CENTRO UNIVERSITÁRIO DE ANÁPOLIS

UNIEVANGÉLICA

CURSOS SUPERIORES DE COMPUTAÇÃO

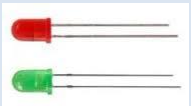
Professor: Alexandre Moraes Tannus - 2018

Arduino: Comunicação serial

1. OBJETIVOS:

- Conhecer os fundamentos de comunicação serial
- Implementar comunicação serial no Arduino.

2. MATERIAIS:

	Uma placa Arduino Uno
	Dois LEDs
	Resistores
	Potenciômetro

3. PARTE TEÓRICA

A interface serial UART (*Universal Asynchronous Receiver Transmitter*) do Arduino é responsável por enviar e receber dados para e do microcontrolador. Para isso são utilizados os pinos nomeados como *Tx* (transmissão) e *Rx* (recepção). No Arduino Uno esses pinos são, respectivamente, *D0* e *D1* e também são utilizados pelo USB para o *upload* dos programas (*sketches*) para o microcontrolador. Em outras placas o número de portas seriais chega a 4, como mostra a Tabela 1

Tabela 1 - Interfaces UART

Placa	Número de portas seriais	Detalhes
Uno	1	Rx é D0 e Tx é D1. Essas portas também são usadas pelo USB.
Leonardo	2	USB dedicado. Segunda porta serial. Rx é D0 e Tx é D1.
Mega 2560	4	USB usa D0 e D1. Três outras portas: Serial1 nos pinos 19 (Rx) e 18 (Tx), Serial2 nos pinos 17 (Rx) e 16 (Tx), e Serial3 nos pinos 15 (Rx) e 14 (Tx).
Due	4	USB dedicado. Porta serial 0 usa D0 (Rx) e D1 (Tx). Três outras portas: Serial1 nos pinos 19 (Rx) e 18 (Tx), Serial2 nos pinos 17 (Rx) e 16 (Tx), e Serial3 nos pinos 15 (Rx) e 14 (Tx).

Para realizar uma comunicação serial é necessário que seja definida previamente a taxa de transmissão de dados que será utilizada tanto pelo transmissor como pelo receptor. Esta taxa é conhecida como *baud rate* e pode assumir os valores de 300, 1200, 4800, 9600, 14400, 19200, 28800, 38400, 57600 e 115200, sendo a taxa *default* do Arduino 9600 *bauds*.

3.1. Comandos seriais

3.1.1. $Serial.begin(baud\_rate)$

O comando $Serial.begin(baud\_rate)$ é utilizado para iniciar uma comunicação serial. O parâmetro de entrada *baud_rate* é utilizado para definir a taxa de transmissão que será utilizada. Caso a placa possua mais de uma porta serial é necessário especificar qual a interface que será utilizada. Para utilizar a interface 2 do Arduino Mega 2560 o comando a ser utilizado é $Serial2.begin(baud\_rate)$. Normalmente este comando é colocado na função **setup**

3.1.2. $Serial.available()$

A função $Serial.available()$ é utilizada para verificar se existem bytes a serem lidos na porta serial. Caso existam ela retorna o número de bytes. Caso contrário é retornado o valor 0, que equivale a um valor lógico falso em linguagem C. Este comando deve ser inserido na função **loop** do *sketch*, conforme mostra o Código 1

```
void loop( ) {
   if (Serial.available) {
      // comandos a serem executados
   }
}
```

Código 1 - Comando Serial.available()

3.1.3. $Serial.read()$

A função $read()$ lê a porta serial e retorna o primeiro byte disponível. Caso não haja informação na porta o retorno é igual a *-1*

3.1.4. $Serial.write()$

A função $write()$ é utilizada para escrever dados na porta serial. Os dados podem ser enviados como *bytes* ou conjuntos de *bytes*.

Três sintaxes são permitidas, conforme mostra a Tabela 2

Tabela 2 - Sintaxe do comando write( )

Sintaxe	Parâmetros
$Serial.write(val)$	val: *byte* symples
$Serial.write(str)$	str: string (série de *bytes*)
$Serial.write(buf, len)$	buf: *array* com vários *bytes* len: tamanho do *buffer*

3.1.5. $Serial.print()$ e $Serial.println()$

As funções *print()* e *println()* são utilizadas para imprimir dados na porta serial. A diferença entre elas é que a segunda adiciona os caracteres de retorno de carro (\r) e de nova linha (\n).

3.2. Biblioteca SoftwareSerial

Caso sejam necessárias mais portas seriais em um determinado projeto a biblioteca ***SoftwareSerial*** pode ser utilizada, permitindo a utilização de outros pares de pinos digitais para realizar a comunicação serial. Para utilizar esta biblioteca é necessário incluí-la no *sketch* através da diretiva *#include*. Além disso, um objeto *SoftwareSerial* deve ser criado. As funções desta biblioteca são bem similares às presentes na interface serial padrão. O uso da biblioteca é mostrado no Código 2

```
#include  <SoftwareSerial.h>

#define RX 10
#define TX 11

SoftwareSerial mySerial(RX, TX);

void setup( ) {
    mySerial.begin(9600);
    mySerial.println("Olá mundo");
}
```

Código 2 - Uso da biblioteca SoftwareSerial

4. PARTE PRÁTICA

4.1. Prática 01 – Imprimindo valores na interface serial

Nesta prática será realizada a leitura de uma entrada analógica e será impresso na porta serial o valor lido em vários formatos numéricos diferentes. Para isso monte o circuito da Figura 1 e utilize o sketch do Código 3

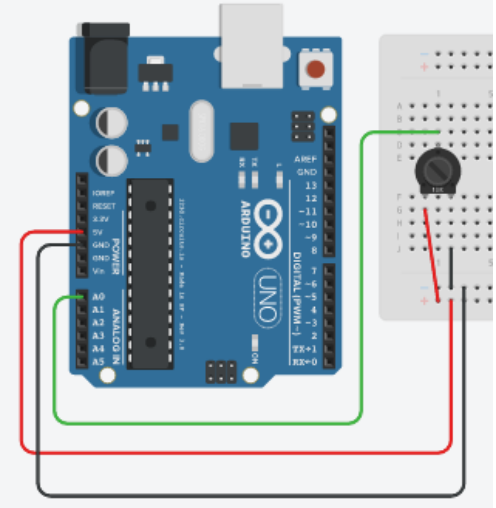


Figura 1 - Circuito da prática 01

```
int analogValue = 0;

void setup( ) {
   Serial.begin(9600);
}

void loop ( ) {
   analogValue = analogRead(A0);
   Serial.println(analogValue);
   Serial.println(analogValue, DEC);
   Serial.println(analogValue, HEX);
   Serial.println(analogValue, OCT);
   Serial.println(analogValue, BIN);

   delay(1000);
}
```

Código 3 - Código da prática 01

4.2. Prática 02 – Enviando dados para a porta serial

A prática 2 utilizará a interface serial para enviar dados para o Arduino para controlar a intensidade luminosa de um LED utilizando PWM. O circuito para esta prática é mostrado na Figura 2 e sketch no Código 4

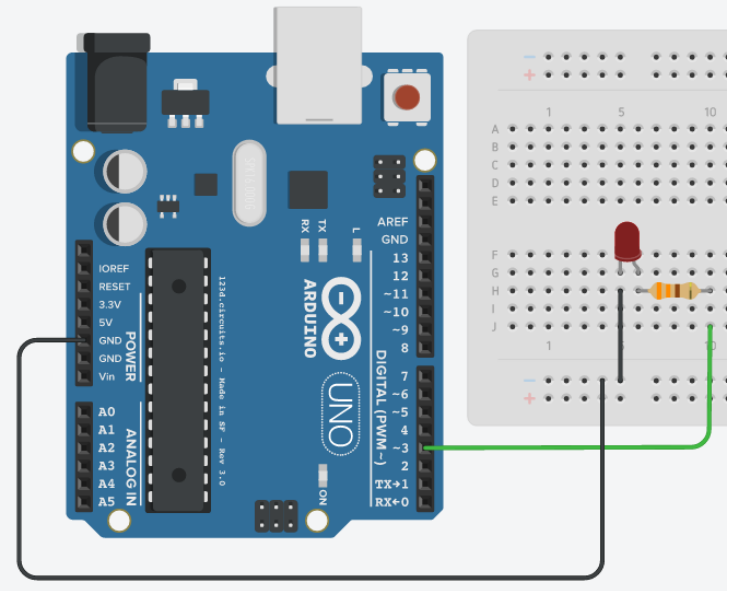


Figura 2 - Circuito da prática 02

```
#define LED 3

int valorPWM;

void setup( ) {
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
}

void loop ( ) {
    if (Serial.available( )) {
        valorPWM = Serial.parseInt( );
        Serial.println(valorPWM);
    }
    analogWrite(LED, valorPWM);
}
```

Código 4 - Código da prática 02

5. REFERÊNCIAS

BANZI, Massimo. *Getting Started with Arduino*. 2ª ed. Sebastopol: O'Reilly, 2011.

EVANS, Martin; NOBLE, Joshua; HOCHENBAUM, Jordan. *Arduino em Ação*. 1ª ed. [S.l.]: Novatec, 2013.

MONK, Simon. *Programação com Arduino: começando com Sketches*. 1ª ed. Porto Alegre: Bookman, 2013.