

What is the basic use of the RTC DS1307?

The basic use of the

Can you provide me with a basic code for Arduino Uno that simply calls for the current date and time using the DS1307 and displays the response in the serial monitor?

Show thinking

Of course. Here is a basic Arduino sketch to read the time from a DS1307 and display it in the Serial Monitor.

C++

None

```
// Include the required libraries
#include <Wire.h>
#include "RTClib.h"

// Create an RTC object
RTC_DS1307 rtc;

void setup () {
  // Start serial communication
  Serial.begin(9600);

  // Wait for Serial Monitor to open (for some boards)
  while (!Serial) {
    ;
  }

  // Initialize the RTC
  if (!rtc.begin()) {
    Serial.println("Couldn't find RTC");
  }
}
```

```
    Serial.flush();
    abort();
}

// --- SET TIME ONCE ---
// The following line should be UNCOMMENTED the first time you
run the sketch
// to set the RTC to the time this sketch was compiled.
// After running it once, COMMENT IT OUT and re-upload the
sketch.
// Otherwise, the time will be reset every time the Arduino
restarts.

// rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

}

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

    // Print the date
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" ");
```

```
// Print the time
Serial.print(now.hour(), DEC);
Serial.print(':');
if (now.minute() < 10) {
    Serial.print('0'); // Add leading zero for minutes
}
Serial.print(now.minute(), DEC);
Serial.print(':');
if (now.second() < 10) {
    Serial.print('0'); // Add leading zero for seconds
}
Serial.print(now.second(), DEC);

Serial.println(); // New line

// Wait for 3 seconds before printing again
delay(3000);
}
```

Prerequisites

1. **Hardware:**
 - Arduino Uno
 - DS1307 RTC Module (with a CR2032 battery inserted)
 - Jumper wires
2. **Library:** You need the "**RTCLib**" by **Adafruit**. To install it:
 - In the Arduino IDE, go to **Sketch -> Include Library -> Manage Libraries...**
 - Search for "RTCLib" and install the one by Adafruit.

How to Use

1. Connect the Hardware:

- **GND** on DS1307 -> **GND** on Arduino
- **VCC** on DS1307 -> **5V** on Arduino
- **SDA** on DS1307 -> **A4** on Arduino Uno
- **SCL** on DS1307 -> **A5** on Arduino Uno

2. Set the Time (First Upload):

- In the code, **uncomment** this line: `// rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));`
- Upload the sketch to your Arduino. This sets the RTC's time to your computer's time at the moment of compilation.

3. Run the Clock (Second Upload):

- **Comment out** the same line again: `// rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));`
- Re-upload the sketch.

4. View the Output:

- Open the Serial Monitor (**Tools -> Serial Monitor**).
- Set the baud rate to **9600**. You will see the current date and time printed every three seconds.

Now that I have this code working, can you help me with code to flash a single neopixel? The neopixel should alternate between dark blue and white each time a new time is listed in the serial monitor. Use the same code provided earlier and add in the new requirements.

Show thinking

Of course. Here is the updated code that controls a single NeoPixel, alternating its color between dark blue and white each time the time is printed.

C++

None

```
// Include the required libraries
#include <Wire.h>
#include "RTClib.h"
#include <Adafruit_NeoPixel.h>

// --- NeoPixel Setup ---
#define NEOPIXEL_PIN 6 // The pin the NeoPixel is connected to
#define NUMPIXELS 1 // The number of NeoPixels in the strip
```

```
// Create a NeoPixel object
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS,
NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);

// --- RTC Setup ---
RTC_DS1307 rtc;

// --- State Variable ---
// This variable will track which color to display next
bool isBlue = true;

void setup () {
  // Start serial communication
  Serial.begin(9600);

  // Initialize the RTC
  if (!rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }

  // --- Initialize NeoPixel ---
  pixels.begin(); // This initializes the NeoPixel library.
  pixels.clear(); // Set all pixels to 'off'
  pixels.show(); // Send the 'off' state to the pixel

  // --- SET TIME ONCE ---
```

```
    // Uncomment this line only on the first upload to set the
time.
    // Then, comment it out and re-upload.
    // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}
```

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

    // Print the date and time to the Serial Monitor
    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(':');
    if (now.minute() < 10) Serial.print('0');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    if (now.second() < 10) Serial.print('0');
    Serial.print(now.second(), DEC);
    Serial.println();

    // --- Update NeoPixel Color ---
    if (isBlue) {
        // Set the pixel to dark blue
```

```
    pixels.setPixelColor(0, pixels.Color(0, 0, 128));
  } else {
    // Set the pixel to white
    pixels.setPixelColor(0, pixels.Color(255, 255, 255));
  }
  pixels.show(); // Send the updated color to the pixel

  // Flip the state for the next loop
  isBlue = !isBlue;

  // Wait for 3 seconds before running the loop again
  delay(3000);
}
```

Hardware Setup

You'll need to connect the NeoPixel in addition to your RTC module.

- **RTC Module:** Keep it connected as before (GND->GND, VCC->5V, SDA->A4, SCL->A5).
- **NeoPixel:**
 - **GND** pin -> **GND** on Arduino
 - **5V** or **VCC** pin -> **5V** on Arduino
 - **Data In (DIN)** pin -> **Digital Pin 6** on Arduino

Required Library

In addition to **RTCLib**, you now need the **Adafruit NeoPixel** library.

- In the Arduino IDE, go to **Sketch** -> **Include Library** -> **Manage Libraries...**
 - Search for "**Adafruit NeoPixel**" and click **Install**.
-

What's New in the Code ✨

1. **NeoPixel Library:** The line `#include <Adafruit_NeoPixel.h>` is added to include the necessary functions for controlling the pixel.
2. **NeoPixel Object:** We define the pin (`D6`), the number of pixels (`1`), and create a `pixels` object to represent our NeoPixel.
3. **State Variable:** A boolean variable `isBlue` is created. This acts as a simple switch to remember which color was last displayed.
4. **Initialization:** In `setup()`, `pixels.begin()` and `pixels.clear()` are called to prepare the NeoPixel for use and ensure it's off at startup.
5. **Color Logic:** In `loop()`, an `if/else` statement checks the `isBlue` variable. It sets the pixel to either dark blue or white. The line `pixels.show()` actually sends the color data to the pixel, making it light up.
6. **Alternating Colors:** The line `isBlue = !isBlue;` is the key to alternating. It flips the value of the boolean from `true` to `false` (and vice-versa) at the end of each loop, ensuring the other color is chosen on the next run.

Great! That worked perfectly. Next, can you provide me with the code to just do the following: flash a neopixel 3 seconds dark blue, 3 seconds white in a loop until stopped? I would like to program this onto a Raspberry Pi Pico.

Show thinking

Of course. Here is the MicroPython code for a Raspberry Pi Pico to flash a single NeoPixel between dark blue and white in a continuous loop.

Python

```
None
# Import necessary libraries

import machine
import neopixel
import time

# --- NeoPixel Configuration ---
PIN_NUM = 28 # GPIO pin connected to the NeoPixel's Data In
(DIN) pin
NUM_LEDS = 1 # Number of NeoPixels
```



```
# Create a NeoPixel object
np = neopixel.NeoPixel(machine.Pin(PIN_NUM), NUM_LEDS)

# --- Color Definitions (R, G, B) ---
DARK_BLUE = (0, 0, 128)
WHITE = (255, 255, 255)

# --- Main Loop ---
while True:
    # Set pixel to dark blue
    np[0] = DARK_BLUE
    np.write() # Send the color data to the pixel
    time.sleep(3) # Wait for 3 seconds

    # Set pixel to white
    np[0] = WHITE
    np.write() # Send the color data to the pixel
    time.sleep(3) # Wait for 3 seconds
```

Hardware Setup

1. Connect your Raspberry Pi Pico to your computer.
 2. Wire the NeoPixel to the Pico:
 - **GND** on NeoPixel -> **GND** on Pico
 - **VCC / 5V** on NeoPixel -> **VBUS** on Pico (this provides 5V from the USB connection)
 - **Data In (DIN)** on NeoPixel -> **GP28** on Pico
-

How to Use

1. Open an IDE compatible with MicroPython, like **Thonny** (which comes pre-installed with Raspberry Pi OS).
 2. If using Thonny, make sure your interpreter is set to **MicroPython (Raspberry Pi Pico)** in the bottom-right corner.
 3. Copy and paste the code into the editor.
 4. Save the file directly to your Raspberry Pi Pico with the name `main.py`. This will make the code run automatically whenever the Pico has power.
 5. Press the **Run** button (or unplug and replug the Pico) to start the light sequence.
-

How the Code Works

- **Libraries:** We import `machine` to control hardware pins, `neopixel` to control the LED, and `time` for the delay.
- **Configuration:** We define the GPIO pin number and create a `neopixel` object named `np` linked to that pin.
- **Colors:** We define the colors `DARK_BLUE` and `WHITE` as tuples of (`Red`, `Green`, `Blue`) values. This makes the code easier to read.
- **Infinite Loop:** The `while True:` loop runs forever until the Pico loses power.
- **Setting Color:** Inside the loop, `np[0] = DARK_BLUE` assigns the blue color to the first (and only) pixel in our object. `np.write()` sends this command to the physical NeoPixel, causing it to light up.
- **Delay:** `time.sleep(3)` pauses the program for 3 seconds. The code then repeats these steps for the white color.