



Universidad
Zaragoza

Trabajo Fin de Máster

Sistema de navegación autónoma de un dirigible

Autonomous navigation system of a blimp

Autor

Rafael Herguedas Gastón

Directores

Gonzalo López Nicolás

Carlos Sagüés Blázquez

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2017



DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./D^a. Rafael Herguedas Gastón,

con nº de DNI 73028785-J en aplicación de lo dispuesto en el art.

14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo

de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la

Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster)
Máster _____, (Título del Trabajo)

Sistema de navegación autónoma de un dirigible

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada
debidamente.

Zaragoza, 17 de noviembre de 2017

Fdo:

R. Herguedas

Sistema de navegación autónoma de un dirigible

Resumen

Este Trabajo Final de Máster tiene por objeto el estudio del control del movimiento de un VANT (Vehículo Aéreo No Tripulado) comercial de tipo dirigible (blimp). Se ha abordado para ello el diseño de un sistema de navegación autónomo, de bajo coste, que permite gobernar un dirigible de pequeño tamaño para alcanzar la posición deseada.

El dirigible estudiado consiste en un globo fusiforme inflado con gas helio al cual se acopla una góndola con los actuadores y material de lastre en la parte inferior. Los actuadores son tres motores eléctricos que permiten desplazar el dirigible en las tres dimensiones del espacio. Para controlar el movimiento del dirigible se han planteado un algoritmo de control cinemático proporcional al error de posición y un sistema de control dinámico en espacio de estados con integradores y estimador de estado. El control en espacio de estados se ha simulado con el software Matlab.

El desarrollo del sistema de navegación se ha llevado a cabo, principalmente, en la plataforma de software *open source* Raspberry Pi versión 3 por medio del lenguaje de programación Python 2.7. Sobre esta plataforma se ha implementado el sensor de medida del movimiento del dirigible, que en este caso ha consistido en una cámara de 5 megapíxeles, así como la interfaz de emisión de infrarrojos necesaria para transmitir las señales a los actuadores.

Con el propósito de imitar a conveniencia las señales infrarrojas que emite el mando proporcionado con el dirigible se ha hecho uso de la librería de software libre para Linux LIRC, con la cual es posible guardar en memoria las señales emitidas y poder reproducirlas a posteriori. Para mejorar las prestaciones del sistema de comunicación, se ha optado finalmente por el módulo *pigpio* de Python.

Por último, para optimizar el proceso de reconocimiento visual del dirigible, se ha hecho uso también de las librerías de software libre OpenCV y Aruco, que proporcionan una serie de marcadores visuales y funciones con las que realizar las tareas de calibración de la cámara, adquisición y tratamiento de los datos de forma altamente eficiente. Estas librerías se han instalado igualmente en la placa Raspberry Pi.

Tabla de contenidos

1. Introducción	1
1.1. Contexto y motivación del estudio.....	1
1.2. Estado de la materia	1
1.3. Objetivos	3
1.4. Alcance	3
2. Dirigible utilizado.....	5
2.1. Descripción general.....	5
2.2. Montaje y puesta a punto	6
2.3. Condiciones de funcionamiento	7
3. Interfaz de comunicación	9
3.1. Descripción del sistema de partida.....	9
3.2. Raspberry Pi	9
3.3. Circuito de recepción y emisión.....	11
3.4. LIRC	12
3.4.1. Configuración.....	12
3.4.2. Grabación de las salidas IR del mando a distancia	13
3.5. Pigpio	14
4. Captura de movimiento.....	16
4.1. Descripción y configuración del sensor	16
4.2. OpenCV y Aruco	17
4.2.1. OpenCV	17
4.2.2. Aruco	19
4.3. Calibración de la cámara.....	20
5. Sistema de control	22
5.1. Modelo cinemático y control proporcional	22
5.2. Modelo dinámico y control en espacio de estados con integradores	24
5.2.1. Modelo dinámico del dirigible.....	24
5.2.2. Control con integradores en serie y estimador de estado.....	26
6. Resultados.....	30

6.1. Pruebas de funcionamiento del controlador proporcional	30
6.1.1. Montaje experimental	30
6.1.2. Resultados obtenidos	31
6.1.3. Fuentes de error	34
6.2. Simulación del controlador en espacio de estados	35
6.2.1. Parámetros del sistema	35
6.2.2. Controlabilidad y observabilidad	36
6.2.3. Resultados de simulación	36
7. Conclusiones y desarrollo futuro	39
7.1. Conclusiones	39
7.2. Desarrollo futuro	39
8. Bibliografía	41
Anexo I: Código en Python del sistema de navegación	45
Anexo II: Código de Matlab	57
Anexo III: Hojas de características de los componentes	61

1. Introducción

1.1. Contexto y motivación del estudio

En los últimos años el auge de los VANT (Vehículos Aéreos No Tripulados) o drones ha alcanzado enormes cotas más allá del sector de la defensa, con el que han estado fuertemente vinculados desde sus primeras apariciones.

Este proyecto ha surgido precisamente del incremento del número de drones en entornos compartidos con las personas y de los riesgos que esto conlleva. El mercado de VANT de uso civil está dominado actualmente por los drones de cuatro hélices o *quadrotors*, que presentan una serie de desventajas que limitan su capacidad para interactuar con seres humanos de forma segura. Estas desventajas son mucho menos acusadas en los drones de tipo dirigible o *blimps*, cuyas propiedades, como demuestran algunos estudios [1], los hacen más aptos para la convivencia con las personas. En la Tabla 1 se puede observar la comparativa de algunas de estas características entre ambos tipos de dron.

Característica	Quadrotor	Blimp
Seguridad de operación	Baja	Alta
Nivel de ruido y vibraciones	Alto	Bajo
Autonomía	Muy baja	Alta
Velocidad	Muy alta	Baja
Maniobrabilidad	Alta	Baja
Resistencia al viento	Media	Muy baja

Tabla 1. Comparativa de características entre drones de tipo quadrotor y blimp

Con la motivación, por tanto, de diseñar un sistema de navegación automatizado para un dron de tipo dirigible en entornos cerrados, de dimensiones reducidas y compartidos con personas, se ha desarrollado este estudio en el área de Ingeniería de Sistemas y Automatización de la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza.

1.2. Estado de la materia

A pesar de que existen registros puntuales de apariciones previas, el uso de VANT se podría considerar que comenzó a principios del siglo 20 con fines bélicos, aunque su despliegue a gran escala no se produciría hasta los años 50 durante la guerra de Vietnam [2]. A partir de entonces estos aparatos se han seguido usando como herramienta bélica con frecuencia, pero en la actualidad las aplicaciones civiles han crecido

exponencialmente en diversos campos como la prevención de incendios, la agricultura, los medios audiovisuales y el entretenimiento.

Por otro lado, los dirigibles, que surgieron a mediados del siglo XIX, entraron en desuso en los años 30 tanto en el campo militar como en el civil tras sufrir una serie de graves accidentes y verse superados en ciertos aspectos por los aeroplanos. El más grave de estos incidentes fue el protagonizado por el dirigible alemán Hindenburg [3], que se incendió cuando aterrizaba en Nueva Jersey el 6 de mayo de 1937. Este hecho fue, entre otros, el que puso fin al uso en aeronaves de hidrógeno como gas propulsor en favor del helio, más caro y pesado pero no inflamable. A partir de entonces y hasta la actualidad, su presencia había quedado relegada a usos publicitarios y recreativos esporádicos.

Hoy día, existen múltiples ámbitos en los que se realizan investigaciones y estudios relacionados con los drones y, en concreto, con los drones dirigible.

Uno de ellos es el de las actuaciones de búsqueda y rescate llevadas a cabo en escenarios de desastre natural, donde el uso de los VANT se presenta como una opción muy eficaz y segura para realizar tareas de reconocimiento aéreo. A este respecto, en el marco de un estudio llevado a cabo en el *Instituto Militar de Engenharia* de Río de Janeiro se realizaron en 2011 unas misiones de reconocimiento con VANT en la región montañosa de Río de Janeiro, devastada por las inundaciones, obteniendo resultados muy positivos [4]. También con el propósito de evaluar las posibilidades de equipos de reconocimiento colaborativos de búsqueda y rescate con robots terrestres y aéreos, se inició una investigación en el *Instituto Superior Técnico* de la Universidad Técnica de Lisboa [5]. En este estudio se realizaron pruebas de seguimiento aéreo de un robot terrestre con drones dirigible basados en visión por computador.

En el campo de la seguridad y la vigilancia en espacios interiores desde el *Georgia Institute of Technology* se han llevado a cabo recientemente experimentos de reconocimiento facial y seguimiento de personas [6] y también de navegación autónoma en entornos sin acceso a GPS con drones dirigible [7].

Además de estas investigaciones, existe una iniciativa llamada *Project Loon* [8] que la empresa Google anunció oficialmente en el año 2013. Este proyecto busca proveer de conexión a Internet a zonas rurales remotas y también a lugares cuyas conexiones se hayan visto interrumpidas mediante una red de globos automatizados de despliegue rápido. En octubre de este año, Google aportó conectividad con este sistema en las zonas de Puerto Rico afectadas por el huracán María.

Otros estudios recientes [9]–[11] se han enfocado también en desarrollar métodos y algoritmos basados en captura de puntos por visión para controlar micro VANT, y un

proyecto de características similares al presente pero centrado en un sistema de control *fuzzy* se desarrolló en 2011 en la Universidad de Edimburgo [12].

1.3. Objetivos

Los objetivos principales del proyecto son los siguientes:

- Seleccionar un dirigible dron de tamaño reducido de entre los que se pueden encontrar en el mercado, con aquellas características que se estimen más adecuadas para la aplicación considerada.
- Diseñar un sistema de navegación autónomo de bajo coste adecuado al modelo de dron seleccionado y al entorno en el que se va a desenvolver (espacios cerrados y reducidos con presencia de personas).
- Comprobar mediante pruebas de funcionamiento la idoneidad del sistema de navegación propuesto.

1.4. Alcance

Las actividades desarrolladas para conseguir los objetivos mostrados en el apartado anterior, que se explican detalladamente en los apartados posteriores, se pueden resumir en las siguientes:

- Buscar información relevante sobre el estado actual de la materia referente a drones dirigible y sistemas de navegación en VANT, además de otros estudios llevados a cabo actualmente sobre drones de uso civil.
- Analizar, valorar y comparar las características de los dirigibles dron para uso recreativo que se pueden encontrar en el mercado, para seleccionar finalmente el que resulte más adecuado.
- Implementar sobre la plataforma de software libre Raspberry Pi 3 un sistema de recepción y emisión de señales infrarrojas (en adelante IR) que permita además grabar y reproducir a conveniencia los comandos procedentes del mando de control remoto con el que se suministra el dirigible, basado en la librería de software libre LIRC.
- Establecer mediante una cámara para Raspberry Pi y las librerías de software de visión por computador OpenCV y Aruco un método para monitorizar en tiempo real la posición y orientación del dirigible con suficiente precisión y flexibilidad.
- Diseñar un algoritmo de navegación, en lenguaje de programación Python, que permita controlar el movimiento del dron para alcanzar posiciones predeterminadas

en el espacio, y que integre tanto el seguimiento visual como la emisión de las órdenes IR.

- Realizar una serie de pruebas de funcionamiento de forma que se puedan identificar y corregir todo lo posible las deficiencias del sistema propuesto.

El siguiente esquema (Figura 1) pretende ilustrar de qué forma se interrelacionan los aspectos anteriores para conformar el sistema de navegación concebido.

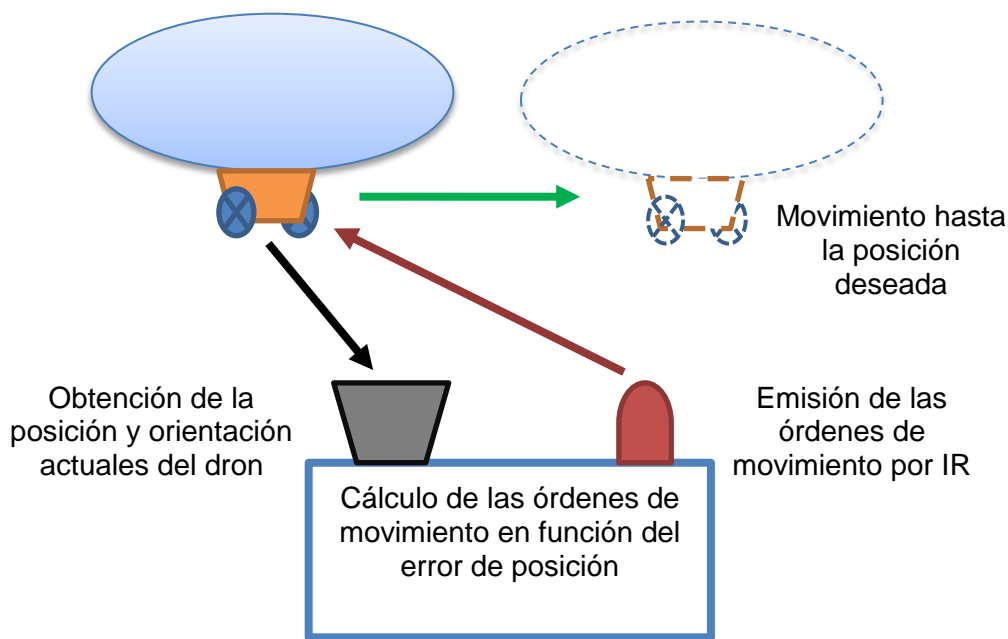


Figura 1. Esquema de funcionamiento del sistema de navegación autónomo

Dado que el trabajo realizado está orientado exclusivamente a entornos cerrados de interior, no se ha considerado la legislación actual sobre aeronaves civiles pilotadas por control remoto [13], regulada por la Agencia Estatal de Seguridad Aérea (AESA).

2. Dirigible utilizado

En este capítulo se presenta y describe el modelo de dirigible seleccionado. También se explica el procedimiento a seguir para su puesta en funcionamiento.

2.1. Descripción general

Como se comentaba en el apartado 1.1, el mercado de drones para uso recreativo está desbordado de drones quadrotor con multitud de características y precios diferentes, pero sin embargo éste no es el caso con los drones de tipo dirigible. De entre los pocos modelos que se pueden encontrar se ha elegido el “ZEP-AIR Explorer II” de la empresa estadounidense Horizon Trade Group LLC, que puede observarse en la Figura 2.



Figura 2. Modelo de dirigible utilizado en este proyecto: ZEP-AIR Explorer II, con detalle de la góndola

El blimp posee un globo de 80 x 40cm (dimensiones del globo desinflado) que es el que proporciona, una vez lleno hasta una capacidad máxima de 75l, el empuje vertical que compensa en parte el peso del conjunto. Para llevar a cabo el movimiento se utilizan tres hélices motorizadas de 5cm de diámetro, una para realizar el ascenso y descenso y dos para realizar los movimientos (avance y giro) en el plano horizontal. Éstas se controlan gracias a un mando de control remoto por medio de señales IR, y se alimentan a través de una pequeña batería recargable de 2.4V.

Las características por las cuales se ha optado finalmente por este modelo son, principalmente, las siguientes:

1. Conexión IR con canales múltiples: La conexión por infrarrojos permite la grabación y reproducción de las órdenes de movimiento de forma relativamente sencilla, sin tener que realizar modificaciones sobre el diseño original del dron. Además, la

arquitectura de canales múltiples permitiría controlar hasta 4 dirigibles en una misma sala, tras haberlos sincronizado previamente en lugares separados.

2. Capacidad de elevación de carga extra: El fabricante especifica una capacidad de carga extra máxima que el dron puede elevar, que en un día frío (25°C) al nivel del mar es de alrededor de 10 gramos.
3. Sencillez de montaje y puesta a punto: A pesar de que hay que llevar a cabo una serie de pasos de montaje hasta tener listo el dirigible, estos no requieren una especial habilidad y todo el material que se precisa viene incluido.
4. Precio asequible: El coste final añadidos los gastos del envío es de 100€ aproximadamente, mayor que el de otros modelos de su categoría (que rondan los 70€) pero por debajo de modelos más avanzados que llegan a superar los 500€.

Los modelos que han sido descartados presentaban, entre otras, las siguientes deficiencias en su diseño y en la información que se proporcionaba sobre ellos: no se especificaba capacidad de elevación de carga extra o ésta era nula; el sistema de comunicación no era suficientemente accesible (radiocontrol clásico, manejo con una app móvil) ni versátil (un solo canal); no permitían el control del movimiento en alguna de las direcciones; escasa robustez mecánica, etc.

2.2. Montaje y puesta a punto

Tal y como se mencionaba en el apartado anterior, el montaje del ZEP-AIR es sencillo, pero es necesario hacer hincapié en ciertos detalles del mismo que influyen en el funcionamiento posterior del dirigible.

Hay dos aspectos que condicionan el correcto funcionamiento del sistema de movimiento del dirigible. Por un lado, al fijar el conector, que une la góndola al globo, a la góndola hay que asegurarse de que el tubo que lo atraviesa queda lo más alineado posible con la góndola. De lo contrario se producirá un desequilibrio que alterará el movimiento. También se modificará el comportamiento del dron si al unir este tubo al globo se hace de forma que el centro de gravedad de la góndola queda alejado de la línea central del globo.

Con respecto al llenado del globo es conveniente seguir las indicaciones del fabricante, que recomienda utilizar helio como gas impulsor y un llenado lento para no producir perforaciones en el material. Además, proporciona una fórmula con la que calcular el empuje en gramos del globo lleno de helio puro a 25°C para altitudes distintas a la del nivel del mar:

$$\text{Empuje [g]} = 40 - x/183 \quad (1)$$

donde x se refiere a la altitud en metros.

El llenado con hidrógeno sería técnicamente posible, siendo además un gas más barato y que proporciona un empuje mayor que el helio al ser más ligero, pero no es recomendable hacerlo al ser también un gas altamente inflamable.

Tras llenar el globo hay que añadir fichas de lastre a la góndola hasta que el dirigible empiece a caer de forma lenta por su propio peso, lo que evita que en caso de perder la señal del mando o de que se produzca algún problema el dirigible ascienda sin control hasta perderse.

2.3. Condiciones de funcionamiento

Con el fin de comprobar algunas de las propiedades indicadas por el vendedor e identificar algunas no especificadas, se ha llevado a cabo una serie de pruebas de funcionamiento con el mando suministrado.

En primer lugar se ha realizado una estimación del peso extra máximo que es capaz de levantar el dirigible, que ha resultado ser para un llenado completo, a unos 300m sobre el nivel del mar y a 22°C de unos 7 gramos. Si se utiliza la fórmula (1) se obtiene un empuje total de 38.36g, el cual se queda en 7.37g al restar el peso de la góndola (31g aproximadamente), lo que daría por válido el resultado de la ecuación.

Por otro lado, el mando a distancia proporcionado con el dron, que se alimenta con una pila de 9V, posee un joystick con el que se determinan las órdenes de movimiento en el plano horizontal. Dado que estos movimientos no se especifican claramente en el manual de usuario ha habido que identificarlos, tras lo cual se han reconocido los siguientes:

- Avance y retroceso puros con dos niveles de potencia, bajo y alto, que se activan empujando gradualmente el joystick hacia delante y hacia atrás.
- Giro en el eje vertical en sentido horario y anti-horario con un nivel de potencia, que se activan desplazando el joystick hacia la derecha o hacia la izquierda.

Además el mando cuenta con una ruleta posterior que, al ir girándola en sentido anti-horario, permite ejecutar las siguientes operaciones:

- Encender el mando, ejerciendo una ligera presión al comenzar a girar la ruleta.
- Emparejar el mando con el dron, girando la ruleta antes de que aparezca un saliente que indica el inicio del envío de potencia al motor vertical.
- Regular la potencia del motor vertical en un amplio rango, en el que se han identificado hasta 5 niveles de potencia.

Por último, con el propósito de estimar el comportamiento cinemático y dinámico para la posterior elaboración del algoritmo de control, se ha manejado el dron explotando todas sus capacidades. Tras observar las respuestas y movimientos del dirigible se ha comprobado que es un sistema de movimiento estable en el plano aunque no tanto en el eje vertical, y se ha determinado también para el desarrollo posterior que los efectos dinámicos son poco representativos con respecto a los cinemáticos a velocidades lentas.

3. Interfaz de comunicación

En este apartado se describe el sistema de comunicación con el dirigible, que sustituye al mando con el que se suministra.

3.1. Descripción del sistema de partida

El ZEP-AIR incorpora, como se indicaba anteriormente, un sistema de emisión y recepción de comandos IR con el que se controla el movimiento del dirigible en las tres dimensiones. Como se puede observar en la Figura 3, el receptor se encuentra en la parte trasera de la góndola del dirigible, y el emisor en la parte frontal del mando a distancia.

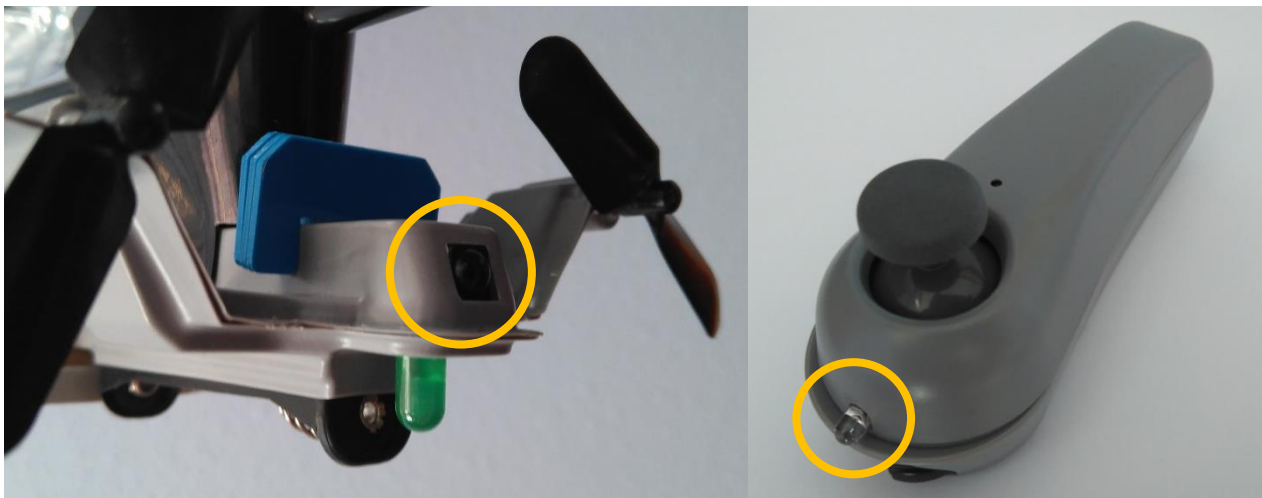


Figura 3. De izquierda a derecha, receptor y emisor del ZEP-AIR

Este método de comunicación es adecuado para entornos cerrados y de reducidas dimensiones como los domésticos, ya que las señales pueden rebotar en las paredes y objetos para alcanzar el receptor con más facilidad. Hay que tener en cuenta también que las condiciones de iluminación pueden perjudicar la comunicación y disminuir enormemente el alcance de las señales [14], ya que las emisiones infrarrojas de las fuentes luminosas como lámparas o el sol interfieren con los datos recibidos y pueden llegar incluso a cegar el receptor. Por ello hay que buscar un compromiso de iluminación del entorno que permita obtener imágenes en la cámara con suficiente resolución sin llegar a afectar a la comunicación de forma negativa. En cualquier caso, el rango de recepción de los pulsos infrarrojos está limitado a un máximo de aproximadamente 10m.

3.2. Raspberry Pi

La base sobre la que se ha llevado a cabo el desarrollo del sistema de navegación y sobre la que se han integrado todos los bloques funcionales, incluido el sistema de comunicación, ha sido la Raspberry Pi versión 3, que se muestra en la Figura 4.



Figura 4. Raspberry Pi versión 3 [15]

Esta plataforma de software libre consta de una serie de capacidades de entre las que destacan las siguientes:

- CPU Broadcom BCM2837 de 64bit y 4 núcleos a 1.2GHz
- 1GB de memoria RAM
- 40 pines GPIO configurables
- 4 puertos USB 2.0
- Puertos HDMI, Ethernet y para la cámara Raspberry Pi
- Conexiones LAN y Bluetooth inalámbricas *on-board*

Además, la Raspberry se configura generalmente sobre el sistema operativo Raspbian, distribución de Linux basada en Debian con diversas modificaciones de compatibilidad que se puede descargar desde la página web de la Fundación Raspberry Pi. El sistema operativo es cargado desde una tarjeta micro SD en el encendido.

Para llevar a cabo la instalación de los sensores y librerías que se indicarán en apartados posteriores, es necesario realizar unos cambios en la configuración por defecto de la placa.

En primer lugar hay que grabar el sistema operativo Raspbian en una tarjeta de memoria micro SD, y ésta debe tener una capacidad de, al menos, 16GB. Vale la pena resaltar estas dos indicaciones ya que, por un lado, el sistema operativo de fácil instalación NOOBS, que se suministra también desde la página web de la fundación, no permite explotar toda la capacidad de almacenamiento de la tarjeta micro SD, y por otro lado para la instalación de la librería de software de visión OpenCV es imprescindible contar con más de 4GB de espacio libre en memoria, lo cual ha resultado impracticable con una tarjeta de 8GB de espacio. Una vez que se haya grabado la imagen de Raspbian e iniciado la Raspberry, el siguiente comando permite acceder al menú de configuración:

```
$ sudo raspi-config
```

Desde el menú que aparece se puede acceder a las opciones avanzadas de configuración, y desde ahí a la opción “*Expand Filesystem*”, que al activarla permite utilizar toda la capacidad disponible en la tarjeta.

También se pueden habilitar desde este menú tanto el puerto de la cámara de la Raspberry, paso obligado para poder acceder a la misma, como la comunicación por SSH. Esto último permite acceder a la Raspberry desde un ordenador para ejecutar comandos y acceder al sistema de archivos a distancia.

Por último, dado que Raspbian es un sistema operativo que se actualiza constantemente es importante tener presente con qué versión se está trabajando. En este caso se ha trabajado con la versión de Raspbian 4.9.41-v7.

3.3. Circuito de recepción y emisión

Con el fin de sustituir el mando por un sistema que permita enviar las órdenes de movimiento de forma automática según determine el algoritmo de navegación, es necesario construir un circuito con el que poder recibir y emitir pulsos IR. Este circuito debe contar, por lo tanto, con un mínimo de un receptor IR y un diodo emisor IR, los cuales servirán respectivamente para leer las señales emitidas por el mando y para poder reproducirlas después de manera controlada.

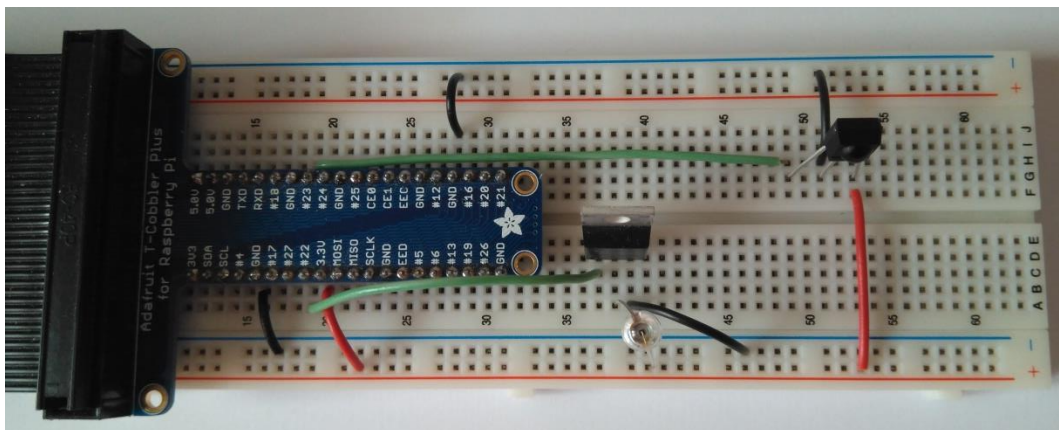


Figura 5. Circuito de recepción-emisión IR conectado a la Raspberry

Como se puede observar en la Figura 5, el circuito que se ha utilizado se conecta a la Raspberry a través de un *cobbler* extensor, de forma que los componentes estén alimentados y conectados a los pines de entrada y salida. Por un lado, el receptor IR que se ha elegido y que sitúa a la derecha de la imagen ha sido el TSSP4056 de Vishay, que se alimenta a 3.3V y cuya salida está conectada al pin 23 de la Raspberry. Por otro, en la zona central de la imagen se encuentran el diodo emisor IR TSHG6400, también de Vishay, y un transistor MOSFET RFP70N06 de Fairchild Semiconductor. Aunque el diodo

IR podría conectarse directamente entre el pin 23 de salida y masa (GND), el uso del transistor permite alimentar el diodo a una corriente más alta que la que proporciona el pin. Para ello la puerta del transistor se conecta al pin 23, la fuente a masa, el drenador al cátodo del diodo y el ánodo de éste a la alimentación de 3.3V. De esta forma el diodo puede recibir hasta 50mA de corriente, lo que permite emitir a una potencia más alta que con los 8mA que proporciona el pin. Cabe mencionar que durante el transcurso de las pruebas de funcionamiento se ha comprobado que la potencia del diodo es aún insuficiente en algunos casos, por lo que se ha conectado el diodo a un cable de forma que pueda ser acercado manualmente al receptor de la góndola. El aumento de potencia se podría conseguir también por medio de un driver de potencia y utilizando la fuente de 5V de la Raspberry, que permite drenar más corriente que la de 3.3V.

Las hojas de características de los componentes utilizados se recogen en el Anexo III.

3.4. LIRC

Una vez que el circuito está preparado es necesario administrar de alguna forma las entradas y salidas que se producirán dentro del mismo. En este caso se ha optado, en un primer momento, por la librería open-source LIRC para realizar esta tarea.

3.4.1. Configuración

LIRC, acrónimo de Linux Infrared Remote Control, permite gestionar la emisión y recepción de señales IR en sistemas operativos Linux [16]. Para instalar este software basta con ejecutar en la línea de comandos lo siguiente:

```
$ sudo apt-get install lirc
```

Tras instalar la librería es necesario modificar el archivo "hardware.conf", que se encuentra en el directorio /etc/lirc/, y añadir las siguientes líneas:

```
LIRCD_ARGS="--uinput"
LOAD_MODULES=true
DRIVER="default"
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"
LIRCD_CONF=""
LIRCMD_CONF=""
```

Esto modifica la configuración por defecto de LIRC para adaptarla a la Raspberry.

Seguidamente, hay que modificar el archivo “modules” del directorio /etc/ para añadir lo siguiente:

```
lirc_dev
lirc_rpi gpio_in_pin=23 gpio_out_pin=22
```

De esta forma se registran la entrada y la salida procedentes del receptor y el diodo IR a los pines 23 y 22 de la Raspberry respectivamente y se vinculan a LIRC.

Por último, hay que registrar el dispositivo en la configuración de arranque añadiendo la siguiente línea al archivo config.txt en el directorio /boot/:

```
dtoverlay=lirc-rpi,gpio_in_pin=23,gpio_out_pin=22
```

Con estos cambios la librería estaría configurada y preparada para ser utilizada en la Raspberry.

Para comprobar que la configuración se ha realizado correctamente es conveniente llevar a cabo una lectura directa del receptor IR. Para esto hay que cambiar la línea en la que aparece `driver = devinput` por `driver = default` del archivo `lirc_options.conf` en el directorio `/etc/lirc/`. Tras esto habría que ejecutar los siguientes comandos en la consola:

```
$ sudo service lircd restart
$ sudo /etc/init.d/lircd stop
$ mode2 -d /dev/lirc0
```

Tras introducir la última línea el receptor quedará a la espera para recibir pulsos IR y mostrar la lectura por pantalla. Si al enfocar y emitir hacia el receptor no se obtiene una salida, esto indicará que la configuración no se ha realizado con éxito, y deberán revisarse los pasos previos realizados.

3.4.2. Grabación de las salidas IR del mando a distancia

Llevada a cabo la configuración de la librería, se puede proceder a la grabación de las órdenes de movimiento del dirigible. La grabación se realiza desde la herramienta “irrecord”, la cual indica de forma sucesiva los pasos a seguir para obtener el fichero de configuración del mando. Los comandos a ejecutar para iniciar la herramienta son:

```
$ sudo /etc/init.d/lircd stop
$ irrecord -f -d /dev/lirc0 ~/lircd.conf
```

Introducidos los comandos, se abrirá la interfaz de grabado en modo *raw*. Según se ha comprobado, este modo es el único con el que se reconocen los pulsos enviados por el mando del ZEP-AIR. Así pues la herramienta solicitará el nombre de los botones a

grabar y registrará el pulso IR correspondiente a cada uno. En este caso, los botones que se han grabado se muestran en la Tabla 2.

Nombre del botón	Movimiento que produce
KEY_POWER	Sincronización mando-dirigible
KEY_FRONT	Avance hacia delante
KEY_RIGHT	Giro vertical en sentido horario
KEY_LEFT	Giro vertical en sentido anti-horario
KEY_STOP	Detención del movimiento en el plano horizontal
KEY_0	Nivel de potencia nulo para el movimiento vertical
KEY_1	Nivel de potencia 1 para el movimiento vertical
KEY_2	Nivel de potencia 2 para el movimiento vertical
KEY_3	Nivel de potencia 3 para el movimiento vertical
KEY_4	Nivel de potencia 4 para el movimiento vertical
KEY_5	Nivel de potencia máxima para el movimiento vertical

Tabla 2. Botones grabados del mando del ZEP-AIR

Cabe mencionar que el grabado de los comandos infrarrojos se ha llevado a cabo con la ayuda del diodo LED rojo que incorpora el mando bajo el joystick, y que se ilumina, tal y como se ha comprobado, cada vez que se emite un pulso IR. Esto ha facilitado y acotado la identificación de las órdenes de movimiento.

Al terminar la grabación se genera un archivo de configuración con el nombre “lircd.conf”, tras lo cual sólo queda copiarlo en el directorio /etc/lirc/ para tener los botones accesibles para el envío.

3.5. Pigpio

En un momento del desarrollo, al comenzar a realizar pruebas de funcionamiento del sistema de navegación, se ha llegado a la conclusión de que la velocidad de envío de los comandos IR y la fiabilidad de los mismos proporcionadas por LIRC eran claramente insuficientes. Estos problemas han sido superados con creces sustituyendo LIRC por el módulo de Python *pigpio* [17].

Este módulo, proporcionado junto con la versión de Python utilizada, es capaz de configurar ondas y pulsos que se pueden generar desde los pines de la Raspberry con una precisión de 1 microsegundo. Esta precisión es posible conseguirla con un servicio llamado *pigpio daemon*, que actúa en capas más profundas del software en las cuales no se producen interrupciones del sistema que puedan alterar la frecuencia de las ondas de salida. Mediante esta capacidad para generar pulsos se ha podido configurar la Raspberry para emitir los comandos IR a través del diodo. Además, desde la página oficial de pigpio

se proporciona un programa en Python que permite grabar de forma bastante precisa (mucho más que LIRC) pulsos IR de cualquier tipo, para poder utilizarlos posteriormente desde el módulo. El reconocimiento correcto de estos pulsos por parte del receptor del dirigible se estima alrededor del 90%, mientras que con LIRC era inferior al 50%.

4. Captura de movimiento

La estimación del estado del sistema se realiza mediante percepción visual. En esta sección se presenta detalladamente cómo se ha implementado y configurado el sistema de visión.

4.1. Descripción y configuración del sensor

Conocer el estado del dirigible es imprescindible para implementar el sistema de navegación de forma adecuada, pues de lo contrario no sería posible calcular cómo de cerca o lejos se está del estado deseado para poder actuar en consecuencia.

De entre las diversas opciones de sensores que se pueden utilizar para medir la posición y la orientación, que son los datos que nos interesa medir como se demostrará más adelante, se ha optado por la medición con un sensor de visión por varios motivos:

- Al poder disponer el sensor en el entorno sin tener que acoplarlo al dirigible se evitan problemas de sobrecarga en el mismo.
- Con una adecuada calibración del sensor y unas condiciones del entorno favorables es posible obtener mediciones de gran precisión, tanto para las rotaciones como para las posiciones relativas al sensor en los tres ejes del espacio.
- El correcto funcionamiento del sistema no depende de si el entorno está abierto o cerrado al exterior, sólo de las condiciones de iluminación del mismo.
- Aunque no se han contemplado otras medidas, un sensor de visión es capaz de obtener más tipos de datos del exterior, lo que ofrece más posibilidades de cara a mejoras futuras del sistema.

En concreto, el sensor elegido por su compatibilidad directa con la Raspberry ha sido la *Raspberry Pi Camera* de 5 megapíxeles, que se muestra en la Figura 6.

Para poder utilizarla es necesario contar con un intérprete de Python e instalar el módulo "picamera[array]":

```
$ pip install "picamera[array]"
```

Con este módulo sería suficiente para poder acceder a las funciones básicas de la cámara, pero para poder obtener posiciones y rotaciones relativas hay que acudir a módulos más completos como OpenCV y Aruco.



Figura 6. Raspberry Pi Camera de 5 megapíxeles con carcasa protectora

4.2. OpenCV y Aruco

4.2.1. OpenCV

OpenCV (*Open Source Computer Vision Library*) es una librería de software open-source concebida para proporcionar una infraestructura común en aplicaciones de visión por computador, y contiene más de 2500 funciones y algoritmos optimizados ([18], [19]). Debido a esta enorme cantidad de información, el espacio de memoria que ocupa en la Raspberry es considerable, y lleva a que para su utilización se necesite una tarjeta SD de al menos 16GB. La instalación de OpenCV es costosa también en tiempo, ya que requiere instalar previamente una serie de paquetes de software con los que se establecerán dependencias.

A continuación se indican los comandos que habría que introducir para instalar todas las dependencias necesarias, detallando brevemente el propósito de cada una:

```
$ # Actualizar los paquetes actualmente instalados:
$ sudo apt-get update
$ sudo apt-get upgrade
$ # Herramientas para configurar la instalación:
$ sudo apt-get install -y build-essential cmake pkg-config
$ # Paquetes necesarios de entrada y salida de imagen y video:
$ sudo apt-get install -y libjpeg-dev libtiff5-dev libjasper-dev
libpng12-dev
$ sudo apt-get install -y libavcodec-dev libavformat-dev libswscale-dev
libv4l-dev
$ sudo apt-get install -y libxvidcore-dev libx264-dev
```

```
$ # Libreria necesaria para compilar algunos modulos de OpenCV:
$ sudo apt-get install -y libgtk2.0-dev
$ # Librerias de optimización:
$ sudo apt-get install -y libatlas-base-dev gfortran
$ # Instalacion o actualizacion de las versiones de Python 2.7 y 3:
$ sudo apt-get install -y python2.7-dev python3-dev
```

Para obtener más información sobre estos paquetes se debe acceder a la documentación que facilita OpenCV desde la página web de la librería.

Tras instalar las dependencias es necesario reiniciar la Raspberry, ya que de lo contrario pueden aparecer problemas en los pasos posteriores. Después, se puede proceder a la instalación propiamente dicha de OpenCV:

```
$ cd ~
$ git clone https://github.com/opencv/opencv.git
$ git clone https://github.com/opencv/opencv_contrib.git
$ pip install numpy
$ cd ~/opencv
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
$       -D CMAKE_INSTALL_PREFIX=/usr/local \
$       -D INSTALL_PYTHON_EXAMPLES=ON \
$       -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
$       -D BUILD_EXAMPLES=ON ..
$
$ make -j1
$ sudo make install
$ sudo ldconfig
```

Hay que resaltar que debido a que han aparecido problemas al utilizar más de una cpu, ha habido que llevar a cabo la instalación solamente con una (`$ make -j1`), lo que ha alargado considerablemente el tiempo de instalación.

Tras el proceso de instalación se ha comprobado que las versiones instaladas de Python y OpenCV han sido, respectivamente, la 2.7.13 y la 3.3.0. Todo el desarrollo posterior se ha realizado sobre estas versiones.

4.2.2. Aruco

Uno de los módulos llamados “contribuidos” que viene incluido con OpenCV es Aruco [20]. Aruco ha sido desarrollado por el grupo “Aplicaciones de la Visión Artificial” del Departamento de Computación y Análisis Numérico de la Universidad de Córdoba, y proporciona una serie de marcadores visuales 2D junto un método que permite localizarlos en el espacio y referenciarlos al sistema de coordenadas de la cámara ([21], [22]). El sistema de coordenadas tanto de uno de los marcadores que se ha utilizado como el de la cámara se muestran conjuntamente en la Figura 7.

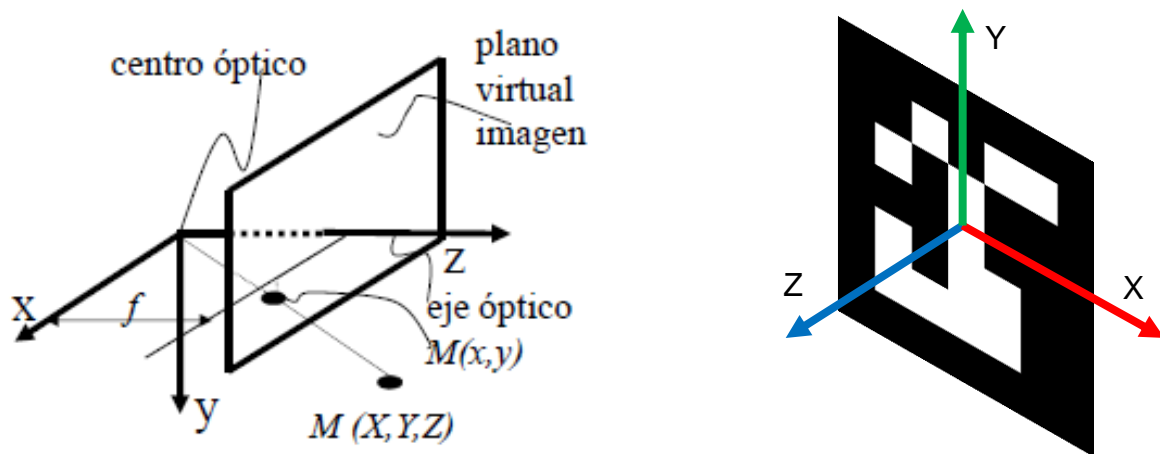


Figura 7. Sistema de coordenadas según el modelo de cámara oscura (pin-hole) [23] a la izda., y sistema de coordenadas de un marcador Aruco a la dcha.

El proceso de detección de los marcadores se basa principalmente en obtener secuencialmente los bordes exteriores de los marcadores, los vértices donde se unen y el código que indican en el patrón interno. Durante este proceso se van descartando potenciales candidatos que no cumplen los requisitos de número mínimo de puntos, número de vértices, distancia a otros candidatos, etc. Dentro del amplio catálogo de marcadores que se suministra se pueden encontrar distintas familias o “diccionarios” que agrupan marcadores con un determinado número de filas y columnas de celdas internas (5x5, 6x6,...), pero salvo la diferente codificación de cada marcador todos proporcionan la misma información y se utilizan de la misma manera. En cualquier caso, debido a cómo se generan los diccionarios y a cómo funcionan los algoritmos de detección, se recomienda utilizar marcadores de familias con pocos miembros si en la aplicación se va a utilizar un número reducido de marcadores. Otra cuestión importante que hay que mencionar es que debe haber un espacio en blanco a partir del borde negro del marcador para que los marcadores sean reconocidos por el software de forma correcta.

Una vez que los marcadores son detectados, se pueden obtener los parámetros extrínsecos de los marcadores respecto a la cámara, que son el vector de rotación y el de traslación del sistema de coordenadas de cada marcador con respecto al de la cámara,

pero para ello hay que suministrar los parámetros intrínsecos de la cámara. Para obtener estos parámetros es necesario llevar a cabo una calibración de la cámara.

4.3. Calibración de la cámara

Cualquier cámara que se utilice para una aplicación de visión debe ser calibrada para conocer de qué manera deforma las dimensiones de aquello que capta.

Se pueden considerar dos tipos de distorsión: radial, causada por imperfecciones de las lentes y que hace que la magnificación de la imagen cambie según se recorren los ejes ópticos, y tangencial, que produce un descentramiento al no estar las lentes perfectamente paralelas al plano de la imagen. Para corregir estos efectos hay que obtener los coeficientes de distorsión (que en OpenCV son 5), que cuantifican numéricamente la distorsión, y la matriz de la cámara, que contiene las distancias focales y las coordenadas del centro óptico. Una vez obtenidos estos parámetros intrínsecos se pueden utilizar siempre que se necesite mientras no se modifiquen las lentes del sensor.

En este caso la calibración de la cámara se ha llevado a cabo con el método del tablero de ajedrez, según el cual se comparan las dimensiones conocidas de los cuadros (30mm) con las obtenidas en la proyección de la imagen. Modificando convenientemente el programa que proporciona OpenCV para realizar la calibración con este método, se ha buscado un patrón de 8x6 cuadros en cada una de las 30 imágenes tomadas para distintas distancias y orientaciones del tablero respecto a la cámara. A modo de ejemplo, se muestra una de ellas junto con su homóloga corregida tras haber obtenido la matriz de la cámara y los coeficientes de distorsión en la Figura 8.

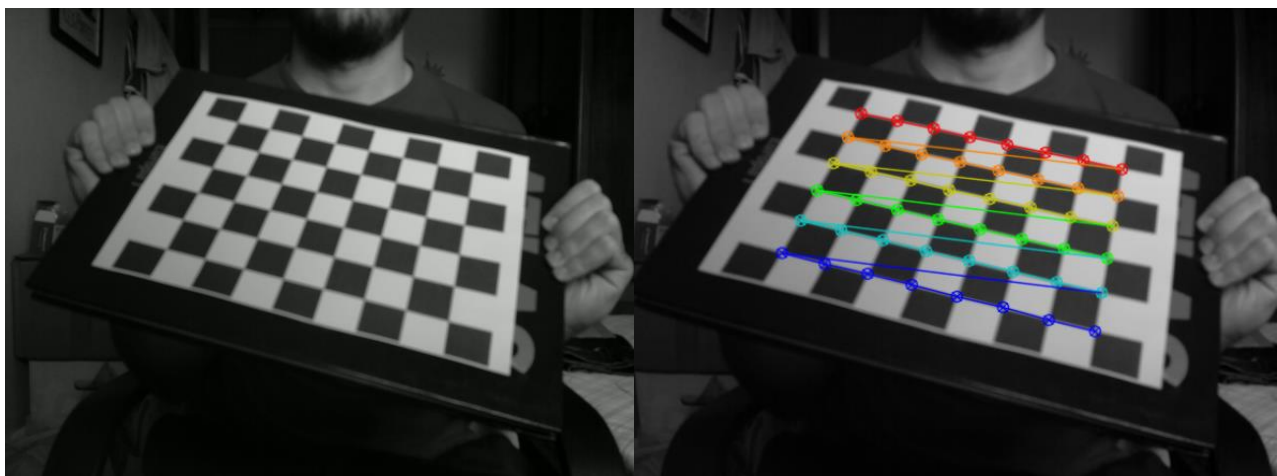


Figura 8. Captura del tablero de ajedrez original (izda.) junto con su homóloga corregida y con el patrón detectado (dcha.)

Tras realizar la calibración, los parámetros intrínsecos de la cámara que se han obtenido han sido los siguientes:

$$m = \begin{bmatrix} 617.89837178 & 0.0 & 333.30257998 \\ 0.0 & 621.41871986 & 234.21505625 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \quad (2)$$

$$distCoefs = [2.54935234E-1 \quad -1.80502677 \quad 1.31060766E-3 \quad 9.54638765E-3 \quad 4.98161962] \quad (3)$$

Como comprobación de que la calibración se ha realizado correctamente, se han colocado dos marcadores Aruco con sus centros a 300mm aproximadamente y se ha calculado su separación con las posiciones relativas a la cámara devueltas por la función *estimatePoseSingleMarkers(...)*. En condiciones no óptimas de iluminación se ha obtenido una distancia redondeada de 309mm, lo que implica una precisión cercana al centímetro. Para el presente caso esta precisión se ha considerado suficiente.

En la Figura 9 se muestra el dirigible con el marcador Aruco acoplado a la parte delantera.



Figura 9. ZEP-AIR con un marcador Aruco acoplado en la parte delantera

5. Sistema de control

El núcleo del sistema de navegación es el esquema de control, que es el que, a partir de las lecturas del estado del sistema proporcionadas por los sensores, calcula las acciones que deben realizar los actuadores. En este apartado se recogen los sistemas de control que se han considerado para los modelos cinemático y dinámico del dirigible.

5.1. Modelo cinemático y control proporcional

Tal y como se mencionaba en el apartado 2.3, al observar el comportamiento del dirigible a velocidades bajas se ha determinado, como primera aproximación, que los efectos dinámicos son poco representativos, y se ha desarrollado un esquema de control proporcional basado en la cinemática del mismo. Este esquema está basado en los que se utilizan para robots terrestres de dos ruedas ([24], [25]).

Además, debido a las características del sistema motriz del dron, se ha considerado que el movimiento en el plano horizontal se encuentra desacoplado con respecto al del eje vertical. El diagrama de la Figura 10 muestra cómo se han definido las velocidades en el modelo en el plano X-Y.

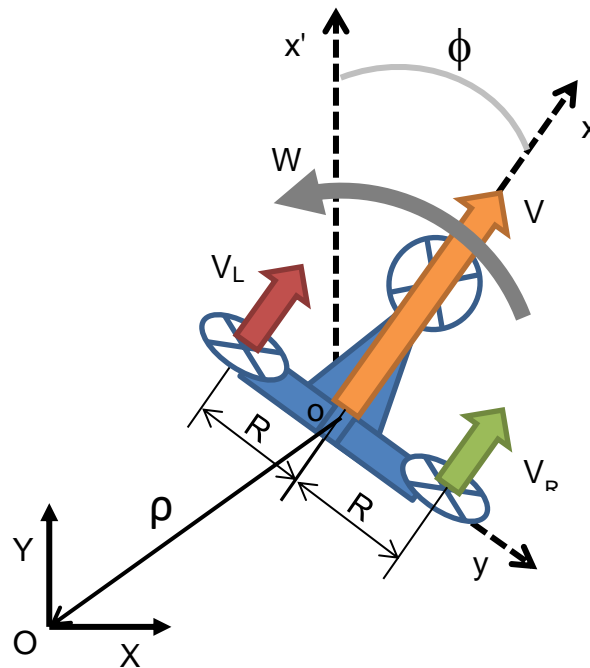


Figura 10. Diagrama de velocidades y distancias principales del plano X-Y consideradas en el modelo cinemático

Las velocidades representadas son: V_R y V_L , que se refieren respectivamente a la velocidad inducida por el motor derecho e izquierdo; V , que se define como la velocidad media en el eje longitudinal de la góndola (ver ec. (4)) y W , que es la velocidad angular de

la góndola (ver ec. (5)). R es la distancia entre el centro de gravedad o del dirigible (que se toma como origen del sistema de coordenadas local del dirigible) y cada uno de los motores laterales, y ρ es la distancia entre el centro de gravedad y el origen del sistema de coordenadas objetivo O .

$$V = \frac{V_R + V_L}{2} \quad (4)$$

$$W = \frac{V_R - V_L}{R} \quad (5)$$

A las velocidades anteriores hay que añadir la que se induce en el eje vertical, V_z , que viene determinada por altura en el eje vertical h , distancia en ese eje entre o y O .

Por otro lado, las ecuaciones que se plantean según el esquema de control proporcional son las siguientes:

$$V = K_V \rho \quad (6)$$

$$W = K_W \phi_{\text{objetivo}} \quad (7)$$

$$V_z = K_z h \quad (8)$$

donde K_V , K_W y K_z son constantes de control, y ϕ_{objetivo} es el ángulo que tiene que girar el dirigible hasta que su eje x coincida con la dirección marcada por ρ .

Finalmente, igualando (4) y (6) por un lado y (5) y (7) por otro, y resolviendo el sistema que se obtiene, se llega a las ecuaciones para V_R y V_L , que son las salidas hacia los motores junto con V_z :

$$V_L = K_V \rho - \frac{K_W R \phi_{\text{objetivo}}}{2} \quad (9)$$

$$V_R = 2 K_V \rho - V_L \quad (10)$$

Es necesario mencionar que dado que no se tiene acceso directo a los motores, sino a través de los comandos que entiende el receptor de la góndola, ha habido que utilizar estas velocidades calculadas por el controlador de la siguiente forma:

Caso 1: $V_R > V_L \rightarrow$ se envía al dirigible el comando KEY_LEFT (giro anti-horario)

Caso 2: $V_R < V_L \rightarrow$ se envía al dirigible el comando KEY_RIGHT (giro horario)

Caso 3: $V_R \approx V_L \rightarrow$ se envía al dirigible el comando KEY_FRONT (avance hacia delante)

5.2. Modelo dinámico y control en espacio de estados con integradores

Tras haber realizado pruebas de funcionamiento con el anterior sistema de control, se ha llegado a la conclusión de que los efectos dinámicos pueden ser más relevantes de lo que se pensaba en un primer momento. Por ello se ha propuesto un esquema de control en espacio de estados con integradores sobre el modelo dinámico del dirigible. A diferencia del control clásico, el control en espacio de estados está considerado como un control de mayor calidad debido a que se realimentan todas las variables de estado, no sólo aquellas que se quiere controlar. Debido a que en este caso no será posible medir todas las variables es necesario introducir observadores de estado, de forma que éstos estimen las que no es posible medir a partir de las variables medidas.

5.2.1. Modelo dinámico del dirigible

En la Figura 11 se muestra el diagrama de fuerzas que actúan sobre el dirigible, donde se pueden observar F_R , F_L y F_Z , que son el empuje generado por cada uno de los motores, $m \cdot g$, que es el peso del dirigible y F_D , que es la fuerza de arrastre. Planteando el equilibrio de fuerzas y de momento actuante en el eje vertical, M_z , se llega a las siguientes ecuaciones:

$$m\ddot{z} = F_Z - m \cdot g \quad (11)$$

$$m\ddot{x} = F_R + F_L - F_D \quad (12)$$

$$M_z = (F_R - F_L) \cdot R \quad (13)$$

donde m es la masa del dirigible y \ddot{z} y \ddot{x} las aceleraciones en los ejes z y x respectivamente.

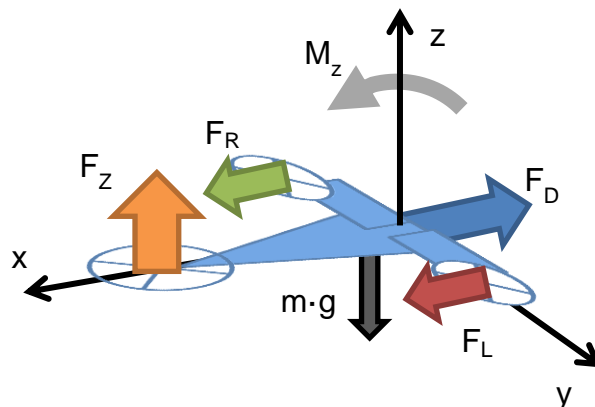


Figura 11. Diagrama de fuerzas actuantes sobre el dirigible

La fuerza de arrastre se puede representar por medio de la siguiente ecuación [26]:

$$F_D = \frac{1}{2} \rho_{aire} \dot{x}^2 C_D A \quad (14)$$

donde ρ_{aire} es la densidad del aire (en este caso, a 20°C y 1 atm), \dot{x} la velocidad en el eje x, C_D el coeficiente de arrastre y A el área frontal expuesta al fluido.

Por otra parte, el momento en el eje vertical se puede calcular con las ecuaciones de rotación de Euler y considerando únicamente velocidad angular en el eje z:

$$M_z = I_z \ddot{\phi} \quad (15)$$

donde I_z es el momento de inercia y $\ddot{\phi}$ la aceleración angular, ambos en z.

Sustituyendo (14) en (12), (15) en (13), y despejando se obtiene lo siguiente:

$$\ddot{z} = \frac{F_z}{m} - g \quad (16)$$

$$\ddot{x} = \frac{F_R + F_L}{m} - \frac{1}{2m} \rho_{aire} \dot{x}^2 C_D A \quad (17)$$

$$\ddot{\phi} = \frac{R(F_R - F_L)}{I_z} \quad (18)$$

Eligiendo como variables de estado x , \dot{x} , z , \dot{z} , ϕ y $\dot{\phi}$ sucede que el sistema obtenido es no lineal, por lo cual es necesario llevar a cabo una linealización en torno al punto de equilibrio. A continuación se muestra el sistema linealizado con variables incrementales (variable – variable en el punto de equilibrio):

$$\ddot{Z} = \frac{F'_Z}{m} \quad (19)$$

$$\ddot{X} = \frac{F'_R + F'_L}{m} - \left(\frac{1}{m} \rho_{aire} \dot{x}_0 C_D A \right) \dot{X} \quad (20)$$

$$\ddot{\Phi} = \frac{R(F'_R - F'_L)}{I_z} \quad (21)$$

donde \ddot{Z} , \ddot{X} , $\ddot{\Phi}$, F'_Z , F'_R y F'_L son las variables incrementales y \dot{x}_0 es la velocidad según x en el punto de equilibrio.

Ahora, expresando el sistema linealizado según la forma canónica con las matrices A, B y C se tiene:

$$\begin{bmatrix} \dot{X} \\ \ddot{X} \\ \dot{Z} \\ \ddot{Z} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} = A \begin{bmatrix} X \\ \dot{X} \\ Z \\ \dot{Z} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + B \begin{bmatrix} F'_R \\ F'_L \\ F'_Z \end{bmatrix} \quad (22)$$

$$\begin{bmatrix} X \\ Z \\ \Phi \end{bmatrix} = C \begin{bmatrix} X \\ \dot{X} \\ Z \\ \dot{Z} \\ \Phi \\ \dot{\Phi} \end{bmatrix} \quad (23)$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (24)$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 1/m & 1/m & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/m \\ 0 & 0 & 0 \\ R/I_z & -R/I_z & 0 \end{bmatrix} \quad (25)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (26)$$

$$a = -\frac{1}{m} \rho_{aire} \dot{x}_0 C_D A \quad (27)$$

5.2.2. Control con integradores en serie y estimador de estado

Una vez que se han compuesto las matrices canónicas del modelo se puede diseñar el control con integradores propiamente dicho. El esquema de este sistema de control con observador de estado se indica en la Figura 12.

En el esquema se muestran las distintas entradas y salidas involucradas en el instante “k”, siendo “r” la referencia o estado de las variables deseado, la “e” el error entre el estado alcanzado y el deseado, la “x_R” el estado del regulador, la “u” las entradas al sistema, la “y” las salidas del mismo y la “x_{est}” las variables de estado estimadas a partir de las que se han podido medir.

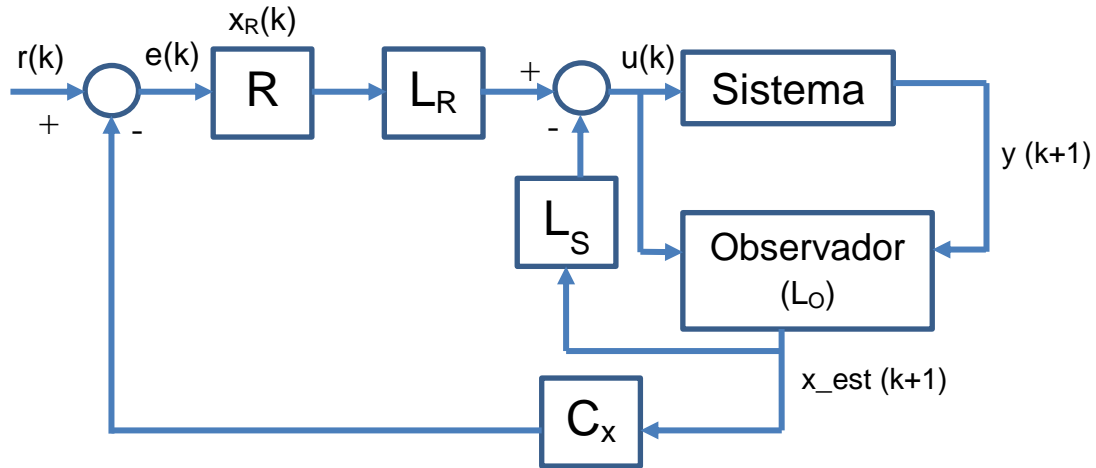


Figura 12. Esquema de control con integradores en serie y observador de estado

La integración del error, que es la que permite obtener un error de posición nulo, se realiza desde las matrices L_R y L_S . Para obtener estas matrices hay que situar en los ejes real e imaginario los polos que reproducirán el comportamiento que se busca, para lo cual se ha fijado el tiempo de respuesta t_r y se ha obtenido la parte real del polo, σ , con el que se obtiene ese tiempo de respuesta:

$$\sigma = 4.75/t_r \quad (28)$$

La expresión anterior es válida para un sistema un polo doble dominante, por lo que los polos del sistema, que son uno por cada variable de estado y otro por cada integrador, se han definido de la siguiente manera:

$$\begin{aligned} polos_{sist} = [& e^{(-T_m \sigma)} \quad e^{(-T_m \sigma + 0.00001)} \quad e^{(-5 T_m \sigma)} \quad e^{(-5 T_m \sigma - 0.001)} \quad e^{(-6 T_m \sigma)} \\ & e^{(-6 T_m \sigma - 0.001)} \quad e^{(-7 T_m \sigma)} \quad e^{(-7 T_m \sigma - 0.001)} \quad e^{(-8 T_m \sigma)}] \end{aligned} \quad (29)$$

donde T_m es el periodo de muestreo del estado.

De esta forma dos polos están en la posición deseada y el resto se consideran dominados al estar a una distancia mayor o igual a 5σ .

Después, hay que construir las matrices F_t y G_t de la siguiente forma:

$$F_t = \begin{bmatrix} F_z & 0 \\ -G_r * C_x & F_r \end{bmatrix} \quad (30)$$

$$G_t = \begin{bmatrix} G_z \\ 0 \end{bmatrix} \quad (31)$$

$$Cx = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (32)$$

donde F_z y G_z son, respectivamente, las matrices A y B discretizadas con T_m , G_r y F_r son matrices unidad por ser control con integradores, y Cx es la matriz selectora de las variables de estado que forman la referencia.

Finalmente, la matriz L_i de la que se extraen L_R y L_S se puede obtener de diversas formas, pero en este caso se ha optado por utilizar la función *place* de Matlab, la cual calcula la matriz de ganancia que en bucle cerrado lleva a los polos del sistema a la posición deseada [27]:

$$L_i = place(F_t, G_t, polos_{sist}) \quad (33)$$

En el presente caso, para que las dimensiones sean consistentes, las primeras 6 columnas de la matriz L_i serían L_s , y L_r estaría compuesta por las tres siguientes con signo negativo.

Con respecto al observador de estado, éste puede ser de dos tipos: predictor o estimador. En este caso se ha optado por implementar un estimador de estado debido a que los estimadores funcionan mejor que los predictores cuando la caracterización de los parámetros del sistema no es demasiado precisa. El estimador de estado, a diferencia del predictor, calcula la estimación en el instante actual $k+1$ con la predicción del instante anterior k .

El cálculo del estimador es muy similar al de las matrices del control integral, con la excepción de que el tiempo de respuesta tr_o utilizado para calcular los polos del mismo se considera que es la mitad del tiempo de respuesta del caso anterior. Esto se hace con el propósito de conseguir una dinámica el doble de rápida que la del regulador, que no afecte al comportamiento del mismo. Los polos obtenidos con la nueva σ_o serían:

$$\begin{aligned} polos_{est} = & [e^{(-T_m \sigma_o)} \quad e^{(-T_m \sigma_o + 0.00001)} \quad e^{(-5 T_m \sigma_o)} \quad e^{(-5 T_m \sigma_o - 0.001)} \quad e^{(-6 T_m \sigma_o)} \quad e^{(-6 T_m \sigma_o - 0.001)}] \end{aligned} \quad (34)$$

Nuevamente se tendrían 2 polos dominantes y, en este caso, 4 dominados; en total, 1 por cada variable de estado del sistema.

Por último, la ley de observación se ha calculado de nuevo con la función *place* de Matlab:

$$L_o = place(F'_z, F'_z C', polos_{est}) \quad (35)$$

donde F'_z y C' son matrices transpuestas.

Cabe mencionar que este sistema de control sobre el modelo linealizado sólo sería válido para variaciones pequeñas de las variables de estado respecto al punto de equilibrio.

6. Resultados

A continuación, se presentan resultados experimentales utilizando el sistema de control diseñado con el dirigible. Para comprobar en profundidad las características del modelo de espacio de estados presentado, también se realiza una evaluación del mismo en simulación. El código en lenguaje Python utilizado en las pruebas puede consultarse en el Anexo I.

6.1. Pruebas de funcionamiento del controlador proporcional

6.1.1. Montaje experimental

Para evaluar el desempeño del controlador proporcional diseñado a partir del modelo cinemático se realizó un primer montaje en el cual la Raspberry Pi se situó en el suelo con la cámara en posición vertical, con el eje óptico “z” mirando hacia arriba. El marcador Aruco, impreso en un papel de alta rigidez para minimizar la deformación debida al peso propio, se acopló a la parte inferior de la góndola del dirigible de forma que el eje “z” del marcador apuntase hacia la cámara y el “x” apuntase en dirección a la parte frontal del dirigible (por cuestiones de cómo se seleccionaron las referencias espaciales).

Con esta disposición de los elementos, que es básicamente la que se aprecia en la Figura 1, el objetivo de navegación era alcanzar un “área deseada” con un radio de 10 cm en el plano horizontal y a una determinada altitud sobre la cámara, desde un punto cualquiera dentro del campo de visión de la cámara.

Debido a que las pruebas realizadas con este montaje resultaron poco satisfactorias, principalmente por el reducido ángulo de captación de la cámara y por el comportamiento vertical del dirigible (mucho menos preciso que en el plano, como se demostrará después), se decidió modificarlo y afrontar un nuevo objetivo de navegación, más centrado esta vez en el movimiento horizontal.

En el segundo diseño de montaje la cámara se coloca en horizontal, de forma que el eje óptico “z” sea paralelo al plano del suelo y el “y” perpendicular al mismo, y con respecto al marcador visual, impreso esta vez en papel normal (lo que reduce el peso y la inercia), éste se acopla ahora a la parte frontal del dirigible de forma que el eje “y” del marcador apunte hacia el suelo y el “z” hacia e (ver Figura 9). En la Figura 13 se muestra de forma orientativa el segundo diseño.

En este caso, se ha establecido como objetivo de navegación que el dirigible alcance la posición de la cámara, orientándose y avanzando hacia el centro óptico de la misma desde una posición y con una orientación cualesquiera dentro del campo de visión de la cámara. Cabe mencionar que la orientación, que se produce girando alrededor del

eje “y”, se da entre unos límites más allá de los cuales no es posible el reconocimiento visual del marcador. Las pruebas se dan por finalizadas cuando el marcador deja de ser reconocido por la cercanía del dirigible a la cámara.

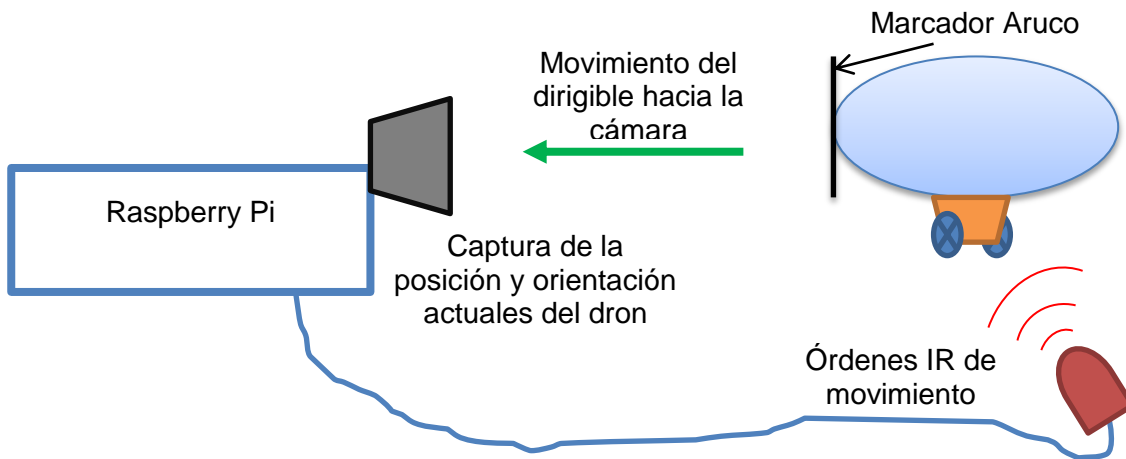


Figura 13. Esquema de la segunda versión del montaje experimental

6.1.2. Resultados obtenidos

Para valorar el comportamiento del esquema de control en el dirigible se ha llevado a cabo una serie de 15 pruebas de funcionamiento en las que se han registrado el error de posición en el plano hasta el objetivo, el error de orientación angular, el error de altitud y las tres velocidades V_R , V_L y V_Z con respecto al tiempo de avance del experimento. A continuación se indican los datos obtenidos.

En la gráfica de la Figura 14 se muestran los datos obtenidos de error de posición en el plano hasta el objetivo con respecto al tiempo. Se puede diferenciar en las curvas un primer tramo en el que la variación del error es menor debido a que el dirigible se está orientando antes de avanzar hacia la cámara, y un segundo tramo en el que la velocidad de disminución del error es mayor cuando el dirigible avanza ya orientado. La duración de las pruebas es en el 67% de los casos menor de 25 segundos, aunque el máximo llega a los 40.7s en la prueba 14. Con respecto al error de posición, el error final máximo se encuentra (excluyendo la prueba 9) en la prueba 5, y es de 739.77mm, aunque en la mayoría de los casos está alrededor del medio metro y llega a un mínimo de 241.15mm en la prueba 1. Destaca sobre el resto de repeticiones la prueba 9, en la que la tendencia del error es al principio decreciente para después aumentar rápidamente. Esto es debido a que en un determinado momento del ensayo, a los 22s aproximadamente, el dirigible reconoció un comando de forma errónea y avanzó marcha atrás (movimiento posible pero no considerado en el control) a gran velocidad, hasta dejar de ser captado por la cámara. A pesar de ello, en todos los demás casos el error sigue una tendencia decreciente.

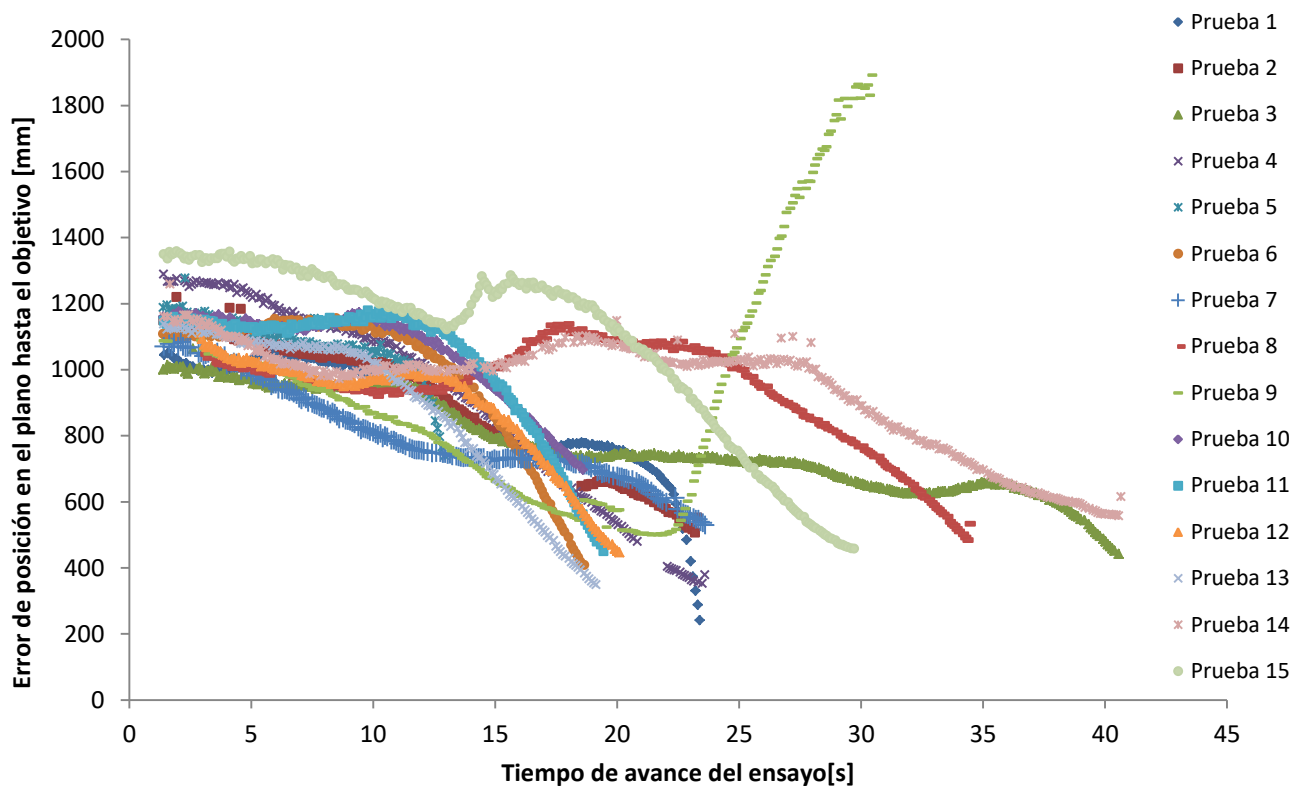


Figura 14. Distancia en el plano (X-Z) hasta el objetivo frente al tiempo de avance del ensayo

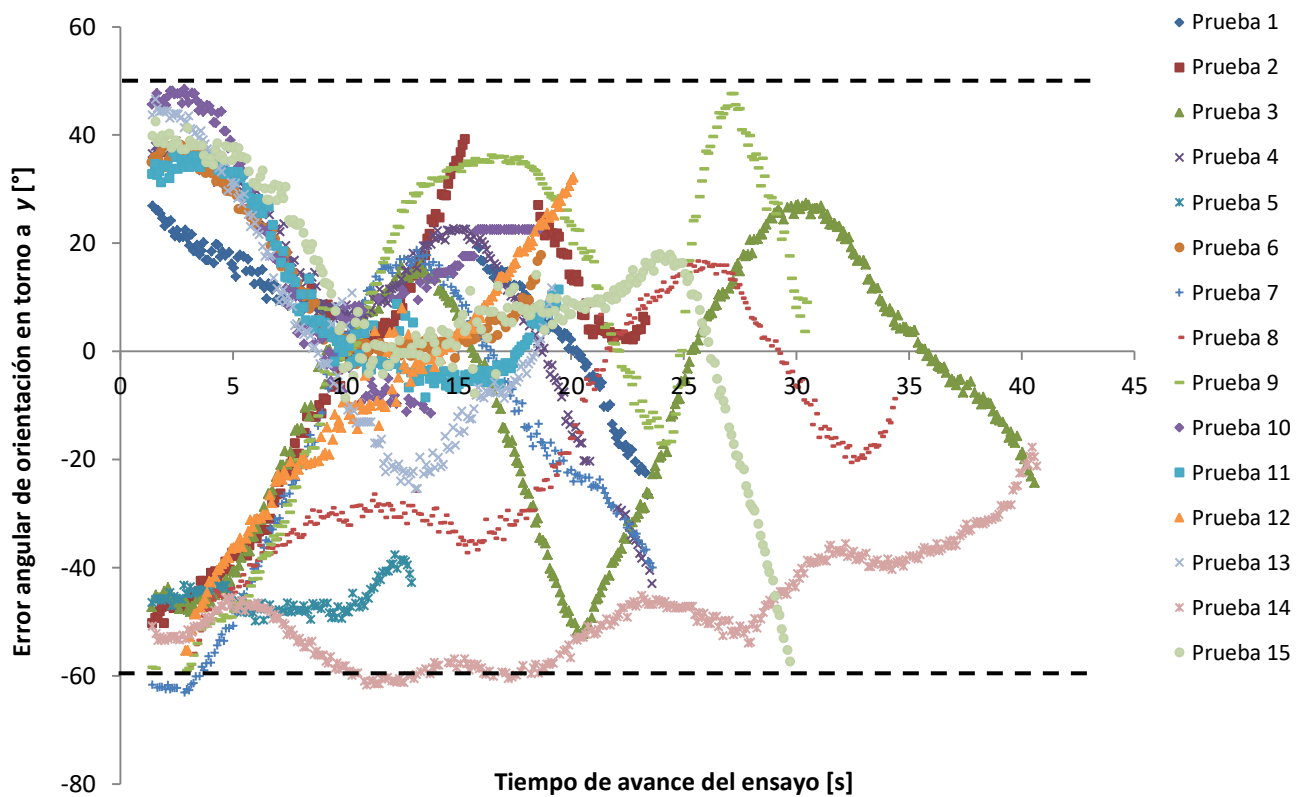


Figura 15. Error angular de orientación en torno a y frente al tiempo de avance del ensayo

En la Figura 15 se representa el error angular de orientación frente al tiempo. Se puede ver como la evolución es mucho más diversa entre pruebas que en el caso anterior, pudiendo identificarse desde tendencias mayormente monótonas en algunos casos hasta comportamientos oscilantes, como el de la prueba 3. Esto puede ser debido entre otros factores a la inercia del eje vertical, no considerada en el modelo cinemático pero cuya influencia es, presumiblemente, relevante y también a que la medición con Aruco del ángulo no es tan precisa como la posición. Esto último se puede comprobar en la dispersión observada en los datos pertenecientes a una misma prueba, que cubren un área mayor que los del error de posición en el plano. Es remarcable también el hecho de que en los primeros instantes de tiempo, hasta el segundo 10 aproximadamente, el error angular sigue en prácticamente todos los casos una tendencia decreciente partiendo tanto de los ángulos iniciales positivos como de los negativos. Si comparamos estos resultados con los anteriores se puede deducir que la inercia hace que en ciertas ocasiones el dirigible avance de forma lateral. Este sería el caso de la prueba 15, en la que el error de posición en el plano disminuye continuamente mientras que el error angular aumenta. Además, se puede ver cómo el rango de detección de la orientación en torno a “y” está entre 50 y -60 grados.

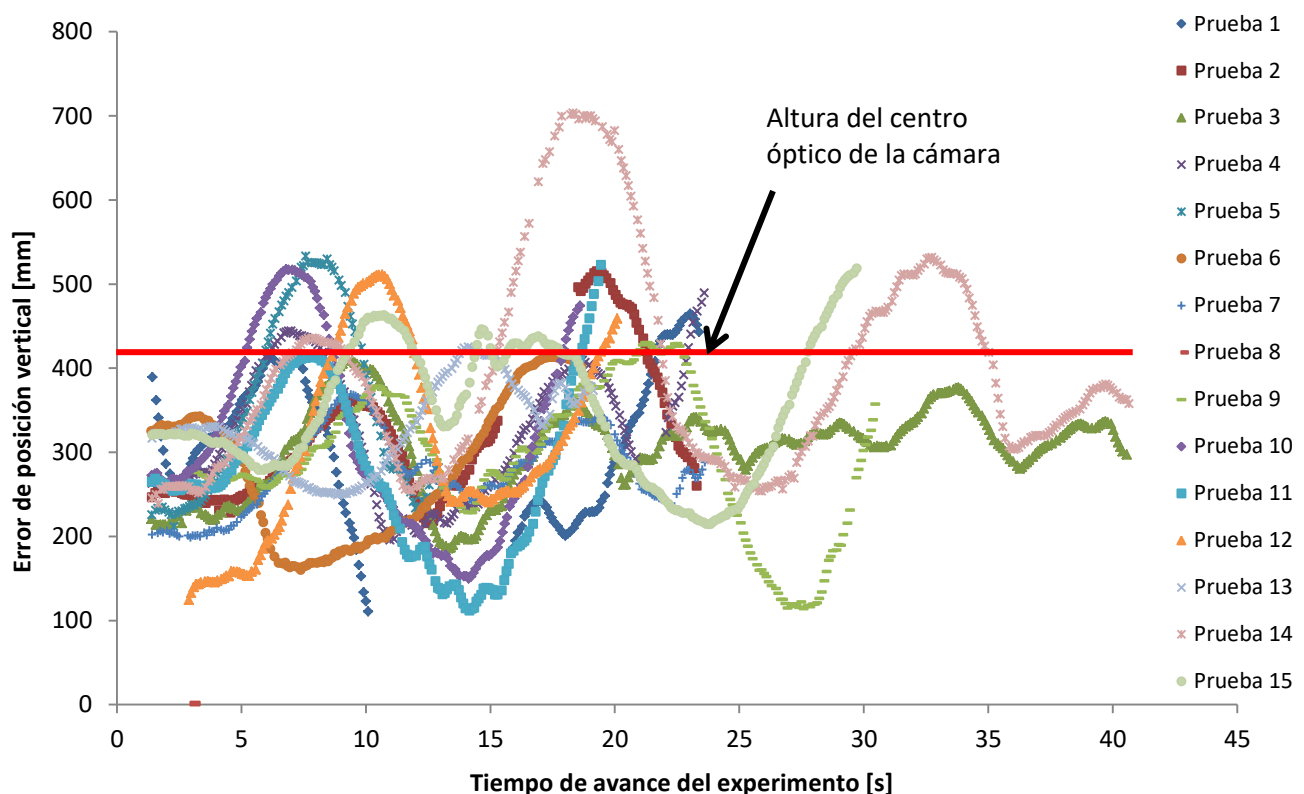


Figura 16. Error de posición vertical frente al tiempo de avance del ensayo

Finalmente se muestra en la Figura 16 el error de altitud a lo largo del tiempo. Se ha incluido en la gráfica una línea horizontal que indica la altura a la que se sitúa el centro óptico de la cámara, que como se puede observar queda la mayor parte del tiempo por debajo de la

posición del dirigible en todas las pruebas. Esto se debe a que ha habido que ajustar el empuje vertical subiendo por encima del centro de la cámara la altura objetivo (hasta 410 mm), de forma que el dirigible se mantuviese el mayor tiempo posible dentro del rango visual de la cámara. Se puede ver también cómo la tendencia del error es oscilante en todos los casos, hecho que se explica teniendo en cuenta que la regulación de la altura está altamente discretizada, dejando solo 5 niveles de potencia como grados de libertad, y que los efectos dinámicos no son despreciables en el eje vertical. Las oscilaciones están acotadas tanto por encima como por debajo de la línea del centro de la cámara unos 300mm.

Cabe mencionar que las discontinuidades observadas en los datos, en algunos casos, son debidas a que la cámara ha dejado de captar el marcador, bien porque se ha producido un giro fuera de los límites de detección, o porque el dirigible se ha elevado más allá del rango visual en vertical o porque el software ha sido afectado por las condiciones de iluminación del entorno.

6.1.3. Fuentes de error

Las fuentes de error en este caso son muy numerosas, aunque las más relevantes podría considerarse que son las siguientes:

- **Peso del dirigible y energía de la batería variables:** La masa del dirigible está en constante cambio debido a la pérdida paulatina de helio, que escapa a través de las paredes del globo, lo que hace que el control de la posición vertical tenga que ser ajustado constantemente. Un sistema de control con rechazo de perturbaciones podría contrarrestar este efecto. Por otro lado, la batería del dirigible no sólo tiene una duración muy reducida, sino que además con el transcurso del tiempo la potencia que transmite a los motores va disminuyendo, lo que afecta a la velocidad que pueden alcanzar.
- **Restricciones debidas a los actuadores:** Dado que la acción sobre los actuadores está limitada a las órdenes que se permiten desde el mecanismo interno del dirigible, el control no puede aplicar directamente las salidas calculadas, lo que perjudica al desempeño del mismo. Además los comandos IR que se envían al dirigible desde la Raspberry son aproximados al haber tenido que grabarlos con unos dispositivos cuya precisión no es absoluta, lo que en ocasiones lleva a que sean malinterpretados por el receptor del dirigible o a que directamente no sean recibidos.
- **Iluminación:** La iluminación es un factor que puede llegar a perjudicar tanto por exceso a la transmisión de los pulsos IR (tal y como se indicaba en el apartado 3.1) como por defecto a la identificación del marcador por parte de la cámara. Dado que la evolución es inversa entre los dos sistemas ha habido que buscar una solución

intermedia que sin lugar a dudas no ha sido la óptima que se podría conseguir con pruebas específicas.

- Otros factores del entorno: La temperatura del entorno, la altitud a la que se realizan los ensayos y las corrientes de aire que atraviesan el lugar por el cual se mueve el dirigible son circunstancias que afectan a la navegación, ya sea por modificar el empuje que puede generar el globo de helio o por perturbar el estado más allá de la capacidad de reacción de los actuadores.

6.2. Simulación del controlador en espacio de estados

6.2.1. Parámetros del sistema

Los parámetros cuyos valores se deben obtener e introducir en el modelo para calcular los valores numéricos de las matrices son, como se indicaba en el apartado 5.2.1, la masa del dirigible, la densidad del aire, el coeficiente de arrastre del dirigible, el área frontal del mismo expuesta al flujo, el momento de inercia en el eje vertical y la distancia entre los motores y el centro de la góndola.

Como masa del dirigible se han considerado 31g, el peso aproximado de la góndola, menos para el caso vertical, donde se ha considerado que la mayor que puede tener sería, debido a la compensación del peso por el empuje del globo, igual a una ficha de lastre, cuya masa es de 1g. La densidad del aire se ha tomado para 20°C y 1atm, lo que da un valor de 1.204kg/m³. Con respecto al coeficiente de arrastre se ha establecido como 0.15, semejante al que presentan los balones de fútbol americano, cuya forma se podría aproximar a la del dirigible. El área frontal expuesta se ha calculado como la de una elipse, y su valor es de 0.116m². Por otro lado, el momento de inercia en “z” se ha calculado según el teorema de los ejes paralelos de Steiner, considerando los motores como tres masas puntuales de igual valor que agrupan la masa total del dirigible, y se ha obtenido un resultado de 6.675e-05kg·m². Finalmente, se ha medido la distancia entre cada uno de los motores y el centro de la góndola, para llegar a una longitud de 35mm.

Con estos parámetros se han calculado las matrices A y B del sistema:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -6.7719E-4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (36)$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 32.2581 & 32.2581 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1000 \\ 0 & 0 & 0 \\ 524.3446 & -524.3446 & 0 \end{bmatrix} \quad (37)$$

Las matrices se han discretizado después con un periodo de muestreo de 0.1 segundos, similar al tiempo de ciclo de la captura y procesamiento de imágenes.

6.2.2. Controlabilidad y observabilidad

Antes de abordar el diseño del sistema de control en espacio de estados es necesario comprobar que el sistema es controlable y observable, para lo cual hay que construir en primer lugar las matrices de controlabilidad y observabilidad.

La matriz de controlabilidad se expresa de la siguiente forma:

$$\xi = [B \ AB \ \dots \ A^{n-1}B] \quad (38)$$

donde n es el número de variables de estado.

Por otra parte, la matriz de observabilidad se indica a continuación:

$$\theta = \begin{bmatrix} C \\ C \ A \\ \vdots \\ C \ A^{n-1} \end{bmatrix} \quad (39)$$

Una vez que se han obtenido las dos matrices anteriores, se dice que el sistema es controlable si ξ tiene rango completo, y este mismo criterio es aplicable a θ para decir que el sistema es observable.

En el caso abordado, se han obtenido una matriz ξ de dimensiones 6x18 y una matriz θ 18x6, que debido a su gran tamaño no se recogen en este apartado. Ambas han resultado tener rango = 6, por lo que se puede afirmar que el sistema modelado es controlable y observable.

6.2.3. Resultados de simulación

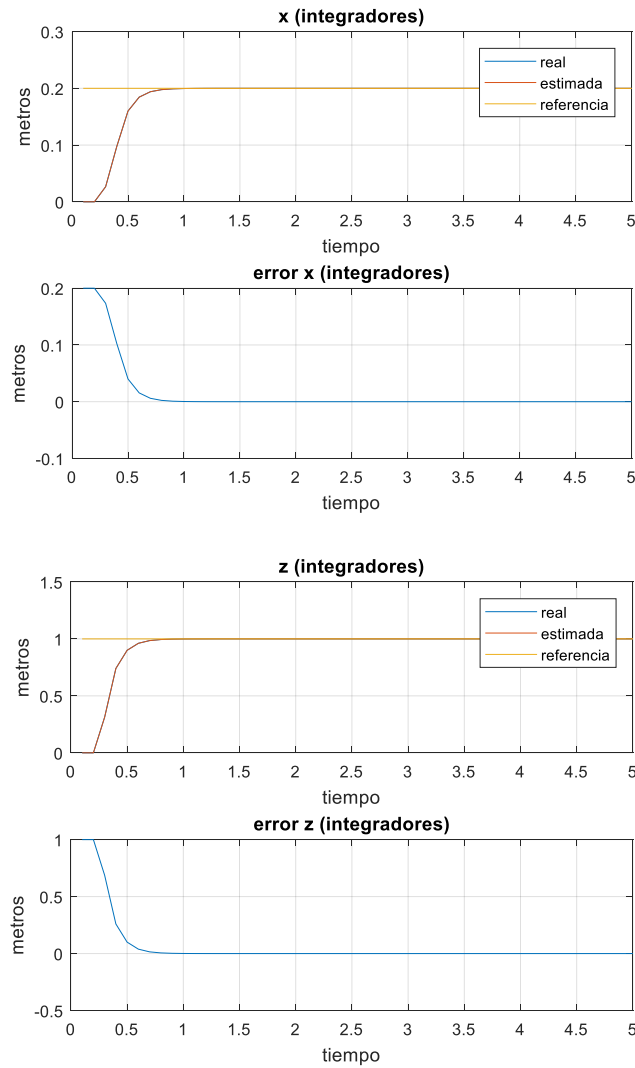
Siguiendo el procedimiento descrito en el apartado 5.2.2, y fijando un tiempo de respuesta de 0.5s, se han obtenido las leyes de control y observación que se muestran bajo estas líneas (ver Anexo II):

$$L_R = \begin{bmatrix} 1.4921 & -0.0849 & -0.0248 \\ 1.5033 & -0.0868 & -0.1429 \\ -0.0075 & 0.0620 & 0.0013 \end{bmatrix} \quad (40)$$

$$L_S = \begin{bmatrix} 3.7880 & 0.2698 & -0.1278 & -0.0021 & 0.0578 & 0.0137 \\ 3.8048 & 0.2701 & -0.1309 & -0.0022 & -0.3096 & -0.0179 \\ -0.0113 & -1.8924E-4 & 0.1926 & 0.0165 & 0.0021 & 3.9529E-5 \end{bmatrix} \quad (41)$$

$$L_o = \begin{bmatrix} 1.0 & 8.3396E-9 & 4.3628E-13 \\ 8.5039 & 0.0074 & 2.4167E-7 \\ 8.3400E-9 & 1.0 & 3.8647E-11 \\ 0.0074 & 9.9991 & 5.6345E-6 \\ 1.7712E-12 & 3.3853E-11 & 1.0 \\ 2.3632E-7 & 4.5221E-6 & 8.5037 \end{bmatrix} \quad (42)$$

Una vez que se han obtenido las leyes se han introducido junto al objetivo de navegación o posición deseada al bucle de simulación. En éste se ha simulado durante 5s la evolución de las variables de estado del sistema y del regulador y las salidas calculadas a partir del mismo. La posición objetivo marcada ha sido de 0.2m hacia delante, 1m hacia arriba y con un giro vertical de 90° (1.57rad).



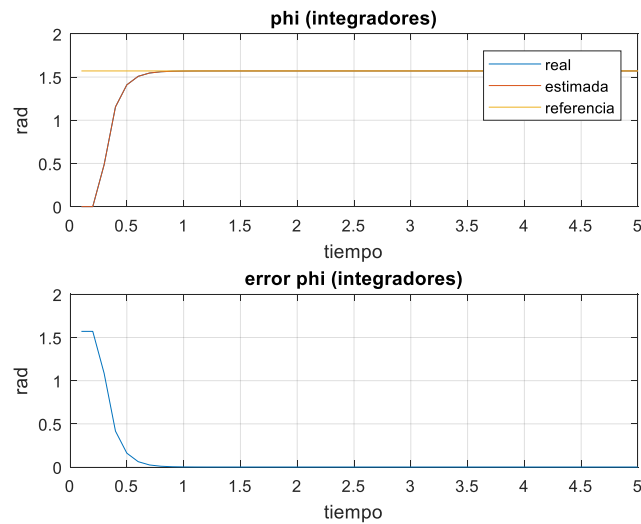


Figura 17. Evolución de las variables de estado x , z y ϕ y del error de estas respecto a la posición de referencia frente al tiempo de simulación en seg.

En la Figura 17 se muestra la evolución de las variables de estado x , z y ϕ reales y estimadas (que son idénticas al tratarse de una simulación) conforme avanza el tiempo de simulación. Se puede observar cómo todas ellas alcanzan la posición deseada con error de posición nulo en el tiempo de respuesta establecido.

Se puede observar en la Figura 18 como las tres acciones siguen la misma tendencia, y además se puede comprobar como todas se hacen nulas al llegar al estado deseado en el tiempo de respuesta.

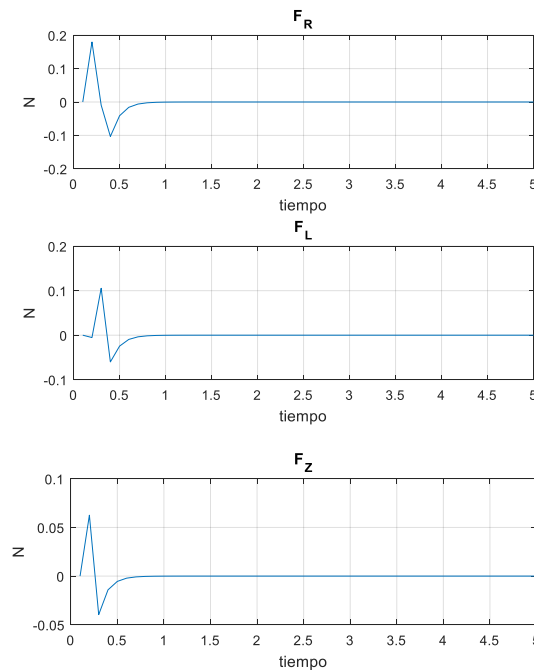


Figura 18. Evolución de las acciones de salida F_R , F_L y F_Z con respecto al tiempo de simulación

7. Conclusiones y desarrollo futuro

7.1. Conclusiones

A lo largo de este estudio se han abordado diferentes tareas cuyo propósito era alcanzar los objetivos marcados al inicio del mismo. Así pues, afrontando las distintas dificultades que han surgido y que se han indicado a lo largo de este escrito, se ha puesto en marcha el dirigible y se ha implementado un sistema de percepción, comunicación y control con el que conseguir un sistema de navegación autónoma del dirigible.

Resumidamente, las conclusiones que se desprenden de este trabajo, con los distintos desarrollos que ha implicado, son las siguientes:

- El sistema de navegación implementado en el dirigible basado sobre el modelo cinemático es capaz de orientarse y avanzar hacia el objetivo de forma aceptable, especialmente en el plano horizontal. Los efectos dinámicos no considerados y las características de los actuadores hacen que el movimiento en z sea mucho más impreciso e inestable, por lo que sería necesario realizar alguna mejora en este sentido.
- El sistema de navegación simulado basado en el modelo dinámico del dirigible ha demostrado ser correcto ya que proporciona los resultados teóricos esperados, aunque para validar su eficacia y ajustar los parámetros involucrados sería necesaria su implementación sobre el dirigible real y la posterior realización de pruebas.
- La metodología seguida para la consecución de los objetivos ha resultado ser acertada en muchos aspectos, lo que ha permitido rápidos avances. A pesar de esto también se ha llegado a algunos problemas de difícil solución que han consumido mucho tiempo y esfuerzo hasta que se han podido reparar, como ha sido el caso con la librería LIRC.
- Se ha comprobado cómo el dirigible es mucho más apto para la interacción con humanos que un dron quadrotor, ya que, entre otras cuestiones, durante todo el desarrollo de los trabajos no se ha producido ni un solo incidente relacionado con el mismo.

7.2. Desarrollo futuro

Con objeto de corregir algunas de las deficiencias encontradas en el desarrollo actual y de mejorar en general el funcionamiento del sistema de navegación, se proponen las siguientes acciones:

- Modificar el sistema de actuación sobre los motores del dirigible. Para conseguir esto habría que acceder al interior de la góndola, y realizar ahí las modificaciones. Esto permitiría tener un control de mayor calidad al poder aplicar de forma más directa las salidas calculadas por el sistema de control.
- Implementación del sistema de navegación sobre modelos de dirigible más avanzados. Al contar con un dirigible de más calidad las fuentes de error debidas al mismo se reducen, y además se puede contar con mayores capacidades tecnológicas que amplían las posibilidades del sistema de navegación. Ejemplos de mayores capacidades serían la posibilidad de elevar cargas mayores o un sistema de comunicación más fiable y con mayor alcance.
- Disponer de una cámara con mayor campo de visión. Esto se podría conseguir de diversas formas: con la cámara actual acoplándole una lente de “ojo de pez” (ver Figura 19), disponiendo de una segunda cámara adicional, dotando a la cámara de un mecanismo que le permita moverse y/o orientarse, etc. Con un campo de visión mayor los límites de la zona de control serían también mayores y se mejoraría la detección del marcador visual a la vez que se conseguiría un sistema de navegación más flexible y versátil.
- Ajuste fino de los parámetros del sistema de control. Caracterizando con mayor precisión los parámetros que intervienen en el control se conseguiría una mayor robustez ante perturbaciones y una consecución más precisa de los objetivos de navegación.

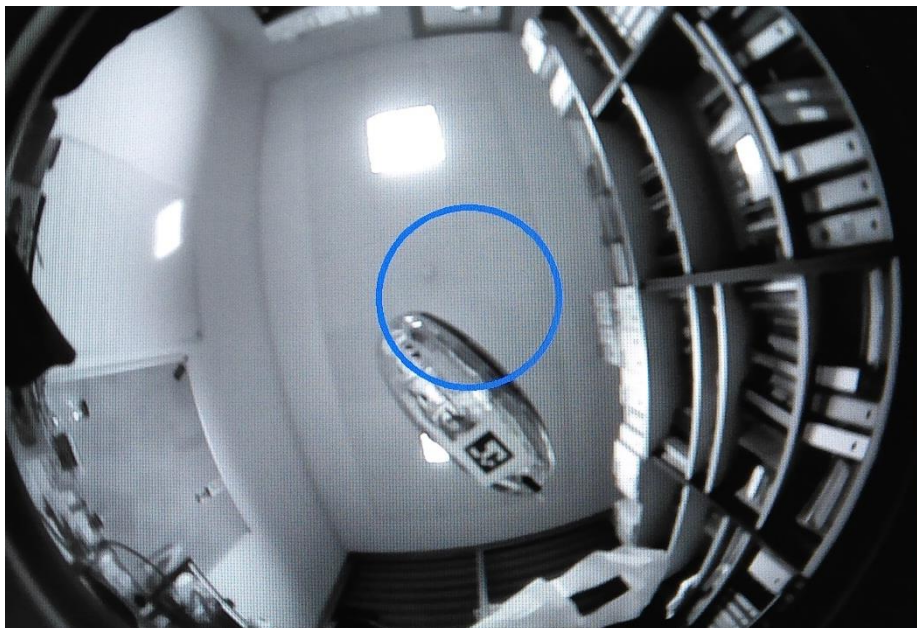


Figura 19. Vista del dirigible desde abajo con una lente de ojo de pez acoplada a la cámara

8. Bibliografía

- [1] C. F. Liew y T. Yairi, «Quadrotor or blimp? Noise and appearance considerations in designing social aerial robot», en *Human-Robot Interaction (HRI), 2013 8th ACM/IEEE International Conference on*, 2013, pp. 183–184.
- [2] W. Wagner, *Lightning Bugs and Other Reconnaissance Drones*. Armed Forces Journal, 1982.
- [3] D. S. Lawson, *Engineering Disasters: Lessons to be Learned*. ASME Press, 2005.
- [4] J. M. M. Neto, R. A. da Paixão, L. R. L. Rodrigues, E. M. Moreira, J. C. J. dos Santos, y P. F. F. Rosa, «A surveillance task for a UAV in a natural disaster scenario», en *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*, 2012, pp. 1516–1522.
- [5] P. U. Lima y R. Ventura, «Search and Rescue Robots: The Civil Protection Teams of the Future», 2012, pp. 12-19.
- [6] E. Ackerman, «Blimps Seem Like the Friendliest Kind of Indoor Flying Robots», *IEEE Spectrum: Technology, Engineering, and Science News*, 06-ago-2017. [En línea]. Disponible en: <https://spectrum.ieee.org/autaton/robotics/drones/blimps-seem-like-the-friendliest-kind-of-indoor-flying-robots>. [Accedido: 04-nov-2017].
- [7] «Georgia Tech Lab Develops Micro Blimps for Indoor Navigation and Sensing», *vairdo*, 29-dic-2015. .
- [8] «Project Loon». [En línea]. Disponible en: <https://x.company/loon/>. [Accedido: 11-nov-2017].
- [9] A. L. Majdik, D. Verda, Y. Albers-Schoenberg, y D. Scaramuzza, «Air-ground Matching: Appearance-based GPS-denied Urban Localization of Micro Aerial Vehicles: Air-ground Matching: Appearance-based GPS-denied Urban Localization of Micro Aerial Vehicles», *J. Field Robot.*, vol. 32, n.º 7, pp. 1015-1039, oct. 2015.
- [10] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, y D. Scaramuzza, «Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor Micro Aerial Vehicle: Autonomous, Vision-based Flight and Live Dense 3D Mapping», *J. Field Robot.*, vol. 33, n.º 4, pp. 431-450, jun. 2016.
- [11] C. Troiani, A. Martinelli, C. Laugier, y D. Scaramuzza, «Low computational-complexity algorithms for vision-aided inertial navigation of micro aerial vehicles», *Robot. Auton. Syst.*, vol. 69, pp. 80-97, jul. 2015.

- [12] L. M. Alkurdi y R. B. Fisher, «Visual Control of an Autonomous Indoor Robotic Blimp», *Robot. Vis. Technol. Mach. Learn. Vis. Appl.*, pp. 352–370, 2012.
- [13] Agencia Española de Seguridad Aérea, «Aeronaves civiles pilotadas por control remoto», *BOE*, n.º 252, p. 83976 a 83982, oct. 2014.
- [14] A. J. C. Moreira, R. T. Valadas, y A. M. de Oliveira Duarte, «Performance of infrared transmission systems under ambient light interference», *IEE Proc. - Optoelectron.*, vol. 143, n.º 6, pp. 339-346, dic. 1996.
- [15] «Raspberry Pi - Teach, Learn, and Make with Raspberry Pi», *Raspberry Pi*. [En línea]. Disponible en: <https://www.raspberrypi.org/>. [Accedido: 23-nov-2017].
- [16] «LIRC - Linux Infrared Remote Control». [En línea]. Disponible en: <http://www.lirc.org/>. [Accedido: 13-nov-2017].
- [17] «pigpio library». [En línea]. Disponible en: <http://abyz.me.uk/rpi/pigpio/index.html>. [Accedido: 23-nov-2017].
- [18] «OpenCV library». [En línea]. Disponible en: <https://opencv.org/>. [Accedido: 13-nov-2017].
- [19] G. R. Bradski y A. Kaehler, *Learning OpenCV: computer vision with the OpenCV library*, 1. ed., [Nachdr.]. Beijing: O'Reilly, 2011.
- [20] «ArUco: a minimal library for Augmented Reality applications based on OpenCV | Aplicaciones de la Visión Artificial». [En línea]. Disponible en: <https://www.uco.es/investiga/grupos/ava/node/26>. [Accedido: 14-nov-2017].
- [21] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, y M. J. Marín-Jiménez, «Automatic generation and detection of highly reliable fiducial markers under occlusion», *Pattern Recognit.*, vol. 47, n.º 6, pp. 2280-2292, jun. 2014.
- [22] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, y R. Medina-Carnicer, «Generation of fiducial marker dictionaries using Mixed Integer Linear Programming», *Pattern Recognit.*, vol. 51, pp. 481-491, mar. 2016.
- [23] G. López, «Apuntes de Visión y Robótica». Universidad de Zaragoza, 2017.
- [24] M. T. Rashid, A. A. Ali, R. S. Ali, L. Fortuna, M. Frasca, y M. G. Xibilia, «Wireless underwater mobile robot system based on ZigBee», 2012, pp. 117-122.
- [25] R. P. M. Chan, K. A. Stol, y C. R. Halkyard, «Review of modelling and control of two-wheeled robots», *Annu. Rev. Control*, vol. 37, n.º 1, pp. 89-103, abr. 2013.
- [26] F. M. White, *Fluid Mechanics*, 4. ed. McGraw-Hill.

Bibliografía

[27] «Pole placement design - MATLAB place - MathWorks España». [En línea]. Disponible en: <https://es.mathworks.com/help/control/ref/place.html>. [Accedido: 21-nov-2017].

Anexo I: Código en Python del sistema de navegación

En este anexo se recoge el código en lenguaje Python que gobierna el funcionamiento del sistema de navegación. Este código se ejecuta en la Raspberry Pi.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
=====

=   Blimp drone navigation system 1. Frontal camera mode   =
=====

Version: 2.1 (3.11)
Date: 19/11/2017
@author: rherguedas
Institution: Systems Engineering and Automation area,
             EINA (School of Engineering and Architecture of Zaragoza),
             University of Zaragoza

-----

Description:
This program implements a navigation system of a blimp drone
"ZEP-AIR Explorer II" in the Raspberry Pi 3-B platform.
Position is monitorized with a Raspberry Pi 5Mpx camera and the
open-source libraries "OpenCV" and "Aruco"
Orders are given in the form of IR signals by means of the open-source
linux "LIRC" library.

-----

-----

Disclaimer:
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

Anexo I: Código en Python del sistema de navegación

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```
-----  
"""  
  
# Necessary libraries  
from picamera.array import PiRGBArray  
from picamera import PiCamera  
import cv2  
import cv2.aruco as aruco  
import calibParameters as cp  
  
import numpy as np  
  
from multiprocessing import Process, Pipe  
import time  
import os  
  
import pigpio # http://abyz.co.uk/rpi/pigpio/python.html  
import blimpKeys  
  
#####  
  
def carrier(gpio, frequency, micros):  
    """  
    Generate carrier square wave.  
    """  
    wf = []  
    cycle = 1000.0 / frequency  
    cycles = int(round(micros/cycle))  
    on = int(round(cycle / 2.0))  
    sofar = 0  
    for c in range(cycles):  
        target = int(round((c+1)*cycle))
```

Anexo I: Código en Python del sistema de navegación

```
    sofars += on

    off = target - sofars

    sofars += off

    wf.append(pigpio.pulse(1<<gpio, 0, on))

    wf.append(pigpio.pulse(0, 1<<gpio, off))

    return wf

# Child process definition: IR routine
def secondary(conn):

    code = [] # key code

    pi.set_mode(GPIO, pigpio.OUTPUT) # IR TX connected to this GPIO
    pi.wave_add_new()

    emit_time = time.time()

    conn.send("SEND") # ask keys to the main loop
    keys = conn.recv() # receive keys from the main loop
    conn.send("SEND")

    print("Sending...")

    # Send continuously the IR commands
    while keys[0] != "END":
        for key in keys:
            #print("----- Sending " + str(key) + " -----")
            code = key

            # Create wave
            marks_wid = {}
            spaces_wid = {}

            wave = [0]*len(code)
```

Anexo I: Código en Python del sistema de navegación

```
        for i in range(0, len(code)):
            ci = code[i]
            if i & 1: # Space
                if ci not in spaces_wid:
                    pi.wave_add_generic([pigpio.pulse(0, 0, ci)])
                    spaces_wid[ci] = pi.wave_create()
                    wave[i] = spaces_wid[ci]
            else: # Mark
                if ci not in marks_wid:
                    wf = carrier(GPIO, FREQ, ci)
                    pi.wave_add_generic(wf)
                    marks_wid[ci] = pi.wave_create()
                    wave[i] = marks_wid[ci]

        delay = emit_time - time.time()

        if delay > 0.0:
            time.sleep(delay)

        pi.wave_chain(wave)

        while pi.wave_tx_busy():
            time.sleep(0.002)

        emit_time = time.time() + GAP

        # wave reset, to repeat with new parameters
        for i in marks_wid:
            pi.wave_delete(marks_wid[i])

        marks_wid = {}

        for i in spaces_wid:
            pi.wave_delete(spaces_wid[i])
```


Anexo I: Código en Python del sistema de navegación

```
spaces_wid = {}

time.sleep(0.1) # pulse ammount limiter

if conn.poll() == True:
    keys = conn.recv()
    conn.send("SEND")

print("Ending...")

pi.stop() # disconnect from Pi
conn.close() # pipe connection closed

#####
# Reference position and marker size
refPos = [0., -410., 0.] # mm
markerSize = 100. # marker size in mm
#refOri = 0.0

# Blimp parameters
R = 35. # mm. Half of the distance between right and left propellers
verticalRange = 5 # adjust depending on the blimp lifting force. Maximum = 6

# IR parameters
GPIO = 22 # IR emitter pin
FREQ = 38.0 # kHz
GAP = 0.1 # s

# Keys assignment
KEY_POWER = blimpKeys.KEY_POWER
KEY_STOP = blimpKeys.KEY_STOP
KEY_FRONT = blimpKeys.KEY_FRONT
KEY_RIGHT = blimpKeys.KEY_RIGHT
KEY_LEFT = blimpKeys.KEY_LEFT
KEY_0 = blimpKeys.KEY_0
```

Anexo I: Código en Python del sistema de navegación

```
KEY_1 = blimpKeys.KEY_1
KEY_2 = blimpKeys.KEY_2
KEY_3 = blimpKeys.KEY_3
KEY_4 = blimpKeys.KEY_4
KEY_5 = blimpKeys.KEY_5

wordZ = "" # empty z word

# Control constants
Kv = 1.
Kw = 2.3
Kz = 0.75

vLmax = Kv * 1500. # max left propeller velocity for max remaining distance =
2000 mm
vRmax = 2 * Kv * 2000. # max right propeller velocity for max remaining distance
= 2000 mm
vZmax = Kz * 800. # max vertical propeller velocity

tolZero = vLmax/100. # zero tolerance

# Miscellaneous parameters
deg2rad = np.pi/180.
count = 1
tol = 0.017 # equality tolerance
maxRotDiff = 25 * deg2rad # degrees to consider z inversion issue
signal = 0
rep = 8

#####

# Create results file
f = open("results.txt", mode = 'w')
f.write("Time\tRemaining distance\tAngle error\tAltitude error\tvL\tvR\tvZ\n")

# Pipe connection settings
```

Anexo I: Código en Python del sistema de navegación

```
parent_conn, child_conn = Pipe()
p = Process(target=secondary, args=(child_conn,))

# Camera initialization
resolutionX = 640 #1024
resolutionY = 480 #768

camera = PiCamera()
camera.resolution = (resolutionX, resolutionY)
camera.framerate = 32
camera.color_effects = (128,128)

rawCapture = PiRGBArray(camera, size=(resolutionX, resolutionY))

# Camera warm up
time.sleep(0.1)

# Get Aruco dictionary of markers
aruco_dict = aruco.getPredefinedDictionary(cv2.aruco.DICT_5X5_50)

# Pipe connection start
os.system("sudo pigpiod") # start pigpio daemon
pi = pigpio.pi() # connect to Pi
p.start()

request = parent_conn.recv()
if request == "SEND":
    parent_conn.send([KEY_POWER, KEY_POWER])
    time.sleep(6)

# Capture frames
for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):

    parameters = aruco.DetectorParameters_create()
```

Anexo I: Código en Python del sistema de navegación

```
image = frame.array # raw NumPy array representing the image

corners, ids, rejectedImgPoints = aruco.detectMarkers(image, aruco_dict,
parameters=parameters)

# Start the navigation system whenever a marker is detected
if corners != []:
    rvecs, tvecs, objPoints = aruco.estimatePoseSingleMarkers(corners,
markerSize, cp.cameraMatrix, cp.distCoeffs)

    # rvecs, tvecs --> rotation and translation of each detected marker
    # with respect to the camera center (origin of coordinates)

    trans = tvecs[0,0]

    # Rotations processing
    m, jacob = cv2.Rodrigues(rvecs)
    Euler, mtxR, mtxQ, Qx, Qy, Qz = cv2.RQDecomp3x3(m)

    #image = aruco.drawDetectedMarkers(image, corners)
    image = aruco.drawAxis(image, cp.cameraMatrix, cp.distCoeffs, rvecs,
tvecs, 100.)

    #averX = int(round((corners[0][0,0,0] + corners[0][0,1,0] +
corners[0][0,2,0] + corners[0][0,3,0])/4))

    #averY = int(round((corners[0][0,0,1] + corners[0][0,1,1] +
corners[0][0,2,1] + corners[0][0,3,1])/4))

    #cv2.line(image, (resolutionX/2, resolutionY/2), (averX,averY),
(59,131,189), 3) # draw line to the objective

#####
# Control algorithm
#####

# Equations
remDist = trans[2] # remaining Z distance
angErr = - np.arctan(trans[0]/trans[2]) - Euler[1] * deg2rad # angular
error

if signal == 1: # filter to avoid z inversion effect
```

Anexo I: Código en Python del sistema de navegación

```
        if (angErr > (angErr_ant + maxRotDiff)) or (angErr < (angErr_ant -
maxRotDiff)):
            angErr = angErr_ant # inversion detected

        else:
            angErr_ant = angErr # no inversion detected

    else:
        angErr_ant = angErr
        signal = 1

    posY = trans[1] # Y position

    vL = Kv * remDist - (Kw * R * angErr)/2 # left propeller velocity
    vR = 2 * Kv * remDist - vL # right propeller velocity
    vZ = Kz * (posY - refPos[1]) # vertical propeller velocity
    #vZ = Kz * (refPos[1] - posY) # vertical propeller velocity

    # Outputs to the IR interface
    #-----
    # Vertical
    keyWordZ = KEY_0
    for i in range(0, verticalRange):
        if (vZ > vZmax/verticalRange*(verticalRange-i)):
            wordZ = "KEY_" + str(verticalRange-i)
            break

    if wordZ != "":
        if wordZ == "KEY_5":
            keyWordZ = KEY_5
        elif wordZ == "KEY_4":
            keyWordZ = KEY_4
        elif wordZ == "KEY_3":
            keyWordZ = KEY_3
        elif wordZ == "KEY_2":
```

```
        keyWordZ = KEY_2

    elif wordZ == "KEY_1":

        keyWordZ = KEY_1

#-----

# Horizontal

if (vR <= (vL * (1 + tol))) and (vR >= (vL * (1 - tol))) or\
    (vL <= (vR * (1 + tol))) and (vL >= (vR * (1 - tol))):

    keyWordXY = KEY_FRONT

    printXY = "KEY_FRONT" # debugging purpose

elif vR > (vL * (1 + tol)):

    keyWordXY = KEY_LEFT

    printXY = "KEY_LEFT"

elif vL > (vR * (1 + tol)):

    keyWordXY = KEY_RIGHT

    printXY = "KEY_RIGHT"

elif (vR <= tolZero) and (vL <= tolZero):

    keyWordXY = KEY_STOP

    printXY = "KEY_STOP"

# Send keys to the IR interface if it is prepared

if parent_conn.poll() == True:

    request = parent_conn.recv()

    if request == "SEND":

        parent_conn.send([keyWordXY, keyWordZ])

# Write results

f.write(str(time.clock()) + "\t" + str(remDist) + "\t" + str(angErr) +
"\t" + str(posY - refPos[1]) \
        + "\t" + str(vL) + "\t" + str(vR) + "\t" + str(vZ) + "\n")

# Debugging fragment

if count == rep:

    print("Remaining distance: ", remDist)

    print("XY key: " + printXY)
```

Anexo I: Código en Python del sistema de navegación

```
        print("Z key: " + wordZ)
        print("Euler Y: ", Euler[1])
        print("Angular error: ", angErr/deg2rad)
        #print("Angle: ", np.arctan(trans[0]/trans[2])/deg2rad)
        print("Altitude error: ", posY - refPos[1])
        print("vL: ", vL)
        print("vR: ", vR)
        print("vZ: ", vZ)
        print("-----")
        count = 0
        count += 1

        wordZ = ""

    else: # if the marker is not detected, the blimp should not fly

        if parent_conn.poll() == True:
            request = parent_conn.recv()
            if request == "SEND":
                parent_conn.send([KEY_STOP, KEY_0])

        signal = 0

    # Show the frame in the screen
    #cv2.circle(image, (resolutionX/2, resolutionY/2), 100, (255,255,000), 3)
    # approximate objective zone
    cv2.imshow("Frame", image)

    key = cv2.waitKey(1) & 0xFF

    # Clear the stream for the next frame
    rawCapture.truncate(0)

    # 'q' or 'Q' key = quit and finish the program
```

Anexo I: Código en Python del sistema de navegación

```
    if (key == ord("q")) or (key == ord("Q")):
        break

# Release the capture and switch blimp motors off
parent_conn.send(["END", ])

f.close() # close results file
cv2.destroyAllWindows()
# End
```


Anexo II: Código de Matlab

Este anexo contiene el código introducido en Matlab para calcular el modelo dinámico del dirigible y el sistema de control con integradores y estimador de estado, así como para simularlo a continuación.

```
%% Modelo dinámico dirigible en Espacio de Estados
%=====

% Definición de constantes
m = 31.0e-3; %kg, masa de la góndola
mz = 1.0e-3; %kg, masa en vertical, compensada
area = pi*0.185*0.2; %m^2, área frontal, elipse
R = 35.0e-3; %m
d = 65.0e-3; %m
rho = 1.204; %kg/m^3
dx_0 = 0.001; %m/s
Cd = 0.15;
I3 = 10.0e-3 * d^2 + 2 * 10.0e-3 * R^2; %kg*m^2

a = -(1/m * rho * dx_0 * Cd * area);

% Definición de matrices
A = [0.0 1.0 0.0 0.0 0.0 0.0;
     0.0 a 0.0 0.0 0.0 0.0;
     0.0 0.0 0.0 1.0 0.0 0.0;
     0.0 0.0 0.0 0.0 0.0 0.0;
     0.0 0.0 0.0 0.0 0.0 1.0;
     0.0 0.0 0.0 0.0 0.0 0.0];

B = [0.0 0.0 0.0;
     1/m 1/m 0.0;
     0.0 0.0 0.0;
     0.0 0.0 1/mz;
     0.0 0.0 0.0;
     R/I3 -R/I3 0.0];

C = [1.0 0.0 0.0 0.0 0.0 0.0;
     0.0 0.0 1.0 0.0 0.0 0.0;
     0.0 0.0 0.0 0.0 1.0 0.0];

D = 0.0;

%% Discretización
%=====

T_muestreo = 1e-1; %s

[F_z,G_z]=c2d(A,B,T_muestreo);

%% Controlabilidad y Observabilidad
%=====

% Controlabilidad
chi_cont = [B A*B (A^2)*B (A^3)*B (A^4)*B (A^5)*B];
rango_chi = rank(chi_cont);
```

Anexo II: Código de Matlab

```
% rango_chi = n = 6 --> Controlable

% Observabilidad
theta_obs = [C; C*A; C*(A^2); C*(A^3); C*(A^4); C*(A^5)];
rango_theta = rank(theta_obs);
% rango_theta = n = 6 --> Observable

%% Sistema de control con integradores
%=====

tr = 0.5; %s

sigma = 4.75/tr;
polos_int = [ exp(-T_muestreo*sigma) exp(-T_muestreo*sigma+0.00001)...
              exp(-T_muestreo*5*sigma) exp(-T_muestreo*(5*sigma-0.001))...
              exp(-T_muestreo*6*sigma) exp(-T_muestreo*(6*sigma-0.001))...
              exp(-T_muestreo*7*sigma) exp(-T_muestreo*(7*sigma-0.001))...
              exp(-T_muestreo*8*sigma)];

F_r = eye(3);
G_r = eye(3);

Cx = [1.0 0.0 0.0 0.0 0.0 0.0;
      0.0 0.0 1.0 0.0 0.0 0.0;
      0.0 0.0 0.0 0.0 1.0 0.0]; % consigna en x, z y phi

F_t = [F_z      zeros(6,3);
      -G_r*Cx      F_r];
G_t = [G_z;
      zeros(3,3)];

Li = place(F_t,G_t,polos_int);
Ls = Li(:,1:6);
Lr = -Li(:,7:9);

%% Observador de estado: Estimador
%=====

tr_o = tr/2; %s %dinámica del estimador dos veces más rápida que la del sistema

sigma_o = 4.75/tr_o;
polos_est = [ exp(-T_muestreo*sigma_o) exp(-T_muestreo*sigma_o+0.00001)...
              exp(-T_muestreo*5*sigma_o) exp(-T_muestreo*(5*sigma_o-0.001))...
              exp(-T_muestreo*6*sigma_o) exp(-T_muestreo*(6*sigma_o-0.001))];

Lo = (place(F_z',F_z'*C',polos_est))';

%% Simulador espacio de estados: Integradores
%=====

tfinal = 5.0;

% En base al periodo de muestreo, deberemos simular n pasos:
N = tfinal/T_muestreo;

t = T_muestreo:T_muestreo:tfinal;

% Establecemos la consigna como valor constante para todo k
```

Anexo II: Código de Matlab

```
%0.2m hacia delante, 1m hacia arriba y giro de 90°
r1 = 0.2 *ones(1,N);
r2 = 1.0 *ones(1,N);
r3 = pi/2 *ones(1,N);
r = [r1; r2; r3];

% Inicializamos vector de variables de estado y de acción para guardar los
% resultados
x_est = zeros(6,N);
x_pre = zeros(6,N+1);
x = zeros(6,N+1);
xR = zeros(3,N+1);
y = zeros(3,N);
u = zeros(3,N);

% Simulamos el sistema:
for k=1:N,
    % Determinamos el valor de la salida en el instante actual
    y(:,k) = C * x(:,k);

    % Obtenemos la estimación del estado
    x_est(:,k) = x_pre(:,k) + Lo * (y(:,k) - C * x_pre(:,k));

    % Actualizamos el estado del regulador
    xR(:,k+1) = F_r * xR(:,k) + G_r * (r(:,k) - Cx * x(:,k));

    % Calculamos la acción en el instante actual
    u(:,k) = Lr * xR(:,k) - Ls * x_est(:,k);

    % Actualizamos el estado
    x(:,k+1) = F_z * x(:,k) + G_z * u(:,k);

    % Obtenemos la predicción del estado para el instante siguiente
    x_pre(:,k+1) = F_z * x_est(:,k) + G_z * u(:,k);

end;

% Representamos gráficamente las variables de estado y las salidas:
figure;
subplot(2,1,1);
plot(t,x(1,1:N),t,x_est(1,1:N),t,r(1,1:N)); title('x (integradores)');
xlabel('tiempo'); ylabel('metros'); grid; legend({'real', 'estimada',
'referencia'});
subplot(2,1,2);
plot(t,r(1,1:N) - x(1,1:N)); title('error x (integradores)'); xlabel('tiempo');
ylabel('metros'); grid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
subplot(2,1,1);
plot(t,x(3,1:N),t,x_est(3,1:N),t,r(2,1:N)); title('z (integradores)');
xlabel('tiempo'); ylabel('metros'); grid; legend({'real', 'estimada',
'referencia'});
subplot(2,1,2);
plot(t,r(2,1:N) - x(3,1:N)); title('error z (integradores)'); xlabel('tiempo');
ylabel('metros'); grid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
subplot(2,1,1);
```

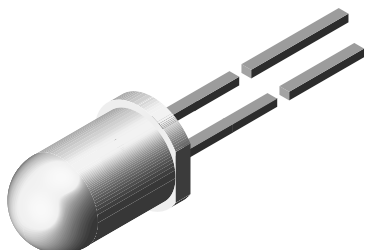
Anexo II: Código de Matlab

```
plot(t,x(5,1:N),t,x_est(5,1:N),t,r(3,1:N)); title('phi (integradores)');
xlabel('tiempo'); ylabel('rad'); grid; legend({'real', 'estimada',
'referencia'});
subplot(2,1,2);
plot(t,r(3,1:N) - x(5,1:N)); title('error phi (integradores)');
xlabel('tiempo'); ylabel('rad'); grid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
subplot(2,1,1);
plot(t,u(1,1:N)); title('F_R'); xlabel('tiempo'); ylabel('N'); grid;
subplot(2,1,2);
plot(t,u(2,1:N)); title('F_L'); xlabel('tiempo'); ylabel('N'); grid;
figure;
subplot(2,1,1);
plot(t,u(3,1:N)); title('F_Z'); xlabel('tiempo'); ylabel('N'); grid;
```

Anexo III: Hojas de características de los componentes

En este anexo se incluyen las hojas de características de los componentes utilizados en el circuito de recepción y emisión IR.

High Speed Infrared Emitting Diode, 850 nm, GaAlAs Double Hetero



94 8389

DESCRIPTION

TSHG6400 is an infrared, 850 nm emitting diode in GaAlAs double hetero (DH) technology with high radiant power and high speed, molded in a clear, untinted plastic package.

FEATURES

- Package type: leaded
- Package form: T-1 $\frac{3}{4}$
- Dimensions (in mm): \varnothing 5
- Peak wavelength: $\lambda_p = 850$ nm
- High reliability
- High radiant power
- High radiant intensity
- Angle of half intensity: $\phi = \pm 22^\circ$
- Low forward voltage
- Suitable for high pulse current operation
- High modulation bandwidth: $f_c = 18$ MHz
- Good spectral matching with CMOS cameras
- Compliant to RoHS Directive 2002/95/EC and in accordance to WEEE 2002/96/EC



RoHS
COMPLIANT
GREEN
(5-2008)**

Note

** Please see document "Vishay Material Category Policy":
www.vishay.com/doc?99902

APPLICATIONS

- Infrared radiation source for operation with CMOS cameras (illumination).
- High speed IR data transmission.

PRODUCT SUMMARY

COMPONENT	I_e (mW/sr)	ϕ (deg)	λ_p (nm)	t_r (ns)
TSHG6400	70	± 22	850	20

Note

- Test conditions see table "Basic Characteristics"

ORDERING INFORMATION

ORDERING CODE	PACKAGING	REMARKS	PACKAGE FORM
TSHG6400	Bulk	MOQ: 4000 pcs, 4000 pcs/bulk	T-1 $\frac{3}{4}$

Note

- MOQ: minimum order quantity

ABSOLUTE MAXIMUM RATINGS ($T_{amb} = 25^\circ\text{C}$, unless otherwise specified)

PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
Reverse voltage		V_R	5	V
Forward current		I_F	100	mA
Peak forward current	$t_p/T = 0.5$, $t_p = 100 \mu\text{s}$	I_{FM}	200	mA
Surge forward current	$t_p = 100 \mu\text{s}$	I_{FSM}	1	A
Power dissipation		P_V	180	mW
Junction temperature		T_j	100	$^\circ\text{C}$
Operating temperature range		T_{amb}	- 40 to + 85	$^\circ\text{C}$
Storage temperature range		T_{stg}	- 40 to + 100	$^\circ\text{C}$
Soldering temperature	$t \leq 5$ s, 2 mm from case	T_{sd}	260	$^\circ\text{C}$
Thermal resistance junction/ambient	J-STD-051, leads 7 mm soldered on PCB	R_{thJA}	230	K/W

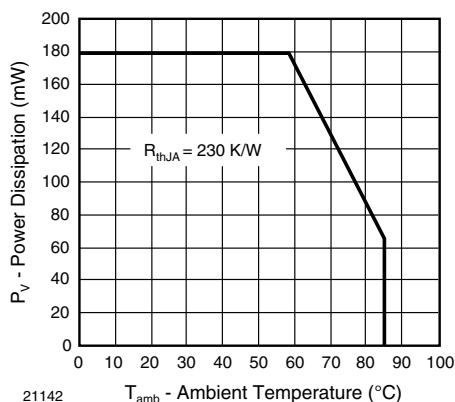


Fig. 1 - Power Dissipation Limit vs. Ambient Temperature

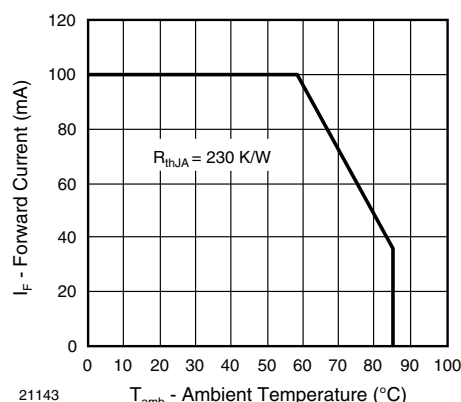


Fig. 2 - Forward Current Limit vs. Ambient Temperature

BASIC CHARACTERISTICS ($T_{amb} = 25\text{ }^{\circ}\text{C}$, unless otherwise specified)						
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
Forward voltage	$I_F = 100\text{ mA}$, $t_p = 20\text{ ms}$	V_F		1.5	1.8	V
	$I_F = 1\text{ A}$, $t_p = 100\text{ }\mu\text{s}$	V_F		2.3		V
Temperature coefficient of V_F	$I_F = 1\text{ mA}$	TK_{VF}		- 1.8		mV/K
Reverse current	$V_R = 5\text{ V}$	I_R			10	μA
Junction capacitance	$V_R = 0\text{ V}$, $f = 1\text{ MHz}$, $E = 0$	C_j		125		pF
Radiant intensity	$I_F = 100\text{ mA}$, $t_p = 20\text{ ms}$	I_e	45	70	135	mW/sr
	$I_F = 1\text{ A}$, $t_p = 100\text{ }\mu\text{s}$	I_e		700		mW/sr
Radiant power	$I_F = 100\text{ mA}$, $t_p = 20\text{ ms}$	ϕ_e		50		mW
Temperature coefficient of ϕ_e	$I_F = 100\text{ mA}$	TK_{ϕ_e}		- 0.35		%/K
Angle of half intensity		ϕ		± 22		deg
Peak wavelength	$I_F = 100\text{ mA}$	λ_p		850		nm
Spectral bandwidth	$I_F = 100\text{ mA}$	$\Delta\lambda$		40		nm
Temperature coefficient of λ_p	$I_F = 100\text{ mA}$	TK_{λ_p}		0.25		nm/K
Rise time	$I_F = 100\text{ mA}$	t_r		20		ns
Fall time	$I_F = 100\text{ mA}$	t_f		13		ns
Cut-off frequency	$I_{DC} = 70\text{ mA}$, $I_{AC} = 30\text{ mA pp}$	f_c		18		MHz
Virtual source diameter		d		3.7		mm

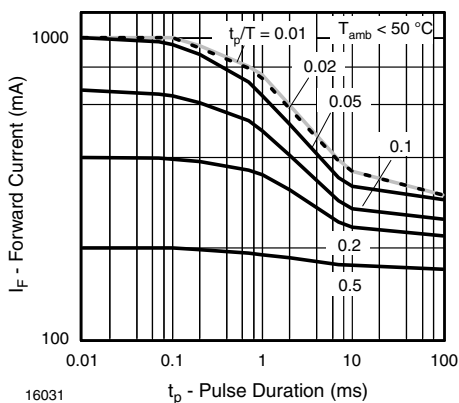
BASIC CHARACTERISTICS ($T_{amb} = 25\text{ }^{\circ}\text{C}$, unless otherwise specified)


Fig. 3 - Pulse Forward Current vs. Pulse Duration

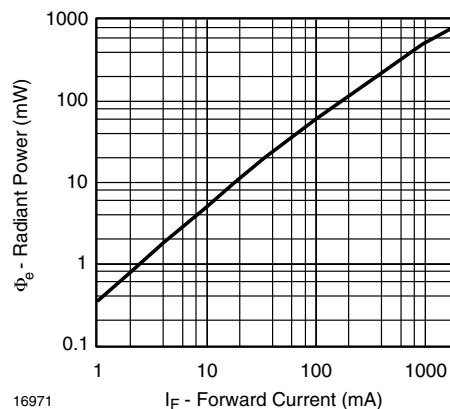


Fig. 6 - Radiant Power vs. Forward Current

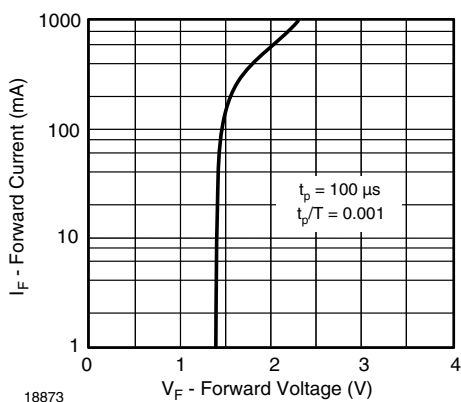


Fig. 4 - Forward Current vs. Forward Voltage

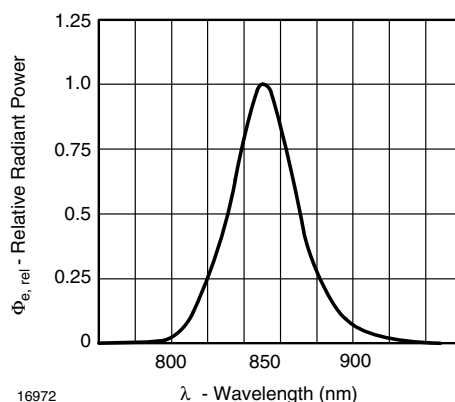


Fig. 7 - Relative Radiant Power vs. Wavelength

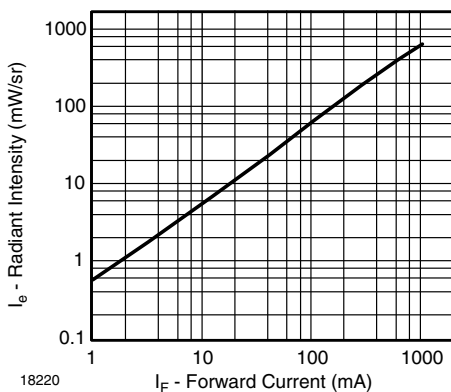


Fig. 5 - Radiant Intensity vs. Forward Current

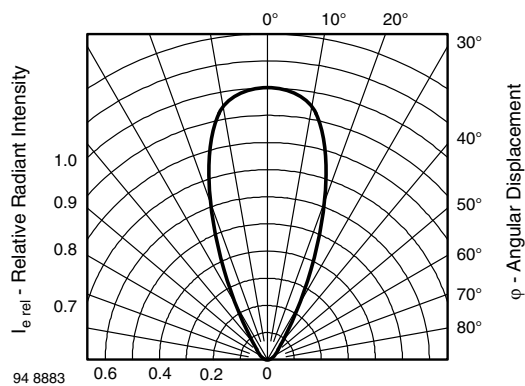
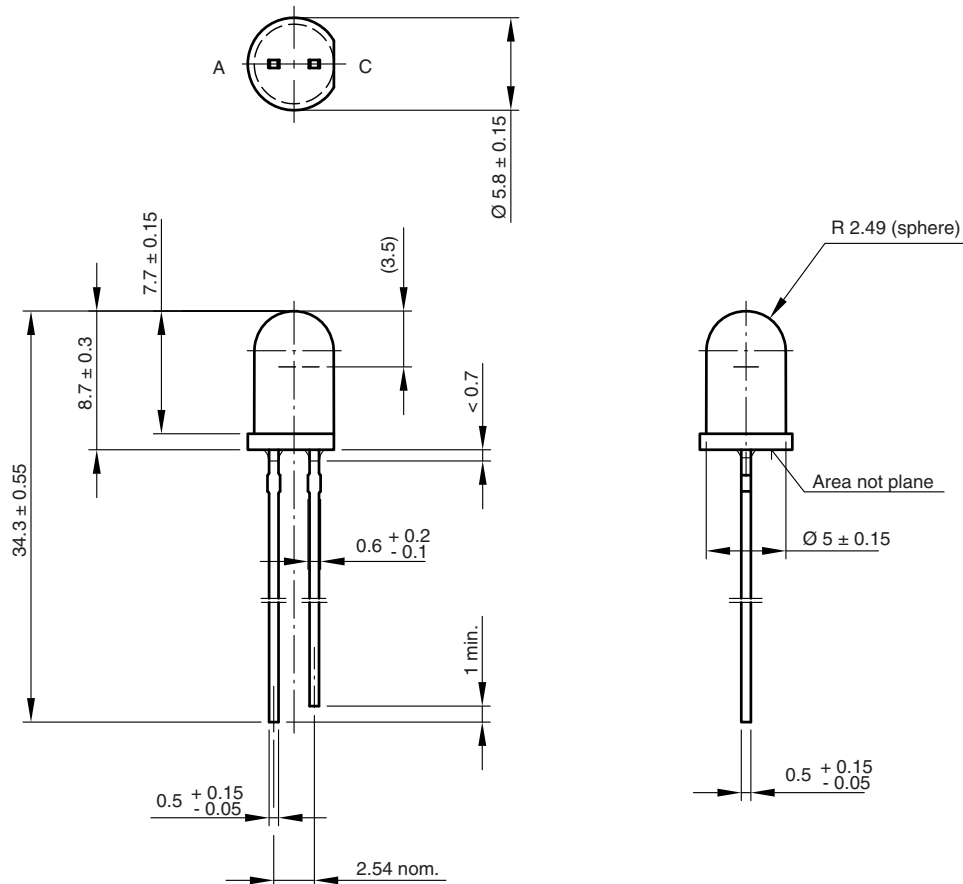


Fig. 8 - Relative Radiant Intensity vs. Angular Displacement

PACKAGE DIMENSIONS in millimeters


Drawing-No.: 6.544-5259.06-4

Issue: 6; 19.05.09

19257



Disclaimer

ALL PRODUCT, PRODUCT SPECIFICATIONS AND DATA ARE SUBJECT TO CHANGE WITHOUT NOTICE TO IMPROVE RELIABILITY, FUNCTION OR DESIGN OR OTHERWISE.

Vishay Intertechnology, Inc., its affiliates, agents, and employees, and all persons acting on its or their behalf (collectively, "Vishay"), disclaim any and all liability for any errors, inaccuracies or incompleteness contained in any datasheet or in any other disclosure relating to any product.

Vishay makes no warranty, representation or guarantee regarding the suitability of the products for any particular purpose or the continuing production of any product. To the maximum extent permitted by applicable law, Vishay disclaims (i) any and all liability arising out of the application or use of any product, (ii) any and all liability, including without limitation special, consequential or incidental damages, and (iii) any and all implied warranties, including warranties of fitness for particular purpose, non-infringement and merchantability.

Statements regarding the suitability of products for certain types of applications are based on Vishay's knowledge of typical requirements that are often placed on Vishay products in generic applications. Such statements are not binding statements about the suitability of products for a particular application. It is the customer's responsibility to validate that a particular product with the properties described in the product specification is suitable for use in a particular application. Parameters provided in datasheets and/or specifications may vary in different applications and performance may vary over time. All operating parameters, including typical parameters, must be validated for each customer application by the customer's technical experts. Product specifications do not expand or otherwise modify Vishay's terms and conditions of purchase, including but not limited to the warranty expressed therein.

Except as expressly indicated in writing, Vishay products are not designed for use in medical, life-saving, or life-sustaining applications or for any other application in which the failure of the Vishay product could result in personal injury or death. Customers using or selling Vishay products not expressly indicated for use in such applications do so at their own risk. Please contact authorized Vishay personnel to obtain written terms and conditions regarding products designed for such applications.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document or by any conduct of Vishay. Product names and markings noted herein may be trademarks of their respective owners.

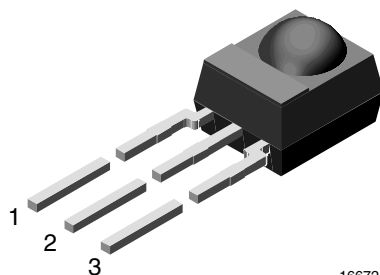
Material Category Policy

Vishay Intertechnology, Inc. hereby certifies that all its products that are identified as RoHS-Compliant fulfill the definitions and restrictions defined under Directive 2011/65/EU of The European Parliament and of the Council of June 8, 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment (EEE) - recast, unless otherwise specified as non-compliant.

Please note that some Vishay documentation may still make reference to RoHS Directive 2002/95/EC. We confirm that all the products identified as being compliant to Directive 2002/95/EC conform to Directive 2011/65/EU.

Vishay Intertechnology, Inc. hereby certifies that all its products that are identified as Halogen-Free follow Halogen-Free requirements as per JEDEC JS709A standards. Please note that some Vishay documentation may still make reference to the IEC 61249-2-21 definition. We confirm that all the products identified as being compliant to IEC 61249-2-21 conform to JEDEC JS709A standards.

IR Sensor Module for Reflective Sensor, Light Barrier, and Fast Proximity Applications



16672

MECHANICAL DATA

Pinning:

1 = OUT, 2 = GND, 3 = V_S

APPLICATIONS

- Reflective sensors for hand dryers, towel or soap dispensers, water faucets, toilet flush
- Vending machine fall detection
- Security and pet gates
- Person or object vicinity activation
- Fast proximity sensors for toys, robotics, drones, and other consumer and industrial uses

FEATURES

- Up to 2 m for presence and proximity sensing
- Uses modulated bursts of infrared light
- 940 nm peak wavelength
- PIN diode and sensor IC in one package
- Low supply current
- Shielding against EMI
- Visible light is suppressed by IR filter
- Insensitive to supply voltage ripple and noise
- Supply voltage: 2.5 V to 5.5 V
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912



RoHS
COMPLIANT
HALOGEN
FREE
GREEN
(5-2008)

DESCRIPTION

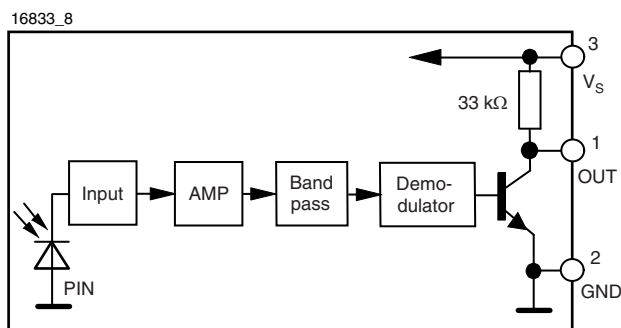
The TSSP40.. series are compact infrared detector modules for presence and fast proximity sensing applications. They provide an active low output in response to infrared bursts at 940 nm. The frequency of the burst should correspond to the carrier frequency shown in the parts table.

This component has not been qualified according to automotive specifications.

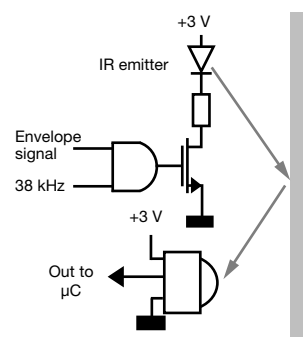
PARTS TABLE

Carrier frequency	38 kHz	TSSP4038
	56 kHz	TSSP4056
Package	Mold	
Pinning	1 = OUT, 2 = GND, 3 = V_S	
Dimensions (mm)	6.0 W x 6.95 H x 5.6 D	
Mounting	Leaded	
Application	Presence sensors, fast proximity sensors	

BLOCK DIAGRAM



PRESENCE SENSING



ABSOLUTE MAXIMUM RATINGS				
PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
Supply voltage (pin 3)		V_S	-0.3 to +6.0	V
Supply current (pin 3)		I_S	5	mA
Output voltage (pin 1)		V_O	-0.3 to 5.5	V
Voltage at output to supply		$V_S - V_O$	-0.3 to ($V_S + 0.3$)	V
Output current (pin 1)		I_O	5	mA
Junction temperature		T_j	100	°C
Storage temperature range		T_{stg}	-25 to +85	°C
Operating temperature range		T_{amb}	-25 to +85	°C
Power consumption	$T_{amb} \leq 85\text{ °C}$	P_{tot}	10	mW

Note

- Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect the device reliability.

ELECTRICAL AND OPTICAL CHARACTERISTICS ($T_{amb} = 25\text{ °C}$, unless otherwise specified)						
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
Supply current (pin 3)	$E_v = 0, V_S = 5\text{ V}$	I_{SD}	0.55	0.7	0.9	mA
	$E_v = 40\text{ klx, sunlight}$	I_{SH}	-	0.8	-	mA
Supply voltage		V_S	2.5	-	5.5	V
Transmission distance	$E_v = 0$, test signal see fig. 1, IR diode TSAL6200, $I_F = 200\text{ mA}$	d	-	25	-	m
Output voltage low (pin 1)	$I_{OSL} = 0.5\text{ mA}$, $E_e = 2\text{ mW/m}^2$, test signal see fig. 1	V_{OSL}	-	-	100	mV
Minimum irradiance	Pulse width tolerance: $t_{pi} - 5/f_0 < t_{po} < t_{pi} + 6/f_0$, test signal see fig. 1	$E_{e\text{ min.}}$	-	0.4	0.7	mW/m ²
Maximum irradiance	$t_{pi} - 5/f_0 < t_{po} < t_{pi} + 6/f_0$, test signal see fig. 1	$E_{e\text{ max.}}$	50	-	-	W/m ²
Directivity	Angle of half transmission distance	$\phi_{1/2}$	-	± 45	-	deg

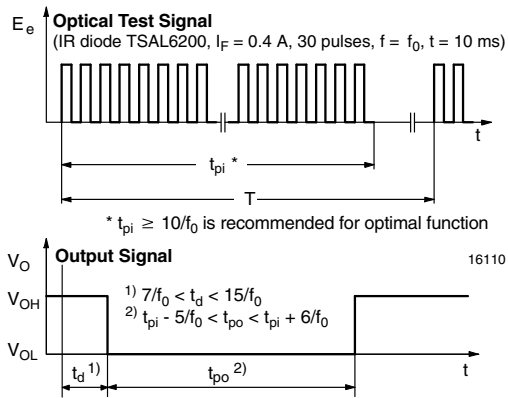
TYPICAL CHARACTERISTICS ($T_{amb} = 25\text{ }^{\circ}\text{C}$, unless otherwise specified)


Fig. 1 - Output Active Low

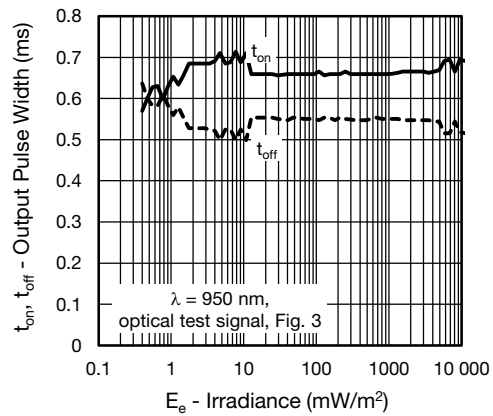


Fig. 4 - Output Pulse Diagram

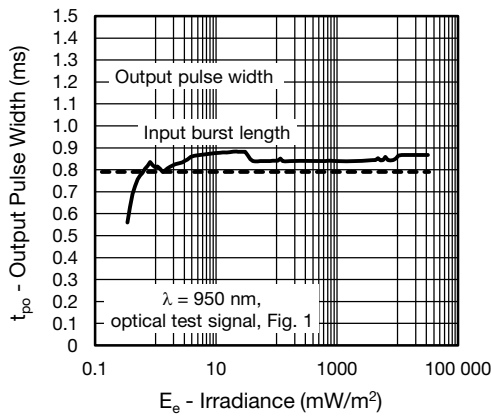


Fig. 2 - Pulse Length and Sensitivity in Dark Ambient

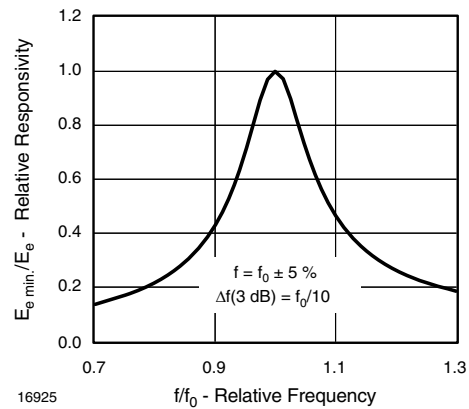


Fig. 5 - Frequency Dependence of Responsivity

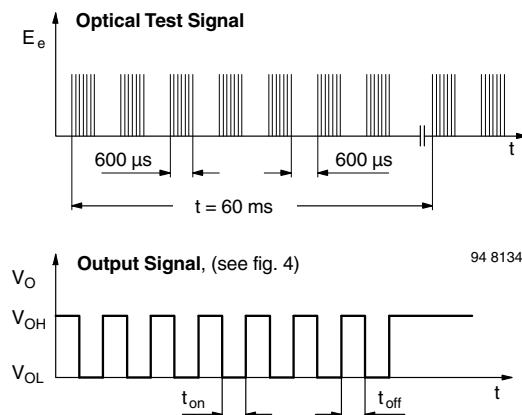


Fig. 3 - Output Function

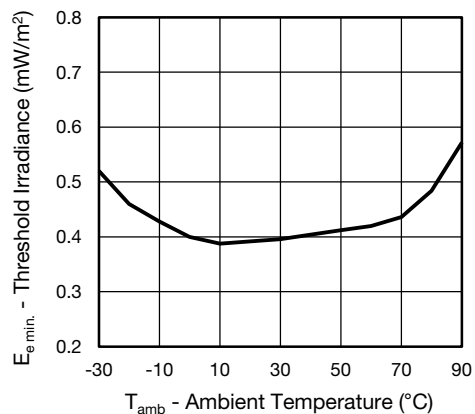


Fig. 6 - Sensitivity vs. Ambient Temperature

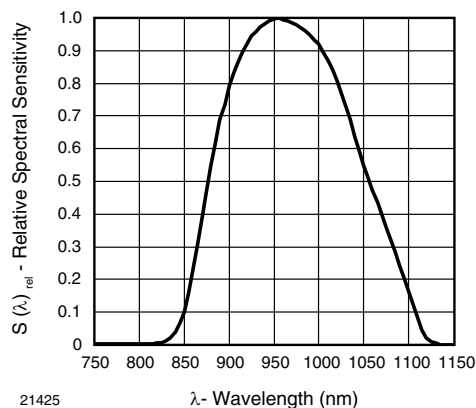


Fig. 7 - Relative Spectral Sensitivity vs. Wavelength

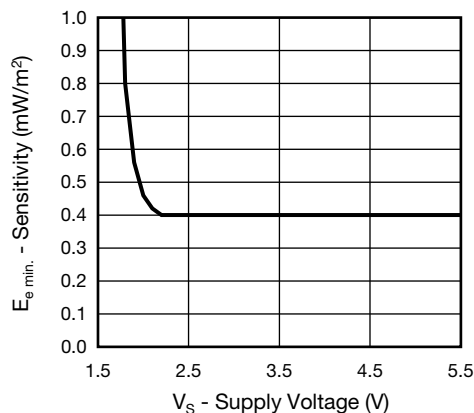


Fig. 9 - Sensitivity vs. Supply Voltage

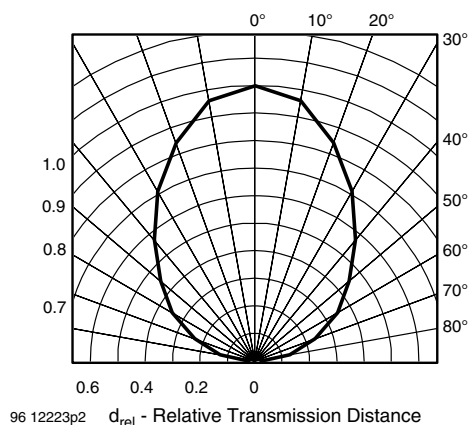
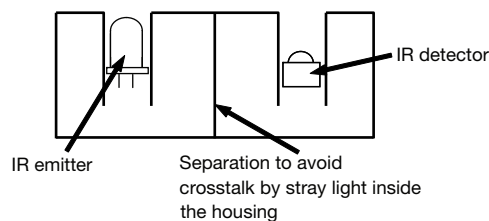


Fig. 8 - Directivity

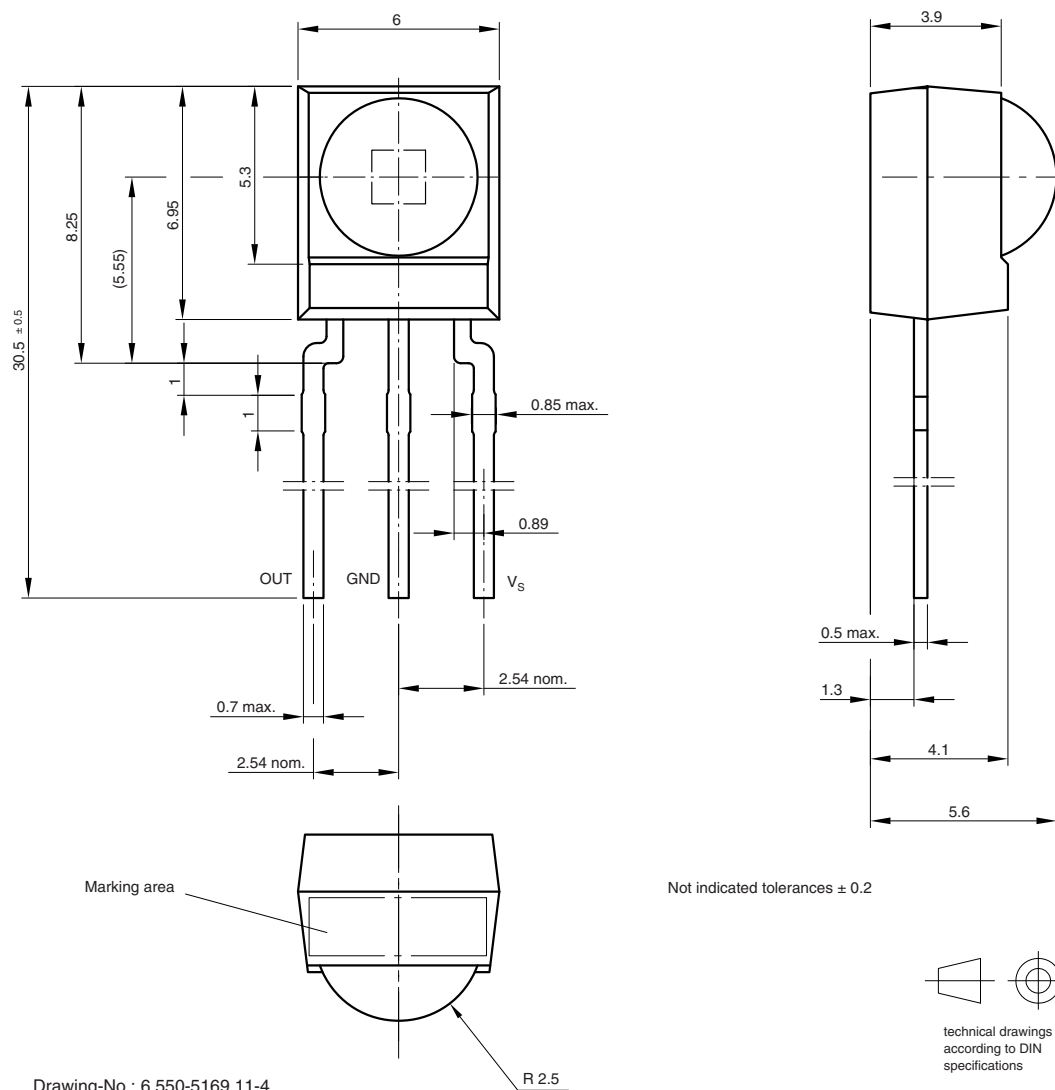
The typical application of these devices is a reflective or beam break sensor with active low “detect” or “no detect” information contained in its output. The TSSP4056 is also suitable for fast (~ 5 ms) proximity sensor applications for ranges between 10 cm and 2 m. Please see application note “Vishay’s TSSP4056 Sensor for Fast Proximity Sensing” (www.vishay.com/doc?82741).

Example for a sensor hardware:



There should be no common window in front of the emitter and detector in order to avoid crosstalk via guided light through the window.

PACKAGE DIMENSIONS in millimeters

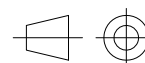


Drawing-No.: 6.550-5169.11-4

Issue: 13; 17.12.08

16003

Not indicated tolerances ± 0.2



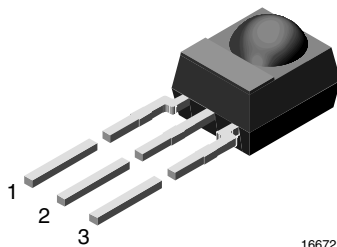
technical drawings
according to DIN
specifications



IR Receiver Modules for Remote Control Systems

Vishay offers stock molded IR receivers in four different packages:

- Loose packed in tubes, mounted on tape for reel or ammopack, or packed bulk in plastic bags.
- Vishay IR receiver with metal holders are packed in plastic trays. Vishay IR receiver with plastic holders are packed in plastic tubes.



FEATURES

- Material categorization:
For definitions of compliance please see www.vishay.com/doc?99912



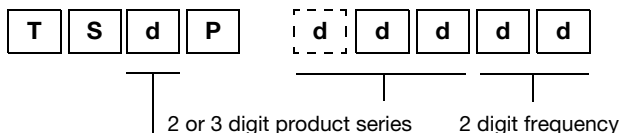
RoHS
COMPLIANT
GREEN
(5-2008)

AVAILABLE FOR

- TSOP348..
- TSOP344..
- TSOP343..
- TSOP341..
- TSOP44...
- TSOP48...
- TSOP41...
- TSOP324..
- TSOP323..
- TSOP322..
- TSOP321..
- TSOP24...
- TSOP22...
- TSOP21...
- TSOP345..
- TSOP325..
- TSOP43...
- TSOP23...
- TSSP4..
- TSMP4..

LOOSE PACKED IN TUBE

ORDERING INFORMATION



O = for IR receiver applications
M = for repeater/learning applications
S = for sensor applications

Note

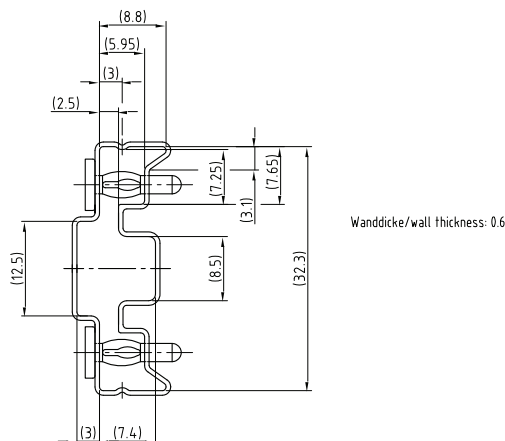
- d = "digit", please consult the list of available devices create a valid part number.

Example: TSOP4838

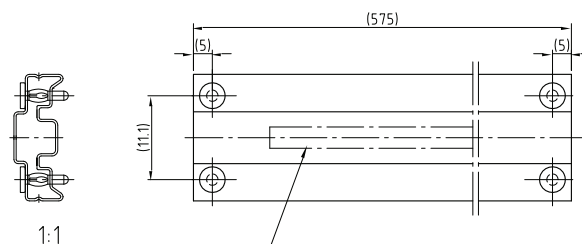
PACKAGING QUANTITY

- 90 pieces per tube
- 24 tubes per carton

PACKAGING DIMENSIONS in millimeters



Wanddicke/wall thickness: 0.6



1:1

Drawing-No.: 9.700-5185.0-4
Rev. 13; Date: 20.11.03
20273-1

Druck / Printing for tubes
1.400-5548.0-3 Version 1

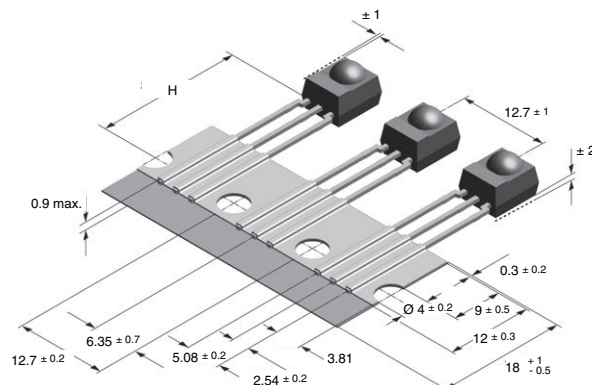
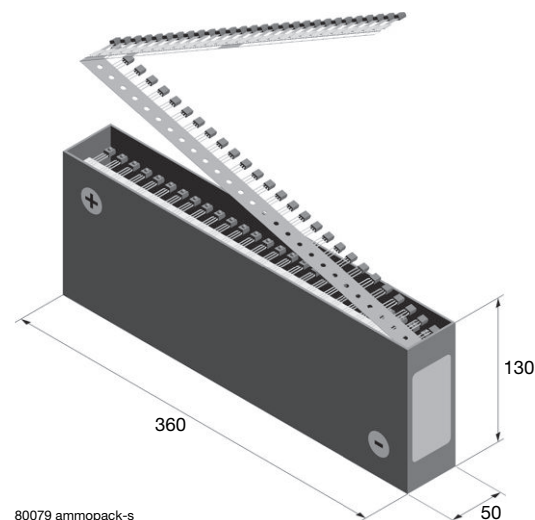
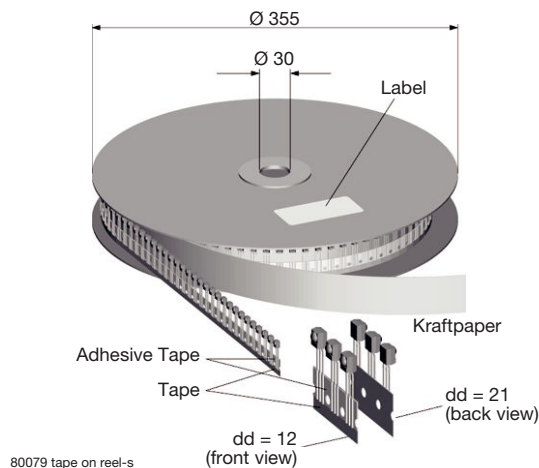


TAPE AND REEL/AMMOPACK

Up to 3 consecutive components may be missing if the gap is followed by at least 6 components. A maximum of 0.5 % of the components per reel quantity may be missing. At least 5 empty positions are present at the start and the end of the tape to enable insertion.

Tensile strength of the tape: > 15 N

Pulling force in the plane of the tape, at right angles to the reel: > 5 N



VERSION	DIMENSION "H"
BS	20 ± 0.5
PS	23.3 ± 0.5
OS	26 ± 0.5

ORDERING INFORMATION

T S d P

O = for IR receiver applications
M = for repeater/learning applications
S = for sensor applications

d d d d d

2 or 3 digit product series
2 digit frequency

S S 1

SS1 for T and R, bulk or ammpack

d d d d

dd = BS, PS or OS
Tape and reel
dd = 12 or 21

Z

Ammpack

Note

- d = "digit", please consult the list of available devices create a valid part number.

Example: TSOP4838SS1BS12

TSOP2238SS1BS12Z

PACKAGING QUANTITY

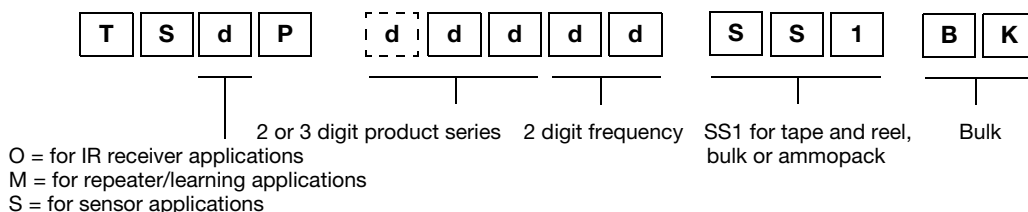
- 1000 pieces per reel
- 1000 pieces per ammpack



BULK PACKAGING

The option “BK” signifies bulk packaging in conductive plastic bags. A maximum of 0.3 % of the components per box may be missing.

ORDERING INFORMATION



Note

- d = “digit”, please consult the list of available devices create a valid part number.

EXAMPLE: TSOP4838SS1BK

TSOP2238SS1BK

PACKAGING QUANTITY

- 250 pieces per bag (each bag is individually boxed)
- 6 bags per carton

OUTER PACKAGING

CARTON BOX DIMENSIONS in millimeters			
KINDS OF CARTON BOX	THICKNESS	WIDTH	LENGTH
Packaging Plastic Tubes (Normal/auxiliary devices)	80	150	600
Packaging Plastic Trays (Devices with metal holders)	120	290	490
Tape and Reel Box (Taping in reels)	400	310	410
Ammo-Box (Zigzag taping)	50	130	350



Disclaimer

ALL PRODUCT, PRODUCT SPECIFICATIONS AND DATA ARE SUBJECT TO CHANGE WITHOUT NOTICE TO IMPROVE RELIABILITY, FUNCTION OR DESIGN OR OTHERWISE.

Vishay Intertechnology, Inc., its affiliates, agents, and employees, and all persons acting on its or their behalf (collectively, "Vishay"), disclaim any and all liability for any errors, inaccuracies or incompleteness contained in any datasheet or in any other disclosure relating to any product.

Vishay makes no warranty, representation or guarantee regarding the suitability of the products for any particular purpose or the continuing production of any product. To the maximum extent permitted by applicable law, Vishay disclaims (i) any and all liability arising out of the application or use of any product, (ii) any and all liability, including without limitation special, consequential or incidental damages, and (iii) any and all implied warranties, including warranties of fitness for particular purpose, non-infringement and merchantability.

Statements regarding the suitability of products for certain types of applications are based on Vishay's knowledge of typical requirements that are often placed on Vishay products in generic applications. Such statements are not binding statements about the suitability of products for a particular application. It is the customer's responsibility to validate that a particular product with the properties described in the product specification is suitable for use in a particular application. Parameters provided in datasheets and / or specifications may vary in different applications and performance may vary over time. All operating parameters, including typical parameters, must be validated for each customer application by the customer's technical experts. Product specifications do not expand or otherwise modify Vishay's terms and conditions of purchase, including but not limited to the warranty expressed therein.

Except as expressly indicated in writing, Vishay products are not designed for use in medical, life-saving, or life-sustaining applications or for any other application in which the failure of the Vishay product could result in personal injury or death. Customers using or selling Vishay products not expressly indicated for use in such applications do so at their own risk. Please contact authorized Vishay personnel to obtain written terms and conditions regarding products designed for such applications.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document or by any conduct of Vishay. Product names and markings noted herein may be trademarks of their respective owners.

N-Channel Power MOSFET 60V, 70A, 14 mΩ

These are N-Channel power MOSFETs manufactured using the MegaFET process. This process, which uses feature sizes approaching those of LSI circuits, gives optimum utilization of silicon, resulting in outstanding performance. They were designed for use in applications such as switching regulators, switching converters, motor drivers and relay drivers. These transistors can be operated directly from integrated circuits.

Formerly developmental type TA78440.

Ordering Information

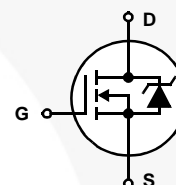
PART NUMBER	PACKAGE	BRAND
RFP70N06	TO-220AB	RFP70N06

NOTE: When ordering use the entire part number. Add the suffix 9A to obtain the TO-263AB variant in tape and reel, e.g. RF1S70N06SM9A.

Features

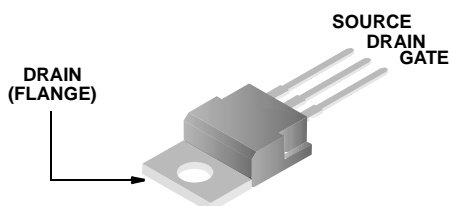
- 70A, 60V
- $r_{DS(on)} = 0.014\Omega$
- Temperature Compensated PSPICE® Model
- Peak Current vs Pulse Width Curve
- UIS Rating Curve (Single Pulse)
- 175°C Operating Temperature
- Related Literature
 - TB334 "Guidelines for Soldering Surface Mount Components to PC Boards"

Symbol



Packaging

JEDEC TO-220AB



RFP70N06

Absolute Maximum Ratings $T_C = 25^{\circ}\text{C}$, Unless Otherwise Specified

	RFP70N06	UNITS
Drain to Source Voltage (Note 1)	60	V
Drain to Gate Voltage ($R_{GS} = 20k\Omega$) (Note 1)	60	V
Continuous Drain Current	70	A
Pulsed Drain Current (Note 3)	Refer to Peak Current Curve	
Gate to Source Voltage	± 20	V
Single Pulse Avalanche Rating	Refer to UIS Curve	A
Power Dissipation	150	W
Linear Derating Factor	1.0	W/ $^{\circ}\text{C}$
Operating and Storage Temperature	-55 to 175	$^{\circ}\text{C}$
Maximum Temperature for Soldering		
Leads at 0.063in (1.6mm) from Case for 10s.	300	$^{\circ}\text{C}$
Package Body for 10s, See Techbrief 334	260	$^{\circ}\text{C}$

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

1. $T_J = 25^{\circ}\text{C}$ to 150°C .

Electrical Specifications $T_C = 25^{\circ}\text{C}$, Unless Otherwise Specified

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Drain to Source Breakdown Voltage	BV_{DSS}	$I_D = 250\mu\text{A}$, $V_{GS} = 0\text{V}$ (Figure 11)	60	-	-	V
Gate Threshold Voltage	$V_{GS(TH)}$	$V_{GS} = V_{DS}$, $I_D = 250\mu\text{A}$ (Figure 10)	2	-	4	V
Zero Gate Voltage Drain Current	I_{DSS}	$V_{DS} = 60\text{V}$, $V_{GS} = 0\text{V}$	-	-	1	μA
		$V_{DS} = 0.8 \times \text{Rated } BV_{DSS}$, $T_C = 150^{\circ}\text{C}$	-	-	25	μA
Gate to Source Leakage Current	I_{GSS}	$V_{GS} = \pm 20\text{V}$	-	-	± 100	nA
Drain to Source On Resistance (Note 2)	$r_{DS(ON)}$	$I_D = 70\text{A}$, $V_{GS} = 10\text{V}$ (Figure 9)	-	-	0.014	Ω
Turn-On Time	$t_{(ON)}$	$V_{DD} = 30\text{V}$, $I_D \approx 70\text{A}$, $R_L = 0.43\Omega$, $V_{GS} = 10\text{V}$, $R_{GS} = 2.5\Omega$ (Figure 13)	-	-	190	ns
Turn-On Delay Time	$t_{d(ON)}$		-	10	-	ns
Rise Time	t_r		-	137	-	ns
Turn-Off Delay Time	$t_{d(OFF)}$		-	32	-	ns
Fall Time	t_f		-	24	-	ns
Turn-Off Time	$t_{(OFF)}$		-	-	73	ns
Total Gate Charge	$Q_g(TOT)$	$V_{GS} = 0\text{V}$ to 20V	-	120	156	nC
Gate Charge at 10V	$Q_g(10)$	$V_{GS} = 0\text{V}$ to 10V		65	85	nC
Threshold Gate Charge	$Q_g(TH)$	$V_{GS} = 0\text{V}$ to 2V		5.0	6.5	nC
Input Capacitance	C_{ISS}	$V_{DS} = 25\text{V}$, $V_{GS} = 0\text{V}$, $f = 1\text{MHz}$ (Figure 12)	-	2250	-	pF
Output Capacitance	C_{OSS}		-	792	-	pF
Reverse Transfer Capacitance	C_{RSS}		-	206	-	pF
Thermal Resistance, Junction to Case	$R_{\theta JC}$		-	-	1.0	$^{\circ}\text{C/W}$
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	TO-220	-	-	62	$^{\circ}\text{C/W}$
		-	-	-	-	-

Source to Drain Diode Specifications

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Source to Drain Diode Voltage	V_{SD}	$I_{SD} = 70\text{A}$		-	1.5	V
Reverse Recovery Time	t_{rr}	$I_{SD} = 70\text{A}$, $dI_{SD}/dt = 100\text{A}/\mu\text{s}$		-	52	ns

NOTES:

2. Pulse test: pulse width $\leq 300\text{ms}$, duty cycle $\leq 2\%$.
3. Repetitive rating: pulse width is limited by maximum junction temperature. See Transient Thermal Impedance curve (Figure 3) and Peak Current Capability Curve (Figure 5).

Typical Performance Curves $T_C = 25^\circ\text{C}$, Unless Otherwise Specified

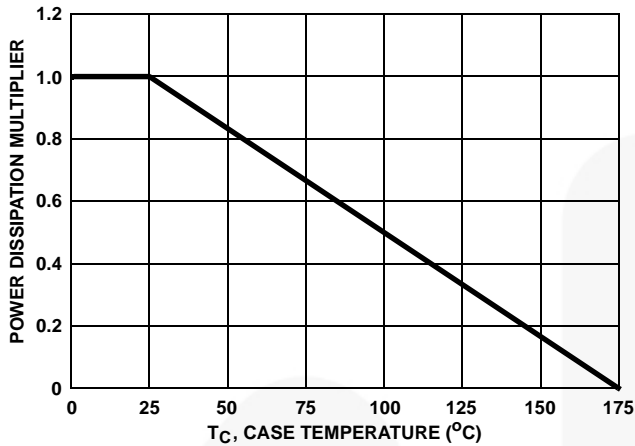


FIGURE 1. NORMALIZED POWER DISSIPATION vs CASE TEMPERATURE

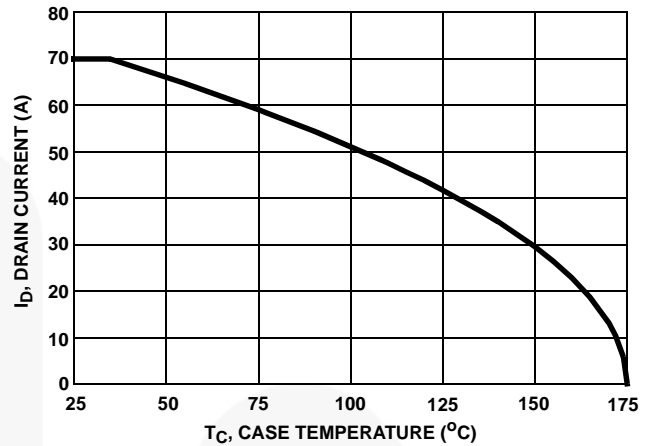


FIGURE 2. MAXIMUM CONTINUOUS DRAIN CURRENT vs CASE TEMPERATURE

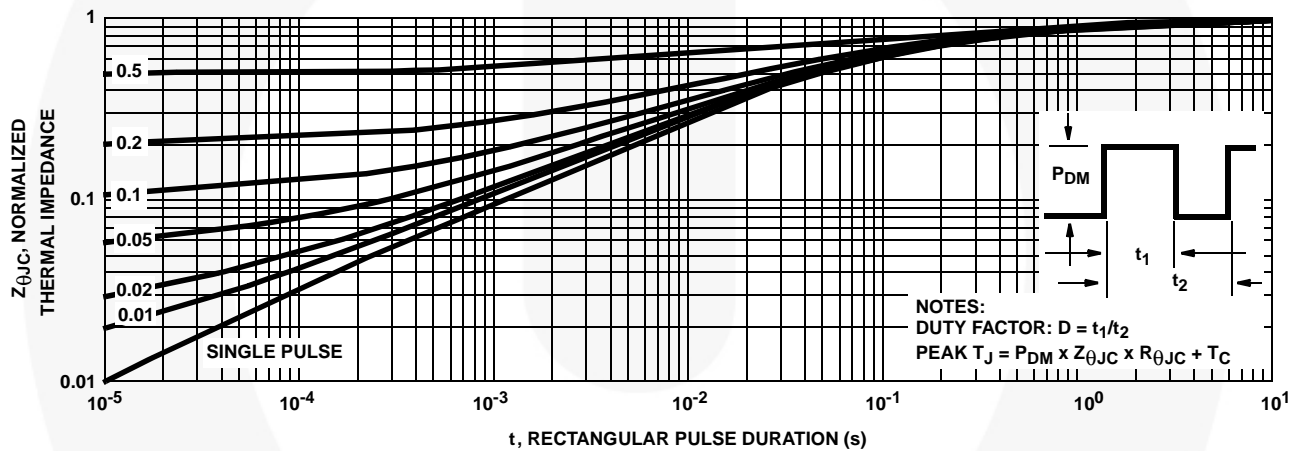


FIGURE 3. NORMALIZED MAXIMUM TRANSIENT THERMAL IMPEDANCE

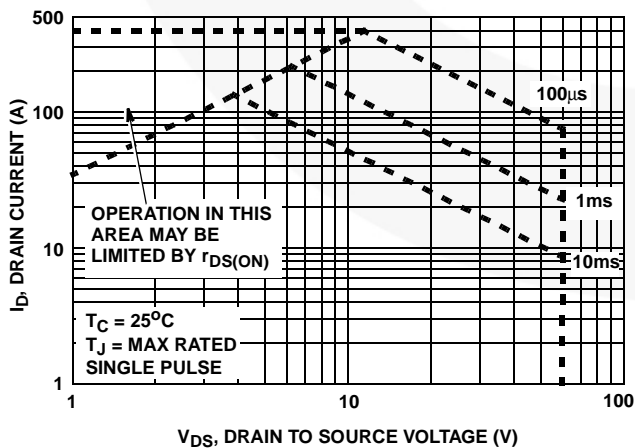


FIGURE 4. FORWARD BIAS SAFE OPERATING AREA

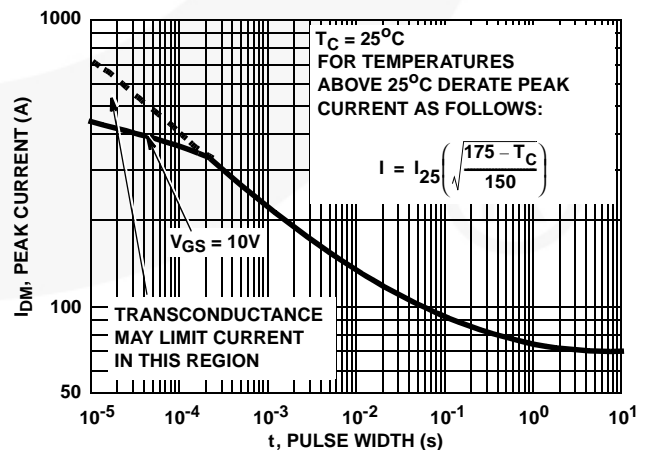
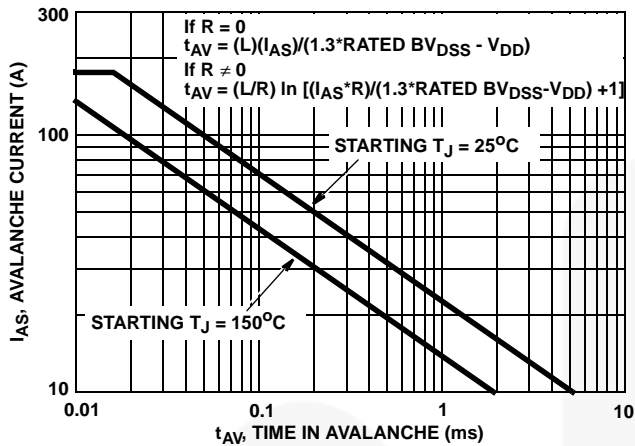


FIGURE 5. PEAK CURRENT CAPABILITY

Typical Performance Curves $T_C = 25^\circ\text{C}$, Unless Otherwise Specified (Continued)



NOTE: Refer to Fairchild Application Notes AN9321 and AN9322.

FIGURE 6. UNCLAMPED INDUCTIVE SWITCHING CAPABILITY

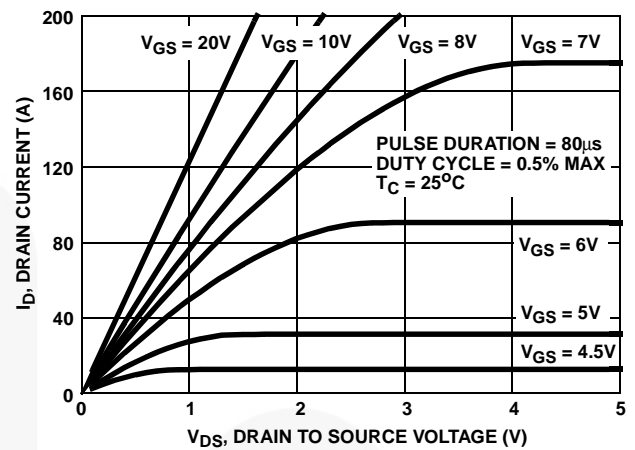


FIGURE 7. SATURATION CHARACTERISTICS

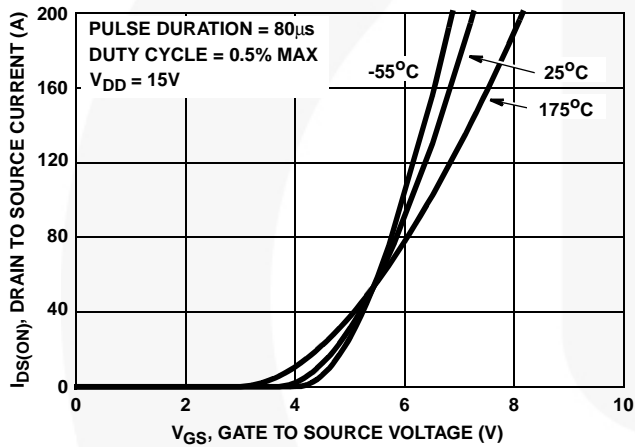


FIGURE 8. TRANSFER CHARACTERISTICS

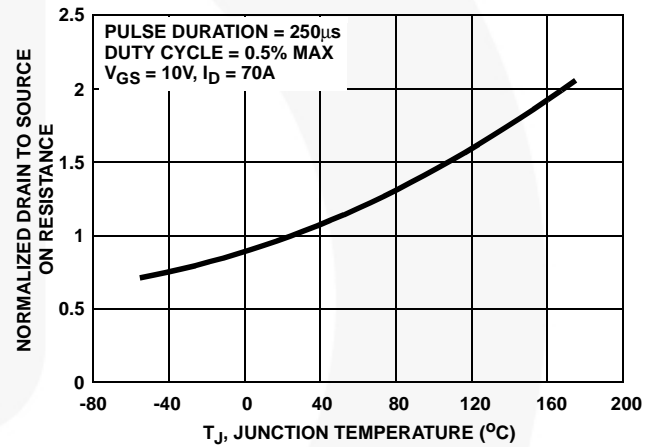


FIGURE 9. NORMALIZED DRAIN TO SOURCE ON RESISTANCE vs JUNCTION TEMPERATURE

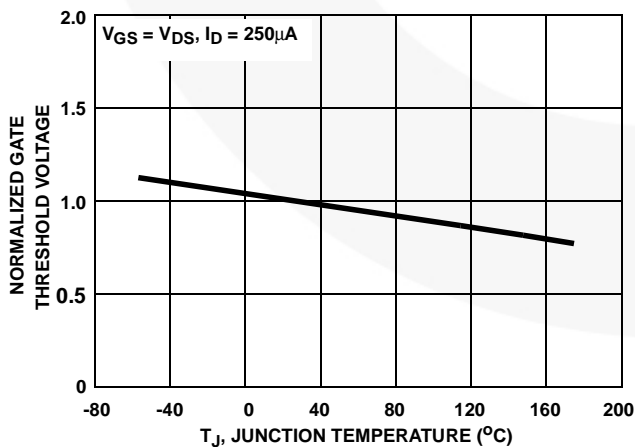


FIGURE 10. NORMALIZED GATE THRESHOLD VOLTAGE vs JUNCTION TEMPERATURE

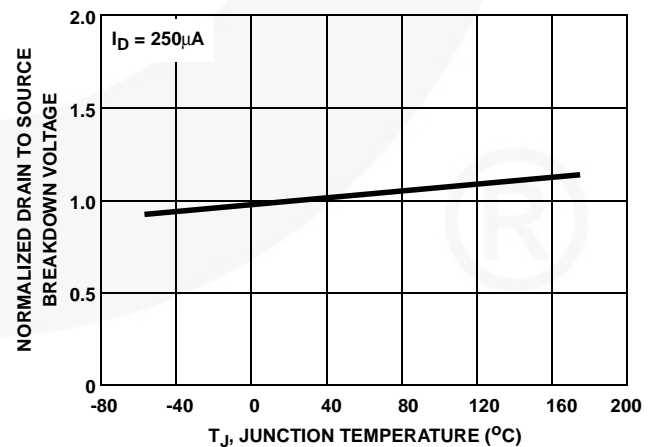


FIGURE 11. NORMALIZED DRAIN TO SOURCE BREAKDOWN VOLTAGE vs JUNCTION TEMPERATURE

Typical Performance Curves $T_C = 25^\circ\text{C}$, Unless Otherwise Specified (Continued)

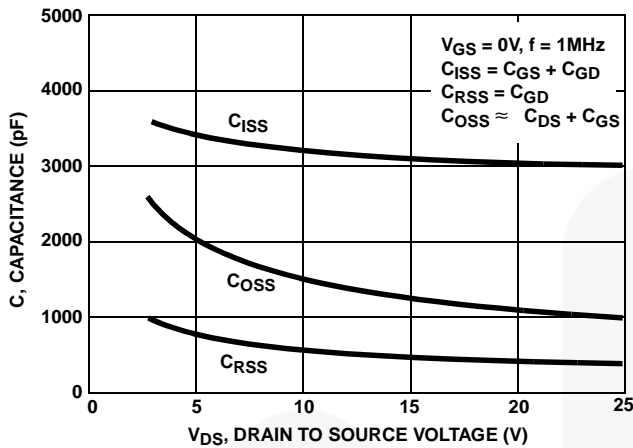
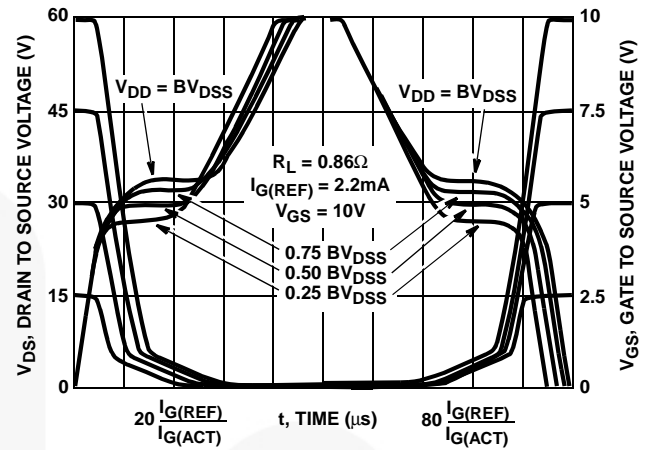


FIGURE 12. CAPACITANCE vs DRAIN TO SOURCE VOLTAGE



NOTE: Refer to Fairchild Application Notes AN7254 and AN7260.

FIGURE 13. NORMALIZED SWITCHING WAVEFORMS FOR CONSTANT GATE CURRENT

Test Circuits and Waveforms

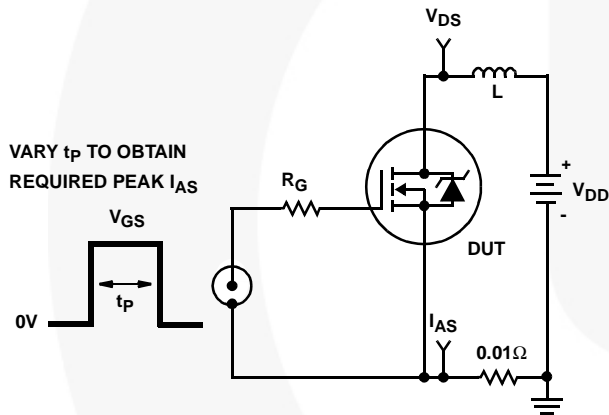


FIGURE 14. UNCLAMPED ENERGY TEST CIRCUIT

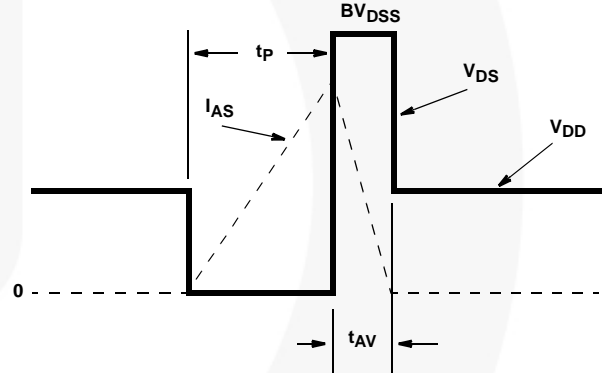


FIGURE 15. UNCLAMPED ENERGY WAVEFORMS

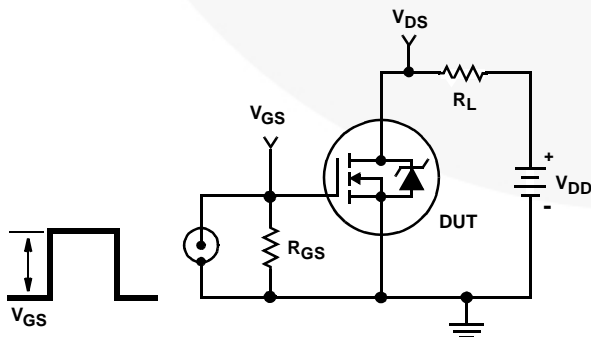


FIGURE 16. SWITCHING TIME TEST CIRCUIT

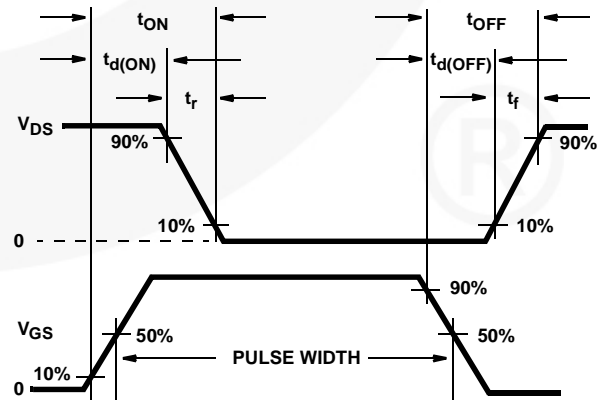


FIGURE 17. SWITCHING WAVEFORMS

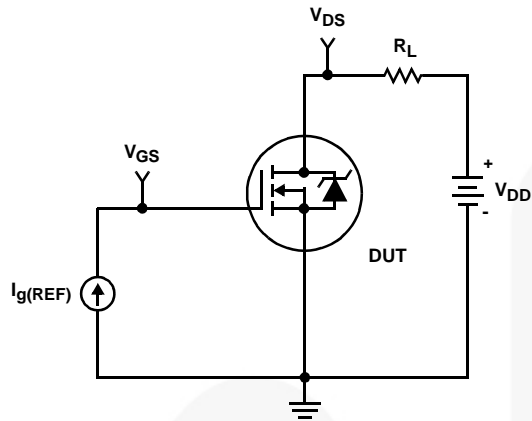
Test Circuits and Waveforms (Continued)

FIGURE 18. GATE CHARGE TEST CIRCUIT

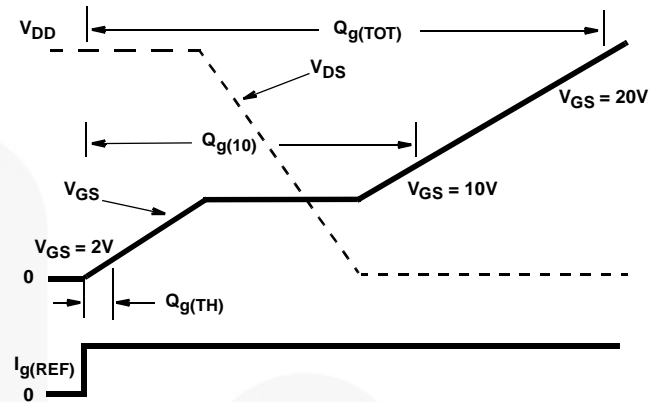




FIGURE 19. GATE CHARGE WAVEFORM



TRADEMARKS

The following includes registered and unregistered trademarks and service marks, owned by Fairchild Semiconductor and/or its global subsidiaries, and is not intended to be an exhaustive list of all such trademarks.

AccuPower™	F-PFS™	PowerTrench®	Sync-Lock™
AX-CAP®*	FRFET®	PowerXS™	SYSTEM GENERAL®*
BitSiC™	Global Power Resource SM	Programmable Active Droop™	TinyBoost®
Build it Now™	GreenBridge™	QFET®	TinyBuck®
CorePLUS™	Green FPS™	QS™	TinyCalc™
CorePOWER™	Green FPS™ e-Series™	Quiet Series™	TinyLogic®
CROSSVOLT™	Gmax™	RapidConfigure™	TINYOPTO™
CTL™	GTO™	TM	TinyPower™
Current Transfer Logic™	IntelliMAX™	Saving our world, 1mW/W/kW at a time™	TinyPWM™
DEUXPEED®	ISOPLANAR™	SignalWise™	TinyWire™
Dual Cool™	Marking Small Speakers Sound Louder and Better™	SmartMax™	TranSiC™
EcoSPARK®	MegaBuck™	SMART START™	TriFault Detect™
EfficientMax™	MICROCOUPLER™	Solutions for Your Success™	TRUECURRENT®*
ESBC™	MicroFET™	SPM®	μSerDes™
	MicroPak™	STEALTH™	
Fairchild®	MicroPak2™	SuperFET®	UHC®
Fairchild Semiconductor®	MillerDrive™	SuperSOT™-3	Ultra FRFET™
FACT Quiet Series™	MotionMax™	SuperSOT™-6	UniFET™
FACT®	mWSaver®	SuperSOT™-8	VCX™
FAST®	OptoHiT™	SupreMOS®	VisualMax™
FastvCore™	OPTOLOGIC®	SyncFET™	VoltagePlus™
FETBench™	OPTOPLANAR®		XS™
FPS™			

*Trademarks of System General Corporation, used under license by Fairchild Semiconductor.

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION, OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS. THESE SPECIFICATIONS DO NOT EXPAND THE TERMS OF FAIRCHILD'S WORLDWIDE TERMS AND CONDITIONS, SPECIFICALLY THE WARRANTY THEREIN, WHICH COVERS THESE PRODUCTS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.

As used here in:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support, device, or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

ANTI-COUNTERFEITING POLICY

Fairchild Semiconductor Corporation's Anti-Counterfeiting Policy. Fairchild's Anti-Counterfeiting Policy is also stated on our external website, www.Fairchildsemi.com, under Sales Support.

Counterfeiting of semiconductor parts is a growing problem in the industry. All manufactures of semiconductor products are experiencing counterfeiting of their parts. Customers who inadvertently purchase counterfeit parts experience many problems such as loss of brand reputation, substandard performance, failed application, and increased cost of production and manufacturing delays. Fairchild is taking strong measures to protect ourselves and our customers from the proliferation of counterfeit parts. Fairchild strongly encourages customers to purchase Fairchild parts either directly from Fairchild or from Authorized Fairchild Distributors who are listed by country on our web page cited above. Products customers buy either from Fairchild directly or from Authorized Fairchild Distributors are genuine parts, have full traceability, meet Fairchild's quality standards for handling and storage and provide access to Fairchild's full range of up-to-date technical and product information. Fairchild and our Authorized Distributors will stand behind all warranties and will appropriately address and warranty issues that may arise. Fairchild will not provide any warranty coverage or other assistance for parts bought from Unauthorized Sources. Fairchild is committed to combat this global problem and encourage our customers to do their part in stopping this practice by buying direct or from authorized distributors.

PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative / In Design	Datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	Datasheet contains preliminary data; supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve design.
No Identification Needed	Full Production	Datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve the design.
Obsolete	Not In Production	Datasheet contains specifications on a product that is discontinued by Fairchild Semiconductor. The datasheet is for reference information only.

Rev. 166