

Proyecto timbre

Informe técnico

Un trabajo presentado para la materia de
Montaje de Proyectos Electrónicos



Krapp Ramiro – Golmar Elias – Pisacane Juan Cruz

Instituto tecnológico San Bonifacio

Departamento de electrónica
26 de octubre de 2021

Hecho en L^AT_EX
Versión 1.0

Índice

1. Propósito del proyecto	3
1.1. Problematica	3
1.2. Solucion que encontramos	3
2. Diagrama en bloques	4
3. Componentes del proyecto	5
3.1. Arduino	5
3.1.1. ¿Qué es arduino?	5
3.1.2. ¿Cómo es usado en este proyecto?	5
3.2. Protocolo I ² C	5
3.2.1. ¿Qué es el el protocolo I ² C?	5
3.3. Modulo RTC DS3231	6
3.3.1. ¿Qué es el RTC DS3231?	6
3.3.2. ¿Cómo es usado en este proyecto?	6
3.4. Keypad 4x4	6
3.4.1. ¿Qué es el keypad 4x4?	6
3.4.2. ¿Cómo es usado en este proyecto?	6
3.4.3. ¿Cuál es su equivalencia eléctrica?	7
3.4.4. ¿Cuál es su pinout?	7
3.5. Display LCD I ² C 20x4	8
3.5.1. ¿Qué es el display LCD I ² C?	8
3.5.2. ¿Cómo es usado en este proyecto?	8
4. Diagrama esquematico	9
5. PCB	10
6. Diagrama de flujo	11
7. Código de programa	14
8. Explicación del código	21
8.1. Setup_deficiones.ino	21
8.2. menus.ino	22
8.3. acciones.ino	23
9. Bitácora	24
9.1. Krapp	24
9.1.1. Las 3 cosas principales que aprendí son las siguientes:	25
10. Anexos	26
11. Bibliografía y consultas	27

El índice tiene hipervínculos incorporados!

Toca en cada sección y automáticamente tu lector de pdfs te llevará a esa página

Tenemos [Repositorio en GitHub](#)

<https://github.com/grupo9-mpe3-2021/arduino-proyecto-timbre>

Todo list

TODA ESTA SECCION ESTA DESACTUALIZADA!!!	21
creo que ese pinout esta mal	21
falta completar esta parte del menu D	22
meter fotos aca	26

Esto no va en el documento final!!!!

Propósito del proyecto

El propósito de este proyecto es garantizar un timbre automatizado para los horarios de recreo en las preceptorías.

Este es un trabajo hecho para la materia de Montaje de Proyectos Electrónicos. En este proyecto aplicamos multiples conocimientos de programación, diagramado de circuitos, armado de PCBs, y otros conocimientos varios.

Problematica

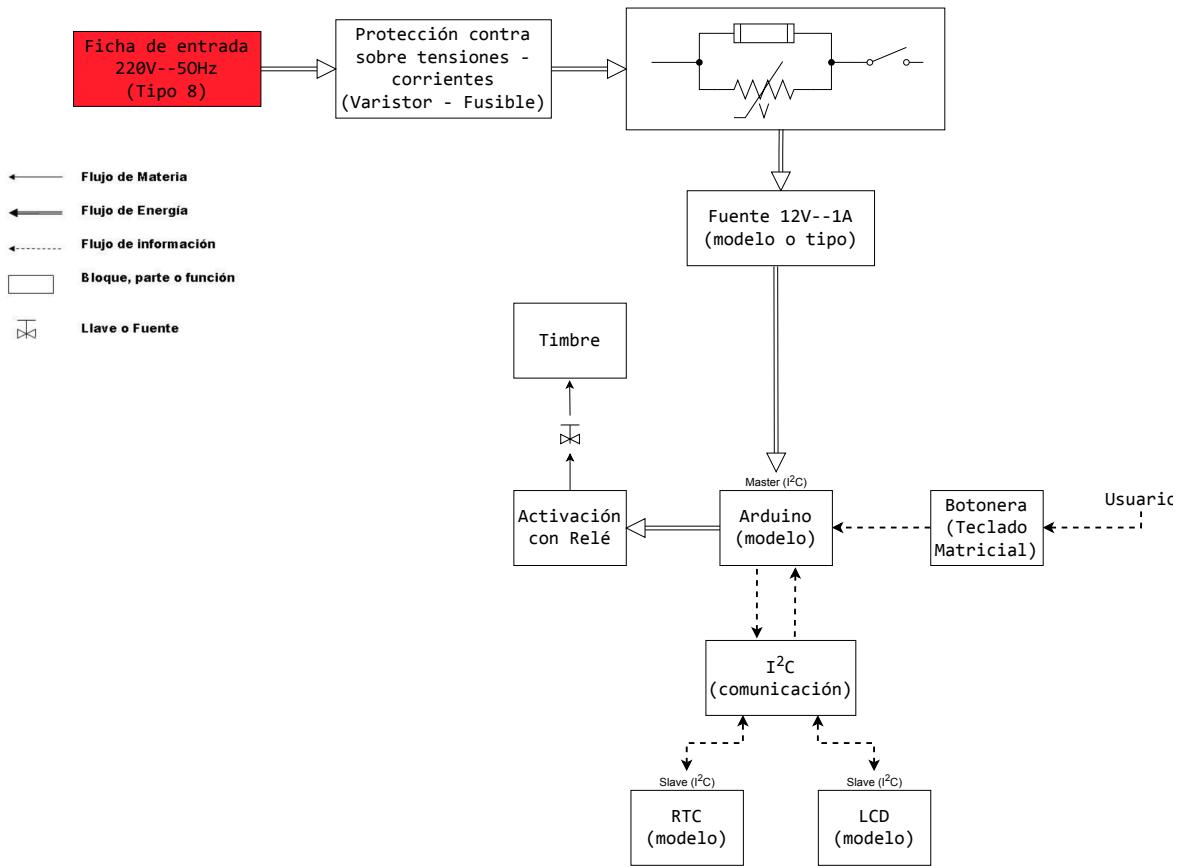
Nos encontramos con una problematica, y era la necesidad de crear un sistema automatizado para los multiples horarios que el colegio maneja. Los preceptores del colegio necesitaban un sistema que les permitiera manejar las alarmas de una forma fácil y sencilla, y este proyecto es la solución a ello.

Solucion que encontramos

Nuestra solucion fue crear un sistema embebido de facil uso, que automatiza un sistema de alarmas, con la posibilidad de agregar alarmas, eliminarlas, activar una alarma de forma manual y ver la hora actual

Diagrama en bloques

El siguiente diagrama representa cómo están relacionados entre sí todos los bloques del proyecto.



Componentes del proyecto

Arduino

¿Qué es arduino?

Arduino es una plataforma de creación de proyectos electrónicos, la cual está basada en hardware y software libre, pensada para ser flexible y fácil de utilizar para los creadores y desarrolladores.

Esta plataforma permite crear diferentes programas usando código en C++, los cuales se cargan al microcontrolador de la placa y permiten al usuario hacer lo que deseé con la misma.

Esta placa tiene unos pines de entrada y salida de datos ubicados en los costados, los cuales permiten conectar los distintos módulos y componentes que sean necesarios para el proyecto a realizar.

Por ejemplo, se puede utilizar un pin para detectar el estado de un botón, y otro pin con un LED conectado, el cual, determinado por el código programado, se podría prender cuando el botón sea pulsado.



Figura 1: Arduino UNO

¿Cómo es usado en este proyecto?

En este proyecto el arduino es usado como placa madre o *mother-board*, en el cual se montan todos los componentes y módulos, y en el cual se graba el código del programa

Protocolo I²C

¿Qué es el protocolo I²C?

El protocolo I²C es un protocolo de comunicación entre circuitos integrados, en el cual se definen dispositivos a funcionar como maestros o *master*, y dispositivos a funcionar como esclavos o *slave*.

Lo que se hace es establecer comunicación entre los dispositivos usando dos líneas:

- **SCL (Serial Clock):** Es la línea que transmite la señal de sincronía. Eléctricamente se trata de una señal a colector o drenador abierto. En un dispositivo esclavo se trata de una entrada, mientras que en un dispositivo maestro es una salida.
El dispositivo maestro genera la señal de sincronía, necesaria para mantener la comunicación entre los dispositivos.
- **SDA (Serial Data):** Es la línea que transmite los datos de forma semi-bidireccional.
Eléctricamente se trata de una señal a colector o drenador abierto. Es gobernada por el emisor, sea éste un maestro o un esclavo.
Sobre esta línea se montan los datos a transmitir entre los dispositivos.



Figura 2: Logo de I²C

Modulo RTC DS3231

¿Qué es el RTC DS3231?

El DS3231 es un reloj en tiempo real (RTC) I²C extremadamente preciso y de bajo costo con un oscilador de cristal integrado con compensación de temperatura (TCXO) y un cristal.

El dispositivo incorpora una alimentación a batería ECR2032 y mantiene un cronometraje preciso cuando se interrumpe la alimentación principal del dispositivo. La integración del oscilador de cristal en el propio módulo mejora la precisión a largo plazo del dispositivo además de evitar la necesidad de implementar uno externo.

El RTC mantiene información de segundos, minutos, horas, día, fecha, mes y año. La fecha al final del mes se ajusta automáticamente por meses con menos de 31 días, incluidas las correcciones por año bisiesto. El reloj funciona en formato de 24 horas o de 12 horas con un indicador AM / PM.

Se proporcionan dos alarmas programables de hora del día y una salida de onda cuadrada programable. La dirección y los datos se transfieren en serie a través de un bus bidireccional I²C.



Figura 3: DS3231

¿Cómo es usado en este proyecto?

En este proyecto el DS3231 es usado en un enlace I²C para mantener un registro preciso de la fecha y hora actual.

Esto es vital para poder mostrar en pantalla la fecha y hora, además de para una correcta activación de las alarmas.

Keypad 4x4

¿Qué es el keypad 4x4?

El keypad 4x4 es un teclado de 16 botones con una distribución de 4 filas y 4 columnas, el cual suele ser usado como teclado para la introducción de información.

La ventaja que el mismo posee es que permite conectar un total de 16 botones al arduino usando solamente 8 pines (cantidad de filas + cantidad de columnas)

¿Cómo es usado en este proyecto?

En este proyecto el mismo se conecta al Arduino para gran parte de la interacción con el usuario que requiere una entrada de información. Se usa para cambiar de menus, agregar alarmas, eliminar alarmas, activar la alarma de forma manual, etc...

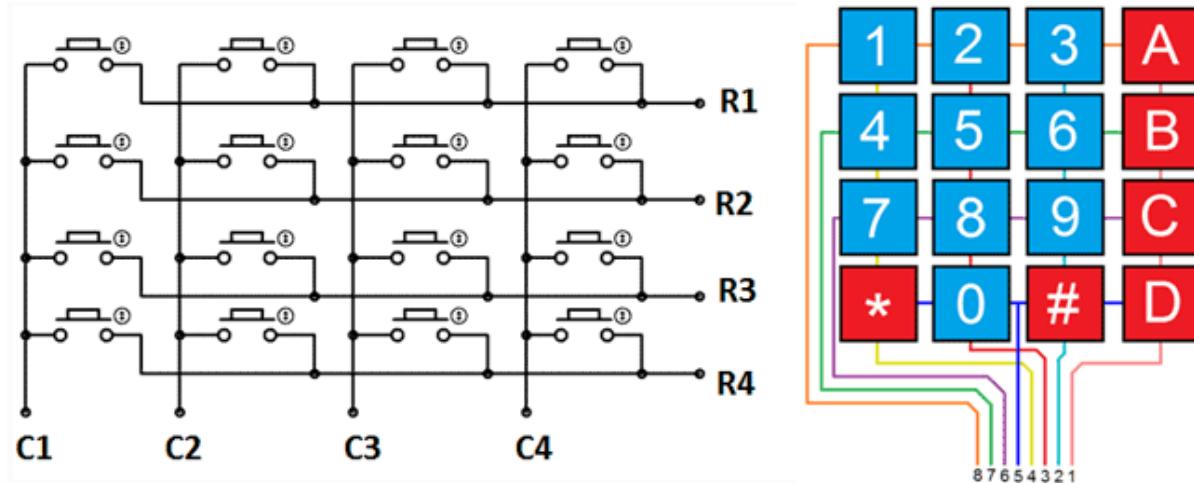
Una de las razones por las que se eligió el uso de este componente es por la facilidad que proporciona al usuario a la hora de introducir información, acompañado de su cómodo tamaño para el ingreso de datos, frente a otras alternativas como el uso de botones en distribución arriba/abajo/izquierda/derecha.



Figura 4: Una foto de un keypad 4x4

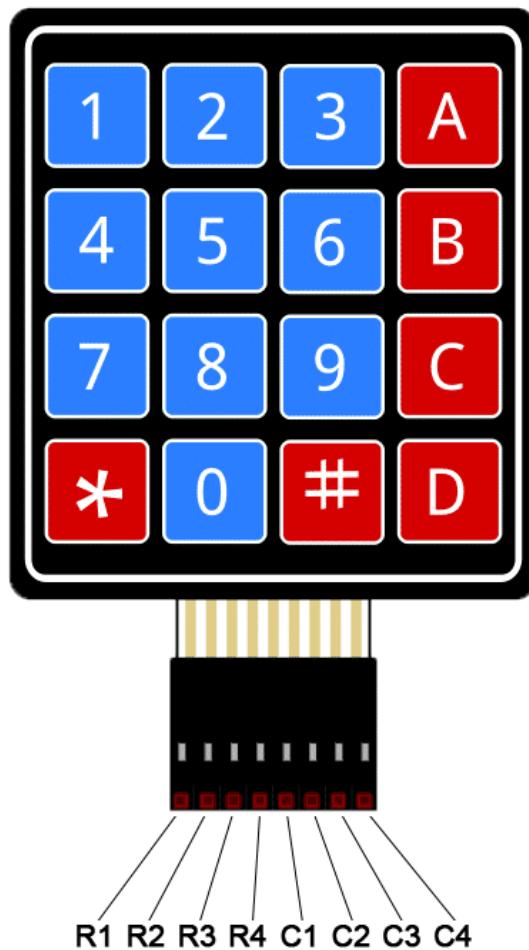
¿Cuál es su equivalencia eléctrica?

Este mismo puede ser visto como una matriz de 16 botones colocados de la siguiente forma:



¿Cuál es su pinout?

Su pinout es el siguiente, siendo R1-4 las filas y C1-4 las columnas



Display LCD I²C 20x4

¿Qué es el display LCD I²C?

El display LCD I²C es una pantalla de cristal líquido de 20 columnas x 4 filas que utiliza el protocolo I²C, el cual proporciona la capacidad de imprimir información usando este mismo protocolo.

¿Cómo es usado en este proyecto?

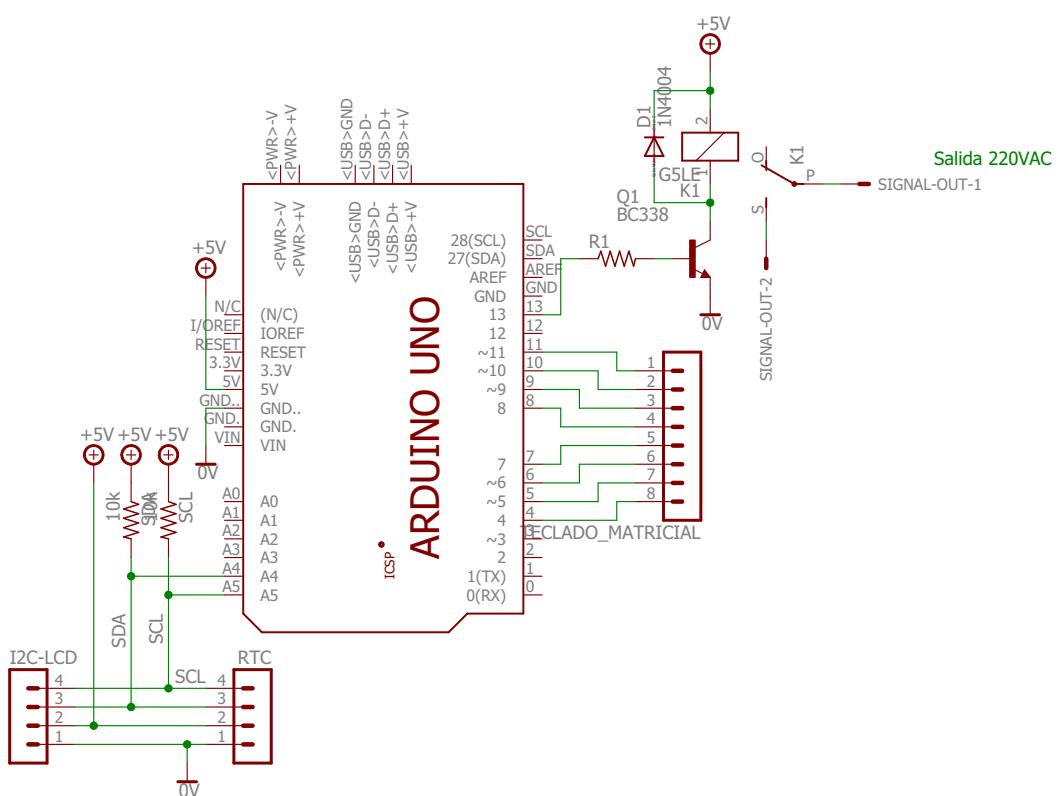
Este display es utilizado para todas las tareas que requieran transmitir información de forma visual al usuario, significando esto tareas tales como:

- Imprimir la fecha y hora
- Imprimir los menus
- Informar la activacion de una alarma



Figura 5: Display LCD 20x4 I²C

Diagrama esquematico



13/10/2021 04:13 p.m. f=1.35 G:\TImbre\PCB timbre.sch (Sheet: 1/1)

Figura 6: El diagrama de las conexiones del PCB, hecho en Eagle

PCB

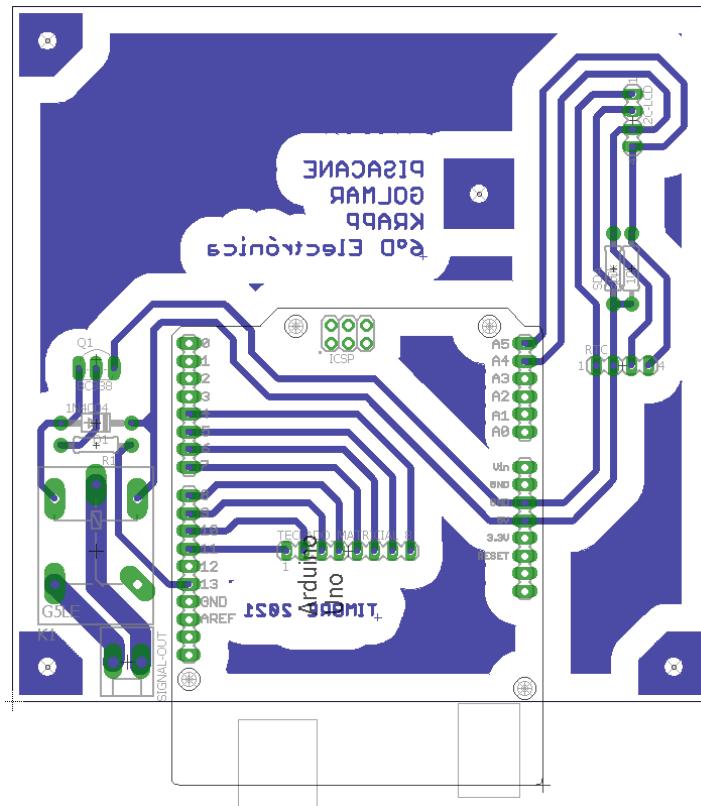


Figura 7: El diseño del PCB hecho en Eagle

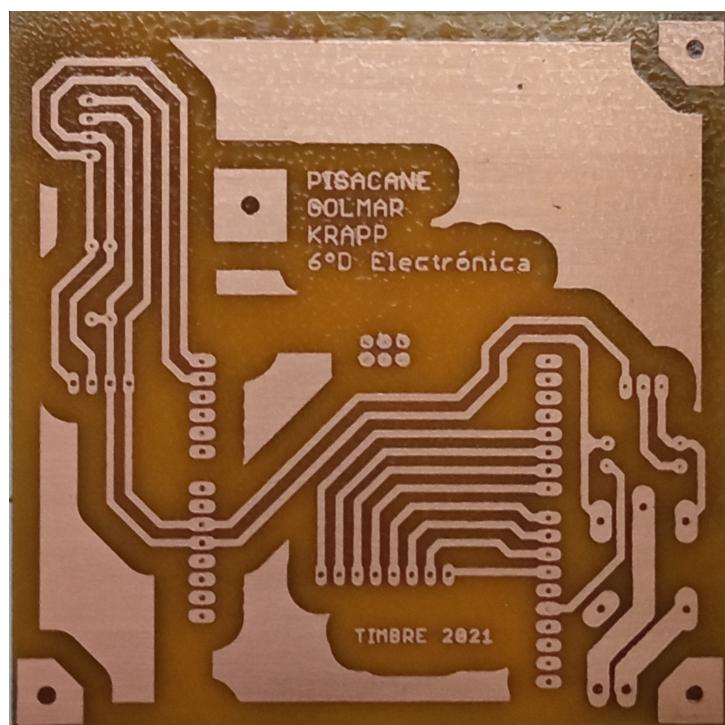
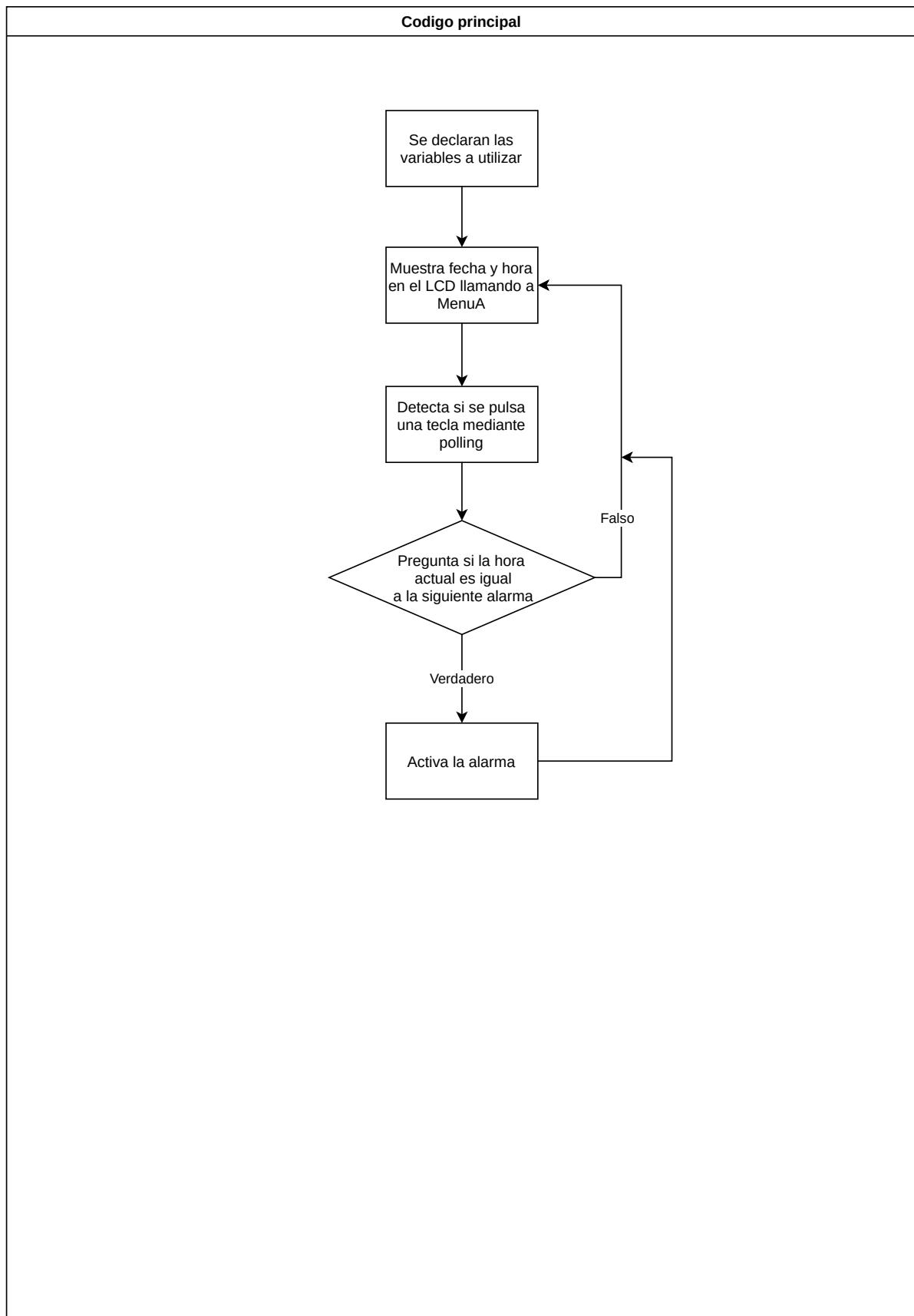
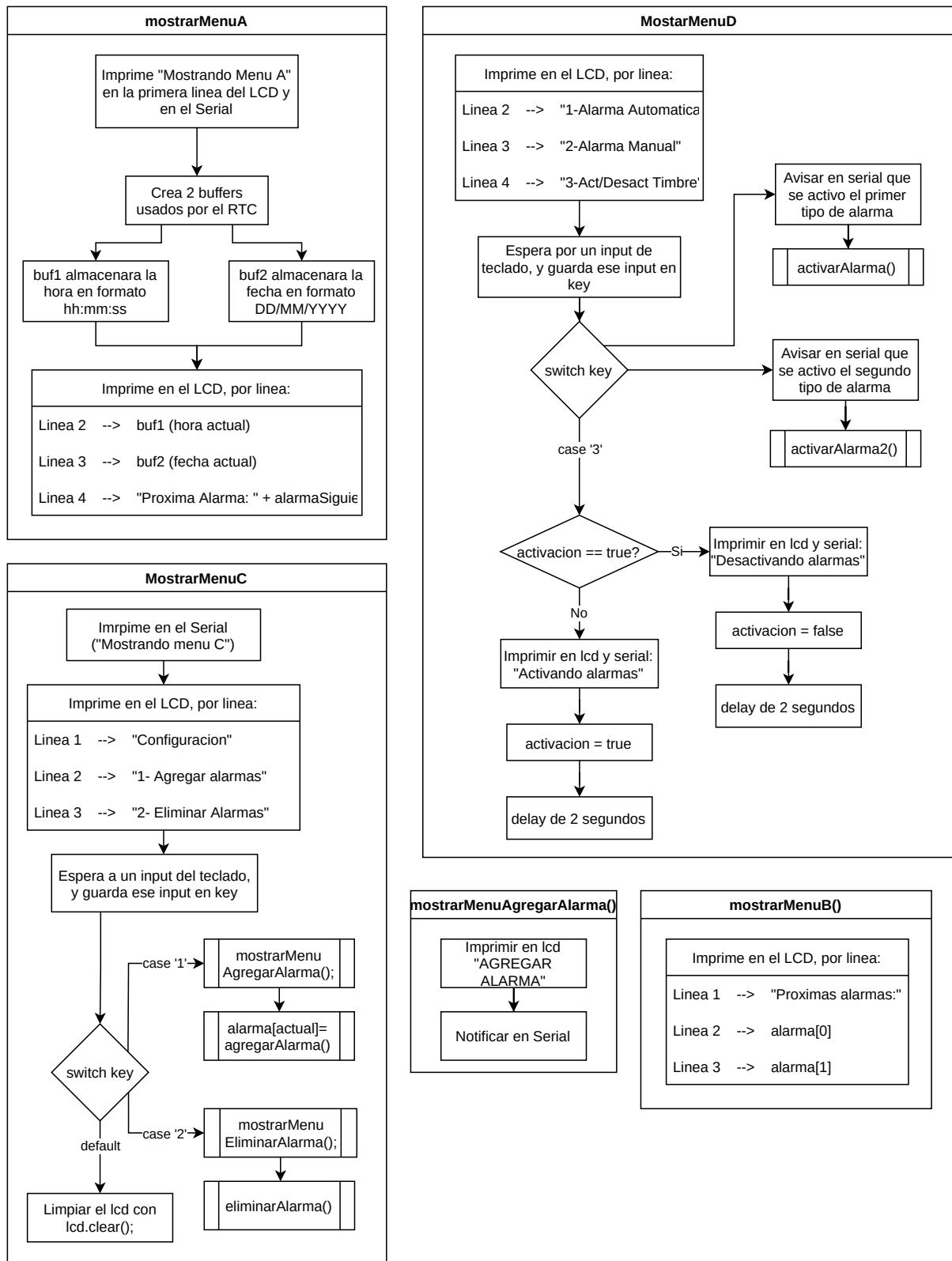
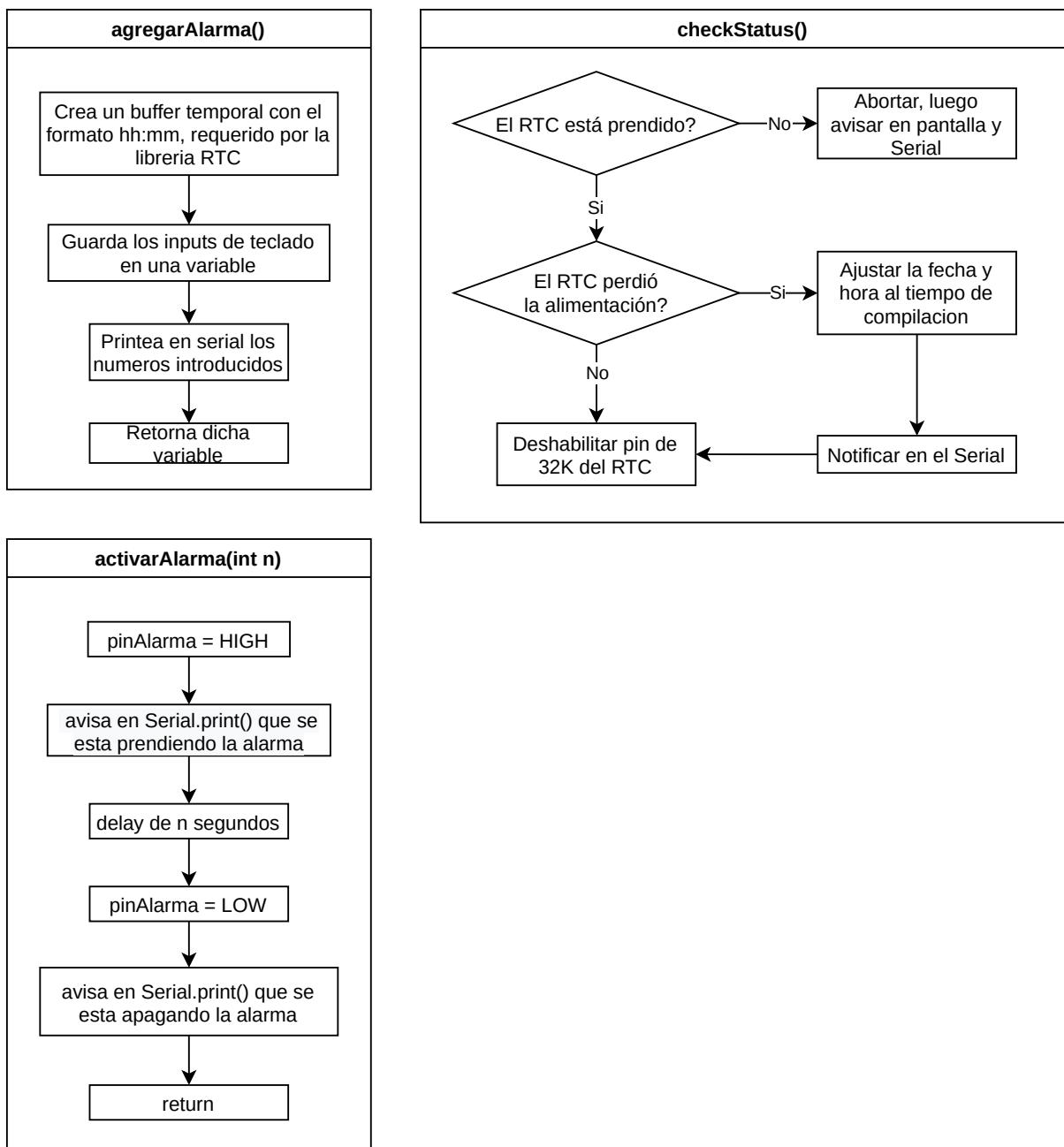


Figura 8: El PCB volcado a la placa de cobre

Diagrama de flujo







Código de programa

Código principal

```
1 #include <LiquidCrystal_I2C.h>
2 #include <Keypad.h>
3 #include <Wire.h>
4 #include <RTClib.h>
5 #define pinINT 2
6 #define timbre 13
7
8 LiquidCrystal_I2C lcd(0x27,20,4);
9 RTC_DS3231 rtc;
10
11
12 //String alarma1 = "21:46"; //defino el formato de la alarma
13 //String alarma2 = "hh:mm"; //defino el formato de la alarma
14
15 int cantidadAlarmas=20; //Cantidad de alarmas. hay que jugar con este valor y size of alarma[] para poder
→ correr bien el codigo. si no se bugea.
16 String alarma[20]; //String alarma[cantidadAlarmas];
17 bool activacion = true;
18 int actual = 0;//define donde arrancan a sonar las alarmas.
19
20 // Variables del RTC :
21 int factorCompensacion = 7; // corregir, creo q no hace nada xd. en distintas versiones perdió su
→ utilidad. hay q conectar el rtc a un código q solo cargue la hora y desde ahí ajustarla.
22 char date[10] = "hh:mm:ss";
23 String daysOfTheWeek[7] = { "Domingo", "Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado" };
24 String monthsNames[12] = { "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto",
→ "Septiembre", "Octubre", "Noviembre", "Diciembre" };
25
26 //Variables del Keypad
27 char key;
28 const int cantidadFilas = 4; //cuatro filas
29 const int cantidadColumnas = 4; //cuatro columnas
30 char teclas[cantidadFilas][cantidadColumnas] = {
31     {'1', '2', '3', 'A'},
32     {'4', '5', '6', 'B'},
33     {'7', '8', '9', 'C'},
34     {'*', '0', '#', 'D'}
35 };
36 byte pinFilas[cantidadFilas] = {4,5,6,7}; //este es el pinout de las filas del teclado
37 byte pinColumnas[cantidadColumnas] = {8,9,10,11}; //este es el pinout de las columnas del teclado
38 Keypad keypad = Keypad( makeKeymap(teclas), pinFilas, pinColumnas, cantidadFilas, cantidadColumnas );
39
40
41 void setup() {
42     Serial.begin(9600);
43
44     // Setup LCD.
45     lcd.init();
46     lcd.backlight();
47
48     // Setup RTC.
49     checkStatus();
50     //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
51     //alarmas default.
52
53     alarma [0] = "07:40";
54     alarma [1] = "09:40";
55     alarma [2] = "10:00";
56     alarma [3] = "12:00";
57     alarma [4] = "13:00";
58     alarma [5] = "15:45";
59     alarma [6] = "16:00";
60     alarma [7] = "17:15";
61     alarma [8] = "22:40";
62     alarma [9] = "23:54";
63
64     alarma [10] = "06:30";
65     alarma [11] = "06:35";
66     alarma [12] = "14:25";
```

```

67     alarma [13] = "16:25";
68     alarma [14] = "17:00";
69     alarma [15] = "18:00";
70     alarma [16] = "XX:XX";
71     alarma [17] = "XX:XX";
72     alarma [18] = "XX:XX";
73     alarma [19] = "XX:XX";
74
75     actualizarAlarma(); //necesito que se ponga al corriente segun que alarma es la entrante.
76
77     Serial.println(F("Setup finalizado"));
78
79     /* con goto y (retorno): podemos volver a donde querramos, a octi le funciona :D.
80      * podriamos volver de los menus con goto a ver si funciona, es la mejor manera creo yo.
81      * hay que probar, si agrega, si elimina y si suenan las alarmas
82      * estamos peor q sanabria daps de tirar la silla.
83    */
84 }
85
86 void loop() {
87
88     DateTime now = rtc.now(); // actualiza la hora.
89     //campanita();
90     lcd.setCursor (6,0);
91     lcd.print (F("Inicio"));
92     mostrarMenuA();
93
94     key = keypad.getKey(); // lee lo que te deje en la pestaña de menus.
95     switch (key) { // navegacion de los menus
96
97         case 'B':
98             lcd.clear();
99             lcd.setCursor (2,0);
100            lcd.print(F("Mostrando menu B")); Serial.println(F("Mostrando menu B"));
101            delay(1000); //modificar o reorganizar.
102            lcd.clear();
103            mostrarMenuB();
104            break;
105
106        case 'C':
107            lcd.clear();
108            lcd.setCursor (2,0);
109            lcd.print(F("Mostrando menu C")); Serial.println(F("Mostrando menu C"));
110            mostrarMenuC();
111            //alarma1= agregarAlarma();
112            break;
113
114        case 'D':
115            lcd.clear();
116            lcd.setCursor (2,0);
117            lcd.print(F("Mostrando menu D")); Serial.println(F("Mostrando menu D"));
118            mostrarMenuD();
119            break;
120
121    }
122
123    char buf3[] = "hh:mm";
124    now.toString(buf3); // pasa la hora al string buf3, y elimina los segundos como parametro.
125    Serial.println (buf3);
126
127    if (activacion == true && alarma[actual].compareTo(buf3) == 0 ){ // permite desactivar y activar las
128        ↪ alarmas.
129        activarAlarma();
130        actual++;
131    }
132    else if (activacion == false && alarma[actual].compareTo(buf3) == 0 ){
133        actual++;
134    }
135    if (actual == cantidadAlarmas ){
136        actual = 0;
137    }
138 }
```

Declaracion de funciones

```
1 void checkStatus() {
2     // si el rtc no se inicializa:
3     if (!rtc.begin()) {
4         lcd.print(F("No se encontro RTC")); Serial.println(F("No se encontro RTC"));
5         abort();
6     }
7     else {
8         Serial.println(F("Se ha encontrado el RTC correctamente"));
9     }
10    // si el rtc perdió la alimentación:
11    if (rtc.lostPower()) {
12        // ajusta la fecha y hora al tiempo de compilación
13        Serial.println(F("El RTC perdió la alimentación")); Serial.println(F("ajustando la fecha y hora al
→ tiempo de compilación"));
14        rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
15    }
16    rtc.disable32K();
17    Serial.println(F("Finalizado el checkeo de status"));
18 }
19
20 int activarAlarma() { //falta cambiar por millis()
21     lcd.clear();
22
23     digitalWrite(timbre, HIGH); Serial.println(F("Activando alarma set1!"));
24     lcd.setCursor(2, 1);
25     lcd.print(F("Alarma Activada"));
26     delay(3000); // cambiar tiempo de activación.
27
28     digitalWrite(timbre, LOW); Serial.println(F("Desactivando alarma!"));
29     lcd.clear();
30     lcd.setCursor(0, 1);
31     lcd.print(F("Alarma Desactivada"));
32     delay(1500);
33
34     lcd.clear();
35 }
36
37 int activarAlarma2() { //testear funcionamiento.
38
39     lcd.clear();
40     digitalWrite(timbre, HIGH); Serial.println(F("Activando alarma set2!"));
41
42     lcd.setCursor(4, 1);
43     lcd.print(F("-Pulsar '*'"));
44     lcd.setCursor(0, 2);
45     lcd.print(F("Desactiva la Alarma")); Serial.println(F("Pulsar '*' para desactivar la Alarma"));
46
47     keypad.waitForKey() == '*' // espera a que se pulse '*' para desactivar la alarma.
48     lcd.clear();
49     digitalWrite(timbre, LOW); Serial.println(F("Desactivando alarma!"));
50 }
51
52 String agregarAlarma() {
53
54     lcd.setCursor(0, 1);
55     lcd.print(F("Formato: 'hh:mm'")); Serial.print(F("Agregando Alarma..."));
56
57     /*
58      Esta función recibe inputs del teclado y los mete a un String llamado alarma, el cual luego retorna
59      Args:
60          ninguno
61      Returns:
62          un String con la hora introducida en el teclado
63     */
64
65     String aux1 = "hh:mm"; //defino el formato de la alarma //habría que poner esto afuera <-- no se que
66     // quisiera decir con este comentario, evidentemente con el return se puede hacer miAlarma = agregarAlarma()
67     // y listo
68     lcd.setCursor(0,2);
69     for (int i = 0 ; i < 5 ; i++) {
70         if (aux1[i] != ':') { //para evitar que me sobreescriba el ":"
```

```

70     aux1[i] = keypad.waitForKey();
71     lcd.print(aux1[i]);    Serial.print(F("introducida la tecla "));    Serial.println(aux1[i]);
72   }
73 }
74 Serial.print(F("Saliendo!!!!"));
75 return aux1;
76 }
77 void actualizarAlarma() { // fija la siguiente alarma luego de un reset o power on.
78
79   DateTime now = rtc.now();
80
81   char buf4[] = "XX:XX";
82   char buf5[] = "hh:mm";
83   now.toString(buf5);
84   actual = 0;
85   int aux2 = 0;
86   for (int y = 0; y < cantidadAlarmas ; y++){
87     if ( alarma[y].compareTo(buf4) == 0 ){
88
89     }
90   }
91
92   for (int z = 0; z < cantidadAlarmas ; z++) { //alarma[actual].compareTo(buf4) <= 0
93     if ( alarma[actual].compareTo(buf5) <= 0 ) { // permite desactivar y activar las alarmas.
94       actual++;
95       if (actual == cantidadAlarmas ) {
96         actual = 0;
97       }
98     }
99     Serial.println (F("alarma actualizada"));
100  }
101 }
102 /*
103  * bool checkearAlarmaIgualHora(DateTime date, String alarma) {
104  * Esta funcion checkea si la hora actual es igual a la hora de la alarma
105  * Args:
106  *     Hora actual
107  *     La hora de la alarma (string)
108  * Returns:
109  *     true o false dependiendo si alarma == date
110  *
111  * Serial.println(F("Checkeando si la alarma es igual a la hora... "));
112  * char buffer[] = "hh:mm";      // hago un buffer con el formato que quiero para mi hora
113  * date.toString(buffer);        // guardo la hora en el buffer
114  * String aux = String(buffer); // convierto mi char[] en String para poder compararlo con la alarma
115  * //necesito el true de abajo,
116  *
117  * return aux.equals(alarma[actual]); // pregunto si la hora actual y la hora de la alarma son ==
118  * // Si es es asi, devuelve true, en caso contrario, devuelve false
119  *
120  */
121 }
122
123 compareTo()
124 [StringObject Function]
125 Description
126
127 Compares two Strings, testing whether one comes before or after the other, or whether they're equal. The
128 strings are compared character by character, using the ASCII values of the characters. That means, for
129 example, that 'a' comes before 'b' but after 'A'. Numbers come before letters.
130 Syntax
131 myString.compareTo(myString2)
132 Parameters
133
134 myString: a variable of type String.
135 myString2: another variable of type String.
136 Returns
137
138 a negative number: if myString comes before myString2.
139 0: if String equals myString2.
140 a positive number: if myString comes after myString2.

```

¹⁴¹

¹⁴² */

Declaracion de menus

```
1 void mostrarMenuA() {
2     // mejorar estetica
3     lcd.setCursor (15,0);
4     if (activacion == true){
5         lcd.print(F("T:Act"));
6     }
7     else if (activacion == false) {
8         lcd.print (F("T:Des"));
9     }
10
11     DateTime now = rtc.now();
12
13     char buf1[] = "hh:mm:ss";
14     char buf2[] = "DD/MM/YYYY";
15
16     lcd.setCursor(0, 1);    lcd.print(now.toString(buf1));      //Serial.println(now.toString(buf1));
17     lcd.setCursor(0, 2);    lcd.print(now.toString(buf2));      //Serial.println(now.toString(buf2));
18     lcd.setCursor(0, 3);    lcd.print(F("Prox Alarma: "));   lcd.print(alarma[actual]);
19     //Serial.println("Sigiente Alarma: ");
20     //delay(999);    //delay para evitar que se pase de rosca el display.
21 }
22
23 void mostrarMenuB() {
24     // ver si funciona
25     lcd.setCursor(0, 0);    lcd.print(F("Proximas alarmas:"));
26     lcd.setCursor(0, 1);    lcd.print(F("Alarma 1: "));    lcd.print(alarma[actual]);
27     //Serial.println(F(alarma[actual]));
28     lcd.setCursor(0, 2);    lcd.print(F("Alarma 2: "));    lcd.print(alarma[actual+1]);
29     //Serial.println(F(alarma[actual+1]));
30     lcd.setCursor(0, 3);    lcd.print(F("Alarma 3: "));    lcd.print(alarma[actual+2]);
31     //Serial.println(F(alarma[actual+2]));
32     for (int i=0 ; i<cantidadAlarmas ; i++){
33         Serial.println (alarma[i]);
34     }
35
36     keypad.waitForKey();
37     lcd.clear();
38 }
39
40 void mostrarMenuC() {
41     // ver si funciona
42     lcd.setCursor(0, 1);    lcd.print(F("Configuracion:"));
43     lcd.setCursor(0, 2);    lcd.print(F("1- Agregar Alarmas"));
44     lcd.setCursor(0, 3);    lcd.print(F("2- Eliminar Alarmas"));
45
46     key = keypad.waitForKey();  Serial.print(F("introducida la tecla "));  Serial.println(key);
47     switch (key) {
48         case '1':
49             lcd.clear();
50             mostrarMenuAgregarAlarma(); // corregir funcionamiento
51             alarma[2] = agregarAlarma();
52             lcd.clear();
53             break;
54         case '2':
55             lcd.clear();
56             mostrarMenuEliminarAlarma(); // corregir funcionamiento
57             //eliminarAlarma();
58             lcd.clear();
59             break;
60     }
61 }
62
63 void mostrarMenuEliminarAlarma(){ // por ahora magia hace esto xd.
64     Serial.println(F("Mostrando menu de eliminar alarma"));
65 }
66
67 int mostrarMenuAgregarAlarma() {
68     Serial.println(F("Mostrando menu para agregar alarma"));
69     lcd.setCursor(0, 0);    lcd.print(F("AGREGAR ALARMA "));
```

```

69 }
70
71 void mostrarMenuD() { // activa el timbre hasta que toques de nuevo D.
72     // ver si funciona
73     lcd.setCursor (0,1);
74     lcd.print (F("1-Alarma Automatica"));
75
76     lcd.setCursor (0,2);
77     lcd.print (F("2-Alarma Manual"));
78
79     lcd.setCursor (0,3);
80     lcd.print (F("3-Act/Desact Timbre"));
81
82     key = keypad.waitForKey();
83     switch (key){
84
85         case '1':
86             lcd.clear();
87             Serial.println(F("Activando caso ,alarma Set1"));
88             activarAlarma();
89             break;
90
91         case '2':
92             lcd.clear();
93             lcd.print(F(" Activando Alarma")); Serial.println(F("Activando caso ,alarma Set2"));
94             activarAlarma2();
95             break;
96
97         case '3':
98             lcd.clear();
99             if (activacion == true){
100
101                 lcd.setCursor(3,1);
102                 lcd.print("*Advertencia*");
103
104                 lcd.setCursor(0,2);
105                 lcd.print(F("Desactivando Alarmas")); Serial.println(F("La activacion de alarmas fue
106             ↪ desactivada."));
107
108                 activacion = false;
109
110                 delay (2000);
111                 lcd.clear();
112             }
113             else if (activacion == false){
114
115                 lcd.setCursor(3,1);
116                 lcd.print(F("*Advertencia*"));
117
118                 lcd.setCursor(2,2);
119                 lcd.print(F("Activando Alarmas")); Serial.println(F("La activacion de alarmas fue
119             ↪ activada."));
120
121                 activacion = true;
122
123                 delay (2000);
124                 lcd.clear();
125             }
126             break;
127             default: lcd.clear();
128     }
}

```

Explicación del código

Se dividió en 3 pestañas principales

TODA ESTA SECCION ESTA DESACTUALIZADA!!!

Setup_deficiones.ino

En esta pestaña se declaran las variables a utilizar, se declara el void setup() y se declara el programa principal en void loop().

Lo más destacable es lo siguiente

Primero se declaran las librerías a utilizar, las cuales son:

1. LiquidCrystal_I2C — Para el enlace I²C con el LCD
2. Keypad — Para usar el keypad 4x4
3. Wire — Librería auxiliar
4. RTClib — Para obtener la fecha y hora del RTC

Y luego se declaran los pines a utilizar, los cuales son

1. pinINT — en el pin 12
2. timbre — en el pin 13

Luego, se declara lo siguiente

```
1 LiquidCrystal_I2C lcd(0x27, 20, 4);
2 RTC_DS3231 rtc;
```

Lo cual hace dos cosas:

Primero, se declara que el display lcd usara el prefijo lcd para sus funciones, por ejemplo, lcd.print(), o lcd.clear(). Luego, se declaran 3 cosas

- 0x27 — La dirección de memoria
- 20 — La cantidad de filas
- 4 — La cantidad de columnas

Lo segundo, declara que el RTC DS3231 va a usar el prefijo rtc para sus funciones, por ejemplo, para invocar la función now() se usaría rtc.now()

Luego, en

```
1 char teclas[cantidadFilas][cantidadColumnas] = {
2 {'1', '2', '3', 'A'},
3 {'4', '5', '6', 'B'},
4 {'7', '8', '9', 'C'},
5 {'*', '0', '#', 'D'}
6 };
7 byte pinFilas[cantidadFilas] = {4,5,6,7}; //este es el pinout de las filas del teclado
8 byte pinColumnas[cantidadColumnas] = {8,9,10,11}; //este es el pinout de las columnas del teclado
9 Keypad keypad = Keypad( makeKeymap(teclas), pinFilas, pinColumnas, cantidadFilas, cantidadColumnas );
```

Se declara una matriz que será la distribución de las teclas del teclado matricial y los pines a donde debe estar conectado en en el Arduino.

Luego, se llama a la función mostrarMenuA(), e inmediatamente después de eso, se compara la hora actual con la siguiente alarma

creo que ese pinout esta mal

menus.ino

En esta pestaña se declaran las funciones que pertenecerán a los menus, los cuales serían printeados en el LCD

En esta pestaña se definen las siguientes funciones:

- mostrarMenuA()
El cual muestra la fecha y hora actual

- mostrarMenuB()
El cual muestra las proximas alarmas

- mostrarMenuC()
El cual muestra el menu de configuración, que tiene 3 opciones:

1. Agregar Alarmas — Invoca la función mostrarMenuAgregarAlarma()
2. Eliminar Alarma — Invoca la función mostrarMenuEliminarAlarma()
3. Alarma manual — Activa manualmente la alarma

- mostrarMenuD()
El cual muestra las siguientes 3 opciones:

- 1.
- 2.
- 3.

- mostrarMenuEliminarAlarma()
El cual muestra en el LCD el menu para eliminar las alarmas
- mostrarMenuAgregarAlarma()
El cual muestra en el LCD el menu para añadir alarmas

falta com-
pletar esta
parte del
menu D

acciones.ino

En esta pestaña se declaran todas las funciones que funcionaran como acciones, como activarAlarma() o agregarAlarma() En esta pestaña definí las siguientes funciones:

- checkStatus()

Hace las siguientes cosas:

- Revisa si el RTC se inicio correctamente
- Revisa si el RTC perdio su alimentación
- Desactiva el pin de 32K

- activarAlarma()

Activa la alarma

- agregarAlarma()

Recibe un input de teclado matricial, y eso lo guarda en una variable

- checkearAlarmaIgualHora (DateTime date, String alarma)

Esta funcion checea si la hora actual es igual a la alarma pasada por el segundo parametro

Bitácora

Krapp

Primero investigue cómo obtener la hora del RTC e imprimirla en el arduino.
A partir de este momento, me dí cuenta de dos cosas que ayudarían al desarrollo del código:

- Comenzar a subir el código a un repositorio remoto en github
- Hacer Serial.print en las acciones principales para cuestiones de debug

Luego hice un código que imprimía la hora en un display LCD I²C, pero era altamente ineficiente:

```
1 void printDate(DateTime date)
2 {
3
4     lcd.print(daysOfTheWeek[date.dayOfTheWeek()]);
5     lcd.print("  ");
6
7     sprintf(buf, "%02d", date.day());
8     lcd.print(buf);
9     lcd.print('/');
10
11    sprintf(buf, "%02d", date.month());
12    lcd.print(buf);
13    lcd.print('/');
14
15    sprintf(buf, "%04d", date.year());
16    lcd.print(buf);
17
18    lcd.setCursor(0, 1);
19    lcd.print((int)rtc.getTemperature());
20    lcd.print((char)0xDF);
21    lcd.println("C");
22    lcd.print(" ");
23
24    sprintf(buf, "%02d", date.hour());
25    lcd.print(buf);
26    lcd.print(':');
27
28    sprintf(buf, "%02d", date.minute());
29    lcd.print(buf);
30    lcd.print(':');
31
32    sprintf(buf, "%02d", date.second());
33    lcd.print(buf);
34    lcd.print(" ");
35 }
```

Mirando en la documentación de la librería RTCLib, encontre un código que imprimía la hora en el Serial, que era mucho más simple y eficiente, entonces lo adapte a lcd con el siguiente resultado:

```
1 DateTime now = rtc.now();
2
3 char buf1[] = "hh:mm:ss";
4 char buf2[] = "DD/MM/YYYY";
5
6 lcd.setCursor(0, 1);    lcd.print(now.toString(buf1));      //Serial.println(now.toString(buf1));
7 lcd.setCursor(0, 2);    lcd.print(now.toString(buf2));      //Serial.println(now.toString(buf2));
8 lcd.setCursor(0, 3);    lcd.print("Prox Alarma: ");    lcd.print(alarma[actual]);
↪ //Serial.println("Siguiente Alarma: ");
```

Al ver que el proyecto se iba a hacer gigante, hablé con mis compañeros y propuse dividir el código en distintas pestañas, cada una con su propia razón para existir.

Acercándose el fin del més de octubre, comence a realizar el informe técnico, ya que el mismo equivalía al 50 % de la nota final. Entonces, se tomó la decisión de escribir el informe en L^AT_EX.

Lo primero que hice fue definir la estructura principal del informe colocando multiples section, y a partir de ahí comencé a escribir, con la ayuda de mis compañeros, las secciones que correspondian.

Cuando me encontraba con algo que necesitaba hacer y no sabía como, simplemente lo buscaba en internet, y facilmente obtenía la respuesta.

Para escribir el archivo .tex, empecé a usar TexMaker, pero luego más tarde comence a usar el editor de texto Neovim debido a su gran cantidad de atajos de teclados y ligereza en cuanto a recursos.

Las 3 cosas principales que aprendí son las siguientes:

1. Dividir el código en distintas funciones y archivos con su función específica
2. Mantener un historial de cambios de forma ordenada usando git
3. Hacer código pensando en el debug futuro

Anexos

meter fotos
aca



Figura 9: Una foto de la conexión del keypad 4x4 al arduino UNO

Bibliografía y consultas

1. Arduino — Documentacion libreria keypad
<https://www.arduino.cc/reference/en/libraries/keypad/>
2. Adafruit — Documentación libreria RTClib
<https://github.com/adafruit/RTClib>
3. Overleaf —Guia para aprender L^AT_EX, usado para hacer el documento
<https://www.overleaf.com/learn/latex/>
4. Documentación para introducir código fuente en L^AT_EX
<https://github.com/gpoore/minted>
5. Apuntes sobre el bus I²C
https://sanbonifacio.kimeln.com.ar/pluginfile.php/11197/mod_resource/content/1/Bus%20I2C.pdf
6. Especificación del bus I²C
<https://I2C.info/I2C-bus-specification>