Roving Wireless Sensor Network

ECE 561 Project Report, May 2009

Steven Turner and Mark Woolston
Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado, USA

***Abstract*—Mobile wireless sensor networks show great potential in a number of important disciplines such as search-and-rescue, military reconnaissance, environmental monitoring, and extraterrestrial exploration. This project focuses on the embedded system design acting as a test bed for experimentation.**

Keywords-component; networks; sensors; wireless; autonomous; robots; distributed; rover; energy harvesting;

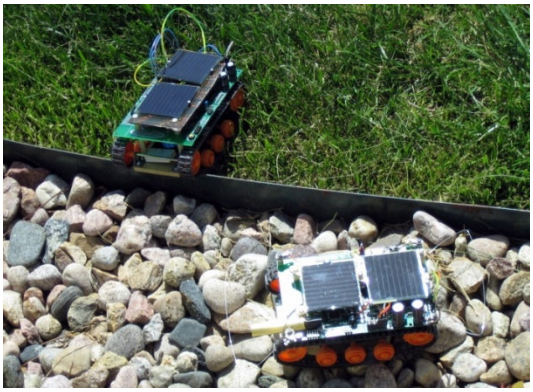

Figure 1: Rovers Under Test

I. OVERVIEW AND MOTIVATION

Our very society is in essence a network of distributed capabilities and distributed intelligence. Although some may clearly have more impact than others, no one individual is responsible for the success or failure of our species as a whole. The tasks of keeping our society running are distributed throughout this ad-hoc network, and even significant disasters can eventually be overcome, or at least worked around. In varying degrees and scales this theme is recurrent throughout both our human society and the natural world, from our global food network to the running of university department, from a colony of ants to a pride of lions, from a flock of birds to coral reef to the very cells making up our bodies. The distribution of tasks among a multitude of similar yet specialized, semi-autonomous entities is perhaps the system most proven to be successful. We are only just beginning to successfully emulate such systems by explicit design, though not for lack of trying.

Although the preceding paragraph may seem out of place in the context of a paper concerning sensor networks, it is not. We, as ordinary individuals, are nodes of perhaps the largest distributed sensor network on the planet. (It is not necessarily surprising then, that as individual nodes, we are generally not aware of what the ultimate purpose may be.) In contrast, while the vast majority of our engineered hardware systems to date have been comprised of specialized parts or subsystems, there was little or no ability for one subsystem to adapt to the task of another that was no longer able to perform. Extremely high stakes tasks such as the Voyager space probes tended to be engineered for the utmost reliability, with perhaps some redundancy. With that route came very high costs and long development cycles. They were essentially single, very high uptime sensor nodes for a multi-master-single-slave peer-to-peer wireless network. On a day to day basis, any discrepancies required a cessation of activities until new commands were received from a command center – one of the master nodes.

Communication between these master node located on Earth and the sensor node located on Mars can also have an enormous propagation delay which greatly reduces system efficiency. For example the closest distance from the Earth to Mars in the last 60,000 years was 55.758*10^9 meters in 2003. If the RF signals travel at the speed of light (3*10^8 meters / second) then the one way signal propagation delay is 185 seconds. If this communication can be minimized and collaboration between multiple nodes located on the Mars surface can replace some of this communication then system efficiency and necessary system throughput will be greatly increased.

NASA's reduced-cost Mars rovers however, although still individually complicated, took a slightly more distributed approach. At the highest level, one could consider the pair to be two identical sensor nodes of a multi-path, multi-hop tree network where each rover could communicate with one of at least three Mars orbiting satellites. The satellites serve as routing nodes to the Earth-based command centers, which can be considered coordinator nodes. A large advantage, however, was the addition of software that provided the rovers with a limited amount of autonomy. They no longer had to wait for a command from Earth to achieve their essential goals – valid scientific data acquisition – or to find a way around the small rock in their path any particular day. Even their mechanical systems contained sensor networks, such as their six wheels with rotation and motor current sensors that have resulted in a drive system with exceptional fault tolerance.

In light of the first paragraph, it is the view of this author that NASA has actually taken a small step towards a design plan that, far from being unknown, has already been proven by nature. The use of two identical, mobile rovers as mobile

sensor nodes greatly increased the scientific return while at the same time lowering total costs and the risks of total failure. We endeavor to create a low-cost platform capable of taking an additional step, by further reducing mobile sensor node (rover) complexity, increasing the number of nodes and utilizing the increased routing possibilities to incorporate modern sensor network techniques.

II. Project Goals

This project involves the design and implementation of a very low-cost, mobile sensor network that can serve as an experimental platform for several areas of interest. Examples include swarm behavior, distributed tasking, self-localization, surveillance, exploration, machine intelligence, biomimicry, energy harvesting, and multipath routing in a dynamic environment. At the highest networking level, the basic platforms will extend upon the simple sensor networks of the Mars rovers as indicated in Table I. Like the Mars rovers, our sensor nodes are intended to be mobile and to harvest the energy of the sun, allowing the potential for years of functionality before battery replacement is needed. Unlike the Mars rovers, these mobile nodes are ultimately intended, with additional software development, to work together to accomplish a task, and to utilize more advanced wireless networking to coordinate behavior.

Design Aspect	Sensor Network		
	Pre-rover Lander	Mars Rovers	This Project
Network Topology	Point-to-point	Tree multipath	Mesh multipath
Node Count	1	2	5 (3 mobile)
Maximum Hops	1	2	Arbitrary
Reliability Requirements	Extremely high	High	Low
Node Cost	Extremely high	Very high	$200

Table 1: Design Exploration

It is readily apparent that a significant challenge when designing a system is where to draw the distinction between networks, subnets, nodes, software and hardware partitioning, and components. Component cost and development time played a significant role in determining the tradeoffs made during the design of this system. Due primarily to time and budgetary constraints, the major focus of the project was to build proof-of-concept hardware and implement a minimal subset of networking features.

Our initial project goals were:

- Three sensor nodes plus a coordinator node
- At least one node is mobile
- 2.4 GHz, IEEE 802.15.4 wireless communication with a single coordinator node on a PC
- Battery pack also rechargeable via USB connection
- USB connectivity for configuration and testing
- Solar energy harvesting
- Forward-facing light sensor
- Ambient temperature sensor
- Battery voltage and temperature monitoring
- Navigation by wireless remote commands
- Small size
- Cost per unit between $150 and $200
- Ability to handle varied terrain
- Obstacle detection

We felt that available time was the greatest threat to success, rather than an inability to figure out any particular task, with cost as a secondary issue since we were self-funded. Prior experience with hardware design played a role in tipping the tradeoff point between hardware and software slightly more towards hardware, as hardware design actually progressed at a faster pace.

III. System Architecture

The rovers were designed to provide low-cost educational and research hardware for experimenting with many concepts. Three 8-bit Atmel AVR microcontrollers, networked together on one board, provide the opportunity for experimenting with single-master-multiple-slave, multi-master, and routed-message serial communication models. The motor subsystem and behavioral system controllers are both connected to an on-board USB-to-serial converter and can be accessed independently through a terminal application by the use of addressed commands. Two controllers communicate through their own dedicated UART interface. A 2-wire I^2C serial interface bus connects all three of the microcontrollers with each other and with a serial EEPROM. Dual H-bridge motor drivers allow PWM control of two DC motors, track motion sensors provide feedback for speed control design, a battery charger IC provides the ability to control the charging of several battery chemistries, and a MOSFET switch array allows control of a pair of solar panels for energy scavenging. Light and temperature sensors serve as sensors for basic sensor network applications, and digital inputs for corner contact or proximity sensors are provided. The UART asynchronous serial and I^2C synchronous serial busses are externally available, allowing for additional functionality. Potentially, GPS, accelerometers, a flux-gate compass, or a camera could be added via the serial busses.

A 2.4 GHz IEEE 802.15.4 wireless link is incorporated, ultimately providing the capacity to link multiple mobile nodes together with a wireless stack such as Zigbee. Finally, since the platform is low cost and self mobile, all manner of network topologies can be experimented with in a dynamic physical setting by utilizing multiple units. This allows experiments to be performed that exercise concepts such as mesh networking, network self-discovery, robust routing, self-healing networks, sensor networking, distributed processing, and swarm behavior, all in the real world.

True to the concept of distributed network architecture, the rover's functions have been distributed among the three Atmel microcontrollers. An ATmega324P microcontroller is tasked with serving as the "brains" of the rover, coordinating behavior and reading environmental sensors. Communications and networking are assigned to an ATmega1281 microcontroller, and an ATmega168 functions as a peripheral controller, monitoring and controlling vital system functions (Figure 2).

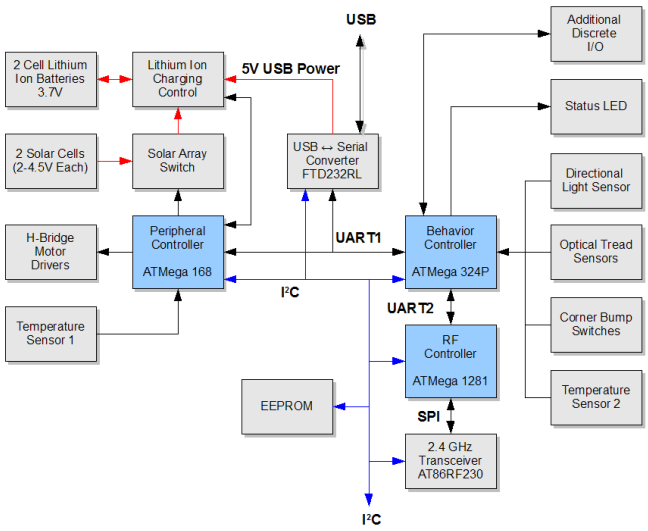

Figure 2: System Overview

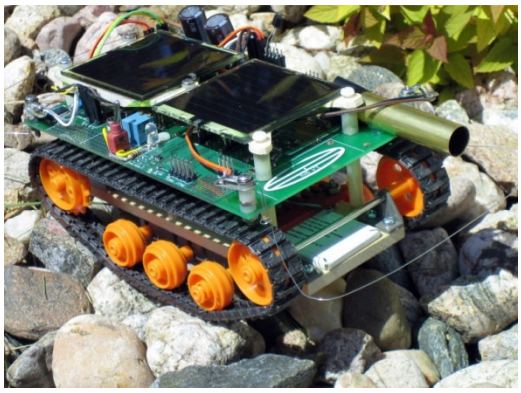

Figure 3: Rover Platform

IV. HARDWARE

A. *Mechanical Design*

In order to keep costs down, the mobile chassis was based upon a Tamiya TAM70108 tracked vehicle chassis and TAM70097 dual-motor gearbox, which are available for about $24. The gearbox uses two Mabuchi FA-130 DC electric motors which operate from 1.5 - 3 VDC, with a 1.5V no load output of 9100RPM at 200mA, a stall torque of 26g-cm (0.36oz-in) at 2.2A, and a maximum efficiency output of 6990RPM. They are specified to deliver 6g-cm (0.08oz-in) torque at 660mA current draw.

A 4" by 6.25" custom designed circuit board is mounted antenna-forward to the wooden tracked vehicle platform with nylon standoffs. The batteries are mounted to the platform itself, and the solar panels on a lightweight hand-fabricated support structure. Four bumper switches made from music wire are attached to the PCB corners. A cadmium sulfide photoresistive light sensor is mounted in a tube for directionality, and the tube to the circuit board in a forward facing manner.

The two completed units weigh 13 ounces each, are less than 8" long, less than 4.5" wide, and 3.5" high. The ground clearance is slightly less than ½". Photographs of a rover unit are shown in Figures 3 & 4.

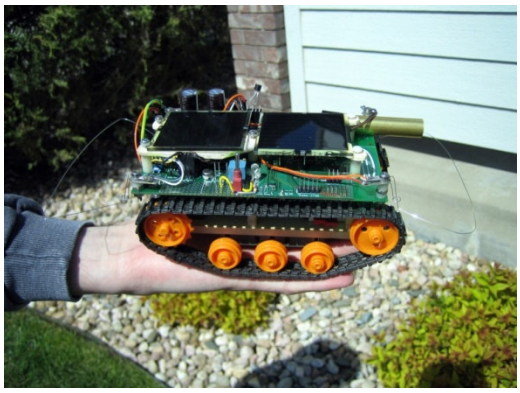

Figure 4: Rover Physical Size

B. *Circuit Design*

Since no low-cost existing platform could be found which allows exercising the full range of concepts we wished to make available, we designed our own electronics and circuit board layout. Although the great amounts of computing power in various 32-bit microprocessor families was enticing, both budgetary and energy consumption concerns led us to mid-range 8 and 16-bit microcontrollers. Atmel AVR 8-bit microcontrollers were ultimately selected, primarily due to the greater availability of free, unrestricted C language development environments and libraries, with Texas Instruments MSP430 series and Freescale S12X series losing out due to the excessively restrictive limitations on C code development placed upon the free versions of their development environments. For the Atmel AVR devices, cost, performance, and feature set were at least equivalent, and prior experience with the AVR microcontrollers also played a role. This is discussed further in the Software section.

The microcontrollers ultimately chosen were an ATmega168 for the peripheral functions, an ATmega324 for the behavioral and outside-world sensor functions, based upon its combination of A/D converters, PWM modules, small size, decent memory resources, dual UARTs, SPI, availability, compatibility with tools, and relatively low cost. An ATmega1281 is provided for networking and radio control. Programming and debugging are provided through 6-pin ISP/debugWire connectors and 10-pin JTAG connectors, depending on the processor.

Although the rovers are intended to be wireless, it was felt necessary to include a means of communication and fast battery charging during the development period. Since most portable computers no longer offer traditional serial ports, USB seemed the logical choice. FTDI offers the FT232R USB-to-serial-UART converter, which implements a virtual serial port over the USB. Royalty-free drivers are available for most popular PC platforms. This provided a means of having the AVR microcontrollers' UARTs communicate with a PC via USB with little more complexity than a traditional asynchronous serial port. The UART bus was connected to both the Peripheral Controller and the Behavioral Controller in a single-master multiple-slave manner. In order to prevent

conflicts, each microcontroller was addressed individually, and that they each enable their transmitters only for the duration of their own transmissions, a task which the Atmel AVRs are easily able to do.

Power for the rovers is provided by a pair of paralleled, 3.7V, 670mA Li-Ion cells. Lithium-ion cells were chosen for their high energy density, light weight, and comparatively low cost. A Linear Technologies LTC4062 Li-Ion battery charger IC was chosen due to its relative ease of use, low supporting parts count, and availability of spice simulation models. The circuitry has been designed based upon manufacturer application examples. Charge rate, charge time, and low-voltage threshold detection are all adjustable with trimmer potentiometers. As such, with proper adjustment, the charger is expected to be capable of charging any lithium-based secondary battery with a charge-voltage limit of 4.1-4.2V. The charger input voltage is provided by the diode-OR'ing of the USB 5V and the solar array output voltage.

Energy harvesting can be performed with two 2.375" square, 4.5V copper indium diselenide solar panels capable of delivering 20-50mA in direct sunlight. The solar panels are be connected to a low-R_{on} MOSFET array so that they can be dynamically switched in series or parallel by the peripheral microcontroller for greatest efficiency, depending on light conditions and system load. This is non-trivial, as it needs to route currents at float voltages that may be either lesser or greater than the readily available gate drive voltages. A 1-Farad supercapacitor accumulates the charge to allow the battery charger IC to run in a discontinuous mode in the event of marginal solar energy. In addition, a low-Ron MOSFET has been placed across the output Schottky diode that leads to the battery charger, allowing the microcontroller to remove the 300-400 mV diode drop for improved efficiency. These circuits are wholly original work, and were designed and simulated in ltSpice. Total array output voltage and the voltage of one panel alone are resistor divided by 11:1 and brought to the peripheral microcontroller's A/D inputs.

The two DC motors are driven in pulse-width-modulation mode by a pair of discrete ultra-low on-resistance MOSFET H-bridges also completely of our own design. The MOSFET's are driven in turn by the PWM outputs of the peripheral microcontroller. The software and hardware tasks of driving the H-bridge have been partitioned for reduced software loading through the use of demultiplexers, allowing the software to specify merely the PWM speed signal and a Boolean direction signal for each motor.

A comparison of surplus solar cells and Li-Ion batteries was made, and sufficient quantity of the winners were purchased from BGmicro for $2.79 and $0.99 each, respectively. Significant effort was put into identifying MOSFETs that represented the best "bang for the buck" for the H-bridges and solar array control, with targets of sub 3V operation, extremely low Ron, small size, high current capability (as appropriate), availability, and pricing significantly less than $2 each. as The chosen MOSFETs include NXP's PSMN2R5-30YL and Fairchild's FDS4465 for the motor driver output stage, as well as Fairchild's FDC6420C and Vishay's Si1988DH for the solar array switch network.

For better control of the motors and for use in dead-reckoning, high-sensitivity reflective infrared emitter and detector pairs have been mounted above the rover treads on each side. The emitters are driven in a pulsed manner to save battery power.

The microcontrollers draw their power from the battery through a simple RC filter designed to remove the worst of the H-bridge switching noise. A voltage reference is necessary because we are powering the IC's directly from the battery, yet still need to monitor the battery voltage. Texas Instruments' REF3320AIDBZT 2.048V reference IC was chosen for its simplicity and phenomenally low 3.9uA current draw. 2.048V was chosen for the A/D references voltage so that one bit would represent an integer number of millivolts. With a 10-bit converter, 2.048V/1024bits = 2mV/bit. Most of the monitored voltages come through 11:1 resistor dividers, so they are represented by (2.048V/1024bits)*11 = 22mV/bit. An MCP9700 temperature sensor IC was selected due to its low power, low cost, ease of use, and high accuracy. With this IC, the A/D converter formula becomes 0.2^{o}C/bit – 50^{o}C. Two temperature sensor inputs are on board, one for monitoring the battery temperature, and the other for ambient environmental temperature. The directional photoresistive light sensor is read by an A/D channel on the behavioral microcontroller rather than the peripheral controller, as light-seeking is one intended behavior for the rovers.

The 2.4GHz IEEE 802.15.4 radio circuitry is based upon several Atmel sources, including the RZRaven Zigbee evaluation kit and Atmel's 2.4GHz PCB antenna application notes, which also included PCB layout examples from which we based our antenna pattern. Special attention has been paid to the critical transmission line, antenna, and ground plane placement dimensions, as well as providing ample power supply bypassing capacitors for all IC's.

C. Printed Circuit Board Layout

Throughout the circuit board layout process, careful attention was paid to maintaining as complete as possible a ground plane on the back side of the board. Traces were kept short and wide whenever currents were involved, and large planar regions utilized as much as possible. The AT86RF230 radio IC and the LTC4062 Li-Ion battery charger IC were only available in leadless packages, requiring special attention to layout for those tiny and power hungry parts. The schematic and printed circuit board layout were captured with the free version of the Eagle PCB layout package. Advanced Circuits was selected as the PCB manufacturer of choice due to prior positive experience with them.

D. Hardware Design Issues

A nuisance issue was discovered with the light (free) version of Eagle layout software – it only allowed us to make boards smaller or equal to 3.2 inches by 4 inches, and we needed 6.25 by 4 inches to facilitate optimal antenna and tread sensor placement on the tracked vehicle chassis. In order to work around this, portions of the circuit were laid out separately and the free GerbMerge utility used to join the multiple Gerber layout files together. This unfortunately made it much harder to ensure accuracy between the circuit board and schematics.

E. *Design Reuse*

In order to facilitate progress, not all of the rover design was done from scratch. The USB circuit was reused from one of the author's senior design projects, which in turn was based upon a manufacturer application note. The AT86RF230 radio IC and ATmega1281 RF microcontroller designs were borrowed from Atmel's RZRaven evaluation boards and related application notes as much as possible. The Li-Ion battery charger circuit was derived from a Linear Technologies application note. The 2.4GHz antenna was obtained from an Atmel application note. Manufacturers' data sheets provided a great deal of information, including the voltage reference circuit. And of course the tracked vehicle chassis and dual motor gearboxes are purchased in kit form.

F. *Original Work*

Much original work is present. The overall system architecture is our own contrivance, with the exception of the ATmega1281 and AT86RF230 pairing. The motor driver circuitry, solar array circuitry, sensor circuitry, tread sensors, and the majority of "glue" circuitry is wholly our own. With the exception of some circuitry in the high-frequency area of the AT86RF230 IC, the circuit board layout is completely our own.

V. SOFTWARE

Most of the system logic is performed in software. As shown in the hardware section, the system consists of separate peripheral, behavioral, and radio controllers. The software was written in the C language, and compiled using the open source WinAVR software tools.

Many commands have been implemented in order to control system peripherals, read sensors, and check status information.

A. *Peripheral Controller*

The peripheral controller is an Atmel ATMega 168 8-bit microcontroller running at 8MHz. It has functions to check the status information, display the system uptime, read a temperature sensor, read a specified analog to digital channel, disable the battery charger, control the solar array switch, control the motors, and read the specified memory address. The system boot up message shows these commands.

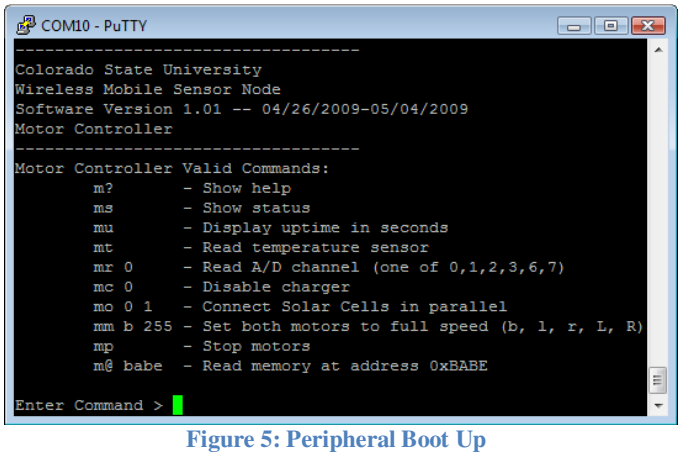

Figure 5: Peripheral Boot Up

The status message contains the information listed below using the command "ms". All commands to the peripheral controller are prefixed with the "m" address character. The "s" character is an abbreviation for status. This command shows that the battery charger is currently enabled, the battery charge rate is set to C/5, and the battery is currently charging. The battery voltage is shown to be 3.85 volts. The charge voltage is 4.356 volts.

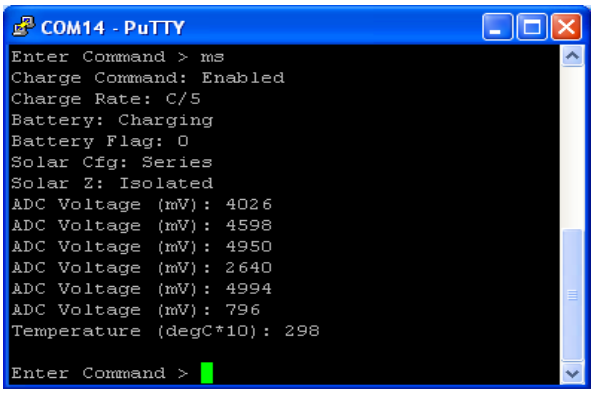

Figure 6: Peripheral Status

The solar cells from above are configured in the series configuration. The third ADC voltage shows the combined voltage from the solar cells is 4.246 volts with a single cell outputting 1.936 volts. The solar array can be switched into a parallel configuration as shown below. The analog readings then show the voltage output from the parallel connected cells. The example below shows the cells voltages to be 2.224 and 2.222 volts. This is a difference of only 1 bit.

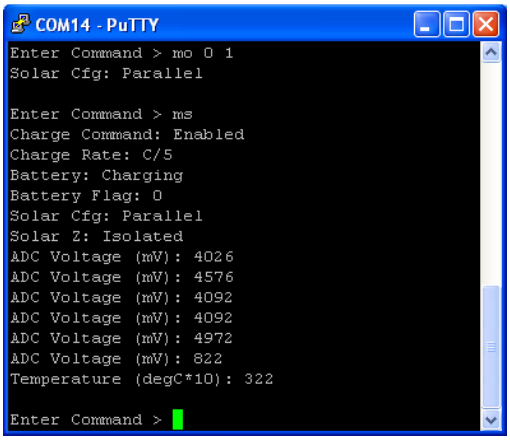

Figure 7: Solar Parallel Configuration

Motor control is achieved by using a pulse width modulated signal generated from a timer module. The H-Bridge design has a direction input for each motor to switch directions. This approach was chosen in order to make the software development easier while requiring only one timer module for both motors. Each module has two independent output compare register. This means that each motor speed can be set independently. The motor speed has a PWM resolution of 8 bits since this is the size of the timer module counter.

Several different motor commands have been implemented, as shown below. The speed value is specified as a positive or

negative number between 0 and 255. A negative number represents movement in the opposite direction. The "b" character moves both motors at the specified speed.

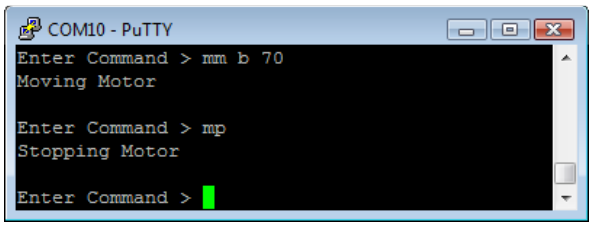

Figure 8: Both motors commanded forward at 27% speed

The capital letters "L" or "R" specify a pivot command, causing the motors to move in the opposite directions at the specified speed.

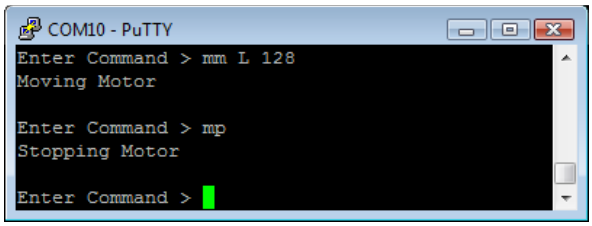

Figure 9: Motors Pivot Left

The lower case "r" and "l" commands move only one motor at the specified speed.

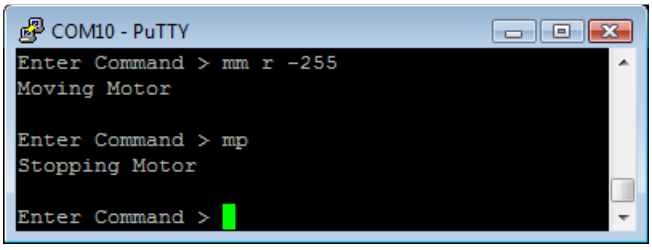

Figure 10: Right tread backwards at full speed

B. *Behavioral Controller*

The radio communications software stack is processor intensive. Our system also contains many peripherals requiring more I/O capabilities than provided on one processor. This led us to implement a separate processor for the main application algorithms. An ATMega324P microcontroller was chosen to run the main application code. The processor was chosen for the reasons listed below.

- Has the required two UART modules
- Uses Atmel's PicoPower technology, minimizing energy consumption.
- Contains what we consider an ample amount of program memory while allowing debugging via inexpensive debugger hardware.

This processor is running from an 8MHz external crystal. One of the timer modules is clocked externally by a 32.768 KHz crystal in order to provide a system real time clock used for the task scheduler.

The behavioral controller has diagnostic functions similar to the peripheral controller. The boot up message for this controller is shown below.

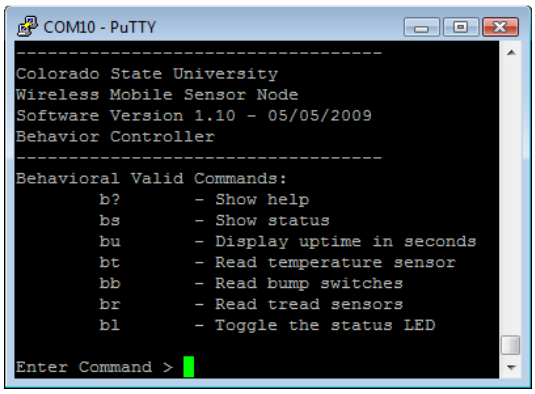

Figure 11: Behavioral Boot Up

Functions included on this controller show status information, display the system uptime in seconds, read another temperature sensor, read all four corner bump switches, read optical tread sensors, and toggle a status LED. These commands use curly braces as message framing characters.

The system uptime in seconds is obtained from the scheduler as shown in the following section.

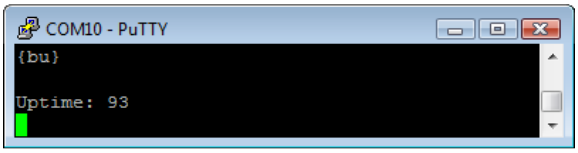

Figure 12: System Uptime

All four corner bump switches are read by this controller. These are implemented as described in the hardware section. These are simple digital signals with a logic low level indicating the switch has been triggered.

The function to read the bump switches is called in the scheduler with a frequency of 128 Hz.

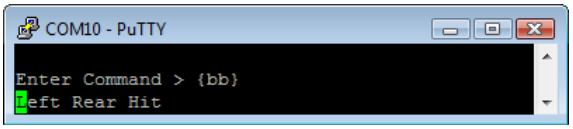

Figure 13: Bump Switch Command

The tread sensors are described in the hardware section. A task is scheduled to read these sensors at 128 Hz. This function turns on the LEDs, compares the sensor output with the previous value, increments the odometer counter is this value is different from the last read, and then turns off the LEDs to conserve power.

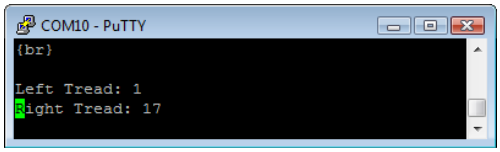

Figure 14: Tread Sensor Command

C. *Radio Controller*

The radio controller implements the radio communications stack, and communicates with the 2.4 GHz transceiver IC over a synchronous serial peripheral interface.

The software for this controller has not been implemented yet due to time constraints. The current plan is to implement either an IEEE 802.15.4 link, or a ZigBee stack called "BitCloud" provided by Atmel.

This controller will interact with the behavioral controller through a dedicated UART link.

D. *Communication*

Both the peripheral and behavioral controller UART modules are connected to the same bus. Care had to be taken to ensure data was not corrupted because of simultaneous transmissions. Each command is prefixed with a one byte character address. The appropriate controller recognizes its address and executes the requested command. The controller transmitter is enabled to send a reply message, and then disabled again which effectively disconnects the transmitter from the bus. This UART connection tunnels the data through a USB to serial converter chip. This enables our board to be plugged into a computer USB port which is mapped as a virtual serial port.

There is a separate UART communication bus between the behavioral and radio controllers. This is used to control the radio module, and to communicate over the 2.4 GHz radio link. ATmega1281 radio controller communicates with the Atmel AT86RF230 2.4 GHz transceiver using a synchronous serial peripheral interface.

All three controllers are connected to a common 2-wire I^2C synchronous serial interface. I^2C uses an addressing and arbitration scheme that can be used to allow all devices to communicate with each other.

E. *Task Scheduler*

A simple best-effort static task scheduler has been implemented in our project. This uses the controller timer module running on an external 32.768 KHz crystal. The module is configured so that an output compare match interrupt is triggered at a frequency of 128 Hz. A 32 bit current time variable is incremented at each interrupt.

Tasks are scheduled by registering them with the scheduler. A register task function is used. The parameters to this function are a function pointer, task period, and an initial offset. These tasks are stored in an array of structures. The time interval period is defined to be integer increments of 1/128 seconds. The period would be set to 128 to define a task run every second.

At every time interval an execute task function is run to determine is any of the scheduled tasks should be run. A loop checks every scheduled task to see if the current time minus the last run time is greater than the period. This determines if the task should be run again.

F. *Software Issues*

Several software issues have yet to be resolved due to time constraints. The hope was to be able to use ZigBee stack software provided by Atmel. This proved to be more difficult than expected, and has yet to be completed. The software is complex and pre-compiled for specific development platforms. Not all of the source code is available to the end user, but is provided in a pre-compiled API format.

Inter-processor communications through the 2-wire serial interface have also not been completed at this time. This is again due to time constraints.

VI. Fault Tolerance

This system was not designed specifically with fault tolerance in mind. There are however many design features that can be used to provide minimal functionality in the presence of a fault, and the systems were designed to minimize fault propagation.

The peripheral controller alone handles items such as motor control, battery charging control, and solar panel array switching. This allows offloading many of the lower level drivers, but if this processor stopped functioning none of these features would be available. However, the behavioral controller and wireless communications would be unaffected until battery power failed.

The behavioral controller provides higher level application logic. Fault tolerance could be built around failure of this processor by detection of the failure by one of the other processors. If this processor was no longer responsive then either the peripheral or radio controller could be programmed to provide some kind of limited functionality. As long as the behavioral microcontroller's failure does not jam on it's transmit pin, the other microcontrollers could still communicate.

The radio processor is the only possible path for the radio communications. If this processor became unresponsive then these communications would no longer be available. The mobile node, however, could continue to perform the requested tasks and gather data in an isolated manner. In addition, both the I^2C and UART serial busses connect the radio microcontroller with the behavioral microcontroller, potentially allowing communication between the two to remain functional even if one interface fails.

VII. Testing

Two mobile sensor node "rovers" have been built. Testing has been performed as much as possible given that some software functionality is not yet implemented. The Li-Ion battery charger circuit, solar array circuit, motor drivers, temperature sensors, peripheral and behavioral microcontrollers are all functioning well. In the USB circuit however, it was necessary to make a small wire fix to ensure the UART outputs of the USB chip did not exceed the lower supply voltages of the microcontroller. A problem with excessive sensitivity to ambient light has also turned up in the tread sensors which will need to be resolved. It may become necessary to switch to modulated sensors to filter out the effects of ambient light levels.

The solar array has proven to work quite well in bright sunlight, and does in fact charge the battery. It appears that overvoltage protection for the energy storage supercapacitor may actually be necessary, as the solar panels can put out up to 9V unloaded, as is the case when the battery is fully charged and the charger IC stops drawing current.

Mobility on several surfaces has been tested as well. The rovers move quite well on smooth surfaces such as concrete, and also over small obstacles less than ½" in height. Across larger obstacles, however, the rovers sometimes "high-centered" on the centrally positioned free axles. Attempts to move across 4" high grass have also shown that the center of gravity of the rovers may be too high, as they would occasionally tip over.

Testing has also shown that tread slip can be significant issue in rough terrain. This will require closing a control loop on the motor drive command and tread motion sensors to ensure travel in the correct direction. It will also complicate the use of dead-reckoning for even simple relative positioning; putting higher demands on RF based localization.

Finally, under aggressive and sustained pivoting actions on rough concrete, the thermoplastic treads have occasionally slipped off the wheels. This may be correctable through improved tensioning, or may require a different tread design altogether.

VIII. Future Goals

Time constraints prohibited us from accomplishing everything planned with this project. Future plans include implementing the radio link, automated behavior programming, and providing better inter-processor communications. We also anticipate exploring at least some of the following areas with these rovers:

- Communication with neighbor sensor nodes
- Multi-hop networking by explicit routing tables
- Configuration via wireless
- Light seeking behavior on an individual basis
- A subset of the ZigBee stack
- Mesh networking
- Route self-discovery
- Escaping from corners and other obstacles
- Signal strength and quality reporting
- Limited localization and positioning
- Distance traveled measurement
- Network self-configuration
- Semi-automatic expansion of group covered area to roughly optimal density
- Light seeking behavior on a group basis
- Collision avoidance
- Dynamic repositioning based on signal quality or "loss" of a node, such that no orphan nodes remain
- ZigBee mesh networking
- Ultrasonic localization and positioning
- Adding a flux-gate compass
- Ultrasonic obstacle detection and ranging
- Mounting a camera
- Distributed sensor networking
- Swarm behavior experiments

A full ZigBee stack is the current plan for providing the radio link. This would be used in order to gather sensor node data, and to provide for collaboration and swarm behavior between multiple mobile nodes. This link would also be used to provide for sensor localization.

The automated behavior programming could include many possibilities. A simple starting point would be a light seeking algorithm. This could include many aspects of the project such as reading light sensor data, optical tread sensor odometer readings, solar charging algorithm, reading corner bump switches, and RF communication of sensor data to a base station. Investigating and mapping a room using swarm behavior is another future goal for automated behavior programming.

IX. Conclusion

This was an ambitious project to undertake in a single semester course. Many areas of study were integrated into this project such as circuit design, printed circuit board layout and assembly, hardware and software partitioning, software design including schedulers, and state machine logic. We hoped to accomplish more in this time frame, but are happy with the work that was completed.

We were also able to achieve our cost goals for the system with a per unit cost of $147.21 not including miscellaneous small components. The per unit estimated cost was $200. Since these prices were for low quantities we would expect large scale costs to be significantly less.