Robots Móviles

Practica No. 4

Comunicación entre Procesos Usando un Blackboard para Operar un Robot Móvil

Objetivo: Usando un Blackboard interconectar procesos en C/C++ y Python que reciben y envían comandos a los sensores y actuadores del robot.

Desarrollo: Para cada uno de los siguientes apartados, realizar el software que se pide.

Duración: Una semana

1. El BlackBoard es una herramienta de paso de mensajes y repositorio de variables compartidas desarrollada en el laboratorio de Bio-Robótica por el MI. Mauricio Matamoros. Sirve para comunicar diferentes programas bajo los esquemas comando- respuesta y publicación suscripción. La manera en que es implementado es a través de un esquema cliente-servidor, donde el BlackBoard toma el rol de cliente y cada módulo o que se quiere conectar, se comporta como servidor. El BlackBoard utiliza un archivo de configuración XML para conocer la ubicación de cada módulo (direccion IP y puerto) para establecer una conexión y poder comunicarse con ellos. Cada módulo puede entonces enviar mensajes que corresponden a comandos, que deberán ser ejecutados por otro módulo, o bien, por el mismo BlackBoard, en caso de ser comandos estándar que el entienda. Se puede escribir a una variable compartida y todos los modulos que esten suscritos a ella recibirán una notificación de que el valor ha cambiado, para que puedan actuar en consecuencia si fuera necesario.
Para facilitar la implementación de nuevos modulos, se han desarrollado APIs de desarrollo que abstraen todos los detalles de conexión y manejo de mensajes de la aplicacion para la que fue disenada el modulo. Existen APIs para los lenguajes C#, C++, Python y Clips

Descargue en la página de las prácticas el archivo denominado Blackboard_Robot_Movil, el cual contiene los binarios del Blackboard, así como código en C++ y Python para operar un robot móvil.

2. El código en C++ requiere de la herramienta de compilación BOOST, instalela utilizando el siguiente comando:

```
$ sudo apt-get install libboost-all-dev
```

Una vez instalado BOOST colocarse en el directorio: ~/robotics/BB_compilation_C++

Edite CMakeLists.txt si es necesario cambiando los directorios fuentes y reconstruya el proyecto con cmake, si cmake no esta instalado, instalelo con:

```
$ sudo apt-get install cmake
```

Solamente la primera vez que se utilize cmake, primero hay que borrar el archivo CmakeCache.txt y el directorio Cmakefiles:

```
$ rm CmakeCache.txt
$rm -r CMakeFiles
```

Después ejecute cmake:

```
$ cmake CMakeLists.txt
```

Compile con make el código C/C++ que controlará a los actuadores y sensores del robot:

```
$ make
```

Mono es una utilería de Linux que permite ejecutar programas desarrollados en el sistema operativo de Windows de Microsoft. Si no esta instalado en su sistema, instalelo para ejecutar el Blackboard.exe de la siguiente forma:

```
$ sudo apt-get install mono-complete
```

Una vez instalado ejecute el Blackboard.exe :

```
$ mono ~/robotics/Blackboard/BlackBoard_bin/Blackboard.exe BB_robot.xml
```

El programa Blackboard.exe utiliza el archivo de configuración xml BB_robot.xml el cual contiene la configuración de los programas que se interconectarán con el Blackboard, indicando las direcciones IPs, puertos y variables compartidas.
Para esta practica se utilizaran las siguientes variables compartidas: **sensor, actuator y gui**.
La variable **sensor**, como su nombre lo indica contiene los datos de los sensores: laser, fotoresistencias, etc. La variable **actuator** contiene los datos que se le envía a los actuadores y **gui** contiene los datos introducidos en la interface gráfica.
El apendice A muestra este archivo. En el GUI del Blackboard inicialicelo con el botón de Start, éste esperará que se conecten los demás procesos a través de el.

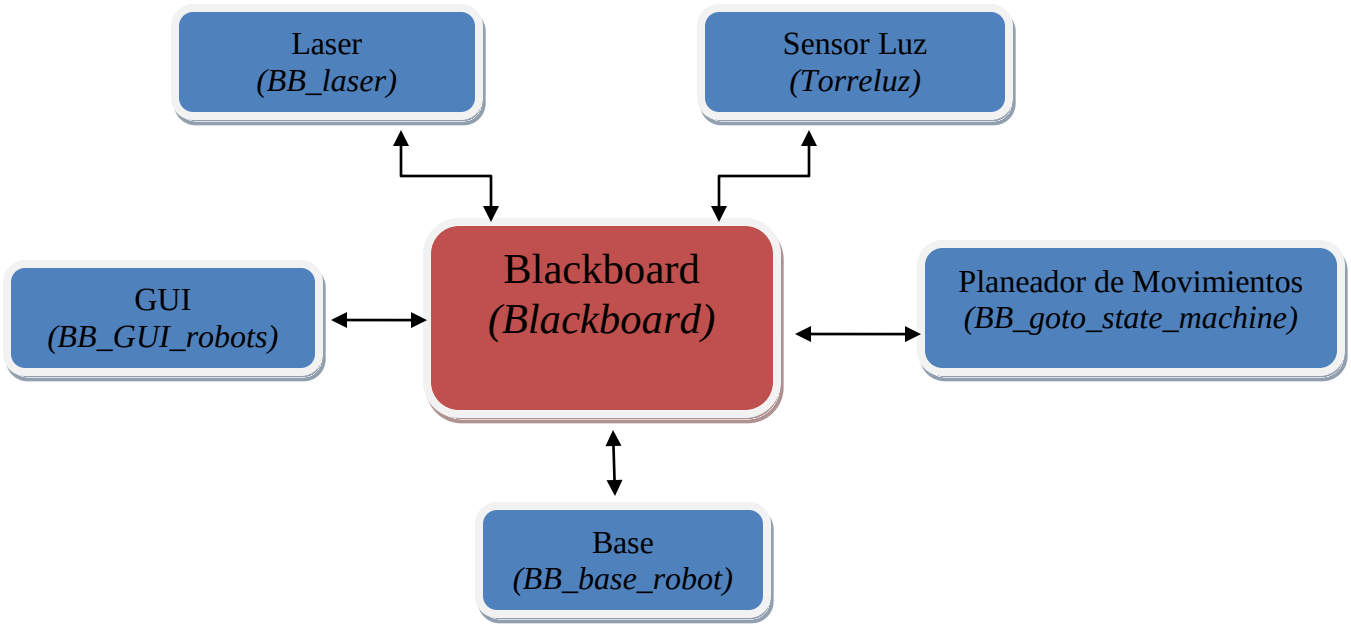
Blackboard utiliza librerías que se encuentran compiladas en el directorio
~ /robotics/Blackboard/librerias/uRobotics

Si los archivos compilados no funciona en su sistema Linux, por incompatibilidad del procesador, recompilelos de nuevo en la carpeta ~ /robotics/Blackboard/librerias/uRobotics de la siguiente forma:

```
$ rm CmakeCache.txt
```

$ rm CmakeFiles
$ cmake CmakeList.txt
$make

3. Para utilizar los modulos del código de Python copie el directorio ~/robotics/Blackboard/librerias/pyRobotics al directorio /usr/lib/python2.7/pyrobotics o la directorio que contenga la versión de Python que esta utilizando.



4. Para observar que todos los módulos están funcionando bien, en la primera parte de la practica se usará los simuladores y en la segunda un robot real, el robot Turte-Bot:

4.1.- En una X-terminal ejecute el código del Blackboard con su archivo de configuración XML:

Situarse primero en su directorio:
$ cd ~/robotics/Blackboard/BlackBoard_bin/
$ mono Blackboard.exe BB_robot.xml

4.2.- Active cada uno de los módulos con los que se conectará el Blackboard:

4.2.1. En una X-terminal ejecute el código, escrito en Python, que simula la base del robot:
Situarse primero en su directorio:
$ cd ~/robotics/bb_actuators
$ python BB_simulator_base_robot.py

4.2.2. En una X-terminal ejecute el código, escrito en C/C++, que simula las lecturas de un laser:

Situarse primero en su directorio:

```
$ cd  ~/robotics/BB_compilation_C++/bin
$ ./BB_simulator_laser
```

4.2.3. En una X-terminal ejecute el código, escrito en C/C++, que simula las lecturas de un sensor de luz:

Situarse primero en su directorio:

```
$ cd  ~/robotics/BB_compilation_C++/bin
$ ./Sim_TorreLuz
```

4.2.4. En una X-terminal ejecute el código del planeador de movimientos, escrito en C/C++, que contiene la máquina de estados que controla los movimientos del robot:

Situarse primero en su directorio:

```
$ cd  ~/robotics/BB_compilation_C++/bin
$ ./BB_GoTo_State_Machine
```

4.2.5. Por último en una X-terminal ejecute el código, escrito en Python, del GUI que opera al robot:

Situarse primero en su directorio:

```
$ cd  ~/robotics/gui
$ python BB_robots.py
```

4.3.- En el GUI del Blackboard asegúrese que todos los módulos estén conectados con el mostrando en sus iconos de la lista de módulos el color verde.

4.4.- En la ventana GUI_ROBOTS seleccione el comportamiento número 3. En la ventana gráfica del medio ambiente, PLANNER, seleccione un origen con el botón izquierdo del mouse y un destino con el motor derecho de éste. Seleccionar el botón "Execute Robot Command" para que el robot alcance el destino. Si todos los módulos están bien interconectados se deberá ver el movimiento del robot simulado.

5. Repita el punto 4 usando sensores y el robot reales, el asistente del curso le mostrará como utilizar el robot Turtle-Bot.

5.1 En la computadora de éste correrán los siguientes procesos:

* Para la base del robot BB_base_robot.py
* Para las lecturas del laser BB_laser.py
* Para las lecturas del sensor de luz TorreLuz.

5.2 La computadora del estudiante se deberá conectar al mismo punto de acceso del robot, en ésta correrán los siguientes procesos: el Blackboard, la interface GUI y el planeador de movimientos.

APENDICE A

Configuración de los módulos que el Blackboard interconectará, formato en XML

```
<?xml version="1.0" encoding="UTF-8"?>

<blackboard version="1.0">

    <configuration>
        <name>BLACKBOARD</name>
        <port>2300</port>
        <sendAttemps>2</sendAttemps>
        <commands />
        <startupSequence />
        <shutdownSequence />
    </configuration>

    <sharedVariables>
        <var name="sensor" type="string"/>
        <var name="actuator" type="string"/>
        <var name="gui" type="string"/>
    </sharedVariables>
    <modules>
        <module name="LASER_SENSOR">
            <ip>127.0.0.1</ip>
            <ip>192.168.190.110</ip>
            <port>2020</port>
            <aliveCheck>true</aliveCheck>
            <commands>
                <command name="cmd_one" answer="True" parameters="True" timeout="2000" />
                <command name="cmd_two" answer="True" parameters="True" timeout="6000" />
            </commands>
        </module>
        <module name="MOTION_PLANNER">
            <ip>127.0.0.1</ip>
            <ip>192.168.190.110</ip>
            <port>2030</port>
            <aliveCheck>true</aliveCheck>
            <commands>
                <command name="cmd_one" answer="True" parameters="True" timeout="2000" />
               <command name="cmd_two" answer="True" parameters="True" timeout="6000" />
            </commands>
        </module>
```

```
        <module name="BASE_ROBOT">
            <ip>127.0.0.1</ip>
            <ip>192.168.190.110</ip>
            <port>2040</port>
            <aliveCheck>true</aliveCheck>
            <commands>
                <command name="cmd_one" answer="True" parameters="True" timeout="2000" />
                <command name="cmd_two" answer="True" parameters="True" timeout="6000" />
            </commands>
        </module>
        <module name="GUI">
            <ip>127.0.0.1</ip>
            <ip>192.168.190.110</ip>
            <port>2050</port>
            <aliveCheck>true</aliveCheck>
            <commands>
                <command name="cmd_one" answer="True" parameters="True" timeout="2000" />
                <command name="cmd_two" answer="True" parameters="True" timeout="6000" />
            </commands>
        </module>

       <module name="LIGHT_SENSOR">
            <ip>127.0.0.1</ip>
            <ip>192.168.190.110</ip>
            <port>2011</port>
            <aliveCheck>true</aliveCheck>
            <commands>
                <command name="cmd_one" answer="True" parameters="True" timeout="2000" />
                <command name="cmd_two" answer="True" parameters="True" timeout="6000" />
            </commands>
        </module>

    </modules>

</blackboard>
```