CI-2657 Robótica

Tarea 1

Miembros del Equipo

Nombre	Carné	Fecha
Andrey Pérez Pérez	B25076	22-08-2016
Kevin Delgado Sandí	B22214	22-08-2016
Ricardo Aguilar Vargas	B10141	22-08-2016
Gabriel Bermúdez Mora	B10954	22-08-2016

Resumen

Sensor framework

LeJOS 0.5.0-alpha introduce un “framework” uniforme para los sensores. LeJOS considera un sensor una pieza de hardware que mide uno de muchos aspectos del medio en el que está el sensor.

El framework de LeJOS 0.5.0-alpha se compone de los siguientes elementos:

- Sensores: los sensores son una pieza de hardware que miden algún aspecto del medio ambiente en el que trabajan. La mayoría de los sensores pueden medir más de un tipo de información, por ejemplo un sensor de color no solo puede sensar el color de una superficie, sino también la intensidad de luz del medio.
  A las distintas formas en que un sensor puede operar, se le llama modos.

- Nombres: a la hora de programar una clase para un sensor específico se debe seguir la siguiente convención:
  - El primer componente del nombre tiene que ver con la empresa que fabrica el sensor, por ejemplo para los sensores hechos por Lego, el componente sería EV3, si la empresa es NXT, el componente será NXT.
  - El segundo componente del nombre de la clase será el nombre completo del sensor.
  - La tercera parte (opcional) del nombre de la clase será una “V” seguida de la versión del sensor.

  De esta forma el nombre de la clase para un sensor ultrasónico será **EV3UltrasonicSensor.**

- Modos: algunos sensores tienen distintas formas de operar. Estos son llamados modos. Cada modo provee de distinta información al programa que lo implemente. Además en un mismo programa se puede mezclar el uso de los distintos modos que un sensor tenga.
- Muestras: a cada medida que un sensor puede realizar a través de los modos con los que trabaje, se le llama muestra. Una muestra básicamente es uno o más valores tomados en un momento dado. Por ejemplo un sensor de color devuelve un valor si está trabajando en modo luz ambiental y devuelve 3 valores en modo RGB.

- Unidades estándares: LeJOS utiliza unidades estándares para que no haya problemas de compatibilidad con sensores de distintos fabricantes. LeJOS utiliza el sistema internacional de unidades. En este sentido, LeJOS retorna las distancias en metros, la aceleración en metros por segundo y los ángulos en grados.

- Sistema de coordenadas: LeJOS utiliza el sistema de coordenadas cartesianas. El eje **x** positivo apunta en la misma dirección en la que se conecta el sensor. El eje **y** apunta a la izquierda del eje x. Y el eje **z** es ortogonal a la intersección del eje 'x' y 'y'.

- Filtros: un filtro se utiliza para alterar las muestras o el flujo de las mismas. LeJOS viene con algunos filtros preimplementados, estos se encuentran en el paquete **lejos.robotics.filter**. Un ejemplo de como utilizar un filtro, es aplicar un filtro sobre las últimas cinco muestras de un sensor ultrasónico para obtener el promedio de desplazamiento.

- Compatibilidad hacia atrás: en general LeJOS 0.5.0-alpha no provee compatibilidad hacia atrás, sin embargo hay algunas excepciones. Existen algunos adaptadores que se pueden encontrar en el paquete **lejos.robotics**, estos permiten que el programador no tenga que modificar todo el código para trabajar con un sensor antiguo.

Motors

Hay muchos tipos de motores que se pueden utilizar con EV3, por ejemplo:

- EV3 Large motor
- EV3 Medium motor
- NXT motor
- PF Motors
- Tetrix motors
- RCX Motor
- Servos

Los primeros 3 de la lista anterior se pueden conectar directamente al EV3, sin necesidad de cables de conversión o adaptadores.

Estos motores se pueden dividir en dos grandes grupos:

Motores Regulados: los motores regulados tienen la velocidad controlada y pueden ser programados para que giren en un ángulo específico. Ejemplos de estos motores son los EV3 y los NXT.

Motores no regulados: este tipo de motores trabaja en una forma más simple, en el sentido en que solo se conectan y se les indica que vayan hacia adelante o hacia atrás y luego de un tiempo, paren. Ejemplos de estos motores son los PF y los RCX.
Cabe mencionar que los EV3 y los NXT pueden programarse para que trabajen como un motor no regulado, dependiendo de la aplicación.

LeJOS tiene una serie de interfaces y clases para trabajar con motores regulados y no regulados, por ejemplo para trabajar con un EV3 large motors, se puede utilizar lo siguiente:

RegulatedMotor m = new EV3LargeRegulatedMotor(MotorPort.A);

Y luego usar el motor como esto: *m.rotate(360);*

Los cuatro puertos para motores del EV3 se pueden referenciar así: *MotorPort.A, MotorPort.B, MotorPort.C and MotorPort.D*

Para motores regulados se deberían utilizar las clases correspondientes al tipo de motor, así para un EV3 large motors, se debe usar *EV3LargeRegulatedMotor,* para NXT motors sería NXTRegulatedMotor y así dependiendo del tipo de motor.

Algunos de los métodos que se pueden utilizar para motores regulados son:

- forward() : ir hacia adelante hasta parar
- backwards : ir hacia atrás hasta parar
- stop: detiene el motor
- flt: quita la energía del motor, pero no lo para
- setSpeed(int speed) : configura la velocidad del motor
- rotate(int angle) : rota en un ángulo positivo en grados
- rotateTo(int angle) : rota hacia un ángulo dado en grados

Para motores no regulados se pueden utilizar los siguientes métodos:

- setPower(int amount)
- Forward()
- msDelay (int amount)

- stop()

Los paquetes que LeJOS trae para trabajar con motores son:

- lejos.robotics
- lejos.hardware.motor: las clases principales para motores
- lejos.hardware.device: en este paquete se encuentran las clases para trabajar con controladores de terceros
- lejos.hardware.device.tetrix: para trabajar con motores Tetrix

Ejemplos de código fuentes

Ejemplo de sensor

```
// get a port instance
Port port = LocalEV3.get().getPort("S2");

// Get an instance of the Ultrasonic EV3 sensor
SensorModes sensor = new EV3UltrasonicSensor(port);

// get an instance of this sensor in measurement mode
SampleProvider distance= sensor.getMode("Distance");

// initialize an array of floats for fetching samples.
// Ask the SampleProvider how long the array should be
float[] sample = new float[distance.sampleSize()];

// fetch a sample
while(true)
  distance.fetchSample(sample, 0);
```

Ejemplo de filtro

```
// ...
// stack a filter on the sensor that gives the running average of the last
5 samples
SampleProvider average = new MeanFilter(distance, 5);
// initialise an array of floats for fetching samples
```

```
float[] sample = new float[average.sampleSize()];
// fetch a sample
average.fetchSample(sample, 0);
```