**Explorando o Módulo RTC DS3231 com Arduino**

Conheça com detalhes o módulo **DS3231**, um **relógio de tempo real** (Real Time Clock) que traz maior precisão ao controle do tempo com Arduino

Existem muitos tutoriais sobre **RTC** com **Arduino** na Internet, mas a maioria é dedicada ao DS1307 e não existe muito material sobre o **DS3231** em português que aborde suas características, como definição de **alarmes** e medição da **temperatura**.

Neste tutorial, vamos explicar como tirar o máximo desse ***pop star*** do Mundo Arduino, embora já existam novas versões pipocando por aí, como DS3234.

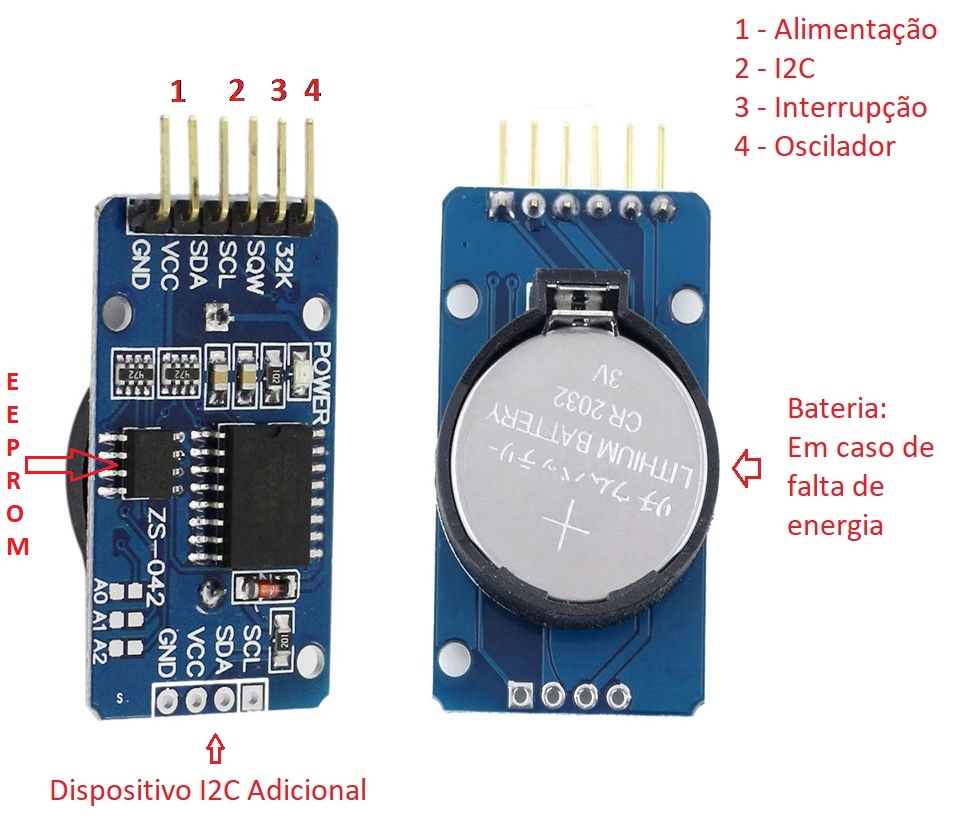
**Apresentando o CI DS3231**

RTC ds3231

O DS3231 é um chip (circuito integrado) fabricado pela [Maxim Integrated](https://www.maximintegrated.com/en.html" \t "_blank) com as seguintes características, entre outras :

* Controle completo e preciso de tempo (Datas e horas);;
* Definição de até dois alarmes;
* Sensor de temperatura;
* Sinal programável do oscilador de onda quadrada;
* Interface de comunicação I2C;
* Entrada para Bateria.

**O módulo DS3231**

Módulo DS3231

Existem no mercado vários módulos diferentes disponíveis para o ds3231 de diversos fabricantes mas a maioria apresenta, pelo menos, a seguinte interface:

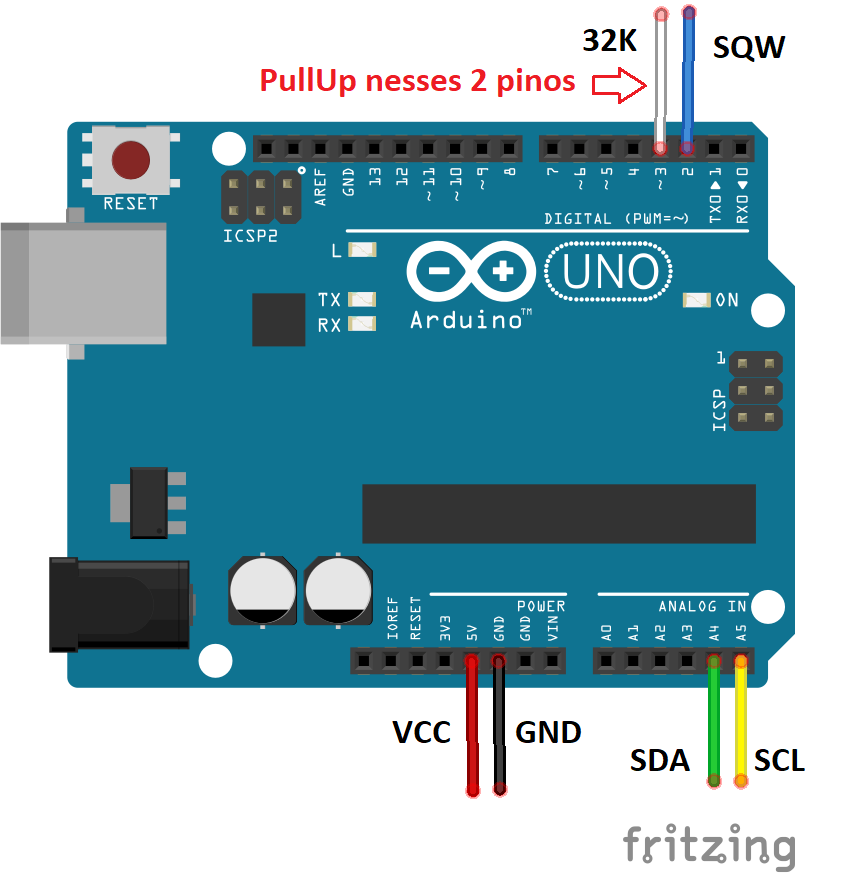
1. Pinos de Alimentação: VCC e GND – 3.3V ou 5V
2. Pinos de Comunicação: SCL e SDA – comunicação I2C – Endereço padrão: 0x57
3. Pino de interrupção: SQW – Usado para disparo de alarmes
4. Pino do oscilador – Saída de onda quadrada do cristal.

Observações:

* Alguns módulos, como esse da figura, disponibilizam uma segunda interface I2C para ligar um dispositivo em cascata, como um LCD, por exemplo.
* A bateria (CR2032 ou compatível) dura, em média, 5 anos e é usada quando ocorre queda da energia principal
* Esse módulo possui uma memória **EEPROM** (AT24C32) de 32K, útil para armazenar a configuração do relógio e como **DATA LOGGER.** O endereço da memória pode ser configurado através dos jumpers A0, A1 e A2,
* Devido à sua tensão de trabalho, esses módulos funcionam bem em outras plataformas como o ESP\*, Attiny, etc. É preciso tomar cuidado apenas na escolha da library que deve ser compatível
* Veja mais detalhes no diagrama desse módulo nas referências mais abaixo.

**Onde comprar: [Curto Circuito](https://www.curtocircuito.com.br/" \t "_blank)**

**Esquema de ligação**

Arduino e DS3231

Observações:

* Use o pino SQW se deseja controlar alarmes através de interrupções;
* O pino 32k raramente é utilizado, apenas se quiser obter os pulsos do oscilador de cristal (para escovadores de bits);
* No Arduino Mega, os pinos SDA e SCL são o 20 e 21, respectivamente. Consulte esse [link](https://www.arduino.cc/en/Reference/Wire" \t "_blank) para outras placas.

**Bibliotecas**

Separei aqui as principais bibliotecas disponíveis para o DS3231, com base em número de recursos e frequência de atualização:

* [RTClib](https://github.com/adafruit/RTClib) – Library patrocinada pela Adafruit, o que é uma boa referência, mas não possui métodos para obter a temperatura e precário controle de alarmes. É um fork da Jeelab’s RTC library
* [DS3231](https://github.com/NorthernWidget/DS3231" \t "_blank) – Library mantida pela [Northtern Widget](http://northernwidget.com/" \t "_blank)
* [DS3232RT](https://github.com/JChristensen/DS3232RTC" \t "_blank) – Library de Jack Cristensen bem completa e com boa taxa de atualização. Inclui métodos para gravação na EEPROM
* [Rtc3231](https://github.com/Makuna/Rtc" \t "_blank) – Outra biblioteca bem completa e ativa de Michael Miller
* [DS3231\_Simple](https://github.com/sleemanj/DS3231_Simple" \t "_blank) – Nunca foi tão fácil trabalhar com RTC. Autor: James Sleeman

**Programando o DS3231**

Todos exemplos de código aqui disponibilizados se baseiam na biblioteca Rtc3231 de Michael Miller. Essa biblioteca disponibiliza também o objeto RtcDateTime que representa uma estrutura de dados para armazenamento de Datas e horários, bem como o objeto RtcTemperature para obtermos a temperatura do RTC .

Obs: Para instalar essa lib, use o próprio **Gerenciador de bibliotecas** da IDE do Arduino no menu **Sketch.** Pesquise por “Makuna RTC”.

A seguir, vamos descrever alguns comandos da library e depois alguns exemplos de código…

**Controlando o status do rtc**

Durante o funcionamento do relógio, muitas vezes precisamos monitorar ou gerenciar seu status. A lib RtcDS3231 disponibiliza os seguintes métodos para isso:

**begin** – Inicia a sincronização do relógio

**IsDateTimeValid** – Retorna um valor booleano indicando se a data está válida.  
A data pode ficar inválida quando há perda de energia da bateria ou porque a data nunca foi setada.

**GetIsRunning** – Retorna um valor booleano indicando se o RTC está ativado, ou seja, está computando as horas.

**SetIsRunning** – Ativa (true) ou desativa (false) o relógio

**Enable32kHzPin** – Habilita/Desabilita o pino nerd com o pulso do oscilador a uma taxa de 32KHz. Nem todas as bibliotecas oferecem essa opção de “baixo nível”

**SetSquareWavePin** – Possibilita configurar o pino de interrupção para usar com os alarmes ou outras funções. existe um método para configurar a frequência desse pino também.

**Setando e obtendo data e hora**

Existem várias ocasiões em que necessitaremos configurar a data e hora do RTC, ou seja, acertar o relógio:

1. Quando ligamos o módulo pela primeira vez;
2. Quando acaba a energia da bateria
3. Horário de verão
4. Etc.

Para isso,usamos o comando **SetDateTime** e passamos para ele um objeto **RtcDateTime** com a data e hora desejados.

Exemplos:

**//Acerta a data em 1 de Dezembro de 2017 as 5 da manhã:  
RtcDateTime data = RtcDateTime(2017,12,1,5,0,0);**  
**Rtc.SetDateTime(data);**

**//Acerta o RTC com a data da compilação do sketch:**  
**RtcDateTime compileDateTime(\_\_DATE\_\_,\_\_TIME\_\_);**  
**Rtc.SetDateTime(compileDateTime);**

Para obter a data e horário atuais, fazemos o contrário:

**RtcDateTime now = Rtc.GetDateTime();**  
**Serial.print(now.day());  
Serial.print(now.month());  
Serial.print(now.year());**

**Trabalhando com Alarmes**

Para trabalharmos com alarmes, precisamos definir uma interrupção do Arduino que será acionada quando o alarme for atingido. Os seguintes métodos estão disponíveis:

**SetSquareWavePin** – Esse método habilita o pino SQW, definindo qual dos dois alarmes iremos utilizar.

**SetAlarmOne e SetAlarmTwo** – Define a data/horário de cada alarme através de um objeto DateTime.

**LatchAlarmsTriggeredFlags** – Método necessário para que os alarmes se efetivem.

**Temperatura**

Para exibirmos a temperatura do RTC, simplesmente fazemos como no exemplo abaixo:

**RtcTemperature temp = Rtc.GetTemperature();**  
**Serial.print(temp.AsFloat());**

**GRAVANDO DADOS NA EEPROM DO MÓDULO**

Nem todos os módulos do DS3231 disponibilizam uma EEPROM e, por isso, nem todas as bibliotecas oferecem funções de gravação de dados. Portanto não vamos abordar aqui esse recurso mas, se quiser ter acesso à memória do módulo, pode usar as bibliotecas abaixo que oferecem até opções para fazer um pequeno datalog:

* [DS3232RT](https://github.com/JChristensen/DS3232RTC" \t "_blank)
* [DS3231\_Simple](https://github.com/sleemanj/DS3231_Simple" \t "_blank)

**Exemplos**

Você pode baixar os exemplos também no [github](https://github.com/JoseCintra/Arduino/tree/master/RTC" \t "_blank).

Exemplo 1: Operações de inicialização e exibição do horário do RTC:

Exemplo simples de inicialização e exibição do horário

<http://josecintra.com/blog/explorando-rtc-ds3231-arduino/>

<https://drive.google.com/file/d/0ByRmxf0qDBMUSVVndGtqekdUOWM/view>

<https://www.flickr.com/photos/jgustavoam/sets/72157663264163678>

<https://learn.adafruit.com/ds3231-precision-rtc-featherwing>