Abstract #202709

Wireless Sensor Array Demonstration

John Militzer, **Steve Semmer**, NCAR, Boulder, CO.

Abstract- To meet today's need for more surface measurements, wireless communication technology must be incorporated. Hardwired communication prohibits the ability to deploy large arrays of multiple sensors in a timely fashion and inhibits opportunities to deploy instruments in optimum locations per the needs of sensing. WISARD was a development which incorporated wireless technology for two types of measurements: (1) solar radiation, and (2) soil parameters. A second goal in the development was to demonstrate the concept of distributed processing by dedicating a microprocessor to each sensor. This paper presents some of the technical challenges encountered, the technology used, and the results of a successful deployment during a winter field deployment.

I. Introduction

The first use of wireless communication at NCAR for surface meteorological research dates back to the 1970s [1]. The communication equipment of that time had limited range, simple networking capability, and had high power requirements. With advances in technology, the use of wireless communication expanded. One of the first uses of these advances was by Industry to remotely monitor equipment in industrial plants. In 2002 the Center for Embedded Networked Sensing (CENS) was created to focus on wireless sensing networks, WSN, and set out to explore new technologies to expand the use of wireless communication.

In parallel with advancements in wireless communication, the need to increase the spatial distribution of measurements of surface meteorological parameters became apparent. It was recognized that a single point measurement of soil parameters rarely represented the region of interest. To properly characterize a complex environment, such as a forest canopy or diverse rangeland, multiple measurements are needed.

In this paper we discuss the development of a wireless sensor network dedicated to surface meteorological research.

II. History 2003 – 2008

A. Previous Efforts

Based on feedback from the scientific community, we started exploring techniques that would allow a significant increase in surface measurements. We quickly realized that some form of wireless communication would be required. A search of commercially available, low powered wireless systems led to the MICA2 platform [2]. In 2004 we deployed a WSN soil temperature network based on the MICA2 using the TinyOS software package. 72 temperature elements were sampled at 3 locales with 4 sites each. Six probes were attached to each MICA and a 5th unit received and forwarded the 30-second

sample messages to a local data system. Although TinyOS supports mesh networking, only a simple star configuration was needed. This MICA2 deployment provided useful insights into using a WSN in our applications. One issue was that the MICA's 12-bit analog-to-digital converter (ADC) was inadequate for research grade measurements despite oversampling. Another issue was the TinyOS software package. It required staff to learn a new unique language and became cumbersome.

In 2007 and 2008 we partnered with the University of Colorado to develop a WSN to measure photosynthetic active radiation (PAR), in a forest canopy [3]. The goal was to design a network consisting of 324 PAR sensors distributed vertically at 3 levels on 9 trees. Each level had 12 PAR sensors divided between 4 wands. Each wand also included an attitude sensor. The engineering task was broken into 3 parts: (1) sensor design and interfacing (2) transducer calibration and (3) wireless network software to handle the flow of data in a distributed processing environment. While NCAR handled the sensor issues, C.U.'s Department of Computer Sciences focused on wireless networking using their Mantis Operating System (MOS) [4]. MOS offered the advantage of standard 'C' language programming versus the TinyOS environment. On each tree at each level a dedicated processor/radio module was used to collect data from the 4 wands. In 2007 this board was based on a MICA2. In 2008 the board was based on a TelosB. Ultimately the Wireless mesh networking efforts were unsuccessful largely due to the science requirement for continuous 1-second sampling of all PAR transducers. A soil array was also deployed using mesh communications. It could only support 1-minute reporting while suffering excessive battery consumption due to multiple message rebroadcasts and inadequate sleep cycles. All mesh techniques attempted ultimately proved unreliable and cumbersome. A simple polled staggered TDMA reporting method worked well at 3s but could not be fully tested at the 1s rate prior to deployment. Code space restrictions (MICA) and a complex RF environment presented other difficulties. Instead, 3 NCAR/EOL data systems (DSM) were employed to collect processor board messages using hard-wired links from 3 trees in 3 separate groves. The DSMs subsequently relayed data via line-of-sight point-to-point radios to a distant hub and ultimately NCAR facilities via the internet.

B. Lessons

Scientifically these were successful efforts. Sensor performance and sampling met or exceeded the research objectives. Mixed results investigating various wireless techniques had clear implications. (1) Development of a wireless multi-hop, self-healing mesh network is a major effort beyond the scope of most observational researchers. (2) Limiting the number of radios and/or sampling rates enhances probability of success using mesh and hopping techniques. (3) Accurate time stamping of data is critical. (4) Elaborate RF based time-of-arrival estimations to synchronize time-keeping and/or triangulate site locations in a mesh environment are complicated and involve excessive overhead. (5) Adapting commercially available Wireless Networking Radios is our preferred approach to meet future field obligations.

III. Wisard Development

We viewed Wisard as a logical extension of our primary high-end Linux-based data acquisition system, DSM. The DSM already supports multiple interfaces, high rate sampling, secondary product derivations, data archival and long-range radio/internet/cell-modem capabilities. Wisard implements

wireless connections with sensor clusters located up to 100 meters from the main DSM or further with multiple hops.

NCAR/EOL developments are driven by the needs of the scientific community. One characterization often requested is the surface energy budget. This requires a suite of both soil and solar radiation measurements. For soil, we use a set of 4 separate sensors at each measurement location. To get a true representation of the surface one must deploy multiple sets of these instruments. Incoming solar radiation does not require multiple measurements. Outgoing and albedo parameters do because readings can be quite different over non-homogeneous ground. Obtaining a representative surface energy budget then may require radiation and/or soil measurements to be taken at 4 to 6 distinct locations for a specific site. Without the use of a wireless network there is a physical challenge connecting all of these sensors to a central data acquisition system. Cable integrity may be compromised (rodents) or routing restricted (roads, ditches). Also, conductors lying on the ground can experience high amounts of induced current from lightning.

WISARD Sensors	
sensor	measurement
Kipp & Zonen CG4	longwave radiation
Kipp & Zonen CM21	shortwave radiation
REBS Q7 Net radiometer	net radiation
NCAR soil temperature probe	4 levels of temperature
Decagon	soil moisture
REBS heat flux plate	soil heat flux
Hukesflux TP01	thermal conductivity

A. Design Goals

Focusing on this suite of sensors, we set out to achieve these goals: (1) distributed processing, (2) research grade measurements, (3) near real-time data at a continuous 5-second rate, (4) accurate time stamping, (5) two-way mesh-network communications, and (6) local backup storage. We did not require extremely low power consumption since we could scale up our power systems. Sampling rates faster than 1 sample/s were not required for the sensors that we wanted to spatially distribute.

B. Hardware Implementation

We chose to address many of these goals by expanding on two successful approaches: the intelligent sensor and a server node (mote) to support them.

1) Intelligent Sensors.

Intelligent Sensors provide distributed processing at each measurement location. Embedded calibrations and unit identifications permit sensors to be arbitrarily moved from location to location, or replaced, without the need for editing a centralized database elsewhere. An intelligent sensor consists of the following components: (1) a microprocessor, (2) a common communication protocol, (3) a transducer coupled to the medium being measured, (4) a front-end interface dedicated to a specific transducer(s), and (5) a unique sensor calibration. The Microchip PIC18F2520 was selected as the microprocessor having adequate memory and digital I/O lines for our intelligent sensor boards. The PIC18F2520 also provides two options for communication, RS232

and I2C. Both options were implemented adding flexibility to the sensor. Figure 1 is an example of the interface board used with a Kipp & Zonen CG4 pyrgeometer.

Wisard sensor transducers output slow changing DC voltages ranging from ±10 mV to 0-2 V full scale. To handle this wide voltage range, the Microchip MCP3424 18bit differential ADC was chosen. The MCP3424 has up to 4 input channels and 4 gain settings allowing a resolution down to ±1.9 µV per bit. This eliminates the need for any signal amplification. A reference voltage source is also provided on the interface board for transducer excitation if required. Calibration of the sensor consists of two steps. The primary calibration focuses on the response of the transducer. Regression analysis of the raw measurements compared with traceable standards produce a unique set of coefficients that are stored in the microprocessor's non-volatile memory. The secondary calibration deals with any temperature dependence of the electronic components. By including a temperature sensor on the board and subjecting it to temperatures from -30 C to 40C, a temperature correction can be applied to the electronics, Figure 2. Each sensor's microprocessor board is put inside a radiometer or encased in a weatherproof cover and potted to prevent damage from exposure to the elements. This also provides further isolation from high-frequency temperature and electromagnetic fluctuations.

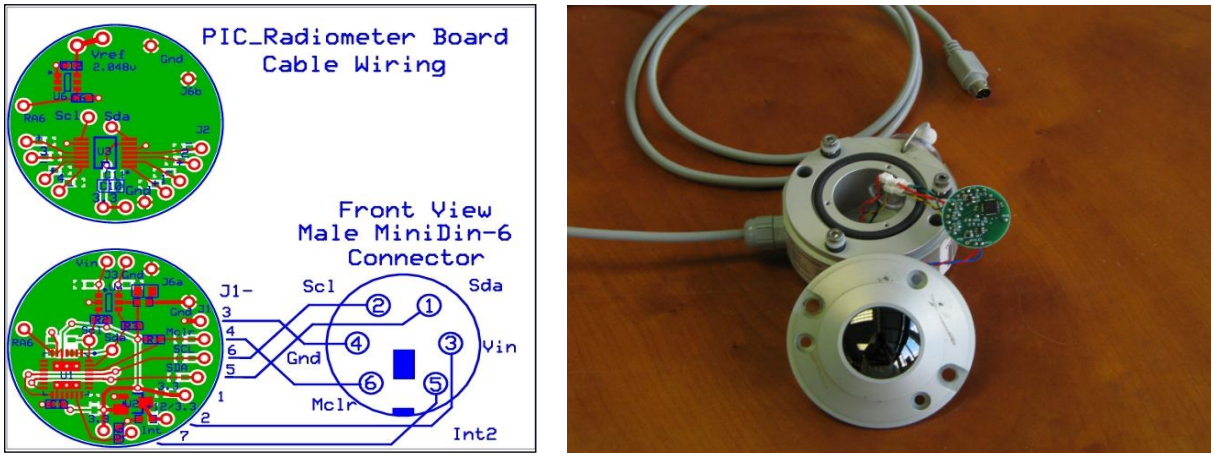


Figure 1: The pyrgeometer electronics required a stack configuration. The processor board included a single channel ADC to measure the thermopile transducer. The second board had a 4 channel ADC to measure multiple thermistors imbedded in the sensor

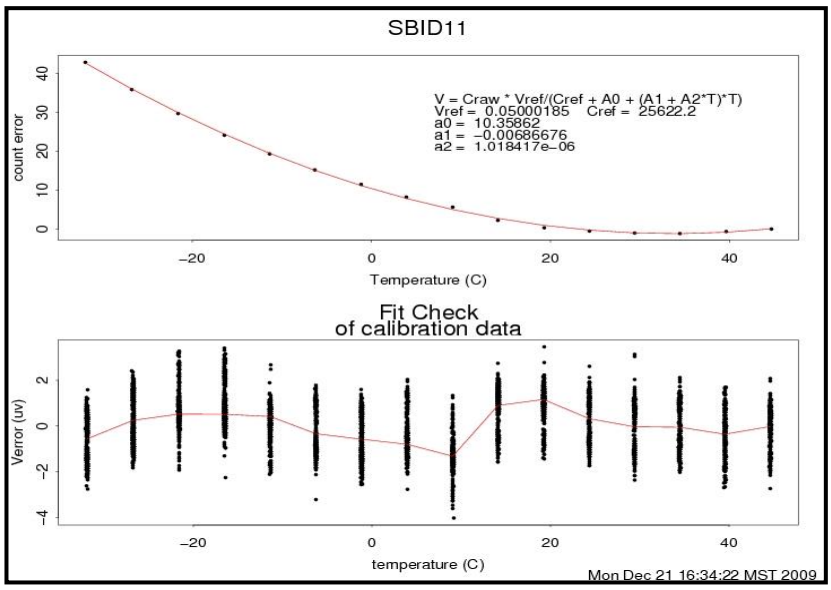


Figure 2: ADC temperature dependency. The upper plot shows ADC count error with respect to temperature. The lower plot shows correction to ADC output (1 count = 1.9 uV)

2) Server Nodes.

Server nodes (motes) acquire data messages from a cluster of intelligent sensors. They were designed around the Microchip PIC24FJ64GB004 Processor, and provide all the power, connectors, interfacing and assorted hardware to support basic operations. It is a single board design with all components fitting inside a standard enclosure available from Decagon Devices Inc. A mote polls each sensor, packs their data into a single message and telemeters it over a wireless link to the primary DSM. Up to 5 sensors may be attached to a mote over a standard DIN-6 cable that provides both power and communications. We chose an I2C interface for communications. I2C is a two wire bus (plus power) with a master device (mote) providing the clock and initiating data transfers from multiple bussed slave devices (sensors). Sensors can be attached and removed from the mote dynamically. A push-button is provided to trigger the mote to scan for what sensors are attached when one is added or removed. Mote software periodically re-scans to determine if any have been added or lost.

3) Radio Communications.

Mesh networking offers distinct advantages over simple star connections. Motes can be 'chained' to relay data when not in direct contact with the DSM or added and removed dynamically. At about the time we began Wisard, Digi Corp. began offering a new commercial product supporting mesh networking. Because of previously successful experiences using Digi(Maxstream) radios, we based our design using their embedded Xbee-DigiMesh (DM) modules. Although DM is a proprietary technique it is a good alternative to Zigbee based solutions. Both offer message hopping and self-healing capabilities. However, DM provides peer-to-peer rather than parent-child relationships. All nodes can be both a router and end-device. This significantly improves the ease with which a mesh network can be setup. In addition each can be put into synchronized sleep mode to save power. All radios synchronize between themselves automatically. Although Xbee's

consume a relatively large amount of energy compared with other sensor-network packages (ex. Mica, Telos), they are a drop-in solution and have gained wide acceptance especially in the industrial community.

4) Stand-Alone Capability.

Mote boards were designed to operate independently, to provide an accurate time stamp with each sample collected and to store all data locally. This allows stand-alone operation and data acquisition to continue if the radio link or the primary DSM goes down. Long-term operations in this mode are feasible; however two-way communication and the associated control capability would be lost. Local data storage is provided using microSD flash-memory. Data messages are stored in files using the standard 'FAT-16 or 32' file format. A 2GB flash is adequate for storing six months of data for 5 sensors being sampled every second.

Accurate time keeping is established using a local GPS module, the NEO-6 manufactured by Ublox. The GPS site location is also archived along with the attached sensor identifications and data. Because the gps consumes a high amount of power relative to other components, it is cycled on at startup to synchronize operations and then turned off. Thereafter it is periodically enabled to update a local clock. GPS synchronization typically occurs well within 30-seconds. The local clock is based on either the PIC's internal real-time clock calendar (RTCC) or a high quality but higher power temperature controlled crystal oscillator circuit (TCXO). Since the highly accurate GPS pulse-per-second signal is not available most of the time, either of these sources can trigger the processor at the beginning of every second. Thus, sample times are quantized to one second intervals. The RTCC can also be programmed to interrupt the processor after an extended period (seconds to weeks) including during deep-sleep mode. For more information about the clocks and time-keeping see "Timing Tests" below.

The mote board and sensors operating voltage is nominally 3.3VDC. Primary power can be supplied from a wide ranging input of 4.7-28VDC and is bucked down to the required level. Mote and sensor operation typically consumes a high amount of power relative to other sensor network applications: ~0.3W. Major sources of power consumption are: GPS radio (up to 100mA non-continuous), Xbee radio (47mA), sensors (<20mA), mote board (<8mA) and DC-DC step-down efficiency (~80-90%). Motes are typically provided a nominal 12VDC during field operations. We purposely chose to 'overkill' the power system using a 12V, 5Watt PV and 7AmpHour battery. Even during the middle of winter in Salt Lake City with several successive days of overcast this source was more than adequate to sustain continuous operations. A '3.3VDC'-battery connector is provided so that a small AA pack can be installed under the mote board. This is used for radio range tests and/or short term observations lasting up to a day or two.

Vital status information is monitored and reported along with sensor data. In particular this includes power parameters: primary battery voltage and current, operating voltage (Vcc) and sensor current. Mote software monitors and reports these values at a selectable rate. If the battery voltage drops below a definable level, the sensors, local storage and radio are shut down. The PIC then goes into background monitoring mode awaiting the battery to recharge before resuming data collection. Other status information includes performance parameters generated by the Xbee radio: missed syncs, successful transmits, signal strength, etc.

In addition to the primary Xbee communications port, an RS232 interface provides an alternative data destination path and operator console. The console port can also be used for long-range radio operations when interfaced to a Digi 9Xtend 900MHz modem. Operational parameters

and rates can all be read or reset by user command. These include data sampling rates, power monitoring, sensor scanning, local storage, GPS sync, Xbee reset, and Xbee status monitoring rates. Xbee operating parameters needed for DSM base communications can be read or changed. The GPS, sensors and/or file system can be turned on or off manually. Vital parameters are stored in EEPROM. These include mote identifiers, all rates, Xbee parameters, and base-line frequencies for the TCXO. LEDs are used as a visual clue for on-site maintenance. These include a 1-second clock heart-beat and sample-telemetry in progress indications. Other blinking patterns indicate reboot, console switching, gps message acquisition/lock and DSM messages received.

C. Software.

1) Development Environment.

The intelligent sensors and the mote boards are both based on Microchip Technology processors. The sensors use a low-power 8-bit processor whereas and the mote boards use a more highly integrated 16-bit processor. Code is written in the standard 'C' language, with occasional use of assembly for low-level access. Microchip's MPLAB Integrated Development Environment (available for free download) is used for compiling, linking, programming and debugging the software.

2) Basic Approach.

Sensor and mote software are interrupt driven 'state-machines'. This applies to I/O functions as well as for scheduled events. To save energy the processor is put to sleep when not performing these activities. The mote can be operated in two modes: self-timed based on 1-second intervals and Xbee-sleep based (see Testing for more information.) In self-timed mode either the TCXO or the internal RTCC is used to provide the triggering. When a 1-second event occurs, a 4-byte date/time value is incremented as well as a series of soft programmable counter rates, setting their associated flags when appropriate. In Xbee-sleep mode the 1-second event still updates the date/time value, however data sampling is initiated when the radio reaches a network synchronized 'wake-cycle'. The main processing loop becomes active after interrupt handling completes, testing flags and taking appropriate actions: sample sensors, monitor power, scan for new sensors, process operator-commands, turn-on GPS, check Xbee status, etc. The main software loop uses other timers to prevent lockup or discontinue activities that do not complete. For example if the GPS fails to lock in its assigned window, it is turned off and restarted later. A watchdog timer is similarly enabled to provide a fail-safe if for example I2C lockup should occur. When complete messages are ready to send to the destination port, they are transmitted as fast as possible. Message delivery happens within 30ms at 38400bps. Other I/O interrupt handlers are used to test whether sensor data packets have been ingested before the next one is polled, whether command or GPS messages have been received, or an ADC conversion has completed etc.

3) Sensor Types.

Sensor types are defined by their unique I2C addresses. Each has a specific message format. This approach is similar to many commercial I2C devices. It allows a mote to quickly upload and pack sensor data for delivery to the DSM and local store without excessive handling. Although more are possible (extended 10bit I2C addressing), we declared 120 I2C sensors addresses. Some are for status, internal declarations or specific commercial devices. Most are segmented in blocks of four with each set assigned to one of the specific Wisard devices supported: soil, radiometric, met

and generic sensors. Having four ids each allows multiple sensors of the same type to be attached. When a sensor's assigned I2C address is polled, it averages, calibrates and sends scaled integer engineering units to the mote. This response takes 1-20ms depending upon the sensor. A sensor's I2C address byte precedes its data in an output message. This type identifier tells the DSM or host computing facility how to unpack and interpret its data. Every sensor also has a unique serial number distinct from its type. A mote periodically scans all possible I2C addresses to determine which sensors are attached. If a sensor at a specific address is there, it responds with its serial number. These serial number messages become part of the archived data record, automatically documenting sensor locations and changes.

4) Wisard Message Format.

Data messages sent to the DSM/local-store have a predefined structure. We use a binary format with no additional encryption to help conserve radio energy. Three message types have been defined:

IDxxx	':'	Version#	MsgType	Contents as defined by 'Version#'	CRC	EOM

Sensor Serial Number Messages

NodeID	':'	Version#	0x00	SensorTypeIDs with S/N's for each	CRC	EOM

Data Messages

NodeID	':'	Version#	0x01	Sequence#	SensorTypeIDs with data	CRC	EOM

Comment Messages

NodeID	':'	Version#	0x02	Printable Character String	CRC	EOM

All messages are stored in the data archive but only data messages are processed by a DSM. Comment messages are sent by a mote to notify when certain conditions occur or for reference information. These include startup messages (mote serial number, software version, clock source, bootup cause), local storage file cycling, low-battery-shutdown, etc. GPS satellite lock messages are also comments, documenting site location and updates to the mote's local clock.

Each message begins with 'NodeID', which is a printable ASCII string representing the mote's individual identifier: ex "ID17". Similar to the sensors, a mote's identifier is distinct from its serial number and its id can be easily reassigned and stored via operator command. The DSM NIDAS (NCAR In-situ Data System) software currently tags these IDs in its own xml configuration principally to help assign names in derived products and display titles, however there is no hard and fast reason this is required. It is possible for a base to receive two or more distinct message streams from different motes having the same ID#. The Version# is an important feature. It indicates which lookup-table a DSM or host should use to associate sensor types with data. An entirely different set of 120 addresses and formats can be associated between version numbers. We currently use '1', however certain sensor types are being discontinued and new sensors are being added with '2.' Data messages also include a sequence number value from 0-255. These help identify if message loss occurs and possibly help sorting operations. A simple 1-byte CRC is added to test for bit errors. No message length is sent, instead a 3-byte end-of-message marks its termination. We use ETX (0x03), EOT (0x04) and CR (0x0D). The 3 character sequence greatly diminishes the probability of having false EOM triggers occurring within the binary data stream.

Motes can also send messages in two other forms. The first is 'ASCII-printable.' This is intended for attaching a mote to a 'dumb-terminal.' It consists of space-delimited decoded sensor

names, addresses and their scaled integers. The second is 'wisard-printable'. This is the same decoded ASCII printable string but it is packaged as a wisard comment message. It is used as a maintenance tool for operators watching streaming DSM mote messages.

5) Operator Interactions.

Two-way communications allow mote sensing parameters to be dynamically adjusted and saved remotely. All operating parameters are stored in EEPROM and are retrieved upon boot-up. Serial commands received through the Xbee or console ports can be used to read or reset these values. This allows motes and Xbee-radios to be easily reassigned and relocated without the need to edit and reprogram their software. Commands are simple ASCII strings terminated with a CR. For convenience Xbee base radios broadcast commands to all motes on its assigned channel and network. An individual mote can be addressed using its id number prefixed to the command. For example to check all mote data rates would be "dr<cr>". Each would respond with a comment message "IDxx=5". To reset mote ID17's data rate to report every 10s would be "#17dr=10<cr>". To boost its radio power would be "#17xb=pl4", etc.

6) Time-Keeping.

One important sensor type is a data sample's local time-tag.' When a mote initiates a polling cycle, it attaches its 4-byte time-tag as sensor type 0x0B. At the other end, all messages received by the DSM are tagged with its own heading including a 4-byte Unix date/time stamp. Although redundant to the mote time-stamps, uncertainty about radio message delivery can reveal significant differences.

IV. Field Performance

A. Laboratory testing at NCAR:

Initial tests involved (1) validating sensor quality, (2) mote functionality, (3) I2C communications, and (4) clock performance. After testing indoors we moved Wisard outside. This allowed a complete test of (5) radio communications including the ability to handle multi-hopping.

1) Sensor Quality.

To ensure research quality measurements each sensor was individually calibrated. When possible the sensors were calibrated in our laboratory. The soil moisture and thermal properties probes were tested in a homogeneous mixture of sand with water for comparison with gravimetric standards. Other sensors were calibrated by the manufacturer. As mentioned above each sensor's processor board was calibrated to eliminate errors due to temperature.

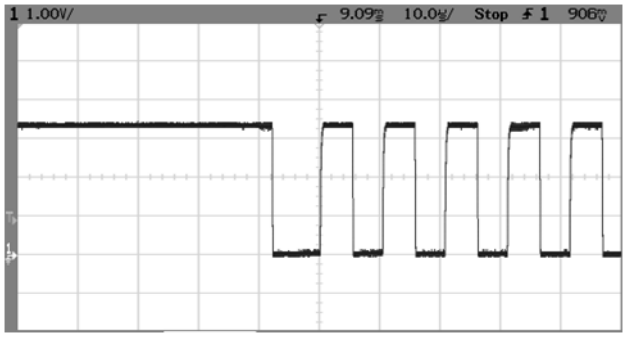
2) Mote functionality.

Most hardware and software validation tests were straightforward. Local storage and communications were confirmed. Power traces were tested up to 1A. The resettable protection was checked at 400mA. Switchable power components were tested to their specifications, well above design requirements: 150mA (GPS), 200mA (Sensors), 500mA (3.3 supply). GPS and radio operations were validated and confirmed for non-interference from PIC operations, oscillators, and the 330kHz switching DC-DC regulator noise. TVS surge protectors for power lines were not tested

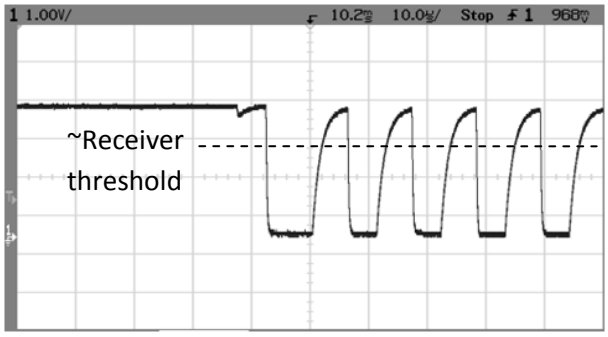
but components have been in other applications. Temperature chamber profiles were run over several days from -30 to +50 to complete the basic tests.

3) I2C Sensor Communications.

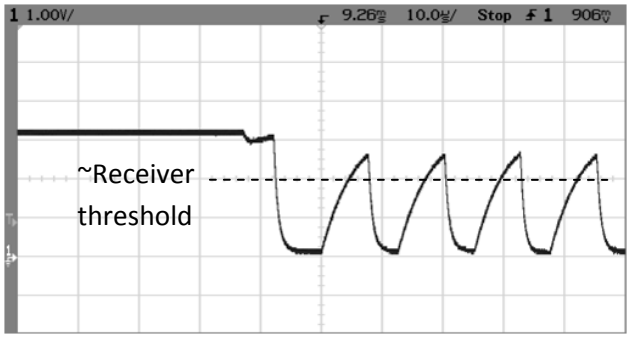
Validating I2C communications was more involved. Our use of the I2C bus is somewhat non-standard. Rather than having a 'long buss' with short segments tied to clients, our sensors are tethered in essentially a star topology. The original design provided both ESD (Electro-Static Discharge) component protection plus inductive ferrites to filter spurious noise. These further complicated the signal environment. Spurious capacitance (ex. ESD TVS's, long/stranded cables, etc) on high-speed communication lines deteriorate signal transitions. In severe cases they introduce bit errors or failure of signaling to reach proper thresholds needed for reception. Poorly matched impedances and 'star' configurations can introduce reflections/ghosts and spiking/ringing causing further signal degradation. These problems worsen with higher speed signaling (I2C 100kHz vs 400kHz). Various stress tests were performed with long and dissimilar cable lengths, multiple sensors, different bus terminating resistors/capacitors, etc. Electrical signal integrity and digital performances were monitored and compared, Figure 3. From these tests we chose to remove the ferrites, adjust the ESD components and maintain bit rates at 100 kHz. We found that for the cables we chose, lengths could extend to 50 feet without the use of buffer-drivers.



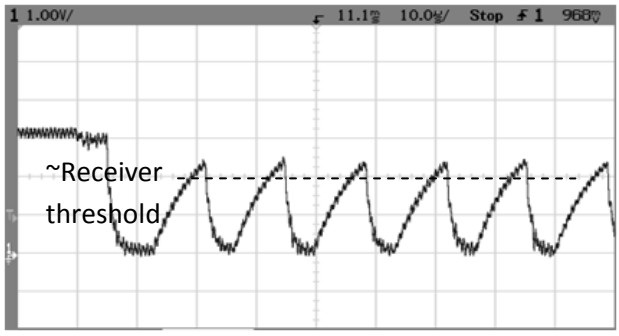
Ideal, non-loaded I2C Clock Signal



Bussed I2C Clock Signal at end of 15'



I2C Clock: 3-Sensors 25' cables, 1-Sensor 15', 1-stub



I2C Clock: Improper termination, 100' cable: Bad

Figure 3: I2C Signal Quality

4) Clock Performance / Time-Keeping.

An important aspect of the Wisard development was establishing a local 'real-time' clock. Having an accurate time tag when data is collected is critical. This is especially true when correlating data from different sources. Wireless communication latency complicates these issues.

Initial testing focused on a modified approach of the Bresenham Algorithm for generating a good internal timer from imperfect clocks. (See http://www.romanblack.com/one_sec.htm) Temperature chamber tests quickly revealed that the internal PIC oscillator was inadequate to provide good time keeping. Microchip provides application notes (AN1155) on how to use a temperature sensor and lookup tables to calibrate its RTCC to within ±3s/month. However, we did not implement the RTCC on our prototype boards. Instead we decided to try an external TCXO having ±2.5ppm stability over temperature. This was done in part to evaluate whether sub 1-second sampling could be supported. Testing of the TXCO revealed good performance using the Bresenham method, in most cases to easily within 0.5s/month over temperature cycling. We introduced an 'adjust-clock-drift' command value to help achieve this performance. Thus, an operator or DSM software can slightly adjust the center frequency and gradually realign any drift experienced during extended exposure to narrow temperature ranges. We discovered that although 0.1 and 0.01 second time slicing was feasible, we could not reliably support data rates below 1.0s. This was in part due to the large number of processor interrupts needed to achieve good results. At 1-second timing, the processor interrupts still needed to be about 70/s and much greater for faster rates. Rather than select another TCXO with an order of magnitude better stability to avoid calibration and adjustment issues, we chose to put that expense toward the GPS and RTCC instead for our production boards. The RTCC method also saved some energy and reduced timing interrupts to 1/s to help improve mote response. We did not run extensive RTCC temperature chamber tests to confirm Microchip specifications, but instead focused on outdoor testing to confirm radio telemetry.

5) Radio Communications.

Six motes with sensors and two 'Xbee-repeaters' were deployed in a mesh configuration outdoors around our laboratory building, Figure 4. At least two hopping paths were established. Some motes were within line-of-sight of the DSM base. Others required up to 2 hops to reach the base. Several operated at the ~90m range limit for Xbee's. These tests were begun in winter and operated until late summer before leaving for the PCAPS experiment. Initial tests were based on the prototype boards but later tests involved version-2, production units.

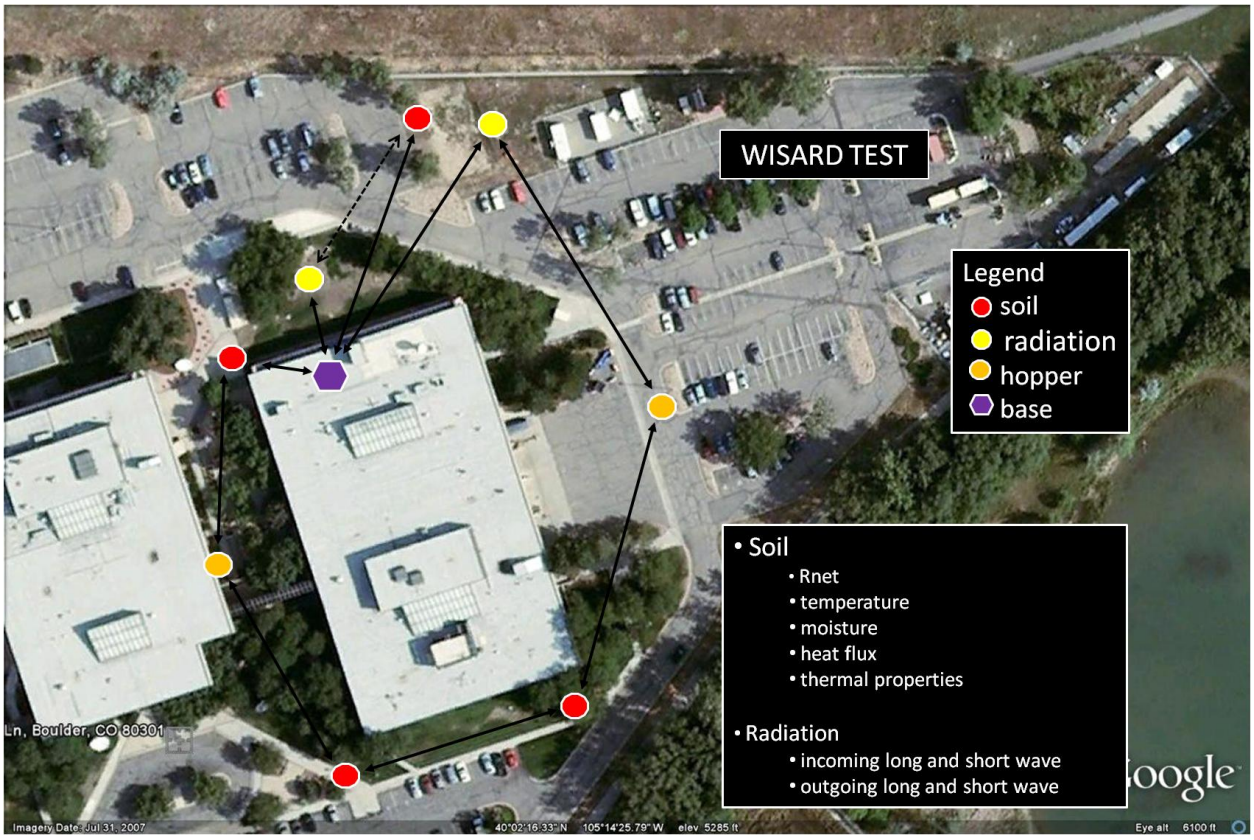


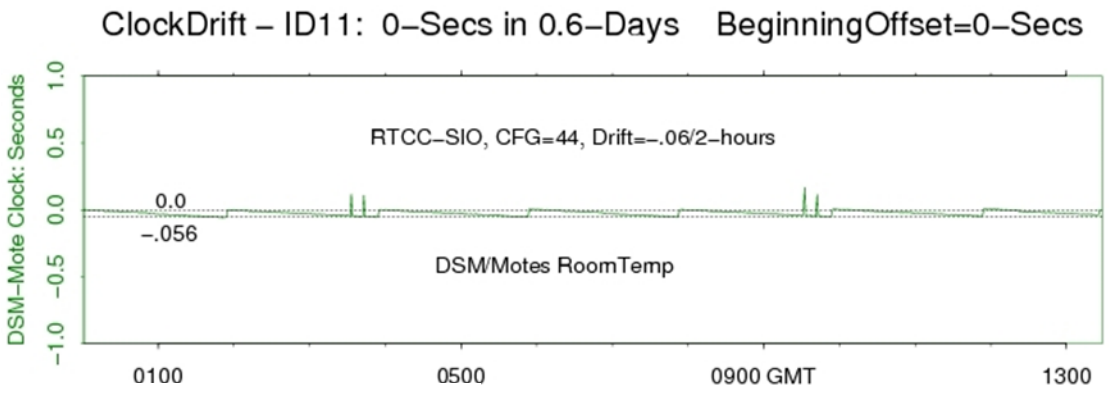
Figure 4

The radio environment around the laboratory contributed to the test challenge. There are many Wifi routers and clients active in the 2.4Ghz band (U.S. 2.412-2.462Ghz, channels 1-11) that the Xbee modules operate in (ISM 2.402-2.483Ghz). BlueTooth modules also operate in this band. The ISM licensing requires radios to employ spread-spectrum telemetry. Xbee DigiMesh modules use a Direct Sequence Spread Spectrum (DSSS) method as opposed to frequency hopping. DSSS involves a unique binary 'chip-code' coordinated at transmitting and receiving ends. A transmitter generates pseudo-random noise over an entire spectral band and sends the broadened signal. A receiver de-spreads the spectrum using the same code and mathematically correlates to generate signal/noise values to determine probability of having gotten coherent information. Interfering signals introduce more noise but normally correlations remain high.

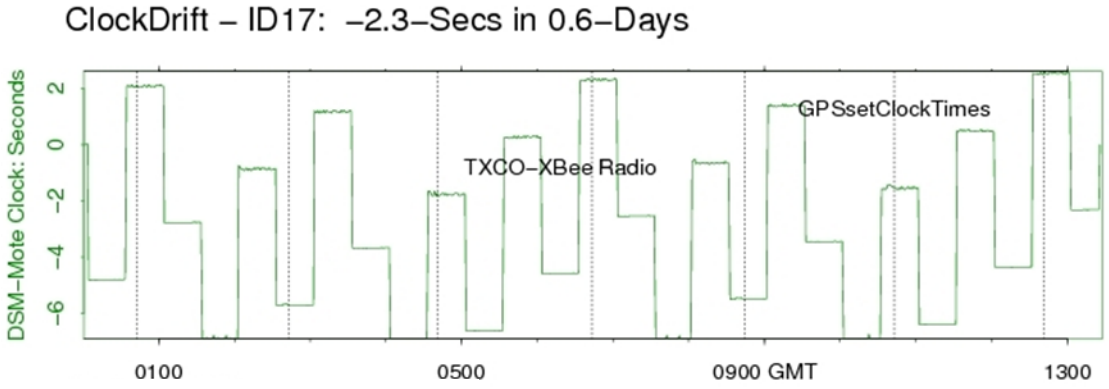
During this 7 to 9 month deployment we were able to test two modes of operation with the Xbee radios; continuous awake mode and synchronized sleep mode. Data Rates were attempted at 1, 2, 5, and 10s. The continuous-awake mode tests were successful with sampling rates as fast as 2s. One-second sampling was possible but overall responsiveness and reliability decreased. Mesh hopping, including re-organized path routing was demonstrated by power cycling individual motes. In some cases this also involved low-battery shutdowns. Two-way communications were demonstrated as well. In order to command one or more motes it was necessary to reduce the data-rate during interactions to improve readability and responsiveness. Occasionally some radios lost synchronization and did not re-establish it for extended periods. We also had periods where two-way command and control was inadequate. The solution we used was to add a heart-beat command in the motes. If not received within a definable period, the mote physically reset its Xbee. We also periodically reset the DSM Xbee base. These adjustments resulted in reliable operation.

The Xbee sleep mode proved much less acceptable. There were long periods of drop-outs and attempts to resynchronize were not always successful. More problematic was the inability to establish good 2-way communications. Base operator commands rarely worked. These problems were in part due to our short sample periods and using an early version of the Xbee firmware. We were never able to adjust Xbee parameters to ensure long-term success.

Station time-keeping was monitored during outdoor testing. The RTCC drift rates were high using default calibration parameters: ~1s/day. PIC processors do include a software resettable RTCC calibration adjustment to compensate for differences between individual oscillators. It can also be used the same way as with the 'adjust-clock-drift' method noted above (see Microchip's tech. note). We did not take full advantage of this feature. Instead we set mote GPS time synchronization to occur every 12 hours. A comparison was made between interfacing a mote to our DSM over a serial link versus through the Xbee radio to observe latency issues. Timing 'errors' could be substantial with radio messaging. The DSM time-of-arrival tag versus mote clock differences were often up to several seconds. They also jumped or changed drift rate from periodic self-resynchronization of the radios and forced Xbee reset times. GPS synchronization of the serially connected mote, however, combined with an adjusted RTCC calibration setting maintained its DSM time-difference to within 0.05s. Figure 5 shows and example of the difference between the serially connected mote and a mote reporting over radio link.



Serial Link: GPS sync 2-hours GPSrtcTest/WithSensors/power.2011Apr1/Stn1/Stn1.ID11.txt



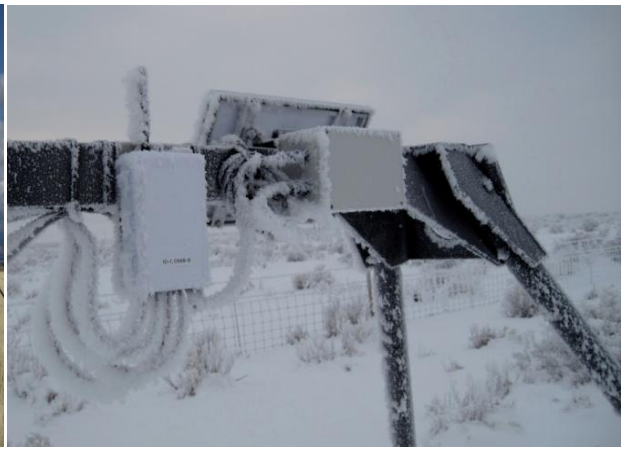
Radio Link: GPS sync 2-hours GPSrtcTest/WithSensors/power.2011Apr1/Stn1/Stn1.ID17.txt

Figure 5: Difference between DSM message arrival time and reported mote value.

B. PCAPS Field Experiment

In the winter of 2011 Wisard was deployed at 7 sites in and around Salt Lake City for the Persistent Cold-Air Pool Study project, PCAPS. Each site had a suite of soil probes and radiation sensors. Two sites had a second set of soil measurements. The soil probes were installed approximately 1 month before the study began to allow for proper coupling with the environment. The instrumentation operated successfully for over 3 months even during harsh weather conditions.

Site-6: 2 Soil Installations, 1 Radiometer



Site-1: Data Collection/Telemetry continued to work

Site Setup: Mote waited for Battery to charge

Soil sensors require equalization time

There were problems but none significantly jeopardized overall success. Several motes had episodes when their data were missing in the DSM archive. Many of these outages were individual sensor problems causing I2C bus polling to fail. In these cases, the motes were continually getting reset by the watchdog timer. The biggest problem was with soil temperature probes. The original potting compound used was faulty and wet conditions electrically shorted the signals. Two motes had their I2C circuits damaged and had to be replaced. Sensor connector corrosion caused some outages until all were sealed with conductive grease. On some occasions motes shut themselves down overnight or during extended overcast when their batteries drained below the test limit. Some outages were recoverable using the mote local data storage files during post project processing. These were usually from a DSM or radio communications outage. Lost radio communications created gaps but were normally re-established by the periodic Xbee reset in motes or the DSM base receiver. A couple of radios failed from moisture. Approximately 2 station-weeks of data were recovered using this local data storage.

Some mote timing problems were experienced. The primary archive is stamped with the DSM's time-of-message-arrival. Mote RTCC clock drifts became excessive in some conditions. The GPS clock reset should have been more frequent than every 12 hours. There were a few cases where the GPS jumper positions were set incorrectly resulting in data being incorrectly time tagged. By default, mote software boots up with date/time set to 'zero.' Until GPS synchronization occurs, the values increment properly but are incorrect. There were a few occasions when a GPS reported a good satellite lock but the date/time were bogus. Any data messages having non-deterministic times could not be recovered.

During the deployment mote firmware was updated in all units. One change added a test for GPS date/time sanity, as well as a timer to turn it off if proper lock is not achieved. EEPROM handling was added to allow configuration changes to become permanent. Operator command handling was adjusted to improve Xbee parameter setups and to manually switch between the Xbee and RS232 ports.

Overall mote data message recovery was above 95% despite problems outlined above.

V. Future/Ongoing Efforts:

From the lessons learned during PCAPs and evolving requirements we are continuing to make refinements to Wisard. We have a 3rd version mote board based on a more capable PIC processor having significantly expanded program and data memory space, more I/O ports and a USB interface. Increased data memory allows message caching: 1-s sample rates might be maintained with reporting reduced to 30 s. New base station/DSM software needed to support caching remains to be written. The expanded memory combined with USB interface may be used to ingest camera images. To save power, migrating to Xbee sleep mode operation remains a priority. Recent tests with the newest Xbee firmware from Digi have been encouraging. Expanded I/O ports and program space on the version3 design allow it to be used as an intelligent base/repeater in addition to normal sensor sampling. Intelligent command handling is needed to provide reliable two-way communications during Xbee sleep mode operation. The new PIC may also expand upon the distributed processing concept. Sensor measurements may be averaged in the mote to reduce message rates and save radio power. Additional time-keeping tests need to be performed with the Bresenham method and dynamic temperature compensated RTCC calibration adjustments. At this time we do not see any need for major modifications to the sensor interface design.

Acknowledgements

The National Center for Atmospheric Research is funded by the National Science Foundation. Mention of products in this paper does not imply official endorsement by UCAR/NCAR.

References

[1] F.V.Brock and P.K.Govind, “Portable Automated Mesonet in Operation,” J.Appl.Meteor., 16, pp. 299-310, 1977

[2] J. Hill and D. Culler, “Mica: a wireless platform for deeply embedded networks,” *IEEE Micro*, vol. 22, no. 6, pp. 12–24, November/December 2002

[3] Lynette Laffea and Han, Monson, Williams, Sun, Semmer, Militzer, Oncley, Burns – “Comprehensive Monitoring of Carbon Sequestration Processes in Sub-Alpine Forest using Wireless Sensor Arrays”, Sensys Conference Paper 2006

[4] Bhatti, Carlson, Dai, Deng, Rose, Sheth, Shucker, Gruenwald, Torgerson, Han, “MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms,” Mobile Networks & Applications Journal, August 2005.