No capítulo 7 do livro Arduino Computer Vision Program, trata-se o assunto sobre como estabelecer uma conexão sólida entre o computador e o Arduino. Estudaremos este capítulo para ter uma visão geral das diferentes possibilidades de comunicação com o arduino.

No contexto da visão computacional, a comunicação entre o Arduino e o computador é necessária para combinar os dados do sensor com a visão computacional. Esta conexão também é requerida para traduzir o resultado do processo de visão computacional em uma saída física.

Também observa-se as melhores práticas de troca de informação entre dispositivos. Procura-se projetar um canal de comunicação robusto entre o Arduino e um agente inteligente. Estuda-se tanto as interfaces de comunicação com fio como sem fio. Além da conexão física, a ideia por trás da troca de informação e modelagem de informação é muito importante.

**Comunicação com o Arduino**

A Arduino Uno R3 é uma placa de desenvolvimento construída com um microcontrolador ATmega328. As capacidades de comunicação dependem desse microcontrolador, que possui uma interface periférica serial, *Serial Peripheral Interface* (SPI) e um módulo transmissor receptor síncrono assíncrono *universal, Universal Synchronous Asynchronous Receiver Transmitter* (USART).

Um dos pontos fortes do Arduino é a grande comunidade de suporte e as bibliotecas de software livre para várias Shields e módulos. Possui um rico conjunto de Shields de comunicação que permitem ao usuário escolher a mais apropriada. A união de Shields e bibliotecas de software livre aceleram a prototipação de aplicações.

Em uma abordagem de nível de sistema é possível dizer que a interface de comunicação é requerida para transferência de informação entre dispositivos. Neste esquema tanto o Arduino quanto o controlador visual devem concordar com as propriedades físicas e com o conteúdo dos dados transferidos.

Quando o Arduino envia a informação do sensor ao controlador visual, este deve conhecer o formato do dado e ser capaz de entendê-lo. Este consenso é possível através de um protocolo de comunicação no nível de aplicação. Ao implementar um bom protocolo de comunicação, é possível transferir dados em uma forma eficiente. **Estudaremos o protocolo de comunicação de nível de aplicação.**

Vale destacar que é possível utilizar o protocolo de comunicação em qualquer interface de comunicação física, seja com fio o sem fio. Com isso, a comunicação de nível de aplicativo é independente do hardware. O que significa que com o mesmo software, é possível enviar informação através de uma conexão cabeada o sem fio.

Em um esquema de comunicação de duas vias, tanto o sistema Arduino como o controlador visual devem ser capazes de empacotar dados em uma mensagem antes de enviá-lo. De maneira similar, é necessário analisar a mensagem de protocolo recebida para recuperar o dado dentro da mensagem. Isto significa que precisamos implementar um empacotador de mensagens e um analisador de mensagens.

Na comunicação do Arduino com o computador, ambas plataformas devem trabalhar no mesmo tipo de canal de comunicação. A melhor forma de conectar o Arduino a um microcomputador é usando a interface serial universal, *Universal Serial Bus* (USB).

Falta.

**Comunicação com Fio**

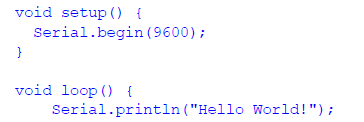
**Comunicação via USB**

A maneira mais simples de se comunicar com o Arduino é utilizando a interface serial USB integrada. Mediante a comunicação serial, é possível intercambiar dados entre a placa Arduino e outros dispositivos tais como computadores com capacidade de comunicação serial. Serial significa que um byte é transmitido de cada vez. Embora possa parecer lento, é uma maneira muito comum e relativamente fácil de transmitir dados.

Todos os Arduinos possuem pelo menos uma interface serial. A comunicação serial é realizada por meio de um módulo USART. Os pinos digitais 0 e1 estão conectados a um CI que converte os sinais para USB. Neste caso, o pino 0 é usado para RX (receber) e o pino 1 é usado para TX (transmitir).

A interface USB do Arduino Uno trabalha com o perfil Virtual COM port. Para estabelecer um canal de comunicação, basta conectar uma extremidade do cabo USB no Arduino e a outra extremidade no computador

Comunicação em uma via.- A comunicação em uma via consiste em enviar dados a partir do Arduino. Nesta caso, enviaremos uma mensagem Hello World! a partir do Arduino para o computador. Monitoramos a mensagem usando o Arduino IDE. Crie um novo sketch no Arduino IDE e introduza o seguinte código:





Carregue o sketch no seu Arduino Uno R3. O código deve enviar uma mensagem Hello World! por segundo.

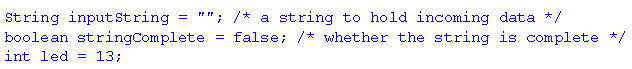
O método Serial.begin() é usado para iniciar uma comunicação serial com uma taxa específica de bauds. A taxa baud define a taxa de transferência de dados da comunicação serial. Neste caso a porta serial foi iniciala em 9600 baud. Assim o Arduino poderá enviar 9600 símbolos por segundo para transferir a mensagem. É possível ajustar a taxa baud a uma de várias opções predefinidas.

O método Serial.println() é usado para enviar uma string através do porto serial. Neste caso uma mensagem Hello World! simples. Para monitorar esta mensagem, podemos utilizar o Arduino IDE. Precisamos identificar o porto serial que está sendo utilizado. Para isso escolha **Arduino IDE | Tools | Serial Port**. Depois abra o monitor serial, para isso escolha **Arduino IDE | Tools | Serial Monitor.** O monitor serial mostrará a seguinte tela:

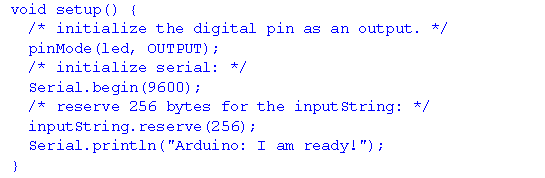


Comunicação em duas vias.- A comunicação em duas vias significa que além de enviar dados a partir do Arduino é possível que o Arduino receba dados, a partir do computador por exemplo. Por exemplo, é possível enviar dados do monitor serial do Arduino IDE para o Arduino, para isso, é preciso que o Arduino seja capaz de receber dados sobe uma linha serial. O Arduino deve ser capaz de entender um conjunto específico de comandos e reagir a cada comando de maneira significativa.

Temos aqui o código requerido para formar o sketch:

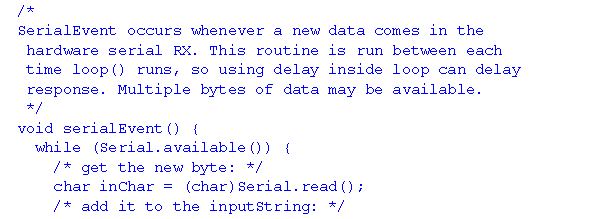


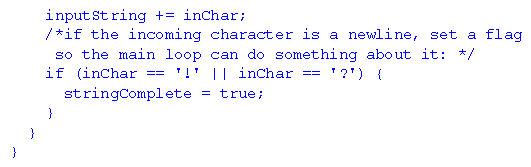
Primeiro, criamos a variável inputString, uma String que serve para armazenar os dados de entrada e uma variável booleana stringComplete para indicar o término da recepção da string. Finalmente, designamos o pino 13 para utilizar ele junto ao método digital.write para acender e apagar um led.



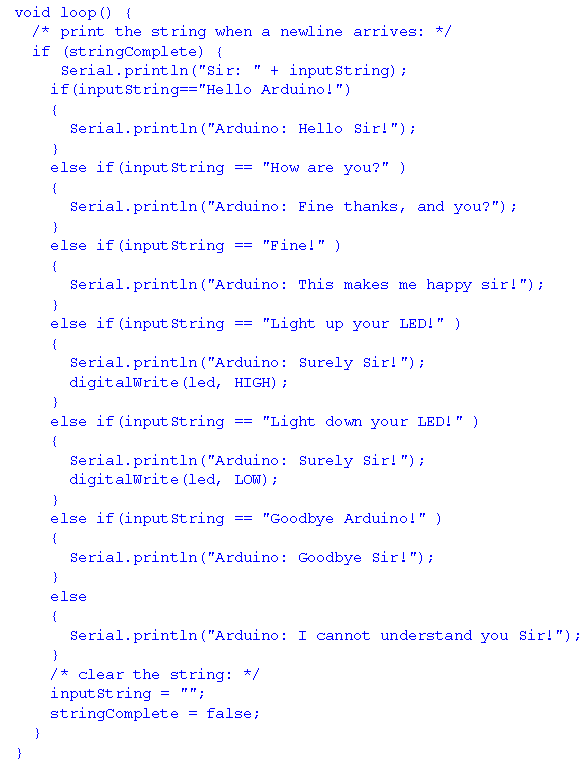
Em seguida temos a função de setup(). Primeiro, inicializa-se o pino do led como saída, para isso usa-se a função pinMode(). Depois, fixa-se a taxa de transmissão da porta serial (em bauds). Porque 9600??? Reserva-se 256 bytes para a string de entrada. Finalmente, envia-se uma mensagem de boas vindas. Até aqui não existe nada novo. No entanto, o código possui uma grande diferença com relação ao código anterior.

A função serialEvent() introduz o uso conduzido por eventos, *event-driven*, para a captura de mensagens de entrada.

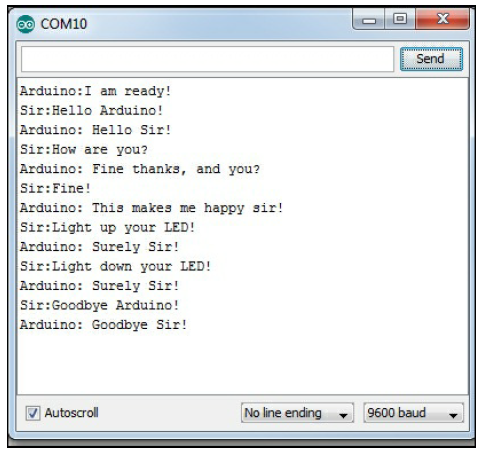




Quando uma nova string entra na linha RX do Arduino, a função de escuta serialEvent() começa a registrar a string enviada desde o computador. Cada caráter é adicionado um depois do outro na variável inputString. A condição if (inchar == ‘!’ || inchar == ‘?’), verifica se string de entrada terminou. Buscar que significam os símbolos ‘!’ e ‘?’. Se a condição é verificada a flag stringComplete é fixada em true.



No loop principal, o sketch espera até que uma string esteja completa, para isso confere o valor da flag booleana stringComplete. A string enviada é reproduzida no monitor serial e dependendo da mensagem enviada uma resposta é enviada de volta. O sketch reconhece seis possíveis mensagens inclusive duas mensagens envolvendo acender e apagar o led. Caso nenhuma das mensagens seja reconhecida, uma resposta pertinente é enviada. Após a mensagem de resposta, a string de entrada é apagada e a flag booleana definida como falso.



**Comunicação via Ethernet**

Ethernet é uma família de tecnologias de redes. Com a função Ethernet no Arduino, é possível intercambiar dados de qualquer lugar usando a conexão de internet. Existem muitas possibilidades de aplicação, como controlar um robô através de uma web page ou obter um e-mail quando a imagem de um instruso é capturada quando você não está em casa. **Arduino Ethernet Shield** permite conectar seu Arduino à Internet ou à qualquer rede baseada em Ethernet. A figura ilustra a placa Arduino na esquerda e a Ethernet Shiled na direita.



Informação adicional sobre a Arduino Ethernet Shield pode ser obtida na seguinte página: <http://www.arduino.cc/en/Main/ArduinoEthernetShield>.

**Comunicação sem Fio**

**Comunicação via Bluetooth Low Energy**

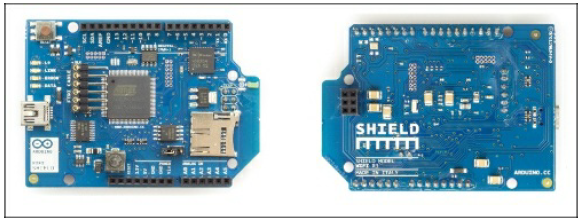
**Comunicação via ZigBee**

**Comunicação via Wi-Fi**

Wi-Fi é uma tecnologia de rede local. Os módulos Wi-Fi permitem conectar dispositivos a Internet, assim como são apropriados para comunicação local. Note que, usando a conectividade Wi-Fi, podemos controlar um robô remotamente, acessando ele pela internet. Só não esqueça de conectar o robô a uma rede local Wi-Fi. É possível fazer um robô que perambule pela casa e envie fotos de sua casa enquanto percorre os cômodos da casa.

Introduzimos a Arduino Wi-Fi Shield para usar qualquer classe de aplicação robótica sem fio que conecte através do Wi-Fi. A Wi-Fi é melhor aproveitada quando se requer velocidade de comunicação alta. Também permite conectar a redes sem fio de alto nível. A conexão sem fio para a internet faz com que a sua aplicação seja mais interessante. Se precisa conectar o Arduino a internet sem fio, é melhor utilizar a Wi-Fi Shield. A documentação sobre a Arduino Wi-Fi Shield e como utilizar ela em aplicações pode ser encontrada em:

<http://www.arduino.cc/en/Main/ArduinoWiFiShield>



A figura ilustra a placa Arduino Wi-Fi Shield.

**Comunicação via radio frequência**

Links:

http://people.cs.vt.edu/~shaffer/Book/

http://www.brpreiss.com/books/opus4/

http://www.ida.liu.se/~ulfni53/lpp/

https://www.railstutorial.org/book