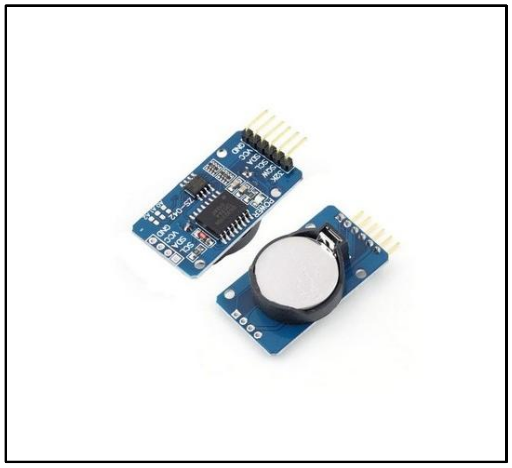
Arduino DS3231 AT24C32 I2C Real Time Clock Module c/w CR2032 Battery



Introduction

RTC DS3231 is a low-cost, extremely accurate I²C real-time clock (RTC), with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, disconnect the main power supply and maintains accurate timekeeping. Integrated oscillator improved long-term accuracy of the device and reduces the number of components of the production line. The DS3231 is available in commercial and industrial temperature ranges, using a 16-pin 300mil SO package.

RTC maintains seconds, minutes, hours, day, date, month, and year information. Less than 31 days of the month, the end date will be automatically adjusted, including corrections for leap year. The clock operates in either the 24 hours or band / AM / PM indication of the 12-hour format. Provides two configurable alarm clock and a calendar can be set to a square wave output. Address and data are transferred serially through an I²C bidirectional bus.

A precision temperature-compensated voltage reference and comparator circuit monitors the status of VCC to detect power failures, provide a reset output, and if necessary, automatically switch to the backup power supply. In addition, / RST pin is monitored as generating µP reset manually.

Save time and high precision addition, DS3231 also has some other features that extend the system host of additional features and a range of options. The device integrates a very precise digital temperature sensor, through the I²C * interface to access it (as the same time).

This temperature sensor accuracy is ± 3 ° C. On-chip power supply control circuit can automatically detect and manage the main and standby power (i.e., low-voltage battery) to switch between the power supply. If the main power failure, the device can continue to provide accurate timing and temperature, performance is not affected. When the main power re-power or voltage value returns to within the allowable range, the on-chip reset function can be used to restart the system microprocessor.

Components

- Arduino Uno
- Breadboard
- DS2321 AT24C32 High precision clock module I2C
- Jumper wires

Objectives:

In this experiment, we will monitor the date and time functions using a DS3231 RTC connected via I2C and Wire Lib.

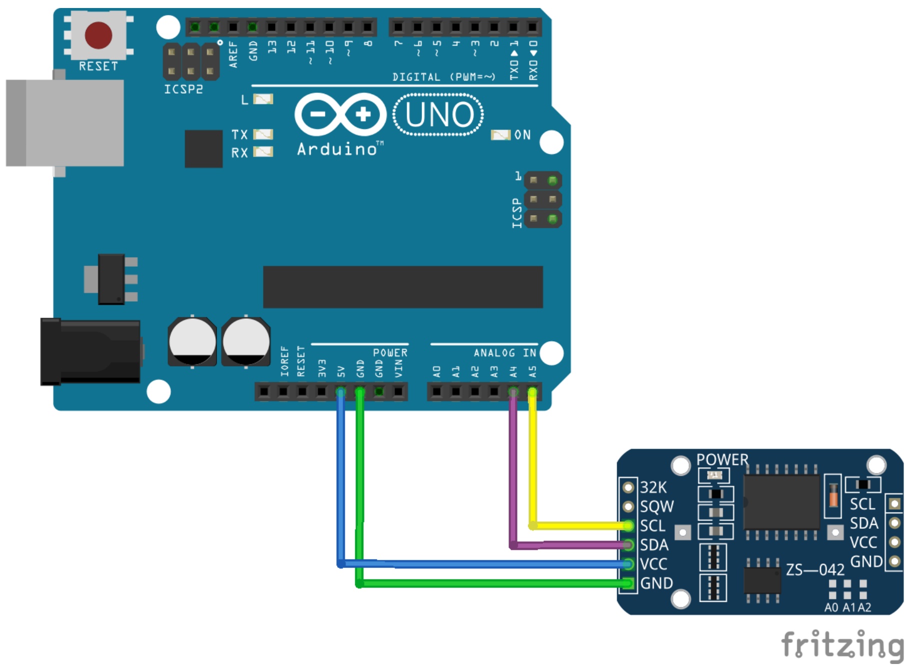
Procedures:

Step 1: Build the circuit connection as shown in table below.

The connection between the Serial I2C Adapter and the Arduino Uno board:

DS2321 AT24C32 High precision clock module I2C	Arduino Uno
GND	GND
VCC	5V
SDA	A4
SCL	A5

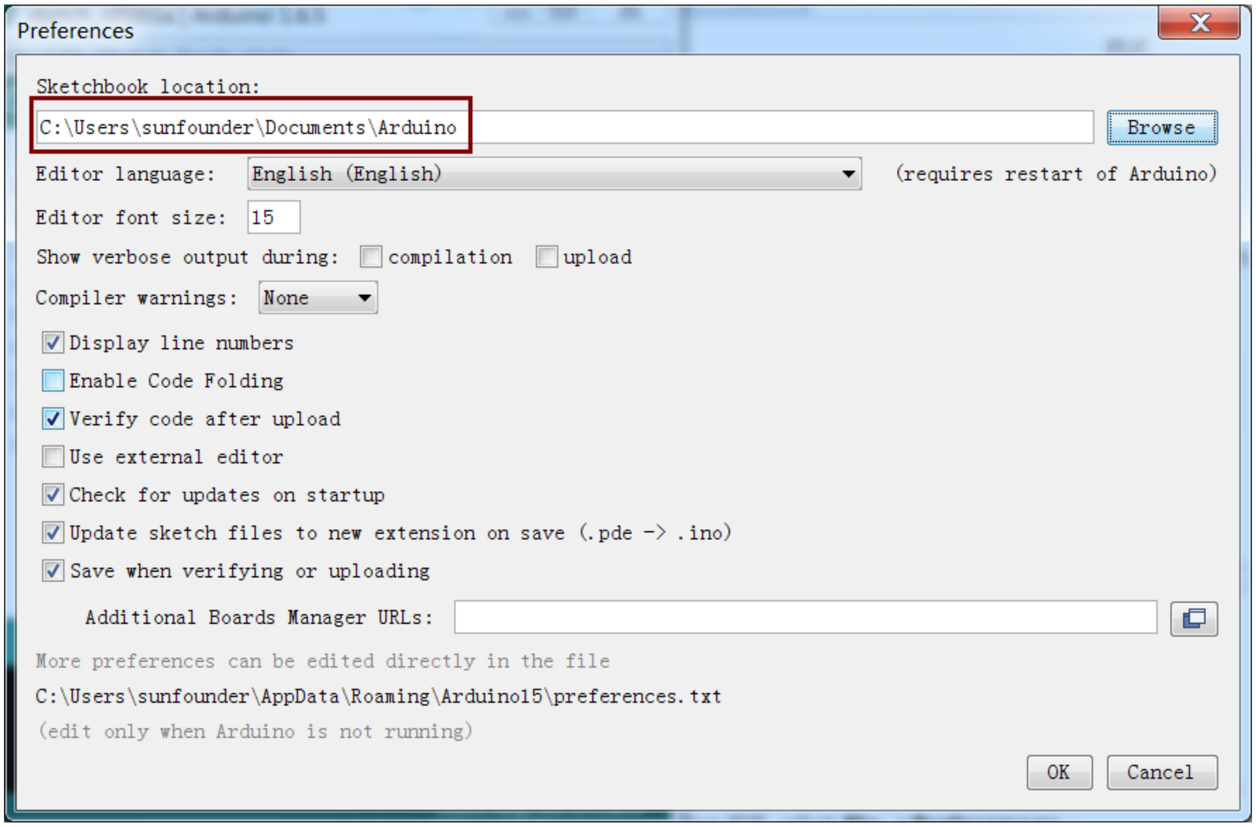
please make sure the wiring is same as the picture given



Step 2: Insert the sample programming provided below by copy and paste it into Arduino IDE.

```
// Date and time functions using a DS3231 RTC connected via I2C and Wire Lib

#include <Wire.h>
#include "RTClib.h"  // Credit: Adafruit

RTC_DS1307 RTC;

void setup() {
  // Begin the Serial connection
  Serial.begin(9600);

  // Instantiate the RTC
  Wire.begin();
  RTC.begin();

  // Check if the RTC is running.
  if (! RTC.isrunning()) {
    Serial.println("RTC is NOT running");
  }
  // This section grabs the current datetime and compares it to
  // the compilation time.  If necessary, the RTC is updated.
  DateTime now = RTC.now();
  DateTime compiled = DateTime(__DATE__, __TIME__);
  if (now.unixtime() < compiled.unixtime()) {
    Serial.println("RTC is older than compile time! Updating");
    RTC.adjust(DateTime(__DATE__, __TIME__));
  }

  Serial.println("Setup complete.");
}

void loop() {
  // Get the current time
  DateTime now = RTC.now();

  // Display the current time
  Serial.print("Current time: ");
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(' ');
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.print(now.second(), DEC);
  Serial.println();

  delay(10000);
}
```

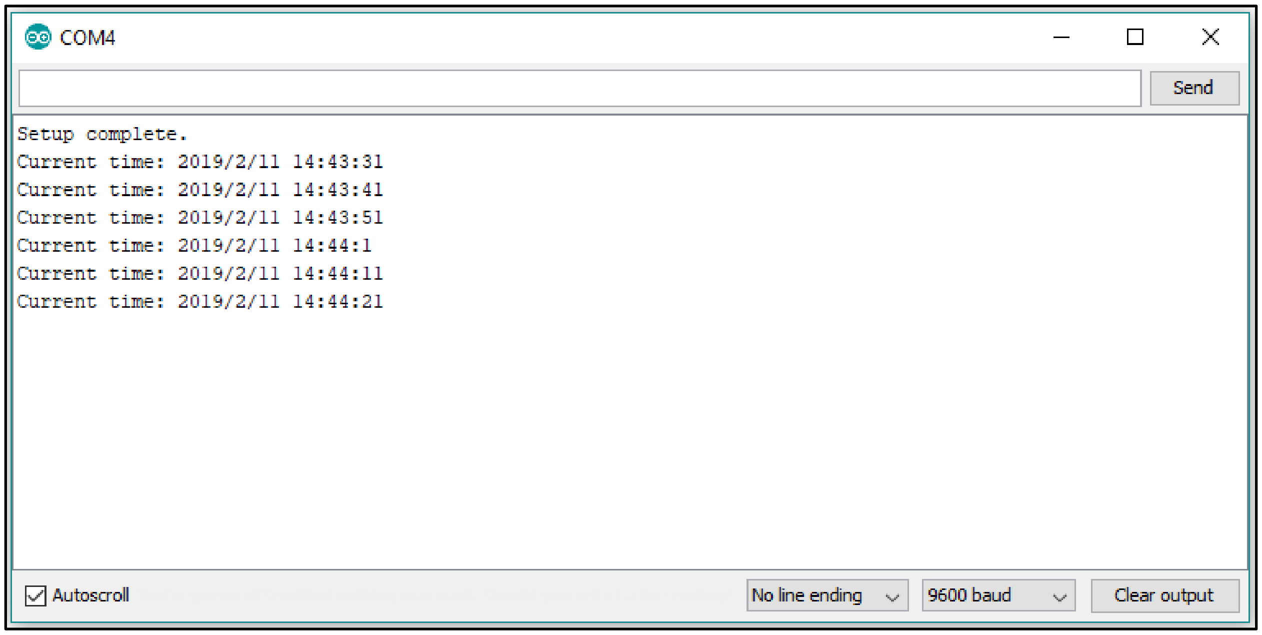
Step 3: Since in some code, the libraries needed are not included in Arduino, so you need to add them before compiling. Unzip the downloaded file. Copy the folders under the Library folder to the libraries folder in Arduino (if you cannot find the path in Arduino, open Arduino IDE, click File ->Preferences, and you can see the path in the Browse box, as shown in the following diagram). Compile the program.



Sketchbook location: C:\Users\sunfounder\Documents\Arduino
Editor language: English (English) (requires restart of Arduino)
Editor font size: 15
Show verbose output during: [ ] compilation [ ] upload
Compiler warnings: None
[✓] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Check for updates on startup
[✓] Update sketch files to new extension on save (.pde -> .ino)
[✓] Save when verifying or uploading
Additional Boards Manager URLs:
More preferences can be edited directly in the file
C:\Users\sunfounder\AppData\Roaming\Arduino15\preferences.txt
(edit only when Arduino is not running)

Step 4: Upload the sketch to the Arduino Uno board

Step 5: Open the serial monitor to observe the result as shown below.



```
Setup complete.
Current time: 2019/2/11 14:43:31
Current time: 2019/2/11 14:43:41
Current time: 2019/2/11 14:43:51
Current time: 2019/2/11 14:44:1
Current time: 2019/2/11 14:44:11
Current time: 2019/2/11 14:44:21
```

[✓] Autoscroll   No line ending   9600 baud