TM1637 4-Digit 7-Segment Display with Arduino. (Digital Clock example)

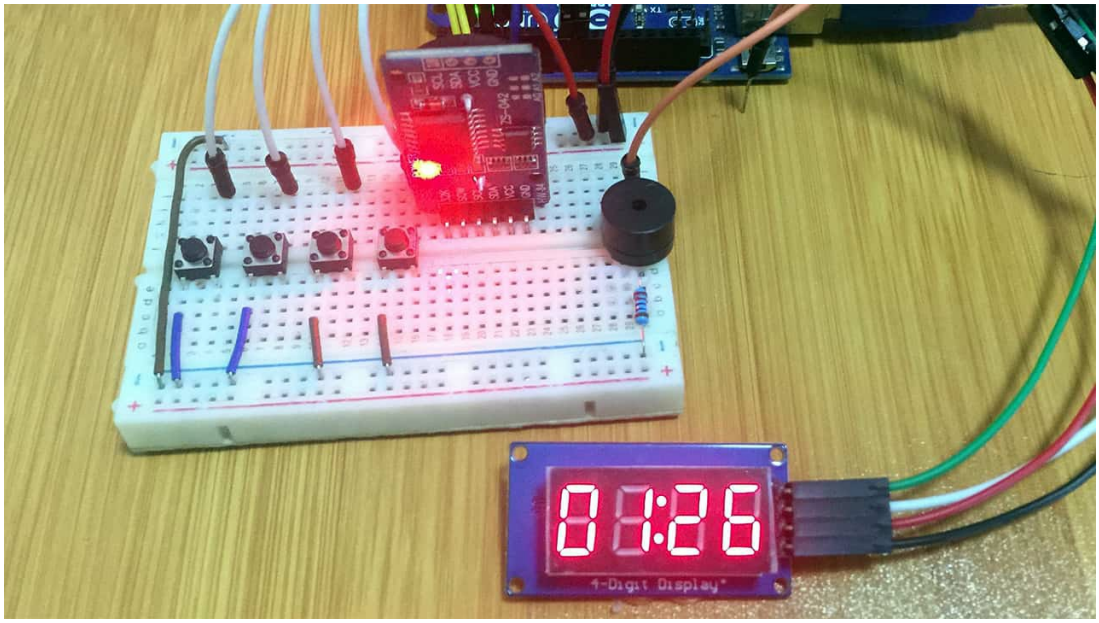


Table of Contents

4 digit 7-segment displays are among the commonest display components in several of our household appliances like digital radios, clocks, refrigerators, air conditioners and many others. They are used to display information in form of numbers and some letters. This data can be sensor readings like on cookers and coolers or time in digital clocks.

The TM1637 4 digit 7 segment display is among these displays and in this tutorial I would like to show how this display can be interfaced with Arduino in a number of applications.

TM1637 4 Digit 7 Segment Display Hardware Overview.

4 digit 7 segment displays are simply an arrangement of four seven segment display modules that use the multiplexing technique so that they can be used on microcontrollers with a limited amount of

Popular Posts


DIY Lithium-ion 18650 Battery Charger using TP4056 Module.


How to Use a Digital Multimeter.


L293D Motor Driver Shield for Arduino


Interfacing 5V 4-Channel Relay Module with Arduino


DC motor control using L298N motor driver and Arduino


Obstacle Avoiding Robot car using Arduino

I/O pins. I have explained in detail how the seven segment displays work and how they can be used with Arduino in a previous post that you can check out using the link below.

- How to use 7-segment Display with Arduino.

The common 4 digit displays usually have twelve pins used to communicate with the microcontroller using multiplexing. However this is a big disadvantage because in case a project requires connecting other components to the microcontroller then the number of I/O pins will not be enough.

The TM1637 4 digit 7 segment display module solves the above problem by reducing the number of pins needed to control the four displays using only four pins instead of twelve! This module mainly uses only the CLK and DIO pins for serial communication with a microcontroller.

Pinout of the TM1637 4 digit 7 segment display module.

This display module has only four pins which are:
CLK(Clock): It is used for data input and output at rising edge.
DIO(Data Input/ Output): for serial data input or output
VCC: connected to 5V power supply. The module can work on +5V regulated power and higher voltages may lead to permanent damage.
GND: connected to the ground.

[amazon box="B01DKISMXK"]

Why use the TM1637 Display Module?

Apart from the reduced number of pins used by this display module to connect to microcontrollers, other characteristics that make it preferred in a number of applications include;1 .It is relatively cheap compared to other display modules. You can buy one for less than $2.
2 .The TM1637 display module has an operating current consumption of about 80mA and can operate on both 3.3v and 5V. This makes it suitable for use in low power electronics circuits.
3 .It is very potable and durable therefore can withstand even adverse environmental conditions for a long period of time.
4 .Several pre-written programs for controlling this display module are available for free which makes its use very simple even for non-programmers.

Connecting the TM1637 Display Module to Arduino.

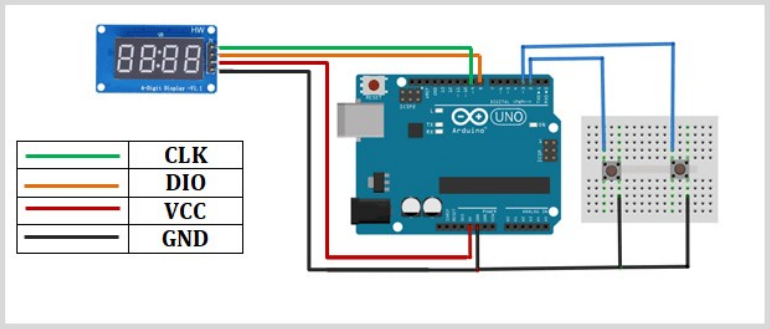
Using this display module with Arduino is very simple because you only need to connect the CLK and DIO pins to any of the digital pins, the VCC is connected to 5V and the GND to ground of the Arduino.

There are several libraries that you can use to simplify the work of programming this display however I recommend the TM1637Display.h library which has a number of functions for controlling how digits and some letters are displayed. These functions are going to be described using various examples.

Download the TM1637Display library : TM1637Display Library

i).Up-Down Counter

In this example we use the TM1637 display module to make a simple counter with two push buttons. The setup is as shown below where the buttons are connected to pin 2 and 3 of the and the CLK and DIO pins are connected to pin 8 and 9 of Arduino respectively.



Code for running the counter.

```
#include <TM1637Display.h>

#define UP 2
#define DOWN 3

const int CLK = 8; //Set the CLK pin connection to the display
const int DIO = 9; //Set the DIO pin connection to the display

const uint8_t blank[] = {0x00, 0x00, 0x00,0x00};

TM1637Display display(CLK, DIO); //set up the 4-Digit Display.

int num = 0;

void setup()
{
    pinMode(UP, INPUT_PULLUP);
    pinMode(DOWN,
INPUT_PULLUP);
    display.setBrightness(0x0a); //set the diplay to maximum brightness
    display.setSegments(blank);//clear display
}
```

Description of code.

This example shows the commonest basic functions of the TM1637Display.h library which are; `setBrightness()`: for setting the brightness of the display. There are seven levels of brightness therefore the value can be in 0x0a to 0x0f.

`showNumberDec()`: this is the most used function and it is for displaying decimal number digits. This function can take four arguments that is, showNumberDec (int num, bool leading_zero, uint8_t length, uint8_t pos), the first argument is the number to be displayed, the second is determining leading zeros and can either be true or false, the third argument is the number of digits to be displayed and the last argument is for selecting the position where the number to be displayed should begin from.

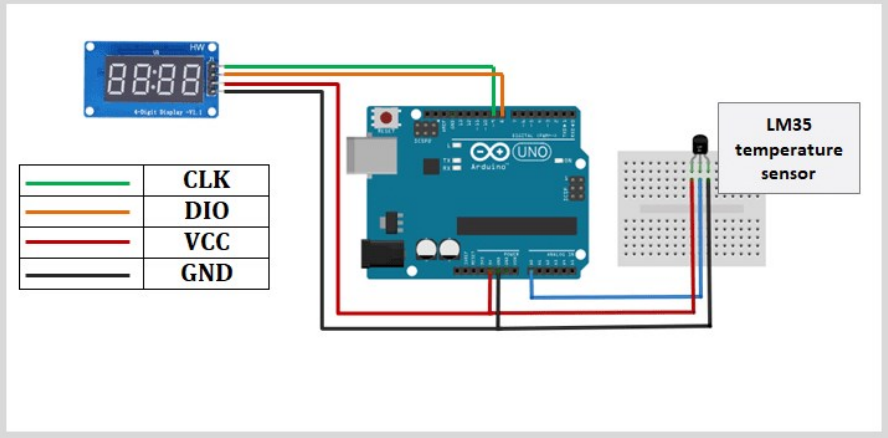
For example `display.showNumberDec(25,false,2,1)` displays the number 25 without leading zeros on the second and third digit of.

ii). Displaying Temperature from the LM35 Temperature sensor.

The main objective of this example is to show how to create custom characters on the display. In this case we need to create a “degree” symbol and letters C and F for displaying temperature in degrees Celsius and Fahrenheit.

Here we are going to connect the LM35 temperature sensor signal pin to analog pin A0. You can make reference to a previous tutorial to learn how this sensor works with Arduino using this link:

- LM35 Temperature Sensor with Arduino.



Code for displaying Temperature on the TM1637 Display.

```
#include <TM1637Display.h>

const int CLK = 9; //Set the CLK pin connection to the display
const int DIO = 8; //Set the DIO pin connection to the display

const uint8_t blank[] = {0x00, 0x00, 0x00,0x00};

const int tempPin = A0;

// Create degree Celsius symbol:
const uint8_t celsius[] = {
  SEG_A | SEG_B | SEG_F | SEG_G,  // Circle
  SEG_A | SEG_D | SEG_E | SEG_F   // C
};

// Create degree Fahrenheit symbol:
const uint8_t fahrenheit[]
= {
  SEG_A | SEG_B | SEG_F | SEG_G,  // Circle
  SEG_A | SEG_E | SEG_F | SEG_G   // F
};
```

To create a “degree” symbol and letters C and F we need to form arrays representing the activated LEDs of an individual 7 segment display showing the required characters. For example the array below is for ºC ;

```
const uint8_t celsius[] = {
SEG_A | SEG_B | SEG_F | SEG_G, // superscript o
SEG_A | SEG_D | SEG_E | SEG_F // C
};
```

The degree symbol “º” is created by activating segments a, b ,f and g and the letter “C” is created by turning on segments a, d, e and f.

To display the created characters we use the `setSegments()` function which passes an array of values for individual segments to the display.

iii).TM1637 4-digit 7-segment display with DS3231 RTC

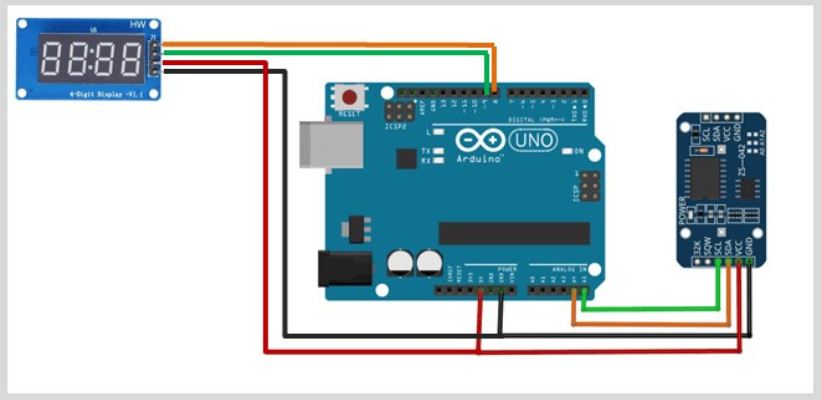
Most electronics hobbyists are fascinated with creation of digital clocks and as such this tutorial would not be complete if I don't make one using this display. The digital clock I am going to make is based on the DS3231 RTC module.

DS3231 is a Real Time Clock which is able to keep the time and date even when the power supply to an attached circuit is off. This is possible due the presence of a backup CR2032 battery to run the module in absence of external power. This module also includes a temperature sensor.

You can have a more detailed look at how this module works with Arduino from the link below;

- DS3231RTC Interfacing with Arduino.

The setup of the clock with Arduino and the TM1637 display module is as shown below.



Code for digital clock using DS3232RTC and TM1637 Display.

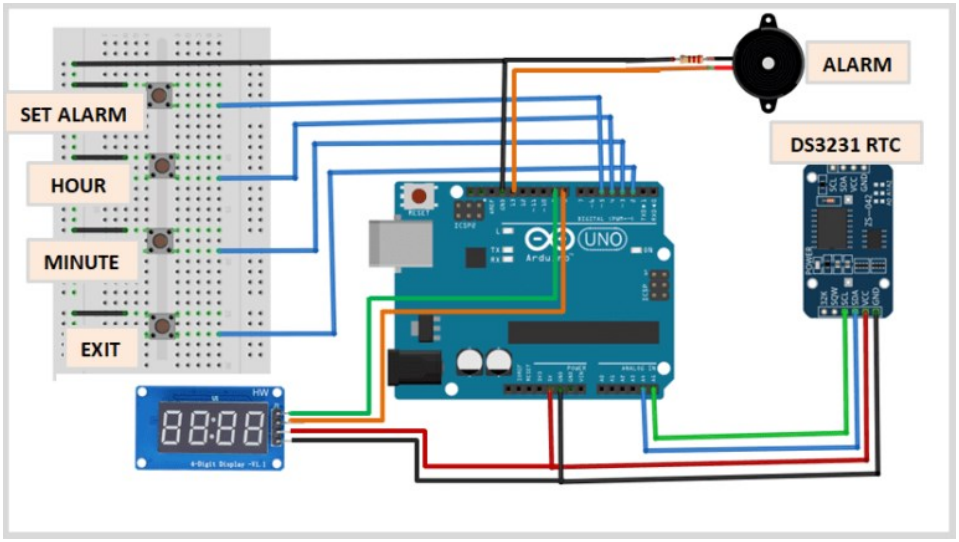
```
#include "RTClib.h"
#include <TM1637Display.h>
// Define the connections pins:
#define CLK 9
#define DIO 8
// Create rtc and display object:
RTC_DS3231 rtc;

TM1637Display display = TM1637Display(CLK, DIO);
const uint8_t blank[] = {0x00, 0x00, 0x00,0x00};

void setup() {
  // Begin serial communication at a baud rate of 9600:
  Serial.begin(9600);
  // Wait for console opening:
  delay(3000);
  // Check if RTC is
connected correctly:
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
```

Alarm clock using the TM1637 Display and DS3231 RTC

We can now modify the clock given in the example above so that it can have more functions. I will add four push buttons for adjusting the time and a buzzer that is going to act as an alarm. The setup is as shown below.



Code for controlling the alarm clock.

```
#include <Wire.h>
#include "RTClib.h"
#include <TM1637Display.h>

RTC_DS3231 rtc;

#define CLK 8
#define DIO 9

TM1637Display display(CLK, DIO);

const uint8_t blank[] = {0x00, 0x00, 0x00,0x00};

int setButton = 2; // pushbutton for setting alarm
int hourButton = 3; // pushbutton for hour
int minButton = 4; // pushbutton for minutes
int exitButton = 5; // pushbutton for exit of set alarm
int buzzer = 13;

int t, a, Hour, Min, h, m;
int set_time, alarm_time, auto_alarmStop, set_alarm_min, set_alarm_hour;
```

Working of the alarm clock.

From the code above, to set the alarm time we press the SET ALARM button which makes the clock enter the alarm setting mode. Then we can adjust the hour and minutes for the required alarm time using the HOUR and MINUTE buttons respectively.

The EXIT button is pressed after setting the alarm time so that the clock goes back to its normal operating mode. When the time set as alarm time reaches, the buzzer will turn on sounding an alarm that can be stopped manually by pressing the EXIT button or if not interrupted the alarm will keep on sounding for one minute and then goes off.

Providing more control of the alarm clock.

The code can be modified further by enabling the use of Arduino EEPROM so that we can be able to set the time of the DS3231RTC manually and even set the alarm time. This code will then be written as shown below;

```
#include <Wire.h>
#include "RTClib.h"
#include <TM1637Display.h>
#include <EEPROM.h>

RTC_DS3231 rtc;

#define CLK 8
#define DIO 9
#define pinjam 2
#define pinalarm 3
#define pinenter 4
#define pinmati 5
#define buzzer 13
```

```
TM1637Display display(CLK, DIO);


int state, a, t, tj, ta, te, tm ;
int setjam, setalarm, enter, mati;
```

Working of the alarm clock using above code.

The working of this clock is a bit confusing since it involves pressing more than one button to enter the desired mode. Let me explain how to use this clock in detail below.

Adjusting time.

To adjust the time of the RTC you press the SET ALARM and MINUTE buttons simultaneously so that the clock enters the time setting mode. In this mode you can adjust the time using the HOUR and SET ALARM buttons.

After adjusting the time press the MINUTE button and the set time will now be the current time stored in the DS3231RTC and also shown by the TM1637 display.

Setting the Alarm

To set the alarm you press the HOUR and MINUTE buttons simultaneously so that the clock enters the alarm setting mode. In this mode, you can set the alarm time using the HOUR and SET ALARM buttons.

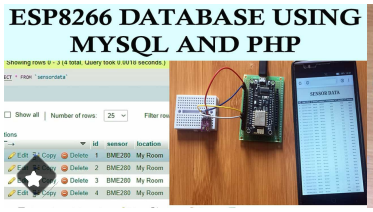
After setting the alarm press the MINUTE button and the set alarm will be stored in the EEPROM and the clock will go back to its normal operating mode. Just like in the first case, when the time set as alarm time reaches, the buzzer will turn on sounding an alarm that can be stopped manually by pressing the EXIT button or if not interrupted the alarm will keep on sounding for one minute and then goes off.

If you are in either the time or alarm setting mode, the EXIT button can also be used in case you want to go back to the normal mode without setting the time or alarm.

PREVIOUS STORY
DS3231 RTC with Arduino. (Including Digital Clock using OLED Display)

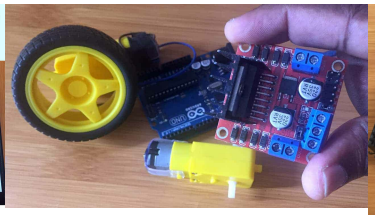
NEXT STORY
2.4″ ILI9341 TFT Touch Screen with Arduino.

YOU MAY ALSO LIKE...



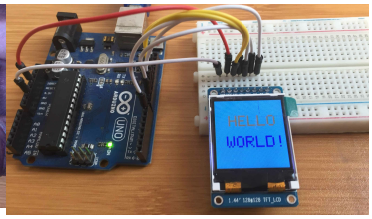
How to Connect ESP8266 TO MYSQL Database Using PHP and Arduino IDE.

JULY 7, 2021



DC motor control using L298N motor driver and Arduino

JULY 13, 2021



Using the 1.44″ TFT ST7735 Color display with Arduino.

JULY 10, 2021

Recent Posts

Smart Car Parking System using Arduino, ESP8266 NodeMCU and Blynk server.

DIY Lithium-ion 18650 Battery Charger using TP4056 Module.

How to use PCA9685 16-Channel 12-Bit PWM Servo Driver with Arduino.

How to Use a Digital Multimeter.

ESP8266 Nodemcu Webserver using Arduino IDE (Example of Controlling LED over WiFi)

How to Choose the Right RAM Upgrade for your Laptop or Desktop PC.

How to Extend C Drive using Windows Disk Management.

How to Delete Recovery Partition Using ‘Diskpart’ in Windows 10.

CATEGORIES

- 8051 Microcontroller
- Arduino
- Blog
- Computer Hardware
- Displays
- ESP8266
- Popular
- Software
- Windows 10 Support