Universidad Nacional de Rosario
Facultad de Ciencias Exactas Ingeniería y Agrimensura
Escuela de Ingenieria Electrónica

Informe proyecto final

"Desarrollo e implementación de un sistema de sensado y posicionamiento en prototipo de robot desmalezador"

AUTOR:
Mujica, Martin (M-6014/3)

DIRECTOR:
Kofman, Ernesto

DIRECTOR EXTERNO:
Pire, Taihú

fecha de entrega: –/–/–

2018

Índice

1. Introducción

1.1. Robot desmalezador

Este proyecto esta enmarcado dentro del desarrollo de un robot desmalezador, llevado adelante por el CIFASIS, Rosario.

El objetivo del proyecto global es desarrollar un robot móvil, autónomo y eficiente que pueda desplazarse por el campo, recorriendo los surcos siguiendo determinada trayectoria, y que al hacerlo sea capaz de detectar las malezas. Al detectar un objetivo deberá ser capaz de aplicar el producto químico de forma local, evitando de esta forma dañar el entorno u otras personas.

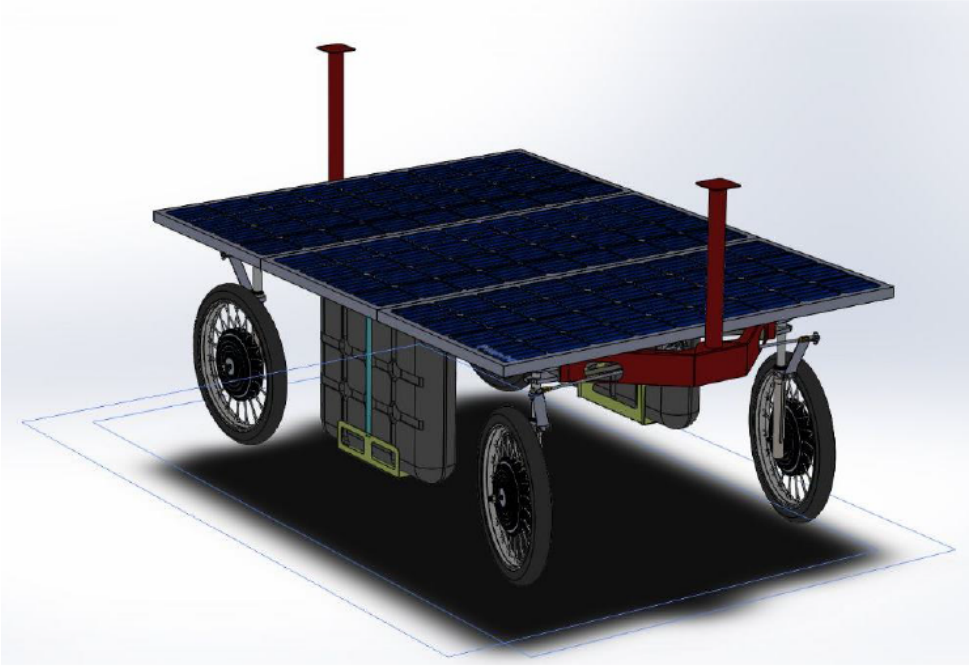


Figura 1: Esquema básico del robot desmalezador al comienzo de este trabajo.

1.2. Automatización en la agricultura

La agricultura es una de las más antiguas y relevantes industrias de la humanidad. Su automatización, con el objetivo de incrementar la productividad y liberar a las personas de tareas arduas e incluso comprometedoras para su salud, ha sido una de las lineas tradicionales de investigación y desarrollo en robótica.

Automatizar el trabajo agrícola cuenta con diversos desafíos, los cuales varían según el objetivo en cuestión. Sin embargo un problema importante y común a prácticamente todas las tareas es la localización del robot en su entorno y la creación de un mapa del mismo. Esto es de vital importancia ya que es esta información la que permitirá aplicar métodos de control para la navegación y, mediante el mapa creado, la forma de interactuar con los objetos de su entorno (por ejemplo las malezas en este caso).

En cuanto al mapeo y posicionamiento mediante sensores visuales, el entorno agrícola representa un desafío particular, debido a que es un ambiente homogéneo, con grandes variaciones de luz y el terreno es irregular. Son estas las razones por las que los métodos de SLAM utilizados (por ejemplo, S-PTAM[18] [17]) deben ser adaptados y probados previamente para que el método sea robusto y los resultados sean confiables.

1.3. Objetivos

Este trabajo en particular, tiene como objetivo la puesta en funcionamiento de los sensores necesarios para el posicionamiento del robot en el entorno agrícola. Permitiendo a partir de los mismos, localizar al robot durante su desplazamiento.

Para hacerlo es necesario implementar los siguientes sensores:

- Cámara estéreo: con el objetivo de ser el sensor principal para la localización y mapeo.
- Unidad Inercial: para obtener la orientación del robot.
- GPS-RTK: que proporcione una trayectoria confiable que sirva como referencia de comparación con la obtenida mediante la cámara. Durante este trabajo se hará referencia a este dispositivo como GPS, es decir omitiendo las siglas RTK.

Una vez instalados, los mismos proporcionaran información que, por medio de un método de SLAM que utilice principalmente la información de la cámara, permita obtener la posición del robot en el entorno agrícola. De esta forma, brindar una referencia de la posición del robot sin necesidad de recurrir al GPS.

Dichos resultados serán evaluados, adquiriendo noción del error obtenido, limitaciones o problemas que pudieran generarse, para conocer la viabilidad de implementar este método como una de las referencias de posición que serán utilizadas para la navegación del robot desmalezador. Es en este punto donde será utilizado el GPS, como referencia confiable para la comparación de resultados.

Una vez que el método de localización utilizado este funcionando a partir de los sensores y conociendo su cualidades, limitaciones y ajustes necesarios, el mismo se implementará para estar a disposición como una herramienta más del robot mientras realice su trabajo en el entorno agrícola.

Todo el proceso que se llevara a cabo esta expresado, en forma simplificada, en la Figura 2

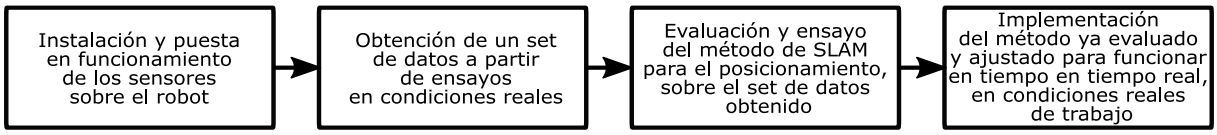


Figura 2: Esquema en diagrama de bloques del procedimiento, simplificado, del proyecto.

2. Marco Teórico

2.1. Modelo de cámara

Al utilizar una cámara como sensor se vuelve necesario poder expresar sus parámetros y obtener una referencia de que es lo que se ve representado en la imagen. Para determinar la relación entre los puntos ubicados en el mundo real, y la ubicación de los mismos en las imágenes obtenidas por medio de una cámara se recurre a un modelo, como por ejemplo el conocido como "pinhole". El modelo, por medio de una transformación proyecta los puntos en un plano, la misma es la siguiente:

$$p = K\,[R|t]\,P \tag{1}$$

o, expresada a partir de los componentes de las matrices:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

donde

- (X,Y,Z) son las coordenadas de un punto en el mundo real.
- (u,v) son las coordenadas de la proyección del punto en la imagen expresado en píxeles.
- K es la matriz intrínseca.
- (c_x, c_y) es el punto principal de la imagen, usualmente el centro de la imagen.
- (f_x, f_y) son las distancias focales expresadas en píxeles.
- s es el parámetro *skew* que en la mayoría de las cámaras es cero, pero podría no serlo si los píxeles no son perfectamente cuadrados

En la Figura 3 puede verse de forma ilustrada el modelo junto con la representación de sus parámetros.

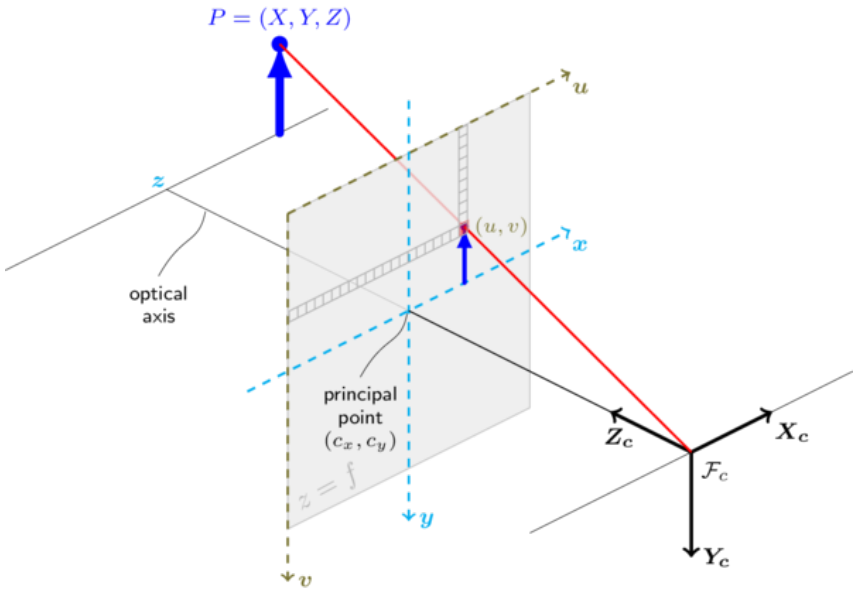
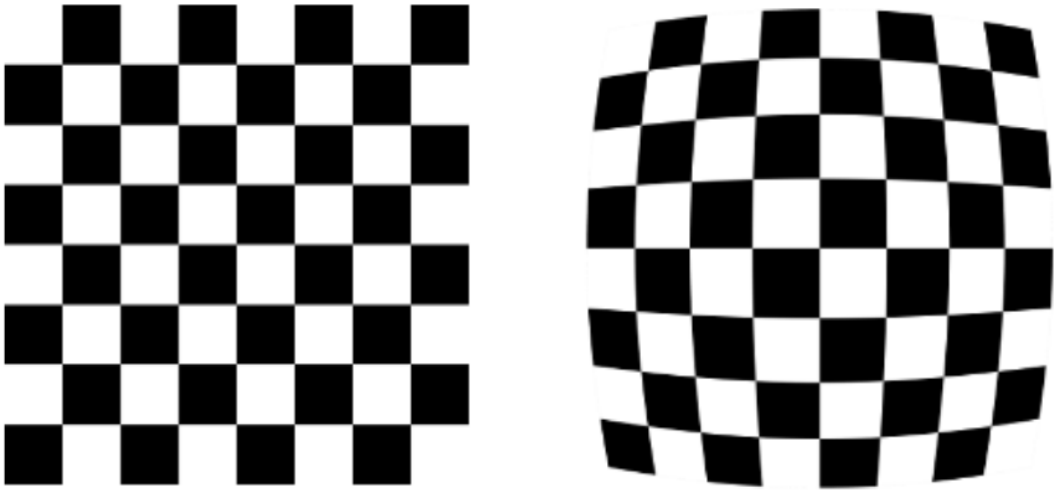


Figura 3: Ilustración del modelo pinhole. El centro de coordenadas es el centro de la cámara, P un punto en el mundo real, (u,v) la proyección de P en la imagen en píxeles.

Por otro lado, los lentes reales usualmente poseen cierta distorsión(efecto que puede verse en la Figura 4), que generalmente consiste de distorsión radial y tangencial. Es por esto que al modelo se le agregan los parámetros de distorsión radial: $k_1, k_2, k_3, k_4, k_5, k_6$ y los de distorsión tangencial: p_1, p_2. En muchos casos, como los términos superiores a k_2 se corresponden a coeficientes de orden alto, suelen ignorarse. De forma que los parámetros característicos resultan ser (k_1, k_2, p_1, p_2).



(a) Imagen sin distorsión

(b) Imagen con distorsion radial

Figura 4: Efecto de distorsión en los lentes.

2.2. Cámara estéreo

Una cámara estéreo es un tipo de cámara que posee dos lentes con un sensor de imagen para cada una de ellos. Esto permite simular la visión humana binocular y capturar imágenes tridimensionales.

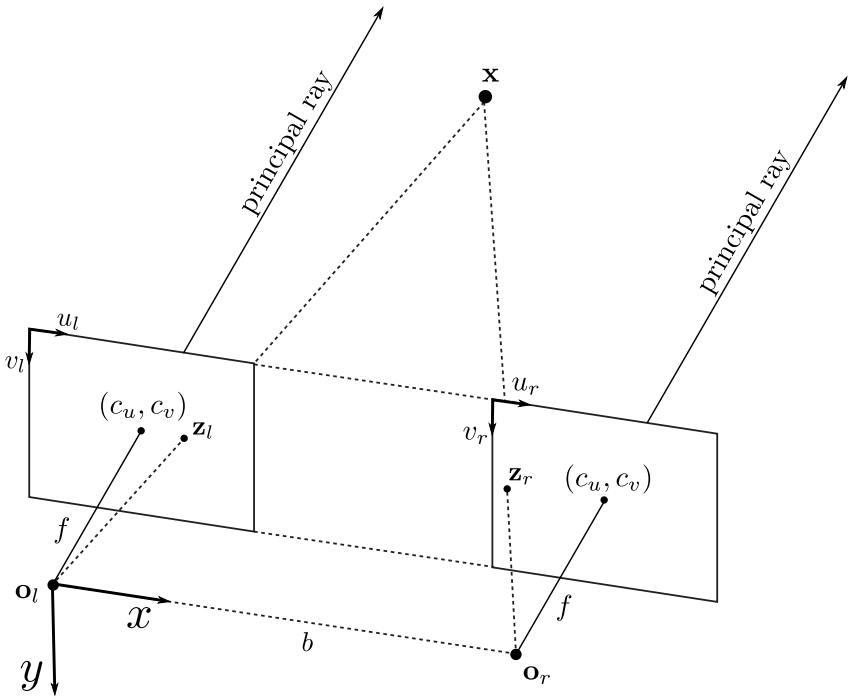


Figura 5: Principio de funcionamiento de la visión estéreo o binocular. o_i y o_r son los centros de las cámaras, (c_u, c_v) el centro de cada imagen, X un punto en el mundo real, Z_l y Z_r son las proyecciones del punto en cada una de las imágenes.

En la Figura 5 puede apreciarse el esquema de funcionamiento de este tipo de cámaras, donde b es la linea de base (o baseline en ingles) y f es la distancia focal. Un punto en el mundo real (X) se verá representado como un punto en cada una de las imágenes dependiendo de su profundidad (Z_l , Z_r), por lo tanto en función de la disparidad (desplazamiento relativo de los objetos en las dos imágenes) y aplicando semejanza de triangulos, es posible conocer la distancia del baseline de la cámara al objeto la cual suele denominarse Z.

$$Z = \frac{f\,b}{d}, \tag{2}$$

donde

$$d = uL - uR. \tag{3}$$

Por lo tanto es necesario conocer ciertos parámetros de la cámara, como son la distancia focal, la linea de base y a su vez la ubicación del punto de interés en ambas imágenes.

Para los dos primeros, es necesario obtener los valores de calibración de la cámara, tanto intrínsecos (dependientes de la geometría interna de los lentes) como extrínsecos (traslación y rotación existente entre ambos lentes). Mientras que para obtener los píxeles que representan

el mismo punto en ambas imágenes, es necesario buscar la correspondencia del mismo punto en ambas imágenes.

Este último problema puede simplificarse ampliamente incluyendo el concepto de plano epipolar descripto en la Figura 6. El punto del mundo real, su representación en ambas imágenes y el baseline estarán en el mismo plano, denotado como π. Por lo tanto conociendo el la linea de base y el punto en una de las imágenes, podemos saber sobre que segmento se encontrará el punto en la otra.

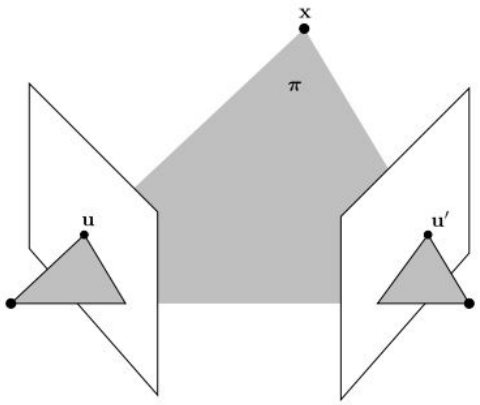


Figura 6: Plano epipolar y la representación de un punto sobre las imágenes de ambas cámaras.

Para poder aplicar este concepto a las imágenes obtenidas por ambas cámaras es necesario realizar la rectificación de las imágenes. Esto consiste en proyectarlas sobre un plano común, produciendo de esta manera que las lineas epipolares coincidan con filas de las imágenes. Es equivalente a considerar que las dos cámaras son paralelas y la transformación de un lente al otro es simplemente la traslación de las distancia correspondiente al baseline tal como puede verse en la Figura 5. De esta forma, al tener las imágenes rectificadas, puede aplicarse la ecuación (2) y la búsqueda de la correspondencia de un punto en ambas imágenes queda restringida a una fila de la imagen.

2.3. SLAM

La reconstrucción 3D del entorno encuentra grandes utilidades en la navegación de robots en forma autónoma dado que puede proporcionar mucha información del entorno.

En este marco se plantea el problema de la localización y mapeo simultaneo conocido como SLAM (*Simultaneous Localisation and Mapping*)[12] [9]. Las técnicas basadas en SLAM tienen como objetivo que el robot pueda generar un mapa del entorno en forma incremental, es decir, actualizándolo en la medida que el robot se mueva por el entorno real. Pero también ser capaz de localizarse en cada instante dentro de ese mapa generado.

Existen múltiples enfoques para este problema así como también las soluciones se basan en la utilización de diversos tipos de sensores (en general en combinación de más de uno) como sonares, GPS, infrarrojos, lasers y cámaras.

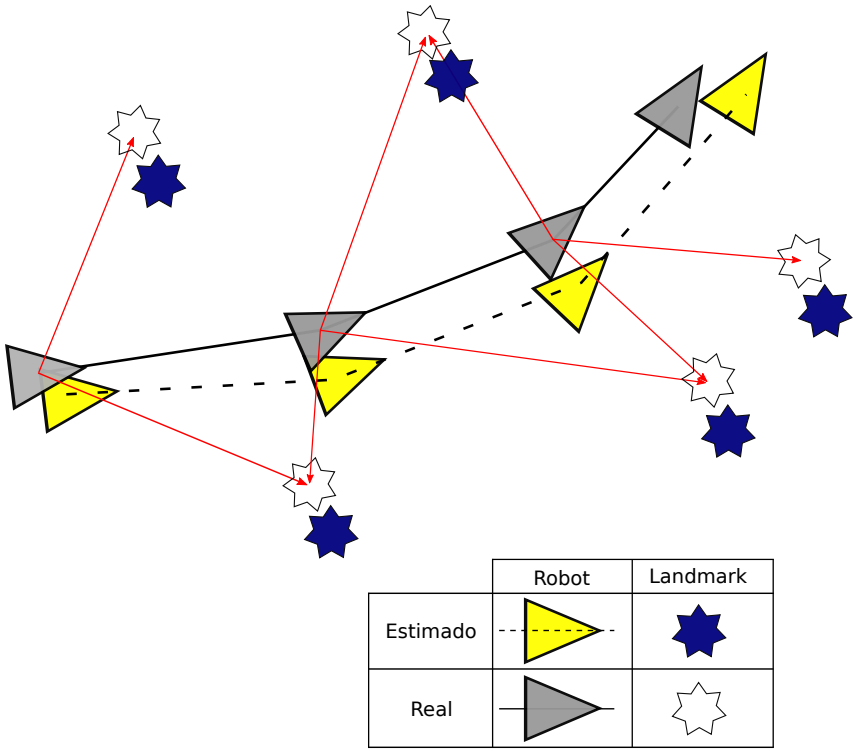


Figura 7: Ejemplo conceptual de SLAM.

En la Figura 7 se ilustra la idea del mapeo y localización en forma simultanea, donde el robot a partir de puntos denominados landmarks (puntos de elementos del entorno) genera su mapa y, en función de la distancia a los mismos, estima su posición dentro de su mapa. También puede verse en al imagen como el mapa generado no coincide exactamente con el entorno real, esto ocurre por los errores de los sensores en las mediciones. A este problema se agrega que a medida que avanza, el robot continuara midiendo su distancia a los puntos del entorno, pero estimando su posición a partir de la ubicación de dichos puntos en su mapa, que ya contenían cierto grado de error.

A pesar de esto, se han desarrollado una gran variedad de métodos de SLAM que presentan buenos resultados y gran robustez, brindando una poderosa herramienta para la navegación de robots en ambientes muy diversos e incluso cambiantes.

2.3.1. SLAM Visual

Uno de los métodos de SLAM de gran interés en la actualidad es el realizado a partir de cámaras, particularmente las estéreos [14]. Las ventajas de las mismas son que brindan gran cantidad de información del entorno y son sensores pasivos, es decir no interfieren con el desplazamiento del robot, ni con el entorno, ni con otros sensores.

Al utilizar imágenes como entrada de estos métodos, la información que se busca extraer son los puntos salientes o *features*, puntos cuyo gradiente es alto. A partir de los mismos es que se crea continuamente el mapa como fue descripto previamente.

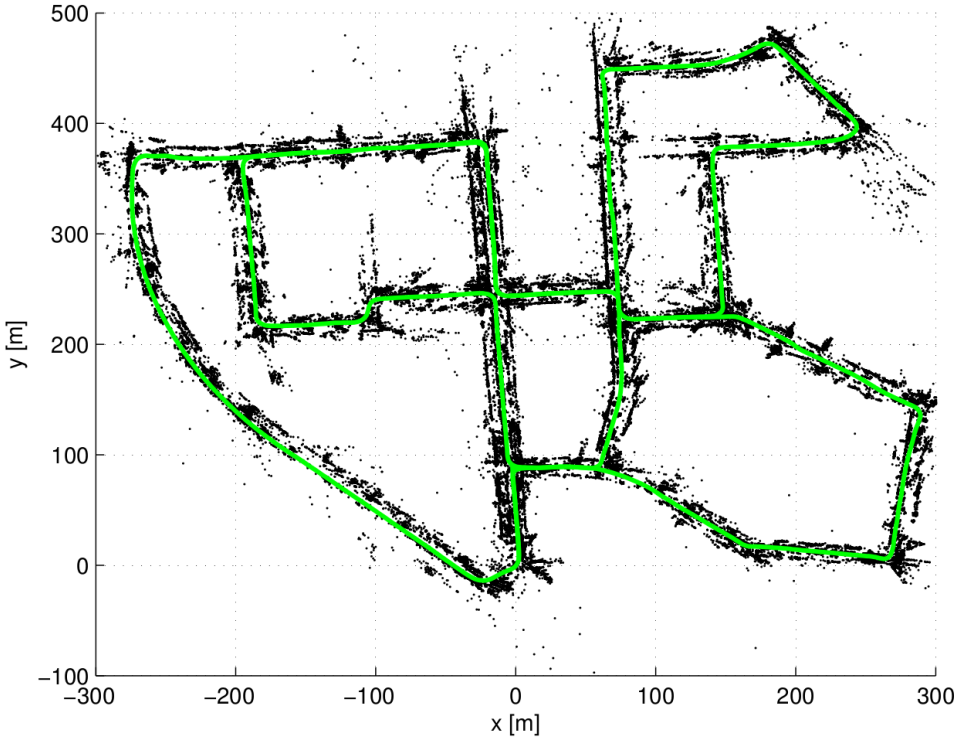


Figura 8: Ejemplo de utilización de algoritmo de SLAM visual por medio del método ORB sobre el set de datos de Kitti [16]. En verde la trayectoria del robot, los puntos en negros son puntos salientes utilizados para generar el mapa.

En la Figura 8 puede observarse la trayectoria descripta por un robot-auto obtenida por un método de SLAM, es notable que el algoritmo mantuvo consistencia aun a velocidades de movimiento considerables.

2.4. GPS-RTK

El sistema de Posicionamiento Global (o GPS por sus siglas en ingles) es un sistema que permite determinar la posición de un objeto en toda la tierra con una precisión, que habitualmente es de algunos metros. Para determinar las posiciones de dichos objetos, el sistema utiliza de al menos 3 de los 24 satélites orbitando en torno a la Tierra.

Al contar con 3 satélites, en teoría es posible obtener la posición exacta (latitud, longitud y altitud) de un objeto. El método utilizado consiste en medir el tiempo que demora la recepción del mensaje de cada satélite, y en base a este retardo, calcular la distancia al mismo. Sin embargo, estos métodos cuentan con ciertas fuentes de error que conllevan a tener, en general, prestaciones bajas cuando se requiere alta precisión. La causas de estos errores son por ejemplo el retardo que pueda tener la señal en la ionósfera, el error posible de los satélites o incluso la electrónica utilizada en el receptor. Debido a estos factores es que un GPS simple tiene acompañado un error de al menos 2 o 3 metros.

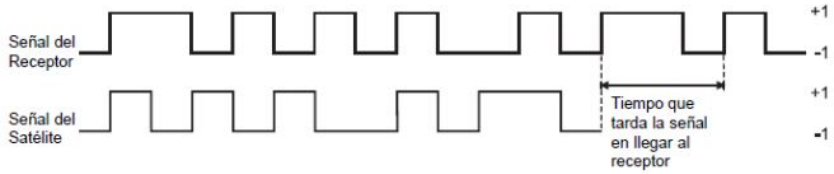


Figura 9: Comparación señal recibida del satélite con la generada localmente.

Por estos motivos es que se desarrollaron diversas técnicas para mejorar la precisión de los sistemas GPS, entre las cuales se destaca el posicionamiento diferencial. Este método, consiste en utilizar una estación base cuya posición es conocida de antemano, calcular su localización y poder estimar el error de las mismas. De forma que este error pueda usarse para corregir las mediciones realizadas sobre el objeto a localizar, al considerar que los errores están fuertemente correlacionados en los receptores próximos.

Cuando estas correcciones son enviadas por una base en tiempo real, durante el movimiento del rover (o dispositivo que se está desplazando), el sistema se denomina posicionamiento global cinético en tiempo real (o GPS-RTK por sus siglas en inglés)[19].

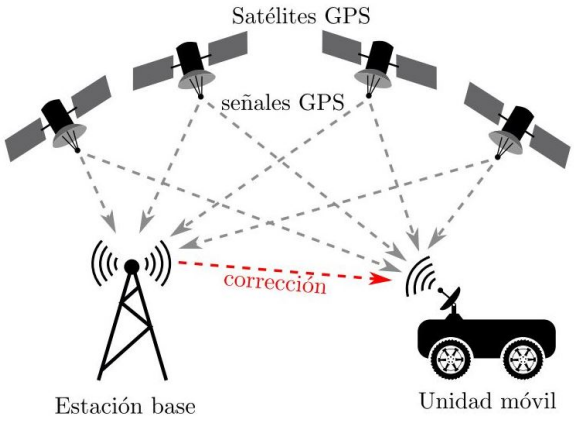


Figura 10: Principio de funcionamiento del GPS-RTK.

La estación base retransmite la fase de la portadora que midió, y las unidades móviles comparan sus propias medidas de la fase con la recibida de la estación de referencia. Esto permite que las estaciones móviles calculen sus posiciones relativas con precisión de centímetros, al mismo tiempo en que sus posiciones relativas absolutas son relacionadas con las coordenadas de la estación base.

Existen dos estados comunes en los que se puede encontrar un GPS-RTK: Fix y Float. La diferencia es consecuencia de un proceso llamado resolución de ambigüedad, que consiste en determinar la cantidad de longitudes de onda entre el dispositivo y el satélite, cuyo resultado puede ser un número entero (estado Fix) o uno flotante (estado Float). Al obtener un número entero de longitudes de onda, el algoritmo utilizado permite obtener resultados de posicionamiento de muy alta precisión.

Como desventaja, este tipo de sistemas requiere la adquisición de, al menos, dos dispo-

sitivos para que uno trabaje como estación base. Además, ambos deben poder establecer una comunicación estable que permita el envío de las correcciones al rover. El que funcionará como base debe establecerse en una posición fija y la distancia a la que podrá trabajar el dispositivo móvil de la base no podrá superar los 20 kilómetros para que su error sea similar al de la base, permitiendo una corrección eficiente.

2.5. Unidad de medición inercial

Las unidades de medición inercial (o IMU por sus siglas en inglés) son dispositivos que cuentan con varios sensores que permiten medir sus movimientos traslacionales y rotacionales[10]. Entre los sensores que pueden encontrarse están los acelerómetros, giróscopos y magnetómetros.

En función de cuantos de estos sensores contiene el dispositivo se lo considera de 3, 6 o 9 ejes respectivamente. El uso de la información proporcionada por cada uno de ellos puede ser analizada en forma separada, o combinada por medio de la utilización de filtros para obtener posicionamientos o, más usualmente, orientaciones de los dispositivos. Es particularmente útil integrar estas unidades a robots o vehículos móviles para poder conocer su orientación en todo momento.

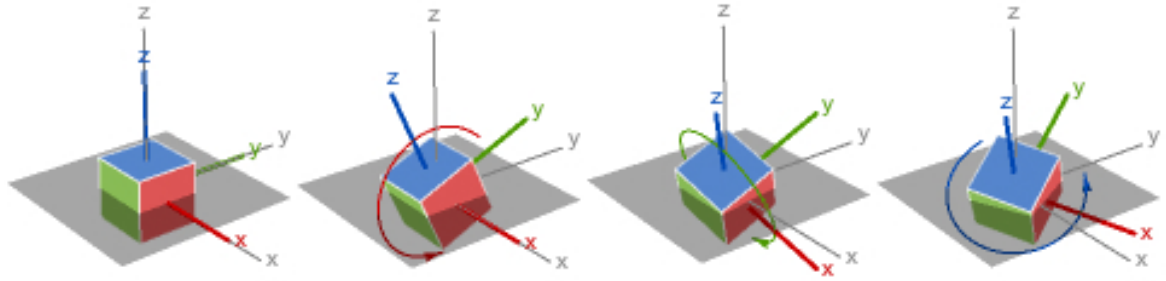


Figura 11: Ejemplo ilustrativo de la orientación de un objeto.

2.5.1. Acelerómetros

Es un sensor que permite obtener mediciones de la aceleración a la que está expuesto, en cada uno de sus 3 ejes (x,y,z).

Para comprender su funcionamiento básico, es útil imaginar un cubo con una esfera en su interior. Si el cubo no se encuentra afectado por el campo gravitacional, dicha esfera se mantendrá flotando en el centro, como puede verse en la Figura 12. Cada par de paredes opuestas representa un eje, como no puede medirse la aceleración en forma directa, se mide la fuerza ejercida sobre estas paredes del cubo. En el caso de la Figura 12, ninguna fuerza es ejercida sobre las paredes.

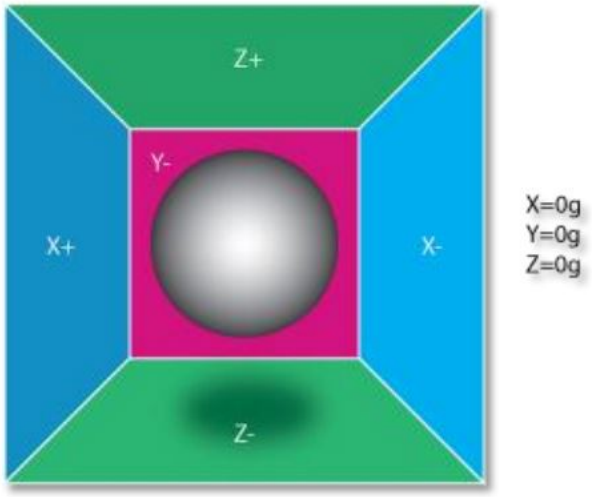


Figura 12: Ejemplo ilustrativo de funcionamiento de un acelerómetro al no estar sometido a la fuerza gravitacional.

Si ahora el cubo se desplaza a la izquierda con una aceleración de $9,8\frac{m}{s^2}$, lo cual suele expresarse como 1g, la situacion es la de la Figura 13. Puede verse que una fuerza aparece en el eje x, que puede expresarse como -1g.

Con este principio es posible determinar la aceleración del cubo en función de las fuerzas sensadas sobre las paredes del mismo. Evidentemente en el caso de que el movimiento sea en varios ejes, aparecerán fuerzas en más de una pared del cubo.

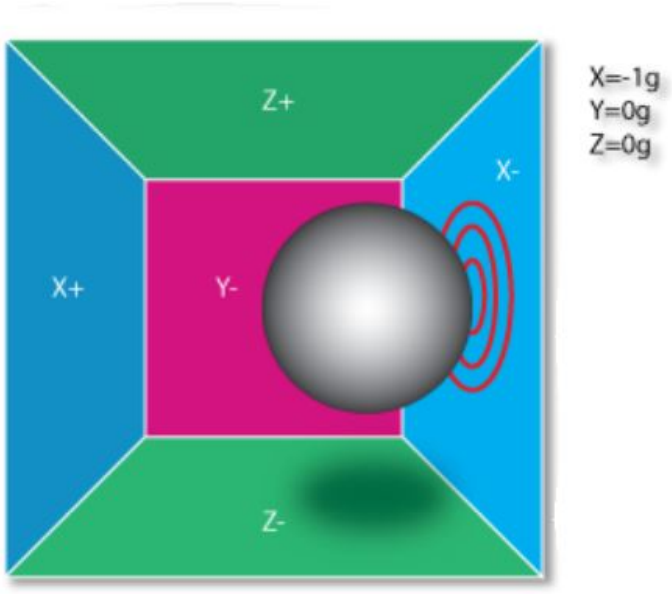


Figura 13: Ejemplo ilustrativo de funcionamiento de un acelerómetro estar sometido a una aceleración en el eje x.

Una situación interesante es la del cubo en reposo, al estar sometido a la fuerza gravita-

cional, por ejemplo en la dirección del eje Z, como se aprecia en la Figura 14 esto implicaría una medición de -1g según el esquema propuesto cuando en realidad el cubo no esta acelerado. Esto es efectivamente lo que ocurre, en este tipo de dispositivos, al estar en reposo puede medirse una aceleración de $9,8\frac{m}{s^2}$ en la dirección correspondiente según, cuál sea su disposición en el espacio.

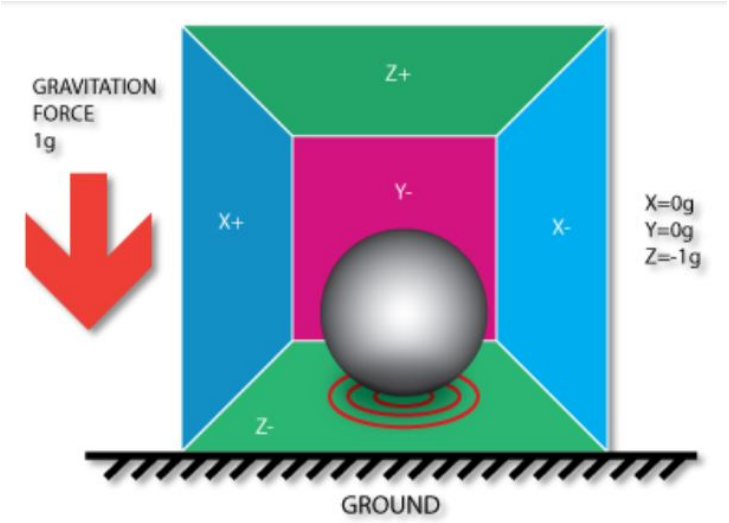


Figura 14: Ejemplo ilustrativo de funcionamiento de un acelerómetro estar en reposo, estando sometido a la fuerza gravitacional.

Existen distintos tipos de acelerómetros en función de cómo se realiza la medición de la fuerza en cada eje. Puede ser, por ejemplo, mecánico (utilizando resortes) o electrónico (sensando campos eléctricos o magnéticos).

En general una vez obtenidos estos valores, deben pasar por un conversor analógico digital para obtener el valor digital de la medición.

2.5.2. Giróscopos

Los giróscopos o giroscopios son dispositivos que permiten medir velocidad angular, en revoluciones por segundo o grados por segundo. De igual forma que los acelerómetros, no realizan las mediciones de forma directa, sino que lo hacen de forma indirecta. En este caso aprovechando el efecto Coreolis.

$$\overrightarrow{F_c} = -2m(\overrightarrow{w} \times \overrightarrow{v}) \tag{4}$$

Como puede verse en la ecuación (4) al medir la fuerza de Coreolis a la que esta sometido el dispositivo puede calcularse la velocidad angular del mismo. Esto puede realizarse de distintas maneras, mediante una pequeña masa que se desplaza o nuevamente, aprovechando efectos capacitivos.

Al igual que con los sensores de aceleración lineal, las mediciones suelen requerir un conversor analógico digital para poder obtener el valor final de la medición.

2.5.3. Combinación acelerómetros y giróscopos

Para obtener la orientación de un dispositivo es usual combinar los efectos tanto de acelerómetros como de giróscopos. Esto es así debido a que pueden complementarse adecuadamente, supliendo las falencias de ambos sensores.

Por ejemplo los giróscopos son dispositivos que pueden calcular la variación de los ángulos en cada uno de los ejes, sin embargo acumulan error en las mediciones que provocan desviaciones con el tiempo. Esto no ocurre con los acelerómetros, por lo que pueden ser usados para corregir esos desvíos. En contraposición los giróscopos no son sensibles a la gravedad.

Para realizar la fusión de ambos sensores, el enfoque es ponderar adecuadamente cada uno de los sensores, considerando también los errores de ambos utilizando modelos ruidosos. Un ejemplo de esto es la utilización de un filtro de variables complementarias[21] o también podría realizarse por medio de un filtro de Madgwick[15].

Por otro lado, también existe un método de fusión de sensores muy difundido conocido como filtro de Kalman[20]. Es un algoritmo recursivo que permite la fusión de información que esta inmersa en ruido blanco.

3. Desarrollo

3.1. Sensores utilizados

3.1.1. Cámara Zed

La cámara utilizada es la *Zed Stereo Camera* de stereolabs[1]. La misma permite obtener imágenes RGB tanto del lente izquierdo como derecho y a partir de las mismas la información de profundidad. La misma posee un baseline (o distancia entre cada uno de los lentes) de 12 cm y permite obtener imágenes en diferentes calidades, llegando incluso a 2K, esta información se detalla en la Tabla 1. Puede verse que puede trabajar a un alto frame-rate, y cabe destacar que la resolución esta expresada como la suma de ambas imágenes al ponerlas una al lado de la otra (es decir, el ancho de cada imagen es la mitad del expresado en la tabla).

Figura 15: Cámara estéreo Zed de StereoLabs.

Modo	Frames por segundo	Resolución(lado a lado)
2.2K	15	4416x1242
1080p	30	3840x1080
720p	60	2560x720
WVGA	100	1344x376

Cuadro 1: Modos de la cámara Zed.

La conexión se realiza por medio de USB 3.0 cuya velocidad permite garantizar los valores dados en la tabla 1.

La forma de conectarse con la cámara es por medio del cable USB 3.0 que viene integrado con la misma.

3.1.2. GPS-RTK Emlid Reach

Como GPS de correcciones en tiempo real se utilizaron dos módulos Reach GNSS de Emlid, como los de la Figura 16. Los mismos son módulos de bajo costo pero que permiten alta precisión una vez que alcanzan el estado fix, al trabajar de forma diferencial.

[1] www.stereolabs.com/

Figura 16: Módulo GPS reach de Emlid.

Los módulos son alimentados con 5v por medio de un cable micro-USB o el puerto DF13. La antena receptora de la señal de GPS debe ser colocada sobre un plato conductor de al menos 10 cm x 10 cm para evitar interferencias tal como se ilustra en la Figura 17

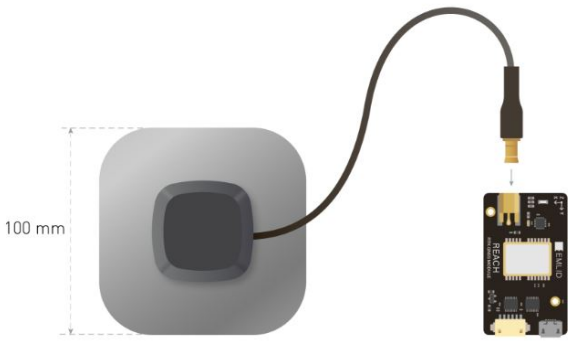


Figura 17: Integración del receptor de señales GPS al módulo Emlid Reach.

Como debe existir una comunicación estable entre los dos módulos para que uno, al actuar como base, envíe las correcciones al otro, estos traen incorporado un módulo WiFi que le permite crear una red propia, actuando como hotspot o conectarse a una creada externamente. Por otro lado posee un conector UART que posibilita el agregado de un modulo RF utilizando protocolo serie para la comunicación.

Estas correcciones pueden ser recibidas en formato RTCM2, RTCM3, OEM4, OEM3, UBX, SS2, HEMIS, SKYTRAQ, SP3. Dentro de las cuales RTCM3 es la más utilizada. En cuanto a la información proporcionada por este módulo, esta puede ser recibida por una conexión serie, TCP o Bluetooth. De igual forma para la comunicación entre ambos módulos.

La información de la posición del modulo, puede expresarse en diversos formatos estandarizados para el posicionamiento satelital

- LLH: Información simple de latitud, longitud y altitud, junto con el estado de la solución.
- XYZ: Información de la posición X,Y,Z utilizando el sistema de coordenadas ECEF, junto con el estado de la solución.
- ENU: Protocolo para el posicionamiento expresado como las componentes este, norte y arriba en función de la linea de base y el estado de la solución.
- NMEA 0183: El más utilizado ya que permite la utilización de diferentes tipos de sentencias dentro del mismo protocolo para brindar información diferente en cada una. Las más importantes son GGA y RMC.
- ERB: Utilizado para la comunicación con ArduPilot.

Todos los parámetros y configuraciones pueden modificarse mediante la interfaz gráfica denominada ReachView, la cual es accesible vía TCP.

3.1.3. IMU LSM6DS0

La unidad inercial utilizada es la LSM6DS0 de ST Microelectronics. La misma posee tanto un acelerómetro como un giróscopo de 3 ejes cada uno. Pueden ajustarse sus rangos máximos de medición (tanto en excursión positiva como negativa) entre los valores 2/4/8/16 g para el acelerómetro y 245/500/2000 dps para el giróscopo.

La comunicación con el integrado puede realizarse por I^2C o por SPI. La información obtenida a partir del integrado es un valor en cuentas, representativo del valor sensado. Es decir, el dispositivo realiza la medición analógica, pasa por un conversor analógico-digital y esa información es enviada por el canal de comunicación utilizado.

Por ejemplo si se elige como máximo de medición 2g, la excursión de la lectura sera de 4g (-2g a +2g) y el menor valor posible, que es el bit menos significativo, saldrá de la relación:

$$sensibilidad = \frac{M}{C} \tag{5}$$

C: es la capacidad máxima de representación, que depende de la cantidad de bits disponibles en el integrado.
M: es la la variación máxima que puede tomar la medición.

En este caso, como ejemplo, el dispositivo es de 16 bits por lo que puede representar 2^{16} valores diferentes, y si la excursión máxima es de 4g entonces se obtiene:

$$sensibilidad = \frac{4}{2^{16}} = 0{,}000061\ g/LSb \tag{6}$$

Mismo criterio puede aplicarse para el calculo de los valores del giróscopo del dispositivo.

Estos valores de sensibilidad ya están calculados en la hoja de datos en función el valor máximo que se seleccione. Cabe destacar que este integrado esta incluido dentro de una cámara Tara, por lo que la comunicación es gestionada por este dispositivos que realiza la conversión y calcula los resultados en grados por segundo (DPS) para el giróscopo y mili-unidades de gravedad para el acelerómetro (mg).

3.2. Computadora utilizada

Para poder obtener la información de todos los sensores en forma sincronizada, guardar y procesar la información se utilizo una MINI-PC Intel® NUC Kit NUC6CAYH[2] (Intel Celeron J3455 CPU, quad-core 2,3 GHz and 8 GB DDR3 RAM Memory) con Ubuntu 16.04. A la misma se le incluyo un disco de estado solido con el objetivo de minimizar los tiempos de escritura que pudieran dificultar la obtención del set de datos

Figura 18: Mini PC NUC kit NUC6CAYH.

La computadora que puede verse en la Figura 18, es de tamaño reducido (aproximadamente 11x11x6 cm), posee disipador en lugar de couler (el cual podría traer aparejados problemas con la tierra y el polvo al trabajar en el campo) y dispone de cuatro puertos USB. Puede alimentarse entre 12 y 19 voltios, haciéndola viable de ser montada sobre el robot desmalezador que posee alimentaciones de 5, 12 y 36 voltios.

3.3. Routers utilizados

Para establecer la red inalámbrica que será utilizada (como sera explicado posteriormente) para permitir la comunicación entre ambos módulos GPS y con la computadora, se decidió utilizar dos Routers WiFi. La utilización de dos Routers se explica en la necesidad de cubrir grandes distancias de cientos de metros al trabajar en una zona agrícola y, por otro lado, que lo antena incorporada en los módulos GPS Emlid no posee gran alcance. Por esto es necesario incorporar un router en las cercanías de cada módulo, ya que los mismos no pueden detectar redes a distancias superiores a 10 o 15 metros.

Dado que la atenuación en espacio libre (FLS) puede calcularse de la siguiente manera:

$$FLS[dB] = 20\ \log(d) + 20\ \log(f) + K \tag{7}$$

donde d es la distancia entre las antenas en Km, f la frecuencia de la señal en MHz y K, con esas unidades de d y f, es 32.44. Trabajando a 2.4 GHz y si se requiere un alcance de 1000 metros entre las antenas, de la ecuación (7) se obtiene la(8):

[2] `www.intel.com`

$$FLS[dB] = 20\ \log(1Km) + 20\ \log(2400MHz) + 32{,}44 = 100dB \tag{8}$$

Los routers utilizados son un MikroTik Metal G-52SHPac[3], destinado a estar junto al módulo GPS de la base, y un MikroTik Groove GA-52HPacn[3], que esta pensado para estar montado sobre el robot junto al módulo GPS que estará allí ubicado.

(a) MikroTik Metal

(b) MikroTik Groove

Figura 19: Routers utilizados.

Ambos dispositivos están diseñados para exteriores, poseen una potencia de transmisión de 24 dBm y sensibilidad de -90 dBm. Ambos vienen equipados con antenas de 6dBi por lo que en función de la ecuación (8) la señal llegaría al receptor con una potencia mayor a la de la sensibilidad. Si bien estos resultados son completamente teóricos y desprecian otras perdidas (como conectores y cables) el resultado obtenido deja un margen para los mismos, considerando además, que las distancias reales de utilización del robot no superaran los 500 metros y en la zona de trabajo no habrá obstáculos en la linea de vista entre los routers.

Los dos pueden ser alimentados entre 10 y 30 voltios, la diferencia más notable entre ambos es que el MikroTik Groove consume como máximo 5W en lugar de los 11.5W del MikroTik Metal. Considerando que se busca la autonomía del robot en cuestión energética, esta reducción es importante.

3.4. Puesta en funcionamiento de los sensores

Para que cualquier algoritmo de navegación que desee implementarse sea eficiente, es necesario que la información de la localización del robot sea confiable y esté a disposición con el menor retardo posible. Por esto se diseñaron varios subprogramas para obtener la información de cada uno de los sensores y, según sea necesario, guardarla o procesarla. Esta última distinción se debe a que existen dos situaciones posibles de funcionamiento:

1. Obtener set de datos para su simulación posterior y ajuste de algoritmos.

[3] `https://mikrotik.com/`

2. Obtener la información y procesarla en tiempo real para el posicionamiento en situación de funcionamiento real.

En el diseño se utilizó el primer enfoque, ya que en primera instancia fue necesario obtener un conjunto de datos que permitan perfeccionar los métodos de localización. Por lo tanto el enfoque con cada uno de los sensores fue obtener la información y guardarla, cualquiera sea el formato, junto con el indicador del momento en el que fue obtenida (es decir, ademas del dato sensado se agrega el *timestamp* del mismo).

Dados los sensores disponibles y la no necesidad, para el objetivo planteado, de que todas las mediciones fueran obtenidas en el mismo instante (por ejemplo, mediante una señal de *"trigger"* que indique a todos los sensores que adquieran una medición), se priorizó conocer el instante en el que son obtenidas cada una de las mediciones. De esta forma, es posible determinar el tiempo transcurrido entre distintas mediciones realizadas. Al utilizar la misma computadora para la obtención de los datos, el reloj de la misma es el que determinará dichos tiempos (*timestamps*), evitando problemas de sincronización. En este punto es importante destacar que dado que la computadora posee un sistema operativo, podrían producirse inconvenientes al trabajar a altas frecuencias ya que al requerirle a la NUC que guarde el tiempo de una medición, el sistema operativo podría posponer por un determinado tiempo esta acción debido a la multiplicidad de tareas que realiza simultáneamente. A pesar de esto, dadas las bajas frecuencias de casi todos los sensores y las bajas velocidades del robot, se decidió utilizar la NUC con Ubuntu ya que esto simplificó muchas de las tareas a realizadas (por ejemplo la puesta en funcionamiento de cámaras).

3.4.1. Obtención de las imágenes a partir de la cámara Zed

Para obtener las imágenes izquierda y derecha de a cámara, se diseño un subprograma en C++ que crea una instancia de la clase ZedCamera definida por OpenCV[4], una librería que facilita el tratamiento y procesamiento de imágenes. Por medio del constructor de dicha clase se especifica el frame-rate y la resolución a utilizar.

Posteriormente se utiliza un sistema del tipo productor-consumidor para poder realizar dos procesos simultáneos:

- Obtener las imágenes del buffer de la cámara cuando estén disponibles y ponerlas en cola.
- Guardarlas en la carpeta de destino cuando estén en cola.

La gran ventaja es que se evitan posibles problemas de retardos en los tiempos de escritura, que podrían producir perdida de imágenes o errores en la sincronización entre cada imagen y su timestamp. Al ser realizado de esta forma, aun si el proceso de escritura requiriera un tiempo mayor al disponible antes de la aparición de la nueva imagen, estas quedaran en cola con el timestamp correspondiente al momento en que aparecieron en el buffer. Este funcionamiento, que se mantiene durante la obtención de datos, es descripto en la Figura 20.

[4]https://opencv.org/

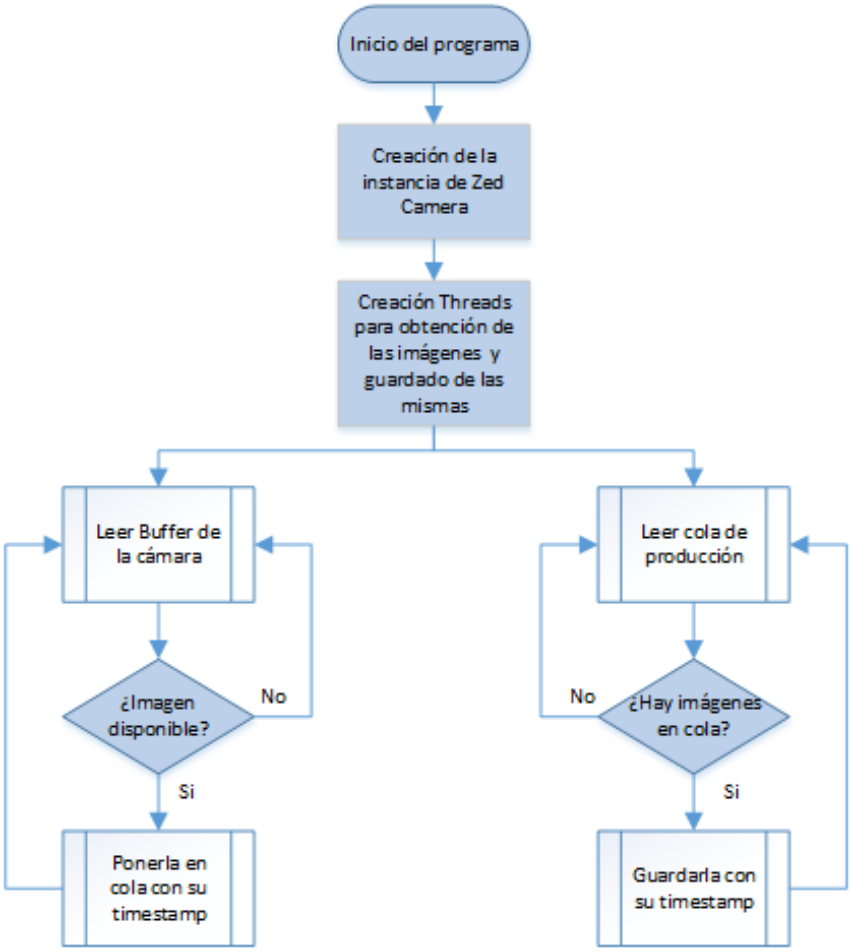


Figura 20: Diagrama de flujo simplificado para la obtención y guardado de las imágenes obtenidas de la Zed.

Por otro lado, al inicializar el objeto ZedCamera, se ajustó para que la exposición de la cámara sea automática ya que, de no ser así, las grandes variaciones de luz que hay en exteriores afectaría la calidad de las imágenes. El tamaño utilizado de las imágenes es 672x376, ya que para el propósito planteado no es necesaria una imagen de gran tamaño. Por otro lado aumentar el tamaño de las imagenes trae aparejado dos inconvenientes:

1. Espacio ocupado en el disco: esto podría complicar la obtención de sets de datos extensos, obligando a la utilización de discos de mayor capacidad, que al ser de estado solido, tendrían un precio muy superior.

2. Tiempo de procesamiento de las imágenes: este aspecto es clave para el funcionamiento en tiempo real, si se utilizaran imágenes más grandes, todo algoritmo utilizado requeriría mayores tiempos de procesamiento para el reconocimiento de puntos en las imágenes.

Las imágenes se guardan en un directorio denominado zed, donde el nombre de cada una de las imágenes es el siguiente:

$$<cámara>\_<timestamp>.png$$

Cámara sera "left.º right"segun corresponda, mientras que el timestamp sera expresado en segundos con seis decimales.

3.4.2. Obtención información GPS-RTK

El módulo GPS-RTK de Emlid esta diseñado para ser utilizado en conjunto con la aplicación ReachView por medio de TCP. En primera instancia el mismo modulo actúa como hotspot, generando una red a la cual se puede acceder por medio del IP 192.168.42.1 para realizar las primeras configuraciones. Al hacerlo se le configura una nueva red Wi-Fi que es la que será utilizada en forma definitiva.

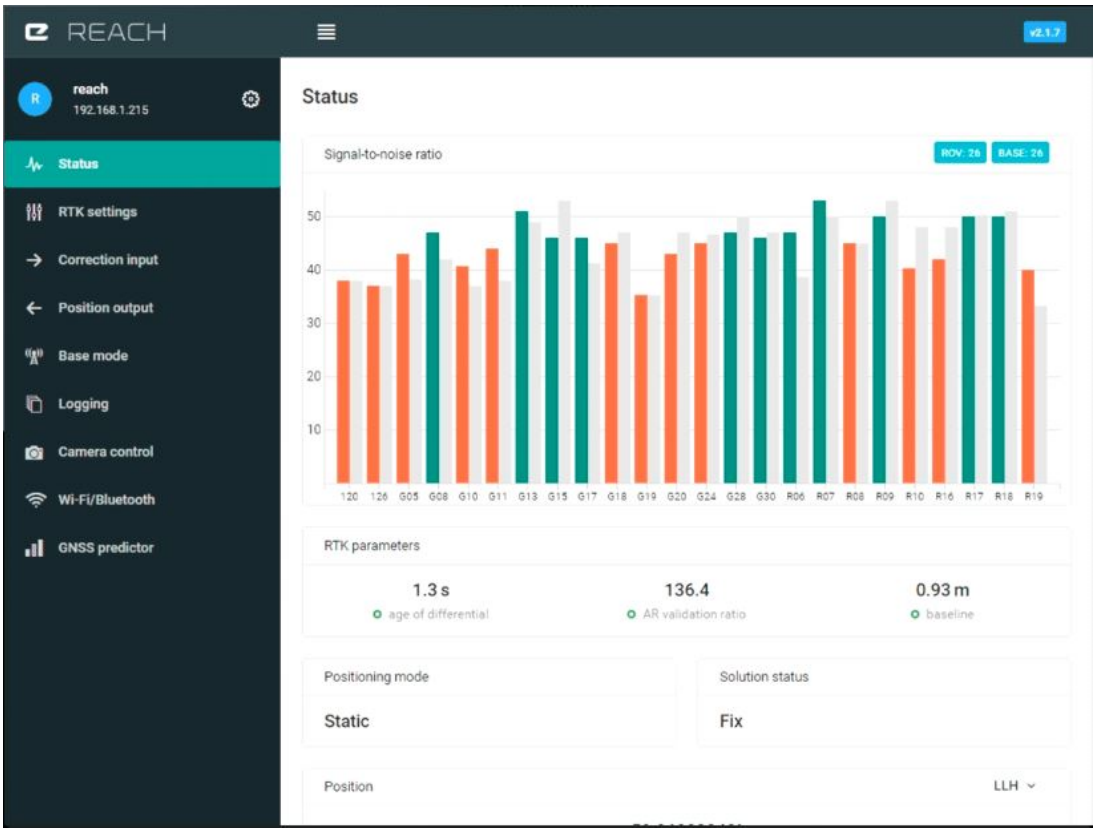


Figura 21: Ejemplo interfaz grafica ReachView.

Esto se realizó en ambos módulos, y a partir de ese momento fue necesario configurarlos según la función de cada uno (como base o rover)

Configuración de la base:

- Nombre del dispositivo: Base
- Recibir información de los satélites: GPS, Glonass, SBAS, QZSS
- Frecuencia de actualización de la información: 5Hz
- Formato correcciones a enviar: RTCM3

- Enviar correcciones vía: TCP
- Especificar correciones de la base en forma: Average (promediar)

Configuración del Rover:

- Nombre del dispositivo: Rover
- Modo de posicionamiento: Kinematic
- Modo de resolución de la ambigüedad: Fix-and-hold
- Recibir información de los satélites: GPS, Glonass, SBAS, QZSS
- Frecuencia de actualización de la información: 5Hz
- Formato correcciones a recibir: RTCM3
- Recibir correcciones vía: TCP
- Formato posición: NMEA 0183
- Enviar posición vía: TCP

Una vez configurados y al conectarlos a una red en común, el dispositivo funcionando como Rover es capaz de transmitir los mensajes correspondientes por medio de TCP, pero es necesario conectarse a su puerto (por medio de su IP). Por esto se creó un subprograma en C++ que genere la conexión a dicho puerto, reciba continuamente las lineas de mensaje del GPS y las guarde en un archivo de texto junto con el timestamp del momento de recepción. este funcionamiento esta representado como diagrama de flujo en la Figura 22

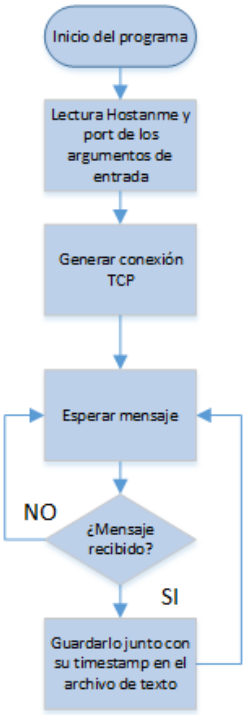


Figura 22: Esquema en diagrama de flujo del programa que leer la información del GPS recibida por TCP.

Durante la obtención de datos es posible supervisar el funcionamiento de los módulos por medio de la aplicación. Esto fue realizado para determinar la confiabilidad de los resultados y el momento en el que el GPS entra en modo Fix ya que por lo general desde que se realizaba la conexión demoraba varios minutos en ocurrir.

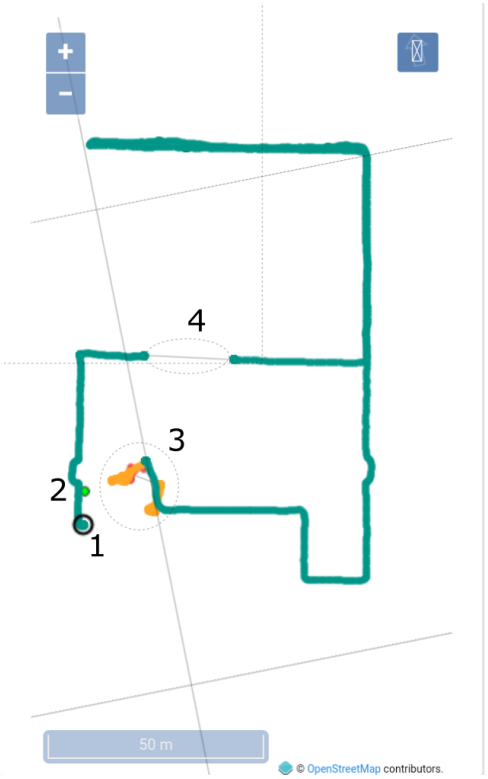


Figura 23: Ejemplo trayectoria obtenida de la interfaz ReachView con los módulos Emlid Reach actuando en estado Fix. Experiencia realizado en la cancha de futbol de la Siberia, Rosario.

Se realizaron varios ensayos utilizando los módulos para corroborar su funcionamiento y la configuración adecuada, en los cuales se obtuvieron datos de trayectos como el de la Figura 23. En la misma se marcaron varios puntos de interés para interpretar las gráficas:

1. Posición actual.
2. Posición de la base.
3. Posiciones iniciales. Puntos en estado float en naranja, puntos en estado fix en verde.
4. Tramo donde no hay ningún punto debido a que se perdió la conexión del dispositivo que esta supervisando la trayectoria, sin embargo durante este tramo el envío de datos por TCP. continua ininterrumpido.

A partir de esta experiencia se determinó que deben esperarse varios minutos para que la medición sea estable y precisa, pero una vez alcanzado el estado fix los módulos mostraron un gran desempeño.

En la Figura 21 pueden verse las barras que representan el nivel de señal recibido de cada satélite, información del estado de la solución, posición y también *age of differential* que proporciona información de demoras en las correcciones y *Ambiguity Resolution validation ratio*, que es una referencia de la confiabilidad de la solución, si es mayor a 3 se la considerará en estado Fix.

3.4.3. Obtención información IMU

Para recuperar la información de la unidad inercial (acelerómetro y giróscopo), la comunicación debe realizarse con la cámara Tara ya que el integrado esta incorporado en la misma. La conexión con la cámara se realiza por USB y para recuperar la información, al igual que en el caso de las imágenes de la Zed, se realizo un subprograma en C++. En este caso nuevamente se utilizan Threads para sincronizar la lectura de la información y la escritura de la misma.

En primer lugar se crea un objeto TaraImuRecorder sobre el cual se habilitan el acelerómetro y el giróscopo, se configuran los valores máximos de cada medición, que producirán el valor de la sensitividad, es decir, el valor analógico correspondiente a un cambio en el bit menos significativo. La frecuencia elegida fue 119Hz, la excursión del acelerómetro entre -2g y 2g, y para el giróscopo entre -245 dps y 245dps. Esto fue así debido a que las aceleraciones y variaciones de ángulos del robot no serían de gran importancia, por lo que se buscó priorizar la precisión de las mediciones. En cuanto a la frecuencia, se seleccionó una que sea mucho mayor a la de trabajo de las cámaras, por lo que 119Hz era adecuada.

Posteriormente se crean los dos hilos, el que leerá la IMU y actualizará el buffer , y el que tomará la información, cuando sea correcta, y la escribirá en el archivo de texto. La información es correcta cuando está completa y se corresponde a la medición de la frecuencia impuesta. Para mantener el sincronismo entre estos dos procesos se utiliza un recurso llamado mutex. Su funcionamiento es simple, permite la utilización de un recurso compartido habilitando la lectura y escritura del mismo por un hilo (o thread) cuando el mismo disponga del mutex y bloqueándolo para todos los otros hilos. Una vez que se termina de hacer uso del recurso, se libera el mutex, permitiendo que otros recursos puedan hacer uso del mismo.

Este funcionamiento es descripto en la Figura 24. Cabe destacar que, al igual que en el caso de las imágenes obtenidas de la cámara Zed, cada medición es guardada en el archivo de texto junto con el timestamp del momento donde fue leído.

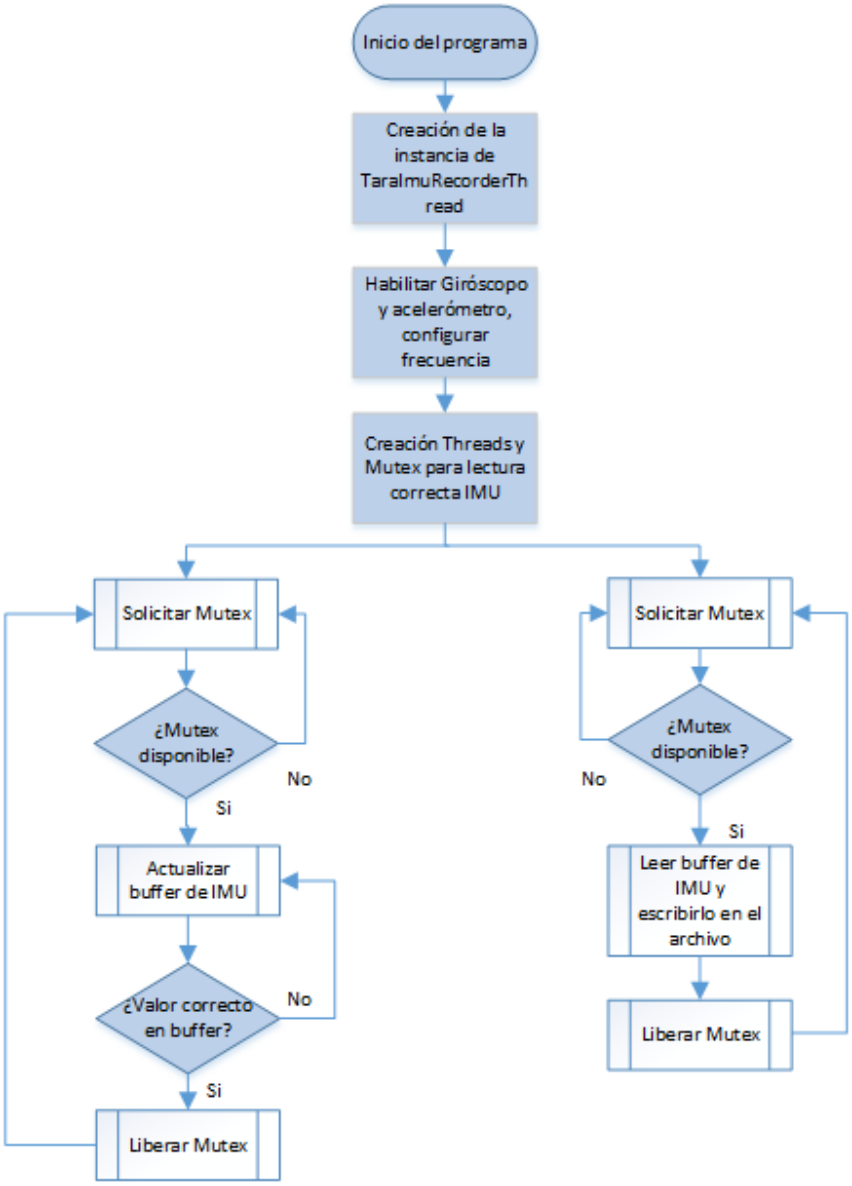


Figura 24: Diagrama de flujo simplificado para la obtención y guardado de las mediciones de la IMU integrada en la cámara Tara.

Los valores obtenidos son guardados en una archivo de texto, dentro de un directorio llamado "tara". En el archivo los datos son escritos en una linea para cada medición donde cada una de ellas posee la siguiente estructura:

$$<timestamp>\ IMU:\ <GyroX>\ <GyroY>\ <GyroZ>\ <AccX>\ <AccY>\ <AccZ>$$

Los timestamps estan expresados en segundos, las mediciones del giróscopo en grados por segundo (DPS) y las mediciones del acelerómetro en milésimas de la fuerza de gravedad (mg).

3.4.4. Lectura de la información de los motores

El robot cuenta con cuatro motores brushless acoplados a las ruedas que junto con un motor paso a paso para la dirección permiten el desplazamiento del mismo. Para obtener información de estos cinco motores el grupo del proyecto encargado de la puesta en funcionamiento del robot utilizó sensores de efecto hall en los motores brushless y un encoder absoluto en la salida de la reducción colocada en el paso a paso de la dirección.

Si bien la puesta en funcionamiento de estos sensores ya había sido realizada y no corresponde a este desarrollo, sí forma parte la lectura de los mismos que permitirá obtener información de la odometría del robot.

Para poder obtener información proveniente del microcontrolador, implementado por el grupo del sistema de tracción y dirección del robot,y de esta forma leer la información de los sensores de los cinco motores se estableció una comunicación serie. Desde el microcontrolador aprovechando el puerto UART que disponía el mismo, y desde el lado de la PC (correspondiente a este desarrollo) por medio de un módulo conversor USB-UART que se conecta a un puerto USB de la PC. Esta configuración básica es ilustrada en la Figura 25

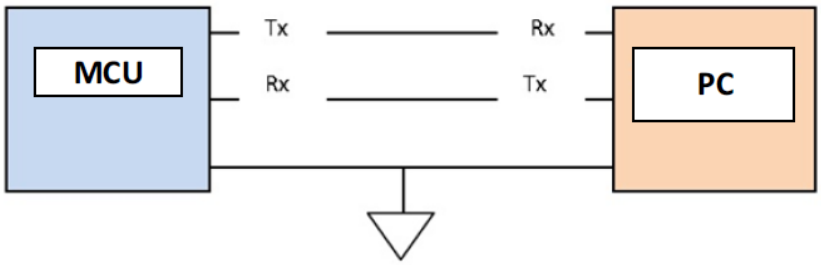


Figura 25: Esquema básico de la comunicación serie establecida entre la PC y el microcontrolador.

Cabe destacar que esta comunicación es de utilidad tanto para la lectura de información proveniente del microcontrolador como para enviar comandos de movimiento desde la PC si fuera necesario, en lugar de utilizar el control remoto.

La trama del mensaje definida para la transmisión de la información consistía en una linea encerrada entre paréntesis con la información de la velocidad, pulsos detectados por los sensores, y medición de corriente de cada uno de los cuatro motores y finalmente la dirección y sentido, obtenidos del motor paso a paso. Este protocolo definido puede observarse en la Figura 26. Puede apreciarse que a la información se le agrega, al principio del mensaje, el modo de funcionamiento del robot.

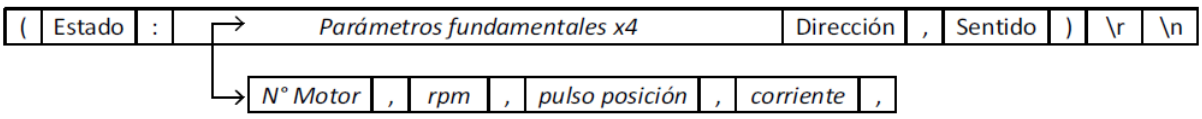


Figura 26: Protocolo de la información enviada del microcontrolador a la PC con las lecturas de los motores.

- *Estado:*
  1. Duty_REMOTO
  2. RPM_REMOTO
  3. Duty_PC
  4. RPM_PC
- *Nº Motor:* "0"(rueda DI) , "1"(rueda DD), "2"(rueda TI) , "3"(rueda TD)
- *RPM:* velocidad angular instantánea de la rueda del robot en rpm
- *Pulso posición:* variable proporcional a la distancia recorrida por el motor
- *Corriente:* corriente consumida por el motor
- *Dirección:* valor entero entre [-100,100] proporcional al ángulo de la dirección del robot
- *Sentido:* "0çuando el robot avanza hacia adelante, "1çuando el robot se mueve marcha atrás.
- $\backslash r$ $\backslash n$*:* Indicadores de finalización de una linea de información

Se decidió de común acuerdo entre ambos grupos que la frecuencia de transmisión de los mensajes sea 10Hz, considerando que permitía obtener mediciones a velocidades similares a las de los otros sensores y tampoco comprometía el funcionamiento del microcontrolador que debía encargarse de muchas otras tareas.

Para leer la información de estos sensores, que pasando por el conversor USB-UART, es recibida por medio del puerto USB, se realizó un sub-programa en C++ muy simple que abre el puerto donde se realizó la conexión, lee las lineas una por una y las escribe en un archivo de formato texto en un directorio llamado "motorsçon la estructura siguiente:

<timestamp> Motors: <mensaje>

Donde mensaje sería la linea recibida con la estructura explicitada en la Figura 26

Un esquema en diagrama de fujo del programa puede verse en la Figura 27

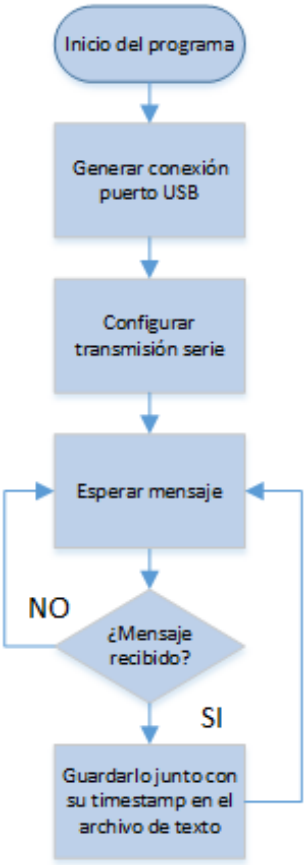


Figura 27: Esquema en diagrama de flujo del programa para leer y guardar la información de los motores.

3.5. Puesta en funcionamiento enlace WiFi

Como se mencionó previamente, para comunicar los dos módulos GPS entre sí y con la PC montada sobre el robot, y para comunicarse con esta última de forma remota, fue necesario establecer una red WiFi utilizando los dos routers MikroTik.

Se generó una red cuyo nombre es robot_desmalezador a partir del router Metal que es quién realizará posteriormente la asignación de IP's fijos a los dispositivos para facilitar la comunicación. La red se generó en 2,4GHz para evitar problemas de compatibilidad de otros dispositivos, y dado que en el área rural no habrá una congestión del ancho de banda por otros dispositivos, esto no sería un problema.

En primer lugar se creo un bridge entre el puerto ethernet y el wlan (wireless lan) dado que en este router en particular los mismos son dispositivos separados. Este bridge (puente) es el que funcionará como DHCP y entregará IP a los clientes que se conecten a la red. Los detalles de la red generada desde el router Metal son los siguientes:

- Nombre red: robot_desmalezador
- DHCP server: Bridge1 (puente entre el puerto ethernet y wlan)

- wlan: ap bridge, 2Ghz-B/G/N - 20/40MHz Ce
- Dirección de la red: 192.168.88.0
- Gateway: 192.168.88.1
- Mascara de red (Netmask): 255.255.255.0
- DHCP pool: 192.168.88.10 - 192.168.88.20
- WDS: dynamic bridge1

Para incluir el segundo router (Groove) en la red, aumentando el alcance de la misma e integrando los dispositivos montados en el robot, se creó un puente WDS (Wireless Distribution System) entre ambos routers. El mismo se configuró de forma dinámica.

La configuración de este router es similar a la primera, nuevamente se generó el puente entre el puerto ethernet y el puerto wlan y la configuración resultante es la siguiente:

- Nombre red: robot_desmalezador
- wlan: ap bridge, 2Ghz-B/G/N - 20/40MHz Ce
- Dirección de la red: 192.168.88.0
- WDS: dynamic bridge1

Con esta disposición es el router que se encuentra en la base quien genera la red y provee de IP a cada unos de los dispositivos. Es por esto que, como se mencionó previamente, a partir de este se asignaron IP fijos a determinados dispositivos para facilitar las comuniaciones. Ejemplo de esto son los siguientes:

- *Router Metal (en la base):* 192.168.88.1
- *Router Groove (en el robot):* 192.168.88.5
- *Módulo GPS en la base:* 192.168.88.14
- *Módulo GPS en el robot:* 192.168.88.15
- *PC NUC:* 192.168.88.10

Estas asignaciones eran necesarias para la lectura del puerto TCP con el cual se obtienen los datos del GPS montado en el robot desde la PC, para el enlace de comunicación entre ambos módulos GPS y para poder ejecutar comandos sobre la NUC de forma remota por SSH (Secure Shell).

Por otro lado, se realizaron determinados filtros por MAC sobre algunos de los dispositivos desde los routers utilizados. Esto se realizó con el objetivo de mantener los dispositivos del robot siempre conectados al router del robot, mientras que los dispositivos de la base siempre conectados al router de la misma. Si bien esto ocurriría de forma usual sin necesidad de intervenir, al pasar el robot cerca de la base podría realizarse la desconexión y posterior conexión (para

pasar de un enlace con uno de los routers al otro) de, por ejemplo, algún módulo GPS generando así una interrupción en el enlace que conllevaría a la perdida de información.

Para evitar todo tipo de problemas, se agregaron a los routers filtrados por MAC que eviten la conexión del dispositivo en cuestión. Quedando como resultado que el router de la base no permite la conexión del GPS del robot, mientras que el router del robot no permite la conexión del GPS de la base. La PC NUC no debió ser filtrada debido a que su conexión sería realizado por Ethernet.

3.5.1. Lectura de todos los sensores

Al disponer de sub-programas que permitan obtener la información de cada uno de los sensores, se buscó facilitar la utilización de los mismos de forma remota. Es por esto que se dispuso en primer lugar, que la PC montada pueda ser encendida remotamente utilizando la red WiFi generada. Para poder hacerlo se utiliza el comando "wakeonlan"seguido de la dirección MAC de la NUC, desde la terminal de Linux de otra computadora.

Por otro lado, considerando que la comunicación con la NUC se realizaría por SSH, se creó un pequeño programa de bash (*Bourne-again shell*) denominado *"sensors-recorder"*. Estos denominados "scripts de bash"son un conjunto de instrucciones y comandos que permiten automatizar determinadas tareas. En este caso dicho conjunto de instrucciones permite ejecutar todos los otros programas, enviándole todos los parámetros que necesiten para su ejecución. Además este script, previo al guardado de la información, creará los directorios donde se guardarán las imágenes y archivos de texto con las lecturas de los sensores, permitiendo que la información se guarde de forma ordenada, como se muestra en la Figura 28.

El directorio "datasets"sera creado en el directorio "home", solamente si no ha sido creado aun, en caso de que ya existiera, se lo utilizará incorporando las carpetas internas a medida que los sets de datos sean generados. De esta forma, se creara una carpeta por cada vez que se ejecute el programa de bash que utiliza todos los sensores, la misma tendrá como nombre el día y horario en que fue ejecutado. Dentro del mismo se crean las carpetas correspondientes a cada sensor y en su interior estarán los archivos correspondientes.

Cabe destacar que en la carpeta tara están las imágenes de la cámara ya que el programa utilizado las obtiene, sin embargo las mismas no serán utilizadas, sólo se hará uso de la información obtenida de la IMU incorporada.

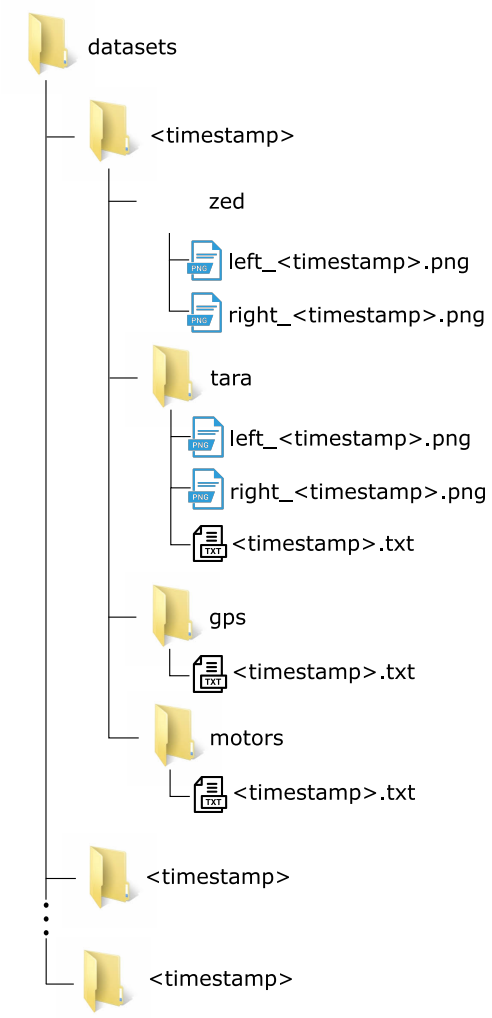


Figura 28: Esquema de como se ordenan los directorios y archivos de la información obtenida de los sensores.

3.6. Montaje sensores

Al disponer de todos los sensores en condiciones de ser utilizados y recuperar su información, fue necesario implementar los mismos sobre el robot, de forma que pudiera trabajar en el entorno agrícola y realizar pruebas en el lugar de trabajo con todos los sensores en funcionamiento.

Para cada uno de los sensores se realizó de forma diferente, pero con el criterio único de que las posiciones de los mismos no variara con respecto a los otros durante un mismo ensayo. Esto es importante porque posteriormente se proporcionará la información de la posición y orientación relativa de cada uno de los sensores para poder comparar o combinar la información obtenida de cada uno de ellos.

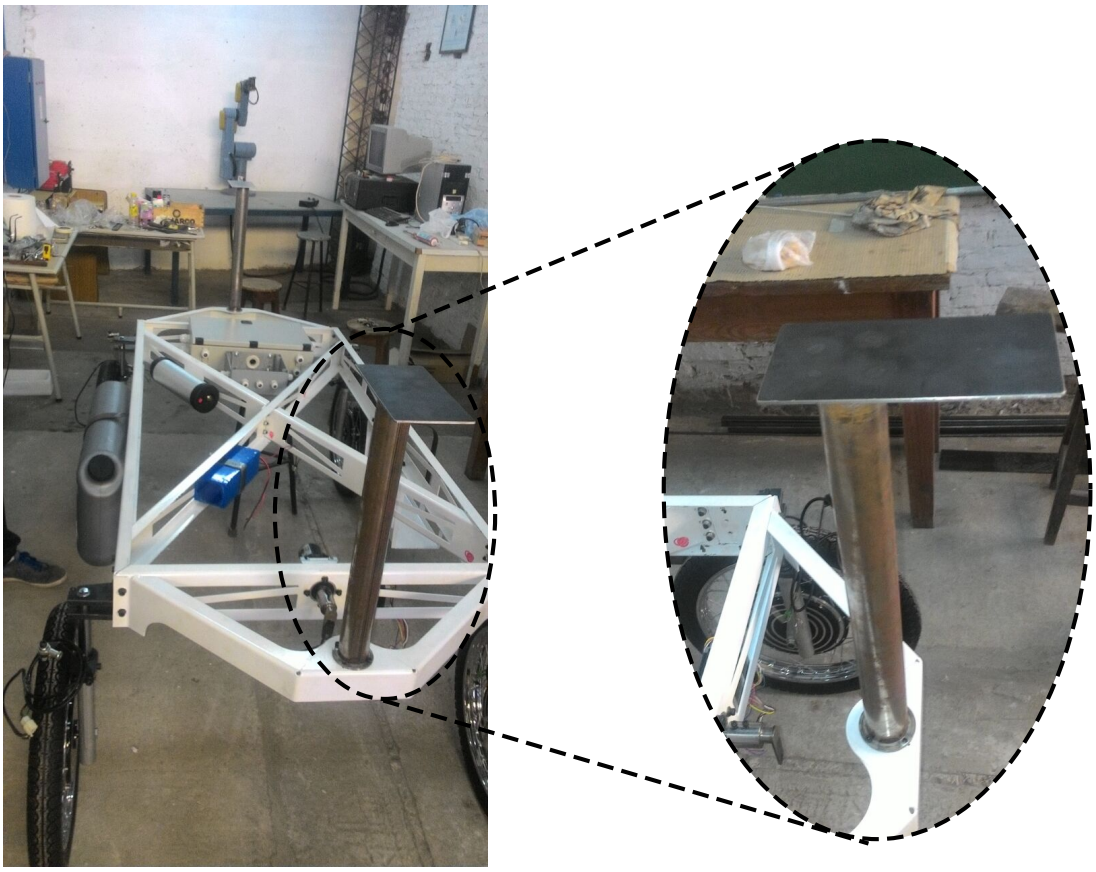


Figura 29: Robot sin los paneles solares donde se pueden observar las torres en la parte delantera y trasera donde serán colocados los sensores.

Para ubicarlos sobre el robot, el mismo cuenta con dos torres, una en la parte delantera y otra en la parte trasera. Estas torres fueron utilizadas como bases para colocar diferentes sensores según fuera conveniente. Las mismas pueden verse en la Figura 29. En un principio ambas torres fueron idénticas, con la forma que puede apreciarse en la imagen, pero posteriormente se modificó la torre delantera.

3.6.1. Montaje torre delantera

Para la torre delantera se incorporó una "L"de metal como puede apreciarse en la Figura 30 con el objetivo de permitir la inclusión tanto de los sensores como de las cámaras y sus respectivos gimbals utilizados por el grupo de detección de malezas.

Figura 30: Torre delantera del robot con el agregado de la L para incluir los sensores y también los Gimbals que se ven en la foto para las cámaras del grupo de detección de malezas.

La cámara se colocó en la torre delantera como puede verse en la Figura 31, la misma tiene visión hacia el frente del robot y, aprovechando punto pivot que la misma posee para modificar su orientación, puede variarse el ángulo de inclinación de la misma.

Sobre la Zed, se colocó la cámara tara que contiene la IMU integrada, el objetivo de esto es mantener la unidad inercial junto a la cámara utilizada en todo momento para que la información obtenida de ambas pueda ser combinada de manera simple.

Por encima de ambas, en la parte superior de la "Lïncorporada, y separada por un varilla vertical, se agregó la antena del GPS. La misma consiste en el receptor de la señal y una planchuela de 20cm x 20cm cuyo objetivo es mejorar la recepción de la señal y evitar posibles interferencias generadas por dispositivos electrónicos ubicados en las cercanías.

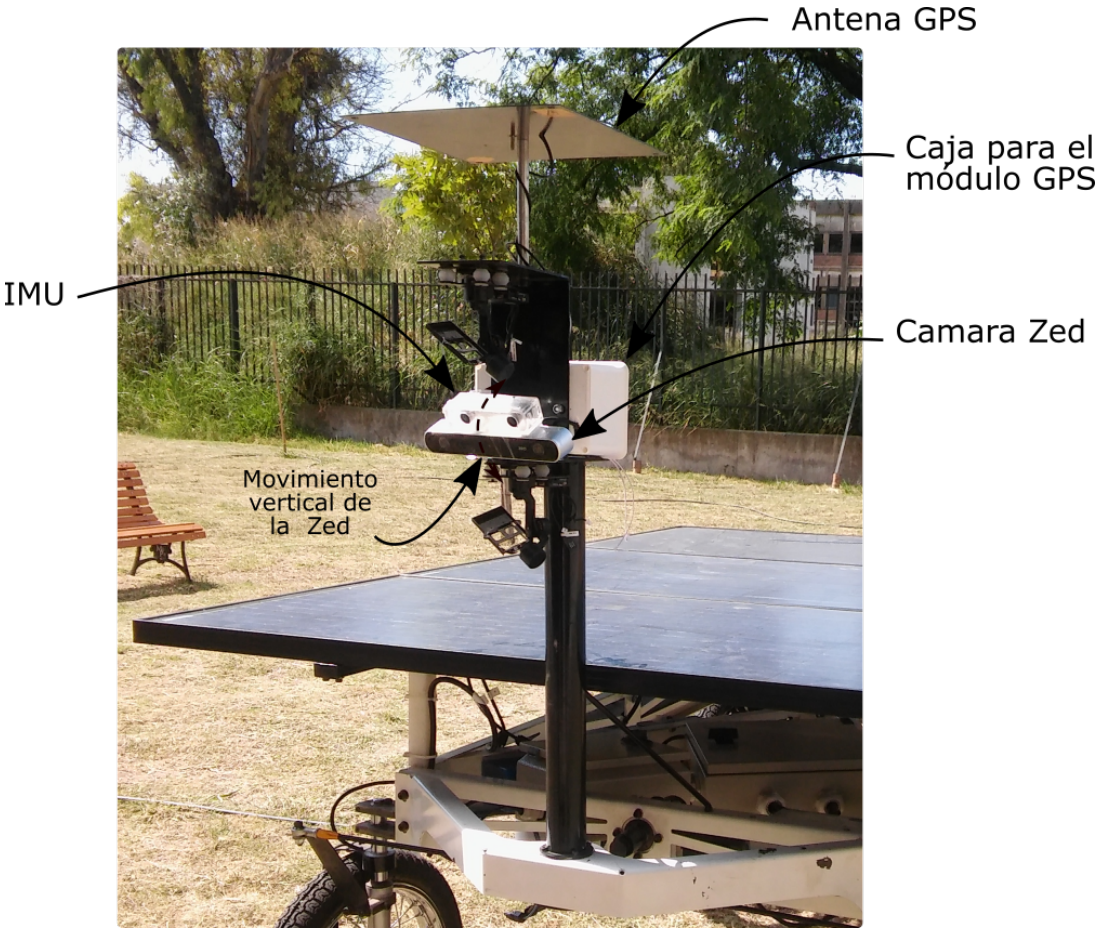


Figura 31: Torre delantera del robot con los sensores incorporados.

Apoyada sobre la L, como se indica en la Figura 31 se encuentra una caja plástica utilizada para ubicar el módulo GPS, cuya distancia a la antena debe ser de varios centímetros, pero no excesiva porque dificultaría la transmisión de la información recibida por la antena. Esta caja se implementó en plástico ya que este módulo debe conectarse por WiFi con el router del robot, de forma que si fuera de metal, además de un costo y peso mayor, podría dificultar la recepción de la señal.

3.6.2. Montaje torre trasera

Por su parte la torre trasera se mantuvo como en la Figura 29 ya que solamente se debía incorporar el router trasero del robot, por lo que la única modificación realizada fue el agregado de una varilla metálica en posición vertical para poder colocar el router de esta manera, mejorando la visibilidad de la antena del mismo misma. Se evitó colocar más elementos en esta torre para evitar interferir con la señal de WiFi, ya que mantener el enlace estable a largas distancias era una de las prioridades.

Figura 32: Torre trasera del robot con el router incorporado.

3.6.3. Montaje estación base

Como se mencionó previamente, el GPS al ser diferencial requiere de otro módulo en una ubicación fija actuando como referencia. Por esto se diseñó una estación base donde se utilizó el modulo GPS Reach, junto con la antena, de forma similar a como se implementó en el robot.

Debido a que se necesitaba más de un metro de altura para la ubicación de la antena del robot se implementó, en primer lugar, un trípode que permita regular la altura de la misma manteniendola estable. Sobre la parte superior del trípode se colocó una caja plástica con el módulo GPS, de idéntica forma a lo realizado en el robot. Finalmente en la parte superior de la caja, y separado nuevamente por una varilla metálica en posición vertical, se colocó el receptor de la señal GPS, ubicado sobre una lamina metálica de (20x20) cm que evite posibles interferencias en el receptor GPS. La estructura resultante puede verse en la Figura 33.

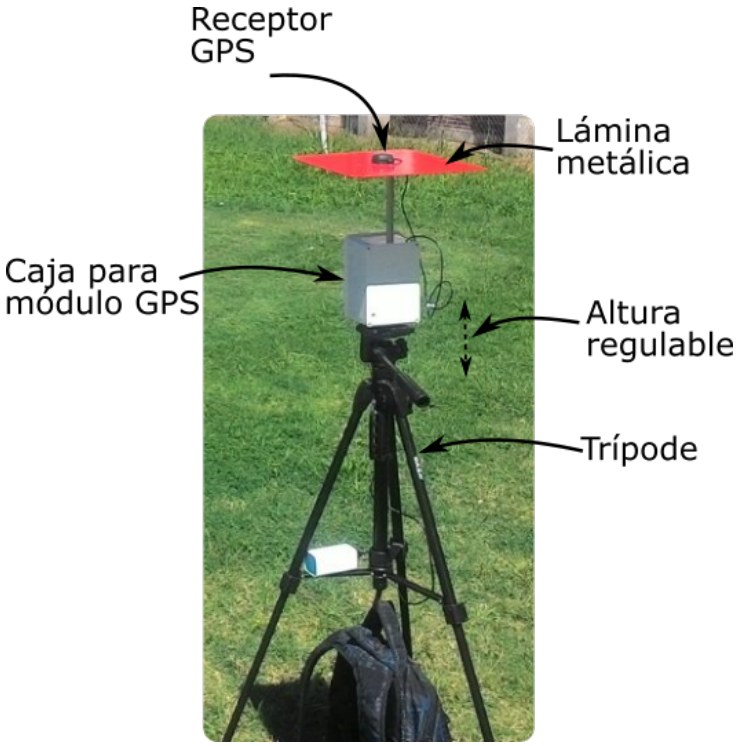


Figura 33: Estación base diseñada para utilizar el módulo GPS de base.

Como se observa en la imagen, la caja plástica se encuentre ajustada al trípode, y es en esta donde se colocaron el módulo GPS para protegerlo junto con un módulo transmisor receptor de radio, que no fue utilizado, pero representa una alternativa para realizar la comunicación entre los dos Emlid Reach. Esta disposición puede verse en la Figura 34, el módulo esta conectado a la alimentación, al receptor GPS sobre la antena y puede conectarse al módulo radio si se desea.

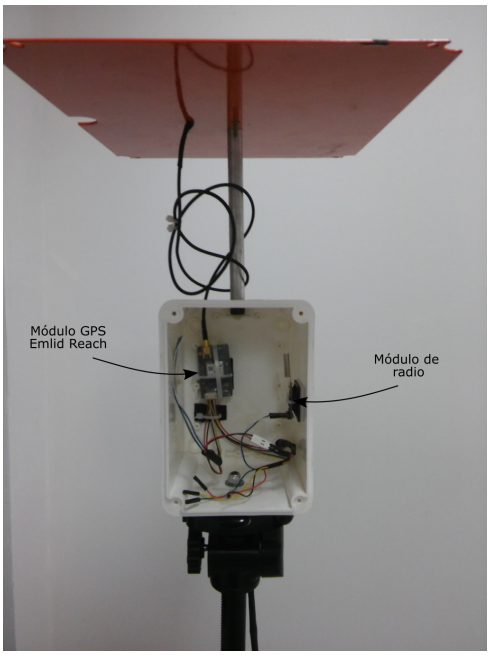


Figura 34: Interior de la caja de la estación base con el módulo GPS y el módulo radio agregado pero no utilizado.

3.6.4. Montaje Computadora Intel NUC

Para colocar la computadora a bordo del robot, se necesitaba un entorno cerrado para evitar el ingreso de tierra u otros residuos y debía ser de metal ya que de esa forma se mejora la disipación térmica. Por estas razones se utilizó una caja metálica de 20cm x 20cm x 15cm que fue colocada debajo de los paneles, adherida a la estructura que sujeta una de las baterías, como se ilustra en la Figura 35

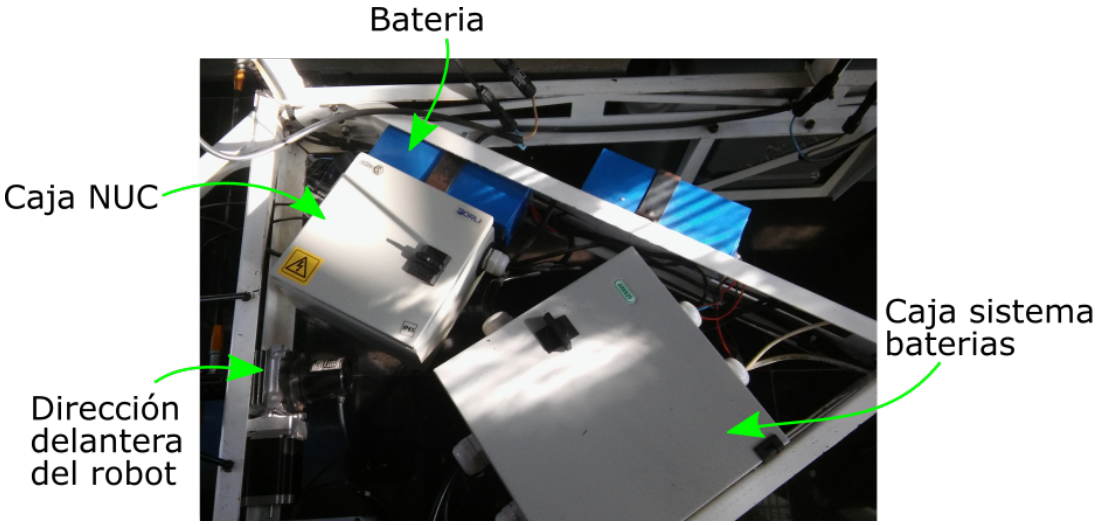


Figura 35: Ubicación de la caja donde se colocó la NUC, la misma esta unida a la estructura que sujeta la batería.

3.6.5. Alimentación y conexión de todos los sensores

Una vez que todos los sensores fueron ubicados en su posición definitiva, se obtuvo como resultado la distribución de los mismos sobre el robot como se muestra en la Figura 36. Cabe destacar en la imagen los sensores de efecto hall de las cuatro ruedas incorporados en las mismas, y el encoder absoluto acoplado a la reducción del paso a paso de la dirección para la lectura de la misma. Estos sensores fueron incorporados por el grupo de Tracción y Dirección del robot, y la información será obtenida por medio de la comunicación establecida entre el microcontrolador que ellos incorporaron y la NUC.

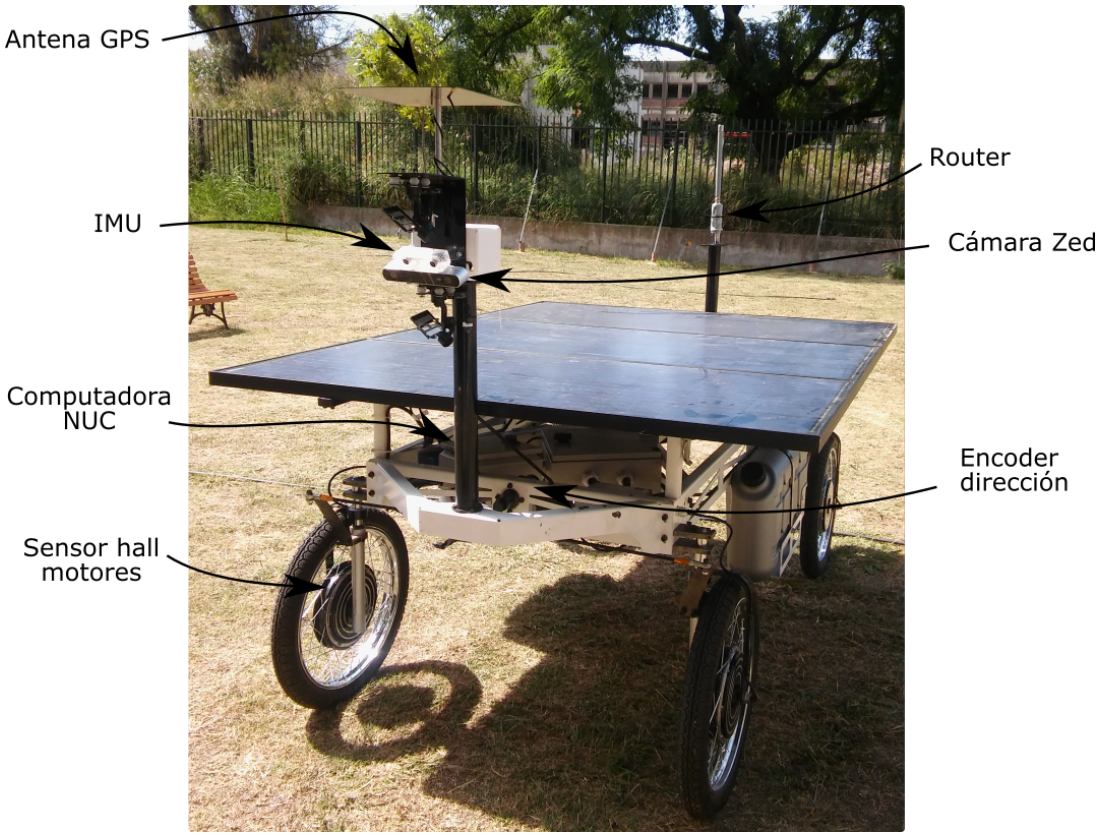


Figura 36: Esquema con todos los sensores, computadora y router colocados en el robot.

El paso siguiente fue proveer de la alimentación de cada uno de ellos de forma correcta según las especificaciones de los mismos, los requerimientos eran los siguientes:

- NUC: 12V y conexión ethernet con el PoE (Power over ethernet)
- GPS Emlid: 5V y conexión a la antena de GPS
- Camara Zed: Conexión USB con la NUC
- IMU: Conexión USB con la NUC
- Router: Conexión Ethernet al PoE (Power over ethernet)

Disponiéndose para realizarlo de las tensiones disponibles en el robot las cuales son 5V, 12V y 36V en la caja de potencia del robot. Por esto se tomaron las tensiones de 5 y 12 voltios, llevándolas hasta la caja de la NUC junto con la conexión de masa. Es en este punto donde se distribuyeron hacia la NUC, el conector PoE y la torre delantera, donde se ubica el módulo GPS.

Posteriormente se realizaron las conexiones de los dos cables USB desde la NUC hasta la cámara Zed y la IMU, pasando por el interior de la torre delantera y la caja plástica del GPS. En cuanto a la torre trasera, se conectó el router por medio de un cable Ethernet, que pasando por el interior de la torre se conecta al PoE que está ubicado en el interior de la caja de la NUC.

La última conexión en realizarse fue la comunicación serie entre el microcontrolador, ubicado en la caja de control del robot, con la NUC. Para esta se enviaron los cables de Tx, Rx, y masa envainados que, dentro de la caja de la NUC, serían conectados a un conversor USB-UART que permitió la conexión con el puerto USB de la NUC.

En la Figura 37 se ilustra el resultado final de las conexiones realizadas para proveer de tensión a los sensores y relacionarlos con la computadora a bordo.

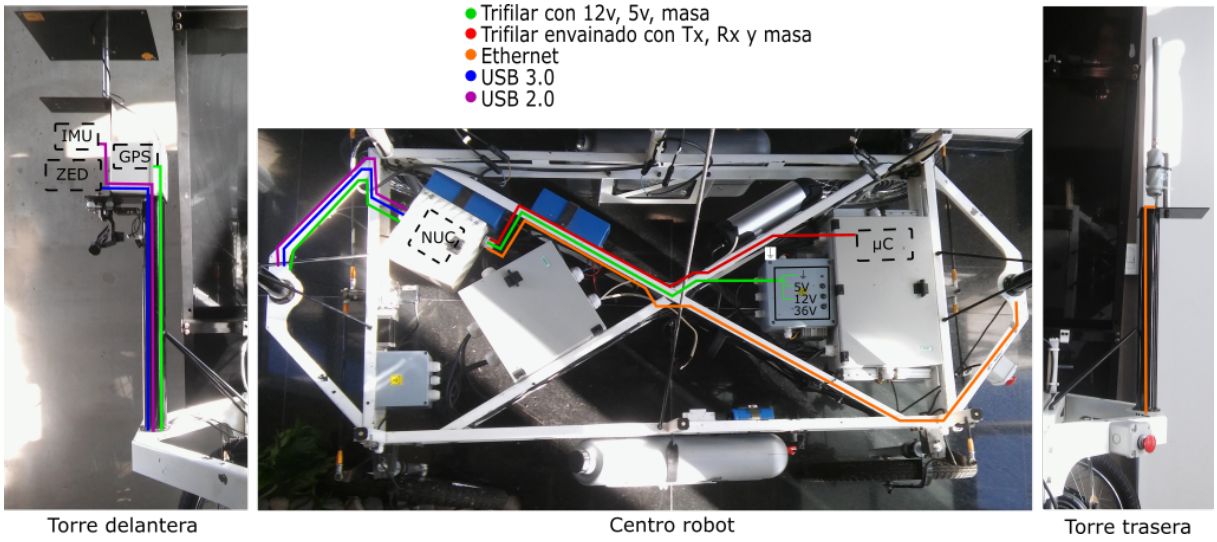


Figura 37: Esquema de las conexiones entre los sensores y la alimentación de los mismos sobre el robot.

Por otro lado, en la Figura 38 puede verse el interior de la caja, conteniendo a la computadora NUC junto con las conexiones realizadas entorno a ella. Por un lado la alimentación que va a una pequeña bornera para distribuirse hacia el conector PoE, a la NUC y saliendo de la caja hacia la torre delantera. Además el cable Ethernet conectado al Poe y el adaptador USB-UART conectado al puerto de la computadora, permitiendo la conexión serie con el microcontrolador.

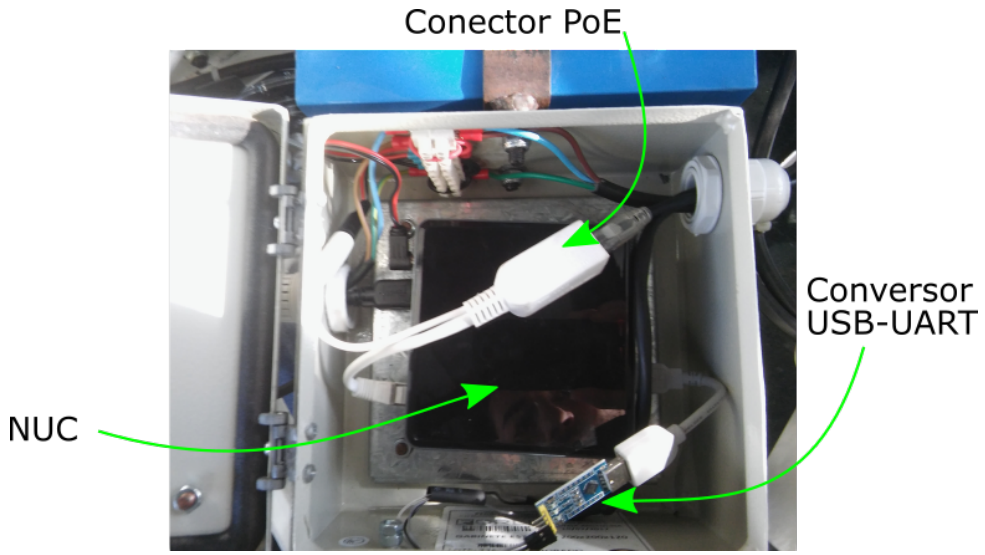


Figura 38: Interior caja NUC, donde puede verse la computadora con sus conexiones y cables de alimentación.

Adicionalmente al conexionado de los sensores sobre el robot, también se debieron realizar las conexiones y la alimentación del módulo de GPS de la base. Esto está representado en la Figura 39. El módulo necesita la alimentación de 5V y el router de la base su conexión Ethernet por medio del conector PoE.

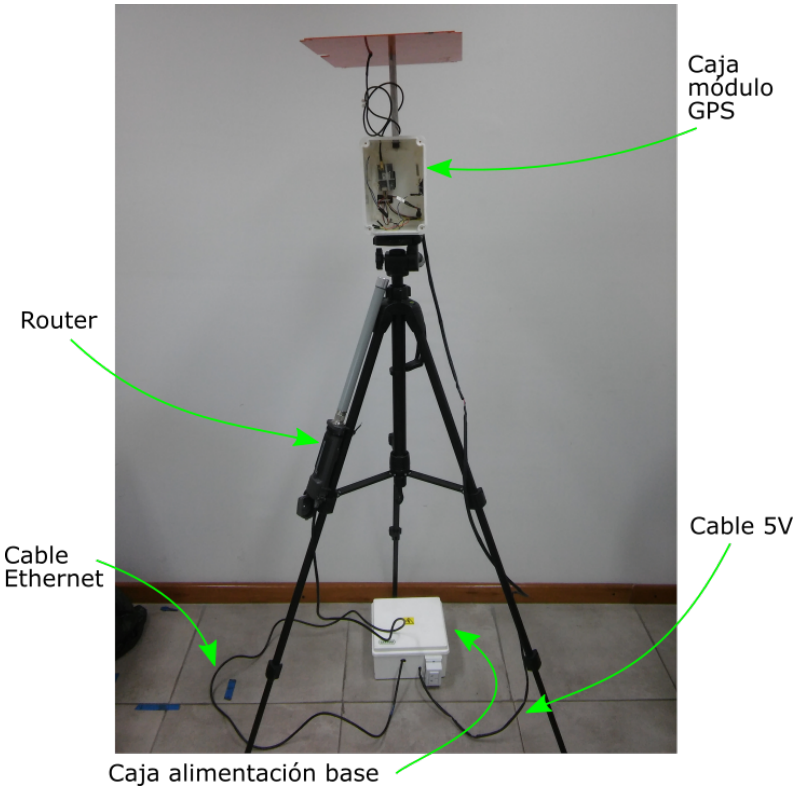


Figura 39: Estación base compuesta del trípode, caja del módulo GPS, caja de alimentación, router Wifi y las conexiones entre ellos.

Para alimentar los sensores de la base se utilizó una batería de 12V, conectada en paralelo al PoE para alimentar el router, y a un inversor que posee como salidas 220 voltios de corriente alterna y un conector USB de 5 voltios. Este último es el que será utilizado para alimentar el módulo GPS. En cuanto a la salida de 220 voltios, si bien no es utilizada por ninguno de los sensores, está a disposición para conectar otros dispositivos como cargadores de celulares o de notebooks (siempre y cuando la batería pueda proporcionar suficiente corriente y a su vez no se descargue antes de finalizar el trabajo). Estas conexiones realizadas en el interior de la caja, pueden verse en la Figura 40.

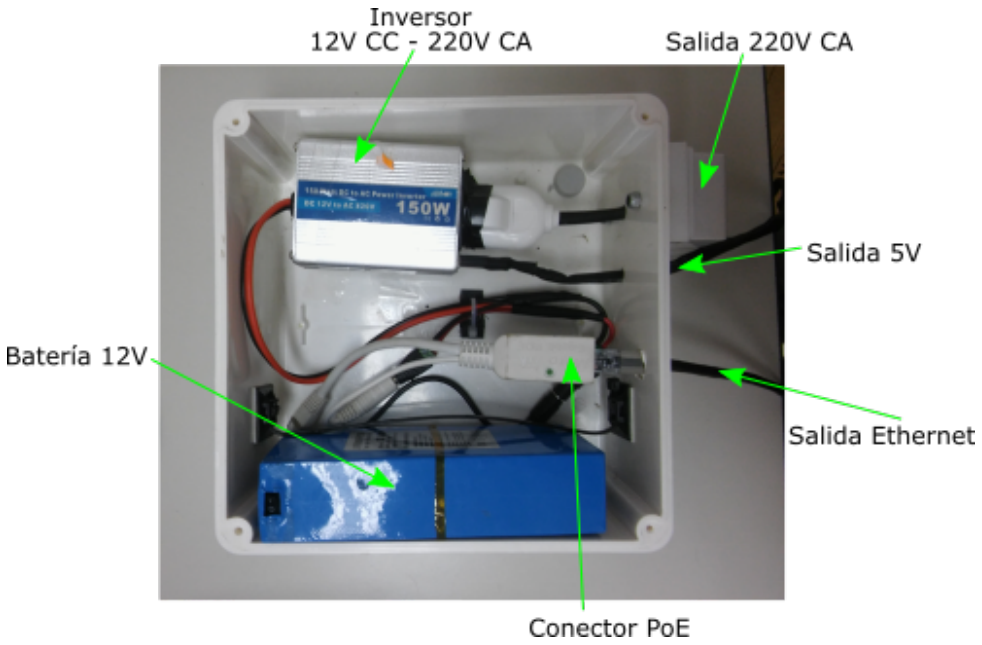


Figura 40: Interior de la caja de la base donde se encuentra la batería para alimentar a los sensores de la base y sus conexiones.

3.7. Obtención del set de datos en condiciones reales de funcionamiento

Al tener todos los sensores en funcionamiento sobre el robot y los programas para la obtención de datos y guardado de los mismos terminados, se procedió a realizar varias pruebas dentro del predio de la Siberia para corroborar que el funcionamiento sea correcto.

Posteriormente, durante el mes de Diciembre, se realizaron las pruebas con el objetivo de obtener los datos que permitieran el desarrollo de los algoritmos de localización, mapeo, navegación y detección de malezas. Estas pruebas en condiciones reales de trabajo se realizaron en los terrenos pertenecientes a la Facultad de Ciencias Agrarias de la Universidad Nacional de Rosario - Campo Experimental Villarino, Zavalla. Las mismas consistieron en la utilización del robot junto con todos sus sensores durante alrededor de 6 horas repartidas en dos días. Cabe destacar que como esta fue la prueba inicial, el objetivo era comprobar el funcionamiento y obtener el set de datos mencionado, el robot fue controlado mediante el control remoto incluido por el grupo del sistema de Tracción y Dirección del robot.

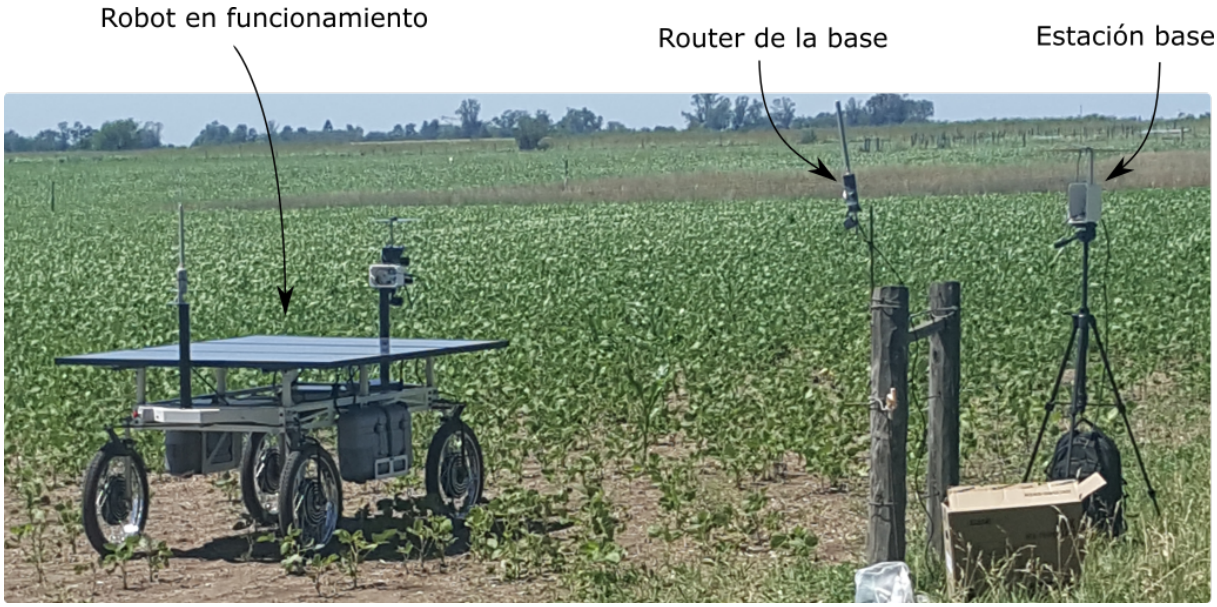


Figura 41: Esquema con todos los sensores, computadora y router colocados en el robot.

Todos los elementos y sistemas en funcionamiento, en el ambiente rural pueden observarse en la Figura 41 mientras que en la Figura 42 pueden apreciarse otras imágenes del robot recorriendo los surcos de la plantación de soja del campo de Zavalla.

Figura 42: Imagen del robot en el campo con todos los sensores en funcionamiento junto a la estación base.

A partir de estos dos días de recolección de datos se pudieron obtener 54 Gb de información correspondientes a aproximadamente una hora neta de adquisición de datos. Estos, fueron obtenidos en 8 trayectos distintos, donde el entorno variaba ampliamente, desde surcos bien marcados hasta zonas donde la altura de las plantas de soja dificultaba detectar el camino.

Estas gran variedad de imágenes obtenidas pueden observarse en la Figura 43

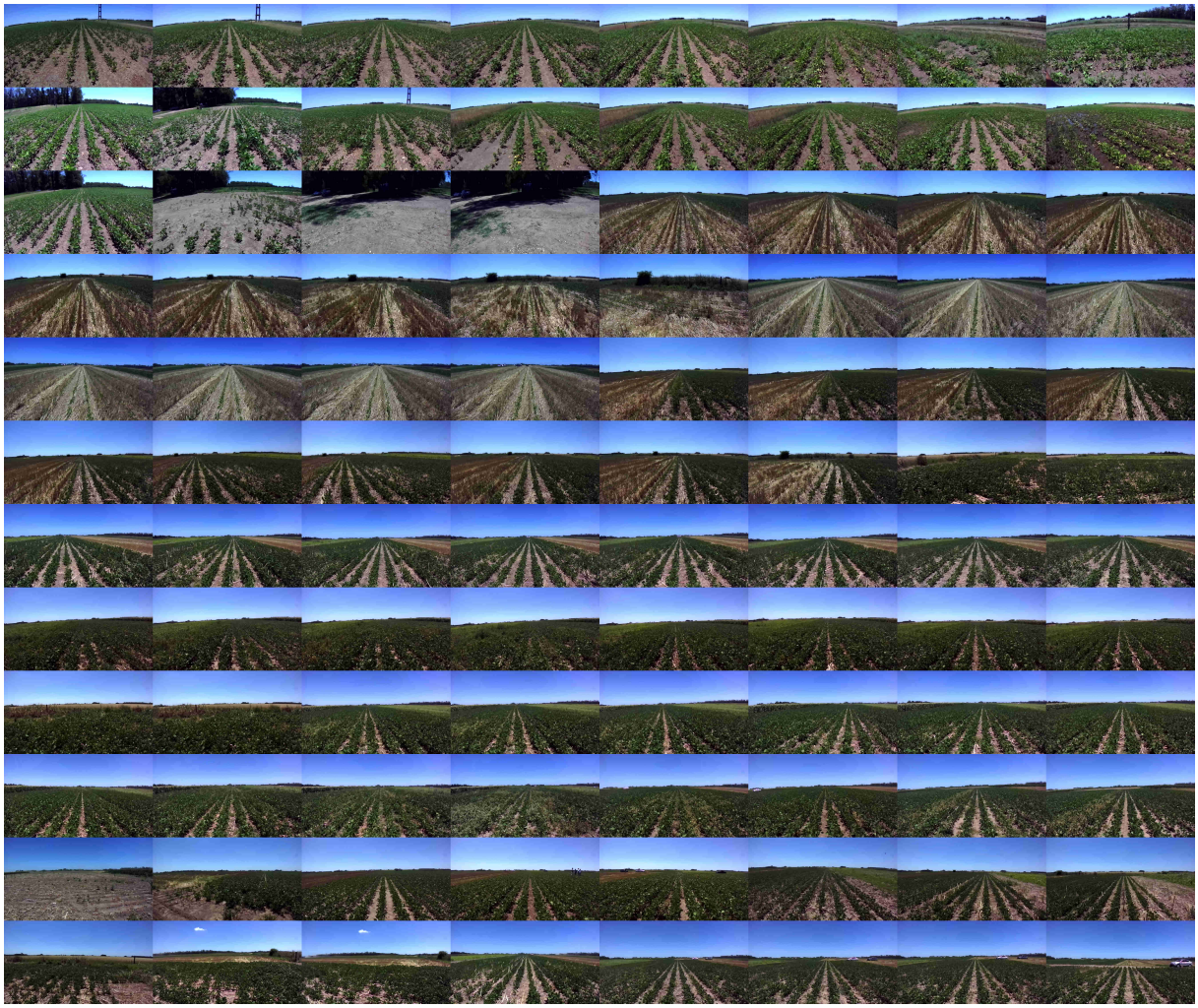


Figura 43: Imágenes del robot desplazándose por el campo durante uno de los días de adquisición de datos.

3.8. Procesamiento del set de datos

Una vez obtenidos los sets de datos, se procedió a seleccionar las mejores partes, es decir, donde el funcionamiento fue óptimo y no surgieron inconvenientes de ningún tipo. A partir de esto se conservaron 6 secuencias que en su total representan una duración de 34,7 minutos.

Los datos de cada uno de los sensores fueron obtenidos en un formato determinado por su simpleza para el guardado, sin embargo, fue necesario posteriormente convertirlo y adaptarlo para que sigan los estándares definidos en robótica y de esta forma, que este set de datos sea utilizable por otras personas. Adicionalmente, esto permitiría que utilicemos algoritmos ya creados (como algunos de SLAM) donde la información de entrada debe estar en los formatos definidos por ROS.

3.8.1. Sistema Operativo Robotico (ROS)

El sistema operativo robótico (ROS, por sus siglas en inglés) es un framework orientado al desarrollo de software para robots. El mismo posee las características de un sistema operativo como: abstracción del hardware, paso y mantenimiento de paquetes, control de dispositivos de bajo nivel, etc. Su funcionamiento esta basado en una estructura de grafos, donde el procesamiento toma lugar en los nodos, los cuales envían y reciben mensajes por medio de canales o tópicos (topics en ingles).

Esta estructura presenta una gran ventaja ya que permite el funcionamiento simultaneo de gran cantidad de nodos, donde uno podría publicar su información en un tópico que a su vez podría ser leído por todos los demás. Este funcionamiento se puede observar en la Figura 44. Cabe destacar que la información se publica en forma de mensajes, lo cual presenta una nueva ventaja de trabajar con este formato ya que una gran cantidad de mensajes están estandarizados permitiendo la re-utilización de programas y evita problemas de compatibilidad en la comunicación entre nodos.

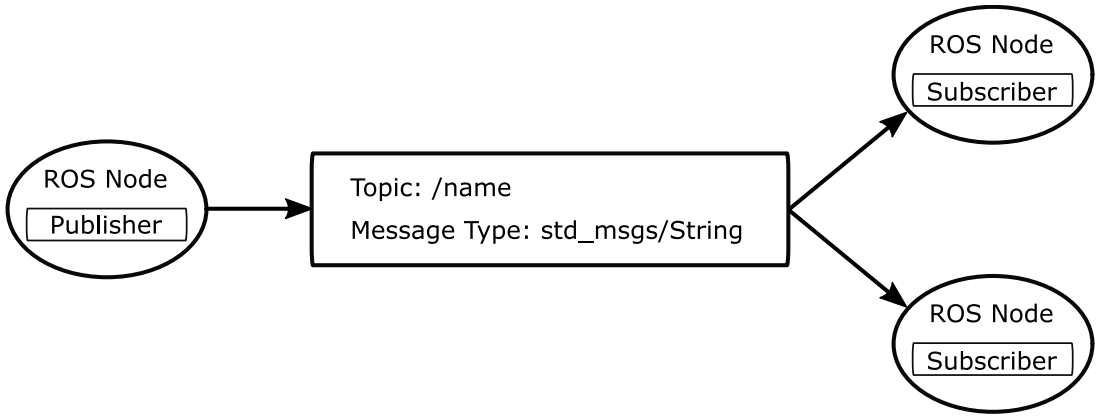


Figura 44: Esquema de funcionamiento de la metodología publicar-suscribirse utilizada por ROS para el manejo de la información.

Por otro lado ROS presenta modalidades para visualizar y leer información que pueden ser útiles en el campo de la robótica, ejemplo de esto, y que sera utilizado en este proyecto, es la posibilidad de simular tiempo real. Esto se realiza indicándole a ROS que se trabajara con los tiempos que aparezcan en los mensajes y no con el tiempo del reloj actual, de forma que si dos mensajes contenían un segundo de diferencia, serán publicados con un segundo de diferencia en el tópico. Para realizar esto se utiliza también lo que se denomina Rosbag"que es un conjunto agrupado de mensajes con su timestamp que serán publicados en tópicos designados, esto es útil si, como en nuestro caso, se posee un set de datos y se desea guardarlo en un formato que pueda ser utilizado por distintos nodos. Por lo tanto como se detallará posteriormente nuestros set de datos será guardado en un Rosbag que publicará en diferentes tópicos (por tener información de muchos sensores) donde cada mensaje será una lectura del mismo. De esta forma será posible utilizar nodos que se suscriban a esta información y observar como operan sobre la misma en un tiempo real simulado.

A partir de esta estructura de funcionamiento el objetivo sobre el que se trabajó en esta etapa fue convertir las lecturas de cada uno de los sensores en mensajes que siguieran los formatos ya definidos para los mismos por la comunidad de ROS y guardarlos en un Rosbag que pudiera

ser utilizado por otros algoritmos que funcionarán como nodos.

3.8.2. Creación del Rosbag

Se diseñó un programa en Python que permite obtener información de una secuencia de datos guardada, la cual consiste en las imágenes de la cámara Zed, la información de la IMU, de los motores y del GPS, y las guarda en un rosbag como mensajes (en distintos tópicos), cada uno con el timestamp del momento en el que fue adquirido.

El programa requiere como parámetros de entrada:

- Carpeta con las imágenes de la cámara donde cada una este nombrada:

  $$<cámara>\_<timestamp>.png$$

- Archivo de texto de las mediciones de la IMU donde cada linea de información tenga el siguiente formato:

  $$<timestamp>\ IMU:\ <GyroX>\ <GyroY>\ <GyroZ>\ <AccX>\ <AccY>\ <AccZ>$$

- Archivo de texto con las mediciones del GPS con cada linea de información con el siguiente formato:

  $$<timestamp>\ GPS\text{-}RTK:\ <sentencia\ NMEA>$$

- Archivo de texto con las mediciones de los motores con cada linea de información con el siguiente formato:

  $$<timestamp>\ Motors:\ <mensaje>$$

- Archivo de tipo yaml con las calibraciones, ajustes y transformaciones entre los sensores. [5]

- Archivo tipo .bag de salida, que sera el rosbag conteniendo todos los mensajes en el formato utilizado por ROS

Por otro lado la salida, como se mencionó, es el rosbag generado con toda la información expresada en mensajes con su timestamp para cada sensor. Por lo tanto el rosbag generado tendrá los siguientes tipos de mensajes:

- **sensor_msgs/Image.msg:** Para cada una de las imágenes (izquierda y derecha), obtenidas de la cámara Zed se genera un mensaje que contiene la siguiente información:
  - Header: encabezado con el número de secuencia de los mensajes, timestamp y identificador.
  - Height: altura de la imagen en número de filas
  - Width: ancho de la imagen en número de columnas

[5] Estas calibraciones y transformaciones son detalladas en la sección 3.8.3

  - encoding: codificación de los pixeles
  - Is_bigendian: Según sea los datos sean o no representados en formato natural
  - Step: longitud en bytes de una fila
  - Data: matriz de datos de la imagen

- **sensor_msgs/CameraInfo.msg:** Con la información de los parámetros intrínsecos y extrínsecos de ambas cámaras de la Zed necesarios para el procesamiento de las mismas. Cada mensaje contiene los siguientes campos:

  - Header: encabezado con el número de secuencia de los mensajes, timestamp y identificador.
  - Height: altura de la imagen en número de filas
  - Width: ancho de la imagen en número de columnas
  - Encoding: codificación de los pixeles
  - Is_bigendian: según sea los datos sean o no representados en formato natural
  - Step: longitud en bytes de una fila
  - Data: matriz de datos de la imagen

- **sensor_msgs/Imu.msg:** Los mensajes de las mediciones del acelerómetro y el giróscopo que contienen:

  - Header: encabezado con el número de secuencia de los mensajes, timestamp y identificador.
  - Orientation: orientación en el espacio del sensor, expresado como un Cuaternión
  - Orientation_covariance: Matriz de covarianza de la orientación
  - Angular_velocity: vector de velocidad angular de la IMU
  - Angular_velocity_covariance: matriz de covarianza de la velocidad angular
  - Linear_velocity: vector de velocidad lineal
  - Linear_velocity_covariance: matriz de covarianza de la velocidad lineal

- **sensor_msgs/NavSatFix.msg:** De la información obtenida del GPS, utilizando los datos de los mensajes "GGA"de las sentencias NMEA. Estas incluyen la siguiente información sobre el posicionamiento y la precisión de las mediciones:

  - Header: encabezado con el número de secuencia de los mensajes, timestamp y identificador.
  - Status: estado del fix obtenido por el GPS
  - Latitude: medición de latitud en grados, positivo es norte de la linea del ecuador
  - Longitude: medición de longitud en grados, positivo es al este del primer meridiano
  - Altitude: medición de altitud en metros por encima del elipsoide WGS 84
  - Position_covariance: matriz de covarianza de la posición
  - Position_covariance_type: tipo de covarianza entre desconocido, aproximado, conocida la diagonal, conocida completamente.

- **nav_msgs/Odometry.msg:** La información de la velocidad de los motores junto con la dirección del robot son traducidas en un vector tri-dimensional para la velocidad lineal y otro para la velocidad angular y se expresan en los siguientes campos:
  - Header: encabezado con el número de secuencia de los mensajes, timestamp y identificador.
  - Child_frame_id: Referencia al identificador del sistema de coordenadas que será su "hijo"(es decir, se extenderá de esta)
  - Pose: posición en x,y,z junto con la orientación expresada como un cuaternión
  - Twist: velocidad tanta lineal como angular
- **tf/tfMessage.msg:** Todas las transformaciones extrínsecas entre los sensores en función de la localización de los mismos sobre el robot. Las mismas son expresadas como un cuaternión de rotación y un vector tri-dimensional de traslación. Cada una de las transformaciones contiene los campos siguientes:
  - Header: encabezado con el número de secuencia de los mensajes, timestamp y identificador.
  - Child_frame_id: identificador del sistema de coordenadas con respecto al cual se aplica la transformación.
  - Transform: vector de traslación y cuaternión de rotación para ir de un sistema de coordenadas al otro

Un esquema simplificado del programa se muestra en diagrama de flujo en la Figura 45.

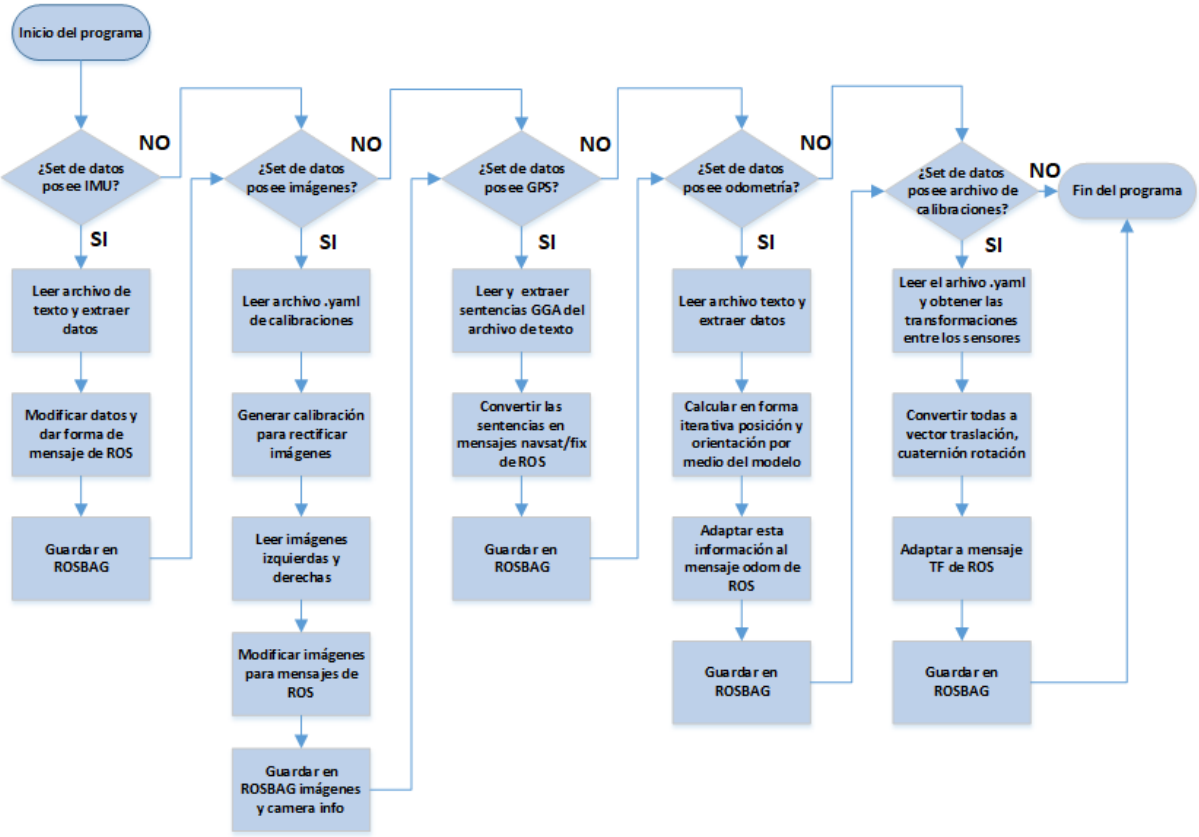


Figura 45: Esquema en diagrama de flujo del programa diseñado en Python para generar los rosbags a partir del set de datos.

A partir de este programa se creó un rosbag para cada una de las secuencias utilizadas de forma de disponer de cada una de ellas para ser utilizadas por nodos (programas que trabajan en el entorno de ROS) que permitan obtener información de la localización, orientación y condiciones de trabajo del robot.

Por ejemplo, uno de los rosbags con una secuencia contiene la información mostrada en la Figura 46

```
duration:    13.1s
start:       Dec 26 2017 12:29:36.87 (1514302176.87)
end:         Dec 26 2017 12:29:49.93 (1514302189.93)
size:        294.7 MB
messages:    3071
compression: none [205/205 chunks]
types:       geometry_msgs/TwistStamped [98d34b0043a2093cf9d9345ab6eef12e]
             nav_msgs/Odometry          [cd5e73d190d741a2f92e81eda573aca7]
             sensor_msgs/CameraInfo     [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/Image          [060021388200f6f0f447d0fcd9c64743]
             sensor_msgs/Imu            [6a62c6daae103f4ff57a132d6f95cec2]
             sensor_msgs/NavSatFix      [2d3a8cd499b9b4a0249fb98fd05cfa48]
             tf2_msgs/TFMessage         [94810edda583a504dfda3829e70d7eec]
topics:      /gps/fix                     66 msgs    : sensor_msgs/NavSatFix
             /gps/vel                     66 msgs    : geometry_msgs/TwistStamped
             /imu                       1863 msgs    : sensor_msgs/Imu
             /odom                       132 msgs    : nav_msgs/Odometry
             /stereo/left/camera_info    203 msgs    : sensor_msgs/CameraInfo
             /stereo/left/image_raw      203 msgs    : sensor_msgs/Image
             /stereo/right/camera_info   203 msgs    : sensor_msgs/CameraInfo
             /stereo/right/image_raw     203 msgs    : sensor_msgs/Image
             tf                          132 msgs    : tf2_msgs/TFMessage
```

Figura 46: Información contenida en un rosbag de una pequeña parte de una de las secuencias.

Puede verse el tipo de mensajes contenidos, los tópicos donde se publicaran y la cantidad de mensajes de cada uno de ellos. Por otro lado también se puede ver la duración del mismo, el instante de comienzo y de fin.

3.8.3. Obtención de la calibración de los sensores

Como se mencionó previamente, el programa desarrollado necesita que se le provea de entrada un archivo en formato ".yamlçon las calibraciones de los sensores. Esta información es necesaria por dos razones:

1. Obtener los parámetros que caractericen al modelo utilizado de la cámara Zed, ya que esta información debe publicarse en los mensajes çamera_info"para poder ser utilizado en conjunto con las imágenes por otro programas.

   En este aspecto se utilizó un modelo de cámara del tipo "pinhole"junto con un modelo de distorsión radial tangencial (o "plumb bob"). Para obtener los parámetros del mismo se realizó la calibración de la cámara, es importante notar que en este punto no se hace referencia a ajustar valores para el funcionamiento de la cámara, sino que el objetivo es obtener los parámetros de calibración del modelo de la Zed utilizado.

   Para realizar la calibración se utilizaron los algoritmos de kalibr[6], el procedimiento consiste en la utilización de la cámara estéreo enfrente de un patrón determinado, como un tablero de ajedrez o, en nuestro caso, de un aprilgrid, como el de la Figura 47.

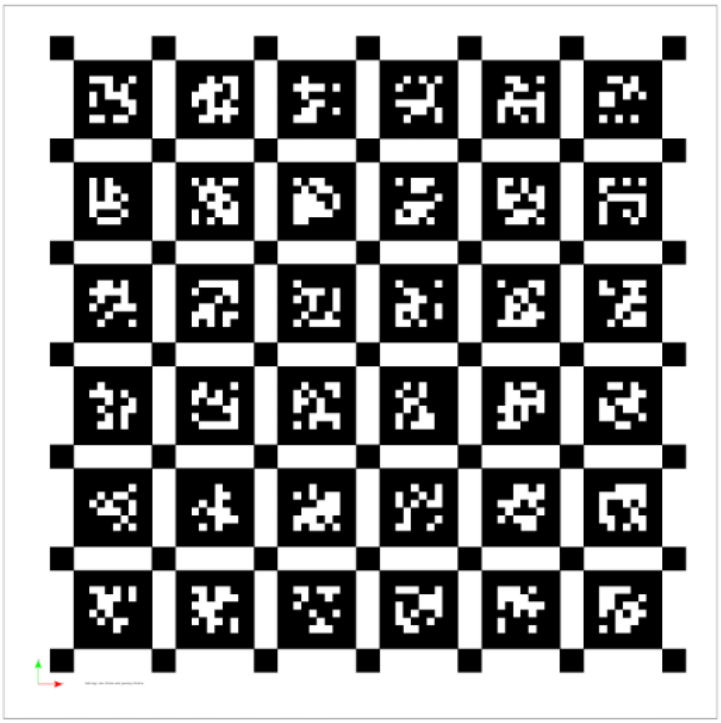


Figura 47: Aprilgrid con los tags utilizados para la parametrización de la camara Zed mediante Kalibr.

[6]https://github.com/ethz-asl/kalibr/wiki/multiple-camera-calibration

A partir del programa de kalibr, que puede detectar los bordes de cada uno de los april tags utilizados mientras la cámara se mueve (o el aprilgrid lo hace) se estiman los parámetros necesarios para la calibración de la cámara. En la Figura 48 se muestra el aprilgrid durante la detección. Una vez terminado se obtiene un archivo del tipo yaml, que en nuestro caso posee los siguientes parámetros

- Cámara izquierda
  - Modelo de cámara: Pinhole
  - Distancia focal: [354.57962540811184, 353.08984763520596]
  - Punto principal: [330.2183813317979, 181.34015968024195]
  - Modelo de distorsión: radial-tangencial (o plumb bob)
  - Coeficientes de distorsión: [-0.1754419005129358, 0.026835506945626218, 0.00295551795749725, 0.0005392680274165596]
- Cámara derecha
  - Modelo de cámara: Pinhole
  - Distancia focal: [354.5232356767637, 354.1796892346424]
  - Punto principal: [324.4316707690572, 197.9043065273593]
  - Modelo de distorsión: radial-tangencial (o plumb bob)
  - Coeficientes de distorsión: [-0.1771689373865328, 0.02803729512231533, 0.002997368118414364, 0.0005088900747159647]

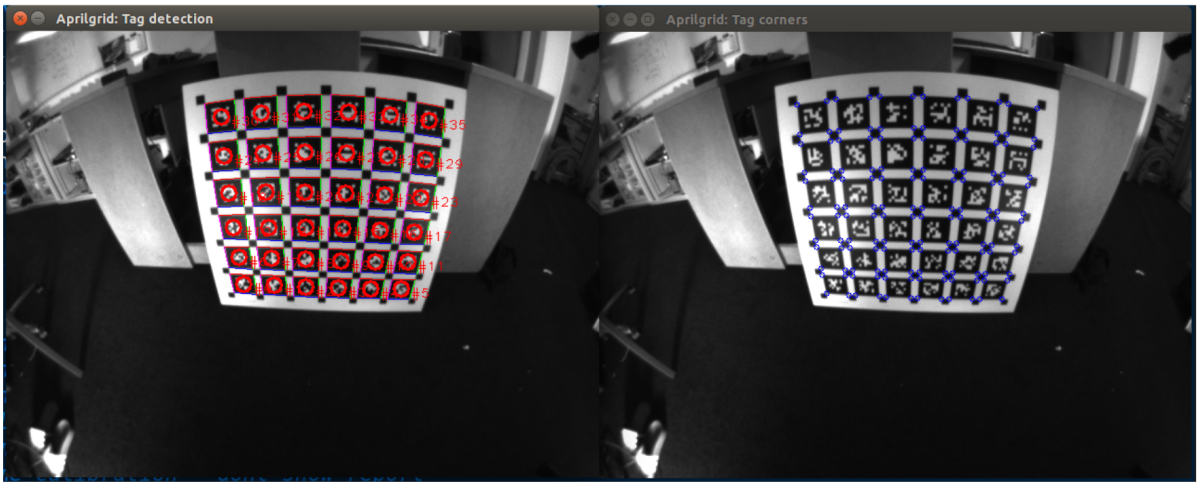


Figura 48: Detección de los tags y sus aristas.

Esta información se incorporara a los mensajes camera_info del rosbag. Por otro lado utilizando este algoritmo se obtuvieron las transformaciones entre el sistema de coordenadas de cada cámara.

2. Obtener las transformaciones que permitan relacionar los sistema de coordenadas de cada uno de los sensores. Esta información es necesaria por ejemplo, si se desea realizar la fusión de más de un sensor para obtener información respecto a la posición del robot de una forma más robusta y confiable.

   La necesidad de estas transformaciones podría ejemplificarse de la siguiente manera: Tanto las cámaras como la IMU obtendrán información respecto al avance del robot, de forma

que si el robot avanza hacia adelante, ambos proporcionaran esa información. El problema es que para uno de los sensores moverse hacia adelante podría implicar avanzar en el eje X, mientras que para el otro, podría implicar avanzar en el eje Z. Se deduce de esto que existe una rotación entre ambos sistemas coordenados que permite vincular la información de uno con el otro.

A su vez dado que los sensores no están ubicados de forma superpuesta, sino distribuidos sobre el robot, siempre existirá una diferencia debido a esta distancia entre los mismos. Este efecto se puede representar mediante una traslación entre los sistemas coordenados de cada sensor. Ejemplo de esta traslación podría ser la transformación para llevar puntos de la cámara derecha a la cámara izquierda, como lo ilustrado en la Figura 49 donde puede verse que ambos sistemas poseen el eje Z hacia el frente y el eje Y hacia abajo, pero existe una traslación en el eje X entre ambos. En el caso de las cámaras estéreos, esta distancia es fija y se la conoce como baseline.

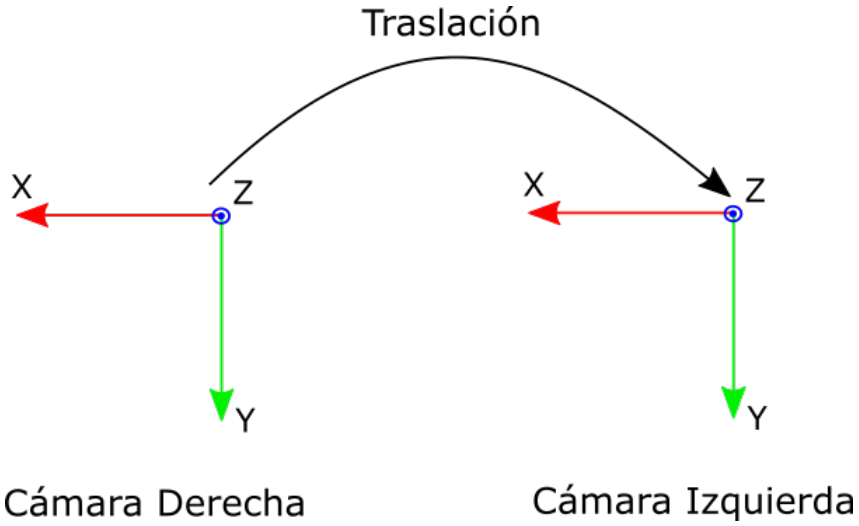


Figura 49: Traslación entre los sistemas de coordenadas de las cámaras de la Zed.

Si bien finalmente los mensajes que se incluirán en el rosbag consistirán de un vector de tres elementos de traslación y un cuaternión de cuatro elementos para la rotación, estas transformaciones no son obtenidas directamente en esta forma en todos los casos. Por el contrario, algunas se obtienen expresadas de diferente manera y fueron llevadas a la forma deseada por medio del programa mencionado en 3.8.2.

En cuanto a la transformación existente entre cada una de las cámaras y a su vez, entre ellas y la IMU se obtuvo nuevamente por medio de los algoritmos de Kalibr[7]. Donde se aplicó el mismo procedimiento de desplazar la cámara, unida estáticamente a la IMU, frente al patrón (aprilgrid en nuestro caso). El programa a partir de esa información permite obtener las transformaciones relativas entre las dos cámaras (ambas pertenecientes a la cámara estéreo Zed) y la IMU.

Para obtener la transformación entre el GPS y la IMU se realizaron mediciones sobre el robot, esto fue posible debido a que las distancias y tamaño de los sensores eran mayores al caso previo, donde realizar dichas mediciones era complicado y los errores que se cometieran serían más significativos.

[7]https://github.com/ethz-asl/kalibr/wiki/camera-imu-calibration

Para la odometría (generada a partir de las mediciones de los motores y de la dirección del robot) se decidió ubicar su origen de coordenadas en el punto central del eje trasero, ya que se corresponde con el modelo[8] que se utilizó, similar a un auto, donde se utilizan las mediciones de las ruedas traseras y la dirección ubicada en el punto central de la parte delantera. Nuevamente, la transformación entre este sistema de coordenadas y el de la IMU se obtuvo a partir de realizar mediciones sobre el robot.

Para seguir los estándares de ROS[9] [10], se incorporó un sistema de coordenadas denominado base_link que cumple el rol de ser un punto identificatorio del robot, al estar fijo a la posición en la cual se lo designe (que puede ser elegida donde sea más conveniente). En nuestro caso se ubica al base_link en el mismo lugar que la odometría, es decir, en el centro del eje trasero de las ruedas. Además, se designó al sistema de la coordenadas como frame "padre"de base_link, siendo este último frame "padre"de los otros sensores.

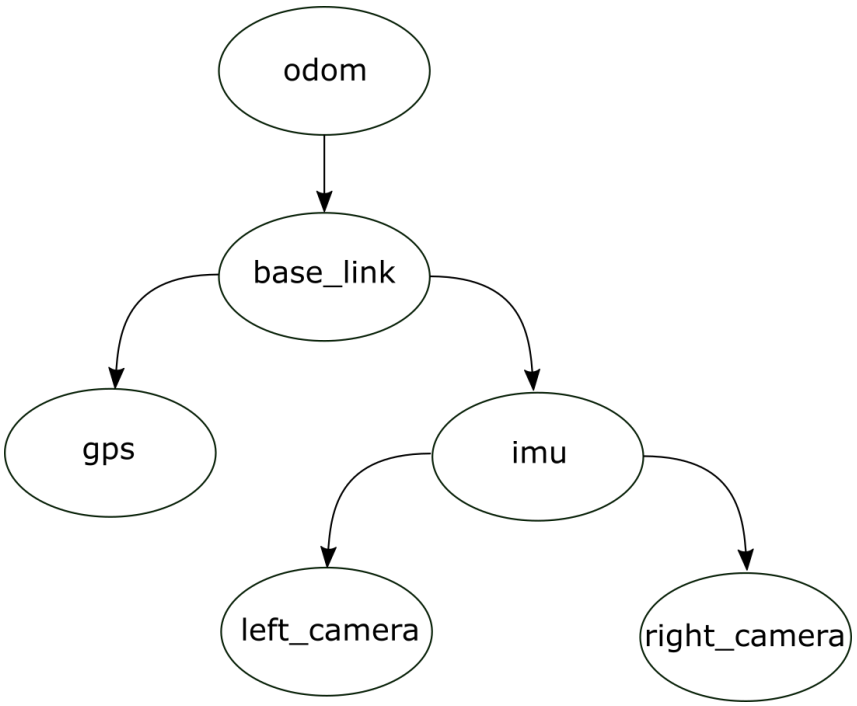


Figura 50: Árbol de transformaciones de los sistemas de coordenadas de los sensores del robot.

Un caso particular de estas relaciones es el GPS, que no proporciona sus mediciones en función de algún eje de coordenadas que pueda estar fijo al robot ya que su orientación depende de los puntos cardinales de la Tierra, diferenciándose de los otros sensores. Esta situación plantea la inexistencia de una rotación fija entre los puntos obtenidos en el sistema de referencia del GPS y los obtenidos por medio de los otros sensores.

A partir de estas mediciones y adaptando la forma de expresar las transformaciones entre un sistema de coordenadas y el otro a un vector de traslación y un cuaternión de rotación, se obtuvieron los resultados de la Tabla 2:

[8] Este modelo utilizado se detalla en la sección 3.10
[9] www.ros.org/reps/rep-0103.html
[10] www.ros.org/reps/rep-0105.html

Frame ID	Child Frame ID	x(m)	y(m)	z(m)	qx	qy	qz	qw
odom	base_link	0	0	0	0	0	0	1
base_link	gps	1.8	-0.030	1.593	-	-	-	-
base_link	imu	1.96	-0.017	1.02	-0.866	-2.33e-5	-0.499	4.012e-5
imu	left_camera	-0.031	-0.077	0.026	0.058	0.019	0.703	0.708
imu	right_camera	-0.030	0.042	0.033	0.064	0.012	0.703	0.708

Cuadro 2: Transformaciones estáticas de los sistemas de referencia de los sensores del robot expresadas en traslaciones y cuaterniones.

En resumen, el archivo .yaml posee la información obtenida de kalibr respecto a las transformaciones entre las cámaras y la IMU, junto con las calibraciones intrínsecas de la cámara. Además, la transformación entre el GPS, la odometría y la IMU con respecto al base_link. Adicionalmente, en el archivo se encuentran algunas correcciones de offset de las lecturas de la IMU.

3.9. Trayectorias obtenidas por medio del GPS

Como se detalló al inicio, uno de los objetivos del presente trabajo es obtener el posicionamiento y la trayectoria realizada por el robot, sin necesidad de la utilización del GPS. Sin embargo para poder evaluar los resultados obtenidos, por ejemplo a partir del método de SLAM visual, es necesario contar con la trayectoria real realizada por el robot. Para disponer de una referencia confiable, es que se obtuvieron las trayectorias realizadas por el robot a partir del módulo GPS-RTK incorporado. Las trayectorias obtenidas y procesadas por medio del programa desarrollado para la creación del set de datos, son las seis ilustradas en la Figura 51

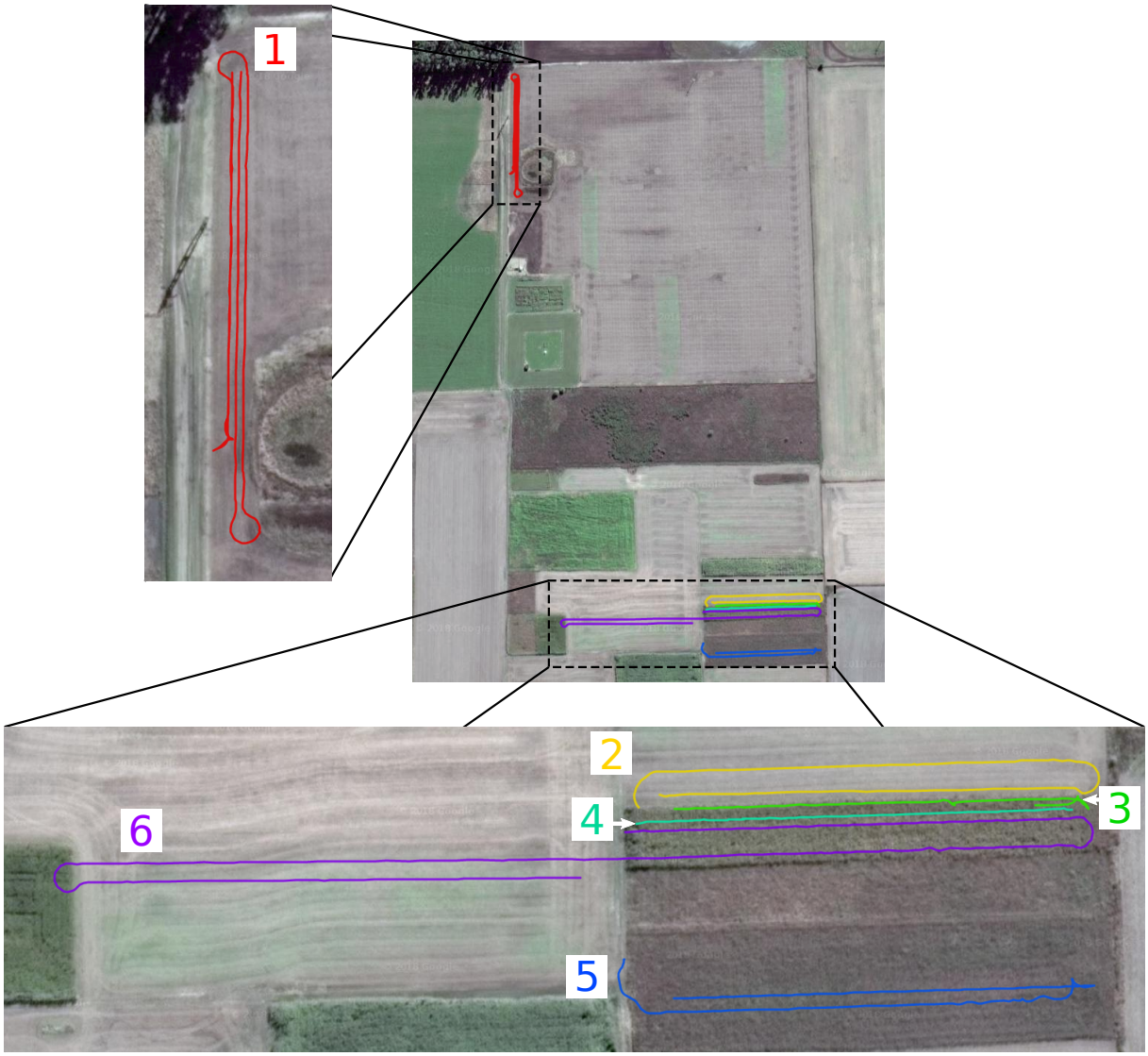


Figura 51: Trayectorias obtenidas por medio del GPS durante los días de adquisición de datos en Zavalla, Santa Fe.

Dado que la información del GPS viene expresada como latitud-longitud-altitud en forma global, la misma no es directamente comparable con la obtenida por los métodos de posicionamiento desarrollados previamente. Esto se debe a que estos últimos explicitan su posición de forma relativa a su posición inicial y la orientación del sensor.

Por esto es que fue necesario realizar la conversión de los datos obtenidos por el GPS, esto se realizó mediante la proyección UTM, que permite obtener dichas mediciones en el sistema UTM, el cual permite representar los puntos de la superficie sobre un plano. Si bien posee ventajas y desventajas, es confiable en distancias cortas y en zonas alejadas de los polos. Para mejorar el desempeño de estas transformaciones, no se realiza la conversión de toda la Tierra, sino que se divide en husos de 6º de longitud cada uno, como puede verse en la Figura 52. De esta forma las referencias sobre las que se trabaje estarán ligadas al huso correspondiente. En el caso de las cercanías de Rosario, el huso es 20 H.

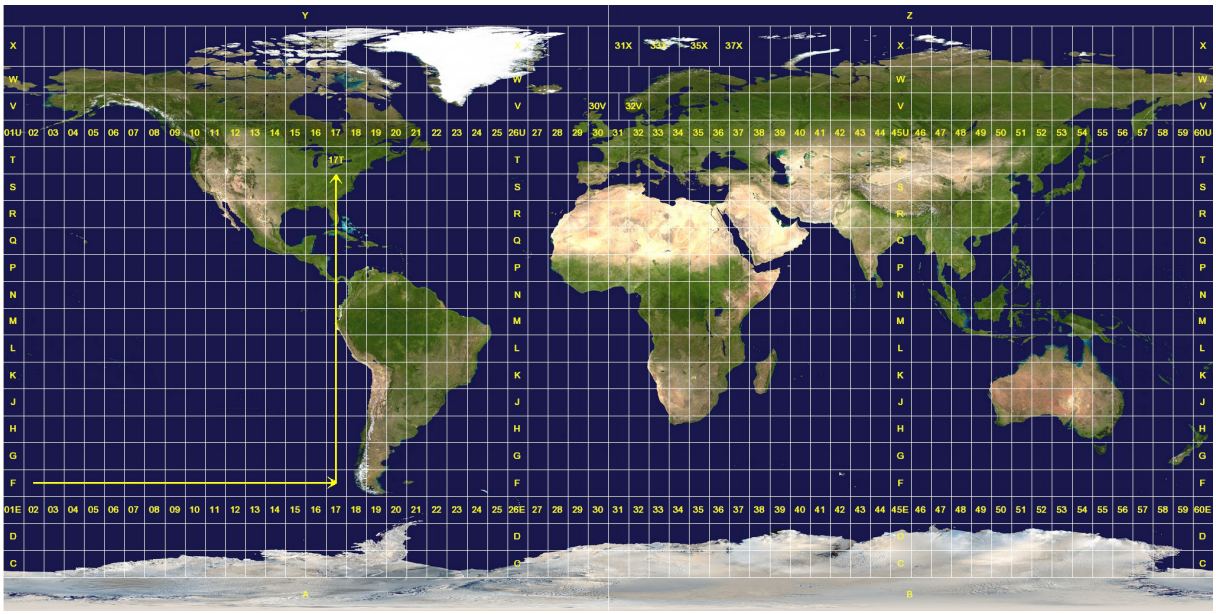


Figura 52: Proyección de la Tierra sobre el plano utilizando el sistema UTM.

A partir de este sistema, la información que se obtendrá será en dos dimensiones, dada en metros y con referencias que crecen hacia el Este y hacia el Norte. De esta forma el eje "Xçoincidirá con las referencias Este-Oeste mientras que el eje "Y"sera la representación de Norte-Sur. A partir de este sistema, se pueden obtener las posiciones y trayectorias realizados en forma relativa a la posición inicial adoptada.

Sin embargo, para que estas trayectorias sean comparables con las que se obtengan por otros métodos de posicionamiento (por ejemplo, utilizando la odometría o por medio de SLAM visual que se desarrollaran en secciones posteriores) es necesario realizar una transformación más, que permita que el robot comience desplazándose sobre el eje "Y"(en lugar de que este eje apunte hacia el Norte). Esto se debe a que es la forma en la que se obtienen los resultados de los otros sensores, donde sus referencias no están sujetas a las referencias de la Tierra, sino a su posición inicial y un eje de referencia hacia el frente.

Para poder obtener la trayectoria comenzando en el eje "y", pero manteniendo la forma y relaciones de aspecto lo que se hace es rotar la trayectoria para que su comienzo coincida con el sentido positivo del eje "Y". Es decir se aplica la transformación siguiente:

$$x_{rot} = (x - y\ \tan(\theta))\ \cos(\theta) \tag{9}$$

$$y_{rot} = \frac{y}{\cos(\theta)} + [(x - y\ \tan(\theta)]\ \sin(\theta) \tag{10}$$

Donde θ es el ángulo entre la dirección en la que comienza el robot y el Norte de la Tierra.

De esta forma las trayectorias son rotadas para que comiencen alineadas con el eje "Yçomo puede verse en la Figura 53 donde en azul esta una parte de una trayectoria del GPS en su formato original y en rojo, la misma una vez aplicada la rotación.

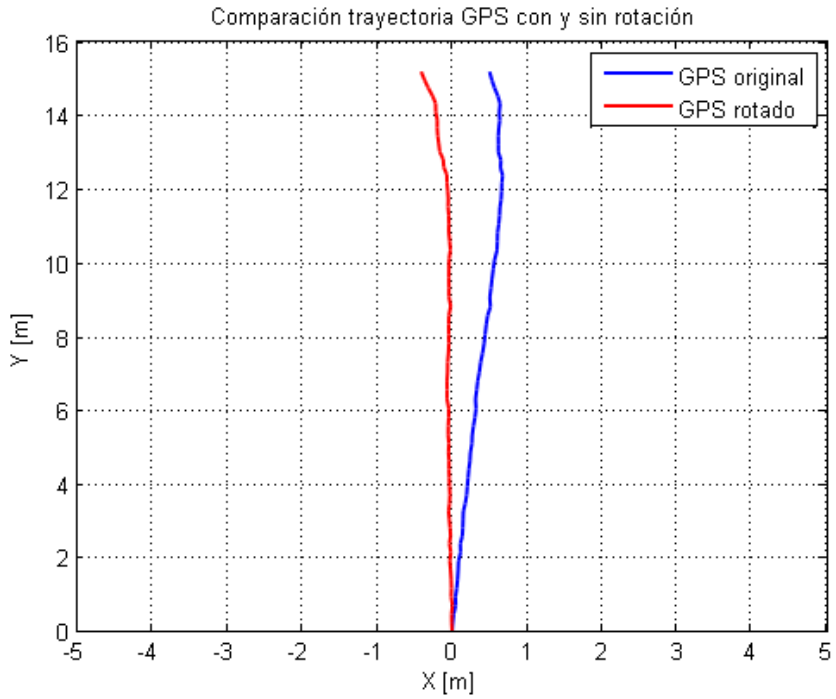


Figura 53: Ejemplo de aplicación de la rotación de la trayectoria del GPS para alinear su comienzo con el sentido positivo del eje "Y".

3.10. Localización por medio de la odometría obtenida a partir de los motores

Para obtener la posición y la velocidad del robot, en función de las mediciones de los motores, tanto de las ruedas como de la dirección, del robot se utilizo un modelo cinemático[22] derivado del modelo de Ackerman. Para este modelo se tuvieron en cuenta la velocidad lineal del vehiculo, obtenida como el promedio de los dos motores traseros y en ángulo girado por el motor paso a paso que controla la dirección del robot.

La información de los motores, que fue obtenido por medio del software que realizaba la comunicación con el microcontrolador, estaba expresada en RPM, por lo que fue convertida a metros por segundo de la forma siguiente:

$$v = (v_m \, \pi \, d)/60 \tag{11}$$

Donde v es la velocidad en m/s, v_m es la velocidad en RPM y d es el diámetro de las ruedas que equivale a 0,57 metros.

Por otro lado, la dirección del robot fue obtenida como un valor entre [-100,100] representando un giro máximo a la izquierda y un giro máximo a la derecha respectivamente. Se calculó que dicho ángulo máximo se correspondía con 19º por lo tanto cada unidad en la que venía expresada la dirección equivale a 0,19º.

Por último, también fue necesaria la información la distancia entre el eje trasero y el eje delantero, la cual es de 1,6 metros.

3.10.1. Modelo cinemático

El modelo utilizado [22] es útil para bajas velocidades y define la posición a partir de (x,y) y del angulo θ de orientación. A su vez estas tres son las variables de estado utilizadas por el modelo de la forma siguiente:

$$\dot{x} = v\,\cos\theta \tag{12}$$

$$\dot{y} = v\,\sin\theta \tag{13}$$

$$\dot{\theta} = \frac{v}{L}\,\tan(-\,\delta) \tag{14}$$

El sistema permite obtener la posición del centro del eje trasero en función de la velocidad lineal del robot, el ángulo de dirección (que no es el mismo que el ángulo de la orientación del robot en función al sistema de coordenadas) y la distancia entre el eje delantero y trasero del robot. Esto puede apreciarse en la Figura 54.

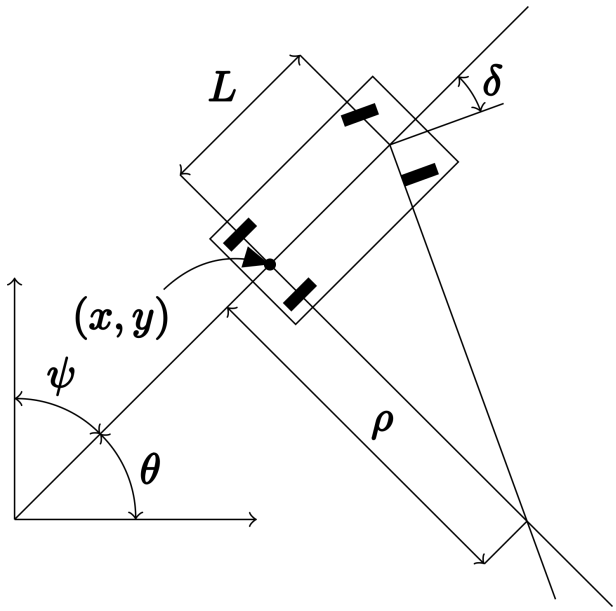


Figura 54: Modelo utilizado para el robot similar al de un auto, con tracción en las ruedas traseras y dirección delantera.

Al realizar las simulaciones se pudo detectar que existía un error debido a una clara tendencia del robot hacia la izquierda. Esto puede atribuirse a una desalineación de la dirección del robot, provocando que este gire levemente a la derecha sin ser detectado por el encoder el motor paso a paso, ya que le mismo se mantuvo sin rotar. El problema radica en que cuando esto ocurría, la dirección era corregida mediante el control remoto (girando levemente a la izquierda), modificación que sí aparecía en el encoder del motor paso a paso. En resumen sólo aparecen en las mediciones las correcciones, pero no la causa de las mismas, por lo que el robot supone que esta desplazándose a la izquierda cuando en realidad se mantenía derecho.

Para intentar mejorar el desempeño del modelo a pesar de este inconveniente, se tomaron 8 tramos rectos obtenidos del set de datos y se estimo la media de la dirección, obteniéndose de esta forma un "offset"de -5.67 en al medición que equivale a 1.07º a la izquierda, lo cual fue

corregido en una nueva versión de los datos para obtener resultados que sean más acordes al experimento realizado.

3.10.2. Simulaciones realizadas

A partir del modelo implementado y con el ajuste del offset se procedió a simular el funcionamiento del mismo en algunas de las secuencias del set de datos con el objetivo de evaluar su desempeño y posible limitaciones.

A partir de la simulación en la secuencia número uno se obtuvieron los resultados de la Figura 55 donde puede verse que si bien las trayectorias poseen similitudes hay un error considerable en comparación al GPS.

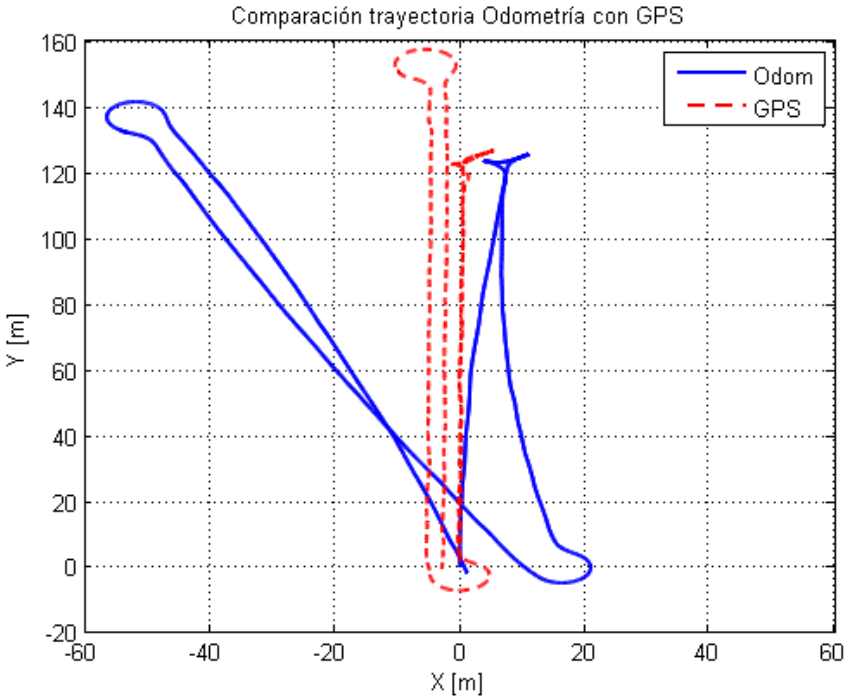


Figura 55: Comparación de las trayectorias realizadas por el modelo generado por la odometría del robot y el GPS sobre la secuencia nº 1 del set de datos.

Este error podría atribuirse a que la utilización de la odometría de las ruedas para el calculo de la posición en forma directa es muy sensible a ruidos o eventos externos que pudieran afectar el funcionamiento. Ejemplo de esto es la inclinación del terreno que podría inducir al robot a desplazarse hacia uno de sus lados en un sentido y luego al otro al regresar. Esta situación plantea la misma dificultad que el error en la alineación, el robot se desvía levemente hacia alguno de los costados pero esto no es percibido por el encoder, pero si lo son las correcciones realizadas por medio del control remoto (efecto que puede apreciarse en la Figura 56, la cual muestra la trayectoria realizada a partir de la secuencia dos).

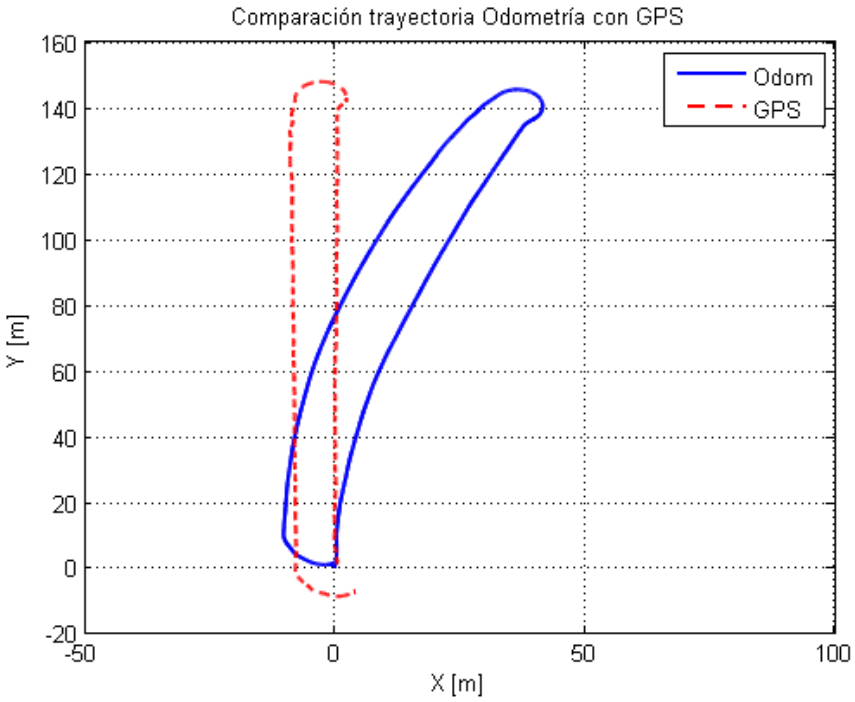


Figura 56: Comparación de las trayectorias realizadas por el modelo generado por la odometría del robot y el GPS sobre la secuencia nº 2 del set de datos.

Por otro lado si se simula la secuencia número cuatro del set de datos puede apreciarse, en la Figura 57, como la odometría es una buena referencia de distancias recorridas (siempre y cuando el robot no deslice o patinen sus ruedas).

A partir de estos resultados es evidente que la utilización de la odometría por si misma no es viable, pero si que puede ser una herramienta para proporcionar información sobre distancias recorridas o, por ejemplo, validación de datos ya que al ser calculada por medio de integrar velocidades la misma nunca proporcionará puntos que "salten"de una posición a otra muy alejada en cuestión de milésimas de segundos (esto sí podría ocurrir con el GPS si tuviera una medición errónea)

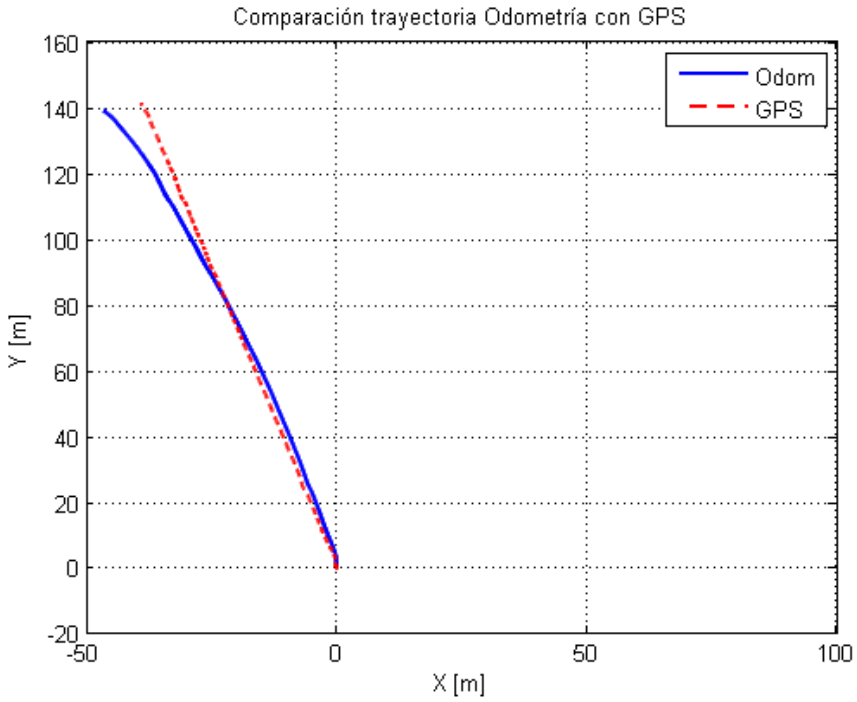


Figura 57: Comparación de las trayectorias realizadas por el modelo generado por la odometría del robot y el GPS sobre la secuencia nº 4 del set de datos.

3.11. Localización por medio de SLAM visual

Si bien la cámara estéreo Zed tendrá la función indispensable de ser el instrumento utilizado para la detección de malezas, puede también ser muy útil para la localización del robot y obtener la trayectoria realizada por el mismo. Esto puede realizarse empleando un método de SLAM visual, donde por medio de las imágenes se obtiene la posición del robot frame a frame con respecto a los objetos del entorno detectados en las mismas.

Existen muchos algoritmos con diferentes enfoques para realizar esta tarea, en este caso se utilizó S-PTAM[11](de sus siglas en inglés, Stereo Parallel Tracking and Mapping), un método de SLAM que permite obtener gran cantidad de información sobre la posición y desplazamiento de la cámara, así como también del entorno (a partir de los features o puntos que utiliza para estimar su posición).

Como se había mencionado previamente, este método funciona basado en ROS, por lo que su utilización y los parámetros de entrada y salida siguen los estándares planteados por este framework. De esta forma, haber adaptado nuestro set de datos y el tipo de mensajes a utilizar, para poder trabajar con nodos (programas, como S-PTAM) de ROS, era indispensable.

[11]https://github.com/lrse/sptam

3.11.1. S-PTAM

Este método de localización y mapeo visual, fue desarrollado para poder trabajar en tiempo real y presentar buenos desempeños en espacios abiertos permitiendo estimar, de forma precisa, la localización del robot. Una de las particularidades de este método es que desacopla las tareas de localización y mapeo, permitiendo un mejor aprovechamiento del poder de computo en procesadores con múltiples núcleos.

Como se mencionó previamente, este programa es trabaja sobre ROS, utilizando este medio para recibir la información de entrada y publicar los resultados. Además, posee ciertos parámetros de funcionamiento que pueden ser modificados en forma previa a la inicialización del mismo, es decir, antes de comenzar el proceso de localización.

Figura 58: Esquema del nodo de S-PTAM con la información que necesita de entrada y la de salida.

Para el funcionamiento del método, tal como se muestra en la Figura 58, este debe disponer de cuatro entradas, las cuales serán mensajes publicados en canales específicos ya determinados. Estas entradas, que en la figura mencionada aparecen con nombres únicamente descriptivos son:

- Imagen izquierda: Imágenes de la cámara izquierda ya rectificadas y no distorsionadas. Las mismas deben ser mensajes de tipo *sensor_msgs/Image* publicados en el tópico "/stereo/left/image_rect"
- Info cámara izquierda: Datos de calibración de la cámara izquierda. Los mensajes deben ser del tipo *snsor_msgs/CameraInfo* y publicados en el tópico "/stereo/left/camera_info"
- Imagen derecha: Imágenes de la cámara derecha ya rectificadas y no distorsionadas. Las mismas deben ser mensajes de tipo *sensor_msgs/Image* publicados en el tópico "/stereo/right/image_rect"
- Info cámara derecha: Datos de calibración de la cámara derecha. Los mensajes deben ser del tipo *snsor_msgs/CameraInfo* y publicados en el tópico "/stereo/right/camera_info"

Por otro lado, las salidas del programa también son mensajes publicados en tópicos definidos y que deberán ser escuchados por algún otro nodo para poder hacer uso de ellos. Estas salidas o resultados del sistema S-PTAM son:

- Mapa global: Nube de todos los puntos utilizados para el seguimiento durante la trayectoria. Los mismos deben ser mensajes de tipo *sensor_msgs/PointCloud2* publicados en el tópico "/sptam/global_map"

- Mapa local: Nube de los puntos que son altamente probables de encontrar en el frame actual. Los mismos deben ser mensajes de tipo *sensor_msgs/PointCloud2* publicados en el tópico "/sptam/local_map"
- Mapa localizado: Nube de puntos detectados por la cámara. Los mismos deben ser mensajes de tipo *sensor_msgs/PointCloud2* publicados en el tópico "/sptam/tracked_map"
- Frames clave: Todos los frames claves detectados durante la trayectoria. Los mismos deben ser mensajes de tipo *sensor_msgs/Path* publicados en el tópico "/sptam/keyframes"
- Frames claves locales: Frames claves utilizados para generar el mapa actual. Los mismos deben ser mensajes de tipo *sensor_msgs/Path* publicados en el tópico "/sptam/local_keyframes"
- Posición cámara: Posición actual de la cámara después de la localización. Las mismas deben ser mensajes de tipo *sensor_msgs/PoseWithCovarianceStamped* publicados en el tópico "/sptam/camera_pose"

Si bien no se trabajó ni modificó el funcionamiento interno del método, sí fue posible variar algunos de los parámetros que utiliza el nodo con el objetivo de mejorar el desempeño del mismo en función del entorno de trabajo. Los parámetros que pueden modificarse al inicializar el nodo y que fueron tratados para optimizar los resultados son:

- MatchingDistance: Distancia en términos de probabilidad de coincidencia al realizar el emparejamiento (matching, en inglés) de los puntos. De forma predeterminada, el valor es 25. Si se aumenta este valor, se obtendrán más puntos (o matches) pero con el riesgo de realizar la coincidencia entre dos punto,s en frames distintos, que no corresponden al mismo punto en el punto real.
- EpipolarDistance: Distancia en lineas de la linea epipolar donde intentar encontrar el mismo punto en las proyecciones de ambas cámaras. El valor predeterminado es cero, es decir sólo busca en la fila de la linea epipolar. Al aumentar este valor se acepta que el punto, en una de las cámaras, no se encuentre exactamente en la fila correspondiente a la linea epipolar (como debería ocurrir en forma teórica si la calibración y rectificación fueran perfectas). Aumentar excesivamente este valor podría generar coincidencias incorrectas y aumentar el costo computacional.
- FrustumNearPlaneDist: Límite inferior del campo de visión, es decir a partir de que distancia de la cámara,se buscará detectar puntos. El valor por defecto es 0,1.
- FrustumFarPlaneDist: Límite superior del campo de visión, es decir hasta que distancia de la cámara, se buscará detectar puntos. El valor por defecto es 1000. Este parámetro es de interés ya que en función del mismo se modificará la detección de puntos en el horizonte. Estos son los puntos que son siempre visibles y útiles para la orientación del sistema.

El procedimiento utilizado para poner en funcionamiento S-PTAM consiste en, desde la terminal de Ubuntu, inicializar el nodo. Esto se realiza por medio de un archivo *.launch* que permite:

1. Utilizar previamente un nodo que rectifica y evita la distorsión de las imágenes previo a ser utilizadas por S-PTAM (como se mencionó previamente, el método necesita las imágenes de esta manera). El nodo utilizado es "stereo_image_proc"[12].

2. Da valores a algunos de los parámetros necesarios para que S-PTAM funcione correctamente en ROS y mapea de forma correcta los nombres de los tópicos de salida del nodo de rectificación para que coincidan con los esperados por los tópicos de entrada de S-PTAM.

3. Inicializar S-PTAM.

3.11.2. Ajuste de parámetros por medio de simulaciones en tiempo real sobre el set de datos obtenido

Para simular el funcionamiento del método en condiciones reales de trabajo (es decir, en el entorno en el que trabajará el robot) pero además, simulando el funcionamiento en tiempo real, es que se utilizó la cualidad de ROS de trabajar en tiempo simulado. Puntualmente, esto permite que el reloj del sistema se coloque en el momento indicado por el primer timestamp de un rosbag y, a partir de allí avancen los mensajes guardados, en concordancia con el timestamp de cada uno. Es decir, si entre dos mensajes consecutivos había un segundo de diferencia entre sus timestamps, S-PTAM los recibirña con un segundo de diferencia. Es de esta manera que, si los timestamps se corresponden con los momentos en los que fueron obtenidas las mediciones, este comportamiento sera equivalente a proveerle los mensajes con las mediciones en el momento que son obtenidas.

Evaluar este comportamiento es importante por dos motivos, en primer lugar, permite evidenciar el funcionamiento del método y de la localización obtenida mientras el rosbag avanza (avanza el tiempo ,y cuando se dan los tiempos de los timestamps, los mensajes del rosbag son enviados), observando la trayectoria descripta por el robot. En segundo lugar, puede comprobarse si la computadora sobre la que se están ejecutando los programas es capaz de realizar el procesamiento necesario para estimar la posición en el tiempo disponible entre la llegada de dos *frames*. Esto es importante ya que si no ocurre de esta manera se "perderían" *frames*, llevando a errores en el posicionamiento y a momentos donde se desconoce la localización del robot.

Simulando sobre la secuencia número dos del set de datos, se analizó el funcionamiento para realizar el acondicionamiento de los parámetros disponibles para modificar de S-PTAM y de esta forma, buscar optimizar su funcionamiento en cuanto a disminuir el error con respecto a la trayectoria real (obtenida por el GPS) y en cuanto a la robustez del método. En este aspecto, es clave que la computadora utilizada sea capaz de realizar el procesamiento, evitando errores, perdida de *frames* o fallas en el programa.

En primer lugar se dispone de la configuración por defecto para la Zed:

- MatchingDistance: 25
- EpipolarDistance: 0
- FrustumNearPlaneDist: 0.1
- FrustumFarPlaneDist: 1000

[12] http://wiki.ros.org/stereo_image_proc

A partir de esto, se disminuyó *MatchingDistance* a 20, con el objetivo de volver más selectiva la comparación de los puntos *frame* a *frame*, esto es coherente con el trabajo en un entorno monótono, donde la similitud entre puntos distintos es alta. A partir de esto se obtuvieron los resultados de la Figura 59 donde puede verse que la gráfica se aproxima más a la real, pero no mantiene el paralelismo entre los tramos rectos, indicando posibles problemas de orientación, puntualmente al entrar o salir de las curvas. Esto podría deberse también a que al girar, los surcos dejan de ser visibles, perdiéndose muchos de los puntos que habían sido encontrados en los bordes entre la tierra y la soja.

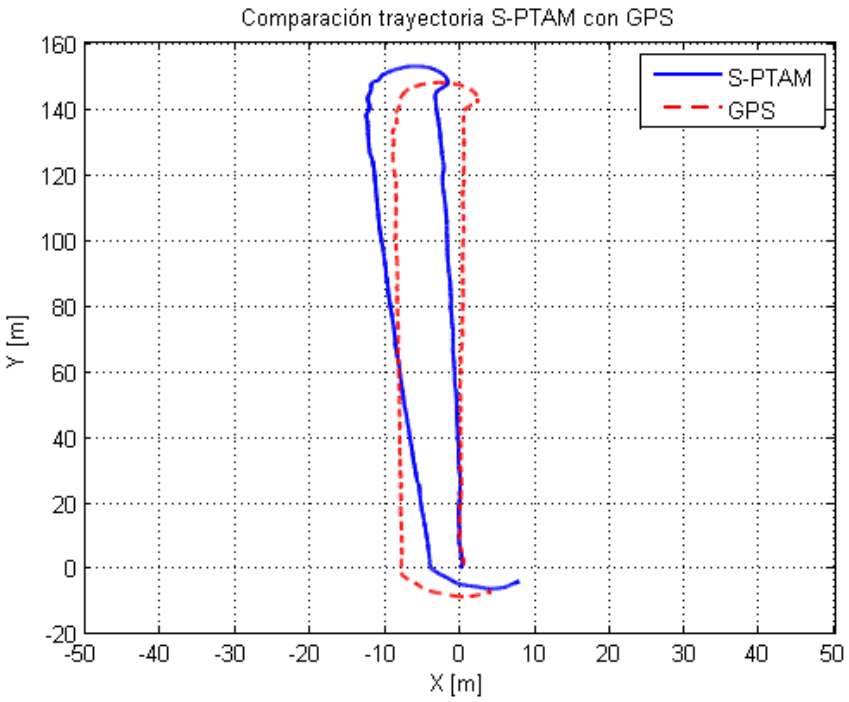


Figura 59: Simulación de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 2 del dataset.

Una posibles solución planteada para intentar mejorar el desempeño fue aumentar la distancia máxima a la que intenta encontrar *features*. El objetivo de esta modificación es detectar puntos en el horizonte o en árboles y estructuras lejanas, los cuales mantendrán su posición y serán visibles durante todo el recorrido. En función de esto se aumentó *FrustumFarPlaneDist* a 10000. Puede verse en la Figura 60 como mejora el desempeño del método, siendo mas fiel a la trayectoria real. Los caminos de ida y vuelta mantienen de mejor forma el paralelismo que existía, debido a la forma en la que están trazados los surcos. Sin embargo persisten dos problemas, en primer lugar que el giro se realizó aproximadamente 5 metros mas lejos de lo que debía. Esto sugiere que se acumuló un error en la trayectoria recta de aproximadamente 3,3 % sobre la real. Por otro lado puede verse que al volver a entrar al surco, luego de realizar el giro de 180º, no entra perfectamente recto, como se supone que debía hacer, sino que con cierto ángulo que es lo que genera que, al regresar, las trayectorias se separen.

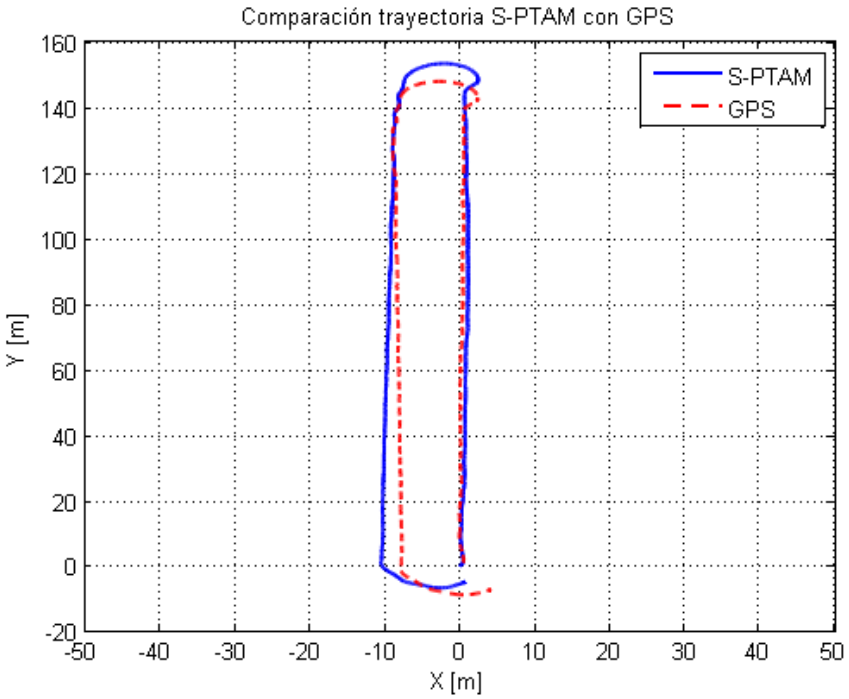


Figura 60: Simulación de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 2 del datase aumentando FrustumFarPlaneDist a 10000.

Este último punto es interesante ya que el método considera que continua yendo alineado y paralelo a los surcos pero, al realizar el giro, la alineación entre el camino de ida y el de vuelta no es la correcta. Es decir, en su mapa los surcos observados en el camino de ida y los observados en el camino de vuelta no son paralelos, cuando sí lo eran realmente. Este efecto puede observarse en la Figura 61 donde en verde se ve una parte de la trayectoria en un sentido y en el otro y, en torno a las mismas, los puntos o *features* detectados. En esta nube de puntos puede apreciarse la forma de los surcos y, a partir de los mismos, el error en la creación de su propio mapa debido a que el método consideró que la salida y entrada a los surcos, luego del giro, no fueron paralelos.

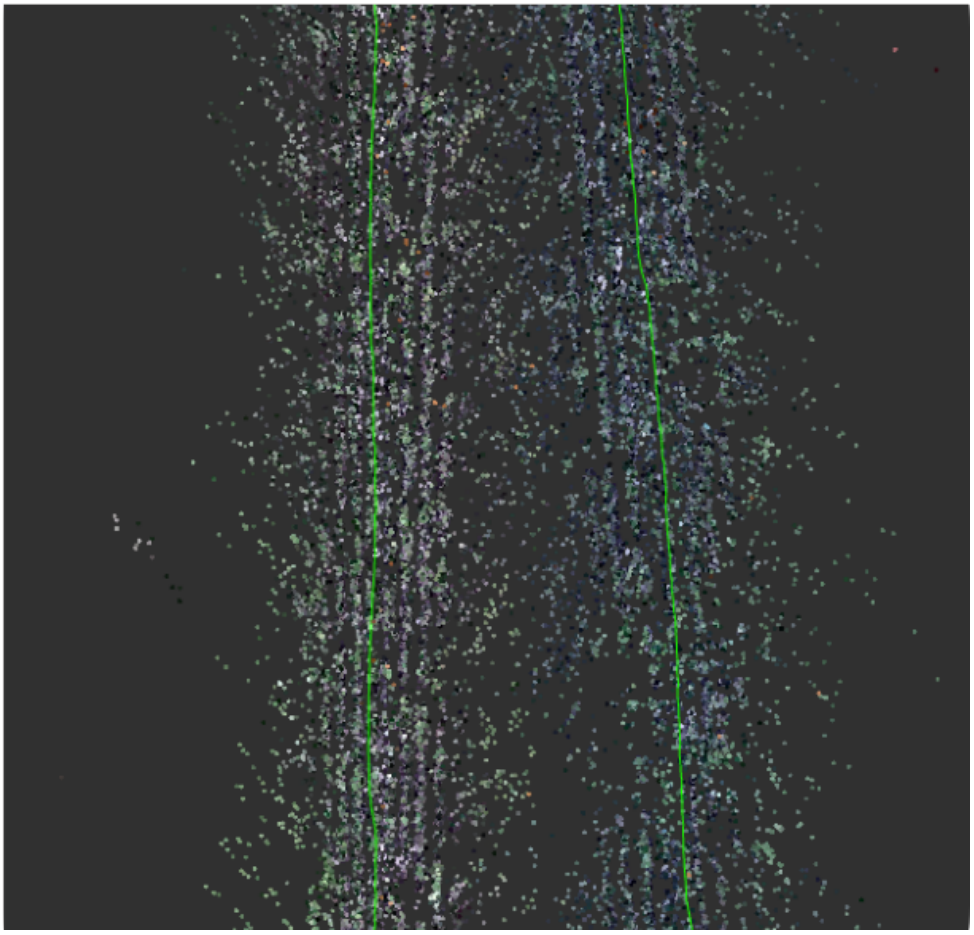


Figura 61: Parte de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 2 del dataset con parte del recorrido en ambas direcciones y la nube de puntos obtenida.

Otra posibilidad es variar *EpipolarDistance*, aumentándolo por ejemplo a uno, permitiendo de esta forma que se intente encontrar el mismo punto en ambas cámaras con una fila de diferencia en lugar de que sea exclusivamente en la linea epipolar. Sin embargo como puede verse en la Figura 62 esto conduce a errores en la detección de puntos debido a lo monótono del ambiente. Por lo tanto este valor se conservó en cero.

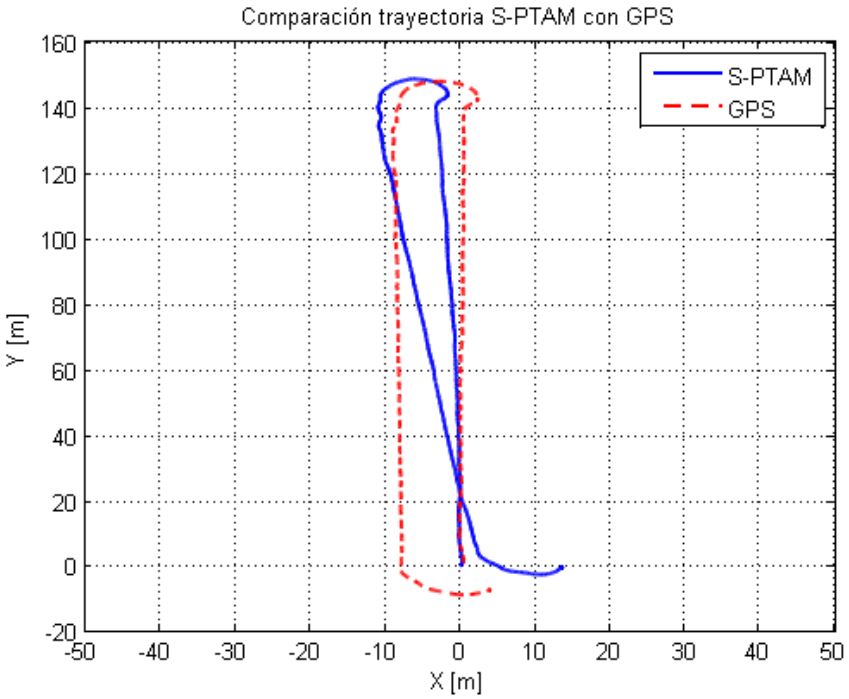


Figura 62: Simulación de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 2 del dataset aumentando EpipolarDistance a uno.

Finalmente, se disminuyó aun más el valor de *MatchingDistance*, con el objetivo de volver el reconocimiento de los puntos aun más selectivo, sin embargo si bien la trayectoria obtenida en la Figura 63 mejora en algunos aspectos, presenta cierto errores. Particularmente, en la región marcada, donde se realiza un estudio en detalle, puede verse que el algoritmo falló en la detección de los puntos generando una indeterminación en la posición. Esto podría deberse a que al exigirle demasiada selectividad para el emparejamiento de los puntos entre *frame* y *frame*, el método es incapaz de detectar suficientes puntos para obtener su posición, provocando la indeterminación.

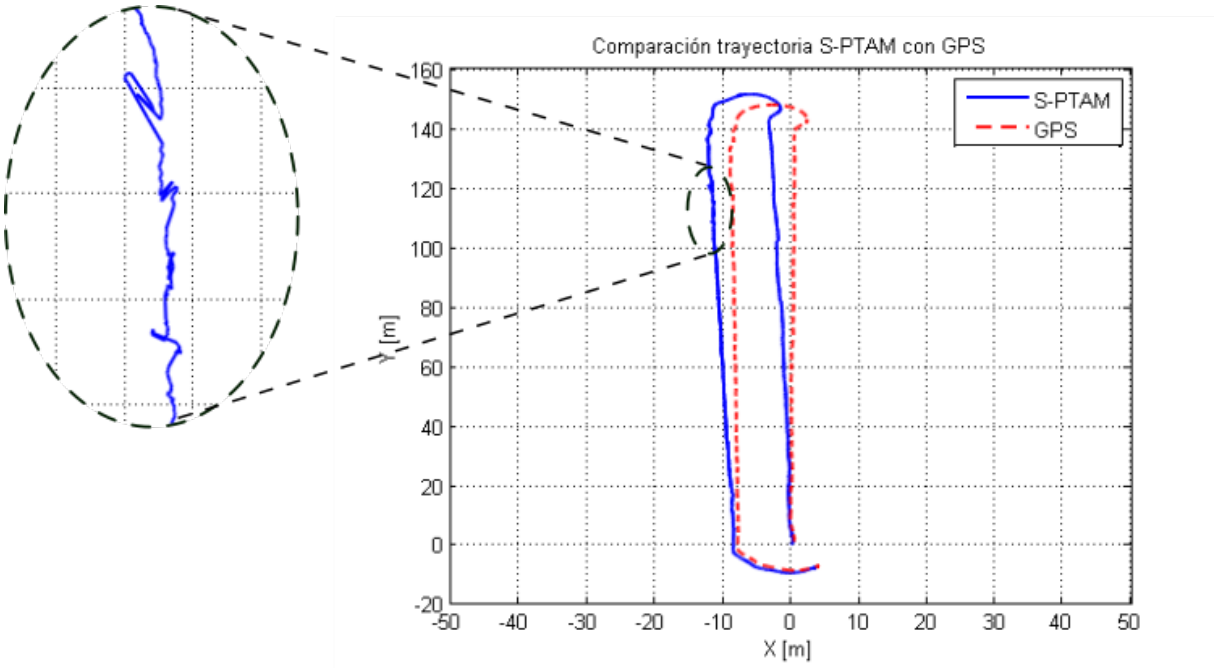


Figura 63: Simulación de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 2 del dataset diminuyendo excesivamente MatchingDistance.

A partir de estos resultados y de repetir las simulaciones, se configuró el archivo de parámetros de la siguiente forma:

- MatchingDistance: 20
- EpipolarDistance: 0
- FrustumNearPlaneDist: 0.1
- FrustumFarPlaneDist: 10000

3.12. Simulaciones Finales y evaluación de resultados

3.12.1. Simulaciones de SPTAM

En base a la configuración obtenida de los parámetros del nodo de S-PTAM, se realizaron simulaciones sobre las secuencias obtenidas del set de datos y se compararon los resultados con las trayectorias del GPS en la misma secuencia. Estas simulaciones fueron realizadas en tiempo real en una computadora Intel® i7 CPU 970 de 3.20 GHz con 12 núcleos y 8Gb de memoria RAM.

El orden en el que se presentan las simulaciones realizadas está dado por la dificultad que representa cada una de las secuencias para el método donde, duraciones mayores, giros, segmentos de marcha atrás, y, por sobre todo, la poca visibilidad del surco debido al tamaño de la soja, aumentan la dificultad de la implementación y el buen desempeño del método. En este último aspecto puede verse en la Figura 64 que las primeras cuatro secuencias poseen buena

visibilidad de los surcos, por el contrario, las últimas dos poseen una mayor altura de la soja que dificulta la visibilidad. Esto las convierte en las secuencias mas complicadas para el localización y mapeo, al menos, en cuanto a la dificultad impuesta por el entorno.

(a) secuencia nº 1

(b) secuencia nº 2

(c) secuencia nº 3

(d) secuencia nº 4

(e) secuencia nº 5

(f) secuencia nº 6

Figura 64: Imágenes representativas de la visibilidad de los surcos en cada una de las secuencias del set de datos.

En complemento a la secuencia número dos, que ya fue desarrollada para realizar la parametrización, se simuló el método sobre las secuencias número tres y cuatro. Estas corresponden a lo que originalmente fue una trayectoria única, pero que se la dividió en dos debido a que personas interfirieron con la visibilidad de la cámara. A partir de estas simulaciones se obtuvieron las Figuras 65 y 66

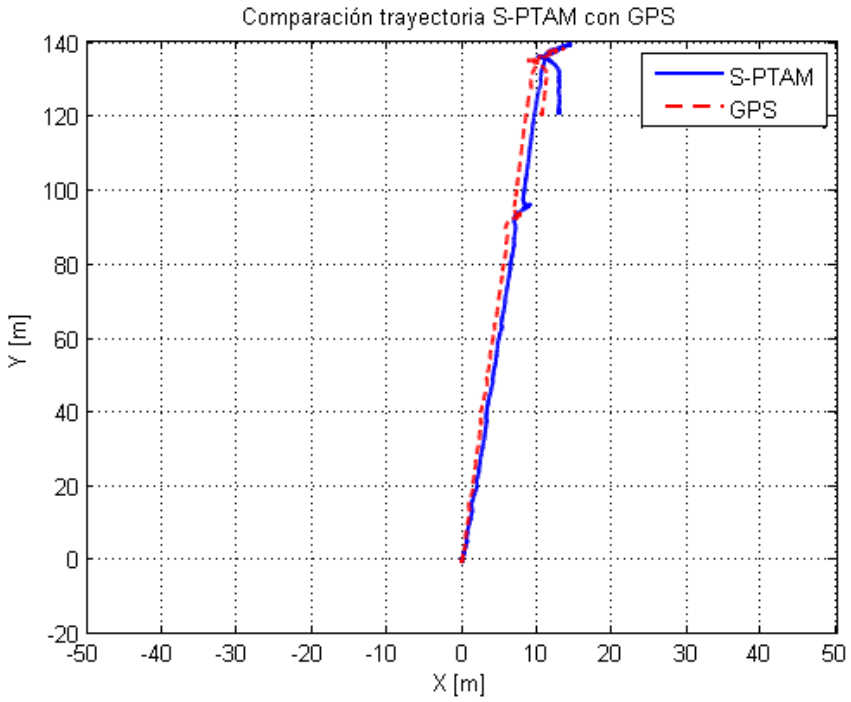


Figura 65: Simulación de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 3 del dataset.

En la simulación de la secuencia 3 puede verse como la trayectoria descripta por S-PTAM se asemeja en gran medida a la obtenida por medio el GPS, desviándose levemente (uno o dos metros, en la peor situación). En la parte final de la secuencia el robot realizó un movimiento en marcha atrás y el posterior giro para colocarse nuevamente de frente. Puede verse como la simulación, al igual que el GPS, denotan este comportamiento. Puede destacarse, que al igual que en las simulaciones ya realizadas, luego de estos giros se presentan algunos inconvenientes en la orientación, esto puede verse como al finalizar el proceso de marcha atrás y posterior giro, la trayectoria regresa con un ángulo un poco mayor al del GPS, produciendo, si la secuencia continuara, que probablemente no regresara completamente paralelo al camino de ida.

En cuanto a la simulación de la trayectoria obtenida a partir de la secuencia número cuatro, puede verse que la misma consiste en una trayectoria recta sin grandes giros, a excepción de leves desviaciones y correcciones producto de la manipulación del robot por medio del control remoto. En esta simulación puede apreciarse la gran coincidencia de S-PTAM con el funcionamiento real del robot, realizando esta sección recta de cerca de 145 metros, donde se acumuló un error que no alcanza los 2 metros, producto de algún pequeño error del método durante la secuencia.

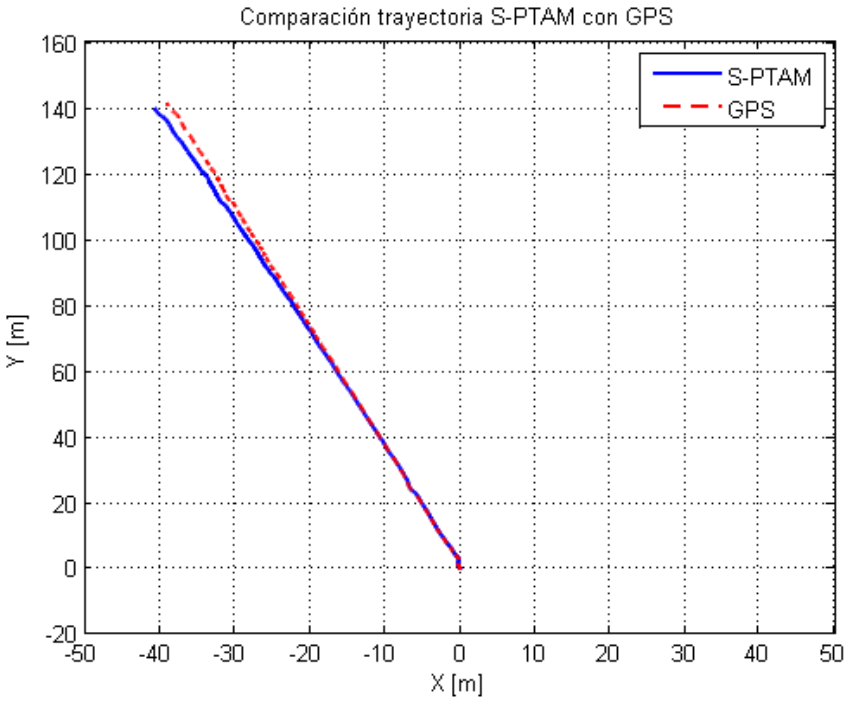


Figura 66: Simulación de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 4 del dataset.

Posteriormente, se simuló el funcionamiento sobre la secuencia número uno que, si bien no posee mayores complicaciones que las anteriores en cuanto al entorno y visibilidad de los surcos, sí posee un grado mayor de dificultad debido a la duración de la trayectoria y a que posee varios giros y cambios de dirección. A partir de la información de esta secuencia del set de datos se simuló el funcionamiento de S-PTAM, obteniéndose la trayectoria de la Figura 67. En la gráfica puede verse como las trayectorias difieren ampliamente en forma global, esto se debe principalmente a problemas en orientación o reingreso posterior a un giro, donde nuevamente las trayectorias no mantienen el paralelismo entre si.

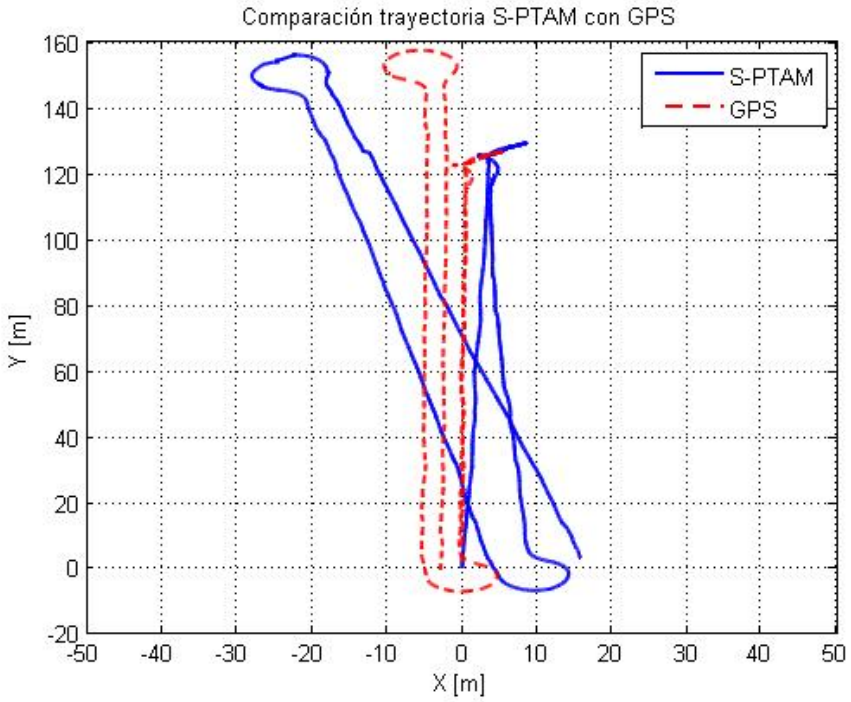


Figura 67: Simulación de la trayectoria obtenida por S-PTAM a partir de la secuencia n°1 del dataset.

En lo que respecta a las secuencias cinco y seis (que como se mencionó, son las que presentan las mayores dificultades) no pudieron ser simuladas en forma completa debido a que su complejidad dificultó ampliamente la detección de los puntos. Esta situación llevó a que el método no pudiera seguir la trayectoria realizada, generando que se interrumpa el seguimiento de los puntos y la creación del mapa.

A partir de estos resultados, de la utilización de S-PTAM para la localización y mapeo en el entorno agrícola, pudo verse que, con serias consideraciones sobre el entorno, la capacidad de procesamiento y los posibles inconvenientes de orientación, la utilización de S-PTAM es una posibilidad para la obtención de gran cantidad de información sobre la posición del robot. Además brinda información relativa del robot en función del entorno, lo cual no sería posible con un GPS. Pero como por le momento la computadora utilizada en el robot es la NUC, la cual fue implementada para la obtención de datos, se buscó implementar el método y replicar las simulaciones en ella.

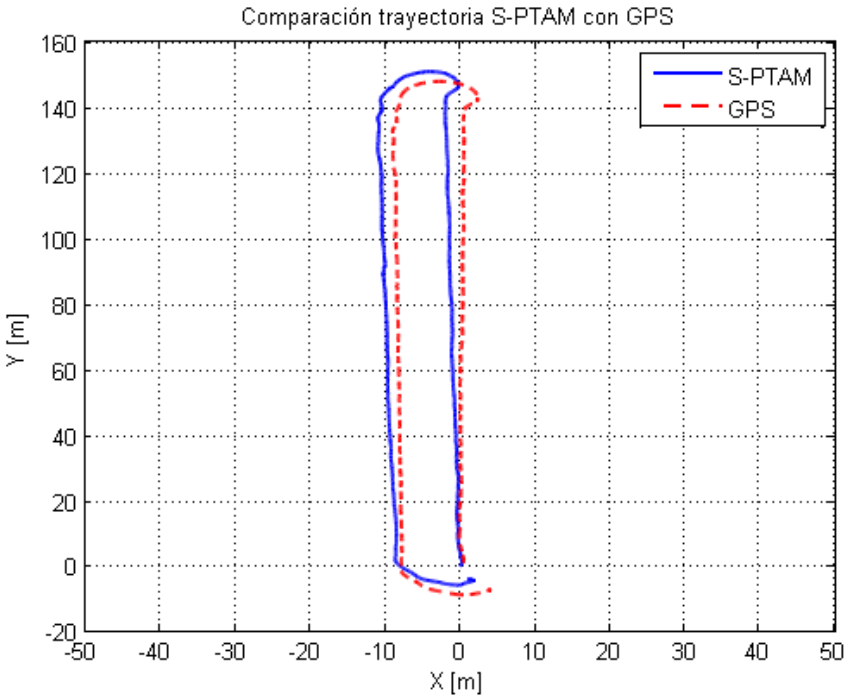


Figura 68: Simulación realizada sobre la NUC, de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 2 del dataset.

Se realizó, en primera instancia, la simulación sobre la secuencia número dos, en tiempo real y manteniendo la configuración que se había implementado en la simulaciones previas. De esta forma se obtuvo la Figura 68 donde puede apreciarse que el comportamiento continúa siendo similar al real (o al determinado por el GPS). Por otro lado, la utilización del método se vuelve menos robusta ya que no siempre se obtienen estos resultados. En algunas ocasiones S-PTAM no es capaz de realizar el procesamiento entre *frame* y *frame*, por lo que no es capaz de mantener el seguimiento de la trayectoria realizada. Evidentemente esto es consecuencia de la diferencia en las capacidades de procesamiento entre las dos computadoras y la cantidad de núcleos de los que dispone se dispone para trabajar.

También se simuló en la NUC la secuencia número cuatro en tiempo real, obteniéndose como resultado la Figura 69. Los resultados son similares a los obtenidos en la Figura 66, incluso parecen mejores que los simulados previamente. Esto podría deberse a que el S-PTAM no es completamente determinístico, sino que utiliza métodos probabilísticos, generando resultados que pueden variar levemente entre una simulación y otra.

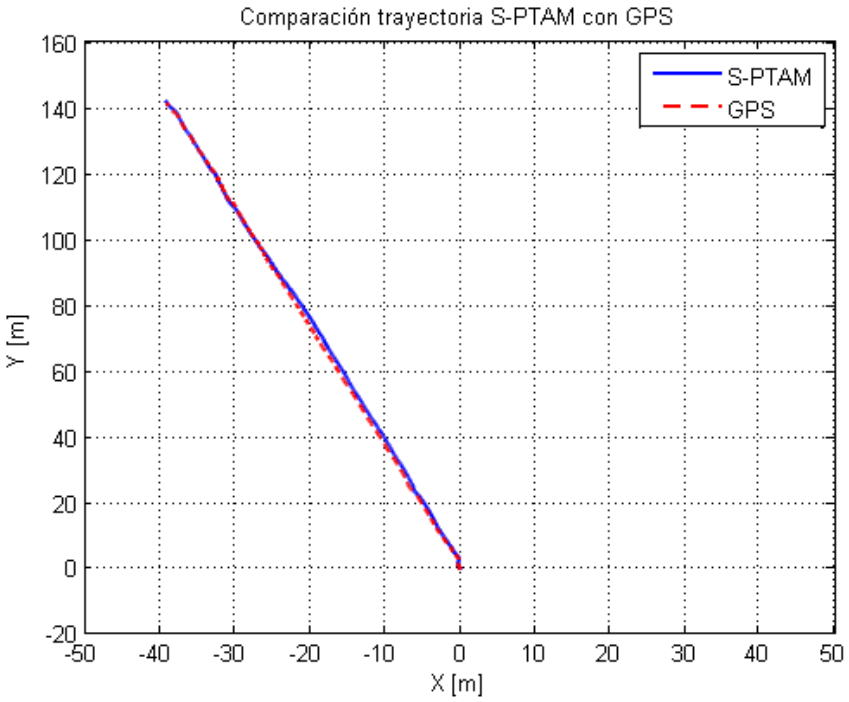


Figura 69: Simulación realizada sobre la NUC, de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 4 del dataset.

Al simular la secuencia número tres, no pudo generarse la trayectoria completa en tiempo real, generando la detención de la simulación. Esto ocurrió principalmente en la parte donde el robot realiza marcha atrás y gira (la parte de mayor complicación y dificultad para que el método detecte los puntos). La trayectoria obtenida es la de la Figura 70 donde puede verse en la sección marcada, que la trayectoria no es completa, la misma fue interrumpida por el error del método.

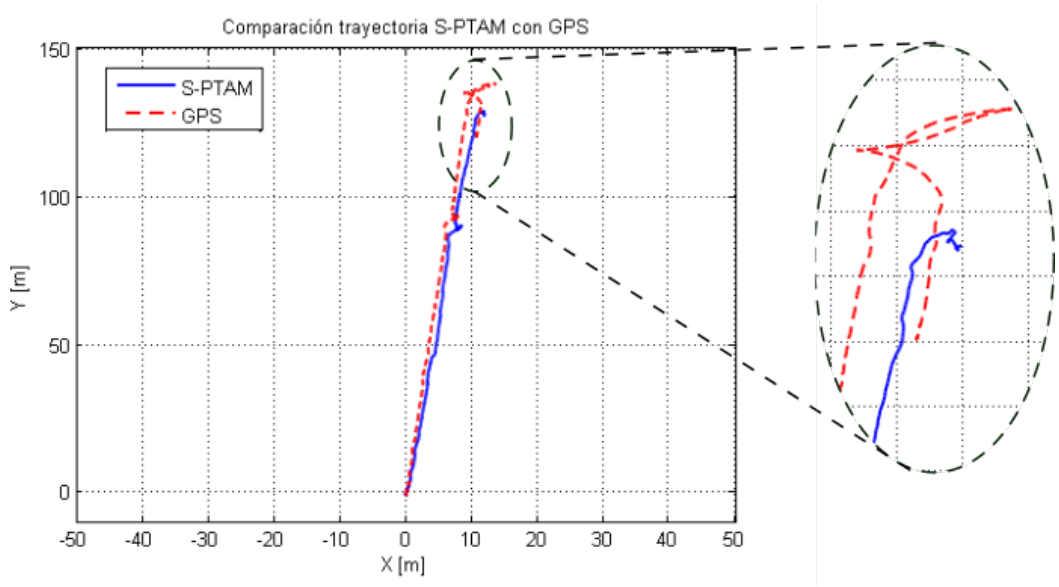


Figura 70: Simulación realizada sobre la NUC, de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 3 del dataset.

Para obtener la simulación completa y constatar que el problema fue exclusivamente de procesamiento, se afectó el tiempo de los mensajes en un factor de 0,9, es decir, en lugar de trabajar en tiempo real, se lo escaló por 0,9. De esta forma, dos mensajes con un segundo de diferencia entre ellos (según sus *timestamps*), serán enviados con 1,1 segundos de diferencia. Al realizar esta modificación no estamos simulando un funcionamiento idéntico de tiempo real, pero sí podemos corroborar si los errores son debidos al método o a la incapacidad de procesar la información en tiempo real. Con esta modificación se obtuvo la gráfica de la Figura 71

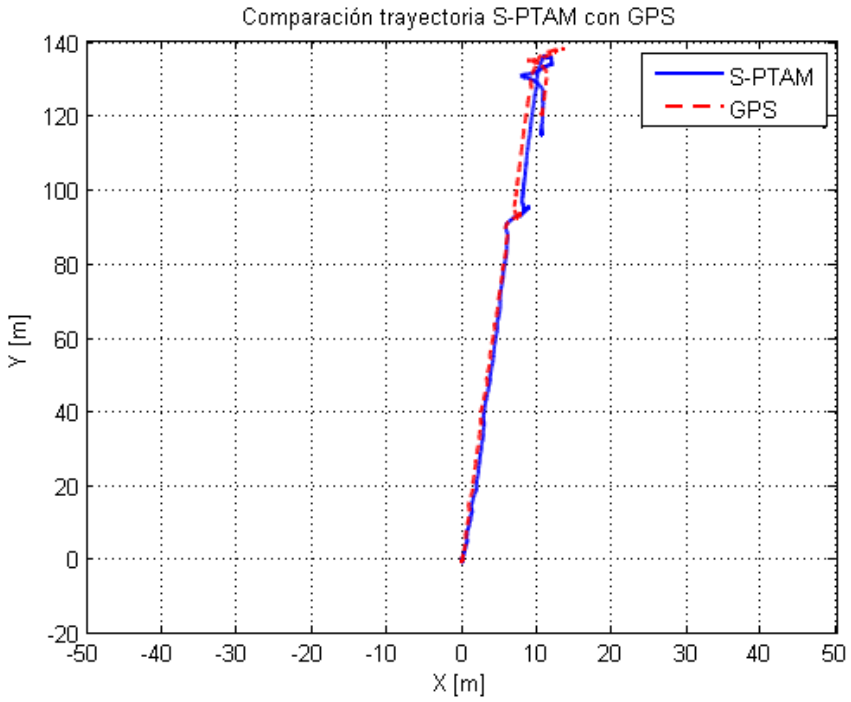


Figura 71: Simulación realizada sobre la NUC, con el tiempo escalado, de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 3 del dataset.

Por medio del mismo procedimiento, se afecto el tiempo por un factor de 0,5 para simular en la NUC la secuencia número uno que, como se mencionó previamente, posee una dificultad mayor a las anteriores por ser muy extensa y con numerosos giros y cambios de dirección. A partir de esto se obtuvo la Figura 72 donde puede verse que, al igual que en la simulación realizada en la otra computadora, S-PTAM mantiene la forma de la trayectoria en buena medida, pero posee errores de orientación.

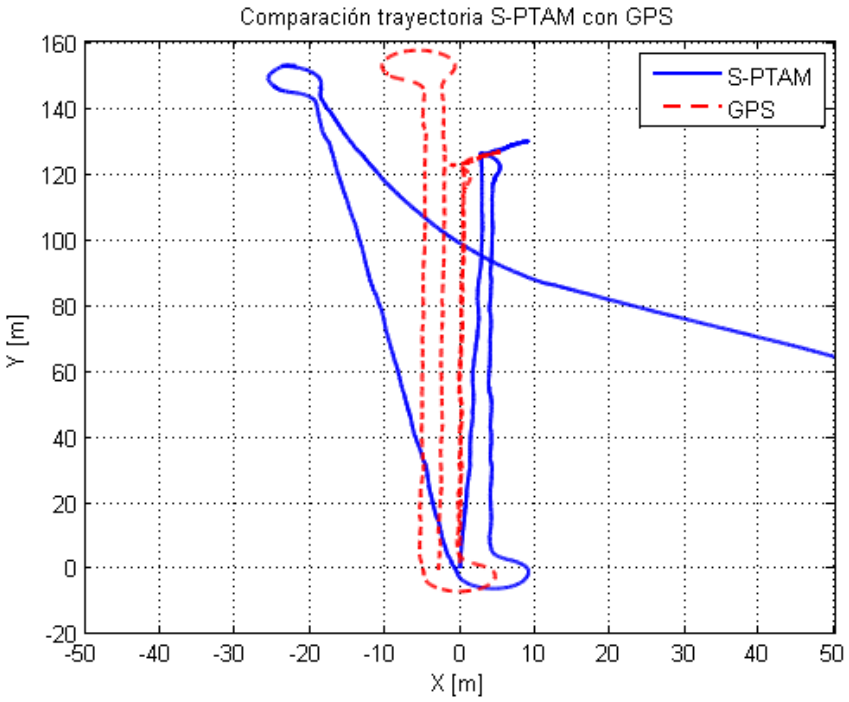


Figura 72: Simulación realizada sobre la NUC, con el tiempo escalado, de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 1 del dataset.

Con el objetivo de intentar aligerar el procesamiento necesario para la utilización del método, se intentó trabajar a una frecuencia de imágenes menor, es decir, utilizando solamente algunas de las adquiridas en lugar de todas. Las dos posibilidades planteadas, a partir de las 15 imágenes por segundo obtenidas originalmente, eran utilizar S-PTAM sobre un rosbag con imágenes guardadas a 7,5 Hz o a 5 Hz, ya que debía mantenerse una frecuencia constante de imágenes.

Se simularon los dos casos, sin embargo, los resultados mostraron que el funcionamiento empeoró, imposibilitando la creación de una trayectoria completa. Esto se debe a que si bien el método dispone de más tiempo para procesar cada imagen y calcular su posición, las diferencias entre una imagen y la otra son mayores, dificultando la detección del mismo punto en *frames* distintos.

Finalmente, para intentar mejorar los resultados sobre la nuc, reduciendo los tiempos de procesamiento, se disminuyó el tamaño de las imágenes a la mitad. De esta forma las nuevas imágenes tienen una resolución de (336 x 188) píxeles, permitiendo que el procesamiento de las mismas sea en un tiempo menor. Al afectar el tamaño de las imágenes es necesario ajustar la calibración del modelo de la cámara, para que sea acorde a estos nuevos valores. Puntualmente se actualizó a la nueva resolución, pero además se dividieron por dos los parámetros intrínsecos. En cuanto a la configuración de S-PTAM, se disminuyó de 1000 a 500 la cantidad de *features* a detectar ya que la imagen es más chica y se dividió por dos el *matchingCellSize* (utilizando en este caso el valor 15). El último parámetro determina el tamaño de las celdas en las que se subdivide la imagen. Al disminuir el tamaño de la imagen, el tamaño de estas celdas también debió ser disminuido. Esta subdivisión de las imágenes se realiza para sólo intentar detectar un punto, de una imagen a otra, en las cercanías de donde se encontraba. De otra forma, se

realizarían detecciones de puntos en regiones muy distintas de las imágenes, generando errores por *"matchear"* puntos que no se corresponden.

Por medio de esta modificación se volvió a simular la secuencia número tres, donde sólo había sido posible obtener la trayectoria completa afectando el tiempo por un factor de 0,9, en este caso se lo simula en tiempo real obteniéndose la gráfica de la Figura 73

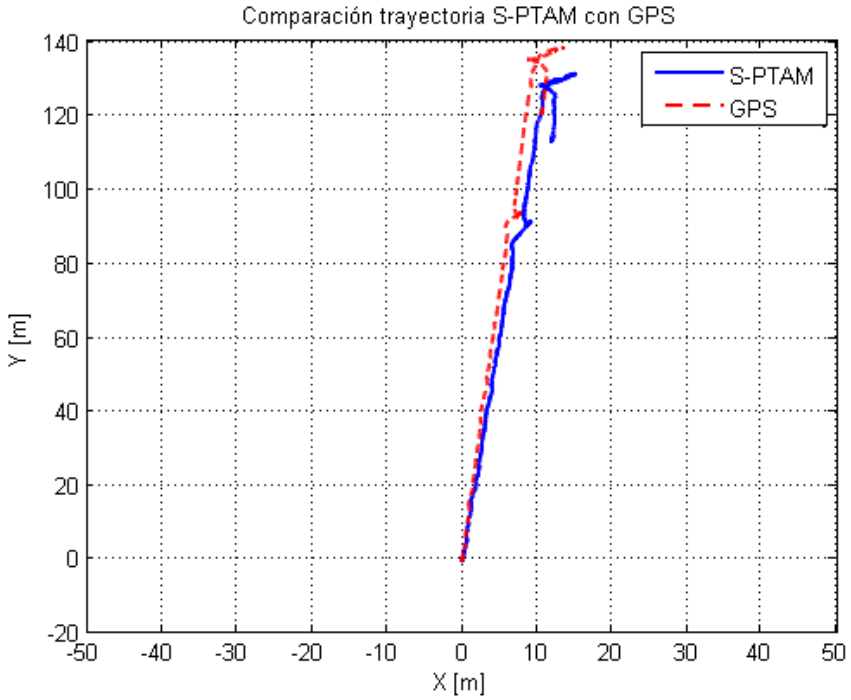


Figura 73: Simulación realizada sobre la NUC, con el tiempo escalado, de la trayectoria obtenida por S-PTAM a partir de la secuencia nº 1 del dataset.

Puede verse como se obtuvo una trayectoria similar a los casos anteriores pero trabajando en tiempo real, de todas formas, para que esta reducción de las imágenes permita mejorar significativamente el rendimiento debería analizarse a que tamaño llevarlas y en función de esto, realizar nuevamente la calibración por medio de Kalibr. Además, sería necesario adaptar nuevamente los parámetros de configuración de S-PTAM para encontrar el mejor funcionamiento.

4. Conclusiones

A lo largo del desarrollo del proyecto se analizaron las posibilidades para la inclusión de un sistema de sensado sobre el prototipo de robot desmalezador desarrollado en el CIFASIS. A partir del estudio se incluyeron los sensores necesario para permitir la adquisición de toda la información del robot que pudiera ser necesaria para conocer su posición, orientación y trayecto realizado en el tiempo. Para la selección de los sensores a utilizar se tuvo en consideración el entorno particular en el que se desplazará el robot, las lentas velocidades de trabajo, la autonomía del robot y, además, proveer de la información que pudiera ser necesaria a futuro para el desarrollo de métodos de localización que combinen distintos sensores o herramientas de adquisición de información.

En función de lo analizado se incluyeron en el robot una cámara estéreo, una unidad inercial, un GPS-RTK (que provee la referencia de mayor precisión para la evaluación de métodos de localización), la computadora que lee los sensores y los routers WiFi que generan la red inalámbrica que permite la comunicación del robot. Además de la incorporación de los sensores, se diseñaron e implementaron los programas que permiten el funcionamiento y adquisición de los datos provistos por todos los sensores, de forma de poder adquirir datos en el entorno de trabajo del robot para poder analizarlos y utilizarlos en el desarrollo de estrategias de detección de malezas, navegación y control del robot.

A partir del funcionamiento de los sensores en el campo de Zavalla, se generó un set de datos en condiciones reales de funcionamiento, estos datos fueron modificados y adaptados para que estén en concordancia con los estándares de ROS y, de esta forma, poder ser utilizados por algoritmos que trabajen bajo estos estándares. De esta forma se desarrollaron seis secuencias de trabajo con variaciones en el entorno, distancias recorridas y formas de desplazarse por los surcos.

Se utilizaron las secuencias obtenidas del set de datos para simular métodos de localización, por medio de un modelo cinemático que utiliza la odometría de las ruedas y por otro lado, por medio de un método de SLAM visual denominado S-PTAM. Se analizaron los resultados, determinando las ventajas, inconvenientes y limitaciones de cada uno. Obteniéndose de esta forma resultados que permiten visibilizar la confiabilidad de utilizar la odometría para distancias recorridas en situaciones ideales donde el robot no desliza, en contraposición, cualquier ruido en la dirección resultara en desviaciones de la trayectoria real. Por otro lado, los métodos de SLAM visual, utilizando la cámara estéreo, no se ven afectados por ruido en mediciones de los motores, errores en la alineación de las ruedas o si el robot desliza en lugar de tener un avance perfecto. Sus mayores inconvenientes son, en primer lugar, los errores de orientación al girar y volver a ingresar a los surcos y el alto costo computacional que posee el método.

En vista de los resultados obtenidos por medio de las simulaciones de S-PTAM, se puede considerar al mismo como una herramienta más para el posicionamiento del robot, ya que proporciona información muy valiosa y confiable del mismo, siempre y cuando los surcos sean visibles. Es evidente que necesita complementarse con otros sensores que faciliten la orientación al realizar giros, valiéndose por ejemplo de que los surcos son todos paralelos, o por medio de la incorporación de la información de la unidad inercial.

Como conclusión del proyecto, se incorporó y puso en funcionamiento un conjunto de sensores sobre el robot, de forma que, actualmente el mismo posee la capacidad de adquirir información de su entorno y estado de funcionamiento. Sé generó utilizando el sistema de sensado

del robot un set de datos real y confiable(ya que se utilizó el set de datos para la evaluación de métodos de localización) para permitir la creación y evaluación de algoritmos y métodos de navegación del robot. En el caso particular de este informe, se utilizó el set de datos para la evaluación de la viabilidad de la localización por medio de S-PTAM, detallando por medio de simulaciones su desempeño sobre el entorno de trabajo del robot y posibles enfoques de mejora del mismo.

4.1. Trabajos a futuro

En base a los resultados obtenidos, se plantean posibles mejoras al diseño, tanto para la adquisición de datos como para la localización del robot:

- Utilización de una unidad inercial de 9 ejes, es decir, que disponga de un magnetómetro que provea información de la orientación del robot en relación a los puntos cardinales de la Tierra. De esta forma la información obtenida del robot, en lugar de ser relativa a la posición y orientación inicial del robot, será absoluta (al igual que lo es el GPS).
- Realización de la fusión de sensores para mejorar los métodos de localización, por ejemplo, al combinar la información obtenida por S-PTAM con la obtenida por la IMU. De esta forma al ponderar la información de la IMU sobre la orientación del robot (siendo esta más confiable) se podrían suplir las dificultades de S-PTAM vistas durante la simulación en cuanto a la orientación durante y posteriormente a los giros. Una forma posible de realizar esta combinación sería por medio de un filtro de Kalman
- Adaptación de S-PTAM para funcionar de forma correcta y robusta sobre la NUC, para lo cual es necesario reducir el tiempo de procesamiento de las imágenes, aligerando el costo computacional que representa. Por otro lado sería posible incorporar nuevas restricciones al método, basado en el entorno particular donde se desplazará el robot (por ejemplo incorporando el concepto de que los surcos son paralelos). Esto facilitaría la localización del robot por medio del mismo utilizando sólo la cámara Zed.

Referencias

[1] Coordenadas utm. http://www.elgps.com/documentos/utm/coordenadas_utm.html. (Accessed on 06/05/2018).

[2] Free space path loss fspl — formula calculator — radio-electronics.com. http://www.radio-electronics.com/info/propagation/path-loss/free-space-formula-equation.php. (Accessed on 05/10/2018).

[3] Lsm6ds0 - inemo inertial module, 3-axis accelerometer, 3-axis gyroscope, always-on eco power mode - stmicroelectronics. http://www.st.com/en/mems-and-sensors/lsm6ds0.html. (Accessed on 05/08/2018).

[4] Posicionamiento en tiempo real con gps rtk — mundogeo. http://mundogeo.com/blog/2000/01/01/posicionamiento-en-tiempo-real-con-gps-rtk/. (Accessed on 04/29/2018).

[5] Reach rtk docs. https://docs.emlid.com/reach/. (Accessed on 05/01/2018).

[6] Rtk — geog 862: Gps and gnss for geospatial professionals. https://www.e-education.psu.edu/geog862/node/1845. (Accessed on 04/29/2018).

[7] stereo_image_proc - ros wiki. http://wiki.ros.org/stereo_image_proc. (Accessed on 06/06/2018).

[8] Abyarjoo, F., Barreto, A., Cofino, J., and Ortega, F. R. Implementing a sensor fusion algorithm for 3d orientation detection with inertial/magnetic sensors. In *Innovations and advances in computing, informatics, systems sciences, networking and engineering*. Springer, 2015, pp. 305–310.

[9] Bailey, T., and Durrant-Whyte, H. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine 13*, 3 (2006), 108–117.

[10] Calatayud, D. F. Ingeniería en geometría y topografía, 2015.

[11] D'Alessandro, A. R. Mapeo denso en tiempo real sobre sistemas de slam basados en visión estéreo, 2018.

[12] Durrant-Whyte, H., and Bailey, T. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine 13*, 2 (2006), 99–110.

[13] Hartley, R., and Zisserman, A. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[14] Howard, A. Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (2008), IEEE, pp. 3946–3952.

[15] Madgwick, S. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK) 25* (2010).

[16] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics 31*, 5 (2015), 1147–1163.

[17] Pire, T., Fischer, T., Castro, G., De Cristóforis, P., Civera, J., and Jacobo Berlles, J. S-PTAM: Stereo Parallel Tracking and Mapping. *Robotics and Autonomous Systems (RAS) 93* (2017), 27 – 42.

[18] Pire, T., Fischer, T., Civera, J., De Cristóforis, P., and Jacobo berlles, J. Stereo Parallel Tracking and Mapping for robot localization. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)* (September 2015), pp. 1373–1378.

[19] Pistarelli, M., Pire, T., and Kofman, E. Caracterizacion de un sistema gps rtk de bajo costo.

[20] Sun, S.-L., and Deng, Z.-L. Multi-sensor optimal information fusion kalman filter. *Automatica 40*, 6 (2004), 1017–1023.

[21] Valenti, R. G., Dryanovski, I., and Xiao, J. Keeping a good attitude: A quaternion-based orientation filter for imus and margs. *Sensors 15*, 8 (2015), 19302–19330.

[22] Weinstein, A. J., and Moore, K. L. Pose estimation of ackerman steering vehicles for outdoors autonomous navigation. In *Industrial Technology (ICIT), 2010 IEEE International Conference on* (2010), IEEE, pp. 579–584.

5. Anexos

5.1. Intel NUC

Intel® NUC Board NUC6CAYB

Technical Product Specification

Regulatory Model: NUC6CAY

November 2017
Order Number: J46865-003

The Intel NUC Board NUC6CAYB may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are documented in the Intel NUC Board NUC6CAYB Specification Update.

1 Product Description

1.1 Overview

1.1.1 Feature Summary

Table 1 summarizes the major features of the board.

Table 1. Feature Summary

Form Factor	4.0 inches by 4.0 inches (101.60 millimeters by 101.60 millimeters)
Processor	• Soldered-down quad-core Intel® Celeron® processor J3455 with up to 10 W TDP — Intel® HD Graphics 500 — Integrated memory controller — Integrated PCH
Memory	• Support for DDR3L 1600/1866 MHz SO-DIMMs • Support for 1600/1866 MHz memory speeds • Support for 4 Gb and 8 Gb memory technology • Support for up to 8 GB of system memory with two SO-DIMMs • Support for non-ECC memory • Support for 1.35 V low voltage JEDEC memory • 2 GB DDR3L 1600 MHz SO-DIMM pre-installed ***(included in Intel NUC Kit NUC6CAYS only)***
Graphics	• Integrated graphics support with Intel® Graphics Technology: — High Definition Multimedia Interface* (HDMI*) 2.0 full-sized back panel connector — VGA header (a *VGA cable is provided with the Intel NUC kits*)
Audio	• Intel® High Definition (Intel® HD) Audio via the HDMI v2.0 interface • Realtek ALC283 HD Audio via a stereo microphone/headphone 3.5 mm jack on the front panel • Compressed 5.1/7.1 digital audio through a mini-TOSLINK jack on the back panel • Digital microphone (DMIC) array header for support of digital voice assistants, such as Microsoft* Cortana (*dual digital array microphones are included with Intel NUC Kit NUCC6AYH and Intel NUC Kit NUC6CAYS*)
Peripheral Interfaces	• USB 3.0 ports: — Two ports are implemented with external front panel connectors (one blue and one amber charging capable) — Two ports are implemented with external back panel connectors (blue) • USB 2.0 ports: — Two ports via two single-port internal 1x4 1.25 mm pitch headers (white) — One port is reserved for an M.2 2230 Type E Module • Consumer Infrared (CIR)
Storage	• One SATA 6.0 Gb/s port (black) — Supports one 2.5“ SSD or HDD up to 9.5mm • One full-sized SDXC slot • 32 GB Embedded MultiMediaCard (e·MMC) onboard storage module ***(included in Intel NUC Kit NUC6CAYS only)***

continued

Table 1. Feature Summary (continued)

Expansion Capabilities	One M.2 Module supporting M.2 2230 cards (key type E) ***(prepopulated with Intel® Dual Band Wireless-AC 3168 module)***
BIOS	• Intel® BIOS resident in the Serial Peripheral Interface (SPI) Flash device • Support for Advanced Configuration and Power Interface (ACPI), Plug and Play, and System Management BIOS (SMBIOS)
LAN Support	Gigabit (10/100/1000 Mb/s) LAN subsystem using the Realtek* 8111HN Gigabit Ethernet Controller
Hardware Monitor Subsystem	Hardware monitoring subsystem, based on an ITE IT8987D embedded controller, including: • Voltage sense to detect out of range power supply voltages • Thermal sense to detect out of range thermal values • One processor fan header • Fan sense input used to monitor fan activity • Simple fan speed control
Wireless	Intel® Dual Band Wireless-AC 3168 module • Intel's 3rd-generation 802.11ac, Dual Band, 1x1 Wi-Fi + Dual Mode Bluetooth 4.2 • Maximum Transfer speed up to 433Mbps • Supports Intel® Smart Connect Technology • Pre-installed in M.2 2230 slot
Operating System	Supports Microsoft* Windows* 10 Home and Microsoft* Windows* 10 Pro • Intel NUC Kit NUC6CAYS comes with Windows 10 Home pre-installed on the eMMC storage device • Other operating system (OS) support may be available. Please check your OS distributor for support details.
Additional Features	• Integrated HDMI CEC • Intel® Platform Trust Technology

5.2. GPS-RTK Emlid Reach

Electrical specs

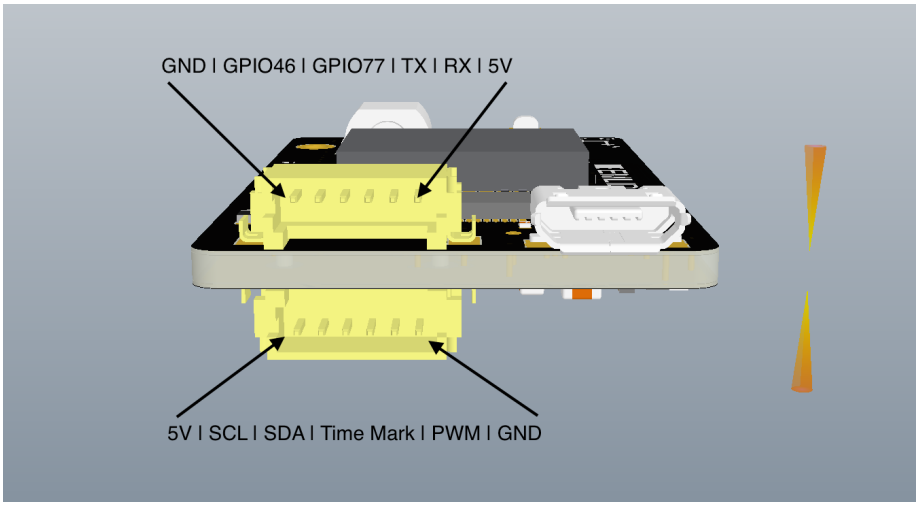
Maximum ratings

Name	Value
Inout voltage on USB and DF13 connectors	4.75 - 5.5 V
Logic levels on all pins	3.3 V
Max input voltage on all pins	5.5 V
Antenna DC bias	3.3 V
Antenna output current	100 mA
Max current consumption @5V	500 mA
Normal current consumption @5V	200 mA
Current limit on USB OTG	1000 mA
Temperature range	0 +40 C (-40 +85 C)^

Officially Intel Edison is rated 0C to 40C, but Intel also claims they are performing temperature tests with good results, but just are not yet ready to officially rate Edison as extended

temperature range device. Users report successful tests down to -40 C.

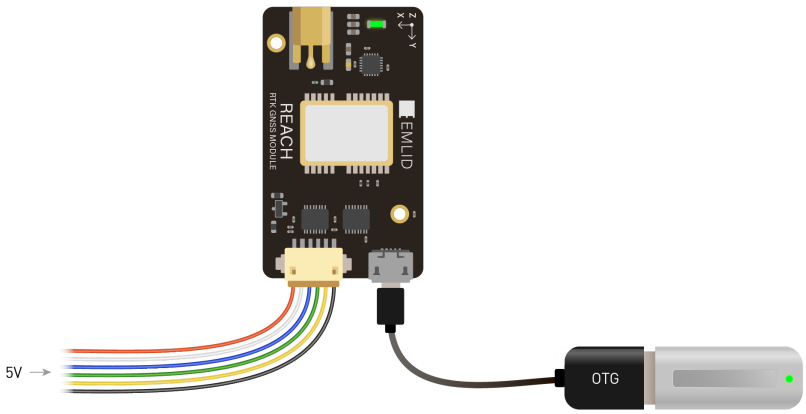
Connectors pinout



- GPIO46, GPIO77, PWM, SCL, SDA, TX, RX are connected to Intel Edison via buffers and are 3.3 V logic level, 5 V tolerant.
- TX and RX belong to UART1 on Intel Edison
- SCL and SDA belong to I2C1 on Intel Edison, I2C1 also could be used to communicate with internal magnetometer in MPU9250.
- PWM belongs to PWM3 GPIO183 on Intel Edison

- Time Mark input is connected directly to U-blox chip for low latency, it includes an over-voltage clamp, pull up and current limiting resistor.

USB OTG



Reach can both receive power from USB, acting as a device and source power to the port acting as a host. To use Reach in OTG mode you will need to connect 5V power source to DF13 connector pins (5 V, GND) and use OTG USB cable.

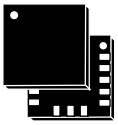
5.3. IMU LSM6DS0

life.augmented

LSM6DS0

iNEMO inertial module:
3D accelerometer and 3D gyroscope

Datasheet - production data



LGA-16L (3x3x0.86 mm)

Features

- Analog supply voltage: 1.71 V to 3.6 V
- Independent IOs supply (1.71 V)
- "Always on" eco power mode down to 1.8 mA
- 3 independent acceleration channels and 3 angular rate channels
- ±2/±4/±8/±16 *g* full scale
- ±245/±500/±2000 dps full scale
- SPI/I^2C serial interface
- Embedded temperature sensor
- Embedded FIFO
- ECOPACK®, RoHS and "Green" compliant

Applications

- GPS navigation systems
- Impact recognition and logging
- Gaming and virtual reality input devices
- Motion-activated functions
- Intelligent power saving for handheld devices
- Vibration monitoring and compensation
- Free-fall detection
- 6D orientation detection

Description

The LSM6DS0 is a system-in-package featuring a 3D digital accelerometer and a 3D digital gyroscope. ST's family of MEMS sensor modules leverages the robust and mature manufacturing processes already used for the production of micromachined accelerometers and gyroscopes.

The various sensing elements are manufactured using specialized micromachining processes, while the IC interfaces are developed using CMOS technology that allows the design of a dedicated circuit which is trimmed to better match the sensing element characteristics.

The LSM6DS0 has a full-scale acceleration range of ±2/±4/±8/±16 *g* and an angular rate range of ±245/±500/±2000 dps. The LSM6DS0 has two operating modes in that the accelerometer and gyroscope sensors can be either activated at the same ODR or the accelerometer can be enabled while the gyroscope is in power-down.

The LSM6DS0 is available in a plastic land grid array (LGA) package.

Table 1. Device summary

Part number	Temp. range [°C]	Package	Packing
LSM6DS0	-40 to +85	LGA-16L (3x3x0.86 mm)	Tray
LSM6DS0TR	-40 to +85		Tape and reel

5.4. Mikrotik Metal 52ac

Metal 52 ac

The Metal is a completely waterproof and rugged outdoor wireless device with a super high output power.

Now with 802.11ac support, a Gigabit Ethernet port and a selectable wireless band (2.4GHz or 5GHz, up to 80 MHz wide channel). The new Gigabit port will help you utilize the full benefit of 802.11ac high speed wireless.

The Metal 52 ac supports 2.4GHz, 5GHz and all the legacy wireless modes as well (802.11 b/g/n for 2.4GHz and 802.11 ac/n/a for 5GHz).

Fully sealed, industrial design metal case, powered by RouterBOARD and RouterOS. With 1300mW of output power - to reach the last mile, in any conditions. The Metal comes with an AP software license, so you can attach your favorite antenna to use it as an AP, to make wireless point-to-point links or as a CPE- whatever you prefer!

The Metal 52 ac includes a Dual Band 2.4/5GHz Omni directional antenna (6dBi 2.4GHz, 8dBi 5GHz), so you can use the unit right out of the box, or use your own antenna.

Wireless specifications

5 GHz		
RATE	Tx (dBm)	Rx (dBm)
1MBit/s	31	-97
11MBit/s	31	-91
6MBit/s	31	-93
54MBit/s	28	-77
MCS0	31	-93
MCS7	27	-72

2.4 GHz		
RATE	Tx (dBm)	Rx (dBm)
6MBit/s	31	-93
54MBit/s	27	-78
MCS0	30	-93
MCS7	26	-74
MCS9	22	-69

MikroTik Metal 52 ac

Specification

Product code	RBMetalG-52SHPacn (International) RBMetalG-52SHPacn-US (USA)			
CPU	QCA9556			
CPU nominal frequency	720 MHz			
Size of RAM	64 MB			
Memory type	Flash			
Memory size	16 MB			
10/100/1000 Ethernet ports	1			
Wireless	5 GHz		2.4 GHz	
Operating frequency	International	5150 - 5875 MHz	International	2412 - 2484 MHz
	USA	5170 - 5250 MHz 5725 - 5835 MHz	USA	2412 - 2462 MHz
Protocols	802.11a/n/ac		802.11b/g/n	
Chains	Single-chain			
Wireless chip	QCA9889			
PoE in	Yes			
Supported input voltage	10 - 30 V			
PCB temperature monitor	Yes			
Voltage monitor	Yes			
Dimensions	177 x 44 x 44 mm			
Operating temperature	-40°C .. +70°C tested			
License level	4			
Operating System	RouterOS			
Max power consumption	11 W			
Weight	Package: 778g; Unit: 300g			

Included

24V 0.8A Power adapter

2x Mounting loops

Gigabit PoE injector

Angled N connector

2x Metal rings

Dual Band 2.4/5Ghz (6dBi 2.4Ghz, 8dBi 5GHz)

5.5. Mikrotik Groove 52ac

Groove 52 ac

Our smallest outdoor series model - a fully featured wireless RouterBOARD powered by RouterOS. Weatherproof, durable and ready to use. Now with 802.11ac support, a Gigabit Ethernet port and a selectable wireless band (2.4GHz or 5GHz, up to 80 MHz wide channel). The new Gigabit port will help you utilise the full benefit of 802.11ac high speed wireless.

The Groove supports 2.4GHz, 5GHz and all the legacy wireless modes as well (802.11 b/g/n for 2.4GHz and 802.11 ac/n/a for 5GHz).

Two models are available, Groove for CPE and PtP connections, and GrooveA which also supports AP mode.

The device has a built-in N-male connector and pole attachment points, so you can attach it to an antenna directly, or use a standard antenna cable. LED signal indicators make it easy to install and align.

The Groove A (AP model) includes a Dual Band 2.4/5GHz Omni directional antenna (6dBi 2.4GHz, 8dBi 5GHz), so you can use the unit right out of the box, or use your own antenna.

Box contains: Groove unit, mounting loops, PoE injector, power adapter, 6/8dBi 2.4/5GHz Omni antenna (Only A model).

24V 0.38A Power adapter

Omni antenna

Mounting loops

Gigabit PoE injector

Specifications

Product code	RBGrooveGA-52HPacn (International, AP version) RBGrooveGA-52HPacn-US (USA, AP version) RBGrooveG-52HPacn (International) RBGrooveG-52HPacn-US (USA)			
CPU nominal frequency	QCA9556 720 MHz			
Size of RAM	64 MB			
Storage	16 MB Flash			
10/100/1000 Ethernet ports	1			
Wireless	5 GHz		2.4 GHz	
Operating frequency	International	5150 - 5875 MHz	International	2412 - 2484 MHz
	USA	5170 - 5250 MHz 5725 - 5835 MHz	USA	2412 - 2462 MHz
Protocols	802.11ac		802.11b/g/n	
Channel width	5*/10*/20/40/80 MHz		5/10/20/40 MHz	
Wireless chip model	QCA9889			
Chains	Single-chain			
PCB temperature monitor	Yes			
Voltage monitor	Yes			
PoE in	Yes			
Supported input voltage	11 V - 30 V (passive PoE)			
Operating temperature	-40 to 70° C			
License level	4 (RBGrooveGA version) 3 (RBGrooveG version)			
Dimensions	177 x 44 x 44 mm			
Power consumption	5 W			

Wireless specifications

Rate (2.4 GHz)	Tx	Rx
1MBit/s	24	-97
11MBit/s	21	-91
6MBit/s	24	-93
54MBit/s	21	-77
MCS0	24	-93
MCS7	20	-72

Rate (5 GHz)	Tx	Rx
6MBit/s	26	-93
54MBit/s	21	-78
MCS0	26	-93
MCS7	21	-74
MCS9	19	-69

*Supports only 802.11an protocol.