Facultad de Ingeniería, UNAM. DIE.
Departamento de Ingeniería Electrónica.
Sistemas Embebidos.

Sistemas Embebidos

**Reporte Final**

# Proyecto Teléfono Inteligente con Inversor controlado por Ancho de Pulso utilizando un módulo GSM controlado por Microcontrolador

| Nombre completo de los alumnos: | N° de cuenta |
|---|---|
| Reyes Herrera Iván | 316286327 |
| Roa López Jonathan Raúl | 316089290 |
| **Fecha de elaboración**: 11/12/23 | **Grupo:** 2 |

ING. MOISES EUGENIO RUEDA GUTIERREZ

Semestre 2024-1

Facultad de Ingeniería

## Contenido

# Proyecto Teléfono Inteligente con Inversor controlado por Ancho de Pulso utilizando un módulo GSM controlado por Microcontrolador

## 1. Objetivo del trabajo

- Crear un sistema embebido con un propósito específico mediante la aplicación de los conocimientos adquiridos en clase, proponiendo así una solución para abordar un problema real.

## 2. Introducción

### Sistemas Embebidos

Un sistema embebido se trata de un sistema informático que opera mediante un microprocesador o microcontrolador, destinado a ejecutar una o algunas funciones específicas, a diferencia de las computadoras de propósito general, como las computadoras personales (PC). En los sistemas embebidos, la mayoría de los elementos, como la tarjeta de vídeo, audio, módem, entre otros, están integrados en la placa base.

Utilidades:

- Control Total: Ofrecen una personalización prácticamente completa, permitiendo a los programadores utilizar su propio código para modificar la interfaz del sistema y su funcionalidad. Esto posibilita la adaptación a diversos entornos de manera eficiente.

- Conectividad y Adaptabilidad: Estos periféricos pueden conectarse fácilmente a un ordenador para extraer datos, y su integración con otros dispositivos es extremadamente sencilla. Proporcionan una alta conectividad y adaptabilidad en entornos variados.

- Minimización de costes: Los dispositivos embebidos están compuestos por módulos electrónicos, lo que reduce significativamente su coste. Además, su mantenimiento es facilitado, ya que la sustitución de componentes es simple y económica.

- Diseño Modular: Los dispositivos embebidos son fácilmente trasladables, desmontables y reorganizables, lo que permite su integración en cualquier lugar y en cualquier otro sistema electrónico de manera eficaz.

- Tiempo de respuesta: Deben ejecutar acciones en lapsos de tiempo inmediatos, lo que resulta en un tiempo de respuesta extremadamente corto, especialmente en productos industriales que requieren operaciones rápidas.
- Accesibilidad: Los sistemas embebidos más simples se han vuelto accesibles para cualquier usuario, democratizando su uso y permitiendo que cualquier persona trabaje con ellos de manera fácil y eficiente.

## 2.1 Inversor de CD a CA

Los inversores, también conocidos como convertidores de corriente continua a corriente alterna (CD-CA), son dispositivos de potencia diseñados para transformar la corriente directa en corriente alterna. En un escenario ideal, un inversor debería recibir corriente continua como entrada desde una fuente de alimentación y generar en su salida una señal de voltaje senoidal pura con la frecuencia y amplitud deseadas.

Las aplicaciones comunes de los inversores de potencia incluyen:

- Control de la velocidad en motores de corriente alterna.
- Uso en sistemas de alimentación ininterrumpida (SAI).
- Funcionamiento de dispositivos de corriente alterna alimentados por batería.
- Aplicaciones como hornos de inducción, entre otras.

A continuación, se muestra el diagrama de un inversor:

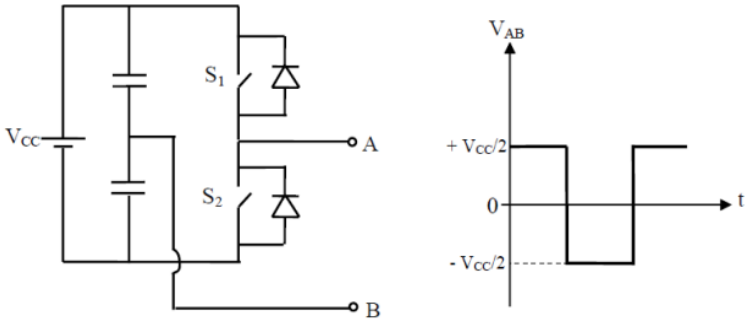


*Ilustración 1 Diagrama de un inversor*

## 2.2 Modulación PWM

La modulación por ancho o de pulso (en inglés pulse width modulation PWM) es un tipo de señal de voltaje utilizada para enviar información o para modificar la cantidad de energía que se envía a una carga. Esta acción tiene en cuenta la modificación del proceso de trabajo de una señal de tipo periódico.

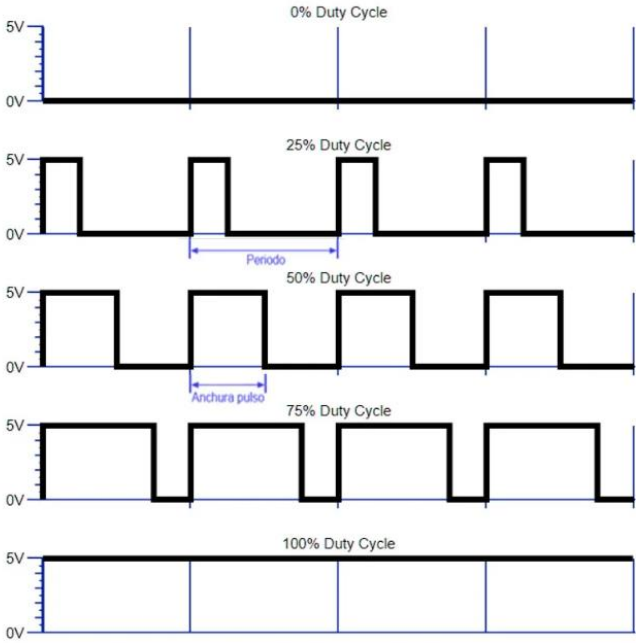


*Ilustración 2 Modulación PWM*

## 2.3 Teléfono Inteligente

Un teléfono inteligente es un dispositivo móvil avanzado que combina funciones de telefonía con capacidades computacionales, de conectividad y multimedia. Estos dispositivos están diseñados para ofrecer una variedad de servicios y aplicaciones más allá de las funciones básicas de llamadas telefónicas y mensajes de texto.

Las características distintivas de un teléfono inteligente incluyen acceso a Internet y la capacidad de realizar múltiples tareas. Los smartphones también suelen tener sensores, como acelerómetros y giroscopios, que permiten una interacción más intuitiva y diversas funciones, como la detección de movimiento.

## 2.4 Modem GSM

El Sistema Global para las Comunicaciones Móviles (Global System for Mobile Communications o GSM) es un estándar de comunicación móvil digital que facilita el envío y recepción de datos y señales de voz a través de la red. Su principal destacado reside en su accesibilidad y amplio alcance.

Este estándar posibilita la conexión de teléfonos a ordenadores, permitiendo realizar diversas tareas como enviar y recibir mensajes, correos electrónicos, navegar por internet, transmitir datos o SMS, entre otras funciones. El módulo GSM, reconocido por su seguridad, es ampliamente utilizado en diversos sectores y actualmente constituye el estándar más extendido a nivel mundial.

Desarrollado en 1982 para describir los protocolos de redes móviles digitales de segunda generación (2G), el GSM surgió para abordar los problemas de compatibilidad presentes en las redes hasta esa fecha. Los sistemas 2G, aunque compuestos por características básicas, son esenciales y seguros. La incorporación de seguridad en el sistema GSM lo posiciona como el estándar de telecomunicaciones más seguro disponible en su época. Las características del módulo GSM incluyen:

- Soporte para ISDN (Red Digital de Servicios Integrados).
- Lista telefónica.
- Funciones de alarma y reloj.
- Facilita la implementación de servicios adicionales.

- Número de marcación constante.
- Uso eficiente del espectro.
- Servicio de mensajes cortos.
- Conversaciones telefónicas cifradas.
    - Roaming Internacional.

La restricción en el ancho de banda es una limitación de la tecnología GSM, lo cual podría ocasionar interferencias de manera no predecible.

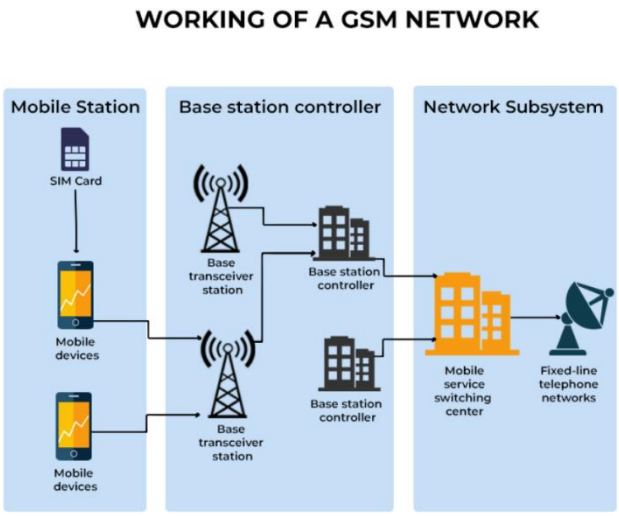


*Ilustración 3 Funcionamiento u operación de GSM*

## 2.5 Comandos Hayes (Comandos AT)

El lenguaje de comandos Hayes es un conjunto desarrollado por la empresa Hayes Communications con el propósito de gestionar diversos tipos de módems y asegurar su compatibilidad. Está compuesto por cadenas de texto cortas que se combinan para formar comandos completos destinados a operaciones como desconectar, marcar y ajustar los parámetros de conexión de los módems.

Este conjunto de instrucciones se ha convertido prácticamente en un estándar abierto para la configuración y parametrización de módems.

La presencia de los caracteres "AT" al inicio de todos los comandos, que representan "Atención", ha llevado a que este conjunto también se conozca como comandos AT. La mayoría de los módems para computadoras personales siguen las especificaciones del conjunto de comandos AT.

Un dispositivo que implementa el conjunto de comandos Hayes se considera compatible con Hayes. Parte de estos comandos fue incorporada por la ITU-T (Unión Internacional de Telecomunicaciones - Telecomunicación) en el protocolo V.25ter, actualmente V.250. La adopción de este estándar eliminó la necesidad de desarrollar controladores específicos para diferentes tipos de módems.

Los comandos Hayes se clasifican en cuatro grupos distintos:

1. **Básicos (AT…):**

   Estos comandos fueron los primeros en definirse y desempeñan funciones fundamentales.

2. **De registro (ATSi=, o ATSi?):**

   Estos comandos alteran los valores de los registros internos del módem o solicitan información sobre sus valores.

3. **Extendidos (AT&..):**

   Comandos adicionales que se añadieron posteriormente a las definiciones de los comandos básicos. Por lo general, cumplen funciones un poco más complejas que los comandos básicos.

4. **De Propiedad (AT/…):**

   Estos son definidos por el fabricante del equipo. Cuando se envía cualquier comando al módem, este responde con el resultado de la operación, como "OK", "ERROR", "CONNECT", entre otros.

## 2.6 Protocolo de comunicación UART

Un protocolo de comunicación es un conjunto de reglas y convenciones que define cómo se deben llevar a cabo la transmisión y recepción de datos entre dos o más

dispositivos para que la comunicación sea exitosa. Estas reglas establecen el formato de los mensajes, la secuencia de acciones que deben llevarse a cabo, y las respuestas esperadas en diferentes situaciones.

El protocolo de comunicación UART (Universal Asynchronous Receiver/Transmitter) es un estándar de transmisión de datos que ha sido fundamental en la interconexión de dispositivos electrónicos. A diferencia de otros protocolos, como el I2C o el SPI, el UART no requiere un reloj compartido entre los dispositivos conectados. En cambio, se basa en la transmisión y recepción de datos de forma asincrónica.

La comunicación UART implica dos líneas principales: la línea de transmisión de datos (TX) y la línea de recepción de datos (RX). Estas líneas permiten la transferencia de información en ambos sentidos entre dos dispositivos, estableciendo así una conexión punto a punto. La asincronía en la comunicación significa que no hay un reloj de sincronización compartido entre los dispositivos; en cambio, ambas partes deben acordar de antemano la velocidad de transmisión (baud rate) para asegurar una comunicación eficiente.

Los datos en el UART se transmiten en la forma de tramas:

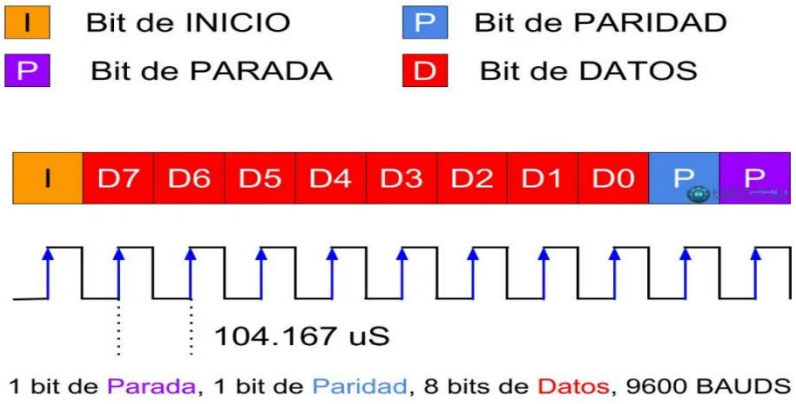


*Ilustración 4 Trama de protocolo UART*

## 3. Definición del problema

En áreas remotas donde no hay acceso constante a la red eléctrica, los residentes enfrentan el desafío de mantener sus teléfonos celulares cargados para la

comunicación esencial. Además, es crucial contar con un sistema de alerta en caso de emergencias, como tormentas o eventos naturales.

Es por eso por lo que desarrollamos un sistema que integre un inversor de corriente continua a corriente alterna (CD-CA) con paneles solares y almacenamiento de energía, como baterías recargables. Este sistema proporcionaría la capacidad de cargar teléfonos celulares mediante tomas de corriente USB, permitiría el envío y recepción de llamadas y mensajes, y ofrecería funciones de alarma para emergencias, como sirenas o luces intermitentes.

La solución no solo resolvería el problema de la carga de teléfonos en áreas remotas, sino que también brindaría un medio para mantener a los residentes informados y seguros durante situaciones de emergencia.

# 4.Descripción del sistema utilizado

## 4.1 Material utilizado:

- Tableta Protoboard.
- Tarjeta de Desarrollo TIVA TM4C1294NCPDT.
- Teclado Matricial 4x4.
- Display LCD 16x2.
- 1 potenciómetro de 1 K[Ω].
- 8 resistencias de 2.2 K[Ω] a ¼ de Watt.
- 8 resistencias de 3.3 K[Ω] a ¼ de Watt.
- 1 Resistencia de 4.7 [Ω] a ½ de Watt.
- Módulo SIM 808 GSM GPRS con GPS.
- Eliminador De Voltaje 5v / 2a, Jack 2.1mm - Fuente De Poder.
- Tarjeta SIM Telcel (debe ser 2G).
- 2 cables auxiliares de Audio Jack 3.5mm.
- 1 micrófono Electre.
- 1 potenciómetro de 5 K[Ω]
- Tarjeta Fenólica Perforada de 8x12 cm.
- 2 borneras de 3 Entradas.
- 2 bases Dip de 8 pines.
- Headers macho.
- Headers Hembra.
- 1.5 m de cable calibre 12.
- Cable calibre 22.

- 2 resistencias de 220 [Ω] a 1 W.
- 2 optoacopladores 6n135.
- 2 resistencias de 47 K[Ω].
- 2 reguladores de Voltaje 7805.
- 2 capacitores de 0.1 [µF].
- 2 resistencias de 1 K[Ω] a ½ [W].
- 2 transistores 2N222A.
- 2 diodos NTE4920.
- 2 transistores IRFP064V.
- 1 transformador de 128 V a 24 V.
- 1 socket con foco.

Equipo de cómputo:

- PC o Laptop.

Equipo de laboratorio:

- Fuente de Voltaje.
- Osciloscopio.

## 4.2 Descripción del hardware del inversor:

Para comprender el funcionamiento del inversor diseñado, es esencial comprender el funcionamiento de una fuente conmutada.

Una fuente conmutada opera al alternar el encendido y apagado de un transistor, llevándolo a su región de saturación y posteriormente a su región de corte, según si se le suministra un nivel lógico alto o bajo.

Utilizando el mismo funcionamiento de una fuente conmutada se diseñó un inversor, donde se utilizó modulación PWM con el fin de controlar el ciclo de trabajo, esto para controlar la tensión que recibe la carga. Como requisito se necesita tener una frecuencia de 60 Hz, ya que es la frecuencia a la que opera el Sistema Eléctrico Nacional.

En el esquema, podemos observar la presencia de un optoacoplador. Este componente se empleó para aislar la sección de potencia de la sección electrónica, evitando daños a los circuitos electrónicos, como ocurre en este caso con la TIVA TM4C1294NCPDT.

Como sugiere su nombre, el optoacoplador opera con principios ópticos. Al pasar corriente a través del diodo emisor de luz infrarroja incorporado, emite una señal infrarroja, lo que provoca que el transistor TBJ cambie de la región de corte a la región de saturación, permitiendo así el flujo de corriente entre el colector y el emisor.

Dado que el transistor interno está activado, esto ocasionará que el transistor 2n2222A se desactive, resultando en la activación del MOSFET. En contraste, cuando el LED infrarrojo no emite señal, el transistor interno entra en la región de corte. En consecuencia, la corriente fluye a través de la base del transistor 2n2222A, lo que lo activa. Sin embargo, no se logrará polarizar el MOSFET IRFP064V, resultando en la desactivación del MOSFET.

El diodo Zener presente en el esquema se coloca estratégicamente para prevenir daños al circuito causados por la elevada corriente proveniente del transformador utilizado; este diodo Zener cumple la función de supresor de transitorios. El capacitor mostrado en el esquema actúa como una especie de reserva de energía similar a una batería, suministrando los 5V necesarios para polarizar el transistor T2. Asimismo, el regulador de voltaje presente en el esquema se utiliza para reducir los 15V provenientes del transformador, proporcionando así una salida de 15V de corriente continua.

Cuando conectamos el devanado secundario de nuestro transformador, obtendremos en el devanado primario un voltaje que varía en función del ciclo de trabajo del PWM, y podemos ajustar este voltaje gracias a la presencia de un potenciómetro.

### 4.2.1. Diagrama de bloques del hardware del inversor:

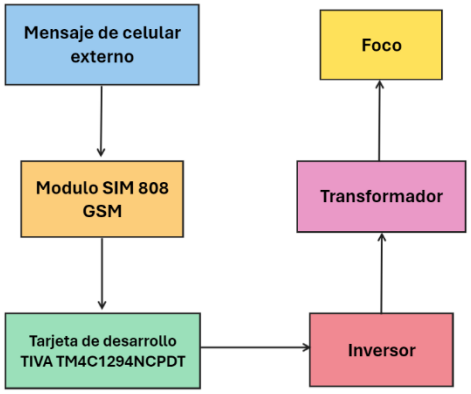


*Ilustración 5 Diagrama de bloques de hardware de inversor*

### 4.2.2 Diagrama esquemático

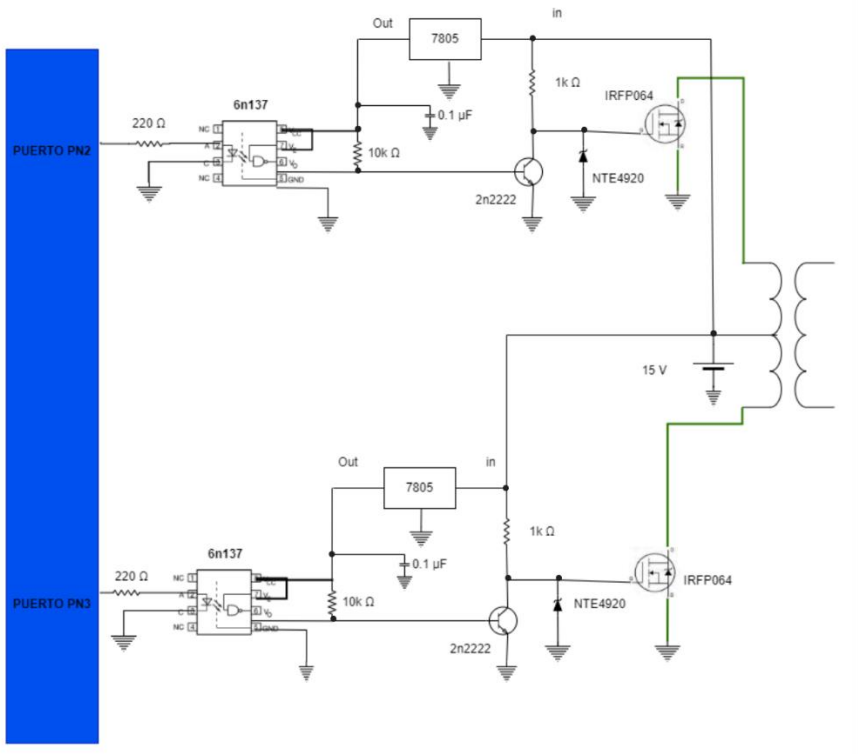


*Ilustración 6 Diagrama de conexiones de inversor*

Notas:

1. Todos los cables de color verde son de calibre 18.

2. Las tierras son comunes excepto las conectadas a los pines 4 de los optoacopladores, las cuales son GND de la tarjeta de desarrollo.

### 4.2.3 Software utilizado solo para parte inversor

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


void iniports();
void initADC();
uint16_t readPotentiometer();

uint8_t arreglo[12][25]={{0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},//menor
                         {0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0},
                         {0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0},
                         {0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0},
                         {0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0},
                         {0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0},
                         {0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0},
                         {0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},
                         {0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0},
                         {0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0},
                         {0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0},//mayor
                         {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0}};//Arreglo de
lobulos


uint8_t cont, contpot, renglon, pol=0; // contador  general, contador potencia, pol-
>polaridad
uint8_t maskb2= 0b00000100;
uint8_t maskb3= 0b00001000;
uint16_t potenciometro;
float lec_pot;
unsigned int pot;

int main(void)
{
    iniports();
    initADC();
    for(;;){
```

```
        // Lectura
        potenciometro = readPotentiometer();
        renglon=(int)potenciometro*12.00/4095.00;
        if(renglon==12){
           renglon=11;
        }
        // Utiliza el valor de lectura

        for(contpot=0; contpot<25; contpot++){
           if(arreglo[renglon][contpot]==1 && pol==1){
              GPIO_PORTF_AHB_DATA_R|=maskb2;
           }
           else if(arreglo[renglon][contpot]==0 && pol==1){
              GPIO_PORTF_AHB_DATA_R&=~maskb2;
           }
           else if(arreglo[renglon][contpot]==1 && pol==0){
              GPIO_PORTF_AHB_DATA_R|=maskb3;
           }
           else if(arreglo[renglon][contpot]==0 && pol==0){
              GPIO_PORTF_AHB_DATA_R&=~maskb3;
           }
           SysCtlDelay(1750);
        }
        pol^=0x01;
     }
     return 0;
}


void iniports(){
     SYSCTL_RCGCGPIO_R |= 0X30; //0000.0000.0011.0000 Habilita el puerto F y
E
     while ((SYSCTL_PRGPIO_R&0X30)==0);

     //BITS 2 Y 3 COMO SALIDAS
     GPIO_PORTF_AHB_DATA_R|=0X00;
     GPIO_PORTF_AHB_DEN_R|=0X0C;
     GPIO_PORTF_AHB_DIR_R|=0X0C;
     GPIO_PORTE_AHB_DATA_R|=0X00;
     GPIO_PORTE_AHB_DIR_R&=~0x10;
     GPIO_PORTE_AHB_DEN_R|=0x10;
}

void initADC()
{
```

```
    SYSCTL_RCGCGPIO_R |= (1<<4);
    SYSCTL_RCGCADC_R |= (1<<0);

    GPIO_PORTE_AHB_AFSEL_R |= (1<<3);
    GPIO_PORTE_AHB_DEN_R |= ~(1<<3);
    GPIO_PORTE_AHB_AMSEL_R |= (1<<3);

    ADC0_ACTSS_R &= ~(1<<3);
    ADC0_EMUX_R &= ~0XF000;
    ADC0_SSMUX3_R =0;
    ADC0_SSCTL3_R |=(1<<1)|(1<<2);
    ADC0_ACTSS_R |=(1<<3);

    SYSCTL_RCGCGPIO_R |= 0X1000;
    SYSCTL_PLLFREQ0_R |= SYSCTL_PLLFREQ0_PLLPWR;

    while((SYSCTL_PLLSTAT_R&0X01)==0);

    SYSCTL_PLLFREQ0_R &= ~SYSCTL_PLLFREQ0_PLLPWR;
}

uint16_t readPotentiometer()
{
    ADC0_PSSI_R |= (1<<3);
    while((ADC0_RIS_R & 8)==0);
    unsigned int pot=ADC0_SSFIFO3_R;
    ADC0_ISC_R=8;
    return pot;
}
```

### 4.2.4 Resultado inversor individualmente

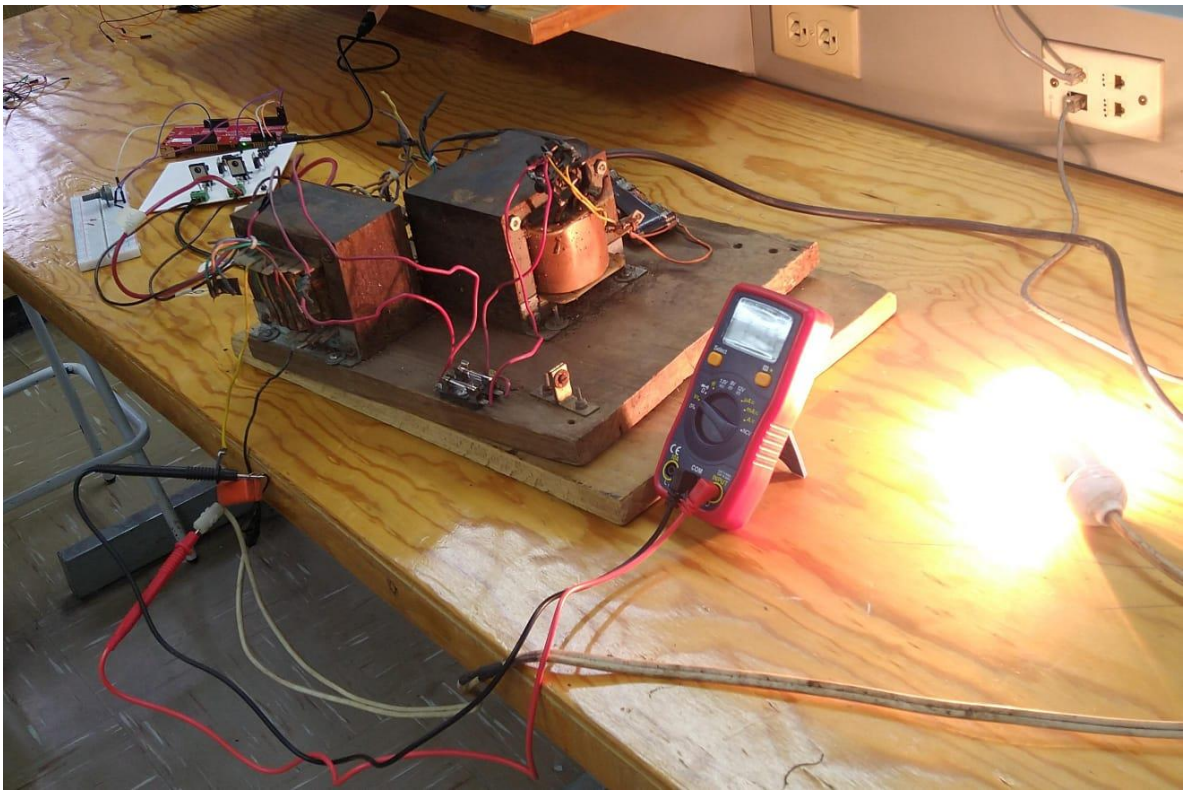

*Ilustración 7 Inversor individualmente*

## 4.3 Descripción del hardware del teléfono(Proyecto completo. Teléfono e Inversor)

El proyecto realizado consiste en la integración de un Módem SIM 808 GSM GPRS con GPS, el cual es gestionado por la Tarjeta de Desarrollo Tiva TM4C1294NCPDT. Esta conexión se establece mediante tres cables:

- Rx: el cual transmite información desde el módem hacia la Tarjeta mediante el protocolo UART.
- Tx: el cual transmite información desde la Tarjeta hacia el módem utilizando el protocolo UART.
- GND: Representa la tierra común.

Se utiliza un teclado matricial para escribir los diferentes comandos que controlaran el celular.

De igual manera, la Tarjeta de Desarrollo está conectada a un Display LCD 16x2, el cual permite la visualización de las distintas operaciones llevadas a cabo por el módem, como realizar llamadas, enviar y recibir llamadas y mensajes. La luminosidad de la pantalla del Display LCD 16x2 se puede ajustar mediante un potenciómetro conectado directamente al Display.

### 4.3.1 Diagrama de bloques del hardware completo

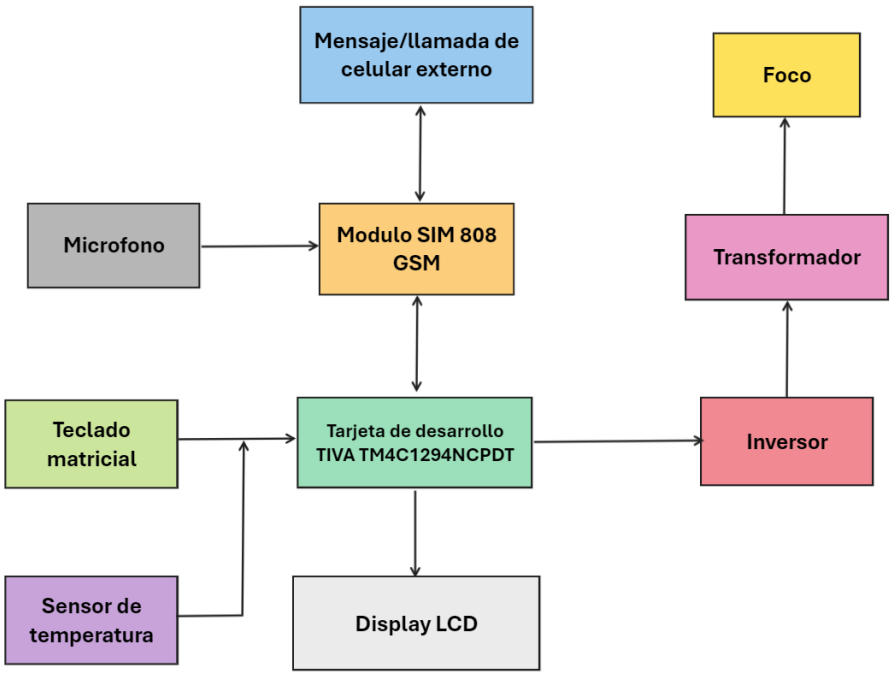


*Ilustración 8 Diagrama de bloques de hardware de celular incluyendo ya el inversor*

### 4.3.2 Diagrama Esquemático

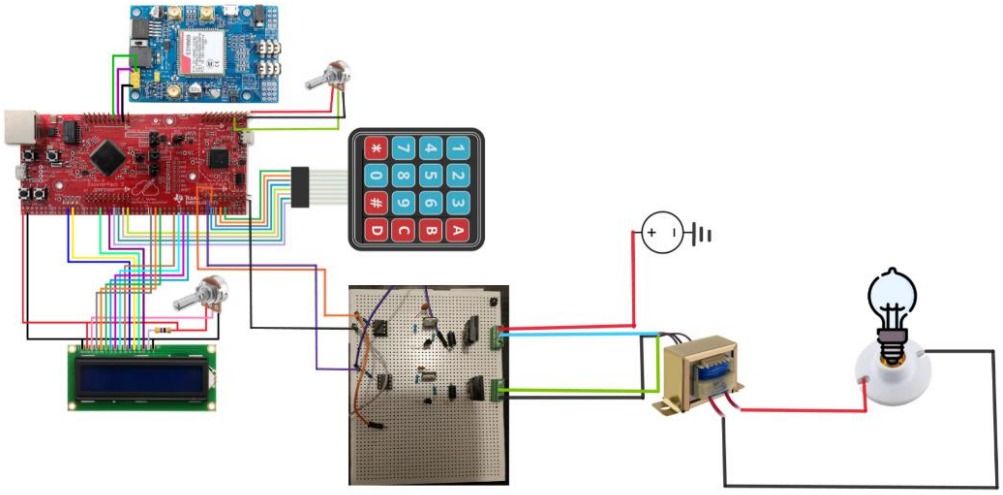

*Ilustración 9 Diagrama esquemático del proyecto completo, celular e inversor*

## 4.4 Descripción de software del proyecto completo:

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

*Ilustración 10 Librerías*

Se incluyen las librerías del microcontrolador , entre ellos la carpeta Tivware ya que se hacen uso de por ejemplo interrupciones dentro del código.

```
7 int numt[10],
8     numt1[10]={'5','5','4','8','8','0','2','1','8','8'};//CELULAR PREDEFINIDO
```

*Ilustración 11 Definición celular*

Nombramos nuestro celular predefinido al cual se le llamara en una función cuando este se utilice en el ADC o en el mensaje predefinido.

```
uint8_t d_uint8Dato;
int CM, entrante, i, j, dato, ren, col, pantalla, z, recibe, llamada_m,k,edo=0,potencia=0x30, pulso=0,delay,aux,limite,
char Rx[80],mensaje_mem[80];
char envio[32];



char recibido[32],error[5]={'E','R','R','O','R'},puerta[13]={'P','U','E','R','T','A',' ','E','N',' ','U','S','O'},
        apertura[19]={'P','U','E','R','T','A ',' ','A','B','I','E','R','T','A',' ',' ',' ',' ',' ',' '},
        cerradura[10]={'C','E','R','R','A','D','A',' ',' ',' '};

unsigned char Valcon,Temp,Temp2;
char var, pvar;
```

*Ilustración 12 Variables*

Definiciones de variables en general, varias de ellas se utilizan en diferentes partes por lo que se hacen globales.

```
//DEFINICION DE VALORES HEXADECIMALES
int hexa [4][4]={{0x31 , 0x32, 0x33, 0x41},
                 {0x34, 0x35, 0x36, 0x42},
                 {0x37, 0x38, 0x39, 0x43},
                 {0x2A, 0x30 , 0x23, 0x44}};

//DEFINICION DE CARACTERES TECLADO
char numeros [4][4]={{'1' , '2', '3', 'A'},
                     {'4', '5', '6', 'B'},
                     {'7', '8', '9', 'C'},
                     {'*', '0' , '#', 'D'}};

//VALORES ASOCIADOS A TECLADO NUMERICO
char letras[10][5] = {
    {' '},      // 0
    {'1'},      // 1
    {'A', 'B', 'C', '2'},      // 2
    {'D', 'E', 'F', '3'},      // 3
    {'G', 'H', 'I', '4'},      // 4
    {'J', 'K', 'L', '5'},      // 5
    {'M', 'N', 'O', '6'},      // 6
    {'P', 'Q', 'R', 'S', '7'}, // 7
    {'T', 'U', 'V', '8'},      // 8
    {'W', 'X', 'Y', 'Z', '9'}  // 9
};
```

*Ilustración 13 Declaración matrices*

Se hace una matriz para definir los valores asociados a cada tecla, tanto la parte hexadecimal así como los diferentes valores en el caso del teclado para mensajería.

```
// ARREGLO PARA INVERSOR , LA MODULACIÓN SE LIMITA A 3 , YA QUE UNA MAYOR PODRIA EMNTREGARNOS VO
//ALGO QUE VIMOS EN EL DESARROLLO DEL CURSO
const int p [11][21]= {{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0},
                       {0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},
                       {0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0},
                       {0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0},
                       {0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0},
                       {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0}};
```

*Ilustración 14 Parte inversor*

El siguiente paso nuevamente es de definición , en este caso definimos los lóbulos tal como en el primer caso del curso , se divide en esa cantidad de veces para poder reducir el voltaje a su salida ya que este dependerá directamente de los lóbulos, en nuestro caso, se utilizan 4 modos, los cuales son los primeros cuatro arreglos , ya que de ocuparse los de mayor cantidad provocaría requerir un mayor factor de potencia y por ende mayor voltaje de salida

```
//FUNCIONES

void LCD(void);
void BUSY (void);
void BUSY1 (void);
void _E(void);
void data (unsigned char c);
void comando (unsigned char d);
void CLS(void);
void BYTEDIS(unsigned char DByte);
void HOME(void);
void LEECON(void);
void cfg(void);
void winnum(void);
void salida(void);


char UART6Tx(char dato);
void escribe_pantalla(void);

void INIM();
void mensaje();
void recibe_mensaje();
void limpia_d();
void envia_mensaje();
void adc(void);
```

*Ilustración 15 Funciones*

Se definen las diferentes funciones que se utilizaran a lo largo del proyecto.

```
void LCD(void)  //INICIALIZA EL DISPLAY A DOS LINEAS, 4 BITS DE INTERFAZ, CURSOR Y DESTELLO.
{
    //PRIMER METODO DE INICIALIZACION
    PORTCON=0X00;
    PORTDIS=0X38; // **D7-D4 = 0011

    SysCtlDelay(533000);                //Delay 100ms
    _E();                              //toggle E
    SysCtlDelay (53000);                  //10ms
    _E();                              //toggle E
    SysCtlDelay (53000);                  //10ms
    ;PORTDIS=0X38;
    _E();                              //toggle E
    SysCtlDelay (53000);                  //10ms
    comando(0X39);   //8 BITS , 2 LINEAS, 5X7

    comando(enciende);  // ENCIENDE DISPLAY CON CURSOR Y DESTELLO


    comando(CLR); // BORRAR DISPLAY

    comando(0X07); // AL ESCRIBIR EL CURSOR SE INCREMENTA Y SE DESPLAZA
    comando(0X38); // //8 BITS , 2 LINEAS, 5X7 REQUERIDO POR LOS DISPLAYS CLONES
}
```

*Ilustración 16 Función display*

Una de las primeras es la inicialización del display.

La cual depende de otras dos.

```
void comando (unsigned char d)// ENVIA UN COMANDO AL REGISTRO DE CONTROL DEL LCD
{

    PORTDIS = d;               //envia CMD al LCD
    bitclr(PORTCON,RS);              //direcciona registro de controldel  LCD
    _E();                          //toggle E
    BUSY1(); //VERIFICA LA BANDERA DE BUSY
    }
```

*Ilustración 17 Función comando*

Comando en donde se direcciona registro de control y se verifica bandera.

```
void BUSY1(void)
    {SysCtlDelay(10066);}
```

*Ilustración 18*

```
1166 void _E(void)  //GENERA UN PULSO DE 1.25 uS EN LA TERMINAL E DEL DISPLAY
1167 {
1168     bitset(PORTCON,E);        // E=1
1169     SysCtlDelay (3);  //retraso de 500 nS
1170     bitclr(PORTCON,E);  // E=0
```

*Ilustración 19*

```
void data (unsigned char c)  //ESCRIBE UN DATO ALFANUMERICO AL DISPLAY
{
    //ENVIA dato
    PORTDIS = c;

    bitset(PORTCON,RS);    //ENVIA A REGISTRO DE DATOS
    _E();                 //TOGGLE _E()
    BUSY1(); //PREGUNTA POR LA BANDERA DE BUSY
    bitclr(PORTCON,RS);    //niveles de control a 0
    }
```

*Ilustración 20 Función data*

Finalmente con data podemos seleccionar el valor alfanumérico a partir de valores en hexadecimal, en el código se empleará repetidamente este para el muestreo en display.

```
1096 void HOME(void){
1097     pantalla=1;
1098     comando(CLR);
1099
1100     if(edo==0){
1101             comando(0X80);
1102
1103             data(0X00); //nulo
1104             data(0x45); //E
1105             data(0x53); //S
1106             data(0x54); //T
1107             data(0x41); //A
1108             data(0x44); //D
1109             data(0x4F); //O
1110             comando(0x87); //
1111             data(0x4F); //O
1112             data(0X4E); //N
1113             comando(0x8A); //
1114             data(0x4D); //M
1115             data(0x3D); //=
1116             data(potencia); //PULSOS
1117             comando(0x8E); //
1118             data(0x20); //
1119        }else{
1120             comando(0X80);
1121             data(0X00); //nulo
1122             data(0x45); //E
1123             data(0x53); //S
1124             data(0x54); //T
1125             data(0x41); //A
1126             data(0x44); //D
1127             data(0x4F); //O
1128             comando(0x87); //
1129             data(0x4F); //O
```

*Ilustración 21 Parte de función home*

Dentro de la función home utilizamos data para el muestreo de información en display en este caso empleamos el estado de nuestro inversor y además de ello el nivel del modulación en el que este se encuentra

```
void cfg(){
    // Habilitar el reloj del puerto  A, H, J, K, M, N, P, y Q
    SYSCTL_RCGCGPIO_R |= 0X7B81;
    k=123;
    //CONFIGURA EL TIMMER 3
    SYSCTL_RCGCTIMER_R = 0x08; //ACTIVA RELOJ TIMER 3
    k=1234; //TIEMPO PARA ESTABILIZAR RELOJES
    //CONFIGURAMOS TIMER 3 SUBTIMER A
    TIMER3_CTL_R = 0X0; //DESCATIVA EL TIMER A UTILIZAR
    TIMER3_CFG_R = 0X04; //ACTIVAMOS LA FUNCION DE 16 BITS DEL TIMER
    TIMER3_TAMR_R = 0X02; //CUENTA HACIA ABAJO
    TIMER3_TAILR_R = 0X18CD; //CUENTA PRECARGADA
    TIMER3_ICR_R = 0x1; //LIMPIA BANDERA DE INTERRUPCION
    TIMER3_IMR_R = 0X01; //TIMEOUT
    TIMER3_TAPR_R = 0X0; //PRESCALADO
    NVIC_EN1_R= 1<<(35-32); //ACTIVA INTERRUPCION TIMER 3
    TIMER3_CTL_R |= 0X01; //ACTIVA TIMER

    //CONFIGURAMOS PUERTO N (INVERSOR)
    GPIO_PORTN_DIR_R = 0X0C; //ACTIVAMOS LA salida DIGITAL PN2 y PN3
    GPIO_PORTN_DEN_R = 0X0C;  //HABILITAMOS PN2 y PN3

    //Configura pines del puerto K y M  (PANTALLA)
    GPIO_PORTK_DATA_R=0X00;
    GPIO_PORTK_DEN_R=0XFF;
    GPIO_PORTK_DIR_R=0XFF;
    GPIO_PORTM_DATA_R|=0X00;
    GPIO_PORTM_DEN_R |=0X07;
    GPIO_PORTM_DIR_R |=0X07;
    // Configurar pines del puerto H y Q(TECLADO)
    GPIO_PORTH_AHB_DIR_R |= 0x0F;
    GPIO_PORTH_AHB_DEN_R |= 0X0F;
    GPIO_PORTQ_DIR_R &= ~0X0F;
    GPIO_PORTQ_DEN_R |= 0X0F;
    GPIO_PORTQ_PUR_R |= 0X0F;
```

*Ilustración 22 Función cfg*

Para el caso de la función CFG se trata de como su nombre lo indica toda la parte de configuración, la habilitación de puertos para todo el proyecto en este caso de trata de los relojes de los puertos se configura el timer , los puertos n(pn2 y pn3) para el caso del inversor, k y m para el display

Y el h y q para el teclado matricial

```
//Priorirdad de control de interrupciones
NVIC_PRI21_R = (NVIC_PRI21_R&0x00000000)|0xFFFFFFFF;
NVIC_EN1_R |=0x80000;
NVIC_EN2_R =0xF00000;          //habilita la interrupci n Q0-Q3 respectivo a 84,85,86 y 87 en

SYSCTL_RCGCUART_R |=0X00C1; //HABILITAR UART0,UART6, UART6
UART0_CTL_R &=~0X0001;  //DESHABILITAR UART
UART0_IBRD_R=26; //IBDR=int(48000000/16*115200))= 26.0416
UART0_FBRD_R =2 ; //FBRD= 0.1267*64 =2
UART0_LCRH_R =0X0060; //8 BITS, HABILITAR FIFO 70  y desbilita 60
UART0_IM_R |=0x10;
NVIC_EN0_R =0x20;  //es la posicion E0 en el espacio 0
UART0_CTL_R= 0X0301 ; //HABILITAR RXE, TXE Y UART


UART6_CTL_R &=~0X0001;  //DESHABILITAR UART
UART6_IBRD_R = 26; //IBDR=int(48000000/16*115200))= 26.0416
UART6_FBRD_R =2 ; //FBRD= 0.1267*64 =2
UART6_LCRH_R =0X0060; //8 BITS, HABILITAR FIFO 70  y desbilita 60
UART6_IM_R |=0x10;
NVIC_EN1_R |=0x8000000;
UART6_CTL_R= 0X0301 ; //HABILITAR RXE, TXE Y UART


GPIO_PORTA_AHB_PCTL_R = (GPIO_PORTA_AHB_PCTL_R&0XFFFFFF00)+0X00000011; //UART0
GPIO_PORTA_AHB_AMSEL_R &= ~0X03; //DESHABILITAR FUNCION ANLOGICA EN PA0-1
GPIO_PORTA_AHB_AFSEL_R |= 0X03; //HABILITAR FUNCION ALTERNA EN PA0-1
GPIO_PORTA_AHB_DEN_R |= 0X03; //HABILITAR FINCION I/O DIGITAL

GPIO_PORTP_PCTL_R = (GPIO_PORTP_PCTL_R&0XFFFFFF00)+0X00000011; //UART6
GPIO_PORTP_AMSEL_R &= ~0X03; //DESHABILITAR FUNCION ANLOGICA EN PP0-1
GPIO_PORTP_AFSEL_R |= 0X03; //HABILITAR FUNCION ALTERNA EN PP0-1
GPIO_PORTP_DEN_R |= 0X03; //HABILITAR FINCION I/O DIGITAL
```

*Ilustración 23 cfg*

Se habilita los puertos UART y se utilizan las interrupciones para algunas de estas

```
void adc(){
   //CONFIGURACION
    SYSCTL_RCGCADC_R|=0XF; //HABILITA RELOJ ADC0
    SYSCTL_RCGCGPIO_R|=0x10; //HABILITA PE
    b=123; //TIEMPO DE ESTABILIZACI N DE RELOJES
    //CONFIGURA PUERTO E
    GPIO_PORTE_AHB_AFSEL_R = 0x01; //FUNCION ALTERNA PE0
    GPIO_PORTE_AHB_DEN_R = 0x0; //SE APAGA FUNCION DIGITAL PE0
    GPIO_PORTE_AHB_AMSEL_R =0x0F; //HABILITA ENTRADA ANALOGICA
    //CONFIGURACION DEL CONVERTIDOR ADC0 SECUENCIADOR 1
    ADC0_SSPRI_R = 0x02103; //SECUENCIADOR 3 ES EL DE MAYOR PRIORIDAD
    ADC0_ACTSS_R = 0x0; //APAGA SECUENCIADOR 1
    ADC0_EMUX_R = 0x0; //DISPARO POR SOFTWARE
    ADC0_SSMUX1_R = 0x3210;
    ADC0_SSEMUX1_R = 0x0;
    ADC0_SSCTL1_R = 0x2000;
    ADC0_ACTSS_R = 0x02; //HABILITA SECUENCIADOR 1
}
```

*Ilustración 24 Función adc*

En la siguiente se realiza la configuración y habilitación para el adc0.

```
int main(void){
    uint32_t g_ui32SysClock;
    g_ui32SysClock = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |SYSCTL_OSC_MAIN |SYSCTL_USE_PLL |SYSCTL_CFG_VCO_480), 48000000);
    adc();
    cfg();
    LCD();
    HOME();
    INIM();
    SysCtlDelay(53300);
    while(1){

        ADC0_PSSI_R = 0X02; //INICIA CONVERSION EN SECUENCIADOR 1
             while((ADC0_ACTSS_R & 0x10000)==0x10000);
             lecturaPE3=ADC0_SSFIFO1_R;
             lecturaPE2=ADC0_SSFIFO1_R;
             lecturaPE1=ADC0_SSFIFO1_R;
             lecturaPE0=ADC0_SSFIFO1_R;
             a=(lecturaPE0)/54;
             if(Rx[3]=='G'&&i==6||Rx[10]== 0xA &&i==11||Rx[3]=='E'&&Rx[4]=='G'&&i<23) {
                       i=0;
                       z=0;
                       pantalla=1;
                   }
             if (a>27&&tc==0){
```

*Ilustración 25 Función principal*

En la imagen superior vemos una pequeña parte de nuestra función principal , en la cual hacemos uso de varias de las funciones previamente declaradas , principalmente de inicialización como LCD y cfg para configurar los puertos.

```
            case'A':
                edo=1;
                break;

            case 'E':
                edo=0;
                break;

            case 'I':
                if (pulso<=3){
                    pulso++;
                }
                break;
            case 'R':
                if (pulso>0){
                    pulso=pulso-1;
                }
                break;
            }
            switch (pulso){
                case 0:
                    potencia=0x30;
                    break;
```

*Ilustración 26 Casos inversor*

La imagen nos muestra parte del código de la función UART6 en la cual podemos ver parte referente al uso del inversor y en este caso la configuración según la letra recibida en el mensaje será la responsable del estado que siga el inversor.

```
void tecladoQ(void){
```

*Ilustración 27 Función teclado*

Finalmente una función a recalcar será la de teclado con la cual podremos ir manipulando las pantallas a partir del teclado siguiendo la guía de matrices previamente definida y mostrada en este mismo reporte.

### 4.4.1 Diagrama de flujo del software

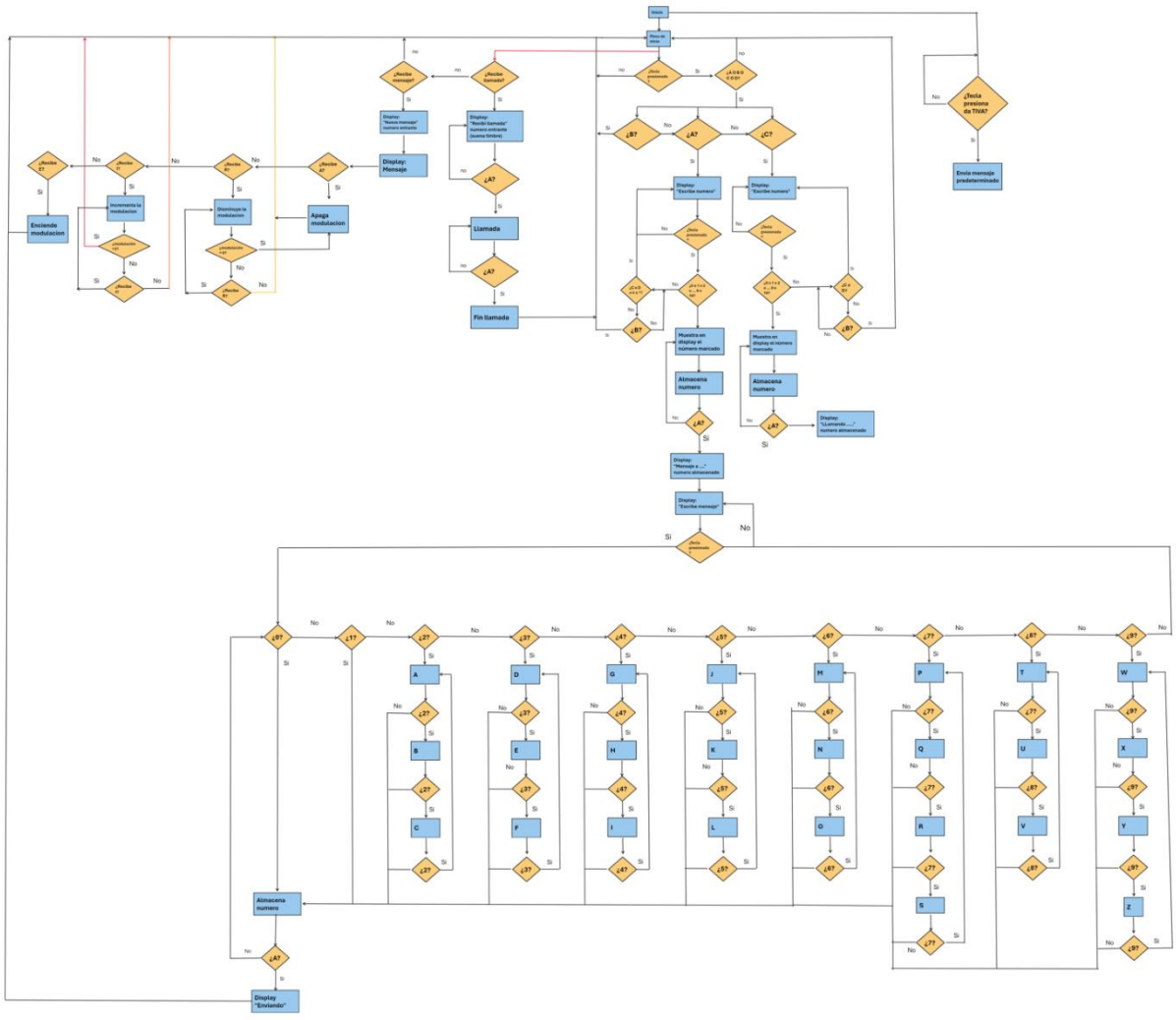

*Ilustración 28 Diagrama de bloques software*

A continuación, se muestra parte por parte el diagrama para una mejor apreciación:

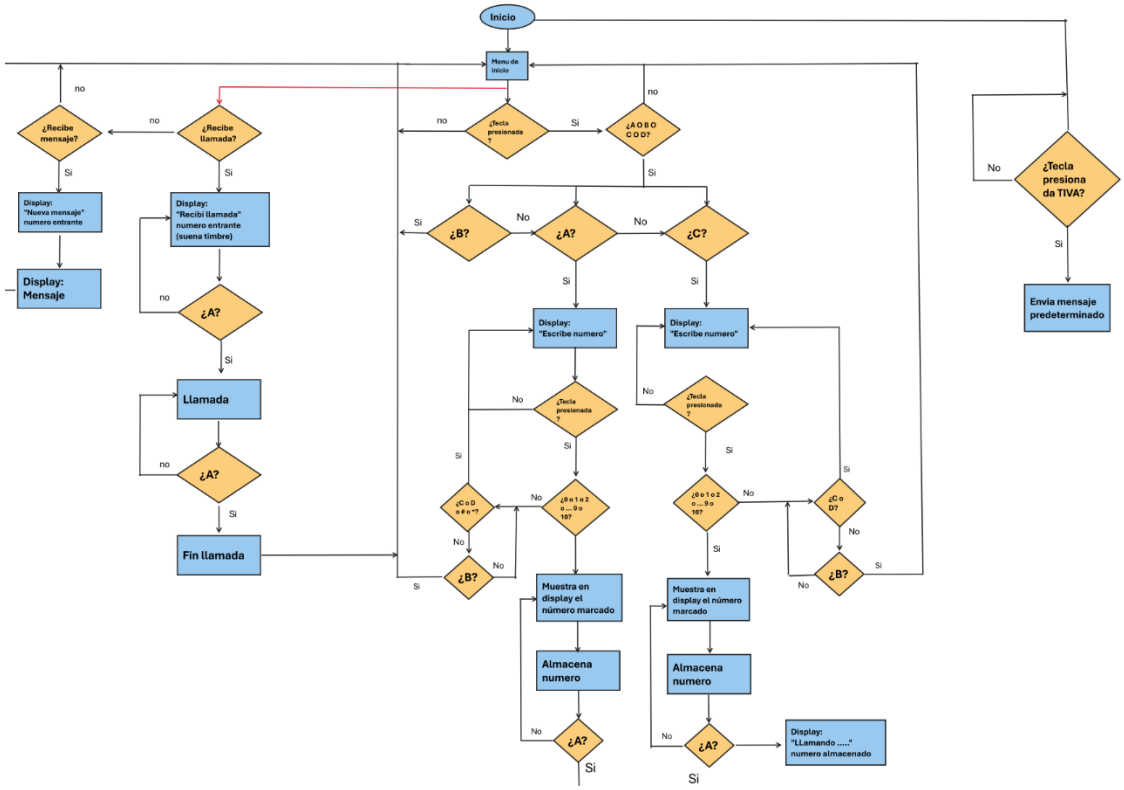


*Ilustración 29*

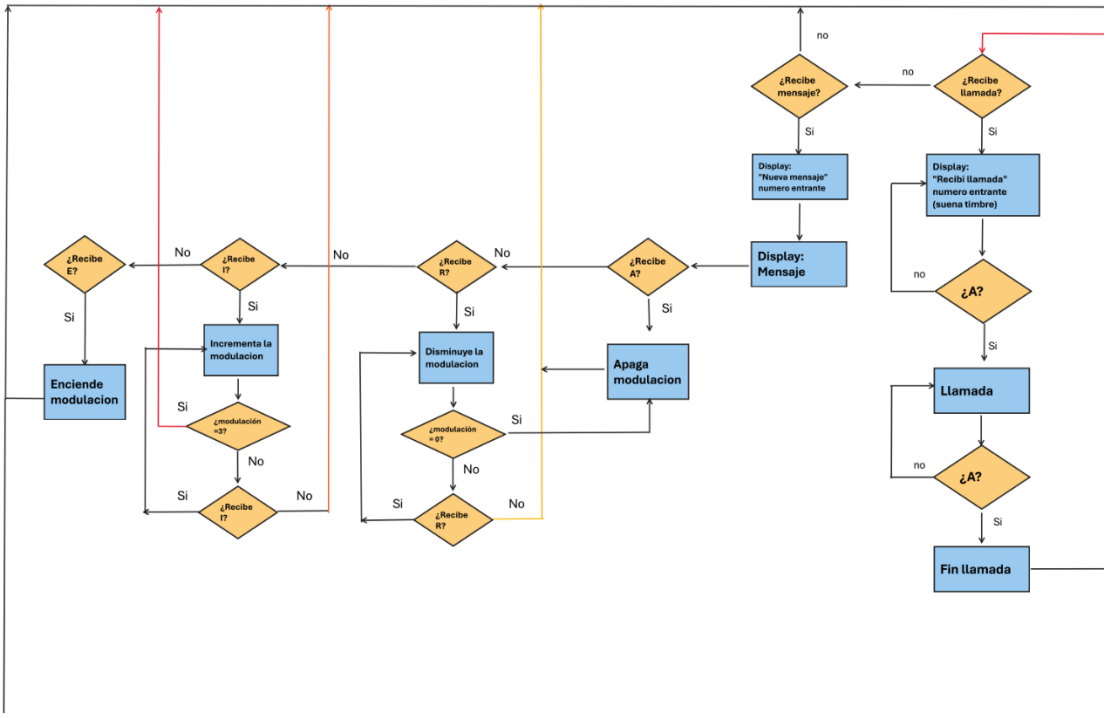


*Ilustración 30*

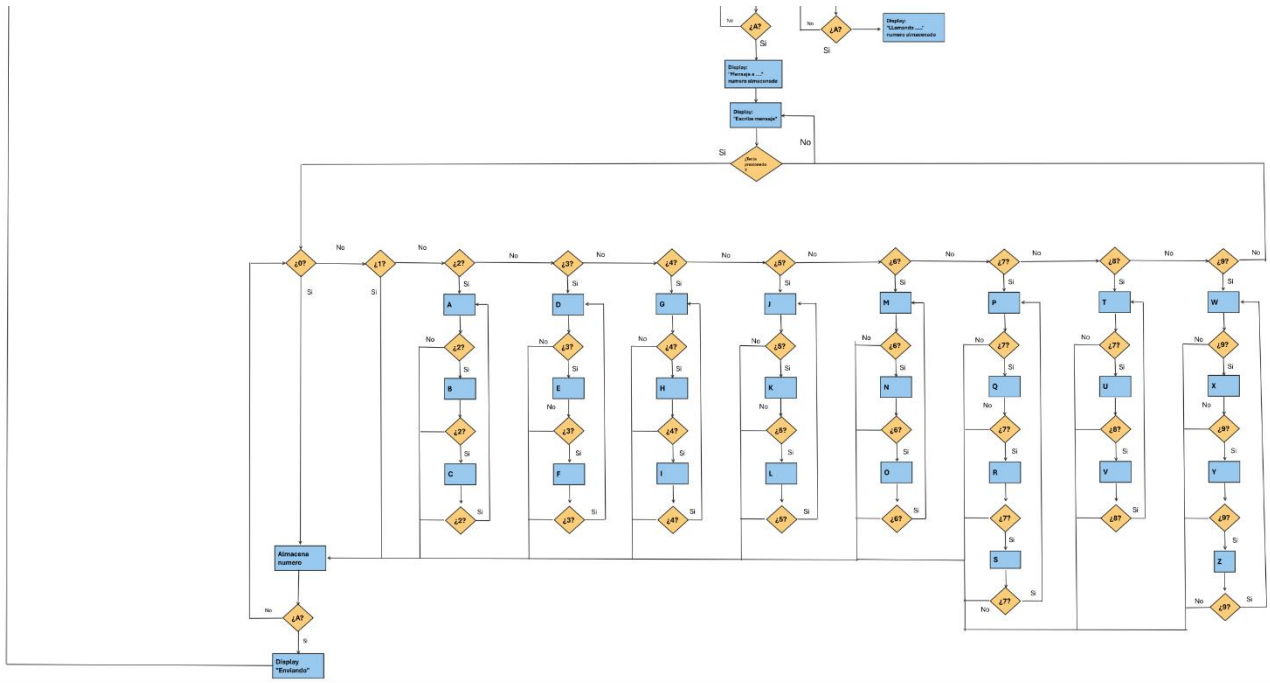

*Ilustración 31*

## 4.5 Software utilizado

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

int numt[10],
    numt1[10]={'5','5','4','8','8','0','2','1','8','8'};//CELULAR PREDEFINIDO


uint8_t d_uint8Dato;
int CM, entrante, i, j, dato, ren, col, pantalla, z, recibe,
llamada_m,k,edo=0,potencia=0x30, pulso=0,delay,aux,limite,
espacio,c,valido,lecturaPE3,lecturaPE2,lecturaPE1,lecturaPE0,a,b,tc;
char Rx[80],mensaje_mem[80];
char envio[32];
```

```
char recibido[32],error[5]={'E','R','R','O','R'},puerta[13]={'P','U','E','R','T','A',' ','E','N',' ','U','S','O'},
      apertura[19]={'P','U','E','R','T','A ',' ','A','B','I','E','R','T','A',' ',' ',' ',' ',' ',' ',' '},
      cerradura[10]={'C','E','R','R','A','D','A',' ',' ',' ',' '};

unsigned char Valcon,Temp,Temp2;
char var, pvar;


// ARREGLO PARA INVERSOR , LA MODULACIÓN SE LIMITA A 3 , YA QUE UNA MAYOR PODRIA ENTREGARNOS VOLTAJES DEMASIADO ALTOS
//ALGO QUE VIMOS EN EL DESARROLLO DEL CURSO
const int p [11][21]= {{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0},
                       {0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},
                       {0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0},
                       {0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0},
                       {0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0},
                       {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0}};

int output=0, ele=0, pola=0;//pola es el ciclo, si pola es 0, el ciclo es positivo, si es 1, es negativo



//DEFINICIONES

#define bitset(Var,bitno) ((Var) |= 1 << (bitno))
#define bitclr(Var,bitno) ((Var) &= ~(1 << (bitno)))

#define PORTDIS GPIO_PORTK_DATA_R
#define PORTCON GPIO_PORTM_DATA_R

//bits de control en PORTM
#define RS   0x00
#define R_W  0x01
#define E    0x02

#define BIT7 0X80

#define enciende 0X0F
```

```
#define CLR 0x01

//DEFINICION DE VALORES HEXADECIMALES
int hexa [4][4]={{0x31 , 0x32, 0x33, 0x41},
                 {0x34, 0x35, 0x36, 0x42},
                 {0x37, 0x38, 0x39, 0x43},
                 {0x2A, 0x30 , 0x23, 0x44}};

//DEFINICION DE CARACTERES TECLADO
char numeros [4][4]={{'1' , '2', '3', 'A'},
                     {'4', '5', '6', 'B'},
                     {'7', '8', '9', 'C'},
                     {'*', '0' , '#', 'D'}};

//VALORES ASOCIADOS A TECLADO NUMERICO
char letras[10][5] = {
   {' '},      // 0
   {'1'},      // 1
   {'A', 'B', 'C', '2'},      // 2
   {'D', 'E', 'F', '3'},      // 3
   {'G', 'H', 'I', '4'},      // 4
   {'J', 'K', 'L', '5'},      // 5
   {'M', 'N', 'O', '6'},      // 6
   {'P', 'Q', 'R', 'S', '7'}, // 7
   {'T', 'U', 'V', '8'},      // 8
   {'W', 'X', 'Y', 'Z', '9'}  // 9
};


//FUNCIONES

void LCD(void);
void BUSY (void);
void BUSY1 (void);
void _E(void);
void data (unsigned char c);
void comando (unsigned char d);
void CLS(void);
void BYTEDIS(unsigned char DByte);
void HOME(void);
void LEECON(void);
void cfg(void);
void winnum(void);
void salida(void);
```

```
char UART6Tx(char dato);
void escribe_pantalla(void);

void INIM();
void mensaje();
void recibe_mensaje();
void limpia_d();
void envia_mensaje();
void adc(void);


//NUESTRA FUNCION PRINCIPAL
int main(void){
   uint32_t g_ui32SysClock;
   g_ui32SysClock = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ
|SYSCTL_OSC_MAIN |SYSCTL_USE_PLL |SYSCTL_CFG_VCO_480),
48000000);
   adc();
   cfg();
   LCD();
   HOME();
   INIM();
   SysCtlDelay(53300);
   while(1){

      ADC0_PSSI_R = 0X02; //INICIA CONVERSION EN SECUENCIADOR 1
         while((ADC0_ACTSS_R & 0x10000)==0x10000);
         lecturaPE3=ADC0_SSFIFO1_R;
         lecturaPE2=ADC0_SSFIFO1_R;
         lecturaPE1=ADC0_SSFIFO1_R;
         lecturaPE0=ADC0_SSFIFO1_R;
         a=(lecturaPE0)/54;
         if(Rx[3]=='G'&&i==6||Rx[10]== 0xA
&&i==11||Rx[3]=='E'&&Rx[4]=='G'&&i<23) {
                  i=0;
                  z=0;
                  pantalla=1;
               }
         if (a>27&&tc==0){
            tc=1;

            UART6Tx((char)(0x41)); //A
            UART6Tx((char)(0x54)); //T
            UART6Tx((char)(0x2B)); //+
```

```
UART6Tx((char)(0x43)); //C
UART6Tx((char)(0x4D)); //M
UART6Tx((char)(0x47)); //G
UART6Tx((char)(0x46)); //F
UART6Tx((char)(0x3D)); //=
UART6Tx((char)(0x31)); //1
UART6Tx((char)(0x3B)); //;
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //

for (delay = 0; delay < 10000000; delay++) {}

UART6Tx((char)(0x41)); //A
UART6Tx((char)(0x54)); //T
UART6Tx((char)(0x2B)); //+
UART6Tx((char)(0x43)); //C
UART6Tx((char)(0x4D)); //M
UART6Tx((char)(0x47)); //G
UART6Tx((char)(0x53)); //S
UART6Tx((char)(0x3D)); //=
UART6Tx((char)(0x22)); //"

//Se envia numero
for(z=0;z<10;z++){
   UART6Tx((char)numt1[z]);
}

UART6Tx((char)(0x22)); //"
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //

for (delay = 0; delay < 100000; delay++) {}

for(z=0;z<19;z++){
   UART6Tx((char)apertura[z]);
}

UART6Tx((char)(0x1a)); //
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //

for (delay = 0; delay < 100000; delay++) {}
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //
valido=0;
```

```
    INIM();

}else if(a<23 && tc==1){
   tc=0;
   limpia_d();
   HOME();
   UART6Tx((char)(0x41)); //A
   UART6Tx((char)(0x54)); //T
   UART6Tx((char)(0x2B)); //+
   UART6Tx((char)(0x43)); //C
   UART6Tx((char)(0x4D)); //M
   UART6Tx((char)(0x47)); //G
   UART6Tx((char)(0x46)); //F
   UART6Tx((char)(0x3D)); //=
   UART6Tx((char)(0x31)); //1
   UART6Tx((char)(0x3B)); //;
   UART6Tx((char)(0x0a)); //
   UART6Tx((char)(0x0d)); //

   for (delay = 0; delay < 100000; delay++) {}

   UART6Tx((char)(0x41)); //A
   UART6Tx((char)(0x54)); //T
   UART6Tx((char)(0x2B)); //+
   UART6Tx((char)(0x43)); //C
   UART6Tx((char)(0x4D)); //M
   UART6Tx((char)(0x47)); //G
   UART6Tx((char)(0x53)); //S
   UART6Tx((char)(0x3D)); //=
   UART6Tx((char)(0x22)); //"
   //Se envia numero
   for(z=0;z<10;z++){
      UART6Tx((char)numt1[z]);
   }

   UART6Tx((char)(0x22)); //"
   UART6Tx((char)(0x0a)); //
   UART6Tx((char)(0x0d)); //

   for (delay = 0; delay < 100000; delay++) {}

   for(z=0;z<10;z++){
      UART6Tx((char)cerradura[z]);
   }
```

```
                UART6Tx((char)(0x1a)); //
                UART6Tx((char)(0x0a)); //
                UART6Tx((char)(0x0d)); //

                for (delay = 0; delay < 100000; delay++) {}
                UART6Tx((char)(0x0a)); //
                UART6Tx((char)(0x0d)); //
                valido=0;
                INIM();

              }
          if(recibe==1){
             if (CM==1){
                recibe_mensaje();}

             }else if(CM==2) {
                llamadaR();
             }
          }
       }
    void adc(){
      //CONFIGURACION
       SYSCTL_RCGCADC_R|=0XF; //HABILITA RELOJ ADC0
       SYSCTL_RCGCGPIO_R|=0x10; //HABILITA PE
       b=123; //TIEMPO DE ESTABILIZACI N DE RELOJES
       //CONFIGURA PUERTO E
       GPIO_PORTE_AHB_AFSEL_R = 0x01; //FUNCION ALTERNA PE0
       GPIO_PORTE_AHB_DEN_R = 0x0; //SE APAGA FUNCION DIGITAL PE0
       GPIO_PORTE_AHB_AMSEL_R =0x0F; //HABILITA ENTRADA ANALOGICA
       //CONFIGURACION DEL CONVERTIDOR ADC0 SECUENCIADOR 1
       ADC0_SSPRI_R = 0x02103; //SECUENCIADOR 3 ES EL DE MAYOR
    PRIORIDAD
       ADC0_ACTSS_R = 0x0; //APAGA SECUENCIADOR 1
       ADC0_EMUX_R = 0x0; //DISPARO POR SOFTWARE
       ADC0_SSMUX1_R = 0x3210;
       ADC0_SSEMUX1_R = 0x0;
       ADC0_SSCTL1_R = 0x2000;
       ADC0_ACTSS_R = 0x02; //HABILITA SECUENCIADOR 1
    }
    void cfg(){
       // Habilitar el reloj del puerto  A, H, J, K, M, N, P, y Q
       SYSCTL_RCGCGPIO_R |= 0X7B81;
       k=123;
       //CONFIGURA EL TIMMER 3
       SYSCTL_RCGCTIMER_R = 0x08; //ACTIVA RELOJ TIMER 3
```

```
    k=1234; //TIEMPO PARA ESTABILIZAR RELOJES
    //CONFIGURAMOS TIMER 3 SUBTIMER A
    TIMER3_CTL_R = 0X0; //DESCATIVA EL TIMER A UTILIZAR
    TIMER3_CFG_R = 0X04; //ACTIVAMOS LA FUNCION DE 16 BITS DEL TIMER
    TIMER3_TAMR_R = 0X02; //CUENTA HACIA ABAJO
    TIMER3_TAILR_R = 0X18CD; //CUENTA PRECARGADA
    TIMER3_ICR_R = 0x1; //LIMPIA BANDERA DE INTERRUPCION
    TIMER3_IMR_R = 0X01; //TIMEOUT
    TIMER3_TAPR_R = 0X0; //PRESCALADO
    NVIC_EN1_R= 1<<(35-32); //ACTIVA INTERRUPCION TIMER 3
    TIMER3_CTL_R |= 0X01; //ACTIVA TIMER

    //CONFIGURAMOS PUERTO N (INVERSOR)
    GPIO_PORTN_DIR_R = 0X0C; //ACTIVAMOS LA salida DIGITAL PN2 y PN3
    GPIO_PORTN_DEN_R = 0X0C;  //HABILITAMOS PN2 y PN3

    //Configura pines del puerto K y M  (PANTALLA)
    GPIO_PORTK_DATA_R=0X00;
    GPIO_PORTK_DEN_R=0XFF;
    GPIO_PORTK_DIR_R=0XFF;
    GPIO_PORTM_DATA_R|=0X00;
    GPIO_PORTM_DEN_R |=0X07;
    GPIO_PORTM_DIR_R |=0X07;
    // Configurar pines del puerto H y Q(TECLADO)
    GPIO_PORTH_AHB_DIR_R |= 0x0F;
    GPIO_PORTH_AHB_DEN_R |= 0X0F;
    GPIO_PORTQ_DIR_R &= ~0X0F;
    GPIO_PORTQ_DEN_R |= 0X0F;
    GPIO_PORTQ_PUR_R |= 0X0F;
    GPIO_PORTQ_IS_R &= ~0x00;       // PQ es sensible por flanco
    GPIO_PORTQ_IBE_R &= ~0x0F;      // PQ no es sensible a dos flancos
    GPIO_PORTQ_IEV_R &= ~0x0F;      // PQ detecta eventos de flanco de bajada
    GPIO_PORTQ_ICR_R = 0x0F;        // limpia la bandera 0
    GPIO_PORTQ_IM_R |= 0x0F;        // Se desenmascara la interrupcion y se
envia al controlador de interrupciones
    //Configura puerto J
    GPIO_PORTJ_AHB_DIR_R &= ~0X03;
    GPIO_PORTJ_AHB_DEN_R |= 0X03;
    GPIO_PORTJ_AHB_PUR_R |= 0X03;
    GPIO_PORTJ_AHB_IS_R &= ~0x00;       // PQ es sensible por flanco
    GPIO_PORTJ_AHB_IBE_R &= ~0x03;      // PQ no es sensible a dos flancos
    GPIO_PORTJ_AHB_IEV_R &= ~0x03;      // PQ detecta eventos de flanco de
bajada
    GPIO_PORTJ_AHB_ICR_R = 0x03;        // limpia la bandera 0
```

```
    GPIO_PORTJ_AHB_IM_R |= 0x03;        // Se desenmascara la interrupcion y
se envia al controlador de interrupciones
    //Priorirdad de control de interrupciones
    NVIC_PRI21_R = (NVIC_PRI21_R&0x00000000)|0xFFFFFFFF;
    NVIC_EN1_R |=0x80000;
    NVIC_EN2_R =0xF00000;          //habilita la interrupci n Q0-Q3 respectivo a
84,85,86 y 87 en NVIC

    SYSCTL_RCGCUART_R |=0X00C1; //HABILITAR UART0,UART6, UART6
    UART0_CTL_R &=~0X0001;  //DESHABILITAR UART
    UART0_IBRD_R=26; //IBDR=int(48000000/16*115200))= 26.0416
    UART0_FBRD_R =2 ; //FBRD= 0.1267*64 =2
    UART0_LCRH_R =0X0060; //8 BITS, HABILITAR FIFO 70  y desbilita 60
    UART0_IM_R |=0x10;
    NVIC_EN0_R =0x20;  //es la posicion E0 en el espacio 0
    UART0_CTL_R= 0X0301 ; //HABILITAR RXE, TXE Y UART


    UART6_CTL_R &=~0X0001;  //DESHABILITAR UART
    UART6_IBRD_R = 26; //IBDR=int(48000000/16*115200))= 26.0416
    UART6_FBRD_R =2 ; //FBRD= 0.1267*64 =2
    UART6_LCRH_R =0X0060; //8 BITS, HABILITAR FIFO 70  y desbilita 60
    UART6_IM_R |=0x10;
    NVIC_EN1_R |=0x8000000;
    UART6_CTL_R= 0X0301 ; //HABILITAR RXE, TXE Y UART


    GPIO_PORTA_AHB_PCTL_R =
(GPIO_PORTA_AHB_PCTL_R&0XFFFFFF00)+0X00000011; //UART0
    GPIO_PORTA_AHB_AMSEL_R &= ~0X03; //DESHABILITAR FUNCION
ANLOGICA EN PA0-1
    GPIO_PORTA_AHB_AFSEL_R |= 0X03; //HABILITAR FUNCION ALTERNA EN
PA0-1
    GPIO_PORTA_AHB_DEN_R |= 0X03; //HABILITAR FINCION I/O DIGITAL

    GPIO_PORTP_PCTL_R =
(GPIO_PORTP_PCTL_R&0XFFFFFF00)+0X00000011; //UART6
    GPIO_PORTP_AMSEL_R &= ~0X03; //DESHABILITAR FUNCION ANLOGICA
EN PP0-1
    GPIO_PORTP_AFSEL_R |= 0X03; //HABILITAR FUNCION ALTERNA EN PP0-
1
    GPIO_PORTP_DEN_R |= 0X03; //HABILITAR FINCION I/O DIGITAL

}
```

```
//DEFINICION DE MENSAJE
MENSAJEPRED (void){
   SysCtlDelay (320000);
   k=GPIO_PORTJ_AHB_DATA_R;
   switch(k){

   case 0:
      comando(CLR);
      comando(0X80);
      data((char)(0x20));
      for(z=0;z<5;z++){
         data((char)(error[z]));
      }
      SysCtlDelay (32000000);
      HOME();
      break;
   case 1:
      comando(CLR);
      comando(0X80);
      data((char)(0x20));  //
      data((char)(0x45));  // E
      data((char)(0x4E));  // N
      data((char)(0x56));  // V
      data((char)(0x49));  // I
      data((char)(0x41));  // A
      data((char)(0x20));  //
      data((char)(0x4D));  // M
      data((char)(0x45));  // E
      data((char)(0x4E));  // N
      data((char)(0x53));  // S
      data((char)(0x41));  // A
      data((char)(0x4A));  // J
      data((char)(0x45));  // E
      data((char)(0x20));  //
      data((char)(0x41));  //A

      comando(0XC0);//CURSOR A 2DO RENGLON

      for(z=0;z<10;z++){
         numt[z]=numt1[z];
         data((char)(numt[z]));
      }
      SysCtlDelay (32000000);
      comando(CLR);
```

```
        comando(0X80);
        data(0x20); //
        data(0x45); //E
        data(0x53); //S
        data(0x43); //C
        data(0x52); //R
        data(0x49); //I
        data(0x42); //B
        data(0x45); //E
        data(0x20); //
        data(0x4D); //M
        data(0x45); //E
        data(0x4E); //N
        data(0x53); //S
        data(0x41); //A
        data(0x4A); //J
        data(0x45); //E
        data(0x2E); //.
        data(0x2E); //.
        data(0x2E); //.
        SysCtlDelay (28000000);
        comando(CLR);
        comando(0X80);
        pantalla=3;
        c=-1;
        break;
    case 2:
        comando(CLR);
        comando(0X80);
        data((char)(0x20));  //
        data((char)(0x45));  // E
        data((char)(0x4E));  // N
        data((char)(0x56));  // V
        data((char)(0x49));  // I
        data((char)(0x41));  // A
        data((char)(0x20));  //
        data((char)(0x4D));  // M
        data((char)(0x45));  // E
        data((char)(0x4E));  // N
        data((char)(0x53));  // S
        data((char)(0x41));  // A
        data((char)(0x4A));  // J
        data((char)(0x45));  // E
        data((char)(0x20));  //
        data((char)(0x41));  //A
```

```
comando(0XC0);//CURSOR A 2DO RENGLON
for(z=0;z<10;z++){
   data((char)(numt1[z]));
}
SysCtlDelay (32000000);
comando(CLR);
comando(0X80);
UART6Tx((char)(0x41)); //A
UART6Tx((char)(0x54)); //T
UART6Tx((char)(0x2B)); //+
UART6Tx((char)(0x43)); //C
UART6Tx((char)(0x4D)); //M
UART6Tx((char)(0x47)); //G
UART6Tx((char)(0x46)); //F
UART6Tx((char)(0x3D)); //=
UART6Tx((char)(0x31)); //1
UART6Tx((char)(0x3B)); //;
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //

for (delay = 0; delay < 10000000; delay++) {}

UART6Tx((char)(0x41)); //A
UART6Tx((char)(0x54)); //T
UART6Tx((char)(0x2B)); //+
UART6Tx((char)(0x43)); //C
UART6Tx((char)(0x4D)); //M
UART6Tx((char)(0x47)); //G
UART6Tx((char)(0x53)); //S
UART6Tx((char)(0x3D)); //=
UART6Tx((char)(0x22)); //"

//Se envia numero
for(z=0;z<10;z++){
   UART6Tx((char)numt1[z]);
}

UART6Tx((char)(0x22)); //"
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //

for (delay = 0; delay < 100000; delay++) {}

for(z=0;z<13;z++){
   UART6Tx((char)puerta[z]);
```

```
        }

        UART6Tx((char)(0x1a)); //
        UART6Tx((char)(0x0a)); //
        UART6Tx((char)(0x0d)); //

        for (delay = 0; delay < 100000; delay++) {}
        UART6Tx((char)(0x0a)); //
        UART6Tx((char)(0x0d)); //

        valido=0;
        INIM();
        limpia_d();
        HOME();
        break;
    }

    GPIO_PORTJ_AHB_ICR_R=0x03; //Limpiamos la bandera de recepci n de
datos
}

UART6_INT (void){
    d_uint8Dato=((char)(UART6_DR_R&0xFF));
    if(d_uint8Dato=='+'){
        for(z=0;z<80;z++){
            Rx[z]=0; //Se limpia el arreglo de recepci n
        }
    }
    if((d_uint8Dato=='+')||(valido==1)){
        valido=1;
        if (pantalla!=4){
        pantalla=0;
        }
        Rx[i]=d_uint8Dato;

        if((Rx[7]=='+')&&(i>6 && i<17)){
            CM=1;
        }else if ((Rx[7]=='"')&&(i>7 && i<18)){
            CM=2;
            numt[i-8]=Rx[i];
        }

        if((i>49)&&(d_uint8Dato==10)){
            valido=0;
            recibe=1;
```

```
    }
    i++;
}
if (CM == 1){
if(i>49&&d_uint8Dato==10){
    z=0;
    k=0;
    i=0;

    while(Rx[z]!=10){
        z++;
    }
    z++;
    pvar=Rx[z];
    while(Rx[z]!=10){
        recibido[i]=Rx[z];
        z++;
        i++;
    }
    if(Rx[7]=='+'){
        for (z=0;z<10;z++){
            numt[z]=Rx[z+10];
        }
    }else{
        for (z=0;z<10;z++){
            numt[z]=Rx[z+9];
        }
    }
        switch(pvar){

        case'A':
            edo=1;
            break;

        case 'E':
            edo=0;
            break;

        case 'I':
            if (pulso<=3){
                pulso++;
            }
            break;
        case 'R':
            if (pulso>0){
```

```
                pulso=pulso-1;
            }
            break;
        }
        switch (pulso){
            case 0:
                potencia=0x30;
                break;
            case 1:
                potencia=0x31;
                break;
            case 2:
                potencia=0x32;
                break;
            case 3:
                potencia=0x33;
                break;
            default:
                potencia=0x45;
            }
        for(z=0;z<80;z++){
            Rx[z]=0;
        }
    }

  }
  UART6_ICR_R=0x0F; //Limpiamos la bandera de recepci n de datos
}

void LCD(void)  //INICIALIZA EL DISPLAY A DOS LINEAS, 4 BITS DE
INTERFAZ, CURSOR Y DESTELLO.
{
  //PRIMER METODO DE INICIALIZACION
  PORTCON=0X00;
  PORTDIS=0X38; // **D7-D4 = 0011

  SysCtlDelay(533000);              //Delay 100ms
  _E();                         //toggle E
  SysCtlDelay (53000);              //10ms
  _E();                         //toggle E
  SysCtlDelay (53000);              //10ms
  ;PORTDIS=0X38;
  _E();                         //toggle E
  SysCtlDelay (53000);              //10ms
  comando(0X39);   //8 BITS , 2 LINEAS, 5X7
```

```
    comando(enciende);  // ENCIENDE DISPLAY CON CURSOR Y DESTELLO


    comando(CLR); // BORRAR DISPLAY

    comando(0X07); // AL ESCRIBIR EL CURSOR SE INCREMENTA Y SE
DESPLAZA
    comando(0X38); // //8 BITS , 2 LINEAS, 5X7 REQUERIDO POR LOS
DISPLAYS CLONES
}

void INIM(){
    //Comandos de SIM808

    //Suprime el eco del SIM808 al recibir un comando
    UART6Tx((char)(0x41));  // A
    UART6Tx((char)(0x54));  // T
    UART6Tx((char)(0x45));  // E
    UART6Tx((char)(0x30));  // 0
    UART6Tx((char)(0x0a));
    UART6Tx((char)(0x0d));

    SysCtlDelay(53300);

    //Configura lectura en modo texto
    UART6Tx((char)(0x41));  // A
    UART6Tx((char)(0x54));  // T
    UART6Tx((char)(0x2B));  // +
    UART6Tx((char)(0x43));  // C
    UART6Tx((char)(0x4D));  // M
    UART6Tx((char)(0x47));  // G
    UART6Tx((char)(0x46));  // F
    UART6Tx((char)(0x3D));  // =
    UART6Tx((char)(0x31));  // 1
    UART6Tx((char)(0x3B));  // ;
    UART6Tx((char)(0x0a));
    UART6Tx((char)(0x0d));
    SysCtlDelay(53300);


    //Configuramos el mOdulo para que muestre los SMS por el puerto serie.
    UART6Tx((char)(0x41));  // A
    UART6Tx((char)(0x54));  // T
    UART6Tx((char)(0x2B));  // +
```

```
    UART6Tx((char)(0x43));  // C
    UART6Tx((char)(0x4E));  // N
    UART6Tx((char)(0x4D));  // M
    UART6Tx((char)(0x49));  // I
    UART6Tx((char)(0x3D));  // =
    UART6Tx((char)(0x32));  // 2
    UART6Tx((char)(0x2C));  // ,
    UART6Tx((char)(0x32));  // 2
    UART6Tx((char)(0x2C));  // ,
    UART6Tx((char)(0x30));  // 0
    UART6Tx((char)(0x2C));  // ,
    UART6Tx((char)(0x30));  // 0
    UART6Tx((char)(0x2C));  // ,
    UART6Tx((char)(0x30));  // 0
    UART6Tx((char)(0x3B));  // ;
    UART6Tx((char)(0x0a));
    UART6Tx((char)(0x0d));
    SysCtlDelay(53300);

}

void tecladoQ(void){
    //FunciOn de interrupciOn del puerto Q para teclado
    int p_tecla;
    p_tecla=0;
    col=-1;
    ren=-1;
    SysCtlDelay (320000);
    //delay tiempos de rebote de 15 [ms], entonces con 20[ms] es suficiente

    //EvalUa en que puerto se dio la interrupciOn
    if((GPIO_PORTQ_DATA_R&0x0F)!=0x0F){
        if((GPIO_PORTQ_DATA_R&0x0F)==0x0E&&(p_tecla!=1)){
            ren=0;
        }
        else if((GPIO_PORTQ_DATA_R&0x0F)==0x0D&&(p_tecla!=1)){
            ren=1;
        }
        else if((GPIO_PORTQ_DATA_R&0x0F)==0x0B&&(p_tecla!=1)){
            ren=2;
        }
        else if((GPIO_PORTQ_DATA_R&0x0F)==0x07&&(p_tecla!=1)){
            ren=3;
        }
        else{
```

```
    //mas de una tecla pulsada
}

//SE MODIFICA EL ESTADO DEL PUERTO H A 0
GPIO_PORTH_AHB_DATA_R &= 0XF0;
//ENCENDEMOS LAS ENTRADAS H3, H2, H1
GPIO_PORTH_AHB_DATA_R |= 0X0E;
//evaluamos si el puerto sigue siendo diferente de 0x0F
if(((GPIO_PORTQ_DATA_R&0x0F)!=0x0F)&&(p_tecla!=1)){
    col=0;
    p_tecla=1;
    var=numeros [col][ren];
}
GPIO_PORTH_AHB_DATA_R &= 0XF0;
//ENCENDEMOS LAS ENTRADAS H3, H2, H0
GPIO_PORTH_AHB_DATA_R |= 0X0D;
if(((GPIO_PORTQ_DATA_R&0x0F)!=0x0F)&&p_tecla!=1){
    col=1;
    p_tecla=1;
    var=numeros [col][ren];
}
GPIO_PORTH_AHB_DATA_R &= 0XF0;
//ENCENDEMOS LAS ENTRADAS H3, H1, H0
GPIO_PORTH_AHB_DATA_R |= 0X0B;
if(((GPIO_PORTQ_DATA_R&0x0F)!=0x0F)&&p_tecla!=1){
    col=2;
    p_tecla=1;
    var=numeros [col][ren];
}
GPIO_PORTH_AHB_DATA_R &= 0XF0;
//ENCENDEMOS LAS ENTRADAS H2, H1, H0
GPIO_PORTH_AHB_DATA_R |= 0X07;
if(((GPIO_PORTQ_DATA_R&0x0F)!=0x0F)&&p_tecla!=1){
    col=3;
    p_tecla=1;
    var=numeros [col][ren];
}
GPIO_PORTH_AHB_DATA_R &= 0XF0;

if(var=='*'){
        if (pantalla==2||pantalla==5){
        limpia_d();
        GPIO_PORTQ_ICR_R = 0x0F;
        winnum();
        }
```

```
            }
            //Si presiona A en pantalla=2  pasa a escritura de mensaje y en
pantalla=3 enviamos mensaje ya escrito
            else if(var=='A'){
               if(pantalla==1){
                  pantalla=2;
                  GPIO_PORTQ_ICR_R = 0x0F;
                  winnum();
               }else if(pantalla==2&&j==10){
                  c=-1;
                  j=0;
                  pantalla=3;
                  mensaje();
               }
               else if(pantalla==5&&j==10){
                  GPIO_PORTQ_ICR_R = 0x0F;
                  llamada();
                  limpia_d();
               }else if(pantalla!=4){
                  GPIO_PORTQ_ICR_R = 0x0F;
                  envia_mensaje();
               }
            }
            //Presionar B es respectivo a regresar a la pantalla anterior, dependiendo
de la pantalla en la que se encuentre
            else if(var=='B'){

               if(pantalla!=3&&pantalla!=0&&pantalla!=4){
                  limpia_d();
                  pantalla=1;
                  GPIO_PORTQ_ICR_R = 0x0F;
                  HOME();
               }
               else if (pantalla==3){
                  pantalla=2;
                  limite=-1;
                  for(z=0;z<32;z++){
                     envio[z]=0;
                  }
                  GPIO_PORTQ_ICR_R = 0x0F;
                  winnum();
               }
            }else if(var=='C'){
               if(pantalla==1){
                  pantalla=5;
```

```
                GPIO_PORTQ_ICR_R = 0x0F;
                winnum();
              }
            }
        else if(var=='#'){
            GPIO_PORTQ_ICR_R = 0x0F;
        }
        //Pantalla=3 es la pantalla para escribir el mensaje, por eso se llama a la funci
n escribe_pantalla()
        else if(pantalla==3){
            GPIO_PORTQ_ICR_R = 0x0F; //limpia el registro de interrupci n
            escribe_pantalla();
        }
        else{
            if(j<10){
                data(var);
                numt[j]=hexa[col][ren];
                j++;
            }
        }
        //Espera a que se deje de presionar la tecla actual
        while((GPIO_PORTQ_DATA_R&0x0F)!=0x0F){
        }
        //Limpia la bandera
        GPIO_PORTQ_ICR_R = 0x0F;
    }
}


void winnum(){
    //Limpia y escribe en el display
    comando(CLR);

    comando(0X80);

    data(0X00); //nulo
    data(0x45); //E
    data(0x53); //S
    data(0x43); //C
    data(0x52); //R
    data(0x49); //I
    data(0x42); //B
    data(0x45); //E
    comando(0x88); //
    data(0X4E); //N
```

```
    data(0x55); //U
    data(0x4D); //M
    data(0x45); //E
    data(0x52); //R
    data(0x4F); //O

    comando(0XC0);//CURSOR A 2DO RENGLON
}


void mensaje(){
    comando(0X80);

    data(0x4D); //M
    data(0x45); //E
    data(0x4E); //N
    data(0x53); //S
    data(0x41); //A
    data(0x4A); //J
    data(0x45); //E
    comando(0X88);
    data(0x41); //A
    for(z=0; z<8; z++){
        data(0x2E);
    }

    SysCtlDelay (28000000);

    comando(CLR);

    comando(0X80);
    data(0x20); //
    data(0x45); //E
    data(0x53); //S
    data(0x43); //C
    data(0x52); //R
    data(0x49); //I
    data(0x42); //B
    data(0x45); //E
    data(0x20); //
    data(0x4D); //M
    data(0x45); //E
    data(0x4E); //N
    data(0x53); //S
```

```
    data(0x41); //A
    data(0x4A); //J
    data(0x45); //E
    data(0x2E); //.
    data(0x2E); //.
    data(0x2E); //.

    SysCtlDelay (28000000);

    comando(CLR);

    comando(0X80);

}

void escribe_pantalla(void){
    //Funci n de escritura de mensaje
    int local,aux_mem;
    local=var-48;

    if(aux!=local&&c<32){
        aux=var-48;
        limite=0;
        if(aux_mem!=20){
        c++;
        }
    }
    else{
        limite++;
    }

    //En caso de presionar C regresaremos dos espacios
    if(aux==19&&c>0){
        c=c-2;
    }else if(aux==20&&c>=0){
        c++;
    }

    //Cambiamos de posici n en la pantalla
    if(c<16){
        espacio=(0X80)+c;
    }
    else{
        espacio=(0xC0)+c-16;
    }
```

```
    comando(espacio);

    //Al pulsar 0 y 1 hay dos caracteres posibles
    //Al pulsar 7 y 9 hay cinco caracteres posibles
    //Las demas tienen 4 estados
    if(aux<2){
        data(letras[aux][0]);
        envio[c]=letras[aux][0];
    }
    else if(aux==7 || aux==9){
        data(letras[aux][limite]);
        envio[c]=letras[aux][limite];
        if(limite==4){
            limite=-1;
        }
    }
    else if(aux>1&&aux<9){
        data(letras[aux][limite]);
        envio[c]=letras[aux][limite];
        if(limite==3){
            limite=-1;
        }
    }
    else{
        var=-1;
    }
    aux_mem=aux;
    while((GPIO_PORTQ_DATA_R&0x0F)!=0x0F);
}

void envia_mensaje(){
    int delay;
    comando(CLR);
    comando(0X80);

    data(0x20); //
    data(0x45); //E
    data(0x4E); //N
    data(0x56); //V
    data(0x49); //I
    data(0x41); //A
    data(0x4E); //N
    data(0x44); //D
    data(0x4F); //O
```

```
UART6Tx((char)(0x41)); //A
UART6Tx((char)(0x54)); //T
UART6Tx((char)(0x2B)); //+
UART6Tx((char)(0x43)); //C
UART6Tx((char)(0x4D)); //M
UART6Tx((char)(0x47)); //G
UART6Tx((char)(0x46)); //F
UART6Tx((char)(0x3D)); //=
UART6Tx((char)(0x31)); //1
UART6Tx((char)(0x3B)); //;
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //

for (delay = 0; delay < 10000000; delay++) {}
data(0x2E); //.

UART6Tx((char)(0x41)); //A
UART6Tx((char)(0x54)); //T
UART6Tx((char)(0x2B)); //+
UART6Tx((char)(0x43)); //C
UART6Tx((char)(0x4D)); //M
UART6Tx((char)(0x47)); //G
UART6Tx((char)(0x53)); //S
UART6Tx((char)(0x3D)); //=
UART6Tx((char)(0x22)); //"


//Se envia numero
for(z=0;z<10;z++){
   UART6Tx((char)numt[z]);
}

UART6Tx((char)(0x22)); //"
UART6Tx((char)(0x0a)); //
UART6Tx((char)(0x0d)); //

for (delay = 0; delay < 10000000; delay++) {}
data(0x2E); //.

for(z=0;z<=c;z++){
   UART6Tx((char)envio[z]);
}

UART6Tx((char)(0x1a)); //
```

```
    UART6Tx((char)(0x0a)); //
    UART6Tx((char)(0x0d)); //

    for (delay = 0; delay < 100000; delay++) {}
    UART6Tx((char)(0x0a)); //
    UART6Tx((char)(0x0d)); //

    for (delay = 0; delay < 10000000; delay++) {}
    data(0x2E);
    UART6_ICR_R=0x0F;
    for(z=0;z<80;z++){
        Rx[z]=0;
    }
    valido=0;
    INIM();
    limpia_d();
    HOME();
}

char UART6Tx(char dato){
    //Enviamos el dato por el puerto UART6
    while   ((UART6_FR_R&0X0020)!=0); // espera a que TXFF sea cero
    UART6_DR_R=dato;
}

void recibe_mensaje(){
    comando(CLR);
    comando(0X80);

    data((char)(0x20));  //
    data((char)(0x4E));  // N
    data((char)(0x55));  // U
    data((char)(0x45));  // E
    data((char)(0x56));  // V
    data((char)(0x4F));  // O
    data((char)(0x20));  //
    data((char)(0x20));  //
    data((char)(0x4D));  // M
    data((char)(0x45));  // E
    data((char)(0x4E));  // N
    data((char)(0x53));  // S
    data((char)(0x41));  // A
    data((char)(0x4A));  // J
    data((char)(0x45));  // E
```

```
    comando(0XC0);//CURSOR A 2DO RENGLON

    for(z=0;z<10;z++){
        data((char)(numt[z]));
    }

    SysCtlDelay(32000000);

    comando(CLR);
    comando(0X80);

    data(0x20);  //
    for(z=0;z<(i-1);z++){
        if(z==16){
            comando(0XC0);
        }
        data(recibido[z]);
    }
    SysCtlDelay(32000000);
    for(z=0;z<32;z++){
        recibido[z]=0;
    }
    UART6_ICR_R=0x0F;
    i=0;
    recibe=0;
    HOME();
}


void HOME(void){
    pantalla=1;
    comando(CLR);

    if(edo==0){
          comando(0X80);

          data(0X00); //nulo
          data(0x45); //E
          data(0x53); //S
          data(0x54); //T
          data(0x41); //A
          data(0x44); //D
          data(0x4F); //O
          comando(0x87); //
          data(0x4F); //O
```

```
      data(0X4E); //N
      comando(0x8A); //
      data(0x4D); //M
      data(0x3D); //=
      data(potencia); //PULSOS
      comando(0x8E); //
      data(0x20); //
    }else{
      comando(0X80);
      data(0X00); //nulo
      data(0x45); //E
      data(0x53); //S
      data(0x54); //T
      data(0x41); //A
      data(0x44); //D
      data(0x4F); //O
      comando(0x87); //
      data(0x4F); //O
      data(0X46); //F
      data(0x46); //F
      comando(0x8B); //
      data(0x4D); //M
      data(0x3D); //=
      data(potencia); //PULSOS
      data(0x20); //


    }



comando(0XC0);//CURSOR A 2DO RENGLON

data(0X20); //S
data(0x20); //M
data(0x20); //S
data(0x20); //=
data(0x20); //A
data(0x20); //
data(0x20); //
data(0x20); //
data(0x20); //C
data(0x20); //A
data(0X20); //L
data(0x20); //L
```

```
    data(0x20); //=
    data(0x20); //C
    data(0x20); //
    data(0x20); //
}


void _E(void)  //GENERA UN PULSO DE 1.25 uS EN LA TERMINAL E DEL
DISPLAY
{
    bitset(PORTCON,E);        // E=1
    SysCtlDelay (3);  //retraso de 500 nS
    bitclr(PORTCON,E);  // E=0
}

void data (unsigned char c)  //ESCRIBE UN DATO ALFANUMERICO AL
DISPLAY
{
    //ENVIA dato
    PORTDIS = c;

    bitset(PORTCON,RS);    //ENVIA A REGISTRO DE DATOS
    _E();              //TOGGLE _E()
    BUSY1(); //PREGUNTA POR LA BANDERA DE BUSY
    bitclr(PORTCON,RS);    //niveles de control a 0
    }



void comando (unsigned char d)// ENVIA UN COMANDO AL REGISTRO DE
CONTROL DEL LCD
{

    PORTDIS = d;              //envia CMD al LCD
    bitclr(PORTCON,RS);             //direcciona registro de controldel  LCD
    _E();                        //toggle E
    BUSY1(); //VERIFICA LA BANDERA DE BUSY
    }



void CLS(void)
{ comando(0x01);
       }
```

```
void BUSY (void)  //PREGUNTA POR EL ESTADO DE LA BANDERA BUSY Y
ESPERA HASTA QUE SEA CERO
{
   do LEECON( );
   while ((Valcon & BIT7) != 0);
   }


void LEECON(void){
// LEE EL VALOR DEL REGISTRO DE CONTROL DEL DISPLAY Y REGRESA EL
CONTENIDO EN VALCON
       PORTDIS=0;
       GPIO_PORTK_DIR_R=0x00; //PORTK como entrada
       bitset(PORTCON,R_W); // LEER PUERTO DE CONTROL
        bitset(PORTCON,E);      //ACTIVA E
        SysCtlDelay(2); //espera 384 nS
        Temp=PORTDIS; // LEE PARTE ALTA DEL BUS DE DATOS
        bitclr(PORTCON,E);
        bitclr(PORTCON,RS);
        bitclr(PORTCON,R_W);
        GPIO_PORTK_DIR_R=0XFF; //REGRESA A LA CONDICION ORIGINAL
DEL PUERTO K A SALIDA
         Valcon=Temp;
}

void BYTEDIS(unsigned char DByte){ //escribe un byte a pantalla
   Temp2=DByte;
   Temp2=Temp2>>4;
   if (Temp2<=0x09)
      Temp2+=0x30;
   else
      Temp2+=0x37;
   data(Temp2);
   Temp2=DByte&0x0f;
   if (Temp2<=0x09)
      Temp2+=0x30;
   else
      Temp2+=0x37;
   data(Temp2);
}

void BUSY1(void)
   {SysCtlDelay(10066);}

void limpia_d(){
```

```
    j=0;
    c=-1;
    i=0;
    for(z=0;z<10;z++){
        numt[z]=0;
    }
    for(z=0;z<32;z++){
        envio[z]=0;
    }
}
//Interrupci n de puerto UART0, se utiliza para pruebas de comunicaci n
UART0Rx (void){
    d_uint8Dato=((char)(UART0_DR_R&0xff));  //Data Register, donde se guardan
los datos
    UART6Tx(d_uint8Dato);
}
void salida (void){
    if(edo==0){
        if (pola==0)
        {
            output=p[pulso][ele];
            if (output==0){
                GPIO_PORTN_DATA_R = 0x0;
            }else{
                GPIO_PORTN_DATA_R = 0x04;
            }
            if (ele<20)
                ele++;
            else{
                ele=0;
                pola=pola+1;    //Cuando termina de mandar el arreglo cambia de
polaridad
            }
        }
        if (pola==1)
            {
            output=p[pulso][ele];
            if (output==0){
                GPIO_PORTN_DATA_R = 0x0;
            }else{
                GPIO_PORTN_DATA_R = 0x08;
            }
            if (ele<20)
                ele++;
            else{
```

```
            ele=0;
            pola=pola-1;    //Cuando termina de mandar el arreglo cambia de
polaridad
          }
          }

   }else{
      GPIO_PORTN_DATA_R =0x0;
   }

   TIMER3_ICR_R = 0x1; //Limpia bandera de interrupcion
}


void llamada(void){
   comando(0X80);
   data(0x4C); //L
   data(0x4C); //L
   data(0x41); //A
   data(0x4D); //M
   data(0x41); //A
   data(0x4E); //N
   data(0x44); //D
   data(0x4F); //O
   for(z=0; z<8; z++){
      data(0x2E);
   }

   //Comando para hacer llamada
   UART6Tx((char)(0x41)); //A
   UART6Tx((char)(0x54)); //T
   UART6Tx((char)(0x44)); //D
   //Escribe el n mero
   for(z=0;z<10;z++){
      UART6Tx((char)(numt[z]));
   }

   UART6Tx((char)(0x3B)); //;
   UART6Tx((char)(0x0a));
   UART6Tx((char)(0x0d));

   while((GPIO_PORTQ_DATA_R&0x0F)!=0x0F);

   //Usamos la letra B para colgar la llamada
   while(var!='B'){
```

```
        int p_tecla=0, delay;
        col=0;
        ren=0;
        for (delay = 0; delay < 10000; delay++) {}
        //delay tiempos de rebote de 15 [ms], entonces con 20[ms] es suficiente
        if((GPIO_PORTQ_DATA_R&0x0F)!=0x0F){
            if((GPIO_PORTQ_DATA_R&0x0F)==0x07&&(p_tecla!=1)){
                ren=1;
            }
            GPIO_PORTH_AHB_DATA_R &= 0XF0;
            GPIO_PORTH_AHB_DATA_R |= 0X0D;
            if(((GPIO_PORTQ_DATA_R&0x0F)!=0x0F)&&p_tecla!=1){
                col=3;
                p_tecla=1;
                var=numeros [ren][col];
            }
            GPIO_PORTH_AHB_DATA_R &= 0XF0;
        }
    }

    GPIO_PORTQ_ICR_R = 0x0F;

    //Comando para colgar
    UART6Tx((char)(0x41)); //A
    UART6Tx((char)(0x54)); //T
    UART6Tx((char)(0x48)); //H
    UART6Tx((char)(0x3B)); //;
    UART6Tx((char)(0x0a));
    UART6Tx((char)(0x0d));

    //Limpia la bandera de interrupci n
    UART6_ICR_R=0x0F;
    var=0;
    HOME();
    pantalla=1;
    }

void llamadaR(void){
    int delay;
    pantalla=4;
    comando(CLR);
    comando(0X80);

    data(0x20); //
    data((char)(0x52));  // R
```

```
data((char)(0x45));  // E
data((char)(0x43));  // C
data((char)(0x49));  // I
data((char)(0x42));  // B
data((char)(0x49));  // I
data((char)(0x20));  // espacio
data((char)(0x4C));  // L
data((char)(0x4C));  // L
data((char)(0x41));  // A
data((char)(0x4D));  // M
data((char)(0x41));  // A
data((char)(0x44));  // D
data((char)(0x41));  // A

comando(0XC0);//CURSOR A 2DO RENGLON

for(z=0;z<10;z++){
   data((char)(numt[z]));
}


//B cuelga o rechaza y A contesta llamada
for (delay = 0; delay < 30000000; delay++) {
   if(var=='B'){
      UART6Tx((char)(0x41));
      UART6Tx((char)(0x54));
      UART6Tx((char)(0x48));
      UART6Tx((char)(0x3B));
      UART6Tx((char)(0x0a));
      UART6Tx((char)(0x0d));
      entrante=0;
      UART6_ICR_R=0x0F;
      break;
   }
   else if((var=='A')&&(entrante==0)){
      UART6Tx((char)(0x41));
      UART6Tx((char)(0x54));
      UART6Tx((char)(0x41));
      UART6Tx((char)(0x0a));
      UART6Tx((char)(0x0d));
      entrante=1;
      GPIO_PORTQ_ICR_R = 0x0F;
      var=0;
   }
}
```

```
    UART6_ICR_R=0x0F;
    i=0;
    var=0;
    recibe=0;
    CM=0;
    INIM();
    limpia_d();
    HOME();

}
```

## 5. Lista de Partes, Estimación Económica

*Tabla 1*

| Material | Costo |
|---|---|
| Tableta Protoboard | $ 150.00 |
| Tarjeta de Desarrollo Tiva TM4C1294NCPDT | $ 1,200.00 |
| Teclado Matricial 4x4 | $ 45.00 |
| Display LCD 16x2 | $ 100.00 |
| Potenciometro de 1 K[Ω] | $ 15.00 |
| 8 Resistencias de 2.2 K[Ω] a ¼ de Watt | $ 8.00 |
| 8 Resistencias de 3.3 K[Ω] a ¼ de Watt | $ 8.00 |
| 1 Resistencia de 4.7 [Ω] a ½ de Watt | $ 1.00 |
| Módulo SIM 808 GSM GPRS con GPS | $ 490.00 |
| Eliminador De Voltaje 5v / 2a | $ 80.00 |
| Tarjeta SIM Telcel. | $ 50.00 |
| Tarjeta Fenolica Perforada de 10x15 cm | $ 33.00 |
| Pines hembra (170 aprox) | $ 148.00 |
| Pines macho (50 aprox.) | $ 20.00 |
| Tableta Protoboard | $ 150.00 |
| 1 Potenciometro de 5 K[Ω] | $ 10.00 |
| Tarjeta Fenolica Perforada de 8x12 cm | $ 33.00 |
| 2 Borneras de 3 Entradas | $ 10.00 |
| 2 Bases Dip de 8 pines | $ 6.00 |
| Cable calibre 18 (1.5 m) | $ 15.00 |
| Cable calibre 22 | $ 12.00 |
| 2 Resistencias de 220 [Ω] a 1 W | $ 4.00 |
| 2 Octoacopladores 6n137 | $ 28.00 |
| 2 Resistencias de 10 K[Ω] | $ 10.00 |
| 2 Reguladores de Voltaje 7805 | $ 16.00 |
| 2 Capacitores de 0.1 [µF] a 400 [W] | $ 10.00 |
| 2 Resistencias de 1 K[Ω] a ½ [W] | $ 2.00 |
| 2 Transistores 2N2222 | $ 10.00 |
| 2 Diodos NTE4920 | $ 60.00 |
| 2 Transistores IRFP064 | $ 100.00 |
| 1 Transformador de 128 V a 24 V | $ 199.00 |
| 1 socket con foco | $ 60.00 |
| Total | $ 3,083.00 |

## 6. Comentarios, Conclusiones

El presente proyecto funciono correctamente, ya que, el teléfono creado hace/recibe llamadas y mensajes, emite alarmas mediante un mensaje de celular a un número predeterminado y realiza de manera eficiente corriente alterna, además de que permite controlar la modulación para aumentar la tensión que recibe la carga a través del inversor.

**Roa López Jonathan Raúl**

El Teléfono Inteligente con Inversor controlado por Ancho de Pulso utilizando un módulo GSM controlado por Microcontrolador es un sistema embebido ya que utiliza hardware y software que fueron diseñados e implementados a la medida para resolver la problemática planteada.

Este sistema se considera inteligente debido a que puede realizar conexiones con sistemas exteriores, en este caso, bandas 2G. Además de poder realizar diversos tipos de tareas como emitir alarmas, mandar/recibir mensajes y llamadas y controlar el funcionamiento de un inversor de corriente CD-CA.

El programa utilizado corre en un microcontrolador TIVA que permite realizar todas las tareas que se ejecutaran en este sistema. Este programa implementa una modulación PWM para controlar los lóbulos entregados al inversor, comunicación UART que nos permitirá comunicarnos con el modem para la comunicación con el exterior. Además, en este proyecto utilizamos protocolo TCP/IP permitiendo así una comunicación más confiable y eficiente.

Este proyecto permitió reforzar los diferentes conocimientos adquiridos a lo largo de la carrera, si bien se realizó con éxito, hubo unos cuantos inconvenientes, por ejemplo, la técnica a la hora de soldar y realizar conexiones. También la configuración del teléfono celular, además de la señal que recibe que tiene que ser de buena calidad o si no hay que reiniciar el sistema, entre otras. Estas son problemáticas que se resolvieron para la realización y correcto funcionamiento de este sistema, además de que nos aportaron conocimiento y técnica.

**Reyes Herrera Iván**

Dentro del proyecto del teléfono inteligente con inversor controlado por ancho de pulso, logramos sacar provecho de algunas herramientas que nos brinda la tarjeta de desarrollo Tiva TM4C1294, aprovechando los puertos de comunicación UART para poder establecer una comunicación y control del sistema SIM808, esto nos

ayudó a poder sacar provecho de este para poder a través de esta comunicación utilizar otras herramientas que estuvieran interconectadas.

Como ya se establece en el reporte estas herramientas adicionales a la comunicación fue el poder utilizar el inversor desarrollado en la primer parte del curso para poder controlarlo a partir de una comunicación externa al centro SIM.

Con ello podemos recapitular y entender lo que es un sistema embebido, un sistema o circuito desarrollado a la medida para un determinado problema, como lo fue el visto en este proyecto.

Dentro del desarrollo hubo complicaciones con el desarrollo de las tarjetas perforadas, pues en algunos casos como lo fue el circuito inversor, presentamos fallas de corto circuito, por lo que trabajarlas de mejor manera y con un mejor orden en los cables, es un área de oportunidad a mejorar en futuras entregas, además de poder expandir a utilizar otras herramientas con las que se pueda comunicar nuestro sistema, para poder hacerlo algo más personalizado a la industria.

## 7. Bibliografía

Reyes, Y. (2012). *Diseño Didáctico de un Convertidor CD-CA Monofásico de Baja Frecuencia Ajustable.* https://ribuni.uni.edu.ni/1957/1/40050.PDF

# 8. Anexos. Memoria de diseño

## 8.1. Inversor.

Tenemos el diagrama del inversor:

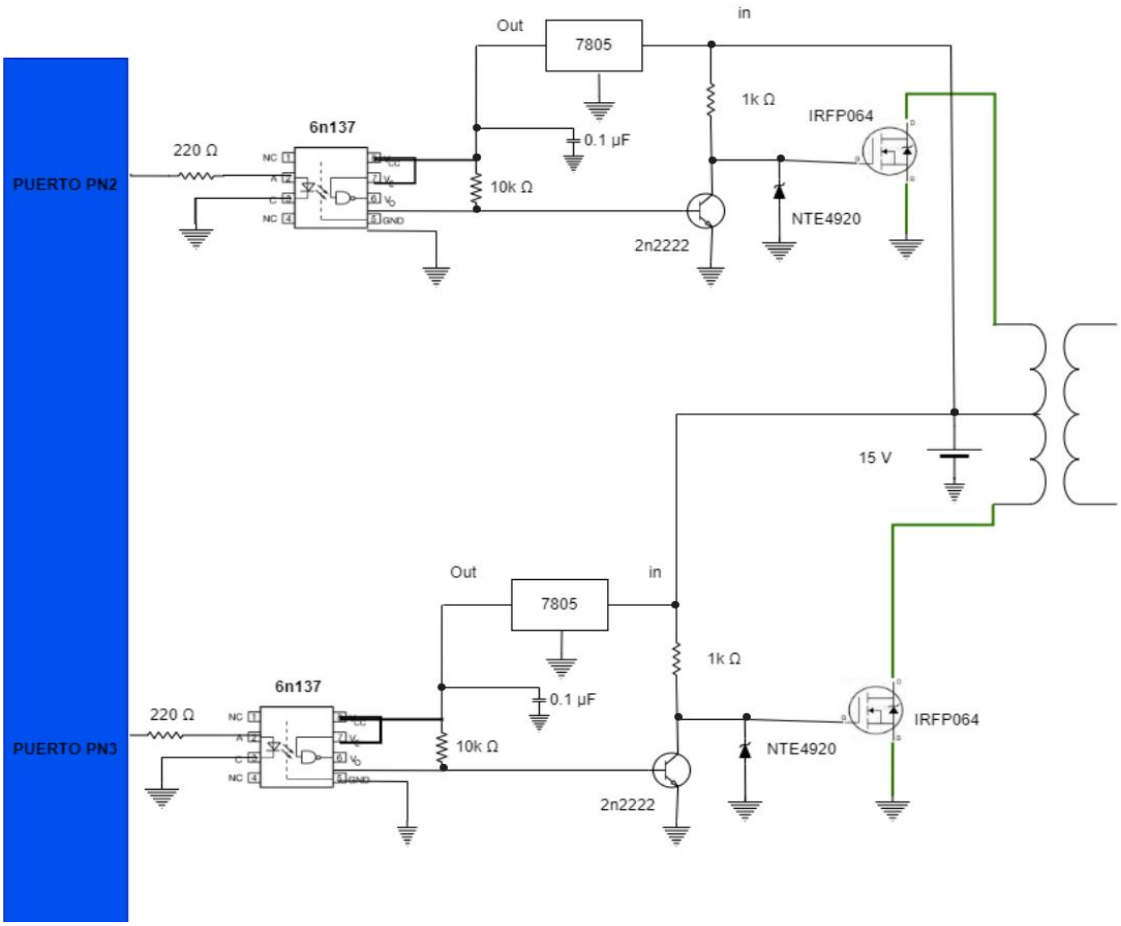


*Ilustración 32 esquemático del inversor*

**6n137 U1**

Función: Aislamiento eléctrico del circuito A con el B. Al encenderse un led infrarrojo interno, cambia el nivel lógico de su salida de 0 a 1 o de 1 a 0,Alimentación de 5VDC.

Sustituto. Solamente el 6n137M dado que el circuito se diseño para contener esa huella.

Proveedor : Ag electrónica, Unit electronics.

Contacto de compra: AG │ Detalles 6N 137 (agelectronica.com)

**U2 7805**

Función: Regulador de voltaje fijo de 5 VCD con salida de 1 A. Alimentará a U1 y servirá como fuente de voltaje a Q1 a través de su resistencia R1 Y a Q2 con su resistencia R2.

Sustituto: Cualquier regulador de la familia 78xx con salida fija de 5 Volts con corriente superior a 300Ma.

Proveedor : Ag electrónica, Unit electronics.

Contacto de Compra: AG | Detalles KA 7805 (agelectronica.com)

**Q1 2N2222A**

Función: Driver de encendido para FET Q2.

Sustituto: Transistor NPN de pequeña señal(BC527,2N3904).

Proveedor: Ag electrónica.

Contacto de compra: AG | Detalles 2N 2222A (agelectronica.com)

**Q2 IRFP064**

Función: MOSFET de potencia.

Sustituto: Con características similares(10%) en corriente y voltaje.

Proveedor: Ag electrónica.

Contacto de compra: AG | Detalles IRFP 064 (agelectronica.com)

**NTE4920**

Función: Transorbedor

Proveedor: Ag electrónica.

Contacto de compra: AG | Detalles NTE 4920 (agelectronica.com)

**R**

Función: R proporciona la corriente de polarización de 10 ma+-5% al led infrarrojo dentro del 6n137.

Sustituto: Puede ser sustituida con R=120 ohms ¼ W

Del analisis:

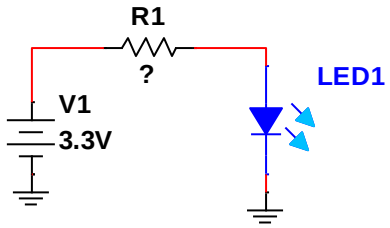


*Ilustración 33 Circuito de análisis*

$$3.3\text{V} = 1.55_{\text{vr}} + 1.75$$

$$\text{R} = \frac{\text{E}}{\text{I}} = \frac{(1.55)}{0.01} = 155 \text{ ohm}$$

$$\text{R} \approx 150 \text{ ohm}$$

$$\text{P} = \text{I}^2 * \text{R} = (0.01)^2 * 150 = 0.015 \text{ W}$$

$$\text{P} \approx \frac{1}{4}\text{W}$$

## 8.2. Teléfono

**Sim808 Gsm Gprs Gps Con Antena Gps**

Función: Actúa como nuestra central de comunicación, para su alimentación necesitaremos una fuente de 5V 2 A. Debido a que cuenta con regulador de voltaje es directo el poder trabajar con el que con otros Sim.

Proveedor: Unit electronics

Contacto de compra: [Sim808 Gsm Gprs Gps Con Antena Gps - UNIT Electronics (uelectronics.com)]