Informe práctica final

Fernando Santana & Jorge Calvar

Microprocesadores

Profesor: Francisco Martín Martínez

Índice

Introducción

En este informe, se explica el desarrollo de la práctica final, que consiste en la creación de un dron controlado por el PIC32. El dron consta de los siguientes sensores:

- **Módulo Bluetooth:** lo que permitirá la comunicación con la app controladora en el móvil.
- **Luces LED:** parpadearán para dar un poco de alegría al dron.
- **Sensor ultrasonidos:** muestra la cercanía de objetos al dron.
- **Cuatro motores:** se conectan a través de 4 ESCs (*Electronic Speed Controllers*) y se controlan con ondas PWM de entre 1 y 2us a nivel alto, de manera similar a un servo.
- **Giroscopio:** utilizaremos el MPU6050 que utiliza el protocolo de comunicaciones I2C. Desgraciadamente, no hemos conseguido funcionamiento práctico por errores del chip, pero comentamos el código en este informe.

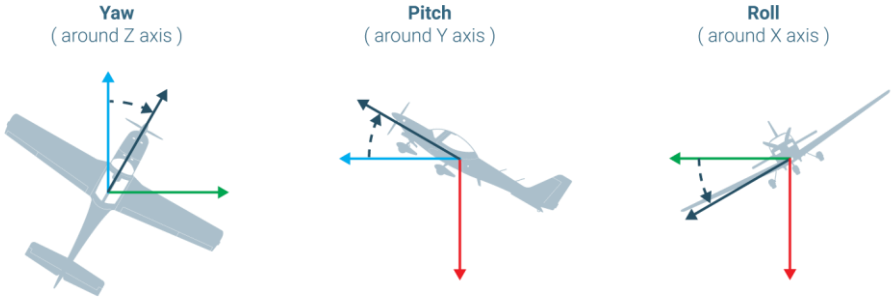
Funcionamiento de una aeronave

En esta sección explicaremos de manera breve los fundamentos del funcionamiento de un vehículo aéreo.

Ejes de movimiento

Un aparato aéreo puede, en una misma posición, cambiar su orientación siguiendo tres ejes diferentes. Estos movimientos se denominan:

- **Yaw**: movimiento en el eje vertical.
- **Roll**: movimiento en el eje longitudinal.
- **Pitch**: movimiento en el eje transversal.



Además, en un dron hay un cuarto movimiento que difiere de los anteriores en que no es un cambio de orientación, sino de posición. Se denomina **Throttle** y aumentando o disminuyendo su valor se consigue un cambio de posición siguiendo la recta perpendicular al plano del dron.

Control

El control se establece normalmente utilizando algoritmos PID utilizando la información del giroscopio. Nosotros, por simplicidad, hemos creado un algoritmos simple (aunque no funcional, por lo que el dron no se mantendrá en equilibrio) que se expone a continuación:

Motor	Instrucción
Front Left	Throttle + Roll – Pitch - Yaw
Front Right	Throttle – Roll – Pitch + Yaw
Back Left	Throttle + Roll + Pitch + Yaw
Back Right	Throttle – Roll + Pitch - Yaw

Se observa que **Throttle** es proporcional en todos los motores, **Roll** cambia de signo en los motores de la derecha y la izquierda, **Pitch** cambia de signo en los motores de delante y detrás, y **Yaw** cambia de signo según los motores de las diagonales (que se mueven en direcciones inversas).

Código

En esta sección se comentan las diferentes unidades de código de las que se compone el proyecto.

Receiver (Jorge)

La función de este *driver* es descifrar las instrucciones que se reciben por parte del controlador del móvil. Por tanto, este módulo hace uso del *driver_uart* desarrollado en clase. Estas instrucciones están codificadas en ASCII y siempre vienen precedidas por una 'S' y finalizan con '\n'. Las posibles instrucciones son:

Instrucción	Descripción
D / d	Permite activar / desactivar las hélices.
P / p	Controla el funcionamiento de los LEDs.
RPX*Y*	Recibe el Roll y el Pitch.
TYX*Y*	Recibe el Throttle y el Yaw.

Los asteriscos se refieren a un número entre 0 y 1000.

void initReceiver();

Se ejecuta al principio para realizar las inicializaciones pertinentes.

void computeRX();

Se ejecuta recurrentemente (normalmente en cada iteración del bucle de scan) para leer los caracteres que ha recibido el *driver_uart*. Si se recibe un salto de línea (\n) se llamará a la función *executeCommand()*.

void executeCommand();

Lee la instrucción recibida haciendo uso de las siguientes funciones y guarda los valores correspondientes en variables globales.

char* readNextCommand();

Obtiene del buffer la siguiente instrucción a procesar.

void readCoordinates(char* command, int len_command, int8_t start_i, int x[], int y[]);

Lee las coordenadas de una instrucción RP o TY y las guarda en los punteros x e y.

void checkIntegrity(char* command, int len_command);

Comprueba que la instrucción recibida tiene un formato adecuado. Esta función es imprescindible para evitar errores.

void sendInfo();

Debe ejecutarse periódicamente, envía información del estado de los motores y la distancia leída por el sensor de ultrasonidos a la aplicación móvil para ser mostrada en la pantalla.

Ultrasonidos (Fernando)

Este módulo se encarga de las tareas relacionadas con el sensor de ultrasonidos. Para medir la longitud de la señal ECHO se utilizará el Timer 5. Por otro lado, para mediar el tiempo de activación de la señal Trig utilizaremos el Timer 1. Este timer será aprovechado a su vez para crear una función que devuelve los ticks desde el inicio del programa. Nos hubiera gustado que los ticks se incrementasen cada microsegundo, pero esto no ha sido posible debido a que la frecuencia del reloj no es lo suficientemente elevado. Por tanto, el incremento se produce cada 10 microsegundos.

Las funciones principales del módulo son:

void InizializarTemp();

Inicializar el temporizador 1.

void InicializarUltra();

Inicializar el temporizador 5 y lo conecta con el pin que se conectará a ECHO.

void iniciarMedidad();

Inicia una medida de distancia si no se está en proceso una medida.

void InterrupcionTimer1();

Incrementa ticks y controla la emisión de Trigger y el parpadeo de los LEDs.

Cuando por la UART se envía una P se activa el modo policía que invierte los Leds azul y rojo cada 0,25 segundos.

void InterrupcionTimer5();

Al finalizarse la medida de ECHO, la guarda en la variable correspondiente.

int getDistancia();

Punto de acceso externo para leer la última distancia medida por el sensor de ultrasonidos.

Int32_t micros();

Pese a su nombre, devuelve los ticks (múltiplos de 10us) desde el inicio del programa.

Output (Jorge)

void initOutput();

Inicializa el Timer 2 y los Output Compare 1-4 necesarios para generar las ondas PWM que controlan los motores.

void writeMotors();

Actualiza el ciclo de trabajo de los PWM.

void computeMotors();

Obtiene utilizando la fórmula explicada en el apartado de Control el valor correspondiente para cada motor utilizando la información recibida del móvil.

Sensors (Fernando y Jorge)

Este módulo obtenido del proyecto de código de abierto Multiwii lee la información del giroscopio utilizado: el MPU6050. Hace uso para ello del módulo I2C.h ya que la comunicación se realiza utilizado este protocolo.

Las funciones principales son:

void GYRO_Common();

Realiza el calibrado del giroscopio y garantiza que no se produzcan saltos demasiado bruscos.

void Gyro_getADC();

Lee los valores actuales del giroscopio accediendo a los registros correspondientes.

void initSensors();

Inicializa el driver I2C y el giroscopio aplicando la configuración necesaria.

Otros

Otros archivos de código incluyen:

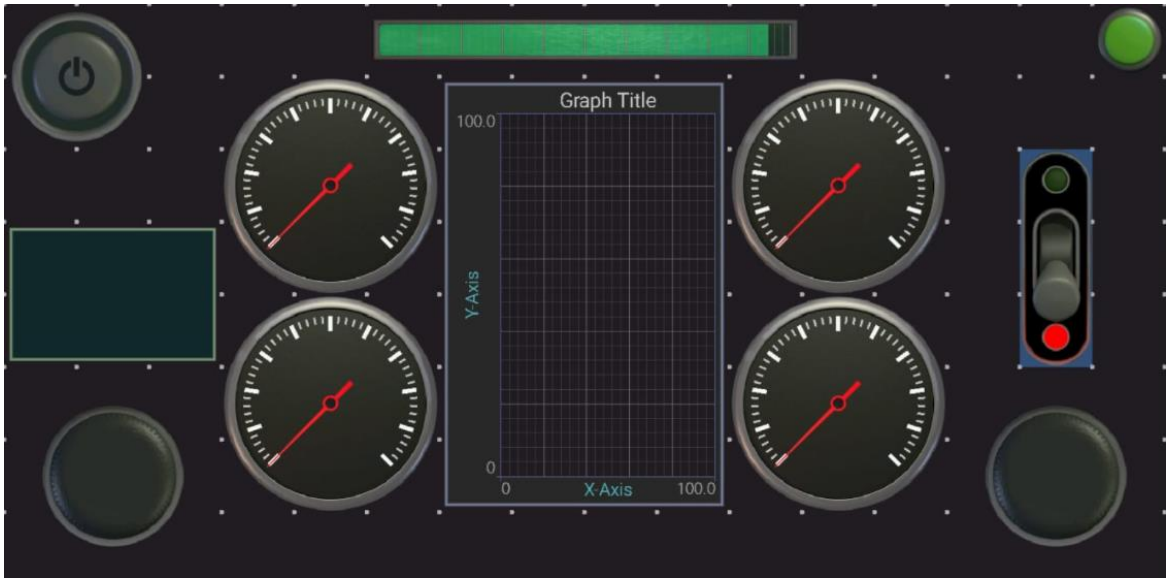
Utils.c: funciones de utilidad para ahorrar código, especialmente simplifica el trabajo con cadenas.

I2C.c: funciones desarrolladas en clase y derivadas para la comunicación I2C.

App controladora

Para controlar el dron se ha utilizado la aplicación *Bluetooth Electronics* de Keuwlsoft para Android https://www.keuwl.com/apps/bluetoothelectronics/

En la aplicación se ha diseñado un mando y la conexión al microcontrolador se realiza utilizando la UART controlada por Bluetooth.



Con el botón de arriba a la izquierda se envía una D que arma los motores.

En la pantalla que se sitúa debajo podemos ver los comandos enviados por la UART lo que se ha utilizado sobre todo para el proceso de debug.

En la parte superior vemos una barra donde se representa la distancia medida por el ultrasonidos. Pese a que el sensor puede medir entre 4cm y 4m, la medida en la barra se representa hasta los 2m ya que las distancias que nos interesan son cortas. Esto es porque este sensor se utilizaría inicialmente para medir la distancia con el suelo, aunque se plantearía la inclusión de hasta 4 sensores ultrasonidos más para medir la distancia en todos los lados.

En la parte inferior tenemos los dos joysticks que controlan los motores cuyo resultado se muestra en los cuatro medidores de aguja.

Los medidores están colocados acorde al dron para que se vea claramente la potencia a la que funciona cada motor ya que no se puede apreciar simplemente viéndolo girar.

A la derecha encontramos un interruptor para activar el modo policía que envía una P por la UART activando el parpadeo intermitente de 3 Leds.

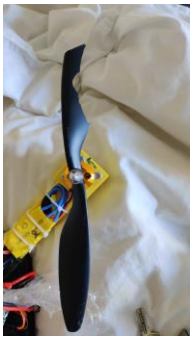
La pantalla del medio se pretendía utilizar para recibir los datos del giroscopio y acelerómetro mediante I2C para lo que se incluía también un botón en la esquina superior derecha que calibrara el giroscopio.

Prueba real

Para comprobar el funcionamiento global del dron y especialmente que los motores tienen suficiente fuerza como para levantar el peso del dron hemos realizado una prueba física en la que hemos activado los motores e incrementado relativamente la intensidad del throttle.

Tras llegar más o menos a la mitad de potencia el dron se levanta de manera brusca y se desequilibra. La causa más improbable de esta inestabilidad inicial es la distribución asimétrica de la masa del dron.

En el primer fotograma se puede observar el primer momento tras el levantamiento del suelo. A continuación, viene representado el desequilibrio y por último una imagen en la que se observa la hélice rota tras impactar contra el suelo:



Como conclusiones podemos extraer:

- Es recomendable distribuir el peso de los componentes de manera lo más simétricamente posible en los ejes x e y, para facilitar la estabilidad tras el despegue.
- Es imprescindible el uso de un software avanzado de estabilidad mediante giroscopio para garantizar la estabilidad tras el despegue. Aunque se puede lograr una versión con mínima funcionalidad con un giroscopio es recomendable tener también un acelerómetro. Un magnetómetro también contribuye.

Conclusión

Al comenzar el proyecto conocíamos la dificultad de realizar un dron completamente funcional que lograra estabilizarse y volar además de lo peligrosas que podían ser las pruebas por lo que nos centramos en demostrar que los motores funcionaban correctamente acorde a los joysticks, que se midiera la distancia de manera precisa y en añadir otras salidas como unos leds intermitentes activables desde el mando. Gran parte de la dificultad para que un dron funcione del todo reside en su estabilización y diseño. Los algoritmos de estabilización utilizando los datos del giroscopio acelerómetro son complejos y adaptar largos códigos que ya está diseñado para otros microcontroladores al PIC32 es complejo.

En clase se observó perfectamente como los 4 motores reaccionaban perfectamente a todos los posibles movimientos de los joysticks. El sensor ultrasonidos que se ubicaría en la parte inferior para medir la distancia con el suelo y ayudar en el despegue y aterrizaje también medía de manera precisa la distancia

Durante el desarrollo del proyecto nos hemos enfrentado a numerosos obstáculos, muchos de los cuales hemos conseguido resolver entendiendo los errores cometidos y aprendiendo de ellos. Sin embargo, también hay errores en el uso de la PIC32 que no tienen explicación. Por ejemplo, los inicializar el output después de la UART provocaba un fallo en el programa. Esto solo se solucionó al invertir el orden de inicialización.