

**Profesor:**

Ing. Sebastián Serrudo

**Autor:**

Gutierrez, Agustin

Lujan, Sergio

Alberoni, Cristian

**RADIONAVEGACION 2**

**TRABAJO PRÁCTICO:**

**Control VHF 20A**

**INSTITUTO UNIVERSITARIO AERONAUTICO**

*Córdoba, 17 de octubre de 2014*

**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 DISEÑO 8

3.2.1 Módulo – Fuente de alimentación 8

3.2.2 Módulo – Analógico I/O 8

3.2.3 Módulo – Digital I/O 10

3.2.4 Diseño de gabinete 13

4. RESULTADOS 16

4.1 Módulo – Fuente de alimentación 16

4.2 Módulo – Analógico I/O 19

4.3 Módulo – Digital I/O 22

4.4 Integración de módulos y gabinete 27

5. CONCLUSIONES 28

6. REFERENCIAS 29

7. ANEXO 1 - Firmware 30

1. OBJETIVOS

El objetivo del presente trabajo práctico es diseñar y construir un control remoto para el equipo de comunicaciones Rockwell Collins VHF 20ª (Ilustración ).



Ilustración 1

1. 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 si , si no la forma de control del mismo.

* 1. Características generales del VHF 20A

.Este equipo presenta las siguientes características

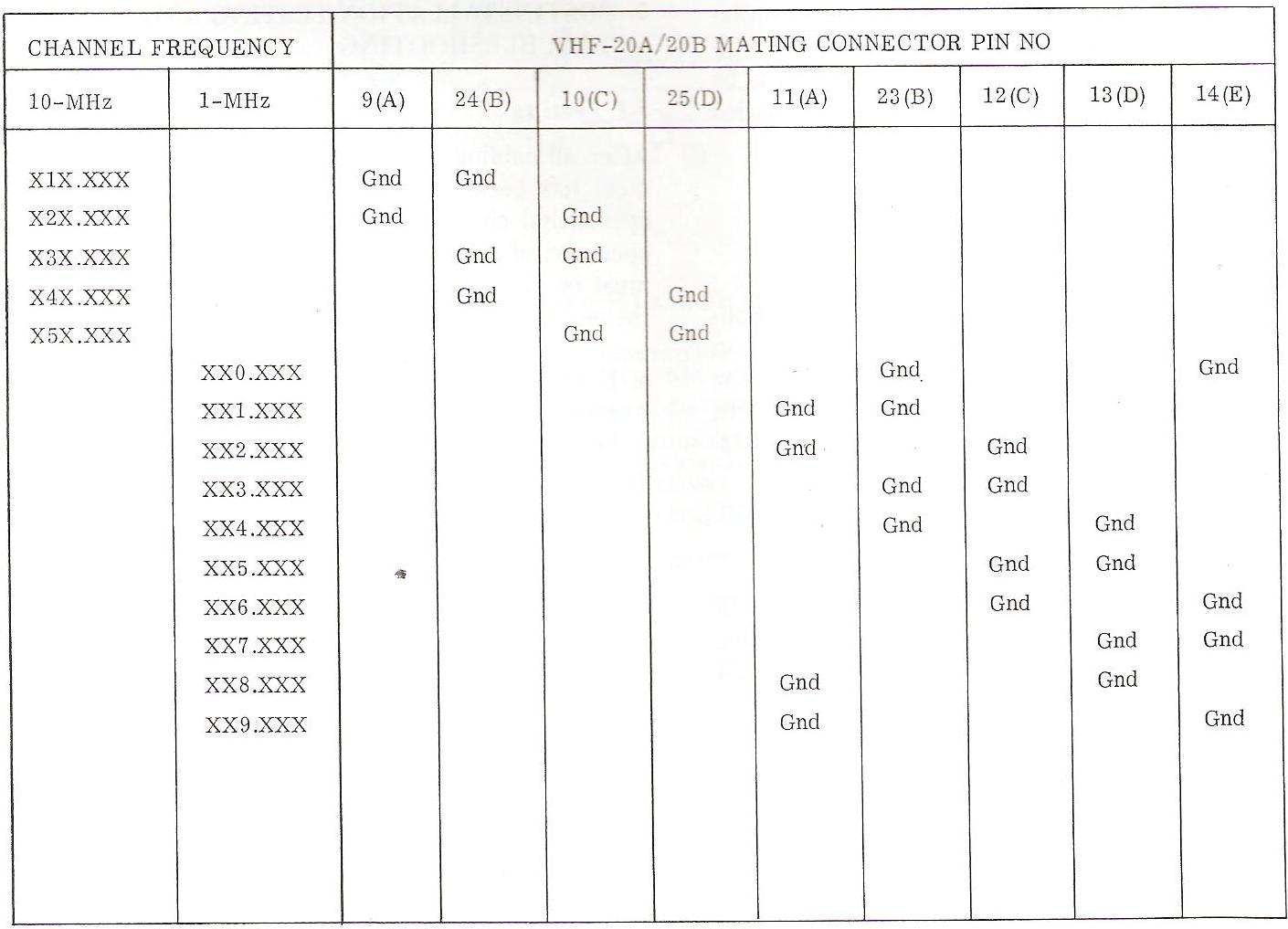
Tabla 1

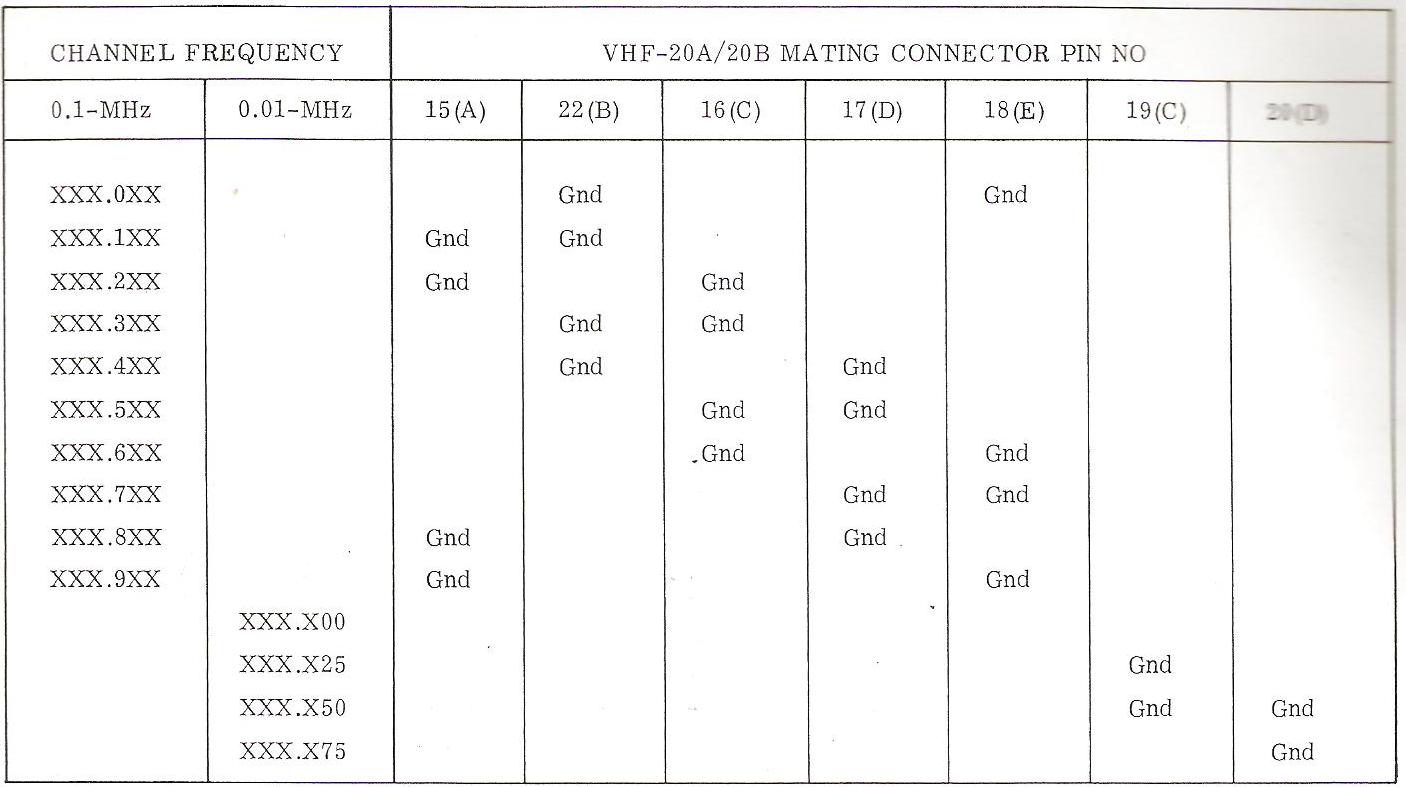
|  |  |  |  |
| --- | --- | --- | --- |
| **Weight:** | **5.6 lbs.** | **Dimensions:** | **3.750"W x 3.50"H x 13.9"L** |
| **Related documents:** | DO-138, Env Category AD/A/JNG/AAAEXXXXX; TSO C37b and C38b, class 1, FCC rules and regulations, parts 15 and 87 | **Temperature**: | -65 to +131 C |
| **Altitude:** | 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.

* 1. 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.





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

1. DESARROLLO
   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
  1. DISEÑO

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

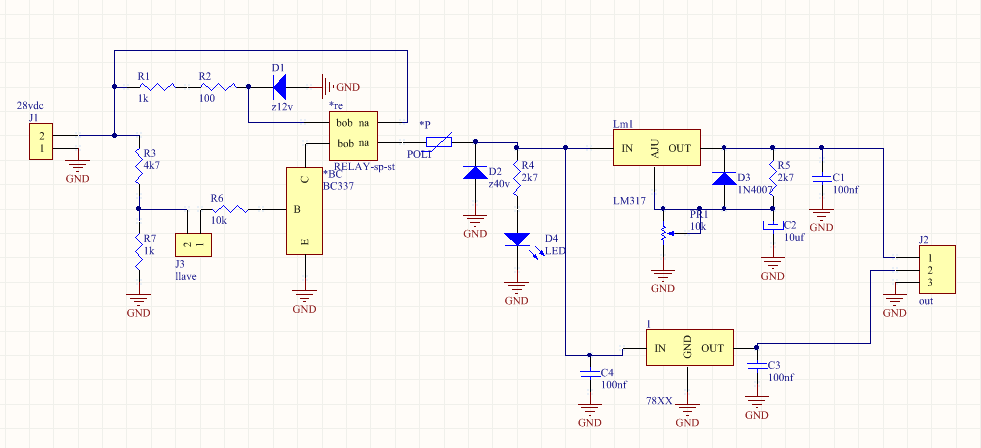
Fuente de Alimentación

Módulo Digital

Módulo analógico

* + 1. Módulo – Fuente de alimentación

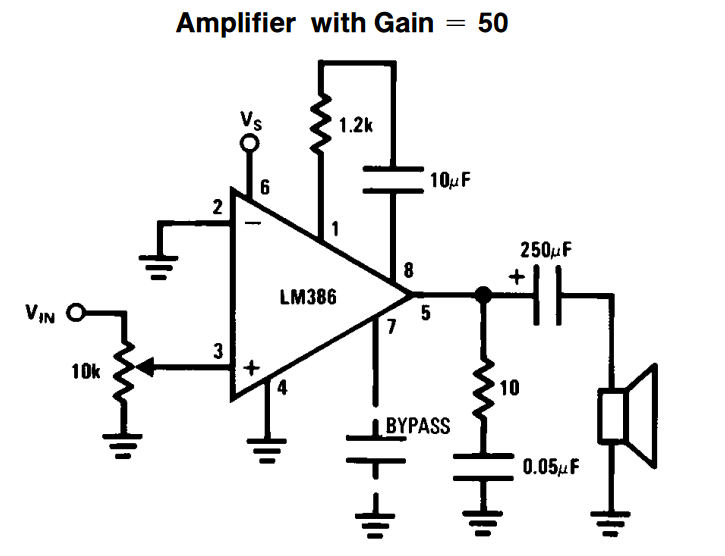
.



Circuito de fuente de alimentación

* + 1. Módulo – Analógico I/O

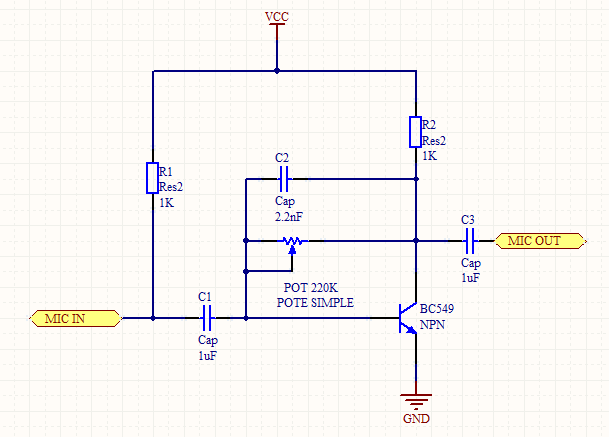
Se amplificó las señales procedentes del VHF agregándoles un control de volumen, además se realizó una adaptación de impedancia para las señales de audio colocando alta impedancia en la entrada y a la salida del amplificador se presentan los 8Ω que se dirigirán a los audífonos.



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.

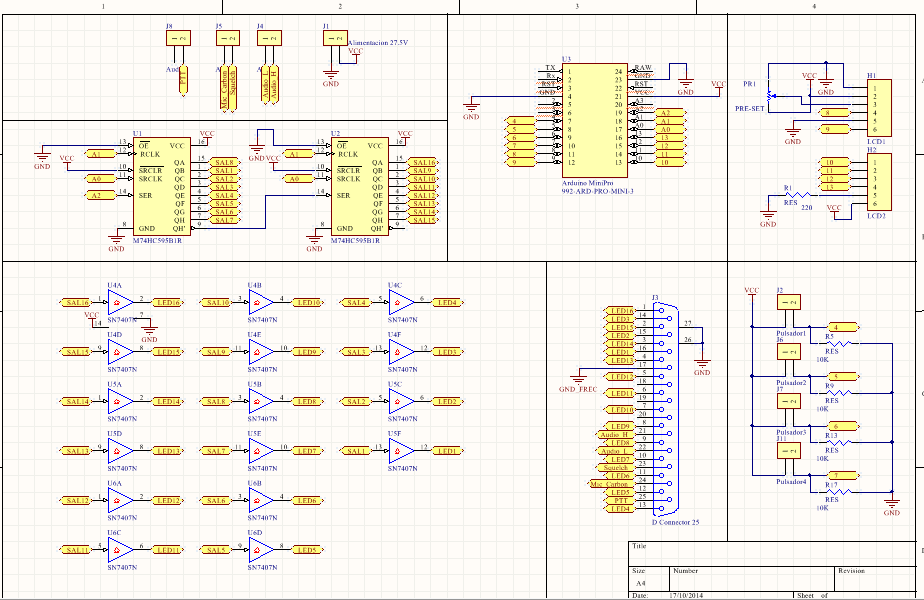
Por otra parte se debía adaptar la entrada del VHF para el micrófono la cual estaba diseñada para que se utilicen micrófonos de carbón y no los de uso actual (Electret), para ello se utilizó el siguiente circuito.



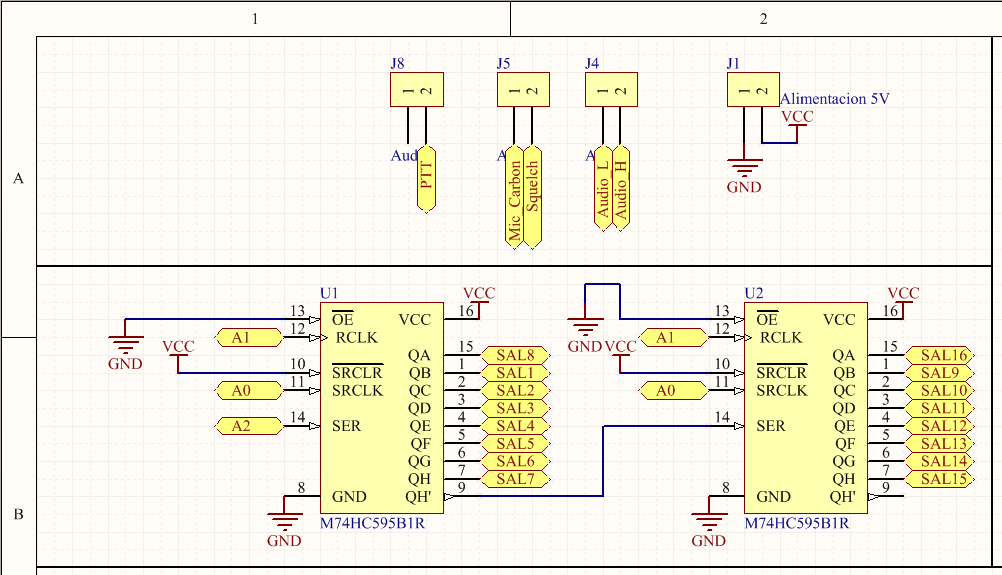
Circuito adaptador de micrófono

* + 1. Módulo – Digital I/O

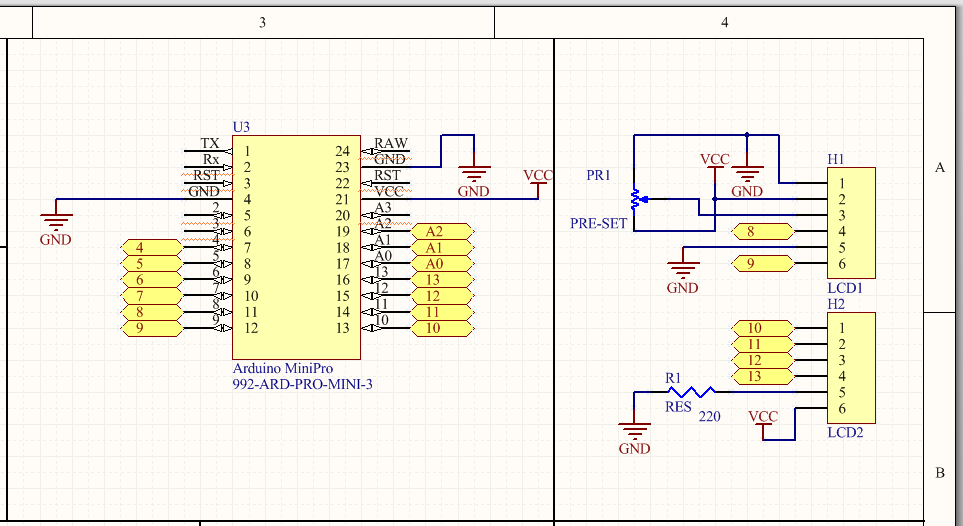
A continuación se muestra un circuito esquemático del módulo digital. Este circuito utiliza un microcontrolador ATMEGA328 incluido en la placa Arduino Mini Pro que tiene el firmware necesario (Ver ANEXO) para selección de frecuencia y control de las salidas para la codificación “2 de 5” según ARINC 410.



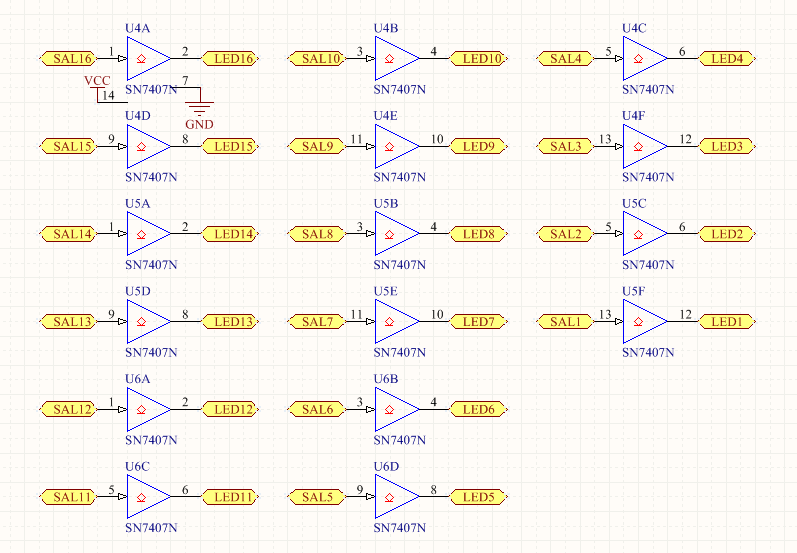
Diseño general



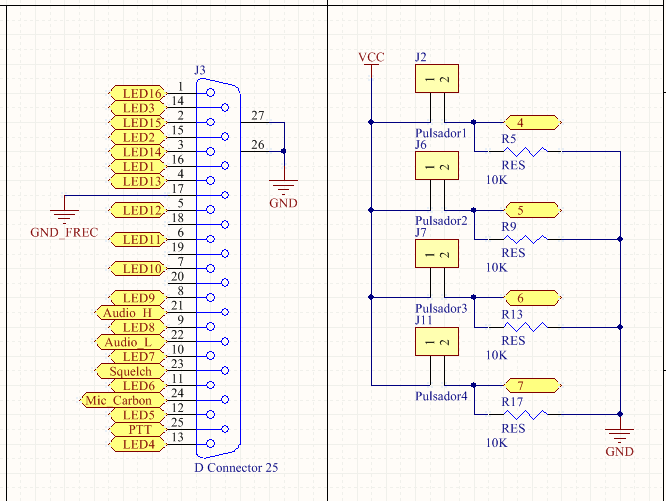
Circuito con 74HC595 para salidas



Circuito de Arduino Mini Pro y salidas para LCD



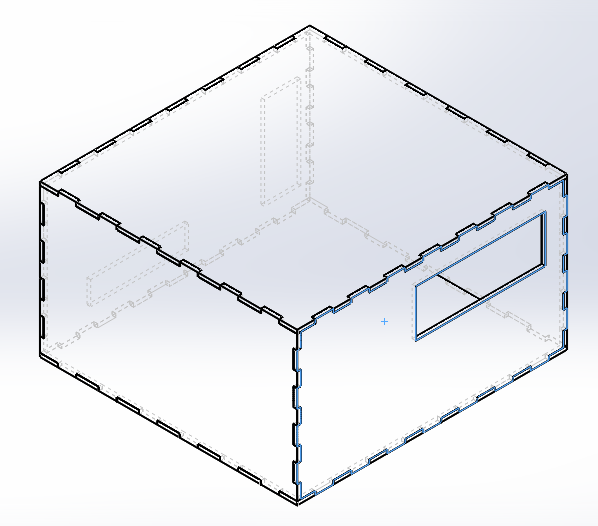
Circuito de Buffer Open Colector 74VLC07

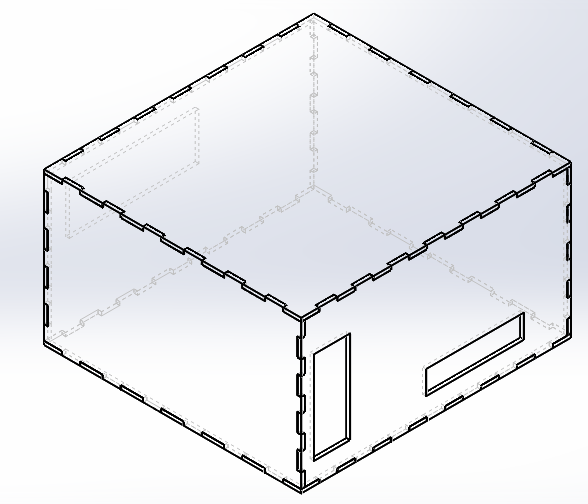


Circuito de Conector de salida DB25 y circuito de pulsadores de entrada

* + 1. Diseño de gabinete

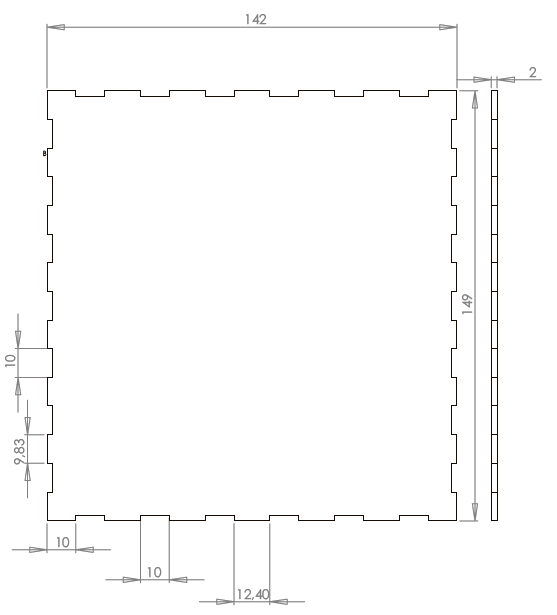
Fotos de Gabinete en SolidWorks de vista frontal y posterior



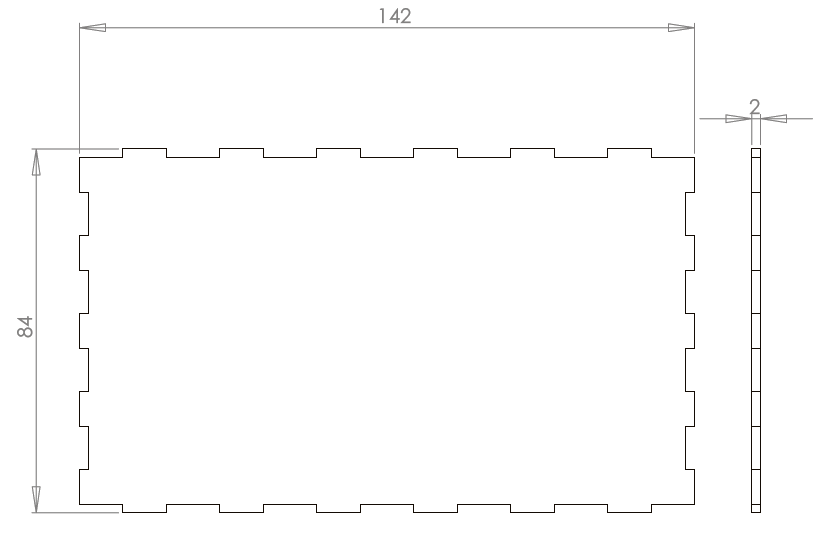


Planos constructivos del gabinete

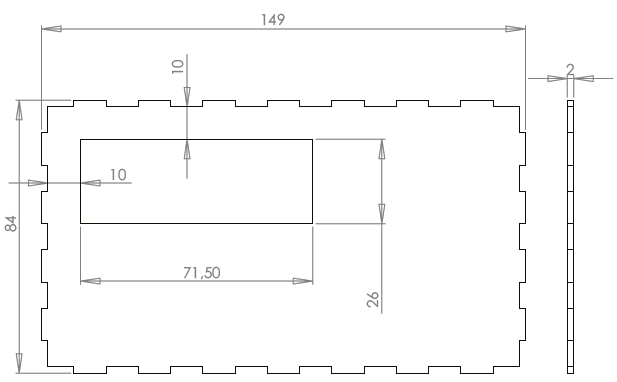
Tapa base y tapa techo



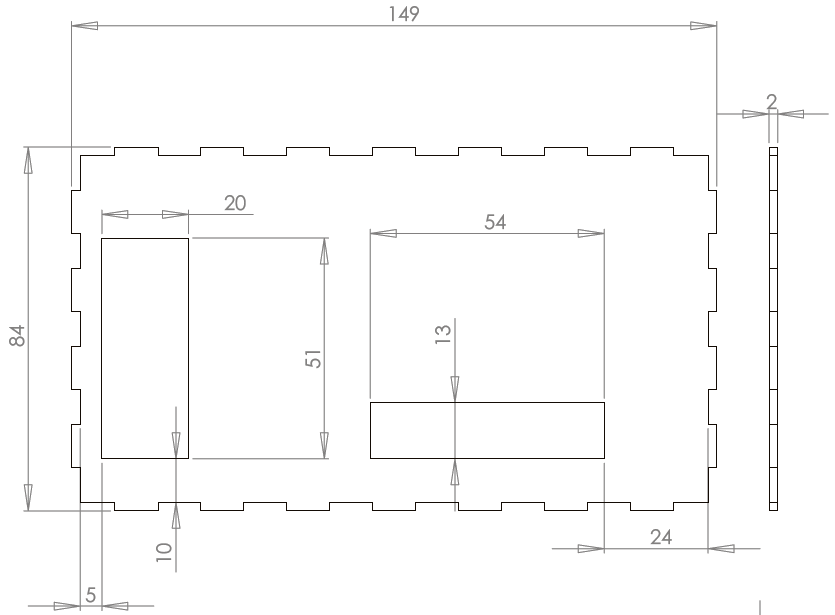
Tapa Laterales X2



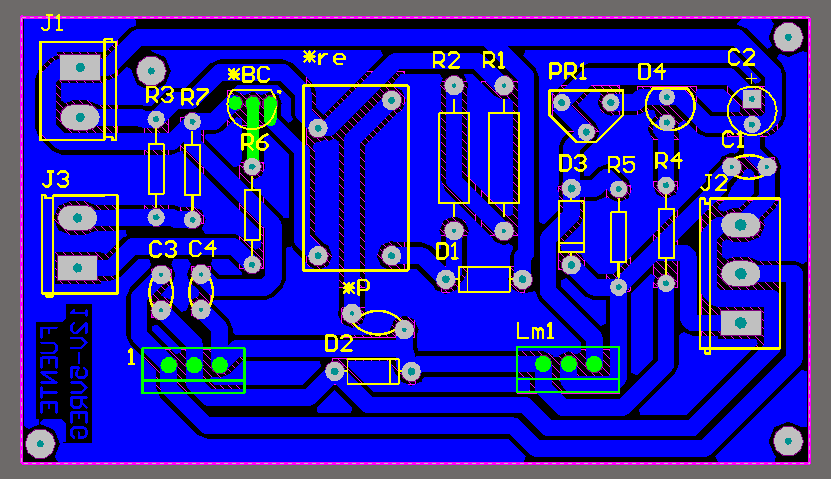
Tapa de Frente

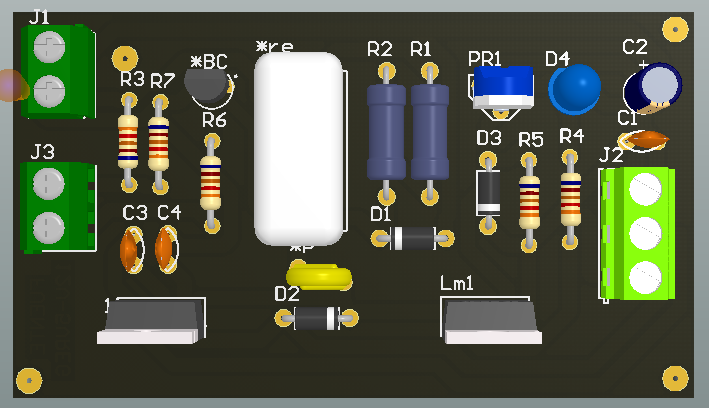


Tapa Trasera



1. RESULTADOS
   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.



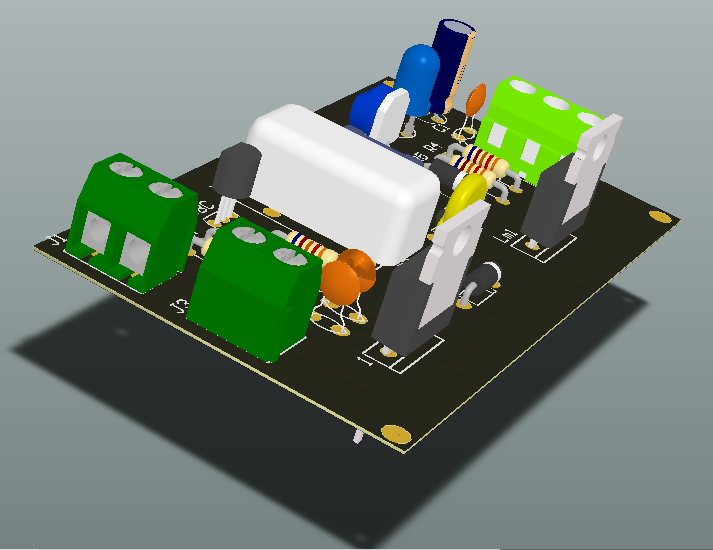
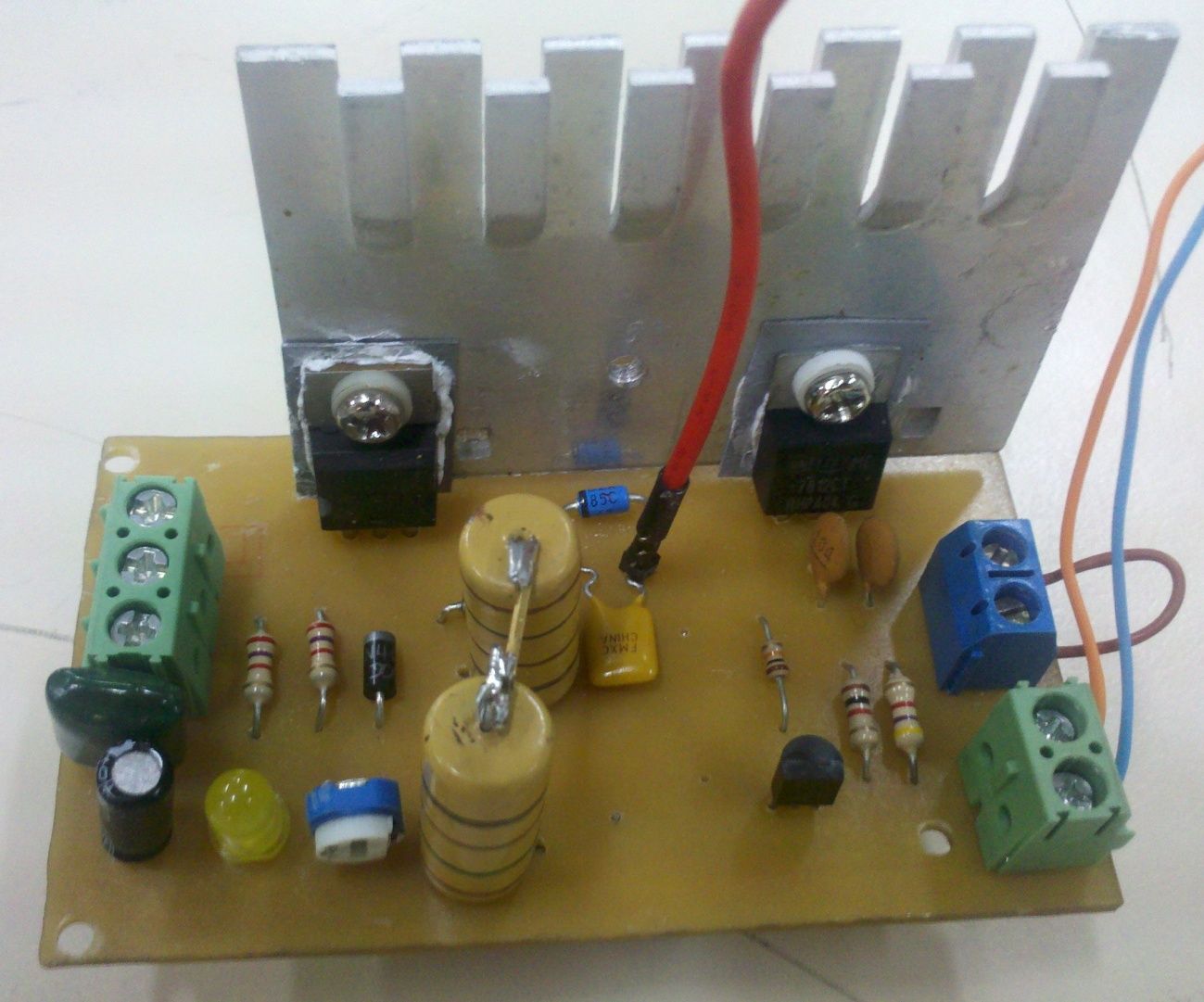
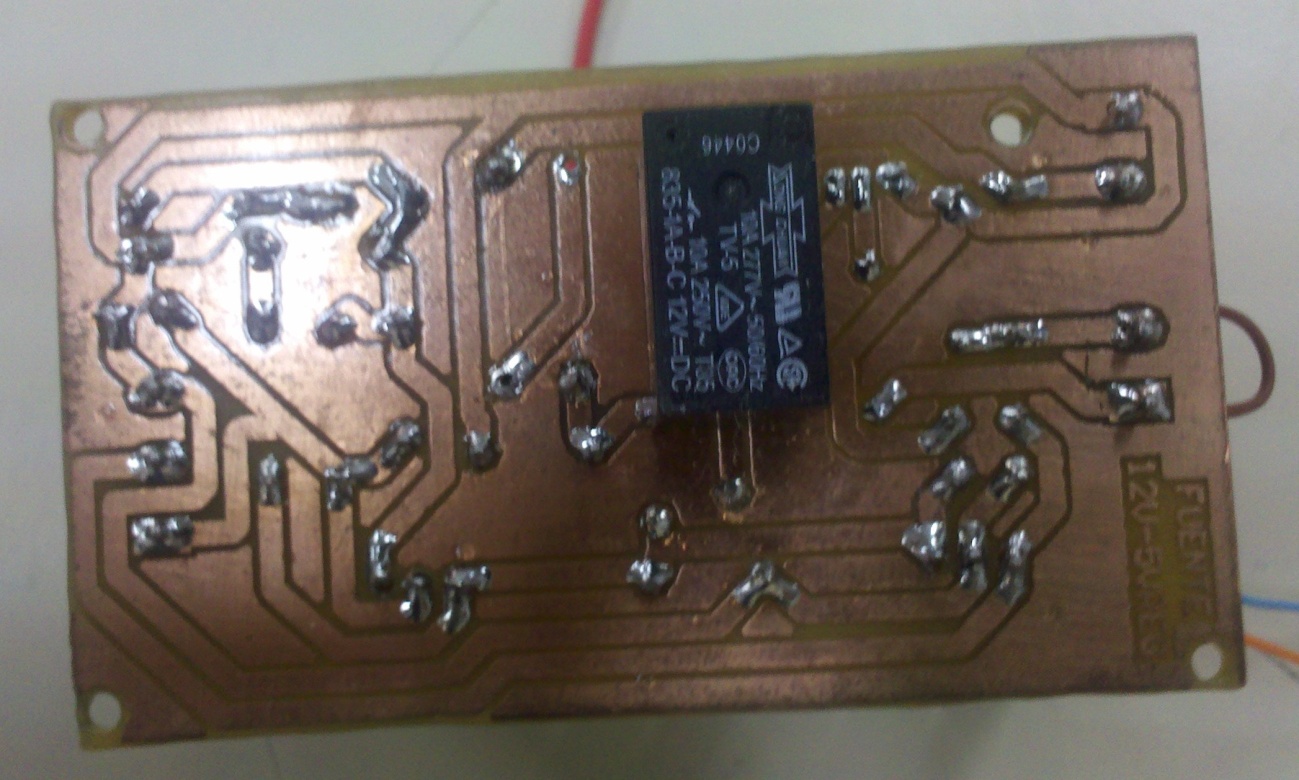
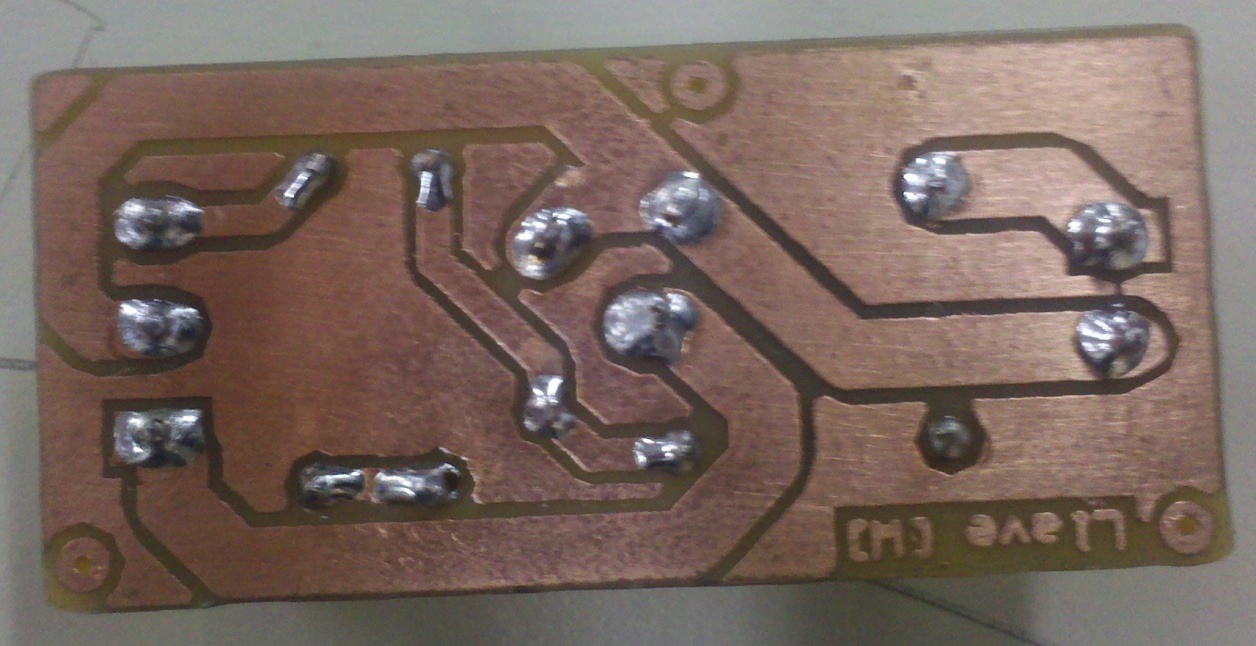


Foto real de Fuente



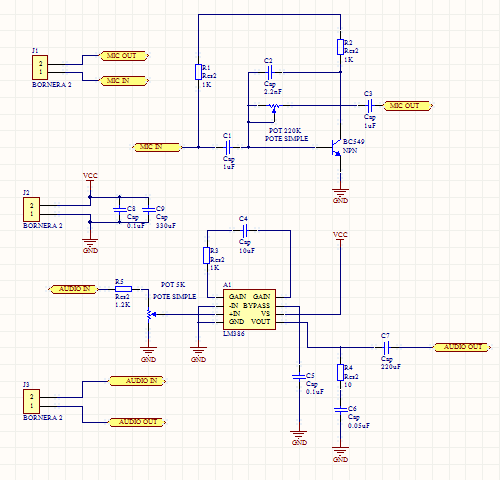




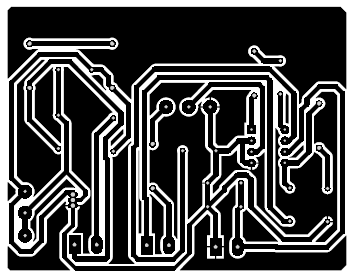


* 1. Módulo – Analógico I/O

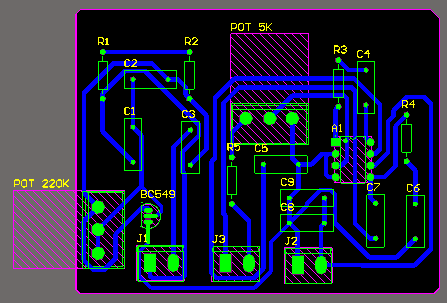
A continuación se muestra la materialización de los circuitos a implementar, los cuales fueron realizados utilizando Altium Designer Release 10.



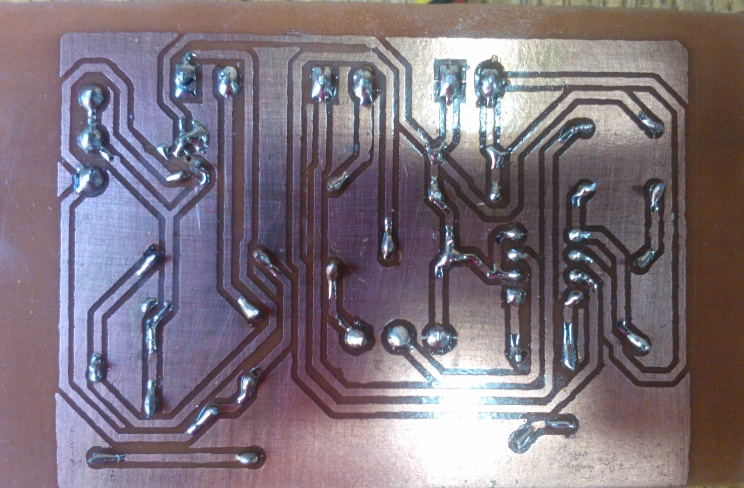
En la Figura se ve el PCB a realizar en Altium.

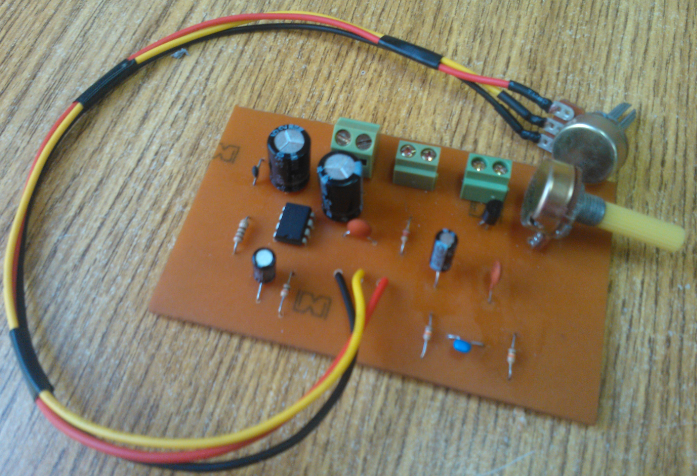


En la Figura se presenta la distribución de los componentes sobre el PCB.



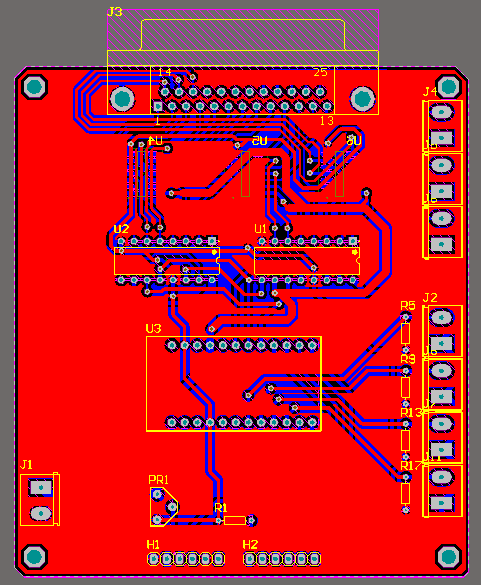
Por último en la Figura se ve la implementación final del circuito.



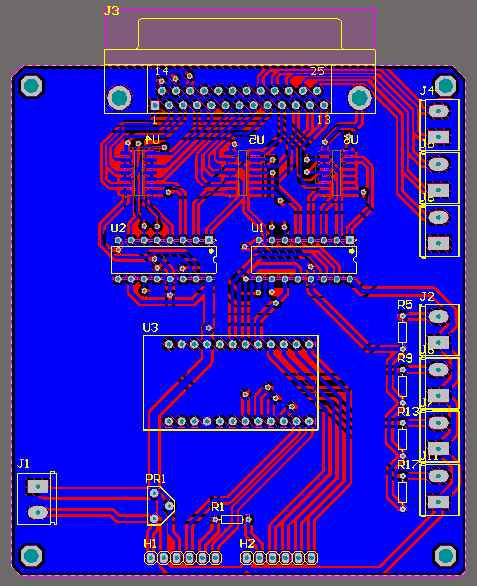


* 1. Módulo – Digital I/O

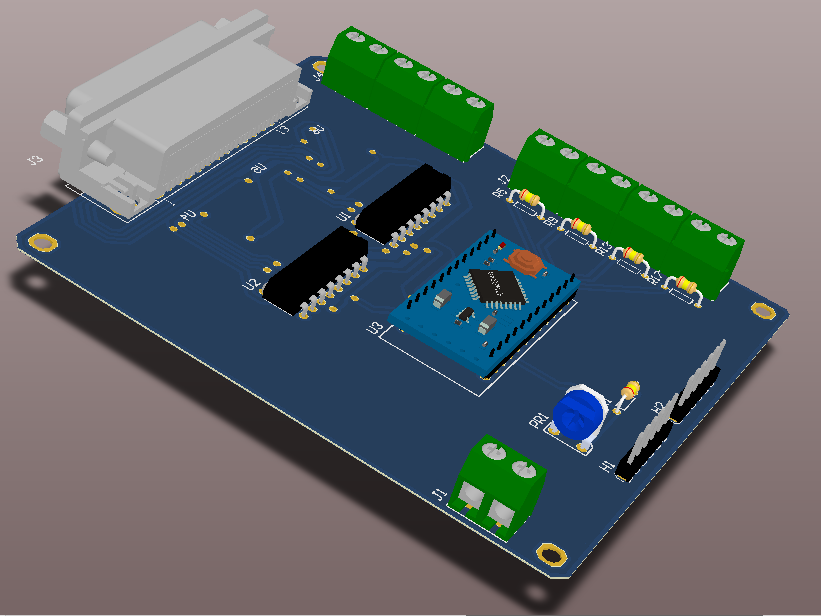
PCB capa TOP

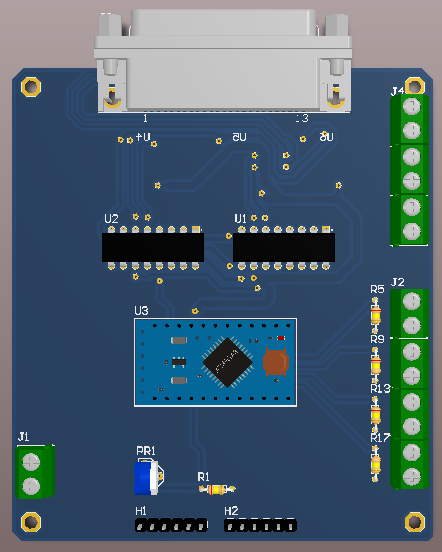


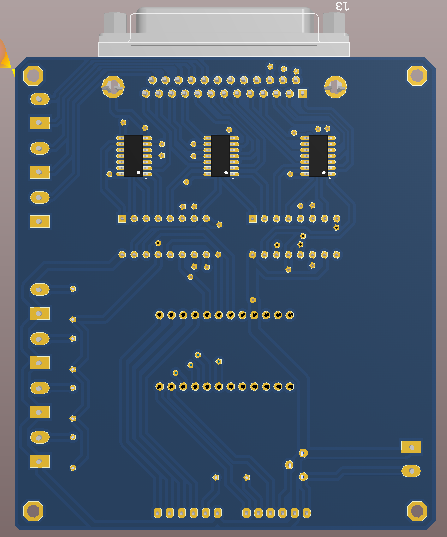
PCB capa BOTTON



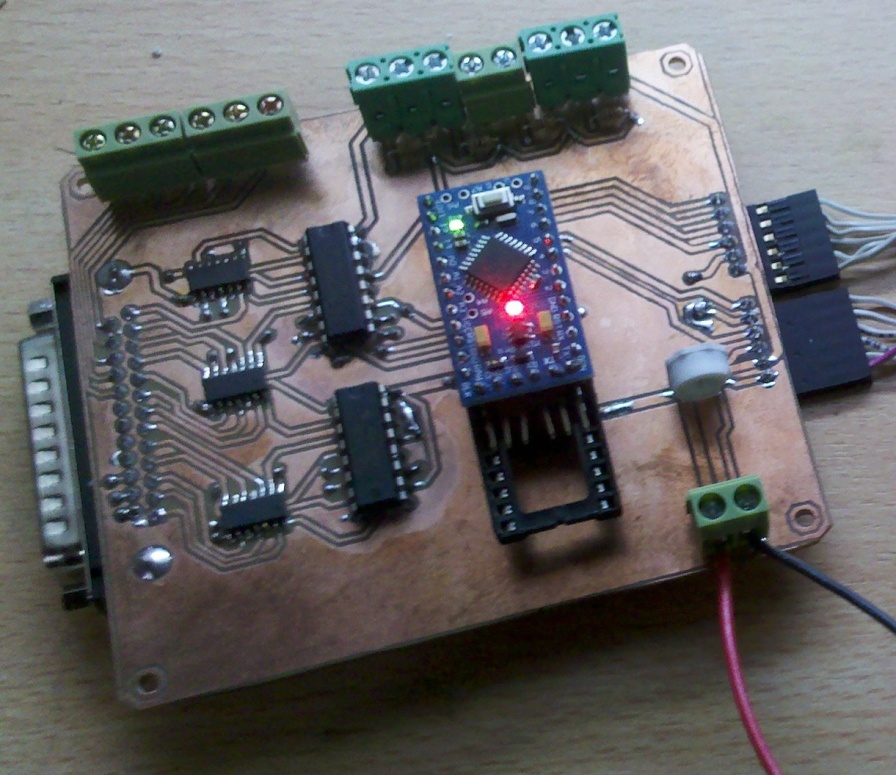
Captura de 3D







Fotos de Placa terminada



* 1. Integración de módulos y gabinete

1. CONCLUSIONES

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

El prototipo logró ser implementado gracias a los conocimientos previos de los alumnos, por lo que puede presentar deficiencias en su diseño e implementación, pero que al momento de ser probado funcionó correctamente.

1. REFERENCIAS
2. “Collins VHF-20A/20B VHF Transceiver ” Instruction book – 523-0765213-10511A
3. Datasheet conector audio <http://www.cui.com/product/resource/sj5-43502pm.pdf>
4. ANEXO 1 - Firmware

/\*

\* 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];

}