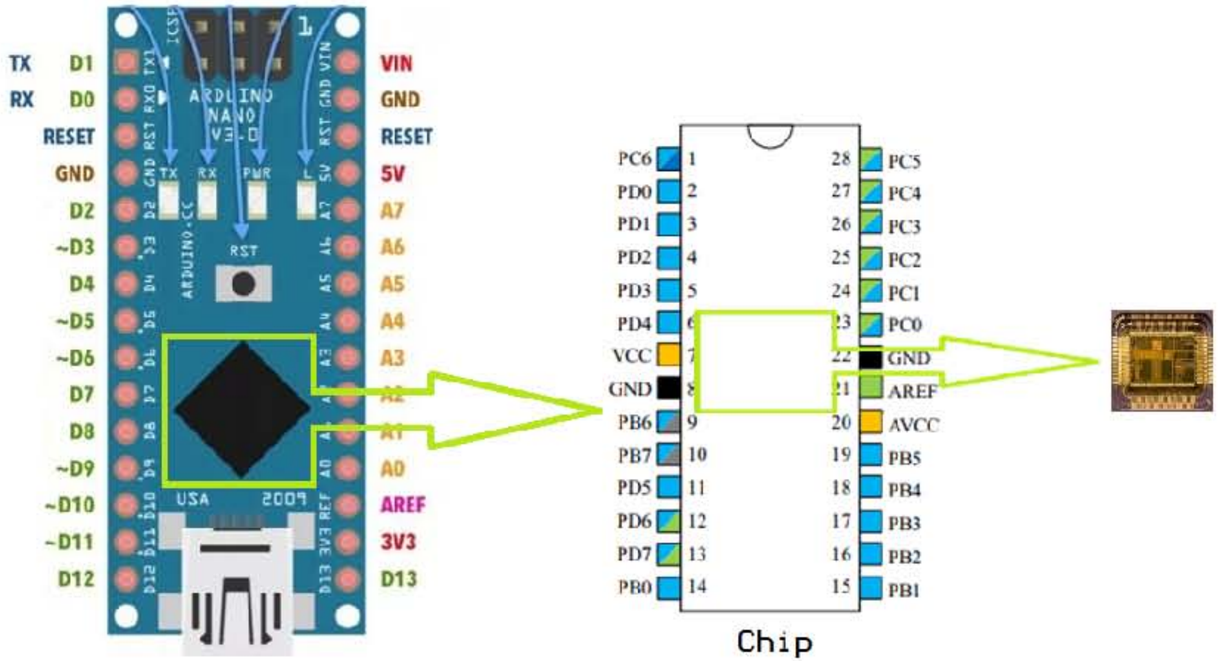
EEPROM NO NANO

Electrically Erasable Programable Read Only Memory



Plataforma

Chip

© Prof. Engº esp Luiz Antonio Vargas Pinto
www.vargasp.com
Revisado 09/2023

© Prof. Vargas

Características

- A família Microchip, assim como a grande maioria dos microcontroladores, trazem em seu interior uma quantidade de memória EEPROM - como a base do Arduino é o ATMEGA328p, as características são as mesmas.

- O Arduino nano possui 1024 endereços de EEPROM
  - De 0x000 a 0x3FF
- Armazena um byte em cada endereço

© Prof. Vargas

Sintaxe

```
EEPROM.write(endereco,dado)
```

- Cuidado pois endereço é **int** e dado é **byte**
  - ☒ **byte** ocupa 1 byte e cabe de **0** a **255**
  - ☒ **int** ocupa 2 bytes e cabe de **0** a **65535**

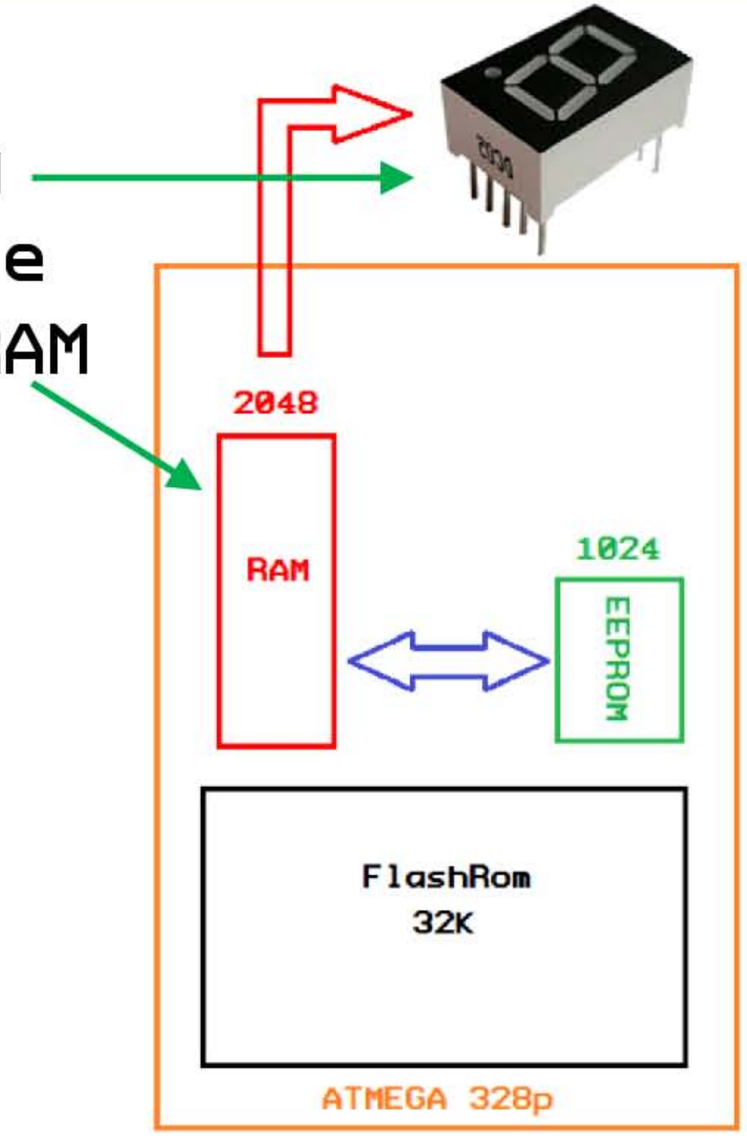
```
Destino = EEPROM.read(endereco)
```

- Destino (**byte** ou **int**)
- Endereço deve ser **int**

© Prof. Vargas

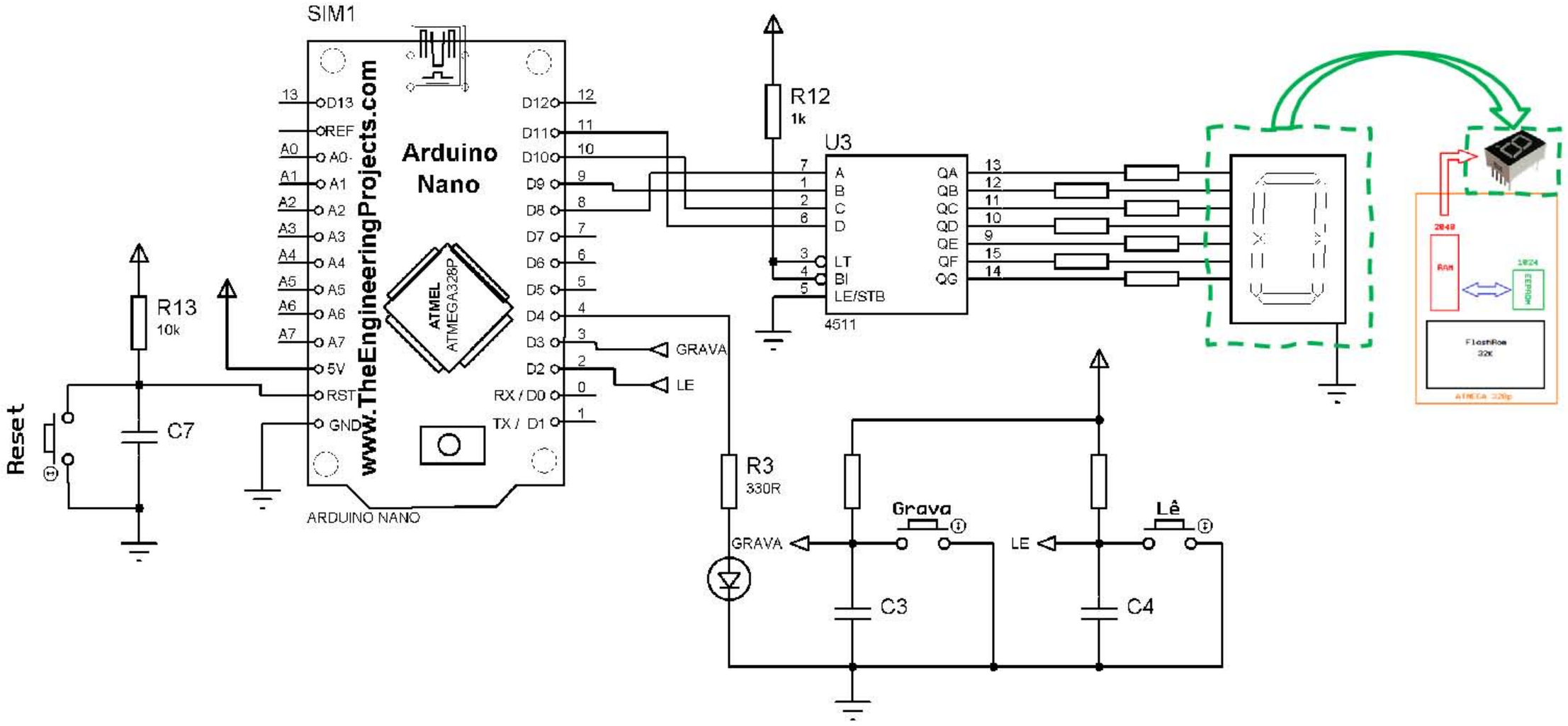
Proposta para teste da EEPROM

- O diagrama explica que o display mostra apenas o conteúdo da RAM e que há uma comunicação entre a RAM e a EEPROM



© Prof. Vargas

Esquema de teste



© Prof. Vargas

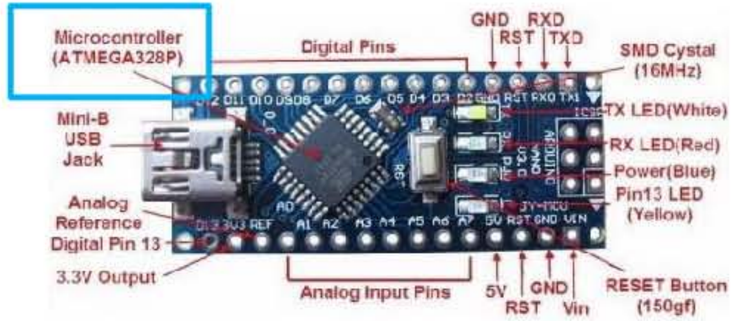
Programa

```
/*-----------------------------------------------------------------
  Programa : EEPROM1.ino
  Autor    : Prof. Vargas
  Data     : 20/03/2023

  Descrição:
  Testa o uso da EEPROM interna do ARDUINO que em última instância é a
  do AVR que inicia em 0x000 e vai até 0x3FF.  Este programa apresenta
  no display de 7 segmentos o valor contido nos endereços do vetor de
  dados de A[0]a A[9] que são RAM. Uma sequência de dados consecutivos
  de 0 a 9 são gravados na EEPROM nos endereços  0x00 a 0x09  para que
  se possa ter a percepção da alteração ao se ler a EEPROM.
    O botão LE faz com que o microcontrolador pegue os dados da EEPROM
  e os coloque na RAM.
    O botão GRAVA preenche a EEPROM com 0s e 1s alternadamente, os quais
  podemos ler com o botão LE e que serão mostrados no display de 7 seg.
  -----------------------------------------------------------------*/
#include <EEPROM.h>
const byte Grava = 3;     //Batizamos o port de I/O digital 3 de Grava
const byte Le = 2;        //Batizamos o port de I/O digital 2 de Le
const byte LED_Grava = 4; //Batizamos o port de I/O digital 4 de LED_Grava
byte Botao,Aux;
char k, endereco, dado, result;
char A[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

```

© Prof. Vargas

```
26 void setup()
27 {
28   pinMode(Grava, INPUT); //Definimos Led como Output.
29   pinMode(Le, INPUT);    //Definimos Switch como Input.
30   pinMode(LED_Grava,OUTPUT);
31   PORTB = 0x00;          // 0 para o display
32   DDRB  = 0b11111111;    // Port imita o Hardware
33
34   //preenche EEPROM original sequencial
35   digitalWrite(LED_Grava,HIGH);
36   for (endereco = 0; endereco < 10; endereco++)
37   {
38     EEPROM.write(endereco, endereco);
39   }
40   delay(200);
41   digitalWrite(LED_Grava,LOW);
42
43   //preenche dados visuais
44   A[0] = 8;
45   A[1] = 5;
46   A[2] = 7;
47   A[3] = 0;
48   A[4] = 2;
49   A[5] = 6;
50   A[6] = 4;
51   A[7] = 1;
52   A[8] = 3;
53   A[9] = 9;
54 }
55
```

RAM



© Prof. Vargas

```
56  void loop()   // Loop permanente
57  {
58    for (k = 0; k < 10; k++)
59    {
60      PORTB = A[k];    //Apresenta dados
61      Botao = digitalRead(Grava);  //Recolhe estado do botao
62      if (Botao == LOW)    //GRAVA foi apertado
63      {
64        digitalWrite(LED_Grava,HIGH); //acende o LED
65        dado = 0;
66        for (endereco = 0; endereco < 10; endereco++)
67        {
68          EEPROM.write(endereco, dado);
69          dado = !dado;
70        }
71        do
72        {
73          Botao = digitalRead(Grava);
74        }while(Botao == LOW); //espera tirar o dedo do botão
75
76        digitalWrite(LED_Grava,LOW); //apaga o LED
77      }
78      Botao = digitalRead(Le);  //Recolhe estado do botão
79      if (Botao == LOW)    //LE foi apertado
80      {
81        for (endereco = 0; endereco < 10; endereco++)
82        {
83          A[endereco] = EEPROM.read(endereco);
84        }
85        do
86        {
87          Botao = digitalRead(Le);
88        }while(Botao == LOW); //espera tirar o dedo do botão
89      }
90      delay(500);
91    }
92  }
93
```

© Prof. Vargas

© Prof. Vargas