

SISTEMA DE COMUNICACIÓN SERIAL LM4F120 - PC

Stellaris Launchpad LM4F120

Descripción breve

Sistema de intercambio de mensajes entre la tarjeta Stellaris Launchpad LM4F120 y la pc por medio del puerto serie de la tarjeta con implementación de un display lcd para el despliegue de mensajes y un teclado matricial 4x4.

Juan Miguel Vargas Sánchez
jmvarsan@gmail.com

Sistema de comunicación serial LM4F120 – PC

Objetivo

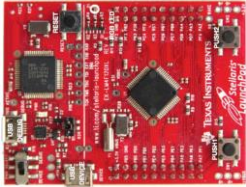



Con esta práctica se pretende mostrar el uso del módulo de comunicación serial para la transferencia de datos entre la tarjeta Stellaris Launchpad y la computadora también se hace uso de librerías útiles para la implementación de un display LCD y un teclado numérico 4x4.


Resumen

Primeramente se hicieron pruebas con la librería del LCD enviando una simple cadena para ser mostrada en la pantalla del LCD, posteriormente se hizo una librería para el teclado matricial 4x4, se configuro el módulo UART5 para comunicación serial y UART0 para usarlo como consola de información, el resultado final del uso de estos recursos es un sistema de comunicación serial y un teclado matricial con el despliegue de mensajes y datos del teclado a través del display LCD.

Desarrollo

El material utilizado para esta práctica es el siguiente

Tarjeta Stellaris Launchpad LM4F120H5QR	
Display LCD JHD-162 ^a	
Teclado matricial 4x4	
1 resistor de 470 ohms	

1 potenciómetro de 10 Kohms	
-----------------------------	--

Conexiones

En la figura 1 se muestran las conexiones del display con la tarjeta Stellaris Launchpad LM4F120, el puerto a utilizar es el puerto B, específicamente los pines mostrados en la tabla 1.

Puerto B	Pines LCD
PB0	RS
PB1	E
PB4	DB4
PB5	DB5
PB6	DB6
PB7	DB7

Tabla 1. Conexiones LM4F120- LCD.

El resto de pines del LCD se conectan de la siguiente manera:

Pines LCD	
VSS	GND
VCC	5 V
VEE	GND
R/W	GND
DB0-DB3	Sin conexión
LED+	Resistor 470 ohms
LED-	GND

Tabla 2. Resto de conexiones del LCD.

La librería del LCD trabaja con una interfaz de 4 bits por esta razón solo se utilizan los pines PB4-PB5 para la transmisión de datos de la tarjeta al LCD y los pines PB0 y PB1 para la configuración del LCD. La transmisión de mensajes entre la tarjeta y la pc se hace por medio del módulo UART_5 de la tarjeta, donde el pin 4 funciona como receptor y el pin 5 como transmisor, el puerto virtual que corresponde a l módulo UART_0 se usa como consola para ver logs de información sobre el sistema. Por último el teclado matricial se encuentra trabajando en conjunto con el resto del sistema, para ello se desarrolló su propia librería así como el uso de interrupciones externas.

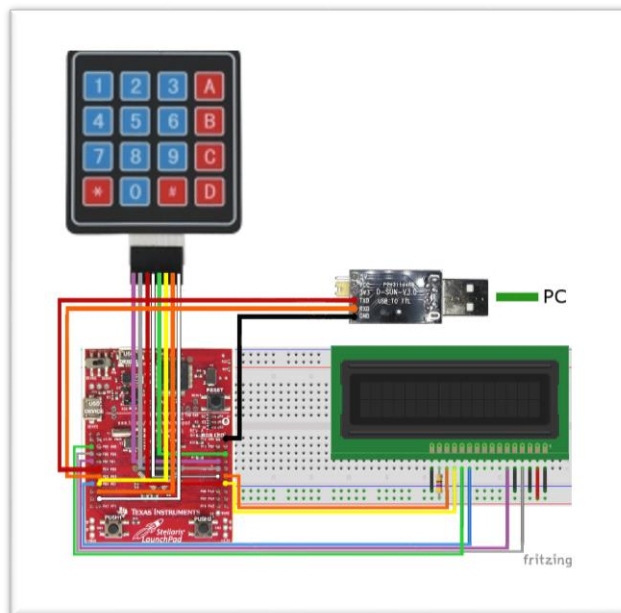


Figura 1. Conexiones del sistema de intercambio de mensajes por puerto serie, con implementación de display LCD y teclado matricial.

La parte del hardware se muestra en la figura 3

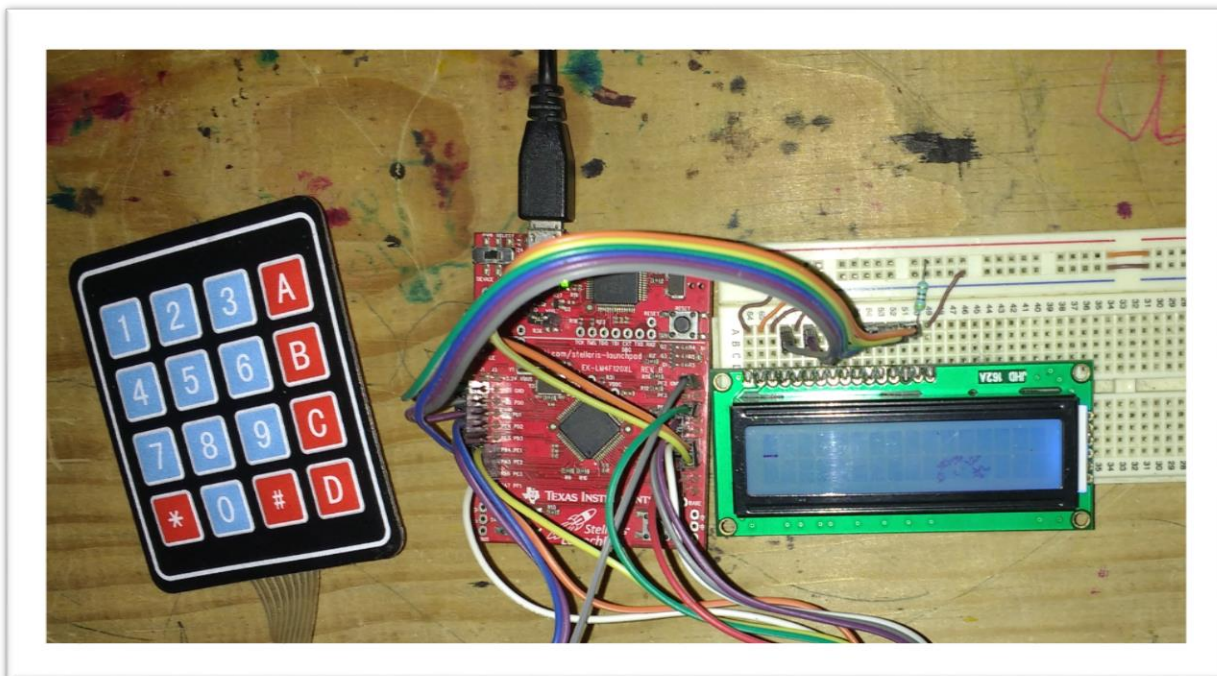


Figura 2. Sistema de comunicación serial con LCD.

Software

En el archivo Lcd.h se le asignan nombre al puerto B el cual se utilizara para la transferencia de datos entre la tarjeta y el LCD, así como los pines y las funciones a llamar durante la ejecución del programa principal.

Lcd.h

```
#define LCDPORT          GPIO_PORTB_BASE
#define LCDPORTENABLE    SYSCTL_PERIPH_GPIOB
#define RS                GPIO_PIN_0
#define E                GPIO_PIN_1
#define D7                GPIO_PIN_4
#define D6                GPIO_PIN_5
#define D5                GPIO_PIN_6
#define D4                GPIO_PIN_7

void Lcd_comando(unsigned char);
void Lcd_borrar(void);
void Lcd_Puts(char*);
void Lcd_Goto(char,char);
void Lcd_init(void);
void Lcd_Putch(unsigned char);
```

El archivo Lcd.c contiene cada una de las funciones definidas en el Lcd.h, en la función Lcd_init() se inicializa el display donde se habilitan los periféricos y configuran para que funcionen como un bus de salida de datos también haciendo uso de la función Lcd_comando() en la cual se definen comando preestablecidos en el Lcd los cuales encienden, limpian y ajusta la entrada de datos del display.

Lcd.c

```
#include "inc/hw_ints.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "Lcd.h"

void Lcd_init() {
    SysCtlPeripheralEnable(LCDPORTENABLE);
    GPIOPinTypeGPIOOutput(LCDPORT, 0xFF);

    SysCtlDelay(50000);

    GPIOPinWrite(LCDPORT, RS, 0x00 );

    GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, 0x30 );
    GPIOPinWrite(LCDPORT, E, 0x02);
    SysCtlDelay(10);
    GPIOPinWrite(LCDPORT, E, 0x00);

    SysCtlDelay(50000);

    GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, 0x30 );
    GPIOPinWrite(LCDPORT, E, 0x02);
    SysCtlDelay(10);
```

```

    GPIOPinWrite(LCDPORT, E, 0x00);

    SysCtlDelay(50000);

    GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, 0x30 );
    GPIOPinWrite(LCDPORT, E, 0x02);
    SysCtlDelay(10);
    GPIOPinWrite(LCDPORT, E, 0x00);

    SysCtlDelay(50000);

    GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, 0x20 );
    GPIOPinWrite(LCDPORT, E, 0x02);
    SysCtlDelay(10);
    GPIOPinWrite(LCDPORT, E, 0x00);

    SysCtlDelay(50000);

    Lcd_comando(0x28);
    Lcd_comando(0xC0);
    Lcd_comando(0x06);
    Lcd_comando(0x80);
    Lcd_comando(0x28);
    Lcd_comando(0x0f);
    Lcd_borrar();
}
void Lcd_comando(unsigned char c) {

    GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, (c & 0xf0) );
    GPIOPinWrite(LCDPORT, RS, 0x00);
    GPIOPinWrite(LCDPORT, E, 0x02);
    SysCtlDelay(50000);
    GPIOPinWrite(LCDPORT, E, 0x00);

    SysCtlDelay(50000);

    GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, (c & 0x0f) << 4 );
    GPIOPinWrite(LCDPORT, RS, 0x00);
    GPIOPinWrite(LCDPORT, E, 0x02);
    SysCtlDelay(10);
    GPIOPinWrite(LCDPORT, E, 0x00);

    SysCtlDelay(50000);
}

```

La función Lcd_Putch() recibe un carácter el cual es enviado y desplegado en el display.

```

void Lcd_Putch(unsigned char d) {

    GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, (d & 0xf0) );
    GPIOPinWrite(LCDPORT, RS, 0x01);
    GPIOPinWrite(LCDPORT, E, 0x02);
    SysCtlDelay(10);
    GPIOPinWrite(LCDPORT, E, 0x00);

    SysCtlDelay(50000);
}

```

```

        GPIOPinWrite(LCDPORT, D4 | D5 | D6 | D7, (d & 0x0f) << 4 );
        GPIOPinWrite(LCDPORT, RS, 0x01);
        GPIOPinWrite(LCDPORT, E, 0x02);
        SysCtlDelay(10);
        GPIOPinWrite(LCDPORT, E, 0x00);

        SysCtlDelay(50000);

    }

```

La función Lcd_Goto() recibe dos datos los cuales deben ser números que representan la posición donde se ubicara el cursor en el display.

```

void Lcd_Goto(char x, char y){

    if(x==1)
        Lcd_comando(0x80+((y-1)%16));
    else
        Lcd_comando(0xC0+((y-1)%16));

}

```

La función Lcd_borrar() no recibe ningún dato y sirve para limpiar el display, por último la función Lcd_Puts() recibe una cadena la cual es enviada carácter por carácter con la función Lcd_Putch al display LCD.

```

void Lcd_borrar(void){
    Lcd_comando(0x01);
    SysCtlDelay(10);
}

void Lcd_Puts( char* s){

    while(*s)
        Lcd_Putch(*s++);

}

```

En el programa principal primeramente se importan las librerías necesarias para el sistema así como las variables globales necesarias.

```

#include "inc/hw_types.h"
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/uart.h"
#include "driverlib/timer.h"
#include "utils/uartstdio.h"
#include "Lcd.h"
#include "keypad.h"

unsigned char dataFromUart[32];
int key_get;

```

Método InitConsole

En este método se hace una configuración básica de la uart 0, donde dicho modulo hace la función de una consola para mostrar logs de información durante la ejecución del sistema haciendo uso de la librería uartstdio.

```
void InitConsole(void)
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
    UARTStdioInit(0);
}
```

Método blinkled

Como su nombre lo dice este método sirve para ejecutar un parpadeo en el led rojo de la tarjeta stellaris, el cual sirve para notificar al usuario cuando se ejecutó alguna acción en el sistema.

```
void blinkled(void)
{
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);
    SysCtlDelay(SysCtlClockGet()/75);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
}
```

Método InitUARTComm

Este método hace la configuración necesaria para usar el módulo UART 1 de la tarjeta Stellaris, en el cual se habilitan periféricos y pines a utilizar, así como la configuración para la comunicación serial y habilitación de interrupciones.

```
void InitUARTComm(void)
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART5);

    GPIOPinConfigure(GPIO_PE4_U5RX);
    GPIOPinConfigure(GPIO_PE5_U5TX);
    GPIOPinTypeUART(GPIO_PORTE_BASE, GPIO_PIN_4 | GPIO_PIN_5);

    UARTConfigSetExpClk(UART5_BASE, SysCtlClockGet(), 115200,
    UART_CONFIG_WLEN_8|UART_CONFIG_PAR_NONE|UART_CONFIG_STOP_ONE);

    IntEnable(INT_UART5);
    UARTIntEnable(UART5_BASE, UART_INT_RX | UART_INT_RT);
}
```


Método sendDataUART

Este método sirve para hacer el envío de datos desde la tarjeta Stellaris hacia la PC por medio del módulo UART5.

```
void sendDataUART(unsigned char *data){
    while(UARTBusy(UART5_BASE));
        while(*data != '\0')
            UARTCharPut(UART5_BASE, *data++);
        blinkled();
}
```

Método UARTIntHandler

El método UARTIntHandler es el handler que se dispara cuando una interrupción es generada al momento de recibir datos de la PC en la tarjeta Stellaris a través de la comunicación serial, cuando es recibido el mensaje enviado desde la PC este es desplegado en el display con ayuda de la librería de Lcd.

```
void
UARTIntHandler(void)
{
    unsigned long ulStatus;
    unsigned char *recibe = dataFromUart;

    ulStatus = UARTIntStatus(UART5_BASE, true);
    UARTIntClear(UART5_BASE, ulStatus);
    while(UARTCharsAvail(UART5_BASE))
    {
        *recibe = UARTCharGetNonBlocking(UART5_BASE);
        *recibe++;
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
        SysCtlDelay(SysCtlClockGet() / 150);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
    }
    Lcd_borrar();
    Lcd_Puts(recibe);
}
```

Como parte de la configuración del proyecto es necesario hacer la configuración del handler UARTIntHandler en el archivo "lm4f120h5qr_startup_ccs.c", agregando las siguientes líneas:

```
#include <stdint.h>

.
.
.

// External declaration for the reset handler that is to be called when the
// processor is started
//
//*****
extern void _c_int00(void);
extern void UARTIntHandler(void);
```

```

.
.
.
IntDefaultHandler,          // SSI3 Rx and Tx
IntDefaultHandler,          // UART3 Rx and Tx
IntDefaultHandler,          // UART4 Rx and Tx
UARTIntHandler,             // UART5 Rx and Tx
IntDefaultHandler,          // UART6 Rx and Tx
IntDefaultHandler,          // UART7 Rx and Tx
.
.
.

```

Programa principal del sistema

Ya dentro del programa principal primero se define y configura el reloj del sistema el cual está trabajando a 40 MHz, posteriormente se habilitan el puerto F y se configura como salida, específicamente los pines PF1 y PF2 esto para hacer uso del LED rgb y así usarlo como indicador cuando se envía un dato a la pc o se recibe un dato desde esta, seguido de esto se hace la configuración de todos los módulos utilizados en el sistema.

También se hace la habilitación de las interrupción maestro y se envían los mensajes de información a la interfaz de comunicación serial y la consola de logs.

Main.c

```

int main(void) {
    SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2|GPIO_PIN_1);

    InitConsole();
    UARTprintf("UARTPrintf initialized ...\n");
    InitUARTComm();
    UARTprintf("UART initialized ...\n");
    initPorts();
    UARTprintf("Port E initialized ...\n");
    initIntPortE();
    UARTprintf("Port E interrupt initialized ...\n");
    Lcd_init();
    UARTprintf("LCD initialized ...\n");

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
    SysCtlDelay(SysCtlClockGet() / 150);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);

    IntMasterEnable();

    UARTprintf("Led indicator in LM4F120 configured ...\n");
    UARTprintf("Program started succesfully\n");
}

```

```

    unsigned char *buf = "Data send from LM4F120 to PC\n\n";
    sendDataUART(buf);
    while(UARTBusy(UART5_BASE));
    buf = "Type something and you send it for show it in the LCD, also you can
press some button from keypad 4x4 and see result in the LCD.";
    sendDataUART(buf);

    while (1) {
    }
}

```

En la figura 3 se muestra la consola donde se recibe y envían mensajes entre la pc y la tarjeta

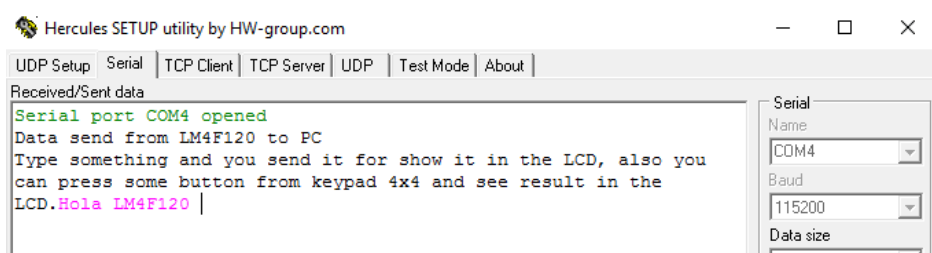


Ilustración 3. Consola de recepción y envío de mensajes.

El mensaje enviado desde la pc en el display Lcd se ve de la siguiente manera

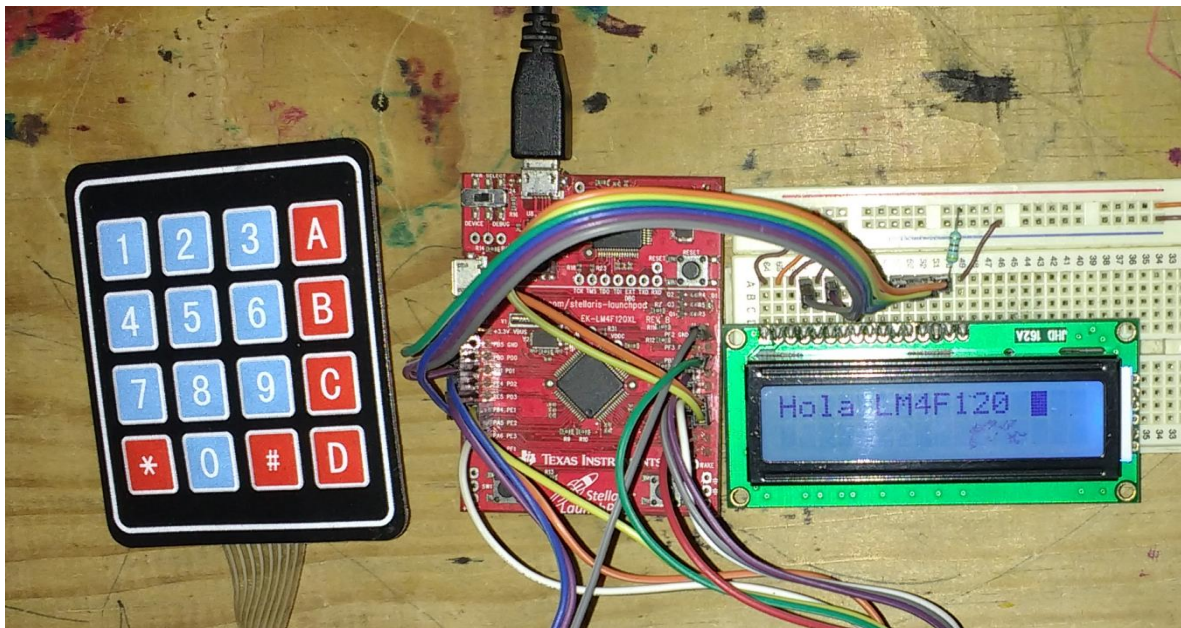


Ilustración 4. Mensaje enviado desde la pc y desplegado en el display Lcd.

Teclado matricial

Conexiones

Los puertos utilizados para la conexión del teclado matricial fueron el puerto C configurado como salidas para los renglones y el puerto D como entradas para las columnas.

Pin teclado	Pin LM4F120
PIN 1	PC4
PIN 2	PC5
PIN 3	PC6
PIN 4	PC7
PIN 5	PE0
PIN 6	PE1
PIN 7	PE2
PIN 8	PE3

Tabla 3. Conexión de pines entre el teclado y a tarjeta.

Software

El funcionamiento del Software es muy básico, en la salida del puerto C de los pines PC4-PC7 se hace un barrido, asignando los valores 0111, 1011, 1101, 1110 y a su vez se escanean las columnas para así identificar la tecla presionada.

A continuación se muestra el código contenido en el archivo keypad.h, en este archivo es donde se definen los puertos y pines a utilizar para la lectura del teclado matricial.

Keypad.h

```
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"

#ifndef KEYPAD_H_
#define KEYPAD_H_

#define ROWPORT GPIO_PORTC_BASE
#define COLPORT GPIO_PORTE_BASE
#define ROWPORTENABLE SYSCTL_PERIPH_GPIOC
#define COLPORTENABLE SYSCTL_PERIPH_GPIOE
#define ROW1 GPIO_PIN_4
#define ROW2 GPIO_PIN_5
#define ROW3 GPIO_PIN_6
#define ROW4 GPIO_PIN_7
#define COL1 GPIO_PIN_0
#define COL2 GPIO_PIN_1
#define COL3 GPIO_PIN_2
#define COL4 GPIO_PIN_3
#define OUTP1 0xE0
#define OUTP2 0xD0
#define OUTP3 0xB0
```

```

#define OUTP4 0x70
#define INP1 0x0E
#define INP2 0x0D
#define INP3 0x0B
#define INP4 0x07

extern unsigned long getKey(void);
extern void initPorts(void);

#endif

```

Por otro lado en el archivo keypad.c están contenidas las funciones que ejecutan el barrido y escaneo a los puertos asignados al teclado matricial, primeramente se habilitan los periféricos y se configuran el puerto C como salida y el puerto D como entrada, todo esto contenido en la función **initPorts**.

```

void initPorts (void){
    SysCtlPeripheralEnable(ROWPORTENABLE);
    SysCtlPeripheralEnable(COLPORTENABLE);
    GPIOPinTypeGPIOOutput(ROWPORT, ROW1 | ROW2 | ROW3 | ROW4);
    GPIOPinTypeGPIOInput(COLPORT, COL1 | COL2 | COL3 | COL4);
    GPIOPadConfigSet(COLPORT, COL1 | COL2 | COL3 | COL4, GPIO_STRENGTH_4MA,
    GPIO_PIN_TYPE_STD_WPU);
}

```

Posteriormente en la función **getKey** se encuentra la lógica que hace el barrido y lectura del teclado matricial, primeramente se definen el arreglo “sequence[4]” y variables necesarias en este proceso, como pueden notar se aplica un For para hacer el corrimiento de los renglones, dentro de este for se encuentra la instrucción “**GPIOPinWrite**(ROWPORT, ROW1 | ROW2 | ROW3 | ROW4, sequence[i])” la cual escribe en el puerto el primer valor del arreglo sequence e inmediatamente se hace la lectura de las columnas para localizar la tecla presionada, y así se sigue consecutivamente asignando los valores de 13, 11 y 7 (1011, 1101 y 1110 respectivamente), dependiendo de la tecla que se haya presionado esta función regresa un carácter entre 1 y 16 en hexadecimal y este es mostrado en el display LCD.

```

unsigned long getKey(void) {
    unsigned int sequence[4] = {OUTP1, OUTP2, OUTP3, OUTP4};
    int i, row, data;

    row = 0;

    for(i = 0; i < 4; i++){
        GPIOPinWrite(ROWPORT, ROW1 | ROW2 | ROW3 | ROW4, sequence[i]);
        asm("nop");
        data = GPIOPinRead(COLPORT, COL1 | COL2 | COL3 | COL4);

        if(data != 0xFF){
            /*Sets delay 120 ms*/
            SysCtlDelay(SysCtlClockGet()/15);
            switch(data){

```

```

        case INP1:
            return row;
        case INP2:
            return row + 1;
        case INP3:
            return row + 2;
        case INP4:
            return row + 3;
    }
    row += 4;
}

return 0xFF;
}

```

Para obtener el valor de la tarjeta, se hace uso de interrupciones externas para saber en que momento se ha presionado una tecla, por ello al inicio del programa principal se hace la configuración de los pines que utiliza el teclado matricial.

```

void initIntPortE(void){
    GPIOIntTypeSet(GPIO_PORTE_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_LOW_LEVEL);
    IntEnable(INT_GPIOE);
    GPIOPinIntEnable(GPIO_PORTE_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
}

```

Por ultimo en el programa principal cada que se detecta una interrupción externa en los pines correspondientes en el puerto E se procede a hacer la obtención del valor de la tecla presionada en el teclado matricial como se muestra a continuación.

```

void GPIOIntHandler (void){
    unsigned long ulStatus;
    unsigned char *key_char;

    ulStatus = GPIOPinIntStatus(GPIO_PORTE_BASE, true);
    GPIOPinIntClear(GPIO_PORTE_BASE, ulStatus);

    key_get = getKey();
    if(key_get != 0xff){
        switch(key_get) {

            case 0 :
                key_char = "0";
                break; /* optional */

            case 1 :
                key_char = "1";
                break; /* optional */

            case 2 :
                key_char = "2";
                break; /* optional */

            case 3 :
                key_char = "3";

```

```

        break; /* optional */

    case 4 :
        key_char = "4";
        break; /* optional */

    case 5 :
        key_char = "5";
        break; /* optional */

    case 6 :
        key_char = "6";
        break; /* optional */

    case 7 :
        key_char = "7";
        break; /* optional */

    case 8 :
        key_char = "8";
        break; /* optional */

    case 9 :
        key_char = "9";
        break; /* optional */

    case 10 :
        key_char = "10";
        break; /* optional */

    case 11 :
        key_char = "11";
        break; /* optional */

    case 12 :
        key_char = "12";
        break; /* optional */

    case 13 :
        key_char = "13";
        break; /* optional */

    case 14 :
        key_char = "14";
        break; /* optional */

    case 15 :
        key_char = "15";
        break; /* optional */
    }

    Lcd_borrar();
    Lcd_Puts(key_char);
    UARTprintf("Key pressed: %x\n", key_get);
}
}

```

Cabe mencionar que para hacer uso de la interrupción externa es necesario hacer configuraciones extras en el archivo ""

```
.
.
.
// processor is started
//
//*****
extern void _c_int00(void);
extern void UARTIntHandler(void);

.
.
.

IntDefaultHandler,          // GPIO Port B
IntDefaultHandler,          // GPIO Port C
IntDefaultHandler,          // GPIO Port D
GPIOIntHandler,             // GPIO Port E
IntDefaultHandler,          // UART0 Rx and Tx
IntDefaultHandler,          // UART1 Rx and Tx
IntDefaultHandler,          // SSI0 Rx and Tx
.
.
.
```

A continuación se muestra un ejemplo de cómo despliega el valor enviado desde el teclado matricial.

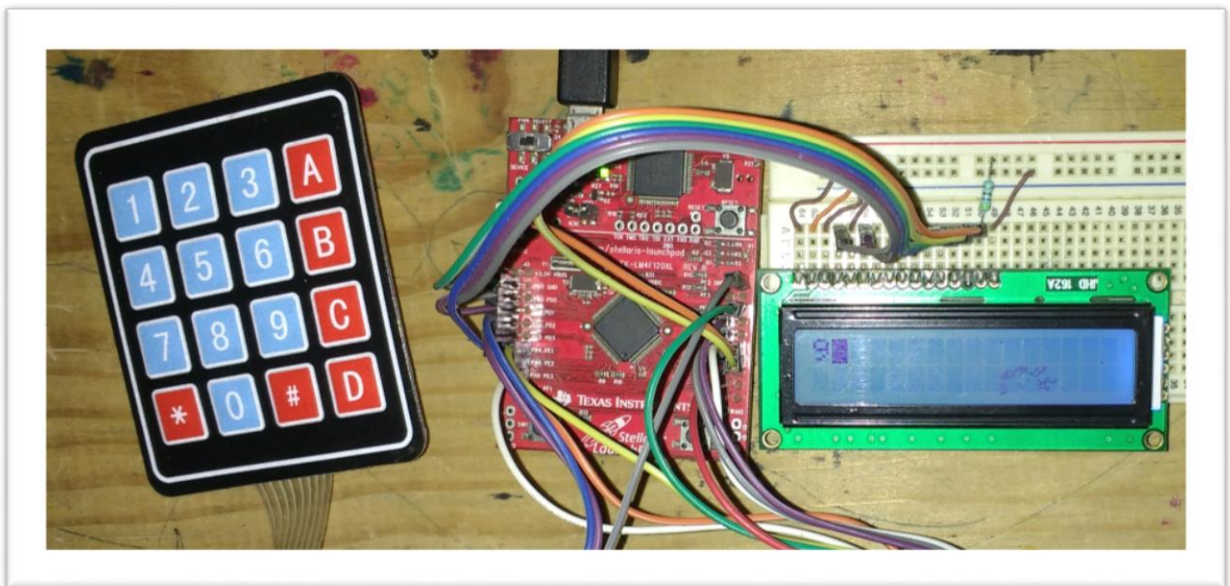


Ilustración 5. tecla 9 presionada y desplegada en el display.

Por otro lado en la consola de informacion se muestra lo siguiente

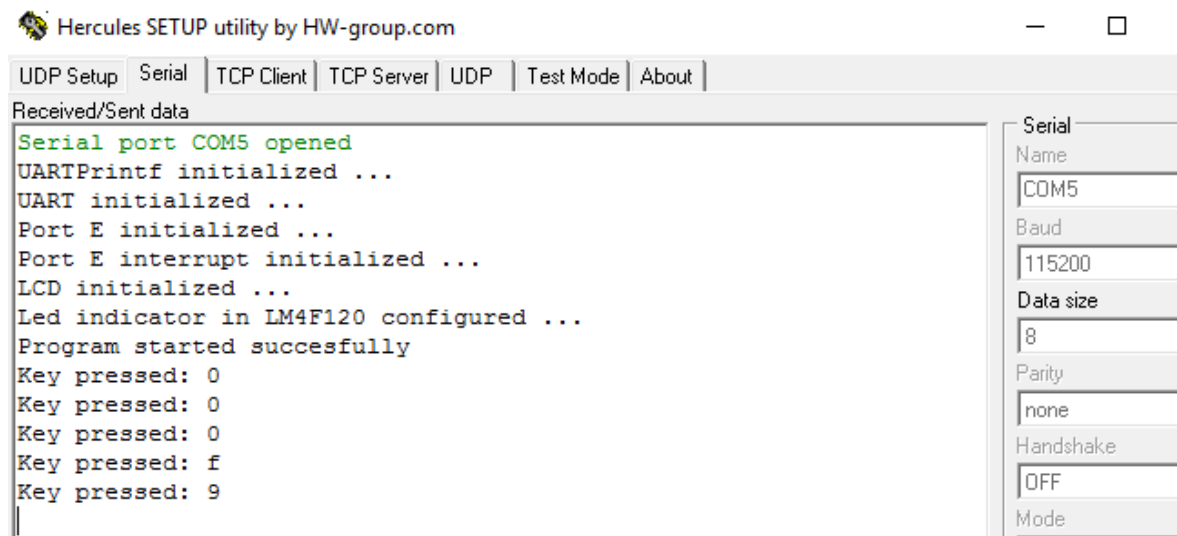


Ilustración 6. consola de log.

Conclusiones

El recurso de la comunicación serial en la tarjeta puede ser de gran utilidad pues con ello es posible crear un sistema con el cual se monitoreen variables físicas en tiempo real y estar monitoreando el comportamiento de estas remotamente, el uso del teclado matricial solo se hizo con el objetivo de ingresar datos directamente al Lcd y enviar estos a la pc por el puerto serial lamentablemente aún no se ha corregido el conflicto que hay con la función de inicialización de la librería del Lcd.

Bibliografía

- Bibliografía
- [Getting Started with the Stellaris® EK-LM4F120XL LaunchPad Workshop](#)
- [Stellaris® Peripheral Driver Library USER'S GUIDE](#)
- <http://codeforfree.weebly.com/tutorial-4--serial-communication-using-stellaris-launchpad.html>
- <http://www.mcu-turkey.com/stellaris-launchpad-16x2-lcd/>
- <http://tutorial.cytron.com.my/2012/08/15/project-17-%E2%80%93-interface-with-4x4-keypad-and-2x16-lcd/>