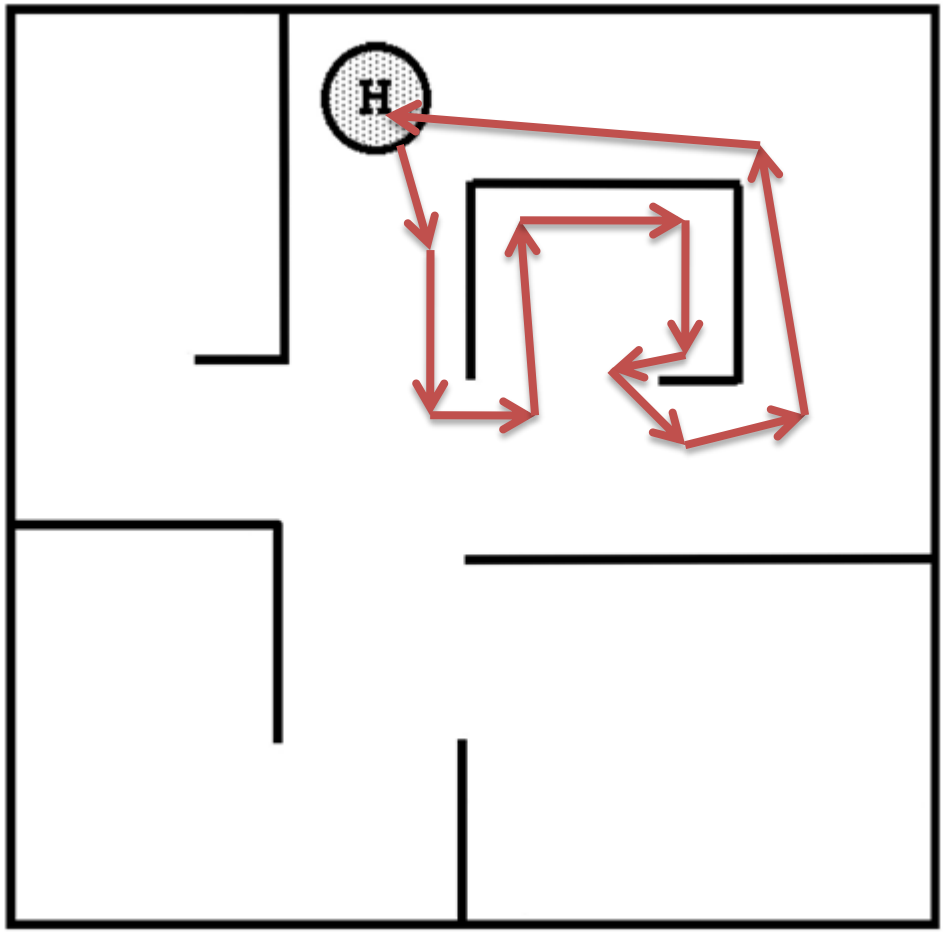
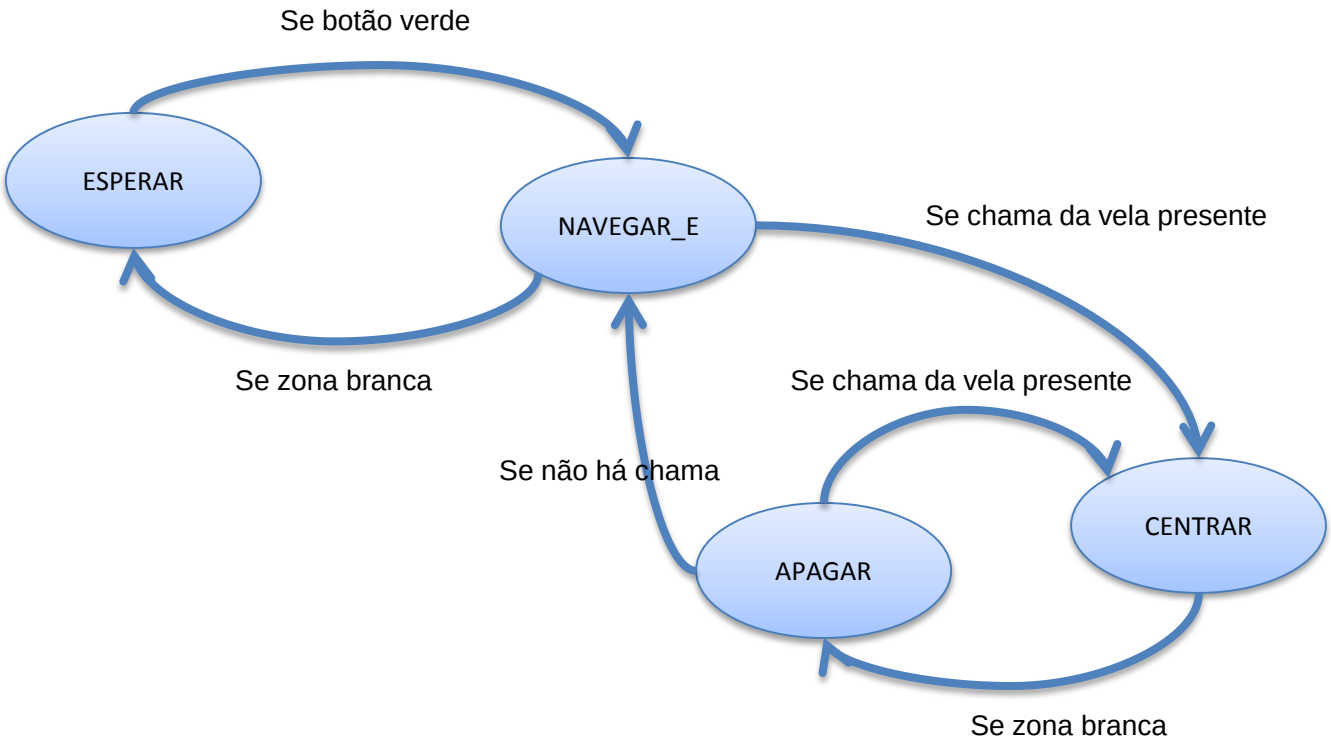
Programação de um controlador baseado em comportamentos para o robô móvel RB-1 fazer de robô bombeiro!

Missão do Robô

- Extinguir a vela no quarto ilha
  - Navegar até ao quarto ilha
  - Detectar e extinguir a vela
  - Regressar à base



Máquina de Estados do Controlador



Programação

```
//1º Criação de um novo projecto
public class RB_1 {
    public static void main(String args[]) {
    }
}
```

Programação

```
//2º Identificadores dos possíveis estados
   (comportamentos)
private static final int ESPERAR = 0;
private static final int NAVEGAR_E = 1;
private static final int CENTRAR = 2;
private static final int APAGAR = 3;
```

Programação

```
//3º Objectos para os diferentes recursos de hardware a usar
private static Display display;
private static Motor motorE;
private static Motor motorD;
private static Motor ventoinha;
private static AnalogInput fotoTE;
private static AnalogInput fotoTD;
private static AnalogInput linha;
private static RangeFinder sonarE;
private static RangeFinder sonarD;
private static IntelliBrainDigitalIO botaoVerde;
private static IntelliBrainDigitalIO botaoVermelho;
```

Programação

```
//4º Estrutura da máquina de estados
try {
	//Criação dos objectos
	. . .
	int estado = ESPERAR;

	while(true) {
		//leitura dos sensores
		. . .
		switch (estado) {
			case ESPERAR:
				. . .
				break;
			case NAVEGAR_E:
				. . .
				break;
			case CENTRAR:
				. . .
				break;
			case APAGAR:
				. . .
				break;
		}
	}
}
catch (Throwable t) {
	t.printStackTrace();
}
```

Programação

```
//5º Criação dos objectos

//***** LCD
display = IntelliBrain.getLcdDisplay();
display.print(0, "Robô Bombeiro");

//***** Motores
motorE = new ContinuousRotationServo(IntelliBrain.getServo(1), false, 14);
motorD = new ContinuousRotationServo(IntelliBrain.getServo(2), true, 14);

//***** Ventoinha
ventoinha = IntelliBrain.getMotor(2);

//***** Sensores Chama
fotoTE = IntelliBrain.getAnalogInput(1); // Sensor chama esquerda
fotoTD = IntelliBrain.getAnalogInput(2); // Sensor chama direita
linha = IntelliBrain.getAnalogInput(6); // Sensor linha

//***** Sensores Sonar
sonarE = new ParallaxPing(IntelliBrain.getDigitalIO(5));
sonarD = new ParallaxPing(IntelliBrain.getDigitalIO(6));

//***** Botões
botaoVerde = IntelliBrain.getDigitalIO(1);
botaoVerde.setPullUp(true);
botaoVermelho = IntelliBrain.getDigitalIO(2);
botaoVermelho.setPullUp(true);
```

Programação

```
//6º Leitura dos sensors

//***** Leitura dos sensores
boolean linhaB = (linha.sample() < limiteLinha);
boolean chamaE = (fotoTE.sample() > limiteChamaE);
boolean chamaD = (fotoTD.sample() > limiteChamaD);

sonarE.ping();
Thread.sleep(100);
int distE = (int) (sonarE.getDistanceCm() + 0.5f);
sonarD.ping();
Thread.sleep(100);
int distD = (int) (sonarD.getDistanceCm() + 0.5f);
```

Programação

```
//7º Comportamento ESPERAR
while(botaoVerde.isSet());
estado = NAVEGAR_E;
```

Programação

```
//8º Comportamento NAVEGAR_E
//***** Navegação
if(distD < minDistF) {  //Se parede na frente
      rodar(-90);
}
if(distE < minDistE) {  //Se muito perto da parede, virar à direita
      arco(powerBase, 5);
}
else if(distE > maxDistE) {  //Se muito longe da parede, virar à esquerda
      arco(powerBase, -5);
}
else {  //Senão, ir em frente
      avancar(powerBase + 2);
}

//***** Condições
if(chamaE || chamaD) estado = CENTRAR;
```

Programação

```
//9º Comportamento CENTRAR
//***** Centrar
if(chamaE && chamaD) {  //Se vela em frente, avança
      avancar(5);
}
else if(chamaE) {  //Se vela à direita, vira à direita
      arco(5, 3);
}
else if(chamaD) {  //Se vela à esquerda, vira à esquerda
      arco(5, -3);
}
else {  //Se perder vela, procura-a
      //não implementado
}

//***** Condições
if(linhaB) estado = APAGAR;
```

Programação

```
//10º Comportamento APAGAR
//***** Apagar
apagarVela();

//***** Condições
estado = NAVEGAR_E;
```

Programação

```
//11º Funções
public static void mostrarEstado(int estado) {
	switch (estado) {
		case ESPERAR:
			display.print(1, "ESPERAR");
			break;
		case NAVEGAR_E:
			display.print(1, "NAVEGAR_E");
			break;
		case CENTRAR:
			display.print(1, "CENTRAR");
			break;
		case APAGAR:
			display.print(1, "APAGAR");
			break;
	}
}
```

Programação

```
public static void mostrarValores(int v1, int v2) {
    display.print(0, Integer.toString(v1));
    display.print(1, Integer.toString(v2));
}


public static void arco(int power, int factor) {
    motorE.setPower(power + factor);
    motorD.setPower(power - factor);
}

public static void avancar(int power) {
    motorE.setPower(power);
    motorD.setPower(power);
}
```

Programação

```
public static void rodar(int graus) {
	if (graus < 0) {
		graus = -graus;
		motorE.setPower(powerRodar);
		motorD.setPower(-powerRodar);
	}
	else {
		motorE.setPower(-powerRodar);
		motorD.setPower(powerRodar);
	}

	try {
		Thread.sleep(graus * factorRodar);
	}
	catch (Throwable t) {
		t.printStackTrace();
	}
	parar();
}
```

Programação

```
public static void parar() {
    motorE.setPower(0);
    motorD.setPower(0);
}

public static void apagarVela() {
    try {
            ventoinha.setPower(16);
            rodar(45);
            rodar(-45);
            ventoinha.setPower(0);
    }
    catch (Throwable t) {
            t.printStackTrace();
    }
}
```

Programação

```
//12º Constantes
private static final int limiteLinha = 100;

//Tipo A: 500
//Tipo B: 700
private static final int limiteChamaE = 700;
private static final int limiteChamaD = 700;

private static final int powerBase = 8;
private static final int powerRodar = 5;

private static final int minDistE = 10;
private static final int maxDistE = 15;
private static final int minDistF = 15;
private static final int factorRodar = 10;
```

Programação

```
//13º Imports
import com.ridgesoft.intellibrain.IntelliBrain;
import com.ridgesoft.io.Display;
import com.ridgesoft.robotics.PushButton;
import com.ridgesoft.robotics.Motor;
import com.ridgesoft.robotics.ContinuousRotationServo;
import com.ridgesoft.robotics.AnalogInput;
import com.ridgesoft.robotics.RangeFinder;
import com.ridgesoft.robotics.sensors.ParallaxPing;
import com.ridgesoft.intellibrain.IntelliBrainDigitalIO;
```