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.

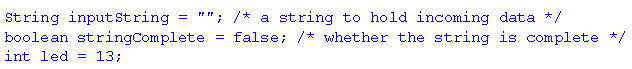
Na comunicação do Arduino com o computador, ambas plataformas devem trabalhar no mesmo tipo de canal de comunicação. A melhor forma de conectar o Arduino a um microcomputador é usando a interface serial universal, *Universal Serial Bus* (USB).

Falta.

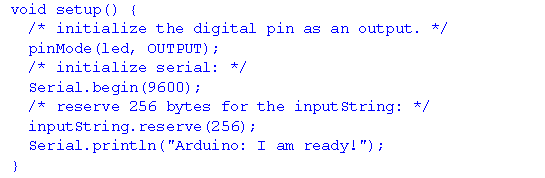
**Comunicação com Fio**

Comunicação em duas vias.- Também 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.

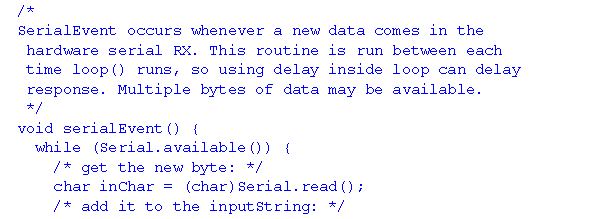
Temos aqui o código requerido para formar o sketch:

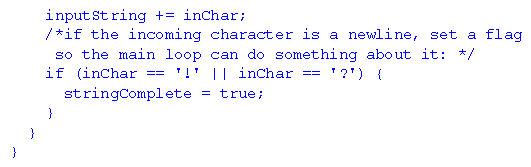


Primeiro, criamos a variável inputString, uma String que serve para armazenar os dados de entrada e uma variável booleana stringComplete para indicar o término da recepção da string. Finalmente, designamos o pino 13 para utilizar ele junto ao método digital.write para acender e apagar um led.

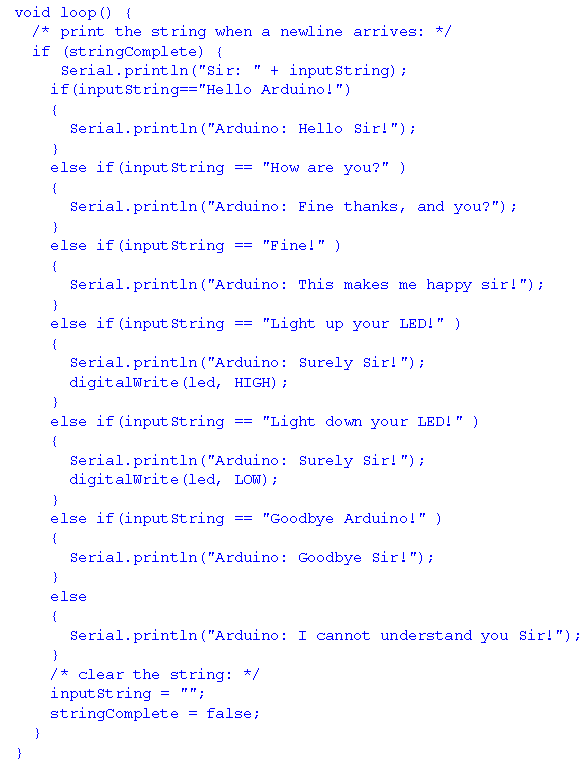


Em seguida temos a função de setup(). Primeiro, inicializa-se o pino do led como saída, para isso usa-se a função pinMode(). Depois, fixa-se a taxa de transmissão da porta serial (em bauds). Porque 9600??? Reserva-se 256 bytes para a string de entrada. Finalmente, envia-se uma mensagem de boas vindas.





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.



No loop principal, o sketch espera até que uma string esteja completa. 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.

