“MANEJO DE INTERRUPCIONES EXTERNAS EN ARDUINO”

Trabajo Preparatorio N°5
Laboratorio de Sistemas Embebidos

Melanny Cecibel Dávila Pazmiño
Ingeniería en Telecomunicaciones
Facultad de Eléctrica y Electrónica
Quito, Ecuador
melanny.davila@epn.edu.ec

***Abstract*—En el siguiente documento se presenta el sustento teórico acerca del uso de interrupciones externas en Arduino, todo esto con el fin de implementar un ejemplo práctico que permita apreciar su utilidad en aplicaciones básicas.**

***Index Terms*—Arduino, interrupción, registro, ISR.**

I. Introducción

Los microprocesadores incorporan el concepto de interrupción, que es un mecanismo que permite asociar una función a la ocurrencia de un determinado evento. Es así como en programación, una interrupción es una señal recibida por el procesador o MCU, para indicarle que debe “interrumpir” el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situaciónc [1].

II. Objetivos

- Familiarizar al estudiante a través de un ejercicio de ejemplo con el uso y manejo de interrupciones externas la plataforma Arduino.
- Conocer las fuentes de interrupciones en la plataforma Arduino, hardware asociado, vectores de interrupción y aplicaciones [2].

III. Cuestionario

A. Definir en sus propias palabras qué son las interrupciones, sus tipos y cómo las procesa el microcontrolador de Arduino.

Las interrupciones son básicamente la alteración del flujo natural de un programa. Son útiles para hacer que las cosas sucedan automáticamente en los programas de microcontroladores y pueden ayudar a resolver problemas de sincronización [3]. Las interrupciones pueden ser activadas de maneras distintas como se presentan a continuación:

- **Interrupciones por Timer:** Son disparadas cuando se cumple un cierto tiempo definido, son muy usadas para controlar y leer sensores cada determinado tiempo [3].
- **Interrupciones por hardware:** Se utilizan para detectar eventos externos. Simplemente cuando el evento se produce se acciona una bandera de interrupción que detiene y altera el flujo natural del programa para atender esa interrupción, básicamente es un orden de prioridades. La condición de disparo se refiere a si la interrupción se va a ejecutar cuando detecte el paso del pin de interrupción de un estado alto a un estado bajo o cuando detecte el paso de un estado bajo a uno alto [4].

B. Enumere las instrucciones y registros para habilitar y atender las interrupciones. Describa con un diagrama el proceso de atención de una interrupción externa y los parámetros de disparo para una plataforma Arduino.

Las instrucciones para realizar una interrupción dentro de un código son las siguientes: attachInterrupt(digitalPinToInterrupt(pin), ISR, mode). Donde digitalPinToInterrupt() se usa para poder especificar el pin que sirve para realizar la interrupción, con anterioridad se debe verificar que pines son usados para realizar interrupciones en cada placa. El ISR, de las siglas en inglés “Interruption Service Routine” es la subrutina que se va a usar durante la interrupción. Mode se refiere al modo de detección de la interrupción [5].

Los registros usados para atender las interrupciones son los siguientes:

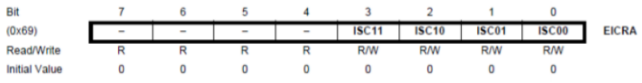

Fig. 1. EICRA – Registro A de Control de Interrupciones Externas [5]

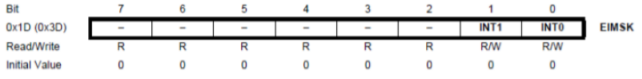

Fig. 2. EIMSK – Registro de Máscara de Interrupción Externa [5]

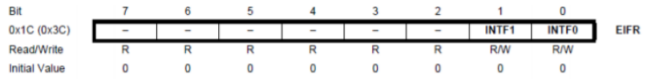

Fig. 3. EIFR – Registro de Bandera de Interrupción Externa [5]

Fig. 4. PCICR – Registro de Control de Interrupción por Cambio de Estado de Pin [5]

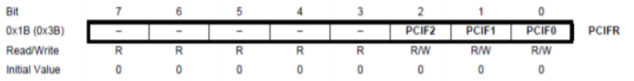

Fig. 5. PCIFR – Registro de Banderas de Interrupción por Cambio de Estado de Pin [5]

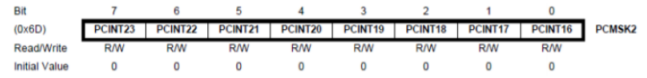

Fig. 6. PCMSK2 – Registro 2 de Máscara de Cambio de Estado de Pin [5]

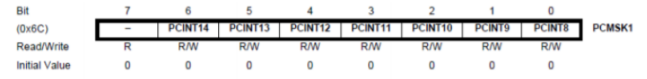

Fig. 7. PCMSK1 – Registro 1 de Máscara de Cambio de Estado de Pin [5]

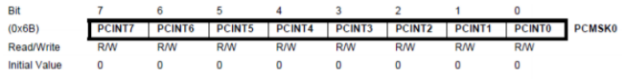

Fig. 8. PCMSK0 – Registro 0 de Máscara de Cambio de Estado de Pin [5]

Todos los dispositivos que deseen comunicarse con el procesador por medio de interrupciones deben tener asignada una línea única capaz de avisar al CPU cuando le requiere para realizar una operación, esta línea se denomina IRQ. Las IRQ son líneas que llegan al controlador de interrupciones, un componente de hardware dedicado a la gestión de las interrupciones, y que está integrado en la MCU [6].

El controlador de interrupciones debe ser capaz de habilitar o inhibir las líneas de interrupción y establecer prioridades entre las mismas. Cuando varias líneas de petición de interrupción se activan a la vez, el controlador de interrupciones utilizará estas prioridades para escoger la interrupción sobre la que informará al procesador principal. También puede darse el caso de que una rutina de tratamiento de interrupción sea interrumpida para realizar otra rutina de tratamiento de una interrupción de mayor prioridad a la que se estaba ejecutando [6].

Secuencia cuando se dispara una interrupción:

- El microcontrolador completa la instrucción que está siendo ejecutada.
- El programa de Arduino que se está ejecutando, transfiere el control a la Interrupt Service Routine (ISR). Cada interrupción tiene asociada una ISR que es una función que le dice al microcontrolador que hacer cuando ocurre una interrupción.
- Se ejecuta la ISR mediante la carga de la dirección de comienzo de la ISR en el contador del programa [5].
- La ejecución del ISR continua hasta que se encuentra el RETI (return from the interrupt instruction).
- Cuando ha finalizado ISR, el microcontrolador continua la ejecución del programa donde lo dejó antes de que ocurriera la interrupción.

Instrumentos para definir una interrupción:

- Un pin de Arduino que recibirá la señal de disparo.
- Una condición de disparo.
- Una función que se ejecutará, cuando se dispara la interrupción (Llamada call back function) [6].

La condición de disparo puede ser:

- LOW: la interrupción se dispara cuando el pin es LOW.
- CHANGE: se dispara cuando pase de HIGH a LOW o viceversa.
- RISING: dispara en el flanco de subida (Cuando pasa de LOW a HIGH).
- FALLING: dispara en el flanco de bajada (Cuando pasa de HIGH a LOW) [5].

C. Usando una tabla, compare: el número de pines y los modos de las fuentes de interrupción externas de los siguientes modelos de Arduino Uno, Nano, Mega, Leonardo y Yum.

TABLA I
COMPARACIÓN ENTRE MODELOS DE ARDUINO [7]

Placa	Pines	Modo de la fuente
Arduino Uno	2, 3	Interrupción externa por pin RB0/INT
Arduino Nano	2, 3	Interrupción externa por pin RB0/INT
Arduino Mega	2, 3, 18 19, 20, 21	Interrupción externa por pin RB0/INT y RB4 a RB7
Arduino Leonardo	0, 1, 2, 3, 7	Interrupción externa por pin RB0/INT y RB4 a RB7
Arduino Yun	0, 1, 2, 3, 7	Interrupción externa por pin RB0/INT y RB4 a RB7

Con el fin de aplicar lo descrito en los numerales anteriores, escriba un programa para ser ejecutado en la plataforma Arduino UNO en simulación. Considere que: en un partido de básquet, los equipos se identifican como equipo rojo y equipo verde. Con ayuda de dispositivo de información display de 7 segmentos, LEDs RGB y pulsadores, diseñar un circuito que realice la siguiente función; El display mostrará el total de canastas anotado por cada equipo cuando el pulsador que simula la meta es accionado. El LED RGB se encenderá con

el color del equipo que anotó. La información será mostrada por un intervalo de 5 segundos, después de los cuales solo el segmento "g" del display se encenderá y apagará con una frecuencia de 2 Hz. Incluya una rutina que evite que los dos interruptores puedan ser accionados de manera simultánea. El sistema se limitará al uso de un solo display por lo que no se considera un puntaje mayor que quince (0-F). Implementar el código utilizando Interrupciones externas, la simulación del circuito deberá ser realizada en el aula de Tinkercad.

A continuación, se presenta el esquemático realizado en el software de simulación Tinkercad junto con el código fuente que permite registrar el marcador de un partido de básquet.

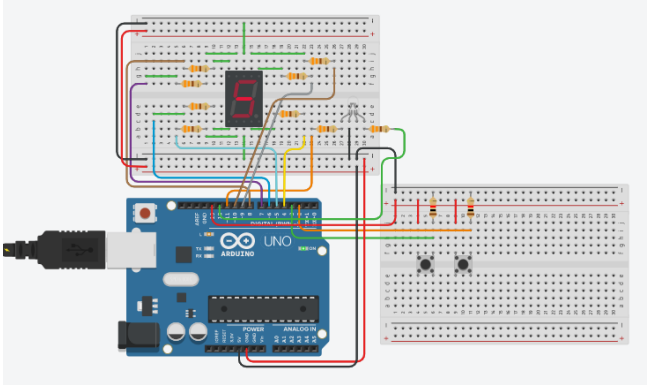

Fig. 9. Esquemático implementado

```
/*En el siguiente codigo se implementa una
    aplicacion de cuenta de anotaciones realizadas durante un partido de
    baloncesto usando un solo display de 7 segmentos
    identificando al equipo que realizo la anotacion*/

    int marcadorRojos = 0;
    int marcadorVerdes = 0;
    int res = 1;
    void setup()
    {
    Serial.begin(9600);/* Se establece una velocidad
    de transmision*/
    pinMode(4,OUTPUT);
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    pinMode(7,OUTPUT);
    pinMode(8,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(13,OUTPUT);
    digitalWrite (13,HIGH);
    attachInterrupt(digitalPinToInterrupt(2),aumentarRojos, RISING );
    attachInterrupt(digitalPinToInterrupt(3), aumentarVerdes, RISING);
    }


    void display (int a, int b, int c, int d, int e,
    int f, int g)
    {
    digitalWrite (10,a);/*Se establece que los pines 2,3,4,5,6,7,8
    31 coloquen como salida las variables que a,b,c,d,e,f,g*/
    digitalWrite (9,b);
    digitalWrite (4,c);
    digitalWrite (5,d);
    digitalWrite (6,e);
    digitalWrite (7,f);
    digitalWrite (8,g);
    }
    void loop()
    {
    encenderSegmentos(20);
    digitalWrite (13,HIGH);
    }

    void aumentarVerdes()
    {
    digitalWrite (13,LOW);
    marcadorRojos ++;
    encenderSegmentos(marcadorRojos);
    digitalWrite(12, HIGH);
    delay(10000);
    digitalWrite(12, LOW);
    }

    void aumentarRojos()
    {
    digitalWrite (13,LOW);
    noInterrupts();
    marcadorVerdes ++;
    encenderSegmentos(marcadorVerdes);
    digitalWrite(11, HIGH);
    delay(10000);
    digitalWrite(11, LOW);
    }

    void encenderSegmentos(int mensaje)
    {
    if (mensaje == 0)
    {
    display(1,1,1,1,1,1,0);//Configuracion para el 0
    }
    if (mensaje == 1)
    {
    display(0,1,1,0,0,0,0); //Configuracion para el 1
    }
    if (mensaje == 2)
    {
    display(1,1,0,1,1,0,1);//Configuracion para el 2
    }
    if (mensaje == 3)
    {
    display(1,1,1,1,0,0,1); //Configuracion para el 3
    }
    if (mensaje == 4)
    {
    display(0,1,1,0,0,1,1); //Configuracion para el 4
    }
    if (mensaje == 5)
    {
    display(1,0,1,1,0,1,1); //Configuracion para el 5
    }
    if (mensaje == 6)
    {
    display(1,0,1,1,1,1,1); //Configuracion
```

```
    para el 6
95          }
96          if (mensaje == 7)
97          {
98          display(1,1,1,0,0,0,0); //Configuracion
    para el 7
99          }
100         if (mensaje == 8)
101         {
102         display(1,1,1,1,1,1,1); //Configuracion
    para el 8
103         }
104         if (mensaje == 9)
105         {
106         display(1,1,1,0,0,1,1); //Configuracion
    para el 9
107         }
108         if (mensaje == 11)
109         {
110         display(1,1,1,0,1,1,1); //Configuracion
    para la A
111         }
112         if (mensaje == 12)
113         {
114         display(0,0,1,1,1,1,1); //Configuracion
    para la b
115         }
116         if (mensaje == 13)
117         {
118         display(1,0,0,1,1,1,0); //Configuracion
    para la C
119         }
120         if (mensaje == 14)
121         {
122         display(0,1,1,1,1,0,1); //Configuracion
    para la d
123         }
124         if (mensaje == 15)
125         {
126         display(1,0,0,1,1,1,1); //Configuracion
    para la E
127         }
128         if (mensaje == 16)
129         {
130         display(1,0,0,0,1,1,1); //Configuracion
    para la F
131         }
132         if (mensaje == 20)
133         {
134         display(0,0,0,0,0,0,0);/*Se limpian las
    entradas
135         anteriores*/
136         digitalWrite(8, HIGH);
137         delay(250);
138         digitalWrite(8, LOW);
139         delay(250);
140         }
141         }
142
```

Código 1: Partido de básquet

REFERENCES

[1] "Interrupts() - Arduino Reference". https://www.arduino.cc/reference/en/language/functions/interrupts/interrupts/ (accedido dic. 26, 2020).

[2] E. Espinosa, E. Tatayo, "MANEJO DE ESTRUCTURAS Y SUBRUTINAS EN ARDUINO". C.P. SISTEMAS EMBEBIDOS, Accedido: dic. 27, 2020. [En línea].

[3] C. Veloso, "Como usar las interrupciones en arduino - Timer y Hardware", Electrónica analogica y digital, may 13, 2016. https://www.electrontools.com/Home/WP/como-usar-las-interrupciones-en-arduino/ (accedido dic. 27, 2020).

[4] L. Llamas "Qué son y cómo usar interrupciones en Arduino". https://www.luisllamas.es/que-son-y-como-usar-interrupciones-en-arduino/ (accedido dic. 27, 2020).

[5] "Interrupciones con Arduino a través de un ejemplo práctico", Programar fácil con Arduino. https://programarfacil.com/blog/arduino-blog/interrupciones-con-arduino-ejemplo-practico/ (accedido dic. 28, 2020).

[6] "Interrupciones", Aprendiendo Arduino, nov. 13, 2016. https://aprendiendoarduino.wordpress.com/2016/11/13/interrupciones/ (accedido dic. 28, 2020).

[7] S. Espinosa, "El Microcontrolador ATMega328P". http://www.utm.mx/ fsantiag/Micros-MEOSIA/Notas-AVR-Parte2.pdf (accedido dic. 28, 2020).