Practica 1: Control de ServoMotores del Skybot v6

Ignacio Herrero Reder, Juan Pedro Bandera

Esta práctica sirve de introducción al control de los servomotores que se usarán en el microbot Skybot. Aprenderemos a configurar los recursos de la placa de control TIVA EK-TM4C123GXL para generar ondas PWM con ciclo de trabajo controlado, que determinarán la velocidad de rotación de los servos. Como los servomotores del Skybot han sido trucados para permitir un movimiento de giro completo, la velocidad de giro estará relacionada con el error entre el valor correspondiente a la posición fija del potenciómetro y el valor deseado programado para la señal PWM; a mayor error, mayor será la velocidad de movimiento del servo. Aprenderemos a caracterizar – de forma muy básica - la curva que relaciona el ciclo de trabajo de la señal PWM con la velocidad del servo[1]. A continuación nos centraremos en la coordinación de los dos servos de nuestro microbot para obtener diferentes tipos de movimientos (rectilíneo, curva, rotación,...). Opcionalmente trabajaremos con el control del movimiento de Skybot mediante comandos enviados desde el PC a través del interfaz USB/serie, tanto desde un programa de terminal como desde un interfaz gráfico QT.

1 Introducción

En esta asignatura trabajaremos con un micro ARM de la familia TIVA-C (TM4C123GH6PM) y el entorno de desarrollo *Code Composer Studio (CCS)*. Los estudiantes de esta asignatura ya deben tener conocimientos acerca de ambos aspectos, al haber trabajado con componentes parecidos en asignaturas como Microcontroladores y Sistemas Empotrados.

Para esta práctica, especialmente en los primeros ejercicios propuestos, recomendamos encarecidamente refrescar los conocimientos acerca de TIMERS, PWM, y PUERTOS E/S de la familia de la placa escogida para el control del microbot, la TIVA, que se trató en la asignatura Sistemas Empotrados; o, si se usa otra distinta, su documentación correspondiente. Así mismo, podéis repasar los aspectos de configuración de proyectos y depuración con CCS que se impartieron en asignaturas anteriores (Sistemas Empotrados y Microcontroladores). Solamente se tendrán que realizar algunos cambios, si se utiliza otro micro o placa de desarrollo distinta, que se indicarán en

1 Cuando trabajemos en “bucle cerrado” podremos hacer una caracterización más precisa de dicha curva.

el siguiente apartado.

2 Creación de un proyecto en CCS para la EXP-

Las instrucciones para crear un proyecto CCS con la placa TIVA ya se indicaron asignaturas anteriores (Sistemas Empotrados - Microcontroladores). Se recomienda consultar la documentación correspondiente.

3 Ejercicio 1. Generación de ondas PWM con ciclo de trabajo. Programación mediante HAL/API.

El año pasado, en la asignatura “Sistemas Empotrados”, se aprendió a programar aplicaciones utilizando librerías de alto nivel (*TIVAWare* para la *TIVA Launchpad*) que permitían abstraerse de la programación a bajo nivel y concentrarse en los aspectos algorítmicos y de control de la aplicación a desarrollar. Además, el uso de este tipo de librerías suele permitir la migración de aplicaciones escritas para un determinado micro de la familia a otros micros, sin apenas esfuerzo, al compartirse la mayoría de funciones de API. En el documento [TIVAxxDRLIB], se pueden consultar las funciones de alto nivel disponibles en la librería TIVAWare.

En este ejercicio programaremos los recursos del micro TM4C123GH6PM de nuestro placa de desarrollo para generar una señal PWM con ciclo de trabajo controlable mediante los botones de la placa. La señal, que saldrá por el terminal PF2 de la placa, se conectará a la señal de datos (cable blanco o naranja), de un servo, para controlar su velocidad en función del ciclo de trabajo seleccionado mediante un pulsador. También se generará una segunda señal con un valor de amplitud fijo, por el terminal PF3.

Para comprobar que los servomotores “funcionan”, hemos proporcionado un proyecto de test[2] que implementa este primer ejercicio, solo que dicho proyecto usa librerías, por lo que no es posible ver su código completo. De los dos servos proporcionados, uno debería estar ajustado correctamente, por lo que al pasar el proyecto de test, permanecerá parado -o casi- hasta pulsar los botones de la placa. Una vez que compruebes que los servos funcionan, deberás implementar el código del ejercicio que implemente la misma funcionalidad que el proyecto de test, TestServosTIVA.

Materiales necesarios para el ejercicio:

2 Está en la carpeta “Ejemplos y código de partida”

- Data-Sheet TM4C123GH6PM [TM4CDS] y Errata-Sheet [TM4CES]
- Apuntes Sistemas Empotrados: puertos y Timers [SISTEMP]
- Ejemplos de uso de periféricos de la TIVA (en paquete TIVAware)
- Apuntes Microbotica, funcionamiento PWM[3]
- Guía de usuario placa *TIVA EK-TM4C123GXL* [TIVAUG]
- ServoMotor FUTABAS3003 (incorporado en el kit del robot; en este caso, procura colocar la carcasa del robot "en alto", de manera que las ruedas queden girando "en el aire")

Para la realización del ejercicio partiremos de un esqueleto de código fuente en donde se indica, mediante comentarios, las instrucciones que es necesario programar para la implementación de la aplicación:

código esqueleto – simplificado - para el ejercicio 1: PWM-Servo

```
/* ************************************************************************ */
/* (ver cabecera en el código del programa)                                 */
/* ************************************************************************ */
#include <stdint.h>
#include <stdbool.h>
// Librerias que se incluyen tipicamente para configuracion de perifericos y pinout
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/pin_map.h"
// Libreria de control del sistema
#include "driverlib/sysctl.h"
// Incluir librerias de periférico y otros que se vaya a usar para control PWM y gestión
// de botones  (TODO)

#define PERIOD_PWM XYXYXY  // TODO: Ciclos de reloj para conseguir una señal periódica de
50Hz (según reloj de periférico usado)
#define COUNT_1MS XXXX   // TODO: Ciclos para amplitud de pulso de 1ms (max velocidad en
un sentido)
#define STOPCOUNT YYYY  // TODO: Ciclos para amplitud de pulso de parada (1.52ms)
#define COUNT_2MS ZZZZ   // TODO: Ciclos para amplitud de pulso de 2ms (max velocidad en
el otro sentido)
#define NUM_STEPS 50    // Pasos para cambiar entre el pulso de 2ms al de 1ms
#define CYCLE_ INCREMENTS (abs(COUNT_1MS-COUNT_2MS))/NUM_STEPS  // Variacion de amplitud
tras pulsacion

int main(void){
    uint32_t ui32Period, ui32DutyCycle;
```

3 Si decides usar la TIVA, recuerda que tiene módulos generadores de señales PWM , por lo que puede sea mejor no usar los Timers. Consulta la documentación y los ejemplos TIVAware.

```
    // Elegir reloj adecuado para los valores de ciclos sean de tamaño soportable
    SysCtlClockSet(¿?¿?¿);

  // Configura pulsadores placa TIVA (int. por flanco de bajada)


  // Configuracion  ondas PWM: frecuencia 50Hz, anchura inicial= valor STOPCOUNT, 1540us
para salida por PF2, y COUNT_1MS (o COUNT_2MS ) para salida por PF3(puedes ponerlo
inicialmente a  PERIOD_PWM/10)
      // Opcion 1: Usar un Timer en modo PWM (ojo! Los timers PWM solo soportan cuentas
      //  de 16 bits, a menos que uséis un prescaler/timer extension)
      // Opcion 2: Usar un módulo PWM(no dado en Sist. Empotrados pero mas sencillo)
      // Opcion 1: Usar un Wide Timer (32bits) en modo PWM (estos timers soportan
      //  cuentas de 32 bits, pero tendréis que sacar las señales de control pwm por
      //  otros pines distintos de PF2 y PF3)


  // Codigo principal, (poner en bucle infinito o bajo consumo)
}

// Rutinas de interrupción de pulsadores
// Boton Izquierdo: modifica  ciclo de trabajo en CYCLE_INCREMENTS para el servo
conectado a PF2, hasta llegar a  COUNT_1MS
// Boton Derecho: modifica  ciclo de trabajo en CYCLE_INCREMENTS para el servo conectado
a PF2, hasta llegar a COUNT_2MS
```

Rellena las instrucciones que faltan y añádelo a un proyecto. Puedes partir de algún proyecto antiguo de Sistemas Empotrados, o utilizar algún proyecto de ejemplo de la librería TivaWare (hay ejemplos de uso de Timer como PWM y de los módulos PWM).

Para determinar el valor de `STOPCOUNT`, ten en cuenta las indicaciones del servo: en teoría una anchura de pulso de 1520uS mantendría al servo parado (si has trucado correctamente el servo) . Si para este valor no se para modificarlo (por ahora se recomienda usar ese valor como una primera aproximación y usar los pulsadores para ajustar la anchura de pulso hasta lograr que el servo se detenga. En ese momento puedes parar la aplicación, observar el valor del registro y cambiar `STOPCOUNT` a este valor para futuras ejecuciones. Puede que prefieras poner, momentáneamente, `NUM_PASOS` a otro valor mayor para este ajuste fino inicial). **Ten en cuenta que los periféricos que puedes usar (Timer o PWM) suelen tener un tamaño máximo de registro de 16 bits, por lo que un reloj del periférico a mucha frecuencia puede resultar en valores demasiado grandes.**

Necesitarás usar valores de reloj menores que los “habituales” -lo que implica relojes del sistema a menor frecuencia de la habitual y/o factores de división altos en los relojes de periférico-.

En cuanto a los valores de `MINCOUNT` y `MAXCOUNT`, ahora mismo es muy complicado calcularlos (estamos trabajando en “bucle abierto”), por lo que se recomienda colocarlos “ a ojo” (simplemente acelerando el motor, observando cuando parece que la rueda ya no va más rápido al pulsar el botón, y depurando el programa para ver el valor en ciclos)[4]. En realidad, la velocidad máxima del servo en uno y otro sentido depende de varios factores:

1. Máxima variación de error en el circuito de realimentación del servo
2. Máxima corriente que se puede entregar al motor → cambiará cuando el motor se alimente a diferente tensión (5V o 3.3V) o con diferente batería (un puerto USB tiene un límite de 500mA; una pila puede llegar a ofrecer más corriente)
3. Características y simetría de la curva error/velocidad del servo.

Construye la aplicación y descárgala en la placa (se recomienda conectar en primer lugar las conexiones del servo a la placa, antes de conectar el cable USB de la placa al PC). Pon en marcha la aplicación, y comprueba que funciona (al pulsar los botones debería aumentar y disminuir la velocidad de giro del servo conectado a PF2 -CON_3 exterior en la Booster Pack-; el servo conectado a PF3 -CON_3 interior- se debe mover a velocidad fija desde el inicio). Calcula los valores más adecuados para `para MINCOUNT, STOPCOUNT y MAXCOUNT`. (No pierdas mucho tiempo en ajustarlos al detalle ya que !puede que tengamos que volver a reajustarlos!)

4 Ejercicio 2. Caracterización básica de los servos de Skybot

Utiliza el programa generado en el apartado anterior, en el que se hace uso de la API TIVAware para la programación de una onda PWM de control de un servo para realizar (!de nuevo!) la caracterización básica de los servos izquierdo y derecho de tu Skybot. Abre los servos (con cuidado!! pregunta a los profesores al menos la primera vez!!) y, teniendo establecido el ciclo de trabajo a la anchura de parada (1520uS), mueve ligeramente el potenciómetro hasta que el motor se pare. Con cuidado vuelve a cerrar los servos, procurando no volver a mover el potenciómetro. No te preocupes si no consigues parar exactamente los servos en la anchura de parada: el servo sufre golpes y rozamientos que pueden mover ligeramente el potenciómetro cambiando el valor teórico.

4 Cuando trabajemos en bucle cerrado veremos formas de caracterizar mejor las velocidades máximas que puede alcanzar el servo.

Una vez cerrado, modifica ligeramente el valor de STOPCOUNT hasta conseguir que se corresponda con un estado de parada (puede que en vez de 1520us sea finalmente 1500 o 1580us).

También puedes realizar una estimación de los valores máximos y mínimos de cuenta para cada servo simplemente observando en que momento la rueda parece “dejar de ir más rápido”. En vez de usar unos valores fijos `MINCOUNT`, y `MAXCOUNT`, (que serán distintos para cada servo), otra aproximación es considerar la curva de velocidad respecto al ciclo de trabajo como lineal y simétrica con respecto a la cuenta de parada, `STOPCOUNT` y asignar un mismo límite de variación con respecto al valor “central” (esto es , `MINCOUNT = STOPCOUNT – VALOR`, y `MAXCOUNT = STOPCOUNT + VALOR`). Más adelante podremos realizar una estimación un poco más precisa de los valores máximo de velocidad en uno y otro sentido.

5 Ejercicio 3. Control simultáneo de ambos servos

En los apartados anteriores hemos trabajado con una sola señal PWM variable, la obtenida en el pin PF2. En este ejercicio se propone que la señal PWM por PF3 también sea variable, para poder conseguir diferentes tipos de desplazamiento para el robot Skybot.

5.1 Ejercicio 3A. Movimiento rectilíneo del Skybot

Diseña un programa o aplicación que permite controlar un movimiento rectilíneo del Skybot, tanto hacia delante como hacia atrás. Partiendo desde una posición de parada, al pulsar el botón izquierdo de la TIVA, el robot debe avanzar de forma recta a una velocidad cada vez mayor, hasta alcanzar una velocidad límite; si el robot está avanzando, la pulsación del botón derecho dará lugar a una disminución de la velocidad de avance hasta llegar al estado de parada y, a partir de aquí, un movimiento de retroceso a cada vez mayor velocidad hasta llegar a la velocidad máxima de retroceso. Puedes programar un Timer que se encargue de ir reduciendo y aumentando paulatinamente la anchura de la onda PWM a un determinado ritmo para no tener que apretar mas los botones mas que una sola vez.

Ten en cuenta que ahora, para cambiar la velocidad, deberemos modificar simultáneamente el control de ambos servos, de forma que las ruedas se mantengan moviéndose a la misma velocidad. Acuérdate también que las ruedas se colocan **en oposición**.

5.2 Ejercicio 3B. Movimiento curvo del Skybot

Para poder mover al Skybot siguiendo una trayectoria curva, basta con aplicar un **movimiento diferencial** a sus ruedas, es decir, mover estas a diferentes velocidades. Si la rueda izquierda se mueve a más velocidad que la derecha, el microbot girará a la derecha con un ángulo

mayor cuanto mayor sea la diferencia de velocidad; si por el contrario se mueve a menos velocidad, el microbot girará hacia la izquierda con un ángulo mayor cuanto mayor sea su diferencia.

En este ejercicio se arrancará con el robot moviéndose hacia delante a la mitad de la máxima velocidad posible de avance (ambos servos con el mismo valor de avance, pero en sentido contrario). Uno de los servos (por ejemplo, el derecho), permanecerá moviéndose a esa velocidad de arranque, sin cambios, durante toda la ejecución de la aplicación. El otro, y considerando que al arranque está funcionando a mitad de velocidad de avance, disminuirá su velocidad al pulsar el botón izquierdo, hasta como mucho su estado de parada; y aumentará su velocidad al pulsar el botón derecho hasta alcanzar su velocidad máxima. De esta forma se conseguirá realizar diversos movimientos curvos en el microbot, siempre en un sentido de avance.

5.3 Ejercicio 3C. Movimiento rotatorio del Skybot

El movimiento con trayectoria curva conseguido en el ejercicio anterior a través del movimiento diferencial de sus servos tiene el inconveniente de que es necesario un cierto ángulo de giro en la trayectoria, lo cual impide la realización de giros rápidos. No obstante, si consideramos que cada uno de los servos tiene dos posibles sentidos de giro, podemos conseguir en el Skybot un movimiento de rotación sobre su punto central sin más que aumentar simultáneamente la velocidad de giro de uno de los servos en un sentido, y la del otro servo **en el otro sentido**. De esta forma podemos conseguir un movimiento rotatorio que permita realizar giros rápidos.

En este ejercicios se propone partir de un estado con ambos servos parados e imprimir un movimiento de rotación hacia la izquierda al pulsar el botón izquierdo, cada vez más rápido a medida que se pulsa mas veces dicho botón. Si al estar rotando el microbot hacia la izquierda se pulsa el botón derecho, la velocidad de rotación hacia la izquierda irá disminuyendo hasta llegar al estado de parada. A partir de entonces, las pulsaciones del botón derecho imprimirán un movimiento de rotación hacia la derecha cada vez más rápido hasta llegar a su valor máximo.

Ejercicios Opcionales **(necesitas un segundo cable USB-microUSB)**

6 Ejercicio 4. Comunicación entre la placa y el PC

Tras realizar los ejercicio anteriores habrás podido observar como es complicado, usando solo dos botones, obtener en una misma aplicación una cierta variedad de movimientos del Skybot, a menos que se programen los botones en modo multifunción. Sin embargo, es posible realizar un control a través de un interfaz o terminal de usuario, lo cual nos permitiría un número casi ilimitado de posibles interacciones con los servos, sin mas que programar los comandos adecuados.

Como inconveniente de este método, se necesita un cable de conexión fija entre el PC y el microbot, por lo que este método de comunicación no es interesante si se supone que el robot se va a mover.

En la asignatura *Sistemas Empotrados* se pudieron ver varios ejemplos de programación de la USCI1 de la placa para una comunicación serie mediante un *backchannel* implementado en la conexión USB a través del depurador. Para ello se usan las funciones de API correspondientes en la librería *utils* de TIVAWare.

- Importa el proyecto de ejemplo *USB_Serial*[5] a tu espacio de trabajo construye la aplicación, descárgala a la placa, y ejecútala. Averigua cual es el puerto de comunicaciones abierto por la placa TIVA. Para ello – en Windows - accede al *Administrador de dispositivos,* y explora el apartado *Puertos (COM y LPT)*. Allí deberá existir una entrada *TIVAware USB Serial Port (COMXX).* Apunta el número del puerto COM.
- Utilizando un programa de terminal – por ejemplo, puTTy – abre una conexión serie a 9600 baudios por el puerto indicada en el apartado anterior. El resto de configuraciones son las seleccionadas por defecto (8-N-1, y control de flujo: “ninguno” o “Xon/Xoff”).
- Si arrancas el puTTy antes de 10segundos desde la ejecución del códiogo, verás aparecer un mensaje de “bienvenida” en el terminal. Además, partir de ese momento la pulsación en el terminal de las teclas 'r', 'g', 'b' y 'a' provocarán diferentes respuestas en los LEDs de la placa.
- Observa las respuestas y analiza la configuración y funcionamiento de la aplicación de ejemplo. Se usará como base para el siguiente ejercicio.

7 Ejercicio 5. Control de los servomotores del Skybot mediante las teclas de un terminal

Partiendo de la aplicación de ejemplo de comunicación serie del apartado anterior, y de las diferentes aplicaciones implementadas para el control de diversos tipos de movimiento con el Skybot, implementa una aplicación que permita el control del movimiento de los servomotores del Skybot mediante la pulsación de teclas en un terminal puTTy, enviadas a la placa a través del canal de comunicación serie. La pulsación de las teclas 'q' y 'w' deberá permitir aumentar la velocidad de las ruedas izquierda y derecha, respectivamente en sentido de “avance” del microbot; la pulsación de las teclas 'a' y 's' reducirá dicha velocidad en los respectivos servos, si el microbot estaba avanzando, hasta llegar al valor de parada, a partir del cual se aumentará la velocidad de los servos

5 Este proyecto YA incorpora el sistema operativo FreeRTOS, por lo que te podría servir también de ejemplo para la estructura de la práctica final de la asignatura.

pero en sentido de retroceso. La pulsación de la tecla 'p' detendrá inmediatamente ambos servos, dejando al sistema en condiciones similares a las existentes en el arranque de la aplicación. **Nota: ten en cuenta que, al no tener comunicaciones inalámbricas, deberás mantener conectado el cable USB_Device para enviar las ordenes. Es muy recomendable que os busquéis algún elemento sobre el que colocar el robot, para que las ruedas giren sin estar en contacto con el suelo.**

8 Ejercicio 6. Control de los servomotores del Skybot mediante un GUI QT

El curso pasado, en la asignatura “Sistemas Empotrados' se estudió como diseñar interfaces gráficos de usuario (GUI) mediante la librería QT. Estos interfaces permitían interactuar con una placa microcontroladora mediante el envío de comandos con información asociada, resultantes de las acciones sobre los elementos del GUI QT. Se propone realizar un control similar de los servomotores del Skybot. Para ello usaremos el mismo sistema de comandos USB que empleamos el año pasado en Sistemas Empotrados empleando el interfaz QT “TwoServosGUI”, de la carpeta Ejercicio6.

- Analiza en primer lugar el código y estructura de la aplicación QT *TwoServosGUI* para averiguar que tipo de comandos es capaz de enviar hacia la placa TIVA, y cual es la información asociada.

- Adapta la aplicación TIVA de ejemplo (control de LEDs) para poder controlar la velocidad de los servos mediante los dos *sliders* y el botón del interfaz QT. Recuerdo de nuevo que los motores están colocados “en oposición”, por lo que un valor en uno tiene efecto en el otro sentido en el motor opuesto.

Referencias

[SISTEMP]J.M. Cano, E. González, I. Herrero, Apuntes de la asignatura "Sistemas Empotrados", 2014
[TIVAxxDRLIB]Texas Instrument, TIVAWare Peripheral Driver Library Users Guide.v2.1.4.178, 2017
[TM4CDS]Texas Instrument, Tiva™TM4C123GH6PM Microcontroller Data Sheet, 2014
[TM4CES]Texas Instrument, Tiva™ C SeriesTM4C123xMicrocontrollersSiliconRevisions6 and 7 Silicon errata, 2016
[TIVAUG]Texas Instrument, Tiva™C SeriesTM4C123GLaunchPadEvaluationBoard Users Guide, 2014