UNIVERSIDAD DE ANTIOQUIA

PROYECTO DE INVESTIGACIÓN: INTERRUPCIONES

ALUMNO

ANDREA OSPINA HINCAPIÉ
andrea.ospinah@udea.edu.co
INFORMÁTICA II
INGENIERÍA ELECTRÓNICA

DOCENTE

AUGUSTO ENRIQUE SALAZAR JIMÉNEZ
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

3 DE JULIO DE 2020

PROYECTO DE INVESTIGACIÓN: INTERRUPCIONES

El concepto de las interrupciones es un concepto de suma relevancia para comprender por completo el funcionamiento de los microprocesadores. Si bien existen diferencias en la forma como estas pueden ser implementadas según el microprocesador que se esté utilizando, existen aspectos generales de las interrupciones que se presentarán a continuación por medio de la respuesta a importantes preguntas:

¿Qué es una interrupción en el contexto de los microprocesadores?

Una interrupción, como podría dar a entender su nombre, es una señal [1, p.217] que es enviada al microcontrolador para que, por medio de un mecanismo de interrupciones [1, p.217] se detenga el programa principal, de manera que se genere una respuesta, conocida como ISR (*Interrupt Service Routine*), y posteriormente se continúe el programa principal en el punto en el que fue interrumpido [2, p.1]. Para comprender por completo este concepto puede pensarse en una gran empresa con múltiples departamentos como el departamento de marketing, el departamento financiero, o el departamento de desarrollo, además de tener un presidente que se encuentra realizando una tarea (programa principal). De repente el presidente recibe una llamada del jefe del departamento de marketing (interrupción) en la cual le indica que es necesario hacer una reunión para discutir la publicidad de un nuevo producto. El presidente procede entonces a convocar una reunión (subrutina). Al finalizar esta reunión el presidente continúa con sus tareas hasta que reciba una nueva llamada del mismo u otro departamento. La situación anterior es, entonces, un ejemplo de cómo funciona el concepto de las interrupciones.

¿Se puede hablar de la historia de las interrupciones?

Existe un claro antecedente del concepto de interrupción: el *polling*, el cual consiste en que el microprocesador realice un monitoreo constante de un evento [2, p.1].

Continuando con el ejemplo de la empresa, el *polling* sería equivalente a que en lugar de que el presidente reciba la llamada de los departamentos, este se encargue de llamarlos constantemente hasta que la tarea asignada a cada departamento sea completada y se pueda realizar la reunión. Este mecanismo fue reemplazado en los microcontroladores por una unidad de interrupciones ya que el *polling* es poco eficiente y tiene el potencial de causar la pérdida de información importante sobre los eventos o sobre el programa principal [2, p.2].

En términos de implementación se le atribuye el primer uso de interrupciones a distintas máquinas, por ejemplo, la UNIVAC 1103 (1953) suele ser mencionada como una de las primeras computadoras en utilizar un sistema de interrupciones [3, p.181]por algunos autores. Sin embargo en su manual original estas no son mencionadas [4]. Por tanto también suele atribuirse el primer uso de interrupciones a la UNIVAC 1103A (1956)[4][5].

¿Que tipo de interrupciones existen?

Existen múltiples tipos de interrupciones pero se destacan los siguientes:

-Interrupciones de hardware vs interrupciones de software: Las interrupciones de software son aquellas que se producen internamente por una instrucción llamada explícitamente dentro del conjunto de instrucciones que se ejecutan por el microprocesador. Por otro lado, las interrupciones de hardware pueden subdividirse a su vez en interrupciones de hardware internas e interrupciones de

hardware externas. Las interrupciones de hardware externas son aquellas producidas por periféricos o hardware diferente al procesador principal, mientras que las interrupciones de hardware internas son las generadas por limitaciones del hardware para ejecutar instrucciones como operaciones matemáticas ilegales [6].

-Vectorizadas vs no vectorizadas: Como se explicará en las siguientes preguntas, las ISR suelen ser almacenadas en un vector en la memoria del microcontrolador. Las interrupciones vectorizadas son aquellas cuyas ISR tienen una posición predefinida en el vector, mientras que para las no vectorizadas, al ser estas recibidas por el microcontrolador, este envía una señal para que el hardware externo envíe el vector de interrupción donde se encuentra la ISR. [7, p.195]

-Enmascarables vs no enmascarables: Las interrupciones enmascarables son aquellas que pueden ser ocultadas al procesador cuando no es deseable que se interrumpa el programa principal. Este control puede realizarse por medio de software de manera global(activando o desactivando todas las interrupciones) o de manera individual. Las interrupciones no enmascarables son aquellas que, sin importar el estado global de las interrupciones, es necesario que sean transmitidas al procesador. Este último es el caso de interrupciones como RESET [8, p.11].

¿Cómo se hace la implementación de interrupciones a nivel de hardware?

A nivel de hardware, y para realizar el control de interrupciones, se utiliza un dispositivo denominado unidad de interrupciones, el cual se encarga de indicarle al microprocesador cuándo se presenta una interrupción de manera que pueda ser atendida. De esta forma el microprocesador consultará los vectores de interrupciones donde se encuentran almacenadas, en orden de prioridad, la dirección en memoria de las ISR que corresponden a cada interrupción o una instrucción para que se ejecute la ISR [8, p.9]. Adicionalmente, al producirse una interrupción, debe guardarse (normalmente en el stack) la información del estado del procesador (registros y banderas) antes de atender la interrupción para que sea posible retornar al programa principal una vez la ISR sea ejecutada [2, p.4]. Este último aspecto puede variar dependiendo del microcontrolador utilizado así que es responsabilidad del programador asegurarse de que se guarde la información necesaria del procesador [8, p.10]. Por último, para el caso en el que presenten varias interrupciones de manera simultánea, éstas suelen ser ejecutadas en orden de prioridad según una lista que se haya definido, por medio del hardware,en cada microcontrolador. Esta lista no puede ser cambiada.

¿Cómo se implementan las interrupciones por software?

Para la el manejo de interrupciones el software puede jugar distintos papeles como brindar el control de enmascarar o no interrupciones que se presenten por hardware. En el caso de interrupciones de software existen microcontroladores que las implementan como forma de comunicación con el sistema operativo. Este tipo de interrupciones también son conocidas como llamadas al sistema y en los microcontroladores en que son implementadas suelen permitir que se cambie entre modo supervisor y modo usuario [9, p.236]. Para que se pueda realizar este último proceso es necesario tener una interfaz llamada la *Application Binary Interface* (ABI), en la cual se tienen definidos aspectos como las llamadas al sistema y las instrucciones asociadas a estas. la ABI, como su nombre lo indica, utiliza lenguaje máquina por lo que es independiente de cualquier lenguaje de programación. Sin embargo, los aspectos definidos en esta, y que son utilizados para interrupciones de software, dependen de la arquitectura del microcontrolador [10, p.24].

Implementación de interrupciones

Para la implementación de interrupciones con arduino debe considerarse:

Interrupciones en Arduino Uno

Para Arduino Uno existen interrupciones externas asociadas a las interrupciones IN0 e IN1 que se encuentran en los pines digitales 2 y 3. Para la activación de las interrupciones solo se permiten los modos LOW (pin en 0V), CHANGE (se presenta cambio de estado), RISING (cambio de bajo a alto) y FALLING (cambio de alto a bajo). Además, para declarar variables que se modificarán en la ISR se declaran como volatile y es importante que la ISR creada no contenga delays y sea lo más simple posible. [2, p.11-14]. Para especificar las subrutinas se utiliza la siguiente función:

```
attachInterrumpt(PinToAttach,ISR_Name,Mode);
```

Con una ISR de tipo void y que no reciba parámetros.

Ejemplo de implementación

El siguiente ejemplo utiliza dos pulsadores para generar interrupciones a un programa principal en el que se encienden y apagan dos LEDs durante un periodo definido por una variable, la cual será modificada por medio de una interrupción creada al presionar uno de los pulsadores, de manera que el periodo disminuya de manera que no sea menor a un periodo mínimo definido en el programa. Adicionalmente, el segundo pulsador creará una interrupción cuyo efecto será apagar los LEDs oscilantes y encender un tercer LED durante tres segundos. Esto se presentará una vez se haya finalizado la oscilación de los LEDs que se está ejecutando en el momento en que se presiona el pulsador.

En la siguiente imagen se presentan las conexiones realizadas para la implementación del ejemplo:

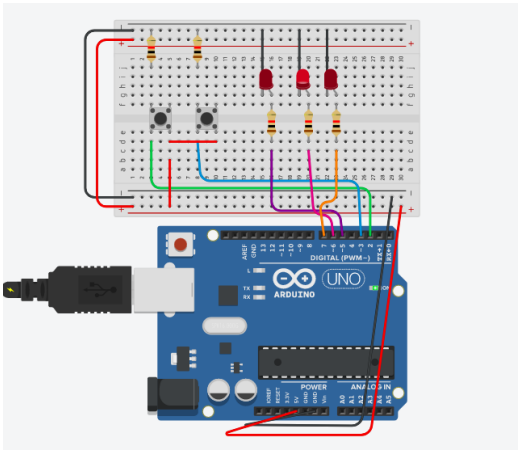

Imagen: Conexiones de ejemplo de implementación

El código fuente (EjemploInterrupciones.ino) de la implementación puede encontrarse en este repositorio.

Y un video del funcionamiento del ejemplo se encuentra en este link: `https://www.youtube.com/watch?v=WgSvHcRHfT4&feature=youtu.be`

REFERENCIAS BIBLIOGRÁFICAS

[1] M. Burrell, *Fundamentals of computer architecture*. Macmillan International Higher Education, 2003.

[2] F. Reyes Cortés and J. Cid Monjaraz, “Arduino: aplicaciones en robótica, mecatrónica e ingenierías,” *Mexico, Alfaomega, Barcelona, Marcombo*, 2015.

[3] S. H. Kaisler, *Birthing the Computer: From Relays to Vacuum Tubes*. Cambridge Scholars Publishing, 2016.

[4] M. Smotherman, “History and overview of interrupts and interrupt systems,” 2017. [Online]. Available: https://people.cs.clemson.edu/~mark/interrupts.html

[5] P. A. Laplante and S. J. Ovaska, *Real-time systems design and analysis: tools for the practitioner*. John Wiley and Sons, 2011.

[6] J. J. Labrosse, J. Ganssle, R. Oshana, C. Walls, K. E. Curtis, J. Andrews, D. J. Katz, R. Gentile, K. Hyder, and B. Perrin, *Embedded software: know it all*. Newnes, 2008.

[7] V. Udayashankara, *Microcontroller*. Tata McGraw-Hill Education, 2009.

[8] I. Susnea and M. Mitescu, *Microcontrollers in practice*. Springer Science & Business Media, 2005, vol. 18.

[9] M. Rafiquzzaman, *Microprocessors and microcomputer-based system design*. CRC press, 1995.

[10] W. Ecker, W. Müller, and R. Dömer, “Hardware-dependent software,” in *Hardware-dependent Software*. Springer, 2009, pp. 1–13.