Porting the ZigBit 900 Platform to TinyOS

Team Project

Richard Huber
Thomas Fahrni

Advisors:
Roland Flury
Philipp Sommer

Supervisor:
Prof. Dr. Roger Wattenhofer

Distributed Computing Group
Computer Engineering and Networks Laboratory (TIK)
Department of Information Technology and Electrical Engineering

February 2009

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed
Computing Group

Abstract

The *ZigBit900* module of MeshNetics [1] combines a low-power microcontroller (Atmel ATmega1281 [2]) and a radiomodule (ZigBee) in the 900MHz-Band (Atmel AT86RF212 [3]) on a single chip. *Meshbean900* and *Pixie* are two new nodes for wireless embedded sensor networks, using the ZigBit900 module. *TinyOS* [4] is a popular operating system for wireless embedded sensor networks. In this team project we extended TinyOS to support the ZigBit900 module. Therefore we added a new platform for each of the two nodes and adapted existing driver software (especially for microcontroller, radio module, LEDs, user buttons and UART) to the new environment.

Contents

1 Introduction

TinyOS [4] is an open source operating system designed for wireless embedded sensor networks. Its component library includes network protocols, distributed services, sensor drivers and tools for various different tasks. The component-based architecture with an event-driven execution model match the requirements effected by the constraints of power and memory. The programming language used to write applications and drivers is called 'NesC', which is in fact a dialect of C. In the past years, TinyOS has become a de facto standard in academic research on wireless sensor networks.

The ZigBit900 radio module by MeshNetics [1] combines a low-power microcontroller (Atmel ATmega1281 [2]) and a radio module (ZigBee) in the 900MHz-Band (Atmel AT86RF212 [3]) on a single chip. The two components are internally connected such that no external devices are necessary, except of a power source and an antenna.

MeshNetics offers development boards called *Meshbean900* [6] to demonstrate the capabilities of their ZigBit900 Module. The Meshbean boards can be programmed with *BitCloud* [7], MeshNetics own proprietary operating system. Up to now there was no support for TinyOS. Another sensornode using the ZigBit900 module is Pixie, developed at the TIK. It's a very basic platform, which can be used and extended in further projects.

The task of this team project was to extended TinyOS to support the ZigBit900 chip. Furthermore the Meshbean900 and the Pixie schould be intergrated into TinyOS as new platforms.

2 ZigBit900

The ZigBit900 module was developed by MeshNetics. It wires internally Atmel's ATmega1281V microcontroller and AT86RF212 RF transceiver. The module includes all the necessary passive components, so it can easily be mounted on a simple PCB with a minimum of required external connections. This makes it easy and fast to develop new prototypes. Also software debugging becomes easier, because hardware failures hardly happen. A block diagram of the ZigBit900 module is shown in Figure 2.1.

2.1 ATmega1281 Microcontroller

ATmega1281 is a low-power 8-bit microcontroller. It is part of Atmel's well known AVR series. The microcontroller runs with up to 16 MIPS throughput at 16 MHz and there is 128-KByte flash program memory and 8-KByte SRAM included. It uses the common programming interfaces JTAG and ISP, which both are accessable from outside the ZigBit900 (see Section 2.3).

2.2 AT86RF212 RF Transceiver

The AT86RF212 is a fully integrated 800/900MHz-band transceiver implementing the IEEE 802.15.4 standard. Its low current consumption makes it optimal for the use in wireless embedded sensor networks. The communication between the microcontroller and the RF module is fully encapsulated inside the ZigBit900 and uses the SPI bus.

2.3 Pin Assignment

Most - but not all - of the pins of the microcontroller and the RF transceiver are connected to the external pins of the ZigBit module. Unfortunately the way these pins are connected is not documented in the product datasheet [5]. Since the module comes boundled with an embedded software stack called *BitCloud* [7], the header files of this software could give us some hints on how this connection is done. The results we found are listed in Table A.1.
We also found a datasheet of an older version of the ZigBit module, where a table indicates

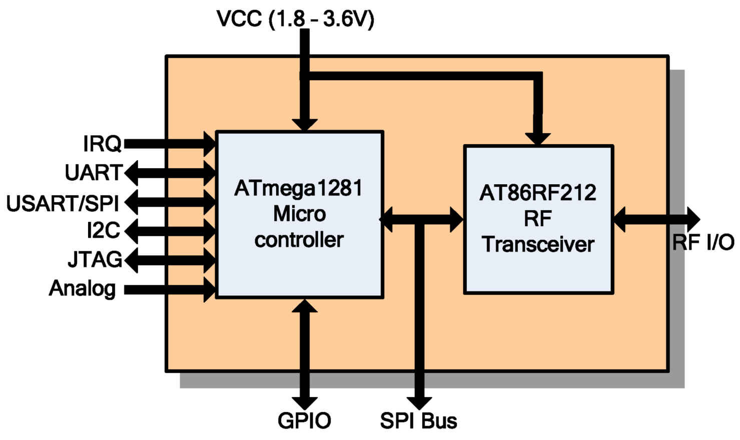


Figure 2.1: Block diagram of the ZigBit 900 module
(ZigBit Datasheet [5])

which pin of the microcontroller is connected to a given external pin of the ZigBit (see Appendix A.1). Although this older version uses a different radio module, all the pins we used were connected exactly the way it is described in this datasheet. Anyway, developers must be aware that this table has to be seen as a proposal and not as a fact.

3 The Meshbean900 Platform

The Meshbean900 platform is a development board, distributed by MeshNetics. It comes with a light- and a temperature sensor as well as a number of other peripheral devices, such as dip-switches, buttons and several connectors. A picture of the Meshbean900 is shown in figure 3.1. This platform offers a well-defined and well-tested environment, which makes software development easier since failures in the hardware can be disregarded. Unfortunately, the pins of the ZigBit900 which are internally connected to the ISP port of the ATmega1281 are not connected on the board. So the Meshbean900 platform has to be programmed by a JTAG programmer.[1]

3.1 The *.platform* File

Every platform in TinyOS has its own *.platform* file. It indicates the order of the include directories and some compile options. For the Meshbean900 platform, the *.platform* file belongs to the directory *tos/platforms/meshbean900*.

```
push( @includes, qw(

  %T/platforms/meshbean900
  %T/platforms/meshbean900/chips/atm1281/pins
  %T/platforms/meshbean900/chips/atm128/i2c
  %T/platforms/meshbean900/chips/rf212
  %T/platforms/meshbean900/chips/tsl2550
  %T/platforms/meshbean900/chips/lm73
  %T/platforms/meshbean900/chips/ds2411
  %T/platforms/iris
  %T/platforms/iris/chips/rf230
  %T/platforms/micaz
  %T/platforms/mica
  %T/chips/rf230
  %T/chips/atm1281
```

[1] Anyway the ISP bus can be accessed by soldering very thin conductors to the corresponding pins of the ZigBit900 module. But this can only be an emergency solution for example if one deactivates the JTAG interface by setting the wrong fusebits.

```
  %T/chips/atm1281/adc
  %T/chips/atm1281/timer
  %T/chips/atm128
  %T/chips/atm128/adc
  %T/chips/atm128/pins
  %T/chips/atm128/spi
  %T/chips/atm128/i2c
  %T/chips/atm128/timer
  %T/lib/timer
  %T/lib/serial
  %T/lib/power
  %T/lib/diagmsg
  ) );

  @opts = qw(

  -gcc=avr-gcc
  -mmcu=atmega1281
  -fnesc-target=avr
  -fnesc-no-debug
  -fnesc-scheduler=TinySchedulerC,TinySchedulerC.TaskBasic,TaskBasic,
     TaskBasic,runTask,postTask
  );

$ENV{'CIL_MACHINE'} =
    "version_major=3 " .
    "version_minor=4 " .
    "version=avr-3.4.3 " .
    "short=2,1, " .
    "int=2,1 " .
    "long=4,1 " .
    "long_long=8,1 " .
    "pointer=2,1 " .
    "enum=2,1 " .
    "float=4,1 " .
    "double=4,1 " .
    "long_double=4,1 " .
    "void=1,1 " .
    "fun=1,1 " .
    "wchar_size_size=2,2 " .
    "alignof_string=1 " .
    "max_alignment=1 " .
    "char_wchar_signed=true,true " .
    "const_string_literals=true " .
    "big_endian=false " .
    "underscore_name=false " .
    "__builtin_va_list=true " .
    "__thread_is_keyword=true";
```

Listing 3.1: **.platform:** Defining include directories for the Meshbean900 platform

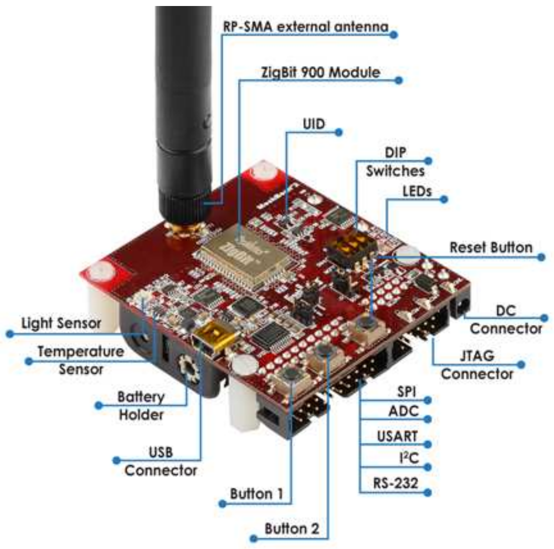


Figure 3.1: The Meshbean900 development board
(MeshNetics website [6])

3.2 The *platform.h* File

General declarations applying to the whole platform can be stated in the *platform.h* file, stored in the *tos/platforms/meshbean900* directory. For the Meshbean900 platform there are no such declarations, so the *platform.h* file is empty.

3.3 ATmega1281 Hardware Driver

The ATmega1281 is already used by several other platforms. Therefore all necessary information about the hardware driver can be found in the *tos/chips/atm1281* directory.

3.4 Makefiles

Every platform needs a makefile which provides an easy way to compile and install applications. We could reuse the existing makefile from the Iris platform with just a small change: The makefile from the Iris node disabled JTAG by removing the *JTAGEN* fusebit. This was particularly important because JTAG is the only way to program the Meshbean.

```
#-*-Makefile-*- vim:syntax=make
#$Id: iris.target,v 1.4 2008/07/09 15:36:50 sallai Exp $

PLATFORM = meshbean900
SENSORBOARD ?= micasb
PROGRAMMER ?= avrdude


AVRGCC_SUPPORTS_ATM1281 = $(shell avr-gcc --target-help 2>&1 | grep -
    c atmega1281)
ifneq ($(AVRGCC_SUPPORTS_ATM1281),1)
```

```
  $(error Found avr-gcc with no ATMega1281 support. For details on
      upgrading your toolchain, please see "http://www.isis.vanderbilt
      .edu/projects/NEST/tinyos-2.x-iris/doc/html/install-tinyos-iris.
      html")
endif

ifeq ($(PROGRAMMER),avrdude)
  PROGRAMMER_PART ?= -pm1281 -P usb -U efuse:w:0xff:m -U lfuse:w:0xc2
      :m
endif

ifeq ($(PROGRAMMER),uisp)
  PROGRAMMER_PART ?= -dpart=ATmega1281 --wr_fuse_e=ff
endif

#Setting Fusebits as default
AVR_FUSE_H ?= 0x99
AVR_FUSE_L ?= 0xff

ifdef RF230_CHANNEL
PFLAGS += -DRF230_DEF_CHANNEL=$(RF230_CHANNEL)
endif

$(call TOSMake_include_platform,avr)

meshbean900: $(BUILD_DEPS)
        @:
```

Listing 3.2: **meshbean900.target:** Makefile for the Meshbean900 platform

3.5 LEDs

The Meshbean900 comes with three LEDs: a red, a green and a yellow one. Since LEDs are part of almost every platform, we were able to reuse existing code. The only task was to find out, which LED is wired to which pin of the ATmega1281 (see Section 2.3). In the *PlatformLedsC.nc* file, these connections are stated:

```
configuration PlatformLedsC
{
  provides interface GeneralIO as Led0;
  provides interface GeneralIO as Led1;
  provides interface GeneralIO as Led2;

  uses interface Init;
}

implementation
{
  components HplAtm128GeneralIOC as IO;
  components PlatformP;

  Init = PlatformP.MoteInit;

  Led0 = IO.PortB5;  // Red LED
  Led1 = IO.PortB6;  // Green LED
  Led2 = IO.PortB7;  // Yellow LED
}
```

Listing 3.3: **PlatformLedsC.nc:** Defining pins for LEDs

3.6 User Buttons

The Meshbean900 platform has two user buttons, each providing a *GET* (polling) and a *NOTIFY* (interrupt) interface. In this case we did not just have to find the GeneralIO port - like we did for the LEDs - but also the corresponding interrupt addresses. This matching is done in the *HplUserButtonsC.nc* file.

```
configuration HplUserButtonsC
{
        provides interface GeneralIO as GeneralIOUB1;
        provides interface GpioInterrupt as GpioInterruptUB1;

        provides interface GeneralIO as GeneralIOUB2;
        provides interface GpioInterrupt as GpioInterruptUB2;

        provides interface Init;
}
implementation
{
        components HplAtm128GeneralIOC as GeneralIOC;
        components HplAtm128InterruptC as InterruptC;

        // GeneralIO
        GeneralIOUB1 = GeneralIOC.PortE6;
        GeneralIOUB2 = GeneralIOC.PortE7;

        // Interrupt UB1
        components new Atm128GpioInterruptC() as
           InterruptUserButton1C;
        InterruptUserButton1C.Atm128Interrupt -> InterruptC.Int6;
        GpioInterruptUB1 = InterruptUserButton1C.Interrupt;

        // Interrupt UB2
        components new Atm128GpioInterruptC() as
           InterruptUserButton2C;
        InterruptUserButton2C.Atm128Interrupt -> InterruptC.Int7;
        GpioInterruptUB2 = InterruptUserButton2C.Interrupt;

        // Init
        components HplUserButtonsP;
        Init = HplUserButtonsP;
        HplUserButtonsP.GeneralIOUB1 -> GeneralIOUB1;
        HplUserButtonsP.GeneralIOUB2 -> GeneralIOUB2;
}
```

Listing 3.4: **HplUserButtonsC.nc:** Defining pins and interrupt addresses for user buttons

The pullup resistors of the ATmega1281, required to indicate whether a user button is pressed or not, are switched off by default. The init procedure therefore has to activate them. This is done in the *HplUserButtonsP.nc* file.

```
module HplUserButtonsP
{
        uses interface GeneralIO as GeneralIOUB1;
        uses interface GeneralIO as GeneralIOUB2;
        provides interface Init;
}
implementation
{
        command error_t Init.init()
        {
                // Setting internal pullup resistors for both user
                   buttons
                call GeneralIOUB1.set();
                call GeneralIOUB2.set();

                return SUCCESS;
        }
}
```

Listing 3.5: **HplUserButtonsP.nc:** Activate pullup resistors for user buttons

3.7 Serial Port (UART)

The Meshbean uses a UART to USB Bridge to provide a USB interface. On the ZigBit900 module this converter chip is connected to UART1. To get it working we first had to add UART1 to the atm128 driver. This is done with the file *chips/atm128/Atm128Uart1C.nc* which is almost a copy of *Atm128Uart0C.nc*.
Then we declared UART1 as the platforms default serial port with *PlatformSerialC.nc*. The platforms default baudrate is 57600 and can be changed in *hardware.h*.
The file *chips/atm128/Atm128Uart.h* contained some wrong constants, which produced really strange results. The correct ones can be found in the ATmega 1281 datasheet and are:

```
ATM128_38400_BAUD_8MHZ_2X  = 25
ATM128_57600_BAUD_8MHZ_2X  = 16
```

Furthermore there exists the fusebit CKDIV8 (Divide clock by 8) which is set by default, for compatibility reasons. With this fusebit set, the clock-division factors would be completely different. In the makefile we specified this fusebit to be automatically removed.

3.8 Radio Module Hardware Driver

The radio module used on the ZigBit900 (AT86RF212) is very similar to the one used on the Iris platform (AT86RF230). Both radio modules use the same state machine and SPI protocol. Thus it was obvious to try to reuse the hardware driver of the Iris platform. After a few adaptions, the radio module turned out to run well with the hardware driver of the RF230. Still a problem is the difference between the two platforms in the wiring of the microcontroller and the radio module. On the Iris platform, interrupt signals from the radio module are sent to a pin of the microcontroller where the hardware is able to set a timestamp. The pin, which captures this interrupt on the ZigBit900 is not able to do this, therefore the software interrupt handler has to take over this task. Since the operating system may be in a atomic section, there could be an unpredictable time delay between the reception of an interrupt and the acquisition of the corresponding timestamp. Because of this delay, an exact synchronization is not possible with the current software version.

4 The Pixie Platform

The Pixie platform is a wireless sensor node, developed at the TIK. The developers wanted to design a platform, which is easily adaptable to any given problem. The result was a sensor node with only a ZigBit900 module, an antenna, a reset button and three LEDs. A picture of the first Pixie sensor node can be seen in figure 4.1. Since every pin of the ZigBit900 is wired to a connector, any kind of additional hardware (e.g. sensors, flash memory) can be mounted with a minimal effort. The simple design and the few hardware parts allows it to produce this sensor node fast and cheap.

4.1 Differences between Pixie and Meshbean900

In contrast to the Meshbean900, the Pixie has no sensors by default. Besides this, both nodes are almost identical and use the same software. The only difference concerns the LEDs, they are connected to different ports, as seen in Listing 4.1.

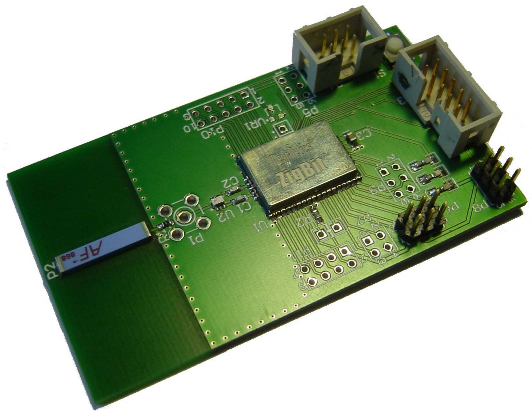


Figure 4.1: The Pixie Sensor Node

```
  Led0 = IO.PortG2;  // Red LED
  Led1 = IO.PortD7;  // Yellow LED
  Led2 = IO.PortD6;  // Green LED
```

Listing 4.1: **PlatformLedsC.nc:** (excerpt) Defining pins for LEDs

4.2 Programming the Pixie

On the Pixie sensornode, the JTAG and the ISP Interface are both available for programming. You can choose which one you prefer.

4.3 The Name *Pixie*

Pixies are mythical creatures of folklore, related to fairy. They are usually depicted with pointed ears, and often wearing a green outfit and pointed hat. In Devon (GB) pixies are said to be invisibly small, and harmless or friendly to man. These attributes are exactly the same we expect from a sensor node. Unfortunately, pixies are also said to sometimes play tricks on humans. As every engineer knows, such devices and software also sometimes seems to play tricks on developers, so *Pixie* seems to be the perfect name for our new sensor node.

5 Setting up the Development Environment

There are two major steps required to run an application on the ZigBit900:

- Compile the source code
- Download the output of the compiler to the ZigBit900

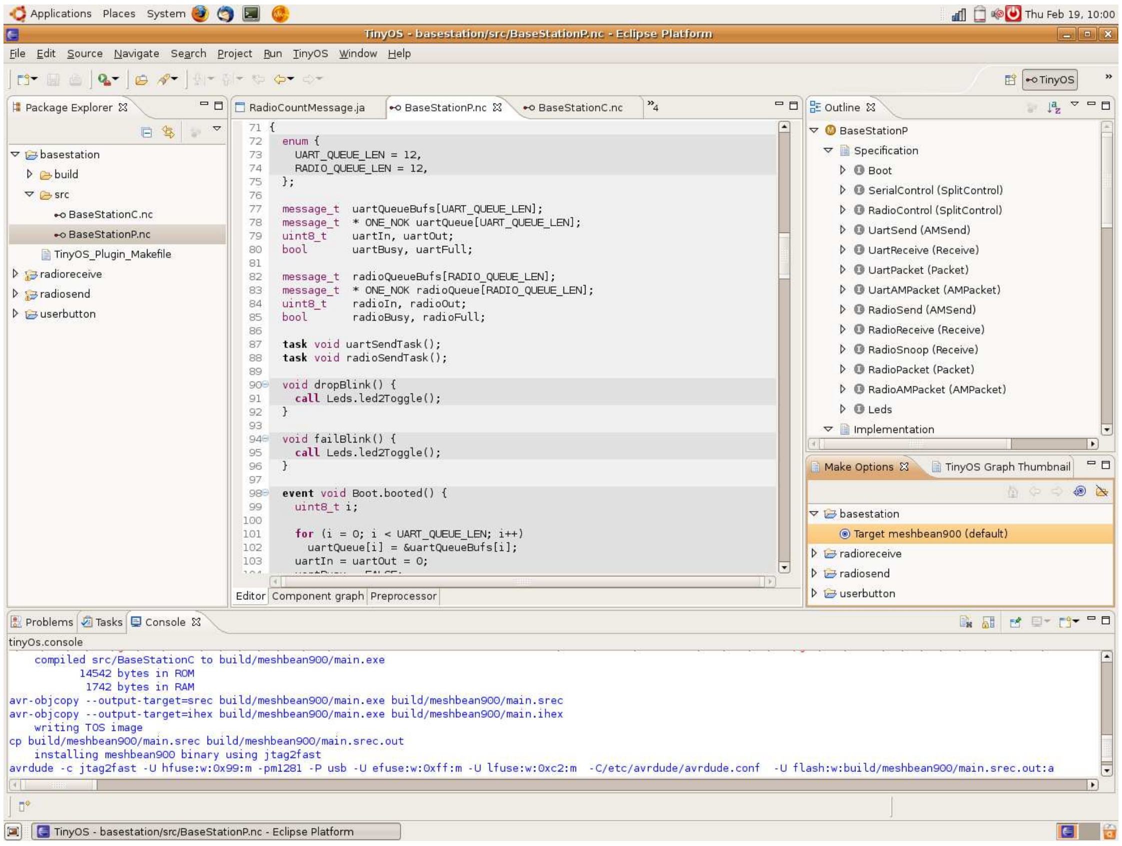
Compiling is done by the *NesC* compiler *ncc*, which is started by command line instructions. Makefiles can be used for the automation of this process. The executable files are downloaded to the microcontroller by *AVRDUDE*, using either the ISP or the JTAG protocol. Although it is possible to deal with these tools by hand, we strongly recommend to use the *Eclipse plugin* [8] instead, since it takes all these processes out of the hand of the developer.

5.1 Installing TinyOS

Before sourcecode can be compiled, TinyOS must be installed on the system. There are several methods how this can be done, depending on the operating system and the users habit. The various installation procedures are well documented on the TinyOS webpage [4].

5.2 The TinyOS 2 Plugin for Eclipse

There is a TinyOS 2 plugin for Eclipse, developed by the Distributed Computing Group at the ETH Zurich. It aims to provide developers with all the convenience functions expected from a modern development environment, such as realtime code validation and syntax highlighting to name but a few. A screenshot of the IDE can be seen in Figure 5.1. The plugin makes it possible to compile and flash an application with one single mouse click, so the developper does not have to mess around with makefiles and command line instructions. The plugin also provides a *component graph*, which graphically shows the dependences between the configurations and modules (see Figure 5.2).



```
{
  enum {
    UART_QUEUE_LEN = 12,
    RADIO_QUEUE_LEN = 12,
  };

  message_t  uartQueueBufs[UART_QUEUE_LEN];
  message_t  * ONE_NOK uartQueue[UART_QUEUE_LEN];
  uint8_t    uartIn, uartOut;
  bool       uartBusy, uartFull;

  message_t  radioQueueBufs[RADIO_QUEUE_LEN];
  message_t  * ONE_NOK radioQueue[RADIO_QUEUE_LEN];
  uint8_t    radioIn, radioOut;
  bool       radioBusy, radioFull;

  task void uartSendTask();
  task void radioSendTask();

  void dropBlink() {
    call Leds.led2Toggle();
  }

  void failBlink() {
    call Leds.led2Toggle();
  }

  event void Boot.booted() {
    uint8_t i;

    for (i = 0; i < UART_QUEUE_LEN; i++)
      uartQueue[i] = &uartQueueBufs[i];
    uartIn = uartOut = 0;
```

```
    compiled src/BaseStationC to build/meshbean900/main.exe
           14542 bytes in ROM
            1742 bytes in RAM
avr-objcopy --output-target=srec build/meshbean900/main.exe build/meshbean900/main.srec
avr-objcopy --output-target=ihex build/meshbean900/main.exe build/meshbean900/main.ihex
    writing TOS image
cp build/meshbean900/main.srec build/meshbean900/main.srec.out
    installing meshbean900 binary using jtag2fast
avrdude -c jtag2fast -U hfuse:w:0x99:m -pm1281 -P usb -U efuse:w:0xff:m -U lfuse:w:0xc2:m  -C/etc/avrdude/avrdude.conf  -U flash:w:build/meshbean900/main.srec.out:a
```

Figure 5.1: Screenshot of the Eclipse IDE with the TinyOS 2 plugin

The installation of the plugin starts directly form within Eclipse. For more information about the plugin and its setup visit the project's webpage [8].

5.3 AVRDUDE

Avrdude is a command line tool to program various AVR microcontrollers with many different programmers. Depending on the installation procedure you chose for TinyOS, you probably already have avrdude installed. Note that the Avrdude package in Stanford's TinyOS repository for Ubuntu was compiled without USB support and can therefore not be used with the *AVR JTAGICE mkII* programmer. We solved this issue by installing the package from the original Ubuntu repository.

5.4 AVR JTAGICE mkII

To download the compiled program to the microcontroller, we used the *AVR JTAGICE mkII* programmer. In Figure 5.4 you can see a picture of this programmer. AVRDUDE knows the JTAGICE mkII as *jtagmkII* and *jtag2slow* (running the programmer with default speed 19200 Baud) or as *jtag2* and *jtag2fast* (running at 115200 Baud). We decided to use the fast version and therefore had to create a new file in the */support/make/avr* folder called *jtag2fast.extra*. The content of this file can be seen in Listing 5.1.

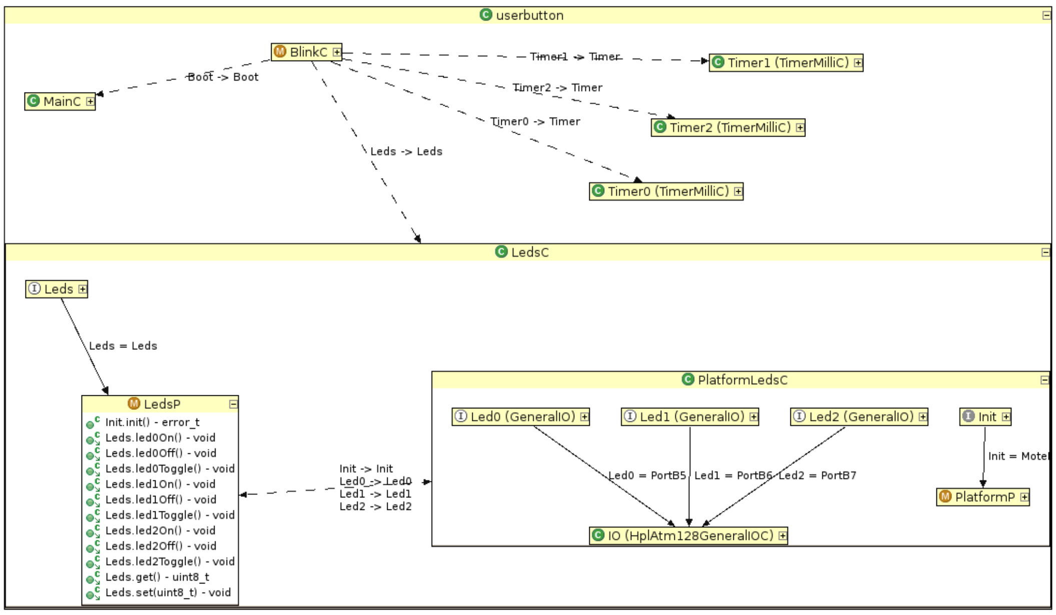
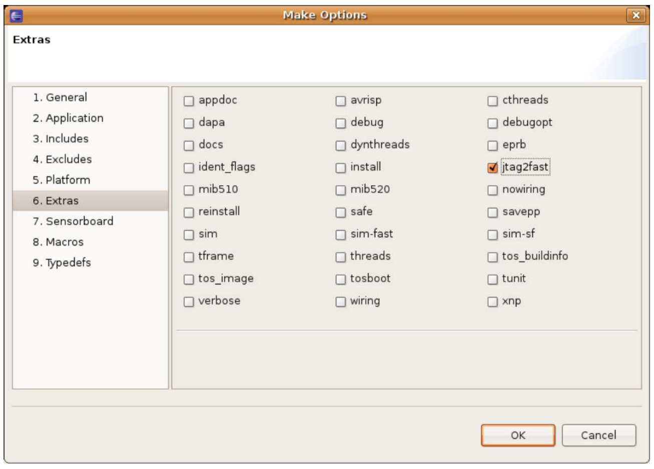


Figure 5.2: A Component Graph generated by the TinyOS 2 plugin

```
#-*-Makefile-*- vim:syntax=make
#$Id: jtag2fast.extra,v 1.7 2008/06/18 20:22:51 razvanm Exp $

PROGRAM = jtag2fast

ifeq ($(PROGRAMMER),avrdude)
    ifdef BOOTLOADER_IMG
      ifeq ($(shell [ -f /bin/cygwin1.dll ] && echo cygwin),cygwin)
        BOOTLOADER_IMG := $(shell cygpath -m $(BOOTLOADER_IMG))
      endif
    endif
    PROGRAMMER_FLAGS = -c jtag2fast -U hfuse:w:$(AVR_FUSE_H):m $(
       PROGRAMMER_PART) $(PROGRAMMER_EXTRA_FLAGS) $(
       PROGRAMMER_EXTRA_FLAGS_MIB)
    PROGRAMMER_INSTALL_SREC_FLAGS = -U flash:w:$(INSTALL_SREC):a
    PROGRAMMER_INSTALL_BOOTLOADER_FLAGS = -V -D -U flash:w:$(
       BOOTLOADER_IMG):a
endif

program: FORCE
        @echo "    installing $(PLATFORM) binary using jtag2fast"
        $(PROGRAMMER) $(PROGRAMMER_FLAGS) $(
           PROGRAMMER_INSTALL_SREC_FLAGS)

program_bl: FORCE
        @echo "    installing $(PLATFORM) bootloader using jtag2fast"
        $(PROGRAMMER) $(PROGRAMMER_FLAGS) $(
           PROGRAMMER_INSTALL_BOOTLOADER_FLAGS)
```

Listing 5.1: **jtag2fast.extra:** Introducing the AVR JTAGICE mkII programmer



[✓ jtag2fast]

Figure 5.3: Selecting the jtag2fast programmer in Eclipse

Figure 5.4: The JTAGICE mkII programmer

With this *jtag2fast.extra* file, Eclipse automatically creates a checkbox in the “Make Options/Extras”-Dialog (see Figure 5.3). Whenever this checkbox is selected, Eclipse tells AVR-DUDE to use the JTAGICE mkII programmer. The procedure of integrating another programmer will be exactly the same as long as the programmer is supported by AVRDUDE. A list of all supported programmers can be found in the *avrdude.conf*-file.
To allow AVRDUDE to use the programmer without root privileges, some additional UDEV rules are necessary. In Ubuntu you can do this with a new file */usr/udev/rules.d/51-atmel-programmer.rules* with this content:

```
# JTAGICE mkII
SUBSYSTEM=="usb_device", SYSFS{idVendor}=="03eb", SYSFS{idProduct}=="
    2103", GROUP="users", MODE="0666"
# AVRISP mkII
SUBSYSTEM=="usb_device", SYSFS{idVendor}=="03eb", SYSFS{idProduct}=="
    2104", GROUP="users", MODE="0666"
# Dragon
SUBSYSTEM=="usb_device", SYSFS{idVendor}=="03eb", SYSFS{idProduct}=="
    2107", GROUP="users", MODE="0666"
```

Listing 5.2: **51-atmel-programmer.rules:** Allow users to use the programmer

6 Evaluation

We tested the developped software by running test applications which indicate user interactios by either switching LEDs or by sending text messages to a computer using the UART.

6.1 Demo Application

To demonstrate the functionality of the sensor nodes, we developed a test application as shown in Figure 6.1. First we connected a light sensor to one of our Pixie nodes. The Pixie was then programmed to periodically sample the sensor and broadcast the measurement. On a Meshbean node we installed the TinyOS Basestation [4], which simply dumps everything it receives to the PC using the UART. On the PC we could now display the current sensor value in almost realtime.

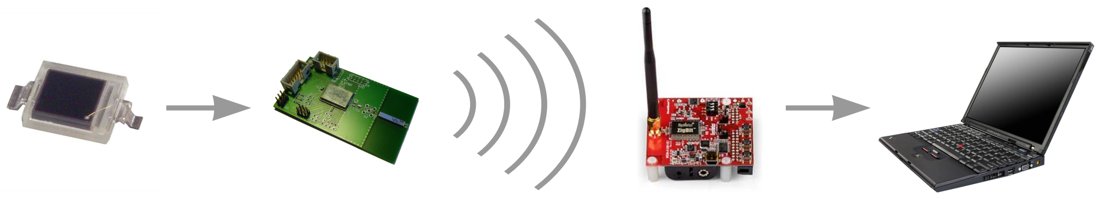


Figure 6.1: Block diagram of light sensor application

7 Conclusion

In this group project we implemented support for the ZigBit 900 module in TinyOS, reusing the hardware drivers of the ATmega1281 microcontroller and the AT86RF230 radio module. We introduced two new platforms called *Meshbean900* and *Pixie* for the two sensor nodes, which use the ZigBit 900 module. The microcontroller as well as the radio communication turned out to run very well with the abovementioned software. For the Meshbean900 platform we also implemented access to the onboard LEDs and user buttons. This software is well tested and runs solid. Furthermore we adapted an UART interface, so the meshbean 900 platform can communicate with a computer.

7.1 Further topics

The process of integrating the ZigBit 900 module and the corresponding platforms to TinyOS is not yet completed. As mentined in Section 3.8, the software is currently not able to run a synchronization. Since synchronization is an important skill of a wireless sensor network, further research should be spent on this topic. Other fields of interest could be:

- Measure the power consumption of the different platforms in order to estimate the battery lifetime.
- Evaluate the range of the radio in different environments and determine whether there is a difference in the range when the chip antenna is used compared to an external antenna.
- Develop drivers for the hardware on the Meshbean900 platform, which is not yet supported (i.e. DIP-Switches, temperature sensor, light sensor, ADC-converter, etc.)
- Additional hardware for the Pixie platform (like flash memory modules) could be implemented.

A Appendix

A.1 Tables of Pin Assignment

ATmega1281 Port	AT86RF212 Port
B5	CLKM
E5	IRQ
B0	/SEL
A7	/RST
B4	SLP_TR
B2	MOSI
B3	MISO

Table A.1: Connection of ATmega1281 and AT86RF212 inside the ZigBit900

ZigBit Pin	Function
9,22,23	Digital Ground
25,24	Digital supply voltage (Vcc)
36	Analog Ground
44,46,48	RF analog ground
45,47	Differential RF Input/Output

Table A.2: Functions of ZigBit pins not directly related to ATmega1281

ATmega 1281 Port	ZigBit Pin
PB0	-
PB1	1
PB2	3
PB3	2
PB4	-
PB5	4
PB6	5
PB7	6
PD0	11
PD1	12
PD2	13
PD3	14
PD4	15
PD5	16
PD6	17
PD7	18
PE0	38
PE1	39
PE2	40
PE3	41
PE4	37
PE5	-
PE6	43
PD7	42
PF0	33
PF1	32
PF2	31
PF3	30
PF4	29
PF5	26
PF6	28
PF7	27
PG0	19
PG1	20
PG2	21
PG3	7
PG4	-
PG5	36
PG6	-
PG7	-
AREF	34
XTAL_In	10
RESET	8

Table A.3: Pin assignment ZigBit / ATmega1281

Bibliography

[1] ZigBit 900 OEM Module, MeshNetics
http://www.meshnetics.com/zigbee-modules/zigbit900/

[2] Atmel ATmega1281, Atmel Corporation
http://www.atmel.com/dyn/products/Product_card.asp?PN=ATmega1281

[3] Atmel AT86RF212, Atmel Corporation
http://www.atmel.com/dyn/products/Product_card.asp?PN=AT86RF212

[4] TinyOS
http://www.tinyos.net/

[5] ZigBit 900 product datasheet
http://www.meshnetics.com/dl.php?id=41

[6] Meshbean Development Boards, MeshNetics
http://www.meshnetics.com/dev-tools/meshbean

[7] BitCloud (software stack from MeshNetics)
http://www.meshnetics.com/wsn-software/bitcloud/

[8] TinyOS 2 Plugin for Eclipse
http://www.tos-ide.ethz.ch