// RTC Programmable Clock  
// sw1 Menu and Select

// sw2 and sw3 Change Values

// SW4 Home, back , Alarm Off

#include

#include "RTClib.h"

RTC\_DS3231 rtc;

int dot\_status = 0;

int my\_hour = 0, my\_min = 0, my\_date = 0, my\_month = 0, my\_second = 0, my\_year;

int alarm\_hour = 0, alarm\_min = 0;

const int a = 10, b = 11, c = 4, d = 3, e = 2, f = A3, g = A2;

const int dot = 9;

const int s1 = A1, s2 = A0, s3 = 13, s4 = 12;

boolean sw1\_status = 0, sw2\_status = 0, sw3\_status = 0, sw4\_status = 0;

int switch\_status = 0;

const int sw1 = 5, sw2 = 6, sw3 = 8, sw4 = 7;

int delay\_time = 3 ;

char my\_display[3] = " ";

int buzzer\_pin = 1;

int buzzer\_status = 0;

int beep\_delay = 20;

int default\_alarm\_hour = 19, default\_alarm\_min = 00;

boolean alarm\_status = 1;

int hour\_address = 0, min\_address = 1;

void setup() {

// put your setup code here, to run once:

#ifndef ESP8266

while (!Serial);

#endif

delay(100);

if (! rtc.begin()) {

Serial.println("Couldn't find RTC");

while (1);

}

DateTime now = rtc.now();

my\_year = now.year();

alarm\_hour = EEPROM.read(hour\_address);

alarm\_min = EEPROM.read(min\_address);

pinMode(dot, OUTPUT);

pinMode(a, OUTPUT);

pinMode(b, OUTPUT);

pinMode(c, OUTPUT);

pinMode(d, OUTPUT);

pinMode(e, OUTPUT);

pinMode(f, OUTPUT);

pinMode(g, OUTPUT);

pinMode(s1, OUTPUT);

pinMode(s2, OUTPUT);

pinMode(s3, OUTPUT);

pinMode(s4, OUTPUT);

pinMode(sw1, INPUT);

pinMode(sw2, INPUT);

pinMode(sw3, INPUT);

pinMode(sw4, INPUT);

pinMode(buzzer\_pin, OUTPUT);

}

void loop() {

DateTime now = rtc.now(); // get\_time\_date();

check\_switch(); // check if any switch is pressed

my\_hour = now.hour();

my\_min = now.minute();

my\_date = now.day();

my\_month = now.month();

int\_to\_string(my\_hour, my\_min); // convert the data into a string

digitalWrite(dot, (now.second()) % 2); // blink the dot for 1 sec

send\_to\_display(); // display the data to seven segments

if ((my\_min == alarm\_min ) && alarm\_status ) // Check for alarm

{

if (my\_hour == alarm\_hour )

{

if (digitalRead(sw4))

{

delay(100);

buzzer\_status = 0;

}

Serial.println("Alarm!!");

digitalWrite(buzzer\_pin, ((now.second() % 2)) && buzzer\_status);

}

}

else

{

buzzer\_status = 1;

digitalWrite(buzzer\_pin, 0);

}

}

void get\_time\_date()

{

DateTime now = rtc.now();

my\_hour = now.hour();

my\_min = now.minute();

my\_date = now.day();

my\_month = now.month();

my\_second = now.second();

}

void int\_to\_string(int i1, int i2 )

{

if (i1 < 10 && (i2 < 10))

{

sprintf(my\_display, "0%d0%d", i1, i2);

}

else if (i1 < 10 && (i2 >= 10))

{

sprintf(my\_display, "0%d%d", i1, i2);

}

else if (i1 >= 10 && (i2 < 10))

{

sprintf(my\_display, "%d0%d", i1, i2);

}

else if (i1 >= 10 && (i2 >= 10))

{

sprintf(my\_display, "%d%d", i1, i2);

}

}

void send\_to\_display()

{

segment\_1();

display\_(0); // to display 1st digit

delay(delay\_time);

low();

segment\_2();

display\_(1); // to display 2nd digit

delay(delay\_time);

low();

segment\_3();

display\_(2); // to display 3rd digit

delay(delay\_time);

low();

segment\_4();

display\_(3); // to display 4th digit

delay(delay\_time);

low();

}

void display\_(int i)

{

switch (my\_display[i])

{

case '0' :

zero();

break;

case '1' :

one();

break;

case '2' :

two();

break;

case '3':

three();

break;

case '4':

four();

break;

case '5':

five();

break;

case '6':

six();

break;

case '7':

seven();

break;

case '8':

eight();

break;

case '9':

nine();

break;

case 'p':

pattern();

break;

case 'q':

pattern\_2();

break;

case 'A':

capital\_A();

break;

case 'L':

capital\_L();

break;

case 'F':

capital\_F();

break;

case 'n':

small\_n();

break;

}

}

void segment\_1()

{

digitalWrite(s1, 0);

digitalWrite(s2, HIGH);

digitalWrite(s3, HIGH);

digitalWrite(s4, HIGH);

}

void segment\_2()

{

digitalWrite(s1, HIGH);

digitalWrite(s2, 0);

digitalWrite(s3, HIGH);

digitalWrite(s4, HIGH);

}

void segment\_3()

{

digitalWrite(s1, HIGH);

digitalWrite(s2, HIGH);

digitalWrite(s3, 0);

digitalWrite(s4, HIGH);

}

void segment\_4()

{

digitalWrite(s1, HIGH);

digitalWrite(s2, HIGH);

digitalWrite(s3, HIGH);

digitalWrite(s4, 0);

}

void all\_segment\_off()

{

digitalWrite(s1, HIGH);

digitalWrite(s2, HIGH);

digitalWrite(s3, HIGH);

digitalWrite(s4, HIGH);

}

void all\_segment\_on()

{

digitalWrite(s1, 0);

digitalWrite(s2, 0);

digitalWrite(s3, 0);

digitalWrite(s4, 0);

}

void low()

{

digitalWrite(a, 0);

digitalWrite(b, 0);

digitalWrite(c, 0);

digitalWrite(d, 0);

digitalWrite(e, 0);

digitalWrite(f, 0);

digitalWrite(g, 0);

}

void one()

{

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(a, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

digitalWrite(e, LOW);

digitalWrite(d, LOW);

}

void two()

{

digitalWrite(b, 1);

digitalWrite(c, 0);

digitalWrite(a, 1);

digitalWrite(f, 0);

digitalWrite(g, 1);

digitalWrite(e, 1);

digitalWrite(d, 1);

}

void three()

{

digitalWrite(a, 1);

digitalWrite(b, 1);

digitalWrite(c, 1);

digitalWrite(d, 1);

digitalWrite(e, 0);

digitalWrite(f, 0);

digitalWrite(g, 1);

}

void four()

{

digitalWrite(a, 0);

digitalWrite(b, 1);

digitalWrite(c, 1);

digitalWrite(d, 0);

digitalWrite(e, 0);

digitalWrite(f, 1);

digitalWrite(g, 1);

}

void five()

{

digitalWrite(a, 1);

digitalWrite(b, 0);

digitalWrite(c, 1);

digitalWrite(d, 1);

digitalWrite(e, 0);

digitalWrite(f, 1);

digitalWrite(g, 1);

}

void six()

{

digitalWrite(a, 1);

digitalWrite(b, 0);

digitalWrite(c, 1);

digitalWrite(d, 1);

digitalWrite(e, 1);

digitalWrite(f, 1);

digitalWrite(g, 1);

}

void seven()

{

digitalWrite(a, 1);

digitalWrite(b, 1);

digitalWrite(c, 1);

digitalWrite(d, 0);

digitalWrite(e, 0);

digitalWrite(f, 0);

digitalWrite(g, 0);

}

void eight()

{

digitalWrite(a, 1);

digitalWrite(b, 1);

digitalWrite(c, 1);

digitalWrite(d, 1);

digitalWrite(e, 1);

digitalWrite(f, 1);

digitalWrite(g, 1);

}

void nine()

{

digitalWrite(a, 1);

digitalWrite(b, 1);

digitalWrite(c, 1);

digitalWrite(d, 1);

digitalWrite(e, 0);

digitalWrite(f, 1);

digitalWrite(g, 1);

}

void zero()

{

digitalWrite(a, 1);

digitalWrite(b, 1);

digitalWrite(c, 1);

digitalWrite(d, 1);

digitalWrite(e, 1);

digitalWrite(f, 1);

digitalWrite(g, 0);

}

void pattern()

{

digitalWrite(a, 1);

digitalWrite(b, 1);

digitalWrite(c, 1);

digitalWrite(d, 0);

digitalWrite(e, 0);

digitalWrite(f, 0);

digitalWrite(g, 0);

}

void pattern\_2()

{

digitalWrite(a, 0);

digitalWrite(b, 0);

digitalWrite(c, 0);

digitalWrite(d, 1);

digitalWrite(e, 1);

digitalWrite(f, 1);

digitalWrite(g, 0);

}

void capital\_A()

{

digitalWrite(b, HIGH);

digitalWrite(c, LOW);

digitalWrite(a, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(e, HIGH);

digitalWrite(d, HIGH);

}

void capital\_L()

{

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(a, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, LOW);

digitalWrite(e, HIGH);

digitalWrite(d, HIGH);

}

void capital\_F()

{

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(a, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(e, HIGH);

digitalWrite(d, LOW);

}

void small\_n()

{

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(a, LOW);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

digitalWrite(e, HIGH);

digitalWrite(d, LOW);

}

void check\_switch()

{

switch\_status = read\_switch\_status();

if (switch\_status)

{

if (sw4\_status)

{

display\_alarm();

}

if (sw1\_status)

{

beep();

delay(300);

menu(); // go to menu if switch one is pressed

delay(200);

}

else if (sw2\_status)

{

display\_date(); // display date if switch 2 is pressed

}

else if (sw3\_status)

{

display\_temperature(); // display tempreature if switch 3 is pressed

}

}

}

int display\_on()

{

long t1 = 0;

t1 = millis();

my\_display[0]='A';

my\_display[1]='L';

my\_display[2]='0';

my\_display[3]='n';

while (millis() - t1 >= 800)

{

send\_to\_display();

}

}

int display\_off()

{

long t1 = 0;

t1 = millis();

my\_display[0]='A';

my\_display[1]='0';

my\_display[2]='F';

my\_display[3]='F';

while (millis() - t1 >= 800)

{

send\_to\_display();

}

}

boolean read\_switch\_status()

{

sw1\_status = digitalRead(sw1);

sw2\_status = digitalRead(sw2);

sw3\_status = digitalRead(sw3);

sw4\_status = digitalRead(sw4);

return (sw1\_status || sw2\_status || sw3\_status || sw4\_status);

}

void display\_temperature()

{

Serial.println("Display Temperature");

float temperature = 0;

int x1 = 0, x2 = 0;

long t = 0;

temperature = rtc.getTemperature();

x1 = int(temperature);

x2 = (temperature - x1) \* 100;

Serial.println(temperature);

int\_to\_string(x1, x2);

t = millis();

while ((millis() - t) <= 2000) // display temperature for 2 seconds

{

send\_to\_display();

digitalWrite(dot, HIGH);

}

}

void display\_alarm()

{

Serial.println("Display Alarm");

long t = 0;

int\_to\_string(alarm\_hour, alarm\_min);

t = millis();

while ((millis() - t) <= 2000) // display Alarm time for 2 seconds

{

send\_to\_display();

digitalWrite(dot, HIGH);

}

}

void display\_date()

{

long t = 0;

int\_to\_string(my\_date, my\_month);

t = millis();

while ((millis() - t) <= 1500) // display date for 1.5 seconds

{

send\_to\_display();

digitalWrite(dot, HIGH);

}

int\_to\_string(int(my\_year / 100), (my\_year - 2000)); // divide 4 digits into two seperate two digits

t = millis();

while ((millis() - t) <= 1000) // display year for 0.5 seconds

{

send\_to\_display();

digitalWrite(dot, LOW);

}

}

void menu()

{

Serial.println("entered menu");

delay(100);

while (!read\_switch\_status())

{

digitalWrite(dot, 0);

DateTime now = rtc.now();

if (now.second() % 2)

{

display\_pattern\_menu\_2(); // display an animation when entered into menu

}

else

{

display\_pattern\_menu();

}

}

if (sw1\_status)

{

// Set the clock parameters

delay(100);

if (set\_time())

{

return 1;

}

else if (set\_date())

{

return 1;

}

else if (set\_year())

{

return 1;

}

else if (set\_alarm())

{

return 1;

}

}

else

{

beep();

}

}

int set\_time()

{

int time\_chandged\_status = 0;

beep();

delay(100);

read\_switch\_status();

delay(100);

while (!sw1\_status)

{

read\_switch\_status();

if (sw2\_status)

{

time\_chandged\_status = 1;

delay(150);

my\_hour++;

if (my\_hour > 23)

{

my\_hour = 0;

}

}

else if (sw3\_status)

{

time\_chandged\_status = 1;

delay(100);

my\_min++;

if (my\_min > 59)

{

my\_min = 0;

}

}

else if (sw4\_status)

{

beep();

return 1;

}

int\_to\_string(my\_hour, my\_min);

send\_to\_display();

digitalWrite(dot, HIGH);

}

if (time\_chandged\_status)

{

DateTime now = rtc.now();

my\_date = now.day(); // get the current date

my\_month = now.month();

//now set the current date as it is and set updated time

rtc.adjust(DateTime(my\_year, my\_month, my\_date, my\_hour, my\_min, 0));;

beep();

delay(50);

}

get\_time\_date();

delay(200);

return 0;

}

int set\_date()

{

int date\_chandged\_status = 0;

read\_switch\_status();

beep();

delay(100);

while (!sw1\_status)

{

read\_switch\_status();

if (sw2\_status)

{

date\_chandged\_status = 1;

delay(150);

my\_date++;

if (my\_date > 31)

{

my\_date = 0;

}

}

else if (sw3\_status)

{

date\_chandged\_status = 1;

delay(150);

my\_month++;

if (my\_month > 12)

{

my\_month = 0;

}

}

else if (sw4\_status)

{

beep();

return 1;

}

int\_to\_string(my\_date, my\_month);

send\_to\_display();

digitalWrite(dot, HIGH);

}

if (date\_chandged\_status)

{

DateTime now = rtc.now();

my\_hour = now.hour(); // get the current time

my\_min = now.minute();

rtc.adjust(DateTime(my\_year, my\_month, my\_date, my\_hour, my\_min, 0));

beep();

delay(50);

}

get\_time\_date();

delay(200);

return 0;

}

int set\_year()

{

int year\_chandged\_status = 0;

read\_switch\_status();

beep();

while (!sw1\_status)

{

read\_switch\_status();

if (sw2\_status)

{

year\_chandged\_status = 1;

delay(150);

my\_year--;

if (my\_year < 2000 )

{

my\_year = 2000; // Constrain the year lower limit to 2000

}

}

else if (sw3\_status)

{

year\_chandged\_status = 1;

delay(150);

my\_year++;

if (my\_year > 2070) // Constrain the year upper limit to 2070

{

my\_year = 2070;

}

}

else if (sw4\_status)

{

beep();

return 1;

}

int\_to\_string(int(my\_year / 100), (my\_year - 2000));

send\_to\_display();

digitalWrite(dot, LOW);

}

if (year\_chandged\_status)

{

DateTime now = rtc.now();

my\_hour = now.hour(); // get the current time

my\_min = now.minute();

my\_date = now.day(); // get the current date

my\_month = now.month();

//now set the current time and date as it is and set updated year

rtc.adjust(DateTime(my\_year, my\_month, my\_date, my\_hour, my\_min, 0));

beep();

delay(50);

}

get\_time\_date();

delay(200);

return 0;

}

int set\_alarm()

{

int alarm\_chandged\_status = 0;

read\_switch\_status();

beep();

while (!sw1\_status)

{

read\_switch\_status();

if (sw2\_status)

{

alarm\_chandged\_status = 1;

delay(150);

alarm\_hour++;

if (alarm\_hour > 23 )

{

alarm\_hour = 0;

}

}

else if (sw3\_status)

{

alarm\_chandged\_status = 1;

delay(150);

alarm\_min++;

if (alarm\_min > 59)

{

alarm\_min = 0;

}

}

else if (sw4\_status)

{

beep();

return 1;

}

int\_to\_string(alarm\_hour, alarm\_min);

send\_to\_display();

digitalWrite(dot, HIGH);

}

EEPROM.write(hour\_address, alarm\_hour); // Set the Alarm parameters to EEPROM of ATmega328p

EEPROM.write(min\_address, alarm\_min);

get\_time\_date();

beep();

delay(200);

return 0;

}

void display\_pattern\_menu()

{

my\_display[0] = 'q';

my\_display[1] = 'p';

my\_display[2] = 'q';

my\_display[3] = 'p';

send\_to\_display();

}

void display\_pattern\_menu\_2()

{

my\_display[0] = 'p';

my\_display[1] = 'q';

my\_display[2] = 'p';

my\_display[3] = 'q';

send\_to\_display();

}

void beep()

{

digitalWrite(buzzer\_pin, HIGH);

delay(beep\_delay);

digitalWrite(buzzer\_pin, LOW);

}