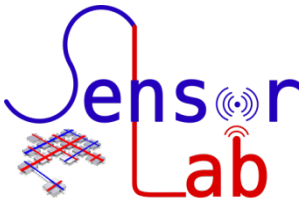

Georg-August-Universität Göttingen
Faculty of Mathematics and Computer Science
Institute of Computer Science
Telematics Group
Sensorlab

Lab 1 – Getting in Touch

Chencheng Liang – University of Goettingen – 11603960

Introduction

This lab get us to know the basis of Wireless Sensor Networks (WSN) which including TinyOS, nesC, and mote programming. First some knowledge of nesC is introduced, then by using nesC, we can directly programming the motes and control them.

The first example is to install Blink in a mote, first compiling with the command 'make iris', then use the command ' make iris reinstall.NodeID mib520,/dev/ttyUSB0' to install the program into the mote, then the LEDs in the mote are toggling.

Next we will write our own application which is simple and the detail is in Assignment 1. There are some questions about the theory of TinyOS and WSN, such as the requirements of operating system for mores, where the WSN is used, and understanding nesC coding, which are in Question 1.1,1.2 and 1.3.

The main purpose of this lab as the title is to get in touch with WSN in the both respects of software and hardware.

Question 1.1 [1]

Water quality monitoring: To give real time data of water quality to keep the drinking water is safe.

Air pollution monitoring: To give real time data of air quality, and also can collect the data the forecasting the weather.

Traffic controlling: by communicating and collecting coordinate data of the vehicles, to realize the traffic control.

Question 1.2 [2]

Efficient utilization of energy and power: Because the sensors have limited power supply(usually batteries), the operating system should optimize the use of energy, and only run the necessary program on the motes.

Small memory usage: Many PC operating systems are over 1GB now, but the sensors do not have so much memory, and sensors do not need so many functions which the general OS provided. Sensors only need to collect data and send them to the servers.

Support diversity in design usage: The OS for wireless sensor nodes must provide programming interface, so we can get the control of the nodes.

Remote control: Remote programming and controlling are also necessary, because if we want to change the program inside the sensors, we can do it remotely, rather than get them out of the environment and take them back to reprogram.

Question 1.3

The BlinkAppC is the configuration, it has 6 components and 5 other components(MianC, LedsC, Timer0, Timer1, Timer2) connect to component BlinkC.

In module BlinkC, it use the interfaces which other 5 components provided.

In BlinkC when the mote booted, three timers start at different time(250,500,1000).

After timer0 started at 250 ms print a debug message (Timer 0 fired at what time), then toggle the led0.

After timer1 started at 500ms print the similar message and toggle the led1.

After timer2 started at 1000ms, print the similar message and toggle the led2.

Assignment 1

Create a file RadioConfiguration.nc as configuration. Inside of it, wire MainC, LedsC, ActiveMessageC and TimerMilliC to MyRadioC which will implement this assignment.

Use SplitControl to imitate the radio.

In MyRadioC, first in the booted event which connected with MainC, we start a SplitControl which connected to the ActiveMessageC.

Then after the start() done, in the startDone event, we call Leds.led2On() to open LED3, then start a timer by calling call Timer.startOneShot(3000). After 3 seconds, the time is over then the event Timer.fired() will happen, and inside of it we stop the SplitControl. After the SplitControl done, The event SplitControl.stopDone() is happened. Inside of it we use call Leds.led2Off() to switch off the LED3, and use call Leds.led0On() to switch the LED1 on.

Lessons Learned & Conclusion

In this lesson I learned what is Wireless Sensor Networks (WSN). In the class room, some concepts and kits are introduced, such as TinyOS, nesC and sensor network architecture.

TinyOS is a operating system used in motes, because motes have characteristics, such as small and low power . NesC is a dialect of C, the main reason of using it is that it is optimized to reduce RAM, the code size, and prevent low-level bugs. In nesC the configuration connect the components.

After the components are connected, we the components can use or provide interface for each other. The module implements the components. Some interfaces need the events such as Boot, and we can write the code in the event function.

when the event happen, the code in that function will be executed. Usually all the program begin with Boot.booted event, and we can call other command in this event to implement our needs. If we call a command, that command must be implemented, and a corresponding event will be signaled after the command finished.

In the lab, it was my first time to use the motes and the programming modules. It is a huge step for me to see the Blink program work on the mote. The first application makes me understand how to control the motes and understand how to use nesC.

I think that nesC is more like a framework, because I used to work with Java Web programming. When user press that button then there is a function corresponding to that button.

In nesC when a event happened, there is a event function corresponding to it. And we can do programming under that function. Before I do this lab, I cannot direct control a hardware, now with TinyOS and nesC, I can do it.

References

[1] https://en.wikipedia.org/wiki/Wireless_sensor_network

[2] https://www.slideshare.net/snecute/tinyos