EnginuiTech

RAVS Process Report

Submitted by	EnginuiTech
Contact	Bill Moats School of Engineering Science Simon Fraser University wmoats@sfu.ca
Submitted to	Andrew Rawicz School of Engineering Science Simon Fraser University Steve Whitmore School of Engineering Science Simon Fraser University

April 20, 1999

Table of Contents

List of Figures

List of Tables

1. Introduction

For the entire Spring ’99 semester, Enginuitech has been developing the Remote Automated Vending Statistics (RAVS) System – which is intended to monitor the pop levels, temperature, and customer purchases in a pop machine and e-mail the servicer when a pop level gets low. The details of the system are contained in the functional and design specifications which are available at the company website (www.sfu.ca/~wmoats). This document discusses the current state of the RAVS system and outlines the different phases of the project, the challenges experienced during the development of the prototype system, and the necessary modifications that were made. Included in this report is a section authored by each individual team member discussing how Enginuitech functioned as a team, what valuable technical experience was gained from the project, and what changes would have made the project development smoother.

2. Current State of the RAVS System

As defined in the RAVS project documentation, the RAVS system is intended to automatically monitor the status of a pop machine (pop quantity, temperature, and customer puchases) and report important information such as an empty bin or high temperature to the user in an e-mail or by means of paging. The system is composed of two high level systems – the monitoring unit (contained inside the pop machine) and the host system (which manages multiple pop machines and reports to the user of the system). The high level system diagram is shown in Figure 1.

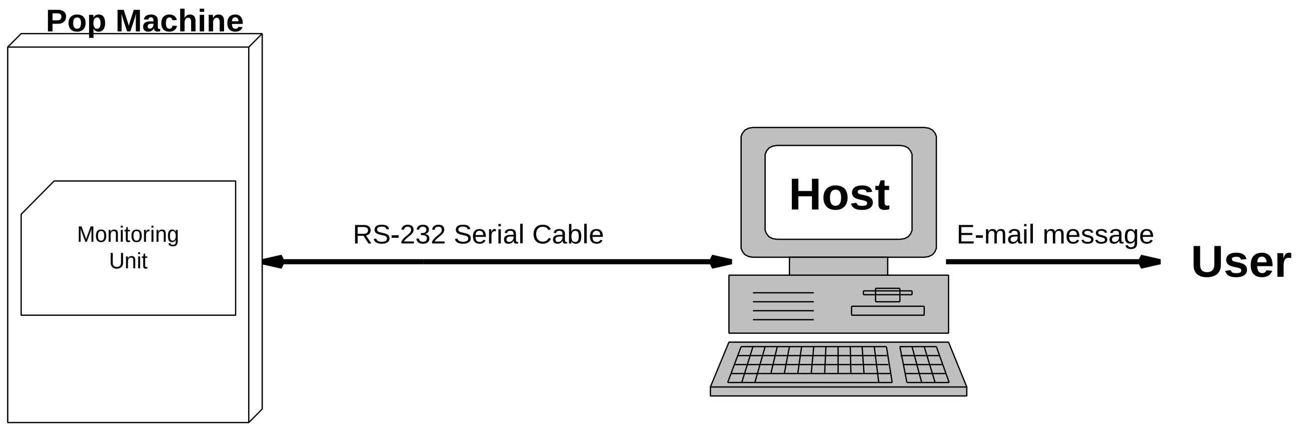


Figure 1 High Level RAVS System Diagram

These systems were developed independently with a common communications protocol (VIP) and will thus be discussed separately in the following sections.

2.1 Monitoring Unit

The RAVS monitoring unit prototype consist of the systems shown in Figure 2 (which were specified in the design specification).

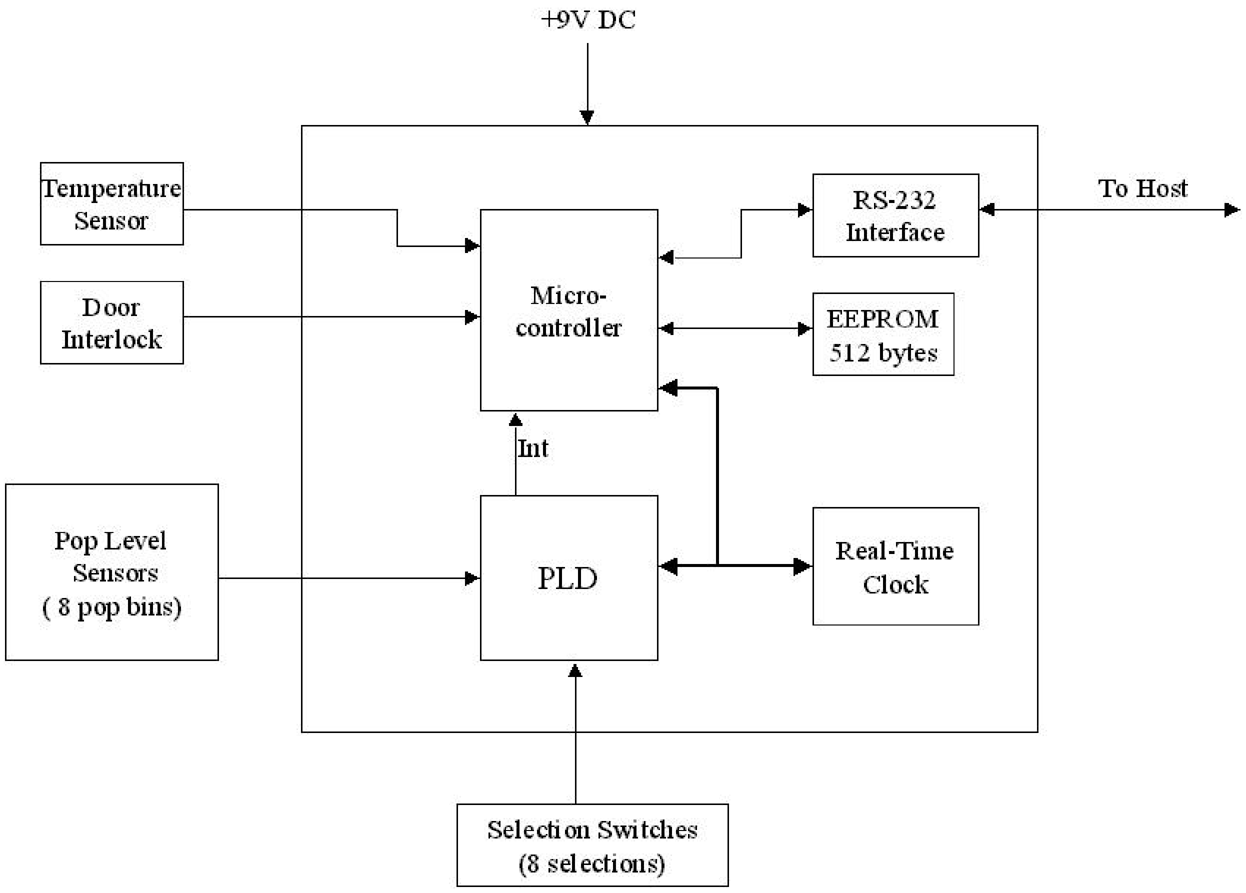


Figure 2 Monitoring Unit High Level System Diagram

2.1.1 Hardware

The monitoring unit hardware remains relatively unchanged from what was specified in the design specification. A PIC 16C74A microcontroller supervises and controls all operations on board the monitoring unit and communicates to the host system over a serial RS-232 connection. This serial connection operates at 300 baud which is slow enough to allow the microcontroller a considerable amount of time to process the incoming messages.

Due to the large number of sensor inputs, the microcontroller is interfaced to the sensors through a MAX7128 EPLD, which allows the microcontroller to access the sensors by means of memory mapping. This scheme also allows upgradability to a much larger number of sensors. The real-time clock allows the logging of the time when a purchase is made (we found this to be more convenient than writing complex timing software).

The door interlock switch is connected to the front door of the pop machine to allow the monitoring unit to update its internal settings when the pop machine has been serviced.

The pop level sensors are implemented as infrared reflective object sensors as described in the design specification. The infrared emitters are modulated with a 1KHz square wave (generated by the real-time clock) and are demodulated and converted to a digital signal with arrays of op-amps, envelope-follower (AM detector), and high speed comparitors. Some minor changes that were necessary in this circuit are described in Section 3.1.1.

The customer selections are detected by placing reed switches in the path of the mechanical vending mechanism of the pop machine such that every time a pop is sold the switch is momentarily closed. The microcontroller software then records these selections with the current time and stores the records in the EEPROM (Xicor X25043).

The monitoring unit interface consists of three LEDs. One indicates if the unit is powered, and two are used to indicate if the unit is transmitting or receiving information to or from the host.

The RAVS prototype system is set up to monitor the status of two pop bins with three quantity sensors per bin although all of the hardware is present to implement eight bins, we found two bins would be sufficient to demonstrate the functionality of the project. The software and hardware is all present to implement the temperature sensor, however due to time constraints we were unable to build the temperature sensor itself.

2.1.2 Software/Operation

The software in the monitoring unit

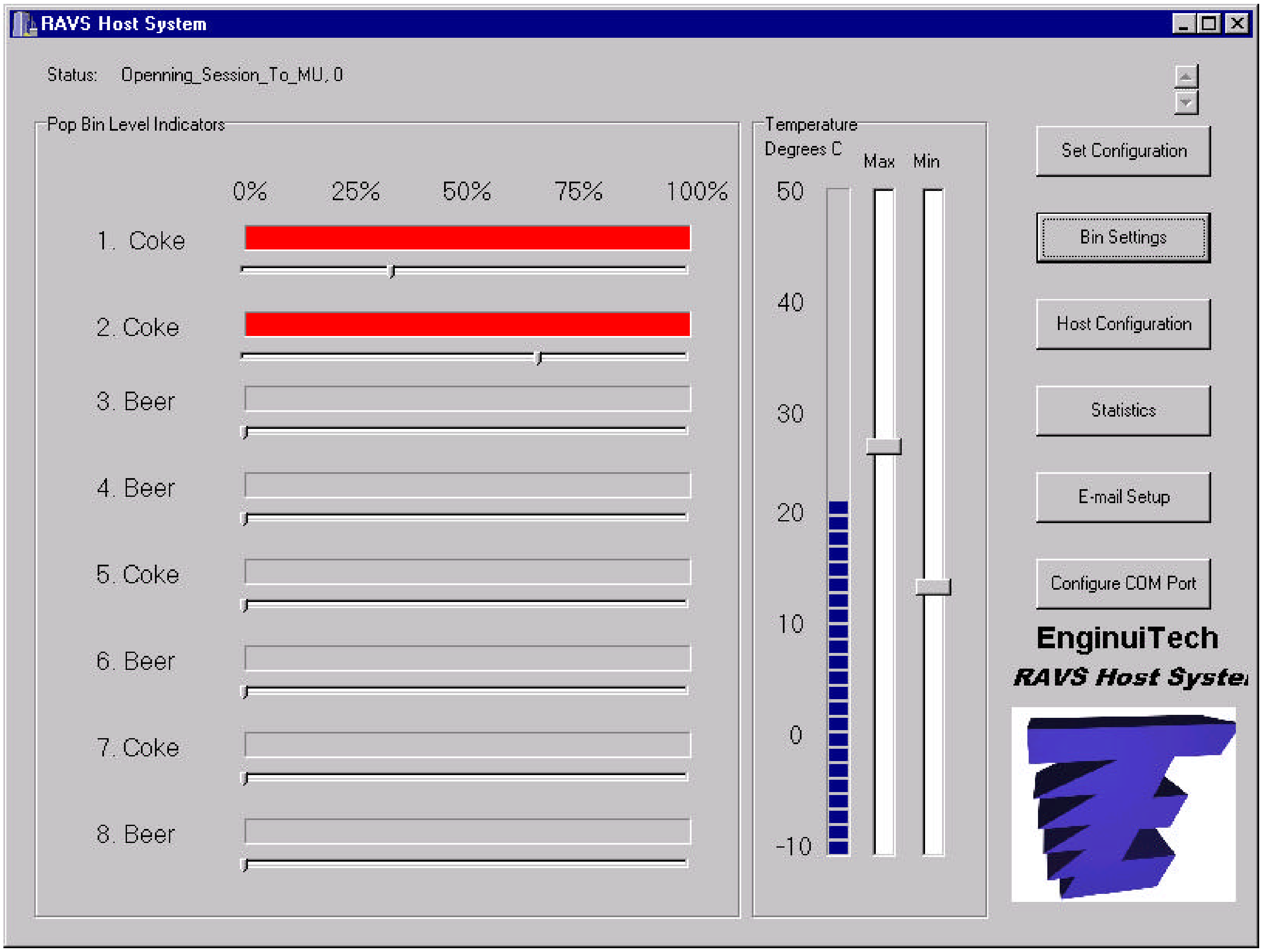
- Implements the Vending Information Protocol (VIP) as defined in the design specification
- Monitors the level of pop in the machine on request, when a pop has been sold, or when the door of the pop machine is closed
- Stores up to 15 selection logs in the EEPROM and begins a download to the host after receiving 8 logs or when requested by host
- Sets and reports alarms to the host based on the configurable conditions
- Stores and uses configuration parameters (alarms on/off, pop alarm levels, temperature range)

2.2 Host

The host was responsible for downloading the information from the Monitoring Unit as well as provide an interface between the Monitoring Unit and the user. The following features of the Host have been implemented:

- Display current pop level in the machine.
- Display the current temperature.
- Download sales logs and record them when an alarm occurs or when user requests them.
- Updates the information on the current MU status whenever an alarm occurs or when user requests a sales logs download.
- Send an alphanumeric message to a fido phone when an alarm occurs.
- Send an e-mail message when an alarm occurs.
- Set the current date time on the Monitoring Unit.
- Log all MU alarms including buffer full and overflow, pop level low, and temperature out of range, and power up.
- Configure the minimum allowable pop levels.
- Configure the valid temperature range.
- Enable/Disable the pop level alarms, and temperature alarms.
- Assign flavor labels to bin monitored.
- Select com port at which the Monitoring Unit is attached to.
- Configure the pager information.
- Configure the e-mail information.

Below is a screen shot of the current graphical user interface for the host:



Screen Shot of the Host

3. System Modifications

The RAVS system prototype performs all the primary functions it was designed to perform. Certain portions of the system were downgraded in size in order to make the project manageable and able to be constructed in the relatively short development time and certain modifications were made to the sensor circuity to make them funcional.

Since the purpose of the RAVS system was to prove the concept is feasible, we have implemented the system for one monitoring unit and one host system, which are connected with null modem cable. The monitoring unit and host are both capable of monitoring eight pop bins, but we have produced enough sensors to monitor two pop bins for demonstration purposes.

3.1 Monitoring Unit

3.1.1 Sensors

When we began the RAVS project we decided that we wanted to sense

- The quantity of the pop in 16 pop bins of the vending machine
- The temperature of the interior of the pop machine
- If the machine has been tipped
- If the door of the pop machine is open

Initially we intended to implement the pop quantity sensors with an infrared “break” sensor where an infrared emitter would send a narrow beam of infrared light across the pop bin and would be detected on the other side. If a pop was present the signal would be blocked and thus the quantity of pop could be estimated. We eventually discovered that the range of the transmitter and detector was extremely small (~15 cm max). Thus modified the system to be a reflective object sensor such that the exact same circuitry was used but the light beam is bounced off the object and back to the sensor.

Part way through the project, the tipping sensor was scrapped due to lack of interest by industry contacts (Coca-Cola) and the complexity and cost of implementing the sensor. The number of pop bins to monitor was reduced to eight to simplify the project (and pin count) which allowed us to use a much simpler programmable logic device. The number of bins could easily be increased by using a slightly larger EPLD.

There were some changes made in the sensor circuitry from the one in the design specification. The old and new schematics are shown in
Figure 3 and Figure 4.

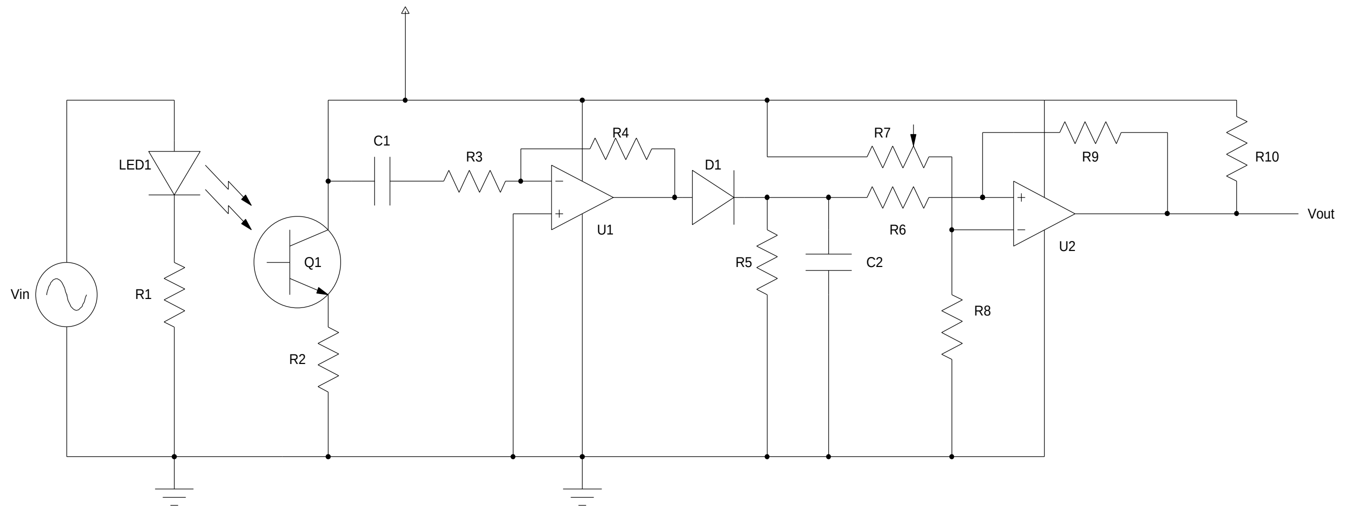


Figure 3 Original IR Sensing Circuit

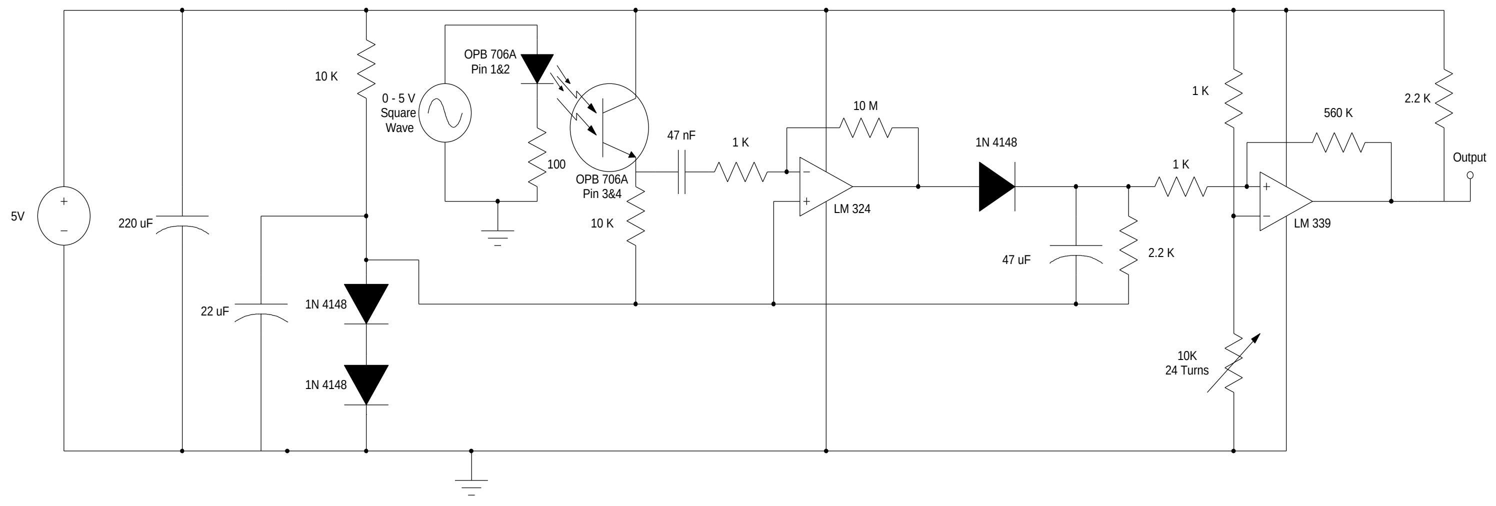


Figure 4 Final IR Sensing Circuit

The main difference in the current design is the implementation of the virtual ground. Do to the low voltage level from the photo-transistor, we found that the op-amp was unable to amplify the signal with a single voltage supply. To remedy this problem we simulated a dual voltage supply using the extra circuitry shown in Figure 4 creating a “virtual ground”. This caused the op-amp to operate in it’s active region and could thus amplify the small signal. An additional change that is only obvious on the proto-board was the implementation of voltage references for each comparator integrated circuit. Do to the difference in the semiconductors, we found that the difference circuits had different threshold voltages, this caused complications in comparing the signals to a single reference.

3.1.2 Data Storage / Software

We decreased the number of logs for the monitoring unit to maintain from 200 down to 15 after observing that the number of pops purchased in an hour is relatively low and it would be more effective to store fewer records and download them more often. It also simplified the design allowing all records to be stored in an EEPROM .

We also made a minor change in the way the data log and the pop levels are updated to the host. Instead of the monitoring unit periodically generating “download” interrupts we made the host periodically request the data log. This change simplified the software and the hardware considerably since the complex interrupt method with the real-time clock was replaced with a timed function in the Windows software (which is much easier to write)

3.1.3 Power Supply

In order to comply with CSA requirements and reduce the chances of injury of a team member due to electrical shock, we reduced the power supply requirements of the monitoring unit from 110VAC to 9VDC. This change also requires the system to include an external wall adapter.

3.2 Host

Most of the proposed functionality for the host in the design specifications was implemented. However, the system architecture of the host design was simplified since that, for demonstrational purposes, the host was only in charge of managing a single Monitoring Unit. For comparative purposes, the original and simplified host architecture are shown below:

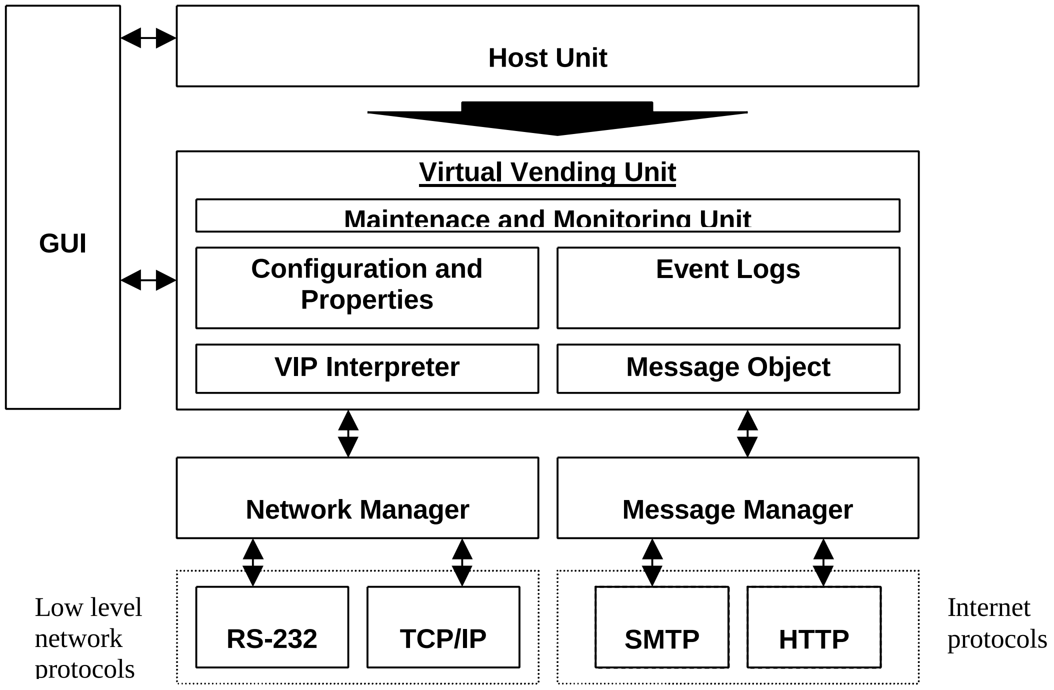


Figure 5 Original System Architecture

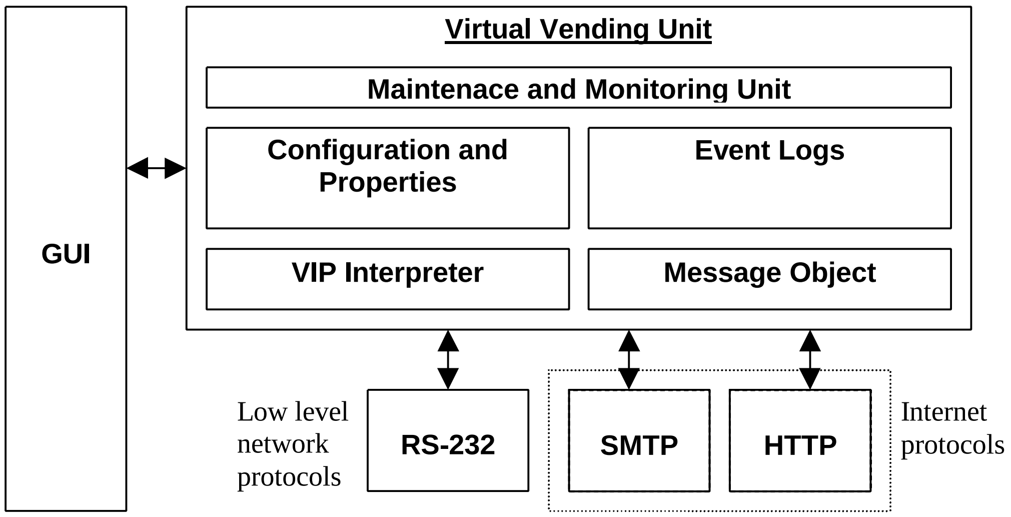


Figure 6 Simplified System Architecture

Also, because we were not going to add any support for new networking protocols as well as new Internet protocols in the near future, we did not create a **Network Manager** nor a **Message Manager** to allow further expansion of supported protocols.

The following is a list of other miscellaneous changes we made on the host design:

- Because we did not have a **Host Unit**, parameters such as vending machine name, and vending machine ID were not stored.
- Because of time constraints, we did not write the necessary code required to periodically update the Monitoring Unit status. Instead, the status was only updated when an alarm even occurred, or when the user explicitly requested it by downloading the sales logs.
- To simplify the user interface code, the pager and messages sent in response to an alarm was hard coded instead of being modifiable by the user.
- We no longer felt that the door status provided very useful information to the user so it was not displayed on the Graphical User Interface.
- Host initiated actions such as requesting for sales logs, and current bin level status was also logged for debugging purposes.
- The sales logs was written to a file separate from the main events log file so there was no need to write a module to parse the log file and produce a separate sales log file.

3.3 Vending Information Protocol (VIP)

The Vending Information Protocol (VIP) changed very little throughout the development of the system. Several debugging commands were added to ease the testing of the software in the system and some extra parameters were added to a few commands to reduce the number of communications necessary to read information from the monitoring unit.

4. Future Plans for the RAVS System

The team members of Enginuitech intend to complete and optimize the RAVS system after the completion of the ENSC 370 course and possibly pursue the marketing of the system.

4.1 Completion of System Features

Once the team has sufficient time without interference from coursework, we intend to implement the temperature sensor and all eight pop bin sensors and complete a full test in a pop machine over an reasonably extended period of time (one week or more). Once the system has been proven and all the bugs removed we will design and fabricate a PCB and proper housing structure for the monitoring unit (with all sensor circuitry contained within the box). Due to reliability and cost concerns, we may pursue a different method of pop quantity sensing as described in the enhancements section.

The windows software will be modified to allow multiple pop machines to be monitored – assuming a new communications medium is implemented (as described in the next section)

4.2 Enhancements to the RAVS System

The RAVS system as a whole could be upgraded to use a more realistic communications system, preferably a wireless system which would put least number of constraints on the location of a pop machine. Some viable communications systems could be wireless or dial-up modems, wireless RS-232 transceivers, or two-way paging systems. The network protocol could be slightly changed to allow all communications to take place on the same communications channel - thus using resources much more efficiently for large numbers of pop machines in close proximity to one another.

4.3 Monitoring Unit

Several modifications and enhancements could be made to the monitoring unit itself to make it more practical, economical, and marketable.

The monitoring unit could be enhanced by modifying the sensing method to something more economical and reliable such as mechanical switches, weight sensors, or mechanical height sensing (large sliding potentiometer). This simplification in the sensing method would reduce a great deal of the sensor circuitry required on board the monitoring unit.

The baud rate between the host and the monitoring could be increased to allow faster communications, which would allow the networking of many monitoring units.

The onboard real-time clock could be replaced with software to reduce the amount of hardware required in the system. A minor side effect of this is that the time would have to be set every time the monitoring unit is powered up. Also this would reduce the chip count to four, which would greatly reduce the cost.

The number of selection records maintained by the monitoring unit could easily be increased (change in constant in software) such that it could manage a practical number of sales before requiring communication with the host software.

4.4 Host

A needed enhancement to the host before it can be marketed is the ability to manage multiple vending machines by completely implementing the original system architecture of the host. Also an alternate graphical user interface has to be created to display current vending machine status is a more compact display. Configuration dialog windows will also have to be modified to support configuration of different vending machines. Field tests also need to be done on the user interface to verify its usability.

By adding modules such as **Network Managers**, the host can also be extended to support the addition of new networking protocols for communication with the Monitoring Unit. This feature will allow the host to communicate with the MU over different network mediums as they become available.

The host will also need the ability to auto-update its components, especially the paging components which use the web interface, and will need periodic updates as paging, and paging cellular providers change their web interfaces.

Finally, the Application Programming Interfaces need to be further defined and refined for each of our software modules. That way, new components can be easily added and integrated into the host. Also, interfaces can also be created for third party developers to create plugins such as add-ons which provide statistical analysis packages or the ability to interface to a database system.

5. Deviations in Budget and Time

Table 1 shows the comparison of our proposed budget and our actual cost for the RAVS system development. Note that several of the major costs and some of integrated circuits were donated or sampled.

Table 1 Budget Comparison

Materials	Proposed Cost	Actual Cost
Sensors (infrared, temperature, selection)	$100	$23.94
Used Pop Machine	$500	Borrowed (EUSS)
Used PC and Modem	$50	Borrowed
Microcontroller, Clock, Memory	$200	$72.44
Cables, Wires, Passive Components	$75	$30.23
PC Board	$80	Not fabricated
Miscellaneous (Box, Connectors, Proto-Board, Sockets, Sensor Ics)	$50	$79.82
TOTAL	$1055	$210.43

The RAVS project has come greatly under budget thanks to donations and samples of components from distributors and the use of the EUSS pop machine for our sensor development and demonstration. Another reason for being greatly under budget is we overestimated the cost of several of the components such as sensors and integrated circuits.

The miscellaneous section is over budget for several reasons. At the outset of the project we had envisioned different requirements for the sensors, thus we didn't budget for a significant amount of sensor circuitry. We also didn't expect the high cost of a metal enclosure and connectors. Since we eventually decided not to produce a PC board (in the interest of time) we included the proto-board in the miscellaneous section of the budget.

The Gantt chart showing our proposed schedule and actual time usage is shown in Figure 3.

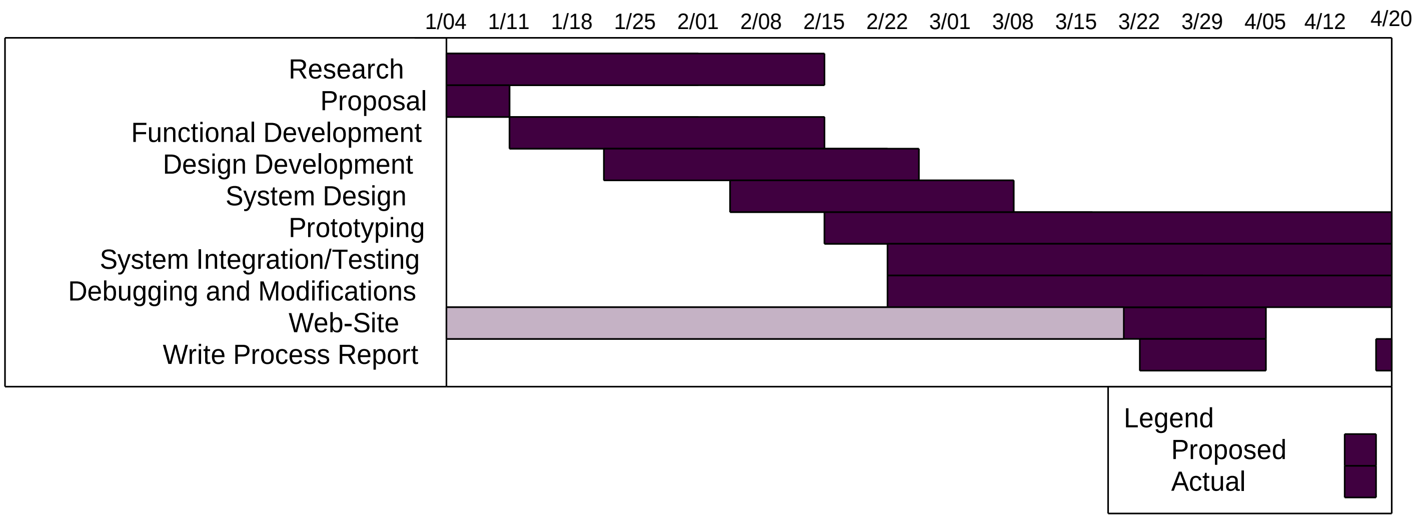


Figure 7 Proposed and Actual Time Usage

It is notable that for most of our schedule we extended over the allotted time by a great deal with the exception of the web page, which was completed ahead of schedule. The main reason for our difficulty in conforming to the schedule was optimism in the time needed for writing the rather intensive software, and difficulty in work around a quite differing and demanding courseloads. The largest deviations from our time schedule occurred at the end of the semester because our design extended over final exams and the completion of large projects in other courses.

6. Group Dynamics and Experience

6.1 Bill Moats

During the development of the RAVS system, for the spring '99 semester, I have gained a considerable ammount of technical experience and experience working as a team.

Technically, I have become extermely fluent in the microchip assembly language and most of the features and functions of the PIC 16c74A, the Xicor X25043, and the Dallas real time clock. Although I had several frustrating nights interfacing all of the different components in the monitoring unit, I feel I have learned a great deal about the different types of interfaces (parrallel, serial,and bus multiplexed). I found the most interesting technical challenge of the project was developing a system that could detect, log, and process so many real time events. I have also learned a lot about Windows programming and event handling.

I found that detailed clear documentation of a design is a vital portion of group design. During the project we developed a communications protocol (VIP) to fascilitate communications between the host and the monitoring unit. Subsequently, we generated some very detailed company documentation for the team members to refer to during development and the integration of the two systems was virtually painless. I found it is also important that when a change is necessary, to make sure everyone in your group knows about it. A few software glitches occurred in the testing of the system because of a few functions not being updated with some changes made by another group member.

Throughout this project, I think our group learned to appreciate that the simplest solution is often. If we were to repeat this project I believe we would start with our initial design and attempt to simplify the implmentation further until we felt that it couldn't be simplified any further.

As a group, I think we all learned to listen to each other's ideas and compromise with each other's work schedule. Since all of our group members took 16 or more credits this semester (including very time consuming courses like ENSC 325 and ENSC 383) as deadlines approached in other courses we found it very difficult to schedule times for our group to meet and work together. I would recommend for future ENSC 370 groups to take a slightly reduced course load and make sure that other group members have similar course schedules.

6.2 Shane Schneider

As a member of EnginuiTech these last four months, I have had the pleasure of designing and building a Remote Automated Vending Statistics (RAVS) System. My major contribution has been writing and developing the VHDL code for the EPLD, writing and verifying the high level functions of the microcontroller, and contributing with the implementation of our communications protocol in both the MU and the host. From this project, I have learned more here than my last co-op work term. I got to see a complete design coming from a basic idea we created in January, through the design process, to a working prototype in April.

From a technical standpoint, this project has increased my knowledge in hardware design. Before this project, I would look at a circuit and find it difficult to understand why all the specific chips or components were needed. Coming from a mostly software background, I found our early design work on our circuit fascinating as we would discuss our desired functionality and which chips we would need to implement. To be able to look at our design and understand why we have each component and what it does, brings great pride and increases my confidence as a design engineer.

This project also increased my software knowledge as I would write both VHDL code and complicated assembly language code for the EPLD and microcontroller to perform the various functions our design required. This project allowed me to learn and write VHDL code for our EPLD, a language I had never worked with before this semester. This VHDL knowledge will serve me well, especially in my upcoming work term where I will be expected to work with VHDL. On the microcontroller, our assembly code reached the physical limitations of program memory several times, so optimization techniques had to be learned as I reduced our utilization of the chip’s memory from 98% to 80%, while still adding more functionality to the code.

This project also allowed me to apply the documentation knowledge I learned from my last work term. Design documentation is a large component of any new design. There were several times (usually during conflicts) where we would pull out either our functional or design specifications and verify how we would implement a certain feature and when we could consider the feature completed. It was nice to see how knowledge learned on a work term, applied to a completely different design.

Other important lessons from this project did not come from design, but was related to group dynamics and how I participated. For the most part, our group worked together well. We did have differences of opinions, but we were quick to resolve. As a group, we needed to learn to listen to each other’s opinions and not to dominate the discussion with our own. Another source of conflicts would come from discussing totally different features at the same time. One particular example would be with me emphasizing how a FIFO buffer would be better to store the sales log while the rest of my group would be discussing how we should monitor the quantity levels. I needed to learn patience and after discussing the quantity sensors, we could then discuss the FIFO buffer. Both of these are lessons I needed to learn and will take with me into my career. I know I am allowed to be advocate about my opinions, but I do not need to dominate my partners. It would not be wise to argue with my boss about a solution and find out he has the same solution, or to bring to discussion a solution to a problem we are not discussing at the moment. From this project, I have also learned to listen to my partners more and to trust that they are just as capable (or even more so) with their knowledge and skill about a particular item.

This project has been very insightful and I have learned a lot in technical, documentation, and interpersonal skills. I am glad to have worked on the RAVS System and to be a part of EnginuiTech and its members.

6.3 Nestor Siu

For the RAVS system, I was involved with implementing the high level PIC microcontroller code to support the VIP protocol, as well as create underlying low level supporting modules, such as the VIP protocol processing, e-mail message support, and paging support, for the Host . This experience has given me exposure to writing micro-controller assembly code, as well as writing event driven code for Windows based applications.

More importantly, however, this experience has given me a taste on how intricate group work can be. For example, before writing any software, coding rules and guidelines, as well as naming conventions need to be decided upon before any line of code is written. This way, when the modules are written by different members, they can be more easily integrated together without running into problems such as conflicting variable names, and it can make debugging each other’s code much easier. Also, before starting a project, each of the software module’s functionality and programming interface need to be clearly defined. This way, the task of writing software can be more easily divided, and the code can be more easily re-integrated together at the end.

Ultimately, my most interesting experience for this project has been to come to a group consensus when everyone had different point of views. Fortunately, we were able to overcome this problem by rationally presenting our views and considering each other’s point of view.

6.4 Brad Oldham

The most important learning aspect of the last semester is that of the power of groups. As the project progressed, it was interesting to see how the system was integrated with minimal difficulty (most of the time). Additionally, since we had clear communication protocols between the sub-units, we had simpler integration then expected.

From a technical aspect, I expanded my knowledge of sensors and the analog circuitry that interpret them. Additionally, I learned how time consuming a simple circuit can be just do to poor solders or wiring mistakes. The coding experience that I had was with debugging code. From this experience I learned about coding for micro-controllers and the concerns that go into the communication with external components.

The core difficulty with respect to group dynamics was course conflicts. The fact that no two members had similar courses caused problems with scheduling. These scheduling problems lead to the necessity to modularize the project. There were no major inter-personal problems within the group, just high tension around deadlines and after late nights.