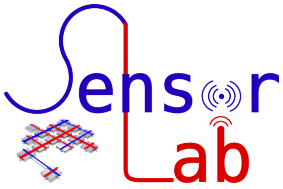
Faculty of Mathematics and Computer Science

– Georg-August University Göttingen –

Telematics Group

Lab 2 - Gathering Sensor Data

Sensorlab



Contents

1 Introduction

In the first lab you have gotten acquainted with the general concepts of TinyOS. This lab will introduce actual use of wireless networking, as well as use of the MTS420 sensor board, including a number of sensors (light, humidity, temperature and air pressure).

Further Information

- Since this lab uses a number of different interfaces, the TinyOS interface reference should come in handy:
  `http://www.btnode.ethz.ch/static_docs/tinyos-2.x/nesdoc/iris/`
- Also take a look at the tutorial slides on Github:
  `https://github.com/tinyos/tinyos-main/blob/master/apps/AntiTheft/tutorial-slides.pdf`

2 Networking

To wirelessly communicate between motes via radio, the following interfaces are generally used:

```
uses interface Packet;
uses interface AMPacket;
uses interface AMSend;
uses interface Receive;
uses interface SplitControl as AMControl;
```

To look up events and commands provided by these interfaces (for AMControl look up SplitControl, it is usually renamed via "as" by convention) in the API documentation at:
`http://www.btnode.ethz.ch/static_docs/tinyos-2.x/nesdoc/iris/`

These interfaces are provided by the following components:

```
components ActiveMessageC;
components new AMSenderC(42); // 42 is an arbitrary ID for the network
components new AMReceiverC(42); // it has to be equal between motes so
                               // they can communicate.
// Example wiring:
App.Packet -> AMSenderC;
App.AMPacket -> AMSenderC;
App.AMControl -> ActiveMessageC;
App.AMSend -> AMSenderC;
App.Receive -> AMReceiverC;
```

For examples of its usage see:
`http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf`

You should also read:
`http://tinyos.stanford.edu/tinyos-wiki/index.php/Mote-mote_radio_communication`

For a full example application, you can take a look at the application in: `/opt/tinyos-2.1.2/apps/tutorials/BlinkToRadio/`
Take a look at the wiring of the components, as well as the definition of the packet type and used constants in the header file. You can also have a look at the examples in apps such as **RadioSenseToLeds** and **RadioCountToLeds** for an overall idea of how these applications work.

2.1 Assignment 2.1 (15 Points)

Make a copy of the `Blink` application and add radio communication to it. Increase a counter value every time a blink timer fires, and have the mote broadcast this counter value, so it can be received by a base station, the use of which will be explored in the next section. It should still blink on its own.

Hint: In the example wiring code above, the ID used with `AMSenderC` and `AMReceiverC` was 42. A receiver only receives packets sent by senders using the same ID. Since it is likely that more than one person is working on this assignment at the same time, please customize your network/packet IDs to some value of your choosing.

3 Gathering data from base station

Copy the program found in `/opt/tinyos-2.1.2/apps/BaseStation` into your home directory and install it on the BaseStation. For the output interpretation see:
`http://tinyos.stanford.edu/tinyos-wiki/index.php/Mote-PC_serial_communication_and_SerialForwarder#BaseStation_and_net.tinyos.tools.Listen`
Next run the following command on the PC connected to the BaseStation to see the packets sent by the mote from the previous assignment.

```
java net.tinyos.tools.Listen -comm serial@/dev/ttyUSB1:iris
```

It is to be noted that *ttyUSB1* in this case is different from *ttyUSB0* used for flashing. The `ttyUSB1` device is created when connecting a mote, even though it is not listed with the `ls /dev/ttyUSB*` command. It is used for listening to serial output. If multiple programming boards are attached to

the PC, they are listed as *ttyUSB0*, *ttyUSB2* etc., the corresponding device names for serial connections will be *ttyUSB1*, *ttyUSB3* and so forth.
If you take a look at some other example programs, you should find that some applications offer special Java programs for gathering data from the base station, which output things in a nicer format than a simple hex dump of the packet.

3.1 Assignment 2.2 (5 Points)

Include output from running the serial listening application in your report and explain the output.

4 Data acquisition from the sensor nodes

To acquire data from the sensors, the following interface will be helpful:

```
interface Read<val_t> {
  command error_t read();
  event void readDone(error_t ok, val_t val);
}
```

The following components provide the sender interface for the different sensors on the MTS400 boards:

```
       components new Accel202C();     // Senses: X and Y acceleration
       components new Intersema5534C(); // does not implement Read, but the
                                        // Intersema interface which has
                                        // basically identical command and
                                        // event except that readDone
                                        // receives an array of two uint16_t
                                        // values as val (uint16_t val[2])
                                        // Senses: temp, air pressure
       components new SensirionSht11C(); // Senses: temp, humidity
       components new Taos2550C();     // Senses: visible light, infrared
```

To build an application with support for the MTS400 sensorboard, build it as follows:

```
# Just building:
SENSORBOARD=mts400 make iris
# Build and install on mote:
SENSORBOARD=mts400 make iris install.2 mib520,/dev/ttyUSB0
# Build and reinstall on mote without rebuilding:
SENSORBOARD=mts400 make iris reinstall.3 mib520,/dev/ttyUSB0
```

- For a full example application including wiring, check:
  `/opt/tinyos-2.1.2/apps/tests/mts400`
- You should also read:
  `http://tinyos.stanford.edu/tinyos-wiki/index.php/Sensing`

- Another source of information is: `http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf`

4.1 Assignment 2.3 (30 Points)

Try building the example application and install it on a mote, remove it from the programming board, insert batteries and start it. Program another mote with the regular BaseStation application in:
`/opt/tinyos-2.1.2/apps/BaseStation`

Now you can build the Java application in:
`/opt/tinyos-2.1.2/apps/tests/mts400`
Remember to copy the `BaseStation` and `mts400` directories to your home directory and work on it there:

```
javac Mts400Tester.java
```

And run it to see the output from the BaseStation:

```
java Mts400Tester -comm serial@/dev/ttyUSB1:iris
```

Make a copy of the mts400 TinyOS application and try to modify it so that led2 is switched on or off, depending on the amount of visible light. Test it by moving the sensor mote from a bright to a dark place and back.

Also modify `Mts400Tester.java` and `DataMsg.h` sin such a way that the the sending mote's address is displayed for each received packet. In the raw packet, the sending mote's ID should be stored as a big endian 16bit value near the beginning of the packet. In your TinyOS application, you can get a mote's ID from the `TOS_NODE_ID` constant. Remove the output from the acceleration sensor.
Note: `DataMsg.java` should be automatically generated from `DataMsg.h`, when building the application.
Include output and source code for any modifications in the lab report.

Hint: Again you will want to customize your network/packet IDs so you don't receive packets from others who are also working on this assignment. This time, a symbolic name is used in the component declaration, but you can find and change its definition in the `DataMsg.h` header file.