REPUBLICA BOLIVARIANA DE VENEZUELA

UNIVERSIDAD NACIONAL EXPERIMENTAL

POLICTÉNICA (UNEXPO)

“VICERRECTORADO LUIS CABELLARO MEJIAS”

PROYECTO

LLEGADA DE LUDO

Docente: Jorge Lara

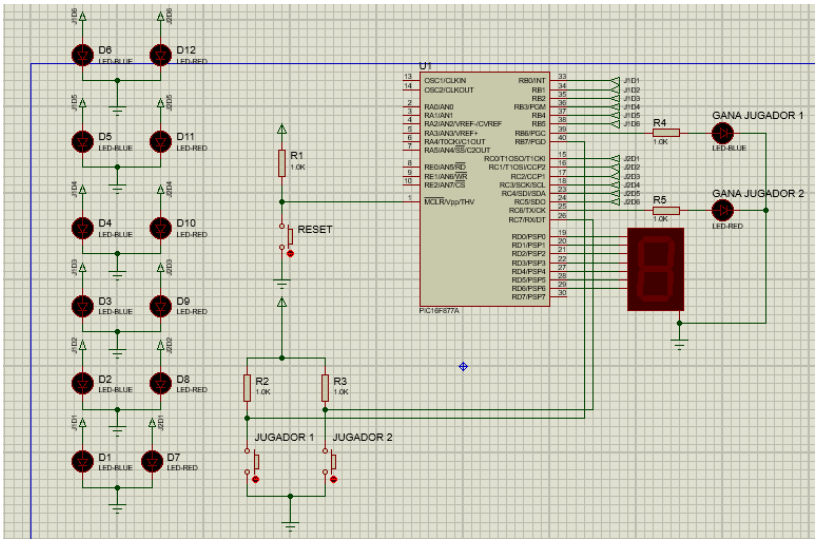
Estudiante: Victor Apolinares

Expediente: 2017203056

Caracas, Marzo del 2021

- Este proyecto trata del funcionamiento del circuito digital y electrónico sobre un juego llamado ludo, el cual solo se trabajará con la parte de llegada que son solo seis casillas y los dos jugadores obviamente empezaran en la casilla 1.

Circuito en Proteus



- Como se visualiza en la imagen anterior se ven dos columnas, cada columna le pertenece a un jugador y cada fila pertenece a las casillas del ludo.
- En el display se visualizará un numero aleatorio del 1 al 6, que simulará un dado virtual.
- Cada jugador tiene un botón que hará la simulación de lanzar el dado.
- Cuando gane un jugador se visualiza un led de ganador del mismo color del led del jugador, en este caso el jugador 1 visualiza el led de color azul y el jugador 2 de color rojo.

Funcionamiento y condiciones del juego:

- Al principio del juego, se tiene que determinar el jugador que comenzara primero jugando, por eso cada jugador tiene que lanzar su dado y el jugador que saque el numero mas alto comienza jugando, si sacan el mismo numero lo tienen que volver a intentar hasta que uno de los dos jugadores saque el número más alto.
- Si un jugador pulsa el botón para lanzar su turno no hace nada el programa, por que esta esperando el turno del jugador correspondiente.
- Si el juego termino y quiere volver a jugar, le puede dar a el botón de RESET para comenzar una partida nueva.
- Las reglas son las misma que el ludo normal, tipo si un jugador esta en la casilla 4 y lanza el dado y saca un 5, el jugado va pasando por la casilla 5 y 6, como llegamos hasta la casilla 6, el jugador debe retroceder casillas. Por lo tanto, vuelve a la casilla 5,4 y terminara en la casilla 3.
- Para que el juego termine debe a ver un ganador, se declara un ganador cuando un jugador termina en la casilla 6, y prende el led correspondiente del color del jugador ganador.

CODIGO

- El código esta comentado y ya dice el funcionamiento de cada subrutina.

```
; ZONA DE DATOS *****************************************************************

LIST       P=16F877A                                    ; Procesador utilizado.

INCLUDE  <P16F877A.INC>                                 ; Incluyendo la librería del PIC.

__CONFIG   _CP_OFF &  _WDT_OFF & _PWRTE_ON & _XT_OSC ; Configuración de ciertos fragmentos del PIC.

        CBLOCK 0x20                                     ; En esta posición empieza la RAM de usuario.

        Contador                                        ; Contador necesario para hacer el numero aleatorio.

        Jugador1_Casillas                               ; Indica el número de casilla que va el jugador 1.

        Jugador2_Casillas                               ; Indica el número de casilla que va el jugador 2.

        Numero_Turno                                    ; Guarda el número cuando lanza el dado para determinar
                                                        ; quien comienza jugando.

        ENDC

        #DEFINE         Pulsador1           PORTB,RB7 ; Pulsador para que juegue el jugador 1.

        #DEFINE         Pulsador2           PORTC,RC7 ; Pulsador para que juegue el jugador 2.

        #DEFINE F_GanaJugador1              PORTB,RB6 ; Bandera que indica que gana el jugador 1.

        #DEFINE F_GanaJugador2              PORTC,RC6 ; Bandera que indica que gana el jugador 2.

        #DEFINE         Display             PORTD     ; Toda la línea del puerto D para al display 7segmentos.
```

```
        Casilla_MAX         EQU    .6              ; Casilla máxima del juego de ludo.

; ZONA DE CÓDIGOS ******************************************************************

        ORG    0                                   ; El programa comienza en la dirección 0.
Inicio                                              ; Inicio del programa.
        bcf            STATUS,RP1
        bsf            STATUS,RP0                   ; Acceso al banco 1.
        movlw b'10000000'                   ; Configura RB7 como entrada y las demás como salida.
        movwf TRISB                         ; Configuramos las líneas del puerto B.
        movlw b'10000000'                   ; Configura RC7 como entrada y las demás como salida.
        movwf TRISC                         ; Configuramos las líneas del puerto C.
        clrf   TRISD                        ; Configuramos las líneas del puerto D como salida.
        bcf            STATUS,RP0           ; Acceso al banco 0.
        clrf   PORTB
        clrf   PORTC
        clrf   PORTD                        ; Limpia los puertos, B,C y D.
Principal                                           ; Etiqueta Principal del programa.
        clrf   Jugador1_Casillas
        clrf   Jugador2_Casillas            ; Comienzan en la casilla 1.
        movlw 0x01                          ; Movemos el dato literal para configurar la salida de los puertos.
        movwf PORTB
        movwf PORTC                         ; Los dos jugadores comienza en la casilla 1.
        call   InicializaContador           ; Inicializa el dado electrónico.
```

```
;
;Subrutina"Determina_Turno"----------------------------------------------------------------
;
; Esta subrutina indica cual jugador comenzara jugando primero, compara  el que saco el numero mayor
; el que saca el numero mayor comienza jugando y si los dos por casualida sacan el mismo numero
; se vuelve a ejectutar la misma subrutina hasta que un jugar saque un numero mas alto que el otro.
;
Determina_Turno
        clrf    Numero_Turno            ; Borramos lo que tenga el registro.
Espera_LanzaTurno1
        btfsc   Pulsador1               ; ¿Presiono el botón el jugador uno?
        goto    Espera_LanzaTurno1      ; No, Por lo tanto, espere que presione.
        call    Dado1                   ; Si, Por lo tanto, el jugador uno lanzo el dado.
        movf    Contador,W              ; Mueve el dato del contador a W.
        movwf   Numero_Turno            ; Guarda el número que saco lanzando el dado con el jugador 1,
Espera_LanzaTurno2
        btfsc   Pulsador2               ; ¿Presiono el botón el jugador dos?
        goto    Espera_LanzaTurno2      ; No, Por lo tanto, espere que presione.
        call    Dado2                   ; Si, Por lo tanto, el jugador dos lanzo el dado.
        movf    Numero_Turno,W  ; Recupera el valor del número que saco el jugador uno lanzando el dado.
        subwf   Contador,W              ; Restamos para ver que jugador lanzo el número más alto.
        btfss   STATUS,C                ; ¿El jugador 1 saco un número más alto?
        goto    EsperaJugador1          ; Si, por lo tanto, comienza jugando.
        btfss   STATUS,Z                ; ¿El jugador 2 saco un número más alto?
        goto    EsperaJugador2          ; Si, Por lo tanto, comienza jugando.
        goto    Determina_Turno ; No, Los dos sacaron el mismo número por lo tanto se vuelve a hacer el
                                ; mismo proceso.
EsperaJugador1          ; Espera que el jugador uno lanzó el dado y el jugador dos espera su turno.
        btfss   Pulsador1               ; ¿Presiono el botón el jugador uno?
        goto    Juega1                  ; Si, Lanza el dado el jugador 1.
```

```
        goto    EsperaJugador1          ; No, Espera a que juegue.
EsperaJugador2          ; Espera que el jugador dos lanzó el dado y el jugador uno espera su turno.
        btfss   Pulsador2               ; ¿Presiono el botón el jugador dos?
        goto    Juega2                  ; Si, Lanza el dado el jugador 2.
        goto    EsperaJugador2          ; No, Espera a que juegue.
;
; Etiqueta "Juega1 y Juega2"----------
; - Mientras que el jugador 1 o el jugador 2 mantenga el pulsador presionado se genera un numero "Aleatorio"
; en la subrutina "DadoElectronico".
;
; - Si el jugador gana prende el leds ganador y pone el microcontrolador en bajo consumo y si no llega a la casilla 6
; sede el turno al jugador otro jugador.
;
; - Si un jugador llega a la casilla 6 y aún tiene que avanzar lo que hace ahora es retroceder los espacios que le quedan
; Ejemplo: Si un jugador está en la casilla 4 y lanza el dado y sale el número 3, el jugador queda en la casilla 5.
;
; Jugador 1 comienza a jugar----------
;
Juega1
        call    Dado1                   ; Genera el numero aleatorio.
MueveAdelante1
        call    Retardo_500ms   ; Hacemos un retardo de 500ms para que se visualiza como se traslada el
                                ; jugador.
        incf    Jugador1_Casillas,F     ; Incrementa en uno el numero donde estaba el jugador 1.
        movf    Jugador1_Casillas,W     ; Mueve el dato de donde está ubicado el jugador 1 al registro W.
        call    TablaCasillas   ; El jugador va pasando por las casillas depende del número que saco el dado.
        movwf PORTB                     ; Muestra el led moviendo el jugador 1.
        movlw Casilla_MAX               ; Mueve el valor de la casilla máxima a W.
```

```
        subwf  Jugador1_Casillas,W  ; Restamos para comparar si se pasó de la casilla máxima.
        btfsc  STATUS,C             ; ¿El jugador llego a la casilla 6?
        goto   MueveAtras1          ; Si, Ahora decrementa en casillas.
        decfsz Contador,F           ; Decrementa en uno al contador y salta si es cero.
        goto   MueveAdelante1       ; Aun no es cero, sigue pasando por las casillas.
```

CompruebaJugador1

```
        btfsc  PORTB,RB5            ; ¿Esta en la casilla 6?
        goto   GanaJugador1         ; Si, Gana el jugador 1. Prende el LEDs ganador.
        goto   EsperaJugador2       ; Le sede el turno al jugador 2.
```

MueveAtras1

```
        movlw 0x05
        movwf Jugador1_Casillas
```

Mueve1

```
        call   Retardo_500ms    ; Hacemos un retardo de 500ms para que se visualiza como se traslada el
                                ; jugador.
        decf   Jugador1_Casillas,F  ; Decrementa en uno el numero donde estaba el jugador 1.
        movf   Jugador1_Casillas,W  ; Mueve el dato de donde está ubicado el jugador 1 al registro W.
        call   TablaCasillas    ; El jugador va pasando por las casillas depende del número que saco el dado.
        movwf  PORTB            ; Muestra el leds moviendo el jugador 1.
        decfsz Contador,F       ; Decrementa en uno al contador y salta si es cero.
        goto   Mueve1           ; Aun no llega a cero, sigue moviéndose el jugador.
        goto   EsperaJugador2   ; Le sede el turno al jugador 2.
```

```
;
; Comienza a jugar el segundo jugador----------------------------------------------------------------------
;
Juega2
        call    Dado2                   ; Genera el numero aleatorio.
MueveAdelante2
        call    Retardo_500ms   ; Hacemos un retardo de 500ms para que se visualiza como se traslada el
                                ; jugador.
        incf    Jugador2_Casillas,F     ; Incrementa en uno el numero donde estaba el jugador 1.
        movf    Jugador2_Casillas,W     ; Mueve el dato de donde está ubicado el jugador 1 al registro W.
        call    TablaCasillas   ; El jugador va pasando por las casillas depende del número que saco el dado.
        movwf PORTC                     ; Muestra el leds moviendo el jugador 1.
        movlw Casilla_MAX               ; Mueve el valor de la casilla máxima a W.
        subwf  Jugador2_Casillas,W ; Restamos para comparar si se pasó de la casilla máxima.
        btfsc   STATUS,C                ; ¿El jugador llego a la casilla 6?
        goto    MueveAtras2             ; Si, Ahora decrementa en casillas.
        decfsz  Contador,F              ; Decrementa en uno al contador y salta si es cero.
        goto    MueveAdelante2          ; Aun no es cero, sigue pasando por las casillas.
CompruebaJugador2
        btfsc   PORTC,RC5               ; ¿Esta en la casilla 6?
        goto    GanaJugador2        ; Si, Gana el jugador 2. Prende el LEDs ganador.
        goto    EsperaJugador1          ; Le sede el turno al jugador 2.
MueveAtras2
        movlw 0x05
        movwf Jugador2_Casillas
Mueve2
        call    Retardo_500ms       ; Hacemos un retardo de 500ms para que se visualiza como se traslada el
                                    ; jugador.
        decf    Jugador2_Casillas,F ; Decrementa en uno el numero donde estaba el jugador 1.
        movf    Jugador2_Casillas,W ; Mueve el dato de donde está ubicado el jugador 1 al registro W.
        call    TablaCasillas   ; El jugador va pasando por las casillas depende del número que saco el dado.
```

```
        movwf PORTC                     ; Muestra el leds moviendo el jugador 1.
        decfsz  Contador,F              ; Decrementa en uno al contador y salta si es cero.
        goto    Mueve2                          ; Aun no llega a cero, sigue moviéndose el jugador.
        goto    EsperaJugador1          ; Le sede el turno al jugador 2.

GanaJugador1
        bsf             F_GanaJugador1          ; Enciende la bandera que gana el jugador 1.
        sleep                   ; Pone al microcontrolador en bajo consumo. (Sale de aquí con un reset)

GanaJugador2
        bsf             F_GanaJugador2          ; Enciende la bandera que gana el jugador 2.
        sleep                   ; Pone al microcontrolador en bajo consumo. (Sale de aquí con un reset)

;
;Subrutina "TablaCasillas"----------------------------------------------------------------------
;
;Esta tabla se realiza estudiando las casillas de un ludo y ver como se prende poco a poco el led
;que es donde esta situado el jugador.
;


TablaCasillas
        addwf  PCL,F                    ; Salto indexado, salta a la configuración deseada.
Tabla0
        DT 0x01,0x02,0x04,0x08,0x10,0x20,0x20; Configuraciones de las posiciones de las casillas.
```

```
;
;Subrutina "DadoElectronico"----------------------------------------------------------------
;
;Dado1--> Jugador1 , Dado2-->Jugador2.
;
;Se ponen dos subrutinas o un dado para cada jugador para detectar cuando un jugador pulsa un botón
;y esperar que se deje de pulsar el boton para que salga de la subrutina y para que se vea
;cada jugador moviendose por las casillas.
;
Dado1                   ; Mientras se mantenga pulsado se quedará en esta subrutina.
        btfsc   Pulsador1           ; ¿Pulsador presionado? ¿(Pulsador1)=0?
        goto    Fin                 ; No, sale de la subrutina.
        call    Retardo_50micros    ; Hacemos un retardo de 50 microsegundos.
        btfsc   Pulsador1           ; Comprueba si es un rebote.
        goto    Fin                 ; Era un rebote y sale fuera.
        call    IncrementaVisualiza ; Incrementa el contador y lo visualiza.
DejoPulsar
        btfss   Pulsador1           ; ¿Dejo de pulsar? ¿(Pulsador1)=1?
        goto    Dado1               ; No, Sigue el generando un numero el dado.
        goto    Fin
;Subrutina para generar un dado electrónico para el segundo jugador-----------------------------
Dado2                   ; Mientras se mantenga pulsado se quedara en esta subrutina.
        btfsc   Pulsador2           ; ¿Pulsador presionado? ¿(Pulsador)=0?
        goto    Fin                 ; No, sale de la subrutina.
        call    Retardo_500micros   ; Hacemos un retardo de 50 microsegundos.
        btfsc   Pulsador2           ; Comprueba si es un rebote.
        goto    Fin                 ; Era un rebote y sale fuera.
        call    IncrementaVisualiza ; Incrementa el contador y lo visualiza.
DejoPulsar2
        btfss   Pulsador2           ; ¿Dejo de pulsar? ¿(Pulsador)=1?
```

```
        goto    Dado2                   ; No, Sigue el generando un numero el dado.
Fin
        return                          ; Retorna de la subrutina.
;
;Subrutina"IncrementaVisualiza"----------------------------------------------------------------------------------------
;
; Incrementa y visualiza en el display 7 segmentos el numero que va generando el dado.
;
IncrementaVisualiza                     ; Subrutina para incrementar y visualizar en el display.
        incf    Contador,F              ; Incrementa el contador en uno.
        movlw d'7'                      ; Movemos el dato literal para restar con el contador.
        subwf  Contador,W               ; Operación aritmética, W=Contador-W
        btfss   STATUS,C                ; ¿(Contador)<7?
        goto    Visualiza               ; Contador<7 visualiza normal
InicializaContador                      ; Subrutina para inicializar el contador, Dejarlo en cero.
        movlw 0x01                      ; Mueve el valor 1 al contador.
        movwf           Contador                ; No, Era igual o Mayor. Por tanto, resetea.
Visualiza                               ; Subrutina para Visualizar el contador en el Display.
        movf Contador,W                 ; Mueve el valor del contador al registro W.
        call Numero_a_7Segmentos ; Pasa el numero a 7Segmentos para poder verlo.
        movwf Display                   ; Mueve el dato al Display para poder visualizar.
        movlw 0x01
        return                          ; Retornamos de la subrutina.



;Fin de la subrutina ----------------------------------------------------------------------------
        INCLUDE <DISPLAY_7S.INC>        ; Incluimos la librería del Display 7Segmentos.
        INCLUDE <RETARDOS.INC>          ; Incluyendo la librería para hacer retardos.
        END                             ; Fin del programa.
```