Universidad Tecnológica Nacional

Proyecto Final

Implementación de sistema de control para la operación de vehículo terrestre no tripulado

---

Autor:

- Jesús Exequiel Benavídez

Director:

- Enrique Sergio Burgos

Proyecto final presentado para cumplimentar los requisitos académicos para acceder al título de Ingeniero en Electrónica

en la

Facultad Regional Paraná

Fecha : Marzo de 2019

Declaración de autoría:

Yo declaro que el Proyecto Final “Implementación de sistema de control para la operación de vehículo terrestre no tripulado” y el trabajo realizado son propio. Declaro:

- Este trabajo fue realizado en su totalidad, o principalmente, para acceder al título de grado de Ingeniero en Electrónica, en la Universidad Tecnológica Nacional, Regional Paraná.
- Se establece claramente que el desarrollo realizado y el informe que lo acompaña no han sido previamente utilizados para acceder a otro título de grado o pre-grado.
- Siempre que se ha utilizado trabajo de otros autores, el mismo ha sido correctamente citado. El resto del trabajo es de autoría propia.
- Se ha indicado y agradecido correctamente a todos aquellos que han colaborado con el presente trabajo.
- Cuando el trabajo forma parte de un trabajo de mayores dimensiones donde han participado otras personas, se ha indicado claramente el alcance del trabajo realizado.

Firmas:

-

Fecha:

Agradecimientos:

Me gustaría comenzar agradeciendo a mis padres, que me brindaron la posibilidad de estudiar sin tener que preocuparme por ninguna otra cosa. A mis amigos, novia y compañeros, por el apoyo y la ayuda incondicional de siempre. Gracias también a la UTN y a todos los docentes de la carrera.

Jesús Exequiel Benavídez

Universidad Tecnológica Nacional

Abstract

Facultad Regional Paraná

Ingeniero en Electrónica

Implementación de sistema de control para la operación de vehículo terrestre no tripulado

Jesús Exequiel Benavídez

Abstract:

In the present work we designed and built a kit for navigation software development, this kit consists of a vehicle in where all electronic and low level programming was resolved. It was used, a scaled vehicle controlled by a microcontroller TM4C1233PH6 in its low level, with the module NEO-6 like GPS device, the LSM9DS1 like Unit of Inertial Measurement, a Model Predictive Control was applied for the vehicle's speed and an algorithm of data fusion. All this was communicated with a Raspberry Pi where the navigation algorithm will be implemented.

We obtained a vehicle that meets the objectives set, allows navigation algorithm development on its Raspberry Pi platform, controlling the low level thanks to a Python package that performs the abstraction. The kit was obtained for a lower value than otder similar kits on the market.

Keywords:

Autonomous vehicle control, Mobile Robotic, Navigaton algorithm, Predictive Control.

Resumen:

En el presente trabajo se diseñó y construyó un kit para desarrollo de software de navegación, este kit consiste en un vehículo al que se le resolvió toda la electrónica y la programación de bajo nivel. Se utilizó en él, un vehículo a escala controlado en el bajo nivel por un microcontrolador TM4C1233PH6, con el módulo NEO-6 como dispositivo GPS, el LSM9DS1 como Unidad de Medición Inercial, se aplicó un Modelo de Control Predictivo para el control de la velocidad del vehículo y un algoritmo de fusión de datos. Esto se comunicó con una Raspberry Pi donde se implementarán los algoritmos de navegación.
Se obtuvo un vehículo que cumple con los objetivos planteados, permite diseñar algoritmos de navegación en su plataforma Raspberry Pi, controlando el bajo nivel gracias a un paquete en lenguaje Python que realiza la abstracción. El kit se consiguió por un valor menor que otros kits similares en el mercado.

Palabras Clave:

Algoritmo de navegación, control de vehículo autónomo, control predictivo, robótica móvil.

Reconocimientos:

Quiero comenzar reconociendo a Sergio cuya ayuda, orientación, observaciones y consejos fueron fundamentales. A Leonardo por sus clases de control predictivo y por su constante guía indicándome los pasos que debía tomar. A Guido, porque cada vez que me veía se arremangaba y me daba una mano en lo que fuera. Al Centro de Investigación en Señales, Sistemas e Inteligencia Computacional Sinc(i), que me brindaron todos los materiales y me prestaron sus instrumentos y herramientas para llevar adelante el proyecto. Por último, pero no menos importante, quiero reconocer a Félix, siempre dispuesto a ayudar ante mis consultas.

Índice:

Lista de Figuras:

Lista de Tablas

Lista de Abreviaciones

- A: Amper.
- BEC: Battery Eliminato Circuit.
- CAN: Controler Area Network.
- C: Coulomb.
- cm: Centímetros.
- CRC: Código de Redundancia Cíclica.
- DMA: Direct Memory Access
- EEPROM: Electrical Erase Programable Read Only Memory.
- ESC: Electronic Speed Controler.
- g: Gravedad (unidad de aceleración).
- GPS: Sistema de posicionamiento global.
- gr: Gramos.
- Hz: Hercio.
- I2C: Inter-Integrated Circuit.
- IMU: Unidad de medición inercial.
- kg: Kilogramo.
- mAh: Miliamper hora.
- mm: Milímetros.
- MPC: Model Predictive Control.
- ms: Milisegundos.
- NMEA: National Marine Electronics Association.
- PWM: Pulse With Modulation.
- RPM: Revoluciones Por Minuto.
- Servo: Servoactuador.
- SPI: Serial Peripheral Interface.
- UART: Universal Asynchronous Receivers / Transmitters.
- V: Voltios.
- µF: Microfaradios.

Dedicatorias:

Le quiero dedicar este trabajo a mis padres, María y Javier. A mi novia Karen. A mi hermana Sol. A mis hermanos de otros padres, Matias, Agustin, Exequiel, Brian y Alexis. A mis compañeros de batallas que parecían interminables, Pablo, Agustin y Victor.

Capítulo 1: Introducción

En un mundo donde la robótica avanza a paso firme, es imposible imaginar un futuro sin robots en todos los aspectos de nuestras vidas. Primero los robots entraron en plantas industriales, reemplazando al humano en trabajos riesgosos y repetitivos mejorando así la eficiencia del proceso y calidad de los productos manufacturados, así como reduciendo la tasa de accidentes en trabajadores. Luego pasó a explorar ambientes hostiles para la vida humana, donde podemos mencionar las profundidades oceánicas y otros planetas. En la actualidad se pueden ver robots en lugares como quirófanos y hasta en tareas de atención al público, como es el caso del hotel japonés “Henn-na” atendido por robots.

En cualquiera de los rubros que se observen, la mayoría de los robots en ellos aplicados son del tipo fijo, ya que se encuentran quietos en algún sitio y su trabajo se basa en manipular y modificar piezas. Pero a éstos los podemos diferenciar de los robots cuyo trabajo se basa en trasladarse, donde podemos mencionar no sólo a los vehículos de exploración sino que también pueden mencionarse robots utilizados en la organización de depósitos (como los que utilizan algunas empresas de logística) y los vehículos autónomos que son una de las ramas de la robótica actualmente con mayor expansión.

En torno a éstos últimos se basará este proyecto final. En la robótica móvil los vehículos autónomos, tanto terrestres como aéreos, juegan un papel fundamental ya que prometen mejorar la eficiencia en el transporte de cargas, reducir los tiempos de viajes y mejorar considerablemente la seguridad, esto último es debido a que eliminan el factor humano principal causa de accidentes.

Este tipo de vehículos presenta desafíos únicos, ya que no sólo se requiere el manejo y el control del vehículo propiamente dicho, sino que a demás requiere de algoritmos que permitan establecer trayectorias para la navegación y que así cumplan objetivos deseados, que pueden ser la disminución del tiempo de viaje, la disminución en el consumo de energía en el desplazamiento y la disminución de la misma intervención humana, entre otros. A su vez cada objetivo presenta sus propios problemas, en donde el principal problema es el costo computacional que tiene normalmente la ejecución de los algoritmos de navegación; lo que limita su uso a los dispositivos que son capaces de ejecutarlos.

En relación a las problemáticas arriba mencionadas, en este Proyecto Final se desarrolló un kit que resuelve el manejo y control de un vehículo terrestre no tripulado, así como la obtención y filtrado de datos de interés; con el objetivo de abstraer esta etapa y que el usuario sólo se encargue del desarrollo del software de navegación y de los algoritmos de guiado. Lo que presenta un enfoque único ya que otros kits que se comercializan requieren de una programación a bajo nivel del sistema, implicando que el usuario deba resolver cosas como, por ejemplo, la implementación del control de la velocidad del motor y de los algoritmos de fusión de datos, entre otras.

Capítulo 2: Desarrollo

Para comprender el enfoque que se tomó al llevar a cabo el proyecto, presento a continuación un diagrama de bloques donde se pueden ver cada una de las partes que contiene este proyecto.

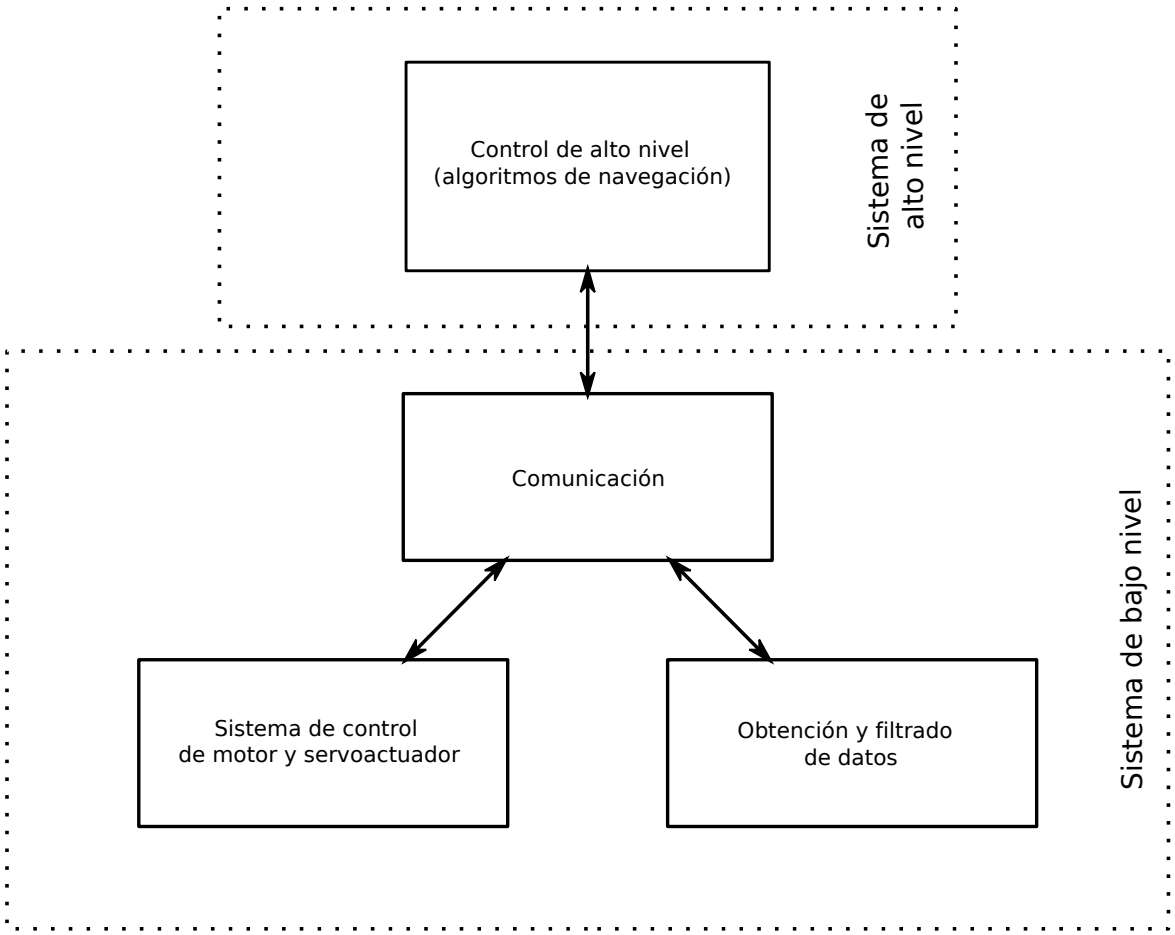

Figura 1: Diagrama General

Podemos ver en el diagrama que hoy das partes principales: un sistema de bajo nivel y un sistema de alto nivel. En el sistema de alto nivel es donde el usuario aplicará sus algoritmos de navegación. Mientras que el sistema de bajo nivel se encargá de realizar las tres tareas fundamentales representadas con bloques mas pequeños, un bloque que resuelve la comunicación, uno cuya tarea es la de controlar la velocidad del motor y la posición los sevoactuadores, y por último un bloque que tiene la funcionalidad de obtener, filtrar y fusionar los datos de interés.

Dentro del alcance de este proyecto se encuentra la implementación de todo el sistema de bajo nivel y también se decidió qué plataforma de hardware se utilizará como sistema de alto nivel.

A)Características constructivas

Antes de analizar cada bloque detalladamente, haré la aclaración de que el kit construyó partiendo de la base de un auto de modelismo en escala 1:10 aproximadamente. La construcción tuvo como premisa los siguientes puntos:

- Montaje y desmontaje simples. Por lo que las placas fueron montadas sobre laminas acrílicas que se sujetaron al vehículo mediante varillas de acero roscadas de 3,8 [mm].
- Conexiones robustas y accesibles fácilmente, esto fue resuelto utilizando borneras para los conexionados.
- Aislación de ruido electromagnético, lo que se resolvió separando la electrónica digital de la electrónica de potencia y también se agregaron láminas de aluminio conectadas a una masa común.
- Ubicación de la antena del GPS con respecto a los limites externos del auto conocida.
- Módulo inercial cercano al centro de gravedad del vehículo. Esto se debe a que, de no ser así, sería sometido a momentos angulares resultantes no solo del movimiento del vehículo sino que también del movimiento de este módulo respecto del centro de gravedad.
- Montaje de servoactuador firme y resistente. Este punto se debe ya que el brazo de acción es corto y por una cuestión de dimensiones es imposible hacerlo mas grande. Por lo tanto se requiere de un servo de gran torque, el utilizado tiene un torque de 9,4[kg . Cm] y esto nos obliga a utilizar un montaje fuerte. Para cumplir con este punto los soportes con los que se montó se hicieron en duraluminio.
- Batería liviana, pequeña y con la capacidad suficiente para alimentar todo el vehículo. Por lo que se optó por utilizar una batería a base de Litio-Polímero de dos celdas en serie, lo que da un voltaje de 7,4[V]; con un almacenamiento de 2200 [mAh] y una capacidad de liberación de la energía de hasta 70[C].

B)Sistema de bajo nivel

Este sistema se encuentra conformado por una plataforma microcontrolada, una placa de distribución de energía y por los diferentes sensores que se necesitaron. A continuación profundizaré en la elección de la plataforma microcontrolada y luego en los circuitos de la

placa de distribución de energía. Con respecto a los sensores utilizados, explicaré su elección y aplicación dentro de los sub-bloques “Obtención y filtrado de datos” y “Sistema de control de motor y servoactuador”

Se evaluó el uso de diferentes plataformas de microcontroladores, entre ellas se destacan la plataforma EDU-CIAA, la plataforma Piccolo de Texas Instrument y la plataforma Tiva C, también, de Texas Instrument.

Con respecto a la plataforma EDU-CIAA se pueden mencionar como ventajas, el hecho de que se trata de una plataforma libre, esto significa que su circuito se encuentra disponible. El microcontrolador que utiliza esta basado en un núcleo principal ARM Cortex M4 con un núcleo auxiliar ARM Cortex M0. También hay que destacar que forma parte del Proyecto CIAA, que es una iniciativa argentina llevada adelante por la industria y diferentes universidades. Como desventaja hay que mencionar, en primer lugar, su alto costo; así como también, su tamaño (se trata de una placa de grandes dimensiones) y, también, que el microcontrolador que posee (un LPC4337 de la firma NXP) tiene un poder de cálculo innecesariamente grande para este proyecto.

Cuando se analizó la plataforma Piccolo, se encontró como ventaja que el microcontrolador que posee, un TMs320F28027 de la firma Textas Instrument, es un “Microcontrolador de Tiempo Real” es decir que esta fuertemente orientado al control de sistemas, se puede mencionar también que cuenta con módulos específicos, por ejemplo, un módulo PWM para controlar “modulaciones por ancho de pulso” y un módulo QEP que ofrece funcionalidades útiles para el manejo de “encoders de cuadratura”. La desventaja principal que se puede mencionar es que se trata de un microcontrolador cuyo núcleo es propiedad intelectual de Texas Instrument, y forma parte sólo de la familia Piccolo por lo tanto no permitiría la migración del proyecto a otras plataformas, también cabe mencionar el desconocimiento de quien realizó el proyecto sobre la programación de estos microcontroladores.

Por último se analizó la plataforma fue la Tiva C de Texas Instrument. Esta plataforma cuenta con un microcontrolador basado en un núcleo ARM Cortex M4F, lo que permitiría migración del proyecto a cualquier otro microcontrolador con el mismo núcleo sin gran dificultad. Es una plataforma de bajo costo y pequeñas dimensiones. El poder de cálculo del microcontrolador es suficiente para este proyecto. También hay que mencionar que es una plataforma conocida por quien ejecutó este proyecto final. Como desventaja se puede mencionar que este microcontrolador no cuenta con módulos de PWM ni de QEP, pero dicha desventaja se pueden salvar configurando módulos de Timers para que funcionen generando modulación PWM (resolviendo el primer caso) y como contador para así contar los pulsos generados por los encoders.

La plataforma finalmente elegida fue la Tiva C de Texas Instrument.

En la distribución de energía y señales digitales se utilizó una placa diseñada específicamente para este proyecto. La placa cuenta con borneras para el acceso a la

tensión regulada y para conectar allí los sensores, cuenta también con una serie de pines donde se conectó la plataforma Tiva C y con un regulador de voltaje calibrado para regular la tensión de 5[V] a 3,3[V] para los sensores que así lo requieren.

Como fuente de energía se utilizó la batería y para regular la tensión a 5[V] se aprovechó el “Battery Eliminator Circuit”, con el que cuenta el ESC (Electronic Speed Controller), que entrega 5[V] regulados y hasta 2[A], analizaremos el uso del ESC mas adelante.

El circuito utilizado es muy simple y sólo merece la pena observar el circuito adicional del regulador MIC29302, que como se mencionó anteriormente es el regulador que baja la tensión de 5[V], entregados por el BEC del ESC, a 3,3[V] necesarios para muchos sensores. A continuación se muestra el circuito en detalle:

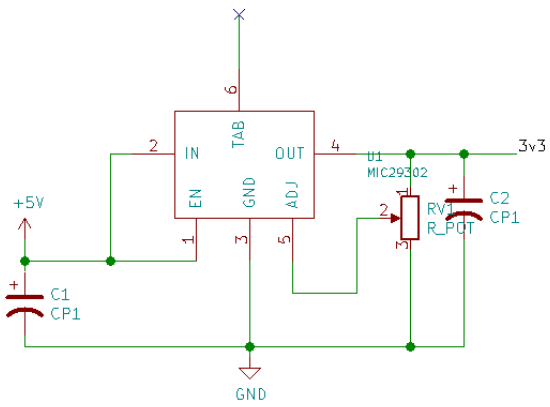

Figura 2: Circuito Regulador de Voltaje

Los capacitores CP1 y CP2 son de 100[µF] y 2,2[µF] respectivamente. Mientras que la resistencia variable RV1 que se observa es un preset, de forma que una vez calibrada para regular a 3,3[V] no sea movida de su lugar. También es importante mencionar que cuando se diseñó esta placa de distribución de energía se agregaron en ella resistencias de “pull-up” para darle funcionalidad a los pulsadores con los que cuenta la placa de desarrollo Tiva C.

1)Sistema de control de motor y servoactuador

En este sub-bloque nos encontramos con dos objetivos claros. Uno de estos objetivos es el de lograr un control de velocidad de motor y el otro objetivo es el de lograr un control de posición del servoactuador de la dirección del auto.

Los enfoques que se utilizaron para alcanzar los objetivos ya mencionados fueron completamente diferentes.

Control del servoactuador:

El servo utilizado es un MG996R, las especificaciones de éste se pueden ver a continuación:

- Peso: 55 [gr].
- Dimensiones: 40,7 x 19,7 x 42,9 [mm].
- Torque: 9,4[kg .cm].
- Voltaje de operación: 4,8[V] a 7,2[V].
- Rango de movimiento angular: 120°.

La operación del servoactuador se logra generando una señal periódica, cuyo período debe ser de 20[ms], y tiempo de su estado activo debe estar comprendido entre 1 y 2 [ms] de esta forma el brazo del actuador se mueve entre 0° y 120° respectivamente.

Por una cuestión constructiva del vehículo, el rango de movimiento libre de las ruedas es de tan sólo 50°, correspondientes con un movimiento desde 25° hasta 75° del servoactuador.

En este caso se optó por hacer un control a lazo abierto, debido a la dificultad que agregaría el sensado del ángulo de las ruedas.

En la siguiente imagen se puede ver el montaje del servo y el parte del mecanismo de dirección.

Figura 3: Montaje Servo de Dirección

Se terminó aplicando un simple control proporcional donde a la ganancia del sistema se la hizo aproximadamente unitaria. Esto último se logró aplicando una capa de software que permite establecer la “consigna” con el mismo valor del ángulo que se desea y esta capa de software hace las conversiones necesarias para que el valor escrito en el registro que genera el tiempo de “duty” del PWM que actúa sobre el servo genere en éste un movimiento acorde a esa “consigna”.

A continuación podemos ver una gráfica del sistema final implementado:

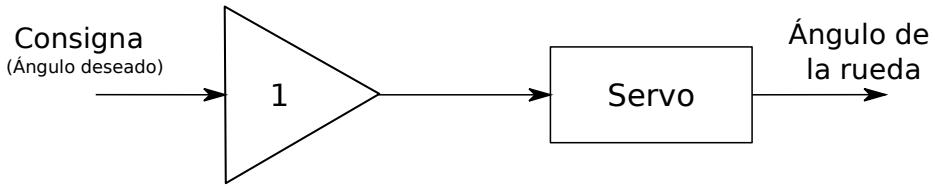

Figura 4: Esquema del control de la dirección

Control de velocidad del motor:

El vehículo cuenta con un motor de corriente continua, cuyo valor de tensión máxima que se le puede aplicar es de 9[V]. Otros valores relevantes en las características del motor no

pudieron ser obtenidos ya que formaba parte del auto de modelismo utilizado y no cuenta con una chapa de datos.

Este motor esta ubicado en la parte trasera del vehículo en posición transversal y transmite mecánicamente el movimiento a las ruedas traseras. La transmisión del auto cuenta con un sistema de diferencial que le permite, eventualmente, girar mas a una rueda que a la otra. Esto último le permite al vehículo tomar curvas sin que ninguna de las dos ruedas pierda tracción levantándose del piso, ya que al tomar una curva la rueda exterior recorre una mayor trayectoria que la rueda interior.

El control utilizado para manejar la velocidad de las ruedas, y con ello la velocidad del vehículo, fue un control del tipo MPC (modelo de control predictivo) con retroalimentación.

Para explicar el procedimiento que se llevó a cabo comenzaré esquematizando el sistema final deseado. Este esquema se puede ver en la siguiente imagen:

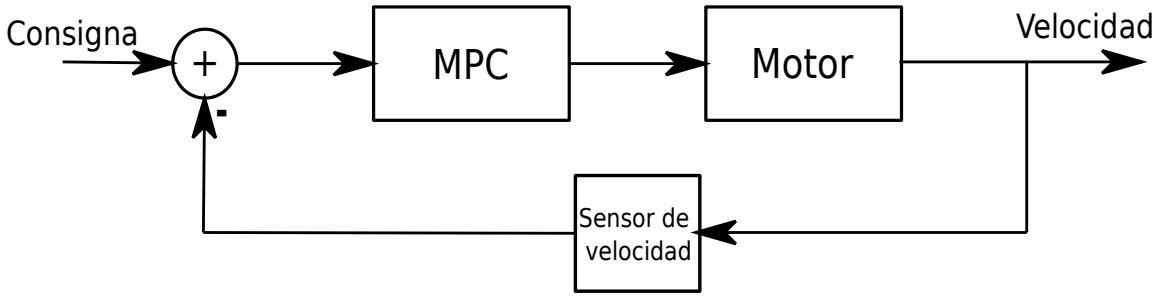

Figura 5: Esquema del control de la velocidad

En la imagen podemos ver tres bloques diferentes en el sistema. Un bloque llamado “MPC” que representa al controlador, un bloque que representa al motor, de nombre homónimo, y un bloque que donde queda representado el “Sensor de velocidad”.

Sensor de velocidad:

Al encarar el control, el primer problema que se encontró fue que ni el vehículo, ni el motor, contaban con algún tipo de sensor de velocidad, de forma, que fue necesario evaluar cómo hacer la realimentación de velocidad.

En un principio se pensó en utilizar encoders de cuadratura comerciales, pero esta posibilidad se descartó debido a que eran necesarias modificaciones mecánicas del sistema de transmisión para poder engranar con éste el encoder.

La otra posibilidad evaluada fue la construcción de encoders de cuadratura tanto magnéticos como ópticos. En el primer caso la idea consistía en adherir a la rueda imanes y mediante un sensor de efecto “Hall” detectar la presencia del campo magnético de los imanes. En el caso del encoder óptico, se adheriría a la cara interna de la rueda una cinta

de papel opaco con un patrón de franjas negras y blancas impreso y mediante sensores infrarrojos reflectivos detectar la presencia de las franjas blancas y negras.

El sensor finalmente implementado fue el encoder óptico. El patrón de franjas mencionados se puede observar en la siguiente imagen:

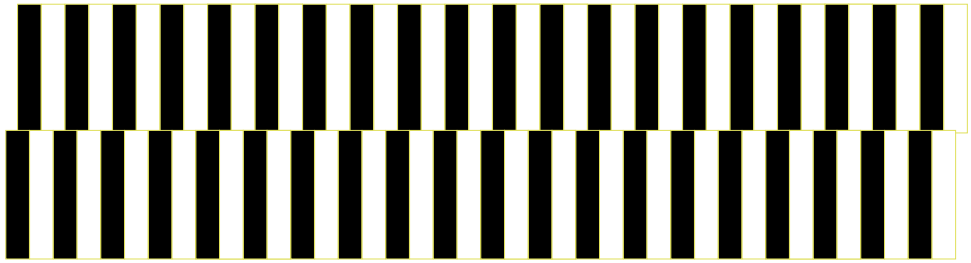

Figura 6: Patrón del encoder de cuadratura

Para lograr la cuadratura necesaria para detectar el sentido de giro de la rueda se agregó un patrón adicional con un desplazamiento de un 25% con respecto al primer patrón y un segundo sensor reflectivo para este segundo patrón. De forma que al girar las ruedas los sensores reflectivos generan señales cuadradas con un desfasaje entre ellas de un cuarto de ciclo adelantando una de las señales respecto de la otra, cuando la rueda gira en un sentido, y atrasándola cuando la rueda gira en el sentido contrario.

Los sensores infrarrojos reflectivos utilizados fueron los CNY70, debido a que son componentes fáciles de conseguir en el mercado argentino.

A continuación se puede ver un esquema para aclarar el principio de funcionamiento del sensor.

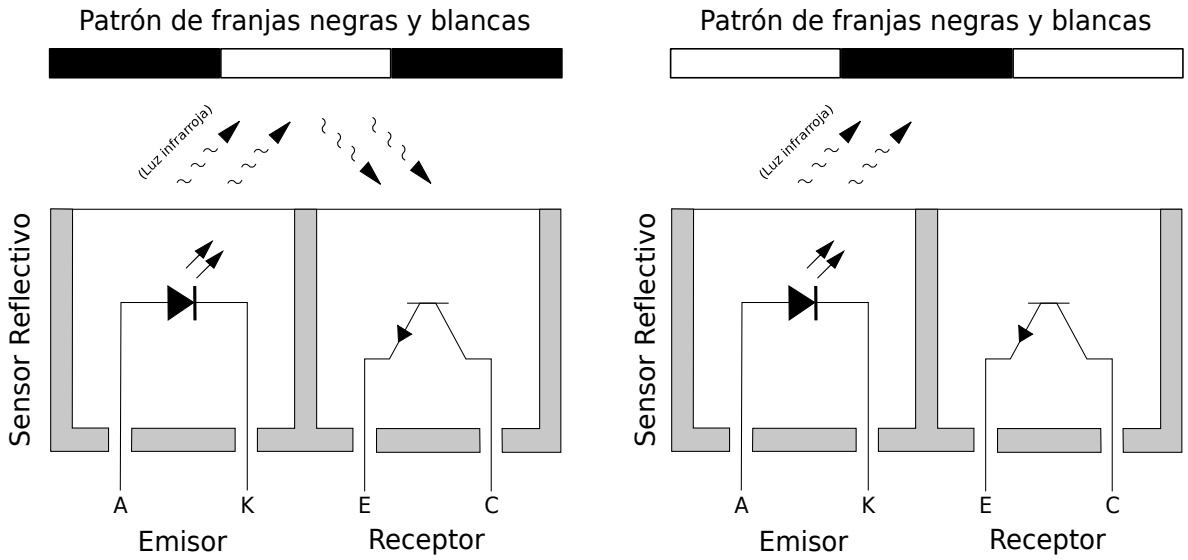

Figura 7: Diagrama del funcionamiento del encoder

El sensor reflectivo esta compuesto por un emisor que emite luz infrarroja y un receptor que al recibir luz infrarroja es excitado y conduce la corriente eléctrica. De esta forma y como se puede ver en la Figura 7 (imagen izquierda) cuando el emisor ilumina una franja blanca la luz se refleja y excita al receptor, mientras que cuando ilumina una franja negra (imagen derecha) la luz es absorbida y el receptor no es excitado.

El circuito construido para la conformación del encoder se puede observar en el siguiente esquema circuital.

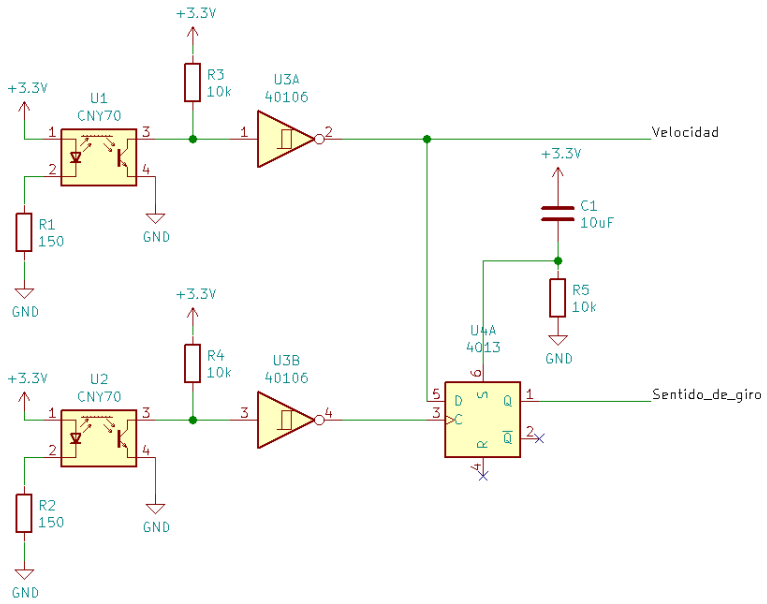

Figura 8: Esquema del circuito del encoder

Se puede ver que a la salida de los sensores reflectivos se colocaron compuertas negadoras tipo Schmitt Trigger, estas cumplen la función de buffers (es decir aumentan el nivel de la señal y le dan mayor inmunidad al ruido eléctrico). Luego de esto, una de las señales es directamente usada para la medición de la velocidad, y a su vez ambas entran en un Flip-Flop tipo D que nos permite detectar cual de las dos señales adelanta a la otra. Y de éste último componente se obtiene el sentido de giro de la rueda.

En la práctica este circuito se implemento en placas diferentes, una para el sensor infrarrojo y sus resistencias y otra para el buffer y el Flip-Flop. También hay que mencionar que en la implementación final se colocaron capacitores entre los bornes de alimentación para limpiar la alimentación de eventuales ruidos eléctricos, las resistencias utilizadas fueron resistencias de montaje superficial y se consideró también la necesidad de agujeros para el soporte mecánico.

A continuación podemos ver una de las placas de sensores obtenida.

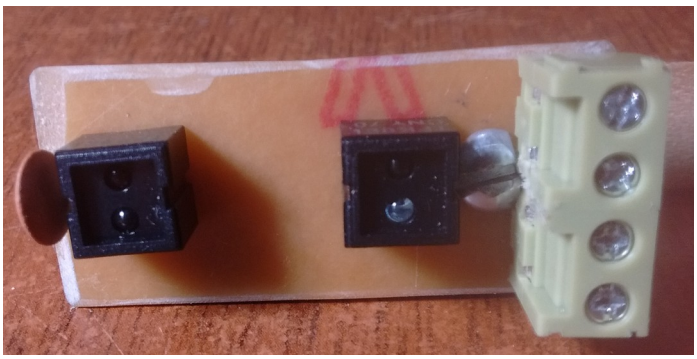

Figura 9: Foto del encoder montado

La placa central (la que tiene el buffer y el Flip-Flop D) quedó montada en el vehículo como se observa en la siguiente imagen.

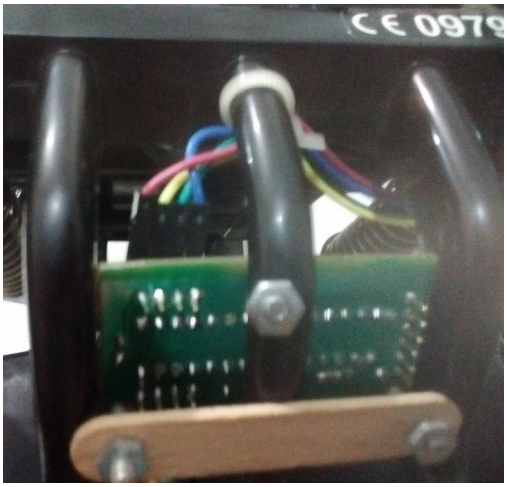

Figura 10: Foto de la placa central del encoder

Por cuestiones de ruido eléctrico también se agregaron armaduras cubriendo la placa para así aislarla del ruido principalmente generado por el motor. También los cables utilizados fueron blindados.

Un dato importante, que no se mencionó hasta ahora, es que todo el sistema de sensado hasta aquí explicado fue aplicado a ambas ruedas traceras, por lo tanto se construyeron dos placas con sensores y en la placa central se tiene la circuitería para ambas.

Por software la medición de la velocidad se logró contando los flancos ascendentes y descendentes de la señal “Velocidad” que se observa en el circuito de la Figura 8 mediante el uso de la función de “contador” con la que se pueden configurar los módulos de “Timer” del microcontrolador. Consultando los registros que llevan la cuenta de forma periódica se puede calcular la velocidad angular de las ruedas.

La fórmula utilizada para el cálculo es la siguiente:

$$Velocidad = Cantidad_{pulsos} \times \left(\frac{360}{40}\right) \times Frecuencia$$

Figura 11: Fórmula calculo de velocidad

Aquí "*Cantidad*$_{pulsos}$" es la cantidad de pulsos leídos del registro del contador.

La fracción (360/40) realiza la conversión de cantidad de pulsos a grados, la operación es simple cuando se tiene en cuenta que en una vuelta de rueda (es decir 360° de giro) se cuentan 40 pulsos.

Por último "*Frecuencia"* es la frecuencia con la que se consulta el registro del contador, esto es equivalente a dividir por el período de consulta.

Para terminar, observando las respuestas en una experiencia preparada para ensayar la respuesta del motor a una señal escalón en su entrada (esta experiencia se explicará mas adelante), se vio que las mediciones de velocidad tenían una varianza alta. Esto nos llevó a implementar un filtro digital pasa bajos para reducir, de esta manera, el ruido de alta frecuencia que se observaba en las mediciones. A continuación vemos una gráfica con las mediciones realizadas previas a aplicar el filtro.

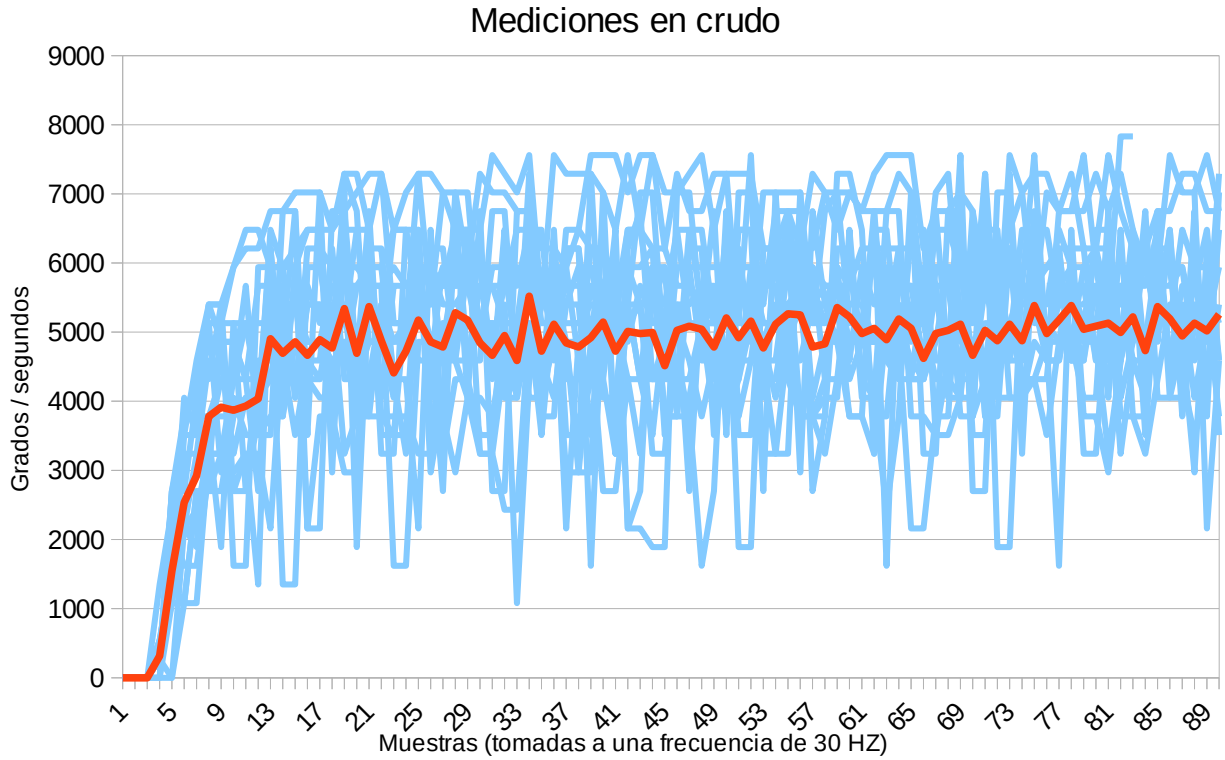

Figura 12: Gráfica de muestras tomadas

Aquí vemos en celeste las mediciones individuales y en color rojo la gráfica del promedio de las mediciones. Se puede observar claramente que la desviación estándar de las

muestras respecto de la media es alta. En particular la desviación estándar máxima fue de 1579,47[°/s].

El filtro que se decidió aplicar es conocido generalmente como filtro MAF (por siglas en ingles de “Filtro de media móvil”), en particular el usado es tipo MAF Exponencial. En pocas palabras es un filtro donde se promedian todas las muestras pero ponderándolas de manera exponencial, por lo que las muestras mas antiguas tienen menor influencia en el filtrado de la muestra actual.

En la siguiente expresión matemática podemos ver el cálculo que se realiza para aplicar el filtrado.

$$V_f[t] = \alpha \,.\, V_m[t] + (1-\alpha) \,.\, V_f[t-1]$$

Figura 13: Expresión del filtro MAF aplicado

Donde $V_f[t]$ y $V_f[t-1]$ representan la velocidad filtrada actual y la última anterior. Mientras que $V_m[t]$ representa la velocidad medida (es decir sin filtrar) en la muestra actual. Por último el valor α es un parámetro del filtro que permite darle mayor o menor filtrado.

El valor del parámetro α que se utilizó en el filtro aplicado fue de 0,3. Las mediciones realizadas con el filtro digital implementado se pueden ver graficadas a continuación:

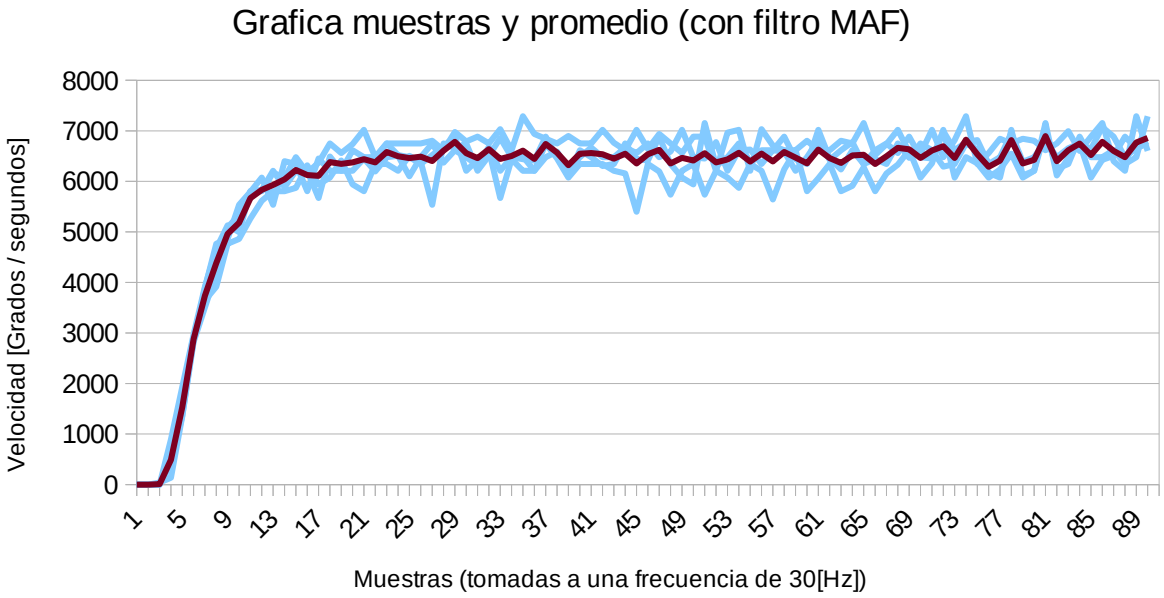

Figura 14: Gráficas de muestras tomadas luego de aplicado el filtro

Se pueden observar en celeste las mediciones tomadas (con filtro aplicado), mientras que la media se graficó en color marrón. Observamos, en contraste con la gráfica de las muestras sin filtrar, que las mediciones filtradas tienen una desviación estándar con

respecto a la media mucho menor. En particular la desviación estándar máxima medida entre todas estas muestras fue de 684,06[°/s].

Motor:

En la Figura 5 observábamos el sistema con todas las partes involucradas en el control de la velocidad. Ahora trataremos el bloque llamado “Motor”.

Para poder realizar el control de un sistema, lo primero que debemos obtener es su modelo matemático, que corresponde con la representación matemática del sistema físico. Las expresiones matemáticas que rigen el funcionamiento se pueden obtener mediante un procedimiento denominado “Identificación de sistema”.

En nuestro sistema es de interés el control de la velocidad del vehículo, por lo que deberíamos obtener el modelo matemático que tiene el motor por sí solo y agregar a este el modelo del sistema físico que debe mover, en otras palabras nuestro sistema deberá tener la dinámica que impone el motor y la que impone la masa del vehículo. Consideramos despreciable en este análisis la dinámica que tiene el circuito correspondiente al “Puente H” (que es el circuito electrónico que regula la energía que se le entrega al motor) y la correspondiente al sensor de velocidad, ya que son circuitos electrónicos y su dinámica es normalmente despreciable respecto a las impuestas por las características físicas de los sistemas.

Obtener la dinámica de la masa del vehículo no es un proceso simple de realizar. Por lo tanto, se optó por obtener el modelo matemático del sistema en conjunto. Es decir, el modelo del motor y el de la masa del auto juntos como parte de un único sistema.

Para lograr la obtención del sistema mencionado en el párrafo anterior se desarrolló una experiencia donde se impuso en el motor una señal tipo “escalón”. Esto significa que al vehículo que se encontraba en un estado de reposo se lo acelera con un valor conocido de consigna a la entrada del motor. En este este proyecto el motor se controló con un ESC320A que es un circuito con configuración de “Puente H”, pero con la particularidad de que su control es mediante una señal de PWM con un período de 20[ms] con un nivel de “duty” que debe tener una duración de entre 1 y 2 [ms]. Siendo 1[ms] el valor que impone en los bornes del motor la tensión máxima negativa y 2[ms] el que impone la tensión máxima positiva; por lo tanto 1,5[ms] se utiliza para mantener el motor frenado.

El programa que se escribió y compiló para la ejecución de esta experiencia queda representado en el siguiente diagrama de flujo.

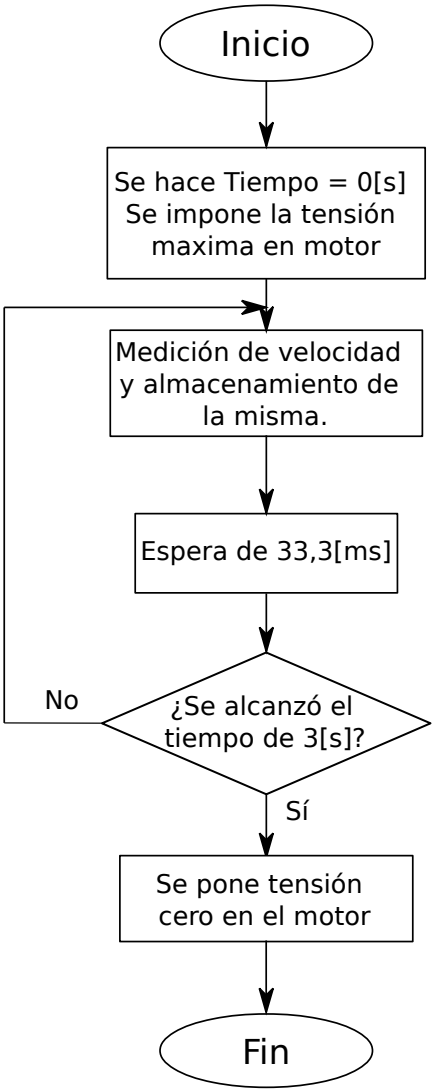

Figura 15: Diagrama de flujo de la experiencia del motor

Al hacer la identificación del sistema lo primero a observar son las características de la curva que nos indican si es un sistema de primer o de segundo orden. Nuestro sistema tiene características de un sistema de segundo orden, esto se debe a que habíamos decidido modelizar el sistema motor y sistema masa del vehículo en conjunto. Debido a que la dinámica del motor era claramente muy pequeña en comparación con la dinámica de la masa del vehículo, se decidió aproximar el sistema a un sistema de primer orden. Por lo tanto sólo nos interesaron los valores de la ganancia en continua del sistema y del tiempo de respuesta de éste.

Los datos almacenados durante los 3[s] que duró la prueba fueron analizados con el software Octave. Haciendo uso de la función MOESP, obtuvimos el modelo matemático del sistema donde su función de transferencia en el dominio de Laplace es:

$$H(s)=\frac{0{,}1931}{0{,}4884*s+1}$$

Figura 16: Función de transferencia del sistema Motor-Vehículo

La gráfica comparativa de la respuesta simulada a base del sistema modelizado con las mediciones realizadas en la experiencia se pueden ver en la siguiente imagen.

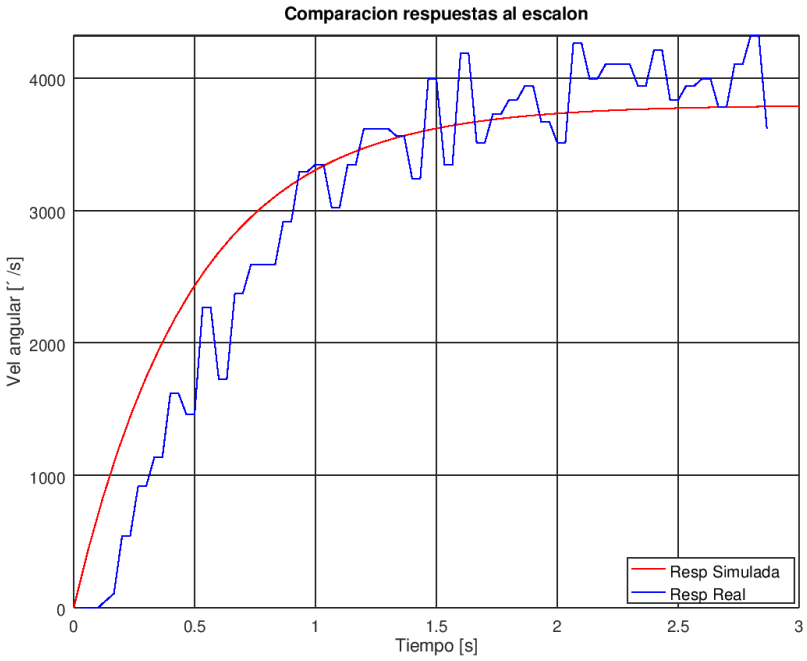

Figura 17: Grafica de la respuesta del sistema modelizado y el real

Este mismo sistema tiene una ecuación de estado con la siguiente forma:

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix}=\begin{pmatrix}0 & 1\\ 0 & -2{,}048\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}+\begin{pmatrix}0\\ 0{,}3956\end{pmatrix}u$$

Figura 18: Ecuación de estado del sistema Motor-Vehículo

Donde x_1 y x_2 son la posición y la velocidad del motor respectivamente y u corresponde con la entrada del sistema, y esta restringida a valer sólo ente ±19660 ya que estos son los valores límites que podemos utilizar para (aplicando una suma que corresponde con el

valor medio) escribir en el registro que regula el “duty” de la señal de PWM que generamos con el microcontrolador para manejar el ESC.

A partir de esta ecuación de estado se avanzó sobre el control MPC, pero los resultados fueron que el valor de “off set” (es decir el error que tiene el sistema una vez pasado el transitorio) era muy altos. Con esto en mente se procedió a modificar el modelo matemático manualmente agregando a él un efecto integral.

Matemáticamente se comenzó planteando:

$$u = u(k-1) + \Delta u$$

Figura 19: Expresión del reemplazo de u(k)

Luego este reemplazo se aplicó a la ecuación de estado del sistema y se despejó Δu como la nueva entrada del sistema. De esta forma la ecuación de estado de nuestro sistema quedó:

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ u(k) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -2{,}048 & 0{,}3956 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \\ u(k-1) \end{pmatrix} + \begin{pmatrix} 0 \\ 0{,}3956 \\ 1 \end{pmatrix} \Delta u$$

Figura 20: Ecuación de estados con efecto integrador incluido

Es importante aclarar acá que ahora nuestra “entrada” al sistema se transformó en una variable de estado.

Controlador MPC:

El modelo de control predictivo es una estrategia de control basada en una optimización numérica. Dicha optimización se aplica a una función de costo, que representa la dificultad de llevar al sistema a la consigna deseada pasando a través de todos sus estados y generando así las entradas necesarias para ello. Este tipo de control admite restricciones, tanto de estados como de entrada. Esto último significa que mantendrá siempre la entrada dentro de los límites que las restricciones representan, y los estados también se mantendrán dentro de estas restricciones. Para ejemplificarlo podríamos decir que, si tenemos un componente electrónico cuya entrada soporta entre -2[V] y 5[V] entonces estos valores serán nuestras restricciones de entrada; mientras tanto imaginemos un motor de combustión interna donde por cuestiones constructivas su eje no puede superar una velocidad angular superior a las 5500 RPM, siendo esta velocidad angular una variable de estado en el modelado matemático del sistema; entonces el sistema tendrá restricción de estado ya que una de sus variables de estado tiene un límite

que no debe superar. Este problema de optimización con restricciones se escribe matemáticamente en una forma reducida de la siguiente forma.

$$min_u \frac{1}{2} \sum_{j=0}^{N-1} \left(\left(x_j - xRef_j \right)^T Q \left(x_j - xRef_j \right) + \left(u_j - uRef_j \right)^T R \left(u_j - uRef_j \right) \right)$$

$$sujeto\ a: \quad \begin{aligned} x_{(j+1)} &= Ax_j + Bu_j \\ u_{lb} &\leq uj \leq u_{ub} \\ e_{lb} &\leq K_x x_j + K_u u_j \leq e_{ub} \end{aligned}$$

Figura 21: Expresión de la optimización del MPC

La primera expresión que podemos ver corresponde a la operación de optimización, optimizando para la variable *u*. Abajo de esa expresión vemos las tres expresiones a las que esta sujeta la optimización, la primera corresponde a la ecuación de estado del sistema a controlar, la segunda representa las restricciones de entrada y por último la tercer expresión representa restricciones cruzadas donde nuestras variables de estado sumada nuestra entrada no debe salir de los límites exigidos (esto nos garantiza que el sistema no se vaya a ir fuera de las restricciones en tiempos futuros).

El problema de optimización que plantea el controlador predictivo busca encontrar el camino a través del cuál el sistema converja a la consigna deseada de la forma menos costosa posible, o se podría decir también que lo busca hacer de la forma mas fácil posible.

En la práctica para implementar el controlador MPC se utilizó una herramienta, que esta provista en forma de paquete de Python, llamada “µAO MPC” desarrollada en el Instituto para la Ingeniería en Automatización de la Universidad de Magdeburg.

Esta herramienta permite cargar el sistema a controlar descripto a través de sus matrices de estados para poder generar las entradas predichas óptimas y simular así el sistema en una computadora personal que contenga un intérprete del lenguaje de programación Python. Pero la característica por la que se eligió esta herramienta es que cuenta con la funcionalidad de generar el código C para implementar el controlador en aplicaciones microcontroladas. Por lo tanto la implementación del controlador MPC se realizó a partir de este código generado.

Para generar el código C fue necesario definir las matrices Q y R que se observan en la expresión de la Figura 21 . En particular a la matriz Q se armó de manera tal que al multiplicarla por las variables de estado pierdan importancia las variables que no son de interés controlar y ganen mayor importancia las que sí son de interés. Esto nos llevó a definir la siguiente matriz Q:

$$Q=\begin{pmatrix} 0.0000001 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.495 \end{pmatrix}$$

Figura 22: Matriz de ponderación del control MPC

Si recordamos que nuestro vector de estados esta compuesto por x_1, x_2 y u, que representan la posición del rotor, la velocidad angular y la entrada del motor respectivamente; podemos observar que la matriz Q vuelve completamente despreciable a la posición angular del rotor, le da importancia a la velocidad angular (que es lo que deseamos controlar) y le da una importancia mucho menor al control de la entrada del motor.

Con respecto a la matriz R, que en nuestro caso particular como se trata de un sistema con una única entrada tiene dimensión unitaria, se debieron probar valores hasta lograr uno que ofreciera mejor comportamiento. Cabe aclarar que este valor, es en cierta forma un nivel de la sensibilidad con la que el controlador manipula la entrada del sistema u, siendo mas sensible mientras mas pequeño es el valor. Para nuestro caso el valor que ofreció el mejor comportamiento fue con:

$$R=0{,}0242$$

2)Obtención y filtrado de datos

Los datos de interés para los algoritmos de navegación de alto nivel, son la posición absoluta y la orientación que tiene el vehículo respecto al norte magnético de la tierra. Estos datos se obtienen a partir del módulo GPS y de la unidad magnético inercial (IMU). Como cada uno de ellos ofrece problemáticas particulares se estudiarán por separado.

Módulo GPS:

Se utilizó el módulo GY-NEO6MV2, este módulo esta basado en el dispositivo NEO6M que es un chip que tiene la funcionalidad de receptor GPS. El módulo tiene las siguientes características:

- Incorpora una antena cerámica,
- Soporta una tensión de alimentación de entre 3[V] y 5[V],
- Tiene una memoria EEPROM para guardar las configuraciones cuando el módulo se desenergiza,
- Tiene un led indicador,
- Sus dimensiones son: 23[mm] x 30[mm],

- El tamaño de la antena es de 12[mm] x 12[mm].

Este módulo trabaja enviando tramas de datos que se encuentran definidas por el protocolo NMEA-0183, por medio de una comunicación serie. Esto nos presenta dos dificultades una que radica en la comunicación y la otra es la extracción de los datos de interés de la trama NMEA-0183.

Para resolver la dificultad impuesta por la comunicación con el módulo GPS se resolvió utilizar un módulo UART del microcontrolador configurado a 9600 [Baudios] que es a lo que trabaja el módulo GPS. Debido a la carga computacional que requería atender a la recepción de los datos y moverlos en espacios de memoria se hizo uso del módulo uDMA que tiene el microcontrolador utilizado, este módulo tiene acceso directo a la memoria de programa y a los registros de los módulos de hardware del microcontrolador, por lo tanto éste se configuró de manera que cada dato que llegue sea copiado automáticamente en vectores sin intervención del núcleo del microcontrolador.

El protocolo NMEA-0183 establece una trama de datos que tiene la siguiente forma:

```
$GPRMC,192143.00,A,3146.21497,S,06029.72376,W,0.047,,251018,,,A*7C
$GPVTG,,T,,M,0.047,N,0.087,K,A*2F
$GPGGA,192143.00,3146.21497,S,06029.72376,W,1,05,1.57,91.1,M,18.1,
M,,*6B
$GPGSA,A,3,24,29,25,12,31,,,,,,,,2.52,1.57,1.98*0F
$GPGSV,3,1,10,02,44,127,,05,15,066,20,06,04,143,,12,55,106,30*7C
$GPGSV,3,2,10,14,09,254,,24,27,009,39,25,62,190,33,29,54,256,41*73
$GPGSV,3,3,10,31,14,220,18,32,08,275,*7F
$GPGLL,3146.21497,S,06029.72376,W,192143.00,A,A*69
```

La trama cuenta de tres partes un comando, los datos y el código de redundancia cíclica. El comando esta indicado al comienzo y sigue a un símbolo $ y esta conformado por un conjunto de letras como por ejemplo GPRMS, los datos son todos los valores que siguen al comando y que se encuentran separados por comas y por último el CRC se encuentra al final después de un carácter asterisco y esta expresado en base hexadecimal.

Para la extracción de los datos se utilizó una algoritmo de procesamiento de los flujos de datos NMEA-0183 que fue desarrollado por Sergio Burgos y presentado en el Congreso Argentino de Sistemas Embebidos 2016 bajo el título “Implementación en lenguaje C de un algoritmo para procesamiento de flujos de datos NMEA-0183”. El algoritmo esta basado en un funcionamiento de máquina de estado a medida que van llegando los datos es capaz de detectar en qué parte de la trama se encuentra y así separar el comando de los datos y verificar si el CRC es correcto. Luego delega la separación de los datos a funciones de parseo que son llamadas dependiendo del comando NMEA del que se trate.

Unidad de medición inercial (IMU):

Para las mediciones de aceleración lineal, velocidad angular e intensidad de campo magnético se utilizó el módulo inercial LSM9DS1 de la firma “ST”.

Este circuito integrado tiene las siguientes características:

- Permite medir tanto las aceleraciones, la velocidad angular y la intensidad de campo magnético en tres ejes,
- Cuenta con interfaces de comunicación “I2C” y “SPI”,
- Soporta tensiones de alimentación en el rango de 1,9[V] a 3,6[V],
- Todas las mediciones tienen una precisión de 16bits.
- Contiene un sensor de temperatura integrado,
- Incorpora una cola FIFO,
- Fondos de escala configurables para el acelerómetro, ±2, ±4, ±8 y ±16 [g],
- Fondos de escala configurables para el giróscopo, ± 245, ±500 y ±2000 [°/s],
- Fondos de escala configurables para el magnetómetro, ±2, ±4, ±12 y ±16 [Gauss].

Para la a aplicación desarrollada se configuró esta unidad inercial de la siguiente forma, la escala de medida del acelerómetro va desde -4[g] hasta 4[g], la escala del giróscopo quedó comprendida entre -2000[°/s] y 2000[°/s], ambos (acelerómetro y giróscopo) toman muestras de forma periódica con una frecuencia de 119[Hz]. Por último el fondo de escala del magnetómetro se configuró entre -12 y 12[Gauss] y su frecuencia de toma de muestras es de 80[Hz].

El interés en las medidas tomadas a través de este módulo radica en la obtención de la orientación relativa al campo magnético terrestre, de forma de poder saber en qué dirección esta “mirando” el vehículo. Para conseguir este objetivo se probó con algoritmos de fusión de datos. Es decir algoritmos que usan los datos entregados por esta IMU y los utilizan para conformar la orientación del vehículo.

Los algoritmos estudiados fueron los algoritmos conocidos bajo el nombre de sus autores como “algoritmo Mahony” y el “algoritmo Madgwick” ambos algoritmos están descriptos en sus respectivas publicaciones. El código fuente en lenguaje C de ambos algoritmos fue escrito por Sebastian O. H. Madgwick, y se encuentra disponible como código libre. También se consideró la implementación de un filtro Kalman complementario, pero debido a su complejidad computacional fue descartado rápidamente.

El problema que se encontró previo a realizar las pruebas de los algoritmos antes mencionados fue el de la calibración de la Unidad de Medida Inercial. En particular la medición con mayor sensibilidad a errores inducidos externamente es la del magnetómetro y requiere de calibración sin excepción.

La medición de la intensidad de campo magnético terrestre puede sufrir de diferentes tipos de errores, que se pueden separar en dos grupos: los errores por “distorsión de campo magnético” y los errores del sensor. Los errores por distorsión del campo

magnético que pueden ser producidos por los efectos denominados de “Hierro fuerte”, que es producido por la presencia de campos magnéticos permanentes en las cercanías del sensor, y “Hierro blando”, generado por la desviación del campo magnético que producen los metales ferromagnéticos que pueden encontrarse cerca del sensor (este efecto introduce errores de diferentes magnitudes dependiendo la orientación del sensor). Los errores del sensor responden a errores de desplazamiento de cero, errores de factores de escala, no ortogonalidad de sistema de ejes (*x, y, z*) y por ruido. El error por desplazamiento de cero significa que el sensor medirá cero cuando se encuentra en presencia de cierta magnitud de campo magnético, el error en el factor de escala significa que la medición en los diferentes ejes está sujeta a diferentes valores de escala, el error de no ortogonalidad de los ejes significa que alguno de los ejes puede estar midiendo componentes de campo magnético que le corresponden a otro de los ejes y por último el error por ruido significa que la medida puede ser afectada por ruidos del tipo magnético (particularmente por el ruido electromagnético).

A continuación escribiré la expresión matemática que incorpora todos estos errores, para explicar el procedimiento realizado para obtener los valores de calibración.

$$m = C_{no} C_{sf} C_{si} \left(m_i + b_H \right) + b_{zb} + \varepsilon$$

Figura 23: Expresión de los errores en la medición tomada

En esta expresión:

- ***m***, es la magnitud medida del campo magnético, tiene dimensiones de matriz de 3x1,
- $\boldsymbol{m_i}$, es la magnitud real del campo magnético (la que deseamos obtener), sus dimensiones de de una matriz de 3x1,
- C_{no}, es el error de no ortogonalidad y es una matriz de dimensión 3x3,
- C_{sf}, es el error de factor de escala y tiene dimensiones de matriz de 3x3,
- C_{si}, corresponde con el error de hierro blando y se escribe como matriz de 3x3,
- $\boldsymbol{b_H}$, es el error producido por el efecto de hierro duro, es una magnitud vectorial con 3 componentes,
- $\boldsymbol{b_{zb}}$, es el error de desplazamiento de cero y tiene el efecto de una magnitud vectorial,
- **ε**, representa al ruido y tiene magnitudes vectoriales.

Debido a la forma en la que afectan todos estos errores a nuestra medida es muy difícil obtener los valores de cada uno de ellos por separado, pero sus valores individuales carecen de importancia para la calibración del magnetómetro.

A la expresión de la Figura 23 se la puede escribir de la siguiente forma:

$$m = Km_i + b + \varepsilon$$

Figura 24: Expresión de los errores simplificada

donde se agruparon todos los errores matriciales en una única matriz *K*, los errores de magnitud vectorial fueron simplificados a un único vector ***b***. se decidió también despreciar el ruido **ε**.

Por lo que para obtener la medición calibrada del campo magnético terrestre, debemos hacer:

$$m_i = K^{-1}(m - b)$$

Figura 25: Expresión de medición real despejada

donde K^{-1} corresponde con la matriz inversa de la matriz *K*.

De esta expresión, y gracias a la unidad de medición inercial podemos obtener el valor ***m***, por lo que para lograr la calibración deseada sólo nos hace falta de la matriz inversa de *K* y del vector ***b***.

Para la obtención de estos dos valores se evaluaron diferentes experiencias.

La primer prueba que se evaluó fue mediante el uso de un software llamado “MotionCal” que es gratuito y abierto pero no libre, esto significa que se puede obtener gratuitamente tanto su ejecutable como su código fuente pero que no puede copiarse ni modificarse.

Una captura de imagen del software en cuestión se puede ver a continuación:

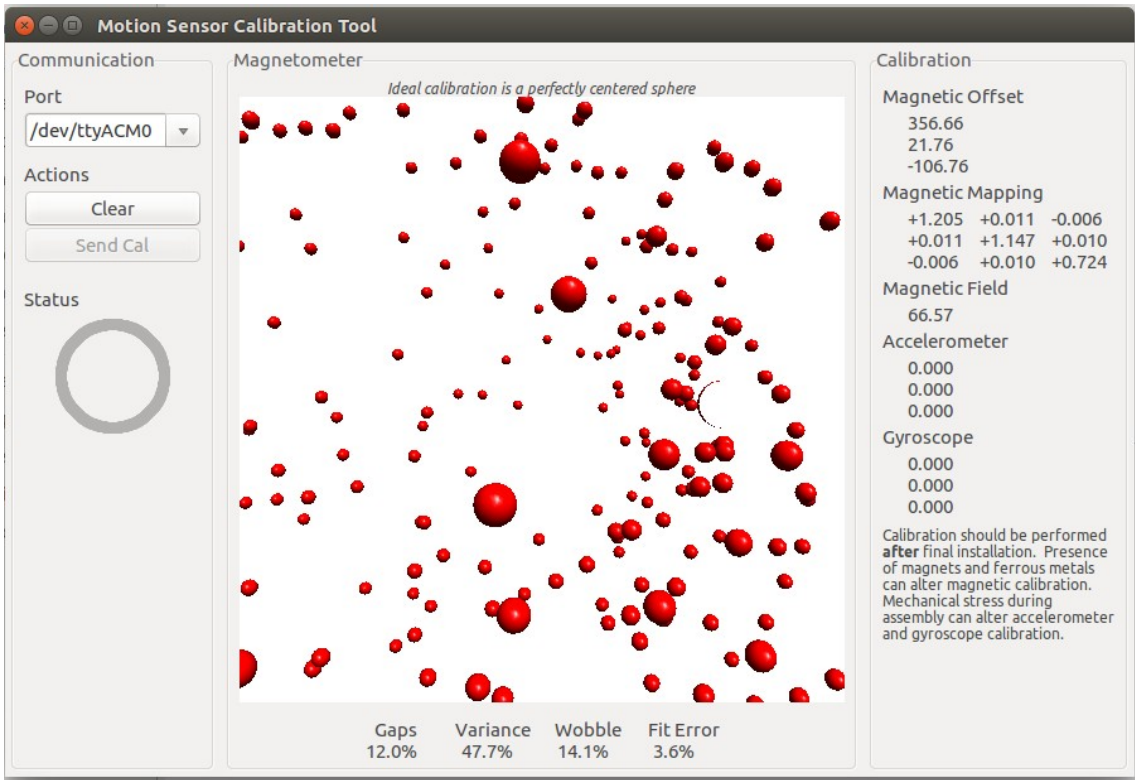

Figura 26: Captura de pantalla del software MotionCal

En la parte izquierda de la ventana se puede elegir el puerto por donde se esperan recibir los datos; en su zona central vemos una nube de puntos de diferentes tamaños, estos puntos representan la intensidad de campo magnético medida por el magnetómetro mediante su distribución (con una medición perfectamente calibrada los puntos deben quedar distribuidos formando una esfera perfecta); a la derecha de la ventana vemos una serie de valores bajo el título “Calibration”, entre estos podemos encontrar un “Magnetic offset” que corresponde con nuestro vector buscado ***b*** y una matiz llamada “Magnetic mapping” que correspondería con la matriz *K*.

Una vez que el software se encuentra en funcionamiento y comienza a recibir datos por el puerto, éste comienza a graficar los puntos y a calcular los valores de calibración necesarios. El usuario debe mover el objeto que cuenta con el magnetómetro en todos los sentidos y direcciones posibles, de forma de poder asegurar que el software cuente con la mayor cantidad de datos posibles para la generación de los valores de calibración.

El inconveniente que se encontró al usar este software fue la falta de documentación sobre en qué se centra su funcionamiento y sobre cómo se deben enviar los valores del acelerómetro, del giróscopo y del magnetómetro, en qué unidades deben encontrarse. Tampoco queda claro si la matriz “Magnetic mapping” corresponde con la matriz *K* que nosotros deseamos encontrar o si corresponde con su matriz inversa.

Una vez descartado el uso del software antes mencionado para la obtención de los parámetros de calibración se procedió a probar el algoritmo propuesto en el artículo

titulado “*Complete Tri-Axis Magnetometer Calibration with a Gyro Auxiliary*”. Donde los autores describen un proceso mediante el cual se toman mediciones que pueden ser analizadas por un algoritmo (desarrollado por ellos) para la obtención de los valores de *K* y de ***b***. Justifican, también, matemáticamente el desarrollo y los cálculos realizados en su algoritmo.

La obtención de las medidas se realiza siguiendo el siguiente proceso que explicaré a continuación.

Primero debemos imaginar que el sensor magnetómetro se encuentra dentro de una caja cuyas caras se están numeradas del 1 al 6. A su vez, los pares de caras 1-2, 3-4 y 5-6 son caras opuestas del cubo. El sensor debe estar colocado en el cubo de forma tal que el eje Z de su “frame” crezca en el sentido que “sale” atravesando la cara 1 del cubo (por lo tanto “entra” por la cara 2), el eje Y “sale” por la cara 3 del cubo (entrando por la 4) y por último el eje X “sale” a través de la cara 5 del cubo. De forma que quedaría imaginariamente algo así.

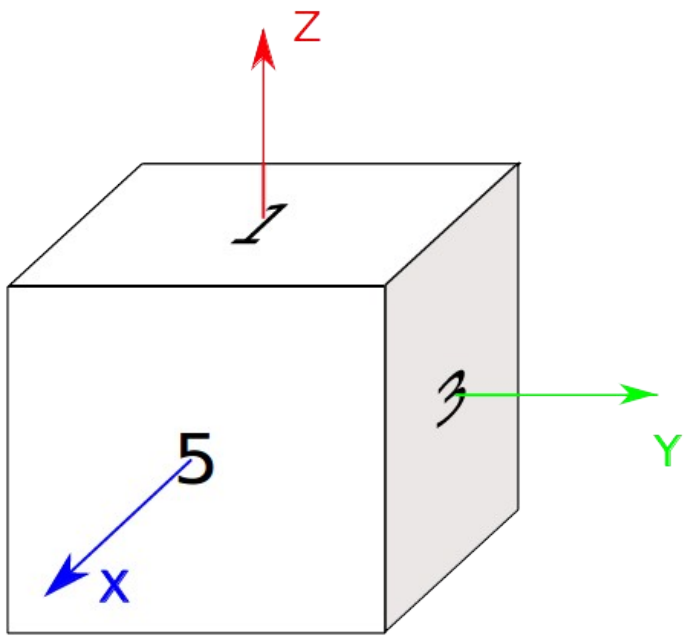

Figura 27: Sistema de ejes para la calibración de la IMU

Durante el desarrollo de esta prueba se debieron tomar mediciones del magnetómetro y del giróscopo a una frecuencia constante. Las muestras deben ser tomadas colocando el cubo que imaginamos anteriormente con la cara que tiene el número 1 hacia arriba en una mesa giratoria y debemos girarlo (no importa el sentido de giro) dando por lo menos una vuelta completa, pero para mejor precisión se recomienda un mayor número de vueltas. A este procedimiento lo esquematizamos en la siguiente imagen.

Los cálculos que se realizan como parte del algoritmo en el articulo, involucran la interpolación de valores de medición de forma que queden angularmente equidistantes por lo tanto es importante definir un ángulo de incremento entre muestras; para nuestro caso fue de 0,5°.

Al realizar las mediciones de esta manera, siempre nos aseguramos en el eje que queda en posición vertical se mida h_T que es la magnitud del campo geomagnético perpendicular a la tierra en la coordenada donde nos encontremos. Este valor puede ser obtenido utilizando un magnetómetro previamente calibrado, cartas de modelos magnéticos terrestres o aplicando la siguiente operación matemática.

$$h_T = \frac{\Delta\theta}{2160\,n} \sum_{i=1}^{360/\Delta\theta} \left(m_{z,i}^{1} - m_{z,i}^{2} + m_{y,i}^{3} - m_{y,i}^{4} + m_{x,i}^{5} - m_{x,i}^{6} \right)$$

Figura 28: Expresión componente perpendicular del campo geomagnético

En esta fórmula Δθ corresponde al ángulo utilizado para la interpolación de las muestras. Las $m_{z,i}^{1}$ con acento circunflejo corresponden a las muestras interpoladas, en su subíndice se indica qué eje es el vertical y el número de iteración de la sumatoria y en su superíndice se indica cuál es la cara del cubo hacia arriba de la muestra, esto último no debe confundirse con una potencia.

Para nuestro caso se probó con los datos obtenidos de tabla (se consultó "*The US/UK Wolrd Magnetic Model for 2015-2020*") y mediante la implementación de la fórmula de la Figura 28. Y en ambos casos el resultado fue el mismo.

El algoritmo se encuentra implementado una fórmula descripta en lenguaje Matlab como parte de un anexo a la misma publicación de Deng Yang. Para su utilización se transcribió a lenguaje Octave.

El resultado obtenido fue el siguiente:

$$K = \begin{pmatrix} -2{,}127 & -2{,}55 & -6{,}873 \\ 7{,}179 & 2{,}4 & 3{,}906 \\ -2{,}342 & -8{,}897 & 2{,}726 \end{pmatrix}$$

$$b = \begin{pmatrix} 34{,}866 \\ 11{,}274 \\ -17{,}424 \end{pmatrix}$$

Figura 29: Matrices de calibración de medición de la IMU

De este resultado se desprende el valor de la matriz inversa de K, que es el siguiente:

$$inv\_K = \begin{pmatrix} -0{,}488 & -0{,}052 & 0{,}007 \\ 0{,}152 & 0{,}43 & -0{,}062 \\ -0{,}037 & 0{,}01 & 0{,}365 \end{pmatrix}$$

Figura 30: Matriz inversa de K para aplicar en la calibración

La gráfica de los valores medidos debe formar, aproximadamente una esfera centrada en el origen de coordenadas. En nuestro caso las mediciones realizadas previas a la calibración dieron como resultado la siguiente gráfica.

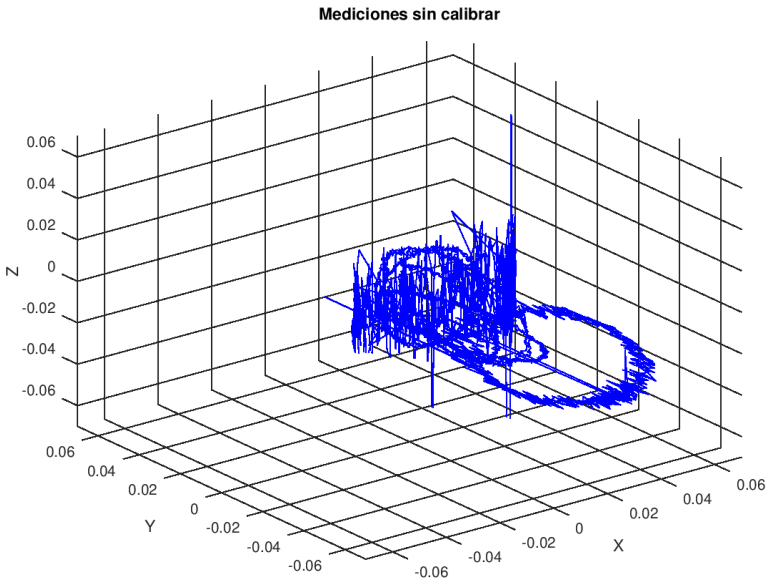

Figura 31: Mediciones con la IMU sin calibrar

Esta gráfica, si bien no se aprecia con suficiente claridad, forma un elipsoide que no se encuentra centrado en el origen y se puede observar estirado en su eje Y.

Sin embargo los valores calibrados arrojaron una gráfica con la siguiente forma.

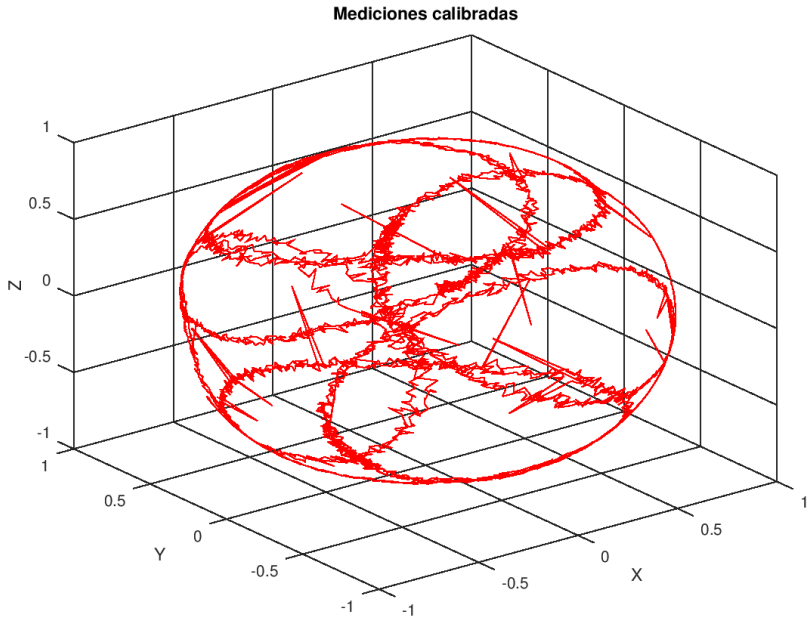

Figura 32: Mediciones con la IMU calibrada

Como podemos ver los valores calibrados sí se encuentran circunscribiendo una esfera centrada en el origen de coordenadas. También cabe aclarar que a éstos valores se los normalizó, es por esto que los límites de la gráfica se encuentran comprendidos entre -1 y 1 en sus tres ejes.Retomando el tema de la fusión de los datos, cuando contamos con un giróscopo y un acelerómetro, podemos establecer un sistema de posición referido a la dirección de la aceleración de la gravedad. Si a ese sistema le agregamos una medición de un magnetómetro podemos obtener un sistema de posición no solo referido a al vector de aceleración de la gravedad sino que también referido a un sistema de referencia magnético, usualmente referido al norte magnético. La necesidad de utilizar algoritmos de fusión de datos radica en conformar una medición precisa y eficiente, donde nuestro resultado sea en un sistema de representación coherente y único.

Como ya se mencionó, se probaron el algoritmo de Mahony y el algoritmo de Madgwick. Se terminó tomando la decisión de utilizar el algoritmo de Madgwick, pese a su mayor carga computacional (en comparación con el de Mahony), pero la perfonmance de éste esta validada con el uso de los tres sensores (acelerómetro, giróscopo y magnetómetro), no así el algoritmo de Mahony.

Para la aplicación del algoritmo hay que configurar la frecuencia de muestreo, que en nuestro caso se optó por usar 65[Hz], y un parámetro β que coincide con una ganancia del algoritmo y se calcula, según la publicación realizada por el mismo Madgwick, a partír de la fórmula:

$$\beta=\sqrt{\frac{3}{4}}\omega_{max}$$

Figura 33: Parámetro de configuración del algoritmo de Madgwick

Donde ω_{max} corresponde con el error de medición máximo del giróscopo. En nuestro β quedo definido en 4,569.

El resultado del algoritmo se encuentra dado en cuaterniones. Los cuaterniones forman parte de un sistema de representación, donde la orientación de un cuerpo sólido es representada mediante una notación de cuatro dimensiones con números complejos.

3)Comunicación

El objetivo de este bloque es el de intercambiar información con el sistema de alto nivel. Durante esta etapa se resolvieron cuestiones como, qué protocolo de comunicación se utilizaría, cómo se establecerían las tramas de datos a enviar y recibir, incluyendo también detalles de bajo nivel relacionados a la comunicación.

En un principio se evaluó el uso del protocolo de comunicación CAN (Controler Area Network), que es un protocolo desarrollado por la compañía alemana “Robert Bosch GmbH”. Esta concebido en base a una filosofía del tipo “Maestro – esclavo”, es decir que tenemos un procesador principal que delega tareas a otros dispositivos, comunicados todos a través de un bus. La topología del bus CAN esta normalizada y, además, es un protocolo ampliamente utilizado en vehículos debido a su amplia inmunidad al ruido.

El protocolo CAN fue descartado debido a que en nuestro caso se busca comunicar un sistema de alto nivel contra uno de bajo nivel, es decir se tiene un maestro y un único esclavo. Además nuestro sistema de alto nivel (se profundizará sobre él mas adelante) no cuenta con módulos de comunicación CAN nativos, lo que hacía que la aplicación del mismo agregara complejidad al proyecto. También se debe mencionar que en nuestro vehículos las distancias son cortas, por lo tanto las líneas de comunicación no están tan expuestas al ruido como si lo estarían en un vehículo de grandes dimensiones y mayor complejidad en equipos.

Finalmente se optó por una comunicación del tipo SPI (Serial Peripheral Interface), que es un protocolo de comunicación originalmente desarrollado por la firma “Motorola”. Este protocolo de comunicación esta orientado al manejo de dispositivos periféricos desde un sistema digital central, soporta un número pequeño de periféricos y es capaz de conseguir altas tazas de transmisión de bits para dispositivos conectados a pequeñas distancias.

La elección de esta interfase de comunicación se hizo principalmente por la razón de que en nuestro caso se busca comunicar un maestro con un único dispositivo esclavo.

También la plataforma utilizada para nuestro maestro cuenta con un módulo de comunicación SPI.

Para establecer este tipo de comunicación se utilizan cuatro lineas de señales, estas se detallan en la lista que se presenta, a continuación:

- SCLK, es la señal a través de la cuál el dispositivo maestro genera el clock que le da sincronización a la comunicación.
- MOSI (Master Output Slave Input), esta señal es por la cuál el maestro envía los datos y el esclavo de la comunicación los recibe.
- MISO (Master Input Slave Output), a través de esta señal el esclavo envía los datos al maestro.
- SS/SELECT, finalmente esta señal es la que le permite al maestro indicar qué periférico es su otro interlocutor en la comunicación.

Por lo tanto la conexión entre los sistemas queda dada por estas señales, en la siguiente imagen podemos ver un esquema con ellos y con la dirección de estas señales.

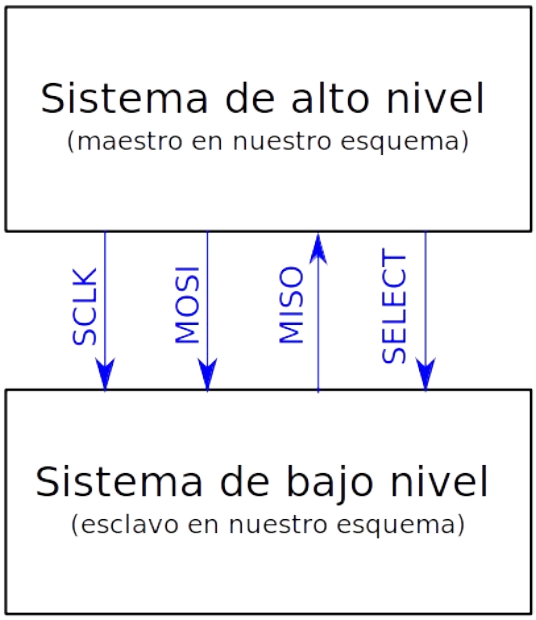

Figura 34: Esquema de la lineas de comunicación entre sistemas

Luego de establecido el protocolo de comunicación y de realizadas las conexiones se definió la velocidad de transferencia de datos entre los sistemas en 1[Mbits/s], es decir

que se transmiten un millón de bits en un segundo, se eligió el modo 0 de funcionamiento. El protocolo SPI soporta cuatro modos de funcionamiento, numerados de 0 a 3, donde se definen la manera en la que se inicia y se termina una transmisión de datos. También se decidió que el tamaño de palabra para la comunicación fuera de 8 bits.

Una vez establecido el medio por el que se realiza la comunicación se debió establecer la forma en la que se enviarán datos y qué datos se envían entre los sistemas de alto nivel y de bajo nivel.

Comenzando por resolver este segundo interrogante se estableció que los datos de importancia a transmitir desde el sistema de bajo nivel al de alto nivel (para que éste pudiera realizar los algoritmos de navegación) son:

- Coordenada Latitud.
- Coordenada Longitud.
- Ángulos de Euler (conocidos en aeronavegación como “Yaw”, “Pitch” y “Roll”).

Con estos datos se conformó la siguiente trama:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**
Campo:	Lon_gr				Lat_gr				Lon_mn								Lat_mn								Yaw				Pitch				Roll					F	F	CRC	0

Tabla 1: Distribución de datos en la trama de datos

En esta tabla podemos ver en la primer fila el índice de los bytes que conforman la trama. La segunda fila contiene los nombres de la información que contiene.

- **Lon_gr:** este bloque contiene los grados de la coordenada longitud, es un valor del tipo entero signado de 32 bits de largo. Si su valor es negativo significa que son longitudes ubicadas en el hemisferio oriental (Este).
- **Lat_gr:** este bloque contiene la información de grados de la coordenada de latitud. Al igual que el dato del bloque anterior es un valor entero signado de 32 bits que en caso de ser un valor negativo corresponde a latitudes en el hemisferio Sur.
- **Lon_mn:** aquí contenemos la información de los minutos con todos sus decimales de la coordenada Longitud. Es un dato del tipo “double”, lo que significa que su representación computacional es de punto flotante y el valor se encuentra almacenado en 64 bits.
- **Lat_mn:** este valor es homólogo al anterior pero corresponde con la coordenada Latitud.

- **Yaw:** en este bloque la información que se guarda es el ángulo de Euler **Ψ** , denominado “Yaw”. Es un dato del tipo “float” de 32bits.

  Pitch: Idem al anterior, con la diferencia que indica el ángulo **Θ**, denominado también “Pitch”.

- **Roll:** Idem a los anteriores, indica el ángulo **Φ**, conocido también como ángulo “Roll”.
- **F:** estos campos están completos a nivel binario con valores 1, a mayor nivel si se lo trabaja como enteros de 8 bits representan el número 255.
- **CRC:** aquí se encuentra el código de redundancia cíclica calculado con los bytes que van desde el byte de índice 0 al byte con índice 37 en la tabla.
- **0:** este campo esta completo a nivel binario con ceros, es utilizado para alinear el resto de los datos. Podemos decir que es un indicador de inicio de nueva trama, de esta forma sabemos dónde termina la trama y dónde inicia una nueva.

En el sistema de bajo nivel para que la transmisión de datos no involucre una carga innecesaria al procesador, debido a que este sistema es un sistema esclavo y debe esperar de la petición del maestro para enviar la información, se decidió utilizar el módulo DMA que contiene el microcontrolador “TM4C1233PH6”. Para mantener una copia siempre actualizada de los datos a enviar, se establecieron dos vectores en donde se guardan los datos nuevos. El módulo de acceso directo a memoria conmuta entre estos vectores y coloca sus datos en el registro de envío del módulo SPI. De forma que el procesado sólo interviene para recargar la información adecuada en los canales DMA, indicando en ellos en qué dirección de memoria se encuentra la información a enviar.

Como se explica en el párrafo anterior el módulo DMA se encarga de abastecer al módulo de comunicación SPI de los datos que se van a transmitir. Mientras que el flujo de ejecución del procesador se muestra en el diagrama de flujos siguiente:

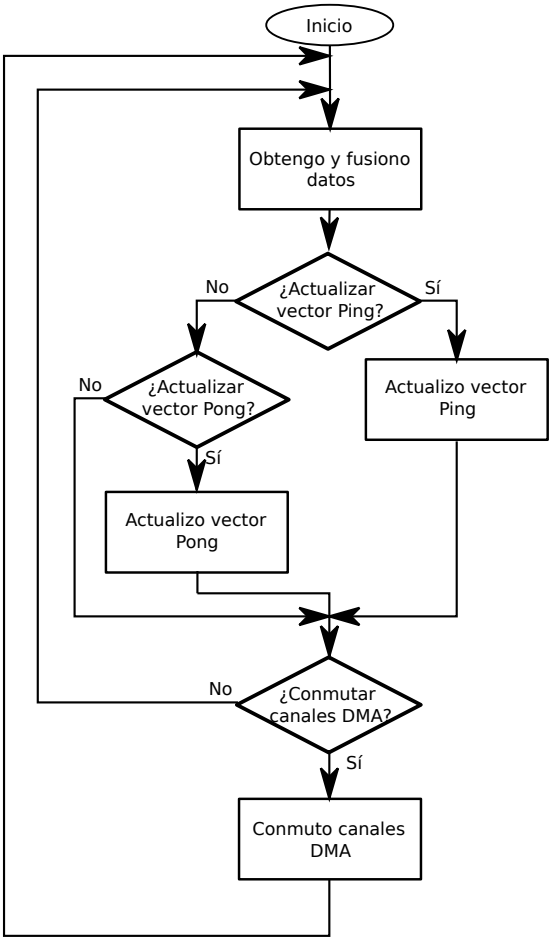

Figura 35: Diagrama de flujo del envío de los datos

En el diagrama de flujos anterior podemos ver cómo el procesador mantiene actualizado los vectores “Ping” y “Pong”. Estos son los espacios de memoria desde los que luego el módulo de Acceso Directo a Memoria (DMA) se encarga de tomar la información para cargarla en el registro de dato a enviar del módulo SPI. Otro detalle importante en el diagrama de flujos es que no tiene salida ni condición de salida, esto se debe a que el procesador ejecuta esta tarea como parte de un ciclo que no tiene final.

Hasta ahora se habló de lo referido a cómo se envían todos los datos sensados y fusionados por el microcontrolador al sistema de alto nivel. Con respecto a cómo el sistema de alto nivel le “dice” al sistema de bajo nivel qué debe hacer, se armó la siguiente trama:

Byte:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Campo:	Inicio	Función	Dato				CRC	Final

Tabla 2: Distribución de la información en la trama de ordenes

Para esto se trata de una trama mucho mas corta que la anterior. Los números de la primera fila de la tabla indican el índice de los bytes. Mientras que en la segunda fila vemos los campos de los que se trata, éstos se detallan, a continuación

- **Inicio,** este es un indicador de inicio de trama.
- **Función,** es un valor entero sin signo de 8 bits que representa la tarea que el sistema de alto nivel le ordena al de bajo nivel que ejecute. Como por ejemplo, modificar la consigna del control del moto.
- **Dato,** en este campo se envía la información correspondiente a la función que se desea, en el caso de la nueva consigna para el sistema de control aquí se envía la nueva consigna. Este campo es un dato entero con signo de 32 bits.
- **CRC,** es el código de redundancia cíclica, calculado con todos los datos anteriores.
- **Final,** corresponde al marcador de fin de la trama.

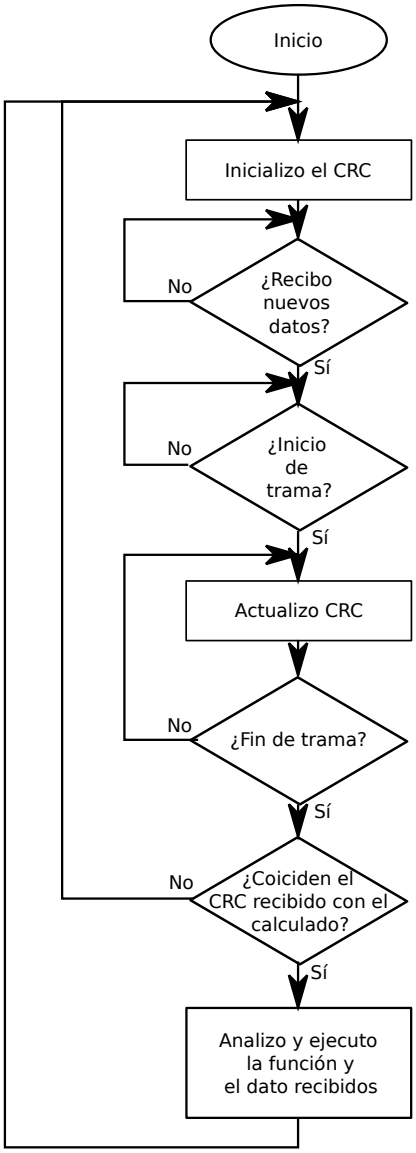

Figura 36: Diagrama de flujo de la recepción de la orden

De esta forma el sistema de bajo nivel para interpretar la orden que recibe ejecuta el algoritmo representado por el siguiente diagrama de flujos. Se puede ver en este diagrama que el algoritmo se ejecuta indefinidamente, es decir que el procesador analiza constantemente la llegada de los datos buscando un marcador de inicio de trama, cuando lo encuentra avanza actualizando el código de redundancia cíclica por cada dato nuevo que le llega. En caso de que el CRC calculado con el recibido coincidan procede a analizar la función que recibió, en caso de que no coincidan reinicia el cálculo del CRC y vuelve a esperar encontrar un nuevo marcador de inicio de trama.

C)Sistema de alto nivel

En este bloque se definió qué plataforma se utilizaría para que el usuario desarrolle el código de navegación del vehículo y se estableció también un paquete en lenguaje Python, que resuelve funciones de la comunicación entre los sistemas de alto nivel y de bajo nivel.

Para este bloque fue requisito fundamental utilizar una plataforma con alto poder de cálculo. No se establecieron como requisito características determinísticas, ni de tiempo real y si bien es muy útil tampoco se buscó específicamente plataformas con Sistemas Operativos.

Se analizaron como posibles plataformas, la BeagleBone Black, la plataforma Raspberry Pi y la plataforma Orange Pi. Entre las características que se utilizaron para la comparación se evaluaron, el costo, el tamaño, la facilidad de obtención (que se consiga en el mercado argentino), el voltaje y corriente de alimentación y que tengan una comunidad de usuarios grande y solidaria. Analizamos a continuación cada plataforma pormenorizado.

BeagleBone Black

Esta plataforma cuenta con un alto poder de cálculo, se consigue en Argentina, requiere una tensión de alimentación de 5[V] y 1[A] de corriente, se encuentra disponible un sistema operativo específico para las BeagleBone Black basados en una distribución Debian. Sus dimensiones son: 86,4[mm] x 53,34[mm], tiene un peso de 39,7[gr]. Cuenta también con una buena comunidad de usuarios y con documentación que cumple con las espectativas. Pero tiene la desventaja de su alto costo aquí en Argentina.

Raspberry Pi 3B+

Se trata de una placa con un procesador potente, tiene un sistema operativo específico basado en Debian (también soporta diferentes distribuciones GNU/Linux), sus requisitos de alimentación son de 5[V] con 2,5[A], tiene dimensiones de 85,6[mm] x 53,98[mm]. Es una plataforma muy popular, lo que cuenta con una comunidad de usuarios muy grande y también se puede encontrar mucha documentación respecto a ella. Se comercializa ampliamente en Argentina y a un buen costo.

Orange Pi

Esta plataforma cuenta con mejores prestaciones que las anteriores en lo referido a poder de cálculo, soporta diferentes versiones de sistemas operativos GNU/Linux, no cuenta con un sistema operativo específico. Tiene dimensiones de 85[mm] x 55[mm], con 70[gr] de peso. Es una plataforma mas económica que las evaluadas anteriormente, pero tiene la desventaja de la dificultad de conseguirla en el mercado argentino. La comunidad de usuarios de placas Orange Pi es pequeña y también hay poca documentación disponible.

Analizadas todas las opciones se optó por la plataforma Raspberry Pi, utilizando su sistema operativo Raspbian.

D)Control del vehículo por el usuario

Finalmente para completar la solución que ofrece este kit, se desarrolló una clase en Python, en forma de paquete para este lenguaje de programación. Un objeto de esta clase resuelve toda la comunicación desde el lado de la Rapberry Pi, de esta forma se puede implementar un algoritmo de navegación sin andar preocupándose por cómo armar la trama, como transmitirla, cómo verificar la trama recibida, etc.

Esta clase se llama VehiculoANT (por Vehículo Autónomo No Tripulado), y tiene los siguientes métodos:

- **actualizar_datos():** Mediante este método se actualizan los valores de los datos leídos desde el microcontrolador (latitud, longitud, yaw, pitch, roll). Luego estos pueden ser accedidos desde los atributos que llevan los mismos nombres.
- **actualizar_velocidad(nueva_velocidad):** Este método se utiliza para actualizar la consigna del controlador de la velocidad del vehículo.
- **actualizar_direccion(nueva_direccion):** Mediante este método, nuestro objeto es capaz de modificar la consigna del control de la dirección del vehículo
- **frenar_auto():** Este método se utiliza para modificar la consigna de velocidad del control de velocidad, llevando ésta a cero. También se puede usar actualizar_velocidad(0), en su lugar.

Los atributos que contiene este objeto, y son atributos que se pueden consultar, son los siguientes:

- **Latitud:** en este atributo contiene la información de la latitud a la que seencuenta el vehículo y se actualiza cada vez que se ejecuta el método “actualizar_datos()”.
- **Longitud:** almacena el valor de la coordenada longitud en la que se encuentra el vehículo. Se actualizan usando el método “actualizar_datos()”.
- **Yaw:** este atributo almacena el ángulo de Euler denominado de esta forma. Se actualizan usando el método “actualizar_datos()”.
- **Pitch:** similar al atributo anterior pero en este caso corresponde con el ángulo llamado Pitch. Se actualizan usando el método “actualizar_datos()”.
- **Roll:** Idem. Se actualizan usando el método “actualizar_datos()”.

Capítulo 3: Resultados

Del desarrollo expuesto en el capitulo anterior se obtuvo como resultado un vehículo que resuelve todas las problemáticas de bajo nivel que presenta la robótica móvil cuando se pretende desarrollar algoritmos de navegación y trayectorias.

A continuación podemos observar algunas imágenes del montaje de los diferentes componentes, comenzando por el montaje de la placa de distribución de energía y la plataforma de desarrollo Tiva C.

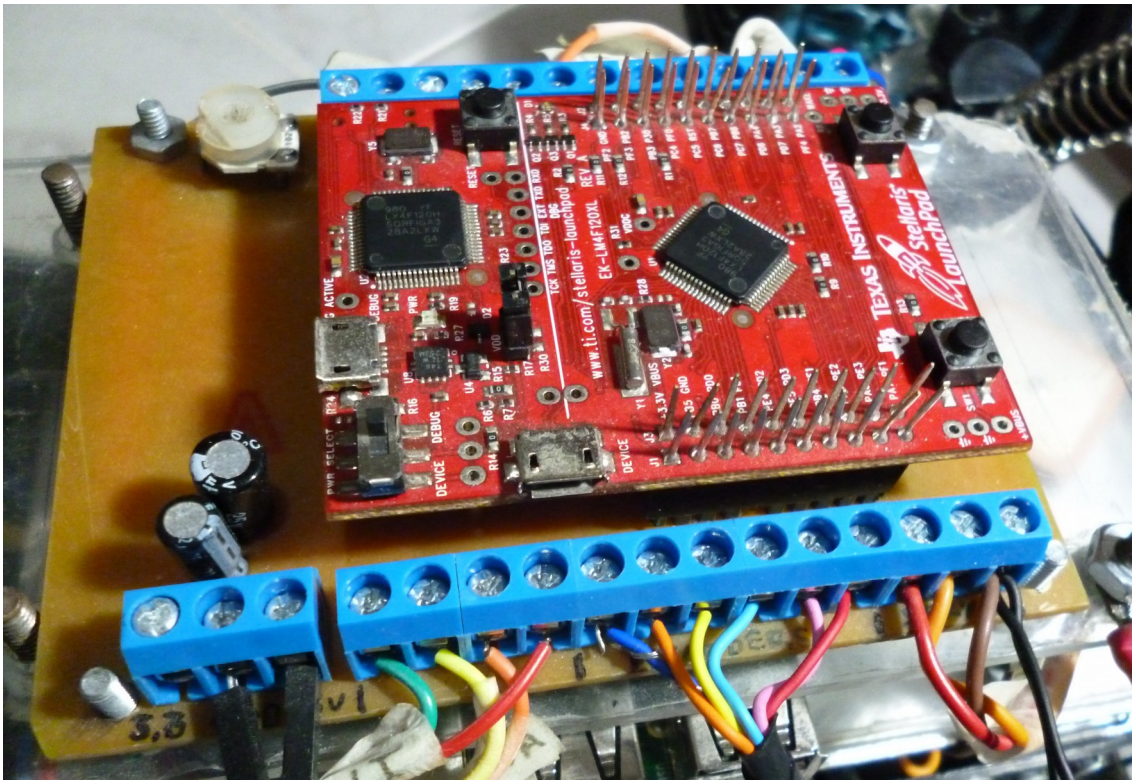

Figura 37: Montaje Tiva C

En la siguiente imagen vemos el montaje de la Rapberry Pi.

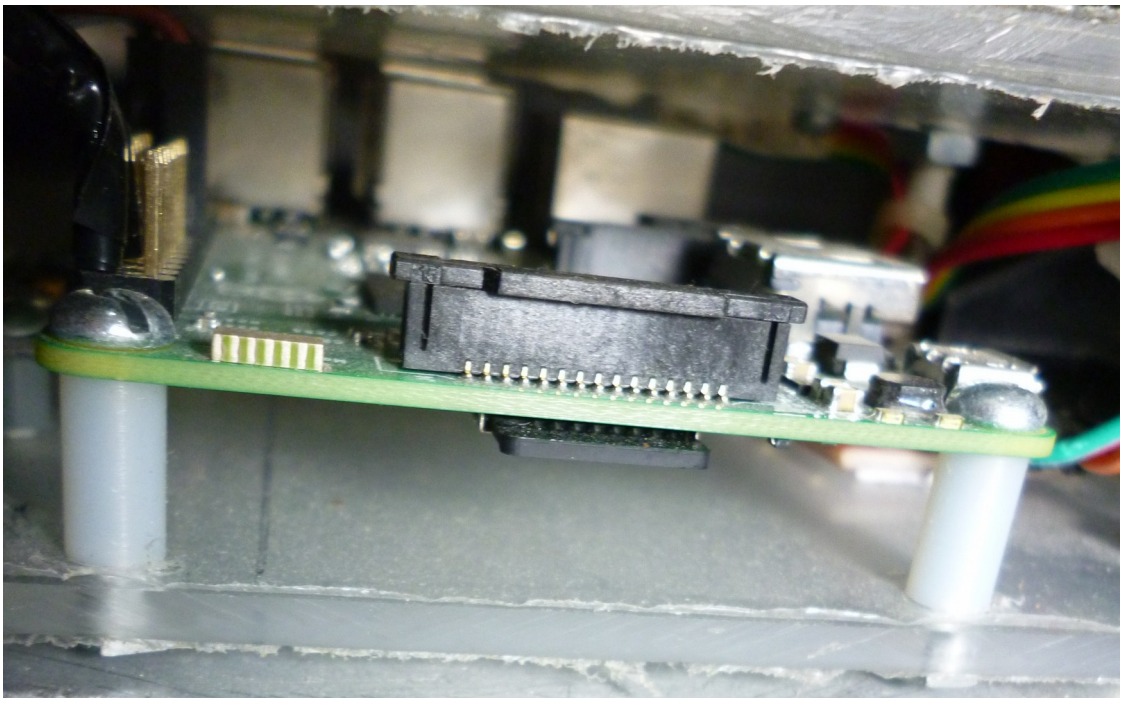

Figura 38: Montaje Raspberry Pi

En la siguiente imagen podemos ver la placa central de los encoders, se observa como fue cubierta con una placa de material conductor de forma de aislarla del ruido electromagnético.

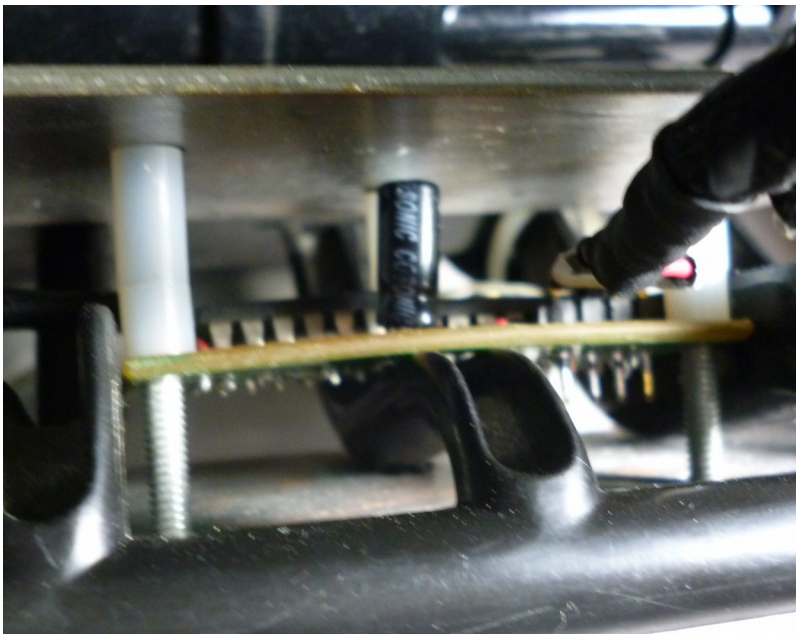

Figura 39: Montaje placa central de los encoders

A continuación podemos ver en detalle el montaje de la placa del encoders con el patrón de franjas negras y blancas en el interior de la rueda

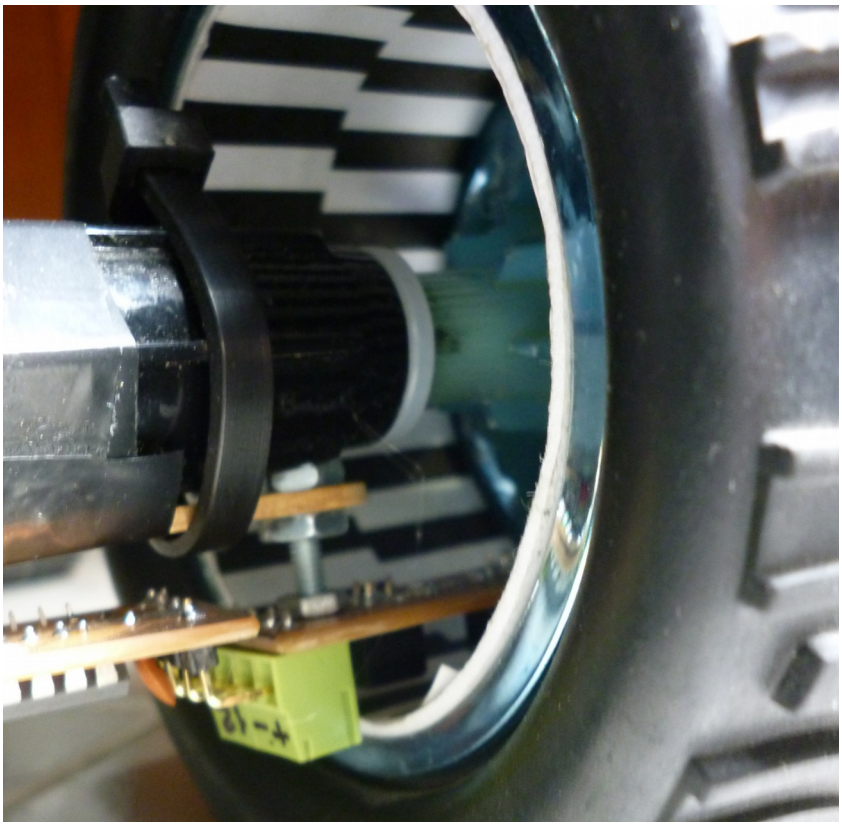

Figura 40: Montaje encoders y patron de franjas negras y blancas

En la siguiente imagen podemos ver el montaje de la batería y del ESC.

Figura 41: Montaje batería y ESC

Para finalizar vemos en la siguiente imagen la implementación completa del vehículo.

Figura 42: Montaje Final del Vehículo

Capítulo 4: Análisis de Costos

Los costos de los materiales utilizados en el proyecto se presentan en la siguiente tabla:

Material	Costo [us$]
TI Tiva C Launchpad	12.99
LSM6DS1	24.5
GPS Emlid Reach	265
Placas de Circuito impreso	10
Varillas metálicas	2.4
Planchas de acrílico	1.5
Auto a escala 1:8	195
Raspberry Pi	40
Insumos varios	5.5
Total	**556,89**

Tabla 3: Tabla de costos

Para analizar los honorarios de la persona que llevó adelante el proyecto, hay que considerar el tiempo dedicado. Este tiempo fue de 600 horas, que correspondió con una prolongación del proyecto de tres meses. Esto según el arancel mínimo impuesto por el colegio de ingenieros especialistas de Entre Rios, de 80 Ingenios mensuales para un ingeniero con menos de 5 años de experiencia laboral y con 1 ingenio tazado en $345, da un total de $82800. En este caso se hizo la valoración en pesos argentinos.

Considerando una taza de cambio de pesos a dólar de $42 por dólar, el costo final del proyecto asciende a u$s 2528,3.

Para una producción en serie consideraremos un costo de materiales redondeado en u$s 520. Con respecto a la mano de obra debemos tener en cuenta que el armado del kit lleva unas 8 horas, el precio final de la mano de obra en pesos sería de $1104, que en dólares equivale a u$s26,2. Por lo que el costo final por vehículo es de unos u$s 556. Vendiendo cada unidad a un precio de u$s750, la amortización de los costos del diseño se lograría luego de vender 14 unidades.

Capitulo 5: Discusión y Conclusión

El vehículo logrado cumple las expectativas iniciales, ya que se logró un kit cuyo control y electrónica a bajo nivel se encuentran resueltos. El posible precio de venta de u$s750, es un precio verdaderamente competitivo, si lo comparamos con kits como el “Lego Mindstorm” cuyo costo asciende a aproximadamente unos u$s1000.

Entre algunas mejoras que podrían aplicar al proyecto deberíamos mencionar el uso de un control con realimentación para el sistema de dirección del vehículo. También se podrían conseguir mejores resultados en la medición de velocidad implementando modulación por ancho de pulso sobre el led del sensor infrarrojo reflectivo. Podrían hacerse pruebas para evaluar el verdadero costo computacional del filtro Kalman complementario para utilizarlo como algoritmo de fusión de datos.

Capítulo 6: Literatura Citada

(1) Madgwick S.O.H., et al. “Estimation of IMU and MARG orientation using a gradient descent algorithm”, *IEEE International Conference on Rehabilitation Robotics (2011).*

(2) Mahony R., et al. “Nonlinear Complementary Filters on the Special Orthogonal Group”, *IEEE Transactions on automatic control, vol 53, no. 5, (*2008).

(3) Yang Deng, et al. “Complete Tri-Axis Magnetometer Calibration with a Gyro Auxiliary”, *Sensors Journal*, *vol 17* (2017). doi:10.3390/s17061223.

(4) Cannon Mark , “C21 Model Predictive Control” Recuperado de http://www.eng.ox.ac/. (2017).

(5) Burgos E. S., et al, “Implementación en lenguaje C de un algoritmo para procesamiento de flujos de datos NMEA-0183”, “*CASE 2016 Libro de trabajos en modalidad Foro Tecnológico y Posters”.(2016)*

(6) Madgwick Sebastian O. H. “An efficient orientation filter for inertial and inertial/magnetic sensor arrays” (2010).

(7) Moving Average Filter. http://en.wikipedia.org/wiki/Moving_average#Exponential_moving_average , consultado última vez en Febrero de 2019.

(8) µAO-MPC: microcontroller Applications Online Model Predictive Control, sitio web: http://ifatwww.et.uni-magdeburg.de/syst/muAO-MPC/, consultado última vez en Febrero de 2019.