Bogotá 14 de Septiembre del 2012

TEATRO DE L A MEMORIA EN MUNDOS VIRTUALES
UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ

ENTREGABLE #5
César Augusto Pantoja
Estudiante auxiliar ingeniería electrónica

PROGRAMACIÓN Y PUESTA EN DEL SISTEMA DE LECTURA DE ENCODER PARA LA REALIMENTACIÓN DE LA POSICION ENTRE EL PODIO REAL Y EL VIRTUAL.

SINCRONIZACIÓN ENTRE EL PODIO VIRTUAL Y EL REAL

El aplicativo en Java 3D cuenta con un podio y un escenario virtual, los cuales se mueven en las coordenadas (x,y,z), dependiendo de las posiciones del podio real; para esto cada articulación cuenta con un encoder incremental adaptado al eje de cada motor, y por medio de la tarjeta lectora de encoder se puede determinar la posición y dirección del movimiento. El sistema sigue el diagrama de bloques mostrado en la figura 12.

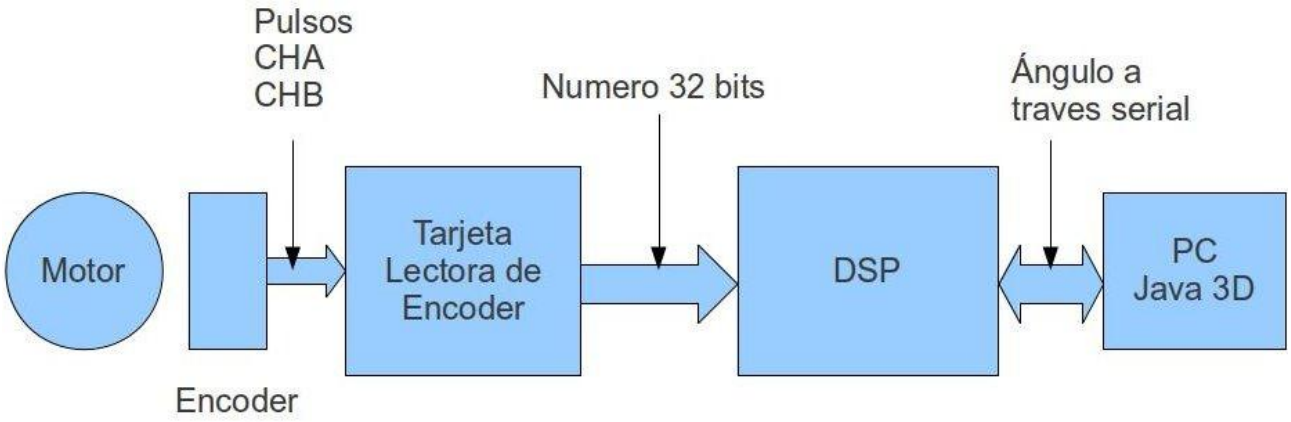


Figura 12.

HARDWARE

ENCODER

El encoder utilizado es marca AVAGO referencia HEDS-5500 el cual consta de dos canales CHA y CHB y 500 CPR (pulsos por revolución) Figura 13. Cuya configuración de pines aparece en la figura 14.

Figura 13.

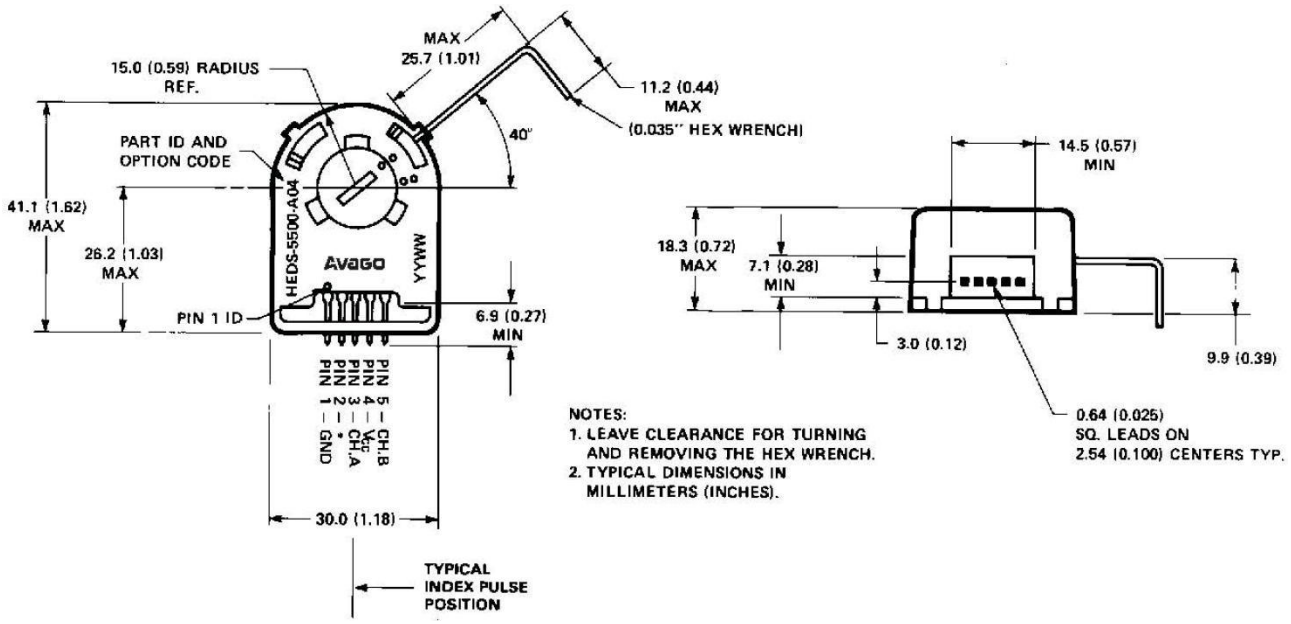


Figura 14.

TARJETA LECTORA DE ENCODERS

Consta principalmente de dos circuitos integrados HCTL 2032, también se encarga de suplir el voltaje de alimentación de los encoder y entregar el valor del conteo equivalente a la cantidad de giro de cada articulación.

El circuito integrado HCTL 2032 (Figura 15), es un contador/Decodificador de cuadratura, el cual contiene la lógica necesaria para decodificar las señales provenientes del encoder y así incrementar/decrementar un contador de 32 bits.

Cada HCTL 2032 contiene además dos contadores de 32 bits y dos canales; uno por cada encoder y el valor correspondiente al giro del encoder es entregado al DSP por medio de un bus de ocho bits, por lo cual es necesario del manejo de una secuencia lógica para obtener el valor completo de 32 bits.

El HCTL 2032 puede recibir las señales CHA, CHB, CHI, de un encoder estándar, pero el encoder utilizado en el podio solo contiene las señales CHA, CHB, así que para que funcionara fue necesario conectar GND en los pines 17 y 19 (CHIx y CHIy) por medio de un puente soldado encima de la PCB.

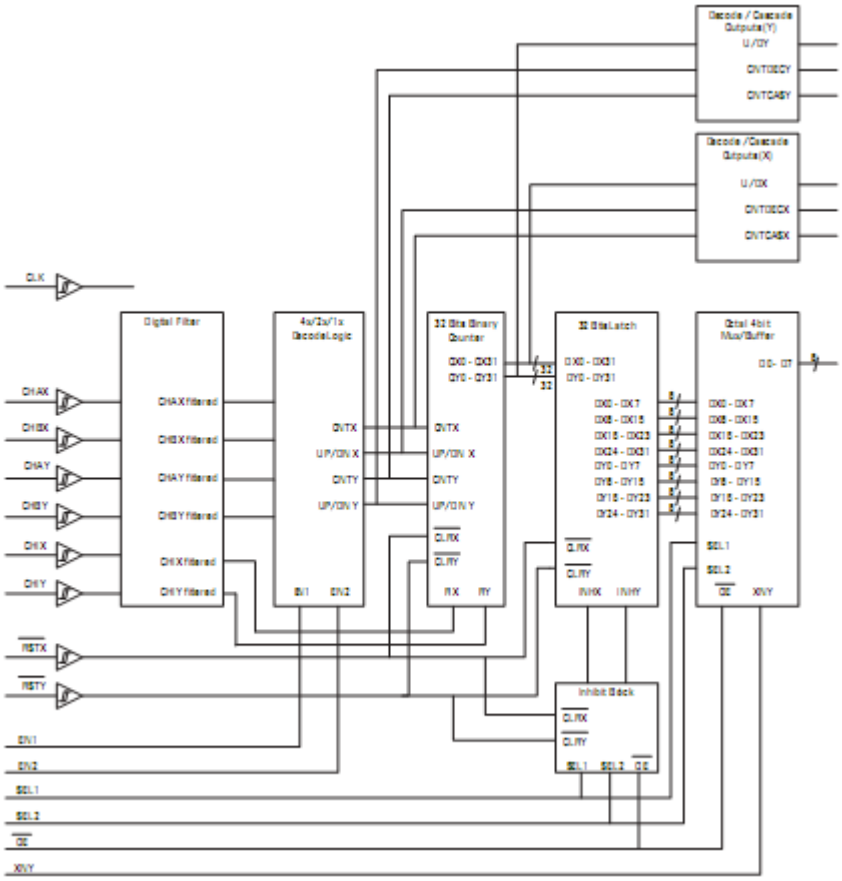


Figura 15.

SOFTWARE

La aplicación de control en el DSP, se encarga de lo siguiente:

1. *Crear la secuencia lógica necesaria para la lectura del contador obteniendo un byte por vez, basado en el esquema de la figura 16.*
2. *Guardar cada byte en un vector binario y reconstruir el número completo*
3. *Convertir el valor del contador de un vector binario a un entero.*
4. *Hacer las operaciones de conversión a grados de giro*
5. *Envío del dato por el puerto serial hacia la aplicación de java3d del usuario.*

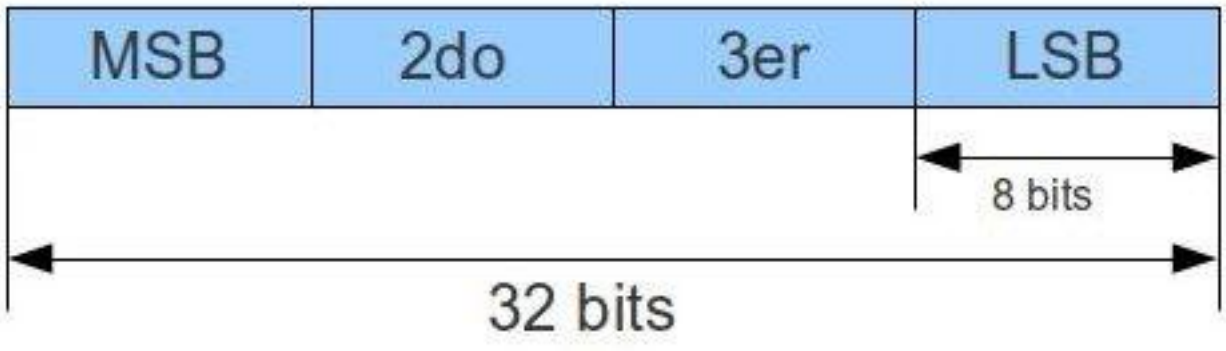


Figura 16.

ALGORITMO DE LECTURA DE LOS CONTADORES

1. Se deshabilita el reset de cada contador RSTX y RSTY (pines 12 y 11 respectivamente).
Con un “1” se se borran los contadores y con un “0” se deshabilita el reset.

2. Se selecciona el eje que se quiere leer mediante la entrada XY (Pin32 del circuito HCTL2032).
En donde con un “0” elige el eje X y un “1” el eje Y.

3. Se habilita la salida OE (output enable pin 7) para permitir la lectura segura del buffer.
Con un “1” se deshabilita la salida y con un “0” se habilita las salida.
Valor entregado ”0”

4. Se selecciona el byte a leer mediante los pines SEL1 (pin 6) y SEL2 (pin 26)
- SEL1=1,SEL2=0.
Se lee el byte LSB
LSB = DATO

- SEL1=0,SEL2=0.
Se lee el 3º byte..
THIRD = DATO

- SEL1=1,SEL2=1.
Se lee el 2º byte..
SECOND =DATO

- SEL1=0,SEL2=1.
MSB = DATO

4. Se DEShabilita la salida OE (output enable pin 7) paso necesario para hacer cambio de X/Y
Con un “1” se deshabilita la salida y con un “0” se habilita las salida.
Valor entregado “1”

ALGORITMO DE OBTENCIÓN DE LOS ÁNGULOS

Después de finalizada la obtención de los datos, obtenemos 4 bytes, los cuales se deben convertir en enteros así:

For i =0:7
ValorDecimal = DATO[i] *(2^i) + ValorDecimal;

Cada contador entregó un número de 32 bits de los cuales no se requieren sino 16 bits para obtener el valor máximo de giro así:

El encoder del motor número uno, tiene acoplado un reductor con una relación de transmisión de 24:1 osea que por cada 24 vueltas del motor, el eje de salida da un giro, asi que:
500(CPR) * 24 = 12000 pulsosPorRevolucion. #motor 1 reductor 24:1
500(CPR) * 30 = 15000 pulsosPorRevolucion. #motor 2 reductor 30:1
500(CPR) * 60 = 30000 pulsosPorRevolucion. #motor 3 reductor 60:1
500(CPR) * 6.3 = 3150 pulsosPorRevolucion. #motor 4 reductor 6.3:1

El valor máximo es de 30000 y se requieren solo 15 bits, así que se divide en valor de 32 bits en

parte baja y parte alta
ContadorL = THIRD * 256 + LSB;
ContadorH = MSB * 256 + SECOND;

La parte baja se utiliza obtener el ángulo de giro en CW (sentido horario) y la parte alta para obtener el ángulo en sentido (CCW) a partir de un punto Cero.

Luego cada pulso tiene un valor de:
360/12000 = 0.03 Grados por cada pulso motor 1
360/15000 = 0.024 Grados por cada pulso motor 2
360/30000 = 0.012 Grados por cada pulso motor 3
360/3150 = 0.114 Grados por cada pulso motor 4

Así que finalmente el giro en grados se obtiene:

```
if (contadorL <= 6000) {
   angulo1 = contadorL * 0.03;
   }
   else {
   angulo1 = (contador1 - 12000) * 0.03;
   }
if (contadorL <= 7500) {
   angulo2 = contadorL * 0.024;
   }
   else {
   angulo2 = (contadorH - 15000) * 0.024;
   }

   // Calculo del Angulo 3
   if (contadorL <= 15000) {
   angulo3 = contadorL * 0.012;
   }
   else {
   angulo3 = (contadorH - 30000) * 0.012;
   }

   // Calculo del Angulo 4
   if (contadorL <= 1575) {
   angulo4 = contadorL * 0.114;
   }
   else {
   angulo4 = (contadorH - 3150) * 0.114;
   }
```

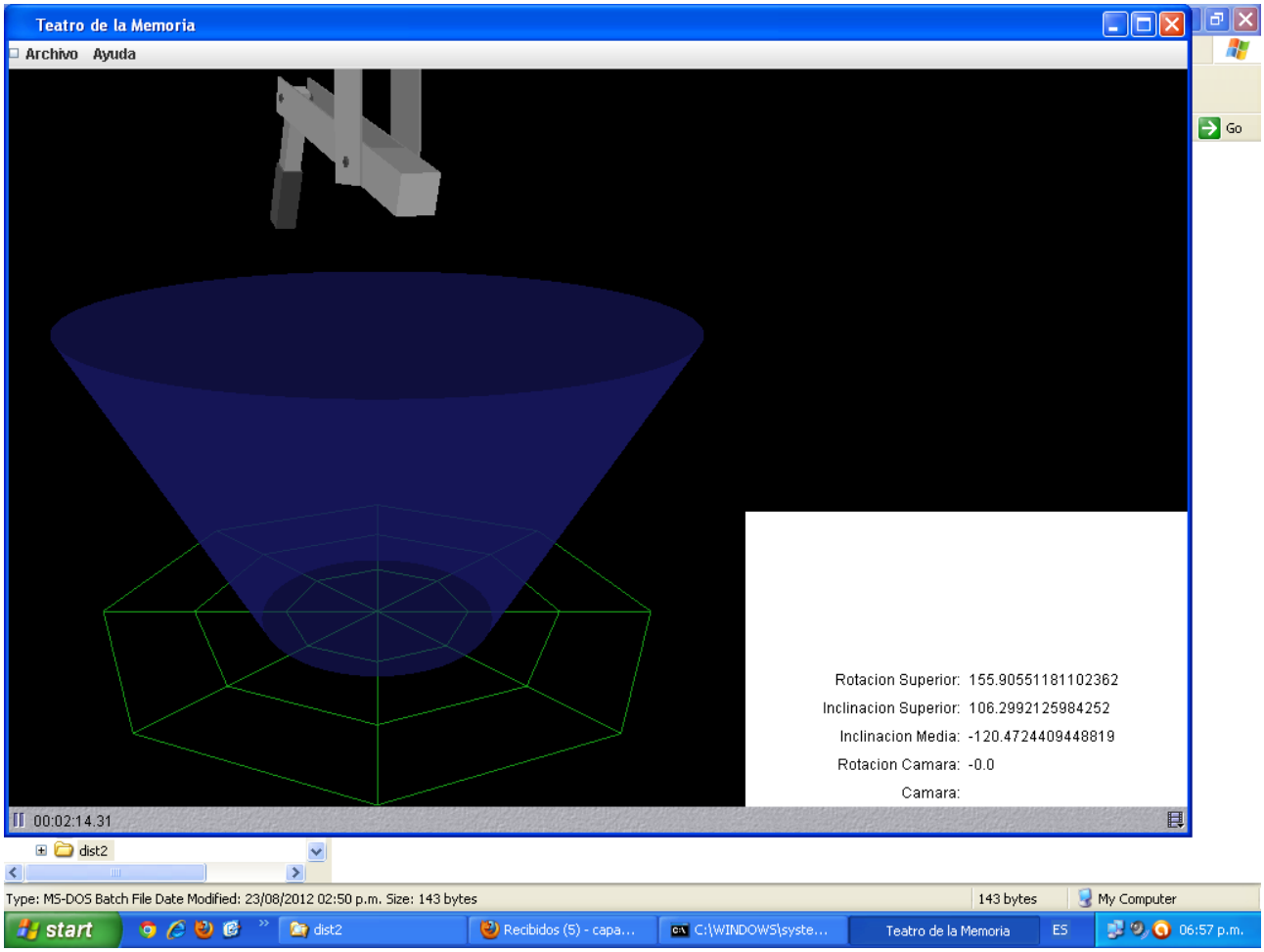
Finalmente el dato es “empaquetado” para ser enviado por el puerto serial

```
   angulo1 = angulo1 * 127/180;
   angulo2 = angulo2 * 127/180;
   angulo3 = angulo3 * 127/180;
   angulo4 = angulo4 * 127/180;
```

La aplicación se construyó en simulink (Debido al tamaño se adjunta archivo web)
Despues de creada la aplicación en Simulink, se crean los archivos en .c para su posterior compilación en Code Composer Studio, pero los bloques Embeddeb Matlab function 1 y Embeddeb Matlab function 2, no crearon un código completo, es decir matlab omitió muchas líneas, esto quizá debido a algún parámetro de configuración que optimiza de forma no deseada para este caso, se probaron muchas formas de configuración y no se obtuvo el código deseado, por lo que fue

necesario modificar manualmente el código generado en Code Composer Studio de la siguiente forma (también se entrega en achivo adjunto):
Matlab genera 16 archios de los cuales nos interesan 3 archivos llamados:
RLV3DS_TME_PODIO_main.c
RLV3DS_TME_PODIO.c
RLV3DS_TME_PODIO.h

Se modificaron RLV3DS_TME_PODIO.c y RLV3DS_TME_PODIO.h



Aspecto final del aplicativo 3D con los ángulos

CONCLUSIONES
Para hacer funcionar la tarjeta lectora de encoder se tuvo que hacer un puente entre GND y los pines 17 y 19 (CHIx y CHIy).
Se intentaron muchas configuraciones con el fin de corregir el código generado por Matlab
Matlab omitió líneas de código en simulink, en los bloques Embeddeb Matlab function 1 y Embeddeb Matlab function 2, por lo que fue necesario modificar manualmente en lenguaje c los archivos

RLV3DS_TME_PODIO.c
RLV3DS_TME_PODIO.h