Proyecto 3 - temario 1

Organización de computadoras y assembler
Sección 30

Ing. Roger Díaz

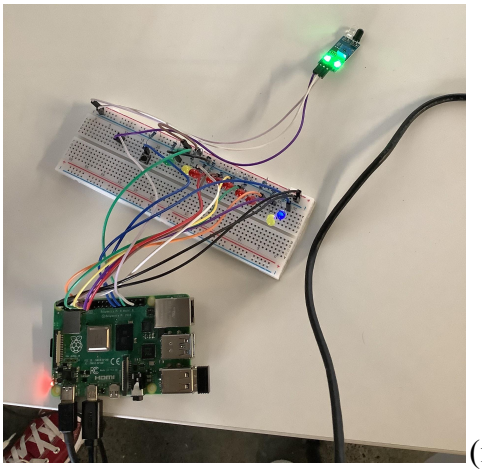
Cindy Mishelle Gualim Pérez 21226
Jose Roberto Rodriguez Reyes 21060

Guatemala 5 de mayo de 2022

Introducción

El desarrollo de este proyecto, se enfoca en el diseño (imagen 1.1) y programa en lenguaje ensamblador ARM , para controlar 8 leds de corrimiento, que tendrán un comportamiento de activación/desactivación , donde adicionalmente se tienen dos leds para identificar los jugadores, que serán controlados al presionar la letra “Y” del teclado y un pulsador conectado a los puertos GPIO para generar un incremento, el jugador que presione para el último estado es el que gana el juego, el diseño del juego cuenta con una señal de reloj para controlar el circuito, con un periodo de 1.5 segundos, donde el programa debe leer si ha sido la tecla o el botón después de los 1.5 segundos después de ser pulsado el anterior.

Adicionalmente en el programa se mostrará un mensaje en pantalla de cuál ha sido el jugador ganador y este activará el led correspondiente al jugador, el reset del juego se dará al presionar la tecla “q” y con un sensor digital que su activación genera el reset del juego al estado inicial.


(imagen 1.1)

Materiales

- Raspberry pi4
- Jumpers macho macho
- Jumpers macho hembra
- 10 LED
- 12 Resistencias de 330 ohms
- Proto Board
- Cable HDMI
- Push botton

Solución

Para la resolución del temario asignado lo primero que se hizo fue inicializar los pines de lectura y escritura, posteriormente para cada led se crearon los pines anteriormente mencionados, para poder cambiar el estado de las leds entre encendido y apagado, luego se creó una función llamada, "esperandoBoton" (imagen 2.1), en la cual utilizando los pines de lectura se detecta si el botón ha sido presionado y si lo ha sido se inicia el programa, también se creó la clase "esperandoTecla" (imagen 2.2), en el cual de manera similar, verifica si ha sido presionada o no la tecla "y", al igual que con el botón, si ha sido presionado se inicia el programa. Posteriormente se realizó la función de "identificarEstado" (imagen 2.3), en el que se compara el pin del jugador, ya sea 1 o 2, utilizando el comando cmp, con los valores del 1 al 8, agregando que se implementó la función "print1" (imagen 2.4), en la cual utilizando los pines de salida creados previamente y la función de identificarEstado , se encendió el led que le toca respectivamente al jugador 1, y se implementó la función "print2" (imagen 2.5), que funciona de exactamente la misma manera, pero para el jugador 2, inalmente se implementó una espera de 1.5 segundos para cada led y haciendo uso de las funciones creadas previamente, se llama a cada led, mientras va avanzando el "juego", hasta que un jugador se encuentre en el último estado y el juego muestre el mensaje en pantalla y encienda el led respectivo al jugador ganador .

```
esperandoBoton:
    ldr r0, =mensajeboton
    bl printf

    mov r0, #21
    bl digitalRead

    cmp r0, #0
    bne print2:
```

(imagen 2.1)

```
esperandoTecla:
    ldr r0, =mensajetecla
    bl printf

    ldr r0, =formato
    ldr r1, =tecla
    bl scanf

    ldr r5, =tecla
    ldrb r5, [r5]

    cmp r5, #0x79
    beq print1
```

(imagen 2.2)

```
identificarEstado:
    // identificar en que estado está el juego
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #1
    beq primeraLed
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #2
    beq segundaLed
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #3
    beq terceraLed
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #4
    beq cuartaLed
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #5
    beq quintaLed
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #6
    beq sextaLed
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #7
    beq septimaLed
    ldr r0, =siguienteEstado
    ldr r0, [r0]
    cmp r0, #8
    beq octavaLed
```

(imagen 2.3)

```
print1:
    ldr r0, =mensajeultimojugador
    bl printf

    ldr r0, =jugador1
    bl printf

    mov r0, #23
    mov r1, #0
    bl digitalWrite

    mov r0, #22 // se prende la luz de jugador 1
    mov r1, #1
    bl digitalWrite

    b identificarEstado
```

(imagen 2.4)

```
print2:
    ldr r0, =mensajeultimojugador
    bl printf

    ldr r0, =jugador2
    bl printf

    mov r0, #23
    mov r1, #1
    bl digitalWrite

    mov r0, #22 // se prende la luz de jugador 2
    mov r1, #0
    bl digitalWrite

    b identificarEstado
```

(imagen 2.5)

Especificación del uso de los registros y puertos GPIO utilizados

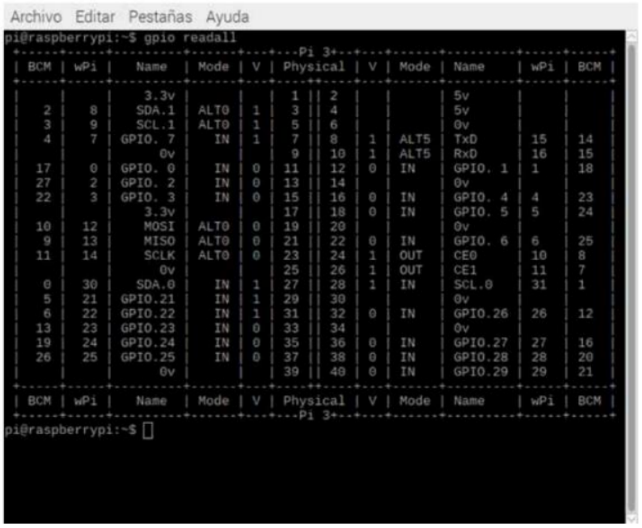
Los GPIOS utilizados son del 0 al 7, 21,25,22 y 23 tal y como se muestra en las imágenes 3.1 y 3.2, con el manejo de comandos que se muestra en la imagen 3.3.

```
myPinmodes:
    // Inicializar los pines de salida
    mov r0, #0 // salidas
    mov r1, #1
    bl pinMode

    mov r0, #1
    mov r1, #1
    bl pinMode

    mov r0, #2
    mov r1, #1
    bl pinMode

    mov r0, #3
    mov r1, #1
    bl pinMode

    mov r0, #4
    mov r1, #1
    bl pinMode

    mov r0, #5
    mov r1, #1
    bl pinMode

    mov r0, #6
    mov r1, #1
    bl pinMode

    mov r0, #7
    mov r1, #1
    bl pinMode

    mov r0, #21 // entradas
    mov r1, #0
    bl pinMode
```

(imagen 3.1)

```
mov r0, #21 // entradas
mov r1, #0
bl pinMode

mov r0, #25
mov r1, #0
bl pinMode

mov r0, #22 // salida jugador 1
mov r1, #1
bl pinMode

mov r0, #23 // salida jugador 2
mov r1, #1
bl pinMode

b esperandoTecla
```

(imagen 3.2)



(imagen 3.3)

Diagrama de flujo del programa hecho en una herramienta de software o algoritmo narrativo

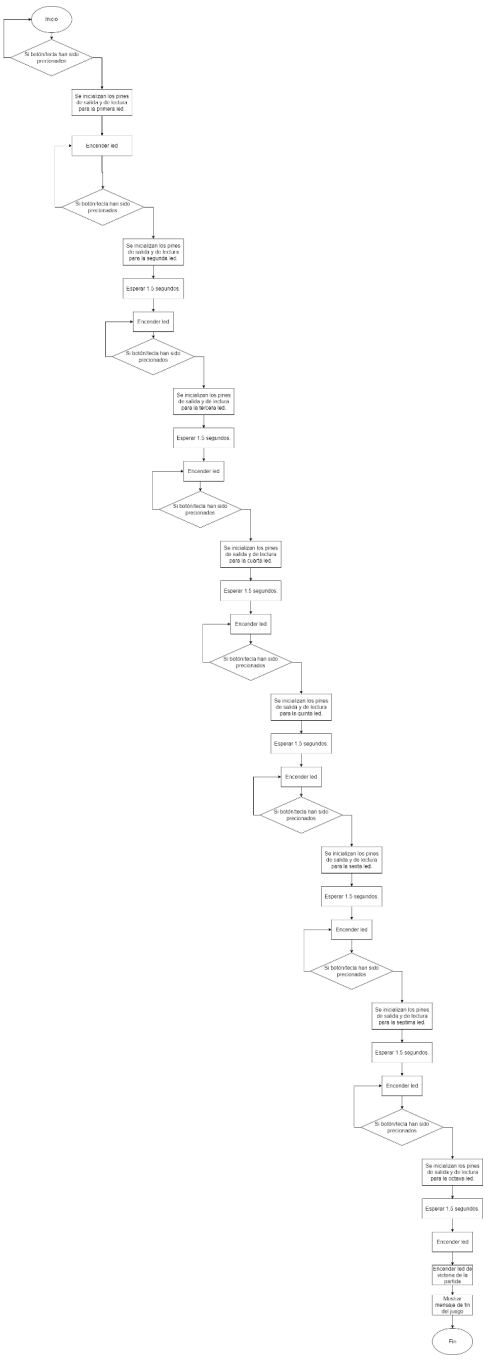
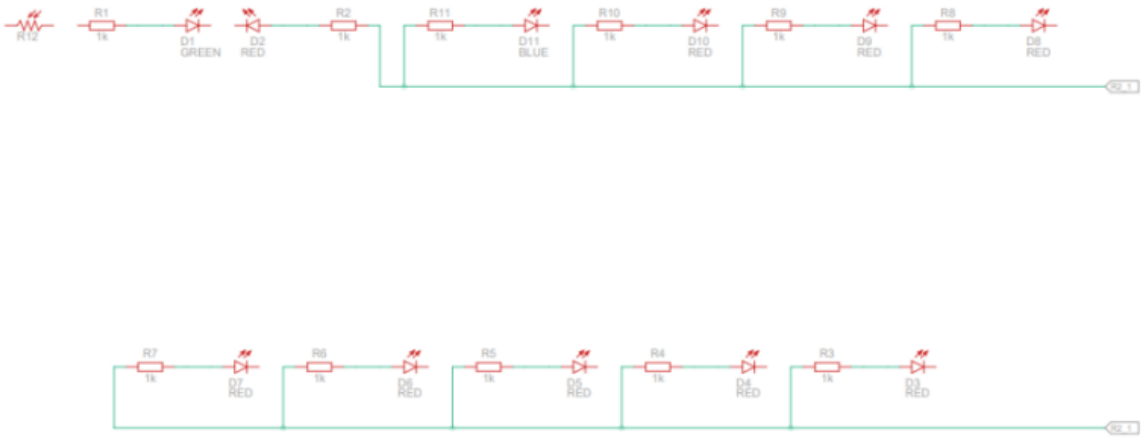


Diagrama del circuito



Conclusiones:

- El uso adecuado de los puertos GPIO es importante para el buen funcionamiento del programa.
- La planeación lógica para la implementación del programa es importante para el orden y buen funcionamiento de este.
- El armado adecuado del circuito es tan importante como el funcionamiento del programa en software.

Bibliografía:

Paul A. Carter. (2007). Lenguaje Ensamblador para PC. 2022, de - Sitio web: http://pacman128.github.io/static/pcasm-book-spanish.pdf

Simon Humphreys. (2022). Assembly Language on the Pi: “Learning how to walk again”. 2022, de HelloWorld Sitio web: https://helloworld.raspberrypi.org/articles/hw16-assembly-language-on-the-pi