INFORME DE PROYECTO Y DISEÑO ELECTRÓNICO

Prof.: Pujol Gabriel

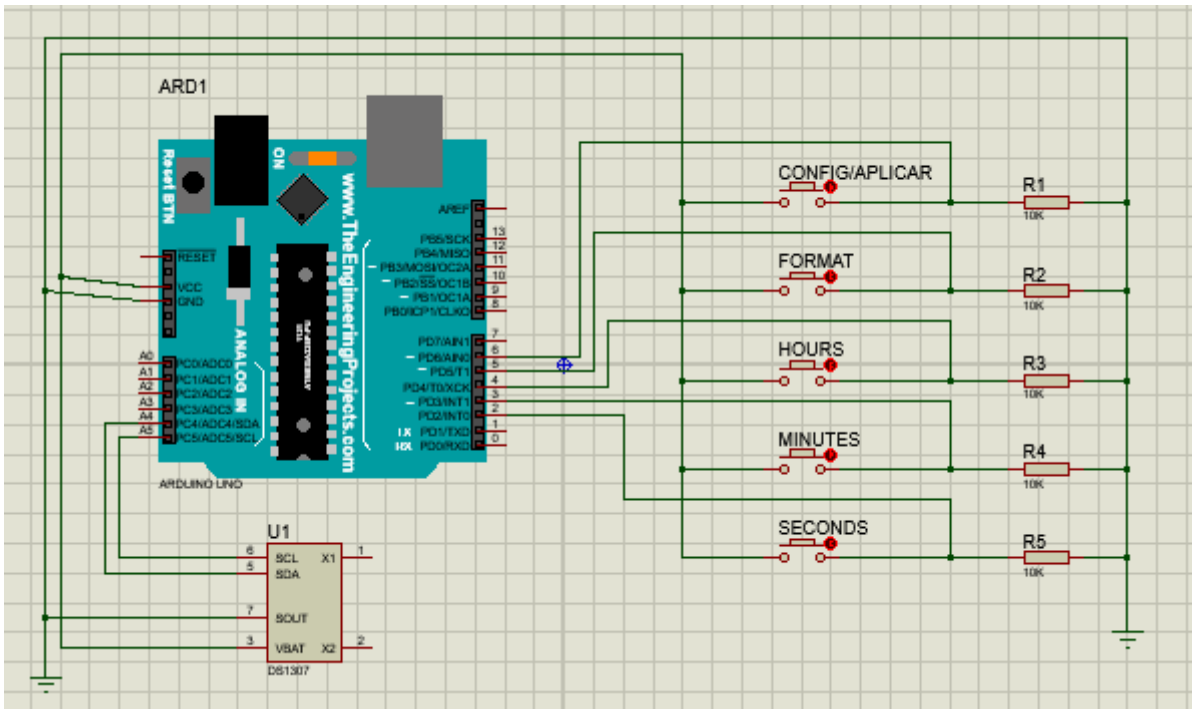
Alumno : Gavotti Lautaro Exequiel
Curso: 7mo 4ta
Año: 2022

Indice:

1. Introducción

El proyecto es un reloj, utilizando Arduino UNO y como RTC el módulo DS3231, que muestra en la pantalla las horas, minutos y segundos. Este se puede configurar con una botonera el tiempo en pantalla y cambiar entre formato 24hs y 12hs AM/PM. La configuración del tiempo se retiene una vez desconectada la placa gracias al RTC y el formato se retiene gracias al EEPROM incorporado en el Arduino UNO.

2. El esquema es el siguiente:



3. Código:
https://github.com/gabii99/rtc_proyect/blob/main/pyde_rtc_first.ino

4. Explicación del codigo:

```
1  #include <Wire.h>      // incluye libreria para interfaz I2C
2  #include <RTClib.h>      // incluye libreria para el manejo del modulo RTC
3  #include <EEPROM.h>      // para el manejo de retención de estados
4
5  RTC_DS3231 rtc;     // crea objeto del tipo RTC_DS3231
6
7  //variables
8  int xhour,xminute,xsecond;
9  int set_hour, set_minute, set_second;
10 int lastState;
11
12 //pin set
13 const int pinButtonConfig = 2;
14 const int pinButtonFormat = 3;
15 const int pinButtonH = 4;
16 const int pinButtonM = 5;
17 const int pinButtonS = 6;
18
19 //flags
20 bool configFlag = false;
21 bool hourFlag = false;
22 bool minuteFlag = false;
23 bool secondFlag = false;
24 bool aplicarFlag = false;
25 bool changeFormat = false;
```

[1 - 3] Empezamos con la inclusión de las librerias para utilizar el RTC y el EEPROM
[5 - 25] Se establecen las variables, las flags y se realiza un setting de los pines a utilizar.
- las variables “xhour, xminute, xsecond” son usadas mas adelante para cambiar el valor del tiempo
- las variables “set_hour, set_minute, set_second” son las que retienen los valores del tiempo una vez realizada la configuración.
- la variable “lastState” se usa como retención de valor con EEPROM para el cambio entre formato 24hs y 12hs AM/PM.
[13 – 17] Se establecen constantes para el setting de los pines y tener mayor comodidad

```
27 void setup() {
28   Serial.begin(9600);
29
30   if (! rtc.begin()) {     // si falla la inicializacion del modulo
31    Serial.println("Modulo RTC no encontrado !");
32    while (1);
33   }
34   if (rtc.lostPower()) {      // si hay problemas de energia
35     Serial.println("RTC lost power, let's set the time!");
36     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
37   }
38
39   //setting inputs
40   pinMode(pinButtonConfig, INPUT);
41   pinMode(pinButtonFormat, INPUT);
42   pinMode(pinButtonH, INPUT);
43   pinMode(pinButtonM, INPUT);
44   pinMode(pinButtonS, INPUT);
45
46   //rtc.adjust(DateTime(__DATE__, __TIME__));
47 }
```

[27 – 47] la función ‘void setup()’ lo lee la placa una sola vez al compilar
[28] Para desplegar la pantalla donde se verá el reloj
[30 – 33] un if statement por seguridad si no llega a iniciar el módulo
[34 – 37] otro if statement por seguridad si llega a haber problemas de energía
[39 – 44] realizando el setting de los pines como ‘input’ utilizando las variables anteriores entre las lineas [13 – 17]
[46] la linea comentada fue para configurar el RTC a la hora actual del equipo el cual compila

```
49  void loop() {
50
51    while(configFlag == false) {
52
53      //para después de haber desconectado el arduino
54
55      //recuerda la configuracion del tiempo
56      DateTime nowTime = rtc.now();
57      //para poder cambiar el tiempo desde el que esta establecido
58      xhour = nowTime.hour();
59      xminute = nowTime.minute();
60      xsecond = nowTime.second();
61      //mantener los tiempos configurados
62      set_hour = nowTime.hour();
63      set_minute = nowTime.minute();
64      set_second = nowTime.second();
65      //recuerda el estado ya escrito en el eeprom
66      lastState = EEPROM.read(0);
67
68      //muestrar el tiempo
69      printTime();
70
71      // ir a modo configuracion
72      while (digitalRead(pinButtonConfig) == HIGH) {
73        delay(3000);
74        if (digitalRead(pinButtonConfig) == HIGH && configFlag == false) {
75          Serial.println("modo configuración");
76          configFlag = true;
77        }
78      }
79    }
```

NOTA!: Entre las linea [49 – 168] está escrito la función ‘void loop()’

- La función ‘void loop()’ esta constituida por dos partes separadas: una es la que muestra en pantalla cada segundo desde el tiempo establecido y otra que deja en pausa a la anterior y permite configurar el tiempo. De esta forma permite leer el código comodamente.

[51 – 79] Primera parte el loop donde muestra el tiempo en pantalla. Este muestreo ocurre mientras la bandera (flag) ‘configFlag’ tenga el valor ‘false’, que ya esta determinado con ese valor previamente en las primeras lineas.
[57 – 66] Entre estas lineas se establece todo lo que necesita retener un valor después de haber desconectado la placa:
- [56] Esta linea toma el tiempo establecido actualmente en el RTC para después poder utilizar sus variables internas
- [58 – 60] Si no se retuviera el valor de estas variables, cada vez que se quiera cambiar las horas, minutos y segundos se haría desde 0 y no desde el tiempo que marca el reloj

- [62 – 64] Si no se retuviera el valor de estas variables, cada vez que se desconecte, se vuelva a conectar, se quiera configurar el tiempo y sin haber configurado nada, al aplicar los cambios se establecería el tiempo a 00hs 00mm 00ss.
- [66] Esta linea lee la celda que ya fue escrita mas adelante
[69] Llama a la función 'printTime()' que contiene el código que muestra el tiempo (se verá al final)

[72 – 78] Modo configuración: este pedazo de código permite salir del 'while' que mantiene muestrando el tiempo. Se hace manteniendo mínimo 3 segundos el botón 'pinButtonConfig' y soltándolo. De esta forma solo se podrá configurar el tiempo si se realiza este paso, sin esto ninguno de los otros botones funcionaría

```
81  //modo configuracion
82  while(configFlag == true) {
83    //reset de estados
84    hourFlag = false;
85    minuteFlag = false;
86    secondFlag = false;
87    aplicarFlag = false;
88    changeFormat = false;
89
90    //aplicar cambios
91    while (digitalRead(pinButtonConfig) == HIGH) {
92      delay(3000);
93      if (digitalRead(pinButtonConfig) == HIGH && configFlag == true) {
94        rtc.adjust(DateTime(2011, 11, 11, set_hour, set_minute, set_second));
95        Serial.println("cambios aplicados (soltar el botón para imprimir el tiempo)");
96        configFlag = false;
97      }
98    }
```

- Entrando al 'modo configuración' (la segunda parte del 'void loop()')

[84 – 88] Reset de estados de los flags que se usan dentro del modo configuración
[91 – 98] Comprobación del botón pinButtonConfig nuevamente para realizar los cambios establecidos o antes de hacer alguno

```
99    //cambiar entre formato 24hs y 12hsAM/PM
100   while (digitalRead(pinButtonFormat) == HIGH && changeFormat == false) {
101     delay(3000);
102     if (digitalRead(pinButtonFormat) == HIGH) {
103       if ( lastState == 40 ) {
104         Serial.println("formato AM/PM");
105         EEPROM.write(0, 39);
106         lastState = EEPROM.read(0);
107       }
108       else {
109         Serial.println("formato 24hs");
110         EEPROM.write(0, 40);
111         lastState = EEPROM.read(0);
112       }
113       changeFormat = true;
114     }
115   }
```

[100 – 115] Comprobación del botón ‘pinButtonFormat’: mientras se mantiene pulsado va intercalando entre el formato 24hs y el formato 12hs AM/PM cada segundos, una vez soltado el respectivo botón se establecerá el último formato impreso en la pantalla

```
116     //configurar horas
117     while (digitalRead(pinButtonH) == HIGH && hourFlag == false) {
118       delay(700);
119       if (digitalRead(pinButtonH) == HIGH) {
120         xhour ++;
121         if (xhour > 23) {
122           xhour = 0;
123         }
124         Serial.println(xhour);
125       }
126       else {
127         set_hour = xhour;
128         Serial.print("hora configurada ");
129         Serial.println(set_hour);
130       }
131       hourFlag = true;
132     }
```

[116 – 132] Comprobación del botón ‘pinButtonH’: por cada 700 mili-segundos que se mantiene pulsado va a ir imprimiendo el valor de la hora establecida anteriormente +1 hasta que se suelte el botón y se establecerá la hora indicada en la pantalla dentro de la variable ‘set_hour’. Si el valor es mayor a 23 vuelve a imprimir desde el valor 0

```
133     //configurar minutos
134     while (digitalRead(pinButtonM) == HIGH && minuteFlag == false) {
135       delay(700);
136       if (digitalRead(pinButtonM) == HIGH) {
137         xminute ++;
138         if (xminute >= 60) {
139           xminute = 0;
140         }
141         Serial.println(xminute);
142       }
143       else {
144         set_minute = xminute;
145         Serial.print("minuto configurado ");
146         Serial.println(set_minute);
147       }
148       minuteFlag = true;
149     }
```

[133 – 149] Comprobación del botón ‘pinButtonM’: por cada 700 mili-segundos que se mantiene pulsado va a ir imprimiendo el valor del minuto establecido anteriormente +1 hasta que se suelte el botón y se establecerá el minuto indicado en la pantalla dentro de la variable ‘set_minute’. Si el valor es igual o mayor de 60 vuelve a imprimir desde el valor 0

```
150     //configurar segundos
151     while (digitalRead(pinButtonS) == HIGH && secondFlag == false) {
152       delay(700);
153       if (digitalRead(pinButtonS) == HIGH) {
154         xsecond ++;
155         if (xsecond >= 60) {
156           xsecond = 0;
157         }
158         Serial.println(xsecond);
159       }
160       else {
161         set_second = xsecond;
162         Serial.print("segundo configurado ");
163         Serial.println(set_second);
164       }
165       secondFlag = true;
166     }
167   }
168 }
```

[150 – 166] Comprobación del botón ‘pinButtonM’: por cada 700 mili-segundos que se mantiene pulsado va a ir imprimiendo el valor del minuto establecido anteriormente +1 hasta que se suelte el botón y se establecerá el minuto indicado en la pantalla dentro de la variable ‘set_minute’. Si el valor es igual o mayor de 60 vuelve a imprimir desde el valor 0

```
170 void printTime() {
171   DateTime nowTime = rtc.now();
172   if ( lastState == 39 ) {    //esta en formato AM/PM
173     Serial.print(nowTime.twelveHour());
174     Serial.print(":");
175     Serial.print(nowTime.minute());
176     Serial.print(":");
177     Serial.print(nowTime.second());
178     Serial.print(" ");
179     if (nowTime.isPM() == 0){
180       Serial.println("AM");
181     }
182     else{
183       Serial.println("PM");
184     }
185   }
186   if ( lastState == 40 ) {    //esta en formato 24hs
187     Serial.print(nowTime.hour());
188     Serial.print(":");
189     Serial.print(nowTime.minute());
190     Serial.print(":");
191     Serial.println(nowTime.second());
192   }
193   //imprimir a cada segundo
194   delay(1000);
195 }
```

- Ahora podemos ver la función ‘printTime()’ utilizada en la primera parte del ‘void loop()’

[171] Esta linea toma el tiempo establecido actualmente del RTC DS3231
[172 – 185] Utilizando un if statement y gracias a las lineas [100 – 115] se puede imprimir el tiempo en formato 12hs AM/PM.

- Dentro del mismo se puede ver que gracias a la función ‘twelveHour()’ se puede imprimir la hora establecida pero en formato 12hs. Eso si, para saber si es AM o PM se utiliza la función ‘isPM()’ [186 – 194] Y por último si con anterioridad estaba establecido como formato 24hs solo se imprime la hora, minutos y segundos de forma normal; y con el agregado de un delay para que solo se imprima una vez cada segundo que pasa dentro del RTC

5. USO DE DOCUMENTACIÓN:

RTClib DS3231:
https://adafruit.github.io/RTClib/html/class_r_t_c___d_s3231.html

EEPROM.h:
https://docs.arduino.cc/learn/built-in-libraries/eeprom