Project Report

Real-Time Clock (RTC) Display Using Arduino DS1307, and TM1637

implementation
by
Nandan Sahu
November 2024

Table of Contents

1. Introduction

1.1 Project Overview

This project involves creating a real-time clock (RTC) display using an **Arduino Uno**, **DS1307 RTC module**, and **TM1637 seven-segment display**. The DS1307 RTC module keeps track of the current time, even when the power is off, by using a backup battery. The Arduino retrieves this time data from the RTC module and displays it on the TM1637 display in **HH** format. This project demonstrates the principles of timekeeping and data display in embedded systems, making it ideal for applications in clocks and other time-based systems.

1.2 Purpose and Motivation

In many embedded systems, maintaining accurate time is crucial, especially in systems that rely on scheduled tasks, logs, or events. This project aims to build a simple, reliable time display system, illustrating the use of an RTC module with a seven-segment display. This is an ideal project for hobbyists, students, and educators, as it offers hands-on experience with I2C communication, RTC modules, and display control with Arduino. The motivation behind this project is to create a visually simple and effective clock display that remains accurate without requiring regular adjustments.

1.3 Applications

The project has several potential applications, including:

- **Digital Clocks**: A straightforward way to display time in a digital format.
- **Home Automation**: Time-based triggers for controlling lights, alarms, or other devices.
- **Educational Purposes**: A practical project for learning I2C communication, RTC functionality, and seven-segment display usage.
- **Data Logging Systems**: Useful in data logging projects where timestamps are required, such as environmental monitoring or event tracking.

2. Objectives

The primary objectives of this project are as follows:

- **To design a real-time clock display** that accurately shows the current time in HH
  format on a seven-segment display.
- **To interface the DS1307 RTC module with Arduino** for real-time clock data retrieval and ensure it maintains accurate time even during power loss.
- **To control the TM1637 seven-segment display** using Arduino, displaying time information in a readable and user-friendly manner.
- **To implement I2C communication** between the Arduino and DS1307 RTC module for efficient data transfer.
- **To develop coding skills for Arduino** by writing a program that reads time from the RTC and updates the display.
- **To explore real-world applications** of RTC modules and digital time displays in embedded systems.

These objectives aim to build practical skills in embedded systems, RTC module usage, and display control with Arduino.

3. Components Required

3.1 Arduino Uno

The Arduino Uno microcontroller board acts as the main control unit, managing data flow between the RTC module and the display, and running the code to read and update time continuously.

3.2 DS1307 RTC Module

The DS1307 Real-Time Clock (RTC) module is used to maintain accurate time. It includes an onboard battery backup, allowing it to keep track of time even when power is off. It communicates with the Arduino using the I2C protocol.

3.3 TM1637 4-Digit Seven-Segment Display

The TM1637 display module consists of four seven-segment displays that can display hours and minutes in HH format. It is compact, easy to use, and ideal for displaying numeric data, making it suitable for a clock project.

3.4 CR2032 Battery

The CR2032 coin cell battery powers the DS1307 RTC module, allowing it to retain time data even when the Arduino is powered off.

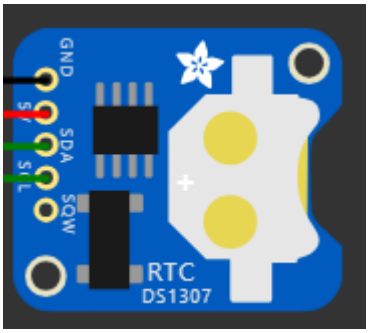
3.5 Connecting Wires

Jumper wires are used to connect the components (Arduino, RTC module, and seven-segment display) on a breadboard or directly.

4. Technical Background

4.1 DS1307 RTC Module

The **DS1307 Real-Time Clock (RTC) module** is a widely used time-keeping device in embedded systems. It is an I2C-based RTC that can keep track of seconds, minutes, hours, days, date, month, and year. It includes a 56-byte battery-backed SRAM and operates on a backup power source, allowing it to keep time even when the main power supply is disconnected. The DS1307 uses a CR2032 battery to maintain its internal clock when not powered by the main circuit, making it suitable for applications that require time continuity. Communication with the DS1307 is carried out using the I2C protocol, making it easy to connect to the Arduino.



DS1307 RTC Module

4.2 TM1637 4-Digit Seven-Segment Display

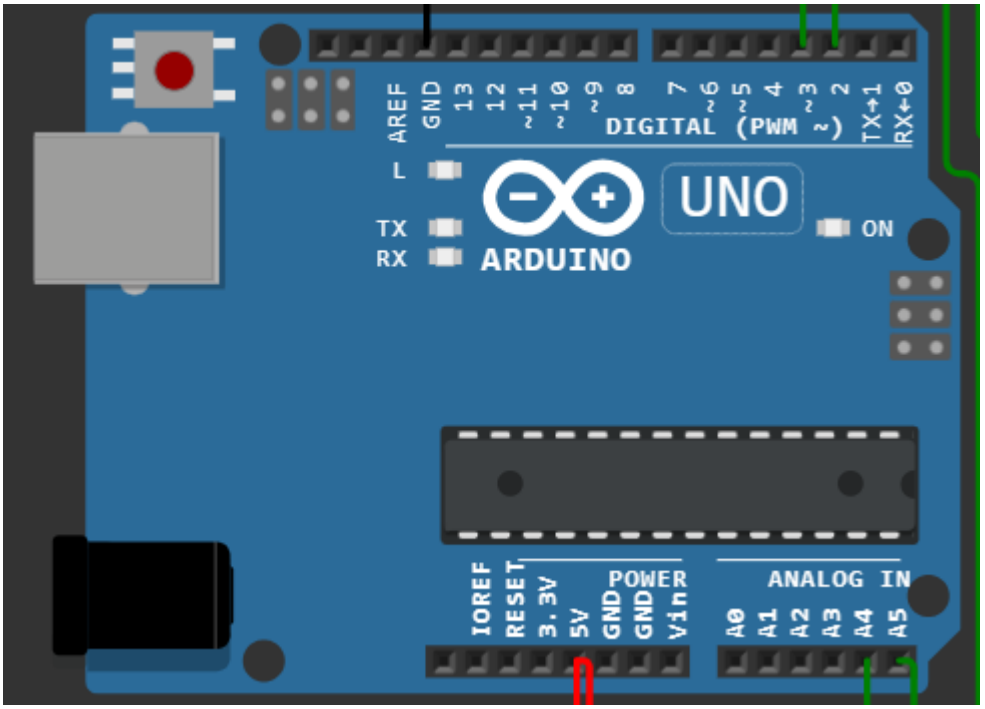
The **TM1637** is a four-digit, seven-segment display module that allows users to display numeric data in a compact and readable format. It supports two-wire communication, making it ideal for use with microcontrollers like the Arduino. Each digit can display numbers from 0 to 9 and some additional characters, allowing for an HH

time format. The TM1637 display is controlled by two pins: a clock (CLK) and data (DIO) pin. This display is commonly used for clocks, counters, and any projects requiring simple numeric output.

TM1637

4.3 Arduino Uno

The **Arduino Uno** is a popular microcontroller board based on the ATmega328P processor. It has 14 digital input/output pins, of which 6 can be used as PWM outputs, and 6 analog input pins. For this project, the Arduino Uno acts as the main processing unit that fetches time data from the DS1307 RTC and displays it on the TM1637. It communicates with the DS1307 using the I2C protocol (SDA and SCL pins) and with the TM1637 display using digital pins for clock and data.



Arduino Uno

4.4 I2C Protocol

I2C (Inter-Integrated Circuit) is a communication protocol used for connecting low-speed devices. It uses two bidirectional lines: **SDA**

(Serial Data) and **SCL (Serial Clock)**. In this project, the DS1307 RTC module communicates with the Arduino via I2C, allowing the transfer of time data between the two devices. The I2C bus allows multiple devices to communicate using only two lines, making it ideal for small projects with limited wiring.

5. Circuit Design

5.1 Wiring Diagram

The circuit design for this project involves connecting the **DS1307 RTC module** and **TM1637 seven-segment display** to the Arduino Uno. Here's how each component is connected:

- **DS1307 RTC Module**
  - **SDA** (Data Line) → **A4** on Arduino
  - **SCL** (Clock Line) → **A5** on Arduino
  - **VCC** → **5V** on Arduino
  - **GND** → **GND** on Arduino
- **TM1637 4-Digit Seven-Segment Display**
  - **DIO (Data)** → Digital **Pin 2** on Arduino
  - **CLK (Clock)** → Digital **Pin 3** on Arduino
  - **VCC** → **5V** on Arduino
  - **GND** → **GND** on Arduino

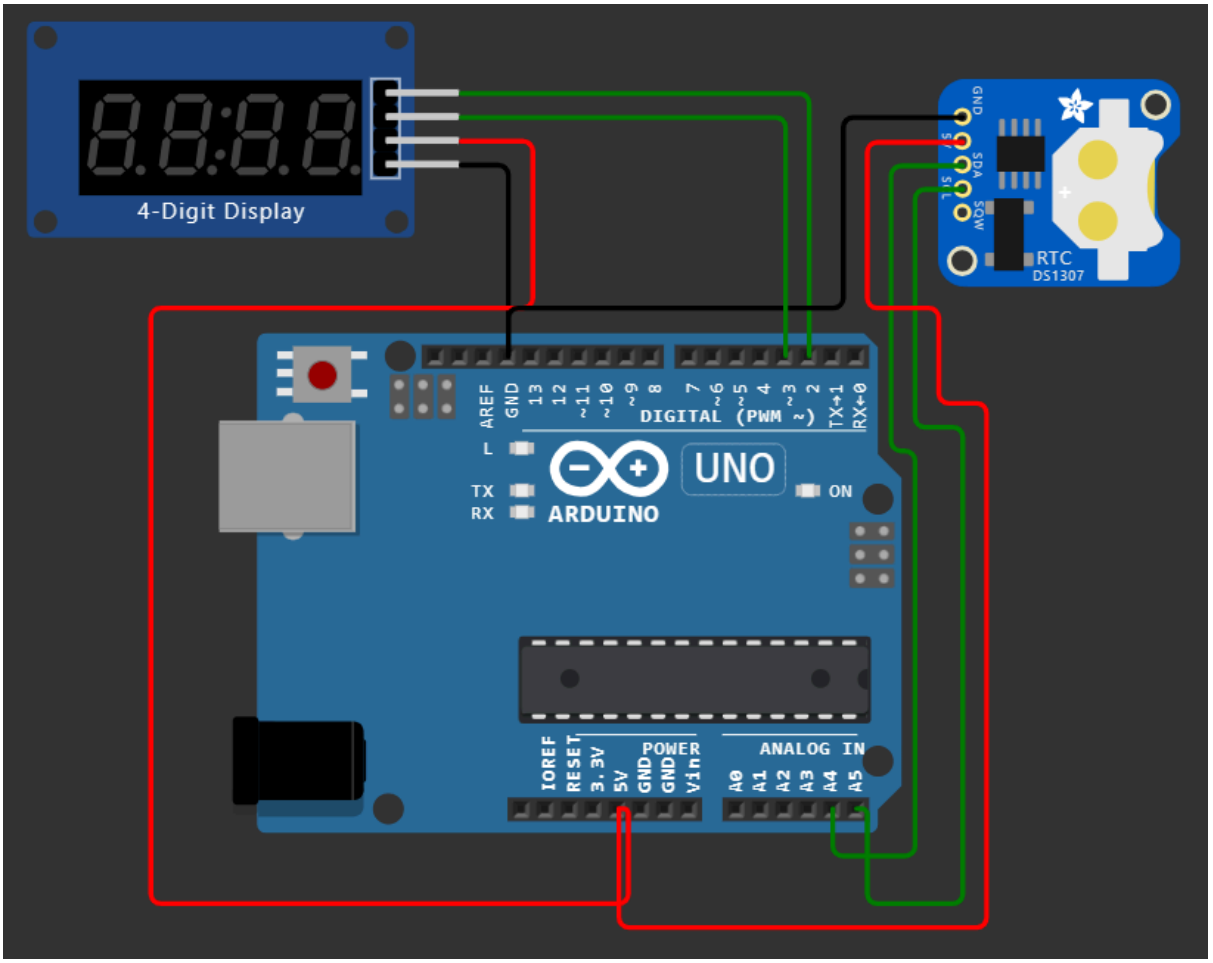
5.2 Circuit Diagram Explanation

The Arduino Uno serves as the central processing unit that connects to the DS1307 RTC and TM1637 display. The DS1307 RTC module uses the I2C communication protocol, with the SDA (data) and SCL (clock) lines connected to the corresponding A4 and A5 pins on the Arduino. This setup allows the Arduino to retrieve real-time data from the RTC module.

The TM1637 display is connected to the Arduino through digital pins 2 (DIO) and 3 (CLK), enabling simple two-wire communication. The Arduino continuously retrieves the time from the DS1307 and sends it to the TM1637 display to update the HH

format in real-time. The **5V** power and **GND** connections are shared between the Arduino, DS1307, and TM1637, creating a common ground for the circuit.

This circuit design allows for efficient data transfer and display of time in a compact setup, making it suitable for real-time clock applications.



Circuit Diagram

6. Code Implementation

The following Arduino code retrieves the current time from the DS1307 RTC module and displays it on the TM1637 4-digit, 7-segment display. This code initializes the RTC module, sets up the display, and continuously updates the display with the current time in a 12-hour format (HH) with a colon between the hour and minute.

6.1 Code Explanation

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

#define CLK 2  // Clock pin for TM1637
#define DIO 3  // Data pin for TM1637

RTC_DS1307 rtc;  // Create an RTC object
TM1637Display display(CLK, DIO);  // Create a display object
```

- **Wire.h** and **RTClib.h**: Libraries required for I2C communication and interacting with the DS1307 RTC module.
- **TM1637Display.h**: Library to control the TM1637 display.
- **CLK** and **DIO**: Define the clock and data pins for the TM1637 display.
- **RTC_DS1307 rtc**: Create an RTC object to communicate with the DS1307.
- **TM1637Display display(CLK, DIO)**: Create a display object for the TM1637 module.

```
void setup() {
  Serial.begin(9600);  // Initialize serial communication
  display.setBrightness(0x0f);  // Set display brightness (0x00 to 0x0f)

  if (!rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);  // Halt if RTC is not found
```

```
  }

  if (!rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
}
```

- **Serial.begin(9600)**: Initializes serial communication for debugging purposes.
- **display.setBrightness(0x0f)**: Sets the display brightness to the maximum (0x0f).
- **rtc.begin()**: Initializes the RTC. If not found, a message is displayed, and the program halts.
- **rtc.isrunning()**: Checks if the RTC is running. If not, it sets the RTC to the current date and time of the computer (at the time of upload).

```
void loop() {
  DateTime now = rtc.now();  // Get the current date and time
  // Convert to 12-hour format
  int hour = now.hour();
  int minute = now.minute();

  if (hour == 0) {
    hour = 12;  // Midnight case
  } else if (hour > 12) {
    hour -= 12;  // Convert to 12-hour format
  }
  // Create a 4-digit number (HHMM) for display
  int displayTime = hour * 100 + minute;  // Format: HHMM

  display.showNumberDecEx(displayTime, 0b01000000, true);  // Display
time with colon
  // 0b01000000 lights up the colon (dots) between hours and minutes
  delay(1000);  // Update every second
}
```

- **DateTime now = rtc.now()**: Retrieves the current time from the RTC module.
- **int hour = now.hour(); int minute = now.minute();**: Stores the current hour and minute.
- **12-Hour Format Conversion**: Converts the hour into 12-hour format for a user-friendly display. Midnight (0) is displayed as 12, and any hour greater than 12 is adjusted.
- **int displayTime = hour * 100 + minute;**: Formats the time as a 4-digit number (HHMM) to display.
- **display.showNumberDecEx(displayTime, 0b01000000, true);**: Displays the formatted time on the TM1637 with a colon lit between the hour and minute digits.
- **delay(1000)**: Updates the time display every second.

6.2 Code

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

#define CLK 2  // Clock pin for TM1637
#define DIO 3  // Data pin for TM1637

RTC_DS1307 rtc;  // Create an RTC object
TM1637Display display(CLK, DIO);  // Create a display object

void setup() {
  Serial.begin(9600);  // Initialize serial communication
  display.setBrightness(0x0f);  // Set display brightness (0x00 to 0x0f)

  if (!rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);  // Halt if RTC is not found
  }

  if (!rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
```

```
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
}

void loop() {
  DateTime now = rtc.now();  // Get the current date and time

  // Convert to 12-hour format
  int hour = now.hour();
  int minute = now.minute();

  if (hour == 0) {
    hour = 12;  // Midnight case
  } else if (hour > 12) {
    hour -= 12;  // Convert to 12-hour format
  }

  // Create a 4-digit number (HHMM) for display
  int displayTime = hour * 100 + minute;  // Format: HHMM

  display.showNumberDecEx(displayTime, 0b01000000, true);  // Display
time with colon
  // 0b01000000 lights up the colon (dots) between hours and minutes

  delay(1000);  // Update every second
}
```

7. Results and Observations

After setting up the project and uploading the code, the TM1637 4-digit, 7-segment display successfully shows the current time in a 12-hour format with a colon between the hours and minutes. The time is updated every second, giving a real-time display effect. The following observations were noted during testing:

1. **Display Brightness**: The brightness setting of the TM1637 was set to the maximum (0x0f) to ensure clarity, especially in bright environments.
2. **12-Hour Format**: The code successfully converts the time from 24-hour to 12-hour format, displaying "12" for midnight and noon.
3. **Real-Time Clock Accuracy**: The DS1307 RTC maintains accurate time, even after power disconnection, due to its backup battery.
4. **Performance**: The Arduino Uno handles the TM1637 display updates and RTC communication efficiently with no noticeable lag.

The project worked as expected, with a stable display of accurate real-time information, indicating that the circuit design and code implementation were successful.

8. Applications

This real-time clock project has several practical applications, such as:

- **Digital Clocks**: Can be used as a basic digital clock for home, office, or educational purposes.
- **Timing Systems**: Useful for projects that require precise timing, like alarms, reminders, or countdowns.
- **Embedded Systems**: Suitable for embedded projects requiring time tracking, such as automated lighting or HVAC systems.
- **Learning Tool**: Provides a practical example for beginners learning about Arduino, RTCs, and 7-segment displays.

9. Limitations

Despite its effectiveness, this project has some limitations:

1. **Limited Accuracy of DS1307**: The DS1307 RTC may drift slightly over time, leading to small inaccuracies if not periodically synchronized.
2. **No AM/PM Indicator**: The 7-segment display does not indicate AM or PM, which may cause confusion if the user cannot determine whether it is morning or evening.
3. **Limited Display Format**: The TM1637 display is limited to four digits, which restricts additional information like seconds or date.
4. **Basic Functionality**: The project provides only a basic real-time clock without advanced features like alarms or timers.

10. Conclusion

In conclusion, this project successfully demonstrated a simple and effective way to display real-time clock data using an Arduino Uno, DS1307 RTC module, and TM1637 7-segment display. The project illustrates the ease with which Arduino can be integrated with timekeeping modules and display units to create functional applications. It provides a foundational understanding of I2C communication, time formatting, and display control, making it a valuable learning experience for beginners and a useful utility for basic timekeeping needs.

This real-time clock can be further enhanced by adding features like AM/PM indicators, seconds display, or integrating with other modules to expand its functionality. The project thus serves as both a standalone tool and a foundation for more complex time-based projects.

11. References

1. **Datasheets**:
    - DS1307 Real-Time Clock (RTC) Module Datasheet
    - TM1637 4-Digit 7-Segment Display Datasheet
2. **Arduino Documentation**:
    - Arduino Official Documentation: https://www.arduino.cc/reference/en/
    - Arduino Wire Library Documentation: https://www.arduino.cc/en/Reference/Wire
3. **Libraries Used**:
    - RTClib Library: https://github.com/adafruit/RTClib
    - TM1637Display Library: https://github.com/avishorp/TM1637
4. **Tutorials and Guides**:
    - Arduino Projects Hub: https://create.arduino.cc/projecthub
    - RTC and TM1637 Display Integration Guides from Online Resources