Sensor Framework y Motors en la Plataforma leJOS

Andrés Bejarano B20902
Julio César Guzmán A82939
Carlos Mata B13980
José Pablo Ureña B16692

1 de abril de 2016

Índice

1. Introducción

El siguiente documento busca aportar conocimientos sobre los sensores y motores relacionados al hardware, basado en un sistema de Lego Mindstorm.

1.1. Objetivos

- Acercar a los estudiantes que desarrollan la investigación así como compañeros de clase, al conocimiento de elementos de hardware propios de los legos MindStorm.
- Conocer los diferentes tipos de motores y su clasificación para el buen manejo de los mismos.
- Reconocer las diferentes especificaciones y características para los sensores que pueden adaptarse a los legos MindStorm.

1.2. Enunciado

A continuación, se listan las características principales y propiedades mismas del *sensor framework* así como los motores disponibles para el usuario.

2. Motores

Los motores permiten dar movimiento o bien funcionalidades externas (agarre, palanca) al robot en su desempeño. Existen una diversa cantidad de motores a disposición de los usuarios para ensamblar los robots MindStorm, a continuación se listan los mismos:

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

Existen una serie de diferencias entre dichos motores que van más allá de su construcción y estética. Por ejemplo, los motores EV3 large y medium, así como el NXT son los únicos que logran conectarse al EV3 directamente, mientras que el resto requiere adaptadores o cables.

2.1. Tipos de Motores

Existen dos tipos principales de motores, y esto lo determina específicamente el hecho que si el motor posee o no un *encode*. Un encoder es un codificador montado sobre un motor eléctrico que permite obtener información a través de señales, sobre la velocidad y posición del motor.

Si el motor posee encoder, se considera un motor *regulado*, sino posee encoder se considera un motor *no regulado*. Un motor regulado es aquel al cual puede regularse su velocidad y ángulo de rotación. Mientras que un motor no regulado sólo "sabe" moverse hacia atrás o adelante por cierto tiempo hasta detenerse.

2.2. Código y uso de motores

La plataforma leJOS permite al programador acceder y configurar los motores del EV3, a través de una serie de interfaces sin importar el tipo de motor, regulado o no regulado. A. Se puede observar a continuación, una forma de acceder un motor regulado:

```
RegulatedMotor m = new EV3LargeRegulatedMotor(MotorPort.A);
```

Este código muestra una forma de acceder un motor regulado tipo EV3 Large. Una de las ventajas en leJOS es la forma de acceder a métodos propios del motor, por ejemplo:

```
m.rotate(360);
```

Rotación del motor.
B. Por otra parte, se puede acceder a un motor no regulado de la forma:

```
EncoderMotor em = new UnregulatedMotor(MotorPort.C);
```

De esta forma, puede utilizarse un motor como EV3 o NXT de forma no regulada, especificando otras funcionales como por ejemplo:

```
em.setPower(100);
em.forward();
Delay.msDelay(1000);
em.stop();
```

Esto permite el manejo directo del motor de manera no regulada, ya que simplemente se indica su velocidad, delay y momento de detenerse.

3. Sensor Framework

Los **sensores** en leJOS son piezas de *hardware* que se utilizan para medir características específicas del entorno en el que se encuentra como por ejemplo colores y sonidos. Los sensores poseen distintas formas en las que pueden operar conocidas como **modos**. Un sensor necesita conectarse a un puerto y elegir un modo para poder realizar mediciones debido a que por sí solo no lo puede hacer. Las mediciones realizadas se llaman **muestras** y están compuestas por uno o más valores medidos en un momento determinado.

Los sensores de leJOS utilizan las unidades del Sistema Internacional de Unidades para realizar sus mediciones, mientras que para brindar ubicaciones se utiliza el sistema de coordenadas cartesiano.

3.1. Características de la implementación

leJOS define clases para trabajar con sensores, conocidas como *sensor classes* las cuales implementan una interfaz llamada *SensorModes* que le permite conocer los modos disponibles para el sensor y utilizar un modo en específico. Estas clases se encuentran en el paquete llamado *lejos.hardware.sensor* . Por su parte LeJOS implementa los modos como una clases que implementa la interfaz llamada *SampleProvider*. Dentro de la interfaz destacan principalmente los siguientes dos métodos:

- void fetchSample(float[] sample, int offset): es el encargado de tomar las mediciones.
- int getSampleSize(): devuelve la cantidad de elementos de una muestra.

Los nombres de las *sensor classes* se componen de tres partes: el nombre de la empresa que creó el sensor, el nombre del sensor y la versión del sensor (no es indispensable).

Dentro de la clase sensor existe un método llamado *getAvailableModes()* con los modos soportados por el sensor.

3.1.1. Filtros

Los filtros en leJOS permiten alterar una muestra o el flujo de la misma. Se pueden aplicar varios filtros durante el proceso de muestreo, para esto se aplican los filtros uno después del otro sobre la muestra tomada por el sensor. Los filtros se encuentran en el paquete llamado *lejos.robotics.filter*.

3.1.2. Ejemplos

A continuación se presentan dos ejemplos relacionados con el uso de sensores y filtros.

Ejemplo de utilización de un sensor El proceso para utilizar un sensor consiste de cinco pasos:

1. Asignar un puerto.
2. Inicializar el sensor.
3. Asignar un modo o *SampleProvider*
4. Crear un arreglo para guardar las muestras.
5. Tomar las muestras.

Los primeros cuatro pasos se realizan una vez mientras que el último puede repetirse varias veces, tal y como lo muestra la Figura 1.

Ejemplo de utilización de un filtro El proceso para utilizar filtros se basa en .ªpilarüno o más filtros justo después de asignar un modo para el sensor. De esta forma los filtros se comportan como un conjunto de *SampleProvider* que se aplica antes de la etapa de medición de las muestras, tal y como se muestra en la Figura 2.

3.1.3. Notas sobre compatibilidad

La compatibilidad con sensores de versiones anteriores no es usual, sin embargo existen adaptadores en el paquete *lejos.robotics* que permiten la compatibilidad con algunos sensores de versiones anteriores. Un adaptador funciona como un filtro más a la hora de realizar una medición. Asimismo se pueden definir propios adaptadores en caso que no se encuentre el adaptador adecuado.

```
// 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);
```

Figura 1: Toma de muestra de distancia con el 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.getModeName("Distance");

// 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);
```

Figura 2: Utilización de un filtro en el proceso de medición.

Referencias

[1] Griffiths, L. (2016). leJOS / EV3 Wiki / Sensor Framework. [online] Sourceforge.net. Available at: https://sourceforge.net/p/lejos/wiki/Sensor%20Framework/ [Accessed 1 Apr. 2016].