Interacção com o IntelliBrain

Prof. Carlos Carreto

Este tutorial é baseado no Capítulo 4 do manual Exploring Robotics with the IntelliBrain-Bot da RidgeSoft e demostra as funcionalidades de interface existentes no IntelliBrain. O objectivo é aprender a programar o IntelliBrain de modo a poder interagir com pessoas.

Usar output de texto

Uma maneira simples para o robô IntelliBrain-Bot interagir com as pessoas, é através da exibição de mensagens de texto no seu ecrã LCD. O ecrã LCD tem espaço para exibir duas linhas de texto, cada uma com dezasseis caracteres.

Para podemos trabalhar com o LCD, é necessário declara um objecto do tipo da classe Display. Aceda ao JavaDoc da API e reveja a informação sobre a classe Display. Identifique os métodos da classe que permitem escrever no LCD.

O objecto deve ser criado através do método getLcdDisplay da Classe IntelliBrain. Reveja também a informação sobre este método no JavaDoc da API.

Use o que aprendeu para desenvolver um pequeno programa que escreva o seu nome na primeira linha do LCD.

1. Criar um novo projecto com o nome “Interact”.

2. Adicionar os imports necessários para usar as classes Display e IntelliBrain.

```
import com.ridgesoft.intellibrain.IntelliBrain;
import com.ridgesoft.io.Display;
```

3. Declarar como variável global da classe, um objecto do tipo Display para representar o LCD.

```
private static Display display;
```

4. Criar o objecto no método main.

```
display = IntelliBrain.getLcdDisplay();
```

5. Adicionar a instrução para escrever o seu nome na primeira linha do LCD. O LCD tem duas linhas; a primeira é identificada por 0 e a segunda por 1.

```
display.print(0, "Interact");
```

6. O programa completo fica como se segue:

```
import com.ridgesoft.intellibrain.IntelliBrain;
import com.ridgesoft.io.Display;

public class Interact {
  private static Display display;

  public static void main(String args[]) {
    display = IntelliBrain.getLcdDisplay();
    display.print(0, "Interact");
  }
}
```

7. Testar o programa no robô.

O programa mostra o nome no LCD e termina logo a seguir.

Usar LEDs

Uma das maneiras mais simples do robô comunicar com as pessoas é através de Light Emitting Diodes (LEDs).

A aplicação mais comum dos LEDs é indicar se os aparelhos estão ou não ligados. O IntelliBrain tem um LED verde que se mantem aceso enquanto a placa estiver a ser alimentada. Este LED não pode ser controlado, mas o IntelliBrain tem 6 LEDs à direita do LED de alimentação que podemos controlar a partir dos nossos programas. Os dois primeiros LEDs à direita do LED de alimentação são o LED de status e o LED de fault.

O LED de status, que é verde, é ligado sempre que o botão STAR é premido. O LED de fault, que é vermelho, é ligado sempre que o botão STOP é premido. Podemos controlar o estado destes LEDs quando os botões não estão a ser premidos. No entanto, se a Java Virtual Machine encontrar um problema, ligará o LED de fault.

Os 4 LEDs a seguir, são LEDs que podemos ligar e desligar como quisermos.

Para aprender a controlar estes LEDs vamos acrescentar código ao programa em desenvolvimento. Num tutorial anterior vimos como programar o robô para executar manobras simples. Vamos expandir o código que desenvolvemos, de modo a controlar os LEDs para que estes indiquem como é que os motores estão a funcionar em dado momento.

Vamos começar por programar uma manobra para o robô navegar descrevendo um quadrado.

1. Adicionar os imports necessários para controlar os motores.

```
import com.ridgesoft.robotics.Motor;
import com.ridgesoft.robotics.ContinuousRotationServo;
```

2. Declarar dois objectos para representar os motores do robô

```
private static Motor leftMotor;
private static Motor rightMotor;
```

3. Instanciar os dois objectos no método main.

```
leftMotor = new ContinuousRotationServo( IntelliBrain.getServo(1),
false, 14);
rightMotor = new ContinuousRotationServo( IntelliBrain.getServo(2),
true, 14);
```

4. Criar um método para permitir especificar a velocidade a aplicar aos dois motores, assim como o tempo que os motores devem funcionar a essa velocidade.

```
public static void go( int leftPower, int rightPower, int
milliseconds)
{
  leftMotor.setPower(leftPower);
  rightMotor.setPower(rightPower);
  try {
    Thread.sleep(milliseconds);
  }
    catch (InterruptedException e) {}
}
```

5. Criar um método stop para parar os motores.

```
public static void stop() {
  leftMotor.stop();
  rightMotor.stop();
}
```

6. Criar um método que use os métodos go e stop para manobrar o robô de modo a descrever um quadrado.

```
public static void maneuverSquare(int size)
{
  for (int i = 0; i < 4; i++) {
    go(16, 16, size);
    go(8, -8, 600);
  }
  stop();
}
```

Por conveniência as unidades de size estão em milissegundos e não em unidades de distância. Se preferir, use os métodos que desenvolveu em tutorias anteriores para o robô navegar uma dada distancia ou para rodar um dado ângulo.

7. Adicionar o método maneuverSquare no final do método main.

```
maneuverSquare(3000);
```

8. Testar o programa no robô.

Se necessário, afine os valores para o robô navegar num quadrado.

De seguida vamos expandir o programa de modo a usar LEDs para indicar o funcionamento dos motores.

Usaremos os 4 LEDs mais à esquerda (imediatamente acima do botão STOP).

Os LEDs impares são verdes e os pares são vermelhos. Os dois primeiros LEDs serão usados para indicar o estado do motor da esquerda e os dois LEDs a seguir serão usados para indicar o estado do motor da direita. O LED verde será ligado quando o motor rodar no sentido de avançar e o LED vermelho será ligado quando o motor rodar no sentido de recuar. Os LEDs serão desligados quando o motor parar.

1. Rever a informação no JavaDoc da API sobre o método getUserLed da classe IntelliBrain e sobre a classe LED.

2. Adicionar o import da classe LED

```
import com.ridgesoft.io.LED;
```

3. Declarar os objectos para representar os LEDs que pretendemos controlar.

```
private static LED leftFwdLED;
private static LED leftRevLED;
private static LED rightFwdLED;
private static LED rightRevLED;
```

4. Adicionar ao método main as instruções necessárias para criar os objectos dos LEDs.

```
leftFwdLED = IntelliBrain.getUserLed(1);
leftRevLED = IntelliBrain.getUserLed(2);
rightFwdLED = IntelliBrain.getUserLed(3);
rightRevLED = IntelliBrain.getUserLed(4);
```

Para controlar os LEDS como indicado anteriormente, usaremos instruções condicionais

```
if (condition1) {
  conditionally executed statements
  :
}
else if (condition2) {
  conditionally executed statements
  :
}
else {
  conditionally executed statements
  :
}
```

5. Acrescentar o seguinte bloco para controlar os LEDs correspondentes ao motor da esquerda.

```
if (leftPower > 0) {
 leftFwdLED.on();
 leftRevLED.off();
}
else if (leftPower < 0) {
 leftFwdLED.off();
 leftRevLED.on();
}
else {
 leftFwdLED.off();
 leftRevLED.off();
}
```

O controlo dos LEDs do motor da direita é feito de forma similar.

Adicione o código anterior ao método go, imediatamente a seguir às instruções que definem a velocidade do motor.

6. Acrescentar um código similar para controlar os LEDs da direita. Copie e cole o bloco anterior e substitua “left” por “right.”

7. Adicionar as instruções seguintes ao método stop para desligar os LEDs.

```
leftFwdLED.off();
leftRevLED.off();
rightFwdLED.off();
rightRevLED.off();
```

8. Testar o programa no robô.

Observe os LEDs à medida que o robô executa a manobra. Verifique se os LEDs ligam e desligam como pretendido.

Usar o Thumbwheel

O thumbwheel do IntelliBrain funciona como o controlo de volume de um rádio. Nesta secção vamos usá-lo para permitir ao utilizador indicar qual o tamanho do lado do quadrado navegado pelo robô.

O thumbwheel funciona como um input analógico. Quando o programa lê o valor do thumbwheel obtém um valor entre 0 e 1023. A leitura varia em função da posição da roda. Quando esta estiver totalmente rodada no sentido contrário aos ponteiros do relógio, o valor lido será de 0. Quando estiver totalmente rodada no sentido oposto, o valor lido será 1023. Quando a roda estiver numa posição intermédia, o valor lido estará entre 0 e 1023.

Complete os seguintes passos para expandir o programa de modo a usar o thumbwheel para indicar o tamanho do lado do quadrado.

1. Rever a informação no JavaDoc da API sobre o método getThumbWheel da classe IntelliBrain e sobre a classe interface AnalogInput.

2. Adicionar o import necessário para a classe interface AnalogInput.

```
import com.ridgesoft.robotics.AnalogInput;
```

3. Declarar um objecto do tipo AnalogInput para representar o thumbwheel.

```
private static AnalogInput thumbwheel;
```

4. Adicionar a instanciação do objecto ao método main.

```
thumbwheel = IntelliBrain.getThumbWheel();
```

5. Adicionar a seguinte instrução ao método main para visualizar o valor lido do thumbwheel na segunda linha do LCD. Adicione a instrução imediatamente antes da chamada do método maneuverSquare.

```
display.print(1, "Thumbwheel: " + thumbwheel.sample());
```

6. Modificar o parâmetro de entrada do método maneuverSquare de modo a usar o valor do thumbwheel para controlar o tamanho do quadrado.

```
maneuverSquare(thumbwheel.sample() * 5);
```

Recorde que o intervalo de valores obtido do thumbwheel é de 0 a 1023. Na instrução anterior o valor é multiplicado por 5 e passado para o método maneuverSquare. O tamanho é especificado em termos de milissegundos que o robô anda em frente ao longo de cada lado do quadrado.

7. Testar o programa no robô.

Execute o programa várias vezes, rodando primeiro o thumbwheel para diferentes posições. Observe o valor mostrado na segunda linha do LCD. O tamanho do lado do quadrado navegado pelo robô depende o valor do thumbwheel.

Usar Botões

A funcionalidade do thumbwheel que adicionamos na secção anterior tem uma grande deficiência. Não permite efectuar a selecção de valores de forma interactiva com o thumbwheel, quando o programa está a funcionar. Podemos resolver o problema fazendo com que o robô apenas execute a manobra quando o botão START for premido uma segunda vez. Até lá, o programa permitirá ao utilizador seleccionar o valor com o thumbwheel.

Siga os seguintes passos para incluir essa funcionalidade.

1. Rever o JavaDoc da API no que se refere à classe interface PushButton e ao método getStartButton da classe IntelliBrain.

2. Adicionar o import para a classe interface PushButton.

```
import com.ridgesoft.robotics.PushButton;
```

3. Declarar um objecto para representar o botão START.

```
private static PushButton startButton;
```

4. Adicionar a seguinte instrução para instanciar o objecto.

```
startButton = IntelliBrain.getStartButton();
```

5. Adicionar um ciclo while à volta da instrução print do thumbwheel. O ciclo deve executar enquanto o butão START não for novamente premido.

```
startButton.waitReleased();
while (!startButton.isPressed()) {
  display.print(1, "Thumbwheel: " + thumbwheel.sample());
}
```

A instrução que precede o ciclo espera que o botão seja libertado (após ter sido premido a primeira vez). Embora o utilizador possa apenas demorar alguns milissegundos a premir o botão, esse tempo é suficiente para o programa iniciar a execução e executar a condição do ciclo while. A instrução que precede o ciclo é necessária para evitar que a condição do ciclo seja testada com o botão START ainda a ser premido pela primeira vez.

6. Testar o programa no robô.

O programa agora permite que o utilizador ajuste o valor com o thumbwheel e veja o valor actual no LCD. Quando o utilizador voltar a premir o botão START o programa sai do ciclo e o

valor escolhido é usado na manobra do robô. Teste o programa escolhendo diferentes valores com o thumbwheel.

Ensinar ao Robô Novos Comportamentos

Até agora programámos um comportamento para o robô que consiste em navegar descrevendo um quadrado. A seguir vamos adicionar um novo comportamento e criar uma interface que permite ao utilizador escolher o comportamento que o robô deve executar.

O novo comportamento consiste em fazer o robô danças de forma aleatória. Altere o programa seguindo os seguintes passos.

1. Rever a informação do JavaDoc da API sobre a classe Random. Esta classe tem o método nextInt que devolve um número aleatório no intervalo (-4,294,967,295 to 4,294,967,296). O método também pode ser chamado com um argumento inteiro. Nesse caso devolve um número aleatório entre 0 e o valor do argumento.

2. Para fazermos com que o robô dance executando movimentos aleatórios, necessitamos gerar valores aleatórios entre -16 e 16 que é o intervalo de velocidades para os motores.

Uma maneira muito fácil de converter um valor inteiro para um dado intervalo, é usando o operador do resto da divisão inteira (%). Assim, para convertermos o número aleatório gerado pelo método nextInt no intervalo indicado anteriormente, para o intervalo (-16, 16), basta calcular o resto da divisão inteira do número gerado por 17.

```
int leftPower = random.nextInt() % 17;
int rightPower = random.nextInt() % 17;
```

Podemos tornar a dança ainda mais aleatória programando o robô para executar cada passo de dança durante um tempo também aleatório, por exemplo entre 100 e 499 milissegundos.

```
int time = random.nextInt(400) + 100;
```

Adicionar o seguinte método ao programa.

```
public static void dance(int seed) {
  Random random = new Random(seed);
  while (true) {
    int leftPower = random.nextInt() % 17;
    int rightPower = random.nextInt() % 17;
    int time = random.nextInt(400) + 100;
    go(leftPower, rightPower, time);
  }
}
```

Este método executa um ciclo infinito dentro do qual o robô é programado para executar passos de dança, controlando os motores com valores de velocidade aleatórios, durante intervalos de tempo também aleatórios.

3. Adicionar o import da classe Random.

```
import java.util.Random;
```

4. Substituir a chamada do método maneuverSquare pela chamada do método dance(thumbwheel.sample());

5. Testar o programa no robô.

Agora que programamos dois comportamentos para o robô, vamos alterar o programa de modo a que o utilizador possa escolher o comportamento a executar pelo robô.

Podemos fazer isso criando um menu controlado pelos botões STOP e START. O botão STOP será usado para escolher o comportamento a executar e o botão START será usado para executar esse mesmo comportamento.

1. Declarar um novo objecto para representar o botão STOP.

```
private static PushButton stopButton;
```

2. Instanciar o objecto através do método getStopButton. Use o método setTerminateOnStop para configurar o botão STOP de modo a que não termine o programa quando for premido.

```
stopButton = IntelliBrain.getStopButton();
IntelliBrain.setTerminateOnStop(false);
```

3. Declare uma variável local para controlar qual o comportamento a executar. Inicialize-a a 1. Adicione esta instrução imediatamente antes do ciclo do startButton no método main .

```
int trick = 1;
```

4. Adicione as seguintes instruções no ciclo do startButton para escolher o próximo comportamento cada vez que o botão STOP for premido

```
if (stopButton.isPressed()) {
  stopButton.waitReleased();
  trick++;
  if (trick > 2)
    trick = 1;
}
```

Como só há 2 comportamentos, se o valor da variável local trick for 2, a próxima vez que o botão STOP for premido a segunda instrução if faz com que a variável fique com o valor 1.

5. Inserir instruções no ciclo de modo a ser escrito no LCD o nome do comportamento seleccionado.

```
if (trick == 1) {
  display.print(0, "Maneuver Square");
}
else {
  display.print(0, "Dance");
}
```

6. Após o comportamento ter sido seleccionado, o botão STOP pode novamente ser reconfigurado para terminar o programa caso seja premido. Inserir a seguinte instrução a seguir ao ciclo.

```
IntelliBrain.setTerminateOnStop(true);
```

7. Substituir a chamada do comportamento da dança com as seguintes instruções de modo a que seja executado o comportamento que foi seleccionado pelo utilizador.

```
if (trick == 1) {
  maneuverSquare(thumbwheel.sample() * 5);
}
else {
  dance(thumbwheel.sample());
}
```

8. Testar o programa no robô.

Verifique se o menu está a funcionar correctamente seleccionado os dois comportamentos programados.

O controlo do comportamento a executar é realizado através de instruções if. Ao adicionarmos mais comportamentos, necessitaremos de mais instruções if. Não há qualquer problema nisso, mas se o número de comportamentos for grande, é melhor usarmos a instrução switch em vez de instruções if.

Uma instrução if do tipo

```
if (trick == 1) {
   // trick one statements
}
  else {
 // trick two statements
}
```

Pode ser substituída por uma instrução switch que é equivalente, mas mais eficiente.

```
switch (trick) {
case 1:
  // trick one statements
  break;
case 2:
  // trick two statements
  break;
```

```
}
```

Modifique o programa seguindo os seguintes passos, de modo a usar instruções switch em vez de instruções if.

1. Substituir a instrução if do ciclo startButton com uma instrução switch equivalente.

```
switch (trick) {
  case 1:
    display.print(0, "Maneuver Square");
    break;
  case 2:
    display.print(0, "Dance");
    break;
}
```

2. Substituir a instrução if a seguir ao ciclo startButton com a instrução switch equivalente.

```
switch (trick) {
  case 1:
    maneuverSquare(thumbwheel.sample() * 5);
    break;
  case 2:
    dance(thumbwheel.sample());
    break;
}
```

3. Testar o programa no robô.

O programa deverá funcionar como anteriormente.

Uma outra característica da instrução switch é o case default. Caso exista este case, o mesmo será usado se a variável do switch não tiver correspondência com os diferentes case implementados. Podemos usar o case default no nosso programa para eliminar a instrução

```
if (trick > 2)
 trick = 1;
```

Siga os seguintes passos.

1. Eliminar a instrução if no ciclo startButton.

2. Adicionar um case default antes do case 1, mas não adicionar uma instrução break.

```
switch (trick) {
default:
  trick = 1;
case 1:
  // case one statements
```

Cada vez que o valor de trick não for 1 ou 2, será seleccionado o case default case, o qual voltara a atribuir o valor 1 à variável trick.

3. Testar o programa no robô.

O programa deverá funcionar como anteriormente.

Ao usarmos instruções switch em vez de instruções if, faremos com que o programa seja mais fácil de manter e mais eficiente.

Usar o Buzzer

O controlador IntelliBrain inclui um buzzer, que permite interagir com as pessoas através de som.

Vamos acrescentar código ao programa de modo a ser produzido um feedback de áudio cada vez que um botão é premido.

1. Rever a informação do JavaDoc da API sobre a classe Speaker e o método getBuzzer da classe IntelliBrain.

2. Adicionar o import da classe Speaker.

```
import com.ridgesoft.io.Speaker;
```

3. Declarar um objecto do tipo Speaker para representar o buzzer.

```
private static Speaker buzzer;
```

4. Adicionar uma instrução no método main para criar o objecto.

```
buzzer = IntelliBrain.getBuzzer();
```

5. Adicionar chamadas ao método beep cada vez que os botões START ou STOP são premidos.

   Há dois locais onde estas chamadas devem ser adicionadas:

   a. Como primeira instrução do if do stopButton.isPressed
   b. Imediatamente depois do ciclo startButton.

```
buzzer.beep();
```

6. Testar o programa no robô.