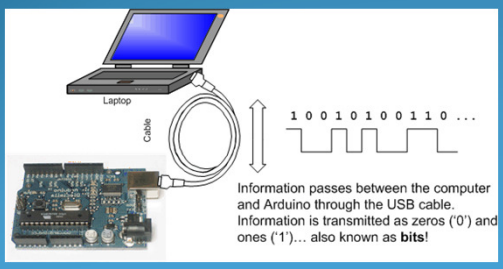
Comunicação Serial com Arduino

Prof. Cláudio A. Fleury

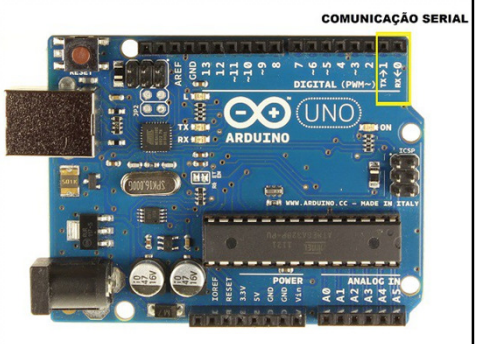
Nov-2011



55 Slides

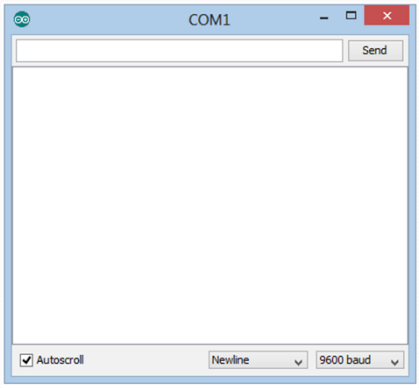
Pinos para Comunicação Serial

- O sinal de comunicação no Arduino UNO, é um sinal TTL de 5V.
- Para comunicação com um computador ou outro dispositivo que não tenha o mesmo nível de tensão é necessário um conversor de nível: TTL/ RS232, TTL/RS485, TTL/USB, ...



Terminal (monitor) Serial

- Além do recurso de *upload* através da comunicação serial, a IDE tem um **terminal serial** que auxilia na recepção e envio de dados para a placa sem a necessidade de se recorrer a uma ferramenta externa.



Comunicação Ponto a Ponto

Peer-to-Peer = P2P

- Comandos
  - Configuração da Porta Serial

```
// configura a porta serial do Arduino
Serial.begin(9600);
```

  - Qtde de bytes disponíveis para leitura na Porta Serial

```
// bytes disponíveis para leitura
n = Serial.available();
```

Comandos de Entrada de Dados

- Leitura de um byte na entrada da Porta Serial
  (remove o primeiro byte do buffer de entrada da porta serial)

```
// lê um byte da porta serial
carac = Serial.read();
// carac= -1 se não existir byte a ser lido
```

- Leitura de um byte na entrada da Porta Serial
  (não remove o byte do buffer de entrada da porta serial)

```
// retorna o primeiro byte do buffer serial
carac = Serial.peek();
// chamadas sucessivas a essa função retorna
// o mesmo byte, o primeiro do buffer
```


Comandos de Saída de Dados

- Gravação de um byte "binário" na saída da Porta Serial

```
// escrita de um byte binário na porta serial
Serial.write(byte);
int bytesEnv = Serial.write("Arduino");
```

- Gravação de um byte "ASCII"na saída da Porta Serial

```
// escrita de um byte ASCII na porta serial
Serial.print(carac);
// idem, porém com <CR/LF> ao final da escrita
Serial.println(string);

Serial.print(78)      // envia "78"
Serial.print(1.23456) // envia "1.23"
Serial.print('N')     // envia 'N'
Serial.print("Olá!")  // envia "Olá!"
```

Exemplo

```
// Timer Regressivo Fixo (5'15") com exibição na porta serial
// Prof. Cláudio – Out/2014
long tempo;
int  minu = 5, segu = 15;

void setup() {
  Serial.begin(9600);
  tempo = millis();
}

void loop() {
  if(millis()-tempo > 100) {
    tempo = millis();
    Serial.print("Timer: ");
    Serial.print(minu); Serial.print(":"); Serial.println(segu);
    atualizaTempo();
  }
}

void atualizaTempo(void) {
  if(--segu < 0) {
    segu = 59;
    if(--minu < 0){
      Serial.println("Tempo esgotado!");
      while(1);
    }
  }
}
```


Exercício 1

- Considerando o Timer Regressivo do exemplo anterior, o usuário deverá informar o intervalo de temporização desejado (minutos e segundos, dois bytes cada) no monitor serial do Arduino, antes de iniciar a contagem regressiva de tempo.

dica: use os comandos

`Serial.read()`

`Serial.available()`

```
// Timer Regressivo com exibição na porta serial
// Prof. Cláudio - Set/2016

// variáveis globais ----------------------------------------------------
long tempo;
int  minu = 1, segu = 5;

void setup() {
  Serial.begin(9600); tempo = millis();
  programaTempo();
}

void loop() {
  if(millis()-tempo > 100) {
    tempo = millis();
    Serial.print("Timer: "); mostra(minu); Serial.print(":");
    mostra(segu); Serial.println(""); atualizaTempo();
  }
}

// funções auxiliares ---------------------------------------------------
void atualizaTempo(void) {
  if(--segu < 0) {
    segu = 59;
    if(--minu < 0) {
      Serial.println("Tempo esgotado!");
      while(1); } } }

void mostra(int x) {        // mostra valor 'x' com duas casas
  if (x < 10)
    Serial.print("0");
  Serial.print(x);
}

void programaTempo(void) {  // aguarda envio do tempo (mm,ss) pela Serial
  int dzn, und;
  Serial.print("Minutos (mm):"); while(Serial.available() < 2);
  dzn = Serial.read()-'0'; und = Serial.read()-'0'; minu = 10*dzn + und;
  Serial.print("Segundos (ss):"); while(Serial.available() < 2);
  dzn = Serial.read()-'0'; und = Serial.read()-'0'; segu = 10*dzn + und;
  Serial.print(minu); Serial.print(":"); Serial.println(segu);
}
```

Exercício 2

- Acrescentar ao Timer do exercício anterior as seguintes funções:
  - Pausa ('P') – interrompe a temporização momentaneamente, até que o comando 'C' seja recebido pela porta serial.
  - Continua ('C') – continua a temporização a partir do ponto em que havia sido interrompida pelo comando 'P'.
  - Reprogramação ('R') – reprograma o tempo, reiniciando a temporização.


```
// Timer Regressivo Fixo com exibição na porta serial
// Prof. Cláudio - Set/2016
long tempo;    int  minu, segu;

void setup() {
 Serial.begin(9600); programaTempo(); tempo = millis(); }

void loop() {
 if(millis()-tempo > 100) {
   tempo = millis();
   Serial.print("Timer: "); mostra2C(minu);
   Serial.print(":"); mostra2C(segu); Serial.println("");
   atualizaTempo();   }
 if(Serial.available() > 0)
    switch(Serial.read())  {
       case 'P':  while(Serial.read() != 'C');  // 'P' pausa o temporizador
                  break;                        // 'C' continua
       case 'R':  programaTempo();              // 'R' reprograma o temporizador
    }
}

void atualizaTempo(void) {
 if(--segu < 0) {
   segu = 59;
   if(--minu < 0) {
    Serial.println("Tempo esgotado!");
    while(1); } } }

void mostra2C(int x) {                          // mostra valor 'x' com duas casas
 if (x < 10) Serial.print("0");
 Serial.print(x);
}

int leiaNumSerial(int digitos) {
 int dzn, und;
 while(Serial.available() < 2);
 dzn = Serial.read() - '0';   und = Serial.read() - '0';
 return(10*dzn + und);
}

void programaTempo(void) {
 Serial.println(); Serial.println(">>> Temporizador <<<");
 Serial.print("Minutos  (99): "); minu = leiaNumSerial(2); Serial.println(minu);
 Serial.print("Segundos (99): "); segu = leiaNumSerial(2); Serial.println(segu);
}
```

Prática 0

Depuração de *Sketchs*

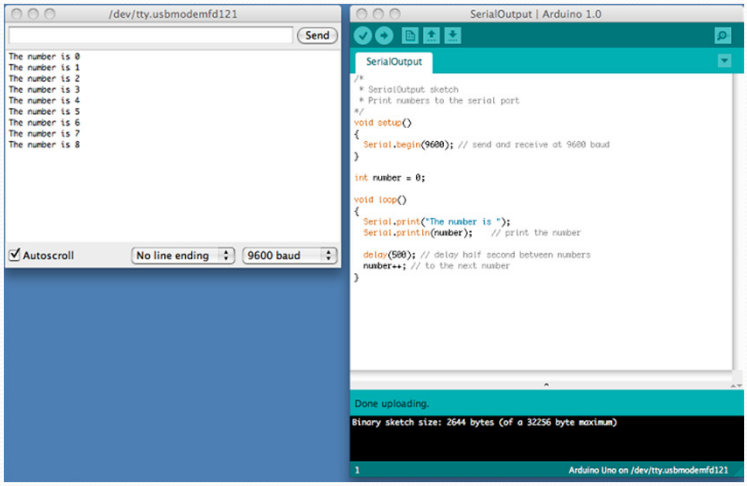
Envio de Informações do Arduino para o PC

Prática 0

- Mostra mensagens e/ou valores de variáveis no Monitor Serial do IDE do Arduino

```
/*
 *  Saída Serial: envia valores numéricos p/ porta serial
 */

void setup() {
  Serial.begin(9600);       // envia/recebe a 9600 baud
}

int num = 0;

void loop() {
  Serial.print("Número: ");
  Serial.println(num);      // mostra o número no monitor
  delay(500);               // atraso meio seg. entre números
  num++;                    // próximo número
}
```


Prática 0



Experimente outro programa de acesso a porta serial como o COOLTERM.

Prática 1

Uso da função *SerialEvent*()

Prática 1

- *SerialEvent*() é chamada depois de um *loop*() se existir algum byte disponível no *buffer* de entrada da Porta Serial
- Quando novos dados seriais chegam, o *sketch* adiciona-os em uma String
- Quando um caracter ASCII '*linhanova*' (LF – linefeed – '\n') é recebido, o laço envia o conteúdo da string para a saída da porta serial, e limpa a variável para iniciar um novo ciclo
- Um bom teste desse sketch seria usá-lo com um receptor GPS que envia sentenças NMEA 0183

Prática 1

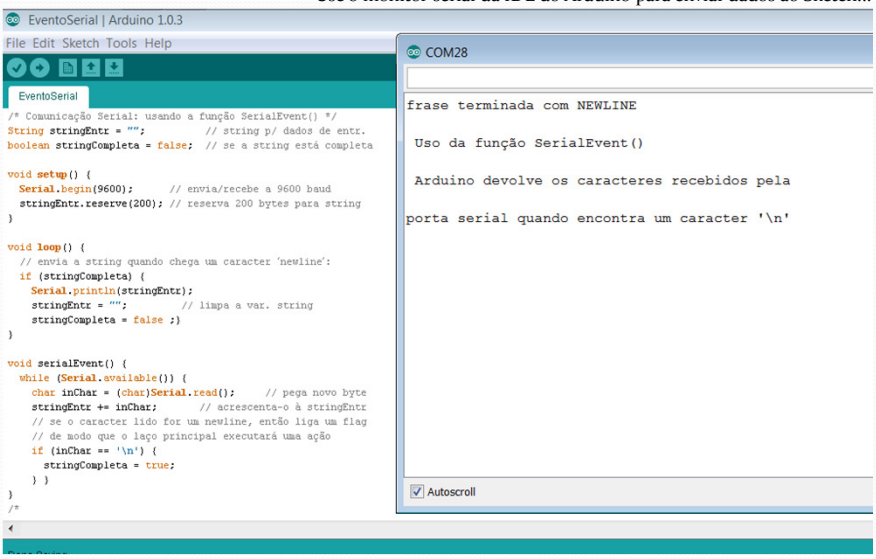
```
/*
 *  Comunicação Serial: usando a função SerialEvent()
 */
String stringEntr = "";           // string p/ dados de entr.
boolean stringCompleta = false;  // se a string está completa

void setup() {
  Serial.begin(9600);       // envia/recebe a 9600 baud
  stringEntr.reserve(200); // reserva 200 bytes para string
}

void loop() {
  // envia a string quando chega um caracter 'newline':
  if (stringCompleta) {
    Serial.println(stringEntr);
    stringEntr = "";        // limpa a var. string
    stringCompleta = false;
  }
}
```


Prática 1

```
/*
  SerialEvent ocorre sempre que chegam novos dados entrada
  serial (pino RECEPTOR). Esta rotina é executada após cada
  loop() → o uso de delay() no loop() pode atrasar a resposta.
  Vários bytes de dados podem estar disponíveis.
*/

void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();     // pega novo byte
    stringEntr += inChar;       // acrescenta-o à stringEntr
    // se o caracter lido for um newline, então liga um flag
    // de modo que o laço principal executará uma ação
    if (inChar == '\n')
      stringCompleta = true;
  }
}
```

Prática 1

Use o monitor serial da IDE do Arduino para enviar dados ao Sketch...

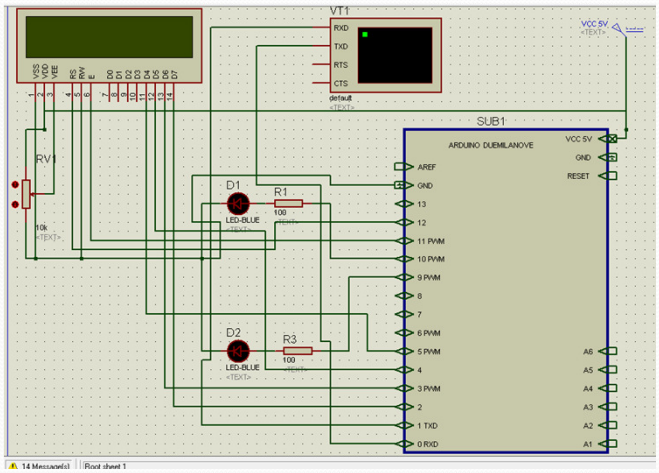


Prática 2

Handshaking – Chamada e Resposta Serial

Prática 2

- Comandar remotamente (à distância, pelo PC) o brilho de um LED (valor de 0 a 255) ligado numa saída analógica do Arduino



Prática 2

- Comandar remotamente o acendimento proporcional de um LED ligado a uma porta de saída analógica

```
#include <LiquidCrystal.h>
// inicia LCD com os números dos pinos da interface
LiquidCrystal lcd(12,11,5,4,3,2);  // RS, E, D4, D5, D6, D7
const int pinoLed = 9;        // o pino no qual o LED está ligado
int pos = 0;                  // posição de armazenamento do caracter rxdo

void setup() {
  Serial.begin(9600);         // inicia a comunicação serial
  pinMode(pinoLed, OUTPUT);   // inicia o ledPin como saída
  pinMode(10, OUTPUT);
  lcd.begin(16, 2);           // número de linhas e colunas do LCD: 16 x 2
  lcd.println("Aguardando CMD: "); lcd.print("999<ENTER>");
  Serial.println("Aguardando Comando Remoto (999<ENTER>): ");
}

int decodifica(char *s) {     // int decodifica(char s[])
  int soma, i;
  for(soma=i=0; (s[i]!=0) && (i<pos); i++)
    soma = soma*10 + (s[i]-48);
  return soma;
}
```

Prática 2

- Comandar remotamente o acendimento proporcional de um LED ligado a uma porta de saída analógica (continuação)

```
void loop() {
  char j, n, carac, seq[20];
  int brilho;

  n = Serial.available();           // qtde de dados enviados pelo remoto
  for(j=0; j<n; j++) {
    carac = Serial.read();
    if(carac == '.' || pos > 2)     // lê bytes até encontrar CR (13 = 0x0D)
      break;
    seq[pos++] = carac;
  }
  if((carac == '.') || (pos > 2)) {
    seq[pos] = '\0';
    Serial.print(" - "); Serial.println(seq);
    brilho = decodifica(seq);
    constrain(brilho,0,255);
    pos = 0;
    lcd.setCursor(0, 1); lcd.print("               ");
    lcd.setCursor(0, 1); lcd.print(brilho);
    analogWrite(pinoLed, (byte)brilho); // ajusta o brilho do LED }
    while(Serial.available())           // algo para ser lido?
      char dados = Serial.read();       // se sim, leia-o
}
```

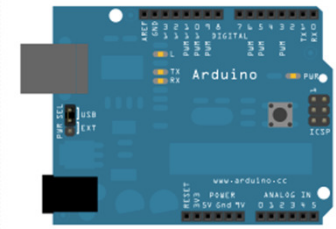

Prática 3

Jogo de Adivinhação

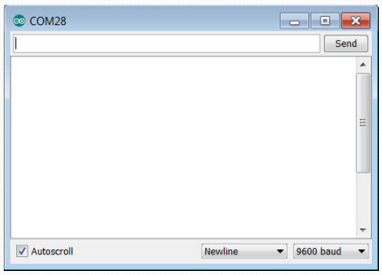
Prática 3

- Voce escolhe um número entre 0 e 100 e o Arduino advinhará...

ARDUINO



MONITOR SERIAL



Prática 3

- Voce escolhe um número entre 0 e 100 e o Arduino advinhará...

```
/* Adivinha o número de 0 a 100 escolhido pelo usuário...  */

boolean respondido = true;
int palpite, intervalo = 100;

void setup() {
  Serial.begin(9600);      // envia/recebe a 9600 baud
  Serial.println("Jogo de Adivinhacao de numero (0 a 100):\n");
  Serial.println("Responda as tentativas com '+' ou '-' se num. escolhido");
  Serial.println("por vc for > ou < que o indicado por mim (Arduino),");
  Serial.println("ou '.' (digito zero) se eu tiver adivinhado!");
  Serial.println();
  palpite = intervalo/2;
  intervalo /= 2;
}

void loop() {
  if(respondido) {
    Serial.print("ARDUINO: O numero eh ");
    Serial.print(palpite);
    Serial.print("? [+ . -]\n");
    respondido = false; }
}
```

Prática 3

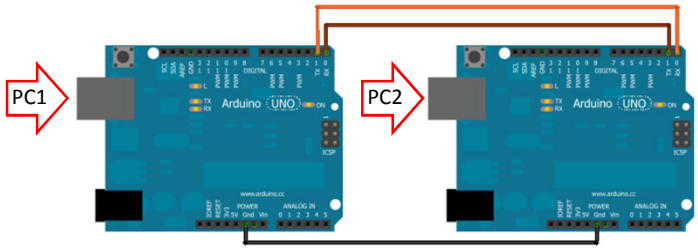
- Comandar remotamente o brilho de um LED... (continuação)

```
/* Adivinha número de 0 a 100 escolhido pelo usuário...  */

void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();     // pega resposta
    if (inChar == '+') {
      intervalo = max(1,intervalo/2);
      palpite += intervalo;
      respondido = true; }
    else if (inChar == '-') {
      intervalo = max(1,intervalo/2);;
      palpite -= intervalo;
      respondido = true; }
    else if (inChar == '.') {
      Serial.print("\nARDUINO: Este eh o numero: ");
      Serial.println(palpite);
      Serial.println("ARDUINO: Vamos jogar denovo? (RESET-me)"); }
    else {
      Serial.print(inChar,HEX); Serial.println("???"); }
    delay(50);
    while (Serial.available()) {     // descarta bytes adicionais no buffer
      Serial.read(); } }
}
```


Arduino–Arduino usando Biblioteca

EasyTransfer

- Transferência serial de dados entre dois Arduinos
  - Conecte TX em RX, RX em TX, GND em GND
  - Alimente cada Arduino pela USB com respectivo PC
  - Acompanhe o desenvolvimento da comunicação
    - Pelo **Monitor Serial** da IDE de cada ARDUINO, e/ou
    - Pelos LEDs na saída digital 13 (TX pisca na proporção 1:2 em relação ao RX)



Arduino - Arduino

```
#include <EasyTransfer.h>     // Arduino TRANSMISSOR

EasyTransfer ET;              // objeto de transferência fácil (biblioteca)
// Declara a estrutura de dados para usar a biblioteca EasyTransfer
// Obs.: a estrutura de dados deve ser a mesma no TX e no RX...
struct DATA_STRUCTURE {
 int  int_data;
 char char_data;
};

DATA_STRUCTURE meusDados;
int contador = 0, pacote = 0;
byte estadoLED = LOW;

void setup() {
 Serial.begin(9600);
 pinMode(13,OUTPUT);
 ET.begin(details(meusDados), &Serial); //Iniciando porta Serial p/ EasyTransfer
}

void loop() {
 meusDados.char_data = (char) contador;
 meusDados.int_data = contador++;

 ET.sendData();
 delay(1000);
 if(++pacote >= 5) {
   estadoLED = !estadoLED;
   digitalWrite(13,estadoLED);
   pacote = 0;
 }
}
```

Prof. Cláudio A. Fleury - Nov-2011

28

Arduino - Arduino

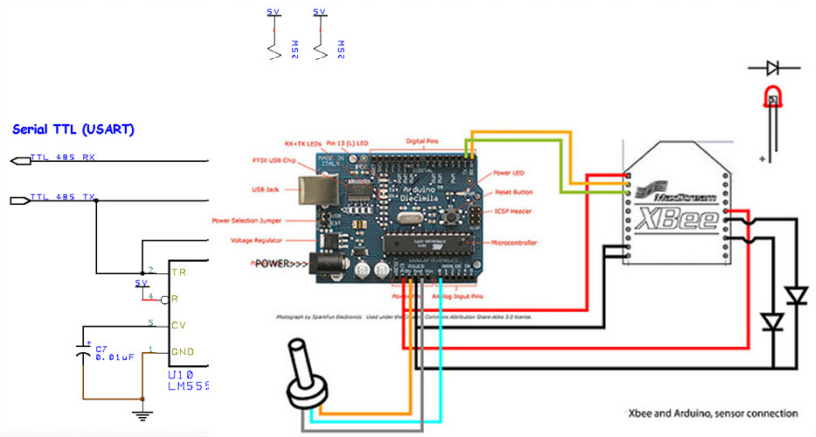
```
#include <EasyTransfer.h>     // Arduino RECEPTOR

EasyTransfer ET;              // objeto de transferência fácil (biblioteca)
// Declara a estrutura de dados para usar a biblioteca EasyTransfer
// Obs.: a estrutura de dados deve ser a mesma no TX e no RX...
struct REGISTRO {
 int  int_data;
 char char_data;
};

REGISTRO meusDados;
int contador = 0, pacote = 0;
byte estadoLED = LOW;

void setup() {
 Serial.begin(9600);
 pinMode(13,OUTPUT);
 ET.begin(details(meusDados), &Serial); //Iniciando porta Serial p/ EasyTransfer
}

void loop() {

  if(ET.receiveData()) {
    Serial.println(meusDados.char_data);
    Serial.println(meusDados.int_data);
    delay(250);
    pacote++;
  }
  if(pacote >= 9) {
    estadoLED = !estadoLED;
    digitalWrite(13,estadoLED);
    pacote = 0;
  }
}
```

Prof. Cláudio A. Fleury - Nov-2011

29

Comunicação Serial em Rede

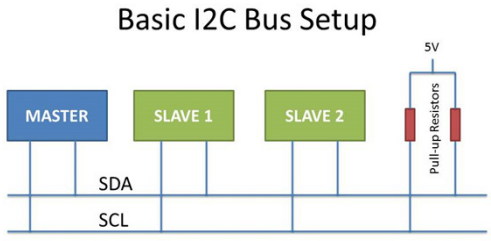
- **Barramentos** - interligação de vários nós de comunicação
  - **I²C** - *Inter-Integrated Circuit Bus*
    - Curto alcance, conexão de componentes numa mesma placa de circuito impresso (PCB)

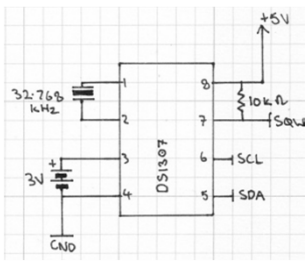


RTC – *Real Time Clock*

Comunicação Serial em Rede

- *Inter-Integrated Circuit Bus* (I²C)
  - Exemplo
    - Relógio de Tempo Real: **DS1307**





Comunicação Serial em Rede

- *Inter-Integrated Circuit Bus* (I^2C)
  - Exemplo
    - Relógio de Tempo Real (RTC – *Real Time Clock*) **DS1307**: tem 8 registradores para armazenar horário e data correntes

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/AM							
03H	0	0	0	0	0	DAY			Day	01–07
04H	0	0	10 Date		Date				Date	01–31
05H	0	0	0	10 Month	Month				Month	01–12
06H	10 Year				Year				Year	00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H–FFH

Para alterar um único registrador todos os 8 registradores precisam ser reescritos.

Comunicação Serial em Rede

- *Inter-Integrated Circuit Bus* (I^2C)
  - Exemplo DS1307

Lendo dados em um DS1307:
1. Reset o registrador para a primeira posição,
2. Requisite sete bytes de dados,
3. Receba-os em sete variáveis.

O endereço do dispositivo DS1307 é 0×68.

Exemplo de código C:

```
// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val) {
  return ( (val/10*16) + (val%10) );
}

// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val) {
  return ( (val/16*10) + (val%16) );
}
```

```
#define DS1307_I2C_ADDRESS 0x68 // each I2C object has a unique bus address
                                // the DS1307 address is 0x68
Wire.beginTransmission(0x68);
Wire.send(0);                   // nova versão: Wire.write(0);
Wire.endTransmission();
Wire.requestFrom(DS1307_I2C_ADDRESS, 7);
*second     = bcdToDec(Wire.receive());
*minute     = bcdToDec(Wire.receive());
*hour       = bcdToDec(Wire.receive());
*dayOfWeek  = bcdToDec(Wire.receive());
*dayOfMonth = bcdToDec(Wire.receive());
*month      = bcdToDec(Wire.receive());
*year       = bcdToDec(Wire.receive());
```

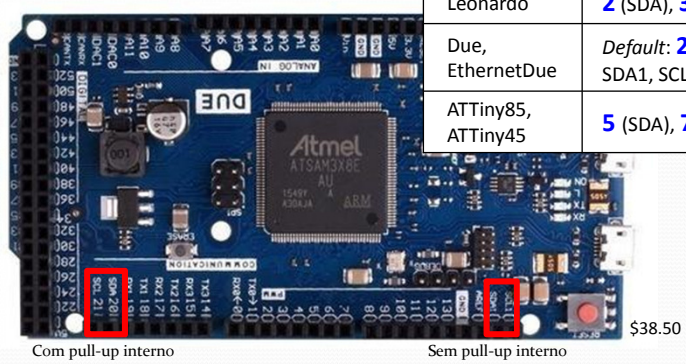
Comunicação Serial em Rede

- I²C para Arduino
  - Biblioteca ***Wire***
    - Usa endereços de **7** bits: **0** a **127** (**0** a **7** são reservados)
    - Biblioteca ***Wire*** herda características (deriva) da classe de fluxos de bits (***Stream***), compatível com as funções de leitura e escrita: *read*() e *write*()
    - Métodos
      - *begin*(), *begin(endereço)*, *requestFrom(endereço*, cont), *beginTransmission(endereço)*, *endTransmission*(), *write*(), *available*(), *byteread*(), *onReceive(alça)*, *onRequest(alça)*
    - Versões de endereços I²C: com **7** e **8** bits
      (oitavo bit na versão de **7** bits determina a operação: RD ou WR)

Comunicação Serial em Rede

- I²C para Arduino
  - Pinagem

Placa	Pinos I2C / TWI
Uno, Nano, Pro Mini	**A4** (SDA), **A5** (SCL)
Mega2560	**20** (SDA), **21** (SCL)
Leonardo	**2** (SDA), **3** (SCL)
Due, EthernetDue	*Default*: **20** (SDA0), **21** (SCL0) SDA1, SCL1 (perto do AREF)
ATTiny85, ATTiny45	**5** (SDA), **7**(SCL)



$38.50

Com pull-up interno

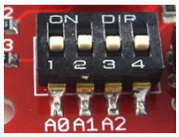
Sem pull-up interno

Prática 4

Memória Serial – I²C

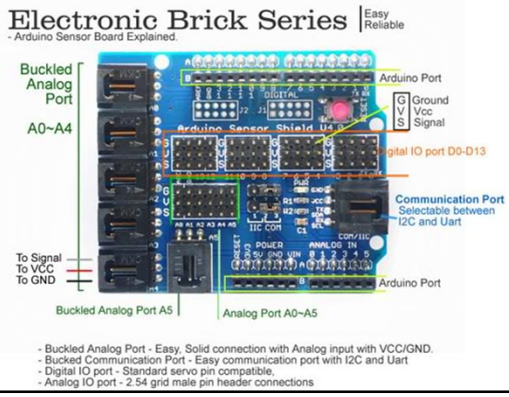
Prática 4

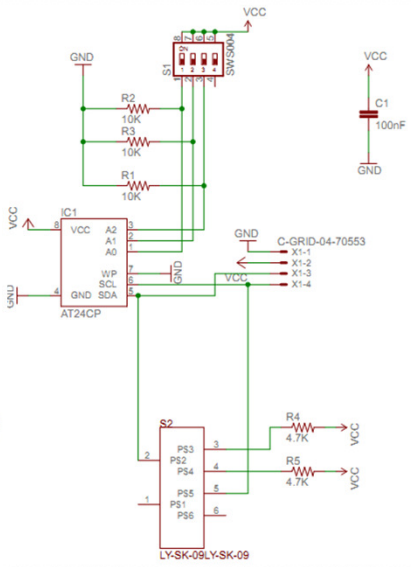
- Módulo EEPROM Serial com Arduino
  - Chips **AT24Cxx** tem endereço base I²C **0x50**
    - 24C01: 1 kbit = 128x8 bit;
    - 24C02: 2 kbit = 256 x 8 bit
    - 24C04: 4 kbit = 512 x 8 bit;
    - 24C08: 8 kbit = 1024 x 8 bit
    - 24C16: 16 kbit = 2048 x 8 bit
  - Os últimos três bits de endereço podem ser definidos pelas chaves DIP Switches de acordo com a quantidade de módulos EEPROM (*bricks*) usados
  - Ajuste do sinal RS do *brick*
    - Define como será conectado os sinais SDA e SCL na transmissão I²C, usando resistores pull-up ou não (somente o *brick* mais próximo ao Arduino deve ter RS ligado)



Prática 4

- Conecte o *brick* ao **Arduino Sensor Shield**
  - Coloque o strap I²C/COM na posição I²C





Código Prática 4

```
#include <Wire.h>
#define EEPROM_ADDR 0x50      // endereço do barramento I2C da EEPROM 24LC256 256K
void setup() {
 Wire.begin();                // adere ao barr. I2C (ender. opcional para MASTER)
 Serial.begin(9600); Serial.println("Teste de Gravacao:");
 for (int i=0; i<20; i++) {  // laço para os primeiros 20 slots
   i2c_eeprom_write_byte(EEPROM_ADDR,i,i+65); // ender. grav. + 65 'A' ou 97 'a'
   Serial.print(". ");
   delay(10); }               // AGUARDA TERMINO DA GRAVACAO
 delay(500); Serial.println("\nReading Test:");
 for (int i=0; i<20; i++) {  // laço para os primeiros 20 slots
   Serial.print(i2c_eeprom_read_byte(EEPROM_ADDR, i),HEX); Serial.print(" "); }
 // ajustes para o teste de páginas...
 byte PageData[30];           // vetor dos dados a serem gravados na página
 byte PageRead[30];           // vetor dos lados lidos de uma página
 for (int i=0; i<30; i++)     // zera ambos vetores
   PageData[i] = PageRead[i] = 0;
 for (int i=0; i<30; i++)
   PageData[i] = i+33;        // preenche o vetor com caracter de teste: 33 '!'
 Serial.println("\nTeste de Gravacao em Pagina:");
 i2c_eeprom_write_page(EEPROM_ADDR, 100, PageData, 28 ); // máx. 28 bytes/pag
 Serial.println("Teste de Leitura em Pagina:");
 i2c_eeprom_read_buf(EEPROM_ADDR, 100, PageRead, 28);
 for (int i=0; i<28; i++) {
   Serial.print(PageRead[i],HEX); // mostra dados lidos da Página
   Serial.print(" "); }
}
```

Prática 4

Código (cont.1)

```
void loop() { }

void i2c_eeprom_write_byte(int deviceaddress,unsigned int eeaddress,byte data ) {
 int rdata = data;
 Wire.beginTransmission(deviceaddress);
 Wire.write((int)(eeaddress >> 8));     // byte da parte alta do Endereço (MSB)
 Wire.write((int)(eeaddress & 0xFF));  // byte da parte baixa do Endereço (LSB)
 Wire.write(rdata);
 Wire.endTransmission();
}

// Address é ender. de página, 6 bits (63). Mais que isso e volta-se ao inicio,
// até 28 bytes de dados no máximo, por causa da bib. Wire (buffer de 32 bytes)
void i2c_eeprom_write_page(int devaddr,unsigned eeaddrpg,byte* data,byte length){
 Wire.beginTransmission(devaddr);
 Wire.write((int)(eeaddrpg >> 8));      // byte da parte alta do Endereço
 Wire.write((int)(eeaddrpg & 0xFF));    // byte da parte baixa do Endereço
 for (byte c = 0; c < length; c++)
   Wire.write(data[c]);
 Wire.endTransmission();
 delay(10);                             // precisa algum atraso
}
```

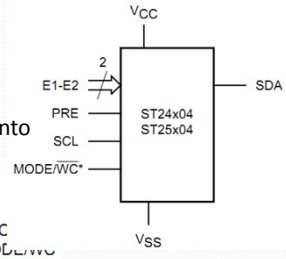

Prática 4

Código (cont.2)

```
byte i2c_eeprom_read_byte(int deviceaddress, unsigned int eeaddress) {
 byte rdata = 0xFF;
 Wire.beginTransmission(deviceaddress);
 Wire.send((int)(eeaddress >> 8));     // Address High Byte
 Wire.send((int)(eeaddress & 0xFF));  // Address Low Byte
 Wire.endTransmission();
 Wire.requestFrom(deviceaddress,1);
 if (Wire.available())
   rdata = Wire.read();
 return rdata;
}

// should not read more than 28 bytes at a time!
void i2c_eeprom_read_buf(int devadd, unsigned int eeadd, byte *buffer, int len) {
 Wire.beginTransmission(devadd);
 Wire.send((int)(eeadd >> 8)); // Address High Byte
 Wire.send((int)(eeadd & 0xFF)); // Address Low Byte
 Wire.endTransmission();
 Wire.requestFrom(devadd,len);
  for ( int c = 0; c < len; c++ )
   if (Wire.available())
     buffer[c] = Wire.read();
}
```

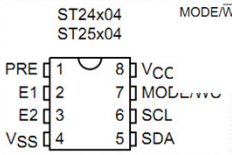
EEPROM Serial 24C04

- Características
  - **4** kbits organizados em **2** blocos de **256x8** bits
  - Um milhão de ciclos de RD/WR
  - Até 4 unidades/slaves por bus I²C
  - Oper.s RD/WR são iniciadas por condição START gerada pelo Master, seguida de 7 bits (cód. Identif. 1010) mais um bit RD/WR e terminadas por um bit de reconhecimento

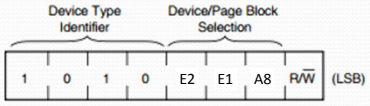


PRE	Write Protect Enable
E1-E2	Chip Enable Inputs
SDA	Serial Data Address Input/Output
SCL	Serial Clock
MODE	Multibyte/Page Write Mode (C version)
$\overline{WC}$	Write Control (W version)
V_{CC}	Supply Voltage
V_{SS}	Ground

* $\overline{WC}$ signal is only available for ST24/25W04 products.



Slave Address Format



Prof. Cláudio A. Fleury - Nov-2011

42

EEPROM Serial 24C04

- Seleção (endereçamento)

definiido pelo padrão I2C: disp. de memória

0 p/ gravação
1 p/ leitura

	Device Code				Chip Enable		Block Select	$\overline{RW}$
Bit	b7	b6	b5	b4	b3	b2	b1	b0
Device Select	1	0	1	0	E2	E1	A8	$\overline{RW}$

Note: The MSB b7 is sent first.

definem a memória no bus (1 de 4)

definem o bloco a usar (0 ou 1)

- Modos de Operação

	Mode	$\overline{RW}$ bit	MODE	Bytes	Initial Sequence
Leitura	Current Address Read	'1'	X	1	START, Device Select, $\overline{RW}$ = '1'
	Random Address Read	'0'	X	1	START, Device Select, $\overline{RW}$ = '0', Address,
		'1'			reSTART, Device Select, $\overline{RW}$ = '1'
	Sequential Read	'1'	X	1 to 512	Similar to Current or Random Mode
Gravação	Byte Write	'0'	X	1	START, Device Select, $\overline{RW}$ = '0'
	Multibyte Write (2)	'0'	V_{IH}	4	START, Device Select, $\overline{RW}$ = '0'
	Page Write	'0'	V_{IL}	8	START, Device Select, $\overline{RW}$ = '0'

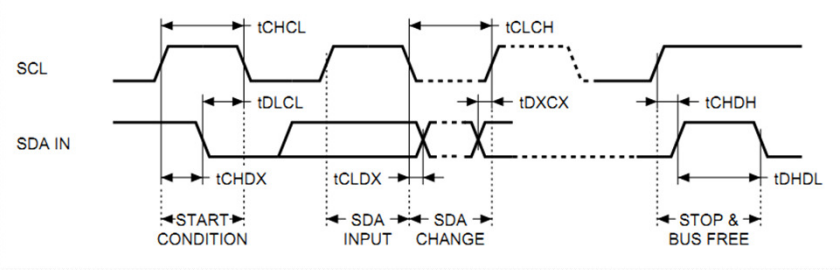
Notes: 1. X = V_{IH} or V_{IL}
2. Multibyte Write not available in ST24/25W04 versions.

Prof. Cláudio A. Fleury - Nov-2011

43

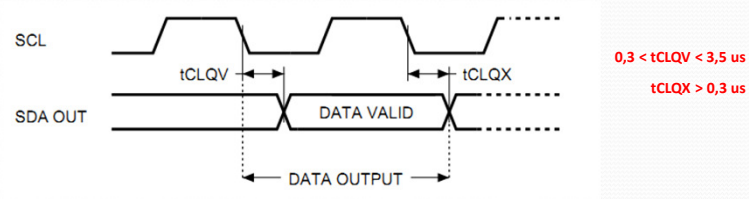
EEPROM Serial 24C04

- Formas de Onda
  - Leitura: S → M



EEPROM Serial 24C04

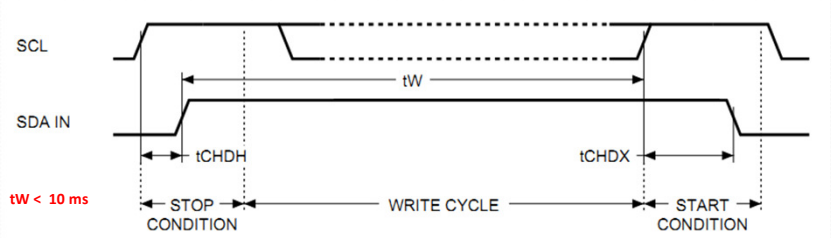
- Formas de Onda



0,3 < tCLQV < 3,5 us

tCLQX > 0,3 us

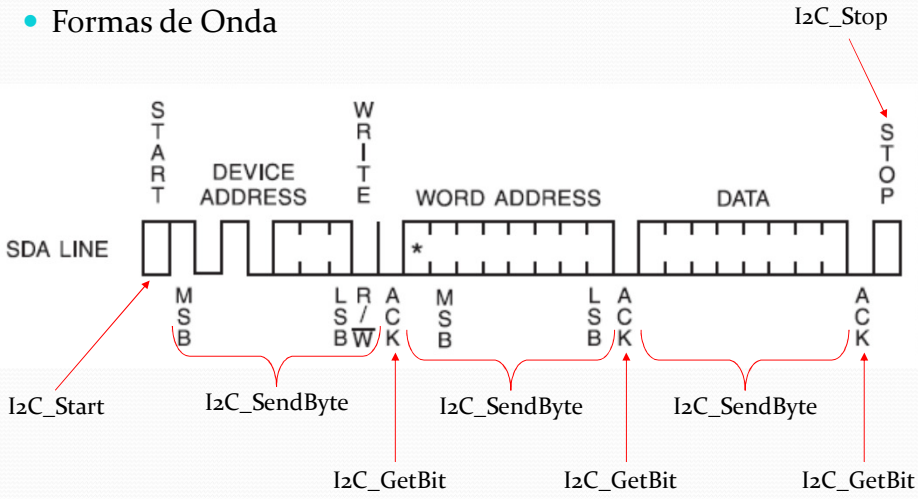
  - Gravação: M → S



tW < 10 ms

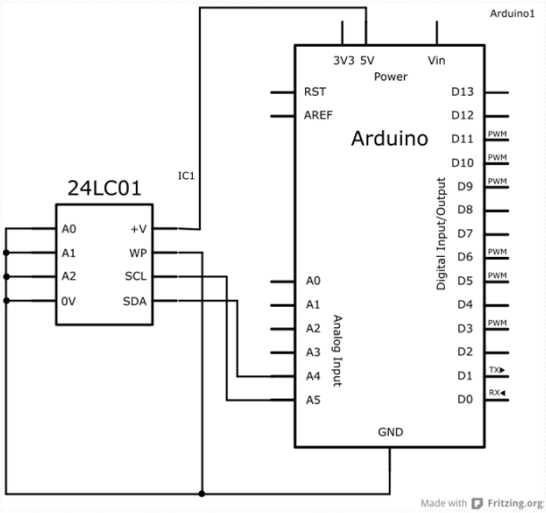
EEPROM Serial 24C04

- Formas de Onda



Exercício

- Realizar os mesmos testes usando a EEPROM Serial 24C01 ou 24C04



Exercício

- Realizar os mesmos testes usando a EEPROM Serial 24C04

```
//  Uso do barramento I2C com EEPROMs Seriais (24C01, 20C02, 24C04, 24C08, 24C16)
//  Para uma única memória no barramento, conecte assim:
//     EEPROM 8(Vcc) ao +5V;          EEPROM 5(SDA) ao Analog In 4
//     EEPROM 6(SCL) ao Analog In 5; EEPROM 4(GND),7(WP),1(A0),2(A1),3(A2) ao GND

#include <Wire.h>
const byte DEVADDR = 0x51;

void setup() {
  byte msg1[] = "Mensagem1.";   // dados para gravar
  byte msg2[] = "Qualquer coisa";
  byte msg3[] = "Fim do programa!";
  byte msgf[16] = { 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
                    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff };
  Wire.begin();
  Serial.begin(9600);
  eeprom_write_page(DEVADDR, 0x000, msgf, 16); // apaga páginas que serão gravadas
  eeprom_write_page(DEVADDR, 0x010, msgf, 16);
  eeprom_write_page(DEVADDR, 0x020, msgf, 16);
  eeprom_write_page(DEVADDR, 0x100, msgf, 16);
  eeprom_write_page(DEVADDR, 0x1f0, msgf, 16);
  Serial.println("Depois de limpar as páginas em 0x000, 0x100 e 0x1f0:");
  eeprom_dump(DEVADDR, 0, 512);
  eeprom_write_page(DEVADDR, 0x000, msg1, sizeof(msg1)); // grava algo na EEPROM
  eeprom_write_page(DEVADDR, 0x100, msg2, sizeof(msg2));
  eeprom_write_page(DEVADDR, 0x1f0, msg3, 16);
  Serial.println("Dados gravados na Memória.");
}
```


Exercício

- Realizar os mesmos testes usando a EEPROM Serial 24C04

```
void loop()
{
  Serial.println("eeprom_read_byte, iniciando em 0");
  for (int i = 0; i < 16; i++) {
    byte b = eeprom_read_byte(DEVADDR, i);
    Serial.print(b, HEX); Serial.print(' '); }
  Serial.println();
  Serial.println("eeprom_read_buffer, iniciando em 0");

  byte buffer[16];
  eeprom_read_buffer(DEVADDR, 0, buffer, sizeof(buffer));
  for (int i = 0; i < sizeof(buffer); i++) {
    char outbuf[6];
    sprintf(outbuf, "%02X ",buffer[i]); Serial.print(outbuf); }
  Serial.println();

  for (int i = 0; i < sizeof(buffer); i++) {
    if (isprint(buffer[i]))
      Serial.print(buffer[i]);
    else
      Serial.print('.'); }
  Serial.println();

  Serial.println("eeprom_dump(DEVADDR, 0, 512)");
  eeprom_dump(DEVADDR, 0, 512); Serial.println();
  delay(20000);
}
```

Exercício

- Realizar os mesmos testes usando a EEPROM Serial 24C04

```
void eeprom_write_byte(byte deviceaddress, int eeaddress, byte data) {
  // tres lsb's do byte deviceaddress são os bits 8-10 do eeaddress
  byte devaddr = deviceaddress | ((eeaddress >> 8) & 0x07);
  byte addr    = eeaddress;
  Wire.beginTransmission(devaddr);
  Wire.write(int(addr));
  Wire.write(int(data));
  Wire.endTransmission();
  delay(10);
}

// Endereço inicial das pág.s (16 bytes): 0x000, 0x010, 0x020, ...
// Em dispositivos "page write" o último byte deve estar na mesma pág. do primeiro byte
// Nenhuma verificação é feita nesta função!
void eeprom_write_page(byte deviceaddress, unsigned eeaddr, const byte *data, byte length) {
  // tres lsb's do byte deviceaddress são os bits 8-10 do eeaddress
  byte devaddr = deviceaddress | ((eeaddr >> 8) & 0x07);
  byte addr    = eeaddr;
  Wire.beginTransmission(devaddr);
  Wire.write(int(addr));
  for (int i = 0; i < length; i++)
    Wire.write(data[i]);
  Wire.endTransmission();
  delay(10);
}
```


Exercício

- Realizar os mesmos testes usando a EEPROM Serial 24C04

```
// TODO: mudar o tipo de dado para 'int' e retornar -1 se não puder ler.
int eeprom_read_byte(byte deviceaddress, unsigned eeaddr) {
  byte rdata = -1;
  // tres lsb's do byte deviceaddress são os bits 8-10 do eeaddress
  byte devaddr = deviceaddress | ((eeaddr >> 8) & 0x07);
  byte addr    = eeaddr;
  Wire.beginTransmission(devaddr);
  Wire.write(int(addr));
  Wire.endTransmission();
  Wire.requestFrom(int(devaddr), 1);
  if (Wire.available())
    rdata = Wire.read();
  return rdata;
}
// Retorna bytes lidos da memória
// Devido ao tamanho do buffer na bibl. Wire, não leia mais que 30 bytes por vez!
int eeprom_read_buffer(byte deviceaddr, unsigned eeaddr, byte * buffer, byte length) {
  // tres lsb's do byte deviceaddress são os bits 8-10 do eeaddress
  byte devaddr = deviceaddr | ((eeaddr >> 8) & 0x07);
  byte addr    = eeaddr;
  Wire.beginTransmission(devaddr);
  Wire.write(int(addr));
  Wire.endTransmission();
  Wire.requestFrom(devaddr, length);
  int i;
  for (i = 0; i < length && Wire.available(); i++)
    buffer[i] = Wire.read();
  return i;
}
```

Exercício

- Realizar os mesmos testes usando a EEPROM Serial 24C04

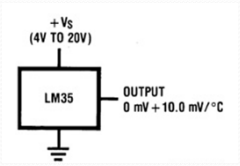
```
// A apresentação é como em "hexdump -C", sempre mostra 16 bytes
void eeprom_dump(byte devaddr, unsigned addr, unsigned length) {
  // começa com o início de uma página de 16 bits que contem o primeiro byte a ser mostrado
  unsigned startaddr = addr & (~0x0f);
  // 'stopaddr' é o endereço da próxima página depois do último byte
  unsigned stopaddr  = (addr + length + 0x0f) & (~0x0f);

  for (unsigned i = startaddr; i < stopaddr; i += 16) {
    byte buffer[16]; // guarda uma página da EEPROM
    char outbuf[6];  // local para três dígitos hexadecimal e ': \0'
    sprintf(outbuf, "%03x: ", i);
    Serial.print(outbuf);
    eeprom_read_buffer(devaddr, i, buffer, 16);
    for (int j = 0; j < 16; j++) {
      if (j == 8)
        Serial.print(" ");
      sprintf(outbuf, "%02x ", buffer[j]);
      Serial.print(outbuf);
    }
    Serial.print(" |");
    for (int j = 0; j < 16; j++)
      if (isprint(buffer[j]))
        Serial.print((char)buffer[j]);
      else
        Serial.print('.');
    Serial.println("|");
  }
}
```

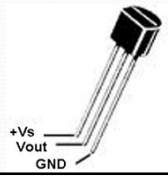

Desafio

Registrador de Temperaturas

Desafio

- Usando um RTC DS1307, um sensor de temperatura LM35, e uma memória EEPROM Serial 24C04 (512B), faça um registrador mensal de temperaturas ambiente (-50 a 127^{0}C)
  - Registre a máxima e a mínima temperatura do dia, com respectivos horários
  - Registre o valor médio da temperatura a cada duas horas
  - Precisão mínima das medidas: 1^{o}C
  - Use um display LCD 16x2 e teclas para exibir valores medidos
  - Use porta serial para realizar a descarga dos dados medidos para o PC





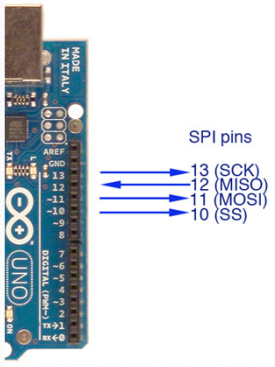
Prof. Cláudio A. Fleury - Nov-2011

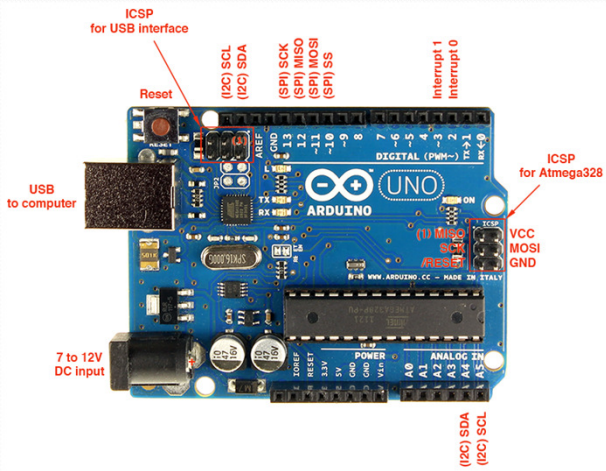
54

Pinos SPI no Arduino

E/S digitais

ICSP





Prof. Cláudio A. Fleury - Nov-2011

55

Prática 5

I²C com dois Arduinos

Prática 5

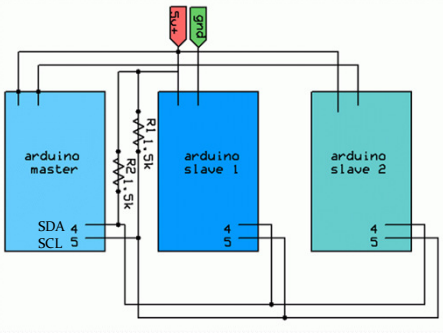
- Comunicação entre dois Arduinos usando I²C

Material:

2x Arduino, UNO
2x Resistor, 1k5 Ω
2x
1x Sensor de Temperatura, TMP36

Procedimento:

Arduino-1 conectado ao Sensor de Temperatura, TMP36, envia ao Arduino-2 a temperatura.
Arduino-2 exibe temperatura no LCD.



Prática 6

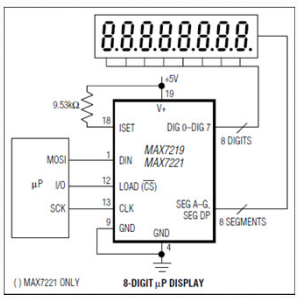
Ethernet

Prática 6

- Comunicando com o *shield* **Ethernet W5100**
  - Conexão do Arduino a um roteador Ethernet para envio/recepção de dados via Internet
  - Você poderá ler dados de fora de sua rede e enviar dados para a Internet e torná-los visíveis através de um navegador WEB

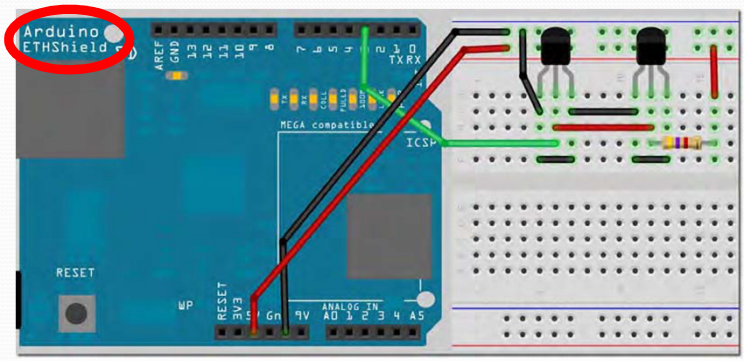
Material:

1x Arduino Ethernet Shield w5100
2x Sensores de temperatura DS18B20
2x resistores de 4k7 Ω



Prática 6

Conexões



Prática 6

Código

```
// Projeto baseado no exemplo Arduino Webserver de David A. Mellis e Tom Igoe
#include <SPI.h>
#include <Ethernet.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// Fio de dados é ligado ao pino 3 do Arduino
#define ONE_WIRE_BUS            3
#define TEMPERATURE_PRECISION   12
float tempC, tempF;

// Configura uma instância oneWire para comunic. com outros dispositivos OneWire
// (não apenas CI's de temperatura Maxim/Dallas)
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// vetores dos endereços dos dispositivos
DeviceAddress insideThermometer  = { 0x10,0x7A,0x3B,0xA9,0x01,0x08,0x00,0xBF };
DeviceAddress outsideThermometer = { 0x10,0xCD,0x39,0xA9,0x01,0x08,0x00,0xBE };
byte mac[] = { 0x48,0xC2,0xA1,0xF3,0x8D,0xB7 };
byte ip[]  = { 192,168,0,104 };

// Inicia o servidor na porta 80
Server server(80);
```

Prática 6

Código (cont.1)

```
void setup() {
  Ethernet.begin(mac, ip);                  // inicia ethernet
  server.begin();                           // inicia servidor
  sensors.begin();                          // inicia a biblioteca de sensores
  sensors.setResolution(insideThermometer, TEMPERATURE_PRECISION);
  sensors.setResolution(outsideThermometer, TEMPERATURE_PRECISION);
}

// função para pegar a temperatura de um dispositivo
void getTemperature(DeviceAddress deviceAddress) {
  tempC = sensors.getTempC(deviceAddress);
  tempF = DallasTemperature::toFahrenheit(tempC);
}

#define clp client.println

void loop() {
  sensors.requestTemperatures();
  Client client = server.available();     // aguarda a chegada de clientes
  if (client) {
    boolean BlankLine = true;             // requis. http final.c/ linha em branco
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
```

Prof. Cláudio A. Fleury - Nov-2011

62

Prática 6

Código (cont.2)

```
        if (c == '\n' && BlankLine) {       // fim da requisição HTTP
          getTemperature(insideThermometer);
          clp("HTTP/1.1 200 OK");           // reposta HTTP padrão
          clp("Content-Type: text/html\n");
          clp("<html><head><META HTTP EQUIV=""refresh""CONTENT=""5""">\n");
          clp("<title>Servidor Web Arduino</title></head>");
          clp("<body>\n"); clp("<h1>Servidor Web Arduino</h1>");
          clp("<h3>Temperatura Interna</h3>");
          clp("Temp C:"); clp(tempC); clp("<br/>");
          clp("Temp F:"); clp(tempF); clp("<br/>");
          getTemperature(outsideThermometer);
          clp("<h3>Temperatura Externa</h3>");
          clp("Temp C:"); clp(tempC); clp("<br/>");
          clp("Temp F:"); clp(tempF); clp("<br/>");
          break; }
        if (c == '\n') {                    // iniciando uma nova linha
          BlankLine = true; }
        else if (c != '\r') {               // linha atual tem um caracter
          BlankLine = false;}
      }
    }
    delay(10);                              // tempo p/ o browser receber dados
    client.stop();                          // fecha conexão
  }
}
```

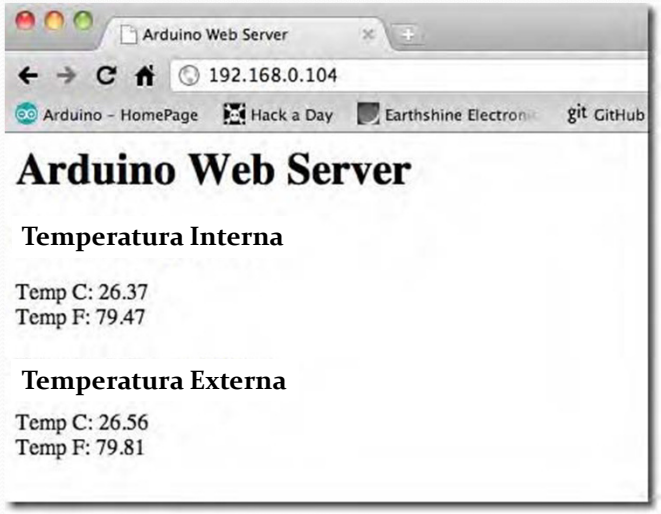
Prof. Cláudio A. Fleury - Nov-2011

63

Prática 6

Resultados

No seu navegador digite o endereço IP e a porta, por exemplo, **192.168.0.104:80**



CXO Serial com Modem's

- Comandos Hayes ou Comandos AT ("*attention*!")
  - Linguagem de configuração de modems
  - Desenvolvida para o **Hayes Smartmodem**, 300 bps, da extinta *Hayes Microcomputer Products*, fundada por Dennis Hayes em 1981
  - Comando Hayes padrão
    - AT + comando (palavra-chave) seguida ou não de parâmetros
    - Mais informações: en.wikipedia.org/wiki/Hayes_command_set
    - Exemplo:
      - `AT+CPIN=1234` `% informa código PIN`
      - `AT&V` `% mostra status do modem`
      - `ATI` `% Status (Fabricante, Modelo, Revisão, IMEI, possibilidades)`

[1] troque % pelo número do módulo.

Módulos p/ Bluetooth

Models	VDD	Size(mm)	Flash	Chip	BT Version
HM-01	3.3V	26.9*13*2.2	8M	BC417143	V2.1+EDR
HM-02	2.5-3.7V	26.9*13*2.2	6M	BC3/BC4	V2.1
HM-03	2.5-3.7V	27.4*12.5*4.3	6M	BC3/BC4	V2.1
HM-04	3.3V	Not for sale			
HM-05	2.5-3.7V	13.5*18.5*2.3	6M	BC3/BC4	V2.1
HM-06	2.5-3.7V	13.5*18.5*2.3	6M	BC3/BC4	V2.1
HM-07	2.5-3.7V	13.5*18.5*2.3	8M		V2.1+EDR
HM-08	3.3V	26.9*13*2.5	8M	Class 1	V2.1+EDR
HM-09	2.5-3.7V	26.9*13*2.2	8M		V2.1+EDR
HM-10	2-3.7V	26.9*13*2.2	256Kb	CC2540/1	V4.0 BLE
HM-11	2.5-3.7V	13.5*18.5*2.2	256Kb	CC2540/1	V4.0 BLE
HM-15	5V	65*32*16	256KB	CC2540	V4.0 BLE

[1] troque % pelo número do módulo.

Módulo HC-05 (Bluetooth)

- Comandos Hayes na configuração do módulo:
  - `AT+NAME=ARDUINO%`[1]
  - `AT+UART=57600,0,0`
- Arduino UNO
  - Somente uma UART: pinos 0 (RX) e 1 (TX) → comunicação com PC
  - Biblioteca **SoftwareSerial**
    - Simulação de UART em outros dois pinos de saída digital, para comunicação entre Arduino e módulo HC-05
    - Exemplo:

```
#include <SoftwareSerial.h>
SoftwareSerial SWSerial(10, 11);   // pino 10: RX;  pino 11: TX
```

- Diferença entre os módulos HC-05 e HC-06
  - HC-05 pode ser configurado nos modos Master/Slave/Loopback
  - HC-06 pode ser configurado apenas no modo Slave
    - Modo Master: conecta-se a outros dispositivos Bluetooth
    - Modo Slave: recebe conexões de outros dispositivos Bluetooth
    - Modo Loopback: recebe dados do Master e envia de volta esses mesmos dados, usado geralmente em testes

[1] troque % pelo número do módulo.

Comandos AT

Comando	Função	HC-05	HC-06
AT	Teste	✓	✓
AT+RESET	Reset	✓	
AT+VERSION	Mostra a versão de software	✓	✓
AT+ORGL	Restaura configurações padrão	✓	
AT+ADDR?	Mostra o endereço do módulo BT	✓	
AT+NAME	Mostra/altera o nome do módulo BT	✓	✓
AT+RNAME?	Mostra o nome do módulo BT remoto	✓	
AT+ROLE	Seleciona modo master/slave/loopback	✓	
AT+PSWD	Altera a senha do módulo	✓	✓
AT+UART	Altera a velocidade (baud rate)	✓	✓
AT+RMAAD	Remove a lista dos dispositivos pareados	✓	
AT+INQ	Inicia a varredura por dispositivos BT	✓	
AT+PAIR	Efetua o pareamento com BT remoto	✓	
AT+LINK	Efetua a conexão com o BT remoto	✓	

[1] troque % pelo número do módulo.

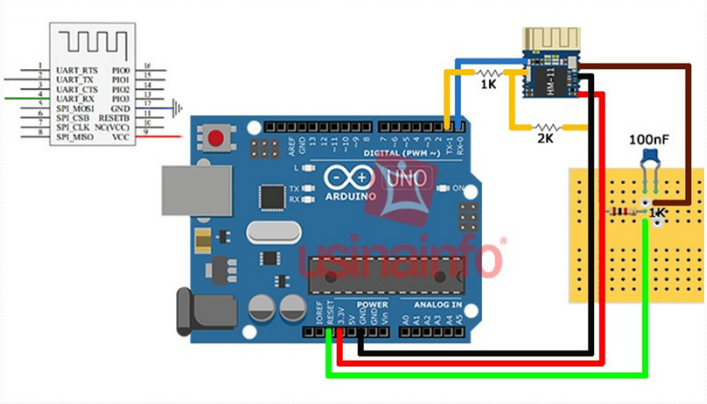
Módulo HM-11 (Bluetooth)

- Comunica com iPhones/iPads e dispositivos Android, diferentemente dos modelos HC-05, HC-06 e HC-08 que são destinados a um único sistema
- Especificação:
  - CI de comunicação: CC2541; alimentação: 3.3VDC; versão: Bluetooth 4.0 BLE; banda: ISM de 2,4 GHz; alcance: 10 m em área aberta; modulação: GFSK (*Gaussian Frequency-Shift Keying*); segurança: autenticação e criptografia; potência TX: 23-6 dBm, 0 dBm, 6 dBm (modificável via comando AT); sensibilidade RX: -93 dBm; comunicação DTE-DCE: UART (TTL); taxa de comunicação padrão: 115200 bps; corrente na TXO: 15 mA; corrente na RXO: 8,5 mA; corrente em sono profundo: 600 uA; banda de frequência: 2.402G ~ 2.480G; taxa dados (máx): 1 Mbps; impedância de entrada de RF: 50 Ohm; cristal OSC de banda base: 16 MHz; temperatura de funcionamento: -40 ~ +65 ° C; senha padrão (PIM): 000000 ou 123456; dimensões (CxLxA): 18,7x13,5x1,6mm; peso: 0,5 g.
- Módulos
  - HM-01 a HM-09 (Bluetooth Vs. 2.1) - usam chip CSR
  - HM-10 a HM-15 (Bluetooth Vs. 4.0 BLE) - usam chip TI

[1] troque % pelo número do módulo.

Módulo HM-11 (Bluetooth)

- Permite comunicação com iPhones/iPads e dispositivos Android, diferentemente dos modelos HC-05, HC-06 e HC-08 que são destinados a um único sistema



Fontes

- Using the I2C Bus, www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html
- lusorobotica.com/index.php/topic,33.0.html
- HMC6352 bússola digital + Arduino Diecimila, http://lusorobotica.com/index.php/topic,36.0.html
- Como conectar microcontroladores múltiplos de Arduino com o I2C, http://hacknmod.com/hack/how-to-connect-multiple-arduino-microcontrollers-using-i2c/pt/
- DS1307 example, http://tronixstuff.wordpress.com/2010/10/20/tutorial-arduino-and-the-i2c-bus/
- http://filipeflop.wordpress.com/