Instituto Federal Fluminense Campus Quissamã
Curso Integrado Informática
Professor: Renato
Turma: 2° Ano de Informática
Aluno: Ruan Silva de Carvalho
Assunto: Relatório

O que é

O Módulo I2C é um barramento serial multimestre desenvolvido pela Philips que é usado para conectar periféricos de baixa velocidade a uma placa mãe, a um sistema embarcado ou a um telefone celular. É ideal para ser utilizado em projetos envolvendo LCDs, podendo estar presente em projetos com Arduino ou outros Microcontroladores que tenham suporte ao protocolo I2C onde há diversos dispositivos que comunicarão entre si com apenas duas linhas de dados. Alguns dos elementos mais importantes de um sistema dotado de dispositivos que possuem a necessidade de interagir entre si para que um determinado resultado possa ser obtido, pois, estes possibilitam a troca de informações entre os mesmos através do estabelecimento de regras, padrões e modelos que visam permitir a conexão entre os dispositivos.

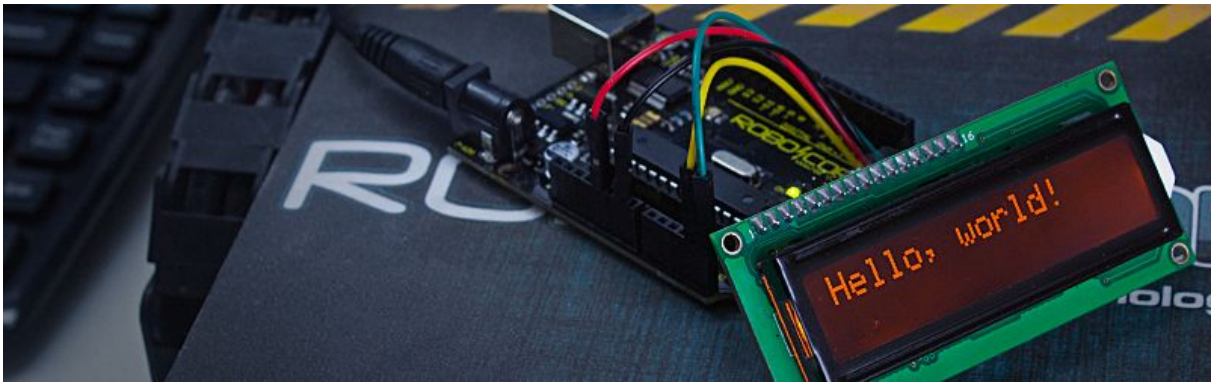


Figura 1: https://www.robocore.net/upload/tutoriais/36_header_H.png

Para que serve

A ideia principal é definir um endereço hexadecimal para cada dispositivo e no momento de comunicação somente o dispositivo solicitado responderá.
DIP SWITCH: Define o endereço do seu dispositivo (podendo ir de 0x20 a 0x27).
Trimpot (potenciômetro): Para quantificar a luminosidade do Backlight do seu Display LCD.
Conectores Latch (ou IDC): Utilizando Jumpers M/F ou F/F você conectar o módulo tanto com seu Mestre quanto a seus Slaves.

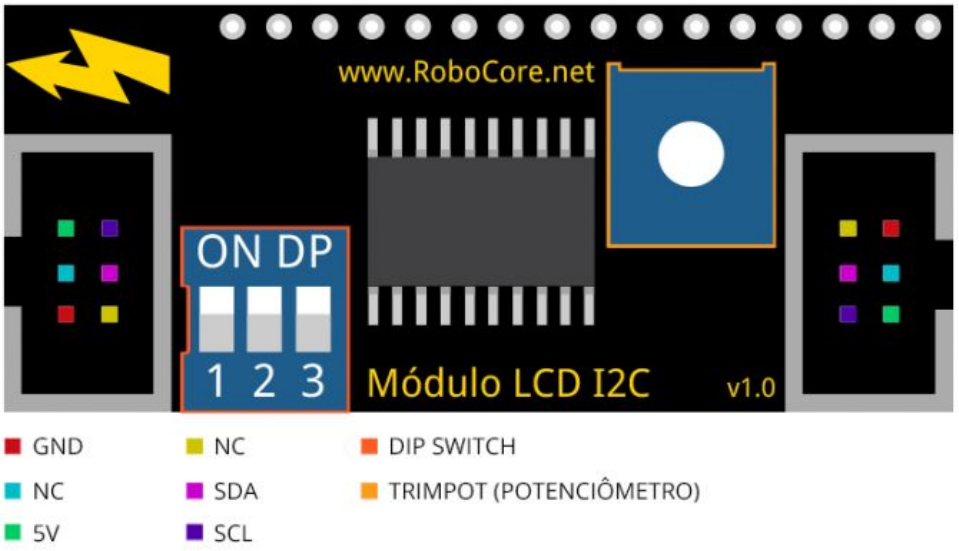


Figura 2: https://www.robocore.net/upload/tutoriais/36_img_3_M.png?222

Exemplo de Uso

Siga estas etapas para conectar dois Arduino UNOs usando o I2C:
Conecte os pinos A4 e A5 em um Arduino aos mesmos pinos do outro.
A linha GND tem que ser comum para ambos os Arduinos. Conecte-o com um jumper.

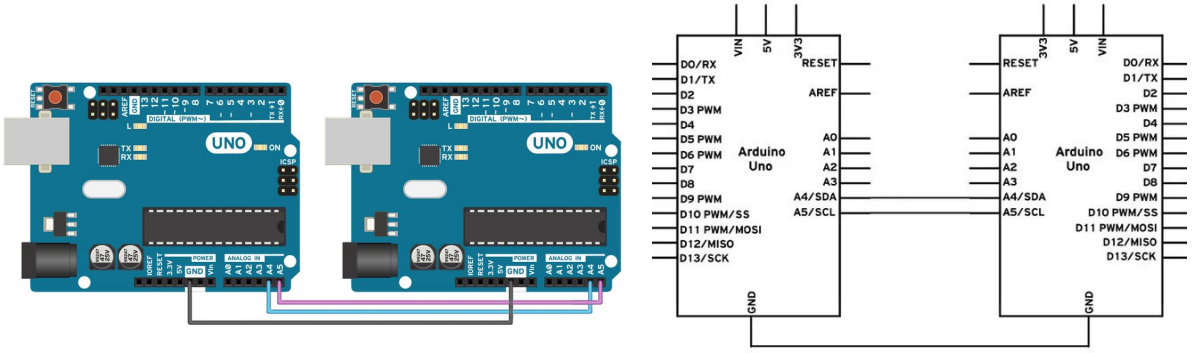


Figura 3: https://cdn.instructables.com/ORIG/FUW/HR26/IBL61ZFK/FUWHR26IBL61ZFK.png?auto=webp&width=1024&fit=bounds
Figura 4: https://cdn.instructables.com/ORIG/FCK/O6PG/IBL61ZB9/FCKO6PGIBL61ZB9.png?auto=webp&fit=bounds

Exemplo Código

O código a seguir é dividido em duas partes: o código mestre e o código escravo , que são executados em dois Arduinos diferentes. Primeiro, vamos dar uma olhada no código mestre:

Código Mestre:

```
// Incluir a biblioteca Wire necessária para o I2C <br> #include
int x = 0;
void setup () {
  // Iniciar o barramento I2C como mestre
```

```
  Wire.begin ();
}
void loop () {
  Wire.beginTransmission (9); // transmitir para o dispositivo # 9
  Wire.write (x); // envia x
  Wire.endTransmission (); // pare de transmitir
  x ++; // Incremento x
  if (x> 5) x = 0; // `resetar x uma vez que fique 6
  atraso (500);
}
```

Código Escravo:

```
// Incluir a biblioteca de cabos necessária para o I2C <br> #include <Wire.h>
int LED = 13;
int x = 0;
void setup () {
  // Definir o pino do LED como saída
  pinMode (LED, OUTPUT);
  // Iniciar o barramento I2C como escravo no endereço 9
  Wire.begin (9);
  // Anexe uma função para disparar quando algo é recebido.
  Wire.onReceive (receiveEvent);
}
void receiveEvent (int bytes) {
  x = Wire.read (); // leia um caractere do I2C
}
void loop () {
  // Se o valor recebido for 0 LED intermitente por 200 ms
  if (x == '0') {
    digitalWrite (LED, ALTO);
    atraso (200);
    digitalWrite (LED, BAIXO);
    atraso (200);
  }
  // Se o valor recebido for 3 LED piscando por 400 ms
  if (x == '3') {
    digitalWrite (LED, ALTO);
    atraso (400);
    digitalWrite (LED, BAIXO);
    atraso (400);
  }
}
```

Ligação do módulo I2C ao display LCD:

```
// Programa : Display LCD 16x2 e modulo I2C
// Autor : Arduino e Cia

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
// Inicializa o display no endereco 0x27
LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7,3, POSITIVE);

void setup()
{
 lcd.begin (16,2);
}

void loop()
{
  lcd.setBacklight(HIGH);
  lcd.setCursor(0,0);
  lcd.print("Arduino e Cia !!");
  lcd.setCursor(0,1);
  lcd.print("LCD e modulo I2C");
  delay(1000);
  lcd.setBacklight(LOW);
  delay(1000);
}
```

Prática

Desenvolvimento e Teste de um Monitor de Barramento I2C para Proteção Contra Falhas Transientes

A comunicação entre circuitos integrados tem evoluído em desempenho e confiabilidade ao longo dos anos. Inicialmente os projetos utilizam barramentos paralelos, onde existe a necessidade de uma grande quantidade de vias, utilizando muitos pinos de entrada e saída dos circuitos integrados resultando também em uma grande suscetibilidade a interferências eletromagnéticas (EMI) e descargas eletrostáticas (ESD). Na sequência, ficou claro que o modelo de barramento serial possuía ampla vantagem em relação ao predecessor, uma vez que este utiliza um menor número de vias, facilitando o processo de leiaute de placas, facilitando também a integridade de sinais possibilitando velocidades muito maiores apesar do menor número de vias.

Este trabalho faz uma comparação entre os principais protocolos seriais de baixa e média velocidade. Nessa pesquisa, foram salientadas as características positivas e negativas de cada protocolo, e como resultado o enquadramento de cada um dos protocolos em um segmento de atuação mais apropriado.

O objetivo deste trabalho é utilizar o resultado da análise comparativa dos protocolos seriais para propor um aparato de hardware capaz de suprir uma deficiência encontrada no protocolo serial I2C, amplamente utilizado na indústria, mas que possui restrições quando a aplicação necessita alta confiabilidade. O aparato, aqui chamado de Monitor de Barramento I2C, é capaz de verificar a integridade de dados, sinalizar métricas sobre a qualidade das comunicações, detectar falhas transitórias e erros permanentes no barramento e agir sobre os dispositivos conectados ao barramento para a recuperação de tais erros, evitando falhas. Foi desenvolvido um mecanismo de injeção de falhas para simular as falhas em dispositivos conectados ao barramento e, portanto, verificar a resposta do monitor. Resultados no PSoC5, da empresa Cypress, mostram que a solução proposta tem um baixo custo em termos de área e nenhum impacto no desempenho das comunicações.

Barramento I2C (MIC098)

Com o objetivo de facilitar a comunicação entre os circuitos em controles industriais, equipamentos de consumo e muitos outros tipos de aplicativos, a Philips lançou em 1992 um novo tipo de barramento que permitiu o uso de dois fios para a troca de informações de forma eficiente. O novo barramento, denominado Inter IC ou I2C, possibilita o uso de uma enorme variedade de componentes padronizados, que podem trocar dados de forma simples e eficiente. Tal foi o sucesso desse barramento que em pouco tempo ele se tornou um padrão mundial, um dos preferidos para as aplicações que envolvem todo o tipo de troca de dados entre circuitos utilizando cabos trançados. Esse sucesso levou ao aparecimento da versão 2.0, em 1998. O que é o I2C e como ele funciona, é o que veremos neste artigo de 2001. A ideia básica do uso do barramento I2C é permitir a fácil integração em componentes de interface, que possam trabalhar diretamente com seus sinais simplificando assim a comunicação com outros componentes da mesma família. Na época que a Philips lançou este barramento, mais de 150 circuitos integrados ICMOS já eram compatíveis com ele.

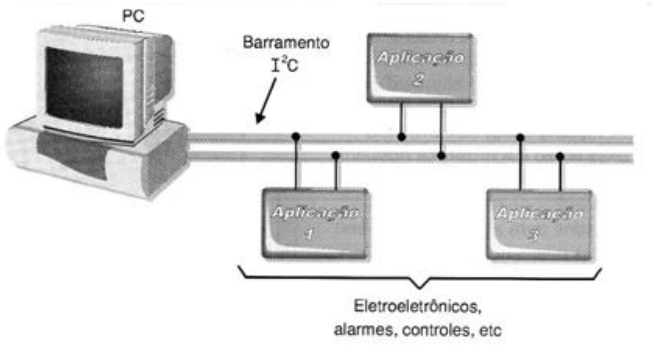


Figura 5: https://www.newtoncbraga.com.br/images/stories/microcontroladores/mic0098_0001.png

Para que os circuitos possam se comunicar de maneira eficiente, dentro da ideia básica do barramento I2C foi previsto que eles deveriam apresentar as seguintes características básicas: Empregar apenas dois fios: uma serial data line (SDA) e uma serial clock line (SCL). Cada dispositivo que fosse conectado ao barramento deveria ser endereçado por software com um único endereço, e relações simples entre mestre/escravo deveriam estar disponíveis o tempo todo. No tipo final foram usados 10 bits para esse endereçamento. Deveriam ser previstos recursos para detectar colisão de dados de modo a não haver dano aos dados transferidos em caso de transmissão simultânea. Seriam utilizadas inicialmente velocidades de transferência de 100 kbits/s no modo standard ou 400 kbit/s no modo rápido. Posteriormente, o low speed mode foi omitido. Deveriam ser previstos circuitos de rejeição de picos e transientes on-chip para preservar a integridade dos dados. O número de Cls conectados a um mesmo barramento seria limitado apenas pela capacitância máxima do barramento, de 400 pF. A adoção deste padrão poderia facilitar em muito o projeto de equipamentos que precisam se comunicar com outros de forma simples e eficiente, uma vez que não seriam necessárias interfaces adicionais. Mais do que isso, componentes de todos os tipos poderiam incorporar os circuitos de interface, o que significa que para interliga-los bastaria usar um par trançado sem a necessidade de circuitos de interface adicionais.

Características e Vantagens

Vantagens que podem ser atribuídas ao protocolo I2C. Destacam-se entre elas:

Organização funcional em blocos, providenciando um simples diagrama esquemático final.

Não há necessidade dos projetistas desenvolverem interfaces. Todos os dispositivos integram as interfaces “on-chip”, o que aumenta a agilidade no desenvolvimento.

Endereçamento e protocolo de transferência de dados totalmente definido via software.

Possibilidade de inclusão ou exclusão de dispositivos no barramento sem afetá-lo ou outros dispositivos conectados a este.

Diagnóstico de falhas extremamente simples. O mal funcionamento é imediatamente detectado.

Desenvolvimento simplificado do software através do uso de bibliotecas e módulos de software reutilizáveis.

Facilidade no desenvolvimento de placas de circuito impresso, devido a quantidade de interconexões.

Adicionalmente, utilizando as vantagens da tecnologia CMOS na fabricação dos dispositivos, temos:

Baixíssimo consumo de corrente.

Alta imunidade à ruídos.

Ampla faixa de tensões p/ alimentação.

Ampla faixa de temperatura p/ operação.

Características Gerais do Barramento I2C:

Suporta qualquer tecnologia de produção.

Duas vias de comunicação: serial data (SDA) e serial clock (SCL), ambas bidirecionais, conectadas ao positivo da fonte de alimentação através de um resistor de pull-up. Enquanto o barramento está livre ambas as linhas ficam em nível lógico alto.

A taxa de transferência máxima é de 100kbit/s no modo padrão (standart), ou 400kbit/s no modo rápido (fastmode).

Informação de carry entre dispositivos conectados.

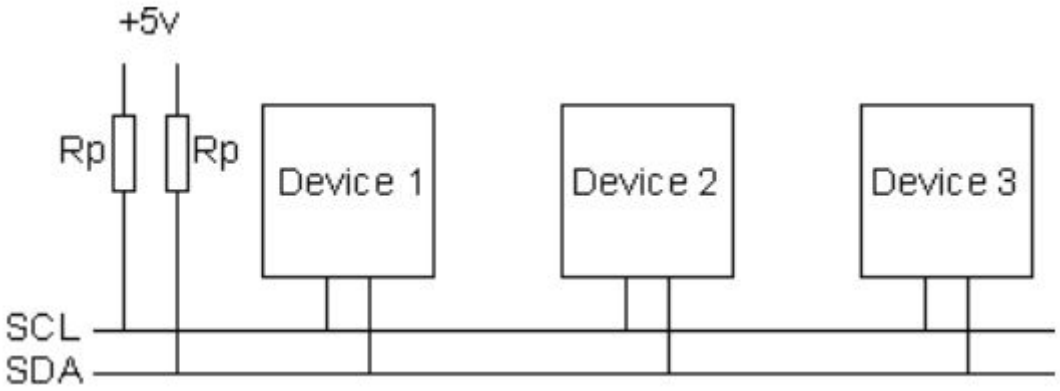
Todo dispositivo possui um endereço único no barramento, independente de sua natureza.

Qualquer dispositivo conectado pode operar com transmissor ou receptor. Claro que isso depende da natureza do dispositivo – um LCD não vai operar como transmissor, assim como um teclado não operará como receptor. Independente disso, qualquer dispositivo *endereçado* é chamado de escravo (slave).

O número de interfaces conectadas fica dependente da capacitância máxima do barramento, que é de 400pF.

Hardware:

O barramento I2C é composto de dois fios, SDA e SCL, e alimentação (VDD), tipicamente de 3.3V ou 5V. Os fios de comunicação possuem pull-ups, como pode ser visto na figura abaixo:

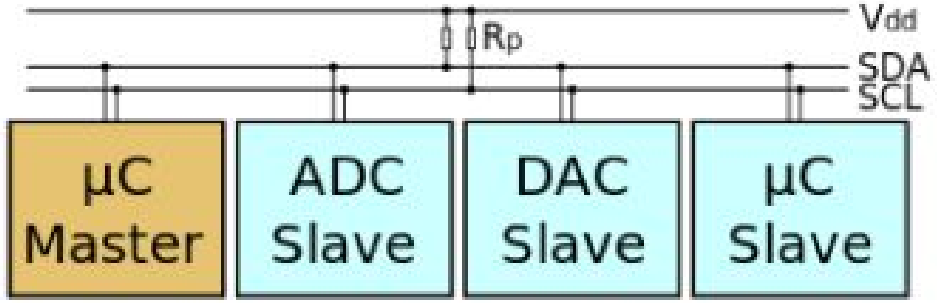


O número de “nós” em um único barramento é limitado tanto pelo tamanho do endereço, que pode ser de 7 bits, 10 bits e até 16 bits; como por restrição de espaço, já que não se pode ultrapassar poucos metros de fios, pois a capacitância total máxima, algo em torno de 400pf, impede o funcionamento correto do barramento.

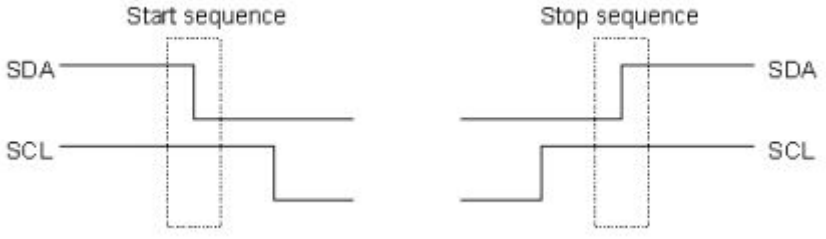
Software:

O protocolo I2C tem dois tipos de dispositivos: **Master** e **Slave**. Onde o Master (mestre em inglês), é a unidade de controle responsável por coordenar todos os periféricos (Slaves, escravos em inglês).

A linha SCL é responsável pelo clock do barramento, e a linha **SDA** pela transmissão de dados.

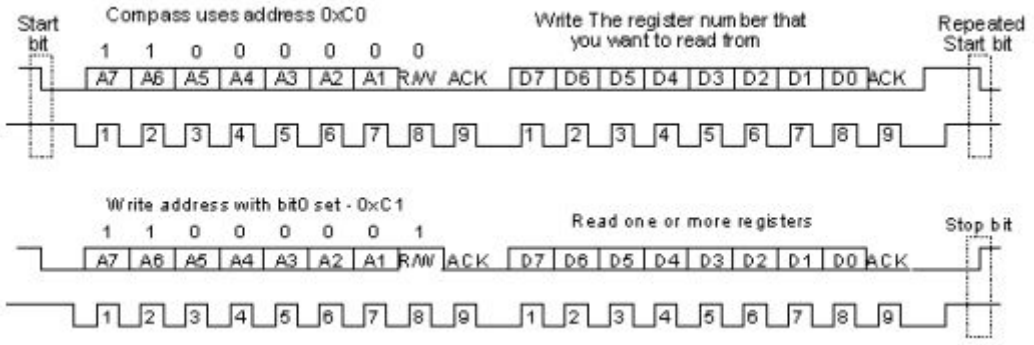


Como se pode perceber, no estado neutro do barramento I2C são mantidos o valor digital **alto** em ambas as linhas de comunicação, para se iniciar a comunicação, SDA é trazido para o valor digital **baixo** pelo mestre. Para escrever dados no barramento, SCL pulsa, e a cada pulso, o valor em SDA é lido como um bit, começando do MSB.



Logo após SDA ser trazida pra baixo, o mestre escreve o endereço do dispositivo que ele deseja se comunicar, por exemplo 0xC0, caso o dispositivo exista, ele responderá como um ACK, um

pulso na linha SCL. Então começa a transferência de dados, o mestre escreve o endereço do registrador no escravo que ele deseja ler ou escrever (R/W) e opera então, em sequência, podendo ler/escrever um ou mais registrador.



OBS.: WATCHDOG TIMER

Trata-se de um timer que, uma vez ligado, precisa ser rearmado periodicamente. Se ele não for rearmado dentro de um tempo limite, o microcontrolador é automaticamente reiniciado.

Para usar o watchdog precisamos descer um nível abaixo da biblioteca do Arduino, usando a biblioteca do compilador C. Por este motivo, as informações abaixo se aplicam somente aos modelos baseados nos microprocessadores AVR (ATmega e ATtinny). No seu programa você precisa incluir o header wdt.h:

```
#include <avr/wdt.h>
```

Para ligar o watchdog, chame a função wdt_enable, informando o tempo limite:

```
wdt_enable(WDTO_2S);
```

O valor no exemplo acima corresponde a 2 segundos; estão também disponíveis WDTO_8S, WDTO_4S, WDTO_1S, WDTO_500MS, WDTO_250MS,WDTO_120MS, WDTO_60MS, WDTO_30MS e WDTO_15MS. A temporização do watchdog é fornecida por um oscilador interno ao microcontrolador e independe do clock de execução das instruções.

Para rearmar o watchdog, basta chamar wdt_reset(). Normalmente você irá colocar chamadas nos pontos normais de execução, para garantir que não ocorra o reset durante o funcionamento correto.

O exemplo abaixo permite ver o watchdog em funcionamento, para rodá-lo conecte um botão entre o pino 3 e GND. Se você mantiver o botão apertado por mais de 2 segundos o Arduino reiniciará.


```
#include <avr/wdt.h>

const int pinBotao = 3;
const int pinLED = 13;

void setupt() {
  wdt_enable(WDTO_2S);
  pinMode (pinBotao, INPUT_PULLUP);
  pinMode (pinLED, OUTPUT);
  Serial.begin(9600);
```

```
  Serial.println("Reset");
  digitalWrite (pinLED, HIGH);  // para indicar o reset
  delay (500);
  digitalWrite (pinLED, LOW);
}

void loop() {
  wdt_reset();
  while (digitalRead(pinBotao) == LOW) {
    delay (10);
  }
}
```

Referências Bibliográficas:

https://www.arduinoecia.com.br/modulo-i2c-display-16x2-arduino/
https://www.robocore.net/tutoriais/primeiros-passos-com-modulo-i2c.html
https://portal.vidadesilicio.com.br/i2c-comunicacao-entre-arduinos/
https://www.instructables.com/id/I2C-between-Arduinos/
https://www.lume.ufrgs.br/bitstream/handle/10183/150164/001008274.pdf?sequence=1
https://www.newtoncbraga.com.br/index.php/microcontrolador/143-tecnologia/12085-conheca-o-barramento-i2c-mic098
http://www3.eletronica.org/artigos/protocolo-de-comunicacao-i%C2%B2c
http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C.pdf
https://www.filipeflop.com/blog/wachdog-e-eeprom-do-arduino/