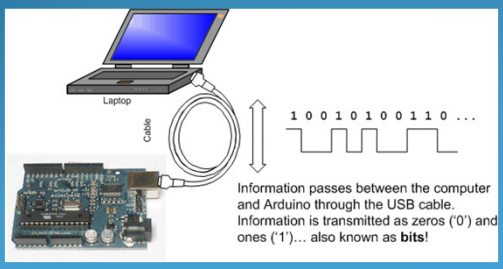
UART
I2C

Comunicação Serial com Arduino

Prof. Cláudio A. Fleury

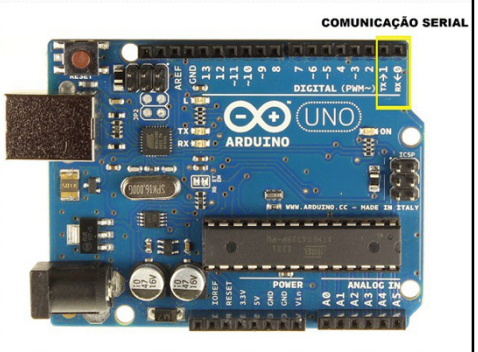
Nov-2011



55 Slides

Pinos para Comunicação Serial

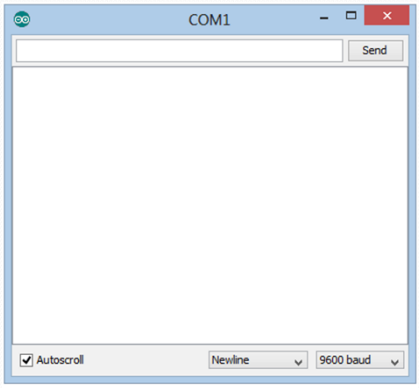
- O sinal de comunicação no Arduino UNO, é um sinal TTL (5V) gerado pela UART* do microcontrolador
- Para comunicação com um PC ou outro dispositivo que não tenha o mesmo nível de tensão é necessário um conversor de nível: TTL/ RS232, TTL/RS485, TTL/USB, TTL/3,3V...



* *Universal Asynchronous Receiver/Transmitter*

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.



Prof. Cláudio A. Fleury - Nov-2011

3

Peer-to-Peer = P2P

Comunicação Ponto a Ponto

- Comandos
  - Configuração da Porta Serial

```
// configura a porta serial do Arduino
Serial.begin(9600, SERIAL8N1);
```

  - Qtde de bytes disponíveis para leitura na Porta Serial

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

  - Função executada quando existir dados disponíveis

```
// executada qdo houver dados disponíveis
void serialEvent() {
  // comandos
}
```

default →

SERIAL_5N1
SERIAL_6N1
SERIAL_7N1
SERIAL_8N1
SERIAL_5N2
SERIAL_6N2
SERIAL_7N2
SERIAL_8N2
SERIAL_5E1
SERIAL_6E1
SERIAL_7E1
SERIAL_8E1
SERIAL_5E2
SERIAL_6E2
SERIAL_7E2
SERIAL_8E2
SERIAL_5O1
SERIAL_6O1
SERIAL_7O1
SERIAL_8O1
SERIAL_5O2
SERIAL_6O2
SERIAL_7O2
SERIAL_8O2

Prof. Cláudio A. Fleury - Nov-2011

4

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);
}
```

Prof. Cláudio A. Fleury - Nov-2011

9

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);
}
```

Prof. Cláudio A. Fleury - Nov-2011

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

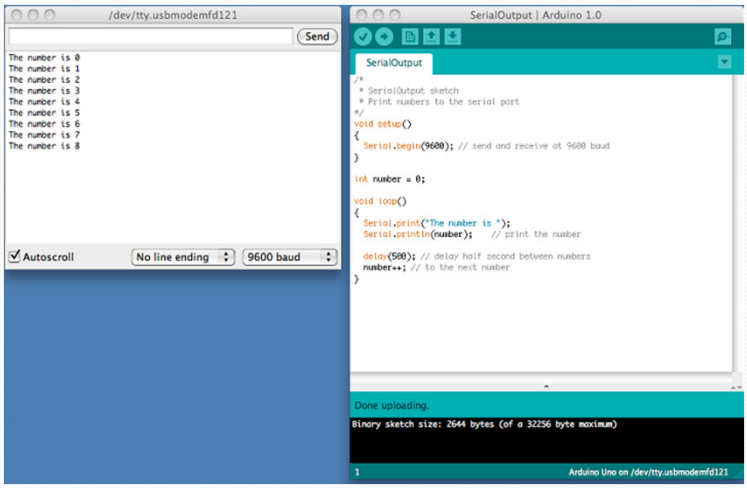
```
/*
 *  Saída Serial: envia valores numéricos p/ porta serial
 */

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 uma passada do *loop*() se existir algum byte disponível no *buffer* de entrada da Porta Serial
- Quando novos dados seriais chegam, o *sketch* adiciona-os numa 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;
  }
}
```

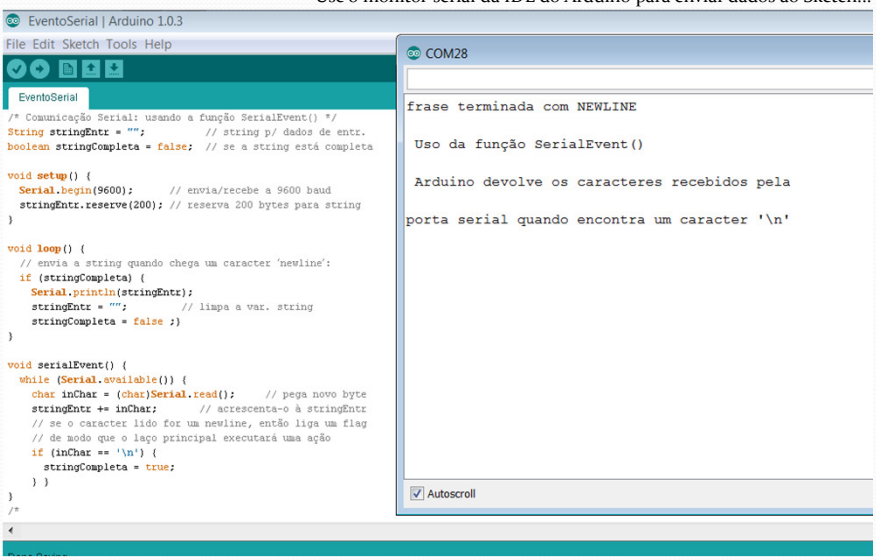
Prática 1

```
/*
  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...

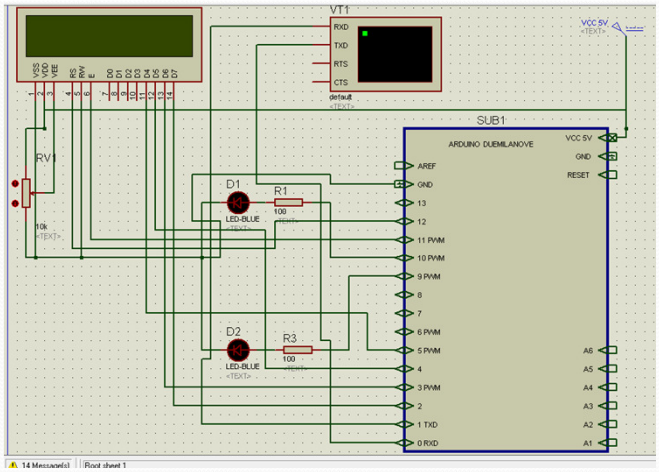


Prática 2

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
}
```

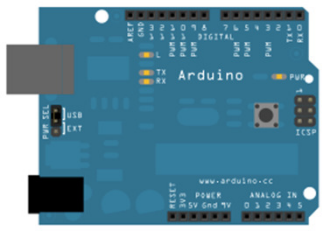
Prática 3

Jogo de Adivinhação

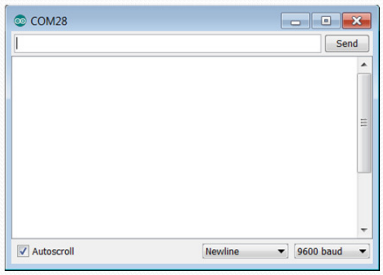
Prática 3

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

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

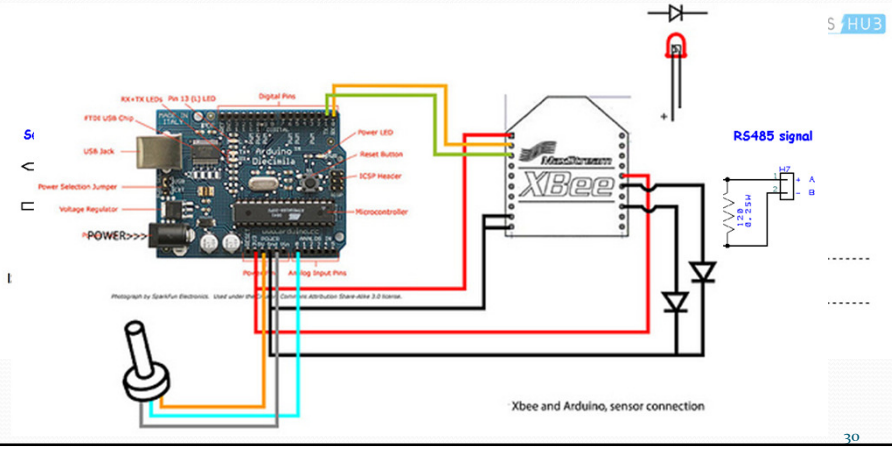
- Comandar remotamente o brilho de um LED... (continuação)

```
/* 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(); } }
}
```

Comunicação Serial em Rede

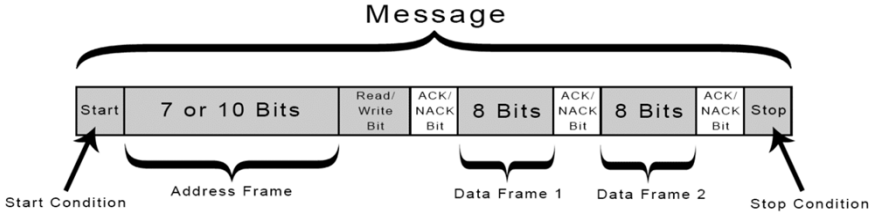
- **Barramentos** - interligação de vários nós de comunicação
  - **I²C** - *Inter-Integrated Circuit Bus*
    - Curto alcance, conexão de componentes numa mesma placa de circuito impresso (PCB)



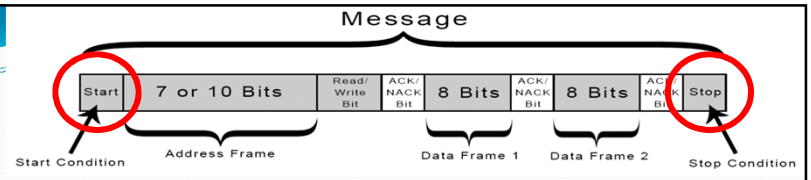
Xbee and Arduino, sensor connection

I2C

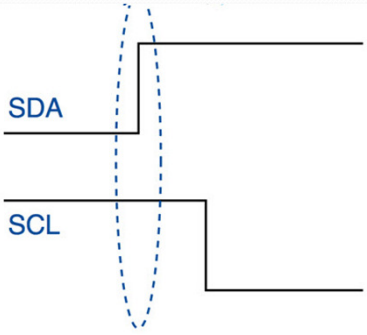
- Funcionamento
  - Dados são transferidos via mensagens
  - Mensagens são divididas em quadros de dados (frames)
  - Cada mensagem possui um quadro com o endereço do escravo e um ou mais frames com os dados que se quer transmitir
  - A mensagem também inclui condições de início e parada, bits de leitura/gravação e bits ACK/NACK entre cada *frame*:

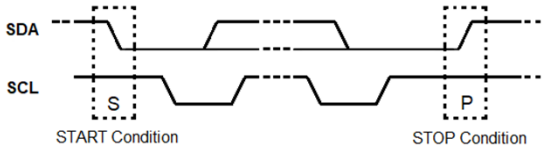


I2C



- Funcionamento
  - **Condição de início** (*Start*) de comunicação: SDA muda de HIGH para LOW antes que SCL mude de HIGH para LOW
  - **Condição de Parada** (*Stop*) de comunicação: SDA muda de LOW para HIGH depois que SCL muda de LOW para HIGH

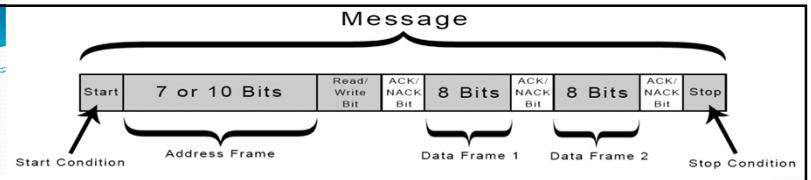




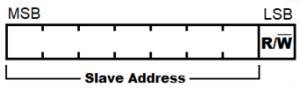
Prof. Cláudio A. Fleury - Nov-2011

32

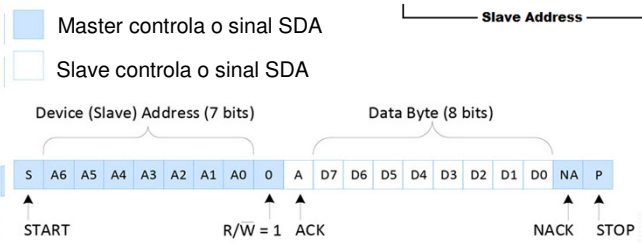
I2C



- Funcionamento
  - **Quadro de Endereço do *Slave***: 7 ou 10 bits que identifica o escravo com o qual o mestre deseja falar
  - **Bit de leitura/gravação**: especifica se o mestre está enviando dados (gravando = LOW) para o escravo ou solicitando dados do mesmo (lendo = HIGH)
  - **Bit ACK/NACK**: todo quadro é seguido de um bit de reconhecimento (ACK = LOW) ou não reconhecimento (NACK = HIGH). Se quadro for recebido com sucesso, então um bit ACK é enviado ao *Master* pelo *Slave* endereçado
  - Obs.: bit mais significativo (MSB) é enviado primeiro



Mestre recebendo dados do Escravo



Prof. Cláudio A. Fleury - Nov-2011

33

Comunicação Serial em Rede

- I²C para Arduino
  - Biblioteca ***Wire***: `#include <Wire.h>`
    - 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 **10** bits
      (oitavo bit na versão de **7** bits determina a operação: RD ou WR)

Comunicação Serial em Rede

- I²C para Arduino com Bibliot. **Wire**
  - **Wire.begin(endereço)** - inicia a biblioteca ***Wire*** e conecta o dispositivo ao barramento I2C como mestre ou escravo. O parâm. ***endereço*** do escravo (7 bits) é opcional e, se não for usado, o dispositivo se liga ao barramento como mestre: **Wire.begin()**
  - **Wire.read()** - lê um byte recebido, transmitido de um dispositivo escravo após uma chamada à função **requestFrom()** ou transmitido de um mestre para um escravo
  - **Wire.write()** - envia dados (para escravo ou mestre).
    Escravo envia dados ao mestre quando a função **Wire.requestFrom()** for usado pelo mestre.
    Mestre envia dados para escravo usando **Wire.write()** entre chamadas de **Wire.beginTransmission()** e **Wire.endTransmission()**
  - **Wire.beginTransmission(endereço)** - inicia uma transmissão para o escravo c/ **endereço**. Depois, construa a fila de bytes a ser transmitido com a função **Wire.write()** e, em seguida, transmita-os chamando a função **endTransmission()**

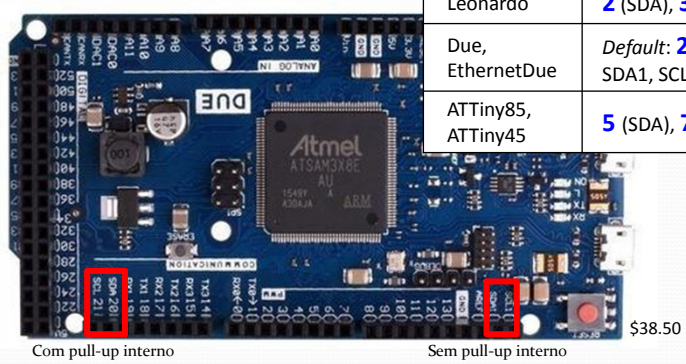
Comunicação Serial em Rede

- I²C para Arduino com Bibliot. **Wire**
  - **Wire.onRequest()** - função chamada quando um mestre solicita dados ao escravo via **Wire.requestFrom()**. Nessa função usamos **Wire.write()** para enviar dados ao mestre
  - **Wire.onReceive()** - função chamada quando um dispositivo escravo recebe dados de um mestre. Nessa função usamos **Wire.read()** para ler os dados enviados pelo mestre
  - **Wire.requestFrom(endereço, quantidade)** - função usada no mestre para solicitar bytes de um dispositivo escravo. A função **Wire.read()** é usada para ler os dados enviados pelo escravo. O **endereço** de 7 bits do escravo. A **quantidade** é o número de bytes a ser retornado pelo escravo

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

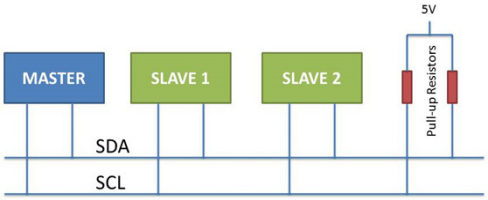
Com pull-up interno

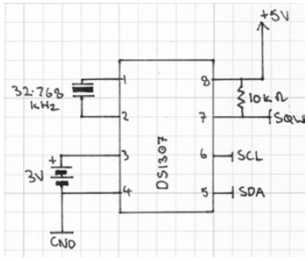
Sem pull-up interno

Comunicação Serial em Rede

- *Inter-Integrated Circuit Bus* (I^2C)
  - Exemplo de CI que usa I^2C
    - Relógio de Tempo Real: **DS1307**

RTC – *Real Time Clock*





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.

```
// 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) );
}
```

Exemplo de código C:

```
#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();   // nova versão: Wire.read();
*minute     = bcdToDec(Wire.receive();
*hour       = bcdToDec(Wire.receive();
*dayOfWeek  = bcdToDec(Wire.receive());
*dayOfMonth = bcdToDec(Wire.receive());
*month      = bcdToDec(Wire.receive());
*year       = bcdToDec(Wire.receive());
```

Prof. Cláudio A. Fleury - Nov-2011

40