

A Wireless Interactive Teaching Solution

Keith A. Berkoben
ES100
Spring 2004

Table of Contents

I Motivation	4
II Concept	4
III Design Considerations and Ideal Specifications	4
.B System infrastructure	
.C Software Applications	
IV Focus and Approach	6
V Platform: Processor and Radio	6
.A Wireless Technology Selection	
.B Processing Platform	
.1 Mica2dot	
.2 TinyOS	
VI Hardware Prototypes	9
.A Alpha Prototype	
.1 Goals	
.2 Hardware	
.i LCD	
.ii Keypad	
.3 Firmware Development	
.i Keypad Driver	
.ii LCD Driver	
.4 Prototype Evaluation	
.B Beta Prototype	
.1 Goals	
.2 Hardware	
.i Power Supply	
.ii Power Switching	
.3 Firmware	
.4 Prototype Evaluation	
VII Wireless Data Transfer Protocol	15
.A Data Transfer Protocol v1	
.1 Goals	
.2 Implementation	
.i PC	
.ii Remote Node	
.a Data Integrity	
.b Perceived Latency Reduction	

.3 Evaluation	
.B Data Transfer Protocol v2	
.1 Goals	
.2 Implementation	
.i PC	
.ii Remote Node	
.3 Evaluation	
VIII Application Software	22
.A Student Device	
.B PC	
IX System Cost	25
X Conclusions and Suggested Future Work	26
.A Summary of achievement	
.B Future Work	
Sources Consulted	27
Appendix A: Project Cost	28
Appendix B: Production cost in Lots of 1000	30
Appendix C: Circuit Diagrams for Beta Prototype	Omitted from online copy
Appendix D: LCD and Printing Software Source Code	Omitted from online copy
Appendix E: Keypad Driver Source Code	Omitted from online copy
Appendix F: Student Node Application Source Code	Omitted from online copy
Appendix G: PC Application Source Code	Omitted from online copy

I Motivation

It is a commonly accepted view that students learn best when they are involved in the learning process – learning interactively. In an interactive learning environment students' participation is integral to the progression of the lesson, helping the students to be more attentive and reinforcing the connection between student and educator. Conversely, an interactive environment benefits the educator by making him or her aware of students' progress and any difficulties they might have with the material.

Unfortunately, it is very difficult to implement an interactive teaching environment in a large lecture class. Time and space constraints make it impossible for an educator to interact with each student personally, and a lack of existing infrastructure hinders the use of hard-wired technology to allow remote interaction between educators and their students. Furthermore, a large class of 30 or more would likely provide more data than a professor could process manually in real-time, necessitating an automated data acquisition solution to facilitate processing and visualization of data.

II Concept

In the hands-off environment of a large lecture, the primary mode of dynamic interaction between student and educator is question & answer. In this environment, an interactive teaching solution that provides an efficient means of including an entire class in a question & answer session and visualizing/logging the evoked data would be very useful. The system should be easy to use and efficient enough that a lecturer can easily utilize it multiple times during a single lecture. It should also allow students to provide visualizable/logable feedback to their lecturer in real-time.

III Design Considerations and ideal specifications

III.A Student device

The student input device must be capable of text output and numeric input. Graphics output is optional, but desirable. The device must be able to communicate with the rest of the system. Long battery life is essential, but not at the expense of excessive size or weight. Ideal specifications are as follows:

Display: Backlit LCD capable of displaying 160 characters

Keypad: 16 keys capable of repeating keys and simple multi-key Combinations

Program Memory: 100k*

Non-Volatile Storage: 500k*

Processing: 4Mhz*

Battery-life: 100 hours of continuous use with 20% LCD on-time

Size: Approximately the size of a graphing calculator 190x85x25mm

Weight: not to exceed 300g*

Communication: See III.b
COST \$75

*These numbers provide maximum flexibility, but may be scaled back for many applications

III.B System Infrastructure

The Interactive Teaching system must be easy to install and scale easily. Ideal specifications are as follows:

Communication: All nodes must be linked wirelessly. Wireless protocol should be capable of encryption and resistant to crosstalk from nearby systems. Bandwidth is determined by scalability.

Scalability: Wireless link and software must scale to 250 users.

PC link: Base station must plug directly into an existing port on Lecturer PC or Laptop. A single software release must be compatible with many operating systems. All data must display on Lecturer computer in real-time with logging capability.

III.B Software Applications

The completed system must include the following features:

- Multiple choice Question and answer
- A two-button 'faster' or 'slower' lecturer feedback interface (FSB)
- A real-time PC graphical interface that allows the lecturer to:
 - a)Control all aspects of the Q&A system
 - b)Visualize FSB responses
 - c)Visualize Q&A responses organized by user and by question
- Users must be able to log-in to the system with a username and password (This feature was implemented but is not explicitly discussed in this paper, as it was relatively straightforward.

System Overview

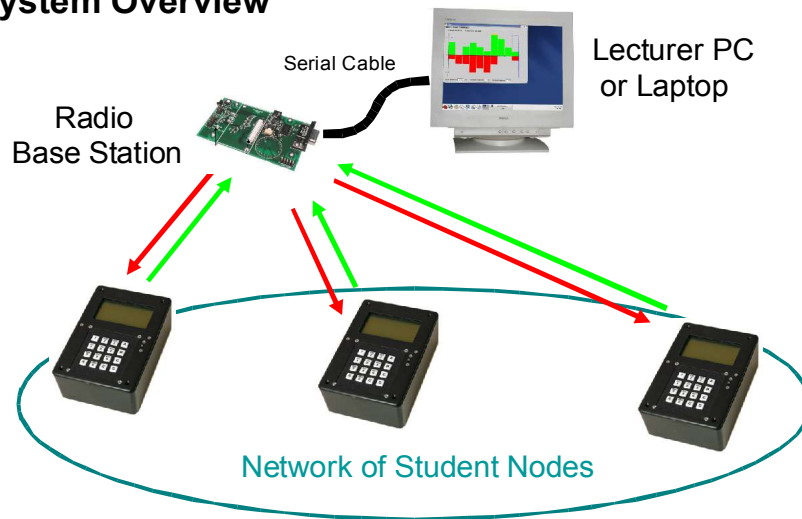


Fig. 1

IV Focus and Approach

The goal of this project was to produce a complete, functioning software infrastructure with compatible hardware to support the Interactive Teaching System (ITS) as described in II. It was considered essential that a demonstrable beta version of the system be completed for proof of concept by the end of the project no matter what simplifications were required in order to do so. Primary emphasis was placed on the software because robust protocols for data transfer, integrity, and visualization provide a stable system backbone, allowing the ITS to support a variety of hardware with minimal modifications.

Despite emphasis on the software backbone, it was necessary to build several prototype devices with which to test the software. These prototypes utilized the same hardware platform as the final hardware and integrate the same types of peripherals to the degree that they support all of the functions necessary for implementation of the system concept (II). It was essential that these devices were stable and reliable, but adherence to the ideal hardware specifications for size, weight, cost, and battery life as outlined in III.A was secondary to the rapid completion of hardware prototypes that could be used to refine the software backbone.

V. Platform: Processor and Radio

The platform for the student devices (nodes) and base station was selected to be as simple as possible while still fulfilling the requirements of section II and III such that system cost might be minimized. Because processing requirements were relatively

modest, the primary selection was made based upon the selection of wireless technology and ease of implementation.

V.A Wireless technology selection

Wireless technologies considered were infrared, 802.11b (wireless Ethernet), and lower bandwidth ISM radios:

	Infrared	802.11b	Low Bandwidth ISM
Common Uses	Television remote controls	Wireless Ethernet	Cordless Phones Garage Door Openers
Specific Advantages (not incl. cost)	Reliable transmitter can be built from scratch	High bandwidth, robust anti-collision and data integrity protocols, scalable	designed for implementation with simple devices, reliable
Major disadvantages (not incl. cost)	Requires line of sight transmission. Two-way communication difficult	hardware implementation difficult without significant external circuitry	Limited throughput, no built in data integrity or anti-collision protocols
Cost	< \$5	> \$40	\$7-10

Table 1 Wireless Technology Comparison

While 802.11b would have been the obvious choice for scalable wireless transmission, the hardware and protocols were far too complex for this application. Support for the protocol would add unnecessary hardware, size, power-consumption, and prohibitive cost. The Infrared solution is extremely simple and inexpensive to implement, and has already been implemented successfully in a system similar to the one under development here¹, but the requirement for line-of-sight transmission and the difficulty of developing a robust 2-way infrared protocol makes the technology undesirable for this application. Low Bandwidth ISM radio has the advantage of low-cost and simple hardware implementation, but it is necessary to build a custom radio protocol that is robust enough to scale to many users. Nonetheless, it is the most appropriate technology for this application.

V.B Processing Platform

V.B.1 Mica2dot

Once the decision to use Low bandwidth ISM wireless was made, Crossbow's mica2dot² (dot) was the clear choice for processing platform. The dot incorporates the following features on a single quarter-sized chip:

¹ Infrared solution implemented by EduCue: <http://www.educue.com/Home.htm>

² Crossbow products can be found at: www.xbow.com

Processor: Atmel Atmega128 @4mhz
Program Memory: 128K
RAM: 4k
Radio: 433mhz ISM
UART: 18.7k baud
I/O: 8-bit with 2 hardware interrupts and peripheral power supply (40mA)
Other: Support TinyOS

Using the dot simplifies the construction of the nodes and base station in two important ways. 1) The dot ships with an ISM radio built in, eliminating the need for custom design. 2) Dot support for TinyOS simplifies programming and provides building blocks for robust radio communication (see V.B.2). With 8-bit I/O and two hardware interrupts, peripherals can be connected directly to the dot without the use of any external Programmable Integrated Circuits (PICs). The dot is also optimized for low power consumption with an average draw of less than 20mA and the ability to enter a 16uA (@3v) sleep state³.

V.B.2 TinyOS

TinyOS is a single-threaded (with interrupts) open-source operating system designed at University of California Berkeley⁴. The operating system eases software development by abstracting the Atmel AVR microprocessor machine instructions to a higher level C-variant language, called nesC, which can be written and compiled for the dot on a PC.

The standard TinyOS distribution includes a number of software components that are helpful in this application:

TinySec - a hard-coded private-key based encryption algorithm, can be used to eliminate crosstalk between nearby systems and provides secure data transmission within a keyed system. The protocol uses a single shared key that is hard-coded into all nodes in a given system. It is transparent to application-level programs, and can be installed easily with minimal modification to radio stack code. Although this project will not implement *TinySec*, it is important to note that it can be quickly implemented using TinyOS online documentation⁵.

TOSBase - an application that turns a dot (or it's larger cousin, the mica2, which is used as the base station in this project) into a relay station that transmits radio messages to a PC through the MIB500 or MIB510 interface boards and vice versa. *TOSBase* allows all of the message handling to be implemented on the PC, where there is more programming and resource flexibility.

³ See mica2dot datasheet: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0043-04_A_MICA2DOT.pdf

⁴ For more information go to: www.tinyos.net

⁵ See the TinySec documentation here: <http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/tinysec.pdf>

Active Message protocol - implements a basic packet-based messaging system and implements CSMA⁶

Data integrity and packet handling protocols will be built on top of these existing components to fill out the complete data transfer requirements.

VI Hardware Prototypes

VI.A Alpha Prototype

VI.A.1 Goals

The aim of the alpha prototype was to demonstrate the integration of the keypad and LCD screen with the dot platform in a bench-model design, utilizing an external power supply. In this process it was also necessary to write firmware to support the peripherals.

VI.A.2 Hardware

Hardware selections were made primarily on the basis of ease of integration and versatility of function. In accordance with IV, cost was a secondary consideration.

VI.A.2.i LCD

Due to the limited I/O inputs of the dot platform, it was necessary to select a serially interfaced LCD module. An unpackaged LCD would require 11 I/O lines, which would require the implementation of a data bus and reassignment of some of the dot's dedicated pins. There are three major companies that produce ready-made serial LCDs:

Scott Edwards Electronics: www.seetron.com

Matrix Orbital: www.matrixorbital.com

CrystalFontz: www.crystalfontz.com

Of these three companies, Scott Edwards Electronics had, by far, the greatest variety of displays – including large graphic LCDs. The display that best suited the versatility requirement was the Scott Edwards G12864. It is capable of displaying 168 characters on a single screen, which is important in a device where significant amounts of text must be read. The display can also print graphics, which was a feature that this project was initially expected to utilize. It also accepts flexible power supplies (7v unregulated AC, 9v unregulated DC, and 5v regulated DC). Unfortunately, the LCD package also contains features we did not expect to use, including built-in non-volatile memory and customizable fonts. Cost for the device is higher than would be acceptable for a production model (\$170), and size is less than optimal, but was an ideal device for prototyping because of its versatility and ease-of-use.

⁶ Carrier Sense Multiple Access (CSMA) is a common anti-collision algorithm. See detail here: <http://webs.cs.berkeley.edu/tos/papers/mobicom.pdf>

In order to communicate with the LCD module, the dot's 0-3v UART had to be converted to a 0-5v RS-232 signal. This was done using a standard RS-232 transceiver manufactured by Maxim (MAX3232C). Capacitor values were selected for a 3v supply according to the Table 2 of the datasheet⁷.

VI.A.2.ii Keypad

Using a 16-key layout for the keypad, it was possible to connect a matrix unit directly to the dot's I/O pins. Because matrix keypads are extremely common devices, the main selection criteria were cost, simplicity, and availability. The classroom environment does not necessitate any special features such as water resistance, and backlighting is not necessary. The least expensive 16-key model stocked by DigiKey⁸ was the Grayhill Series 96 keypad. Its useful features include rear panel mount design, alphanumeric legend, and a 1,000,000 operation/button guarantee. The only external components required for interfacing the keypad with the dot are four 1N4001 switching diodes. Wiring was done according to the diagram in Atmel application note AVR240⁹

VI.A.3 Firmware Development

Firmware for the peripherals was designed to provide simple abstract interfaces that could be easily wired into higher-level software. Each is described below.

VI.A.3.i Keypad Driver

The keypad driver is an interrupt triggered polling scheme adapted from the one described in Atmel application note AVR240¹⁰. Program flow is as follows:

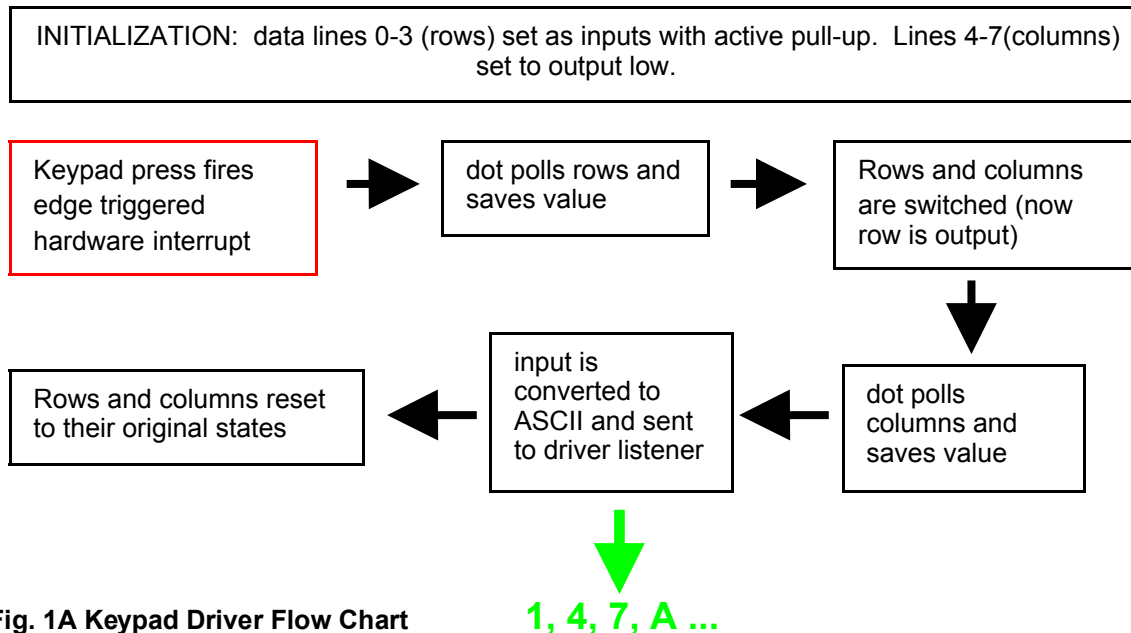


Fig. 1A Keypad Driver Flow Chart

⁷ See page 11 of datasheet: <http://pdfserv.maxim-ic.com/en/ds/MAX3222E-MAX3246E.pdf>

⁸ DigiKey is a major electronic component distributor. See www.digikey.com

⁹ See application note here: http://www.atmel.com/dyn/resources/prod_documents/doc1232.pdf

¹⁰ See footnote 9

The keypad can also be enabled/disabled by high-level software through the driver interface.

VI.A.3.ii LCD Driver

The LCD driver is a split-phase¹¹ module that abstracts TinyOS's *GenericComm* byte-level UART control module, allowing user applications to pass pointers to full lines of text for printing. The module assumes that no single text line will contain enough long-execution-time characters (like 'clear screen') that the 64-character buffer of the LCD screen will be overflowed, but always checks that the buffer is empty (by sending 'buffer probe' character to LCD) before signaling completion of a send to user application. Program Flow is as follows:

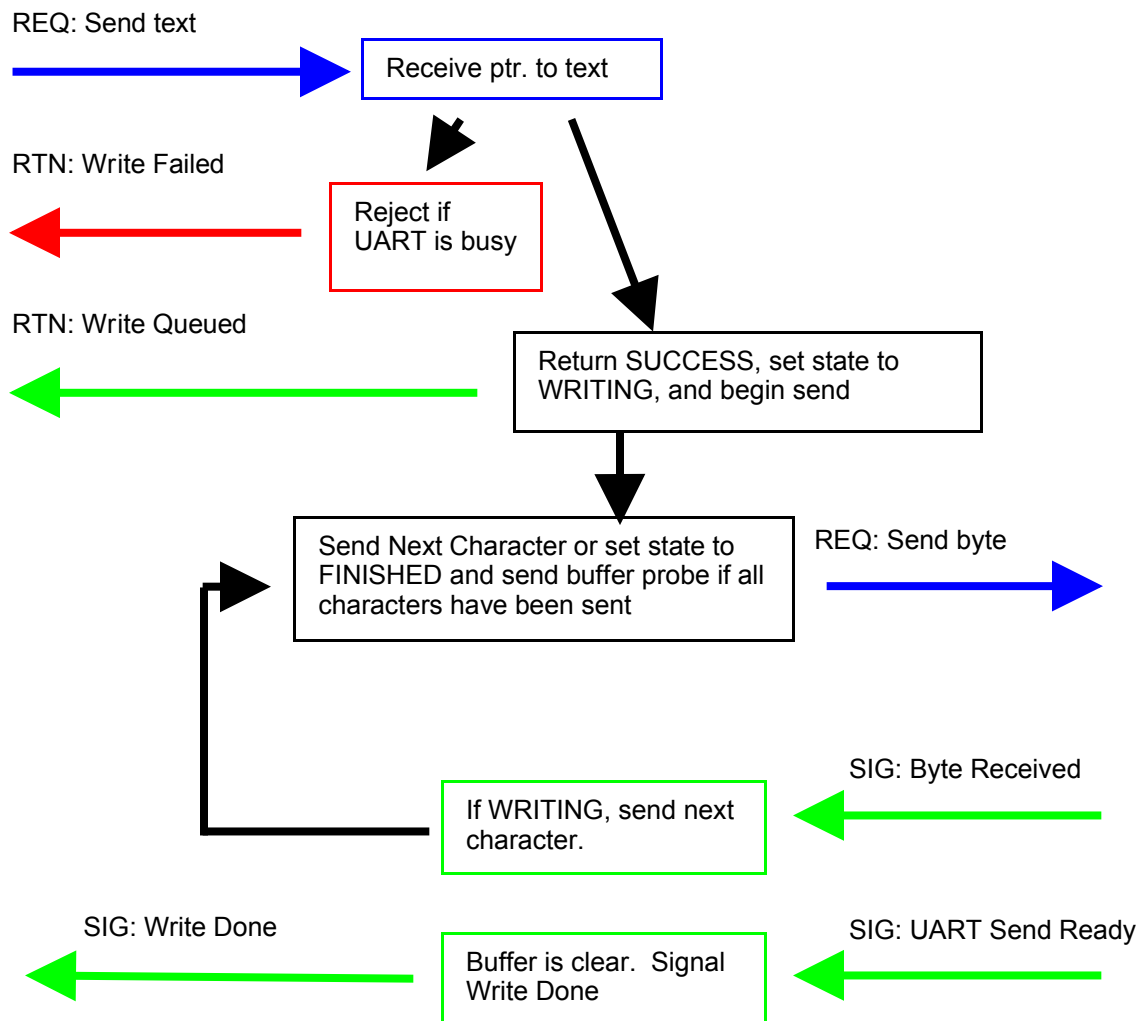


Fig. 1B LCD Driver Flow Chart

¹¹ Split-phase refers to a technique used in TinyOS's single-threaded architecture where a program initiates a hardware process and returns immediately, allowing the computation thread to perform other tasks until the triggered process finishes and signals that it may be continued in the computation thread. This scheme minimizes wasted processor time.

Layered on top of this driver is a module to handle simultaneous print requests from multiple higher-level modules and handle paging of data. To see the code for this module, see WriteAllM.nc in appendix D.

V.A.4 Prototype Evaluation

Once debugged, the alpha prototype was very successful. All of the components worked reliably and were easy to interface with higher-level software. In the process of building the prototype, it was discovered that the pinout on the dot, and corresponding MDA500 prototyping board, was incorrect with respect to the pin definitions in the TinyOS distribution¹². Specifically, the pin designated as INTO in the mica2dot's hardware.h file is labeled as INT1 in the dot manual¹³ and on the MDA500¹⁴. Due to the success of alpha, all of its components and drivers were carried over into beta.

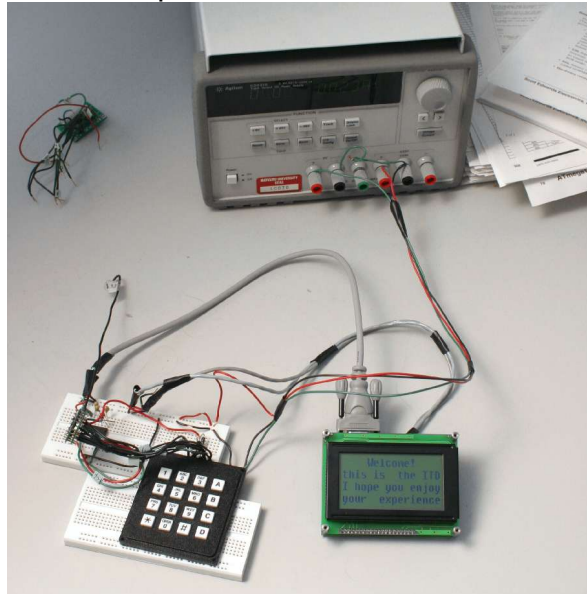


Fig. 2 Alpha prototype

VI.B Beta Prototype

VI.B.1 Goals

The aim of the beta prototype was threefold. 1) Design a power supply that allows beta to run off of a battery. 2) Add hardware that allows the dot to shut down peripherals when they are not in use. 3) Package device in a housing that will allow it to be portable. Due to the short duration of the project, it was necessary to avoid waiting for multiple component orders and PCB prototypes to arrive. As a result, immediate availability of components in through-hole packages was a major consideration. By

¹² Hardware file is <TOSDIR>/tos/platform/mica2dot/hardware.h in the distribution

¹³ See hardware users manual page 19: http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_User_Manual_7430-0021-05_A.pdf

¹⁴ See image of MDA500 on its datasheet here: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0047-01_B_MTS.pdf

limiting selection to available through-hole components it was possible to breadboard and hand-build the necessary circuits. This decision facilitated rapid, inexpensive debugging.

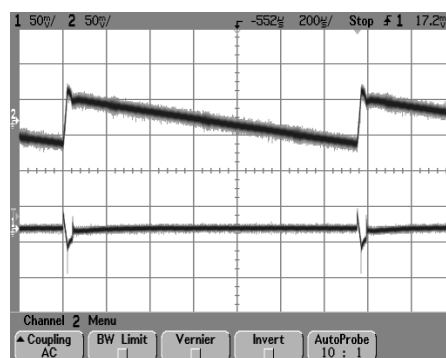
VI.B.2.i Hardware: Power Supply

In order for the nodes to run from a battery, it was necessary to create a dual supply within the device to power the LCD and the dot respectively. The former accepts 7V unregulated AC, 9V unregulated DC, or 5V regulated DC. The latter requires a regulated DC supply between 3.0 and 3.3V.

The initial design powered the device from a 3V cell, using the Linear Technology LT1073 to step-up the voltage to 5V for powering the LCD. With this design, it is possible to switch off the LT1073 with the peripherals, allowing a savings of $48\mu\text{A}$ ¹⁵ @3v during sleep mode. Unfortunately the peak current draw of the required inductor was large enough to add a 150mV ripple to the 3v supply, which was enough to disrupt radio transmission. Although it may have been possible to correct the ripple using a large (~300uF) low ESR capacitor to ground on both the low and high voltage lines, this design was scrapped.

The second design used the same LT1073 as a step-down DC-DC converter to power the dot. Although this design incurs a $48\mu\text{A}$ current penalty in sleep mode, it yields a few advantages. First, it provides a stable power supply to the dot with a battery supply anywhere from 3V to 36V. This allows complete versatility in peripheral selection, and because step-down mode produces much lower peak loads on the battery, the high-voltage supply remains stable enough to power peripherals without additional voltage regulators. Second, step-down mode ensures that peripherals will fail before the dot's power supply is compromised. This is essential because the dot's behavior becomes erratic as its power supply weakens, often causing its state machine to enter undefined states. If peripherals fail first, the user will know to change the battery before the dot's supply fails.

The power supply implemented in this prototype used the second design with a 9V battery supply. The following scope traces were obtained from the PCB version of the beta prototype with LCD on:



¹⁵ Half of the value given at 1.5v in LT1073 Datasheet <http://www.linear.com/pdf/1073fa.pdf>

Fig. 2A Scope traces from PCB power supply. Upper trace is 3V, lower is 9V. Vertical scale is 50mv, horizontal is 200us. Note perturbations in 9V supply when the LT1073 charges the 3V supply.

From the scope traces, voltage ripple seems to be approximately 50mv the high voltage rail, and 70mv on the low voltage. A ripple this large is not ideal, but it does not noticeably interfere with radio communication. A circuit diagram of the beta prototype can be found in appendix C.

VI.B.2.ii Power Switching

Because the LCD draws a large amount of current, it is important to turn it off when not in use. It is also useful to shut down the serial transceiver. To accomplish this, the dot's PWM1B pin is used as a switch. This pin directly powers the transceiver, and is also fed to the inverting input of a comparator. The comparator switches the 9V supply to the LCD using a p-channel MOSFET¹⁶.

VI.B.3 Firmware

No firmware modules were written explicitly to handle power switching, but it is important to note that the LCD driver¹⁷ must be disabled before switching off the power. This disables the UART's hardware interrupt, which would otherwise fire when the transceiver's power is cut and the serial receive line drops to 0V.



Fig. 3 Beta Prototype (left-right) In enclosure, open, PCB version

VI.B.4 Prototype Evaluation

Beta radio range is similar to that of a dot, without peripherals, running off of a battery (~25M indoors through walls). This suggests that the ripple noted in VI.B.2.i is sufficiently small to be ignored. Power consumption of glue circuitry was smaller than 1mA, but the measurement tool used was not precise enough to give a better estimate. The constraint of building with only components available in through-hole packages was a successful one. The prototype was built quickly and inexpensively, and allowed a functional PCB to be designed on the first attempt. As a whole, the beta prototype was a success, but future prototypes must consider power consumption more closely.

¹⁶ See appendix C for circuit diagram

¹⁷ See printer driver WritePageM.nc in appendix D

VII Wireless Data Transfer Protocol

In order to send messages¹⁸ to its users, the ITS builds on the *ActiveMessage* radio protocol included with the TinyOS distribution. Building on this protocol requires that messages be broken up into parts (packets) by the PC and reassembled at each student node. Two versions of software were written to address issues of data integrity, latency and scalability related to the transfer, reception, and assembly of message packets.

VII.A.1 Data Transfer Protocol v1: Goals

In this iteration of the Protocol, it was important to ensure data integrity and minimize latency in communication between the PC and a single node. This iteration also ensured that questions were received in the proper order, and that nodes assembled all the parts of each message correctly. The FSB protocol was also implemented in this version, but due to its simplicity, it is covered only briefly.

VII.A.2.i Implementation: PC

All PC side software is written in Java, and interfaces with the base station using pre-built TinyOS tools. The PC protocol must send messages to all of the users, keep track of the status of each user to ensure that all users receive all of the messages, handle requests for missing packets, and log responses as they are received.

At the PC, messages are divided up into packets, each with 23 data bytes. Packaged with the data is the following header information:

- Message number (1-254)
- Current part number (1-255)
- Total number of parts (1-255)
- Node ID of origin (0-254)
- Number of bytes in data field (0-23)
- Next message to expect after this one (not used in this version)

These messages are embedded in an *Active Message* structure and sent over the radio to each node as requested. Once a node receives one of these packets, it has sufficient information to ensure that it receives the rest. It is the job of the PC to ensure that each node receives at least one packet of each message. This is done by setting a timeout timer for the node each time it is sent a new question and waiting for a node response. If the node does not respond with a packet request, message acknowledgement or answer before the timer fires, the PC will resend the first packet of the message and reset the timer. A simplified representation of program flow is as follows:

¹⁸ "Messages" and "questions" are used interchangeably throughout this paper. Both refer to a complete question in the Q&A application.

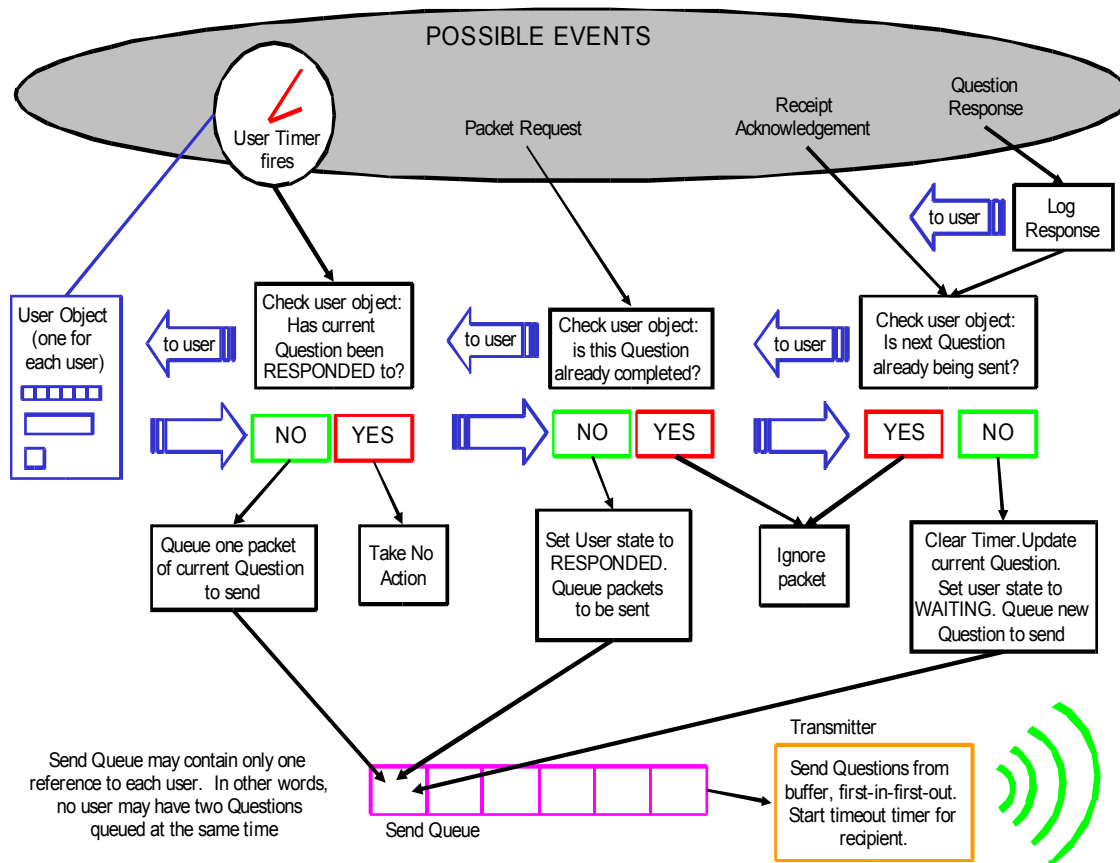


Fig. 4 PC Data Integrity Protocol

The FSB protocol is not shown in figure 4. Its implementation is very simple on the PC side, consisting only of a timer, which is used to limit a user to a hard-coded FSB rate. Otherwise, the protocol simply sends an acknowledgement to the node that sent the FSB packet and alerts the GUI (Graphical User Interface) of the received FSB's value (faster or slower) for visualization.

VII.A.2.ii Implementation: Remote Node

The task of the remote node is to assemble all the packets from each message in the correct order. Each node is responsible for requesting the packets it lacks and ignoring packets it does not need. In this version, remote nodes assume that they will be sent messages in the correct order. They do NOT assume packets are in the correct order. It is also the task of the node to reduce perceived latency.

VII.A.2.ii.a Data integrity

Each node maintains a data structure to keep track of the message it is currently building, and a list to track which messages it has already received. The structure contains, among other elements, a list of which packets have been received. The message list and the structure are consulted each time a new packet is received. The

node does not enforce data integrity on FSB packets. Program flow for the data integrity protocol is as follows:

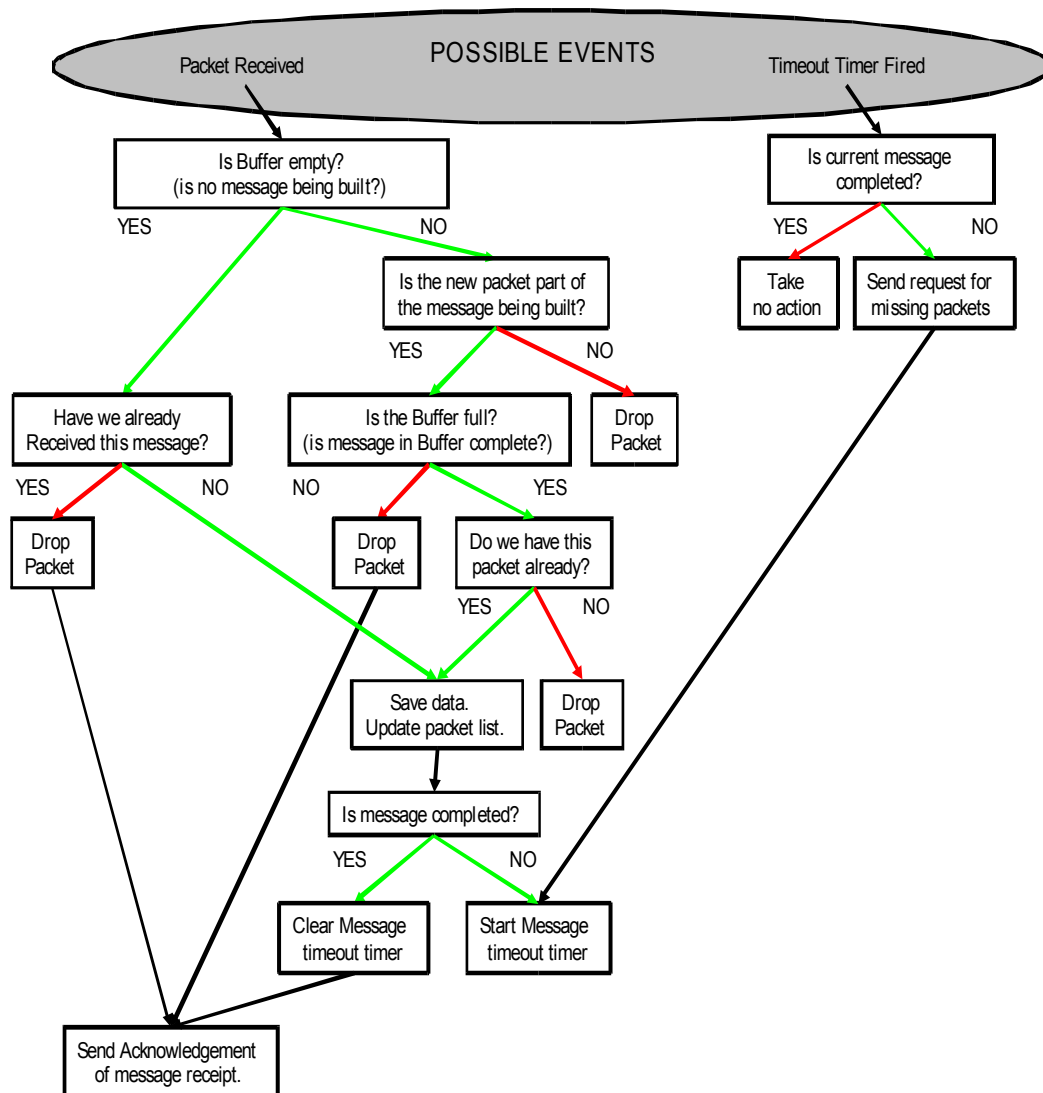


Fig. 5 Remote Node Data Integrity protocol

VII.A.2.ii.b Perceived Latency Reduction

In order to decrease perceived latency, each node implements a double buffering scheme that allows the user to read one message while another is being built in the background. Implementation on the node side is straightforward – two identical message buffers simply rotate as needed. On the PC side, double buffering causes some difficulty because the PC can no longer expect answers and acknowledgements to arrive in a specified order, but the PC can reliably sort out how it should react based on what it knows about which messages have been acknowledged, which messages have been responded to, and which message is currently being sent.

VII.A.3 Evaluation:

The first version transfer protocol exhibited extremely reliable transfer of messages. Once debugged, the protocol was able to transmit all of the message parts in every test case¹⁹, even when significant packet loss was induced by placing large physical separations and physical barriers between the node and the base station. Additionally, the protocol sent very few unnecessary packets²⁰ as measured by the number of repeat Receipt Acknowledgements received by the PC.

The major drawback of the protocol was lack of scalability. According to the PC program's internal timer, the average time to send a packet from the PC was approximately 100ms. Further investigation determined that nearly all of this time was required to physically transmit the message, with the message processing described in VII.A.2.i requiring but a few milliseconds at the high-end. Because of this low throughput, the average message in our test batch required over two seconds to send, causing the nodes' responses to outpace the PC application after only three nodes were added (fig. 6)

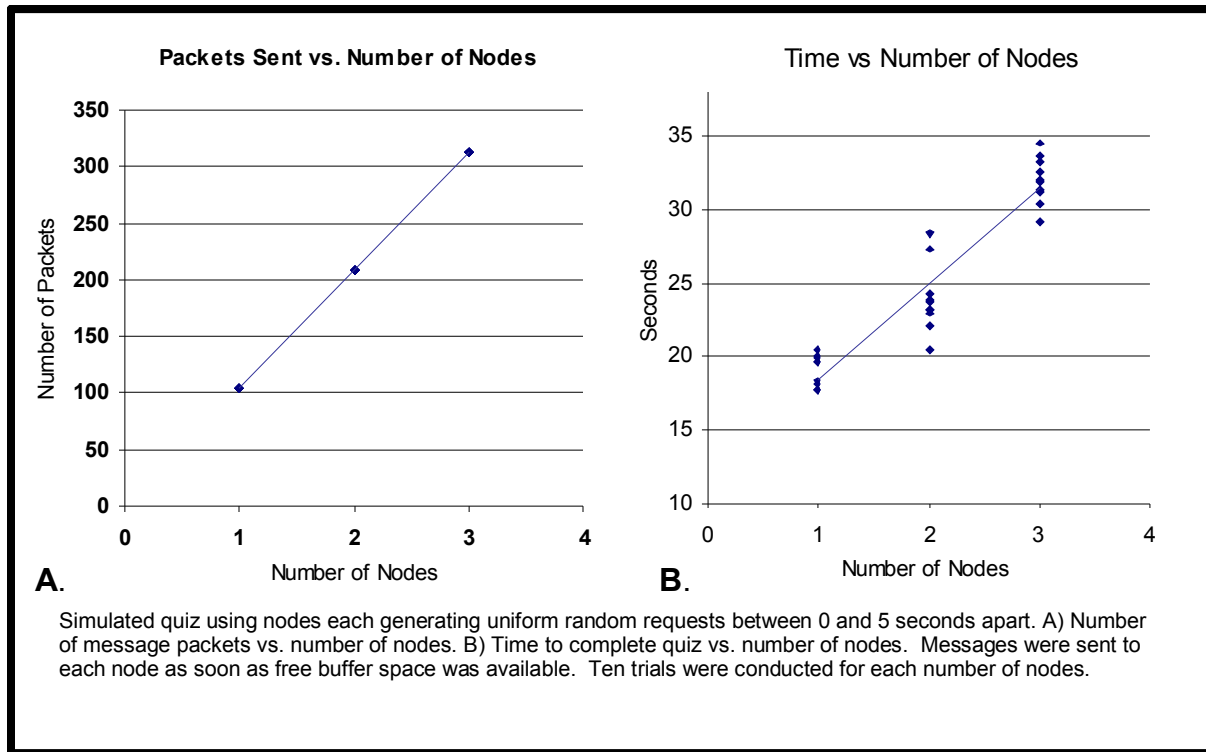


Fig. 6 Protocol v1 Test Quiz Results

As seen in figure fig. 6B, sending time scales linearly with number of nodes, even when only a few nodes are present. This effect is exaggerated in the figure because each node generated requests at a very high rate (once every 2.5 seconds on average), but it is still indicative of a major scaling problem. Assuming it takes 30 seconds for a

¹⁹ 40 tests total, each with 5 messages of average length just under 21 packets

²⁰ an unnecessary packet is defined as a packet that has already been received by the node

student to respond to a question, the system could only handle a maximum of 15 nodes (assuming message size of 20 packets) before falling behind in servicing requests.

VII.B Data Transfer Protocol v2

As noted in the evaluation of v1, the straightforward method of sending each message directly to nodes as requested was not scalable to large numbers of users. (Even if one were to delve into the TinyOS distributed Java code that handles the packet transmission and optimize it to send packets faster, there would ultimately be a low user limit beyond which the system would scale linearly) As a result, it was necessary to rework the transmission algorithm such that it made better use of available bandwidth.

VII.B.1 Goals

The aim of v2 was to build v1's robustness of data integrity into an algorithm that floods all message packets to all users at the same time, in an attempt to save bandwidth.

VII.B.2.i Implementation: PC

In order to implement the goals outlined in VII.B.1, the v1 code was modified in two ways. First, the Message queue was reorganized as an array of vectors with each array cell representing a question²¹ queued to be sent, and each associated vector representing the users the PC expects to receive the packets of that question (fig. 7). The queue is populated by the following rules:

When a user requests packets from a given message, the message is added to the queue array, and the user is added to the vector of users expected to receive packets from that question.

No message may appear more than once in the queue array.

If a user requests a message that is already in the array, the user is simply added to the vector of users expected to receive packets from that message.

If, for any reason, a user's request changes before its previous request is processed, the old request is deleted, and the previously requested message will not be sent unless other users are also waiting for it.

Second, the 'next message to expect' field mentioned in VII.A.2.i is filled with the next question in the quiz sequence (algorithm assumes all nodes receive the questions in the same order). This field is used by the nodes to differentiate packets they need from those that they don't.

²¹ A question is considered queued even if only one packet of that question is queued to be sent

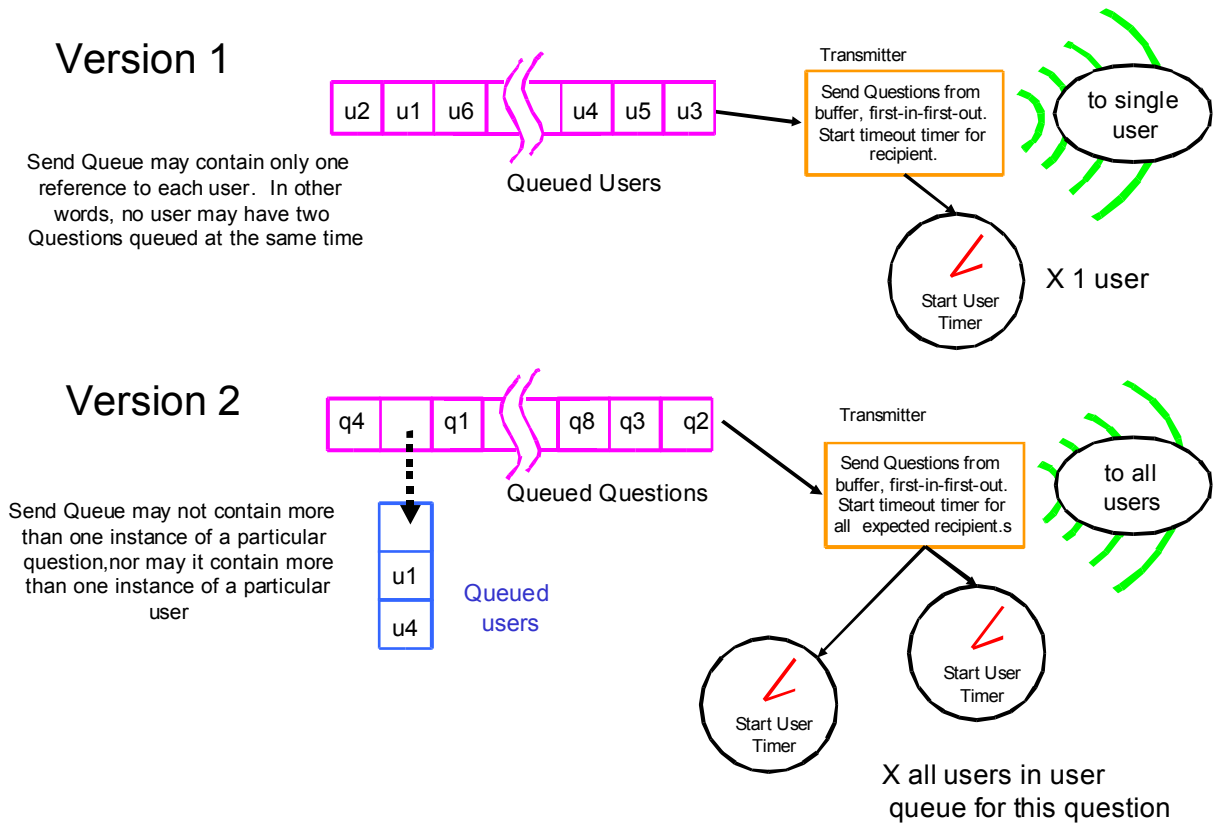


Fig. 7 Send Queues v1 vs. v2

VII.B.2.ii Implementation: Remote Node

Remote nodes were modified to support flooding by using the newly implemented 'next message to expect' field in the message packets. Using this field, each node can ensure that it processes messages in the right order, even if it does not receive them that way. The changed portion of the protocol from fig. 5 is below, with new boxes in red (if document is in color) (fig. 8).

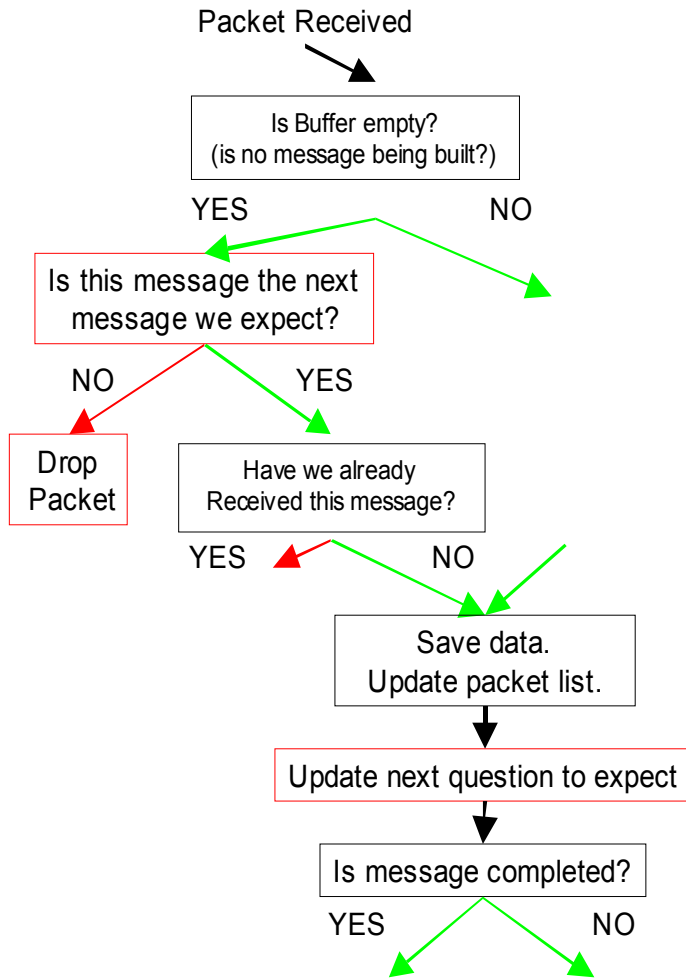


Fig. 8 Node Data Integrity Modified for Flooding

In addition to the above change, the remote node includes the status of its alternate buffer whenever it sends a response or acknowledgement to the PC, allowing the PC to more precisely determine what packets a given node is in need of.²²

VII.B.3 Evaluation

Extensive testing showed v2 to be as reliable as v1. More importantly, it also showed v2 to be much more scalable. In sixty trials of the quiz used to test v1, the new algorithm showed a large amount of bandwidth sharing and significantly decreased execution time when compared with v1 (fig. 9B). Using the new algorithm in this test, packet transmission with respect to node number loosely approximates a logarithmic increase ($R^2 = .566$) (fig. 9A).

²² For more information on how this works, see the source code in appendix F

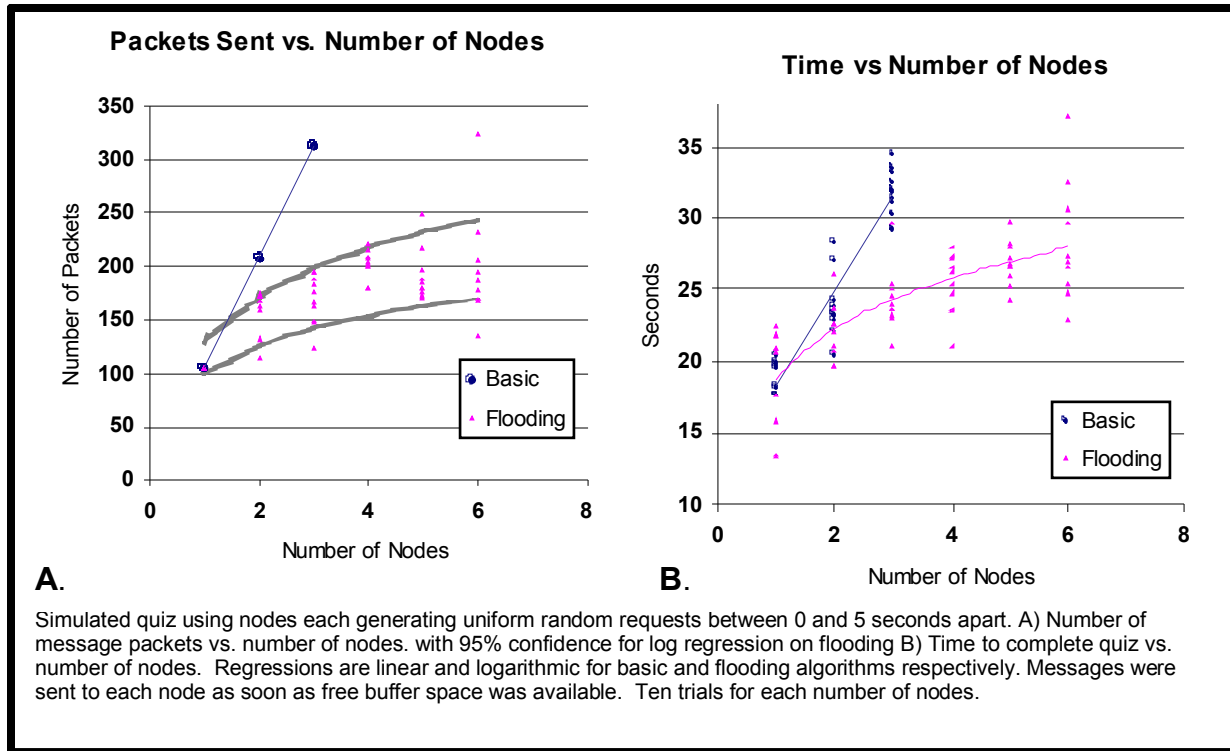


Fig. 9 Protocol v2 Test Quiz Results vs. Protocol v1 Results

Because the v2 algorithm depends on multiple users requesting the same question at the same time, it becomes more efficient as requests for a particular question become more clustered. This is often a desirable feature, as seen in our very closely clustered test scenario. Conversely, less bandwidth is shared as requests for any given question become more temporally spread out. This is not problematic as long as users work through a quiz at roughly the same average rate, but could become burdensome in a long quiz if the overall response rate remains high while the individual users all work through the quiz at different rates. In this scenario, the algorithm would become less and less efficient as the requests for any particular question begin to spread temporally, but the request load on the system would not decrease. If the students are spread evenly enough apart in the quiz, the v2 algorithm may begin to resemble the v1 algorithm, which would be problematic. There was insufficient time to test or model this final scenario. More research is suggested to evaluate its consequences.

VIII Application Software

Although it is one of the more straightforward aspects to explain, the user interface is one of the most important aspects of the ITS. Because the ITS is designed for use in real-time teaching situations, it is essential to include simple, intuitive tools for data visualization and user control. This is most important on the PC side, where the user's options are numerous.

VIII.A Student Device

The student interface is very simple. It is also fully automated. The nodes automatically display any questions that they receive, provided the users have finished with the previous ones, and successful submissions of responses are signaled by a flash of the dot's LED²³. The student has the ability to freely scroll text that overflows the LCD screen, and his or her question responses are checked by the software to ensure that they do not exceed the appropriate range. The student may press use FSB by pressing either the 'faster' or 'slower' button at any time once the node is logged in.

VIII.B PC

The PC interface consists of three screens. The first screen displays a list of users and their responses to any pending questions. The application highlights correct responses in green. Fig. 10 explains the button functions.

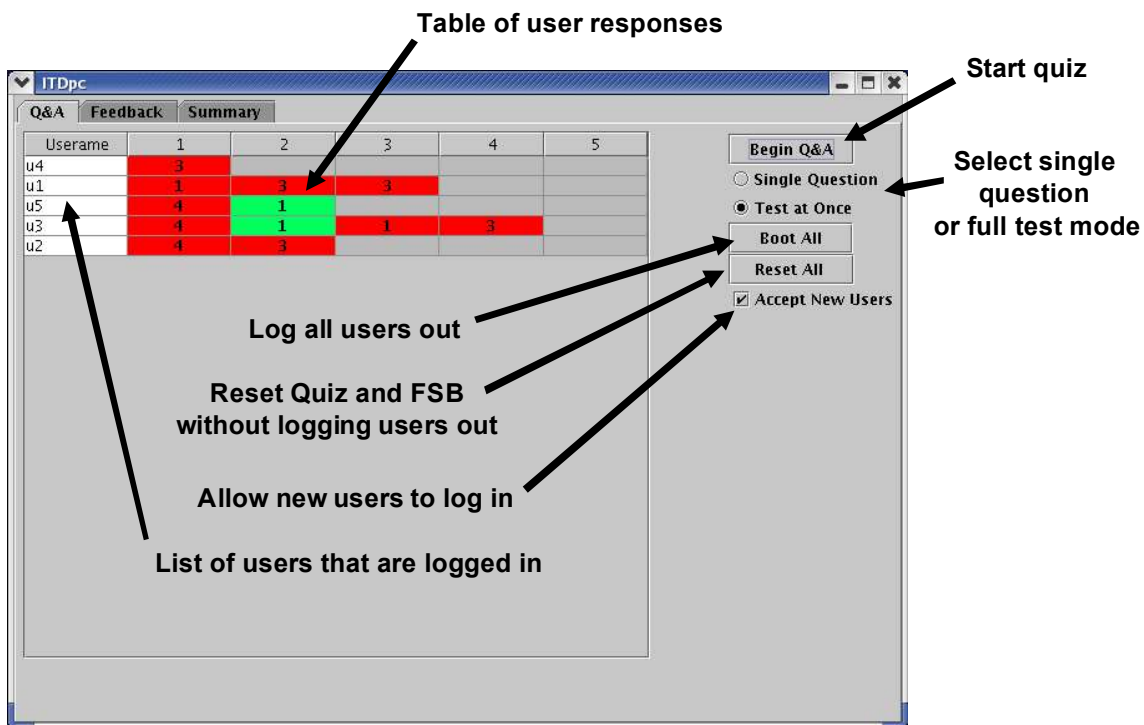


Fig. 10 Q&A Control Screen

The second screen shows a histogram of FSB feedback. Green represents 'faster' and red represents 'slower'. A 'faster' response indicates to a lecturer that he or she is moving too slowly, while a 'slower' response indicates he or she is moving too quickly.

²³ Note: current hardware prototype does not allow dot's LED to be seen. This must be corrected in the next prototype.

One very important feature of this screen is the 'Show Fast and Slow' option. In 'Show Fast and Slow' mode, the histogram shows 'faster' and 'slower' responses simultaneously. In situations where a class is stratified into two groups, one of which is confused and the other bored, this option will provide a striking graphical representation of their bipolar feedback (Fig. 11). This representation would be masked if response averaging is enabled.

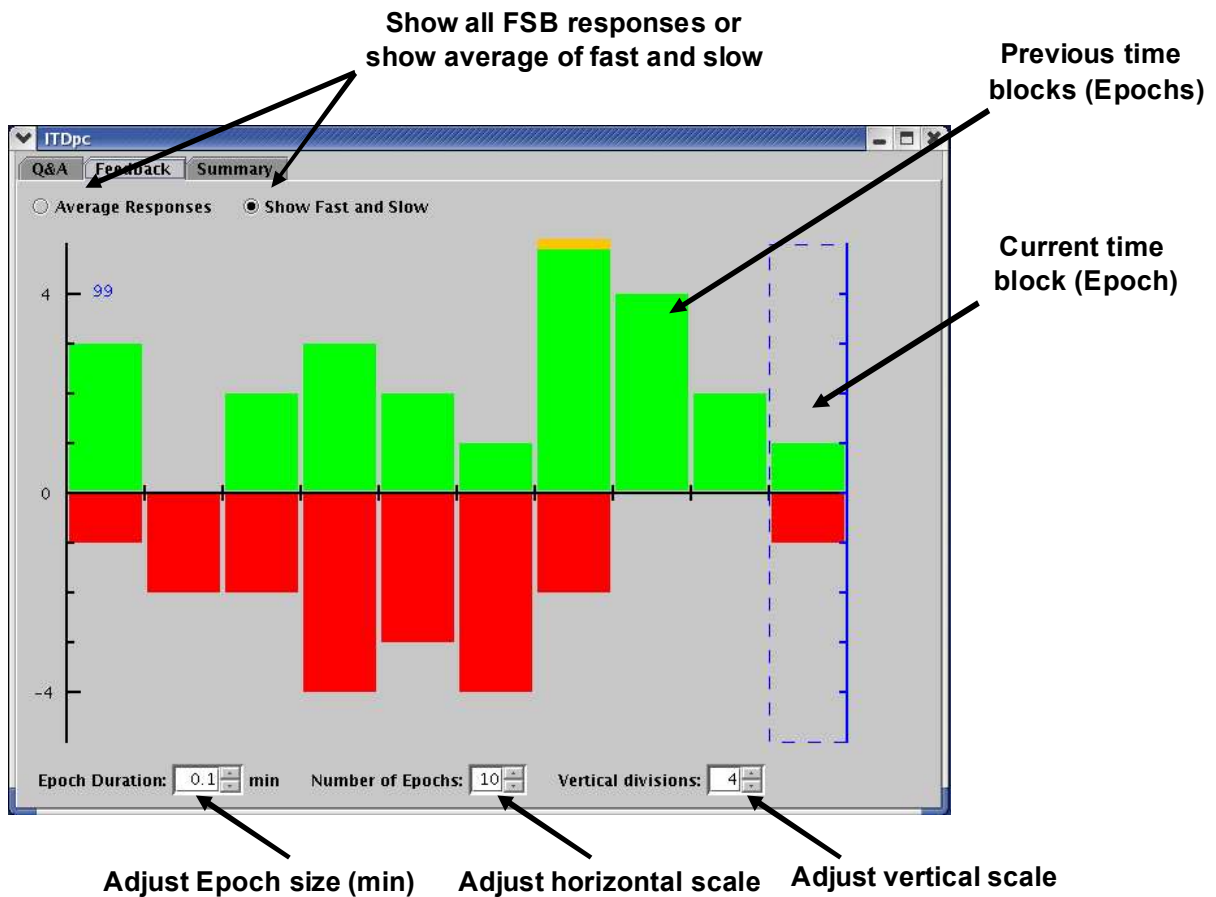


Fig. 11 Feedback Summary Screen

The third screen shows a real-time summary of student responses, indexed by question number. The correct response can be highlighted by selecting a checkbox (Fig. 12).

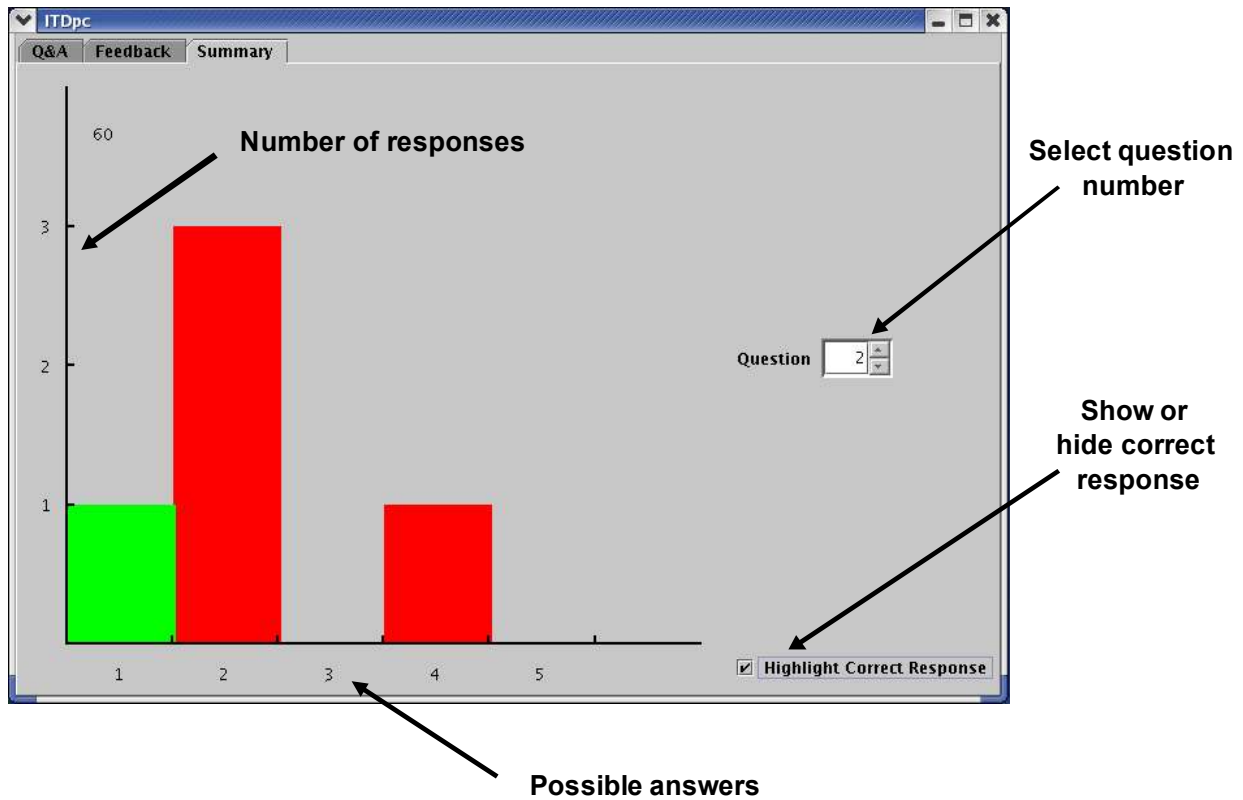


Fig. 12 Q&A Summary by Question

IX System Cost

Development cost for this project was approximately \$560. Itemized costs can be found in appendix A. Production cost for the handheld student device in 1000 count was calculated at \$108 not including packaging and assembly²⁴. The dot and the LCD make up the majority of this cost. Even with a node cost of \$108, the ITS is considerably less expensive, albeit less full featured, than it's major competitor, the TI-Navigator²⁵, but it would be optimal to decrease assembled node cost to \$75 or less. A cost reduction could be achieved by custom designing a processor platform that is less full-featured and less expensive than the dot, and making larger production runs to take advantage of bulk pricing. Production cost for the Base station was not calculated because it would make up a negligible part of the cost in large systems. The current

²⁴ Cost based on production run of 1000 units. See appendix B for details

²⁵ TI-Navigator is a wireless data acquisition system designed for use with Texas Instruments calculators. The system costs \$4000 for a 32-person set-up, not including calculators. For more information see: <http://education.ti.com/us/product/tech/navigator/features/features.html>

retail price for the base station is: \$80, not including the mica2 or mica2dot mote that plugs into it.

X Conclusions and Suggested Future Work

X.A Summary of Achievement

At the close of this project, the ITS has taken significant steps toward a finalized, mass producible product. The system's data transfer protocols are reliable and efficient; issues of encryption and crosstalk have been addressed; a functional prototype PC GUI has been constructed; applications for Question & Answer and Lecturer feedback have been written; and the system has been thoroughly tested in an uncontrolled environment with excellent results.

X.B Future Work

The largest concern for system viability, as noted in IX, is hardware cost. This is an issue that must be addressed more thoroughly before the ITS can be mass-produced. It is not cost effective to buy the dot platform as a stand-alone and plug it into the device. Future designs must scale down the dot, as explained in IX, and custom build it onto a single board containing all of the glue electronics. In this process, more time should be devoted to the issue of power consumption. This project dealt with power consumption only on a macro-level. Further investigations must focus more closely on the details of power consumption control – making component selections for low quiescent current and implementing of software support for duty cycling and long periods of processor sleep. Future software design and testing should include additional testing of the v2 data transfer protocols using varying response times and larger numbers of nodes. Revisions should also be made to implement more robust error checking and optimize for efficiency within the ITS code. Most importantly, the issue of PC side radio throughput must be addressed, as current throughput is less than 1/5 of intuitively expected throughput based on the radio's data rate²⁶.

²⁶ For more information on dot's radio see the Crossbow wireless tutorial: http://www.xbow.com/Support/Support_pdf_files/Motetraining/Wireless.pdf

Sources Consulted

Atmel Atmega128 Manual,

http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

Hayter, Anthony J. Probability and Statistics for Engineers and Scientists,
Duxbury, Pacific Grove, CA. 2002.

Horowitz and Hill. The Art of Electronics, Second Ed., Cambridge University
Press, New York. 1989.

Kernighan, B. and Ritchie, D. The C Programming Language, Prentice Hall,
Englewood Cliffs, NJ, 1988.

TinyOS Documentation, <http://tinyos.net/tinyos-1.x/doc/>

Van Der Linden, Peter. Just Java 2, Fifth Ed., Sun Microsystems Press, Palo
Alto, CA, 2002.

Appendix A: Itemized Project Cost

Description	Manufacturer	Part #	qty	cost/ea.	Total	Notes
LCD	Scott Edwards Electronics	G12864	2	\$170.00	\$340.00	
Keypad	Gravhill	96BB2-006-R	2	\$12.87	\$25.74	
matrix > serial converter	E-Lab	EDE1144	2	\$7.00	\$14.00	Not used
prototyping board	3m	923253-I	2	\$15.44	\$30.88	
Serial Transceiver	Maxim	MAX3232EEPE	5	\$4.95	\$24.75	Smallet package was 5
Low ESR capacitor 100uF	Sanvo	OS-CON	2	N/C		
DC-DC Converter	Linear Technology	LT1073	1	N/C		
Comparator	Texas Instruments	LM311	1	N/C		
P-Channel MOSFET	Discrete Semiconductor	ZVP3310A	1	N/C		
Misc. wire/hardware	N/A	N/A	1	\$15.00	\$15.00	
Misc. Passives	N/A	N/A	1	\$15.00	\$15.00	@Radioshack prices
Batteries	N/A	N/A	7	\$4.00	\$28.00	@Radioshack prices
mica2	Crossbow	MPR410	1	N/C		
mica2dot	Crossbow	MPR510	6	N/C		
mica2dot breakout board	Crossbow	MDA500	5	N/C		
Programing board	Crossbow	MIB500	1	N/C		
FOR PCB						
PCB prototype (3 boards)		custom	1	\$50.00	\$50.00	Price is estimate. Have not received bill.
Serial Transceiver	Maxim	MAX3232CWE	2	\$2.86	\$5.72	
Low ESR capacitor 220uF	Kemet	T495C227K006AS	2	\$2.30	\$4.60	
DC-DC Converter	Linear Technology	LT1073CS8	1	\$5.50	\$5.50	
Comparator	Texas Instruments	LM311DR	2	\$0.40	\$0.80	
P-Channel MOSFET	Discrete Semiconductor	ZVP3310A	2	\$1.00	\$2.00	
Inductor 120uH	JW Miller Magnetics	6000-121K	2	\$1.51	\$3.02	
Misc passives	N/A	N/A		N/C		
				TOTAL	\$565.01	

Appendix B: Production Cost in Lots of 1000

Description	Manufacturer	Part #	qty	cost/ea.	Total	Notes
PCB		custom	1	\$5.00	\$5.00	Ballpark Estimate
Keypad	Grayhill	96BB2-006-R	1	\$6.95	\$6.95	
mica2dot	Crossbow	N/A	1	\$50.00	\$50.00	Ballpark estimate from modifying mica bill of materials http://webs.cs.berkeley.edu/tos/hardware/design/ORCAD_FILES/MICA/MICA_BOM.xls
LCD	AZ Displays	AGM1264F-FL-GBD	1	\$23.81	\$23.81	
Serial Backpack	N/A	N/A	1	\$12.00	\$12.00	
Serial Transceiver	Maxim	MAX3232CWE	1	\$1.85	\$1.85	Must custom build based on design such as Scott Edward Electroncis' 'Serial Backpack'
Low ESR capacitor 220uF	Kemet	T495C227K006AS	1	\$1.75	\$1.75	
DC-DC Converter	Linear Technology	LT1073CS8	1	\$3.45	\$3.45	
Comparator	Texas Instruments	LM311DR	2	\$0.15	\$0.30	
P-Channel MOSFET	Discrete Semiconductor	ZVP3310A	2	\$0.28	\$0.56	
Inductor 120uH	JW Miller Magnetics	6000-121K	2	\$0.48	\$0.96	
Switching Diode	Discrete Semiconductor	1N4001	4	\$0.03	\$0.12	
Schottky Diode	Discrete Semiconductor	1N5818	1	\$0.11	\$0.11	
Misc passives	N/A	N/A	1	\$1.00	\$1.00	

TOTAL	\$107.86
--------------	-----------------

