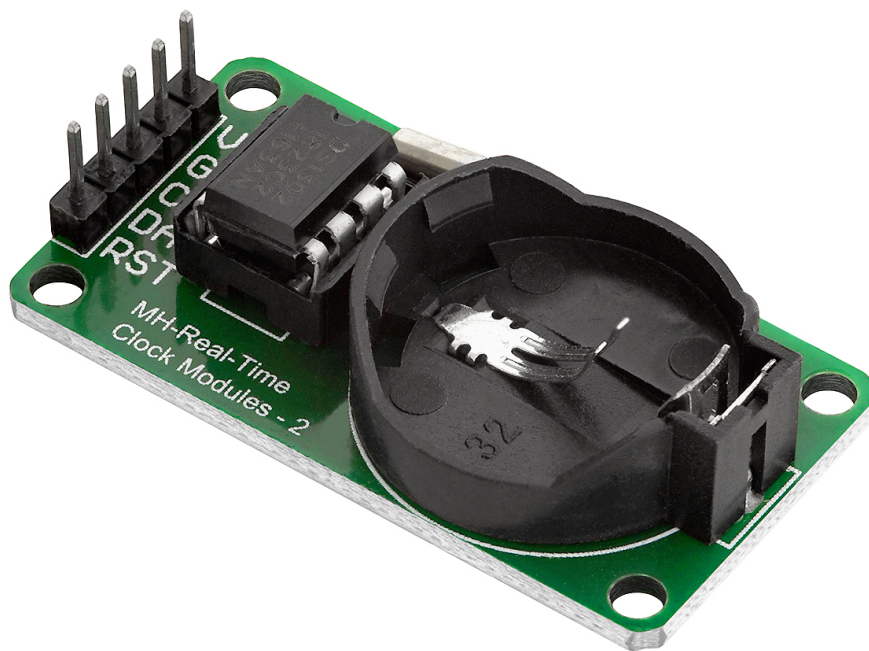


AZ-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery DS1302 Real-time Clock Module*.
On the following pages, you will be introduced to how to use and set-up this
handy device.

Have fun!



Az-Delivery

Table of Contents

Introduction.....	3
Specifications.....	4
The pinout.....	5
How to set-up Arduino IDE.....	6
How to set-up the Raspberry Pi and Python.....	10
Connecting the module with Atmega328p.....	11
Library for Arduino IDE.....	12
Sketch example.....	13
Connecting the module with Raspberry Pi.....	22
Libraries and tools for Python.....	23
Python script.....	24



Introduction

The DS1302 Real-time Clock module is used as a time synchronization device in applications where precise timings are essential. The module is used in digital clocks, computer motherboards, digital cameras, embedded systems, etc.

The module contains a real-time clock and 31B of RAM and provides clock and calendar functions. It can provide seconds, minutes, hours, weekdays, month days, months and years information. The module can be set to work in 12 or 24 hour format. It has AM/PM indicator ability and it is designed to operate with very low power consumption, less than $1\mu W$.

The power supply for the module is in the range from 2V to 5V.

The module uses very simple synchronous serial communication otherwise known as *ThreeWire Interface*. Only three wires are required for data exchange between a microcontroller and a module. Data can be transferred to and from the module one byte at a time or in a burst of up to 31 bytes.

There is an on-board battery holder, but AZ-Delivery does not ship the module with a battery. Use a battery as a back-up power supply for the module.

Az-Delivery

Specifications

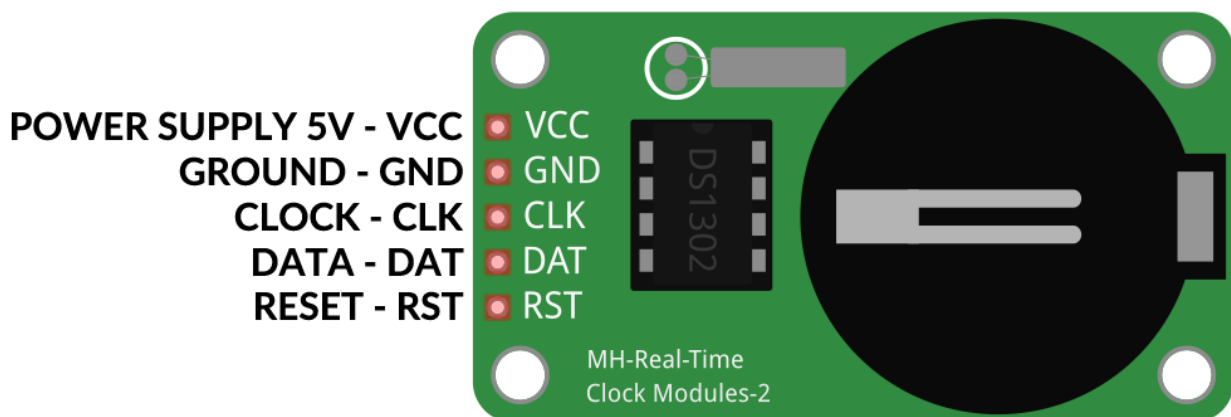
Power supply voltage	from 2V to 5V
Operating temperature	from 0°C to +70°C
Communication interface	SSPI [ThreeWire]
Battery backup	One battery holder without a battery!
Time of day alarms	2
Low power consumption	less then 1mA
Dimensions	43 x 23 x 20mm [1.7 x 0.9 x 08in]

The DS1302 chip uses the external 32.768kHz crystal. The on-board oscillator circuit does not require any external resistors or capacitors to operate. When using a crystal with the specified characteristics, the startup time is usually less than one second. The DS1302 module can also use the external oscillatror, but this is not covered in this eBook.

The module has a battery holder for one 3V coin cell battery. The battery can be used as a back-up power supply. When the main power supply is disconnected, the automatic detection of on-board chip switches to battery back-up.

The pinout

The DS1302 Real-time Clock module has five pins. The pinout is shown on the following image:



How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the [link](#) and download the installation file for the operating system of choice.

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle containing the Arduino logo (an infinity symbol with a minus sign on the left and a plus sign on the right). To the right of the logo, the text reads: **ARDUINO 1.8.9**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side of the page, there is a teal sidebar with the following options: **Windows** Installer, for Windows XP and up; **Windows** ZIP file for non admin install; **Windows app** Requires Win 8.1 or 10 with a "Get" button; **Mac OS X** 10.8 Mountain Lion or newer; **Linux** 32 bits; **Linux** 64 bits; **Linux** ARM 32 bits; **Linux** ARM 64 bits; [Release Notes](#); [Source Code](#); [Checksums \(sha512\)](#).

For *windows* users, double click on the downloaded .exe file and follow the instructions in the installation window.

Az-Delivery

For *Linux* users, download a file with the extension `.tar.xz`, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two `.sh` scripts have to be executed, the first called `arduino-linux-setup.sh` and the second called `install.sh`.

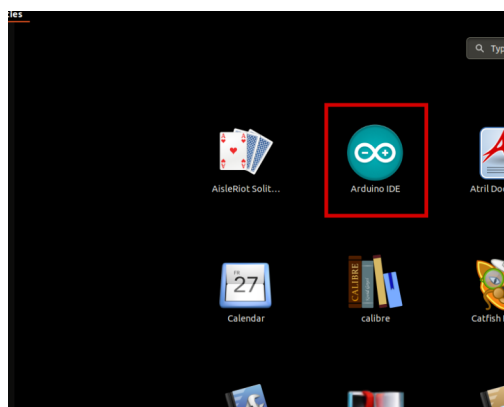
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

```
sh arduino-linux-setup.sh user_name
```

user_name - is the name of a superuser in Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

The second script, called `install.sh`, has to be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.



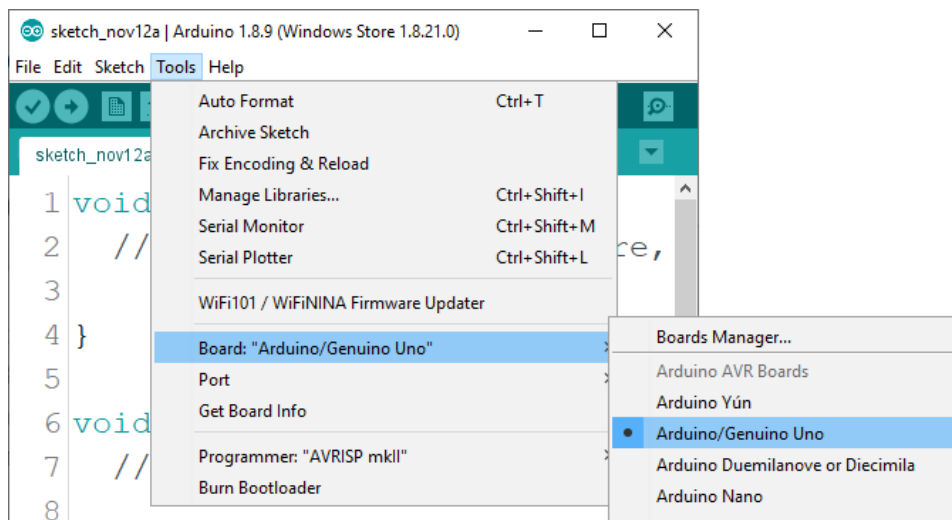
Az-Delivery

Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an Atmega328p board. Open freshly installed Arduino IDE, and go to:

Tools > Board > {your board name here}

{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:



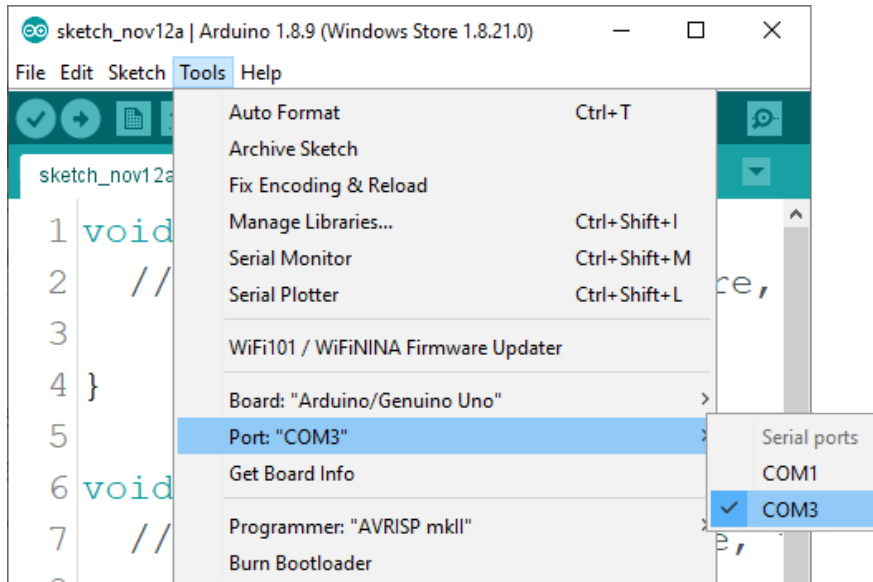
The port to which the Atmega328p board is connected has to be selected.

Go to: *Tools > Port > {port name goes here}*

and when the Atmega328p board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

Az-Delivery

If the Arduino IDE is used on Windows, port names are as follows:



For *Linux* users, for example, port name is `/dev/ttyUSBx`, where *x* represents integer number between 0 and 9.



How to set-up the Raspberry Pi and Python

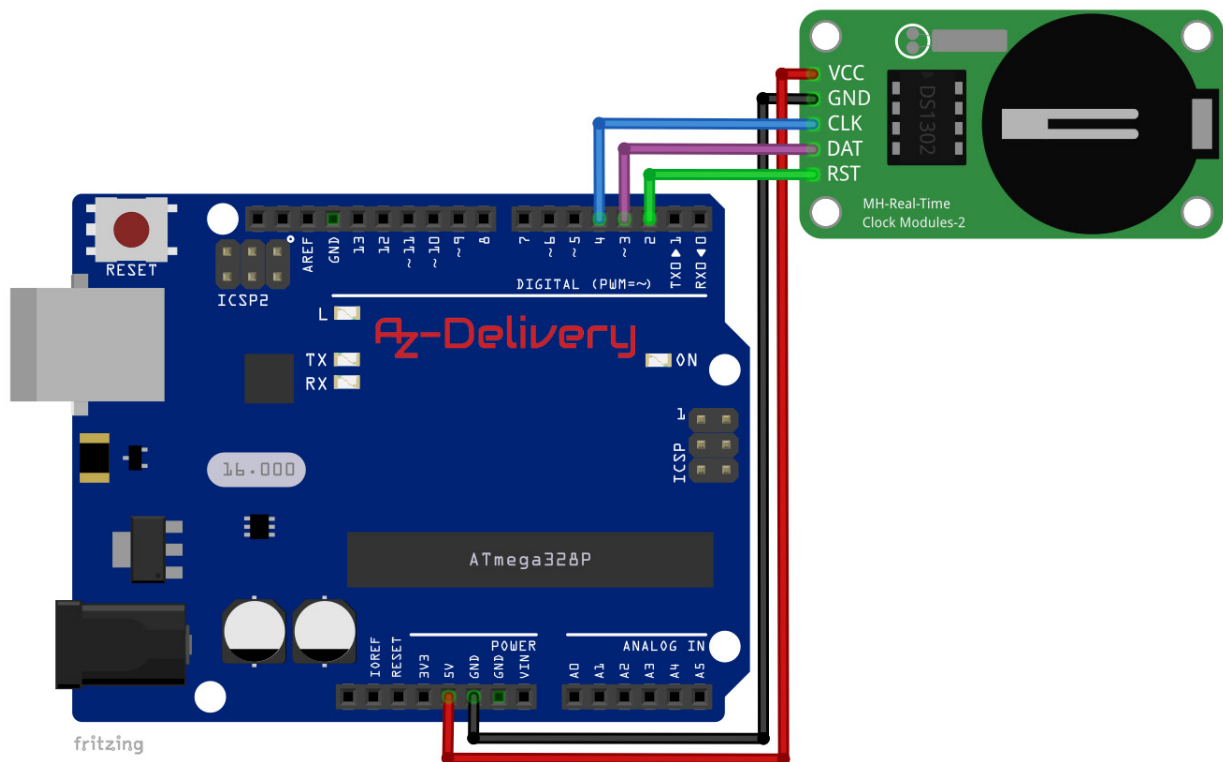
For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the *Headless* mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that are used in this mode are the Raspberry Pi itself, power supply and internet connection. All of this is explained minutely in the free eBook: [Raspberry Pi Quick Startup Guide](#)

The *Raspbian* operating system comes with *Python* preinstalled.

Az-Delivery

Connecting the module with Atmega328p

Connect the module with the Atmega328p as shown on the following connection diagram:



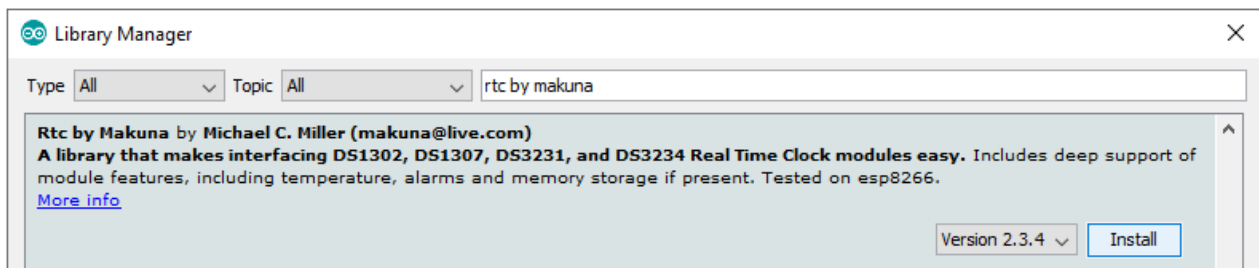
RTC pin	Mc pin	Wire color
VCC	5V	Red wire
GND	GND	Black wire
CLK	D4	Blue wire
DAT	D3	Purple wire
RST	D2	Green wire

Library for Arduino IDE

To use the module with Atmega328p, it is recommended to download an external library. The library that is used in this eBook is called the *RTC by Makuna*. To download it, open Arduino IDE and go to:

Tools > Manage Libraries.

When the new window opens, type *RTC by Makuna* in the search box and install the library made by *Michael C. Miller*, as shown in the following image:



With the library comes two sketch examples, to open one, go to:

File > Examples > RTC by Makuna > DS1302_Memory

With this sketch the module can be tested. The sketch on the following page is simplified and is more user friendly version.

Az-Delivery

Sketch example

```
#include <ThreeWire.h>
#include <RtcDS1302.h>

ThreeWire myWire(3, 4, 2); // DAT/IO, CLK/SCLK, RST/CE
RtcDS1302<ThreeWire> Rtc(myWire);

void setup ()
{
  Rtc.Begin();
  Serial.begin(9600);
  //Uncomment to write current PC time to the RTC
  RtcDateTime cdt = RtcDateTime(__DATE__, __TIME__);
  //Uncomment to enter manually date and time to the RTC
  //RtcDateTime cdt = RtcDateTime("Jan 14 2015", "09:55:20");
  Rtc.SetDateTime(cdt);
}

void loop ()
{
  RtcDateTime pdt = Rtc.GetDateTime();
  printDateTime(pdt);
  Serial.println();
  delay(2000);
}
```

AZ-Delivery

```
void printDateTime(const RtcDateTime& dt) {
    //Day of the week
    Serial.print("Day of the week: ");
    if (dt.DayOfWeek() == 1) {
        Serial.println("Monday");
    }
    else if (dt.DayOfWeek() == 2) {
        Serial.println("Tuesday");
    }
    else if (dt.DayOfWeek() == 3) {
        Serial.println("Wednesday");
    }
    else if (dt.DayOfWeek() == 4) {
        Serial.println("Thursday");
    }
    else if (dt.DayOfWeek() == 5) {
        Serial.println("Friday");
    }
    else if (dt.DayOfWeek() == 6) {
        Serial.println("Saturday");
    }
    else if (dt.DayOfWeek() == 7) {
        Serial.println("Sunday");
    }
    // Current Date
    Serial.print("Current Date: ");
    if (dt.Day() < 10) {
        Serial.print("0");
        Serial.print(dt.Day());
    }
    else {
        Serial.print(dt.Day());
    }
}
```

AZ-Delivery

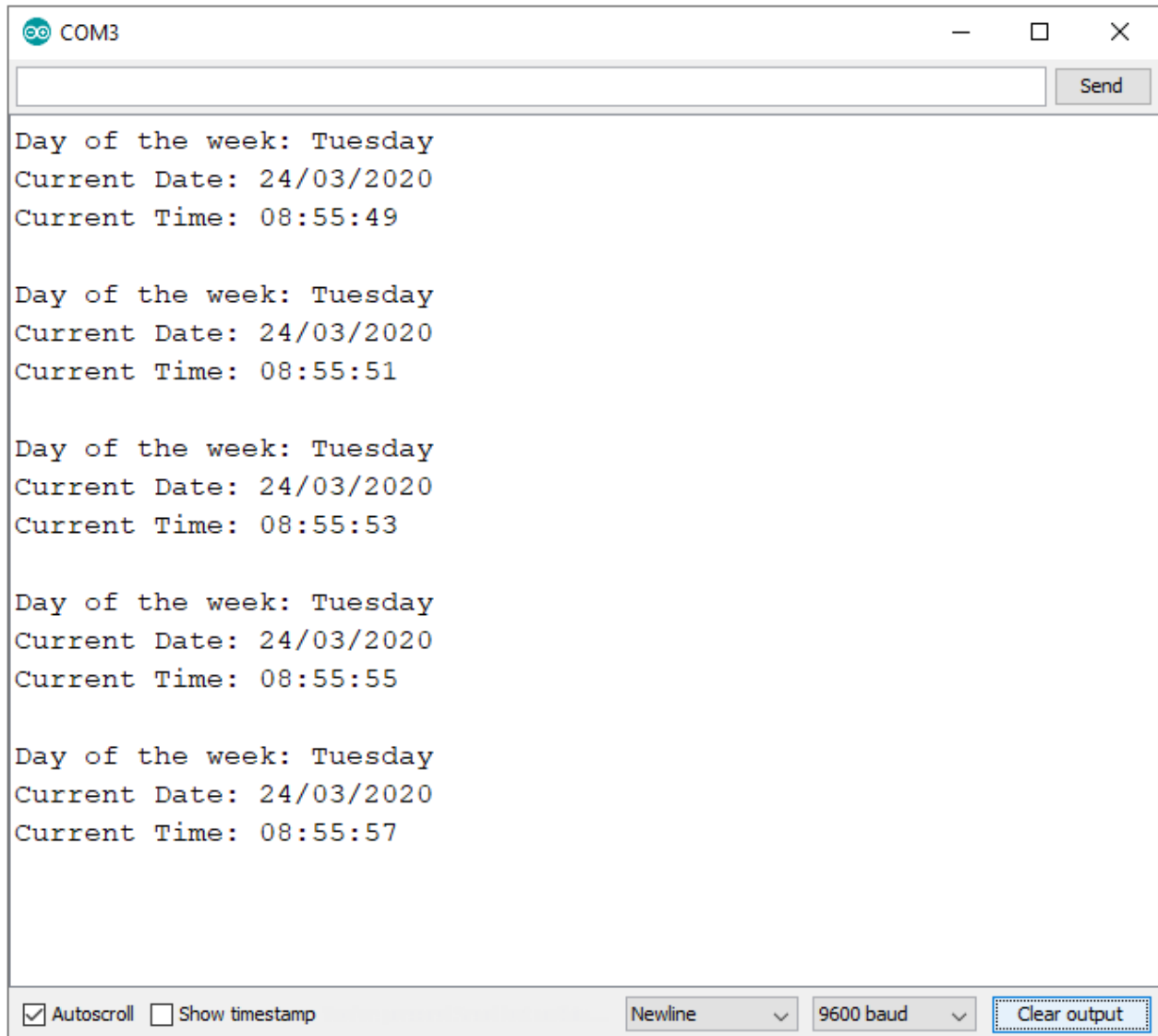
```
//one tab
Serial.print("/");
if (dt.Month() < 10) {
    Serial.print("0");
    Serial.print(dt.Month());
}
else {
    Serial.print(dt.Month());
}
Serial.print("/");
Serial.println(dt.Year());
//Current Time
Serial.print("Current Time: ");
if (dt.Hour() < 10) {
    Serial.print("0");
    Serial.print(dt.Hour());
}
else {
    Serial.print(dt.Hour());
}
Serial.print(":");
if (dt.Minute() < 10) {
    Serial.print("0");
    Serial.print(dt.Minute());
}
else {
    Serial.print(dt.Minute());
}
}
```

Az-Delivery

```
//one tab
Serial.print(":");
if (dt.Second() < 10) {
  Serial.print("0");
  Serial.print(dt.Second());
  Serial.println();
}
else {
  Serial.print(dt.Second());
  Serial.println();
}
}
```


Az-Delivery

Upload the sketch to the Atmega328p and run the Serial Monitor (*Tools > Serial Monitor*). The result should look like as on the following image:



Az-Delivery

At the beginning of the sketch two libraries called *ThreeWire* and *RtcDS1302* are imported. These libraries contain functions that are used for communication between the module and Atmega328p.

The object *myWire* which represents the communication interface is created with the following line of code: `ThreeWire myWire(4, 5, 2);` where numbers 4, 5, and 2 represent digital pins on the Atmega328p through which the module is connected. (*DATA-IN*, *RST* and *SCK*, respectively).

The second object called *Rtc* represents the module itself. To create it, *myWire* object is used like in the following line of code:

```
RtcDS1302<ThreeWire> Rtc(myWire);
```

At the beginning of *setup()* function, communication between the module and Atmega328p is started, with the following line of code: `Rtc.Begin();`

Then the serial interface with the baud rate of 9600bps is started.

There are two ways that can be used to set the date and time of the module. The first is to use the current system date and time and the second is to use the custom date and time values.

Az-Delivery

To set the current system date and time into the module, the following line of code is used:

```
RtcDateTime cdt = RtcDateTime(__DATE__, __TIME__);
```

With this line of code, the object called *cdt* is created, where `__DATE__` `__TIME__` arguments are used. The values of these arguments represent the current system date and time data.

To set the custom date and time values, the following line of code is used:

```
RtcDateTime cdt = RtcDateTime("Jan 14 2015", "09:55:20");
```

where the following format for date and time has to be used:

```
("MMM DD YYYY", "HH:MM:SS")
```

where: "Month DayOfMonth Year", "Hour:Minute:Second"

At the beginning of the *loop()* function data is read from the module after which the data is stored into *pdt* object. Storing data in *pdt* object is done with the following line of code:

```
RtcDateTime pdt = Rtc.GetDateTime();
```

Then function called *printDateTime()* is used to output data from the *pdt* object and it will be explained later in the text.

At the end of the *loop()* function, a delay of two seconds is set-up (*delay(2000)*). This way, output in the Serial Monitor is updated every two seconds.

Az-Delivery

The `printDateTime()` function has one argument and returns no value. The argument type is `RtcDateTime` which represents date and time data from the RTC. In the header of the function is the following:

```
const RtcDateTime& dt
```

where `dt` is the name of the parameter (argument); `&` means that the data is passed to the function by reference (what this means is not in the scope of this eBook); `const` means that data is read-only.

By default, when the date and time data is displayed in the Serial Monitor, numeric values that are less than 10 are displayed as single digit numbers (2:23:1 – hour:minute:second).

The original sketch is modified to display leading zeros for single digit numbers.

When data is read from the RTC module, the name of the weekday is a numeric value. To display names of weekdays, the following lines of code is used (for example, Monday):

```
if (dt.DayOfWeek() == 1) {  
    Serial.println("Monday");  
}
```

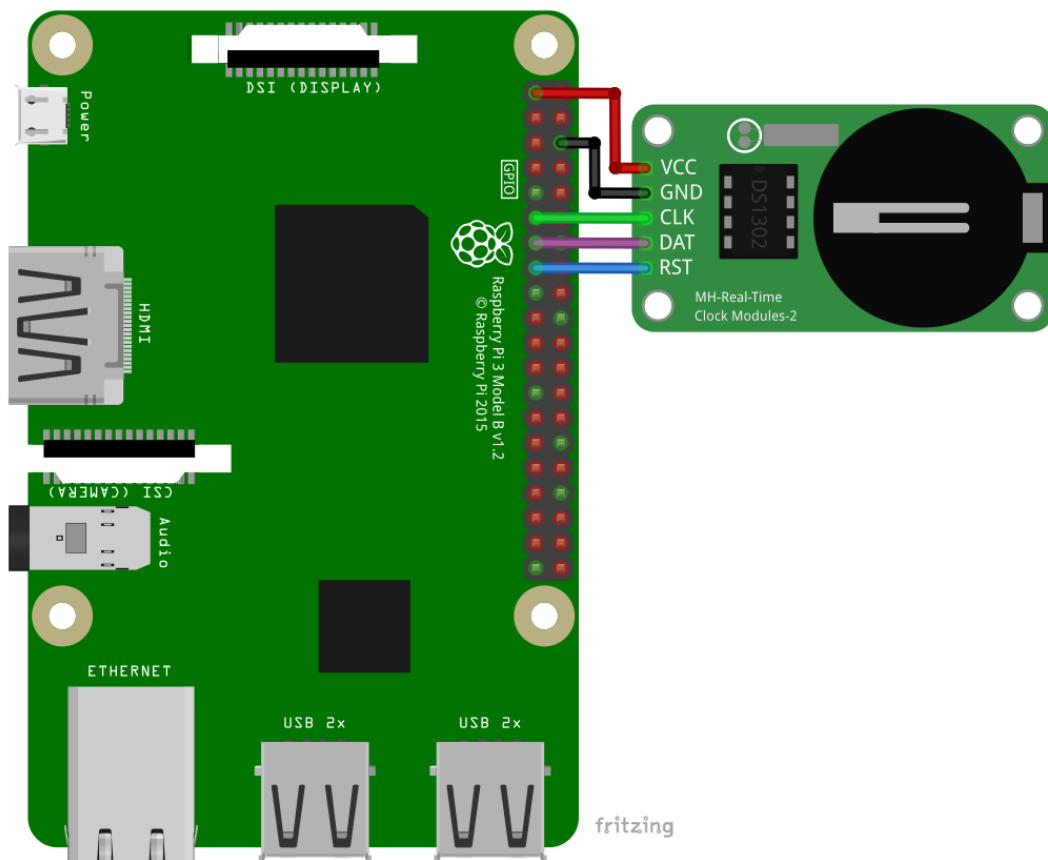
Az-Delivery

All other numeric values from data read from the RTC module are displayed with leading zero (for example adding zero to seconds), the following lines of code are used:

```
if (dt.Second() < 10) {  
    Serial.print("0");  
} else {  
    Serial.print(dt.Second());  
}
```

Connecting the module with Raspberry Pi

Connect the module with the Raspberry Pi as shown on the following connection diagram:



RTC pin	Raspberry Pi pin	Physical pin	Wire color
VCC	3V3	1	Red wire
GND	GND	6	Black wire
CLK	GPIO17	11	Green wire
DAT	GPIO27	13	Purple wire
RST	GPIO22	15	Blue wire

Libraries and tools for Python

To use the module with the Raspberry Pi, it is recommended to download an external library. In order to download an external library used in this eBook, the *git* app and *rpi.gpio* library should be installed. To do so, open the terminal and run the following commands, one by one:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo apt-get install -y python3-rpi.gpio git
```

The external library used in this eBook is called *rpi.rtc* and to download it open the terminal and run the following command:

```
git clone https://github.com/sourceperl/rpi.rtc.git
```

To install the library, run the following command:

```
sudo python3 setup.py install
```

The directory where the library is installed contains another sub-folder called *scripts*. There are two example scripts called *ds1302_get_utc* and *ds1302_set_utc*. To use these script files, file names have to be renamed with the **.py* extension.

Az-Delivery

Python script

```
import sys
import pyRPiRTC
import time
from datetime import datetime

my_format = '%d/%m/%Y %H:%M:%S'

rtc = pyRPiRTC.DS1302(clk_pin=11, data_pin=13, ce_pin=15)

def set_date_time(time, f=my_format):
    global rtc
    dt = datetime.strptime(time, f)
    rtc.write_datetime(dt)

print('Press CTRL + C to end the script!')
try:
    # set_date_time('01/01/2020 09:13:55') # uncomment this to set date/time
    while True:
        cdt = rtc.read_datetime()
        print('Current date: {}'.format(cdt.strftime('%d/%m/%Y')))
        print('Current time: {}\n'.format(cdt.strftime('%H:%M:%S')))
        time.sleep(1)

except ValueError:
    sys.exit('error with RTC chip, check wiring')

except KeyboardInterrupt:
    print('\nScript end!')

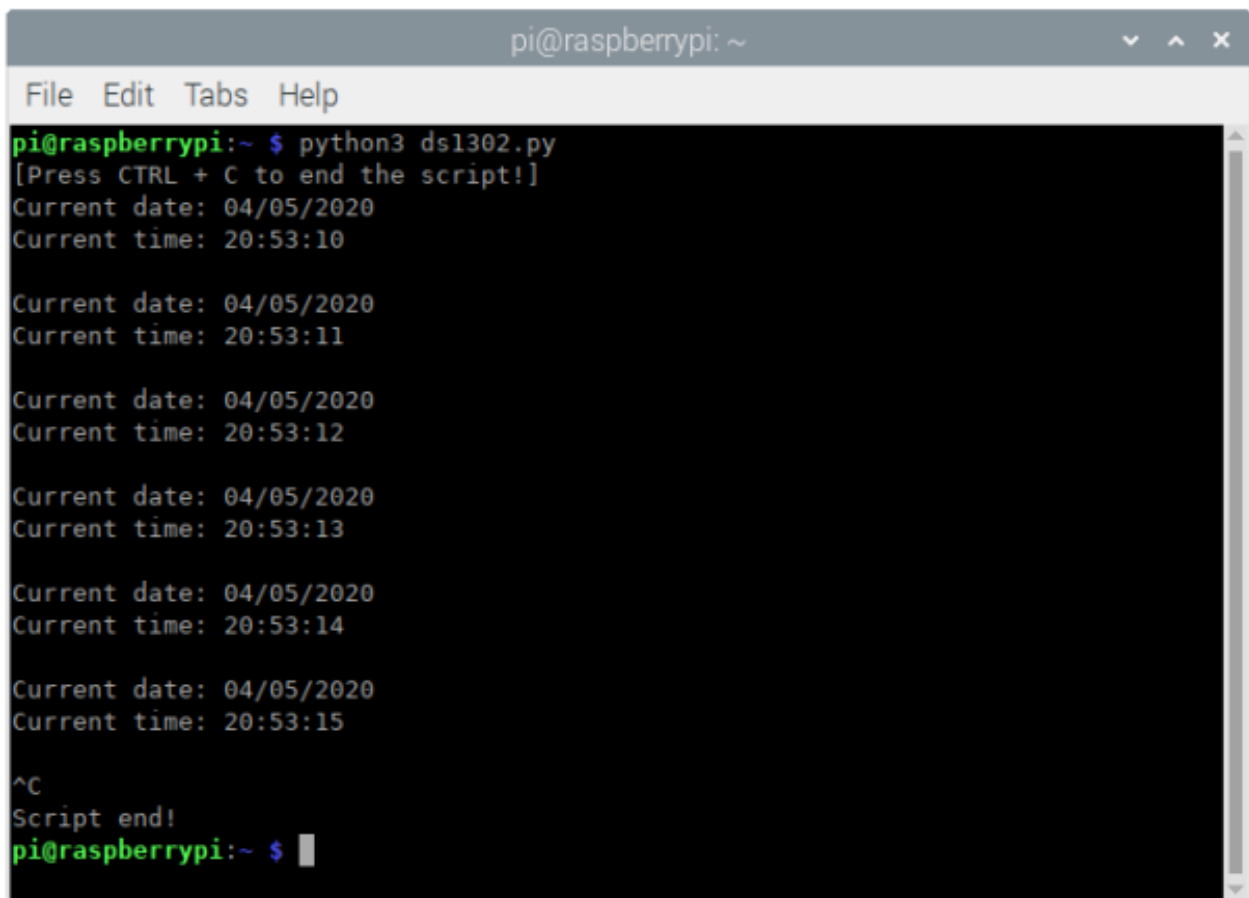
finally:
    rtc.close() # clean close
```


Az-Delivery

Save the script by the name *ds1302.py*. To run the script, open the terminal in the directory where the script is saved and run the following command:

```
python3 ds1302.py
```

The result should look like as on the following image:



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ python3 ds1302.py  
[Press CTRL + C to end the script!]  
Current date: 04/05/2020  
Current time: 20:53:10  
  
Current date: 04/05/2020  
Current time: 20:53:11  
  
Current date: 04/05/2020  
Current time: 20:53:12  
  
Current date: 04/05/2020  
Current time: 20:53:13  
  
Current date: 04/05/2020  
Current time: 20:53:14  
  
Current date: 04/05/2020  
Current time: 20:53:15  
  
^C  
Script end!  
pi@raspberrypi:~ $
```

To stop the script press 'CTRL + C' on the keyboard.

Az-Delivery

The script starts with importing of two libraries `sys` and `time`, one script `pyRPiRTC` and one function `datetime` from `datetime` library.

Next, the string variable called `my_format` is created. This variable contains the string which represents the date and time format used in the `datetime()` function.

```
my_format = '%d/%m/%Y %H:%M:%S'
```

What a specific letter in this string means can be read on the following [link](#). Slashes “/” and colons “:” are used as separators.

After this, the object called `rtc` is created with the following line of code:

```
rtc = pyRPiRTC.DS1302(clk_pin=11, data_pin=13, ce_pin=15)
```

where `11`, `13` and `15` are the names of the GPIO pins (BCM) on which pins of the module are connected.

Next, the function called `set_date_time()` is created. The function has two arguments and returns no value. The first argument represents the string with a specific date and time which is saved in the module. This argument has to be a string in the following format:

```
'01/01/2020 09:13:55'
```

where `01/01/2020` is day/month/year and `09:13:55` is hour:minute:second.

Az-Delivery

The second argument is called *f* and it represents the date and time format used for *datetime()* function. This argument is optional and its value is set by default to the value stored in *my_format* variable. The second argument is optional as it can be seen in the following line of code:

```
def set_date_time(time, f=my_format):  
    global rtc  
    dt = datetime.strptime(time, f)  
    rtc.write_datetime(dt)
```

The *set_date_time()* function sets the specific date and time in the module. Use it once to set the date and time, after that only read the data from the module.

After this, the *try-except-finally* block of code is created with two *except* parts. At the beginning of the *try* block there is one comment, which shows how to use the *set_date_time()* function. Uncomment this and set the desired date and time in order to write the date and time data into the RTC module. After this, the indefinite loop block is created (*while True:*). In the indefinite loop block, first, the RTC data is read with the following line of code: *cdt = rtc.read_datetime()* where *cdt* object is created and the date and time data are stored in this object.

Az-Delivery

Next, the data from *cdt* object is displayed in the terminal

The output can be seen on the image of the script output.

The first *except* block is used to catch the *ValueError*. This error happens if the data can not be read from the RTC module. When this error happens, the message: *error with RTC chip, check wiring* is displayed in the terminal.

The second *except* block is used to catch the *KeyboardInterrupt*. The keyboard interrupt happens when the *CTRL + C* is pressed on the keyboard. When this happens, the message: *Script end!* is displayed in the terminal.

The *finally* block of code is executed at the script end. When this block of code is executed, the function called *close()* is executed, which disables all GPIO pin modes and interfaces used in the script.

AZ-Delivery

Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the internet.

If you are looking for the high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

<https://az-delivery.de>

Have Fun!

Impressum

<https://az-delivery.de/pages/about-us>