Control de servomotores mediante un microcontrolador ARM

David Martínez Piqué

Grado en Ingeniería Electrónica Industrial y Automática

Trabajo dirigido por: Jordi Palacín, Departamento de Informática e Ingeniería Industrial
Universitat de Lleida, Escuela Politécnica C/ Jaume II, 69, Campus Cappont, Lleida 25001
Teléfono: 676161027; e-mail: david_martinez_pique@hotmail.com

Resumen

En este trabajo se ha realizado el control de servomotores de modelismo y digitales mediante el uso del microcontrolador STM32F407VGT6 de la compañía *ST Microelectronics* basado en la arquitectura ARM. Los servomotores utilizados son el *Futaba* S3003 de la compañía *Futaba Corporation* y los servomotores digitales *Dynamixel* AX-12A y XL-320 de la compañía ROBOTIS. Se ha programado un conjunto de librerías que han permitido controlar simultáneamente hasta 32 servomotores *Futaba* S3003, hasta 254 servomotores AX-12A y hasta 253 servomotores XL-320.

1. Introducción

La utilización de servomotores de modelismo y kits robóticos permite desarrollar acciones y movimientos mecánicos de una forma muy sencilla. El objetivo de este trabajo es aprovechar las ventajas que ofrece un microcontrolador ARM para realizar el control de todos estos tipos de servomotores.

2. Servomotores

2.1. Servomotores de modelismo *Futaba* S3003

Para poder controlar los servos de modelismo se debe generar una señal PWM de una frecuencia de 50Hz cuya duración de pulso determina directamente la posición del servo. El microcontrolador utilizado tiene la capacidad de generar señales PWM a nivel de hardware mediante la configuración de *timers* internos. Se ha realizado una selección de los *timers* internos con capacidad de generar la señal PWM necesaria y posteriormente se han configurado el *Prescaler*, el *Clock_Division* y el *PWM_Period* para generar la señal digital de control dependiendo de si el *timer* utiliza el bus APB1 (*clock* de 84 MHz) o el bus APB2 (*clock* de 168 MHz) (Tab. 1).

A continuación, se han comprobado todos los pines disponibles del microcontrolador que pueden generar dicha señal PWM y se han seleccionado de forma que ningún pin se solape con otras funciones del microcontrolador. Se ha logrado configurar 12 *timers* con un total de 32 canales para generar señales PWM. La Tab. 2 muestra la numeración de los servos con su respectivo *timer*, canal y pin.

TIMXCLK	APB1 – 84 MHz	APB2 – 168 MHz
Clock_Division	1	1
Prescaler	199	199
Timer tick frequency (KHz)	420	840
PWM_FREQ (Hz)	50	50
PWM_PERIOD (ticks)	8399	16799
Miliseconds for 0º	0.48ms (2.4%)	0.48ms (2.4%)
Miliseconds for 180º	2.28ms (11.4%)	2.28ms (11.4%)

Tab 1. Parámetros de configuración de los timers

SERVO	TIMx CHx	PIN	SERVO	TIMx CHx	PIN	SERVO	TIMx CHx	PIN
1	TIM1 CH1	PE9	12	TIM3 CH4	PB1	23	TIM8 CH3	PC8
2	TIM1 CH2	PE11	13	TIM4 CH1	PB6	24	TIM8 CH4	PC9
3	TIM1 CH3	PE13	14	TIM4 CH2	PB7	25	TIM9 CH1	PE5
4	TIM1 CH4	PE14	15	TIM4 CH3	PD14	26	TIM9 CH2	PE6
5	TIM2 CH1	PA15	16	TIM4 CH4	PD15	27	TIM10 CH1	PB8
6	TIM2 CH2	PB3	17	TIM5 CH1	PA0	28	TIM11 CH1	PB9
7	TIM2 CH3	PB10	18	TIM5 CH2	PA1	29	TIM12 CH1	PB14
8	TIM2 CH4	PB11	19	TIM5 CH3	PA2	30	TIM12 CH2	PB15
9	TIM3 CH1	PB4	20	TIM5 CH4	PA3	31	TIM13 CH1	PA6
10	TIM3 CH2	PB5	21	TIM8 CH1	PC6	32	TIM14 CH1	PA7
11	TIM3 CH3	PB0	22	TIM8 CH2	PC7			

Tab 2. Numeración de los servos: timer, canal y pin.

Finalmente, se ha implementado la librería *pwm.h* que recoge todas las funciones para la inicialización de los pines, de los *timers*, la configuración del PWM y el control general de los servos *Futaba* S3003. Entre otras se han desarrollado las siguientes funciones:

- **void PWM_INIT_ALL(void):** Sirve para inicializar los 32 pines, los 12 *timers* y las 32 señales PWM.
- **void PWM_GPIO_INIT(**Tim_TypeDef Tim**):** Sirve para inicializar el pin que corresponde al número de servo indicado por parámetro.
- **void PWM_INIT(**Tim_TypeDef Tim**):** Sirve para inicializar el *timer* y la señal PWM que corresponde al número de servo indicado por parámetro.
- uint8_t **PWM_SERVOMOTOR(**Tim_TypeDef Tim, uint16_t degree**):** Sirve para posicionar el número de servo introducido a los grados indicados.
- uint8_t **PWM_OFF(**Tim_TypeDef Tim**):** Sirve para desactivar la señal PWM del número de servo indicado por parámetro.

• uint8_t **PWM_ON(**Tim_TypeDef Tim**):** Sirve para activar la señal PWM del número de servo indicado por parámetro.

2.2. Servomotores digitales AX-12A y XL-320

El control de los servos digitales AX-12A y XL-320 requiere enviar una secuencia de datos usando una comunicación *Half-Duplex UART*, lo que permite realizar operaciones (escrituras) o consultas (lecturas) sobre los registros de los servos que determinan su funcionamiento. En los servos digitales *Dynamixel* AX-12A esta secuencia de datos es controlada por el protocolo de comunicación 1.0, en el cual, se distingue dos tipos de paquetes de datos.

En primer lugar, el denominado *Instruction packet*, un paquete de información que es enviado desde el microcontrolador hacia los servomotores y que se constituye de los datos mostrados en la Tab. 3.

En segundo lugar, el denominado *Status packet*, un paquete de información que es enviado desde el servo hacia el microcontrolador y que se constituye de los datos mostrados en la Tab. 4, que son similares a los anteriores.

0xFF	0xFF	ID	LENGTH	INSTRUCTION	PARAMETER 1..	PARAMETER N	CHECKSUM

Tab 3. Instruction packet del Dynamixel AX-12A

0xFF	0xFF	ID	LENGTH	ERROR	PARAMETER 1..	PARAMETER N	CHECKSUM

Tab 4. Status packet del Dynamixel AX-12A

0xFF: Los dos bytes 0xFF indican el inicio de un paquete.

Id: Es el identificador del servo.

Length: Es la longitud del paquete. Se calcula como: número de parámetros + 2.

Instruction: Es la instrucción a realizar (Read, Write, Reset, etc).

Error: Representa los errores durante la comunicación.

Parameter 1...N: Es la información adicional necesaria que define el parámetro que se está enviando.

Checksum: Es el cálculo para comprobar que el *Instruction packet* es correcto. Se calcula como: Checksum = ~ (Id + Length + Instruction + Parameter 1...N).

En los servos digitales *Dynamixel* XL-320 esta secuencia de datos es controlada por el protocolo de comunicación 2.0 que es similar al descrito anteriormente aunque permite desarrollar más acciones de control.

Finalmente, todo este proceso de envío y recepción de secuencias de datos se ha llevado a cabo programando unas librerías denominadas ax12a.*h y xl320.h* donde se recogen todas las funciones para la inicialización de los pines, la configuración del UART en modo *Half-Duplex*, la generación de las interrupciones para adquirir los datos y la programación de los protocolos de comunicación para poder enviar y recibir los paquetes correctamente. A continuación se definen las funciones básicas y su utilidad:

• **void GPIO_Config(void):** Sirve para inicializar los pines TX y RX de la comunicación *Half-Duplex UART*.

• **void Init_USART(void):** Sirve para inicializar la comunicación *Half-Duplex UART* y configurar la interrupción en la línea RX para recibir los datos.

• **void TXSend(**uint8_t *InstructionPacket, uint8_t TXNumBytes**):** Sirve para enviar el paquete *Instruction packet* byte por byte mediante la comunicación UART.

• **void USARTx_IRQHandler(void):** Esta función se ejecuta cada vez que salta la interrupción cuando se recibe un byte por la línea RX. Sirve para proceder a tratar el dato recibido y ser guardado en un buffer.

• **void RX_Receive(void):** Sirve para analizar los datos recibidos y proceder a descartarlos o guardarlos según convenga. La función indicará que una nueva secuencia de datos está disponible para ser leída.

• uint8_t **Check_RXFinish(void):** Esta función comprueba byte por byte que la secuencia de datos recibidos sean los del *Status packet* para proceder a guardarlos en variables para su posterior lectura.

3. Resultados

Se ha realizado una serie de pruebas para verificar el control de los servomotores. Se han programado unos comandos para poder interactuar con hasta 32 servomotores analógicos, controlando individualmente el ángulo y estado de activación.

Igualmente, se ha realizado una serie de pruebas para verificar el control de los servomotores digitales directamente sobre un kit de robot humanoide que utiliza los servomotores digitales *Dynamixel* AX-12A. Este test ha permitido comprobar experimentalmente que utilizando las funciones programadas en el microcontrolador es capaz de controlar simultáneamente todos los servos del robot humanoide. La Fig. 2 muestra una secuencia de movimientos programada en el robot.

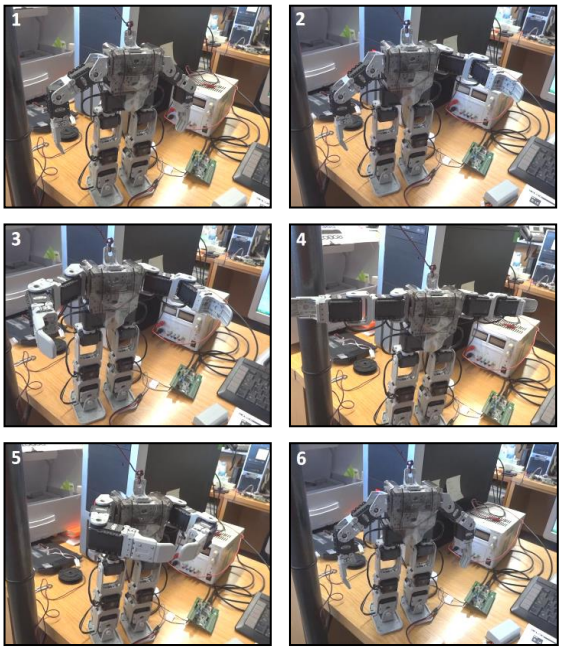

Fig. 2. Secuencia de movimientos de los servos AX-12A mediante el microcontrolador ARM

4. Conclusiones

Se ha implementado un sistema de control de hasta 32 servos analógicos y hasta 254 servos digitales mediante un microcontrolador ARM tipo STM32F407VGT6. El sistema se ha probado en un kit de robot humanoide.

Video en: *http://robotica.udl.cat/demos/humanoid.mp4*