Procesadores Multinúcleo
Grupo: 1
Semestre 2024-1

Proyecto Final

Control Remoto de una casa mediante GSM / Tivafono

Rojas Juárez Adolfo

Enlace a Repositorio: `https://github.com/AdolfoRojasJ/Tivaphone`

1. Introducción

En este reporte se explica el desarrollo realizado en el microcontrolador TIVA C Series para configurar su funcionalidad como un teléfono así como manejo remoto de ciertas cualidades de un hogar mediante mensajes SMS a este mismo.

2. Antecedentes

Los microcontroladores TIVA C Series son una serie de microcontroladores lanzados por Texas Instruments con puertos analógicos, bajo consumo eléctrico, con un procesador mononúcleo ARM Cortex M4 incluído a 125 MHz. Entre sus características se incluye memoria Flash de 256 KB, SRAM de 32 KB, una EEPROM de 2KB, 3 comparadores analógicos, interfaces seriales incluyendo 8 UART's, 4 SSI/SPI, 2 CAN y 6 I2C (Morales, M. 2013).
El módulo A6 es un componente que incluye GSM/GPRS/GPS, funciona mediante comandos AT que se envían por comunicación en serie.

3. Material

- Tiva C Series TM4C1294NCPDT.
- Tiva C Series TM4C129ENCPDT.
- SIM Telcel.
- Módulo IoT A6.
- Display LCD 16x2.
- Teclado Matricial.
- Placas fenólicas.
- 2 Fuentes 5V 1A.
- 4 Leds blancos.
- Led Rojo.
- Buzzer.

- Motor a pasos con su respectivo Driver.
- Push Button.
- 2 Servomotores.
- Resistencias.
- Micrófono.
- Mini Amplificador PAM8403.
- Bocina de mínimo 8 Ohms

4. Hardware

Se utilizaron 2 microcontroladores TIVA C Series conectados vía UART como se muestra en la Figura 1.

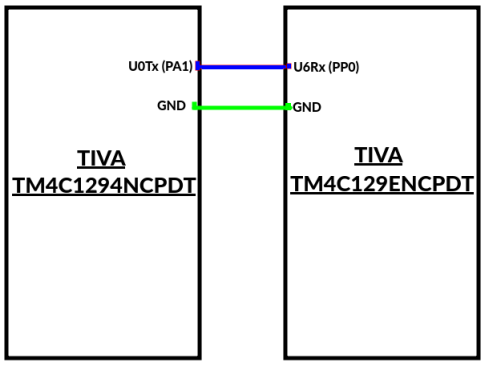

Figura 1: Circuito de conexión UART entre ambas TIVA's

La TIVA TM4C1294NCPDT será utilizada para la parte del teléfono, cuando un mensaje sea enviado a la TIVA esta **transmitirá** via UART un dato a una variable que la TIVA TM4C129ENCPDT **recibirá** y dependiendo el valor realizara una de las acciones relacionadas a control. La Tiva que realiza las acciones de teléfono tiene alambrado el módulo GSM, un teclado matricial, un Display LCD 2x16, un potenciómetro, un micrófono y un amplificador de audio junto a una bocina, las conexiones se muestran en el diagrama de la Figura 2.

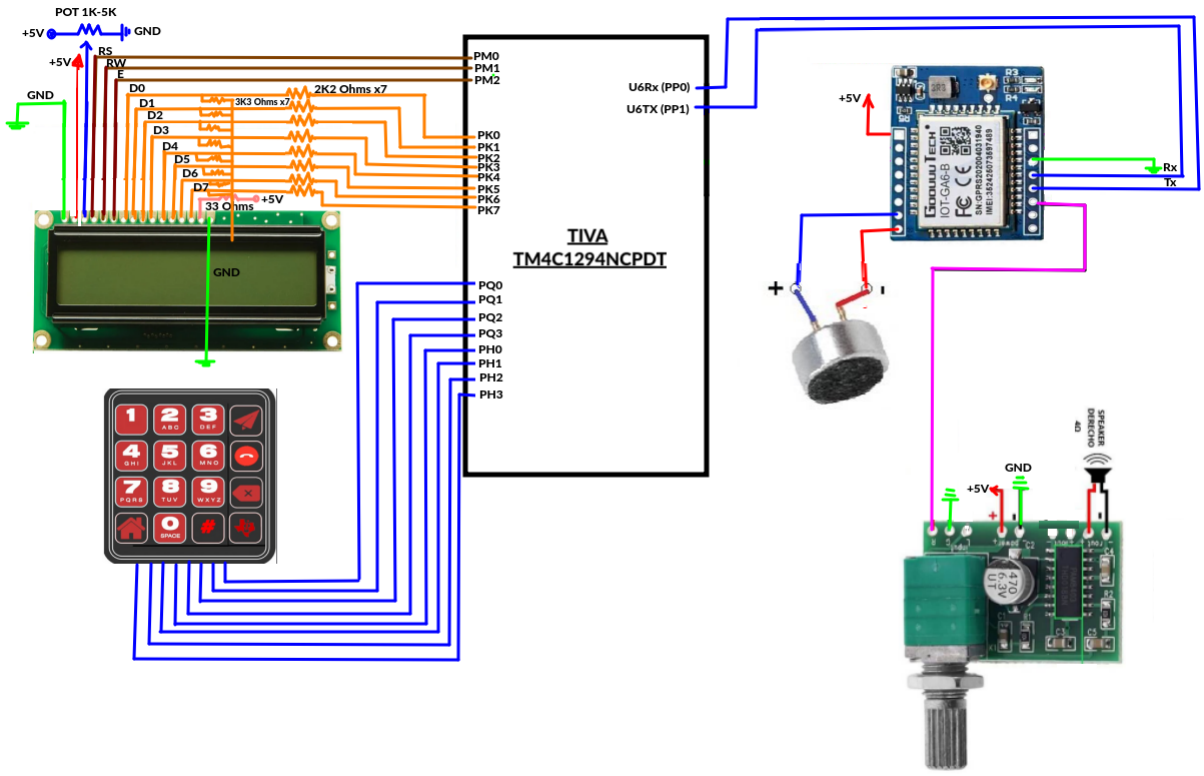

Figura 2: Circuito de conexión TIVA TM4C1294NCPDT

Mientras que la TIVA que está conectada a la casa de 6 habitaciones tiene alambrados leds a forma de simular los focos del hogar en las habitaciones 1, 2 y 3, un buzzer, un led rojo, un motor a pasos y un sensor de temperatura en la habitación 4 a manera de simular una alarma de incendios, un botón y un servomotor para el acceso al hogar en la habitación 5 y un servomotor para un dispensador de comida para mascotas en la habitación, las conexiones se pueden ver en la Figura 3.

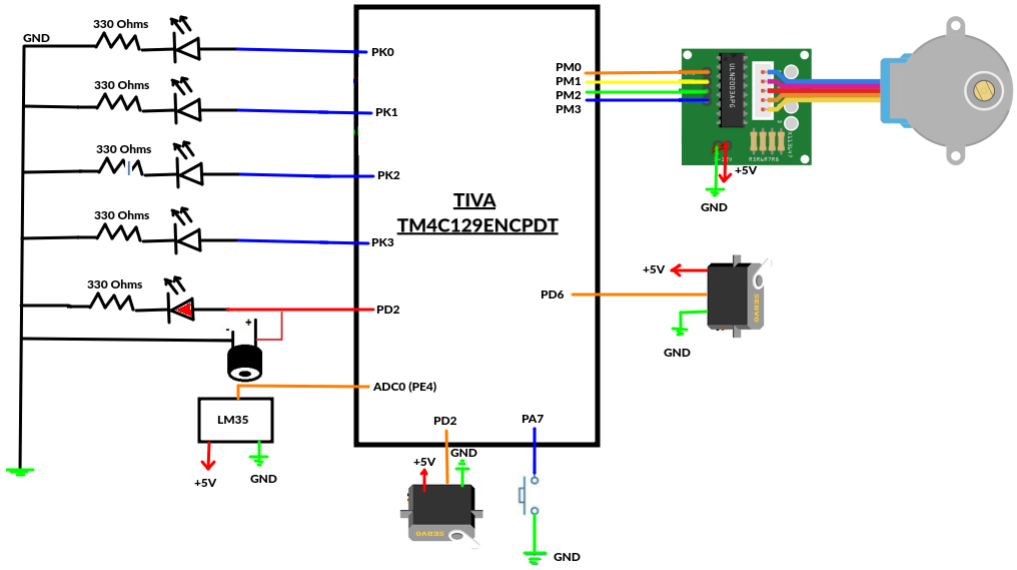

Figura 3: Circuito de conexión TIVA TM4C129ENCPDT

5. Software

Se utilizó el IDE Code Composser Studio para la programación, todo se realizó en lenguaje C, se usaron las librerías externas necesarias para trabajar con el microcontrolador, las cuales se consiguieron del siguiente enlace: `https://www.ti.com/tool/SW-TM4C`.

6. Instrucciones

En esta sección se explicarán los pasos necesarios para el correcto funcionamiento del Tiváfono y la conexión entre TIVA's para tener control vía SMS.

6.1. Configuración del proyecto en Code Composser

Se explicará como configurar correctamente el proyecto en el IDE.

6.1.1. Creación de un Proyecto en Code Coposser

Se dá clic en *New* y se elige la opción *CCS Project*, se nombra el proyecto como el usuario desee, cuidando que el Target sea *TIVA TM4C1294NCPDT* y la opción Connection sea *Stellaris In-Circuit Debug Interface*, una vez se elijan estas opciones, se crea el proyecto.

6.1.2. Inclusión de librerías externas

Se da clic derecho en el proyecto y se selecciona la opción *Properties*, damos clic en *Include Options* y añadimos la ruta donde instalamos las librerías del SDK (Tivaware).
Igualmente en las opciones de *Arm Linker* damos clic en *File Search Path*, y añadimos la librería *driverlib*, la cuál se encuentra en la ruta donde descargamos Tivaware y de ahí accedemos a: *driverlib/ccs/Debug/driverlib.lib*

6.2. TIVA 1. TIVAFONO

El pimer código se basa en un teléfono, el cúal es capaz de realizar llamadas, mandar mensajes y recibir mensajes para lanzar señales de control a la segunda TIVA, a manera de resúmen se explicará en las siguientes líneas partes fundamentales del código, pero el código completo se puede visualizar en el repositorio Github.
Las librerías utilizadas en el código fueron

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include "inc/tm4c1294ncpdt.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"
#include <stdio.h>
#include <string.h>
```

Figura 4: Librerías Utilizadas

Se declararon variables y arreglos los cuáles ayudaron a almacenar datos de interés, algunas funcionaron como banderas o contadores, igualmente se declararon diversas funciones las cuales ayudaron a manejar los cambios en el Display LCD o algunas fueron para Envío y Recepción de Datos por UART. El código principal se dividió a manera que el teléfono funciona de la siguiente forma:
MODO 1 TELEFONO (A)

- OPCION 1 LLAMAR (A)
- OPCION 2 COLGAR (B)
- OPCION 3 BORRAR (C)
- OPCION 4 REGRESA A HOME (*)

MODO 2 MENSAJE (B)

- OPCION 1 ENVIAR (A)
- OPCION 3 ELIMINAR TODO EL MENSAJE (C)
- OPCION 4 REGRESA A HOME (*)

MODO 3 CONTROL (C)

- OPCION 1 ABRIR PUERTA
- OPCION 2 ENCENDER FOCO 1
- OPCION 3 ENCENDER FOCO 2
- OPCION 4 ENCENDER FOCO 3
- OPCION 5 ENCENDE FOCO 4
- OPCION 6 ACTIVAR DISPENSADOR AUTOMÁTICO DE COMIDA PARA MASCOTAS
- OPCION 7 ENCENDER MATRIZ LED

- OPCION 8 ENVIAR SEÑAL DE CONTROL DE POTENCIA
- OPCION 9 APAGAR ALARMA DE INCENDIOS

Las opciones relacionadas a realizar llamadas y mensajes se hacen enviando caracteres por UART al A6, el cual admite comandos AT, por ejemplo para realizar una llamada ocupamos el comando: ATD55xxxxxxxx;
Donde las x se sustituyen por el número deseado, la lista de comandos se obtuvo de: `http://profesores.fi-b.unam.mx/use/se2024-1/gsm/comandosGSMbreviario.pdf` La función que es la inicialización de los puertos que se usaron se coloca a continuación:

```
void INIPORTS(void)
{
    SYSCTL_RCGCGPIO_R |= 0X6A81;
    while ((SYSCTL_PRGPIO_R&0x6A81)!=0x6A81);

    SYSCTL_RCGCUART_R |=0X41; //UART 0 Y 6

    //PUERTO A
    GPIO_PORTA_AHB_PCTL_R = (GPIO_PORTA_AHB_PCTL_R&0XFFFFFF00)+0X00000011;
    GPIO_PORTA_AHB_DEN_R|=0X03;
    GPIO_PORTA_AHB_AMSEL_R &= ~0X03;
    GPIO_PORTA_AHB_AFSEL_R |= 0X03;

    //PUERTO H
    GPIO_PORTH_AHB_DATA_R=0X00;
    GPIO_PORTH_AHB_DEN_R|=0XFF;
    GPIO_PORTH_AHB_DIR_R=0X0f;

    //PUERTO K
    GPIO_PORTK_DATA_R=0X00;
    GPIO_PORTK_DEN_R|=0XFF;
    GPIO_PORTK_DIR_R|=0Xff;

    //PUERTO M
    GPIO_PORTM_DATA_R=0X00;
    GPIO_PORTM_DEN_R|=0X07;
    GPIO_PORTM_DIR_R|=0x07;

    //PUERTO P
    GPIO_PORTP_PCTL_R = (GPIO_PORTP_PCTL_R&0XFFFFFF00)+0X00000011;
    GPIO_PORTP_DEN_R|=0X03;
    GPIO_PORTP_AMSEL_R &= ~0X03;
    GPIO_PORTP_AFSEL_R |= 0X03;

    //PUERTO Q
    GPIO_PORTQ_DEN_R|=0XFF;
    GPIO_PORTQ_DIR_R=0X00;
    GPIO_PORTQ_PUR_R=0X0F;

    //UART 0
    UART0_CTL_R &=~0X0001;
    UART0_IBRD_R = 8;
    UART0_FBRD_R =43;
    UART0_LCRH_R =0X0070;
    UART0_CTL_R= 0X0301 ;

    //UART 6
    UART6_CTL_R &=~0X0001;
    UART6_IBRD_R = 8 ; //115200
    UART6_FBRD_R =43 ;
    UART6_LCRH_R =0X0070; //8 BITS, HABILITAR FIFO
    UART6_CTL_R= 0X0301 ;
}
```

Figura 5: Inicialización de Puertos

Se coloca el valor 0x6A81 en el Reloj para dar a entender que usaremos los puertos A,H,K,M,P y Q, los puertos A y P se usarán para UART, los puertos K y M son usados para señales de control

del Display de 16x2 y los puertos H y Q son usados para señales del teclado Matricial.

En el modo control se manda a llamar a la función que lee lo recibido por UART de parte del teléfono, la cuál se muestra a continuación:

```
char UART6_Lee_Dato(void)
{
while((UART6_FR_R&0X0010)!=0);
d_uint8Message=((char)(UART6_DR_R&0xff));
}
```

Figura 6: Función de Recepción de Datos del teléfono

La función Almacena el contenido en un arreglo, y si los elementos 73 a 76 (Ahí se almacena el contenido del mensaje) son una cadena previamente definida se le da un valor a una variable que será enviada por el otro UART a la TIVA 2 mediante la siguiente función:

```
char UART0_Escribe_Dato(char dato1)
{
while   ((UART0_FR_R&0X0020)!=0);
UART0_DR_R=dato1;
}
```

Figura 7: Función de Envío de Datos a la TIVA 2

Como recordamos del diagrama de la Figura 1, la TIVA 1 envía datos a través del UART0, mientras que la TIVA2 lo recibe a través del UART6.

6.3. TIVA 2. CASA

Este circuito se centrará simplemente en una TIVA que tiene las conexiones de todos los leds, servomotores, y cosas que serán controladas, el funcionamiento se limita a recibir por UART una variable con un valor preestablecido, y dependiendo ese valor, se manda a realizar la acción correspondiente, las librerías utilizadas en el código fueron

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c1294ncpdt.h"
#include "driverlib/sysctl.h"
#include "driverlib/rom_map.h"
#include <math.h>
```

Figura 8: Librerías Utilizadas

La función que es la inicialización de los puertos que se usaron se coloca a continuación:

```
void INIPORTS(void)
{
    //TIMERS
    SYSCTL_RCGCGPIO_R |= 0x00002A1B; //A,B,D,E,K,M,P
    SYSCTL_RCGCTIMER_R |= 0X13; //TIMER 0 & 1 & 4
    SYSCTL_RCGCADC_R  = 0x01; //ADC0
    SYSCTL_RCGCI2C_R |= 0x0001;//I2C
    SYSCTL_RCGCUART_R |=0X40;//UART
    ui32Loop = SYSCTL_RCGCGPIO_R;

    //PUERTO A
    GPIO_PORTA_AHB_DIR_R &= ~0x80;
    GPIO_PORTA_AHB_DEN_R |= 0x80;
    GPIO_PORTA_AHB_PUR_R |= 0x80;
    GPIO_PORTA_AHB_IS_R &= ~0x80;
    GPIO_PORTA_AHB_IBE_R &= ~0x01;
    GPIO_PORTA_AHB_IEV_R &= ~0x80;
    GPIO_PORTA_AHB_ICR_R = 0x80;
    GPIO_PORTA_AHB_IM_R |= 0x80;

    //PUERTO B
    GPIO_PORTB_AHB_AFSEL_R |= 0x0C;
    GPIO_PORTB_AHB_ODR_R |= 0x08;
    GPIO_PORTB_AHB_DIR_R |= 0x0C;
    GPIO_PORTB_AHB_DEN_R |= 0x0C;
    GPIO_PORTB_AHB_PCTL_R|=0x00002200;

    //PUERTO D
    GPIO_PORTD_AHB_DEN_R |= 0x45;
    GPIO_PORTD_AHB_DIR_R |= 0x45;
    GPIO_PORTD_AHB_DATA_R = 0x00;
    GPIO_PORTD_AHB_AFSEL_R = 0x45;
    GPIO_PORTD_AHB_PCTL_R = 0x03000303;

    //PUERTO E
    GPIO_PORTE_AHB_DIR_R = 0x00;
    GPIO_PORTE_AHB_AFSEL_R |= 0x10;
    GPIO_PORTE_AHB_DEN_R = 0x00;
    GPIO_PORTE_AHB_AMSEL_R |= 0x10;

    //PUERTO K
    GPIO_PORTK_DEN_R |= 0x0F;
    GPIO_PORTK_DIR_R |= 0x0F;
    GPIO_PORTK_DATA_R = 0x00;

    //PUERTO M
    GPIO_PORTM_DIR_R|=0xFF;
    GPIO_PORTM_DEN_R|=0XFF;
    GPIO_PORTM_DATA_R=0x00;

    //PUERTO P
    GPIO_PORTP_PCTL_R = (GPIO_PORTP_PCTL_R&0XFFFFFF00)+0X00000011;
    GPIO_PORTP_DEN_R|=0X03;
    GPIO_PORTP_AMSEL_R &= ~0X03;
    GPIO_PORTP_AFSEL_R |= 0X03;
```

```
    //TIMER 0
    TIMER0_CTL_R=0X00000000;
    TIMER0_CFG_R= 0X00000004;
    TIMER0_TAMR_R= 0X0000000A;

    //TIMER 1
    TIMER1_CTL_R=0X00000000;
    TIMER1_CFG_R= 0X00000004;
    TIMER1_TAMR_R= 0X0000000A;

    //TIMER 4
    TIMER4_CTL_R=0X00000000;
    TIMER4_CFG_R= 0X00000004;
    TIMER4_TAMR_R= 0X0000000A;

    //I2C0
    I2C0_MCR_R = 0x00000010;
    I2C0_MTPR_R = TPR;

    //ADC0
    SYSCTL_PLLFREQ0_R |= SYSCTL_PLLFREQ0_PLLPWR;
    while((SYSCTL_PLLSTAT_R&0x01)==0);
    SYSCTL_PLLFREQ0_R &= ~SYSCTL_PLLFREQ0_PLLPWR;
    ADC0_ISC_R = 0x0008;                // Se recomienda Limpia la bandera RIS del ADC0
    ADC0_ACTSS_R = 0x0000;
    ADC0_PC_R = 0x01;
    ADC0_SSPRI_R = 0x0123;
    ADC0_EMUX_R = 0x0000;
    ADC0_SSEMUX3_R = 0x00;
    ADC0_SSMUX3_R = (ADC0_SSMUX3_R & 0xFFFFFFF0) + 9;
    ADC0_SSCTL3_R = 0x0006;
    ADC0_IM_R = 0x0000;
    ADC0_ACTSS_R |= 0x0008;

    //UART 6
    UART6_CTL_R &=~0X0001;
    UART6_IBRD_R = 8 ; //115200
    UART6_FBRD_R =43 ;
    UART6_LCRH_R =0X0070; //8 BITS, HABILITAR FIFO
    UART6_CTL_R= 0X0301 ;

    //INTERRUPCIONES
    NVIC_EN0_R= 0x01;
}
```

Figura 9: Inicialización de Puertos

Para este código se habilitaron los puertos A,B,D,E,K,M y P.
En este código se cuenta con una interrupción, la cual corresponde al botón del puerto A, la cual cambia el estado de la bandera *door* para hacer que el servomotor cambie a estado de puerta cerrada.

```
void GPIOPortA_Handler(void)
{
    GPIO_PORTA_AHB_ICR_R = 0x80;
    door=1;
}
```

Figura 10: Función de interrupción

Para que la interrupción tenga efecto, se añade como una función externa en el archivo:
tm4c129encpdt_startup_ccs.c
Incluído dentro del proyecto.
La función encargada de recibir los datos de la otra TIVA se muestra a continuación:

```
char UART6_Lee_dato(void)
{
    d_uint8Dato=((char)(UART6_DR_R&0xff));//lecturayescritura
    return d_uint8Dato;
}
```

Figura 11: Función de Recepción de datos de la TIVA 1

Esta función se manda a llamar en el programa principal, lee la variable que almacenará el contenido que será enviado de la otra TIVA.

7. Conclusión

El proyecto presenta ciertas limitaciones que debido a tiempo no fueron controladas, por ejemplo, el teléfono es capaz de realizar llamadas y enviar mensajes, sin embargo no puede contestar una llamada que le entre, así como no puede revisar mensajes que recibe a menos que esté en Modo Control, sin embargo la comunicación entre ambos procesadores se realiza de manera correcta, una Tiva obtiene un dato que se envía a la otra mediante UART y ejecuta la acción adecuada.

8. Referencias

- Miguel M. (2013). *An Introduction to the Tiva C Series Platform of Microcontrollers* (). Recuperado 2 de Diciembre de 2023, de `https://www.ti.com.cn/cn/lit/wp/spmy010/spmy010.pdf`
- Texas Instruments. (2014). Tiva™ C Series TM4C1294 Connected LaunchPad Evaluation Kit EKTM4C1294XL User's Guide (SPMU365B).
- I2C Info: A Two-wire Serial Protocol. I2C info - I2C bus, interface and protocol, 2020. Recuperado 14 de Octubre de 2023, de `https://i2c.info/`