****

**UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE**

**CENTRO DE ENSINO SUPERIOR DO SERIDÓ**

**DEPARTAMENTO DE CIÊNCIAS EXATAS E APLICADAS**

**CURSO DE SISTEMAS DE INFORMAÇÃO**

ALCINDO LOPES DE FARIAS JÚNIOR

ROBERTO DE SOUSA ROCHA

**CONSTRUINDO UM RELÓGIO COM ARDUINO**

Caicó-RN

2018

ALCINDO LOPES DE FARIAS JÚNIOR

ROBERTO DE SOUSA ROCHA

**CONSTRUINDO UM RELÓGIO COM ARDUINO**

Trabalho elaborado ao curso de graduação em Sistemas de Informação, como requisito da primeira avaliação da disciplina **Microcontroladores,** sob a orientação do professor Luiz Paulo de Assis Barbosa, relativo ao período 2018.1.

Caicó-RN

2018

**SUMÁRIO**

[**INTRODUÇÃO**](#_6j882u5eqli) **4**

[**OBJETIVOS**](#_ym1wl0ptp39t) **4**

[OBJETIVO GERAL](#_b6o73slmtro) 4

[OBJETIVOS ESPECÍFICO](#_8u02edakno90) 4

[**REVISÃO BIBLIOGRÁFICA**](#_39g8w7xyk6yh) **4**

[ARDUINO](#_siw1ymm1y0wb) 4

[ARDUINO NANO](#_sqt1kwmc40lo) 5

[RTC DS3231](#_f6rqzkq00zei) 6

[REGISTRADORES](#_o4dbxyxghfyr) 7

[Registrador de tempo](#_k601deduls63) 7

[Registrador de data](#_u1e886eoxt3v) 8

[Registrador de Alarme](#_qs6z5c1a23z) 8

[Registradores de Controle (0x0E) e Status (0x0F)](#_grgu4tg014s2) 9

[Registrador de Temperatura](#_3vhbipfzdxvh) 10

[DISPLAY 16X02 COM MÓDULO I2C](#_mqu353sih5t7) 11

[BUZZER](#_3on5shb81uuw) 12

[**CONFIGURANDO O AMBIENTE**](#_a2op0zb813r) **13**

[INSTALAÇÃO E CONFIGURAÇÃO DA IDE DO ARDUINO](#_talaz10qug) 13

[INSTALANDO NO WINDOWS 10](#_hu7im7ypaald) 13

[CONFIGURANDO A IDE DO ARDUINO PARA ARDUINO NANO](#_fypl46cgoe9l) 15

[BIBLIOTECAS](#_10ljh8nl3vxw) 16

[LiquidCrystal\_I2C](#_pk2mpvrx125m) 16

[Baixando a biblioteca](#_3bnzitkyeg0z) 16

[Evitando conflito com a biblioteca nativa do arduino](#_23erl624daew) 17

[Instalando a biblioteca na IDE do Arduino](#_hz40bhtduemx) 17

[Timer](#_2jgpnh9ia3x) 18

[Wire](#_fxbwshhtr9c0) 18

[**PROJETO**](#_lup40jlvxo2p) **18**

[RTC DS3231](#_ur92daslzaro) 19

[DISPLAY 16x02](#_mf9prquq19x7) 19

[BUZZER](#_3xvhlg7pf2v3) 20

[BOTÕES](#_ubhriopeiuw9) 20

[FUNÇÕES DO CÓDIGO](#_su5cqhp4946n) 20

[Funções de leitura e escrita nos registradores](#_ivm9vsfnumtp) 20

[Funções de entrada de dados](#_ce9ds4jwfqjb) 20

[Funções de formatação das informações para exibição](#_figd7rcg0pk8) 21

[Funções para exibir as informações](#_m76xh2c23acp) 21

[Funções para alterar exibição de informações com botões](#_ek1wkm961bc2) 21

[Funções do alarme](#_a0oi5xhhmwxp) 22

[Funções de conversores](#_a4p7x9z5oy05) 22

[CÓDIGO FONTE](#_qndaplg6alwy) 23

[**REFERÊNCIAS**](#_fzwrbaffwm7b) **23**

# INTRODUÇÃO

Neste projeto mostraremos como desenvolver um relógio utilizando um arduino, um RTC DS3231, um Display 16x2 e um buzzer emissor de efeitos sonoros. O mesmo terá as horas, minutos, segundos, data e alarme como opções para o usuário, exibidos no Display, sendo registrados no RTC e controlados pelo arduino. O buzzer servirá para acionar efeitos sonoros quando o alarme for ativado.

# OBJETIVOS

## OBJETIVO GERAL

O objetivo deste projeto consiste em coletar e inserir dados no módulo RTC DS3231 de acordo com o que o usuário desejar através das opções disponibilizadas. Esse RTC será controlado através de um arduino e as informações e opções exibidas em um display 16x02.

## OBJETIVOS ESPECÍFICO

* Exibir os dados em display 16x02
* Alterar através dos botões a exibição dos dados no display entre temperatura e data
* Alterar através dos botões o formato de exibição de hora (12AM/PM ou 24h)
* Exibir os dados no monitor serial
* Configurar, ativar e desativar o alarme através da comunicação serial
* Configurar o horário através da comunicação serial
* Configurar a data através da comunicação serial

# REVISÃO BIBLIOGRÁFICA

Para se aprofundar neste projeto, devemos primeiramente introduzir alguns conceitos básicos das ferramentas e equipamentos que iremos utilizar no mesmo. Embasado nisso, veremos a seguir o que são arduinos, o arduino nano, o Display 16x2, o RTC DS3231 e o buzzer.

## ARDUINO

Segundo MONK (2013, p. 01), “o arduino é uma pequena placa de microcontrolador contendo um plugue de conexão USB (universal serial bus) que permite a ligação com um computador. Além disso, contém diversos outros terminais que permitem a conexão com dispositivos externos, como motores, relés, sensores luminosos, diodos, laser, alto-falantes e outros. Os arduinos podem ser energizados por um computador através do plugue USB, por uma bateria de 9V ou por uma fonte de alimentação. Eles podem ser controlados diretamente pelo computador, ou então, podem ser programados pelo computador e, em seguida, desconectados, permitindo assim que trabalhem independentemente do computador.”

## ARDUINO NANO

Em termos práticos, os arduinos possuem funcionamento semelhante ao de um pequeno computador, no qual, pode-se programar a maneira como suas entradas e saídas devem se comportar em meio aos diversos componentes externos que podem ser conectados nas mesmas.

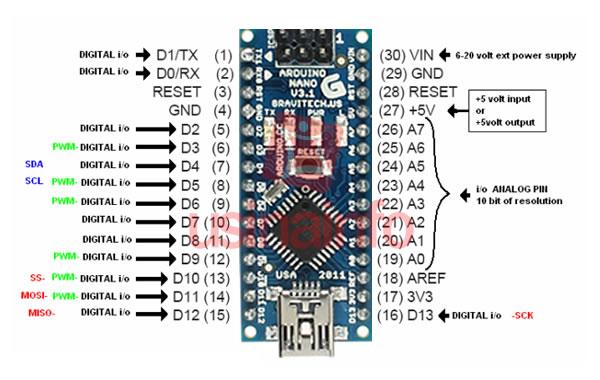


Figura 3.1 - Portas do Arduino Nano (imagem retirada da internet)

O Arduino Nano é uma placa compacta baseada no microcontrolador ATmega328 (para placas Arduino Nano versão 3.x), ou ATmega168 (versão 2.x). Ao contrário das outras placas, não possui conector para alimentação externa, sendo alimentada por um conector USB Mini-B. É uma placa desenvolvida pela Gravitech.

**Especificações:**

– Microcontrolador: ATmega328

– Tensão de Operação: 5V

– Tensão de Entrada: 7-12V

– Portas Digitais: 14 (6 podem ser usadas como PWM)

– Portas Analógicas: 8

– Corrente Pinos I/O: 40mA

– Memória Flash: 32KB (2KB usado no bootloader)

– SRAM: 2KB

– EEPROM: 1KB

– Velocidade do Clock: 16MHz

– Dimensões: 45 x 18mm

## RTC DS3231

O Real Time Clock (RTC) DS3231 é um relógio de tempo real de alta precisão e baixo consumo de energia. Em sua placa vem embutido um sensor de temperatura e um cristal oscilador para melhorar sua exatidão.



Figura 3.2 - Frente e verso do DS3231 (imagem retirada da internet)

**Especificações:**

– Tensão de operação: 3,3-5V

– Chip: DS3231

– Computa segundos, minutos, horas, dias da semana, dias do mês, meses e anos (de 2000 a 2099)

– Sensor de temperatura com ± 3 °C de exatidão.

– Chip de memória: AT24C32 (capacidade de 32K bytes que podem ser usadas como RAM estendida do microcontrolador)

– Interface I2C

– Circuito de detecção de falha de energia

– Consome menos de 500nA no modo bateria com oscilador em funcionamento

– Faixa de temperatura: 0 a 40°C

– Dimensões: 38 x 22 x 14mm

– Peso: 8g

### REGISTRADORES

Todos os ajustes no módulo DS3231 são feitos através dos registradores internos que o módulo possui. Cada registrador é formado por 8 bits, e cada bit é específico para uma determinada função. Há bit que indica se a hora está no formato 12 AM/PM ou 24H, se o alarme está ativado ou se o alarme disparou. Todos os bits de cada registrador em conjunto servem para ajustar tempo (hh:mm:ss), alarme ou data.

### 

#### Registrador de tempo

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ENDEREÇO | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 | FUNÇÃO | ALCANCE |
| 0x00 | 0 | valor dezena do segundo | | | valor unidade do segundo | | | | segundos | 0-59 |
| 0x01 | 0 | valor dezena do minuto | | | valor unidade do minuto | | | | minutos | 0-59 |
| 0x02 | 0 | 12/**24** | **AM**/PM .  20 H | 10 H | valor unidade da hora | | | | horas | 1-12 + AM/PM  00-23 |

Tabela 3.2.1 - Registradores de hora minuto e segundo

O tempo pode ser acessado através de registradores específicos para cada unidade de tempo, sendo o registrador com endereço 0x00 para os segundos, 0x01 para os minutos e 0x02 para as horas, como mostra a tabela 3.2.1.

No registrador 0x00 os bits 0, 1, 2 e 3 representam o valor da unidade podendo ir de 0-9. Os bits 4, 5 e 6 representam o valor da dezena podendo ir de 0 a 5. O bit 7 não é utilizado.

No registrador 0x01 os bits 0, 1, 2 e 3 representam o valor da unidade podendo ir de 0-9. Os bits 4, 5 e 6 representam o valor da dezena, podendo ir de 0 a 5. O bit 7 não é utilizado.

No registrador 0x02 os bits 0, 1, 2 e 3 representam o valor da unidade. Os bits 4, 5 e 6 podem ter dois comportamentos diferentes, um para quando está configurado para usado a hora no formato AM/PM e outro comportamento no formato 24H.representa. No formato 24H os bits 4 e 5 representam o valor da dezena, podendo ir de 0 a 2 . O bit 6 é usado para indicar se o comportamento está AM/PM ou 24H, para usá-lo como 24H, o bit 6 deve está com valor igual a 0. No formato AM/PM o bit 4 representa a dezena, podendo ir de 0 a 1. O bit 5 representa em qual parte do dia está, 0 indica que é AM e 1 indica que é PM. O bit 6 terá valor igual a 1 para indicar que o formato de hora é AM/PM. Em ambos os formatos de hora o bit 7 não é utilizado.

Os registradores de tempo armazenam os valores em codificação BCD (binary-coded-decimal). O BCD codifica o sistema decimal em binário, dos números (decimais) 0 a 9, onde cada número é representado pelo seu equivalente binário. Como mostra a figura 3.3, um exemplo seria o valor 36 que é convertido os dígitos 3 e 6 separadamente para seu valor em binário.

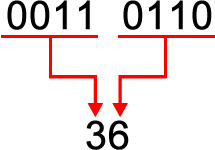


Figura 3.3 - Representação BCD do valor decimal 36

#### Registrador de data

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ENDEREÇO | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 | FUNÇÃO | ALCANCE |
| 0x03 | 0 | 0 | 0 | 0 | 0 | dia | | | dia da semana | 1-7 |
| 0x04 | 0 | 0 | valor dezena do mês | | dia do mês | | | | dia do mês | 01-31 |
| 0x05 | Século | 0 | 0 | 10 mês | mês | | | | data | 1-12 |
| 0x06 | 10 dezena do ano | | | | ano | | | | ano | 00-99 |

Tabela 3.2.2 - Registradores de dia do mês, dia da semana, mês e ano

A data pode ser acessado através de registradores específicos, sendo o registrador 0x03 para dia da semana, 0x04 para dia do mês , 0x05 para o mês e 0x06 para o ano, como mostra a tabela 3.2.2.

No registrador 0x03 apenas os bits 0, 1 e 2 são utilizados, esses bits representam o valor do dia da semana, podendo ir de 1 a 7. Os bits 3, 4, 5, 6 e 7 não são utilizados.

No registrador 0x04 os bits 0, 1, 2 e 3 representam o valor da unidade podendo ir de 0-9. Os bits 4 e 5 representam o valor da dezena, podendo ir de 1 a 2. Os bits 6 e 7 não são utilizados.

No registrador 0x05 os bits 0, 1, 2 e 3 representam o valor da unidade podendo ir de 0-9. O bit 4 representa o valor da dezena, podendo ir de 0 a 1. Os bits 5 e 6 não são utilizados. O bit do século (bit 7 do registro do mês) é alternado entre 0 e 1 quando o registrador de anos altera de 99 para 00

No registrador 0x05 os bits 0, 1, 2 e 3 representam o valor da unidade podendo ir de 0-9. Os bits 4, 5, 6 e 7 representam o valor da dezena, podendo ir de 0 a 9.

#### Registrador de Alarme

O DS3231 contém dois alarmes de hora do dia / data. O alarme 1 pode ser ajustado escrevendo para os registros de 07h a 0Ah. Os alarmes podem ser programados (pela ativação do alarme e pelos bits INTCN do registrador de controle presentes na tabela xx) para ativar a saída INT / SQW em uma condição de correspondência de alarme. O bit 7 de cada um dos registros de alarme de hora do dia / data são bits de máscara (Tabela 3.3). Quando todos os bits de máscara para cada alarme são lógicos 0, um alarme só ocorre quando os valores nos registradores de tempo correspondem aos valores correspondentes armazenados nos registros de alarme de hora do dia / data. Os alarmes também podem ser programados para repetir cada segundo, minuto, hora, dia ou data. A Tabela 3.2.3 mostra as configurações possíveis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DI/**DT** | ALARME 1 CADASTRO DE MÁSCARA (BIT 7)  A1M4 A1M3 A1M2 A1M1 | | | | TAXA DE ALARME |
| X | 1 | 1 | 1 | 1 | Alarme uma vez por segundo |
| X | 1 | 1 | 1 | 0 | Alarme quando os segundos coincidem |
| X | 1 | 1 | 0 | 0 | Alarme quando minutos e segundos coincidem |
| X | 1 | 0 | 0 | 0 | Alarme quando horas, minutos e segundos coincidem |
| 0 | 0 | 0 | 0 | 0 | Alarme quando a data, horas, minutos e segundos coincidem |
| 1 | 0 | 0 | 0 | 0 | Alarme quando o dia, horas, minutos e segundos coincidem |

Tabela 3.2.3 - Taxa de alarme

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ENDEREÇO | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 | FUNÇÃO | ALCANCE |
| 0x07 | A1M1 | 10 segundos | | | segundos | | | | alarme 1 segundos | 0-59 |
| 0x08 | A1M2 | 10 minutos | | | minutos | | | | alarme 2 minutos | 0-59 |
| 0x09 | A1M3 | 12/**24** | **AM**/PM  24 H | AM/PM | hora | | | | alarme 3 hora | 1-12 + AM/PM  00-23 |

Tabela 3.2.4 - Registradores de alarme

Os registradores de alarme 0x07 a 0x09 funcionam igual aos registradores de tempo da tabela 3.4. Exceto pelo bit 7, que é utilizado na taxa de alarme, da tabela 3.2.4

#### Registradores de Controle (0x0E) e Status (0x0F)

O DS3231 possui dois registradores adicionais (controle e status) que controlam os alarmes e saída de onda quadrada. Porém só será explicado o funcionamento dos bits de 0 e 2 do registrador de controle (0x0E) e os bits 0 e 1 do registrador de status (0x0F).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| EOSC | BBSQW | CONV | RS2 | RS1 | INTCN | A2IE | A1IE |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

Tabela 3.2.5 - Bits do registrador de controle

Para ativar o alarme, o bit 0 (A1IE) e o bit 2 (INTCN) devem ser definidos para lógica 1. O bit 0 ativa o ativa o alarme 1 e o bit 2 ativa a interrupção. Quando os dois estão definidos para lógica 1, o bit sinalizador de alarme (A1F do registrador de status) inicia o sinal de interrupção quando os registradores de alarme e os registradores de hora coincidirem.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| OSF | 0 | 0 | 0 | EN32kHz | BSY | A2F | A1F |
| 1 | 0 | 0 | 0 | 1 | X | X | X |

Tabela 3.2.6 - Bits do registrador de status

O bit 0 do registrador de status (0x0F) será responsável por lançar a interrupção. Quando a interrupção é lançada, o bit passa de lógica 0 para 1. Enquanto o bit estiver em lógica 1 novas interrupções não serão lançadas.

#### Registrador de Temperatura

O DS3231 possui um sensor de temperatura com precisão de 0,25ºC e está acessível através dos registradores 0x11 e do registrador 0x12.

No registrador 0x11 está presente a parte inteira da temperatura. Do bits 0 ao ao bit 6 representa o valor da temperatura e o bit 7 representa o sinal para identificar se a temperatura está positiva ou negativa, como mostra a tabela 3.2.7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Sign | data | data | data | data | data | data | data |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabela 3.2.7 - Bits do registrador de temperatura 0x11

O registrador 0x12 representa a parte fracionária da temperatura e só são utilizados os bits 6 e 7. O valor do dos bits estão entre 0 e 3 e representam a precisão de 0,25ºC. Sendo assim, o valor está entre 0 e 75ºC, como mostra a tabela 3.2.8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Data | Data | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabela 3.2.8 - Bits do registrador de temperatura 0x12

## DISPLAY 16X02 COM MÓDULO I2C



Imagem 3.4 - Display 16x02

Para diminuir a quantidade de pinos utilizadas para controlar o display, é utilizado um módulo de I2C que reduz a utilização de 6 pinos para 2 pinos.

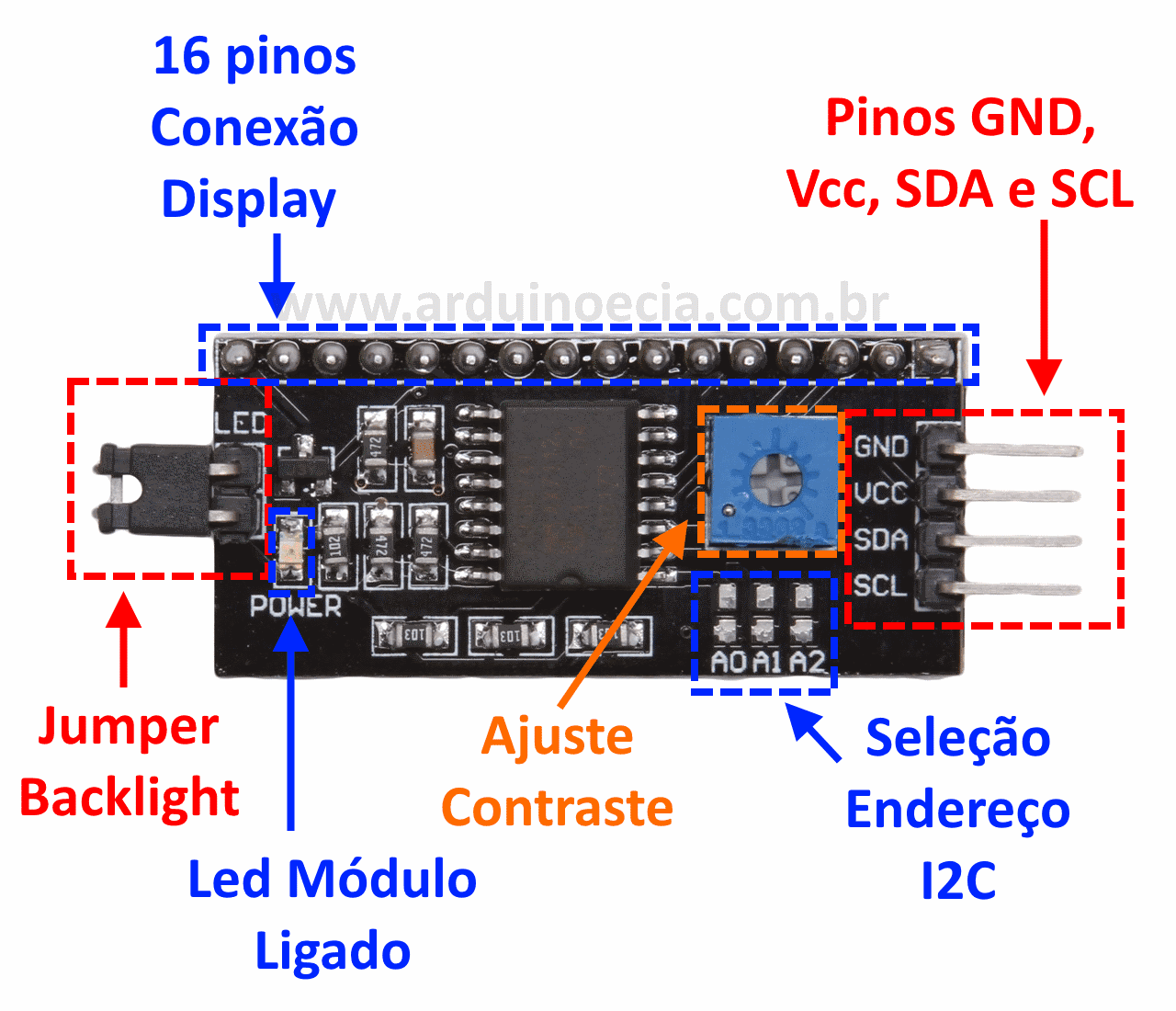


Figura 3.5 - Portas do módulo I2C para display 16x02

## BUZZER

O Buzzer Ativo 5V é um componente indicado para você que precisa adicionar efeitos sonoros em projetos eletrônicos como alarmes, sistemas de sinalização, jogos, brinquedos, etc.



Figura 3.6 - Buzzer

O buzzer do tipo ativo contém um circuito oscilador embutido, assim basta você energizar o componente para que o mesmo comece a emitir um beep contínuo.

**Especificações:**

– Buzzer tipo ativo

– Tensão de operação: 4 à 8VDC

– Corrente de operação: 30mA

– Saída de som mínima (a 10cm): 85dB

– Frequência de ressonância: 2300±300 Hz

– Temperatura de operação: -27 a +70 °C

– Material: ABS

– Cor: Preto

– Dimensões: 11,8 x 9mm

# CONFIGURANDO O AMBIENTE

## INSTALAÇÃO E CONFIGURAÇÃO DA IDE DO ARDUINO

No momento, seu Arduino é somente uma placa com alguns componentes eletrônicos. Para que ela faça algum trabalho útil, você deve dar-lhe instruções, e é por isso que você precisa do IDE do Arduino. O IDE do Arduino fornece tudo o que é necessário para programa-lo, incluindo vários exemplos de programas ou sketches que demonstram como conectá-lo e comunicar-se com alguns dispositivos comuns.

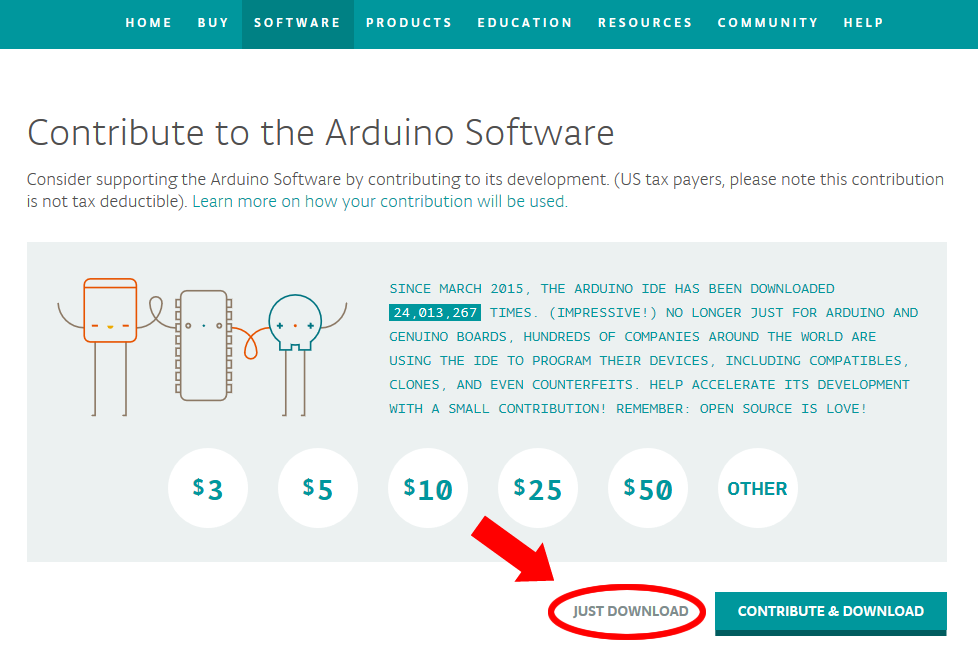
A IDE é de código aberto e pode ser baixada gratuitamente no link (<https://www.arduino.cc/en/Main/Software>). A IDE está disponível para Windows, Mac OS X e Linux.

### INSTALANDO NO WINDOWS 10

A IDE do Arduino para Windows pode ser baixado no formato .zip e .exe. Como mostra na figura 4.1, clique na opção **Windows Installer, for Windows XP and up** para baixar o arquivo .exe

Figura 4.1 – Escolhendo a opção para baixar o instalador da IDE do Arduino no formato .exe.

Em seguida, clique em **JUST DOWNLOAD** para baixar o arquivo, como mostra a figura 4.2.

Figura 4.2 – Clicando no botão para baixar o arquivo do instalador da IDE do Arduino.

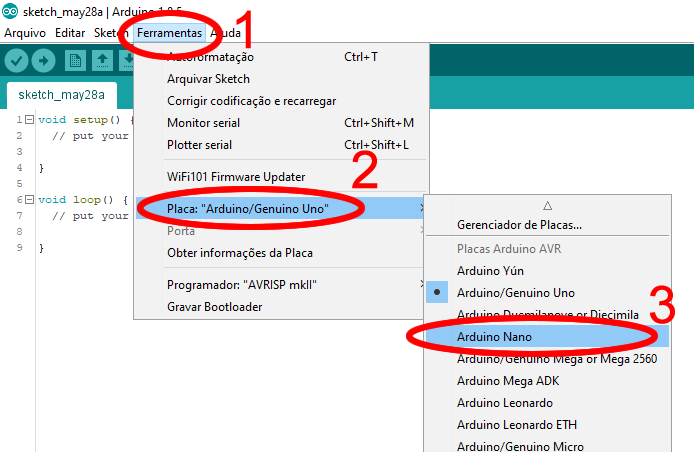
Após o término do download, navegue até a pasta escolhida para armazenar o arquivo baixado (normalmente na pasta Downloads) e dê um duplo clique para realizar a instalação padrão de qualquer programa para Windows (next, next, finish), e pronto, IDE instalada.

### CONFIGURANDO A IDE DO ARDUINO PARA ARDUINO NANO

Com a IDE do Arduino já instalada, e no projeto utilizamos o Arduino Nano, é preciso configurar a IDE para ser utilizada com a nossa placa Arduino Nano.

O passo seguinte após a instalação da IDE é conectar o Arduino Nano ao seu computador por meio de um cabo USB. A alimentação para a placa do Arduino é fornecida pelo cabo USB.

Inicie a aplicação Arduino, clicando no seu ícone - isso vai carregar a IDE do Arduino-. Para trabalhar com a placa Arduino, precisa definir duas configurações no menu. A primeira é o tipo de placa, encontrado em Ferramentas > Placas, como mostrado na figura 4.3.

Figura 4.3 - Selecionando o tipo de placa Arduino Nano

O próximo passo é selecionar a porta serial que conecta o computador a placa Arduino Nano. Isso pode ser encontrado em Ferramentas > Porta, como mostra a figura 4.4.

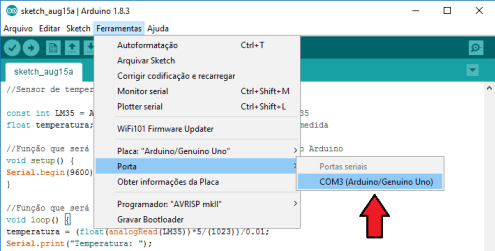
****

Figura 4.4 - Selecionando o tipo de placa Arduino Nano

Após selecionar a porta, a IDE está configurada para funcionar com o Arduino Nano.

## BIBLIOTECAS

### LiquidCrystal\_I2C

Para facilitar a utilização do display, precisamos instalar uma biblioteca que contém várias funções. A IDE já vem com uma biblioteca instalada própria para displays, porém nós precisamos de uma biblioteca para utilizar o display através da comunicação I2C.

#### Baixando a biblioteca

Para baixar a biblioteca, será necessário entrar no link (<https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads/>) e clicar na versão mais atual da biblioteca para que o download se inicie, como mostra a figura 4.5

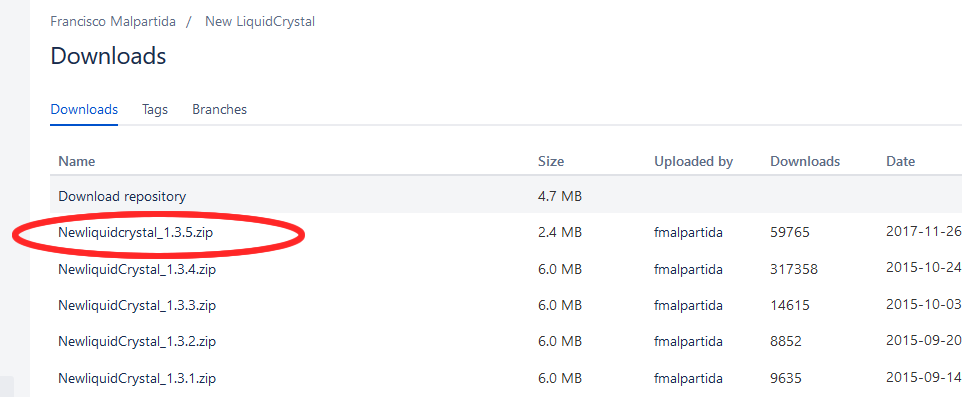


Figura 4.5 - Escolhendo a versão da biblioteca para baixar

#### Evitando conflito com a biblioteca nativa do arduino

Após ter baixado a biblioteca no site, deve-se alterar o nome do arquivo ZIP e também alterar o nome da pasta que está dentro do arquivo ZIP (ambas para um mesmo nome). Considerando que o nome atual pode gerar problemas devido aos pontos entre o número da versão no momento da instalação da biblioteca na IDE do arduino, sugiro renomear o arquivo ZIP e a pasta para LiquidCrystal\_I2C.

#### Instalando a biblioteca na IDE do Arduino

Após renomear o nome do arquivo e da pasta, teremos que importar o arquivo para a IDE do Arduino. Sendo assim, abra a IDE do Arduino, e siga o caminho Sketch > Incluir Biblioteca > Adicionar biblioteca .ZIP, como mostra a figura 4.6.

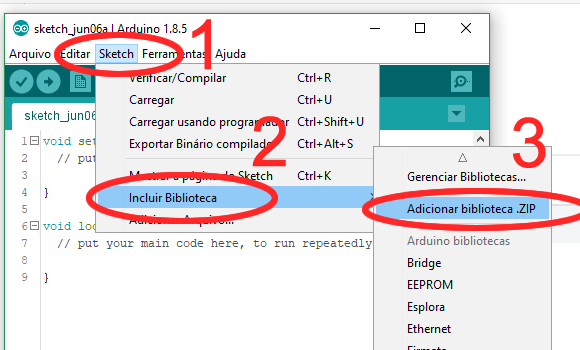


Figura 4.6 - Caminho para incluir a biblioteca

Após clicar em Adicionar biblioteca uma pasta do Windows abrirá e basta seguir até o caminho onde o arquivo foi baixado, clicar no arquivo e no botão abrir.

### Timer

A biblioteca Timer tem funções muito úteis, uma dessas funções foi utilizada no projeto para com a finalidade de executar partes do código em tempo definido pelo programador.

A função **every**(tempo, função) recebe dois parâmetros. O primeiro é o intervalo de tempo que a função será chamada e o segundo parâmetro é a função que será chamada.

A biblioteca não está presente na IDE do Arduino, mas pode ser encontrada em um repositório do github através do link (https://github.com/JChristensen/Timer/).

### Wire

Esta biblioteca que já vem instalada na IDE do Arduino permite que o arduino se comunique com dispositivos I2C. Na placa Arduino Nano a comunicação é feita através dos pinos A4 (SDA) e A5 (SCL).

Para mais informações sobre a biblioteca, a mesma é encontrada no site oficial do arduino através do link: <https://www.arduino.cc/en/reference/wire>.

# PROJETO

Após conhecer os equipamentos e configurar o ambiente, falaremos sobre o projeto realizado.

* 1. **LISTA DE ITENS**

Os itens utilizados no projeto são:

* 1x Arduino Nano
* 1x RTC DS3231
* 1x Display 16x02
* 1x Módulo I2C para Display 16x02
* 2x Botão pull down
* 3x Resistor (1x 10kΩ, 2x 220Ω)
* 1x Buzzer 5V
* 1x Protoboard 830 furos
* 1x Cabo USB de alimentação para Arduino Nano
* Fios Jumpers
  1. **MONTAGEM**

Para que todo o circuito funcione, é preciso fazer as conexões corretamente. A figura 5.1 ilustra como foram feitas as conexões.

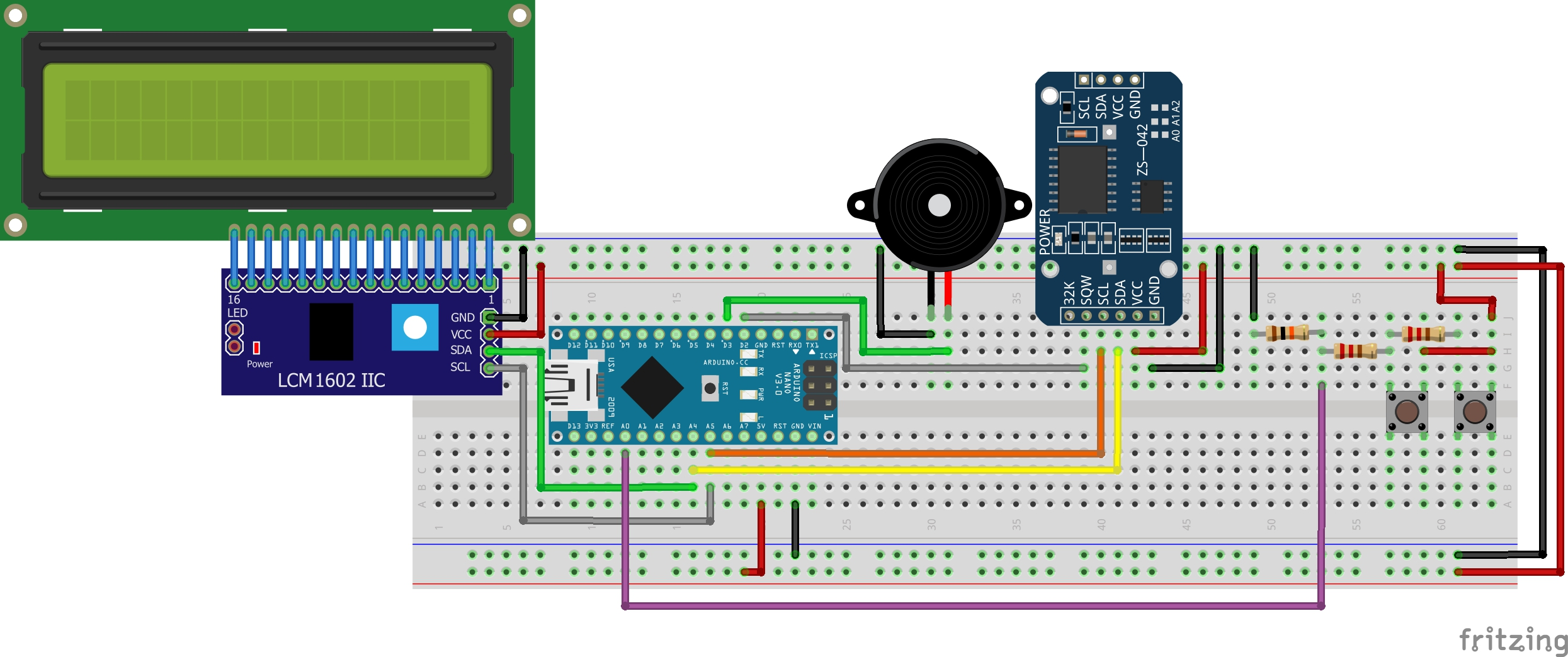


Figura 5.1 - conexão entre os componentes

### RTC DS3231

O RTC DS3231 conta com cinco pinos de conexão que são ligados aos pinos 5V, GND, A5, A4 e D2 do arduino.

|  |  |  |
| --- | --- | --- |
| RTC DS3231 | Arduino Nano | Cor do fio |
| VCC | 5V | Vermelho |
| GND | GND | Preto |
| SCL | A5 | Laranja |
| SDA | A4 | Amarelo |
| SQW | D2 | Cinza |

Tabela 6.1 - ligação entre o RTC e o Arduino

### DISPLAY 16x02

O display 16x02 conta com quatro pinos de conexão que são ligados aos pinos 5V, GND, A5 e A4 do arduino.

|  |  |  |
| --- | --- | --- |
| Display 16x02 | Arduino Nano | Cor do fio |
| VCC | 5V | Vermelho |
| GND | GND | Preto |
| SCL | A5 | Cinza |
| SDA | A4 | Verde |

Tabela 6.2 - ligação entre o Display e o Arduino

### BUZZER

O buzzer conta com apenas dois pinos de conexão que são ligados aos pinos D3 e GND do arduino.

|  |  |
| --- | --- |
| Buzzer | Arduino Nano |
| VCC | D3 |
| GND | GND |

Tabela 6.3 - ligação entre o Buzzer e o Arduino

### BOTÕES

Considerando que o arduino possui poucas portas de entrada e saída, a conexão com os botões foi feita de forma que apenas uma porta analogica seja utilizada. Os botões servem para alterar as opções, e nesse projeto utilizamos dois botões, sendo que para diferenciá-los, os mesmos passam através de resistores, onde um botão passa por somente um resistor e o outro passa por 2 resistores, diferenciando o valor recebido pela porta analógica do arduino.

## FUNÇÕES DO CÓDIGO

### Funções de leitura e escrita nos registradores

As funções que se comunicam diretamente com os registradores são compostas por duas funções, uma de leitura e outra de escrita

* **readDS3231Reg()** é a função responsável por retornar o valor do registrador. Ela recebe por parametro o numero do registrador que se quer obter o valor.
* **writeDS3231Reg()** é a função responsável por escrever algum valor espesifico no registrador. Ela recebe por parametro o numero do registrador que se quer escrever e o valor da escrita.

### Funções de entrada de dados

A configuração do relógio é feita através da entrada serial, e pra isso foram usadas quatro funções, sendo elas para: receber os dados, dividir e realizar as tarefas definidas.

* **receberValor()** recebe a entrada pela serial e envia para a função dividirString().
* **dividirString()** divide os dados recebidos e envia por partes para a função setTempo()
* **setTempo()** Cada parte dos dados recebidos são verificadas e realizam as ações a quais foram configuradas.
* **converte()** serve para ajustrar a entrada quando é para alterar o valor da hora. Essa função converte a entrada de dados para o formato 24h ou am/pm, dependendo de como o relógio esteja configurado para exibir a hora.

### Funções de formatação das informações para exibição

Foram criadas quatro funções para formatação de exibição da temperatura, tempo, data e alarme.

* **getTemperatura()** retorna uma string formatada da temperatura em graus celsius.
* **getTempo()** retorna uma string formatada do tempo atual, separando hora, minuto e segundo pelo caractere especial : (dois pontos).
* **getData()** retorna uma string formatada da data atual, separando dia, mês e ano com barra (/).
* **getAlarme()** retorna uma string formatada do tempo configurado no alarme, separando hora, minuto e segundo pelo caractere especial : (dois pontos).

### Funções para exibir as informações

Para exibir informações no display e monitor serial, são usadas 3 funções:

* **exibir()** é uma função que foi programada para ser executada 1 vez por segunda, o proposito dela é que todas as funções de exibição sejam executadas quando ela for chamada.
* **ExibirSerial()** função executada quando a função exibir() é chamada, ela exibe no monitor serial todas as informações de temperatura, tempo, data e alarme.
* **exibirDisplay()** função executada quando a função exibir() é chamada, ela exibe no display todas as informações de temperatura, tempo, data e alarme.

### Funções para alterar exibição de informações com botões

É possível alterar a exibição entre data e temperatura, formato am/pm para 24h através dos botões. Para isso quatro funções foram criadas.

* **botaoPrecionado()** esta função identifica quando um botão foi pressionado e chama a função identificaBotaoPrecionado().
* **identificaBotaoPrecionado()** esta função identifica qual botão foi pressionado, e chama a função configurada para cada botão.
* **altExibicaoTemperaturaData()** altera a exibição entre temperatura e data.
* **altFormatoDeHora()** altera a exibição da hora entre o formato 24h e am/pm.

### Funções do alarme

No código também há funções para ativar e desativar o alarme, ao todo são seis funções destinadas a controlar o alarme.

* **verificarAlarmeAtivo()** esta função é responsável por verificar se o alarme está ativado para disparar no tempo programado. Ela retorna true caso o alarme esteja ativado ou retorna false caso ele esteja desativado.
* **alarmando()** é uma função que foi programada para ser executada 1 vez por segunda, caso o alarme tenha sido acionado, ela é responsável por acionar o buzzer e piscar a luz do display.
* **ativarAlarme()** função responsável por ativar ou desativar o alarme, de forma que caso ativado, o alarme dispara quando estiver no tempo programado. um ícone é exibido no display para identificar quando o alarme está ativado ou não.
* **gatilhoAlarme()** função executada quando o arduino recebe uma interrupção no pino D2. responsável por informar que o alarme deve ser disparado.
* **clearAlarme1()** função responsável por limpar o registrador do rtc quando o alarme é disparado.
* **periodoDoAlarme()** função responsável por configurar o período quem o alarme deve executar. Ou seja, se o alarme deve disparar a cada segundo, uma vez por minuto, por hora ou por dia, com base na tabela 3.4.

### Funções de conversores

Há funções destinadas a alterar a hora do formato 24h para o formato am/pm e vice versa, como também a funções para converter do padrão decimal para bcd e do padrão bcd para decimal

* **de12para24()** converte a hora do padrão am/pm para o padrão 24h.
* **de24para12()** converte a hora do padrão 24h para o padrão am/pm.
* **decToBcd()** converte os valores decimais para binários BCD.
* **bcdToDec()** converte os valores binários BCD para decimais.

## CÓDIGO FONTE

O código fonte do projeto pode ser encontrado em um repositório no github através do link (https://github.com/jrrf/projeto1\_display\_rtc).

# REFERÊNCIAS

Real Time Clock RTC DS3231 – FilipeFlop. Disponível em: <https://www.filipeflop.com/produto/real-time-clock-rtc-ds3231/> Acesso em: 20/05/2018

Codificação binária decimal – Wikipédia, a enciclopédia livre Disponivel em: <<https://pt.wikipedia.org/wiki/Codifica%C3%A7%C3%A3o_bin%C3%A1ria_decimal>> Acesso em: 14/06/2018

MONK, Simon. Programação com Arduino: começando com Sketches; tradução de Anatólio Laschuk. 1. ed. Porto Alegre: Bookmant, 2013.