Ficha 3

Exercícios de preparação da actividade 3

Sítio:	Elearning UAb
Unidade curricular:	FATAC - Sensores e Actuadores (DMAD 2013-14)
Livro:	Ficha 3
Impresso por:	José Coelho
Data:	Quarta, 4 Junho 2014, 11:03

Índice

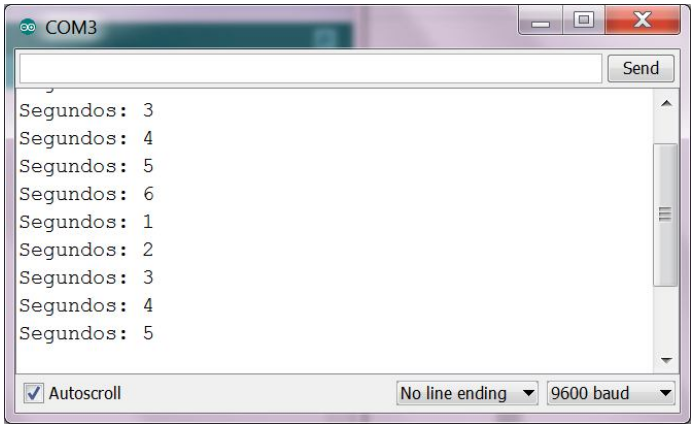
Como ler informação do Arduino no PC

A comunicação pode ser feita no Arduíno através do objeto Serial, com uma estrutura bastante simples. No programa seguinte, a cada segundo, é enviado o número de segundos desde que se carregou no botão de reset:

```
void setup()
{

	Serial.begin(9600);

}

void loop()
{

	static int segundos=0;
	if(segundos!=millis()/1000)
	{

		segundos=millis()/1000;
		Serial.print("Segundos: ");
		Serial.println(segundos);

	}

}
```

As duas únicas funções utilizadas neste programa, são o Serial.begin, e o Serial.print/println. A primeira é necessária para iniciar as comunicações, sendo na segunda função que é enviada informação. É idêntica ao print no Processing, mas apenas se pode dar um argumento. Se forem necessários dois ou mais valores, tem de se chamar a função uma vez por cada valor. É o que foi feito no exemplo em cima, em que se coloca o texto "Segundos: " e de seguida o número de segundos desde que o Arduíno foi inicializado.

Para se ver o resultado, no ambiente Arduíno, tem de se abrir o Serial Monitor (Tools » Serial Monitor):



Na figura pode-se ver que foi feito um reset após o segundo 6, voltando a contagem ao segundo 1.

A documentação de suporte do Arduíno sobre comunicação, é dada na página http://arduino.cc/en/Reference/Serial.

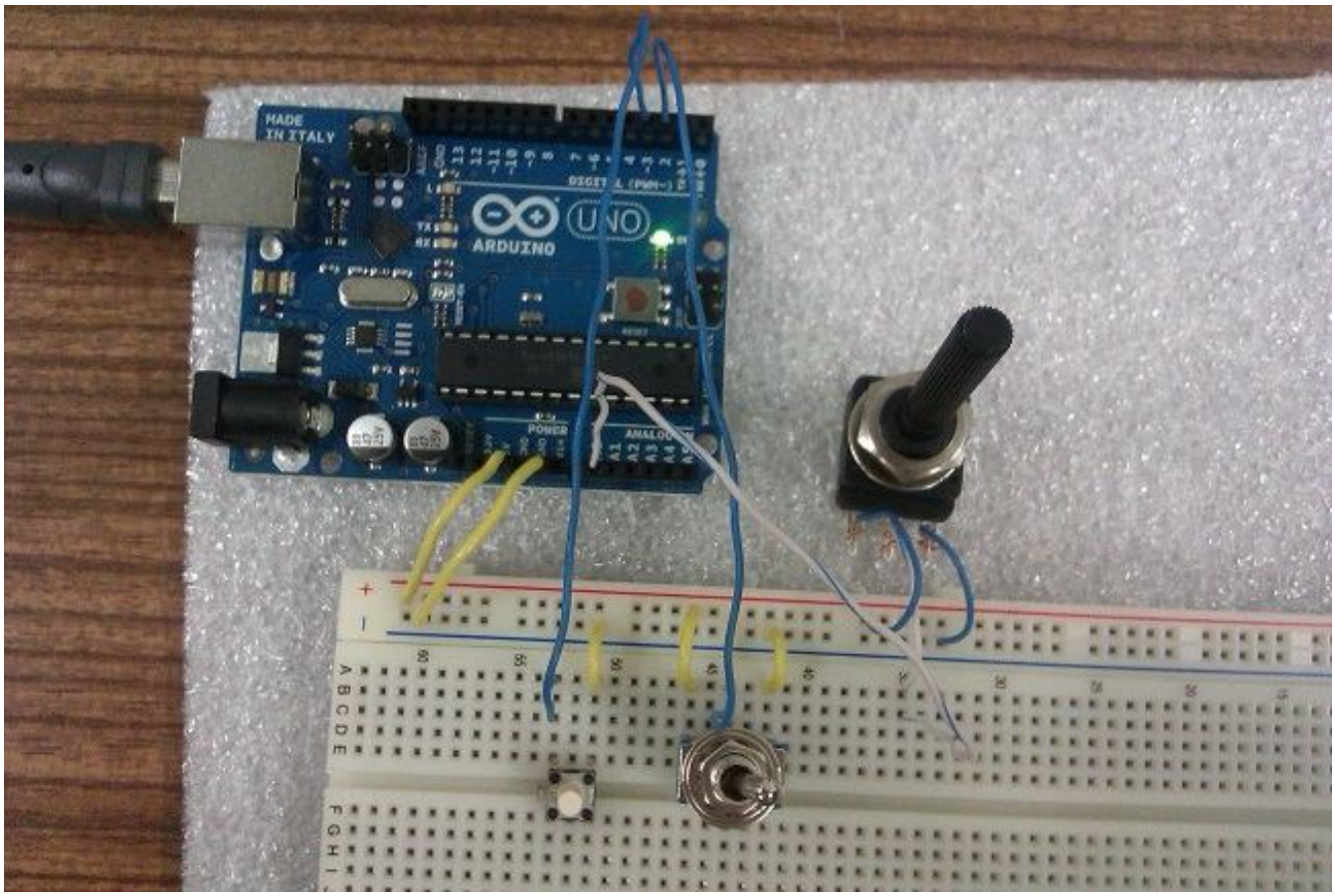
A visualização da comunicação no Serial Monitor, é útil para debug do programa do Arduíno, mas pouco interessante para montagens e comunicação com o computador, daí o resto da ficha centrar-se na comunicação com um programa em Processing, de modo a poder-se fazer uso de todos os recursos do computador, e do Arduíno, numa só instalação.

Como ler informação dos sensores no processing?

No Processing pode-se ler informação de uma entrada em série, estando a documentação referente a comunicação em http://processing.org/reference/libraries/serial/Serial.html. A versão do Processing 2 de 64 bits não suporta a comunicação em série, pelo que é necessário utilizar a versão 32 bits. Simulando o Serial Monitor do Arduíno num programa em Processing, ficamos com o seguinte programa:

```
import processing.serial.*;

Serial myPort;

void setup()
{

	println(Serial.list());
	myPort = new Serial(this, Serial.list()[0], 9600);

}

void draw()
{

	while (myPort.available() > 0) {

		String inBuffer = myPort.readString();

		if (inBuffer != null) {

			print(inBuffer);

		}

	}

}
```

A impressão da lista de comunicações é opcional, mas convém colocar para verificar se não há algum problema. Neste caso o Processing fica a ler a informação que chega do Arduíno, já em texto e pronto a mostrar para o utilizador. Vamos agora pretender ler do Arduíno informação sobre o estado de sensores. Para tal vamos montar um botão, interruptor e um potenciómetro, como se pode ver na imagem seguinte:



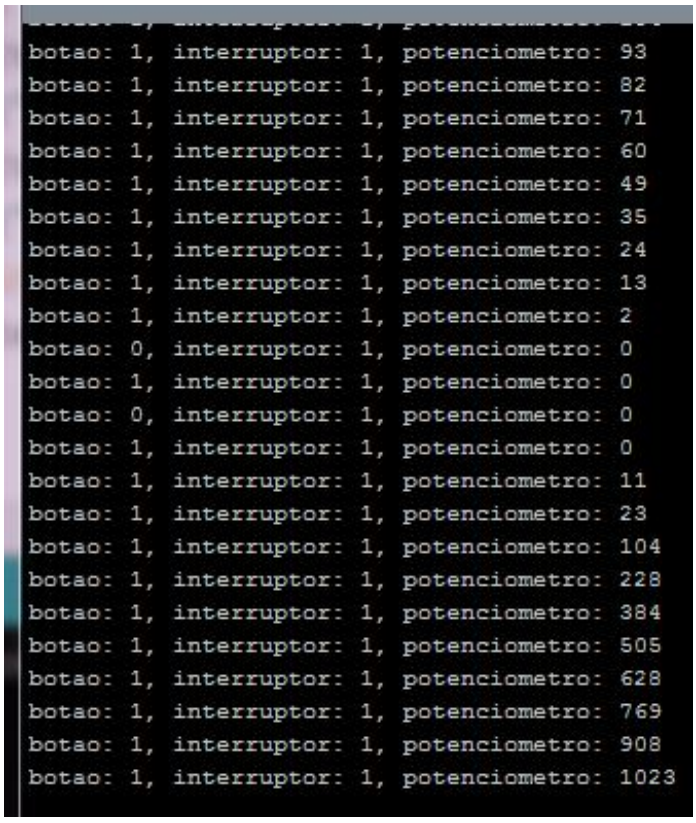
O programa no Arduíno passa a ler informação, e a enviar a informação ao Processing:

```
int botao=3, interruptor=2, potenciometro=A0;

void setup()
{

	pinMode(botao, INPUT_PULLUP);
	pinMode(interruptor, INPUT);

	Serial.begin(9600);

}

void loop()
{

	static int vbotao=LOW;
	static int vinterruptor=LOW;
	static int vpotenciometro=0;
	static int venviado=0;
	int atualizado=0;

	if(vbotao!=digitalRead(botao)) {

		vbotao=digitalRead(botao);
		atualizado=1;

	}
```

```
    if(vinterruptor!=digitalRead(interruptor)) {

        vinterruptor=digitalRead(interruptor);
        atualizado=1;

    }

    if(vpotenciometro!=analogRead(potenciometro)) {

        // enviar informação apenas se for muito diferente

        vpotenciometro=analogRead(potenciometro);
        if(abs(vpotenciometro-venviado)>10)

            atualizado=1;

    }

    if(atualizado) {

        Serial.print("botao: ");
        Serial.print(vbotao);
        Serial.print(", interruptor: ");
        Serial.print(vinterruptor);
        Serial.print(", potenciometro: ");
        Serial.println(vpotenciometro);
        venviado=vpotenciometro;

    }

}
```

Convém controlar a informação enviada, de modo a não enviar informação duplicada. No caso do potenciómetro, apenas é enviada nova informação, no caso do valor anteriormente enviado, ser muito diferente do atual, caso contrário não envia nada já que a diferença poderá ser devido a uma oscilação da leitura.

Mantendo o programa no Processing intacto, o resultado é o seguinte:



Repare-se que há sempre alteração dos valores de leitura, caso contrário não é feita uma nova comunicação apenas para reproduzir a informação já enviada.

Pretende-se no entanto, utilizar os valores lidos do Arduíno, no Processing, e não apenas mostrar um texto enviado do Arduíno. Para tal, há que processar a informação recebida. Pode-se eventualmente enviar apenas informação numérica, e nesse caso a comunicação é mais rápida, mas no caso de se pretender ler mas que uma variável, tem que se fazer à mesma um processamento da informação de entrada, pelo que deixa-se aqui um exemplo de como se pode processar a informação chegada do Arduíno, mantendo a legibilidade para o observador:

```
import processing.serial.*;

Serial myPort;
String entrada; // string de entrada

void setup()
{

	// List all the available serial ports:
	println(Serial.list());
	// Open the port you are using at the rate you want:
	myPort = new Serial(this, Serial.list()[0], 9600);

}

void draw()
{

	while (myPort.available() > 0) {

		String inBuffer = myPort.readString();
```

```
            if (inBuffer != null) {

                entrada = entrada + inBuffer;
                print(inBuffer);


                // verificar se tem dados para processar
                String [] tokens=splitTokens(entrada, " ,:\n\r");
                for(int i=0;i<=tokens.length-6;i++)

                    if(tokens[i].equals("botao"))
                    {

                        // processar potenciómetro
                        int valor=int(tokens[i+5])/4;
                        // processar interruptor (inverte o valor)
                        if(int(tokens[i+3])!=0)

                            valor=255-valor;

                        // processar botão (mostra o valor ou não)
                        if(int(tokens[i+1])!=0)

                            background(valor);

                        else

                            background(0);

                        // limpar dados
                        entrada="";

                    }

            }

        }

}
```

Este programa, vai trocar o background conforme o valor do potenciómetro, sendo esse valor invertido pelo interruptor. No caso do botão estar em baixo, o valor fica preto. Este programa em no entanto um problema. Como o Arduíno envia informação em série, o Processing lê a cada momento a informação recebida, poderá acontecer que o Processing analise informação incompleta, e a dê como completa. Esse caso ocorre neste exemplo: "botao: 1, interruptor: 0, potenciometro: 622". Esta informação enviada, pode ser lida pelo Processing "botao: 1, interruptor: 0, potenciometro: 62" ou mesmo "botao: 1, interruptor: 0, potenciometro: 6", pelo que é necessário adicionar na informação enviada pelo Arduíno, uma parte constante para saber que a mensagem está completa e pode ser processada:

Programa do Arduíno, no final dos Serial.print:

```
Serial.println(" ."); // sinal fim de mensagem
```

No programa do Processing, trocar i<=tokens.length-6 por i<=tokens.length-7, para que a mensagem

apenas seja processada quando existirem 7 tokens, sendo a inicial "botao".

No caso de não se pretender uma descrição da mensagem, é mesmo assim conveniente existir uma token de início (neste caso "botao"), e uma token de fim (neste caso "."), e pelo meio os dados por ordem.

Como controlar actuadores do processing?

Pretende-se agora o processo inverso, isto é, enviar informação do Processing para o Arduíno. Por exemplo, pretende-se obter a posição do rato, e colocar um LED com uma intensidade dependente do valor de X.

```
import processing.serial.*;

Serial myPort;
String entrada; // string de entrada

void setup()
{

	myPort = new Serial(this, Serial.list()[0], 9600);

}

void draw()
{
}

void mouseMoved()
{

	if(pmouseX!=mouseX)

		myPort.write(255*mouseX/width);

}
```

No Arduíno a informação é recebida byte a byte. O Processing pode enviar logo uma string, mas há que ter em atenção que no Arduíno não temos muita capacidade de computação, pelo que é preferível enviar logo a informação na melhores condições para ser processada o Arduíno, neste caso o valor da intensidade do LED. O programa no Arduíno fica assim (ligar apenas um LED):

```
int led=5;

void setup()
{

	pinMode(led, OUTPUT);
	Serial.begin(9600);

}

void loop()
{

	// leitura, e reflexo no LED
	if(Serial.available()>0)
	{
```

```
            int valor=Serial.read();
            analogWrite(led,valor);

        }

}
```

A simplicidade é bastante elevada, já que apenas se envia um tipo de dados. Este programa pode ser adicionado ao anterior, e nesse caso a posição do rato controla a intensidade do LED, e os sensores controlam a cor de fundo da aplicação no processing.

Caso se pretenda enviar mais que um tipo de informação, tem que se reservar um número para início de mensagem, sendo recebidos de seguida todos os valores.

O que se faz com a informação recebida, fica naturalmente livre. Poderá não ser controlar o valor de um LED, mas sim enviar um número para um display de 7 segmentos, ou chamar uma função que exista no programa do Arduíno.

Numa situação extrema, pode-se no programa do Arduíno, enviar informação sobre todos os sensores, e receber valores para enviar para todos os atuadores. Esta estratégia permite que se desenvolva o programa apenas no Processing, sendo tudo controlado no Processing. No entanto, funcionalidades que requeiram precisão em termos sequenciais, como comunicar para um circuito integrado, ou ler/enviar uma sequência de dados para um sensor/atuador, convém ser feito no programa do Arduíno de modo a evitar eventuais atrasos devido a problemas na comunicação sem série.