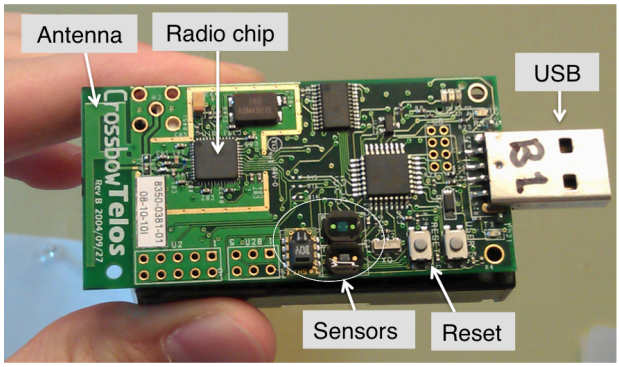
Lab Environment

Sensor nodes

The sensor node or mote that you will use in the labs is the Crossbow Telos b, also known as TMote Sky. Since the motes are fragile, you are required to hold them from the lower part as it is depicted below (the black plastic where the batteries are supposed to be placed). Never hold them from the printed circuit board. Remember that you will be the responsible for the motes during the labs.



The important mote components used in the labs are shown in the picture: reset button, sensors, radio chip, antenna and USB connector.

Programming environment

To code your applications you can use the editor that you like.

TinyOS is also available for other Linux distributions, Mac OS X or Cygwin in the Windows environment. However, we will not support you with installation and necessary patches.

TinyOS directory

You will perform the labs with the TinyOS version 2. You can have a look to the /courses/TDDI07/tinyos-main/tos directory, where all the components, interfaces and some applications can be found.

apps	TinyOS example applications
doc	Documentation
tools	TinyOS simulation tools
tools/java	Base directory for all TinyOS java packages
tos	Base directory of TinyOS NesC source code.
tos/interfaces	**Source code for all interface declarations**
tos/lib	Libraries
tos/platform	Platform specific sources
tos/platform/pc	The platform for TOSSIM simulation
tos/system	Source for system interface implementations
tos/types	A few system wide definitions and types

Setting up git

You first need to clone the git repository using the following command

```
git clone https://gitlab.liu.se/TDDI07/tinyos-lab
```

Then you should create a new project on gitlab.liu.se using one of your LiU-IDs, and clone the code there, and set the origin url accordingly (replacing LIUID with your id):

```
git push --set-upstream git@gitlab.liu.se:LIUID/tinyos-lab.git master

git remote set-url origin git@gitlab.liu.se:LIUID/tinyos-lab.git
```

Add your lab assistant as a reporter in your project, this will be the way that you hand in your code.

Steps to compile and install applications

Step 1 (needs to be done every time you login): Set environment variables using the following command in your lab folder:

```
. TDDI07-settings.sh
```

(note the initial dot with a space before the file name)

Step 2: Attach the motes and export the MOTECOM variable

To show the list of devices attached and in which port they are available you can use the following command:

```
motelist
```

Step 3: Compile and install the application

To compile the code for the telosb platform you have to run the following command in your application folder. The compilation errors will help you to find out errors in your program as well:

```
make telosb
```

Install the application in the default mote without compiling the code again:

```
make telosb reinstall
```

The following command installs the application in the mote attached to the USB0 port. Moreover, it assigns the TOS_NODE_ID=1. This is really useful when you are using more than one mote connected to the USB ports:

```
make telosb reinstall.1 bsl,/dev/ttyUSB0
```

Step 4: Serial communication commands

If you are using serial communication in the lab, the following are the commands to listen to the serial communication via USB from the computer. The first one prints raw data received from the mote in hexadecimal and the second one is used to print the printf commands of the mote (replace X with the correct port number):

```
java net.tinyos.tools.Listen -comm serial@/dev/ttyUSBX:telosb
java net.tinyos.tools.PrintfClient -comm serial@/dev/ttyUSBX:telosb
```

Compiling and installing your first application

In order to try these commands, you can copy the Blink application from /courses/TDDI07/tinyos-main/apps to your folder, compile and install it in one of your motes.

To compile the code:

```
tddi07@wcu-desktop:~/Blink$ make telosb
mkdir -p build/telosb
    compiling BlinkAppC to a telosb binary
ncc -o build/telosb/main.exe  -Os -O -mdisable-hwmul -fnesc-separator=__ -
Wall -Wshadow -Wnesc-all -target=telosb -fnesc-cfile=build/telosb/app.c -
board= -DDEFINED_TOS_AM_GROUP=0x22 -
DIDENT_APPNAME=\"BlinkAppC\" -DIDENT_USERNAME=\"tddi07\" -
DIDENT_HOSTNAME=\"wcu-desktop\" -DIDENT_USERHASH=0x9b0a5586L
-DIDENT_TIMESTAMP=0x4e8c5a6cL -DIDENT_UIDHASH=0x8572cef3L
BlinkAppC.nc -lm
    compiled BlinkAppC to build/telosb/main.exe
          2648 bytes in ROM
            54 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe
build/telosb/main.ihex
    writing TOS image
```

Check that a mote is attached and its path:

```
tddi07@wcu-desktop:~/Blink$ motelist
Reference  Device          Description
---------- ---------------- ----------------------------------------------
XBTFXN4V   /dev/ttyUSB0     XBOW Crossbow Telos Rev.
```

Install the application in the mote, either using the simple command or specifying where the mote is and the options:

```
tddi07@wcu-desktop:~/Blink$ make telosb reinstall
or
tddi07@wcu-desktop:~/Blink$ make telosb reinstall.0 bsl./dev/ttyUSB0
cp build/telosb/main.ihex build/telosb/main.ihex.out
    found mote on /dev/ttyUSB0 (using bsl,auto)
    installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out
MSP430 Bootstrap Loader Version: 1.39-telos-8
```

```
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
2680 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out build/telosb/main.ihex.out
```

Now the application is working in the mote. The expected outcome is to see the leds of the mote blinking. You can inspect and understand the code to see how.