- **Setup**

Step 1

Unpack "matlab_tools.tar.gz" in /path_to_tinyos_2.x/tools/

- Result:
  - |-- tools
  - | |-- Bootstrap
  - | |-- Makefile.am
  - | |-- README
  - | |-- build.xml
  - | |-- configure.ac
  - | |-- **matlab**
  - | |-- platforms
  - | |-- release
  - | `-- tinyos

Step 2

Unpack "matlab_java_code.tar.gz" in /path_to_tinyos_2.x/support/sdk/java/net/tinyos/

- Result:
  - |-- net
  - | |-- Makefile
  - | `-- tinyos
  - | |-- Makefile
  - | |-- comm
  - | |-- **matlab**
  - | |-- message
  - | |-- mviz
  - | |-- packet
  - | |-- sf
  - | |-- sim
  - | |-- tools
  - | `-- util
  - `-- tinyos.jar

Step 3

Add to the CLASSPATH the path of the matlab libraries. In my case /opt/matlab/java/jar/jmi.jar. To do that, we have different options:

Option 1: (not permanent) In a terminal:

```
export CLASSPATH=$CLASSPATH:/opt/matlab/java/jar/jmi.jar
```

Option 2: (permanent) Modify the ~/.bashrc (/home/xxxx/.bashrc) and add this line at the end of the file:

```
export CLASSPATH=$CLASSPATH:/opt/matlab/java/jar/jmi.jar
```

Option 3: (permanent) Modify the tinyos.sh

Step 4

Compile the matlab classes. Go to /path_to_tinyos_2.x/support/sdk/java/net/tinyos/matlab and "`make matlab`". If there is any problem in this point you have to check if the CLASSPATH has the jmi.jar.

Step 5

Recompile the tinyos.jar to include the matlab classes. Go to /path_to_tinyos_2.x/support/sdk/java/ and "`make tinyos.jar`". You could check that the matlab classes are in the tinyos.jar. (tinyos.jar/net/tinyos/matlab/).

Step 6

Now you have all the classes in tinyos. The last step is setting the java classpath in Matlab. Again, we have you options:

Option 1: (not permanent) In Matlab command window:

```
javaclasspath('/path_to_tinyos_2.x/support/sdk/java/tinyos.jar');

javaclasspath('/path_to_tinyos_2.x/tools/matlab/class/');
```

Option 2: (permanent) Modify the classpath.txt of Matlab. In my case is in /opt/matlab/toolbox/local/classpath.txt. Add at the end of the file the folowing line, with your correct path:

```
/path_to_tinyos_2.x/support/sdk/java/tinyos.jar

/path_to_tinyos_2.x/tools/matlab/class/
```

Step 7

Set the new classpath to Matlab.

```
/path_to_tinyos_2.x/tools/matlab/
```

And the next that that Matlab start, the **startup.m** script will be launched. It is necessary to set some common variables and structures.

Step 8

Execute tos-install-jni to setup this library. In a terminal:

```
sudo tos-install-jni
```

Note: I have to do this step every time that I restart Matlab. If not, I see a message like that:

```
getenv JNI library not found. Env.getenv will not work (run the tos-install-jni tool, see man tos-install-jni for more details)
```

- **Folders in tools/matlab**
  - **apps**: where the testApp is
  - **class**: classes of the packets that we want to send
  - **comm**: files with function to connect, receive, ...
    - **comm/callbacks**: files with some callbacks, for example timerFired or when a packet is received.

- ## Usage

Step 1: Load in a tmote the BaseStation program (/dev/ttyUSB0) and BlinkToRadio attached in another tmote.

Step 2: Init the SerialForwarder in a terminal:

```
java  net.tinyos.sf.SerialForwarder -comm serial@/dev/ttyUSB0:tmote
```

Step 3: Matlab-> Init the variables

```
testApp init
```

Step 4: Matlab-> Register the listener and start the program

```
testApp startP
```

Now every received packet Matlab will send one with the counter increased. So the result of this testApp is that only Matlab increase the counter, the BlinkToRadio set its value with the received packet.

Step 5: Matlab-> Stop or disconnect each connection

```
testApp stopP

or

testApp disconnectP
```

- **Usage**