Tanzania Humanitarian Project

Internet-On-Demand System using GSM Data Link

Team 1: Drew Newell, Jeff Pawloski, Ivan Gallardo, Qilong Gou, Kenneth Jordan

4/27/2012
Table of Contents

Acknowledgements...........................................................................................................2
Chapter 1..........................................................................................................................3
  Background.....................................................................................................................3
  Introduction....................................................................................................................3
Chapter 2.........................................................................................................................4
Chapter 3.........................................................................................................................8
  GSM Communication......................................................................................................8
  Ethernet Monitor.........................................................................................................12
  Power-over-Ethernet Control......................................................................................17
  Power Supply...............................................................................................................19
  Battery Monitor..........................................................................................................22
Chapter 4.........................................................................................................................25
Chapter 5.........................................................................................................................34
Appendix I......................................................................................................................36
Appendix II.....................................................................................................................42
Appendix III....................................................................................................................43
Acknowledgements

We would like to take the opportunity to thank the following groups and individuals for their assistance in the completion of this project:

Dr. Timothy Hogan
Dr. Erik Goodman
Dr. Jennifer Olson
Dr. Kurt DeMaagd
Dow Chemical Corporation
Eric Tarkleson
Roxanne Peacock
Michigan State University Office of Study Abroad
Chapter 1

Background

Beginning in 2008, MSU faculty and students have traveled to Mto wa Mbu, Tanzania to help improve the quality of education through technological advancement. Over the course of 4 years, the teams that have traveled over there have installed computer labs, a satellite uplink to connect to the internet, and solar panels. The solar panels charge batteries that are used to power the satellite system and the computer labs at schools that are not on the electric power grid.

Because the satellite system is connected directly to the solar batteries, it is only turned on for a 4 hour period each school day so as not to completely drain the battery. This means that the internet is only accessible for a limited time each day. Teachers wanting to use the internet as a tool must plan their day around this 4 hour period which can be a major inconvenience. Also, they must finish using the internet before the end of the period because the system shuts down regardless of any internet activity.

In the summer of 2011, a design team for ECE 480 attempted to create an internet-on-demand system so that these schools could use the internet at any time. Their design was based largely on the use of radio frequency (RF) antennae to transmit FM signals from each school to the satellite system. In theory, each school would be able to remotely access the internet with a radio signal and turn it off using the same method. This group was unsuccessful in implementing their design due in large part to line-of-sight issues between RF towers in Tanzania.

Introduction

This year the same project was given in the hopes that a successful method for internet-on-demand could be realized. Through the use of problem solving methods described in Chapter 2, it was determined that the Global Satellite for Mobile (GSM) network could be used to create the internet-on-demand system without running into the same issues that previous groups had encountered.

The GSM is a telecommunications network used for transmitting both voice and data messages in countries worldwide, including Tanzania. Utilizing the GSM network will allow for signal transmission without running into the line-of-sight issues that previous groups had when using the RF antennae.
Microcontrollers will be necessary to control the signals sent from the schools to the satellite uplink station. The Arduino Uno microcontroller was selected for its combination of price and efficiency. A device would be placed at each school with a push button. When the button is pushed, the Arduino on the device at the school would send a signal to the Arduino placed at the satellite station to turn on the satellite.

Since the satellite is powered by the solar batteries, power conservation is a top concern. Constant depletion and recharging of the batteries would cause them to need to be replaced often. To keep the batteries at maximum life expectancy they should only be in use when necessary. With this in mind, an internet monitor will be placed on the Arduino at the satellite station that will shut down the satellite system after 15 minutes if no activity is detected on the internet.

Another step to conserve power is to monitor the battery capacity. Ideally, the battery would stay above 50% capacity in order to achieve the maximum lifetime. With the battery monitor, if the power in the battery drops below 50% of capacity, the system will shut down so as not to completely drain the battery. The monitor will constantly check the battery to make sure the system is not in operation at low capacity.

During the process of analyzing the existing system for the internet, it was found that the router for the internet system had a WiMAX antenna for each school. Each antenna draws power. In the proposed system, it does not make sense to turn on every antenna if only one school requires the internet. With this in mind, a Power-over-Ethernet (POE) system was created that will turn on only the antenna for the school that requested the internet. This will reduce the amount of power drawn on the batteries when only one school wants to use the internet.
Chapter 2

To implement the project, several subsystems needed to be identified and evaluated. Throughout this evaluation process, the customer’s requirements were always the most influential factor. To help the team clearly recognize what systems were needed, how they were to be used and why they were needed, a FAST diagram of was developed for the project.

Figure 1 – FAST Diagram for the Internet-On-Demand project
The primary objective of the project was to allow peripheral schools to access the internet on-demand. To perform this task, the satellite and modem need to be activated, system vitals need to be checked and the router needs to be powered. The first task is common to almost all the systems and that is to energize the power inverter. The power inverter is energized only when it receives the proper signal from the Arduino, which receives the GSM request signal. Before any parts of the system are energized, however, the system checks the health of the batteries to determine if the system can be powered.

Through several interviews with the sponsor, critical customer requirements were identified for the project. One requirement was that the interface with the system should be very simple. Though this requirement did not depend on the selection matrix outcome, the problem was addressed by using a one button input system. Another requirement of the project was that it should be low powered with a self-diagnostic system in place. The final requirement was that it used a reliable technology to perform the data link objective.

To address the critical customer requirements, the following solutions were evaluated.

<table>
<thead>
<tr>
<th>Engineering Criteria</th>
<th>Importance</th>
<th>Possible Solutions for Remote Internet on Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Between Remote Locations</td>
<td>5</td>
<td>GSM Link 9</td>
</tr>
<tr>
<td>Data Processing</td>
<td>2</td>
<td>Radio Link 1</td>
</tr>
<tr>
<td>Minimal Power Consumption</td>
<td>3</td>
<td>Wireline/Fiber optics 9</td>
</tr>
<tr>
<td>Internet Monitoring</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Hardware Platform</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Maturity of technology</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Stability/Safety</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>132 100 164</td>
</tr>
</tbody>
</table>

*Figure 2 – Solution Selection Matrix*

Based on Figure 2, the ideal solution from a purely engineering stand point is to use wireline communications. Wireline and fiber optic communications represent a mature technology
which is highly reliable. The GSM Link alternative was more attractive than the radio link option from an engineering stand point but it did not necessarily represent the best option. However, after a feasibility matrix (Figure 3) was constructed, it became obvious that a GSM link was the best option for the project.

From analysis of the feasibility matrix, the GSM link option was chosen. The wireline option would require too much time and money for the scope of this project. The radio link option was discarded because of the line-of-sight issues that last year’s team encountered. Both the selection matrix and the feasibility matrix indicated that using the existing GSM network in Tanzania was the best way to execute our sponsor’s requirements.

<table>
<thead>
<tr>
<th>Feasibility</th>
<th>Importance</th>
<th>Possible Solutions for Remote Internet on Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GSM Link</td>
</tr>
<tr>
<td>Cost</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Technical</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Safety/Stability</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Time</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>93</td>
</tr>
</tbody>
</table>

**Figure 3 – Feasibility Matrix**
Chapter 3

Communication between schools – A GSM Solution

The GSM network is the most widely used wireless communication method around the globe. The advantage of using this network is that there is a very large source of high quality information. In addition to the use of a GSM Module, the Arduino platform is a widely open-source micro-processing platform with extensive tutorials and examples freely available on the web. These advantages led the team to develop and interface a communication method using the Arduino Uno (R3) and the Telit SM5100B GSM Module that is widely used in cellular devices all over the world.

Figure 4 – Arduino Uno, Antenna Module, and the GSM Module

These two devices are connected via a serial connection from pins 2 and 3 on the Arduino Uno board and pins 2 and 3 on the breakout board that was purchased from SparkFun Electronics[1]. Creating a serial connection using the Arduino interface was relatively simple with the NewSoftSerial library. Since this module is a quad-band module, meaning it can operate on any GSM frequency worldwide, the device is required to be set to operate on the US frequencies. This is achieved by uploading the following code to the Arduino and using serial band rates of 9600.
#include <NewSoftSerial.h>  //Used for string manipulations
#include <string.h>

char incoming_char=0;      //Will hold the incoming character from the Serial Port.
NewSoftSerial cell(2,3);   //Create a 'fake' serial port. Pin 2 is the Rx pin, pin 3 is the Tx pin.

void setup()
{
  Serial.begin(9600);
  cell.begin(9600);
}

void loop()
{
  if(cell.available() >0)
  {
    incoming_char=cell.read();   //Get the character from the cellular serial port.
    Serial.print(incoming_char); //Print the incoming character to the terminal.
  }
  if(Serial.available() >0)
  {
    incoming_char=Serial.read(); //Get the character coming from the terminal
    cell.print(incoming_char);   //Send the character to the cellular module.
    Serial.print(incoming_char); // ECHO to terminal
  }
}

Figure 5 – Arduino GSM interface code

Using this interface, the system was able to be programmed to only receive text message data and drop all cell phone requests, or any other external form of communication, over the GSM network. The entire list of commands can be found using the Telnit SM5100B datasheet, found at [2].

When properly supplied with a 5V power source and a serial terminal on the computer, the GSM Module will communicate specific commands signifying connection to the network.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+SIND: 1</td>
<td>SIM Card Inserted</td>
</tr>
<tr>
<td>+SIND: 10</td>
<td>Status of the phonebook</td>
</tr>
<tr>
<td>+SIND: 11</td>
<td>Successful connection to network</td>
</tr>
<tr>
<td>+SIND: 3</td>
<td>AT Module partially ready</td>
</tr>
<tr>
<td>+SIND: 4</td>
<td>AT Module ready</td>
</tr>
</tbody>
</table>

Figure 6 – Sample GSM commands
During this phase of implementation, an implementation problem arose caused by an insufficient power source. By using a sufficient power source that could supply up to 2 Amps of current during the registration phase the problem was corrected. It was discovered after researching first time connection methods, where the module will send high powered signals to search for cell phone towers in the local area. If insufficient power was supplied, the GSM Module would respond with a "+SIND:8" signal implying no reception was available or insignificant power was available to secure a reliable connection.

Now that the system was connected to the network, it required a method to receive text messages and decipher those text messages. To do this, the module was configured to parse all serial data, and do specific things upon receiving specific lines of data from the serial port.

```c
if(inString == "+SIND:8")
{
    Serial1.println("AT+CNMI=1,1,0,0,0"); // set SMS mode to text
    delay(200);
    Serial1.println("AT+CNMI=3,3,0,0"); // set module to send SMS data to serial out upon receipt
    delay(200);
}

if(inString == "+SINd:0")
{
    digitalWrite(led1, HIGH);
    digitalWrite(led2, LOW);
    led1_on = 1;
    led2_on = 0;
    onNetwork = 0;
}

if(inString == "+SINDull")
{
    //lcd.print("Connected! S:11");
    digitalWrite(led2, HIGH);
    digitalWrite(led1, LOW);
    led1_on = 0;
    led2_on = 1;
    onNetwork = 1;
}

if(inString == "PING")
{
    Serial1.println("ATNP");
}
```

**Figure 7 – Module configuration code**

**Implementing a clock and LCD**

Once the system could reliably receive turn on signals, a second condition for the device was created to check the Internet monitoring system and battery voltage testing system every 15 minutes. The first peripheral added to the system was a real time clock. By implementing a two-wire I2C interface with a DS1307 timing module, the device can count in seconds without...
being interrupted by other subroutines on the system. With this module connected, the device will run a checking routine that will convert voltages from an analog value to a digital 10 bit value, and if those voltages are above threshold the system will continue to be powered. This routine will also read a digital signal from the Internet monitoring system. As long as the digital value is high, the inverter will continue to be powered.

![Digital Clock and LCD Schematic](image)

**Figure 8 – Digital Clock and LCD Schematic**

A monochrome LCD was also installed using 4 of the digital pins on the Arduino. This allowed the team to debug the system by outputting data received through SMS and also other important information such as battery voltage levels, status of the Internet monitoring system, and what control ports were active for the Ethernet relay system. Using a potentiometer, members of the team are able completely power off the system when information is not necessary, to reduce power usage. These two additional systems were relatively easy to implement but provided significant help when debugging our design. Also, a significant advantage to using these devices is that they can all be powered through the 5V line on the micro-controller so that the only power needed to be supplied is a 12V to 5V DC-DC buck.

![LCD attached to Arduino](image)

**Figure 9 – LCD attached to Arduino**
Using this GSM Module and microcontroller, the requirements of wireless communication, timing methods, and communication between two external monitoring systems (Internet and battery voltage) were satisfied.

**Ethernet Monitor**

The current system in Tanzania is setup so that the satellite turns on and off after a set timeframe. In order to create a system that would truly give internet-on-demand, it was unanimously agreed that having a timed system would not work. Therefore, it was decided that the system would need to monitor the internet activity of the network in order to keep the satellite on when there were users using the internet and turn it off when user internet activity was not detected. In order to realize this, the use of the Arduino Uno and Arduino Ethernet Shield was needed.

![Arduino Uno Ethernet Monitor](image)

**Figure 10 – Arduino Uno Ethernet Monitor**

The group first approached the process of monitoring the internet in terms of monitoring packets sent through the router. When the internet is in use, the computer sends data to the network and also receives data from the network in the form of packets. Packets are the main sign of internet activity because they show data being sent from the computer to the network and also show the computer attempting to retrieve data from the network. As such, it was believed that monitoring these packets in a process called “Packet Sniffing” would be the best route for the Ethernet monitor. Many weeks of researching how to implement such a method revealed many complicated aspects of the approach. It was seen that even when the
internet was not in use, the computer would still send and receive packets. This meant that even if the computer was not using the internet, there would still be a stream of packets. As such, monitoring if the packet stream was stopped, indicating that the internet was not being used by any computer on the network would not work. The group attempted to work around this issue by adding a filter for the packet sniffer in order to only view packets that indicated internet activity. The problem that arose from this approach was now it was necessary to know what packets indicated internet activity. This approach was abandoned because the programming need to filter the packets was far too complicated for the simple job we needed it for. After discovering the complexity and also the potential privacy issues surrounding packet sniffing, the group quickly attempted to think of another approach to monitoring the internet.

The next approach was a far more simple solution. It simply relied on a single router that already had a built in activity monitor. The router had the ability to detect when there was no Ethernet activity by any of the devices connected to it. In response, it would power down in order to save power. Though this router was probably the most ideal and simple solution to monitoring the internet, it was seen as a risky approach. In the scenario that the router did not turn off even when there was no Ethernet activity from the computers on the network, it would be difficult to locate the problem and fix it compared to a system that was built from scratch. The cost of the router itself was over a hundred dollars and it still was not guaranteed to work. Therefore, the group continued to research other solutions.
The last approach proposed was also a simple yet feasible solution. It dealt with having the Arduino Uno connect to the router that connects to the satellite and ping the computers on the network to see if they were still in use. When a computer replies, it meant the computer was on and not in sleep mode. If none of the computers reply, it meant they were all in sleep mode or turned off. In this case it would signal another Arduino Uno connected to it to turn off the satellite. The problem with this approach was that it did not monitor the internet activity, but monitored the status of the computer. It also required the computers to be turned off or be put into sleep mode in order to turn off the satellite. The approach revolved around the idea that, “If the computer itself is not in use, neither is the internet”. This approach made the computer itself monitor its own activity by setting the power settings to go into sleep mode after a certain amount of time. After it is in sleep mode, the computer loses the ability to reply to ping requests, same as if it were shut down. Since both programming the Arduino Uno to ping each computer on the network and making sure each computer went into sleep mode after a certain time were seen as simple tasks, it was decided that this approach would be the most feasible.

The Arduino Uno Ethernet monitor would first get a signal from another Arduino Uno signaling that an internet request was made. This would cause the Ethernet monitor to cycle
through all the IPs of all the computers on the network and ping each on. Since an internet request was just made, it is most likely that it will get a reply from the computer of the school who made the request. After the Ethernet monitor sees this reply, it will go back to the first IP in the list and repeat the process of pining each IP. The Ethernet monitor does not check the rest of the IPs after it sees a reply because it is only necessary for one computer to reply to keep the satellite on.

![Ethernet monitor Receiving Ping Reply](image)

**Figure 12 – Ethernet monitor Receiving Ping Reply**
After the computer is put to sleep or shut off, it will not reply and the Ethernet monitor will proceed to ping the rest of the IPs. If none of them reply, then it will signal the other Arduino Uno to turn off the satellite. Pin 3 of the Arduino Uno is set high to signal this situation.

In order to prevent harm to the Ethernet monitor due to rats and the environment in rural Tanzania, the Ethernet monitor was put into a plastic casing. Upon arrival in Tanzania, the code will be updated appropriately to match the network. The IPs of each computer on the network will be programmed into the Ethernet shield. The program can be viewed in appendix III.

Figure 13 – Ethernet monitor with no Ping Replies
Figure 14 – Ethernet Monitor with Casing

**Power-over-Ethernet Control**

Making sure the system used as little power as possible was a top concern in every part of the design of the project. Each school is connected to the internet by a WiMAX antenna that connects to the router. WiMAX is a standard for wireless broadband internet that is known for its high data rate transmission. Each WiMAX antenna uses 60W of power when it is turned on.

In the current system, when one school wants to use the internet every antenna is turned on in the router. This means that every time any school would request the internet, 240 W would be used. The solution to this problem is to only turn on the antenna for the school that is requesting to use the internet. That way, if only one school wanted to use the internet it would only use 60W.

The Power-over-Ethernet (PoE) system created for the project acts as a switching station. When a school requests the internet the signal would be sent to the PoE system. The switch in the system would close, allowing the signal to pass through to the antenna and turn it on.
The Clare LCA710 power relay was used to create the switch. The power relay begins as a switch in the open position. When it receives the 5V signal from the Arduino, the switch closes and supplies power to the WiMAX antenna.

While there are several methods available, a power relay was thought to be the easiest way to make a switch for the system. The Clare LCA710 was chosen after an extensive search on the internet because of its simplicity, cost and ability to handle the necessary power specifications. It can handle up to 60 V and 1.8 A, which should be more than enough to meet the needs of the project.

Power is routed to the antenna through a registered jack 45 (RJ45) breakout. The RJ45 is an Ethernet connector that will aid in the transfer of the signal from the relay to the antenna. One will be placed on either side of the power relay in order to transmit the signal.

![Diagram](image)

**Figure 15 – Layout for the PoE system**

With the Power-over-Ethernet system in place, the overall power draw on the system is significantly reduced. It also creates a more efficient method for activating the internet for each school.
**Power Supply**

Due to the power constrictions of this project, designing an efficient power supply was important. This power supply was used to power the microcontroller at the home base using deep cycle solar batteries. Through experimentation, the Arduino with GSM shield were found to draw 500mA with a peak draw of 1.5A on the rare occasions when the shield is first trying to establish connection to the network. Due to relatively low power demands, a LM117 voltage regulator was used.

<table>
<thead>
<tr>
<th></th>
<th>Typical (A)</th>
<th>Max (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino/GSM Shield</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>LM117</td>
<td>1.5</td>
<td>2.2</td>
</tr>
</tbody>
</table>

*Figure 15 – Current draws for the Arduino and Voltage Regulator*

![Schematic for the Power Supply](image)

\[ V_{OUT} \approx 1.25V \left(1 + \frac{R_2}{R_1}\right) \]

*Figure 16 – Schematic for the Power Supply*

To verify the circuit worked correctly, it was simulated using PSPICE. This simulation can be found in Appendix 3.

Due to Tanzania’s proximity to the equator, heat dissipation concerns were present when designing the power supply. The high end of the typical current draw from the Arduino and shield combination is 500mA. During periods where a battery may be charging, a 12V deep cycle battery can have a voltage in the range of 14V. Using these stipulations as worse case
scenarios, the power dissipated by the LM117 was calculated. Additionally, assuming that a worse case ambient temperature of 32°C can be obtained, the equivalent thermal resistance of the LM117 was calculated. The TO-220 package was used for the calculations below.

\[ P_D = (V_{IN} - V_{OUT}) \times I_L = (14 - 5) = 4.5 \text{ W} \]

\[ T_R = T_J - T_A \]

\[ T_J = 150^\circ C \text{ for the TO-220 package} \]

\[ T_R = 150 - 32 = 118^\circ C \]

Thermal Resistance: \[ \frac{P_D}{T_R} = \frac{118}{4.5} = 26.2^\circ C/W \]

The TO-220 has a thermal resistance due to the junction to case connection of is 4°C/W.

\[ \theta_{JA} \leq 26.2 - 4 = 22.2^\circ C/W \]

The heat sink chosen could easily meet this requirement with a thermal resistance of 13.4°C/W.

The chosen chassis was an aluminum 1.65”×2.63”× 3” enclosure. Aluminum was chosen to allow for proper grounding of the enclosure. To ensure proper air flow, some ventilation holes were drilled in the top of the enclosure. The following pattern was found to allow for sufficient movement of air.
The rough placement of the holes for the input and output voltages and the mounting bolts are displayed below.

Figure 17 – Enclosure diagram

Figure 18 – Enclosure Diagram
**Battery Monitoring Circuit (Switchable Voltage Divider):**

The battery monitoring circuits were made with two parts: a voltage divider circuit and a MOSFET switch circuit. The voltage divider was made of several resistors aligned in series. All of the resistors were above 100 kΩ so that the current passed through the circuit well below 1 mA.

The MOSFET switch circuit consisted of one P-channel MOSFET and one N-channel MOSFET. The MOSFETs worked together as a logic circuit. The N-channel MOSFET would control the P-channel MOSFET by an output voltage from the Arduino microcontroller.

![Battery Monitor Diagram](image)

M1 is N-channel MOSFET and M2 is P-Channel MOSFET

V_{lg}(Vsw) is from Arduino Microcontroller and controls the logic of the circuit.

R2:R3 = 2:1

**Figure 19 – Battery Monitor Diagram**

In order to avoid any damage to the battery, the battery monitor circuits needed to be inserted into the system. The GSM shield and Arduino Microcontroller package are the core part of the system, so the functions of the Arduino were largely considered when designing the battery monitor. First, the Arduino would read an analog signal (voltage) from external circuit. Second, the Arduino would send an analog or digital output (voltage) itself. Third, the Arduino
input and output both have voltage limits from 0 Volts to 5 Volts. Based on these findings, the voltage divider was decided to be used in the design.

Next, the team aimed to minimize the power consumption and avoid any possible influence to the whole solar power system. The controllable switch circuit was determined to be the best solution. A switch based on a Bipolar Junction Transistor (BJT) was first used and tested, but the redundant qualifications made this circuit impossible to switch between the cut-off and forward-active modes. Because of this, the MOSFET switch circuit was designed and tested. It met the specifications and was used in the battery monitor circuit.

The switch circuit was controlled by a logic input from the Arduino. The circuit could switch between ‘open’ and ‘short’. When the N-channel MOSFET gained a ‘HIGH’ at its logic control terminal, it worked as a short circuit and made the logic control terminal P-channel MOSFET ‘LOW’. After this, the P-channel MOSFET worked as a short circuit as well and allowed the voltage being viewed to go through the switch circuit to the voltage divider.

When the N-channel MOSFET gained ‘LOW’, it worked as an open circuit. At the moment R1 in the schematic allowed node 2 to be ‘HIGH’. This set the P-channel MOSFET as an open circuit, too.

Because the Arduino controlled the logic terminal of the switch circuit, as well as read the voltage output, the Arduino program was needed to imbed battery monitoring circuits in the system. The program would normally set ‘LOW’ to the output pin which connected to the logic gate of the N-channel MOSFET. For every 15 minutes, the program would set ‘HIGH’ to the output pin and allow the circuit to work for a few seconds. This would prevent the system from wasting energy.

As mentioned previously, BJT were tested but were not able to perform both cut-off and forward-bias modes in the system. The forward-active of the NPN BJT needed voltage at the collector greater than at the base and greater than at the emitter. While, the cut-off of NPN BJUT demanded voltage at base less than at emitter and at collector. There were three terminals and three different voltage values where only one was controllable and had a range
from 0 volt to 5 volts (Vcontr). The other two voltage values were 12 volts (Vs) and 0 volts (Vgnd). Wherever Vcontr was put and at whatever its value was set, it was obvious that those three voltage values could not achieve both forward-active and cut-off of NPN BJT. The PNP BJT was tried as well, but the same results were obtained. Therefore, BJTs were considered infeasible for the battery system.

Because the four batteries which need to be monitored are connected in series, it was decided to monitor each battery cell at the beginning which means that four switchable voltage divider circuits would be connected in parallel to their corresponding battery cells. However, those four batteries only have one ground terminal with the rest accumulated on the bottom ones. The same ratio of voltage divider would no longer work properly because the highest the input voltage could reach would be 70 volts. A different ratio of voltage divider was then considered.

In order to minimize costs and elements, four switch circuits share one resistor divider. Although it worked properly during testing, there was risk using this method because it was possible to make one of those switch circuits’ output voltage greater than the input voltage. This resulted in a potentially huge reverse current flow through the intrinsic diode in the P-channel MOSFET. Therefore, four individual switchable voltage divider circuits with different dividing ratio were finally determined to be used for the system.
Chapter 4

Testing the GSM Network

The first thing required to start testing the communication system was to obtain SIM cards for the GSM shield. These are essentially access codes that allow devices to connect to the GSM network via service providers. Dr. Kurt DeMaagd assisted the team in obtaining two SIM cards that allowed access to the T-Mobile network. After obtaining these SIM cards, connecting to the network and sending/receiving text messages became a number one priority. By implementing a monochrome LCD screen, debugging our code became relatively easy. As shown below, SIND:8 received from the GSM chip signifies that the device was dropped from the network, and SIND:11 signifies the device successfully registered to the network.

Figure – Connected GSM devices

The first test relied on sending a text message from one of the devices to a team member’s cell phone. On each remote device, the source code was successfully uploaded. With a momentary switch the device reliably connected to the GSM network and sent text messages to an external cell phone. This process became the turn-on signal. These devices, as shown below, will be connected to the computers at each one of the remote schools. They will be powered through USB and with the press of a single button will send a text message to the server, and relay that the school wants access to the internet.

The second stage of testing consisted of running a GSM module and configuring it to receive text messages and using it to decode the messages to see if a correct turn on signal had
been received. To achieve this, the device had been configured to serially distribute all text message data to the arduino. From that point the device was configured to turn on a LED when it received a specific message. This signified that the device could understand when it received a passphrase and would not just turn on if any random text message was sent to it. The LCD screen was also configured to show the status of the device, and the last text message that it received regardless of if it was a correct passphrase. In addition to testing receipt of text messages, the device was configured to automatically drop cell phone calls when a cell phone request was identified. This was achieved by using the serial message “ATH”.

Finally, testing of the turn on system and real time clock was performed. Text messages were sent to the GSM device and if appropriate passphrases were received, a tone was played out through a speaker that was connected to the device. A RGB LED also attached to the device turned blue, signifying internet access was requested and the device was now outputting a 5V digital signal, which will be attached to the external switch on the power inverter located at the satellite system. A separate tone was configured for six different passphrases, signifying the six schools that are included in the WiMAX network. In addition to the sound that was played and the LED, a digital output pin was set high which is attached to an Ethernet relay in the device that Ken designed. At the receipt of a turn-on passphrase, a fifteen minute timer started which could be viewed on the LCD screen. After fifteen minutes of time, a routine would be initiated that read the digital input pin from the Ethernet monitoring system Ivan designed, and the analog pins from the battery monitor system Koy designed. If either of these two reads failed or were below threshold, all output pins were set to zero which also was displayed on the LCD screen.
Testing the Power Supply

The objective of this power supply design was to power an Arduino using a 12 volt deep cycle solar battery. The chosen LM117 circuitry allowed for the flexibility of exactly dialing in a 5 volt supply voltage. The simulation below shows how, with careful adjustment of the feedback potentiometer, an output voltage of 5 volts can be achieved. The PSPICE code can be found in Appendix 3.

Figure – Connected GSM device
Though the simulation verified the functionality of the circuit, a prototype was made and it too operated as expected.

The final version of the circuit had to perform under conditions different from the simulation and the prototype. It had to not only maintain a 5 volt output voltage for an extended period of time, but it also had to be able to dissipate the heat that this operation generated. Below is a screen capture of a scope displaying the expected voltage output from the test.
To test the heat dissipation ability of the circuit, it was operated continuously for several hours. After the test, the circuit was checked for any appreciable heat accumulation and none was found. Though exact temperatures were not recorded for this test, not much information would be gathered by doing this. The only point of concern was that the circuit could operate for extended periods of time without overheating.

**Testing the Battery Monitoring Circuits**

With one switchable divider circuit, two power supplies were used. One simulated the battery and the other simulated the Arduino. A 12 Volt input was set for the battery and 5 V represented a ‘HIGH’ from the Arduino, while ground was ‘LOW’.

**Figure** - *Scope output of the Power Supply*
The results read from the multimeter are shown in the table below.

<table>
<thead>
<tr>
<th>With 12 Vin (Power Supply)*</th>
<th>Switch Circuit State</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>SHORT</td>
</tr>
<tr>
<td>RM1</td>
<td>&gt;100 Mega Ohms</td>
</tr>
<tr>
<td>RM2</td>
<td>&gt;100 Mega Ohms</td>
</tr>
<tr>
<td>V(4)</td>
<td>16 milli Volts</td>
</tr>
<tr>
<td>V(5)</td>
<td>5-6 milli volts</td>
</tr>
</tbody>
</table>

*Power Supply in the lab might not output exactly 12 V because accuracy wasn’t measured

From the table shown in the figure it was clear that the circuit worked properly.
The next test was for multiple switch divider circuits. Because car batteries were not available in the lab, power supplies were used to simulate the situation. Two power supplies were connected in series. The first power supply’s (Vp1) negative terminal was connected to its ground and its positive terminal connected to the second power supply’s (Vp2) negative terminal.

There was another power supply (Vcontr) which simulated the Arduino. This power supply’s negative terminal connected to Vp1’s ground terminal so that all those power supplies share a common ground.

The steps to test the circuit are as follows: A) Connect Vp1’s positive terminal to the first switch circuit’s Vin and Vp2’s positive terminal to the second switch circuit’s Vin. B) Connect Vp1’s negative to the switchable divider circuits’ ground. C) Connect Vsw pins of both switchable divider circuits to ground (‘LOW’). Once this was complete, the voltage at node 5 was checked with the multimeter. (State 0)

The first switchable divider circuit’s Vsw was changed to Vcontr with a ‘HIGH’ input. The voltage at node 5 was measured with the multimeter again. (State 1)

Similarly, the voltage at node 5 was tested once more when the first switchable divider circuit’s Vsw was connected back to ground (‘LOW’) and the second switchable divider circuit’s Vsw was changed to Vcontr with a ‘HIGH’ input. (State 2)

At state 0, the output was several millivolts, so it worked correctly. At state 1, because the results were pretty linear and Vp2 had almost no effect on the output, it worked correctly again. At state 2, both Vp1 and Vp2 influence the output, proving the circuit worked properly.

**Testing the Ethernet Monitor**

The Ethernet monitor was first tested in an already setup network with many objects already connected to its wireless network. The Ethernet monitor was programmed with an IP that matched the network and that did not conflict with any of the other IPs on the network. The test was conducted with a wireless printer and a laptop connected to the wireless network.
To determine if the Ethernet monitor was performing as expected, it was connected to the router via Ethernet cable and powered up via USB to the desktop computer also on the network. Since the desktop computer was needed to monitor the output of the Ethernet monitor, the desktop’s IP was left out of the list of IPs to ping programmed into the Ethernet monitor.

![Initial Testing of Ethernet Monitor in Dorm](image)

*Figure – Initial Testing of Ethernet Monitor in Dorm*

With the Ethernet monitor now able to ping IPs on the network, including the two objects connected to the network, it was tested to see if it pinged successfully or not. At first it would turn on a red LED to signal that it got a reply from an object that was active on the network. It would then repeat the cycle of going through all the IPs again. By turning off both the printer and putting the laptop to sleep mode, the Ethernet monitor would cycle through all the IPs and then see that none of them reply, thus turning off the red LED. If either printer or the laptop were to become active again, the red LED would turn on. In the end of the initial testing, the Ethernet monitor was able to ping objects on the network, recognize that they were active or on, and recognize when all the objects in the network were inactive.

The final test on the complete Ethernet monitor was done using an old router given to use for testing purposes. The router was used to simulate the wireless network the computers in Tanzania connect to. A laptop was connected to the wireless network of the router. After determining the laptop’s IP, the IP was programmed into the Ethernet monitor. The Ethernet
monitor was then attached to the router and allowed to ping address on the network it was programmed to ping. It was able to ping the laptop a see that it was still active. The LED would stay blue until the laptop was put into sleep mode, in which case it would turn red. The laptop was then put into sleep mode in order to test this functionality. After a short amount of time where the Ethernet shield pinged all the IPs it was programmed to, the LED turned red. This showed that the Ethernet shield was functioning properly and could be used on other networks with little problems.

**Figure** – *Final Testing of Ethernet Monitor; Active Laptop on Network*
Figure – Final Testing of Ethernet Monitor; Laptop in Sleep mode
Chapter 5

Conclusion

During the period of this project, the team successfully created a wireless turn on system that will be installed in Africa. The main objective—to develop a system that can remotely turn on a satellite system, has successfully been completed. Using the SMS standard for sending text messages, devices that will be placed at remote schools, can individually signal request for internet. A timer has been developed so that every fifteen minutes the main system will run two routines to check both internet connectivity using ping signals, and battery voltage levels using analog-to-digital conversion.

In addition to technical specifications, this project required additional requirements for heat, stability, and safety to be used in elementary through high schools. To counter this, heat sinks, fuses to our power lines, and fitted enclosures were added to each individual component. We also tested our main device for a period of two days to ensure that it would run without interruption. During this time, text messages were sent to the device at random times using team member’s cell phones. These text messages were sent serially through USB and stored on a laptop that was left running in the ECE 480 lab. Upon re-arrival in the lab, the device was still connected to the GSM network and all text messages were successfully received to the device.

Furthermore, implementation of an additional device, not included in the specifications was achieved. Using power relays and Ethernet breakout boards, the device is able to control power through PoE lines that power each individual WiMAX antenna. Using this, power consumption is reduced significantly by cutting 60W of power, per antenna.

The total budget from the sponsor was $1,500 and an additional $500 was supplied from the Department of Electrical and Computer Engineering. Of this budget the project was fulfilled using only approximately $1500. This included 7 devices to install in schools and a main device that will be installed at the satellite system. Adding additional schools to the system
would require an additional $150 which includes a GSM Module, microcontroller, antenna, and cables for assembly.

Overall the team was able to satisfy all of the sponsor’s requirements except for one thing. The project’s sponsor had requested that internet access be signaled upon a user opening an internet browser. After testing this, internet activity monitoring was achieved using the TinyProxy method with which previous teams had been successful. However, this solution only works on Linux systems and, during the study abroad trip this summer, Windows operating systems will be installed on all new computers located at the schools. In response to this different available proxy methods were researched, however open-source proxy servers written in Windows source-code are extremely difficult to find. Passive network monitoring was also tested using Ethernet communication line taps, but this turned out to cause packet failures. Another alternative method would to use an active network tap, one that is commercially produced and connected to a power supply. This would allow monitoring of every packet sent and received on a network and allow each school to request internet access without the push of a button. The downside to this solution is that network taps run in the several hundreds of dollars, and installing one in each school would be very costly. Although this specification was not met, implementation of an alternate method was achieved. This method was achieved through using ping requests on a microcontroller connected to the main router. Computers will be put into sleep mode every 15 minutes that a user is inactive. In sleep mode computers will deactivate their network interface cards (NIC) and ping requests to those NIC’s will fail. Using a method that continuously pings computers we are able to find out which computers are still active and can control the network reliably.
Appendix I

Qilong Gou

My technical role for the project was to design a solution that could measure the voltage across the solar power batteries. Considering the functions and limits of Arduino which is the core part of our project, Arduino could read analog signal (voltage) from external circuit, could send analog or digital output (voltage) itself, and have limits of input and output voltage from 0 Volts to 5 Volts. Because the batteries are 12 Volts Car batteries, it was essential to design a circuit which could proportional lower the output voltage in the range of Arduino’s limits. Hence, I decided to use voltage divider to achieve this goal.

Because the batteries are DC voltage source, resistors divider is selected rather than transformer. After selecting this, I aimed at minimizing the power consumption and avoiding possible influence to the whole solar power system. So, a switch circuit seemed to be necessary. MOSFET switch was the final decision, because of its logic gate property.

Additionally, I combined voltage divider circuits and MOSFET switch circuits. I tested it first with one 12 Volts power supply. The circuit worked properly and was able to cut off current flow through the voltage divider part while applying ‘LOW’ to the logic control terminal. However, when I tried to test the circuits with two power supplies connected in series to simulate the practical situation, I figured out that there was only on ground pin through those batteries package. Hence, different dividing ratio was selected.

Finally, I soldered all the elements on a PC board with adding a fuse between ground and the circuit to prevent the whole system from potential damage. I soldered all the pins to a ten pins connector so that we could easily connect eternal wires to the boards.
Drew Newell

My main technical role on the team was to design a voltage regulator that we could use to continuously power an Arduino microcontroller. The robustness, safeness and effectiveness of the circuit were all primary concerns for the team. The regulator I designed is responsible for powering the main Arduino that turns on and off the internet up-link system. Therefore, it was of the utmost importance that the regulator operated reliably.

Several challenges had to be addressed when designing this circuit. The first challenge was designing an enclosure. The final design of the module fit in a 1.65”×2.63”×3” box. A small size was important because the module had to fit in a small cabinet that contained other pieces of equipment. I chose aluminum as the enclosure material for a couple of reasons. Because animals in the area have the tendency to chew wires, it was important to choose a durable material in which this will not be a problem. The second reason that aluminum was chosen was to allow for proper grounding within the unit and prevent any kind of short circuit damage to the battery. I also put several strategically placed ventilation holes in the enclosure to increase air flow. I wanted to allow for some degree of air movement but I also wanted the enclosure secure enough so that fingers could not accidently reach into the circuitry. Because of the tropical climate of Tanzania, I expected a year round ambient temperature of 32 degrees Celsius. To keep the LM117 regulator cool, I used a deluxe heat sink that could dissipate the heat in this environment.

Though we will be using only one of these modules for our design day presentation, several others will be made for the project in Tanzania. Beyond the design of the power regulator, I occasionally assisted with the other electrical engineering aspects of the project. I mostly made suggestions and bounced ideas off of the team members responsible for the design of the battery voltage monitor and the power-over-ethernet relay system.
My main technical role for the project was to develop a solution to monitoring the internet activity on the network. This was a crucial part of our system. Even if our system granted the schools of Tanzania internet on demand, if it would still be hampered by having a timing system programmed into it to shut down the satellite. This would make it so that the users would have to make another internet request after the satellite shuts down. As such, our group saw it reasonable to monitor internet activity in order to determine if the satellite should be shut off or not.

During my research on how to accomplish this task, I looked into what network activity looked like during activity. This led me to research into packet sniffing. Since packets were basically the data being sent to computer from the network and to the network from the computer, I found it to be an excellent starting point as to monitoring the activity of the network. I soon found out that attempting to monitor the stream of packets and determining if it meant that the internet was being used or not was a complicated task. Not only was the programming aspect complicated but the concept of packet sniffing led to many privacy questions.

I then looked into simpler solutions. This led me to find the D-Link green routers. These routers had the built-in ability to determine if there was any Ethernet activity occurring on any of the devices it was connected to. If it did not detect any activity, it would automatically shut down. This router had everything I was looking for in an Ethernet monitor but I questioned its reliability. There was no real guarantee that it work if we bought it.

I finally thought of another solution that involved pinging all the computers on the network. By doing so it would be possible to see if a computer was on, in sleep mode or off. This is technically not monitoring internet activity but monitoring computer status. My thought was that if the computer is in sleep mode or off, no one is using the internet. When all the computers were off or in sleep mode, then it would be fine to turn off the satellite. I was able
to find a program that allowed the Arduino to ping computers using an Arduino Ethernet shield. It was successfully tested in my own dorm using the laptop and printer connected on my network to simulate the computers connected to the network in Tanzania.

Jeff Pawloski

My main technical contribution to this project was interfacing and programming the GSM module and microcontrollers to communicate signals between schools and the satellite control station. I also interfaced a timing device with our controlling device and developed the smaller turn on devices that would be installed at the remote schools. Additionally, I programmed routines on the control device that would read the digital output from the Internet monitor and use analog-to-digital conversion to test battery voltages. I also interfaced a liquid crystal display with the device so that outputs from these routines could be easily identifiable. This allowed significantly easier testing of our system.

When designing this system, I learned invaluable knowledge of serial communication between devices, and how wireless devices are connected to wireless networks. I found that when sending serial data in between modules, you have to account for processor speeds and variable baud rates. At the beginning of programming our microcontroller would operate faster than the GSM module could serially send bytes. This caused some problems when searching for specific strings of bytes as the controller would think that multiple different serial communications were taking place in quick repetition. To fix this a 20 millisecond delay was added to the routine after every byte was recorded on the microcontroller. This allowed the serial buffer to hold information immediately before retrieving it. I also researched connection methods to the GSM network and found that GSM devices can have a very large current overshoot when in areas of low signal such as in the Engineering building. In some cases the module can draw up to 2A (although outside of the engineering building, max current draw never surpassed 150mA) of current, while only having an operating current of 40~50 mA.
I also helped research different methods that could be used in monitoring Internet activity, and helped decide that the most reliable method to monitor internet connectivity was monitoring active computer connections through ping signals. This was decided because a reliable Windows open-source proxy server was impossible to find.

In addition to these technical contributions, I also soldered the main GSM device, and all six of the additional GSM devices to be installed at the remote schools.

Kenneth Jordan

My main technical role on this project was to develop a method to turn the WiMAX antennae on the wireless router for each school individually. This would allow each school to access the internet while reducing the amount of power that the router used while turned on.

Collaborating with my teammates, I determined that the best method to turn of antennae individually would be to create a switching system that could direct the proper signal to the correct antenna. After doing some research, a Power-over-Ethernet system was shown to be a very cost effective and efficient way to deliver the signals as required.

The components for the Power-over-Ethernet system were found by determining the power needs for the system and researching potential power relays that would work for the design. The Clare LCA710 was the best fit for the purposes of the design. The registered jack 45 (RJ45) was a necessary component for transferring the signal from the PoE system to the WiMAX antenna.

I was also responsible for developing the push-button mechanism for the Arduino device that was placed in the schools. We began with a rocker switch because we thought it might be a better method for turning the system on and off. After discussing it with the
sponsor, we determined that the use of momentary switches was a better method for signaling from the schools to the satellite system.

Secondary technical roles included helping to test and develop the battery monitor with Koy, soldering my portion of the project, assisting the others with their devices, and adding my thoughts to the overall project.
Appendix II

Appendix III

Power Supply PSPICE code

LM117 ARDUINO POWER SUPPLY
VBAT 1 0 12
C1 1 0 0.1U
X1 1 2 3 LM117HV
R1 2 3 270
R2 3 0 810
RL 2 0 10
C2 2 0 1U
.SUBCKT LM117HV 1 2 3
* IN OUT ADJ
RB1 7 6 1
RB2 6 5 124
RBC 15 5 247
RC 1 14 0.742
RI 2 4 100MEG
RP 9 8 100
RZ 8 10 .104
CBC 13 15 2.479N
CPZ 10 2 0.796U
DBK 14 13 DLM117HV
DFB 6 12 DLM117HV
DSC 6 11 DLM117HV
EB 7 2 8 2 1
EFB 12 2 POLY(2) (13,5) (6,5) 3.52 -135M 0 12.1M -107.9M
EP 9 2 4 2 265.5
ESC 11 2 POLY(2) (13,5) (6,5) 2.45 0 0 0 -107.9M
IADJ 1 4 48.4U
QP 13 5 2 QLM117HV
VREF 4 3 1.2782
.MODEL QLM117HV NPN (IS=30F BF=50 VAF=9.27 NF=1.604)
.MODEL DLM117HV D (IS=30F N=1.604)
.ENDS LM117HV
.TRAN .1U 16U 0 .01U
.PROBE
.END
Ethernet Monitor Code

`#include <SPI.h>
#include <Ethernet.h>
#include <ICMPPing.h>`

/*
 * Internet Monitor
 * The Arduino Uno will ping all given IPs to check if the
 * Computer with the given IP is off or in sleep mode
 * The computer will reply if not in such a state
 * Blue - Intent Request made and there has been a single reply
 * Red - All computers did not reply - Can turn off satellite
 * => Digital Port 3 for HIGH signal
 */

// mac address for Ethernet shield
// Not needed
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
// ip address for ethernet shield
byte ip[] = {192,168,0,196};
// Start IP
byte pingAddr[] = {192,168,0,101};

// DO NOT CHANGE!
SOCKET pingSocket = 3;

// Stores text of Ping reply
char buffer [256];

void setup()
{
    // start Ethernet
    Ethernet.begin(mac, ip);
    Serial.begin(9600);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    digitalWrite(5, HIGH);
    digitalWrite(4, LOW);
    digitalWrite(3, LOW);
}

void loop()
{
    ICMPPing ping(pingSocket);
    ping(4, pingAddr, buffer); // Ping address

    // Used for debugging purposes
    int x = pingAddr[3]; // Get last part of IP address
    Serial.println(x); // Prints last part of IP address
    Serial.println(buffer); // Prints Ping reply

    pingAddr[3]++; // Go to next address
// If the request timed out
// Full reply: Request Timed Out
if (buffer[2] == 'q')
{
    Serial.println("Timed Out\n");
}
// Get a reply so we can keep the blue LED on
// Also we can recycle through the IPs again
else
{
    Serial.println("Reply");
    digitalWrite(5, HIGH);
    digitalWrite(3, LOW);
    Serial.println("Reset\n");
    Serial.println("Reset\n");
    pingAddr[3] = 101;
}
delay(500);

// If it goes through all the computers
// Then that means no computers are currently active
// Turn on Red LED
if (pingAddr[3] > 105)
{
    Serial.println("Reset\n");
    pingAddr[3] = 101;
    digitalWrite(3, HIGH);
    digitalWrite(5, LOW);
}
}