Manipulación de los Pines GPIO de la Tarjeta Raspberry Pi empleando el Lenguaje C mediante el Entorno de Desarrollo Code::Blocks

Antonio Pérez Bautista
Email: apbesimez@hotmail.com
Centro de Innovación y Desarrollo Tecnológico en Cómputo, IPN.

Resumen

Este trabajo tiene como propósito exponer la implementación del lenguaje de programación C en el entorno de desarrollo integrado Code::Blocks para la manipulación de los pines de entrada y salida de propósito general del sistema embebido Raspberry Pi®. El lenguaje C es uno de los más importantes en el área de la Informática y Cómputo, ya que es un lenguaje con una sintaxis lógica y sencilla, así como la disponibilidad de diversas funciones que facilitan el desarrollo de programas de gran nivel. Contar con un buen ambiente de desarrollo es fundamental para un programador, por eso la herramienta Code::Blocks es escogida para la creación de programas en C por las herramientas y ambiente que maneja. La tarjeta Raspberry Pi® ha ganado gran popularidad para el desarrollo de aplicaciones y proyectos en las áreas de la Electrónica, Robótica y Mecatrónica por la disponibilidad de pines GPIO por los cuales se interactuar con dispositivos y equipos; y si le sumamos un excelente lenguaje como C y un ambiente de desarrollo como Code::Blocks, se tendrá una magnifica herramienta para

I. Introducción

La minicomputadora Raspberry Pi®, ver figura 1, incorpora un circuito integrado BCM2835 de tipo SoC (System on a chip) de la compañía Broadcom como microprocesador y unidad de procesamiento de gráficos (GPU). El BCM2835 se compone de un microprocesador ARM1176JZF-S que funciona a 700 MHz y de un GPU Broadcom VideoCore® IV, así como de 512 MB de memoria RAM la cual es compartida entre el procesador y la GPU [1].

Figura 1. Sistema embebido Raspberry Pi®

Los pines de Entrada y Salida de Propósito General conocidos como GPIO (General Purpose Input Output) son los puertos por los cuales se pueden conectar dispositivos electrónicos como interruptores, sensores digitales, transistores, controladores de motores, pantallas LCD, leds,

resistencias y demás componentes necesarios para desarrollar proyectos. Además de tener las funciones de entradas y salidas digitales, los pines GPIO cuentan con funciones especiales, de los que destacan diversos protocolos de comunicación como I2C, SPI, entre otros [2].

Los modelos "A" y "B" de la tarjeta Raspberry Pi® cuentan con 26 pines GPIO, mientras en los modelos "A+" y "B+" tienen 40. Debido a que estos pines están conectados directamente al chip del procesador, es importante tomar todas las precauciones al momento de realizar las conexiones a las terminales. El voltaje al que operan los pines GPIO es de 3.3 Volts, y no de 5 Volts, por lo que si la tarjeta es sometida a este nivel de voltaje se podría dañar, ya que no cuenta con protección contra sobretensiones.

II. Antecedentes

En el año 2006 en la Universidad de Cambridge, Inglaterra, el Dr. Eben Upton al darse cuenta de que los aspirantes a la carrera de informática habían disminuido considerablemente hasta un cincuenta por ciento, decidió crear un equipo de cómputo barato y que interesara de nuevo a los estudiantes para cursar las áreas de la informática e ingeniería [3]. En el 2008 los cuatro creadores originales de la minicomputadora Raspberry Pi®, Eben Upton, Rob Mullins, Jack Lang y Alan Mycroft junto con Pete Lomas y David Braben crearon la Fundación Raspberry Pi©, la figura 2 muestra el logo que representa a esta institución, que es la encargada de administrar la producción de las minicomputadoras así como las actualizaciones y desarrollo de aplicaciones para ésta. Tres años después de la creación de la fundación se empezaron a producir masivamente las primeras minicomputadoras para su venta [4].

Figura 2. Logo de la Fundación Raspberry Pi®

El lenguaje de programación C, el cual es una evolución del lenguaje B y BCPL, fue creado en 1972 por Dennis Ritchie, con la colaboración de Ken Thompson en los laboratorios Bell y se implementó originalmente en la computadora DEC PDP-11. En 1978 se publicó la primera edición de este lenguaje que se conoce como K&R C o C clásico. En 1983, el Instituto Nacional Americano de Estándares conocido por sus siglas en inglés como ANSI creó el comiteX3J11 cuyo objetivo era definir un estándar para regular el lenguaje C. En 1989 fue aprobado el estándar ANSI X3159:1989, la cual también recibe el nombre de ANSI C o C89. En el año de 1990, la Organización Internacional de Estándares, conocida como ISO por sus siglas en inglés, adopto el estándar ANSI C y se creó el ISO/IEC 9899, conocido también como C90 o ANSI/ISO C, aunque ambas versiones son la misma [5].

III. Desarrollo

Primero se abre una Terminal y se ingresa los comandos para actualizar los archivos y ficheros del sistema operativo:

pi@raspberrypi ~# sudo apt-get update
pi@raspberrypi ~# sudo apt-get upgrade

Si no se cuenta con el programa Code::Blocks se ingresa el siguiente comando para instalar el entorno de desarrollo integrado:

pi@raspberrypi ~# sudo apt-get install codeblocks

Una vez instalado el programa Code::Blocks, se descarga el archivo para el manejo de los pines GPIO mediante el lenguaje C. Esto se realiza ingresando desde el navegador de internet de la tarjeta Raspberry Pi® al enlace siguiente y dando click en la opción "SNAPSHOT":

https://git.drogon.net/?p=wiringPi;a=summary

Una vez descargado el archivo, se descomprime, esto se puede hacer mediante el gestor de archivo o por línea de comando en una Terminal. Para realizarlo por línea de comando entonces se ingresa:

pi@raspberrypi ~# tar xfz wiringPi-98bcb20.tar.gz

El nombre del archivo puede variar porque este se actualiza con frecuencia. Una vez descomprimido el archivo se ingresa a la carpeta, esto se realiza con el siguiente comando:

pi@raspberrypi ~# cd wiringPi-98bcb20

Una vez dentro de la carpeta se procede a la instalación de los archivos mediante el comando:

pi@raspberrypi ~# ./build

Ya terminada la instalación, el siguiente paso es la configuración del entorno de desarrollo integrado Code::Blocks para poder hacer uso de la biblioteca Wiringpi. Para hacer esto se realizan los siguientes pasos:

1. Ir a opción "SETTINGS".
2. Luego seleccionar "COMPILER AND DEBUGGER"
3. Escoger la pestaña "LINKER SETTINGS"
4. En el recuadro "LINK LIBRARIES", dar click en botón "ADD"
5. Aparecerá una ventada llamada "ADD LIBRARY", dar click en el botón "BROWSE  "
6. Se abrirá una nueva ventana llamada "CHOOSE LIBRARY TO LINK" en donde se busca la dirección "/usr/lib/libwiringPi.so" que es donde se encuentra la biblioteca y se acepta los cambios.
7. Luego se ingresa de nuevo a "SETTINGS"

8. Después se selecciona “ENVIRONMENT”
9. En el recuadro que dice “TERMINAL TO LAUNCH CONSOLE PROGRAMS” se añade la palabra “SUDO”, quedando de la siguiente manera: “sudo xterm -T $TITLE –e”

Con esto queda configurado el programa Code::Blocks para poder manipular los pines GPIO de la tarjeta.

IV. Experimentación y resultados

EL primer programa que se debe de ejecutar es el encendido y apagado de un led, esto se hace como prueba para comprobar el funcionamiento y correcta configuración.

Se abre el programa Code::Blocks y se crea un nuevo archivo C, el cual tendrá el siguiente código:

```
#include <stdio.h>
#include <wiringPi.h>

// LED Pin - wiringPi pin 1 es BCM_GPIO 18.
#define         LED     1

int main (void){
        printf ("Raspberry Pi programa blink\n") ;
        wiringPiSetup () ;
        pinMode (LED, OUTPUT) ;
        for (;;){
                digitalWrite (LED, HIGH) ;      // Encendido
                delay (500) ;                   // Retardo en mS
                digitalWrite (LED, LOW) ;       // Apagado
                delay (500) ;                   // Retardo en mS
        }
        return 0 ;
}
```

Luego se arma el circuito mostrado en la figura 3, para lo que se necesita de un led y una resistencia de 470 Ω o 560 Ω.

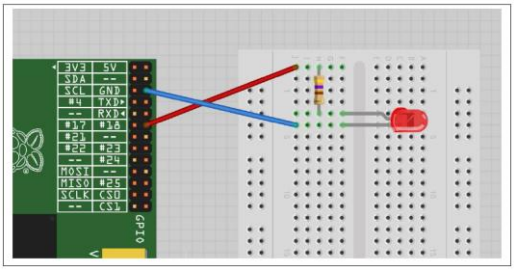

Figura 3. Circuito para experimentación

Después se compila el programa para cerciorarse de ningún error y, de no haber errores, se ejecuta el programa para comprobar que el led se enciende y se apaga cada medio segundo. La figura 4 muestra los resultados.

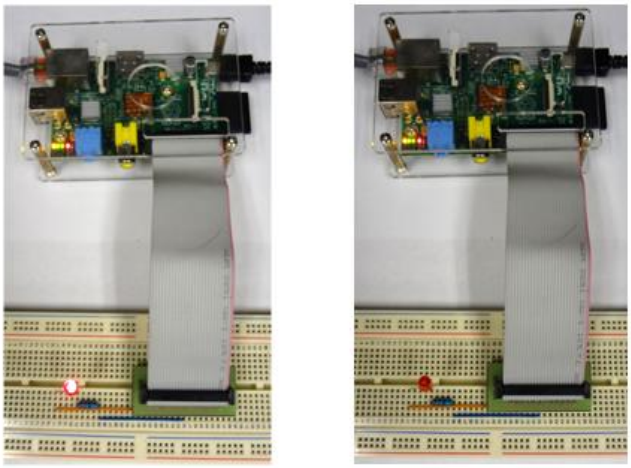

Figura 4. Resultado de las pruebas
(a) Led encendido
(b) Led apagado

V. Conclusiones

El leguaje C a pesar de ser desarrollado en la década de los 80's, sigue siendo un lenguaje de programación más empleado para el desarrollo de proyectos y aplicaciones por brindar un conjunto de funciones e instrucciones las cuales tiene un gran desempeño. Diversos microcontroladores como los PIC's y AVR, así como el sistema embebido Arduino, pueden ser programados con este lenguaje por su robustez y fácil desarrollo de código. El sistema Raspberry Pi® soporta sin ningún problema el desarrollo de programas en lenguaje C, dando como resultado la creación de diversos proyectos utilizando este lenguaje en las áreas de la Informática, Realidad virtual y aumentad, Inteligencia Artificial, entre otras; no solo queda en cuestiones de software, el desarrollo de aplicaciones en el que intervengan dispositivos reales como en la Robótica, Mecatrónica, Domótica y proyectos de Electrónica en general mediante el uso de los pines GPIO de la tarjeta, da como resultado un gran sistema para la creación de excelentes proyectos en estas áreas del conocimiento.

Referencias y recursos electrónicos

[1] Donald Norris (2013). *12 proyectos Raspberry Pi®*. Editorial Estribor.

[2] Matt Richardson y Shawn Wallace (2013). *Getting started with Raspberry Pi®*. O'REILLY.

[3] Enlaces: el mini computador Raspberry Pi. Última consulta: 4 de octubre de 2015. http://www.dw.de/enlaces-ventana-abierta-al-mundo-digital-2013-12-24/e-17285285-9797

[4] Wolfram Donat (2013). *Learn Raspberry Pi® Programming with Python*. Apress.

[5] Luis Joyanes Aguilar y Ignacio Zahonero Martínez (2010). *Programación en C, C++, Java y UML*. Mc Graw Hill.