiente paso se debe entrar a la pestaña de *Controllers y Estimators* y habilitar el estimador Kalman como se muestra en 22.

El siguiente paso es forzar el uso de CPX en el puerto UART2, para hacer esto se debe regresar al menú principal y entrar a la opción *Expansion deck configuration*. Dentro de esta pestaña se debe llenar la configuración *Expansion deck configuration* con el siguiente archivo *cpxOverUART2*. Por ultimo seleccionar la opción *Support wired external host using CPX on UART* y cambiar la configuración a *y*. El resultado de estos cambios se muestra en 23.

9.3. CPX y Router Interno

Crazyflie Packet eXchange es el protocolo interno que permite la comunicación entre los diferentes microcontroladores dentro del dron, así como la comunicación entre el dron y un agente externo. Este protocolo maneja el UART de manera automática, por lo que se decidió utilizar sus funciones para recibir los mensajes.

Para conseguir esto se modificaron los archivos *cpx_internal_router.c* y *cpx_uart_transport.c*. En el primero se declaro un nuevo *queue* con el nombre *robotatQueue* para manejar todos los archivos entrantes del UART, se modifico la función *cpxInternalRouterReceiveRobotat* que se encarga de recibir un puntero a una variable CPX y guardar el ultimo dato dentro de *robotatQueue* y se modifico la función *cpxInternalRouterRouteIn* que se maneja los datos entrantes y los guarda en sus respectivos*queue*. Se modifico la función *cpxInternalRouterInit* donde se inicializa *robotatQueue* al momento de encender el dron.

En el segundo archivo solamente se comentó, de la función *CPX_UART_RX*, la línea 9.2 que realiza una verificación CRC: pues no se logro comprender el calculo de esta función.

```
ASSERT(crc == calcCrc(&uartRxp));
```

Listing 9.2: Verificación CRC

Por último se creó un *queue* únicamente para almacenar todos los mensajes recibidos correctamente por medio de UART dirigidos a la función *Robotat*. Este *queue* es leída continuamente en el *loop* principal de la tarea para extraer los datos envíados.

9.4. Modificaciones al filtro Kalman y controlador

Por último se necesita agregar las variables de posición recién creadas al estimador, para hacer esto no se contaba con una documentación adecuada por parte de Bitcraze por lo que

se tuvo que hacer ingeniería inversa para la forma correcta de ingresar estos valores al filtro Kalman.

Después de leer varios código dentro del firmware se llegó a la siguiente función que permite ingresar los valores de sensores externos al dron en el filtro.

```
estimatorEnqueuePosition(&ext_pos);
```

Listing 9.3: Función enqueue del Estimador

Esta función recibe un puntero de una estructura, de la cual se tenía poco información. Volviendo a investigar dentro del firmware se encontró que la estructura estaba formada de las siguientes partes.

```
typedef struct positionMeasurement_s {
    union {
        struct {
            float x;
            float y;
            float z;
        };
        float pos[3];
    };
    float stdDev;
    measurementSource_t source;
} positionMeasurement_t;
```

Listing 9.4: Estructura Position

Donde x, y, z representan la posición en coordenas rectangulares del dron; *stdDevPos* representan la desviación estándar de la posición y *source* es una variable que registra de donde proviene la información.

Para colocar *setpoints* dentro del controlador se utiliza la siguiente función.

```
commanderSetSetpoint(&setpoint, COMMANDER_PRIORITY_EXTRX);
```

Listing 9.5: Función enqueue del commander

Con la cual se pueden colocar datos dentro del *queue* del *commander* para su ejecución. Esta función recibe dos parámetros, el primero es un puntero hacia una estructura *setpoints_t* mientras que el segundo es un *intenger* que representa la prioridad de los datos. Para obtener la mayor relevación dentro del commander se utilizo *Commander Priority Extrx* según se lo observado en ejemplos de Bitcraze.

La estructura tipo *setpint_t* tiene la siguiente forma.

```
typedef struct setpoint_s {
  uint32_t timestamp;

  attitude_t attitude;       // deg
  attitude_t attitudeRate;   // deg/s
  quaternion_t attitudeQuaternion;
  float thrust;
  point_t position;          // m
  velocity_t velocity;       // m/s
  acc_t acceleration;        // m/s^2
  bool velocity_body;        //

  struct {
    stab_mode_t x;
    stab_mode_t y;
    stab_mode_t z;
    stab_mode_t roll;
    stab_mode_t pitch;
    stab_mode_t yaw;
    stab_mode_t quat;
  } mode;
} setpoint_t;
```

Listing 9.6: Estructura setpoint_t

Esta estructura tiene la peculiaridad de no almacenar la información de un solo tipo de *setpoint*, si no que puede modificarse para ajustarse a la información que se tenga a disposición utilizando los diferentes modos que admite el *commander*.

Cabe resaltar que para ingresar datos al *commander* antes se debe iniciar en modo manual, para esto se debe ingresar un setpoint con todas sus variables en ceros. En la tarea *robotat.c* se hace de la siguiente manera.

```
setHoverSetpoint(&setpoint, 0.0, 0.0, 0.0, 0.0);
commanderSetSetpoint(&setpoint, COMMANDER_PRIORITY_EXTRX);
```

Listing 9.7: Ingreso al Modo Manual del Commander

9.5. Resultados

Como resultado de este capítulo se obtiene el siguiente *loop* que describe el funcionamiento de la tarea *robotat.c*.

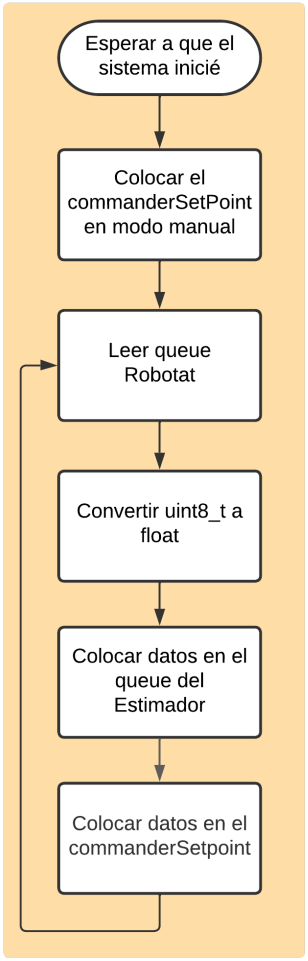

Figura 16: Lógica robotat.c

CAPÍTULO 10

Validación con Optitrack

En este capítulo se busca combinar los resultados conseguidos anteriormente y lograr que la información extraída del sistema de captura de movimiento llegue sin errores hacía el filtro Kalman del dron.

10.1. Metodología

En esta tarea se busca que el estado estimado del dron se encuentre lo más cercano a la información extraída del sistema Optitrack.

10.2. Frame de Crazyflie para Markers de Optitrack

El primer paso para lograr con este objetivo fue construir un *frame* en el cual se puedan colocar *markers* y así obtener la posición del Crazyflie de manera correcta. El primer paso fue crear un proyecto en Inventor donde guardar todos los archivos. Después se utilizó el modelo encontrado en [19] para crear el armazón y modificando las soportes de los *markers* para mejorar su solidez se llego al resultado mostrado en 17.

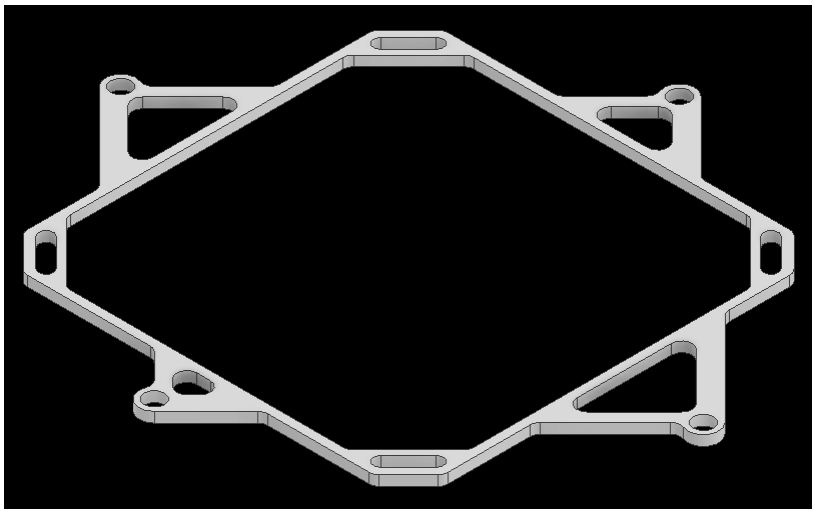

Figura 17: Frame Crazyflie

Este modelo fue fabricado utilizando impresión 3D, utilizando PLA como material. Y se obtuvo el siguiente resultado con un peso de 8.65 gramos como se muestra en 34

10.3. Resultados

Para corroborar el correcto funcionamiento del filtro Kalman, se hicieron pruebas moviendo de forma manual el dron sobre la plataforma de trabajo al mismo tiempo que utilizo el cliente de Bitcraze para graficar las datos que recibía el dron y el estado estimado.

Resultados						
No. de prueba	Optitrack (m)			Kalman (m)		
	X	Y	Z	X	Y	Z
1	0.71	0.04	1.18	0.71	-0.16	1.18
2	0.66	0.04	1.12	0.66	-0.14	1.12
3	1.06	0.52	1.14	1.07	0.37	1.14
4	1.01	0.98	1.02	0.99	0.81	1.06

Cuadro 2: Resultados de estimación con EKF

Resultados			
No. de prueba	Error (%)		
	X	Y	Z
1	0.00	5.00	0.00
2	0.00	4.50	0.00
3	0.00	0.29	0.00
4	0.00	0.17	0.04

Cuadro 3: Resultados de estimación con EKF

Como se muestra en 35, todos los aditamentos junto al dron superan el peso máximo de 42 gramos que es capaz de soportar el Crazyflie 2.0, por lo que al momento de hacer pruebas el dron tenía problemas tanto de equilibrio como para elevarse, esto impedía obtener resultados concluyentes pues el correcto funcionamiento del controlador depende de la dinámica del sistema al momento de volar acción que no se pudo llevar acabo.

CAPÍTULO 11

Conclusiones

- Se identificaron las ventajas y desventajas del paquete de herramientas *Crazyswarm*, donde se encontró que el implementarlo junto al ecosistema *Robotat* no cumplía con los requisitos del proyecto, por lo que se decidió por desarrollar una alternativa utilizando las herramientas dadas por Bitcraze y comunicación WiFi.
- Se desarrolló el hardware para una antena WiFi utilizando una placa Wemos D1 mini, la cual se puede construir con componentes locales que además cumple con las limitaciones del dron en cuanto tamaño, peso y alimentación.
- Se consiguió ingresar dentro del estimador de un Crazyflie 2.0 información de su posición obtenida por medio del sistema Optitrack logrando aproximar su estado correctamente.
- Se logró integrar correctamente al ecosistema Robotat un Crazyflie 2.0 utilizando un ESP8266 como deck WiFi obteniendo valores de latencia similares a la alternativa Crazyswarm.

CAPÍTULO 12

Recomendaciones

- Se recomienda migrar del Crazyflie 2.0 al Crazyflie 2.1, no sólo porque el modelo 2.0 se encuentra descontinuado sino también para mantener una homogeneidad entre los agentes a utilizar.
- Se recomienda fabricar la PCB de la antena con una empresa fuera de la universidad, para crear un modulo más liviano, barato y replicable así como con una mejor calidad de soldadura.
- Se recomienda utilizar el *frame* diseñado para crear una versión mejorada donde sea más fácil crear cuerpos únicos por medio del posicimiento distinto de los marcadores.
- Se recomienda hacer más pruebas sobre latencia al incrementar el número de agentes que a su vez aumenten el número de peticiones al servidor.
- Se recomienda hacer más pruebas físicas utilizando la función setpoint para comprobar su correcto funcionamiento.

CAPÍTULO 13

Bibliografía

[1] C. Perafán, «Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica,» Universidad Del Valle de Guatemala, Guatemala, ene. de 2021.

[2] F. Sanabria, «Diseño e implementación de una plataforma de pruebas para sistemas de control para el dron Crazyflie 2.0,» Universidad Del Valle de Guatemala, Guatemala, ene. de 2022.

[3] J. A. Preiss*, W. Hönig*, G. S. Sukhatme y N. Ayanian, «Crazyswarm: A large nano-quadcopter swarm,» en *IEEE International Conference on Robotics and Automation (ICRA)*, Software available at https://github.com/USC-ACTLab/crazyswarm, IEEE, 2017, págs. 3299-3304. DOI: 10.1109/ICRA.2017.7989376. dirección: https://doi.org/10.1109/ICRA.2017.7989376.

[4] *What is motion capture and how does it work?: Mo-Sys*, feb. de 2022. dirección: https://www.mo-sys.com/what-is-motion-capture-and-how-does-it-work/.

[5] *What is motion capture and how does it work?: Mo-Sys*, feb. de 2022. dirección: https://www.mo-sys.com/what-is-motion-capture-and-how-does-it-work/.

[6] Q. Quan, *Introduction to multicopter design and Control*. Springer Singapore, 2018.

[7] J. D. Guerra, «Control de actitud de un cuadricóptero,» Tesis doct., 2018.

[8] Dirección: https://www.bitcraze.io/documentation/hardware/crazyflie_2_0/crazyflie_2_0-datasheet.pdf.

[9] G. Silano y L. Iannelli, «CrazyS: A software-in-the-loop simulation platform for the Crazyflie 2.0 nano-quadcopter,» *Studies in Computational Intelligence*, págs. 81-115, 2019. DOI: 10.1007/978-3-030-20190-6_4.

[10] G. Silano y L. Iannelli, «CrazyS: A Software-in-the-Loop Simulation Platform for the Crazyflie 2.0 Nano-Quadcopter,» en *Robot Operating System (ROS): The Complete Reference (Volume 4)*, A. Koubaa, ed. Cham: Springer International Publishing, 2020, págs. 81-115, ISBN: 978-3-030-20190-6. DOI: 10.1007/978-3-030-20190-6_4. dirección: https://doi.org/10.1007/978-3-030-20190-6_4.

[11] International Telecommunication Union, *ARTICLE 1 - Terms and Definitions*, https://life.itu.int/radioclub/rr/art1.pdf, 2019.

[12] *What is wi-fi? - definition and types*, dic. de 2021. dirección: https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html#~q-a.

[13] Abr. de 2021. dirección: https://www.ibm.com/docs/es/aix/7.2?topic=protocol-tcpip-protocols.

[14] R. Barry, *Mastering the FreeRTOS Real Time Kernel*. Real Time Engineers Ltd, 2016.

[15] D. L. Hall, M. E. Liggins, J. Llinas y D. L. Hall, *Multisensor Data Fusion: Theory and Practice*. CRC Press, 2008.

[16] C. H. Lim, L. Y. Ong, T. S. Lim y V. C. Koo, «Kalman filtering and its real-time applications,» *Real-time Systems*, 2016. DOI: 10.5772/62352.

[17] Dirección: https://www.bitcraze.io/documentation/hardware/breakout_deck/breakout_deck-datasheet.pdf.

[18] Dirección: https://www.wemos.cc/en/latest/d1/d1_mini.html.

[19] P. Florence, «crazyflie-CAD,» *Github*, nov. de 2022. dirección: https://github.com/peteflorence.

CAPÍTULO 14

Anexos

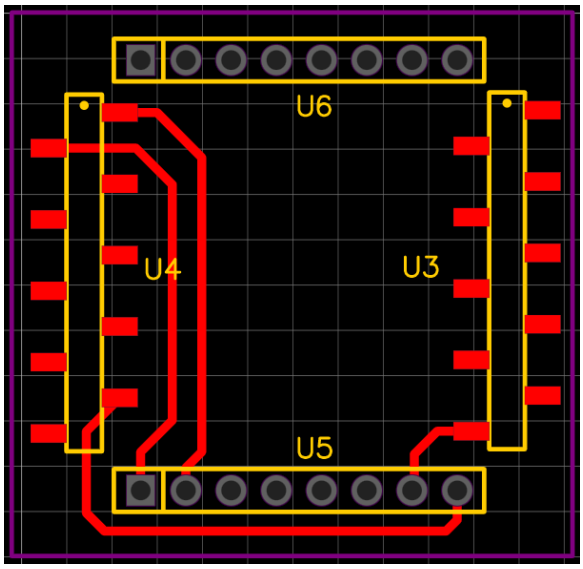

Figura 18: PCB Deck.

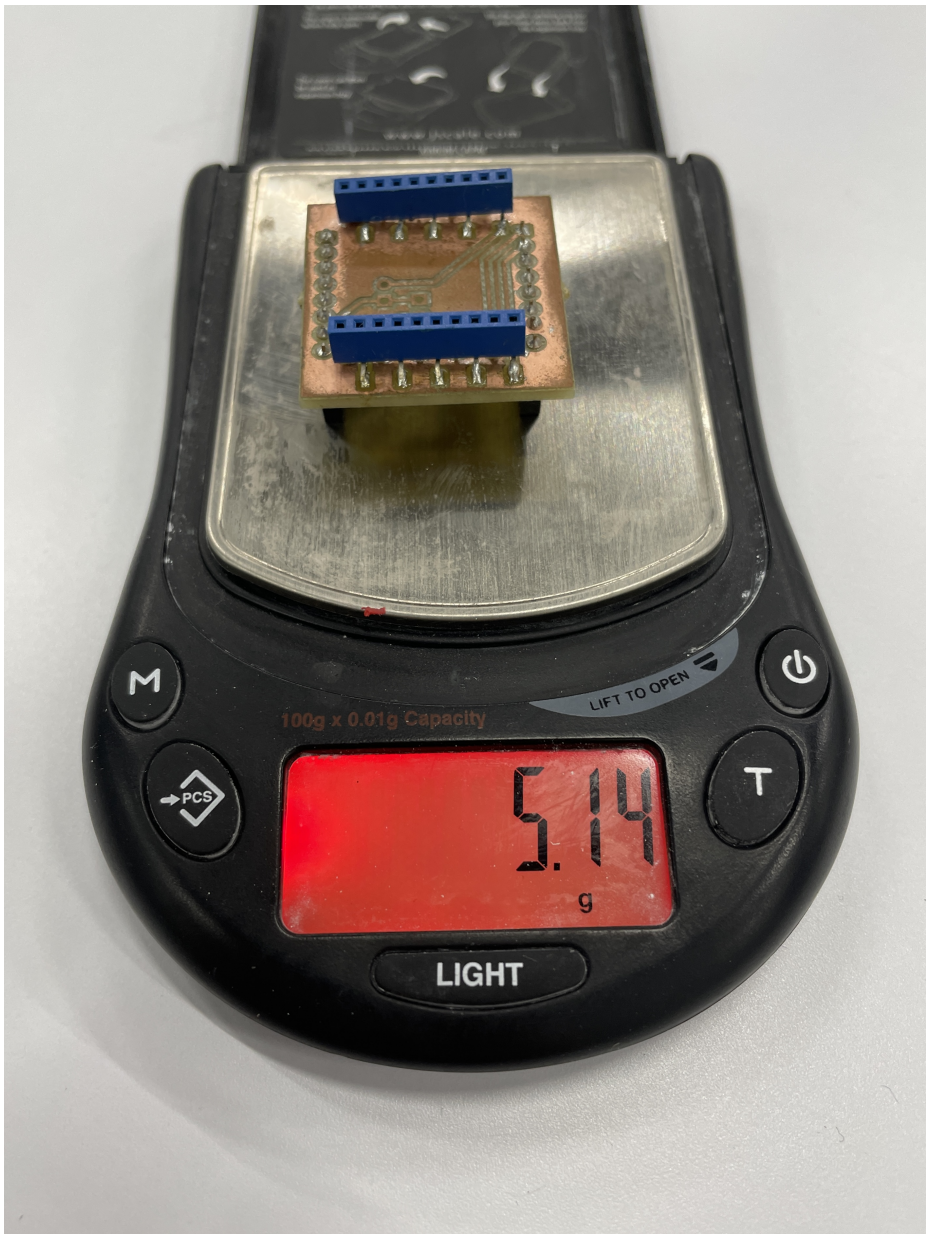

Figura 19: Peso Deck Crazyflie 2.0.

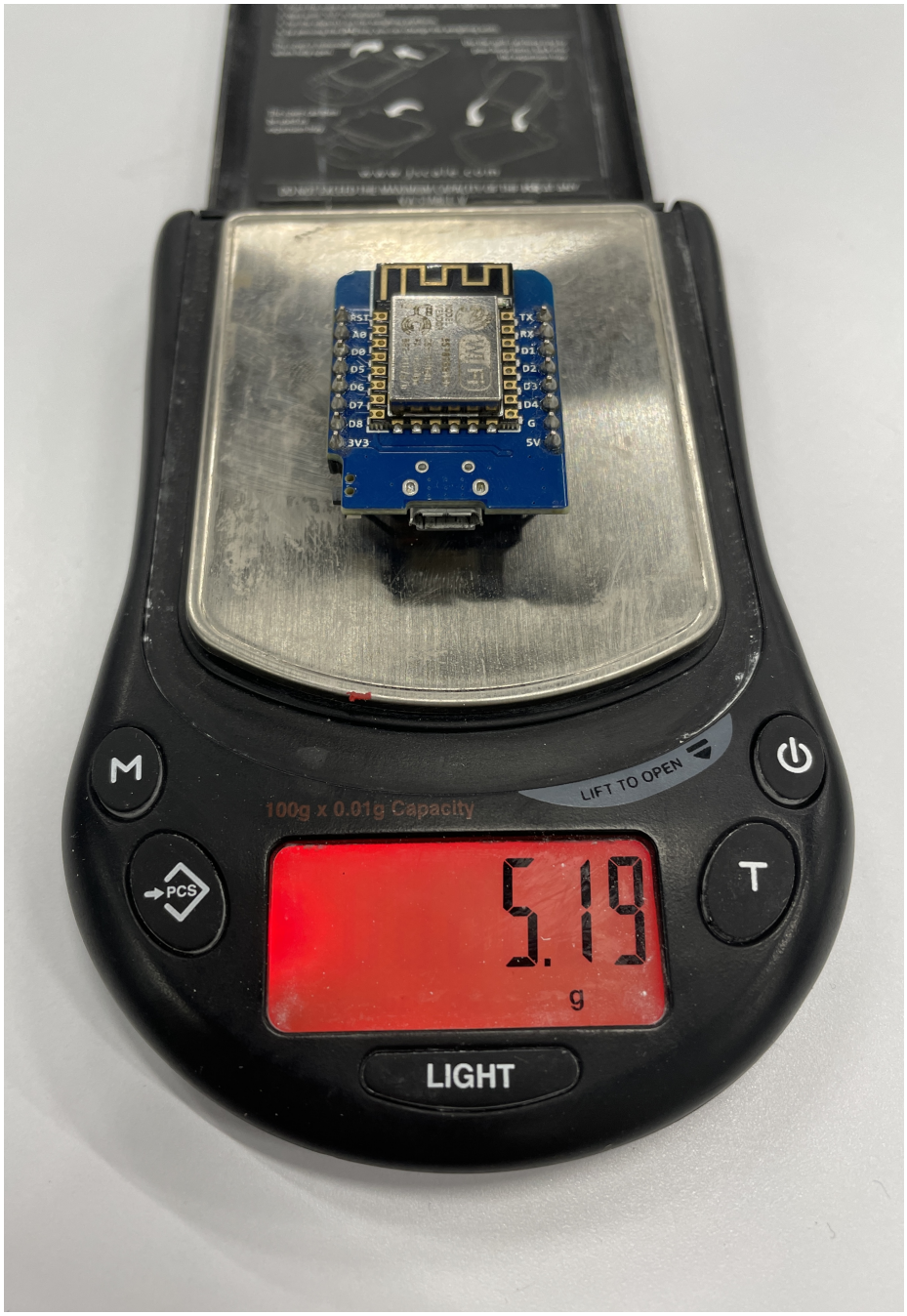

Figura 20: Peso Wemos D1.

Figura 21: Peso Deck y Wemos D1.

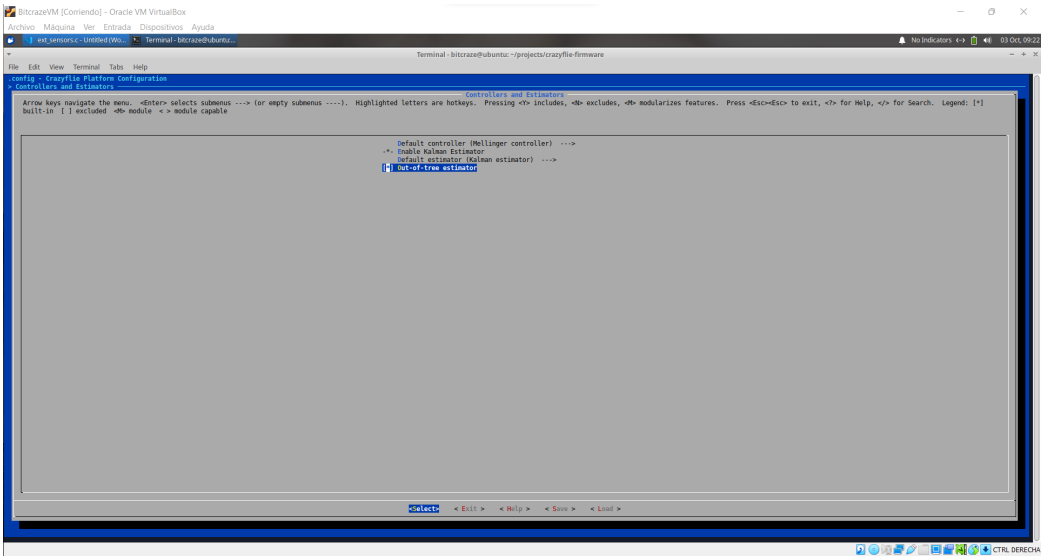

Figura 22: Configuración filtro Kalman en Kbuild

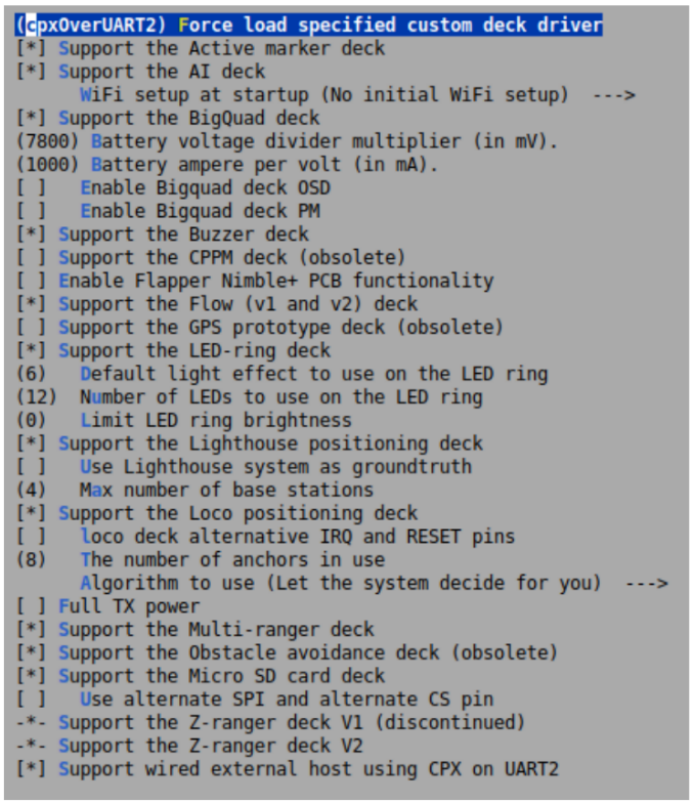

Figura 23: Configuración expansion deck configuration

Resultados de latencia.

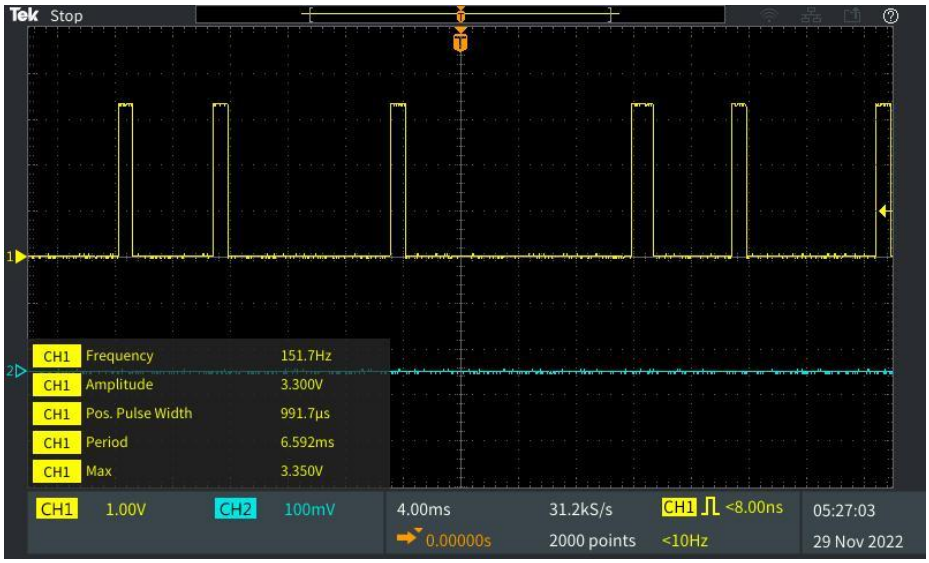

Figura 24: Resultados latencia

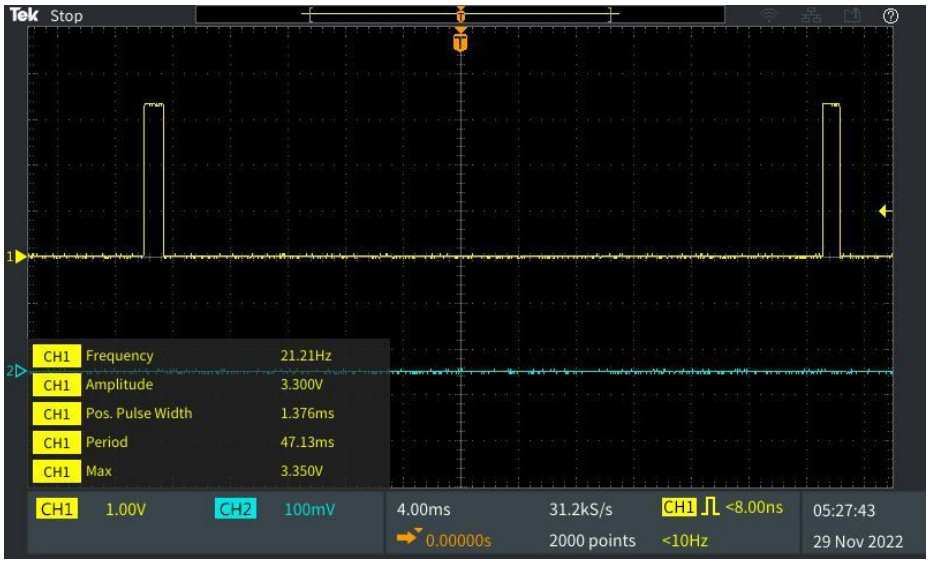

Figura 25: Resultados latencia

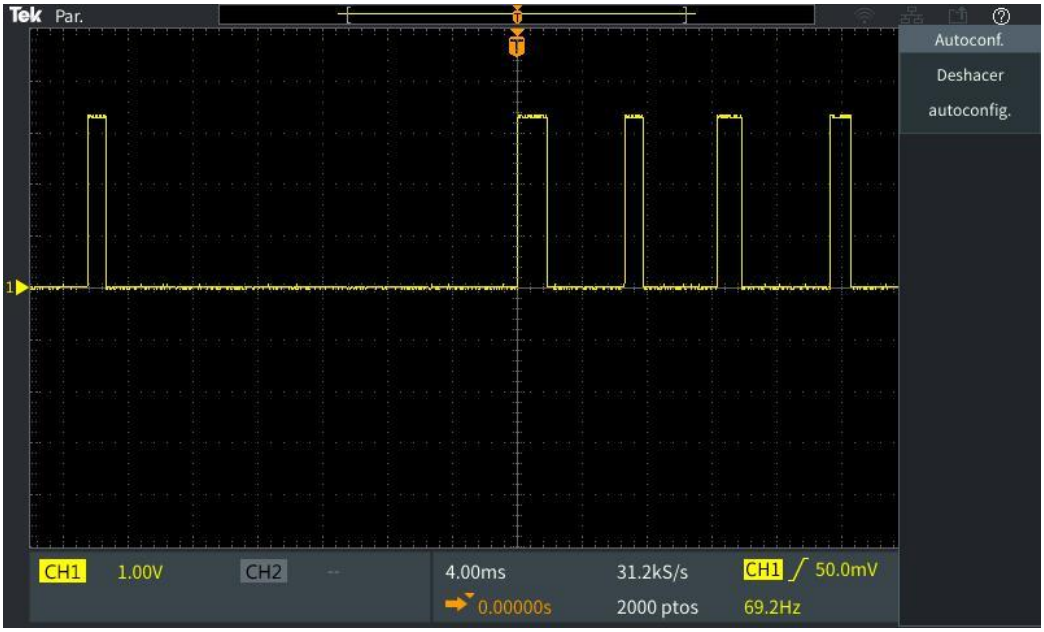

Figura 26: Resultados latencia

Figura 27: Resultados latencia

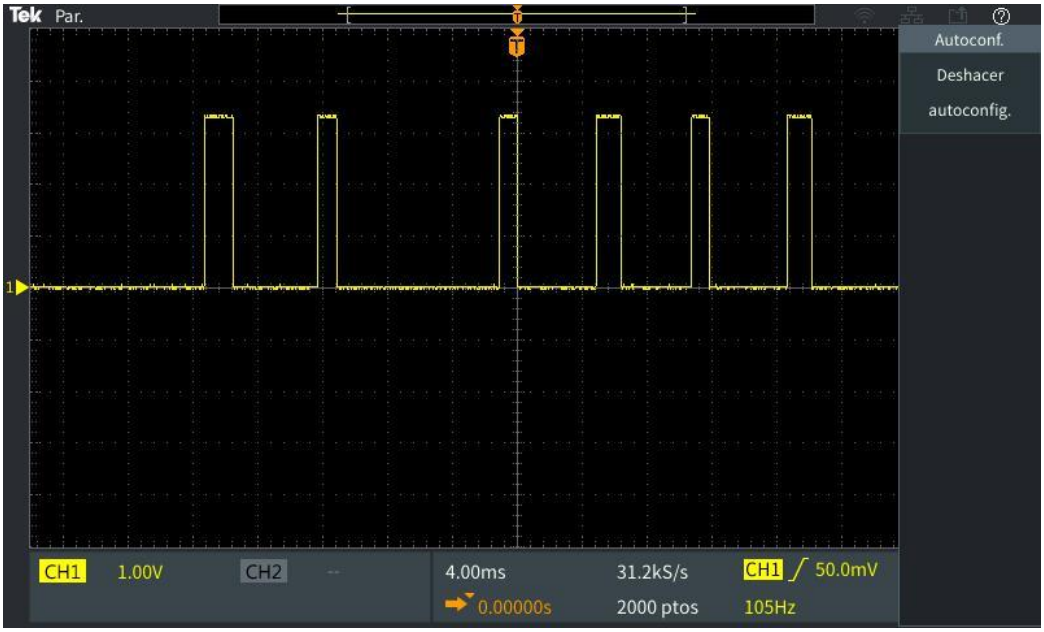

Figura 28: Resultados latencia

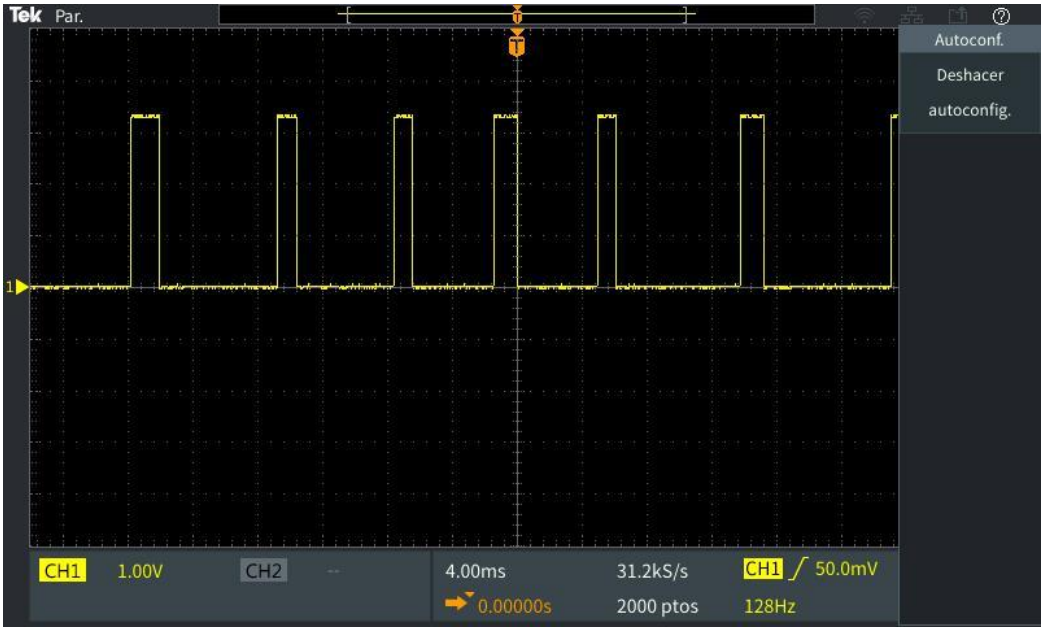

Figura 29: Resultados latencia

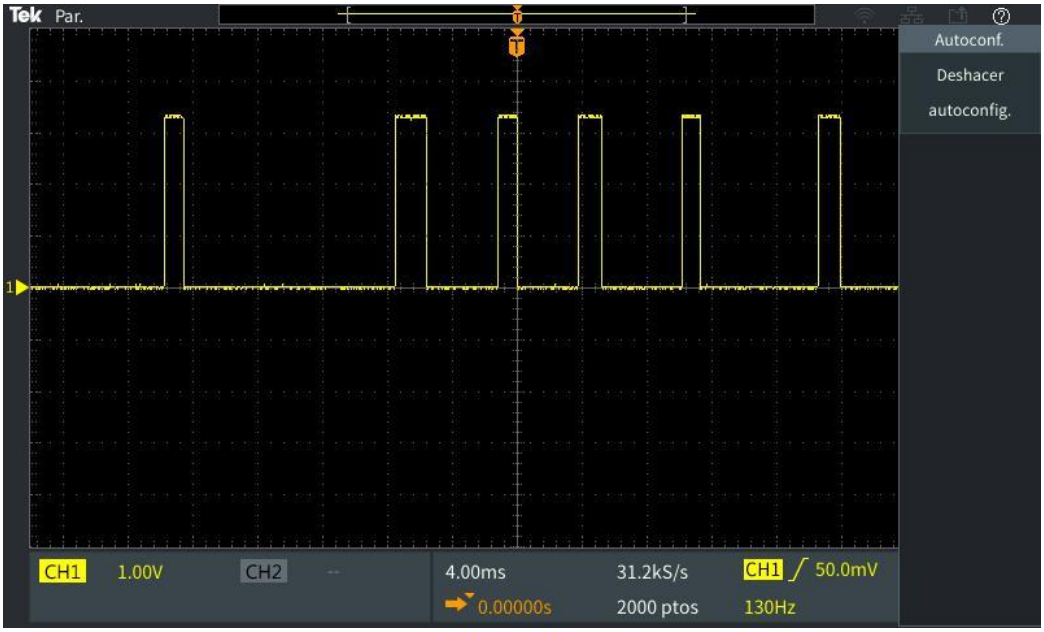

Figura 30: Resultados latencia

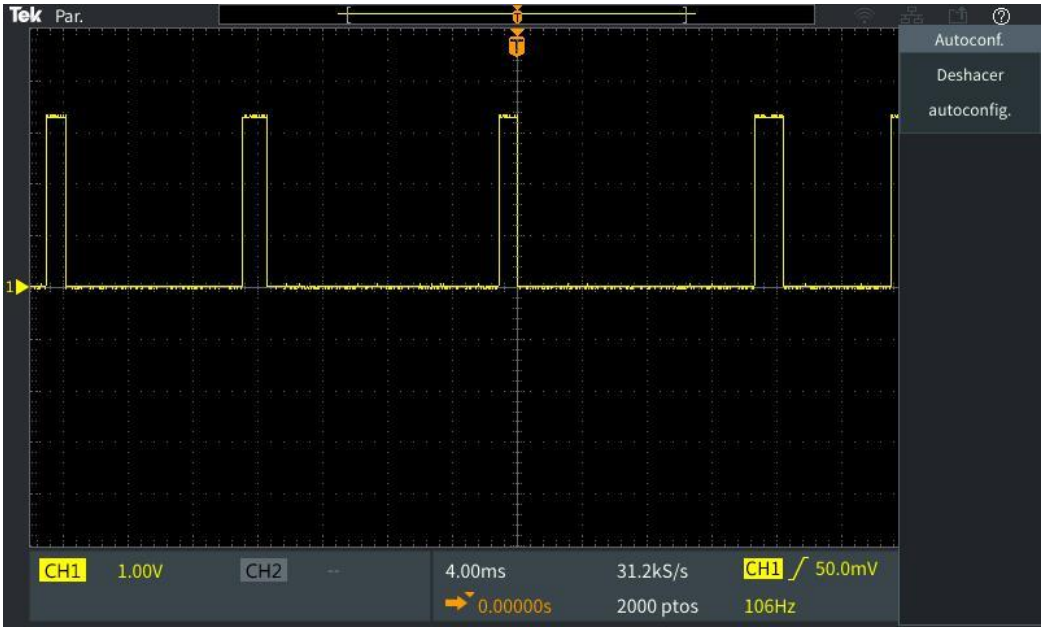

Figura 31: Resultados latencia

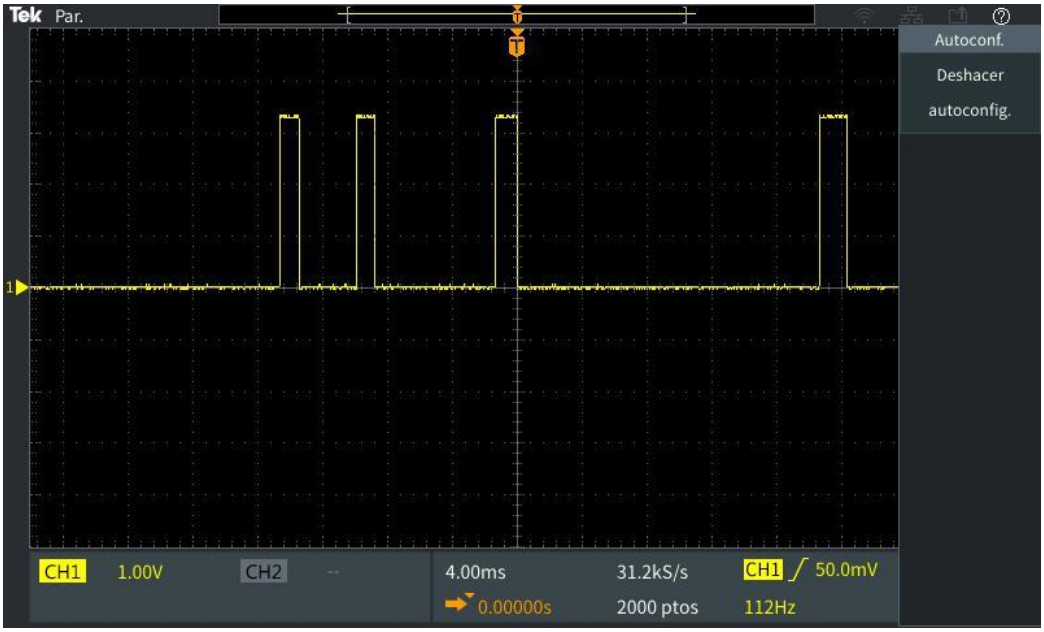

Figura 33: Resultados latencia

Figura 32: Resultados latencia

Figura 34: Peso de Frame para Crazyflie 2.0

Figura 35: Peso final Crazyflie 2.0

Resultados de filtro Kalman.

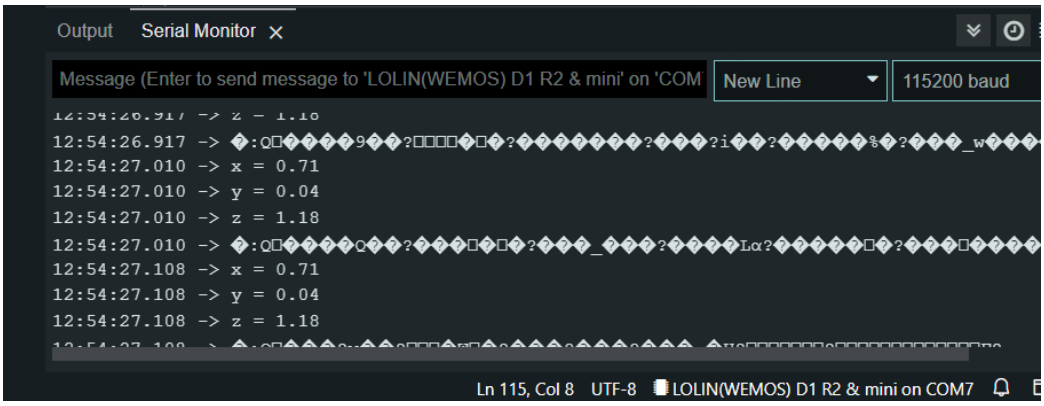

Figura 36: Resultados filtro Kalman

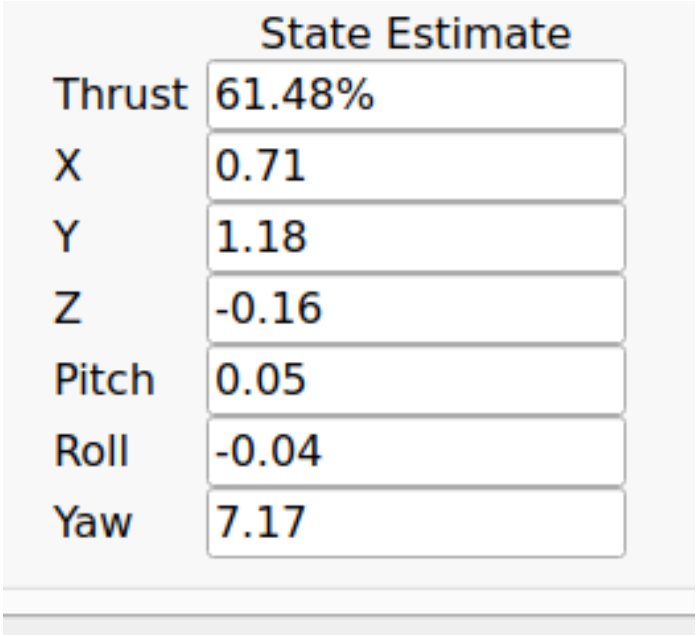

Figura 37: Resultados filtro Kalman

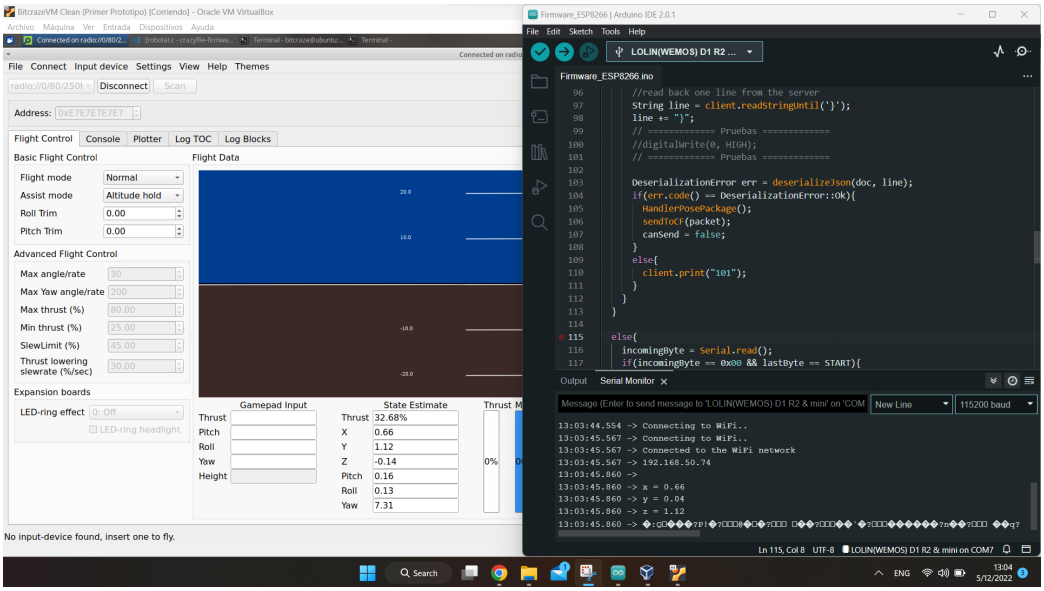

Figura 38: Resultados filtro Kalman

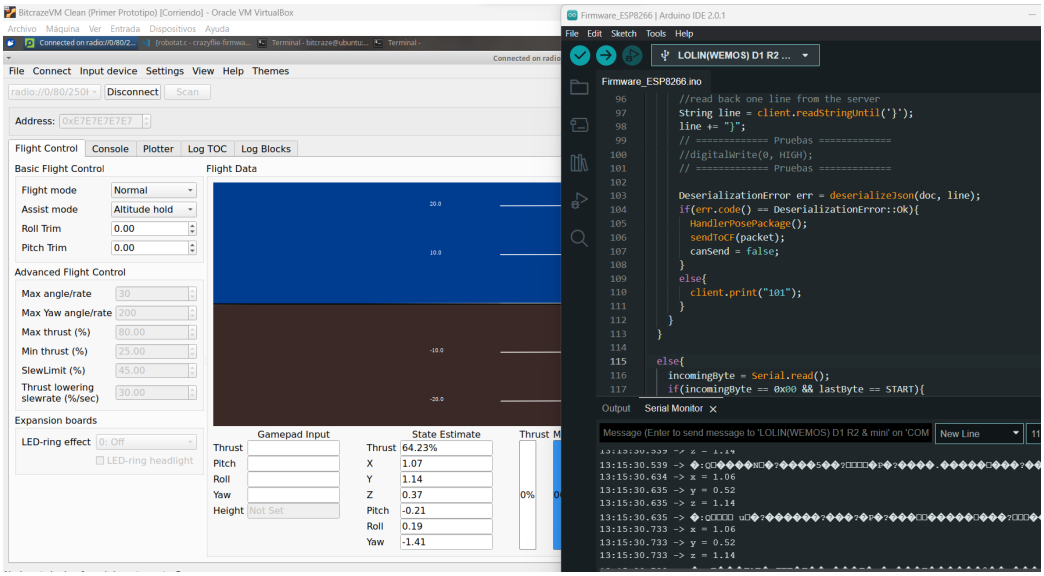

Figura 39: Resultados filtro Kalman

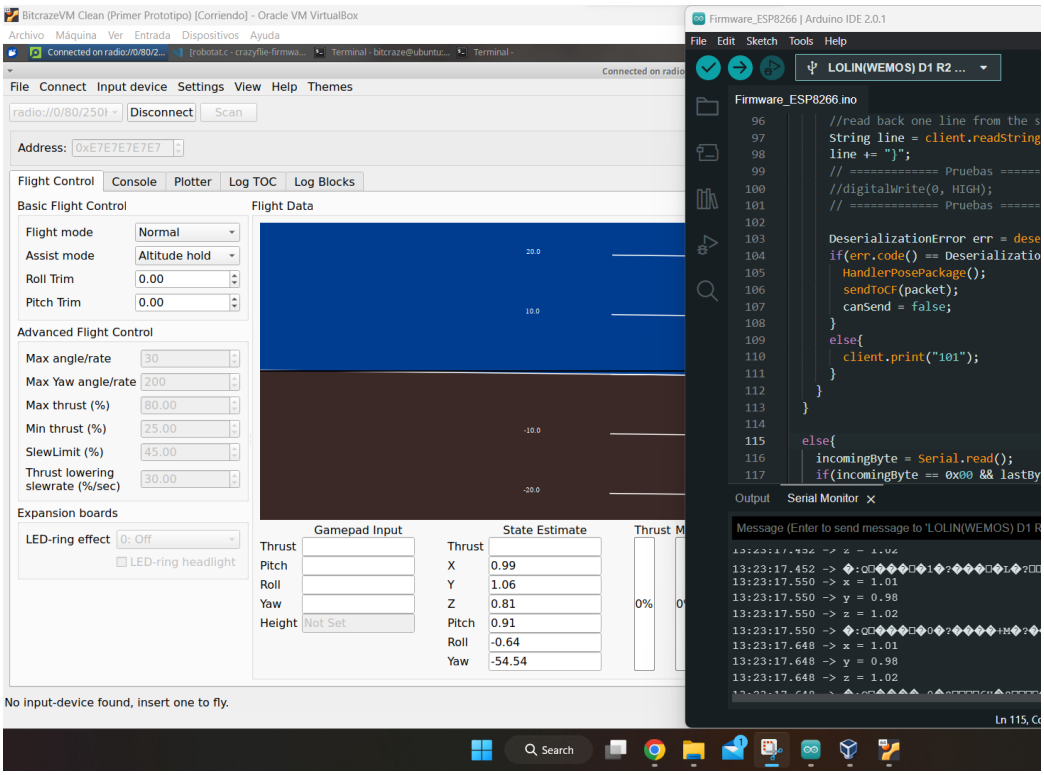

Figura 40: Resultados filtro Kalman

Todos los archivos relacionados con el desarrollo de este trabajo de graduación se encuentran en el siguiente repositorio:
https://github.com/EmilioG-18062/Robotat-Crazyflie2X

CAPÍTULO 15

Glosario

CRC: Cyclic Redundancy Check. 34

Datagrama: Unidad de transferencia básica asociada con una red de conmutación de paquetes. 21

GPIO: General Purpose Input Output. 30

RF: Radiofrecuencias. 23