

Laboratorio No. 10

Diseño programas en lenguaje Ensamblador para el control de circuitos.

I. Competencias a desarrollar

Diseñar un programa básico en lenguaje ensamblador del procesador ARM que implemente la secuencia asignada.

II. Instrucciones

Esta actividad se desarrollará en los grupos de trabajo y los temarios seleccionados para el Proyecto 03.

Por medio del control de puertos GPIO de la Raspberry PI y de programación, implementarán un circuito físico con la secuencia asignada por su catedrático.

III. Temarios

1. Juego: corrimiento de bits activos

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador, muestre (uno a uno) los estados de la secuencia mostrada en la siguiente figura. El cambio de estado se permitirá cada 1.5 segundos (es decir, si el botón se presiona cada 0.1s, solo se permitirá el cambio de estado cada 1.5s).

Al finalizar la secuencia no se permitirá que se genere ningún cambio a los leds cuando se presiona el pulsador. La activación de la salida digital DO de un sensor, permitirá el reinicio de la secuencia al estado inicial.



Figura 1.

2. Juego: corrimiento de bits inactivos

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador, muestre (uno a uno) los estados de la secuencia mostrada en la siguiente figura. El cambio de estado se permitirá cada 0.8 segundos (es decir, si el botón se presiona cada 0.1s, solo se permitirá el cambio de estado cada 0.8s).

Al finalizar la secuencia no se permitirá que se genere ningún cambio a los leds cuando se presiona el pulsador. La activación de la salida digital DO de un sensor, permitirá el reinicio de la secuencia al estado inicial.



Figura 2.

3. Juego: Acumulador de bits

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador, muestre (uno a uno) los estados de la secuencia mostrada en la siguiente figura. El cambio de estado se permitirá cada 2.0 segundos (es decir, si el botón se presiona cada 0.1s, solo se permitirá el cambio de estado cada 2.0s).

Al finalizar la secuencia no se permitirá que se genere ningún cambio a los leds cuando se presiona el pulsador. La activación de la salida digital DO de un sensor, permitirá el reinicio de la secuencia al estado inicial.



Figura 3.

4. Juego: Gusanito

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador, muestre (uno a uno) los estados de la secuencia mostrada en la siguiente figura. El cambio de estado se permitirá cada 1.0 segundos (es decir, si el botón se presiona cada 0.1s, solo se permitirá el cambio de estado cada 1.0s).

Al finalizar la secuencia no se permitirá que se genere ningún cambio a los leds cuando se presiona el pulsador. La activación de la salida digital DO de un sensor, permitirá el reinicio de la secuencia al estado inicial.



Figura 4.

5. Juego: acumulador de segmentos

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador, muestre (uno a uno) los estados de la secuencia mostrada en la siguiente figura. El cambio de estado se permitirá cada 1.0 segundos (es decir, si el botón se presiona cada 0.1s, solo se permitirá el cambio de estado cada 1.0s).

Al finalizar la secuencia no se permitirá que se genere ningún cambio a los leds cuando se presiona el pulsador. La activación de la salida digital DO de un sensor, permitirá el reinicio de la secuencia al estado inicial.



Figura 5.

6. Simon says - Letras

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador muestra una letra generada de forma aleatoria (de las letras en la figura 06). Las letras deberán ser almacenadas en una etiqueta de tipo asciz, ascii, o byte, máximo 15 letras (para esta entrega).

Luego de haber generado y guardado las 15 letras aleatoriamente, usará la activación de la salida digital DO, de un sensor, para reiniciar al estado inicial (borrar las letras almacenadas en la etiqueta).

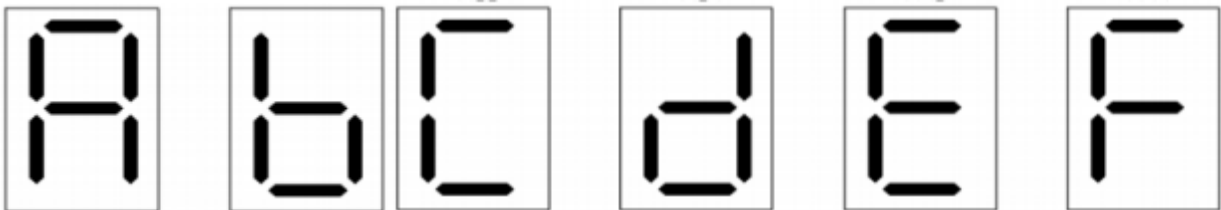


Figura 6.

Se debe imprimir en pantalla el status de la etiqueta para validar que se ingresando las letras y que al final, estas se eliminan.

7. Simon says - números

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador muestra un número generado de forma aleatoria (de los números en la figura 06). Los números deberán ser almacenados en una etiqueta de tipo asciz, ascii, o byte, máximo 15 números (para esta entrega).

Luego de haber generado y guardado los 15 números aleatoriamente, usará la activación de la salida digital DO, de un sensor, para reiniciar al estado inicial (borrar los números almacenados en la etiqueta).



Figura 7.

Se debe imprimir en pantalla el status de la etiqueta para validar que se ingresando los números y que al final, estos se eliminan.

8. Simon says - vocales

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador muestra una vocal generada de forma aleatoria (de las vocales en la figura 06). Las vocales deberán ser almacenadas en una etiqueta de tipo asciz, ascii, o byte, máximo 15 vocales (para esta entrega).

Luego de haber generado y guardado las 15 vocales aleatoriamente, usará la activación de la salida digital DO, de un sensor, para reiniciar al estado inicial (borrar las vocales almacenadas en la etiqueta).

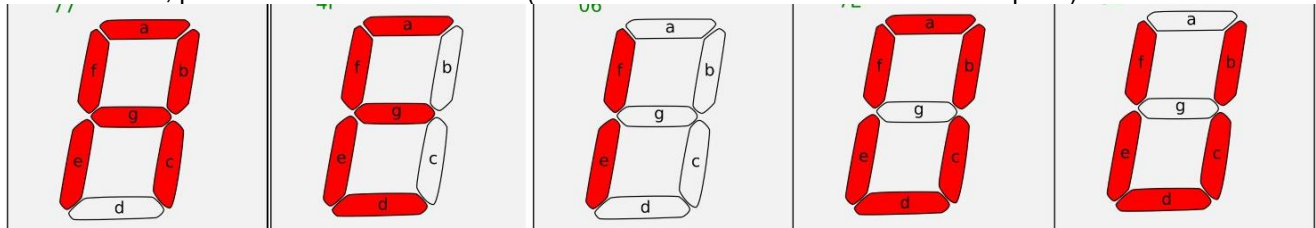


Figura 8.

9. Simon says - colores

Desarrollar un programa en lenguaje Assembler ARM, que al presionar un botón pulsador active leds de forma aleatoria (de los 5 leds mostrados en la figura 06). Los colores del led deberán ser almacenados en una etiqueta de tipo asciz, ascii, o byte, máximo 25 colores (para esta entrega).

Luego de haber generado y guardado los 15 colores aleatoriamente, usará la activación de la salida digital DO, de un sensor, para reiniciar al estado inicial (borrar los colores almacenadas en la etiqueta).

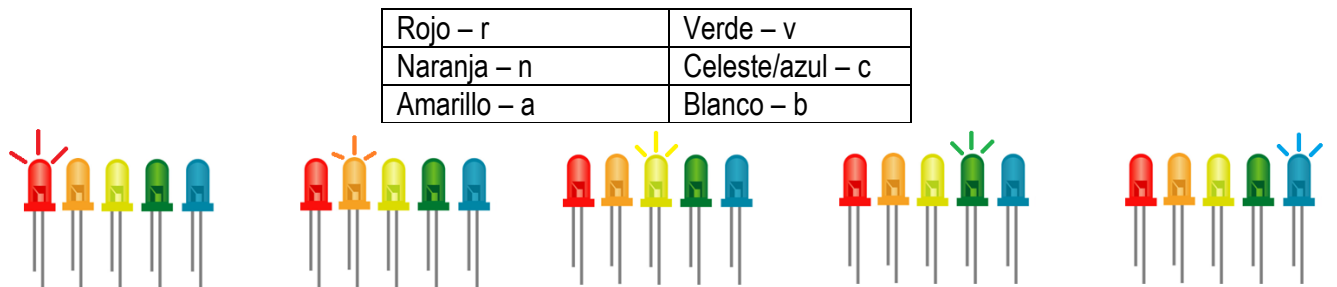


Figura 9.

13 Carrito inteligente seguidor de línea.

Diseñar y programar el funcionamiento de los puertos GPIO de la Raspberry en lenguaje ensamblador ARM, para el control de un carro dirigido mediante un seguidor de línea negra.

Al presionar la tecla “l”, se produce el movimiento de un motor (que produce el giro hacia la izquierda)

Al presionar la tecla “r”, se produce el movimiento de un motor (que produce el giro hacia la derecha)

Al presionar la tecla “s”, se detienen los motores.




Incluir los sensores necesarios para realizar el control del movimiento del carro.



Figura 11.

IV. Evaluación

Todos los integrantes del grupo deben presentar el circuito y programa funcional.

Criterios de evaluación	Nivel 3 Experto 	Nivel 2 Aprendiz 	Nivel 1 Novato 
Funcionamiento del Programa 60%	El programa funciona con todos sus requerimientos: ingreso de datos, despliegue de resultados correctos y salida correcta al sistema operativo 60%	El programa funciona en al menos el 60% de sus requerimientos de funcionamiento. 35%	El programa funciona en menos del 50% de sus requerimientos. 20% o menos
Programación defensiva 10%	Los programas tienen muy buena programación defensiva, proporcionando mensajes oportunos ante situaciones inesperadas. 10%	Los programas tienen programación y proporciona algunos mensajes oportunos ante situaciones inesperadas. 5%	Los programas tienen muy poca o ninguna programación defensiva, y proporciona pocos o ningún mensaje oportuno ante situaciones inesperadas. 0%
Circuito 20%	El circuito es funcional y ordenado 20%		
Documentación 10%	La documentación incluye encabezado y comentarios representativos en los bloques de código más importantes. Los nombres de las variables son significativos. 10%	Falta documentación en el encabezado o en bloques de código Los nombres de las variables son medianamente significativos. 5%	Falta gran parte de la documentación del código Los nombres de las variables no expresan ningún significado. 0%