_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISEÑO DE UN ROBOT DE DOS RUEDAS AUTOESTABILIZADO A CONTROL REMOTO

Miguel Alós Verdú

Tutor: Vicente Torres Carot

Cotutor: Ángel Héctor García Miquel

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2015-16

Valencia, 23 de noviembre de 2015

Resumen

El propósito de este proyecto es el diseño e implementación de un robot autónomo de dos ruedas autobalanceado basado en el modelo de un péndulo invertido. El robot incluirá sensores para obtener información de su inclinación respecto al eje normal al plano. Para este proposito se utiliza una unidad de medición inercial que incorpora un acelerómetro y un giroscopio. Estos sensores proporcionan datos que necesitan de un procesado previo y un filtrado paso bajo para eliminar el ruido, lo cual permite obtener un ángulo de inclinación estimado con más fiabilidad.

El control de los motores se realiza a partir de controladores PID implementados sobre un microcontrolador. Éstos se realimentan del ángulo de inclinación sensado y la velocidad media aplicada a los motores. El único parámetro a introducir para su funcionamiento es la velocidad deseada, que es 0 para el balanceo y valores inferiores o superiores para el movimiento en una dirección.

Se ha desarrollado una aplicación para dispositivo móvil Android que será la encargada de controlar los movimientos del robot mediante conexión Bluetooth. Además dispondrá de controles para cambiar los parámetros de los controladores PID y una gráfica con información de las variables de los mismos.

Resum

El propòsit d'aquest projecte és el disseny i la implementació d'un robot autònom de dues rodes autobalancejat basat en el model d'un pèndul invertit. El robot inclourà sensors per a obtenir informació de la seua inclinació respecte a l'eix normal al pla. Per a aquest propòsit s'utilitza una unitat de mesura inercial que incorpora un acceleròmtre i un giroscopi. Aquestos sensors proporciones dades que necessiten d'un processat previ i un filtrar pas baix per a eliminar el soroll, el que permet obtenir un angle d'inclinació estimat amb més fiabilitat.

El control dels motors es realitza mitjançant controladors PID implementats sobre un microcontrolador. Aquests es realimenten de l'angle d'inclinació sensat i la velocitat mitjana aplicada als motors. L'únic paràmetre a introduir per al seu funcionament és la velocitat desitjada, sent 0 per al balanceig i valors inferiors o superiors per a un moviment en una direcció.

Una aplicació en dispositiu mòbil Android serà l'encarregada de controlar els moviments del robot mitjançant una connexió Bluetooth. A més, disposarà de controls per a canviar els paràmetres dels controladors PID i una gràfica amb informació de les variables d'aquests.

Abstract

The purpose of this thesis is the design and implementation of a two-wheeled self-balancing robot based on the inverted pendulum model. The robot will have sensors in order to obtain information about the inclination related to the floor normal axis. For this purpose, an inertial measurement unit that includes an accelerometer and a gyroscope will be used. These sensors get data that needs some previous processing and being low pass filtered to eliminate the noise. That will allow getting a better-estimated angle.

Motor control is made through PID controllers implemented on a microcontroller device. These controllers get the inclination angle and the average speed as feedback. The only parameter that needs to be applied is the desired speed, being 0 for balancing or some other value to drive backward or forward.

An Android application will be developed for controlling the robot direction and rotation through a Bluetooth connection. Furthermore, it will include options to change the PID controller parameters and a chart with valuable data about the PID variables.

Índice

Capítulo 1. Introducción

El auge en el diseño de vehículos autobalanceades ha ido en aumento desde que en 2001 el inventor estadounidense Dean Kamen inventara el Segway. El Segway es un vehiculo de transporte personal autobalanceado y giroscópico, lo cual le permite tener una gran maniobrabilidad y ser capaz de sortear obstáculos con mayor facilidad que otros vehículos en un entorno urbano. Utiliza motores eléctricos y silenciosos, que hace del Segway un vehículo más respetuoso con el medio ambiente. Es capaz de alcanzar una velocidad de aproximadamente 20km/h.

Figura 1.1 Segway en el museo del robots de Nagoya

Las primeras unidades de Segway se vendieron por mas de 100.000$, un precio bastante superior a los 4.000$ o 5.000$ por los que se pueden adquirir ahora mismo. Aun así es un precio bastante elevado para obtener un gran número de consumidores particulares.

Figura 1.2 Airwheel

Figura 1.3 Hoverboard

En los últimos años han aparecido otros vehículos de transporte personal autobalanceados como son el Hoverboard o el Airwheel, que se pueden obtener por unos 300$ y 1000$ respectivamente. El Airhweel además tiene la particularidad de autobalancearse solo con una rueda. Este decremento de precios podría hacer posible, en un futuro no muy lejano, que su uso como transporte urbano fuese mas extendido.

El desarrollo de vehículos autobalanceados en miniatura también ha sido un fenómeno al alza en los últimos tiempo y factores como la aparición de plataformas hardware y software libre junto con la filosofía DIY (Do It Yourself) son algunos de los responsables de este suceso. El autobalanceo de este tipo de vehículos es posible mediante su correcta sensorización y procesado de la información, así como con el desarrollo de un sistema de control robusto. Es por esyo que en la construcción de este tipo de vehículos se ven implicadas diversas disciplinas como son la electrónica, mecánica, programación y teoría de control.

Un robot de dos ruedas autoestabilizado se podría definir como un péndulo invertido (Figura 1.4) con una base móvil el cual queremos mantener en posición horizontal, en la que tendrá una posición de equilibrio inestable. Esta inestabilidad debe ser compensada con el movimiento de

la base del péndulo que se habrá de mover con una velocidad y aceleración dependiente del ángulo en el que se encuentre el péndulo respecto de su posición de equilibrio.

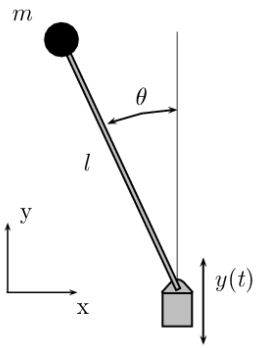

Figura 1.4 Péndulo invertido

En el caso de que lo que busquemos no sea una estabilidad alrededor de la posición de equilibrio sino un movimiento de la base en una dirección determinada, el objetivo será intentar mantener siempre un ángulo de inclinación sin llegar a la posición de equilibrio con el movimiento continuo de la base.

Capítulo 2. Objetivos del TFG

El objetivo de este trabajo es diseñar y construir un robot autobalanceado. Sabiendo cual es el problema a tratar se pueden definir los siguientes objetivos a lograr durante la realización de este trabajo:

- Construir una plataforma a semejanza de un péndulo invertido. Se construirá en madera de contrachapado y deberá tener espacio suficiente para la integración de todos los módulos hardware necesarios.

- Estudiar los módulos hardware necesarios para el funcionamiento del sistema. Se necesitará un microcontrolador capaz de realizar las tareas, sensores para adquirir el ángulo de inclinación respecto al eje normal al plano y un módulo de comunicación para realizar un sistema de control remoto. Además necesitaremos baterías para alimentar el sistema.

- Obtener el ángulo de inclinación de la plataforma en tiempo real y con precisión. Se utilizará una unidad de medida inercial. Se hará un estudio de los datos recibidos y el procesamiento necesario para obtener la información deseada.

- Desarrollar un controlador para el movimiento de la base de manera que consiga mantener la posición deseada y se evite la desestabilización total del péndulo. Se estudiarán los controladores electrónicos en general y se profundizará en el control PID, que será implementado.

- Establecer una comunicación con la plataforma mediante otro dispositivo y así poder lograr un control remoto que permita interactuar con la plataforma. Se utilizará la tecnología Bluetooth para establecer la comunicación con un dispositivo móvil con sistema operativo Android.

- Desarrollar una aplicación para el dispositivo móvil. Se deberá diseñar un protocolo de comunicación entre el dispositivo móvil y el robot. La aplicación tendrá funcionalidades para controlar remotamente el robot, configurar los parámetros de los controladores e información visual acerca de algunas variables interesantes.

Capítulo 3. Módulos Hardware

El robot ha sido construido a partir de la integración de distintos módulos hardware encargados de realizar las tareas necesarias para la adquisición de datos, procesamiento, movimiento y comunicación. A continuación se presenta la lista de los módulos implementados:

- Microcontrolador integrado en plataforma Arduino UNO.
- Unidad de medición inercial (IMU) MPU-6050
- Módulo Bluetooth HC-05
- Driver para motores DC
- 2 Motores DC de 12V
- Dispositivo móvil
- Batería

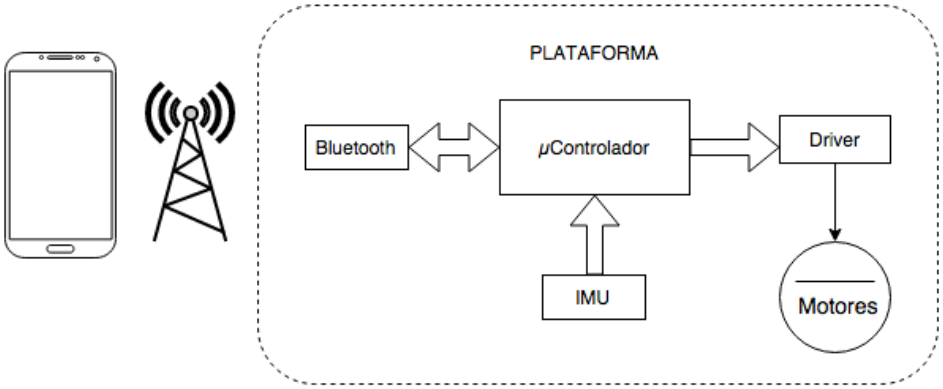

Figura 3.1 Esquema del sistema completo

Estos módulos se colocarán sobre una plataforma realizada en madera a modo de péndulo invertido para conseguir la autoestabilización.

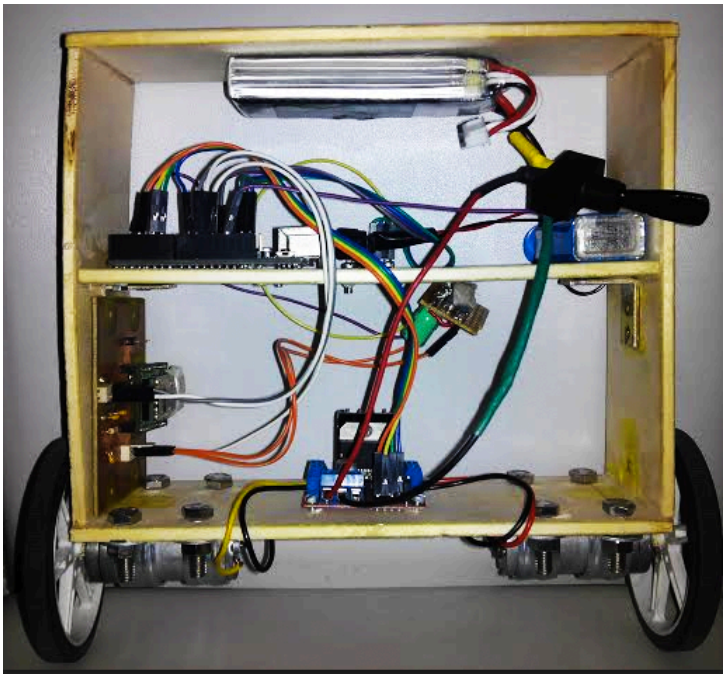

Figura 3.2 Plataforma con los módulos instalados

3.1 Microcontrolador

El microcontrolador es el cerebro de nuestro sistema. Se va a encargar de adquirir los datos necesarios para el conocimiento de la inclinación, procesarlos y controlar los actuadores (motores DC, en este caso). Además va a establecer la comunicación a través del módulo Bluetooth con el móvil que va a servir como control remoto.

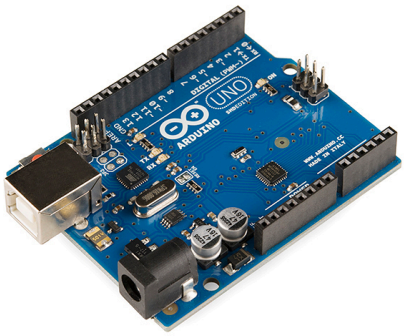

Figura 3.3 Placa microcontroladora Arduino UNO.

Los requerimientos del microcontrolador para la interfaz con los demás módulos son los siguientes:

- 1 puerto serie para la comunicación con el módulo Bluetooth
- Conexión I2C para la interfaz con la unidad de medición inercial
- 4 puertos con salida PWM para regular la velocidad de los 2 motores en las 2 direcciones.
- Suficiente velocidad de proceso para el control de la plataforma autoestabilazada

Arduino es una plataforma de hardware libre basada en una placa formada por el microcontrolador y todos los elementos necesarios para su funcionamiento y programación, así como pines I/O para la conexión de periféricos.

La placa escogida es el modelo UNO ya que dispone de un tamaño reducido y cumple todos los requerimientos del proyecto.

Las características principales del microcontrolador que lleva el Arduino UNO (ATmega328p) son las siguientes:

- Frecuencia de reloj a 16MHz
- Voltaje de operación 5V
- 14 pines de E/S digitales. 6 de los cuales proporcionan señal PWM
- 6 pines analógicos. ADC de 10bits.
- 2 pines aptos para interrupciones programadas
- 20mA de corriente por pin
- Comunicación: UART (TTL), SPI, I2C.
- Comunicación USB con el ordenador mediante un chip USB-to-serial
- 3 temporizadores con interrupciones programables

El entorno de desarrollo utilizado para la programación será el Arduino IDE.

3.2 Unidad de Medición Inercial. IMU

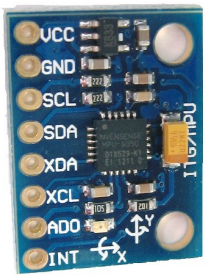

Figura 3.4 Módulo basada en el IMU digital MPU6050

La unidad de medición inercial o IMU (del inglés Inertial Measurement Unit) es un sensor de tipo MEMS (Microelectromechanical system o sistema micro electromecánico) que contiene un acelerómetro, un giroscopio y, en ocasiones, un magnetómetro. Estas unidades son ampliamente utilizadas en vehículos que necesitan un sistema de guiado inercial como son las aeronaves o submarinos. Asimismo, también están siendo utilizadas en dispositivos móviles como asistencia de la navegación GPS o simplemente como un método más de control del dispositivo.

Para el propósito de averiguar el ángulo de inclinación de la plataforma autoestabilizada solo necesitaremos un IMU que incorpore acelerómetro y giroscopio. Se usará una placa basada en el IMU digital InvenSense MPU-6050 debido a que cumple todos los requerimientos y su precio es bajo.

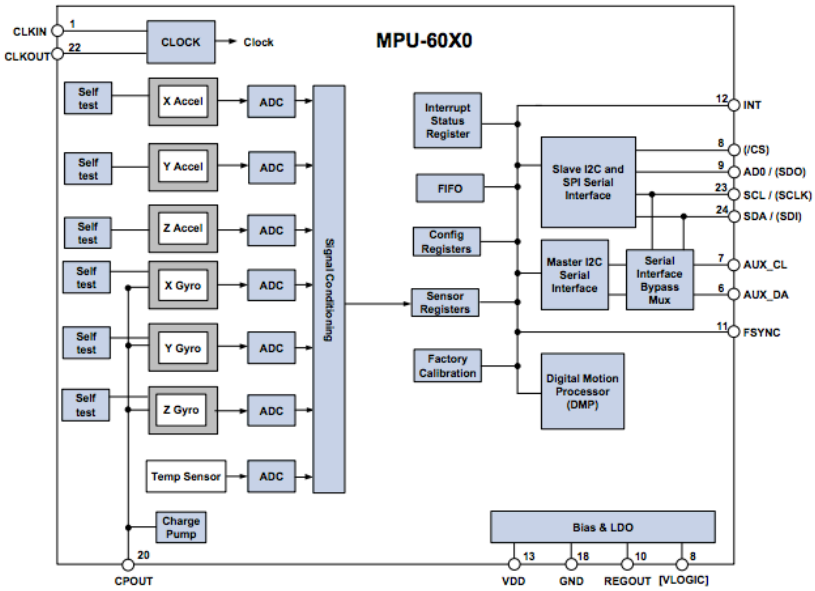

Figura 3.5 Diagrama de bloques del MPU-60X0

El MPU-6050 cuenta con las siguientes características:

- 6 grados de libertad. Ejes [x,y,z] en acelerómetro y giroscopio.
- Conversión digital con 16 bits de resolución.
- Fondo de escala del acelerómetro programable entre ±2g y ±16g
- Fondo de escala del giroscopio programable entre ±250º/s y ±2000º/s

- Comunicación I2C.
- Alimentación a 3.3V pero tolera 5V.
- Sensor de temperatura integrado,
- Procesador DMP (Digital Motion Processor) capaz de procesar las medidas obtenidas por el acelerómetro y giroscopio para darnos información útil. Sin embargo, InvenSense no da la documentación necesaria para su uso.

3.2.1 Acelerómetro

El acelerómetro se encarga de medir la fuerza que produce la aceleración en una determinada dirección. En posición estable vamos a tener una aceleración en el eje z (Suponiendo que tenemos colocado el sensor de forma que el eje z sea normal al plano) igual a la fuerza de gravedad. En un caso ideal, en el que la única fuerza que afectara al acelerómetro fuera la fuerza de gravedad, podríamos obtener el ángulo de inclinación sin necesidad del giroscopio.

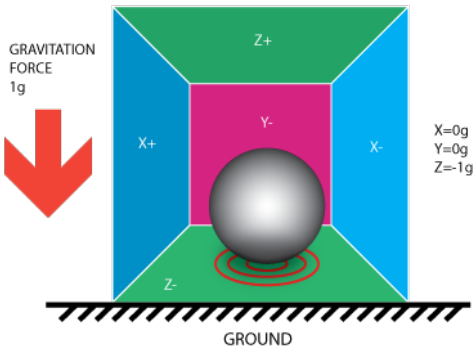

Figura 3.6 Modelización del acelerómetro con aceleración en eje z

Un movimiento del acelerómetro va a suponer que la aceleración que antes se tenía exclusivamente en el eje normal al plano pase a tener unas componentes distintas iguales a la aceleración en ese instante.

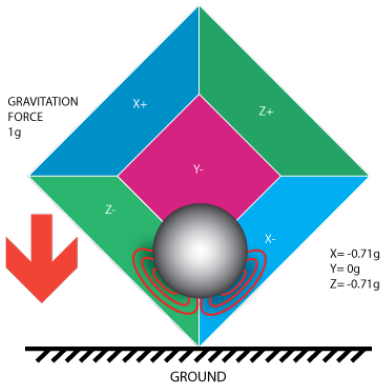

Figura 3.7 Modelización de un acelerómetro con aceleración en 2 componentes

En los sensores de tipo MEMS los acelerómetros se suelen construir con una estructura de tipo peine como se puede ver en la figura 3.8. En esta estructura hay una parte móvil de masa conocida y otra fija entre las cuales se crean capacidades. El rango de fondos de escala de este tipo de acelerómetros puede llegar a los cientos de g.

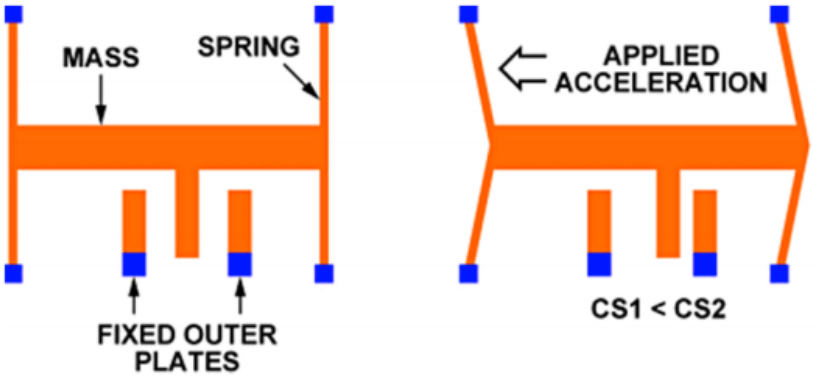

Figura 3.8 Acelerómetro tipo MEMS

Cuando existe una aceleración, las capacidades entre las 2 estructuras cambiarán y será posible medir el cambio de distancia entre ellas. Midiendo este cambio de distancia de la masa tendremos un valor de la fuerza en ese instante y con lo cual podremos obtener la aceleración con la segunda ley de Newton:

$$F = m * a = m * g$$

3.2.1 Giroscopio

Un giroscopio se compone de un eje perpendicular al plano alrededor del cual rota rápidamente un cuerpo simétrico relativamente pesado y tiende a vencer cualquier fuerza que intente desestabilizarlo de dicho plano. En un sensor tipo MEMS el giroscopio no es realmente como en la figura 3.9 pero se basa en el mismo principio.

El giroscopio nos va a proporcionar una medida de aceleración angular en cada uno de los ejes. Esta medida será importante para obtener el ángulo de inclinación porque va a permitir, mediante un procesado previo, eliminar el ruido del acelerómetro y obtener un valor estimado de la inclinación mucho más exacto.

Figura 3.9 Giroscopio

Los MEMS se realizan a partir de estructuras vibrantes. Estas estructuras vibran siempre en la misma dirección y por consecuente cualquier cambio se puede medir.

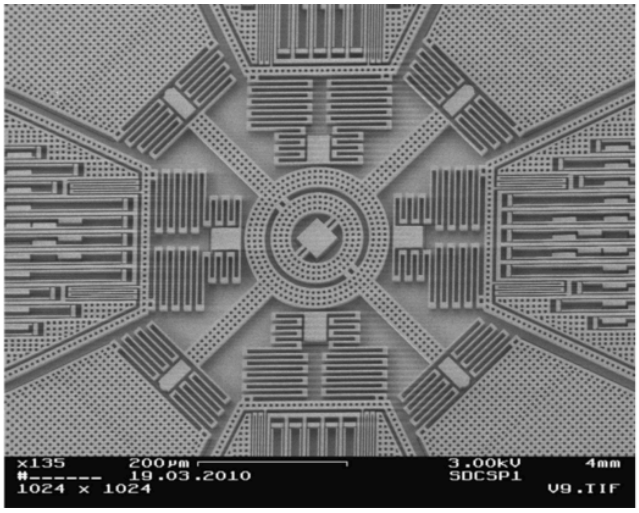

Figura 3.10 Giroscopio MEMS

En la figura 3.10 se puede ver un giroscopio de tipo MEMS hecho a partir de cuatro masas que vibran en direcciones opuestas con la misma amplitud. Cuando existe una rotación, el efecto coriolis produce una fuerza en dirección ortogonal a la rotación y dirección vibración de las masas, forzando a la estructura a un movimiento secundario en esta dirección que nos servirá para medir la velocidad de rotación

3.3 Módulo de Comunicación

Existen diferentes alternativas para realizar la comunicación con un sistema de control remoto. Por ejemplo, los módulos Zigbee utilizan un protocolo de comunicación eficiente y su consumo es realmente bajo, además ofrecen múltiples configuraciones para realizar diferentes topologías de red. Sin embargo su precio es elevado y no permiten una conexión directa con un dispositivo móvil.

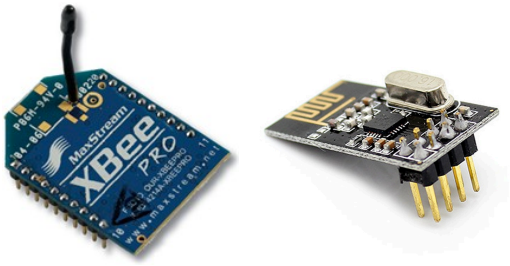

Figura 3.11 Módulo ZigBee (izquierda) y nRF2401L (derecha)

La alternativa económica a los módulos Zigbee son los módulos nRF2401L. A pesar de su bajo precio (alrededor de los 2€) también se descartan por no permitir una conexión directa con el dispositivo móvil.

Los módulos Wi-Fi permitirían una conexión directa. El módulo ESP8266 cuesta alrededor de los 2€, por lo tanto es una opción a considerar. Además existe la posibilidad de realizar tanto una conexión punto a punto, como a través de un router local o incluso por internet.

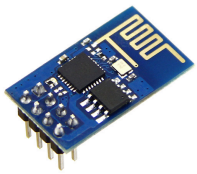

Figura 3.12 Módulo Wi-Fi ESP8266

Market Name	Wi-Fi™	Bluetooth™	ZigBee™
Underlying Standard	802.11b	802.15.1	802.15.4
Application Focus	Web, Email, Video	Cable Replacement	Monitoring & Control
Battery Life	Hours	Days	Years
Enumeration Latency	Up to 3 sec	Up to 10 sec	30ms
Network Size	32	7	Up to 65536
Bandwidth (K bits/s)	11,000+	720	250
Range (meters)*	1 - 100	1 - 10+	1 - 100+
Network Architecture	Star	Star	Star, Tree, Mesh
Optimized For	Speed	Low cost, Convenience	Reliability, Low power, Low cost, Scalability

Figura 3.13 Comparativa Wi-Fi, ZigBee y Bluetooth

Un módulo con tecnología Bluetooth también cumpliría los requerimientos por un precio similar al de los módulos Wi-Fi. Finalmente esta es la opción elegida por disponibilidad inmediata. Para la realización de la comunicación Bluetooth entre la plataforma autobalanceada y el control remoto (en este caso un teléfono móvil con Android), se utilizará un módulo HC-05, el cual actúa como interfaz entre el dispositivo móvil y la placa Arduino a través del puerto serie del mismo.

Figura 3.14 Modulo Bluetooth HC-05

Las características del módulo son las siguientes:

- Bluetooth V2.0+EDR con transmisión de datos a 3Mbps en la banda de 2.4GHz.
- Puerto serie (UART) con velocidad de transmisión (baud rate) programable mediante comandos AT.
- Alimentación a 3.3V. Se necesitará un conversor de 5V a 3.3V para la interfaz con Arduino.
- Antena integrada.
- Posibilidad de trabajar en modo maestro o esclavo.

La conexión a realizar va a ser punto a punto, por lo que va a ser indiferente configurarlo en modo maestro o esclavo. En el caso de querer comunicarse con diversos dispositivos y hacer una conexión de punto a multipunto se debería configurar como maestro.

El envío y recepción de datos se realizará desde el punto de vista del microcontrolador como si fuera un dispositivo de comunicación serie TTL usual.

3.4 Driver para motores DC

Como driver de corriente para los motores DC usaremos un módulo basado en el circuito integrado L298. El L298 es un circuito integrado basado en dos puentes en H, por lo tanto el módulo nos servirá para controlar los dos motores. La elección de este módulo se ha debido al bajo precio y a que es capaz de conducir mucha más corriente de la que realmente se va a necesitar.

Las características principales del L298 son:

- Hasta 46V de tensión aplicada a carga
- Hasta 2A de corriente por canal. Se puede llegar a 4A puenteando los canales.
- Voltaje de control 5V
- Salida de sensado de corriente

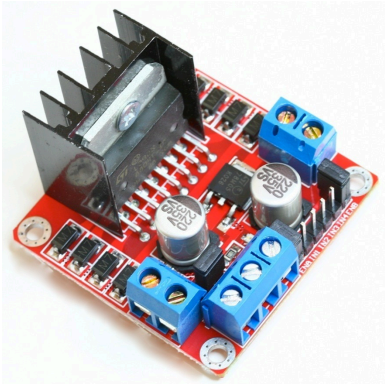

Figura 3.15 Módulo basado en el driver L298

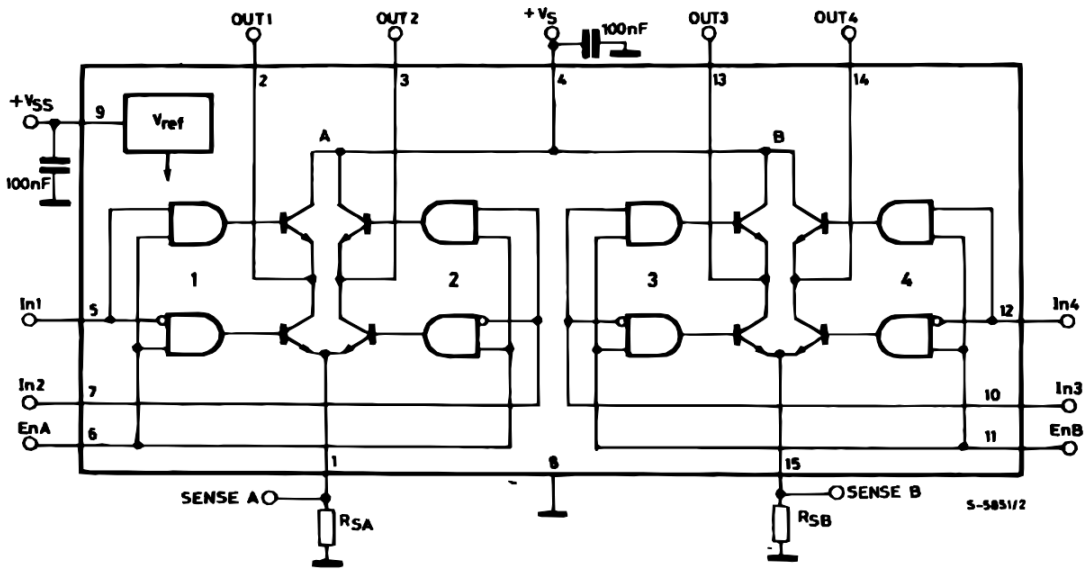

Figura 3.16 Esquemático del C.I. L298

3.4.1 El puente en H

Un puente en H es una configuración típica que permite aplicar una tensión a través de una carga y cambiar su polaridad. Esta característica permite a los motores DC girar en ambas direcciones.

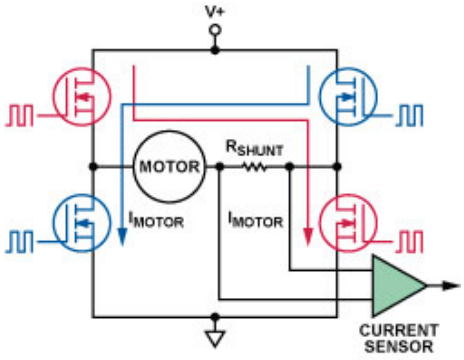

Figura 3.17 Estructura de un puente en H

El nombre de puente en H viene de la estructura del circuito (3.16) en el que está basado. Dispone de 4 interruptores controlados electrónicamente por el microcontrolador que van a permitir el paso de corriente en cualquiera de las direcciones dependiendo del par de interruptores que esté conduciendo en ese instante.

Otra de las ventajas del puente en H es que nos permite controlar la corriente por la carga que se va a traducir en un control de velocidad en nuestro motor. Este control se realiza mediante una señal PWM que actuará sobre el par de interruptores.

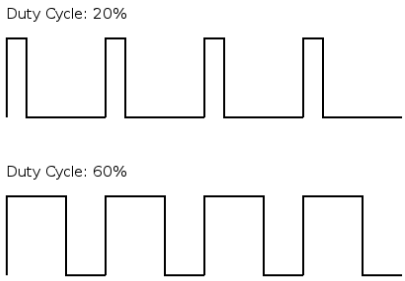

Figura 3.18 Señal PWM

Una señal PWM (Pulse Width Modulation) en una señal periódica de onda cuadrada a la cual se le modulará el ancho del pulso. Un ciclo de trabajo (Duty cycle en inglés) del 25% significa que el pulso estará a nivel alto la cuarta parte del tiempo.

Si aplicamos la señal PWM sobre el interruptor, este sólo conducirá el tiempo en el que el pulso esté a nivel alto. La corriente (o velocidad en nuestro caso) variará de forma lineal con la anchura del pulso en estado alto y en consecuencia tendremos un control total y sencillo de la velocidad.

3.5 Motores DC

Los motores DC son muy sencillos de controlar como se ha visto en el apartado anterior, solo es necesaria la aplicación de una tensión en bornes del motor. La característica principal para conseguir el autobalanceo es un motor de alto par.

El alto par se consigue a través de una caja reductora la cual limitará nuestra velocidad máxima, por lo que habrá que llegar a un compromiso entre una velocidad máxima adecuada y un par suficiente.

Figura 3.19 Motor DC

Las características de los motores utilizados son:

- 12V de voltaje máximo.
- 500mA de consumo
- Velocidad de salida del motor: 6000rpm.
- Caja reductora 1:20.
- Velocidad en el eje: 300rpm.
- Par: 1kg·cm.

3.6 Dispositivo Móvil

Se usará un dispositivo móvil con sistema operativo Android para interactuar con el robot. Además dispondrá de otras funcionalidades como son la configuración de los parámetros de los controladores e información visual de variables de interés. El dispositivo móvil utilizado será un BQ modelo Aquaris E5. Se elige por disponibilidad, ya que cumple sobradamente los requerimientos para este tipo de aplicación. Cualquier dispositivo móvil con tecnología Bluetooth debería cumplir los requisitos.

Figura 3.20 BQ Aquaris E5

Las características principales del dispositivo son las siguientes:

- Sistema operativo Android
- Procesador MediaTek Quad Core a 1.3GHz
- Conexión Bluetooth V4.0, Wi-Fi y 3G.
- GPU Mali-400 a 500MHz
- 1Gb de memoria RAM
- 16Gb de almacenamiento

El entorno de programación que se utilizará para el desarrollo de la aplicación será Android Studio.

3.7 Baterías

El hecho de que el robot sea un sistema móvil, hace estrictamente necesario el uso de batería para su alimentación. Se van a colocar 2 baterías, una para el microcontrolador, el módulo Bluetooth y la IMU; y la segunda estrictamente para los motores. Se hace de esta manera porque los motores crean un ruido de alimentación elevado, el cual puede provocar el reseteo del microcontrolador.

Figura 3.21 Pila 6LR61 de 9V

La alimentación del microcontrolador está recomendada en el rango de 7V y 15V y el consumo de corriente de éste y los otros periféricos no va a ser elevado. Se elige como alimentación una pila no recargable 6lr6l de 9V. Esta batería será suficiente para alimentar el sistema de forma ininterrumpida durante un largo tiempo.

Figura 3.22 Batería Li-Po de 3 celdas

Los motores requieren una batería de 12V y que sea capaz de hacer una entrega de corriente rápida, lo cual hará que no se produzcan grandes caídas de tensión. Las baterías que cumplen mejor estas características son las baterías de polímeros de litio (Li-Po). Las celdas de las baterías de polímeros de litio tienen un voltaje aproximado de 3.7V voltios por celda, con lo cual necesitaremos una batería de 3 celdas. La capacidad de la batería será de 1300mAh, proporcionando teóricamente alrededor de una hora de autonomía a los motores en consumo máximo.

Capítulo 4. Adquisición y Tratamiento de Datos de la IMU

La unidad de medición inercial tiene un acelerómetro que es capaz de darnos una aceleración en cada uno de los ejes del sistema cartesiano y un giroscopio que proporciona una velocidad angular, de la cual se puede obtener una rotación.

Estos datos no son útiles tal cual se leen de la IMU, ya que se necesita un ángulo de inclinación para saber el ángulo exacto respecto al plano. Sin embargo, sí que es posible procesarlos y así obtener información útil.

El objetivo es conocer el ángulo de inclinación respecto del eje z tal como se muestra en la figura 4.1. Este ángulo α será la información necesaria que utilizaremos para retroalimentar el control de velocidad de la plataforma.

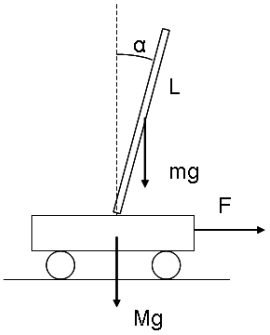

Figura 4.1. Esquema de la plataforma

A continuación se van a analizar los datos que recibimos de cada uno de los sensores y cómo se van a combinar los datos del acelerómetro para conseguir información sobre la inclinación.

4.1 Lectura de Datos de la IMU

Para obtener el valor de aceleración o la velocidad angular a partir del dato de salida de la IMU, se necesita hacer una conversión desde valor decimal (representado con 16 bits) usando el fondo de escala configurado. Esta conversión se puede realizar mediante la siguiente ecuación.

$$x = \frac{(x_{IMU} \pm offset)}{\frac{2^b}{2}} * FE = \frac{2(x_{IMU} \pm offset)}{2^b} * FE \quad (4.1)$$

donde:

x_{IMU} – Valor obtenido de la lectura de la IMU

offset – Offset medido en la calibración

b – Bits de resolución del ADC de la IMU

F.E. – Fondo de escala configurado

Podemos obtener la sensibilidad para un determinado fondo de escala con la siguiente ecuación:

$$S = \frac{\frac{2^b}{2}}{F.E.} = \frac{32768}{F.E.} \quad (4.2)$$

Para fondos de escala de ±2g y ±250º/s se obtiene una sensibilidad de 16364[valores/g] y 131[valores/(º/s)]. Esto significa que teóricamente podemos tener una resolución de la 1/16384 parte de 1g o la 1/131 parte de 1º/s. Aumentando al doble el fondo de escala de los sensores se reduce la sensibilidad a la mitad y por lo tanto perdemos resolución. Con esto se concluye que

la mayor resolución se obtiene configurando los sensores con el fondo de escala mínimo necesario para nuestra aplicación.

4.2 Configuración de la IMU

La plataforma no va a someterse grandes aceleraciones ni cambios angulares, por lo que se debe llegar a un compromiso para obtener la mayor resolución sin que la señal sature.

El método para obtener la mejor configuración será utilizar el fondo de escala de menor valor y comprobar que nuestras medidas no saturen sometiendo a la plataforma a fuerzas similares a las que soportará en funcionamiento.

La mayor resolución se obtendrá configurando el acelerómetro con un fondo de escala de ±2g. Así mismo, en el giroscopio se obtendrá configurándolo a ±250º/s. A continuación se muestran los valores obtenidos directamente del sensor sin ningún tipo de procesamiento.

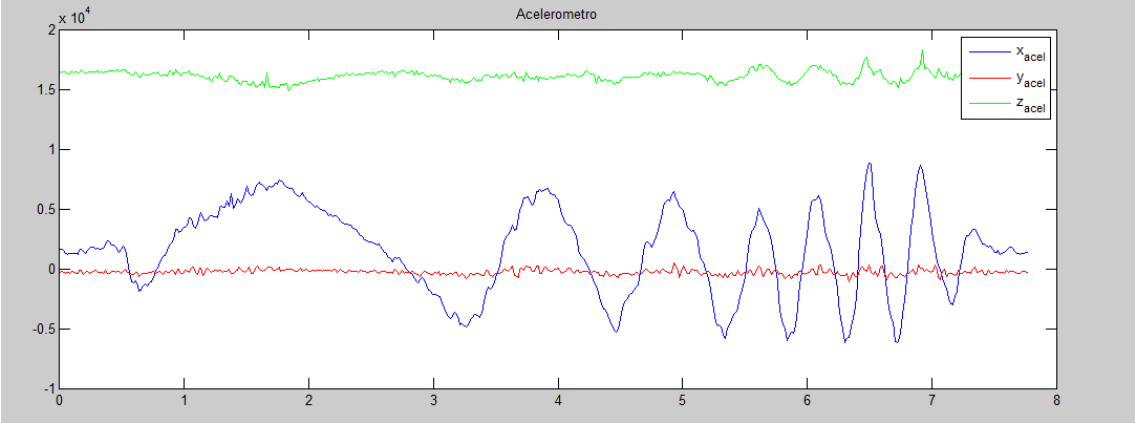

Figura 4.2. Datos obtenidos del acelerómetro

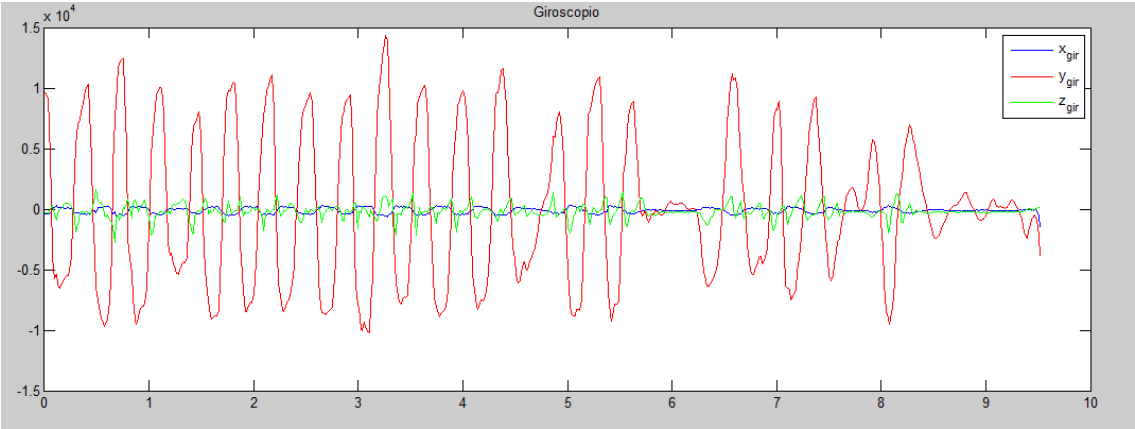

Figura 4.3. Datos obtenidos del giroscopio

De estas dos gráficas se puede concluir que la configuración con fondos de escala mínimos es la mejor. En el acelerómetro se puede ver como la mayor variación de aceleración se produce en el eje x y se encuentra en un rango aproximado de ±7000. Además, se puede ver como en el eje z una aceleración casi constante de valor 16000 aproximadamente, lo cual equivale a la fuerza de gravedad. Por otro lado, en el giroscopio, los mayores cambios de velocidad angular se dan en el eje y, y no tenemos ningún valor por encima de los 15000 (menos de la mitad del fondo de escala).

Además, por motivos de fabricación, los datos obtenidos en acelerómetro y giroscopio en una posición estática tienen un offset que se debe compensar mediante un ajuste a cero. Este ajuste a cero se puede realizar colocando la IMU en una posición estática en la que los valores que se deben obtener son conocidos, y después sumar o restar el valor de offset que observemos en la lectura.

4.3 Obtención de Información del Acelerómetro

El acelerómetro da una aceleración en unidades de g, es decir, en múltiplos y submúltiplos de la fuerza de gravedad. Supongamos que se obtiene una lectura de una aceleración en las 3 componentes tal como se muestra en la figura 4.4.

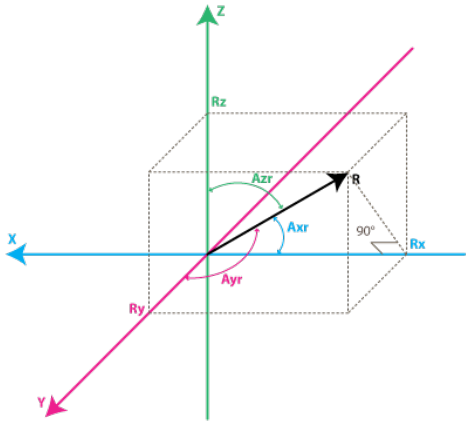

Figura 4.4. Representación de aceleración en las 3 componentes

Se definen Rx, Ry y Rz como la proyección del vector R sobre cada uno de los ejes. Podemos obtener la siguiente relación usando el teorema de Pitágoras en 3 dimensiones.

$$R = \sqrt{R_x^2 + R_y^2 + R_z^2} \quad (4.3)$$

Estos 3 valores son los que el acelerómetro va a proporcionar. Si el acelerómetro no estuviera sometido a otras fuerzas externas, el vector R sería el vector referido a la dirección de la fuerza de gravedad.

Una vez hallado R, es muy sencillo obtener los ángulos respecto a cada eje usando una relación trigonométrica.

$$A_{xr} = \cos^{-1}\left(\frac{R_x}{R}\right) \quad (4.4)$$

$$A_{yr} = \cos^{-1}\left(\frac{R_y}{R}\right) \quad (4.5)$$

$$A_{zr} = \cos^{-1}\left(\frac{R_z}{R}\right) \quad (4.6)$$

Donde [Axr, Ayr, Azr] ya son los ángulos de inclinación en cada componente.

Se ha considerado el caso ideal en el que la única fuerza actuando sobre la plataforma va a ser la fuerza de gravedad, pero esta afirmación no es cierta. El mismo movimiento del vehículo va a provocar otras fuerzas no deseadas y que por lo tanto, se considerarán como ruido. Esa es la razón por la que se necesitará combinar los datos del acelerómetro con los del giroscopio.

4.4 Obtención de Información del Giroscopio

El giroscopio mide el valor de velocidad angular, de la cual se puede obtener información del cambio de ángulo en el intervalo de tiempo entre 2 mediciones. Estos datos no van a servir para la obtención directa de un ángulo de inclinación absoluto, como se ha visto con el acelerómetro. En cambio, esta variación de ángulo que se obtiene, se va a poder procesar para obtener resultados que se podrán combinar con los datos obtenidos con el acelerómetro y así eliminar el ruido.

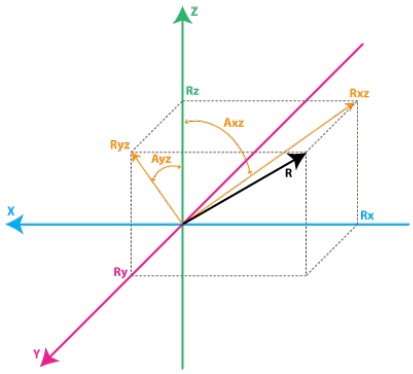

Figura 4.5. Representación de cambio angular en las 3 componentes

Se define Axz como la variación de ángulo entre el eje x y el z en el tiempo. Si en el instante de tiempo t1 el giroscopio encuentra en Axz1 y en el instante t2 está en Axz2, entonces se define Atxz

$$A_{txz} = \frac{A_{axz2} - A_{xz1}}{t2 - t1} \left[\frac{^\circ}{s}\right] \quad (4.7)$$

que será el valor que proporcionará el sensor.

Para obtener simplemente la variación en grados bastaría con multiplicar los datos obtenidos por el giroscopio por el intervalo de tiempo en el que han sido medidos.

4.5 Combinando los datos del acelerómetro y giroscopio

El acelerómetro proporciona datos para el cálculo del ángulo de inclinación, pero habrá ruido debido a las fuerzas de aceleración que produce el vehículo al moverse. El algoritmo para el combinado de los datos va a consistir en definir el ángulo de inclinación que se obtiene del acelerómetro como posición absoluta y, cuando haya movimiento, usar la información del giroscopio para hacer un filtrado y eliminar el ruido.

Se define Racel como el vector de aceleraciones en las 3 componentes

$$R_{acel} = [R_{Xacel}, R_{yacel}, R_{zacel}] \quad (4.8)$$

y su módulo

$$|R_{acel}| = \sqrt{R_{xacel}^2 + R_{yacel}^2 + R_{zacel}^2} \quad (4.9)$$

para obtener el vector aceleración normalizado.

$$R_{acel_n} = \frac{R_{acel}}{|R_{acel}|} \quad (4.10)$$

También se define el vector Rest que será la estimación de la fuerza de gravedad en cada eje eliminando el ruido

$$R_{est} = [R_{estx}, R_{esty}, R_{estz}] \quad (4.11)$$

En la primera iteración del procesado, el vector estimado va ser igual al vector obtenido directamente del acelerómetro, ya que no tenemos ningún dato previo. Se van a definir las variables obtenidas en la iteración anterior como x(t-1) y las variables en el periodo actual como x(t). Siendo x la variable a utilizar.

Observando la figura 4.5, el giroscopio solo nos da el ratio que se ha definido como Atxz. Se puede obtener la variación de ángulo Axz

$$A_{xz} = A_{txz} * t \quad (4.12)$$

siendo t el intervalo entre la obtención de las 2 medidas.

Se puede observar que

$$A_{xz} = \tan^{-1}\left(\frac{Rx}{Rz}\right) \quad (4.13)$$

Si se conocen los valores estimados de la iteración anterior podemos decir que

$$A_{xz}(t-1) = \operatorname{atan}\left(\frac{R_{estx}(t-1)}{R_{estz}(t-1)}\right) \quad (4.14)$$

La función atan(x) devuelve un valor en un rango [-90º,90º], sin embargo se necesita que el valor de Axz esté entre [-180º,180º]. Este propósito se conseguirá con la función atan2(x). Por lo tanto se redefine

$$A_{xz}(t-1) = \operatorname{atan2}\left(\frac{R_{estx}(t-1)}{R_{estz}(t-1)}\right) \quad (4.15)$$

y por lo tanto

$$A_{xz}(t) = A_{xz}(t-1) + A_{txz} * t \quad (4.16)$$

Sin embargo, la variable que se quiere obtener para que sea análoga y poder combinarla con la medida del acelerómetro es Rx. Se define un vector Rgir que contiene las variables que deseamos obtener

$$R_{gir} = \left[R_{xgir}, R_{ygir}, R_{zgir}\right] \quad (4.17)$$

Al haber normalizado el vector Racel podemos asumir que

$$\left|R_{gir}\right| = 1 \quad (4.18)$$

Para obtener Rxgir seguiremos la siguiente deducción

$$R_{xgir} = \frac{R_{xgir}}{1} = \frac{R_{xgir}}{\sqrt{R_{xgir}^2 + R_{ygir}^2 + R_{zgir}^2}} = \frac{\frac{R_{xgir}}{\sqrt{R_{xgir}^2 + R_{zgir}^2}}}{\frac{\sqrt{R_{xgir}^2 + R_{ygir}^2 + R_{zgir}^2}}{\sqrt{R_{xgir}^2 + R_{zgir}^2}}} =$$

$$= \frac{sen(A_{xz})}{\frac{\sqrt{1 + R_{ygir}^2}}{\sqrt{R_{xgir}^2 + R_{zgir}^2}}} = \frac{sen(A_{xz})}{\frac{\sqrt{(1 + R_{ygir}^2) * R_{zgir}^2}}{\sqrt{(R_{xgir}^2 + R_{zgir}^2) * R_{zgir}^2}}} \quad (4.19)$$

Observamos que

$$\frac{R_{zgir}}{\sqrt{R_{xgir}^2 + R_{zgir}^2}} = \cos(A_{xz}) \qquad (4.20)$$

y

$$\frac{R_{ygir}}{R_{zgir}} = \tan(A_{yz}) \qquad (4.21)$$

Sustituyendo

$$R_{xgir} = \frac{\sin(A_{xz})}{\sqrt{1 + \cos(A_{xz})^2 * \tan(A_{yz})}} \qquad (4.22)$$

y análogamente

$$R_{ygir} = \frac{\sin(A_{yz})}{\sqrt{1 + \cos(A_{yz})^2 * \tan(A_{xz})}} \qquad (4.23)$$

Con todo este proceso se ha conseguido obtener Rxgir y Rygir en función de los datos que se pueden obtener directamente del giroscopio. Faltaría obtener Rzgir que se puede obtener a partir de la ecuación 4.11. El signo supondremos que es igual al de la estimación que se ha hecho en la iteración anterior, que al tener en cuenta también la medida del acelerómetro será una aproximación bastante fiable.

$$R_{zgir} = signo(R_{zgir}) * \sqrt{1 - (R_{xgir}^2 + R_{ygir}^2)} \qquad (4.24)$$

donde

$$signo(R_{zgir}) = signo(R_{estz}(t-1)) \qquad (4.25)$$

Hay que tener cuidado cuando el valor de Rzgir sea cercano a 0, ya que observando la ecuación X.X se puede ver que se está usando como referencia para calcular Axz y Ayz en el dominador. Este efecto produciría una falta de precisión al trabajar con números de valor muy elevado en coma flotante. La solución a este problema es reasignar Rgir al valor de Rest obtenido en la iteración anterior.

4.6 Filtrado

El filtrado de la información es lo que se usará para obtener el vector Rest definido en el apartado anterior. Se usará un filtro que tenga en cuenta los valores de los vectores Racel y Rgir y consiga eliminar el ruido.

La literatura propone el uso de filtros Kalman o filtros de media ponderada. Ambos filtros tienen un principio de funcionamiento similar, sin embargo, el filtro Kalman tiene una mayor precisión pero su coste computacional es mucho más elevado que el del filtro de media ponderada.

La particularidad del filtro Kalman es, que a partir del ruido u otras imprecisiones que se han dado en muestras anteriores en diferentes medidas de la misma variable, es capaz de obtener una salida mucho más precisa dándole un peso diferente y adaptativo a cada una de las entradas. El cálculo de la predicción previa a la obtención de la salida es lo que produce su alto coste computacional.

El filtro de media ponderada también tiene en cuenta las diferentes medidas, pero los pesos asignados a cada una no son adaptativos y dependientes de muestras anteriores, sino que se trata de valores fijos obtenidos de forma experimental. Este algoritmo obtiene una precisión menor pero es aceptable para la mayoría de las aplicaciones.

El autobalanceo del vehículo requiere que el código utilizado se pueda ejecutar en el menor tiempo posible para tener un resultado óptimo. Por este motivo se elige implementar el filtro de media ponderada y comprobar los resultados. Así se podrá concluir si su precisión será suficiente.

El filtro de media ponderada se puede definir matemáticamente de la siguiente manera:

$$R_{iest} = \frac{k_1 * R_{iacel} + k_2 * R_{igir}}{k_1 + k_2} \quad (4.26)$$

donde:

i – Componente [x,y,z] a calcular.

k_1 – Peso aplicado a la medida del acelerómetro.

k_2 – Peso aplicado a la medida del giroscopio.

Se puede simplificar esta expresión de la siguiente manera

$$R_{iest} = \frac{k_1 * R_{iacel} + k_2 * R_{igir}}{k_1 + k_2} = \frac{\frac{k_1}{k_1} * R_{iacel} + \frac{k_2}{k_1} * R_{igir}}{\frac{k_1}{k_1} + \frac{k_2}{k_1}} =$$

$$= \frac{R_{iacel} + k * R_{igir}}{1 + k} \quad (4.27)$$

Proponiéndose valores de k en un rango entre 5 y 20.

Por último, por trigonometría se obtiene el ángulo de inclinación de cada componente:

$$\alpha_x = \cos^{-1}\left(\frac{R_{xest}}{|R_{est}|}\right) \quad (4.28)$$

$$\alpha_y = \cos^{-1}\left(\frac{R_{yest}}{|R_{est}|}\right) \quad (4.29)$$

$$\alpha_z = \cos^{-1}\left(\frac{R_{zest}}{|R_{est}|}\right) \quad (4.30)$$

4.7 Resultados

A continuación se van a presentar diferentes gráficas con el ángulo obtenido a partir de la salida del acelerómetro (con ruido) y el ángulo obtenido a partir de la salida del filtro con diferentes pesos.

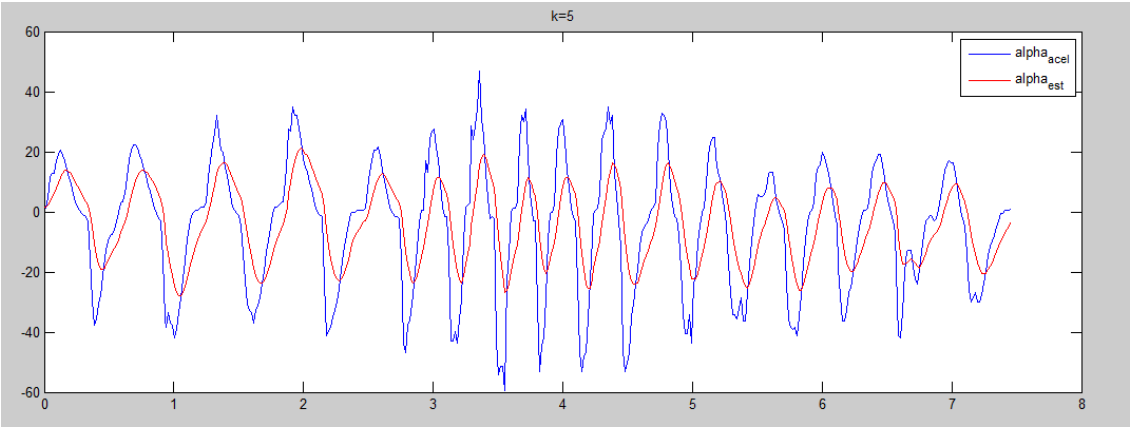

Figura 4.6. Datos filtrados con peso k=5

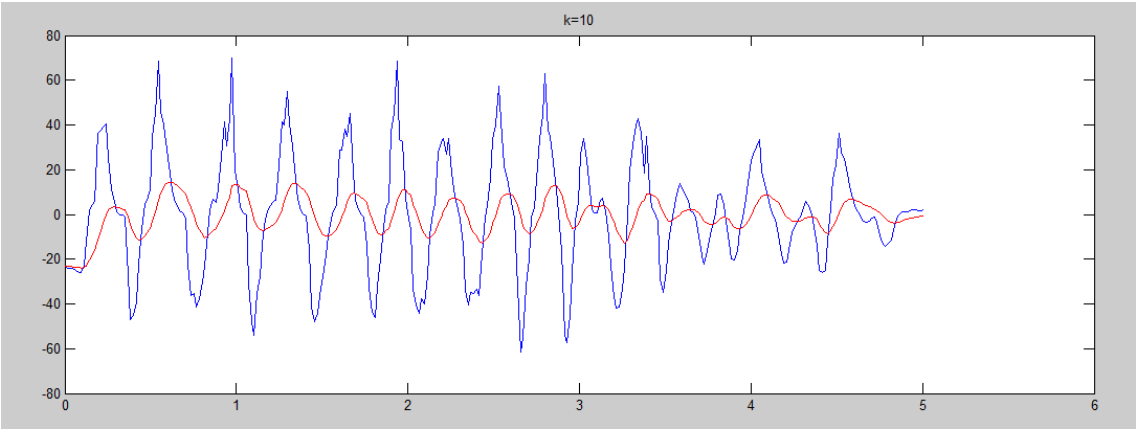

Figura 4.7. Datos filtrados con peso k=10

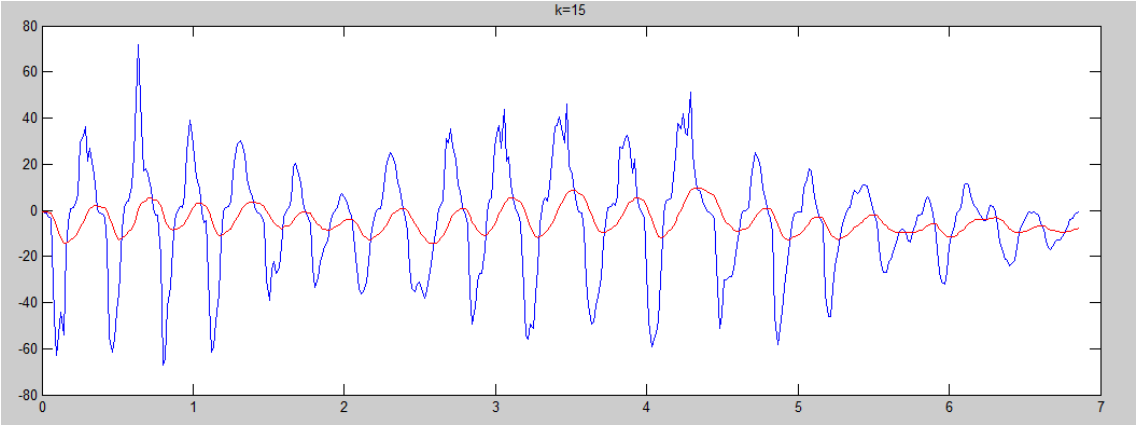

Figura 4.8. Datos filtrados con peso k=15

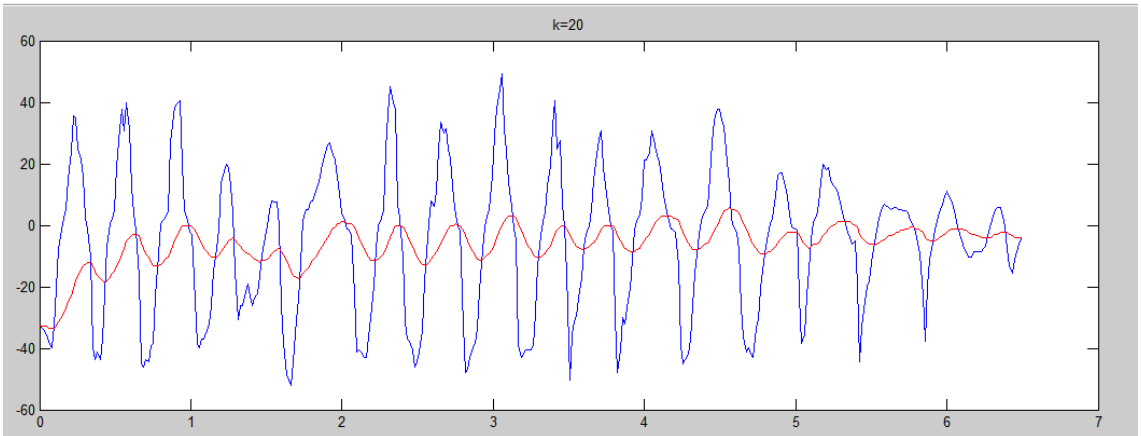

Figura 4.9. Datos filtrados con peso k=20

El factor k se puede definir como el ratio de peso que se le da a la medida del giroscopio sobre la medida del acelerómetro. El efecto del filtro de media ponderada es el de un filtro paso bajo como se puede ver en las gráficas. Aumentando el valor de k se produce un efecto de retraso de señal y con lo cual una pérdida de amplitud máximo, alejándose así de la medida real en cada instante. La medida del acelerómetro es totalmente irreal, ya que para tomar estos datos el ángulo no ha excedido de ±30º aproximadamente y, sin embargo, se ven picos de casi 80º.

Capítulo 5. Control del Robot

Un sistema de control es una serie de componentes que trabajan de manera conjunta con una cierta inteligencia para conseguir lograr un objetivo como es la automatización de un proceso. En un sistema electrónico esta inteligencia suele estar proporcionada por la circuitería o con un microcontrolador. Los componentes suelen ser sensores y actuadores que son los encargados de interactuar con el mundo físico.

El control electrónico es, por norma general, mucho más eficaz que el control humano. Esto es debido a que el control electrónico basa sus decisiones en una serie de rutinas fijas previamente prediseñadas mientras que el control humano tiene libertad de decisión. Además el tiempo de reacción es mucho más rápido cuando hablamos de sistemas de control electrónicos.

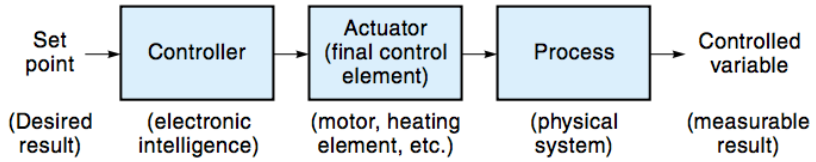

Figura 5.1 Diagrama de bloques de un sistema de control electrónico en lazo abierto.

Todo sistema de control tiene al menos dos componentes: un controlador (inteligencia) y un actuador que actuará sobre el sistema físico. El punto de control (En inglés set point) es la variable que introduciremos al controlador para que sea obtenida a la salida.

Podemos distinguir dos tipos de controladores:

- Controladores en lazo abierto
- Controladores en lazo cerrado (con retroalimentación)

Los controladores en lazo abierto tienen una topología igual a la de la figura 5.1. En un sistema de control de este tipo si, por ejemplo, se quiere rotar 30º un motor que tiene una velocidad angular de 5º/s, el controlador debería aplicar un pulso de 6 segundos para que el motor idealmente rote los 30º. El problema es que el motor tendrá una fricción que hará que su velocidad de rotación no sea exactamente de 5º/s y por lo tanto vamos a cometer un error que no se va a poder solucionar, ya que no hay manera de saber cuantos grados ha rotado el motor realmente.

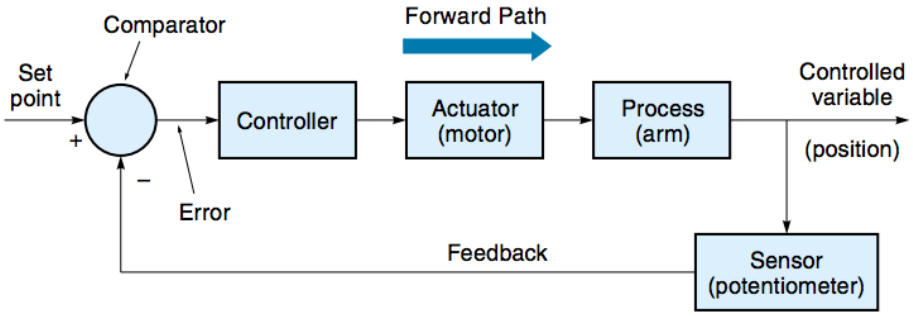

Figura 5.2. Diagrama de bloques de un sistema de control con retroalimentación

La solución a los problemas de los sistemas de control en lazo abierto consiste en sensar la variable controlada en el sistema dinámico. Al set point que hemos definido se le restara la variable sensada en el sistema dinámico creando una señal de error. Sobre esta señal de error es sobre la cual el controlador aplicará su inteligencia para el control de los actuadores.

La eficacia de un controlador se puede medir con dos parámetros:

- La respuesta transitoria. Es el periodo de tiempo entre un cambio a la entrada (set point) del controlador y el momento en el que se alcanza el estado estacionario. En la figura 5.3 esta representado como TS.
- Error de estado estacionario. Es el error que se mantiene, una vez alcanzado el estado estacionario, entre la variable de entrada (set point) y la variable sensada a la salida. En la figura 5.3 está representado como ESS

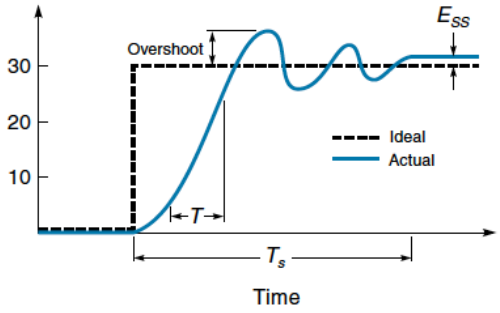

Figura 5.3 Respuesta de un controlador

5.1 El controlador PID

El controlador PID es un tipo de controlador usado en la mayoría de los sistemas automatizados. El control PID está compuesto per tres controladores que suman sus efectos: proporcional, integrador y diferencial. Matemáticamente se puede describir como:

$$\mathrm{PID_{out}} = \mathrm{K_P} * \mathrm{E(t)} + \mathrm{K_I} * \left(\int E(t)dt \right) + \mathrm{K_D} * \frac{\mathrm{dE(t)}}{\mathrm{dT}} =$$

$$= \mathrm{K_P} * \mathrm{E} + \mathrm{K_I} * \left(\sum \mathrm{E\Delta T} \right) + \mathrm{K_D} * \frac{\Delta \mathrm{E}}{\Delta \mathrm{T}} \qquad (5.1)$$

Siendo:

$\mathrm{K_P}$ – Ganancia del control proporcional.

$\mathrm{K_I}$ – Ganancia del control integrador.

$\mathrm{K_D}$ – Ganancia del control diferencial.

E – Señal de error.

$\sum \mathrm{E\Delta T}$ – Suma de los errores anteriores.

$\frac{\Delta \mathrm{E}}{\Delta \mathrm{T}}$ – Pendiente (derivada) de la señal de error.

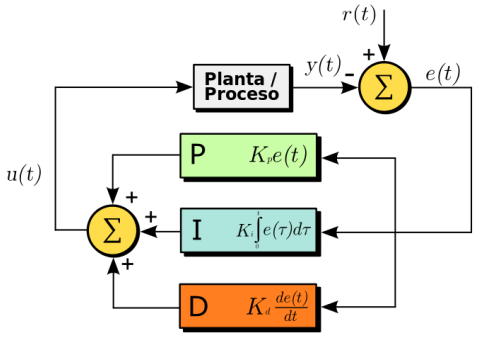

Figura 5.4 Diagrama de bloques de un controlador PID

5.1.1 El control proporcional

$$P_{out} = K_p * E \quad (5.2)$$

El control proporcional idealmente multiplicará la señal de error por una constante hasta que la señal de error sea 0. La realidad es que cuando el error sea pequeño la salida del controlador (en el caso de que solo tengamos la componente proporcional) no va a ser suficiente para mover el actuador y tendremos un error de estado estacionario grande. El caso anterior se da cuando se elige una constante K_P pequeña. En el caso de elegir una ganancia proporcional elevada el problema va a ser una oscilación continua alrededor del valor deseado como set point.

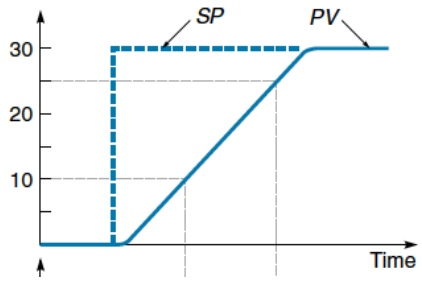

Figura 5.5 Respuesta ideal de un control proporcional

Un valor óptimo de ganancia proporcional será entre aquel que, como máximo, proporcione una pequeña sobreoscilación (nunca superior al 30%) y el que quede cerca del set point sin provocar sobreoscilaciones.

5.1.2 El control integral

$$I_{out} = K_I * \left(\sum E\Delta T\right) \quad (5.3)$$

El control integral va a multiplicar una constante K_I por la suma de todos los errores anteriores y su misión va a ser solucionar el error de estado estacionario producido por el control proporcional. Este control va a introducir sobreoscilaciones en la respuesta que tendrán que ser corregidas ya que si no será difícil alcanzar el estado estacionario. Esta será la tarea del control derivativo o diferencial.

5.1.3 El control diferencial

$$D_{out} = K_D * \frac{\Delta E}{\Delta T} \quad (5.4)$$

La derivada de la señal de error proporciona la pendiente de la misma. Esto es interesante debido a que nos va a permitir reducir la salida del controlador a medida que nos acercamos al punto deseado a la salida. Esta reducción de la salida se va a traducir en una reducción de las sobreoscilaciones provocadas por los controladores proporcional e integral. De manera contraria, si el error esta aumentando, se suma a la acción de los otros controladores hasta conseguir que empiece a disminuir.

5.2 Implementación del Control

El autobalanceado o movimiento del vehículo va a necesitar de un control de los motores (actuadores) dependiente del ángulo de inclinación, lo cual será la variable que usara en la retroalimentación. El set point para el autobalanceo es idealmente 0, pero en realidad este será un valor cercano por dos causas:

- Offset de la IMU. Aunque se hayan corregido la mayor parte de este problema seguirá habiendo un pequeño error.
- Construcción de la plataforma y ubicación de la IMU. La plataforma se ha intentado construir con la mayor exactitud pero no está asegurado que la IMU vaya a estar colocada de manera que el eje z de la misma sea completamente normal al plano. Es la causa que provocará el mayor error.

Este problema hará que el sistema necesite dos controladores. El primero para controlar la velocidad de los motores según el error en el ángulo de inclinación del vehículo y otro para corregir el error de ángulo en el cual la plataforma está perpendicular al plano.

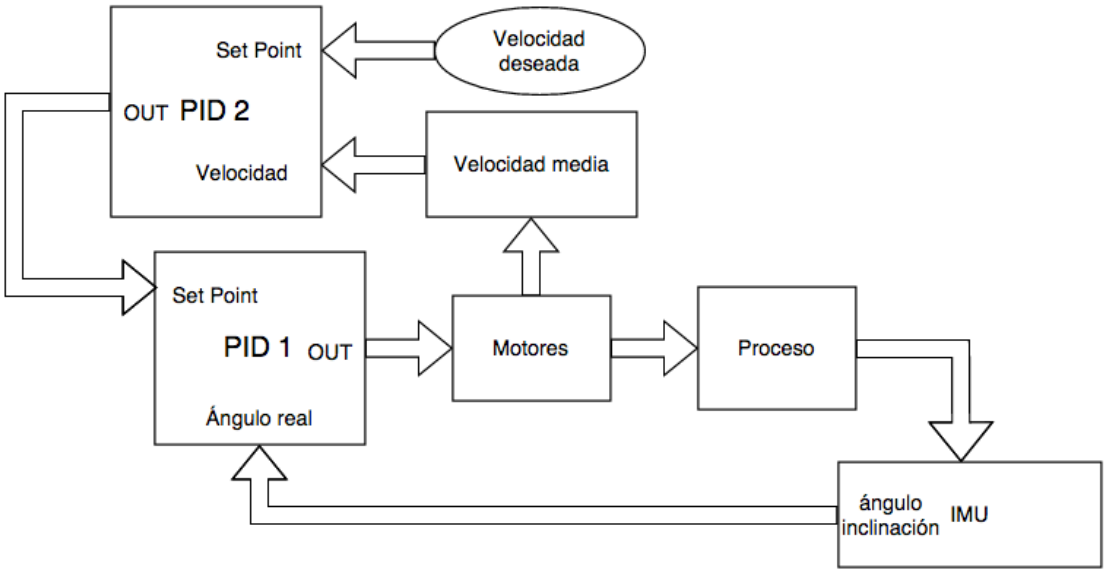

Figura 5.6 Esquema de control propuesto

Como se puede ver la en la figura 5.6 se ha propuesto un sistema con dos controladores PID. El PID 1 será el encargado de calcular la velocidad de los motores a partir del ángulo deseado y está retroalimentado por el ángulo de inclinación obtenido a partir de los datos de la IMU. El ángulo introducido en el PID 1 como set point no será introducido manualmente, si no que será calculado por el PID 2.

El PID 2 funciona a partir de la señal de error creada por la velocidad media deseada y la obtenida de los motores. Por ejemplo, para el balanceo en un punto estático se necesita que la velocidad media sea 0. Si el vehículo está avanzando más en una dirección la velocidad media en los motores será diferente a 0 y el PID regulará el ángulo de set point introducido en el PID 1 hasta que la señal de error sea nula.

Para calcular la velocidad media no tenemos ningún sensor en el vehículo. Para la obtención este parámetro con precisión se necesita colocar encoders en los ejes de los motores DC, pero no se han colocado por su elevado precio y la dificultad de emplazamiento de los mismos en la plataforma. Se obtendrá de forma aproximada a partir de la señal PWM aplicada a los motores, ya que en una superficie plana se puede asegurar que la precisión será suficiente para este propósito. En una superficie no plana no se puede asegurar una relación entre velocidad y señal PWM ya que, para una velocidad equivalente, en una pendiente descendente la señal PWM será menor que en una pendiente ascendente.

A parte de corregir el offset de ángulo, también servirá para conseguir el avance o retroceso de la plataforma. Esto se consigue proporcionando ángulos mayores (en módulo) al ángulo con el

que se consigue el autobalanceo producto de introducir velocidades diferentes a 0 como set point del PID 2.

5.2.1 Ajuste de los parámetros

Hay diversos métodos para definir los parámetros de un PID. A día de hoy existen muchas herramientas informáticas que los calculan de forma automática. Esto cálculos se hacen a partir de la toma de medidas de la respuesta del proceso con diferentes parámetros, finalizando el proceso con la sugerencia de los parámetros más adecuados. Sin embargo, estas herramientas denominadas "autotuners" funcionan relativamente bien ante situaciones en que se manejan señales de bajo ruido a la entrada del controlador.

Otra manera de ajustar los parámetros es de forma manual, la cual se utilizará para definir los parámetros de nuestro controlador. Al realizar este tipo de ajuste, es necesario tener en cuenta qué efecto produce el cambio de cada uno de los parámetros del PID. El conocimiento de estos efectos es vital para ser capaz de saber qué parámetro ajustar observando el comportamiento del sistema. Se van a describir los efectos de aumentar cada uno de los 3 parámetros en la siguiente tabla.

Parámetro	Tiempo de subida	Sobreoscilacion	Tiempo de estabilización	Error del estado estacionario	Estabilidad
K_P	Disminuye	Aumenta	Poco cambio	Disminuye	Disminuye
K_I	Disminuye	Aumenta	Aumenta	Disminuye	Disminuye
K_D	Poco cambio	Disminuye	Disminuye	Ningún efecto	Aumenta para valores pequeños de K_D

Tabla 5.1 Efectos al aumentar el valor de los parámetros del controlador PID

El primer PID que se ajustará será el PID 1, que actuará directamente sobre los motores. Para ello se colocarán 3 potenciómetros externos conectados en forma de divisor de voltaje resistivo a las entradas analógicas del microcontrolador (cada potenciómetro actuará sobre un parámetro del controlador). Por último, antes de realizar el ajuste de parámetros, se definirá el set point como ß=0º para que el robot oscile alrededor del eje z.

La metodología para el ajuste de parámetros será la siguiente:

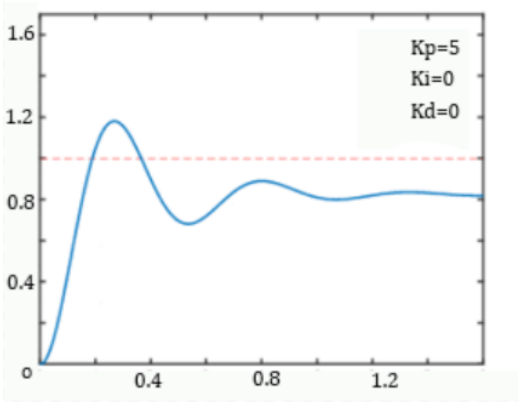

Figura 5.7 Ajuste del parámetro proporcional del controlador

1) Siendo K_I y K_D nulas, se aumenta K_P hasta conseguir una o varias oscilaciones alrededor del set point. Se podrá visualizar que el robot intentará mantener el equilibrio oscilando en el eje z pero no durará más que unos pocos segundos hasta que pierda el equilibrio completamente.

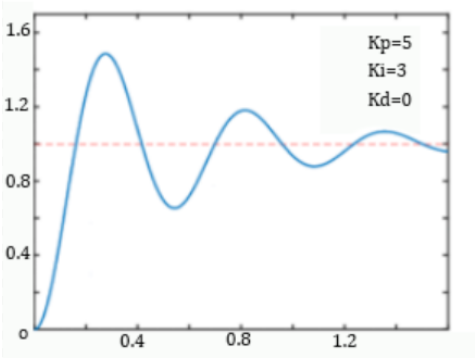

Figura 5.8 Ajuste del parámetro integrador del controlador

2) Se mantiene el parámetro K_D a 0 y se aumentará K_I. Esto conseguirá que se produzcan sobreoscilaciones durante mas tiempo pero el sistema seguirá siendo inestable. Ésto se traduce en que el robot conseguirá mantener el equilibrio durante más tiempo, pero con sobreoscilaciones demasiado grandes que en un momento dado le harán perder el equilibrio.

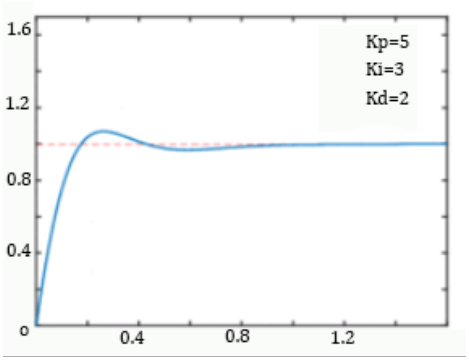

Figura 5.9 Ajuste del parámetro diferencial del controlador

3) El último paso consiste en aumentar K_D. El aumento de K_D reducirá las sobreoscilaciones y le dará estabilidad al sistema. Con un valor adecuado el robot debe ser capaz de mantener el equilibrio durante todo el tiempo.

Tras realizar las tres iteraciones, si la respuesta del sistema es satisfactoria se debe proceder a un ajuste fino de los parámetros. En caso contrario se pondrán todos los parámetros a 0 y se volverá a iniciar el proceso.

Una vez el PID 1 tiene los parámetros adecuados, se tiene que proceder al ajuste del PID 2. Esta vez se introducirá como set point una velocidad media nula y se intentará conseguir que la plataforma se autobalancee manteniendo la posición inicial. El método de elección de parámetros es análogo al utilizado para el PID 1.

5.3 Resultados

Los datos se van a medir ajustando los set point para el balanceo alrededor del eje z. En teoría el único set point sobre el que podemos actuar es sobre la velocidad media deseada. Sin embargo se puede forzar a que el set point inicial del PID 1 sea $\alpha=0º$ y desde ese momento sea el PID 2 quien controle este ángulo

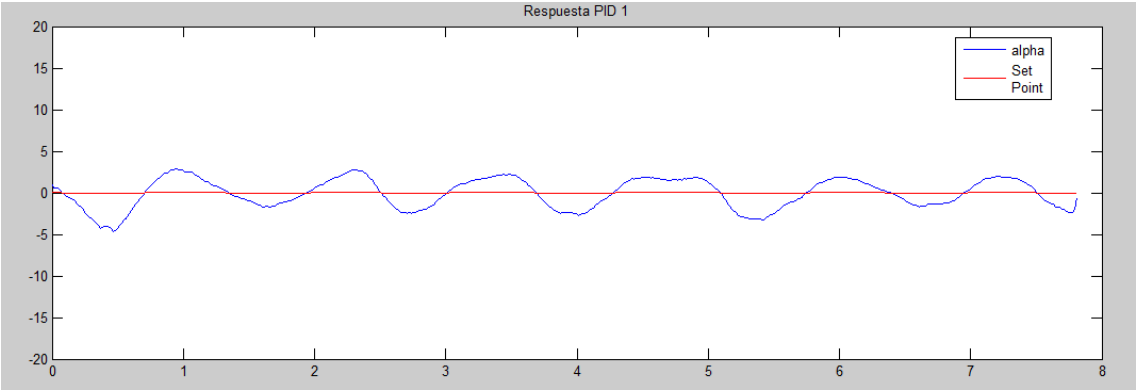

Figura 5.10 Respuesta del PID 1

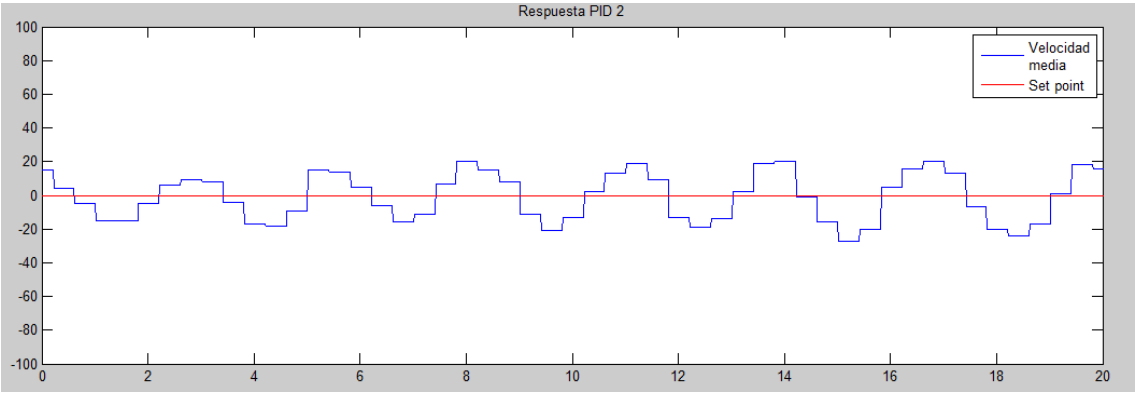

Figura 5.11 Respuesta del PID 2

Se puede ver que en los 2 PID se producen oscilaciones de nivel aceptable y constantes alrededor del set point. Además, la toma de medidas se ha hecho con conexión por cable, por lo que la medida se ha visto influenciada debido a la perturbación del movimiento del vehículo. Los escalones que se producen en la señal de velocidad media son debidos a que no se actualiza en cada iteración.

Capítulo 6. Implementación Software

El microcontrolador va a ser el encargado de contener toda la inteligencia que hará que el vehículo se autobalancee de forma autónoma. Esto quiere decir que tendrá que tomar datos de los sensores, procesarlos y controlar los actuadores. Además tendrá que mantener la comunicación con el dispositivo móvil remoto.

Por otra parte, el dispositivo móvil se encargará de controlar el movimiento y la rotación del robot, será capaz de configurar los controladores y mostrará por pantalla de forma gráfica algunos datos interesantes procedentes del microcontrolador.

Por lo tanto se va a distinguir la implementación en dos partes:

- El software en el microcontrolador
- El software en el dispositivo móvil

Además como interfaz entre ellos tendremos un protocolo de comunicación.

6.1 Protocolo de Comunicación

El protocolo empleado para la comunicación se basa en enviar los valores necesarios separados por comas entre un carácter inicial para sincronismo y un carácter final para detectar el final del mensaje. Además se envía un carácter para indicar qué son los parámetros que van a llegar a continuación. La cadena a enviar tendría el siguiente aspecto:

CS,Cnt,p1,p2,…,pn,CF

Donde:

CS – Carácter de sincronización

Cnt – Indica el contenido de los parámetros que se van a recibir

Pi – Valor del parámetro i

CF– Carácter que indica final de mensaje

El algoritmo para recepción se ilustra en el siguiente diagrama de flujo:

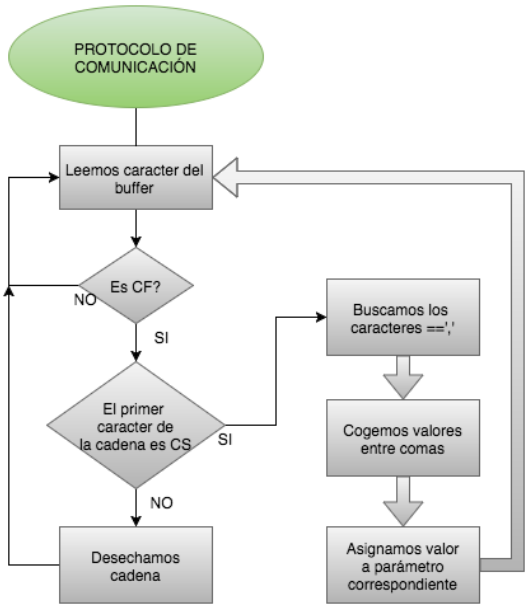

Figura 6.1 Diagrama de flujo del algoritmo de recepción del protocolo de comunicación

6.2 Software en el microcontrolador

El desarrollo de software en el microcontrolador se va a realizar usando el entorno de desarrollo que ofrece Arduino. El entorno Arduino IDE es software libre de código abierto y está basado en un entorno llamado Processing, que a su vez también se trata de un proyecto de software de código abierto que usa un lenguaje basado en Java para su programación.

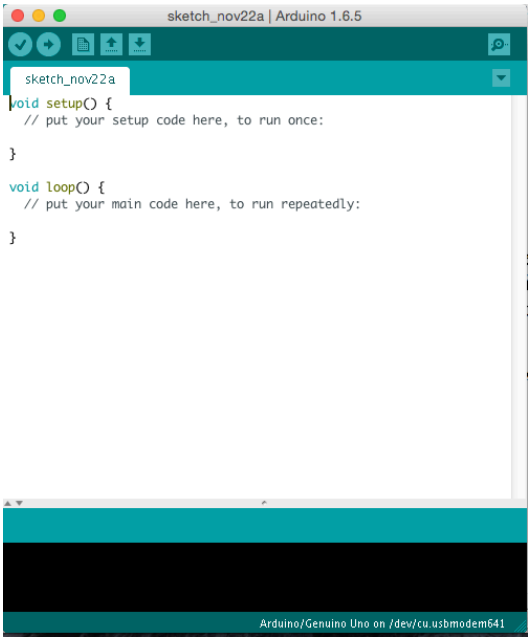

Figura 6.2 Entorno Arduino IDE.

A los programas que luego se implementarán en el microcontrolador se les denomina “sketches” y están estructurados en dos partes: el “setup” y el “loop”. El “setup” es la parte del código que se ejecutará solo una vez al encender el microcontrolador. Aquí se inicializan variables o servicios como la comunicación serie, se configuran periféricos como la IMU, etc. Por otro lado, el “loop”, es el código que se ejecutará continuamente en forma de bucle. Aquí es donde se escriben las rutinas que se deberán repetir.

Una vez escrito el código, el IDE permite subir el programa directamente al microcontrolador a través que se encontrará conectado a un puerto USB. Arduino IDE no incorpora debugger, pero se puede utilizar el terminal serie que lleva integrado para observar variables por pantalla.

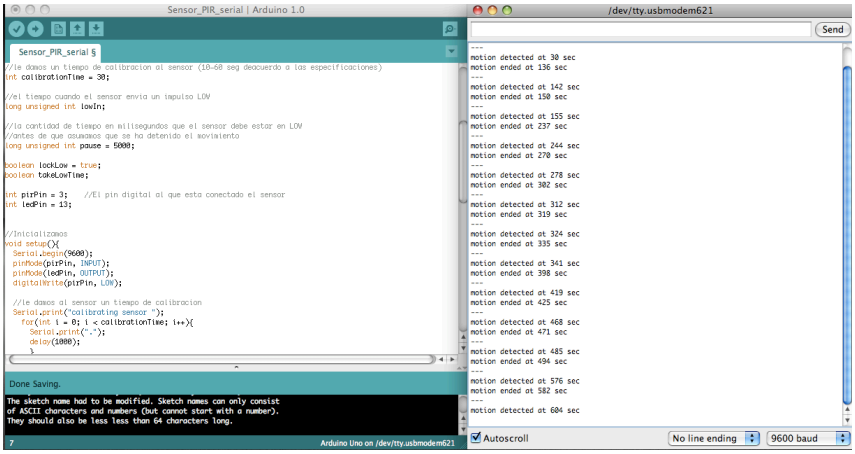

Figura 6.3 Terminal serie de Arduino IDE

6.2.1 Algoritmo

Para explicar el algoritmo primero se va a exponer un diagrama de flujo a nivel general del setup y el loop y a continuación se entrará en detalles importantes de cada uno de ellos.

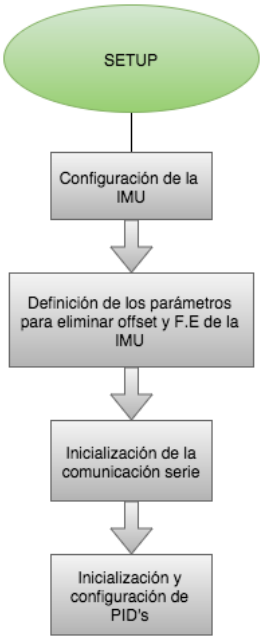

Figura 6.4 Diagrama de flujo de la inicialización del microcontrolador

-Configuración de la IMU:

Se configuran los fondos de escala del acelerómetro y giroscopio. Esta configuración se realiza mediante I2C. I2C exige enviar 1. El registro del periférico del cual se a leer o escribir un registro 2. El registro el cual se va a leer o escribir y 3. El valor a escribir en caso de esté en modo escritura.

```
Wire.begin();                 //Se inicia transmisión I2C
Wire.beginTransmission(MPU);  //MPU es la dirección de la IMU
Wire.write(0x6B);             //Registro para "Despertamos" a la IMU
Wire.write(0);                //en caso de que esté en modo sleep
Wire.endTransmission(true);
Wire.beginTransmission(MPU);
Wire.write(0x1B);             //Registro para configurar el F.E. del acelerometro
Wire.write(0);                //Configuración a +-2g
Wire.endTransmission(true);
Wire.beginTransmission(MPU);
Wire.write(0x1C);             //Registro para configurar el F.E. del giroscopio
Wire.write(0);                //Configuración a +-250°/s
Wire.endTransmission(true);
```

-Definir parámetros para eliminar offset y F.E de la IMU

```
//Ajuste offset acelerometro
config.ajusteOffset[x_acc] = 800;
config.ajusteOffset[y_acc] = -120;
config.ajusteOffset[z_acc] = 100;
//Ajuste sensiblidad acelerometro
for(i=0;i<=2;i++){
  config.Sens[i] = 16384;
}
//Ajuste offset giroscopio
config.ajusteOffset[x_gyro]=-100;
config.ajusteOffset[y_gyro]=200;
config.ajusteOffset[z_gyro]=-60;
//Ajuste sensibilidad giroscopio
for(i=3;i<=5;i++){
  config.Sens[i] = 131;
}
```

Los parámetros para eliminar el offset se han calculado previamente de forma experimental. Se definen para luego sumarlos o restarlos del valor obtenido. El fondo de escala se define para el posterior procesamiento de los datos.

-Inicialización de la comunicación serie

Hay que definir la velocidad de transmisión en baudios a la cual se realizará la comunicación. La comunicación con el módulo Bluetooth se realizará a 9600 baudios. La comunicación Bluetooth se hará por una comunicación serie programada por software. Se dejará el puerto serie del microcontrolador libre para la programación y análisis de fallos.

```
//Inicialización de la comunicación serie
Serial.begin(9600);
mySerial.begin(9600);
```

-Inicialización y configuración de los PID

Se definen los parámetros K_P, K_I y $K_$ el limite de saturación inferior y superior de los datos a la salida del controlador y el tiempo de muestreo.

```
//Inicialización PID
Setpoint = 0;
myPID.SetMode(AUTOMATIC);           //Automatic para "ON"
myPID.SetOutputLimits(-255, 255);   //Saturar salida entre -255 y 255
myPID.SetSampleTime(10);            // Configuración de tiempo de muestro
myPID.SetTunings(Kp, Ki, Kd);       //Se colocan los parámetros
```

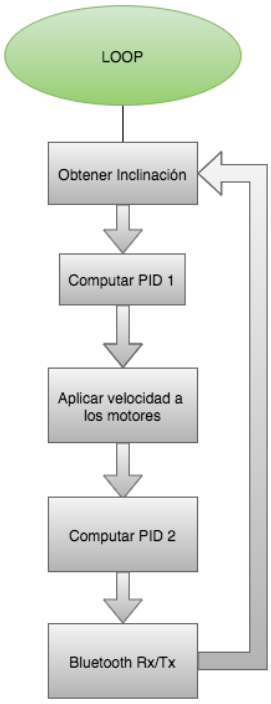

Figura 6.5 Diagrama de flujo del bucle del microcontrolador

-Obtener la inclinación

La obtención de la inclinación pasa por los siguiente pasos:

- Lectura de valores la IMU. Se leen 2 bytes por dato mediante una comunicación I2C.
- Procesado de datos. Se procesan los datos recibidos para obtener información útil.
- Filtrado. Se filtran los datos para finalmente obtener un ángulo de inclinación estimado.

```
Wire.beginTransmission(MPU);
Wire.write(0x3B);                  //En el registro 3B empiezan...
Wire.endTransmission(false);       //..los valores del acel y gir
Wire.requestFrom(MPU,14,true);     //Se leen los 14 registros siguiente
an[0]=Wire.read()<<8|Wire.read(); //Se lee un byte, se desplaza 8 bits a la...
an[1]=Wire.read()<<8|Wire.read(); //...inzquierda y se hace un OR con el...
an[2]=Wire.read()<<8|Wire.read(); //...siguiente byte
Tmp=Wire.read()<<8|Wire.read();
an[3]=Wire.read()<<8|Wire.read();
an[4]=Wire.read()<<8|Wire.read();
an[5]=Wire.read()<<8|Wire.read();
```

-Cómputo de PID 1

El PID 1 es el encargado de controlar la velocidad de los motores a partir del ángulo obtenido y del ángulo deseado. La media de las salidas anteriores de este PID también servirá como retroalimentación del PID 2.

-Aplicar velocidad a los motores

Se usará la salida del PID 1 que estará en el rango [-255,255] como valor para la señal PWM que se aplicará al puente en H. Estos valores se refieren al ciclo de trabajo de la señal PWM donde 0 será 0% y 255 será 100%. Los valores negativos indican un cambio de polaridad en la tensión aplicada al motor y, por lo tanto, un cambio en la dirección de funcionamiento.

Además se tendrá en cuenta el rango de valores en los que el motor esta "muerto". Normalmente, con ciclos de trabajo menores al 10% no se obtiene ninguna respuesta en el motor.

```
// Equivalencia entre pin con motor y dirreción
//3 Izq adelante
//5 Izq atras
//6 Der adelante
//9 Der atras


spdder=Output-irotacion/2;  //Inyectamos rotación a cada motor
spdizq=Output+irotacion/2;

if(Input<-90||Input>90){ //Si el robot ha caido paramos los motores
  analogWrite(5, 0);
  analogWrite(9, 0);
  analogWrite(3, 0);
  analogWrite(6, 0);
  average=0;
  Setpoint=1.5;
  for(i=0;i<99;i++);{
    readings[i]=0;
  }
  delay(5000);          //Esperamos 5 segundos para que de tiempo a...
}                       //...volverlo a colocar
if(spdder<=0){          //Decidimos la polaridad dependiendo del signo
  analogWrite(9, 0);
  analogWrite(6,constrain((-1)*spdder*0.949+13,0,255));  //Saturamos a 255...
}                                                        //...y eiminamos la zona muerta
else{
  analogWrite(6, 0);
  analogWrite(9,constrain(spdder*0.949+13,0,255));
}
if(spdizq<=0){
  analogWrite(5, 0);
  analogWrite(3, constrain((-1)*spdizq*0.968+8,0,255));
}
else{
  analogWrite(3, 0);
  analogWrite(5, constrain(spdizq*0.949+13,0,255));
}
```

La rotación se inyectará directamente a cada motor sumándole o restándole un cierto porcentaje de ciclo de trabajo a la salida del PID 1.

-Cómputo de PID 1

Con la media de las salidas del PID 1, este controlador será el encargado de definir el ángulo de set point que se introducirá, también, en el PID 1.

-Bluetooth Rx/Tx

Se comprobará si hay algún dato en el buffer de entrada del puerto serie y, si es así, se procesará para obtener los valores recibidos. También se enviaran datos de las variables de los controladores PID para la visualización gráfica en el dispositivo móvil.

6.1.2 Cómputo de PID

En este apartado se va a exponer el diagrama de flujo y el código para el computo de los controladores PID.

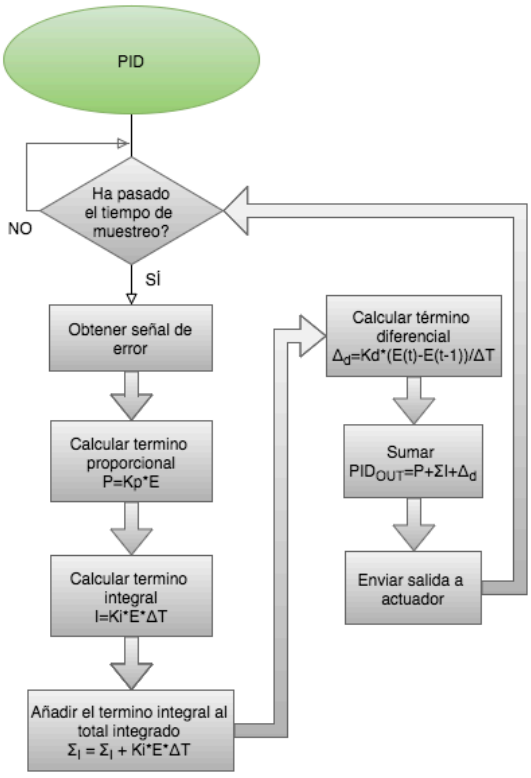

Figura 6.6 Diagrama de flujo para el computo del PID

```
bool PID::Compute() {
if(!inAuto)
          return false;
unsigned long now = millis();
unsigned long timeChange = (now - lastTime);
if(timeChange>=SampleTime){       /*Compute all the working error variables*/
          double input = *myInput;
          double error = *mySetpoint - input;
          ITerm+= (ki * error);
          if(ITerm > outMax)
                    ITerm= outMax;
          else if(ITerm < outMin)
                    ITerm= outMin;
         double dInput = (input - lastInput);         /*Compute PID Output*/
         double output = kp * error + ITerm- kd * dInput;
```

```
            if(output > outMax)
                        output = outMax;
            else if(output < outMin)
                        output = outMin;
            myOutput = output;  /*Remember some variables for next time*/
            lastInput = input;
            lastTime = now;
            return true;
            }
else
            return false; }
```

6.3 Software en el Dispositivo Móvil

El entorno de desarrollo integrado para el sistema operativo Android es el Android Studio. Su lanzamiento fue anunciado en 2010, apareciendo su primera versión estable en diciembre de 2014. Reemplazó a la plataforma Eclipse que se había utilizado hasta el momento para realizar todo el proceso de desarrollo de aplicaciones.

Las aplicaciones se crean en forma de proyecto. Este proyecto contendrá todas las clases java necesarias para la ejecución: librerías, recursos (imágenes, sonidos, etc), layouts para las actividades, archivos de traducción y todos los elementos de configuración necesarios. Todos estos elementos del proyecto son accesibles desde el navegador del entorno.

Una vez realizada la aplicación nos permite subirla a un dispositivo real o un dispositivo virtual (creado por el usuario dentro del entorno o externamente), ofreciendo información en tiempo real del estado de la misma y mostrando por consola mensajes predeterminados por el programador en las rutinas. Además incorpora un debugger, esto nos permite parar la aplicación en cierta instrucción para comprobar el estado de las variables y, a partir de ahí, continuar ejecutando en modo normal o paso a paso.

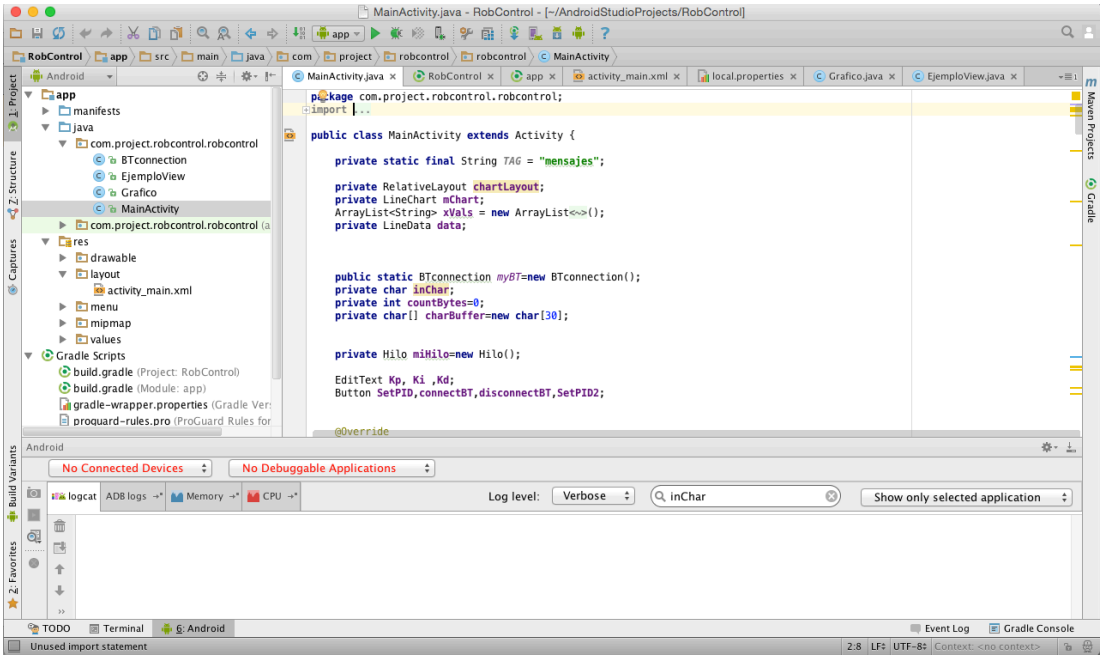

Figura 6.7 Entorno de desarrollo Android Studio

6.2.1 La aplicación

La aplicación consta de una sola actividad en la que se pueden realizar todas las funciones. El layout se compone de los siguientes elementos:

- Sliders para controlar la velocidad y rotación
- Cajas de texto para configurar los parámetros de los PID
- Grafica para visualizar las variables de los PID
- Botones para la conexión y desconexión con el módulo Bluetooth.

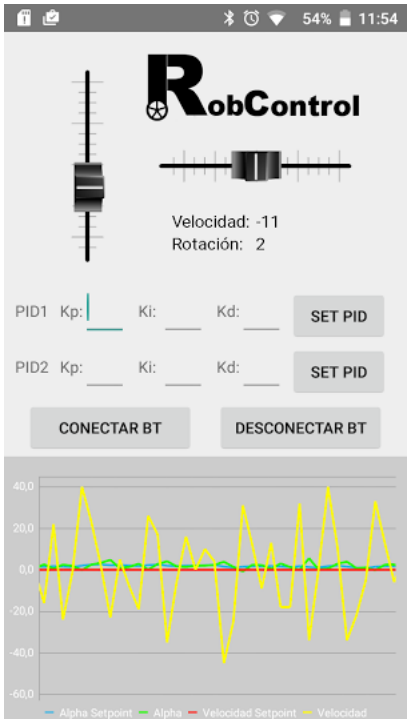

Figura 6.7 Aplicación para dispositivo móvil Android

Las clases de java que incorpora el proyecto son las siguientes:

- MainActivity: será la clase principal de la actividad. En ella se gestiona el ciclo de vida de la aplicación realizando las tareas necesarias en cada momento, como por ejemplo el cerrar la conexión Bluetooth al salir y dejar de ejecutar todos los hilos de ejecución. Además es la que gestionará las gráficas donde aparecen los datos de interés, gestionará las tareas a realizar en caso de que se pulse algún botón e implementará el procesado de datos recibidos.

- BTconnect: gestionará toda la conexión bluetooth, definirá los algoritmos para la conexión, desconexión, recepción y envío de datos. Se instanciará desde la actividad principal y sus métodos para el envío y recepción de datos serán accesibles desde el exterior.

- Gráfico: se utilizará para crear objetos a partir de imágenes que luego serán capaces de ser mostrados por pantalla.

- MyView: extiende la clase View de Android y servirá para gestionar todos los gráficos que hayamos creado a partir de la clase Grafico. Además reaccionará al control táctil para mover los sliders.

Capítulo 7. Conclusiones y Líneas Futuras

El robot ha sido diseñado y construido desde cero y ha cumplido las expectativas planteadas de antemano. Los motores tienen suficiente potencia para mantener el equilibrio pero están limitados a la hora de vencer ángulos de inclinación elevados. El microcontrolador elegido tiene suficiente velocidad de procesado para desarrollar las tareas necesarias, pero está en el límite.

El procesado de datos de la unidad de medición inercial nos permite un ángulo aproximado bastante cercano a la realidad y con poco ruido, lo cual ayuda a obtener una mejor respuesta de los controladores.

Los controladores PID tienen una respuesta adecuada. El balanceo estático se desarrolla de manera correcta. El control durante el movimiento en ambas direcciones puede ser mejorado. El uso de encoders mejoraría la respuesta de los controladores.

La aplicación en el dispositivo móvil es capaz de controlar el robot y configurar los controladores PID de manera correcta. El protocolo diseñado para la comunicación es efectivo para desechar mensajes corruptos.

Como líneas futuras se propone:

- Sustituir microcontrolador por uno con más capacidad de proceso
- Sustituir motores por otros más potentes
- Incorporar encoders en para sensar la velocidad de manera real
- Mejorar los controladores PID
- Incorporar sensor de distancia para evitar obstáculos
- Incorporar optoacopladores para seguimiento de líneas
- Ampliar funcionalidades del control remoto

Bibliografía

[1] Cristopher T. Kilian, "Modern Control Technologies: Components and systems," *3rd edition,* 2006.

[2] Starlino, "A guide to use IMU in embedded applications," http://www.starlino.com/imu_guide.html

[3] University of Michigan, "Introduction: PID controller design," http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID

[4] Android, "Developer's Guide", http://developer.android.com/develop/index.html

[5] Jinghua Zhong," PID Controller Tuning: A Short Tutorial", http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf, 2006