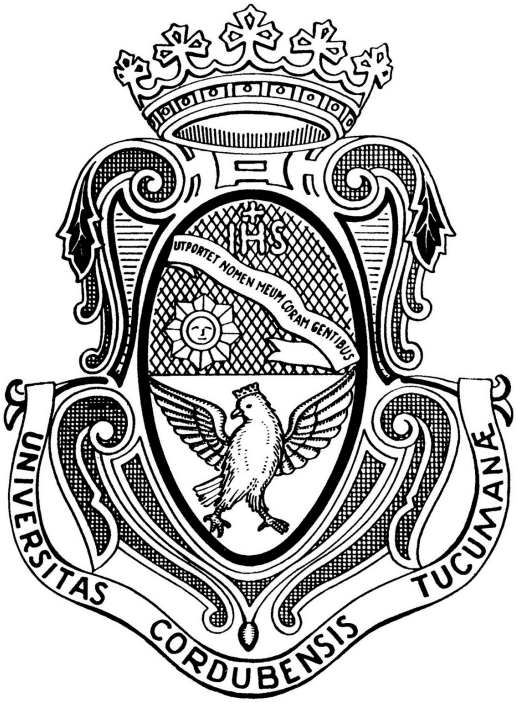
Trabajo práctico Integrador

Electrónica digital III

Brazo Robótico



Grupo 1

Integrantes

- Gina Commisso
- Ignacio Borgatello
- Ignacio Javier Goity

Profesor

- Ayarde, Martín

1. Cálculos implementados

a. Motor paso a paso 28byj-48

i. Especificaciones del motor

El motor 28BYJ-48 es un motor paso a paso unipolar tiene un ángulo de paso de 5,625 grados a medios pasos. A paso completo, tendremos que tener en cuenta que la caja reductora reduce la velocidad del motor en una relación de 1:64. Es decir, el motor interno gira a 2048 pasos por vuelta en paso completo o 0.175° por paso.

Modelo: 28BYJ-48
Tipo de motor: unipolar
Número de conductores: 5
Fases: 4
Resistencia del devanado: 50 Ω
Tensión de alimentación: 5-12 VDC
Potencia: 20 W
Reducción de velocidad: 1:64 (por la caja reductora)
Consumo de corriente: 240 mA
Torque: 0.034 Newton/metro o 0.34 kg/cm

ii. Cálculo de giro en grados

Para evitar el uso de variables tipo *float* en nuestro código, definimos que el grado de paso de ambas articulaciones sea:

$$GRADOS\_PASO\_MOTOR \ = \ 7 \ = \ 0,7 \ * \ 10$$

Por otro lado, para lograr que ambos motores giren la cantidad adecuada de pasos, implementamos la siguiente fórmula:

$$pasos \ = \ (\frac{grados_{ingresados} * 10}{GRADOS\_PASO\_MOTOR}) \ * \ 4$$

dónde *grados ingresados* son la cantidad de grados que el usuario ingresa por teclado, y 4 representa las fases o bobinas internas del motor.

iii. Límite de giro

Debido a que el motor interno gira a 2048 pasos por vuelta en paso completo o 0.175° por paso, el límite que implementamos requiere que la cantidad de grados máximos en MODO_MANUAL no supere los 2048 pasos.

En MODO AUTOMÁTICO y MODO GUARDAR, simplemente corroboramos que el valor ingresado por usuario no sea mayor a 360°

b. Delay anti rebote

Según el fabricante del teclado matricial 4x4, el tiempo de rebote de los pulsadores implementados en este componente es ≤5 ms. Mediante un retardo por software, realizamos este para que sea de al menos 50 ms.

El anti rebote fue realizado con un ciclo for que va hasta 500000. La razón de esto es que el *for* requiere 11 ciclos de instrucción. por lo tanto si el clock del sistema es de 100Mhz:

$$500\,000 \ * \ 11 \ * \ \frac{1}{100e6} = \ 55\,ms$$

c. Timer 1

Utilizamos este periférico para realizar el escaneo del teclado matricial. Como necesitamos ir rotando 1s y 0s, mediante el M1.0 y un valor determinado en el registro match rotamos el valor alto de las filas.

$$tiempo \ = \ PR \ * \ MR \ = \ 1000us \ * \ 10 \ = \ 10\,ms,$$

d. UART

Para la comunicación serie entre el LPC1769 y la PC, utilizamos el periférico UART2 y el *CP2102 Classic USB to UART Bridge* para establecer la comunicación. Para esto, fue necesario establecer una velocidad de $\frac{bits}{s}$.

Debido a que implementamos este periférico en conjunto con el GPDMA, nos bastó con utilizar la velocidad por defecto de $9600\ baudios$, debido a que no dependeremos del *core* para esta tarea.

Por otro lado, configuramos una transmisión de 8 bits, con 1 bit de parada y sin bit de paridad.

e. DMA

Para el controlador DMA, configuramos transmisión de memoria a periférico mediante la utilización del canal 0, que si bien es el de mayor prioridad, en nuestro caso, no tendremos otro tipo de periférico involucrado más que el UART. El tamaño de la transferencia, y la región de memoria a transmitir varía según el mensaje que queremos mostrar en pantalla.

Siguiendo las instrucciones del manual, cada vez que necesitamos enviar un nuevo mensaje, configuramos nuevamente el DMA.

f. Timer 0, potenciometro y entrada analógica

i. ADC y Timer 0

Mediante un potenciómetro controlamos la velocidad del movimiento del brazo robótico. Para esto, nos fue necesario utilizar una entrada analógica (Pin 0.23) y el M0.1 para modificar indirectamente la frecuencia de muestreo, debido a que no requerimos que funcione a su máxima capacidad. En principio, tenemos los siguientes valores:

$$ft\ =\ 13\ MHz\ =\ 100MHz\ /\ 8$$

$$fm\ =\ 13\ MHz\ /\ 65\ ciclos\ =\ 200\ KHz$$

Al configurar el Timer 0 para que interrumpa cada 25 ms, y mediante el uso de la función *Toggle* del pin M0.1, hacemos que el ADC funcione cada flanco de bajada de dicho pin. Por ende, obtenemos una frecuencia de muestreo de:

$$Ts = 50\,ms$$

$$fm = \frac{1}{2 * 25ms} = 20\,Hz$$

ii. Potenciómetro

Por último, utilizamos un potenciómetro de 10KΩ debido a que el capacitor del Sample and Hold requiere de cierto tiempo para llegar a un nivel que sea representativo de nuestro valor a muestrear. Por ende, el fabricante del LPC1769 detalla que hasta este valor, no se presentará ningún problema. Por encima de este límite, corremos riesgo de mostrar valores incorrectos.

Table 19. ADC characteristics (full resolution)

V_{DDA} = 2.5 V to 3.6 V; T_{amb} = −40 °C to +85 °C unless otherwise specified; ADC frequency 13 MHz; 12-bit resolution.[1]

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IA}	analog input voltage			0	-	V_{DDA}	V
C_{ia}	analog input capacitance			-	-	15	pF
E_D	differential linearity error		[2][3]	-	-	±1	LSB
$E_{L(adj)}$	integral non-linearity		[4]	-	-	±3	LSB
E_O	offset error		[5][6]	-	-	±2	LSB
E_G	gain error		[7]	-	-	0.5	%
E_T	absolute error		[8]	-	-	4	LSB
R_{vsi}	voltage source interface resistance		[9]	-	-	7.5	kΩ
$f_{clk(ADC)}$	ADC clock frequency			-	-	13	MHz
$f_{c(ADC)}$	ADC conversion frequency		[10]	-	-	200	kHz

iii. Velocidad de giro

Dentro de la variable velocidad guardamos el valor obtenido en una conversión del ADC. Luego, convertimos esa variable a un valor representativo en ticks que ralentice o acelere el movimiento del motor. Los límites de frecuencia que adoptamos, en concordancia con lo que dice el fabricante del motor, son 50 Hz a 300 Hz.

$$delay\,[Hz] = 0.06 * velocidad + 50\,Hz$$

$$\frac{SystemCoreClock}{delay\ [Hz]} = delay\ [Ticks]$$

Debido a que un ciclo *for* tarda 11 ciclos de instrucción, debemos retardar por software $\frac{delay[Ticks]}{11}$.

g. Interrupciones y prioridades

Implementamos el siguiente orden de prioridades de interrupciones:

1. EINT3_IRQHandler()
2. TIMER1_IRQHandler()
3. ADC_IRQHandler()

En principio, necesitamos que el teclado posea la mayor prioridad ya que es la base del funcionamiento de brazo robótico. Desde allí controlamos modos, instrucciones y ejecución de los movimientos. Luego, el Timer 1, como mencionamos, requiere de una prioridad intermedia ya que funciona en conjunto con el teclado. Sin embargo, en la subrutina de interrupción del teclado (EINT3), inmediatamente deshabilitamos la cuenta del Timer para hacer un catch de la tecla presionada. Por último, el ADC es quien menos prioridad necesita ya que su función no es de relevante frente a las otras dos al momento del funcionamiento.