INSTITUTO UNIVERSITARIO AERONAUTICO

CIRCUITOS DIGITALES 3

TRABAJO FINAL:
Control Remoto para transceptor VHF 20A

Autor:
Gutierrez, Agustín
Alberoni, Cristian

Profesores:
Ing. Lancioni, Walter
Dr. Ing. Laprovita, Agustín

Córdoba, 23 de julio de 2015

INDICE

1. OBJETIVOS 3
2. INTRODUCCION 3
2.1 Características generales del VHF 20A 3
2.2 Funcionamiento - entradas y salidas 4
3. DESARROLLO 7
3.1 ESPECIFICACIONES DE DISEÑO 7
3.2 HERRAMIENTAS DE SOFTWARE 7
3.3 MSP430G2553 y Placa LaunchPad MSP-EXP430G2 8
3.4 DISEÑO 9
3.4.1 Módulo – Fuente de alimentación 10
3.4.2 Módulo – Analógico I/O 10
3.4.3 Módulo – Digital I/O 12
3.4.3.1 Circuito esquemático módulo digital 13
3.4.3.2 Circuito esquemático módulo MSP430 16
3.4.4 Diseño de firmware 16
3.4.5 Diseño de gabinete 23
4. RESULTADOS 27
4.1 Módulo – Fuente de alimentación 27
4.2 Módulo – Analógico I/O 30
4.3 Módulo – Digital I/O 33
4.4 Módulo MSP430 36
4.5 Integración de módulos y gabinete 39
5. CONCLUSIONES 41
6. REFERENCIAS 42
7. ANEXO 1 – Firmware Msp430g2553 43
8. ANEXO 2 - Firmware Arduino 44

1. OBJETIVOS

El objetivo del presente trabajo final de la materia Circuitos Digitales 3 de la carrera Ingeniería en Electrónica, es diseñar y construir un control remoto para el equipo de comunicaciones Rockwell Collins VHF 20A (Ilustración) que se encuentra disponible en el laboratorio de comunicaciones aeronáuticas. El mismo deberá respetar los estándares de diseño del equipo transceptor detallado en el manual de mantenimiento del equipo.

Ilustración 1

2. INTRODUCCION

Se detallará la estructura de control del equipo de comunicaciones sin entrar a profundizar en su funcionamiento específico, ya que no es tema de este práctico mostrar el equipo de radio en sí, sino la forma de control del mismo.

2.1 Características generales del VHF 20A

.Este equipo presenta las siguientes características

Tabla 1

Weight:	5.6 lbs.	Dimensions:	3.750"W x 3.50"H x 13.9"L
Related documents:	DO-138, Env Category AD/A/JNG/AAAEXXXXX; TSO C37b and C38b, class 1, FCC rules and regulations, parts 15 and 87	**Temperature**:	-65 to +131 C
Altitude:	55000 ft. max.	**Shock**:	6 g for 11 ms, operating
Frequency Range:	118.000 to 135.975 MHz	**Frequency stability:**	+-0.0015%
Channel Spacing:	25 kHz	**Frequency control:**	2-out-of-5, in accordance w/ ARINC 410
Channel change time:	Less than 50 ms	**Transmit-receive interval:**	Less than 50 ms
Radiated rfi:	Meets RTCA Paper DO-138, DO-109, DO-110	**RF power output:**	20 watts nominal; 16 watts minimum
Output impedance:	52 ohms, unbalanced	**Harmonic Content:**	60 dB down from carrier minimum
Spurious content:	90 dB from carrier minimum	**Modulation capability:**	90% amplitude modulation
Modulator input:	Carbon mic input 0.125 V adjustable, 150 ohms	**Sidetone output**:	100 milliwatts into 600 ohms with 90% modulation
Duty cycle:	1-minute transmit; 4-minute receive	**Noise level:**	50 dB below 85% modulation at 1000 Hz
Audio response:	6-dB variation from 300 to 2500 Hz	**Audio distortion:**	15% max, 85% modualtion
Receiver Sensitivity:	6 dB (signal + noise)/noise for 3-uV signal; 30 dB (signal + noise)/noise for 100-uV signal	**Receiver Selectivity:**	6 db= +-8 kHz min, 60 db= +-17 kHz max (for -001 unit); 6db= +-15 Khz min, 60 dB= +-35 Khz min (for -002 unit)
AGC range:	Maximum 3-dB variation, 5 to 200,000 uV	**Receiver audio output:**	100 mW into 600 ohms, balanced
Receiver audio response:	6-dB variation from 300 to 2500 Hz, 1000-Hz reference	**Receiver audio distortion:**	7% max, 30% modulated signal

En la Tabla se pueden apreciar características marcadas de amarillo, indicando que el equipo de control debe adaptarlas para de esta formar poder usar tanto los micrófonos como los parlantes con los que se cuenta en el laboratorio.

2.2 Funcionamiento - entradas y salidas

Tal como muestra la Tabla , el control de frecuencia se realiza según la norma ARINC 410 la cual define el uso del “2 de 5”. A continuación se muestran en detalles las tablas que describen el control del VHF 20A.

CHANNEL FREQUENCY		VHF-20A/20B MATING CONNECTOR PIN NO								
10-MHz	1-MHz	9(A)	24(B)	10(C)	25(D)	11(A)	23(B)	12(C)	13(D)	14(E)
X1X.XXX		Gnd	Gnd							
X2X.XXX		Gnd		Gnd						
X3X.XXX			Gnd	Gnd						
X4X.XXX			Gnd		Gnd					
X5X.XXX				Gnd	Gnd					
	XX0.XXX						Gnd			Gnd
	XX1.XXX					Gnd	Gnd			
	XX2.XXX					Gnd		Gnd		
	XX3.XXX						Gnd	Gnd		
	XX4.XXX						Gnd		Gnd	
	XX5.XXX							Gnd	Gnd	
	XX6.XXX							Gnd		Gnd
	XX7.XXX								Gnd	Gnd
	XX8.XXX					Gnd			Gnd	
	XX9.XXX					Gnd				Gnd

CHANNEL FREQUENCY		VHF-20A/20B MATING CONNECTOR PIN NO						
0.1-MHz	0.01-MHz	15(A)	22(B)	16(C)	17(D)	18(E)	19(C)	20(D)
XXX.0XX			Gnd			Gnd		
XXX.1XX		Gnd	Gnd					
XXX.2XX		Gnd		Gnd				
XXX.3XX			Gnd	Gnd				
XXX.4XX			Gnd		Gnd			
XXX.5XX				Gnd	Gnd			
XXX.6XX				Gnd		Gnd		
XXX.7XX					Gnd	Gnd		
XXX.8XX		Gnd			Gnd			
XXX.9XX		Gnd				Gnd		
	XXX.X00							
	XXX.X25						Gnd	
	XXX.X50						Gnd	Gnd
	XXX.X75							Gnd

En la Ilustración se puede ver el conector trasero que posee y en la tabla siguiente se muestra el pinout correspondiente, el cual se deberá respetar.

Función		DB-25	VHF
10 MHz	A	1	9
	B	2	24
	C	3	10
	D	4	25
1 MHz	A	5	11
	B	6	23
	C	7	12
	D	8	13
	E	9	14
0,1 MHz	A	10	15
	B	11	22
	C	12	16
	D	13	17
	E	14	18
0,01 MHz	C	15	19
	D	16	20
Masa comm free		17	3
		18	-
		19	-
		20	-
Audio H		21	5
Audio L		22	6
Squelch		23	4
Mic. Carbón		24	7
PTT		25	8

Ilustración 2

3. DESARROLLO

Para el desarrollo se partirá de las especificaciones de diseño fijadas y discutidas para que el equipo de control remoto pueda, luego de ser construido, cumplir con la mayoría de las funciones del control remoto comercial del transceptor.

Posteriormente se mostrará el esquema de diseño general de las partes que componen el sistema y seguido a esto se detallarán cada uno de los circuitos.

Finalmente se dedicará un punto aparte al desarrollo del firmware

3.1 ESPECIFICACIONES DE DISEÑO

A continuación se detallan las especificaciones de diseño requeridas:

- ARINC 410 (sintonía) 2 de 5
- Control de sintonía cada 25KHz
- Control On/Off
- Control Squelch
- Control de audio continuo
- Salida de Audio de 8Ohm
- Entrada Micrófono Electret
- Display para mostrar información
- Selector de frecuencia
- Alimentación según norma MIL-STD-704 28v
- Gabinete cerrado
- Informe técnico
- Manual de operaciones
- Manual de instalación
- Procedimiento de prueba
- Resultado de las pruebas

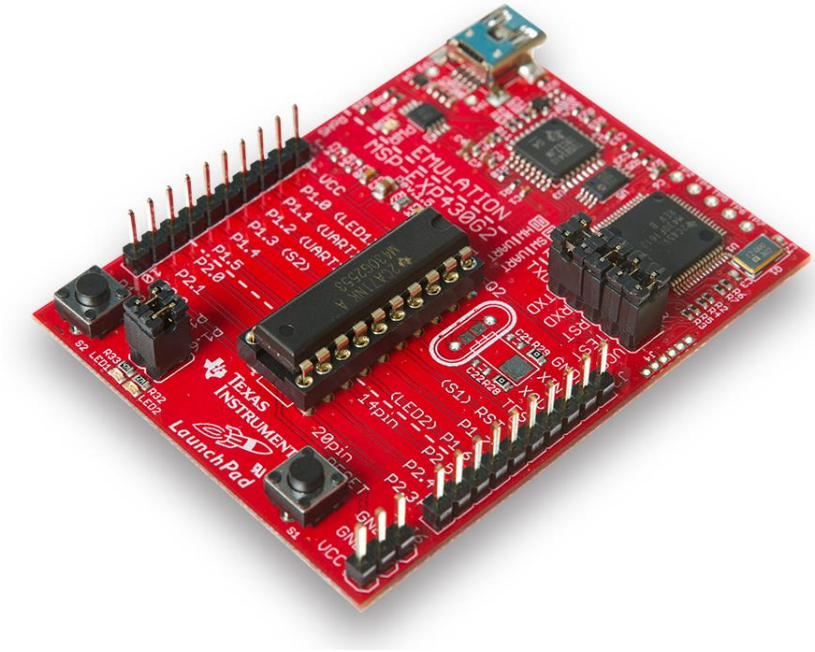
3.2 HERRAMIENTAS DE SOFTWARE

- Altium Designer 10
- Altium Designer 14
- Kicad
- SolidWorks 2013
- IAR Embedded Workbench for MSP430 IDE 5.60.7
- Multisim 11.0

- Doxygen

3.3 MSP430G2553 y Placa LaunchPad MSP-EXP430G2

En este proyecto se decidió utilizar el microcontrolador msp430g2553 por la cantidad de documentación disponible en la red y porque cumple con todos los requerimientos de memoria, procesamiento, puertos de entrada/salida necesarios. Para lo cual se decidió adquirir la placa LaunchPad MSP-EXP430G2 que cuenta con el microcontrolador anteriormente mencionado y con su programador/debuger por Spy-Bi-Wire, como se ve en la imagen.

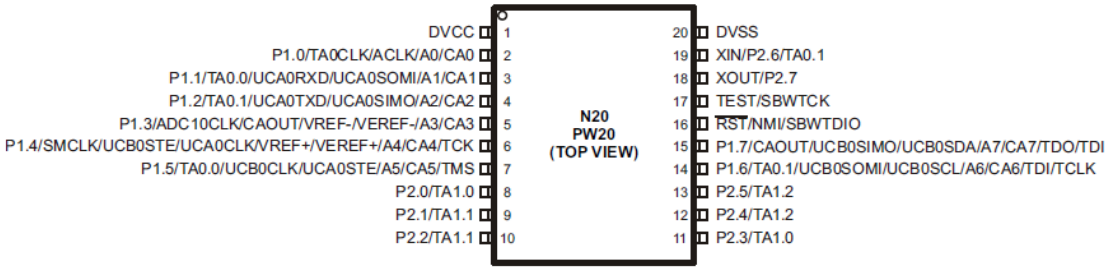


A continuación se describirá al microcontrolador:

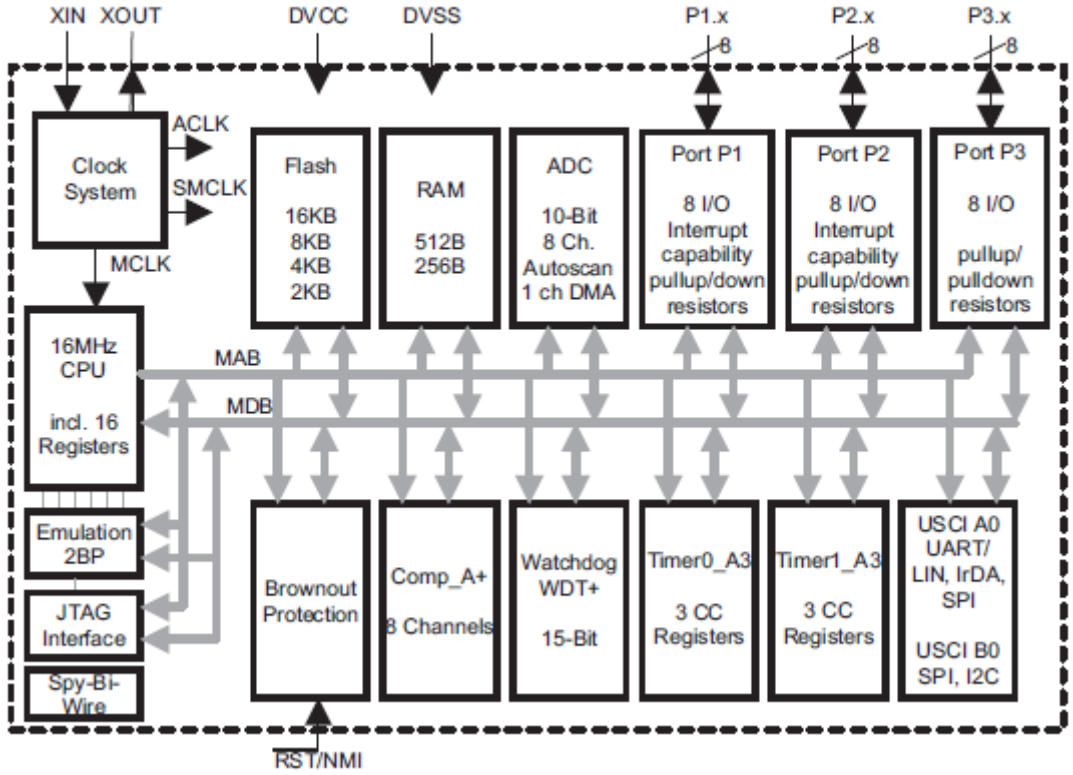
El microcontrolador MSP430G2553 de Texas Instruments (Ver Figura), combina con los cinco modos de bajo consumo está optimizado para lograr la vida extendida de la batería en aplicaciones de mediciones portátiles.

El dispositivo presenta un CPU de 16 bits RISC, registros de 16 bits y generadores constantes que contribuyen a la máxima eficiencia del código. La serie MSP430G2 contiene dos temporizadores integrados de 16 bits con 3 módulos de comparación y captura asociados a cada uno, una interfaz de comunicación serial universal, un conversor A/D de 10 bits con referencia integrada y 16 Puertos I/O, como se muestra en la figura.

Device Pinout, MSP430G2x13 and MSP430G2x53, 20-Pin Devices, TSSOP and PDIP

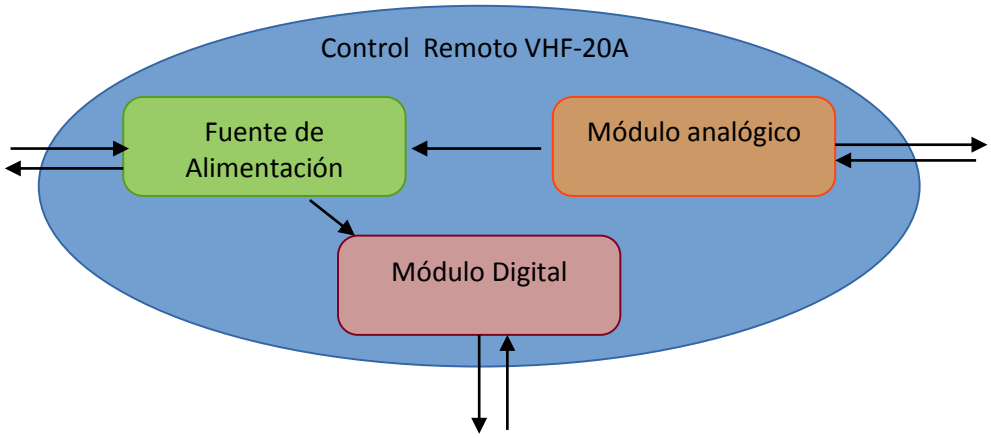


Functional Block Diagram, MSP430G2x53



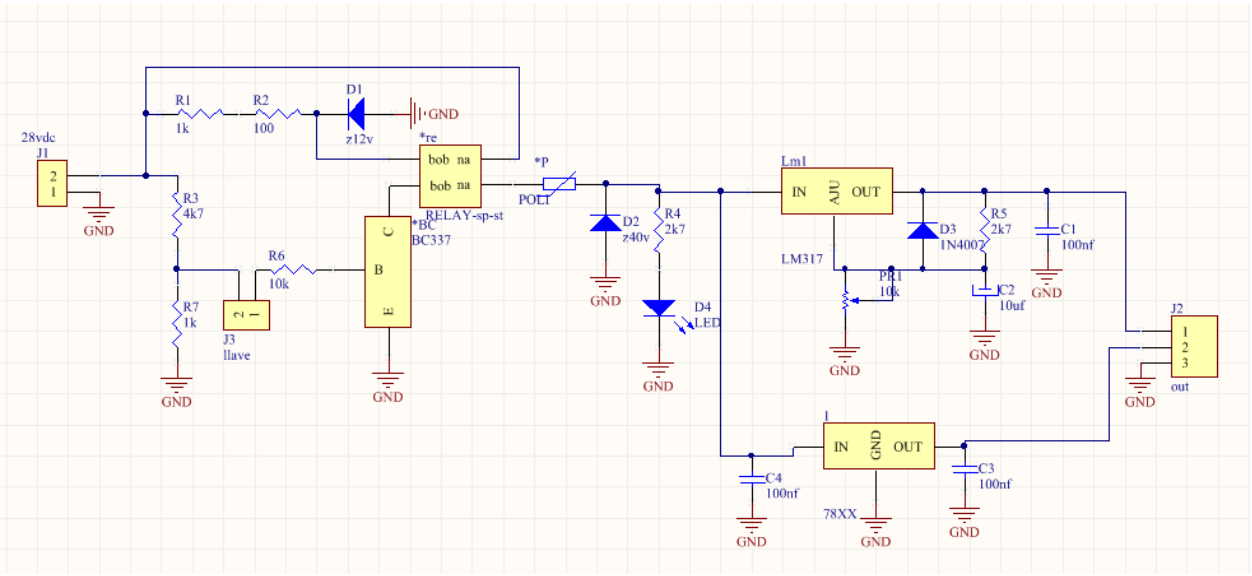
3.4 DISEÑO

El diseño del circuito se dividió en tres módulos tal como muestra la figura



3.4.1 Módulo – Fuente de alimentación

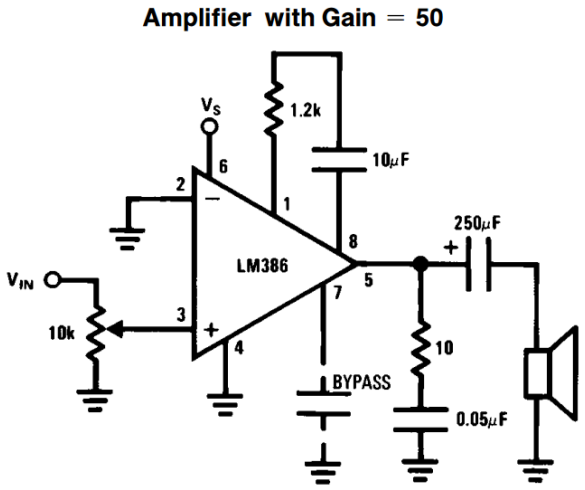
.



Circuito de fuente de alimentación

3.4.2 Módulo – Analógico I/O

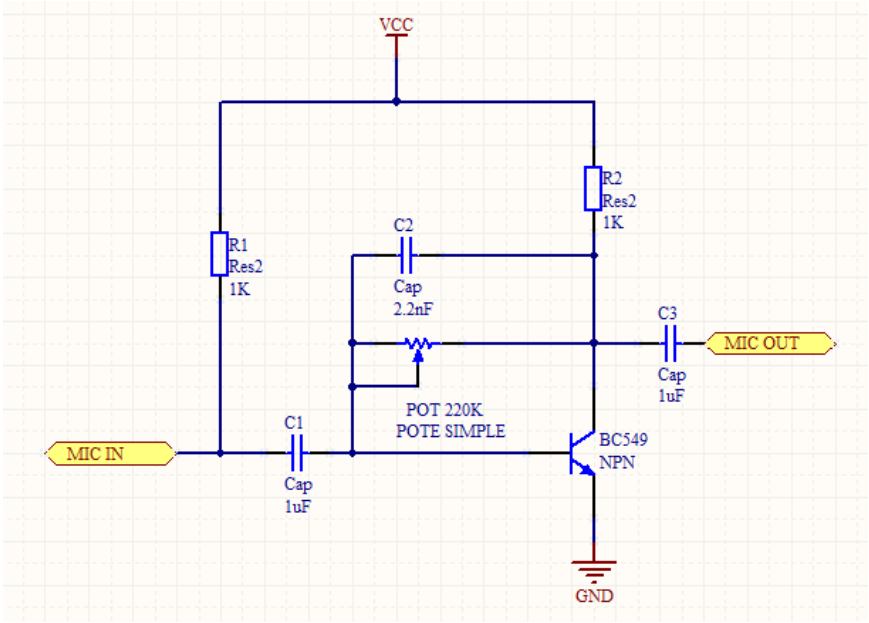
Se amplificó las señales procedentes del VHF agregándoles un control de volumen, además se realizó una adaptación de impedancia para las señales de audio colocando alta impedancia en la entrada y a la salida del amplificador se presentan los 8Ω que se dirigirán a los audífonos.



Circuito amplificador de audio

Para realizar estos objetivos se utilizó el integrado LM386 el cual es un amplificador diseñado para utilizarse en aplicaciones de bajo consumo y logra brindar una potencia de 325 mW máxima. La configuración utilizada fue extraída de la hoja de datos del componente para obtener una ganancia de 50 veces.

Por otra parte se debía adaptar la entrada del VHF para el micrófono la cual estaba diseñada para que se utilicen micrófonos de carbón y no los de uso actual (Electret), para ello se utilizó el siguiente circuito.

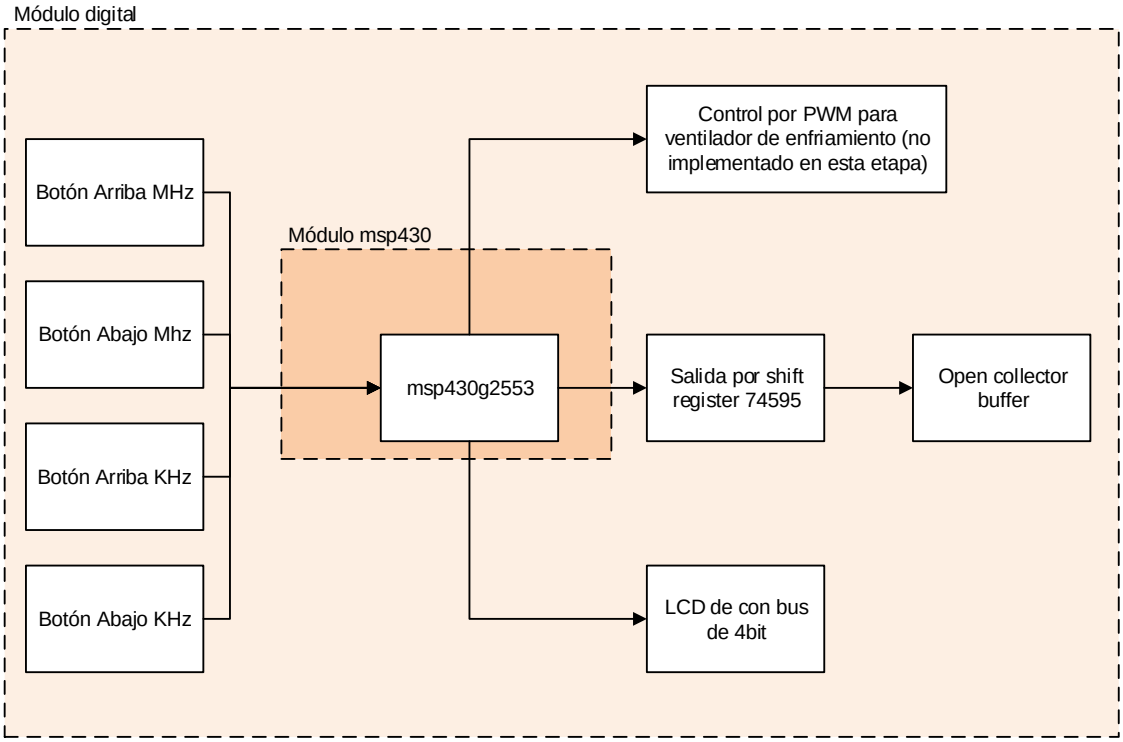


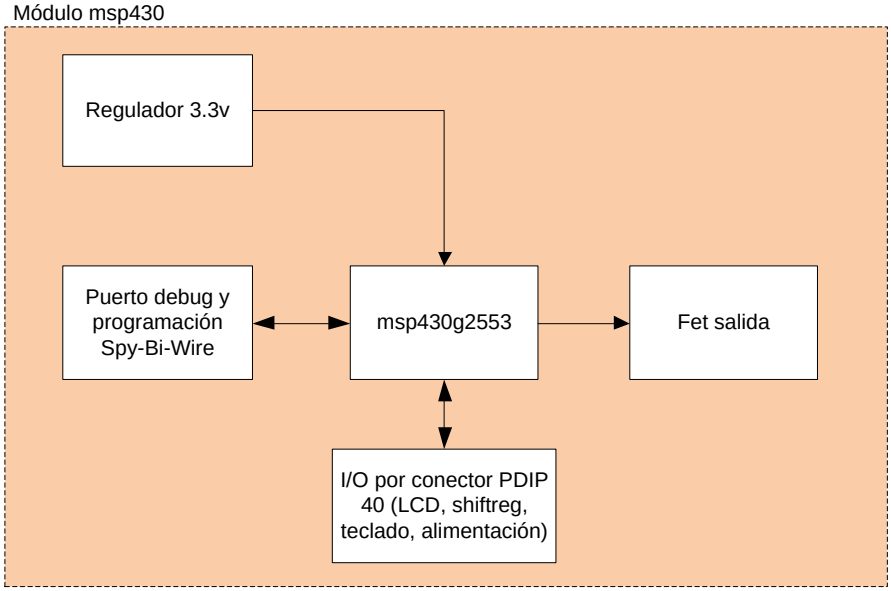
Circuito adaptador de micrófono

3.4.3 Módulo – Digital I/O

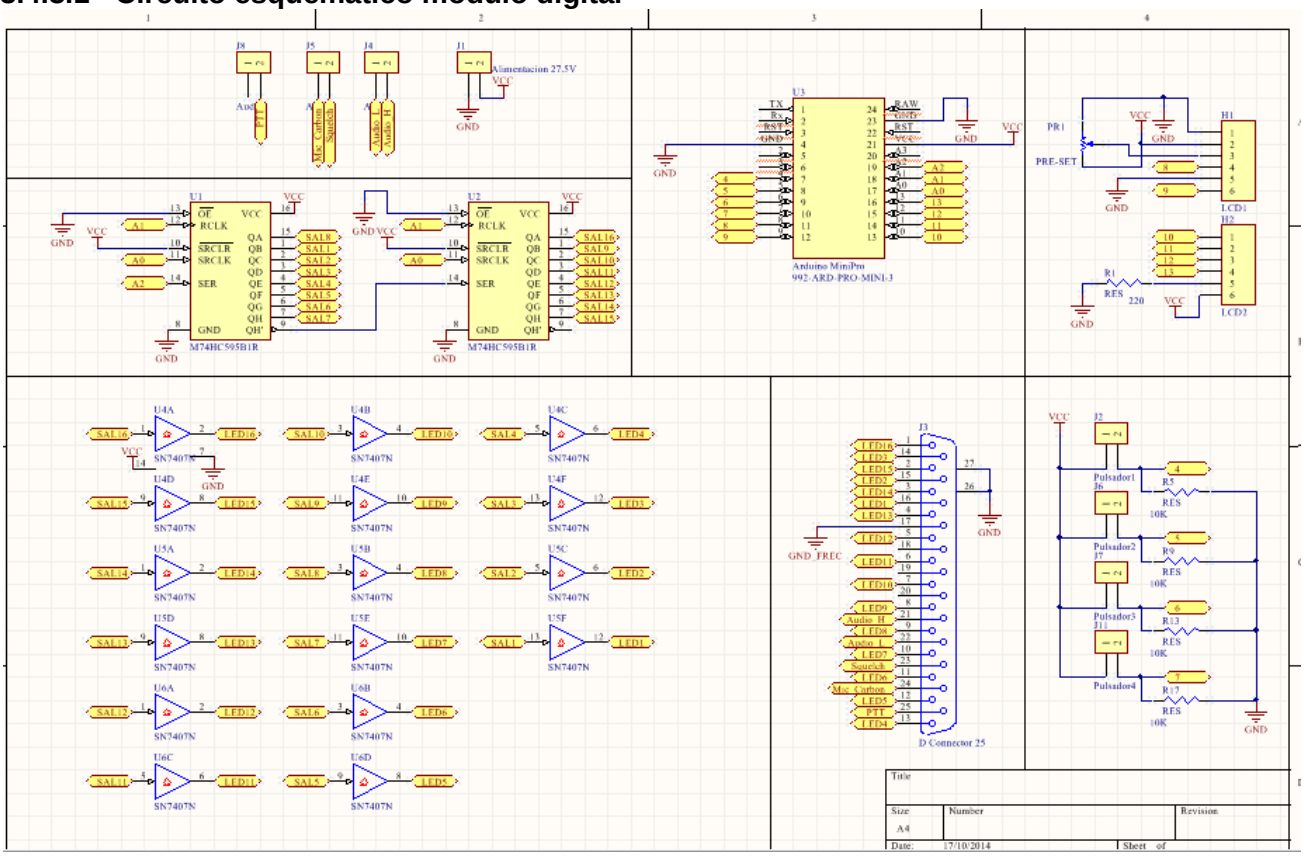
A continuación se muestra un circuito esquemático del módulo digital. Originalmente este circuito fue diseñado e implementado con un microcontrolador ATMEGA328 incluido en la placa Arduino Mini Pro (el cual también tiene el firmware necesario en ANEXO) para la selección de frecuencia y control de las salidas que realizan la codificación “2 de 5” según ARINC 410. A fin de cumplir con los requisitos de la materia y demostrar lo aprendido, se decidió cambiar el módulo de Arduino Mini Pro por uno de diseño propio que contenga a un microcontrolador MSP430 de Texas Instruments, específicamente se decidió colocar un msp430g2553.

Esquema de diseño

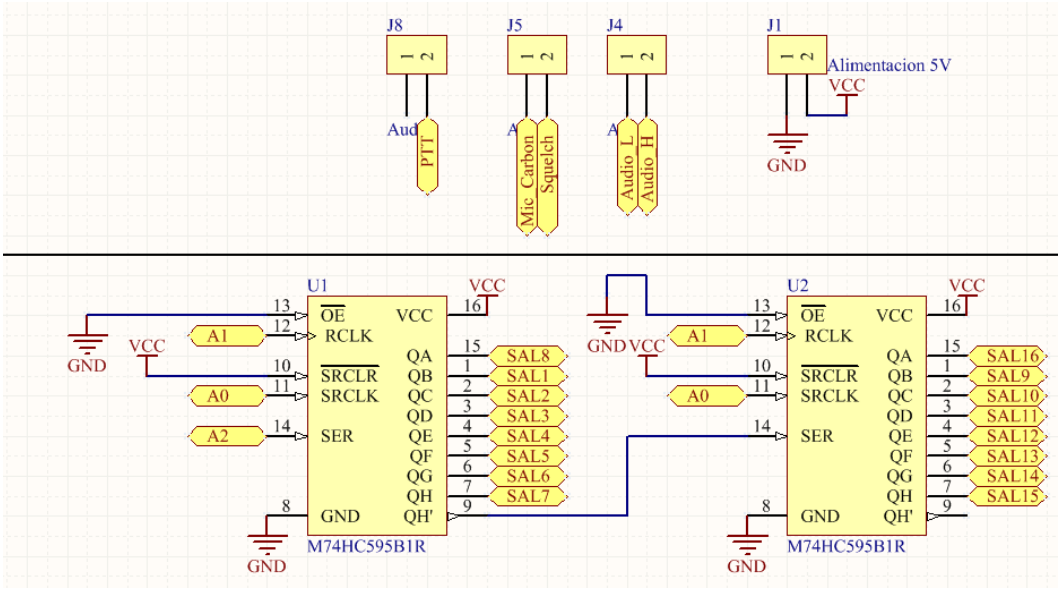




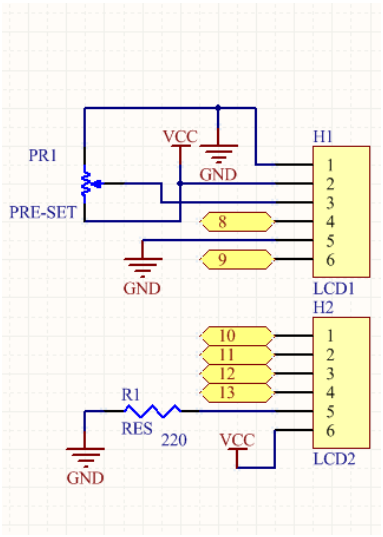
3.4.3.1 Circuito esquemático módulo digital



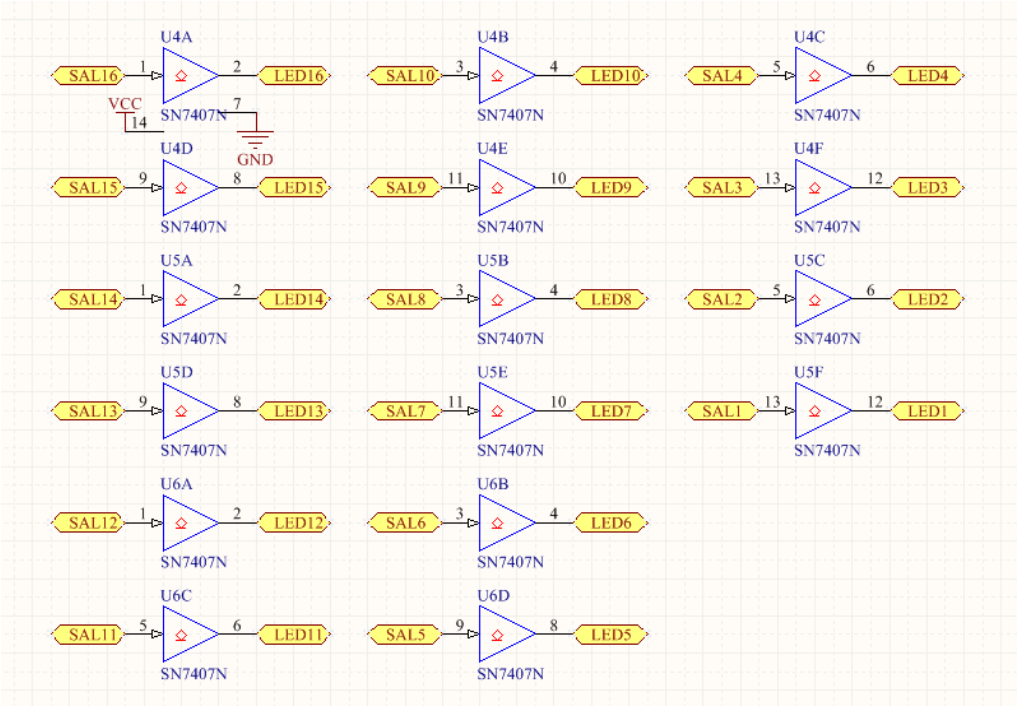
En este esquemático general se puede ver la placa Arduino mini pro, pero en realidad es solo el conector DIP40 que se usó como sócalo para ambas placas, de manera de poder intercambiarlas.



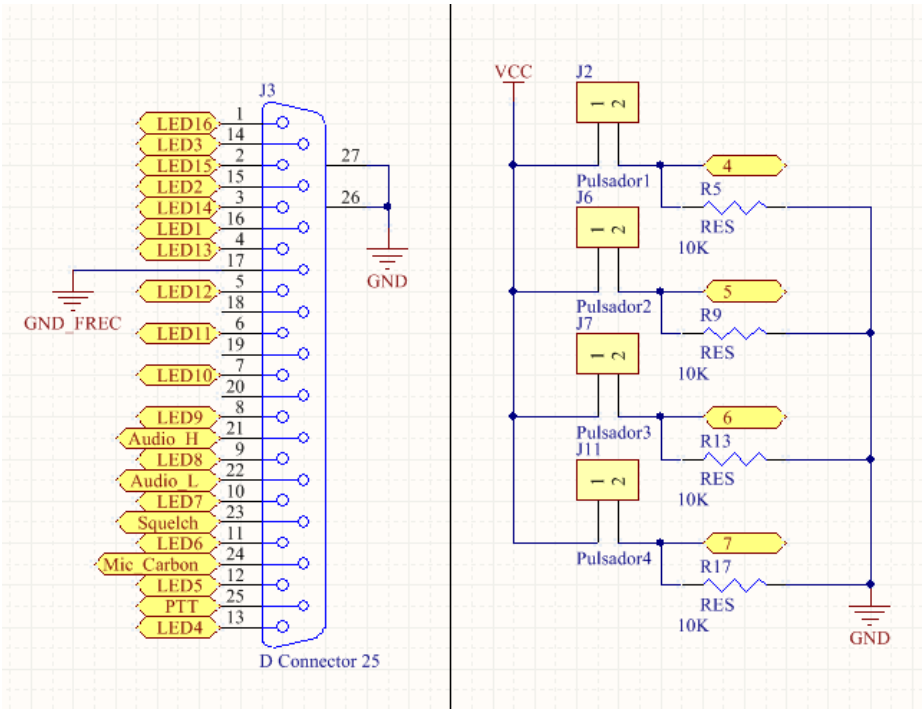
Circuito con 74HC595 para salidas



Circuito de salidas para LCD

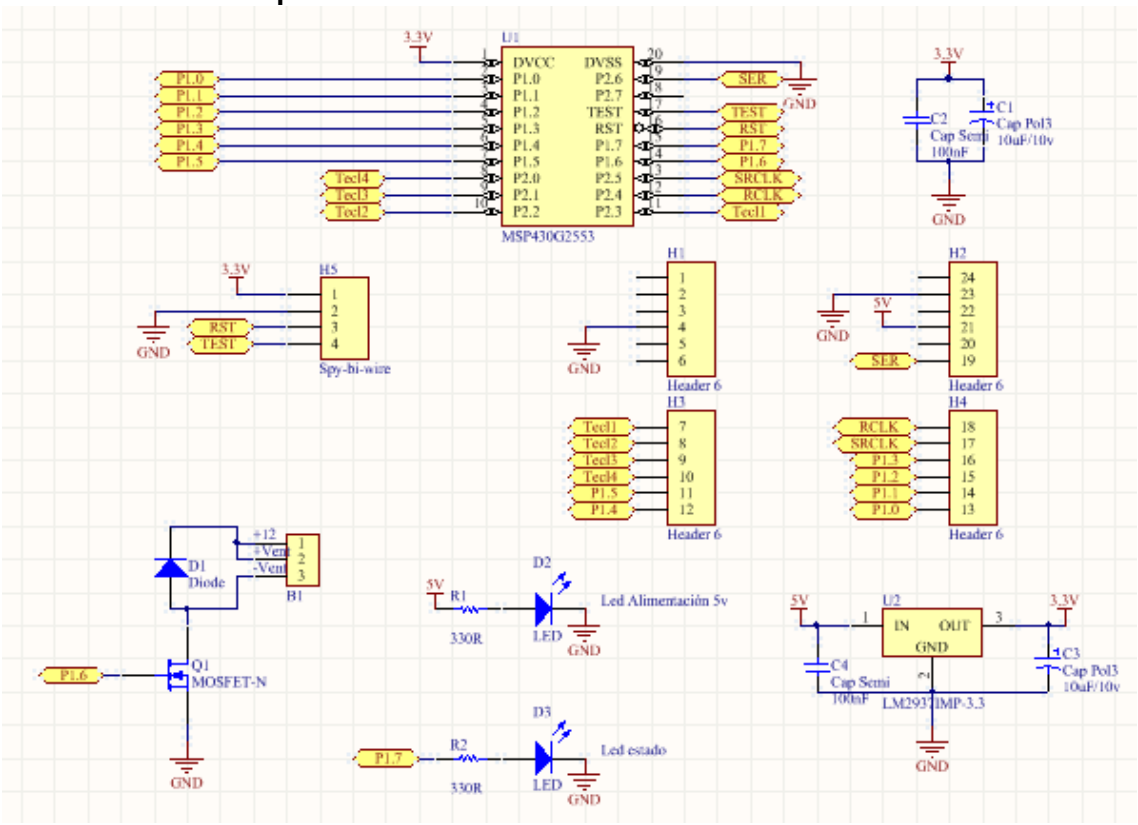


Circuito de Buffer Open Colector 74VLC07



Circuito de Conector de salida DB25 y circuito de pulsadores de entrada

3.4.3.2 Circuito esquemático módulo MSP430



3.4.4 Diseño de firmware

Funciones

El equipo debe poder operar en dos modos: Modo normal y modo de test

- Modo Normal: Iniciar por defecto en este modo. Se debe poder modificar la frecuencia mostrada en el display LCD de 16x1 y dar la correcta salida a esa frecuencia, a su vez debe indicar su estado a través de un led que “blinkea” cada cierto tiempo.
- Modo de test: Ingresa cuando son presionado los botones de arriba de MHz y arriba de KHz. Para salir de este modo se debe presionar el botón de debajo de KHz. Se debe poder ejecutar rutinas de prueba para las salidas digitales que cumplen con el procedimiento de ensayo del control remoto. A su vez el led de actividad debe aumentar la frecuencia de “blinkeo” para indicar que se está en este modo.

Implementación:

Se propuso como objetivo puramente académico, implementar el firmware usando otras técnicas distintas al procesamiento secuencial (bucle scan). En el proceso se analizaron las alternativas de “Primer plano/Segundo Plano” (Foreground / Background), Sistemas operativos en tiempo real cooperativos (RTOS cooperative) y Sistemas operativos en tiempo real expropiativos (RTOS preemptive) que aportan sus ventajas y desventajas con su respectiva complejidad de menor a mayor. Como resultado se decidió optar por un manejador de tareas cooperativo que tiene la ventaja de tener un manejador de tareas (scheduler) más sencillo que los RTOS expropiativos pero las tareas deben estar cuidadosamente programadas de manera que devuelvan el control del procesador al manejador.

Se decidió tomar un manejador de tareas llamado LMOS, realizado originalmente para los microcontroladores PIC de baja gama como son las familias 16f y 18f, ya que es muy pequeño y provee de las siguientes características:

- Inicialización y carga de tareas
- Demoras del tipo iddle wait
- Sincronización con semáforos con timeout
- Manejo de estados de tarea (destroyed, stopped, running, waiting y delayed)
- Recursos de gestión de tareas (cuanto se utiliza del micro, tiempo de uso por micro, etc)

Se realizó la adaptación de LMOS para que funcione en los microcontroladores de la familia MSP430 y posteriormente se escribieron las bibliotecas correspondientes al uso de los periféricos de proyecto como son display LCD, teclado, manejo de registros de desplazamiento y led de actividad.

El manejador de tareas LMOS cuentas con los siguientes métodos para el control del scheduler:

Control del scheduler:

- schedInit() : Maneja la inicialización del scheduler
- vTimerInit(10Mhz, 1000) : Inicializa y configura el reloj del sistema (reloj de sistema, tick del sistema)

- intEnablePriority(ON) : Habilitación de prioridades en las interrupciones
- intLowEnable(ON) : Habilitación de las interrupciones de baja prioridad
- intHighEnable(ON) : Habilitación de las interrupciones de alta prioridad
- schedule() : Método de contextualización

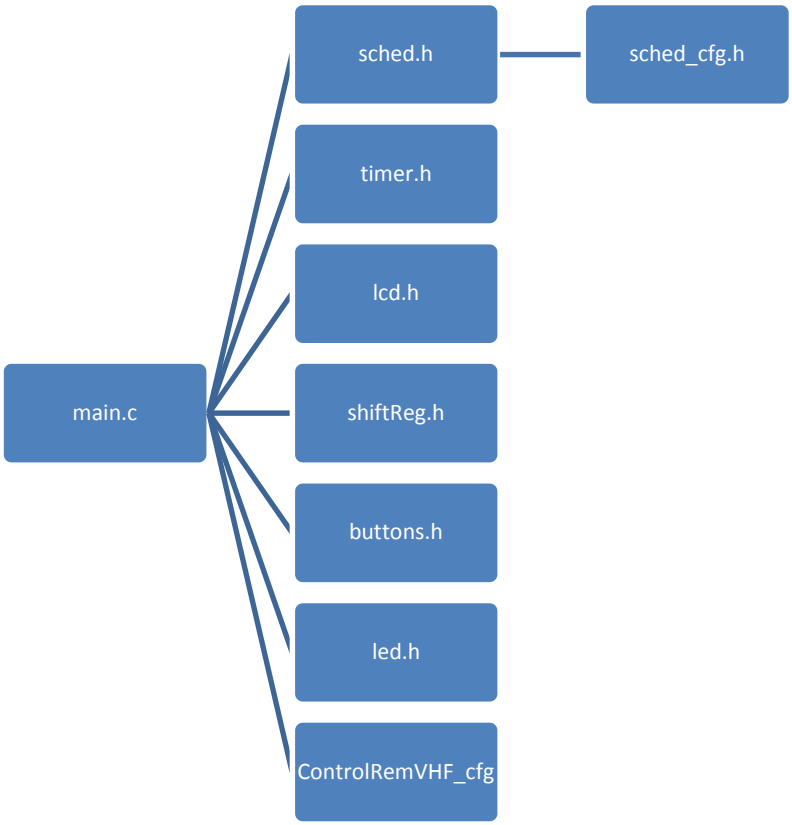
Control de la tarea:

- schedTaskCreate(vTask1, TCBP_TASK_1) : Creación de una tarea (función vTask1) que utiliza el Task control block TCBP_TASK_1
- schedTaskCurrDestroy() : Destruir la tarea en curso (desde una tarea)
- schedTaskDestroy(TCBP_TASK_1) : Destruir la tarea TCBP_TASK_1
- schedDelay(n) : Dormir la tarea actual por n ticks de sistema
- schedStep(ctx) : Guardar el contexto actual antes de entregar el procesador al scheduler
- schedGetDelay(TCBP_TASK_1) : Consultar la demora actual de la tarea (tiempo antes de despertar)
- schedGetCurrDelay() : Consultar la demora de la tarea actual
- schedTaskStop(TCBP_TASK_1) : Detener la tarea TCBP_TASK_1 por tiempo indefinido
- schedTaskCurrStop() : Detener la tarea actual por tiempo indefinido
- schedTaskStart(TCBP_TASK_1) : Arrancar tarea TCBP_TASK_1
- schedTaskState(TCBP_TASK_1) : Consultar el estado actual de una tarea

Comunicación entre tareas y el hardware:

- schedSignalSemInt(EVENT1) : Activar el semáforo EVENT1 desde una interrupción
- schedSignalSem(EVENT1) : Activar un semáforo EVENT1 desde una tarea
- schedWait(TIMEOUT, EVENT1) : Esperar al semáforo EVENT1 hasta un máximo de TIMEOUT ticks
- schedTry(EVENT1) : Consumir (1) contabilizaciones del evento EVENT1 sin perder el contexto si no tiene alguna.

Diagrama de organización de proyecto:

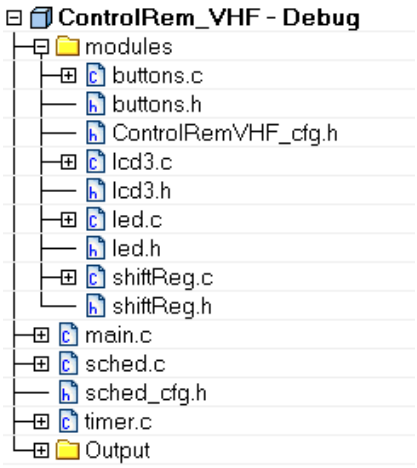


main.c: Contiene la implementación de la aplicación.

sched.h: Módulo del manejador (scheduler module) encargado de la contextualización del Sistema.

timer.h: Módulo encargado del manejo del tiempo (interrupción, ticks del Sistema)

sched_cfg.h: Contiene las definiciones para la configuración del Sistema



Configuración

En este archivo se configura:

- La cantidad máxima de tareas concurrentes (SCHED_TASK_MAX)
- La máxima cantidad de eventos a manejar simultáneamente (SCHED_EVENT_MAX)
- El tamaño de los registros para eventos contadores (SCHED_EVENT_SIZE)
- El tamaño de los registros para temporización (SCHED_DELAY_SIZE)
- El tamaño de los registros para la cantidad de contextos permitidos (SCHED_STEP_SIZE)
- Habilitación de reentrancia (SCHED_REENTRY)
- Nivel de protección al scheduler (SCHED_PROTECT_LEVEL)
- Tamaño de los registros para las prioridades de las tareas (SCHED_PRIORITY_LEVELS)

Led.h: Biblioteca para manejo de led de actividad.

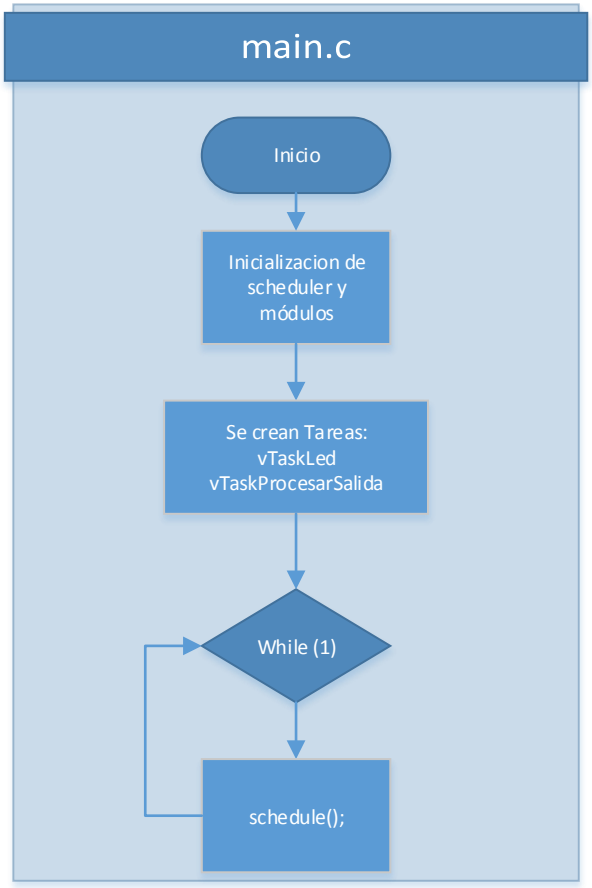
Lcd3.h: Biblioteca para el manejo del display LCD 16x1 y 16x2 caracteres alfanuméricos

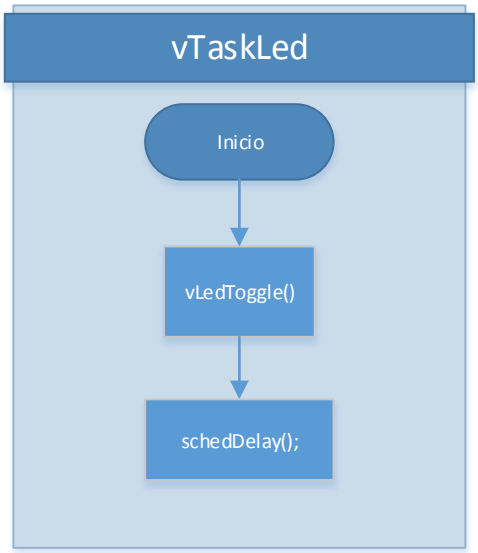
shiftReg.h: Biblioteca para el manejo de los registros de desplazamientos 74595

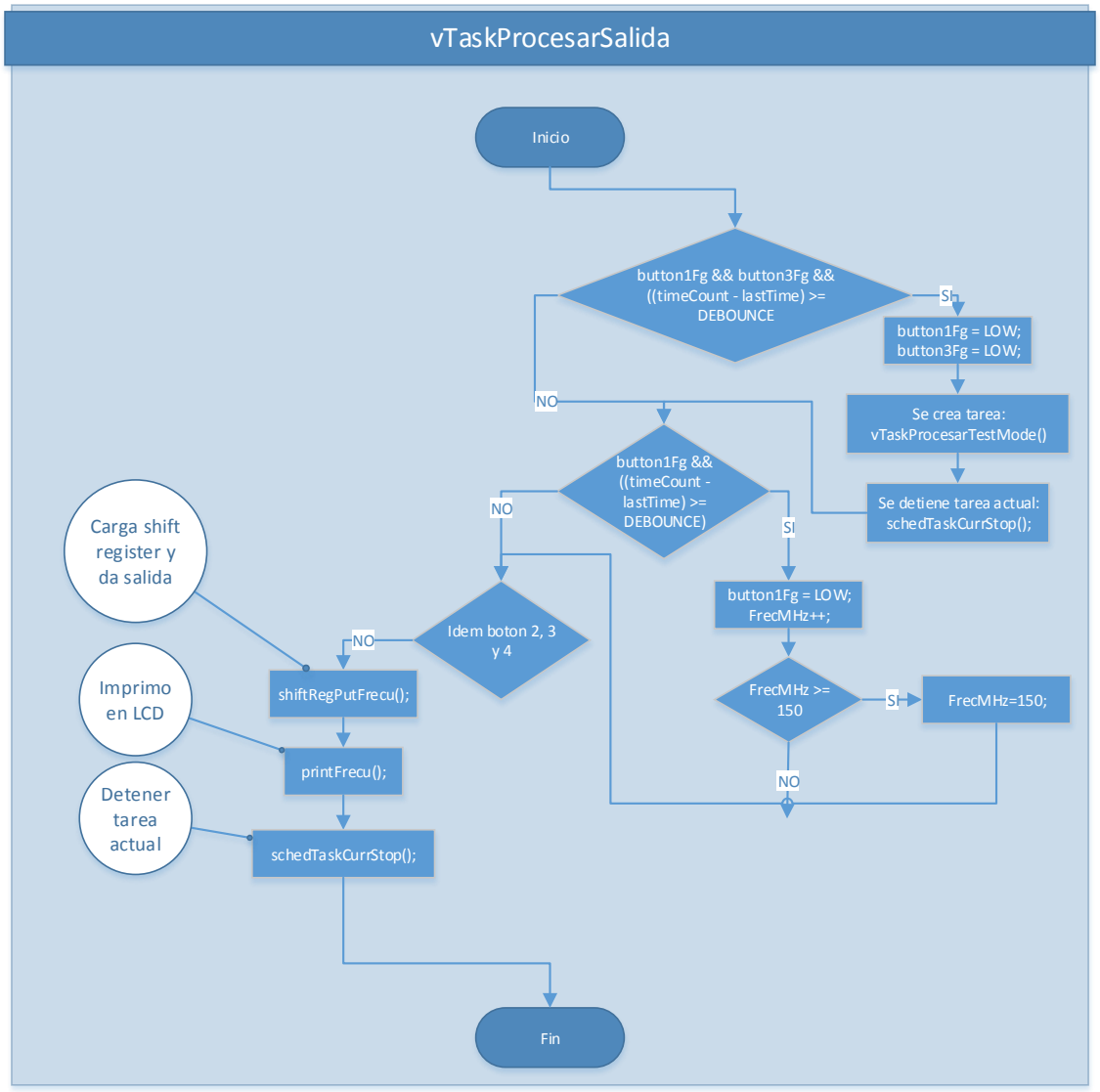
buttons.h: Biblioteca para el manejo de teclado de 4 teclas.

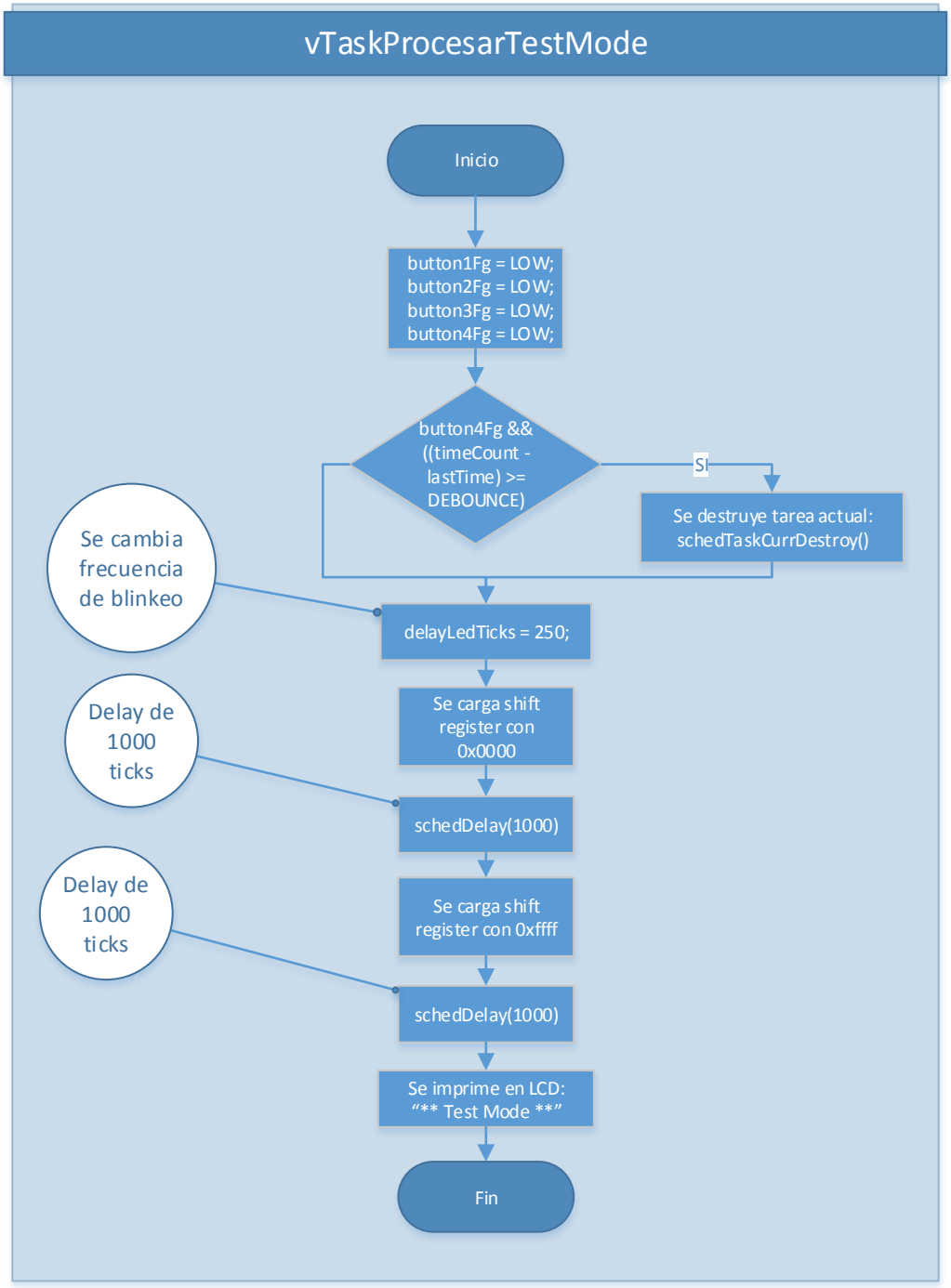
ControlRemVHF_cfg.h: Biblioteca de definiciones de aplicación para el control remoto.

Diagrama de Flujo principal:


main.c
Inicio
Inicializacion de scheduler y módulos
Se crean Tareas: vTaskLed vTaskProcesarSalida
While (1)
schedule();

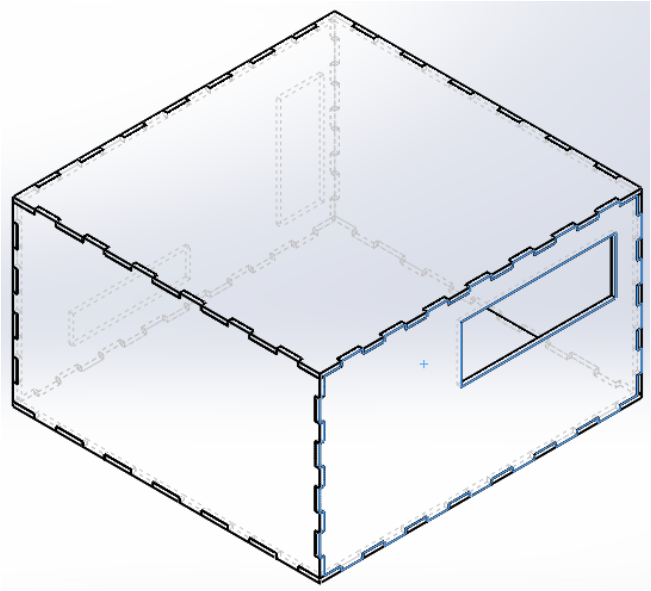

vTaskLed
Inicio
vLedToggle()
schedDelay();

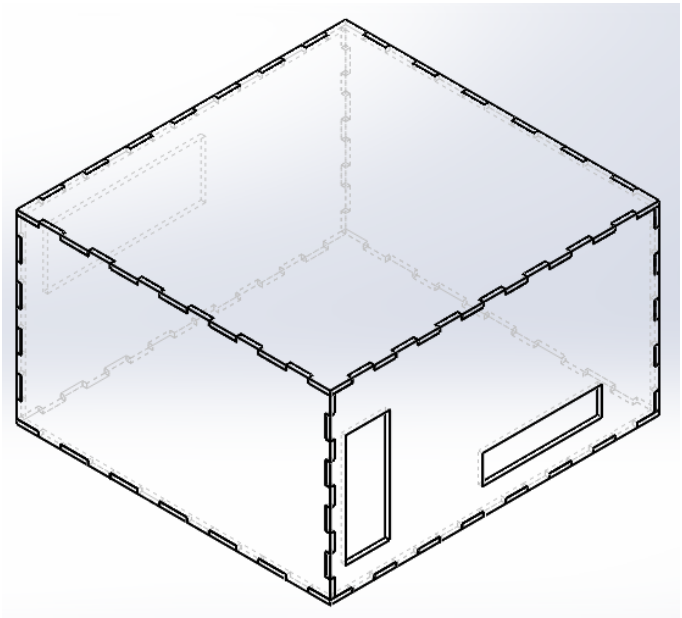

vTaskProcesarSalida
Inicio
button1Fg && button3Fg && ((timeCount - lastTime) >= DEBOUNCE
SI
button1Fg = LOW;
button3Fg = LOW;
Se crea tarea: vTaskProcesarTestMode()
NO
button1Fg && ((timeCount - lastTime) >= DEBOUNCE)
Se detiene tarea actual: schedTaskCurrStop();
NO
SI
button1Fg = LOW;
FrecMHz++;
Idem boton 2, 3 y 4
NO
FrecMHz >= 150
SI
FrecMHz=150;
NO
Carga shift register y da salida
shiftRegPutFrecu();
Imprimo en LCD
printFrecu();
Detener tarea actual
schedTaskCurrStop();
Fin



3.4.5 Diseño de gabinete

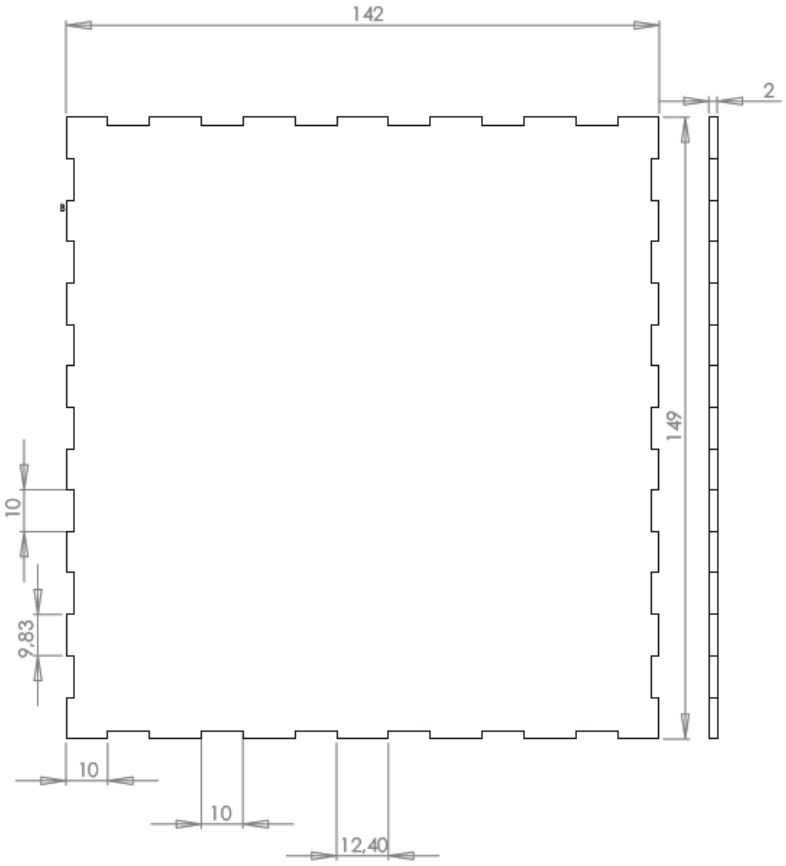
Fotos de Gabinete en SolidWorks de vista frontal y posterior



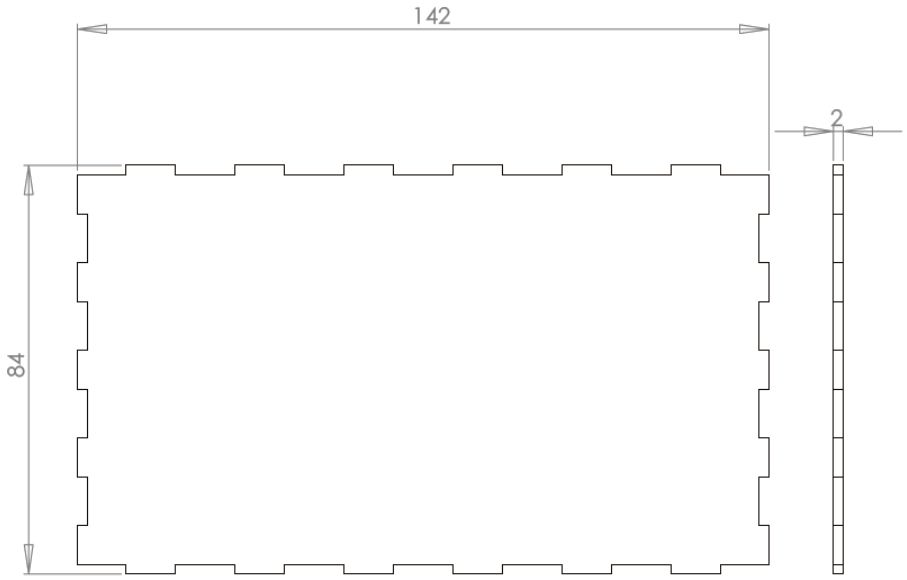


Planos constructivos del gabinete

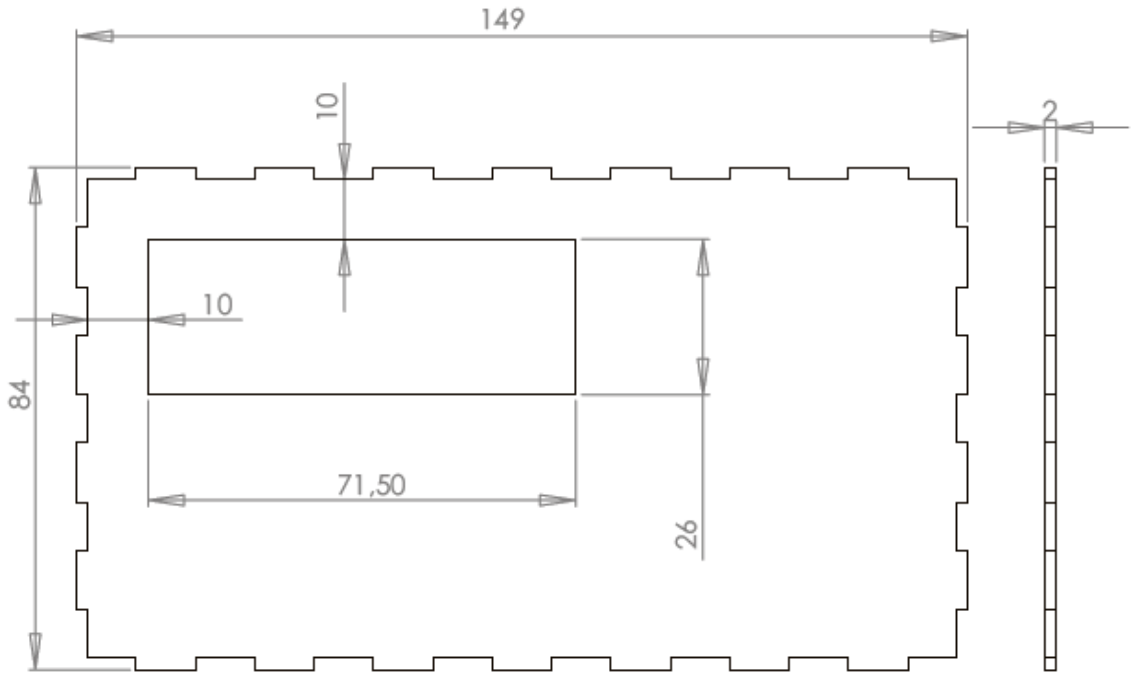
Tapa base y tapa techo



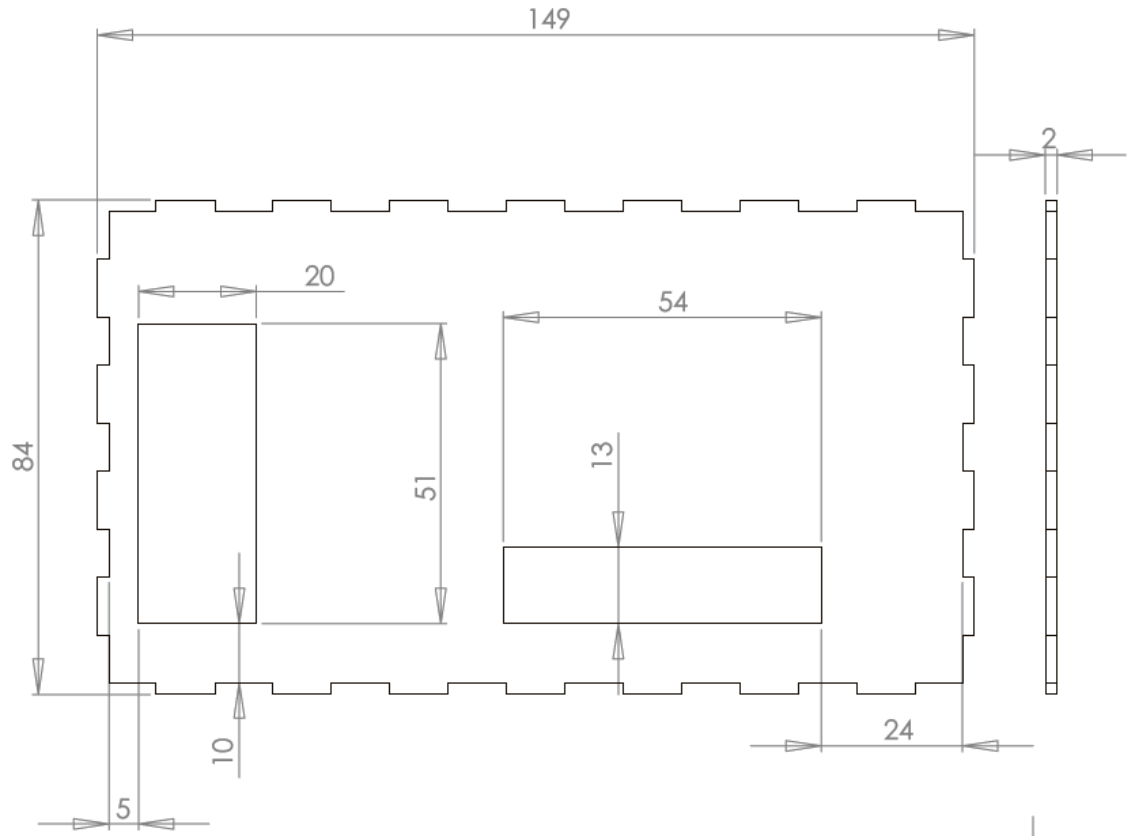
Tapa Laterales X2



Tapa de Frente



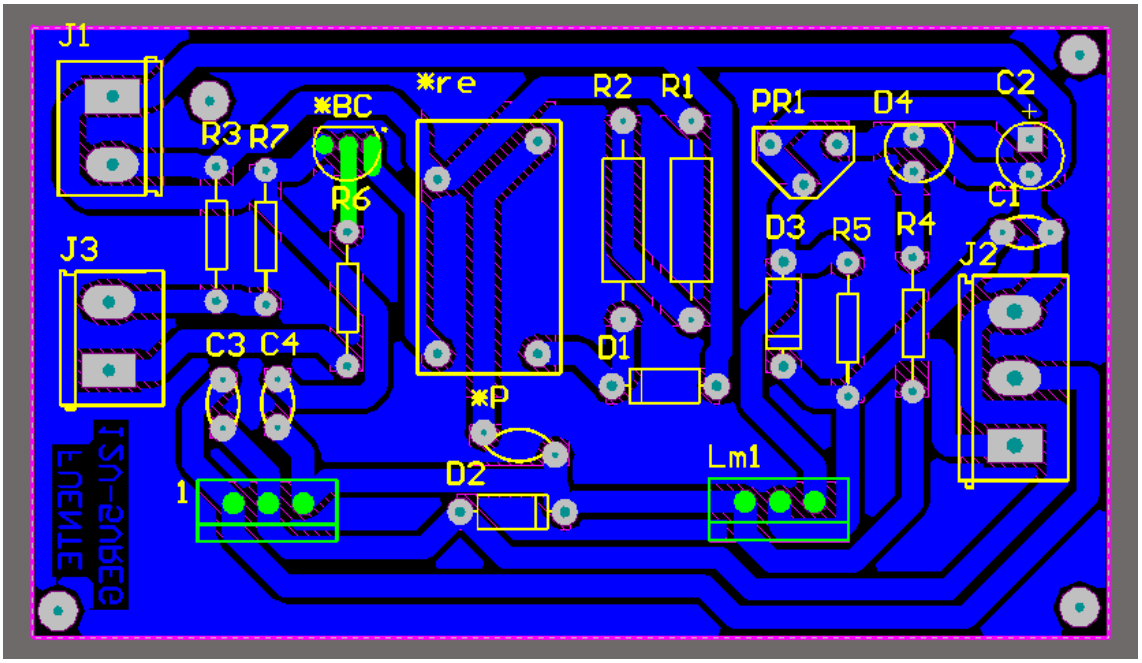
Tapa Trasera



4. RESULTADOS

4.1 Módulo – Fuente de alimentación

A continuación se muestra la materialización de los circuitos a implementar, los cuales fueron realizados utilizando Altium Designer Release 10.



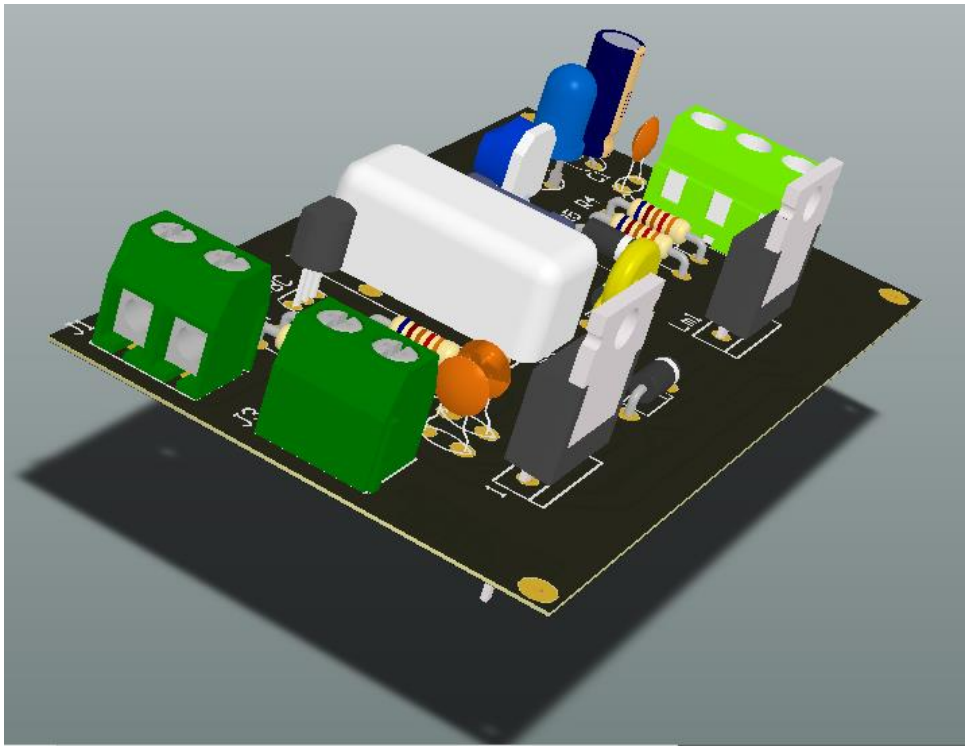
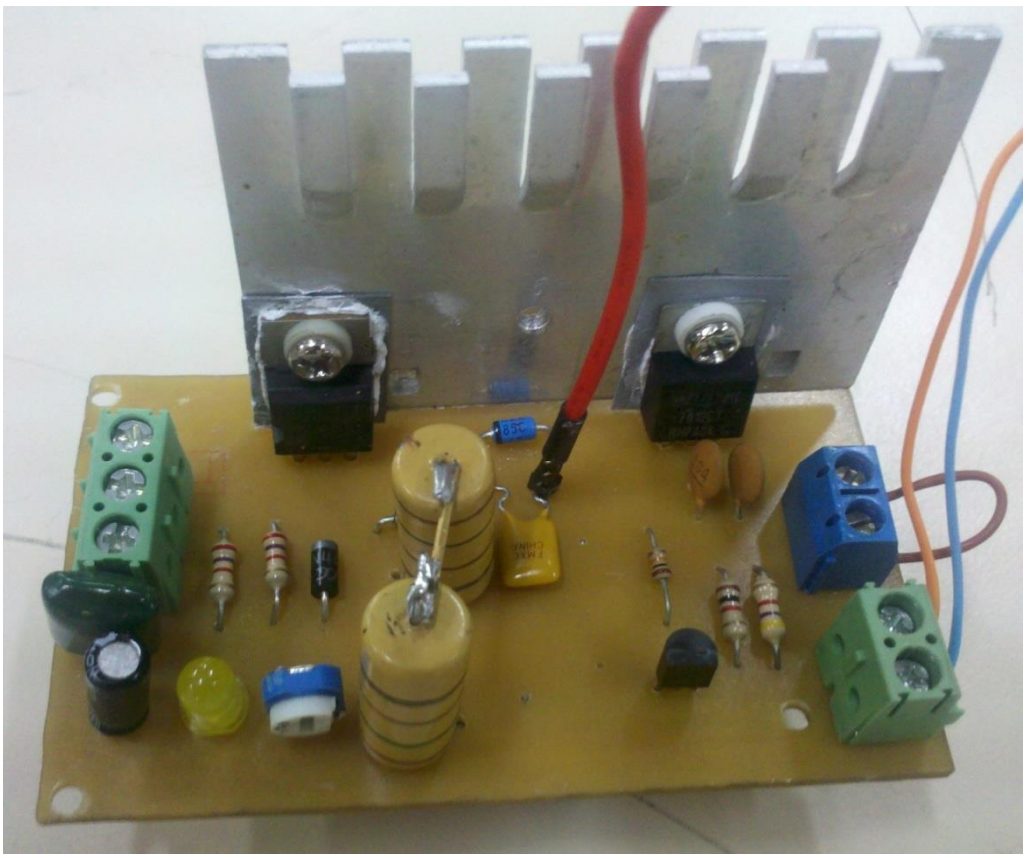
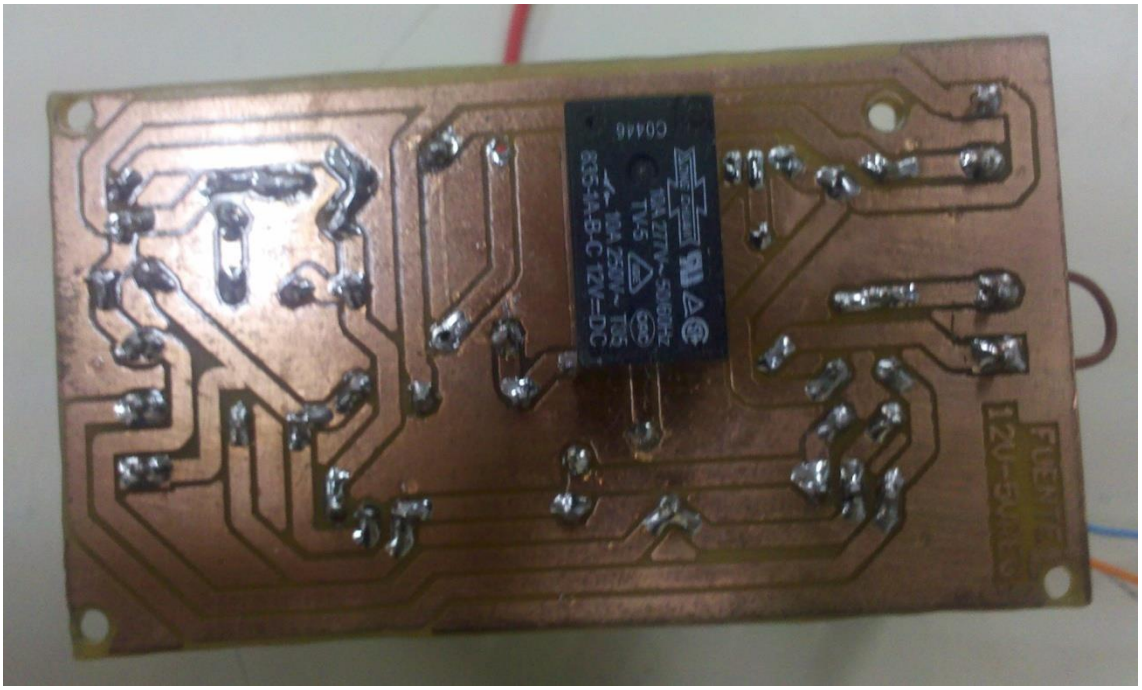
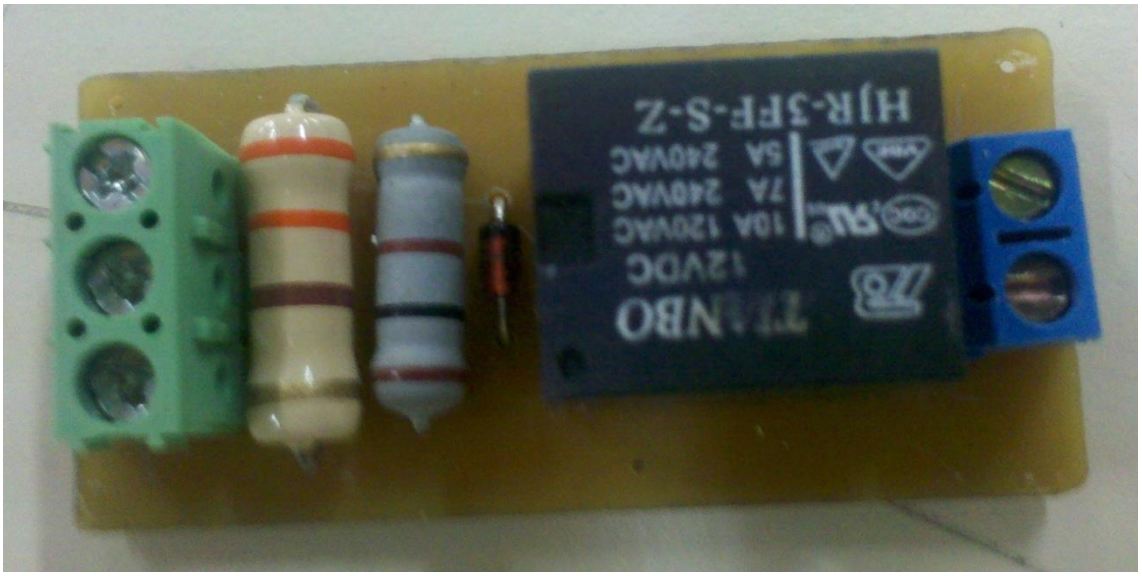
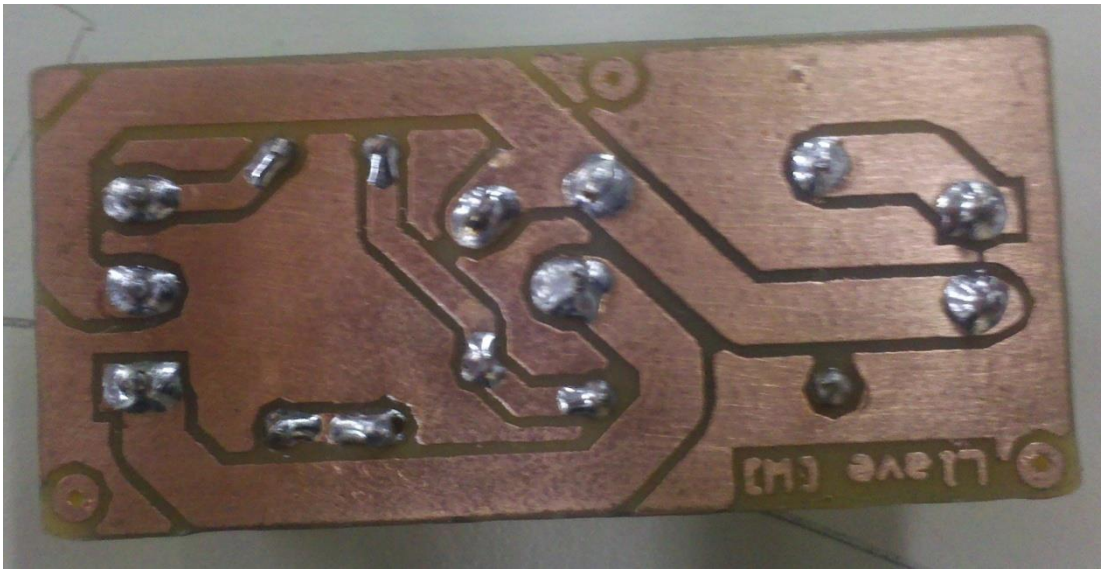


Foto real de Fuente



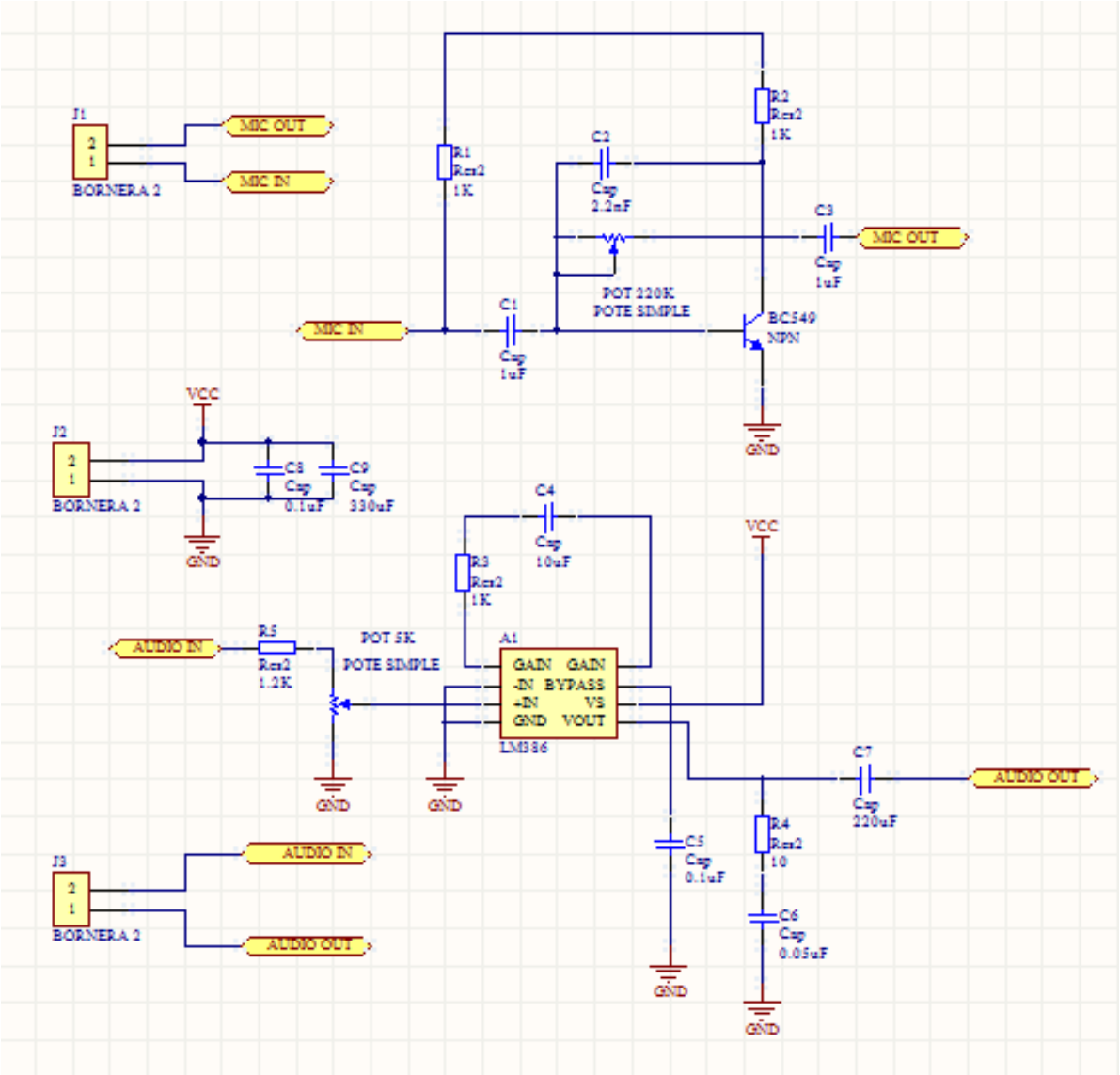

FUENTE
12V-5A


HJR-3FF-S-Z
12VDC
10A 120VAC
7A 240VAC
5A 240VAC

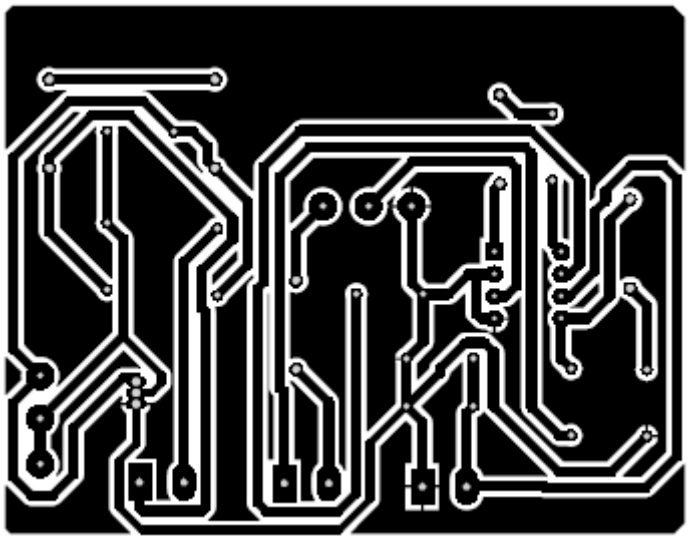


4.2 Módulo – Analógico I/O

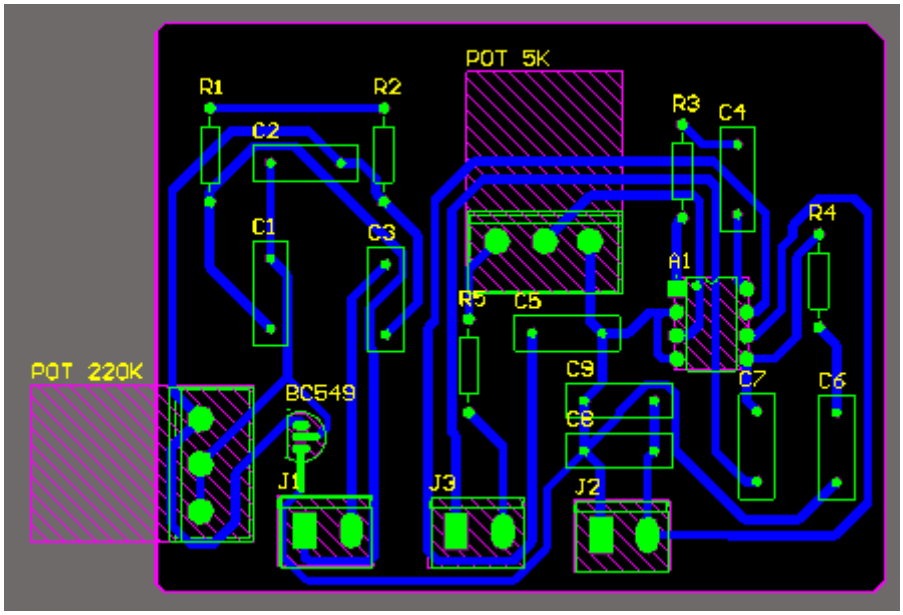
A continuación se muestra la materialización de los circuitos a implementar, los cuales fueron realizados utilizando Altium Designer Release 10.



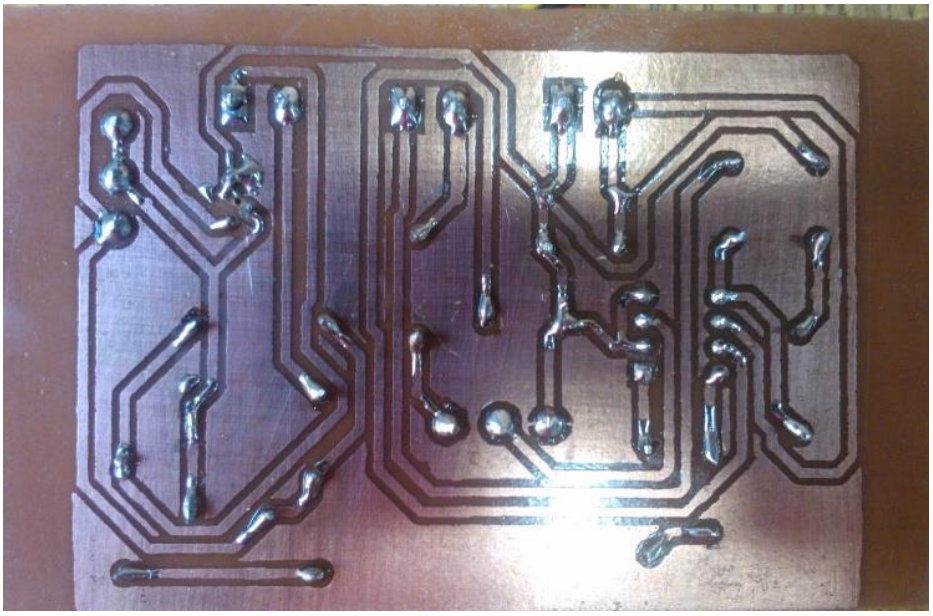
En la Figura se ve el PCB a realizar en Altium.

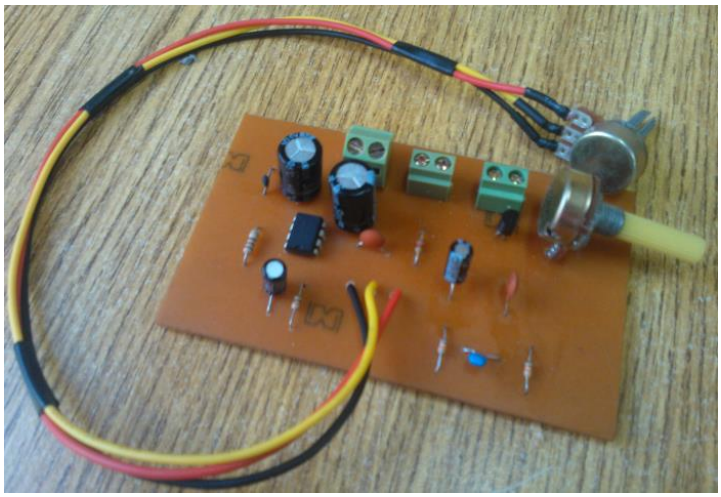


En la Figura se presenta la distribución de los componentes sobre el PCB.



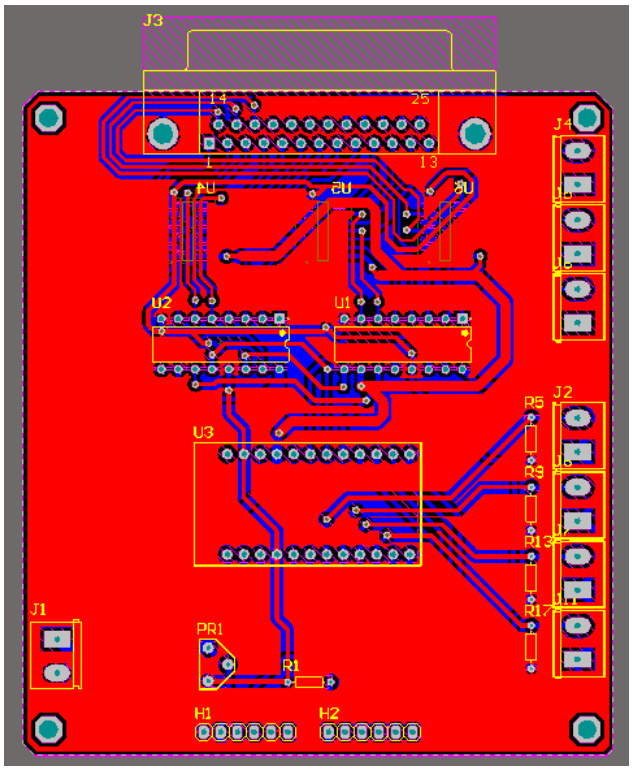
Por último en la Figura se ve la implementación final del circuito.



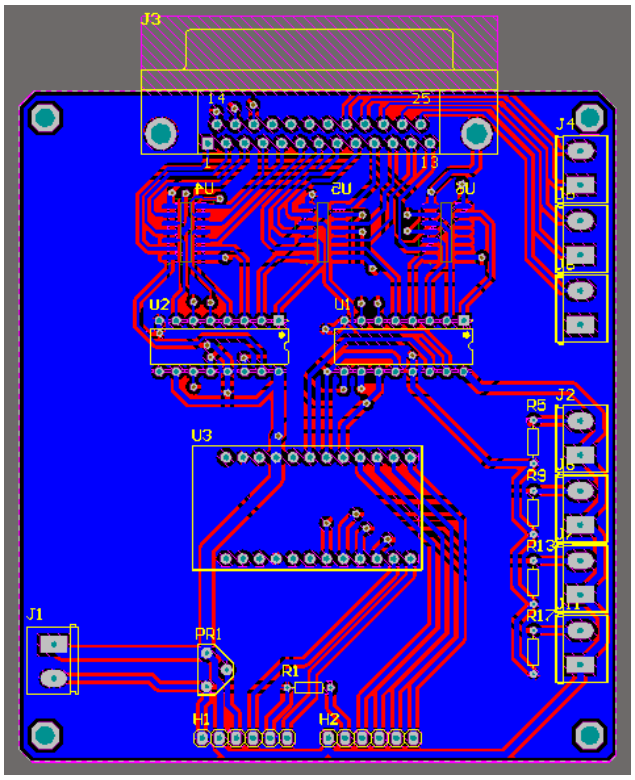


4.3 Módulo – Digital I/O

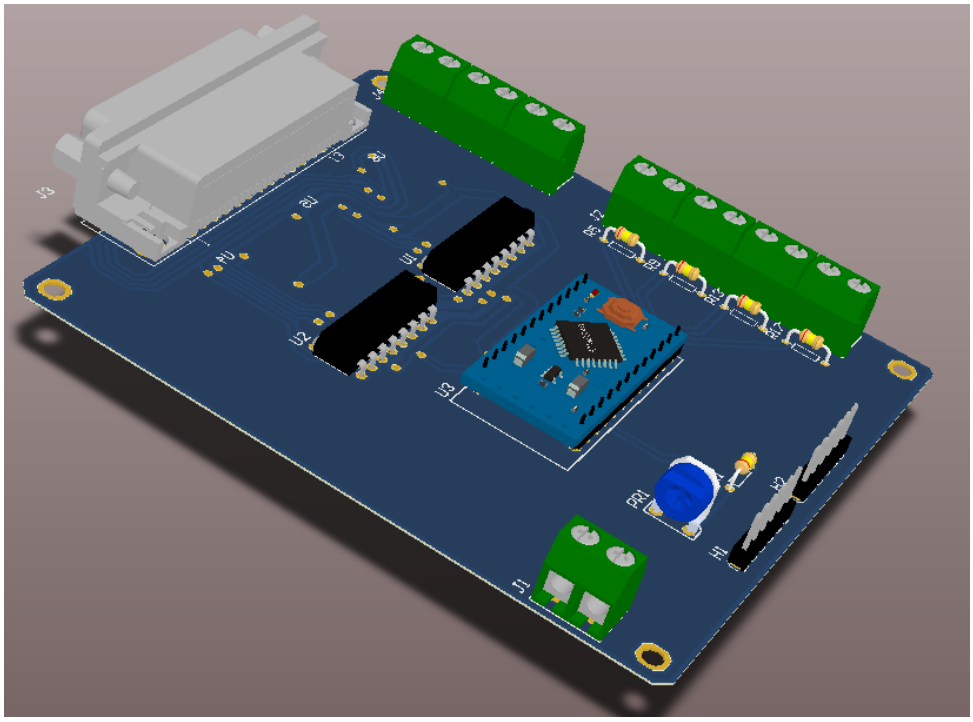
PCB capa TOP

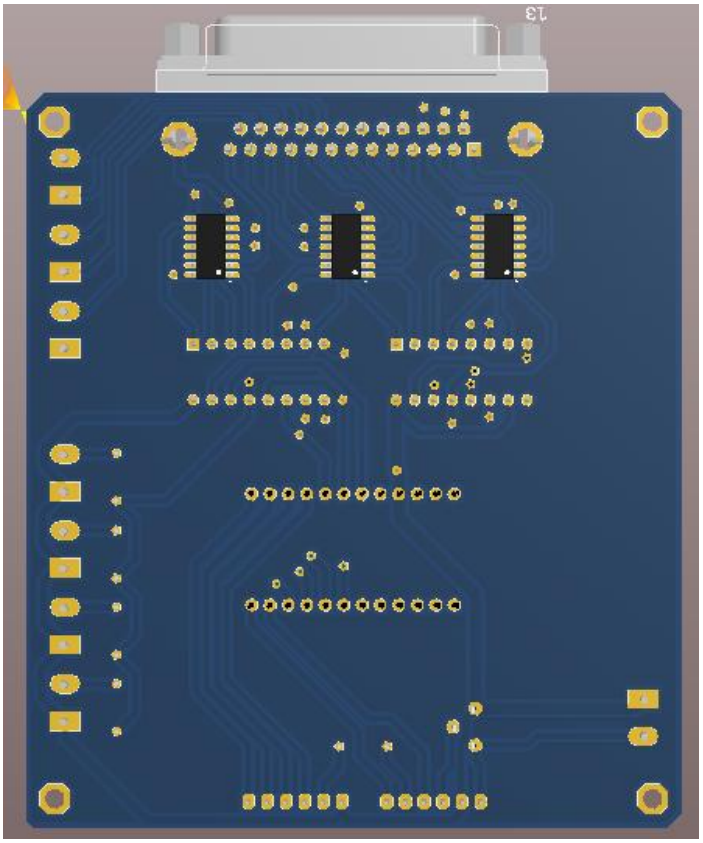


PCB capa BOTTON

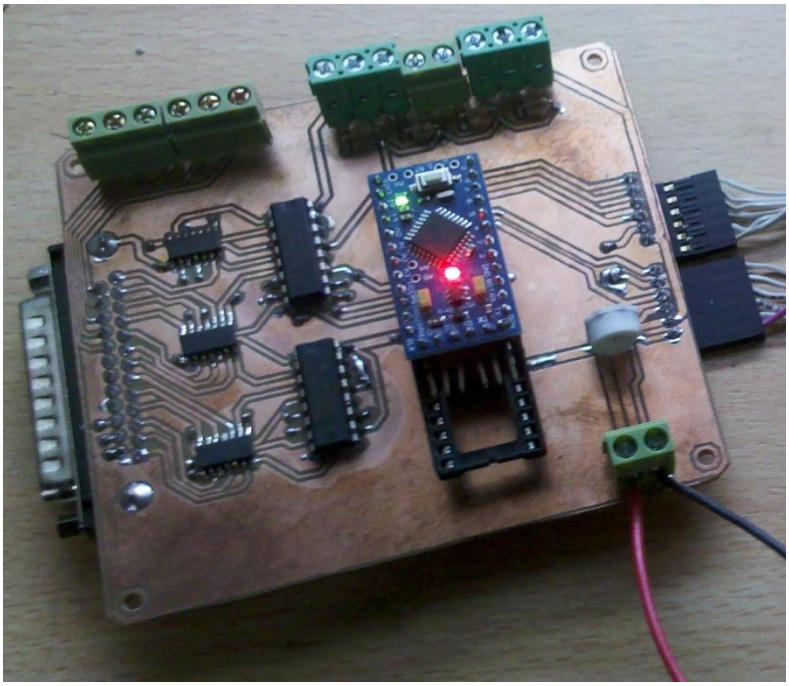


Captura de 3D con módulo Arduino Mini Pro



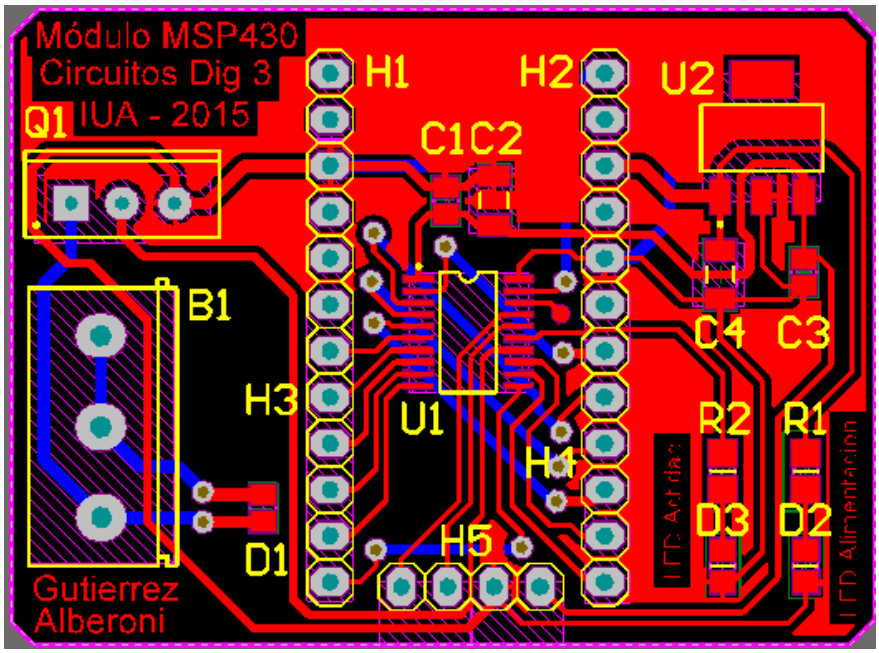


Fotos de Placa terminada con Arduino Mini Pro

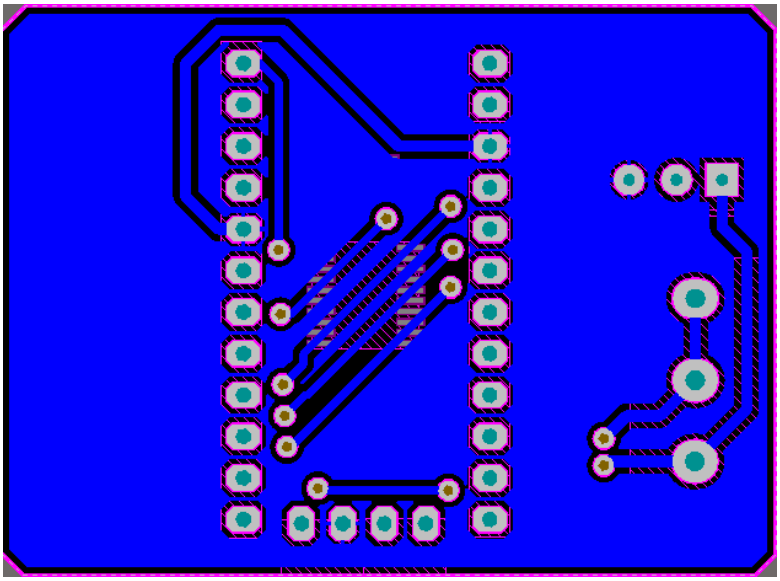


4.4 Módulo MSP430

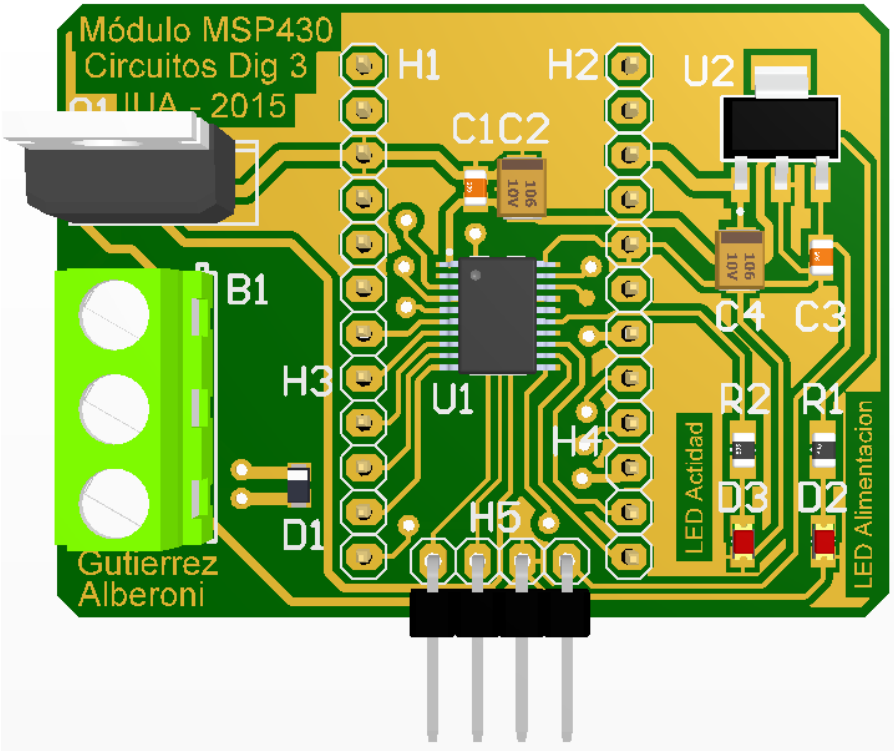
PCB TOP



PCB BOTTON



3D PCB


Módulo MSP430
Circuitos Dig 3
2015
H1
H2
U2
C1C2
B1
H3
U1
H4
C4
C3
R2
R1
D3
D2
H5
D1
LED Actidad
LED Alimentacion
Gutierrez
Alberoni

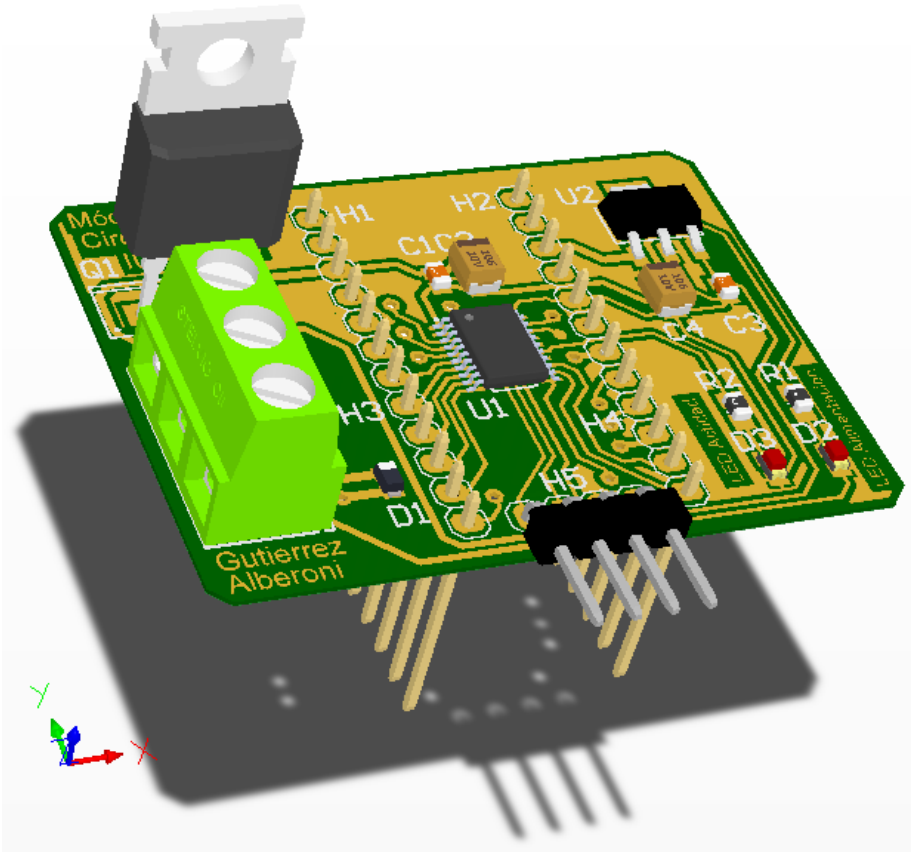
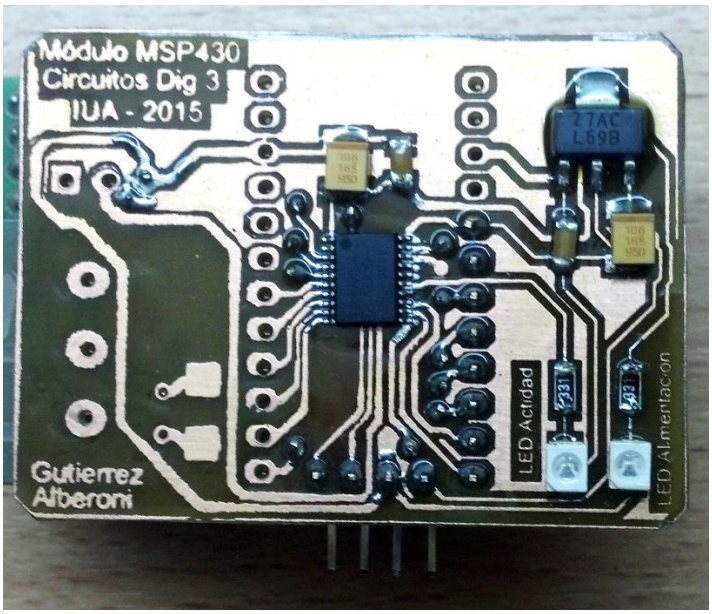
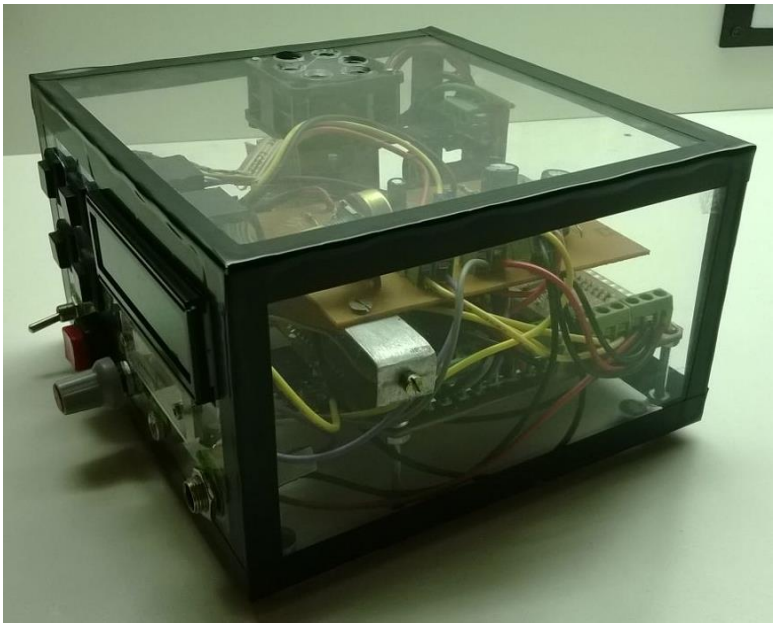


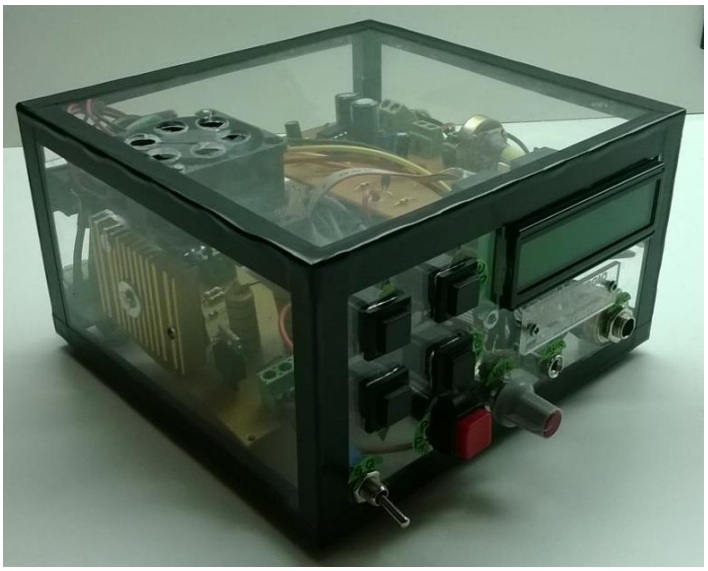
Foto Placa terminada

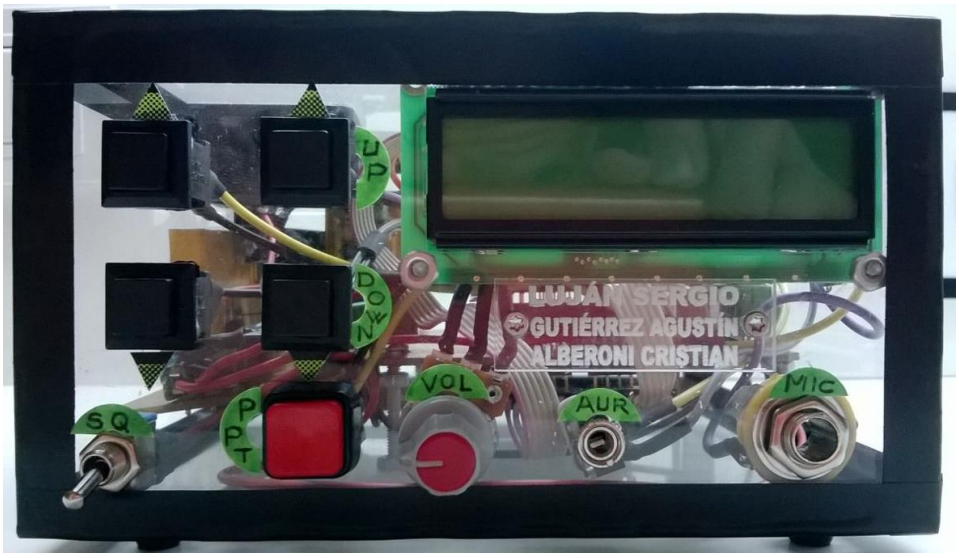


Fotos de Placa terminada con Msp430g2553

4.5 Integración de módulos y gabinete






UP
DOWN
SQ
PTT
VOL
AUR
MIC
GUTIÉRREZ AGUSTÍN
ALBERONI CRISTIAN


Fr. 117.525 MHz
UP
DOWN
SQ
PTT
VOL
AUR
MIC
GUTIÉRREZ AGUSTÍN
ALBERONI CRISTIAN

5. CONCLUSIONES

Como resultado del diseño y desarrollo se obtuvo un prototipo experimental funcional de lo solicitado que incorpora las especificaciones dadas.

Durante el proceso de diseño se dedicó la mayor cantidad de tiempo a la diagramación de los flujogramas del firmware y los esquemáticos generales de manera de poder tener un sólido punto de partida para el desarrollo del prototipo. Obteniendo de esta forma, una idea general de trabajo de un proyecto de este tipo, abarcando la mayor cantidad de etapas del diseño y utilizando la mayor cantidad de herramientas de software y hardware que se disponen.

6. REFERENCIAS

[1] "Collins VHF-20A/20B VHF Transceiver " Instruction book – 523-0765213-10511A

[2] "Datasheet conector audio" http://www.cui.com/product/resource/sj5-43502pm.pdf

[3] "MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller (Rev. J) " http://www.ti.com/lit/gpn/msp430g2553

[4] "MSP430x2xx Family User's Guide (Rev. J)" http://www.ti.com/lit/pdf/slau144

[5] "Datasheet 74HC595" http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf

[6] "SN74LVC07A Hex Buffer/Driver With Open-Drain Outputs (Rev. U) http://www.ti.com/lit/gpn/sn74lvc07a

[7] "LMOS source code" Ing. Marcelo Lorenzati http://www.sistemasembebidos.com.ar/cms/index.php?option=com_content&view=article&id=63:mini-scheduler&catid=36:programacion&Itemid=41 (13/04/2015)

[8]

7. ANEXO 1 – Firmware Msp430g2553

El firmware completo del proyecto, junto con informes y hardware se encuentran disponibles en:

https://github.com/calberoni/RemoteControl-VHF-20A.git

8. ANEXO 2 - Firmware Arduino

```
/*
 * control_VHF_20A.ino
 *
 * Copyright 2014 Cristian Alberoni <calberoni873@alumnos.iua.edu.ar>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.
 *
 *
 */


#include <LiquidCrystal.h>
//definiciones Pines para LCD
#define RS  8
#define ENABLE  9
#define DB4  10
#define DB5  11
#define DB6  12
#define DB7  13
//definiciones Pines para Botones
#define  BTN1  4
#define  BTN2  5
#define  BTN3  6
#define  BTN4  7
//definiciones Pines para 74hc595
#define  CLOCK    10 //Frecuencia 100Hz
#define  PIN_CLK  14
#define  PIN_LTCH 15
#define  PIN_DATA 16
#define  PIN_OE   12
```

```
#define  PIN_RST  11

int frecu = 116;
int decFrec = 525;

int cont =0;

int boton1_flag = 0;
int boton2_flag = 0;
int boton3_flag = 0;
int boton4_flag = 0;

int decM;
int uniM;
int cienK;
int decK;
int uniK;
int deK;

/*
Configuracion
 */

int decMegas[]={
  0x03,0x05,0x06,0x0A,0x0C,0xFF,0x00,0x0E};
int uniMegas[]={
  0x12,0x03,0x05,0x06,0x0A,0x0C,0x14,0x18,0x09,0x11,0xFF,0x00,0xFE};  //Se agrega 0xFF --
> prende todo, Se agrega 0x00 --> prende todo
int cientosKilos[]={
  0x12,0x03,0x05,0x06,0x0A,0x0C,0x14,0x18,0x09,0x11,0xFF,0x00,0xFE};  //Se agrega 0xFF --
> prende todo, Se agrega 0x00 --> prende todo
int decenKilos[]={
  0x00,0x01,0x03,0x02};

int salida[15] = {
  0};

/* int decMeg = decMegas[2];
 int uniMeg = uniMegas[5];
 int cienKil = cientosKilos[2];
 int decKil = decenKilos[1];*/
int decMeg;
int uniMeg;
int cienKil;
```

```
int decKil;

int button1State;
int button2State;
int button3State;
int button4State;

int lastButton1State = LOW;
int lastButton2State = LOW;
int lastButton3State = LOW;
int lastButton4State = LOW;


long lastDebounceTime = 0;
long debounceDelay = 100;

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd( RS, ENABLE, DB4, DB5, DB6, DB7);

void setup(){
  Serial.begin(9600);
  pinMode(BTN1, INPUT);
  pinMode(BTN2, INPUT);
  pinMode(BTN3, INPUT);
  pinMode(BTN4, INPUT);

  pinMode(PIN_LTCH, OUTPUT);
  pinMode(PIN_CLK, OUTPUT);
  pinMode(PIN_DATA, OUTPUT);
  pinMode(PIN_OE, OUTPUT);
  pinMode(PIN_RST, OUTPUT);

  lcd.begin(16, 2);
  lcd.print("Control ");
  lcd.setCursor(0,1);
  lcd.print("VHF 20A");
  // Se realiza un test de prueba de interfase salida
  test();

  delay(2000);

  lcd.setCursor(0,0);
  lcd.print("Fr. ");
  lcd.print(frecu);
```

```
  lcd.print(".");
  lcd.setCursor(0,1);
  lcd.print(decFrec);
  lcd.print(" MHz");
  port_write();
}

void loop(){
  check_botones();
  actualizar_lcd();
}


void check_botones(void){
  // read the state of the switch into a local variable:
  int bot1 = digitalRead(BTN1);
  int bot2 = digitalRead(BTN2);
  int bot3 = digitalRead(BTN3);
  int bot4 = digitalRead(BTN4);

  if (bot1 != lastButton1State) {
    lastDebounceTime = millis();
  }
  if (bot2 != lastButton2State) {
    lastDebounceTime = millis();
  }
  if (bot3 != lastButton3State) {
    lastDebounceTime = millis();
  }
  if (bot4 != lastButton4State) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (bot1 != button1State) {
      button1State = bot1;
      if (button1State == HIGH) {
        boton1_flag=1;
      }
    }
    else{
      button1State = 0;
    }
    if (bot2 != button2State) {
```

```
      button2State = bot2;
      if (button2State == HIGH) {
        boton2_flag=1;
      }
    }
    else{
      button2State = 0;
    }
    if (bot3 != button3State) {
      button3State = bot3;
      if (button3State == HIGH) {
        boton3_flag=1;
      }
    }
    else{
      button3State = 0;
    }

    if (bot4 != button4State) {
      button4State = bot4;
      if (button4State == HIGH) {
        boton4_flag=1;
      }
    }
    else {
      button4State = 0;
    }
  }
  else{
    cont=0;
  }

  lastButton1State = bot1;
  lastButton2State = bot2;
  lastButton3State = bot3;
  lastButton4State = bot4;
}


void actualizar_lcd(void){
  if(boton1_flag){  //Boton activado
    cont++;
    Serial.print(cont);
    Serial.print("\n");
```

```
  if (cont >= 5){
    frecu = frecu +5;
  }
  else {
    frecu++;
  }

  if (frecu >= 151){
    frecu = 151;
  }
  //actualizo LCD
  //Desarmo el numero
  uniM = frecu%10;
  decM = (frecu/10)%10;

  lcd.setCursor(0,0);
  lcd.print("Fr. ");
  lcd.print(frecu);
  lcd.print(".");


  decMeg = decMegas[decM - 1];
  uniMeg = uniMegas[uniM];
  cienKil = cientosKilos[cienK];
  decKil = decenKilos[decK];

  port_write();
  boton1_flag = 0;
}

if(boton2_flag){  //Boton activado
  cont++;

  if (cont >= 5){
    frecu = frecu-5;
  }
  else {
    frecu--;
  }

  if (frecu <= 116){
    frecu = 116;
  }
  //actualizo LCD
```

```
  uniM = frecu%10;
  decM = (frecu/10)%10;

  lcd.setCursor(0,0);
  lcd.print("Fr. ");
  lcd.print(frecu);
  lcd.print(".");

  decMeg = decMegas[decM - 1];
  uniMeg = uniMegas[uniM];
  cienKil = cientosKilos[cienK];
  decKil = decenKilos[decK];

  port_write();
  boton2_flag = 0;
 }
 if(boton3_flag){  //Boton activado
  cont++;

  if (cont >= 5){
   decFrec = decFrec +100;
  }
  else {
   decFrec = decFrec + 25;
  }

  if(decFrec >= 1000 ){
   decFrec = 0;
  }
  cienK = (decFrec/100)%10;

  uniK = decFrec%10;
  deK = (decFrec/10)%10;

  if(uniK==0 && deK==0){
   decK = 0;
  }
  if(uniK==5 && deK==2){
   decK = 1;
  }
  if(uniK==0 && deK==5){
   decK = 2;
  }
  if(uniK==5 && deK==7){
```

```
    decK = 3;
  }

  decMeg = decMegas[decM - 1];
  uniMeg = uniMegas[uniM];
  cienKil = cientosKilos[cienK];
  decKil = decenKilos[decK];

  lcd.setCursor(0,1);
  lcd.print(decFrec);
  lcd.print(" MHz  ");
  port_write();
  boton3_flag = 0;
 }
 if(boton4_flag){  //Boton activado
  cont++;

  if (cont >= 5){
    decFrec = decFrec - 100;
  }
  else {
    decFrec = decFrec - 25;
  }

  if(decFrec < 0 ){
    decFrec = 975;
  }
  cienK = (decFrec/100)%10;

  uniK = decFrec%10;
  deK = (decFrec/10)%10;

  if(uniK==0 && deK==0){
    decK = 0;
  }
  if(uniK==5 && deK==2){
    decK = 1;
  }
  if(uniK==0 && deK==5){
    decK = 2;
  }
  if(uniK==5 && deK==7){
    decK = 3;
  }
```

```
    decMeg = decMegas[decM - 1];
    uniMeg = uniMegas[uniM];
    cienKil = cientosKilos[cienK];
    decKil = decenKilos[decK];
    lcd.setCursor(0,1);
    lcd.print(decFrec);
    lcd.print(" MHz  ");
    port_write();
    boton4_flag = 0;
   }
}
void port_write(void){

  digitalWrite(PIN_OE,HIGH);
  digitalWrite(PIN_LTCH, LOW);
  digitalWrite(PIN_CLK, LOW);
  digitalWrite(PIN_RST,LOW);
  delay(100);

  digitalWrite(PIN_RST,HIGH);
  digitalWrite(PIN_OE,LOW);

  //Armo la trama
  for(int i=0;i<=15;i++){
    if(i==0){
      salida[7] = decMeg & 0x01;
      decMeg = decMeg >>1;
    }
    if(i>0 && i<=3) {
      salida[i-1] = decMeg & 0x01;
      decMeg = decMeg >>1;
    }
    if(i>3 && i<=7) {
      salida[i-1] = uniMeg & 0x01;
      uniMeg = uniMeg >>1;
    }
    if(i==8){
      salida[15] = uniMeg & 0x01;
      uniMeg = uniMeg >>1;
    }
    if(i>8 && i<=13){
      salida[i-1] = cienKil & 0x01;
      cienKil = cienKil >>1;
```

```
    }
    if(i>13 && i<=15){
      salida[i-1] = decKil & 0x01;
      decKil = decKil >>1;
    }
  }

  //Mando la trama

  for(int j=0; j<=15; j++){
    digitalWrite(PIN_DATA,salida[j]);
    digitalWrite(PIN_CLK,HIGH);
    delay(CLOCK);
    digitalWrite(PIN_CLK,LOW);
    delay(CLOCK);
  }
  digitalWrite(PIN_LTCH,HIGH);
  delay(CLOCK);
  digitalWrite(PIN_CLK,LOW);
}

void test(void){

  // Todo prendido
  decMeg = decMegas[6];
  uniMeg = uniMegas[11];
  cienKil = cientosKilos[11];
  decKil = decenKilos[0];
  port_write();
  delay(3000);
  //Todo apagado

  decMeg = decMegas[5];
  uniMeg = uniMegas[11];
  cienKil = cientosKilos[11];
  decKil = decenKilos[2];
  port_write();
  delay(3000);

  decMeg = decMegas[5];
  uniMeg = uniMegas[10];
  cienKil = cientosKilos[11];
  decKil = decenKilos[2];
  port_write();
```

```
  delay(3000);

  decMeg = decMegas[5];
  uniMeg = uniMegas[10];
  cienKil = cientosKilos[10];
  decKil = decenKilos[2];
  port_write();
  delay(3000);

  decMeg = decMegas[5];
  uniMeg = uniMegas[10];
  cienKil = cientosKilos[10];
  decKil = decenKilos[0];
  port_write();
  delay(3000);

  decMeg = decMegas[7];
  uniMeg = uniMegas[12];
  cienKil = cientosKilos[12];
  decKil = decenKilos[3];
  port_write();
  delay(5000);
}

void init_salidas(void){
  uniM = frecu%10;
  decM = (frecu/10)%10;
  decMeg = decMegas[decM - 1];
  uniMeg = uniMegas[uniM];

  cienK = (decFrec/100)%10;

  uniK = decFrec%10;
  deK = (decFrec/10)%10;

  if(uniK==0 && deK==0){
    decK = 0;
  }
  if(uniK==5 && deK==2){
    decK = 1;
  }
  if(uniK==0 && deK==5){
    decK = 2;
  }
```

```
  if(uniK==5 && deK==7){
    decK = 3;
  }

  cienKil = cientosKilos[cienK];
  decKil = decenKilos[decK];
}
```