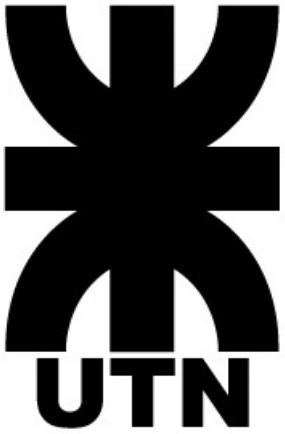
Universidad Tecnológica Nacional



Facultad Regional Córdoba

Proyecto Final

Hardware HAT de sensor odométrico basado en la fusión de visión y mediciones inerciales

Alumnos

Campetella Nazareno - 78006
Vera Causich Mariano - 66878

Director

Perez Paina, Gonzalo

2 de febrero de 2023

Índice

1. Introducción

Para que un robot móvil pueda realizar una tarea de forma autónoma o semi-autónoma es necesario que sea capaz de resolver varios problemas dentro de los cuales se encuentra la localización, o sea, determinar su posición y orientación (pose) dentro de su entorno de trabajo. En robots móviles con ruedas esto se logra principalmente utilizando técnicas de dead-reckoning (navegación por estima) a partir de mediciones de codificadores ópticos incrementales acoplados en las ruedas, conocido como odometría. Este tipo de localización tiene diferentes fuentes de errores tanto sistemáticos, que producen una deriva en el tiempo, como no sistemáticos que incrementan de forma monótona la incertidumbre de la estimación. Resulta entonces común utilizar diferentes sensores junto a técnicas de fusión sensorial para obtener una mejor estimación.

La Figura 1 muestra dos robots móviles con ruedas cuya tracción es del tipo diferencial. La Figura 1(a) muestra el robot experimental RoMAA-II (Robot Móvil de Arquitectura Abierta) y la Figura 1(b) al robot industrial tipo AMR Aimu. Ambos robots fueron desarrollados en el Centro de Investigación en Informática para la Ingeniería (CIII).

(a) RoMAA-II

(b) AMR Aimu

Figura 1: Robots móviles con rueda de tracción diferencial

2. Marco de desarrollo

El proyecto PID-UTN 8477 “Navegación y control de robot móvil industrial AMR basado en ROS2”(2022-2025) tiene dentro de sus objetivos ïmplementar algoritmos de navegación y control para robots móviles aplicados a la logística interna en la industria utilizando el Sistema Operativo de Robótica ROS”. La propuesta para estas practicas se enmarca dentro de los objetivos del PID-UTN antes mencionado y pretende lograr la primera interacción en el diseño del hardware que permita desarrollar un sensor de odometría visual-inercial aplicado a la navegación de robots móviles con ruedas como el Robot RoMAA-II y el robot Aimu.

3. Propuesta de desarrollo

3.1. Objetivos generales

Desarrollar una placa que disponga de una cámara y sensores inerciales compuesto por acelerómetros y giróscopos para ser utilizada como plataforma experimental en el desarrollo de algoritmos de fusión sensorial para la estimación de la pose de un robot móvil con ruedas. Dicha placa deberá estar diseñada como HAT (Hardware Attached on Top) para una computadora (SBC) Raspberry Pi.

3.2. Requerimientos

Los requerimientos técnicos dispuestos por el CIII son:

- Contar con una cámara tipo global shutter y lente con montaje M12, resolución VGA o superior y tasa de muestreo de 30 Hz o superior.
- Seleccionar acelerómetro y giróscopo de 3 ejes con rango de medición de +/- 2g o superior, 500 dps o superior, respectivamente. Resolución mínima de 12 bits.
- Acceso a memoria EEPROM.
- Alimentación del HAT mediante Raspberry Pi (5 V).
- Iluminación activa de la cámara.
- PCB de formato adecuado para actuar como HAT de la computadora SBC (Single Board Computer) Raspberry Pi.
- Documentación en repositorio GitLab del CIII.

3.3. Objetivos particulares

Estos son los objetivos particulares que se deben cumplir en las practicas:

- Seleccionar los sensores a utilizar (cámara y unidad inercial) según los requerimientos técnicos y que sean vigentes en el mercado.
- Diseñar el circuito esquemático para la conexión de los sensores a una computadora Raspberry Pi.
- Instalar un sistema operativo que sea compatible con ROS y los demás componentes.
- Diseñar la placa de circuito impreso (PCB) de forma integra y profesional. A partir de este diseño se deberán obtener las transformaciones rígidas entre los diferentes sensores necesarias para la fusión de sus mediciones.
- Desarrollar un paquete ROS (Robot Operating System) con el/los nodo/s adecuado/s para obtener mediciones y ajustar los parámetros de los sensores utilizados.

- Proporcionar una iluminación adecuada de acuerdo al entorno de trabajo.
- Documentación sobre la toma de decisiones.

3.4. Ensayos

Se pondrá a disposición un robot RoMAA-II completamente funcional para ensayar el hardware desarrollado. Luego, para validar los requerimientos y objetivos de estas practicas tomaremos los siguientes pasos:

- Para la verificación de la correcta lectura de los sensores se utilizarán las herramientas gráficas RQt del framework de robótica ROS que permiten graficar las mediciones de los sensores inerciales y visualizar la cámara.
- Se utilizará la herramienta gráfica RViz de ROS para visualizar las transformaciones rígidas entre los sensores.

4. Plataforma de experimentación

4.1. RoMAA

El Robot Móvil de Arquitectura Abierta (RoMAA)[1] es un robot de tracción diferencial desarrollado íntegramente en el Centro de Investigación en Informática para la Ingeniería (CIII), para ser utilizado como plataforma de experimentación en los campos de investigación de robótica móvil y visión robótica. Dicho desarrollo, incluye la mecánica, el sistema embebido de a bordo para el control de bajo nivel del robot y el software de programación y control de alto nivel que corre sobre la PC de a bordo del robot. La programación del robot se realiza utilizando una librería de comunicación que permite enviar comandos y recibir información de la odometría mediante mensajes de alto nivel. En la figura 2 podemos ver el modelo prototipo del RoMAA junto a sensores y computadora a bordo.

Figura 2: Prototipo del RoMAA

El diseño y desarrollo del RoMAA surge ante la necesidad de disponer de un vehículo adecuado para el ámbito de la investigación, capaz de adaptarse a distintos experimentos en las áreas de la robótica móvil y visión por computadora. Esta necesidad implica poder acceder sin restricciones a los distintos componentes del robot, incluyendo hardware, firmware, software de alto nivel, etc. Para lo cual la mejor alternativa es un desarrollo de una plataforma de arquitectura abierta, esencial para ser adecuada a las necesidades particulares de cada usuario/investigador.

4.2. RoMAA-II

A partir de la evaluación de las características constructivas, funcionales y de costos de fabricación del prototipo se decide introducir algunas modificaciones en el diseño, tanto en el hardware

como en el software, lo que deriva en la nueva plataforma móvil RoMAA-II desarrollada en el marco del proyecto homologado por la Universidad Tecnológica Nacional. Las principales modificaciones están relacionadas a: la mecánica del sistema de tracción, el hardware del controlador embebido junto con el firmware asociado, para el cual se crearon librerías especificas de la arquitectura del microcontrolador. El RoMAA-II pretende mejorar las características tanto constructivas como de prestaciones del prototipo RoMAA.

Figura 3: Versión II del RoMAA

La arquitectura abierta implica tener acceso completo al hardware (circuitos esquemáticos y placas de circuitos impresos) y al firmware del controlador diferencial encargado del control de bajo nivel, además de disponer del código fuente del conjunto de programas incluyendo las librerías de comunicación y los módulos de mas alto nivel que se ejecutan en la PC de a bordo. Esta arquitectura abierta es esencial para poder adecuar la plataforma de experimentación a las necesidades particulares de cada usuario/investigador.

5. Raspberry Pi

La Raspberry Pi se refiere a computadoras SBC (Single Board Computer) de bajo coste y formato compacto destinado al desarrollo para hacer accesible la informática a todos los usuarios. Surge de la necesidad de promocionar la enseñanza de informática en las escuelas y estas acabaron siendo mas popular de lo que se esperaba, donde vemos que en la actualidad son usadas en el área de la robótica. [2]

Figura 4: Logo Raspberry Pi

Todos los diseños de Raspberry Pi se basan en el hardware libre y habitualmente se utilizan también sistemas operativos libres basados en GNU/Linux. Para este microcomputador se ha desarrollado Raspberry Pi OS (Antes conocido como Raspbian) que es una versión personalizada de Debian, de ahí el nombre. Adicionalmente se pueden instalar sistemas operativos.

Es el buen rendimiento y el bajo coste en donde radica el gran éxito de la Raspberry Pi. Cuentan además con gran conectividad y de conexiones GPIO que permiten desarrollar una gran variedad de proyectos educativos.

El funcionamiento es el mismo que una PC de escritorio, ya que el nivel fundamental tiene los mismos componentes. Las placas Raspberry Pi se basan en un SoC de arquitectura ARM de bajo consumo y buen rendimiento. Se acompañan por un modelo de memoria RAM, cuya capacidad varia según el modelo. Disponen de varias salidas de vídeo y en las versiones mas nuevas se agrega un jack de 4 polos, para entrada de micro y salida de audio. Cuentan con un lector de tarjetas donde se instala el sistema operativo y varios puertos USB. También dispone de una gran cantidad de conectores GPIO para poder desarrollar una gran cantidad de proyectos.

5.1. Raspberry Pi 3 modelo B/B+

5.1.1. Descripción

Para este proyecto contamos con dos Raspberry Pi 3, una con modelo B y otra B+. A continuación, vamos a dar breves descripciones y especificaciones de estas.

Raspberry Pi 3 modelo B: Sacada a la luz en el año 2016, renueva procesador, una vez mas de la compañía Broadcom, un Quad-Core, pero pasa de 900 MHz a 1.2 GHz. Mantiene la RAM en

1 GB. Su mayor novedad fue la inclusión de Wi-Fi y Bluetooth (4.1 Low Energy) sin necesidad de adaptadores.

Raspberry Pi 3 modelo B+: Apareció en marzo del 2018 para actualizar la Raspberry Pi modelo B y entre sus mejoras cuenta con un nuevo procesador y mejor conectividad, asi pasa de tener 1.2 GHz a tener 1.4 GHz y en cuanto a la conectividad inalámbrica ahora incorpora doble banda a 2.4 GHz y 5 GHz, y su nuevo puerto Ethernet se triplica, pasa de 100 Mbits/s en el modelo anterior a 300 Mbits/s en su nuevo modelo, también cuenta con Bluetooth 4.2 (Low Energy). [3]

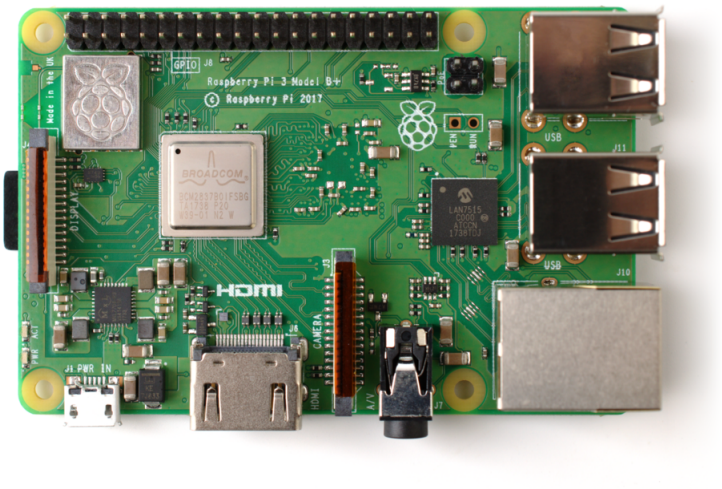


Figura 5: Raspberry Pi 3 modelo B+

5.1.2. Especificaciones

En la tabla 1 podemos apreciar las especificaciones y comparaciones entre los dos modelos disponibles a usar en el proyecto, si bien la elección final de la placa a usar claramente sera el modelo B+ por su mayor rendimiento, tener dos placas disponibles es realmente útil para la división de tareas en el grupo de trabajo.

Especificaciones	**Raspberry Pi 3 modelo B**	**Raspberry Pi 3 modelo B+**
SoC	Broadcom BCM2837	
CPU	1.2 GHz 64-bit quad-core ARMv8	1.4 GHz 64-bit quad-core ARMv8
Instrucciones	RISC de 64 bits	
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1, 1080p30 H.264/MPEG-4 AVC	
Memoria	1 GB (compartidos con la GPU)	
Puertos USB 2.0	4	
Entradas de video	Conector MIPI CSI que permite instalar un módulo de cámara.	
Salidas de video	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD.	
Almacenamiento	MicroSD	
Conectividad de Red	Puerto RJ-45 de 10/100Mbps vía hub USB. Wi-Fi 802.11bgn . Bluetooth 4.1.	Puerto RJ-45 de 10/100/1000Mbps vía hub USB limitado a 300 Mbits/s. Wi-Fi 802.11ac de doble banda. Bluetooth 4.2 BLE.
Periféricos bajo nivel	17 x GPIO y un bus HAT ID	
Consumo energético	800 mA (4.0 W)	
Fuente de alimentación	5 V vía Micro USB o puerto GPIO	
Dimensiones	85 mm x 53 mm	

Cuadro 1: Especificaciones Raspberry Pi modelo B/B+

5.1.3. GPIO

GPIO son las siglas de General Purpose Input/Output, es decir, Entrada/Salida de propósito general y son una gran cantidad de pines disponibles en la Raspberry Pi que contienen entradas y salidas totalmente configurables por el usuario para que se pueda interactuar con esta. De esta forma, la Raspberry Pi no solo dispone de una serie de puertos e interfaces para conectar varios dispositivos, sino que agrega estos pines GPIO para que puedas agregar otros dispositivos electrónicos o proyectos que creados por el usuario.

En la figura 6 podemos apreciar el pinout del GPIO del modelo de Raspberry Pi usada en el proyecto. Entre ellos podemos clasificarlos como: Pines de alimentación (5 V, 3.3 V y GND), Pines configurables por el usuario (GPIO) y Pines especiales destinado a conexiones o interfaces (SDA, SCL, TXD, RXD, MOSI, MISO, SCLK, CE0, CE1, etc.) [4]

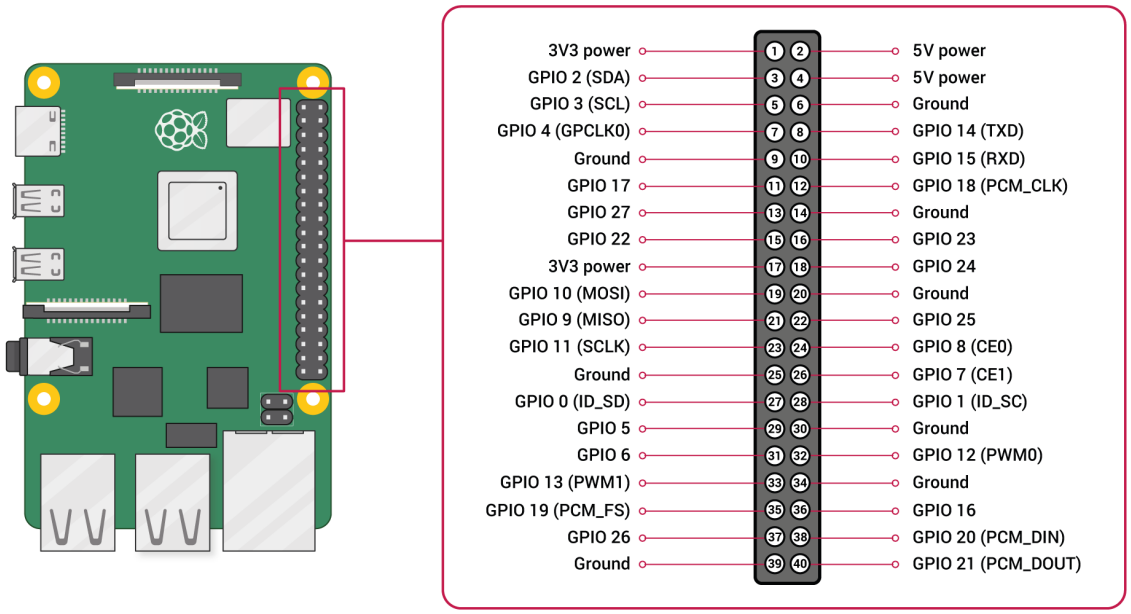


Figura 6: GPIO de Raspberry Pi 3 modelo B/B+

6. Requerimientos HAT

Las placas Raspberry Pi con conectores GPIO de 40 W (Modelo B+ y en adelante) se han diseñado específicamente teniendo en cuenta las placas complementarias. Estas placas admiten 'HAT' (Hardware Attached on Top).

6.1. Requerimientos

Los requerimientos a tener en cuenta son propios de Raspberry y se encuentran en un repositorio Github [5] detallando cada unos de estos.

- Cumplir requerimientos básicos de una placa adicional.
- Que tenga una ID EEPROM valida (incluyendo la información del vendedor, mapa GPIO y un árbol de información de dispositivos valido).
- Debe tener un conector GPIO de 40 conectores de tamaño completo.
- Tiene que cumplir con las especificaciones mecánicas.
- El conector GPIO tiene que contar con al menos 8 mm de espacio entre el HAT y la Raspberry.
- Si se retroalimenta a través del conector GPIO, el HAT debe poder suministrar 1.3 A de forma continua a la Raspberry, sin embargo se recomienda 2 A continuos.

6.2. Especificaciones mecánicas

Uno de los principales requerimientos y que podemos ver de forma mas notable son sus especificaciones mecánicas, es decir, la forma y medidas que se requieren para que la placa sea tratada como HAT. En la figura 7 podemos apreciar todo lo requerido.

6.3. Memoria EEPROM

El mayor cambio con las placas complementarias HAT en comparación con las placas más antiguas diseñadas para los modelos A y B es que el encabezado de 40 W tiene 2 pines especiales (ID_SC e ID_SD) que están reservados exclusivamente para conectar una 'EEPROM de ID'. La ID EEPROM contiene datos que identifican la placa, le dice al Pi cómo deben configurarse los GPIO y qué hardware hay en la placa. Esto permite que el software Pi identifique y configure automáticamente la placa adicional en el momento del arranque, incluida la carga de todos los controladores necesarios.

Esta debe contar con los siguientes requerimientos:

- Debe ser de tipo 24CXX y tener 3.3 V I2C.
- 16 bit de tipo direccionables.

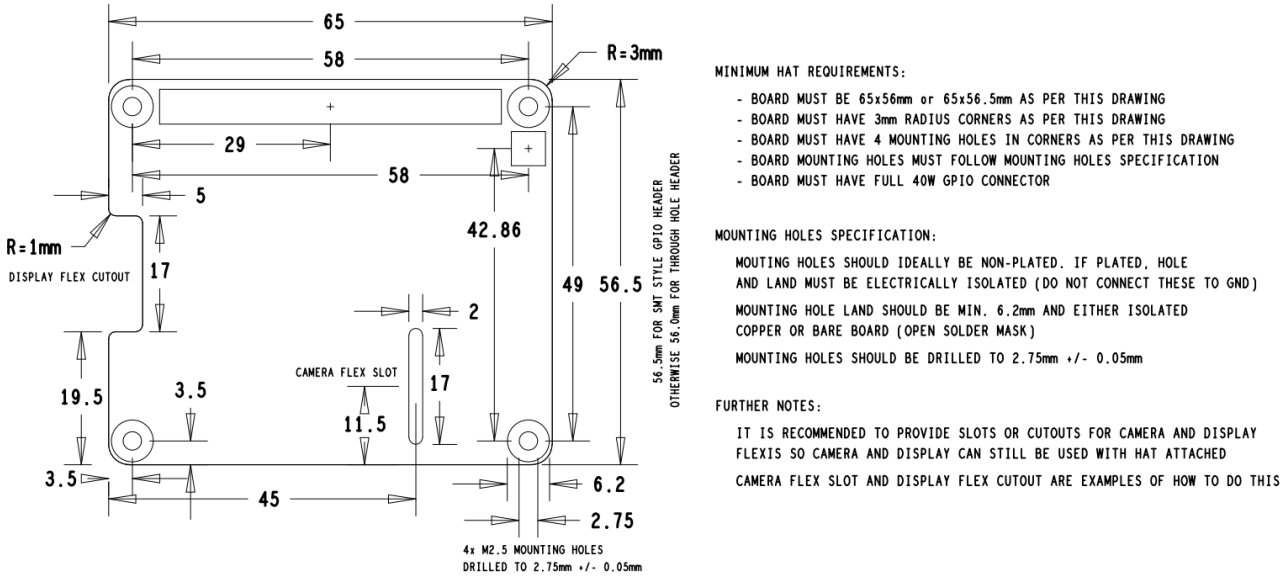


Figura 7: Especificaciones mecánicas HAT

- Contar con un pin de protección contra escritura y este debe ser compatible y proteger toda la memoria del dispositivo.
- No puede tener EEPROM donde los bits de dirección inferior I2C seleccionan la pagina.
- Solo se admite el modo I2C de 100 KHz.
- No son compatibles dispositivos que realizan ensanchamiento de clock I2C.

Una pieza recomendada que satisface las restricciones anteriores es CAT24C32, que es un dispositivo de 32 Kbit (4 Kbyte). El tamaño mínimo de EEPROM requerido es variable y depende del tamaño de las cadenas de datos del proveedor en la EEPROM y de si se incluye un blob de datos de árbol de dispositivos (y su tamaño) y si se incluye cualquier otro dato específico del proveedor [6].

7. Fusión sensorial

Una unidad de medición inercial (IMU, por sus siglas en ingles) contiene mediciones de orientación, velocidad angular y aceleración para ser usado en la robótica. Si bien esta es muy eficaz, a menudo tiene errores sistemáticos otorgando inexactitudes y desviaciones al sistema. Por eso, se agrega una cámara para funcionar como sensor óptico o visual. Esto se puede lograr a través de flujo óptico, seguimiento de características y transformación de perspectiva, todas técnicas de visión por computadora. El resultado de estos dos sensores se fusionarían para mejorar la estimación del robot.

La fusión de sensores es la combinación de mediciones de múltiples sensores diferentes para crear una medición más precisa. La estimación más precisa se deriva utilizando un filtro de Kalman extendido o similar basado en las mediciones de entrada. El objetivo de este proyecto es preparar el hardware necesario para luego desarrollar mediante software la fusión de los sensores que contiene el HAT. Esto va a lograr estabilizar los datos ruidos de la IMU (Inertial Measurement Unit) fusionando estos con el movimiento extraído de los cálculos hechos por el flujo óptico.

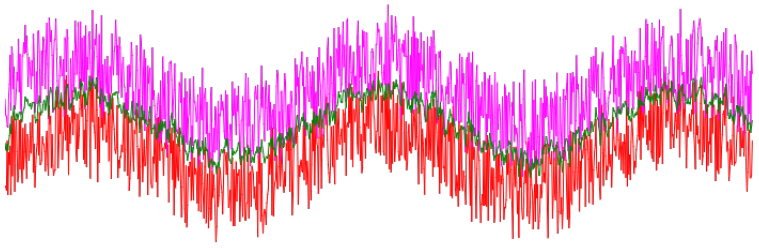


Figura 8: Ejemplo de fusión sensorial

La figura 8 ejemplifica el principio básico que implica la fusión de sensores, en los que dos conjuntos de datos (rosa y rojo) se fusionan para formar la estimación (verde). Esta información fue extraída de este proyecto sobre flujo óptico [7].

8. Contexto del sistema

Como podemos observar en la figura 9 se encuentra el diagrama en bloques de nuestro sistema. Nuestro bloque principal va a ser la plataforma experimental RoMAA donde montaremos el computador SBC Raspberry Pi junto al HAT desarrollado en el proyecto, el cual se compone principalmente de 3 secciones: Memoria EEPROM, Unidad de Medición Inercial y una Cámara.

La Memoria EEPROM nos servirá para identificar y configurar nuestro HAT en cualquier modelo 3B/B+ que se le conecte e ira conectada al GPIO de está a través de los conectores del HAT y su protocolo de comunicación sera I2C.

La unidad de medición inercial sera la encargada de devolvernos datos de acelerómetro y giróscopo del robot. También estará conectada a través del GPIO y su comunicación es I2C.

Por ultimo, la cámara nos otorgara flujo óptico a nuestro sistema y estará conectada a través de un flex a la entrada MIPI de nuestra Raspberry Pi. Adicionalmente, esta tendrá una placa circular con iluminación activa para mejorar su interacción con el ambiente.

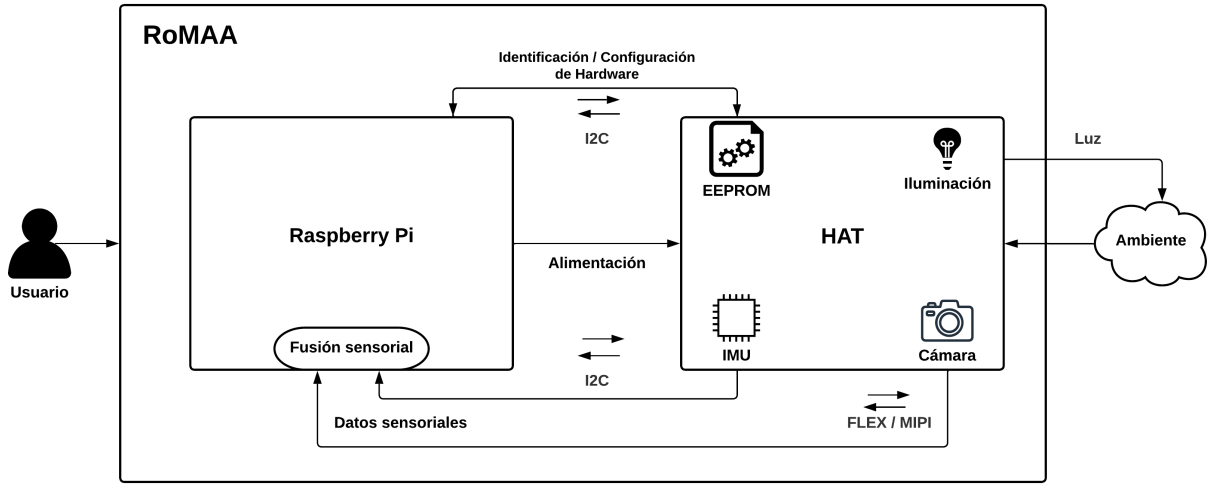


Figura 9: Diagrama en bloques del sistema

9. Desarrollo

9.1. Selección de sensores y memoria

Inicialmente, la principal tarea fue seleccionar los elementos principales y mas importantes de nuestro sistema. Donde acordamos con el grupo de trabajo que aparte de cumplir con los requerimientos técnicos también sean vigentes en el mercado y no estén obsoletos.

9.1.1. Cámara

La cámara seleccionada para el proyecto es una con sensor OmniVision OV9281 y del fabricante INNO-MAKER [8].

Figura 10: Cámara OV9281 INNO-MAKER

Esta cumple con creces los requerimientos técnicos que necesitamos. Sus principales características son:

- El producto cuenta con un flex totalmente compatible con Raspberry Pi.
- Modelos compatibles 4/3B+/3B/Zero/ZeroW/CM3/CM3+.
- Es un dispositivo compatible con V4L2 (Video For Linux 2).
- Cuenta con la posibilidad de accionar de manera externa el modo trigger, LED y modo flash strobe y también ganancia programable.
- Su sensor OV9281 es monocromático (blanco y negro), global shutter, CMOS, hasta 253 fps y de 1 MP.

- Tiene una gran variedad de modos de trabajo, 12 en total, donde podemos variar su salida, su resolución y su framerate. En la tabla 2 podemos observar todos sus modos disponibles.
- Dispone de la posibilidad de reemplazar el lente.

Modo	**Resolucion**	**Formato de salida**	**Framerate**
1	1280x800	Y10	120 fps
2	1280x800	Y8	144 fps
3	1280x800	Y10	EXT_TRIG
4	1280x800	Y8	EXT_TRIG
5	1280x720	Y10	120 fps
6	1280x720	Y8	144 fps
7	1280x720	Y10	EXT_TRIG
8	1280x720	Y8	EXT_TRIG
9	640x400	Y10	210 fps
10	640x400	Y8	253 fps
11	640x400	Y10	EXT_TRIG
12	640x400	Y8	EXT_TRIG

Cuadro 2: Modos de uso de cámara

Otros parámetros técnicos que podemos encontrar en esta cámara son:

- Resolución de sensor ADC: 8 Bit/ 10 Bit
- Salida CSI-2: RAW 8 o RAW10
- Cantidad de píxeles: 1280 x 800
- Tamaño de píxel: 3 (H) x 3 (V)um
- Velocidad CSI-2: 800 Mbps
- Modos de salida: Modo streaming y modo de trigger externo
- Resolución de shutter: 1 unidad horizontal
- Tamaño de plaqueta: 39mm x 39mm
- Conector: Cable flex 15 pines x 1 mm
- Agujeros de montaje: 4 x Diámetro 2.2 mm
- Agujeros de lente: 2 x Diámetro 2 mm

En la figura 11 podemos observar un diagrama en bloque del lente proporcionado por su fabricante OmniVision [9].

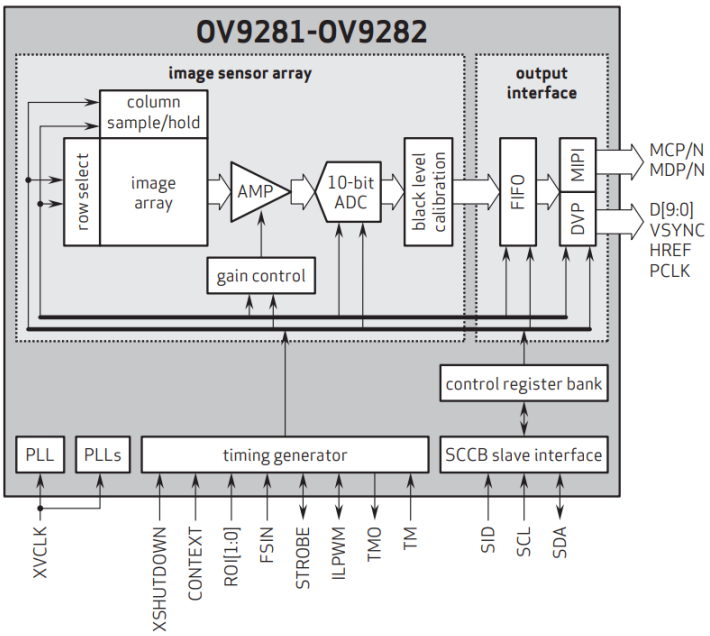


Figura 11: Diagrama en bloques del lente OmniVision OV9281

9.1.2. IMU: Unidad de medición inercial

La selección de la IMU fue un proceso largo y bastante complicado. Ya que aparte de cumplir con todos los requisitos técnicos decidimos ir por un modelo que sea vigente en el mercado y no quede obsoleto en un tiempo corto después del desarrollo del proyecto.

Las opciones que podemos encontrar en el mercado son tres familias: familia MPU, familia LSM y familia ICM.

La familia MPU cumple con los requisitos técnicos, son muy baratos y podemos encontrarlos fácilmente en mercados locales (Argentina, Córdoba). Algunos ejemplos de estos son el MPU6050 con acelerómetro y giróscopo y el MPU9250 con acelerómetro, giróscopo y magnetómetro (no utilizado en el proyecto). La principal desventaja que cuenta esta familia es que son obsoletos, es decir ya no se fabrican ni se continua mas su producción y esto es no es bueno para el proyecto ya que se piensa su uso para el desarrollo en el futuro.

La familia LSM cumple con creces los requisitos técnicos, son mas caros que la anterior familia y se encuentran muy pocos en mercados locales y limitados en stock. Algunos ejemplos de estos son LSM6DS3 con acelerómetro y giróscopo, LSM6DSL una versión renovada del anterior y LSM9DS0 con acelerómetro, giróscopo y magnetómetro. Las principales desventajas que encontramos es que solo algunos modelos estaban vigentes y que en el ámbito local no se podían encontrar fácilmente.

La familia ICM cuenta con las mejores prestaciones, son solo un poco mas caras que la anterior y no están disponibles en el mercado local. Algunos de estos son ICM42670 con acelerómetro

y giróscopo de gran rendimiento y ICM42605 con acelerómetro y giróscopo de un rendimiento premium. La principal desventaja es que debíamos hacer una compra internacional para adquirirlos y tal vez poner en pausa el proyecto hasta que se haga su llegada.

Finalmente, entre tanta búsqueda nos encontramos que había un modulo en Argentina que contenía la IMU LSM6DS3 la cual cumplía con todo lo solicitado. Decidimos comprarlo ya que tenia un adquisición muy rápida y podíamos trabajar con el inmediatamente sin tener que pausar el proyecto. Nuestro plan A es trabajar con este modulo mientras compramos en el exterior el mismo sensor IMU para luego de desarrollar un circuito similar en nuestro HAT soldarlo en este. Pero también debíamos pensar en planes de contingencia por si no conseguimos una IMU similar. Entonces el plan B consta en desoldar la IMU del modulo y volver a soldarla en nuestro HAT, algo muy difícil de realizar ya que se trata de un encapsulado LGA y de un tamaño diminuto (3mm x 2.5 mm x 0.83 mm), esto seria muy riesgoso de hacer y podría quemar el componente. El plan C es directamente soldar el modulo sobre el HAT desarrollado haciendo un especie de "Frankenstein" entre las dos placas. Obviamente no es lo que buscamos en un desarrollo profesional e integro.

El modulo que compramos en Argentina se trata del modulo SparkFun Breakout LSM6DS3 que lo podemos observar en la figura 12. Esta pequeña placa tiene muchos usos en la robótica como por ejemplo para detectar golpes, inclinación, movimiento, toques, cuenta pasos y también medir temperatura. Su acelerómetro es capaz de leer hasta 6.7 kS/s y el giroscopio hasta 1.7 kS/s para una detección de movimiento muy precisa. Cuenta con un buffer de hasta 8 kB de datos, esto hace posible que el modulo pueda almacenar datos entre lecturas, albergar otros sensores y controlar pines de interrupción.

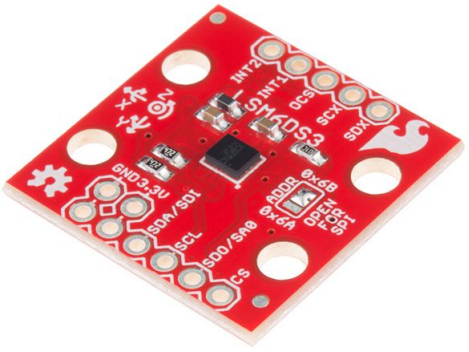


Figura 12: Sparkfun Breakout LSM6DS3

El protocolo de comunicación que tiene con nuestra Raspberry Pi es I2C y podemos ver sus pines asociados en la placa (SCL y SDA). La alimentación que requiere es de 3.3 V lo cual facilita realizarla desde nuestra propia Raspberry Pi por que tiene una salida exactamente del voltaje que necesitamos.

Características principales:

- Consumo de energía: 0.9 mA en modo normal y 1.25 mA en modo alto rendimiento.
- Cuenta con una smart FIFO de 8 kByte.
- Rango de medición: 2/4/8/16 g y 125/245/500/1000/2000 dps
- Alimentación: 1.71 V a 3.6 V
- Interfaz serial SPI/I2C con función de sincronización de datos del procesador principal.
- Sensor de temperatura integrado

9.1.3. Memoria EEPROM

Con la elección de la memoria EEPROM fue muy simple, ya que la propia Raspberry Pi nos recomienda en su repositorio de HATs [4] la memoria 24C32 y que por suerte pudimos adquirirla en el mercado local de la ciudad de Córdoba.

Precisamente la memoria EEPROM AT24C32AN de Atmel que podemos ver en la figura 13. Proporciona 32.768 bits de memoria serial de lectura programable y borrable organizada en 4096 palabras de 8 bits cada una. Cuenta con una función de conexión en cascada hasta 8 dispositivos que comparten un bus de datos en común en dos hilos. El dispositivo esta optimizado para aplicaciones automotrices pero servirá estupendamente para nuestro proyecto. Su protocolo de comunicación es I2C. Tiene un pequeño tamaño de 4.8 mm x 3.81 mm x 1.35 mm.

Figura 13: Memoria EEPROM AT24C32AN de Atmel

En la figura 14 observamos su diagrama en bloques con su funcionamiento y el comportamiento de cada uno de los pines.

Características:

- Voltaje estándar de operación: 2.7 V.
- Organización interna de 4096 palabras de 8 bits cada una.
- Interfaz serial Two-wire.

- Schmitt trigger y entradas de filtro para supresión de ruido.
- Protocolo de transferencia de datos bidireccional.
- Velocidad de clock de 400 KHz.
- Pin de protección de escritura.
- 32 byte de escritura por pagina.
- Ciclo de escritura de tiempo propio (5 ms max.)
- Alta fiabilidad:
  - Resistencia: 1 millón de ciclos de escritura.
  - Retención de datos: 100 años.
- Libre de plomo y halógenos.
- Paquete 8 patas SOIC.

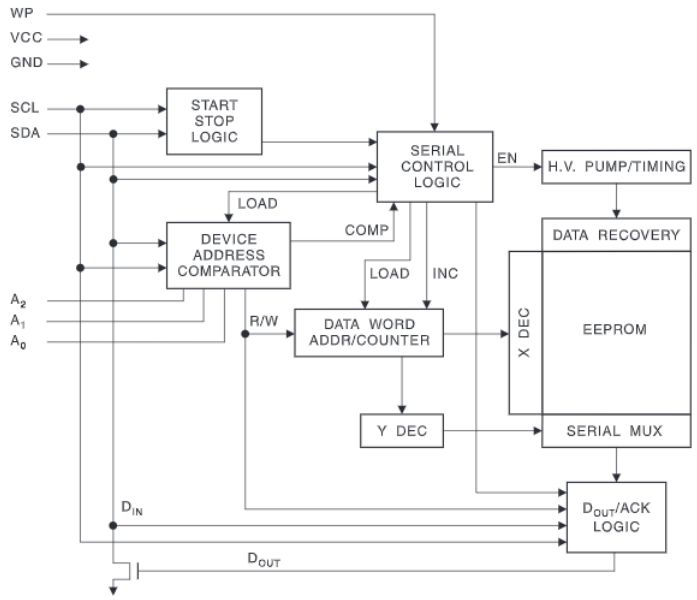


Figura 14: Diagrama en bloques de AT24C32AN de Atmel

9.2. Instalación de sistema operativo

La principal virtud que debe tener el sistema operativo a instalar es que se pueda instalar ROS fácilmente en este. Por eso seleccionamos un sistema operativo basado en Ubuntu para su simple uso y sencilla instalación de ROS.

9.2.1. Ubuntu MATE

El sistema elegido es Ubuntu MATE, una distribución Linux producida por desarrolladores Argentinos que destaca por su interfaz muy personalizable, su simplicidad y poco consumo de recursos. En la figura 15 explayamos su logo.

Ubuntu MATE nace del cambio de interfaz que tiene Ubuntu en el año 2010, dejando atrás GNOME 2 esto causo mucha molestia en algunos usuario ya acostumbrados a la interfaz gráfica. Uno de estos usuarios era un argentino llamado Germán Perugorría que dio inicio al proyecto de una alternativa a la nueva interfaz de Ubuntu creando un entorno llamado MATE (por la famosa infusión de Argentina) y años después Martin Wimpress y Alan Pope crean la distribución Ubuntu MATE con este famoso entorno. A los pocos años fue aceptada como distribución oficial de Linux.

Figura 15: Logo Ubuntu MATE

Su instalación es bastante sencilla. Buscando en su pagina oficial Ubuntu MATE podemos encontrarnos con diferentes versiones para nuestro modelo (Raspberry Pi modelo 3 B/B+). Lo primero a elegir es si la arquitectura del sistema es de 32 bits o 64 bits. Por descarte, elegimos una arquitectura de 64 bits ya que el procesador de nuestra sistema lo soporta y su versión 20.04, pero resulto desastroso ya que la memoria RAM quedaba muy limitada y ante un simple proceso se llenaba dejando el sistema inutilizable. Por esto decidimos instalar un sistema operativo que sea de arquitectura de 32 bits y nos decantamos por algo intermedio en cuanto a su versión como lo es Ubuntu MATE 18.04. Vale la pena aclarar que a partir de la versión 22.04, ROS no es compatible.

El sistema operativo funciono de maravilla y pudimos trabajar en el eficazmente hasta que nos dimos con conflictos en la cámara que en la sección 9.4.1 explayaremos con mas profundidad.

Por eso decidimos trabajar con Raspberry OS que es el sistema operativo que viene por defecto en los computadores Raspberry Pi.

9.2.2. Raspberry Pi OS

Es el sistema operativo oficial de los sistemas Raspberry Pi y está mantenido por los propios desarrolladores de estos computadores SBC. Inicialmente recibía la denominación de Raspbian, al estar basado en la distribución de Linux, Debian. Actualmente aún se le denomina de esta manera.

Se ha optimizado esta distribución para funcionar en equipos que se basan en procesadores de arquitectura ARM. Raspberry Pi OS incluye diferentes paquetes y programas de manera nativa. Algo interesante es que como interfaz gráfica utiliza PIXEL (Pi Improved X-Window Enviornment Lightweight). Está una adaptación de la interfaz sencilla y ligera LXDE.

Para su instalación montamos la tarjeta microSD a una PC y por medio del programa Raspberry Pi Imager seleccionamos la distribución elegida y la escribimos en la tarjeta ya lista para iniciar. Aparte de la inmensa diversidad de imágenes ya propias del programa tenemos la posibilidad de montar cualquier distribución con la imagen descargada en nuestra PC. Desde la web Raspberry Pi Images podemos descargar distribuciones Raspberry Pi OS en su versión "Legacy", es decir, mas antiguas. Esto nos sera de utilidad a la hora de elegir un sistema operativo compatible con nuestra cámara.

El sistema operativo es estupendamente rápido y cuenta con todo lo necesario para nuestro proyecto, sin embargo, el principal problema es que no se puede instalar ROS tan fácilmente en el. Por eso usaremos una alternativa: Docker. En la sección 9.9.1 explicamos como instalar y usar ROS en nuestro sistema con Raspberry Pi OS.

9.3. Iluminación activa

Se desarrollo una placa circular de iluminación que va montada a la cámara, esta ayudara a mejorar las imágenes captadas por la cámara en entornos mas complejos. Para su diseño utilizaremos KiCad 6.0 (versión mas actualizada en la fecha) donde elaboramos tanto su esquemático como su PCB para luego enviar a elaborar a alguna empresa de circuitos impresos.

9.3.1. Desarrollo esquemático y PCB

Después de los cálculos eléctricos se elaboro un esquemático de lo que va a ser la iluminación que ayudara a la cámara. El esquematico se cargo a KiCad como podemos ver en la figura 16 donde podemos apreciar los 12 LEDs con sus resistencias para adecuar la corriente suministrada. También, podemos observar un MOSFET junto a otra resistencia para comandar el encendido/apagado de los LEDs por medio de un Pin de señal. Adicionalmente agregamos los conectores de +5V, Pin de señal y GND que van a estar conectados a nuestra placa HAT. Se agrega los 4 agujeros de montaje a la cámara.

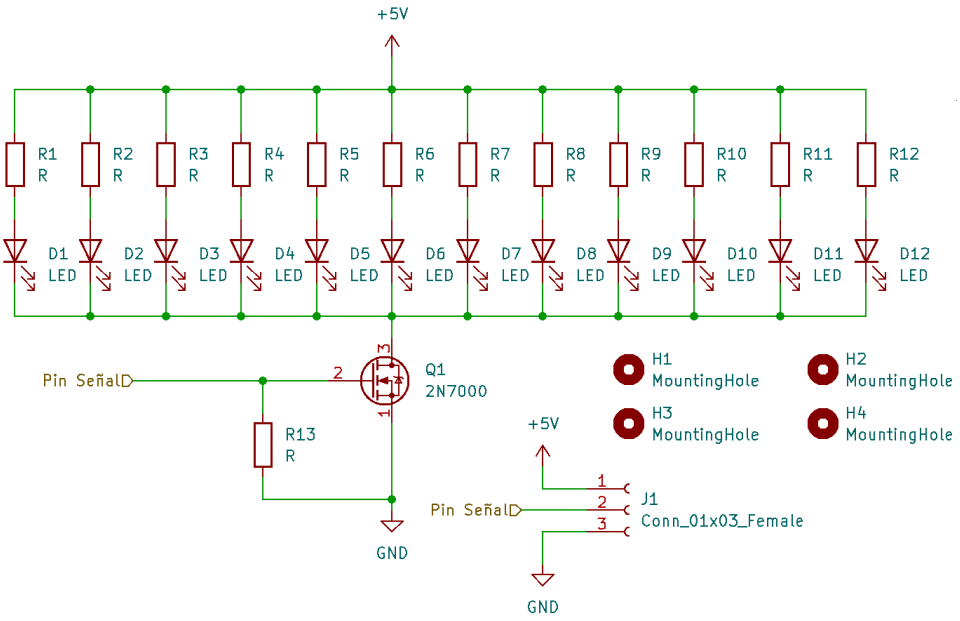
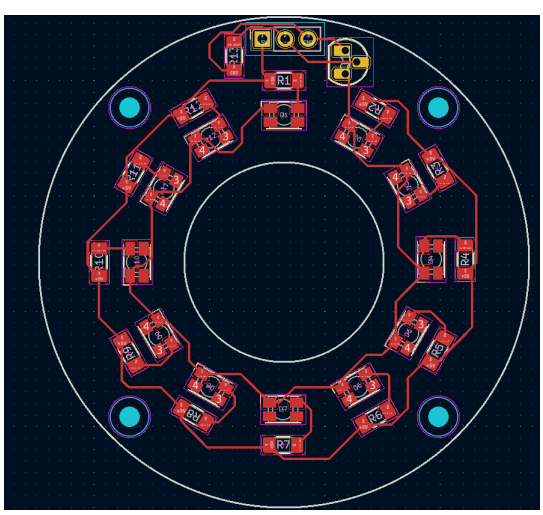


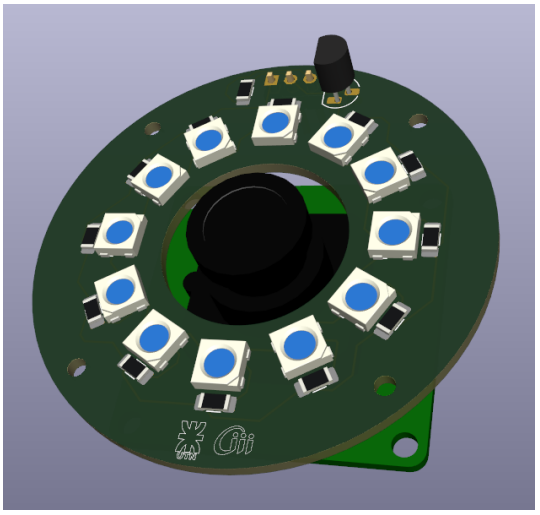
Figura 16: Esquemático iluminación activa

Con lo respectivo a su PCB se asignaron los footprints a cada componente correspondiente a los comprados que veremos en la sección 9.6. Se calculo las dos circunferencias que conforman el contorno de la placa teniendo en cuenta al tamaño de la cámara. También, la coincidencia de agujeros como así también su tamaño.

En la figura 17 se puede apreciar el PCB ya ruteado (a) y su modelo 3D junto a la cámara (b) para hacer una estimación de como quedaría una vez elaborado.



(a) PCB de placa



(b) Modelo 3D junto a cámara

Figura 17: Placa de iluminación activa para cámara OV9281

9.3.2. Tipo de iluminación

Primeramente solo habíamos pensando un tipo de iluminación: luz blanca. Pero luego de charlas con el equipo decidimos hacer dos placas de iluminación: una que contenga luz blanca convencional y otra muy similar pero con LEDs infrarrojos. Esto nos dará mas diversidad en las aplicaciones que se podrán hacer con el proyecto desarrollado.

9.4. Prototipo del sistema

La primer gran tarea que tuvo el equipo es realizar un prototipo de la placa a desarrollar y probar las distintas secciones que funcionen correctamente como conjunto e individualmente.

9.4.1. Cámara

Sin duda alguna, esta fue la parte del proyecto mas conflictiva que tuvimos. La instalación física de la cámara es muy sencilla ya que simplemente va conectada por medio de un flex al puerto MIPI-camara de la Raspberry Pi como vemos en la figura 18. El problema principal estuvo en los drivers de la cámara.

Para empezar, el proyecto arranco con Ubuntu MATE instalado y sin poder detectar la cámara de manera nativa. El fabricante INNO-MAKER cuenta con Drivers oficiales de la cámara pero nos encontramos con un obstáculo que fue que el link del repositorio a donde estaban los archivos del driver no estaba disponible (fue borrado). Por esto decidimos contactar con el fabricante solicitando un repositorio actual. A la semana tuvimos una respuesta con un nuevo link del repositorio Repositorio OV9281 INNO-MAKER. Procedemos instalación del driver por medio de un script, **autoinstall_driver.sh**. Sin embargo, fallo acusando un error de compatibilidad con el Kernel del sistema operativo instalado.

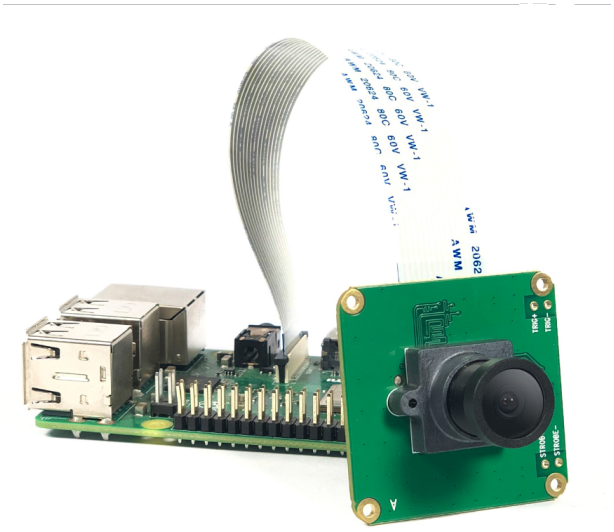


Figura 18: Raspberry Pi y Cámara OV9281

El driver esta preparado solo para Kernels propios de Raspberry Pi OS y con este no tenemos la posibilidad de instalar ROS nativamente. Decidimos cambiar el rumbo y probar otras posibilidades como drivers ArduCAM, drivers Libcamera, instalación de drivers manualmente, etc.; Lamentablemente ninguna de estas opciones funciono.

Luego, se intento cambiar el kernel de nuestro sistema operativo por medio de la siguiente web Change default kernel la cual nos dio resultados poco prometedores.

Tomamos la decisión de instalar un sistema operativo Raspberry Pi OS en su versión Legacy que contenga un kernel compatible con los drivers de la cámara. Esto funciono a la perfección con los drivers del fabricante la cámara fue detectada inmediatamente. En resumen, con Raspberry Pi OS funciona la cámara pero no podemos instalar ROS y con otro sistema operativo podemos instalar ROS y no detecta la cámara. Continuamos con Raspberry Pi OS con una nueva alternativa: Docker. Este es un contenedor en donde podemos instalar y correr una gran variedad de aplicaciones y conectarla con nuestro sistema operativo original. Entonces el plan que seguimos es tener Raspberry Pi OS con la cámara e instalar ROS en un contenedor Docker.

9.4.2. IMU

Para conectar la unidad de medición inercial a la Raspberry Pi solo necesitamos el pin de GND a un pin GND de la GPIO (pin 6 por ejemplo), el pin de alimentación a un pin 3.3 V de la GPIO (pin 1 por ejemplo) y los pines de comunicación SDA y SCL al SDA y SCL de la GPIO (pin 3 y 5 respectivamente) podemos comprobar los pines en la sección 5.1.3.

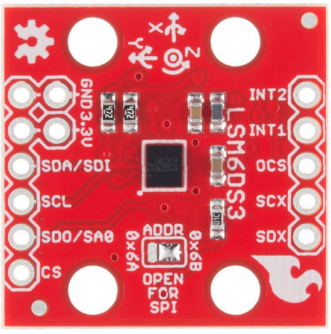


Figura 19: Sparkfun Breakout LSM6DS3

Luego de realizar la conexión al dispositivo comprobamos que se haya detectado este mismo en nuestro sistema operativo a través de un paquete de herramientas del sistema llamado “i2c-tools”. Por medio del comando “i2cdetect” vemos que la comunicación entre el modulo y la Raspberry Pi es correcta y en la dirección en donde se encuentra.

Para poner a prueba nuestro modulo desarrollamos un script en Python que toma los datos crudos extraídos por el sensor y los convierte a unidades razonables de aceleración, ángulos y giróscopo. En la figura 20 se observa el modulo instalado con la salida del script con los resultados arrojados.

Figura 20: Sparkfun Breakout LSM6DS3 funcionando con el script desarrollado

9.4.3. Memoria EEPROM

La construcción del modulo destinado a la memoria EEPROM se llevo a cabo con la ayuda del paquete de software libre KiCad, en el cual se plasmo el circuito esquemático recomendado por el GitHub perteneciente a Raspberry Pi en el cual especifica, ademas, las condiciones de diseño para una placa HAT. Posteriormente se realizo la ubicación de componentes y el ruteado de pistas. El método empleado para la fabricación del modulo consiste en la transferencia térmica del toner del circuito impreso en papel couche, a la placa de cobre para finalmente pasarlo por ácido férrico. Una vez finalizada dicho proceso, se realizaron las perforaciones pertinentes para los componentes de tipo THT y se sueldan. Para la verificación del modulo se realizo un testeo funcional que consiste en crear un archivo de texto, que contenga la información básica que describa el contenido de la placa HAT en la cual sera incluida dicha memoria, convertirlo en un archivo binario, grabarlo en la memoria mediante los pines de conexión específicos para dicha función, para luego leer el contenido de la memoria y así obtener nuevamente el archivo de texto creado en primera instancia.

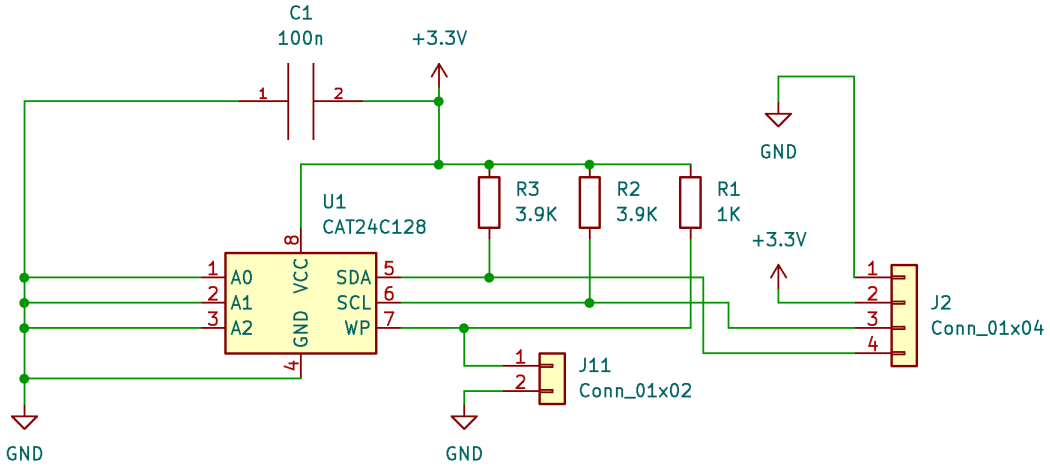


Figura 21: Esquemático memoria EEPROM

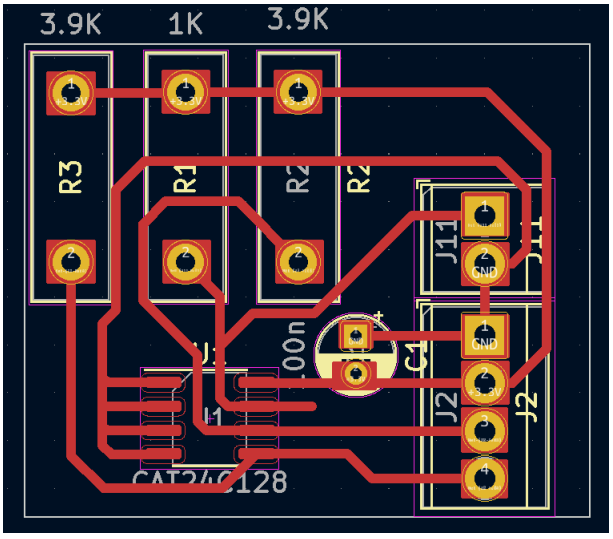
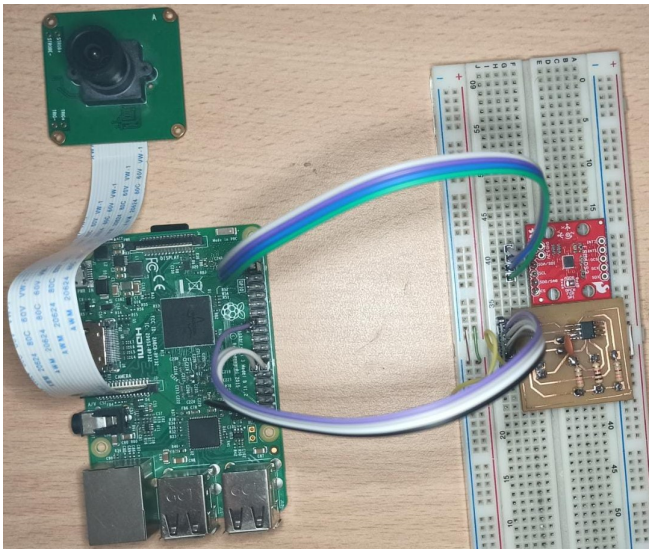


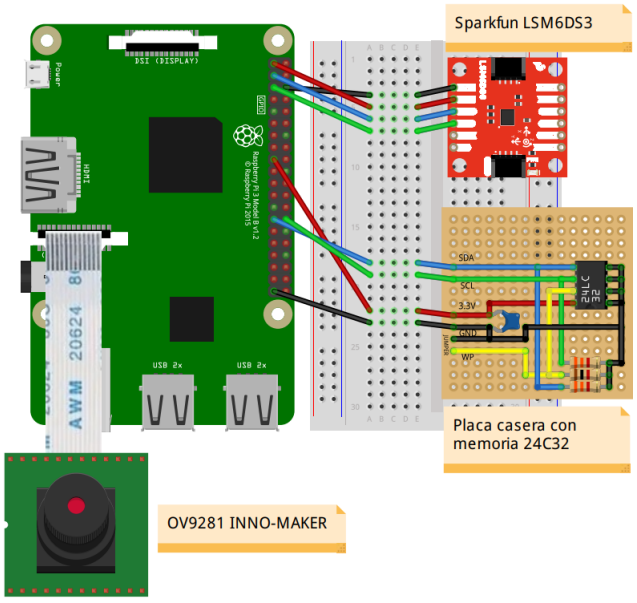
Figura 22: PCB memoria EEPROM

9.5. Prototipo completo

Para finalizar, una vez comprobado que cada parte del sistema funciona lo probamos como conjunto realizando la conexión simultáneamente de todos estos como podemos ver en la figura 23 donde vemos el prototipo completamente conectado y funcional (a) y una simulación de las conexiones con mas detalle en el programa Fritzing (b).



(a) Foto de prototipo del sistema



(b) Conexiones de prototipo

Figura 23: Prototipo completo del sistema funcionando en simultaneo

9.6. Compra de componentes

Mediante la plataforma digital de la distribuidora de componentes DigiKey se realizaron principalmente la selección de los distintos encapsulados de los componentes seleccionados y la posterior compra, haciendo hincapié en las dimensiones de los termínales de conexión para la fabricación de la placa. La compra se realizo con este distribuidor internacional debido a la escasez en el mercado nacional de determinados componentes fundamentales para el proyecto.

9.7. Esquemático y PCB

El diseño de la placa se realizo respetando las especificaciones mecánicas del proyecto, utilizando componentes del tipo SMD ubicados en la parte superior del HAT. Como podemos ver en la figura 24 los pines de conexión empleados para la comunicación entre Raspberry Pi y el sensor de medición inercial son GPIO(3) y GPIO(4) correspondientes a la comunicación I2C, mientras que la comunicación con la memoria EEPROM se usan los pines GPIO(0) y GPIO(1) propios para la identificación de HAT que utiliza mismo protocolo de comunicación I2C.

Ambos dispositivos están alimentados con 3.3 V, a diferencia de la placa auxiliar para iluminación en la cual se conecta a 5 V. El manejo de la iluminación se controla a partir del pin GPIO(22).

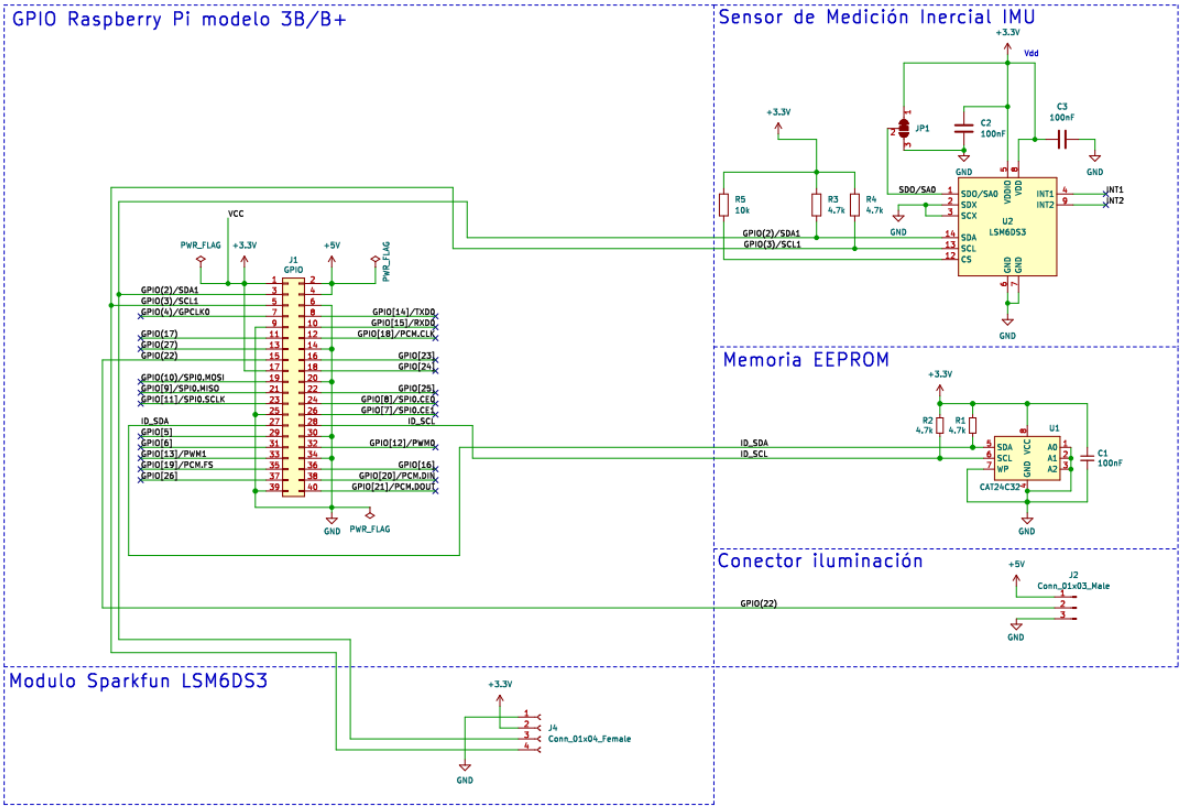
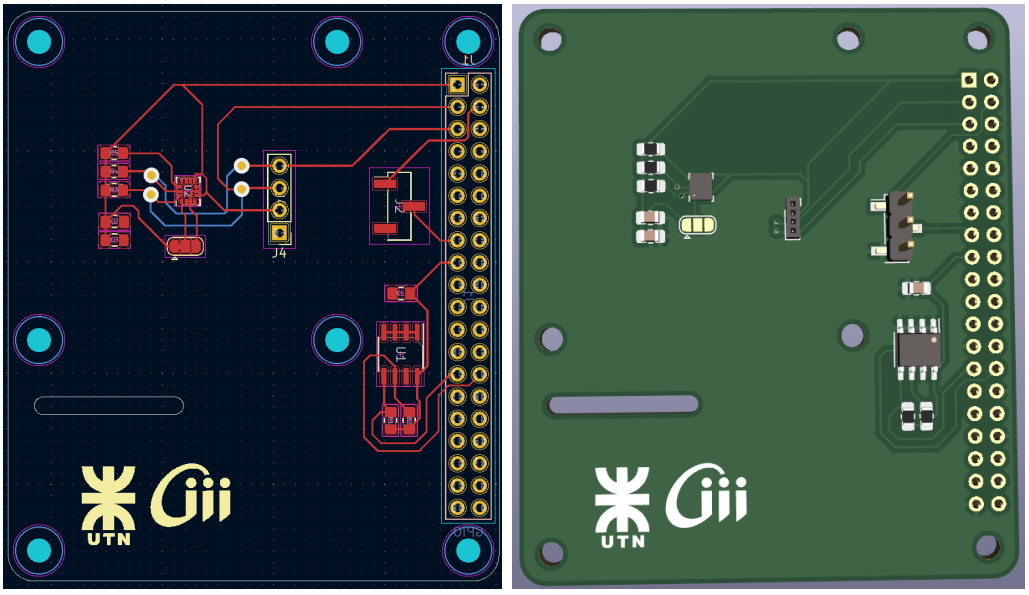


Figura 24: Esquemático HAT

En la figura 25 se pueden observar (a) el diseño PCB y (b) su modelo 3D. El mayor reto fue replicar las especificaciones mecánicas dadas por Raspberry Pi HATs dadas en la sección 6.2 la cual cuenta con medidas muy precisas. Decidimos eliminar la abertura dispuesta para el display y solo dejar la abertura para la cámara que vamos a utilizar. También se añadieron agujeros con las dimensiones de la cámara para que, si se desea, se pueda atornillar a la plaqueta esta misma. Por ultimo, a modo de prueba añadimos un modelo 3D de Raspberry Pi 3 B/B+ para

comprobar que mecánicamente todo coincida logrando resultados muy satisfactorios. Esto se puede ver en la figura 26.



(a) PCB de HAT (b) Modelo 3D de HAT

Figura 25: Prototipo completo del sistema funcionando en simultaneo

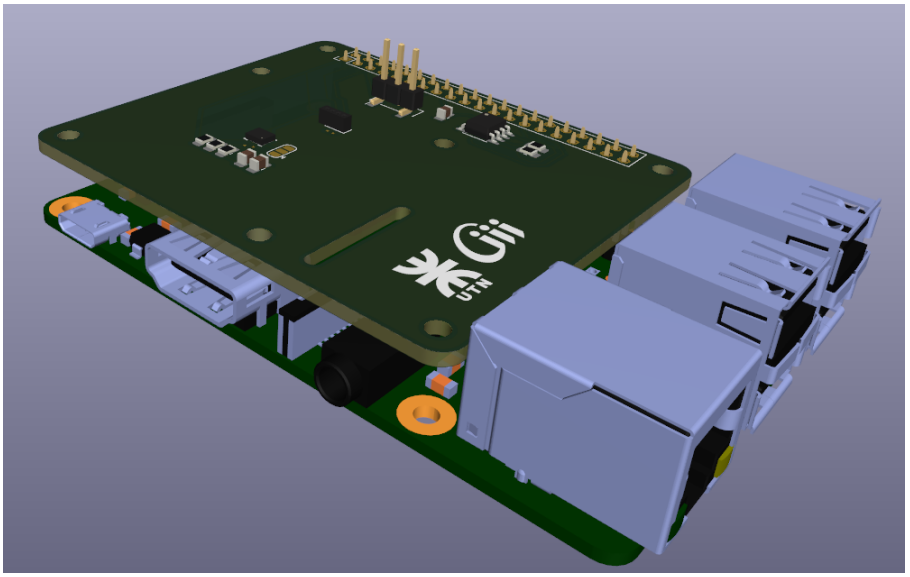
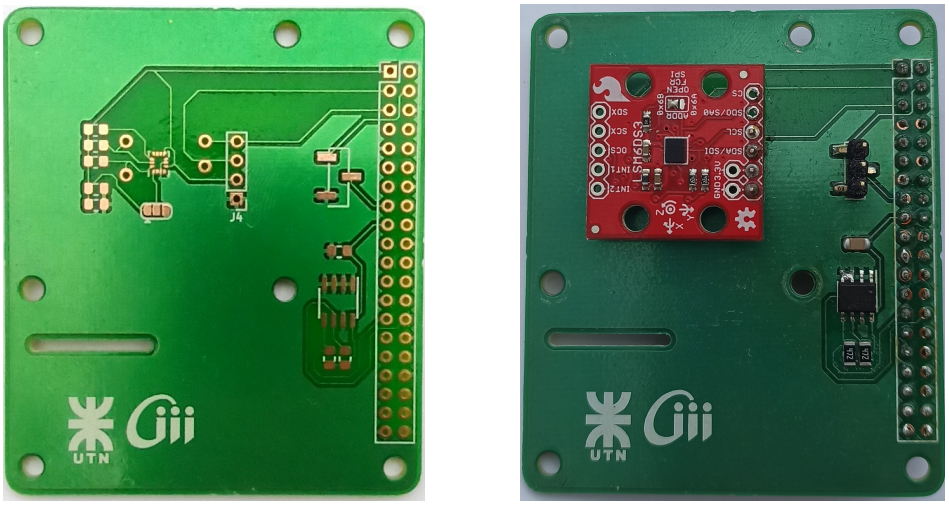


Figura 26: Modelo 3D de HAT conectado a Raspberry Pi 3 B/B+

9.8. Hardware del sistema

Para el proceso de fabricación se mando a elaborar en una empresa local de Córdoba llamada CITEM. Se elaboraron las diferentes placas con un acabado profesional. Podemos apreciar en la figura 27(a) un primer vistazo de nuestro HAT.

A continuación de esta etapa se realizo el soldado de nuestra plaqueta y todos los ensayos de prueba pertinentes respecto al Hardware. En la figura 27(b) ilustramos el HAT ya soldado esperando a ser puesto a prueba en la computadora Raspberry Pi.



(a) (b)

Figura 27: (a) PCB de HAT recién fabricado. (b) HAT soldado.

Por ultimo, se ensamblo mecánicamente y eléctricamente cada una de las partes para obtener el sistema completo listo para ser puesto en marcha. En la figura 28 exponemos el sistema ya completo con todas sus partes ancladas (cabe destacar que por cuestiones estéticas se obvio el cable flex de la cámara).

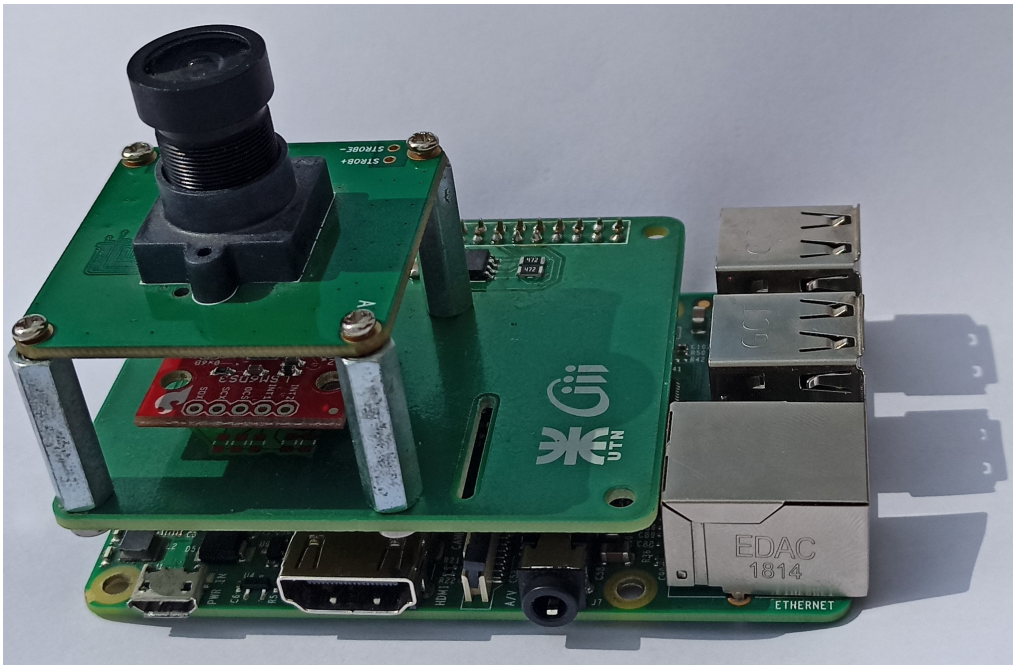


Figura 28: Hardware del proyecto completo

9.9. ROS

El sistema operativo de robot (ROS)[10] es un conjunto de bibliotecas de software y herramientas que lo ayudan a crear aplicaciones de robot. Desde controladores hasta algoritmos de última generación y potentes herramientas de desarrollo.

ROS es un meta sistema operativo de código abierto para tu robot. Provee de servicios que se esperarían de un sistema operativo, incluyendo abstracción de hardware, control de dispositivos de bajo nivel, implementación de funcionalidades comunes, pasaje de mensaje entre procesos y manejo de paquetes. También brinda herramientas y librerías para obtener, construir, escribir y correr código a través y mediante varias computadoras.

Figura 29: Logo ROS

El objetivo primario de ROS es soportar la reutilización de código en la investigación y desarrollo dentro de la robótica. ROS es una estructura distribuida de procesos llamados Nodos que permite el diseño individualizado: pero fácilmente acoplable a los demás procesos. Estos procesos

se pueden agrupar en Paquetes y Pilas, que fácilmente pueden ser intercambiados, compartidos y distribuidos. En apoyo de este objetivo principal de compartir y colaborar; hay varios otros objetivos dentro del marco ROS:

- Liviano: ROS está diseñado para ser lo más liviano posible, de modo que el código escrito para ROS se pueda usar con otros frameworks o estructuras de desarrollo de software para robots.
- Bibliotecas agnósticas: el modelo de desarrollo preferido es escribir bibliotecas agnósticas en ROS con interfaces funcionales limpias.
- Independencia del lenguaje: ROS es fácil de implementar en cualquier lenguaje de programación moderno como Python y C++.
- Pruebas sencillas: ROS tiene un marco de prueba de integración / unidad incorporado llamado rostest que facilita la activación y desactivación de dispositivos de prueba.
- Escalado: ROS es apropiado para grandes sistemas de rutinas de ejecución y para grandes procesos de desarrollo.

El software para ROS se prueba principalmente en sistemas Ubuntu y Mac OS X, aunque la comunidad ROS ha estado contribuyendo con soporte para Fedora, Gentoo, Arch Linux y otras plataformas Linux.

9.9.1. Instalación

La instalación de ROS es sencilla en sistemas operativos Ubuntu. Sin embargo, el sistema operativo elegido para que nuestros componentes sean todos compatibles es Raspberry Pi OS y se complica inmensamente la instalación en este. Para solucionar este problema decidimos utilizar un contenedor Docker.

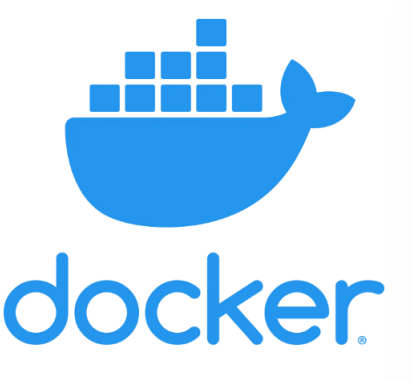


Figura 30: Logo Docker

Un contenedor es una unidad de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable de un entorno informático a otro. Un contenedor de Docker es un paquete de software ligero, independiente y ejecutable

que incluye todo lo necesario para ejecutar una aplicación: código, runtime, herramientas del sistema, bibliotecas del sistema y configuraciones. Los contenedores aíslan el software de su entorno y garantizan que funcione pese a las diferencias entre sistema operativo host y el contenedor. [11]

Docker nos facilito en una enorme medida la instalación de ROS en nuestro sistema ya que por medio de este solo con el comando en terminal "docker run ros:noetic" podemos ejecutar ROS (versión 1) en el contenedor.

9.9.2. Creación de paquete

Para demostrar los requisitos técnicos del proyecto se realizo un paquete en ROS con varios nodos asociados a la IMU y Cámara. Se hizo un archivo launch para lanzar los scripts asociados. Estos scripts lanzan varios gráficos del estilo plot donde se pueden ver los datos crudos tomados por la IMU y una pequeña ventana en donde podemos apreciar la imagen tomada por la cámara. Esto se puede observar en la figura 31 extraida de un video casero hecho como demostración.

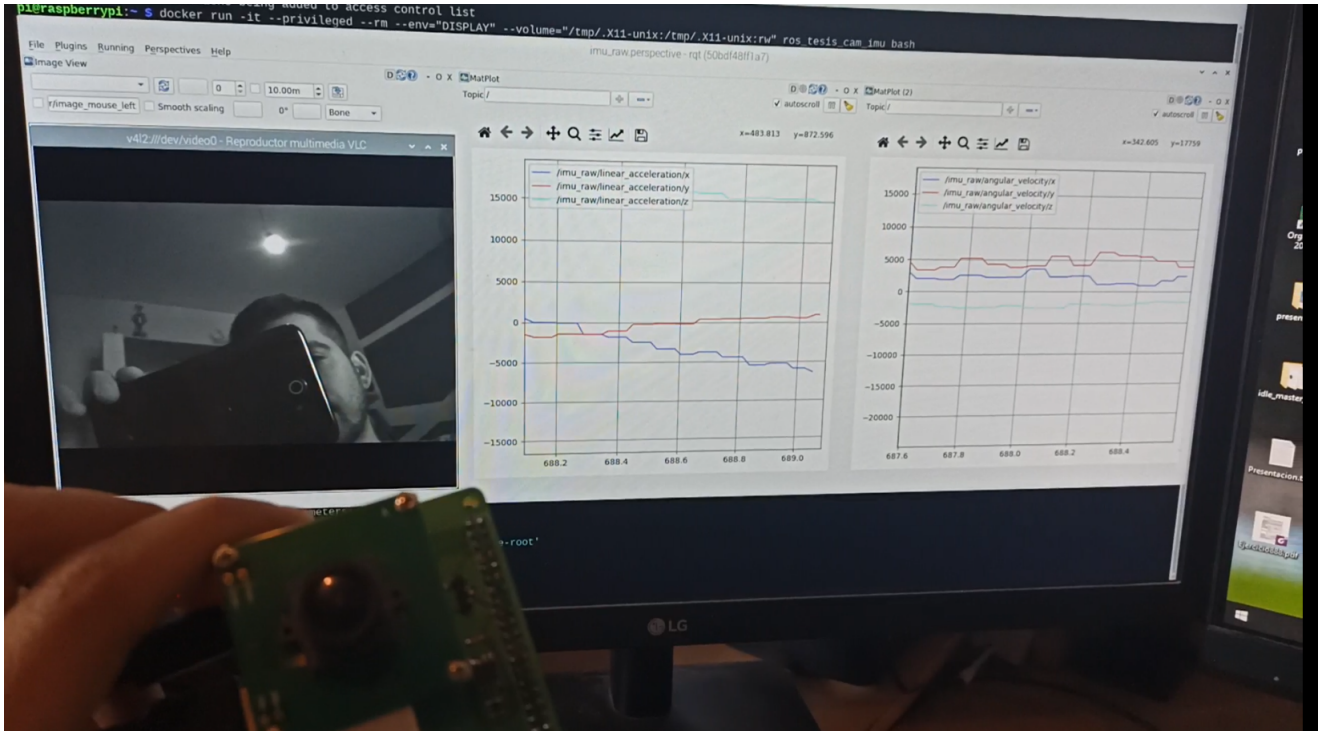


Figura 31: Demostración del sistema funcionando en ROS

10. Conclusión

Para finalizar el presente informe queremos mencionar y remarcar los objetivos planteados inicialmente como la utilización de componentes vigentes, realizar la primera interacción en el diseño de hardware para futuros desarrollos de software, que el diseño de la placa cumpla los estándares impuestos de los fabricantes de Raspberry Pi, se verificó que el funcionamiento de nuestro proyecto es el esperado, para el cuál fue diseñado, el testeo fue realizado mediante la plataforma de prueba estipulado. Dichos objetivos fueron cumplido respetando los plazos acordados y a los que se sumaron la fabricación de la placa PCB de manera profesional con la contratación, luego de solicitar múltiples presupuestos, de una empresa especializada en el tema de la ciudad de Córdoba. Elevando la calidad y presentación del proyecto a un nivel similar al de un producto comercial.

Referencias

[1] Navegación Autónoma de Robot Móvil en Ambientes Parcialmente Estructurados utilizando Visión Artificial, Gonzalo Pérez Paina.

[2] Review Raspberry Pi.

[3] Raspberry Pi Wiki.

[4] GPIO Raspberry Pi.

[5] Repositorio Github Raspberry Pi HATs.

[6] Datasheet CAT24C32.

[7] Proyecto de fusión sensorial en robot Jackal.

[8] OV9281 INNO-MAKER.

[9] OmniVision OV9281.

[10] ROS Web oficial.

[11] Contenedores Docker.