ENHANCING REAL TIME CAPABILITIES OF NANO-RK FOR TELOSB PLATFORM

CSC714-FALL 2011 FINAL REPORT

NORTH CAROLINA STATE UNIVERSITY
INSTRUCTOR: DR. FRANK MUELLER

TEAM MEMBERS
Devendra K Modium
Krishna Priya Kolla

1. INTRODUCTION

A wireless sensor network consists of spatial distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. These networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

Nano-RK is a fully preemptive reservation-based real-time operating system (RTOS) that can be used in wireless sensor networks. It includes a light-weight embedded resource kernel (RK) with rich functionality and timing support using less than 2KB of RAM and 18KB of ROM. Nano-RK supports fixed-priority preemptive multitasking for ensuring that task deadlines are met, along with support for CPU, network, as well as, sensor and actuator reservations.

Nano-RK currently supports full functionality for the Firefly sensor networking platform as well as the MicaZ motes. Our aim is to enhance the capabilities of Nano-RK for the Telos Rev. B platform (TI MSP430F1611). Motes that are generally used in wireless networks require radio stack to communicate with other motes in the network. Sensors are required to collect data from the environment they are posted in. These two components are critical for the usability of a mote in a sensor network. This project aims at implementing the radio stack and light and temperature sensor modules for Nano-RK on the TI MSP430F1611 platform

Telos Rev.B platform has the following components, Chipcon CC2420 (radio transceiver), photo synthetically active radiation sensor (light sensor), temperature sensor along with other sensors. Currently Nano-RK doesn't support the above mentioned components for Telos Rev.B platform. The main contributions of this project are,

1. We have implemented modules to provide support for the radio transceiver using Chipcon CC2420
2. We have provided support for light sensor (Hamamatsu S1087-01) for TI MSP430F1611 platform.
3. We have provided support for Temperature sensor (Sensirion SHT11) for TI MSP430F1611 platform.
4. We have implemented sleep mode using low power mode 3 (LPM3) on TI MSP430F1611 platform.
5. We have also provided support for the User Button on the above platform.

2. IMPLEMENTATION AND TESTING

2.1 Radio Stack Implementation

The goal of this module to is to extend the Nano-RK to have a working radio stack so that motes can communicate using RF in Nano-RK. The base Nano-RK code, upon which we started working on, has the higher level APIs to send and receive packets. After analyzing the code, we have

found the base Nano-RK doesn't have the essential functionality of selecting and initializing USART module for SPI interface to CC2420, configuring CC2420 and hardware abstraction library (HAL) functions of communicating CC2420 over SPI bus. Necessary functionality is added to the Nano-RK code to make radio stack work.

Configuration of the CC2420 is performed using general I/O pins and the MSP430's SPI interface. Basic initialization of the CC2420 is done by asserting the RESET pin active low for at least 1 μs. The VREG_EN signal must also be set for the transceiver to operate. After these two conditions are met, the MCU can start operating the RF transceiver over the SPI bus. Before any further action can take place, the CC2420 oscillator must be allowed time to stabilize. Bit 6 of the status byte holds the state of the oscillator.

MSP430F1611 has two identical USART modules USART0 and USART1. In our implementation, USART1 module is used in UART mode for asynchronous serial communication with terminal and USART0 module is used in SPI mode for synchronous communication with hardware module CC2420.

MantisOS along with MSP430F1611 hardware data sheets are taken as reference for extending Nano-RK to include radio stack functionality wherever needed. To integrate the RF communication with Nano-RK periodic task model, polling is used in the RF receiver. In each cycle, RF receiver checks for the packet arrival and reads the packet from CC2420 hardware RX buffers if one exists.

2.2 *Radio Stack Testing*

Basic testing of Radio module is done to make sure correct functionality is added to Nano-RK. In this test, both transmitter and receiver contain a periodic task with configurable timing parameters (period, phase, execution time). In each cycle the transmitter sends a packet with value 1 or 0 alternatively in the payload. The transmitter sets the Red LED when it sends one and clears it when it sends a 0. The receiver task runs at a higher frequency as compared to the Transmitter. In each cycle the receiver checks if a packet arrived over radio, if value 1 is received it sets the Red LED and clears it if it receives a 0. The receiver task does nothing if no packet is received. Since polling method is used, the frequency of receiver task is made higher than the transmitter task. Results show that the two motes are able to successfully communicate over radio and good synchronization is achieved over radio by configuring the timing parameters of transmitter and receiver tasks appropriately.

Apart from this test, separate tests were done where large messages (size 40 bytes) are sent from one mote to other over radio. The CRC check passed for these messages. We also tested them using UART. Byte by byte of payload from transmitter is checked with receiver to make sure that no bytes got corrupted.

2.3 Temperature Sensor Implementation

The aim of this module is to implement a driver for the temperature sensor SHT11 in Nano-RK, so that applications can access readings from the temperature sensors. Telos Rev.B platform has

Sensirion SHT11 temperature/humidity sensor. The SHT11/SHT15 sensors are calibrated and produce a digital output. The calibration coefficients are stored in the sensor's onboard EEPROM. The sensor is coupled with a 14-bit A/D converter.

Implementation of the Temperature driver involves code for sensor initialization and also api to read from the sensor. Initialization of senor involves setting direction of the POWER, SCLK and DATA pins of the SHT11 sensor followed by setting SCLK pin low and DATA pin high. However the sensor isn't powered yet. Reading the temperature from the sensor involves predefined steps. It starts with a transmission start sequence which involves falling edge on DATA pin, low pulse on SCLK pin followed by rising edge on DATA. The sensor can be used to get Temperature as well as Humidity sensor readings. To obtain the temperature readings, after the transmission sequence appropriate command needs to be sent to sensor using DATA and SCLK pins. Once ACK is obtained from the SHT11, waiting for measurement to complete is necessary before the temperature readings can be read from the sensor and supplied to application.

Nano-RK has generic API for accessing any driver. All the above explained implementation of the temperature sensor is written complying with the Nano-RK generic driver API.

The raw sensor reading obtained from the sensor can be converted into Celsius using the following formula,

Temperature in celsius = -39.60 + 0.01*(raw_ temp_sensor _reading)

2.4 Temperature Sensor Testing

Testing of the Temperature sensor is done by comparing the obtained temperature readings with those obtained from the sensor driver already implemented for TinyOS. To avoid any sensor hardware offsets, same Tmote is used while obtaining readings from Nano-RK and TinyOS. Temperature readings are recorded in different locations with different temperatures. The temperature readings of the SHT11 sensor are noted using the implemented sensor driver and UART implemented for Nano-RK and also readings of the SHT11 sensor of the same mote are obtained using TinyOS in same environmental conditions. The results obtained are shown in table 1. It can be seen that temperatures obtained in Nano-RK using the implemented SHT11 sensor driver are identical to the temperatures obtained in TinyOS.

Location	Nano-RK	TinyOS
1	19	19
2	23	23
3	25	26

Table 1. Measured Temperature in Celsius for NanoRK and TinyOS

2.5 Light Sensor Implementation

The Telos Rev.B motes are equipped with a Hamamatsu S1087 photo-diode for sensing photo synthetically active radiation and Hamamatsu S1087-01 photo-diode for sensing the entire visible spectrum including infrared radiation. Both of them give their measurements in lx. These analogue sensors are connected to the mote's micro-controller ADC pins, S1087 to ADC4 and S1087-01 to ADC5. We have implemented a software driver so that applications can access the readings from the Hamamatsu S1087-01 component. The implementation is done complying with the generic NanoRK API.

Based on the graphs available in the Hamamatsu S1087-01 datasheet, the current of the sensor, I, may be converted to Lux using the formula

$$lx = 0.769 * 10^6 * I * 1000 \text{ ---------- (1)}$$

In order to obtain the value of I we need to follow these steps,

Light sensor values are read using the micro controller's 12-bit ADC. To convert the raw value obtained from the ADC to the corresponding voltage, we perform the calculation:

$$V(Sensor) = (value/4096) * Vref \text{ ----------(2)}$$

The photo-diodes create a current through a 100kOhm resistor. After calculating the voltage using equation (2) above, we convert the voltage into a current using V=IR

$$I = V(Sensor) / 100000$$

Finally we can obtain the light sensor reading in lx using the (1).

2.6 Light Sensor Testing

Testing is done using two applications specified in Homework 1. The first application obtains the light sensor reading periodically and sets an LED if the reading is greater than a threshold value. The LED is cleared if the value is less than the threshold. It was observed that the LED goes off whenever the light sensor is covered with a finger and stays on otherwise.

The second application involves one mote which toggles an Led with every period. The other mote obtains the light sensor readings at a higher frequency and sets an LED if the reading is greater than a threshold otherwise the LED is cleared. For this application we first have to obtain the light sensor reading for ambient light and set the threshold accordingly. In order to do this, for the first few cycles the ambient light reading is obtained with all LEDs off and a threshold is calculated based on the average of the ambient light readings. The actual sensing application begins only after the first 10 cycles. It was observed that both the motes toggle their LEDs in a synchronized manner provided they are placed close enough to each other.

2.7 Power Saving Mode Implementation

Power is very expensive and critical resource for operation of Motes in wireless networks. Implementing low power mode as and when needed is essential to conserve battery power. MSP430 platform is designed for ultralow-power applications and uses different operating modes. This

platform has five different power saving modes. The low-power modes 0–4 are configured with the CPUOFF, OSCOFF, SCG0,and SCG1 bits in the status register The advantage of including the CPUOFF, OSCOFF, SCG0, and SCG1 mode-control bits in the status register is that the present operating mode is saved onto the stack during an interrupt service routine. Program flow returns to the previous operating mode if the saved SR value is not altered during the interrupt service routine. Program flow can be returned to a different operating mode by manipulating the saved SR value on the stack inside the interrupt service routine. The mode-control bits and the stack can be accessed with any instruction. When setting any of the mode-control bits, the selected operating mode takes effect immediately. Wake up is possible through all enabled interrupts.

The most important factor for reducing power consumption is using the MSP430's clock system to maximize the time in LPM3. LPM3 power consumption is less than 2 μA typically, with a real-time clock function and all interrupts active. A 32-kHz watch crystal is used for the ACLK and the CPU has a 6-μs wake-up.

Currently Nano-RK has high level framework, to decide when to enter and exit sleep mode. But it doesn't have the essential functionality on how to enter and exit the low power mode. In the current Nano-RK, when the idle task is scheduled, it checks if the next wakeup period is greater than NRK_SLEEP_WAKEUP_TIME it calls nrk_sleep which is supposed to enter low power mode. We have implemented the nrk_sleep to enter the low power mode using Timer, which is sourced from the ACLK. In the nrk_sleep, TimerB is started and then CPU enters LPM3 mode where CPU, SMCLK, MCLK and DCO are off whereas ACLK is on. Once TimerB expires, the interrupt service routine of TimerB is called, where the CPU exits from the low powering mode.

2.8 Power Saving Mode Testing

Testing the functionality of the power saving mode is done using LEDs, by tracking the entering/exiting of low power mode. The system is stable with the functionality of low power mode and is running as expected with the basic tasks. Testing is done with different number of tasks in system with different timing configurations and it is observed that system is stable.

2.9 User Button Implementation

The Telos Rev. B motes have both a reset button and a user button. Currently NanoRK does not support the user button for this platform. We implemented functions to provide applications access to the button state. This involves enabling the user button and implementing the interrupt service routine. Applications can access the button state using a global variable BUTTON_STATUS. Initially this variable is set to B_ON. Whenever the user button is pressed an interrupt is generated. The interrupt service routine toggles the BUTTON_STATUS between B_ON and B_OFF.

2.10 User Button Testing

The User button implementation is tested using a simple application which toggles an LED every period second only when the BUTTON_STATUS is on. It is observed that initially since the

BUTTON_ STATUS is B_ON the Led toggles every second. When the user button is pressed the LED stops toggling. The LED toggling can be turned back on by pressing the user button again. The User button is also used for the test cases for the Radio and Light sensor, in order to make them similar to the tests mentioned in Homework 1.

3. CONCLUSION:

Necessary functionality to enable and use the radio transceiver (Chipcon cc2420), light sensor (Hamamatsu s1087-01) and temperature sensor (Sensirion SHT11) in NanoRK for the TI MSP430F1611 platform are implemented and tested for correctness**.** Power saving mode is implemented using low power mode 3 and tested. Also necessary functions to enable and use the user button are implemented. These added functionalities make NanoRK more usable in Wireless sensor networks for TI MSP430F1611platform.

4. FUTURE WORK:

1. Currently we are using only Low power mode 3 to implement nrk_sleep. This implementation can be improved by using different low power modes depending on the value of sleep time.
2. We have implemented the basic low level functionality to use the radio transceiver on Telos Rev.B platform. Higher level protocols like TDMA suitable for various applications can be implemented on top of this and tested.

TIMELINE

October 24 - November 1 (Devendra and Krishna):

Study required data sheets and refer to implementation of Radio stack for TI MSP430x5438

November1 - November 8 (Devendra and Krishna):

Completion of radio stack module in Nano-RK

November 8 - November 20 (Devendra):

Completion of Light & Temperature sensor modules in Nano-RK

November 8 – November 20(Krishna):

Completion of power save mode and user button module in Nano-RK

November 20 – December 2(Devendra):

Testing the code for radio communications, light sensor and temperature sensor functionality with the same text cases as in HW1

November 20 – December 2(Krishna):

Testing the code for radio communications and power save mode and User Button.

December 2 – December 6 (Krishna and Devendra):

Writing final report, thorough testing with integrated code and writing test cases for submission

WEBSITE

http://www4.ncsu.edu/~kkolla/CSC714/proj.html

PROJECT FOLDERS IN SUBMITTED CODE FOR DIFFERENT TEST CASES

The submitted file nano_RK_proj.tar.gz has the following folders for different test cases,
Radio Stack: basic_rf
Light Sensor: basic_light, blink_light
Temperature Sensor: basic_temp
User Button: button_test
Power Saving mode: sleep_test

Each of these folders has a Readme file that provides necessary details to run the tests.

We have also submitted the file nano_RK_proj_sleep.tar.gz, this file can be used to track the entering and exiting of low power modes using LEDs. Use project sleep_test for this test case.

REFERENCES

- *Nano-RK: an Energy-aware Resource-centric RTOS for Sensor Networks,* 26th IEEE International Real Time Systems Symposium RTSS05 (2005) by Anand Eswaran, Anthony Rowe, Raj Rajkumar.
- MSP430x1xx Family Data Sheet
- MSP4301611 Part Specific Data Sheet
- Nano-RK website www.nanork.org
- Mantis OS website www.mantisos.org
- Wireless Sensor Networks http://en.wikipedia.org/wiki/Wireless_sensor_network
- CSC 714 Lecture Slides