Project Number:RXV-IRL2

DESIGN OF A HEALTH MONITORING DEVICE

A Major Qualifying Project Report

Sponsored by the University of Limerick:

Submitted to the Faculty

of the

WORCSTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

_____

Adam Broders

Date:[_____]

_____

Dylan Fitzgerald

Date:[_____]

_____

Paul Ingemi

Date:[_____]

Approved:

_____

Professor Richard F. Vaz, Advisor

Abstract

Home medical monitoring systems allow care providers to reduce their patient load, but no available systems offer portable operation. This effectively tethers patients to a specific location. In conjunction with the University of Limerick, our team designed and implemented a proof-of-concept portable medical monitor able to transfer medical data wirelessly. Our completed project supports USB and 802.11b, includes a display and basic user interface, and runs Linux, making it a highly flexible platform for future progression toward marketability.

Executive Summary

In the Republic of Ireland and across the world, a shortage of medical care providers and increasing demand for medical care have created frustratingly long queues for service. Many patients who require frequent monitoring are greatly inconvenienced or under served. Additionally, patients who have conditions such as diabetes that require constant monitoring, but to have a physician perform this monitoring in person would be wholly unreasonable. In an effort to satisfy demand for more convenient diabetic monitoring, many in-home testing and monitoring devices have developed and evolved.

More recently, in-home diabetic testing and monitoring solutions have evolved from mere blood-sugar meters to relatively comprehensive monitoring devices, some with the ability to transmit data to a central monitoring facility over the Internet. These solutions, however, are proprietary and non-portable; purchasers are locked into a small subset of potential glucometer vendors, and their lifestyle is restricted by the requirement to periodically transfer data in their home. The University of Limerick, realizing the potential to improve the quality both of self-measurement results and of patients lives, sponsored our project to design and create a proof-of-concept next-generation monitoring device.

The simple goal of achieving wireless connectivity and portable operation could be achieved by either retrofitting these capabilities onto an existing monitoring solution or implementing the project all in software for a palmtop computer. The University of Limerick, however, wanted a flexible, reliable, future-proof design. We decided the appropriate way to achieve this goal was to build a new, modular hardware device running a freely-available general-purpose operating system. In order to complete the project within our time constraints, we used a starter kit board to build our prototype. We hold the expectation that future engineers can cull unnecessary functionality and greatly enhance and miniaturize our design.

We ran the development board on a custom built 2.6.14 Linux kernel and root file system. The root file system contains our application code which implements a command-oriented GUI, software to download information from FreeStyle Mini glucometers, and securely wirelessly transfer that data to a central database via HTTPS POST queries.

The hardware starter kit we used already included facilities for data storage, battery operation, serial communications with glucometers, and host-side USB. In order to meet our requirements we needed to add an LCD display, a user interface, a battery, and wireless networking. We also performed some design work for client-side USB, which would allow us to easily update the device's firmware in the field. We were unable to realize our design because we did not have the required board construction tools available to us.

The Sony ACX705AKM-7 LCD we chose was electrically compatible with our development board, but required a +3.8VDC source, which the board doesn't provide. In order to efficiently provide this voltage, we used the National Semiconductor LM2621 DC-DC boost converter driven off our +3V line. This system runs at about 80% efficiency, and fits on the same PCB as our other LCD circuitry. We modified the Linux driver for our processor's built-in LCD controller to generate a 3 MHz clock signal with the appropriate synchronization pulses.

To implement a user interface, we added four pushbuttons to the system on the backside of our LCD PCB. The buttons were connected to GPIO pins on our CPU, with pull-up resistors to +3V and a direct connection to ground on button depression. We enhanced an existing Linux driver to make the buttons act like keyboard buttons.

In the interest of speeding device development, we used commodity hardware to implement wireless networking. Since the development kit already provided a USB host port, we purchased a consumer 802.11g USB network adapter and plugged it in. We cross compiled the open source Linux driver to run on the board. In future device revisions, this implementation could be reduced to a single-chip and antenna solution, significantly lowering per-unit cost.

We used Duracell alkaline batteries to demonstrate battery operation due to immediate availability. Even in our simplistic design of three AA batteries wired in series, based on an

experimentally obtained power consumption measurement at a minimum load of 300 and a maximum load of 600 mA, we expect the batteries to last between 2.5 and 5 hours.

While not a production-ready product, our proof-of-concept device clearly demonstrates the possibilities for next-generation home monitoring solutions. Future engineers can build off our research, device selection, and code base to arrive at a market-ready product that can free diabetic patients from their homes and care providers from their queues.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Without the contributions of the following individuals, our project could never have been what it became.

Professor Richard F. Vaz

Dr. Mark Southern

Seamus Clifford

Professor Eamonn Murphy

Professor Eamonn McQuade

John Harris

Charlotte Tuohy

Professor John Nolan

Claire Ryan

1 Introduction

Medical professionals need the capability to assess the health of their patients and deliver feedback without requiring time-consuming outpatient services. Care providers may have hundreds or even thousands of patients to manage, presenting a tremendous time burden. Under these conditions, patients with medical conditions requiring tight monitoring and control, such as diabetics, may not receive the attention they need. To address the problem, health monitoring can be decentralized and brought closer to the patients. This enables patients to manage their own health while allowing them, at their discretion, to receive expert advice remotely. With self-monitoring, patients are freed from the burden of structuring their lives around a physician-mandated schedule, and physicians can effectively manage a greater patient load.

Many vendors offer medical devices for home patient testing and monitoring with data logging capabilities to help patients manage their own health. Unfortunately, many of these devices have limited processing power and storage capacity. This forces the user to rely on personal computers or pen-and-paper journals to log and analyze their long term health variations. As an alternative to manual logging, separate, dedicated devices known as personal health monitors are available. These personal health monitors interface with many self-testing devices to provide long term data storage. They also facilitate transmission of this data to the appropriate care providers.

The goal of this project was to extend the concept of personal health monitoring beyond the current state of the art, specifically catering to the needs of the Irish health care industry. Technologically, this is achieved by the addition of such features as wireless data transfer, centralized data storage, portable form factor, battery operation, and two-way communication over said wireless data link. However, technology is but a means to an end; we want to create a tool that reduces the need for time-consuming doctor-patient visits and empower those patients to help themselves, yet give them the expert support they need.

The presented design is a proof of concept implementation focusing on use of glucometer sensors. This focus allows physicians affiliated with our project to evaluate the usefulness of the device without requiring excessive development time. If the result is marketable, the feature set can then be expanded or culled down to the most market-friendly options, and the programming can be expanded to handle other medical conditions.

2 Background

In this chapter, we present first a general overview of diabetes mellitus in the interest of introducing the reader to the wants and needs of a typical patient. Following this, we review personal health monitoring options that already exist for diabetics.

2.1 Diabetes Mellitus

Diabetes Mellitus is a chronic, progressive illness where one's body is subject to excessively high levels of glucose in the blood. This condition is caused by problems with the pancreas, which in turn limit the body's ability to break down and use carbohydrates.

Diabetics fall into one of several categories.

Type I diabetes, or early-onset diabetes, usually occurs when the body's immune system attacks the cells in the pancreas that produce insulin, utterly destroying the body's ability to burn carbohydrates. This condition usually occurs in children and young adults, and is incurable. Type I diabetics must use injectable insulin to survive. [UMaryland Medical:1] Type II diabetes, or late-onset diabetes, is less severe. In Type II diabetics, insulin is still produced, but it either doesn't work as well or isn't produced in sufficient quantities. Type II diabetes, like Type I diabetes, is incurable, and can cause severe complications leading to death if left untreated. [nlm.nih.gov:1]

Two other conditions fall under the umbrella of diabetes: pre-diabetes and gestational diabetes. In pre-diabetes, blood glucose levels are elevated to the point where they could be causing permanent damage, but are not high enough to be diagnosed as diabetes. With proper lifestyle changes, pre-diabetics can delay or entirely prevent a full case of Type II diabetes. [diabetes.org:6] Gestational diabetes is a Type II-like diabetes that develops during pregnancy. While greater research is needed to precisely name the cause, the American Diabetes Association suggests that gestational diabetes occurs when the additional demands of supporting a baby are too great for a mother's body to handle, and blood sugar levels rise to dangerous levels. Women with gestational diabetes need to treat the condition like “normal” diabetes, possibly including insulin injections and blood-glucose level monitoring. Fortunately, almost all cases of gestational diabetes end altogether when a pregnancy is over[diabetes.org:5].

The initial symptoms of diabetes, especially during the onset of Type II diabetes, are relatively mild; the Diabetes Federation of Ireland estimates that as many as half of diabetics in Ireland are undiagnosed. These symptoms include excessive hunger and/or thirst, irritability, unexpected weight loss, fatigue, vision problems, and frequent urination[diabetes.org:1].

Diabetes can strike any individual, but some people are at higher risk of developing the disease. Persons of non-white ethnicity and the elderly are statistically of increased vulnerability to diabetes. Additionally, persons leading an inactive lifestyle or who are overweight seem to overstress their pancreas, greatly increasing their risk of diabetes. Due to the increasing levels of obesity and inactivity, diabetes cases are expected to increase significantly in the near future [Healthy People 2010:1].

Diabetes is an expensive disease. In Ireland in 2002, €350 million, or about 10% of the health care budget, was spent on diabetes alone. However, even in light of the expected increase in diagnoses, this cost could potentially be reduced; again in 2002, about 60% of this expenditure was applied toward addressing complications [Healthy People 2010:1]. If the disease were better managed by patients, this cost could fall.

2.1.1 Complications and Hazards

The consequences of self-treatment mistakes by diabetics on insulin can be severe. One of the most dangerous mistakes is overdosing on insulin, such as by injecting the morning dose in the evening. In these cases the patient becomes hypoglycemic, or dropped to dangerously low levels [Texas Children's:1]. If left untreated, the patient could lose consciousness or even die.

Diabetics are subject to a number of severe complications, especially if the disease is

poorly managed. These can include damage to the heart, kidneys, eyes, skin, or nervous system, and can escalate fatally if untreated. Additionally, diabetics are especially susceptible to depression. Therefore blood glucose levels must be closely watched and controlled to limit diabetic complications [diabetesireland.ie:1].

2.1.2 Blood-sugar measurement

Currently, the only reliable way to measure blood-sugar levels involves invasive testing. Diabetic patients must stick themselves, usually in the finger, to obtain a blood sample. The sample is placed on a test strip or directly obtained by a lancet, and then analyzed by a blood-glucose meter. Based on the result of the test, the patient or care provider can then either verify that blood-sugar levels are acceptable, or take corrective action.

A large variety of glucometers are available on the market, differentiated by features such as level of invasiveness, size, ease-of-use, cost, memory, and compatibility with other devices. Some of these features, such as invasiveness and size, are largely driven by the evolution of glucose monitoring technology towards a robust test that requires a minimum amount of blood. The others are matters of preferences and cost.

Some glucometers have RS232 or USB interfaces to extract readings. Others have proprietary data connectors. Typically the manufacturer produces data cables to connect to USB or RS232. Yet others in the low end of the market do not provide a data port.

Many glucometers have built-in memory and other extra features. The Ascensia Elite XL is an example of a glucometer with built-in memory capable of storing 120 measurements. Another, the FreeStyle Flash 4, can be programmed with up to four alarms.

In addition, for those glucometers which have data cables, some manufacturers provide data journaling software for PCs or their own data storage devices. These software and device systems tend to be proprietary in nature and do not interoperate with other vendor's sensors. An example of this, the FreeStyle CoPilot, further discussed in Section 2.2.3, only interoperates with Abbott Laboratories' sensors.

2.1.3 Blood-sugar control

The key to diabetes management is tight control of blood glucose levels. Depending on the type and severity of the case, patients and doctors may select from a variety of medications and habitual adjustments to suit each individual case.

The simplest solution to lowered insulin levels is to supplement or replace the insulin directly. Many brands, strengths, and time-release varieties of injectable insulin are available, and combinations can be tailored to the individual patient. For Type I diabetics, insulin injections are a fact of life; without it they will die. Type II diabetics may or may not require insulin depending on the severity of their case. Type II diabetics who do not normally require insulin may require it when sick or in an emergency situation [diabetesireland.ie:2].

For added lifestyle flexibility, many diabetics use insulin pumps rather than syringes. An insulin pump delivers small doses of insulin continuously, allowing the patient to tailor their insulin dose to their current activities. Pumps are most commonly used by people with Type I diabetes due to their utter dependence on insulin, but Type II diabetics can also use a pump [diabetes.org:2].

Type II diabetics can sometimes manage their diabetes with the assistance of diabetes pills, which come in several varieties. Some of these can be prescribed in combination to achieve the desired effect. All such pills lower blood-sugar levels. Pills are not appropriate therapy for Type I diabetics, as they will not eliminate insulin dependence [diabetesireland.ie:3][diabetes.org:3].

One of the most effective ways to manage diabetes has nothing to do with medication: habit adjustment. By losing weight, regulating one's diet, quitting smoking, or regularly exercising, diabetics can significantly improve their condition and prevent

complications. Type II diabetics may find that these modifications improve their condition to the point at which medication is unnecessary [diabetes.org:4].

2.2 Medical Telemetry

There are several systems that help diabetes patients review their glucose history include Tadiran Spectralink Ltd's MDKeeper, Health Hero Network's Health Buddy, Abbott Laboratories' CoPilot, Niklas Andersson's Prototype Cell Phone Application, NASA's CPOD, and Cybernet Medical's MedStar. These systems are compared in *Table 1: Comparison of Medical Health Monitors*. Note that despite the wide variety of form factors, no device contains all the functionality needed to have a truly flexible portable device with support for a wide, vendor-independant variety of sensors.

Table 1: Comparison of Medical Health Monitors

Device	Wireless	Battery	Sensors	Form Factor	Two-way communication	Memory
MDKeeper	No	Yes	Built-in	Watch	No	unknown
Health Buddy	No	No	USB/RS232 Sensors	Small Base Station	Yes	64MB
CoPilot	Limited Range*	Yes*	Abbott Laboratories Sensors	PC Software	Yes	Effectively Unlimited
Unnamed Prototype Java Cell Phone Application	Yes	Yes	Bluetooth® Sensors	Downloadable Cell Phone Software	Yes	Cell Phone Memory
CPOD	Limited Range	Yes	Specifically Supported Sensors	Small Chest Device	No	8 Hours Logging
MedStar	No	Yes	Cybernet Medical Sensors	Small Base Station	No	"hundreds of measurements"

** The CoPilot has the wireless and battery capabilities of the host Windows PC*

2.2.1 Tadiran Spectralink Ltd. MDKeeper

The MDKeeper is a life monitoring system in a watch form-factor that incorporates sensors such as an ECG to monitor the wearer's vital signs [Siemens/BenQ:1]. When the MDKeeper detects that the person requires medical attention, it uses an internal cell phone (GSM/GPRM) to alert a "medical center." Siemens states that it can also use this link to upload recorded sensor readings.

2.2.2 Health Hero Network's Health Buddy

The Health Buddy is an ARM-powered, Linux-based device with a bright color LCD display that accepts up to 4 USB and 1 RJ45 medical sensors, can store up to 64MB of sensor data, and has a built-in modem to transmit recorded data to a health care provider [Health Hero:1]. It also has built in programs for "over 50 health management programs including NCQA certified: Hypertension, Chronic Obstructive Pulmonary Disease, Diabetes and Heart Failure." [Health Hero:1]

2.2.3 Abbott Laboratories FreeStyle CoPilot

Abbott Laboratories sells a data cable and CoPilot software to work with their FreeStyle glucose meter product [Freestyle]. On their website, they advertise the software as a free data acquisition, sharing, and reporting solution for Windows computers. They also offer their own central data warehouse. The data cable is available separately for $20USD. While the feature set of the CoPilot is extensive, it lacks support for sensors from other manufacturers and requires a Windows PC.

2.2.4 Niklas Andersson's Prototype for Transmission of Glucometer Data by Wireless Technology

Andersson's research paper describes a Bluetooth® capable cell phone loaded with

Java software designed to interrogate Bluetooth® glucometers within range and transmit their readings using cell data services [Andersson]. This technique is clearly portable, wireless, and extensible to other sensor types. Furthermore, it leverages existing hardware for novel ends. Unfortunately It is not useful for sensors that are not Bluetooth® capable and may be expensive given the need for cellular data services.

2.2.5 NASA's CPOD (Developed by Stanford's Lifeguard project)

News-Medical.net describes NASA's wireless health monitoring system for Astronauts as "...a compact, portable, wearable device -- a single piece of equipment that gathers a wide variety of vital signs [Temperature, blood pressure, ECG, and O2]. About the size of a computer mouse, a CPOD is worn around the waist. It's comfortable enough to be worn while sleeping. It's non-invasive. It takes only minutes to don. Importantly, it can track a person's physiologic functioning as they go about their normal routine -- they don't have to be tethered to some stationary device. It can store data for eight-hour periods for later downloading; alternatively, it can send it wirelessly, in real time, to some other device." [News-Medical] Interestingly, the CPOD appears to integrate sensor and monitor functionality, handing off more intensive monitoring and analysis to a separate device.

2.2.6 Cybernet Medical's MedStar

The Medstar is a small hand-held device that records medical data from Cybernet Medical sensors, including a blood pressure sensor, a weight scale, a glucose meter, and a pulse oximeter. These sensors are directly connected to the device which then sends the data over a telephone line to Cybernet Medical's central server. [cybernetmedical.com] Cybernet Medical provides a subscription web-based management and monitoring interface. The Cybernet Medical approach appears to be a complete, if closed, solution. Note that the closed nature of the Cybernet Medical system restricts the hardware that may be used to those directly and specifically supported by Cybernet Medical.

3 Project Overview

As described in the introduction our product was to be a portable medical telemetry device designed for European use. Mark Southern, our University of Limerick project leader, in conjunction with Prof. John Nolan, the affiliated physician, specified design suggestions for the device that we have examined, edited, and elaborated on according to our research and limitations. From this list we have created a basic design that we, throughout the project, have been implementing piece by piece.

3.1 Device Requirements

Mark Southern, our University of Limerick project leader, in conjunction with Prof. John Nolan, the affiliated physician, specified approximate requirements for the device that we have examined, edited, and elaborated on as per our research and limitations.

- **Portable**

  Most currently available devices are designed to stay in one location. This implies that the user must leave the device behind for a potentially lengthy period whenever they travel. Making our device portable allows a patient to live their lives without having to worry about frequent home visits for monitoring. To meet this requirement, the device must be battery-powered, physically small – perhaps emulating the form factor of the Palm or PocketPC organizers – manage power appropriately to avoid overtaxing the battery, and have wireless communication capability.

- **Built-in Programmable ID for user identification**

- **Potential for Broad Sensor Compatibility**

  Our design must have the capacity to handle different forms, types, and brands of sensors. This creates a large potential customer base, and allows purchasers of new sensors to choose from a wide variety of glucometers. This implies an additional requirement for multiple hardware options for interfacing to different glucometers; in designing our proof of concept device, we concentrated on USB and RS-232 communications.

- **Built-in Clock**

- **Large Data Buffer (At least 1 megabyte)**

- **Secure Data Sharing**

  Secure data transmission and storage is critical for legal reasons. Generally speaking, medical data cannot be released to individuals or organizations without the patient's explicit approval in writing. Thus, the device must be designed to protect, within the bounds of processing power the patient's data from inadvertent or maliciously unapproved distribution.

- **Secure Off-site Storage**

  The need for secure data transmission and storage is detailed above.

- **Well-designed User Interface**

The range of patients that this product will be marketed to are of widely varied technical ability, making a well designed user interface critical. If only a small portion of potential users can understand and use the interface, not only will it limit market acceptance of the device -- it may also mar the public perception of future devices.

- **Firmware Upgradeable**

  A device such as ours emphasizing wide compatibility with many sensor devices will absolutely need to have its firmware updated and appended to add new support for new devices as they come to market.

- **Feedback Capability**

  In order to reduce the need for outpatient services, our device must not only allow care specialists to access appropriate medical data collected in the home, but also allow those specialists to have at least limited communications with the patient.

- **Flexible End-user Setup**

3.2 Device Recommendations

In addition, we found that other requirements both given to us in the initial specification and unearthed in our research were less requirements and more considerations for future product development. Given time, these considerations could be implemented, provided that all essential product requirements were already met.

1) Clock auto-calibration
2) Physical System Expandability (Physically 25% Scalable)
3) Low Price Point (targeting $150)
4) Pregnancy Management
5) Storage/accessibility of Essential Patient Information
   1. Blood Glucose Level History
   2. A1C History
   3. Current risk factors for complications
   4. Medicine list, history, scheduling
6) Pediatric Use

The sponsors of this project see device functionality, followed by abundance of features, as the most important requirements. So long as the finished device can demonstrate features that justify further development and marketing, it will be successful in their eyes. In contrast, an extremely well designed device that doesn't actually work is a failure.

3.3 Overall System Feature Design

Based on the customer's feature requirements, we've built up a list of systems implementing the features necessary to satisfy the requirements. These features are listed in the following tables and a block diagram of how these systems interact is shown in Figure 1. Those features slated for future implementation are marked as such.

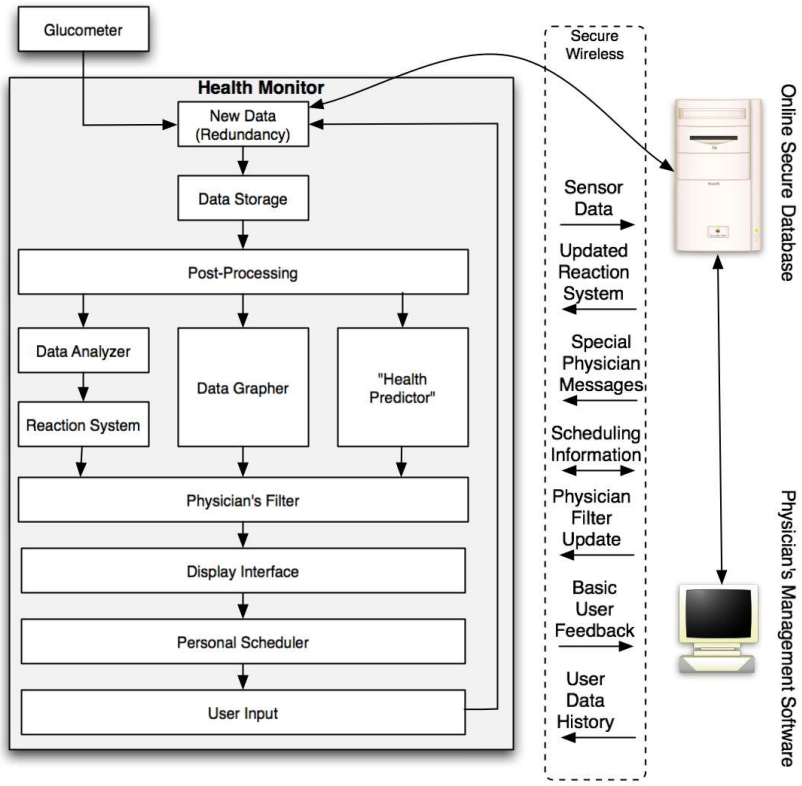


Figure 1: System Interaction

Health Monitor: This is the physical unit to monitor the patient's health.
New Data (Redundancy): New data from user input or a connected glucometer is stored until the unit is moved in range of a wireless access point. After confirmation of data transmission to the secure online database, this data will be moved to normal data storage, or flushed as necessary.
Data Storage: Stored medical history of this particular patient. This is kept full at all times to limit connection with the online secure database when analyzing the individual's history.
Data Grapher: The user may request a graphical representation of data. We are currently looking into appropriate graphing methods. Current ideas include two week, two day, pattern-recognizing, and user configurable graphs. A pattern recognizing graphs would show the strongest repetition of the individuals glucometer readings.
Data Analyzer: Immediately upon data transfer, algorithms configured by the physician search through the data looking for issues to immediately notify the reaction system. [Future implementation]
Reaction System: When the data analyzer calls this system, it will in turn alert the patient or care provider as necessary, depending on the algorithm's alert settings. [Future implementation]
"Health Predictor": Some kind of system to allow an approximation of the patient's current condition. This could include such things as recognizing pattern issues, noting particularly bad test times, or even giving a "percentage from optimum" health meter. All of these are under consideration at the moment. [Future implementation]
Physician's Filter: We anticipate that some patients may have issues in addition to diabetes that would make the full presentation of data unwise. This filter will allow the physician to control what data the patient may access. What should be restricted is still under debate. [Future implementation]
User Input: Some glucometers may not be able to directly connect to the health monitor. This would allow the user to directly input the data in these cases. If necessary, data points entered in this manner could be marked as human-input.
Personal Scheduler: A planner system with flexible, configurable reminders and appointments. This is much like what you would have on a palmtop computer's scheduler. This can assist patients who want help remembering when to take measurements, or to track physician appointments. [Future implementation]
Display Interface: A screen to display feedback, notices, and other pertinent information.

Online Secure Database: This is simply a safe storage unit for all user data. This is a single system that all of a physician's or hospital's patients will store their data on. [Partial future implementation; some database implementation occurred for this project]

Physician's Management Software: Software for the physician to manage the online secure database, view graphs, and do other analysis as required. This portion of the project is only in theoretical stages. [Future implementation]

Secure Wireless: A wireless communication medium to the secure database. This also implies a connection to the physician's notices, comments, or any other communication seen below. The options listed below would only be accessible when connected to a wireless network.
Sensor Data: New sensor data is transferred over this medium for safe, secure storage on the database.
Updated Reaction Systems: Update the device's data reaction system based on the physician's preferences. [Future implementation]
Special Physician Messages: Messages from the physician would be displayed for the user to read on the health monitor. If a reply is required it will be sent back as basic user feedback. Multiple-choice questions would be ideal. [Future implementation]
Scheduling Information: Transfer scheduling information between the physician and patient to facilitate an appointment. [Future implementation]
Physician Filter Update: Update the device's data filter system based on the physician's preferences. [Future implementation]
Basic User Feedback: Return a reply from a special physician message. [Future implementation]
User Data History: Allows access to patients measurement history for data analysis or graph interrogation. [Future implementation]

The primary data path illustrated in first receives new data, places it in a queue, displays it to the user, and, when a wireless connection is available, uploads it to a secure online database. The other data paths implement additional features such as graph generation, pattern analysis, and health analysis. Access to this functionality is further filtered by the physician's settings and then displayed for the user. Also illustrated is use of a personal scheduler to help the user organize their daily measurements, periodic physician's appointments, or other related temporal events. Note also the variety of data flowing between doctor and patient, or server and device.

3.4 Possible Designs

There are six overall approaches to meeting the product requirements that we considered. In general, design approaches were evaluated in terms of their feasibility, how well they would satisfy the project requirements, expandability, and system cost. Generally speaking, feasibility was the most important factor, followed by the other three in respective order. Along with these factors, we also considered the ease of troubleshooting the design, the risk of overrunning the time allotted, and the additional capabilities each approach provides beyond the requirements.

3.4.1 PCI-Based Approach

In this design, a microprocessor is connected to a PCI interface, which in turn is used to connect to peripherals such as USB, wireless networking, and video output. These devices would be controlled by a small Linux operating system running on the microprocessor. The operating system in turn runs our software and manages the hardware.

Pros

PCI based computers are tried-and-true technology that operate reliably with the GNU/Linux operating system. This configuration would save time because the operating system would already contain drivers for the hardware we use. Furthermore, there is a large supply of cheap PCI based peripherals such as Bluetooth® chipsets.

Cons

The major drawback of this design is the complexity of the system. The PCI bus

requires 32 traces as well as power and ground. Additionally, if a PCI controller is not built into the microprocessor, then system would require an external PCI controllers simply to control the bus.

System Summary

This solution is well supported by the hardware and software industry. However, the approach may require significant extra work during PCB layout.

3.4.2 USB-Based Approach

This approach is similar to the prior one, except we directly interface the microprocessor to the USB controller or USB bus, and then run wireless and other peripherals off of the USB bus. These peripherals can be separate from the device itself and plugged in or removed as warranted.

Pros

By eliminating the PCI bus, system complexity is reduced. The use of hot-pluggable USB wireless devices, while initially more expensive than their PCI-based counterparts, allows the device to be more modular and have a smaller base system while being easily user-upgradeable with newer USB devices.

Cons

This system relies on third party hardware which may or may not provide power management. We can exert some control over this by turning off devices that are not currently required, but power efficiency may not be maximized. Also, since third-party non-commodity hardware is used for all devices, the initial investment in all parts is higher.

System Summary

This system is expandable, reliable, customizable, and less complex than the system using the PCI bus.

3.4.3 8-bit Microcontroller

All hardware is directly linked and controlled by a low-end microcontroller. The microcontroller is programmed without a conventional general-purpose operating system.

Pros

This approach doesn't require expensive components such as a PCI bus or a microprocessor capable of running a general-purpose OS.

Cons

This configuration requires an incredible amount of programmer and designer time, and expertise in all areas of the device design, in order just to get the basic hardware operational. This is because it would be difficult to utilize existing drivers for USB and Bluetooth®, or other peripherals needed by the device. This would necessitate custom development of multiple drivers from scratch. This would require a thorough, pin-by-pin, signal-by-signal understanding of every component in the system. This is unreasonable for a ten week project. Furthermore, extending the device to talk with many different sensors would require a larger time investment than in other approaches.

System Summary

This design is impractical given the time requirements. It would be more appropriate in a long term project to develop an extremely cheap, low-power version of this device.

3.4.4 PocketPC

Purchase an appropriate PocketPC or compatible device, and write software to run the features we require.

Pros

This system is easily realizable. The hardware is all currently designed and constructed for us. All that remains is writing the application software.

Cons

This system irrevocably relies on third party hardware. While this is convenient in the design stage, it could have adverse effects on the product's lifespan. Vendor lock-in is highly plausible, causing price hikes on hardware and providing a significant risk should the vendor discontinue the supporting product. If we try to avoid this problem by targeting the generic PocketPC rather than a specific model, then inevitably we will encounter either a lack of required hardware such as a serial port, or newer pocket PCs with slight hardware changes causing unreliable behavior. Unreliable behavior is unacceptable for a medical device.

System Summary

This device would initially be quick to develop and rapid to deploy. It is sensitive to third party product price hikes and hardware unavailability. Additionally, limiting this sensitivity with generic PocketPC support may lead to unpredictable behavior.

3.4.5 FPGA

Design the entire system on a FPGA chip by linking together pre-existing licensable processor and peripheral cores.

Pros

This device is physically very simple and compact, requires minimal physical hardware work, is programmable for quick hardware debug situations, and is very inexpensive in quantity.

Cons

FPGAs are expensive in small quantities, and licensing cores can be prohibitive. We also do not have experience with large-scale FPGA projects.

System Summary

Although a good creative idea, this system is not plausible because of our cost and talents. Otherwise, this would be an excellent solution allowing us to almost completely control the hardware and reconfigure it during firmware updates. The only limits would be the number of pins we have available and the requirement for analog systems for certain hardware.

3.4.6 Cell Phone

Write a Java applet targeting cell phones and use the phone's resources to perform the required tasks.

Pros

Cellular phones are relatively ubiquitous, so this approach would greatly aid in concealing the user's condition from other people. This design is also small, compact, and familiar. Additionally, since we can assume that the user would only consider purchasing a mobile that they can operate with whatever physical disabilities they may have, we need not worry overmuch about physical accessibility.

Cons

This system has a large startup cost, namely, a Java and Bluetooth® capable cell-phone for every user. Additionally, since data transfer would likely occur over GPRS, recurring costs would be high. Unfortunately the market for cell phones is highly variable and as with the pocket PC we would be forced to restrict the development to a single cell phone or be at the mercy of third party cell phone hardware issues.

System Summary

This solution is much like the pocket PC but with even more issues dealing with cost and an increased variability of host systems. The cost of deployment was the main reason this solution was eliminated from consideration.

3.4.7 System Comparison

We compared the systems as shown in *Table 2: Weighted design matrix for possible design solutions*. Note the row of per-column weights at the top. The USB based approach has the highest score, followed by the PCI-based approach. These results helped us narrow our list of usable approaches. No matter what approach we chose, an eventual market-oriented device would require a great deal of functionality that will not be implementable in the time we have available. As such, we are using the USB-based approach, because it does in fact have a low cost, does in fact have a large amount of functionality, and has a lot of components already built for us. The only other systems that would compare are the PocketPC-based and PCI-based approaches, as either would reach many of the final goals, allow a good deal of functionality, and could cost comparatively less for a one-shot device in both situations. Unfortunately the PocketPC systems are inappropriate for use as a medical device, owing to potential for vendor lock-in or hardware variations. The PCI approach could provide functionality nearly identical to the USB approach, but would require more direct hardware work and construction. In retrospect, our attention to this detail proved fortunate.

As a result of our analysis we chose the USB system.

Table 2: Weighted design matrix for possible design solutions

	Cost	Time	Exceeds Req. By	Time risk	Debugability	Expandability	Functionality	Additional Content	Totals
Weight	3	8	10	6	5	5	5	4	
PCI	4	8	10	6	5	8	8	10	300
USB	5	0	5	0	8	5	5	5	357
8-bit MCU	3	10	4	7	10	4	4	5	175
PocketPC	2	8	6	7	7	8	5	6	281
FPGA	3	8	6	7	8	4	3	3	296
Cell Phone	3	8	10	6	5	5	5	4	262

4 Electrical System Design

In this section, we examine the critical design choices involved in processor and hardware selection. We discuss in detail hardware aspects of the device, especially the LCD, USB, and power systems.

4.1 Critical Design Choices

The overall system design was primarily dictated by a few critical decisions: the microprocessor and the development board.

4.1.1 Microprocessor Selection

We used an iterative refinement process to select our microprocessor. Initially, we compiled a collection of systems based on Linux support. Following this, we conducted in-depth research for each microprocessor to assess their capabilities. The critical features that swayed our decision are listed in Table 3. Black indicates a major road block to progress, and gray indicates a concern.

Microprocessor	Memory	USB	Video	Speed	Power
SH7727	MMU, DMA	OHCI (needs client controller supported by linux)	LCD cntrlr	160MHz, 208MIPS	650 -> 190 mA @ 3.3V
Sharp LH79520	SRAM	No (needs client controller supported by linux)	LCD cntrlr	?	43.5 -> 3.9 mA @ 3.6V
OMAP 5912	SDRAM ctrl	OTG supported	LCD cntrlr	?	"Has power management"
S3C24x0	SDRAM ctrl	Device+Host	LCD cntrlr	280MHz	? (seems like it would be low)
AT91RM9200	SDRAM ctrl	Device+Host	No	180MHz	+-35 -> 1.44 mA @ 3.3v

Table 3: Microprocessor Choice Chart

We required both USB host and client subsystems. The first two chips on the list, the SH7727 and the LH79520, were both eliminated because we could not locate a USB client controller without a PCI interface requirement. Also, our research revealed that the SH7727's power consumption was considerably higher than the other systems available.

From here we had to choose the best solution as all of the last three met our major requirements. We chose to pass over the AT91RM9200 as it had no built-in video driver. We preferred using a Linux supported, pre-built single-chip solution.

The OMAP 5912 and the S3C24A0 were the only two qualified systems remaining. Either could have satisfied our requirements. Both claimed to have good power management and both had the required peripherals with expansion available for the future. We chose the OMAP 5912 because, while the S3C24A0 claimed Linux support on the manufacturer's website, the OMAP had an existing Linux development community to demonstrate support.

4.1.2 TMD OMAP5912 OSK Development Board

Our development time was greatly limited, and as such the construction of a single PCB with all systems designed from scratch around the OMAP processor was an unrealistic goal, owing mainly to the ball grid array package used by the processor. In order to address this issue for our proof-of-concept device, we found and acquired an inexpensive project starter board, the OMAP5912OSK. Based on the OMAP 5912 processor and with most of the components we needed to start running and testing the basics of the system on-board, the OSK turned out to be a wise acquisition. Fortunately, the Linux community also supported the development board. For full functionality, we still needed to add a display and USB subsystems, and ideally a battery and power management system.

Once we acquired the OSK, our development process split into several separate strands. Software could immediately be tested and run on the board and slowly updated as more parts became available, while external hardware support boards were designed and constructed separately. In terms of hardware, we decided that this shifted our focus to getting basic and extended functionality on a known chip architecture, and leaving to future engineers much of the hardware design process for streamlining, miniaturizing, and otherwise modifying our product to make it suitable for the market. More to the point, we would be leaving the system with the University of Limerick with schematics for external

LCD and USB OTG[1] systems, but leave development of a single-board solution for a future project team. With this in mind, we followed up the prototype with a set of design recommendations that could guide the next stage in product development.

4.2 Power System

One of the primary requirements for our design was portability. In order to operate portably, the device must have some sort of self-contained power supply -- in this case a battery. The limited capacity of a battery, relatively high cost per capacity increase, and inverse relationship between battery capacity and portability means that we needed to engineer Bridge-IT for low power consumption. In this power system section we outline our expected power use, how the battery system itself could be constructed, and the ways in which we conserved power for the system.

4.2.1 Power Budget Balancing

The power budget for the OMAP 5912 OSK specifies typical and maximum current drains for various power sources. The USB V_{BUS} source may supply a maximum of 440 mA, but specifies optimum operation at 250 mA. The DC power source on the expansion slots specifies a maximum load of 500 mA, and the 3.3 V source on the expansion slots to a 125 mA maximum. Since the 500 mA line on the expansion slot and the 250 mA line on the USB V_{BUS} are from the same source, and the USB system is built to handle all of its current with its own cutoff system, all 750 mA of current can be grouped together and passed to the USB V_{BUS} interface.

The idea was to power three separate devices from these power sources: the USB V_{BUS}, the USB OTG transceiver with associated components, and the LCD subsystem. The USB V_{BUS} output requires up to 440 mA at 5 volts, the USB OTG transceiver system requires 35 mA at 3.3 volts, and the LCD subsystem will never exceed 40 mA at 3.3 volts.

The USB V_{BUS} line takes its power directly from the battery and as such will be converting voltages as low as 3.3 volts into 5 volts at up to 440 mA. This is 2.2 watts of power. At 3.3 volts this is up to 667 mA of current.

The 750 mA from the battery is exclusively powering the USB V_{BUS} power pump. By simple arithmetic, one can see that there is an extra 83 mA of extra current sourcing ability on the DC power line, leaving room for the USB V_{BUS} power pump's inefficiencies.

Additionally we will be using some of the current from the 3.3 volt line on the expansion slot. We are limited to 125 mA to power the LCD panel and the USB OTG transceiver system. We have budgeted 40mA at 3.3V for the LCD subsystem, and the USB OTG transceiver circuit requires no more than 35 mA on the same 3.3 volts. Thus, our maximum power requirement on this supply line is 75mA, leaving 50 mA remaining for future expansion.

4.2.2 Battery System

In order to design a battery operated device, we followed a step by step process outlined by the OMAP5912 OSK documentation. We calculated estimated power consumption based on initial specifications from the development board specification, then made and adjustments for our expected use of the board and additional peripherals. Based on these calculations, our team created a list of possible batteries.

Unfortunately time constraints prevented the physical addition of a battery to the device. To assist future designers, we have conducted battery research, and below provide recommendations and considerations for the selection of a battery. In addition, we below provide instructions on how to add a battery to the device.

Bridge-IT draws a maximum of 2500 mA in its final form as it stands connected to the development board. By referencing this specification, we can pick a battery by selecting

1 On-The-Go

the smallest Lithium Ion or Lithium Ion Polymer battery that can support such a current. Lithium-based chemistries are given exclusive consideration because the OMAP5912's companion power chip, the TPS65010, has support for power management exclusively of Lithium chemistry cells. Cell capacity within the range of 2500 mAh to 1250 mAh can be considered as reasonable due to the typical (if conservative) 1C to 2C maximum discharge rate of Lithium chemistry batteries. To ensure the selected battery works to specification, simply multiply the discharge (C) rate by the capacity (mAh) of the battery to get the maximum discharge rating (mA):

$$mA[max] = C \cdot mAh$$

Equation 1: Maximum Discharge Rating

We found that the system typically uses 440 mA at 4.5 volts. This translates to approximately 600 mA of current from a battery at 3.3 volts, assuming no power loss from the voltage conversion. Using the batteries described above, 5 hours of continuous runtime is not unrealistic, even without using any power management beyond Linux-controlled defaults. Implementing appropriate power management would further extend this runtime.

For testing purposes, three AA Duracell batteries were attached directly to the DC power input lines. This allows testing of the system on lower than 5VDC input. We discovered that this did not affect system performance or operation. This test led us to expect that the system would operate normally with a a Lithium Ion or Lithium Polymer battery voltage of 3.3 volts, as long as the V_{BUS} system is upgraded to handle proper USB power requirements.

To attach a Lithium Ion or Lithium Polymer battery, a few modifications must be made to the OSK board, and a few components must be added. Initially, resistors R79, R69, and R71 must be removed. To modify the rate of charge, R74 can be replaced with a resistor between 825Ω and 8250Ω; a smaller resistance will increase the charge current. The CHCONFIG register must then be modified in the TPS65010 to accept such modifications. Pin two and three from JP2 must be connected the battery's positive terminal, while pin four must be attached to the negative terminal. Finally, a 103AT thermistor must have one lead connected to ground through pin four of JP4, the other lead connected to pin three of the same connector, and the thermistor's casing itself attached firmly to the side housing of the battery. With these changes in place, the system should run directly off the battery, and charge it when connected to an external power source.

4.2.3 Power Conservation

Since this system is running off a battery of limited capacity, strict power management should be implemented. In order to reduce the power usage, a few adjustments and controls can be made. We have already created a system to manage the USB V_{BUS} power pump and have a few recommendations on other controllable systems.

One of the largest single power users in the design is the USB V_{BUS} power pump. At any time, a USB port may source up to 440mA at 5V. This is equivalent to a maximum of 667 mA directly from a single Lithium cell at its cutoff voltage of about 3V. Since two of these ports exist, the potential load on the battery is doubled. To ease this situation, we decided to power only one USB device at a time. Additionally, idle USB devices are turned off until needed again, and wireless access via USB is only activated occasionally according to user-configurable settings. While this system is implemented fully in hardware, unfortunately time constraints prevented its realization in software.

In future versions we recommend all parts of the system conserve their power use. Similar to the operation of the USB wireless, systems should be powered only when their operation is necessary. There is no need to run such devices when they are not performing an important task. For example, the display should be turned off after a period of inactivity. Non-volatile storage systems could be unmounted and powered off when the user is not

accessing the data. LED's could blink instead of staying fully on. The processor should be put into lower power states when not in use. Essentially, everything could be turned off and on in order to ensure longer run times.

4.3 USB System

The USB system has been built as defined in the documentation for the ISP1301 USB Transceiver, OMAP 5912, and OMAP 5912 OSK, with a few innovative changes to enhance power and space use, as well as the inclusion of safety voltage cut-off components. The two novel portions of this USB system are the multiplexing of the USB V_{BUS} power line, and the safety voltage cutoff system. Two physical ports were designed in software with one in hardware. Only one was implemented using the already constructed hardware built into the development board.

4.3.1 USB Host and USB On-The-Go

This system was designed to use two ports, a USB host and USB OTG subsystem. The USB host port is built into the OMAP5912OSK, while the USB OTG used one of the internal OMAP5912 USB controllers in conjunction with a separate USB OTG transceiver designed for another board. Due to time constraints, the USB OTG board was never fully constructed. Most of its functionality is demonstrable with the USB host port. Our USB OTG design is available for future development purposes.

As previously mentioned, we used the ISP1301 USB transceiver for the OTG design. USB OTG transceivers are currently uncommon, as On-The-Go is a relatively new specification. Our search for compatible transceivers revealed only two devices. As the ISP1301 provided us with built-in ability to control an external V_{BUS} power source and excellent documentation, our choice was clear.

4.3.2 USB Host

The USB host hardware we used was already implemented on the OMAP5912OSK board. Thus, the only work required to get this port operational was modifications to the Linux kernel.

The hardware USB implementation on the OMAP5912OSK board is very simple. Both USB controller and transceiver are built directly into the OMAP5912 CPU. The built-in controller is flexible: it may be used to implement host- or client-side USB, or the On-The-Go specification. To complete the implementation for the OSK, TI had only to add a few resistors, a power pump, and a transient voltage suppressor. Figure 2 represents this configuration in a schematic diagram.

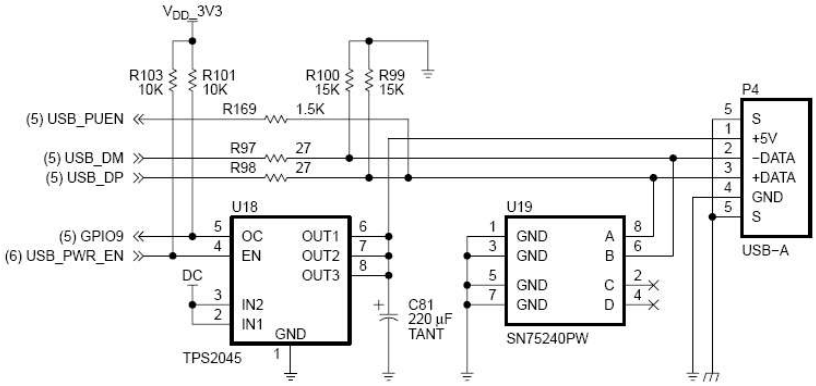


Figure 2: OMAP5912OSK USB Host Schematic

4.3.3 USB V_{BUS} Multiplexing

The USB Specification requires that a USB root power hub have the ability to source either 100 or 500 mA of current at 5 volts. For a hand-held device, this is a significant and challenging requirement. Our device must be able to source power to two such hubs. For future expansion over our current power use, we designed multiplexing of these hubs in order to improve on size and power use. We did not actually implement the design because several related components required more advanced board construction technology than what we had available.

With each power line of the USB interface design a MOSFET is set in place to switch the power back and forth. These MOSFETs are intended to simulate a physical removal of the USB components. Whenever a USB device is not needed, simply adjust the software to expect the device removal (eg. unmount filesystems, bring down network connections), and then turn off the MOSFET eliminating the electrical connection. In addition, the USB OTG Transceiver chip would be disabled when its corresponding lines are disconnected in order to save even more power.

By multiplexing the V_{BUS} line, only 500 mA at 5 volts should are required to run all the USB peripherals. In a wall outlet application this would not affect the system a great deal; however, in a portable application, this improvement over the original figure will reduce the minimum size of the battery by approximately 1.5”x1.5”x.125”, and by 667 mAh. Additionally, only one power source would be required for the USB V_{BUS} lines, thereby saving on parts, cost, and space. From a design perspective this would decrease the weight of the device, decrease the size of the device, decrease the cost of the device, and allow for faster charging.

The preliminary driver code for the VBUS multiplexing line can be seen in Appendix B. This code is untested and as such may not work. Additionally, semaphore locking functions have been commented out for testing purposes.

4.3.4 USB Safety Voltage Cutoffs

Because the USB system has direct access to the battery, we ensured that no part of the system that we created would draw current from the battery beyond its maximum discharge point.

The TPS65010 system power manager provides a 3.3 volt power line to the exterior slots regulated by a LDO (low drop out) voltage regulator. This means that the regulator will work with voltages from the battery very close to 3.3 volts, but not lower. For reference, this is also the highest voltage provided by the TPS65010.

This voltage was designed to power and turn on MOSFETs throughout the USB system in order to manually turn off components when 3.3 volts could no longer be supplied. A large pull-down resistor was added to this design in order to ensure that the power line would in fact drop to ground when disconnected, and in turn turn off the components.

4.4 LCD Panel Design

We needed a graphic LCD display that was cheap, easily readable, and electrically compatible with the OMAP5912 chip. To fit those requirements, we chose to use the Sony ACX705AKM LCD module with the OMAP5912's built-in LCD controller. The reasons for our choice, as described below, are mostly due to the availability of LCD panels.

Graphic LCD modules can be divided into two categories: those with intelligent LCD controllers and those without. The intelligent LCD controllers almost universally accept one address line, one control line, and eight data lines, allowing it to be memory mapped or attached to the GPIO lines of a microprocessor. Through that interface, the intelligent controller accepts commands that operate on its video memory, such as writing pixel data. Then, the controller

refreshes the LCD with the contents of its video memory.

This type of panel appears to be aimed at hobbyist and other low-volume uses because of the ease of integration provided by the intelligent controller's interface. As a consequence, these are also the more expensive modules.

Both active and passive matrix LCD modules that come without an intelligent controller have their own well-standardized interfaces. An active matrix LCD panel typically requires a horizontal sync signal, a vertical sync signal, the pixel data, and a pixel clock signal. The passive matrix version is similar. While the interface itself is simple, it is not easy to use because the panels have no memory and must be fed a constant stream of pixel data at a high clock rate. For this reason, these types of modules are less popular among hobbyists and are sold only in large quantities. They also tend to be cheaper.

The OMAP5912 features a built-in LCD controller that can produce the signal required to drive these standard, “unintelligent” LCD modules. In the interests of keeping costs low and tuning the design for high volume production, our design uses this feature to drive a "dumb" graphic LCD panel. Unfortunately most of these panels are not marketed in single quantities, so our decision to use the Sony ACX705AKM was based almost entirely on availability. Furthermore, its size (71.4mm x 52.0mm) lends itself to an easily readable display, and it uses an LED frontlight instead of an EL backlight. The benefit of the LED frontlight is that unlike an EL backlight, it does not require an inverter to generate a high AC voltage to drive it. The drawback is that the frontlight consists of LEDs in series, which require a high DC voltage to turn on.

To connect the panel electrically, we connect the hsync, vsync and pixel clock signals from the OMAP chip to the LCD panel. The OMAP is driving 16 pixel output lines, however the LCD panel uses nine bit color. Therefore, we connect only nine of the 16 pixel output lines to the LCD panel. The lines are chosen such that they correspond with the most significant bits of the red, green, and blue color channels produced by the OMAP chip. We chose the most significant bits so that the native 16 bit color space would map cleanly to the 9 bit color space, however the choice is arbitrary, because we control the colors generated in software and therefore control all 16 bits. See Appendix C for a full schematic of our circuit to accomplish this.

Two electrical modifications were made to the standard data sheet design we had initially intended. 150Ω resistors were placed on the data lines and a 10Ω resistor was placed in series with the inductor in the boost power supply circuit.

During our testing of the LCD board, we found the output of the OMAP5912 GPIO to be 3.5 VDC when high, which exceeded the maximum operating input voltage for the data lines on our LCD. To address this issue, we placed resistors on all the data lines in series with protective resistors on the LCD itself. The resistors function to limit the current and prevent the collector-gate PN junctions in the LCDs logic gates from forward biasing.

In order to boost the supply voltage to the LCD, we used a National Semiconductor LM2621 DC-DC converter. This boost converter was necessary because the TPS65010 on the OSK board provided +3.0 and +3.3VDC supply lines, but the LCD module also required a +3.8VDC supply line. During testing we found that the output voltage of this converter often reached 6 volts, and occasionally 14 volts. After much testing and analysis, we determined that while the inductor was shorted from +3V to ground, too much current was drawn from the power source. This caused the TPS65010 to enter current limiting mode. Because of this sudden change the inductor and system around it would react in an unexpected manner, briefly raising the output voltage and endangering the LCD's circuitry. In order to avoid this effect, a 10Ω resistor was placed in series with the inductor to limit its maximum current to approximately 330 mA. After performing this modification, the output voltage held steady at a safe operable level, while drawing only a few more milliamps of current. For a full schematic of this module, see Appendix C.

4.4.1 Pushbutton Design

For each button, a GPIO sense line is connected via a pull up resistor to +3V, and by a normally-open button to ground. When the button is pressed, the GPIO line is shorted to ground; when depressed, the GPIO line is pulled high. The only issue we encountered was that our pull up resistors were initially too large and created a voltage divider with the internal resistance and current leakage of the OMAP5912 chip. This kept the voltage on the line from reaching logic high. For our prototype this was fixed by reducing the resistance from 300K to 10K resistors; for future designs we recommend instead using pull down resistors.

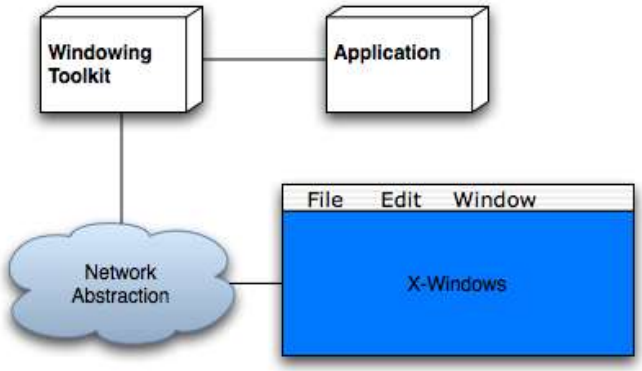
5 Software Design

In the interest of saving money and time, we chose to have our software run under the GNU/Linux operating system. All team members were already familiar with UNIX®-style operating systems, so this left us in relatively familiar territory. We used the GNU toolchain (such as GCC and binutils) to build the embedded software. The two main infrastructure-related options are the support programming languages, which most of the embedded application code will be written in, and the windowing toolkits, which help provide the GUI for the device.

5.1 Overall Software Architecture

Traditionally, GNU/Linux programs are written in C or C++, using either the GTK or QT windowing toolkits interfacing with the X-Window windowing system. This is shown in *Figure 3*. C and C++, discussed further below, are low-level languages which, while powerful, are not generally time efficient languages to express most programs. GTK, QT, and X-Windows are fairly large packages. While these obstacles are normally easily overcome or ignored outright on traditional computer systems, in an embedded environment with extremely limited development time, they become major concerns.

Figure 3: X-Window Based Software Stack



X-Window provides the graphical interface to the user. It communicates with the application software using IP. The application uses a windowing toolkit to abstract away the details of communicating with X-Window.

One popular alternative to the traditional C(++)/window toolkit/X-Window set is the Python language and the wxPython windowing toolkit. The benefit of this setup is that Python is a fairly high level language and wxPython is well-documented and widely used. The downside is that X-Window may still be required and both Python and wxPython are large packages.

The above approaches are technically feasible, but for purposes of minimizing both development time and memory footprint, we chose to take a more streamlined approach. This approach is to use the Lua language and Simple DirectMedia Layer. The Lua language compiler and runtime take no more than 100KB of memory, making it fairly lightweight, while SDL allows direct control of the video output without requiring windowing systems. The downside of this approach is that the features that windowing toolkits provide, such as container-based UI elements and dynamic resizing of those UI elements, required some reimplementation.

5.1.1 Lua

We chose to write the application code in an interpreted language called Lua because of the simplicity of the language, the ability of Lua to support object oriented and aspect oriented program designs, the ease at which Lua can be embedded and linked to C code, and

its extremely small memory requirements. We considered other languages, such as pure C, Java, and Python, but their feature sets and install base were generally inappropriate for our device.

Pure C is often a good choice for embedded systems because it is low level to the point where it can be used to write an operating system. Unfortunately, it is also error-prone, requiring care while implementing even the most basic algorithms. For instance, in the eighth annual International Conference on Functional Programming (ICFP) programming contest, only 5% of the C submissions did not crash during testing, and more interestingly, the ones that crashed had a larger mean time between failures than buggy submissions in other languages [icfpc]. In other words, the nature of the bugs in C code make them more difficult to find because they don't show up early in testing.

Java is a mature language designed for embedded use as part of Sun's "the network is the computer" paradigm, and it is used to build cross platform applications. Ideally we would use Java, but it has two major implementation issues: it is not well supported under Linux, and the standard Java distribution is too large for embedded usage. While Sun has created the Java 2 Mobile Edition, or J2ME, to correct the problems with using the standard edition in embedded devices, we do not have the time to investigate and implement it.

Python is a high level interpreted language which is popular for quickly building applications, however it suffers the same distribution size problem as java. The base language alone is approximately 600k compressed [pythononhandhelds], and it comes packaged with many other libraries. This does not make it ideal for embedding.

Lua is a small interpreted language designed to be embedded into existing C applications. For compatibility with existing C applications, Lua has been written in ANSI C, a universal standard that almost every C compiler supports. Furthermore, Lua is very small. The parser, compiler, and interpreter compiles down to about 100k of object code [luavspython]. This is in contrast to Java which consumes a few megabytes of disk space before adding the SWING gui toolkit[javaembedded]. Using both the C language and an embedded Lua scripting language, we get the ease of programming provided by Lua along with the opportunity to use C as the need arises.

5.1.2 Lua GUI Library

To code our graphical user interface, we developed a Lua GUI toolkit based on a subset of the Java SWING toolkit. Specifically, we adopted the container model and we adapted many of the SWING semantics. This allows anyone who is comfortable with SWING to use the toolkit, and allows us to benefit from the maturity of SWING's design. SWING isn't perfect, so we adopted some ideas from the W3C DOM event model. We also replaced the listener design pattern used by both the W3C DOM event model and the Java SWING toolkit with lambda closures provided by Lua.

The container-based GUI model treats each GUI element, or widget, as a container which can contain other GUI elements. For instance as shown in Figure 4, a window can contain a few button widgets and a scrolling widget. The scrolling widget contains other widgets.

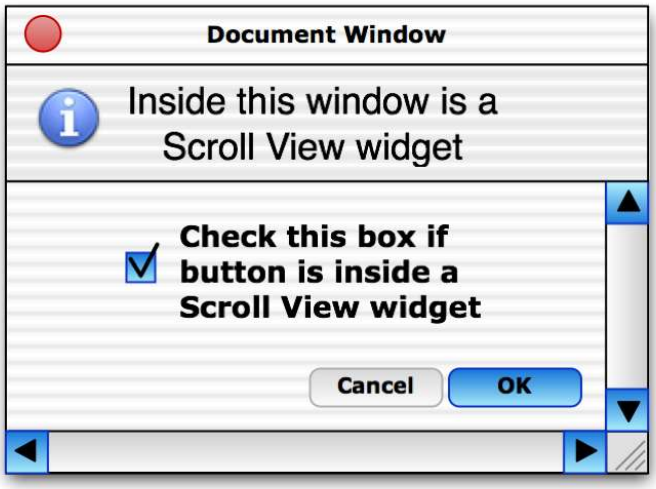


Figure 4: Window containing buttons and scrolling widget which contains more widgets

Layout managers which are attached to each container determines the size and position of each subcontainer. To do this, they query each subcontainer for the minimum size it can occupy and its preferred size. For instance, as shown in Figure 5, the default layout manager, gui:vertical_layout_manager, stacks each sub-container vertically and positions them to be as wide as possible and as tall as preferred.

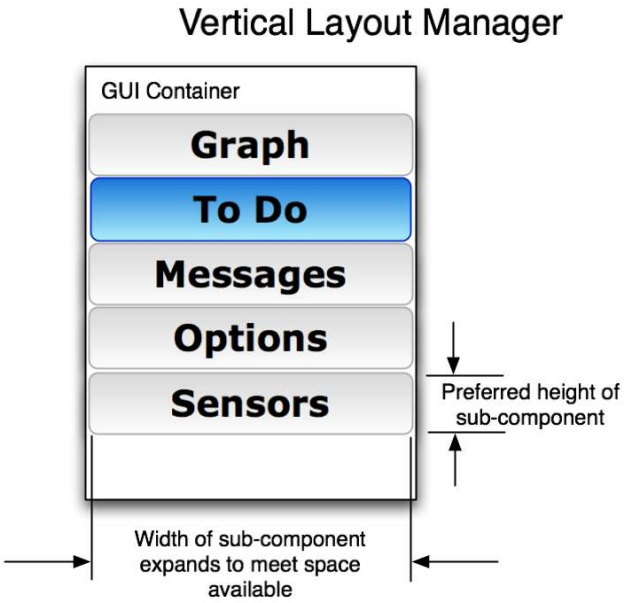


Figure 5: Buttons stacked by gui:vertical_layout_manager

The use of container based layouts ended up being extremely helpful. When we built the LCD display system, we found that the display operates in landscape mode rather than vertically. All we needed to do to adjust was change the screen size in the screen object container and the rest of the GUI resized as shown in Figure 6.

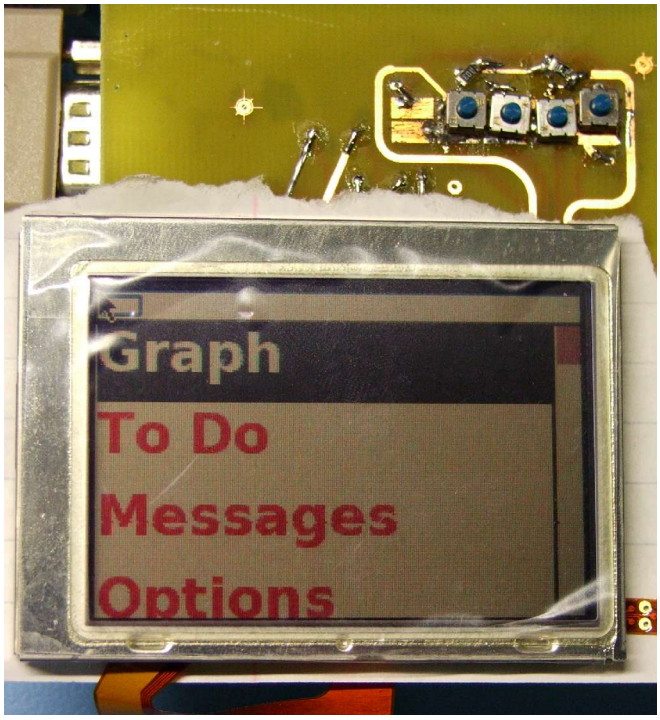


Figure 6: Landscape Oriented GUI

Events, which are notifications of button presses or other actions, are first sent to the container that is said to have the *focus*. As shown in Figure 7, the event is passed as the only parameter to the container's handle_event function. The handle_event function either returns true if it has handled the event or false if it has not. The default behavior of a container is not to handle an event. The event bubbles up to the parent containers until eventually one of them handles the event or the event has bubbled up to the root of the tree.

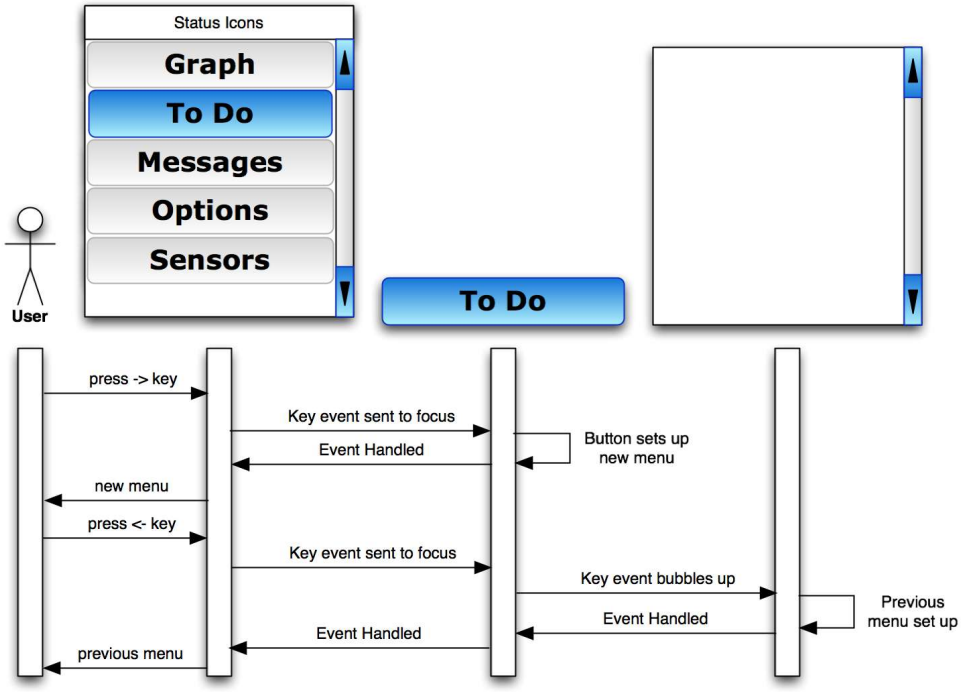


Figure 7: Event sent to focus and then bubbling up to container that handles it

In an early version of the application software shown in Figure 8, pressing the

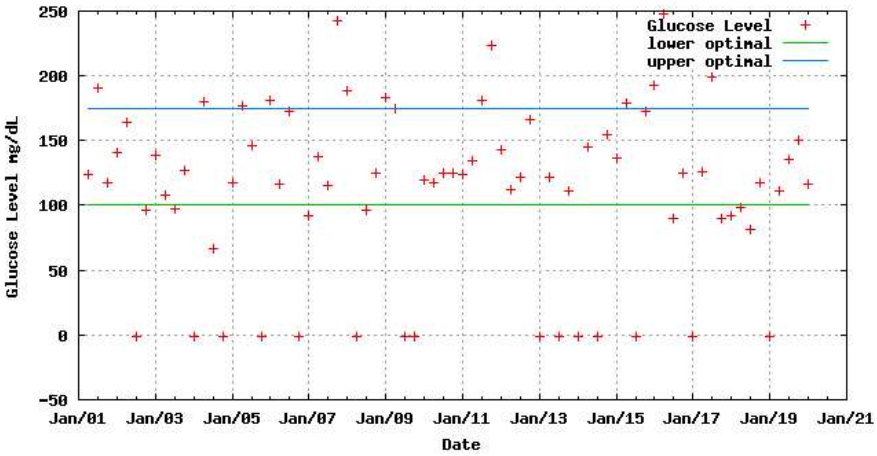
right button would result in an event being sent to the selected on screen button, which has white on black text. The handle_event procedure for the button accepts the event and performs an action. In this case it causes the To Do list to be shown. But if the user pressed the up, down, or left button instead, the on screen button would not handle the event and, instead, it would bubble up to the scroll area that contains the buttons. The scroll area handles the event by either moving the selection up or down, in the case of the up or down buttons, or goes to a previous menu when the left button is pressed.

Figure 8: Screenshot of early version of menu with To Do selected

5.1.3 Software Data Management

The software includes the capability to display stored information in a human readable format. In a future implementation, some data, such as blood glucose data, could to be pre-processed to extract patterns meaningful to the user. For example, the plot of raw blood glucose data in Figure 9 can show when the user has glucose levels outside the optimal range, however, it does not show meaningful long-term trends.

Figure 9: Blood Glucose Levels over Three Weeks During Morning, Afternoon, Evening, and Night

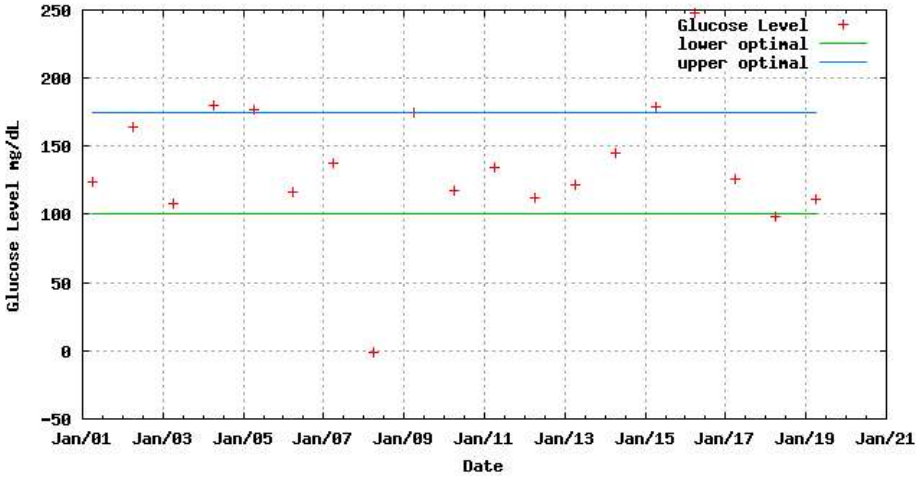


This plot of glucose data taken from a diabetic monitoring company shows glucose levels in the morning, afternoon, evening, and night over the course of three weeks. The lower and upper limits of the optimal range for this patient are shown delimited by two lines. Source: informedcare.com, Example: Glucose Data, http://informedcare.com/glucoseData.html

Part of the reason that long-term trends are not apparent is that glucose levels vary

over the course of a day, and therefore weekly variations are masked by daily variations. After pre-processing the data by removing all but the samples taken in the morning, as shown in Figure 10, we can view the same data and conjuncture that the patient has maintained tight glucose control early in the month and has considerable variation towards the end.

Figure 10: Morning Blood Glucose Levels over Three Weeks



This plot shows only the glucose levels recorded during the morning.

Some blood glucose plots have lines between sample points. Unfortunately, since blood glucose data is only taken three or four times a day, the combined data doesn't have enough resolution to linearly interpolate between samples. One solution, used in the plots above, is to plot only the data points. Another solution we looked into is using insulin-glucose models to provide a best-guess of blood glucose levels between samples.

Additionally, we optimized the displayed information for the small output screen size and limited input capabilities of the device. For instance, computer data plotting applications allow zooming to any arbitrary region of the plot using a mouse, but our device doesn't have a mouse. Using soft-buttons for this purpose is tedious. Therefore, we found alternative methods of setting the zoom level appropriately for the data.

5.1.4 Software Updates

Bridge-IT's software can be upgraded by several means: network push, firmware reload, and emergency firmware recover. The ability to update the software through the network is a consequence of Bridge-IT's network protocol described later which is built around running programs received over the network. The other two need to be specifically implemented, allow the user to update the firmware without a network connection, and are described below.

Handling Firmware Updates and Data Transfer Over USB

One of the stated requirements for Bridge-IT is the ability to update firmware over the USB port. To avoid end-user frustration, it also needs to be fail-safe. The customer should not be able to damage their device unless they intend to make a non-approved modification.

The straightforward approach is to accept a USB bulk transfer and write it directly into the firmware flash, but if there is an error in the transfer, or if the firmware image is corrupted, the device is left in an unusable state. Writing the firmware to RAM first is not workable because we may simply not have enough RAM to fit the firmware. The solution is to write the firmware into the CompactFlash device, check it for integrity, make sure the battery is charged or that the device is on line power, and then write the new firmware into internal flash.

We can make one further optimization to minimize writing code, support USB firmware updates from almost any host PC, and as a bonus, give users access to their data over USB. That optimization is to use the file-backed storage gadget. The file-backed storage gadget is a driver in the Linux kernel which allows us to make any block device appear as a USB mass storage device on the host PC. Modern general-purpose operating systems, including MacOS, Linux, and Windows, treat mass storage devices as removable hard disks. In this way, we can make the CompactFlash device appear as an external hard disk on the host PC.

From there, the process of updating the firmware is as follows:

1) User plugs Bridge-IT into the host PC's USB port.

2) Bridge-IT notices that it is plugged into a USB master, unmounts the CompactFlash device and activates the file-backed storage gadget.

3) The CompactFlash device appears as a removable hard disk on the host PC.

4) The user saves the new firmware into the removable hard disk and then unplugs Bridge-IT from the host PC's USB port.

5) Bridge-IT notices that it is no longer plugged into a USB master, mounts the CompactFlash device, and notices the new firmware image file. The image's integrity is checked and mounted read-only. If the firmware image contains an executable INSTALL program, the program is run; otherwise, the image is unmounted and the install fails.

6) The install program then checks to see if the internal battery is charged, and if so, writes the firmware into the internal flash memory and reboots Bridge-IT into the new firmware.

We use a separate INSTALL program to complete the install process so that any harmful bugs in the process, such as a bug in the flash code, can be fixed by changing the firmware update file. As a bonus, it takes no extra effort on our part.

An example of a simple INSTALL program is shown in Listing 1. It uses built-in GNU commands to perform the reflash operation.

Listing 1: INSTALL example

```
#!/bin/sh

flash_eraseall /dev/mtdblock3

dd if=/mnt/CompactFlash/imagefile of=/dev/mtdblock3
count=1024
```

Failed Update Recovery

A firmware update may fail to leave the device in a usable state, despite the precautions in the update procedure, for any of the following reasons: electrical failure, bad firmware update code, or bad firmware update. If the update fails, Das U-Boot can be used to restore the internal flash memory. While this is not the most user friendly of methods, it could be improved upon to make accessibility easier in future versions, and it is highly unlikely that any user would experience a flash failure.

Das U-Boot is the built-in bootloader that runs when the device is first turned on, and prepares the OMAP chip to run the operating system. If it detects anything over the device's RS232 port before it boots the operating system, it will halt the boot process and enter its interactive mode. From there, a new firmware can be uploaded via Kermit and written to flash.

If the bootloader is overwritten or otherwise inaccessible, the OMAP can be put into full-boot mode. This is done on the OMAP5912 OSK board by setting jumper JP3 to the left (pins 1 and 2). From this mode, Das U-Boot can be restored with a program called omapfl. First, run omapfl as root. If it is not root, it will fail silently.

Listing 2: Das U-Boot Reloaded

```
./omapfl 2nd.bin dasuboot.bin
```

Then, remove the serial cable from the RS232 port on the device and connect the device to the computer. When the device is rebooted into full-boot mode, it will act as a USB client and omapfl will send it a small program called 2nd.bin. The OMAP chip then runs 2nd.bin, which accepts the bootloader program over USB and writes it into the internal flash memory. At this point the Das U-Boot can complete the restoration.

The reason why we didn't modify the omapfl program ourselves to be easily usable by home users is that home users cannot easily put the OMAP CPU into full-boot mode. If a production version of the device offers an easy way to do this, then omapfl can be packaged with the bootloader as a standalone program and linked against the windows libusb dll.

5.1.5 Health Monitor Data Flow

One of the base requirements of this system is to transfer data from the glucometer to the end database storage system. This section outlines that data transmission from the glucometer to the health monitor and in turn through wireless form the health monitor to the

database storage system.

Storage of Event Data on CompactFlash

One of the primary goals of Bridge-IT is to record data from other devices. We call each of these measurements an event. We would ideally like the event data to be stored as in a format that is compact, and easy to read, write, and analyze by not only our application software but also any other software. To meet those goals we chose to use the FAT32 file system with a time-based directory structure.

We chose to use FAT32 because it is one of two filesystems that almost every operating system supports, including MacOS, Linux, and Windows from Win95 up to present day. The other one is FAT16. FAT16 is a popular choice for flash filesystems because it is simpler to implement than FAT32, however FAT32 greatly improves limits on minimum file size, maximum disk size, and maximum number of files per folder. Linux supports both filesystems, and consequentially Bridge-IT will work with CompactFlash cards formatted for FAT16, ext2, jffs2 or any other filesystem that Linux supports. On the other hand, if the CompactFlash card is not already formatted, we will format it FAT32.

One concern when dealing with Flash-based memories is the limited number of erase cycles before flash memory cells stop working. Specialized filesystems such as jffs2 perform wear-leveling, distributing the erases across the entire flash device so that no one memory cell gets repeatedly erased to the point of failure. While FAT32 provides no wear-leveling, it's not a big concern for three reasons: First, Bridge-IT only adds data to the flash device and therefore does not accrue many erases. Second, many flash memory devices have built-in translation layers that perform wear-leveling behind the scenes. Third, modern flash devices can sustain up to ten million erase cycles [wikipedia:1].

The event data itself will be ordered by time. This is because the data is naturally indexed by time (they are events), and because we expect that any analysis of the data will be time-based. If the data needs to be indexed along a different dimension, it can either be re-indexed as necessary, or the Bridge-IT application software can be upgraded to keep such an index in another file.

In order to avoid hitting against the limited number of files that can be in a folder under FAT32, the files are split into separate directories, first by year and then by month. In that scheme, an event occurring on 25 June 2004 would be filed in the June directory of the 2004 directory.

The use of Lua as our application language makes data representation in CompactFlash files extremely straightforward because Lua is designed for the task. Two events may be written to a file as shown in Listing 3.

Listing 3: Measurement Events Expressed in Lua

```
-- Event 1
bi_data.event{
        time = 1127725316    -- time in seconds since UNIX epoch
        (IEEE Std 1003.1-2001)
        glucose_level = 56
}

-- Event 2
bi_data.event{
        time = 1127725432    -- time of 2nd event
        glucose_injection = 120        -- amt of glucose injected
}
```

Not only is this human-readable, it is a valid Lua program. A Lua-based application can read these data files simply by defining a function called "event" – see Listing 4 -- and running the data files as executable code.

Listing 4: Defining function event() in Lua

```
function event(new_event)
        event_table = event_table or {}        -- make sure event_table exists
        event_table.insert(new_event)          -- add event to the list of events
end
```

The actual implementation is slightly more complicated to run under limited memory conditions. Specifically, we use proxy tables for lazy loading and weak-linked tables to allow the garbage collector to purge the data when not referenced from anywhere. Lazy loading means that the data is only loaded into memory when it is needed. Garbage collection is the process whereby, in response to low memory, the Lua interpreter periodically checks what pieces of memory are being actively used, and reclaims any that are not.

Minimizing data size can be done in two ways: by designing the data storage format to minimize the number of bits needed to code the data, and by using compression to decrease information redundancy. For example, compression of glucose data can be done by using a glucose level predictor and only writing the error signal between the prediction and the actual data. The number of bits used by the error signal can then be minimized using Huffman encoding. The better the predictor is, the less error signal we expect, and the less bits will be needed to represent the error signal.

Unfortunately, compression comes at the cost of readability, both for human beings and for software written by human beings. Implementing compression and decompression code would represent a barrier to anyone who wishes to interoperate with our application software. Because CompactFlash memory is extremely cheap and therefore can cheaply store large amounts of information, it makes more sense to use a readable, if larger data format than an unreadable, lean format. Compression may make some sense for opportunistic wireless data transfer because it decreases the amount of time that the user has to be within range of an access point, however existing applications can be layered in to provide such compression, and time constraints on the project implementation push this lower on the priority list.

Network Transport of Data

When Bridge-IT is connected to a wireless network, it tries to push its stored data to a central server. To do this, it makes an HTTPS connection to a URL specified by the vendor of the device. In this example, we will imagine the URL is https://www.bridge-it.com/upload_data.php.

upload_data.php returns a program which Bridge-IT runs through its Lua interpreter. Typically this program tells Bridge-IT to upload data it has older than a certain date. For example:

Listing 5: Autogenerated Lua program to select data to cull or upload

```
-- If the user elects to automatically delete old data, this will do it
bridgeit.data.autodelete{
        cull_before = 1532890348
}
-- upload any new data since 1532890348 seconds after the UNIX epoch
bridgeit.data.upload{
        url = "http://www.bridge-it.com/upload_data.php"
        cull_before = 1532890348
}
```

The upload function is implemented as a version of the same function that saves data onto the CompactFlash, except instead it's sent over the network.

To ensure the patient's medical data does not enter unwanted hands, an additional layer is added to the connection: encryption. We chose to use SSL/TLS encryption, commonly recognized by HTTPS URLs.

5.1.6 Database Design

In order to have our data logger transfer measurement records to a central server, we needed a target database. Lacking precise technical specifications for this database's structure, we created a basic but functional design, and ensured the supporting code base was flexible enough to accept future modifications or total database replacement. In the interest of rapid deployment and compatibility across a wide range of available database servers, we used the open-source MySQL database server.

The database holds static information about a target patient, glucometer, and data logger, and links it to data read from our device. All data read from the glucometers is transferred: namely, the time and date reported for a measurement by the glucometer, blood-glucose level, and the test strip number. The data logger attaches a sequence number to each measurement for tracking purposes; this, too, is transferred. Finally, the date and time of data transfer from glucometer to data logger and from data logger to database is recorded. The recording of multiple timestamps can aid detection of anomalous data, such as the appearance of several glucometer samples timestamped for several months earlier than the transfer dates.

As previously mentioned, in order to simulate a production database, the glucometer data listed above is linked against static patient, glucometer, and data logger information. Patient information includes a patient ID number, along with generic personal information such as first and last names, address, and phone numbers; glucometer and data logger information is limited to serial numbers, model names, and per-model comments to track consistent issues.

The full database structure and example queries are detailed in the dbcreate.sql file attached in Appendix A: Database Design. A visual synopsis of the database, which lists the tables and their fields, is shown in Figure 11.

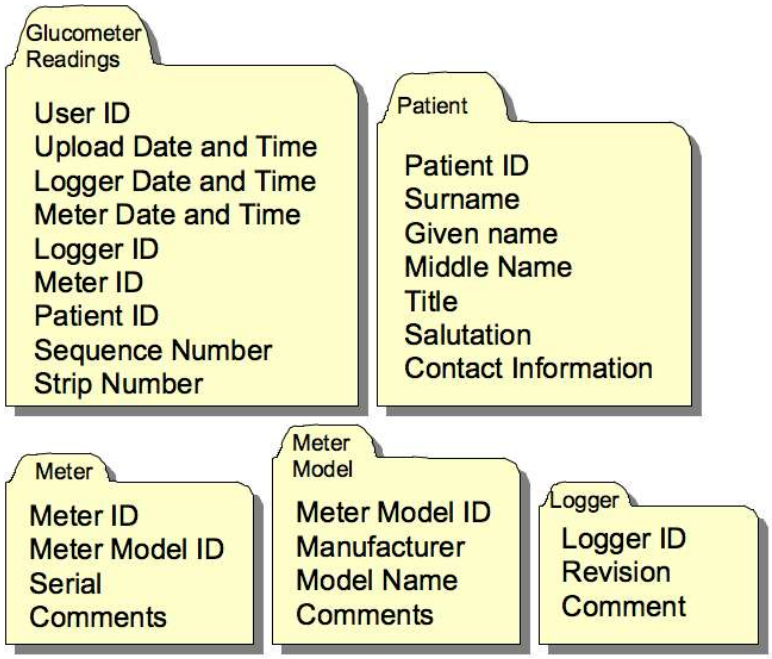


Figure 11: Visual Representation of Database

5.1.7 User Interface Design

Like most user interfaces, the goal of Bridge-IT's UI is to be user friendly, however it is equally important that the user feels in control. This is because the user cannot be compelled to use our device; rather, the user needs to want to use it, and putting the user in a position of power is a means to that end.

Command Oriented Systems

To ensure a user friendly system, we've made the device command oriented, which means we offer the user a list of commands that the user can choose from. In the interests of display readability, we minimize the amount of extraneous graphics and maximize the size of the text on the screen. Therefore, the screen real estate should be

mostly text (the commands) with lines to delineate separate commands as shown in Figure 12 below.

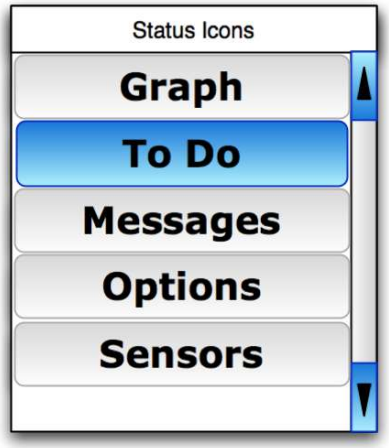


Figure 12: Mock up of GUI

We chose not to use icons because icons need to have labels. Without labels, an icon wouldn't be discoverable. In other words, the user wouldn't know what an icon represents without actually trying it. Anyone who has used Microsoft Word can identify with this issue. With labels, we run into problems with readability of small font sizes. As such, icons don't make sense for this device.

The small size of the screen and readability concerns make it difficult enumerate all possible options on one screen. Scrolling helps, however selecting an option in a list takes linear time so it doesn't scale to a large number of options. By arranging the options in a hierarchy such as in Figure 13, average seek time for an option approaches logarithmic time. In other words, as the cumulative number of items (n) in the menus grows, we can expect the time it takes users to find any particular item to grow proportionally to log(n).

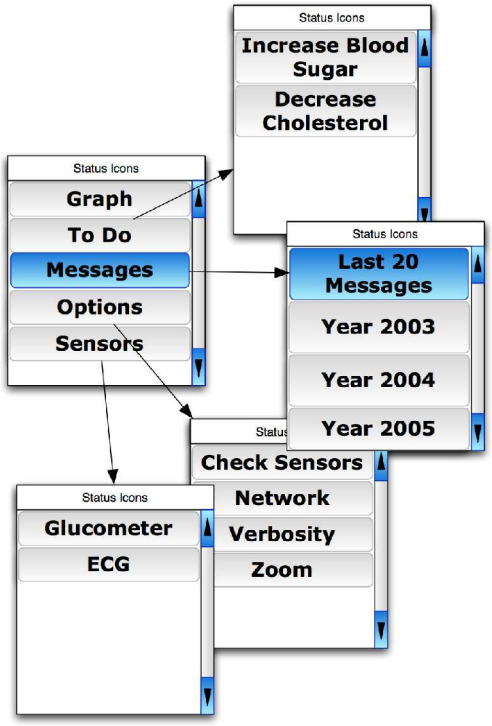


Figure 13: Hierarchy GUI

Note this design resembles that used by the Apple iPod. It's likely that the iPod engineers faced similar UI design requirements.

User Interface Functionality

The main options Bridge-IT offers are to graph the data, read messages, set preferences, and associate with a wireless network. The messages facility allows you to view messages from care providers, who may in turn have access to the data uploaded by the device.

Messages are sorted by date, so to select a message you first choose whether you want one of the last 20 messages or you choose which year to examine. If you choose a year, you are then presented with the months in that year. Upon selecting a month, you are presented with the messages that you received during that month. If at any level in that month/year hierarchy there are 20 messages or less, the list of 20 messages will be displayed instead in chronological order.

The user preferences allow the user to choose various personalization settings including the zoom level of the graphs, the polling interval to check for wireless networks, etc.

The graphing facility is shown in Figure 14. The markings on the horizontal axis are sparse so that the axis is readable on a small display, and on the y axis numbers are replaced by a gradient bar. By eliminating numbers, the y axis is easier to read, and furthermore it is an innovative solution to an issue we noticed in early software simulations: changes in the y axis are disorienting. With the gradient bar, perspective is easier to maintain.

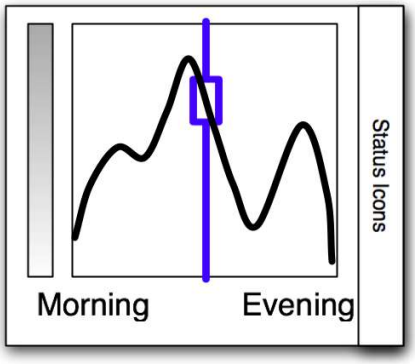


Figure 14: Graph

The line going down the middle of the Figure 14 shows the "current position" cursor. The big number on the top-left of the graph shows the glucose level at the current position, and any zooming operations occur around the current position. By scrolling, the user can pan the current position across the graph, and by clicking the user obtains a menu. The menu allows zooming, skipping to points of interest, or quitting the graph utility.

Audible User Interface

We did not implement the Audible User Interface because the Linux OMAP sound driver does not work correctly. We did not have the time to debug it. That said, we did load a speech synthesizer onto the OMAP5912OSK board and generated speech sound files to demonstrate the capability. Attempting to play generated speech on the OSK hardware resulted in the first few phonemes played repeatedly in a loop which is likely a driver problem. What follows is our unimplemented design of how to make

Bridge-IT audibly accessible.

Audible accessibility is provided by using the flite speech synthesis system to perform text to speech and by use of audio cues for user actions. The flite speech synthesizer by default uses diphone synthesis to generate speech, and can be extended to perform unit selection synthesis or domain-specific synthesis. Although speech generated with diphone synthesis sounds robotic, it has two properties that natural sounding speech does not have: it is unambiguous and it can be sped up to 150 words per minute or more with little loss of clarity. For these reason, speech synthesizers using this speech synthesis method are popular among visually disabled people. Using flite, Bridge-IT can speak what the user has selected, and can also read messages.

Audible cues offer the second component to the audible user interface. While scrolling menus, Bridge-IT produces two tones: a tone that represents the current menu and a tone that represents the current selection. As the user scrolls up or down, the menu tone remains constant while the selection tone changes pitch higher or lower.

Audible cues work slightly differently on the graph. While the user is scrolling the graph, an aural representation of the graph is synthesized and played. There are two aural rendering methods: first, a tone can be amplitude modulated by the glucose level at the current position on the graph. When the user scrolls forward and the glucose dips, the amplitude goes down. Likewise, when the glucose level rises, the amplitude does as well. This could also be implemented with pitch instead of amplitude.

The second, more interesting aural rendering method is to treat the y axis of the graph as the frequency and the x axis as the amplitude. To generate the sound, we perform an inverse Fourier transform of the graph and play that. While this method requires more sophisticated hearing, it is already used in existing technology to give blind people sight through the use of digital sound synthesis and a CCD camera [seeingwithsound].

5.2 Building and Installing a Linux Kernel

We built our Linux Kernel using the ELDK toolchain provided at http://www.denx.de/twiki/bin/view/DULG/ELDK using the kernel source downloaded from http://source.mvista.com/. Using the ELDK toolchain is not recommended; instead use the OpenEmbedded toolchain described later in the paper. We first attempted several times to use git, Linus Torvald's collaborative version system, to download the source via http. Each attempt took a few days and failed. The alternative method, rsync, is deprecated and in theory should not be used. In reality we used rsync without git and quickly obtained a usable Linux kernel source tree. To date, using git to obtain the source will still fail, and the maintainer of the source recommends using rsync over http.

Once we obtained the source code, we created the .config file shown in Appendix D and we used "make uImage" to build the kernel. This creates an image file called / arch/arm/boot/uImage. We set up the tftpd daemon, a small program that implements the trivial file transfer protocol, to serve a folder called "reflash" and we copied the image into the folder.

To flash the kernel image onto our device, we interrupt Das U-boot and issue the command

```
tftpboot 0x10000000 uImage
```

This tells U-boot to download the image and copy it into address 0x10000000h which, according to Figure 15, corresponds to the beginning of RAM memory. Next, we need to erase the flash area allocated for the kernel, so we issue the command

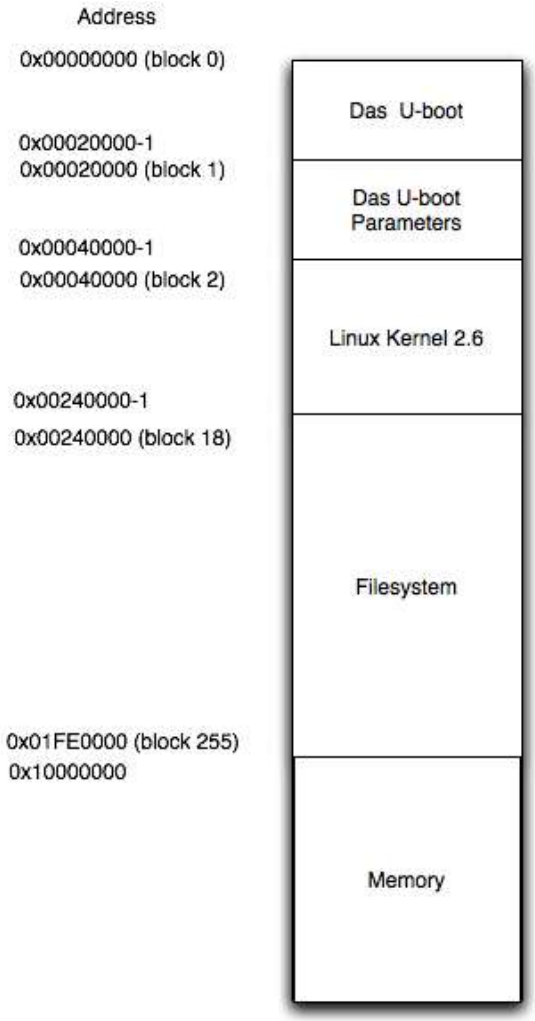


Figure 15: Memory Map (Mapped Devices not shown)

```
erase 1:2-17
```

This erases flash sectors 2 through 17. Each flash sector is 128kb, so this corresponds to address 0x40000h (128kb * 2) through address 0x220000h (128kb * 17). This contradicts the literature that comes with the OSK board. That is because the 2.6 line of kernels map the flash memory more efficiently than the older 2.4 kernel that ships with the OSK. After erasing, the new kernel can be copied into flash memory.

```
cp.b 0x10000000 0x40000 $(filesize)
```

This copies the data from RAM into the flash area allocated for the kernel. We can then boot the kernel

```
bootm 0x40000
```

By setting boot parameters, we can change the behavior of Bridge-IT's startup. For development, we want the kernel to use files on our development machine. To do this, we

exported a directory containing the root filesystem, or the files the device needs to start up, using the Network FileSystem, or NFS. Then, we set the bootargs parameter to give the device an IP address on startup and the location of the root filesystem on the network:

```
setenv bootargs console=ttyS0,115200n8 rw
nfsroot=10.50.68.77:/opt/armroot noinitrd mem=30M
ip=10.50.68.200:10.50.68.1:10.50.68.1:255.255.240.0:debian::on
```

When the bootloader starts the kernel, it passes the bootargs parameter to the kernel. These bootargs tell the kernel that its console is on ttyS0 (the RS232 port) using 115200 baud. It's root filesystem is on the network at 10.50.68.77 in directory /opt/armroot. Furthermore, the IP of the device on boot is 10.50.68.200, the gateway is 10.50.68.1, and the subnet mask is 255.255.240.0. Since we needed to assign the device an unused IP, we used arping to check IPs until we found one that was free. To save the bootargs parameter so that it persists on restart:

```
saveenv
```

When we need Bridge-IT to boot off of it's internal flash memory instead of a filesystem on the network, we set bootargs:

```
setenv bootargs console=ttyS0,115200n8 rw rootfstype=jffs2
root=/dev/mtdblock3 noinitrd mem=30M
ip=10.50.68.200:10.50.68.1:10.50.68.1:255.255.240.0:debian::on
```

This tells the kernel to boot off of the third flash device, which is mapped to memory location 0x240000 also known as flash sector 18, and that it should expect a jffs2 filesystem.

To automatically boot into linux, we can set the bootcmd parameter:

```
setenv bootcmd bootm 0x40000
```

After the root file system is set up as explained in the next section, kernel modules, or small pieces of the kernel that can be dynamically loaded or unloaded, can be built and installed into the root filesystem

```
make modules
export INSTALL_MOD_PATH=/opt/armroot/
make modules_install
```

5.3 Building the ZyDAS ZD1211 Linux Wireless Driver

ZyDAS open sourced the driver for their wireless 802.11bg chipsets however it doesn't

build cleanly against the OMAP 2.6.14 kernel. This is because it was written to support kernel 2.6.4, and since then some of the internal kernel APIs have changed. To fix this, we changed a function call from verify_area to access_ok in zd1205.c and we commented out all usages of URB_ASYNC_UNLINK in zd1211.c.

To cross compile the module, we changed the CC, CPP and MODPATH directives in the makefile to

```
CC=arm-linux-gcc
CPP=arm-linux-g++
MODPATH := /opt/armroot/lib/modules/$
(KERNRELEASE)
```

and we set the KSRC environment variable

```
export KSRC=/opt/linux-omap-2.6.git/
```

At this point we can run "make" to build the module

5.4 Adding Kernel Support for Hardware Buttons

We noticed an existing kernel driver that handles buttons in /arch/arm/plat-omap/gpio-switch.c. It accepts a list of switches from the bootloader and generate hotplug events when the switches are triggered. It also shows the state of the switches via the sysfs filesystem in /sys.

We avoided the additional complexity of a modified version of Das U-boot by modifying the gpio-switch code to load a hard-coded list of switches. Furthermore, we added code to gpio-switch.c to register the switches as input devices and inject input events when the switches are triggered. This allowed us to assign keys to each switch. When a switch triggers, the corresponding key pressed event is sent through the kernel. Our four current switches are mapped to the left, up, and down arrow keys as well as the return key.

5.5 Adding Kernel Support for Custom LCD Panel

The Linux OMAP LCD driver was set up to drive the Mistral LCD board, a $799 expansion board [minstrel]. We adjusted the timing parameters and LCD controller registers to handle our cheaper $40 prototype LCD expansion card. This was done by adjusting the screen size, hsync, and vsync timing parameters in drivers/video/omap/lcd_osk.c:

```
static struct lcdc_video_mode mode240x320 = {
        /*.x_res = 160,
        .y_res = 240,*/
        .x_res = 240,
        .y_res = 160,
        /*.pixel_clock = 12500,*/
        .pixel_clock = 6000,  // this is ignored
        .bpp = 16,      // ignored
        .hsw = 9,       // horizontal sync width
        .hfp = 16,              // horizontal front porch (delay before horizontal sync)
        .hbp = 16-9,    // horizontal back portch (delay after start of hsync)
        .vsw = 2,       // vertical sync width -- min is 2, but be safe
        .vfp = 20,      // vertical front porch
        .vbp = 20-2,    // vertical back porch
        .pcd = 12,      // pixel clock divider
};
```

Listing 6: Modification to lcd_osk.c

We also changed drivers/video/omap/omap_lcdc.c to force the pixel clock divider to be 50 and to invert the hsync and vsync signals.

```
printk("%% pcd %d\n", l & 0x000000FF);
l = (l & 0xFFFFFF00) + 50;  // used to be 30
printk("%% pcd %d\n", l & 0x000000FF);
printk("%% hsync %d vsync %d\n", (l>>21)&1,
(l>>20)&1);
l |= 1 << 21;    // invert hsync
l |= 1 << 20;    // invert vsync
printk("%% hsync %d vsync %d\n", (l>>21)&1,
(l>>20)&1);
```

Listing 7: Modification to omap_lcdc.c

With these changes, the LCD panel worked correctly.

5.6 Building and Installing a Root Filesystem

The Linux Kernel requires a root filesystem which contains, among other things, the init program that the kernel runs after it has loaded. Unlike the kernel, the root filesystem is neither

simple to build nor trivial to obtain for three reasons. First, a fully functional system requires many programs from a variety of different sources. Second, the developers of these programs may or may not have enabled their program to be cross-compileable, and the software will need to be cross compiled to the ARM architecture. Third, the limited flash memory in the device cannot support the size of a binary based distribution such as Debian.

Initially, we obtained a pre-built root filesystem from http://www.uclibc.org/ which is built with their small C library. We removed the /dev/random device node which did not work and replaced it with a symlink to /dev/urandom. Furthmore, we compiled and installed the udev package on it. Unfortunately, the limited programs on the pre-built filesystem as well as the difficulty of cross compiling new software for it meant that we needed a better approach.

OpenEmbedded provides a set of programs that cross compile and build root filesystems from the original source code. This was our solution. Unfortunately it took us 1.5 months to get it up and running. The first obstacle to using OpenEmbedded is its high resource requirements. Our development machine was a 400MHz computer with 64 megabytes of RAM and an 8 GB hard disk before installing the OS. Bitbake, OpenEmbedded's build tool, needs at least 4 GB to build a basic rootfs and requires as much memory as it can get. More taxing, however, is Monotone, the versioning system needed to download OpenEmbedded's data files. Monotone is CPU bound and can take days to download and verify the data files on a 400MHz machine.

The default version of Monotone in Debian's repositories is too old so we obtained a more recent version from an OpenEmbedded developer. We downloaded a snapshot of the OpenEmbedded database and attempted to use Monotone to update it to the latest version. Unfortunately Monotone reports errors when performing the update. We were told the errors were not harmful. While this is technically true, eventually we realized that it is harmful in the sense that Monotone halts the update after raising the error. To force the update to finish, we used a shell script to continuously run Monotone overnight.

```
for i in `seq 1 20`
do
        monotone --db=/opt/oe/oe.db pull monotone.vanille.de
"org.openembedded.{dev,dreambox}"
done
```

Listing 8: Shell script to repeatedly pull from monotone repository

Then, to get the OpenEmbedded data files out of the monotone database, we performed a checkout:

```
monotone --db=/opt/oe/oe.db checkout --branch=org.openembedded.dev
```

The second obstacle we encountered was incomplete documentation on setting up bitbake's local.conf file. After a long period of trial, error, and asking for help, we determined that the following settings are necessary beyond what is described in the getting started guide:

```
CVSDATE = 10052005        # set this to your own date
preferred-gcc-version = 3.4.4
preferred-gcc-cross-version = 3.4.4
preferred-gcc-native-version = 3.4.4
```

These settings prevent OpenEmbedded from continuously downloading and building newer versions of the software and prevents OpenEmbedded from using GCC 4.0 which doesn't work.

Once local.conf is set up, the initial OpenEmbedded image can be built

```
bitbake bootstrap-image
```

and a script that sets up a cross compiling environment can be built

```
bitbake devshell
```

For development, we then mounted the bootstrap-image onto a folder and exported the folder using NFS. We also exported the deploy (/opt/oe/build/tmp/deploy/ipks/) directory where OpenEmbedded stores everything that it builds. This allowed us to build a program on our development machine and install it directly on Bridge-IT.

To burn the root filesystem onto the internal flash memory, we first generated a jffs2 image file:

```
mkfs.jffs2       -l --eraseblock 0x20000 -r /opt/armroot -o paul_date.jffs2
```

Then, using U-boot, we transferred the image into flash memory location 0x240000 (flash block 18):

```
tftpboot 0x10000000 paul_date.jffs2
erase 1:18-255
cp.b 0x10000000 0x240000 $(filesize)
```

5.7 Building the GUI

To build the GUI we needed a Lua interpreter as well as the SDL, POSIX, and cairo modules. We also need the SDL libraries. OpenEmbedded builds Lua for us for free:

```
bitbake Lua
```

It also builds cairo and pango, the graphic and text layout engine respectively, for free.

Unfortunately OpenEmbedded's SDL libraries target an X Windows back end which we have been trying to avoid. Therefore we copied and made our own version of the SDL build scripts that targets the framebuffer device. We changed the EXTRA_OECONF directive in the build script by setting --disable-video-x11 and --enable-video-fbcon. Lastly we changed the name of the build script to libsdl-fbcon_1.2.7.bb to reflect the use of the framebuffer instead of X11. It built beautifully.

```
bitbake -b libsdl/libsdl-fbcon_1.2.7.bb
```

It did not run beautifully. The programmers of the framebuffer output drivers for SDL hard coded a list of valid display sizes (modes) and so SDL refuses to use a display whose size is not on the list. While this may be necessary to drive a CRT display, it fails to handle the case of a custom LCD panel. Therefore we added our LCD's size to the list, added bogus timing information, and recompiled.

The SDL module for Lua requires and comes as part of LuaCheia, an expanded Lua distribution. LuaCheia was originally important because Lua itself did not allow loading of external shared libraries. Lua 5.0, which we are using, has the capability to load shared libraries and consequentially LuaCheia is not necessary. Unsurprisingly, OpenEmbedded does not build LuaCheia. Furthermore, attempting to cross compile LuaCheia fails and the complexity of the build scripts make it difficult to fix.

Instead, we extracted those parts of LuaCheia that are necessary to build the SDL module, namely gluahost and SDL. Then, we built our own makefiles (build scripts) which cross compile and link the modules correctly. We similarly created our own makefiles for the POSIX module and our own cairo module.

The actual Lua code that implements the GUI did not need to be modified, save for changing the dimensions of the screen, and surprisingly worked on first try.

5.8 Reverse Engineering the FreeStyle Mini Protocol

There is no existing open source capable of communicating with the FreeStyle glucometer, so we performed black box testing. We used HHD Serial Monitor to capture the RS232 communication link between the FreeStyle and the communication software. From those captures as well as experimental observations, we developed the following protocol specification:

```
RS232 Port Setup:      19200 baud, 8 data bits, 1 stop bit, no flow control

ASCII Protocol (BNF):
        command := 'mem' | <cr>

<cr> command:
        Device response: none

        Side effects: device buffer is cleared

'mem' command:
        Device response := <serial number><cr><date time><log><cr>'end'
        serial number := [A-Za-z0-9\-]
        date time := [put stuff here]
        log := 'LOG EMPTY' | <log entries>
        log entries := <log entry><cr><log entries> | <log entry>
        log entry := <date time>

        Side effects: none
```

Listing 9: BNF specification for FreeStyle wire protocol

based on this specification, we produced a C program called testrs232 which dumps data from any connected FreeStyle device. The Lua application code then performs the parsing.

6 Recommendations

In addition to the specific, per-module recommendations we included in the documentation above, we have general recommendations future product development. These recommendations range from a change to a microprocessor, miniaturization and optimization of circuit design, and implementation of features that were not considered for the proof of concept design.

Our choice of microprocessor was based not only on capabilities, but also on availability. Texas Instruments' OMAP line includes a variety of compatible processors, of which many in the Wireless PDA Solutions category appear to be more suitable for our application. However, TI states "OMAP processors for wireless handsets & PDAs are intended for high-volume wireless OEMs and ODMs and are not available through distributors," and as such that CPU family was unsuitable for our initial proof-of-concept device. The OMAP1710 in particular has all the functionality of the OMAP5912 and more, requires less board space, is software-compatible, and uses about half the power. It also includes a built-in WLAN controller, allowing significant component count reduction. Use of this processor could reduce power and board space requirements, component counts, and circuit complexity.

In our prototype, we used a starter kit from Texas Instruments with a significant amount of unnecessary functionality. In general, use of a starter kit as part of a market-oriented product is restricted to the initial design stages, such as our own. By designing a fully custom circuit, board space, component count, power use, and cost per unit could be reduced.

7 Conclusions

In regards to our original product requirements, we progressed as following:

- **Portable**

In terms of form factor, our product is not yet conveniently portable; we chose to focus more on functionality than miniaturization. Our prototype has a fully operational wireless system. We feel that this was one of the most important requirements to satisfy for portability.

- **Built-in Programmable ID for user identification**

This is easily implemented in software

- **Potential for Broad Sensor Compatibility**

Our device can communicate over RS-232 and USB interfaces. This covers all common glucometer interfaces, satisfying this requirement.

- **Built-in Clock**

Our selected development board has a built-in real-time clock. Synchronizing this against an NTP network timeserver would be a simple matter of software installation. Synchronizing against the central server is already supported as a side effect of the network protocol.

- **Large Data Buffer (At least 1 megabyte)**

Our selected development board has 32MB of built-in flash, and supports expansion through CompactFlash cards.

- **Secure Data Sharing**

The network protocol fully supports secure, encrypted communication.

- **Secure Off-site Storage**

We implemented a database-backed off site storage system for demonstration purposes however we expect that a production system will integrate with a preexisting database. For that reason, we kept the server-side code small, simple and generic enough to be adapted for any use.

- **Well-designed User Interface**

We have designed and prototyped a command-oriented user interface that is easily readable and employs conventions that are already in widespread use. The implementation itself is incomplete in that scroll bars currently do not function and the available widgets are limited.

- **Firmware Upgradeable**

  We created all code necessary for simple firmware upgrades, but did not completely implement client-side USB. Completing this product requirement is a simple hardware addition, but this in turn would require more advanced construction techniques than those to which we had access. Additionally it was discovered that drivers for the particular system we had plans on implementing were not as complete as we had thought through research.

- **Feedback Capability**

  Feedback is functional and can be demonstrated and messages can be displayed however the current software does not display stored messages.

- **Flexible End-user Setup**

  The prototype does not currently implement user-specific features. To demonstrate the concept, a placeholder options menu has been created to demonstrate flexible setup options.

Despite significant and frustrating difficulties involved in piecing together the many components involved in our design, we successfully completed a functioning, usable proof-of-concept device, and were pleased with the result.

Bibliography

University of Maryland Medical Center. *Endocrinology Health Guide - Type 1 Diabetes* <http://www.umm.edu/endocrin/diabmel.htm>

U.S. National Library of Medicine, National Institutes of Health. *MedlinePlus Medical Encyclopedia: Type 2 diabetes* 2005 <http://www.nlm.nih.gov/medlineplus/ency/article/000313.htm>

American Diabetes Association. *What is Pre-Diabetes?* <http://diabetes.org/pre-diabetes.jsp>

American Diabetes Association. *Gestational Diabetes Resource Guide* <http://diabetes.org/gestational-diabetes.jsp>

. *All About Diabetes* <http://diabetes.org/about-diabetes.jsp>

. *Healthy People 2010* <http://www.healthypeople.gov/document/html/volume1/05diabetes.htm>

"What To Do: I Gave the Wrong Dose of Insulin." 1999 <http://www2.texaschildrenshospital.org/internetarticles/uploadedfiles/125.pdf>

Diabetes Federation of Ireland. *Diabetes Care: Securing The Future...more* <http://www.diabetesireland.ie/view.asp?ID=937>

Diabetes Federation of Ireland. *Medication and Devices* <http://www.diabetesireland.ie/view.asp?ID=335>

American Diabetes Association. *Insulin Pumps* <http://diabetes.org/type-1-diabetes/insulin-pumps.jsp>

Diabetes Federation of Ireland. *Medication & Devices* <http://www.diabetesireland.ie/view.asp?ID=740>

American Diabetes Association. *Other Diabetes Medications* <http://diabetes.org/type-2-diabetes/oral-medications.jsp>

American Diabetes Association. *Diabetes Learning Center for the Recently Diagnosed* <http://diabetes.org/all-about-diabetes/diabetes-learning-center.jsp>

BenQ Mobile GmbH & Co.. *The MDKeeper: A "Mini-hospital" for seniors and people requiring medical monitoring worn on the wrist* 2005 <http://communications.siemens.com/cds/frontdoor/0,2241,hq_en_0_110776_rArNrNrNrN,00.html>

Health Hero Network. *Helath Buddy Appliance* 2005
<http://www.healthhero.com/products_services/health_buddy2.html>

Abbott Diabetes Care. *FreeStyle CoPilot*
<http://www.abbottdiabetescare.com/news/20050831_deviceremoval.aspx>

Andersson, Niklas. "Prototype for Transmission ofGlucometer Data by WirelessTechnology." 2003
<http://www.csd.uu.se/courses/course-material/xjobb/docs-reports/Niklas_Andersson-2003.pdf>

"NASA develops "Black Box" to monitor vital signs." 2004 <http://www.news-medical.net/?id=483>

. *CybernetMedical.com* 2005 <http://www.cybernetmedical.com/>

OMAP5912OSK Target Module Hardware Reference Guide
<http://focus.ti.com/lit/ug/spru715/spru715.pdf>

. *ICFPC 2005 Talk* 2005 <http://icfpc.plt-scheme.org/>

. *Python on Handhelds* 2005
<http://www.handhelds.org/moin/moin.cgi/PythonOnHandhelds>

. *Lua vs Python* 2005 <http://Lua-users.org/wiki/LuaVersusPython>

. *Java and Embedded Linux Team Up* 2005
<http://www.linuxdevices.com/articles/AT7873839273.html>

Wikipedia. *Flash Memory* <http://en.wikipedia.org/wiki/Flash_memory>

. *Seeing with Sound* 2005 <http://www.seeingwithsound.com/>

. *Minstrel Software Web Store* 2005
<http://www.mistralsoftware.com/html/services/webstore.php>

8 Appendix A: Database Design

```
# MySQL-style DB create script
create database bridge_it;
use bridge_it;

create table patient (
  patient_id char(255) primary key,
  surname char(255),
  given_name char(255),
  middle_name char(255),
  title char(15),  -- title, ie. Jr., III., etc.
  salutation char(15), -- salutation, ie. Mr., Dr., etc.
  addr_line1 char(255),
  addr_line2 char(255),
  addr_line3 char(255),
  city char(255),
  county char(255),
  country char(255),
  pri_phone char(25),
  sec_phone char(25)
);

create table logger (
  logger_id char(255) primary key,  -- enter unit serial #
  revision char(50),                -- unit revision
  comment text                      -- comments for issue tracking
);

create table meter_model (
  meter_model_id bigint unsigned auto_increment primary key,
  manufacturer char(255),
  modelname char(255),
  comments text              -- comments for issue tracking
);

create table meter (
  meter_id bigint unsigned auto_increment primary key,
  serial char(255),          -- unit serial number
  comments text,              -- comments for issue tracking
  meter_model_id bigint unsigned,
  FOREIGN KEY (meter_model_id) REFERENCES meter_model (meter_model_id)
);

create table gluc_readings (
  UID bigint unsigned auto_increment primary key,
  upload_datetime datetime,  -- timestamp for upload to db
  logger_datetime datetime,  -- timestamp for sensor->logger xfer
  sensor_datetime datetime,  -- timestamp for sensor at time of reading
  sequence_no int,           -- entry sequence number
  strip_no int,              -- test strip number (from sensor)
  logger_id char(255),
  meter_id bigint unsigned,
  patient_id char(255),
  FOREIGN KEY (logger_id) REFERENCES logger (logger_id),
  FOREIGN KEY (meter_id) REFERENCES meter (meter_id),
  FOREIGN KEY (patient_id) REFERENCES patient (patient_id)
);
```

```
# example SQL queries for database:

-- add glucometer data:
insert into gluc_readings
  [(col1_name, col2_name, ..., coln_name)]
  VALUES (col1_data1, col2_data1, ..., coln_data1),
    (col1_data2, col2_data2, ..., coln_data2),
    ...
  ;

-- or more to the point:
INSERT INTO gluc_readings
  (upload_datetime, logger_datetime, meter_datetime,
    logger_id, meter_id, patient_id, sequence_no, strip_no)
VALUES (uploadDateTime, loggerDateTime, meterDateTime, loggerID,
    meterID, patientID, sequenceNo, stripNo);

--(and we'll leave logger_id and patient_id static; device_id we can link
-- against multiple glucometers if you'd like, or leave that static as well)

-- extract all data from most recent sequence number given logger_id:
SELECT * FROM gluc_readings WHERE logger_id='loggerID' ORDER BY sequence_no DESC
LIMIT 1;

-- extract all data given a patient ID
SELECT * FROM gluc_readings WHERE patient_id='patientID' ORDER BY (pick from
anything) (DESC or ASC) (LIMIT if you prefer);
```

9 Appendix B: Preliminary Untested USB V_{BUS} Switching Driver

Makefile:

```
GCC = arm-linux-gcc
OPTS = -Wall -c -D__KERNEL__ -DMODULE -I/opt/linux-omap-2.6.git/include

usbswitch.o:      usbswitch.c
      $(GCC) $(OPTS) usbswitch.c
```

usbswitch.c:

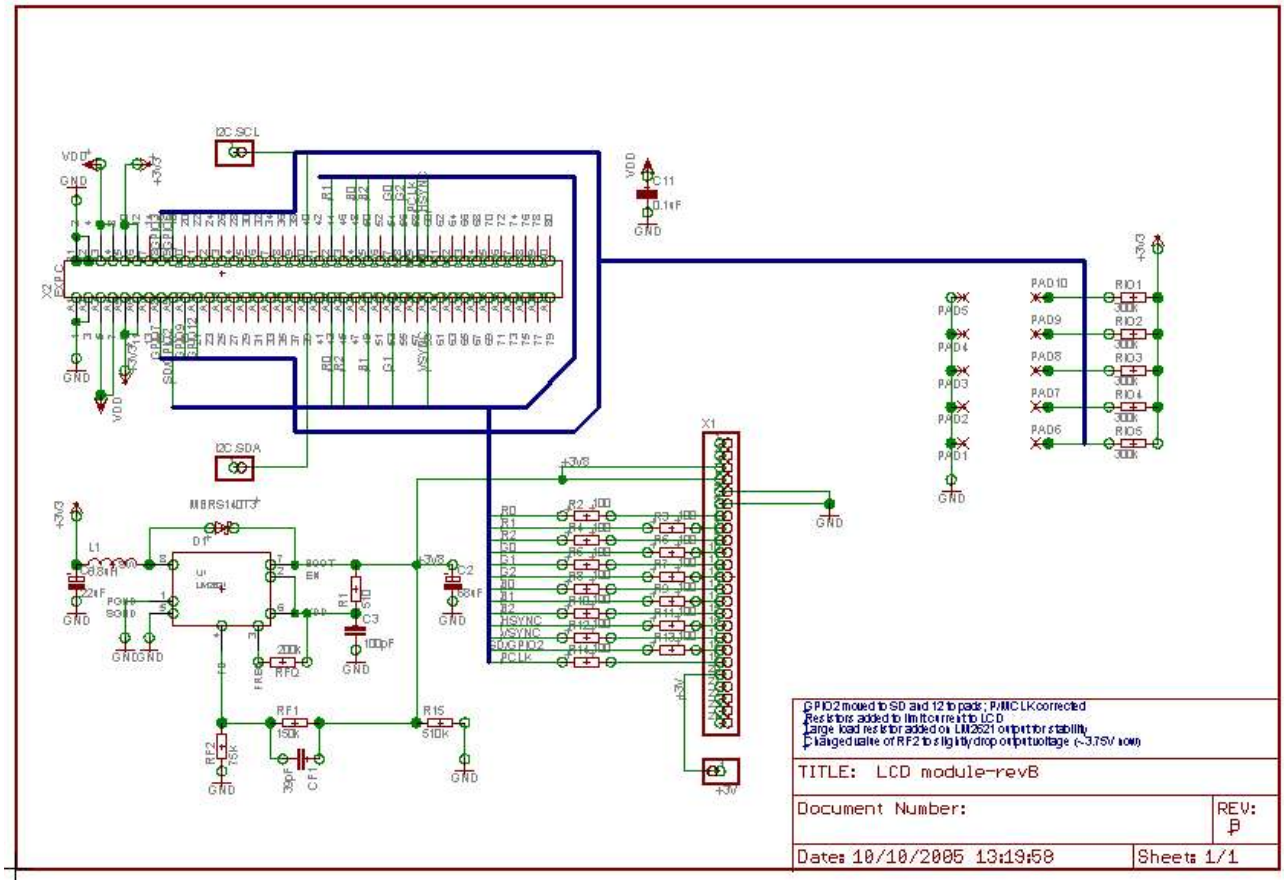
```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/wrapper.h>
#include <linux/types.h>
#include <asm/arch/gpio.h>
#include <linux/spinlock.h>

#define MAJOR_NUMBER 60
#define NAME "usbswitch"
#define PIN_NUMBER 8
#define INPUT 1
#define OUTPUT 0
#define USB_OTG 1
#define USB_OTG_CHAR '1'
#define USB_HOST 0
#define USB_HOST_CHAR '0'

int init_module();

void cleanup_module();

ssize_t usbswitch_read(struct file *filestruct, char *buffer, size_t size,
loff_t *position_ignore);

ssize_t usbswitch_write(struct file *filestruct, const char *buffer, size_t
size, loff_t *position_ignore);

int usbswitch_open(struct inode *inode, struct file *filp);

int usbswitch_close(struct inode *inode, struct file *filp);
/*
static struct file_operations usbswitch_fops = {
  THIS_MODULE,
  NULL,
  usbswitch_read,
  usbswitch_write,
  NULL,
  NULL,
  NULL,
  NULL,
  usbswitch_open,
  NULL,
  usbswitch_close,
  NULL,
  NULL,
  NULL,
```

```
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL
};
*/

static struct file_operations usbswitch_fops = {
  owner: THIS_MODULE,
  read: usbswitch_read,
  write: usbswitch_write,
  open: usbswitch_open,
  release: usbswitch_close
};

static rwlock_t usb_lock;

int init_module(){
  int ret = 0;
  usb_lock = RW_LOCK_UNLOCKED;
  ret = register_chrdev(MAJOR_NUMBER, NAME, &usbswitch_fops);
  if(ret < 0){
    printk("KERN_WARNING: Could not assign major number %d to usbswitch
driver\n", MAJOR_NUMBER);
    return ret;
  }
  ret = omap_request_gpio(PIN_NUMBER);
  if(ret != 0){
    printk("KERN_WARNING: Unable to aquire USB GPIO Switch Pin\n");
    return ret;
  }
  omap_set_gpio_direction(PIN_NUMBER, OUTPUT);
  return 0;
}

void cleanup_module(){
  int ret = 0;
  omap_free_gpio(PIN_NUMBER);
  ret = unregister_chrdev(MAJOR_NUMBER, NAME);
  if(ret == -EINVAL){
    printk("KERN_WARNING: Could not unregister USB switch driver\n");
    return;
  }
  return;
}

int usbswitch_open(struct inode *inode, struct file *filp){
  //Do nothing, we don't need to worry about this.
  return 0;
}

int usbswitch_close(struct inode *inode, struct file *filp){
  //Do nothing, we don't need to worry about this.
  return 0;
```

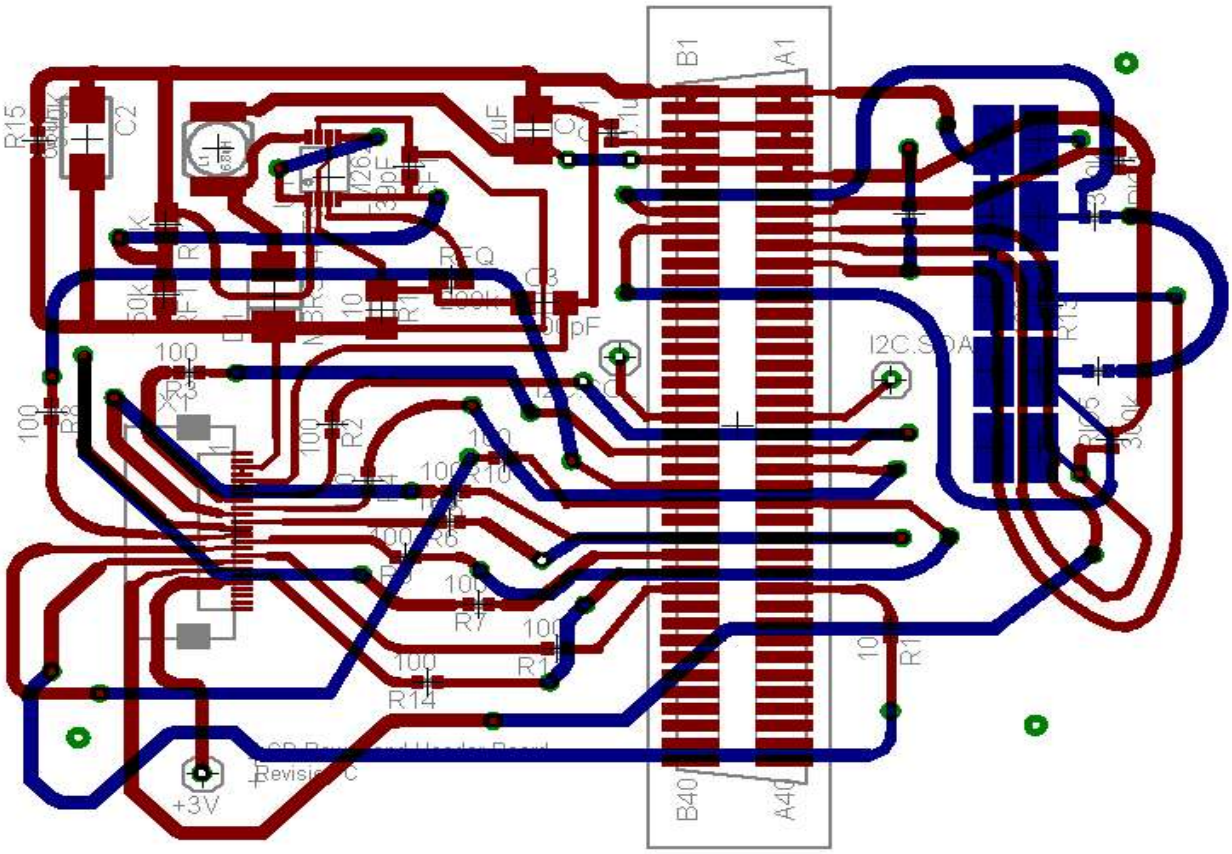
```
}

static int position;

ssize_t usbswitch_read(struct file *filestruct, char *buffer, size_t size,
loff_t *position_ignore){
  //unsigned long flags;

  if(size <= 0)
    return 0;

  //read_lock_irqsave(&usb_lock, flags);

  if(position == USB_OTG)
    buffer[0] = USB_OTG_CHAR;
  else if(position == USB_HOST)
    buffer[0] = USB_HOST_CHAR;
  else{
    //read_unlock_irqsave(&usb_lock, flags);
    return 0;
  }
  //read_unlock_irqsave(&usb_lock, flags);
  return 1;
}

ssize_t usbswitch_write(struct file *filestruct, const char *buffer, size_t
size, loff_t *position_ignore){
  //unsigned long flags;
  int check = size;

  //write_lock_irqsave(&usb_lock, flags);

  while(check > -1){
    if(buffer[check] == USB_HOST_CHAR){
      omap_set_gpio_dataout(PIN_NUMBER, USB_HOST);
      position = USB_HOST;
      //write_unlock_irqsave(&usb_lock, flags);
      return size;
    }
    if(buffer[check] == USB_OTG_CHAR){
      omap_set_gpio_dataout(PIN_NUMBER, USB_OTG);
      position = USB_OTG;
      //write_unlock_irqsave(&usb_lock, flags);
      return size;
    }
    check--;
  }
  //write_unlock_irqsave(&usb_lock, flags);
  return 0;
}
```

10 Appendix C: LCD and USB Module Schematics, PCB Layouts, and Part Lists

LCD Module Schematic:



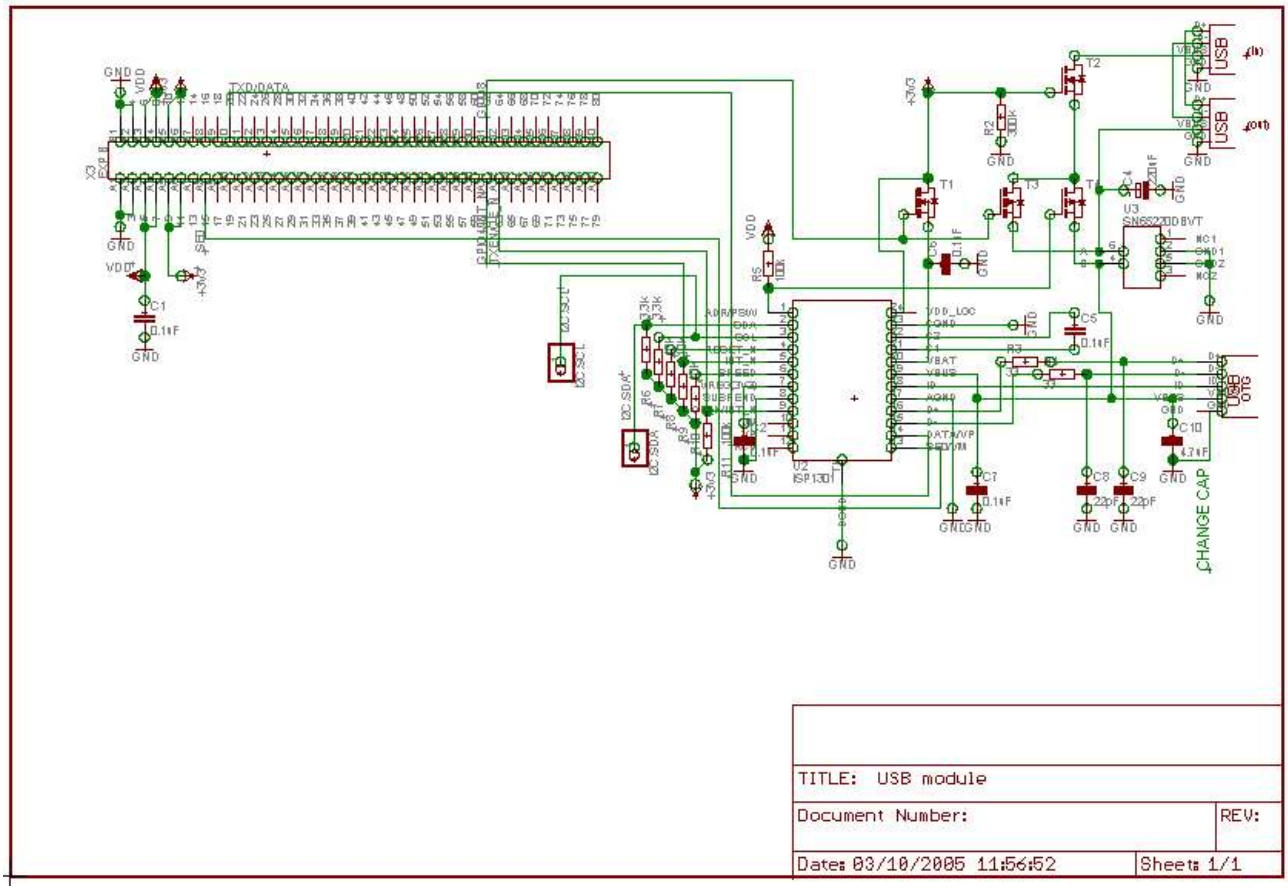
LCD Module PCB:



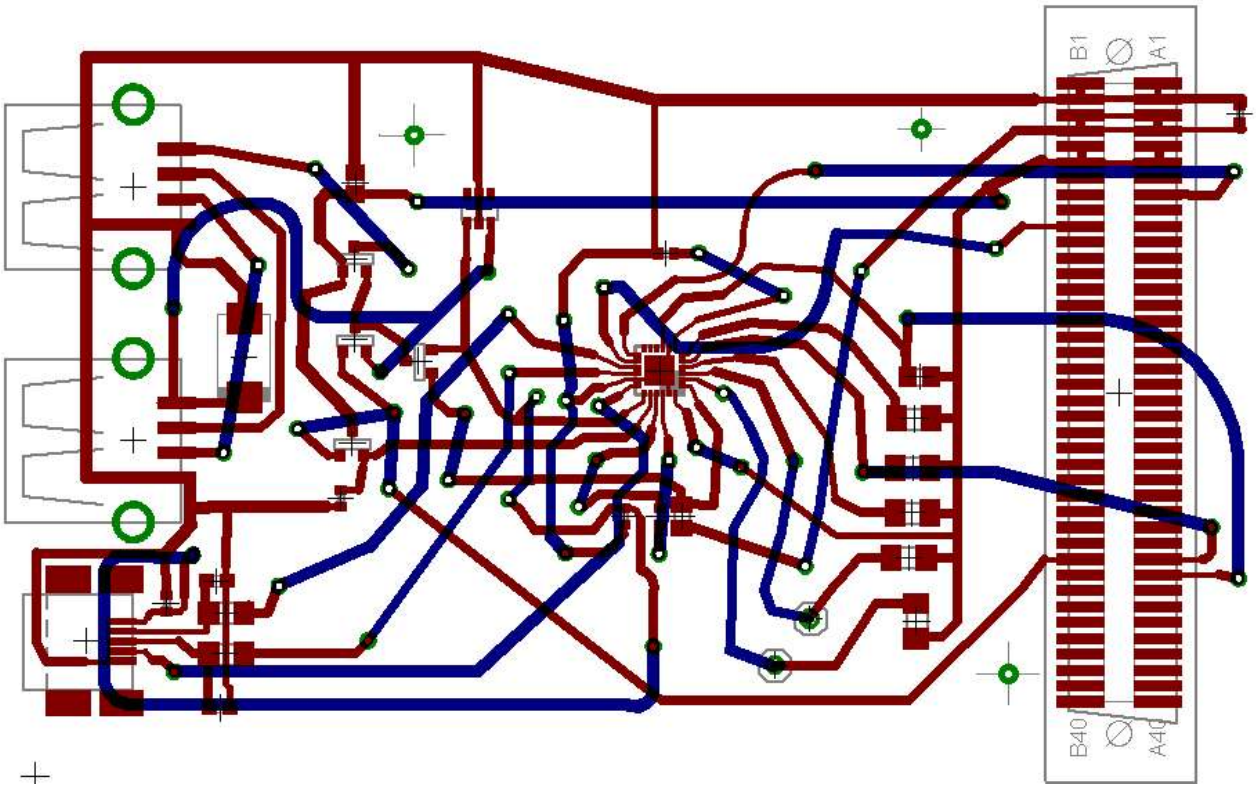
LCD Modules Part List (Exported from Eagle)

Part	Value	Package	Library	Position (mil)	Orientation
+3V		1X01	pinhead	(687 245)	R180
C1	22uF	B/3528-21R	rcl	(1552 2122)	R270
C2	68uF	CT6032	rcl	(386 2096)	R270
C3	100pF	C1206	rcl	(1580 1616)	R0
C11	0.1uF	C0402	rcl	(1754 2112)	R90
CF1	39pF	C0603	rcl	(1228 2016)	R270
D1	MBRS140T3	403A-03	diode	(874 1634)	R90
I2C.SCL		1X01	pinhead	(1776 1459)	R180
I2C.SDA		1X01	pinhead	(2485 1393)	R0
L1	6.8uH	DS1608	inductor-coilcraft	(729 2065)	R90
PAD1		SMD5	wirepad	(2753 1869)	MR180
PAD2		SMD5	wirepad	(2753 1638)	MR180
PAD3		SMD5	wirepad	(2753 1416)	MR180
PAD4		SMD5	wirepad	(2753 1195)	MR180
PAD5		SMD5	wirepad	(2754 2093)	MR180
PAD6		SMD5	wirepad	(2871 1194)	MR0
PAD7		SMD5	wirepad	(2871 1416)	MR0
PAD8		SMD5	wirepad	(2871 1638)	MR0
PAD9		SMD5	wirepad	(2871 1868)	MR0
PAD10		SMD5	wirepad	(2871 2093)	MR0
R1	510	M1206	rcl	(1156 1598)	R270
RF1	150k	R0805	rcl	(588 1640)	R270
RF2	73.2k	R0805	rcl	(593 1846)	R270
RFQ	200k	R0805	rcl	(1338 1686)	R0
RIO1	300k	R0402	rcl	(3081 2032)	R270
RIO2	300k	R0402	rcl	(3019 1868)	MR180
RIO3	300k	R0402	rcl	(3081 1664)	R90
RIO4	300k	R0402	rcl	(3027 1420)	MR180
RIO5	300k	R0402	rcl	(3061 1216)	R90
U1	LM2621	MINI-SO	smd-special	(1016 1984)	R90
X1		XF2H-2415-1	con-omron	(772 952)	R270
X2	EXP C	FX2-80S-1.275V	con-hirose	(2084 1258)	R90

USB Module Schematic:



USB Module PCB:



USB Modules Part List (Exported from Eagle):

Part	Value	Package	Library	Position (mil)	Orientation
C1	0.1uF	C0402	rcl	(3800 2090)	R90
C2	0.1uF	C0402	rcl	(1990 1650)	R180
C4	220uF	SMC_D	rcl	(660 1320)	R90
C5	0.1uF	C0402	rcl	(1970 820)	R90
C6	0.1uF	C0402	rcl	(962.697 876.098)	R270
C7	0.1uF	C0402	rcl	(1860 820)	R270
C8	22pF	C0603	rcl	(572.697 616.098)	R0
C9	22pF	C0603	rcl	(582.697 216.098)	R0
C10	4.7uF	C0402	rcl	(412.697 546.098)	R90
I2C.SCL		1X01	pinhead	(2444 497.803)	R180
I2C.SDA		1X01	pinhead	(2332.504 350.701)	R180
R2	300K	R0805	rcl	(1010 1870)	R270
R3	33	R1206	rcl	(602.697 386.098)	R180
R4	33	R1206	rcl	(602.697 516.098)	R180
R5	100k	M0805	rcl	(2040 820)	R270
R6	3.3K	M1206	rcl	(2780 490)	R90
R7	3.3K	M1206	rcl	(2756 690)	R180
R8	10K	M1206	rcl	(2767 831)	R180
R9	10K	M1206	rcl	(2768 974)	R180
R10	10K	M1206	rcl	(2772 1128)	R180
R11	100k	M0805	rcl	(2790 1260)	R180
T1		SOT-23	zetex	(1000.402 1050.201)	R180
T2		SOT-23	zetex	(1008 1631.602)	R180
T3		SOT-23	zetex	(1008.898 1377.165)	R180
T4		SOT-23	zetex	(1210 1310)	R270
U2	ISP1301	HVQFN24	micro-philips	(1970 1280)	R180
U3	SN65220DBVT	SOT23-6	texas	(1400 1790)	R180
X3	EXP B	FX2-80S-1.275V	con-hirose	(3420 1210)	R90
X4	USB-OTG-S	USB-MAB-S	con-usb	(162.697 426.098)	R0
XUSBIN	USB-A-S	USB-A-S	con-usb	(310 1860)	R0
XUSBOUT	USB-A-S	USB-A-S	con-usb	(310 1060)	R0

11 Appendix D: Bridge-IT Kernel .config

```
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.14-rc1-omap1
# Wed Oct 12 17:12:35 2005
#
CONFIG_ARM=y
CONFIG_MMU=y
CONFIG_UID16=y
CONFIG_RWSEM_GENERIC_SPINLOCK=y
CONFIG_GENERIC_CALIBRATE_DELAY=y

#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
CONFIG_CLEAN_COMPILE=y
CONFIG_BROKEN_ON_SMP=y
CONFIG_INIT_ENV_ARG_LIMIT=32

#
# General setup
#
CONFIG_LOCALVERSION=""
CONFIG_LOCALVERSION_AUTO=y
CONFIG_SWAP=y
CONFIG_SYSVIPC=y
# CONFIG_POSIX_MQUEUE is not set
# CONFIG_BSD_PROCESS_ACCT is not set
CONFIG_SYSCTL=y
# CONFIG_AUDIT is not set
CONFIG_HOTPLUG=y
CONFIG_KOBJECT_UEVENT=y
# CONFIG_IKCONFIG is not set
CONFIG_INITRAMFS_SOURCE=""
# CONFIG_EMBEDDED is not set
CONFIG_KALLSYMS=y
# CONFIG_KALLSYMS_EXTRA_PASS is not set
CONFIG_PRINTK=y
CONFIG_BUG=y
CONFIG_BASE_FULL=y
CONFIG_FUTEX=y
CONFIG_EPOLL=y
CONFIG_CC_OPTIMIZE_FOR_SIZE=y
CONFIG_SHMEM=y
CONFIG_CC_ALIGN_FUNCTIONS=0
CONFIG_CC_ALIGN_LABELS=0
CONFIG_CC_ALIGN_LOOPS=0
CONFIG_CC_ALIGN_JUMPS=0
# CONFIG_TINY_SHMEM is not set
CONFIG_BASE_SMALL=0

#
# Loadable module support
#
CONFIG_MODULES=y
CONFIG_MODULE_UNLOAD=y
# CONFIG_MODULE_FORCE_UNLOAD is not set
CONFIG_OBSOLETE_MODPARM=y
# CONFIG_MODVERSIONS is not set
```

```
# CONFIG_MODULE_SRCVERSION_ALL is not set
CONFIG_KMOD=y

#
# System Type
#
# CONFIG_ARCH_CLPS7500 is not set
# CONFIG_ARCH_CLPS711X is not set
# CONFIG_ARCH_CO285 is not set
# CONFIG_ARCH_EBSA110 is not set
# CONFIG_ARCH_CAMELOT is not set
# CONFIG_ARCH_FOOTBRIDGE is not set
# CONFIG_ARCH_INTEGRATOR is not set
# CONFIG_ARCH_IOP3XX is not set
# CONFIG_ARCH_IXP4XX is not set
# CONFIG_ARCH_IXP2000 is not set
# CONFIG_ARCH_L7200 is not set
# CONFIG_ARCH_PXA is not set
# CONFIG_ARCH_RPC is not set
# CONFIG_ARCH_SA1100 is not set
# CONFIG_ARCH_S3C2410 is not set
# CONFIG_ARCH_SHARK is not set
# CONFIG_ARCH_LH7A40X is not set
CONFIG_ARCH_OMAP=y
# CONFIG_ARCH_VERSATILE is not set
# CONFIG_ARCH_IMX is not set
# CONFIG_ARCH_H720X is not set
# CONFIG_ARCH_AAEC2000 is not set

#
# TI OMAP Implementations
#
CONFIG_ARCH_OMAP_OTG=y
CONFIG_ARCH_OMAP1=y
# CONFIG_ARCH_OMAP2 is not set

#
# OMAP Feature Selections
#
# CONFIG_OMAP_RESET_CLOCKS is not set
CONFIG_OMAP_BOOT_TAG=y
# CONFIG_OMAP_BOOT_REASON is not set
CONFIG_OMAP_GPIO_SWITCH=y
CONFIG_OMAP_MUX=y
# CONFIG_OMAP_MUX_DEBUG is not set
CONFIG_OMAP_MUX_WARNINGS=y
# CONFIG_OMAP_MPU_TIMER is not set
CONFIG_OMAP_32K_TIMER=y
CONFIG_OMAP_32K_TIMER_HZ=128
# CONFIG_OMAP_DM_TIMER is not set
CONFIG_OMAP_LL_DEBUG_UART1=y
# CONFIG_OMAP_LL_DEBUG_UART2 is not set
# CONFIG_OMAP_LL_DEBUG_UART3 is not set
CONFIG_OMAP_SERIAL_WAKE=y

#
# OMAP Core Type
#
# CONFIG_ARCH_OMAP730 is not set
# CONFIG_ARCH_OMAP15XX is not set
CONFIG_ARCH_OMAP16XX=y

#
```

```
# OMAP Board Type
#
# CONFIG_MACH_OMAP_INNOVATOR is not set
# CONFIG_MACH_OMAP_H2 is not set
# CONFIG_MACH_OMAP_H3 is not set
CONFIG_MACH_OMAP_OSK=y
# CONFIG_OMAP_OSK_MISTRAL is not set
# CONFIG_MACH_OMAP_GENERIC is not set

#
# OMAP CPU Speed
#
# CONFIG_OMAP_CLOCKS_SET_BY_BOOTLOADER is not set
# CONFIG_OMAP_ARM_216MHZ is not set
CONFIG_OMAP_ARM_192MHZ=y
# CONFIG_OMAP_ARM_168MHZ is not set
# CONFIG_OMAP_ARM_120MHZ is not set
# CONFIG_OMAP_ARM_60MHZ is not set
# CONFIG_OMAP_ARM_30MHZ is not set
# CONFIG_OMAP_DSP is not set

#
# Processor Type
#
CONFIG_CPU_32=y
CONFIG_CPU_ARM926T=y
CONFIG_CPU_32v5=y
CONFIG_CPU_ABRT_EV5TJ=y
CONFIG_CPU_CACHE_VIVT=y
CONFIG_CPU_COPY_V4WB=y
CONFIG_CPU_TLB_V4WBI=y

#
# Processor Features
#
# CONFIG_ARM_THUMB is not set
# CONFIG_CPU_ICACHE_DISABLE is not set
# CONFIG_CPU_DCACHE_DISABLE is not set
# CONFIG_CPU_DCACHE_WRITETHROUGH is not set
# CONFIG_CPU_CACHE_ROUND_ROBIN is not set

#
# Bus support
#
CONFIG_ISA_DMA_API=y

#
# PCCARD (PCMCIA/CardBus) support
#
CONFIG_PCCARD=y
# CONFIG_PCMCIA_DEBUG is not set
CONFIG_PCMCIA=y
CONFIG_PCMCIA_LOAD_CIS=y
CONFIG_PCMCIA_IOCTL=y

#
# PC-card bridges
#
# CONFIG_TCIC is not set
CONFIG_OMAP_CF=y

#
# Kernel Features
```

```
#
# CONFIG_PREEMPT is not set
CONFIG_NO_IDLE_HZ=y
# CONFIG_ARCH_DISCONTIGMEM_ENABLE is not set
CONFIG_SELECT_MEMORY_MODEL=y
CONFIG_FLATMEM_MANUAL=y
# CONFIG_DISCONTIGMEM_MANUAL is not set
# CONFIG_SPARSEMEM_MANUAL is not set
CONFIG_FLATMEM=y
CONFIG_FLAT_NODE_MEM_MAP=y
# CONFIG_SPARSEMEM_STATIC is not set
# CONFIG_LEDS is not set
CONFIG_ALIGNMENT_TRAP=y

#
# Boot options
#
CONFIG_ZBOOT_ROM_TEXT=0x0
CONFIG_ZBOOT_ROM_BSS=0x0
CONFIG_CMDLINE="mem=32M console=ttyS0,115200 initrd=0x10400000,8M root=/dev/ram0
rw"
# CONFIG_XIP_KERNEL is not set

#
# CPU Frequency scaling
#
# CONFIG_CPU_FREQ is not set

#
# Floating point emulation
#

#
# At least one emulation must be selected
#
CONFIG_FPE_NWFPE=y
# CONFIG_FPE_NWFPE_XP is not set
# CONFIG_FPE_FASTFPE is not set
# CONFIG_VFP is not set

#
# Userspace binary formats
#
CONFIG_BINFMT_ELF=y
# CONFIG_BINFMT_AOUT is not set
# CONFIG_BINFMT_MISC is not set
# CONFIG_ARTHUR is not set

#
# Power management options
#
CONFIG_PM=y
# CONFIG_APM is not set

#
# Networking
#
CONFIG_NET=y

#
# Networking options
#
CONFIG_PACKET=y
```

```
# CONFIG_PACKET_MMAP is not set
CONFIG_UNIX=y
# CONFIG_NET_KEY is not set
CONFIG_INET=y
CONFIG_IP_MULTICAST=y
# CONFIG_IP_ADVANCED_ROUTER is not set
CONFIG_IP_FIB_HASH=y
CONFIG_IP_PNP=y
CONFIG_IP_PNP_DHCP=y
CONFIG_IP_PNP_BOOTP=y
# CONFIG_IP_PNP_RARP is not set
# CONFIG_NET_IPIP is not set
# CONFIG_NET_IPGRE is not set
# CONFIG_IP_MROUTE is not set
# CONFIG_ARPD is not set
# CONFIG_SYN_COOKIES is not set
# CONFIG_INET_AH is not set
# CONFIG_INET_ESP is not set
# CONFIG_INET_IPCOMP is not set
# CONFIG_INET_TUNNEL is not set
CONFIG_INET_DIAG=y
CONFIG_INET_TCP_DIAG=y
# CONFIG_TCP_CONG_ADVANCED is not set
CONFIG_TCP_CONG_BIC=y
# CONFIG_IPV6 is not set
# CONFIG_NETFILTER is not set

#
# DCCP Configuration (EXPERIMENTAL)
#
# CONFIG_IP_DCCP is not set

#
# SCTP Configuration (EXPERIMENTAL)
#
# CONFIG_IP_SCTP is not set
# CONFIG_ATM is not set
# CONFIG_BRIDGE is not set
# CONFIG_VLAN_8021Q is not set
# CONFIG_DECNET is not set
# CONFIG_LLC2 is not set
# CONFIG_IPX is not set
# CONFIG_ATALK is not set
# CONFIG_X25 is not set
# CONFIG_LAPB is not set
# CONFIG_NET_DIVERT is not set
# CONFIG_ECONET is not set
# CONFIG_WAN_ROUTER is not set
# CONFIG_NET_SCHED is not set
# CONFIG_NET_CLS_ROUTE is not set

#
# Network testing
#
# CONFIG_NET_PKTGEN is not set
# CONFIG_NETFILTER_NETLINK is not set
# CONFIG_HAMRADIO is not set
# CONFIG_IRDA is not set
CONFIG_BT=y
# CONFIG_BT_L2CAP is not set
# CONFIG_BT_SCO is not set

#
```

```
# Bluetooth device drivers
#
# CONFIG_BT_HCIUSB is not set
# CONFIG_BT_HCIUART is not set
# CONFIG_BT_HCIBCM203X is not set
# CONFIG_BT_HCIBPA10X is not set
# CONFIG_BT_HCIBFUSB is not set
# CONFIG_BT_HCIDTL1 is not set
# CONFIG_BT_HCIBT3C is not set
# CONFIG_BT_HCIBLUECARD is not set
# CONFIG_BT_HCIBTUART is not set
# CONFIG_BT_HCIVHCI is not set
CONFIG_IEEE80211=y
# CONFIG_IEEE80211_DEBUG is not set
CONFIG_IEEE80211_CRYPT_WEP=y
CONFIG_IEEE80211_CRYPT_CCMP=y
CONFIG_IEEE80211_CRYPT_TKIP=y

#
# Device Drivers
#

#
# Generic Driver Options
#
CONFIG_STANDALONE=y
CONFIG_PREVENT_FIRMWARE_BUILD=y
CONFIG_FW_LOADER=y

#
# Memory Technology Devices (MTD)
#
CONFIG_MTD=y
# CONFIG_MTD_DEBUG is not set
# CONFIG_MTD_CONCAT is not set
CONFIG_MTD_PARTITIONS=y
# CONFIG_MTD_REDBOOT_PARTS is not set
CONFIG_MTD_CMDLINE_PARTS=y
# CONFIG_MTD_AFS_PARTS is not set

#
# User Modules And Translation Layers
#
CONFIG_MTD_CHAR=y
CONFIG_MTD_BLOCK=y
# CONFIG_FTL is not set
# CONFIG_NFTL is not set
# CONFIG_INFTL is not set

#
# RAM/ROM/Flash chip drivers
#
CONFIG_MTD_CFI=y
# CONFIG_MTD_JEDECPROBE is not set
CONFIG_MTD_GEN_PROBE=y
# CONFIG_MTD_CFI_ADV_OPTIONS is not set
CONFIG_MTD_MAP_BANK_WIDTH_1=y
CONFIG_MTD_MAP_BANK_WIDTH_2=y
CONFIG_MTD_MAP_BANK_WIDTH_4=y
# CONFIG_MTD_MAP_BANK_WIDTH_8 is not set
# CONFIG_MTD_MAP_BANK_WIDTH_16 is not set
# CONFIG_MTD_MAP_BANK_WIDTH_32 is not set
CONFIG_MTD_CFI_I1=y
```

```
CONFIG_MTD_CFI_I2=y
# CONFIG_MTD_CFI_I4 is not set
# CONFIG_MTD_CFI_I8 is not set
CONFIG_MTD_CFI_INTELEXT=y
# CONFIG_MTD_CFI_AMDSTD is not set
# CONFIG_MTD_CFI_STAA is not set
CONFIG_MTD_CFI_UTIL=y
# CONFIG_MTD_RAM is not set
# CONFIG_MTD_ROM is not set
# CONFIG_MTD_ABSENT is not set
# CONFIG_MTD_XIP is not set

#
# Mapping drivers for chip access
#
# CONFIG_MTD_COMPLEX_MAPPINGS is not set
# CONFIG_MTD_PHYSMAP is not set
# CONFIG_MTD_ARM_INTEGRATOR is not set
# CONFIG_MTD_EDB7312 is not set
CONFIG_MTD_OMAP_NOR=y
# CONFIG_MTD_PLATRAM is not set

#
# Self-contained MTD device drivers
#
# CONFIG_MTD_SLRAM is not set
# CONFIG_MTD_PHRAM is not set
# CONFIG_MTD_MTDRAM is not set
CONFIG_MTD_BLKMTD=y
# CONFIG_MTD_BLOCK2MTD is not set

#
# Disk-On-Chip Device Drivers
#
# CONFIG_MTD_DOC2000 is not set
# CONFIG_MTD_DOC2001 is not set
# CONFIG_MTD_DOC2001PLUS is not set

#
# NAND Flash Device Drivers
#
CONFIG_MTD_NAND=y
CONFIG_MTD_NAND_VERIFY_WRITE=y
# CONFIG_MTD_NAND_TOTO is not set
CONFIG_MTD_NAND_IDS=y
# CONFIG_MTD_NAND_DISKONCHIP is not set
# CONFIG_MTD_NAND_NANDSIM is not set
CONFIG_MTD_NAND_OMAP_HW=y

#
# Parallel port support
#
# CONFIG_PARPORT is not set

#
# Plug and Play support
#

#
# Block devices
#
# CONFIG_BLK_DEV_COW_COMMON is not set
CONFIG_BLK_DEV_LOOP=y
```

```
# CONFIG_BLK_DEV_CRYPTOLOOP is not set
# CONFIG_BLK_DEV_NBD is not set
# CONFIG_BLK_DEV_UB is not set
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_COUNT=16
CONFIG_BLK_DEV_RAM_SIZE=8192
CONFIG_BLK_DEV_INITRD=y
# CONFIG_CDROM_PKTCDVD is not set

#
# IO Schedulers
#
CONFIG_IOSCHED_NOOP=y
CONFIG_IOSCHED_AS=y
CONFIG_IOSCHED_DEADLINE=y
CONFIG_IOSCHED_CFQ=y
# CONFIG_ATA_OVER_ETH is not set

#
# ATA/ATAPI/MFM/RLL support
#
CONFIG_IDE=y
CONFIG_BLK_DEV_IDE=y

#
# Please see Documentation/ide.txt for help/info on IDE drives
#
# CONFIG_BLK_DEV_IDE_SATA is not set
CONFIG_BLK_DEV_IDEDISK=y
# CONFIG_IDEDISK_MULTI_MODE is not set
CONFIG_BLK_DEV_IDECS=y
# CONFIG_BLK_DEV_IDECD is not set
# CONFIG_BLK_DEV_IDETAPE is not set
# CONFIG_BLK_DEV_IDEFLOPPY is not set
# CONFIG_IDE_TASK_IOCTL is not set

#
# IDE chipset support/bugfixes
#
CONFIG_IDE_GENERIC=y
# CONFIG_IDE_ARM is not set
# CONFIG_BLK_DEV_IDEDMA is not set
# CONFIG_IDEDMA_AUTO is not set
# CONFIG_BLK_DEV_HD is not set

#
# SCSI device support
#
# CONFIG_RAID_ATTRS is not set
# CONFIG_SCSI is not set

#
# Multi-device support (RAID and LVM)
#
# CONFIG_MD is not set

#
# Fusion MPT device support
#
# CONFIG_FUSION is not set

#
# IEEE 1394 (FireWire) support
```

```
#

#
# I2O device support
#

#
# Network device support
#
CONFIG_NETDEVICES=y
# CONFIG_DUMMY is not set
# CONFIG_BONDING is not set
# CONFIG_EQUALIZER is not set
# CONFIG_TUN is not set

#
# PHY device support
#
# CONFIG_PHYLIB is not set

#
# Ethernet (10 or 100Mbit)
#
CONFIG_NET_ETHERNET=y
CONFIG_MII=y
CONFIG_SMC91X=y
# CONFIG_DM9000 is not set

#
# Ethernet (1000 Mbit)
#

#
# Ethernet (10000 Mbit)
#

#
# Token Ring devices
#

#
# Wireless LAN (non-hamradio)
#
CONFIG_NET_RADIO=y

#
# Obsolete Wireless cards support (pre-802.11)
#
# CONFIG_STRIP is not set
# CONFIG_PCMCIA_WAVELAN is not set
# CONFIG_PCMCIA_NETWAVE is not set

#
# Wireless 802.11 Frequency Hopping cards support
#
# CONFIG_PCMCIA_RAYCS is not set

#
# Wireless 802.11b ISA/PCI cards support
#
# CONFIG_HERMES is not set
# CONFIG_ATMEL is not set
```

```
#
# Wireless 802.11b Pcmcia/Cardbus cards support
#
# CONFIG_AIRO_CS is not set
# CONFIG_PCMCIA_WL3501 is not set
# CONFIG_HOSTAP is not set
CONFIG_NET_WIRELESS=y

#
# PCMCIA network device support
#
# CONFIG_NET_PCMCIA is not set

#
# Wan interfaces
#
# CONFIG_WAN is not set
CONFIG_PPP=y
CONFIG_PPP_MULTILINK=y
# CONFIG_PPP_FILTER is not set
# CONFIG_PPP_ASYNC is not set
# CONFIG_PPP_SYNC_TTY is not set
# CONFIG_PPP_DEFLATE is not set
# CONFIG_PPP_BSDCOMP is not set
# CONFIG_PPPOE is not set
# CONFIG_SLIP is not set
# CONFIG_SHAPER is not set
# CONFIG_NETCONSOLE is not set
# CONFIG_NETPOLL is not set
# CONFIG_NET_POLL_CONTROLLER is not set

#
# ISDN subsystem
#
# CONFIG_ISDN is not set

#
# Input device support
#
CONFIG_INPUT=y

#
# Userland interfaces
#
CONFIG_INPUT_MOUSEDEV=y
# CONFIG_INPUT_MOUSEDEV_PSAUX is not set
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
# CONFIG_INPUT_JOYDEV is not set
# CONFIG_INPUT_TSDEV is not set
CONFIG_INPUT_EVDEV=y
# CONFIG_INPUT_EVBUG is not set

#
# Input Device Drivers
#
# CONFIG_INPUT_KEYBOARD is not set
# CONFIG_INPUT_MOUSE is not set
# CONFIG_INPUT_JOYSTICK is not set
# CONFIG_INPUT_TOUCHSCREEN is not set
# CONFIG_INPUT_MISC is not set

#
```

```
# Hardware I/O ports
#
# CONFIG_SERIO is not set
# CONFIG_GAMEPORT is not set

#
# Character devices
#
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_HW_CONSOLE=y
# CONFIG_SERIAL_NONSTANDARD is not set

#
# Serial drivers
#
CONFIG_SERIAL_8250=y
CONFIG_SERIAL_8250_CONSOLE=y
# CONFIG_SERIAL_8250_CS is not set
CONFIG_SERIAL_8250_NR_UARTS=4
# CONFIG_SERIAL_8250_EXTENDED is not set

#
# Non-8250 serial port support
#
CONFIG_SERIAL_CORE=y
CONFIG_SERIAL_CORE_CONSOLE=y
CONFIG_UNIX98_PTYS=y
CONFIG_LEGACY_PTYS=y
CONFIG_LEGACY_PTY_COUNT=256

#
# IPMI
#
# CONFIG_IPMI_HANDLER is not set

#
# Watchdog Cards
#
# CONFIG_WATCHDOG is not set
CONFIG_OMAP16XX_RNG=y
# CONFIG_NVRAM is not set
CONFIG_RTC=y
CONFIG_OMAP_RTC=y
# CONFIG_DTLK is not set
# CONFIG_R3964 is not set

#
# Ftape, the floppy tape device driver
#

#
# PCMCIA character devices
#
# CONFIG_SYNCLINK_CS is not set
# CONFIG_RAW_DRIVER is not set

#
# TPM devices
#

#
# I2C support
```

```
#
CONFIG_I2C=y
CONFIG_I2C_CHARDEV=y

#
# I2C Algorithms
#
# CONFIG_I2C_ALGOBIT is not set
# CONFIG_I2C_ALGOPCF is not set
# CONFIG_I2C_ALGOPCA is not set

#
# I2C Hardware Bus support
#
# CONFIG_I2C_PARPORT_LIGHT is not set
# CONFIG_I2C_STUB is not set
# CONFIG_I2C_PCA_ISA is not set
CONFIG_I2C_OMAP=y

#
# Miscellaneous I2C Chip support
#
# CONFIG_SENSORS_DS1337 is not set
# CONFIG_SENSORS_DS1374 is not set
# CONFIG_SENSORS_EEPROM is not set
# CONFIG_SENSORS_PCF8574 is not set
# CONFIG_SENSORS_PCA9539 is not set
# CONFIG_SENSORS_PCF8591 is not set
# CONFIG_SENSORS_RTC8564 is not set
# CONFIG_ISP1301_OMAP is not set
CONFIG_TPS65010=y
CONFIG_SENSORS_TLV320AIC23=y
# CONFIG_GPIOEXPANDER_OMAP is not set
# CONFIG_SENSORS_MAX6875 is not set
# CONFIG_I2C_DEBUG_CORE is not set
# CONFIG_I2C_DEBUG_ALGO is not set
# CONFIG_I2C_DEBUG_BUS is not set
# CONFIG_I2C_DEBUG_CHIP is not set

#
# Hardware Monitoring support
#
CONFIG_HWMON=y
# CONFIG_HWMON_VID is not set
# CONFIG_SENSORS_ADM1021 is not set
# CONFIG_SENSORS_ADM1025 is not set
# CONFIG_SENSORS_ADM1026 is not set
# CONFIG_SENSORS_ADM1031 is not set
# CONFIG_SENSORS_ADM9240 is not set
# CONFIG_SENSORS_ASB100 is not set
# CONFIG_SENSORS_ATXP1 is not set
# CONFIG_SENSORS_DS1621 is not set
# CONFIG_SENSORS_FSCHER is not set
# CONFIG_SENSORS_FSCPOS is not set
# CONFIG_SENSORS_GL518SM is not set
# CONFIG_SENSORS_GL520SM is not set
# CONFIG_SENSORS_IT87 is not set
# CONFIG_SENSORS_LM63 is not set
# CONFIG_SENSORS_LM75 is not set
# CONFIG_SENSORS_LM77 is not set
# CONFIG_SENSORS_LM78 is not set
# CONFIG_SENSORS_LM80 is not set
# CONFIG_SENSORS_LM83 is not set
```

```
# CONFIG_SENSORS_LM85 is not set
# CONFIG_SENSORS_LM87 is not set
# CONFIG_SENSORS_LM90 is not set
# CONFIG_SENSORS_LM92 is not set
# CONFIG_SENSORS_MAX1619 is not set
# CONFIG_SENSORS_PC87360 is not set
# CONFIG_SENSORS_SMSC47M1 is not set
# CONFIG_SENSORS_SMSC47B397 is not set
# CONFIG_SENSORS_W83781D is not set
# CONFIG_SENSORS_W83792D is not set
# CONFIG_SENSORS_W83L785TS is not set
# CONFIG_SENSORS_W83627HF is not set
# CONFIG_SENSORS_W83627EHF is not set
# CONFIG_HWMON_DEBUG_CHIP is not set

#
# Misc devices
#

#
# Multimedia Capabilities Port drivers
#

#
# Multimedia devices
#
# CONFIG_VIDEO_DEV is not set

#
# Digital Video Broadcasting Devices
#
# CONFIG_DVB is not set

#
# Graphics support
#
CONFIG_FB=y
# CONFIG_FB_CFB_FILLRECT is not set
# CONFIG_FB_CFB_COPYAREA is not set
# CONFIG_FB_CFB_IMAGEBLIT is not set
CONFIG_FB_SOFT_CURSOR=y
# CONFIG_FB_MACMODES is not set
CONFIG_FB_MODE_HELPERS=y
# CONFIG_FB_TILEBLITTING is not set
# CONFIG_FB_S1D13XXX is not set
CONFIG_FB_OMAP=y
CONFIG_FB_OMAP_INTERNAL_LCDC=y
# CONFIG_FB_OMAP_EXTERNAL_LCDC is not set
CONFIG_FB_OMAP_DMA_TUNE=y
# CONFIG_FB_VIRTUAL is not set

#
# Console display driver support
#
# CONFIG_VGA_CONSOLE is not set
CONFIG_DUMMY_CONSOLE=y
CONFIG_FRAMEBUFFER_CONSOLE=y
CONFIG_FONTS=y
CONFIG_FONT_8x8=y
# CONFIG_FONT_8x16 is not set
# CONFIG_FONT_6x11 is not set
# CONFIG_FONT_7x14 is not set
# CONFIG_FONT_PEARL_8x8 is not set
```

```
# CONFIG_FONT_ACORN_8x8 is not set
# CONFIG_FONT_MINI_4x6 is not set
# CONFIG_FONT_SUN8x16 is not set
# CONFIG_FONT_SUN12x22 is not set
# CONFIG_FONT_10x18 is not set

#
# Logo configuration
#
CONFIG_LOGO=y
# CONFIG_LOGO_LINUX_MONO is not set
# CONFIG_LOGO_LINUX_VGA16 is not set
CONFIG_LOGO_LINUX_CLUT224=y
# CONFIG_BACKLIGHT_LCD_SUPPORT is not set

#
# Sound
#
CONFIG_SOUND=y

#
# Advanced Linux Sound Architecture
#
CONFIG_SND=y
CONFIG_SND_TIMER=y
CONFIG_SND_PCM=y
# CONFIG_SND_SEQUENCER is not set
CONFIG_SND_OSSEMUL=y
CONFIG_SND_MIXER_OSS=y
CONFIG_SND_PCM_OSS=y
# CONFIG_SND_RTCTIMER is not set
# CONFIG_SND_VERBOSE_PRINTK is not set
# CONFIG_SND_DEBUG is not set

#
# Generic devices
#
# CONFIG_SND_DUMMY is not set
# CONFIG_SND_MTPAV is not set
# CONFIG_SND_SERIAL_U16550 is not set
# CONFIG_SND_MPU401 is not set

#
# ALSA ARM devices
#
CONFIG_SND_OMAP_AIC23=y

#
# USB devices
#
# CONFIG_SND_USB_AUDIO is not set

#
# PCMCIA devices
#

#
# Open Sound System
#
# CONFIG_SOUND_PRIME is not set

#
# USB support
```

```
#
CONFIG_USB_ARCH_HAS_HCD=y
CONFIG_USB_ARCH_HAS_OHCI=y
CONFIG_USB=y
# CONFIG_USB_DEBUG is not set

#
# Miscellaneous USB options
#
CONFIG_USB_DEVICEFS=y
# CONFIG_USB_BANDWIDTH is not set
# CONFIG_USB_DYNAMIC_MINORS is not set
# CONFIG_USB_SUSPEND is not set
# CONFIG_USB_OTG is not set

#
# USB Host Controller Drivers
#
# CONFIG_USB_ISP116X_HCD is not set
CONFIG_USB_OHCI_HCD=y
# CONFIG_USB_OHCI_BIG_ENDIAN is not set
CONFIG_USB_OHCI_LITTLE_ENDIAN=y
# CONFIG_USB_SL811_HCD is not set

#
# USB Device Class drivers
#
# CONFIG_OBSOLETE_OSS_USB_DRIVER is not set

#
# USB Bluetooth TTY can only be used with disabled Bluetooth subsystem
#
# CONFIG_USB_ACM is not set
# CONFIG_USB_PRINTER is not set

#
# NOTE: USB_STORAGE enables SCSI, and 'SCSI disk support' may also be needed;
see USB_STORAGE Help for more information
#
# CONFIG_USB_STORAGE is not set

#
# USB Input Devices
#
# CONFIG_USB_HID is not set

#
# USB HID Boot Protocol drivers
#
# CONFIG_USB_KBD is not set
# CONFIG_USB_MOUSE is not set
# CONFIG_USB_AIPTEK is not set
# CONFIG_USB_WACOM is not set
# CONFIG_USB_ACECAD is not set
# CONFIG_USB_KBTAB is not set
# CONFIG_USB_POWERMATE is not set
# CONFIG_USB_MTOUCH is not set
# CONFIG_USB_ITMTOUCH is not set
# CONFIG_USB_EGALAX is not set
# CONFIG_USB_YEALINK is not set
# CONFIG_USB_XPAD is not set
# CONFIG_USB_ATI_REMOTE is not set
# CONFIG_USB_KEYSPAN_REMOTE is not set
```

```
# CONFIG_USB_APPLETOUCH is not set

#
# USB Imaging devices
#
# CONFIG_USB_MDC800 is not set

#
# USB Multimedia devices
#
# CONFIG_USB_DABUSB is not set

#
# Video4Linux support is needed for USB Multimedia device support
#

#
# USB Network Adapters
#
# CONFIG_USB_CATC is not set
# CONFIG_USB_KAWETH is not set
# CONFIG_USB_PEGASUS is not set
# CONFIG_USB_RTL8150 is not set
CONFIG_USB_USBNET=y
CONFIG_USB_NET_AX8817X=y
CONFIG_USB_NET_CDCETHER=y
# CONFIG_USB_NET_GL620A is not set
CONFIG_USB_NET_NET1080=y
# CONFIG_USB_NET_PLUSB is not set
# CONFIG_USB_NET_RNDIS_HOST is not set
# CONFIG_USB_NET_CDC_SUBSET is not set
CONFIG_USB_NET_ZAURUS=y
CONFIG_USB_ZD1201=y
CONFIG_USB_MON=y

#
# USB port drivers
#

#
# USB Serial Converter support
#
# CONFIG_USB_SERIAL is not set

#
# USB Miscellaneous drivers
#
# CONFIG_USB_EMI62 is not set
# CONFIG_USB_EMI26 is not set
# CONFIG_USB_AUERSWALD is not set
# CONFIG_USB_RIO500 is not set
# CONFIG_USB_LEGOTOWER is not set
# CONFIG_USB_LCD is not set
# CONFIG_USB_LED is not set
# CONFIG_USB_CYTHERM is not set
# CONFIG_USB_PHIDGETKIT is not set
# CONFIG_USB_PHIDGETSERVO is not set
# CONFIG_USB_IDMOUSE is not set
# CONFIG_USB_LD is not set
# CONFIG_USB_TEST is not set

#
# USB DSL modem support
```

```
#

#
# USB Gadget Support
#
# CONFIG_USB_GADGET is not set

#
# MMC/SD Card support
#
# CONFIG_MMC is not set

#
# Synchronous Serial Interfaces (SSI)
#
CONFIG_OMAP_UWIRE=y
# CONFIG_OMAP_TSC2101 is not set

#
# File systems
#
CONFIG_EXT2_FS=y
# CONFIG_EXT2_FS_XATTR is not set
# CONFIG_EXT2_FS_XIP is not set
# CONFIG_EXT3_FS is not set
# CONFIG_JBD is not set
# CONFIG_REISERFS_FS is not set
# CONFIG_JFS_FS is not set
# CONFIG_FS_POSIX_ACL is not set
# CONFIG_XFS_FS is not set
# CONFIG_MINIX_FS is not set
# CONFIG_ROMFS_FS is not set
CONFIG_INOTIFY=y
# CONFIG_QUOTA is not set
CONFIG_DNOTIFY=y
CONFIG_AUTOFS_FS=y
CONFIG_AUTOFS4_FS=y
# CONFIG_FUSE_FS is not set

#
# CD-ROM/DVD Filesystems
#
# CONFIG_ISO9660_FS is not set
# CONFIG_UDF_FS is not set

#
# DOS/FAT/NT Filesystems
#
CONFIG_FAT_FS=y
CONFIG_MSDOS_FS=y
CONFIG_VFAT_FS=y
CONFIG_FAT_DEFAULT_CODEPAGE=437
CONFIG_FAT_DEFAULT_IOCHARSET="iso8859-1"
# CONFIG_NTFS_FS is not set

#
# Pseudo filesystems
#
CONFIG_PROC_FS=y
CONFIG_SYSFS=y
CONFIG_TMPFS=y
# CONFIG_HUGETLB_PAGE is not set
CONFIG_RAMFS=y
```

```
# CONFIG_RELAYFS_FS is not set

#
# Miscellaneous filesystems
#
# CONFIG_ADFS_FS is not set
# CONFIG_AFFS_FS is not set
# CONFIG_HFS_FS is not set
# CONFIG_HFSPLUS_FS is not set
# CONFIG_BEFS_FS is not set
# CONFIG_BFS_FS is not set
# CONFIG_EFS_FS is not set
# CONFIG_JFFS_FS is not set
CONFIG_JFFS2_FS=y
CONFIG_JFFS2_FS_DEBUG=0
CONFIG_JFFS2_FS_WRITEBUFFER=y
# CONFIG_JFFS2_COMPRESSION_OPTIONS is not set
CONFIG_JFFS2_ZLIB=y
CONFIG_JFFS2_RTIME=y
# CONFIG_JFFS2_RUBIN is not set
CONFIG_CRAMFS=y
# CONFIG_VXFS_FS is not set
# CONFIG_HPFS_FS is not set
# CONFIG_QNX4FS_FS is not set
# CONFIG_SYSV_FS is not set
# CONFIG_UFS_FS is not set

#
# Network File Systems
#
CONFIG_NFS_FS=y
CONFIG_NFS_V3=y
# CONFIG_NFS_V3_ACL is not set
# CONFIG_NFS_V4 is not set
# CONFIG_NFS_DIRECTIO is not set
# CONFIG_NFSD is not set
CONFIG_ROOT_NFS=y
CONFIG_LOCKD=y
CONFIG_LOCKD_V4=y
CONFIG_NFS_COMMON=y
CONFIG_SUNRPC=y
# CONFIG_RPCSEC_GSS_KRB5 is not set
# CONFIG_RPCSEC_GSS_SPKM3 is not set
# CONFIG_SMB_FS is not set
# CONFIG_CIFS is not set
# CONFIG_NCP_FS is not set
# CONFIG_CODA_FS is not set
# CONFIG_AFS_FS is not set
# CONFIG_9P_FS is not set

#
# Partition Types
#
# CONFIG_PARTITION_ADVANCED is not set
CONFIG_MSDOS_PARTITION=y

#
# Native Language Support
#
CONFIG_NLS=y
CONFIG_NLS_DEFAULT="iso8859-1"
CONFIG_NLS_CODEPAGE_437=m
# CONFIG_NLS_CODEPAGE_737 is not set
```

```
# CONFIG_NLS_CODEPAGE_775 is not set
# CONFIG_NLS_CODEPAGE_850 is not set
# CONFIG_NLS_CODEPAGE_852 is not set
# CONFIG_NLS_CODEPAGE_855 is not set
# CONFIG_NLS_CODEPAGE_857 is not set
# CONFIG_NLS_CODEPAGE_860 is not set
# CONFIG_NLS_CODEPAGE_861 is not set
# CONFIG_NLS_CODEPAGE_862 is not set
# CONFIG_NLS_CODEPAGE_863 is not set
# CONFIG_NLS_CODEPAGE_864 is not set
# CONFIG_NLS_CODEPAGE_865 is not set
# CONFIG_NLS_CODEPAGE_866 is not set
# CONFIG_NLS_CODEPAGE_869 is not set
# CONFIG_NLS_CODEPAGE_936 is not set
# CONFIG_NLS_CODEPAGE_950 is not set
# CONFIG_NLS_CODEPAGE_932 is not set
# CONFIG_NLS_CODEPAGE_949 is not set
# CONFIG_NLS_CODEPAGE_874 is not set
# CONFIG_NLS_ISO8859_8 is not set
# CONFIG_NLS_CODEPAGE_1250 is not set
# CONFIG_NLS_CODEPAGE_1251 is not set
# CONFIG_NLS_ASCII is not set
CONFIG_NLS_ISO8859_1=m
# CONFIG_NLS_ISO8859_2 is not set
# CONFIG_NLS_ISO8859_3 is not set
# CONFIG_NLS_ISO8859_4 is not set
# CONFIG_NLS_ISO8859_5 is not set
# CONFIG_NLS_ISO8859_6 is not set
# CONFIG_NLS_ISO8859_7 is not set
# CONFIG_NLS_ISO8859_9 is not set
# CONFIG_NLS_ISO8859_13 is not set
# CONFIG_NLS_ISO8859_14 is not set
# CONFIG_NLS_ISO8859_15 is not set
# CONFIG_NLS_KOI8_R is not set
# CONFIG_NLS_KOI8_U is not set
# CONFIG_NLS_UTF8 is not set

#
# Profiling support
#
# CONFIG_PROFILING is not set

#
# Kernel hacking
#
# CONFIG_PRINTK_TIME is not set
# CONFIG_DEBUG_KERNEL is not set
CONFIG_LOG_BUF_SHIFT=14
CONFIG_DEBUG_BUGVERBOSE=y
CONFIG_FRAME_POINTER=y
# CONFIG_DEBUG_USER is not set

#
# Security options
#
# CONFIG_KEYS is not set
# CONFIG_SECURITY is not set

#
# Cryptographic options
#
CONFIG_CRYPTO=y
# CONFIG_CRYPTO_HMAC is not set
```

```
# CONFIG_CRYPTO_NULL is not set
# CONFIG_CRYPTO_MD4 is not set
# CONFIG_CRYPTO_MD5 is not set
# CONFIG_CRYPTO_SHA1 is not set
# CONFIG_CRYPTO_SHA256 is not set
# CONFIG_CRYPTO_SHA512 is not set
# CONFIG_CRYPTO_WP512 is not set
# CONFIG_CRYPTO_TGR192 is not set
# CONFIG_CRYPTO_DES is not set
# CONFIG_CRYPTO_BLOWFISH is not set
# CONFIG_CRYPTO_TWOFISH is not set
# CONFIG_CRYPTO_SERPENT is not set
CONFIG_CRYPTO_AES=y
# CONFIG_CRYPTO_CAST5 is not set
# CONFIG_CRYPTO_CAST6 is not set
# CONFIG_CRYPTO_TEA is not set
CONFIG_CRYPTO_ARC4=y
# CONFIG_CRYPTO_KHAZAD is not set
# CONFIG_CRYPTO_ANUBIS is not set
# CONFIG_CRYPTO_DEFLATE is not set
CONFIG_CRYPTO_MICHAEL_MIC=y
# CONFIG_CRYPTO_CRC32C is not set
# CONFIG_CRYPTO_TEST is not set

#
# Hardware crypto devices
#

#
# Library routines
#
# CONFIG_CRC_CCITT is not set
# CONFIG_CRC16 is not set
CONFIG_CRC32=y
# CONFIG_LIBCRC32C is not set
CONFIG_ZLIB_INFLATE=y
CONFIG_ZLIB_DEFLATE=y
```

12 Appendix E: Lua Cairo Bindings

Makefile:

```
all:  lcairo.c
      $(CC) $(CPPFLAGS) $(CPPFLAGS)/cairo $(CPPFLAGS)/glib-2.0 -O2 -fpic -c -o
lcairo.o lcairo.c
      $(CC) $(LDFLAGS) -shared -fpic -lglib-2.0 -lgobject-2.0 -lSDL -lcairo
-lpango-1.0 -lpangocairo-1.0 -o lcairo.so lcairo.o
```

lcairo.c:

```
#include "lauxlib.h"
#include "SDL/SDL_video.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <pango/pangocairo.h>
#include <sys/types.h>

// Cairo data types

#define GETCCONTEXT(l, n)     (cairo_t*)(lua_touserdata(l, n))
#define GETCSURFACE(l, n)     (cairo_surface_t*)(lua_touserdata(l, n))

// Pretend to be gluax for SDL module

#define UD_MAGIC_WORD  0xa0a0

static int tag_surface= 0;          // 'SDL_Surface*'

struct s_Userdata  // (just after the Lua-internal fields)
{
#ifdef UD_MAGIC_WORD
    uint magic;
#endif

    int tag;      // tells the 'type' of this userdata (emulates 4.0 tags)

      void* ptr;  // application-provided pointer (if any)

      lua_CFunction gc_func;  // garbage collection (object specific)

      // application-specific data block follows if 'size' mode is used.
};

// </pretend>


static void
draw_text (cairo_t *cr)
{
#define RADIUS 150
#define N_WORDS 10
#define FONT "Sans Bold 27"

  PangoLayout *layout;
  PangoFontDescription *desc;
  int i;
```

```
  /* Center coordinates on the middle of the region we are drawing
   */
  cairo_translate (cr, RADIUS, RADIUS);

  /* Create a PangoLayout, set the font and text */
  layout = pango_cairo_create_layout (cr);

  pango_layout_set_text (layout, "Paul", -1);
  desc = pango_font_description_from_string (FONT);
  pango_layout_set_font_description (layout, desc);
  pango_font_description_free (desc);

  /* Draw the layout N_WORDS times in a circle */
  for (i = 0; i < N_WORDS; i++)
    {
      int width, height;
      double angle = (360. * i) / N_WORDS;
      double red;

      cairo_save (cr);

      /* Gradient from red at angle == 60 to blue at angle == 300 */
      red   = (1 + cos ((angle - 60) * G_PI / 180.)) / 2;
      cairo_set_source_rgb (cr, red, 0, 1.0 - red);

      cairo_rotate (cr, angle * G_PI / 180.);

      /* Inform Pango to re-layout the text with the new transformation */
      pango_cairo_update_layout (cr, layout);

      pango_layout_get_size (layout, &width, &height);
      cairo_move_to (cr, - ((double)width / PANGO_SCALE) / 2, - RADIUS);
      pango_cairo_show_layout (cr, layout);

      cairo_restore (cr);
    }

  /* free the layout object */
  g_object_unref (layout);
}

int create_pic (lua_State *L)
{
  cairo_t *cr;
  cairo_surface_t *s;
  struct s_Userdata     *img_surface_handle, *img_data_registry_ref;
  unsigned char         *img_data;

  /* allocate some memory to write image data to. Also fake glua */
  //img_surface_handle = (struct s_Userdata*)lua_newuserdata (L, sizeof(struct
s_Userdata));
  img_surface_handle = (struct s_Userdata*)malloc(sizeof(struct s_Userdata));
  lua_pushlightuserdata(L, (void*)img_surface_handle);
  #ifdef UD_MAGIC_WORD
      img_surface_handle->magic = UD_MAGIC_WORD;
  #endif
  img_surface_handle->tag = tag_surface;
  img_surface_handle->gc_func = NULL;

  img_data = (unsigned char*)lua_newuserdata (L, 2*RADIUS*2*RADIUS*4);

  /* glua uses registry to keep img_data around until surface is freed... we'll
use it too */
```

```
  lua_pushstring(L, "glua_attach_");
  lua_pushnumber(L, (uint)img_surface_handle);
  lua_concat(L, 2);
  img_data_registry_ref = (struct s_Userdata*)lua_newuserdata (L, sizeof(struct
s_Userdata));
  img_data_registry_ref->magic = UD_MAGIC_WORD;
  img_data_registry_ref->tag = 0;
  img_data_registry_ref->gc_func = NULL;
  img_data_registry_ref->ptr = (void*)img_data;
  lua_settable(L, LUA_REGISTRYINDEX);

  s = cairo_image_surface_create_for_data(img_data, CAIRO_FORMAT_ARGB32,
2*RADIUS, 2*RADIUS, 2*RADIUS*4);

  img_surface_handle->ptr = (void*)SDL_CreateRGBSurfaceFrom(img_data, 2*RADIUS,
2*RADIUS,
                                    32, 2*RADIUS*4, 0xff000000, 0x00ff0000,
0x0000ff00,
                                    0x000000ff);


  cr = cairo_create(s);
  cairo_set_source_rgb (cr, 1.0, 1.0, 1.0);
  cairo_rectangle (cr, 0, 0, 2 * RADIUS, 2 * RADIUS);
  cairo_fill (cr);
  draw_text (cr);

  cairo_destroy (cr);

  cairo_surface_destroy (s);

  return 2;
}

int l_cairo_save(lua_State *L) {
      cairo_t *cr = GETCCONTEXT(L, 1);

      cairo_save(cr);
      return 0;
}

int l_cairo_restore(lua_State *L) {
      cairo_t *cr = GETCCONTEXT(L, 1);

      cairo_restore(cr);
      return 0;
}

int l_cairo_set_antialias(lua_State *L) {
      cairo_t     *cr = GETCCONTEXT(L, 1);
      double      antialias = lua_tonumber(L, 2);

      cairo_set_antialias(cr, antialias);
      return 0;
}

int l_cairo_set_line_width(lua_State *L) {
      cairo_t     *cr = GETCCONTEXT(L, 1);
      double      width = lua_tonumber(L, 2);

      cairo_set_line_width(cr, width);
      return 0;
}
```

```
int l_cairo_set_source_rgb(lua_State *L) {
      cairo_t     *cr = GETCCONTEXT(L, 1);
      double      r = lua_tonumber(L, 2),
                  g = lua_tonumber(L, 3),
                  b = lua_tonumber(L, 4);

      cairo_set_source_rgb(cr, r, g, b);
      return 0;
}

int l_cairo_set_source_rgba(lua_State *L) {
      cairo_t     *cr = GETCCONTEXT(L, 1);
      double      r = lua_tonumber(L, 2),
                  g = lua_tonumber(L, 3),
                  b = lua_tonumber(L, 4),
                  a = lua_tonumber(L, 5);

      cairo_set_source_rgba(cr, r, g, b, a);
      return 0;
}

int l_cairo_rectangle(lua_State *L) {
      cairo_t     *cr = GETCCONTEXT(L, 1);
      double      x1 = lua_tonumber(L, 2),
                  y1 = lua_tonumber(L, 3),
                  x2 = lua_tonumber(L, 4),
                  y2 = lua_tonumber(L, 5);

      cairo_rectangle(cr, x1, y1, x2, y2);
      return 0;
}

int l_cairo_stroke(lua_State *L) {
      cairo_t     *cr = GETCCONTEXT(L, 1);

      cairo_stroke(cr);
      return 0;
}

int l_cairo_fill(lua_State *L) {
      cairo_t     *cr = GETCCONTEXT(L, 1);

      cairo_fill(cr);
      return 0;
}

int l_pango_cairo_draw_text(lua_State *L) {
      cairo_t           *cr = GETCCONTEXT(L, 1);
      const char  *font = (const char*) luaL_checkstring(L, 2);
      const char  *text = (const char*) luaL_checkstring(L, 3);
      PangoFontDescription *desc = pango_font_description_from_string (font);
      PangoLayout *layout = pango_cairo_create_layout(cr);

      // Setup font
      pango_layout_set_text (layout, text, -1);
      pango_layout_set_font_description (layout, desc);
      pango_font_description_free (desc);

      // Draw
      cairo_save(cr);
      pango_cairo_update_layout (cr, layout);
      pango_cairo_show_layout (cr, layout);
```

```
	cairo_restore(cr);

	// Cleanup
	g_object_unref (layout);

	return 0;
}

int l_cairo_move_to(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);
	double	x = lua_tonumber(L, 2),
		y = lua_tonumber(L, 3);

	cairo_move_to(cr, x, y);
	return 0;
}

int l_cairo_line_to(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);
	double	x = lua_tonumber(L, 2),
		y = lua_tonumber(L, 3);

	cairo_line_to(cr, x, y);
	return 0;
}

int l_cairo_close_path(lua_State *L) {
	cairo_t *cr = GETCCONTEXT(L, 1);

	cairo_close_path(cr);
	return 0;
}

int l_cairo_translate(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);
	double	tx = lua_tonumber(L, 2),
		ty = lua_tonumber(L, 3);

	cairo_translate(cr, tx, ty);
	return 0;
}

int l_cairo_scale(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);
	double	sx = lua_tonumber(L, 2),
		sy = lua_tonumber(L, 3);

	cairo_scale(cr, sx, sy);
	return 0;
}

int l_cairo_rotate(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);
	double	rot = lua_tonumber(L, 2);

	cairo_rotate(cr, rot);
	return 0;
}

int l_cairo_reset_clip(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);

	cairo_reset_clip(cr);
```

```
	return 0;
}

int l_cairo_clip(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);

	cairo_clip(cr);
	return 0;
}

int l_cairo_clip_preserve(lua_State *L) {
	cairo_t	*cr = GETCCONTEXT(L, 1);

	cairo_clip_preserve(cr);
	return 0;
}

int l_cairo_image_surface_create(lua_State *L) {
	double			width = lua_tonumber(L, 1);
	double			height = lua_tonumber(L, 2);
	cairo_surface_t	*s;
	unsigned char	*img;

	img = lua_newuserdata(L, width*height*4);	// allocate room for 32bit
pixs
	s = cairo_image_surface_create_for_data(img, CAIRO_FORMAT_ARGB32,
                                                        width, height,
width*4);
	lua_pushlightuserdata(L, (void*)s);

	// change order from img, surface => surface, img

	lua_pushvalue(L, -2);
	lua_remove(L, -3);
	return 2;
}

int l_cairo_surface_destroy(lua_State *L) {
	cairo_surface_t *s = GETCSURFACE(L, 1);
	cairo_surface_destroy(s);
	return 0;
}

int l_cairo_context_create(lua_State *L) {
	cairo_surface_t	*s = GETCSURFACE(L, 1);
	cairo_t			*cr = cairo_create(s);
	lua_pushlightuserdata(L, (void*)cr);
	return 1;
}

int l_cairo_context_destroy(lua_State *L) {
	cairo_t		*cr = GETCCONTEXT(L, 1);
	cairo_destroy(cr);
	return 0;
}

int l_cairo_image_to_SDL_surface(lua_State *L) {
	unsigned char		*img = (unsigned char*) lua_touserdata(L, 1);
	double			width = lua_tonumber(L, 2);
	double			height = lua_tonumber(L, 3);
	struct s_Userdata *SDL_surface =
				(struct s_Userdata*) lua_newuserdata( L,
				sizeof(struct s_Userdata) );
```

```
	// Fake glua
	#ifdef UD_MAGIC_WORD
		SDL_surface->magic = UD_MAGIC_WORD;
	#endif
	SDL_surface->tag = tag_surface;
	SDL_surface->gc_func = NULL;

	// Allocate SDL surface
	SDL_surface->ptr =
			(void*)SDL_CreateRGBSurfaceFrom(img, width, height,
			32, width*4, 0x00ff0000, 0x0000ff00, 0x000000ff,
			0xff000000);

	return 1;
}


int setup_sdl_with_surface (lua_State *L) {     // we hacking bastards!
	struct s_Userdata *captured = lua_touserdata(L, 1);
	tag_surface = captured->tag;
	lua_pushnumber(L, tag_surface);
	return 1;
}

static const struct luaL_reg cairolib [] = {
	  {"create_pic", create_pic},
	{"setup_sdl_with_surface", setup_sdl_with_surface},

	{"cairo_save", l_cairo_save},
	{"cairo_restore", l_cairo_restore},
	{"cairo_set_antialias", l_cairo_set_antialias},
	{"cairo_set_line_width", l_cairo_set_line_width},
	{"cairo_image_surface_create", l_cairo_image_surface_create},
	{"cairo_surface_destroy", l_cairo_surface_destroy},
	{"cairo_context_create", l_cairo_context_create},
	{"cairo_context_destroy", l_cairo_context_destroy},
	{"cairo_image_to_SDL_surface", l_cairo_image_to_SDL_surface},
	{"cairo_move_to", l_cairo_move_to},
	{"cairo_line_to", l_cairo_line_to},
	{"cairo_close_path", l_cairo_close_path},
	{"cairo_translate", l_cairo_translate},
	{"cairo_scale", l_cairo_scale},
	{"cairo_rotate", l_cairo_rotate},
	{"cairo_reset_clip", l_cairo_reset_clip},
	{"cairo_clip", l_cairo_clip},
	{"cairo_clip_preserve", l_cairo_clip_preserve},
	{"pango_cairo_draw_text", l_pango_cairo_draw_text},
	{"cairo_set_source_rgb", l_cairo_set_source_rgb},
	{"cairo_set_source_rgba", l_cairo_set_source_rgba},
	{"cairo_rectangle", l_cairo_rectangle},
	{"cairo_stroke", l_cairo_stroke},
	{"cairo_fill", l_cairo_fill},
	  {NULL, NULL}  /* sentinel */
};

int luaLM_import(lua_State *L) {
	  luaL_openlib(L, "cairo", cairolib, 0);
	  return 0;
}

const char * luaLM_version( void )
{
```

```
        return LUA_VERSION;
}
```

13 Appendix F: Lua Application Code

data_objects.lua

```
-- Class to handle persistant time-based data
-- Loads and caches data from storage on demand, writes back data that
-- have been added
-- (NOTE: The persistant store needs to be redone... the code isn't very good.
-- also try to use linux hotplug events rather than scanning files)

-- If you need storage of real-time data, something like rrd would probably
-- be better. Sure, it loops around, but that's the performance tradeoff.
-- It's a good tradeoff because, in general, people are more concerned about
-- their current health

persist_data = {
      write_back_buffer = {},       -- Strong linked

      -- cache_objects = {},             -- Weak linked

      index = {
            time = {},                  -- default indices
            seq = {}
      },
      -- file_mapper = nil,             -- maps cache_objects to filenames on
write
      index_density = 100,              -- number of entries before one index
entry and file on disk

      sequence_number = 0,          -- every entry gets a unique sequence number

      -- persist_data constants
      CLEAN = 0,
      DIRTY = 1,
      UNLOADED = 2
}

function index_pair(min, max, reference)
      return {['min'] = min, ['max'] = max, ['reference'] = reference}
end

function cache_object(data, filename, state)
      return {['data'] = data, ['filename'] = filename, ['state'] = state}
end


-- make the data cache weak
-- setmetatable(persist_data.data_cache, {__mode = "v"})

function persist_data:new(object)
      object = object or {}
      setmetatable(object, self)
      self.__index = self
      return object
end

-- not used
function persist_data:find_indices(index_type, min, max)
      -- finds any index
      local the_index = self.index[index_type]
      local results = {}
```

```
	-- now find indices such that [min max] intersects with [v.min v.max]
	for i,v in pairs(the_index) do
		if	(v.min>=min and v.max<=max) or
			(v.min<min and v.max>=min) or
			(v.min<=max and v.max>max) then
			table.insert(results, v.reference)
		end
	end
	return results
end

function persist_data:index_index(index_type, index)
	-- this function gets the index index corresponding to a data index
	-- it's not the most efficient method, but it works.
	local the_index = self.index[index_type]
	for i,v in pairs(the_index) do
		if v.reference == index then
			return i
		end
	end
	return 0	-- darn, couldn't find it
end

function persist_data:update_index_for(curr_data_index, element)
	local index_type, index,c,d
	for index_type, index in pairs(self.index) do
		-- can element be referenced by this index?
		if element[index_type] then
			-- find the reference to our data_index in the indice
			local found_at_least_one = false
			for c,d in pairs(index) do
				if d.reference == curr_data_index then
					found_at_least_one = true
					-- test if the index needs to be updated
					if d.max < element[index_type] then
						d.max = element[index_type]
					end
					if d.min > element[index_type] then
						d.min = element[index_type]
					end
				end -- does this indice refer to our data array?
			end -- pairs(index elements)
			if not found_at_least_one then
				-- this data array is not indexed by index_type, yet
				-- so fix that
				table.insert(index, index_pair(
					element[index_type], element[index_type],
curr_data_index)
					)
			end
		end -- indexedby
	end -- pairs(indices)
end

function persist_data:get_range(index_type, min, max)
	-- first, find the data indices that may contain the data
	-- then search the indices for data within range
	local data_indices = self:find_indices(index_type, min, max)
	local results = {}
	local a,b,c,d
	for a,b in pairs(data_indices) do
		local data_array = self.cache_objects[b].data
```

```
            for c,d in pairs(data_array) do
                  if d[index_type] then               -- make sure the element is
indiced
                        if d[index_type]>=min and d[index_type]<=max then
                              table.insert(results, d)
                        end   -- bingo!
                  end   -- indiced
            end -- pairs(data)
      end -- pairs(data_indices)
      return results
end

function persist_data:unique(element)
      -- we only know how to check if glucometer data is unique
      if not element.type then return true end
      if element.type ~= "gluc" then return true end
      scan_data_hives = self:get_range("time", element.time, element.time)
      for i,v in pairs(scan_data_hives) do
            if v.value == element.value then
                  return false                        -- same glucose reading at
same time
            end
      end
      return true
end

-- This function is mostly for network transport
-- unlike serialization, it tries to pack a very specific structure (fmt)
-- with the parameters (...)
function persist_data:format(index_type, min, max, type, fmt, ...)
      local report_data = self:get_range(index_type, min, max)
      local results = ""
      local a, datum, index,d
      for a, datum in pairs(report_data) do
            if type == nil or datum.type == type then      -- filter data by
type
                  local newargs = {}
                  for index,d in arg do
                        newargs[index] = datum[arg[index]] or "" -- replace with
param
                  end
                  print(">",unpack(newargs),"<")
                  results = results .. string.format(fmt, unpack(newargs))
            end
      end
      return results
end

function persist_data:insert(element)
      -- is the data array empty? Bootstrap it
      if not self.cache_objects then
            -- no data, no current filename (not on disk), and dirty
            self.cache_objects = {cache_object({}, "", self.DIRTY)}
      end
      -- is the data already in the database?
      if not self:unique(element) then return end
      -- first check to see if the current data array is growing too large
      local curr_data_index = table.getn(self.cache_objects)
      local curr_record_index = table.getn(
                  self.cache_objects[curr_data_index].data
            )
      if curr_record_index >= self.index_density then
            -- outgrown array, make new one
```

```
            curr_data_index = curr_data_index + 1     -- new data array
            self.cache_objects[curr_data_index] = cache_object({}, "", true)
            for i,v in pairs(self.index) do                 -- for each indice
                  if element[i] then                              -- does indice
apply to element?
                        table.insert(v,
                              index_pair(element[i], element[i],
curr_data_index))
                  end
            end
      end
      local data_array = self.cache_objects[curr_data_index].data
      -- set sequence number
      self.sequence_number = self.sequence_number+1
      element.seq = self.sequence_number
      -- add our element
      table.insert(data_array, element)
      self.cache_objects[curr_data_index].state = self.DIRTY     -- make sure
we're dirty
      -- now update all indices for data_cache[curr_data_index] with new data
element
      self:update_index_for(curr_data_index, element)
end

function persist_data:serialize(iostream, indent, obj)
      if type(obj) == "number" then
            iostream:write(obj)
      elseif type(obj) == "string" then
            iostream:write(string.format("%q", obj))
      elseif type(obj) == "table" then
            iostream:write(indent, "{\n")
            for k,v in pairs(obj) do
                  iostream:write(indent, k, " = ")
                  self:serialize(iostream, indent.."\t", v)
                  iostream:write(",\n")
            end
            iostream:write(indent, "}\n")
      else
      error("cannot serialize a " .. type(obj))
      end
end

function persist_data:flush()
      if not self.cache_objects then return end
      -- write DIRTY data to disk
      local i, v, new_filename, file_data
      for i,v in pairs(self.cache_objects) do
            if v.state == self.DIRTY then
                  -- It's dirty, write it
                  new_filename = self.filemapper(v)
                  print(new_filename)
                  if new_filename then                -- some hives may not be
saved
                        tmp_filename = os.tmpname()
                        f, err = io.open(tmp_filename, "w")
                        if f then
                              self:serialize(f,"",v)
                              io.close(f)
                              -- mv the temporary file over the old file to
replace it
                              -- atomically
                              os.rename(tmp_filename, v.filename)
                              -- now rename it to the actual name we want. Since
```

```
the name
                              -- isn't really that important and would get fixed
on the
                              -- next flush, it is ok if we got interrupted
between the
                              -- first and 2nd rename operation
                              os.rename(v.filename, new_filename)
                              v.filename = new_filename
                              v.state = self.CLEAN
                        else
                              error(err)
                        end
                  end
            end
      end
end

function time_filename_mapper(obj)
      if not obj.data[1] then return nil end -- must have at least one datum
      smallest_time =   2147483640                    -- pretty big time for
smallest
      for i,v in pairs(obj.data) do
            if v.time then
                  if v.time<smallest_time then
                        smallest_time = v.time
                  end
            end
      end
      if smallest_time == 2147483640 then return nil end
      return string.format("/tmp/lua/%s.txt",         -- FIXME: check for
preexisting file
            os.date("%m%d%Y.txt",
            tonumber(smallest_time) ))
end

-- bi_data is a singleton

bi_data = {
      events = persist_data:new{          -- allow events to persist

      },

      params = {},                                -- associative list of parameters
                                                        -- doesn't persist atm

      files_to_scan = {},                         -- scan files for data from other
progs
      files_to_scan_delays = {},          -- delay between scans (s)
      files_to_scan_lasttimes = {}  -- last time scanned (s)
}

function bi_data:event(evt)
      self.events:insert(evt)
end

function bi_data:add_file_to_scan(file, delay, scan_function)
      self.files_to_scan[file] = scan_function
      self.files_to_scan_delays[file] = delay
      self.files_to_scan_lasttimes[file] = 0
end

function bi_data:scan_files()
      -- only scan files if they exist
```

```
	for file, scanner in pairs(self.files_to_scan) do
		-- check to see if we're due to scan it again
		if os.time() - self.files_to_scan_lasttimes[file]
			>= self.files_to_scan_delays[file] then

			self.files_to_scan_lasttimes[file] = os.time()
			f = io.open(file, "r")
			if f then
				if scanner(file, f) then
					-- if scanner "handled" the file, ie returned true
					-- then delete the file
					-- I expect most scanners won't delete files
					io.close(f)
					if posix then
						posix.unlink(file)			--
we should have posix
					else
						os.execute("rm -rf "..file)		-- not
great
					end
				else
					io.close(f)
				end
			end
		end
	end
end

-- file scanners

function wifi_signal_info_scanner(filename, iostream)
	data = iostream:read("*all")
	_, _, qlink, qlevel, qnoise = string.find(data, "(%d+)%.%s+(%d+)%.%s+(%
d+)%.%s+")
	if qnoise then
		bi_data.params['wi_link'] = qlink
		bi_data.params['wi_level'] = qlevel
		bi_data.params['wi_noise'] = qnoise
	end
	return false	-- do not kill the file
end

function wifi_network_scanner(filename, iostream)
	local data = iostream:read("*all")
	local networks = {}
	local i=0
	while true do
		i, _, essid = string.find(data, "ESSID:\"(.*)\"", i+1)
		if i == nil then break end
		table.insert(networks, essid) -- add network
	end
	wireless_networks = networks  -- make global
	return true -- kill the file
end

freestyle_parse_month_table = {
	['Jan'] = 1, ['Feb'] = 2, ['Mar'] = 3, ['Apr'] = 4, ['May'] = 5, ['Jun'] =
6,
	['Jul'] = 7, ['Aug'] = 8, ['Sep'] = 9, ['Oct'] = 10, ['Nov'] = 11,
	['Dec'] = 12
}

function freestyle_parse_datetime(h_datetime)
```

```
      -- first break down the string
      _, _, h_month, day, year, hour, minute =
                        string.find(tostring(data),
                        "(%a+)%s+(%d+)%s+(%d+)%s+(%d+):(%d+)")
      -- convert from human readable month to numeric month
      month = freestyle_parse_month_table[h_month]
      -- build time descriptor table
      datetime = {
            ['hour'] = hour,
            ['min'] = minute,
            ['day'] = day,
            ['year'] = year,
            ['month'] = month
      }
      return os.time(datetime)      -- generate UNIX epoch time
end

function skip_blank_lines(iostream)
      data = "non_nil"
      while data do
            data = iostream:read("*line")
            if data ~= "" then return data end
      end
end

function freestyle_data_scanner(filename, iostream)
      local data
      -- find serial number
      serial_no = tostring(skip_blank_lines(iostream) or "")
      unknown_data_line = skip_blank_lines(iostream)
      meter_date = tostring(skip_blank_lines(iostream) or "")
      if meter_date then
            meter_date = freestyle_parse_datetime(meter_date)
      end
      no_entries = tonumber(skip_blank_lines(iostream) or 0)
      data = ""
      while not string.find(data, "END") do     -- for each entry
            data = iostream:read("*line")
            if not data then  -- no more data
                  break
            end
            if data ~= "" then            -- parse line
                  _, _, reading, h_date_time, strip_id, unknown =
                        string.find(data,
                        "(%d+)%s+(%a+%s+%d+%s+%d+%s+%d+:%d+)%s+(%d+)%s+(0x%w+)")
                  if reading then
                        -- unix_meter_time = freestyle_parse_datetime
(h_date_time)
                        bi_data:event{
                              ['type'] = "gluc",
                              ['time'] = os.time(),
                              -- ['meter_time'] = unix_meter_time,
                              ['meter_id'] = serial_no,
                              ['strip_no'] = strip_id,
                              ['value'] = reading
                        }
                  end
            end
      end

      return true -- kill the file
end
```

```
function lua_data_scanner(filename, iostream)
      dofile(filename)
      return true
end

bi_data:add_file_to_scan("/proc/net/wireless", 1, wifi_signal_info_scanner)
bi_data:add_file_to_scan("/tmp/gluc_data", 1, lua_data_scanner)
-- bi_data:add_file_to_scan("/tmp/gluc_data", 1, freestyle_data_scanner)
bi_data:add_file_to_scan("/tmp/wifi_data", 1, wifi_network_scanner)

print(">",bi_data.events:format("seq", 1, 1000, "gluc", "\n") )

-- test functions
-- a = persist_data:new()
-- a.filemapper = time_filename_mapper

-- for i=1,200 do
--    local blah = {time = i}
--    a:insert(blah)
--    bi_data:event(blah)
--end
```

gui_app.lua

```
-- Create a container to contain both the scroll view and the icon bar
layout_container = gui_container:new()
function layout_container:get_preferred_size() return {x = 65535, y = 65535} end
my_screen:add_element(layout_container)

-- Create icon bar
my_icon_bar = icon_bar:new()
layout_container:add_element(my_icon_bar)

menu_scroll = nil

menu_stack = {}                    -- keeps track of old menus

function menu(the_menu)
      local focus_set = false;

      -- Each menu is kept in a FILO stack. This keeps track of where user is
      table.insert(menu_stack, menu_scroll)

      -- Now replace currently active menu with new one
      layout_container:remove_element(menu_scroll)
      menu_scroll = scroll_view:new()
      layout_container:add_element(menu_scroll)

      -- make sure user can back out of menu
      function menu_scroll:on_back()
            menu_go_back()
            return true             -- event handled
      end

      -- Build new menu
      for i,menu_item_data in pairs(the_menu) do
            local menu_item = gui_button:new(menu_item_data)
            menu_scroll:add_element(menu_item)
            if not focus_set then
                  menu_item:grab_focus()
                  focus_set = true
            end
```

```
	end

	-- tell screen to perform layout
	my_screen:layout_container()
end

function menu_go_back()
	local menu_stack_size = table.getn(menu_stack)

	-- check to see if we can go back
	if menu_stack_size < 1 or menu_stack[menu_stack_size] == nil then
		-- already at topmost menu
		return
	end

	-- take previous menu off of the stack
	local back_menu = menu_stack[menu_stack_size]
	table.remove(menu_stack)

	-- activate it
	layout_container:remove_element(menu_scroll)
	menu_scroll = back_menu
	layout_container:add_element(menu_scroll)

	-- restore focus
	layout_container.focus = menu_scroll
end

function wifiescape(text)
	local hex = ""
	local number
	for i = 1, string.len(text) do
		number = string.byte(text, i)
		hex = hex .. string.format("\\x%x", number)
	end
	return "`printf \""..hex.."\"`"
end

function connect_to_wifi(wifi)
	if not wifi then return end
	menu { [1] = { caption = "Connecting ..." } }
	print("iwconfig wlan0 essid "..wifiescape(wifi))
	-- would be nice if we could yield here!
	os.execute("iwconfig wlan0 essid \""..wifi.."\"")    -- THIS IS NOT SECURE
	menu_go_back()
end

function wifi_menu()
	local menuitems = {}
	if not wireless_networks then
		menu { [1] = {caption = "Turn Radio On", font = "Sans Bold 12",
			on_action = function () -- since DHCP doesn't work, we
manually set network
				os.execute("ifconfig wlan0 up; ifconfig wlan0
10.50.68.240 netmask 255.255.240.0;rm -rf /tmp/wifi_data")
				menu_go_back()
			end
			} }
		return
	end
	table.insert(menuitems,
		{	caption = "Turn Radio Off",
			font = "Sans Bold 12",
```

```
                on_action = function ()
                        os.execute("ifconfig wlan0 down; rm -rf /tmp/wifi_data")
                        wireless_networks = nil
                        menu_go_back()
                end
        })
    for i,network in pairs(wireless_networks) do
        local the_network = network
        table.insert(menuitems,
                {   caption = network,
                    font = "Sans Bold 12",
                    on_action = function ()
                            connect_to_wifi(the_network)
                    end
                })
    end
    menu(menuitems)
end

function messages_menu()
    menu {
        [1] = {
            caption = "Last 20\nmessages",
            font = "Sans Bold 12"
        },
        [2] = {
            caption = "Last week",
            font = "Sans Bold 12"
        },
        [3] = { caption = "1999" },
        [4] = { caption = "2000" },
        [5] = { caption = "2001" },
        [6] = { caption = "2002" },
        [7] = { caption = "2003" },
        [8] = { caption = "2004" },
    }
end

function to_do_menu()
    menu {
        [1] = {
            caption = "Lower morning\nglucose",
            font = "Sans Bold 12"
        },
        [2] = {
            caption = "Enjoy the day!",
            font = "Sans Bold 12"
        }
    }
end

function options_menu()
    menu {
        [1] = {
            caption = "Check for wifi\n every 10s",
            font = "Sans Bold 12"
        },
        [2] = {
            caption = "Graph Zoom\n(5x)",
            font = "Sans Bold 12"
        }
    }
end
```

```
-- The [1], [2], [3], etc are extraneous, but I think they look better that way.
menu {
	[1] = {
		caption = "Graph",
		on_action = function (self) print("graphing.") end
	},
	[2] = {
		caption = "To Do",
		on_action = to_do_menu
	},
	[3] = {
		caption = "Messages",
		on_action = messages_menu
	},
	[4] = {
		caption = "Wifi",
		on_action = wifi_menu
	},
	[5] = {
		caption = "Options",
		on_action = options_menu
	}
}
```

gui_objects.lua

```
gui = {}

-- misc util
function point(x, y)
	return { ["x"] = x, ["y"] = y }
end

-- vertical layout manager
function gui:vertical_layout_manager(operation)
	-- use minimum vertical size and total width

	self.elements = self.elements or {}

	local min_width = 0		-- the minimum width is the width of the
widest comp.
	local min_height = 0		-- the minimum height is the height of the
widest comp.
	for i,v in pairs(self.elements) do
		local min = v:get_min_size()
		if min.x > min_width then
			min_width = min.x
		end
		min_height = min_height + min.y
	end

	if operation == "preferred" then
		return point(65535, min_height)
	end
	if operation == "min" then
		return point(min_width, min_height)
	end
	if operation == "max" then
		return point(65535, 65535)
	end
	if operation == "layout" then
```

```
            -- set each component's size to their preferred heights
            -- and the component's width
            local insets = self:get_insets()
            local position = point(insets[1], insets[2])
            local cwidth = self:get_size().x - insets[1] - insets[3]
            local cheight_left = self:get_size().y - insets[2] - insets[4]

            for i,v in pairs(self.elements) do
                  -- print(self, " laying out ", i, v)
                  local pref = v:get_preferred_size()
                  v:set_location(point(position.x, position.y))

                  -- choose smaller of pref.y or remaining y area
                  local sety = math.min(cheight_left, pref.y)

                  -- set the component size
                  v:set_size(point(cwidth, sety))

                  -- print(" > ", position.x, position.y, cwidth, sety, " height
used ", sety, " height left ", cheight_left)

                  -- tell component to lay itself out
                  v:layout_container()

                  -- increment the vertical position and decrement room left
                  position.y = position.y + sety
                  cheight_left = cheight_left - sety
            end
      end
end

-- base clase for gui widgets
gui_container = {
      -- size = {0,0},
      -- elements = {},
      -- layout_manager = vertical_layout_manager()
}

function gui_container:new(object)
      object = object or {}
      setmetatable(object, self)
      self.__index = self
      return object
end

function gui_container:parent()
      return getmetatable(getmetatable(self))
end

function gui_container:add_element(element, layout_manager_flags)
      self.elements = self.elements or {} -- only way to have local non-
singleton

      -- a table element can't have a nil value, so we use an empty table
      -- if there are no layout manager flags
      layout_manager_flags = layout_manager_flags or {}

      -- insert element into list
      table.insert(self.elements, element)

      -- make us the element's parent
      element.parent_container = self
end
```

```
function gui_container:remove_element(element)
	self.elements = self.elements or {}
	for i,v in pairs(self.elements) do
		if v == element then
			table.remove(self.elements, i)
		end
	end
end

function gui_container:get_preferred_size()
	self.layout_manager = self.layout_manager or gui.vertical_layout_manager
	return self:layout_manager("preferred")
end

function gui_container:get_max_size()
	self.layout_manager = self.layout_manager or gui.vertical_layout_manager
	return self:layout_manager("max")
end

function gui_container:get_min_size()
	self.layout_manager = self.layout_manager or gui.vertical_layout_manager
	return self:layout_manager("min")
end

function gui_container:get_size(the_size)
	self.size = self.size or point(0,0)
	return self.size
end

function gui_container:set_size(the_size)
	self.size = the_size
end

function gui_container:get_insets()
	return {0, 0, 0, 0}
end

function gui_container:get_location(the_location)
	self.location = self.location or point(0,0)
	return self.location
end

function gui_container:set_location(the_location)
	self.location = the_location
end

function gui_container:layout_container()
	self.layout_manager = self.layout_manager or gui.vertical_layout_manager
	self:layout_manager("layout")
end

-- The idea behind the events it that the event goes to the object with focus.
-- If that object doesn't handle it, it bubbles up the hierarchy until it finds
-- an object that does handle it or it can't bubble anymore

function gui_container:handle_event(event)
	return false		-- we do not handle events by default
end

function gui_container:focus_event(event)
	-- check to see if there is a focused object
	if self.focus ~= nil and self.focus ~= self then
```

```
            -- if there is, then send it the event
            self.focus:focus_event(event)
      else
            -- otherwise start bubbling the event up
            self:bubble_event(event)
      end
end

function gui_container:bubble_event(event)
      -- does this object consume it?
      if not self:handle_event(event) then
            -- percolate up
            if self.parent_container ~= nil then     -- I should have a parent
if I got an event
                  self.parent_container:bubble_event(event)
            end
      end
end

function gui_container:do_grab_focus(object)
      -- helper function for grab_focus that actually does the work
      self.focus = object
      if self.parent_container ~= nil then
            self.parent_container:do_grab_focus(self)
      end
end

function gui_container:grab_focus()
      self:do_grab_focus(self)
end

function gui_container:paint(cr)
      self.elements = self.elements or {} -- only way to have local non-
singleton
      -- if we're debugging, draw container rectangle
      pos = self:get_location()
      size = self:get_size()
      -- cairo.cairo_set_source_rgba(cr, 0, 0, 0, 1)
      -- cairo.cairo_rectangle(cr, 0, 0, size.x, size.y)
      -- cairo.cairo_stroke(cr)
      -- iterate through contained elements and draw them
      for i,v in pairs(self.elements) do
            loc = v:get_location()                                    -- get
container loc
            size = v:get_size()                                             --
get container size
            cairo.cairo_save(cr)                                      -- save
paint context
                  cairo.cairo_translate(cr, loc.x, loc.y)        -- set origin
                  cairo.cairo_rectangle(cr, 0, 0, size.x, size.y)      -- clip
bounds
                  cairo.cairo_clip(cr)                                -- set
clip
                  v:paint(cr)                                               --
paint container
            cairo.cairo_restore(cr)                                   --
restore context
      end
end

-- screen widget
gui_screen = gui_container:new{
      width = 240,
```

```
      height = 160,
      bpp = 16
}

function gui_screen:init()
      self:init_SDL()
      self:sdl_cairo_hack()
      self:init_cairo()
end

function gui_screen:init_SDL()
      -- Initialize SDL
      if SDL.SDL_Init(SDL.SDL_INIT_VIDEO) < 0 then
            print("Couldn't initialize SDL: ",SDL.SDL_GetError())
            -- exit(1)
      end
      videoflags = SDL.SDL_HWSURFACE + SDL.SDL_ANYFORMAT
print("setting video mode")
self.surface = SDL.SDL_SetVideoMode(self.width, self.height,
                  self.bpp, 0)
print("set video mode!\n")
      if self.surface == nil then
            print("Couldn't set ", self.width, "x", self.height,
                        " video mode: ", SDL.SDL_GetError())
            -- exit(2)
      end
      if not cheia then -- if not on Paul's computer
            SDL.SDL_ShowCursor(0)
      end
end

function gui_screen:init_cairo()
      self.cairo_surface, self.img_data =
            cairo.cairo_image_surface_create(self.width, self.height)
      self.sdl_cairo_surface =
            cairo.cairo_image_to_SDL_surface(self.img_data, self.width,
self.height)
      self.cr = cairo.cairo_context_create(self.cairo_surface)
end

function gui_screen:sdl_cairo_hack()
      test_surface = SDL.SDL_CreateRGBSurface(0, 10, 10, 4, 1,1,1,1)
      cairo.setup_sdl_with_surface(test_surface)
      SDL.SDL_FreeSurface(test_surface);
end

function gui_screen:close()
      cairo.cairo_context_destroy(self.cr)
      cairo.cairo_surface_destroy(self.cairo_surface)
      SDL.SDL_FreeSurface(self.sdl_cairo_surface)
      SDL.SDL_Quit()
end

function gui_screen:paint()
      -- first erase cairo screen
      cairo.cairo_set_source_rgba(self.cr, 1, 1, 1, 1)
      cairo.cairo_rectangle(self.cr, 0, 0, self.width, self.height)
      cairo.cairo_fill(self.cr)

      -- now get all gui components painted
      self:parent().paint(self, self.cr)

      if not temp_rect then
```

```
            temp_rect = {}
            temp_rect.x = 0
            temp_rect.y = 0
            temp_rect.w = self.width
            temp_rect.h = self.height
      end

      -- now do screen flip
      SDL.SDL_BlitSurface(self.sdl_cairo_surface, NULL, self.surface, temp_rect)
      SDL.SDL_UpdateRect(self.surface, 0, 0, 0, 0)
end

function gui_screen:get_size()
      return point(self.width, self.height)
end

function gui_screen:get_location()
      return point(0,0)
end

-- button widget
gui_button = gui_container:new()

function gui_button:get_preferred_size()
      return point(65535, 40)
end

function gui_button:handle_event(event)
      if event.type == SDL.SDL_KEYDOWN then
            local sym = event.key.keysym.sym
            if sym == SDL.SDLK_RIGHT or sym == 13 then
                  self:on_action()
                  return true
            end
      end
      return false
end

function gui_button:on_action()
end

function gui_button:paint(cr)
      self:parent().paint(self, cr)             -- let parent set up
      self.caption = self.caption or ""
      self.font = self.font or "Sans Bold 20"

      if self.focus == self then                      -- we have the focus
            -- white on black
            local size = self:get_size()
            cairo.cairo_set_source_rgba(cr, 0, 0, 0, 1)
            cairo.cairo_rectangle(cr, 0, 0, size.x, size.y)
            cairo.cairo_fill(cr)
            cairo.cairo_set_source_rgba(cr, 1, 1, 1, 1)
      else                                                  -- someone else has
focus
            -- blue on white
            cairo.cairo_set_source_rgba(cr, 0, 0, 1, 1)
      end
      cairo.pango_cairo_draw_text(cr, self.font, self.caption)
end

-- scroll widget
```

```
scroll_view = gui_container:new()

function scroll_view:get_preferred_size()
      return point(65535, 65535)
end

function scroll_view:get_insets()
      return {0, 0, 15, 0}    -- 15 pixels for scroll bar
end

function scroll_view:on_back()
      return false                  -- not handled by default
end

function scroll_view:handle_event(event)
      local rel_select = nil
      if event.type == SDL.SDL_KEYDOWN then
            local sym = event.key.keysym.sym
            if sym == SDL.SDLK_UP then
                  rel_select = -1
            end
            if sym == SDL.SDLK_DOWN then
                  rel_select = 1
            end
            if sym == SDL.SDLK_LEFT then
                  return self:on_back()
            end
      end
      if rel_select ~= nil then
            -- get position associated with focused object
            local found = 0
            for i,v in pairs(self.elements) do
                  if v == focus then
                        found = i
                        break
                  end
            end
            if found + rel_select > 0 then           -- make sure new selection
is within
                  if found + rel_select <= table.getn(self.elements) then    --
range
                        if focus ~= nil then
                              focus.focus = nil            -- hack; should
percolate loss
                                                                   -- of
focus down
                        end
                        focus = self.elements[found+rel_select]
                        focus:grab_focus()                 -- bit of a hack
                                                                  -- we should
percolate focus down
                                                                  -- however it's
tough to know which
                                                                  -- objects
support getting focus
                        return true;
                  end
            end
      end
      return false
end

function scroll_view:paint(cr)
```

```
	self:parent().paint(self, cr)			-- let parent set up

	-- get our size
	size = self:get_size()

	-- draw scroll bar
	cairo.cairo_set_source_rgba(cr, 0, 0, 0, 1)
	cairo.cairo_set_line_width(cr, 2)
	cairo.cairo_rectangle(cr, size.x-15, 0, size.x, size.y)
	cairo.cairo_stroke(cr)

	-- draw scroll indicator
	cairo.cairo_set_source_rgba(cr, 0.3, 0.3, 0.3, 0.3)
	cairo.cairo_rectangle(cr, size.x-13, 0, 12, 20)
	cairo.cairo_fill(cr)
end

-- icon bar widget

icon_bar = gui_container:new()

function icon_bar:get_preferred_size()
	return point(65535, 15)
end

function icon_bar:paint(cr)
	self:parent().paint(self, cr)		-- let parent set up

	-- get our size
	local size = self:get_size()

	-- draw icon bar
	cairo.cairo_set_source_rgba(cr, 0, 0, 0, 1)
	cairo.cairo_set_line_width(cr, 1)
	cairo.cairo_rectangle(cr, 0, 0, size.x, 15)
	cairo.cairo_stroke(cr)

	-- draw battery icon
	cairo.cairo_set_source_rgba(cr, 0, 1, 0, 1)
	cairo.cairo_rectangle(cr, 3, 3, 18, 15-6)
	cairo.cairo_stroke(cr)
end

-- graph widget

graph_widget = gui_container:new()

function graph_widget:get_preferred_size()
	return point(65535, 65535)
end

function graph_widget:paint(cr)
	self:parent().paint(self, cr)		-- let parent set up

	-- get our size
	local size = self:get_size()

	-- draw box
	cairo.cairo_set_source_rgba(cr, 0, 0, 0, 1)
	cairo.cairo_set_line_width(cr, 1)
	-- leave a little room for the y axis gradient bar and x axis labels
	cairo.cairo_rectangle(cr, 20, 0, size.x-20, size.y-30)
	cairo.cairo_stroke(cr)
```

```
	-- set up points
	local pts = {point(1,1), point(3,5), point(6,2), point(9,7)}

	-- get range (assume pts are ordered)
	local range_max = pts[getn(pts)]
	local range_min = pts[1]
	local range = range_max - range_min

	-- get domain
	local domain_max = 0
	local domain_min = 65535
	for i,v in pairs(pts) do
		if v.y > domain_max then
			domain_max = v.y
		end
		if v.y < domain_min then
			domain_min = v.y
		end
	end
	local domain = domain_max - domain_min

	-- draw gradient bar
	cairo.cairo_rectangle(cr, 3, 3, 20-3-3, size.y-30)
	cairo.cairo_stroke(cr)

	-- graph points
	for x = 1,size.x-40 do
		-- convert x coordinate to point
	end
end
```

main.lua

```
if cheia then
	-- stuff for pauls computer
	cheia.load("SDL")
else
	-- load for everyone elses
	f = loadlib("../modules/gluahost.so", "luaLM_import")
	f()
	-- This app requires the luaSDL binding.
	assert(gluahost.gluahost("../modules/sdl_module.so","SDL"))
end

dofile("gui_objects.lua")


f = loadlib("../modules/lcairo.so", "luaLM_import")
f()
f = loadlib("../modules/lposix.so", "luaopen_posix")
f()

dofile("data_objects.lua")

print("loaded stuff")

-- Small fix for Paul's computer... uses a modified version of lposix.so
-- Person who compiled lua for OS X didn't enable the appropriate functions
-- remove if not useful to you (probably the case)
if posix.io_popen then
	io.popen = posix.io_popen
```

```
	io.close = posix.io_close
	os.tmpname = function()
		return "/tmp/lua_paul_bridgeit"
	end
end

-- Lua testing grounds

dofile("server_protocol.lua")

function event_manager()
	event_buffer = event_buffer or SDL.SDL_Event_new()
	while SDL.SDL_PollEvent(event_buffer) ~= 0 do
		if event_buffer.type == SDL.SDL_QUIT then
			running = false
		end
		my_screen:focus_event(event_buffer) -- send event to focus
	end
end

running = true

-- Create a screen to output stuff
my_screen = gui_screen:new()
my_screen:init()

dofile("gui_app.lua")

if server_protocol then
	server_protocol:do_update()
end

while running do
	server_protocol:update_task()
	bi_data:scan_files()
	event_manager()
	my_screen:paint()
end

my_screen:close()
```

server_protocol.lua

```
-- This implements the server protocol
-- Basic idea: ask server for script, then run the script.

-- Current implementation blocks if server is not available. This should be
-- modified to pass a timeout to wget and some form of select() call to
-- see if wget has returned data... or maybe an asynch read call would work
-- better semantic-wise

server_protocol = {
	url = "http://10.50.68.77/synch.php",
	errurl = "http://10.50.68.77/error.php"
	-- url = "http://127.0.0.1/~michaelingemi/sim/synch.php",
	-- errurl = "http://127.0.0.1/~michaelingemi/sim/synch.php"
}

-- inherit from io
-- setmetatable(server_protocol, {__index = io})

function server_protocol:new(obj)
```

```
	obj = obj or {}
	setmetatable(obj, self)
	self.__index = self
	return obj
end

function server_protocol:htmlencode(text)
	local hex = ""
	local number
	for i = 1, string.len(text) do
		number = string.byte(text, i)
		if number<16 then
			hex = hex .. string.format("%%0%x", number)
		else
			hex = hex .. string.format("%%%x", number)
		end
	end
	return hex
end

function server_protocol:open()
	assert(not self.file_handle, "already opened!")
	assert(self.task, "no task defined.")

	-- set up temporary file to use to pass POST data
	self.tmp_file_name = os.tmpname()
	assert(self.tmp_file_name, "Can't get a temporary file name!")
	self.tmp_file_handle = io.open(self.tmp_file_name, "w")
	assert(self.tmp_file_handle, "Can't get a temporary file handle!")

	if self.data then
		-- stick data in a POST variable... just in case its very long
		self.tmp_file_handle:write(
			string.format("data=%s", server_protocol:htmlencode
(self.data))
		)
	end
	io.close(self.tmp_file_handle)

	-- seq number
	seq = 0
	if bi_data then
		seq = bi_data.events.sequence_number;
	end

	-- id & pass
	logger_id = logger_id or 1
	logger_password = logger_password or "long1024bitpass"

	-- open wget
	self.file_handle = io.popen(
		string.format(
			"wget --post-file %s -q -O - \"%s?ver=0.1&seq=%s&li=%s&lp=%s&%
s\"",
			self.tmp_file_name, self.url, seq, logger_id, logger_password,
			self.task)
		)

	-- Real cheap form of inheritence from io
	self.read = function(self, ...)
		return self.file_handle:read(unpack(arg))
	end
end
```

```
function server_protocol:close()
	self.file_handle = self.file_handle or {}
	io.close(self.file_handle)

	-- make sure we remove the temporary POST data file
	if self.tmp_file_handle then
		os.remove(self.tmp_file_name)
	end
end

function server_protocol:report_error(err)
	local s = server_protocol:new{
		task = "error",
		data = err
	}
	s:open()
	d = s:read("*a")
	s:close()
end

function server_protocol:send_task_data_1way(the_task, the_data)
	local s = server_protocol:new{
		task = the_task,
		data = the_data
	}
	s:open()
	local d = s:read("*a")
	s:close()
	print("oneway send: ",d)
end

function server_protocol:do_update()
	local s = server_protocol:new {
		task = "update"
	}
	s:open()
	local task = s:read("*all")
	err = s:close()
	if err == 0 or err == nil then
		print(task)
		local ok, err = pcall(function() loadstring(task)() end)
		if ok then
			-- pcall succeeded, task must not have erred
			return true
		else
			-- pcall failed, task erred
			print("failed: " .. err)
			server_protocol:report_error(err)
			return false
		end
	else
		return false
	end
end

function wait_yield(amt)		-- would've used this for update_task if I
could get coroutines working
	local last_time = os.time()
	while os.time()-last_time<amt do
		coroutine.yield()
	end
end
```

```
server_protocol.state = 1           -- coroutines aren't working "attempt to
yield across metamethod/C-call boundary"
                                                -- so I'm using a state machine.
State 1 = last conn.
                                                -- successful, state 2 = last
conn. unsucessful
server_protocol.last_time = os.time()

function server_protocol:update_task()
      --if not wireless_networks then return end     -- for now, don't poll when
you don't have a net
      if self.state == 1 then
            if os.time() - self.last_time < 5 then return end
      end
      if self.state == 2 then
            if os.time() - self.last_time < 1 then return end
      end
      print("doing update")
      if server_protocol:do_update() then
            self.state = 1
      else
            self.state = 2
      end
      self.last_time = os.time()
end
```